UNIVERSITÀ DELLA CALABRIA

UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Dottorato di Ricerca in

Information and Communication Engineering for Pervasive Intelligent Environments

CICLO

XXIX

SCALABLE DATA ANALYSIS: METHODS, TOOLS AND APPLICATIONS

Settore Scientifico Disciplinare ING-INF/05

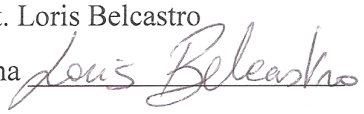
Coordinatore: Ch.mo Prof. Felice Crupi

Firma 

Supervisore/Tutor: Ch.mo Prof. Domenico Talia

Firma 

Dottorando: Dott. Loris Belcastro

Firma 

Loris Belcastro

Scalable Data Analysis: methods, tools and applications

Supervisor: Domenico Talia

July 3, 2017

To my family

Preface

In the last years the ability to produce and gather data has increased exponentially. Every day huge amounts of data are collected from several sources, such as social networks, sensors, mobile devices. To extract helpful knowledge from such big data, novel technologies, architectures, and algorithms have been developed by data scientists for capturing and analyzing complex and/or high velocity data.

The goal of this thesis is studying, designing and exploiting models, technologies, tools and systems for Big Data analysis, especially on Clouds, to support scalable distributed knowledge discovery applications. The work is organized in two main parts. The first part focuses on methods and tools for supporting scalable execution of distributed knowledge discovery applications and, in general, solutions for dealing with Big Data issues. The second part presents data analysis applications and methodologies for extracting knowledge from large datasets.

As result of the first research activity, we integrated the MapReduce model into the workflow formalism provided by the Data Mining Cloud Framework (DMCF), a systems developed at the University of Calabria for creating and executing scalable data analysis application on Clouds. By implementing a DMCF data mining application whose workflow includes MapReduce computations, we were able to achieve a nearly linear speedup, thanks to the combined scalability provided by the DMCF workflows languages and by the MapReduce framework.

The second research activity led to the design and implementation of *Geocon*, an open-source, scalable, and service-oriented middleware designed to help developers to implement context-aware mobile applications. Geocon provides a service and a client library for storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments).

As result of the third research activity, we present *G-RoI*, a novel data mining technique that exploits the indications contained in geotagged social media

items to discover the boundaries of a Place-of-Interest (PoI), commonly called Region-of-Interest (RoI). To assess the quality of the proposed technique, an experimental evaluation was carried out on a set of PoIs located in the center of Rome and Paris, using a large set of geotagged photos published in Flickr over six years. The experimental results show that G-RoI is able to detect more accurate RoIs than existing techniques, regardless of shapes, areas and densities of PoIs, and without being influenced by the proximity of different PoIs.

Finally, we designed a predictor of the arrival delay of scheduled flights due to weather conditions, as several studies have shown that weather is one of the primary causes of flight delays. Accurate prediction of flight delays is an important problem to be addressed given the economic impact of flight delays to both airlines and travelers. In our model, the predicted arrival delay takes into consideration both flight information (origin airport, destination airport, scheduled departure and arrival time) and weather conditions at origin airport and destination airport according to the flight timetable. The results show a high accuracy in prediction of delays and a good scalability, which means the proposed solution identifies a very useful pattern of flight delay that may help airlines in reducing delays.

Rende, Cosenza, Italy *Loris Belcastro*

February 2017

Prefazione

Negli ultimi anni la capacità di generare e collezionare dati è aumentata in maniera esponenziale. Ogni giorno, infatti, grandi moli di dati vengono collezionati da diverse sorgenti, tra cui social network, sensori, dispositivi mobili. Questi grandi quantitativi di dati, spesso riferiti col termine "Big Data", presentano una serie di caratteristiche, quali la complessità e la velocità di generazione, che li rendono estremamente difficili da gestire. Per tali ragioni, nuove tecnologie, architetture ed algoritmi sono stati sviluppati per analizzare grandi moli di dati ed estrarre da questi conoscenza utile

Questo lavoro di tesi ha come obiettivo lo studio e l'impiego di modelli, tecnologie e sistemi per l'analisi di Big Data, che possono essere utilizzate, soprattutto in ambiente Cloud, per sviluppare applicazioni scalabili e distribuite per la scoperta di conoscenza. Il lavoro è organizzato in due parti principali. La prima parte si focalizza su metodi e strumenti per l'esecuzione scalabile di applicazioni distribuite per l'estrazione di conoscenza e, in generale, di soluzioni per far fronte alle problematiche connesse ai Big Data. Nella seconda parte, invece, vengono presentate alcune applicazioni di data analysis e metodologie per l'estrazione di conoscenza da grandi volumi di dati.

La prima attività di ricerca ha avuto come risultato l'integrazione del modello MapReduce all'interno del formalismo dei workflow messo a disposizione da Data Mining Cloud Framework (DMCF), un sistema, sviluppato all'Università della Calabria, per creare ed eseguire applicazioni scalabili per l'analisi di dati sul Cloud. Integrando operazioni MapReduce all'interno dei workflow, in particolare, si è dimostrato come sia possibile costruire applicazioni di data mining scalabili, con valori di speedup prossimi a quelli ideali.

La seconda attività di ricerca ha portato alla progettazione e all'implementazione di *Geocon*, un middleware orientato ai servizi, open-source e scalabile, progettato per facilitare lo sviluppo di applicazioni mobili context-aware. Geocon mette a disposizione una libreria di servizio ed una libreria client per la memorizzazione, indicizzazione ed estrazione di informazioni relative ad entità che sono comunemente utilizzate in questi scenari, come utenti, luoghi, eventi ed altre risorse (es. foto, media, commenti).

Come risultato della terza attività di ricerca, presentiamo *G-RoI*, una nuova tecnica di data mining che sfrutta le indicazioni contenute nel dati geolocalizzati provenienti dai social network per estrarre regioni di interesse (RoI), intese come i confini geografici che contraddistinguono l'area associata ad un punto di interesse (PoI). Per validare la qualità della tecnica proposta, è stata eseguita una valutazione sperimentale su un insieme di PoI nel centro di Roma e Parigi, utilizzando un dataset di grandi dimensioni costituito da foto pubblicate su Flickr in un periodo di sei anni. I risultati sperimentali ottenuti hanno dimostrato come G-RoI sia in grado di rilevare RoI più accurate rispetto alle tecniche esistenti, indipendentemente dalle caratteristiche di forma, dimensione, densità e prossimità dei PoI considerati.

Infine, è stato progettato un predittore del ritardo di arrivo dei voli dovuto alle condizioni meteorologiche. Diversi studi, infatti, hanno dimostrato che il meteo è una delle cause principali del ritardo aereo. La predizione accurata dei ritardi aerei è un problema molto importante, con un grande impatto economico sia per le compagnie aeree che per i passeggeri. Nel nostro modello, il ritardo predetto prende in considerazioni le informazioni del volo (es. aeroporti di origine e di destinazione, orari previsti per la partenza e l'arrivo) e le condizioni meteorologiche agli aeroporti di origine e di destinazione in accordo agli orari di volo programmati. I risultati hanno mostrato un'elevata accuratezza nella predizione dei ritardi e una buona scalabilità, evidenziando come la soluzione proposta sia in grado di fornire informazioni utili alle compagnie aeree per la riduzione dei ritardi.

Rende, Cosenza, Italy *Loris Belcastro*

February 2017

Acknowledgements

During the time of writing of this PhD thesis I received support and help from many people.

Firstly, I would like to express my sincere gratitude to my supervisor, Prof. Domenico Talia, for his generous support, motivation, and great knowledge, who helped me in each step to complete my PhD course and related research.

Besides my supervisor, I would like to thank Prof. Paolo Trunfio and Dr. Fabrizio Marozzo, for their help, insightful comments, and stimulating discussions. I must say that it was a pleasure to work with them.

A special thanks to my colleagues, friends, for their indispensable support that was essential to reach this goal.

And finally, I should like to conclude by expressing my deep gratitude to my family, in particular to my wife Francesca, for their love, support and essential encouragement.

Contents

List of Figures

List of Tables

1

Introduction

In the last years the ability to produce and gather data has increased exponentially. In fact, in the Internet of Things' era, huge amounts of digital data are generated by and collected from several sources, such as sensors, cams, in-vehicle infotainment, smart meters, mobile devices, web applications and services. The huge amount of data generated, the speed at which it is produced, and its heterogeneity in terms of format (e.g., video, text, xml, email), represent a challenge to the current storage, process and analysis capabilities. In particular, thanks to the growth of social networks (e.g., Facebook, Twitter, Pinterest, Instagram, Foursquare, etc.), the widespread diffusion of mobile phones, and the large use of location-based services, every day millions of people access social network services and share information about their interests and activities. Those data volumes, commonly referred as Big Data, can be exploited to extract useful information and to produce helpful knowledge for science, industry, public services and in general for humankind.

Although nowadays the term Big Data is often misused, it is very important in computer science for understanding business and human activities. In fact, Big Data is not only characterized by the large size of datasets, but also by the complexity, by the variety, and by the velocity of data that can be collected and processed. So, we can collect huge amounts of digital data from sources, at a very high rate that the volume of data is overwhelming our ability to make use of it. This situation is commonly called "data deluge".

In science and business, people are analyzing data to extract information and knowledge useful for making new discoveries or for supporting decision processes. This can be done by exploiting Big Data analytics techniques and tools. As an example, one of the leading trends today is the analysis of big geotagged data for creating spatio-temporal sequences or trajectories tracing user movements. Such kind of information is clearly highly valuable for science and business: tourism agencies and municipalities can know the most visited places by tourists, the time of year when such places are visited, and other useful information [13][81]; transport operators can know the places and routes where is it more likely to serve passengers[154] or crowed areas where more

transportation resources need to be allocated[152]; city managers may exploit social media analysis to reveal mobility insights in cities such as incident locations[82], or to study and prevent crime events [84][57].

But it must be also considered that Twitter and Facebook produce more then 20 TB of data every day. Then to extract value from such kind of data, novel technologies and architectures have been developed by data scientists for capturing and analyzing complex and/or high velocity data. In this scenario data mining raised in the last decades as a research and technology field that provides several different techniques and algorithms for the automatic analysis of large datasets. The usage of sequential data mining algorithms for analyzing large volumes of data requires a very long time for extracting useful models and patterns. For this reason, high performance computers, such as many and multi-core systems, Clouds, and multi-clusters, paired with parallel and distributed algorithms are commonly used by data analysts to tackle Big Data issues and get valuable information and knowledge in a reasonable time.

In this context, Cloud computing is probably the most valid and cost-effective solution for supporting Big Data storage and for executing sophisticated data analytic applications. In fact, thanks to elastic resources allocation and high computing power, Cloud computing represents a compelling solution for Big Data analytics, allowing faster data analysis, that means more timely results and then greater data value.

The goal of this thesis is studying, designing and exploiting models, technologies, tools, and systems for Big Data analysis, especially on Clouds, to support scalable distributed knowledge discovery applications. The work is organized in two main parts. The first part focuses on methods and tools for supporting scalable execution of distributed knowledge discovery applications and, in general, solutions for dealing with Big Data issues. The second part presents data analysis applications and methodologies for extracting knowledge from large datasets.

In the first part of this thesis, two research approaches are presented: the first one is the use of the MapReduce programming model for processing large datasets on Clouds and, in particular, the use of the MapReduce paradigm in combination with the workflow paradigm to enable scalable data processing or to implement knowledge discovery applications on Clouds; the second one is instead the design and implementation of a scalable middleware for context-aware applications, which can be deployed on Clouds.

As a result of the first research approach, we present an extension for integrating the MapReduce model into the workflow engine provided by Data Mining Cloud Framework (DMCF), a framework for supporting the scalable execution of distributed knowledge discovery applications. More in detail, we describe how workflows, created using VL4Cloud or JS4Cloud (i.e., the formalisms provided by DMCF for designing workflows), can include MapReduce algorithms and tools, and how these workflows are executed in parallel on DMCF to enable scalable data processing on Clouds.

The second research approach led to the design of Geocon, an open-source service-oriented middleware designed to help developers to implement context-aware mobile applications. Geocon provides a service and a client library for storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments). To deal with very the large number of users and resources involved in such kind of applications, Geocon is designed to scale horizontally on a multiple nodes. Despite not being directly involved in the data analysis process, Geocon represents a solution for improving the capabilities of collecting, storing, indexing, and querying huge amount of data produced by mobile context-aware applications. Data collected and stored exploiting Geocon can be subsequently fed to a data analysis application for extracting valuable information.

With regards to data analysis applications proposed in the second part of this thesis work, we focused on two research topics: the first one is the use of knowledge discovery methodologies for extracting highly valuable information exploiting data coming from social networks; the second one is instead the use of parallel and distributed machine learning techniques, coupled with the MapReduce programming model, for extracting high valuable information from large datasets.

As a result of this research topic, we present *G-RoI*, a novel mining technique that exploits indications contained in geotagged social media items to discover the boundaries of a Place-of-Interest (PoI), also called Region-of-Interest (RoI). We experimentally evaluated the accuracy of G-RoI in detecting the RoIs associated to a set of PoIs in Rome and Paris, using a large set of geotagged photos published in Flickr over six years.

The result of the second research work is a predictor of the arrival delay of a scheduled flight due to weather conditions. This work was motivated by the fact that several studies have shown that weather is one of the primary causes of flight delays, with a significant economical impact (i.e., the cost of flight delays for US economy was estimated to be more than 30 billion in 2007). The data preparation and mining tasks have been implemented as MapReduce programs that have been executed on a Cloud infrastructure to achieve scalability. The results show a high accuracy in prediction of delays and a good scalability, which means the proposed methodology identifies a very useful pattern of flight delay that may help airlines in reducing delays.

1.1 Publications

The following publications have been produced while accomplishing this thesis.

1.1.1 Journals

- L. Belcastro, F. Marozzo, D. Talia, P. Trunfio, "*Using Scalable Data Mining for Predicting Flight Delays*". ACM Transactions on Intelligent Systems and Technology (ACM TIST), vol. 8, n. 1, October 2016.
- L. Belcastro, F. Marozzo, D. Talia, P. Trunfio, "*G-RoI: Automatic Region-of-Interest detection driven by geotagged social media data*". ACM Transactions on Knowledge Discovery from Data (ACM TKDD). Under review, 2016.
- L. Belcastro, F. Marozzo, D. Talia, P. Trunfio, "*A Scalable Middleware for Context-aware Mobile Applications*". Concurrency and Computation: Practice and Experience. Under review, 2017.

1.1.2 Book Chapters

- L. Belcastro, F. Marozzo, D. Talia, P. Trunfio, "*Big Data Analysis on Clouds*". In: Handbook of Big Data Technologies, S. Sakr, A. Zomaya (Editors), Springer, 2017.

1.1.3 Papers in refereed conference proceedings

- L. Belcastro, G. Di Lieto, M. Lackovic, F. Marozzo, P. Trunfio, "*Geocon: A Middleware for Location-aware Ubiquitous Applications*". Proc. of the First International Workshop on Ultrascale Computing for Early Researchers (UCER 2016), 2016.
- L. Belcastro, F. Marozzo, D. Talia, P. Trunfio, "*Programming Visual and Script-based Big Data Analytics Workflows on Clouds*". Post-Proc. of the High Performance Computing Workshop 2014, Cetraro, Italy, Advances in Parallel Computing, vol. 26, pp. 18–31, IOS Press, 2015.

1.2 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 discusses models, technologies and research trends in Big Data analysis on Clouds.

Then next two chapters discuss solutions that can be used for developing respectively data analysis and mobile context-aware applications, also in presence of Big Data. In particular, Chapter 3 presents the Data Mining Cloud Framework (DMCF) and discusses how its workflow paradigm has been integrated with the MapReduce model, while Chapter 4 discusses the middleware Geocon for developing mobile context-aware applications.

Next two chapters present data mining applications that made use of big datasets for extracting useful information. Specifically, Chapter 5 presents the *G-RoI* mining technique for extracting RoIs from geotagged social media data gathered from social networks, while Chapter 6 presents a predictor

of the arrival delay of a scheduled flight due to weather conditions. Finally, Chapter 7 discusses conclusions and future work.

2

Big Data Analysis on Clouds: models, tools and solutions

Every day huge amounts of data are collected from several sources, such as social networks (e.g., Facebook, Twitter, Pinterest, Instagram, Foursquare, etc.), sensors, mobile devices. Such data volumes, commonly referred as Big Data, can be exploited to extract useful information and to produce helpful knowledge for science, industry, public services and in general for humankind.

However, the process of knowledge discovery from Big Data is not so easy, mainly due to data characteristics, as size, complexity and variety that require to address several issues. In this scenario data mining raised in the last decades as a research and technology field that provides several different techniques and algorithms for the automatic analysis of large datasets.

Cloud computing is a valid and cost-effective solution for supporting Big Data storage and for executing sophisticated data mining applications. Big Data analytics is a continuously growing field, where novel and efficient solutions (i.e., in terms of platforms, programming tools, frameworks, and data mining algorithms) spring up everyday to cope with the growing scope of interest in Big Data.

This chapter discusses models, technologies and research trends in Big Data analysis on Clouds. In particular, the chapter presents representative examples of Cloud environments that can be used to implement applications and frameworks for data analysis, and an overview of the leading software tools and technologies that are used for developing scalable data analysis on Clouds.

The remainder of the chapter is organized as follows. Section 2.1 provides a brief discussion about existing Big Data definitions. Section 2.2 provides a brief introduction to Big Data Analytics. Section 2.3 introduces the main Cloud computing concepts. Section 2.4 describes representative examples of Cloud environments that can be used to implement applications and frameworks for data analysis in the Cloud. Section 2.5 provides an overview of the leading software tools and technologies used for developing scalable data analysis on Clouds. Section 2.6 discusses some research trends and open challenges on Big Data analysis. Finally, Section 2.7 concludes the chapter.

2.1 Big Data definitions

Nowadays the term Big Data is often misused, but it is very important in computer science for understanding business and human activities. Several definitions for Big Data have been proposed in literature, but reaching a global consensus about what it means is not easy.

Although not explicitly mentioning the term "Big Data", the first definition was proposed by Doug Laney (an analyst of the META Group, now Gartner) in a 2001 report, in which a three-dimensional model, also known as the "3Vs" (*Volume*, *Velocity*, and *Variety*) model, is used to describe Big Data. Many years after, this model continues to be one of the most used, considering that much of the industry and the research (e.g., IBM [69] and Microsoft researchers [104]) still adopt the Big Data definition provided by Gartner [56]:"*Big Data is high volume, high velocity, and/or high variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation*".

According to the above definition, Big Data is not only characterized by the large size of datasets, but also by the variety (i.e., data from multiple repositories, domains, or types), and by the velocity of data that can be collected and processed. In fact, we can collect huge amounts of digital data from several sources, at a very high rate that the volume of data is overwhelming our ability to make use of it. This situation is commonly called "data deluge".

Many other definitions have been proposed in literature that extended the "3Vs" model by introducing other features, like "Value" [71][54], "Veracity" [122], "Complexity" [4], etc.

About that, a 2011 report from IDC [54] provided a "4Vs" definition for Big Data: "*Big data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery, and/or analysis*". This definition delineates a new feature of Big Data, i.e., value, that refers to the capability to create lot of value for organizations, societies and consumers from data analyses.

On the contrary, in 2012 Gartner [14] and IBM [68] extended the "3Vs" model by introducing the Veracity as the fourth "V". Veracity includes questions about quality, reliability, and uncertainty of captured data and the outcome of analysis of that data.

Finally, the definition of Big Data provided by NIST [42] introduced a new feature of Big Data, i.e., the variability: "*Big Data consists of extensive datasets – primarily in the characteristics of volume, variety, velocity, and/or variability – that require a scalable architecture for efficient storage, manipulation, and analysis*". Variability is defined as the changes in other data characteristics that produce variance in data meaning, in lexicon, with a potential huge impact on data homogenization.

From the discussion above, it is clear that finding a common definition of Big Data is very difficult. In this regard, a brief review on Big Data definitions have been conducted by De Mauro, et al. [34] that aims to define a consensual definition of Big Data.

2.2 Introduction to Big Data Analytics

Data Analytics plays a crucial role in today's business by helping companies to improve operational efficiency and gain competitive advantages over business rivals. According to a very popular definition[94]: "*Data Analytics is the science of examining raw data with the purpose of drawing conclusions about that information*".

Since 1950s, organizations were using basic data analytics techniques to discover useful information about market trends or insights. In most cases, such kind of analysis was essentially provided to manually examine numbers in spreadsheets with the aim of extracting information that could be used for future decisions. But today, to give organizations a competitive edge, data analytics must identify insights for immediate decisions. Today, most organizations collect all the data – often really huge amounts – that streams into their businesses for extracting meaningful and valuable information for business (e.g., for making better decisions or planning marketing strategies) and science (e.g., verify or disprove models or theories) purposes.

Big Data Analytics refers to advanced data analytics techniques applied to Big Data sets. These techniques include data mining, statistics, data visualization, artificial intelligence, machine learning, natural language processing, etc. The usage of Big Data Analytics produces several benefits, especially in big companies, in terms of cost reduction for storing and querying large amount of data, effectiveness of decision making, ability to provides services that better meet customers' needs. Some Big Data Analytics application fields are discussed in the following:

- *Text Analytics*: it is the process of deriving information from text sources (e.g., document, social networks, blogs, web sites) [55], which can be used for sentiment analysis, content classification, text summarization, etc.
- *Predictive analytics*: it is the process of predicting future events or behaviors by exploiting models developed using a variety of statistical, machine learning, data mining, and other analytical techniques [110].
- *In-Memory analytics*: it is the process that loads data to be analyzed directly into systems random access memory (RAM), instead of storing data on physical disks. In such way, in-memory analytics approach greatly reduces queries and calculation times, allowing business intelligence applications to support faster business decisions [95].
- *Graph analytics (or Network Analysis)*: it analyzes the behavior of various connected components through the use of network and graph theories,

which is useful for investigating structures in social networks [111]. Examples of graph analytics path, connectivity, community, and centrality analysis.

- *Prescriptive Analytics*: it is a form of advanced analytics which examines data or content to optimize decisions about what actions to take in order to maximize profit, given a set of constraints and key objectives[119]. It is characterized by techniques such as graph analysis, simulation, complex event processing, neural networks, recommendation engines, heuristics, and machine learning.

Despite the great benefits discussed above, dealing with big data is not a bed of roses. In fact, the process of knowledge discovery from Big Data is generally complex, mainly due to data characteristics, as size, complexity and variety, that require to address several issues. Moreover, Big Data analytics is a continuously growing field where so novel and efficient solutions (i.e., in terms of platforms, programming tools, frameworks, and data mining algorithms) spring up everyday to cope with the growing scope of interest in Big Data. In this context, Cloud computing is a valid and cost-effective solution for supporting Big Data storage and for executing sophisticated data analytics applications.

2.3 Cloud computing

This section introduces the basic concepts of Cloud computing, which provides scalable storage and processing services that can be used for extracting knowledge from Big Data repositories. In the following we provide basic Cloud computing definitions (Section 2.3.1) and discuss the main service distribution and deployment models provided by Cloud vendors (Section 2.3.2).

2.3.1 Basic concepts

In the last years, Clouds have emerged as effective computing platforms to face the challenge of extracting knowledge from Big Data repositories in limited time, as well as to provide effective and efficient data analysis environments to both researchers and companies. From a client perspective, the Cloud is an abstraction for remote, infinitely scalable provisioning of computation and storage resources. From an implementation point of view, Cloud systems are based on large sets of computing resources, located somewhere "in the Cloud", which are allocated to applications on demand [10]. Thus, Cloud computing can be defined as a distributed computing paradigm in which all the resources, dynamically scalable and often virtualized, are provided as services over the Internet. As defined by NIST (National Institute of Standards and Technology) [105] Cloud computing can be described as: "*A model for enabling convenient, on-demand network access to a shared pool of configurable computing*

resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

From the NIST definition, we can identify five essential characteristics of Cloud computing systems, which are on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. Cloud systems can be classified on the basis of their service model and their deployment model.

2.3.2 Cloud service distribution and deployment models

Cloud computing vendors provide their services according to three main distribution models:

- *Software as a Service (SaaS)*, in which software and data are provided through Internet to customers as ready-to-use services. Specifically, software and associated data are hosted by providers, and customers access them without need to use any additional hardware or software. Examples of SaaS services are Gmail, Facebook, Twitter, Microsoft Office 365.
- *Platform as a Service (PaaS)*, in an environment including databases, application servers, development environment for building, testing and running custom applications. Developers can just focus on deploying of applications since Cloud providers are in charge of maintenance and optimization of the environment and underlying infrastructure. Examples of PaaS services areWindows Azure, Force.com, Google App Engine.
- *Infrastructure as a Service (IaaS)*, that is an outsourcing model under which customers rent resources like CPUs, disks, or more complex resources like virtualized servers or operating systems to support their operations (e.g., Amazon EC2, RackSpace Cloud). Compared to the PaaS approach, the IaaS model has a higher system administration costs for the user; on the other hand, IaaS allows a full customization of the execution environment.

The most common models for providing Big Data analytics solution on Clouds are PaaS and SaaS. IaaS is usually not used for high-level data analytics applications but mainly to handle the storage and computing needs of data analysis processes. In fact, IaaS is the more expensive delivery model, because it requires a greater investment of IT resources. On the contrary, PaaS is widely used for Big Data analytics, because it provides data analysts with tools, programming suites, environments, and libraries ready to be built, deployed and run on the Cloud platform. With the PaaS model users do not need to care about configuring and scaling the infrastructure (e.g., a distributed and scalable Hadoop system), because the Cloud vendor will do that for them. Finally, the SaaS model is used to offer complete Big Data analytics applications to end users, so that they can execute analysis on large and/or

complex datasets by exploiting Cloud scalability in storing and processing data.

Regarding deployment models, Cloud computing services are delivered according to three main forms:

- *Public Cloud*: it provides services to the general public through the Internet and users have little or no control over the underlying technology infrastructure. Vendors manage their proprietary data centers delivering services built on top of them.
- *Private Cloud*: it provides services deployed over a company intranet or in a private data center. Often, small and medium-sized IT companies prefer this deployment model as it offers advance security and data control solutions that are not available in the public Cloud model.
- *Hybrid Cloud*: it is the composition of two or more (private or public) Clouds that remain different entities but are linked together.

As outlined in [85], users access Cloud computing services using different client devices and interact with Cloud-based services using a Web browser or desktop/mobile app. The business software and users data are executed and stored on servers hosted in Cloud data centers that provide storage and computing resources. Resources include thousands of servers and storage devices connected each other through an intra-Cloud network. The transfer of data between data center and users takes place on wide-area network. Several technologies and standards are used by the different components of the architecture. For example, users can interact with Cloud services through SOAP-based or RESTful Web services [118] and Ajax technologies allow Web interfaces to Cloud services to have look and interactivity equivalent to those of desktop applications. Open Cloud Computing Interface (OCCI)[1] specifies how Cloud providers can deliver their compute, data, and network resources through a standardized interface.

2.4 Cloud solutions for Big Data

At the beginning of the Big Data phenomenon, only big IT companies, such as Facebook, Yahoo!, Twitter, Amazon, LinkedIn, invested large amounts of resources in the development of proprietary or open source projects to cope with Big Data analysis problems. But today, Big Data analysis becomes highly significant and useful for small and medium-sized businesses. To address this increasing demand a large vendor community started offering highly distributed platforms for Big Data analysis. Among open-source projects, Apache Hadoop is the leading open-source data-processing platform, which was contributed by IT giants such as Facebook and Yahoo.

Since 2008, several companies, such as Cloudera, MapR, and Hortonworks, started offering enterprise platform for Hadoop, with greats efforts to improve

[1] OCCI Working Group, http://www.occi-wg.org

Hadoop performances in terms of high-scalable storage and data processing. Instead, IBM and Pivotal started offering its own customized Hadoop distribution. Other big companies decided to provide only additional softwares and support for Hadoop platform developed by external providers: for example, Microsoft decided to base its offer on Hortonworks platform, while Oracle decided to resell Cloudera platform. However Hadoop is not the only solution for Big Data analytics. Out of the Hadoop box other solutions are emerging. In particular, in-memory analysis has become a widespread trend, so that companies started offering tools and services for faster in-memory analysis, such as SAP, that is considered the leading company with its Hana[2] platform. Other vendors, including HP, Teradata and Actian, developed analytical database tools with in-memory analysis capabilities. Moreover, some vendors, like Microsoft, IBM, Oracle, and SAP, stand out from their peers for offering a complete solution for data analysis, including DBMS systems, software for data integration, stream-processing, business intelligence, in-memory processing, and Hadoop platform.

In addition, many vendors decided to focus whole offer on the Cloud. Among these certainly there are Amazon Web Services (AWS) and 1010data. In particular, AWS provides a wide range of services and products on the Cloud for Big Data analysis, including scalable database systems and solutions for decision support. Other smaller vendors, including Actian, InfiniDB, HP Vertica, Infobright, and Kognitio, focused their big-data offer on database management systems for analytics only. Following the approach in [133], the remainder of the section introduces representative examples of Cloud environments: Microsoft Azure as an example of public PaaS, Amazon Web Services as the most popular public IaaS, OpenNebula and OpenStack as examples of private IaaS. These environments can be used to implement applications and frameworks for data analysis in the Cloud.

2.4.1 Microsoft Azure

Azure[3] is the Microsoft Cloud proposal. It is environment providing a large set of Cloud services that can be used by developers to create Cloud-oriented applications, or to enhance existing applications with Cloud-based capabilities. The platform provides on-demand compute and storage resources exploiting the computational and storage power of the Microsoft data centers. Azure is designed for supporting high availability and dynamic scaling services that match user needs with a pay-per-use pricing model. The Azure platform can be used to perform the storage of large datasets, execute large volumes of batch computations, and develop SaaS applications targeted towards end-users. Microsoft Azure includes three basic components/services:

[2] https://hana.sap.com
[3] https://azure.microsoft.com

- *Compute* is the computational environment to execute Cloud applications. Each application is structured into roles: *Web role*, for Web-based applications; *Worker role*, for batch applications; *Virtuam Machines role*, for virtual-machine images.
- *Storage* provides scalable storage to manage: binary and text data (*Blobs*), non-relational tables (*Tables*), queues for asynchronous communication between components (*Queues*). In addition, for relational databases, Microsoft provides its own scalable Cloud database services, called Azure SQL Database.
- *Fabric controller* whose aim is to build a network of interconnected nodes from the physical machines of a single data center. The Compute and Storage services are built on top of this component.

Microsoft Azure provides standard interfaces that allow developers to interact with its services. Moreover, developers can use IDEs like Microsoft Visual Studio and Eclipse to easily design and publish Azure applications.

2.4.2 Amazon Web Services

Amazon offers compute and storage resources of its IT infrastructure to developers in the form of Web services. Amazon Web Services (AWS)[4] is a large set of Cloud services that can be composed by users to build their SaaS applications or integrate traditional software with Cloud capabilities. It is simple to interact with these service since Amazon provides SDKs for the main programming languages and platforms (e.g. Java, .Net, PHP, Android).

AWS compute solution includes *Elastic Compute Cloud* (EC2), for creating and running virtual servers, and *Amazon Elastic MapReduce* for building and executing MapReduce applications. The Amazon storage solution is based on *S3 Storage Service*, with a range of storage classes designed to cope with different use cases (i.e., Standard, Infrequent Access, and Glacier for long term storage archive). A full set of database systems are also proposed: *Relational Database Service (RDS)* for relational tables; *DynamoDB* for non-relational tables; *SimpleDB* for managing small datasets; *ElasticCache* for caching data. Even though Amazon is best known to be the first IaaS provider (based on its EC2 and S3 services), it is now also a PaaS provider, with services like Elastic Beanstalk, that allows users to quickly create, deploy, and manage applications using a large set of AWS services, or Amazon Machine Learning, that provides visualization tools and wizards for easily creating machine learning models.

2.4.3 OpenNebula

OpenNebula [125] is an open-source framework mainly used to build private and hybrid Clouds. The main component of the OpenNebula architecture is

[4] https://aws.amazon.com

the Core, which creates and controls virtual machines by interconnecting them with a virtual network environment. Moreover, the Core interacts with specific storage, network and virtualization operations through pluggable components called Drivers. In this way, OpenNebula is independent from the underlying infrastructure and offers a uniform management environment. The Core also supports the deployment of Services, which are a set of linked components (e.g., Web server, database) executed on several virtual machines. Another component is the Scheduler, which is responsible for allocating the virtual machines on the physical servers. To this end, the Scheduler interacts with the Core component through appropriate deployment commands.

OpenNebula can implement a hybrid Cloud using specific Cloud Drivers that allow to interact with external Clouds. In this way, the local infrastructure can be supplemented with computing and storage resources from public Clouds. Currently, OpenNebula includes drivers for using resources from Amazon EC2 and Eucalyptus [109], another open source Cloud framework.

2.4.4 OpenStack

OpenStack[5] is an open source Cloud operating system realesed under the terms od the Apache License 2.0. It allows the management of large pools of processing, storage, and networking resources in a datacenter through a Web-based interface. Most decisions about its development are decided by the community to the point that every six months there is a design summit to gather requirements and define new specifications for the upcoming release. The modular architecture of OpenStack is composed by four main components, as shown in Figure 2.1.

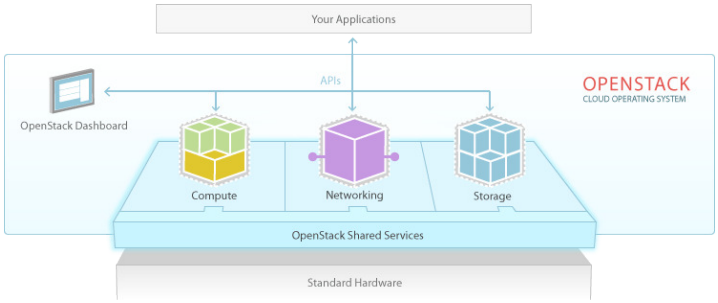


Fig. 2.1. OpenStack architecture (source: openstack.org).

OpenStack Compute provides virtual servers upon demand by managing the pool of processing resources available in the datacenter. It supports different virtualization technologies (e.g., VMware, KVM) and is designed to

[5] https://www.openstack.org/

scale horizontally. *OpenStack Storage* provides a scalable and redundant storage system. It supports Object Storage and Block Storage: the former allows storing and retrieving objects and files in the datacenter. *OpenStack Networking* manages the networks and IP addresses. Finally, *OpenStack Shared Services* are additional services provided to ease the use of the datacenter, such as Identity Service for mapping users and services, Image Service for managing server images, and Database Service for relational databases.

2.5 Systems for Big Data Analytics in the Cloud

In this section we describe the most used tools for developing scalable data analysis on Clouds, such as MapReduce, Spark, workflow systems, and NoSQL database management systems. In particular, we discuss some frameworks commonly used to develop scalable applications that analyze big amounts of data, such as Apache Hadoop, the best-known MapReduce implementation, and Spark. We present also some powerful data mining programming tools and strategies designed to be executed in the Cloud for exploiting complex and flexible software models, such as the distributed workflows. Workflows provide a declarative way of specifying the high-level logic of an application, hiding the low-level details. They are also able to integrate existing software modules, datasets, and services in complex compositions that implement discovery processes. In this section we presented several data mining workflow systems, such as the Data Mining Cloud Framework, Microsoft Azure Machine Learning, and ClowdFlows. Moreover, we discuss about NoSQL database technology that recently became popular as an alternative or as a complement to relational databases. In the last years, several NoSQL systems have been proposed for providing more scalability and higher performance than relational databases. We introduce the basic principles of NoSQL, described representative NoSQL systems, and outline interesting data analytics use cases where NoSQL tools are useful. Finally, we present a brief overview of well known visual analytics tools, that help users in analytical reasoning by interactive visual interfaces.

2.5.1 MapReduce

MapReduce is a programming model developed by Google [35] in 2004 for large-scale data processing to cope efficiently with the challenge of processing enormous amounts of data generated by Internet-based applications.

Since its introduction, MapReduce has proven to be applicable to a wide range of domains, including machine learning and data mining, social data analysis, financial analysis, scientific simulation, image retrieval and processing, blog crawling, machine translation, language modelling, and bioinformatics. Today, MapReduce is widely recognized as one of the most important programming models for Cloud computing environments, being it supported

by Google and other leading Cloud providers such as Amazon, with its Elastic MapReduce service[6], and Microsoft, with its HDInsight[7], or on top of private Cloud infrastructures such as OpenStack, with its Sahara service[8].

Hadoop[9] is the most used open source MapReduce implementation for developing parallel applications that analyze big amounts of data. It can be adopted for developing distributed and parallel applications using many programming languages (e.g., Java, Ruby, Python, C++). Hadoop relieves developers from having to deal with classical distributed computing issues, such as load balancing, fault tolerance, data locality, and network bandwidth saving.

The Hadoop project is not only about the MapReduce programming model (Hadoop MapReduce module), as it includes other modules such as:

- *Hadoop Distributed File System (HDFS)*: a distributed file system providing fault tolerance with automatic recovery, portability across heterogeneous commodity hardware and operating systems, high-throughput access and data reliability.
- *Hadoop YARN*: a framework for cluster resource management and job scheduling.
- *Hadoop Common*: common utilities that support the other Hadoop modules.

In particular, thanks to the introduction of YARN in 2013, Hadoop turns from a batch processing solution into a platform for running a large variety of data applications, such as streaming, in-memory, and graphs analysis. As a result, Hadoop became a reference for several other frameworks, such as: Giraph for graph analysis; Storm for streaming data analysis; Hive, which is a data warehouse software for querying and managing large datasets; Pig, which is as a dataflow language for exploring large datasets; Tez for executing complex directed-acyclic graph of data processing tasks; Oozie, which is a workflow scheduler system for managing Hadoop jobs. Besides Hadoop and its ecosystem, several other MapReduce implementations have been implemented within other systems, including GridGain, Skynet, MapSharp and Twister [45]. One of the most popular alternative to Hadoop is Disco, which is a lightweight, open-source framework for distributed computing. The Disco core is written in Erlang, a functional language designed for building fault-tolerant distributed applications. Disco has been used for a variety of purposes, such as log analysis, text indexing, probabilistic modeling and data mining.

[6] http://aws.amazon.com/elasticmapreduce/
[7] http://azure.microsoft.com/services/hdinsight/
[8] http://wiki.openstack.org/wiki/Sahara
[9] http://hadoop.apache.org/

2.5.2 Spark

Apache Spark[10] is another Apache framework for Big Data processing. Differently from Hadoop in which intermediate data are always stored in distributed file systems, Spark stores data in RAM memory and queries it repeatedly so as to obtain better performance for some class of applications (e.g., iterative machine learning algorithms) [148]. For many years, Hadoop has been considered the leading open source Big Data framework, but recently Spark has become the more popular so that it is supported by every major Hadoop vendors. In fact, for particular tasks, Spark is up to 100 times faster than Hadoop in memory and 10 times faster on disk. Several other libraries have been built on top of Spark: *Spark SQL* for dealing with SQL and DataFrames, *MLlib* for machine learning, *GraphX* for graphs and graph-parallel computation, and *Spark Streaming* to build scalable fault-tolerant streaming applications.

For these reasons, Spark is becoming the primary execution engine for data processing and, in general, a must-have for Big Data applications. But even though in some applications Spark can be considered a better alternative to Hadoop, in many other applications it has limitations that make it complementary to Hadoop. The main limitation of Spark is that it does not provide its own distributed and scalable storage system, that is a fundamental requirement for Big Data applications that use huge and continually increasing volume of data stored across a very large number of nodes. To overcome this lack, Spark has been designed to run on top of several data sources, such as Cloud object storage (e.g., Amazon S3 Storage, Swift Object Storage), distributed filesystem (e.g., HDFS), no-SQL databases (e.g., HBase, Apache Cassandra), and others. Today an increasing number of big vendors, such Microsoft Azure or Cloudera, offer Spark as well as Hadoop, so developers can choose the most suitable framework for each data analytics application.

With respect to Hadoop, Spark loads data from data sources and executes most of its tasks in RAM memory. In this way, Spark reduces significantly the time spent in writing and reading from hard drives, so that the execution is far faster than Hadoop. Regarding task recovering in case of failures, Hadoop flushes all of the data back to the storage after each operation. Similarly, Spark allow recovering in case of failures by arranging data in Resilient Distributed Datasets (RDD), which are a immutable and fault-tolerant collections of records which can be stored in the volatile memory or in a persistent storage (e.g., HDFS, HBase). Moreover, Spark's real-time processing capability is increasingly being used by Big Data analysts into applications that requires to extract insights quickly from data, such as recommendation and monitoring systems.

Several big companies and organizations use Spark for Big Data analysis purpose: for example, Ebay uses Spark for log transaction aggregation and analytics, Kelkoo for product recommendations, SK Telecom analyses mobile usage patterns of customers.

[10] http://spark.apache.org

2.5.3 Mahout

Apache Mahout[11] is an open-source framework that provides scalable implementations of machine learning algorithms that are applicable on big input. Originally, the Mahout project provided implementations of machine learning algorithms executable on the top of Apache Hadoop framework. But the comparison of the performance of Mahout algorithms on Hadoop with other machine learning libraries, showed that Hadoop spends the majority of the processing time to load the state from file system at every intermediate step [123].

For these reasons, the latest version of Mahout goes beyond Hadoop and provides several machine learning algorithms for collaborative filtering, classification, and clustering, implemented not only in Hadoop MapReduce, but also in Spark, H2O[12]. Both Apache Spark and H2O process data in memory so they can achieve a significant performance gain when compared to Hadoop framework for specific classes of applications (e.g., interactive jobs, real-time queries, and stream data) [123]. In addition, the latest release of Mahout introduces a new math environment, called Samsara [89], that helps users in creating their own math providing general linear algebra and statistical operations. In the following, some examples for each algorithm's category are listed: analyzing user history and preferences to suggest accurate recommendations (collaborative filtering), selecting whether a new input matches a previously observed pattern or not (classification), and grouping large number of things together into clusters that share some similarity (clustering) [112]. In the future, Mahout will support Apache Flink[13], an open source platform that provides data distribution, communication, and fault tolerance for distributed computations over data streams.

2.5.4 Hunk

Hunk[14] is a commercial data analysis platform developed by Splunk for rapidly exploring, analyzing and visualizing data in Hadoop and NoSQL data stores. Hunk uses a set of high-level user and programming interfaces to offer speed and simplicity of getting insights from large unstructured and structured datasets. One of the key components of the Hunk architecture is the *Splunk Virtual Index*. This system decouples the storage tier from the data access and analytics tiers, so enabling Hunk to route requests to different data stores. The analytics tier is based on *Splunks Search Processing Language* (SPL) designed for data exploration across large, different datasets. The Hunk web framework allows building applications on top of the Hadoop Distributed File System (HDFS) and/or the NoSQL data store.

---

[11] http://mahout.apache.org/
[12] http://www.h2o.ai
[13] https://flink.apache.org/
[14] http://www.splunk.com/en_us/products/hunk.html

Developers can use Hunk to build their Big Data applications on top of data in Hadoop using a set of well known languages and frameworks. Indeed, the framework enables developers to integrate data and functionality from Hunk into enterprise Big Data applications using a web framework, documented REST API and software development kits for CSharp,Java, JavaScript, PHP and Ruby. Also common development languages such as HTML5 and Python can be used by developers.

The Hunk framework can be deployed on on-premises Hadoop clusters or private Clouds and it is available as a preconfigured instance on the Amazon public Cloud using the Amazon Web Services (AWS). This public Cloud solution allows Hunk users to utilize the Hunk facilities and tools from AWS, also exploiting commodity storage on Amazon S3, according to a pay-per-use model. Finally, the framework implements and makes available a set of applications that enable the Hunk analytics platform to explore, explore and visualize data in NoSQL and other data stores, including Apache Accumulo, Apache Cassandra, MongoDB and Neo4j. Hunk is also provided in combination with the Cloudera's enterprise data hub to develop large-scale applications that can access and analyze Big Data sets.

2.5.5 Sector/Sphere

Sector/Sphere[15] is a Cloud framework designed at the University of Illinois-Chicago to implement data analysis applications involving large, geographically distributed datasets in which the data can be naturally processed in parallel [63]. The framework includes two components: a storage service called *Sector*, which manages the large distributed datasets with high reliability, high performance IO, and a uniform access, and a compute service called *Sphere*, which makes use of the Sector service to simplify data access, increase data IO bandwidth, and exploit wide area high performance networks. Both of them are available as open source software[16]. Sector is a distributed storage system that can be deployed over a wide area network and allows users to ingest and download large datasets from any location with a high-speed network connection to the system. The system can be deployed over a large number of commodity computers (called nodes), located either within a data center or across data centers, which are connected by high-speed networks.

In an example scenario, nodes in the same rack are connected by 1 Gbps networks, two racks in the same data center are connected by 10 Gbps networks, and two different data centers are connected by 10 Gbps networks. Sector assumes that the datasets it stores are divided into one or more separate files, called slices, which are replicated and distributed over the various nodes managed by Sector.

The Sector architecture includes a Security server, a Master server and a number of Slave nodes. The Security server maintains user accounts, file

[15] http://sector.sourceforge.net/
[16] http://sector.sourceforge.net

access information, and the list of authorized slave nodes. The Master server maintains the metadata of the files stored in the system, controls the running of the slave nodes, and responds to users' requests. The Slaves nodes store the files managed by the system and process the data upon the request of a Sector client. Sphere is a compute service built on top of Sector and provides a set of programming interfaces to write distributed data analysis applications. Sphere takes streams as inputs and produces streams as outputs. A stream consists of multiple data segments that are processed by Sphere Processing Engines (SPEs) using slave nodes. Usually there are many more segments than SPEs. Each SPE takes a segment from a stream as an input and produces a segment of a stream as output. These output segments can in turn be the input segments of another Sphere process. Developers can use the Sphere client APIs to initialize input streams, upload processing function libraries, start Sphere processes, and read the processing results.

2.5.6 BigML

BigML[17] is a system provided as a Software-as-a-Service (SaaS) for discovering predictive models from data and it uses data classification and regression algorithms. The distinctive feature of BigML is that predictive models are presented to users as interactive decision trees. The decision trees can be dynamically visualized and explored within the BigML interface, downloaded for local usage and/or integration with applications, services, and other data analysis tools. Recently, BigML launched its PaaS solution, called *BigML PredictServer*, which is a dedicated machine image that can be deployed on Amazon AWS. An example of BigML prediction model is shown in Figure 2.2.

Extracting and using predictive models in BigML consists in multiple steps, as detailed as follows:

- *Data source setting and dataset creation.* A data source is the raw data from which a user wants to extract a predictive model. Each data source instance is described by a set of columns, each one representing an instance feature, or field. One of the fields is considered as the feature to be predicted. A dataset is created as a structured version of a data source in which each field has been processed and serialized according to its type (numeric, categorical, etc.).
- *Model extraction and visualization.* Given a dataset, the system generates the number of predictive models specified by the user, who can also choose the level of parallelism level for the task. The interface provides a visual tree representation of each predictive model, allowing users to adjust the support and confidence values and to observe in real time how these values influence the model.
- *Prediction making.* A model can be used individually, or in a group (the so-called ensemble, composed of multiple models extracted from different

[17] https://bigml.com

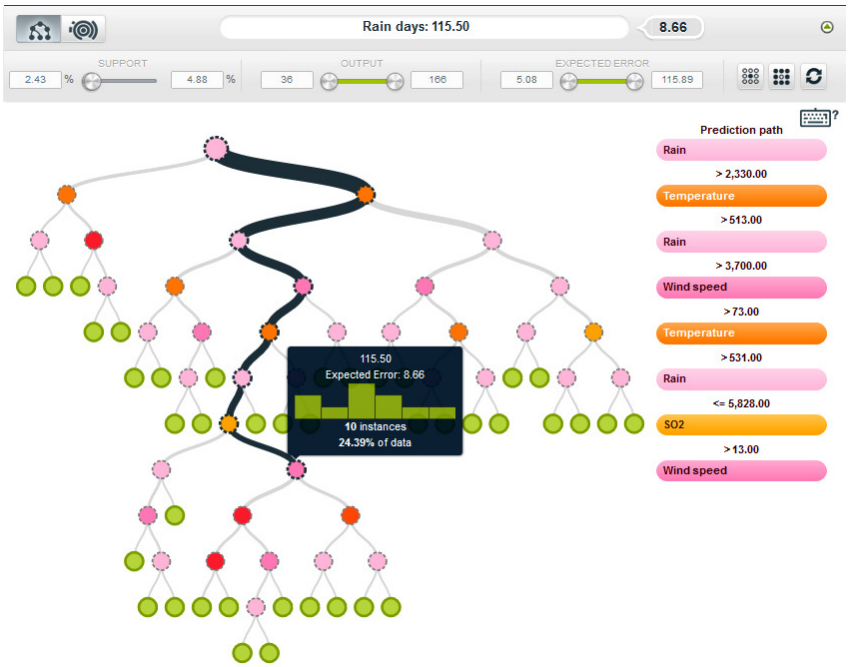


Fig. 2.2. Example of BigML prediction model for air pollution (source: bigml.com).

parts of a dataset), to make predictions on new data. The system provides interactive forms to submit a predictive query for a new data using the input fields from a model or ensemble. The system provides APIs to automate the generation of predictions, which is particularly useful when the number of input fields is high.

- *Models evaluation.* BigML provides functionalities to evaluate the goodness of the predictive models extracted. This is done by generating performance measures that can be applied to the kind of extracted model (classification or regression).

2.5.7 Kognitio Analytical Platform

Kognitio Analytical Platform[18], available as Cloud based service or supplied as a pre-integrated appliance, allows users to pull very large amounts of data from existing data storage systems into high-speed computer memory, allowing complex analytical questions to be answered interactively. Although Kognitio has its own internal disk subsystem, it is primarily used as an analytical

[18] www.kognitio.com

layer on top of existing storage/data processing systems, e.g., Hadoop clusters and/or existing traditional disk-based data warehouse products, Cloud storage, etc. A feature called *External Tables* allows persistent data to reside on external systems. Using this feature the system administrator, or a privileged user, can easily setup access to data that resides in another environment, typically a disk store such as the above-mentioned Hadoop clusters and data warehouse systems. To a final user, the Kognitio Analytical Platform looks like a relational database management system (RDBMS) similar to many commercial databases. However, unlike these databases, Kognitio has been designed specifically to handle analytical query workload, as opposed to the more traditional on-line transaction processing (OLTP) workload. Key reasons of Kognitios high performance in managing analytical query workload are:

- Data is held in high-speed RAM using structures optimized for in-memory analysis, which is different from a simple copy of disk-based data, like a traditional cache.
- Massively Parallel Processing (MPP) allows scaling out across large arrays of low-cost industry standard servers, up to thousands nodes.
- Query parallelization allows every processor core on every server to be equally involved in every query.
- Machine code generation and advanced query plan optimization techniques ensure every processor cycle is effectively used to its maximum capacity.

Parallelism in Kognitio Analytical Platform fully exploits the so-called shared nothing distributed computing approach, in which none of the nodes share memory or disk storage, and there is no single point of contention across the system.

2.5.8 Data Analysis Workflows

A workflow consists of a series of activities, events or tasks that must be performed to accomplish a goal and/or obtain a result. For example, a data analysis workflow can be designed as a sequence of pre-processing, analysis, post-processing, and interpretation steps. At a practical level, a workflow can be implemented as a computer program and can be expressed in a programming language or paradigm that allows expressing the basic workflow steps and includes mechanisms to orchestrate them.

Workflows have emerged as an effective paradigm to address the complexity of scientific and business applications. The wide availability of high-performance computing systems, Grids and Clouds, allowed scientists and engineers to implement more and more complex applications to access and process large data repositories and run scientific experiments in silico on distributed computing platforms. Most of these applications are designed as workflows that include data analysis, scientific computation methods and complex simulation techniques. The design and execution of many scientific appli-

cations require tools and high-level mechanisms. Simple and complex workflows are often used to reach this goal. For this reason, in the past years, many efforts have been devoted towards the development of distributed workflow management systems for scientific applications. Workflows provide a declarative way of specifying the high-level logic of an application, hiding the low-level details that are not fundamental for application design. They are also able to integrate existing software modules, datasets, and services in complex compositions that implement scientific discovery processes.

Another important benefit of workflows is that, once defined, they can be stored and retrieved for modifications and/or re-execution: this allows users to define typical patterns and reuse them in different scenarios [17]. The definition, creation, and execution of workflows are supported by a so-called Workflow Management System (WMS). A key function of a WMS during the workflow execution (or enactment) is coordinating the operations of the individual activities that constitute the workflow. There are several WMSes on the market, most of them targeted to a specific application domain. In the following we focus on some well-known software tools and frameworks designed implementing data analysis workflows on Clouds systems.

Data Mining Cloud Framework

The Data Mining Cloud Framework (DMCF) [96] is a software system that we developd at University of Calabria for allowing users to design and execute data analysis workflows on Clouds. DMCF supports a large variety of data analysis processes, including single-task applications, parameter sweeping applications, and workflow-based applications [97].

A workflow in DMCF can be developed using a visual- or a script-based language. The visual language, called *VL4Cloud*, is based on a design approach for high-level users, e.g., domain expert analysts having a limited knowledge of programming paradigms. The script-based language *JS4Cloud* is provided as a flexible programming paradigm for skilled users who prefer to code their workflows through scripts. Both languages implement a data-driven task parallelism that spawns ready-to-run tasks to Cloud resources. DMCF provides users with a Web-based interface for composing applications and submitting them for execution to a Cloud platform, according to a Software-as-a-Service approach.

Recently, DMCF has been extended to include the execution of MapReduce tasks [12]. Several details about DMCF and its extension for supporting MapReduce tasks have been discussed in Chapter 3.

Microsoft Azure Machine Learning

Microsoft Azure Machine Learning (Azure ML) is a SaaS that provides a Web-based machine learning IDE (i.e., integrated development environment)

for creation and automation of machine learning workflows. Through its user-friendly interface, data scientists and developers can perform several common data analysis/mining tasks on their data and automate their workflows. Using its drag-and-drop interface, users can import their data in the environment or use special readers to retrieve data form several sources, such as Web URL (HTTP), OData Web service, Azure Blob Storage, Azure SQL Database, Azure Table. After that, users can compose their data analysis workflows where each data processing task is represented as a block that can be connected with each other through direct edges, establishing specific dependency relationships among them. Azure ML includes a rich catalog of processing tools that can be easily included in a workflow to prepare/transform data or to mine data through supervised learning (regression e classification) or unsupervised learning (clustering) algorithms. Optionally, users can include their own custom scripts (e.g., in R or Python) to extend the tools catalog. When workflows are correctly defined, users can evaluate them using some testing dataset. An example of workflow built on Microsoft Azure Machine Learning is shown in Figure 2.3.

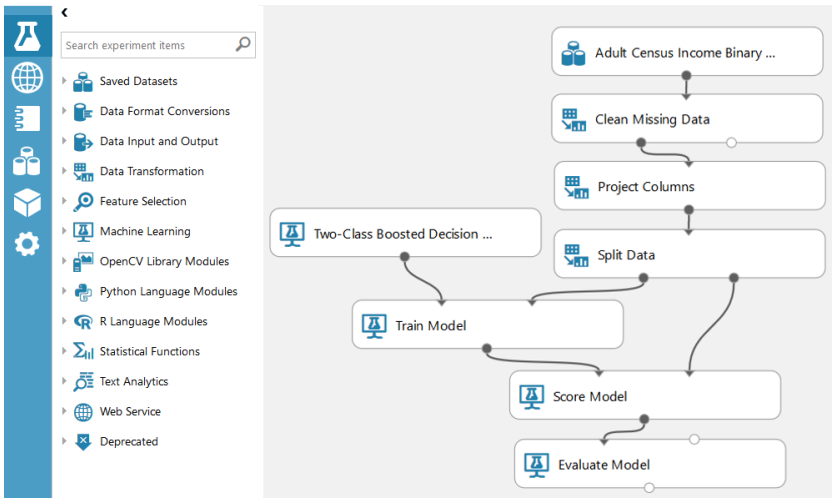


Fig. 2.3. Example of Azure Machine Learning workflow (source: studio.azureml.net).

Users can easily visualize the results of the tests and find very useful information about models accuracy, precision and recall. Finally, in order to use their models to predict new data or perform real time predictions, users can expose them as Web services. Always through a Web-based interface, users can monitor the Web services load and use by time. Azure Machine Learning is a fully managed service provided by Microsoft on its Cloud platform; users do

not need to buy any hardware/software nor manage virtual machine manually. One of the main advantage of working with a Cloud platform like Azure is its auto-scaling feature: models are deployed as elastic Web services so as users do not have to worry about scaling them if the models usage increased.

ClowdFlows

ClowdFlows [80] is an open source Cloud-based platform for the composition, execution, and sharing of data analysis workflows. It is provided as a software as a service that allows users to design and execute visual workflows through a simple Web browser and so it can be run from most devices (e.g., desktop PCs, laptops, and tablets). ClowdFlows is based on two software components: the workflow editor (provided by a Web browser) and the server side application that manages the execution of the application workflows and hosts a set of stored workflows. The server side consists of methods for supporting the client-side workflow editor in the composition and for executing workflows, and a relational database of workflows and data. The workflow editor includes of a workflow canvas and a widget repository. The widget repository is a list of all available workflow components that can be added to the workflow canvas. The repository includes a set of default widgets. Figure 2.4 shows an example of workflow built on CloudFlow.

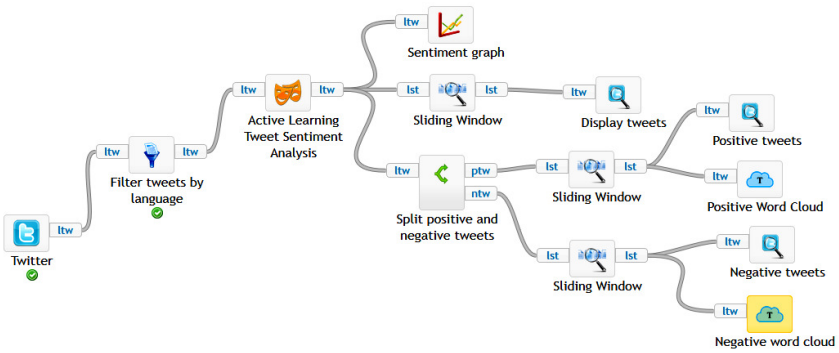


Fig. 2.4. Example of CloudFlow workflow (source: clowdflows.org).

According to this approach, the CloudFlows service-oriented architecture allows users to include in their workflow the implementations of various algorithms, tools and Web services as workflow elements. For example, the Weka's algorithms have been included and exposed as Web services and so they can be added in a workflow application. ClowdFlows is also easily extensible by importing third-party Web services that wrap open-source or custom data mining algorithms. To this end, a user has only to insert the WSDL URL of a

Web service to create a new workflow element that represents the Web service in a workflow application.

Pegasus

Pegasus [37] is a workflow management system developed at the University of Southern California for supporting the implementation of scientific applications also in the area of data analysis. Pegasus includes a set of software modules to execute workflow-based applications in a number of different environments, including desktops, Clouds, clusters and grids. It has been used in several scientific areas including bioinformatics, astronomy, earthquake science, gravitational wave physics, and ocean science. The Pegasus workflow management system can manage the execution of an application expressed as a visual workflow by mapping it onto available resources and executing the workflow tasks in the order of their dependencies. In particular, significant activities have been recently performed on Pegasus to support the system implementation on Cloud platforms and manage computational workflows in the Cloud for developing data-intensive scientific applications (Juve et al., 2010) (Nagavaram et al, 2011). The Pegasus system has been used with IaaS Clouds for workflow applications and the most recent versions of Pegasus can be used to map and execute workflows on commercial and academic IaaS Clouds such as Amazon EC2, Nimbus, OpenNebula and Eucalyptus (Deelman et al., 2015). The Pegasus system includes four main components:

- the Mapper, which builds an executable workflow based on an abstract workflow provided by a user or generated by the workflow composition system. To this end, this component finds the appropriate software, data, and computational resources required for workflow execution. The Mapper can also restructure the workflow in order to optimize performance, and add transformations for data management or to generate provenance information.
- the Execution Engine (DAGMan), which executes in appropriate order the tasks defined in the workflow. This component relies on the compute, storage and network resources defined in the executable workflow to perform the necessary activities. It includes a local component and some remote ones.
- the Task Manager, which is in charge of managing single workflow tasks by supervising their execution on local and/or remote resources.
- The Monitoring Component, which monitors the workflow execution, analyzes the workflow and job logs and stores them into a workflow database used to collect runtime provenance information. This component sends notifications back to users notifying them of events like failures, success and completion of workflows and jobs.

The Pegasus software architecture includes also an error recovery system that attempts to recover from failures by retrying tasks or an entire workflow,

re-mapping portions of the workflow, providing workflow-level checkpointing, and using alternative data sources, when possible. The Pegasus system records provenance information including the locations of data used and produced, and which software was used with which parameters. This feature is useful when a workflow must be reproduced.

Swift

Swift [144] is a implicitly parallel scripting language that runs workflows across several distributed systems, like clusters, Clouds, grids, and supercomputers. The Swift language has been designed at the University of Chicago and at the Argonne National Lab to provide users with a workflow-based language for grid computing. Recently has been ported on Clouds and exascale systems. Swift separates the application workflow logic from runtime configuration. This approach allows a flexible development model.

As the DMCF programming interface, the Swift language allows invocation and running of external application code and allows binding with application execution environments without extra coding from the user. Swift/K is the previous version of the Swift language that runs on the Karajan grid workflow engine across wide area resources. Swift/T is a new implementation of the Swift language for high-performance computing. In this implementation, a Swift program is translated into an MPI program that uses the Turbine and ADLB runtime libraries for scalable dataflow processing over MPI. The Swift-Turbine Compiler (STC) is an optimizing compiler for Swift/T and the Swift Turbine runtime is a distributed engine that maps the load of Swift workflow tasks across multiple computing nodes. Users can also use Galaxy [59] to provide a visual interface for Swift.

The Swift language provides a functional programming paradigm where workflows are designed as a set of code invocations with their associated command-line arguments and input and output files. Swift is based on a C-like syntax and uses an implicit data-driven task parallelism [146]. In fact, it looks like a sequential language, but being a dataflow language, all variables are futures, thus execution is based on data availability. When input data is ready, functions are executed in parallel. Moreover, parallelism can be exploited through the use of the foreach statement. The Turbine runtime comprises a set of services that implement the parallel execution of Swift scripts exploiting the maximal concurrency permitted by data dependencies within a script and by external resource availability. Swift has been used for developing several scientific data analysis applications, such as prediction of protein structures, modeling the molecular structure of new materials, and decision making in climate and energy policy.

2.5.9 NoSQL Models for Data Analytics

With the exponential growth of data to be stored in distributed network scenarios, relational databases exhibit scalability limitations that significantly

reduce the efficiency of querying and analysis [1]. In fact, most relational databases have little ability to scale horizontally over many servers, which makes challenging storing and managing the huge amounts of data produced everyday by many applications.

The NoSQL or non-relational database approach became popular in the last years as an alternative or as a complement to relational databases, in order to ensure horizontal scalability of simple read/write database operations distributed over many servers [24]. Compared to relational databases, NoSQL databases are generally more flexible and scalable, as they are capable of taking advantage of new nodes transparently, without requiring manual distribution of information or additional database management [128]. Since database management may be a challenging task with huge amounts of data, NoSQL databases are designed to ensure automatic data distribution and fault tolerance [53]. In the remainder of this section, we describe some representative NoSQL systems, and discuss some use cases for NoSQL databases, with a focus on data analytics.

NoSQL databases provide ways to store scalar values (e.g., numbers, strings), binary objects (e.g., images, videos), or more complex values. According to their data model, NoSQL databases can be grouped into three main categories [24]: Key-value stores, Document stores, Extensible Record stores.

Key-value stores provide mechanisms to store data as (key, value) pairs over multiple servers. In such kind of databases a distributed hash table (DHT) can be used to implement a scalable indexing structure, where data retrieval is performed by using key to find value [24].

Document stores are designed to manage data stored in documents that use different formats (e.g., JSON), where each document is assigned a unique key that is used to identify and retrieve the document. Therefore, document stores extend key-value stores because they provide for storing, retrieving, and managing semi-structured information, rather than single values. Unlike the key-value stores, document stores generally support secondary indexes and multiple types of documents per database, and provide mechanisms to query collections based on multiple attribute value constraints [24].

Finally, Extensible Record stores (also known as Column-oriented data stores) provide mechanisms to store extensible records that can be partitioned across multiple servers. In this type of database, records are said to be extensible because new attributes can be added on a per-record basis. Extensible record stores provide both horizontal partitioning (storing records on different nodes) and vertical partitioning (storing parts of a single record on different servers). In some systems, columns of a table can be distributed over multiple servers by using column groups, where pre-defined groups indicate which columns are best stored together.

A brief comparison of noSQL databases is shown in Table 2.1. For a more detailed comparison see also [66][86][107].

	DynamoDB	Cassandra	Hbase	Redis	CouchDB	BigTable	MongoDB	Neo4j
Type	KV	Col	Col	KV	Doc	Col	Doc	Graph
Data Storage	MEM FS	HDFS CFS	HDFS	MEM FS	MEM FS	GFS	MEM FS	MEM FS
MapReduce	yes	yes	yes	no	yes	yes	yes	no
Persistence	yes	yes	yes	yes[19]	yes	yes	yes	yes
Replication	yes	yes	yes	yes	yes	yes	yes	yes
Scalability	high	high	high	high	high	high	high	high
Performance	high	high	high	high	high	high	high	high, variable
High availability	yes	yes	yes	yes	yes	yes	yes	yes
Language	Java	Java	Java	Ansi-C	Erlang	Java Python Go Ruby	C++	Java
License	Prop.	Apache2	Apache2	BSD	Apache2	Prop.	AGPL3	GPL3

Legend: *FS=File System; MEM=In-Memory; KV=Key-Value; Doc=Document; Col=Column;*

Table 2.1. Comparison of some NoSQL databases.

Google Bigtable

Google Bigtable[20] is a popular table store. Built above the Google File System, it is able to store up to petabytes of data and supporting tables with billions of rows and thousands of columns. Thanks to its high read and write throughput at low latency, Bigtable it is an ideal data source for batch MapReduce operations [27] and other applications oriented to the processing and analysis of large volumes of data.

Data in Bigtable are stored in sparse, distributed, persistent, multi-dimensional tables composed of rows and columns. Each row is indexed by a single row key, and a set of columns that are grouped together into sets called *column families*. Instead, a generic column is identified by a column family and a *column qualifier*, which is a unique name within the column family. Each value in the table is indexed by a tuple (row key, column key, timestamp). To improve scalability and to balance the query workload, data are ordered by row key and the row range for a table is dynamically partitioned into contiguous blocks, called *tablets*. These tablets are distributed among different Bigtable cluster's nodes (i.e., *Tablet Servers*). To improve load balancing, the Bigtable master is able to split larger and merge smaller tablets, redistributing

[19] With limits due to the fact that last queries can be lost as explained in http://redis.io/topics/persistence

[20] https://cloud.google.com/bigtable/

them across nodes as needed. To ensure data durability, Bigtable stores data on Google File System (GFS) and protects it from disaster events through data replication and backup. Bigtable can be used into applications through multiple clients, including *Cloud Bigtable HBase*, a customized version of the standard client for the industry-standard Apache HBase.

Apache Cassandra

Apache Cassandra[21] is a distributed database management system providing high availability with no single point of failure. Born at Facebook and inspired by Amazon Dynamo and Google BigTable, Apache Cassandra is designed for managing large amount of data across multiple data centers and Cloud availability zones.

Cassandra uses a masterless ring architecture, where all nodes play an identical role, that allows any authorized user to connect to any node in any data center. This is a really simple and flexible architecture that allows to add nodes without service downtime. The process of data distribution across nodes is very simple and no programmatic operations are needed by the developers.

Since all nodes communicate each other equally, Cassandra has no single point of failure, that ensures continuous data availability and service uptime. Moreover, Cassandra provides very customizable data replication service that allows to replicate data across nodes that participate in a ring. In this manner, in case of node failure, one or more copies of the needed data are available on other nodes.

Cassandra also provides built-in and customizable replication, which stores redundant copies of data across nodes that participate in a Cassandra ring. This means that if a node in a cluster goes down, one or more copies of data stored on that node is available on other machines in the cluster. Replication can be configured to work across one data center, many data centers, and multiple Cloud availability zones. Focusing on performance and scalability, Cassandra reaches a quite linear speedup, that means the OPS (Operations Per Second) capacity can be increased by adding new nodes (e.g., if 2 nodes can handle 10,000 OPS, 4 nodes will support 20,000 OPS, and so on).

Many companies have successfully deployed and benefited from Apache Cassandra including some large companies such as: Apple (75,000 nodes storing over 10 PB of data), Chinese search engine Easou (270 nodes, 300 TB, over 800 million reqests per day), and eBay (over 100 nodes, 250 TB), Netflix (2,500 nodes, 420 TB, over 1 trillion requests per day), Instagram, Spotify, eBay, Rackspace, and many more.

Neo4j Graph Database

If we need to take into account real time data relationships (e.g. create queries using data relationships), NoSQL databases are not the best choice. In fact,

[21] http://cassandra.apache.org/

relationship-based or graph databases has been created for naturally supporting operations on data that use data relationships. Graph databases provide a novel and powerful data modeling technique that does not store data in tables, but in graph models [120], with several benefits in storing and retrieving data connected by complex relationships.

There are several graph data models, such as Neo4j, OrientDB, Virtuoso, Allegro, Stardog, InfiniteGraph. Among all we focus on Neo4j. Neo4j is an open-source NoSQL graph database implemented in Java and Scala that is considered the most popular graph database used today. The Neo4j source code and issue tracking are available on GitHub, with a large support community. It is used today by a very large number of organizations working in different sectors, including software analytics, scientific research, project management, recommendations, and social networks.

In the Neo4j graph model, each node contains a list of relationship records that refer to other nodes, and additional attributes (e.g. timestamp, metadata, key-value pairs, and more). Each relationship record must have a name, a direction, a start node and an end node, and can contains additional properties. One o more labels can be assigned both to nodes and relationships. In particular, such labels can be used for representing the roles a node plays in the graph (e.g., user, address, company, and so on) or for associating indexes and constraints to groups of nodes. Figure 2.5 shows an example of a graph model used for detecting bank fraud.

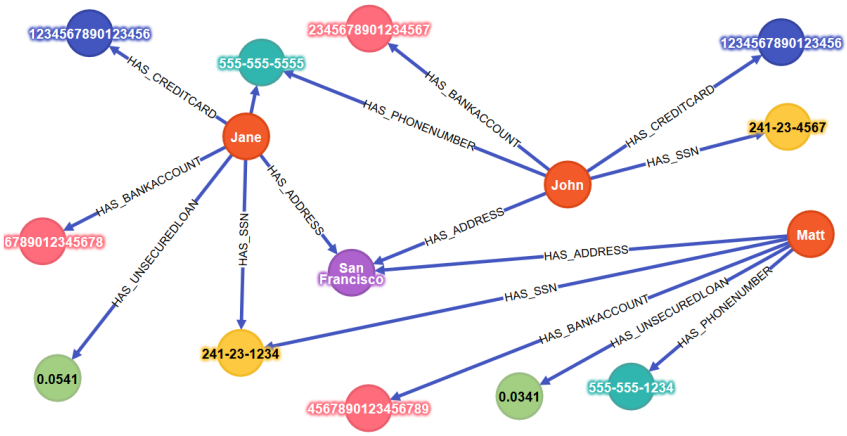


Fig. 2.5. Example of Bank Fraud Graph Dataset (source: neo4j.com).

Moreover, Neo4j clusters are designed for high availability and horizontal read scaling using master-slave replication. Focusing on performance, Neo4j is thousands of times faster than SQL in executing traversal operation. The

traversal operation consists of visiting a set of nodes in the graph by moving along relationships (e.g., find potential friends in social network from user friendship). With such kind of operations, graph models allow to take into account only the data that is required, without d oing expensive grouping operations as done by relational database during join operations [139]. Queries in Neo4j are written using *Cypher*, a declarative and SQL-based language for describing patterns in graphs. Cypher is a relative simple but very powerful language, that allows to execute queries in a easy way on a very complex graph database.

2.5.10 Visual Analytics

A primary problem in data analysis is to interpret results easily. To overcome this problem, in the last years, great progress has been made in the field of visual analytics. As defined by [135], visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces. Nowadays, people use visual analytics tools and methodologies to extract synthetic information from often confusing data and use them in further analysis or business operations. The power of visual analytics techniques relies on human brain capabilities to process graphics faster than text. In particular, through a graphical data presentation, the human brain could be able to find complex and often hidden patterns and relationships in data that are difficult to discover using automatic methods. Also in the Big Data context, the tools used to visualize results and to interact with data play a key role. Thus, in order to support data presentation and interaction also in presence of Big Data, innovative methodologies (e.g, interactive charts, animations, diagrams, and much more) have been developed.

In particular, to ride the wave of visual analytics technologies, several big IT company, such as Microsoft, Google, and SAS, developed advanced data presentation and data visualization tools able to interact with existent Big Data platforms, including Hadoop-based ones. For example, Microsoft extended Excel functions to allow integration with its Big Data solution. In particular, Excel's users can be connected to Azure Storage associated to an Hadoop HDInsight cluster using the Microsoft Power Query for Excel add-in. Once data has been retrieved, users can exploit Excel functions to make more interesting charts or graphs.

Google Fusion Tables[22] is an other alternative for turning data into graphics in a very easy way. It allows to load tabular data, filter and summarize across hundreds of thousands of rows, and create geo maps, heat maps, graphs, charts, animations, and more. Also Google Charts[23] are a powerful Javascript library for making interactive charts for browsers and mobile devices. Google Charts allows to create several types of charts, from simple line charts to

[22] https://tables.googlelabs.com
[23] https://developers.google.com/chart

complex hierarchical tree maps. In the field of maps and location-based applications, advanced platforms, such as Google Maps[24], Mapbox[25], can be used to create interactive and dynamic maps, display additional layers on a map or generate routes. In the field of visual data analysis, several Big Data start-ups spring up in the last years. Tableau[26], for example, is a Big Data company from Stanford with multinational operations in fifteen cities, and more than 39,000 customer accounts in 150 countries. It developed software solutions for easily creating complex charts from huge amount of data. In fact, thanks to its Cloud analytics platform, Tableau allows users to manipulate data through a simple web control panel. In this way, users can interact directly with data to find interesting insights. Among all the competitors in this field, SAS[27] probably stands out among its peers.

SAS Visual Analytics, in fact, represents a complete solution for advanced data visualization and exploratory analyses. Thanks to its drag-and-drop capabilities and no code requirements, it allows users to easily solve complex issues using several sophisticated techniques for data analysis (e.g. decision trees, network diagrams, scenario analysis, path analysis, sentiment analysis) and business intelligence. In addition, exploiting in-memory processing, SAS software makes analytics applications faster.

2.5.11 Big Data funding projects

Open-source projects discussed in the previous sections (e.g., Hadoop, Spark, and NoSQL databases) have been widely used in several public funding projects. As examples:

- BigFoot project[28] is a cloud-based solution featuring scalable and optimized engines to store, process and interact with Big Data. It has received funding from the European Union's Horizon 2020 program.
- Optique[29] is a EU funding project with a total budget of about 14 million EUR. It is aims to provide a novel end-to-end OBDA (Ontology-Based Data Access) [106][22] solution for improving Big Data access. In particular, Optique platform allows to quickly formulate intuitive queries exploiting user vocabularies and conceptualizations, and executing them using massive parallelism.

Also government agencies invested large amount of money on Big Data technologies in many public sector fields, such as intelligence, defense, weather forecasting, crime prediction and prevention, and scientific research.

[24] https://www.google.com/maps
[25] https://www.mapbox.com/
[26] http://www.tableau.com
[27] https://www.sas.com
[28] http://bigfootproject.eu/
[29] http://optique-project.eu

As example, US Administration invested more that 250 million USD for Big Data research and development initiative across multiple agencies and departments. Moreover, in 2014 UK government decided to invest about 73 million GBP in Big Data and other analytics technologies with the goals of creating 58,000 new jobs in Britain by 2017, contributing 216 billion GBP to the countrys economy.

2.5.12 Historical review

In this section a brief historical review of Big Data is presented. Undoubtedly, main events in Big Data evolution are due to big IT and Internet companies, like Google and Yahoo, who faced first the need of new solutions for tackling the rise of Big Data. A significant role in this context has been played by Hadoop and its related projects, that made Big Data analytics accessible also to a larger number of organizations.

Hadoop was created by Doug Cutting and it has its origins in Apache Nutch (2002), an open source web search engine, itself a part of the Lucene project (2000). After Google released the Google File System (GFS) paper (October 2003) and the MapReduce paper (December 2004), Cutting went to work with Yahoo and decided to build open source frameworks based on them: in 2006 Yahoo! created Hadoop based on GFS and MapReduce, and one year later, it started using Hadoop on a 1000 node cluster. In 2006, Yahoo Labs created Pig based on Hadoop, and then donated it to the *Apache Software Foundation* (ASF). In few years, several other projects was created around Hadoop and, in a short time, graduated to a Apache Top Level Project: HBase (2008), Hive (2008), Cassandra (2008), Storm (2011), Giraph (2011), and so on. At the same time, many Hadoop distributor was founded, such as Cloudera (2008), MapR (2009), Hortonworks (2011). A short history of Hadoop and related project is shown in Figure 2.6.

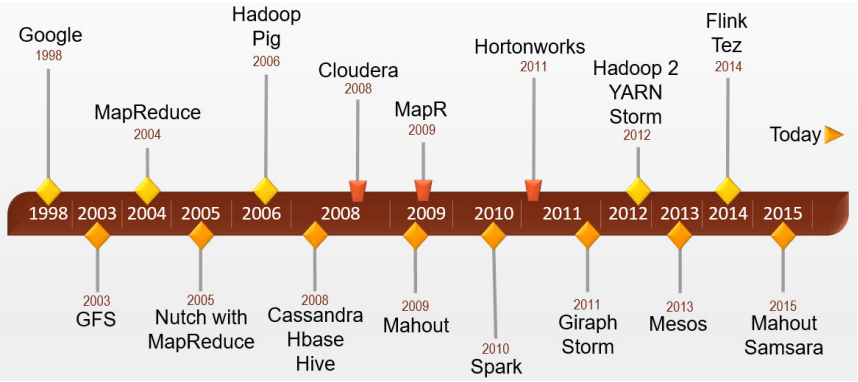


Fig. 2.6. A short Hadoop ecosystem's history.

Spark represents another milestone in Big Data analytics. Spark was initially created at UC Berkeley's AMPLab in 2009, open sourced in 2010 under a BSD license, and donated to the ASF in 2013. Finally, in February 2014, Spark became a Top-Level Apache Project and declared the most active ASF project. As discussed before, Spark is nowadays considered the primary execution engine for several Big Data applications, sometimes used to complement Hadoop.

2.5.13 Summary

It is not easy to summarize all the features of the systems discussed till now or to do a proof comparison among them. Some of those systems have common features and, in some cases, using one rather than another is an hard choice. In fact, given a specific data analytics task, such as a machine learning application, it is possible to use several tools. Some of those are widely used commercial tools, provided through cloud services, that can be easily used by no skilled people (e.g., Azure Machine Learning or Amazon Machine Learning); other are open-source frameworks that require skilled users who prefer to program their application using a more technical approach. In addition, choosing the best solution for developing a data analytics application may depend on many other factors, such as budget (e.g., often high-level services are easy-to-use but more expensive than low-level solutions), data format, data source, the amount of data to be analyze and its velocity, and so on. Table 2.2 presents a brief comparison of the Big Data analytics systems.

Hadoop represents the most used framework for developing distributed Big Data analytics applications. In fact, Hadoop-ecosystem is undoubtedly the most complete solution for any kind of problem, but at the same time it is thought for high skilled users. On the other hand, many other solutions are designed for low-skilled users or for low-medium organizations that do not want to spend resources in developing and maintaining enterprise data analytics solutions (e.g., Microsoft Azure Machine Learning, Amazon Machine Learning, Data Mining Cloud Framework, Kognitio Analytical, or BigML). Finally, other solutions have been created mainly for scientific research purposes and, for this reason, they are poorly used for developing business applications (e.g., Sector/Sphere, Pegasus).

Choosing the best database solution for creating a Big Data application is another key-step, so several aspects need to be considered. To decide what kind of database to adopt, the first aspect to be considered is probably the classes of queries will be run. So graph databases are probably the best solution for representing and querying highly connected data (e.g., data gathered from social network) or that have complex relationships and/or dynamic schema. In any other case, when non-graph data are analyzed, graph databases could result in really bad performance. About that, summary considerations on graph databases are presented in Table 2.3.

Systems/ Tools	Analytics								Open-source	Cloud model
	Streaming	Graph	In-Memory	Machine Learning	SQL	Data flow	Data processing	Workflow		
Hadoop	x	x		x		x	x	x	x	IaaS
Spark	x	x	x	x	x		x		x	IaaS
Mahout			x	x					x	IaaS
Oozie								x	x	IaaS
Tez								x	x	IaaS
Giraph		x							x	IaaS
Storm	x								x	IaaS
Hive					x				x	IaaS
Pig						x			x	IaaS
Hunk										SaaS
Sector /Sphere							x		x	SaaS
BigML				x						SaaS, PaaS
Kognitio Analytical			x		x					PaaS
DMCF				x		x	x	x		SaaS, PaaS
Microsoft Azure ML				x			x	x		SaaS
Amazon ML	x		x	x			x			SaaS
Pegasus								x	x	IaaS
ClowdFlows								x	x	PaaS
Swift								x	x	IaaS

Table 2.2. A brief comparison of most common Big Data analytics systems.

Another aspect to be considered in choosing the best database solution should be the CAP (Consistency, Availability, and Partition) capabilities offered, because distributed NoSQL database systems can't be fully CAP compliant. In fact, the CAP theorem, also named Brewer's theorem[60], states that a distributed system can't simultaneously guarantee all three of the following properties:

- Consistency (C), that means all nodes see the same data at the same time;
- Availability (A), that means every request will receive a response within a reasonable amount of time;
- Partition (P) tolerance, that means the system continues to function also if arbitrary network partitions occur due to failures.

Thus if a distributed database system guarantees Consistency and Partitioning, it can never ensure Availability. Similarly, if you need a full Availability and Partition tolerance, you can't have Consistency, anyway not immediately. In fact, on a distributed environment, data changes on one node need some time to be propagated to the other nodes. During that time the copies will be mutually inconsistent, that may lead to the possibility of reading not updated data. To try to overcome this limitation, the *Eventual Consistency* property is usually provided: it ensures that the system, sooner or later, will become consistent. This is a weak property, so if the adopted database system only provides eventual consistency, the developer must be aware that exists

the possibility of reading inconsistent data. NoSQL databases usually offer a balance among CAP properties, which is the key difference among the different available solutions. For each database family, some summary considerations are also provided for Key-Value databases (Table 2.4), Column-oriented (Table 2.5), and Document-oriented databases (Table 2.6).

Graph databases	
Horizontal scaling	Poor horizontal scaling.
When to use	For storing objects without a fixed schema and linked together by relationships; when users can done naturally their reasoning about data via graph traversals instead of using complex SQL queries.
CAP tradeoff	Usually prefer availability over consistency
Pros	Powerful data modeling and relationships representation; locally indexed connected data; easy to query.
Cons	Highly specialized query capabilities that make them the best for graph data, but not suitable for non-graph data.

Table 2.3. Summary considerations about graph databases.

Key-value databases	
Horizontal scaling	Very high scale provided via sharding.
When to use	When you have a very simple data schema or extreme speed scenario (like real-time)
CAP tradeoff	Most solutions prefer consistency over availability.
Pros	Simple data model; very high scalability, data can be accessed using query language like SQL.
Cons	Some queries could be inefficient or limited due to sharding (e.g., join operations across shards); no API standardization; maintenance is difficult; poor for complex data.

Table 2.4. Summary considerations about Key-Value databases.

2.6 Research Trends

Big Data analysis is a very active research area with significant impact on industrial and scientific domains where is important to analyze very large and complex data repositories. In particular, in many cases data to be analyzed are stored in Cloud platforms and elastic computing Clouds facilities are exploited

Column-oriented databases	
Horizontal scaling	Very high scale capabilities.
When to use	When you need consistency and higher scalability performance than a single machine (i.e., usually using more than 1,000 nodes), without using indexed caching front end.
CAP tradeoff	Most solutions prefer consistency over availability.
Pros	Higher throughput and stronger concurrency when it is possible to partition data; multi-attribute queries; data is naturally indexed by columns; support semi-structured data.
Cons	More complex than the document stores; poor for interconnected data.

Table 2.5. Summary considerations about Column-oriented databases.

Document-oriented databases	
Horizontal scaling	Scale provided via replication or replication and sharding.
When to use	When your record structure is relatively small and it is possible to store all of its related properties in a single doc.
CAP tradeoff	In most cases prefer consistency over availability.
Pros	High scalability and simple data model; generally support secondary indexes, multiple types of documents per database, and nested documents or lists; MapReduce support for adhoc querying.
Cons	Eventually consistent model with limited atomicity and isolation; poor for interconnected data; query model is limited to keys and indexes.

Table 2.6. Summary considerations about Document-oriented databases.

to speedup the analysis. This section outlines and discusses main research trends in Big Data analytics and Cloud systems for managing and mining large-scale data repositories.

As we discussed, scalable data analytics requires high-level, easy-to-use design tools for programming large applications dealing with huge, distributed data sources. Moreover, Clouds are widely adopted by many organizations, however several existing issues remain to be addressed, so that Cloud solutions can improve their efficiency and competitiveness at each business size, from medium to large companies. This requires further research and development in several key areas such as:

- Programming models for Big Data analytics. Big Data analytics programming tools require novel complex abstract structures. The MapReduce model is often used on clusters and Clouds, but more research is needed to develop scalable higher-level models and tools. State-of-the-art solu-

tions generated major success stories, however they are not mature and suffer several problems from data transfer bottlenecks to performance unpredictability. Several other processing models have been proposed as alternative to MapReduce, such as Dryad [70] or Pregel [92], but they have never been widely used by developers.

- Data storage scalability. The increasing amount of data generated needs even more scalable data storage systems. As discussed in the previously, traditional RDBMS systems are not the best choice for supporting Big Data applications in the Cloud, and that leads to the popularity of noSQL platforms[24]. Several noSQL solutions have been proposed, with good experimental results in term of performance gain, but several other improvements are still needed [141][129]. In fact, RDBMS systems have been around for a long time, are quite stable and offers lots of features. In the other hand, most noSQL systems are in its early version and several additional features have yet to be improved or implemented, such as integrating capabilities from DBMS (e.g., indexing techniques), facilities for ad-hoc queries, and more.
- Data availability. Cloud service provides have to deal with the problem of granting service and data availability. Especially in presence of huge amounts of data, granting high-quality service is an opened challenge. Several solutions have been proposed for improving exploitation, such as using a cooperative multi-Cloud model to support Big Data accessibility in emergency cases [83], but more studies are still needed to handle the continue increasing demand for more real time and broad network access to Cloud services.
- Data and tool interoperability and openness. Interoperability is a main issue in large-scale applications that use resources such as data and computing nodes. Standard formats and models are needed to support interoperability and ease cooperation among teams using different data formats and tools. The *National Institute of Standards and Technology* (NIST) just released the Big Data interoperability framework[30], a collection of documents, organized in 7 volumes, which aim to define some standards for Big Data.
- Data quality and usability. Big Data sets are often arranged by gathering data from several heterogeneous and often not well-known sources. This leads to a poor data quality that is a big problem for data analysts. In fact, due to the lack of a common format, inconsistent and useless data can be produced as a result of joining data from heterogeneous sources. Defining some common and widely adopted format would lead to data that are consistent with data from other sources, that means high quality data. Since real-world data is highly susceptible to inconsistency, incompleteness, and noise, finding effective methodologies for data preprocessing is still an open challenge for improve data quality and the analysis results [28]. In this re-

[30] http://www.nist.gov/itl/bigdata/bigdatainfo.cfm

gard, an interesting discussion about challenges of data quality in the Big Data has been presented in [20].

- Integration of Big Data analytics frameworks. The service-oriented paradigm allows running large-scale distributed workflows on heterogeneous platforms along with software components developed using different programming languages or tools. Scalable software architectures for fine grain in-memory data access and analysis. Exascale processors and storage devices must be exploited with fine-grain runtime models. Software solutions for handling many cores and scalable processor-to-processor communications have to be designed to exploit exascale hardware [103][41].
- Tools for massive social network analysis. The effective analysis of social network data on a large scale requires new software tools for real-time data extraction and mining, using Cloud services and high-performance computing approaches [93][102]. Social data streaming analysis tools represent very useful technologies to understand collective behaviors from social media data. Tools for data exploration and models visualization. New approaches to data exploration and models visualization are necessary taking into account the size of data and the complexity of the knowledge extracted. As data are bigger and bigger, visualization tools will be more useful to summarize and show data patterns and trends in a compact and easy-to-see way.
- Local mining and distributed model combination. As Big Data applications often involve several local sources and distributed coordination, collecting distributed data sources to a centralized server for analysis is not practical or in some cases possible. Scalable data analysis systems have to enable local mining of data sources and model exchange and fusion mechanisms to compose the results produced in the distributed nodes [147]. According to this approach the global analysis can be performed by distributing the local mining and supporting the global combination of every local knowledge to generate the complete model.
- In-memory analysis. Most of the data analysis tools query data sources on disks while, differently from those, in-memory analytics query data in main memory (RAM). This approach brings many benefits in terms of query speed up and faster decisions. In-memory databases are, for example, very effective in real-time data analysis, but they require high-performance hardware support and fine-grain parallel algorithms [134][155]. New 64-bit operating systems allow to address memory up to one terabyte, so making realistic to cache very large amount of data in RAM. This is why this research area is very promising.

2.7 Conclusions

In the last years the ability to gather data has increased exponentially. Advances and pervasiveness of computers have been the main driver of the very

huge amounts of digital data that today are collected and stored in digital repositories. Those data volumes can be analyzed to extract useful information and producing helpful knowledge for science, industry, public services and in general for humankind. However, the huge amount of data generated, the speed at which it is produced, and its heterogeneity, represent a challenge to the current storage, process and analysis capabilities. Then to extract value from such kind of data, novel technologies and architectures have been developed by data scientists for capturing and analyzing complex and/or high velocity data. In this scenario was born also the Big Data mining field as a discipline that today provides several different techniques and algorithms for the automatic analysis of large datasets. But, the process of knowledge discovery from Big Data is not so easy, mainly due to data characteristics, and to get valuable information and knowledge in shorter time, high performance and scalable computing systems are needed. In many cases, Big Data are stored and analyzed in Cloud platforms.

Clouds provide scalable storage and processing services that can be used for extracting knowledge from Big Data repositories, as well as software platforms for developing and running data analysis environments on top of such services. In this chapter we provided an overview of Cloud technologies by describing the main service models (SaaS, PaaS, and IaaS) and deployment models (public, private or hybrid Clouds) adopted by Cloud providers. We also described representative examples of Cloud environments (Microsoft Azure, Amazon Web Services, OpenNebula and OpenStack) that can be used to implement applications and frameworks for data analysis in the Cloud. The development of data analysis applications on Cloud computing systems is a complex task that needs to exploit smart software solutions and innovative technologies. In this chapter we presented the leading software tools and technologies used for developing scalable data analysis on Clouds, such as MapReduce, Spark, workflow systems, and NoSQL database management systems. In particular, we particularly focused on Hadoop, the best-known MapReduce implementation, that is commonly used to develop scalable applications that analyze big amounts of data. As we discussed, Hadoop is also a reference tool for several other frameworks, such as Storm, Hive, Oozie and Spark. Moreover, besides Hadoop and its ecosystem, several other MapReduce implementations have been implemented within other systems, including GridGain, Skynet, MapSharp, and Disco.

As such Cloud platforms become available, researchers are increasingly porting powerful data mining programming tools and strategies to the Cloud to exploit complex and flexible software models, such as the distributed workflow paradigm. Workflows provide a declarative way of specifying the high-level logic of an application, hiding the low-level details. They are also able to integrate existing software modules, datasets, and services in complex compositions that implement discovery processes. In this chapter we presented several data mining workflow systems, such as Data Mining Cloud Framework, Microsoft Azure Machine Learning, ClowdFlows.

Then we also discussed NoSQL database technology that became popular in the latest years as an alternative or as a complement to relational databases. In fact, NoSQL systems in several application scenarios are more scalable and provide higher performance than relational databases. We introduced the basic principles of NoSQL, described representative NoSQL systems, and outlined interesting data analytics use cases where NoSQL tools are useful. Finally, some research trends and open challenges on Big Data analysis has been discussed, such as scalable data analytics requirements of high-level, easy-to-use design tools for programming large applications dealing with huge distributed data sources.

3

Workflows and MapReduce: Integration on Clouds

Cloud computing systems provide large-scale computing infrastructures for complex high-performance applications, such as those that use advanced data analytics techniques for extracting useful information from large, complex datasets. However, combining Big Data analytics techniques with scalable computing systems will produce new insights in a shorter time [132].

Although a few Cloud-based analytics platforms are available today, current research work foresees that they will become common within a few years. As discussed in Chapter 2, some current solutions are open source systems such as Apache Hadoop and ClowdFlows, while others are proprietary solutions provided by companies such as Google, IBM, Microsoft, EMC, BigML, Hunk, and Kognitio. As such platforms become available, researchers are increasingly porting powerful data mining programming tools and strategies to the Cloud to exploit complex and flexible software models, such as the distributed workflow paradigm. Over the last years, data analysis workflows became really effective in expressing task coordination and they can be designed through visual- and script-based programming paradigms.

Moreover, the increasing use of service-oriented computing in many application domains is accelerating this trend. Developers and researchers can adopt the Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) models to distribute Big Data analytics applications as high-level services on Clouds. This approach creates a new way to deliver data analysis software that is called Data Analytics-as-a-Service (DAaaS).

This chapter describes the Data Mining Cloud Framework (DMCF) that was developed at the University of Calabria according to this approach. In the DMCF, data analysis workflows can be designed through a visual- or a script-based formalism. The visual formalism, called *VL4Cloud*, is a very effective design approach for high-level users, e.g., domain expert analysts having a limited knowledge of programming languages. As an alternative, the script-based language, called *JS4Cloud*, offers a flexible programming approach for skilled users who prefer to program their workflows using a more

technical approach. The DMCF's workflow model has been integrated with the MapReduce paradigm. In such way, VL4Cloud/JS4Cloud workflows can include MapReduce tasks and can be executed in parallel to support scalable data analysis on Clouds.

The remainder of the chapter is organized as follows. Section 3.1 presents the DMCF by introducing its architecture, the parallel execution model and the workflow-based programming paradigm offered by VL4Cloud and JS4Cloud. Section 3.2 describes how the VL4Cloud and JS4Cloud languages have been extended to integrate with the MapReduce model. Section 3.2.3 describes a data mining application implemented using the proposed approach. Section 3.3 discusses related work. Finally, Section 3.4 concludes the chapter.

3.1 Data Mining Cloud Framework

The Data Mining Cloud Framework (DMCF) is a software system developed for allowing users to design and execute data analysis workflows on Clouds. DMCF supports a large variety of data analysis processes, including single-task applications, parameter sweeping applications, and workflow-based applications. A Web-based user interface allows users to compose their applications and to submit them for execution to a Cloud platform, according to a Software-as-a-Service approach.

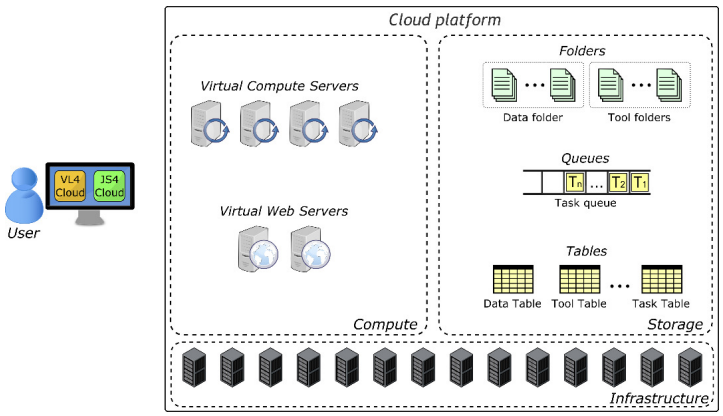


Fig. 3.1. Architecture of Data Mining Cloud Framework.

The DMCF's architecture includes a set of components that can be classified as storage and compute components [96] (see Figure 3.1).

The storage components include:

- A *Data Folder* that contains data sources and the results of knowledge discovery processes. Similarly, a *Tool folder* contains libraries and executable

files for data selection, pre-processing, transformation, data mining, and results evaluation.
- The *Data Table*, *Tool Table* and *Task Table* that contain metadata information associated with data, tools, and tasks.
- The *Task Queue* that manages the tasks to be executed.

The compute components are:

- A pool of *Virtual Compute Servers*, which are in charge of executing the data mining tasks.
- A pool of *Virtual Web Servers* host the Web-based user interface.

The user interface provides three functionalities:

i) *App submission*, which allows users to submit single-task, parameter sweeping, or workflow-based applications;
ii) *App monitoring*, which is used to monitor the status and access results of the submitted applications;
iii) *Data/Tool management*, which allows users to manage input/output data and tools.

The DMCF's architecture has been designed as a reference architecture to be implemented on different Cloud systems. However, a first implementation of the framework has been carried out on the Microsoft Azure Cloud platform[1] and has been evaluated through a set of data analysis applications executed on a Microsoft Cloud data center.

The DMCF takes advantage of cloud computing features, such as elasticity of resources provisioning. In DMCF, at least one Virtual Web Server runs continuously in the Cloud, as it serves as user front-end. In addition, users specify the minimum and maximum number of Virtual Compute Servers. The DMCF can exploit the auto-scaling features of Microsoft Azure that allows dynamic spinning up or shutting down Virtual Compute Servers, based on the number of tasks ready for execution in the DMCF's Task Queue. Since storage is managed by the Cloud platform, the number of storage servers is transparent to the user.

The remainder of the section outlines applications execution in the DMCF, and describes the DMCF's visual- and script-based formalisms used to implement workflow applications.

3.1.1 Applications execution

For designing and executing a knowledge discovery application, users interact with the system performing the following steps:

1. The Website is used to design an application (either single-task, parameter sweeping, or workflow-based) through a Web-based interface that offers both the visual programming interface and the script.

[1] https://azure.microsoft.com/

2. When a user submits an application, the system creates a set of tasks and inserts them into the Task Queue on the basis of the application requirements.
3. Each idle Virtual Compute Server picks a task from the Task Queue, and concurrently executes it.
4. Each Virtual Compute Server gets the input dataset from the location specified by the application. To this end, file transfer is performed from the Data Folder where the dataset is located, to the local storage of the Virtual Compute Server.
5. After task completion, each Virtual Compute Server puts the result on the Data Folder.
6. The Website notifies users as soon as their task(s) have completed, and allows them to access the results.

The set of tasks created on the second step depends on the type of application submitted by a user. In the case of a single-task application, just one data mining task is inserted into the Task Queue. If users submit a parameter sweeping application, a set of tasks corresponding to the combinations of the input parameters values are executed in parallel. If a workflow-based application has to be executed, the set of tasks created depends on how many data analysis tools are invoked within the workflow. Initially, only the workflow tasks without dependencies are inserted into the Task Queue [3].

3.1.2 Workflow formalisms

The DMCF allows creating data mining and knowledge discovery applications using workflow formalisms. Workflows may encompass all the steps of discovery based on the execution of complex algorithms and the access and analysis of scientific data. In data-driven discovery processes, knowledge discovery workflows can produce results that can confirm real experiments or provide insights that cannot be achieved in laboratories. In particular, DMCF allows to program workflow applications using two languages:

i) *VL4Cloud* (Visual Language for Cloud), a visual programming language that lets users develop applications by programming the workflow components graphically[98].
ii) *JS4Cloud* (JavaScript for Cloud), a scripting language for programming data analysis workflows based on JavaScript[101].

Both languages use two key programming abstractions:

i) *Data* elements denote input files or storage elements (e.g., a dataset to be analyzed) or output files or stored elements (e.g., a data mining model).
ii) *Tool* elements denote algorithms, software tools or complex applications performing any kind of operation that can be applied to a data element (data mining, filtering, partitioning, etc.).

Another common element is the task concept, which represents the unit of parallelism in our model. A *task* is a Tool, invoked in the workflow, which is intended to run in parallel with other tasks on a set of Cloud resources. According to this approach, VL4Cloud and JS4Cloud implement a *data-driven task parallelism*. This means that, as soon as a task does not depend on any other task in the same workflow, the runtime asynchronously spawns it to the first available virtual machine. A task T_j does not depend on a task T_i belonging to the same workflow (with $i \neq j$), if T_j during its execution does not read any data element created by T_i.

In VL4Cloud, workflows are directed acyclic graphs whose nodes represent data and tools elements. The nodes can be connected with each other through direct edges, establishing specific dependency relationships among them. When an edge is being created between two nodes, a label is automatically attached to it representing the type of relationship between the two nodes. Data and Tool nodes can be added to the workflow singularly or in array form. A data array is an ordered collection of input/output data elements, while a tool array represents multiple instances of the same tool. Figure 3.2 shows an example of data analysis workflow developed using the visual workflow formalism of DMCF[99].

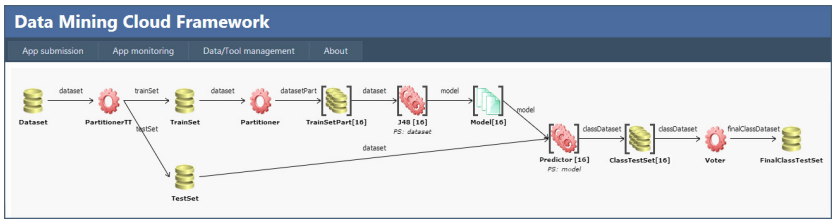


Fig. 3.2. Example of data analysis application designed using VL4Cloud.

In JS4Cloud, workflows are defined with a JavaScript code that interacts with Data and Tool elements through three functions:

i) *Data Access*, for accessing a Data element stored in the Cloud;
ii) *Data Definition*, to define a new Data element that will be created at runtime as a result of a Tool execution;
iii) *Tool Execution*, to invoke the execution of a Tool available in the Cloud.

Once the JS4Cloud workflow code has been submitted, an interpreter translates the workflow into a set of concurrent tasks by analysing the existing dependencies in the code. The main benefits of JS4Cloud are:

1. it extends the well-known JavaScript language while using only its basic functions (arrays, functions, loops);

2. it implements both a data-driven task parallelism that automatically spawns ready-to-run tasks to the Cloud resources, and data parallelism through an array-based formalism;
3. these two types of parallelism are exploited implicitly so that workflows can be programmed in a totally sequential way, which frees users from duties like work partitioning, synchronization and communication.

Figure 3.3 shows the script-based workflow version of the visual workflow shown in Figure 3.2. In this example, parallelism is exploited in the for loop at line 7, where up to 16 instances of the J48 classifier are executed in parallel on 16 different partitions of the training sets, and in the for loop at line 10, where up to 16 instances of the Predictor tool are executed in parallel to classify the test set using 16 different classification models.

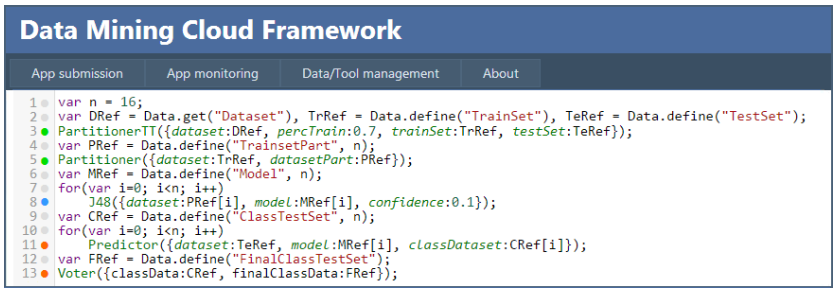


Fig. 3.3. Example of data analysis application designed using JS4Cloud.

Figure 3.3 shows a snapshot of the parallel classification workflow taken during its execution in the DMCF's user interface. Beside each code line number, a colored circle indicates the status of execution. This feature allows user to monitor the status of the workflow execution. Green circles at lines 3 and 5 indicate that the two partitioners have completed their execution; the blue circle at line 8 indicates that J48 tasks are still running; the orange circles at lines 11 and 13 indicate that the corresponding tasks are waiting to be executed.

3.2 Extending VS4Cloud/JS4Cloud with MapReduce

In this section, we describe how the DMCF has been extended to include the execution of MapReduce tasks. In particular, we describe the MapReduce programming model, why it is widely used by data specialists, and how the DMCF's languages have been extended to support MapReduce applications.

3.2.1 Motivations

As introduced in Chapter 2, MapReduce and its best-known implementation Hadoop[2] have become widely used by data specialists to develop parallel applications that analyze big amount of data. Hadoop is designed to scale up from a single server to tens of thousands of servers, and has become the focus of several other projects, including Spark[3] for in-memory machine learning and data analysis, Storm[4] for streaming data analysis, Hive[5] as data warehouse software to query and manage large datasets, and Pig[6] as dataflow language for exploring large datasets.

Algorithms and applications written using MapReduce are automatically parallelized and executed on a large number of servers. Consequently, MapReduce has been widely used to implement data mining algorithms in parallel. Chu et al. [30] offer an overview of how several learning algorithms can be efficiently implemented using MapReduce. More in details, the authors demonstrate that MapReduce shows basically a linear speedup with an increasing number of processors on a variety of learning algorithms such as K-means, neural networks and Expectation-Maximization probabilistic clustering. Ricardo project [33] is a platform that integrate R[7] statistical tools and Hadoop to support parallel data analysis. The use of MapReduce for data intensive scientific analysis and bioinformatics is deeply analyzed in [46]. For the reasons discussed above and for the large number of MapReduce algorithms and applications available online, we designed an extension of the DMCF's workflow formalism to support also the execution of MapReduce tools.

3.2.2 Integration model

In DMCF, a Tool represents a software tool or service performing any kind of process that can be applied to a data element (data mining, filtering, partitioning, etc.).

As shown in Figure 3.4, three different types of Tools can be used in a DCMF workflow:

i) A *Batch Tool* is used to execute an algorithm or a software tool on a Virtual Compute Server without user interaction. All input parameters are passed as command-line arguments.
ii) A *Web Service Tool* is used to insert into a workflow a Web service invocation. It is possible to integrate both REST and SOAP-based Web services [113].

[2] https://hadoop.apache.org/
[3] https://spark.apache.org/
[4] https://storm.apache.org/
[5] https://spark.apache.org/
[6] https://pig.apache.org/
[7] https://www.r-project.org/

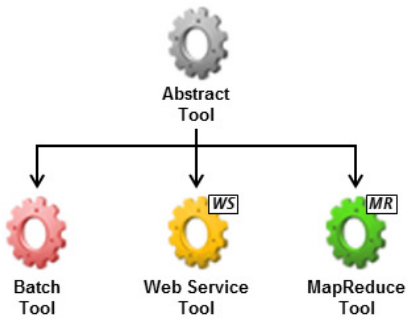


Fig. 3.4. Types of Tools available in DMCF.

iii) A *MapReduce Tool* is used to insert into a workflow the execution of a MapReduce algorithm or application running on a cluster of virtual servers.

For each Tool in a workflow, a *Tool descriptor* includes a reference to its executable, the required libraries, and the list of input and output parameters. Each parameter is characterized by *name, description, type*, and can be *mandatory* or *optional*. In more detail, a MapReduce Tool descriptor is composed by two groups of parameters: *generic parameters*, which are parameters used by the MapReduce runtime, and *applications parameters*, which are parameters associated to specific MapReduce applications. In the following, we list a few examples of generic parameters:

- *mapreduce.job.reduces*: the number of reduce tasks per job;
- *mapreduce.job.maps*: the number of map tasks per job;
- *mapreduce.input.fileinputformat.split.minsize*: the minimum size of chunk that map input should be split into;
- *mapreduce.input.fileinputformat.split.maxsize*: the maximum size of chunk that map input should be split into;
- *mapreduce.map.output.compress*: enable the compression of the intermediate mapper outputs before being sent to the reducers.

Figure 3.5 shows an example of MapReduce Tool descriptor for an implementation of the Random Forest algorithm. As shown by the descriptor, the algorithm can be configured with the following parameters: a set of input files (*dataInput*), the number of trees that will be generated (*nTrees*), the minimum number of elements for node split (*minSplitNum*), the column containing the class labels (*classColumn*), and the output models (*dataOutput*). The DMCF uses this descriptor to allow the inclusion of a RandonForest algorithm in a workflow, and to execute it on a MapReduce cluster.

```
"MapReduceRandomForest": {
    "libraryList": ["java.exe", "MapReduceRandomForest"],
    "executable": "java.exe -jar .... ",
    "parameterList": [{
        "name": "dataset", "flag": "-in",
        "mandatory": true, "parType": "IN",
        "type": "file", "array": false,
        "description": "Input file or folder"
    },{
        "name": "nTrees", "flag": "-nTrees",
        "mandatory": false, "parType": "OP",
        "type": "int", "array": false,
        "description": "The number of trees",
        "value": "100"
    },{
        "name": "minSplitNum", "flag": "-minSplitNum",
        "mandatory": false, "parType": "OP",
        "type": "int", "array": false,
        "description": "Minimum number of elements for node split",
        "value": "2"
    },{
        "name": "classColumn", "flag": "-class",
        "mandatory": false, "parType": "OP",
        "type": "int", "array": false,
        "description": "Column index to use as the class",
        "value": ""
    }
    ...
    ,{
        "name": "output", "flag": "-out",
        "mandatory": true, "parType": "OUT",
        "type": "file", "array": false,
        "description": "Output file or folder"
    }]
}
```

Fig. 3.5. Example of MapReduce descriptor in JSON format.

3.2.3 A Data Mining Application Case

In this section, we describe a DMCF data mining application whose workflow includes MapReduce computations. Through this example, we show how the MapReduce paradigm has been integrated into DMCF workflows, and how it can be used to exploit the inherent parallelism of the application.

The goal of this application is to implement a predictor of the arrival delay of scheduled flights due to weather conditions. The predicted arrival delay takes into consideration both implicit flight information (origin airport, destination airport, scheduled departure time, scheduled arrival time) and weather forecast at origin and destination airports. In particular, we consider the closest weather observation at origin and destination airports based on scheduled flight departure and arrival time. If the predicted arrival delay of a scheduled flight is less than a given threshold, it is classified as an on-time flight; otherwise, it is classified as a delayed flight.

Two open datasets of airline flights and weather observations have been used. The first dataset is the *Airline On-Time Performance* (AOTP) provided by RITA - *Bureau of Transportation Statistics*[8], which contains data for domestic US flights by major air carriers. The second one is the Quality *Controlled Local Climatological Data* (QCLCD) dataset available from the National Climatic Data Center[9], which contains hourly weather observations from about 1,600 U.S. stations. For data classification, a MapReduce version of the Random Forest (RF) algorithm has been used.

More details about the methodology and algorithms used for developing the flight delay predictor can be found in Chapter 6.

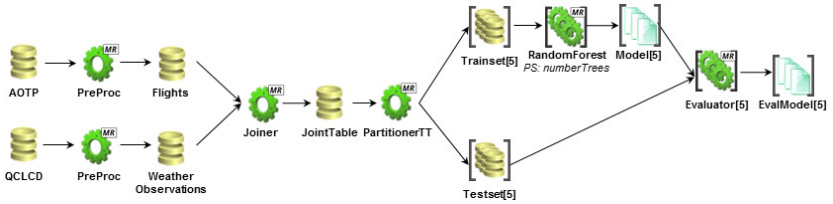


Fig. 3.6. Flight delay analysis workflow using DMCF with MapReduce.

Using DMCF, we created a workflow for the whole data analysis process (see Figure 3.6). The workflow begins by pre-processing the AOTP and the QCLCD datasets using two instances of *PreProc* Tool. These steps look for possible wrong data, treating missing values, and filtering out diverted and cancelled flights and weather observations not related to airport locations. Then, a *Joiner* Tool executes a relational join between *Flights* and *Weather Observations* data in parallel using a MapReduce algorithm. The result is a *JointTable*. Then, a *PartionerTT* Tool creates five pairs of $\langle Trainset, Testset \rangle$ using different delay threshold values. The five instances of training set and test set are represented in the workflow as two data array nodes, labelled as *Trainset[5]* and *Testset[5]*.

Then, five instances of the *RandomForest* Tool analyze in parallel the five instances of *Trainset* to generate five models (*Model[5]*). For each model, an instance of the *Evaluator* Tool generates the confusion matrix (*EvalModel*), which is a commonly used method to measure the quality of classification. Starting from the set of confusion matrices obtained, these tools calculate some metrics, e.g., accuracy, precision, recall, which can be used to select the best model.

[8] http://www.transtats.bts.gov
[9] http://cdo.ncdc.noaa.gov/qclcd/QCLCD

3.3 Related work

Several systems have been proposed to design and execute workflow-based applications[131], but only some of them currently work on the Cloud and support visual or script-based workflow programming. The most known systems are Taverna [145], Orange4WS [114][38], Kepler [87], E-Science Central (e-SC) [67], ClowdFlows [80], Pegasus [36][73], WS-PGRADE [74] and Swift [144].

In particular, Swift is a parallel scripting language that executes workflows across several distributed systems, like clusters, Clouds, grids, and supercomputers. It provides a functional language in which workflows are modelled as a set of program invocations with their associated command-line arguments, input and output files. Swift uses a C-like syntax consisting of function definitions and expressions that provide a data-driven task parallelism. The runtime includes a set of services that implement the parallel execution of Swift scripts exploiting the maximal concurrency permitted by data dependencies within a script and by external resource availability. Swift users can use Galaxy [59] to provide a visual interface for Swift [90].

For comparison purposes, we distinguish two types of parallelism levels: workflow parallelism, which refers to the ability of executing multiple workflows concurrently; and task parallelism, which is the ability of executing multiple tasks of the same workflow concurrently. Most systems, including DMCF, support both workflow and task parallelisms, except for ClowdFlows and E-Science Central that focus on workflow parallelism only. Most systems are provided according with the SaaS model (e.g., E-Science Central, ClowdFlows, Pegasus, WS-PGRADE, Swift+Galaxy and DMCF), whereas Taverna, Kepler and Orange4WS are implemented as desktop applications that can invoke Cloud software exposed as Web Services. All the SaaS systems are implemented on top of Infrastructure-as-a-Service (IaaS) Clouds, except for DMCF that is designed to run on top of Platform-as-a-Service (PaaS) Clouds. DMCF is one of the few SaaS systems featuring both workflow/task parallelism and support to data/tool arrays. However, differently from the data/tool array formalisms provided by the other systems, DMCF's arrays make explicit the parallelism level of each workflow node, i.e., the number of input/output datasets (in case of data arrays) and the number of tools to be concurrently executed (in case of tool arrays). Furthermore, DMCF is the only system designed to run on top of a PaaS. A key advantage of this approach is the independence from the infrastructure layer. In fact, the DMCF's components are mapped into PaaS services, which in turn are implemented on infrastructure components. Changes to the Cloud infrastructure affect only the infrastructure/platform interface, which is managed by the Cloud provider, and therefore DMCF's implementation and functionality are not influenced. In addition, the PaaS approach facilitates the implementation of the system on a public Cloud, which free final users and organizations from any hardware and OS management duties.

3.4 Conclusions

Data analysis applications often involve big data and complex software systems in which multiple data processing tools are executed in a coordinated way. Big data refers to massive, heterogeneous, and often unstructured digital content that is difficult to process using traditional data management tools and technique. Cloud computing systems provide elastic services, high performance and scalable data storage, which can be used as large-scale computing infrastructures for complex high-performance data mining applications.

Data analysis workflows are effective in expressing task coordination and can be designed through visual and script-based formalisms. According to this approach, we described the Data Mining Cloud Framework (DMCF), a system supporting the scalable execution of data analysis computations on Cloud platforms. A workflow in DMCF can be defined using a visual or a script-based formalism, in both cases implementing a data-driven task parallelism that spawns ready-to-run tasks to Cloud resources. In this chapter, we presented how the DMCF workflow paradigm has been integrated with the MapReduce model. In particular, we described how VL4Cloud/JS4Cloud workflows can include MapReduce algorithms and tools, and how these workflows are executed in parallel on DMCF to enable scalable data processing on Clouds. Finally, we described a workflow application example that exploits the support to MapReduce provided by DMCF.

4

A scalable middleware for context-aware applications

Thanks to the large diffusion of mobile technologies and location-based services, it is possible to provide ubiquitous access to context-aware information. Mobile context-aware computing is a paradigm in which mobile applications can discover and take advantage of contextual information (e.g., date and time, user position, nearby users) [121, 39]. Some examples of context-aware mobile applications are: interactive trolleys to help shoppers finding groceries [15], monitors to remind medication for elderly [3], location-aware telephone call forwarding [142], and targeted advertisement based on social group information [2].

A core functionality of any context-aware ubiquitous system is storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments). The goal of this chapter is to design and provide a service-oriented middleware, called *Geocon*, which can be used by mobile application developers to implement such functionality. Geocon can be used to discover location-aware content, to share context-related information, and to facilitate interaction among users of mobile apps. Some examples of services that can be implemented in a mobile app using Geocon are: *i*) discovery of cultural places to be visited during a trip; *ii*) publication of user reviews about hotels and restaurants; *iii*) find nearby free-time activities of a user and his/her friends; *iv*) sharing of real-time information about events, traffic, etc.

A key benefit for developers using Geocon is the possibility to focus on the front-end functionality provided by their mobile application, without the need of implementing by scratch back-end components for data storing, indexing and searching, since they are provided by the middleware. In order to represent information about users, places, events and resources of mobile context-aware applications, Geocon defines a metadata model that can be extended to match most application requirements [11]. The widely-used JavaScript Object Notation (JSON) format is employed to represent such metadata. The architecture of the middleware includes a *geocon-service* that exposes methods for storing,

searching and selecting metadata about users, resources, events and places of interest, and a *geocon-client* library that allows mobile applications to interact with the service through the invocation of local methods. The interaction between service and client is based on the REST model.

Given the huge number of users, places, events and resources that may be involved in context-aware ubiquitous applications, scalability plays a fundamental role [100]. Geocon was designed to ensure scalability through the use of a NoSQL indexing and search engine, Elasticsearch [61], that can scale horizontally on multiple nodes as the system load increases. Elasticsearch may be used in combination with an external NoSQL database (e.g. MongoDB), which is more focused on constraints, correctness and robustness.

Compared to related work, Geocon is the only publicly available (and open source) Cloud-oriented system that provides a scalable middleware for context-aware mobile applications. Geocon was designed to be deployed on a public/private Cloud infrastructure, thus allowing an elastic resource allocation in a pay-per-use manner [132]. To assess the scalability of Geocon in a real case scenario, we used it to develop a location-aware mobile application, called *GeoconView*. The application allows users to share information about events exploiting the Geocon middleware for storing, indexing, and retrieving such information.

We evaluated the performance of Geocon varying the number of Cloud machines used to run the Geocon software components, the number of events stored in the middleware, and the number of queries submitted to the system. The Geocon software components have been deployed on Microsoft Azure, a public Cloud platform that provides on-demand computing and storage resources [99]. The experimental results show that the latency speedup is basically independent from the number of queries per second, but is significantly higher when the system stores a larger number of events (e.g., 2000k vs 500k). For instance, when the system stores 2000k events, with 15 queries per second, the latency speedup passes from 1.74 using 2 data nodes, to 4.74 using 8 data nodes.

The remainder of the chapter is organized as follows. Section 4.1 discusses related work. Section 4.2 describes the metadata model. Section 4.3 describes the middleware architecture and components. Section 4.4 describes the performance experiments carried out to assess the scalability of Geocon. Finally, Section 4.5 concludes the chapter.

4.1 Related Work

Several research projects and software systems have been proposed to support the implementation of context-aware mobile applications.

CRUMPET [115] (*Creation of User-Friendly Mobile Services Personalised for Tourism*) was a European research project aimed to deal with issues related to the mobility of tourists. In particular, the system provides tourists

with information filtered by mobile users' positions and interests. Yu and Chang [153] extended CRUMPET to overcome the limitations of handheld devices regarding the screen size and transmitting bandwidth.

The COMPASS system [136] (*Context-aware Mobile Personal Assistant*) was developed to provide users with relevant information and services. The relevance is determined exploiting information extracted from the user profile (e.g., preferences, interests, locations visited). COMPASS uses two search criteria for selecting relevant services/information: *i*) strict criteria, for discarding irrelevant results; and *ii*) soft criteria, for sorting and assigning a score to remaining results.

Driver and Clarke [43] proposed a framework to support the development of mobile trails-based applications. A trail is a scheduled collection of activities, such as to-do lists, that can be properly reordered when context change. The framework supports context-based activity schedule composition, identification of whether or not schedule reordering is required following context change and subsequent automatic schedule reordering as appropriate.

MobiSoC [64] is a service-oriented middleware for capturing, managing, and sharing the social state of physical communities. This state is composed of people profiles, place profiles, people-to-people affinities, and people-to-places affinities. The middleware provides real-time recommendations about people, places, and events, and delivers customized information based on users' geo-social context. The latency time of MobiSoC has been evaluated varying the query type and the number of users, showing the limitation of having a fixed number of computing nodes.

Context Toolkit [40] is a framework designed to support the development of context-aware applications. It consists in a set of widgets, which are software components with a common interface used to separate applications from context acquisition issues. The toolkit provides developers different components responsible for acquiring, aggregating and interpreting context information.

CaMWAF [88] is a framework designed to support the development of context-aware mobile applications and simplify the exchange of context information in heterogeneous environments. It allows developers to easily create cross-platform context-aware applications using common web technologies (e.g., HTML5, CSS3, JavaScript). To deal with the resource limitation of mobile devices, CaMWAF delegates the execution of intensive tasks to the server.

Malcher et al. [91] proposed a client middleware for developing context- and location-aware applications with capabilities of data sharing, dynamic deployment of new components, and combination of basic collaboration services. Given its client-side approach, the middleware does not take into account the server side architecture and related scalability issues.

SALES [31] is a middleware for contextual data dissemination in heterogeneous wireless communication networks. It proposes a hierarchical distributed architecture, some caching techniques for reducing context data traffic (e.g., a locality-based policy to speed up accesses to context data strictly related with locality), and two models for representing data (i.e., key-value and object-

based model). Concerning data representation, the key-value model allows reducing management overhead, especially in terms of required bandwidth, while the object-oriented model facilitates development by supporting extendibility.

CARISMA [23] is a mobile computing middleware that exploits the reflection principle for enhancing the development of adaptive and context-aware mobile applications. It provides developers with a set of primitives for describing how context changes should be handled using policies. Such policies use a micro-economic approach, which relies on a particular type of sealed-bid auction to take decision during application execution.

EgoSpaces [72] is an agent-based middleware for developing applications in ad-hoc mobile environments. It proposes an agent-centered notion of context, called a view, which is a collection of relevant data (or context). Each agent can operate over multiple views (which can be redefined over time as needs change) that include data/resources associated with the agent.

Table 4.1 summarizes the main features of the related systems discussed above, in comparison with Geocon's features. For each system, the table indicates: (*i*) the main goal of the system; (*ii*) whether or not the system was designed to be deployed on the Cloud; (*iii*) whether or not the server side focuses on scalability; (*iv*) whether or not the system is publicly available.

System	**Goal**	**Cloud**	**Scalable**	**Publicly available**
CRUMPET [115]	Content filtering based on mobile user's position	No	No	No
COMPASS [136]	Services and information filtering on user's preferences and interests	No	No	No
Driver and Clarke [43]	Context-aware management of user activities	No	No	No
MobiSoC [64]	Middleware for developing mobile social applications	No	Yes	Yes
Context Toolkit [40]	Composition of widgets for accessing context information	No	No	Yes
CaMWAF [88]	Context-aware applications using HTML5, CSS3 and JS	No	No	No
Malcher et al. [91]	Client middleware for local and remote data exchange	No	No	No
SALES [31]	Contextual data dissemination in heterogeneous wireless networks	No	Yes	No
CARISMA [23]	Context-aware applications exploiting reflection and micro-economic policies	No	Yes	No
EgoSpaces [72]	Agent-based middleware for applications in ad-hoc mobile environments	No	No	Yes
Geocon	Scalable middleware for context-aware mobile applications	Yes	Yes	Yes

Table 4.1. Comparison with related systems.

As shown in the table, Geocon is the only system explicitly designed for a Cloud, with a set of back-end components for data storing, indexing and searching, that can be easily deployed on any public/private Cloud infrastructures. Moreover, Geocon provides ad hoc scalability mechanisms, which are fundamental to provide satisfactory services as the amount of users and/or data to be managed grow. The test results demonstrate that Geocon scales well, thus allowing the development of mobile applications with a large number of users. It is worth noticing that three related works ([64] [31] [23]) highlight the importance of the scalability problem, but do not provide experimental evaluations on this aspect.

In summary, Geocon is the only publicly available (and open source) Cloud-oriented system that provides a scalable middleware for context-aware mobile applications. An important added value, not highlighted in the table, is the methodology provided by Geocon that defines an expendable metadata model supported by scalable set of back-end components for data storing, indexing and searching. This allows developers to focus on the front-end functionality provided by their mobile applications, without worrying on low-level back-end aspects and scalability issues that are managed transparently by Geocon.

4.2 Metadata Model

We defined a metadata model for representing information about users, places, events and resources of mobile context-aware applications. The model identifies a number of categories for indexing items in the domain of interest, which are generic enough to satisfy most of the application contexts. In particular, the metadata model is divided into four categories:

- *User*: defines basic information about a user (e.g., name, surname, e-mail).
- *Place*: describes a place of interest (e.g., square, restaurant, airport), including its geographical coordinates.
- *Event*: describes an event (e.g., concert, exhibition, conference), with information about time and location.
- *Resource*: defines a resource (e.g., photo, video, web site, web service) associated to a given place and/or event, including its Uniform Resource Identifier (URI).

Tables 4.2-4.5 present the basic metadata fields for each of the four categories listed above. Metadata are meant to be extensible, i.e., it is possible to include additional fields based on the specific application. For example, the user schema may be extended to include birth date, city, linked social network accounts, etc.

To represent metadata, the *JavaScript Object Notation* (JSON) is used. JSON is a widely-used text format for the serialization of structured data that is derived from the object literals of JavaScript [44]. Figure 4.1 shows

Table 4.2. Basic User metadata.

Name	Type	Description
id	String	Unique user identifier
name	String	Given name
surname	String	Family name
email	String	E-mail
token	String	Authentication token

Table 4.3. Basic Place metadata.

Name	Type	Description
id	String	Unique place identifier
name	String	Name of the place
description	String	Textual description of the place
latitude	Real	Latitude of the place
longitude	Real	Longitude of the place
address	String	Full address of the place
user_id	String	Id of the user who created the place

Table 4.4. Basic Event metadata.

Name	Type	Description
id	String	Unique event identifier
name	String	Name of the event
description	String	Textual description of the event
start_date	String	Date and time when the event begins
end_date	String	Date and time when the event ends
place_id	String	Id of the place where the event is held
user_id	String	Id of the user who created the event

Table 4.5. Basic Resource metadata.

Name	Type	Description
id	String	Unique resource identifier
name	String	Name of the resource
description	String	Textual description of the resource
URI	String	Link to the resource
place_id	String	Id of the place to which the resource is associated
event_id	String	Id of the event to which the resource is associated
user_id	String	Id of the user who created the resource

an example of JSON metadata describing a User. Beyond the basic metadata (id, name, etc.), it includes some additional fields (city, linked accounts and food preferences).

Figure 4.2 shows an example of Place metadata, regarding the "Kabuki" restaurant in Washington, DC, USA, which is tagged as a Japanese and sushi specialties restaurant using an additional "tags" field.

```
{ "id": "jdoe",
"name": "John",
"surname": "Doe",
"email": "john.doe@example.com",
"token": "19800308",
"city": "New York, NY, USA",
"linked-accounts": [
{"name":"facebook", "token":"424911363"},
{"name":"google", "key":"23467223454"}
],
"food-preferences": ["sushi", "pizza"],
"date-created": "2016-03-27T08:05:43.511Z"}
```

Fig. 4.1. Example of User metadata in JSON.

```
{ "id": "534",
"name": "Kabuki",
"description": "Japanese Restaurant",
"latidude": "38.897683",
"longitude": "-77.006081",
"address": "Union Station 50, Washington, DC, USA",
"user_id": "jdoe",
"tags": ["Japanese", "sushi"]}
```

Fig. 4.2. Example of Place metadata in JSON.

4.3 Middleware

This section describes the software components of the Geocon middleware and how these components are deployed within a distributed architecture.

4.3.1 Software components

Figure 4.3 shows the software structure of the middleware, which includes two main components:

- *geocon-service*, which contains a central registry for indexing users, resources, events and places of interest, and exposes methods for storing, searching and selecting metadata about these entities.
- *geocon-client*, which is a client-side library allowing mobile applications to interact with *geocon-service* through the invocation of local methods.

The interaction between service and client is based on the REST model [118]. To this end, a complete support to *CRUD* (Create, Read, Update, and Delete) operations on the metadata has been defined through Java APIs.

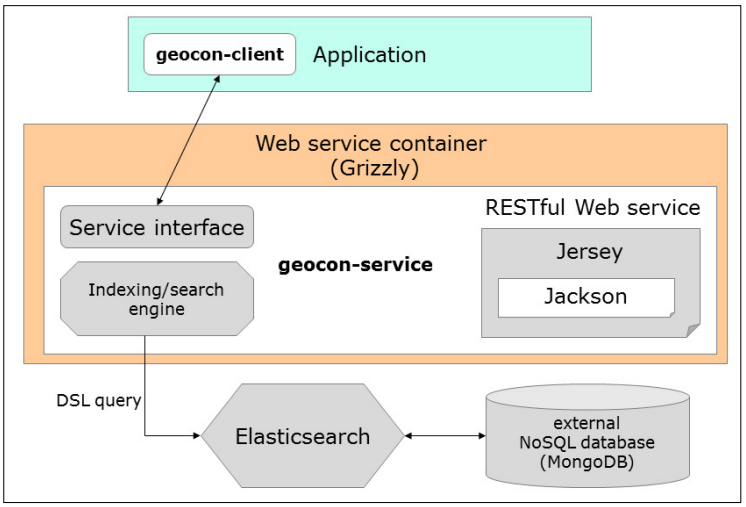


Fig. 4.3. Software components of the Geocon middleware.

Geocon-service

The geocon-service has been implemented as a RESTful Web service and exposed via the Web service container Grizzly[1], which is deployed on a distributed platform to ensures scalability as discussed later in this section.

The framework used in our implementation to develop RESTful Web services is *Jersey*[2], an open source framework that implements JAX-RS (Java API for RESTful Web Services) using annotations to map a Java class to a Web resource, and natively supports JSON representations through the integrated library *Jackson*[3].

The core component of *geocon-service* is the indexing and search engine, which has been implemented using Elasticsearch[4]. Elasticsearch is an open-source, distributed, scalable, and highly available search server based on Apache Lucene[5], and provides a RESTful web interface. Elasticsearch has been chosen because of several benefits, including:

- it is document-oriented, which means that entities can be structured as JSON documents;
- it is schema-free, which means it is able to detect the data structure automatically without need to specify a schema before indexing documents;

[1] `https://grizzly.java.net`
[2] `http://jersey.java.net/`
[3] `http://jackson.codehaus.org/`
[4] `https://www.elastic.co/`
[5] `https://lucene.apache.org/`

- it is horizontally scalable: if more power is needed, other nodes can be added and Elasticsearch will reconfigure itself automatically;
- it has APIs for several programming languages, including Java, which makes it easily integrable with other systems.

Geocon-service uses the query language provided by Elasticsearch, which is a full Query DSL (*Domain Specific Language*) based on JSON. Therefore, queries can be defined through the following main commands:

- *term*: returns all the documents whose specified field contains a given term. The following example returns all the documents whose field *name* contains the word "Mary":

```
{"term" : { "name" : "Mary" }}
```

- *prefix*: returns all the documents whose specified field contains a term beginning with a given prefix. The following example returns all the documents whose field *surname* begins with "Ro":

```
{"prefix" : { "surname" : "Ro" }}
```

- *bool*: returns all the documents containing a boolean combination of queries. It is built using one or more boolean clauses (i.e., *must*, *must_not*, *should*, and the parameter *minimum_should_match* that is the minimum number of clauses to be met). The following example returns all the users whose *name* is "Mary", that are not between 10 and 20 years old, and that like eating sushi or pizza:

```
{"bool" : {
"must" : { "term" : { "name" : "Mary" } },
"must_not" : {
"range" : {"age" : { "from" : 10, "to" : 20 }}
},
"should" : [
{ "term" : { "food-preferences" : "sushi" } },
{ "term" : { "food-preferences" : "pizza" } }
],
"minimum_should_match" : 1
}}
```

- *filter*: returns all the documents filtered according to a given condition. The following example returns all the documents whose field *location* falls within 50km from the center of Los Angeles sorting by distance.

```
{"query": {
"filtered" : {
"filter" : {
"geo_distance" : { "distance" : "50km",
"location" : {"lat" : 34.052235, , "lon" : -118.243683
}}}}},
"sort": [{
"_geo_distance": {
```

```
"location": { "lat" : 34.052235, "lon" : -118.243683},
"order": "asc", "unit": "m", "distance_type": "plane"}}]
}
```

Geocon-client

Geocon-client is the library used by mobile applications to interact with *geocon-service*. The library aims to facilitate communication with the *geocon-service* methods, hiding some low-level details (e.g., authentication, REST invocation, etc.) and providing users with a complete set of functions for executing CRUD operations. These functions are implemented using a set of objects and methods provided by the client library to the application layer.

Geocon-client consists of five classes: four classes are used to represent the metadata categories (*User*, *Place*, *Event* and *Resource*), while a fifth class (*SearchEngine*) is used to expose the methods for storing and searching data on *geocon-service*. For each class representing a metadata category, the *SearchEngine* class provides a set of CRUD methods: *register*, *get*, *update*, and *delete*. As an example, Table 4.6 shows the CRUD methods provided to register, get, update and delete Resource elements in the service.

Table 4.6. CRUD methods for Resource elements.

Method	Description
`register (Resource r)`	Registers a resource to the service
`get (Resource r)`	Returns the metadata of a resource
`update (Resource r)`	Updates the metadata of a resource
`delete (Resource r)`	Deletes a resource

4.3.2 Distributed architecture

This section describes how the Geocon components are deployed within a distributed architecture. As described above, Geocon exploits Elasticsearch (ES) as indexing and search engine that can scale horizontally on a very large number of nodes as the system load increases. ES implements a clustered architecture that uses partitioning to distribute data across multiple nodes, and replication to provide high availability.

There are three types of ES nodes: *i*) *ES Data nodes*, which can hold one or more partitions containing index data; *ii*) *ES Client nodes*, that do not hold index data but handle incoming requests made by client applications to the appropriate data node; and *iii*) *ES Master node* that performs cluster management operations, such as maintaining routing information, coordinating recovery after node failure, relocating data partitions among nodes.

As shown in Figure 4.4, four types of nodes are present in the distributed Geocon architecture:

- *Mobile devices*, which interact with the geocon-service using the geocon-client library.
- *Load Balancer*, which evenly distributes requests from mobile devices to a pool of server nodes.
- *Server nodes*, which are a pool of virtual machines handling the mobile devices' requests. Each request is managed by a geocon-service instance that translates it into an Elasticsearch query. The query is processed by an ES Client node that interacts with the appropriate ES Data nodes. One of Server nodes hosts the ES Master node.
- *Data nodes*, which are a pool of virtual machines running the ES Data nodes, which process the queries upon request of the ES Client nodes.

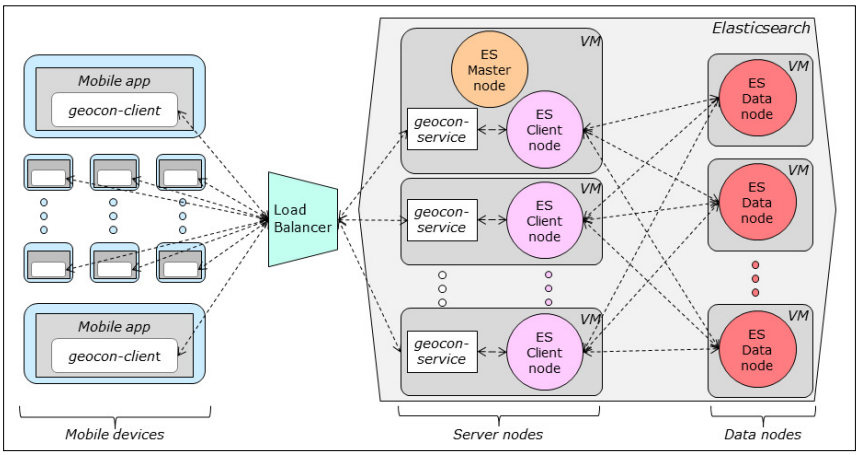


Fig. 4.4. Distributed architecture of the Geocon middleware.

4.4 Performance evaluation

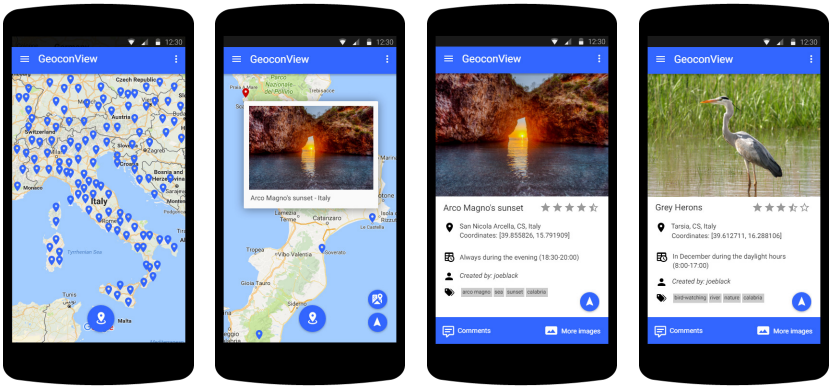
An experimental performance evaluation was carried out to assess the scalability of the Geocon middleware in a real case scenario. The section is structured into three parts: *i*) introduction to the mobile application used as real case; *ii*) description of experimental setup and performance parameters; *iii*) presentation and discussion of the performance results.

4.4.1 Mobile application

In order to assess the scalability of Geocon in a real case scenario, we developed a location-aware mobile application, called *GeoconView*, which allows users to

share information about events, exploiting the Geocon middleware for storing, indexing, and retrieving such information. More in detail, a GeoconView user can share events characterized by the following features: (*i*) *place*: the place where the event will happen; (*ii*) *images*: one or more photos representing the event; (*iii*) *datetimes*: the range of dates and times when the event will occur; (*iv*) *tags*: a set of keywords describing the event; and (*v*) *comments*: user comments and ratings about the event.

Figure 4.5 shows some screenshots of GeoconView. In particular, Figure 4.5(a) visualizes a number of GeoconView's events on a map. Figure 4.5(b) shows a preview of the *Arco Magno's sunset*, a daily event that occurs on a beach in San Nicola Arcella (Italy). Figure 4.5(c) provides full details about this event. Figure 4.5(d) shows a second event describing the arrival of grey herons in Tarsia (Italy) in December.



(a) Geotagged events on a map. (b) Arco Magno's sunset (preview). (c) Arco Magno's sunset (details). (d) Grey herons arrival (details).

Fig. 4.5. GeoconView: A location-aware mobile application based on the Geocon middleware.

To implement the GeoconView application, the basic Event metadata scheme has been extended with additional fields to store URLs of images, descriptive tags, and information about the periodicity (e.g., weekly, monthly, yearly) of the events. Comments and ratings associated to the events have been stored as Resource metadata instances.

4.4.2 Experimental setup and performance parameters

The distributed architecture used for the evaluation is composed of 9 cloud machines hosted by the Microsoft Azure platform. Each machine is equipped with a single-core 1.66 GHz CPU, 3.5 GB of memory, and 50 GB of disk space.

Table 4.7 shows the system parameters that have been used during the evaluation. As shown in the table, we used from 1 to 8 Data nodes, each one running on a separate cloud machine. An additional cloud machine was used to run a Server node. The number of geo-located events stored in the Geocon middleware ranges from 250k to 2000k. A varying number of queries per second (from 15 to 120) was submitted to the system, so as to evaluate its performance under different load levels. Every query asks for the ten active events that are closest to a location that changes randomly from query to query.

For executing the load tests and measuring the performance of the system we used Apache JMeter[6]. The following performance parameters have been evaluated:

- *Latency time*: the average amount of time elapsed from query submission to query answer;
- *Speed-up*: the ratio of the latency time using 1 data node to the latency time using n data nodes, which measures how much performance gain is achieved by distributing data over an increasing number of cloud machines;
- *Scale-up*: the latency time when the problem size is increased linearly with the number of data nodes, which quantifies the capability of the system to handle increasing loads when machines are added to accommodate that growth.

4.4.3 Performance results

Figure 4.6 shows how the latency time changes using a fixed number of data nodes and varying the number of events stored and the number of queries per second submitted to the system. Figure 4.6(a) presents the results obtained using 2 data nodes. For the smallest number of events (250k), the latency time increases from 0.078 seconds with 15q/s, to 0.373 seconds with 120 q/s. For the largest number of events (2000k) the latency time ranges from 0.457 seconds to 1.651 seconds. Figure 4.6(b) shows the results obtained using 8 data nodes. For 250k events, the latency time ranges from 0.063 seconds with 15q/s, to 0.300 seconds with 120 q/s, while for 2000k events the latency time increases from 0.168 seconds to 0.621 seconds. In both cases, the latency time increases linearly with the number of requests per second, independently from the number of events stored in the system.

Table 4.7. System parameters.

Description	Values
Number of data nodes	1, 2, 4, 8
Number of events	250k, 500k, 1000k, 2000k
Queries per second	15q/s, 30q/s, 60q/s, 120q/s

[6] http://jmeter.apache.org/

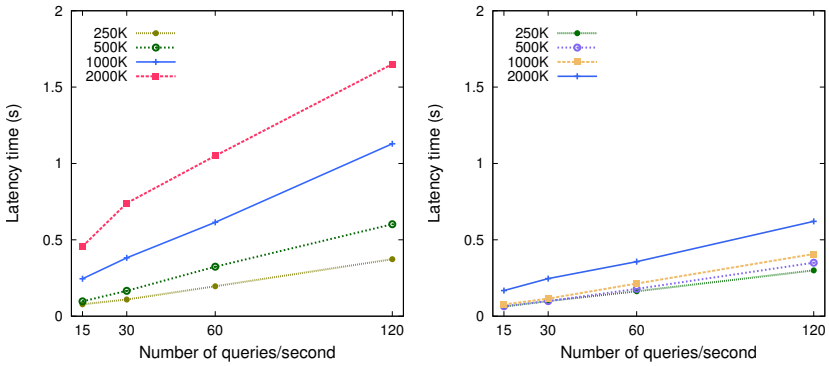


Fig. 4.6. Latency time vs number of queries per second, for different numbers of events stored in the system, using: a) 2 data nodes; b) 8 data nodes.

The scalability of Geocon can be evaluated through Figure 4.7, which shows the speedup obtained varying the number of data nodes and the number of queries per second submitted to the system.

Figure 4.7(a) presents the results obtained with 500k events stored in the system. With a load of 15 queries per second, the speedup increases from 1.54 using 2 data nodes, to 2.29 using 8 data nodes. With the highest load (120 q/s) the speedup passes from 1.62 using 2 nodes, to 2.79 using 8 nodes. Figure 4.7(b) shows the results obtained when the number of events is increased to 2000k. With 15 queries per second, the speedup passes from 1.74 using 2 data nodes, to 4.74 using 8 data nodes. With 120 queries per second, the speedup passes from 1.45 using 2 nodes, to 3.86 using 8 nodes. As expected, the speedup is basically independent from the number of queries per second, but is significantly higher when the system stores a larger number of events (e.g., 2000k vs 500k).

Figure 4.8 measures the application scaleup by showing the latency times obtained when the number of events stored in the system increases proportionally to the number of data nodes used (i.e., from 250k events stored on 1 data node, to 2000k events stored on 8 data nodes). The results show that, for any number of queries per second submitted to the system, the latency time is almost constant. This demonstrates that the amount of data that can be managed increases, almost linearly, with the number of data nodes available.

4.5 Conclusions

Geocon is a service-oriented middleware designed to help developers to implement context-aware mobile applications. Geocon provides a service and a

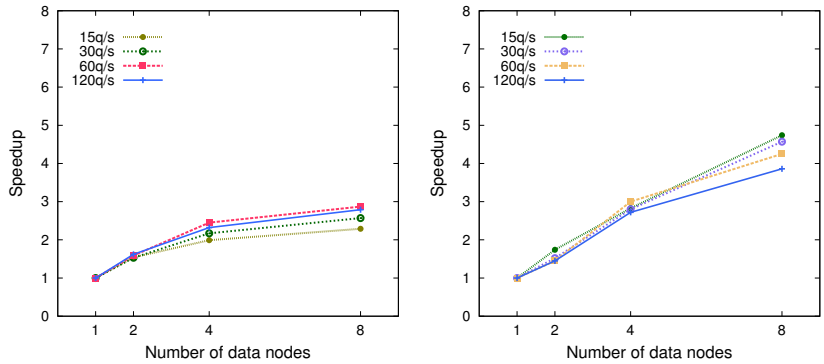


Fig. 4.7. Speedup vs number of data nodes, for different numbers of queries per second, using: a) 500k events; b) 2000k events.

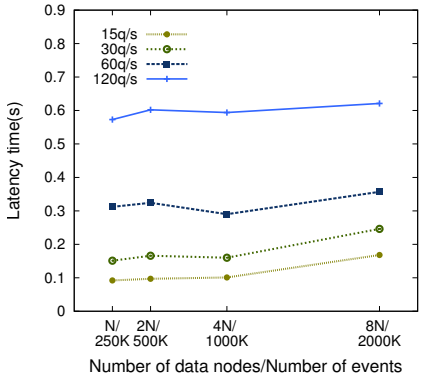


Fig. 4.8. Latency time vs number of data nodes/number of events (scaleup), for different numbers of queries per second.

client library for storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments). A key benefit for developers using Geocon is the possibility to focus on the front-end functionality provided by their mobile application, without the need of implementing by scratch back-end components for data management and querying, which are provided by the middleware.

Geocon defines a metadata model to represent information about users, places, events and resources of mobile context-aware applications, which can be easily extended to match specific application requirements. In order to

ensure a high level of decoupling and efficient communication between client and service, the REST model has been adopted. Moreover, given the huge number of users, places, events and resources that may be involved in context-aware mobile applications, Geocon uses the Elasticsearch engine that can scale horizontally on a multiple nodes.

To assess the scalability of Geocon in a real-world scenario, we developed a location-aware mobile application, called *GeoconView*. The application allows users to share information about events exploiting the Geocon middleware for storing, indexing, and retrieving such information. The experimental results show that the latency speedup is basically independent from the number of queries per second, but is significantly higher when the system stores a larger number of events (e.g., 2000k vs 500k). For instance, when the system stores 2000k events, with 15 queries per second, the latency speedup passes from 1.74 using 2 data nodes, to 4.74 using 8 data nodes. The Geocon middleware is available as open-source software at `https://github.com/SCAlabUnical/Geocon`.

As future works

5

Analysis of geotagged social data

The widespread use of social media makes it possible to extract very useful information to understand the behavior of large groups of people. This is fostered by the large use of mobile phones and location-based services, through which millions of people every day access social media services and share information about the places they visit. In fact, data gathered from social media, such as posts from Twitter and Facebook or photos from Instagram and Flickr, are frequently geotagged. Geotagging is the process of adding geographic metadata (e.g., longitude/latitude coordinates) to text, photos or videos. It allows to locate the exact physical origin of shared information. Exploiting geotagged social data it is possible to extract high-value information that may impact many areas, such as travel recommendations, urban planning, intelligent traffic management, health monitoring, and security.

One of the leading trends in social media research is the analysis of geotagged data to determine if users visited or not interesting locations (e.g., touristic attractions, shopping malls, squares, parks), often called Places-of-Interest (PoIs). Since a PoI is generally identified by the geographical coordinates of a single point, it is hard to match it with user trajectories. For this reason, it is useful to define the so-called *Region-of-Interest* (*RoI*) representing the boundaries of the PoI's area [62].

The analysis of user trajectories through RoIs is highly valuable in many scenarios, e.g.: tourism agencies and municipalities can discover the most visited touristic places and the time of year when such places are visited [13][81]; transport operators can discover the places and routes where is it more likely to serve passengers [154] and crowed areas where more transport facilities need to be allocated [152]. However, compared to trajectory pattern mining from GPS data [58], extracting trajectories from social network data is more challenging because data from social networks are often sparse and irregular, in contrast to GPS traces of mobile devices that are highly available and sampled at regular time intervals [132].

RoI mining techniques are aimed at discovering Regions-of-Interest from PoIs and other data. Existing RoI mining techniques can be grouped into

three main approaches: *predefined shapes* [62], *density-based clustering* [156] and *grid-based aggregation* [19]. Predefined shapes techniques use fixed shapes, such as circles or rectangles, to represent RoIs. In many cases, the use of a predefined shape represents a naïve solution to the RoI mining problem, because a predefined shape is not able to handle PoIs having RoIs with different sizes and shapes. Density-based clustering techniques identify RoIs by clustering the data points according to a density criterion (i.e., number of data points per unit area). Such kind of algorithms are widely used because they are able to reach good results in many cases. However, density-based techniques may fail to distinguish regions that are very close to each other or that have different density. Grid-based aggregation techniques discretize the area in a regular grid and then aggregate the grid cells so as to form a RoI. The grid cells can be aggregate using different aggregation policies. Such kind of algorithms is very sensitive to parameters setting. Thus, may be hard to find a setting for identifying multiple RoIs with different characteristics in the same area.

This chapter presents a novel RoI mining technique, called *G-RoI*, which differs from the existing approaches mentioned earlier as it exploits the indications contained in geotagged social media items (e.g. tweets, posts, photos or videos with geospatial information) to discover the RoI of a PoI with a high accuracy. Given a PoI p identified by a set of keywords, a geotagged item is associated to p if its text or tags contain at least one of those keywords. Starting from the coordinates of all the geotagged items associated to p, *G-RoI* calculates an initial convex polygon enclosing all such coordinates, and then iteratively reduces the area using a density-based criterion. Then, from all the convex polygons obtained at each reduction step, *G-RoI* adopts an area-variation criterion to choose the polygon representing the RoI for p.

Many experiments have been performed to assess the accuracy of G-RoI over real geotagged items extracted from Flickr, one of the most popular photo-sharing social media. The experimental results show that G-RoI is more accurate in identifying RoIs than existing techniques. Over a set of 24 PoIs in Rome, G-RoI achieves better results than existing techniques in 19 cases, with a mean precision of 0.78, a mean recall of 0.82, and a mean F_1 score of 0.77. In particular, the mean F_1 score of G-RoI is 0.34 higher than that obtained with the well-known DBSCAN algorithm. Further experiments have been performed over a set of 24 PoIs in Paris. Also in this case, G-RoI achieved best results in 18 cases, with a mean precision of 0.81, a mean recall of 0.66, and a mean F_1 score of 0.70 (0.23 higher than that obtained with DBSCAN). For the purpose of reproducibility, an open-source version of G-RoI and all the input data used in the experiments are available at `https://github.com/scalabunical/G-RoI`.

The remainder of the chapter is organized as follows. Section 5.1 introduces the main concepts and the problem statement. Section 5.2 discusses related work. Section 5.3 describes the proposed methodology. Section 5.4 compares the performance of G-RoI with the main techniques in literature. Finally, Section 5.5 concludes the chapter.

5.1 Problem definition

A *Place-of-Interest* (*PoI*) is a specific location that someone finds useful or interesting. Generally, PoIs refer to business locations (e.g., shopping malls) or tourist attractions (e.g., squares, museums, theaters, bridges). PoIs are also named as *Point-of-Interest*.

For analyzing users' behavior, it is useful to understand whether a user visited or not a PoI. Since information on a PoI is generally limited to an address or to GPS coordinates, it is hard to match trajectories with PoIs. For this reason, it is useful to define the so-called *Region-of-Interest* (*RoI*) representing the boundaries of the PoI's area [62].

RoIs can be defined as "spatial extents in geographical space where at least a certain number of user trajectories pass through" [58]. Thus, RoIs represent a way to partition the space into meaningful areas and, correspondingly, to associate a label to a place. In literature, RoIs are also named as *regions of attraction* [156] or *frequent (dense) regions* [7].

A *geotagged item* is a piece of information (e.g. tweet, post, photograph or video) to which geospatial information were added. Specifically, a geotagged item g includes the following features:

- *text*, containing a textual description of g.
- *tags*, containing the tags associated to g.
- *coordinates* consists of *latitude* and *longitude* of the place from where p was created.
- *userId*, identifying the user who created g.
- *timestamp*, indicating date and time when g was created.

A geotagged item can be associated to a PoI $\mathcal{P}$ if its text or tags refer to $\mathcal{P}$. The goal of G-RoI is finding a suitable RoI $\mathcal{R}$ that describes the boundaries of $\mathcal{P}$'s area, by analyzing a set of geotagged items associated to $\mathcal{P}$.

5.2 Related work

Existing techniques to find RoIs can be grouped into three main approaches: *predefined shapes*, *density-based clustering* and *grid-based aggregation*. Table 5.1 reports approaches, algorithms, and goals of the main related work.

Predefined shapes. This approach uses predefined shapes (circles, rectangles, etc.) to represent RoIs. For example, Kisilevich et al. [75] define RoIs as circles of fixed radius centered on a set of PoIs whose center coordinates are known. Spyrou and Mylonas [126] used circular RoIs to extract popular touristic routes from Flickr. Specifically, circular shapes are used to translate a trajectory of geospatial points into a sequence of RoIs. Cesario et al. [25] used rectangles to define RoIs representing stadiums for a trajectory mining study. In particular, the RoI of a stadium is the smallest rectangle enclosing

the stadium's area. De Graaff et al. [62] use Voronoi tessellations [138] to define RoIs starting from a set of geographical coordinates representing PoIs.

Density-based clustering. With this approach, RoIs are obtained by clustering a set of geographical locations. For instance, Crandall et al. [32] used the Mean shift clustering algorithm [29] to group the locations of a set of Flickr photos. The RoI is the polygon enclosing the cluster points. Zheng et al. [156] used DBSCAN [47] to discover tourist attraction areas from a set of Flickr photos. DBSCAN was adopted for three main reasons: *i*) it tends to identify regions of dense data points as clusters; *ii*) it supports clusters with arbitrary shape; *iii*) it has a good efficiency on large-scale data. DBSCAN was also used by Altomare et al. [7], with the goal of detecting the regions that are more densely visited based on data from GPS-equipped taxis. Kisilevich et al. [76] used a variant of DBSCAN, named P-DBSCAN, to cluster photos taking into account the neighborhood density (i.e., the number of distinct photo owners in the neighborhood) and exploiting the notion of adaptive density for fast convergence towards high density regions. Density-based approaches need a method to assign a meaning to each RoI found. There are different ways to perform this task. Zheng et al. [156] and Yin et al. [151] assign a name to each cluster by taking the most frequent keyword in the geotagged items. Ferrari et al. [49] automatically associate to each RoI the zip code of the data points in the cluster center.

Grid-based aggregation. This approach discretizes the area under analysis in a regular grid and extract RoIs by aggregating the grid cells. For example, Giannotti et al. [58] divide an area into grid cells and then count the trajectories passing through each cell. Grid cells whose counters are above a certain threshold are expanded to form rectangular shaped RoIs. Cai et al. [19] argued that rectangular expansion produces RoIs that may contain uninteresting low-density cells. For this reason, they proposed a hybrid grid-based algorithm, called Slope RoI, to mine arbitrary RoI shapes from trajectory data. Cesario et al. [26] split the EXPO 2015 area in a grid and associated grid cells to PoIs representing pavilions, in order to discover the behavior and mobility patterns of users inside the exhibition. Shi et al. [124] map geotagged data into grid cells, and then group the cells taking into account spatial proximity and social relationship between places.

The proposed G-RoI technique does not belong to the approaches described earlier and it differs from them in three main respects:

- Differently from approaches using predefined shapes, G-RoI defines RoIs as polygons that are more accurate to model the variety of shapes a PoI can have.
- Density- and grid-based approaches may have troubles in distinguishing RoIs associated to PoIs that are very close to each other [19]. In fact, these approaches cluster data points (or aggregate cells) based on their proximity, even if they belong to different PoIs that are close to each other. As a result, two or more adjacent PoIs may be associated to the

Table 5.1. Comparison with related algorithms.

Related work	Approach	Algorithm	Goal
Kisilevich et al. [75]	Pred. shapes	Circle with fixed radius	Mine travel sequences from Flickr photos
Spyrou-Mylonas [126]	Pred. shapes	Circle with fixed radius	Extract popular touristic routes from Flickr photos
Cesario et al. [25]	Pred. shapes	Rectangle enclosing PoIs	Trajectory mining from Twitter data
De Graaff et al. [62]	Pred. shapes	Voronoi tessellations	RoI extraction from cadastral data
Crandall et al. [32]	Density	Mean shift clustering	Organize a large collection of geotagged Flickr photos
Zheng et al. [156]	Density	DBSCAN	Discover interesting places from Flickr photos
Altomare et al. [7]	Density	DBSCAN	Detect RoIs based on data from GPS-equipped taxis
Kisilevich et al. [76]	Density	P-DBSCAN	Discover attractive areas from collections of Flickr photos
Giannotti et al. [58]	Grid	Popular Regions	Mine rectangular RoI shapes from trajectory data
Cai et al. [19]	Grid	Slope RoI mining	Mine arbitrary RoI shapes from Flickr trajectory data
Cesario et al. [26]	Grid	Grid cell aggregation	Discover mobility patterns from Instagram photos
Shi et al. [124]	Grid	DCPGS-G	Mine RoIs from historical geo-social networks

same RoI. In contrast, G-RoI accurately identifies different RoIs even in the presence of adjacent PoIs, as demonstrated by the experimental results presented in Section 5.4.

- Density- and grid-based approaches algorithms strongly depend on parameters setting (e.g., *eps* and *minNumPoints* for DBSCAN, *cell size* and *minimum support* for Slope RoI), and it is hard to find parameters that produce accurate RoIs over multiple locations with a variety of shapes and data points distributions. In contrast, as demonstrated in Section 5.4, G-RoI is accurate in identifying RoIs over locations characterized by a large variety of shapes and data points distributions, using always the same value for its configuration parameter (a distance threshold between 0 and 1).

5.3 Methodology

Let a PoI $\mathcal{P}$ be identified by one or more keywords $K = \{k_1, k_2, ...\}$. Let G_{all} be a set of geotagged items. Let $G = \{g_0, g_1, ...\}$ be the subset of G_{all} associated to $\mathcal{P}$, i.e., the text or tags of each $g_i \in G$ contains at least one keyword in K. Let $C = \{c_0, c_1, ...\}$ be a set of coordinates, where c_i represents the coordinates of $g_i \in G$. Thus, every $c_i \in C$ represents the coordinates of

a location from which a user has created a geotagged item referring to $\mathcal{P}$. Let cp_0 be a convex polygon enclosing all the coordinates in C, obtained by running the convex hull algorithm [9] on C, described by a set of vertices $\{v_0, v_1, ...\}$.

To find the RoI $\mathcal{R}$ for $\mathcal{P}$, the G-RoI algorithm is composed by two procedures:

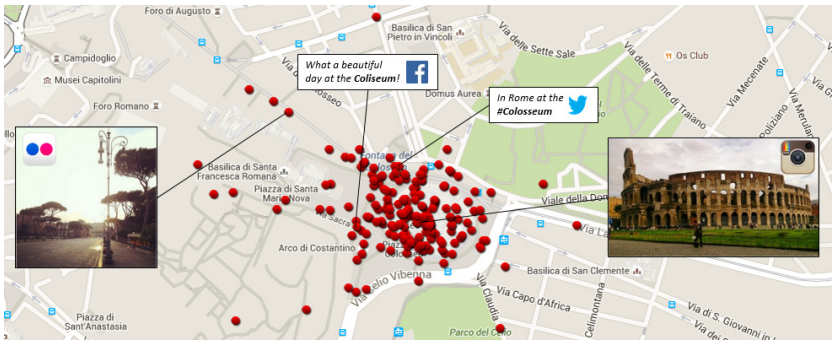
- *G-RoI reduction.* Starting from cp_0, it iteratively reduces the area of the current convex polygon by deleting one of its vertex. A density-based criterion is adopted to choose the next vertex to be deleted. The density of a polygon is the ratio between the number of geotagged items enclosed by the polygon, and its area. At each step, the procedure deletes the vertex that produces the polygon with highest density, among all the possible polygons. The procedure ends when it cannot further reduce the current polygon, and returns the set of convex polygons $CP = \{cp_0, ..., cp_n\}$ obtained after the n steps that have been performed.
- *G-RoI selection.* It analyses the set of convex polygons CP returned by the G-RoI reduction procedure, and selects the polygon representing RoI $\mathcal{R}$ for PoI $\mathcal{P}$. An area-variation criterion is adopted to choose $\mathcal{R}$ from CP. Given CP, the procedure identifies two subsets: a first subset $\{cp_0, ..., cp_{cut-1}\}$ such that the area of any cp_i is significantly larger than the area of cp_{i+1}; a second subset $\{cp_{cut}, ..., cp_n\}$ such that the area of any cp_i is not significantly larger than the area of cp_{i+1}. The procedure returns cp_{cut} as RoI $\mathcal{R}$. This corresponds to choosing cp_{cut} as the corner point of a discrete *L-curve* [65] obtained by plotting the areas of all the convex polygons in CP on a Cartesian plane, as detailed later in this section.

5.3.1 Example

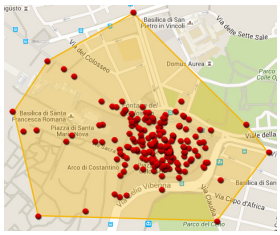
For the sake of clarity and for the reader's convenience, before going into algorithmic details, we describe how the two G-RoI procedures work through a real example. We collected a small sample of 200 geotagged items from different social networks (Flickr, Twitter, Instagram and Facebook), referring to the *Colosseum* in Rome and posted at a maximum distance of 500m from it.

In their posts and photos, the social network users identify the *Colosseum* with different keywords. The Geonames website[1] reports the names used in different languages to identify the Colosseum, such as *Coliseum*, *Coliseo*, *Colise*, and synonymous such as *Flavian Amphitheatre* or *Amphitheatrum Flavium*. All the geotagged items in our sample contain at least one of such keywords. From these items, the 200 coordinates shown in Figure 5.1(a) are extracted. Given the coordinates, the G-RoI reduction procedure calculates the initial convex polygon cp_0 (shown Figure 5.1(b)), and then iteratively reduces the

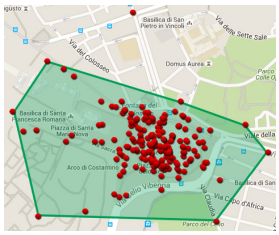
[1] http://geonames.org/



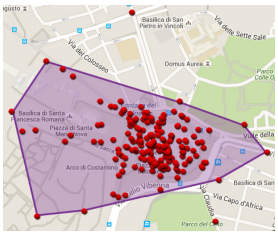
(a) Collection of geotagged items.



(b) Initial convex polygon cp_0.



(c) Generating cp_1 by deleting one vertex from cp_0.



(d) Generating cp_2 by deleting one vertex from cp_1.

Fig. 5.1. G-RoI reduction on Colosseum's geotagged items.

area. Figure 5.1(c) shows polygon cp_1 obtained after the first step by deleting one of the vertices from cp_0. Similarly, Figure 5.1(d) shows polygon cp_2 obtained after cp_1. The G-RoI reduction procedures iterates until it cannot further reduce the current polygon. The output of the procedure is the set of convex polygons $CP = \{cp_0, cp_1, ..., cp_n\}$ obtained at each step. Figure 5.2 shows with different colors all the convex polygons in CP, including the one chosen as RoI $\mathcal{R}$ by the subsequent G-RoI selection procedure.

The G-RoI selection procedure analyses CP to choose RoI $\mathcal{R}$ among all the convex polygons in it. To this end, the procedure extracts from CP an ordered set of Cartesian points $P = \{(0, A_0), (1, A_1), ..., (n, A_n)\}$.

An element $p_i \in P$ is a point (i, A_i), where i is the step in which cp_i was generated, and A_i is the area of cp_i. Figure 5.3(a) plots all the points in P in our example. The graph shows how much the area decreases with the steps performed by the G-RoI reduction procedure. The graph can be divided in two parts:

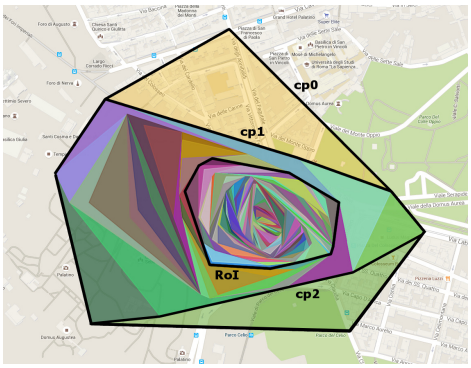


Fig. 5.2. Set of convex polygons in CP identified by the RoI reduction procedure, with indication of RoI $\mathcal{R}$ chosen by the RoI selection procedure.

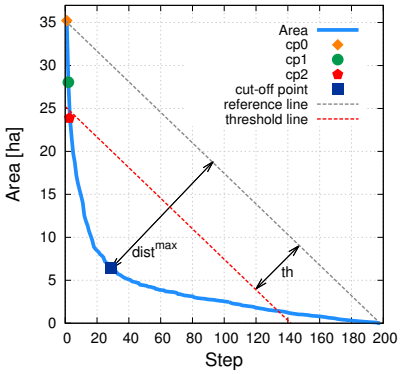
- The first part, from step 0 to a cut-off point p_{cut} (not included), decreases quickly, because at each step the G-RoI reduction procedure cuts a significant portion of area.
- The second part, from p_{cut} to step n, decreases slowly, because at each step the G-RoI reduction procedure cuts only a small portion of area.

The G-RoI selection procedure identifies the point p_{cut} that is located at the maximum distance ($dist^{max}$) from the *reference line* joining the first point and the last point under analysis (p_0 and p_n), as shown in Figure 5.3(a). If the set of points $\{p_{cut}, ..., p_n\}$ follows a linear trend as shown in Figure 5.3(b), i.e., there is no point below a *threshold line* at distance th from the reference line joining the points p_{cut} and p_n, then the procedure returns the polygon corresponding to p_{cut} as RoI $\mathcal{R}$ (see Figure 5.3(c)). Otherwise, the G-RoI selection procedure iterates by finding a new cut-off point from the set of points on the right of p_{cut}, as detailed in the next section.

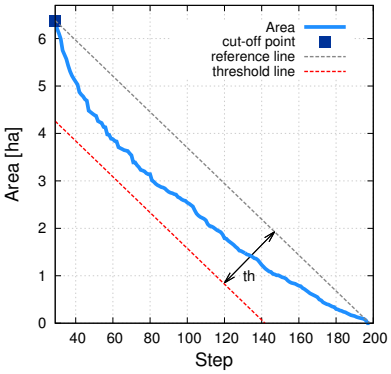
5.3.2 Algorithmic details

Algorithm 1 shows the pseudo-code of the G-RoI reduction procedure. The input is a set of coordinates C and the output is a set of convex polygons CP. Starting from C, the procedure calculates the initial convex polygon cp_0 (line 1). Then, cp_0 is added to CP and is taken as current convex polygon cp (lines 2-3). A do-while block performs the area reduction steps (lines 4-22). At each step, the area of the current convex polygon cp is reduced by deleting one of its vertices. The algorithm ends when it cannot further reduce cp.

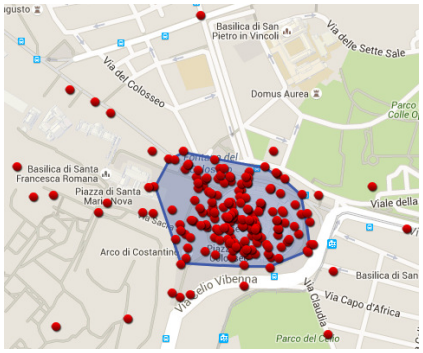
At the beginning of each reduction step, the current maximum density ρ^{max} is set to zero (line 5), while the convex polygon with maximum density cp^{max} and the vertex to be deleted v^{del} are initialized to null (lines 6-7). At each reduction step, for choosing the vertex to be deleted from cp, the



(a) Colosseum's points $0 \leq p_i \leq n$.



(b) Colosseum's points $cut \leq p_i \leq n$.



(c) Convex polygon corresponding to p_{cut}), chosen as RoI $\mathcal{R}$.

Fig. 5.3. G-RoI selection from Colosseum's convex polygons.

algorithm iterates (lines 8-17) on each vertex $v \in cp$ performing the following operations:

- creates a temporary set of coordinates C^{tmp} obtained by deleting v from C (line 9);
- calculates the convex polygon cp^{tmp} from C^{tmp} (line 10);
- calculates the area A^{tmp} of cp^{tmp} (line 11);
- if A^{tmp} is greater than zero (line 12), the density ρ^{tmp} of cp^{tmp} is calculated as the number of coordinates in C^{tmp} divided by A^{tmp} (line 13);
- if ρ^{tmp} is greater than ρ^{max} (line 14), ρ^{tmp} is assigned to ρ^{max} (line 15), cp^{tmp} is assigned to cp^{max} (line 16), and v is assigned to the vertex to be deleted v^{del} (line 17).

Algorithm 1: G-RoI reduction.

Input : Set of coordinates C
Output: Set of convex polygons CP

```
1  cp_0 ← convexHull(C);                          /* Initial convex polygon */
2  CP ← {cp_0};                                   /* Set of convex polygons */
3  cp ← cp_0;                                     /* Current convex polygon */
4  do
5      ρ^max ← 0;                                 /* Current maximum density */
6      cp^max ← ⊢;                    /* Convex polygon with density = ρ^max */
7      v^del ← ⊢;                                 /* Vertex to be deleted */
8      for v ∈ cp do
9          C^tmp ← C − v;
10         cp^tmp ← convexHull(C^tmp);
11         A^tmp ← Area(cp^tmp);
12         if A^tmp > 0 then
13             ρ^tmp ← |C^tmp| / A^tmp;
14             if ρ^tmp > ρ^max then
15                 ρ^max ← ρ^tmp;
16                 cp^max ← cp^tmp;
17                 v^del ← v;
18     if ρ^max > 0 then
19         CP ← CP ∪ {cp^max};
20         cp ← cp^max;
21         C ← C − v^del;
22 while ρ^max > 0;
23 return CP
```

After having iterated on all vertices, if ρ^{max} is greater than zero (i.e., at least one polygon was found) (line 18), the algorithm adds cp^{max} to CP (line 19), assigns cp^{max} to cp (line 20), and deletes v^{del} from C (line 21). Finally, when the current reduction step does not change ρ^{max}, and so it remains equal to zero, which means that the current convex polygon cannot be further reduced (line 22), the algorithm returns the set of convex polygons CP generated (line 23).

Algorithm 2 shows the pseudo-code of the *G-RoI selection* procedure. The input is a set of convex polygons CP and a threshold $th \in (0, 1)$. Given CP, the algorithm creates a set of Cartesian points P, where each point p_i is a pair (i, A_i), with i identifying the step in which cp_i has been generated and A_i representing the area of cp_i (lines 1-4).

Then, the index of the cut-off point *cut* is set to zero (line 5). At each iteration (lines 6-19) the algorithm tries to find a cut-off point p_{cut} that is at the maximum distance from the line $y = 1 - x$ (which links the first and last normalized points in CP), and which is located below the line $y = 1 - th - x$

(i.e., within a threshold distance th from the line $y = 1 - x$). Thus, at the beginning of each iteration, the maximum distance $dist^{max}$ is set to zero (line 7), and the index of the point with maximum distance i^{max} is set to cut (line 8).

Algorithm 2: G-RoI selection.

Input : Set of convex polygons CP; Threshold $th \in (0, 1)$
Output: Region of Interest $\mathcal{R}$.

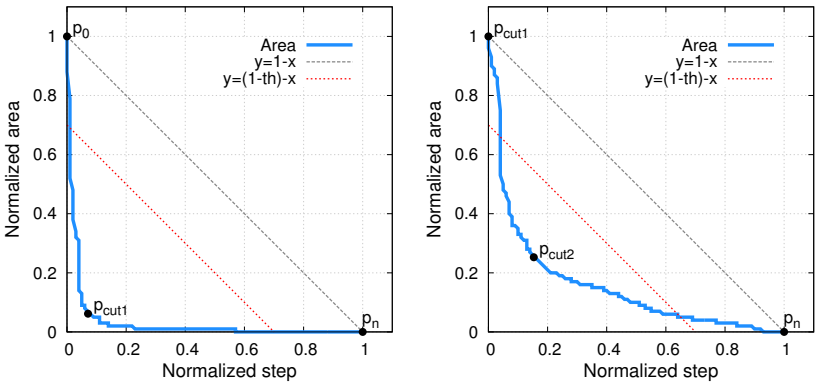
```
1  P ← ∅;                                              /* Set of Cartesian points */
2  for cp_i ∈ CP do
3  |  A_i ← Area(cp_i);
4  |  P ← P ∪ {(i, A_i)};
5  cut ← 0;                                           /* Index of the cut-off point */
6  do
7  |  dist^max ← 0;                                  /* Current maximum distance from y=1-x */
8  |  i^max ← cut;                                   /* Index of the point with dist^max */
9  |  for i ← cut + 1 to n − 1 do                     /* Where n = |CP| − 1 */
10 |  |  x^norm = (P_i.x − P_cut.x)/(P_n.x − P_cut.x);
11 |  |  y^norm = (P_i.y − P_n.y)/(P_cut.y − P_n.y);
12 |  |  if y^norm < 1 − th − x^norm then
13 |  |  |  dist^tmp = (1 − y^norm − x^norm) · √2/2;
14 |  |  |  if dist^tmp ≥ dist^max then
15 |  |  |  |  dist^max ← dist^tmp;
16 |  |  |  |  i^max ← i;
17 |  |  if dist^max > 0 then
18 |  |  |  cut ← i^max;
19 while dist^max > 0;
20 return cp_cut
```

The algorithm iterates (lines 9-16) on each point $p_i \in (p_{cut}, p_n)$ performing the following operations:

- normalizes $p_i.x$ with respect to $[p_{cut}.x, p_n.x]$ and stores such value in x_{norm} (line 10);
- normalizes $p_i.y$ with respect to $[p_n.y, p_{cut}.y]$ and stores such value in y_{norm} (line 11);
- if the normalized point (x_{norm}, y_{norm}) is below the line $y = 1-th-x$ (line 12), $dist^{tmp}$ is calculated as the distance of that point from $y = 1-x$ (line 13).
- if $dist^{tmp}$ is greater than $dist^{max}$ (line 14), $dist^{max}$ is updated to $dist^{tmp}$ (line 15) and i^{max} is updated to i (line 16).

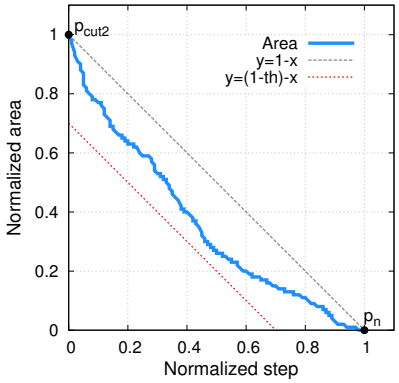
After having iterated on all points in $\{p_{cut}, ..., p_n\}$, if $dist^{max}$ is greater than zero (i.e. a new cut-off point was found) (line 17), cut is updated to i^{max}

(line 18). Finally, when $dist^{max}$ is equal to zero (i.e., there are no points below $y = 1 - th - x$) (line 19), the algorithm returns the convex polygons cp_{cut} as RoI $\mathcal{R}$ (line 20).



(a) Iteration 1: Found cut-off point p_{cut1}.

(b) Iteration 2: Found cut-off point p_{cut2}.



(c) Iteration 3: No cut-off point found.

Fig. 5.4. G-RoI selection procedure: An example with three iterations.

Figure 5.4 shows an example in which G-RoI selection procedure iterates three times to find the cut-off point. At the first iteration, the algorithm analyses the points in $\{p_0, ..., p_n\}$ and finds the first cut-off point p_{cut1} (see Figure 5.4(a)). At the second iteration, the algorithm analyses the points in $\{p_{cut1}, ..., p_n\}$ and finds a new cut-off point p_{cut2} (see Figure 5.4(b)). At the third iteration, the algorithm analyses the points in $\{p_{cut2}, ..., p_n\}$ but it does

not find any cut-off point (see Figure 5.4(c)). Therefore, the algorithm returns as RoI $\mathcal{R}$ the convex polygon corresponding to p_{cut2}.

5.4 Evaluation

We experimentally evaluated the accuracy of G-RoI in detecting the RoIs associated to a set of PoIs, comparing it with three existing techniques: *Circle* [126] (representative of the predefined-shapes approach), *DBSCAN* [156] (density-based clustering), and *Slope* [19] (grid-based aggregation). The analysis was carried out on 24 PoIs located in the center of Rome (St. Peter's Basilica, Colosseum, Circus Maximus, etc.) and 24 PoIs located in the center of Paris (Louvre Museum, Eiffel Tower, etc.) using about 2.3 millions geotagged items published in Flickr from January 2006 to May 2016 in the areas under analysis.

5.4.1 Performance metrics

To measure the accuracy of the algorithms in detecting RoIs, we use *precision* and *recall* metrics. As in [62], let roi_{real} be the real RoI for a PoI, and let roi_{found} be the RoI found by an algorithm. Let us define the true positive area roi_{TP} as the intersection of roi_{found} and roi_{real}. Precision $Prec$ and recall Rec are defined as:

$$Prec = \frac{Area(roi_{TP})}{Area(roi_{found})} \tag{5.1}$$

$$Rec = \frac{Area(roi_{TP})}{Area(roi_{real})} \tag{5.2}$$

A roi_{found} larger than roi_{real} produces a high recall and a low precision, whereas roi_{found} smaller than roi_{real} produces a low recall and a high precision. To rank the results, we combine precision and recall using the F_1 score:

$$F_1 = \frac{2 \cdot Prec \cdot Rec}{Prec + Rec} \tag{5.3}$$

5.4.2 Data source

The evaluation has been performed on geotagged data collected from Flickr[2], which is one of the most used social networks for photo sharing. Flickr shares more than one billion of photos that can be gathered using public APIs, which allow to retrieve metadata about all the photos matching the provided search criteria, e.g. the photos taken in a radius from a given geographical point.

[2] http://flickr.com

Using the APIs, we collected metadata about 2.3 millions geotagged items published in Flickr from January 2006 to May 2016 in the central areas of Rome and Paris. For each photo matching the search criteria, the Flickr APIs returned a metadata element such as the one shown in Figure 5.5.

```
{ "id":"987654321",
"owner":{"id":"123456789@N00","username":"FlickrUser"},
"dateTaken":"May 3, 2015 4:39:24 PM",
"tags":[
{"value":"italy"},{"value":"rome"},{"value":"piazzadispagna"},
{"value":"itali"},{"value":"spanishteps"}
],
"title":"Night at Piazza di Spagna",
"description": "In the Piazza di Spagna, just
      below the Spanish Steps",
"geoData":{ "longitude":12.482045, "latitude":41.905888}
...
}
```

Fig. 5.5. An example of metadata element returned by the Flickr APIs.

Each metadata element was parsed to extract the relevant features associated to geotagged items introduced in Section 5.1 (*text*, *tags*, *coordinates*, *userId*, *timestamp*).

5.4.3 Experimental results

The techniques under analysis need some parameters to work. We made several preliminary tests to find parameter values that perform effectively in all the scenarios, taking into account that the various PoIs are characterized by significant variability of shape, area and density (number of Flickr photos divided by area). For the Circle technique, the radius was set to 260 meters. With DBSCAN, the maximum distance between points is 10 meters and the minimum number of cluster points is 150. For the Slope technique, the square cell side is 55 meters and the minimum cell support is 150. For G-RoI, the threshold th was set to 0.27. The next two sections present the results obtained on 24 representative PoIs in Rome and 24 PoIs in Paris, respectively.

Rome

Figure 5.6 reports a graphical view of six (out of the 24 analyzed) representative PoIs in Rome (St. Peter's Basilica, Circus Maximus, Colosseum, Roman Forum, Arch of Constantine and Trevi Fountain): *i*) purple lines represent

the RoIs found by Circle; *ii*) orange lines represent the RoIs identified by DBSCAN; *ii*) red lines the RoIs found by Slope RoI; *iii*) blue lines those found using G-RoI; *iv*) black dotted lines the real RoIs.

As shown in the figure, the RoIs identified by the Circle technique are very approximative compared to the real ones. This is due to two reasons: *i*) circles cannot be used to represent elongated shapes (e.g. Circus Maximus); *ii*) with a given radius it is difficult to represent well places with very different areas (e.g., Colosseum vs Trevi Fountain). DBSCAN produced accurate results with St. Peter's Basilica and Colosseum, but failed in finding RoIs of places with low density (e.g., Circus Maximus) or very close to a another big place (e.g., Arch of Constantine, which is close to Colosseum). Also Slope failed in distinguishing RoIs from two adjacent places (e.g., Colosseum and Roman Forum) that do not present significant density variations. Moreover, Slope fails in finding good RoIs for places with low density (e.g., with Circus Maximus it found a very small RoI compared to the real one). Differently from the previous techniques, G-RoI is able to represent PoIs characterized by different shapes, areas and densities. In fact, G-RoI works well with both compact and elongated shapes (e.g., Trevi Fountain and Circus Maximus), with both small and large areas (e.g., Arch of Constantine and Roman Forum), and with various densities (from Circus Maximus to Colosseum). In addition, G-RoI accurately distinguishes RoIs of adjacent PoIs (e.g., Arch of Constantine and Colosseum).

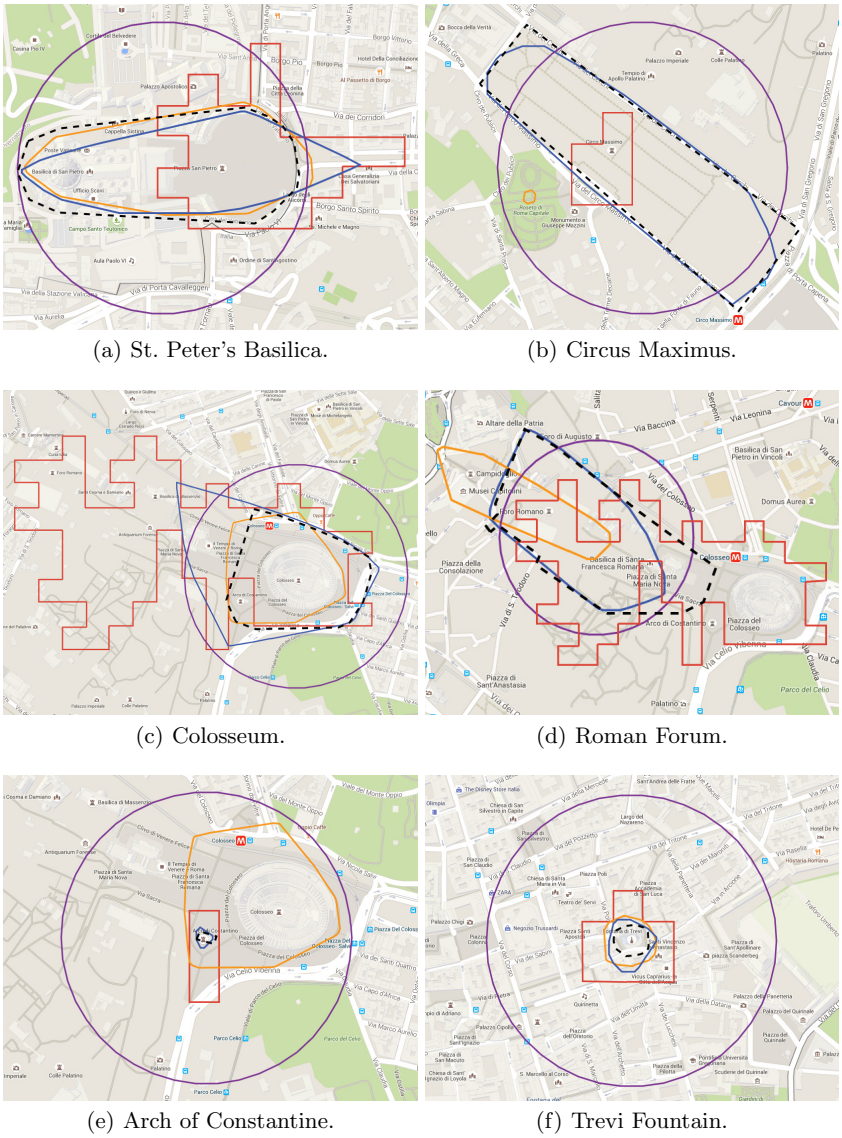
Table 5.2 illustrates the performance (Precision, Recall, F_1 score) of the four techniques, for all the 24 PoIs that have been considered. The last row of the table reports mean values computed over the 24 PoIs.

The results reported in the table confirm that using a predefined shape (the Circle) does not bring to accurate results. In fact, in most cases the Circle produces a very high recall with a low precision (which result in a mean F_1 score of 0.26), which means that the RoI identified by the technique is too large compared to the real one.

DBSCAN achieves the best results (F_1 score ranging from 0.74 to 0.91) with four PoIs - St. Peter's Basilica, Colosseum, Piazza Navona and Mausoleum of Hadrian - which are characterized by a similar density. On average, the precision of DBSCAN was 0.69 and the recall was 0.54, which leads to a mean F_1 score of 0.43. The fact that the precision is higher than the recall, means that the RoIs identified by DBSCAN are too small compared to the real ones.

Slope identifies the best RoI only with one PoI, Palazzo Montecitorio, with an F_1 score of 0.67. On the mean, the precision of Slope was 0.48 and the recall was 0.66, with a mean F_1 score of 0.38. In this case, the precision is lower than the recall, which means that the RoIs identified by this techniques are on average larger than the real ones.

Finally, G-RoI outperformed the other RoI mining techniques in 19 out of 24 PoIs, with a mean precision of 0.78, a mean recall of 0.82, and a mean F_1 score of 0.77 (0.34 higher than the F_1 score of DBSCAN). These results



(a) St. Peter's Basilica. (b) Circus Maximus.

(c) Colosseum. (d) Roman Forum.

(e) Arch of Constantine. (f) Trevi Fountain.

Fig. 5.6. RoIs identified by different techniques: Circle (purple lines), DBSCAN (orange), Slope (red), G-RoI (blue). Real RoIs shown as black dotted lines.

confirm the ability of G-RoI to accurately identify RoIs regardless of shapes, areas and densities of PoIs, and without being influenced by the proximity of

PoI	*Circle*			*DBSCAN*			*Slope*			*G-RoI*		
	Prec	*Rec*	F_1	*Prec*	*Rec*	F_1	*Prec*	*Rec*	F_1	*Prec*	*Rec*	F_1
St. Peter's Basilica	0.39	1.00	0.56	0.96	0.86	**0.91**	0.56	0.50	0.53	0.92	0.78	0.84
Circus Maximus	0.39	0.84	0.53	0.00	0.00	0.00	0.81	0.13	0.22	0.95	0.94	**0.94**
Colosseum	0.33	1.00	0.50	0.90	0.75	**0.82**	0.27	0.83	0.40	0.61	1.00	0.76
Roman Forum	0.62	0.85	0.71	0.61	0.25	0.00	0.44	0.62	0.51	0.95	0.80	**0.87**
Arch of Constantine	0.00	1.00	0.01	0.01	1.00	0.02	0.06	1.00	0.11	0.53	0.85	**0.65**
Trevi Fountain	0.01	1.00	0.03	0.42	1.00	0.59	0.14	1.00	0.24	0.49	1.00	**0.66**
Piazza Colonna	0.02	1.00	0.05	0.93	0.52	0.67	0.18	1.00	0.31	0.92	0.82	**0.87**
Tiber Island	0.14	1.00	0.24	1.00	0.02	0.03	0.40	0.26	0.31	0.72	0.81	**0.76**
Mausoleum of Hadrian	0.11	1.00	0.20	0.86	0.65	**0.74**	0.63	0.59	0.61	0.77	0.59	0.67
Piazza del Popolo	0.11	1.00	0.20	0.98	0.58	0.73	0.60	0.88	0.71	0.60	0.98	**0.74**
Villa Borghese	1.00	0.24	0.38	1.00	0.00	0.00	1.00	0.00	0.01	1.00	0.44	**0.61**
Piazza di Spagna	0.11	1.00	0.20	0.72	0.65	0.68	0.41	0.77	0.54	0.87	0.84	**0.86**
Piazza Venezia	0.09	1.00	0.17	0.57	0.78	0.66	0.13	0.99	0.22	0.52	0.96	**0.68**
Piazza Navona	0.06	1.00	0.11	0.71	0.96	**0.81**	0.23	1.00	0.38	0.49	0.99	0.66
Trastevere	1.00	0.36	0.53	1.00	0.01	0.02	1.00	0.04	0.08	1.00	0.55	**0.71**
Our Lady in Trastev.	0.02	1.00	0.03	0.62	0.98	0.76	0.14	1.00	0.25	0.83	0.94	**0.88**
Capitoline Hill	0.09	1.00	0.17	0.31	1.00	0.47	0.45	0.43	0.44	0.94	0.93	**0.94**
Vatican Museums	0.41	1.00	0.58	0.75	0.51	0.00	0.55	0.78	0.65	0.65	0.87	**0.75**
Pantheon	0.04	1.00	0.09	0.58	0.93	0.72	0.17	1.00	0.29	0.71	0.98	**0.82**
The Mouth of Truth	0.03	1.00	0.06	0.98	0.24	0.38	0.38	0.90	0.54	0.75	0.88	**0.81**
Palazzo Montecitorio	0.04	1.00	0.08	1.00	0.15	0.26	0.79	0.58	**0.67**	0.98	0.42	0.59
Campo de' Fiori	0.02	1.00	0.04	0.56	1.00	0.72	0.24	0.98	0.39	0.77	0.96	**0.85**
St Mary Major	0.12	1.00	0.22	1.00	0.21	0.35	0.88	0.53	0.66	0.86	0.65	**0.74**
Janiculum	0.59	0.70	0.64	0.00	0.00	0.00	1.00	0.03	0.07	0.94	0.78	**0.85**
Mean values	*0.24*	*0.92*	*0.26*	*0.69*	*0.54*	*0.43*	*0.48*	*0.66*	*0.38*	*0.78*	*0.82*	***0.77***

Table 5.2. Precision, Recall, and F_1 score of Circle, DBSCAN, Slope and G-RoI over 24 PoIs in Rome. For each row, the best F_1 score is indicated in bold.

different PoIs. For a complete view of the results produced by G-RoI, Figure 5.7 shows all the 24 RoIs of Rome found by G-RoI, compared with the real ones.

Paris

Figure 5.8 presents a graphical view of six (out of the 24 analyzed) representative PoIs in Paris (Louvre Museum, Eiffel Tower, Champs-Élysées, Notre-Dame, Pompidou Centre, Pont des Arts), while Table 5.3 presents the performance of the four techniques (Circle, DBSCAN, Slope and G-RoI), for all the 24 PoIs that have been considered in Paris.

The experimental results confirm the behavior observed in Rome RoIs. Also in this case, Circle does not compute accurate results, producing a very high recall with a low precision (which results in a mean F_1 score of 0.23).

DBSCAN achieves the best results only with four PoIs (i.e., Notre-Dame, Moulin Rouge, Paris Opera, and Arc de Triomphe). On the mean, the precision of DBSCAN was 0.85 and the recall was 0.42, which means that the

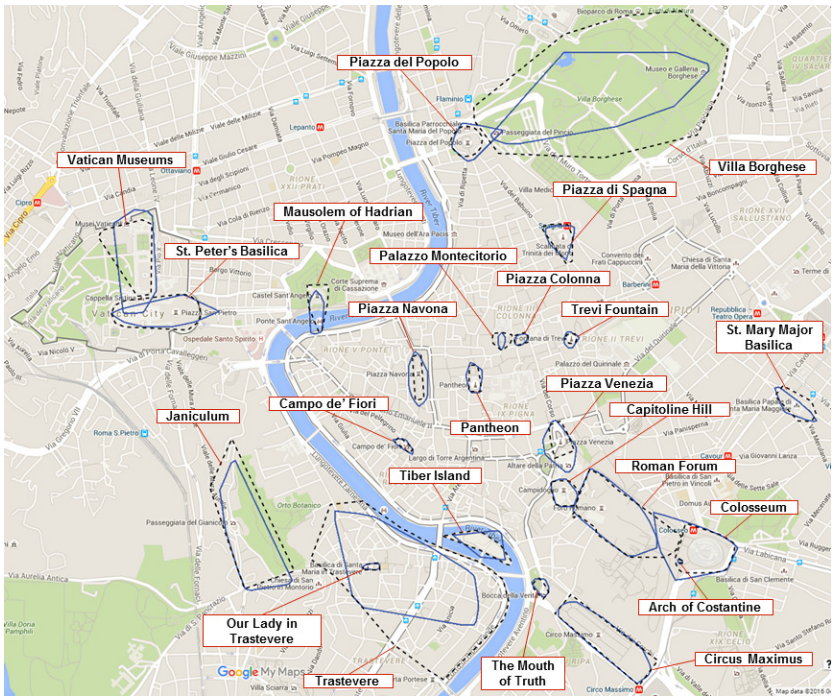
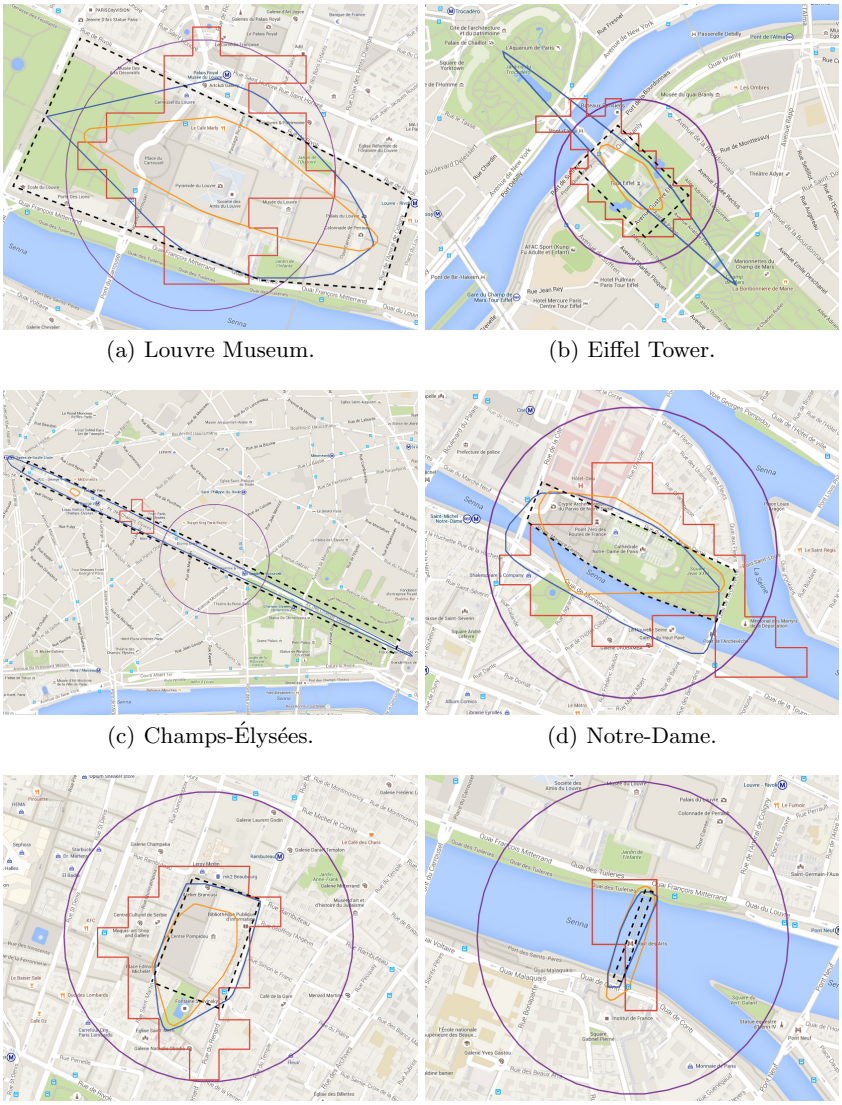


Fig. 5.7. City of Rome: RoIs identified by G-RoI (blue lines) compared with real ones (black dotted lines).

RoIs identified by this techniques are on average smaller than the real ones. Furthermore, Slope identifies the best RoI only for two PoIs (i.e. Eiffel Tower and Place de la Concorde). On average, the precision of Slope was 0.45 and the recall was 0.64, with an average F_1 score of 0.44. In this case, the precision is lower than the recall, which means that the RoIs identified by this techniques are on average larger than the real ones.

Finally, G-RoI outperformed the other RoI mining techniques in 18 out of 24 PoIs, with a mean precision of 0.81, a mean recall of 0.66, and a mean F_1 score of 0.70 (0.23 higher than the mean F_1 score of DBSCAN). In particular, G-RoI results to be the only technique able to identify an accurate RoI for the Champs-Élysées that are characterized by a very elongated shape, achieving a very high F_1 score (0.77). The behavior of G-RoI for the Eiffel Tower deserves to be discussed: differently from the other techniques, G-RoI produces a larger RoI with an elongated shape. This is due to the fact that anyone who wants to take a picture of the Eiffel Tower does not come strictly under it, but at some distance in front of it or behind it. Specifically, most geotagged items on this subject are located at Trocadéro, commonly considered the best place to take picture with Eiffel Tower in background. Overall, also the results on Paris



(a) Louvre Museum. (b) Eiffel Tower.

(c) Champs-Élysées. (d) Notre-Dame.

(e) Pompidou Centre. (f) Pont des Arts.

Fig. 5.8. RoIs identified by different techniques in Paris: Circle (purple lines), DB-SCAN (orange), Slope (red), G-RoI (blue). Real RoIs shown as black dotted lines.

confirm the ability of G-RoI in identifying RoIs characterized by a variety of shapes, areas and densities of PoIs.

PoI	*Circle*			*DBSCAN*			*Slope*			*G-RoI*		
	Prec	*Rec*	F_1	*Prec*	*Rec*	F_1	*Prec*	*Rec*	F_1	*Prec*	*Rec*	F_1
Louvre Museum	0.66	0.72	0.69	1.00	0.36	0.53	0.74	0.49	0.59	0.94	0.69	**0.79**
Tour Eiffel	0.28	1.00	0.44	1.00	0.38	0.55	0.56	0.98	**0.72**	0.46	0.57	0.51
Champs-Élysées	0.18	0.26	0.22	1.00	0.01	0.02	0.65	0.08	0.14	0.95	0.64	**0.77**
Notre-Dame	0.18	1.00	0.30	0.76	0.84	**0.79**	0.32	0.84	0.46	0.53	0.90	0.67
Pompidou Centre	0.13	1.00	0.23	0.82	0.66	0.73	0.37	0.98	0.54	0.78	0.98	**0.87**
Pont des Arts	0.01	1.00	0.02	0.31	1.00	0.48	0.11	0.75	0.19	0.42	1.00	**0.59**
Place de la Concorde	0.26	1.00	0.41	1.00	0.15	0.26	0.74	0.79	**0.77**	0.99	0.43	0.60
Moulin Rouge	0.02	1.00	0.04	0.81	0.86	**0.84**	0.00	0.00	0.00	0.72	0.62	0.67
Place de la Bastille	0.07	1.00	0.13	1.00	0.24	0.39	0.62	0.87	0.73	0.95	0.69	**0.80**
Sacré-Cœur Basilica	0.05	1.00	0.09	0.48	0.90	0.63	0.02	0.01	0.01	0.81	0.63	**0.71**
Jardin des Plantes	0.77	0.79	0.78	1.00	0.00	0.01	1.00	0.09	0.16	0.97	0.84	**0.90**
Saint-Sulpice	0.06	1.00	0.11	1.00	0.09	0.17	0.59	0.57	0.58	0.96	0.48	**0.64**
Pantheon	0.11	1.00	0.19	1.00	0.29	0.45	0.62	0.82	0.70	0.74	0.78	**0.76**
Trocadéro	0.20	1.00	0.34	1.00	0.28	0.43	0.83	0.52	0.64	0.89	0.70	**0.78**
Place de la République	0.08	1.00	0.14	0.97	0.46	0.62	0.58	0.77	0.66	0.98	0.59	**0.74**
Musée de l'Orangerie	0.02	1.00	0.05	1.00	0.52	0.68	0.24	0.88	0.38	0.91	0.70	**0.79**
Galeries Lafayette	0.07	1.00	0.12	0.92	0.26	0.41	0.36	0.83	0.50	0.87	0.76	**0.81**
Arab World Institute	0.04	1.00	0.07	0.96	0.49	0.65	0.28	0.99	0.44	0.96	0.55	**0.70**
Grand Palais	0.17	1.00	0.30	1.00	0.38	0.55	0.61	0.94	0.74	0.83	0.85	**0.84**
Petit Palais	0.05	1.00	0.10	1.00	0.36	0.53	0.07	0.33	0.11	0.78	0.59	**0.67**
Paris Opera	0.07	1.00	0.13	0.90	0.56	**0.69**	0.37	0.84	0.52	0.93	0.49	0.64
Pont Neuf	0.04	1.00	0.08	0.83	0.18	0.30	0.16	0.74	0.27	0.55	0.59	**0.57**
Arc de Triomphe	0.05	1.00	0.10	0.55	0.77	**0.64**	0.30	1.00	0.46	0.50	0.35	0.41
Sorbonne	0.20	1.00	0.33	0.00	0.00	0.00	0.75	0.21	0.33	0.99	0.47	**0.64**
Mean values	*0.16*	*0.95*	*0.23*	*0.85*	*0.42*	*0.47*	*0.45*	*0.64*	*0.44*	*0.81*	*0.66*	***0.70***

Table 5.3. Precision, Recall, and F_1 score of Circle, DBSCAN, Slope and G-RoI over 24 PoIs in Paris. For each row, the best F_1 score is indicated in bold.

5.5 Conclusions

RoI mining techniques are aimed at discovering Regions-of-Interest (RoIs) from Places-of-Interest (PoIs) and other data. Existing RoI mining techniques are based on the use of *predefined shapes*, *density-based clustering* or *grid-based aggregation*. In this chapter we presented *G-RoI*, a novel RoI mining technique that exploits the indications contained in geotagged social media items to discover the RoI of a PoI with a high accuracy.

We experimentally evaluated the accuracy of G-RoI in detecting the RoIs associated to a set of PoIs, comparing it with three existing techniques: *Circle* (predefined-shapes approach), *DBSCAN* (density-based clustering), and *Slope* (grid-based aggregation). The analysis was carried out on a set of PoIs located in the center of Rome, characterized by different shapes, areas and densities, using a large set of geotagged photos published in Flickr over six years. The experimental results show that G-RoI is able to detect more accurate RoIs than existing techniques. Over a set of 24 PoIs in Rome, G-RoI achieved better results than related techniques based on the three classes of existing

algorithms in 19 cases, with a mean precision of 0.78, a mean recall of 0.82, and a mean F_1 score of 0.77. In particular, the F_1 score of G-RoI is 0.34 higher than that obtained with the well-known DBSCAN algorithm.

To better assess the accuracy of G-RoI, further experiments have been run over an additional set of 24 PoIs in Paris. Also in this case, G-RoI achieved best results in 18 cases, with a mean precision of 0.81, a mean recall of 0.66, and a mean F_1 score of 0.70 (0.23 higher than that obtained with DBSCAN). These results confirm the ability of G-RoI to accurately identify RoIs regardless of shapes, areas and densities of PoIs, and without being influenced by the proximity of different PoIs. For the purpose of reproducibility, an open-source version of G-RoI and all the input data used in the experiments are available at `https://github.com/scalabunical/G-RoI`.

6

A scalable data mining technique for flight delay prediction

A viable approach to implement complex algorithms for Big Data analysis is based on the development of machine learning techniques on scalable parallel computing systems. In fact, parallel machine learning algorithms coupled with scalable computing and storage infrastructures can offer an effective way to mine very large and complex datasets by the exploitation of artificial intelligence approaches able to obtain usable results in reasonable time [132].

Advanced machine learning techniques and associated data mining tools can help to understand and predict several complex phenomena and attack many problems in different application areas. This approach can be useful in enabling businesses and research collaborations alike to make informed decisions. In this chapter we describe how to exploit parallel computing techniques coupled with Cloud computing systems to solve a Big Data analytics problem with a significant economical impact: flight delay prediction. Every year approximately 20% of airline flights are delayed or canceled mainly due to bad weather, carrier equipment or technical airport problems. These delays result in significant cost to both airlines and passengers. For instance, the cost of flight delays for US economy was estimated to be $32.9 billion in 2007 [8] and more than half of it was charged to passengers.

The work we present in this chapter aimed at implementing a predictor of the arrival delay of a scheduled flight due to weather conditions, as several studies have shown that weather is one of the primary causes of flight delays [6]. The predicted arrival delay takes into consideration both flight information (origin airport, destination airport, scheduled departure time, scheduled arrival time) and weather conditions at origin airport and destination airport according to the flight timetable.

Two open datasets of airline flights and weather observations have been collected and exploratory data analysis has been performed to discover initial insights, evaluate the quality of data, and identify potentially interesting subsets. Then, data preprocessing and transformation (joining and balancing operations) have been performed to make data ready for modeling. Finally, a parallel version of the Random Forest data classification algorithm has been

implemented, iteratively calibrating its settings to optimize results in terms of accuracy and recall. The data preparation and mining tasks have been implemented as MapReduce programs [35] that have been executed on a Cloud infrastructure. Other than providing the necessary computing resources for our experiments, the Cloud makes the proposed process more general: in fact, if the amount of data increases (e.g., by extending the analysis to many years of flight and weather data), the Cloud can provide the required resources with a high level of elasticity, reliability, and scalability.

The results show a high accuracy in prediction of delays above a given threshold. For instance, with a delay threshold of 15 minutes we achieve an accuracy of 74.2% and a delay recall of 71.8%, while with a threshold of 60 minutes the accuracy is 85.8% and the delay recall is 86.9%. An interesting result of our work is that, even without considering weather conditions, the model achieves an accuracy of 69.1%. This means that there is a persisting pattern of flight delay that is identified by the proposed methodology, which can be used to inform airlines what they should improve in terms of flight schedule to reduce delays. Moreover, the experimental results show the scalability obtained by executing in parallel on the Cloud, using MapReduce, both data preparation and data mining tasks.

The predictions provided by the developed system could be used as a weather-related component in a recommender system for passengers, airlines, airports, and websites specialized in booking flights. Considering delays due to weather conditions, passengers and airlines could estimate whether a flight will be delayed or not; airports could utilize the predictor to assist decision-making in air traffic management; websites allowing to book a single or multi-stop flight could use the system for suggesting the most reliable flight, i.e. the flight having the best likelihood to arrive on time. This is even true for multi-stop flights in which a single delay can lead to the cancellation of the whole flight.

Our approach significantly differs from the system employed by the Federal Aviation Administration (FAA). In fact, FAA uses the Weather Impacted Traffic Index (WITI) [21] to estimate the total delay in a given airport, based on weather information and actual traffic (i.e., queuing delay reflecting excess traffic demand versus capacity). In contrast, our system is able to predict the delay of individual flights, using specific flight information (origin airport, destination airport, scheduled departure and arrival time), in addition to weather conditions at origin and destination airport. Predicting individual flight delays is an important feature that could be used to extend or complement current FAA's system. Another result of our research that could be used to improve the performance of existing prediction tools, including FAA's system, is that accuracy of prediction significantly improves when several weather observations before scheduled flight time are considered, rather than just a single one.

The remainder of the chapter is organized as follows. Section 6.1 introduces the main concepts, briefly describes the datasets used in this work, and

outlines the performance metrics used to assess the quality of results. Section 6.2 explores the large collection of flight data available to identify the subsets of data that are suitable for analysis. Section 6.3 describes the data analysis process implemented to generate the flight delay prediction models, starting from the input data. Section 6.4 presents an evaluation of the obtained results. Section 6.5 discusses related work. Finally, Section 6.6 concludes the chapter.

6.1 Problem definition

This section provides a definition of the main concepts underlying the problem addressed in this work. Moreover, the section provides a short description of the used datasets, and introduces the performance metrics used to assess quality of the results.

6.1.1 Preliminary definitions

Definition 6.1. *(Flight). A Flight F is a tuple $\langle A_o, A_d, t_{sd}, t_{ad}, t_{sa}, t_{aa} \rangle$, where A_o is the origin airport, A_d is the destination airport, t_{sd} is the scheduled departure time, t_{ad} is the actual departure time, t_{sa} is the scheduled arrival time to gate, and t_{aa} is the actual arrival time to gate, all times include dates, hours and minutes.*

Definition 6.2. *(Airport Weather Observation). An Airport Weather Observation O is a tuple $\langle A, t, T, H, W_d, W_s, P, S, V, D \rangle$, where A is the airport, t is the observation time (including date, hours and minutes), T is the temperature, H is the humidity, W_d is the wind direction, W_s is the wind speed, P is the barometric pressure, S is the sky condition, V is the visibility and D is the weather phenomena descriptor.*

Definition 6.3. *(Arrival Delay). The Arrival Delay of a Flight F, denoted $AD(F)$, is the difference between its actual and scheduled arrival times to gate, i.e. $AD(F) = F.t_{aa}$-$F.t_{sa}$, where the dot notation is used to get the different fields of a tuple (e.g. $F.t_{aa}$ refers to t_{aa} of flight F)*

Definition 6.4. *(On-time Flight). Given a flight F and a threshold Th, F is an On-time Flight if $AD(F) < Th$.*

Definition 6.5. *(Delayed Flight). Given a flight F and a threshold Th, F is a Delayed Flight if $AD(F) \geq Th$.*

6.1.2 Problem statement

As mentioned before, the goal of this work is to predict the arrival delay of a scheduled flight due to weather conditions. The predicted arrival delay takes into consideration both flight information (origin airport, destination airport,

scheduled departure time, scheduled arrival time) and weather conditions at origin airport and destination airport according to the flight timetable. The predicted arrival delay of any flight F scheduled to depart from airport A_o at time t_{sd}, and to arrive at airport A_d at time t_{sa}, is an estimate of the arrival delay $AD(F)$. If the predicted arrival delay of a scheduled flight F is less than a given threshold, it is classified as an on-time flight; otherwise, it is classified as a delayed flight. We do not take into account en-route weather conditions because it is not trivial to infer the weather along a flight trajectory. In fact, given the different positions of an aircraft, it is difficult to combine measurements of nearby weather stations, considering that the altitude of the aircraft should also be taken into account [130].

6.1.3 Data sources

The results presented in this chater have been obtained using the Airline On-Time Performance (AOTP) dataset provided by RITA - Bureau of Transportation Statistics[1] for the five-year period beginning January 2009 and ending December 2013. The AOTP dataset contains data for domestic US flights by major air carriers, providing for each flight detailed information such as origin and destination airports, scheduled and actual departure and arrival times, air time, and non-stop distance.

The second data source used in this work is the Quality Controlled Local Climatological Data (QCLCD) dataset available from the National Climatic Data Center[2], considering the same five-year period (January 2009 - December 2013). The large period considered ensures that the inferred model is able to predict delays due to almost every condition, as are excluded only those rare events that did not happen in the large time frame considered. The QCLDC dataset contains hourly weather observations from about 1,600 U.S. stations. Each weather observation includes data about temperature, humidity, wind direction and speed, barometric pressure, sky condition, visibility and weather phenomena descriptor. According to the METAR format [48], the phenomena descriptor (precipitation, obscuration, or other) might be preceded by one or two qualifiers (intensity or proximity to the station and descriptor). For instance, $+SN$ indicates a heavy snow phenomena and $TSGR$ a thunderstorm with hail.

Table 6.1 reports size, number of tuples and number of columns of the datasets used in this work.

6.1.4 Performance metrics

A confusion matrix is a common method used to measure the quality of a classification. It contains information about the instances in an actual and a

[1] http://www.transtats.bts.gov/
[2] http://cdo.ncdc.noaa.gov/qclcd/QCLCD

Name	Size (GB)	N. of tuples	N. of columns
AOTP	13.37	31 millions	109
QCLCD	27.68	233 millions	44

Table 6.1. Datasets specifications.

predicted class. In particular, each row of a confusion matrix represents the instances in an actual class, while each column represents the instances in a predicted class.

Table 6.2 shows the confusion matrix for the problem we addressed. Flights that are correctly predicted as on-time are counted as True Positive (TP), whereas flights that are predicted as on-time but are actually delayed are counted as False Positive (FP). Similarly, flights that are correctly predicted as delayed are counted as True Negative (TN), whereas flights that are predicted as delayed but are actually on-time are counted as False Negative (FN).

	On-time (predicted)	Delayed (predicted)
On-time (actual)	True Positive (TP)	False Negative (FN)
Delayed (actual)	False Positive (FP)	True Negative (TN)

Table 6.2. Confusion matrix.

Starting from the confusion matrix we can calculate some metrics. One of the most frequently used evaluation metrics in machine learning is *accuracy*, denoted *Acc*, which measures the fraction of all instances that are correctly classified.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

Accuracy provides an overall quality measure of a classifier, but it does not provide information about the goodness of a classifier in predicting a specific class. Therefore, recall metrics are often used to measure the quality of a classifier with respect to a given class.

We define *on-time recall*, denoted Rec_o, the ratio between the number of flights correctly classified as on-time (TP), and the total number of flights actually on-time ($TP + FN$). Similarly, the *delayed recall*, denoted Rec_d, is the ratio between the number of flights correctly classified as delayed (TN), and the total number of flights actually delayed ($TN + FP$).

$$Rec_o = \frac{TP}{TP + FN} \qquad Rec_d = \frac{TN}{TN + FP} \tag{6.2}$$

6.2 Data understanding

In this section, we study in depth the airline flights dataset (AOTP) to understand how to filter flights that are really delayed by weather conditions.

As described above, the AOTP dataset contains data on US flights by major air carriers. Table 6.3 reports the percentage of flights per year that have been on time, delayed, canceled or diverted. The Federal Aviation Administration (FAA) considers a flight as *delayed* when it is 15 minutes later than its scheduled time. A *canceled* flight is when the airline does not operate the flight at all for a certain reason. A *diverted* flight is one that has been routed from its original arrival destination to a new arrival destination.

Year	Flights	Ontime	Delayed	Cancelled	Diverted
2009	6,450,285	79.5%	18.9%	1.4%	0.2%
2010	6,450,117	79.8%	18.2%	1.8%	0.2%
2011	6,085,281	79.6%	18.2%	1.9%	0.2%
2012	6,096,762	81.9%	16.7%	1.3%	0.2%
2013	6,369,482	78.3%	19.9%	1.5%	0.2%

Table 6.3. Analysis of flight on-time performance by year.

Since June 2003, US airlines report information about their flights to Bureau of Transportation Statistics (BTS)[3]. In case of delay (or cancellation) the airlines report the causes of delay in five broad categories:

- *Air carrier*: The cause of delay was due to circumstances within the airline's control (e.g. maintenance or crew problems, aircraft cleaning, baggage loading, fueling).
- *Late-arriving aircraft*: A previous flight with the same aircraft arrived late, so causing the present flight to depart late.
- *National Aviation System (NAS)*: Delays due to the National Aviation System that refer to a large set of conditions, such as non-extreme weather conditions, airport operations, heavy traffic volume, and air traffic control.
- *Extreme weather*: Significant meteorological conditions (actual or forecast) that, in the judgment of the carrier, delays or prevents the operation of a flight such as tornado, blizzard or hurricane.
- *Security*: Delays caused by evacuation of a terminal, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 29 minutes at screening areas.

Notice that, a delayed flight can be assigned to a single or multiple delay broad categories. Table 6.4 shows the percentage of delayed flights assigned to each broad categories divided by year. When multiple causes are assigned to one delayed flight, each cause is prorated based on delayed minutes it is responsible for.

[3] http://www.rita.dot.gov/bts/

Year	Air carrier	Late-arriving aircraft	NAS	Extreme weather	Security
2009	26.6%	32.8%	37.0%	3.4%	0.2%
2010	28.9%	35.8%	32.1%	3.1%	0.3%
2011	28.2%	37.0%	31.8%	2.8%	0.2%
2012	29.8%	37.6%	29.6%	2.8%	0.2%
2013	27.8%	38.8%	30.3%	2.9%	0.2%

Table 6.4. Analysis of flight delay causes by year.

Following the *Understanding the Reporting of Causes of Flight Delays and Cancellations*[4] report from BTS, the number of weather-related delayed flights is the sum of: *i*) all delays due to extreme weather; *ii*) the percentage of NAS delays that FAA considered due to weather (e.g., during 2013, 58.3% of NAS delays were due to weather); and *iii*) the late-arriving aircraft related to weather that can be calculated using the proportion of weather related-delays and total flights in the other categories. Table 6.5 reports the percentage of delayed flights assigned to extreme weather, NAS related to weather, late-arriving aircraft related to weather and the total weather delay.

Year	Extreme weather	NAS related to weather	Late-arriving aircraft related to weather	Total weather
2009	3.4%	24.3%	14.5%	42.3%
2010	3.1%	20.4%	14.0%	37.4%
2011	2.8%	20.1%	14.3%	37.2%
2012	2.8%	17.4%	12.6%	32.8%
2013	2.9%	17.7%	14.1%	34.6%

Table 6.5. Analysis of delayed flights due to weather conditions by year.

Figure 6.1 depicts the percentage of delayed flights associated to a single delay cause or a combination of them. For example 13.2% of delayed flights are only due to air carrier delays, 11.9% due to combination of late-arriving aircraft and NAS, or 8.9% due to combination of air carrier delay, late-arriving aircraft and NAS.

Tables 6.4-6.5 and Figure 6.1 helped us to create training datasets containing flights really delayed by weather and to evaluate the goodness of the classification models obtained.

6.3 Data analysis

This section describes the data analysis process implemented to generate flight delay prediction models, starting from the input data. The overall process, rep-

[4] http://www.rita.dot.gov/bts/help/aviation/html/understanding.html

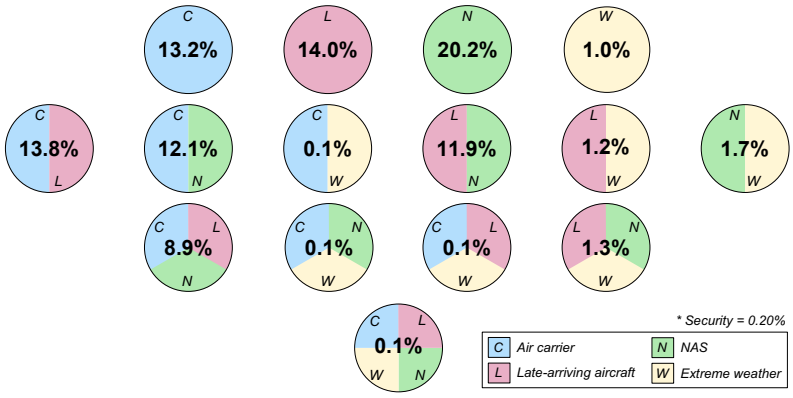


Fig. 6.1. Delayed flights due to a single delay cause or a combination of them.

resented in Figure 6.2, is composed of three main phases: 1) data preprocessing and transformation; 2) target data creation; 3) modeling.

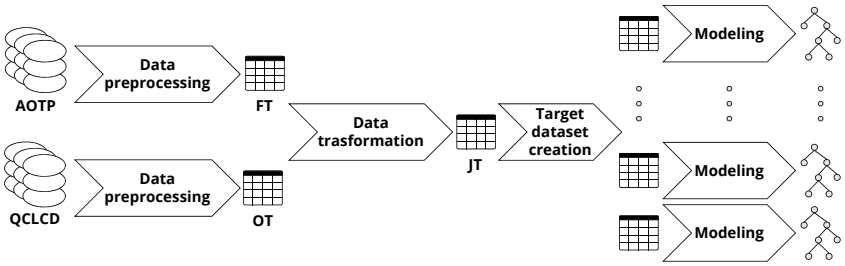


Fig. 6.2. Data analysis process.

6.3.1 Data preprocessing and transformation

As a first operation, data preprocessing was carried out on both flight dataset (AOTP) and the weather dataset (QCLCD) to look for possible wrong data and to treat missing values. Moreover, since our focus is on delayed flights only, we filtered out diverted and canceled flights from the AOTP dataset, obtaining a table referred to as Flight Table (FT). From the QCLCD dataset we removed all the weather observations not related to airport locations, obtaining a Weather Observations Table (OT).

Data transformation mostly refers to the operation of creating a Joint Table (JT) by joining the Flight Table and the Weather Observations Table. In particular, for each flight F in FT, the join operation creates in JT a tuple $\{F, W_o, W_d, C\}$, where:

- F is an array containing flight information (origin airport, destination airport, etc.);
- $W_o = \langle O(A_o, t_{sd}), O(A_o, t_{sd} - 1h), \ldots, O(A_o, t_{sd} - 12h)\rangle$ is an array containing weather observations at origin airport (A_o) from the scheduled departure time (t_{sd}) back to 12 hours before ($t_{sd} - 12h$) with intervals of 1 hour;
- $W_d = \langle O(A_d, t_{sa}), O(A_d, t_{sa} - 1h), \ldots, O(A_d, t_{sa} - 12h)\rangle$ is an array containing weather observations at destination airport (A_d) from the scheduled arrival time (t_{sa}) back to 12 hours before ($t_{sa} - 12h$) with intervals of 1 hour;
- C is the class attribute that indicates if F is on-time or delayed according to a given threshold Th.

In particular, the join operation is split in two steps: the *first join step* combines flight information with weather observations at origin airport, and the *second join step* combines the output of the first step with the weather observations at destination airport. This has been done by modifying the *improved repartition join* algorithm [16] and implementing it by two MapReduce tasks.

The improved repartition join performs a relational join between two tables, that we refer here as A and B. Each map task processes a partition of either A or B. To identify which table a tuple is from, each *map* task emits a *composite key*, consisting of a *join key* and a *table tag*. The join key is used during the partitioning step to assign tuples with the same join key to the same *reduce* task. The table tag is used during the sorting step to put the tuples from A before those from B. Thus, for each join key, the reducer processes first the tuples from A to hold them in memory, and then processes the tuples from B to make the join.

Our modified version of the improved repartition join works as follows. In the first join step, we use a join key $\langle A, D\rangle$, which is the combination of an airport A and a date D. If the mapper receives a tuple from OT, it generates $\langle O.A, Date(O.t)\rangle$ as a join key. Otherwise, if the mapper receives a tuple from FT, it generates $\langle F.A_o, Date(F.t_{sd})\rangle$ as a join key. In this way, a reducer receives all the departure flights and the weather observations of an airport A in a given date D. As table tag we use the table name ("OT" or "FT"). Therefore, the reducer encounters first the weather observations and store them in an array ordered by time. Then, the reducer processes the flights, adding to each of them an array containing the weather observations at origin airport from the scheduled departure time back to 12 hours before. Since that the weather dataset provides hourly weather observations at variable times, we take the closest one to the weather observation time requested.

The second join step is analogous to the first one, with the difference that we take the weather observations at destination instead of origin airports. Figure 6.3 shows an example of data flow (input, intermediate and output tuples) of the first join step.

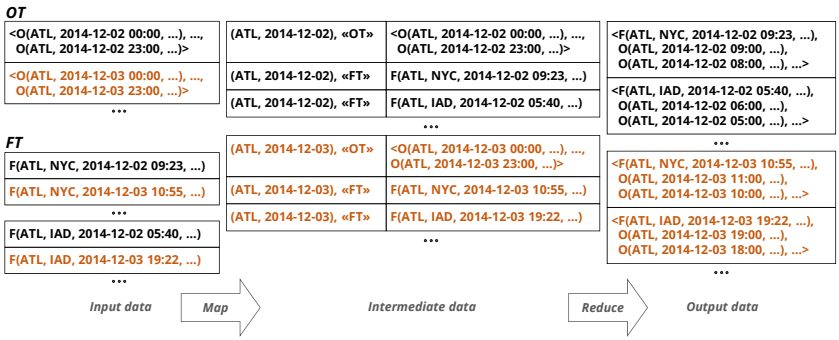


Fig. 6.3. Data flow of the first join step.

The MapReduce pseudo-code of the join process is shown in Algorithm 34.

6.3.2 Target data creation

Since our goal is to predict delayed flights by considering both flight and weather information at origin and destination, we try to select flights that are strictly related to this task. As explained in Section 6.2, selection of delayed flights due to weather conditions is not trivial, because they are distributed in three of the five broad categories (see Table 6.5) and each delay flight can be assigned to multiple broad categories (see Figure 6.1).

Thus, ideally, our target dataset should contain all delayed flights due to *extreme weather* and *NAS related to weather*. We do not take into account *late-arriving aircraft related to weather* because such delays do not depend on weather information at origin and destination airports, but they are due to delay propagation of previous flights originated by the same aircraft. To reach our aim, for each delay threshold considered, four target datasets have been created:

- $D1$ contains delayed flights due only to extreme weather or NAS, or a combination of them.
- $D2$ includes delayed flights affected by extreme weather, plus those ones for which NAS delay is greater than or equal to the delay threshold.
- $D3$ includes delayed flights affected by extreme weather or NAS, even if not exclusively (i.e., they might be also affected by other causes).
- $D4$ contains all delayed flights.

The first three datasets ($D1$, $D2$ and $D3$) are strictly related to our task as defined above, but have been created using different types of filtering. The last dataset contains all delayed tuples and has been created as a reference dataset.

Algorithm 3: MapReduce pseudo-code for the first join step.

```
1  Map(K: null, V: a tuple from a split of either OT or FT)
2      if V is a tuple from OT then
3          join_key ← ⟨V.A, Date(V.t)⟩
4          table_tag ← "OT"
5          tagged_tuple ← add a tag "OT" to V
6          composite_key ← ⟨join_key, table_tag⟩
7          emit(composite_key, tagged_tuple)
8      else
9          join_key ← ⟨V.A_o, Date(V.t_sd)⟩
10         table_tag ← "FT"
11         tagged_tuple ← add a tag "FT" to V
12         composite_key ← ⟨join_key, table_tag⟩
13         emit(composite_key, tagged_tuple)
14         if Date(V.t_sd).plusHours(12) is Date(V.t_sd).plusDays(1) then
15             join_key ← ⟨V.A_o, Date(V.t_sd).plusDays(1)⟩
16             composite_key ← ⟨join_key, table_tag⟩
17             emit(composite_key, tagged_tuple)
18         end
19     end
20
21 Partition(K': a composite key)
22     hashcode ← hash_function(K'.join_key)
23     return hashcode mod #reducers
24
25 Reduce(K': a composite key, LIST_V': a list of tagged tuples for K' first from
   OT then FT)
26     create an array of observations A_O ordered by time
27     create a temporary array of observations A_T
28     for each OT tuple o in LIST_V' do
29         put o in A_O
30     end
31     for each FT tuple f in LIST_V' do
32         A_T ← get_hourly_observations(A_O, f.t_sd)
33         emit(null, merge(f, A_T))
34     end
```

From a set-theoretic point of view, said Di_d the tuples representing delayed flights in Di, where $1 \leq i \leq 4$, the following rule holds:

$$(D1_d \cup D2_d) \subseteq D3_d \subseteq D4_d.$$

Table 6.6 summarizes the features of the four datasets and the percentage of delayed tuples contained when delay thresholds of 15 and 60 minutes are used.

It is worth noticing that the AOTP dataset is unbalanced because the two classes, *on-time* and *delayed*, are not equally represented. For example, during

Dataset ID	% Delayed tuples selected	% Delayed tuples ($Th = 15min$)	# Delayed tuples ($Th = 15min$)	% Delayed tuples ($Th = 60min$)	# Delayed tuples ($Th = 60min$)
D1	*Solo* Extreme U *Solo* NAS U *Solo* (Extreme and NAS)	22.9%	1.3M	15.4%	257k
D2	Extreme U NAS$\geq Th$	37.1%	2.1M	25.9%	433k
D3	Extreme U NAS	58.9%	3.4M	56.8%	950k
D4	All	100.0%	5.8M	100.0%	1.7M

Table 6.6. Features of target datasets.

year 2013, 78.3% of the total flights were on-time while 19.9% were delayed (see Table 6.3). Therefore, also the Joint Table JT is unbalanced, as most of its tuples are related to on-time flights. In order to get accurate prediction models and to correctly evaluate them, we need to use balanced *training sets* and *test sets* in which half the flights are on-time and half are delayed.

To this purpose, we used the random under-sampling algorithm [79], which balances class distribution through random discarding of major class tuples as described in Figure 6.4. In our case, for each target dataset we first divided tuples in on-time and delayed. Then, delayed tuples were randomly added to the training and test sets with a 3:1 ratio. Finally, on-time instances were randomly added, without repetition, until the number of delayed and on-time instances were the same. At the end of this process we obtain a $\langle trainingset, testset \rangle$ pair ready for modeling and evaluation.

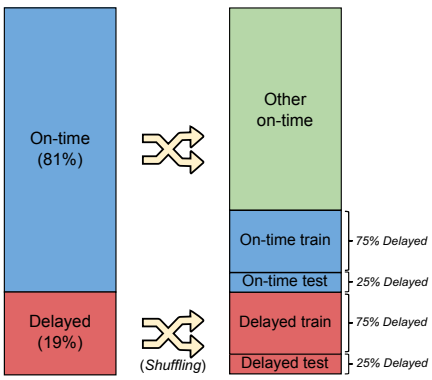


Fig. 6.4. Method used for creating balanced training and test sets.

6.3.3 Modeling

The modeling step is crucial for obtaining accurate data classification. Machine learning and statistics provide several techniques for data modeling. Statistics measures uncertainty in data and to do that it usually uses small and accurate data samples, while machine learning analyzes large datasets without any notion of uncertainty and no prior assumption.

In this work, different machine learning techniques have been studied to evaluate their appropriateness in the considered domain. To this end, we run a series of preliminary experiments on sample datasets to evaluate the performance of different classification algorithms (C4.5, SVM, Random Forest, Stochastic Gradient Descent, Naive Bayes, Logistic Regression). Based on the results of these experiments, the Random Forests (RF) algorithm [18] was selected as it achieved the best performance in terms of accuracy and recall, with limited build time of the model. Other research works exploited RF for flight delay prediction due to its high level of accuracy (e.g., see [117] discussed in Section 6.5).

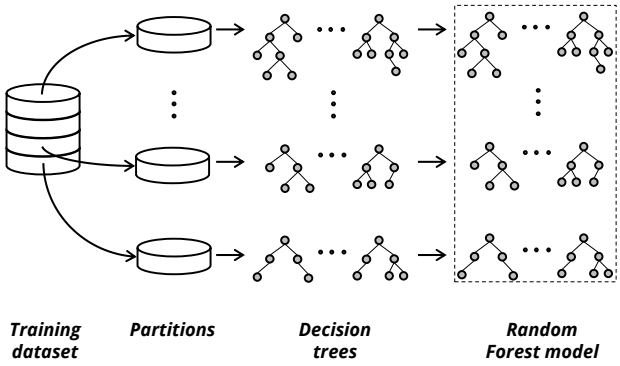
The good performance of the RF technique mainly derives from the fact that this learning technique uses a distributed intelligence approach. The RF approach is based on the creation of several classification trees built on different subsets of data using random subsections of available variables. The global result of this approach is the creation and refinement of a set of valid theories and hypotheses, represented by trees, and the combination of the trees into "a forest of classifiers" whose final decision is based on the votes of the different decision trees. An additional powerful feature of this approach is that is based on decentralized collective behavior without any centralized or hierarchical learning structure. In fact, Random Forest implement a form of emergent artificial intelligence strategy that maximizes the value of data and knowledge through a sort of concurrent learning that combine different models (theories). The Random Forest algorithm is a useful example of a set of cooperating reasoning processes that complement each other to reach a coordinated learning model.

In particular, RF is an ensemble learning method for classification. It creates a collection of different decision trees called $forest$. Each forest tree is built starting from a training dataset obtained applying bagging on the original training set. To enhance the ensemble diversity, further randomness is introduced: at each step, during the best attribute selection, only a small random attributes subset is considered. This set of procedures leads to an ensemble classifier with good performance compared with other classification algorithms [137]. Once forest trees are created, the classification of an unlabeled tuple is performed by aggregating the predictions of the different trees through majority voting.

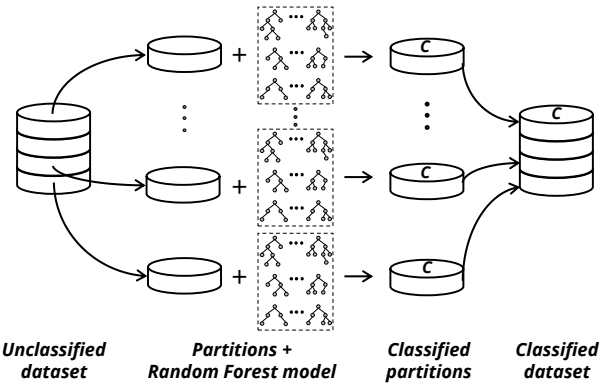
Since the sequential version of Random Forest is not able to deal with large datasets, we used a parallel version implemented in MapReduce. Model creation is performed in three steps, as described in Figure 6.5(a): i) the

training dataset is split into several data *partitions*, each one is sent to a processing node; *ii*) each processing node builds multiple *decision trees* from its data partition and store them on a different output file; and *iii*) finally, all the output files generated are merged to form the *Random Forest model*.

Also prediction, whose goal is estimating the class associated with an unclassified dataset, is composed by three steps (see Figure 6.5(b)): *i*) the *unclassified dataset* is split into different data *partitions*, each one is sent to a processing node; *ii*) each processing node uploads the *Random Forest model* and predicts the class of each tuple in its data partition generating a *classified partition*; and *iii*) finally, all the classified partitions are merged together to form the *classified dataset*.



(a) Model creation.



(b) Prediction.

Fig. 6.5. Parallel version of Random Forest implemented in MapReduce.

Focusing on predicting the class (*on-time* or *delayed*) of a scheduled flight F_s, the predictor takes as input the following data:

- $F_s = \langle A_o, A_d, t_{sd}, t_{sa} \rangle$, where A_o is the origin airport, A_d is the destination airport, t_{sd} is the scheduled departure time, and t_{sa} is the scheduled arrival time.
- An array $W_o = \langle O(A_o, t_{sd}), O(A_o, t_{sd} - 1h), \ldots, O(A_o, t_{sd} - (m-1)h) \rangle$ containing weather forecasts at origin airport (A_o) from the scheduled departure time (t_{sd}) back to $m-1$ hours before ($t_{sd} - (m-1)h$) with intervals of 1 hour.
- An array $W_d = \langle O(A_d, t_{sa}), O(A_d, t_{sa} - 1h), \ldots, O(A_d, t_{sa} - (n-1)h) \rangle$ containing weather forecasts at destination airport (A_d) from the scheduled arrival time (t_{sa}) back to $n-1$ hours before ($t_{sa} - (n-1)h$) with intervals of 1 hour.
- A delay threshold $d \in \{Th_1, Th_2, ...Th_z\}$.

A set of models M is assumed to be available to the predictor. A generic element in M, denoted as $M_{i,j,k}$, represents the model trained using a delay threshold i, considering j weather observations at origin airport, and k weather observations at destination airport.

Therefore, in order to estimate the class associated with F_s, the predictor proceeds as follows: first loads model $M_{d,m,n}$ from M; then provides F_s, W_o and W_d data to $M_{d,m,n}$; finally, returns the class assigned by $M_{d,m,n}$. Since weather forecast data (arrays W_o and W_d) are an important part of the input of the predictor and we can have them several days in advance, using them, the system is able to make predictions in the same time frame.

Given the predictor built on historical data, it is possible to use it to predict the delay of future flight using weather forecasts. However, the outcome of the predictor will be influenced by the accuracy of the forecasts. Of course, the more reliable weather forecasts, the more accurate the predictions. Today the accuracy of weather forecasts done some days in advance is high. In fact, the major weather services available online report an accuracy ranging from about 75% (three days in advance) to about 90% (one day in advance). Evaluating how the accuracy of forecasts affects the accuracy of predictions would be interesting. However, given the high values of whether prediction accuracy some days in advance and the same day of the flight, the predictor results are highly accurate.

6.4 Evaluation

We evaluated the accuracy of our models in predicting flight delays above a given time threshold. Moreover, we evaluated the scalability achieved carrying out the whole data analysis and evaluation process as a collection of MapReduce tasks on a Cloud platform. Specifically, we used *HDInsight*, a service that deploys an Apache Hadoop [143] MapReduce cluster on Microsoft Windows

Azure[5]. Our cluster was equipped with 1 head node having eight 2.2 GHz CPU cores and 14 GB of memory, and 12 worker nodes having four 2.2 GHz CPU cores and 7 GB of memory.

Table 6.7 shows the parameters used for the evaluation tests: *i*) target datasets ($\langle trainingset, testset \rangle$ pairs), as described in Section 6.3.2; *ii*) delay threshold in minutes; *iii*) number of hourly weather observations considered at origin airport; and *iv*) number of hourly weather observations considered at destination airport. Each test has been performed ten times by regenerating disjoint $\langle trainingset, testset \rangle$ pairs for assessing the results. In order to vary the number of weather observations considered at origin and destination airport, in each experiment we asked the classification algorithm to train a new model considering a given subset of features in arrays W_o and W_d, as defined in Section 6.3.1. When we ask the algorithm to consider m observations at origin and n observations at destination, the algorithm trains the model considering the first m observations from W_o (i.e., $O(A_o, t_{sd})$, $O(A_o, t_{sd} - 1h)$, ..., $O(A_o, t_{sd} - (m-1)h)$ and the first n observations from W_d (i.e., $O(A_d, t_{sa})$, $O(A_d, t_{sa} - 1h)$, ..., $O(A_d, t_{sa} - (n-1)h)$. As performance indicators, we used the accuracy (Acc), the on-time recall (Rec_o) and delayed recall (Rec_d). The goal is to maximize Acc with a balanced values of Rec_o and Rec_d.

Parameter	Values
Target dataset	D1, D2, D3, D4
Delay threshold	15, 30, 45, 60, 90
# of hourly weather observations considered at origin airport	0, 1, 3, 5, 7, 9, 11
# of hourly weather observations considered at destination airport	0, 1, 3, 5, 7, 9, 11

Table 6.7. Evaluation parameters.

The first set of experiments helped us to understand how many hourly weather observations have to be considered at origin and destination airport. Figure 6.6-a shows accuracy, on-time and delay recall values obtained varying from 0 to 11 the number of weather observations considered at origin airport, and 0 observations considered at destination airport. Similarly, Figure 6.6-b shows the performance indicators considering from 0 to 11 weather observations at destination airport, and 0 observations at origin airport. In both cases, we used $D2$ as target dataset and 60 minutes as delay threshold.

As expected, the performance indicators improve with the increase of weather observations considered. For example, Figure 6.6-a shows that the accuracy passes from 69.1% without using any weather information, to 80.5% when we consider 11 weather observations at origin airport. In the same way, Figure 6.6-b shows that the accuracy increases from 69.1% to 79.8% passing from 0 to 11 weather observations at destination airport.

[5] http://azure.microsoft.com/en-us/services/hdinsight

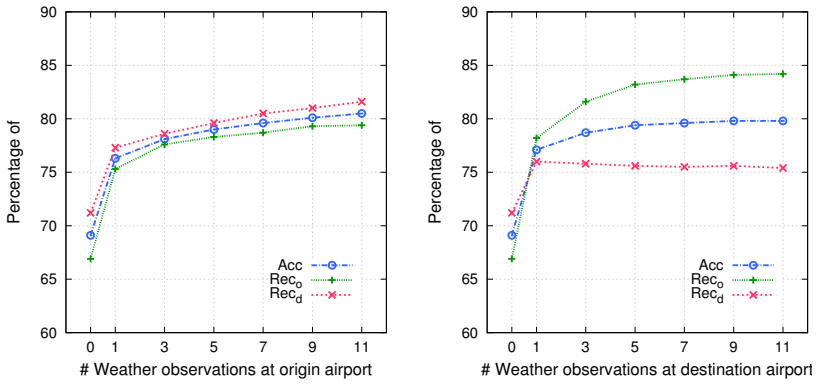


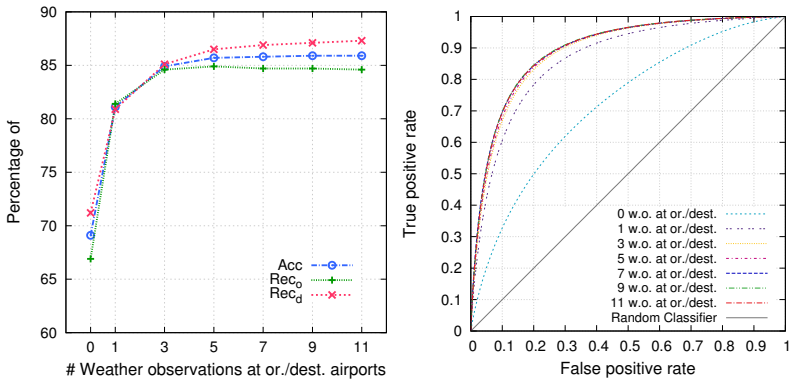
Fig. 6.6. Performance indicators vs number of weather observations considered at origin (a) and destination (b) airport.

As illustrated in Figure 6.6-a, the classifier shows a strongly balanced behavior on both prediction classes considering only weather observations at origin airport. In fact, Rec_o and Rec_d are very close for every number of weather observation considered. On the contrary, Figure 6.6-b shows that we get a slightly lower accuracy and a less balanced behavior in terms of recall considering only weather observations at destination airport. In particular, in this case the model tends to perform better with the on-time class, but worse with the delayed class, in contrast to the performance shown in Figure 6.6-a where Rec_o and Rec_d are always very close.

As an additional remark on the results described above, the improvement that we get by accounting for more than three hours of weather observations prior to the schedule departure/arrival time is limited but valued. Regarding the causality between weather and delay, the results indicate that adverse weather conditions take effect several hours after they end, very likely due to cascading delay effects.

Then, we studied the predictor performance using the same number of weather observations at origin and destination airports (see Figure 6.7-a).

As we expected, combining weather information at origin and destination airports leads to an improvement of the accuracy with a balanced behavior on both prediction classes. As shown in Figure 6.7-a, using 7 weather observations at origin and destination airports the predictors reaches an accuracy of 85.8%, an on-time recall of 84.7% and a delay recall of 86.9%. Figure 6.7-b shows the Receiver Operating Characteristic (ROC) curves for the seven models (obtained varying the number of weather observations at origin and destinations) whose performances are shown in Figure 6.7-a. The ROC curves show that all the models are much more accurate than a random classifier. As expected, this good behavior improves considering a higher number of ob-



(a) (b)

Fig. 6.7. Performance indicators vs number of weather observations at origin and destination airports (a) and ROC curves (b).

servations, but using more than 7 observations does not lead to significant increase of performance.

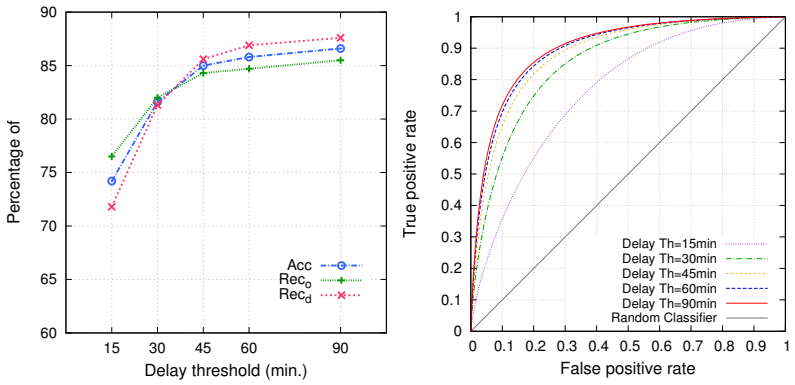
As an additional remark on Figures 6.6 and 6.7, we can note that even without considering weather conditions (i.e., with zero observations at origin and destination airports), the model achieves an accuracy of 69.1%. This is due to the fact that the classification algorithm is able to infer frequent delay patterns from flight information stored in the AOTP dataset, like origin and destination airports (some airports are more subject to delays than others and the algorithm learns that), day of week (delays vary during the week), departure and arrival block time (e.g., delays on early morning flights are less frequent).

We also trained models using weather observations taken every 3 hours, rather than every hour. Table 6.8 reports the predictor performance obtained using 3 observations at both origin and destination airports with 3-hour steps (i.e., at scheduled time, 3 and 6 hours before), compared with that obtained using 7 hourly observations at both origin and destination airports. As shown in the table, using observations every 3 hours does not significantly reduce the predictor performance.

Weather observation considered	*Acc*	Rec_o	Rec_d
3 w.o. at origin (0,3,6) + 3 w.o. at destination (0,3,6)	84.8%	84.3%	85.2%
7 w.o. at origin (0-6) + 7 w.o. at destination (0-6)	85.8%	84.7%	86.9%

Table 6.8. Predictor performance obtained using: 3 observations with 3-hour step (first row); 7 observations every hour (second row).

A second set of experiments was carried out to evaluate the predictor performance by varying the delay threshold. Figure 6.8-a shows the results, obtained using $D2$ as input dataset and considering 7 weather observations at both origin and destination airports.



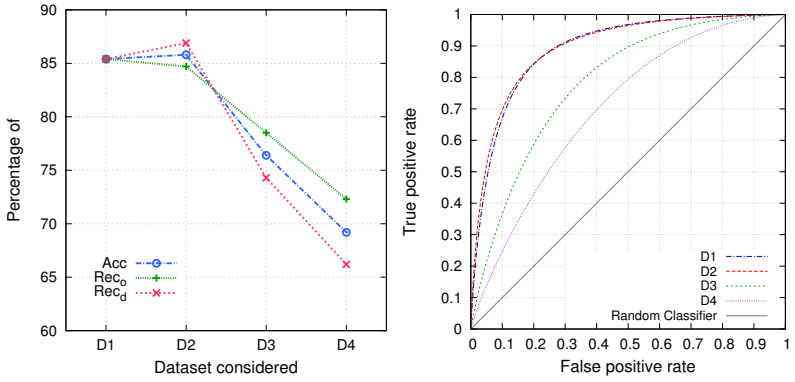
(a) (b)

Fig. 6.8. Performance indicators vs delay threshold (a) and ROC curves (b).

In this case, all the performance indicators improve as the delay threshold increases. For instance, the accuracy passes from 74.2% with a threshold of 15 minutes, to 81.6% with a threshold of 30 minutes, up to 86.6% with a threshold of 90 minutes. Figure 6.8-b shows the ROC curves for five models obtained with $D2$ as input dataset using five delay thresholds (15, 30, 45, 60 and 90 minutes). Also in this case, the ROC curves show that all the models are much more accurate than a random classifier. As expected, this good behavior improves using models with higher delay thresholds.

A third set of experiments was carried out to study the predictor behavior varying the target dataset. Figure 6.9-a shows the results, obtained using 60 minutes as delay threshold and 7 weather observations at both origin and destination airports. Figure 6.9-b provides ROC curves for the models whose performance are shown in Figure 6.9-a. In this case, the ROC curves show that the models trained using $D1$ and $D2$ are much more accurate than a random classifier, but that also using $D3$ and $D4$ leads to better performances than the baseline.

Using $D1$ and $D2$, the predictor achieves almost the same performance, whereas $D2$ includes a greater number of delayed tuples, as described in Table 6.6. Using $D3$ and $D4$, the predictor worsens its performance because they are not appropriate since these target datasets include a greater number of delayed tuples not related to weather.



(a) (b)

Fig. 6.9. Performance indicators vs target dataset (a) and ROC curves (b).

Finally, Table 6.9 reports turnaround times and speedup values of the four data mining phases (*data preprocessing and transformation*, *target data creation*, *modeling*, *evaluation*) when 2, 4, 8 and 12 MapReduce workers are used. The speedup is calculated with respect to the results obtained using 2 workers (i.e., "2x" refers to the use of 4 workers, "4x" to 8 workers and "6x" to 12 workers).

For the *data preprocessing and transformation* phase, the turnaround time decreases from about 3 hours using two workers, to about 35 minutes using 12 workers. Thus, increasing the workers from 2 to 4 (2x), the obtained speedup is 1.9, and it is equal to 5.5 using 12 (6x) workers. For the *target data creation* phase, the turnaround time varies from 2.2 hours using two workers, to 23 minutes using 12 workers. Then the speedup increases respectively from 2 to 5.8. For the *modeling* phase, the turnaround time decreases from 2.5 hours to 25 minutes (with speedup values from 2 to 6), while for the *evaluation* phase, turnaround time decreases from 4.3 hours to 49 minutes (speedup from 1.9 to 5.3). Taking into account the whole data mining process, the turnaround time decreases from 12.2 hours using 2 workers, to 2.2 hours using 12 workers, with a speedup that is very close to linear values (see Figure 6.10). This behavior shows the scalability of the implemented solution that is able to exploit the high-performance features of the Cloud platform.

6.5 Related work

Due to the significant costs for airlines, passengers and society, the analysis and prediction of flight delays have been studied by several research teams. In the following, we discuss the most representative related work and systems.

Some research teams studied and modeled the delay propagation phenomena within an airport network. [51] modeled the US airport network in order

Operation	1x (2 workers) Turn. time	Speed up	2x (4 workers) Turn. time	Speed up	4x (8 workers) Turn. time	Speed up	6x (12 workers) Turn. time	Speed up
Data preprocessing and transformation	03.08.55	-	01:40:52	1.9	00:49:16	3.8	00:34:39	5.5
Target data creation	02:14:06	-	01:06:59	2.0	00:33:19	4.0	00:23:16	5.8
Modeling	02.29.20	-	01:13:12	2.0	00:37:35	4.0	00:24:44	6.0
Evaluation	04.19.28	-	02:14:17	1.9	01:08:51	3.8	00:49:18	5.3
Total	12:11:49	-	06:15:20	1.9	03:09:01	3.9	02:11:57	5.5

Table 6.9. Turnaround time and relative speedup values (calculated with respect to 2 workers) of the four data mining phases.

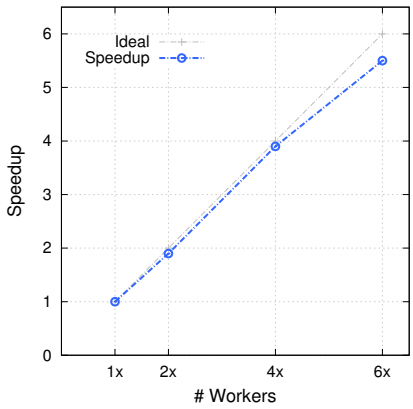


Fig. 6.10. Relative speedup of the whole data mining process.

to study how operational and meteorological issues generate delays that can propagate in the network. The authors developed a simulator to evaluate the effects of airport operations before applying them. Their work highlights that passengers and crew connectivity is a relevant factor that contribute to network congestion. The same authors proposed a similar approach to study delay propagation dynamics in presence of severe meteorological conditions [50]. [116] presented a queuing model for the propagation of delays within a large network of airports, considering the stochastic nature of airports demand and capacity. The goal of the work is to reproduce the trends and behaviors observed in an airport network. [149] used a Bayesian network to estimate the delay propagation in an airport network. Specifically, the authors have investigated and quantified how a flight delay propagates from an airport to others. [5] studied the relationship between the scheduling of the aircraft and crew members and the delay propagation. This work emphasizes how the maximization of aircraft utilization by air-carriers increases the probability of

delay propagation. The main result of this work is a tool for building more robust airline plans.

Another group of related studies investigated how to estimate individual or aggregate variables related to delay for supporting decision making.

[127] described a model to estimate the number of flight delays in an airport at a given time. The authors made use of the WITI system [21], which estimates the total delay in a given airport, based on weather information and actual traffic. [77] applied the WITI metric to the entire NAS, creating the NAS Weather Index (NWX). [108] proposed an aggregate model to analyze departure, en-route and arrival delay for ten major airports in the United States. The work provides several aggregated delay metrics generated averaging the results over 21 days. [150] presented a tool for predicting the generated and absorbed delays at airports. This tool may be used to perform a "what if" analysis by making changes in input factors and observing the predicted effects. [140] presented some machine learning methods to predict the Ground-Delay Programs (GDP) time for a given airport. The GDP is a traffic flow procedure implemented to control the air traffic volume in airports where the airport's acceptance rate being reduced for some reason, such as adverse weather or low visibility. The aim of this work is to improve the planning of GDP duration for supporting air traffic flow management activities.

Our work, differently from those reported here, developed a system able to predict individual flight delay due to weather conditions using information available at the time of prediction. Indeed, the related work discussed above focused on predicting delay propagation in airport network or variables related to delay but not predicting individual flight delay. In addition, some related work like that of [149] use variables that are only available at flight time and not before (i.e., at prediction time).

The work of [117] modeled the US airport network for predicting air traffic delays. Their goal is to predict future departure delays on a particular origin-destination link for a given forecast horizon between 2 and 24 hours. The predictor uses as input variables the delay states of the most influential airports and the global delay state of the entire National Airspace System. As in our work, their predictor uses only variables that are available at time of prediction and the evaluation tests have been performed on balanced datasets where half data are on-time and half delayed flights. However, the system by Rebollo and Balakrishnan, differently from ours, does not use weather data to achieve its goal, which is predicting delay propagation in the airport network independently from cause.

While Rebollo's work focuses on predicting aggregate delays, we focus on predicting individual flight delays. In addition, the prevision horizon of Rebollo's work is limited to a maximum of 24 hours because their predictor needs information about the status of the entire airport network. On the contrary, our work allows a longer prevision horizon because weather forecasts can be available several days in advance (e.g., most online services provide 5-days hourly weather forecasts). Information provided by our predictor could

be used by air traffic management to assist decision, taking into account that accurate weather forecasts are necessary for the reliability of predictions and would be likely to have a strong impact on the efficiency of aviation resulting in costs reduction [78]. About performance, with a delay threshold of 60 minutes and with a balanced dataset, Rebollo et al. work reaches an accuracy of 81% and a delay recall of 76.4%, while our model achieves an accuracy of 85.8% and a delay recall of 86.9%.

Finally, we mention *FlightCaster*, a commercial tool that aims to predict individual flight delays. There are not scientific papers about this tool, but as declared by the founders [52], it seems to reach an 85% of precision and 60% of recall without class balancing, which represent a weak performance if compared to our results.

Table 6.10 summarizes the features of the last two related systems in comparison with our predictor (last row in the table). For each work, the table indicates: (*i*) which is goal of the work; (*ii*) on which data is based the prediction; (*iii*) the performance obtained in the classification problem. As shown in the table, our system achieves a better level of accuracy and delay recall using a balanced dataset.

Related work	Goal	Input data	Performance
Rebollo and Balakrishnan[117]	Delay Propagation in airport network	Aggregate variables presently available	$Acc = 81\%$ $Rec_d = 76.4\%$ (balanced dataset)
FlightCaster [52]	Delay prediction of individual flight	Historical data and weather information	$Pre = 85\%$ $Rec = 60\%$ (unbalanced dataset)
Our work	*Delay prediction of individual flight*	*Historical data and weather information*	$Acc = 85.8\%$ $Rec_d = 86.9\%$ (balanced dataset)

Table 6.10. Related work comparison.

6.6 Conclusions

The main goal of the work presented in this chapter is to predict several days in advance the arrival delay of a scheduled flight due to weather conditions. Accurate prediction of flight delays is an important problem to be addressed given the economic impact of flight delays to both airlines and travelers. In our model, the predicted arrival delay takes into consideration both flight information (origin airport, destination airport, scheduled departure and arrival time) and weather conditions at origin airport and destination airport according to the flight timetable.

Two open datasets of airline flights and weather observations have been analyzed to discover initial insights, evaluate the quality of data, and identify potentially interesting subsets. Then, data cleaning and transformation (joining and balancing operations) have been performed to make data ready for modeling. Finally, a scalable parallel version of the Random Forest data classification algorithm has been developed, iteratively calibrating its settings to optimize results in terms of accuracy and recall. The data preparation and mining tasks have been implemented as MapReduce programs that have been executed on a Cloud infrastructure to achieve scalability.

The results show a high accuracy in prediction of delays above a given threshold. For instance, with a delay threshold of 60 minutes we achieve an accuracy 85.8% and a delay recall of 86.9%. We have obtained such good performance results considering different weather observations at origin and destination airports and selecting flights that are really delayed by weather conditions. Moreover, if weather conditions are not considered the model achieves an accuracy of 69.1%. Therefore, the proposed methodology identifies a very useful pattern of flight delay that may help airlines in reducing delays.

7

Conclusions

In science and business, scientits and professionals analyze huge amounts of data, commonly called Big Data, to extract information and knowledge useful for making new discoveries or for supporting decision processes. This can be done by exploiting Big Data analytics techniques and tools. But it must be also considered that to extract value from such kind of data, novel technologies and architectures have been developed by data scientists for capturing and analyzing data, particularly when large datasets are involved or real-time evaluation is needed. In this scenario, high performance computers, Clouds, and multi-clusters, coupled with parallel and distributed algorithms are commonly to tackle Big Data issues and get valuable information and knowledge in a reasonable time. Cloud computing is probably the most valid and cost-effective solution for high-performance computing on massive number of processors, for supporting Big Data storage and for executing sophisticated data analytic applications. In fact, thanks to elastic resource allocation and high computing power, Cloud computing represents a compelling solution for Big Data analytics, allowing faster data analysis, that means more timely results and then greater data value.

The goal of this thesis was studying, designing and exploiting models, tools, and systems for Big Data analysis, especially on Clouds, to support scalable and distributed knowledge discovery applications. The first part focused on methods and tools for supporting scalable execution of distributed knowledge discovery applications and, in general, solutions for dealing with Big Data issues. The second part presented data analysis applications and methodologies for extracting knowledge from large datasets.

As a first result, we presented an extension for integrating the MapReduce model into the workflow engine provided by Data Mining Cloud Framework (DMCF). More in detail, we described how workflows created using VL4Cloud or JS4Cloud, i.e., the formalisms provided by DMCF for designing workflows, can include MapReduce algorithms and tools. We also described how these workflows are executed in parallel on DMCF to enable scalable data processing on Clouds. By implementing a study case application whose workflow includes

MapReduce computations, we shown how DMCF can be used to easily create workflows that exploit the combined scalability provided by the DMCF workflow languages and by the MapReduce framework.

As a second result, we presented *Geocon*, an open-source service-oriented middleware designed to help developers to implement context-aware mobile applications. Geocon provides a service and a client library for storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments). In order to ensure a high level of decoupling and efficient communication between client and service, the REST model has been adopted. Moreover, given the huge number of users, places, events and resources that may be involved in context-aware mobile applications, Geocon architecture has been designed to scale horizontally on a multiple nodes. To assess the scalability of Geocon in a real-world scenario, we developed a location-aware mobile application, called *GeoconView*, which made use of Geocon for managing information about events. The experimental results show that the latency speedup is basically independent from the number of queries per second, but is significantly higher when the system stores a larger number of events.

As a third result of this thesis work, we presented *G-RoI*, a novel mining technique that exploits the indications contained in geotagged social media items to discover the boundaries of a Place-of-Interest (PoI), commonly called Region-of-Interest (RoI), with a high accuracy. We experimentally evaluated the accuracy of G-RoI in detecting the RoIs, comparing it with three existing techniques: *Circle* (predefined-shapes approach), *DBSCAN* (density-based clustering), and *Slope* (grid-based aggregation). The analysis was carried out on a set of PoIs located in the center of Rome and Paris, characterized by different shapes, areas and densities. The experimental results shown that G-RoI is able to detect more accurate RoIs than existing techniques, regardless of shapes, areas and densities of PoIs, and without being influenced by the proximity of different PoIs.

Finally, we described a predictor of the arrival delay of a scheduled flight due to weather conditions, which is an important problem to be addressed given the economic impact of flight delays to both airlines and travelers. In our model, the predicted arrival delay takes into consideration both flight information (origin airport, destination airport, scheduled departure and arrival time) and weather conditions at origin airport and destination airport according to the flight timetable. Two open datasets of airline flights and weather observations have been analyzed to discover initial insights, evaluate the quality of data, and identify potentially interesting subsets. The data preparation and mining tasks have been implemented as MapReduce programs that have been executed on a Cloud infrastructure to achieve scalability. In particular, a scalable parallel version of the Random Forest data classification algorithm has been developed, iteratively calibrating its settings to optimize results in terms of accuracy and recall. The experimental evaluations shown a high ac-

curacy in prediction of delays and a good scalability, with a speedup that is very close to linear values. That means the proposed methodology identifies a very useful pattern of flight delay that may help airlines in reducing delays. With regard to this research, there is still much needed. In particular, it could be interesting to evaluate how the accuracy of weather forecast influences the accuracy of the prediction. Moreover, the possibility to improve the accuracy of our predictor could be investigated by integrating it with other models for predicting delay propagation in the airport network.

References

1. Abramova, V., Bernardino, J., Furtado, P.: Which nosql database? a performance overview. Open Journal of Databases (OJDB) **1**(2), 17–24 (2014)
2. Adams, P.: Grouped: How small groups of friends are the key to influence on the social web. New Riders (2011)
3. Agarawala, A., Greenberg, S., Ho, G.: The context-aware pill bottle and medication monitor. In: Video Proceedings and Proceedings supplement of the Sixth International Conference on Ubiquitous Computing. University of Calgary (2004)
4. Agrawal, D., Bernstein, P., Bertino, E., Davidson, S., Dayal, U., Franklin, M., Gehrke, J., Haas, L., Halevy, A., Han, J., et al.: Challenges and opportunities with big data. a community white paper developed by leading researchers across the united states. Computing Research Association, Washington (2012)
5. AhmadBeygi, S., Cohn, A., Guan, Y., Belobaba, P.: Analysis of the potential for delay propagation in passenger airline networks. Journal of air transport management **14**(5), 221–236 (2008)
6. Allan, S., Beesley, J., Evans, J., Gaddy, S.: Analysis of delay causality at newark international airport. In: 4th USA/Europe Air Traffic Management R&D Seminar. Santa Fe, USA (2001)
7. Altomare, A., Cesario, E., Comito, C., Marozzo, F., Talia, D.: Trajectory pattern mining for urban computing in the cloud. Transactions on Parallel and Distributed Systems (IEEE TPDS) (2016)
8. Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., Peterson, E., Sherry, L., Trani, A.A., Zou, B.: Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the united states (2010)
9. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**(4), 469–483 (1996)
10. Barga, R., Gannon, D., Reed, D.: The client and the cloud: Democratizing research computing. Internet Computing, IEEE **15**(1), 72–75 (2011)
11. Belcastro, L., Di Lieto, G., Lackovic, M., Marozzo, F., Trunfio, P.: Geocon: A Middleware for Location-Aware Ubiquitous Applications, pp. 234–243. Springer International Publishing, Cham (2016)
12. Belcastro, L., Marozzo, F., Talia, D., Trunfio, P.: Programming visual and script-based big data analytics workflows on clouds. In: Big Data and High

Performance Computing, *Advances in Parallel Computing*, vol. 26, pp. 18–31. IOS Press (2015)
13. Bermingham, L., Lee, I.: Spatio-temporal sequential pattern mining for tourism sciences. Procedia Computer Science **29**(0), 379 – 389 (2014). 2014 International Conference on Computational Science
14. Beyer, M.A., Laney, D.: The importance of big data: a definition. Stamford, CT: Gartner pp. 2014–2018 (2012)
15. Black, D., Clemmensen, N.J., Skov, M.B.: Pervasive computing in the supermarket: Designing a context-aware shopping trolley. Int. J. Mob. Hum. Comput. Interact. **2**(3), 31–43 (2010)
16. Blanas, S., Patel, J.M., Ercegovac, V., Rao, J., Shekita, E.J., Tian, Y.: A comparison of join algorithms for log processing in mapreduce. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 975–986. ACM (2010)
17. Bowers, S., Ludäscher, B., Ngu, A.H., Critchlow, T.: Enabling scientificworkflow reuse through structured composition of dataflow and control-flow. In: Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on, pp. 70–70. IEEE (2006)
18. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
19. Cai, G., Hio, C., Bermingham, L., Lee, K., Lee, I.: Sequential pattern mining of geo-tagged photos with an arbitrary regions-of-interest detection method. Expert Systems with Applications **41**(7), 3514 – 3526 (2014)
20. Cai, L., Zhu, Y.: The challenges of data quality and data quality assessment in the big data era. Data Science Journal **14**, 2 (2015)
21. Callaham, M., DeArmon, J., Cooper, A., Goodfriend, J., Moch-Mooney, D., Solomos, G.: Assessing nas performance: Normalizing for the effects of weather. In: 4th USA/Europe Air Traffic Management R&D Symposium (2001)
22. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The dl-lite family. Journal of Automated Reasoning **39**(3), 385–429 (2007)
23. Capra, L., Emmerich, W., Mascolo, C.: Carisma: context-aware reflective middleware system for mobile applications. IEEE Transactions on Software Engineering **29**(10), 929–945 (2003). DOI 10.1109/TSE.2003.1237173
24. Cattell, R.: Scalable sql and nosql data stores. ACM SIGMOD Record **39**(4), 12–27 (2011)
25. Cesario, E., Congedo, C., Marozzo, F., Riotta, G., Spada, A., Talia, D., Trunfio, P., Turri, C.: Following soccer fans from geotagged tweets at fifa world cup 2014. In: Proc. of the 2nd IEEE Conference on Spatial Data Mining and Geographical Knowledge Services, pp. 33–38. Fuzhou, China (2015). ISBN 978-1- 4799-7748-2
26. Cesario, E., Iannazzo, A.R., Marozzo, F., Morello, F., Riotta, G., Spada, A., Talia, D., Trunfio, P.: Analyzing social media data to discover mobility patterns at expo 2015: Methodology and results. In: 2016 International Conference on High Performance Computing Simulation (HPCS), pp. 230–237 (2016)
27. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS) **26**(2), 4 (2008)

28. Che, D., Safran, M., Peng, Z.: Database Systems for Advanced Applications: 18th International Conference, DASFAA 2013, International Workshops: BDMA, SNSM, SeCoP, Wuhan, China, April 22-25, 2013. Proceedings, chap. From Big Data to Big Data Mining: Challenges, Issues, and Opportunities, pp. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
29. Cheng, Y.: Mean shift, mode seeking, and clustering. Pattern Analysis and Machine Intelligence, IEEE Transactions on **17**(8), 790–799 (1995)
30. Chu, C., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. Advances in neural information processing systems **19**, 281 (2007)
31. Corradi, A., Fanelli, M., Foschini, L.: Implementing a scalable context-aware middleware. In: 2009 IEEE Symposium on Computers and Communications, pp. 868–874 (2009)
32. Crandall, D.J., Backstrom, L., Huttenlocher, D., Kleinberg, J.: Mapping the world's photos. In: Proceedings of the 18th International Conference on World Wide Web, WWW '09, pp. 761–770. ACM, New York, NY, USA (2009)
33. Das, S., Sismanis, Y., Beyer, K.S., Gemulla, R., Haas, P.J., McPherson, J.: Ricardo: integrating r and hadoop. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 987–998. ACM (2010)
34. De Mauro, A., Greco, M., Grimaldi, M.: What is big data? a consensual definition and a review of key research topics. In: AIP Conference Proceedings, vol. 1644, pp. 97–104 (2015)
35. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008)
36. Deelman, E., Singh, G., Su, M.H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., et al.: Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Scientific Programming **13**(3), 219–237 (2005)
37. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., da Silva, R.F., Livny, M., et al.: Pegasus, a workflow management system for science automation. Future Generation Computer Systems **46**, 17–35 (2015)
38. Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., et al.: Orange: data mining toolbox in python. Journal of Machine Learning Research **14**(1), 2349–2353 (2013)
39. Dey, A.K.: Understanding and using context. Personal Ubiquitous Comput. **5**(1), 4–7 (2001)
40. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-computer interaction **16**(2), 97–166 (2001)
41. Dongarra, J., et al.: The international exascale software project roadmap. International Journal of High Performance Computing Applications (2011)
42. Draft, N.: Big Data Interoperability Framework: Volume 1, Definitions. NIST Special Publication, Information Technology Laboratory, Gaithersburg **23** (2014)
43. Driver, C., Clarke, S.: An application framework for mobile, context-aware trails. Pervasive and Mobile Computing **4**(5), 719–736 (2008)

44. ECMA: Ecma-262: ECMAscript Language Specification. Fifth edition. ECMA (European Association for Standardizing Information and Communication Systems) (2009)
45. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.H., Qiu, J., Fox, G.: Twister: A runtime for iterative mapreduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10, pp. 810–818. ACM, New York, NY, USA (2010)
46. Ekanayake, J., Pallickara, S., Fox, G.: Mapreduce for data intensive scientific analyses. In: eScience, 2008. eScience'08. IEEE Fourth International Conference on, pp. 277–284. IEEE (2008)
47. Ester, M., peter Kriegel, H., S, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. pp. 226–231. AAAI Press (1996)
48. Federal meteorological handbook no. 1 (fmh-1), "surface weather observations and reports". `http://www.ofcm.gov/fmh-1/pdf/FMH1.pdf` (2005)
49. Ferrari, L., Rosi, A., Mamei, M., Zambonelli, F.: Extracting urban patterns from location-based social networks. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks, LBSN '11, pp. 9–16. ACM, New York, NY, USA (2011)
50. Fleurquin, P., Ramasco, J.J., Eguiluz, V.M.: Data-driven modeling of systemic delay propagation under severe meteorological conditions. ArXiv e-prints (2013)
51. Fleurquin, P., Ramasco, J.J., Eguiluz, V.M.: Systemic delay propagation in the us airport network. Scientific reports **3** (2013)
52. How flightcaster squeezes predictions from flight data. `http://www.datawrangling.com/how-flightcaster-squeezes-predictions-from-flight-data` (2009)
53. Gajendran, S.K.: A survey on nosql databases. University of Illinois (2012)
54. Gantz, J., Reinsel, D.: Extracting value from chaos. IDC iview **1142**, 1–12 (2011)
55. Gartner, Inc.: Text Analytics Gartner IT Glossary (2016). URL `http://www.gartner.com/it-glossary/text-analytics`
56. Gartner, Inc.: What is Big Data? Gartner IT Glossary (2016). URL `http://www.gartner.com/it-glossary/big-data`
57. Gerber, M.S.: Predicting crime using twitter and kernel density estimation. Decision Support Systems **61**, 115 – 125 (2014)
58. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07, pp. 330–339. ACM, New York, NY, USA (2007)
59. Giardine, B., Riemer, C., Hardison, R.C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., et al.: Galaxy: a platform for interactive large-scale genome analysis. Genome research **15**(10), 1451–1455 (2005)
60. Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News **33**(2), 51–59 (2002)
61. Gormley, C., Tong, Z.: Elasticsearch: The Definitive Guide. O'Reilly Media, Inc. (2015)

62. de Graaff, V., de By, R.A., van Keulen, M., Flokstra, J.: Point of interest to region of interest conversion. In: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL'13, pp. 388–391. ACM, New York, NY, USA (2013)
63. Gu, Y., Grossman, R.L.: Sector and sphere: the design and implementation of a high-performance data cloud. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **367**(1897), 2429–2445 (2009)
64. Gupta, A., Kalra, A., Boston, D., Borcea, C.: Mobisoc: a middleware for mobile social computing applications. Mobile Networks and Applications **14**(1), 35–52 (2009)
65. Hansen, P.C.: Analysis of discrete ill-posed problems by means of the L-Curve. SIAM Review **34**(4), 561–580 (1992)
66. Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U.: The rise of big data on cloud computing: Review and open research issues. Information Systems **47**, 98 – 115 (2015)
67. Hiden, H., Woodman, S., Watson, P., Cala, J.: Developing cloud applications using the e-science central platform. Phil. Trans. R. Soc. A **371**(1983), 20120,085 (2013)
68. IBM: What is big data? Bringing big data to the enterprise? (2016). URL `https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html`
69. IBM, Zikopoulos, P., Eaton, C.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, 1st edn. McGraw-Hill Osborne Media (2011)
70. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: Distributed data-parallel programs from sequential building blocks. SIGOPS Oper. Syst. Rev. **41**(3), 59–72 (2007)
71. Jean-Pierre, D.: Oracle: Big data for the enterprise. USA: Oracle Corporation (2013)
72. Julien, C., Roman, G.C.: Egospaces: facilitating rapid development of context-aware mobile applications. IEEE Transactions on Software Engineering **32**(5), 281–298 (2006). DOI 10.1109/TSE.2006.47
73. Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B.P., Maechling, P.: Data sharing options for scientific workflows on amazon ec2. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–9. IEEE Computer Society (2010)
74. Kacsuk, P., Farkas, Z., Kozlovszky, M., Hermann, G., Balasko, A., Karoczkai, K., Marton, I.: Ws-pgrade/guse generic dci gateway framework for a large variety of user communities. Journal of Grid Computing **10**(4), 601–630 (2012)
75. Kisilevich, S., Keim, D., Rokach, L.: A novel approach to mining travel sequences using collections of geotagged photos. In: M. Painho, M.Y. Santos, H. Pundt (eds.) Geospatial Thinking, *Lecture Notes in Geoinformation and Cartography*, vol. 0, pp. 163–182. Springer Berlin Heidelberg (2010)
76. Kisilevich, S., Mansmann, F., Keim, D.: P-dbscan: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos. In: Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application, COM.Geo '10, pp. 38:1–38:4. ACM, New York, NY, USA (2010)

77. Klein, A., Jehlen, R., Liang, D.: Weather index with queuing component for national airspace system performance assessment. In: FAA/Eurocontrol ATM Sem. Barcelona, Spain (2007)
78. Klein, A., Kavoussi, S., Lee, R.S.: Weather forecast accuracy: study of impact on airport capacity and estimation of avoidable costs. In: Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009) (2009)
79. Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al.: Handling imbalanced datasets: A review. GESTS International Transactions on Computer Science and Engineering **30**(1), 25–36 (2006)
80. Kranjc, J., Podpečan, V., Lavrač, N.: Clowdflows: a cloud based scientific workflow platform. In: Machine Learning and Knowledge Discovery in Databases, pp. 816–819. Springer (2012)
81. Kurashima, T., Iwata, T., Irie, G., Fujimura, K.: Travel route recommendation using geotags in photo sharing sites. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pp. 579–588. ACM, New York, NY, USA (2010)
82. Lee, R., Wakamiya, S., Sumiya, K.: Urban area characterization based on crowd behavioral lifelogs over twitter. Personal and Ubiquitous Computing **17**(4), 605–620 (2013)
83. Lee, S., Park, H., Shin, Y.: Cloud computing availability: multi-clouds for big data service. In: Convergence and Hybrid Information Technology, pp. 799–806. Springer (2012)
84. Lemieux, A.: Geotagged photos: a useful tool for criminological research? Crime Science **4**(1), 3 (2015)
85. Li, A., Yang, X., Kandula, S., Zhang, M.: Cloudcmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp. 1–14. ACM (2010)
86. Lourenço, J.R., Cabral, B., Carreiro, P., Vieira, M., Bernardino, J.: Choosing the right nosql database for the job: a quality attribute evaluation. Journal of Big Data **2**(1), 1–26 (2015)
87. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the kepler system. Concurrency and Computation: Practice and Experience **18**(10), 1039–1065 (2006)
88. Luo, J., Feng, H.: A web-based framework for lightweight context-aware mobile applications. International Journal of Database Theory and Application **9**(4), 119–134 (2016)
89. Lyubimov, D., Palumbo, A.: Apache Mahout: Beyond MapReduce. Chapman and Hall/CRC (2016)
90. Maheshwari, K., Rodriguez, A., Kelly, D., Madduri, R., Wozniak, J., Wilde, M., Foster, I.: Enabling multi-task computation on galaxy-based gateways using swift. In: 2013 IEEE International Conference on Cluster Computing (CLUSTER), pp. 1–3. IEEE (2013)
91. Malcher, M., Aquino, J., Fonseca, H., David, L., Valeriano, A., Endler, M.: A middleware supporting adaptive and location-aware mobile collaboration. In: Mobile Context Workshop: Capabilities, Challenges and Applications, Adjunct Proceedings of UbiComp (2010)

92. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: A system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, pp. 135–146. ACM, New York, NY, USA (2010)
93. Marciani, G., Piu, M., Porretta, M., Nardelli, M., Cardellini, V.: Real-time analysis of social networks leveraging the flink framework. In: Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, DEBS '16, pp. 386–389. ACM, New York, NY, USA (2016)
94. Margaret Rouse: Data Analytics (DA) Definition (2016). URL `http://searchdatamanagement.techtarget.com/definition/data-analytics`
95. Margaret Rouse: In-memory Analytics Definition (2016). URL `http://searchbusinessanalytics.techtarget.com/definition/in-memory-analytics`
96. Marozzo, F., Talia, D., Trunfio, P.: A cloud framework for parameter sweeping data mining applications. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pp. 367–374. IEEE (2011)
97. Marozzo, F., Talia, D., Trunfio, P.: Using clouds for scalable knowledge discovery applications. In: Euro-Par Workshops, *Lecture Notes in Computer Science*, vol. 7640, pp. 220–227. Rhodes Island, Greece (2012)
98. Marozzo, F., Talia, D., Trunfio, P.: Using clouds for scalable knowledge discovery applications. In: European Conference on Parallel Processing, pp. 220–227. Springer (2012)
99. Marozzo, F., Talia, D., Trunfio, P.: A Cloud framework for Big Data Analytics workflows on Azure. In: Post-Proc. of the High Performance Computing Workshop 2012, vol. 23, pp. 182–191. IOS Press, Cetraro, Italy (2013)
100. Marozzo, F., Talia, D., Trunfio, P.: Using clouds for scalable knowledge discovery applications. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7640 LNCS**, 220–227 (2013)
101. Marozzo, F., Talia, D., Trunfio, P.: Js4cloud: script-based workflow programming for scalable data analysis on cloud platforms. Concurrency and Computation: Practice and Experience **27**(17), 5214–5237 (2015)
102. Martin, A., Brito, A., Fetzer, C.: Real-time social network graph analysis using streammine3g. In: Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, DEBS '16, pp. 322–329. ACM, New York, NY, USA (2016)
103. Mavroidis, I., Papaefstathiou, I., Lavagno, L., Nikolopoulos, D.S., Koch, D., Goodacre, J., Sourdis, I., Papaefstathiou, V., Coppola, M., Palomino, M.: Ecoscale: Reconfigurable computing and runtime system for future exascale systems. In: 2016 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 696–701 (2016)
104. Meijer, E.: The World according to LINQ. Commun. ACM **54**(10), 45–51 (2011)
105. Mell, P.M., Grance, T.: Sp 800-145. the nist definition of cloud computing. Tech. rep., National Institute of Standards & Technology, Gaithersburg, MD, United States (2011)
106. Möller, R., Neumann, B.: Ontology-Based Reasoning Techniques for Multimedia Interpretation and Retrieval, pp. 55–98. Springer London, London (2008)

107. Moniruzzaman, A.B.M., Hossain, S.A.: Nosql database: New era of databases for big data analytics - classification, characteristics and comparison. CoRR **abs/1307.0191** (2013)
108. Mueller, E.R., Chatterji, G.B.: Analysis of aircraft arrival and departure delay characteristics. In: AIAA aircraft technology, integration and operations (ATIO) conference (2002)
109. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on, pp. 124–131 (2009)
110. Nyce, C., CPCU, A.: Predictive analytics white paper. American Institute for CPCU. Insurance Institute of America pp. 9–10 (2007)
111. Otte, E., Rousseau, R.: Social network analysis: a powerful strategy, also for the information sciences. Journal of Information Science **28**(6), 441–453 (2002)
112. Owen, S., Anil, R., Dunning, T., Friedman, E.: Mahout in Action. Manning Publications Co., Greenwich, CT, USA (2011)
113. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. big'web services: making the right architectural decision. In: Proceedings of the 17th international conference on World Wide Web, pp. 805–814. ACM (2008)
114. Podpečan, V., Zemenova, M., Lavrač, N.: Orange4WS environment for service-oriented data mining. The Computer Journal (2011)
115. Poslad, S., Laamanen, H., Malaka, R., Nick, A., Buckle, P., Zipl, A.: CRUMPET: creation of user-friendly mobile services personalised for tourism. In: 3G Mobile Communication Technologies, 2001. Second International Conference on (Conf. Publ. No. 477), pp. 28–32 (2001)
116. Pyrgiotis, N., Malone, K.M., Odoni, A.: Modelling delay propagation within an airport network. Transportation Research Part C: Emerging Technologies **27**, 60–75 (2013)
117. Rebollo, J.J., Balakrishnan, H.: Characterization and prediction of air traffic delays. Transportation Research Part C: Emerging Technologies **44**, 231–241 (2014)
118. Richardson, L., Ruby, S.: RESTful web services. "O'Reilly Media, Inc." (2008)
119. River Logic, Inc.: What is Prescriptive Analytics? (2016). URL `http://www.riverlogic.com/technology/prescriptive-analytics`
120. Rodriguez, M.A., Neubauer, P.: The graph traversal pattern. CoRR **abs/1004.1001** (2010)
121. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications, WMCSA '94, pp. 85–90 (1994)
122. Schroeck, M., Shockley, R., Smart, J., Romero-Morales, D., Tufano, P.: Analytics: The real-world use of big data. IBM Global Business Services pp. 1–20 (2012)
123. Shahrivari, S.: Beyond batch processing: Towards real-time and streaming big data. CoRR **abs/1403.3375** (2014)
124. Shi, J., Mamoulis, N., Wu, D., Cheung, D.W.: Density-based place clustering in geo-social networks. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD '14, pp. 99–110. ACM, New York, NY, USA (2014)

125. Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Virtual infrastructure management in private and hybrid clouds. IEEE Internet Computing **13**(5), 14–22 (2009)
126. Spyrou, E., Mylonas, P.: Analyzing Flickr metadata to extract location-based information and semantically organize its photo content. Neurocomputing **172**, 114 – 133 (2016)
127. Sridhar, B., Wang, Y., Klein, A., Jehlen, R.: Modeling flight delays and cancellations at the national, regional and airport levels in the united states. In: 8th USA/Europe ATM R&D Seminar, Napa, California (USA) (2009)
128. Stonebraker, M.: Sql databases v. nosql databases. Commun. ACM **53**(4), 10–11 (2010)
129. Tai, A., Wei, M., Freedman, M.J., Abraham, I., Malkhi, D.: Replex: A scalable, highly available multi-index data store. In: 2016 USENIX Annual Technical Conference (USENIX ATC 16), pp. 337–350. USENIX Association, Denver, CO (2016)
130. Takacs, G.: Predicting flight arrival times with a multistage model. In: Big Data (Big Data), 2014 IEEE International Conference on, pp. 78–84 (2014)
131. Talia, D.: Workflow systems for science: concepts and tools. ISRN Software Engineering **2013** (2013)
132. Talia, D., Trunfio, P.: Service-oriented distributed knowledge discovery. Chapman and Hall/CRC (2012). ISBN 978-1-4398-7531-5
133. Talia, D., Trunfio, P., Marozzo, F.: Data Analysis in the Cloud. Elsevier (2015). ISBN 978-0-12-802881-0
134. Tan, K.L., Cai, Q., Ooi, B.C., Wong, W.F., Yao, C., Zhang, H.: In-memory databases: Challenges and opportunities from software and hardware perspectives. SIGMOD Rec. **44**(2), 35–40 (2015)
135. Thomas, J.J., Cook, K.A.: A visual analytics agenda. Computer Graphics and Applications, IEEE **26**(1), 10–13 (2006)
136. Van Setten, M., Pokraev, S., Koolwaaij, J.: Context-aware recommendations in the mobile tourist application compass. In: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 235–244. Springer (2004)
137. Verikas, A., Gelzinis, A., Bacauskiene, M.: Mining data with random forests: A survey and results of new tests. Pattern Recognition **44**(2), 330–349 (2011)
138. Voronoi, G.: Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs. Journal für die reine und angewandte Mathematik **134**, 198–287 (1908)
139. Vukotic, A., Watt, N., Abedrabbo, T., Fox, D., Partner, J.: Neo4j in action. Manning (2015)
140. Wang, Y., Kulkarni, D.: Modeling weather impact on ground delay programs. Tech. rep., SAE Technical Paper (2011)
141. Wang, Z., Chu, Y., Tan, K., Agrawal, D., El Abbadi, A., Xu, X.: Scalable data cube analysis over big data. CoRR **abs/1311.5663** (2013)
142. Want, R., Hopper, A., Falcão, V., Gibbons, J.: The active badge location system. ACM Trans. Inf. Syst. **10**(1), 91–102 (1992)
143. White, T.: Hadoop: The Definitive Guide, 1st edn. O'Reilly Media, Inc. (2009)
144. Wilde, M., Hategan, M., Wozniak, J.M., Clifford, B., Katz, D.S., Foster, I.: Swift: A language for distributed parallel scripting. Parallel Computing **37**(9), 633–652 (2011)

145. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., et al.: The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic acids research p. gkt328 (2013)
146. Wozniak, J.M., Wilde, M., Foster, I.T.: Language features for scalable distributed-memory dataflow computing. In: Data-Flow Execution Models for Extreme Scale Computing (DFM), 2014 Fourth Workshop on, pp. 50–53 (2014)
147. Wu, X., Zhu, X., Wu, G.Q., Ding, W.: Data mining with big data. IEEE Transactions on Knowledge and Data Engineering **26**(1), 97–107 (2014)
148. Xin, R.S., Rosen, J., Zaharia, M., Franklin, M.J., Shenker, S., Stoica, I.: Shark: Sql and rich analytics at scale. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD '13, pp. 13–24. ACM, New York, NY, USA (2013)
149. Xu, N., Donohue, G., Laskey, K.B., Chen, C.H.: Estimation of delay propagation in the national aviation system using bayesian networks. In: 6th USA/Europe Air Traffic Management Research and Development Seminar. Citeseer (2005)
150. Xu, N., Sherry, L., Laskey, K.B.: Multifactor model for predicting delays at us airports. Transportation Research Record: Journal of the Transportation Research Board **2052**(1), 62–71 (2008)
151. Yin, Z., Cao, L., Han, J., Luo, J., Huang, T.S.: Diversified trajectory pattern ranking in geo-tagged social media. In: SDM, pp. 980–991. SIAM (2011)
152. You, L., Motta, G., Sacco, D., Ma, T.: Social data analysis framework in cloud and mobility analyzer for smarter cities. In: Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on, pp. 96–101 (2014)
153. Yu, C.C., Chang, H.P.: Personalized location-based recommendation services for tour planning in mobile tourism applications. In: International Conference on Electronic Commerce and Web Technologies, pp. 38–49. Springer (2009)
154. Yuan, J., Zheng, Y., Zhang, L., Xie, X., Sun, G.: Where to find my next passenger. In: Proceedings of the 13th International Conference on Ubiquitous Computing, UbiComp '11, pp. 109–118. ACM, New York, NY, USA (2011)
155. Zhang, H., Chen, G., Ooi, B.C., Tan, K.L., Zhang, M.: In-memory big data management and processing: A survey. IEEE Transactions on Knowledge and Data Engineering **27**(7), 1920–1948 (2015)
156. Zheng, Y.T., Zha, Z.J., Chua, T.S.: Mining travel patterns from geotagged photos. ACM Trans. Intell. Syst. Technol. **3**(3), 56:1–56:18 (2012)