Energy Efficient Management and Scheduling of Computational Resources

Mehdi Sheikhalishahi

November 2012

UNIVERSITA DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Department of Electronics, Informatics and Systems

University of Calabria

A thesis submitted for the degree of
PhD in Operations Research
SSD: MAT/09

Coordinator
Prof. Lucio Grandinetti
[illegible]

Supervisor
Prof. Lucio Grandinetti
[illegible]

Candidate
Mehdi Sheikhalishahi
[illegible]

Gestione e Schedulazione "Energy Efficient" di Risorse Computazionali

Mehdi Sheikhalishahi

Novembre 2012

UNIVERSITA DELLA CALABRIA

Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

Dipartimento di Elettronica, Informatica e Sistemistica

Università della Calabria

Tesi sottomessa per il dottorato di ricerca in

Ricerca Operativa
SSD: MAT/09

Coordinatore
Prof. Lucio Grandinetti
[illegible]

Supervisore
Prof. Lucio Grandinetti
[illegible]

Candidato
Mehdi Sheikhalishahi
[illegible]

UNIVERSITY OF CALABRIA

Italy

Energy Efficient Management and Scheduling of Computational Resources

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Department of Electronics, Informatics and Systems

by

Mehdi Sheikhalishahi

2012

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would not have made it here without the help, guidance, and support of many wonderful people. To all of them, I extend my most sincere and heartfelt thanks.

To professor Lucio Grandinetti, my PhD supervisor, besides steering my dissertation work to a successful conclusion, Lucio has supported me tirelessly like a father, standing up for me when I needed it most.

I am grateful to Borja Sotomayor for his support and help, development of Haizea. Also, I gratefully acknowledge Ignacio Martin Llorente, Wolfgang Gentzsch, Jose Luis Vazquez-Poletti, and Natalie jean Bates for their insightful comments at the beginning and during my PhD work.

Last but not the least, to my parents who have sacrificed their life for my life, health, and progress.

ABSTRACT OF THE DISSERTATION

Energy Efficient Management and Scheduling of Computational Resources

by

Mehdi Sheikhalishahi

Doctor of Philosophy in Department of Electronics, Informatics and Systems

University of Calabria, Italy, 2012

Professor Lucio Grandinetti, Chair

0.1 Abstract: English version

Information Technology has a big impact on climate change and global warming as the current and future challenges of the world. In this PhD thesis, we focus on green computing to address these challenges.

With the advent of new technologies, we need to explore these technologies to see if they can address green computing. The first stages of this thesis uncover the flux of technologies and its impact on green landscape. We envision the duality of green computing with technological trends in other fields of computing such as high performance computing (HPC) and cloud computing from one hand; and economy, and business, on the other hand.

In addition, IT's energy consumption and sustainability impact are expected to increase. Contemporary technologies are moving towards Intelligent Computation in order to optimize resource and energy consumption. Intelligent Computations are done with the techniques and mechanisms of new computing technologies such as

infrastructure-adapted-to-applications, virtual machine consolidation, optimization algorithms, hardware and software co-design, and application profiling to optimize resource consumption, and pay-as-you-go business model to reduce costs, etc.

In green world, research on minimizing energy and resource consumption through algorithmic and software techniques such as monitoring, power-aware consolidation, scheduling, optimization algorithms such as bin packing as well as user/application profiling and debugging are other aspects of addressing energy efficiency. These facets of Intelligent Computation constitute the main parts of resource management. In the second part of this Thesis, we address some aspects of Intelligent Computation as part of resource management and scheduling.

More specifically, we introduce the problem of resource management more precisely and describe computing system problems from the resource management point of view. We explore resource management components. Then, we model resource contention metric that is one of the main metrics explored in this thesis. We develop effective energy aware consolidation policies. Finally, we propose a novel autonomic energy efficient resource management and scheduling algorithm.

0.2 Sommario: Italian version

La Tecnologia dellInformazione (Information Technology- IT) ha un grande impatto sul cambiamento climatico e sul riscaldamento globale come le attuali e future sfide del mondo. In questa tesi di dottorato, si focalizza lattenzione sul Green Computing per affrontare queste sfide. E necessario, quindi, esplorare le nuove tecnologie al fine di comprendere se possono supportare il Green Computing.

Le prime fasi di questa tesi sono focalizzate ad analizzare le nuove tecnologie e limpatto che esse hanno sul panorama ecologico. Si configura pertanto il dualismo tra il Green Computing e le nuove tendenze tecnologiche in settori informatici come il Calcolo ad Alte Prestazioni (HPC) ed il Cloud Computing da una parte; l'economia e le imprese, dall'altro. Inoltre, poich il consumo energetico dovuto allIT e limpatto della sostenibilit ecologica sono destinati ad aumentare, le tecnologie moderne (Contemporary Technologies) si stanno orientando verso il Calcolo Intelligente per ottimizzare il consumo di risorse e di energia. Il Calcolo Intelligente supportato dalle tecniche e dai meccanismi di nuove tecnologie informatiche, come lInfrastruttura-Adattata-alle-Applicazioni (Infrastructure-Adapted-to-Applications), il consolidamento della macchina virtuale, gli algoritmi di ottimizzazione, la configurazione e progettazione hardware e software, lanalisi di applicativi per ottimizzare il consumo di risorse ed il modello pay-as-you-go per ridurre i costi, etc.

In un mondo ecologicamente sostenibile, la riduzione del consumo energetico e delle risorse, tramite tecniche algoritmiche e software quali il monitoraggio, il risanamento consapevole della potenza, la schedulazione, gli algoritmi di ottimizzazione (Problemi di Caricamento (Bin Packing), analisi delle applicazioni e degli utenti e debugging) sono altri possibili modi per gestire lefficienza energetica. Tutti questi sono comunque aspetti del Calcolo Intelligente e costituiscono le parti principali della gestione delle risorse.

Nella seconda parte di questa tesi, si rivolge lattenzione su alcuni aspetti del Calcolo Intelligente che riguardano la gestione e la pianificazione/schedulazione di risorse. In particolare, si introduce, in maniera dettagliata ed approfondita, il problema di gestione delle risorse e si descrivono i problemi presentati dai sistemi di calcolo sotto questo punto di vista. Si esplorano, quindi, le componenti relative alla gestione delle risorse e si modella la cosiddetta metrica Resource Contention come una delle principali metriche considerate in questo lavoro. Si sviluppano politiche efficaci per il risanamento consapevole dellenergia ed infine, si propone un innovativo algoritmo per la gestione e la schedulazione efficiente delle risorse energetiche.

CHAPTER 1

Introduction

Climate change and global warming are the two most important challenges of the earth. These problems pertain to a general increase in the world temperatures caused by increased amounts of carbon dioxide around the earth.

Researchers in various fields of science and technology in recent years started to carry out research in order to address these problems by developing environmentally friendly solutions. Green IT and in particular green computing are two new terms introduced in ICT community to address the aforementioned problems.

The foundation of this PhD thesis is on green computing and its goals are manifold. With the advent of new technologies, we need to explore these technologies to see if they can address challenges of the world. Chapter 2 uncovers the flux of technologies and its impact on green landscape. More specifically, it envisiones the duality of green computing with technological trends in other fields of computing such as high performance computing (HPC) and cloud computing from one hand; and economy, and business, on the other hand.

On the other hand, with the explosive growth of Internet-enabled cloud computing and HPC centers of all types, IT's energy consumption and sustainability impact are expected to continue climbing well into the future. Green IT recognizes this problem and efforts are under way in both industry and academia to address it.

Contemporary technologies are moving towards Intelligent Computation in order to optimize resource and energy consumption. Intelligent Computations are done with the techniques and mechanisms of new computing technologies such as infrastructure-adapted-to-applications, virtual machine consolidation, hardware and software co-design, and application profiling to optimize resource consumption, and pay-as-you-go business model to reduce costs, etc.

In green world, research on minimizing energy and resource consumption through algorithmic and software techniques such as monitoring, power-aware consolidation, scheduling as well as user/application profiling and debugging are other aspects of addressing energy efficiency. These solutions are the facets of Intelligent Computation. These facets of Intelligent Computation constitute the main parts of resource management. In this thesis, we address some aspects of Intelligent Computation as part of resource management and scheduling.

The next chapters of this PhD thesis are structured as follows. In chapter 3, we introduce the problem of resource management more precisely and describe computing system problems from the resource management point of view. Chapter 4 discusses and to some extent explores resource management components. Then, chapter 5 models resource contention metric that is one of the main metrics explored in this thesis. In chapter 6, we develop effective energy aware consolidation policies. Finally, chapter 7 proposes a novel autonomic energy efficient resource management and scheduling algorithm. Chapter 8 presents our conclusions and future work.

This thesis makes the following contributions in the field:

- Evolution of resource management and scheduling as new technologies, paradigms, etc. emerge

- Establishing relationships within and between various layers of resource management

- Being general rather than special purpose solution
- Defining resource contention metric as a new performance metric to evaluate scheduling algorithms

CHAPTER 2

Green Computing: a dual technology for cloud computing and HPC

As computing and communication continue to grow, servers, networks and data centers consume more and more energy. For example, IT resources in the US consume more than 1.5% of the total electricity consumption. The power consumption of the data centers in the US in 2006 was 1.5% of the total energy consumed at a cost of more than \$4.5B. With the expected 30-fold increase in data traffic over the next decade, the overall power consumption of data centers and networks will become an issue of vital importance to the IT and telecommunications industries.

While computing and information technologies consume energy, they also enable productivity enhancements and directly contribute to energy efficiency. Some even argue that computing and information technologies are vital, significant and critical in moving towards a low-carbon future [Sma10].

In this chapter, we envision the duality of green computing with technological trends in other fields of computing such as HPC and cloud computing, business, and economy [SG12].

2.1 Proposition

Our proposition in this chapter is that green computing is a dual technology for computing and communication technologies such as HPC and cloud computing. This means that green computing solutions will drive the development of HPC and cloud computing; on the other hand, HPC solutions and cloud computing solutions will drive the development of green computing. Therefore, we may say these technologies are dual to each other, thus we envision green computing as a dual technology for HPC and cloud computing.

Green computing is a challenge for the future of HPC, for example to reach exascale computing, we need huge amounts of energy to operate an exascale system.

On the other hand, HPC provides solutions for green computing and climate change. Complicated processes of new sources of energy, need exascale computing for modeling and simulation.

In addition, these contemporary technologies are moving toward Intelligent Computation in order to optimize resource and energy consumption without losing performance. Intelligent Computations are done with the techniques and mechanisms of new computing technologies such as hardware and software co-design, application profiling, and virtual machine consolidation to optimize resource consumption, and pay-as-you-go business model to reduce costs, etc.

In sum, if we solve the challenges and problems of HPC and exascale, we would implicitly solve many challenges in green computing such as access to new sources of energy, etc. Similarly, if we solve green computing challenges, we would solve HPC and cloud computing challenges such as the challenge of exascale power and building huge cloud data centers. This is the duality theorem we observe among energy efficiency, green computing and other technological trends in computing and commu-

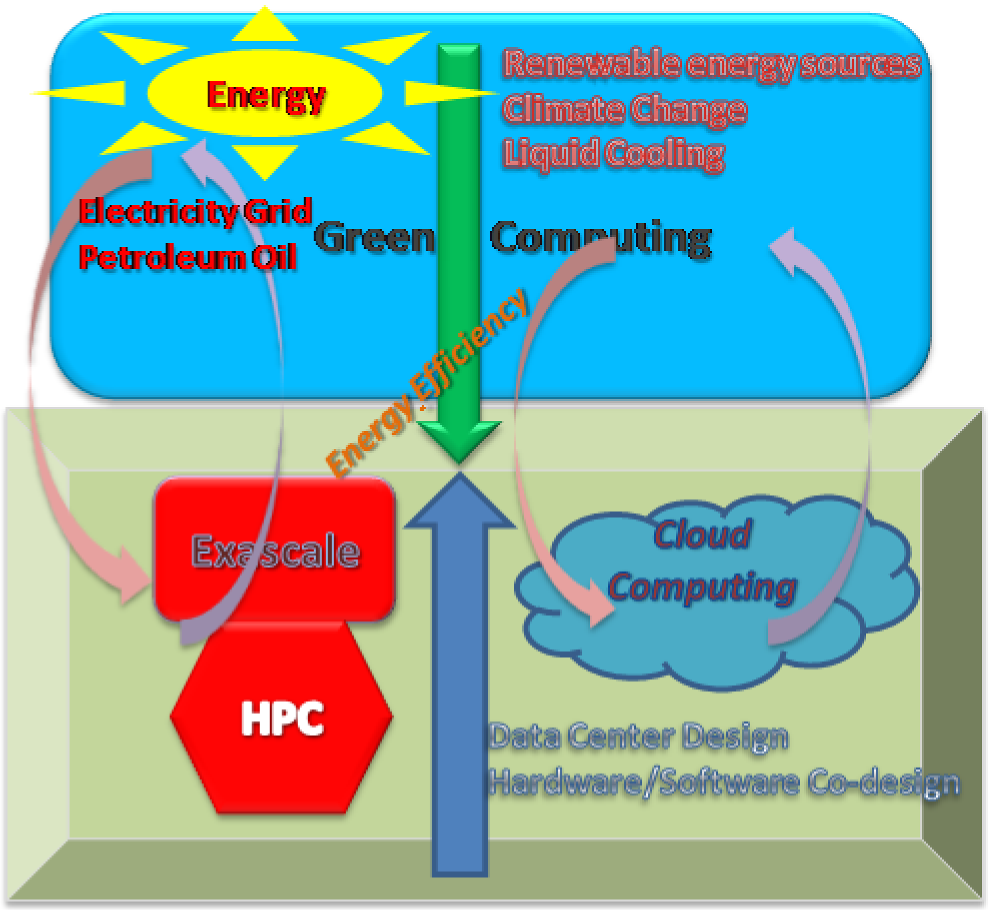


Figure 2.1: Green computing: a dual technology for computing and communication technologies

nication technologies. This proposition is illustrated in Figure 2.1.

In the remainder of this chapter, we go over some research work at the intersection of cloud computing and HPC with green computing to realize our observation and proposition of duality.

2.2 Cloud Computing

In the cloud computing world, IT capabilities are delivered on the fly and on-demand through the Internet when the need arises instead of drawing from desktop computers. Many design and architectural patterns [AFG10] are emerging around cloud computing that makes it difficult to fit everything into a prefect definition.

In a formal definition, we denote cloud computing as an extreme specialization (hyperspecialization) and at the same time a general purpose model of information technology to digest the flux of future IT. More specifically, we connote the real definition of cloud computing as the convergence of the following essential and ideal characteristics of various distributed computing technologies: (1) infinity: large scale data centers; (2) outsourcing: remote, over the Internet; (3) utility: pay per use; (4) economy of scale; (5) self-service: self-provisioning, on the fly;(6) multi-tenancy (7) on-demand; (8) elasticity: scalability, autoscaling; (9) *-abilities: availability, reliability, scalability, sustainability, etc.

The nature and the anatomy of cloud computing is totally green. From the users' point of view, cloud computing makes their IT life easier. They can access IT services without spending too much effort and energy, compared to the other IT models (that are dedicated). On the other hand, from the scientific point of view, the characteristics of cloud computing and deployment models make it a ubiquitous green IT paradigm.

In the following, we review cloud's characteristics and cloud deployment options to highlight that they are cost-, and energy-effective. In addition, from this highlight we observe green computing contribution to cloud computing.

2.2.1 Infinity

Clouds approach infinity [AFG09], we observe this character in some statistics about computing: (1) Estimated market for servers used in the cloud in 2014, according to IDC will be \$6.4 billion. In 2010 the market was \$3.8 billion. (2) Projected market for cloud computing in 2014, according to Gartner will be \$149 billion. (3) Power requirement for new megadata centers in 2011, according to Microsoft was 50 megawatts (MW) while in the dotcom era, it was 1 to 2 MW. (4) Estimated share of work done with virtualization software on new servers in 2014, according to IDC will be 70%. (5) Maximum number of servers that Google says can be managed with new software it's developing, will be 10 million. (6) Based on data from McKinsey and IDC, approximate number of servers in use globally 60 is million.

These statistics with their big numbers demonstrate that power requirement and power management become a major challenge in massive scale systems infrastructures such as cloud computing data centers. For instance, currently Microsoft and Google have data centers with 48 MW and 85 MW power requirement, respectively.

In sum, we conclude that green computing contributes to the development of cloud computing. Green computing's goal is to provide energy and to increase energy efficiency and reduce resource consumption of the cloud infrastructures.

2.2.2 Outsourcing: remote, over the Internet

Cloud computing is often represented with a diagram that contains a cloud-like shape indicating a layer where the responsibility for a service goes from user to provider. This is outsourcing of a service from the user side to the provider side. It is similar to the electrical power we receive each day, cloud computing provides subscribers and users with access to provisioned services.

This feature makes users' life to get access to services easier and without any effort. Outsourcing implicitly contributes to energy efficiency and green computing.

2.2.3 Utility: pay per use

In cloud computing, payment of resource consumption is just like utilities that are paid for by the hour. In other words, service consumption is metered or measured.

When demand for a service varies with time, and when demand is unknown in advance, the utility character of a cloud will definitely make it economically cost-saving.

In the traditional computing models, provisioning a data center for the peak load that it must sustain a few days per month leads to under utilization at other times that is not energy efficient. Instead, cloud computing lets an organization pay by the hour for computing resources, potentially leading to cost savings even if the hourly rate to rent a machine from a cloud provider is higher than the rate to own one.

In addition, with the cost associativity of cloud computing, computations can finish faster, since using 1000 EC2 machines for 1 hour costs the same as using 1 machine for 1000 hours.

2.2.4 Economy of scale

Cloud providers purchase data center infrastructure equipments such as hardware, network, and bandwidth much cheaper than a regular business. The construction and operation of extremely large-scale, commodity-computer datacenters at low-cost locations is the key necessary enabler of cloud computing [AFG09].

The cost of electricity, network bandwidth, operations, software, and hardware is decreased in the factors of 5 to 7 at very large economies of scale. These factors, combined with statistical multiplexing to increase utilization compared to a private cloud, meant that cloud computing could offer services below the costs of a medium-sized data center and yet still make a good profit [AFG09].

Economies of scale for consumers means if you need more storage, it is just a matter of upping your subscription costs with your provider, instead of buying new equipment. If you need more computational cycles, you need not buy more servers; rather you just buy more from your cloud provider.

Economy of scale is cost-effective and directly contributes to energy efficiency and green computing.

2.2.5 Self service: self-provisioning, on the fly

User self-provisioning is one the greatest benefits of the cloud. With that, you have the ability to get applications up and running in a fraction of the time you would need in a conventional scenario.

In a cloud, users prepare their resources on the fly by themselves. On the other hand, infrastructure is adapted to the applications. For instance, Amazon Web Services (AWS) makes it possible for anyone with an Internet connection and a credit card to access the same kind of world class computing systems that Amazon uses to run its $34 billion per year retail operation.

This cloud's characteristic significantly reduces time to solutions and access to services, thus it is energy efficient and green.

2.2.6 Multi-tenancy

In traditional data centers, computing systems suffer from under utilization of computing power and networking bandwidth. Multi-tenant is a business model that provides a secure, exclusive virtualized computing environment in which servers, databases, and other resources are shared by multiple user companies in a cloud environment.

Public cloud is hosted, operated and managed by a third-party vendor from one or more data centers. Since the services are offered to multiple customers with the aforementioned characters, it is multiple tenants over a common infrastructure.

With virtualization and multi-tenant feature, the resources in the cloud are not devoted to specific usages and users. At one time, a cloud resource can be used for an application by a user, and at another time it can be used for another diverse application by the same user or another user.

Virtualization and shared hosting technologies coupled with multicore servers are the enablers of cloud infrastructures to support a large number of disparate applications running simultaneously on multicore servers. Moreover, these technologies enable VM consolidation, infrastructure-adapted-to-applications (Intelligent Computation), and other resource optimization techniques.

This multi-usage feature of cloud addresses energy efficiency and green computing challenges by significantly improving resource utilization and minimizing resource waste.

2.2.7 Public cloud

When a cloud is made available in a pay-as-you-go manner to the public, we call it a public cloud [AFG09]; the service being sold is Utility Computing. Public cloud resembeles the Internet; current examples of public Utility Computing include AWS, Google AppEngine, and Microsoft Azure.

Institutions rely on public clouds for providing their services. This is a high level prespective of Green IT. Public clouds refer to infrastructure provided to the general public by a large industry selling cloud services. Amazon's cloud offering would fall in this category. These services are on a pay-as-you-go basis and can usually be purchased using a credit card.

In public cloud, everything is outsourced to outside of a corporation administrative domain such as security management and day to day operations. These outsourced tasks are handled by the provider of public cloud service offering. Hence, the customer of the public cloud service offering has much lower degree of control and oversight of the physical and logical security aspects of a private cloud.

Public cloud providers optimize energy consumption as a way to offer competitive prices. This is Green IT at a low level powered by market rules. Moreover, some public cloud providers could relay to other public cloud providers for certain services.

HPC applications in public cloud is an economic and green approach. In [VBL12], an astronomy application with a parameter sweep profile from the next mission to Planet Mars is ported to a public cloud environment. This application requires a big quantity of computing resources in a short term with punctual situations. Authors proposed a model for optimal execution of that application on a public cloud infrastructure in terms of time and cost metrics.

2.2.8 Federation and hybrid cloud

Hybrid clouds refer to two or more cloud infrastructures that operate independently but are bound together by technology compliance to enable application portability.

With hybrid cloud and cloud federation, we can build green services. For each service, we should find the most green cloud provider, that is, a provider that consumes the least amount of energy to provide a service.

These deployment models are energy efficient and directly contribute to green computing challenges by improving energy consumption.

2.3 High Performance Computing

HPC is the use of advanced parallel processing systems (usually, above a teraflop or 1012 floating-point operations per second) for running complicated and huge processes quickly, efficiently, and reliably.

The energy (power), cooling, and data center design are the three most prominent challenges of future HPC systems. Power has become the pre-eminent design constraint for future HPC systems. Moreover, the energy cost becomes an increasingly important factor.

We believe green and performance objectives converge to the same point. In this direction, HPC provides solutions for green computing. We cite some research work on the areas of energy and climate that are at the intersection of HPC and green computing.

HPC is used to provide energy, and vice versa energy is required to operate HPC systems. In particular, as exascale systems are emerging, they would need huge amounts of electricity to sustain. HPC systems of today need about 10MW power requirement. For instance, Sequoia, the IBM BlueGene/Q system installed at the Department of Energys Lawrence Livermore National Laboratory, is on the top of the Top500 list with 16.32 petaflop/s [Top12]. It is also one of the most energy efficient systems on the list with 7890kW power requirement. In addition, the target for the future exascale systems' power requirement is 20MW.

ExxonMobil [Exx12] predicts the outlook for energy enabled with supercomputing, ExxonMobil's global energy outlook projects through 2040. The projections indicate that, at that time, the world's population will be $\tilde{8}$ billion, roughly 25% higher than today. Along with this population rise will be continuing economic growth. This combination of population and economic growth will increase energy demand by over 50% versus 2000.

HPC drives the process of finding new sources of energy. Effective technology solutions to the energy challenges rely on modeling complicated processes and that in turn will lead to a strong need for supercomputing. Two examples of the supercomputing need in the oil business are seismic approaches for finding petroleum and petroleum reservoir fluid-flow modeling (also known as ”reservoir simulation”).

HPC and exascale systems are the main tools to solve climate change challenges. HPC will become an even more critical resource to help the broader research community to develop solutions to potential environmental impacts.

On the electricity grid front, power grid applications are exploiting HPC and networking. The first workshop [FPG10] on this challenge discussed the use of HPC and networking for power grid applications. Technological and policy changes make this an urgent priority.

From the new hardware technologies point of view, Nvidia evolutionary systems, i.e. GPUs, Flash technology, and special-purpose hardware systems drive the development of green and energy efficient architecture and systems.

From the processing technologies point of view, Nvidia evolutionary systems, i.e. GPUs, will be among the main building blocks of future exascale systems. Accelerator-based supercomputers now occupy the top eight slots of the most recent the Green500 list, [Gre12], so we observe that heterogeneous computing is extensively examined as a means for achieving computing system energy efficiency.

Hardware and software co-design (Intelligent Computation) is another key solution for green computing and energy efficiency. For instance, GreenFlash [LBL10] is a radically new approach to application-driven hardware and software co-design inspired by design principles from the consumer electronics marketplace. The GreenFlash system addresses some aspects of green computing and exascale system for specific applications. In addition, it significantly reduces cost and accelerates the development cycle of exascale systems. The application that is the target of Green Flash is ”The Earthś Atmosphere at Kilometer Scales.”

In [MMO09], authors presented a hardware/software co-tuning (Intelligent Computation) as a novel approach for HPC system design. In this chapter, in order to substantially improve hardware area and power efficiency, traditional architecture space exploration is tightly coupled with software auto-tuning. Their approach demonstrates that co-tuning would be considered as a key driver for the next generation HPC system design.

In the past decades of HPC design, the performance and speed were the main metrics in supercomputer design. This focus on performance as the ultimate metric has caused other metrics such as energy consumption, reliability, sustainability, availability, and usability to be largely ignored. In addition, this particular emphasis has led to the emergence of supercomputers that consume huge amounts of electrical power and produce so much heat that overpriced cooling facilities must be constructed to ensure proper operation. Therefore, there has been an extraordinary increase in the total cost of ownership (TCO) of a supercomputer.

To that end, the Green500 [Gre12] is developed to encourage sustainable supercomputing by raising awareness to the energy efficiency of such systems. The purpose of the Green500 is to provide a ranking of the most energy efficient supercomputers in the world.

2.4 Green Computing

Green computing is a growing research topic in recent years to address climate and energy challenges. In this section, we review green computing solutions such as liquid cooling that address HPC and cloud computing challenges. In addition, green computing duality with economy and business is envisioned.

2.4.1 Liquid Cooling

Liquid cooling is a heat removal method based on liquids as the heat conductor, as opposed to air cooling that uses air for heat removal. The main mechanism for liquid cooling is convective heat transfer. Liquid cooling is used for cooling large industrial equipments in grid power plants, petroleum stations and recent computer systems. The thermal energy that the water extracts from the equipment can be reused for heating purposes.

In recent years, liquid cooling is used to develop cost-effective cooling mechanism of computing systems. In particular, large computer systems like big data centers and supercomputers such as Eurotech's HPC offering will exploit this cooling mechanism to reduce cooling and operational costs, and reduce construction space.

Aurora is an innovative system of Eurotech's HPC offering [Eur12]. Aurora exploits liquid cooling to design a high density system, that leads to a high energy efficient HPC system. Such a high density is achieved with extensive usage of liquid cooling, carried out with metal plates coupled to all Aurora boards, having coolant flow inside them. Aurora modules have no moving parts, no attached modules, and are hot-replaceable, being connected with miniaturized couplings to the heat removal hydraulic infrastructure. In addition, it features Intel's latest generation processors and chipsets.

Hardcore Computer company [Sol12] has a patented product called Liquid Blade. Liquid Blade is designed to operate in almost any environment and does not require costly, overly-complicated environmental control systems. According to Hardcore Computer, Liquid Blade can deliver up to an 80% reduction in data center cooling costs and reduce construction and on-going operating costs by up to 25%. Liquid Blade has a significant, undeniable and immediate reduction in TCO.

2.4.2 Renewable energy sources

On the renewable energy sources front, the demand for clean energy generation is driving the use of non-dispatchable power sources such as solar and wind [Ren12]. According to a survey by [Ren12], 85 percent of survey respondents voted for more renewable energy options. With 49 percent saying they would have no problem digging deeper into their pockets to support companies committed to renewable energy in the product manufacturing process. Renewable energy sources such as solar and geothermal energies may be used to power on large data centers like HPC data centers.

2.4.3 Intelligent Computation

In green world, research on minimizing energy and resource consumption through algorithmic and software techniques such as monitoring, power-aware consolidation, scheduling as well as user/application profiling and debugging are other aspects of addressing energy efficiency. These solutions are the facets of Intelligent Computation.

Contemporary computing technologies, HPC and in particular cloud, are moving toward Intelligent Computation in order to optimize resource and energy consumption without losing performance. The other aspects of Intelligent Computation are hardware and software co-design, infrastructure-adapted-to-applications, virtual machine consolidation, and pay-as-you-go business model.

Hw/sw co-design is mostly being used for exascale development. It implies user/vendor development for a particular application space. Hw/sw co-design is a facet of HPC Intelligent Computation.

On the other hand, in the cloud computing world, infrastructure is adapted to the applications by the means of enabling technologies i.e virtualization and shared hosting technologies coupled with multicore processors.

While Hw/sw co-design from HPC and Infrastructure-adapted-to-applications from cloud are different approaches, they both contribute to energy efficiency. This observation demonstrates how different techonologies develop different mechanisms.

2.4.4 Economy and Business

Green computing and climate change have a direct and significant impact on economy and business.

More and more companies consider green policy and climate change risks as part of their business policy and strategy. One of the main reasons is that companies with long-term climate change risks and opportunities in their business policy will gain strategic advantage over their competitors [BAR12].

In addition, investors fund businesses and companies that consider the environmental policies and the environmental risks as part of their business strategies. For instance, there were floods in Thailand last year with the total cost of $15 billion to $20 billion that had big impact on the sell and services of companies. The floods had negative impact on the automotive supply industry such as the automaker Daimler. In addition, the floods resulted in shortages of critical IT components and materials of Hewlett-Packard and Dell.

Last but not the least, insurance companies have to take into account climate change risks in their risk analysis and calcucations. They supply investors with information about potential environmental risks, costs and benefits.

Similarly, good economy contributes to the development of green computing and climate change solutions.

2.5 Conclusion

In this section, we summarize our thesis statement and supporting evidence by highlighting all the main findings in the previous sections about cloud computing and HPC. We close this chapter by a set of conclusions and implications about future research.

In this chapter, we have envisioned the duality of green computing with HPC and cloud computing, business, and economy. Green computing solutions drive the development of HPC, cloud computing, economy, and business and vice versa.

Cloud computing as an extreme specialization (hyperspecialization) and as a general purpose model of information technology has a green anatomy. The essential characteristics of cloud computing i.e., infinity, outsourcing, utility business model, elasticity, on-demand, self-provisioning, multi-tenancy, and *-abilities, and deployment models make it an ideal and ubiquitous green IT paradigm. In summary, cloud significanlty reduces effort, energy and resource consumptions.

HPC is used to provide energy, and vice versa energy is required to operate HPC systems. In particular, as exascale systems are emerging, they would need huge amounts of electricity (20MW) to sustain. In addition, HPC drives the process of finding new sources of energy.

On the other hand, HPC and exascale systems are the main tools to solve climate change challenges. HPC will become an even more critical resource to help the broader research community to develop solutions to potential environmental impacts.

By all means, contemporary technologies are moving toward Intelligent Computation in order to optimize resource and energy consumption.

In addition, green computing and climate change have a direct impact on economy and business. Green policy and climate change risks are a part of companies' business policy and strategy. In addition, investors fund businesses and companies that consider the environmental policies and the environmental risks as part of their business strategies.

Energy consumption is a suitable metric to measure how much a technology, a system, etc. are green. Diagram 2.2 illustrates the green duality gap based on energy metric. The difference between supplied energy, i.e. real energy consumption, and demanded energy is part of this gap. In addition to this difference, all the parameters, factors, etc. we have mentioned in this chapter (as technology advantages) should be reflected in the green duality gap formulation. Technology disadvantages has a negative impact on green gap.

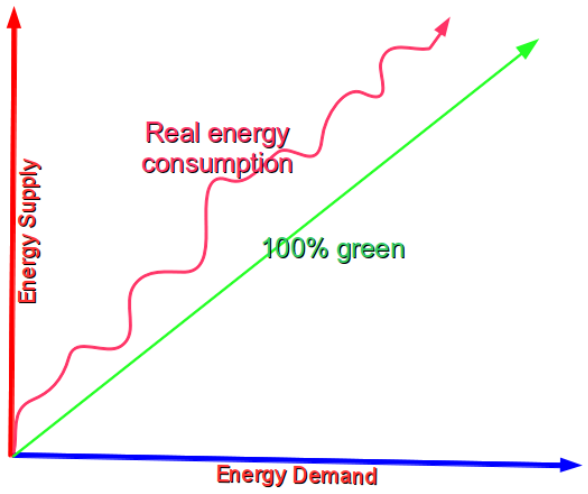


Figure 2.2: Duality gap among green energy and real energy consumption of a system

As much as a technology's or a system's green gap approaches to zero, it becomes greener. The future research in green computing would need to address this gap.

CHAPTER 3

The problem of energy efficient resource management and scheduling

In this chapter, we explore resource management as a facet of Intelligent Computation. Resource management as the main middleware management software system plays a central role in addressing computing system problems and issues. Algorithmic and software techniques such as monitoring, virtual machine power-aware consolidation and scheduling are intelligent aspects of addressing energy efficiency.

At the first sight of exploring green technologies and capabilities, we find power management operations such as DVFS and IDLE-states capabilities at the processor level and virtualization technology at the middleware layer of computing systems as immediate solutions to address energy efficiency goals i.e. energy consumption minimization and heat-dissipation (limitations in wasted energy-removal).

On the other hand, energy efficient operations, new technologies for green computing, and emerging computing paradigms should be exploited at the resource management to coordinate multiple elements within a system for manifold objectives.

We believe green and performance objectives converge to the same point. In the future of resource management systems, we should take into account this note as we demonstrate this insight in this work. In addition, resource management design and architecture should evolve according to the advances in contemporary technologies, computing paradigms, and energy efficient operations to provide new techniques, algorithms, etc. For instance, a comparison between cloud and other paradigms provides some guidelines and insights in the design and development of resource management components. Economic model and accuracy of allocations' parameters (requests') are the main two different characteristics of HPC and cloud paradigms which highly impact scheduling.

In the remainder of this thesis, we explore resource management and scheduling components in detail by taking into account contemporary emerging technologies, computing paradigms, energy efficient operations, relationships within and between various components, etc. in order to evolve these components to address computing system problems in more efficient ways [SG11b].

In computing systems from resource management and scheduling point of view in general, there are the main problems and issues such as Low Utilization, Overload, Poor Performance, Resource Contention, etc., in addition, if we consider Energy Efficient computing (Green IT) the High Energy Consumption is another issue. Moreover, sustainability and reliability are other issues to be addressed by resource management to some extent.

In this thesis, in order to address these problems components of resource management system are explored in detail to seek new developments and evolutions and in parallel this process exploits contemporary emerging technologies, computing paradigms, energy efficient operations, etc. to define, design and develop new metrics, techniques, mechanisms, models, policies, and algorithms. Furthermore, finding and establishing relationships within and between various components is a key consideration in this development process.

In other words, the solution to address the aforementioned problems and issues is all about to answer complex resource management questions which start by When, Which and Where with the help of well established relationships. For instance, Which types of applications might be consolidated together in a server?, When some workloads should be migrated to other servers?, Where a workload should be placed?, and lots of other general and specific questions.

In the next section of this chapter, we review the scheduling problem as it is one the main part of this thesis. Then, some related work on VM-based resource management and energy efficint schduling are presented.

3.1 Scheduling in computing system

The scheduling definition is as follows: a group of independent tasks have to be scheduled on a compute resource e.g. a cluster or a supercomputer, without a deadline i.e. best-effort request or within a given deadline or other types of quality of service such as advance reservation. The compute resource is characterized by a number of processing nodes and a limited storage. Tasks are not known in advance by the scheduler, and are assumed to be ordered in some list (as in a queue) and presented one by one to the scheduler according to this list. Each task has specific resource requirements for its execution, for example, duration, number of required processing nodes, which become known when the task is managed by the scheduler. In grid paradigm, dataset is considered as another resource requirement, so in addition to availability of compute resource, dataset availability must be satisfied in order to schedule a task. Similarly, in different computing paradigms we may have some other additional criteria to be considered.

In a compute resource, jobs arrive over time and they become known at their release date, i.e., the moment the job is presented, its characteristics get available. Scheduling in a computing system is not happening only once, since scheduler cannot wait until all jobs arrive to the system and then schedule all of them together. If a scheduler waits for this time, it must wait for an infinite time, since always new jobs arrive to the system. Therefore, in any computing system we have *online scheduling*. In online scheduling, the scheduling decision for the job has to be taken before the next job is presented. The scheduling decision is irreversible, i.e., decisions of the past are irreversible. At any time all currently available jobs are at the disposal of the decision maker. The scheduling decision for a job may be delayed.

The notion of *online algorithm* is intended to formalize the realistic scenario where the algorithm has not access to the whole input instance, unlike the *offline algorithms*, but it learns the input as soon as it becomes available. There are many contributions in the literature about online algorithms [SWW95] [BE05] for some basic results concerning online scheduling on parallel processors.

In other words, online scheduling can be seen as scheduling with incomplete information at certain points, decisions have to be made without knowing the complete instance depending on the way how new information becomes known.

3.1.1 Maximizing resource utilization by packing techniques

Optimization techniques are at the heart of scheduling. Packing techniques such as bin packing and rectangle packing are the most used operation research methods in computing system scheduling.

Due to multiple resource dimensions in computing systems, resource allocation problem is related to the multi-dimensional version of bin packing, or vector packing. Vector packing is bin packing with multi-dimensional items and bins. In order to make efficient use of all the available system resources, the scheduling algorithm must be able to maintain a job working set which fully utilizes all resources. At the core of this scheduling problem is a *d*-capacity bin-packing problem where each system resource is represented by a capacity in the bin and the requirements of each waiting job are represented by the d capacities of an item in the input list.

The rectangle (or two-dimensional vector) packing problem [WHL02] consists in orthogonally packing a subset of a set of rectangular-shaped boxes, without overlapping, into a single bounding rectangular area, maximizing the ratio between the area occupied by the boxes and the total available area.

In the literature, the offline and the online versions of the rectangle packing problem have been investigated. Although in the offline version all the problem data are known in advance, in the online version, where the over list paradigm is considered, the boxes (with their dimensions) arrive from a list without any knowledge on further boxes. In particular, the boxes along with their dimensions are known one by one. When a new box is presented, it is to be decided if it can be placed into a free rectangular subarea of the bounding rectangular area ,i.e. it has to be accepted or rejected.

The online problem is to accept (place) or reject the incoming boxes, maximizing the ratio between the area occupied by the boxes and the total available area.

Most of the contributions in the literature are devoted to the offline problem that is solved using several approaches based on optimal algorithms. The basic formulation issues and solution procedures for the two-dimensional cutting stock problems were presented in [HS91]. Optimal algorithms for the orthogonal two-dimensional cutting were proposed in [Bea85] and in [HS91] but such techniques may be nonpractical for large instances. Heuristic approaches have been considered in [WHL02] and [HT01] where rejection is also concerned.

In [CGI04] and [CGI03] authors presented online grid scheduling algorithms. The local scheduling problem is modeled as rectangle packing problems. In all these works, one dimension is processor and the other dimension is time.

Vector packing has been studied from both a theoretical standpoint (i.e., guaranteed algorithms) and a pragmatic one (i.e., efficient algorithms). Versions of standard greedy algorithms (First Fit, Best Fit, Worst Fit, Next Fit, etc.) have been proposed for vector packing [KM77] [MCT77].

Previous work in d-capacity bin-packing algorithms analyzed extensions of single capacity bin-packing. These extended algorithms are oblivious to the additional capacities, however, and do not scale well with increasing d. In [LKK99], Leinberger et al. proposed MCB (Multi-Capacity Bin Packing), a particular vector packing algorithm. It uses the additional capacity information to provide better packing. Authors show how these algorithms might lead to better multi-resource allocation and scheduling solutions. We refer to [SSV10] for an extensive review of the literature on vector packing algorithms.

In [IL99] a fully dynamic bin packing (for one-dimensional) proposed by Ivkovic and Lloyd, a formulation where items may arrive or depart at discrete time intervals and the goal is to minimize the maximum number of bins required while limiting re-packing. The packing may be arbitrarily rearranged to accommodate arriving and departing items.

In all, offline scheduling scenario in terms of bin-packing problem is the NP-hard problem [GJ90] as well.

We argue that packing techniques such as bin packing or rectangle packing to be unimportant in online scheduling. First, since we do not have knowledge about future jobs. Second, a packing technique can provide optimal packing for a scheduling cycle, however if we consider the next scheduling cycle (or consecutive cycles), it might fail to propose an optimal packing for both scheduling cycles (or all consecutive cycles), as we see that in recent work to resolve this they used migration, suspend and resume techniques provided by virtualization technology. However, virtualization techniques are not a prefect approach since these operations are heavy and they interrupt computations, consume other resources such as bandwidth, etc. In addition, at every scheduling cycle they repeat these operations, again and again.

On the other hand, because of NP-complexity of packing techniques, they cannot be exploited in the scheduling algorithms. In fact, schedules should be computed quickly, because of the online nature of the problem. As a result, usually non-guaranteed algorithms (i.e., heuristics) that perform well in practice, hopefully close to the offline bound, are used.

3.2 Related Work

In this decade many approaches have been proposed and developed to address the problem of energy efficient computing with special attention on energy consumption optimization, utilization improvement as well as system's reliability. In general, we classify them into the following categories:

- Policies: e.g. Consolidation based policies
- Pricing strategies: The aim is utilization improvement and maximizing profit by attracting more users through offering cheaper prices [Sot10] [Ama10]. If we consider energy efficiency, pricing workloads based on computing system state e.g. offering cheaper prices for applications that will lead to less energy consumption (or higher performance) based on the current cloud status (workloads and resources) compared to the others, are some ideas to come up with. This is also true for Quality of Service (e.g. by price)
- Admission control: In this component of scheduling, again according to state, the scheduler could make decision to accept or reject a job to achieve some goals [Sot10]. For example, accept jobs which lead to utilization improvement and less energy consumption based on the current cloud status (workloads and

resources)

- Algorithms based on energy aware operations: These algorithms incorporate energy aware operations of hardware components that are green e.g. DVFS and cpuidle to improve scheduling metrics such as utilization e.g. by combining energy aware operations we can have some optimization such as increasing/decreasing frequency of processors would result in extension of or reduction of jobs running times, so that we can fill out free spaces in availability window of scheduler. For instance, authors in [LWY09] present an efficient scheduling algorithm to allocate virtual machines in a DVFS-enabled cluster by dynamically scaling the supplied voltages

- Dynamic load balancing algorithms: Algorithms and policies to migrate workloads through heavy migrate operation. Paper [Her09] presents dynamic workload consolidation via load balancing implemented by migration operation

Broadly speaking, we argue the importance of research on the first three categories. The others are not able to approximate global or local optimality in a distributed system, they are better suited for a system i.e. on a physical machine whereas policies and pricing strategies are part of a distributed resource management which could provide some sort of optimality in a distributed system. In the literature, there are many works on the aforementioned topics that most of them are special purpose.

First, we review research works mainly on policies from the scheduling point of view in distributed systems. Eucalyptus [Nur08], Nimbus [Nim10] and Usher [MGV07] as Virtual Infrastructure Managers do not support scheduling policies to dynamically consolidate or redistribute VMs. On the other hand, Scheduling component of OpenNebula [SML09] [Ope10] (in Haizea mode) provides various policies in host selection and admission control.

Research works [DMR09] and [VAN08] are able to dynamically schedule the VMs across a cluster based on their CPU, Memory and Network utilization. Similarly, scheduling algorithms in [Bob07] provide dynamic consolidation and redistribution of VMs for managing performance and SLA i.e. service level agreements violations. VMware's Distributed resource scheduler [VMW10] also performs automated load balancing in response to CPU and Memory pressure.

Latest work by Buyya et. al. [BBA10] presents a very abstract and special purpose vision, challenges, and architectural elements for energy efficient management of cloud computing environments.
In [SKZ08], authors explored consolidation-based policies by taking into account the inter-relationships between energy consumption and performance metrics for a set applications on real hardware.

However, none of these works take into account the impact of resource contention among jobs (as a general metric), autonomic approach, and policy decisions on energy consumption in order to reach a more general solution than special purpose.

In [UDS00], authors seek to develop two resource contention aware performance metrics for a deeper understanding of performance issues than conventional metrics in distributed algorithms. Their metrics model network contention as well as CPU contention. They illustrate the use of these metrics by comparing four Atomic Broadcast algorithms. However, their model is designed to address resource contention in distributed algorithms.

Papers [DMR09] and [VAN08] propose some quasi-autonomic scheduling approaches in a cluster. However, our approach is more general and autonomic taking into account various layers in resource management. In addition, we incorporate some user-provided information in defining resource contention metric as a novel performance metric, as it is the state of the art, being user-aware in scheduler design [SF09].

In [SVC11], a novel job scheduling approach for homogeneous cluster computing platforms. Its key feature is the use of virtual machine technology to share fractional node resources in a precise and controlled manner. Other VM-based scheduling approaches have focused primarily on technical issues or extensions to existing batch scheduling systems, while in [SVC11] authors take a more aggressive approach and seek to find heuristics that maximize an objective metric correlated with job performance. They derive absolute performance bounds and develop algorithms for the online, non-clairvoyant version of scheduling problem. Their results demonstrate that virtualization technology coupled with lightweight online scheduling strategies can afford dramatic improvements in performance for executing HPC workloads.

While this research seems innovative as one of the latest development in the scheduling, however we consider the following points as weakness of this research in practice: first, time-sharing scheme enabled by VM technology capabilities, i.e. preemption and migration, requires too much of these operations that are heavyweight enough to result in significant overhead; second, these algorithms can lead to good platform utilization if the scheduling cycle is short enough.

CHAPTER 4

Towards a General-purpose and Multi-level Approach in Energy Efficient Computing

There are many approaches, mechanisms and algorithms in the literature on the aforementioned problems and issues in the previous chapter; however, most of them are special purpose. A complete approach should be a multi-level and general-purpose (holistic) approach that is architected in all layers of computing paradigms and systems [SDG11].

For instance, in cloud paradigm from the highest level of resource management stack i.e. cloud pricing strategies and admission control policies to the lower level i.e. policies to direct a scheduler in making various decisions e.g. host selection for a specific job, and finally core scheduling algorithms are some of the research work which could be carried out in a holistic scheduling approach. Such an approach should model the relationship between these layers, for example core scheduling information about jobs and resources might be considered in designing cloud pricing strategies, admission control policies and so on [SG11b].

In brief, this chapter lays some groundwork for future researchers in the field of resource management and scheduling to define, design and develop new objectives and it makes the following contributions in the field:

- Evolution of resource management and scheduling as new technologies, paradigms, etc. emerge

- Establishing relationships within and between various layers of resource management
- Being general rather than special purpose solution for all computing paradigms i.e. Cluster, Grid and Cloud

Figure 4.1 depicts the whole schema of our research including resource management components, contemporary technologies, computing paradigms, green IT and their relationships. In addition, Figure 4.2 summarizes some components of a modern resource management system in a layered architecture, it shows how a job makes its way through resource management system.

In this chapter, in order to address these problems components of resource management system are explored in detail to seek new developments and evolutions and in parallel this process exploits contemporary emerging technologies, computing paradigms, energy efficient operations, etc. to define, design and develop new metrics, techniques, mechanisms, models, policies, and algorithms. Furthermore, finding and establishing relationships within and between various components is a key consideration in this development process.

For example, a metric could be modeled as an approximate function of some other (well defined) metrics. A model could take advantages of several techniques, etc.

A consolidation policy might exploit resource contention and utilization metrics in order to address Resource Contention and Utilization issues i.e. to achieve distribution and packing at the same time, respectively. Thus, in this case consolidation policy is modeled as a function of utilization and resource contention metrics.

Advancements in virtualization have led to the construction of large data centers that host thousands of servers and to the selling of virtual machines (VM) to consumers on a per-hour rate. This current pricing scheme employed by cloud computing

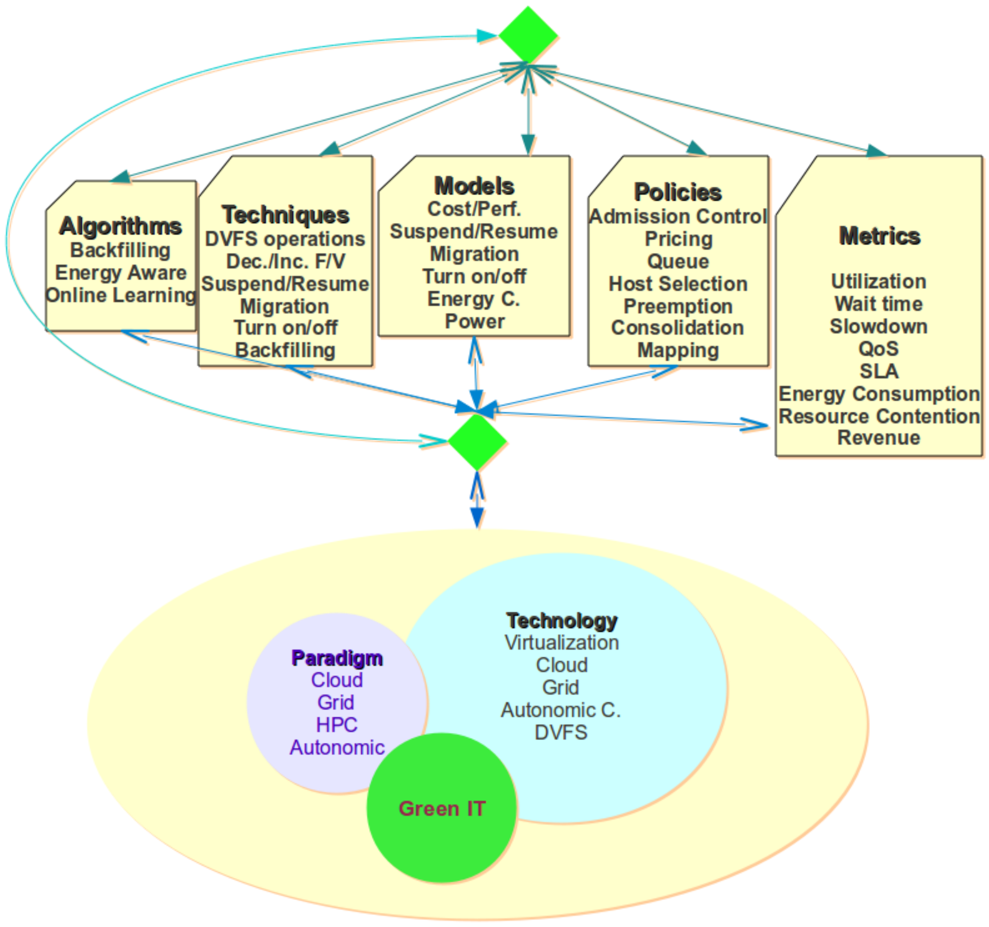


Figure 4.1: Resource management evolution

providers ignores the disparities in consumer usage and in its related infrastructural costs of providing the service to different users. In [Ter12], authors propose a new pricing model based on the liable power consumption of the VM, which they correlate to the VM's proportion of CPU and disk I/O usage. This demonstrates how this research establishes a relationship among pricing scheme and power consumption of a VM based on VM's proportion of CPU and disk I/O usage.

We can enumerate many other relationships in resource management. However,

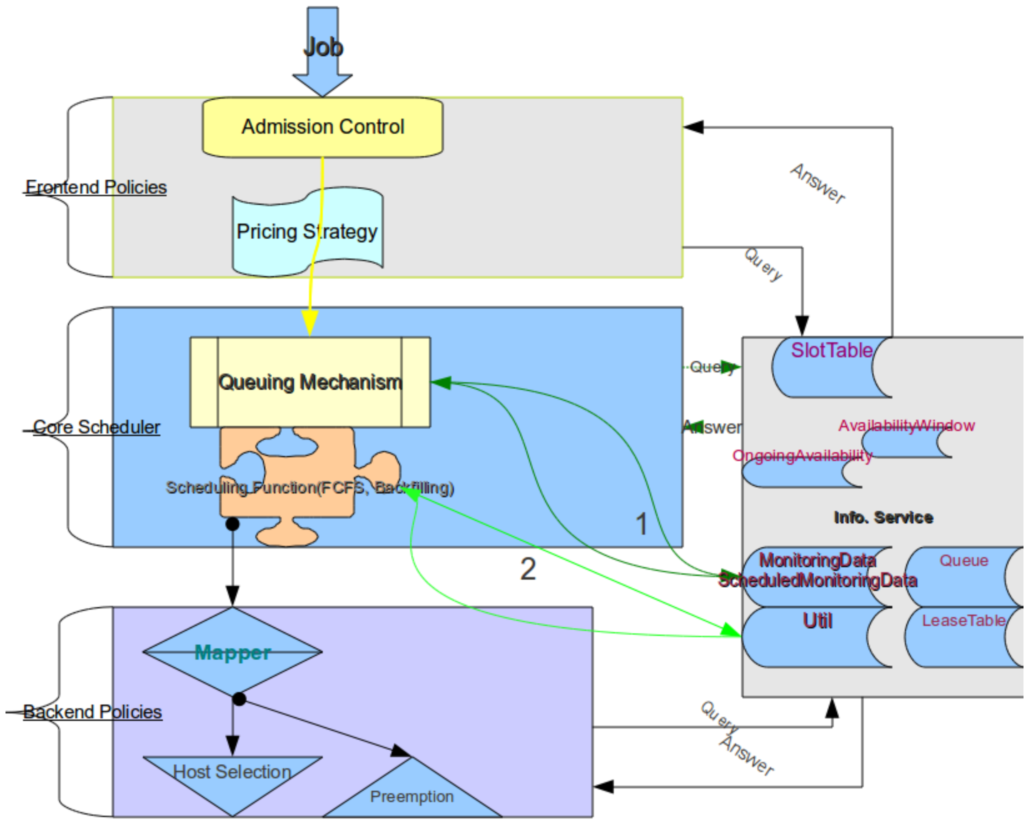


Figure 4.2: Modern resource management system architecture

in some relationships there are tradeoffs, we should model these tradeoffs as well. For example, if we improve utilization, we might face some performance issues such as Resource Contention and Overload. On the other hand, if we improve resource contention, utilization might degrade.

4.1 Metrics

The first step in evolving resource management system is to define, model and develop new metrics to address some problems in better ways than already developed well known metrics. For example, in green computing Energy Consumption is a good

metric to address energy related issues.
Utilization, Wait time, Slowdown, QoS, SLA are some traditional performance oriented metrics in HPC, Grid and Cloud paradigms. Resource Contention is another quasi-performance oriented metric [SLG11] and Revenue is an economic metric introduced by Cloud paradigm. In addition, Energy Consumption metric appeared after the appearance of Green IT term.

There are many metrics to be considered in resource management. We seek to establish some relationships between some of these metrics in order to have a better model, understanding of system behaviour, manifold objectives, etc.
For instance, in [SKZ08] it is demonstrated that Utilization, Poor Performance, and Resource Contention as the main performance metrics and High Energy Consumption as the main energy efficient metric are directly inter-related to each other.
Resource Contention is widely recognized as having a major impact on the performance of computing systems, distributed algorithms, etc. Nonetheless, the performance metrics that are commonly used to evaluate scheduling algorithms (in resource management systems) do not take into account resource contention since researchers are interested in improving the conventional well known performance metrics such as utilization, slowdown and wait time.
On the other hand, in simple terms addressing Resource Contention issue will implicitly address Poor Performance, and High Energy Consumption issues as well as slowdown and wait time metrics; therefore in some environments Energy Consumption optimization and performance issues could be modeled as an approximate function of Resource Contention resolution.
In fact, the following approximate function represents some portion of energy consumption in terms of Resource Contention and Poor Performance:

$$EnergyConsumption \simeq f(ResCont, PoorPerf) \tag{4.1}$$

and the following approximate equation represents poor performance:

$$PoorPerf \simeq g(ResCont) \tag{4.2}$$

so that, we simply model energy consumption as a relative approximate function of resource contention:

$$EnergyConsumption \simeq f(ResCont) \tag{4.3}$$

Therefore, we optimize resource contention metric to achieve energy optimization.

In chapter 5 we define and model resource contention metric and consider it as a first metric for Energy Consumption optimization and performance resolution.

4.2 Techniques

Emerging technologies, paradigms, and energy aware actions provide various techniques to be exploited in resource management. Virtualization technology provides some heavyweight operations such as suspend/resume/migrate and start/stop on virtual machines, these operations are used in many recent research works to improve various metrics such as utilization.
For example, in [SKF08], authors demonstrated when using workloads that combine best-effort and advance reservation requests, a VM-based approach with suspend/resume can overcome the utilization problems typically associated with the use of advance reservations, even in the presence of the runtime overhead resulting from using VMs.

A DVFS-enabled processor provides some energy aware operations i.e. decrease/increase frequency/voltage, transitioning to an idle-state and transitioning to a performance-state.

Time scaling is a technique as a result of DVFS operations which might be exploited in resource management to improve energy consumption and utilization metrics. For instance, by combining energy aware operations we can have some optimization in resource and energy usage that is to fill out free spaces in availability window of a scheduler by extension of or reduction of jobs' running times with the help of increasing/decreasing frequency of processors.

On the other hand, many devices provide the capability to transition to one or more lower-power modes when idle. If the transition latencies into and out of lower power modes are negligible, energy consumption can be lowered simply by exploiting these states. Transition latencies are rarely negligible and thus the use of low-power modes impedes performance. To minimize this impact, it is necessary to alter the workload so that many small idle periods can be aggregated into fewer large ones. This is a workload batching technique to be exploited in such cases.

In addition, techniques for dynamically balancing MPC (Memory accesses per cycle), IPC (Instructions per cycle), utilization and also dynamically scaling the frequency of processors with the help of online learning algorithms [DR09] or other mechanisms are among the other techniques within this domain.

In depth study and research on scheduling strategies [SF05] in particular backfilling mechanisms, revealed that inaccurate estimates generally lead to better performance (for pure scheduling metrics) than accurate ones. This observation proposes the development of new scheduling techniques in HPC and Cloud paradigm according to the differences between these paradigms.

4.3 Models

A model quantifies some parameters in terms of some other parameters such as performance, energy, power and cost. For instance, in green computing a formal cost model quantifies the cost of a system state in terms of power and performance. Sleep states' power rate and their latency i.e. the time required to transition to and from the performance/power state, are examples of parameters in modeling cost vs. performance. In addition, models should specify how much energy will be saved in state transitions and how long it takes for state transitions.

Cost/Performance, Performance/Energy, Cost/Energy, Cost/Power, Suspension/Resumption, Migration, Turn on/off, Energy Consumption, and Power models are examples of some emerging models. Some models emerge as a result of some techniques such as Suspension/Resumption.

The models we seek to design and parameterize in green computing should relate to power consumption and computation rate (performance) or energy consumption and completion time simultaneously. These models are exploited by the scheduling algorithms to select the best state of a processor, memory, disk and network.

Power management actions may affect performance in complex ways because the overall computation rate is a net result of the speed and coordination of multiple elements within the system. For example, doubling the CPU speed may do little to increase the computation rate if the memory transactions do not move any faster. This also indicates that models for the study of energy-computation tradeoffs would need to address more than just the CPU.

Models are also architecture and infrastructure dependent e.g. internal of Multicore and NUMA systems have different features and characteristics to be considered.

In addition, exploiting technology requires models. For instance, we can use the suspend/resume/migrate capability of virtual machines to support advance reservation of resources efficiently [SKF08], by using suspension/resumption as a preemption mechanism, In [SML09] authors presented a model for predicting various runtime overheads involved in using virtual machines in order to support advance reservation requests. It adequately models the time and resources consumed by these operations to ensure that preemptions are completed before the start of a reservation.

4.4 Policies

We categorize policies into frontend and backend policies. Admission control and pricing are frontend policies whereas consolidation, host selection, mapping and preemption belong to backend policies.

Job requests pass through frontend policies before queueing. At the highest level in the cloud interface, we have pricing strategies such as Spot Pricing in Amazon [Ama10] and recent pricing approaches in Haizea [Sot10] or perhaps game theory mechanisms. These mechanisms apply cloud policies that are revenue maximization or improving utilization. Almost these policies have the same goals, and they are energy efficient since they keep cloud resources busy by offering various prices to attract more cloud consumers. A dynamic pricing strategy like offering cheaper prices for applications that will lead to less energy consumption (or higher performance) based on the current cloud status (workloads and resources) compared to the others is an energy efficient pricing schema. Pricing strategies implement cloud administrators' objective i.e. revenue maximization, utilization improvement, etc.

Backend policies could be categorized into three types: general-purpose policies [DMR09], architecture-specific (or infrastructure-specific) policies [HKQ99] application-specific (or workload-specific) policies [KBK07].

General-purpose policies are those that can be applied to most of the computing systems. For instance, CPU/cache-intensive workloads should run at high frequencies, since by increasing frequency the performance scales linearly for a CPU/cache-intensive workload. However, if a task is memory-intensive, the performance improvement is relatively insensitive to increase in frequency, so that a memory-bound workload favors by running at a lower frequency to reduce energy consumption.

Architecture-specific policies are defined based on the architecture or the infrastructure in which computation happens. Also application-specific policies are defined around applications' characteristic.

Technically, workload consolidation [Her09] policy is a sort of policy at the intersection of the last two mentioned policies. Bundling various types of workloads on top of a physical machine is called consolidation.

Furthermore, consolidation-based policies should be designed in such a way to be an effective consolidation. In fact, effective consolidation is not packing the maximum workload in the smallest number of servers, keeping each resource (CPU, disk, network and memory) on every server at 100% utilization, such an approach may increase the energy used per service unit.

That being said, placement of jobs is a critical decision to address Resource Contention. Consolidation policies are one of the sources of information for effective placement of jobs in computing paradigms. In chapter 6, we develop Effective Energy Aware Consolidation based policies. We design consolidation policies according to the resource contention model (metric) and implement them as host selection policies of a distributed system scheduler.

4.5 Algorithms

Algorithms implement techniques, models and policies. For instance, algorithms based on cost/performance models are part of the scheduling to model cost vs. performance of system states. In addition, core scheduling algorithms deal with implementing various backfilling mechanisms, etc. to improve utilization and other optimizations at core of a scheduler. In chapter 7, we propose a novel autonomic energy efficient resource management and scheduling algorithm (architected on different levels of resource management stack). The proposed autonomic scheduling approach answers some *When*-questions to improve energy consumption as it is modeled by resource contention metric. For that, it models interaction between queue mechanism and core scheduler information (about jobs and resources), as a result according to the system state, jobs are reordered and those jobs which satisfy necessary energy aware conditions get admitted to the wait queue and the others are delayed for the next scheduling cycles. From an autonomic scheduling point of view, there are some loops between queue mechanism, scheduling function, end of a job event and core scheduler information. This is an example of a multi-level algorithm over various resource management components.

4.6 HPC, Grid, and Cloud paradigms

In cloud computing, pay-as-you-go on a utility computing basis is the economic model so users pay based on how much time they used cloud resources, while in HPC paradigm there is no general or specific economic model. Similarly, requested allocation time and capacities of resources as allocation parameters are not accurate in HPC whereas they are precise and accurate in cloud computing model. In fact, this is the result of economic model.

In HPC environment, a scheduler makes decisions and reservations according to the requested runtime of jobs which is an estimate of the real runtime; this means jobs might finish earlier or later than the specified requested runtime, however in cloud the requested runtime is the actual runtime.

Therefore, in cloud model users provide which resources they want, the capacity of those resources and precisely for how much time. In this thesis, we take into account these differences and study the impact of these two paradigms on resource contention metric.

However, in-depth study and research on scheduling strategies [SF05] in particular backfilling mechanisms, revealed that inaccurate estimates generally lead to better performance (for pure scheduling metrics) than accurate ones. Here in this work we investigate this observation from resource contention metric point of view by studying HPC versus cloud models. Thus, in experimentations of the next chapters, we have a comparison between cloud and HPC paradigm through simulation experiments.

CHAPTER 5

Resource Contention Metric

In this chapter, we define and develop a Resource Contention metric for HPC workloads in a cluster like environment. We exploit resource contention model of this chapter in the next chapters to develop consolidation policies and so on.

5.1 Resource Contention Metric for HPC Workloads

When there are more than one job with common shared resource(s) being running on a physical host, resource contention happens; this is the basic definition of resource contention.

In order to model resource contention, first we require to model workload characteristics and then we model resource contention metric. In this thesis, we consider HPC workload as a case study.

The precise resource contention metric modeling have to take into account the underlying architecture and infrastructure, since resource contention is dependent on the architecture and the infrastructure in which computation happens. Therefore, according to the environment, there are many parameters in resource contention metric modeling.

5.1.1 Workload Classification and Characteristics

We classify HPC workloads according to their characteristics. Table 5.1 presents a classification of HPC application characteristics in general. For each type of HPC application, this table specifies which resources they consume mostly i.e. they put stress on those resources; in resource contention models only these resource types will be considered since only these resource types make resource contention happens.

Table 5.1: HPC Application Characteristics

HPC Char./Resource	CPU	Cache	Memory	IO	Net-in	Net-out
Compute-intensive	✓	✓	✓	✓	✓	✓
Data-intensive				✓	✓	✓
Memory-intensive		✓	✓		✓	✓
Comm-intensive				✓	✓	✓

5.1.2 Resource Contention Model

Some parameters which might contribute to resource contention modeling are as the following:

- Resource types which make resource contention happens in case of having more than one job running on a physical host. We assume that these resource types are specified by stress-on attribute of a job. In Section 5.1.1, HPC workload as our case study in this thesis are classified according to stress-on attribute.
- The number of consolidated jobs, more jobs translates to more resource contention value. Perhaps in some environments, this is not true.

- The capacities of jobs' resource requirements, more capacities translates to more resource contention value. Again maybe in some environments, this parameter does not contribute.

- How is the contribution of each resource type, e.g. single-instance resource type such as Memory versus multi-instance one such as Multi-core CPU, factors like their level of sequential or parallelism, etc. For instance, single-instance resource types have a sequential access pattern whereas multi-instance resource types have a parallel access pattern, so that the resource contention model should take into account these patterns.

In this thesis, we assume users specify stress-on attribute of their jobs. User provided information and user behavior are exploited in design of contemporary parallel systems schedulers [SF09], and in our research using user provided information is a key to prove its generality.

As we explained earlier, resource contention metric is a novel metric to achieve energy efficiency and resolve performance issues. There are many considerations and parameters in modeling this metric; in this thesis, we use resource type capacities in resource contention model. In this approach, more resource type capacities translates to more resource contention. This seems to be a more realistic approach compared to counting only the number of jobs in which resource type capacities do not contribute to resource contention metric.

Nonetheless, in non-simulated environments i.e. real environments, perhaps there are some mechanics of measuring real resource contention i.e. monitoring system, sensors, etc. However, our previous discussion tries to mimic real scenarios. Therefore, here this metric is developed to just give a quantitative value for comparison of different algorithms, policies and experimentations, however it can be changed or extended to adapt to different environments.

Finally, we model resource contention metric at scheduling cycle t as the following:

$$RC(t) = \sum_{\forall n \in PhyNodes} RC(t,n) \tag{5.1}$$

$$RC(t,n) = \sum_{\forall r \in resTypes} RC(t,n,r) \tag{5.2}$$

where $JobsSchedOn(n,t)$ is the scheduled jobs on node n at time t i.e. at scheduling cycle t when this resource contention calculation happens.

$$resContFlg(t,n,r) = \begin{cases} 1, & \sum_{\forall j \in JobsSchedOn(n,t)} (r \in j.stresson == True ? 1 : 0) > 1 \\ 0, & Otherwise \end{cases} \tag{5.3}$$

$$RC(t,n,r) = \begin{cases} \sum_{\forall j \in JobsSchedOn(n,t) \land r \in j.stresson} j.resReq[r], & resContFlg(t,n,r) = 1 \\ 0, & Otherwise \end{cases} \tag{5.4}$$

Where in the last formula, job's resReq attribute represents capacity of a resource type as job resource requirement and job's stress-on attribute is a set of resource types in which job puts stress on.

Algorithms and mechanisms which will be presented in the next chapters, will try to minimize resource contention among jobs. The corresponding optimization problem models this approach:

$$Mininmize\ RC(t) Subject\ to:\ Computational\ environment\ constraints \tag{5.5}$$

CHAPTER 6

Effective Energy Aware Consolidation Policies

In this chapter, at the policy level, we design and develop energy aware consolidation policies to improve energy consumption via resource contention optimization.

Consolidation policies are one of the sources of information for effective placement of jobs in computing paradigms. In this section, we design and develop two effective energy aware consolidation policies in terms of resource contention model (metric) and implement them as host selection policies of a distributed system scheduler. Effective consolidation is not packing the maximum workload in the smallest number of servers, keeping each resource (CPU, Memory, IO, Net-in, Net-out) on every server at 100% utilization. Such an approach may increase the energy used per service unit.

In particular, the following model is a base model for energy aware consolidation policies that is modeled in terms of resource contention. It calculates the consolidation score of physical host n for job j to be scheduled at scheduling cycle t.

- Let $j.stresson$ be resources which job j puts stress on them.
- Let $j.resReq$ be resource requirements of job j in terms of capacity.
- Let $JS(t, n)$ be the scheduled jobs on physical host n at time t i.e. at scheduling cycle t.

$$Score(Job\, j, Time\, t, Node\, n) = \sum_{\forall r \in j.stresson} ScoreRes(r) \tag{6.1}$$

$\forall r \in j.stresson$

$$ScoreRes(r) = \begin{cases} \sum_{\forall js \in JS(t,n) \wedge r \in js.stresson} js.resReq[r], & Flg(r) = 1 \\ 0, & Otherwise \end{cases} \tag{6.2}$$

$$Flg(r) = \begin{cases} 1, & (\sum_{\forall js \in JS(t,n)} (r \in js.stresson == True ? 1 : 0)) \geq 1 \\ 0, & Otherwise \end{cases} \tag{6.3}$$

Furthermore, the corresponding scheme of a base algorithm for the aforementioned model of an energy aware consolidation policy is presented in algorithms 1 and 2.

Algorithm 1 EAConsolidationPolicy(Job j, Time t)

```
$NodesScore \Leftarrow empty\ dictionary$
**for all** $n$ such that $n \in PhyNodes$ **do**
  $Score \Leftarrow NodeScoreAtTimeT(Job\ j, Time\ t, Node\ n)$
  $NodesScore[n] \Leftarrow Score$
**end for**
**return** $Sort\ NodesScore\ according\ to\ values\ and\ return\ keys\ in\ order$
```

This algorithm in cooperation with the core scheduling part of Haizea, i.e. greedy mapper, implements a matchmaking for finding physical hosts that will have the least resource contention with a job being scheduled. In fact, the placement of a job happens according to the consoliation score of physical hosts regarding that job. Next, we develop two energy aware consolidation policies according to this base policy. The first one is one dimentional taking into account only the time of scheduling for calculating resource contention to measure consolidation score, and the other one is multi-dementional in which it will consider the future reservations and changes in the capacity of physical hosts to calculate resource contention.

Algorithm 2 NodeScoreAtTimeT(Job j, Time t, Node n)

```
$Score \Leftarrow 0$
**for all** $r$ such that $r \in j.stresson$ **do**
    $Flag \Leftarrow False$
    **for all** $js$ such that $js \in JS(t, n)$ **do**
        **if** $r \in js.stresson$ **then**
            $Flag \Leftarrow True$
            break
        **end if**
    **end for**
    **if** $Flag = True$ **then**
        **for all** $js$ such that $js \in JS(t, n)$ **do**
            **if** $r \in js.stresson$ **then**
                $Score \Leftarrow Score + js.resReq[r]$
            **end if**
        **end for**
    **end if**
**end for**
**return** $Score$
```

6.1 Simple Consolidation Policy

This is the simplest policy which according to the status of a physical host at time t, calculates resource contention among the scheduled jobs on a physical host, and then measures consolidation score of a physical host regarding a job being scheduled. Tentative time t is determined by the scheduler as a possible time to schedule job.

In this policy only the schedules and reservations on a physical host at time t will participate in consolidation score and it simply ignores the job time horizon, which perhaps in the future there will be changes in reservations and schedules. In brief, this policy is simply the aforementioned base algorithm.

6.2 Consolidation Policy Over Job Time Horizon

This policy is multi-dimensional over time horizon of a job runtime in which it considers the future reservations and changes of a physical host to calculate resource contention. Therefore, this policy seems to be more precise; it divides future time into time steps in which in each time step the status of a physical host remains unchanged. At the beginning of each time step the consolidation score of a physical host regarding a job is determined (calculated according to the base algorithm), and then it is multiplied by the duration of that time step (end of time step minus start of time step). Finally, the summation of all these values over the job runtime is the final consolidation score of the physical host regarding the job being scheduled. The corresponding algorithm for this policy is presented in algorithm 3.

Algorithm 3 EATimeConsolidationPolicy(Job j, Time t)

```
$NodesScore \Leftarrow empty\ dictionary$
$jobDuration \Leftarrow j.duration$
**for all** $n$ such that $n \in PhyNodes$ **do**
    $Score \Leftarrow 0$
    $time \Leftarrow t$
    $changePoints \Leftarrow getChangePointsAfter(Time = time, until = jobDuration, node = n)$
    **if** $len(changePoints) > 0$ **then**
        **while** $chp \Leftarrow changePoints.getNext()$ **do**
            $tsDuration \Leftarrow chp - time$
            $Score \Leftarrow Score + NodeScoreAtTimeT(j, time, n) * tsDuration$
            $time \Leftarrow chp$
        **end while**
        **if** $time < t + jobDuration$ **then**
            $Score \Leftarrow Score + NodeScoreAtTimeT(j, time, n) * (t + jobDuration - time)$
        **end if**
        $Score \Leftarrow Score/jobDuration$
    **else**
        $Score \Leftarrow NodeScoreAtTimeT(j, time, n)$
    **end if**
    $NodesScore[n] \Leftarrow Score$
**end for**
**return** $Sort\ NodesScore\ according\ to\ values\ and\ return\ keys\ in\ order$
```

6.3 Evaluation by experimentations

We evaluate the two aforementioned consolidation policies against the greedy policy in Haizea to measure how effective they are in handling resource contention for resources when making placement decisions as part of scheduling decision. In this section, first we describe experiments' environment and settings such as resource model, workload traces, and configuration settings, then we present experimental results.

We have considered commodity cluster infrastructure as resource model in this study, i.e. each physical node has processor, memory, IO, Network input, and Network output as resource types and conventional interconnection between them.

As resource contention is our main metric for evaluation and if each physical node hosts at most one job there would be no resource contention, therefore we carry out experiments with multi-instance type processors of Haizea to simulate multicore [Int10] (i.e. more than one core per processor) in real hardware scenario, so physical nodes are able to host more than one job. Other resource types are single-instance. However, these assumptions are not a limitation to affect results.

6.3.1 Workload Traces

We use workload archives from Parallel Workloads Archive [Fei10] as job traces in simulation experiments. In general, there is no workload archive in Parallel Workloads Archive to present HPC application characteristics of jobs, which resource they put stress on, and some other information we need in designing policies such as capacities of Net-in, Net-out and IO resource types of jobs if they consume any of these resources. We synthetically generate these parameters by uniform distributions. At first, a unifrom distribution specifies HPC characteristic of a job. Then, according to HPC characteristic we have at most three uniform distribution to select resource types of a group i.e. one for 'CPU Memory', 'CPU', and 'Memory' group of resources, the other one for 'Net-in Net-out', 'Net-in', and 'Net-out', and the last one for 'IO' resource.

6.3.2 Configurations and Experimentations

We conduct a number of experiments according to the following configuration parameters to cover all the approaches (policies, paradigms, number of cores per physical host) mentioned in this paper: **Workload traces**: SDSC Blue Horizon from the Parallel Workloads Archive [Fei10]. This trace is used and well studied in the other works by Haizea [Sot10]. The no ramp-up/ramp-down utilization of this 30 days of job requests regarding the corresponding site is 67.10%; **Computing Paradigms**: cloud and HPC paradigms; **Host Selection Policies**: greedy, SimpleEAConsolidation, EAConsolidationOverJobTimeHorizon. **Sites**: with multi-instance type CPU i.e. $8, 16$ number of cores per physical nodes. **Backfilling**: conservative.

Finally, we perform a number of experiments with Haizea according to configurations explained earlier.

6.3.3 Results

In these experimentations, we explore the impact of consolidation policies and computing paradigms on Completion Time and Resource Contention metrics. Figure 6.1a shows resource contention graph over scheduling time steps (throughout experimentation time) with 16 cores physical nodes. Similarly, Figure 6.1b shows the aforementioned graph for configurations the same as previous experiments except a site with 8 cores physical nodes. The final Completion Time and Resource Contention metric values for 16 and 8 cores are summarized in Tables 6.2a and 6.2b, respectively.

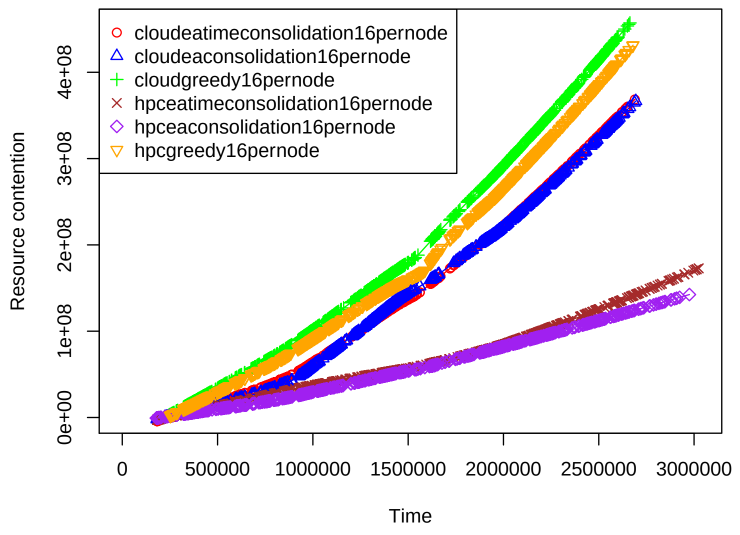
Table 6.1: Completion Time and Resource Contention metric for 16 and 8 cores

Configuration	Time	Resource Contention
cloudeatimeconsolidation	2689207	365667584
cloudeaconsolidation	2694286	372760333
cloudgreedy	2663556	455026120
hpceatimeconsolidation	3058147	201649827
hpceaconsolidation	3019506	170207159
hpcgreedy	2681813	441726388

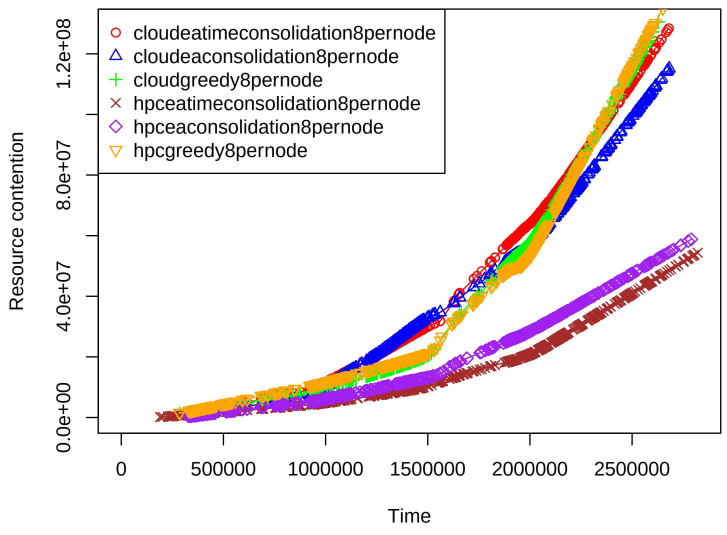
(a) 16 cores

Configuration	Time	Resource Contention
cloudeatimeconsolidation	2681514	128798324
cloudeaconsolidation	2682099	114294790
cloudgreedy	2663556	125467388
hpceatimeconsolidation	2833329	54352992
hpceaconsolidation	2840499	60079428
hpcgreedy	2672287	129578192

(b) 8 cores



(a) 16 cores



(b) 8 cores

Figure 6.1: Resource contention graph for 16 and 8 number of core processors

We write our analysis based on three kinds of comparisons: First, greedy vs. energy aware policies: Almost in all cases (both computing paradigms, and both 8 and 16 cores), energy Aware policies outperform greedy policies. Second, SimpleEAConsolidation vs. EAConsolidationOverJobTimeHorizon: In general, in each computing paradigm and with the same cores, we observe that these two Host Selection Policies approach the same trend and there is no clear outperformance of the one over the other. On the other hand, a more detailed analysis reveals that SimpleEAConsolidation outperforms EAConsolidationOverJobTimeHorizon except two cases in which in one case they are almost the same in graph 6.1a for cloud paradigm (but final value in Table 6.2a indicates that EAConsolidationOverJobTimeHorizon is better) and in the other case SimpleEAConsolidation is worse than EAConsolidationOverJobTimeHorizon. Third, cloud vs. HPC: In spite of the fact that cloud scheduling is more precise than HPC; surprisingly we observe that HPC results are much better than cloud's. In addition, in both paradigms greedy policies have the same trend.

In all, usually in cases which we have better resource contention handling there are degradation in utilization, and consequently completion time of experimentations increases. In these experimentations we used a trace with moderate utlization of 67.10%; perhaps with higher utilization traces which there would be more future reservations (longer wait queue) in backfilling strategies, EAConsolidationOverJobTimeHorizon policy would have better resource contention handling. We consider this as a future work.

CHAPTER 7

Autonomic Scheduling and Energy Efficiency

In this chapter, we propose a novel autonomic energy efficient resource management and scheduling approach architected on different layers of resource management stack [SG11a].

Our approach in this work is similar to Autonomic Computing, and we are inspired by Autonomic Computing developments, due to that we review Autonomic Computing concepts, and in describing our work we mention its corresponding component in Autonomic Computing. In the following sections we describe our autonomic energy efficient scheduling approach which spans in different levels of resource management stack.

7.1 Autonomic Scheduling

Autonomic Computing concept is introduced by IBM in 2001, Paul Horn, senior vice president of IBM Research in an annual meeting of the preeminent technological minds in Arizona suggested a solution: ”build computer systems that regulate themselves much in the same way our autonomic nervous system regulates and protects our bodies.”

It refers to the self-managing characteristics of distributed computing resources in order to optimize its status and automatically adapt itself to changing conditions; it makes decisions on its own (using high-level policies). Therefore, self-* behaviors are

the main characteristics of an autonomic system e.g. self-monitoring, self-adjustment, self-control, self-contained, self-configuration, self-healing, self-optimization and self-protection.
In our model of autonomic scheduling, we take into account the interaction of low level components of resource management stack i.e. core scheduler information with the higher level components of resource management stack i.e. the frontend components such as admission control, pricing strategy and queueing mechanism. Briefly, in our approach according to the system's state from the **core scheduler information** about e.g. resources, jobs, and applications, the scheduler makes decisions in **the queuing mechanism** i.e. whether put a new coming job in the wait queue by setting its state to Pending state immediately, or wait for some more scheduling cycles until the necessary conditions establish by setting job's state to Prepending state.

Figure 4.2 summarizes components of a modern resource management system in a layered architecture, it shows how a job makes its way through resource management system. In particular, it highlights two autonomic loops demonstrated with numbers 1 and 2, the details of these loops are explained in Sections 7.3.1 and 7.3.2.

7.2 Job and Resource Monitoring

The precise status of resources (being full or not), the weight of jobs, how jobs are scheduled, etc. are the key information to be exploited in the higher levels of autonomic scheduling algorithm i.e. inequality formulas in this work.

In this work, we introduce a new job state as Prepending, this is the state of jobs which are accepted to get into the system but they are not put in the queue after their acceptance immediately.
To keep track of the weight of all jobs in different stages i.e. currently running jobs,

scheduled jobs, waiting jobs, and Prepending jobs, we design three dictionary based data structure as the following:

- MonitoringData: This data structure has a key for each resource type, and the corresponding value of a key reports the current consumption of the corresponding resource type. It only reports about (consumption of) stress-on resource types; since stress-on consumption is contributed to resource contention metric. MonitoringData contains information for all available jobs in the system regardless of their state i.e. currently running jobs, scheduled jobs, waiting jobs, and Prepending jobs.

- ScheduledMonitoringData: This data structure is the same as the previous one, except it does not keep information of Prepending jobs.

- MonitoredJobs: This is to represent how much is the total capacity requirements of a job, and after a job gets scheduled this data structure represents how the capacity requirements of a job are satisfied by physical hosts i.e. what portion of job's capacity requirements is satisfied by contributing physical hosts.

7.3 Algorithm

After introducing the mechanics of monitoring service to provide feedback for the higher level components of resource management stack in the previous section, in this section we describe our autonomic scheduling algorithm.

Autonomic algorithm is developed in the following resource management components:

- Queue Mechanism: This is the entry point of a job into the scheduler. First, a job passes through admission control policy, If this policy determines the job can be accepted, as we explain later according to resources' utilization and monitoring information, it is marked as ”Pending” or ”Prepending”

- Scheduling Function: In every scheduling cycle, at first we check Prepending jobs in search of finding jobs which can be put in the Pending state, thus they will be added to the wait queue, so the scheduler takes care of allocating resources to them.
- Job Scheduling: When the precise schedule of a job (resource allocation) is determined by the scheduler, we fill out MonitoredJobs information to specify which nodes are allocated to job, and how much part of a job is scheduled on the allocated nodes.
- Job Ends: Once a job concludes its execution or is cancelled, and no longer requires the acquired resources, the algorithm updates monitoring data.

7.3.1 Queue Mechanism

At Queue Mechanism, the initial fate of a job will be determined as the following:

1. Determines JobResourceRequirements for all resource types
2. Increases MonitoringData with the amount of JobResourceRequirements for only stress-on resources of job
3. Creates a first level key for job identification number (job-id) in MonitoredJobs and then creates a second level set of keys according to stress-on resource types of job and assigns them the corresponding JobResourceRequirements
4. Gets the current system utilization based on stress-on resource types consumption, that is only the consumption of stress-on resource types is considered

5. If for all job's stress-on resource types the inequality formula 7.1 holds then set job's state to Pending and update ScheduledMonitoringData i.e. to increase it with the amount of JobResourceRequirements for only stress-on resources of job

6. Otherwise evaluate the inequality formula 7.2, if it holds for all job's stress-on resource types, then set job's state to Pending and update ScheduledMonitoring-Data, Otherwise set job's state to Prepending

$$Util[resType] + JobResReqs[resType]/Total[resType] <= 1 \tag{7.1}$$

$$MonitoringData[resType] + JobResReqs[resType] <= Total[resType] \tag{7.2}$$

where *Total* indicates the total capacity of each resource type in a site.

7.3.2 Scheduling Cycle

At the begining of every scheduling cycle, the scheduling function looks at all Prepending jobs in search of finding jobs which can be put in Pending state (i.e. in the wait queue) as the following.

1. If the inequality formula 7.3 holds for all stress-on resource types of a job, then set job's state to Pending and update ScheduledMonitoringData.

$$ScheduledMonitoringData[resType]+JobResReqs[resType] <= Total[resType] \tag{7.3}$$

Then, at the end of every scheduling cycle, algorithm examines each physical host as the following:

1. Calculates physical host's availability

2. For each job scheduled on physical host, we calculate what portion of job's resource requirements is satisfied by that host

3. If there is no free capacity for CPU or Memory resource types, this means physical host cannot allocate further resources to (future) jobs, therefore we should not consider the amount of jobs' resource requirements allocated to this physical host to contribute to monitoring data i.e. ScheduledMonitoringData, MonitoringData and MonitoredJobs. In case of a job being involved in monitoring data for that host, we decrease the amount of its allocation on that host from monitoring data variables.

4. Otherwise, if there is free capacity on physical host, this means physical host can allocate further resources to future jobs, therefore we consider the amount of jobs' resource requirements allocated to this physical host to contribute to monitoring data i.e. ScheduledMonitoringData, MonitoringData and MonitoredJobs. In case of a job not being involved in monitoring data for that host, we increase monitoring data variables by the amount of its allocation on that host.

In fact, we have two states for a physical host, being full or not full. We carefully model transitioning between these two states.

7.3.3 Job Scheduling

When for a job the precise schedule is determined by scheduler, we fill out MonitoredJobs to specify where a job is scheduled (i.e. in which hosts) and how much part of a job is scheduled in contributing hosts. For that, we create a second level dictionary key with the physical host identification in MonitoredJobs and then we create a third level dictionary key as a flag to mark this physical host contribution to monitoring data i.e. to model transitioning between being full or not full states of a physical host, finally by creating a third level dictionary keys for each stress-on resource requirements, we assign the amount of resource consumption of a physical host for a job that is scheduled on that physical host.

7.3.4 Job Ends

Once a job finishes, the algorithm updates monitoring data variables by decreasing the amount of the last job's consumption contribution to monitoring data from ScheduledMonitoringData and MonitoringData, and removes associated MonitoredJobs keys for that job.

7.4 Evaluation by Experimentations

We evaluate the aforementioned autonomic energy efficient scheduling to measure how effective it is in handling resource contention among jobs to acquire shared resources when making queueing decision as part of queue mechanism.

7.4.1 Workload Traces

We use workload archives from Parallel Workloads Archive [Fei10] as job traces in simulation experiments. In general, there is no workload archive in Parallel Workloads Archive to present HPC application characteristics of jobs, which resource types they put stress on, and some other information we need in our model and algorithm such as capacities of Net-in, Net-out and IO resource types of jobs if they consume any of these resources.

We synthetically generate these parameters by uniform distributions. At first, a unifrom distribution specifies HPC characteristic of a job, then according to HPC characteristic we have at most three uniform distributions to select resource types of a group i.e. one for 'CPU Memory', 'CPU', and 'Memory' group of resources, the other one for 'Net-in Net-out', 'Net-in', and 'Net-out', and the last one for 'IO' resource. The details are presented in Table 7.1.

Table 7.1: HPC Characteristics distribution in workload trace

HPC Char.:Prob./Resource	CPU Memory	CPU	Memory	IO	Net-in Net-out	Net-in	Net-out
Compute-intensive : $\frac{1}{2}$	$\frac{6}{10}$	$\frac{4}{10}$	0	$\frac{1}{2}$	$\frac{4}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
Data-intensive : $\frac{1}{6}$	$\frac{3}{10}$	0	$\frac{7}{10}$	$\frac{1}{2}$	$\frac{4}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
Memory-intensive : $\frac{1}{6}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{4}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
Comm-intensive : $\frac{1}{6}$	0	0	0	$\frac{1}{2}$	1	0	0

7.4.2 Configurations

We conduct a number of experiments according to the following configuration parameters to cover various settings (autonomic algorithm, policies, number of cores per physical host) mentioned in this chapter:

- Autonomic Algorithm: If autonomic algorithm is enabled, there is an AUTO term in the initial part of configuration name.
- Workload Traces: We used workload traces derived from SDSC Blue Horizon, SDSC DataStar, and KTH IBM SP2 of the Parallel Workloads Archive [Fei10]. We alter these derived traces according to explanations of Section 7.4.1.
- Sites: With multi-instance type CPU i.e. $2, 4, 8, 16$ number of cores per physical nodes.
- Host Selection Policies: We explored three green host selection policies in [SLG11]. In this chapter, we use these two policies i.e. GREEN1 and GREEN2 in configurations to see their behaviour with autonomic scheduling approach.

In addition to the aforementioned variable parameters, we have some constant parameters i.e. aggressive backfilling strategy and cloud paradigm are used in all configurations. Thus, the scheduling function periodically evaluates the wait queue, using an aggressive backfilling algorithm to determine whether any jobs can be scheduled. Also in this chapter we only study cloud paradigm in which the requested runtime of jobs is precise and accurate, unlike HPC paradigm in which it is an estimation.

In sum, a configuration name has three parts; the first part is 'AUTO' if autonomic algorithm is enabled, otherwise it would be blank; the second part presents host selection policy of a configuration i.e. 'GREEN1' or 'GREEN2' used in experimentation; and the third part shows the number of cores per physical node of a site under experimentation starting with the 'CPN' term followed by the number of cores per physical node format. For instance, 'AUTO-GREEN1-CPN8' is a configuration with autonomic algorithm enabled, GREEN1 as host selection policy, and 8 cores per physical node of configuration's site.

Furthermore, simulated cluster of a configuration is modeled after the corresponding workload trace's cluster.

7.4.3 Experimentations

Finally, we perform a number of experiments with Haizea according to the configurations explained earlier on the following derived workload traces:

- BLUE1: derived from SDSC Blue Horizon trace, it is 30 days of job requests starting at 5:02:14:30.
- BLUE2: derived from SDSC Blue Horizon, it is 30 days of job requests starting at 811:23:22:33
- DS: derived from SDSC DataStar, it is 30 days of job requests starting at 49:16:42:21

7.4.4 Results

In experimentations, we explore the impact of autonomic algorithm on the following metrics:

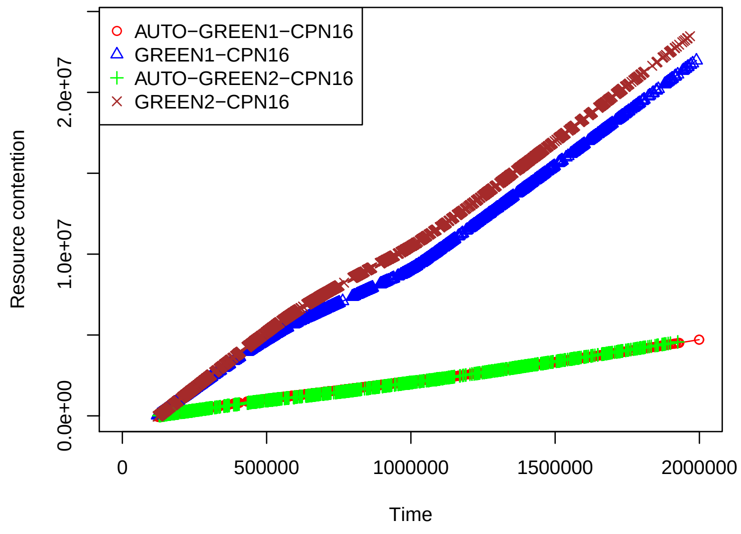
- Completion Time: The time from the start of the trace to when the last job request is completed. The unit of measurement is seconds.
- Resource Contention: Throughout the whole experimentation. The unit of measurement is time based (seconds), but in reality it is just a value for evaluation not counting on the value as a precise metric.

Per each workload trace we group resource contention graph over time based on the experimentation's site i.e. number of cores per physical node. Figures 7.1a, 7.1b, and figures 7.2a, 7.2b demonstrate the aforementioned graph for BLUE1, BLUE2, and DS traces, respectively.

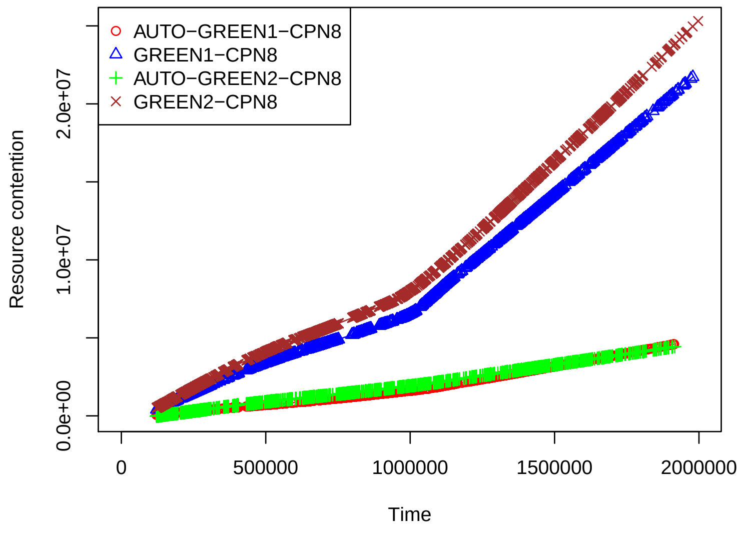
The corresponding final Completion Time and Resource Contention metric values for BLUE1, BLUE2, and DS traces are summarized in Tables 7.3a, 7.3b, Tables 7.4a, 7.4b, and Tables 7.5a,7.5b, respectively.

We observe that for all cases autonomic scheduling approach outperforms non-autonomic one with big improvement.

Tables 7.5 summarizes proportional improvement of autonomic over non-autonomic approach grouped per Trace, Policy, and Site by division operator.

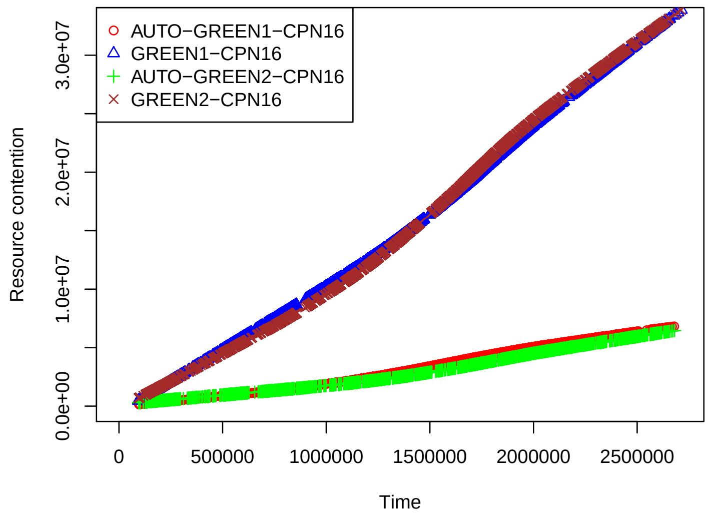


(a) 16 cores

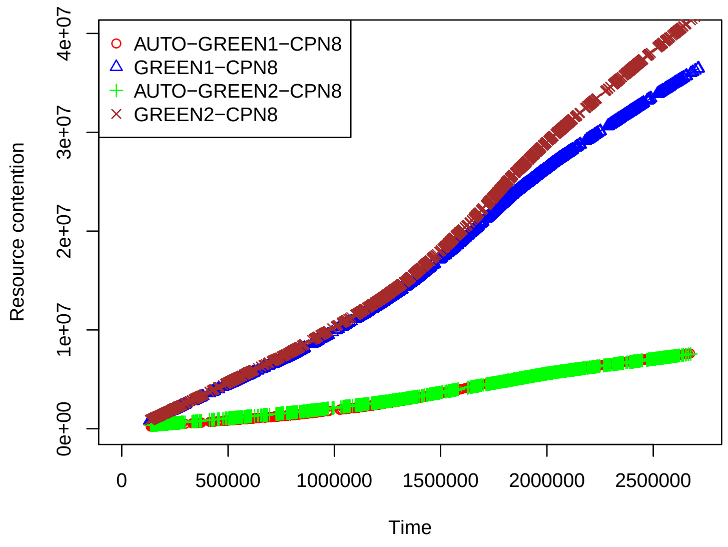


(b) 8 cores

Figure 7.1: BLUE1: Resource contention graph for 16 and 8 number of core processors



(a) 16 cores



(b) 8 cores

Figure 7.2: BLUE2: Resource contention graph for 16 and 8 number of core processors

Table 7.2: BLUE1: Completion Time and Resource Contention metric for 16 and 8 cores

Configuration	Time(sec.)	Resource Contention
GREEN1-CPN16	1989662	22621229
AUTO-GREEN1-CPN16	2006323	4936193
GREEN2-CPN16	1986607	24279215
AUTO-GREEN2-CPN16	2004631	5014698

(a) 16 cores

Configuration	Time(sec.)	Resource Contention
GREEN1-CPN8	1999069	22479516
AUTO-GREEN1-CPN8	1996545	5088787
GREEN2-CPN8	1997208	25174542
AUTO-GREEN2-CPN8	1982155	5546509

(b) 8 cores

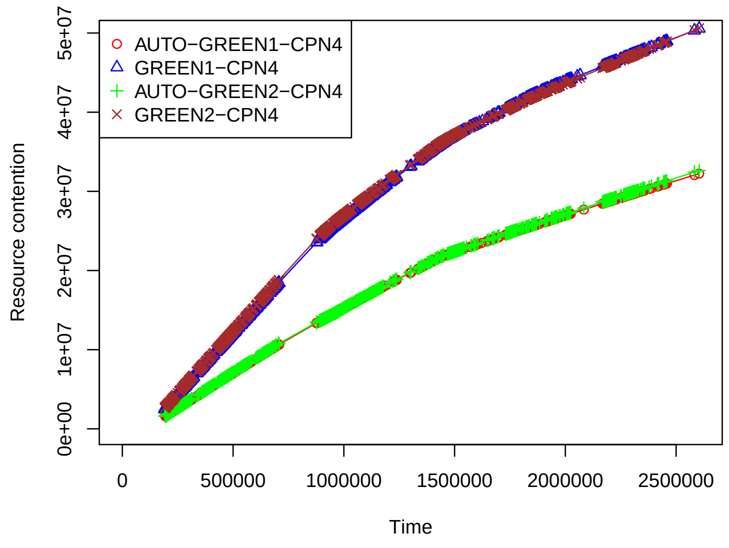
Table 7.3: BLUE2: Completion Time and Resource Contention metric for 16 and 8 cores

Configuration	Time(sec.)	Resource Contention
GREEN1-CPN16	2717401	32340632
AUTO-GREEN1-CPN16	2753694	6754939
GREEN2-CPN16	2716090	32761916
AUTO-GREEN2-CPN16	2753580	6415201

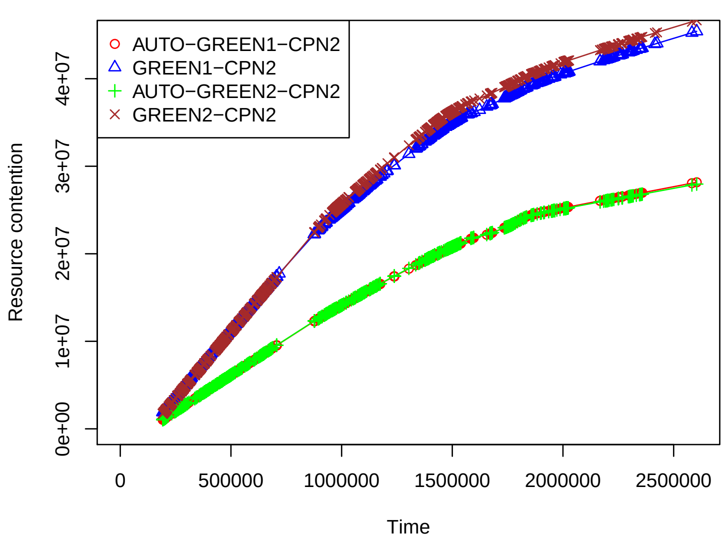
(a) 16 cores

Configuration	Time(sec.)	Resource Contention
GREEN1-CPN8	2717341	35513508
AUTO-GREEN1-CPN8	2759943	7546243
GREEN2-CPN8	2717214	39774280
AUTO-GREEN2-CPN8	2687203	7506258

(b) 8 cores



(a) 4 cores



(b) 2 cores

Figure 7.3: DS: Resource contention graph for 4 and 2 number of core processors

Table 7.4: DS: Completion Time and Resource Contention metric for 4 and 2 cores

Configuration	Time(sec.)	Resource Contention
GREEN1-CPN4	2654637	49242540
AUTO-GREEN1-CPN4	2654637	30790189
GREEN2-CPN4	2654637	49608955
AUTO-GREEN2-CPN4	2654637	31266887

(a) 4 cores

Configuration	Time(sec.)	Resource Contention
GREEN1-CPN2	2654637	43624020
AUTO-GREEN1-CPN2	2654637	27445141
GREEN2-CPN2	2654637	44863890
AUTO-GREEN2-CPN2	2654637	27342088

(b) 2 cores

Table 7.5: Autonomic versus non-autonomic approach improvement

Configuration: Trace-Policy-Site	Autonomic vs. NonAutonomic (/)
BLUE1-GREEN1-CPN16	4.58
BLUE1-GREEN2-CPN16	4.84
BLUE1-GREEN1-CPN8	4.41
BLUE1-GREEN2-CPN8	4.53
BLUE2-GREEN1-CPN16	4.78
BLUE2-GREEN2-CPN16	5.10
BLUE2-GREEN1-CPN8	4.70
BLUE2-GREEN2-CPN8	5.29
DS-GREEN1-CPN4	1.59
DS-GREEN2-CPN4	1.58
DS-GREEN1-CPN2	1.58
DS-GREEN2-CPN2	1.64

CHAPTER 8

Conclusion and Future Work

Green computing is a contemporary research topic in recent years to address climate and energy challenges of the world. Our first step in this thesis was to go over technologies. Thus, we have envisioned the duality of green computing with HPC and cloud computing, business, and economy i.e., green computing solutions drive the development of HPC, cloud computing, economy, and business and vice versa. In order to reach exascale computing, we need huge amounts of energy to operate an exascale system. Thus, green computing is a challenge for the future of HPC. On the other hand, HPC provides solutions for green computing and climate change. In this thesis, we discussed this proposition by looking at technologies in detail.

In addition, we have envisioned that resource management design and architecture should evolve according to the advances in contemporary technologies, computing paradigms, and energy efficient operations to provide new techniques, algorithms, etc. In this thesis, in order to address these problems components of resource management system have been explored in detail to seek new developments and evolutions and in parallel this process exploits contemporary emerging technologies, computing paradigms, energy efficient operations, etc. to define, design and develop new metrics, techniques, mechanisms, models, policies, and algorithms. Furthermore, finding and establishing relationships within and between various components is a key consideration in this development process.

Then, we have proposed effective energy aware consolidation policies. Optimization in energy consumption happens through minimization of resource contention as we have formulated in this thesis. In addition, we have discussed workload characteristics and we developed resource contention metric for CPU, Memory, IO, and network resources as the main parameters in designing policies.

At the last stage, we have proposed autonomic energy efficient scheduling approach. In order to design a multi-level and general-purpose energy efficient distributed system, in this thesis we modeled the relationship between a distributed resource management layers and we reached an autonomic energy efficient approach. This approach models the interaction between **queue mechanism** and **core scheduler information** (about jobs and resources). In this autonomic model there are some loops between **queue mechanism**, **scheduling function**, **end of a job event** and **core scheduler information**.

Finally, we suggest the following works as the future works to be carried out:

- Affinity/grouping algorithms on queue: These algorithms try to group as many jobs (from the queue) as possible and atomically schedule them on a physical host if they don't have any resource conflict. In fact, a grouping algorithm consolidates jobs on a physical host to reduce resource contention
- Exploring the effect of other backfilling strategies such as Lookahead and Aggressive on resource contention
- Lease admission: making relation between aforementioned policies and lease admission, and bringing these policies into this component
- Lease pricing startegies: Like previous one but for pricing
- An HPC workload consists of a preprocessing step, communication/computation steps, and finally a postprocessing step. These steps have different resource requirements, so that according to this, consolidation policies could be improved

References

[AFG09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. "Above the Clouds: A Berkeley View of Cloud Computing." Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[AFG10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. "A view of cloud computing." *Commun. ACM*, **53**(4):50–58, April 2010.

[Ama10] Amazon. "Amazon EC2 Spot Instances." `http://aws.amazon.com/ec2/spot-instances/`, 2010.

[BAR12] FELICITY BARRINGER. "Corporations Slow to Act on Climate Change." http://green.blogs.nytimes.com/2012/09/12/corporations-slow-to-act-on-climate-change-report-says/more-147625, 2012.

[BBA10] Rajkumar Buyya, Anton Beloglazov, and Jemal Abawajy. "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges." In *2010 International Conference on Parallel and Distributed Processing Techniques and Applications*. PDPTA 2010, 2010.

[BE05] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005.

[Bea85] J. E. Beasley. "An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure." *Operations Research*, **33**(1):49–64, January 1985.

[Bob07] N. Bobroff et al. "Dynamic placement of virtual machines for managing SLA violations." In *International Symposium on Integrated Network Management '07*, 2007.

[CGI03] Massimiliano Caramia, Stefano Giordan, and Antonio Iovanella. "An online algorithm for the rectangle packing problem with rejection." In *Proceedings of the 2nd international conference on Experimental and efficient algorithms*, WEA'03, pp. 59–69, Berlin, Heidelberg, 2003. Springer-Verlag.

[CGI04] Massimiliano Caramia, Stefano Giordani, and Antonio Iovanella. "Grid scheduling by on-line rectangle packing." *Networks*, **44**(2):106–119, 2004.

[DMR09] G. Dhiman, G. Marchetti, and T.S. Rosing. "vGreen: A System for Energy Efficient Computing in Virtualized Environments." In *the 14th IEEE/ACM International Symposium on Low Power Electronics and Design*. ISLPED '09, 2009.

[DR09] G. Dhiman and T. Rosing. "System-level power management using online learning." *IEEE Transactions on CAD'09*, 2009.

[Eur12] Eurotech. "Hot water cooled supercomputer." http://www.eurotech.com/en/hpc/hpc+solutions/liquid+cooling, 2012.

[Exx12] Exxonmobil. "The Outlook for Energy." http://www.exxonmobil.com/Corporate/energy_outlook.aspx, 2012.

[Fei10] Dror Feitelson. "Parallel workloads archive." http://www.cs.huji.ac.il/labs/parallel/workload/, 2010.

[FPG10] "1st International Workshop on High Performance Computing, Networking and Analytics for the Power Grid." http://gridoptics.pnnl.gov/sc11/, 2010.

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[Gre12] "The Green500." http://www.green500.org/, 2012.

[Her09] Fabien Hermenier et al. "Entropy: a consolidation manager for clusters." In *In VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (2009)*, pp. 41–50, 2009.

[HKQ99] Inki Hong, Darko Kirovski, Gang Qu, Miodrag Potkonjak, and Mani B. Srivastava. "Power Optimization of Variable-Voltage Core-Based Systems." *IEEE Trans. Computer-Aided Design*, **18**(12):1702–1714, 1999.

[HS91] Robert W. Haessler and Paul E. Sweeney. "Cutting stock problems and solution procedures." *European Journal of Operational Research*, **54**(2):141–150, September 1991.

[HT01] E Hopper and B.C.H Turton. "An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem." *European Journal of Operational Research*, **128**(1):34 – 57, 2001.

[IL99] Zoran Ivkovic and Errol L. Lloyd. "Fully Dynamic Algorithms for Bin Packing: Being (Mostly) Myopic Helps." *SIAM J. Comput.*, **28**:574–611, February 1999.

[Int10] Intel. `http://www.intel.com/support/processors/xeon/sb/cs012641.htm`, 2010.

[KBK07] K. H. Kim, R. Buyya, and J. Kim. "Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters." In *CCGRID*, pp. 541–548, 2007.

[KM77] L. T. Kou and G. Markowsky. "Multidimensional bin packing algorithms." *IBM J. Res. Dev.*, **21**:443–448, September 1977.

[LBL10] LBL. "Green Flash." http://www.lbl.gov/cs/html/greenflash.html, 2010.

[LKK99] William Leinberger, George Karypis, and Vipin Kumar. "Multi-Capacity Bin Packing Algorithms with Applications to Job Scheduling under Multiple Constraints." In *Proceedings of the 1999 International Conference on Parallel Processing*, ICPP '99, pp. 404–, Washington, DC, USA, 1999. IEEE Computer Society.

[LWY09] Gregor von Laszewskiy, Lizhe Wangz, Andrew J. Youngez, and Xi Hez. "Power-Aware Scheduling of Virtual Machines in DVFS-enabled Clusters." In *Cluster 2009*. Cluster 2009, 2009.

[MCT77] K. Maruyama, S. K. Chang, and D. T. Tang. "A General Packing Algorithm for Multidimensional Resource Requirements." *International Journal of Computer and Information Sciences*, **6**(2):131–149, 1977.

[MGV07] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker. "Usher: an extensible framework for managing clusters of virtual machines." In *LISA'07*, 2007.

[MMO09] Marghoob Mohiyuddin, Mark Murphy, Leonid Oliker, John Shalf, John Wawrzynek, and Samuel Williams. "A design methodology for domain-optimized power-efficient supercomputing." In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pp. 12:1–12:12, 2009.

[Nim10] Nimbus. “Nimbus Toolkit Project.” `http://nimbusproject.org/`, 2010.

[Nur08] D. Nurmi et al. “The Eucalyptus open-source cloud-computing system.” In *Cloud Computing and Its Applications'08*, 2008.

[Ope10] OpenNebula. “OpenNebula Cloud Toolkit.” `http://OpenNebula.org/`, 2010.

[Ren12] “World's #1 Renewable Energy Network.” http://www.renewableenergyworld.com/, June, 18 2012.

[SDG11] Mehdi Sheikhalishahi, Manoj Devare, Lucio Grandinetti, and Demetrio Laganà. “A General-purpose and Multi-level Scheduling Approach in Energy Efficient Computing.” In *CLOSER 2011 - International Conference on Cloud Computing and Services Science*, pp. 37–42. SciTePress, 7-9 May, 2011.

[SF05] Edi Shmueli and Dror G. Feitelson. “Backfilling with lookahead to optimize the packing of parallel jobs.” *J. Parallel Distrib. Comput.*, **65**:1090–1107, September 2005.

[SF09] Edi Shmueli and Dror G. Feitelson. “On Simulation and Design of Parallel-Systems Schedulers: Are We Doing the Right Thing?” *IEEE Trans. Parallel Distrib. Syst.*, **20**:983–996, July 2009.

[SG11a] Mehdi Sheikhalishahi and Lucio Grandinetti. “Autonomic Energy Efficient Scheduling.” preprint (2012), to Future Generation Computer Systems, 2011.

[SG11b] Mehdi Sheikhalishahi and Lucio Grandinetti. “Revising Resource Management and Scheduling Systems.” In *CLOSER 2012 - International Conference on Cloud Computing and Services Science*, pp. 37–42. SciTePress, April 18-21, 2011.

[SG12] Mehdi Sheikhalishahi and Lucio Grandinetti. “Green Computing: a dual technology for cloud computing and HPC.” http://www.computer.org/portal/web/computingnow, Nov., 13 2012.

[SKF08] Borja Sotomayor, Kate Keahey, and Ian Foster. “Combining batch execution and leasing using virtual machines.” In *Proceedings of the 17th international symposium on High performance distributed computing*, HPDC '08, 2008.

[SKZ08] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. "Energy Aware Consolidation for Cloud Computing." In *USENIX HotPower'08: Workshop on Power Aware Computing and Systems at OSDI*, 2008.

[SLG11] Mehdi Sheikhalishahi, Ignacio M. Llorente, and Lucio Grandinetti. "Energy Aware Consolidation Policies." In *International Conference on Parallel Computing*, pp. 109–116. IOS Press, 30 August-2 September 2011.

[Sma10] Smart2020. "Smart 2020: Enabling the Low Carbon Economy in the Information Age." http://www.smart2020.org/, 2010.

[SML09] Borja Sotomayor, R.Santiago Montero, I.Martin Llorente, and Ian Foster. "Virtual Infrastructure Management in Private and Hybrid Clouds." *IEEE Internet Computing*, **13**(5):14–22, Sep./Oct. 2009.

[Sol12] LiquidCool Solutions. "LiquidCool Solutions." http://www.liquidcoolsolutions.com/, 2012.

[Sot10] Borja Sotomayor. *Provisioning Computational Resources Using Virtual Machines and Leases*. PhD thesis, Department of Computer Science, University of Chicago, July 7 2010.

[SSV10] Mark Stillwell, David Schanzenbach, Frédéric Vivien, and Henri Casanova. "Resource allocation algorithms for virtualized service hosting platforms." *J. Parallel Distrib. Comput.*, **70**:962–974, September 2010.

[SVC11] Mark Stillwell, Frederic Vivien, and Henri Casanova. "Dynamic Fractional Resource Scheduling vs. Batch Scheduling." *IEEE Transactions on Parallel and Distributed Systems*, **99**(PrePrints), 2011.

[SWW95] David B. Shmoys, Joel Wein, and David P. Williamson. "Scheduling Parallel Machines On-line." *SIAM J. Comput.*, **24**:1313–1331, December 1995.

[Ter12] Kayo Teramoto. "Pay as You Go in the Cloud: One Watt at a Time.", 2012. ACM Student Research Competition Poster Session.

[Top12] "The Top500." http://www.top500.org/, June, 18 2012.

[UDS00] Pèter Urbàn, Xavier Dèfago, and Andrè Schiper. "Contention-Aware Metrics for Distributed Algorithms: Comparison of Atomic Broadcast Algorithms." In *in Proc. 9th IEEE Intl Conf. on Computer Communications and Networks (IC3N 2000*, pp. 582–589, 2000.

[VAN08] A. Verma, P. Ahuja, and A. Neogi. "Power-aware dynamic placement of HPC applications." In *ICS 08*, 2008.

[VBL12] J.L. Vazquez-Poletti, G. Barderas, I.M. Llorente, and P. Romero. “A Model for Efficient Onboard Actualization of an Instrumental Cyclogram for the Mars MetNet Mission on a Public Cloud Infrastructure.” In *Proc. PARA2010: State of the Art in Scientific and Parallel Computing, Reykjavik (Iceland), June 2010*, volume 7133 of *Lecture Notes in Computer Science*, pp. 33–42. Springer Verlag, 2012.

[VMW10] VMWare. “VMware Dynamic Resource Scheduler.” `http://www.vmware.com/files/pdf/drs_datasheet.pdf`, 2010.

[WHL02] Yu-Liang Wu, Wenqi Huang, Siu-chung Lau, C. K. Wong, and Gilbert H. Young. “An effective quasi-human based heuristic for solving the rectangle packing problem.” *European Journal of Operational Research*, **141**(2):341–358, September 2002.