CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Literature review has been done in the area of Cloud computing and certain other optimization techniques applicable to the field of study. The research done by various authors are studied and some of them are discussed in the following section of this chapter.

Navjot Kaur et al (2010) analyzed the comparison of workflow Scheduling algorithms in Cloud Computing. Algorithms are compared with each other on the basis of parameters like total execution time, execution time for algorithm, estimated execution time. The simulation result shows the performance comparison of various algorithms and given better results for workflow scheduling methods in Cloud Environments.

Fatma A. Omara et al (2010) analyzed a dynamic task scheduling model which was used to improve the fuzzy decision in task scheduling on a network of processing elements by introducing new input parameters to an existing fuzzy model and, in the same time, improving the load balance on the network in a dynamic environment. In this model, tasks are generated randomly and served based on First-Come First-Serve basis. The modified fuzzy logic model leads to more precise fuzzy decisions even while dealing with larger number of processors and/or larger number of tasks while increasing the number of involved parameters in the fuzzy model, relative to an existing one.
2.2 A COMPILE-TIME SCHEDULING HEURISTIC FOR INTERCONNECTION-CONSTRAINED

Sih et al (1993) presented compile-time scheduling heuristic called Dynamic Level Scheduling (DLS), which accounts for inter processor communication overhead, when mapping precedence-constrained, communicating tasks onto heterogeneous processor architectures, with limited or possibly irregular interconnection structures. This technique uses dynamically changing priorities to match tasks with processors at each step, and schedules over both spatial and temporal dimensions to eliminate shared resource contention.

This method was fast, flexible, widely targetable, and displays promising performance. Also, a new compile-time scheduling strategy called Dynamic-Level Scheduling (DLS) was proposed and it accounts for inter processor communication overheads, when mapping precedence graphs onto multiple processor architectures.

This technique eliminates shared resource contention by performing scheduling and routing simultaneously to enable the scheduling of all communications as well as all computations. In heterogeneous processing environments, it accounts for varying processor speeds and delivers a more careful apportionment of processing resources. The algorithm was split into two sections to permit a rapid retargeting to any desired multiple-processor architecture by loading in the correct topology-dependent.

The streamlined algorithm was fast enough to be used in interactive environments and is valuable as one of a set of scheduling techniques. This approach attains good scheduling performance by effectively trading off load balancing with inter processor communication, and efficiently overlapping
communication with computation. The DLS technique was also suitable for use in more complicated scheduling techniques as a means of evaluating make span, after preliminary mapping has been determined.

The DLS technique is fast, and iterative approaches are designed to reduce the scheduling bottleneck which may prove beneficial. The interaction between scheduling and routing also merits further examination. Since previous communication resource reservations may block a node, from being scheduled on a certain processor, the rerouting of data transfer paths may facilitate a better node-processor mapping.

Sandeep Tayal et al (2011) established an optimized algorithm based on the Fuzzy-Genetic Algorithm optimization which makes a scheduling decision by evaluating the entire group of task, in the job queue. A two level task scheduling mechanism based on load balancing in cloud computing describes this task scheduling mechanism, which not only satisfies user requirements, but also provides high resource utilization. But, it needs more improvements and this whole algorithm was based on the accuracy of the predicted execution time of each task. Second by, the efficiency of the prediction using Kernel Canonical Correlation Analysis (KCCA) method is highly affected by the choice of task vector.

Alexandru Iosup et al (2011) analyzed the performance of cloud computing services for scientific computing workloads. It described cloud computing services for Many-Task Computing and its applications span a broad range of possible configurations, but utilizing large numbers of computing resources, over short periods of time to accomplish many computational tasks, where the primary metrics are in seconds. The simulation results indicate that the current clouds need an order of magnitude in performance improvement, to be useful to the scientific community, and
shows which improvements should be considered first to address this discrepancy between offer and demand.

Rafael Moreno et al (2011) validated the challenges and viability of deploying a computing cluster on top of a multi-cloud infrastructure spanning four different sites for solving loosely coupled MTC applications. The system was analyzed for the performance of different cluster configurations, using the cluster throughput (i.e., completed jobs per second) as performance metric. Different cluster configurations were compared and the viability of the Multi-Cloud solution was proved from a cost perspective point of view.

Shiyao Chen et al (2011) exemplified on time-varying resource utilization. A transformation is used to reduce the offline problem with time-varying processor capacity with constant capacity. For online scheduling of under loaded system, it is shown that the Earliest Deadline First (EDF) scheduling algorithm achieved better competitive ratio. For the overloaded system, an online scheduling algorithm V-Dover is proposed with asymptotically optimal competitive ratio when a certain admissibility condition holds. The result shows that the proposed V-Dover algorithm outperforms the best known algorithm in all cases compared with other standards.

Shuo Liu et al (2010) presented a novel utility accrual scheduling algorithm for real-time cloud computing services. The algorithm that needs not only to reward the early completions but also to penalize the abortions or deadline misses of real-time tasks. Algorithm carefully chooses the high profitable tasks to execute, and also aggressively removes the tasks that potentially lead to large penalty. The performance of novel utility accrual scheduling algorithm was better than the traditional scheduling algorithms.
such as the Earliest Deadline First (EDF), the traditional utility accrual scheduling algorithm and an early scheduling approach based on the similar model.

Jinhua Hu et al (2010) proposed a scheduling strategy on load balancing of Virtual Machine (VM) resources based on genetic algorithm. This strategy computes the influence it will have on the system ahead, after the deployment of the needed VM resources and then chooses the least-effective solution, through which it achieves the best load balancing and reduces or avoids dynamic migration. This strategy solves the problem of load imbalance and high migration cost by traditional algorithms after scheduling. The simulation results shows that this method can better realize load balancing and proper resource utilization.

2.3 GUEST-AWARE PRIORITY-BASED VIRTUAL MACHINE SCHEDULING FOR HIGHLY CONSOLIDATED SERVER

Dongsung Kim et al (2008) introduced priority-based VM Scheduling Algorithm to reduce scheduling latency of a VM that requires timeliness response. In their scheme, Virtual Machine Monitor (VMM) allocates CPU to each VM, based on the guest-level task information, which was provided by each guest kernel. VMM prioritizes VMs by using the collected information about priorities and status of guest-level tasks in each VM. Their algorithm preferentially treats the VMs that run latency-sensitive applications in response to I/O by inspecting I/O pending status. The proposed algorithm was implemented and evaluated on Xen, which is a virtualization software widely used by many VM researchers. The current VM scheduler manages run queue without considering guest-level tasks, which was the problem addressed in the proposed. The proposed algorithm assigns effective priority, based on guest-level tasks and the algorithm modifies the original run queue management that is simply sorted by credit-based priority.
In the proposed scheme, VMM dynamically re-assigns finer grain priorities than the original credit scheduler to guest domains, based on the information about guest-level tasks in the run queue and wait queue. In addition, VMM infers which domain waits for I/O events, by using the status of shared I/O descriptor rings in domain0. The proposed work suggests the way to scale priority systems of three prominent operating systems: Linux, FreeBSD, and Windows XP. The priority translation modules have not yet implemented.

Although Virtualization Technologies have advanced in terms of high degree of consolidation, the absence of support for latency-sensitive workload could be an obstacle to services that need good quality of responsiveness. To address this problem, the authors introduced a guest-aware priority-based scheduling, which runs on Xen-based system, to preferentially schedule high-priority and latency-sensitive tasks. Their mechanism guarantees CPU fairness because it was implemented over the credit scheduler of Xen. In the proposed mechanism the intrusive way for VMs to send information of guest-level information to VMM was achieved.

2.4 RESOURCE ALLOCATION CHALLENGES IN VIRTUAL MACHINE BASED IT ENVIRONMENTS

Cherkasova et al (2007) presented resource allocation for VMs on CPU Scheduling. As a concrete example, of the types of challenges involved, the CPU schedulers in the Xen VMM were analyzed in the context of traditional workload managers. Workload managers and similar tools were known only to main frames and users of large UNIX environments until few years ago. These technologies have their own requirements from the underlying resource management mechanisms, e.g., CPU Scheduling of VMs. Using Xen and its evolution with three different CPU schedulers; the challenges were explained in choosing the appropriate scheduler features and
parameters, to support desirable application performance, as well as to demonstrate the performance impact of these different choices.

As VM Technologies evolve, their I/O model will certainly undergo some changes. However, it is unlikely that resource allocation problems will disappear anytime soon. For instance, for fully virtualized guests, Xen is considering moving the I/O emulation and it is meant for individual VM in the stub domains. In this case, the problem becomes even more challenging because resources must now be cleverly allocated across twice the number of schedulable entities. Similarly, virtualization-aware I/O devices will ease the problem slightly; however the CPU might still remain on the critical path for most I/O operations. It was conceivable, however, that a combination of better I/O hardware and multi-core processors will alleviate the problem to some extent, in the case, where the number of available cores exceeds the required level of inherent parallelism, in the workload.

Thus, all their experiments have focused particularly on virtualization platform. Their motivation for using Xen, among others, was the availability of its source code and the freedom to modify it. However, the authors stress that, as long as the I/O model involves services being provided, by an entity external to the VM, resource allocation problems will remain. The only difference is that, in Xen's case (which posses a split device driver model and a choice of CPU schedulers), the problems are exposed to the end user and in platforms where the device drivers are in the hypervisor and there is no choice in CPU schedulers, the end user was not directly concerned with these issues. However, the developers of the hypervisor will most certainly need to address the problem of allocating resources to the hypervisor, for providing services while making decisions, on the choice of scheduler and scheduler parameters.
2.5 A MICRO-ECONOMIC MODEL FOR RESOURCE ALLOCATION

Subramanian et al (2002) expelled that; Micro-economic model consists of two principles namely auctioning and commodity market. In commodity approach, the tasks are allocated for individuals as well as for bundled resources. The resources are classified into different classes, based on the CPU cycles, disk space, network bandwidth and memory space. The price of a particular resource was fixed only after equilibrium has been maintained among demand and supply. Suppose both demand increase and supply decrease or vice versa, result reflects an increase in price. Once equilibrium was achieved, further price cannot be changed and follow first come first serve basis.

2.6 COMPARISON OF THE THREE CPU SCHEDULERS IN XEN

Cherkasova et al (2008) work was focused on Central Processing Unit (CPU) Scheduling. As a concrete example of the types of challenges involved by the authors analyzed and compared the CPU schedulers in the Xen VMM in the context of traditional workload managers. Workload managers and similar tools were, known to users of mainframe and users of large UNIX environments. Only a few years ago, these technologies have their own requirements from the underlying resource management mechanisms (e.g., CPU Scheduling of VMs). Using Xen and its evolution with three different CPU schedulers, the authors demonstrated the challenges in choosing the appropriate scheduler features and parameters to support desirable application performance, as well as to demonstrate the performance impact of these different choices. It was projected that, at least for the popular
Xen environment. Much work remains to be done in CPU Scheduling before effective resource isolation guarantees can be delivered, in large-scale deployments.

In the comparison of three proportional-share CPU schedulers for virtual machines, the authors present a sensitivity study, showing how both the CPU scheduling algorithms and the scheduler parameters can drastically impact the performance of I/O intensive applications running on virtual machines. To demonstrate performance sensitivity, three relatively simple system benchmarks, that approximate disk and network intensive workloads, and interrupt driven applications like a web server was used. Also, a simple benchmark, called ALERT, for measuring the CPU allocation errors in these schedulers was introduced. In the process, the authors hope to have motivated the larger problem: resource allocation among VMs is poorly understood, and an initial step is taken towards understanding CPU scheduling issues which, are studied in this paper.

Boeres (2004) analyzed the simulation study of a heterogeneous computational grid using different scheduling algorithms. The notion of ETC perturbation was presented, and this high-performing scheduling algorithm was found to be relatively robust against uncertainties in estimated task completion times.

2.7 THE CHARACTERISTICS OF GRID WORKFLOWS

Ostermann et al (2008) promised to enable a reliable and easy-to-use computational infrastructure for e-Science. To materialize this promise, grids need to provide full automation from the experiment design to the final result. Often, this automation relies on the execution of workflows, that is, of jobs comprising many inter-related computing and data transfer tasks. While several grid work flow execution tools already exist, not much was known
about their workload. This lack of knowledge hampers the development of new workflow scheduling algorithms, and slows the tuning of existing ones.

To address this situation, in this work the authors present an analysis of two workflow-based workload traces from the Austrian Grid. The authors introduce a method for analyzing such traces, focused on the intrinsic and on the environment-related characteristics of the workflows. Then, the authors analyze the workflows executed in the Austrian Grid over the last two years. Finally, the authors identify six categories of workflows based on their intrinsic workflow characteristics. The authors show that the six categories exhibit distinctive environment related characteristics, and identify the categories that are difficult to execute for common workflow schedulers.

As a first step towards understanding grid workflows, the authors study two long-term workload traces from the Austrian Grid. The goal of this work was the analysis of the traces but the comparison of their under laying environment was not performed. However, the data alone are insufficient: there was a need for new methods to extract and analyze the workflow characteristics from the workload traces. Furthermore, there was a need to identify the class of workflows, for which the execution environment yields distinctively poor performance.

Realistic data concerning the characteristics of workflow-based grid workloads, is a key to the adoption and the evolution of grids, but was not readily available to scientists. To address this issue, in this work, the authors present the characteristics of two long-term traces from the Austrian Grid, a grid environment in which workflows are common.

The method identifies two broad classes of workflow characteristics, intrinsic and environment-related. Based on the observed values for the former, the authors devise six classes of workflows with
distinct properties. The analysis of environment-related characteristics reveals that, from the six classes, several classes can be considered as classes of “problem-workflows”, which exhibit one or all of high variability of the work size of their tasks, high make span, poor scalability, and higher than normal failure rate. Overall, the authors proposed that, the workflow speedup was highly dependent on the system used for execution, and the current task success rate requires more fault tolerance mechanisms, especially for large workflows. The authors conclude by addressing the whole grid community with a request for making available their (workflow-based) grid workload traces to other researchers.

2.8 MARKET-BASED RESOURCE ALLOCATION FOR GRID COMPUTING

Gomoluch et al (2003) proposed that; the allocation was a major role in all computing schemes. In market based resource allocation for grid computing, the authors mainly discussed four approaches. They are State Based (current snapshot), Pre-emptive (migrate to different systems), Non-pre-emptive (execute in host itself) and Model Based (predicts system state). Three algorithms were discussed such as Round Robin (RR), Proportional Share Protocol (PSP) and Continuous Double Actions (CDA). The parameters considered in the distributed system are message delay, processing delay, task creation, number and speed of the servers. State Based and Non-Pre-emptive based system was to minimize communication overhead of auctions (e.g. divide and conquer). State based and Pre-emptive are more dynamic and flexible.

2.9 MARKET-ORIENTED GRIDS AND UTILITY COMPUTING

Broberg et al (2008) demonstrated that the resource management techniques (resource allocation, admission control and scheduling) have been
found to be inadequate for many shared Grid and distributed systems that face unpredictable and bulky workloads. Developers doesn’t provide incentives for users to request resources judiciously and do not capture the true value (the utility) of user jobs. Consequently, researchers and practitioners have been examining the appropriateness of ‘market-inspired’ resource management techniques in ensuring that, users are treated fairly, without unduly, favouring one set of users over another. Such techniques aim to smooth out access patterns and reduce the chance of transient overload, by providing incentives for the users, to be flexible about their resource requirements and job deadlines.

The authors examined the recent evolution of these systems, looking at the state of the art in price setting negotiation, Grid economy management, utility driven scheduling, and resource allocation and identified the advantages and limitations of these systems. As Networked resources such as Grids, Clusters and Sensors are being aggregated and shared amongst multiple stake-holders with often competing goals, there has been a rising consensus that, traditional scheduling techniques are insufficient.

However, such staging grounds may be impractical in reality and was in opposition to the free market approach of letting the market work it out by allowing users to suffer the consequences of poor choices. Users need to find the true value of reserve resources. The currency system needs to be well defined (either virtual or real) and receive ongoing care, so that it functions correctly to maximize the utility of the system, avoiding the typical problems of starvation (users run out of money), depletion (users hoard currency or leave the system) and inflation (currency is injected into the system without limit). The use of real money was favoured by many researchers to encourage
more honest valuations of resources in the system. An effective means to calculate and express valuation of resources was needed, for efficient operation of these systems.

Realistic workload model focuses on bags-of-tasks, and the authors validate this model using seven long-term workload traces, taken from large-scale distributed computing systems of various size and application. Finally, the authors explore the large design space of bag-of-task scheduling, in large-scale distributed computing systems along five axes: the scheduling policy, the input workload, the information policy, the scheduling algorithm, and the resource management architecture.

2.10 THE PERFORMANCE OF BAGS-OF-TASKS IN LARGE-SCALE DISTRIBUTED SYSTEMS

Iosup et al (2008) employed large-scale distributed systems such as grids for their computational work, instead of tightly coupled high-performance computing systems. However, while these distributed systems were more cost-effective, their heterogeneity in terms of hardware, software, and systems administration, and the lack of accurate resource information lead to inefficient scheduling. In addition, and in contrast to the workloads of tightly coupled high-performance computing systems, a large part of the workloads submitted to these distributed systems consisted of large sets (bags) of sequential tasks. Therefore, realistic performance analyses of scheduling bags-of-tasks in large-scale distributed systems need to be implemented. Towards the end, the authors introduced a realistic workload model for bags-of-tasks (BoTs), and have explored through trace-based simulations, the design space of scheduling bags-of-tasks in the research. Finally, three new scheduling policies that used inaccurate information only, when scheduling were identified, and was compared against known classes of
proposed scheduling policies. In this research, the authors have presented a realistic and systematic investigation of the performance of scheduling BoTs in large-scale distributed computing systems.

Even though many researches works in computer-science has been devoted to parallel processing, sequential jobs are still predominant in the real world of high-performance and high throughput computing, and often, large numbers of sequential jobs (that share the same executable) are submitted in single groups. The reasons for this phenomenon are relatively high network latencies, complexities of parallel programming models, and nature of the scientific computational work (e.g., repeated simulations, parameter sweeps).

In recent work, the system presented evidence on the predominance and on the characteristics of BoTs in two categories of large-scale distributed computing systems: multi-cluster grids and large-scale cycle-scavenging systems (e.g., Condor-based systems). However, no workload model for BoTs was currently available. Without such a model, researchers employ unrealistic or real traces. The former may lead to tuning the system for cases that cannot exist in practice; the latter have limited use for platforms which have different from the collected ones.

Zhongni Zheng et al (2011) deployed an optimized scheduling algorithm to achieve the optimization or sub-optimization for cloud scheduling problems. Scheduling policy achieved by Parallel Genetic Algorithm (PGA) was much faster than traditional Genetic Algorithm. The results proved that this method improved both the speed of resources allocation and the utilization of system resource. The utilization rate of PGA was better than the RR algorithm and Greedy algorithm.
2.11 SUB OPTIMAL SCHEDULING IN A GRID USING GENETIC ALGORITHMS

Vincenzo Di et al (2002) presented Computing Grid infrastructure with improved overall throughput of the system. It was possible that job submission which included different ontology in resource requests, due to the generality of the GRID infrastructure. Such flexible resource request offered the opportunity to optimize several parameters, from network load to job costs in relation to due time, more generally the Quality of Services (QoS). The result of simulation of the Grid’s job allocation was obtained.

The search strategy for this input case did not converged to the optimal case inside the limited number of trials performed, in contrast with work up to 24 jobs. The benefits of the usage of the genetic algorithms to improve the quality of the scheduling are discussed. The simulation has been obtained using a software environment Grid using Genetic Algorithms (GGAS) suitable to study the scheduling of jobs in a distributed group of parallel machines. The modular structure of GGAS permits to expand its functionalities to include other first level schedule policies with respect to the FCFS that is considered.

Taking advantage of the portability of parallel codes it was possible to build an infrastructure to offer computing power to a large high performance computing audience. To be appealing, such infrastructure should be competitive in terms of costs to the final users. Usually, this corresponds to an optimal usage of the resources, to avoid constraint conflict, which can compromise the usage of this complex and fully automated system. Basically two approaches were proposed to solve the problem, the first approach was based on a distributed resource discovers and allocation system, the second approach was based on a central repository of distributed resources and resources requests.
The first approach was made suitable for small jobs that can easily be accepted by the computing Grid. The second approach was appropriate for large periodical jobs that were scheduled in advance and that can benefit from costs reduction. The investigation was made on the optimization problems in a Grid computing environment where heterogeneous resources usually owned by distinct organizations and with different scheduling policies to be allocated. Genetic algorithm was used for the said purpose. For simple hand designed problem configurations the system was able to provide converge to the optimum. For more complex configurations the optimum wasn’t reached even after certain limits.

Baomin Xu et al (2011) incorporated Berger model theory into Job Scheduling Algorithm. The algorithm established dual fairness constraint. The first constraint was to classify user tasks by QoS preferences, and establish the general expectation function in accordance with the classification of tasks to restrain the fairness of the resources in selection process. The second constraint was to define resource fairness justice function to judge and the fairness of the resources allocation. The algorithm was effective during implementation of user tasks, and with better fairness.

2.12 RESEARCH ON COARSE-GRAINED PARALLEL GENETIC ALGORITHM BASED GRID JOB SCHEDULING

Huifu Zhang et al (2008) suggested that optimizing job scheduling was the major issue in achieving high performance in grid computing systems. The grid workload consisted of multiple jobs and the execution precedence constraint was represented by a Directed Acyclic Graph (DAG). Genetic Algorithms prove useful to resolve large scale combinatorial prediction and optimization problems. A coarse-grained parallel genetic algorithm based grid job scheduling model was proposed. It focused to minimize execution time of jobs and make span of resources and also
improves utilization of resources. The analysis shows that the scheduling system using the coarse-grained parallel genetic algorithm allocated jobs efficiently and effectively.

Wuqi Gao et al (2012) encountered the cloud simulation scheduling algorithm based on multi-dimension QoS. Analytic hierarchy process was introduced into the Resource Scheduling Algorithm to compute every dimensional parameters weight. The scheduling algorithms not only met the customer needs for multi-dimension QoS, but also minimized the finishing time and improved the simulation resource utility rate. The Cloud simulation platform testing based on Cloudsim. Cloud Simulation Resource Scheduling Algorithm (S-CSRSA) guaranteed algorithm satisfaction of user task and good performance in loading equilibrium.

2.13 CLOUD COMPUTING

This work is to clarify terms, provide simple formulas to quantify comparisons between cloud and conventional computing, and identify the top technical and non-technical obstacles and opportunities of Cloud Computing. The view was shaped in part by working since 2005 in the UC Berkeley RAD Lab and in part as users of Amazon Web Services since January 2008 in conducting the research and their teaching. The RAD Lab’s research agenda was to invent technology that leverages machine learning to help automate the operation of datacenters for scalable Internet services.

Cloud Computing refers to both the applications delivered as services over the internet and the hardware and systems software, in the data centers that provide those services. The services themselves have long been referred to as Software as a Service (SaaS), and so the same term was adopted. The data center hardware and software is called as a Cloud. When a
Cloud is made available, in a pay-as-you-go manner to the public, called as a Public Cloud; the service being sold is Utility Computing. Current examples of public Utility Computing include Amazon Web Services, Google App Engine, and Microsoft Azure. The term Private Cloud refers to internal data centers of a business or other organization that are not made available to the public. Thus, Cloud Computing was the sum of SaaS and Utility Computing, but does not normally include Private Clouds. Cloud Computing was generally used for replacing it with one of the other terms only when clarity is expected.

2.14 CHARACTERISTICS OF CLOUD COMPUTING

Liu Peng (2009) enumerates that Cloud Computing was a broader concept than utility computing and related to the underlying architecture in which the services are designed. It may be applied equally to utility services and internal corporate data centers. Cloud Computing concepts are equivalent to Grid Computing. Cloud Computing is the idea that applications run somewhere on the “Cloud” (whether an internal corporate networks or the public Internet) which mean users don’t know or care where the application is running. End user’s has been using web applications for data access.

Cloud Computing allows to develop, deploy and run applications that can easily grow capacity (scalability), work fast (performance), never rarely fail (reliability), all without any concern as to the nature and location of the underlying infrastructure. The way the industry has traditionally built software applications won’t cut it on the cloud. That’s why companies such as Google, Amazon and eBay have developed their own infrastructure software, opting not to reply on products from the large middleware vendors such as Oracle and BEA, who designed the clouds with a very different approach in mind.
For this reason, the end users are seeing the emergence of a new generation of application platform vendors. Although, developers are often lumped together, the differences between utility computing and cloud computing are crucial. Utility Computing relates to the business model in which application infrastructure resources hardware and software are delivered. In conjunction to the above, the characteristics of Cloud including the Scalability, flexibility, dynamic, platform supportability etc was also discussed.

2.15 MARKET-ORIENTED CLOUD COMPUTING

Rajkumar Buyya et al (2008) described that computing paradigms promising to deliver the vision of computing utilities; defines Cloud Computing and provides the architecture for creating market-oriented Clouds by leveraging technologies such as VMs; provides thoughts on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain Service Level Agreement (SLA) oriented resource allocation; presents some representative cloud platforms especially those developed in industries, along with the current work, towards realizing market-oriented resource allocation of clouds by leveraging the 3rd generation Aneka enterprise Grid technology; reveals the early thoughts on interconnecting Clouds for dynamically creating an atmospheric computing environment along with pointers to future community research; and concludes with the need for convergence of competing IT paradigms for delivering the 21st century vision. Cloud provides supply to all their computing needs.

Cloud providers will need to consider and meet different QoS parameters of each individual consumer as negotiated in specific SLAs. To achieve this, Cloud providers should no longer continue to deploy traditional
system-centric resource management architecture that do not provide incentives for them to share their resources but still regard all service requests to be of equal importance. Instead, market-oriented resource management is necessary to regulate the supply and demand of cloud resources at market equilibrium, provide feedback in terms of economic incentives for both Cloud consumers and providers, and promote QoS-based resource allocation mechanisms that differentiate service requests based on their utility.

The high-level architecture for supporting was market-oriented resource allocation in Data Centers and Clouds. There are basically four main entities involved:

- **Users/Brokers**: Users or brokers acting on their behalf submit service requests from anywhere in the world to the Data Center and Cloud, to be processed.

- **SLA Resource Allocator**: The SLA Resource Allocator acts as the interface between the Data Center/Cloud service provider and external users/brokers.

- **Service Request Examiner and Admission Control**: When a service request was first submitted, the Service Request Examiner and Admission Control mechanism interprets the submitted request for QoS requirements, before determining whether to accept or reject the request. Thus, it ensures that there is no overloading of resources, whereby many service requests cannot be fulfilled successfully due to limited resources availability. It also needs the latest status information regarding resource availability (from VM Monitor mechanism) and workload processing (from Service Request Monitor mechanism) in order to make resource allocation
decisions effectively. Then, it assigns requests to VMs and determines resource entitlements for allocated VMs.

- **Pricing:** The Pricing mechanism decides how service requests are charged. For instance, requests can be charged based on submission time (peak/off-peak), pricing rates (fixed/changing) or availability of resources (supply/demand). Pricing serves as a basis for managing the supply and demand of computing resources within the data center and facilitates in prioritizing resource allocations effectively.

- **Accounting:** The Accounting mechanism maintains the actual usage of resources by requests, so that the final cost can be computed and charged to the users. In addition, the maintained historical usage of information can be utilized by the Service Request Examiner and Admission Control mechanism to improve resource allocation decisions.

- **VM Monitor:** The VM Monitor mechanism keeps track of the availability of VMs and their resource entitlements.

- **Dispatcher:** The Dispatcher mechanism starts the execution of accepted service requests on allocated VMs.

- **Service Request Monitor:** The Service Request Monitor mechanism keeps track of the execution progress of service requests.

- **VMs:** Multiple VMs can be started and stopped dynamically on a single physical machine to meet accepted service requests, hence providing maximum flexibility to configure various partitions of resources on the same physical machine to different specific requirements of service requests. In addition, multiple VMs can concurrently run applications.
based on different operating system environments on a single physical machine since every VM is completely isolated from one another on the same physical machine.

- **Physical Machines:** The Data Center comprises multiple Computing servers that provide resources to meet service demands.

## 2.16 DEPLOYING DATABASE APPLIANCES IN THE CLOUD

Aboulnaga et al (2009) presented about the challenges and outline of the tools and techniques which requires addressing the Cloud. An end-to-end solution to one of the tuning problem in the environment was partitioning the CPU capacity of a physical machine among multiple database appliances running on at machine. It outlined the possible future research work. It focused on computing clouds where the user sees a barebones machine with just an operating system and gets full flexibility in installing and configuring software on that machine. These clouds are known as Infrastructure as a Service (IaaS) clouds. A very prominent example of this type of Cloud is Amazon’s Elastic Computing Cloud (EC2), which enables users to rent computing power from Amazon to run their software. Other providers of this style of Cloud Computing include GoGrid and AppNexus. Additionally, many organizations are building IaaS clouds for their internal use.

In IaaS clouds, users are typically given access to Virtual Machines (VMs) on which the software can be installed and perform sequential runs. These virtual machines are created and managed by a Virtual Machine Monitor (VMM) which is a layer of software between the operating system and the physical machine. The VMM controls the resources of the physical machine, and can create multiple VMs that share these physical machine resources. The VMs have independent operating systems running
independent applications, and are isolated from each other by the VMM. The VMM controls the allocation of physical machine resources to the different VMs.

The VMM also provides functionality such as saving and restoring the image of a running VM, or migrating VMs between physical machines. As Cloud Computing becomes more popular as a resource provisioning paradigm, the authors will increasingly see database systems being deployed as virtual appliances on Infrastructure as a Service (IaaS) clouds, such as Amazon’s EC2. Here, some of the challenges associated with deploying these appliances and tuning their performance were outlined, and discussed the tools and techniques required to address these challenges. An end-to-end solution to one of the tuning problem was the partitioning of the CPU capacity of a physical machine among the database appliances running on the machine. The application-informed about tuning, can provide significant benefits to both providers and users of Cloud Computing.

2.17 EFFICIENT RESOURCE MANAGEMENT FOR CLOUD COMPUTING ENVIRONMENTS

Andrew J. Younge et al (2010) put forward that the notion of Cloud Computing has not only reshaped the field of distributed systems but also fundamentally changed how businesses utilize computing today. While Cloud Computing provides many advanced features, it still has some shortcomings such as the relatively high operating cost for both public and private clouds. The area of Green Computing was also becoming increasingly important in the world with limited energy resources and an ever-rising demand for more computational power. Using power-aware scheduling techniques, variable resource management, live migration, and a minimal virtual machine design, overall system efficiency will be vastly improved in a data center based cloud with minimal performance overhead.
The concepts inspired by the notion of utility computing have recently combined with the requirements and standards of Web, to create cloud computing. Currently, it was estimated that data centers consume 0.5 percent of the world's total electricity usage and if current demand continues, it is projected to quadruple by 2020. Currently, there are two competing types of Green Scheduling Systems for Supercomputers; power-aware and thermal aware scheduling. As the prevalence of Cloud computing continues to raise, the need for power saving mechanisms within the Cloud also increases. The major focus was to deliver the efficient usage of power.

2.18 VIRTUAL INFRASTRUCTURE MANAGEMENT IN PRIVATE AND HYBRID CLOUDS

In Borja Sotomayor et al (2009) view, definition of "Cloud" is infrastructure-as-a-service (IaaS) system, in which IT infrastructure was deployed in a provider's data center as virtual machines. With IaaS Clouds growing popularity, tools and technologies are emerging that can transform an organization's existing infrastructure into a private or hybrid cloud. Open Nebula is an open source, virtual infrastructure manager that deploys virtualized services on both a local pool of resources and external IaaS Clouds. Haizea, a resource lease manager, can act as a scheduling back end for Open Nebula, providing features not found in other cloud software or virtualization-based data center management software.

2.19 AN APPROACH TO OPTIMIZED RESOURCE SCHEDULING ALGORITHM FOR OPEN-SOURCE CLOUD SYSTEMS

Based on the deep research on Infrastructure as a Service (IaaS) Cloud systems of open-source, an optimized scheduling algorithm proposed by Hai Zhong et al (2010) enables to achieve the optimization or sub-
optimization for cloud scheduling problems. Since, cloud computing was an on-demand computing paradigm, immediate and automated leasing is a favourite scheduling strategy. However, most cloud computing systems in operation today are proprietary and rely upon infrastructure, that is invisible to the research community, or are not explicitly designed to be instrumented and modified by system researchers.

Though, there are many open source cloud systems for researchers emerges as the development of cloud computing. For IaaS, there are some popular open-source cloud systems, such as Eucalyptus, Open Nebula, and Nimbus etc. Since the existing scheduling algorithms don’t have good consideration of utilization, an optimal scheduling policy for the open-source IaaS cloud environments, possibility to allocate the Virtual Machines (VMs) in a flexible way to permit the maximum usage of physical resources was being investigated here. An Improved Genetic Algorithm (IGA) for the automated scheduling policy was used.

The IGA uses the shortest genes and introduces the idea of Dividend Policy in Economics to select an optimal or suboptimal allocation for the VMs requests. The simulation experiments indicate that the dynamic scheduling policy performs much better than that of the Eucalyptus, Open Nebula, Nimbus IaaS Cloud, etc. The tests illustrate that the speed of the IGA, almost twice the traditional GA scheduling method in grid environment and the utilization rate of resources was always higher than the open-source IaaS Cloud systems. The focus here was on the Grid environment, with an increase in the utilization rate, when compared to the other open source systems.
2.20 THE COST OF DOING SCIENCE ON THE CLOUD: THE MONTAGE EXAMPLE

Deelman et al (2009) illustrated utility grids such as the Amazon Elastic Compute (EC2) Cloud and Amazon Elastic Storage (S3) offer computational and storage resources that can be used on-demand, for a fee by computer and data-intensive applications. The cost of running an application, on such a Cloud, depends on the compute, storage and communication resources it will provide and consume. Different execution plans of the same application may result in significantly different costs. Using the Amazon Cloud fee structure and a real-life astronomy application, the authors studied the cost performance tradeoffs of different execution and resource provisioning plans via simulation. Also trade-offs in the context of the storage and communication fees of Amazon S3 when used for long term application data archival were learned. Their results have shown that by provisioning the right amount of storage and compute resources, cost can be significantly reduced with no significant impact on application performance and have used the following terms for applications.

- The entity that provides a service to the community (the Montage project), user request.
- A mosaic requested by the user from the application.
- The Computing/storage resource used by the application to deliver the mosaic requested by the user.

The Montage application and the Amazon EC2 fee structure as a case study, the authors showed that for a data-intensive application with a small computational granularity, the storage costs were insignificant as compared to the CPU costs. Thus it appears that Cloud Computing offers a cost-effective solution for data-intensive applications. Clouds are still in their
infancy—there are only a few commercial and academic providers. As the field matures, expectations to see a more diverse selection of fees and QoS guaranteed for different resources and services provided by Clouds. It is possible that some providers will have a cheaper rate for computing resources while others will have a cheaper rate for storage and provide a range of quality of service guarantees. As a result, applications will have more options to consider and more execution and provisioning plans to develop to address their computational needs.

Wang et al (2011), explored only one aspect of using Cloud Computing for science, examining the tradeoffs of different workflow execution modes and provisioning plans for cloud resources. Many other aspects of the problem still need to be addressed. These include the start up cost of the application on the cloud, which was composed of launching and configuring a virtual machine and its teardown, as well as the often one-time cost of building a virtual image suitable for deployment on the cloud. The complexity of such an image depends on the complexity of the application. They also did not explore other cloud issues such as security and data privacy. The reliability and availability of the storage and computing resources are also an important concern. According to Amazon sources, the targeted availability of the S3 storage system is 99.9% which is also verified by independent studies. However, when the system goes down, as it did twice in the first 7 months of 2008, the possible impact on the applications was significant. Due to the mainly commercial nature of Cloud Computing, there are expectations and penalties resulting from any violation of the user-provider contract, was clearly spelled out. These and other issues such as scalability of the new computing paradigm are still open questions.
2.21 BENCHMARKING AMAZON EC2 FOR HIGH-PERFORMANCE SCIENTIFIC COMPUTING

Walker et al (2008) represented cloud computers for high-performance Scientific Computing compared to currently available alternatives aim to answer a specific instance of this question by examining the performance of Amazon EC2 for high-performance scientific applications. The application used macro and micro benchmarks to study the performance of a cluster composed of EC2 high-CPU compute nodes and compared this against the performance of a cluster composed of equivalent processors available to the open scientific research community. Based on the results obtained the performance gap in the examined clusters that system builders, computational scientists, and commercial cloud computing vendors need to be aware of.

The opportunity of using commercial cloud computing services for High Performance Computing (HPC) is compulsory. It unburdens the large majority of computational scientists from maintaining permanent cluster fixtures, and it encourages free open-market competition, allowing researchers to pick the best services, based on the price payment willingness. However, the delivery of HPC with commercial Cloud Computing services such as Amazon EC2 is not yet mature. This research has shown that a performance gap exists between performing HPC computations on a traditional scientific cluster and on an EC2 provisioned scientific cluster. This performance gap is seen not only in the MPI performance of distributed-memory parallel programs but also in the single compute node performance for shared-memory parallel programs. For cloud computing to be a viable alternative for the computational science community, vendors will need to upgrade their service offerings, especially in the area of high-performance network provisioning, to cater to this unique class of users.
EFFICIENT PARALLEL PROCESSING OF MASSIVE DATA SETS

Chaiken et al (2008) described a new declarative and extensible scripting language; SCOPE (Structured Computations Optimized for Parallel Execution) was targeted for massive data analysis. The language was designed for ease of use, with no explicit parallelism, while being amenable to efficient parallel execution on large clusters. SCOPE borrows several features from Structure Query Language (SQL). Data is modelled as sets of rows composed of typed columns. The select statement is retained with inner joins, outer joins, and aggregation allowed. Users can easily define their own functions and implement their own versions of operators: extractors (parsing and constructing rows from a file), processors (row-wise processing), reducers (group-wise processing), and combiners (combining rows from two inputs).

SCOPE supports nesting of expressions, but also allows a computation to be specified as a series of steps, in a manner often preferred by programmers. In this context it is mandatory to describe how scripts are compiled into efficient, parallel execution plans and executed on large clusters. Here, a new scripting language, SCOPE (Structured Computations Optimized for Parallel Execution), targeted for large-scale data analysis that is under development at Microsoft was also presented. SCOPE intentionally builds on this knowledge but with simplifications suited for the new execution environment.
2.23 CLOUDSIM: A NOVEL FRAMEWORK FOR MODELLING AND SIMULATION OF CLOUD COMPUTING INFRASTRUCTURES AND SERVICES

Rodrigo et al (2009) explained that Cloud Computing focuses on delivery of reliable, secure, fault-tolerant, sustainable, and scalable infrastructures for hosting internet-based application services. These applications have different composition, configuration, and deployment requirements. Quantifying the performance of scheduling and allocation policy on a cloud infrastructure (hardware, software, services) for different application and service models under varying load, energy performance (power consumption, heat dissipation), and system size was an extremely challenging problem to overcome.

To simplify this process, this research work proposes CloudSim: a new generalized and extensible simulation framework that enables seamless modelling, simulation, and experimentation of emerging Cloud Computing infrastructures and management services. The simulation framework has the following novel features: (i) support for modelling and instantiation of large scale Cloud Computing infrastructure, including data centers on a single physical computing node and java virtual machine; (ii) a self-contained platform for modelling data centers, service brokers, scheduling, and allocations policies; (iii) availability of virtualization engine, which aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node; and (iv) flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.
2.24 MODELLING AND SIMULATION OF SCALABLE CLOUD COMPUTING ENVIRONMENTS AND THE CLOUDSIM TOOLKIT

Rajkumar Buyya et al (2009) illustrated that the processor proposes CloudSim: an extensible simulation toolkit that enables modelling and simulation of Cloud Computing environments. The CloudSim toolkit supports modelling and creation of one or more virtual machines (VMs) on a simulated node of a Data Center, jobs and their mapping to suitable VMs. It also allows simulation of multiple Data Centers to enable a study on federation and associated policies for migration of VMs for reliability and automatic scaling of applications. Although the aforementioned toolkits are capable of modelling and simulating the grid application behaviours in a distributed environment consisting of multiple grid organizations, none of these are able to support the infrastructure and application-level requirements arising from Cloud Computing paradigm. In particular, there was very little or no support in existing Grid simulation toolkits for modelling of on-demand virtualization enabled resource and application management.

Further, Clouds promise to deliver services on subscription-basis in a pay-as-per usage model to cloud customers. Hence, cloud infrastructure modelling and simulation toolkits must provide support for economic entities such as cloud brokers and cloud exchange for enabling real-time trading of services between customers and providers. GridSim offers support for economic-driven resource management and application scheduling simulation. There are a number of important issues which requires detailed investigation along the cloud software stack. Topics of interest to cloud developers include, economic strategies for provisioning of virtualized resources to incoming user's requests, scheduling of applications, resources discovery, inter-cloud negotiations and federation of clouds.
The recent efforts to design and develop cloud technologies focus on defining novel methods, policies and mechanisms for efficiently managing Cloud infrastructures. To test these newly developed methods and policies, researchers need tools that allow them to evaluate the hypothesis, prior to real deployment in an environment where one can reproduce tests. Simulation-based approaches in evaluating Cloud Computing systems and application behaviours offer significant benefits. Cloud developers: (i) to test performance of their provisioning and service delivery policies are repeatable and controllable environment, free of cost. (ii) To tune the performance bottlenecks before real-world deployment on commercial clouds.

2.25 SUMMARY

Literatures over similar research work carried over clearly states that, the accurate identification of task scheduling was indispensable in traffic blocking, controlling, measurement and analysis and would not be competitive for the present market scenario. The possibilities of identifying a suitable scheduling method to reduce the unnecessary time delay during the task allocation for cloud based network are worth considering. Literary evidences reveal that various scheduling algorithms carefully specified and tested in the proposed dynamic fuzzy based scheduling optimizer environment was possible to obtain predictions that are very competitive, even when scheduling a complex phenomenon.