Scheduling Workloads in Cloud Computing

Rizwan Mian
Graduate Student
School of Computing
Queen's University
mian@cs.queensu.ca

ABSTRACT

Cloud Computing (cloud) is emerging as a promising paradigm that offers computing as a utility. In this paper, we look at some scheduling policies for workloads in clouds. Suitability of a cloud simulation for developing and evaluating policies is discussed. We simulate a cloud scenario and report results observed from executing the scenario. Then, we share our experiences with a cloud simulation toolkit and suggest fundamental building blocks of a cloud simulation.

Categories and Subject Descriptors

C.4 [**Performance of Systems**] Modeling techniques; C.2.4 [**Distributed Systems**]: Network operating systems

General Terms

Measurement, Performance, Design, Experimentation.

Keywords

Cloud Simulation, Workflow Management.

1. INTRODUCTION

Grid Computing (grid) refers to technologies that allow consumers to obtain computing on demand, analogous in form and utility to the electrical grid [1]. Grids and related application technologies are enabling scientists and engineers to build more and more complex applications for managing and processing large data sets, and for executing scientific experiments on distributed heterogeneous resources [2]. Clouds aims for the same dream of using computing as a utility [3]. The fundamental vision and concepts are the same. The vision of a global grid has not yet been realised but it might be fair to say that cloud builds on the lessons learnt from building a grid.

The key difference between cloud and grid is that cloud resources are managed and used by different parties. Typically cloud vendors happen to be large establishments like Amazon and Google. In contrast, cloud users vary in scale. This allows separation of concerns. Most existing grid middleware, which enables access to grid resources, is widely perceived as being too heavyweight. The heavyweight nature of middleware, especially on the client-side, has severely restricted the uptake of grids by users [4]. On the other hand, cloud vendors – perhaps learning from grid experience – manage complexity on their side and offer interfaces of varying abstraction to their users. From a hardware point of view, cloud offers the illusion of infinite computing resources to users on demand [3].

As a consequence there are many new application opportunities offered by cloud such as analytics, batch processing [3]. Different applications may result in different types of workloads. Workloads can vary from unrelated and independent tasks to related and structured workflows, which consist of a sequence of connected computational or data tasks. The workloads need to be managed on a cloud. The cloud vendor may provide general workload management policies offering a higher level of abstraction to users. However, the policy may be oblivious to a user's internal workload utility properties such as priority or ordering. In such a case, the user must manage their workloads explicitly.

Presently, clouds offer three service abstractions: infrastructure-as-a-service, platform-as-a-service, and software-as-a-service abbreviated as IaaS, PaaS and SaaS respectively.

In IaaS, the cloud vendors expose their infrastructure as a service to the users. For example, Amazon's EC2[1] expose its infrastructure in terms of virtual machines.

In PaaS, software frameworks or platforms are exposed as a service to the users. The users write their software for a specific platform that the vendor hosts. After writing software, the users upload their software to the vendor's system and runs it on the vendor's system whenever needed. Google App Engine[2] offers this level of abstraction.

In SaaS, software is exposed as a service on the web. SaaS is similar in spirit to web services or web applications. Google's online applications – such as Google Calendar – can be thought of SaaS.

Scheduling policies or policies from grid and general distributed computing may be evaluated in a cloud for their relevance and effectiveness for managing workloads in a cloud. The goals of this study are (a) to develop policies in a cloud, and (b) to develop a framework to evaluate such policies.

The original intent of this project [5] was to study the feasibility of the stated goals in a cloud simulator called CloudSim (csim)

[1] Amazon Elastic Compute Cloud (Amazon EC2), (5.4.10), http://aws.amazon.com/ec2/

[2] Google. (2010) Google App Engine http://code.google.com/appengine/ (last checked 3.5.10)

[6]. Further, some vanilla policies were to be evaluated using csim, similar to evaluation of policies in a grid simulator, GridSim [7]. While developing the policies and a platform for evaluation, many shortcomings were noticed in csim. A question arose whether these are the shortcomings of the tool or lack of maturity of a cloud model. This even begged the question whether a simulation is suitable approach to study policies for a cloud.

The remainder of the paper is structured as follows: Section 2 builds a case for a cloud simulation. Section 3 describes the building blocks of csim. Section 4 describes a simulated cloud scenario and report experiment results from executing the scenario. Section 5 shares actual experiences of simulation using csim. Section 6 lists core features expected from a cloud simulation. Section 7 discusses some research ideas. Section 8 concludes this paper.

2. CASE FOR CLOUD SIMULATION

A cloud vendor may offer many data centers due to, for example, different geographical requirements. Cloud vendors expose resources at a massive scale. Users only pay for the resources that they need. Existing or new policies need to be evaluated and developed to use these massive scale resources. These policies also need to accommodate the fact that user demand vary in reality and scale according to user demand.

To evaluate policies, a cloud environment with massive resources needs to be reproduced. We can start by experimenting with resources at a particular data center (DC). The problem there is that vendors do not expose internals of their cloud infrastructure – not to mention the monetary cost it will endure. An alternative would be to create such a large resources base in an academic laboratory. There are immediate cost factors involved i.e. hoping to get resources at such scale in an academic laboratory is unrealistic. Repurposing existing resources might be an option. However, setting up environments at this scale is labour intensive and can be tricky. There are many implementation details. Anyone who has tried to setup a cloud hopefully appreciates this phenomenon. The irony is that setting up cloud may, perhaps, offer research contributions for experience in setting up a cloud, but does not offer contribution towards policies. Even if a private cloud is available, it may be required that cloud infrastructure needs modifications to study particular behaviour or scenarios.

Real world is extremely detailed. It is argued that an effective simulation models the core concepts of a real system. It does not need to model irrelevant details.

With all these limitations, a cloud simulation is a reasonable way forward. Indeed, it is argued that a simulator facilitates researchers to investigate specific aspects of a system, without getting involved in low-level details [6]. Simulation has been used for studying policies. For example, Sakellariou et al. [8] used simulation to study rescheduling policies of workflows on grids.

3. CLOUDSIM

Csim aims at simulating IaaS. A cloud vendor exposes its infrastructure in terms of physical resources.

Buyya et al. [6] claims that there are four stakeholders in Csim: a user, a broker, a DC and a market. However, only the first three are available in the implementation.

A user has some computational and data requirements, modelled as a cloudlet. The broker acts behalf of the user. The user requests the broker to create certain number of virtual machines (VMs). A VM has certain requirements. For example, x86, 2GHz, 1GB memory, 512MB image size etc. The VMs can be of heterogeneous specification. The broker requests a DC to create these VMs. Assuming, DC can satisfy the requirements of VMs, the VMs are created in the DC. Then, the user submits cloudlets to the broker, which in turn submits them to the DC according to a scheduling policy. The default and only scheduling policy available is RoundRobin (RR). RR submits the cloudlets immediately to the VMs present in the DC without queuing or considering the existing load on these VMs.

The VMs execute these cloudlets and report their completion back to the broker, which reports them to the user.

The architecture of Csim is depicted in Figure 1. It shows the layered implementation of the Csim simulation framework and architectural components.

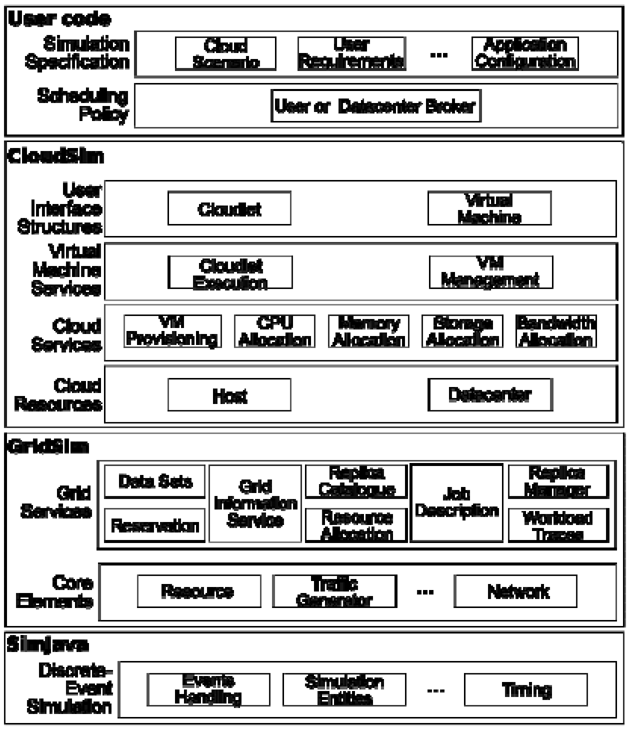

Figure 1 – Layered CloudSim architecture [6]

Buyya et al. [6] builds CloudSim layer on top of GridSim with the intention to reuse exiting simulation libraries and frameworks. GridSim is a grid simulator built on top to SimJava.

SimJava is a discrete event simulation present at the lowest layer. It implements the core functionalities required for higher-level simulation frameworks such as queuing and processing of events, management of the simulation clock, creation of system components (VMs, cloudlet, broker, data center etc.), and communication between components [6].

The layer above SimJava consists of libraries from GridSim toolkit that support fundamental Grid components such as the networks, resources, datasets, workload traces, and information services.

The layer above GridSim is the CloudSim layer. This layer is implemented by programmatically extending the core functionalities exposed by the GridSim layer, and manages the instantiation and execution of core entities (VMs, hosts, DCs, application) during the simulation period. The matchmaking between hosts and VMs takes place at this layer. Cloudlet execution is managed at this layer. This layer would be of particular interest to a cloud vendor since the "effectiveness" of different resource management policies can be studied at this layer.

The top-most layer of the simulation stack is used to define cloud scenario under study. Definitions and configurations of stakeholders take place at this layer. A cloud application consisting of cloudlets and associated VMs is defined at this layer. Datacenter Broker which plays a key part in submitting cloudlets to a DC according to a scheduling policy is defined here.

4. SIMULATING MAPREDUCE SCENARIO ON A CLOUD

It was hoped that using bottom-up approach, existing vanilla policies – such as RR, deadline, budget – would be used to construct the framework. To start with, tasks could have had random duration. To introduce realism, cloud resources would have been made similar to Amazon EC2 instances. Simulated workload would have been based on use cases derived from the Google's MapReduce (MR) case study [9] or case studies of Web-Services of Amazon[1]. Alternately, tasks could have been generated whose duration distribution is heavy-tailed – as [10] observes distribution of lifetimes of UNIX processes. Similarly, the size of simulated cloud could be based on the cluster required for Google services [9, 11]. Once basic framework is achieved, then exotic policies (some mentioned in Future Work) could have been tried. With the limitations and experiences (discussed later), only MR Sort scenario in [9] has been simulated.

In the next few subsections, the configurations of a user, a broker and a DC are described. Note, these configurations are driven by MR sort scenario from [9], and the usage costs for resources and bandwidth from Amazon Web Services (AWS).

4.1 Datacenter configuration

Resources = 1800 hosts

Resource computation capacity = 2GHz[2]

Core per resource = 2 (x 2 hyper-threads) = 2 (effective) cores/host

Memory = 4gb (only 2.5 – 3gb available)

Aggregated bandwidth = 100-200 gbps

Total execution time (tet)[3] = 600s = 10minutes

Bandwidth per core = average aggregated bandwidth / cores = [(100 + 200)/2] / [1800 x 2] = 42mbps

Cost per unit execution per unit time = 0.006c/core/s

Cost per unit bandwidth = 1.7x10^{-6} c/gbps

4.2 User configuration

Data = 10^{10} number of 100 bytes records ~= 1Tb

It is stated that (initial) data is split into 64mb pieces.

Map tasks (M) = 15,000 tasks

Reduce tasks (R) = 4,000 tasks

Number of tasks (#(tasks))= M+R = 15,000 + 4,000 = 19,000 tasks

computational requirement per task (cr) is unclear in the paper.

Assuming, cr = data / (#(tasks) x tet) = (10^{10} x 100)/(19,000 x 600) ~= 88,000 cycles

Assuming data processed by each task (dpt) = data / #(tasks) = (10^{10} x 100) / 19,000 ~= 53mb/task

4.3 Broker Configuration

Execution time is the actual time for useful processing of a task. It includes data transfer.

Not all the tasks desired or submitted for execution are successful. For example, if the deadline is reached or budget exceeded then DC refuses to service anymore tasks. Number of successful tasks are captured by *#(successful tasks)*.

Total task execution time is the aggregation of execution times of all individual tasks and data transfer times for those tasks. It does not include overhead.

Average execution time per task (*avg task time*), as the name implies, is average duration of a task execution and defined as:

Avg task time = Total task execution time / #(tasks)

Total run-time is the duration of the run. It includes both overhead and execution times.

Cloud management overhead is relevant. It includes any time that has not been spent executing the tasks, and includes scheduling, creation of VM, management and other housekeeping.

Overhead is defined as:

Overhead = total run time – success_time

Efficiency (α) is a metric to quantify the relationship of task execution time and overhead, and defined as:

α = 100 x (success_time) / (success_time + overhead)

Throughput (*thruput*) is a ratio of successful tasks per unit time and defined as:

Thruput = #(successful tasks) / total run time

The monetary cost incurred is captured in *Cost*.

4.3.1 Policies

The following policies were used in evaluation:

1. RoundRobin (RR): tasks are submitted in roundrobin to VMs in the cluster regardless of existing load in the cluster.
2. RoundRobin per VM (RR/VM): Each VM exists on a single core. Every VM is assigned one task in a round robin fashion. Pending tasks are queued in the broker till a task completes and the VM become available.

[1] Amazon Web Services, (31.1.10), http://aws.amazon.com/solutions/case-studies/

[2] Units have been expanded in Appendix A (Glossary)

[3] Actual run-time is 891s. 600s was mistakenly chosen for analysis.

3. Deadline: A RR/VM variation with a (soft) deadline (e.g. 1 million seconds) is assigned for the tasks to execute.
4. Budget: A RR/VM variation with a (soft) budget. Tasks are executed on a cloud against a pre-allocated budget (e.g. $50).

Only RR is available in Csim. Csim was extended to provide others policies.

4.4 Experiments

With all the configurations in place, the MR was simulated using the four policies stated above.

Initial set of experiments include data transfers, tasks execution time and management overhead. The reported times are simulated time. Excerpts of the experiments are depicted in Table 1.

First row of Table 1 is the Google MR Sort [9]. The rest of the rows lists the performance of different scheduling policies some with constraints.

Note, overhead is in years. In addition, overhead dominates rest of the metrics. Troubleshooting indicates the problem with simulation of data transfer. Data transfer is not modelled in Csim. So, Csim was extended to account data transfer. Troubleshooting included running Csim with and without data transfer. The results of executing one task is in Table 2. Note, extremely high overhead when data transfer is modelled. It is 500 times more compared to when data transfer is not modelled.

The MR scenario was repeated without data transfer. Results are presented in Table 3. Overhead is still high but we observe much more realistic Total run time.

4.5 Discussion

A complete simulation must include data transfer. With current data transfer extensions, the overhead is unbelievably high. Without data transfer, the Total run time is more realistic but still 10 times higher than total run time of Google MR Sort Scenario [9]. This may be an indication of inappropriate parameter values or even a defect in Csim. Suspicion lies in improper extension of Csim for data transfer. It could also be due to mismanagement of simulated clock. Further, troubleshooting has been left for future.

Instead, further efforts were directed towards documenting experiences with Csim and suggesting some improvements for a cloud simulation.

5. EXPERIENCES WITH CLOUDSIM

Buyya et al. [6] suggests four stakeholders: user, broker, data center (DC) and market. However, Csim only models users, brokers and DC. There is no notion of market in Csim. This has many consequences. DC do not expose the capabilities of its resources. The broker is unaware of the computation and data capabilities of a DC. A broker acts on behalf of a user, requests a VM with certain characteristics to be created in DC. Therefore, matchmaking between VM and host happens in the DC. A DC declines VM creation submission if there are no resources with user specified resource requirement. Otherwise, a DC accepts all incoming VM creation requests even if the resources are already busy.

Csim does not explicitly model the relationship between computation, cost and data. Rather, this relationship is split over multiple places. In addition, it is unclear how they are related from [6].

Buyya et al. [6] do not explicitly distinguish between system time and simulated time. They perform three experiments targeted to evaluate Csim, only reporting simulated time for one experiment.

The user is charged only for memory and storage used on the creation of VM. There is no monetary charge for execution time or bandwidth usage. Further, the input and output data for a task execution has no effect on start or execution time, or the (monetary) cost. Essentially, data transfer is not being modelled for calculating execution time or bandwidth usage.

The cost is being accumulated in a DC. The broker cannot state its budget when submitting tasks to a DC. It is argued that costs be maintained both in the data center and the broker. This will allow the DC to halt further execution of a broker if the budget is up as well as allow broker to manage the tasks amongst multiple DCs.

When a user submits the task to a broker, the broker in turn submits the task to the DC immediately without queuing regardless of the fact if enough resources are available in the DC or not.

The broker performs match-making between requirements of a task and available resources [12]. To evaluate policies, the broker would be a key decision maker. However, it does not have enough information to make effective scheduling decisions. Broker should be aware of computation and data capabilities of a DC before requesting a VM creation.

The computation, data and cost variables are interrelated. However, the units of variables are unclear and confusing. The consequence for lack of clarity was that absurd numbers were being reported when experiments were conducted. For example, the execution time of cloudlets and processing power of resources was unclear. That is, if the processing power is in MIPS then the execution time should be in ms given the size of cloudlet. But this is not so. So, the understanding of the relationship between variables was reverse engineered by experimentation and numerical analysis.

There is poor support for monitoring and collecting statistics.

Dynamic behaviour of a real cloud or unpredictability is not modelled in Csim.

All the parameters of the simulation have to be specified before the simulation is started. That is, once a simulation starts no further parameters can be added or removed. This may be required if we want are unsure of parameters values or parameter values are dependent on the running simulation.

For example, all the tasks have to be specified before starting a simulation. It is not possible to add further tasks into the simulation. There is a partial work around for this problem, which is to add time delay in the broker but this still requires most parameters of the simulation to be specified right from the outset.

5.1 Discussion

With such limitations, Csim needs to be extended with some fundamental features before any interesting studies could be conducted. Critical they may be, such extensions may not themselves contribute to research. Instead, they would become the base for studies. Inspired by the work of Kotsovinos [13] on a practical model of global public computing and by learning from the experiences of Csim, the core features of a cloud are suggested.

6. CORE FEATURES OF A CLOUD SIMULATION

6.1 Fundamental Concepts

Many systems have been developed that address pure computation demands of the applications. Examples include Load Sharing Facility (LSF) [14], (Classic) Condor [15] and Portable Batch System (PBS) [16]. Similarly, many systems address pure data demands of the applications. Examples include Google File System [17], Data Management System in EGEE Datagrid [12] and Gemfire [18, 19]. As a consequence, applications that are both compute and data intensive cannot fully exploit the aggregated power of underlying resources on a grid [20]. Often, ad hoc methods are used to compensate the shortcomings of a middleware. Recent research aims to provide an integrated middleware to offer both computation and data capabilities [21, 22].

These middlewares are deployed on an existing DC, so upfront capital expenditure is not an explicit part of cost equation. Maintenance cost, though important, is often treated as a supplementary issue.

It is argued that workloads fundamentally need computation and data capabilities for some time to perform useful processing. For example, Google MR Sort [9] dealt with computation power of 1,800 hosts and computation requirement of 19,000 tasks, and data requirements of 1TB; and delivered result in 891s. So, workload management policies balance computation, data and time to satisfy application objectives.

Clouds add a fourth dimension i.e. (monetary) cost. Therefore in a cloud simulation, the most interesting aspect to study is the relationship between computation, data, time and cost. It is argued that this relationship underlies clouds and must be made explicit in any cloud simulation.

Stake holders of a cloud deal with this tuple. For example, a user seeks to find a DC that satisfy its computation and data requirements with some time expectations aiming for minimum cost. Similarly, a DC may wish to offer a (computation, data, time) combination while minimizing its capital and maintenance cost, and maximizing its profits (charging cost to its users) in competition with other cloud vendors.

Other factors can really be translated into these dimensions. For example, Quality of Service (QoS) means some kind of guarantees on computation, data and time with increased costs both to a user and a DC.

Higher computation and data capabilities (reducing time) require more servers with more memory (transient and permanent) with costs naturally increasing.

Transferring data over a network translates into data transfer capacity per unit time, which is a measure of data and time. It also requires some computation. So, data transfer has also some costs.

6.2 System vs. Simulated

In a real cloud, all the entities measured are real. There is nothing imaginary. Take the example of time. All reported times are factual. A task, say, took 40s to execute. There is no notion of simulation. In simulation, however, we need to manage simulated time and system time separately. In this case, the system time refers the time taken by the underlying system to execute the simulation. A part from being reasonable, simulation time is uninteresting and irrelevant.

We extend this concept of system and simulated to all the entities or concepts. That is, for computation, data, time and cost.

System and Simulated concepts must be explicitly distinguished.

6.3 Building Blocks of a Cloud Simulation

At present, the user and broker are not separate. Also, there is no concrete notion of a market. If many cloud vendors appear in future, then separate brokers and markets may become relevant.

In this case, there will be four stake holders in a cloud: User, Broker, Datacenter, and a Market. All need to be simulated for a complete cloud simulation.

User: A user has some computation and data requirements.

Broker: A broker acts on behalf of a user and attempts to find the (near) optimal DC that satisfies user's requirements. Initially, a broker is unaware of any DC and it seeks relevant DC in a market.

Datacenter (DC): A DC advertises its computation and data capabilities in a market:

- Computation: In a real cloud, there are many multi-core interconnected hosts. A VM is typically hosted on these cores. User tasks execute on these VM. For a simulation, all these entities are relevant.
- Data: In a real cloud, some memory and storage is required to process transient data and to store persistent data, respectively.

Market: A market is a central place where DCs advertises their capabilities and where broker searches for a DC with required capabilities. Brokers and cloud vendors may enter into negotiation.

A user may bypass broker if it knows an appropriate DC. Similarly, a broker may bypass the market if it knows an appropriate DC.

6.4 Units

The units of data, computation and cost need to be modelled explicitly. Similarly, all the units should be relative to each other. For example, storage space available on a typical cloud resource could be 2GB. Then, the bandwidth per core available to each core should be in megabytes. If task size is in hundreds of MB then processing power of resources should be in GHz. It does not make sense to have task size in mb and processing processing power in kHz.

6.5 Realism

Dynamic: A real cloud is dynamic. Examples include: users and brokers coming and leaving; number of resources changing in a DC; broker becoming too busy and refusing to service anymore users etc.

Unpredictability: A real cloud is unpredictable. Unknown delays and failures may occur. Resources in DC performing house keeping tasks. They may become unavailable. New resources may be added. Network congestion may occur. Packets may be lost.

7. RESEARCH IDEAS

Since cloud emerged in last few years, it is hoped that there are many low hanging fruits. Once evaluated in simulation, the promising policies can be further evaluated on a real cloud.

The platform once developed can be a fundamental base for other interesting studies. The core models for decision making in the policies can either be hard (e.g. analytic, algorithmic etc.) like [23] or soft (e.g. fuzzy controlled, neural networked etc.) like [24]. A comparison of these models can be made for their suitability on the cloud.

Another interesting study is the recently suggested framework of autonomic workload execution [25]. Pairs of (cost models and optimization algorithms) can be empirically evaluated for their effectiveness – effectiveness being minimal execution cost and time in this case.

8. CONCLUSIONS

In this paper, we argue that cloud simulation is a suitable platform to develop and evaluate scheduling policies for workloads in clouds.

We shared our experiences of modelling a cloud scenario with some scheduling policies. We also reflected on developing policies in an existing cloud simulation (csim), followed by observing shortcomings in evaluating those policies.

We then presented core features that should be modelled by a cloud simulation for interesting studies including policies.

9. ACKNOWLEDGMENTS

Prof. Patrick Martin, Andrew Brown, Queen's University.

10. REFERENCES

[1] Foster, I., et al. *Cloud Computing and Grid Computing 360-Degree Compared*. in *Grid Computing Environments Workshop, 2008. GCE '08*. 2008.

[2] Yu, J. and R. Buyya, *A Taxonomy of Workflow Management Systems for Grid Computing.* eprint arXiv:cs/0503025, 2005.

[3] Armbrust, M., et al., *Above the Clouds: A Berkeley View of Cloud Computing*, in *Technical Report No. UCB/EECS-2009-28*. 2009, University of California at Berkeley.

[4] Coveney, P.V., et al., *The application hosting environment: Lightweight middleware for grid-based computational science.* Computer Physics Communications, 2007. **176**(6): p. 406.

[5] Mian, R., *Evaluating Workload Management Techniques for Cloud Computing -- a proposal*, in *Data Management in Cloud Computing*. 2010, Queen's University: Kingston.

[6] Buyya, R., R. Ranjan, and R.N. Calheiros. *Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities*. 2009. Leipzig, Germany: IEEE Computer Society.

[7] Buyya, R. and M. Murshed, *GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing.* Concurrency and Computation: Practice and Experience, 2002. **14**(13-15): p. 1175-1220.

[8] Sakellariou, R. and H. Zhao, *A low-cost rescheduling policy for efficient mapping of workflows on grid systems.* Scientific Programming AxGrids 2004, 2004. **12**(4): p. 253-262.

[9] Dean, J. and S. Ghemawat, *MapReduce: simplified data processing on large clusters.* Communications of the ACM, 2008. **51**(1): p. 107.

[10] Harchol-Balter, M. and A.B. Downey, *Exploiting process lifetime distributions for dynamic load balancing.* ACM Transactions on Computer Systems, 1997. **15**(3): p. 253.

[11] Barroso, L.A., J. Dean, and U. Holzle, *Web search for a planet: The Google cluster architecture.* IEEE Micro, 2003. **23**(2): p. 22-8.

[12] Gagliardi, F., et al., *European DataGrid project: Experiences of deploying a large scale testbed for E-science applications*, in *Performance Evaluation Of Complex Systems: Techniques And Tools - Performance 2002 Tutorial Lectures*. 2002. p. 480-499.

[13] Kotsovinos, E., *Global public computing*, in *Technical Report*. 2005, Computer Laboratory, University of Cambridge: Cambridge.

[14] Zhou, S.N., et al., *Utopia - A Load Sharing Facility For Large, Heterogeneous Distributed Computer-Systems.* Software-Practice & Experience, 1993. **23**(12): p. 1305-1336.

[15] Litzkow, M.J., M. Livny, and M.W. Mutka. *Condor-a hunter of idle workstations*. in *8th International Conference on Distributed Computing Systems.* 1988.

[16] Henderson, R.L. *Job scheduling under the Portable Batch System*. 1995. Berlin, Germany: Springer-Verlag.

[17] Sanjay, G., G. Howard, and L. Shun-Tak, *The Google file system.* SIGOPS Oper. Syst. Rev., 2003. **37**(5): p. 29-43.

[18] Yuhanna, N. and M. Gilpin, *Information Fabric: Enterprise Data Virtualization*, in *Technical White Paper*. 2006, Forrester Research: Cambridge.

[19] Yuhanna, N. and M. Gilpin, *Gemfire: The Enterprise Data Fabric*, in *Technical White Paper*. 2006, Gemstone: Beaverton.

[20] Mian, R., *Statement of Intent (Short)*, in *Application for Universities*. 2008: Toronto.

[21] David, J.D., et al., *Clustera: an integrated computation and data management system.* Proc. VLDB Endow., 2008. **1**(1): p. 28-41.

[22] Michael, I., et al., *Dryad: distributed data-parallel programs from sequential building blocks*, in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*. 2007, ACM: Lisbon, Portugal.

[23] Wolski, R., *Dynamically forecasting network performance using the Network Weather Service.* Cluster Computing, 1998. **1**(1): p. 119-32.

[24] Yu, K.M., et al., *A fuzzy neural network based scheduling algorithm for job assignment on computational grids*, in *Network-Based Information Systems, Proceedings*. 2007, Springer-Verlag Berlin: Berlin. p. 533-542.

[25] Paton, N.W., et al., *Optimizing Utility in Cloud Computing through Autonomic Workload Execution.* IEEE Data Engineering Bulletin, 2009. **32**(1): p. 51-58.

11. Appendix A – Glossary

Symbol	**Definition**
Hz	Hertz
G,g	Giga
B,b	Byte
gbps	gb per second
m	mega
Tb	Tera byte
wk	week
hr	hour
s	second
yr	year
c	Cent
M	Million

Policy	#(successful tasks)	Total task execution time	Success-time (s)	Avg task time (s)	Total run time	overhead(yr)	α (%)	Thruput (tasks/s)	cost($)
Google MR Sort [9]	*19,000*	*Unknown*	*Unknown*	*Unknown*	*600s*	*Unknown*	*Unknown*	*21*	*n/a*
RR	19,000	8.1 wks	1,353.40	256.43	2.56 yrs	2.41	6.05	2.36E-04	326.57
RR/VM	19,000	1.5 wks	1,527.60	48.24	2.56 yrs	2.53	1.14	2.36E-04	89.23
Deadline (1 Ms)	235	3.1 hrs	3.15	48.24	27 wks	0.52	0.07	1.44E-05	18.01
Budget	10,647	5.9 days	428.01	48.24	1.44 yrs	1.42	1.14	2.36E-04	50.00

Table 1 – Studying Google MR Sort [9] on a (simulated) cloud with different policies *with* data transfer.

Data transfer	Policy	#(success-ful tasks)	Total task execution time (s)	Total run time (s)	Over-head (s)	α (%)	cost(c)
modelled	RR	1	48.24	4297.4	4249.16	1.12	0.47
Not modelled	RR	1	44	53.24	9.24	82.64	0.26

Table 2 – Troubleshooting extremely high overhead.

Policy	#(successful tasks)	Total task execution time	Success-time(s)	Avg task time (s)	Total run time (s)	Overhead (s)	α (%)	Thruput (tasks/s)	cost($)
Google MR Sort [9]	*19,000*	*Unknown*	*Unknown*	*Unknown*	*600*	*Unknown*	*Unknown*	*21*	*n/a*
RR	19000	7.34 wks	1,234.45	233.90	6,932	5,697	17.81	2.74	266.64
RR/VM	19000	1.4 wks	1,393.32	44.00	6,756	6,523	3.44	2.81	50.16
Deadline (5300s)	806	9.85 hrs	9.85	44.00	5,544	5,534	0.18	0.15	10.18
Budget ($10)	4338	53 hrs	106.04	44.00	7,534	7,481	0.70	0.58	11.45

Table 3 – Studying Google MR Sort [9] on a (simulated) cloud with different policies *without* data transfer.