A Novel way of Improving CPU Utilization In Cloud

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Computer Science)

by

Y S Rajath

(Roll- 211cs1065)

Under the guidance of

Prof. A K Turuk

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India
May 2013

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled *"**A Novel Way Of Improving CPU Utilization in Cloud**"* submitted by ***Y S Rajath*** is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science and Engineering** with the specialization of **Computer Science** in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: 30 May 2013

Dr. Ashok Kumar Turuk
Professor
Dept. of Computer Science and Engineering
National Institute of Technology, Rourkela
Odisha-769 008

Acknowledgment

I am grateful to numerous local and global peers who have contributed towards shaping this thesis. At the outset, I would like to express my sincere thanks to Dr Ashok Kumar Turuk for his advice during my thesis work. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth. He has helped me greatly and been a source of knowledge.

I am really thankful to my all friends and Research Scholars. My sincere thanks to everyone who has provided me with kind words, a welcome ear, new ideas, useful criticism, or their invaluable time, I am truly indebted.

I must acknowledge the academic resources that I have got from NIT Rourkela. I would like to thank administrative and technical staff members of the Department who have been kind enough to advise and help in their respective roles.

Last, but not the least, I would like to dedicate this thesis to my family, for their love, patience, and understanding.

Y S Rajath

Email : y.s.rajath@gmail.com

List of Figures

List of Abbreviations

SaaS	Software As a Service
IaaS	Infrastructure As a Service
MaaS	Metal As a Service
PaaS	Platform As a Service
VM	Virtual Machine
PACT	Parallelization Contracts
GRAM	Grid Resource Allocation Manager

Abstract

Cloud computing is the collection of best software practices derived from distributed computing systems like cluster and grid. Emerging as a superior design amongst them. Cloud not only has the capability of batch processing and massive massive parallel data computation like its parents but also, things hitherto was not imagined. The main concentration of our work is on IaaS(Infrastructure as a service) aspect of cloud. Wherein, the customer avails the computation resource over simple web interface. though the cloud has most of the good design choices of its parents, certain drawbacks of them have crept in. This may be the result of the lack of efficient preexisting tools or naive implementation/setup of system at certain layer. In our work, we've optimized the performance by identifying some of those implementation choices making the overall system more efficient. This thesis is an extension of the work on Nephele. Which is a parallel data processing framework(still experimental).

Nephele facilitates on demand alloacation of resource by deducing few informations from the users. Using these informations, Nephele splits the overall job to smaller stages and resource for the same is allocated. By doing so, unnecessary resource allocation like in conventional systems is being avoided. thus, eventually increasing the CPU utilization. In our proposal, we deduce few more informations from the user which helps in making more optimal scheduling of the resources.

Contents

Chapter 1

Introduction

1.1 What is Cloud?

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [11]. In science Cloud computing is a synonym for distributed computing over a network and means the ability to run a program on many connected computers at the same time. The popularity of the term Cloud computing can be attributed to its use in marketing to sell hosted services in the sense of Application Service Provisioning that run Client server software on a remote location. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.

Essential Characteristics:

On-demand self-service A consumer can unilaterally provision computing capabilities, such asserver time and network storage, as needed automatically without requiring human interaction with each services provider.

Broad network access Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

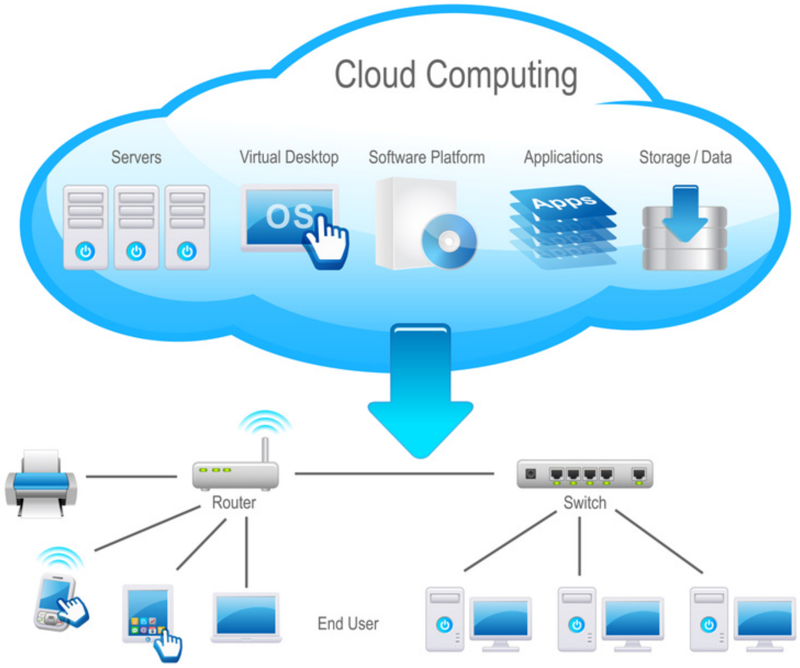

Figure 1.1: Cloud Overview

Resource Pooling The providers computing resources are pooled to serve multiple consumers using a multi - tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

Rapid Elasticity Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out, and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

Measured Service Cloud systems automatically control and optimize resource use by leveraging a metering capability 1 at some level of abstraction appropriate to the type of service (e.g.,storage, processin g, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported

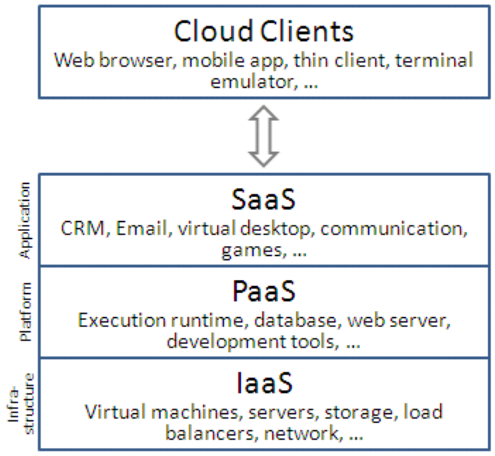
Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
Application
SaaS
CRM, Email, virtual desktop, communication, games, ...
Platform
PaaS
Execution runtime, database, web server, development tools, ...
Infra-structure
IaaS
Virtual machines, servers, storage, load balancers, network, ...

Figure 1.2: Service model

,providing transparency for both the provider and consumer of the utilized service.

Service Model:

Cloud Software as a Service(SaaS) The capability provided to the consumer is to use the providers applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g.,web-based email). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

Cloud Platform as a Service(PaaS) The capability provided to the consumer is to deploy onto the cloud infrastructure consumer -created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations

Cloud Infrastructure as a Service(IaaS) The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

Deployment Models:

Private Cloud The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.

Community Cloud The cloud infrastructure is shared by several organization s and supports a specific community that has shared concerns (e.g.,mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.

Public Cloud The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

Hybrid Cloud The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

1.2 IaaS in Cloud

The main work of this thesis revolves around the idea of IaaS. When we talk about cloud, first thing which people associate with it is virtualisation. This enables host of new possiblities to the cloud as explained before. IaaS comes in one of the service models of cloud. The capability is provided to the consumer wherein, he

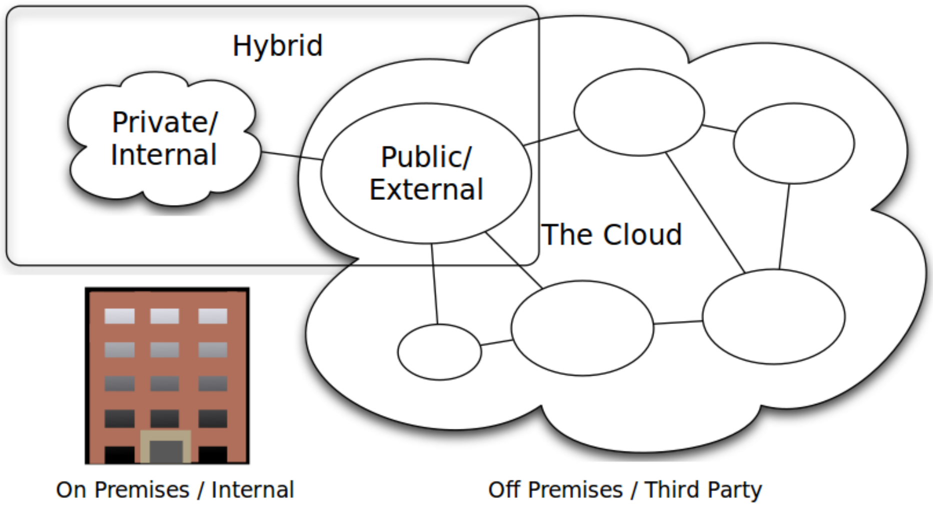

Figure 1.3: Deployment model

can provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. but, this also means that developers need to handle software and tools themselves, from operating system and up to their application. In some cases this is advantageous, for instance when deploying native libraries and tools that applications rely on such as tools to convert and edit images or video files. But in other cases this is not necessary and choosing this service model can be manpower in-effective for companies as developers must focus on meta tasks.

The main focus in IaaS, is to provide the consumer flexibility to not manage or control the underlying cloud in- frastructure but to control over operating systems, storage, deployed applications, and possibly limited control of select networking components. Users have control over which operating system they want, in some cases users can only pick from a set of pre-configured operating systems. It is common for providers to include both Linux and Windows in their selections. Some providers such as Amazon let users upload their own disk images [2]. A similarity to VPS is that operating systems are not manually installed, when selecting an operating system this is copied directly into the instance pre-installed and will therefore be instantly ready for usage.

The following figure1.4 shows the sample architecture of Xen Framework and relative positions of hardware, hypervisor, guest and host machines. The guest machines as shown, is created and controlled by the host(dom0). Properly configured guest system behaves exactly like the host system, giving a view to the user that there are multiple systems. This comes at the cost of heavy duty hardware i.e., multiple cores and surplus amount of RAM. In a simple words, virtualisation enables us to maximize the utilisation of the underlying metal.

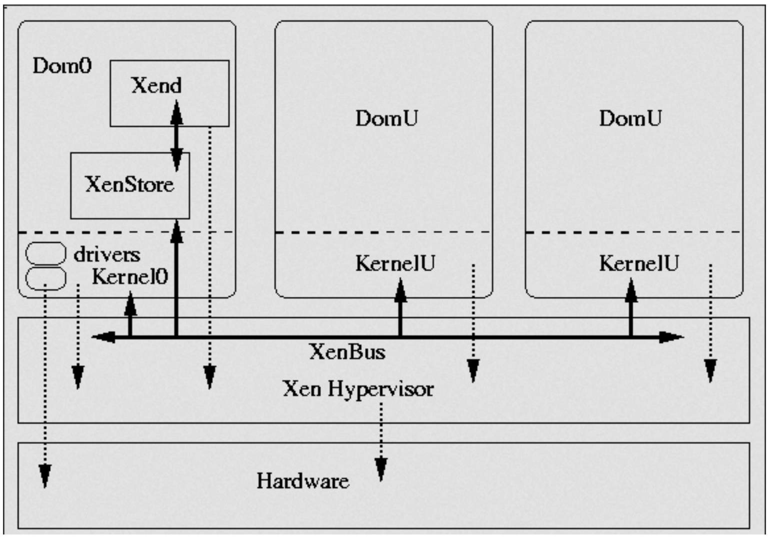

Figure 1.4: Xen Framework

1.3 Motivation for the work

Current data processing frameworks expect the cloud to imitate the static nature of the cluster environments they were originally designed for. though, this approach doesn't affect the cluster systems but, when it comes to cloud, the resource gets locked, reducing the efficiency. for e.g., at the moment the types and number of VMs allocated at the beginning of a compute job cannot be changed in the course of processing, although the tasks the job consists of might have completely different demands on the environment. As a result, rented resources may be inadequate for big parts of the processing job, which may lower the overall processing performance and increase the cost. Same is shown below1.5 The resource allocated for vm3 is unused until vm1 and vm3 finishes its tasks. This resource cannot be used by any

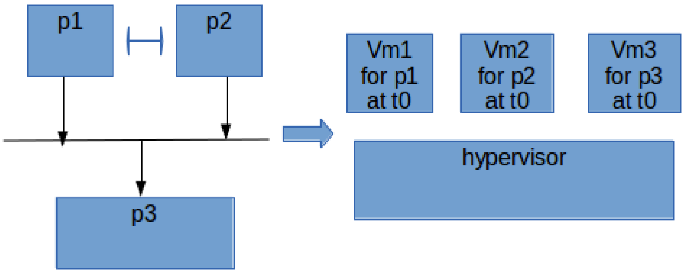

Figure 1.5: Conventional VM's creation strategy

other tasks.

Another problem with the conventional scaling strategy is that, the resource scaling may or may not be possible every time. This is pictorially shown in this figure1.6 The extra resource allocation, like memory in the above example, is possible only if the node has sufficient memory. The task cannot be broken down to two or more parts and schduled on different machines(provided the task can be parallelized). since, we cannot rely on avalibilty of resource in the same node. Nephele's way of scheduling the tasks solves this problem.

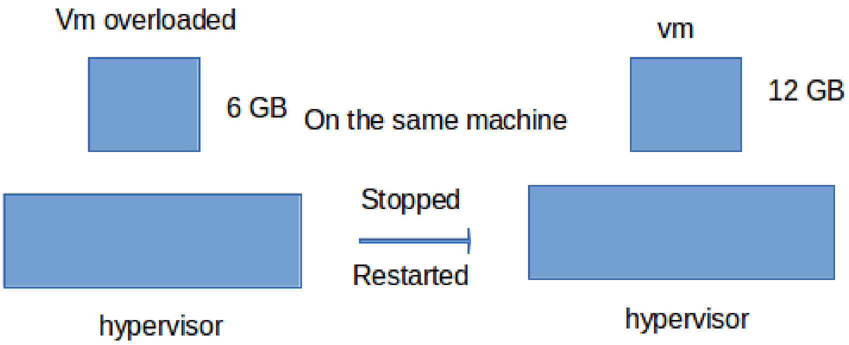

Figure 1.6: Conventional scaling

The solution to these two problems in cloud, opens a various apportunities to make the cloud efficient. These problems were the bottleneck to the performance. The naive cloud management framework and inferior data processing frameworks called for procurement of additional resources to mitigate this issue. Where it is totally uncalled for. And from the customer's perspective, it meant more money.

1.4 Objective of the work

The Nephele enables optimization in hosts of new areas in cloud computing hitherto unexplored due to implementation difficulties. This work shows one such optimization based on image mangement and scheduling of the job. The main objective of this thesis is to, Firstly, increase the CPU utilisation by reducing few delays. Secondly, we propose a model which reduces the unnecessary processes/deamons in the virtual machine instances which takes up cpu time.

1.5 Thesis Organization

Chapter 1 contains definitions and some basic concept on cloud cloud computing. The motivation section shows the limitations of the present cloud scenario and explains how the Nephele architecture opens plethora of areas for research purpose. Then, the objective section later explains the goal of the work.

Chapter 2 is on literature survey i.e., it lists all the publications, white papers and online documentations went through by the author in making this work possible.

Chapter 3 introduces basics of the Nephele architecture. Problem statement they've worked upon and its impact in the field of cloud computing. Later, the chapter ends with few notes on scheduling strategies adopted in cloud and some shortcomings of the same.

Chapter 4 is on the proposal if new strategy in Data processing framework based on Nephele. Challenges faced in implementation of the same is discussed and finally how the problem is tackled.

Chapter 5 is on Simulation and results. The experimental setup used for simulation is mentioned here. Graphical representation of the system is shown later.

Chapter 6 finally concludes the work and future future scope of the work is discussed.

Chapter 2

Literature survey

The work on dynamic resource allocation was first published in 2009 by warneke and adej [17] in a workshop conducted by ACM. They have published IEEE transaction on the same in Aug 2012 [16]. until then, it was just a concept. The work in this transaction shows the simulation done by them and realtime performance analysis viz-e-viz its predecessors [18] i.e., hadoop.

Nephele borrows certain features like, firstly, MapReduce programming model(or the open source version Hadoop) which is designed to run data analysis jobs on a large amount of data stored across a large set of share- nothing commodity servers. Slightly modified version of the same model is given by Alexandrov et. al in PACT [1] which is adopted by nephele. Secondly, the concept of staged execution from Dryad [9] from Isard et al.

Many open source cloud management Framework is available like, OpenStack [14], Cloud Foundry, Eucalyptus [12] and Nimbus is evaluated [13] for suitability in running the simulation in our work. Canonical backed Openstack is available in ubuntu from release 12.04+. Dedicated community and modular design of OpenStack has made this a choice of our framework.

The custom image creation in this work is done by using Boxgrinder. Boxgrinder [6] written in Ruby, is a project by SIG/fedora-cloud enables user to create custom image for cloud. This is deployable in Amazon Web Service. Kickstart templates are also provided in the site for beginners.

The comparison of cloud and grid [8] by Foster et. al. and the topic on GRAM [15] in the journal Grid computing: Making the global infrastructure a reality, sheds light on how jobs are scheduled and what all factors are considered in scheduling the tasks. Finally, askubuntu.com, stackoverflow.com and various other forums have been of great help.

2.1 Nephele - The parallel Data Processing Framework

TODAY a growing number of companies have to process huge amounts of data in a cost-efficient manner. Classic representatives for these companies are operators of Internet search engines, like Google, Yahoo, or Microsoft. The vast amount of data they have to deal with every day has made traditional database solutions prohibitively expensive. Instead, these companies have moved on to clouds which is a next generation of distributed computing. Problems like processing crawled documents or regenerating a web index are split into several independent subtasks, distributed among the avail- able nodes, and computed in parallel.

In order to simplify the development of distributed applications on top of such architectures, many of these companies have also built customized data processing frameworks. Examples are Googles MapReduce, Microsofts Dryad, or Yahoo!s Map-Reduce-Merge [19]. Although these systems differ in design, their programming models share similar objectives, namely hiding the hassle of parallel programming, fault tolerance, and execution optimizations from the developer. Developers can typically continue to write sequential programs. The processing framework then takes care of distributing the program among the available nodes and executes each instance of the program on the appropriate fragment of data.

Cloud computing has emerged as a promising approach to rent a large IT infrastructure on a short-term pay-per-usage basis. Operators of so-called IaaS clouds, like Amazon EC2 [2], let their customers allocate, access, and control a set of virtual machines (VMs) which run inside their data centers and only charge them for the period of time the machines are allocated. The VMs are typically offered in different types, each type with its own characteristics (number of CPU cores, amount of main memory, etc.) and cost.

Nephele is a data processing framework whichThe Application developed using The job manager as seen in the figure determines the dependencies between the

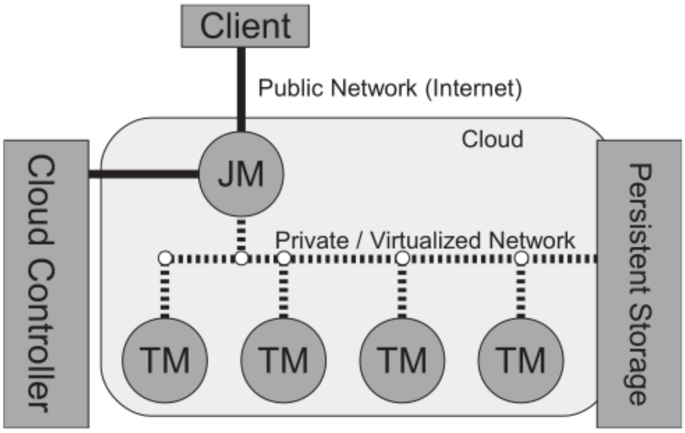

Figure 2.1: Neplele Framework

job and splits it into stages. The resources are allocated only for the next stage and not the entire task. Thus, increasing the overall CPU utilisation

2.2 Scheduling in cloud

Cloud computing is the outcome of mixture of best practices adopted from grid computing, cluster computing and various other high performance/reliable computing. It is a emerging pattern adopted by many companies. Compared with grid computing, cloud computing has some new features, such as location transperency and elasticity. grid computing in general is the integration of fragmented, heterogeneous distribution resources; cloud computing is the large-scale data center resources which are more concentrated. In addition, virtualization technology hides the heterogeneity of the resources in cloud computing. grid is generally used in science computation, and for solving special-purpose domain problem. cloud computing is user-oriented design which provides varied services to meet the needs of different users. It is more commercialized and the resources in cloud computing are packed into virtual resources by using virtualization technology. This calls for resource allocation process.

Present day cloud borrows the same strategy as seen in grid. Though, the strategy seems realistic and optimal, it fails to harness the complete power of the underlying metal. The data processing frameworks like Hadoop [4] aids in making optimal allocation at the time of disaptch. But, during the execution of the job, bad choices are made which calls for unnecessary large amount of redundant resources.

The basic mechanism of cloud computing is to dispatch the computing tasks to the resource pool which constitutes massively large number of computers. It enables a variety of applications to gain on-demand resource for ex: computing power, storage and a variety of software services according to their needs. The job scheduling intricacy in cloud computing is left to the hypervisor layer level. A step above as that of grid systems. This enables lots of new design models for scheduling module. Further, it perpetuates a number of new features which aides in accomplishing different strategies for job scheduling. Hadoop for ex, schedules the task near a node where App data is available i.e., it takes job to the data. This reduces network traffic and other delays. similarly, Dryad allows the developer to specify the order of execution, this reduces unnecessary wastefull resource allocation.

Chapter 3

Proposed Model

In a cloud, when a job is submitted to a shceduler node(gateway between user and cloud), the scheduler node first determines the resource demands. This demand consists of how many VMs to create, what should be the RAM, Secondary memory and no. of cores in each VMs. Using this information, The schduler then fetches a image(from the image store node) to various compute node(node where actual computation takes place) depending on the availability of the resource in it. Then, scheduler issues the command to boot the image(using the Hypervisor) with the specifications as given by the job. The image so booted, can take the job and perform the necessary computation(this is what is meant by Virtual machines). When we say, that there are N No. of virtual machines running in a node, it means that there are N instance of images (same or different) are being booted and monitered by the hypervisor of that node.

The selection of these images by the scheduler is based on the request of the job. Job may request its computation be done in Red-Hat environment, Ubuntu environment, Windows environment or any other generic environment provided by the cloud service or in fact, a user can make his own custom image and ask for the schduler to run the task in his custom environment. When the generic image doesnt have the necessary features, user can utilise the service from other cloud providers or user can put the packages in the app data and it will be installed to the virtual machine.

3.1 Minimal Image Approach

When a user requests for a generic image, lots of services(deamon processes) is not utilised every time. These services are just stealing the CPU cycles. Simple solution to this is, user to run the tasks in his own image. But, this is not the elegant solution to the problem. This unnecessarily increases the network traffic and most importantly, user may not have the knowledge to create his own image. If time is of the essence, then, user can only rely on the vendor to perform this task.

To solve the above problem, we have slightly changed the Nephele framework to include one additional feature in its APIs. Nephele framework derives basic informations like user, task dependency, order of execution etc from the application program. in addition to this, we have added on more attribute called Services-Required(SR). This stores the serivces used by the sub task. This is an array of strings. When the job manager of Nephele, comes across this during scheduling, it fetches the required minimal image(superset of SR) from the store and boots it. The following fig4.2 shows the process.

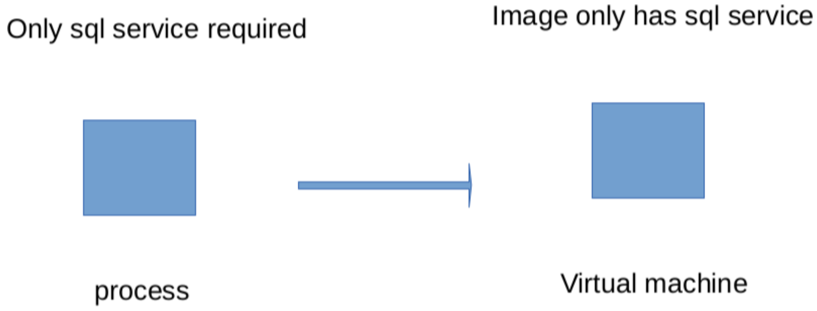

Figure 3.1: Minimal image approach

The information contained in the sub task about the SR is later used in creation of the small custom images containing only that purticular serivces when the nodes are free. This way, the large generic image is progressively broken down to smaller images during the lifetime. So when the next time the sub-task with same SR comes, only that small image is loaded.

3.2 Challenges

The first challenge in implementing our model is that, Generic images are available on every node. But, the large no. of custom images so created in our model are usually stored in distributed location thus, the images of the next stages needs to be prefetched to improve the throughput. As significant delay occurs between the stages in fetching the images. There is also unavoidable network traffic. The simple solution we've adopted is by firstly, sending the dummy request of the next stage to the scheduler. Next, prefetch the images to the nodes returned and make a soft reservation of resources on those nodes. When the request for next job comes, we readily schedule the sub-tasks on the reserved nodes. If the resource is not available, then the default scheduling takes over. The problem with this approach is that it behaves more like a conventional scheduling as seen in Hadoop.

The second challenge is, how to manage the increasing no. of images? the approach we've used is distribute the images evenly among all nodes. This method is simple to implement. But, may cause uneven traffic in the network. This is because, some images are used more often than others or most of the frequently used ones are in the same node.

Chapter 4

Results and Simulations

This section shows the result of various simulations. Firstly, figure 4.1 shows the delay in seconds to load a image of size 50 MB. We see a significant delay in fetching a image from the other node in the same network. Secondly, figure 4.2 shows the RAM usage by two images. The point here is, job which requires only small set of service, if run in a suitable minimal environment, there is lots of RAM available to the task. Also the delay in time required to boot the image is reduced. Finally, figure 4.3 shows the comparison between our proposed model and Nephele. A simple linear, three-staged task with a three different sort of image in each stage is provided as a batch script.

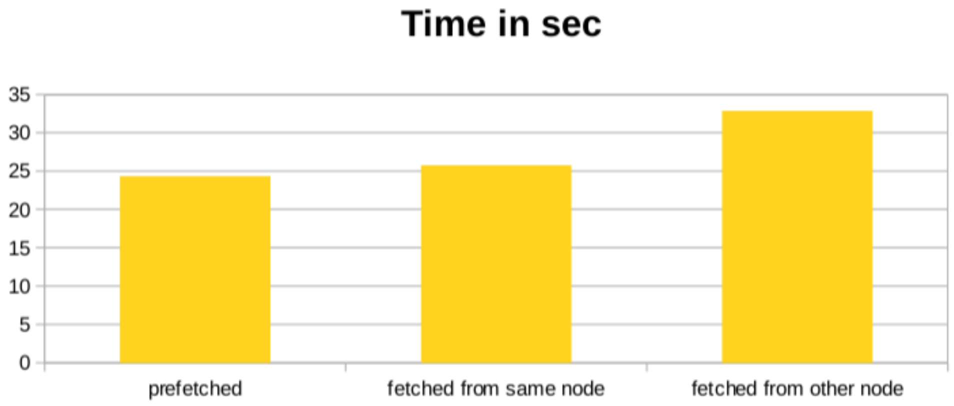

Figure 4.1: Latency in loading the image to RAM

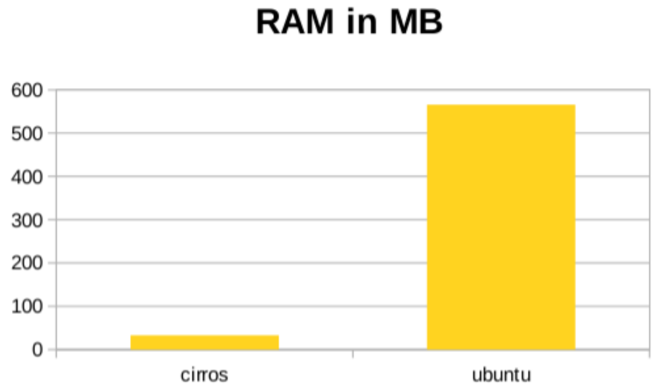

Figure 4.2: Benefit of using a minimal image

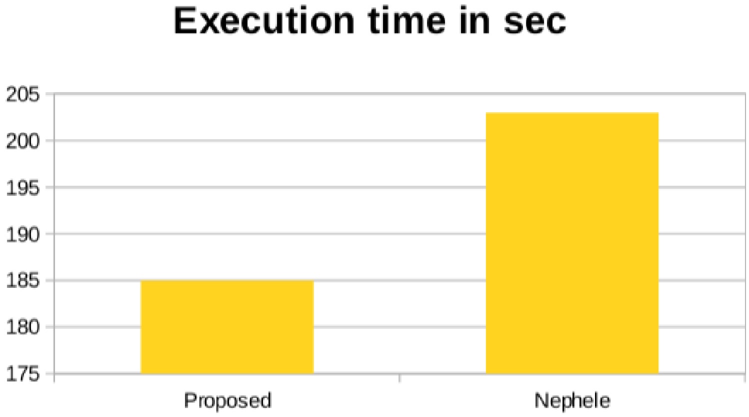

Figure 4.3: Proposed Model vs Nephele

4.1 Limitations

There are lots of limitation in using our model. Firstly, Prefetching images on the node has some adverse effect if the node is low on RAM. This is because, less memory is available to the host. This makes the overall VMs it is supporting seem slower. Secondly, Images takes up more space on disc than conventional approach. There is need for manual or automated deletion of images which are no longer needed.

Chapter 5

Conclusion and Future Scope

We have discussed about what is cloud, how cloud enables efficient model for renting resources on-demand, motivation for our work and the objectives in the begining chapters. The following literature survey shows the various manuals, technical papers and forums which helped in making this thesis. Later on, from our discussion on the present work, proposed model and various simulations, we have seen how the proposed work minimizes wastefull cpu time. Thus increasing the throughput.

The core of our model is the Nephele Data framework. Dynamic behavior of the Nephele data framework, enables us to make runtime decisions based on the application program written by the developer using this framework. We can safely say, that our work was possible only because of this. The next section suggests the future scope of our work.

Future Scope

The strategies used in implementing the work is not totally optimal and there is always a better implementation possible. Not only that, but also, different strategies needs to be employed in different cloud to get the optimal results. these decisions can only be made by analysing the behavior of the cloud under consideration and the results will be specific to that cloud only. There is a scope of improvement within our work and outside our work.

Within our work, firstly, there is a scope of implementing a better scheduling algorithm. Secondly, when the model is implememted, by seeing the cloud behavior, we can analyse the correlation between images by modelling a clustering algorithm to group the related images. Finaly using this results, storage of various images on nodes can be optimized to reduce network traffic.

As we have proposed the model keeping in mind the throughput by keeping track of services required by the tasks similarly, we can work upon issues like reliablity, availability, cost and other aspects.

Bibliography

[1] Alexander Alexandrov, Max Heimel, Volker Markl, Dominic Battré, Fabian Hueske, Erik Nijkamp, Stephan Ewen, Odej Kao, and Daniel Warneke. Massively parallel data analysis with pacts on nephele. *Proceedings of the VLDB Endowment*, 3(1-2):1625–1628, 2010.

[2] AWS Amazon. Amazon web services, 2010.

[3] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[4] Dhruba Borthakur. The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 11:21, 2007.

[5] Damien Cerbelaud, Shishir Garg, and Jeremy Huylebroeck. Opening the clouds: qualitative overview of the state-of-the-art open source vm-based cloud management platforms. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, page 22. Springer-Verlag New York, Inc., 2009.

[6] EC2 Creation Cloud SIG. Boxgrinder : Custom image creation for your cloud. available from: https://fedoraproject.org/wiki/cloud_sig/ec2_creation. 2012.

[7] Brad Fitzpatrick. Memcached: a distributed memory object caching system, 2009.

[8] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee, 2008.

[9] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. *ACM SIGOPS Operating Systems Review*, 41(3):59–72, 2007.

[10] Mark Lutz and David Ascher. *Learning python*. O'Reilly Media, 2009.

[11] Peter Mell and Timothy Grance. The nist definition of cloud computing (draft). *NIST special publication*, 800:145, 2011.

[12] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131. IEEE, 2009.

[13] Junjie Peng, Xuejun Zhang, Zhou Lei, Bofeng Zhang, Wu Zhang, and Qing Li. Comparison of several cloud computing platforms. In *Information Science and Engineering (ISISE), 2009 Second International Symposium on*, pages 23–27. IEEE, 2009.

[14] Open Stack. Openstack manuals. available from: http://docs. openstack.org/ diablo/openstackcompute /starter/content. 2012.

[15] Douglas Thain, Todd Tannenbaum, and Miron Livny. Condor and the grid. *Grid computing: Making the global infrastructure a reality*, pages 299–335, 2003.

[16] D. Warneke and Odej Kao. Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. *Parallel and Distributed Systems, IEEE Transactions on*, 22(6):985–997, 2011.

[17] Daniel Warneke and Odej Kao. Nephele: efficient parallel data processing in the cloud. In *Proceedings of the 2nd workshop on many-task computing on grids and supercomputers*, page 8. ACM, 2009.

[18] Tom White. *Hadoop: The definitive guide*. O'Reilly Media, Inc., 2012.

[19] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D Stott Parker. Map-reduce-merge: simplified relational data processing on large clusters. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1029–1040. ACM, 2007.