An Approach for Operating System Solution for Multicore Processors in Cloud Computing

[1]Dr S. K. JHA, [2]Prof. S.P. KHADANGA, [3]Prof. S.K.OJHA

[1]Associate Professor Cum Principal Sityog Institute of Technology , Aurangabad
[2]Assistant Professor , Gandhi Institute of Advanced Computer Research Rayagada
[3] Assistant Professor , Sityog Institute of Technology , Aurangabad

Abstract : Two new categories of computational technology, multicore processors and cloud computers, have the potential to provide the typical user access to previously unheard-of computational capacity. These new hardware platforms require operating systems (OSes) in order for the user to efficiently utilize all of this processing capability. Current multicore operating systems are not cloud-compatible and cannot scale to many cores. As a result, modern cloud systems place a great degree of complexity on the user, requiring them to handle several system-level issues and manage individual Virtual Machines (VMs). 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 pre-existing 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. The paper 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 infor- mations 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.

Keywords : IaaS, VM, Nephele,Framework, Multocore ,

1.Introduction

The computing power available to the typical computer user is always growing. From mainframes to minicomputers to personal computers, users have advanced from laptops to multicore and cloud computers in recent years. For the purpose of managing devices, allocating resources, and utilizing the hardware's enhanced processing power, new operating systems have historically been created for every new class of computer hardware. New operating systems are required for the newest types of computing hardware, such as multicore and cloud computers, in order to benefit from their enhanced processing capability and to make it easier for users to use elastic hardware resources [3]. By using scheduling algorithms, resource management strategies, and hardware optimization, modern operating systems (OSs) manage requirements while also utilizing the multi-core's parallel processing capabilities. The performance of multi-core processors can be improved with the help of contemporary scheduling techniques like Rate Monotonic Scheduling (RMS) or Dynamic Scheduling like Earliest Deadline First (EDF) [6]. Partitioned scheduling (each core has its own task queue and jobs are statically assigned to certain cores) and real-time scheduling are used for multi-core use. Global scheduling (all cores share a single task queue, enabling dynamic task allocation) and hybrid scheduling are used to reduce inter-core communication and scheduling overhead. Predictable memory management, synchronization, and inter-core communication are the other strategies.

2. What is Multicore Processor ?

A computer processor that has several separate processing units, or cores, on a single integrated circuit (IC) chip is called a multicore processor. Because each core can carry out instructions both independently and simultaneously, processing capacity can be raised and multitasking abilities can be enhanced. The primary difference between a multicore processor and a single core processor lies in the number of independent processing units they possess. A single core processor has only one core, capable of executing instructions one at a time. On the other hand, a multicore processor consists of multiple cores, allowing for parallel execution of instructions [4]. An integrated circuit with two or more processor cores connected for improved performance and lower power consumption is called a multicore processor. This parallelism results in increased processing power and improved multitasking capabilities compared to single-core processors. Through the use of multithreading and parallel processing, these processors also make it possible to process numerous jobs at once more effectively. Having several distinct processors installed on a computer is comparable to a dual core configuration. One way to increase processor performance without going over the realistic bounds of semiconductor design and manufacture is to use multicore processors or microprocessors. Additionally, using multicores guarantees safe functioning in areas like heat generation.

A notion of infinite processing that can be precisely customized to match a user's needs, even if those needs change quickly, is what cloud computing and Infrastructure as a Service (IaaS) promise. IaaS solutions should therefore enable customers to buy simply the appropriate quantity of memory, processing power, input/output, and storage to satisfy their demands at any given moment. Despite the notion, current IaaS solutions do not provide customers with the same experience as if they were obtaining access to a multiprocessor computer with easily swappable memory, disk, cores, and I/O that is infinitely scalable [11]. Instead, because current IaaS platforms lack system wide operating systems, users are forced to manually manage resources and machine boundaries. If cloud computing is to live up to its potential, cloud computers must be as usable as a contemporary multiprocessor computer.

3. Multicore in the Cloud Computing context

3.1 Multicore meets clouds :

Multicore systems' explosive growth might undoubtedly help computing clouds. Uniform commodity multicore clusters can be used as compute servers in Google data centers, for instance. "Coarse grained" parallel applications are typically a good fit for computing grids. Latencies in Grid applications are typically expressed in macroseconds. Typically, computing clouds comprise multiple uniform data centers or computer centers that use commodity high performance multicore clusters. Thus, cloud computing might be used to map "fine grained" parallel applications, which could offer high data transfer rates of multicore computers and minimal cluster latencies.

3.2 Computing Infrastructure Provision from Multicore Cloud :

The context of cloud computing may be as follows :

- a computing Cloud is composed by several data centers or compute centers, and
- each center contains multiple identical commodity multicore clusters.

Users can then rent platforms and computational infrastructures from the computing clouds, which are often virtual machine-based systems. On multicore systems, the consolidation of a group of virtual machines necessitates effective scheduling and management. For example,

- how can a virtual machine be scheduled across multiple cores, multiple processors, or even many clusters?
- How to move a virtual machine when slots of cores are available?

3.3 Programming specifications for cloud computing with multicores :

To meet the needs of multicore-based computing clouds, appropriate programming models and paradigms need be created from the user's perspective :

- **Task Level atomicity :**

Users typically receive computing resources in the form of virtual machines, and one task is assigned to one or more of these machines. Virtual machines are the fundamental building block of resource provisioning and management. User applications should adjust to this by being atomic in the task unit, which is precisely specified with its control, actions, and goal.

- **Stateless Task :**

Rather to promote state ful l task in Grid Computing , Stateless tasks could be easily managed, e.g.,checkpointing, task migration, both in the user level and system level. Stateless tasks therefore can bring scalability and fault tolerance in homogeneous multicore cluster environments, in a Cloud.

4. Operating System Level virtualization

Multi-Core CPU Virtualization is a technology that allows a single physical CPU core to function as multiple 'virtual' cores, thereby improving the overall performance and efficiency of a system. It enables the system to multitask more effectively by splitting the workload between the virtual cores. This core concept of virtualization is key for understanding how modern computing environments, particularly in the field of cloud computing, operate to deliver high-performance solutions. A virtualization platform, such as VMware, is designed to abstract the software environment from the underlying hardware. Virtualization is capable of abstracting physical processor cores into virtual processors or central processing units (vCPUs) which are then assigned to virtual machines (VMs). Each VM becomes a virtual server capable of running its own OS and application. It is possible to assign more than one vCPU to each VM, allowing each VM and its application to run parallel processing software if desired.

Operating system virtualization is a part of virtualization technology and is a type of server virtualization. Operating-system-level virtualization, also known as containerization, refers to an operating system feature in which the kernel allows the existence of multiple isolated user-space instances. Here the kernel of an OS allows more than forked isolated Virtual Space instance to exists. These instances are known as Virtualization engine and the partition is known as called containers, partitions, virtual environments (VEs) or jails (FreeBSD jail or chroot jail), may look ike real computers from the point of view of programs running in them.

In brief, a OS level VM is virtual execution environment that can be forked instantly from the base Operating System. A computer program running on an ordinary operating system can see all resources (connected devices, files and folders, network shares, CPU power, quantifiable hardware capabilities) of that computer. However, programs running inside a container can only see the container's contents and devices assigned to the container. Several containers can be created on each operating system, to each of which a subset of the computer's resources is allocated. Each container may contain any number of computer programs. These programs may run concurrently or separately, even interact with each other. The instances are running on top of the previous host operating system and feature a set of libraries with which applications interact, illustrating how they run on a machine dedicated to their use. With Operating System Virtualization nothing is preinstalled or permanently loaded on the local device and no hard disk is needed. Everything run from the network using a kind of virtual disk.

5. Multi-Core CPU Virtualization

First, it's critical to understand what virtualization of multi-core CPUs is. It basically entails one CPU core acting as several virtual cores. 'Hyper-threading' is a technique that makes a CPU capable of managing several threads of execution at once. It's an advanced technique that increases the CPU's capacity for multitasking, which raises system effectiveness and performance. By effectively establishing an abstraction layer between the hardware and the software, virtualization enables the simultaneous operation of several operating systems on a single physical host.

Although the idea of virtualization has existed for a number of decades, the introduction of cloud computing has greatly increased its popularity. Cloud providers use this technology to create cost-effective, scalable, and adaptable solutions that may be tailored to their customers' unique requirements. These providers can guarantee the best possible use of their hardware resources by employing multi-core CPU virtualization, which lowers expenses and enhances service quality.

The hypervisor is an essential part of the virtualization process. This software controls the virtual machines and distributes resources like memory, storage, and CPU time. The host system is managed by the hypervisor, which also makes sure that every virtual machine has an equal amount of resources. Type 1 hypervisors operate directly on the hardware, while Type 2 hypervisors run on an operating system much like any other software program. In order to keep the virtual computers isolated from one another, the hypervisor is essential. Because it guarantees that an issue in one virtual machine won't impact the others, this is crucial. Additionally, it makes it simple to move virtual machines across several physical hosts, which is helpful for load balancing as well as for preserving redundancy and high availability.

6. Benefits of Multi-Core CPU Virtualization

The virtualization of multi-core CPUs has several advantages. The potential to optimize hardware consumption is among the most obvious. Conventional computer models frequently result in unused or idle resources. However, virtualization ensures efficient use by allowing several virtual machines to share the same physical resources. This lowers expenses while also increasing the overall effectiveness of the system.

The flexibility it provides is an additional advantage. Depending on the needs, virtual machines can be swiftly and simply established, changed, or removed. This enhances responsiveness and agility by enabling the quick rollout of new services and apps. Additionally, virtualization improves the security and dependability of systems. A malfunction or security breach in one virtual machine does not affect the others because they are all separate from one another.

7. Challenges in Multi-Core CPU Virtualization

Virtualization of multi-core CPUs has many advantages, but it also has drawbacks. The intricacy of running and maintaining a virtualized environment is one of the primary obstacles. To guarantee optimum performance, virtual machines must be appropriately configured and watched over. To avoid any one virtual machine monopolizing the system's resources, the hypervisor must also distribute resources efficiently.

The requirement for adequate hardware resources is another difficulty. Although virtualization improves resource efficiency, it does come with a high memory and processing power cost. The host system must therefore be sufficiently configured to meet the requirements of virtualization. Last but not least, while virtual machine isolation improves security, it also makes a potential target for cyberattacks. Thus, in a virtualized environment, strong security measures are crucial.

8. Multicore and Cloud Operating System Challenges

Numerous typical operating system issues are brought on by cloud computing infrastructure and multicore processors. This section outlines the primary issues that we feel exist.

8.1 Scalability : Cloud computing and IaaS systems have been growing in popularity at the same time as the multicore revolution. Programming models and the computing business could be drastically changed by this new paradigm in computing [7]. The demand from users for hosted computing platforms has been a major factor in the rapid growth in the number of computers added by cloud computing providers. A particular cloud user has access to far more resources than a non-cloud user does. With the exception of financial limitations, a user's access to cloud resources is essentially limitless. Thus, it is clear that scalability is a major concern for future OSes in both single machine and cloud systems.

8.2 Demand Variability : The fact that demand is dynamic is one of the main similarities between multicore systems and cloud computing. Additionally, demand fluctuates far more than it did in earlier systems, and the quantity Compared to single-core or fixed-sized cluster systems, the range of accessible resources can be significantly wider.

Current system designers are compelled to match the available resources to the demand in order to achieve optimal power utilization. From chip design to data center operating costs, heat and energy consumption have an impact on computing infrastructure. Consequently, fos aims to minimize heat generation and power consumption while upholding the user-imposed throughput criteria.

8.3 programming Challenges : It's challenging to create cloud apps with multiple components spread across numerous computers. The main cause of this is that virtual machines, which are used in modern IaaS cloud systems, impose an additional layer of indirection. On cloud systems, a large portion of this complexity is forced into the program by fragmenting the application's view of the resource pool, whereas on multiprocessor systems, the operating system handles scheduling and resource management.

Additionally, there isn't a standard programming model for inter-machine and intra-multicore communication. According to the current programming paradigm, a cloud programmer must create a threaded application in order to utilize intra-machine resources, whereas socket programming is utilized to connect with application components running on other computers.

These large-scale hierarchical systems are challenging to program, but they are also becoming increasingly complex to manage and load-balance. In the past, ad hoc solutions like hardware load balancers have been used to address these kinds of problems. These solutions are frequently restricted to the VM level, which is the only level of the hierarchy. However, in the context of FOS, load balancing can be implemented within the system, more finely than at the virtual machine or single machine level, and generically (i.e., it can be applied to any communications rather than just TCP/IP traffic).Additionally, our architecture eliminates the requirement for the application developer to understand load balancing.

9. Related Works

There are several classes of systems which have similarities to Minimal Image Approach: traditional microkernels, distributed OSes, and cloud computing infrastructure.

9.1 Core Affinity in Multicore System

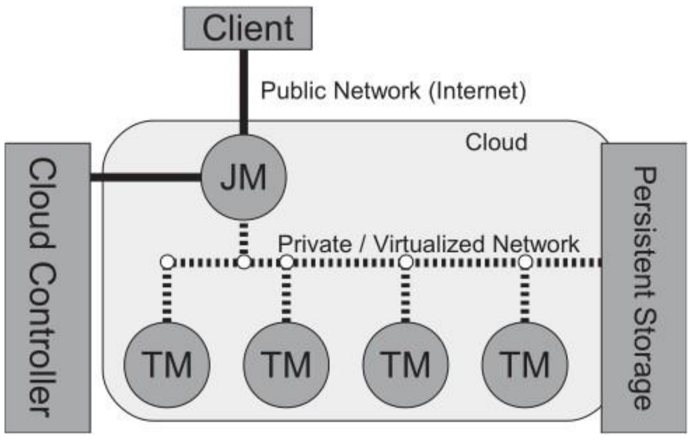
By attaching a process or several processes to a particular CPU core, CPU affinity makes it possible for the process or processes to operate exclusively from that core. When attempting to do a performance It is advisable to run several instances of a process, each on a separate core, when testing on a host with numerous cores. This makes it possible to use the CPU more.

Cache locality is improved when activities that communicate often or share similar resources are grouped on the same core, which results in lower latency and faster data retrieval. Furthermore, core affinity helps prevent resource contention—which can happen when jobs are dispersed across multiple cores—for resources like CPU time or memory bandwidth. The overhead of moving data between cores can be reduced by keeping workloads on the same core, increasing system efficiency overall. Core affinity is especially important in real-time systems because it lowers the chance of missing deadlines and ensures that tasks have predictable execution patterns, which helps fulfill strict timing constraints. In works like those by Liu and Layland[3][i], as well as in more recent research on scheduling algorithms and multicore processor optimization, where core affinity is frequently a crucial component for resource management and performance enhancement in complex systems, this idea is extensively examined.

9.2 Nphele- The parallel Data Processing Frame work

Many of these businesses have also developed specialized data processing frameworks to make it easier to construct distributed applications on top of such infrastructures. MapReduce from Google, Dryad from Microsoft, and Map-Reduce-Merge from Yahoo! are a few examples [19]. Despite having different designs, all systems' programming models have similar goals: to conceal from the developer the difficulties of parallel programming, fault tolerance, and execution optimizations. Usually, developers can keep creating sequential programs. The program is then distributed among the available nodes by the processing framework, which also runs each instance of the program on the relevant data fragment.

A promising method for renting a sizable IT infrastructure on a pay-per-use basis for a brief period of time is cloud computing. Operators of so-called infrastructure as a service (IaaS) clouds, such as Amazon EC2 [2], allow their clients to assign, access, and manage a collection of virtual machines (VMs) that operate within their data centres. They only charge their clients for the time that the machines are assigned. Virtual machines are usually available in a variety of types, each with unique features (such as the number of CPU cores, main memory, etc.) and prices.



Nephele is a framework for data processing that the application was created with As seen in the image, the job manager divides the work into phases and establishes the dependencies between them. The resources are not allotted for the task as a whole, merely for the subsequent phase. Consequently, raising the total CPU utilization .

10. Proposed Model

In a cloud, when a job is submitted to a scheduler 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 scheduler 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 monitored 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 scheduler to run the task in his custom environment. When the generic image doesn’t 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.

11. Minimal Image Approach

Many services (deamon processes) are not always used when a user requests a generic image. The CPU cycles are being stolen by these services. Running the jobs in his own image is a straightforward solution to this problem. However, this is not a sophisticated way to handle the issue. In addition to needlessly increasing network traffic, this also raises the possibility that the user lacks the skills necessary to produce his own image. The user can only depend on the vendor to complete this task if time is of the essence.

We have made a little modification to the Nephele framework to add one more capability to its APIs in order to address the aforementioned issue. The Nephele framework uses the application software to extract fundamental data such as the user, task dependence, execution order, etc. Furthermore, we have included an additional characteristic known as ServicesRequired (SR). The services used by the subtask are stored here. An array of strings is what this is. Nephele's work manager retrieves the necessary minimal image (superset of SR) from the store and boots it when it encounters this during scheduling.

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.

12. Results and Simulation

This section shows the result of various simulations. Firstly, figure 1 shows the Figure 1 first displays the loading time in seconds for a 50 MB image. When attempting to retrieve an image from the other node in the same network, we observe a notable latency. Second, two images' RAM utilization is displayed in figure 2. The key idea is that a job that needs a small collection of services might use a lot of RAM if it is executed in an appropriate minimum environment. Additionally, the amount of time needed to boot the image is decreased. The contrast between Nephele and our suggested model is finally displayed in picture 3. As a batch script, a straightforward linear, three-step task with three distinct types of images in each stage is included .

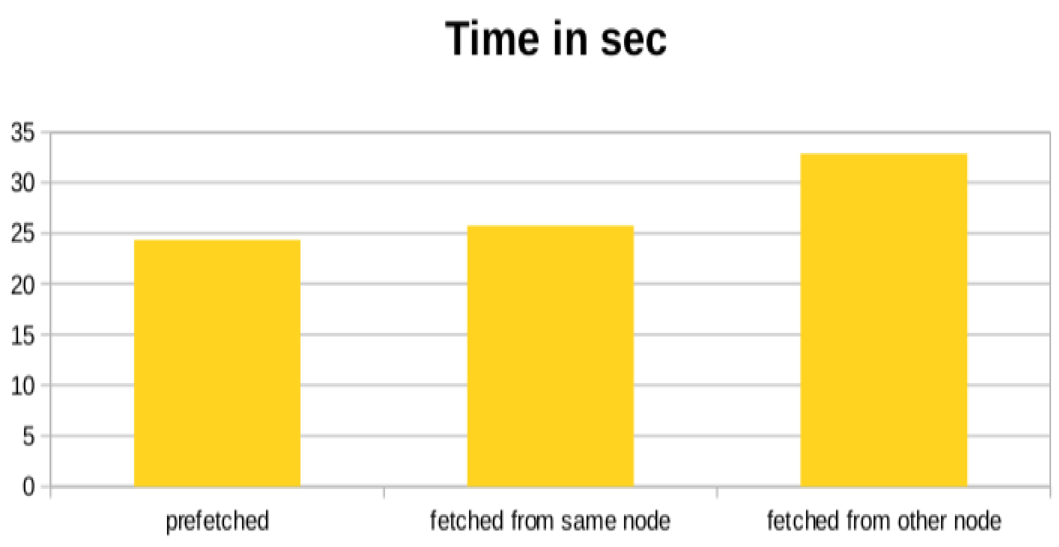


Figure 1 Latency in Loading the Image

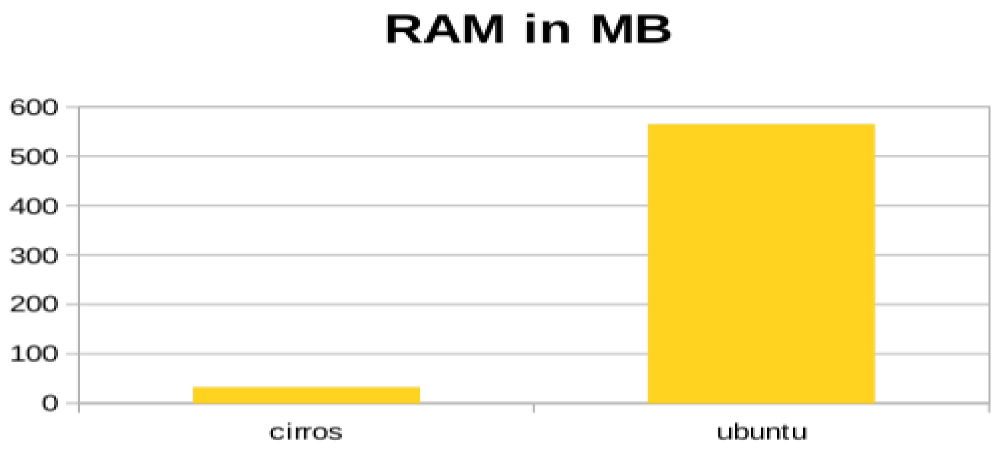


Figure 2 : Benefit of using a minimal image

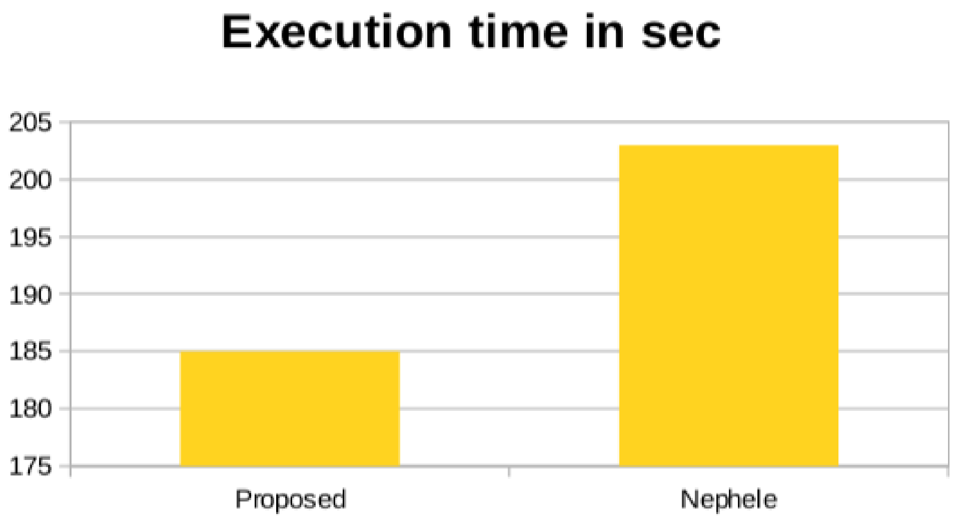


Figure 3 : Amount of time needed to boot the image is decreased

13. Conclusion and Future Scope

In the introductory chapters, we have covered the definition of cloud computing, how it facilitates an effective model for on-demand resource rental, motivation for our work, and the goals. The several manuals, technical articles, and forums that aided in the creation of this thesis are listed in the literature survey that follows. After discussing the current work, the suggested model, and several simulations, we were able to observe how the suggested work reduces unnecessary CPU time. Consequently, the throughput is increased**.**

The Nephele Data structure forms the basis of our model. The Nephele data framework's dynamic behavior allows us to make judgments at runtime based on the application software that the developer has created. We can confidently state that this is the only reason our work was feasible. The scope of our work going forward is suggested in the next section.

Future Scope

The strategies used in implementing the work is not totally optimal and there is al- ways 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 be- havior, 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.

We can address issues like availability, affordability, and reliability because we have suggested a model that considers throughput by monitoring the services needed by the jobs in a similar manner.

14. References

[1] Real-Time Systems: Scheduling, Analysis, and Verification by Jane W. S. Liu

[2] Real-Time Systems by C.M. Krishna and Kang G. Shin

[3] Handbook of Real-Time and Embedded Systems by Insup Lee, Joseph Y.-T. Leung, and James H. Anderson

[4] "Multicore Processors: Resource Allocation and Scheduling" in "Handbook of Multicore Embedded Systems"

[5] "Partitioning and Scheduling in Real-Time Systems" in "Real-Time Systems Design and Analysis"

[6] "Scheduling Algorithms for Multiprogramming in a HardReal- Time Environment" by C. Liu and J.W. Layland (1973)

[7] "Dynamic Load Balancing Using Work-Stealing" by Daniel Cederman and Philippas Tsigas

[8] "Partitioned Scheduling in Multiprocessor Real-Time Systems" by R. Baruah and D. Burns

[9] “Work-Stealing Algorithm Distilled” by Ryan Zheng

[10] “Real Time Systems in Hospital” by Velibor Bozic

[11] “Affinity-Based Task Scheduling on Heterogeneous Multicore Systems Using CBS and QBICTM” by Shoaib

[12] Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA) IJCA Volume 186 – No.62, January 2025

[13] IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)

[14] US Patent 9,491,183 B2 - "Method for Task Partitioning in Multicore Processors"

[15] US Patent 8,425,498 B2 - "Dynamic Load Balancing in Multithreaded Processing Systems"

[16] "Dynamic Load Balancing and Task Scheduling in Parallel Systems" by John Doe

[17] "Scheduling Real-Time Tasks in Multicore Systems: A Study of Partitioning and Core Affinity" by Jane Smith

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

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

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

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

[23] 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.

[24] EC2 Creation Cloud SIG. Boxgrinder : Custom image creation for your cloud. available from: https://fedoraproject.org/wiki/cloud sig/ec2 creation. 2012.

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