ISSN:0975-9646

Performance Evaluation Of Map/Reduce Workload Modeling Based Distributed Resource Handler For Cloud Computing

Purva Keshare, Prof. Sachin Mahajan

Dept. of Computer Science and Engineering,
Jawaharlal Institute of Technology Borawan Khargone (M.P), India

Abstract— **Cloud computing supports scalable processing task which requires heterogeneous resources provided on demand for end user. These resources follow the specific models for their services like infrastructure, platform and software with distributed and parallel. The system primarily uses virtualization technology to cope with the computing needs. By using virtualization dynamic resource requirements can be handled with service optimizations. These dynamic handling suffers from the performance and utilization issues due to their heterogeneous environments and availability. For effectively allotting the resources to different process, they must be analyzed previously for detecting their occupancy and residual means. Servers are continuously monitored for detection of overutilization and underutilization for understanding the loads of the system. Over the last few decades the grid has evolve form Globus based system to Nephele architecture for processing the parallel and distributed jobs. This work gives a novel way to deal with the problems associated with scheduling of distributed resources using runtime handling and statics analysis. The system is capable of doing the task associated with complete job processing and their allocation and de-allocation. During their decisions regarding the distribution of the job and the load the cloud takes the decision based on some previous heuristics and offers robust load handling for bigger size data. At the analytical evaluation the work seems to satisfy the needs of innovative cloud area. In future the trace driven simulations and experimental results demonstrates the good performance.**

Keywords— **Cloud Computing, Grid Computing, Resource Handling, Distributed Resource Handler, Statics Analysis, Scheduling;**

I. INTRODUCTION

Cloud computing is the new paradigm providing the user oriented services which are scalable in nature. Normally this scalability is achieved by implementing the basic construct of cloud i.e. virtualization. Here the issue mainly evolves in managing migrations of multiple virtualization platforms and multiple virtual machines across physical machines without disruption method. Here the applied computing must ensures the load balancing when multiple virtual machines run on multiple physical machines. The field of managing resources and there scheduling start with Cloud based working environment and extended to cloud arena. Various resource discovery mechanisms are being developed in the paradigm of distributed systems. Goal of almost every mechanism is efficient and effective resource management in fault tolerant and scalable manner. Since in the real world of computing the Underlying environment is heterogeneous and highly unpredictable therefore the mechanisms have to be optimized and sometimes combined for proper resource discovery and management. Cloud inherits most of the properties of conventional distributed systems. Resource management in Cloud has more or less same goals of other distributed systems, but with the difference that Cloud is organized in much better way. Aim of this work is to provide an practical implementation scenario through existing Cloud resource discovery mechanisms and introduce a novel Cloud scheduling methodology to enhance the processing criteria in accordance with resource recovery management protocol.

There is a need of cloud computing that responds to various requests more quickly. For that there are various approaches suggested previously to optimize resource grouping based on criteria such as delay, bandwidth and semantics in order to select the resources more quickly and appropriately. Along with that they also gives the new orientation of applying different scheduling methodology on these parallel clouds. Dealing with these varying resource request and devices are termed as the area of dynamic resource allocations (DRA). It deals with the virtualization of machines which cloud be migrated effectively on any host for serving the parallel processing. Virtual machine monitors is the controlling mechanism designed for handling of the dynamic resource requirements of the cloud. It should also support the elastic nature and can be able to expand or compress as per the service requirements. The dynamic results confirmed that the virtual machine which loading becomes too high it will automatically migrated to another low loading physical machine without service interrupt. And let total physical machine loading reaching balance. It is however unclear whether this technique is suitable for the problem at hand and what the performance implications of its use are.

This work emphasizes on tractable solution for scheduling applications in the public cloud. In the same method becomes much less feasible in a hybrid cloud setting due to very high solve time variances. In the cloud model is expected to make such practice unnecessary by offering automatic scale up and down in response to load variation. It also saves on electricity which contributes to a significant portion of the operational expenses in large data centers. The solution also includes a set of heuristics that prevent overload in the system effectively while saving energy used. It traces driven simulation and experiment results demonstrate that our algorithm achieves good performance.

II. Background

Following are the General Issues in Resource Management for Cloud:

(i) Exploitation of underutilized resources

Main idea is to distribute the work load to an underutilized resource over the Cloud. Consider various cases in which machines are in their idle states or in peak utilization states. Therefore if an application is running on a busy machine, further applications or jobs could be executed on some other idle machine(s) on the Cloud. This idea is not new in domain of distributed computing but Cloud provides a framework to exploit such underutilized resources in a very effective and broad way. There are two fundamental requirements in order to execute an application on a remote machine. The perspective application should have the ability to execute remotely without any considerable over head. Secondly the remote machine should satisfy the resource requirements of the application.

(ii) Parallel CPU Capacity

This is one of the major benefits of Cloud. This computing capability has wide industrial application ranging from Bio-medical and High energy physics. The actual utilization of this potential depends on the design of applications us in this computation facility. These applications should have the capability to be divided in to sub applications or sub jobs, so that an application could be submitted to distributed machines. The scalability of application lies in the way of subdivision into sub-jobs i.e. the more the division of application the more is the scalability.

(iii) Virtual resources and virtual organizations for collaboration

Enabling and simplifying collaboration among wider entities has been Cloud’s major contribution. Distributed systems developed in the past have provided this facility and have been quite successful in achieving this goal. Cloud has enhanced this capability to a higher scale incorporating very heterogeneous systems and providing various standards and frameworks. In Cloud users can be dynamically grouped into Virtual Organizations (VOs) with their own policies. Thus these VOs can share their resources in larger Cloud.

(iv) Access to additional resources

Besides the storage and processing resources, Cloud provides access to various additional resources. These resources could range from remote printing to sharing software licenses. There are many expensive scientific equipments with capabilities of remote access, Cloud exploits this state of the art resource access and makes it available on an ordinary end machine.

(v) Resource Balancing

As described earlier Cloud federates vast resources into a single large virtual resource. Various mechanisms have been proposed and developed for resource balancing on the Cloud. For occasions of peak load these resource balancing mechanisms could be vital. This may be done in two ways. Either an unexpected peak could be transferred to a considerably idle machine on the Cloud or in case of full utilization state of Cloud, low priority jobs could be suspended and the jobs with higher priority could be executed. This could be accomplished through proper reservation and scheduling of resources.

(vi) Reliability

Reliability is one of the fundamental goals of any distributed system. Usually hardware reliability is achieved through redundancy of equipment. In Cloud the underlying software technology offers more than hardware based reliability. The Cloud management software resubmits a job to alternate machines in case of failures or in some case a critical job’s multiple instances are executed over different machines.

III. Literature Reviews

During the last few decades there are various approaches developed and analyzed for scheduling purposes. Starting with the operating system enhancements followed by Cloud oriented approaches and the recent support from cloud computing. Among them those which are relating with the work is taken out here as literature survey and summarized as:

The paper [9] covers some of the advanced topics with Cloud development such as implementation of virtualization in operating systems. It is useful in many scenarios: server consolidation, virtual test environments, and for Linux enthusiasts who still cannot decide which distribution is best. One of its recent development example are Kernel-based Virtual Machine, or kvm, is a new Linux subsystem which leverages these virtualization extensions to add a virtual machine monitor (or hypervisor) capability to Linux. Using kvm, one can create and run multiple virtual machines. These virtual machines appear as normal Linux processes and integrate seamlessly with the rest of the system.

Dryad [10] is a general-purpose distributed execution engine for data parallel applications that combines computational vertices with communication channels to form a dataflow graph. Dryad runs the application by executing the vertices of this graph on a set of available computers, communicating as appropriate through files, TCP pipes, and shared-memory FIFOs. The vertices provided by the application developer are quite simple and are usually written as sequential programs with no thread creation or locking. Concurrency arises from Dryad scheduling vertices to run simultaneously on multiple computers, or on multiple CPU cores within a computer. The application can discover the size and placement of data at run time, and modify the graph as the computation progresses to make efficient use of the available resources. The Dryad execution engine handles all the difficult problems of creating a large distributed, concurrent application: scheduling the use of computers and their CPUs, recovering from communication or computer failures, and transporting data between vertices.

An extension of above tool is *DryadLINQ* as suggested in [11]. It is a system and a set of language extensions that enable a new programming model for large scale distributed computing. It generalizes previous execution environments such as SQL, MapReduce, and Dryad in two ways: by adopting an expressive data model of strongly typed .NET objects; and by supporting general-purpose imperative and declarative operations on datasets within a traditional high-level programming language. A DryadLINQ program is a sequential program composed of LINQ expressions performing arbitrary side effect- free transformations on datasets, and can be written and debugged using standard .NET development tools. The DryadLINQ system automatically and transparently translates the data-parallel portions of the program into a distributed execution plan which is passed to the Dryad execution platform. Dryad, which has been in continuous operation for several years on production clusters made up of thousands of computers, ensures efficient, reliable execution of this plan.

The paper [12] addresses the problem of scheduling concurrent jobs on clusters where application data is stored on the computing nodes. This setting, in which scheduling computations close to their data is crucial for performance, is increasingly common and arises in systems such as MapReduce, Hadoop, and Dryad as well as many Cloud-computing environments. This paper introduces a powerful and flexible new framework for scheduling concurrent distributed jobs with fine-grain resource sharing. The scheduling problem is mapped to a graph data structure, where edge weights and capacities encode the competing demands of data locality, fairness, and starvation-freedom, and a standard solver computes the optimal online schedule according to a global cost model. The paper also gives an evaluation implementation of this framework, called as Quincy. It gets better fairness when fairness is requested, while substantially improving data locality.

The paper [13] further gives a detailed study and suggests some of changes in the recently developed model of Map-Reduce. It is used as a programming model that enables easy development of scalable parallel applications to process vast amounts of data on large clusters of commodity machines. Through a simple interface with two functions, map and reduce, this model facilitates parallel implementation of many real-world tasks such as data processing for search engines and machine learning. However, this model does not directly support processing multiple related heterogeneous datasets. While processing relational data is a common need, this limitation causes difficulties and/or inefficiency when Map-Reduce is applied on relational operations like joins. An improvements is made I the paper for Map-Reduce to develop a new approach through Map- Reduce-Merge. It adds to Map-Reduce a Merge phase that can efficiently merge data already partitioned and sorted (or hashed) by map and reduce modules. It also demonstrates that this new model can express relational algebra operators as well as implement several join algorithms.

Carrying forward the previous work on resource scheduling and optimization *the paper [14]* suggested a novel approach SCOPE (Structured Computations Optimized for Parallel Execution) which is a declarative and extensible scripting language. It is declarative because here the users describe large-scale data analysis tasks as a flow of data transformations, w/o worrying about how they are parallelized on the underlying platform. And it is extensible because it have list of user-defined functions and operators. Also it supports structured computations for data transformations consume and produce row sets that conform to a schema with optimized parallel execution. It is a yet another high-level language for large-scale data analysis. It is a hybrid scripting language supporting not only user-defined map-reduce merge operations, but also SQL-flavored constructs to define large-scale data analysis tasks.

The paper [15] suggests a many-task computing to bridge the gap between two computing paradigms, high throughput computing and high performance computing. Many task computing differs from high throughput computing in the emphasis of using large number of computing resources over short periods of time to accomplish many computational tasks, where primary metrics are measured in seconds, as opposed to operations per month. Many task computing denotes high-performance computations comprising multiple distinct activities, coupled via file system operations. Tasks may be small or large, uni-processor or multiprocessor, compute intensive or data-intensive. The set of tasks may be static or dynamic, homogeneous or heterogeneous, loosely coupled or tightly coupled. The aggregate number of tasks, quantity of computing, and volumes of data may be extremely large. At the evaluation point of view, it supports effectively parallel processing.

With the growth in parallel processing for Cloud based application, now some more factors needs to be studied. It is because of a new computing paradigm which raises its ratio in market of distributed processing. It is cloud computing which a combination of distributed, Cloud, autonomic and utility is computing.

The paper [16] covers some of this aspect of developing the above solution for cloud applications. The current processing framework which is used in cloud or cluster computing serves the different behaviour with separated requirements and tools. . As a result, the allocated compute resources may be inadequate for big parts of the submitted job and unnecessarily increase processing time and cost. Thus the paper discusses opportunities and challenges for efficient parallel data processing in clouds and present our ongoing research project Nephele. Nephele is the first data processing framework to explicitly exploit the dynamic resource allocation offered by today's compute clouds for both, task scheduling and execution. It allows assigning the particular tasks of a processing job to different types of virtual machines and takes care of their instantiation and termination during the job execution.

The paper [17] presents a parallel data processor based programming model in collaboration with the so called Parallelization Contracts (PACTs) and the scalable parallel execution engine Nephele. The PACT programming model is a generalization of the well-known map/reduce programming model, extending it with further second-order functions, as well as with Output Contracts that give guarantees about the behavior of a function. It also describes the methods to transform a PACT program into a data flow for Nephele, which executes its sequential building blocks in parallel and deals with communication, synchronization and fault tolerance.

In order to ensure cost-efficient execution in an IaaS cloud, Nephele allows allocate/deallocate instances in the course of the processing job, when some subtasks have already been completed or are already running. However, this just-in-time allocation can also cause problems, since there is the risk that the requested instance types are temporarily not available in the cloud. To cope with this problem, Nephele separates the Execution Graph into one or more so-called Execution Stages. An Execution Stage must contain at least one Group Vertex. Its processing can only start when all the subtasks included in the preceding stages have been successfully processed. The paper [18] develops a profiling subsystem for Nephele which can continuously monitor running tasks and the underlying instances. Based on the Java Management Extensions (JMX) the profiling subsystem is, among other things, capable of breaking down what percentage of its processing time a task thread actually spends processing user code and what percentage of time it has to wait for data.

IV. Problem Identification

Effective resource handling is the key area of work for dynamic allocation and handling using parallel processing. Now, after studying the various research articles, there are various direction of work had been found. Majorly the areas where most of the cloud computing performance for resource utilization depends are load distribution, managing and monitoring resources, scheduling and job queuing, feasible resource estimates with application needs etc. The cloud resource managements and dynamic request handling for resources depends of scheduling and recoveries. In scheduling the major factors which play a vital role in improvement of parallel processing are resource discovery, system selection & job execution. Among them are the sharing of resources among many users, the dependency between tasks and the possible use and production of large data sets. There are some of the identified area of work is detected as the problem is:

Traditional mechanism does not provide dynamic allocation of resources based on the load and scheduling needs. It lacks with the necessary information for taking the dynamic decision. The factors require taking the effective decision on resource handling was: load, queue size, process mapping, allocation strategy etc.

The cloud aims towards achieving the high performance computing with respect to grid based resource utilizations and optimizations. Here the virtualization technology allocates data center resources dynamically based on application demands and support effective computing by optimizing the number of servers in use. By the suggested work implementations, a support has been made to the automatic resource managements and scheduling. Thus for effectively measuring and predicting the resources optimizations above factors are necessary which leads towards improvements in load avoidance, resource handling and scheduling and will handle hidden mapping.

V. Proposed Solution

Resource discovery and management offers the standardization of scheduling and allocation policies by reducing the device dependencies. Cloud provides big data processing on heterogeneous sets of devices with different resource configurations. Somewhere the decisions taken by the providers make the system more complex and lay behind monitoring. Thus, if the occupancy of the resources are not measured and monitored regularly, it will degrade the future performance of user's application deployments. This work aims towards resolving the above problems of resource management using through the suggested distributed resource handler.

The model focus on developing an analytical decision system based on previous behaviors of resources and uses this information for their further controlling and allocations. It also aims towards developing and performance monitoring solution which leads in reduction of risk associated with usages of utilized resources. Also in cloud the single resource based application sharing and computation is easy the networked sharing based computations. Especially if the distributed and parallel processing is concerned, it should be a mandatory task. The direction of work aims towards improving the traditional resource job handling and scheduling for optimized performances in grids and cloud.

The problem of scheduling has been only designed for static, homogeneous cluster setups and disregards the particular nature of a cloud. Thus some computation might involve the benefits associated with the effective and dynamic resource managements. Cloud scheduling involves scheduling of resources over different and dispersed domains. This might involve resource searching on multiple administrative domains to reach a single machine or a single administrative domain for multiple or single resource. As mentioned before we are focusing our research on Cloud scheduler or broker which has to make scheduling decisions on an environment where it has no control over the local resources and this scheduling is closely linked. The suggested system provides the functionalities of resource request, data processing, job handling, and analysis based on statistical behaviour of resources towards allocation and deallocation.

The system starts its functionality with initial request towards the system. It means the application has requested for certain resources. The system forwards the request to the runtime scheduler. This scheduler keeps the track of individual resources attached with the system either in centralized or distributed nature. Now the processor creates the job pool having different resource requirement at distant locations.For mapping the resources with the job the system is proposed with a resource mapper for load analysis, process mapping and requirement filtering. It also records the user access patterns towards its desired resource.

This procedure is not much different than the traditional way of remote authorization i.e. the job would not be permitted to execute if the user has no access on that resource. But in the Cloud environment it is highly possible that application requirements might change according to the matched target resource e.g. depending on the system architecture the memory and computing requirements might change.

After authorization and requirements specification, next step involves the filtration of resources which do not meet the requirements.

The primary functionality which we are integrating with the traditional model is MapReduce. It has two user-defined functions. The Map function takes in a key-value pair, and generates a set of intermediate key-value pairs. The Reduce function takes in all intermediate pairs associated with a particular key, and emits a final set of key-value pairs.

To limit network traffic, users may additionally specify a Combine function that merges multiple intermediate key-value pairs before sending them to the Reduce worker. Both the input pairs to Map and the output pairs of Reduce are placed in an underlying distributed file system (DFS). The run-time system takes care of retrieving from and outputting to the DFS, partitioning the data, scheduling parallel execution, coordinating network communication, and handling machine failures. Once the resources was enlisted then the mapper and the reducer will starts operating with the resources and the controller. Here the cloud tools offer variations with their distributed and big data processing task.

Next phase is on process manager which includes collection of all this information with respect to some parameters of queuing request, load and resource availability for designing the final execution cycle and strategy. It does the pre task preparation required for runtime resource handling and making their job in a scheduled manner.

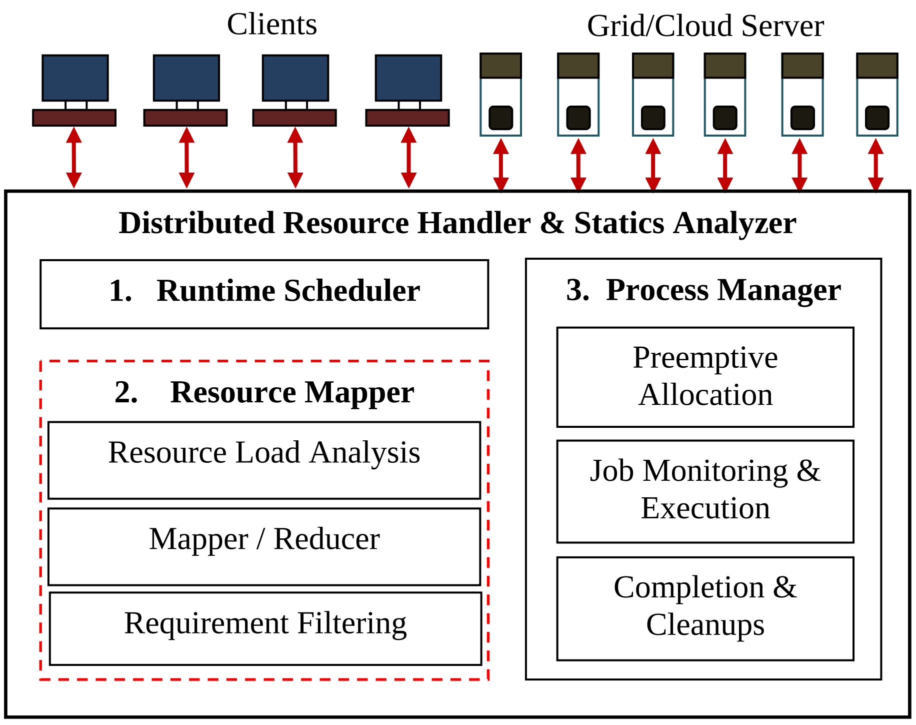


Figure 1: Proposed Workload Modeling Based Distributed Resource Handler

During this scheduling three more sub operation are performed which reserves the resource required for specific application to avoid deadlocks. Next is the job executor and monitoring which measure the performance of device with request to process size. The above identified results are transformed to the physical resource grids for service the applications requests. These steps are very similar to the steps involved in traditional computing paradigm. But these steps are carried out considering the very dynamic nature of Cloud environment.

VI. EXPECTED OUTCOMES

- A flexible, scalable infrastructure management platform has been architected and a prototype implemented
- Measurement of resource usage and end user activities lies hands of the cloud service provider.
- Opaque cost structure due to highly flexible usage of cloud services.
- Stable of cost structure
- The developed model is a resource allocation system that can avoid overload in the system effectively while minimizing the number of servers used.
- It can also deals with the uneven utilization of a server for multi-dimensional resource constraints.
- With the collected data it can be able to detect both computational as well as I/O bottlenecks.

VII. RESULT EVALUATION

This phase will prove the efficiency of solution using result statics analysis. Traditionally, the mechanism does not offers dynamic allocation of resources based on their changing load and scheduling needs. Dynamic decisions must be processed to find the best suitable scheduling algorithms for different task of distributed Mapper. The resource handling factors requires managing the information regarding the queue size, processing unit consumption and allocation strategy selection. The devised solution will prove its effectiveness using parametric evaluations. Scheduling can be dispersed as per the resource availability. We configure the solution on different execution environment of different VM sizes and resources. The solution must be given for controlled and uncontrolled manner. Some of the operations of result analysis are:

- Simulation and modeling of large scale cloud computing environment includes the VM handling.
- Provides the platform for resource mapping and allocation using précised algorithms for managing the solution at IaaS layer.
- It provides quantitative interface for proving the results on defined parameters.

This evaluation requires execution of application on various working scenarios to calculate the values of designed parameters. It will be effectively compared with some of the traditional processes on same parameters. We have selected the quantitative measurements so as the results are in the form of tables and graphs.

Table 1: Mapper Master User Table

S. No	User Name	Services Configured	Service IDs	Scheduling Selected
1	User1	Word Processor	2380	SJF
2	User2	PDF Reader	4532	RR
3	User3	Paint	5160	FCFS
4	User1	NotePad	3356	SJF
5	User2	Media Player	8912	Priority

Table Interpretation: The above table shows the service configuration details for different users. We have also applied some initial scheduling approaches to make the services work accordingly. Here the categorization was made on the basis of their service ID used to uniquely identify them.

Table 2: Reducer Request Details for Resource Allocation

S. No	User	Services	Processor Usage	Memory Usage	Occurrence	Priority
1	User1	Word Processor	1.43	1708032	0	1
2	User2	PDF Reader	0.18	1368034	2	3
3	User3	Paint	0.46	233472	1	2
4	User1	NotePad	0.07	192512	3	1
5	User2	Media Player	1.46	2048965	4	1

Table Interpretation: The above table shows the resource details configured with individual services. Here majorly the processing units and the memory was shown with their occurrence order and priorities. Each service is having its own service request patterns based on their requirements satisfied by the system with priorities.

Table 3: Parametric Evaluation of Job Allocation Using Distributed Mapper

S. No	Approach	Scheduling Selected	Processor Pool Usage (%)	Disk Usage (kB)		Memory Usage (%)		Network Usage (b/s)	
				Read	Write	Virtual	Physical	Received	Sent
1	Traditional	SJF	48.33	14.32	45.00	29.34	42.96	102.50	3.21
	Proposed		41.54	14.26	44.73	28.86	37.94	97.95	0.0
2	Traditional	RR	12.36	5.23	20.54	26.34	39.64	88.32	12.38
	Proposed		9.09	5.35	18.52	28.78	37.68	85.36	14.25
3	Traditional	FCFS	37.99	17.78	20.01	29.66	40.09	68.95	14.32
	Proposed		25.32	13.24	18.36	28.42	37.89	78.21	12.03
4	Traditional	SJF	24.22	16.37	15.21	27.33	39.21	77.70	16.20
	Proposed		15.38	15.42	20.32	28.55	37.99	76.22	12.45
5	Traditional	Priority	18.24	6.33	15.99	28.63	37.70	74.22	12.08
	Proposed		16.92	6.32	15.96	28.79	37.71	74.32	12.36

Table Interpretation: The above made a comparison between traditional and proposed approaches against the resource consumption details. While evaluating the filled records for the field for various attributes we have found some major benefits of proposed distributed mapper approach over traditional one. Evaluation was made on same scheduling method applied for both the execution scenarios. Distributed mapper had reduced the processor consumption by applying Map Reduce algorithm and hence achieves high CPU utilizations. Similar utilization was also achieved with memory and bandwidth. While in case of reading and writing the distributed mapper is giving higher values which mean that the distributed task requires higher I/O redirections with increased IOPS (Input Output per Second) requests.

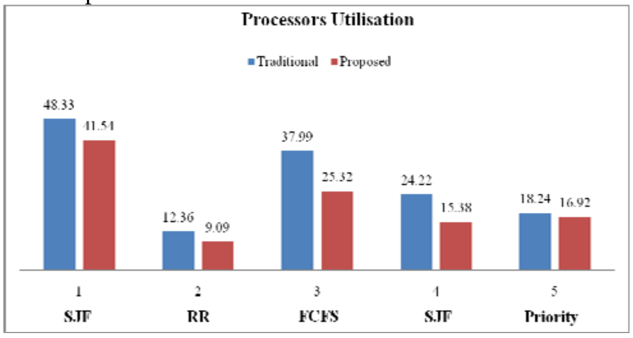
Table 4: Allocation Table for VM Instances

S. No	Memory Units (Total)	Processing Units (Total)	Bandwidth	No of VM Created	Resources Assigned		
					Memory Units (Total)	Processing Units (Total)	Bandwidth
1	4096	Quad Core (2.9 GHz)	2Mbps	4	1024	0.725	512Kbps
2	2048	Dual Core (1.3 GHz to 2.6 GHz)	2Mbps	2	1024	0.650	1Mbps
3	1024	Single Core (1.3 GHz)	1Mbps	1	1024	1.3	1Mbps
4	512	Single Core (1.3 GHz)	512 Kbps	1	512	1.3	512 Kbps

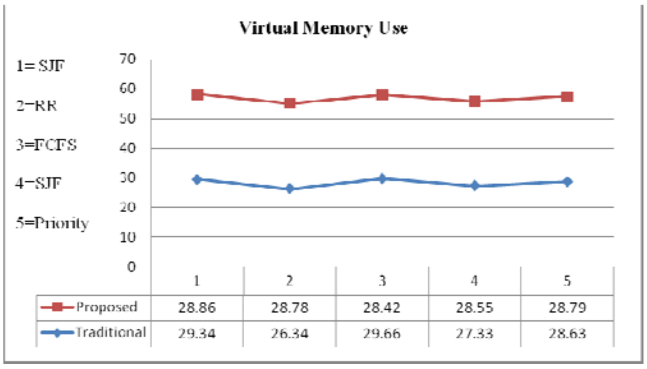
Note: 1 GHz representing 1 thousand million cycles per second (100000000 Cycles/ Sec)

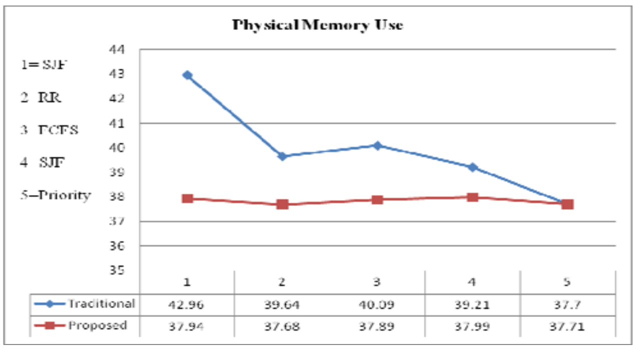
Table Interpretation: The above table shows the virtual machine creation details for different configurations. These VMs are created for executing the solution on different environmental conditions for taking the robust results. It proves that the system is behaving as per our expectations and showing optimal results with distributed mapper based allocations.

From the statistics of table, by comparing the Aneka Services for resource allocation it is found dynamic in nature. Getting the distributed control on the Mapper and reducer somewhere the systems performance is compromised but in terms of the security its robustness gets highly increased. Services, one can easily analyze that in the Aneka 2.0 cloud services, the CPU Power Usage saves with 7.3% and Memory Usage save with 27.2%. Aneka service gives Better performance than any other existing tool for the same working scenarios. Result shows that proposed distributed mapper algorithm performance based on CPU and memory usage is better than other cloud service providers.



Graph Interpretation: The above graphs show a clear comparison that the proposed approach is better utilizing the processing units by implementing the distributed Mapper phenomenon's with Aneka cloud. Here the allocation strategies are controlled using scheduling algorithms so as configured previously but have distributed logics of VM handling. Thus it serves its purposes and gives us an effective resource optimization and management.





Graph Interpretation: The above graph compares the memory utilization between the traditional approach and the Distributed Mapper approach. While looking towards the graph (a) we have found that the virtual memory allocation with distributed mapper had increased elastically with proposed approach. It satisfies the cloud constraints and shows us that virtualization is effectively applied in case of memory for proposed system. While the physical memory is considered the consumptions was managed to get improved utilization using effective allocation and deallocation of memory to different processes according to their scheduling needs. Hence the physical memory occupancy gets reduced and we save the memory for other applications and tasks.

VIII. Benefits of the Work

(i) A flexible, scalable infrastructure management platform has been architected and a prototype implemented
(ii) Measurement of resource usage and end user activities lies hands of the cloud service provider.
(iii) Opaque cost structure due to highly flexible usage of cloud services.
(iv) Stable of cost structure
(v) The developed model is a resource allocation system that can avoid overload in the system effectively while minimizing the number of servers used.
(vi) It can also deals with the uneven utilization of a server for multi-dimensional resource constraints.
(vii) With the collected data it can be able to detect both computational as well as I/O bottlenecks.

IX. Conclusion

The above work develops a resource allocation system that can avoid overload in the system effectively while minimizing the number of servers and other devices used. The work had also introduced a concept to measure the uneven utilization of a server. By minimizing underutilizations and over-utilizations through our designed modules, improvement is expected in multi-dimensional resource constraints.

References

[1] Alessandro Ferreira Leite, Claude Tadonki, Christine Eisenbeis, Tain´a Raiol, Maria Emilia M. T. Walter and Alba Cristina Magalh˜aes Alves de Melo, "Excalibur: An Autonomic Cloud Architecture for Executing Parallel Applications", in ACM Publication, ISSN: 978-1-4503-2714-5/14/04, doi: 10.1145/2592784.2592786, Apr 2014

[2] Sushma K S, Vinay Kumar V , "Dynamic Resource Allocation for Efficient Parallel Data Processing Using RMI Protocol", in International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2, Issue-5, June 2013

[3] M.S.B.Pridviraju & K.Rekha Devi, "Exploiting Dynamic Resource Allocation for Query Processing in the Cloud Computing", in International Journal of Computer Science and Information Technologies (IJCSIT), ISSN: 5206 – 5209, Vol. 3 (5) , 2012,

[4] K.Krishna Jyothi , "Parallel Data Processing for Effective Dynamic Resource Allocation in the Cloud", in International Journal of Computer Applications (0975 – 8887) Volume 70– No.22, May 2013

[5] Yagız Onat Yazır, Chris Matthews, Roozbeh Farahbod, Stephen Neville, Adel Guitouni, Sudhakar Ganti and Yvonne Coady,"Dynamic Resource Allocation in Computing Clouds using Distributed Multiple Criteria Decision Analysis".

[6] Ian Foster and Carl Kesselman, "Globus:A Metacomputing Infrastructure Toolkit", yInformation Sciences Institute, University of Southern California.

[7] James Frey, Todd Tannenbaum, Miron Livny, Ian Foster and Steven Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", University of Wisconsin Argonne National Laboratory,Madison.

[8] Ewa Deelmana, Gurmeet Singha, Mei-Hui Sua, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems", Scientific Programming, IOS Press, ISSN:1058-9244/05, 2005

[9] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin and Anthony Liguori, "kvm: the Linux Virtual Machine Monitor"

[10] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell and Dennis Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", in ACM, doi: 978-1-59593-636-3/07/0003, 2007

[11] Yuan Yu, Michael Isard, Dennis Fetterly, Mihai Budiu, Úlfar Erlingsson, Pradeep Kumar Gunda and Jon Currey, "DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language", in 8th USENIX Symposium on Operating Systems Design and Implementation

[12] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar and Andrew Goldberg, "Quincy: Fair Scheduling for Distributed Computing Clusters", in Microsoft Research, Silicon Valley — Mountain View, CA, USA

[13] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao and D. Stott Parker, "Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters", in ACM, doi: 978-1-59593-686-8/07/0006, 2007

[14] Chaiken, R., Jenkins, B., Larson, P., Ramsey, B., Shakib, D., Weaver, S., and Zhou, "SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets", in PVLDB, 2010

[15] Ioan Raicu1, Ian T. Foster and Yong Zhao, "Many-Task Computing for Grids and Supercomputers", in IEEE, ISSN: 978-1-4244-2872-4/08, 2008

[16] Daniel Warneke and Odej Kao, "Nephele: Efficient Parallel Data Processing in the Cloud", in ACM, ISSN: 978-1-60558-714-1/09/11, 2009

[17] Dominic Battré, Stephan Ewen and Fabian Hueske, "Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing", ACM, ISSN: 978-1-4503-0036-0/10/06, 2010

[18] Daniel Warneke and Odej Kao, "Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud", in IEEE Transaction on Parallel & Distributed Systems, ISSN: 1045-9219/11, doi: 10.1109/TPDS.2011.65,2011