ISSN:0975-9646

Optimized Resource Management Decision System (ORM-DS) for Distributed Infrastructure Management in Cloud Computing

Nandini Kranti, Anand Singh Rajavat

Department of Computer Science & Engineering
Shree Vaishnav Institute Of Technology and Science, Indore (M.P), India

Abstract: **Cloud computing is a business solution based environments which supports scalable computing on demand for the end user. Here the users demanded resources can be provided with the help of cloud models. Mainly the models deal with infrastructure, platform and software with distributed and parallel processing plays the multiplexer role. The system primarily uses virtualization technology to cope with the computing needs. By using virtualization dynamic resource requirements can be handled with service optimizations. Sometimes this dynamic resource 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 analysed 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 suggests the improved scheduling and job execution environments for traditional resource handling approach using ORM-DS (optimized Resource Management Decision System). It also manages and monitors the dynamic allocation and deallocation. The proposed approach is having 7 step process model starts with client request and ends with the gird based resources to the applications as per the requirements. It is based on hierarchical structure of their utilizations with a heuristics support for decision making. In future the trace driven simulations and experimental results demonstrates the good performance.**

Index Terms- **:** Cloud Computing, Grid Computing, Resource Handling, ORM-DS (optimized Resource Management-Decision System);

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 [1]. 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. Now as the distributed model and processing is getting popularity is the market, working on cloud is mandatory. Resource management is very important and complex problems in cloud computing environment. It becomes more complex when resources are distributed geographically with heterogeneous environment and are dynamic in nature [2].

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) [3]. 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.

Resource Management with Distributed and Parallel Processing

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 [4].

Resource management is a complex task involving security, fault tolerance along with scheduling. It is the manner in which resources are allocation, assigned, authenticated, authorized, assured, accounted, and audited. Resources include traditional resources like compute cycles, network bandwidth, space or a storage system and also services like data transfer, simulation etc. A RMS is also responsible for naming the resources in the system, monitoring and reporting the job, resource status and accounting for resource usage [5]. The RMS interacts with the security system to validate user requests, the information service to obtain information about resource availability, the local system to schedule jobs on the local resource management system.

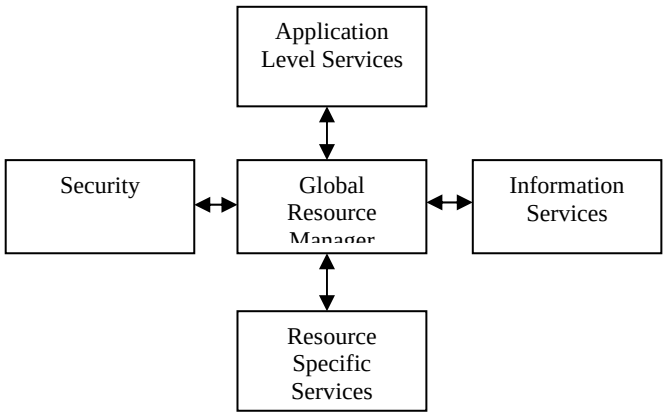

FIGURE 1: CLOUD BASED GLOBAL RESOURCE MANAGEMENT TASK

General Issues in Resource Management for Cloud

(i) Exploitation of underutilized resources
(ii) Parallel CPU Capacity
(iii) Virtual resources and virtual organizations for collaboration
(iv) Access to additional resources
(v) Resource Balancing
(vi) Reliability

Aim of this work is to provide a 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 are some motivational factors behind the Cloud deployment which are outlined here. These factors are one of the driving forces for effective resource management. 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. LITERATURE SURVEY

During the last few decades there are various approaches developed and analysed 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:

In the paper [6] the distributed resources' handling is measured for high-performance applications. These applications use high-speed networks to integrate supercomputers, large databases, archival storage devices, advanced visualization devices, and/or scientific instruments to form networked virtual supercomputers or meta-computers. The Globus system is intended to achieve a vertically integrated treatment of application, middleware, and network. A low-level toolkit provides basic mechanisms such as communication, authentication, network information, and data access. These mechanisms are used to construct various higher-level meta-computing services, such as parallel programming tools and schedulers. The goal with the paper is to build an Adaptive Wide Area Resource Environment (AWARE), an integrated set of higher-level services that enable applications to adapt to heterogeneous and dynamically changing meta-computing environments.

The paper [7] suggested a Condor-G tool which is enhanced form with certain modifications in Globus tool to control the sudden growth in computing and storage resources. It allows users to harness multi-domain resources as if they all belong to one personal domain. Mainly it deals with job management, resource selection, security, and fault tolerance. It serves three requirements for implementing the Clouds:

(i) They want to be able to discover, acquire, and reliably manage computational resources dynamically, in the course of their everyday activities.
(ii) They do not want to be bothered with the location of these resources, the mechanisms that are required to use them, with keeping track of the status of computational tasks operating on these resources, or with reacting to failure.
(iii) They do care about how long their tasks are likely to run and how much these tasks will cost.

Thus the Condor-G system leverages the significant advances that have been achieved in recent years in two distinct areas: (1) security, resource discovery, and resource access in multi-domain environments, as supported within the Globus Toolkit, and (2) management of computation and harnessing of resources within a single administrative domain, specifically within the Condor system. The user defines the tasks to be executed; Condor-G handles all aspects of discovering and acquiring appropriate resources, regardless of their location; initiating, monitoring, and managing execution on those resources; detecting and responding to failure; and notifying the user of termination. The result is a powerful tool for managing a variety of parallel computations in Cloud environments.

The paper [8] describes yet another concept Pegasus (Planning for Execution in Clouds) to map complex scientific workflows onto distributed resources. Pegasus enables users to represent the workflows at an abstract level without needing to worry about the particulars of the target execution systems. Since no single system can optimize a

wide variety of workflows and environments. It allows the users to customize various aspects of the system. In order to collect and organize information about multiple sites, it uses the indexing capabilities of MDS and RLS (the Replication Location Index – RLI). This collective information is utilized by Pegasus in the resource and replica selection decisions. In order to use Pegasus in such an environment, a resource, which could be a user's desktop, needs to be setup to provide Pegasus itself, DAGMan and Condor- G. They provide the workflow execution engine and the capability to remotely submit jobs (hosts) to a variety of Globus-based resources. Some other functionality implemented to achieve the desired goal are: Submit Host, Transformation Catalog (TC), Pool Configuration file and MDS.

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 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-flavoured 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 behaviour 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.

Summary:

(i) A policy issue remains as how to decide the mapping adaptively so that the resource demands of VMs are met while the number of physical machines used is minimized.
(ii) No control over the business assets (data!): The main assets in every company are its data files with valuable customer information.
(iii) Risk of data loss due to improper backups or system failure in the virtualized environment
(iv) High cost and loss of control

III. PROBLEM DEFINITION

Cloud provides a self configured computing with massive data supports by distributed and parallel processing**.** For applying the things practically the distributed concepts are studied thoroughly. It mainly includes grid and cloud based technologies where the resource and their load optimizations depends on various factors and needs to be managed simultaneously. Among them most useful is resource handling and allocations due to its heterogeneous executions. 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 plays a vital role in improvement of parallel processing is 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. As part of the resource allocation problem one may also need to consider parallel applications and their special needs such as how many processors and what type of network interconnects are needed to obtain good performance. There are some of the identified area of work is detected as the problem is:

Scenario: Existing approaches does not deals with the dynamic information mapping. Due to that the execution environment lacks with complete information availability for controlling the dynamic resource allocation and

deallocation. Information required for complete analysis of resource utilizations are:

(i) Load
(ii) Job Queue Length
(iii) Job Submission Servers
(iv) Data Transfer Server
(v) Constrained Scheduling Information's

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.

IV. WORK EMBODIED

The cloud aims towards achieving the high performance computing with respect to grid based resource utilizations and optimizations. Here the virtualization technology allocates data centre 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 by the work following goals are pivoted:

- Overload avoidance: the capacity should be sufficient to satisfy the resource needs of all VMs running on it. Otherwise, it gets overloaded and can lead to degraded performance of its VMs.
- Resource Handling and Scheduling: By this an efforts had been made for improving the resource usages and optimizations through effective mapping of distributed processing in VMs.
- Hidden Mapping: Mapping is largely hidden from the cloud users. Users with the systems service do not know where their VM instances runs. It's up to the cloud provider to make sure the underlying physical machines have sufficient resources to meet their needs.

V. PROPOSED SOLUTION

Managing resource always leads towards improving the device dependencies and will raises the standard of the system. For third party elastic computing offered by cloud, optimizing the resources with their continuous monitoring is a necessary task. Even though, the simple decisions sometimes shows high complexity for fine grained access control of data. 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 an novel optimized resource management decision system (ORM-DS).

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 with GIS.

Description of Model

The proposed model or ORM-DS is a seven step model involves request, discovery, information processing, task and job management, resource cluster management and the complete decision for effective analysis. The step wise operations are given as:

(i) Client or the user makes the initial request to the system. It means the application has requested for certain resources.

(ii) ORM module and decision making system gets this request and performs resource scheduling through a scheduler. As the requirements may vary with time hence the scheduler is of dynamic nature.

(iii) The dynamic scheduler later on forwards the request to resource discovery module. This module starts its operations with authorization filtering which involves the verification of the user submitting job. This determines the access of user on the 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. The next step after authorization is the specification of the requirements given by the user. This might include some static information such as operating system or much dynamic information such as memory. 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 minimum requirement criteria of user application. At the end of this step the scheduler will have the list of the resources for detail investigation and discovery.

(iv) Next phase includes collection of all this information with respect to some parameters of queuing request, load and resource availability.

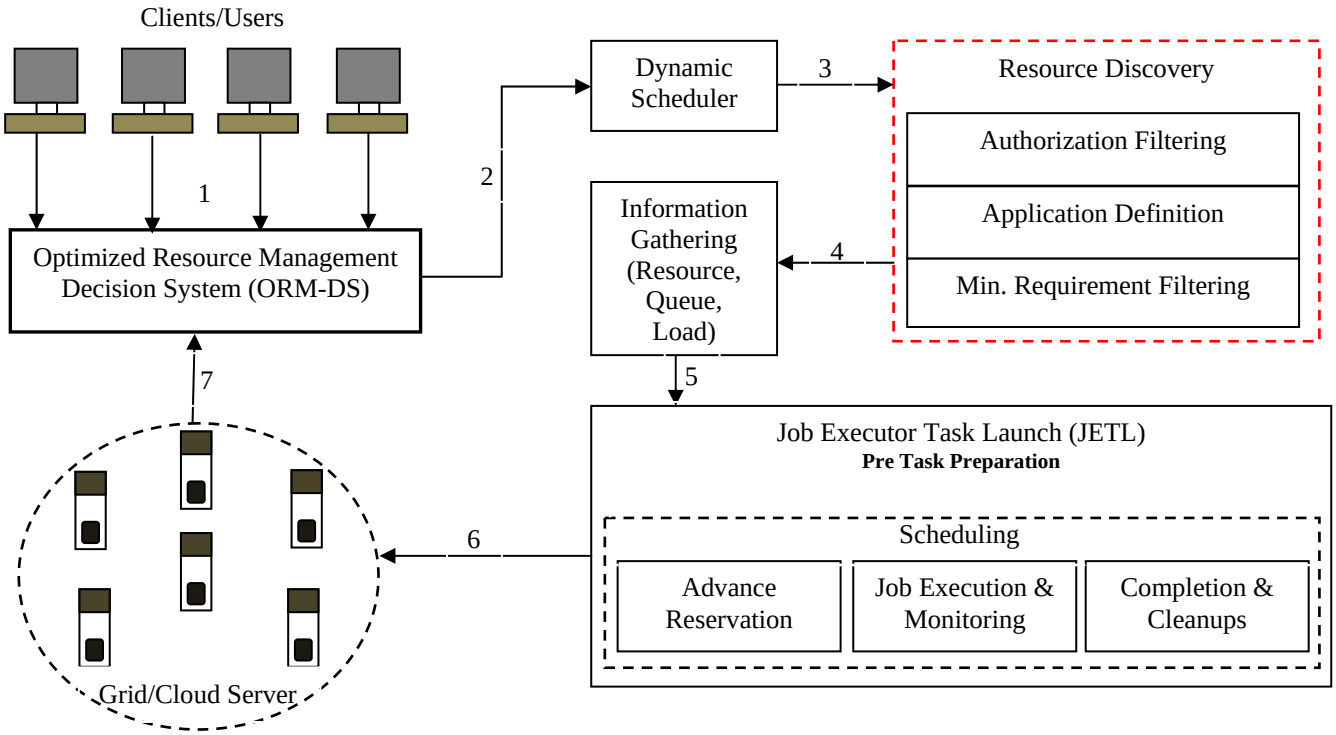

Figure 2: Seven Step Process for Optimized Resource Management Decision System (ORM-DS)

(v) This collective information is later on passed to job executor task launch (JETL) engine. It does the pre task preparation required for dynamic resource handling and making their job in a scheduled manner. 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. Last is the temporary file cleaning for clearing all the intermediate record.

(vi) The above identified results are transformed to the physical resource grids for service the applications requests.

(vii) Finally an object containing all the resource, their execution environment and their scheduled instruction is return back to the ORM- decision system for service the users demands.

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. So by observing the above proposed methodology on initial test result factor's we can easily enhance the capabilities of Cloud through the ORM-DS approach.

VI. Benefits and Applications

(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.

VII. Conclusion

As of now the cloud has evolved exponentially which requires lots of resources to satisfy the scalable demands of users and their heterogeneity. The above work develops an ORM-DS 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. Resource computation, their management and scheduling requires real time computation and monitoring. At the last the successful development of suggested model will leads towards definitive improvement in traditional resource management.

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