**HIGH PERFORMANCE COMPUTING PROJECT**

**CSE4014**

**SLOT: D2+TD2**

**CLASS NUMBER: VL2019201005663**

**REVIEW 1**

**REPORT SUBMISSION**

**LOAD BALANCING IN CLOUD COMPUTING ENVIRONMENT USING IMPROVED WEIGHTED ROUND ROBIN ALGORITHM FOR NON-PREEMPTIVE DEPENDENT TASKS**

**By**

**17BIT0198 PRATEEK AGARWAL**

**17BCE2210 SIMRAN KOUL**

**17BCI0013 ARJUN BHARDWAJ**

**ABSTRACT:**

Cloud computing uses the concepts of scheduling and load balancing to migrate tasks to underutilized VMs for effectively sharing the resources. The scheduling of the non-preemptive tasks in the cloud computing environment is an irrecoverable restraint and hence it has to be assigned to the most appropriate VMs at the initial placement itself. Practically, the arrived jobs consist of multiple interdependent tasks and they may execute the independent tasks in multiple VMs or in the same VM’s multiple cores. Also, the jobs arrive during the run time of the server in varying random intervals under various load conditions. The participating heterogeneous resources are managed by allocating the tasks to appropriate resources by static or dynamic scheduling to make the cloud computing more efficient and thus it improves the user satisfaction. Objective of this work is to introduce and evaluate the proposed scheduling and load balancing algorithm by considering the capabilities of each virtual machine (VM), the task length of each requested job, and the interdependency of multiple tasks. Performance of the proposed algorithm is studied by comparing with the existing methods.

**PROBLEM STATEMENT:**

The Cloud computing has to assign the computational tasks to the most suitable virtual machines from the dynamic pool of the VMs by considering the requirements of each task and the load of the VMs. The requests from the clients are directed to any of the data centres in the cloud. Then again the same requests are directed by the data centre to the most suitable VMs based on the cloud management policies depending on the load of the individual VMs. The two most frequently used scheduling principles in a non-preemptive system are round robin and the weighted round robin (WRR) policies. The round robin policy does not consider the resource capabilities, priority, and length of the tasks. So, the higher priority and lengthy tasks end up with the higher response times. The weighted round robin considers the resource capabilities of the VMs and assigns higher number of tasks to the higher capacity VMs based on the weightage given to each of the VMs. But it failed to consider the length of the tasks to select the appropriate VM, whereas the proposed and implemented algorithm (improved WRR algorithm) additionally considers the length and priority of the tasks in addition and selects the appropriate VM to execute the tasks for the lower response times.

The objective is to optimize the performance of virtual machines using the combination of static and dynamic load balancing by identifying the length of the jobs, resource capabilities, interdependency of multiple tasks, effectively predicting the underutilized VMs, and avoiding the overload on any of the VMs. This additional parameter of “job length” consideration can help schedule the jobs into the right VMs at any moment and is able to deliver the response in a very minimum execution time. The effective scheduling on this algorithm will also minimize the overload on a VM and subsequently it will also minimize the task migrations.

The performance of the improved WRR algorithm was analysed and evaluations of the algorithm with respect to the existing round robin and weighted round robin algorithm were carried out. This work considers that the job contains multiple tasks and the tasks have interdependency between them. A job can use multiple VMs for its various tasks to complete its entire processing instruction. Also, the task can use the multiple processing elements of a single VM based on the configuration and availability.

**OBJECTIVE:**

Load balancing is the process of redistributing the general system work load among all nodes of the distributed system (network links, disk drivers, central processing units…) to improve both resource utilization and job response time while avoiding a situation where some nodes are overloaded while others are under loaded or idle. Load balancing is a vital and inseparable part of cloud computing and elastic scalability. In order to avoid system failure, load balancing is often used by controlling the input traffic and stop sending the workload to resources which become overloaded and non-responsive. This is an inherited feature from grid-based computing which has been transferred to cloud computing.

Reducing job response time while keeping acceptable delays

• Maintaining system stability

• Having fault tolerance ability (using load balancing for implementing failover)

• Improving the general system performance for achieving optimal resource utilization, maximum throughput and avoiding overload

• Improving and maintaining the availability in cloud systems

**REVIEW OF 5 PAPERS:**

**1.Title: Load-balancing algorithms in cloud computing**

Author:

1) Mohammadreza Mesbahi -Islamic Azad University Tehran Science and Research Branch

2) Amir Masoud Rahmani- Islamic Azad University Tehran Science and Research Branch

Date Of Publishing: March 2016

Publisher: International Journal of Modern Education and Computer Science ·

**Problem Statement:**

Cloud computing has proposed a new perspective for provisioning the large-scale computing resources by using virtualization technology and a pay-per-use cost model. Load balancing is taken into account as a vital part for parallel and distributed systems. It helps cloud computing systems by improving the general performance, better computing resources utilization, energy consumption management, enhancing the cloud services’ QoS, avoiding SLA violation and maintaining system stability through distribution, controlling and managing the system workloads. models, and four deployment models”. As the main goal of cloud computing we can mention the better use of distributed resources and applying them to achieve a higher throughput, performance and solving large scale computing problems. Generally speaking, based on the NIST definition of cloud computing we can say best effort for offering the on demand services based on the best use of available shared resources is one of the most important goals of this model. To achieve these kinds of goals, improving the general performance of system, maintain stability, availability and some other features for a cloud computing data centre, we need a mechanism which is called load balancing. Load balancing is one of the central issues and challenges in distributed systems like grid-based systems and cloud computing. It is still a new problem in the cloud computing that needs new architectures and algorithms to promote the traditional approaches. In cloud computing, scheduling and handling many jobs that their arrival pattern, type of service and other properties are so hard to predict cause of the dynamic on-demand network access feature of system, an efficient load balancing mechanism is so necessary to increase the Service Level Agreement (SLA), deliver a robust service and provide other essential system requirements. In addition load balancing is an important topic because it enables other important features such as scalability.

**Objectives:**

Load balancing is the process of redistributing the general system work load among all nodes of the distributed system (network links, disk drivers, central processing units…) to improve both resource utilization and job response time while avoiding a situation where some nodes are overloaded while others are under loaded or idle. Load balancing is a vital and inseparable part of cloud computing and elastic scalability. In order to avoid system failure, load balancing is often used by controlling the input traffic and stop sending the workload to resources which become overloaded and non-responsive. This is an inherited feature from grid-based computing which has been transferred to cloud computing.

Reducing job response time while keeping acceptable delays

* Maintaining system stability
* Having fault tolerance ability (using load balancing for implementing failover)
* Improving the general system performance for achieving optimal resource utilization, maximum throughput and avoiding overload
* Improving and maintaining the availability in cloud systems

Some considerations from the point of architectural which are pointed out are:

Design for providing scalability that could cover all different designing level such as CPU level, machine level, and network level or even could be at the application and data centre level

* The ability of controlling the client requests and transfer to the selected resources according to the load balancer policies
* Fault tolerance for applications
* Scalability of the request handling capacity in a self-organized way
* The ability of handling more complex and higher traffic.

**Methodologies:**

The cloud balancing solutions can be categorized into three categories

a) General Algorithm based solution

b) Artificial Intelligence based solution

c) Architectural based solution

There are many various kinds of load balancing mechanisms and approaches which most of the studies have been classified as two main categories static and dynamic. In static techniques, there are usually prior knowledge and some assumptions about the global status of the system such as job resource requirements, communication time, processing power of system nodes, memory and storage devices capacity and so on. A static approach is a kind of assignment from a set of tasks to a set of resources which can take either a deterministic or a probabilistic form. In addition, this approach is defined usually in the design or implementation of systems.

Dynamic load balancing techniques take into consideration the current state of systems that their decisions are based on. In this technique tasks can move dynamically from an overloaded node to an under-loaded one and this is the main advantage of dynamic load balancing algorithms which can change continuously according to the current state of the system. However designing and implementing a dynamic load balancing algorithm is much more complicated and harder than finding a static solution, but through dynamic mechanisms we can gain a higher performance and have more accurate and efficient solutions. Dynamic load balancing algorithms can be designed in two different ways: distributed and non-distributed. In distributed approaches e.g. , the load balancing process can be executed by all nodes in the system. In addition, in this approach all nodes can communicate with each other for achieving a global goal in the system which is called cooperative or every node can work independently for just achieving a local goal that is non-cooperative form. But, in a non-distributed scheme, the responsibility of balancing the system workload would not be performed by all system nodes. In centralized approach in non-distributed scheme; a single node only can execute the load balancing mechanism among all nodes. In semi-distributed form, the system will be divided into some partitions or clusters and a single node execute the load balancing process in each partition.

****

The dividing load balancing algorithms into two dynamic and static categories is based on the point that algorithms take into account the current state of system for making decision or not. But from another point of view that has mentioned in , load balancing techniques can be divided into two other general categories based on some other factors:

* Based on the way that the system load is distributed and resources is assigned to the tasks (Depending on the system workload)
* Based on the system topology and available information about resources

The first category is designed as:

* Centralized approach
* Distributed approach
* Mixed approach

As we discussed earlier, this classification is based on the fact that who in charge of the load distributing is. The second category is designed as:

* Static approach
* Dynamic approach
* Adaptive approach

Distributed, dynamic and adaptive load balancing mechanisms of these two categories are more suitable for large scale distributed systems such as cloud computing. Adaptive approach adapts the load distribution to the system status changes, by changing their parameters dynamically and even their algorithms.

For making decision based on the current workload of system, dynamic load balancers need to know some information about the state of system. Therefore, a dynamic load balancing mechanism requires some components for gathering and handling these types of information. Four main components of a dynamic load balancer are:

a) Information strategy component: A component of dynamic load balancer is called information strategy component which is in charge of collecting information about status of resources in a system. According to the whole information which is obtained from each processing node in a distributed system, the load balancing process will be able to work efficiently.

b) Triggering Strategy component: A component of dynamic load balancer which determines the appropriate time for starting a load balancing operation is called triggering strategy component. Depending on the type of trigger policy, resources are classified into two categories: sender and receiver. Therefore, load balancing approaches will be called sender-initiated and receiver-initiated respectively

c) Transfer strategy component: A component which is in charge of selecting a task for transferring to another resource is called transfer strategy component. Two general approaches that determine which task should be transferred are: last-received-task and all-current-tasks.

d) Location strategy component: It selects a destination resource and a computing node for transferring a task. There are many approaches for selecting a destination such as: Probing, Random and Negotiation.

**CLOUD LOAD BALANCER CHALLENGES AND CONSIDERATIONS:**

**Geographically Distributed Cloud Nodes**

A load balancing algorithm in cloud environment should take into account the geographically distribution of computing nodes for having an efficient performance. Some load balancing algorithms are designed for closely located nodes and therefore they have no assumption about factors such as communication delays, network bandwidth in LANS and WANs, distance between clients and computing nodes and so on.

**Virtual Machines Migration**

By using virtualization technique in cloud computing, a physical server can contain several virtual machines that work as independent computing nodes. A common way to unload an overloaded physical server is virtual machine migration among load balancing approaches.

**Heterogeneous Nodes and Self-Regulating**

In the past researches of load balancing, the researchers have made an assumption about homogenous nodes during a load balancing designing phase

**Storage and Replication Management**

Development of cloud technology has solved the problem of traditional data storage methods which require high cost of hardware and personnel management through a resource-integrated heterogeneous system that can provide the best storage, the optimal performance and the load balancing

**Load Balancer Complexity**

It was concluded that load balancing algorithms which are trying to collect every detailed piece of information about system do not have a better performance considerably in comparison to algorithms which use very little or no information in information strategy phase

**Energy Efficiency**

One of the important challenging and complicated issues in cloud computing systems is lowering the energy usage of data centres. Data centres have serious negative effects on both the environment and energy resources. Recent researches showed that large-scale data centres consumed about 1.3% of all electricity use of the world and about 2% of all electricity usage for the United States in 2010 . Using internet services, computing applications and data increasing so fast which cause to have more computing resources for processing. The benefits of adoption of using cloud computing services is the economy of scale and energy saving is a key point that allows a set of global resources will be supported by reduced providers.

**Cloud Elasticity and Load Balancer Scalability**

An important feature of cloud computing is elasticity . Based on this property in cloud environments that provide the ability of resource scale up and scale down for users quickly, we can say that a load balancing algorithm in distributed systems like cloud should take into consider the system’s changes in terms of size, topology,… and therefore it should be scalable, adoptable and flexible enough to allow such changes to be handled easily and also can work efficiently. In this condition, it will be able to balance the workload when computing resources and requests increase.

**General Algorithm-based Load Balancing Approaches in Cloud Computing**

In this category, a load balancing mechanism can be implemented or simulated based on proposed algorithm. These approaches usually propose a load balancing solution without taking into consideration special cloud architecture and represented in a general form. In other word, we can say some GAL-based load balancing mechanisms are some classical load balancing methods which are similar to the allocation and scheduling methods in operating system such as: Round-Robin, First Come First Service (FCFS), Minimum Execution Time (MET) and etc. In this section we will give a summary of the current proposed algorithms which are belongs to this category.

**Round-Robin:** One of the most known and the simplest algorithm for dispatching workloads to servers is round-robin that usually have good performance in systems with low workload.

**Weighted round-robin:** This algorithm which is derived from original round-robin has better performance compared to traditional round-robin because it assigns higher weights to servers with higher performance. Therefore, the more capable computing nodes will get more incoming workloads.

**Least-connection:** This algorithm counts the connections associated with each server dynamically and then based on that number it chooses the least count server and assigns new incoming workloads to the server with least connection.

**Weighted least-connection:** This algorithm counts the associated connections of server too, but associated new incoming workloads based on a factor that calculated by multiplying sever weight by its connection number.

Shortest expected delay: In this algorithm, the last response time for each server is considered and then the server with the least response time is selected as the next appropriate server.

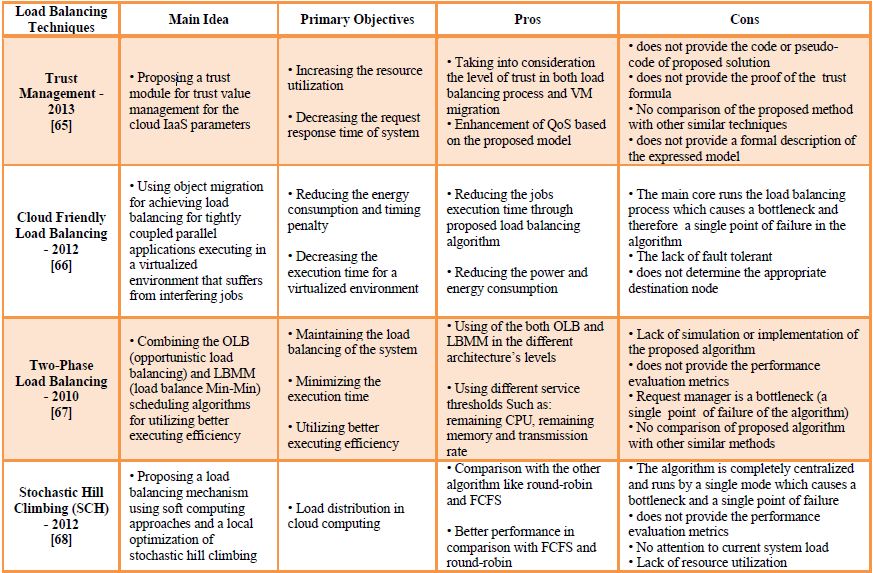
**Architectural-Based Load Balancing Approaches in Cloud Computing:**

Arch-based approaches are those mechanisms that focus on certain architecture and for achieving a load balancing, propose cloud architectures. In this category, algorithms usually not proposed explicitly and load balancing mechanism is represented through architecture components and the relations among them. Generally speaking, this solution usually is an Architectural-based solution and for catching proposed goals, a special cloud computing architecture should be taken into consideration.

**Artificial Intelligence-based Load Balancing Approaches in Cloud Computing**

AI-based load balancing mechanisms are the load balancing solutions that their main ideas are based on the Artificial Intelligence concepts. In the other word, AI-based approaches propose a solution for balancing the workloads in cloud computing environments through known artificial intelligence algorithms and methods by finding some similarity between them and cloud computing components and concepts. AI-based load balancing approaches can be proposed in an Arch-based form. Here we will introduce and consider some current AI-based load balancing mechanisms for cloud computing environments.

**Limitations:**



**2.Title: Load-balancing algorithms in cloud computing**

Author: Einollah Jafarnejad Ghomia, Amir Masoud Rahmania, , Nooruldeen Nasih Qader

Date Of Publishing: 08 April 2017

Publisher: Journal of Network and Computer Applications 88

**PROBLEMS AND OBJECTIVES:**

Cloud computing is a modern technology in the computer field to provide services to clients at any time. In a cloud computing system, resources are distributed all around the world for faster servicing to clients . The clients can easily access information via various devices such as laptops, cell phones, PDAs, and tablets. Cloud computing has faced many challenges, including security, efficient load balancing, resource scheduling, scaling, QoS management, data centre energy consumption, data locking and service availability, and performance monitoring. Load balancing is one of the main challenges and concerns in cloud environments; it is the process of assigning and reassigning the load among available resources in order to maximize throughput, while minimizing the cost and response time, improving performance and resource utilization as well as energy saving . Service Level Agreement (SLA) and user satisfaction could be provided by excellent load balancing techniques. Therefore, providing the efficient load-balancing algorithms and mechanisms is a key to the success of cloud computing environments. Several researches have been done in the field of load balancing and task scheduling in cloud environments.

**PROPOSED METHODOLOGY:**

Load balancing schedulers in Hadoop



Hadoop simplifies cluster programming as it takes care of load balancing, parallelization, task scheduling, and fault tolerance automatically. MapReduce, as the Google privacy strategy, hides the details of parallelization and distribution. Scheduling in Hadoop MapReduce is achieved at two levels: job level and task level. In job level scheduling, jobs are selected from a job queue (based on a scheduling strategy); in task-level scheduling, tasks of the job are scheduled. Scheduling strategies decide when and which machine a task is to be transferred for processing (load balancing). Hadoop uses First-In-First-Out (FIFO) strategy as its default scheduling, but it is pluggable for new scheduling algorithms. The scheduler is a pluggable module in Hadoop, and users can design their own dispatchers according to their actual application requirements. Researchers have developed several scheduling algorithms for the MapReduce environment that contribute to the load balancing. In addition, several load-balancing algorithms are developed as a plugin to standard MapReduce component of Hadoop. As mentioned before, any strategy used for an even load distribution among processing nodes is called load balancing. The main purpose of load balancing is to keep all processing nodes in use as much as possible, and not to leave any resources in an idle state while some other resources are being overloaded. Conceptually, a load-balancing algorithm implements a mapping function between the tasks and processing nodes. According to this definition of load balancing, scheduling algorithms do the task of load balancing. For this reason, we first surveyed and analyzed the load balancing schedulers in Hadoop.

a) FIFO scheduling. FIFO is the default scheduler in Hadoop that operates on a queue of jobs. In this scheduler, each job is divided into individual tasks that are assigned to a free slot for processing. A job dominates the whole cluster and only after finishing a job, the next job can be processed. Therefore, in this scheduler job wait time, especially for short jobs, increases and no jobs could be preempted. The default FIFO job scheduler in Hadoop assumes that the submitted jobs are executed sequentially under a homogeneous cluster. However, it is very common that MapReduce is being deployed in a heterogeneous environment; the computing and data resources are shared for multiple users and applications.

b) Fair scheduler. Facebook developed the fair scheduler. In this algorithm, jobs are entered into pools (multiple queues) and in the case of multiple users; one pool is assigned to each user. Fair scheduler distributes the available resources among the pools and tries to give each user a fair share of the cluster over time, with each pool allocated a minimum number of Map and Reduce slots. If there are free slots in an idle pool, they may be allocated to other pools, while extra capacity in a pool is shared among the jobs. In contrast to FIFO, the fair scheduler supports preemption, therefore if a pool has not received its fair share for a long time, then the scheduler will preempt tasks in pools running over capacity in order to give the slots to the pool running under capacity. In this way, a long batch job cannot block short jobs for a long time.

c) Capacity scheduler. Yahoo! developed the Capacity scheduler to guarantee a fair allocation of resources among a large number of cluster users. For this purpose, it uses queues with a configurable number of task slots (Map or Reduce). Available resources are assigned to queues according to the priorities. If there are free resources in some queues, they are allocated to other queues. Within a queue, the priority of jobs is determined based on the job arrival time, class of the job, and priority settings for users according to the Service Level Agreement (SLA). When a slot in a TaskTracker becomes free, the scheduler chooses a job with the longest waiting time from a queue with the lowest load. Therefore, the capacity scheduler enforces cluster sharing among users, rather than among jobs, as is the case in the fair scheduler.

d) Delay scheduler. The delay scheduler is an optimization of the fair scheduler, which eliminates the locality issues of the latter. The architecture of Hadoop. We consider a scenario in which a slot becomes free and we have to select a task of the job in front of a queue to process. It is possible that the data needed by this task does not exist on the node with a free slot. This is a locality problem. In the delay scheduler, this task is temporarily delayed until a slot in a node with the needed data becomes free. If the delayed time becomes long enough, to avoid starvation, the non-local task is allowed to schedule.

e) Longest Approximate Time to End (LATE). The LATE scheduler was developed to improve the job response time on Hadoop in heterogeneous environments. Some tasks may progress slowly due to CPU high load, race condition, temporary slowdown due to background processes, or slow background processes. These tasks are called speculative tasks. The LATE scheduler tries to find a slow task and execute an equivalent backup task on another node. This execution is called speculative execution. If the new copy of the task executes faster, the whole job performance will improve. The LATE Scheduler assigns priorities to slow or failed tasks for speculative execution and then selects the fastest nodes for that speculative execution. LATE scheduling improves the response time of Hadoop in heterogeneous environments.

f) Deadline constraint scheduler. The deadline constraint scheduler was designed to satisfy the user constraints. The goals of this scheduler are:

(1) to be able to give users immediate feedback on whether the job can be completed within the given deadline or not and proceed with the execution if the deadline

can be met. Otherwise, users have the option to resubmit with modified deadline requirements

(2) to maximize the number of jobs that can be run in a cluster while satisfying the time requirements of all jobs Experiment results showed that when deadlines for the job is different, then the scheduler assigns a different number of tasks to Tasktracker and makes sure that the specified deadline is met.



The load balancer receives users’ requests and runs load-balancing algorithms to distribute the requests among the Virtual Machines (VMs). The load balancer decides which VM should be assigned to the next request. The data centre controller is in charge of task management. Tasks are submitted to the load balancer, which performs load-balancing algorithm to assign tasks to a suitable VM. VM manager is in charge of VMs. Virtualization is a dominant technology in cloud computing. The main objective of virtualization is sharing expensive hardware among VMs. VM is a software implementation of a computer that operating systems and applications can run on. VMs process the requests of the users. Users are located all around the world and their requests are submitted randomly. Requests have to be assigned to VMs for processing. Therefore, the task assignment is a significant issue in cloud computing. If some VMs are overloaded while others are idle or have a little work to do, QoS will decrease. With the decreasing of QoS, users become unsatisfied and may leave the system and never return. A hypervisor or Virtual Machine Monitor (VMM) is used to create and manage the VMs. VMM provides four operations: multiplexing, suspension (storage), provision (resume), and life migration. These operations are necessary for load balancing. Load balancing has to consider two tasks: resource allocation and task scheduling. The result of these two tasks is the high availability of resources, energy saving, increasing the utilization of resources, reduction of cost of using resources, preserving the elasticity of cloud computing, and reduction of carbon emission.

**Load balancing metrics:**

Throughput: This metric is used to calculate the number of

processes completed per unit time.

• Response time: It measures the total time that the system takes to serve a submitted task.

• Makespan: This metric is used to calculate the maximum completion time or the time when the resources are allocated to a user.

• Scalability: It is the ability of an algorithm to perform uniform load balancing in the system according to the requirements upon increasing the number of nodes. The preferred algorithm is highly scalable.

• Fault tolerance: It determines the capability of the algorithm to perform load balancing in the event of some failures in some nodes or links.

• Migration time: The amount of time required to transfer a task from an overloaded node to an under-loaded one.

• Degree of imbalance: This metric measures the imbalance among VMs.

• Performance: It measures the system efficiency after performing a load-balancing algorithm.

• Energy consumption: It calculates the amount of energy consumed by all nodes. Load balancing helps to avoid overheating and therefore reducing energy usage by balancing the load across all the nodes.

• Carbon emission: It calculates the amount of carbon produced by all resources. Load balancing has a key role in minimizing this metric by moving loads from under loaded nodes and shutting them down.

**Taxonomy of load-balancing algorithms:**

Load monitoring: In this step, the load and the state of the resources are monitored

• Synchronization: In this step, the load and state information is exchanged.

• Rebalancing Criteria: It is necessary to calculate a new work distribution and then make load-balancing decisions based on this new calculation.

• Task Migration: In this step, the actual movement of the data occurs. When system decides to transfer a task or process, this step will run.

Dynamic algorithms are divided into two classes: distributed and non-distributed. In the distributed approach, all nodes execute the dynamic load-balancing algorithm in the system and the task of load balancing is shared among them. The interactions of the system nodes take two forms: cooperative and non-cooperative. In the cooperative form, the nodes work together to achieve a common objective, for example, to decrease the response time of all tasks. In the non-cooperative form, each node works independently to achieve a local goal, for example, to decrease the response time of a local task. Non-distributed algorithms are divided into two classes: centralized and semi-distributed. In the centralized form, a single node called the central node executes the load-balancing algorithms and it is completely responsible for load balancing. The other nodes interact with the central node. In the semi-distributed approach, nodes in the system are divided into clusters and each cluster is of centralized form. The central nodes of the clusters achieve load balancing of the system. Static algorithms are divided into two categories: optimal, and sub-optimal.In optimal algorithms, the data centre controller determines information about the tasks and resources and the load balancer can make an optimal allocation in a reasonable time. If the load balancer could not calculate an optimal decision for any reason, a sub-optimal allocation is calculated. In an approximate mechanism, the load-balancing algorithm terminates after finding a good solution, namely, it does not search the whole solution space. After that, the solution is evaluated by an objective function. In a heuristic manner, load-balancing algorithms make reasonable assumptions about tasks and resources. In this way, these algorithms make more adaptive decisions that are not limited by the assumptions. Algorithms in a sender-initiated strategy make decisions on arrival or creation of tasks, while algorithms in a receiver-initiated strategy make load-balancing decisions on the departure of finished tasks. In a symmetric strategy, either sender or receiver makes load-balancing decisions

**Challenges in cloud-based load balancing:**

a**) Virtual machine migration (time and security):**The service-on-demand nature of cloud computing implies that when there is a service request, the resources should be provided. Sometimes resources (often VMs) should be migrated from one physical server to another, possibly on a far location. Designers of load-balancing algorithms have to consider two issues in such cases: Time of migration that affects the performance and the probability of attacks (security issue).

b**) Spatially distributed nodes in a cloud:** Nodes in cloud computing are distributed geographically. The challenge in this case is that the load balancing algorithms should be designed so that they consider parameters such as the network bandwidth, communication speeds, the distances among nodes, and the distance between the client and resources.

c) **Single point of failure:** Some of the load-balancing algorithms are centralized. In such cases, if the node executing the algorithm (controller) fails, the whole system will crash because of that single point of failure. The challenge here is to design distributed or decentralized algorithms.

d)  **Algorithm complexity** The load-balancing algorithms should be simple in terms ofimplementation and operation. Complex algorithms have negative effects on the whole performance.

e)  **Emergence of small data centres in cloud computing** Small data centres are cheaper and consume less energy with respect to large data centres. Therefore, computing resources are distributed all around the world. The challenge here is to design load-balancing algorithms for an adequate response time.

f) **Energy management** Load-balancing algorithms should be designed to minimize theamount of energy consumption. Therefore, they should follow the energy-aware task scheduling methodology . Nowadays, the electricity used by Information Technology (IT) equipments a great concern. In 2005, the total energy consumed by IT equipment was 1% of total power usage in the world. Google data centres have consumed 260 million Watts of energy that is equal to 0.01% of the world’s energy. Research has shown that on an average, 30% of cloud servers exploit 10–15% of the resource capacity. Limited resource utilization increases the cost of cloud centre operations and power usage. Due to the tendency of organizations and users to use cloud services, in the future, the installations of cloud providers will expand and thus the energy usage in this industry will increase rapidly. This increase in energy usage not only increases the cost of energy but also increases carbon-emission. If the number of servers in data centres reaches a threshold, their power usage can be as much as that of a city. High energy consumption has become a major concern for industry and society

**MapReduce optimization for load balancing:**

In the standard Hadoop MapReduce, each data file is divided into fixed-sized blocks and each block has three replicas on three different DataNodes with two rules:

(1) no two copies are on the same DataNode.

(2) no two copies are on the same rack, provided that there are enough racks.

However, in replica placement, the current load of DataNodes is irrelevant. A built-in tool called the balancer executes repeatedly, the balancer moving data blocks from the overloaded Data Nodes to under-loaded ones.The balancer tool is used to balance an imbalanced cluster, but it would be better if we could keep the cluster as balanced as possible from scratch. Furthermore, using the balancer tool to load migration consumes a lot of system resources. Therefore, several researches have tried to provide load-balancing techniques in the Hadoop environment.

**Limitations:**

Only one job at a time uses cluster resources

• Low data locality when running a small job on cluster

• No considering priority or job size

• No-support multiuser execution

• Does not consider the actual workload of nodes for task scheduling

which may result imbalance

• It use the past information

• Is not suitable for environment with dynamic loading

• It does not ensure reliability

Capacity Yes High No Yes Static High No when job fails • Improves resources utilization

• It is flexible and efficient

• It can be used in large clusters

• Users should acquire large amount of information about system to

set a queue

• Killing speculative tasks wastes the work performed by them

• Does not ensure reliability

Deadline Yes High No Yes Dynamic High Yes

• Focus on increasing system

• Works better for large clusters utilization

• Homogeneous nodes are assumed

• Dynamic job deadline can affect the response time

• the job may not execute in the system due to its deadline constraints

**3.A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing**

**Keng-Mao Cho • Pang-Wei Tsai • Chun-Wei Tsai • Chu-Sing Yang**

Received: 12 August 2014/Accepted: 9 December 2014

The Natural Computing Applications Forum 2014

**Objective:**

Virtual machine (VM) scheduling with load balancing in cloud computing aims to assign VMs to suitable servers and balance the resource usage among all of the servers. In an infrastructure-as-a-service framework, there will be dynamic input requests, where the system is in charge of creating VMs without considering what types of tasks run on them. Therefore, scheduling that focuses only on ﬁxed task sets or that requires detailed task information is not suitable for this system. This paper combines ant colony optimization and particle swarm optimization to solve the VM scheduling problem, with the result being known as ant colony optimization with particle swarm (ACOPS). ACOPS uses historical information to predict the workload of new input requests to adapt to dynamic environments without additional task information. ACOPS also rejects requests that cannot be satisﬁed before scheduling to reduce the computing time of the scheduling procedure. Experimental results indicate that the proposed algorithm can keep the load balance in a dynamic environment and outperform other approaches.

**Methodologies:**

**Ant colony optimization with particle swarm:**

We present an efﬁcient scheduling algorithm that is based on ACO to solve VM scheduling with load balancing. Other than the heuristic function and ﬁtness function for the load balancing, an acceleration procedure, Pre-reject. To further reduce the computing time and to improve the scheduling result, this paper combines ACO and PSO . Via the fast convergence of PSO ,ACOPS can reduce a large amount of the computing time that is used in scheduling.

Every time requests come; do Pre-reject; if (There is any request needing to be scheduled.) then Initialization; while (Termination test) do Search; PSO operator; Evaluation; Pheromone update; end Algorithm 1: Pseudo-code of ant colony optimization with particle swarm.

Every time requests come;

**do** Pre-reject;

**if** (There is any request needing to be scheduled) **then**

Initialization;

**while** (Termination test) **do**

Search;

PSO operator;

Evaluation;

Pheromone update;

**end**

**end**

**Algorithm 1**: Pseudo-code of ant colony optimization with particle swarm.

**Summary of the paper:**

This paper proposed a ACOPS approach for VM scheduling with load balancing. ACOPS can serve customized VM requests for which three resources are considered for load balancing, namely the memory, cpu utilization and disk utilization. Furthermore, the server status information, which is the workload of historical requests, is used to predict the workload of the new input requests. To speed up the ACO procedure, we reject requests that are unsatisﬁed before scheduling. Through the procedure of pre-reject, the solution dimensions of the ACO reduce, and the computing time is, thus, saved effectively under a high-load situation. To further reduce the computing time and improve the scheduling result, this paper added a PSO operator into the ACO procedure. By consulting the personal best and global best solutions, the search results will converge quickly and achieve a higher quality. Different from past studies that used ﬁxed task sets and single scheduling to evaluate the performance of their approaches, this paper randomly produced dynamic request streams to simulate real environments via continuous scheduling to verify that ACOPS can maintain load balancing. Experimental results show that ACOPS is faster than traditional ACO and, in most cases, is superior in balancing the system compared to other approaches. Although ACOPS keeps loads more balanced than the other approaches, and the makespan is shorter in single scheduling, much work still remains to be completed in the future. Some areas for future study include: (1) continue improving the performance of the proposed algorithm; (2) considering VM scheduling to be a multi-objective problem and developing a multi-objective optimization algorithm to solve it; and (3) applying the proposed algorithm in a real cloud computing system.

**Limitations:**

There are two conditions for which ACOPS will ﬁnish its search and output the scheduling result. The ﬁrst condition is that the iteration reaches the maximum number of iterations (iterationmax), the value of which is calculated from the maximum ﬁtness evaluations. For example, there are ten ants in ACOPS, and the ﬁtness value will be calculated once for each ant in each iteration. The maximum iteration should be the maximum ﬁtness evaluations divided by ten. The second condition is that the global best solution does not change for a given time; in other words, the result is convergence. Accordingly, we set the given time to be Cthreshold= iterationmax \*0:1.

**4**.**Fog Computing Dynamic Load Balancing Mechanism Based on Graph Repartitioning**

**SONG Ningning1\*, GONG Chao2, AN Xingshuo2, ZHAN Qiang2**

1. School of Mechanical Engineering, University of Science and Technology Beijing, Beijing China
2. School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing China

**Objective:**

Because of cloud computing’s high degree of polymerization calculation mode, it can’t give full play to the resources of the edge device such as computing, storage, etc. Fog computing can improve the resource utilization efficiency of the edge device, and solve the problem about service computing of the delay-sensitive applications. This paper researches on the framework of the fog computing, and adopts Cloud Atomization Technology to turn physical nodes in different levels into virtual machine nodes. On this basis, this paper uses the graph partitioning theory to build the fog computing after Cloud Atomization can build the system network flexibly, and dynamic load balancing mechanism can effectively configure system resources as well as reducing the consumption of the node migration brought by system changes.

**Methodologies:**

There are several underlying algorithms used to facilitate fog/edge computing. In this section, we discuss four algorithms, namely (i) discovery - identifying edge resources within the network that can be used for distributed computation, (ii) benchmarking - capturing the performance of resources for decision-making to maximize the performance of deployments, (iii) load-balancing - distributing workloads across resources based on different criteria such as priorities, fairness, etc, and (iv) placement - identifying resources appropriate for deploying a workload.

**Summary:**

In this survey, we noted that technical challenges to managing the limited resources in fog/edge computing have been addressed to a high degree. However, a few challenges still remain to be made to improve resource management in terms of the capabilities and performance of fog/edge computing. We discuss some future research directions to address the remaining challenges. Fog/edge computing often employs resource-limited devices such as WiFi APs and set-top boxes that are not suitable for running heavy weight data processing tools such as Apache Spark and deep learning libraries. An alternative lightweight data processing tool such as Apache Quarks can be employed in resource-limited edge devices, but it lacks advanced data analytics functions. The imbalance between lightweight implementations and high performance needs to be addressed. In fog/edge computing, containers are widely used because they realize lightweight virtualization. However, efficient GPU resource management in containers has not been explored sufficiently, compared to research in virtual machines. In fog/edge devices, GPUs can be used for data analytics and to assist deep learning algorithms. For example, NVIDIA Jetson TX2 that is a single board computer that hosts an NVIDIA Pascal GPU can be used in the edge with containers. Efficiently managing GPU resources in containers is an open research challenge. Fog/edge computing has gained significant attention over the last few years as an alternative approach to the conventional centralized cloud computing model. It brings computing resources close to mobile and IoT devices to reduce communication latency and enable efficient use of the network bandwidth. In this survey paper, research on resource management techniques in fog/edge computing was studied to identify and classify the key contributions in the three areas of architectures, infrastructure, and algorithms.

**5. Load Balancing in Cloud Computing Using Dynamic Load Management Algorithm**

* Reena Pan war

M.Tech CSE Scholar, Department of Computer

Science and Engineering

Galgotias College of Engineering & Technology

Greater Noida, India

[reena.gcet@gmail.com](mailto:reena.gcet@gmail.com)

* Prof. Dr. Bhawna Mallick

Department of Computer Science and Engineering

Galgotias College of Engineering and Technology

Greater Noida, India

[bhawna.maUick@galgotiacollege.edu](mailto:bhawna.maUick@galgotiacollege.edu)

**Abstract:**

Nowadays, Cloud computing has become essential buzzword in the Information Technology and is a next stage in the evolution of Internet, The Load balancing problem of cloud computing is an important problem and critical component for adequate operations in cloud computing system and it can also prevent the rapid development of cloud computing. Many clients from all around the world are demanding the various services at rapid rate in the recent time. Although various load balancing algorithms have been designed that are efficient in request allocation by the selection of correct virtual machines. In the present paper, a dynamic load management algorithm has been proposed for distribution of the entire incoming request among the virtual machines effectively. Further, the performance is simulated by using Cloud Analyst simulator based on various parameters like data processing time and response time etc. and compared the result with previous designed algorithm VMAssign. Here results after simulation have demonstrated that the present algorithm has distributed the load uniformly among server through efficient usage of resources uniformly.

**Objective:**

Load balancing is a new technique that provides high resource time and resource utilization is effective by assigning the load among various servers, side by side it solves the problem of overutilization and underutilization of virtual machines. Load balancing resolve us the problem of critical overloading and focuses on maximum throughput, optimization of various resource utilization and minimize response time. Load balancing has two major tasks, one is resource allocation or provisioning of resources and other is scheduling

in distributed environment system. An Efficient provision of different resources and scheduling of resources as well as tasks will ensure:

• Resources are available easily.

• Resources are utilized under condition of low/high

load.

• Reduction in cost of using resources.

• Load balancing helps in increasing throughput to the

maximum level and minimum response time.

In cloud computing environment various load balancing algorithm have been facilitated and proposed such as Honeybee-based load balancing technique, Active Clustering, Random Sampling, Throttled Load Balancer, JIQ,WCAP, and Active Monitoring Load Balancer ,and the another one is CLBVM. Hemant S. Mahalle, Parag R. Kaveri and Vi nay Chavan proposed an "Active monitoring load balancer algorithm" which is responsible for maintaining the information related to VM and how many requests are allocated to each VM. When a request is come, it will assigned to the VM which is least loaded, if more than first one is selected. The result of is that Active VM load balancer is able to return id to Data Centre Controller and a request is then send by the data centre controller to VM identified by that id. Then Data Centre sends notification to Active VM Load Balancer for new allocation of request. Shridhar G. Domanal and G.Ram Mohana Reddy , proposed "Modified throttled algorithm" is optimized load balancing technique for distribution of coming jobs request consistently between the servers or virtual machines. The efficiency is evaluated using simulator Cloud Analyst and result is compared with previous algorithm Round Robin and throttled algorithm. Hence response time has been improved related to previous one. Shridhar G. Domanal and G. Ram Mohana Reddy have proposed an algorithm VM-Assign i.e. responsible for distribution of all the coming requests to all the virtual machines that are available efficiently. Here CloudSim Simulator has been used for result analysis and then compared with existing Active-VM load balance algorithm. Our proposed technique resolves the ineffective utilization of the VMs / resources when relates result to previous algorithm. B.Wickremasinghe ,R. N.Calheiros and Rajkumar Buyya states that Throttled Load Balancing Algorithm is fully based on the allocation of request to virtual machine . In this current right virtual machine is checked by load balancer when user request and it can able to approach that load dynamically well and performs the task that had been requested by user. The result is that an index table of available virtual machines has been maintained by the load well and their states i.e. Busy or Available. Firstly then a user requested to the active load balancer to select the applicable Virtual Machine to execute the assigned task.

**Methodology proposed:**

In present we have proposed an efficient algorithm Dynamic Load Management Algorithm has proposed where load is managed by the server by considering the present status of present VMs for request assignment sharply. Therefore,reduction in response time when compared to VMAssign Algorithm.

Algorithm: Dynamic Load Management Algorithm

**Summary:**

In present paper, a dynamic algorithm is proposed which will manage the load incoming by focusing on their present status at cloudlet for all free VMs to be used at request assignment and will take more requests that are dynamic in nature. The response time has been improved efficiently. The

main problem with the existing algorithm is that every time it considers all the virtual machines to check the availability of assigning new load. So it takes more time in request allocation which in turn leads to increased response time. The simulated algorithm has response time better when relates to Optimal VM-Load balancer because we have a dynamic set of available virtual machine and unlike the previous one we don't considers an overloaded machine again and again for scheduling. It leads to the better response time. Hence our proposed algorithm distributes the load nearly efficiently

among VMs with improved time in comparison to the previous algorithms and solves all the issues of ineffective usage of the present VMs. The experimental results has shown that this algorithm have minimum response time and proper resource utilization by using Cloud Analyst tool and checked its performance on various different load distributions. Here Simulation results indicated that the proposed Dynamic Load Management algorithm outperforms the existing virtual machine load balancing algorithms. In future if both dynamic and static load is to be mixed then this algorithm can be improved sufficiently and also by incorporating the paradigms of parallel and high performance computing response time and utilization of VMs may be further optimized.

**Limitation:**

The algorithm cannot be performed without these components:

a) GUI Packages: It is mainly responsible for the graphical user Interface.

b) User Base[J9}: It models a different clients who are treated as individual entity in the simulation and generates traffic for the simulation.

c) Internet- By introducing delay for transmission and data transfer, it can models routing scheme for

Internet traffic .

d) Data Centre Controller: It generally controls Data centre activities.

e) Vm Load Balancer: A Vm Load Balancer is used by Data centre controller for determination of which particular VM is to be used for assignment to which data centre and models load balancing policies.

f) Simulation: Simulation accepts requests and then execute request.

g) Cloud App Service Broker: This component broker handles the traffic routing between data centres and user base.

**OUR METHODOLOGIES:**

The two most frequently used scheduling principles in a non-preemptive system are round Robin and weighted round robin policies. Improved weighted round robin is the proposed algorithm. Existing algorithms are implemented for comparative analysis.

1. Round Robin Algorithm

The round robin algorithm allocates task to the next VM in the queue irrespective of the load on that VM. The Round Robin policy does not consider the resource capabilities, priority, and the length of the tasks. So, the higher priority and the lengthy tasks end up with the higher response times.

2. Weighted Round Robin Algorithm

The weighted round robin considers the resource capabilities of the VMs and assigns higher number of tasks to the higher capacity VMs based on the weightage given to each of the VMs. But it failed to consider the length of the tasks to select the appropriate VM.

3. Improved Weighted Round Robin Algorithm

The proposed improved weighted round robin algorithm is the most optimal algorithm and it allocates the jobs to the most suitable VMs based on the VM’s information like its processing capacity, load on the VMs, and length of the arrived tasks with its priority. The static scheduling of this algorithm uses the processing capacity of the VMs, the number of incoming tasks, and the length of each task to decide the allocation on the appropriate VM.

The dynamic scheduling (at run time) of this algorithm additionally uses the load on each of the VMs along with the information mentioned above to decide the allocation of the task to the appropriate VM. There is a probability at run time that, in some of the cases, the task may take longer execution time than the initial calculation due to the execution of more number of cycles (like a loop) on the same instructions based on the complicated run time data.

In such situations, the load balancer rescues the scheduling controller and rearranges the jobs according to the idle slot available in the other unutilized/underutilized VMs by moving a waiting job from the heavily loaded VMs. The load balancer identifies the unutilized/underutilized VMs through resource prober whenever a task is completed in any of the VMs. If there is no unutilized VM, then the load balancer will not take up any task migration among the VMs. If it finds any unutilized/underutilized VM, then it will migrate the task from the overloaded VM to the unutilized/underutilized VM. The load balancer analyses the resource’s (VM) load only on the completion of any of the tasks on any of the VMs. It never examines the resource’s (VM) load independently at any time to circumvent the overhead on the VMs. This will help in reducing the number of task migrations between the VMs and the number of resource probe executions in the VMs.