APPROACH TO IMPROVE PERFORMANCE ON DEMAND TASK SCHEDULING AND LOAD BALANCING IN CLOUD COMPUTING ENVIRONMENT

S. Chithra[1], Dr. J. Anitha[2]

[1]PG Scholar, Computer Science and Engineering,

[2]Associate Professor, Department of Information Technology,

Sri Ramakrishna Engineering College, Coimbatore, Tamilnadu, (India)

ABSTRACT

Cloud computing is emerging as a new paradigm for manipulating, configuring, and accessing large scale distributed computing applications over the network. Load balancing is one of the main challenges in cloud computing which is required to distribute the workload evenly across all the nodes. Properly dispatching tasks among CPU cores is crucial to reduce response time of jobs, which provides benefit for both systems. A hybrid scheme of task scheduling and load balancing named DeMS is proposed. DeMS consists of three algorithms, including On-Demand scheduling, Querying and Migrating Task (QMT) and Staged Task Migration (STM). The proposed On-Demand scheduling algorithm is used to decrease the communication overhead between a master and slaves. QTM is designed to keep the workload balanced. Data Shuffling is used to represent interactions between stages. The system performance can be increase by reducing the response time and also have high applicability, low latency, avoid overloaded.

Index Terms: Cloud Computing, Task Scheduling, Load Balancer, Load Balancing, Load Balancing Algorithm.

I. INTRODUCTION

Cloud computing is a computing paradigm, where a large pool of systems are connected in private or public networks, to provide dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly.

Cloud computing is a practical approach to experience direct cost benefits and it has the potential to transform a data centre from a capital-intensive set up to a variable priced environment.

The idea of cloud computing is based on a very fundamental principal of reusability of IT capabilities. The difference that cloud computing brings compared to traditional concepts of "grid computing", "distributed computing", "utility computing", or "autonomic computing" is to broaden horizons across organizational boundaries.

There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users. Following are the working models for cloud computing:

- Deployment Models
- Service Models

1.1. Deployment Models

Deployment models define the type of access to the cloud. Cloud can have any of the four types of access: Public, Private, Hybrid and Community.

1.1.1. Public Cloud

The Public Cloud allows systems and services to be easily accessible to the general public. Public cloud may be less secure because of its openness, e.g., e-mail.

1.1.2. Private Cloud

The Private Cloud allows systems and services to be accessible within an organization. It offers increased security because of its private nature.

1.1.3. Community Cloud

The Community Cloud allows systems and services to be accessible by group of organizations.

1.1.4. Hybrid Cloud

The Hybrid Cloud is mixture of public and private cloud. However, the critical activities are performed using private cloud while the non-critical activities are performed using public cloud

1.2. Service Models

Service Models are the reference models on which the Cloud Computing is based. These can be categorized into three basic service models as listed below:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

1.2.1. Infrastructure as a Service (IAAS)

IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc.

1.2.2. Platform As A Service (Paas)

PaaS provides the runtime environment for applications, development & deployment tools, etc.

1.2.3.Software As A Service (Saas)

SaaS model allows to use software applications as a service to end users

II. LOAD BALANCING

Load balancing in clouds is a technique that distributes the excess dynamic local workload evenly across all the nodes. It is used for achieving a better service provisioning and resource utilization ratio, hence improving the overall performance of the system Incoming tasks are coming from different location are received by the load balancer and then distributed to the data center, for the proper load distribution

The aim of load balancing is as follows:

- To increase the availability of services
- To increase the user satisfaction
- To maximize resource utilization
- To reduce the execution time and waiting time of task coming from different location.
- To improve the performance
- Maintain system stability
- Build fault tolerance system
- Accommodate future modification

III. RELATED WORKS

Gang Scheduling in Multi-Core Clusters Implementing Migrations (Papazachos ZC, Karatza HD)[10] A proper scheduling algorithm is essential for the efficient utilization of the available resources of complex distributed systems. The scheduling algorithm is responsible for allocating the available system resources to the existing jobs. The emergence of multi-core processors poses new demands on schedulers. The performance of gang scheduling algorithms for homogeneous and heterogeneous clusters which consist of multi-core processors. Furthermore, a migration schema is suggested which is suitable for scheduling gangs in multi-core clusters. A simulation model is used to provide results on the performance of the system. The main advantages of this the time taken for resources allocation was decreased and system performance is improved. And the limitation is increase the communication overhead between masters and slaves.

Delay Scheduling Achieving locality and Fairness in Cluster Scheduling(Olgac N, Ergenc AF,Sipahi R) [9] As organizations start to use data-intensive cluster computing systems like Hadoop and Dryad for more applications, there is a growing need to share clusters between users. However, there is a conflict between fairness in scheduling and data locality (placing tasks on nodes that contain their input data). We illustrate this problem through our experience designing a fair scheduler for a 600-node Hadoop cluster at Face book. To address the conflict between locality and fairness, we propose a simple algorithm called delay scheduling when the job that should be scheduled next according to fairness cannot launch a local task, it waits for a small amount of time, letting other jobs launch tasks instead. We find that delay scheduling achieves nearly optimal data locality in a variety of workloads and can increase throughput by up to 2x while preserving fairness. In addition, the simplicity of delay scheduling makes it applicable under a wide variety of scheduling policies beyond fair sharing. The main Advantages is Double throughput in an IO-heavy workload and improve response time. And disadvantages are High latency due to increase in waiting time and low applicability due to dependency.

The Power of Two Choices in Randomized Load Balancing (Pratt B, Howbert JJ, Tasman NI, Nilsson)[11] The following natural model: Customers arrive as a Poisson stream of rate _n, _ < 1, at a collection of n servers. Each customer chooses some constant d servers independently and uniformly at random from the n servers and waits for service at the one with the fewest customers. Customers are served according to the first-in first-out (FIFO) protocol and the service time for a customer is exponentially distributed with mean 1.This problem the supermarket model. The Advantages of this was easier to analyze because its behaviour is completely deterministic. And limitations are the supermarket model proves difficult to analyze dependencies task and reduce the system performance.

Randomized Load Balancing with General Service Time Distributions (Husain MF, Doshi P, Khan L, Thuraisingham B) [5] A modularized program for treating randomized load balancing problems with general service time distributions and service disciplines. The program relies on an *ansatz* which asserts that any finite set of queues in a randomized load balancing scheme becomes independent as $n \to \infty$. This allows one to derive queue size distributions and other performance measures of interest. To establish the *ansatz* when the service discipline is FIFO and the service time distribution has a decreasing hazard rate (this includes heavy-tailed service times). Assuming the *ansatz*, we also obtain the following results: (i) as $n \to \infty$, the process of job arrivals at any fixed queue tends to a Poisson process whose rate depends on the size of the queue, (ii) when the service discipline at each server is processor sharing or LIFO with pre-emptive resume, the distribution of the number of jobs is insensitive to the service distribution, and (iii) the tail behaviour of the queue-size distribution in terms of the service distribution for the FIFO service discipline. The main advantage of this System performance is well tuned which reduce the collisions and waiting time. And Limitations is Load balancing is a problem to distribute tasks among multiple resources and Increase overhead communication.

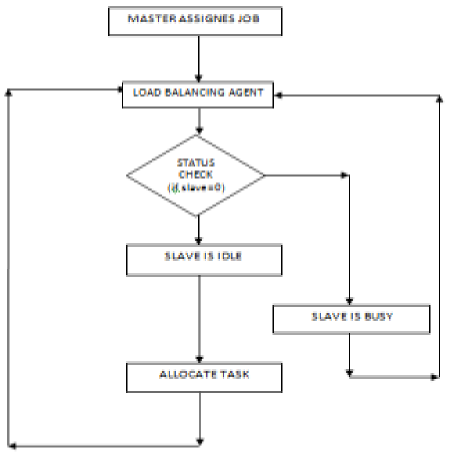
IV. PROPOSED SYSTEM

The proposed hybrid scheduling scheme called DeMS was introduced and On-Demand scheduling method was used to reduce communication over head between masters and slaves. A task migration algorithm is designed to keep the workload balanced. A data shuffling mechanism is employed for dependent tasks.

- For independent tasks, an On-Demand scheduling method is proposed. The probe-based state collection mechanism is renounced to avoid additional communication cost. In our approach, the master keeps a light-weight meta data of each slave. When a slave has enough resources to run a new task, it sends a short frame to the master to indicate its idle state.
- A task migration algorithm is proposed in DeMS to guarantee load balance for a cluster. The master maintains a timer for each slave .When a slave reports an idle state; the master will launch a task migration procedure for another slave which has worked for a long time. Task migration is a big burden for any parallel system. We deal with this issue by enabling the master to maintain a list of the last tasks dispatched to each slave. Thus, a slave need not send the task to be migrated back to the master for rescheduling
- For dependent tasks, dependencies between tasks are abstracted as data shuffling processes. A job is divided into several stages according to the execution order, and proposed Staged Task Migration (STM) algorithm, in which a task migration loop is designed to guarantee the most balanced workload for each stage

4.1. Flow chart for the proposed framework distributing the task

Distributing the task among the slaves using On-Demand scheduling algorithm. On-Demand scheduling method.



2.2. Distributing the task using On Demand scheduling

Each slave has an observer to monitor its task queue. When the observer detects that the slave has enough resources for a new task, it will send an On-Demand request to the master that keeps a light weighted metadata of the slave. In the On-Demand scheduling, we define metadata for a slave as S = {id}; All slaves are indexed by the slave id. The field of state is a Boolean value. When state {1}, the slave is in an idle state. After dispatching a new task to a slave, the master will set its state as 0 and the On-Demand request can reset the state as 1

2.3. Task Migration

In order to dispatch at ask to the slave with the least load, traditional approaches utilize the length of task queue to represent the queuing load of slaves.

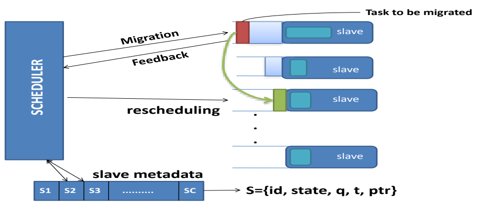


Fig 4.2. Migrating process

That predicting the waiting time of a queued task based solely on queue length is typically in effective. But the total processing time of the queued tasks is unknown to either the slave or the master. Though the On- Demand mechanism can ensure the scheduler to assign the task to an idle slave, it is not necessarily the one with the least load. For the On-Demand scheduling method, its dispatch may lead to long response time of task in some cases. That when a cluster is running with light workload, the scheduler may make the wrong decision because it does know the processing time of the current running task on the slave. On the contrary, if the cluster is running with heavy workload, the master may make similar mistake since the queue length cannot represent the real waiting time directly.

2.4. Data Shuffling

Random Two Choices and On-Demand scheduling methods are designed under the assumption that each task runs independently. This assumption is too strong for real parallel applications. The tasks may be dependent. If a task has predecessors, it cannot start to run until its predecessor shave finished

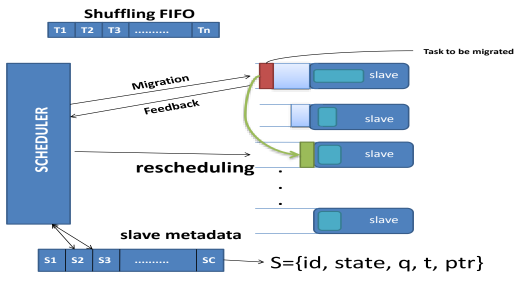


Fig 4.3 Staged Task Migrating.

Based on other distributed computing frameworks (e.g., Spark), we divide a parallel job in to several stages according to the dependencies among tasks. The tasks in one stage run independently, while the tasks in different stages must be executed serially. We term this task scheduling context Data shuffling because inter-communications between stages can be regarded as a process of data transmission among cores. In Data shuffling, the response time of a job is the sum of response time of each stage.

V. PERFORMANCE EVALUATIONS

The experimental tests were conducted among various slaves with varying memory capacity. The result was produced based on migration of tasks between the slaves using On Demand Scheduling, Query and Migrating Task, Stage Task Migration.

5.1. On-Demand vs. Random Two Choices

In order to evaluate the performance of the On-Demand scheduling algorithm in DeMS. The workload of our cluster is related with the number of tasks in a job and the dispatching period of jobs. Response time of a job is determined by the response time of the slowest task.

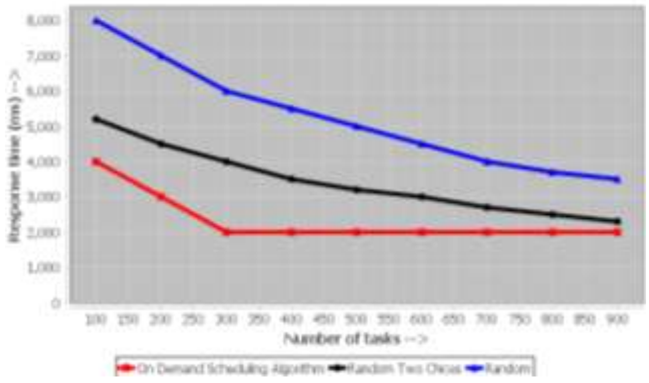


Fig 5.1 On-demand vs. Random Two Choices

With the constant task processing time, On-Demand acts better than it does with the random task processing time.

PARAMETERS	*BEFORE MIGRATION*	*AFTER MIGRATION*
Number of overloaded slaves	*2*	*0*

Table 5.1 Comparison table for load balancing

5.2. Query and Migrating Task

When a cluster works with a heavy work load, each slave has more than one task to be handled and tasks are queued for processing. In this case, the On-Demand algorithm loses its capacity to dispatch tasks on idle slaves. The aim of the task migration process is to balance the work load on slaves in the cluster and this mechanism is designed to reduce the response time of a job. The QMT algorithm on the cluster and run utilize parallel jobs, each of which consists of many tasks, to simulate a heavy work load .The threshold of λ in QMT is set as an empirical constant which is twice of the meant ask processing time.

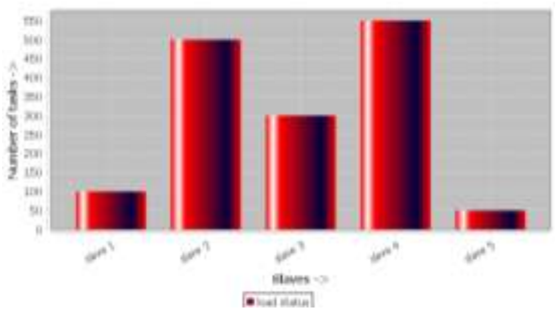


Fig 5.2 Before Migration

Task will be migrated using the formula,

S(t)=overloaded if s(t)>ƛ

Where S(t)=slaves with tasks

ƛ=threshold value

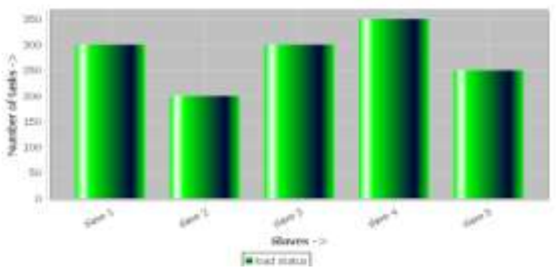


Fig 5.3 After Migration

Figure 5.3 indicates that with the constant task processing time, STM acts better than it does with the random task processing time.

In the random task processing time is used. The mean response time of QMT is almost the same as On-Demand when the dispatching period is longer than 600 ms. Then, the workload of the cluster is increased and QMT acts better than On-Demand.

5.3. Dependent Tasks Scheduling

For a job consisting of dependent tasks, the tasks in the next stage must be suspended until their precursors are finished. We propose an algorithm named STM to handle this kind of tasks. In STM, each stage is regarded as a sub job and a staged task migration process is applied to each stage to seek the least stage response time. In this graph divide a job consisting of 4000 tasks in to two equal groups and assume that the two groups must be executed serially

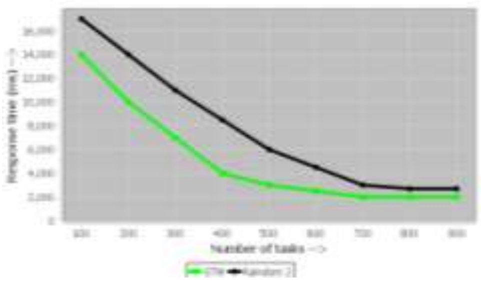


Fig 5.4 Staged Task Migration

VI. CONCLUSION

A hybrid task scheduling and load balancing scheme called DeMS, which consists of an On-Demand scheduling method, a QMT algorithm for task migration and an STM algorithm for dispatching tasks. On-Demand scheduling method can significantly reduce the response time of parallel jobs. QMT and STM are effective for independent and dependent tasks scheduling. The future work of the proposed system is to two fold integrating DeMS with real scheduling frameworks, such as mesosand YARN (2) The network delay and task migration time to design more efficient scheduling methods.

REFERENCES

[1] Bramson M, LuY, Prabhakar B. Randomized "Load balancing with general service time distributions." In: ACMSIGMETRICS performance evaluation review, ACM, New York, vol. 38, pp. 275–86, 2010.

[2] Dean J. "Achieving rapid response times in large online services." In: Berkeley AMP Lab cloud seminar, 2012.

[3] H.D. Karatza, "Performance analysis of gang scheduling in a parallel Proceedings of the 20th European Conference on Modelling and Simulation", Germany, pp. 699–704, 2006.

[4] Hindman B, Konwinski A, Zaharia M, Ghodsi A ,Joseph AD, KatzRH Mesos:"A platform for fine-grained resource sharing in the datacenter" In: NSDI, vol.11, pp. 295–308, 2011.

[5] Husain MF, Doshi P, Khan L, Thuraisingham B, "Storage and retrieval of larger rdf graph using hadoop and mapreduce". In: Cloud computing. Springer, Berlin, pp. 680–6, 2011.

[6] Mitzenmacher M. "The power of two choices in randomized load balancing", IEEE TransParallelDistribSyst, vol. 12, no.10, pp. 1094–104, 2011.

[7] Moschak is IA, Karatza HD " Evaluation of gang scheduling performance and cost in a cloud computing system". JSupercomput, vol. 59, no. 2, pp. 975–92, 2009.

[8] Muralidhar K, Sarathy R." Data shuffling-a new masking approach for numerical data". ManagSci, vol. 52, no.5, pp. 658–70, 2006.

[9] Olgac N, Ergenc AF,Sipahi R "Delay scheduling :a new concept for stabilization in multiple delay systems" JVibControl, vol. 11,no. 9, pp. 1159–72, 2005.s

[10] Papazachos ZC, Karatza HD. "Gang scheduling in multi-core clusters implementing migrations." FutureGenerComputSyst, vol. 27, no.8, pp. 1153–65, 2011

[11] Pratt B, Howbert JJ, Tasman NI, Nilsson EJ, Mr-tandem :parallelx! "Tandem using hadoop map reduce on amazon webservice". vol. 28, no.1, pp. 136–7, 2012.

[12] M, Franklin MJ, Shenker S, StoicaI, "Spark: cluster computing with working sets". In: Proceedings of the 2nd USENIX conference on hot topics in cloud computing, pp. 10, 2010