IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. IX (Mar-Apr. 2014), PP 115-118
www.iosrjournals.org

A System for Task Scheduling and Task Migration in Cloud Environment

Haimantee Mahato, Anjali Munjal, Shreya Chinchalikar.
Guided by Prof. S.C. Chaudhari
Department of Information Technology.
Padmabhooshan Vasantdada Patil Institute of Technology, University of Pune. India

***Abstract**: Large scale data processing is increasingly common in cloud computing systems in recent years. In these systems, files are split into many small blocks and all blocks are replicated over several servers. To process files efficiently, each job is divided into many tasks and each task is allocated to a server to deals with a file block. If the current server is busy, then it migrates the small block of task to another free server.*

Task Scheduling and task migration are the techniques for increasing flexibility and scalability in cloud computing. These techniques are used to perform different goals like balancing and load sharing, reducing response time and improvement of Quality of Service.

In this paper we expect the idealised system for task scheduling and migration in cloud environment in which an initial task will be just a chunk of programming code that will be converted into executable program.

***Keywords:** Cloud Computing, Task Scheduling, Task Migration, Point-cut, Joint-cut.*

I. Introduction

According to NIST ,"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Cloud computing provides users with a variety of storage and computing resources in the cloud computing environment via Internet, users put a lot of information and processor resources of terminal equipment such as personal computer, mobile phone together for work. Cloud computing is a large-scale distributed computing model, which depends on the economic size of the operator that is abstract, virtualized and dynamic. The main content of cloud computing is managed computing power, storage, platforms and services which assigned to the external user on demand through the Internet. Cloud computing is an emerging computation paradigm with the goal of freeing up users from the management of hardware, software, and data resources and shifting these burdens to cloud service providers. Clouds provide a large pool of resources, including high power computing platforms, data centers, storages, and software services. It also provides management to these resources such that users can access them ubiquitously and without incurring performance problems. Cloud computing provides a more abstract resources and services of these resources and services can be divided into three levels, namely the software as a service, platform as a service and facilities that service.

The general architecture of a cloud computing system [1, 2, 3, 4] is illustrated in Fig. 1. In this architecture, a file is split into fixed-size *blocks* which are stored on servers. For fault tolerance, all blocks are replicated and spread over the cluster. To process the file, the scheduler divides a job into small tasks, each of which is allocated to an idle server to deal with a file block concurrently.

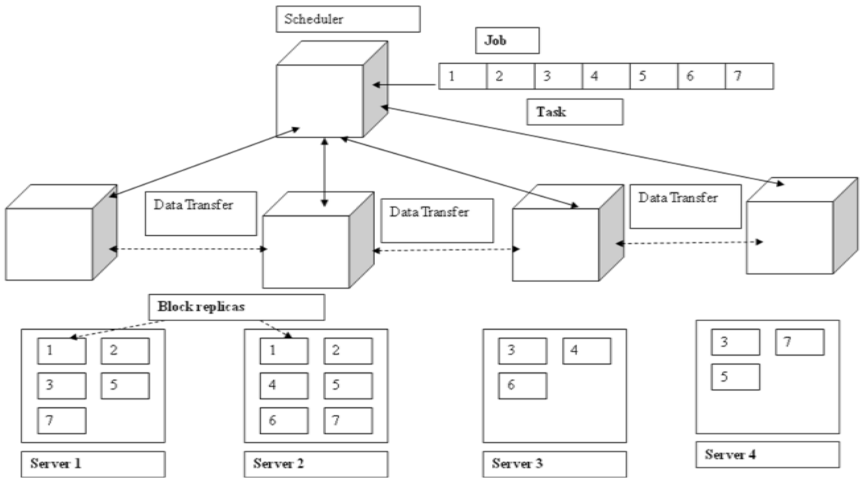

Fig.1. Architecture of Cloud Computing

In **task scheduling** [5], the task scheduler divides the task into subtasks and assigns it to different performer servers. The performer servers perform the subtasks assigned to them. As the task is divided into subtasks and assigned to different performer servers, the total time required for task completion is reduced depending upon the number of performer servers used. Thus, the time required for task completion is reduced and the load on the server is also reduced. In case of two performer servers, the time required for task completion process will be reduced to half and the load on the servers will also be reduced to half.

In **task migration** [6] process, the complete task is assigned to a performer server. The performer server performs the task if it is free. If the performer server is busy in performing the previous task, then the current task is migrated to another performer server to perform the task. In case of all performer servers busy, the current task is put into the queue of first performer server. When the first performer server complete it's task, then it performs the task which is in the queue.

II. Related Work

Task Scheduling and Migration in multiprocessing networks such as cloud computing are similar to task migration in distributed systems. Task migration has been implemented in multiple systems. Various algorithms have been presented for task scheduling and migration. These algorithms are totally made up of beginning stages, stop running task in source processor, task transmission to destination processor, start running task in destination, task running in destination processor, and removing remained information from source processor. Continuing this part, types of presented algorithms are studied [5].

Data-intensive applications are emerged over the past decade, which is an important part of distributed computing. Stork[7] is a specialized scheduler for data placement and data movement in Grid. The main idea of Stork is to map data close to computational resources.

The Gfarm[7] architecture is designed for petascale data-intensive computing. In Gfarm, several greedy scheduling algorithms are implemented to improve data locality.

In **Hadoop Default Scheduler** [8] when a server is idle, the scheduler chooses a data-local task, and then allocates the task to the server. If there is no feasible task, then the scheduler will select a random task.

Delay Scheduling[9] sets a delay threshold. If a server is idle and there is no task prefers the server, the scheduler skips the server and increases the delay counter byone. However, if the delay counter exceeds the delay threshold, the scheduler allocates a remote task to the server and sets the delay counter to be zero.

Good Cache Compute policy is similar to DS. It sets a utility threshold which is the upper bound of the number of idle servers.

MaxCover-BalAssign algorithm [10] works iteratively to produce a sequence of total allocations, and then outputs the best one. Each iteration consists of two phase maxcover and balassign. Since the remote cost is unknown, it calculates the virtual cost which is a prediction of the remote cost. Then it computes a total allocation by taking advantage of the virtual cost.

BAlance-Reduce (BAR) [11] a data locality driven task scheduling algorithm called BAlance-Reduce (BAR). *BAlance:* All tasks are allocated to their preferred servers evenly.
Reduce: Reducing the load on individual systems and minimizing response time.

Drawbacks of existing task scheduling algorithms: In traditional task scheduling, tasks are scheduled by users' requirements. Traditional way for task scheduling in Cloud Computing tends to use the direct tasks of users as the overhead application base. The problem is that there may be no relationship between the overhead application base and the way that different tasks cause overhead costs of resources in cloud systems. This problem leads to over-cost and over-priced in some high-volume simple tasks while under-cost and under-priced in low-volume complex ones.

III. The Proposed Approach

Present approach is enhancement of all the above algorithms. Factually, the results of assessments in the next parts of this paper show that the proposed approach has the advantages of all former methods.

Performance of the proposed approach: In our proposed approach given task will be a chunk of code and our system will convert it into a executable program, which is the task. Given task will be divided into n number of subtasks depending on the number of servers. All the performer server will perform the sub task and will show the execution time. If the performer server is busy then it migrates the allocated subtask to other performer server.

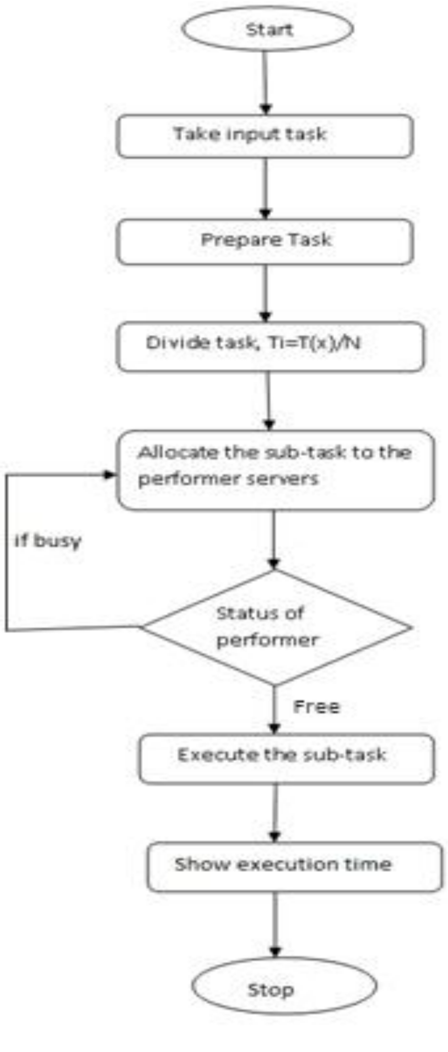

Fig.2. Proposed method flowchart

Algorithm

```
Algorithm for scheduling and migrating the tasks :
Step 1 : The tasks are received by the scheduler
Step 2 : for given tasks prepare task class
Step 3 : Divide task as per the number of scheduler
            $Ti=T(x)/N$
Step 4 : Allocate the sub tasks to the performer server
Step 5: Check the status of performer server
    $Tmig = \int_{0}^{T} \sum_{0}^{N} S(x)$
    i)      If busy go to step 4
    ii)     If free goto step 6
Step 6:Execute the subtask
Step 7: Show execution time
```

Terms used in the algorithm

T: Task

T(x): Total number of task

Ti : Scheduling Task

N : Number of server

S(x): Status of server

IV. Conclusion

In this paper we have studied various methods of task scheduling and task migration in cloud computing by the enhancement of BAR algorithm. We are expecting a system that will overcome all the disadvantages of the previous algorithm. By considering this system, that we are going to develop will show following expected results- Schedule based on number of performance server. Task is migrated if the current resource is busy. Efforts taken for data locality with heuristic approach in less time. The idea is implemented in real time and we are able to reduce the execution time.

Acknowledgement

First and foremost, we would like to thanks our project guide Prof S.C.Chaudhari and Head of the Department Prof. Vaishali.S. Nandedkar for the valuable guidance and advice. They inspired us greatly to work on this project. Their willingness to motive us contributed tremendously to our project.

Besides, we would also like to thank our Principal Dr. Y.V.Chavan, all the staffs and authorities of Padmabhooshan Vasantdada Patil Institute of Technology, University of Pune for providing us with good environment and facilities to complete our project.

References

[1] J. Dean and S. Ghemawat, *"MapReduce: simplified data processing on large clusters," Commun. ACM, vol. 51,no. 1, pp. 107–113, 2008.*

[2] M. Isardet al., *"Dryad: distributed data-parallel programs from sequential building blocks," in EuroSys '07. NewYork, NY, USA: ACM, 2007, pp. 59–72.*

[3] S. Ghemawat *et al.*, "The google file system," in SOSP '03. New York, NY, USA: ACM, 2003, pp. 29–43.

[4] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, Inc., 2009.

[5] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems (3rd Ed).* Springer, 2008

[6] G. Jong Yu, C. Yung Chan and T. Shi Chen*, "Task Migration in Nd Wormhole-Route Mesh Multi computer" ,Journal of Systelns Architecture, Vol. 50, No 4, 2004, pp.177-192.*

[7] O. Tatebe *et al.*, "Grid datafarm architecture for petascale data intensive computing," ser. CCGRID '02. Washington, DC, USA: IEEE Computer Society, 2002.

[8] Hadoop on demand. [Online]. Available: http://hadoop.apache.org/common/docs/r0.18.3/hod.html

[9] M. Zaharia *et al.*, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in EuroSys '10. New York, NY, USA: ACM, 2010.

[10] M. J. Fischer *et al., "Assigning tasks for efficiency inhadoop: extended abstract," in SPAA '10. New York, NY, USA: ACM, 2010, pp. 30–39.*

[11] Jiahui Jin; Junzhou Luo; Aibo Song; Fang Dong; Runqun Xiong; , *"BAR: An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing," Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on , vol., no., pp.295-304, 23-26 May 2011*