www.ijiarec.com

ISSN:2348-2079 Volume-6 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

Automatic Memory Control of Multiple VM using Real Time Task Oriented Task Scheduling Algorithm

[1]Mr. C. Mani M.C.A., M.Phil., M.E., Associate Professor,
[2]Mr. N. Gnanavel Final MCA.,
Department of MCA , Nandha Engineering College (Autonomous), Erode-52.
E-Mail ID: cmanimca@gmail.com, gnanavelnataraj@gmail.com

Abstract **In this paper, to devise a system for automatic memory control based on the balloon driver in VMs. Researchers can download our toolkit. The project aims to optimize the running times of applications in consolidated environments by overbooking and/or balancing the memory pages of VMs. Unlike traditional methods, such as MEB system is lightweight and can be completely integrated into user space without interfering with VMM operation. In this project design a global-scheduling algorithm based on the dynamic baseline to determine the optimal allocation of memory globally. In this existing system evaluate optimized solution to memory allocation using real workloads (DaCapo and PTS) that run across VMs. The virtualization technique enables multiple virtual machines (VMs) to be placed on the same physical hosts and supports the live migration of VMs between physical hosts based on the performance requirements.**

Keywords— **VM, Cloud, DRF, CPU, ATP, ACP, ASJS.**

I. INTRODUCTION

In basic terms, cloud computing is the phrase used to describe different scenarios in which computing resource is delivered as a service over a network connection (usually, this is the internet). Cloud computing is therefore a type of computing that relies on sharing a pool of physical and/or virtual resources, rather than deploying local or personal hardware and software.

Cloud can achieve the same level of computing power as a supercomputer does, but at a much reduced cost. Cloud is like a virtual supercomputer. However, we need to consider about many conditions such as network status and resource status because the members of Cloud are connected by networks.

Cloud is also a heterogeneous system. Scheduling independent tasks on it is more complicated. In order to utilize the power of Cloud computing completely, we need an efficient job scheduling algorithm to assign jobs to resources. This paper focuses on the efficient job scheduling considering the completion time of jobs in a Cloud environment.

The paper considers Adaptive Scoring Job Scheduling algorithm EASJS which aims to decrease job's completion time. We consider not only the computing power of each resource in the Cloud but also the transmission power of each cluster in a Cloud system. The computing power of each resource is defined as the product of CPU speed and available CPU percentage and the transmission power of each cluster is defined as the average bandwidth between different clusters. EASJS uses the status of each resource in the Cloud as parameters to initialize the cluster score of each cluster. The cluster score of each cluster will be adjusted by applying local

II.RELATED WORKS

In this paper the author Michael Armbrust stated that Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it.

In this paper the authors D. Saha and D. Menasce state that most parallel jobs cannot be fully parallelized. In a homogeneous parallel machine-one in which all processors are identical-the serial fraction of the computation has to be executed at the speed of any of the identical processors, limiting the speedup that can be obtained due to parallelism. In a heterogeneous architecture, the sequential bottleneck can be greatly reduced by running the sequential part of the job or even the critical tasks in a faster processor.

Zheng-Yi Huang proposed an adaptive and dynamic scheduling method, called most fit task first (MFTF), for a class of computational Clouds, which are characterized by heterogeneous computing nodes and dynamic task arrivals. Some existing static scheduling methods assume that tasks arrive statically and may not perform well in the case of dynamic task arrivals. Their method can get stable task execution times whether tasks arrive statically or dynamically. They compared the task execution time with other methods to show the performance of the scheduling method.

In the paper the authors stated that a 'Cloud' is an infrastructure for resource sharing. It is used for large-scale data processing, many of the applications being scientific ones. Cloud scheduling is a vital component of a Cloud infrastructure. Reliability, efficiency (in terms of time consumption) and effectiveness in resource utilization are the desired characteristics of Cloud scheduling systems. Many algorithms have been developed for Cloud scheduling. In our previous work, they proposed two scheduling algorithms (the Multilevel Hybrid Scheduling Algorithm and the Multilevel Dual Queue Scheduling Algorithm) for optimum utilization of processors in a Cloud computing environment.

A. Allocating modules to processors

In this paper not aware of any other published analysis of the particular dataset to be discussed. However, in, the authors analyze a similar but not publicly available data set collected within Google. The dataset there covers a much longer period of 20 cell (cluster)-days. The authors identified a set of resource consumption "shapes" for tasks that can be used or cluster scheduling and capacity planning. The authors in seek to map observed data to task "shapes" by quantizing data for task duration, average CPU, and average memory into small, medium, and large levels. The authors then applied the k-means clustering with the initial clusters subject to the small, medium, and large level constraints. The number of final clusters is set to 8, with the cluster merging process driven by lowering the within cluster coefficient of variation.

B. Development and Performance Analysis

In this paper to address the problem of fair allocation of multiple types of resources is to users with heterogeneous demands. It proposes Dominant Resource Fairness (DRF), a generalization of max-min fairness for multiple resources. The intuition behind DRF is that in a multi-resource environment, the allocation of a user should be determined by the user's dominant share, which is the maximum share that the user has been allocated of any resource. In a nutshell, DRF seeks to maximize the minimum dominant share across all users. For example, if user A runs CPU-heavy tasks and user B runs memory-heavy tasks, DRF attempts to equalize user A's share of CPUs with user B's share of memory. In the single resource case, DRF reduces to max-min fairness for that resource. It proposes Dominant Resource Fairness (DRF), a new allocation policy for multiple resources that meets all four of the required properties in the previous section.

III. METHODOLOGY

The scheduling objectives are to improve the system's schedule ability for real-time tasks and save energy. To achieve the objectives, we employed the virtualization technique and a rolling-horizon optimization scheme. First, a rolling-horizon scheduling architecture, and then a task-oriented energy consumption model was analyzed and built. A novel energy-aware scheduling algorithm named EARH for real-time tasks, in which a rolling-horizon policy was used to enhance the system's schedule ability.

Additionally, the resource scaling up and resource scaling down strategies were developed and integrated into EARH, which can flexibly adjust the active hosts' scale so as to meet the tasks' real-time requirements and save energy

The existing methodology presents a workload characterization of nodes by dividing tasks into task classes using the K-means algorithm. However, different from existing methods whose main objective is to understand workload characteristics, the existing system finds accurate workload characterization, while supporting task classification (e.g., labeling) at runtime.

Note that machines are naturally characterized (i.e., there are 10 types of machines in the cluster). Thus, the existing solution will mainly focus on task characterization.

Once the workload characterization has been obtained, the existing system introduces a monitoring mechanism that allows Harmony to capture the runtime workload composition in terms of arrival rate for each task class. To make provisioning decisions, it defines a container as a logical reservation of resources that is meant to host tasks belonging to the same task class.

It designs a load prediction algorithm that can capture the future resource usages of applications accurately. The algorithm can capture the rising trend of resource usage patterns and help reduce the placement churn significantly. It looks inside a VM for application level statistics, e.g., by parsing logs of pending requests. Doing so requires modification of the VM which may not always be possible.

- Each task is considered as separate unit and role back.
- To not apply vertical scaling of VM sin terms of CPU in existing energy-aware model.
- A single task is given to a selected single cluster only.
- Storage capacity of cluster resources is not taken into account.
- Replication strategy is not included

IV. PROPOESD METHODOLOGY

When science and technology advance, the problems encountered become more complicated and need more computing power. In contrast to the traditional notion of using supercomputers, Cloud computing is proposed. Distributed computing supports resource sharing. Parallel computing supports computing power. Cloud computing aims to harness the power of both distributed computing and parallel computing. The goal of Cloud computing is to aggregate idle resources on the Internet such as Central Processing Unit (CPU) cycles and storage spaces to facilitate utilization. When human culture advances, current problems in science and engineering become more complicated and need more computing power to tackle and analyze.

A supercomputer is not the only choice for solving complex problems any more as a result of the speed-up of personal computers and networks. Cloud technology, which connects a number of personal computer clusters with high speed networks, can achieve the same computing power as a supercomputer does, also with a lower cost.

However, cloud is a heterogeneous system. Scheduling independent tasks on it is more complicated. In order to utilize the power of Cloud completely, we need an efficient job scheduling algorithm to assign jobs to resources in a cloud. This papert proposes an Enhanced Adaptive Scoring Job Scheduling algorithm (EASJS) for the Cloud environment. Compared to other methods, it can decrease the completion time of submitted jobs, which may compose of computing-intensive jobs and data-intensive jobs

Enhanced Adaptive Scoring Job Scheduling (EASJS) aims to decrease job's completion time. It considers not only the computing power of each resource in the Cloud but also the transmission power of each cluster in a Cloud system.

It defines the computing power of each resource, the product of CPU speed and available CPU percentage. The transmission power of each cluster is defined as the average bandwidth between different clusters. It should use the status of each resource in the Cloud as parameters to initialize the cluster score of all clusters.

Along with existing system implementation, Enhanced Adaptive Scoring Job Scheduling algorithm (EASJS) for Job splited into 'N' tasks along with Replication Strategy. In addition, jobs are divided into sub tasks and given to one or more clusters. Storage capacity requirement is also included in Cluster Score calculation.

- Each job is considered as sub tasks.
- A single job is given to a selected multiple clusters since jobs are split into tasks.
- Cluster score values are recalculated even during the job is partially completed. This is achieved when a particular sub task is finished.
- Storage capacity of cluster resources is taken into account.
- Multiple P and Q and R values are calculated for each sub task and so cluster assignment is effective than existing system.
- Replication method assists in faster job completion.

In this section, the cluster score is calculated based on the following formula.

$$\mathbf{CS_i = P\ .\ ATP_i + \ Q\ .\ ACP_i}$$

where CS_i is the cluster score for cluster i, a and b are the weight value of ATP_i and ACP_i respectively, the sum of □ and □ is 1, ATP_i and ACP_i are the average transmission power and average computing power of cluster i respectively. ATP_i means the average available bandwidth the cluster i can supply to the job and is defined as:

$$ATP_i = \frac{\sum_{j=1}^{m} Bandwidth\_available_{i,j}}{m-1}, \quad i \neq j$$

where $Bandwidth\_available_{i,j}$ is the available bandwidth between cluster i and cluster j, m is the number of clusters in the entire Cloud system.
Similarly, ACP_i means the average available CPU power cluster i can supply to the job and is defined as:

$$ACP_i = \frac{\sum_{k=1}^{n} CPU\_Speed_k * (1 - load_k)}{n}$$

where CPU_speed_k is the CPU speed of resource k in cluster i, $load_k$ is the current load of the resource k in cluster i, n is the number of resources in cluster i. Also let

$$CP_k = CPU\_Speed_k * (1 - load_k)$$

CP_k indicates the available computing power of resource k.

Because the transmission power and the computing power of a resource will actually affect the performance of job execution, these two factors are used for job scheduling. Since the bandwidth between resources in the same cluster is usually very large, we only consider the bandwidth between different clusters.

Local update and global update are used to adjust the score. After a job is submitted to a resource, the status of the resource will change and local update will be applied to adjust the cluster score of the cluster containing the resource. What local update does is to get the available CPU percentage from Information Server and recalculate the ACP, ATP and CS of the cluster. After a job is completed by a resource, global update will get information of all resources in the entire Cloud system and recalculate the ACP, ATP and CS of all clusters.

In addition with formula, the storage capacity is also calculated like Average Transmission Power and so sum of P,Q and R is 1. All other calculations are used in same scenario as above module. The job is split into tasks with P,Q and R values for each sub task. So one cluster is assigned for one task and others cluster for other tasks. Like wise jobs are considered as replica units and so more clusters are assigned for each job.

V. PERFORMANCE EVALUATION

The following **Table 1** describes experimental result for number of social file search process in existing and proposed hit rate analysis. The table 5.1 contains number of node; existing and proposed probability VM distribution rate details are shown.

S.No.	**Number of File VM Node**	**Existing System**	**Proposing System**
1	50	0.266	0.321
2	100	0.279	0.287
3	150	0.315	0.328
4	200	0.348	0.352
5	250	0.389	0.397

Table 1 Probability VM Distribution Rate Analysis

VI. CONCLUSION

This project proposes an adaptive scoring method to schedule jobs in grid environment. ASJS selects the fittest resource to execute a job according to the status of resources. Local and global update rules are applied to get the newest status of each resource. Local update rule updates the status of the resource and cluster which are selected to execute the job after assigning the job and the Job Scheduler uses the newest information to assign the next job. Global update rule updates the status of each resource and cluster after a job is completed by a resource. It supplies the Job Scheduler the newest information of all resources and clusters such that the Job Scheduler can select the fittest resource for the next job. The experimental results show that ASJS is capable of decreasing completion time of jobs and the performance of ASJS is better than other methods.

In the future, EASJS can be applied to real grid applications. This project focuses on job scheduling. The project can be modified to consider division of file and the replica strategy in data-intensive jobs. Jobs are independent in this project, but they may have some precedence relations in real-life situation. Studying and improving EASJS for such kinds of jobs may be carried out in the future.

REFERENCES

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia, A view of cloud computing, Communications of the ACM 53 (4) (2010) 50–58.

[2] D. Saha, D. Menasce, S. Porto, Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures, Journal of Parallel and Distributed Computing 28 (1) (1995) 1–18.

[3] Sheng-De Wang, I-Tar Hsu, Zheng-Yi Huang, Dynamic scheduling methods for computational grid environment, International Conference on Parallel and Distributed Systems 1 (2005) 22–28.

[4] Syed Nasir Mehmood Shah, Ahmad Kamil Bin Mahmood, Alan Oxley, Dynamic multilevel hybrid scheduling algorithms for grid computing, Procedia Computer Science 4 (2011) 402–411.

[5] Grid Scheduling Dictionary Project ; retrieved December 05, 2010; from http://www.mcs.anl.gov/~schopf/ggf-sched/GGF5/sched-Dict.1.pdf

[6] D. Fernandez-Baca, Allocating modules to processors in a distributed system, IEEE Trans, Software Eng., 1989

[7] S. N. M. Shah, A. K. B. Mahmood and A. Oxley, Development and Performance Analysis of Grid Scheduling Algorithms, Communications in Computer and Information Science, Springer, vol. 55, pp. 170–181, 2009.