Vol. 2, Issue 3 (July - Sept. 2015)
ISSN : 2394-2975 (Online)
ISSN : 2394-6814 (Print)

Optimal Reduces Energy Consumed Framework Using Distributed Resources Material Base Allocation

[I]N.Karthika, [II]D.Ponniselvi
[I]M.Phil Full-Time Research Scholar, [II]M.Sc.,M.Phil., Assistant Professor
[I,II]Dept. of Computer Science and Applications, Vivekanandha College of Arts and Sciences For Women (Autonomous), Elayampalayam, Tiruchengode, Tamil Nadu, India.

Abstract
Data centers consume tremendous amounts of energy in terms of power distribution and cooling. Dynamic capacity provisioning is a promising approach for reducing energy consumption by dynamically adjusting the number of active machines to match resource demands. However, despite extensive studies of the problem, existing solutions have not fully considered the heterogeneity of both workload and machine hardware found in production environments. To address this limitation, this project presents Harmony, a Heterogeneity-Aware dynamic capacity provisioning scheme for cloud data centers. Specifically, it first uses the K-means clustering algorithm to divide workload into distinct task classes with similar characteristics in terms of resource and performance requirements. Then it presents a technique that dynamically adjusting the number of machines to minimize total energy consumption and scheduling delay.

Keywords
Data center, load balancing, Harmony cloud data center, EASJSA, Energy Consumption

I. Objectives

- To calculate cloud area status at a given refresh period.
- To group cloud areas based on their processing ability.
- To make nodes inside the cloud area not treated equally. According to their processing and storage power, the partial job is assigned to them.
- To assign one job to multiple nodes after the job is split according to the nodes capability.
- To split the Jobs are into sub tasks and assign them to more cloud nodes.
- To take in to account both dependent tasks and Independent task scheduling.
- To consider job replication strategy.
- To decrease completion time of jobs.

II. Problem Definition

It 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.
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 data center.
The proposed system covers all the existing system approach. In addition, among all the cloud nodes, Adaptive Scoring Job Scheduling algorithm (ASJS) is applied for cloud nodes resource scheduling so that the given job is split into 'N' tasks along with Replication Strategy.
Adaptive Scoring Job Scheduling (ASJS) aims to decrease job's completion time. It considers only the computing power of each resource in the grid but also the transmission power of each cluster in a grid 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.
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.

III. Review of Literature

1. RELATED WORK

Effective Straggler Mitigation: Attack of The Clones

Authors:
Ganesh Ananthanarayanan, Ali Ghodsi1, Scott Shenker, Ion Stoica
In this paper [1] describers the small jobs, that are typically run for interactive data analyses in data centers, continue to be plagued by disproportionately long-running tasks called stragglers. In the production clusters at Facebook and Microsoft Bing, even after applying state-of-the-art straggler mitigation techniques, these latency sensitive jobs have stragglers that are on average 8 times slower than the median task in that job. Such stragglers increase the average job duration by 47%. This is because current

mitigation techniques all involve an element of waiting and speculation. Instead of this propose full cloning of small jobs, avoiding waiting and speculation altogether. Cloning of small jobs only marginally increases utilization because workloads show that while the majority of jobs are small, it consumes a small fraction of the resources. The main challenge of cloning is, however, that extra clones can cause contention for intermediate data. Delay assignment technique is used in this approach, which efficiently avoids such contention. Evaluation of system, Dolly, using production workloads shows that the small jobs speedup by 34% to 46% after state-of-the-art mitigation techniques have been applied, using just 5% extra resources for cloning.

Managing Server Energy and Operational Costing Hosting Centers

Authors:
Wubi Qin Yiyu Chen Wubi Qin
This paper has presented the first formalism to the problem of reducing server energy consumption at hosting centers running multiple applications towards the goal of meeting performance based SLAs to client requests. Though prior studies have shown energy savings for server clusters by server turn-offs and DVS, these savings come at a rather expensive cost in terms of violating the performance-based SLAs (which are extremely important to maintain the revenue stream). Further, previous proposals have not considered the cost of server turn-offs, not just in terms of time.

Analysis and Lessons from A Publicly Available Google Cluster Trace

Authors:
Yanpei Chen, Archana Ganapathi ,Rean Griffith ,Randy H. Katz
In this paper, publicly available traces are demonstrated in valuable regardless of dataset size. Clearly, more data would allows to generalize finding and gain additional system design insights. It hopes the Google public data release foreshadows a seachange in attitudes towards making production traces publicly available. It argues for a public production system trace repository similar to the Computer Failure Data Repository. To facilitate such a repository while addressing trade secret and user privacy concerns, future work should develop a toolkit with anonymizers, data format converters, and standard algorithms for detailed statistical analysis. First step in this direction represented in this paper.

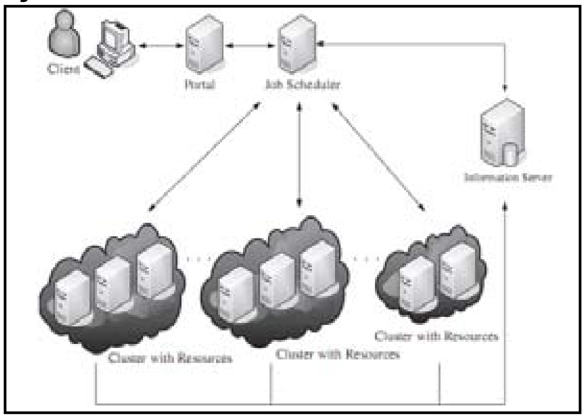
Dynamic Right-Sizing For Power-Proportional Data Centers

Authors:
Minghong Lin and Adam Wierman Lachlan L. H. Andrew Eno Thereska
This paper has provided a new online algorithm, LCP (w),for dynamic right-sizing in data centers. The algorithm is motivated by the structure of the optimal offline solution and guarantees cost no larger than 3 times the optimal cost, under very general settings – arbitrary workloads, general delay cost models, and general energy cost models. Further, in realistic settings the cost of LCP(w) is nearly optimal. Additionally, LCP(w) is simple to implement in practice and does not require significant computational overhead. Additionally, the case studies used to evaluate LCP(w)highlight that the cost and energy savings achievable via dynamic right-sizing are significant. The case studies highlight that if a data center has PMR larger than 3, a cost of toggling a server of less than a few hours of server costs, and less than 40% background load then the cost savings from dynamic right-sizing can be conservatively estimated at larger than 15%.Thus, even if a data center is currently performing valley filling, it can still achieve significant cost savings via dynamic right-sizing.

IV. Enhanced Adaptive Scoring Job Scheduling Algorithm (ASJS)

ASJS aims to decrease job's completion time. We consider not only the computing power of each resource in the cloud but so the transmission power of each cluster in a grid system. The computing power of each resource is defined as the product CPU speed and available CPU percentage and the transmission power of each cluster is defined as the average bandwidth between different clusters. ASJS uses the status of each resource in the grid as parameters to initialize the cluster score of each cluster. The cluster score of each cluster will be adjusted by applying local update and global update. The system will submit a job to the most appropriate resource according to the scores.

System Framework



Cluster score
The cluster score is calculated based on the following formula.
$CS_i = \alpha . ATP_i + \beta . ACP_i$
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 a and b 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 grid 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.

V. Experimental Results

Results and Discussion

The table 1.1 describes the resources allocation for K-Mean clustering and EASJS algorithm. The table contains number of resources allocation for K-Mean and EASJS Algorithm.

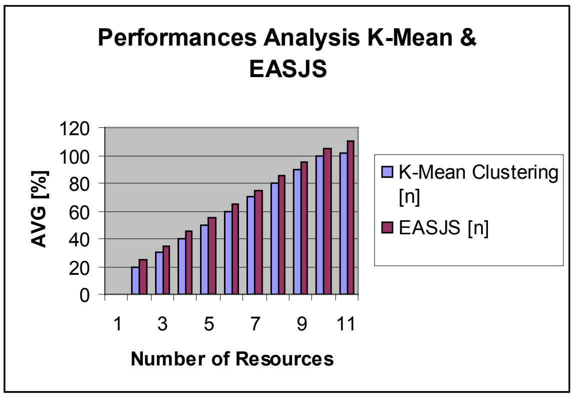


Fig 1.2 : Performances Analysis- Resources Allocation

Table 1.1 Performances Analysis- Resources Allocation

S. No	Number of Resource [n]	K-Mean Clustering [n]	EASJS [n]
1	30	20	25
2	40	30	35
3	50	40	45
4	60	50	55
5	70	60	65
6	80	70	75
7	90	80	85
8	100	90	95
9	110	100	105
10	120	102	110

The table **1.3** describes the resources allocation time complexity for K-Mean clustering and EASJS algorithm per seconds. The figure contains number of resources allocation time details for K-Mean and EASJS algorithm.

Fig 1.3 : Performances Analysis- Resources Allocation Time

S. No	Number of Resource [n]	K-Mean Clustering [Sec]	EASJS [Sec]
1	100	122	118
2	200	143	138
3	300	152	153
4	400	164	156
5	500	173	166
6	600	155	146
7	700	146	142
8	800	172	171
9	900	143	142
10	1000	167	163

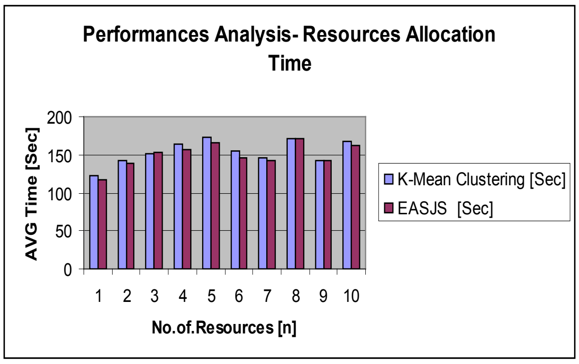


Fig 1.4 : Performances Analysis- Resources Allocation Time

FINDINGS

The following result finding for our experimental works, they are:

- It is found that the cluster selection is efficient if the job is split into sub tasks.
- Resources are effectively utilized and waiting time is less in scheduling next successive job in queue.
- Resources with limited values are also having the chance for job allocation if the job is split into sub tasks.
- Instead of calculating the right cluster after each job completion, the proposed system calculates the clusters availability at regular intervals so that any new job can be assigned even during the execution of current job.
- Overall efficiency of the grid is more compared to existing system.
- Better suitable for jobs which can be split based on RAM, CPU speed and storage location.
- The experimental results show that EASJSA is capable of decreasing completion time of jobs and the performance of EASJS is better than other methods.
- Inter dependant jobs are not combined in the proposed system which may be future work.
- Studying and improving EASJSA for such kinds of jobs may be carried out in the future.

VI. Conclusion and Future Works

This work proposes an adaptive scoring method to schedule jobs in grid environment. EASJS 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 EASJS is capable of decreasing completion time of jobs and the performance of EASJS 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 using genetic algorithm and game theory.

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.*

[8]. *S. Haines, Pro Java EE 5 Performance Management and Scalability; retrieved July 07, 2010;from http://www.theserverside.com/news/1364275/Pro-Java-EE-5-Performance-Management-and-Scalability.*

[9]. *H. Li and R. Buyya, Model-Driven Simulation of Grid Scheduling Strategies, Third IEEE International Conference on e-Science a Grid Computing, 2007.*

[10] *M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing system, Journal of Parallel and Distributed Computing 59 (1999) 107–131.*