CONSOLIDATING BATCH AND TRANSACTIONAL WORKLOADS USING DEPENDENCY STRUCTURE PRIORITIZATION

S.NIVETHITHA[1], V.S.SHANKAR SRIRAM[2]
[1]School of Computing, SASTRA University, Thanjavur-613401, India.
[2] Senior Assistant Professor, School of Computing, SASTRA University, Thanjavur-613401, India.
Email: nivethithasomu@gmail.com, sriram@it.sastra.edu

***Abstract* - Organizations offer efficient services to their customers through cloud. These services can either be a batch or transactional workloads. To offer a real-time service, there comes a need to schedule these workloads in an efficient way. An idea to consolidate these workloads enables us to cut down the energy consumption and infrastructure cost. It will be harder to consolidate both these workloads due to the difference in their nature, performance goals and control mechanisms. The proposed work implements the concept of Dependency Structure Prioritization (DSP) to assign priority to the job. This work tends to make effective resource utilization through reducing the number of job migration and missed deadline jobs by considering the deadline and the priority of the job as the most important evaluation factor.**

Keywords: ***Dependency Structure, Batch Workloads, Transactional Workloads, Consolidation***

I INTRODUCTION

Consolidation means placing different jobs on the same physical machine that gets executed based on some scheduling policy. In order to accomplish this either we need an integrated management tools or a scheduling algorithm that schedules these workloads in an easier way with minimum overhead.

DSP schedules the job based on dependency analysis. By nature, whenever there are n number of jobs, there exist some form of dependency between them. Dependency can be either in the form of open or closed structure. Consider a tree with its nodes marked from J1 to Jn, to have a clear view on dependency structure let us take a part of a graph into consideration. Let J1 at level 1 be the parent of J3 and J4 at level 2 and J2 at level 1 be the parent of J5 and J6 at level 2 and so on like a binary tree. There exists a dependency between each node connected by an edge. In case of an open dependency structure, J1 needs to be executed at some point before J3. Whereas, closed dependency structure, J1 needs to be executed exactly before J3. Sometimes closed dependency structure leads to a situation where J1 executes more than once when the job execution is in order (J1, J2, before the execution of J3 once again J1 needs execution). This form of dependency between jobs has a larger impact on the overall execution time. The node having a greater dependency will be having the highest priority than others. This type of scheduling mechanism will enhance the processor utilization by keeping the CPU busy without letting them idle.

This paper is sectioned as follows: Section 2 enlightens the survey made on heterogeneous workload, Section 3 describes about the proposed work based on DSP, Section 4 details the implementation of DSP and the evaluation strategy and Section 5 concludes the paper.

II RELATED WORK

Aziz Murtazaev and Sangyoon Oh [1] proposed a SERver CONsolidation algorithm named SERCON which reduces the number of nodes and migrations used. This algorithm selects Virtual Machine (VM) in the least loaded node and tries to shifts to the first node, if it fails, then tries on the second node and so on. David Carrera and Malgorzata Steinder et al. [2] proposed a lowest relative performance scheduling policy for fairness allocation of resources using application placement controller along with the control mechanism for online system configuration. Deep Mann and Inderveer Chana [3] created a workload consolidation portal to analyze the energy consumption of varying length heterogeneous workloads in cloud. Diego Didona and Pierangelo Di Sanzo et al. [4] presented some key design choices for Cloud-TM workload analyzer which is responsible for gathering information on the present and future demands of application workloads. Dinesh Kumar and Zon-yin Shae et al. [5] proposed delayed Lookahead Optimizing Scheduler and hybrid Lookahead Optimizing Scheduler algorithm for executing heterogeneous workloads based on runtime elasticity. Hyunjoo Kim and Yaakoub el-Khaura et al. [6] presented a framework to monitor and ensure the satisfaction of application and system heterogeneity.

Mousumi Paul and Debabrata Samanta et al. [7] proposed a scheduling mechanism based on lexi-search using central and local middleware for task partitioning which ensures fault tolerance. Mumtaz Ahmad and Ashraf Aboulnaga et al. [8] proposed a batch scheduling algorithm named Qshuffler which analyses the interaction among queries. Rushi Agrawal and Vaishali Sadaphal [9] studied the scheduling policy of batch workloads in multi-processor and mainframe environments using dependency structures. Saurabh Kumar Garrg and Chu Shin Yeo et al. [10] presented an optimal scheduling strategy to address the problem of high energy consumption impact on environmental condition and the profit of the cloud provider. Saurabh Kumar Garg and Srinivasa K. Gopalaiyengar et al. [11] presented a mechanism for admission control and scheduling heterogeneous workloads by stealing the unused cycles from transactional applications.

Vignesh T. Ravi and Michela Becchi et al. [12] presented a framework for job execution through GPU sharing with the help of computing the affinity score and kernel consolidation. Yung-Ching Hsu and Pangfeng Liu et al. [13] worked on the problem of allocating set of physical servers to the sequence of job in order to reduce the wasted energy due to over-provisioning of resources to a job at the peak time. Zhenhua Liu and Yuan Chen et al. [14] provided an integrated workload management system inorder to reduce the power consumption in the data centers.

III PROPOSED WORK

3.1 Heterogeneous Workload:

Heterogeneous workload comprises of both transactional and batch workloads. Both of them have different processing requirements like flow control and dynamic application placement for transactional workloads and different scheduling policy for batch jobs. Initially, these types of workloads were scheduled on to separate machines. In order to reduce energy and infrastructure cost many researchers contributed their works towards the consolidation of workloads onto a single machine. Transactional workloads need to get serviced in time; in case of batch workloads the completion time and the result of execution are vital.

3.2 System Architecture:

The system architecture of the proposed work consists of a two dual core processor, one for hosting cloud (Xen Virtual Server on Cent OS and Windows 7 as guest OS) and another for dispatching job and a 8-port switch for interconnectivity.

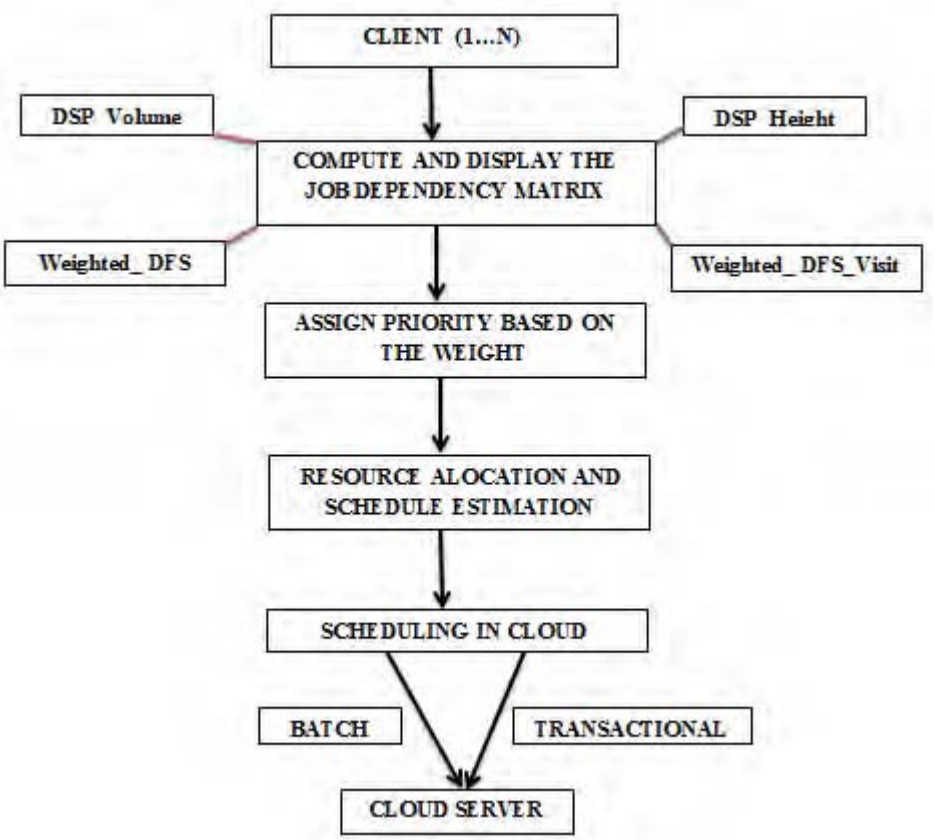

Fig 1. Scheduling Process Using DSP

3.3 Dependency Structure Prioritization:

In this paper, a software engineering technique for test case prioritization named Dependency Structure Prioritization (DSP) is used to assign priority for the job. DSP is based on four algorithms named DSP_volume, DSP_height, Weighted_DFS and Weighted_DFS_Visit. DSP_volume calculates the transitive closure for all the nodes, DSP_height measures the path length (deadline) variation for a current job with the rest, Weighted_DFS prioritizes each job and Weighted_DFS_Visit prioritize job based on graph coverage. The entire scheduling process using DSP has been depicted in Fig 1. These algorithms process the dependency matrix (input file) and assigns priority to each job based on the maximum weight. After assigning priority to each job, the process of scheduling starts. Jobs with the least deadline needs to be scheduled first. The ties are broken by considering the Least Deadline First (LDF) policy and then the priority (maximum weights). The resource requirement for each job should be satisfied through providing necessary CPU cycles and memory. Resources will be offered in terms of Virtual Machine (VM) on the cloud server. After completing the task of resource allocation, the execution of job starts on cloud.

IV RESULTS AND DISCUSSION

Job dependency between jobs exhibits some form of relationship between them. This form of relation has a greater impact on scheduling strategy. Let us consider an example (Table 1) as an initial schedule. Based on the number of jobs, construct a dependency graph.

Table 1. Input Data

JOB	EXECUTION TIME	DEADLINE
J1	**3**	**6**
J2	**3**	**7**
J3	**2**	**20**
J4	**5**	**21**
J5	**6**	**27**
J6	**6**	**28**

Consider a binary tree with its nodes labeled J1, J2, to Jn (randomly) representing the job. A node can either have one or more node as its children.

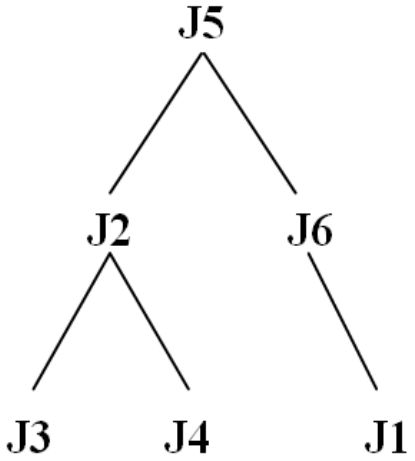

Fig 2. Dependency Graph

Dependency between the each job can be represented as shown in Table 2.

Table 2: Dependency Table (.txt file)

JOB	DEPENDENCY
J1	-
J2	DEP :J3,J4
J3	-
J4	-
J5	DEP :J2,J6
J6	DEP :J1

Compute and display the dependency matrix where dependent jobs are marked with 1 and 0 otherwise (Table 3).

Table 3. Dependency Matrix

	J1	J2	J3	J4	J5	J6	Weight
J1	0	0	0	0	0	0	0
J2	0	0	1	1	0	0	2
J3	0	0	0	0	0	0	0
J4	0	0	0	0	0	0	0
J5	0	1	0	0	0	1	2
J6	1	0	0	0	0	0	1

Each algorithm processes the input based on its own characteristics and displays the output in the form of a dependency matrix with recomputed values. For the same input, each algorithm will display distinct matrix values. Sum up the values in the dependency matrix for calculating the weights for each job. The weights were calculated based on dependency, so the job with highest weight will be assigned with the highest priority.

The next task is to effective schedule the jobs based on the deadline and the priority (dependency) of each job. The order of scheduling is based on (Table 4)

Table 4. Priority Class

PRIORITY (High – 1 Low – 0)	DEADLINE (High – 1 Low – 0)	ORDER OF PRIORITY
0	0	2
0	1	4
1	0	1
1	1	3

The job sequence given in (Table 1) will be scheduled (dual core system) in the following sequence (Table 5).

Table 5. Schedule Estimation

JOB	PRIORITY	TIME	PROCESSOR
J2	1	0 - 3	P1
J1	2	0 – 3	P2
J5	3	3 – 9	P1
J6	4	3 - 9	P2
J3	5	9 - 11	P1
J4	6	9 – 14	P2

The resource allocation (Fig 3) and the utilization ratio (Fig 4) shows that using DSP to schedule the batch and transactional workloads proves to have an higher utilization rate with more number of jobs to get executed in time. This type of scheduling reduces the power consumption and enhances the overall processor utilization of the system.

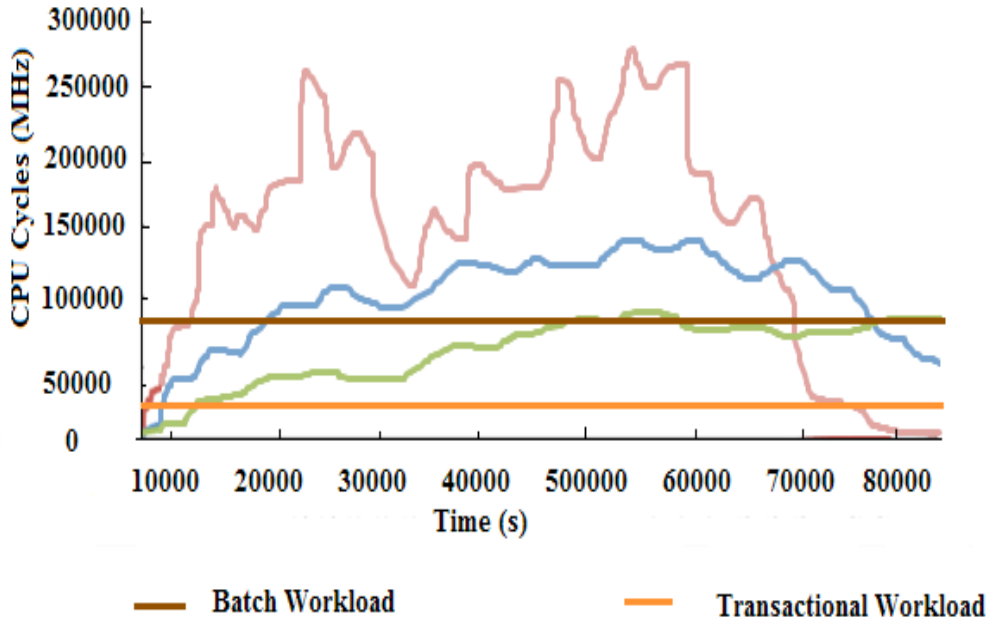

Fig 3. Resource Allocation (CPU Cycles)

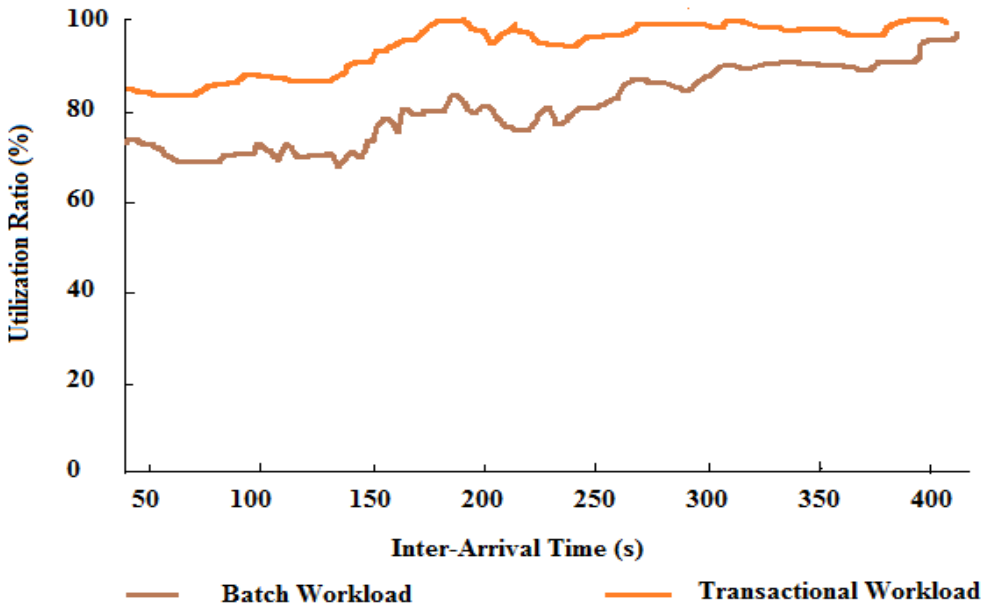

Fig 4. Utilization Ratio of DSP

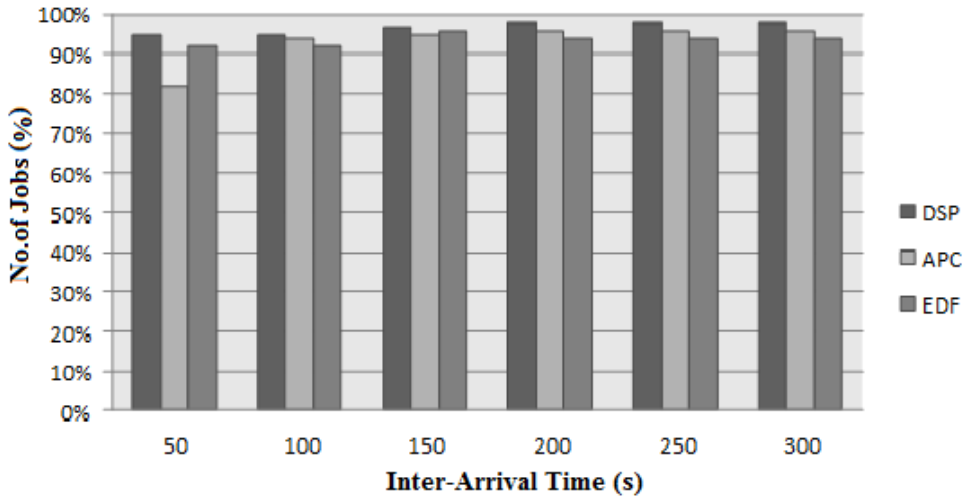

Fig 5. Performance Analysis of DSP

The performance of the DSP algorithm has been charted out in (Fig 5). DSP satisfies more number of jobs than Earliest Deadline First (EDF) and Application Placement Controller (APC) algorithm. Our algorithm schedules the jobs which has the least deadline irrespective of the priority. When there comes a tie up (two jobs having the same deadline), it looks over the priority. DSP proves to give out a better satisfaction (Percentile of jobs) than compared to other existing algorithms for scheduling heterogeneous workloads. DSP ensures better utilization of resources and reduced power consumption through establishing and efficient scheduling strategy.

V CONCLUSION

Scheduling heterogeneous workloads proves to be a most challenging task. With the recent research advancement and integration of existing algorithms in cloud there comes a new way to schedule these workloads in an efficient manner. However, it still faces difficulty in achieving the goal of minimizing job migration and maximizing the deadline satisfaction. Our proposed work uses DSP algorithm to prioritize the jobs and schedules them based on the LDF policy. This type of scheduling ensures to satisfy our goal. This work can be made as an alternative to the existing scheduling techniques on heterogeneous workloads.

VI REFERENCES

[1] Aziz Murtazaev, Sangyoon Oh, “Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing,” IETE Technical Review, 2011, pp 212-231.
[2] Carrera.D, Steinder.M, Whalley.I, Torres.J and Ayguade.E, “Autonomic Placement of Mixed Batch and Transactional Workloads”, IEEE Transactions on Parallel and Distributed Systems, 2012, pp 219 – 231.
[3] Ciciani, B., Didona, D., Di Sanzo, P., Palmieri, R., Peluso, S, Quaglia, F., Romano, P., Automated Workload Characterization in Cloud- based Transactional Data Grids , International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012, pp 1525 - 1533.
[4] Deep Mann and Inderveer Chana, “Heterogeneous Workload Consolidation for Efficient Management of Data Centers in Cloud Computing”, International Journal of Computer Applications, 2012, pp 0975 – 8887.
[5] Dinesh Kumar, Zon-yin Sha and Hani Jamjoom, ”Scheduling Batch and Heterogeneous Jobs with Runtime Elasticity in a Parallel Processing Environment”, International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012, pp 65-78.
[6] Hyunjoo Kim, Yaakoub el-Khamra, Ivan Rodero, Shantenu Jha and Manish Parashar, “Autonomic Management of Application Workflows on Hybrid Computing Infrastructure”, Journal of Scientific Programming, 2011, pp 75-89.
[7] Mousumi Paul, Debabrata Samanta and Goutam Sanyal, “Dynamic job Scheduling in Cloud Computing based on horizontal load balancing”, International Journal of Computer Technology and Applications, 2011, pp 1552-1556.
[8] Mumtaz Ahmad, Ashraf Aboulnaga, Shivnath Babu, Kamesh Munagala, "Interaction-Aware Scheduling of Report Generation Workloads”, The International Journal on Very Large Data Bases archive, 2011, pp 589-615.
[9] Rushi Agrawal and Vaishali Sadaphal, “Batch systems: Optimal scheduling and processor optimization”.
[10] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam, and Rajkumar Buyya, “Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers”, Journal of Parallel and Distributed Computing, 2011, pp 732-749.
[11] Saurabh Kumar Garg, Srinivasa K. Gopalaiyengar, and Rajkumar Buyya, “SLA-Based Resource Provisioning for Heterogeneous Workloads in a Virtualized Cloud Datacenter”, international conference on Algorithms and architectures for parallel processing, 2011, pp 371-384.
[12] Vignesh T. Ravi, Michela Becchi, Gagan Agrawal and Srimat Chakradhar, “Supporting GPU sharing in cloud environments with a transparent runtime consolidation framework”, international symposium
[13] Yung-Ching Hsu, Pangfeng Liu and Jan-Jan Wu, “Job sequence scheduling for cloud computing”, International Conference on Cloud and Service Computing (CSC), 2011, pp 212-219.
[14] Zhenhua Liu, Yuan Chen, Cullen Bash, Adam Wierman, Daniel Gmach, Zhikui Wang, Manish Marwah, Chris Hyser, “Renewable and Cooling Aware Workload Management for Sustainable Data Centers“, International Conference on Measurement and Modeling of Computer Systems, 2012, pp 175-186.