Volume 3, Issue 4, April 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

QoS-Aware Approaches to Real-Time task scheduling on Heterogeneous Clusters

Kalpana Manudhane*
ME(CSE) 2nd semester
G.H. Riasoni College of Engineering & Management,
Amravati, Maharashtra, India

Mr. Avinash Wadhe
M-Tech(CSE)
G.H. Riasoni College of Engineering & Management,
Amravati, Maharashtra, India

Abstract— A QoS guarantee on heterogeneous clusters has become very important need in today's real time system. Many of real time applications have different quality of service (QoS) requirements including reliability, security, data accuracy, data quality and so on. This paper gives study Self-Adaptive QoS-Aware (SAQA) algorithm with QoS requirement of data quality designed for soft real time system, QoS-Aware Fault Tolerant (QAFT) algorithm with QoS requirement of reliability designed for hard real time system and Two-Phase Scheduling Strategy (TPSS) with QoS requirement of security for security-critical real-time applications. SAQA algorithm, when system is in heavy load, can degrade the QoS levels of new tasks to guarantee high schedulability. In contrast, when the system is in light load, SAQA can significantly improve the QoS of new tasks. QAFT can tolerate one node's permanent failures and improve system reliability of clusters. Whereas, TPSS includes two algorithms DSRF and FMSL. This paper conclude with comparison of SAQA, QAFT and TPSS based on guarantee ratio and QoS level average.

Keywords— QoS-aware, Heterogeneous clusters, task scheduling, real time, scheduler model.

I. INTRODUCTION

In large-scale computing, *clusters* have been considered as an attractive platform due to the cost effectiveness, excellent extensibility and usability. Nodes in a cluster usually with their own disks and operating systems are interconnected by high-speed networks, e.g. Myrinet or fast Ethernet, to provide high-speed processing of computationally large or data-intensive applications, such as weather forecasting, molecular modeling (computing the structures and properties of chemical compounds, biological macromolecules, polymers, and crystals), etc. Specially, *heterogeneous clusters* [1] receive a focus in practice because machines (nodes) with different processing powers are usually assembled in a cluster for economical purpose. Nowadays, increasing attention has been drawn to develop and deploy *real-time* applications on heterogeneous clusters. Real-time applications depend not only on results of computation, but also on time instants at which these results become available. Generally, real-time applications are classified into two categories: soft real-time applications and hard real-time applications [2]. As for hard real-time applications, a timing constraints or deadline is hard as the failure to meet it is considered to be fatal error. Whereas, in soft real time system jobs have soft deadlines. An occasional missed deadline is usually considered tolerable. In addition to timeliness requirements, *quality of service (QoS)* [1] requirements must be addressed in various hard and soft real-time systems. Many of real time applications have different quality of service (QoS) requirements including reliability, security, data accuracy and so on.

For example, real-time stock systems require high quality of security (a dimension of QoS) to guarantee the data processing on clusters not being read or altered by malicious users. Another example is that some signal data can be processed using different algorithms. As for decoding block turbo codes, several algorithms can be employed. Note that, high-complexity algorithms can guarantee signal processing having higher QoS level (higher data accuracy) at the expense of processing time. Therefore, developing a QoS-aware infrastructure and scheduling algorithm on top of the structure are the keys for achieving efficient resource utilization, and improve the performance of real-time tasks on clusters. Regarding soft real time systems, most existing QoS-based real-time scheduling algorithms only strive to maximize the QoS-level for each new task on the basis of satisfying timing constraints. These algorithms do not consider the adaptability, which is able to result in some tasks not being accepted when the system is in heavy load, that is achieved by SAQA. The QoS-aware real-time systems must incorporate inherent high reliability features in case of hard real time system. Since the automated flight control system is used in the military battle field, the system must ensure that each task is executed within its deadline even in the presence of hardware or software faults. For the signal processing system, although some tasks missing their deadlines may not result in disaster, the outdated or half-baked processed data may be useless for users, especially in the field of modern information battle. Therefore, the system must guarantee its functional and timing correctness even when faults occur. Such a fault-tolerant mechanism is provided by QAFT [3].

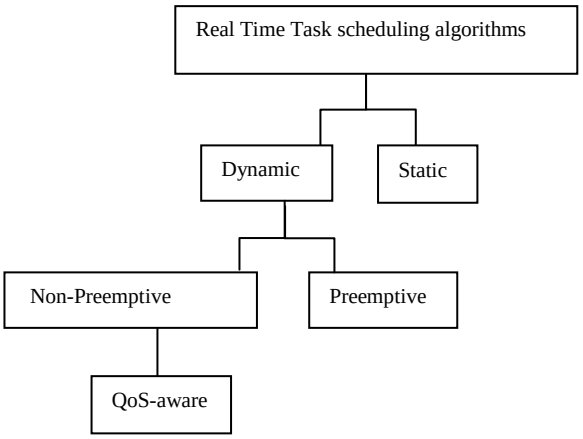
The remainder of this paper is organized as follows: Section II gives literature review describing system, scheduler and task model. In Section III, QoS-aware real time task scheduling algorithms, namely, SAQA, QAFT and TPSS are

described. Section IV gives comparison metrics and a comparative study of these task scheduling algorithms. Finally, in Section V, conclusion is presented that tells which algorithm is suitable for what requirement.

II. LITERATURE REVIEW

Scheduling algorithms for clusters have been found to be NP-complete [4], i.e., it is believed that there is no optimal polynomial time algorithm for them. So, we motivated to heuristic approaches to develop scheduling algorithms.

Real-time scheduling algorithms can be either static (off-line) or dynamic (on-line). In static algorithms, the assignment of tasks to processors (nodes) and the time at which the tasks start execution are determined a priori. Static algorithms are often used to schedule periodic tasks. However, those aperiodic tasks whose characteristics (arrival time, execution time and deadline) are not known a priori must employ *dynamic scheduling algorithms*. This paper focused on dynamically scheduling aperiodic tasks as shown in figure1.



Self-Adaptive QoS-Aware (SAQA)
QoS-Aware Fault Tolerant (QAFT)
Two-Phase Scheduling Strategy (TPSS)

Figure 1. Classification of real time task scheduling algorithms

This paper consider tasks having *no precedence constraints* because tasks with precedence constraints can be scheduled by transforming the precedence graph into independent nodes with new ready time and deadlines.

Scheduling algorithms in this paper are *non-preemptive*, i.e., when a task starts to execute on a processor, no other tasks interrupt its execution, which is more efficient, particularly suitable for soft real-time applications than the preemptive approaches due to reducing the overheads needed for switching among tasks.

The *scheduler model* [1] can be categorized into distributed scheduler model and centralized scheduler model. In a distributed scheduler model, tasks arrive independently at each local scheduler, which produces schedules in parallel with other schedulers. In a centralized scheduler model, all tasks arrive at a central processor called global scheduler, from which they are distributed to nodes in a cluster for further execution. In this paper, we adopt the centralized sort because implementation of a centralized scheduler model is simpler and easier than that of distributed scheduler model. Scheduler model vary according to algorithm employed.

This paper consider a task model [1] as a set $T=\{t_1, t_2,\ldots,t_n\}$ of independent real time tasks with soft deadlines. QoS refers to data quality after processing. One task can execute on only one node. A heterogeneous cluster is composed of a set $N= \{n_1, n_2,\ldots,n_m\}$ of heterogeneous nodes with different processing powers. E_{ij} is expected execution time of task t_i on node n_j and it is assumed to be known . a_i , d_i , f_i represent arrival time, deadline and finish time of task t_i .

Regarding soft real time systems, most existing QoS-based real-time scheduling algorithms only strive to maximize the QoS-level for each new task on the basis of satisfying timing constraints. These algorithms do not consider the adaptability. SAQA achieved adaptability, that is, it able to result in some tasks not being accepted when the system is in heavy load.

The QoS-aware real-time systems must incorporate inherent high reliability features in case of hard real time system. Since the automated flight control system is used in the military battle field, the system must ensure that each task is executed within its deadline even in the presence of hardware or software faults. For the signal processing system, although some tasks missing their deadlines may not result in disaster, the outdated or half-baked processed data may be useless for users, especially in the field of modern information battle. Therefore, the system must guarantee its functional and timing correctness even when faults occur. Such a fault-tolerant mechanism is provided by QAFT [3].

III. QOS-AWARE SCHEDULING ALGORITHMS

Quality of service (QoS) requirements must be addressed in various hard and soft real-time systems Many of real time applications have different quality of service (QoS) requirements including adaptability reliability, security, data accuracy, data quality and so on. Let us study algorithms which provide different quality of services.

A. Self-Adaptive QoS-Aware (SAQA) algorithm

SAQA [1] considers the adaptability for realtime tasks with QoS demands on heterogeneous clusters. When the system is in heavy load, the SAQA algorithm can degrade the QoS levels of new tasks or tasks waiting in local queues of nodes to guarantee high schedulability. The minimum QoS level is acceptable for each task. In contrast, when the system is in light load, SAQA can use slack time to adequately improve the QoS of new tasks.

The centralized scheduler is employed in this scheduler model. In order to improve the scheduling quality, the feedback in the centralized scheduler is added as shown in figure 2.

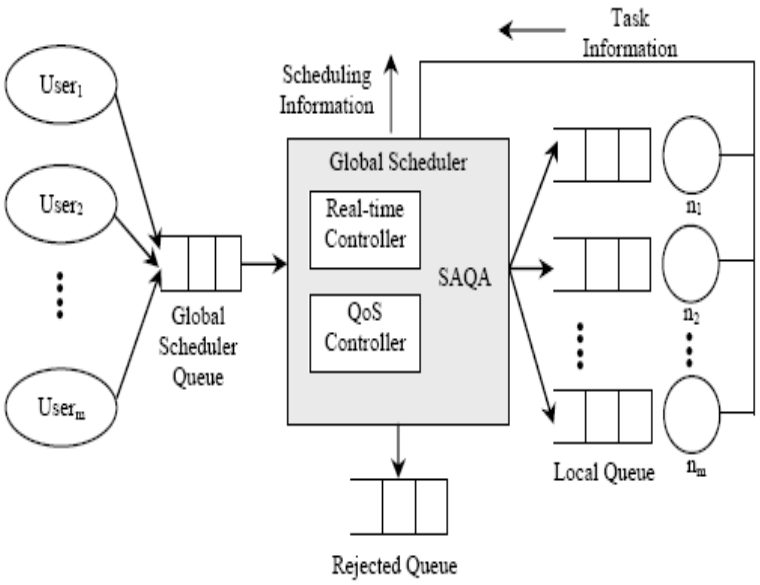


Figure 2: Scheduler model for SAQA

In modeling the QoS requirements of tasks, suppose there are u groups of QoS choices $G=\{g_1, g_2,\ldots,g_u\}$. In the same group , different methods could provide same QoS guarantee but with different QoS levels due to distinct mechanism used. SAQA uses property stated as – if a task can be allocated to a node, it must finish before its deadline, and those waiting tasks whose execution orders (sequence) are later than that of new task in local queue must satisfy their deadlines, as well. The SAQA algorithm. SAQA is a heuristic algorithm that combines the non-preemptive EDF (Earliest Deadline First) algorithm and RF (Response First) algorithm. When a new task arrives, the maximal quality admission test is performed, that is the task is given the maximal QoS level and is inserted into the local queue of a node by the deadline earliest first policy on condition that the task has earliest finish time on this node. If this test can guarantee the timing constraints of the new task and tasks whose execution sequences are later than that of the new task in the same node, allocate the task to the found node. Otherwise, degrade the QoS level of each QoS requirement by the Round-Robin policy until it can be allocated. If the new task getting the minimal QoS level still cannot be allocated, select a node where the sum of QoS levels of tasks in its local queue is largest and then degrade these tasks' QoS levels using Round-Robin method. If all the levels of these tasks being degraded to minimal still miss the deadline of the new task or violate the timing constraints of tasks whose execution sequences are later than that of the new task, it is rejected, or it is allocated to the node.

B. QoS-Aware Fault Tolerant (QAFT) algorithm

QAFT [3] is a non-preemptive fault-tolerant scheduling of real-time aperiodic tasks with QoS needs on heterogeneous clusters. The new dynamic algorithm allows a heterogeneous cluster to tolerate one node's failure.

Dynamic fault tolerant scheduling specially for hard real time uses *primary backup (PB) model*. In the PB model, two copies (primary and backup copy) of one task are scheduled on two different nodes. PB scheme may be active or passive. Back up copy of a task is permitted to execute only if the fault occurs in the primary task. This strategy is referred as passive backup scheme. Whereas, in active backup copy scheme primary and backup copies are executed simultaneously or concurrently. Given a dynamically changing load, our algorithm adaptively switches between the active backup-copy scheme and passive backup-copy scheme. To achieve high flexibility and schedulability, the algorithm maximizes the QoS benefits of real-time tasks by adjusting the tasks' QoS levels.

Following are the assumptions made by algorithm regarding fault model-

1. At one time instant, only one node can be failed.
2. Faults can be transient or permanent and are independent, i.e., a fault on one node will not affect the other nodes.
3. There exists a fault-detection mechanism, such as acceptance tests, to detect node failures. The scheduler will not schedule tasks to a known faulty node any more.

QAFT focus on the centralized scheduler because it is straightforward to design a centralized fault-tolerant scheduler using a backup scheduler that concurrently executes with the primary scheduler as shown in figure 3.

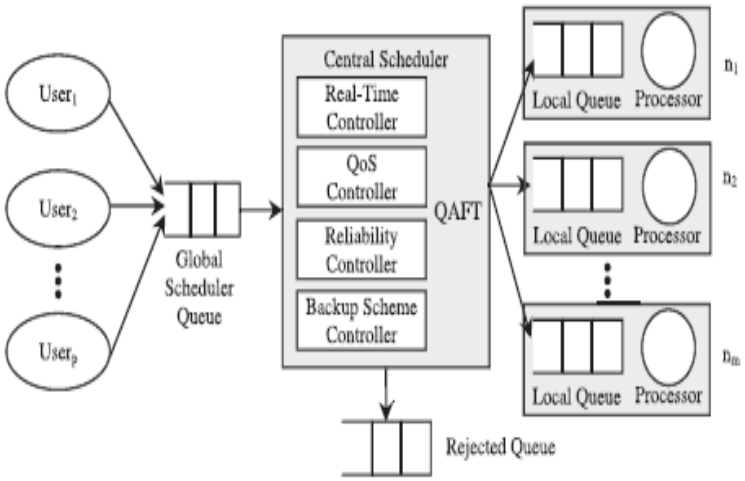


Figure 3: Scheduler model for QAFT.

QAFT algorithm that efficiently considers the QoS needs, system reliability, system resource utilization and schedulability. To understand the QAFT algorithm we should understand following properties:

1. Primary copy and backup copy of a real-time task cannot be allocated to the same node, or when the node meets a fault, both copies will not be successfully finished, thus failing to realize fault tolerance.
2. If a real-time task can be accepted, its primary copy and backup copy must satisfy timing constraints. Also, the primary copy and backup copy of a real-time task may select different QoS levels, which will greatly improve the system's flexibility and schedulability.

At the beginning, the primary copy t_i^P is destined for maximal QoS level. In order to make the corresponding backup copy t_i^B employ passive scheme, t_i^P should execute as earlier as possible to compute the earliest start time est_{ij}^P of t_i^P on node n_j. If t_i^P can be allocated to n_j, it must finish within its deadline. If t_i^P can execute on several nodes, it can select the node which makes the system reliability maximal. At this point, backup copy t_i^B has not been scheduled yet, so its reliability cost of t_i^B is set to zero. If some nodes make the system have identical reliability to accommodate t_i^P, the node on which the start time of t_i^P is earliest must be selected. This operation strives to make t_i^B use passive scheme. If employing a higher QoS level cannot satisfy the timing constraint of t_i^P on any node, degrade one QoS level and select a feasible node again. Otherwise, exist the loop, thus it rejects t_i^P or allocates t_i^P.

The objectives of primary copy allocation in QAFT is to allocate primary copies to the node which makes the system reliability maximal, and to make the QoS levels of primary copies maximal within the timing constraints.

First, the backup copy t_i^B is destined for maximal QoS level. In order to make t_i^B employ passive scheme, t_i^B should begin to execute as late as possible, thus we can calculate the latest start time lst_{ij}^B of t_i^B on node n_j. If t_i^B can be allocated to n_j, t_i^B must finish within its deadline. If the latest start time of t_i^B is later than or equal to the finish time of t_i^P, t_i^B is capable of employing passive scheme. As a result, t_i^B does not need to use active scheme on other nodes any more. If t_i^B can execute using active scheme on some nodes, select the node on which the allocation makes the system reliability maximal. If the latest start time of t_i^B is earlier than the finish time of t_i^P, t_i^B must execute by active scheme. In this condition, the simultaneous execution time of t_i^P and t_i^B should be as little as possible, so select the node on which t_i^B has the latest start time. If t_i^P and t_i^B cannot satisfy timing constraint or minimal QoS level requirement on any node, reject t_i^B and its corresponding primary copy t_i^P, then otherwise allocate t_i^B.

C. TWO-PHASE SCHEDULING STRATEGY (TPSS):

TPSS [5] takes timing constraints and security needs into consideration for security-critical real-time applications. TPSS includes two algorithms DSRF and FMSL. The former aims at enhancing the guarantee ratio when the system is in heavy burden and improving quality of security when the system is in light burden. The latter focuses on making all accepted tasks have small difference in security levels and further improving these tasks' security levels on the whole.

As shown in figure 4, the real-time scheduler is a global scheduler. When a new task arrives, the real-time scheduler collects the feedback information that includes the remaining time of running tasks and waiting time of tasks waiting in local queues so as to determine whether or not the new task can be allocated to a certain node according to the DSRF algorithm located on it. If the new task cannot be allocated, then it will be dropped into the rejected queue, otherwise it will be transferred to a destination node. The global scheduler transfers the tasks' scheduling information to the node for further processing. The scheduling information includes execution orders as well as selected security services of the new task and tasks waiting in this node's local queue. After this operation, the security level (local) controller situated in the destination node begins to work. It makes an effort to balance the security levels of tasks waiting in this node's local queue within the constraint that the finish time of the last task cannot be delayed.

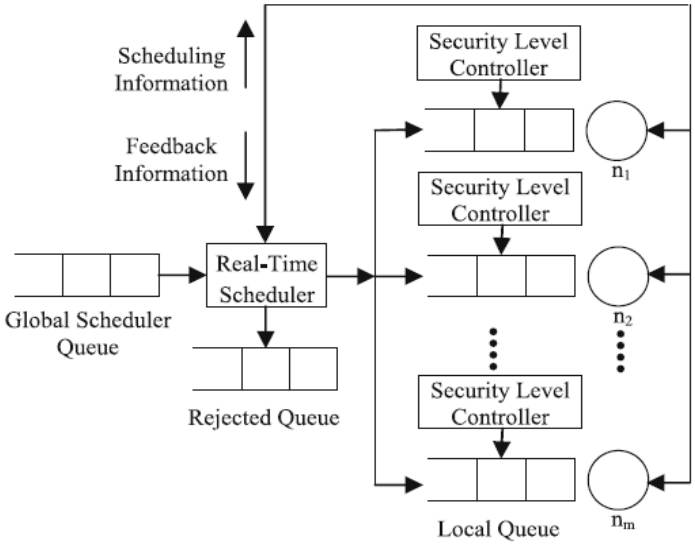


Figure 4: Scheduler model for TPSS

Many security services can be provided for a task. We categorize these security services into groups. In the same group, different security services provide the same type of security but different quality due to the different mechanism used, e.g., SEAL, RC4, DES are all cryptographic algorithms but their security service qualities are different. Suppose there are q groups of security services G ={g_1, g_2, . . . , g_q}. s_{ij} indicates the security level of the ith service in group g_j. A task can select different security services from each group to combine as an integrated security level.

Consider the three security services confidentiality, integrity, and authentication in our security overhead model. Suppose task t_i requires the kth security service in group g_l, if t_i is allocated to node n_j , the security overhead can be described as c_{ij} (S_{kl}).

Let g_1, g_2 and g_3 are confidentiality service, integrity service, and authentication service, respectively. Table1 gives the security level of each security service and its corresponding performance (evaluated on a 90 MHz Pentium machine), where μ_{k1}, μ_{k2} and μ_{k3} represent the performance of the kth security service in group g_1, g_2 and g_3 respectively. The computation overhead caused by encryption and integrity mainly depend on the cryptographic algorithms and hash functions for integrity and the size of data to be protected as well as node processing power on heterogeneous clusters. However, for authentication, the computation overhead only relies on the authentication techniques used and node processing power

TABLE I
SECURITY GROUPS AND LEVELS

Group	**cryptographic algorithms**	**S_{k1}**	**μ_{k1} : KB/ms**
cryptographic algorithms for confidentiality	SEAL	0.08	168.75
	RC4	0.14	96.43
	Blowfish	0.36	37.5
	Knufu/Khafre	0.40	33.75
	RC5	0.46	29.35
	Rijndael	0.64	21.09
	DES	0.90	15
	IDEA	1.0	13.5
hash functions for integrity	hash functions	S_{k2}	μ_{k3} : KB/ms
	MD4	0.18	22.90
	MD5	0.26	17.09
	REPEMD	0.36	12.00
	RIPEMD-128	0.45	9.73
	SHA-1	0.63	6.88
	RIPEMD-160	0.77	5.69
	Tiger	1.00	4.36
Authentication Methods	Authentication methods	S_{k3}	μ_{k3} : ms
	HMAC-MD5	0.55	90
	HMAC-SHA-1	0.91	148
	CBC-MAC-AES	1.00	163

TPSS includes two algorithms DSRF and FMSL. Let us see their workings.

I) DSRF algorithm: The Deadline Sorted Response First (DSRF) algorithm is based on Response First (RF) algorithm that selects the node on which a new arrival task has the earliest finish time. DSRF algorithm consists of three steps as follows.

1. when a new task arrives, the maximal quality admission test is performed. The test is that the task is given the maximal security level and is inserted into the local queue of a node by the deadline earliest first policy on condition that the task has earliest finish time on this node. Step 1 tries to maximize the security level of the new task.
2. If the timing constraints cannot be satisfied on any node, degrade security level of the new task until it can be allocated on a node on the basis of satisfying the constraints. Step 2 also strives to improve the security level of a new task on the premise of accepting it.
3. If step 2 cannot allocate the new task, step 3 select a node where the sum of security levels of tasks in its local queue is largest and then degrade these tasks' security levels using Round-Robin method. If all the levels of these tasks being degraded to minimal security level still miss the deadline of the new task or violate the timing constraints of tasks whose execution orders are later than that of the new task, it is rejected, otherwise it is allocated to the node. Step 3 is used to enhance the guarantee ratio.

II) FMSL algorithm: To make the accepted tasks have fair security services, after a new task is allocated to a node, FMSL will further deal with the tasks in local queues. Fairness of security levels on node n_j can be evaluated by the standard deviation of security levels FLj. The goal of FMSL is to minimize FL_j.

The fair and maximal security levels (FMSL) algorithm is to balance the security levels and to make the total security levels higher than before. The core idea is to degrade the security levels of tasks with relatively maximal security levels so as to enhance the security levels of tasks with relatively minimal security levels. When a new task is allocated to the local queue of one node, copy the tasks in this local queue with relatively maximal security levels and minimal security levels into set maxSet and set minSet, respectively. Degrade the security level of a task in maxSet to enhance the security level of a task in minSet with the constraints that tasks in this local queue must meet their deadlines and the total finish time cannot be delayed.

IV. PERFORMANCE EVALUATION

Let us evaluate performance of node number for SAQA, SQFT and TPSS. Comparison is based on the of following metrics:

1. Guarantee Ratio defined as
   GR=(Total number of tasks guaranteed to meet their deadlines / Total number of tasks) × 100%;
2. QoS Level Average used to test the QoS levels of accepted tasks.

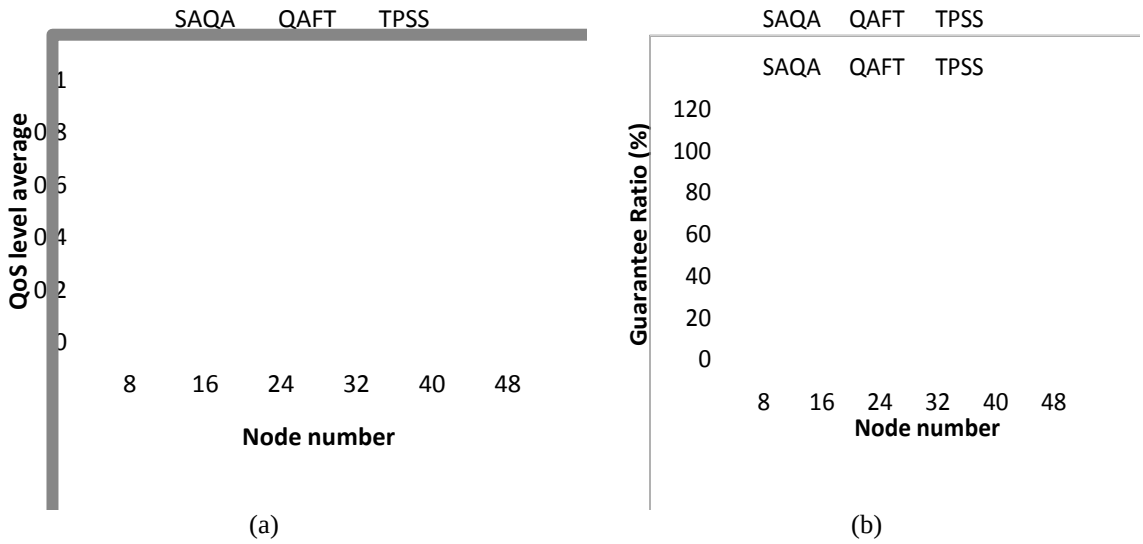


Figure 5: Performance impact of node number

Note that values in graph are in range and not exact values. As shown in figure 5 (a), QoS level average of SAQA is less as compared to TPSS as number of nodes increases till 56. That is because when the system is in heavy load, schedulability is the main objective in SAQA. We also find that when the node number is 56, the guarantee ratio of SAQA is 100 percent. This can be explained that the system is in light load, so SAQA can guarantee higher QoS level for every accepted task. Among the algorithms, QoS level average of TPSS is highest and guarantee ratio of SAQA is highest. Performance of QAFT is lower than SAQA and TPSS for both- guarantee Ratio and QoS level average.

Of course, algorithms have different QoSs that is the different matter.

V. CONCLUSION

This paper gives study of three QoS-aware real time task scheduling algorithms, namely-SAQA, QAFT and TPSS. SAQA is self adaptive algorithm with QoS-requirement of data quality algorithm with QoS-requirement of data quality. QAFT algorithm with QoS requirement of reliability that can tolerate one node's permanent failures and improve system reliability of clusters. TPSS with QoS-requirement of security for security-critical real-time applications.

This paper also evaluated the performance of node number for SAQA, SQFT and TPSS and compare on the basis metrics- guarantee Ratio and QoS level average. Among the algorithms, QoS level average of TPSS is highest and guarantee ratio of SAQA is highest. Of course all algorithms are best in providing QoS that they claim (i.e. SAQA for data quality, QAFT for reliability and TPSS for security).

REFERENCES

[1] Xiaomin Zhu, Jianghan Zhu, Manhao Ma, and Dishan Qiu, "SAQA: A Self-Adaptive QoS-Aware Scheduling Algorithm for Real-Time Tasks on Heterogeneous Clusters",2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 224-232, 2010.

[2] Jane W.S. Liu, Real-Time Systems, PEARSON, 2011.

[3] Xiaomin Zhu, Xiao Qin and Meikang Qiu, "QoS-Aware Fault-Tolerant Scheduling for Real-Time Tasks on Heterogeneous Clusters", IEEE Transactions on Computers, vol. 60, NO. 6, pp. 800-812, JUNE 2011

[4] Haluk Topcuoglu, Salim Hariri, Min-You Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing", IEEE transactions on parallel and Distributed systems, Vol. 13, No. 3,pp. 260-274, 2002.

[5] X. Zhu and P. Lu, "A Two-Phase Scheduling Strategy for Real-Time Applications with Security Requirements on Heterogeneous Clusters," Computers and Electrical Engineering, vol. 35, pp. 980-993, Nov. 2009.

[6] C. M. Krishna, K. G. Shin, Real-Time Systems, USA:McGraw-Hill, 2001.

[7] T. Xie and X. Qin, "Scheduling Security-Critical Real-Time Applications on Clusters", IEEE Trans. Computers, vol. 55,no. 7, pp. 864-879, Jul. 2006.

[8] X. Qin and H. Jiang, "A Dynamic and Reliability-Driven Scheduling Algorithm for Parallel Real-Time Jobs Executing on Heterogeneous Clusters", J. Parallel and Distributed Computing, vol. 65, no. 8, pp. 885-900, Aug. 2005.

[9] V. Subramani, V., R. Kettimuthu, S. Srinivasan, J. Johnston, and P. Sadayappan, "Selective buddy allocation for scheduling parallel jobs on clusters", Proc. IEEE Intl Conf. Cluster Computing (Cluster 2002), pp. 107 C 116, Sept. 2002.

[10] X. Qin and H. Jiang, "A Novel Fault-Tolerant Scheduling Algorithm for Precedence Constrained Tasks in Real-Time Heterogeneous Systems," J. Parallel Computing, vol. 32, no. 5, pp. 331-356, Aug. 2006.

[11] Renan Starke and Romulo Silva de Oliveira, "A Heterogeneous Preemptive and Non-preemptive Scheduling Approach for Real-time Systems on Multiprocessors", 2012 Second Brazilian Conference on Critical Embedded Systems, pp. 70-75, 2012.

[12] L. He, S.A. Jarvis, and D.P. Spooner, "Dynamic Scheduling ofParallel Jobs with QoS Demands in Multiclusters and Grids," Proc. Fifth IEEE/ACM Int'l Workshop Grid Computing (Grid '04), pp. 402-409, Nov. 2004.

[13] Silberschatz, Galvin and Gagne,Operating System Concepts, 7th Edition, 2005.

Kalpana Manudhane: Received the B.E in computer engineering from North Maharshtra University, SSB's COET, Bambhori, Jalgaon (khan) in 2005. She is currently an Assistant Professor in CSE department .in COET, Akola from SGBAU University. She is pursuing for ME CSE from G.H. Raisoni, Amravati. Her research interest include task scheduling, compiler techniques, operating system and programming.

Prof. Avinash P.Wadhe: Received the B.E and from SGBAU Amravati university and M-Tech (CSE) From G.H Raisoni College of Engineering, Nagpur (an Autonomous Institute). He is Currently an Assistant Professor with the G.H Raisoni College of Engineering and Management,Amravati SGBAU Amravati university. His research interest include Network Security , Data mining and Fuzzy system .He has contributed to more than 20 research paper. He had awarded with young investigator award in international conference**.** .

