A QoS-Based Self-Adaptive Scheduling Algorithm for Real-Time Tasks on Heterogeneous Clusters

Xiaomin Zhu *, Peizhong Lu
School of Computer Science,
Fudan University, Shanghai, P. R. China 200433
{*xmzhu, pzlu*}*@fudan.edu.cn*

Abstract

Nowadays, providing quality of service (QoS) guarantees for some applications such as signal data processing has become a critical issue. In this paper, we propose a novel QoS-based self-adaptive scheduling algorithm called QBSA that sufficiently considers the adaptability for real-time tasks with QoS demands on heterogeneous clusters. When the system is in heavy load, the QBSA algorithm can degrade the QoS levels of new arrival 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, QBSA can use slack time to adequately improve the QoS of new arrival tasks. We compare QBSA with SADEF algorithm by simulations. The experimental results indicate that QBSA has admirable adaptability while providing timing and QoS guarantees.

1. Introduction

Clusters are considered as an attractive medium for cost effective parallel computing. Specially, heterogeneous clusters receive a good deal of attention in practice because recently purchased machines (nodes) and old ones are usually assembled in a cluster for economical purpose [1].

Nowadays, diverse real-time applications with QoS requirements have been developed in clusters. 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 [2]. Authentication service, confidentiality service and integrity service can be supported to provide security for tasks and data [4]. In each service, multiple methods can be adopted. For instances, SEAL, RC4, and RC5, are all cryptographic algorithms for confidentiality but their security service qualities are different due to the different mechanism used.

Scheduling algorithms play a key role in obtaining a high performance in cluster computing [3]. Unfortunately, most existing QoS-based real-time scheduling algorithms only strive to maximize the QoS level for each arrival task on the basis of satisfying timing constraints without considering the adaptability, which may result in some tasks cannot be accepted when the system is in heavy load.

Many practical instances of scheduling algorithms have been found to be NP-complete [5]. So heuristic approaches are widely used to solve scheduling problems. Nowadays, increasing attention has been directed to the problem of QoS-based real-time scheduling. Xie and Qin proposed an algorithm named SAEDF to schedule real-time tasks with security requirements [2]. The SAEDF algorithm strives to maximize the QoS level of each new task based on satisfying its timing constraint resulting in the schedulability of tasks would be low. Abdelzaher et al. presented a novel scheme for QoS negotiation in real-time applications [6]. This scheme uses the batch mode, namely arriving tasks are collected and scheduled as a meta-task when a scheduling event is triggered. So, this QoS negotiation scheme focuses mainly on long-lived services that need to hold reserved resources for an extended period of time. In our algorithm, we exploit the immediate mode (schedule a task once it arrives). It is noted that both algorithms are designed for homogeneous systems, making them unsuitable for heterogeneous clusters [7]. As a result, we are motivated in this study to propose a novel scheduling algorithm to self-adaptively improve the QoS of real-time tasks running on heterogeneous clusters according to the system load.

In this paper, we propose a QoS-based self-adaptive scheduling algorithm QBSA, which takes both the timing and QoS needs into account for real-time, independent, and aperiodic tasks on heterogeneous clusters. In our study, tasks arrive dynamically, so our scheduling algorithm is dynamic. In addition, the QBSA algorithm is non-preemptive, which is more efficient, particularly suitable for soft real-time applications than the preemptive approaches due to reducing the overhead needed for switching among tasks [8].

[0]* Xiaomin Zhu now is a Ph. D candidate in Fudan University .

The QBSA algorithm is able to adjust the priority of scheduling objectives according to the system load. When the system is in heavy load, the high guarantee ratio is the main goal. On the other hand, when the system is in light load, high QoS for accepted tasks is the major objective.

2. System model

2.1. Scheduler model

Figure 1 shows the scheduler model. When a new task arrives, the global scheduler collects the information of tasks running on nodes and tasks waiting in local queues to calculate the earliest start time of the new task on each node and decides whether the new task can be allocated or not. If the new task cannot be scheduled, it will be dropped into the rejected queue, otherwise it will be transferred to a destination node. After a new task is allocated to a node, the global scheduler will transfer the tasks' scheduling information to the node. The scheduling information includes execution sequences, selected QoS levels of the new task and tasks waiting in this node's local queue.

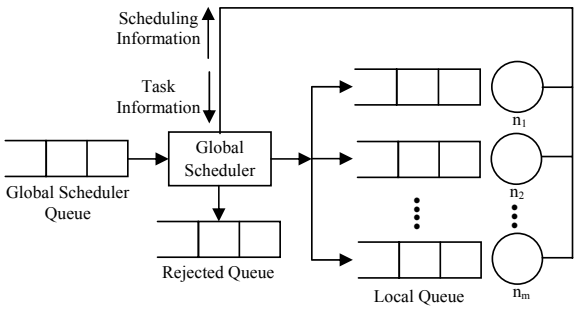


Figure 1. Scheduler Model.

2.2. Task model with QoS requirements

$T = \{t_1, t_2, ..., t_n\}$ is a set of independent real-time tasks. $N = \{n_1, n_2, ..., n_m\}$ is a set of nodes. The execution time is denoted by a matrix $E = (e_{ij})_{n\times m}$, where element e_{ij} denotes the execution time of task t_i on node n_j. a_i, d_i, and f_i represent the arrival time, deadline, and finish time of task t_i, respectively. Let $EST = (est_{ij})_{n\times m}$ be an earliest start time matrix of tasks, where element est_{ij} denotes the earliest start time of task t_i on node n_j. $S = (s_{ij})_{n\times m}$ is an execution sequence matrix, where s_{ij} denotes the execution sequence of task t_i on node n_j . Let $Z = (z_{ij})_{n\times m}$ be a binary matrix, where element $z_{ij} = 1$ if and only if t_i has been assigned to n_j; otherwise $z_{ij} = 0$. Let QoS levels be a set Q, $Q = \{q_1, q_2, ..., q_k\}$, where $q_1 < q_2 < ... < q_k$.

X_i denotes all possible schedules for task t_i, and $x_i \in X_i$ is a scheduling decision of t_i. The QoS level of task t_i adopting schedule x_i can be represented by $q(x_i)$. x_i is a feasible schedule if (1) deadline of d_i can be guaranteed, i.e., $f_i \le d_i$; (2) the QoS requirement is satisfied, i.e., $q_1 \le q(x_i) \le q_k$. On the promise of tasks being accepted, the following function needs to be maximized.

$$\max_{x_i\in X_i} \left\{ \sum_{j=1}^{m}\sum_{i=1}^{n} z_{ij}q(x_i) \Big/ \sum_{j=1}^{m}\sum_{i=1}^{n} z_{ij} \right\}. \quad (1)$$

3. QBSA algorithm

QBSA is a self-adaptive 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 time 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 till 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.

Property 1. If task t_i can be allocated to node n_j, the following inequalities must be satisfied.

$$est_{ij} + e_{ij}\big(q(x_i)\big) \le d_i, \quad (2)$$

$$\forall t_k, s_{kj} > s_{ij} : est'_{kj} + e_{kj}\big(q(x_k)\big) \le d_k, \quad (3)$$

where $est'_{kj} = est_{kj} + e_{ij}\big(q(x_i)\big)$. The earliest start time est_{ij} can be calculated as follows:

$$est_{ij} = a_i + \sum_{s_{kj}<s_{ij}, w_k=1} \Big(e_{kj}\big(q(x_k)\big)\Big) + r_j, \quad (4)$$

where r_j is the remaining execution time of the running task on n_j. if task t_k is waiting in a local queue, $w_k = 1$, else $w_k = 0$.

Property 2. If the QoS level of task t_l waiting in the local queue of node n_j needs to be degraded for accepting new task t_i, the earliest start time est_{lj} must be recalculated.

Case 1: $s_{lj} < s_{ij}$, the new earliest start time est'_{lj} can be recalculated as:

$$est'_{lj} = est_{lj} - \sum_{s_{kj}<s_{lj}} \Big(e_{kj}\big(q(x_k)\big) - e_{kj}\big(q(x_k)\big)'\Big). \quad (5)$$

Table 1. Parameters for simulation studies

Parameter	Value(Fixed)-/(Min, Max, Step)
node Number	(16)-(8, 56, 8)
task Number	(6400)
$powerAverage$	(700)
$powerSpan$	(400)-(50, 650, 100)
$hardnessAverage$	(190)
$hardnessSpan$	(100)
$baseDeadline$	(100)
$baseTime$	(60)
$intervalTime$	(1)-(0.5, 3.0, 0.1)

Case 2: $s_{lj} > s_{ij}$, the est'_{lj} can be recalculated as:

$$est'_{lj} = est_{lj} + e_{ij}\big(q(x_i)\big) - \sum_{s_{kj}<s_{lj}} \Big(e_{kj}\big(q(x_k)\big) - e_{kj}\big(q(x_k)\big)'\Big), \quad (6)$$

where $e_{kj}\big(q(x_k)\big)'$ denotes the new execution time while t_k's QoS level is degraded. Note that after the new earliest start time is recalculated, Property 1 must be satisfied.

The pesudocode of QBSA is given in Figure 2.

4. Performance evaluation

We compare QBSA with SADEF in these metrics: (1) Guarantee Ratio (GR), (2) QoS Level Average (QLA) and (3) Overall Performance (OP, $OP = GR * QLA$ [2]).

4.1. Simulation method and parameters

(1) p_j a positive real number to represent the processing power of node n_j. p_j is uniformly distributed between $powerAverage - powerSpan$ and $powerAverage + powerSpan$.

(2) h_i a positive real number to denote the hardness of task t_i. h_i is uniformly chosen between $hardnessAverage - hardnessSpan$ and $hardnessAverage + hardnessSpan$.

(3) $e_{ij} = (1 + q(x_i)/10) \times baseTime \times (h_i/p_j)$. where $0 \le q(x_i) \le 9$. Parameter $baseTime$ is a random positive real number.

(4) d_i is calculated as follows: $d_i = a_i + \max\{e_{ij}\} + baseDeadline$. $baseDeadline$ is a random positive real number.

(5) a_i is described as: $a_i = a_{i-1} + intervalTime$. $intervalTime$ is a random positive real number.

Table 1 gives the simulation parameters and their values.

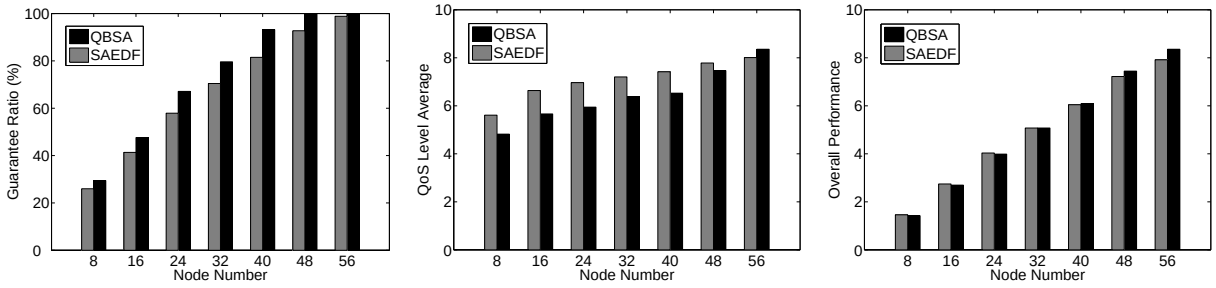
```
1. for each new arrival task t_i do
2.    find ← false; q(x_i) ← q_k;
3.    while q(x_i)! = q_min do
4.       find the node on which t_i has the earliest finish time;
5.       if Property 1 is satisfied then
6.          find ← true; break;
7.       else
8.          degrade one QoS level by Round-Robin policy;
9.       end if
10.   end while
11.   if find == false then
12.      select node n_l on which the sum of QoS levels of
         tasks waiting in its local queue is largest;
13.      put these task into a empty set S;
14.      while find! = true && S! = ∅ do
15.         for each task t_k on node n_l do
16.            if q(x_k)! = q_1 then
17.               degrade one QoS level by Round-Robin policy;
18.               calculate new EST according to Property 2;
19.               if Property 1 is satisfied then
20.                  find ← true; break;
21.               end if
22.            else
23.               remove t_k from S;
24.            end if
25.         end for
26.      end while
27.   end if
28.   if find == true then
29.      allocate t_i to node n_l;
30.      update EST of tasks in the local queue of n_l;
31.   else
32.      reject task t_i;
33.   end if
34. end for
```

Figure 2. The pseudocode of QBSA.

4.2. Performance impact of node number

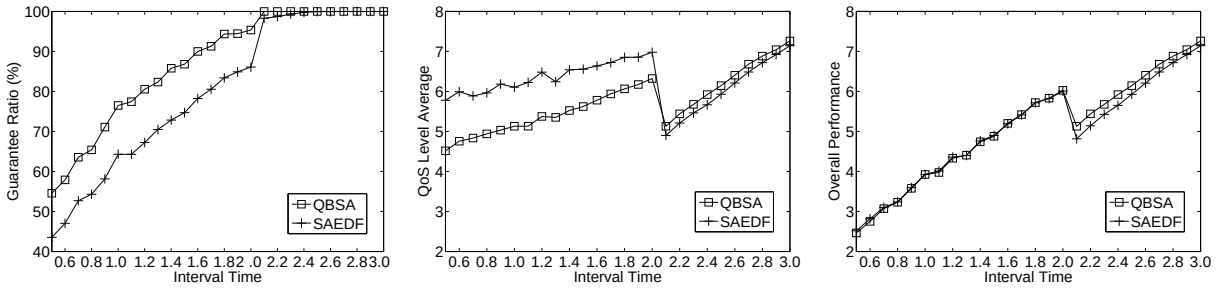
Figure 3(a) shows that QBSA always has higher guarantee ratio than SAEDF when the node number is less than 48. That is because when the system has heavy load, schedulability is the main objective in QBSA. However, SAEDF selects the maximal QoS level for every new accepted task with the timing constraints, which will result in the new task has longer execution time, so the tasks arrive later have later start time. If the start time is delayed, the probability of missing their deadlines increases.

We observe from Figure 3(b) that the QLA of SAEDF is higher than that of QBSA when the node number is less than 48. That is because the SAEDF strives to maximize the QoS levels of all accepted tasks. But SAEDF gets higher QoS level at the cost of guarantee ratio. From Figure 3(a) and Figure 3(b), we also find that when the node number



(a) Guarantee ratio (b) QoS level average (c) Overall performance

Figure 3. Performance impact of node number.



(a) Guarantee ratio (b) QoS level average (c) Overall performance

Figure 4. Performance impact of arrival rate.

is 56, the guarantee ratio of QBSA is 100 percent and the QoS level average is higher than that of SAEDF. This can be explained that the system has light load, so QBSA can guarantee higher QoS level for every accepted task.

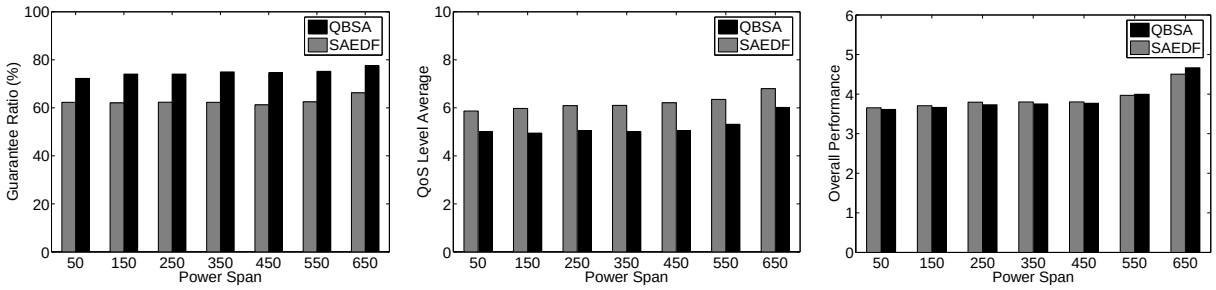
The observation from Figure 3(c) is that when the node number is smaller, QBSA has almost the same overall performance as SAEDF but when the node number is larger, QBSA improves the overall performance than SAEDF. This results can be easily explained that when the node number increases, the system has light load, QBSA provides higher QoS level while guaranteeing higher guarantee ratio.

4.3. Performance impact of arrival rate

Figure 4(a) shows that when $intervalTime$ is smaller, which makes more tasks wait in the local queues of nodes, so some tasks arriving later have later start time and miss their deadlines. With the increase of $intervalTime$, the number of tasks waiting in local queues decreases resulting in tasks have earlier start time. So, the guarantee ratio increases. Figure 4(a) also shows that QBSA has obviously higher guarantee ratio than SAEDF when $intervalTime$ is less than 2.1. That is because QBSA degrades the QoS levels of tasks waiting in local queues to decrease the total execution time of these tasks, so more tasks can be accepted for earlier start time.

Figure 4(b) shows that when the $intervalTime$ is increased from 0.5 to 2.0, SAEDF performs better than QBSA in QoS level average. That is because the SAEDF is only in pursuit of higher QoS level of new arrival task, regardless of the tasks arriving later. On the contrary, QBSA adopts a self-adaptive scheduling policy striving to accept the new tasks by decreasing the QoS levels of tasks waiting in the local queue of the same node if the new task misses its deadline using minimal QoS level. However, when the value of $intervalTime$ is more than 2.0, the condition is just the opposite, QBSA has higher QoS level average than SAEDF. This result can be explained that when tasks arrive slowly, QBSA does not need to accept a new task by degrading the QoS levels of tasks in the same local queue any more, and the QoS becomes a major objective. An interesting observation from Figure 4(b) is that when $intervalTime$ varies from 2.0 to 2.1, the QoS level average of both QBSA and SAEDF decrease suddenly. The reason is that the task heterogeneity resulting in some tasks (long tasks) have longer execution time, these tasks cannot be accepted until the value of $intervalTime$ is large enough.

Figure 4(c) exhibits although QBSA has slightly improvement in overall performance, the self-adaptivity greatly outperforms that of SAEDF.



(a) Guarantee ratio (b) QoS level average (c) Overall performance

Figure 5. Performance impact of node heterogeneity.

4.4. Performance impact of node heterogeneity

Figure 5(a) plots that QBSA basically keeps the same guarantee ratio with the increase of node heterogeneity. This result proves that QBSA is more suitable for heterogeneous clusters than SAEDF. From Figure 5(a), we can find that QBSA always has higher guarantee ratio than SAEDF although the node heterogeneity varies greatly. We attribute the improvement to the fact that QBSA considers the schedulability as its main goal when the node number is less or tasks arrive quickly.

Figure 5(b) shows that SAEDF has higher QoS level average than QBSA, but the improvements is not rational for real-time applications because the guarantee ratio is decreased observed from Figure 5(a). Figure 5(b) also reveals that the performance impact of node heterogeneity for both algorithms are smaller and the impact of QBSA is slighter than that of SAEDF.

Figure 5(c) reveals that the overall performances of QBSA and SAEDF are basically same regardless of the node heterogeneity on the basis of the node number is smaller and tasks arrive quickly, which proves that QBSA is significantly suitable for heterogeneity clusters.

5. Conclusions

In this paper, we proposed a novel self-adaptive scheduling algorithm QBSA for real-time tasks with QoS needs on heterogeneous clusters. The QBSA algorithm produces higher guarantee ratio if the system is in heavy load and higher QoS level if the system is in light load than the existing algorithm SAEDF. The experimental results indicate that QBSA is more suitable for heterogeneous cluster where the arrival rate varies largely, tasks are in higher heterogeneity and some nodes dynamically join or quit the cluster.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (Grant No. 60673082), and the Special Funds of Authors of Excellent Doctoral Dissertation in China (Grant No. 200084).

References

[1] X. Zhu and P. Lu, "Study of Scheduling for Processing Real-Time Communication Signals on Heterogeneous Clusters," *Proc. 9th int'l Symp. Parallel Architectures, Algorithms, and Networks (I-SPAN 2008)*, pp. 121-126, May 2008.

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

[3] Y. Zhang, A. Sivasubramaniam, J. Moreira, and H. Franke, "Impact of Workload and System Parameters on Next Generation Cluster Scheduling Mechanisms," *J. IEEE Trans. Parallel and Distributed Systems, vol. 12, no. 9*, pp. 967-985, Sept. 2001.

[4] C. Irvine and T. Levin, "Toward a Taxonomy and Costing Method for Security Services," *Proc. 15th int'l conf. Computer Security Applications (ACSAC 2006)*, pp. 183-188, Dec. 1999.

[5] J. D. Ullman, "NP-Complete Scheduling Problems," *J. Computer and System Sciences, vol. 10, no. 3*, pp. 384 -393, Oct. 1975.

[6] T. F. Atdelzater, E. M. Atkins and K. G. Shin, "QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control," *IEEE Trans. Computers*, vol. 49, no. 11, pp. 1170-1183, Nov. 2000.

[7] 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.

[8] W. Li, K. Kavi and R. Akl, "A Non-Preemptive Scheduling Algorithm for Soft Real-Time Systems," *J. Computers and Electrical Engineering, vol 33, no. 1*, pp. 12-29, Jan. 2007.