Dynamic Load Balancing Algorithm in the Global-Local Order

Hojiev Sardor Qurbonboyevich[1], Tae-Young Choe[2]
[1](Dept. of IT Convergence Engineering/ Kumoh National Institute of Technology, South Korea)
[2](Dept. of Computer Engineering/ Kumoh National Institute of Technology, South Korea)

ABSTRACT : *Most traditional dynamic load balancing schemes for hierarchical environments have applied local load balancing first and have expands it to the global load balancing. The major problem of the approach is that unnecessary task immigrations can occur, which degrade the system performance. Carefully designed global load balancing scheme eliminates the unnecessary task immigrations. We propose a dynamic load balancing scheme that balances global level first followed by local level. Two thresholds that include communication overheads are applied to the load balancing scheme. Experiments show that the proposed scheme 99.7% and 87.2% of average response time than traditional local-first-global-later load balancing scheme, in the case of a uniformly distributed workload and a single hot spot workload, respectively.*

Keywords - *Dynamic Load Balancing, Grid computing, GridSim, Hierarchical distributed System, Multiple Threshold*

I. INTRODUCTION

Grid computing system is an inexpensive way to use high-end computers [1], and it is a fundamental structure to support backbone and infrastructure of Cloud computing [2]. Grid computing system is used in many fields from computation oriented applications to small sized massive queries like Web services. A computation oriented application is designed to be executed in the multiple processors systems. It is divided and allocated to multiple processors or resources in the programming phase or compile phase. Thus computation oriented applications are targets of static scheduling or static load balancing. On the other side, small sized massive queries invoke dynamically changing workloads. Thus, dynamic load balancing schemes are required in the case [3, 4].

According to types of subjects that control the loads, the dynamic load balancing schemes can be classified as *centralized* or *decentralized*. Decentralized schemes have some advantages against centralized schemes like no single point of failure and free from bottleneck. On the other side, centralized schemes have better performance if the given environment is free from faults. In order to manage the entire system using centralized scheme, there should be a subject called a manager that stores statistics, manages meta data, and makes load balancing Since the manager still suffers from a bottleneck in the case of large system even in the fault free environment, a *hierarchical* scheme comes on stage.

In the hierarchical scheme, nearby resources or processors construct a subset. Multiple subsets construct a higher level subset. Such construction repeats until the entire set is constructed. Since the Grid computing system is composed of multiple clusters, a cluster is mapped to a subset and the grid is mapped to the entire set in two-level layers [5].

The objective of load balancing schemes is to maximize the performance based on criteria like makespan, throughput, utilization, or response time. Since the current trend of computing services moves from computation oriented Grid computing to service oriented Cloud computing, we concentrate on minimizing response time of a request task.

When a dynamic load balancing scheme is designed in a two-level hierarchical model, at least the following should be decided:

- Balancing order
- Task immigration decision

Most dynamic load balancing schemes make load balancing using the local-first-global-later order in the hierarchical model [5, 6, 7, 8, 9, 10]. That is, a local subnet like a cluster starts load balancing by its local manager. Next, a global subnet starts load balancing by the global manager. Yagoubi and Meddeber explained the reason of the load balancing order such that the balancing order reduces response time and communication cost [6]. It looks natural that the area of load balancing starts from small area and expands the operations to

larger area. Unfortunately, there was no proof on the insistence of the balancing order.

We noticed that global-first-local-later load balancing order reduces communication overhead compared to local-first load balancing order, if careful task immigration is supported. For example, assume that an overloaded resource S_i and an under-loaded resource S_j are located in an overloaded cluster C_i. The cluster manager of C_i orders to transfer tasks from S_i to S_j in the local load balancing step. Next, the grid manager orders to transfer tasks in cluster C_i to other under-loaded cluster in the global load balancing step. Then some tasks already immigrated from S_i to S_j must immigrate again. The first local immigrations of the tasks come to be unnecessary.

Global-first-local-later load balancing scheme should be more careful on task immigration because the inter-cluster communication costs more than the intra-cluster communication. Global load balancing should consider not only the amount of load difference but also communication overhead. Analytic model for the global-first-local-later load balancing scheme is too complex because of heterogeneity and dynamically changing grid components. GridSim is a useful simulation tool for the purpose [11].

We propose a global-fist load balancing scheme Global Local Load Balancing (GLLB) in two-level hierarchical grid model. Also an immigration decision making condition is presented in order to make efficient operation of GLLB scheme. The rest of this paper is organized as follows: Section 2 describes environment and system model where our proposed scheme is applied. Section 3 summarizes related previous works. Section 4 presents our proposed dynamic load balancing algorithm GLLB. Experimental results are shown in Section 5. Finally, Section 6 concludes this work.

II. SYSTEM MODEL

Since Grid or Cloud computing system consists of multiple clusters, multi-level hierarchical model is natural. Two-level hierarchical cluster model is considered as the system where a load balancing scheme is applied as shown in Fig. 1.

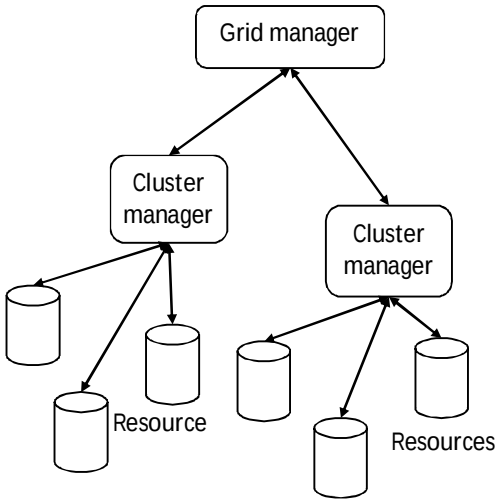

Figure 1 Two-level hierarchical cluster model

The system model is consisted of following entities: resource S_i, cluster manager M_k, and a grid manager *GM*. Multiple resources are connected to a cluster manager located in the same local area network. Multiple cluster managers are connected to a grid manager through wide area networks. A cluster can be described as a set C_k consist of resources in the cluster. Cluster manager M_k takes care of resources in C_k. A grid can be described as a set G consist of cluster managers in the grid.

For each resource, there is a process that stores tasks in its queue and manages statistics for load balancing. The processes can be located in resources if the resources contain a CPU and a memory. Or they can be located in a server where multiple resources are connected by high speed network. Each process represents a resource and a process as an active subject that controls the resource is called a *resource* S_i for notational convenience.

Since communication overheads are different for each inter-cluster and intra-cluster, the latencies and bandwidths are notated as follows:

- λ_k: communication latency within cluster C_k,
- β_k: communication bandwidth within cluster C_k,

- λ : average communication latency between clusters,
- β : Average communication bandwidth between clusters.

For example, if a resource S_i in cluster C_k sends a task of packet size p_i to resource S_j is cluster C_l, the communication overhead of the task immigration is $\lambda_k + \beta_k p_i + \lambda + \beta p_i + \lambda_l + \beta_l p_i$. In order to simplify computation, average communication overheads are used instead of one-to-one communication overheads.

III. PREVIOUS WORKS

Yagoubi et al. proposed a dynamic load balancing scheme HDLA (Heterogeneous Dynamic Load balancing Algorithm) in two-level hierarchical grid model [6]. In the scheme, each worker node estimates the amount of load assigned to the node and sends the cluster manager of the node the amount. Each cluster manager classifies worker nodes according to two thresholds: a lower threshold and an upper threshold. The thresholds are decided by the average and the standard deviation of the amount of loads in the cluster. A node is classified as an *underloaded* if the load of the node is below the lower threshold. A node is an *overloaded* if the load of the node is above the upper threshold. A cluster manager orders overloaded nodes to yield overloaded loads and orders underloaded nodes to accommodate the loads yielded by overloaded nodes.

After the local load balancing, each cluster manager sends the grid manager the total amount of loads in the cluster. The grid manager classifies clusters as underloaded, normally loaded, or overloaded according to two thresholds similar to the local load balancing. The grid manager moves loads from overloaded cluster to underloaded cluster.

HDLA has a major problem that load balancing does not work in the following two situations: first, a group is saturated; second, underloaded node has not sufficient capacity for immigration amount from an overloaded node.

Qurbonboyevich and Choe proposed a dynamic load balancing scheme in two-level hierarchical grid model [12]. Since the scheme balances local level first followed by global load balancing, we call it LGLB (Local-Global Load Balancing) scheme. LGLB scheme proposes an immigration decision that fixes malfunctions of HDLA. When a cluster manager is ordered to send an amount of loads, it proportionally withdraws loads from each worker nodes. Thus, there is no saturated group.

However, LGLB also makes balancing local first and global later. As being explained in the Section 1, HDLA and LGLB scheme can invoke unnecessary communication if a task immigrates during local load balancing and immigrate again during global load balancing at the next time.

IV. PROPOSED SCHEME

Major objective of the proposed scheme is to reduce the communication overhead by reversing load balancing order. Instead of local balancing first, each cluster manager sends the grid manager its statistics. After processing the order from the grid manager, each cluster manager balance in local network if necessary.

A. *Computing Statistics*

Following statistics are maintained in each resource S_i in cluster C_k and are announced to the cluster manager of the cluster periodically:

- $N_k(S_i)$ is the number of tasks running or waiting in the queue of the resource.
- $E_k(S_i)$ is the average execution time of each task assigned to the resource S_i.
- $T_k(S_i)$ is the expected completion time of the last task in the resource S_i. Thus $T_k(S_i) = N_k(S_i) \cdot E_k(S_i)$.
- $P_k(S_i)$ is the average packet size of tasks assigned to the resource S_i. The value is used to measure communication cost of load immigration.

The clocks in resources in a cluster are assumed to be synchronized within a tolerable rate. In other words, a cluster manager does not confuse whether any arrived message is included in which period.

After receiving all messages $(E_k(S_i), N_k(S_i), T_k(S_i), P_k(S_i))$ from resources S_i

in cluster C_k, a cluster manager M_k computes following statistics:

- $N_k = \sum_{S_i \in C_k} N_k(S_i)$. That is, N_k is the number of tasks in cluster C_k.
- $E_k = \sum_{S_i \in C_k} (E_k(S_i) \cdot N_k(S_i)) / N_k$.
- $T_k = \sum_{S_i \in C_k} T_k(S_i) / |C_k|$, where $|C_k|$ is the number of active resources in cluster C_k. Thus T_k is the average completion time of the last tasks in cluster C_k.
- σ_k is the standard deviation of $T_k(S_i)$ for all $S_i \in C_k$.
- $P_k = \sum_{S_i \in C_k} (P_k(S_i) \cdot N_k(S_i)) / N_k$. Thus P_k is the average packet size of tasks assigned to the cluster C_k.

A cluster manager sends a message $(\mathrm{N_k}, T_k, P_k, |C_k|)$ to the grid manager. The grid manager collects all messages and computes grid level statistics as follows:

- $N_{tot} = \sum_{M_k \in G} N_k$.
- $E_{ave} = \sum_{M_k \in G} E_k / N_{tot}$.
- $T_{ave} = \sum_{M_k \in G} T_k / |G|$, where $|G|$ is the number of cluster managers that sent the message.
- σ is the standard deviation of T_k.
- $P_{ave} = \sum_{M_k \in G} (P_k \cdot N_k) / N_{tot}$.

N_{tot} is the total number of tasks in the grid system at the moment, E_{ave} is the average expected execution time of tasks, T_{ave} indicates an average load of the grid system, and P_{ave} is an average amount of total packets.

B. *Classifying Clusters*

The grid manager decides which clusters are overloaded and which clusters are underloaded in order to balance loads between clusters. Various dynamic load balancing algorithms have decided load balancing based on load differences between clusters. Unfortunately, they did not give sufficient attention to other factors like the communication overhead and task execution time. The proposed scheme adds the factors to the thresholds that classify clusters.

Assume that a task is assigned to a resource S_i in cluster C_k as the last task. When the resource computes current completion time as $T_k(S_i)$, the start time of the task is approximated as $T_k(S_i) - E_k(S_i)$ because the execution time of the task is included in the completion time of the resource. When the task is about to immigrate to another resource S_j, the difference of two resource completion time is computed as $T_k(S_i) - E_k(S_i) - T_l(S_j)$. If the difference is less than the communication overhead from S_i to S_j, immigration of the task does not provide any advantage. Thus, not only overhead difference between clusters but also communication overhead should be considered in task immigration decision.

In order to simplify the computation, average communication overhead O_c is defined as follows:

$$O_c = \lambda + \frac{P_{ave}}{\beta} + 2\left(\lambda_{ave} + \frac{P_{ave}}{\beta_{ave}}\right), \qquad (1)$$

where λ_{ave} and β_{ave} are the average communication latency and the average communication bandwidth in intra-cluster communication, respectively. When a task is immigrated, a task should be transfered to gateway of the cluster, be transferred through inter-cluster network, and be transferred to the target resource. Thus, there are two local communication overheads.

Another consideration for cluster classification is the task execution time. If the expected completion time of a resource S_i is $T_k(S_i)$, the start time of the last task in the resource is approximated as $T_k(S_i) - P_k(S_i)$. If the communication overhead is ignored, the completion time of S_i should be greater than that of a target resource in order to profit from immigration. Thus, if S_j is the target resource in cluster C_l, following condition should satisfied:

$$T_k(S_i) - E_k(S_i) > T_l(S_j), \text{ or}$$
$$T_k(\mathrm{S_i}) - T_l(S_j) > E_k(S_i). \qquad (2)$$

By adding communication overhead (Equation 1) to execution time (Equation 2) source resource S_i and target resource S_j should satisfies following condition:

$$T_k(S_i) - T_l(S_j) > E_k(S_i) + O_c. \qquad (3)$$

Load imbalance and communication overhead is combined by selecting the maximum value between a load difference and the half of

communication overhead. Thus, two thresholds that classify clusters are decided as follows:

$$TH_h = T_{ave} + \max\left(\alpha\sigma, \frac{E_{ave}+O_c}{2}\right)$$
$$TH_l = T_{ave} - \max\left(\alpha\sigma, \frac{E_{ave}+O_c}{2}\right) \quad (4)$$

where α is a threshold coefficient in order to control cluster classification range and average expected task execution E_{ave} is used.

Using the Equation 4, the grid manager classifies cluster C_k as follows:

- Cluster C_k is *overloaded*, if $T_k > TH_h$.
- Cluster C_k is *under-loaded*, if $T_k < TH_h$.
- Otherwise, cluster C_k is normally loaded.

If there is at least one overloaded cluster and one underloaded cluster, the grid manager starts global load balancing. Otherwise, the grid manager announces cluster managers to start local load balancing.

C. Global Load Balancing

The grid manager sorts clusters in the descending order of their loads T_k into an ordered pool of clusters. That is, clusters are ordered in the pool as $(C_{O_1}, C_{O_2}, C_{O_3}, \ldots, C_{O_{|G|}})$ where $T_{O_i} \geq T_{O_{i+1}}$ for $1 \leq i < |G|$. The grid manager selects the most overloaded cluster and the most under-loaded cluster, pairs them, extracts the clusters from the pool, and repeats the pairing until there is no more overloaded cluster or under-loaded cluster.

The proposed GLLB scheme tries to minimize the amount of immigration in order to reduce the communication overhead and to prevent vibration between overloaded cluster and under-loaded cluster. Thus, the smaller amount of load is chosen between two amounts that exceed the thresholds. For each cluster pair (C_k, C_l) where C_k is an overloaded cluster and C_l is an under-loaded cluster, the grid manager computes immigration size $M_{k,l}$ as follows:

$$M_{k,l} = \min(T_k - TH_h, TH_l - T_l) \quad (5)$$

The grid manager sends the immigration size $M_{k,l}$ to cluster manager of C_k.

After receiving the immigration size, a cluster manager of cluster C_k computes immigration amount for resources in the cluster. The cluster manager cuts peak loads of resources in order to get effect of local load balancing during global load balancing process. First, the manager sorts the resources in the descending order of load $T_k(S_i)$ such that $T_k(S_1) \geq T_k(S_2) \geq \cdots \geq T_k(S_{|C_k|})$. Sub sums of overloads A_u $(2 \leq u \leq |C_k|)$ are defined as follows:

$$A_u = \sum_{i=1}^{u-1}\left(T_k(S_i) - T_k(S_u)\right) \quad (6)$$

Next, the manager finds index u that satisfies the following condition:

$$A_u < M_{k,l} \leq A_{u+1} \quad (7)$$

where $T_k\left(S_{|C_k|+1}\right) = 0$. Then, the immigration size $M_{k,l}$ is expressed in the following equation with an unknown number x as shown in Fig. 2:

$$M_{k,l} = A_u + x \cdot u,$$

which can be rewritten as follows:

$$x = \frac{M_{k,l} - A_u}{u}. \quad (8)$$

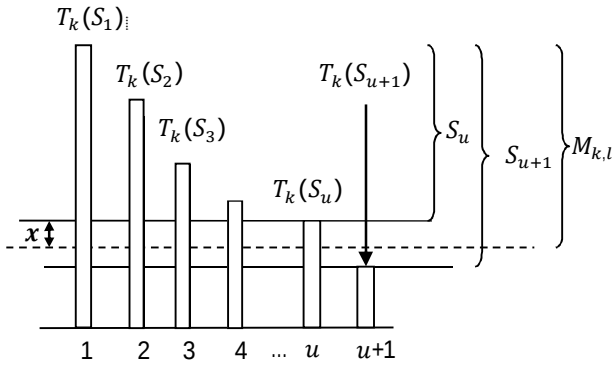

Figure 2 Finding size of immigration for each resource

For resource $S_i (i \leq u)$,

$$M_{k,l}(S_i) = T_k(S_i) - T_k(S_u) + x$$

From Equations 8 and 9, the immigration size $M_{k,l}(S_i)$ from resource S_i is

$$M_{k,l}(S_i) = T_k(S_i) - T_k(S_u) + \frac{M_{k,l} - A_u}{u}$$

The cluster manager tells the immigration size to its overloaded resources. Each resource sends the amount of tasks in the tail of wait queue to the cluster manager. As the result, peaks of the overloaded resources are flattened and the amount of local load balancing reduces. The cluster

manager of cluster C_k sends the tasks to cluster manager of cluster C_l.

When the cluster manager of cluster C_l receives the immigrated tasks, it does the reverse process of the sender cluster manager of C_k. First, the manager sorts resources in the ascending order of load $T_l(S_i)$ such that $T_l(S_1) \leq T_l(S_2) \leq \cdots \leq T_l(S_{|C_l|})$. Sub sums of under-loads B_u $(2 \leq u \leq |C_l|)$ are defined as follows:

$$B_u = \sum_{i=1}^{u-1} (T_l(S_u) - T_l(S_i))$$

Next, the manager finds index u that satisfies the following condition:

$$B_u < M_{k,l} \leq B_{u+1}$$

where $T_l(S_{|C_l|+1}) = \infty$, as shown in Fig. 3.

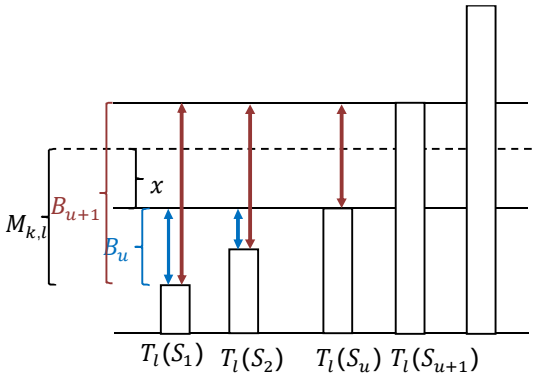

Figure 3 measuring the amount for each under-loaded resource.

The immigration size $M_{k,l}$ is

$$M_{k,l} = B_u + x \cdot u,$$

which can be rewritten as follows:

$$x = \frac{M_{k,l} - B_u}{u}. \quad (10)$$

For resource S_i $(i \leq u)$,

$$M_{k,l}(S_i) = x + T_l(S_u) - T_l(S_j). \quad (11)$$

From Equation 10 and 11, the immigration size to resource S_j is

$$M_{k,l}(S_j) = \frac{M_{k,l} - B_u}{u} + T_l(S_u) - T_l(S_j)$$

The cluster manager forward $M_{k,l}(S_j)$ amount of tasks to resource S_j for $1 \leq i \leq u$. Each resource attaches the immigrated tasks to the tail of the queue. A completed task returns to the client that issues the task.

After ordering to cluster managers for global load balancing, the grid manager moves to local load balancing phase. Since overloaded and under-loaded clusters start local load balancing just after finishing the global load balancing step, the grid manager orders normally loaded clusters to start local load balancing.

A cluster manager of overloaded or under-loaded cluster adjusts expected completion times of resources in the cluster because the values are changed by the global load balancing.

D. *Local Load Balancing*

The proposed GLLB scheme improves the load balancing scheme, LGLB scheme proposed by Qurbonboyevich and Choe [12]. Like the case of the global load balancing, GLLB scheme considers the communication overhead when task immigration is decided. First, the cluster manager of cluster C_k computes local communication overhead O_k as follows:

$$O_k = \lambda_k + P_k / \beta_k.$$

Upper threshold $TH_h(k)$ and lower threshold $TH_l(k)$ for cluster C_k are defined as follows:

$$TH_h(k) = T_k + \max\left(\alpha\sigma_k, \frac{E_k + O_k}{2}\right)$$
$$TH_l(k) = T_k - \max\left(\alpha\sigma_k, \frac{E_k + O_k}{2}\right) \quad (12)$$

Using the Equation 12, the cluster manager classifies a resource S_i as overloaded if $\mathrm{T_k}(S_i) > TH_h(k)$, under-loaded if $\mathrm{T_k}(S_i) < TH_h(k)$, and normally loaded otherwise.

Next, the cluster manager sorts all resources in the descending order of their loads into a pool of resources. That is, resources are ordered in the pool as $(S_{O_1}, S_{O_2}, \ldots, S_{O_{C_k}})$ where $\mathrm{T_k}(S_{O_i}) \geq T_k(S_{O_{i+1}})$ for $1 \leq \mathrm{i} < |C_k|$. The cluster manager selects the most overloaded and the most underloaded cluster, pairs them, extracts them from the pool, and repeats the pairing until there is no more overloaded cluster or under-loaded cluster.

For each resources pair (S_i, S_j) where S_i is an overloaded resource and S_j is an under-loaded

resource in cluster C_k , the cluster manager computes the immigration size $M_{i,j}$ as follows:

$$M_{i,j} = \min(T_k(S_i) - TH_h(k), TH_l(k) - T_k(S_j))$$

The cluster manager sends the immigration size $M_{i,j}$ to resource S_i. After receiving a message that includes the size $M_{i,j}$, the resource S_i takes out tasks of amount $M_{i,j}$ from its queue and sends them to resource S_j. Resource S_j receives the immigrated tasks into its queue.

V. EXPERIMENTAL RESULTS

The proposed GLLB scheme is implemented using Java programming language and simulated on GridSim, a discrete-event based grid simulation toolkit [11]. In order to measure the performance of the proposed scheme and to compare it with other schemes, a grid structure with resources is modeled as shown in Table 1. PE (Processing Element) performance distributes in a range from one hundred to one thousand uniformly.

Table 1 Characteristics of resources in the simulation (PE: Processing Elements, MI: Million Instructions)

Machines per resource	1
PEs per machines	1
PE performance	100-1000 MIs

Tasks used in the simulation are independent each other and their characteristics are shown in Table 2. The distribution of the file sizes ranges from 110 to 140 Kbytes and that of output sizes ranges from 275 to 350 Kbytes uniformly.

Table 2 Characteristics of tasks in the simulation

Task length	100-50,000 MIPS
File size	110-140 KB
Output size	275-350 KB

Characteristics of the simulated grid system are shown in Table 3. For each experiment, each client issues the same number of tasks to its corresponding resource. The number of the issued tasks increases in order to test capacity of the grid system. Parameter α is the threshold coefficient to control the range of normally loaded resources or clusters. Network delays are assumed to be zero.

Table 3 Characteristics of the simulated grid system

Clients	16
Resources per cluster	4
Clusters	4
Tasks	800-16,000
Threshold coefficient α	**0.5**
Arrival rate of tasks	0.25
Local network bandwidth	40,000 bps
Global network bandwidth	10,000 bps

In order to compare performances with other schemes, we implemented LGLB scheme [12] and HDLA scheme [5] in GridSim. We will call the proposed load balancing scheme as *GLLB* (Global-first Local-second Load Balancing). Since the workload for our experiments is generated by individual users, the performance is concentrated on minimizing average response time.

Threshold coefficient α is a factor that decides whether a resource is overloaded, under-loaded, or normally loaded. Figure 4 shows average response time of the proposed GLLB scheme by varying α and the number of tasks. Since value 0.5 of α shows the best performance, the value is used for the following experiments.

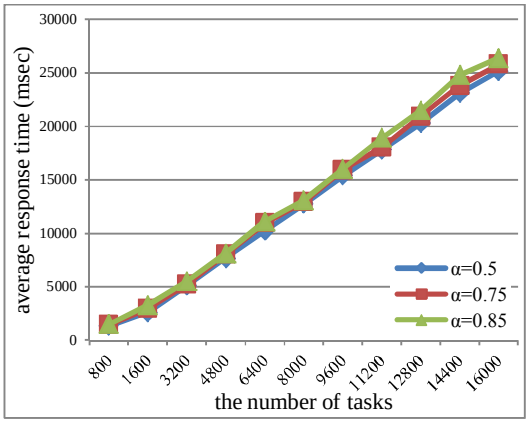

Figure 4 Performance comparison by different values of threshold coefficient α

Figure 5 shows average response times of three schemes when arrival rates of tasks are all the same. In the case, operations of GLLB and LGLB are almost same because most resources and clusters are normally loaded. Only HDLA shows a little abnormal activity.

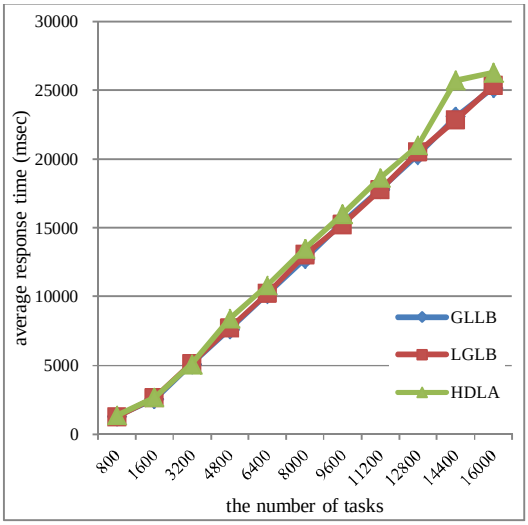

Figure 5 Average response times of three schemes HDLA, LGLB, and GLLB in the same arrival rate of tasks

In order to check the performance of dynamic load balancing, loads should be unbalanced. Thus, in the next experiment, a resource S_1 in the cluster C_1 receives ten times amount of requests compared with one of the other resources. Figure 6 shows average response time of three schemes. Before a cluster manager order to move overloaded tasks from an overloaded resource to an under-loaded resource, it checks if the under-loaded resource has the capability of immigrating tasks. If the mount of requests to be immigrated exceeds the capacity of the under-loaded resource, the load balancing algorithm of HDLA diagnosis the system is 'saturated'. In the situation of resource 1 in the Figure 6, the resource is diagnosed as saturated and loads are not distributed.

LGLB scheme processes local load balancing. Thus a cluster that consists of resources S_1, S_2, S_3, and S_4 is balanced. After the balancing, global balancing is not applied because clusters are located within the two thresholds. Since GLLB scheme makes global balancing at the first step, some loads are distributed globally. Because remainder loads are distributed in the second step, the amount of local load immigration is smaller than that of LGLB scheme.

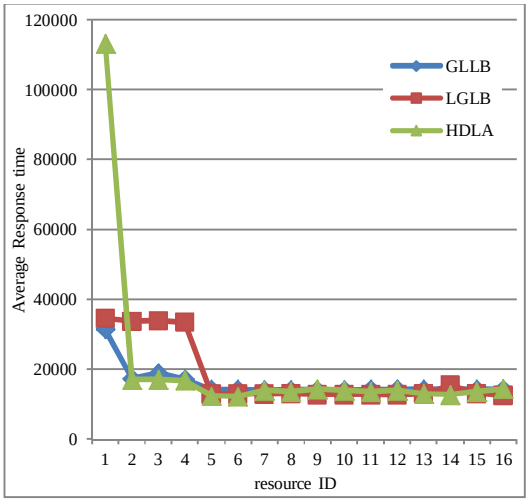

Figure 6 Average response times in the case of highly unbalanced workload

Figure 7 shows the performance comparison of three schemes according to increasing the number of tasks per resource. The number of tasks per resource is decided by the arrival rate of tasks in each resource. The arrival rate is 0.25 in the case of 100 tasks per resource and the rate proportionally increases in order to increase the number of tasks given fixed duration. In the case of 500 tasks per resource, the average response time of GLLB is about 77.4%, compared with that of HDLA.

Figure 8 shows the range of response times in the big scale system with five clusters and 100 resources with one hot spot. Since GLLB distributes overloads to other clusters, loads are more balanced. As the result, GLLB has more stable response time.

VI. CONCLUSIONS

In this paper, we propose two-level hierarchical dynamic load balancing scheme GLLB that balances global loads followed by local load. Such load balancing order reduces the amount of

unnecessary task immigration with a support of careful immigration decision. The immigration decision is composed of statistics and communication overheads. Experimental results show that the average response time of the proposed GLLB is 99.7% and 95%, compared to that of LGLB and HDLA, respectively in the case of the uniformly distributed work-load. In the case of the one hot spot work-load distribution, the average response time of the GLLB is 87.2% and 78.2%, compared to that of LGLB and HDLA, respectively.

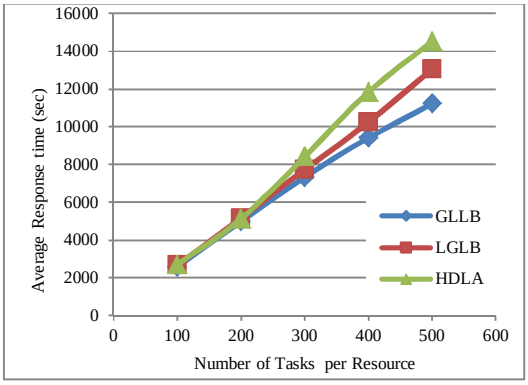

Figure 7 Tendency of average response time as the number of allocated tasks per resource increases

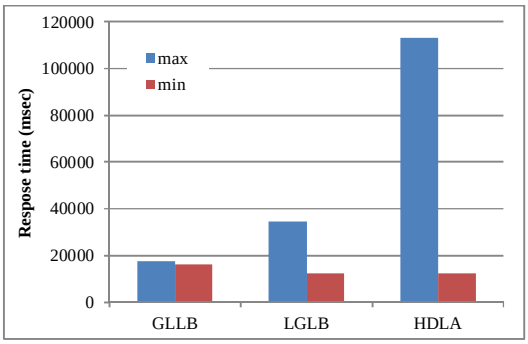

Figure 8 The range of response times in the big scale system with a hot spot resource

REFERENCES

[1] Ian Foster, Carl Kesselman, Jeffrey M Nick, and Steven Tuecke, "Grid services for distributed system integration", *Computer, 35(6)*, pp. 37-46, 2002.

[2] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu, "Cloud computing and grid computing 360-degree compared", in *Grid Computing Environments Workshop, 2008. GCE'08. IEEE*, 2008, pp. 1-10.

[3] Thomas L. Casavant and Jon G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems", *Software Engineering, IEEE Transactions on*, 14(2), pp. 141-154, 1988.

[4] Mohammed Javeed Zaki, Wei Li, and Srinivasan Parthasarathy, "Customized dynamic load balancing for a network of workstations", in *High Performance Distributed Computing, 1996., Proceedings of 5th IEEE International Symposium on. IEEE*, 1996, pp. 282-291.

[5] Belabbas Yagoubi and Yahya Slimani, "Dynamic load balancing strategy for grid computing", *Transactions on Engineering, Computing and Technology*, *13*, pp. 260-265, 2006.

[6] Belabbas Yagoubi and Meriem Meddeber, "Distributed load balancing model for grid computing", *Revue ARIMA*, *12*, pp. 43-60, 2010.

[7] Said Fathy El-Zoghdy, "A load balancing policy for heterogeneous computational grids", *International Journal of Advanced computer Science and Applications, 2(5)*, pp. 93-100, 2011.

[8] Malarvizhi Nandagopal and Rhymend V Uthariaraj, "Hierarchical status information exchange scheduling and load balancing for computational grid environments", *IJCSNS International Journal of Computer Science and Network Security, 10(2)*, pp. 177-185, 2010.

[9] Minakshi Tripathy and CR Tripathy, "Dynamic load balancing with work stealing for distributed shared memory clusters", in *Industrial Electronics, Control & Robotics (IECR), 2010 International Conference on. IEEE*, 2010, pp. 43-47.

[10] Seyed Rasool Moosavi-Nejad, S. S. Mortazavi, and Bijan Vosoughi Vahdat, "Fuzzy based design and tuning of distributed systems load balancing controller", in *5th Symposium on Advances in Science & Technology (SASTech)*, Mashhad, Iran, May 2011.

[11] Rajkumar Buyya and Manzur Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing", *Concurrency and Computation: Practice and Experience, 14(13-15)*, pp. 1175-1220, 2002.

[12] Hojiev Sardor Qurbonboyevich and Tae-Young Choe, "Two-level dynamic load balancing algorithm using load thresholds and pairwise immigration", *International Journal on Computer Science and Engineering (IJCSE), 5(4)*, pp. 211-220, April 2013.