Chameleon: A Load Balancing Model For Public Cloud

Abdul Jaleef p k [#1], P Sivaprakash [*2],D Sumathi [@3]

[#1]*Department of Computer Science and Engineering, PPGIT, Coimbatore, Tamilnadu, India*[1]

[*2]*Department of Computer Science and Engineering, PPGIT, Coimbatore, Tamilnadu, India*[2]

[@3]*Department of Computer Science and Engineering, PPGIT, Coimbatore, Tamilnadu, India*[3]

1jaleefpk@gmail.com

2sivaprakash04@gmail.com

[3]sumathi.cloud73@gmail.com

***Abstract*: The initial resistance to public cloud has begun to lesson and customers are beginning to realize its efficiencies and economic advantages it can provide. As the public confidence increasing end user spending on public cloud services are growing rapidly. to uphold the public confidence there is a strong need to address the issues in cloud computing. Load balancing is one of the central issues in cloud computing, a better load balancing mechanism will be the solution for almost issues in cloud computing. A good load balancing makes cloud computing more efficient and improves user satisfaction. Here introducing a new load balancing technique inspired from chameleon color changing behavior for load balancing which aggregate and use different type strategy based on the situation.**

***Keywords*--- load balancing model; public cloud; cloud partition; game theory; chameleons load balancing model.**

I. INTRODUCTION

Cloud computing, a framework for enabling convenient, and on-demand network access to a shared pool of computing resources is emerging as a new paradigm of large-scale distributed computing. It has widely been adopted by the industry, though there are many existing issues like Load Balancing, Virtual Machine Migration, Server Consolidation, Energy Management, etc. that are not fully addressed . Central to these issues load balancing a crucial one to solve. load balancing is a mechanism to distribute the dynamic workload evenly to all the nodes in the whole cloud to achieve a high user satisfaction and resource utilization ratio. With the increasing popularity of cloud computing, the amount of processing that is being done in the clouds is surging drastically. As the requests of the clients can be random to the nodes they can vary in quantity and thus the load on each node can also vary. Therefore, every node in a cloud can be unevenly loaded of tasks according to the amount of work requested by the clients. This phenomenon can drastically reduce the working efficiency of the cloud.

According to the global research firm Gartner, the global public cloud services market is expected to increase from $111 billion in 2012 to $131 billion worldwide in 2013. To maintain the performance and user satisfaction there is a strong need to address the issues in public cloud. Load balancing in cloud computing systems is one of the main issues. Always a distributed solution is required. Because Since the job arrival pattern is not predictable and the capacities of each node in the cloud differ, and it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfil the required demands. For load balancing problem, workload control is crucial to improve system performance and maintain stability. A good load balancing mechanism has to distribute the dynamic local workload evenly across all the nodes in the whole cloud to avoid a situation where some nodes are heavily loaded while others are idle or doing little work.

There are several cloud computing categories; the load balancing model given in this paper is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The model presented here is inspired from chameleons color changing behavior which changes its behavior based on the situation and chooses the best load balancing strategy based on situation. The rest of the paper discuss about the related works and the proposed system.

II. RELATED WORK

Load balancing is one of the central issues in cloud computing, a better load balancing mechanism will be the solution for almost issues in cloud computing. Cloud computing is efficient and scalable but maintaining the stability of processing so many jobs in the cloud computing environment is a very complex problem. Load balancing schemes depending on whether the system dynamics are important can be either static or dynamic. Static schemes do not use the system information and are less complex while dynamic schemes will bring additional costs for the system but can change as the system status changes. A dynamic scheme is used here for its flexibility.

The existing load balancing method for cloud computing are surveyed and compared in [16]. Each particular method has advantage in a particular area but not in all situations. In This paper discussing a comparative study of three distributed load-balancing algorithms for Cloud computing scenarios and proposing a new cloud load balancing model.

A. HoneybeeForagingBehavior-M.Randlesetal.[15] investigated decentralized honeybee-based load balancing technique that is a nature-inspired algorithm for self-organization. It achieves global load balancing through local server actions. Performance of the system is enhanced with increased sys-tem diversity but throughput is not increased with an increase in system size. It is best suited for the conditions where the diverse population of service types is required.

B. Biased Random Sampling- M. Randles et al. [15] investigated a distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system. The performance of the system is improved with high and similar population of resources thus resulting in an in-creased throughput by effectively utilizing the increased sys-tem resources. It is degraded with an increase in population diversity.

C. Active Clustering- M. Randles et al. [15] investigated a self-aggregation load balancing technique that is a self-aggregation algorithm to optimize job assignments by connecting similar services using local re-wiring. The performance of the system is enhanced with high resources thereby in-creasing the throughput by using these resources effectively. It is degraded with an increase in system diversity. The honeybee-algorithm performs consistently well as system diversity increases. However, despite performing better with high resources and low diversity, both the random sampling walk and active clustering *degrade* as system diversity increases. The honeybee algorithm again performs consistently, but does not increase throughput in line with system size. However, the other approaches are able to utilize the increased system resources more effectively to increase throughput.

The results indicate that the honeybee-based load-balancing approach gives better performance when a diverse population of service types is required. Secondly, results indicate that the random sampling walk performs better in conforming, similar populations, and quickly degrades as the population diversity increases. It is not known how to select appropriate balancing techniques for given applications that will provide a suitable configuration for the application – and provide it in a timely manner. The combination of algorithms is crucial to this process.

III. SYSTEM MODEL

There are several cloud computing categories with this work focused on a public cloud. A public cloud is based on the standard cloud computing model, with service provided by a service provider [11]. A large public cloud will include many nodes and the nodes in different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations.

This load balancing model is inspired from chameleon colour changing behaviour. Where work load are controlled at the balancer. jobs goes to the balancer where the jobs are located then balancer act as low level balancer or high level balancer depending on the status of the partition.

Cloud partitioning is used to manage the large cloud which includes many nodes and the nodes are in different geographical locations. A cloud partition or management region is a subarea of the public cloud with divisions based on the geographic locations.

A node in each management region is chosen as the partition balancer, each balancer connect with many of the other balancers of a CSP according to the information get from the CSP. A load can add to or remove from the management region; also, the selection of balancer node is not a permanent thing but a new head node can be elected if the previous node stops functioning properly due to some inevitable circumstances. The balancer node is chosen in such a way that it has the most number of neighbouring nodes. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs.

When a job arrives at partition where job is located the balancer node of that particular partition act as low-level balancer or high-level balancer depending on the location’s status. If the status is idle or normal the balancer node act as low-level balancer and the job is handled locally using appropriate strategy. If the status is heavy then it acts as high-level balancer which assigns the job to other partition using three criteria.

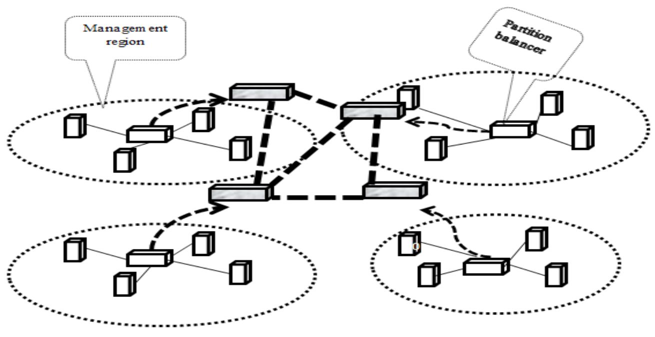

Fig.1. load balancing based on partition

IV. CHAMELEON LOAD BALANCING

This model is inspired from behavior of chameleons which change colour to reflect their moods. By doing so, they send social signals to other chameleons. For example, darker colours tend to mean a chameleon is angry. Lighter colours might be used to attract mates. Some chameleons also change colours to help their bodies adjust to changes in temperature or light. When a chameleon wants to convey a particular mood or message, its brain sends a message to its chromatophores, which then move pigments around to change the chameleon's colour.This biologically-inspired technique is used here for load balancing by aggregating different strategies based on the condition of the nodes and communicating with other partition balancers.

A public cloud is based on the standard cloud computing model, with service provided by a service provider [11]. A large public cloud will include many nodes and the nodes in different geographical locations. We can see these nodes as chameleons. Where the chameleon's changes its color to reflect its moods the nodes changes its status to reflects its load.

The first task is to define the load degree of each node. The node load degree is related to various static parameters and dynamic parameters. The static parameters include the number of CPU's, the CPU processing speeds, the memory size, etc. Dynamic parameters are the memory utilization ratio, the CPU utilization ratio, the network bandwidth, etc.

Step 1 Define a load parameter set: F= {F_1.F_2,.....,F_m} with each $F_i (1 \le i \le m, F_i \in [0,1])$ parameter being either static or dynamic. m represents the total number of the parameters.

Step 2 Compute the load degree as:

Load_degree (N) = $\sum_{i=1}^{m} \propto_i F_i$

$\propto_i$ ($\sum_{i=1}^{n} \propto_i = 1$)are weights that may differ for different kinds of jobs. N represents the current node.

Step 3 Define evaluation benchmarks. Calculate the average cloud partition degree from the node load degree statistics as:

$$Load\_degree_{avg} = \frac{\sum_{i=1}^{n} Load\_degree(N_i)}{n}$$

The bench mark $\text{Load\_degree}_{\text{high}}$ is then set for different situations based on the $\text{Load\_degree}_{\text{avg}}$.

Step 4 Three nodes load status levels are then defined as:

Idle When Load_degree(N)=0;there is no job being processed by this node so the status is charged to Idle.

Normal For 0 <Load_degree(N) ≤ $\text{Load\_degree}_{\text{high}}$, the node is normal and it can process other jobs.

Overloaded When $\text{Load\_degree}_{\text{high}}$ ≤ Load_degree(N), the node is not available and can not receive jobs until it returns to the normal.

In this way the load degrees are calculated and the status is changes according to the load degree. A node in each management region is chosen as the partition balancer, each balancer connect with many of the other balancers of a CSP according to the information get from the CSP.

The load balancer uses the suitable strategy to assign the jobs. The load balancer also changes its status according to the partitions load degree as the chameleons uses different color to indicate its mood and for fine processing the load balancer uses different strategies to suit the condition of the system. Therefore, the current model integrates several methods and switches between the load balance methods based on the system status.

The cloud partition status can be divided into three types:

(1) Idle: When the percentage of idle nodes exceeds α, change to idle status.

(2) Normal: When the percentage of the normal nodes exceeds β, change to normal load status.

(3) Overload: When the percentage of the overloaded nodes exceeds γ, change to overloaded status.

The parameters α, β and γ are set by the cloud partition balancers. Based on the status the balancer behaves differently by using different strategy. A relatively simple method can be used for the partition idle state with a more complex method for the normal state. The load balancers then switch methods as the status changes. Here, the idle status uses an improved Round Robin algorithm while the normal status uses a game theory based load balancing strategy.

A. Load balance strategy for the idle status:

When the cloud partition is idle, many computing resources are available and relatively few jobs are arriving. In this situation, this cloud partition has the ability to process jobs as quickly as possible so a simple load balancing method can be used. There are many simple load balance algorithm methods such as the Random algorithm, the Weight Round Robin, and the Dynamic Round Robin [12]. The Round Robin algorithm is used here for its simplicity.

The Round Robin algorithm is one of the simplest load balancing algorithms, which passes each new request to the next server in the queue. The algorithm does not record the status of each connection so it has no status information. In the regular Round Robin Algorithm, every node has an equal opportunity to be chosen. However, in a public cloud, the configuration and the performance of each node will be not the same; thus, this method may overload some nodes. Thus, an improved Round Robin algorithm is used, which called "Round Robin based on the load degree evaluation".

The algorithm is still fairly simple. Before the Round Robin step, the nodes in the load balancing table are ordered based on the load degree from the lowest to the highest. The system builds a circular queue and walks through the queue again and again. Jobs will then be assigned to nodes with low load degrees. The node order will be changed when the balancer refreshes the Load Status Table.

B. Load balancing strategy for the normal status:

When the cloud partition is normal, jobs are arriving much faster than in the idle state and the situation is far more complex, so a different strategy is used for the load balancing. Each user wants his jobs completed in the shortest time, so the public cloud needs a method that can complete the jobs of all users with reasonable response time. Penmatsa and Chronopoulos[13] proposed a static load balancing strategy based on game theory for distributed systems. And this work provides us with a new review of the load balance problem in the cloud environment. As an implementation of distributed system, the load balancing in the cloud computing environment can be viewed as a game.

Game theory has non-cooperative games and cooperative games. In cooperative games, the decision makers eventually come to an agreement which is called a binding agreement. Each decision maker decides by comparing notes with each other's. In non-cooperative games, each decision maker makes decisions only for his own benefit. The system then reaches the Nash equilibrium, where each decision maker makes the optimized decision. The Nash equilibrium is when each player in the game has chosen a strategy and no player can benefit by changing his or her strategy while the other players' strategies remain unchanged. There have been many studies in using game theory for the load balancing. Grosu et al.[14] proposed a load balancing strategy based on game theory for the distributed systems as a non-cooperative game using the distributed structure. They compared this algorithm with other traditional methods to show that their algorithm was less complexity with better performance. Aote and Kharat[15] gave a dynamic load-balancing model based on game theory. This model is related on the dynamic load status of the system with the users being the decision makers in a non-cooperative game. Since the grid computing and cloud computing environments are also distributed system, these algorithms can also be used in grid computing and cloud computing environments. Previous studies have shown that the load balancing strategy for a cloud partition in the normal load status can be viewed as a non-cooperative game, as described here.

The players in the game are the nodes and the jobs. Suppose there are n nodes in the current cloud partition with N jobs arriving,

μ_i: Processing ability of each node, i=1,....,n

ϕ_j: Time spending of each job

$\Phi = \sum_{j=1}^{N} \phi_j$: Time spent by the entire cloud partition $\Phi < \sum_{i=1}^{N} \mu_i$

S_{ji}: Fraction of job j that assigned to node i $\sum_{i=1}^{N} S_{ji} = 1 \; and \; 0 \le S_{ji} \le 1$

In this model, the most important step is finding the appropriate value of S_{ji}. The current model uses the method called "the best reply" to calculate S_{ji} of each node, with a greedy algorithm then used to calculate S_{ji} for all nodes. This procedure gives the Nash equilibrium to minimize the response time of each job. The strategy then changes as the node's statuses change.

C. Load balancing strategy for overloaded status:

When the partition status is heavy then the balancer switch the load to another partition. Here the balancer uses three criteria, Computing Capacity with Neighbours, Distance and Reputation. The balancer node of each area connected with many other balancer nodes so that it checks the status of each partition and if it is not overloaded then uses the following three criteria to select the best partition.

Computing Capacity with Neighbours (CCN): Assuming that each balancer knows the number, CPU power and load of the overall nodes in its corresponding area, it can easily obtain the Computing Capacity (CC) of an area. If we assume that balancers can assign jobs to their neighbouring areas, we can also consider the computing capacity of these areas. We take this feature into account, defining the CCN metric of one area i with K_i neighbours as

$$CCN_i = \propto * \, CC_i + (1 - \propto) * \sum_{j=0}^{K_i} \frac{CCN_j}{K_i}$$

The parameter allows us to define the relative importance of the local area information and the one from the neighbours in each CC calculation. Te larger the, the more tasks are assigned to the local area and fewer to the neighbouring ones, and vice versa. This parameter governs the amount of load distributed to the powerful areas. The flow of tasks between areas increases with (1-α).

Distance: Every balancer has an attribute called Mass Centre, which represents the area's degree of dispersion. The lower the MC, the more dispersed the nodes in the area are and vice versa. MC is defined as the weighted average, based on the Computing Capacity, of the relative position of the workers in an area. This means the distance of a powerful node will have more influence on the area's MC than the distance of a user with a low CC. The MC is defined formally as

$$MC_i = \frac{\sum_{j=0}^{W_i} \left(\frac{dist_{max} - dist_j}{dist_{max}} * CC_j \right)}{\sum_{j=0}^{W_i} CC_j}$$

Where $dist_{max}$ is the maximum distance between the balancers and the furthest worker in its area, $dist_j$ represents the distance between the balancer and the nodes j and W_i is the number of nodes of balancer i.

The Reputation (R): of a balancer i (R_i) is the probability of a successful service invocation by such a balancer. Formally, reputation is defined as

$$R_i = \frac{ST_i}{TT_i}$$

Where ST_i is the number of successfully executed tasks assigned by balancer i and TT_i is the total number of tasks assigned by the same balancer.

The scheduling procedure is based on a metric, called the scheduling criteria (V). As we will see below, it is used to spread the tasks around the high level, which is made up of interconnected managers. The formal definition of the scheduling criteria of a balancer i, Vi, is as follows:

$$V_i = \left(\frac{MC_i}{MC_{max}} \right) \beta_1 + \left(\frac{CCN_i}{CCN_{max}} \right) \beta_2 + (R_i) \beta_3$$

Where $\beta_1 + \beta_2 + \beta_3 = 1$. The weight assigned to each β term depends on the job attributes. In fine-grained jobs, β_1 would be quite important, as would be β_2 for course-grained jobs and β_3 to guarantee certain QoS and reduce the impact of unstable nodes on job execution times.

V. Experimental results

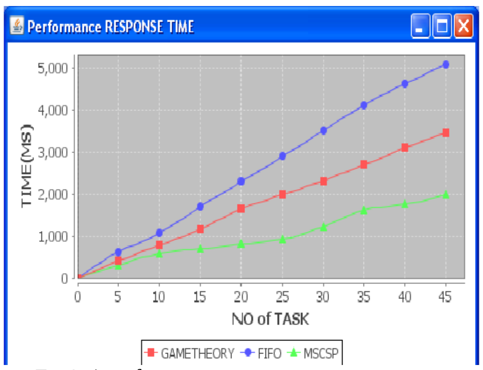

Fig 2 a) performance response time comparison

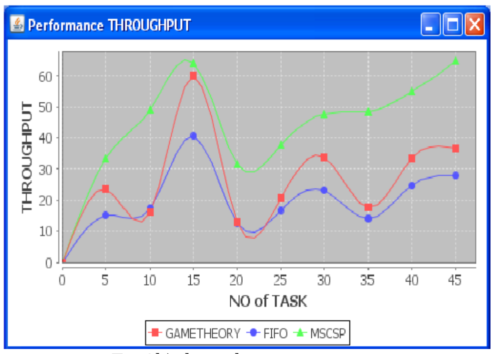

Fig 2b) throughput comparison

Fig. 2. Comparison between fifo,game theory and CLD (a) Response Time (b) Throughput ,With switch mechanism in public cloud, we can achieve effective load balancing for improved performance. Load balancer with switch mechanism uses different strategies in different situations to have an optimal utilization of virtualized resources. This load balance model for public cloud ensures improved performance , availability and responsiveness.

VI. CONCLUSION

The load balancing model presented here is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. we are introduced a new balancing mechanism inspired from chameleons color changing behavior which aggregate different strategy and use these strategy based on the situations. With switch mechanism in public cloud, we can achieve effective load balancing for improved performance.

Load balancer with switch mechanism uses different strategies in different situations to have an optimal utilization of virtualized resources. This load balance model for public cloud ensures availability and responsiveness. This also improves performance and reduces total cost.

VII. FUTURE WORK

1. Cloud division rules

Nodes in a same cluster may be far from other nodes or there will be some clusters in the same geographical area that are still apart. Thus, the framework will need different cloud division methodology

2. Better load status evaluation.

A good algorithm is needed to set Load_degreehigh and Load_degreelow.

3. load balancing strategies. Experimenting new strategies may give good results.

REFERENCES

[1] C.Schroth and T. Janner, (2007). Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. IEEE IT Professional Vol.9, No.3 (pp.36-41), June 2007.

[2] P.A Laplante, Zhang Jia and J.Voas, “What's in a Name? Distinguishing between SaaS and SOA," *IT Professional* , vol.10, no.3, pp.46-50, May-June 2008

[3] W.M.P. van der Aalst, Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72-76, 2003.

[4] R. Ruggaber, Internet of Services SAP Research Vision. In 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007), pp: 3.

[5] G. Briscoe, P. De Wilde (2008) Digital Ecosystems: Optimisation by a Distributed Intelligence. In Proceedings of the 2nd IEEE International Conference on Digital Ecosystems and Technologies, Phitsanulok, Thailand.

[6] R. Sterritt (2005) Autonomic Computing. Innovations in Systems and Software Engineering, 1(1), pp: 79–88.

[7] Martin Randles, A. Taleb-Bendiab and David Lamb, Scalable Self- Governance Using Service Communities as Ambients. In Proceedings of the IEEE Workshop on Software and Services Maintenance and Management (SSMM 2009) within the 4th IEEE Congress on Services, IEEE SERVICES-I 2009 - July 6-10, Los Angeles, CA (To appear).

[8] IBM. IBM Introduces Ready to Use Cloud Computing. IBM Press Release, 15th November 2007. http://www-03.ibm.com/press/us/en/pressrelease/22613.wss

[9] R.L. Grossman, "The Case for Cloud Computing," *IT Professional*, vol.11, no.2, pp.23-27, March-April 2009

[10] Amazon Elastic Compute Cloud (EC2), http://www.amazon.com/gp/browse.html?node=201590011

[11] R. Hunter, The why of cloud, http://www.gartner.com/DisplayDocument?doccd=226469 &ref= g noreg, 2012.

[12] M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis,and A. Vakali, Cloud computing: Distributed internetcomputing for IT and scientific research, InternetComputing, vol.13, no.5, pp.10-13, Sept.-Oct. 2009.

[13] P. Mell and T. Grance, The NIST definition of cloudcomputing, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2012.

[14] Microsoft Academic Research, Cloudcomputing,http://libra.msra.cn/Keyword/6051/cloudcomputing?query=cloud%20computing, 2012.

[15] Google Trends, Cloud computing, http://www.google.com/trends/explore#q=cloud%20computing, 2012.

[16] N. G. Shivaratri, P. Krueger, and M. Singhal, Loaddistributing for locally distributed systems, Computer,vol. 25, no. 12, pp. 33-44, Dec. 1992.

[17] B. Adler, Load balancing in the cloud: Tools, tips andtechniques, http://www.rightscale.com/infocenter/whitepapers/Load-Balancing-in-theCloud.pdf, 2012

[18] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, andC. Mcdermid, Availability and load balancing in cloudcomputing, presented at the 2011 International Conferenceon Computer and Software Modeling, Singapore, 2011.

[19] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh,N. Nitin, and R. Rastogi, Load balancing of nodesin cloud using ant colony optimization, in Proc. 14thInternational Conference on Computer ModellingandSimulation (UKSim), Cambridgeshire, United Kingdom,Mar. 2012, pp. 28-30.

[20] M. Randles, D. Lamb, and A. Taleb-Bendiab, A comparative study intodistributed load balancingalgorithms for cloud computing, in Proc. IEEE 24thInternational Conference on Advanced InformationNetworking and Applications, Perth, Australia, 2010,pp. 551-556.

[21] A. Rouse, Public cloud, http://searchcloudcomputing.techtarget.com/definition/public-cloud, 2012.

[22] D. MacVittie, Intro to load balancing for developers —The algorithms,https://devcentral.f5.com/blogs/us/introto-load-balancing for developers-ndash-the-algorithms,2012.

[23] S. Penmatsa and A.T. Chronopoulos, Game-theoreticstatic load balancing for distributed systems, Journalof Parallel and Distributed Computing, vol. 71, no. 4,pp. 537-555, Apr. 2011.

[24] D. Grosu, A. T. Chronopoulos, and M. Y. Leung, Loadbalancing in distributed systems: An approach usingcooperative games, in Proc. 16th IEEE Intl. Parallel andDistributed Processing Symp., Florida, USA, Apr. 2002,pp. 52-61.