LOAD BALANCING IN OPTICAL GRIDS

Mohamed Abouelela[1] and Mohamed El-Darieby[2]

[1]Electronic Systems Engineering Department,
University of Regina, Regina, SK, Canada
ahmedabm@uregina.ca
[2]Software Systems Engineering Department,
University of Regina, Regina, SK
Mohamed.El-Darieby@uregina.ca

ABSTRACT

This collaboration of geographically distributed domains in multi-domain optical grid environment should be done in a way that maintains the privacy of each participant domain. This calls for a new load balancing hierarchical approach to deal with such environments. In this paper, we propose two load balancing techniques within this hierarchical approach: Network Aware Divisible Load Algorithm (NADLA) and Genetic Algorithm based Load Distribution (GA-LD). Both algorithms are considered as Network Aware. Simulation results show the scalability and feasibility of the proposed approach and the advantages of the two proposed techniques compared to the classical Divisible Load Algorithm (DLA).

KEYWORDS

Divisible Load, Grid Computing, Load Balancing, Optical Grids

1. INTRODUCTION

In the last years, the growing of the scientific and enterprise applications pushes grid systems to more scalable and collaborative direction. Single domain resources are becoming insufficient to deal with application requirements. Grid applications require massive amounts of data to be transferred and processed in geographically distributed sites across the grid. The grid is an inter-connected multi-domain environment where each domain consists of computational, storage and communication resources grouped together for business or administrative reasons. Each domain is independently administrated and is free to deploy different technologies. Achieving better resource utilization calls for running load balancing techniques across different grid domains. This calls for novel load balancing techniques running in multi-domain scalable architecture to achieve better resource utilization without sacrificing domain security or privacy.

The proposed multi-domain optical grid architecture assumes a dynamic on-demand reconfigurable optical network instead of pre-planned statically provisioned optical network. Dynamic reconfigurable optical grids presented in different projects such as Phosphorus [1] and G-lambda Projects [2]. These projects present the structure and different components of optical network deployment in grids, and how these components interact to provide the grid applications with the services and capabilities making it able to manage its grid and optical network resource.

In this paper, we present load balancing techniques in multi-domain hierarchical optical grid architecture that maintains the privacy, integration and scalability requirements. The privacy of a grid domain must be maintained in for confidentiality and commercial competition. For example, internal topology information of a domain should not be revealed to other domains [3]. The domain privacy could be maintained by keeping the full resource information internally, while

DOI: 10.5121/ijgca.2012.3201

sharing aggregated and abstracted values to be used in load balancing purpose by a higher level load balancer. The aggregation process is performed by aggregating resource status data, periodically and on-demand, from different domains in a bottom-up approach over different hierarchical levels. For each domain, the internal topology and resource status information are aggregated and abstracted in six parameters. Those parameters are shared with other domains, while the full data is kept internally. Two alternative load balancing algorithms were introduced: Network Aware Divisible Load Algorithm (NADLA) and Genetic Algorithm based Load Distribution (GA-LD). NADLA is based on the classical Divisible Load Algorithm, while GA-LD is based on Genetic Algorithms. Both algorithms are network aware; they consider the availability and status of networking resources as well as the computing resources when deciding on load distribution. Traditional load balancers considered availability and optimization of computing resources and did not take into account networking resource availability. This is acceptable in computationally intensive applications, and for single domain environment. However, in data intensive applications and for geographically distributed environments, considering network resources become an important factor and cannot be ignored.

The rest of the paper is organized as follows: related work is summarized in Section 2. The proposed architecture is described in Section 3, the load balancing techniques are explained in section 4. Experiments Results and discussions are provided in section 5. Finally, conclusions are offered at the end of the paper.

2. RELATED WORK

Load balancing in high performance grid computing is an area of on-going research and development. Most of these efforts assume a centralized load balancer that has a complete vision of computing resources. This assumption is not valid for large scale world wide- grid networks. Practically, grid network comprises geographically distributed heterogeneous resources interconnected by multi-domains networks. Each domain is managed by a local domain grid manager that is usually not willing to share its internal domain information to others due to security and business confidentiality reasons. Moreover, maintaining and managing, in one centralized location, dynamic data coming from heterogeneous resources located in multi-domain environment is not feasible for large scale networks. To deal with such multi-domain environments, a multi-domain hierarchical architecture is usually proposed.

Phosphorus research Project [1] proposed a multi-domain architecture. In this paper, we propose load balancing techniques within this multi-domain architecture. In such architecture, each domain will maintain its structure and topology internally, while share an abstracted data about its computing and networking resources status with its Resource Manager (RM). The RM is similar to IDB in Phosphorus Project system. The domain RM provides the domain load balancer with the necessary resource status information. The RMs and the load balancers are to be arranged in a multi-level hierarchical architecture.

Load balancing in single domain computing environments is frequently discussed in the literature. Divisible Load Theory (DLT) has been successfully applied to parallel and distributed systems, as well as to grid computing environment [4], [5], [6]. Genetic Algorithms (GA) based approaches were also proposed to schedule Divisible Loads [7], [8]. Integer Linear Programming has also been introduced to model such problems [6].

DLT provides a linear mathematical model and scalable formulations for parallel, distributed, and grid computing environments. The effectiveness of DLT is validated in several real-life applications such as parallel video encoding, image processing, and database applications [4]. The traditional DLA takes into account sites processing capacity and estimated waiting time. It ignores totally the effect of the communication delay. This is acceptable in applications where

data transfer time is negligible compared to jobs executions time. However, many e-science applications, such as linear algebra, image processing, and data mining [9], are becoming more data intensive. In [10], two adaptive and distributed load balancing algorithms considering transfer cost and network heterogeneity were introduced. In [6], the authors use the DLT in a combined lambda grid dimensioning and scheduling problem. None of the above contributions considers networking resources availability and connectivity while scheduling both computational and networking resources using DLT. In [7], a GA based approach that co-schedules computational and networking resources, while considering network resources availability and connectivity was introduced. The main drawback of this approach is the long GA execution time. In this paper, two alternative load distribution scheduling algorithms based on DLT and GA are introduced. Both of them are Network Aware schedulers that consider network resources availability and connectivity.

3. PROPOSED ARCHITECTURE

In multi-domain environments, sites are organized into different domains. A RM maintains topological and resource status information about different computing and networking resources in a domain. The RMs from different domain are arrange in hierarchical way. Sites are considers level-0 in the hierarchy, while RMs are considered level-1. Level-1 RMs are grouped into logical domain. The process of grouping RMs (at one hierarchy level) into logical domains and abstracting such domains via a RM (at the next higher level) is done at all levels of the hierarchy (see Figure 1).

At each level in the hierarchy, networking and computing resource status information is aggregated by the corresponding RM, abstracted and sent to parent RM. Typically six parameters are to be aggregated to be used in the load balancing process:

- Aggregated Processing Capacity C for site/domain. For a certain site, the aggregated processing capacity is a metric of the processing power of that site. It is measured as the computing time per unit data set. The aggregated processing capacity for a domain is calculated as the summation of the capacities of its members (either RMs or sites) Considering the grid system example shown in Figure 1, the aggregated processing capacity for domain 0 is calculated as a function of the processing capacities for all its sites. Such calculation is done by RM-0.

- Aggregated Processing Waiting Time (APWT) for site/domain. For a certain site, APWT is the minimum time by which the computational resources at this site will be available and ready for any new task. The aggregated APWT for a domain is calculated as the average waiting time of its members (e.g., RMs or sites). Considering Figure 1 example, the APWT for domain 0 is calculated as the average APWT value over all its sites members, while the APWT for the top most domain is calculated as the average APWT of RM-0, RM-1, RM-2, and RM-3.

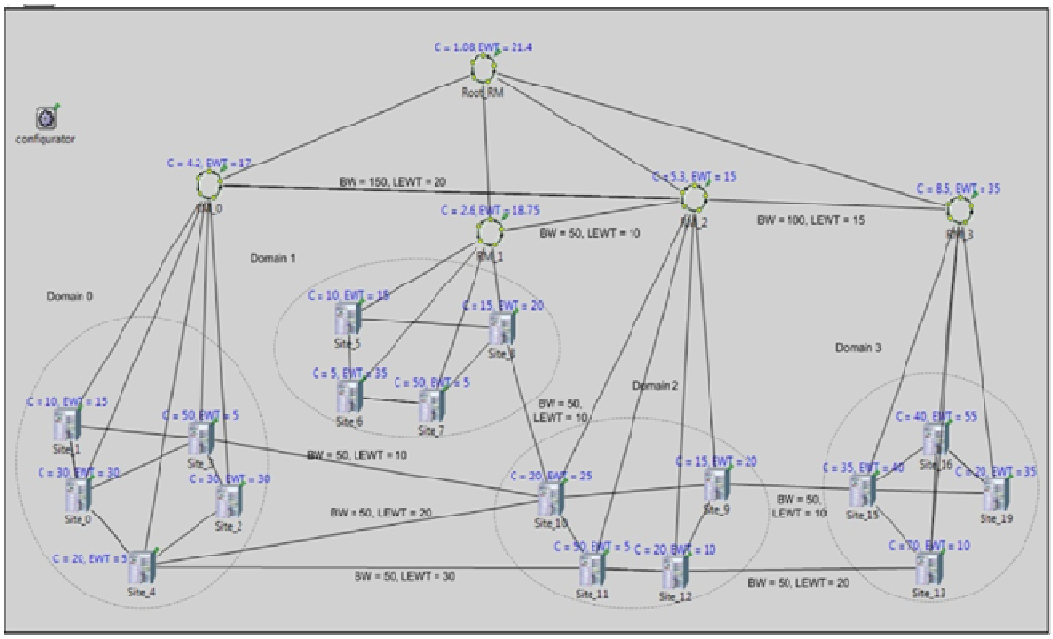


Figure 1. Two Levels Hierarchical Grid Architecture

- Aggregated Link Capacity (Bandwidth) (ALC). Virtual links connecting different RMs at level k represent the inter-domain links connecting different domains at level k-1. The capacity of virtual link connecting two parent RMs at level k is calculated as the summation of capacities of all the inter-domain links connecting the two domains managed by these nodes. In Figure 1 example, the bandwidth for all the inter-domain links is assumed to be 50 Gbps. Three inter-domain links connect domain 0 with domain 2, then the bandwidth of the virtual link between RM-0 and RM-2 equals 150 Gbps.

- Aggregated Link Waiting Time (ALWT) is the minimum time by which the link will be available. ALWT of a virtual link connecting two parent RMs at level k is calculated as a function of link waiting times over all the inter-domain links connecting the two domains managed by these nodes. In Figure 1 example, the ALWT of the Virtual link between RM-0 and RM-2 is calculated as the average over the three inter-domain links between domain 0 and domain 2.

- Estimated Domain Communication Capacity (EDCC). EDCC for a certain domain is a metric for the internal capacity (bandwidth) of the links in the domain. It is calculated as the average link bandwidth over all intra-domain links in this domain. The domain RM will share this value with its parent RM, while the detailed intra-domain links data will be available only at the domain RM. In Figure 1 example, the EDCC for domain 0 is calculated as the average link bandwidth for domain 0 intra-domain links. This value will be shared with the Root RM, while the detailed intra-domain topology and data for domain 0 will be available only at RM-0. This value represent the communication capacity of a domain to avoid exposing the domain internal topology The Root RM will use the EDCC value obtained from different RMs to avoid domains with lower bandwidth while deciding on inter-domain routes.

- Estimated Domain Communication Waiting time (EDCWT): EDCWT for a certain RM is a metric for the internal traffic (congestion) over the links in the domain managed by

this RM. It is calculated as the average link waiting times for all intra-domain links managed by this RM. In Figure 1 example, the EDCWT at RM-0 is calculated as the average link waiting time for domain 0 intra-domain links. The Root RM will use the EDCWT value obtained from different RMs to avoid domains with high internal traffic while deciding on inter-domain routes.line.

4. OPTIMAL LOAD BALANCING TECHNIQUES

The load balancing process is carried out by running the load balancing algorithm in a distributed manner at load balancers from different domain and different hierarchical levels. The first step starts by executing the load balancing technique at the top level (assuming V) with the objective of minimizing the maximum completion time. The load balancing decisions are sent down the hierarchy to level V-1. In the following step, the load balancers at level V-1 runs the load balancing technique, and sends the results to level V-2. This step is repeated down the hierarchy until the load balancing decisions reaches Level-0 sites. This completes the load balancing process.

The objective of the load balancing algorithm is to maintain the load balancing by ensuring that all the computing units will finish the load processing at the same time. We assume that we have n geographically distributed sites. Datasets (loads) of size L_k to be processed at site K, where $k = 1..n$. Those datasets are to be decomposed into smaller subsets to be executed at different sites. We assume that decomposed data can be executed at any site using the same data processing algorithm. The optimization problem is to minimize the maximum completion time by deciding on portions of datasets to be executed at each site (either executed at sites belonging to the same domain or different domains).

In this section, two alternative load distribution algorithms considering networking resources availability were proposed: Network Aware Divisible Load Algorithm (NADLA) and Genetic Algorithm based Load Distribution (GA-LD). Before going forward in introducing those algorithms, a brief introduction of the basic Divisible Load Algorithm (DLA) will be provided.

4.1. DLA

The DLA provides an optimal load distribution based on linear mathematical model. According to DLA literature, the optimality principle states "To achieve optimal load distribution, it is necessary and sufficient that all the participating sinks should stop processing at the same time instant" Assuming n sites, each site is defined by its Processing speed *Ci* (time to process unit dataset) and Estimated Waiting Time *EWTi, i = 1,…,n*, and assuming a job with a total load *L* distributed among different site. The objective is to define a distribution for αi (the number of datasets to be process at site *i*) such that all the sites finish processing at the same time. The job Finish Time at site *i* can be calculate as $JFTi = Ci * \alpha i + EWTi.$.

By applying the DLT optimality principle, we have

$$JFT_i = JFT_j = JFT \quad \forall i, j \in 1, \dots, N, \qquad (1)$$

$$C_i * \alpha_i + EWT_i = C_j * \alpha_j + EWT_j = JFT, \qquad (2)$$

$$\alpha_i = (JFT - EWT_i)/C_i, \qquad (3)$$

but,

$$\sum_{x=1}^{n} \alpha_x = L \qquad (4)$$

By substituting from 3 in 4

$$\sum_{x=1}^{n} (JFT - EWT_x)/C_x = L \qquad (5)$$

$$JFT = \frac{L + \sum_{x=1}^{n} EWT_x/C_x}{\sum_{x=1}^{n} 1/C_x} \qquad (6)$$

Then,

$$\alpha_i = \frac{L + \sum_{x=1}^{n} EWT_x/C_x}{C_i \sum_{x=1}^{n} 1/C_x} - EWT_i/C_i \qquad (7)$$

Equation (7) gives the optimal load distribution as a function of the sites processing speed and estimated waiting time.

4.2. NADLA

Network Aware DLA extends the DLA to consider both computing and networking resources instead of just computing resources. The traditional DLA takes into account sites processing speed and estimated waiting time. It ignores totally the effect of the communication delay. This is acceptable in applications where data transfer time is negligible compared to jobs executions time. However, many e-science applications are becoming more data intensive. Therefore, designing network aware DLA becomes of vital importance.

The main idea of the network aware DLA is to iterate over the basic DLA to adjust its results by taking into account the networking delay. Let's consider the DLA as a black box taking Processing speed $C_i \forall i = 1, \dots, n$, Estimated Waiting Time ($EWT_i \forall i = 1, \dots, n$) and total load L as input parameters and produces optimal load distribution $\alpha_i, \forall i = 1, \dots, n$ (Eq. 12) and Job Finish Time JFT (Eq. 11). The pseudo-code of the Network Aware DLA algorithm is shown in Figure 2. After executing the DLA for the first time and calculating distribution $\alpha_i, \forall i = 1, \dots, n$ and JFT (Step 2), the algorithm calculates Network aware JFT ($JFT_i^{Network\ Aware}$) for each site i (Step 3). $JFT_i^{Network\ Aware}$ is the actual value for the job finish time after considering the data transfer overhead. The difference between the Network aware JFT ($JFT_i^{Network\ Aware}$) and the JFT calculated by the DLA is defined as the $Networking\_Delay_i$ (Step 6). Then, the DLA is applied to a new problem with inputs $L=0$ and $EWT_i = Networking\_Delay_i, \forall i = 1, \dots, n$ (Steps 7, 8 &9). The results of this problem instance are correction values for $\alpha_i, \forall i = 1, \dots, n$ and JFT to be added to them (Step 10 &11). The idea behind applying DLA with inputs $L=0$ and $EWT_i = Networking\_Delay_i$ is to consider the networking delay at each site as site waiting time and try to distribute this value equally among all the sites. $L=0$ guarantees that there is no added load and it is just a re-distribution of the original load (load balancing). The $\alpha_i^{correction}$ values can be either positive, zero or negative, and the summation of $\alpha_i^{correction}$ values over all the sites equals zero. The algorithm stops when the maximum value of the $Networking\_Delay$ overall the sites becomes less than a certain ϵ value. ϵ is a design parameter for the algorithm. Small ϵ value results in better JFT and longer algorithm execution time, while large ϵ value speeds up the algorithm execution and results in worst JFT.

1- Set C_i, EWT_i $\forall i = 1, \ldots, n$ and L according to up to date status information
2- Apply DLA to calculate the load distribution α_i, $\forall i = 1, \ldots, n$ (Eq. 12) and Job Finish Time JFT (Eq. 11)
3- Calculate $JFT_i^{Network\ Aware}$, $\forall i = 1, \ldots, n$ According to the load dist. α_i
4- While $(\max_{\forall i=1,\ldots,n}\{JFT_i^{Network\ Aware} - JFT\} > \epsilon)$
5- {
6- Calculate $Networking\_Delay_i = JFT_i^{Network\ Aware} - JFT, \forall i = 1, \ldots, n$
7- Set $EWT_i = Networking\_Delay_i \forall i = 1, \ldots, n$
8- Set $L = 0$
9- Apply DLA to calculate the load distribution $\alpha_i^{correction}$, $\forall i = 1, \ldots, n$ (Eq. 12) and Job Finish Time $JFT^{correction}$ (Eq. 11)
10- Set $\alpha_i \mathrel{+}= \alpha_i^{correction}$, $\forall i = 1, \ldots, n$
11- Set $JFT \mathrel{+}= JFT^{correction}$
12- Calculate $JFT_i^{Network\ Aware}$, $\forall i = 1, \ldots, n$ According to the load dist. α_i
13- }
14- Populate $\alpha_i, \forall i = 1, \ldots, n$

Figure 2 Network Aware DLA Algorithm

To calculate exact values for *JFT* with data transfer overhead $(JFT_i^{Network\ Aware})$, the allocated links as well as their capacities and waiting times should be known. At this step (load distribution), such data is not available since the resource allocation will be done in the next step. To solve this problem, estimated values for bandwidth and waiting time are calculated for each site. Those values are estimated as an average over all the links connected to this site. Therefore, two values are introduced for each site: site transfer bandwidth and site transfer waiting time. Site transfer bandwidth is calculate as the average bandwidth over all links connected to this site, while site transfer waiting time is calculate as the average waiting time over all links connected to this site. Those values are to be used as estimated values for any data transfer to/from this site.

4.3. GA_LD

A real number chromosome of length n (number of sites) is used for problem representation. The chromosome $\alpha_i \in [0, L]$ represents the load size to be executed at site I, where L is the total load size. Generally, real numbers representation can be implemented either directly as real numbers chromosome or as string of bits that map to real numbers. We choose to use real numbers directly since it outperforms the binary mapping for most problems.

Each chromosome is associated with a fitness value that represents the goodness of this solution. This fitness value is evaluated using a certain objective function. The objective function is to minimize the maximum *JFT* with data transfer overhead $(JFT_i^{Network\ Aware})$ over all the sites. The $(JFT_i^{Network\ Aware})$ is calculated as described in the previous section. In addition, the data transfer capacities and waiting times are estimated using the same described way.

An adjustment is done to the chromosome genes in all the generations before the fitness

calculation (Equation 8). The adjustment is done by modifying the gene values α_i such that $\sum_{i=1}^{n} \alpha_i = L$. This modification ensures that the summation of the loads assigned to all the sites equals to the total load. The objective of this adjustment is to adapt any unfeasible solution to be feasible one, and this speeds up the GA search significantly.

$$\alpha_i^{adjusted} = \left(\alpha_i^{old} \Big/ \sum_{i=1}^{n} \alpha_i^{old} \right) * L, \quad \forall i = 1, \dots, n \qquad (8)$$

5. EXPERIMENTAL RESULTS AND DISCUSSIONS

Simulations were conducted using OMNET++ network simulator (www.omnetpp.org). the performance of the three load balancing algorithms (GA-LD, NADLA, and DLA), measured by their impact on the overall Job Finish Time (*JFT*), Load Balancing Time, Standard Deviation of links utilization (SD_{Links}) and Standard Deviation of computing units utilization ($SD_{ComputingUnits}$) is compared for different network and application parameters.

5.1. Effect of network size

Figure 3 compares the performance of the three load distribution algorithms: GA-LD, DLA and NADLA for different network sizes. As shown in Figure 3a, the JFT using NADLA is better than the other two algorithms especially for larger network sizes (400 & 1024 sites networks). For example, the percentage of reduction in JFT using NADLA compared to DLA is about 10% in 16 sites network, while the same percentage approaches about 44% for 1024 sites network. Regarding the Load Balancing Time, the GA-LD takes much longer time than the other two algorithms for all network sizes. The Load Balancing Time for both DLA and NADLA is almost the same for all network sizes. Figure 3c shows no notable difference in the SD_{Links} among the three algorithms, while Figure 3d shows that DLA results in better $SD_{ComputingUnits}$ for all network sizes. This is expected since the optimality principle for the divisible load theory tries to balance the load among the computing units. For large network sizes, the computing units load balancing is much better using DLA and NADLA over the GA-LD.

5.2. Effect of Load Size

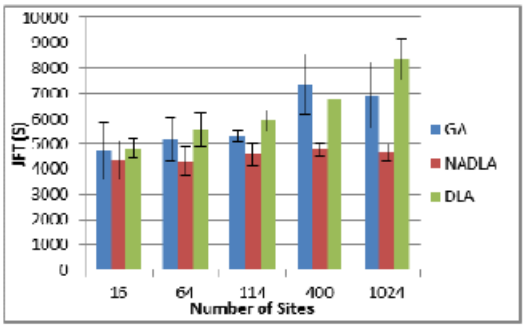
The performance of the algorithms is compared for different average load sizes. The network size is fixed to 64 sites network. Figure 4a shows that for small load sizes the JFT is almost the same for the three algorithms. As increasing the load size, the NADLA outperforms both GA-LD and DLA. Figure 4b reports no change in the Load Balancing Time as increasing the load size for the three algorithms.

5.3. Effect of Hierarchical Depth

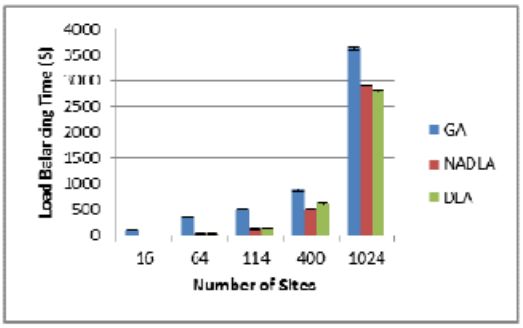
To evaluate the effect of the depth of hierarchy, networks with different number of hierarchical levels from 2 to 5 are considered. Figure 5a shows a small deduction in the JFT as increasing the hierarchy depth for the three algorithms. Figure 5b shows increase in the Load Balancing Time for GA-LD as increasing the hierarchy depth. On the other hand, the Load Balancing Time for NADLA and DLA decreases as increasing the hierarchy depth. This shows the advantage of using NADLA and DLA over GA-LD especially for larger hierarchy depth.

Those advantages in the Load Balancing Time as increasing the depth of the hierarchy come at the cost of increasing. the control overhead. Increasing the depth of the hierarchy increases the required number of RMs to manage the system for networks with the same size. For example, increasing the depth of the hierarchy from 2 to 5 increases the number of RMs by a factor of 7. Controlling and maintaining this hierarchical structure increases the cost and complexity as increasing the number of RMs. This increases the control overhead and communication

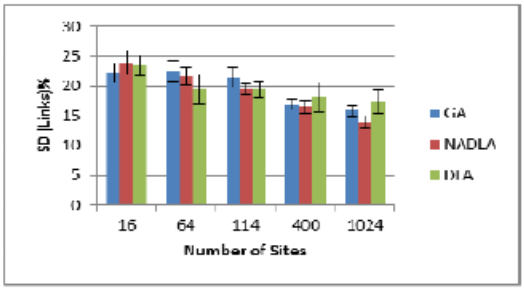
complexity. In [11], a study of a hierarchical routing protocol reported a notable increase in path setup time and communication overhead as increasing the depth of the hierarchy.



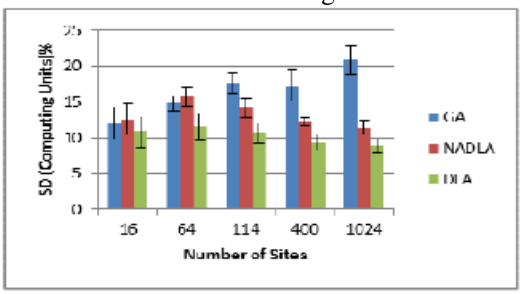
a. Job Finish Time



b. Load Balancing Time

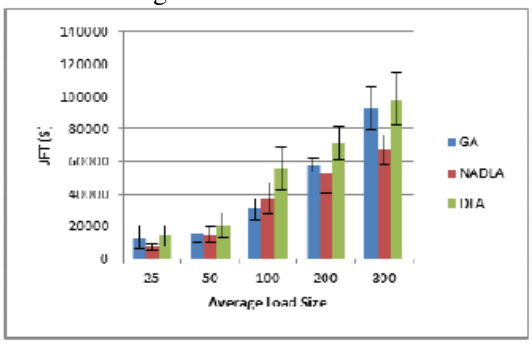


c. SD (Links)

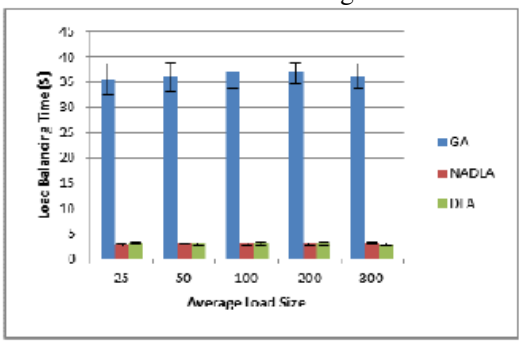


d. SD (Computing Units)

Figure 3 The Effect of network size for different load distribution algorithms

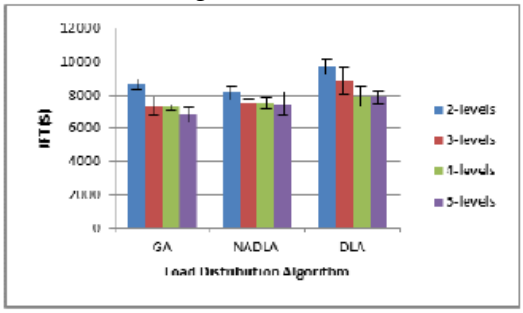


a. Job Finish Time

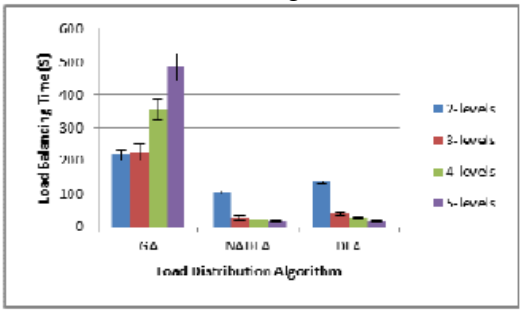


b. Load Balancing Time

Figure 4 Effect of load size for different load distribution algorithms



a. Job Finish Time



b. Load Balancing Time

Figure 5 The Effect of level of hierarchy for different load distribution algorithms

6. CONCLUSIONS

The proposed load balancing techniques as well as the hierarchical approach provided a novel solution for the load balancing problem in multi-domain optical Grid environments. The load balancing was carried out in a top down manner across the hierarchy, by which each domain executes the load balancing algorithm, and shares those results with the lower level load balancers. The hierarchical approach for load balancing maintained the scalability, privacy, and feasibility of the grid system. The proposed aggregation process helped in keeping the domain privacy while integrating with other domains. Domain internal topology and resource status information were kept internally, while sharing six aggregated and abstracted parameters with other domains. Those parameters were used in defining load distribution.

Two load balancing techniques (NADLA and GA-LD) were proposed and compared to the traditional DLA. NADLA extends DLA by taking into account network connectivity and computational and networking resources availability while deciding on load distribution. Simulation results showed that NADLA and GA-LD algorithms outperform the DLA in mostly all the situations. The GA-LD suffered from long scheduling time compared to DLA and NADLA. Studying the effect of the hierarchical depth shows that increasing the number of hierarchical levels results in better load balancing. This advantage came at the cost of increasing control overhead.

REFERENCES

[1] Lee, S.hyun. & Kim Mi Na, (2008) "This is my paper", ABC *Transactions on ECE*, Vol. 10, No. 5, pp120-122.

[2] Gizem, Aksahya & Ayese, Ozcan (2009) *Coomunications & Networks*, Network Books, ABC Publishers.

[1] S. Figuerola, N. Ciulli, M. d. Leenheer, Y. Demchenko, W. Ziegler, and A. Binczewski, "Phosphorus: single-step on-demand services across multi-domain networks for e-science," J. Wang, G.-K. Chang, Y. Itaya, and H. Zech, Eds., vol. 6784. SPIE, 2007, p. 67842.

[2] A. Takefusa, M. Hayashi, N. Nagatsu, H. Nakada, T. Kudoh, T. Miyamoto, T. Otani, H. Tanaka, M. Suzuki, Y. Sameshima, W. Imajuku, M. Jinno, Y. Takigawa, S. Okamoto, Y. Tanaka, and S. Sekiguchi, "G-lambda: Coordination of a grid scheduler and lambda path service over gmpls," Future Generation Computer Systems, vol. 22, no. 8, pp. 868 – 875, 2006.

[3] A. Willner, C. Barz, J. A. Garcia Espin, J. Ferrer Riera, S. Figuerola, and P. Martini, "Harmony - advance reservations in heterogeneous multi-domain environments," inProceedings of the 8th International IFIP-TC 6 Networking Conference, ser. NETWORKING '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 871–882.

[4] S. Viswanathan, B. Veeravalli, and T. G. Robertazzi, "Resource-aware distributed scheduling strategies for large-scale computational clus-ter/grid systems," Parallel and Distributed Systems, IEEE Transactions on, vol. 18, no. 10, pp. 1450–1461, 2007.

[5] C. Yu and D. Marinescu, "Algorithms for divisible load scheduling of data-intensive applications," Journal of Grid Computing, vol. 8, no. 1, pp. 133–155, 03/01 2010.

[6] P. Thysebaert, B. Volckaert, M. De Leenheer, F. De Turck, B. Dhoedt, and P. Demeester, "Dimensioning and on-line scheduling in lambda grids using divisible load concepts,"The Journal of Supercomputing, vol. 42, no. 1, pp. 59–82, 10/01 2007.

[7] M. Abouelela and M. El-Darieby, "Co-scheduling computational and networking resources in e-science optical grids," in GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference, 2010, pp. 1–5.

[8] S. Kim and J. B. Weissman, "A genetic algorithm based approach for scheduling decomposable data grid applications," pp. 406–413 vol.1, 2004.

[9] L. Marchal, Y. Yang, H. Casanova, and Y. Robert, "A realistic network/application model for scheduling divisible loads on large-scale platforms," inIPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers. Washington, DC, USA: IEEE Computer Society, 2005, p. 48.2.

[10] R. Shah, B. Veeravalli, and M. Misra, "On the design of adaptive and decentralized load balancing algorithms with load estimation for computational grid environments," Parallel and Distributed Systems, IEEE Transactions on, vol. 18, no. 12, pp. 1675–1686, 2007.

[11] M. El-Darieby, D. Petriu, and J. Rolia, "Load-balancing data traffic among inter-domain links," Selected Areas in Communications, IEEE Journal on, vol. 25, no. 5, pp. 1022–1033, 2007.

Authors

Mohamed Abouelela received his M.Sc. degree in Engineering Mathematics from Cairo University. He is now a research assistant and Ph.D. candidate affiliated with the Electronic System Engineering Department at University of Regina, Regina, Canada. His main interests include Scheduling Grid resources, Data management in grid systems, and optical grid networks.

Mohamed El-Darieby is an assistant professor of Engineering in University of Regina, Regina, Canada. He received B.Sc. and M.Sc. in electrical engineering from Zagazig University, Egypt and Ph.D in Systems and Computer Engineering at Carleton University, Ottawa, Canada. His research is in the areas of backbone networks, grid computing and wireless mesh networks. Dr. El-Darieby is a member of the Association of Professional Engineers and Geoscientists of Saskatchewan (APEGS).