Joint Flow Control and Processor Allocation for Task Scheduling in Grid over OBS Networks

Yahong Yang, Guiling Wu, Xinwan Li, *Member, IEEE* and Jianping Chen

***Abstract*—This paper presents a model for the joint design of flow control and processor allocation for task scheduling in grid over optical burst switching (GOBS) networks. The GOBS resource allocation is formulated as a utility maximization problem with constraints that arise from the resource and user preferences. The optimization problem is decomposed into two parts: flow control for burst length adjustment and processor allocation for computational capacity adjustment. The parameters from the resource layer are abstracted and provided to a cross-layer optimizer to maximize user's utility function. The performance of the algorithm is evaluated through simulations.**

***Index Terms*—Cross-layer design, Flow control, Grid, Load balancing, OBS**

I. Introduction

Grid is becoming more and more attractive for its ability to provide super computing capacities and better sharing of distributed resources. Optical burst switching (OBS) is considered as a promising solution for underlying grid networks because of its low delay, variable burst length and separate control [1]. In grid environment, resources are widely distributed and owned by many different organizations. A good resource management system is essential to exert the full advantage of grids. The management system is responsible for resource discovery, resource selection, task scheduling and resource maintenance and the scheduling algorithm is one of the most important issues [2]. Most of the existing grid resource allocation and scheduling algorithms mainly focus on isolated layers of the grid architecture. The inflexibility of the strict layering structure results in an inefficient utilization of the resources. In order to achieve optimum user performance, it is necessary to deal with the grid system as a whole since the performance experienced at the application layer strongly depends on the other layers. Cross-layer design is based on information exchange and joint optimization among the multiple layers [3]. This idea is well suited to grid environments where the grid resources are dynamic, autonomous and heterogeneous and the grid user requirements vary over time.

This paper proposes a joint flow control and processor allocation algorithm for GOBS task scheduling. Two requirement criteria, namely deadline and payment, are considered and formulated as a user utility maximization function. The burst flow control algorithm that controls the burst size [4] and processor allocation that partitions the number of system processors [5] are jointly studied to improve grid performance. The performance of the proposed algorithm is evaluated and compared with deadline and budget constrained (DBC) scheduling algorithm [6] by simulations.

The rest of the paper is arranged as followings: Section 2 presents GOBS task scheduling modeling and optimization solutions. In section 3 the simulations are conducted and a comparison is taken between the proposed algorithm and DBC algorithm. Section 4 concludes the paper.

II. Task Scheduling Modeling and Optimization Solutions

Consider a GOBS environment with Q nodes, where some nodes are edge nodes and some nodes act as core nodes. A part of the edge nodes connect F client hosts (denoted as $H_1, H_2, ..., H_F$) to submit user's requirement. The others connect M servers acting as computational resource (denoted as $R_1, R_2, ..., R_M$). Each computational resource includes a certain number of processors, which are different in terms of computational capacities and prices of CPU time. Suppose a grid user has a task consisting of N jobs (denoted as $J_1, J_2, ..., J_N$) to be processed in GOBS environment. He is desired to complete as many jobs as possible under a certain time (denoted as T_0) and budget (denoted as E_0) limits. The objective of the task scheduling is to choose the proper burst duration time and the number of processors so that the grid user utility is maximized subject to the resource and user preferences constraints. Without losing the generality, we made the following simplifications: (1) Jobs are inter-independent. (2) Once a job is initiated, it will be completed without preemption.

The user utility function we proposed is as follows:

Manuscript received May 15, 2009. This work is partially supported by National Science Foundation of China (NSFC) (ID90704002 and 60877012), 863 Project (ID2006AA01Z242 and 2007AA01Z275), Dawn Program for Excellent Scholars by the Shanghai Municipal Education Commission, STCSM Project (09JC1408100) and State Key Lab Project (GKZD030004).

The Authors are with the State Key Laboratory on Advanced Optical Communication Systems and Networks, Shanghai Jiaotong University, 800 Dongchuan Road, Shanghai, China (corresponding author to provide phone: +86-21-34205140; e-mail: jpchen62@sjtu.edu.cn).

978-1-4244-4785-5/09/$25.00 ©2009 IEEE

$$
\begin{aligned}
&Max\{\omega_1(T_0-\sum_{i=1}^{N}\sum_{s=1}^{F}\sum_{k=1}^{M}\frac{\phi_i^{sk}d_i}{B_s^kC/T_a}-\sum_{i=1}^{N}\sum_{k=1}^{M}\frac{\varphi_i^kb_i}{Z_k})\\
&\quad+\omega_2(E_0-v\sum_{i=1}^{N}\sum_{s=1}^{F}\sum_{k=1}^{M}\frac{\phi_i^{sk}d_i}{B_s^kC/T_a}-\sum_{i=1}^{N}\sum_{k=1}^{M}c_k\frac{Z_k}{P_k}\times\frac{\varphi_i^kb_i}{Z_k})\}\\
&s.t.\quad \sum_{s=1}^{F}B_s^k\le\delta T_a\\
&\quad \sum_{k=1}^{M}Z_k\le\sum_{k=1}^{M}\alpha_kP_k\\
&\quad \phi_i^{sk}=0\ or\ 1\quad i=1,2,...,N;\ s=1,2,...,F;\\
&\qquad\qquad k=1,2,...,M\\
&\quad \varphi_i^k=0\ or\ 1\quad i=1,2,...,N;\ k=1,2,...,M\\
&\quad B_s^k>0,\ \ Z_k>0
\end{aligned}
\tag{1}
$$

where w_1 and w_2 stand for the weights of deadline and payment, respectively. They provide the clients with the right to flexibly specify their weights for money and time [3]. The deadline is the overall time spent by a task in the grid. The task deadline can be broken into two parts: computational time and transmission time. Payments include those for computational resource and network bandwidth resource. d_i is the data size of the i th job. If job i is really transmitted from client node s to destination node k, then the value of ϕ_i^{sk} will be 1, otherwise ϕ_i^{sk} is 0. b_i is the computational quantity of the i th job. If we really assign job i to computational resource k, then the value of φ_i^k will be 1, otherwise φ_i^k is 0. B_s^kC/T_a denotes the transmission rate in the path from the client node s to the destination node k, where B_s^k is the burst duration time of flows of this path, T_a is the timer value in a timer-based burst assembler and C is the capacity of the wavelength channel. Z_k denotes the required computational capacity from resource k. v is the cost per unit time of OBS network. c_k is the cost of individual processor per unit CPU time of the k th computational resource and P_k is the computational capacity of individual processor. c_kZ_k/P_k indicates the required cost of processors per unit CPU time when the computational capacity Z_k is assigned to jobs. The constraint condition implies that the aggregate burst duration does not exceed the timer value δT_a while the aggregate computational capacity does not exceed the total computational capacity. $\delta\in(0,1]$ is a target utilization [4]. One can attain a desired compromise between burst loss ratio and goodput by adjusting the value of δ. α_k is the number of processors of the k th computational resource.

Consider the Lagrangian multipliers μ_s^k and η_k as, respectively:

$$
\mu_s^k=\delta T_a-\sum_{s=1}^{F}B_s^k \tag{2}
$$

$$
\eta_k=\left|L_k-\sum_{k=1}^{M}L_k\Big/M\right| \tag{3}
$$

where μ_s^k represents the queuing delay induced by congestion on the path from the source node s to the destination node k. The congestion price is increased when its corresponding wavelength is congested and decreased when under-utilized. η_k represents the difference between the load on computational resource k and the average load, where the load is described by the time consumed on computational resource. It reflects the load balancing of each computational resource and the less the difference the more balanced the resource. L_k is the time spent on the computational resource k. M is the number of computational resources.

Then the Lagrangian form of this optimization problem is as follows:

$$
\begin{aligned}
&L(B_s^k,Z_k;\mu_s^k,\eta_k)\\
&=\omega_1(T_0-\sum_{i=1}^{N}\sum_{s=1}^{F}\sum_{k=1}^{M}\frac{\phi_i^{sk}d_i}{B_s^kC/T_a}-\sum_{i=1}^{N}\sum_{k=1}^{M}\frac{\varphi_i^kb_i}{Z_k})\\
&\quad+\omega_2(E_0-v\sum_{i=1}^{N}\sum_{s=1}^{F}\sum_{k=1}^{M}\frac{\phi_i^{sk}d_i}{B_s^kC/T_a}-\sum_{i=1}^{N}\sum_{k=1}^{M}c_k\times\frac{\varphi_i^kb_i}{P_k})\\
&\quad-(\sum_{s=1}^{F}\sum_{k=1}^{M}\mu_s^kB_s^k-\delta T_a)-\sum_{k=1}^{M}(\eta_kZ_k-\alpha_kP_k)
\end{aligned}
\tag{4}
$$

From Karush-Kuhn-Tucker theorem, we know that the optimal solution is given $\partial L(B,Z)/\partial B=0$ for $\mu_s^k\ge0$, i.e.,

$$
\frac{\partial L(B_s^k,Z_k)}{\partial B_s^k}=(\omega_1+\omega_2\times v)\frac{T_a\sum_{i=1}^{N}\phi_i^{sk}d_i}{(B_s^k)^2C}-\mu_s^k \tag{5}
$$

Then we can obtain the unique optimal burst length $(B_s^k)^*$ that maximizes the grid user utility

$$
(B_s^k)^*=\sqrt{\frac{(\omega_1+\omega_2\times v)\times T_a\sum_{i=1}^{N}\phi_i^{sk}d_i}{\mu_s^kC}} \tag{6}
$$

The processor allocation problem is solved in the similar method. Let $\partial L(y,Z)/\partial Z=0$, then

$$
\frac{\partial L(B_s^k,Z_k)}{\partial Z_k}=\frac{\omega_1\sum_{i=1}^{N}\varphi_i^kb_i^k}{(Z_k)^2}-\eta_k=0 \tag{7}
$$

We can get the optimal computational capacity:

$$
Z_k=\sqrt{\omega_1\sum_{i=1}^{N}\varphi_i^kb_i^k\Big/\eta_k} \tag{8}
$$

Then the unique optimal the number of processors β_k^* maximizing the grid user utility is:

$$
\beta_k^*=\left\lceil\sqrt{\omega_1\sum_{i=1}^{N}\varphi_i^kb_i^k\Big/\eta_k}\Big/P_k\right\rceil,\qquad \beta_k^*\le\alpha_k \tag{9}
$$

By the above analysis, the joint flow-control and processor-allocation (JFCPA) algorithm for GOBS task scheduling is designed as follows:

1. At the grid OBS core node, the queuing delay μ_s^k is

periodically collected and sent to the cross-layer optimizer and the burst assemblers at the OBS edge nodes through the control channel, respectively.

2. At the grid computational resource node, the load balancing factor η_k of each resource is periodically collected and sent to a cross-layer optimizer through the control channel.
3. At the cross-layer optimizer, the user utility, taking into account previously assigned jobs and current grid state (received μ_s^k and η_k), is calculated for each resource. The burst duration time and computational resource with the maximal user utility are selected and assigned to the scheduled job. The message is then sent to the corresponding burst assembler and computational resource node through the control channel.
4. The grid burst assembler updates the burst duration time based locally measurable quantities and the received messages.
5. The grid computational resource node updates its number of processors based locally measurable quantities and the received the message.
6. Repeat the above steps for each job until the current time or expense is beyond the deadline and budget limits.

III. Simulation Analysis

Simulations are carried out on a centralized GOBS platform to compare the performance of JFCPA algorithm and DBC algorithm under budget and deadline constraint.

The OBS based grid topology is shown in Fig. 1, which is from Shanghai Education & Research Network (SHERNET) [7]. It includes 12 edge nodes, 12 core nodes and 48 hosts. The 16 hosts connecting to the four edge nodes located at Shanghai Jiao Tong University (SJTU), Fudan University, Tongji University and East China Normal University, respectively, serve as computational resource hosts. The rest 32 hosts serve as grid user hosts. The edge node has a grid agent that receives job requests and sends them to the grid scheduling node (GSN) through burst control channels, which is a cross-layer optimizer. The GSN, located at SJTU, collects job requests and sends them to the scheduler module of GSN, which carries out task scheduling according to certain scheduling strategy and relative resource information. After completing scheduling, the GSN sends back a response packet to the user through burst control channels, which includes destination of computational resource, network path, etc. Once a user receives the response packet, it begins to send application data using allocated resource by just enough time (JET) protocol [8]. The computational resources begin to deal with relevant applications after receiving the data and return the processing result to the user through traditional OBS protocol. The average data size is 2 Mbytes. The characteristics of computational resources simulated are referred to [6]. The network resource price per unit time is assumed to be two units of money. The assembly timer value is $T_a = 1ms$. The value of δ is set to be 1.0.

Fig. 2 shows the user utility as a function of the deadline. Two average required computational quantities for each application, namely 20 MI (million instructions) and 1000 MI are adopted, where 20 MI represents the case that computational time at server is much shorter than data transmission delay while 1000 MI represents the case that computational time at server is much larger than data transmission delay. As shown in Fig. 2, all user utilities increase with the deadline since a larger deadline brings out more successfully completed jobs. Furthermore, regardless of DBC or JFCPA, the user utility for the average required computational quantity of 20 MI is higher than that for the average required computational quantity of 1000 MI. The reason is that a larger computational quantity results in the completion time quick to exceed deadline constraint so that many job requests are refused. Under these two required computational quantity, JFCPA is superior to DBC. This is due

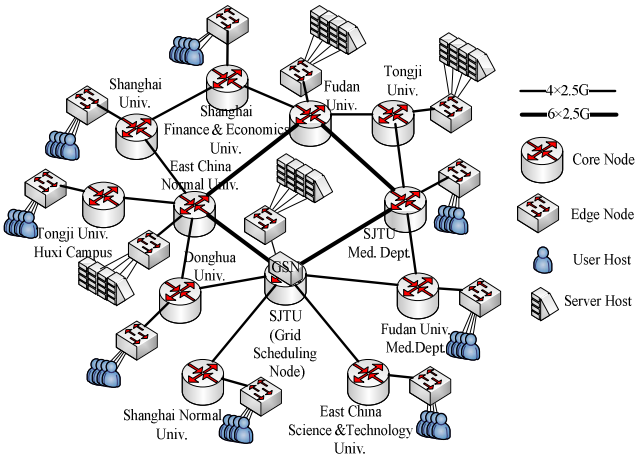

Fig. 1. Simulation Topology.

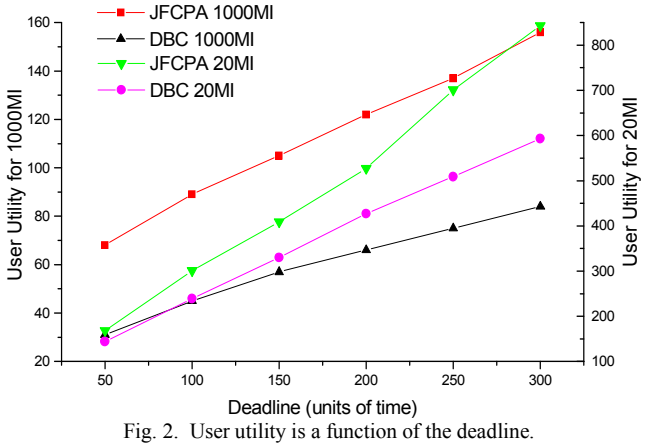

Fig. 2. User utility is a function of the deadline.

to the fact that JFCPA chooses proper grid resources to transmit and process jobs at server, which induces that its jobs can be processed and replied fast so that more job requests are served.

Load balancing degree represents the ratio of the average computational resource load to the maximum resource load. It is used to evaluate the performance of computational resource. As shown in Fig. 3, the load balancing degree of JFCPA is better than that of DBC for the average required computational quantity of 1000 MI. For the average required computational

quantity of 20 MI, there are two different cases. For small deadline (less than about 100 in Fig. 3), the load balancing degree of DBC is higher than that of JFCPA. While for large deadline, the latter becomes better. This can be explained as follows. For the average required computational quantity of 1000 MI, the computational resource has more obvious effect

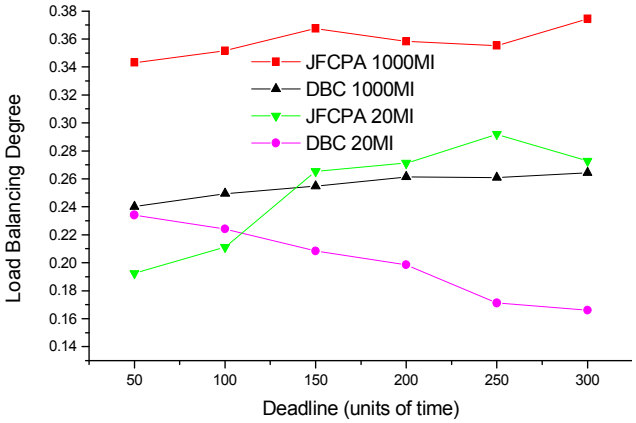

Fig. 3. The effect of deadline on the load balancing degree.

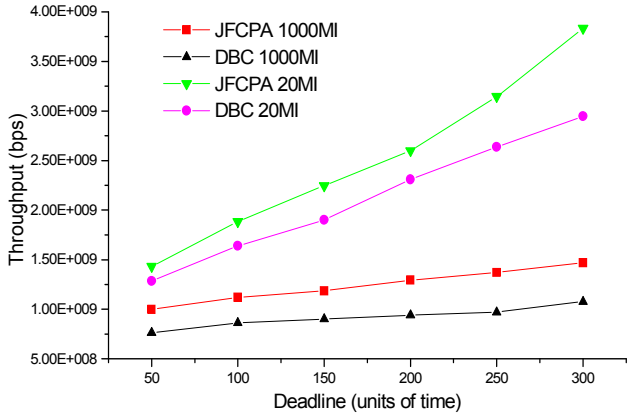

Fig. 4. Dependence of the throughput on the deadline.

than network resource. JFCPA adopts processor allocation policy so that the grid scheduler would like to select idle or light-load computational resource. When average required computational quantity is 20 MI, the bottleneck of grid consists in network resource. In this case JFCPA adopts flow control policy regardless of computational resource resulting in the low load balancing degree. As the deadline increases, the successfully completed jobs also increase and then the processor allocation policy slowly imposes on load balancing.

Fig. 4 examines the throughput with the deadline. Under the mentioned required computational quantities, JFCPA achieves higher throughput than DBC. This is obvious because JFCPA completes more jobs than DBC. At the same time, no matter what JFCPA or DBC algorithm, the case in average required computational quantity of 20 MI achieves higher throughput than the case in average required computational quantity of 1000 MI. The reason is that the former provides higher user utility than the latter (see Fig. 2).

IV. Conclusion

We present a model for the joint design of flow control and processor allocation for task scheduling in grid over OBS networks, where the resulting JFCPA algorithm is to solve a utility maximization problem with constraints from the resource and users. The performance of the proposed algorithm is compared with the DBC algorithm by simulations. Results show that the proposed algorithm can dynamically adjust the grid resources according to the feedback information of grid environment to enhance grid performance

References

[1] C. Qiao, and M. Yoo, "Optical burst switching-a new paradigm for an optical internet," J. High Speed Netw., vol. 8, pp. 69–84, Mar. 1999.

[2] H. Feng, G. Song, Y. Zheng, and J. Xia, "A deadline and budget constrained cost-time optimization algorithm for scheduling dependent tasks in grid computing," Grid Coop. Comput., vol. 3033, pp. 113–120, 2004.

[3] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-layer design for wireless networks," IEEE Commun. Mag., vol. 41, pp. 74–80, Oct. 2003.

[4] W.-S. Park, M. Shin, and S. Chong, "Performance enhancement in OBS network with flow control and edge delay method," in Proc. IEEE GLOBECOM, USA, 2006, pp. 1–6.

[5] C. McCann, and J. Zahorjan, "Processor allocation policies for message-passing parallel computers," in Proc. ACM SIGMETRICS conference on Measurement and modeling of computer systems, USA, 1994, pp. 19–32.

[6] R. Buyya, M. Murshed, and D. Abramson, "A deadline and budget constrained cost-time optimization algorithm for scheduling task farming applications on global grids," Computing Research Repository cs. DC/023020, 2002.

[7] W. Dai, G. Wu, Y. Yang, and J. Chen, "Resource scheduling in centralized OBS-based grids," J. Opt. Netw., vol. 7, pp. 111–118, Feb. 2008.

[8] M. Yoo, and C. Qiao, "Just-Enough-Time (JET): A high speed protocol for bursty traffic in optical networks," in Proc. IEEE/LEOS Technologies for a global information infrastructure, Canada, 1997, pp. 26–27.