

Modeling and performance evaluation of QoS-aware job scheduling of computational grids

Shan Zhiguang¹ Lin Chuang²

(¹Department of Informatization Research, State Information Center, Beijing 100045, China)

(²Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: To achieve high quality of service (QoS) on computational grids, the QoS-aware job scheduling is investigated for a hierarchical decentralized grid architecture that consists of multilevel schedulers. An integrated QoS-aware job dispatching policy is proposed, which correlates priorities of incoming jobs used for job selecting at the local scheduler of the grid node with the job dispatching policies at the global scheduler for computational grids. The stochastic high-level Petri net (SHLPN) model of a two-level hierarchy computational grid architecture is presented, and a model refinement is made to reduce the complexity of the model solution. A performance analysis technique based on the SHLPN is proposed to investigate the QoS-aware job scheduling policy. Numerical results show that the QoS-aware job dispatching policy outperforms the QoS-unaware job dispatching policy in balancing the high-priority jobs, and thus enables priority-based QoS.

Key words: computational grids; job scheduling; quality of service (QoS); performance evaluation; modeling; stochastic high-level Petri net (SHLPN)

Grid computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and high-performance orientation. It is concerned with coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations^[1]. A grid integrates and coordinates resources and users that live within different control domains with the goal of delivering various quality of service (QoS).

Job scheduling is a fundamental issue in achieving high performance on computational grids. It is defined as the process of making scheduling decisions for the incoming jobs involving resources over multiple administrative domains. In general, a job in grid scheduling can be defined to be anything that needs a resource, from a resource request, to a set of applications. And the term resource can mean anything that can be scheduled: a machine, disk space, a QoS network, etc. To schedule a submitted job to the “best” resource that the job can use, typically a computing element, is a complex task. Many research works have focused on job-scheduling algorithms for the grid to achieve satisfactory performance and deliver QoS^[2–4].

This paper focuses on the parallel job scheduling for a hierarchical decentralized grid architecture that consists of multilevel schedulers, as well as heterogene-

ous and geographically distributed resources. In a hierarchical organization, the scheduling controllers are organized in a hierarchy.

This paper introduces mechanisms to correlate priorities of incoming jobs used for back-end server scheduling at the local scheduler (LS) with the job dispatching policies at the global scheduler (GS) for computational grids. This approach enables priority-based QoS. Modeling and performance evaluation of job scheduling on the grids^[5–7] can provide quantitative performance analysis and prediction, and, as such, they are of immense importance towards the establishment of grid infrastructure. This paper presents a modeling and analysis technique based on the stochastic high-level Petri net (SHLPN)^[8] to investigate QoS-aware job scheduling on computational grids.

1 Job Scheduling Architecture

Our study focuses on hierarchical job scheduling architecture of computational grids. For convenience, we consider a two-level hierarchy job scheduling architecture as depicted in Fig. 1.

As shown in Fig. 1, a computational grid system has two-level schedulers: the GS and the LS. The GS acts as a first-level scheduler and has full control over the scheduling of all incoming jobs. It receives the incoming jobs and classifies them into several categories based on their importance, for example, some jobs may have more stringent real-time requirements, or some users may have paid more money for their grid services. The GS can dispatch an incoming job to any grid node according to a job dispatching policy. The GS can con-

Received 2007-04-25.

Foundation item: The National Natural Science Foundation of China (No. 60673054, 90412012).

Biography: Shan Zhiguang (1974—), male, doctor, associate professor, shanzg@mx.cei.gov.cn.

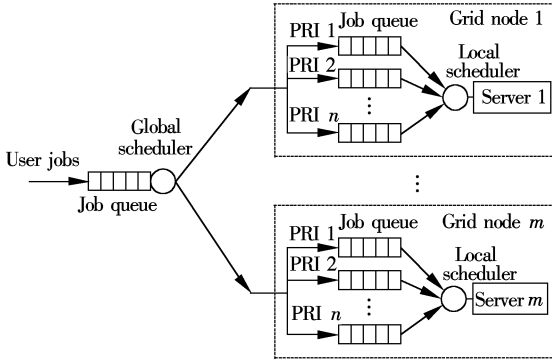


Fig. 1 Job scheduling architecture on computational grids

sider many factors in job dispatching, such as the requirements and preferences of the clients, the status of the resources, the current load of the server, etc. The LS acts as a second-level scheduler and is responsible for selecting a waiting job from the multiple queues of the server according to a job-selecting policy. Different categories of jobs are assigned different priorities and are serviced accordingly.

On the grid nodes, the job classification and prioritization mechanism is consistent with that of the GS. Each grid node provides a set of job waiting queues, one for each priority level. Jobs of the same priority level join the same waiting queue. A threshold is set for each waiting queue to limit the maximum number of waiting jobs. Each node has a single resource server, e. g., a processor, and it handles the jobs selected by the LS.

2 SHLPN Modeling

The reader is assumed to have some basic knowledge of Petri nets such as that given in Ref. [9]. SHLPNs^[8] are high-level Petri nets (HLPN) augmented with exponentially distributed transition firing times. The tokens of an SHLPN model may be atomic or compound tokens that may have multiple attributes. The tokens of the same type run in the same subnet and have the same behavior as well as the same firing rate. A marking of the SHLPN model is a distribution of tokens in each place of the model. A transition represents a class of possible changes of markings and may be associated with an enabling predicate expressed in terms of the token variables or place marking expressions. A transition is enabled whenever the predicates associated with it are satisfied. The state space of a model consists of the set of all markings reachable from the initial marking through the occurrence of transition firing. An SHLPN is homomorphic to a continuous time Markov chain (MC), and there is a one-to-one relationship between markings of the SHLPN and states of the MC.

2.1 System model

We make the following general assumptions for

the grid architecture in Fig. 1.

① User jobs can be classified into n categories and assigned n priority levels accordingly. Priority i is higher than priority k , for $1 \leq i < k \leq n$. Jobs with priority i are denoted by J_i .

② The system is composed of m computing nodes, i. e. m resource servers. The j -th grid node and the j -th server are denoted by N_j and S_j , respectively ($1 \leq j \leq m$). Each node contains n job queues, one for each priority level. Jobs in the same waiting queue have the same priority and are scheduled in a first-come-first-served (FCFS) manner. The n waiting queues contest the single server for service.

③ User jobs are assumed to be submitted according to the Poisson distribution with mean arrival rate λ_i for jobs J_i . A computing node cannot accept any more jobs of a priority level if the corresponding waiting queue has reached its capacity.

④ The service discipline of each server is nonpre-emptive. The service times of jobs in different waiting queues are assumed to be exponentially distributed with different means.

⑤ The GS is assumed to be run frequently enough, so that the queuing delay in its job queue is not considered in our SHLPN modeling. Also, the transmission time of jobs from the GS to the nodes is assumed negligible in a high-speed grid environment.

In Fig. 2, an SHLPN model of the system in Fig. 1 is proposed. In this model, the rectangles denote timed transitions and the black bars denote immediate transitions; circles denote places, and they contain tokens that are denoted by black dots. Jobs arriving and being executed are represented by timed transitions, which are associated with exponentially distributed firing delays. Jobs being dispatched to the grid nodes or competing for mutually exclusive processor resources are represented by immediate transitions, which fire in zero time and can be associated with firing probabilities. Job queues are represented by places, and the numbers of waiting jobs in the queues are represented by the marking of those places. Tokens in this model represent either jobs or processors, and different classes of jobs can be represented by tokens of different colors.

The meanings of the transitions and places in Fig. 2 are described in the following ($1 \leq i \leq n, 1 \leq j \leq m$):

- Transitions

c_i models the arrival of jobs J_i at rate λ_i .

d_{ij} models dispatching jobs J_i to node N_j (server S_j) by the GS. It can be associated with firing probabilities since there may be multiple destination nodes available for an incoming job.

h_{ij} models selecting the jobs J_i to be executed by the LS in node N_j .

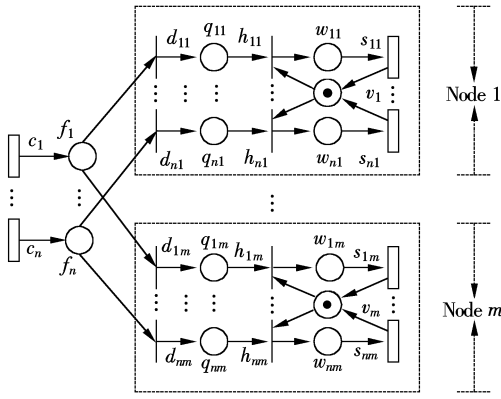


Fig. 2 An SHLPN model of the grid system in Fig. 1

s_{ij} models executing jobs J_i by server S_j . Its mean service rate is at μ_{ij} .

- Places

f_i models the partial function of the GS to distribute jobs J_i . The aggregation of $f_i (1 \leq i \leq n)$ models the GS.

q_{ij} models the job waiting queue of priority i in node N_j with capacity b_{ij} .

w_{ij} models the partial function of the LS in node N_j to select a job J_i to be executed by server S_j . The aggregation of $w_{ij} (1 \leq i \leq n)$ models the LS in node N_j .

v_j models the server resource of node N_j . The token in v_j represents the single server processor.

2.2 Model refinement

To analyze the performance of the SHLPN model in Fig. 2, we can construct the corresponding MC of the model. Based on the MC and state transition rates, we can construct the state transition matrix and obtain all the steady state probabilities to calculate the performance measurements. The software package SPNP^[10] was developed based on this idea. SPNP can be directly used for the performance analysis of the SHLPN model when the size of the model is not too large. However, the SHLPN model with n places of queues is generally equivalent to an n -dimensional MC. So the MC of the model in Fig. 2 is with $m \times n$ dimensions. The state space of the MC grows exponentially with the increase of m, n and b_{ij} . When the state space is large enough, solving the state equations of the SHLPN model is impossible for practical computing systems.

In order to reduce the complexity of the model solution, we simplify the structure of the SHLPN model in Fig. 2 using transition enabling predicates and rate functions. In this way, a complicated model can be equivalently transformed into a compact model. Fig. 3 shows the refined model of Fig. 2.

Compared with Fig. 2, Fig. 3 has deleted places w_{ij}, v_j , transitions h_{ij} , and some relevant arcs and tokens ($1 \leq i \leq n, 1 \leq j \leq m$). The competition relations among the priority queues for the single server that originally expressed by those discarded elements are directly de-

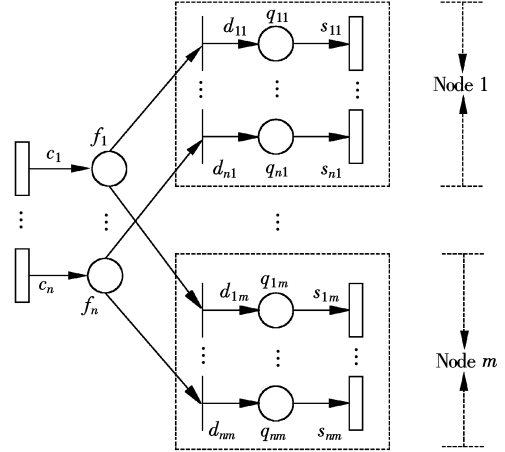


Fig. 3 A refined SHLPN model

scribed in the enabling predicates and firing probabilities of transition s_{ij} . Consequently, in Fig. 3, whether transition s_{ij} is enabled or not, it not only depends on the marking of the place q_{ij} , but also depends on the markings of other places in the same node.

3 Job Scheduling Policies and Performance Metrics

We first introduce the job dispatching policy at the GS and the job selecting policy at the LS, then propose a QoS-aware job dispatching policy based the former two policies. Metrics considered in the performance analysis are also formulated.

3.1 Job selecting policy

We consider the strict priority (SP) scheduling policy for the LS to select jobs from the multiple waiting queues of a computing node. This is a QoS-based policy. It schedules all higher priority jobs before lower priority jobs even when low priority jobs are waiting. The strategy of SP scheduling can be described in the enabling predicate and firing probability of transition s_{ij} in Fig. 3.

The enabling predicate of s_{ij} is

$$(M(q_{ij}) > 0) \wedge (\forall k, 1 \leq k \leq i, M(q_{kj}) = 0)$$

Specially, the enabling predicate of s_{1j} is simply $M(q_{1j}) > 0$.

The firing probability of s_{ij} is

$$P(s_{ij}) = \begin{cases} 1 & \text{if } i \in Q \\ 0 & \text{otherwise} \end{cases}$$

where $Q = \{k \mid M(q_{kj}) > 0 \text{ and } M(q_{lj}) = 0, \text{ for } \forall l, 1 \leq l < k \leq n\}$.

Specially, The firing probability of s_{1j} is

$$P(s_{1j}) = \begin{cases} 1 & \text{if } M(q_{1j}) > 0 \\ 0 & \text{if } M(q_{1j}) = 0 \end{cases}$$

3.2 QoS-unaware job dispatching policy

We consider the typical fewest jobs first (FJF) scheduling policy for the GS to dispatch jobs to a computing node. In this policy, the node with the fewest

waiting jobs is regarded as least loaded and having the most computing resources. The dispatcher GS does not consider the priority levels of jobs, and job classification and prioritization are only performed at the servers. The FJF strategy can be expressed by the enabling predicate and firing probability of transition d_{ij} in Fig. 3. Note that the categories and priorities of the jobs are not explicitly known before they enter the servers. The aggregate of the ready queues in server S_j is denoted by q_j with capacity b_j , and

$$b_j = \sum_{i=1}^n b_{ij}, \quad M(q_j) = \sum_{k=1}^n M(q_{kj})$$

The enabling predicate of d_{ij} is $(M(q_j) < b_j) \wedge (\forall k, 1 \leq k \neq j \leq m, (M(q_j) \leq M(q_k)) \vee (M(q_k) = b_k))$.

The firing probability of d_{ij} is

$$P(d_{ij}) = \begin{cases} \frac{1}{\|Q\|} & \text{if } j \in Q \\ 0 & \text{otherwise} \end{cases}$$

where $Q = \{k \mid M(q_k) = \min(M(q_1), M(q_2), \dots, M(q_m)) \text{ and } M(q_k) < b_k\}$.

This load balancing policy is based on the server load conditions, and a mechanism is required to dynamically feedback the total number of waiting jobs of the servers to the GS.

3.3 QoS-aware job dispatching policy

In general, there is a fundamental mismatch between the QoS-unaware job dispatching policy FJF and the QoS-aware LS scheduling policy SP when they are used together. For instance, the priority-unaware policy FJF may dispatch a job to an unavailable node, where the job queue with the priority level that matches the incoming job has reached its capacity, even though the node has the fewest number of waiting jobs. The reason is that, for an incoming job, only those waiting jobs with higher or equal priorities in the node can affect its waiting time if it is dispatched there, whereas the waiting jobs with lower priorities do not matter since they can only be executed if no higher-priority jobs exist in any higher levels with the SP policy. Consequently, the priorities of jobs must be taken into consideration in the job dispatching policy at the GS.

We propose an extended FJF (E-FJF) policy to implement priority-based QoS-aware job dispatching. In this policy, FJF is modified to consider the priority levels of jobs in dispatching and cooperating with SP scheduling at the LS. For an incoming job, not all the waiting jobs in a node but those with higher or equal priorities are counted to evaluate the node load, while the jobs with lower priorities are omitted for they cannot be executed prior to the job according to the SP policy.

The E-FJF strategy can be expressed by the enabling predicate and firing probability of transition d_{ij} in

Fig. 3.

The aggregate of the job waiting queues in node N_j is denoted by q_j . We define the variable $M_i(q_j)$ to denote the number of waiting jobs in node N_j whose priority level is higher than or equal to that of the incoming job J_i , and

$$M_i(q_j) = \sum_{x=1}^i M(q_{jx})$$

The FTF strategy can be expressed by the enabling predicate and firing probability of transition d_{ij} in Fig. 3.

The enabling predicate of d_{ij} is $(M(q_j) < b_j) \wedge (\forall k, 1 \leq k \neq j \leq m, (M_i(q_j) \leq M_i(q_k)) \vee M(q_{ik}) = b_{ik})$.

The firing probability of d_{ij} is

$$P(d_{ij}) = \begin{cases} \frac{1}{\|Q\|} & \text{if } j \in Q \\ 0 & \text{otherwise} \end{cases}$$

where $Q = \{k \mid M_i(q_k) = \min(M_i(q_1), M_i(q_2), \dots, M_i(q_m)) \text{ and } M(q_{ik}) < b_{ik}\}$.

This policy is based on the load conditions of the computing nodes on the grid, and a mechanism is assumed to dynamically feedback the number of waiting jobs at each priority level to the GS.

3.4 Performance metrics

In SHLPN models, the performance measurements can be obtained based on the steady state probabilities. We introduce the metrics used in this paper for evaluating the performance of the policies.

The throughput of jobs is an important performance measurement. $P[M]$ is the steady state probability for the marking M . The throughput $\text{TH}(s_{ij})$ of transition s_{ij} is

$$\text{TH}(s_{ij}) = \mu_{ij} \sum_{M \in E} P[M] \quad (1)$$

where E is the set of markings under which transition s_{ij} is enabled, and the enabling condition is described in the enabling predicate of transition s_{ij} .

The throughput TH_i of job J_i in the grid system is

$$\text{TH}_i = \sum_{j=1}^m \text{TH}(s_{ij}) \quad (2)$$

The throughput TH of the whole grid system is

$$\text{TH} = \sum_{i=1}^n \sum_{j=1}^m \text{TH}(s_{ij}) \quad (3)$$

Response time of jobs is another important performance metric. In this paper, we only investigate the response time of the computing node, i. e., the elapsed time of a job waiting in the job queue and being executed by the server, whereas the dispatching time by the GS and the transmission time in the networking infrastructure are not considered.

$D(q)$ is the average number of tokens in place q . The response time RT_{ij} by server transition s_{ij} is

$$RT_{ij} = D(q_{ij}) / TH(s_{ij}) \quad (4)$$

The response time RT_i of job J_i is

$$RT_i = \sum_{j=1}^m D(q_{ij}) / \sum_{j=1}^m TH(s_{ij}) \quad (5)$$

4 Numerical Results

We use the software package SPNP to analyze the SHLPN model in Fig. 3. In the example, the SP policy is used with the QoS-unaware job dispatching policy FTF and the QoS-aware policy E-FJF respectively to investigate the performance difference. To avoid the state space explosion and simplify the model solution, without loss of generality, we only consider a computational grid system consisting of two computing nodes and the jobs are assigned two priority levels in our example. Specifically, class 1 jobs are attached with a high priority and class 2 jobs are attached with a low priority. The thresholds of the jobs in the waiting queues are $b_{11} = b_{21} = b_{12} = b_{22} = 12$. We assume the two node servers are heterogeneous and each server can provide different classes of jobs with different service rates: $\mu_{11} = 25$ jobs/s, $\mu_{21} = 20$ jobs/s, $\mu_{12} = 20$ jobs/s and $\mu_{22} = 16$ jobs/s. These service rates are assumed to have been known beforehand by gathering the statistics. The arrival rates of class 1 and class 2 jobs monotonically increase from 5 to 40 jobs/s, and $\lambda_1 = \lambda_2$. This results in a maximum combined job rate $\lambda = \lambda_1 + \lambda_2 = 80$ jobs/s.

Fig. 4 plots the throughput of jobs in jobs/s as a function of the total job arrival rate (which is the sum of class 1 and class 2 jobs). It shows that there is not much difference in the throughput of TH_1 , TH_2 , and TH (for class 1, class 2 and the total jobs, respectively) between the QoS-aware load balancing policy E-FJF and the QoS-unaware load balancing policy FJF. More specifically, according to the numerical results in Fig. 4, TH_1 with the E-FJF policy is actually always greater than that with the FJF policy but the increment is only 0.05% at most, whereas TH_2 with the E-FJF policy is always smaller than that with the FJF policy and the decrement is only 0.09% at most. This shows that the performance benefit to the throughput of the high-priority jobs from the QoS-aware job dispatching policy E-FJF is not very conspicuous.

Fig. 5 plots the response time RT_1 of the high-priority jobs with the QoS-aware job dispatching policy E-FJF and the QoS-unaware policy FJF. As shown in Fig. 5, RT_1 with the E-FJF policy is always less than that with the FJF policy, and the decrement is 11.94% at most. Fig. 6 plots the response time RT_2 of the low-priority jobs with the E-FJF and FJF policies. According to the numeric results in Fig. 6, RT_2 with the E-FJF

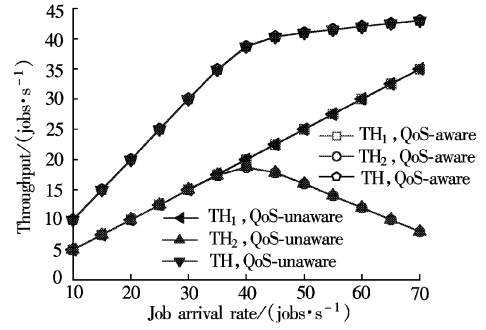


Fig. 4 Comparison of throughput between the QoS-aware and QoS-unaware job dispatching policy

policy is always more than that of the FJF policy, and the increment is 9.89% at most. Figs. 5 and 6 show that the response time of the high-priority jobs can be improved conspicuously with the QoS-aware job scheduling policy E-FJF, whereas that of the low-priority jobs is a bit impaired as the cost.

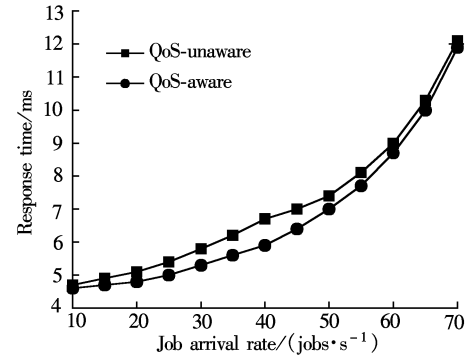


Fig. 5 Comparison of RT_1 between QoS-aware and QoS-unaware job dispatching policy

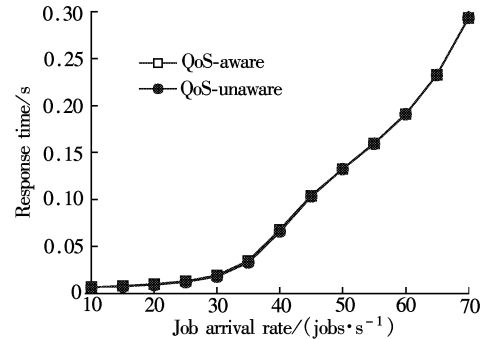


Fig. 6 Comparison of RT_2 between the QoS-aware and QoS-unaware job dispatching policy

Since Fig. 4 has shown that the throughputs are not affected much by the different job scheduling policies, according to Eq. (5), we can deduce that the mean length of the high-priority ready queue with the E-FJF policy must be shorter than that with the FJF policy. This deduction is confirmed by our numerical results of the mean queue length $D(q_{ij})$. Tab. 1 shows several numerical results of mean queue length and throughput. $D_i (i = 1, 2)$ denotes the mean length of the combined ready queue for class i jobs, i. e., $D_1 = D(q_{11}) +$

$D(q_{12})$ and $D_2 = D(q_{21}) + D(q_{22})$ in this example. As shown in Tab. 1, D_1 with the E-FJF policy is less than that with the FJF policy, while the case for D_2 is quite the reverse.

The above numerical results have shown that the QoS-aware job scheduling policy E-FJF outperforms the QoS-unaware job scheduling policy FJF in balancing the high-priority jobs, and thus provides better QoS to the high-priority jobs.

Tab. 1 Numerical results of mean queue length and throughput

Metrics		$\lambda (= \lambda_1 + \lambda_2)$	E-FJF	FJF
Mean queue length	D_1	30. 0	0. 792 0	0. 864 7
		35. 0	0. 975 8	1. 087 5
		40. 0	1. 188 0	1. 331 0
	D_2	30. 0	2. 970 3	2. 680 1
		35. 0	6. 043 5	5. 720 1
		40. 0	12. 638 3	12. 375 7
Throughput	TH_1	30. 0	150. 0	150. 0
		35. 0	175. 0	175. 0
		40. 0	200. 0	200. 0
	TH_2	30. 0	149. 974 2	149. 976 9
		35. 0	173. 905 8	173. 970 1
		40. 0	186. 987 3	187. 239 9

5 Conclusion

This paper focuses on the QoS-aware job scheduling for hierarchical decentralized grid architecture that consists of multilevel schedulers. We introduce mechanisms to correlate priorities of incoming jobs used for job selecting at the LS with the job dispatching policies at the GS for computational grids. An integrated QoS-aware job dispatching policy E-FJF is put forward. Performance modeling and analysis technique based on the SHLPN is proposed to investigate QoS-aware job scheduling on computational grids. Numerical results show that the QoS-aware job scheduling policy E-FJF outperforms the QoS-unaware job scheduling policy

FJF in balancing the high-priority jobs and thus enables priority-based QoS.

References

[1] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: enabling scalable virtual organizations [J]. *International J Supercomputer Applications*, 2001, **15**(3): 200 – 222.

[2] Li Keqin. Job scheduling for grid computing on metacomputers[C]//*Proc of the 19th IEEE International Parallel and Distributed Processing Symposium*. Denver, Colorado, USA, 2005: 180.

[3] He L, Jarvis S A, Spooner D P, et al. Dynamic scheduling of parallel jobs with QoS demands in multiclustes and grids [C]//*Proc of the Fifth IEEE/ACM International Workshop on grid Computing*. Pittsburgh, USA, 2004: 402 – 409.

[4] Fujimoto N, Hagihara K A. Comparison among grid scheduling algorithms for independent coarse-grained tasks[C]//*Proc of SAINT 2004 Workshop on High Performance Grid Computing and Networking*. IEEE Press, 2004: 674 – 680.

[5] Shan H, Olikar L, Biswas R. Job superscheduler architecture and performance in computational grid environments[C]//*Proc of ACM Conference on Supercomputing*. Washington, DC: IEEE Computer Society, 2003: 44 – 58.

[6] Li Keqin. Experimental performance evaluation of job scheduling and processor allocation algorithms for grid computing on metacomputers[C]//*Proc of the 18th International Parallel and Distributed Processing Symposium*. Santa Fe, New Mexico, USA, 2004: 170.

[7] Shan Zhiguang, Lin Chuang, Ren Fengyuan, et al. Modeling and performance analysis of a multiserver multiqueue system on the grid[C]//*Proc of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2003)*. IEEE Computer Society, 2003: 337 – 343.

[8] Lin C, Marinescu D C. Stochastic high-level Petri nets and applications[J]. *IEEE Transactions on Computers*, 1988, **37** (7): 815 – 825.

[9] Murata T. Petri nets: properties, analysis and applications [J]. *Proceedings of the IEEE*, 1989, **77**(4): 541 – 580.

[10] Ciaodo G, Muppala J, Trivedi K S. SPNP: stochastic Petri net package[C]//*Proceedings of the Petri Nets and Performance Models*. Kyoto, Japan, 1989: 142 – 151.

计算网格 QoS-aware 作业调度的建模与性能评价

单志广¹ 林 闯²

(¹ 国家信息中心信息化研究部, 北京 100045)
(² 清华大学计算机科学与技术系, 北京 100084)

摘要: 为了提高计算网格的服务质量(QoS), 研究了包含多层调度器的分级分布式网格体系结构中的 QoS-aware 作业调度问题, 提出了一种将计算网格本地调度器作业选择中所使用的作业优先级与全局调度器的作业分配策略相结合的 QoS-aware 作业调度综合控制策略. 建立了一个具有 2 层调度器的计算网格的随机高级 Petri 网 (SHLPN) 模型, 并且进行模型精化设计以降低模型求解的复杂性. 使用基于 SHLPN 的性能分析技术进行系统性能评价. 数值结果显示 QoS-aware 作业调度策略能够为高优先级的作业提供较 QoS-unaware 作业调度策略更好的 QoS 保证.

关键词: 计算网格; 作业调度; 服务质量; 性能评价; 建模; 随机高级 Petri 网
中图分类号: TP316