Future Generation Computer Systems 26 (2010) 198–206

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An integrated security-aware job scheduling strategy for large-scale computational grids

Chao-Chin Wu *, Ren-Yi Sun

Department of Computer Science and Information Engineering, National Changhua University of Education, Changhua City 500, Taiwan

ARTICLE INFO

Article history:
Received 5 December 2008
Received in revised form
1 August 2009
Accepted 3 August 2009
Available online 6 August 2009

Keywords:
Job scheduling
Computational grid
Fault tolerance
Genetic algorithm
Security

ABSTRACT

All existing fault-tolerance job scheduling algorithms for computational grids were proposed under the assumption that all sites apply the same fault-tolerance strategy. They all ignored that each grid site may have its own fault-tolerance strategy because each site is itself an autonomous domain. In fact, it is very common that there are multiple fault-tolerance strategies adopted at the same time in a large-scale computational grid. Various fault-tolerance strategies may have different hardware and software requirements. For instance, if a grid site employs the job checkpointing mechanism, each computation node must have the following ability. Periodically, the computational node transmits the transient state of the job execution to the server. If a job fails, it will migrate to another computational node and resume from the last stored checkpoint. Therefore, in this paper we propose a genetic algorithm for job scheduling to address the heterogeneity of fault-tolerance mechanisms problem in a computational grid. We assume that the system supports four kinds fault-tolerance mechanisms, including the job retry, the job migration without checkpointing, the job migration with checkpointing, and the job replication mechanisms. Because each fault-tolerance mechanism has different requirements for gene encoding, we also propose a new chromosome encoding approach to integrate the four kinds of mechanisms in a chromosome. The risk nature of the grid environment is also taken into account in the algorithm. The risk relationship between jobs and nodes are defined by the security demand and the trust level. Simulation results show that our algorithm has shorter makespan and more excellent efficiencies on improving the job failure rate than the Min–Min and sufferage algorithms.

1. Introduction

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [1–3]. It enables the dynamic sharing, selection, and aggregation of geographically distributed autonomous and heterogeneous resources at runtime depending on their availability, capability, performance, cost and the users' quality-of-service requirements [4–7]. However, a large-scale grid system is inherently unreliable by nature. The jobs are subject to system failures or delays caused by infected hardware, software vulnerability, network failure, overloaded resource conditions, non-availability of required software components, and distrusted security policy. To address these problems, a variety of fault-tolerance mechanisms and supports have been proposed for Grid systems. For instance, Globus [8] provides a heartbeat service to detect if any faults have occurred in running processes. When being notified of a failure, the application can take the appropriate recovery action. Moreover, the re-execution mechanism is adopted in Netsolve/Gridsolve [9] and Condor-G [10] systems while the replication mechanism in the Mentat system [11] is used for failure recovery.

Despite the fact that many heuristics have been suggested for large-scale job scheduling [12–18], as pointed out by Song et al. [19], they are not applicable in a risky environment. Therefore, Song et al. developed security assurance and risk-resilient strategies and proposed eight job scheduling algorithms for use under various risky conditions to address these problems. Their work [19,20] is built upon related works on Grid security, trust management and job scheduling. Their proposed security-assured job scheduling strategies consider the risk relationship between jobs and nodes using the *security demand* (SD) and the *trust level* (TL). The SD quantifies how much a user's job requires the assurance of secure computing services by a grid site or a cluster node. The TL quantifies how much a user can trust a site for successfully executing a given job. A job is expected to be successfully carried out when SD and TL satisfy a security assurance condition (SD $\leq$ TL) during the job mapping process. Furthermore, they also proposed a space-time genetic algorithm (STGA) based on the messy encoding concept with a space-time guided search mechanism.

* Corresponding author. Tel.: +886 4 7232105; fax: +886 4 7211284.
E-mail address: ccwu@cc.ncue.edu.tw (C.-C. Wu).

0167-739X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2009.08.004

Because its search is history-sensitive, STGA converges much faster than a traditional GA.

Although Song et al. proposed eight job scheduling algorithms for use under various risky conditions [19], all of these algorithms ignore the heterogeneity of fault-tolerance mechanisms supported in grid systems, where the scheduler assumes that all computational nodes adopt the same fault-tolerance strategy. In fact, different computational nodes are usually protected by different fault-tolerance mechanisms because distributed computational nodes are managed by different autonomous domains in a realistic large-scale computational grid. For instance, one grid site adopts the checkpointing mechanism to protect its computational nodes, one adopts the replication mechanism, and one adopts both the checkpointing and replication mechanisms. Furthermore, various fault-tolerance mechanisms may require different hardware and software supports. For instance, the job migration operation with the checkpointing mechanism must periodically transmit the transient process state to the server. When a failure occurs, the server will transmit the last stored process state to the newly assigned computational node for resuming the execution of the failed job. In such a grid environment, all the job scheduling algorithms proposed by Song et al. are not applicable. Therefore, in this paper we propose a security-assured grid job scheduling strategy considering the heterogeneity of fault-tolerance mechanism support.

We propose a genetic algorithm (GA) for job scheduling. In this algorithm, we consider four kinds of fault-tolerance mechanisms, including the job retry, the job migration without checkpointing, the job migration with checkpointing and the job replication mechanisms. Because each fault-tolerance mechanism has different requirements for gene encoding, we propose a new chromosome encoding approach to integrate the four kinds of mechanisms in a chromosome. Simulation results show that our algorithm has shorter makespan and more excellent efficiencies on improving the job failure rate than the Min–Min algorithm.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the system model and the proposed job scheduling algorithm considering the heterogeneity of fault-tolerance strategies. Section 4 demonstrates the experimental results. Finally, Section 5 concludes the paper.

2. Related Work

A variety of job scheduling strategies have been proposed for computational grids. Xiao et al. proposed an incentive-based scheduling scheme that investigates how to maximize the success rate of job execution and minimize the fairness deviation among resources by allowing resource providers and resource consumers to make autonomous scheduling decisions [21]. Doulamis et al. proposed a fair scheduling algorithm that will fairly reduce the CPU rates assigned to the tasks when the resources are insufficient so that the share of resources that each user gets is proportional to the user's weight [22]. Viswanathan et al. proposed a resource-aware dynamic incremental scheduling that handles large volumes of computationally intensive, arbitrarily divisible loads submitted for processing at cluster or grid systems [23]. Lee and Zomaya proposed two algorithms for bag-of-tasks applications, one for data-intensive tasks and one for computation-intensive tasks, which adopt task duplication for scheduling without requiring accurate performance prediction information [24]. Baghban et al. proposed a job scheduling algorithm that will selects the resources based on input jobs, communication links and resource computational capability [25]. Bertin et al. proposed a fully decentralized algorithm that can achieves both optimal path selection and flow control only requiring local information at each slave computing task and at each buffer in the network links [26]. Yu and Chen proposed a genetic algorithm that takes different computing capabilities of computing nodes and dynamic network status into consideration [27]. Nobrega et al. proposed scheduling heuristics that investigate the tradeoff between two factors: the challenging requirement of complete and accurate information about the applications and the grid environment, and the extra consumption of resources incurred by task replication [28]. Li et al. proposed a predictable and grouped genetic algorithm where a job workload estimation algorithm is designed to evaluate a job workload based on its historical execution records and the divisible load theory is employed to predict an optimal fitness value by which the convergence process can be shortened in searching for a large scheduling space [29]. Chang et al. proposed an ant-based algorithm that aims at balancing the entire system load while trying to minimize the makespan of a given set of jobs [30]. In addition, Chang et al. also proposed a job scheduling algorithm that dispatches a job to the site where the needed data are present to reduce data access time and the amount of inter-cluster-communications [31]. Gao et al. developed two algorithms that use their proposed models for predicting completion time to schedule jobs at both system level and application level [32]. Kosar and Balman introduced an interesting data-aware batch scheduler for data-intensive computing [33]. Their scheduler is called Stork that is especially designed to understand the semantics and characteristics of data placement tasks, which can include data transfer, storage allocation and de-allocation, data removal, metadata registration and un-registration, and replica location. Palmieri discussed the job scheduling policy when network resources in a Grid system can be reserved [34]. He argued that reservation of connectivity resources is needed for data-intensive applications to facilitate the transportation of enormous data-sets between Grid nodes in predictable times. The network can be the point-to-point dedicated Wavelength Division Multiplexing (WDM)-based optical transport infrastructure.

Some scheduling strategies especially emphasized the importance of security awareness. Azzedin and Maheswaran [35] proposed a trust model that incorporates the security implications into scheduling algorithms. Humphrey and Thompson [36] proposed usage models for security-aware Grid computing but they did not elaborate on how to design a scheduler by incorporating the security concerns into collaborative computing over distributed cluster environment. Abawajy [37] and Litke [38] suggested replicating jobs at multiple sites to guarantee successful job executions in a Grid environment.

Song et al. [20,19] thought that a job distributed to a remote node may suffer from some infections or malicious attacks. Therefore, a job scheduler must consider the risk when dispatching jobs to remote nodes [20,19]. They proposed a job failure model to represent the risk level in job scheduling. There were two parameters in this model: the security demand (SD) and trust level (TL). SD represents a job's security requirement level, higher SD value represents this job has higher security requirement, so the job needs a more reliable node for job execution. TL represents a node's secure level, higher TL value represents this node can provide a more reliable environment. So different jobs assigned to different nodes would have different risk levels. Based on the job failure model, they proposed three schedule strategies based on different risk levels: (1) Secure mode—jobs were only scheduled to those nodes which can ensure security. (2) Risky mode—jobs were scheduled to any available nodes without considering the risks between jobs and nodes, so it took all possible risks. (3) F-risky mode—jobs were scheduled to available nodes to take at most f risk, where f was a probability.

Moreover, in [19], Song et al. proposed four types of scheduling strategies: (1) Risky mode—jobs were scheduled to any available nodes without considering risks between jobs and nodes, so it took all possible risks. (2) Preemptive mode—failed jobs would be