A Different Strategies of All Efficient Grid-Scheduling Techniques.

Mr. Pottigar Vinayak V.
CSE DEPT,NBNSCOE.
Solapur,MH,India

Dr. Prof. Thool R. C.
Head of IT dept.,SGGS
Nanded,Maharashtra,India

Abstract Computational grids have become an appealing research area as they solve compute-intensive problems within the scientific community and in industry. A grid computational power is aggregated from a huge set of distributed heterogeneous workers; hence, it is becoming a mainstream technology for large-scale distributed resource sharing and system integration. Unfortunately, current grid schedulers suffer from the haste problem, which is the schedule inability to successfully allocate all input tasks. Accordingly, some tasks fail to complete execution as they are allocated to unsuitable workers. Others may not start execution as suitable workers are previously allocated to other peers. This paper is the attempt to survey the scheduling haste problem, and their available solutions. It also presents a reliable grid scheduler. The proposed scheduler selects the most suitable worker to execute an input grid task using a fuzzy inference system. Hence, it minimizes the turnaround time for a set of grid tasks. Moreover, our scheduler is a system-oriented one as it avoids the scheduling haste problem. Experimental results have shown that the proposed scheduler outperforms traditional grid schedulers as it introduces a better scheduling efficiency.

Keywords grids Scheduling, grid computing,

I Introduction

In recent years, due to the dramatic development of net-work technologies and the popularity of the Internet, grid computing has become an appealing research area (Jens et al. 2009; Lee et al. 2009). It is becoming a mainstream technology for large-scale distributed resource sharing and system integration. Moreover, grid technology has emerged as an important tool for solving compute-intensive problems within the scientific community and in industry as they have led to the possibility of using distributed computers as a single, unified computing resource.

Computational grids are the next generation of computer clusters. They aim to maximize the utilization of resources owned by a set of distributed heterogeneous systems (He et al. 2005; Sacerdoti et al. 2003).

Moreover, grids can be considered as the recent instances of meta computing (Wolski et al. 1999). The primary goal of grid computing is to provide a transparent access to geographically distributed heterogeneous resources owned by different individuals or organizations (Jen and Yuan 2009). Hence, the grid provides hardware and software infrastructures to create an illusion of a virtual supercomputer that exploit the computational power aggregated from a huge set of distributed workers (Buyya et al. 1999). This allows the execution of tasks whose computational requirements exceed the available local resources. However, although the notion of grid computing is simple and attractive, its practical realization poses several challenges and open problems that need to be addressed (Min-Jen and Yin-Kai 2009; Michael and William 2008). These challenges include resource discovery, failure management, fault tolerance, resource heterogeneity, reliability, scalability, security, and more importantly the scheduling of incoming tasks among available grid resources (Tseng et al. 2009).

Scheduling is the major puzzle in developing a grid-based computing paradigm (Tseng et al. 2009; Iavarasan et al. 2005). It involves the matching of task or application requirements with the available resources (Tseng et al. 2009). Scheduling in grids can be carried out in three different phases which are: (1) resource discovery, (2) scheduling, and (3) executing (Li and Hadjinicolaou 2008). However, to achieve the expected potentials of the avail-able resources, efficient scheduling algorithms are required (Daoud and Kharma 2008). Unluckily, scheduling algorithms previously employed in computer clusters cannot be used in grids as they run on homogenous and guaranteed resources over the same LAN. A scheduler used in a computer cluster only manages such cluster; hence, it owns the resources with no need to discover new ones (Sacerdoti et al. 2003). Also it assumes both the availability and stability of resources. On the other hand, scheduling in grids is significantly complicated as a result of grid heterogeneity and dynamic nature (Kiran et al. 2009)

Unlike the cluster scheduler, a grid scheduler should have the ability to discover new computing resources over multiple administrative domains (Yan et al. 2005). The dynamic nature of grids is a result of both the network connectivity and grid resources. The network may be unreliable as it cannot guarantee its bandwidth. Moreover, grid resources change their availability and capability over time as they may join or leave the grid without any notification (Shah et al. 2007; Kousalya and Balasubramanie 2008).

Two alternative views may be considered when developing a grid-scheduling system. The first is the user view (UV) while the other is the system view (SV). On one hand, the user aims to achieve the maximum quality of service (QoS); hence, he asks the scheduler to elect the best currently available resources for executing his task. On the other hand, the grid system tries to manage the available resources in a way for achieving the maximum QoS for all users not for a specific one. Based on these alternative views, grid schedulers can be categorized into two major categories, which are: (1) Task-Oriented Grid Schedulers (TOGS), and (2) System-Oriented Grid Schedulers (SOGS). The former supports the user's demands, and therefore, it tries to minimize the execution time of each input task. The latter category supports the system's demands, and therefore, it tries to introduce a high-throughput computing. Unlike TOGS, SOGS aims to maximize the processing ability of the system over the long run. Also, it introduces a better resource management scheme by allocating the grid task to the most suitable resource, not the best available one.

To the best of our knowledge, most of the proposed grid schedulers were task-oriented ones (Aggarwal et al. 2005), which usually suffer from the haste problem. The haste problem is the ability of the scheduler to present a good scheduling performance in the present; however, an overall degraded performance is presented in the long run. More-over, implementing an efficient SOGS has not been addressed yet. Hence, scheduling in grids is still more complex than the proposed solutions. Many hurdles stand in the way of achieving the maximum utilization of grid resources (Liu et al. 2006). Accordingly, scheduling in grids is still an elusive problem that attracts the interests of many researchers (Aggarwal and Kent 2005; Hsin 2005).

This paper discusses the scheduling haste problem in details. Then, a novel system-oriented grid scheduler is introduced as a solution for this problem, which is the first to study the scheduling by mapping the tasks to the suitable workers not the best available ones. This has a great impact in minimizing the turnaround time of executing a set of tasks in the long run. The proposed scheduler maps the suitable task to the available worker in two steps. During the first, a candidate set of the ready-to-run tasks is formulated. Those candidate tasks are the ones whose resource requirements not only be satisfied by the available worker, but also do not waste the worker resources. Such aim was carried out be representing all the ready-to-run tasks as well as the available worker in a proposed n-dimensional parameter space (Parm-Space), then, the k-nearest neighbors technique is used to elect the candidate set of tasks, which are the k-nearest tasks to the available worker in the Parm-Space. During the next step, a novel fuzzy-based matchmaking procedure is applied to choose the most suitable task to be executed on the available worker. Experimental results have shown that the proposed scheduler outperforms traditional ones as it introduces a better scheduling efficiency as well as avoiding the scheduling haste problem.

II Related work

Tremendous amount of research had been introduced in the area of task scheduling.

Chandak et al. (2011) surveys heuristic-based task allocation strategies and their efficiency. This strategy optimizes various performance parameters such as makespan, resource utilization, response time, workload balancing, service reliability, fairness deviation and throughput. A task life cycle model has been suggested in computational grid. We have also proposed a classification of heuristic task allocation strategies for computational grid. In Salehi et al. (2008), based on swarm intelligence, an echo system of adaptive fuzzy artificial ants had been presented for grid load balancing. The ants in this environment can create new ones and may also commit suicide depending on the existing conditions. A new concept called ant level load balancing is presented here for improving the performance of the mechanism. Another Hybrid Ant Colony Optimiza-tion (HACO) scheduling algorithm is proposed in Nithya and Shanmugam (2011). HACO is based on Ant Colony Optimization (ACO) algorithm which uses batch mode heuristic mapping. In this ant colony algorithm each job is considered as an ant and optimal solution is provided with the help of pheromone detail. Heuristic-based ant colony algorithm is used in the second phase of the scheduler. In Saravanakumar and Prathima (2010), a load-balancing algorithm adapted to the heterogeneous grid computing environment has been presented, which is an adaptive decentralized sender-initiated load-balancing algorithm. In Tchernykh et al. (2005), a two-level hierarchy scheduling has been introduced. At the first level, broker allocates computational jobs to parallel workers. At the second level each worker generates schedules of the parallel jobs assigned to it by its own local scheduler. Selection, allocation strategies, and efficiency of proposed hierarchical scheduling algorithms are also discussed in Heymann et al. (2000), a simple but effective scheduling strategy that dynamically measures the execution times of tasks and uses this information to dynamically adjust the number of workers to achieve a desirable efficiency, minimizing the impact in loss of speedup. The scheduling strategy has been implemented using an extended version of MW, a runtime library that allows quick and easy development of master– worker computations on a computational grid.

As per our knowledge, no grid-scheduling strategy till date taking the scheduler haste problem into account. All schedulers tend to choose the best available resource for executing an input task not the suitable one. However, choosing the suitable worker for the input task guarantees the system reliability as it maximizes the capability to execute the incoming future tasks.

III Background and basic concepts

Grid computing provides a high performance computing platform to solve larger scale applications by coordinating and sharing computational power, data storage and network resources across dynamic and geographically dispersed organizations. Scheduling onto the Grid is NP-complete, so there is no best scheduling algorithm for all grid computing systems. An alternative is to select an appropriate scheduling algorithm to use in a given grid environment because of the characteristics of the tasks, machines and network connectivity. Job scheduling is one of the key research area in grid computing. The goal of scheduling is to achieve highest possible system throughput and to match the application need with the available computing resources. Motivation of this study is to encourage and help the amateur researcher in the field of grid computing, so that they can understand easily the concept of scheduling and can contribute in developing more efficient and practical scheduling algorithm. This will benefit interested researchers to carry out further work in this thrust area of research.

A *Cluster computing*

Computer clusters are three-tier structural systems interconnected via two functional relationships (He et al. 2005). Those tiers are: (1) clients, (2) cluster server (master
node or scheduler), and (3) cluster workers. The server is a dedicated machine that receives incoming tasks from the clients; this is a client–server relationship. It is the server's responsibility to schedule the incoming tasks among a set of identical, dedicated, and usually fixed workers for execution; this is a master–slave relationship. In clusters, the underlying interconnection network is usually a private and
high-speed network, which in turn guarantees the high quality of communication service (Sacerdoti et al. 2003). Certainly, this infrastructure is different from computational grids which are heterogeneous and dynamic in both resource and communication availabilities.

As depicted in Fig. 1, scheduling in clusters is quite simple. Initially, incoming tasks as well as their requirements are kept in an input queue. The task requirements include both the expected execution time and the number of workers needed to execute the task. The cluster server (scheduler), on the other hand, continuously reports the availability of cluster workers. Once a worker completes executing a task, it notifies the server that it is available and ready to receive a new one. Many scheduling algorithms can be used to pick a task for execution when a worker is available such as: First Come First Serviced (FCFS), Minimum Request Job First (MRJF), and Shortest Job First (SJF) (Buyya et al. 1999).

B *Task-oriented versus system-oriented grid schedulers*

Recent work in grid scheduling aims to introduce task oriented schedulers. A TOGS manages the scheduling for the benefit of the end user, who has initialized the task, with no awareness of the overall system efficiency. From one point of view, TOGS considers only the current state of the grid system. It gives no awareness to the future needs of the other grid users. This behavior degrades the system performance as the newly incoming tasks may be blocked (starved) for a long period of times (as their execution requirements are not currently available). From another point of view, TOGS is a greedy scheduler as it exploits all the currently available grid resources for the current task to be scheduled. It gives no attention to the needs of the future incoming tasks.

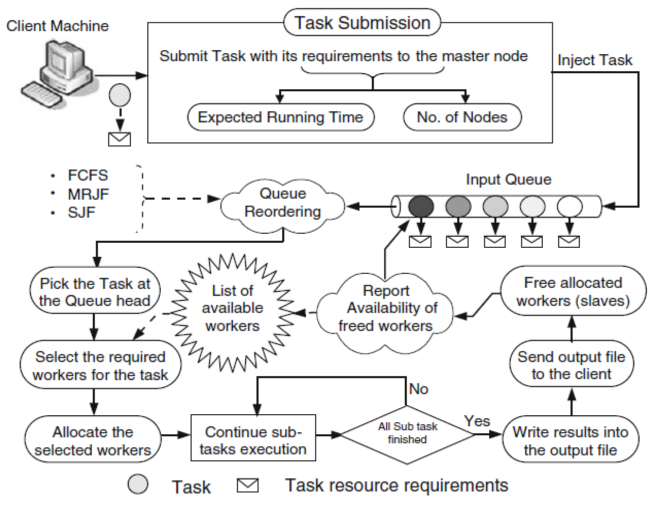
Client Machine
Task Submission
Submit Task with its requirements to the master node
Expected Running Time
No. of Nodes
Inject Task
FCFS
MRJF
SJF
Queue Reordering
Input Queue
Pick the Task at the Queue head
List of available workers
Report Availability of freed workers
Free allocated workers (slaves)
Select the required workers for the task
Send output file to the client
No
Allocate the selected workers
Continue sub-tasks execution
All Sub task finished
Yes
Write results into the output file
Task
Task resource requirements

Fig 1 clustering system infrastructure

IV The proposed Work

Much of the work on the scheduling strategies are carried out. Some of the work which are best suited to be implemented are given with its working strategies.

A *Strategies based on fuzzy matchmaking approach*

The following strategy is something which uses fuzzy matchmaking approach to use the task scheduling in grid computing.

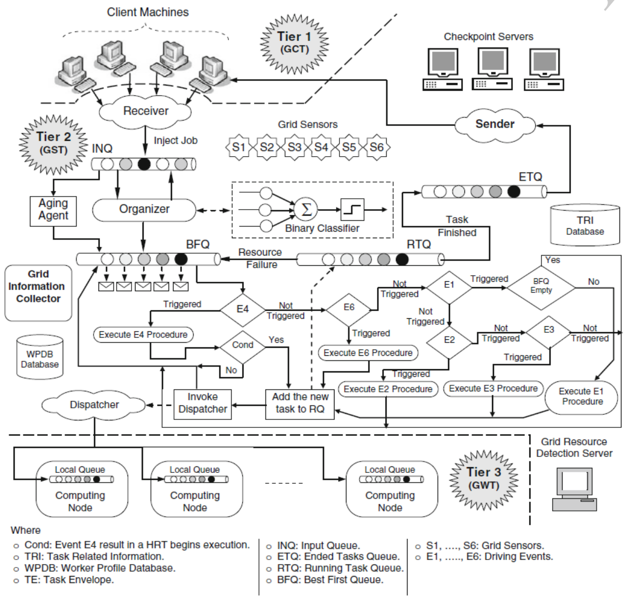

Fig.2. The proposed 3 tier scheduling system structure

As depicted in Fig. 2, the proposed grid-scheduling system consists of three different tiers, which are: (1) grid clients tier (GCT), (2) grid scheduler tier (GST), and (3) grid workers tier (GWT). GCT represents the grid users who are sitting behind a set of client machines and willing to execute their tasks using the grid computational power. On the other hand, GWT provides the infrastructure that creates an illusion of a virtual supercomputer. It exploits the computational power aggregated from a large set of geographically distributed workers. The last tier is the GST, which represents the core of the proposed scheduling framework. GST is responsible of carrying out the scheduling decisions. It determines where and when to execute an input task T by applying a set of heuristic rules. GST also appoints when T should be interrupted and resume its execution. Moreover, GST is responsible of collecting the results of T, and then sending them to the task initiator. Executing a task T in the proposed framework is carried out in the sequential steps shown in Fig. 3. As depicted in the figure, the grid scheduler stands as an interface between the two other tiers. The following subsections describe the internal structure of GST as well as the functionality of its modules in more detail

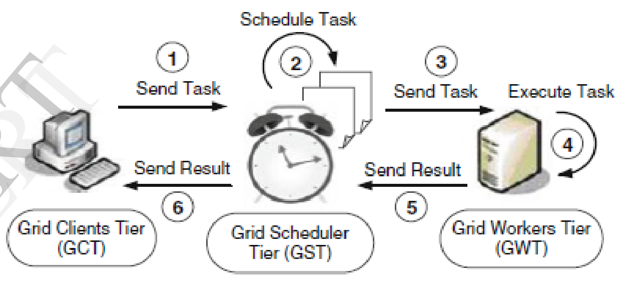

Fig 3 steps to execute a task in a grid

As depicted in the figure, the grid scheduler stands as an interface between the two other tiers. The following subsections describe the internal structure of GST as well as the functionality of its modules in more detail

B *A job grouping based approach for task scheduling:*

Grid computing provides a high performance computing platform to solve applications with large number of independent jobs. However, user jobs developed for grid might be small and of varying lengths according to their computational needs and other requirements. In fact, it is a big challenge to design an efficient scheduler, but there exists some grouping based job scheduling strategy that intends to minimize total processing time by reducing overhead time and computation time, and on the other hand maximizing resource utilization than without grouping based scheduling. Further analysis and research on job scheduling can be carried out to enhance the performance of grouping based scheduling algorithm in grid computing. This study intends to achieve better performance by extending the concept of grouping based job scheduling. Therefore, this paper proposes &x201C;A modified grouping-based job scheduling in computational grid&x201D; with the objective of minimizing overhead time and computation time, thus reducing overall processing time of jobs. The work is verified through simulation and the results obtained shows that the proposed grouping-based scheduling algorithm is better than, others.

C *Resource management for task allocation with Virtual Organization concept.*

The Optimized Resource Management Structure with VO

A. *The resource management frame*

To manage grid resource more efficient, the resource organization and manage constructor must correctly materialize the resource constitutes in grid computing environment. As a result, the VO constructor should display the following mode characters: Wide distributed, part centralized and local application controlled. In order to demonstrate the thought of grid computing, i.e. service oriented, VO must confirm the real times waken rule as "task treatment and job requirement oriented" in allusion to the resource management form and the resource waken course. It should dynamically change its own resource quantity with the transformation of task scale and should also modify the way to dispose information in order to optimize the managing method and the system efficiency[1].

In the process of resource matching with VO, the following assumption always exists: Huge scale task group activate in actual grid computing environment, the VO group that takes charge of scheduling task group and matching resource has already engrossed plentiful resources that belong to the specified area. If some task entity refers a huge scale task requirement T1 this time, the produced virtual organization VO1 must occupy propriety resources to match the requirement. But the idle resources of this area can not satisfy the actual needs, VO1 has to acquire the great mass of resources from Wide Area Grid Computing environment (WAGC). Thus it leads to serious net delay and load, affects the performance of task management heavily, and also results in resource blind search and the phony failure phenomena of the task schedule process. As a result, besides the characters of interactivity, dynamic adaptation and periodicity, the VO should also create proper quantity Sub Virtual Organization (SVO) to assort with the integral management action according to the task character and the resource distribution. Suppose there are many tasks be disposed in the grid computing environment, the three arbitrary task entities put in three sweeping task disposal requirements T1, T2 and T3. System responds the requirement and creates three corresponding virtual organization called VO1, VO2 and VO3. They take charge of resource searching, resource locating, resource possessing and resource matching. Because of the sweeping task group existing in grid environment, great deals of resources are possessed by task group[2]. The three virtual organizations have to gain enough resources from distributed computing resource in WAGC. This causes the three VOs resource distribution very decentralized. So the VO1, VO2 and VO3 must create SVO to manage dispersive resources, as shown in Fig.1. In the map, the VOi -j (i =1,2,3; j is integer) is the SVO of main virtual organization VOi.

B The resource organization strategy of simple task

As we know, the scale of virtual organization that derives from specific task requirement is different with the task scale. Generally speaking, when the task scale is huge, the resource distribution presents wide area, aggregative and clotty characters. So we introduce classified resource management method to optimize the resource management and minimize the task dispose cost. Each resource part via division is managed by the corresponding SVO. This shows that the resource organization based on virtual organization of simple task takes on multilevel allots and concentrative management mode, see

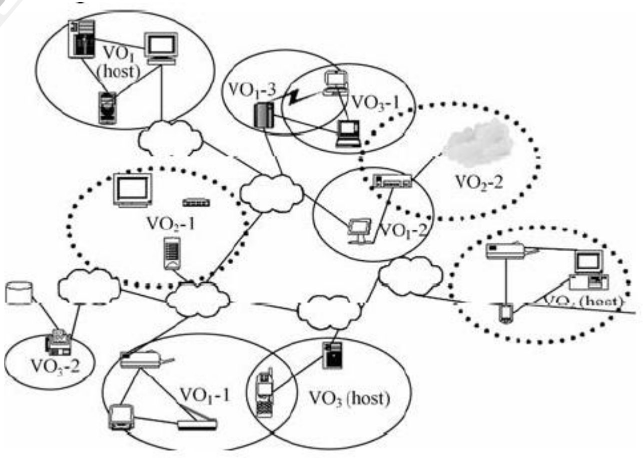

Fig 4.Virtual organization of a nodes in Grid

References

1.Aggarwal M, Kent R (2005) An adaptive generalized scheduler for grid applications. In: The 19th annual international symposium on high performance computing systems and applications (HPCS'05), 2005, pp 15–18

2.Aggarwal M, Kent R, Ngom A (2005) Genetic algorithm based scheduler for computational grids. Int Symp High Per form Comput Syst Appl 15(18):209–215

3.Berman F, Wolski R, Figueira S, Schopf J, Shao G (1996) Application-level scheduling on distributed heterogeneous networks.In: Proceedings of the 1996 ACM/IEEE conference on Supercomputing, 1996, p 39

4.Boutammine S, Millot D, Parrot C (2006) An adaptive Scheduling Method for Grid Computing. Euro-Par 2006:188–197

5.Buyya R (1999) High performance cluster computing: systems and architectures. Prentice Hall, USA
Buyya R, Vazhkudai S (2001) Compute power market: towards a market-oriented grid. In: The 1st international symposium on cluster computing and the grid, 2001, p 574

6.Casanova H, Kim M, Plank J, Dongarra J (1999) Adaptive scheduling for task farming with grid middleware. Int J Supercomp ut Appl High-Perform Comput 13(3):231–240

7. An efficient grid-scheduling strategy based on a fuzzy matchmaking approach Ahmed I. Saleh Published online: 14 September 2012