Global Journal of Technology & Optimization

Balasangameshwara, Global J Technol Optim 2014, 6:1
DOI: 10.4172/2229-8711.1000169

Research Article Open Access

Survay on Job Scheduling, Load Balancing and Fault Tolerance Techniques for Computational Grids

Jasma Balasangameshwara*

Associate Professor, Department of Computer Science and Engineering, Atria Institute of Technology, Bangalore, Karnataka, India

Abstract

Computational grid is a network of loosely coupled, heterogeneous and geographically-dispersed computers acting together to perform a large compute-intensive job. In this article, we focus on the existing approaches to grid scheduling, load balancing and fault-tolerance problems. Although grid scheduling, load balancing and fault tolerance are active research areas in grid computing, these areas have largely been and continue to be developed independent of one another each focusing on different aspects of computing. Hence, in this survey, we hope to show that robust applications that can provide efficient results can be designed by collectively considering these areas. To this end, we first provide an introduction to the motivation, grid scheduling, load balancing and fault tolerance concepts of grid computing and discuss the works that have provided significant contributions to each of these areas since its inception until 2013. We discuss their advantages, disadvantages and analyze their suitability for usage in a dynamic grid environment. We conclude that, while important advancements have been made in each of these areas individually, high performance approaches that cumulatively consider these areas still remain to be explored. We also discuss the research work that is missing and what we believe the community should be considering. To the best of our knowledge, no such survey has been conducted in the literature up to now.

Keywords: Computing grid; Fault-tolerant scheduling; Load balancing

Introduction

Due to the evolution of science and engineering, problems in these fields have become complicated. To resolve such problems, a prime computing facility is a prerequisite.

In the mid-1990s, the term grid was coined to describe technologies that would allow consumers to obtain computing power on demand. A computing grid is an amalgamation of hardware and software infrastructures from different locations that offer dependable, steady and cost-effective access to high-end computational capabilities [1]. They facilitate dynamic sharing, aggregation and selection of geographically distributed, independent computers at run-time based on their accessibility, performance and capability.

Distributed computing is a network of computers, each accomplishing a portion of an overall job to achieve a computational result much quicker than a single computer.

Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can be divided into sub-problems and solved concurrently. Grid computing can either be called as is distributed form of parallel computing or parallel form of distributed computing.

As shown in Figure 1 [1], at the fabric layer, grids provide access to different resource types such as compute, storage and network resources and code repository. The connectivity layer defines core communication and authentication protocols for easy and secure network transactions. Resource layer defines protocols for the publication, discovery, negotiation, monitoring, accounting and payment of sharing operations on individual resources. The collective layer captures interactions across collection of resources and directory services that allows monitoring and discovery of virtual organization resources. The application layer comprises whatever user application built on top of the above protocols and application programming interfaces and operate in the virtual organization environments.

In a grid environment, the job arrival patterns are unpredictable and the computing capabilities unbalanced and unreliable. The computers in a specific grid site may become overloaded while that in other grid sites may be under-loaded [2]. Thus, the heterogeneous and the dynamic nature of the grid requires job scheduling, load balancing and fault-tolerance in order to make the best usage of the performance of the grid computers.

Background

Job scheduling: Job Scheduling is defined as the process of making

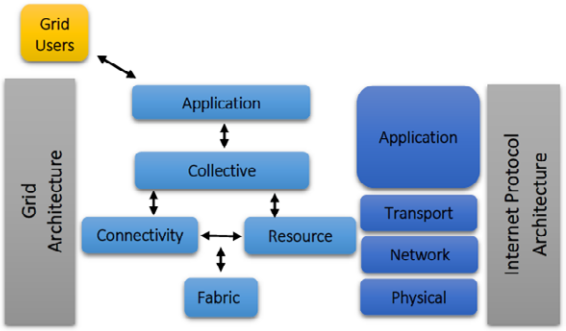

Figure 1: Grid Computing Reference Architecture.

***Corresponding author:** Jasma Balasangameshwara, Associate Professor, Department of Computer Science and Engineering, Atria Institute of Technology, Bangalore, Karnataka, India, Tel: +91 9886 019 686; E-mail: jasma2002@gmail.com

Received November 18, 2014; **Accepted** December 15, 2014; **Published** December 25, 2014

Citation: Balasangameshwara J (2014) Survay on Job Scheduling, Load Balancing and Fault Tolerance Techniques for Computational Grids. Global J Technol Optim 6: 169. doi: 10.4172/2229-8711.1000169

scheduling decisions involving different types of resources over multiple administrative domains.

A job is defined as a program that requires a resource. The term resource is an hardware or software component that can be scheduled [3]. Scheduling is computationally hard; it has been shown that the problem of finding the optimum scheduling in heterogeneous systems is NP-hard [3].

Developments in scheduling research reflect movements from isolated multi-host scenarios to large scale infrastructures. These approaches have brought about a change from local to global scheduling [1]. Figure 2 highlights the grid computing system model.

Grid scheduling characteristics

- **Dynamic structure of computational grid:** The resources in a grid can join or leave the Grid in an unpredictable way.
- **Resource heterogeneity:** Computing resources very disparate in their computing power, ranging from laptops, desktops, clusters, supercomputers and even small computational devices.
- **Job heterogeneity:** Jobs arriving at a grid are diverse and heterogeneous in terms of their memory, network and computational needs.
- **Network heterogeneity:** Grid resources are connected through the Internet using different interconnection networks.
- **Local schedulers:** Grids are constructed by the contribution of the computational resources by institutions, universities, enterprises and individuals.
- **Local resource policies:** It is because of the different ownership of the resources that one cannot assume full control over the grid resources.
- **Job-resource requirements:** Grid schedulers cannot assume full availability and compatibility of resources while scheduling jobs.
- **Large grids:** Efficient resource management and the use of different types of schedulers are required to achieve the scalability for large-scale grids and a large number of jobs.
- **Security:** A job has security requirements and on the other hand, the resources have their own security requirements.

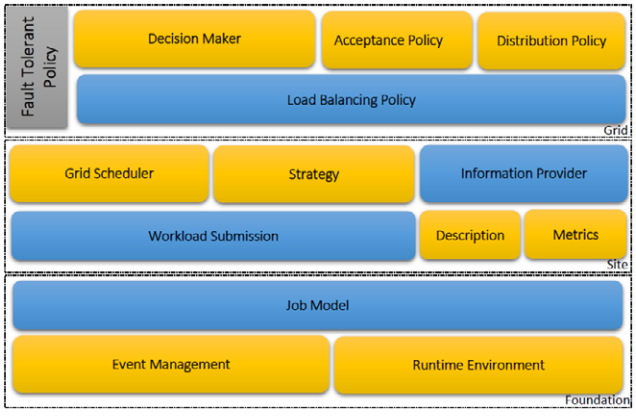

Figure 2: Grid Computing System Model.

Reference	Job Scheduling	Load Balancing	Fault Tolerance
[2]	Yes	No	No
[3]	Yes	No	No
[6]	Yes	No	No
[7]	Yes	Yes	No
[8]	Yes	Yes	No
[9]	Yes	Yes	No
[12]	Yes	No	Yes
[14]	Yes	No	Yes
[15]	Yes	No	Yes

Table 1: Fault Tolerant Scheduling and Load Balancing Summary and Comparison.

Load balancing

Grids functionally unite worldwide scattered computers for creating a universal source of computing power and information. A key trait of grids is that the resources are shared among numerous applications and the quantity of resources available to any given application highly fluctuates over time [4].

Therefore, load balancing is a technique to augment resources, utilizing concurrency, exploiting throughput improvisation and cutting the response time through an appropriate distribution of application. Using multiple components with load balancing instead of a single component may increase the reliability through redundancy [5].

The importance of the load balancing is as follows:

- An unforeseen peak load can be routed to the relatively unoccupied grid system.
- If the grid is fully utilized, then the lowest priority job currently being performed can be either temporarily suspended or cancelled and performed later so as to make room for the higher priority jobs.

Often, grid computing is regarded as a successor of the distributed and parallel computing. Nevertheless the grid computing and distributed computing environment (DCE) are elementarily different. A DCE is predictable: The availability of resources is based on the fact that the reservation and processing speeds are static and known in advance.

A grid environment, nonetheless, is greatly unpredictable as the resources have dissimilar and unknown processing speeds and they can be added or removed at any time. As a result, the dynamic nature of the grid makes load balancing a challenge (Table 1).

Load balancing bharacteristics

- **Optimum resource utilization:** A load balancing algorithm should optimize the utilization of the resources by optimizing the time or cost related to these resources.
- **Fairness:** A load balancing algorithm is said to be fair, meaning that the difference between the heaviest loaded computing resource and the lightest loaded computing resource in the network is minimized.
- **Elasticity:** As the topology of the network or grid goes on changing, the algorithm should be flexible enough to adhere to the changes.

Load balancing policies: The three policies that govern the action of a load balancing algorithm when a load imbalance is detected deal with information, transfer, location and

selection. The information policy is responsible for keeping up-to-date load information about each resource in the system [9].

The transfer policy deals with the dynamic aspects of a system. It uses the resources' load information to decide when a resource becomes eligible to act as a sender.

Location policy selects the partner resources for a job transfer transaction. If the resource is an eligible sender, the location policy seeks out a receiver resource to receive the jobs. If the resource is an eligible receiver, the location policy looks for an eligible sender resource. Once a resource becomes an eligible sender or receiver, a selection policy is used to pick up the queued jobs to be transferred.

A stable symmetrically initiated adaptive algorithm uses the information gathered during polling to classify the resources as overloaded, under-loaded or OK (resources having manageable load). These actions impose a small and constant overhead, irrespective of the number of resources in the system [10-12].

Selection policy uses several criteria to evaluate the queued jobs. Its goal is to select a job that reduces the local load, incurs as little cost as possible in the transfer and has good affinity to the resource to which it is transferred.

A load balancing algorithm in which a computing resource exchanges information and transfers jobs to its physical and/or logical neighbours is called "neighbour-based" load balancing method. The load balancing algorithms in which the computing nodes are partitioned into clusters based on the network transfer delay are called "cluster-based" load balancing methods.

Fault tolerance

Fault tolerance is a vital property for large-scale computing grids. To achieve high level of dependability and availability, the grid infrastructure should be fault-tolerant. As the failure of resources affect the job execution fatally, fault tolerant service is crucial to fulfill the QoS requirements. Also transient faults are tough to identify as they do not result in a failure rather a job state modification leading to incorrect output. The resource failures can either be processor or link failures.

- **Active replication**

This technique is based on the space redundancy i.e. multiple copies of each job are mapped on different processors and run in parallel to tolerate a fixed number of failures. With such a technique, no fault detection mechanism is required [13].

- **Passive replication**

The main idea of this technique is that a backup copy of a job is activated only if a fault occurs while executing its primary copy. Passive replication scheme does not require fault diagnosis and it guarantees to recover all failed jobs. In such a scheme, only two copies of the job are scheduled on different processors [14].

Two techniques can be applied while scheduling primary and backup copies of each job [15].

(1) Backup overloading consists of scheduling backups for multiple primary jobs during the same time slot in order to efficiently utilize the available processor time; and

(2) De-allocation of the resources reserved for the backup jobs when the corresponding primaries are completed successfully.

Related Work

We now describe early work in grid computing related to scheduling, load balancing and fault tolerance that we believe have contributed significantly to these areas.

Grid scheduling techniques

Computational grids offer the potential for applications to aggregate massive bandwidth, computational power, memory and other resources during a single execution.

Katia Leal et al. [2] presents a decentralized model for scheduling independent jobs in Federated Grids.

Federated grids are characterized by allowing resource sharing among several grid infrastructures of different types. Thus, it supports the interconnection of Enterprise grid, Partner and Utility grids to increase the total number of participating grids [2]. Katia Leal's model consists of meta-schedulers, where each meta-scheduler implements a mapping strategy in order to improve the job make-span and response time. Their approach acts as an alternative to centralized, application-centric and ad-hoc solutions. Katia Leal strategy is independent of the following:

- Processor speed
- Job length
- Specialized sensors

The APPLeS (Application Level Scheduling) [3] technique provides a procedure, application software and software environments for adaptively scheduling and deploying applications in heterogeneous multi-user grid environments. APPLeS integrate the static and dynamic resource information, performance prediction, application-specific information and scheduling techniques that adjust to the application execution on the fly. All APPLeS enabled applications are scheduled adaptively and share a common architecture. Each application is fitted with a customized scheduling agent that screens the available resource performance and dynamically produces a schedule for the application. Fundamental to the APPLeS approach is its ability to produce schedule that not only considers predicted expected resource performance but also the variation in that performance.

Unfortunately, the APPLeS have been developed for one application or for a specific set of applications and the designs cannot be easily re-targeted to other applications [2]. Specifically, application related components are embedded in the scheduling software itself. Given such a design, it can be tough to decide which modules need to be replaced to incorporate the requirements of the new application.

Nikolaos et al. [4] proposed three fair scheduling algorithms for the grid environment. In the SFTO (Simple Fair Task Order) policy, the jobs are arranged in the increasing order of their non-adjusted fair completion times. The non-adjusted fair completion times is obtained by the non-adjusted fair computational rates of the jobs by applying a max-min fair sharing algorithm. The AFTO (Adjusted Fair Task Order) scheme is an improved version of SFTO policy where the fair rates are adjusted dynamically each time the jobs become either active or inactive leading to the better exploitation of the processing capacities. In both the schemes, the processor to which the jobs are assigned is found based on the earliest completion time policy with the processor capacity gap taken into account. In the third scheduling scheme called MMFS (Max-Min Fair Scheduling) policy, the selection of a fair task queuing order and the selection of the processor assignment are

addressed simultaneously. The MMFS policy allocates the jobs to the available processor such that the actual job rate after the scheduling is close to that of the SFTO and AFTO as obtained by the max-min fair scheduling schemes. SFTO, AFTO and MMFS are designed for a multiprocessor computing environment. MMFS requires priori knowledge of the job workloads.

Sivakumar Viswanathan et al. [5] developed resource-aware dynamic incremental scheduling (RADIS) strategies for large-scale computational cluster systems. The strategies were designed to specifically take into account large volume of computationally intensive arbitrarily divisible loads with dynamic arrival rates, memory at the destination nodes for holding and processing of loads which varies over time and the load deadline requirements. An efficient "pull-based" scheduling strategy with admission control policy was integrated with RADIS to ensure that the admitted loads are processed within their deadlines. RADIS algorithm used divisible load theory (DLT) for efficiently handling large load volumes. In RADIS, the number of destination computing nodes that participate in processing varies over time. The limitations of RADIS are that it is explicitly designed for computing nodes with in a cluster. Hence, RADIS does not take into account the inter-cluster communication delay and RADIS performs a single-point admissibility test.

Young Choon Lee et al. [6] proposed a multiple queues with duplication (MQD) algorithm for bag of tasks applications for grid environments. In MQD approach, the scheduling decisions are made by indirectly taking into account the recent workload pattern of nodes and by using job duplication strategy. The jobs and nodes are sort in non-increasing order of their workload and initial deliverable processing speed. The initial deliverable processing speed of a node is defined as the node's available processing power during its initial scheduling process. In the initial scheduling process, reliable performance information of nodes compared to their initial deliverable processing speed is gathered by running a job on each node. On completion of the job, the performance ranking of the node on which the job is completed is computed. The performance of a computing node is quantified by dividing the workload of the last job the node completed by the amount of time taken for the job. The performance ranking decides to which node the job is assigned to. The adoption of job replication leads to better schedules and makes the scheduling decisions more resilient against resource failures. A disadvantage of MQD is that it assumed that the amount of computation associated with each job is fixed and cannot be changed.

Load balancing approaches

Jie Li et al. [7] proposed solutions to load balancing problems for a set of multiclass jobs for distributed and parallel computer systems. They constructed a general model which consisted of heterogeneous nodes which are interconnected by a generally configured interconnection network wherein there are several classes of jobs, each of which has its distinct delay function at each node and each communication link. This model was used to formulate the multiclass job load balancing problem as a nonlinear optimization problem where the objective is to minimize the mean response time of the job. The model assumes that the node processing delay and ink communication delay are known in advance. The load balancing approach considered by Jie Li is static and does not depend on the current state of the nodes.

Riky Subrata et al. [8] proposed a game-theoretic approach to load balancing for computational grids. The technique combined the inherent efficiency of centralized approach and the fault-tolerant nature of the decentralized approach. The method takes into consideration the changes in the system states during runtime. Also, the algorithm does not assume any particular distribution for the service times of the jobs. Rather, it only requires the first moment and the second moment of the service time as the input. However, the co-operative games are harder to achieve in a grid environment because of the need for cooperation and control between the players.

The article [9] introduced a behavioral model for parallel applications with great requirements for network, processing power and disk I/O resources. The behavioral model was tested on non-dedicated clusters where application resource demands are not known in advance. The model also addressed the issue of improving the bandwidth of the cluster networks at the software level thereby eliminating the need for the additional hardware.

Menglan Hu et al. [10] proposed requirement-aware strategies with arbitrary processor release times for scheduling multiple divisible loads. The strategies addressed two cases namely a case where the processors' release times are predefined and where the release times are unknown until the processors are released. Their technique also captured the job's processing requirements. Thus each job could only be processed by certain computing nodes. However they did not perform simultaneous processor communications which is very important for grid computing platform in order to achieve higher throughput.

Fault tolerance methods

An enormous amount of research efforts have been dedicated to fault tolerance in the scope of grid environments.

Checkpointing: Katsaros et al. [11] proposed a performance and effectiveness trade-off for check pointing in fault-tolerant distributed systems. The approach considered dynamic check pointing interval reduction if it leads to computational gains. This was computed as the sum of differences between the means for fault-affected and fault-unaffected job response times.

Li et al. [12] proposed a strategy for using adaptive fault tolerance in order to improve the application robustness on a TeraGrid. The strategy combined adaptive check pointing with proactive migration. It optimized the application execution time by considering the failure impact and the prevention costs. It took into account three types of actions namely skip checkpoint, take checkpoint and migrate and the appropriate action was selected based on the predicted failure frequency. Hence, the performance of this strategy depends on the quality of this prediction.

Replication: In passive replication scheme, the fault tolerant mechanisms are embedded within the algorithms. Passive replication scheme was first studied in [13] where one computing node was selected as the primary and all other computing nodes as backups. If the primary computing node fails, then a failover occurs and the backup computing node takes over.

Qin Zheng et al. [14] proposed fault tolerant scheduling strategies for independent, dependent and hybrid jobs with low replication costs using primary-backup approach for computational grids. They analyzed the impact of precedence constraints on scheduling and overloading of backups of dependent jobs and showed that this it limits the schedulability and overloading efficiency significantly. The strategies for independent and dependent jobs do not need sampling as required by traditional backup approaches.

Combined approaches: In [15], a scalable asynchronous low

overhead check pointing and replication based fault tolerance technique for MPI applications were proposed. This technique eliminated the need for central or network storage. The algorithm also demonstrated that effective check pointing could be achieved using the LAM/MPI implementations.

Discussions and Future Research Directions

We have presented a survey on grid scheduling, load balancing and fault tolerance for grid computing systems. We have focused on the core algorithms used in each of these individual areas, their advantages and disadvantages. We have also identified the future work required in each of these areas and have indicated that considering these areas collective will improve these techniques than considering them individually as shown in Table 1.

References

1. Foster, Ian, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared, In Grid Computing Environments Workshop, GCE'08 1-10.
2. Leal K, Huedo E, Ignacia M, Llorente, (2009) A Decentralized Model for Scheduling Independent Tasks in Federated Grids, Future Generation Computer System, 25: 840-852.
3. Berman, Francine, Wolski R, Casanova H, Cirne W, et al. (2003) Adaptive computing on the grid using AppLeS, IEEE Transactions on Parallel and Distributed Systems 14: 369-382.
4. Nikolaos D, Doulamis, Anastasios D, Doulamis, Emmanouel A, et al. (2007) Fair Scheduling Algorithms in Grids, IEEE Transactions on Parallel and Distributed Systems 18(11).
5. Viswanathan, Sivakumar, Veeravalli B, Thomas G, Robertazzi (2007) Resource-Aware Distributed Scheduling Strategies for Large-Scale Computational Cluster/Grid systems, Parallel and Distributed Systems, IEEE Transactions on Parallel & Distributed Systems 18(10): 1450-1461.
6. Lee, Choon Y, Albert Y, Zomaya (2007) Practical scheduling of bag-of-tasks applications on grids with dynamic resilience, Computers, IEEE Transactions on 56(6): 815-825.
7. Li J, Kameda H (1998) Load Balancing Problems for Multiclass Jobs in Distributed/Parallel Computer Systems, IEEE Transactions Computers 47(3): 322-332.
8. Subrata, Riky, Albert Y, Zomaya, Landfeldt B (2008) Game-theoretic approach for load balancing in computational grids, Parallel and Distributed Systems, IEEE Transactions on Computers 19(1): 66-76.
9. Qin X, Jiang H, Manzanares A, Ruan X, Yin S (2010) Communication Aware Load Balancing for Parallel Applications on Clusters, IEEE Transactions on Computers 59(1): 42-52.
10. Menglan HU, Bharadwaj Veeravalli (2011) Requirement-Aware Strategies with Arbitrary Processor Release Times for Scheduling Multiple Divisible Loads, IEEE Transactions on Parallel and Distributed Systems 22(10): 1697-1704.
11. Katsaros P, Angelis L, Lazos C (2007) Performance and Effectiveness Trade-Off for Checkpointing in Fault-Tolerant Distributed Systems, Concurrency and Computation: Practice and Experience 19(1):37-63.
12. Li Y, Lan Z (2007) Using Adaptive Fault Tolerance to Improve Application Robustness on the TeraGrid, Proc. TeraGrid Conf 1-5.
13. Budhiraja N, Marzulla K, Schneider FB, Toueg S (1992) Primary Backup Protocols: Lower Bounds and Optimal Implementations, Proc. Third IFIP Conf. Dependable Computing for Critical Applications (DCCA).
14. Zheng Q, Veeravalli B, Tham CK, (2009) On the Design of Fault-Tolerant Scheduling Strategies Using Primary-Backup Approach for Computational Grids with Low Replication Costs, IEEE Transactions on Computers 58(3):380-393.
15. Walters JP, Chaudhary V (2009) Replication-Based Fault Tolerance for MPI Applications, IEEE Transcations on Parallel and Distributed Systems 20(7): 997-1010.