IJRSET Volume 2, Issue 4 Pages: 18-24

ISSN 2394-739X

International Journal for Research in Science Engineering and Technology

Grid Scheduling and Computing –A Novel Approach for Distributed computing

[1] S.Vidya
[1] Head & Associate Professor,
[1] Departmont of Computer Applications,
[1] Cauvery College for Women,
[1] Trichy.

Abstract:-

Grid computing is a type of circulated computing that includes organizing and sharing computing, application, information stockpiling or system assets crosswise over element and topographically scattered associations. The objective of framework errands planning is to accomplish high framework throughput and to match the application need with the accessible computing assets. This is coordinating of assets in a non-deterministically shared heterogeneous environment. The multifaceted nature of booking issue increments with the measure of the lattice and turns out to be exceedingly hard to explain successfully. To get great systems to take care of this issue another range of examination is executed. This region is in light of created heuristic systems that give an ideal or close ideal answer for substantial networks. In this paper we present an errands planning algorithm for matrix computing. The algorithm is in light of recreated tempering system. The paper demonstrates to hunt down the best errands booking for lattice computing.

Keywords: - Grid computing, Simulated Annealing, Heuristics

1 INTRODUCTION

Grid computing came into being and is currently an active research area. One motivation of Grid computing is to aggregate the power of widely distributed resources, and provide non-trivial services to users. To achieve this goal, an efficient Grid scheduling system is an essential part of the Grid. Rather than covering the whole Grid scheduling area, this survey provides a review of the subject mainly from the perspective of scheduling algorithms.

In this review, the challenges for Grid scheduling are identified. First, the architecture of components involved in scheduling is briefly introduced to provide an intuitive image of the Grid scheduling process.

Then various Grid scheduling algorithms are discussed from different points of view, such as static vs. dynamic policies, objective functions, applications models, adaptation, QoS constraints, strategies dealing with dynamic behavior of resources, and so on. Based on a comprehensive understanding of the challenges and the state of the art of current research, some general issues worthy of further exploration are proposed. Task scheduling is an integrated part of parallel and distributed computing. Intensive research has been done in this area and

many results have been widly accepted. With the emergence of the computational grid, new scheduling algorithms are in demand for addressing new concerns arising in the grid environment. In this environment the scheduling problem is to schedule a stream of applications from different users to a set of computing resources to minimize the completion time. The scheduling involves matching of applications need with resource availability. There are three main phases of scheduling on a grid. Phase one is resource discovery, which generates a list of potential resources.

Phase two involves gathering information about those resources and choosing the best set to match the application requirements. In the phase three the task is executed, which includes file staging and cleanup.

In the second phase the choice of the best pairs of tasks and resources is NP-complete problem. A related scheduling algorithm for the traditional scheduling problem is Dynamic Level Scheduling (DLS) algorithm.

DLS aims at selecting the best subtask-machine pair for the next scheduling. To select the best subtask-machine pair, it provides a model to calculate the dynamic level of the task machine pair.

The overall goal is to minimize the computational time of the application. In the grid environment the scheduling algorithm no longer focuses on the subtasks of an application within a computational host or a virtual organization (clusters, network of workstations, etc.).

The goal is to schedule all the incoming applications to the available computational power. In some simple heuristics for dynamic matching and scheduling of a class of independent tasks onto a heterogeneous computing system have been presented. There are two different goals for task scheduling:

High performance computing and high throughput computing. The former aim is minimizing the execution time of each application and latter aim is scheduling a set of independent tasks to increase the processing capacity of the systems over a long period of time. Our approach is to develop a high throughput computing scheduling algorithm.

2. OVERVIEW OF THE GRID SCHEDULING PROBLEM

A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. It is a shared environment implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities. Resources can be computers, storage space, instruments, software applications, and data, all connected through the Internet and a middleware software layer that provides basic services for security, monitoring, resource management, and so forth. Resources owned by various administrative organizations are shared under locally defined policies that specify what is shared, who is allowed to access what, and under what conditions. The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.

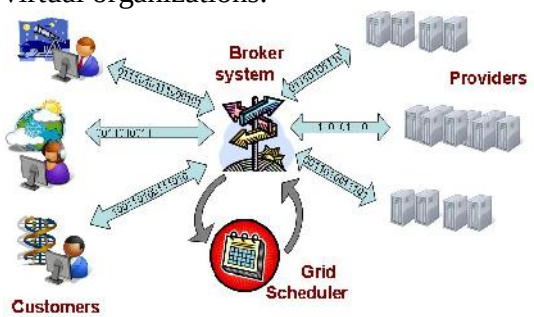

Figure :1 Grid Scheduling

From the point of view of scheduling systems, a higher level abstraction for the

Grid can be applied by ignoring some infrastructure components such as authentication, authorization, resource discovery and access control. Thus, in this paper, the following definition for the term Grid adopted: "A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements". To facilitate the discussion, the following frequently used terms are defined:

• A **task** is an atomic unit to be scheduled by the scheduler and assigned to a resource.

• The **properties** of a task are parameters like CPU/memory requirement, deadline, Priority, etc.

• A **job** (or **metatask**, or **application**) is a set of atomic tasks that will be carried out on a set of resources. Jobs can have a recursive structure, meaning that jobs are composed of sub-jobs and/or tasks, and sub-jobs can themselves be decomposed further into atomic tasks. In this paper, the term job, application and Meta task are interchangeable.

• A **resource** is something that is required to carry out an operation, for example: a processor for data processing, a data storage device, or a network link for data transporting.

• A **site** (or **node**) is an autonomous entity composed of one or multiple resources.

• A **task scheduling** is the mapping of tasks to a selected group of resources which may be distributed in multiple administrative domains.

2.1 The Grid Scheduling Process and Components

A Grid is a system of high diversity, which is rendered by various applications, middleware components, and resources. But from the point of view of functionality, we can still find a logical architecture of the task scheduling subsystem in Grid. For example, Zhu proposes a common Grid scheduling architecture. We can also generalize a scheduling process in the Grid into three stages: resource discovering and filtering, resource selecting and scheduling according to certain objectives, and job submission. As a study of scheduling algorithms is our primary concern here, we focus on the second step.

Based on these observations, Fig. 1 depicts a model of Grid scheduling systems in which functional components are connected by two types of data flow: resource or application information flow and task or task scheduling command flow.

2.2 . Grid Scheduling Algorithms: State of the Art

It is well known that the complexity of a general scheduling problem is NP-Complete. As mentioned in Section 1, the scheduling problem becomes more challenging because of some unique characteristics belonging to Grid computing. In this section, we provide a survey of scheduling algorithms in Grid computing, which will form a basis for future discussion of open issues in the next section.

2.3. A Taxonomy of Grid Scheduling Algorithms

In, Casavant et al propose a hierarchical taxonomy for scheduling algorithms in general-purpose parallel and distributed computing systems. Since Grid is a special kind of such systems, scheduling algorithms in Grid fall into a subset of this taxonomy. From the top to the bottom, this subset can be identified as what follows.

• Local vs. Global

At the highest level, a distinction is drawn between local and global scheduling. The local scheduling discipline determines how the processes resident on a single CPU are allocated and executed; a global scheduling policy uses information about the system to allocate processes to multiple processors to

optimize a system-wide performance objective. Obviously, Grid scheduling falls into the global scheduling branch.

• **Static vs. Dynamic**

The next level in the hierarchy (under the global scheduling) is a choice between static and dynamic scheduling. This choice indicates the time at which the scheduling decisions are made. In the case of static scheduling, information regarding all resources in the Grid as well as all the tasks in an application is assumed to be available by the time the application is scheduled. By contrast, in the case of dynamic scheduling, the basic idea is to perform task allocation on the fly as the application executes. This is useful when it is impossible to determine the execution time, direction of branches and number of iterations in a loop as well as in the case where jobs arrive in a real-time mode. These variances introduce forms of non-determinism into the running program.

Both static and dynamic scheduling are widely adopted in Grid computing.

3. SCHEDULING ALGORITHMS

The parameter sweep applications, created using a combination of task and data parallel models, contain a large number of independent jobs operating different data sets. A range of scenarios and parameters to be explored are applied to the program input values to generate different data sets. The programming and execution model of such applications resemble the SPMD (Single Program Multiple Data) model. The execution model essentially involves processing N independent jobs (each with the same task specification, but a different dataset) on M distributed computers where N is, typically, much larger than M. When the user submits a parameter sweep application containing N tasks along with quality of service requirements, the broker performs the following activities:

Resource Discovery:

Identifying resources and their properties and then selecting resource capable of executing user jobs.

2. Resource Trading: Negotiating and establishing service access cost using a suitable economic model.
3. Scheduling: Select resources that fit user requirements using scheduling heuristic/algorithm and map jobs to them.
4. Deploy jobs on resources [Dispatcher].
5. Monitor and Steer computations
6. Perform load profiling for future usage
7. When the job execution is finished, gather results back to the user home machine [Dispatcher].
8. Record all resource usage details for payment processing purpose.
9. Perform rescheduling: Repeat steps 3-8 until all jobs are processed and the experiment is within the deadline and budget limit.
10. Perform cleanup and post-processing, if required.

3.1. Adaptive Scheduling

An adaptive solution to the scheduling problem is one in which the algorithms and parameters used to make scheduling decisions change dynamically according to the previous, current and/or future resource status. In Grid computing, the demand for scheduling adaptation comes from three points:

the heterogeneity of candidate resources, the dynamism of resource performance, and the diversity of applications, as Fig. 4 shows. Correspondent with these three points,

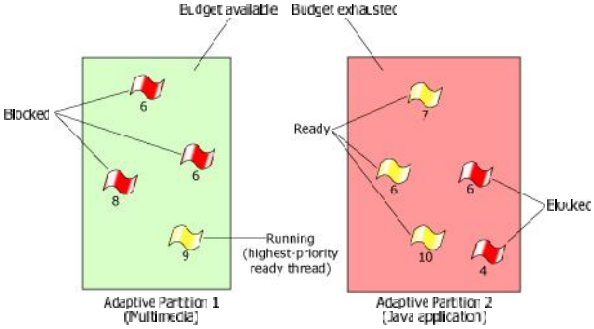

Figure: 2 Adaptive Scheduling

3.2 Data Scheduling

In high energy physics, bioinformatics, and other disciplines, there are applications involving numerous, parallel tasks that both access and generate large data sets, sometimes in the petabyte range. Data sets in this scale require specialized storage and management systems and data Grid projects are carried out to harness geographically distributed resources for such data-intensive problems by providing remote data set storage, access management, replication services, and data transfer protocols.

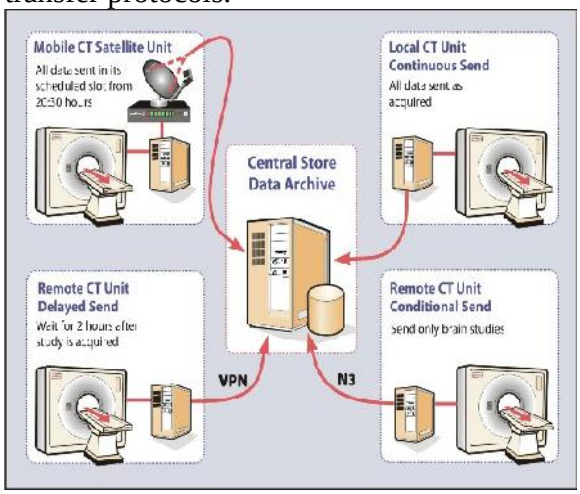

Figure: 3 Data Scheduling

As mentioned in Section 1, one important difference between Grid scheduling and its Traditional counterpart is that, in the latter, the data staging problem usually need not be Considered by scheduling algorithms. This is because the resources on which applications will run are determined before scheduling so that the data staging cost is a constant. The only cost relating to data transmission that should be considered is from the data produced at the run time, e.g., the data dependency in DAGs. In the Grid environment, by contrast, the location where an application finally is processed is usually selected in real time, so that the cost to transfer the input data from the storage sites to the processing sites might vary according to which processing sites are selected as well as which storage sites are used when the data have multiple replicas. Further, assigning a task to the machine that gives its best execution time may result in poor performance due to the cost of retrieving the required input data from data repositories. In, Park et al classify models of cost measured in makespan into five categories, namely 1) Local Data and Local Execution (here, local means where a job is submitted), 2) Local Data and Remote Execution, 3) Remote Data and Local Execution, 4) Remote Data and Same Remote Execution and 5) Remote Data and Different Remote Execution. The algorithms we have discussed in previous subsections can not be directly used to solve these problems, although they are helpful to find feasible answers

CONCLUSION

In spite of the fact that the Lattice has the attributes of heterogeneity and dynamicity, these elements are not straight dispersed in assets, but rather will be somewhat disseminated progressively and provincially as a rule, because of the synthesis of the Matrix assets. Current Lattice assets are generally dispersed in a grouped manner. To defy new difficulties in undertakings booking in a network situation, we display in this study heuristic booking algorithm. The proposed booking algorithm is intended to accomplish high throughput computing in a matrix domain. Thisis a NP-issue and to be tackled needs an exponential time. Accordingly the heuristic algorithm, which discovers a decent arrangement in a sensible time, is created. In this paper heuristic algorithm in view of mimicked tempering system is examined and it essential systems for a framework planning are figured. This algorithm ensures great burden adjusting of the machines and it is connected in element way.

Resources in the same bunch for the most part fit in with the same association and are

moderately more homogeneous and less dynamic in a given period. Inside a group, correspondence expense is normally low and the quantity of utilizations running in the meantime is typically little.

These conveyance properties may convey another plausibility for new algorithms to manage the Lattice challenges.

For instance, by taking multiphase or multilevel systems, a Lattice scheduler can first locate a coarse planning in the worldwide Network and after that a fine timetable in a nearby group.

This kind of method has the accompanying favorable circumstances, At the larger amount, where fine asset data is harder to acquire, the worldwide planning can utilize coarse data, (for example, burden adjusting, correspondence deferral of WAN connections) to give decentralized burden adjusting components. At the lower level, it is simple for nearby planning to use more particular data, (for example, data from a neighborhood forecaster) to settle on versatile choices.

REFRENCES

[1] A. Abraham, R. Buyya and B. Nath, Nature's Heuristics for Scheduling Jobs on Computational Grids, in Proc. of 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), pp. 45-52, Cochin, India, December 2000.

[2] A. K. Aggarwal and R. D. Kent, An Adaptive Generalized Scheduler for Grid Applications, in Proc. of the 19th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'05), pp.15-18, Guelph, Ontario Canada, May 2005.

[3] M. Aggarwal, R.D. Kent and A. Ngom, Genetic Algorithm Based Scheduler for Computational Grids, in Proc. of the 19th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'05), pp.209-215, Guelph, Ontario Canada, May 2005.

[4] A.H. Alhusaini, V.K. Prasanna, and C.S. Raghavendra, A Unified Resource Scheduling Framework for Heterogeneous Computing Environments, in Proc. of the 8th Heterogeneous Computing Workshop, pp.156-165, San Juan, Puerto Rico, April 1999.

[5] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, The Globus Striped GridFTP Framework and Server, in Proc. of the 2005 ACM/IEEE conference on Supercomputing, pp.54-64, Seattle, Washington USA, November 2005.

[6] M, Arora, S.K. Das, R. Biswas, A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments, in Proc. of International Conference on Parallel Processing Workshops (ICPPW'02), pp.:499 – 505, Vancouver, British Columbia Canada, August 2002.

[7] M. Aktaruzzaman, Literature Review and Survey: Resource Discovery in Computational Grids, School of Computer Science, University of Windsor, Windsor, Ontario, Canada.

[8] A. Andrieux, D. Berry, J. Garibaldi, S. Jarvis, J. MacLaren, D. Ouelhadj and D. Snelling, Open Issues in Grid Scheduling, Official Technical Report of the Open Issues in Grid Scheduling Workshop, Edinburgh, UK, October 2003.

[9] R. Bajaj and D. P. Agrawal, Improving Scheduling of Tasks in A Heterogeneous Environment, in IEEE Transactions on Parallel and Distributed Systems, Vol.15, no. 2, pp.107 – 118, February 2004.

[10] R. Freund, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems", 8th IEEE Heterogeneous Computing Workshop (HCW'99), San Juan, Puerto Rico, 1999, 30 44.

[11] N. metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, “Equition of State Calculations by Fast Computing Machines, Journal of Chemistry Physics 21(6), 1953, 1087-1092.
[12] I. H. Osman and C. N. Potts, “Simulated Annealing for Permutation Flow-Shop Scheduling”, Omega 17, 1989, 551-557.
[13] M. Pinedo, Scheduling: Theory, Algorithms and Systems, Prentice Hall, Englewood Clefts, NJ, 1995.
[14] C. R. Reeves (editor), Modern Heuristic Techniques for Combinatorial Problems, Oxford, England, Blackwell Scientific publications, 1993.
[15] V. V. Rene, Applied Simulated Annealing, Berlin, Springer, 1993.
[16] M. J. Schopf, “General Architecture for Scheduling on the Grid”, Special issue of JPDC on Grid Computing, 2002.
[17] G. C. Sih and E. A. Lee, “A Compile-Time Scheduling Heuristic for Inter Connection-Constrained Heterogeneous Processor Architectures”, IEEE transactions Parallel and Distributed Systems Vol 4, 1993, 175-187.