ISSN:0975-9646

Dispatcher Algorithm for Effective Resource Utilization of Server using Volunteer Computing : A Case Study

Jayaram C V, Rampur srinath
Dept.of PGCEA, The National Institute of Engg., Mysore, Karnataka, INDIA,

Abstract -Data partitioning and load balancing are important components of parallel computations. Many different partitioning strategies have been developed, with great effectiveness in parallel applications. But the problem of load-balancing the server is not yet solved completely. New applications and architectures require new partitioning features. Existing algorithms must be enhanced to support more complex applications. This paper discusses the design and implementation of the dispatcher algorithm using volunteer computing for effective utilization of the resource of server. Also presents a case study which examines the implementation of the dispatcher algorithm, by a server, A proper scheduling and efficient load balancing across the network can lead to improve overall system performance and a lower turn-around time for individual tasks.

Keywords: Volunteer Computing, load Balancing, resource utilization, Dispatch Algorithm, Task Scheduling.

1. INTRODUCTION

The term volunteer computing was coined by Luis F.G.Sarmenta, the developer of Bayanihan [1]. The first volunteer computing project on the internet was GIMPS [2], the Great Internet Mersenne Prime Search, which started in January 1996. Volunteer computing is a form of distributed computing in which the general public volunteers processing and storage resources to computing projects. Compared to other types of high-performance computing, volunteer computing has a high degree of diversity. The volunteered computers vary widely in terms of software and hardware type, speed, availability, reliability and network connectivity. Similarly, the applications and jobs vary widely in terms of their resource requirements and completion time constraints. Volunteering computing functionally combines globally distributed computers and information systems for creating a universal source of computing power and information. A volunteer can offer a resource balancing effect by scheduling tasks at machines with low utilization.

Volunteer computing systems, such as SETI@home[7], distributed.net [5], and a rapidly increasing number of other projects [8] for a good listing of different volunteer computing-type projects), are already demonstrating this great potential of volunteer computing. A large part of this success can be attributed to the relative ease with which users are able to join their systems. To volunteer, a user only needs to download an executable file for his own machine architecture and operating system. Install and run the software on the volunteer machines.

2. BACKGROUND

Load balancing is a technique to enhance resources, utilizing parallelism, exploiting throughput improvisation, and to cut response time through an appropriate distribution of the applications [3]. To minimize the decision time is one of the objectives for load balancing which has yet not been achieved. Proper task scheduling is the only efficient way to guarantee that submitted task are completed reliably and efficiently in case of process failure, processor failure, node crash, network failure, system performance degradation, communication delay, addition of new machines dynamically even though a resource failure occurs which changes the distributed environment [4]. Generally, load balancing mechanisms can be broadly categorized as centralized or decentralized, dynamic or static, and periodic or non-periodic [5]. All load balancing methods are designed such as, to spread the load on resources equally and maximize their utilization while minimizing the total task execution time. Selecting the optimal set of tasks for transferring has a significant role on the efficiency of the load balancing method as well as network resource utilization. This problem has been neglected by researchers in most of previous contributions on load balancing, either in distributed systems or in the network environment [6].There are different types of load balancing policies, strategies and categories which give different results to users in different environments or under different circumstances. Load balancing problem has been discussed in traditional distributed systems literature for more than two decades and various algorithms, strategies and policies have been proposed, classified and implemented [9]. Load balancing algorithms can be classified into two categories, static and dynamic.

Static Algorithms

Static load balancing algorithms allocate tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on average load of workstation cluster. The static load balancing is as shown in figure 1

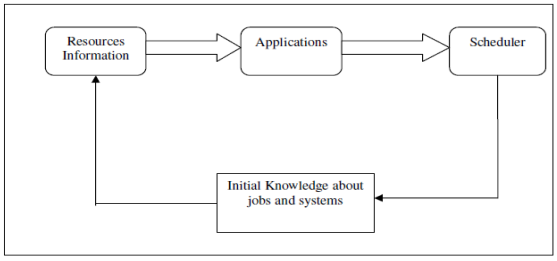

Figure 1 Static Load Balancing [9]

The decisions related to load balance are made at compile time when resource requirements are estimated. The advantage in this sort of algorithm is the simplicity in terms of both implementation as well as overhead, since there is no need to constantly monitor the workstations for performance statistics [10].

However, static algorithms only work well, when there is not much variation in the load on the workstations. Clearly, static load balancing algorithms aren't well suited to a grid environment, where loads may vary significantly at various times. A few static load balancing techniques are [19]:

- Round-Robin Algorithm: tasks are passed to processes in a sequential order, when the last process has received a task the schedule continues with the first process (a new round) [12].
- Randomized Algorithm: allocation of tasks to processes is random [13].

Drawbacks of Static Load Balancing Algorithms

- It is very difficult to estimate a-priori (in an accurate way) the execution time of various parts of a program.
- Sometimes there are communication delays that vary in an uncontrollable way.
- For some problems the number of steps to reach a solution is not known in advance.

Dynamic Algorithms

According to the name dynamic load balancing algorithms takes decision at run time, and use current or recent load information when making distribution decisions. In grid environment with dynamic load balancing allocate/reallocate resources at runtime based on no a priori task information, which determine when and which task has to be migrated [9]. The Dynamic load balancing is as shown in figure 2

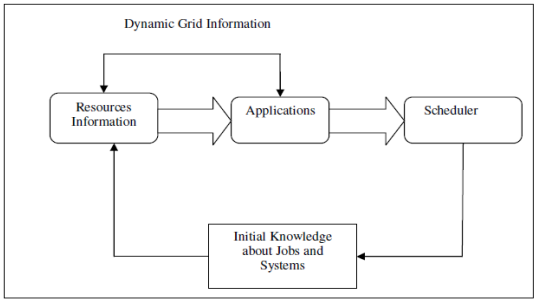

Figure 2 Dynamic Load Balancing [9]

After using effectively dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms. But this comes at the additional cost of collecting and maintaining load information, so it is important to keep these overheads within reasonable limits [14].

Load balancing policies

Load balancing algorithms can be based on many policies; some important policies are defined below [15].

- **Information policy:** This policy specifies what workload information should be collected, when it is to be collected and from where.
- **Triggering policy:** This policy determines the appropriate period to start a load balancing operation.
- **Resource type policy:** This policy classifies a resource as server or receiver of tasks according to its availability status.
- **Location policy:** This policy uses the results of the resource type policy to find a suitable partner for a server or receiver.
- **Selection policy:** This policy defines the tasks that should be migrated from overloaded resources (source) to most idle resources (receiver).

The main objective of load balancing methods is to speed up the execution of applications on resources whose workload varies at run time in unpredictable way. Hence it is significant to define metrics to measure the resource workload. Every dynamic load balancing method must estimate the timely workload information of each resource [16]. This is the key information in a load balancing system where responses are given to following questions

- How to measure resource workload?
- What criteria are retaining to define this workload?
- How to avoid the negative effects of resources dynamicity on the workload?
- How to take into account the resources heterogeneity in order to obtain an instantaneous average workload representative of the system?

Success of a load balancing algorithm depends upon stability of the number of messages (small overhead), support environment, low cost update of the workload, and short mean response time which is a significant measurement for a user. It is also essential to measure the communication cost induced by a load balancing operation, but to achieve all these, anyone would have to face great challenge in grid environment [16].

3. PROPOSED METHOD

The overall architecture is as shown in figure 3, According to the characteristics of applications, we propose an algorithm which dispatches the request by referencing CPU computing power**.** The main effort of this dispatching algorithm is to decide which server a TCP connection is going to be connected with. It is the place where dispatching decisions are made. Once a server is chosen and the

connection is constructed, all remote invocations go through this link are served by this server. Here we can have the channel objects periodically discard connections in purpose for the reconstruction of connections to less load servers. The network processor records the IP and port information of the client and the selected Server in the TCP connection table called TCT for each constructed connection.

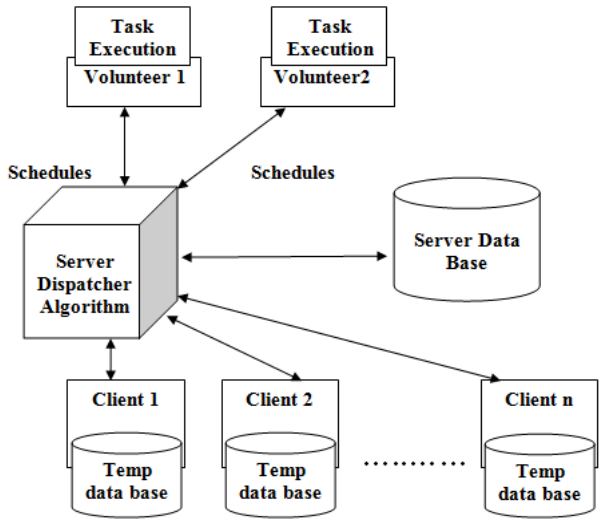

Figure 3Overall Architecture

The remote request packets with the same source IP, the same source port, and the same destination port will be directed to the same destination IP according to TCT. The response packets from the servers are also directed to the correct clients by this connection table. The destination port mentioned TCT is used to identify remoting services Different services distributed and then go to the different ports in our customized client channel objects. Dispatching algorithm is to find the least load server for dispatching. Different scheduling methods can be plugged in for this step. In the following, we propose a method to schedule tasks to the server minimizing the estimated task time.

Algorithm:
Step 1: TCP Configurations
Step 2: Check the TCP connection table entry
Step 3: Check if the incoming packets is in the range of stateful service.
Step 4: Check the packet is singleton or client activated method.
Step 5: Directs the connection construction request to the least load server.
Step 6: If the incoming method is indeed a stateful request, session table is created if the New stateful service.
Step 7: Checking for expiration of the session.
Step 8: Algorithm does TCP header rewriting to forward packets to and from the intended server of that connection.

Step1. Gets the IP address of system and wait for the client request in port number 8002. Step 2 of Algorithm does the checking to see if the incoming packet is already in a TCT entry. In addition, with the destination port plus the IP and port information of the client, we can construct a session table ST. ST is then used to track existing sessions for stateful services. If the network processor finds that no TCP connection exists for the incoming packet and the destination port shows that it belongs to a stateful service, it will then look up ST to find out the previous assigned server for this service. Step 3 checks if the incoming packet is in the range of stateful services. Step 4 checks if the incoming packet is a singleton or a client-activated method. Note that there are three kinds of methods in .Net remoting.

- Single call is stateless
- stateful methods include singleton
- Client-activated methods.

Our scheduling policy fully supports these three semantics for .Net methods. A time field is also kept in ST to determine the expiration of a session. Step 5 directs the connection construction request to least load servers. Once the least server is found, Step 6 checks if the incoming method is indeed a stateful request. A ST will be created if this is a new stateful request. Step 7 does the checking for the expiration of a session. The network processor can also invalidate the content in TCT and ST on purpose in order to reallocate stateful services to new servers for load-balancing issues, the Remoting proxy in the client side will detect a network failure exception and then can try to construct a new TCP connection to the backend. Finally, the algorithm does TCP header rewriting to forward packets to and from the intended server of that connection.

4. A CASE STUDY

Description: The main aim of this application is to effectively utilize the server resource on the network. The application consists of 3 main modules, Server, Client and a Volunteer. This application contains mainly to perform the operation of Watermarking effectively and with faster performance. The function to be performed will be sent to server and in turn the server sends it to volunteers. Depending upon the time taken for the execution of the task by the volunteers, the server allots the task to perform. Here the server uses Dispatching Algorithm to perform the task.

5. EXPERIMENTAL RESULTS

The experiment is conducted to perform 20 tasks distributed in three different scenarios, first is client request to watermark image with text, the server reads the image and waits for the text. As soon as the server receives text message to water mark the server looks for a volunteer RAM utilization and time required will be calculated. Then server dispatches the task on run time to volunteer one or two depends on the RAM resource utilization. Second scenario is client request to watermark image on image, the server reads the target image and waits for the image. Third scenario is

client request to watermark image with image and text and server reads image and waits for image and text. The experimental result is as shown in Table 1.

Volunteer	Time taken (ms) to perform Watermark image with text	Time taken (ms) to perform Watermark image with image	Time taken (ms) to perform Watermark image with image and text
1	0.171450770489879	0.153182824890442	0.0434275289545075
2	0.02139359283702	0.0584792353381	0.1256243217891987

Table 1: Time Taken to complete the tasks by each volunteers

6. CONCLUSION

This paper describes aspects of volunteer computing and introduces numerous concepts which illustrate its grand capabilities. Volunteer Computing is definitely a promising tendency to solve high demanding applications and its related problems. Main objective of the volunteer computing environment is to balance load and achieve high performance. Dynamic nature and complexity of network make load balancing very complex and vulnerable to faults. To maintain entire load of nodes is very hard due to dynamic nature of resources in a network environment. There are a number of factors, which can affect the server performance like load balancing, heterogeneity of resources and resource sharing in the network environment. It focuses on load balancing and presents factors due to which load balancing is initiated, compares existing load balancing algorithms and finally proposes an efficient dispatcher algorithm for network environment.

REFERENCES

[1] Y.Lan, T.Yu (1995) "A Dynamic Central Scheduler Load-Balancing Mechanism", Proc. 14th IEEE Conf. on Computers and Communication, Tokyo, Japan, pp.734-740.

[2] Guy Bernard,& Michel Simatic "A Decentralized and Efficient Algorithm for Load Sharing in Networks of Workstations

[3]. S.Rips "Load Balancing Support for Grid-enabled Applications" NIC Series, Vol. 33, ISBN 3-00- 017352-8, pp. 97-104, 2006.

[4].Belabbas Yagoubi and Yahya Slimani, "Dynamic Load Balancing Strategy for Grid Computing" Proceedings of World Academy of Science, Engineering and Technology Volume 13 May 2006 ISSN 1307-6884.

[5]. Porter, Michael; Mark Kramer. "The Link between Competitive Advantage and Corporate Social Responsibility". Harvard Business Review

[6]. Javier Bustos Jimenez, Robin Hood "An Active Objects Load Balancing Mechanism for Intranet"

[7] SETI@home. SETI@home home page. URL: http://setiathome.ssl.berkeley.edu.

[8] K. Pearson. Internet-based Distributed Computing Projects. URL: http://www.nyx.net/~kpearson/distrib.html

[9].Srikumar Venugopal, Rajkumar Buyya and Ramamohanarao Kotagiri "A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing" Grid Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia Email:fsrikumar, raj, raog@cs.mu.oz.au.

[10].Yuan-Jin Wen; Sheng-De Wang Department of Electrical Engineering, Division of Computer Science, National Taiwan University, Taipei, Taiwan "Minimizing Migration on Grid Environments: An Experience on Sun Grid Engine" Journal of Information Technology and Applications vol. 1 No. 4 March, 2007, pp. 297-30.

[11].Belabbas Yagoubi and Yahya Slimani, "Dynamic Load Balancing Strategy for Grid Computing" Proceedings of World Academy of Science, Engineering and Technology Volume 13 May 2006 ISSN 1307-6884.

[12].Jorge R. Ramos Vernon Rego, Department of Computer Sciences Purdue University West Lafayette, IN 47907, U.S.A., Janche Sang, Department of Computer and Info. Science Cleveland State University Cleveland, OH 44139, U.S.A "An Improved Computational Algorithm for Round-Robin Service" Proceedings of the 2003 Winter Simulation Conference

[13].Panos M. Pardalos, L.Pitsoulis1, T. Mavridou, and Mauricio G.C. Resende, "Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search and GRASP" Center for Applied Optimization and Department of Industrial and Systems Engineering, University of Florida,USA AT&T Bell Laboratories,USA.

[14].Shahzad Malik, "Dynamic Load Balancing in a Network of Workstations", 95.515F Research Report, November 29, 2000.

[15].Kai Lu, Riky Subrata and Albert Y. Zomaya, Networks & Systems Lab, School of Information Technologies, University of Sydney "An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems Considering Desirability of Grid Sites".

[16]. http://www.mersenne.org/"