IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Load Prediction on Grid: a Survey

Rahul Nayak
Department of Computer Science,TIT , Bhopal , M.P. , India
rahuln61@gmail.com

Abstract

Grid computing is a term referring to the association of computer assets from multiple administrative domains to reach a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. In this paper we work on different load generation and prediction scheme. In this paper we also define different load prediction methodologies like neural network, genetic algorithm etc.

Keywords: Grid computing, Distributed system, Load prediction method, Neural network, Genetic algorithm.

Introduction

What distinguishes grid computing from usual high performance computing systems such as cluster computing is that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed. Although a grid can be dedicated to a specialized application, it is more common that a single grid will be used for a variety of different purposes. Grids are often constructed with the aid of general-purpose grid software libraries known as middleware.

Grid size can vary by a considerable amount. Grids are a form of distributed computing whereby a “super virtual computer” is composed of many networked loosely coupled computers acting together to perform very large tasks. For certain applications, “distributed” or “grid” computing, can be seen as a special type of parallel computing that relies on complete computers (with onboard CPUs, storage, power supplies, network interfaces, etc.) connected to a network (private, public or the Internet) by a conventional network interface, such as Ethernet. This is in contrast to the traditional notion of a supercomputer, which has many processors connected by a local high-speed computer bus.

Grids and Conventional Super Computers

“Distributed” or “grid” computing in general is a special type of parallel computing that relies on complete computers (with onboard CPUs, storage, power supplies, network interfaces, etc.) connected to a network (private, public or the Internet) by a conventional network interface, such as Ethernet. This is in contrast to the traditional notion of a supercomputer, which has many processors connected by a local high-speed computer bus.

The primary advantage of distributed computing is that each node can be purchased as commodity hardware, which, when combined, can produce a similar computing resource as multiprocessor supercomputer, but at a lower cost. This is due to the economies of scale of producing commodity hardware, compared to the lower efficiency of designing and constructing a small number of custom supercomputers. The primary performance disadvantage is that the various processors and local storage areas do not have high-speed connections. This arrangement is thus well-suited to applications in which multiple parallel computations can take place independently, without the need to communicate intermediate results between processors. The high-end scalability of geographically dispersed grids is generally favorable, due to the low need for connectivity between nodes relative to the capacity of the public Internet

There are also some differences in programming and deployment. It can be costly and difficult to write programs that can run in the environment of a supercomputer, which may have a custom operating system, or require the program to address concurrency issues. If a problem can be adequately parallelized, a “thin” layer of “grid” infrastructure can allow conventional, standalone programs, given a different part of the same problem, to run on multiple machines. This makes it possible to write and debug on a single conventional machine, and eliminates complications due to multiple instances of the same program running in the same shared memory and storage space at the same time.

Load Balancing Problem

The development of computational grids and the associated middleware has been actively pursued in recent years to deal with the emergence of greedy applications of large computing tasks and amounts of data. Managing such applications leads to some complex problems for which traditional architectures are insufficient. There are many potential advantages of use grid architectures, including the ability to solve large scale advanced scientific and engineering applications whose computational requirements exceed local resources, and the reduction of job turnaround time through workload balancing across multiple computing facilities.

Load Parameters

Cpu Load

The system load is a measure of the amount of work that a computer system performs. The load average represents the average system load over a period of time. It conventionally appears in the form of three numbers which represent the system load.

Cpu Load vs Cpu Utilization

The comparative study of different load indices carried out by Ferrari et al. reported that CPU load information based upon the CPU queue length does much better in load balancing compared to CPU utilization. The reason CPU queue length did better is probably because when a host is heavily loaded, its CPU utilization is likely to be close to 100% and it is unable to reflect the exact load level of the utilization. In contrast, CPU queue lengths can directly reflect the amount of load on a CPU. As an example, two systems, one with 3 and the other with 6 processes in the queue, will probably have utilizations close to 100% although they obviously differ.

Network Load

Network load is calculated in the given five parameters in our system.

1. Speed
2. Bytes Sent:
3. Bytes Received:
4. Download speed
5. upload speed

Data Analysis And Data Format

Analysis of data is a process of inspecting, clean-up, transforming, and modeling data with the purpose of importance useful information, suggesting conclusions, and supporting judgment making. Data analysis has multiple facets and approaches, surrounding diverse techniques under a variety of names, in different business, science, and social science domains.

Data mining is a data analysis technique that focuses on modeling and information discovery for extrapolative rather than purely expressive purposes. Business intelligence covers data analysis that relies heavily on aggregation, focusing on business information. In statistical applications, some people divide data analysis into descriptive statistics, exploratory data analysis (EDA), and confirmatory data analysis (CDA). EDA focuses on discovering new features in the data and CDA on confirming or falsifying existing hypotheses. Predictive analytics focuses on submission of statistical or structural models for predictive forecasting or classification, while text analytics applies statistical, linguistic, and structural techniques to extract and classify information from textual sources, a species of unstructured data. All are varieties of data analysis.

Data integration is a precursor to data analysis, and data analysis is closely linked to data visualization and data dissemination. The term data analysis is sometimes used as a synonym for data modeling

Data analysis is a process, within which several phases can be distinguished.

Data cleaning

Data cleaning is an important procedure during which the data are inspected, and erroneous data are—if necessary, preferable, and possible—corrected. Data cleaning can be done during the stage of data entry. If this is done, it is important that no subjective decisions are made. The guiding principle provided by Adder (ref) is: during subsequent manipulations of the data, information should always be cumulatively retrievable. In other words, it should always be possible to undo any data set alterations. Therefore, it is important not to throw information away at any stage in the data cleaning phase. All information should be saved (i.e., when altering variables, both the original values and the new values should be kept, either in a duplicate data set or under a different variable name), and all alterations to the data set should carefully and clearly documented, for instance in a syntax or a log.

Initial data analysis

The most important distinction between the initial data analysis phase and the main analysis phase, is that during initial data analysis one refrains from any analysis that are aimed at answering the original research question. The initial data analysis phase is guided by the following four questions

The raw data are unstructured and individual listings aren't always clean-cut and complete as far as the fields listed above are concerned, this is problematic.

Thus we select data for experiment purpose in Table format

CPU	SPEED	SEND	RECIVED	DOWNLOAD	UPLOAD
10	1313	2332	3245	322	32
12	2343	3243	4354	3232	32

Prediction Techniques

Neural-Network

Most neural network approaches to the problem of forecasting use a multilayer network trained using the back propagation algorithm. Consider a time series x(1)……x(t). Where it is required to forecast the value of x(t+1) The inputs to the multilayer network are typically chosen as the previous k values x(t-k+1)……..x(t). And the output will be the forecast. The network is trained and tested on sufficiently large training and testing sets, which are extracted from the historical time series. In addition to previous time series values, one can utilize as inputs the values or forecasts of other time series (or external variables) that have a correlated or causal relationship with the series to forecast.

As is the case with many neural-network applications, preprocessing the inputs and the outputs can improve the results significantly. Input and output preprocessing means extracting features from the inputs and transforming the target outputs in a way that makes it easier for the network to extract useful information from the inputs and associate it with the required outputs. Preprocessing is considered an "art," and there are no set rules to choose it. Even some very intuitively appropriate transformations may turn out of no value when checking the actual results. For our case the main inputs are the previous time series values.

Genetic algorithm

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

In a genetic algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Genetic algorithms find application in bioinformatics, phylogenetics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields.

A typical genetic algorithm requires:

1. A genetic representation of the solution domain,
2. A fitness function to evaluate the solution domain.

A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The

fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used.
Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions (usually randomly) and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

Server Load

Load, in computing, is a measure of the amount of processing a computer system is currently performing, usually in the form of a scalar and as some variation on a percentage. - Wikipedia definition.
In a *nix variant, Server load can be calculated by the uptime, top or the w command.

```
[root@localhost ~]# uptime

15:33:18 up 1:33, 3 users, load average: 0.01, 0.05, 0.09
```

The last 3 values show the server load for 1 , 5 and 15 minutes, in that order. This means that for the last 1 minute, the Average server load was 0.01. For the last 5 minutes, the average server load was 0.05 and for the last 15 minutes, the average server load was 0.09.
For Servers with multiple processors, load is calculated by dividing the Load with the number of processors.

Actual load = Total load (as shown in uptime) / no. of CPUs

When there are multiple processors, the load gets evenly distributed among the CPUs. If one processor is busy, the task can choose another processor to perform the computation.

HIGH LOAD

In Internet servers or most of the application server installed to work, High loads are caused by diverse reasons, each requiring a different approach for correction.

POWER USERS

In a virtual hosting environment, there are many users who use the servers for maintaining websites or different data based applications. Some websites are simple, and some are Processor hungry, bandwidth hungry ecommerce applications. If you or your clients have big ecommerce sites with a lot of visitors, your load may shoot up.

Many users have Database driven sites when such sites have lot of visitors, the number of Database connections increases and results in high load. Ideally an Internet server shouldn't have more than 300-400 small to medium websites. If any of these users turn out to be power hungry, they could destabilise all the other users operations by hogging all the resources.

Script Kiddies And Attacks

Many times high loads are caused because the server wasn't secure enough and got cracked. The cracker started running IRC scripts or egg drops. Sometimes spamming could be reason.
In some cases, a valid user turns into a monster and starts illegally abusing the system by spamming or running insecure scripts.

The first indication of a problem is high load. That should give the server administrator enough hints that something is wrong somewhere.

Run "top" on *Nix systems to see which processes are causing the load. Kill such processes and check out where they originated from. You may discover that something is up. Be suspicious about anything that you see that is causing load.

Running back Up On The Server, daily Stats, server Task

Sometimes high load is caused due to valid maintenance tasks such as Daily Backups, Daily stats updates and Cron schedulers. It is normal that the server loads shoot up during these times. There is nothing to worry during this time, provided that high load doesn't sustain for long. Therefore schedule such tasks during non-Business hours when the users are minimum and therefore less likely to be affected.

Overselling

Irresponsible Overselling by Web Hosts is a major reason why web Hosts have Server load problems. In overselling what happens is that the web host sells more resources(like space, bandwidth) than is present. The Web Host assumes that all users are not going to be using the space or bandwidth they purchased. For example, the government builds roads thinking that all the people aren't going to use them at the same time. If the entire population were to step out of their houses, the roads would be totally inadequate.

Overselling is not all that bad, provided the Web Host is responsible in checking server load and acting upon it. Even if a Web host decides to oversell, the load has to be constantly monitored 24/7. If it reaches

2 or 3 fairly regularly, maybe it is time to move some websites to new servers.

Conclusion

From the above discussion and study of different papers we found the following facts that are responsible for high server load.

1. Number of concurrent users are increased
2. Number of Request are larger than the allocated bandwidth
3. No of processors running on the server
4. No of processes are running over the server
5. Different type of script or attacks
6. Server backup running on background

We conclude the above facts after study but working and solving all the problems in one solution is required but it is quite expensive and complex task.

References

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, San Fransisco, CA, 1999.

[2] P.A. Dinda, D.R. O'Hallaron, "Host load prediction using linear models," *Cluster computing 3(4)*, pp. 265-280, 2000.

[3] L. Yang, J.M. Schopf, and I. Foster, "Conservative scheduling: Using predicted variance to improve decisions in dynamic environment," *Supercomputing'03*, pp. 1-16,2003.

[4] P.A. Dinda, "A prediction-based real-time scheduling advisor," *Proc. 16th Int'l Parallel and Distributed Processing Symp.(IPDPS 2002)*, pp. 35-42, 2002

[5] D. Lu, H. Sheng, and P. Dinda, "Size-based scheduling policies with inaccurate scheduling information," *12th IEEE Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, pp. 31-38, 2004.

[6] S. Jang, X. Wu, and V. Taylor, "Using performance prediction to allocate grid resources," Technical report, GriPhyN 2004-25, pp. 1-11, 2004.

[7] L. Yang, I. Foster, and J.M. Schopf, "Homeostatic and tendency-based CPU load predictions," *Int'l Parallel and Distributed Processing Symp. (IPDPS'03)*, pp. 42-50, 2003.

[8] R. Wolski, N. Spring, and J. Hayes, "Predicting the CPU Availability of Time-shared Unix Systems on the Computational Grid," *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, pp. 1-8, 1999.

[9] M. Swany and R. Wolski, "Multivariate resource performance forecasting in the network weather service," *Supercomputing'02*, pp. 1-10, 2002.

[10] R. Wolski, "Dynamically forecasting network performance using the network weather service," *Journal of Cluster Computing*, Vol.1, pp.119-132, 1998.

[11] R. Wolski, "Experiences with predicting resource performance on-line in computational grid settings," *ACM SIGMETRICS Performance Evaluation Review*, Vol.30, No.4, pp. 41-49, 2003.

[12] P.A. Dinda, "The statistical properties of host load," Technical report, CMU, pp. 1-23, 1998.

[13] S. Akioka and Y. Muraoka, "Extended forecast of CPU and network load on computational grid," *2004 IEEE Int'l Symp. on Cluster Computing and the Grid*, pp. 765-772, 2004.

[14] J. Liang, K. Nahrstedt, and Y. Zhou, "Adaptive Multi- Resource Prediction in Distributed Resource Sharing Environment," *2004 IEEE Int'l Symp. on Cluster Computing and the Grid*, pp. 1-8, 2004.

[15] G. Box, G. Jenkins, and G. Reinsel, *Time Series Analysis, Forecasting and Control*, Prentice Hall, 1994.

[16] http://www.cs.cmu.edu/~pdinda/LoadTraces