Load Balancing and Process Management over Grid Computing

Javed hussain¹, Durgesh Kumar Mishra²

¹ Research Scholar, Mewar University, Chittorgarh, Rajasthan, India
² Sri Aurobindo Institute of Technology, Indore, Madhya Pradesh, India

Abstract

As the technology is grown the need of computational efficiency is increases day by day. If we can see around even our mobile phone becomes more efficient and with high computational resources. To empower computation in new principals and concepts changes the face of computation, due to the introduction grid computing and cloud computing the computation achieves new directions of computing. In this paper we start with the traditional computing, and go through the different computing schemes and produces the concept and effects on the resources required to execute any task in local systems and as well as the remote and process migration over efficient computational strategies.

Keywords: computation, grid computing, traditional computing, load parameters and balancing.

1. INTRODUCTION

In this era of technology each and every day in our life interacted with the computers and their complex applications, these applications are help us to perform complex task in few minutes. This fast and easy computation ability is developed using a lot of efforts and work force are required to achieve the efferent computing. As we know that all the computational task are keep two main factors the resources required to compute any task and the time consumed to processes the complete task. Additionally when we talk about the advance computing we discussed about the different network parameters, which is required to provide the request and response of any task execution. In this era of computational advancement maximum applications, interfaces and complex task are become online and these tasks are executed using high efficient cloud and grid servers. To execute this task only some parameters are required to send as requests and they execute complete task in seconds, additionally the out puts are found in remote machine as a real time execution. These evolutionary pressures generate new requirements for distributed application development and deployment. Today, applications and middleware are typically developed for a specific platform (e.g., Windows NT, a flavour of UNIX, a mainframe, J2EE, Microsoft .NET) that provides a hosting environment for running applications. The capabilities provided by such platforms may range from integrated resource management functions to database integration, clustering services, security, workload management, and problem determination—with different implementations, semantic behaviours, and APIs for these functions on different platforms. But in spite of this diversity, the continuing decentralization and distribution of software, hardware, and human resources make it essential that we achieve desired qualities of service (QoS)—whether measured in terms of common security semantics, distributed workflow and resource management performance, coordinated fail-over, problem determination services, or other metrics—on resources assembled dynamically from enterprise systems, service provider systems, and customer systems. We require new abstractions and concepts that allow applications to access and share resources and services across distributed, wide area networks. The execution time of a CPU-bound task on a host is tightly related to the CPU load on the host. In fact, for some applications the relationship between the execution time of a CPU bound task and the measured load during the execution is almost perfectly linear. If we could predict the load during the execution of a task on a host, we could predict easily the execution time of the task on the host. Therefore, host load prediction can be useful for guiding scheduling strategies to achieve high application performance and efficient resource use.

2. BACKGROUND

Traditionally to execute any application required memory and a strong processor for task execution during the execution they consume the resources locally, Where the application is required to execute, additionally to execute any task locally required to enable the local machines with the software frameworks and relevant classes and library.

As the size of data is increases the resource requirements on the system is increases readily due to this the other
applications running on the system becomes slow down. As the above given fig shows a general task execution required memory resources from the local machines. That is depends upon the used library and or application server where the task is executed. In addition we compute the web server task executions and the performance due to the task execution for that purpose we test for client side load and server side memory resources for successfully task executions.

As the above graph shows that the task execution on the remote web server and their impact on local machine due the above study we found that task consumes server resources rather than local client machine resources. But if the number of request arrived for execution the server becomes slowed down and they not respond in the proper manner and sometimes cresses occur in the web servers. To manage the load on these servers various clustering and network load management schemes are proposed and implemented but they are not much effective due to the need of computation efficiency.

6 PROPOSED WORK

To overcome the need of computational efficiency some new efforts are made using the computation using the grid servers, grid is a group or alliances where more than one machine is involve for providing the efficient computing and fast responses. Such problems have been for some time a central concern of the developers of distributed systems for large-scale scientific research. Work within this community has led to the development of Grid technologies. We defined Grid technologies and infrastructures as supporting the sharing and coordinated use of diverse resources in dynamic, distributed “virtual organizations”. We defined essential properties of Grids and introduced key requirements for protocols and services, distinguishing among connectivity protocols concerned with communication and authentication, resource protocols concerned with negotiating access to individual resources, and collective protocols and services concerned with the coordinated use of multiple resources.

Grid computing in general is a special type of parallel computing that relies on complete computers (with on board 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. Grid computing is the association of computer resources from many administrative fields to grasp a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. What differentiates grid computing from orthodox high routine computing systems such as cluster computing is that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed. Although a single grid can be devoted to a particular application, commonly a grid is used for a variety of purposes. Grids are frequently constructed with general-purpose grid middleware software collections or library. Each node on the grid receives a piece of the problem, which consists of a collection of original problem cells (OPCs). An OPC is the smallest piece into which the problem is divided, and each one needs to communicate and share data with its neighbours. Optimal Grid automates this communication and attempts to minimize the amount of network communication needed to solve a problem. When the program for the application is loaded, the middleware automatically partitions the problem using the following procedures:

1. Determine the complexity.
2. Identify the number of nodes available.
3. Use algorithms to predict the optimal number of grid nodes needed to solve the problem.
4. Optionally interact with the user to divide the problem into an optimal number of pieces. Whether the user or Optimal Grid partitions the problem, the middleware predicts the computation time for the problem.
5. Partition the application data into OPCs.
6. Allow the user the option to customize the data. In assessing stress on an airplane wing, for example, the user might decide to remove one or two rivets from a particular place.
7. Launch the program.

“Distributed” or “grid” computing in general is a special type of parallel computing that relies on complete computers (with on board 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. Faults are formed during different kind of resources unavailability, any network related problems such as conventional system. These faults related problem is arises during the following regions. [1]

- Network errors
  - Packet Loss
  - Packet corruption
- Time faults
- Early faults
- Late faults (performance failure)
- Response faults
- Value faults
State transition faults
• Omission faults
• Physical faults
Faulty CPUs
Faulty memory
Faulty storage
Etc.
• Life-cycle faults
Versioning faults
• Interaction faults
Protocol incompatibilities
Security incompatibilities
Policy problems
Timing overhead

The above given problems are related to different types of errors and faults. To design a strong and effective system for fault tolerance strategy required to evaluate different fault factors and their effecting parameters. And also required to keep watch these factors over the job scheduling and their respective effects.

7 IDENTIFIED RESEARCH GAP

Grid computing is an organization of high performance computing devices and these devices are organized by a high speed network architecture major faults are arises are provided in. To simplify these faults and tolerating the effects over grids computational efficiency required to work around the below given problems.
1. There are various faults and their causes and handling additionally prevention techniques are different from each other
2. There are no predictive schemes are found where we estimate a fault comes using upcoming data analysis.
3. There a lot of intermediate check based schemes are proposed but these schemes add some additional time and other resource overhead.
4. There are not any acceptable methods are found which may work under a serious or major breakdown occur
5. There is not any system available to reschedule and release the ideal resources rapidly under overload conditions.

The number of dynamic resources in the grid system increases continuously, so fault tolerance becomes a critical property for applications running on these resources. However, in traditional implementations, when a failure occurs, the whole application is shut down and has to be restarted from the beginning [1]. Some of the existing fault tolerance and recovery mechanisms like checkpoint-recovery and over provisioning are discussed in [2], [3]. In the proposed study work we our main aim to find the possible causes of faults and there effective parameters, additionally design and a robust architecture to predict upcoming faults using data analysis. This predictive scheme helps the designed to prepare future strategy for fault tolerance and resource management.

8 PROPOSED WORK

To overcome the problem of fault detection and prevention we divide complete work under some small modules.

1. Problem analysis: find out different categories of faults and related effects first. After that required to define different parameters by which the effecting parameters are relate to each other.

2. Prepare a data model: after analysis required designing a data model by which system analyse the upcoming data and predict the fault and its reason.

3. Group different fault tolerance technique: create methods and techniques to handle the faults according to need, in other words manage a backup plan for any predictive error.

4. Apply job scheduling and other background task: simulate the tolerance strategy for smooth work driven

Fig shows the proposed work

This Proposed System is given below. It takes parametric data from the network and prepares the strategy for upcoming faults

9 CONCLUSION AND FUTURE WORK

In this paper we provide the actual problem identification and their related work analysis in future we develop grid sim based simulation for our desired task, additionally in future we provide the complete implementation and results analysis, for future work we divide the complete task in three main modules first implementation of the load forecasting, in second module we provide the load balancing scheme and at last work we provide the fault tolerance and avoidance mechanism

References


[6] Performance comparison of hierarchical checkpoint protocols grid computing, Special Issue on Distributed Computing and Artificial Intelligence
