Load Balancing in Distributed Data Base and Distributed Computing System

Lovely Arya

Research Scholar

Dravidian University

KUPPAM, ANDHRA PRADESH

Abstract

With a distributed system, data can be located closer to their point of use. This can reduce communication costs, compared to a central system. Faster response: Depending on how data are distributed, most requests for data by users at a particular site can be satisfied by data stored at that site. This speeds up query processing since communication and central computer delays are minimized. It may also be possible to split complex queries into sub-queries than can be processed in parallel at several sites, providing even faster response. Distributing the data encourages local groups to exercise greater control over "their" data, which promotes improved data integrity and administration. At the same time, user can access non-local data when necessary. Hardware can be chosen for the local site to match the local, not global, data processing work. work group. It is often easier and more economical to add a local computer and its associated data to the distributed network than to expand a large central computer. Also, there is less chance of disruption to existing users than is the case when a central computer system is modified or expanded.

Introducton

There are many different types of distributed computing systems and many challenges to overcome in successfully designing one. The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems. Openness is the property of distributed systems such that each subsystem is continually open to interaction with other systems. Web Services protocols are standards which enable distributed systems to be extended and scaled. In general, an open system that scales has an advantage over a perfectly closed and self contained system. Once something is published in an open system, it cannot be taken back. Pluralism Different subsystems of an open distributed system include heterogeneous, overlapping and possibly conflicting information. There is no central arbiter of truth in open distributed systems. Asynchronously, different subsystems

can come up and go down and communication links can come in and go out between subsystems of an open distributed system. Therefore the time that it will take to complete an operation cannot be bounded in advance.

Development of Distributed Systems

The reasons behind the development of distributed systems were the availability of powerful microprocessors at low cost as well as significant advances in communication technology. The availability of powerful yet cheap microprocessors led to the development of powerful workstations that satisfy a single user's needs. These powerful stand-alone workstations satisfy user need by providing such things as bitmapped displays and visual interfaces, which traditional time-sharing mainframe systems do not support.

When a group of people works together, there is generally a need to communicate with each other, to share data, and to share expensive resources (such as high quality printers, disk drivers, etc). This requires interconnecting computers and resources. Designing such systems became feasible with the availability of cheap and powerful microprocessors, and advances in communication technology. When a few powerful workstations are interconnected and can communicate with each other, the total computing power available in such a system can be enormous. Such a system generally only costs tens of thousands of dollars. On the other hand, if one tries to obtain a single machine with the computing power equal to that of a network of workstations, the cost can be as high as a few million dollars. Hence, the main advantage of distributed system is that they have a decisive price/performance advantage over more traditional time-sharing systems.

Allocation of Task in Distributed Computing System

It consists of multiple heterogeneous computing nodes with their memories and clocks that communicate with each other by message passing mechanism. Over last several years, these have become popular for high performance computing and information processing .The main incentives for choosing DCS are higher system throughput and improved availability. A distributed processing system has the application software portioned into a set of program modules; allocation of these modules to the processors is an important research issue. Several task allocation algorithms for distributed computer system have been reported in literature .These algorithms consider execution time of the different modules of a task, executing on different processing nodes. The task assignment problem (NP-hard) tries to maximize the throughput of the system by minimizing the cost .This cost may be in terms of time that is execution time or in terms of bytes of data transferred. The module are allocated onto processing nodes so as to minimize the time taken .This can be done by maximizing and balancing the utilization of resources while

minimizing the communication cost between these processors. But both criteria are conflicting, as the load balancing calls for distributing the tasks over different processors while minimizing of inter processor communication drives the task assignment to assign all the tasks onto a single processor. The task assignment problem in DCS is NP Complete .Hence satisfactory suboptimal solutions obtainable in a reasonable amount of computation time are generally sought.

Distributed Computing System Models

Various models are used for building distributed computing systems. These models can be broadly classified into five categories-minicomputer, workstation, workstation-workstations server processor-pool, and hybrid.

The minicomputer model is a simple extension of the centralized time-sharing system. A distributed computing system based on this model consists of minicomputers (they may be large supercomputers as well) interconnected by a communication network. Each minicomputer usually has multiple users simultaneous logged on to it. For this, several interactive terminals are connected to each minicomputer each user is logged on to one specific minicomputer, with remote access to other Minicomputers. The network allows a user to access remote resources that are available at some machine other than the one on to which the user is currently logged. The minicomputer model may be used when resource sharing (such as sharing information databases of different types, with each type of database located on a different machine) with remote users is desired.

Distributed Operating System

An operating system as a program that control, the resources of a computer system and provides its users with an interface or virtual machine that is more convenient to use than the bare machine. According to this definition, the two primary tasks of an operating system are as follows

- To present users with a virtual machine that is easier to program than the underlying hardware.
- To manage the various resources of the system. This involves performing such tasks as keeping track of who is using which resource, granting resource requests, accounting for resource usage, and mediating conflicting requests from different programs and users The operating systems commonly used for distributed computing systems can be broadly classified into two types-network operating systems and distributed operating systems. The three most important features commonly used to differentiate between these two types of operating systems are system image, autonomy, and fault tolerance capability.

Features of Distributed Operating System

These features are :-

- **System image**. The most important feature used to differentiate between the two types of operating systems is the image of the distributed computing system from the point of view of its users. In case of a network operating system, the users view the distributed computing system as a collection of distinct machines connected by a communication subsystem. A distributed operating system hides the existence of multiple computers and provides a single-system image to its users. That is, it makes a collection of networked machines act as a virtual uni processor.
- **Autonomy**. In the case of a network operating system, each computer of the distributed computing system has its own local operating system (the operating systems of different computers may be the same or different), and there is essentially no coordination at all among the computers except for the rule that when two processes of different computers communicate with each other, they must use a mutually agreed on communication protocol. With a distributed operating system, there is a single system wide operating system and each computer of the distributed computing system runs a part of this global operating system. The distributed operating system tightly interweaves all the computers of the distributed computing system in the sense that they work in close cooperation with each other for the efficient and effective utilization of the various resources of the system.
- **Fault tolerance capability**. A network operating system provides little or no fault tolerance capability in the sense that if 10% of the machines of the entire distributed computing system are down at any moment, at least 10% of the users are unable to continue with their work. On the other hand, with a distributed operating system, most of the users are normally unaffected by the failed machines and can continue to perform their work normally, with only a 10% loss in performance of the entire distributed computing system. Therefore, the fault tolerance capability of a distributed operating system is usually very high as compared to that of a network operating system

  Distributed systems are classified into three broad categories, namely, the Minicomputer model, the workstation model, and the processor pool mode.

Classification of Load Distributing Algorithms

The basic function of a load-distributing algorithm is to transfer load (tasks) from heavily loaded computers to idle or lightly loaded computers. Load distributing algorithms can be broadly characterized as static, dynamic or adaptive. Dynamic load distributing

algorithms use system state information (the loads at nodes), at least in part, to make load-distributing decisions, while static algorithms make no use of such information. In static load distributing algorithms, decisions are hard-wired in the algorithm using priori knowledge of the system. Dynamic load distributing algorithms have the potential to outperform static load distributing algorithms attributing algorithms have the potential to outperform static load distributing algorithms because they are able to exploit short term fluctuations in the system state to improve performance.

Use of Distributed Systems

The alternative to using a distributed system is to have a huge centralized system, such as a mainframe. For many applications there are a number of economic and technical reasons that make distributed systems much more attractive than their centralized counterparts. Cost. Better price/ performance as long as commodity hardware is used for the component computers. Performance. By using the combined processing and storage capacity of many nodes, performance levels can be reached that are beyond the range of centralized machines. Scalability. Resources such as processing and storage capacity can be increased incrementally. Reliability. By having redundant components the impact of hardware and software faults on users can be reduced. Inherent distribution. Some applications, such as email and the Web (where users are spread out over the whole world), are naturally distributed. This includes cases where users are geographically dispersed as well as when single resources (e.g., printers, data) need to be shared. However, these advantages are often offset by the following problems encountered during the use and development of distributed systems: New component: network. Networks are needed to connect independent nodes and are subject to performance limitations. Besides these limitations, networks also constitute new potential points of failure. Security. Because a distributed system consists of multiple components there are more elements that can be compromised and must, therefore, be secured. This makes it easier to compromise distributed systems. Software complexity. As will become clear throughout this course distributed software is more complex and harder to develop than conventional software; hence, it is more expensive to develop and there is a greater chance of introducing errors.

Distributed Data Collection and Analysis

The monitor both collects and analyzes information regarding the execution of a program in real-time. As a result, the monitor may be used to generate on-line displays of execution information, and it may be combined with tools that use such on-line information in order to change or steer the running program. Furthermore, for dynamic program changes that concern the improvement of program performance (program

tuning), the overhead and latency of distributed information collection and analysis must not significantly reduce the performance gains realized by those changes. In response to these needs, the monitor is designed to permit tradeoffs in the amount of information collected and analyzed, the extent and the accuracy of analysis, and the extent of program adaptation. Such tradeoffs are realized in part by use of alternative means of information collection, such as tracing versus sampling.

Conclusion

In this paper , we have discussed several issues and challenges involved in providing The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems because of balancing the load using load balancing algorithms. Openness is the property of distributed systems such that each subsystem is continually open to interaction with other systems. Web Services protocols are standards which enable distributed systems to be extended and scaled. In general, an open system that scales has an advantage over a perfectly closed and self contained system. A through overview of many applications there are a number of economic and technical reasons that make distributed systems much more attractive than their centralized counterparts. **Cost**. Better price/ performance as long as commodity hardware is used for the component computers. **Performance**. By using the combined processing and storage capacity of many nodes, performance levels can be reached that are beyond the range of centralized machines. **Scalability.** Resources such as processing and storage capacity can be increased incrementally

Reference

- R. Sears and E. Brewer, “Statis: Flexible transactional storage,” in Proceedings of Symposium on Operating Systems Design and Implementation (OSDI), 2006.
- P. G. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price, “Access path selection in a relational database management system,” in Proceedings of ACM-SIGMOD International Conference on Management of Data, pp. 22–34, Boston, June 1979.
- P. Seshadri, H. Pirahesh, and T. Y. C. Leung, “Complex query decorrelation,” in Proceedings of 12th IEEE International Conference on Data Engineering (ICDE), New Orleans, February 1996.
- M. A. Shah, S. Madden, M. J. Franklin, and J. M. Hellerstein, “Java support for data-intensive systems: Experiences building the telegraph dataflow system,” ACM SIGMOD Record, vol. 30, pp. 103–114, 2001.

- L. D. Shapiro, "Exploiting upper and lower bounds in top-down query optimization," International Database Engineering and Application Symposium (IDEAS), 2001.
- A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts. McGraw-Hill, Boston, MA, Fourth ed., 2001.
- M. Steinbrunn, G. Moerkotte, and A. Kemper, "Heuristic and randomized optimization for the join ordering problem," VLDB Journal, vol. 6, pp. 191–208, 1997.
- M. Stonebraker, "Retrospection on a database system," ACM Transactions on Database Systems (TODS), vol. 5, pp. 225–240, 1980.
- M. Stonebraker, "Operating system support for database management," Communications of the ACM (CACM), vol. 24, pp. 412–418, 1981.