A

Seminar report

On

Distributed Computing

Submitted in partial fulfillment of the requirement for the award of degree
of Bachelor of Technology in Computer Science

SUBMITTED TO:

www.studymafia.org

SUBMITTED BY:

www.studymafia.org

Acknowledgement

I would like to thank respected Mr…….. and Mr. ……..for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

Secondly, I would like to thank my parents who patiently helped me as i went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs.

Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time.

Preface

I have made this report file on the topic **Distributed Computing;** I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to ……..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

Contents

Introduction

The numbers of real applications are still somewhat limited, and the challenges--particularly standardization--are still significant. But there's a new energy in the market, as well as some actual paying customers, so it's about time to take a look at where distributed processing fits and how it works.

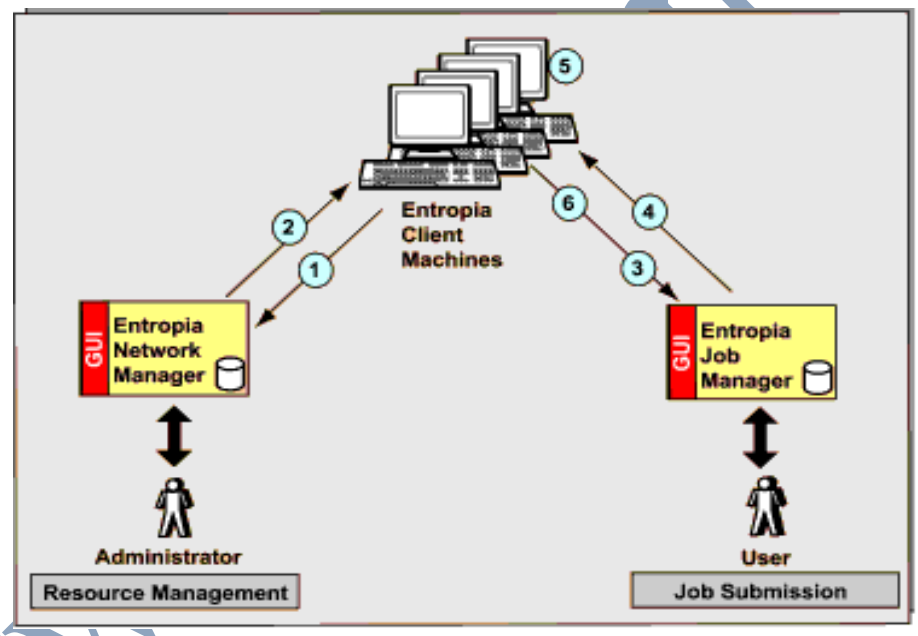
Increasing desktop CPU power and communications bandwidth has also helped to make distributed computing a more practical idea. Various vendors have developed numerous initiatives and architectures to permit distributed processing of data and objects across a network of connected systems.

One area of distributed computing has received a lot of attention, and it will be a primary focus of this paper--an environment where you can harness idle CPU cycles and storage space of hundreds or thousands of networked systems to work together on a processing-intensive problem. The growth of such processing models has been limited, however, due to a lack of compelling applications and by bandwidth bottlenecks, combined with significant security, management, and standardization challenges.

How It Works

A distributed computing architecture consists of very lightweight software agents installed on a number of client systems, and one or more dedicated distributed computing management servers. There may also be requesting clients with software that allows them to submit jobs along with lists of their required resources.

An agent running on a processing client detects when the system is idle, notifies the management server that the system is available for processing, and usually requests an application package. The client then receives an application package from the server and runs the software when it has spare CPU cycles, and sends the results back to the server. If the user of the client system needs to run his own applications at any time, control is immediately returned, and processing of the distributed application package ends.



A Typical Distributed System

Distributed Computing Management Server

The servers have several roles. They take distributed computing requests and divide their large processing tasks into smaller tasks that can run on individual desktop systems (though sometimes this is done by a requesting system). They send application packages and some client management software to the idle client machines that request them. They monitor the status of the jobs being run by the clients. After the client machines run those packages, they assemble the results sent back by the client and structure them for presentation, usually with the help of a database.

The server is also likely to manage any security, policy, or other management functions as necessary, including handling dialup users whose connections and IP addresses are inconsistent. Obviously the complexity of a distributed computing architecture increases with the size and type of environment. A larger environment that includes multiple departments, partners, or participants across the Web requires complex resource identification, policy management, authentication, encryption, and secure sandboxing functionality.

Administrators or others with rights can define which jobs and users get access to which systems, and who gets priority in various situations based on rank, deadlines, and the perceived importance of each project. Obviously, robust authentication, encryption, and sandboxing are necessary to prevent unauthorized access to systems and data within distributed systems that are meant to be inaccessible.

If you take the ideal of a distributed worldwide grid to the extreme, it requires standards and protocols for dynamic discovery and interaction of resources in diverse network environments and among different distributed computing architectures. Most distributed computing solutions also include toolkits, libraries, and API's for porting third party applications to work with their platform, or for creating distributed computing applications from scratch.

Comparisons and Other Trends

Distributed vs. Grid Computing

There are actually two similar trends moving in tandem--distributed computing and grid computing. Depending on how you look at the market, the two either overlap, or distributed computing is a subset of grid computing. Grid Computing got its name because it strives for an ideal scenario in which the CPU cycles and storage of millions of systems across a worldwide network function as a flexible, readily accessible pool that could be harnessed by anyone who needs it.

Sun defines a computational grid as "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to computational capabilities." Grid computing can encompass desktop PCs, but more often than not its focus is on more powerful workstations, servers, and even mainframes and supercomputers working on problems involving huge datasets that can run for days. And grid computing leans more to dedicated systems, than systems primarily used for other tasks.

Large-scale distributed computing of the variety we are covering usually refers to a similar concept, but is more ge

ared to pooling the resources of hundreds or thousands of networked end-user PCs, which individually are more limited in their memory and processing power, and whose primary purpose is not distributed computing, but rather serving their user.

Distributed vs. Other Trends

	Distributed Computing Network	Supercomputer	Cluster
Cost	As low as 1% of the cost of supercomputers, and much less expensive than clusters	Huge initial capital outlay with immediate depreciation	Substantial initial capital outlay with immediate depreciation
Scalability	Unlimited	None	Limited
Hardware Requirements	Minimal - Enterprises use PC resources already owned	Requires dedicated supercomputing hardware with space allocation, cooling, and power considerations	Requires multiple dedicated computers with space allocation, cooling, and power considerations
Administrative Support	One IT staff member can manage tens of thousands of member machines from a single central network management server	IT intensive	IT intensive
Nodes	Member computers are not dedicated - full member computer utility is maintained	Must be dedicated	Must be dedicated
Risk of Failure	Low	High	High
Maintenance	Low - automatically adjusts for member machines that go off line	High. Spare parts, lost productivity downtime, and system administrator time	High. Spare parts, lost productivity downtime, and system administrator time
Obsolescence	Low - self-upgrading software; network value automatically increases as PCs are added or upgraded	High	High

Application Characteristics

Obviously not all applications are suitable for distributed computing. The closer an application gets to running in real time, the less appropriate it is. Even processing tasks that normally take an hour are two may not derive much benefit if the communications among distributed systems and the constantly changing availability of processing clients becomes a bottleneck. Instead you should think in terms of tasks that take hours, days, weeks, and months. Generally the most appropriate applications consist of "loosely coupled, non-sequential tasks in batch processes with a high compute-to-data ratio." The high compute to data ratio goes hand-in-hand with a high compute-to-communications ratio, as you don't want to bog down the network by sending large amounts of data to each client, though in some cases you can do so during off hours. Programs with large databases that can be easily parsed for distribution are very appropriate.

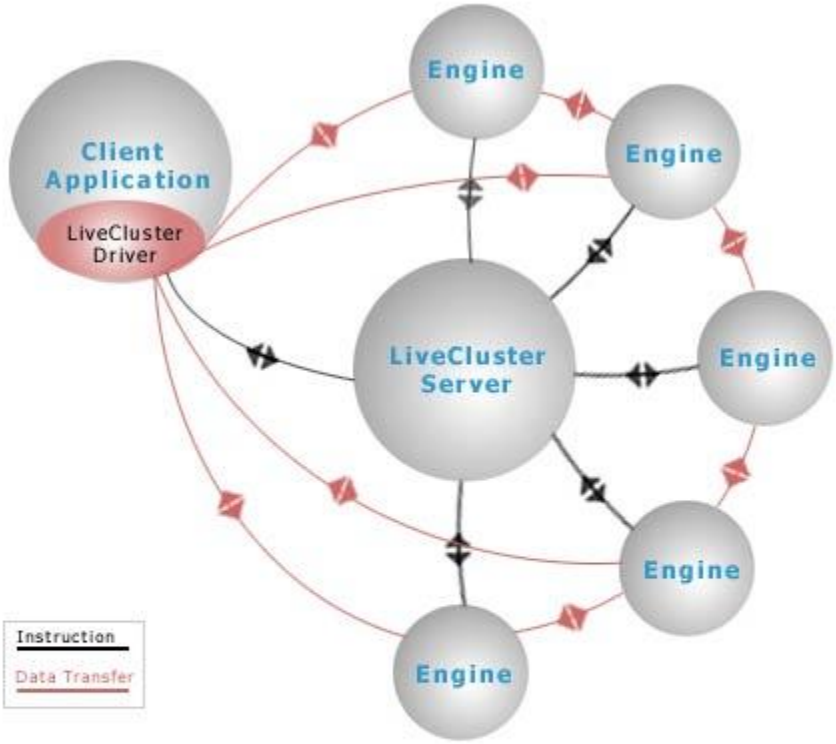
Clearly, any application with individual tasks that need access to huge data sets will be more appropriate for larger systems than individual PCs. If terabytes of data are involved, a supercomputer makes sense as communications can take place across the system's very high speed backplane without bogging down the network. Server and other dedicated system clusters will be more appropriate for other slightly less data intensive applications. For a distributed application using numerous PCs, the required data should fit very comfortably in the PC's memory, with lots of room to spare.

Taking this further, it is recommended that the application should have the capability to fully exploit "coarse-grained parallelism," meaning it should be possible to partition the application into independent tasks or processes that can be computed concurrently. For most solutions there should not be any need for communication between the tasks except at task boundaries, though Data Synapse allows some interprocess communications. The tasks and small blocks of data should be such that they can be processed effectively on a modern PC and report results that, when combined with other PC's results, produce coherent output. And the individual tasks should be small enough to produce a result on these systems within a few hours to a few days

Types of Distributed Computing Applications

The following scenarios are examples of types of application tasks that can be set up to take advantage of distributed computing.

- A query search against a huge database that can be split across lots of desktops, with the submitted query running concurrently against each fragment on each desktop.
- Complex modeling and simulation techniques that increase the accuracy of results by increasing the number of random trials would also be appropriate, as trials could be run concurrently on many desktops, and combined to achieve greater statistical significance (this is a common method used in various types of financial risk analysis).
- Exhaustive search techniques that require searching through a huge number of results to find solutions to a problem also make sense. Drug screening is a prime example.
- Complex financial modeling, weather forecasting, and geophysical exploration are on the radar screens of the vendors, as well as car crash and other complex simulations.
- Many of today's vendors are aiming squarely at the life sciences market, which has a sudden need for massive computing power. Pharmaceutical firms have repositories of millions of different molecules and compounds, some of which may have characteristics that make them appropriate for inhibiting newly found proteins. The process of matching all these **ligands** to their appropriate targets is an ideal task for distributed computing, and the quicker it's done, the quicker and greater the benefits will be.



Example Applications

Financial Services:
- Derivatives Pricing
- Risk Management
- Trading Research
- Asset/Liability Management
- Data Mining

Energy:
- Energy Trading
- Seismic Processing
- Computational Chemistry
- Reservoir Modeling

Security and Standards Challenges

The major challenges come with increasing scale. As soon as you move outside of a corporate firewall, security and standardization challenges become quite significant. Most of today's vendors currently specialize in applications that stop at the corporate firewall, though Avaki, in particular, is staking out the global grid territory. Beyond spanning firewalls with a single platform, lies the challenge of spanning multiple firewalls and platforms, which means standards.

Most of the current platforms offer high level encryption such as Triple DES. The application packages that are sent to PCs are digitally signed, to make sure a rogue application does not infiltrate a system. Avaki comes with its own PKI (public key infrastructure). Identical application packages are typically sent to multiple PCs and the results of each are compared. Any set of results that differs from the rest becomes security suspect. Even with encryption, data can still be snooped when the process is running in the client's memory, so most platforms create application data chunks that are so small, that it is unlikely snooping them will provide useful information. Avaki claims that it integrates easily with different existing security infrastructures and can facilitate the communications among them, but this is obviously a challenge for global distributed computing.

Working out standards for communications among platforms is part of the typical chaos that occurs early in any relatively new technology. In the generalized peer-to-peer realm lies the Peer-to-Peer Working Group, started by Intel, which is looking to devise standards for communications among many different types of peer-to-peer platforms, including those that are used for edge services and collaboration.

The Global Grid Forum is a collection of about 200 companies looking to devise grid computing standards. Then you have vendor-specific efforts such as Sun's Open Source JXTA platform, which provides a collection of protocols and services that allows peers to advertise themselves to and communicate with each other securely. JXTA has a lot in common with JINI, but is not Java specific (thought the first version is Java based).

Intel recently released its own peer-to-peer middleware, the Intel Peer-to-Peer Accelerator Kit for Microsoft .Net, also designed for discovery, and based on the Microsoft.Net platform.

Advantages

- **Economics:-**
  - Computers harnessed together give a better price/performance ratio than mainframes.
- **Speed:-**
  - A distributed system may have more total computing power than a mainframe.
- **Inherent distribution of applications:-**
  - Some applications are inherently distributed. E.g., an ATM-banking application.
- **Reliability:-**
  - If one machine crashes, the system as a whole can still survive if you have multiple server machines and multiple storage devices (redundancy).
- **Extensibility and Incremental Growth:-**
  - Possible to gradually scale up (in terms of processing power and functionality) by adding more sources (both hardware and software). This can be done without disruption to the rest of the system.

Disadvantages

- **Complexity :-**
  - Lack of experience in designing, and implementing a distributed system. E.g. which platform (hardware and OS) to use, which language to use etc.
- **Network problem:-**
  - If the network underlying a distributed system saturates or goes down, then the distributed system will be effectively disabled thus negating most of the advantages of the distributed system.
- **Security:-**
  - Security is a major hazard since easy access to data means easy access to secret data as well.

Conclusion

The advantages of this type of architecture for the right kinds of applications are impressive. The most obvious is the ability to provide access to supercomputer level processing power or better for a fraction of the cost of a typical supercomputer.

Scalability is also a great advantage of distributed computing. Though they provide massive processing power, super computers are typically not very scalable once they're installed. A distributed computing installation is infinitely scalable--simply add more systems to the environment. In a corporate distributed computing setting, systems might be added within or beyond the corporate firewall.

For today, however, the specific promise of distributed computing lies mostly in harnessing the system resources that lies within the firewall. It will take years before the systems on the Net will be sharing compute resources as effortlessly as they can share information.

References

- www.google.com
- www.wikipedia.com
- www.studymafia.org