Distributed Computing

Introduction to Distributed Computing

Summary

- Definitions of Distributed Computing
- Definitions of Distributed System
- What is Distributed Computing
- Examples of Distributed Computing
- Why we Distribute Applications
- Distributed Computing Advantages
- Distributed Computing Disadvantages
- Alternatives and Forms of DC
- Challenges in Distributing Applications

Reference:

http://library.thinkquest.org/C007645/english/0-definition.htm

zulqarnain@iiu.edu.pk

Definitions of Distributed Computing

Different definitions of distributed computing from different recourses and authors like Wikipedia, Sun and Microsoft

► A type of computing in which different components and objects of an application can be located on different computers connected to a network. So, for example, a word processing application might consist of an editor component on one computer, a spell-checker object on a second computer, and a thesaurus on a third computer. In some distributed computing systems, each of the three computers could even be running a different operating system.
One of the requirements of distributed computing is a set of standards that specify how objects communicate with one another.

► A computer system in which several interconnected computers share the computing tasks assigned to the system

► Distributed computing is a programming model in which processing occurs in many different places (or nodes) around a network. Processing can occur wherever it makes the most sense, whether that is on a server, Web site, personal computer, handheld device, or other smart device.

► A type of system that divides a workload to computers connected to a network. The network may be either be enclosed in a room or out in the open, like the Internet. Distributed computing is also referred to as distributed processing, cooperative computing, and collective computing

► Distributed computing means programs located on physically separated computers which must communicate in some way in order to complete a given computing task

► Distributed computing is the process of aggregating the power of several computing entities to collaboratively run a single computational task in a transparent and coherent way, so that they appear as a single, centralized system

Definitions of Distributed System

► A distributed System as one in which hardware and software components located at networked computers communicate and coordinate their action only by only passing messages.

► A distributed system is an application that consists of components running on different computers concurrently. These components communicate via some telecommunications network. Together, they provide some service to users.

An example of a distributed system is the World Wide Web. As you are reading a web page, you are actually using a distributed system.
As you are browsing the web, your web browser running on your own computer communicates with different web servers that provide web pages. Possibly, your browser uses a proxy server to access the web contents stored on web servers faster and more securely. To find these servers, it also uses the DNS system, which is itself a distributed system running on yet another set of computers. Your web browser communicates with all of these servers over the Internet, via a system of routers which are themselves part of a large distributed routing system using different protocol.

Together, web servers, proxies, and browsers, DNS servers and Internet routers comprise a very-large-scale distributed system that provides access to information (web pages and other content) that is spread all over the world.

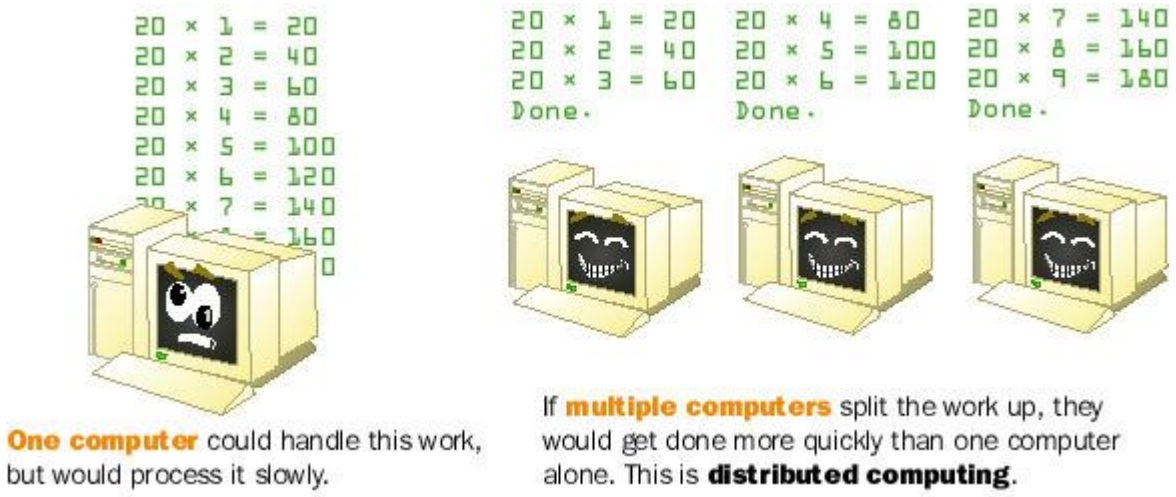
zulqarnain@iiu.edu.pk

What is Distributed Computing?

We are quite familiar with the power of computers. They can store data, record music, balance checkbooks, play games with us, process and print our words, and open a whole encyclopedia before our eyes. It's not a sweeping statement to say that computers have made our lives different. It's hard to envision a life before the age of computers.

If a single computer can change our life, why not connect several of them? Over the past two decades, networks of computers have even further changed the way businesses operate and the way the government functions.

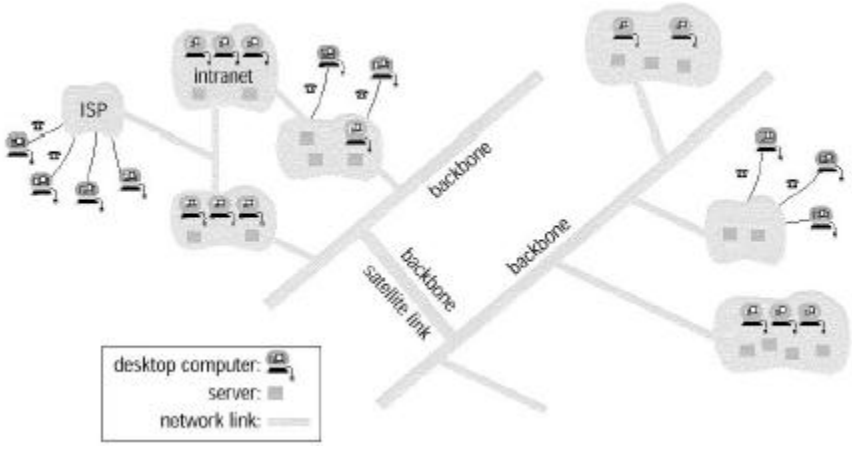
Distributed computing is the next step in computer progress, where computers are not only networked, but also smartly distributes their workload across each computer so that they stay busy and don't squander the electrical energy they feed on. This setup rivals even the fastest commercial supercomputers built by companies like IBM or Cray. When you combine the concept of distributed computing with the tens of millions of computers connected to the Internet, you've got the fastest computer on Earth.



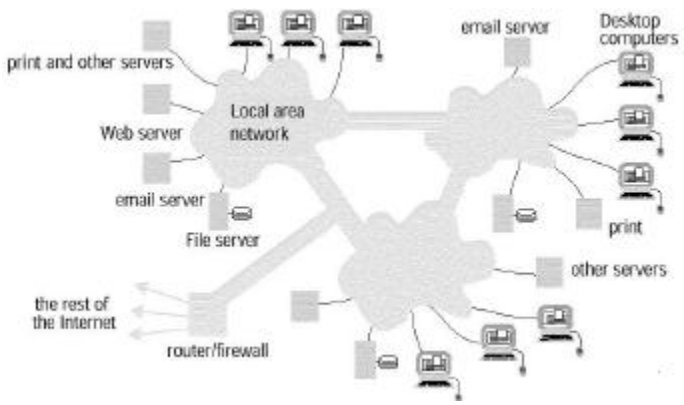
Regular Computing vs. Distributed Computing

Examples of Distributed Computing

- The Internet: Internet is a heterogeneous network of computers and applications implemented through Internet Protocol stack



- Intranets:
  Locally administered network Usually proprietary (e.g., the University campus network) Interfaces with the Internet through (e.g., proxy server, firewalls) Provides services internally and externally



- Telephony systems
- World Wide Web

Why we Distribute Applications

Why would one want to develop a distributed application in the first place? As you know, distribution introduces a whole new set of difficult issues. However, sometimes there is no choice; some applications by their very nature are distributed across multiple computers because of one or more of the following reasons:

- The *data* used by the application are distributed
- The *computation* is distributed
- The *users* of the application are distributed

Data are Distributed

Some applications must execute on multiple computers because the data that the application must access exist on multiple computers for administrative and ownership reasons. The owner may permit the data to be accessed remotely but not stored locally. Or perhaps the data cannot be co-located and must exist on multiple heterogeneous systems for historical reasons.

Computation is Distributed

Some applications execute on multiple computers in order to take advantage of multiple processors computing in parallel to solve some problem. Distributed systems consisting of collections of microcomputers may have processing powers that no supercomputers will ever achieve. 10000 CPUs, each running at 50 MIPS, yields 500000 MIPS such that instruction to be executed in 0.002 nsec. Distributed applications can take advantage of the scalability and heterogeneity of the distributed system. In distributed computation we achieve load distribution such that the overall system performance is optimized.

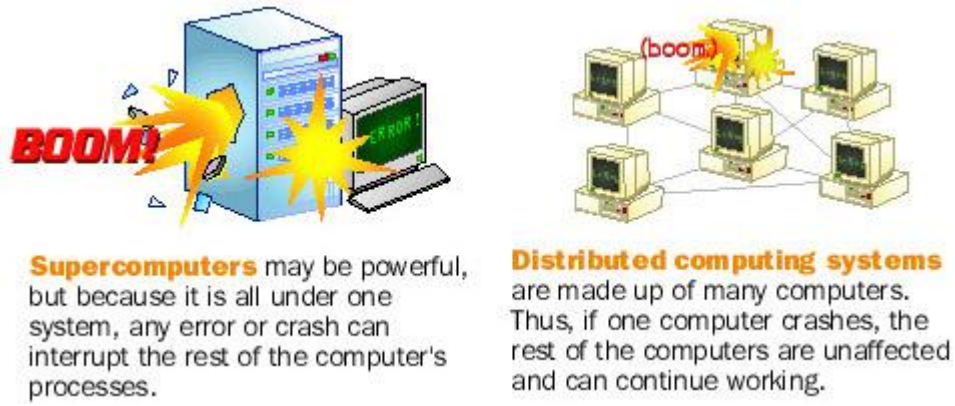
Users are Distributed

Some applications execute on multiple computers because users of the application communicate and interact with each other via the application. Each user executes a piece of the distributed application on his or her computer, and shared objects, typically execute on one or more servers.

Distributed Computing Advantages

Distributed computing first and foremost advantage over traditional supercomputers is its **frugality**, making use of every spare moment your computer's processor is idle. The latest Pentium chip sits unused most of the time while your monitor flashes a screen saver or while your keyboard records your typing. These basic functions use very little processing power, while the rest goes to waste. Distributed computing can take full advantage of a computer's capabilities by keeping it busy with numbers to calculate.

It's also easy on the wallet. If enough users sign up, these linked computers — often referred to as virtual parallel machines — can surpass the fastest supercomputer by as much as four times for a fraction of the supercomputer's cost. More power for less money — what scientist or engineer with a large, overwhelming project could resist the concept of more power for less money? Provide a slick screensaver with an amazing visual of the data that's being processed by the computer, and Internet users will sign up in droves, adding to the computing power. This is one of the reasons SETI@Home's project is so popular.



The Reliability of Distributed Computing

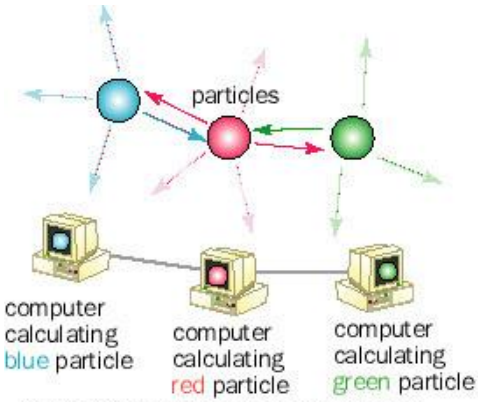
Reliability is no less important than speed. With a supercomputer, any one problem may bring the system to its knees. If you distribute the workload across several computers, however, there are fewer problems since each computer is independent and its problems don't affect the other computers. Less time is spent troubleshooting the problem.

Distributed Computing Disadvantages

All the computers on the Internet may be powerful when combined, but there needs to be something to combine and coordinate all of them to work towards one goal. Server computers are still needed to distribute the pieces of data and collect the results from participating clients. Collecting Internet users and having them sign up for a distributed computing program is not an automatic process: often times, organizations need to attract users with offers and incentives.

Distributed computing must be flexible as well because of the nature of the Internet. Many people who use the Internet still do so using a regular modem, so communication with participating computers can be a slow. For this reason, data must take up as little bandwidth and as little disk space as possible. The data must also be independent of each other.

Certain calculation processes depend on many variables and thus the computer's processor chip needs to consult all the other computers to solve these calculations. Consider an *n*-body simulation, which calculates the effects of a force field on atoms, whether it be a gas or a fluid. One processor calculates the path of one of these atoms. These calculations *depend on* the forces acting on it from other atoms, which are calculated by the other processors. Thus, the processor relies on its fellow processors and needs to communicate with them. Lack of effective communication would compromise the speed of the calculation process. Since most home computers are not continuously online, the Internet does not offer this luxury to the computer's processor.



particles' positions depend on each other, so each computer's calculations depend on each other

Because the Internet is out in the open, security is a major issue and concerns both the organization using distributed computing and the user whose computer is doing the work. The organization needs to be able to trust the data results that the user's computer provides; several problems in the past include the "tweaking" of the software to report a faster processing speed to malicious data with fake alerts, which needed to be recalculated again. The user must also be able to trust the organization — how will the user know whether the organization is *really* processing, say, prime numbers? — and the organization's software — how will the user know whether the software isn't digging through his files?

When dealing with a local distributed network — where the computers are within a contained area, such as a room — the cost of maintenance can skyrocket because each computer has its own share of problems and errors. Unless these computers are used for something else, say, as workstations at a library, this type of distributed computing isn't cost-effective.

Alternatives and Forms of DC

Distributed computing also comes in different forms, other than use of the concept on the Internet. Localized non-Internet forms include **terminal systems**, which have no processor of its own and depends on a central processing server to make it work, and its cousins, the time-sharing system, and thin-client computing. Many libraries employ these efficient yet inexpensive record-keeping terminals, since they are made of essentially one computer hooked to many monitors and keyboards.

Local distributed computing is used in universities, hospitals, and general offices. Computer workstations are hooked up to a local-area network (LAN) and can share files with each other and print to one or many printers. Hospitals use distributed computing systems to keep track of patients. All the computers are able to function independently but can share data, which is especially important in keeping databases synchronized.

Parallel-processing is a different kind of distributed computing where Parallel-processing machines have one computer with many processors, instead of many computers with many processors. It has a cutting edge in speed over its distributed computing cousin, because the processors are able to "talk" with each

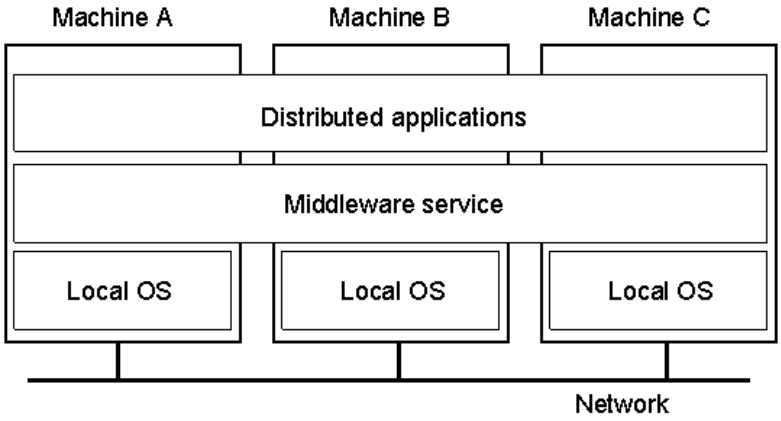
other. It can also compute data with variables that depend on each other, such as the previously-mentioned *n*-body simulation. However, software that makes use of all of these are complex and difficult to program. Ten processors are harder to control than just one. A teacher can control one child more easily than ten children; similarly, ten processors are harder to control than just one.

Clustering is in between parallel supercomputers and distributed computing systems. Several computers are placed together in relatively close proximity and connected to each other via interconnections called nodes, where they can pass data to each other at high speeds. Like distributed computing, it spreads the work across more than one computer. Like parallel processing, the computers can communicate with each other very fast, but still face the same difficulty and complexity in programming the software.

Challenges in Distributing Applications

- **Application Distribution**
  - Partitioning: For distributing an application there is a need to divide the application into small units of distribution. These units are then distributed separately. This partitioning is to be done as per needed because big partitions may cause the separate node of processing to be heavily loaded and may cause decrease in overall performance. On the other hand it may also be possible that these partitions are made so small and unneeded that it takes more time to distribute and communicate the problem than to execute it.
  - Configuration: When we divide the problem into partitions then we must also define the relationships between these partitions. These associations must be designed in a way that the results of these partitions could be combined together to form a complete result.
  - Allocation: Once the Problem is distributed the modules must be bind together in a uniform way in order to server our purpose
- **Heterogeneity**

  Heterogeneity is the concept that the different standards of networks, operating systems, software, hardware or programming languages problems can be raised. To overcome these differences we develop some standards that regulate the interpretability of these differences. One Major technique used to overcome this problem is the divide and conquer rule. We divide the system into horizontal layers, of Operating System, Middle Ware and Application. The Middle layer hides the differences in environment to both upper and lower layer.



  Examples of this layer are RMI, CORBA, DCOM and RPC.
- **Openness:** Openness ensures the extensibility and maintainability of distributed applications. The distributed applications need to be easier to extend and maintain. Lack of these characteristics can make an application useless. Openness removes the possibility of re-

implementation. Openness can be increased by Specification, documentation, published interfaces and standardization.

- **Security:** The most important and crucial challenge to the distributed systems is the security. Communication between multiple nodes needs to be secure enough to avoid corruption and illegal access of the data. The security addresses the following three concepts
    - Confidentiality: Protection Against disclosure to unauthorized individuals
    - Integrity: Protection against alteration or corruption
    - Availability: Protection against interference with mean to access the resources.

  Authentication and cryptography can make the situation better.

- **Scalability**: A distributed system is scalable if it remains effective as the number of users and/or resource increases. Following challenges are faced in scaling application
    - Controlling resource costs
    - Controlling performance loss
    - Preventing resources from running out
    - Avoiding performance bottlenecks
- **Failure handling/ Reliability:** In Distributed applications the failures are more common than centralized applications. This issue is also vital. Failure handling includes
    - Detection of Failures
    - Masking Failures
    - Tolerance
    - Recovery from failure
- **Concurrency:** Concurrency includes handling several simultaneous requests for a resource. We may use Synchronization techniques like semaphores to manage concurrency and deadlock avoidance.
- **Transparency:** Concealing the heterogeneous and distributed nature of the system so that it appears to the user like one system

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be shared by several competitive users
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk