Computer Architecture and Organization Digital Assignment 2

Name: Om Ashish Mishra

Registration Number: 16BCE0789

Slot: B2+TB2

Distributed Computing

Introduction

The word *distributed* in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area. Nowadays it's used in various places, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing.

The properties are commonly used:

- There are several autonomous computational entities (*computers* or *nodes*), each of which has its own local memory.
- The entities communicate with each other by message passing.

A distributed system may have a common goal, such as solving a large computational problem; the user then perceives the collection of autonomous processors as a unit. Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users.

Other typical properties of distributed systems include the following:

- The system has to *tolerate failures* in individual computers.
- The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.

- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.

Distributed computing is a field of computer science that studies distributed systems. A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. Three significant characteristics of distributed systems are: *concurrency of components*, *lack of a global clock*, and *independent failure of components*. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs. There are many alternatives for the message passing mechanism, including pure HTTP, RPC-like connectors and message queues.

A goal and challenge pursued by some computer scientists and practitioners in distributed systems is location transparency; however, this goal has fallen out of favour in industry, as distributed systems are different from conventional non-distributed systems, and the differences, such as network partitions, partial system failures, and partial upgrades, cannot simply be "papered over" by attempts at "transparency".

Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate with each other by message passing.

Parallel and distributed computing

Distributed systems are groups of networked computers, which have the same goal for their work. The terms "concurrent computing", "parallel computing", and "distributed computing" have a lot of overlap, and no clear distinction exists between them. The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel. Parallel computing may be seen as a particular tightly coupled form of distributed computing, and distributed computing may be seen as a loosely coupled form of parallel computing. Nevertheless, it is possible to roughly classify concurrent systems as "parallel" or "distributed" using the following criteria:

- In parallel computing, all processors may have access to a shared memory to exchange information between processors.
- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.

The figures illustrate the difference between distributed and parallel systems. Figure (a) is a schematic view of a typical distributed system; the system is represented as a network topology in which each node is a computer and each line connecting the nodes is a communication link. Figure (b) shows the same distributed system in more detail: each computer has its own local memory, and information can be exchanged only by passing messages from one node to another by using the available communication links. Figure (c) shows a parallel system in which each processor has a direct access to a shared memory.

The situation is further complicated by the traditional uses of the terms parallel and distributed *algorithm* that do not quite match the above definitions of parallel and distributed *systems*. Nevertheless, as a rule of thumb, high-performance parallel computation in a shared-memory multiprocessor uses parallel algorithms while the coordination of a large-scale distributed system uses distributed algorithms.

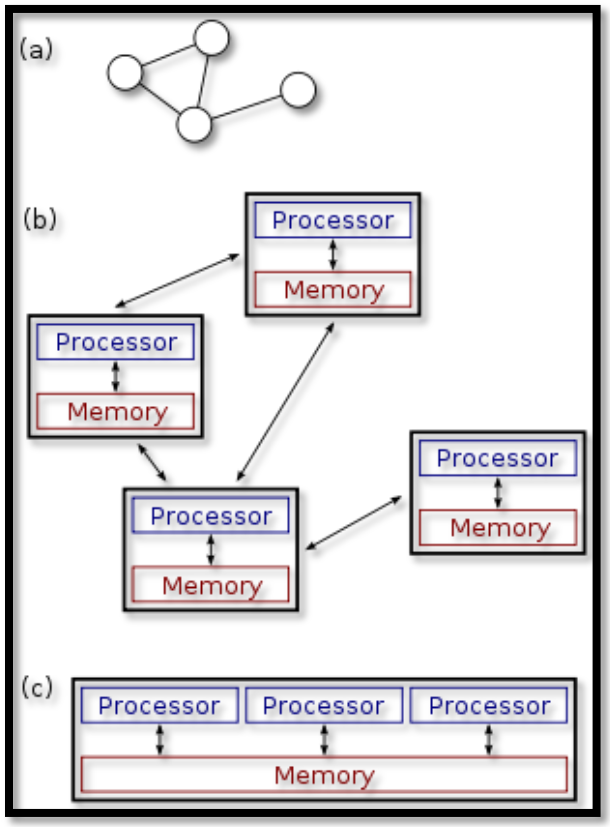

Fig: (a), (b): a distributed system and (c): a parallel system.

Architectures

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect multiple CPUs with some sort of network, regardless of whether that network is printed onto a circuit board or made up of loosely coupled devices and cables. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system.

Distributed programming typically falls into one of several basic architectures: **client–server, three-tier, n-tier, or peer-to-peer**:

- **Client–server**: architectures where smart clients contact the server for data then format and display it to the users. Input at the client is committed back to the server when it represents a permanent change.
- **Three-tier**: architectures that move the client intelligence to middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are three-tier.
- **n-tier**: architectures that refer typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.
- **Peer-to-peer**: architectures where there are no special machines that provide a service or manage the network resources. Instead all responsibilities are uniformly divided among all machines, known as peers. Peers can serve both as clients and as servers.

Another basic aspect of distributed computing architecture is the method of communicating and coordinating work among concurrent processes. Through various message passing protocols, processes may communicate directly with one another, typically in a master/slave relationship. Alternatively, a "database-centric" architecture can enable distributed computing to be done without any form of direct inter-process communication, by utilizing a shared database.

Examples

Examples of distributed systems and applications of distributed computing include the following:

- telecommunication networks:
  - telephone networks and cellular networks,
  - computer networks such as the Internet,

- wireless sensor networks,
- routing algorithms;

➢ network applications:

- World Wide Web and peer-to-peer networks,
- massively multiplayer online games and virtual reality communities,
- distributed databases and distributed database management systems,
- network file systems,
- distributed information processing systems such as banking systems and airline reservation systems;

➢ real-time process control:

- aircraft control systems,
- industrial control systems;

➢ parallel computation:

- scientific computing, including cluster computing and grid computing and various volunteer computing projects,
- distributed rendering in computer graphics

Applications

Reasons for using distributed systems and distributed computing may include:

1. The very nature of an application may *require* the use of a communication network that connects several computers: for example, data produced in one physical location and required in another location.
2. There are many cases in which the use of a single computer would be possible in principle, but the use of a distributed system is *beneficial* for practical reasons. For example, it may be more cost-efficient to obtain the desired level of performance by using a cluster of several low-end computers, in comparison with a single high-end computer. A distributed system can provide more reliability than a non-distributed system, as there is no single point of failure. Moreover, a distributed system may be easier to expand and manage than a monolithic uni processor system.

----------XXXX----------