Lecture 18: Parallel and Distributed Systems

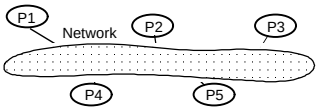
- Classification of parallel/distributed architectures
- SMPs
- Distributed systems
- Clusters

What is a distributed system?

- From various textbooks:
  - "A distributed system is a collection of independent computers that appear to the users of the system as a single computer."
  - "A distributed system consists of a collection of autonomous computers linked to a computer network and equipped with distributed system software."
  - "A distributed system is a collection of processors that do not share memory or a clock."
  - "Distributed systems is a term used to define a wide range of computer systems from a weakly-coupled system such as wide area networks, to very strongly coupled systems such as multiprocessor systems."

What is a distributed system?(cont.)

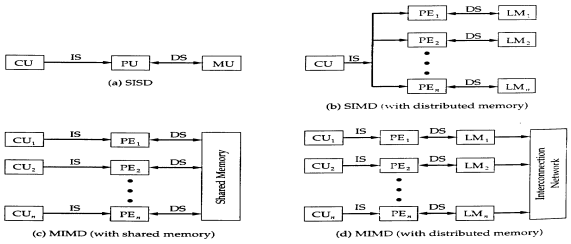
- A *distributed system* is a set of physically separate processors connected by one or more communication links



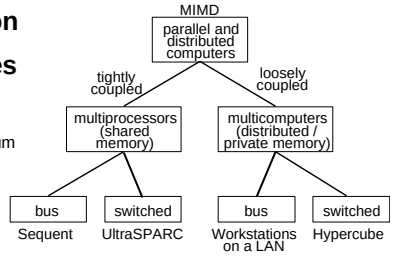
- Is every system with >2 computers a distributed system??
  - Email, ftp, telnet, world-wide-web
  - Network printer access, network file access, network file backup
  - We don't usually consider these to be distributed systems…

What is a distributed system?(cont.)

- Michael Flynn (1966)
  - SISD — single instruction, single data
  - SIMD — single instruction, multiple data
  - MISD — multiple instruction, single data
  - MIMD — multiple instruction, multiple data
- More recent (Stallings, 1993)



Classification of MIMD Architectures



Tanenbaum (date?)

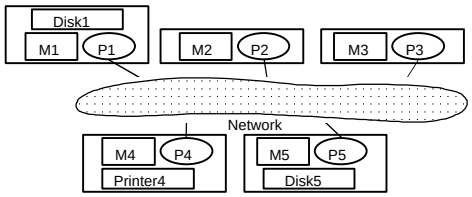
- Tightly coupled ≈ *parallel processing*
  - Processors share clock and memory, run one OS, communicate frequently
- Loosely coupled ≈ *distributed computing*
  - Each processor has its own memory, runs its own OS (?), communicates infrequently

Classification of the OS

- Multiprocessor Operating System
  - Tightly-coupled software (single OS) running on tightly-coupled hardware
  - A process can run on any processor
    - Single ready queue!
  - All memory is shared
  - File system similar to that on non-distributed systems
- Network Operating System
  - Loosely-coupled hardware
  - Loosely-coupled software
    - Each computer runs its own OS
    - User knows which machine he/she is on
  - Goal: share resources, provide global (network) file system
  - Typical utility programs: rlogin, rcp, telnet, ftp

Classification of the OS (cont.)

- "True" Distributed Operating System
  - Loosely-coupled hardware
    - No shared memory, but provides the "feel" of a single memory
  - Tightly-coupled software
    - One single OS, or at least the feel of one
  - Machines are somewhat, but not completely, autonomous



What's wrong with big machines?

- Symmetric multiprocessors (SMP) - multiple processors sharing memory such that the access from each processor to each memory location takes the same amount of time (*uniform* access)
- advantages:
  - single - processor applications can run without modifications (OS has to be modified. Why? How?)
  - inter-processor communication is fast (programs with fine-grain parallelism run fast)
- problems:
  - no fault-tolerance - one processor (or cache) fails, the whole system fails
  - does not scale (or gets inordinately expensive as it does)
    - hard to provide uniform memory access
    - processor speed are increasing faster than memory speed

Why (not) use distributed systems?

- Advantages:
  - Price / performance - network of workstations provides more MIPS for less $ than a mainframe does
  - Higher performance - *n* processors potentially gives *n* times the computational power
  - Resource sharing - expensive (scarce) resources need not be replicated for each processor
  - Scalability - modular structure makes it easier to add or replace processors and resources
  - Reliability Replication of processors and resources yields fault tolerance
- Problems:
  - requires paradigm shift for applications programmer - message passing rather than shared memory
  - communications between processors are slow - fine-grain parallelism is slow
  - have to deal with (administer, upgrade, maintain) multiple machines rather than one

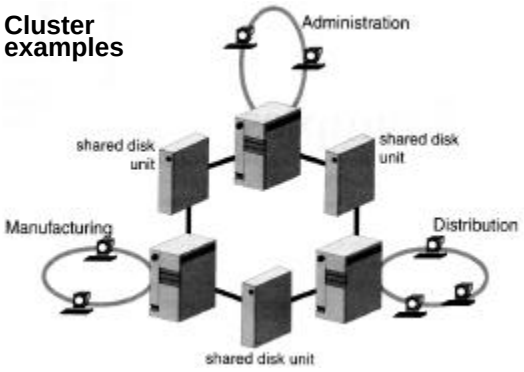
Clusters

- A subclass of distributed systems
- a small scale (mostly) homogeneous (the same hardware and OS) array of computers (located usually in one site) dedicated to small number of well defined tasks in solving of which the cluster acts as one single whole.
- typical tasks for "classic" distributed systems:
  - file services from/to distributed machines over (college) campus
  - distributing workload to all machine on campus
- typical tasks for a cluster:
  - high-availability web-service/file service, other high-availability applications
  - computing "farms".

Clusters (C) vs. Distributed systems (D)

- structure
  - [C] - homogeneous - purchased to perform a certain task
  - [D] - heterogeneous - put together from the available hardware
- scale
  - [C] - small scale - don't have to make sure that the setup scales
  - [D] - medium/large - have to span (potentially) large number of machines
- task
  - [C] - specialized - made to perform a small set of well-defined tasks
  - [D] - general - usually have to be general-user computing environments
- price
  - [C] - (relatively) cheap
  - [D] - free(?)/expensive
- reliability
  - [C] - as good as it needs to be
  - [D] - high/low?
- security
  - [C] - nodes trust each-other
  - [D] - nodes do not trust each other

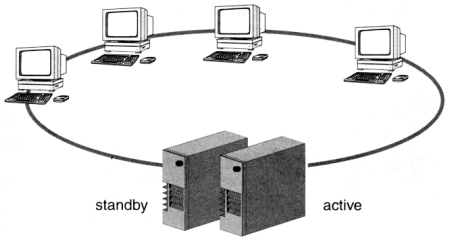
Cluster examples



pictures taken from "In Search of Clusters", G.F. Pfister, 1998

- branches get access to shared information even if one of the links or computers fails

Cluster examples (cont.)



- active machine - serves files to the network of computers
- standby machine -listens to network and updates it's own copy of files
- in case of machine failure - standby machine takes over file service *transparent* to users

Cluster examples (cont.)

- dispatcher machine - sends the web requests to server machines and makes sure that the servers are evenly loaded
- web service continues even if a server fails