Loosely Coupled Distributed Systems
- Users are aware of multiplicity of machines. Access to resources of various machines is done explicitly by:
  - Remote logging into the appropriate remote machine.
  - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism.

Tightly Coupled Distributed Systems
- Users not aware of multiplicity of machines. Access to remote resources similar to access to local resources
- Examples
  - Data Migration - transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task.
  - Computation Migration - transfer the computation, rather than the data, across the system.

Distributed-Operating Systems (Cont.)
- Process Migration - execute an entire process, or parts of it, at different sites.
  - Load balancing - distribute processes across network to even the workload.
  - Computation speedup - subprocesses can run concurrently on different sites.
  - Hardware preference - process execution may require specialized processor.
  - Software preference - required software may be available at only a particular site.
  - Data access - run process remotely, rather than transfer all data locally.

Why Distributed Systems?
- Communication
  - Dealt with this when we talked about networks
- Resource sharing
- Computational speedup
- Reliability
Resource Sharing

- Distributed Systems offer access to specialized resources of many systems
  - Example:
    - Some nodes may have special databases
    - Some nodes may have access to special hardware devices (e.g., tape drives, printers, etc.)
- DS offers benefits of locating processing near data or sharing special devices

OS Support for resource sharing

- Resource Management?
  - Distributed OS can manage diverse resources of nodes in system
  - Make resources visible on all nodes
    - Like VM, can provide functional illusion but rarely hide the performance cost
- Scheduling?
  - Distributed OS could schedule processes to run near the needed resources
  - If need to access data in a large database may be easier to ship code there and results back than to request data be shipped to code

Design Issues

- Transparency - the distributed system should appear as a conventional, centralized system to the user.
- Fault tolerance - the distributed system should continue to function in the face of failure.
- Scalability - as demands increase, the system should easily accept the addition of new resources to accommodate the increased demand.
- Clusters vs Client/Server
  - Clusters: a collection of semi-autonomous machines that acts as a single system.

Why Distributed Systems?

- Resource sharing
- Computational speedup
- Reliability

Computation Speedup

- Some tasks too large for even the fastest single computer
  - Real time weather/climate modeling, human genome project, fluid turbulence modeling, ocean circulation modeling, etc.
- What to do?
  - Leave the problem unsolved?
  - Engineer a bigger/faster computer?
  -Harness resources of many smaller (commodity?) machines in a distributed system?

Breaking up the problems

- To harness computational speedup must first break up the big problem into many smaller problems
- More art than science?
  - Sometimes break up by function
    - Pipeline?
  - Job queue?
  - Sometimes break up by data
    - Each node responsible for portion of data set?
Decomposition Examples
- Decrypting a message
  - Easily parallelizable, give each node a set of keys to try
  - Job queue - when tried all your keys go back for more?
- Modeling ocean circulation
  - Give each node a portion of the ocean to model (N square ft region?)
  - Model flows within region locally
  - Communicate with nodes managing neighboring regions to model flows into other regions

Decomposition Examples (con’t)
- Barnes Hut – calculating effect of bodies in space on each other
  - Could divide space into NxN regions?
  - Some regions have many more bodies
- Instead divide up so have roughly same number of bodies
  - Within a region, bodies have lots of effect on each other (close together)
  - Abstract other regions as a single body to minimize communication

Linear Speedup
- Linear speedup is often the goal.
  - Allocate N nodes to the job goes N times as fast
- Once you've broken up the problem into N pieces, can you expect it to go N times as fast?
  - Are the pieces equal?
  - Is there a piece of the work that cannot be broken up (inherently sequential?)
  - Synchronization and communication overhead between pieces?

Super-linear Speedup
- Sometimes can actually do better than linear speedup!
- Especially if divide up a big data set so that the piece needed at each node fits into main memory on that machine
- Savings from avoiding disk I/O can outweigh the communication/synchronization costs
- When split up a problem, tension between duplicating processing at all nodes for reliability and simplicity and allowing nodes to specialize

OS Support for Parallel Jobs
- Process Management?
  - OS could manage all pieces of a parallel job as one unit
  - Allow all pieces to be created, managed, destroyed at a single command line
  - Fork (process,machine)?
- Scheduling?
  - Programmer could specify where pieces should run and OS could decide
    - Process Migration? Load Balancing?
  - Try to schedule piece together so can communicate effectively

OS Support for Parallel Jobs (con’t)
- Group Communication?
  - OS could provide facilities for pieces of a single job to communicate easily
  - Location independent addressing?
  - Shared memory?
  - Distributed file system?
- Synchronization?
  - Support for mutually exclusive access to data across multiple machines
  - Can’t rely on HW atomic operations any more
  - Deadlock management?
  - We’ll talk about clock synchronization and two phase commit later
Why Distributed Systems?
- Resource sharing
- Computational speedup
- Reliability

Reliability
- Distributed system offers potential for increased reliability
  - If one part of system fails, rest could take over
  - Redundancy, fail-over
- BUT! Often reality is that distributed systems offer less reliability
  - "A distributed system is one in which some machine I've never heard of fails and I can't do work!"
  - Hard to get rid of all hidden dependencies
  - No clean failure model
    - Nodes don't just fail, they can continue in a broken state
    - Partition network = many many nodes fail at once!
    - (Determine who you can still talk to; Are you cut off or are they?)
- Network goes down and up and down again!

Robustness
- Detect and recover from site failure, function transfer, reintegrate failed site
  - Failure detection
  - Reconfiguration

Failure Detection
- Detecting hardware failure is difficult.
- To detect a link failure, a handshaking protocol can be used.
- Assume Site A and Site B have established a link.
  - At fixed intervals, each site will exchange an I-am-up message indicating that they are up and running.
  - If Site A does not receive a message within the fixed interval, it assumes either (a) the other site is not up or (b) the message was lost.
  - Site A can now send an Are-you-up? message to Site B.
- If Site A does not receive a reply, it can repeat the message or try an alternate route to Site B.

Failure Detection (cont)
- If Site A does not ultimately receive a reply from Site B, it concludes some type of failure has occurred.
  - Types of failures:
    - Site B is down
    - The direct link between A and B is down
    - The alternate link from A to B is down
    - The message has been lost
  - However, Site A cannot determine exactly why the failure has occurred.
  - B may be assuming A is down at the same time
  - Can either assume it can make decisions alone?

Reconfiguration
- When Site A determines a failure has occurred, it must reconfigure the system:
  1. If the link from A to B has failed, this must be broadcast to every site in the system.
  2. If a site has failed, every other site must also be notified indicating that the services offered by the failed site are no longer available.
- When the link or the site becomes available again, this information must again be broadcast to all other sites.