Processes

- Threads
- Client-Server Architectures
- Code Migration

Threads

- A Process has a given memory space.
  - Different processes have different memory space.
- Thread - lightweight process
- In a multi-threaded process many threads shares the same memory space.
  - Concurrent programming
- In a distributed system when one process calls another (e.g. a client calls a server) its thread will transfer (logically) to that other process (and another memory space).
  A server “borrows” the clients thread.
  - Distributed **and** concurrent programming

Use of Threads in Distributed Applications

- Threads in Clients
  - Concurrent programming can be used to update a web page gradually to make a more pleasant user experience.
- Threads in Servers
  - Concurrent programming can be used to allow many clients at the same time sharing the server and its resources.
  - here it must be assured that there is no data corruption by use of mutual exclusion.
    This instead might cause deadlock or starvation.

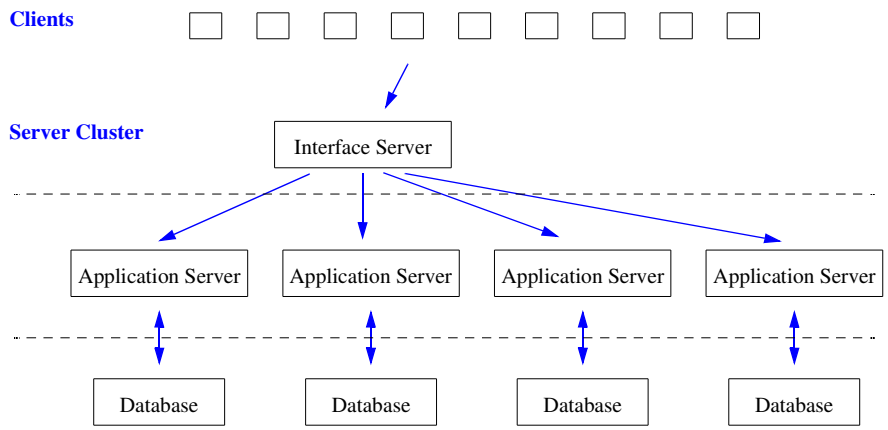
Client-Server Architectures

- Servers can be
  - Stateful:
    Information about the Client state is maintained by the Server.
    - Might be useful if there is multiple messages sent to the server.
    - The information in messages can be reduced.
    - Security might be easier to implement.
    - But can lead to complex protocols and error handling.
  - Stateless:
    Information about the Client state is not maintained by the Server.
    - The normal solution if there is only service requests that are independent of each other and only use a single message each.
    - Simple protocols and error handling must be performed by the client.
    - If it is used for multiple dependant requests the server must rely on the client that is doing right.
    - Security problem. e.g. NFS protocol.

Virtual Machines

- Platforms “independent” of the underlying operating system.
- For Clients:
  - Web browser.
  - Java (machine)
- For Servers:
  - Java machine
- For general distributed system:
  - Java machine
  - Custom made virtual layer

Server Clusters



Code Migration

- To move processes to other computers.
- Can be decided
  - before starting a process
  - while a process is running
- Mainly used for performance issues: Load Balancing
- Can also be used for enhancing availability:
  - If a computer is breaking down, or if it has to be upgraded or given service
    - move its ongoing processes to another computer.
    - *Tandem* systems since 1970ies

Different types of Code Migration (1)

- Weak Mobility
  move before starting a process. Only transfer the code segment.
  - Sender-initiated mobility
    e.g. load balancing
    - Execute at target process
    - Execute in separate process
  - Receiver-initiated mobility
    - Execute at target process
      e.g. java applets
    - Execute in separate process

Different types of Code Migration (2)

- Strong Mobility
  move running process. Must transfer the programs environment (resource segment and execution segment as well as the code segment.
  - Sender-initiated mobility
    - Migrate process
    - Clone process
  - Receiver-initiated mobility
    - Migrate process
    - Clone process

Heterogeneous Systems

- Processes migration between different type of systems/hardware
- The code might have to be compiled differently
- Example for Heterogeneous Hardware (for program *cat* in Linux):
  - in a single system the binary will be in the */bin* directory
  - for a heterogeneous system:
    hidden directories

    /bin/cat ⇒

    /bin/cat/intel
    - binary for Intel processor

    /bin/cat/motorola
    - binary for Motorola processor

    ...
  - One program has a directory with different compiled code for different processors.
  - The system can choose the appropriate code when it has chosen which processor to use.