Scheduling of Jobs / Processes in a Distributed System

- Scheduling in a centralized system:
  - Resource = CPU
  - Consumer = process
  - Scheduling = assign each process to some period of time on the CPU
- Scheduling in a distributed system:
  - Resource = processor / workstation
  - Consumer = computation task / process
  - Scheduling = assign each process to some processor
- Goal: distribute processes among the processors so as to optimize some cost function (e.g., response time, utilization)
  - *Load distribution* — which tasks should be moved, and when?
  - *Process migration* — how to move them

Example Load Distribution Algorithm

- All processors constantly monitor their load — the number of active processes
- When a processor's load goes above some particular threshold, it becomes a "sender"
- The new process that caused it to become a sender is selected for transfer
- The sender polls the other processes, one by one, until it finds a "receiver" — a process with a load below some particular threshold
- The selected process is frozen, transferred (migrated) from the sender to the receiver, and restarted there

Measuring Load

- Number of processes, resource demands on those processes, instruction mixes, architecture and speed of processor
  - But some are swapped out, dead, etc.
  - Remaining service time is unknown
- Length of CPU's ready or I/O queues
  - Correlates well with response time
    - Low overhead, used extensively
  - Don't want to simply measure queue length while a task transfer is in progress
    - Artificially increase queue length whenever processor accepts transfer of a remote task, timeout to avoid network / crash problems
  - Unfortunately, queue length doesn't really correlate with CPU utilization, particularly in an interactive environment
    - Use a background process to monitor CPU utilization (but… this is expensive!)

Advantages of Load Distribution

- Reduce response time for processes
  - Move to lightly loaded node
- Speed up individual jobs
  - Go to faster node
  - Split up process across multiple nodes
- Gain higher throughput
  - Balance system load
  - Mix I/O & CPU bound processes
- Utilize resources effectively
  - Move to node where resources reside
- Reduce network traffic
  - Cluster related processes on same node

Desirable Features of a Good Load Distribution Method

- No *a priori* knowledge about processes
- Dynamic in nature — change with system load, allow process migration
- Quick decision-making capability
- Balanced system performance and overhead — don't reduce system performance collecting state information
- Stability — don't migrate processes so often that no work gets done (better definition later)
- Scalability — works on both small and large networks
- Fault tolerance — recover if one or more processors crashes

Load Distribution vs. Process Migration

- *Load distribution* — deciding which tasks to move from one processor to another, and when to move them
  - Selection of process to migrate
  - Selection of destination node
- *Process migration* is the relocation of a process from its current location (*source node*) to another node (*destination node*)
  - Preemptive — after process starts
  - Non-preemptive — before process starts
  - Mechanics of process migration:
    - Freeze process on source node
    - Transfer address space and state of process from source to destination node
    - Restart process on destination node
    - Forward messages sent to old processor to new processor

Desirable Features of a Good Process Migration Method

- Transparency
  - Access to all objects from everywhere
  - Location-independent system calls
- Minimal interference
  - Minimize freeze time (stopped execution while process is being transferred)
- Minimal residual dependencies
  - Migrated process should not depend in any way on source node, otherwise:
    - Adds to load on source node
    - Failure of source node could affect it
- Efficiency
  - Keep inefficiency to a minimum
    - Time to select process and destination
    - Time required to migrate a process
    - Cost of remote execution afterwards

Process Migration Mechanisms

- Freezing and restarting a process
  - Freezing = execution suspended, external interactions with process are deferred
  - Only an issue for preemptive transfers
  - Before freezing, process must be blocked
    - Blocked immediately
      - If not executing a system call
      - If executing a system call, but sleeping as it waits for a kernel event to occur
    - Blocking is delayed
      - If executing a system call, but sleeping at a non-interruptable priority — must delay until system call is complete
  - After blocking, wait for completion of fast I/O operations (e.g.,disk I/O), but not for slow I/O (e.g., keyboard)
  - Keep track of open files, switch to local copiers of the files if possible
  - Keep same process ID after migration

Process Migration Mechanisms (cont.)

- Transferring the address space & state
  - Entire process state: register contents, scheduling info, memory tables, I/O states, process ID, file info, etc.
    - Must stop execution during transfer
  - Address space: code, data, stack, heap
    - Transfer can take a long time!
    - Can continue execution during transfer
  - 3 alternatives in transfer:
    - Total freeze
      - Stop execution during addr. space transfer
      - Possible long suspension in execution
    - Pre-transfer
      - Continue execution during address space transfer, then freeze process and transfer remaining modified pages
      - Small freeze time = little interruption
    - Transfer on reference
      - Leave address space on source node, only transfer pages when and if they are referenced

Process Migration Mechanisms (cont.)

- Message-forwarding
  - 3 types of messages to forward
    1. Messages received at source node after execution has stopped there, but before execution has started on destination
    2. Messages received at source node after execution has started on destination
    3. Messages sent to process later
  - Resending the message
    - Return or drop type 1 & 2 messages, hope sender will resend to new location
    - Sender can do a "locate" operation to find process at its new location
  - Origin site mechanism
    - Messages are sent to original source site, which forwards them as necessary
  - Link traversal mechanism
    - Type 1 messages are part of migration
    - Type 2 & 3 messages follow a link (forwarding address) left behind

Process Migration in Heterogeneous Systems

- Must translate data
  - Big endian, little endian (bytes & words)
  - Unicode, ASCII, EBCDIC, etc.
  - External data representation
    - Use standard representation for transfer
  - Sinha describes various techniques for migrating the exponent and mantissa of floating point numbers
    - However, many systems now use the IEEE floating point format, for consistency
    - Single precision = 32 bits (1 sign, 8 exponent, 23 mantissa)
    - Double precision = 64 bits (1 sign, 10 exponent, 53 mantissa)
  - Also have to handle signed-infinity and signed-zero, if those values are supported by one or both of the nodes

Classifying Load Distribution Algorithms (Preview)

- How is the load redistributed?
  - Reduce the chance of having one processor is idle, but tasks contending for service at another processor, by transferring tasks to between processors
  - Load balancing
    - Tries to equalize the load at ***all*** processors
    - Moves tasks more often than load sharing; much more overhead
  - Load sharing
    - Tries to reduce the load on the heavily loaded processors only
    - Probably a better solution
- How is system state (load on each processor) used?
- Can a task be transferred to another processor once it starts executing?