Ch 11

- Distributed Scheduling
  - Resource management component of a system which moves jobs around the processors to balance load and maximize overall performance.
  - Typically makes sense in LAN level distributed systems due to latency concerns.
  - Needed because of uneven distribution of tasks on individual processors
    - Can be due to several reasons.
    - Can even make sense for homogeneous systems with (on average) even loads.

- How does one characterize
  - Performance : average response time
  - Load:
    - It has been shown that queue lengths for resources (e.g. CPUs) can be a good indicator.
    - How does one handle the delay of transfer when systems are unevenly loaded and we seek to rectify that ?
      - Timeouts, holddowns
    - Queue length not very appropriate for (nor correlated with) CPU utilization for some tasks (e.g. interactive).

- Load balancing approaches may be
  - Static: Decisions are “hard wired” a-priori into the system based on designers understanding.
  - Dynamic: Maintain state information for the system and make decisions based on them. Better than static, but have more overhead.
  - Adaptive: A subtype of dynamic, they can change the parameters they analyze based on system load.
- Load balancing vs. Load sharing
  - Balancing typically involves more transfers. However, sharing algorithms that transfer in anticipation can also cause more transfers.

- Transfers may be preemptive or non-preemptive
  - Preemptive transfers involve transferring execution state as well as the task. Non-preemptive transfers are essentially “placements”
- Load Distribution System Components
  - Transfer policy: Which node should send, who should receive (threshold based approaches are common)
  - Selection policy: Which task should be moved (new tasks, location independent tasks, long running tasks …)
  - Location Policy: Finding a receiver for a task. Typical approaches are polling or broadcast.
  - Information Policy
    - Demand driven, Periodic, or State Change driven

- Stability in a load sharing system
  - Queuing Theoretic: When total work arrival (tasks + load sharing overhead) is greater than rate at which CPU can work. Alternatively, look at the effectiveness of the algorithm.
  - Algorithmic : Does the algorithm lead to thrashing ?

Sender Initiated LD Algorithms

- The overloaded node attempts to send tasks to lightly loaded node
  - Transfer Policy: If new Tasks takes you above threshold, become sender. If receiving task will not lead to crossing over threshold, then become receiver
  - Selection Policy: Newly arrived tasks
  - Location Policy
    - Random – still better than no sharing. Constrain by limiting the number of transfers
    - Threshold – chose nodes randomly but poll them before sending task. Limited no. of polls. If process fails execute locally.
    - Shortest – Poll all randomly selected nodes and transfer to least loaded. Doesn’t improve much over threshold.
  - Information Policy, Stability

Receiver initiated

- Load sharing process initiated by a lightly loaded node
  - Transfer Policy: Threshold based.
  - Selection Policy: Can be anything
  - Location Policy: Receiver selects upto N nodes and polls them, transferring task from the first sender. If none are found, wait for a predetermined time, check load and try again
  - Information Policy
  - Stability: At high loads, few polls needed since senders easy to find. At low loads, more polls but not a problem. However, transfers will tend to be preemptive.

Symmetric Algorithms

- Simple idea – combine the previous two. One works well at high loads, the other at low loads.
- Above Average Algorithm: Keep load within a range
  - Transfer Policy: maintain 2 thresholds equidistant from average. Nodes with load > upper are senders, Nodes with load < lower are receivers.
  - Location Policy: Sender Initiated:
    - Sender broadcasts “toohigh” message and sets up toohigh alarm
    - Receiver getting toohigh message replies with accept, cancels its toolow alarm, starts an awaitingtask alarm, and increments load value
    - Sender which gets accept message will transfer task as appropriate. If it gets toolow message, it responds with a toohigh to the sender.
    - If no accept has been received within timeout, send out changeaverage message.

- Location Policy: Receiver Initiated
  - A receiver broadcasts a toolow message and sets toolow alarm.
  - Upon receiving toohigh message, do as in sender initiated
  - If toolow alarm expires, send changeaverage message
- Selection Policy
- Information Policy is demand driven, and has low overhead. Each node can change the range individually.

Adaptive Algorithms

- Stable Symmetric Algorithm.
  - Use information gathered during polling to change behaviour. Start by assuming that everyone is a receiver.
  - Transfer Policy: Range based with Uppper and Lower Threshold
  - Location Policy: Sender Initiated component polls node at head of receiver list. Depending on answer, either a task is transferred or node moved to OK or sender list. Same thing happens at the receiving end. Receiver initiated component polls senders list in order, OK list and receivers list in reverse order. Nodes are moved in and out of lists at sender and receiver.
  - Selection Policy – any, Information Policy – Demand driven
  - At high loads, receiver lists get empty preventing future polling and "deactivating" sender component. At low loads, receiver initiated polling is deactivated, but not before updating receiver lists.
  - **Read Section 11.7, 11.8**

Requirements for load distribution

- Scalability
- Location Transparency
- Determinism
- Preemption
- Heterogeniety

Case Studies

- V System
  - Measure system parameters and broadcast. Each node caches “n best” nodes to migrate a job to. When it gets a new job, checks if it is in “ n best” else migrate to one of the others
- Sprite
  - Receivers notify coordinator of status. Sender selects jobs manually. Preemptive transfers to provide “console user” with best service.
- Task Migration Issues
  - Policy mechanism separation, state transfer and restarts, location transparency, performance issues.