Performance Evaluation

- $M/M/1$ is shortest job first no load sharing
- $M/M/9$ is shortest job first with load sharing and globally available information and no delay for migrating systems
- Which Algorithm is best is strongly determined by the network speed
- Starvation
- Infinite Substitution

Performance Evaluation

- $r = 1.0$
- The service time was exponential with a mean of 30 ms.
- ReqBC and RspBc were equal to 1000 bytes for all jobs
- Each job entered the system with 5.00 dollars
- $C = .25$ $\delta = 0.05$

Resource Allocation

- Sealed Bid Auction: no feedback to jobs.
- When a resource becomes idle, the processor "opens" the bids and picks the one with the highest unit price

Pricing Policy

- Processors send out ads giving the price for a millisecond of CPU time and the instruction rate
- Local Prices:
  1. If a resource becomes idle, the price is set to the unit price of the winning bid
  2. If there is no winning bid, the price is set to a predefine minimum
- If a resource is not idle, then price is set to the max of the unit price of the job currently using it and the best bid for the resource
- The processor sends an updated ad to all its neighbors whenever its CPU price changes

Solicit Bids

- New jobs are immediately given a chance to bid
- When some data in the bulletin board changes, all jobs update their budget sets and can reissue bids

Receive New Jobs

- Two types of jobs:
  1. Newly generated locally submitted jobs
  2. Migrants
- New jobs are added to the ready pool
- The ready pool also contains "poor" jobs that can't afford to bid for anything

Processors

1. Process Arrivals: Receive new jobs and place them in a ready pool
2. Solicit Bids : Jobs in the ready pool issue bids, others may retract their current bid and reissue
3. Update Prices: Based on the arriving price messages and local demand, update the bulletin board
4. Allocate Resources: Allocate idle resources based on bids

Bid Generation: Returning Jobs a special case

1. The job retraces the path it took to get to the current processor
2. The job bids the same amount per byte it paid to get to the processor
3. If this is not enough to put the link in the job's budget set then the job uses any surplus funds
4. If it is at the processor from which it entered the system, the job can leave

If there are a lot of hops, this may not be the optimal approach. Hopefully that will not be the case.

Bid Generation

- Job bids the price + a fraction (C_r) of the money it has left over

$$bid = p_k + (m_r - S_k) * C_r$$

- If the job's bid wins then $C_r = C_r(1 - \delta_r)$ otherwise $C_r = C_r(1 + \delta_r)$

- Service Time and Price - Take both service time and price into account

$$(k, S_k) \succ (j, s_j)$$

if

$$C_1 S_k + C_2 ST_k < C_1 S_j + C_2 ST_j$$

C_1 and C_2 are constants.

- Rational: Take both factors into account. The most like a real economy.
- Complexity: $O(A_i)$ where A_i is the number of elements. By using a heap, updates can be reduced to $O(LOG(A_i))$.

- Price - Job prefers the least expensive member of the budget set.

  Rational: Make sure you don't run out of money

- Service time - A job prefers the element of the budget set which will give it the fastest service:

$$\frac{\mu}{r} + ReqBC * d_{ik} + RspBc * d_{ik}$$

  Rational: Get the fastest service

Computing Preferences

- Borrowing from neoclassical economic theory each job has a Preference Relation
- $(S_j, j) \succ (S_k, k)$ means the job prefers service at j for price S_j to service at S_k
- Three preference relations:
  1. Price
  2. Service Time
  3. Price and Service Time

Service Cost

- estimated CPU service cost:

$$\frac{\mu}{BB[k].r} * BB[k].p$$

- Cost to migrate:

$$ReqBC * BB[k].x$$

- Cost to return:

$$Q_t + BB[k].x * RspBc$$

  Where is the amount it has previously set aside for returning.

- Note that the job assumes that the cost per byte will be the same to return from the processor

Computing the Budget set continued

- Budget Set is a set of order pairs (k, S_k) which describes where the job believes it can afford service
- k is the index of the processor
- S_k is the estimated cost

Computing the Budget Set

- Each bulletin board has three fields.
  1. p - The last reported price for a millisecond of CPU time
  2. x - The current rate per byte to migrate to the processor
  3. τ the processing rate

Jobs

1. Compute a Budget Set
2. Compute a Preference Relation on elements of the budget set
3. Generate a Bid for the most preferred element

Processors

- Pure Producers
- Sell resources to jobs
- Do not purchase anything
- Sell CPU time and communication bandwidth
- Independently set the prices
- Each processor can advertise on Bulletin Boards maintained by its neighbors

Jobs

- Pure Consumer
- Enter the system an with initial allocation of money
- No way to earn additional money
- Must purchase

$$\frac{\mu}{\tau_i}$$

  CPU seconds on some processor P_i
- Can migrate from one processor to another but must pay per byte to do so
- All CPU time must be on the same processor

The Economy

- Two types of Agents:
- Jobs: *Pure Consumers*
- Processors: *Pure Producers*

Load sharing algorithms

- Information Policy: How state information is transfered throughout the network
- Transfer Policy: When a job should migrate
- Location Policy: If migrations occur it defines the sources and targets
- Selection Policy: Once the source and targets are defined, it determines which jobs migrate

Jobs

Parameters associated with each jobs

- μ - The CPU service time of the job on a processor with $\tau = 1$
- ReqBC – The number of bytes needed to describe the job
- RspBc – The number of bytes needed to describe the result

Model assumptions

- Work performed is in Jobs
- The only resource needed is CPU time – no data access is required
- Communication bandwidth must be used for jobs to migrate processors

Model

- N processors, which are denoted $P_1, \ldots, P_N$
- Each processor P_i has an associated processing speed parameter τ_i
- Processors are connected by a point-to-point network defined by:

$$E = (i, j, d) : P_i, P_j \text{ are processors, } d > 0$$

- Bidirectional, full duplex communication line connecting processors P_i and P_j with a delay of d seconds

Advantages in this approach

- Complexity is limited
- Provides a unified framework for allocations all resources

Multiple processor Systems

- Dynamic environment
- Unpredictable
- Defy rigorous mathematical modeling and solutions

Microeconomic Models for Resource Sharing in a Multicomputer

UMSA Presentation

David Larochelle