Distributed Oblivious Load Balancing Using Prioritized Job Replication

Amir Nahir, Ariel Orda, Danny Raz

For more details:
Amir Nahir - nahira@cs.technion.ac.il

Technical Report:
http://webee.technion.ac.il/Sites/People/ArielOrda/Info/Other/NOR10.pdf

15/12/2010 Networking in the Cloud Era IBM Seminar
Distributed Oblivious Load Balancing Using Prioritized Job Replication

- Background and motivation
- Model
- Random Assignments
- Job replication with priorities
  - Exponential job lengths
  - Power-law job lengths
- Practical considerations:
  - Networking overhead & Buffer management
  - A hybrid scheme
- Improving a centralized scheme
Motivation

• A single data-center may contain over 10,000 servers
  [“The Datacenter as a Computer”, Luiz André Barroso and Urs Hölzle]

• Management solution requirements:
  - Distributed
  - Minimal overheads
Load Balancing in Large Data-Centers

- Servers distributed in a LAN (Data center)
- Clients distributed all over the network
- Quality measure: mean total time in the system
- Solution evaluation criteria:
  - Improvement in quality measure
  - Overheads
  - Scalability
Model

- N servers, infinite buffers
- Job arrivals: Poisson stream with Rate $N \cdot \lambda$
- Service time: Exponential distribution with mean service time $\mu$
- Stability requirement: $\lambda < \mu$
- No additional overhead: storage, networking time...
Basic Solution: Random Assignments

- A client randomly chooses which server to send the job to

- Expected time in the system \( \frac{1}{\mu - \lambda} \)
Job Replication with Priorities

- The client replicates its job and send two copies: high-priority (HP) and low-priority (LP)
- Servers policy:
  - HPs before LPs
  - Preemptive, non-resume (restart)
- Job completion is based on the first copy which completes
Server Behavior

while (TRUE)
    if (!hpQ.empty())
        job = hpQ.dequeue();
        signal(job.hpServer(), job);
        process(job);
    else
        if (!lpQ.empty())
            job = lpQ.dequeue();
            process(job);
            signal(job.hpServer(), job);

In addition, in case a high-priority job arrives while a low-priority job is being processed, execution is stopped, and the LP job is returned to the head of the queue.
Job Replication with Priorities
Analysis - 2D Markov Model

new job arrives at the low priority queue

new job arrives at the high priority queue
Analysis - 2D Markov Model

Low-priority job completion rate:
A low priority job can complete:
Locally (when the HP queue is empty) - with rate \( \mu \)
orRemotely - with rate \( \mu_1^R \)

High-priority job completion rate:
A high priority job can complete:
Locally - with rate \( \mu \)
orRemotely - with rate \( \mu_2^R \)
Analysis - Remote Completion Rates

- $\mu_2^R$ - the service rate of low priority jobs
  - The (remote) server processes low priority jobs at rate $\mu$ only when the high priority queue is empty and the low priority queue isn’t

  $$\mu_2^R = \mu \cdot \sum_{j=1}^{\infty} p_{0,j}$$

- $\mu_1^R$ - the service rate of high priority jobs
  - The (remote) server processes high priority jobs at rate $\mu$ when the high priority queue is not empty

  $$\mu_1^R = \mu \cdot \sum_{i=1}^{\infty} \sum_{j=0}^{\infty} p_{i,j}$$
Job Replication with Priorities - Performance Improvement

![Graph showing performance improvement with load.

Legend:
- Model, Buffer limit=50
- Simulation, Buffer Limit=50
- Simulation, Unlimited buffers]
Job Replication with Priorities - Performance Improvement: Explanation

![Graph showing performance improvement](image)

Legend:
- Blue line: Low priority job completion ratio
- Red dashed line: M/M/1 normalized queue length

Load vs. Performance Improvement

15/12/2010
Networking in the Cloud Era
IBM Seminar
Power-law Job Lengths

- Job lengths adhere to the following cdf:
  \[ P[X>x] \sim x^{-\alpha} \]
- In simulation, we used Bounded-Pareto
  - \( \alpha = \{0.2, 0.5, 0.8, 1.1, 1.4, 1.7\} \)
  - Maximal job length = \{1000, 10000, 25000\}
  - Minimal length adjusted to guarantee average is 1

15/12/2010 Networking in the Cloud Era IBM Seminar
Bounded-Pareto Job Lengths
while (TRUE)
    if (!hpQ.empty())
        job = hpQ.dequeue();
        signal(job.lpServer(), job);
        process(job);
    else
        if (!lpQ.empty())
            job = lpQ.dequeue();
            if (job.preemptions() == 10)
                drop(job);
                continue;
            process(job);
            signal(job.hpServer(), job);

In addition, in case a high-priority job arrives while a low-priority job is being processed, execution is stopped, and the LP job is returned to the head of the queue.
Bounded-Pareto Job Lengths: Simulation Results
Dropping All Overheads - the "Optimistic" Method

- No complex buffers
- No Signals
- The "optimistic" method:
  - If a low-priority job arrives at a busy server, it is discarded
  - If a low-priority job is preempted, it is discarded
The Optimistic Method - Performance Improvement
A Hybrid Scheme

- The usage of signaling introduces significant improvement at high loads
  - But adds communication overhead

- A Hybrid scheme:
  - The optimistic method is applied as long as the CPU load is low
  - Once the CPU load becomes high, signaling is used
  - Still - no buffers at all
The Hybrid Scheme - Simulation Results

![Graph showing performance improvement vs load for different schemes: Full scheme with signaling and unlimited buffers, Optimistic scheme, and Hybrid scheme. The graph indicates better performance for the hybrid scheme compared to the others.]
A Centralized Scheme

- In centralized system, a dispatcher controls the assignment of jobs to servers
- One commonly used algorithm is Round-Robin
- Augmenting a centralized dispatcher with low-priority replicas can further improve performance
Round-Robin with Low-Priority Replicas - Simulation Results

![Graph showing performance improvement with varying load]

Legend:
- **Blue**: Full scheme with signaling and unlimited buffers
- **Red**: Basic Round-Robin
- **Green**: Round-Robin augmented with low priority replicas
Oblivious Load Sharing: Summary

- Data-center sizes call for distributed management solutions
- Adding low-priority replicas can significantly reduce system service time
- Improvement can be traded off for scheme complexity
- Future work:
  - Exact model
  - Combining the replication technique with other queue management algorithms
  - Applying the replication technique to other domains
  - Multi-tier priorities
For more details

- Amir Nahir:
  - nahira@cs.technion.ac.il

- Technical Report:
  - http://webee.technion.ac.il/Sites/People/ArielOrda/Info/Other/NOR10.pdf
THANK YOU

15/12/2010

Networking in the Cloud Era
IBM Seminar