Case-study: Join-Idle-Queue Load Balancing Algorithm \[1\]

690PE Fall 2016
Bo Jiang

November 8, 2016

In Previous Lectures

• Scheduling for single server (lecture 15)
  – Resource Allocation Policy determines which job to serve
  – Priority policy

• Multiple servers: M/M/c queue (lecture 6)
Distributed Queues

• Each server has its own queue
• Jobs dispatched to servers upon arrival
An Extremely Bad Dispatcher

- Assign all jobs to server 1

- Mean response time \((1 - n\lambda)^{-1}\)
  - Need \(\lambda < n^{-1}\) for stability
How about Random?

• Assign jobs to random server

• Mean response time \((1 - \lambda)^{-1}\)
  – Stable for any \(\lambda < 1\)
Can we do better?

• Want to balance load across servers

• Greedy: Join-Shortest-Queue
  – “Near optimal” for M/G/n/PS \(^2\)
  – Dispatcher must know all queue states

\(^2\) Gupta et al. *Analysis of join-the-shortest-queue routing for web server farms*
Figure 7: Comparison of the first moment of queue length for JSQ, Least Work Left (LWL), Round Robin (R-R) and Random routing policies for $K=2$ and $K=8$ server farms with an anglo job-size distribution.

There are many interesting things to see in Figure 7. First, when observing that OPT-0 is significantly the best routing policy across all job-size distributions of those policies shown. Also JSQ is very close to OPT-0, within no more than 10%. This is surprising because JSQ utilizes only the number of jobs at each queue, whereas OPT-0 uses the remaining sizes of all jobs and the size of the incoming job.

From an insensitivity perspective, we see that there are some policies, e.g., OPT-0 and JSQ, that are nearly insensitive to the job-size distribution, whereas other policies, e.g., LWL and RR, are highly sensitive to the job-size distribution. It is an interesting question whether there is some detectable common characteristic among those routing policies that are nearly insensitive to the job-size distribution under PS server farms. This is an important question in light of the fact that the empirical workloads in Web server farms are very variable.

Turning to the question of optimality, note that the case of deterministic job sizes yields the lowest mean response times, as compared with other job-size distributions, and that all three policies: RR, LWL, and OPT-0, yield the same performance for the case of deterministic job sizes – in fact, they behave identically on every sample path when the job-size distribution is deterministic.

Conjecture 7.1 below hypothesizes that this value is the minimum response time possible across all policies.
Distributed Dispatchers

• More robust and flexible

• Drawbacks of JSQ
  – Large communication overhead
  – Communication time on critical path
Random again

• No communication overhead

• Behave like centralized dispatcher
• Mean response time \((1 - \lambda)^{-1}\)
Compromise: “Power of Two”

- Assign jobs to shorter of two random queues

\[ \text{Poisson}(n\lambda) \]

- Again behave like centralized dispatcher
Analysis for “Power of Two”

- Large system limit \((n \to \infty)\), \(\exp(1)\) service
- \(\pi_k\) : fraction of queues with \(\geq k\) jobs
- \(\mathbb{P}[\text{assign to queue with } \geq k \text{ jobs}] = \pi_k^2\)
- \(\mathbb{P}[\text{assign to queue with } k \text{ jobs}] = \pi_k^2 - \pi_{k+1}^2\)
- Consider set of queues with \(k\) jobs

\[
\begin{align*}
\pi_k - \pi_{k+1} &= \lambda \left(\pi_{k-1}^2 - \pi_k^2\right) \\
n \pi_k - \pi_{k+1} &= n \left(\pi_{k-1} - \pi_{k+1}\right)
\end{align*}
\]
Analysis of “Power of Two”

• For all $k \geq 1$,
  \[ \pi_k - \pi_{k+1} = \lambda(\pi_{k-1}^2 - \pi_k^2) \]

• For $j \geq 1$, sum over $k$ from $j$ to $\infty$
  \[ \pi_j = \lambda \pi_{j-1}^2 \]

• $\pi_0 = 1$
  $\pi_1 = \lambda$
  $\pi_2 = \lambda \pi_1^2 = \lambda^{1+2}$
  $\pi_3 = \lambda \pi_2^2 = \lambda^{1+2+2^2}$
  $\pi_k = \lambda^{1+2+2^2+\ldots+2^{k-1}} = \lambda^{2^k-1}$
Analysis of “Power of Two”

\( \pi_k \) : fraction of queues with \( \geq k \) jobs

- For Random,
  \[
  \pi_k = \lambda^k 
  \]

- For “Power of Two”,
  \[
  \pi_k = \lambda^{2^k-1} 
  \]
  
  – Shorter queue
  – Faster response
Generalization: “Power of d”

- SQ(d): join shortest of d random queues
- Fraction of queues with ≥ k jobs

\[ \pi_k = \lambda \frac{d^k - 1}{d - 1} \]

- Holds for general service time \(^3\)
- Small marginal gain for increasing d

\[^3\] Bramson et al. Randomized Load Balancing with General Service Time Distributions
Limitations of “Power of d”

• Performance gap between “power of d” and “join-the-shortest-queue” remains significant

• Requires communication between dispatchers and servers at the time of job assignment
  – On critical path, increase response time
What’s Efficient Load Balancing?

• Changes arrival rates to servers based on queue lengths
• Want to increase rates to less loaded servers
• In particular, increase rates to idle servers
  - Random: \( \lambda \)
  - SQ(d):
    \[
    \frac{n\lambda(1 - \pi_1^d)}{n(1 - \pi_1)} = \lambda \frac{1 - \lambda^d}{1 - \lambda} = \lambda + \lambda^2 + \cdots + \lambda^d
    \]
  - JSQ:
    \[
    \frac{n\lambda}{n(1 - \pi_1)} = \frac{\lambda}{1 - \lambda} = \lambda + \lambda^2 + \lambda^3 + \cdots
    \]

Reduce Communication Overhead

• Remove communication from critical path?
  – Collect info before job assignment
  – Keep info fresh: focus on idle servers

• Reduce amount of communication?
  – Let idle servers report to and queue at dispatchers
Join Idle Queue

- **I-queue**: list of subset of idle servers
- **Job arrives at dispatcher**
  - If I-queue nonempty, assign job to first idle server
  - Otherwise, assign job to random server
- **Primary Load Balancing**
  - Want random assignment to be rare
  - I-queue nonempty with high probability
Join Idle Queue

- Server completes all jobs
  - Joins exactly one I-queue
    - Avoids withdrawal
  - Which I-queue to join?
    - Random $\rightarrow$ JIQ-Random
    - SQ(d) $\rightarrow$ JIQ-SQ(d)

- Secondary Load Balancing
Busy server in I-queue?

• Possible
• Can be avoided if withdraw from I-queue upon receiving job
• Do not withdraw
  – Too much communication
  – Rare

Diagram:

- Dispatchers
- I-Queues
- Server Queues
- Servers
Idle server in multiple I-queue?

- Possible
- Can be avoided if
  - Dispatcher informs server of random assignment
  - Server checks if job comes from registered I-queue
Analysis in Large System Limit

- n servers
- m dispatchers
- Fix server-dispatcher ratio $r = \frac{n}{m}$
- Let $n, m \to \infty$
- Total arrivals: Poisson $(n \lambda)$
- Arrival at each dispatcher: Poisson $(r \lambda)$
- General service time with mean 1
Analysis of Second Load Balancing System

- **Theorem 1**: Proportion $\rho$ of occupied I-queues satisfies
  - for JIQ-Random
    $$\frac{\rho}{1 - \rho} = r(1 - \lambda)$$
  - for JIQ-SQ(d)
    $$\sum_{k=1}^{\infty} \rho \frac{dk - 1}{d - 1} = r(1 - \lambda)$$

- “Proof”: Show arrivals to I-queues are Poisson; compute average I-queue length in two different ways.
Analysis of Second Load Balancing System

• **Corollary 2**: Arrival rate to idle server with JIQ-Random is \((r+1)\) times that to busy server

• “Proof”:

– Arrival rate to idle server

\[
\frac{n\lambda \rho}{n(1 - \lambda)} + \frac{n\lambda(1 - \rho)}{n} = \lambda(1 - \rho)(1 + r)
\]

*Occupied I-Queues*  
*Empty I-Queues*

– Arrival rate to busy server

\[
\frac{n\lambda(1 - \rho)}{n} = \lambda(1 - \rho)
\]
Proportion of Empty I-Queues

- \( r=10, \ n=500, \ m=50 \)
- Better load balancing increase I-queue availability
- Benefit diminishes at high load
  - Few idle servers
  - Lightly loaded servers should report
Analysis of Primary Load Balancing System

• **Theorem 2**: Queue size $Q$ is distributed as

$$
P[Q = k] = \frac{P[Q_s = k]}{1 - \rho}, \quad k \geq 1
$$

where $Q_s$ is queue size of M/G/1 queue at reduced load $s = \lambda(1 - \rho)$ with same service time distribution and service discipline

• **Corollary 2**: Mean response time is

$$\bar{T} = \frac{\mathbb{E}Q_s}{s}
$$

• **Corollary 3**: Queue size distribution with PS is insensitive to service time distribution
Mean Response Time

- \( r = 10, \exp(1) \)
- Similar perf. for JIQ-Random and JIQ-SQ(d)
Mean Response Time

- PS, General Service
- JIQ-Random superior if load not too high
  - Few idle servers
  - Lightly loaded servers should report
- Not as well as SQ(2) at high load
  - Not as well as SQ(2) at high load
  - Few idle servers
  - Lightly loaded servers should report
Evaluation by Simulation

Fig. 7. Mean response time comparison for SQ(2), JIQ-Random and JIQ-SQ(2), with 7 different service time distributions. The smallest possible mean response time is 2 with a mean service time of 2.

Table 1 Percentage of improvement in queueing overhead of JIQ-Random over SQ(2).

<table>
<thead>
<tr>
<th>PS FIFO</th>
<th>0.5</th>
<th>0.9</th>
<th>0.5</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>R10</td>
<td>42.8%</td>
<td>49.9%</td>
<td>58.0%</td>
<td>33.2%</td>
</tr>
<tr>
<td>R20</td>
<td>68.7%</td>
<td>73.3%</td>
<td>76.9%</td>
<td>65.2%</td>
</tr>
<tr>
<td>R40</td>
<td>83.1%</td>
<td>85.9%</td>
<td>88.9%</td>
<td>81.2%</td>
</tr>
</tbody>
</table>

2. JIQ-SQ(2) achieves close to minimum response time. At a load of 0.5, the JIQ-SQ(2) algorithm achieves a mean queueing overhead less than 5% of the mean service time for the PS service discipline. For both disciplines, the mean response times with \( r = 10 \) never exceed 2 (2, and those with \( r = 20 \) and \( r = 40 \) are essentially 2. At a load of 0.9, for the PS service discipline, the mean response time of the JIQ-SQ(2) algorithm remains close to 2. At \( r = 10 \), it is around 2.9, and at \( r = 40 \), it never exceeds 2.1. The mean response time varies more under the FIFO service discipline. Even there, the JIQ-SQ(2) algorithm has mean response time below 3 for all service time distributions. This represents a 30-fold reduction (3 \( \times 30 \)) for PS and 30 \( \times 30 \)) for both disciplines at \( r = 40 \) and 0.9 load.

3. The JIQ algorithms are near-insensitive with PS in a finite system. Based on the simulation, the JIQ algorithms are near-insensitive to the service time distributions under the PS service discipline. We showed in Section 3 that the response times are insensitive to the service time distributions in the large system limit. The simulation verifies that in a system of 500–600 processors, the mean response times do not vary with service time distributions. JIQ algorithms with extension.

We evaluate the extension of the JIQ algorithms with reporting threshold equal to two at a high load of 0.99 in Fig. 8. This is the region where the performance of the original JIQ algorithms is similar to that of SQ(2), as shown in Fig. 6. However, with reporting threshold equal to two, the JIQ algorithms significantly outperform SQ(2). For instance, with exponential distribution, for which service disciplines do not affect response times, SQ(2) outperforms JIQ-Random with threshold equal to one in Fig. 6, but is outperformed by JIQ-Random with \( r = 10 \) and threshold equal to two, with 88% reduction in queueing overhead.

Observe the interesting phenomenon that the mean queue sizes are no longer monotonically increasing with variance of service times. In particular, the two bimodal distributions have smaller mean queue sizes than distributions with smaller
Observations

• JIQ-Random outperforms SQ(2)
• JIQ-SQ(2) achieves near optimal response time
• JIQ near-insensitive with PS in finite systems