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

COMPSCI 655
Lecture 20
Slides courtesy of Bo Jiang (thanks!)

In previous lectures

- Scheduling for single server (lecture 16)
  - Resource Allocation Policy determines which job to serve
  - Priority policy
- Multiple servers: $M/M/c$ queue (lecture 7)

![Diagram: Poisson distribution to single shared queue with M/M/c servers]
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

\[
\text{Poisson} \ (n\lambda) \quad \text{Exp}(1)
\]
How about Random?

- Assign jobs to random server

- Mean response time
  - 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

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 $K=8$ servers for a PS server farm with an anglo job-size distribution. There are many interesting things to see in Figure 7. First, when one notes that OPT-0 is in fact 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

\[ E[N] \]

OPT-0: each arriving job assigned so as to minimize mean response time of jobs in system assuming no further arrivals.
$E[N]$ = expected number of jobs at any server

PS server

$\lambda = \rho = 0.9, \ K = 8 = \# \ servers$


Nota: OPT-0 and LWL require knowledge of job size
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

- 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

\[
\begin{align*}
\mathbb{P}[\text{assign to queue with } \geq k \text{ jobs}] &= \pi_k^2 \\
\mathbb{P}[\text{assign to queue with } k \text{ jobs}] &= \pi_k^2 - \pi_{k+1}^2
\end{align*}
\]

- Consider set of queues with \( k \) jobs or more

\[
\pi_k - \pi_{k+1} = \lambda \left( \pi_{k-1}^2 - \pi_k^2 \right)
\]

- In equilibrium,

\[
\pi_k - \pi_{k+1} = \text{fraction queues with } k \text{ jobs}
\]
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 $\infty$
  \[ \pi_j = \lambda \pi_{j-1}^2 \]

- $\pi_0 = 1$ as all servers in state 0 (= 0 job or more)
  \[
  \begin{align*}
  \pi_1 &= \lambda \\
  \pi_2 &= \lambda \pi_1^2 = \lambda^{1+2} \\
  \pi_3 &= \lambda \pi_2^2 = \lambda^{1+2+2^2} \\
  \pi_k &= \lambda^{1+2+2^2+\cdots+2^{k-1}} = \lambda^{2^k-1}
  \end{align*}
  \]
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

\( \lambda = 0.8 \)

Graph:
- \( \pi_k \) vs. \( k \)
- Red: Random
- Blue: SQ(2)
**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

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: \( \frac{\lambda}{n\lambda(1 - \pi_1^d)} = \frac{1 - \lambda^d}{1 - \lambda} = \lambda + \lambda^2 + \cdots + \lambda^d \)
  - SQ(d): \( \frac{n\lambda}{n(1 - \pi_1)} = \frac{\lambda}{1 - \lambda} = \lambda + \lambda^2 + \lambda^3 + \cdots \)
  - 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 (JIQ)

• **Server completes all jobs**
  – Joins exactly one I-queue
    • This avoids withdrawal
  – **Which I-queue to join?**
    • Random → JIQ-Random
    • SQ(d) → 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
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 $r = n/m$
- Let $n, m \to \infty$
- Total arrival rate: Poisson $(n\lambda)$
- Arriving job randomly directed to a dispatcher
  - Arrival at each dispatcher: Poisson $(r\lambda)$
- General service time with mean 1
Analysis in large system limit

Two simplifying assumptions:

- Exactly one copy of each idle processor in I-queues (rare events)

- Only idle processors in I-queues (violated when idle server receives random arrival; in studied version of algorithm server does not withdraw from I-queue, too much com.)
Analysis in large system limit

Two types of arrivals:

- Arrivals randomly directed from a dispatcher with an empty I-queue: Poisson

- Arrivals assigned to idle server from I-queue: not Poisson for finite n (depend on queueing process at I-queue that has memory)
  → makes analysis challenging
Analysis of secondary load balancing system

Recall $r=n/m$ with $n,m \to \infty$ and $r$ kept constant

$\rho_n$ Proportion occupied I-queues in system with $n$ servers in equilibrium

$\rho_n \to \rho$ as $n \to \infty$
Analysis of secondary 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)
    $$\sum_{k=1}^{\infty} \rho \frac{d^{k-1}}{d-1} = r(1 - \lambda)$$

- "Proof": Show arrivals to I-queues are Poisson for $n \rightarrow \infty$; compute average I-queue length in two different ways.
Analysis of secondary load balancing system

\( \rho = \text{traffic intensity (or workload) in I-queue} \)

If arrivals Poisson and service times exponential (M/M/1 queue), expected \# customers I-queue is \( \frac{\rho}{1-\rho} \)

In steady-state server idle with probability 1-\( \lambda \)

\( \rightarrow \) expected \# idle servers is \( n(1-\lambda) \)

\( \rightarrow \) expected \# customers I-queue is \( \frac{n(1-\lambda)}{m} = r(1-\lambda) \)

Give equation \( \frac{\rho}{1-\rho} = r(1-\lambda) \)
Analysis of secondary load balancing system

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

- **“Proof”**:  
  - Arrival rate to idle server (see slide 30)
    \[
    \frac{n\lambda\rho}{n(1 - \lambda)} + \frac{n\lambda(1 - \rho)}{n} = \lambda(1 - \rho)(1 + r)
    \]
  - Arrival rate to busy server (see slide 31)
    \[
    \frac{n\lambda(1 - \rho)}{n} = \lambda(1 - \rho)
    \]
- Justification of formula in red box on slide 29
  \[ n\lambda = \text{total arrival rate} \]
  \[ \rho = \text{probability I-queue busy} \]
  \[ \rightarrow n\lambda\rho = \text{total arrival rate to occupied I-queues} \]
  \[ 1-\lambda = \text{probability a server is idle} \]
  \[ \rightarrow n(1-\lambda) = \text{expected number idle servers} \]

As job dispatched from an occupied I-queue to an idle server is
equally likely to join any idle server, the arrival rate to idle
server from jobs coming from I-queue is \[ \frac{n\lambda\rho}{n(1-\lambda)} = \frac{\lambda\rho}{(1-\lambda)} \]

- Justification of formula in red box on slide 29
  \[ n\lambda(1-\rho) = \text{arrival rate to idle I-queues} \]
  Job arriving in idle I-queue is as likely to join any of the n servers,
  and so the arrival rate to idle server from idle I-queue is
  \[ \frac{n\lambda(1-\rho)}{n} = \lambda(1-\rho) \]
A job may be assigned to a busy server only if I-queue it was assigned to is empty.

Arrival rate of jobs assigned to empty I-queue is $n\lambda(1-\rho)$ as $1-\rho$ is the probability that an I-queue is empty.

Each job assigned to empty I-queue is randomly sent to one of the $n$ servers, making $n\lambda(1-\rho)/n = \lambda(1-\rho)$ the arrival rate to busy servers.
Proportion of empty I-Queues

$r=10, \ n=500, \ m=50$

- Markers correspond to simulation results for values of $r$, $n$ and $m$ given above
- Dotted and straight lines correspond to results obtained by asymptotic analysis

The values for different service time distributions are indistinguishable.

3.2. Analysis of primary load balancing system

Using the results from the secondary load balancing system, we can solve for the response time of the primary system. Let $s = (1 - \rho)$, where $\rho$ is the proportion of occupied I-queues computed in Theorem 1.

Theorem 2 (Queue Size Distribution). Let the random variable $Q_n$ denote the queue size of the servers of an $n$-system in equilibrium. Let $Q_s$ denote the queue size of a $M/G^1$ server at the reduced load $s$ with the same service time distribution and service discipline, then

$$P(Q_n = k) \xrightarrow{n \to \infty} P(Q_s = k) \cdot \frac{1}{k}$$

Corollary 3 (Mean Response Time). Let the mean queue size at the servers in the large system limit be $\bar{q}$. It is given by

$$\bar{q} = q_s \cdot \frac{1}{s}$$

with $q_s$ being the mean queue size of the $M/G^1$ server with the same service time distribution and service discipline.

Corollary 4 (Insensitivity). The queue size distribution of the JIQ algorithm with PS service discipline in the large system limit depends on the service time distributions only through its mean.

We defer the proof of Theorem 2 to the Appendix and provide an outline below. Recall that there are two types of arrivals to the processors: one arrival process occurs through the I-queues and only when the processor is idle. The other arrival process occurs regardless of the processor queue length, when a job is randomly dispatched. The rate of each type of arrivals depends on the proportion of occupied I-queues, $\rho$. With probability $\rho$, an incoming job finds an occupied I-queue, and with probability $1 - \rho$, it finds an empty I-queue. Hence the arrival process due to random distribution is Poisson with rate $s = (1 - \rho)$.

Let the arrivals at empty I-queues be colored green, and the arrivals at occupied I-queues colored red. For an idle processor, the first arrival can be red or green, but the subsequent arrivals can only be green. The arrival process of the green jobs is Poisson with rate $s$, but the arrival process of the red jobs is not Poisson. However, observe that a busy period of a server in the JIQ system is indistinguishable from a busy period of an $M/G^1$ server with load $s$. Hence, the queue size distribution differs from that of an $M/G^1$ server only by a constant factor $s$.

Fig. 5 (a) shows the mean queue size for general service disciplines with exponential service time distributions and Fig. 5 (b) shows the mean queue size for FIFO service discipline with Weibull service time distribution with mean 2 and $r=10, n=500, m=50$. 
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

Fig. 4. Proportion of empty I-queues with r = 10. Line curves are obtained from Theorem 1. Markers are from simulations with n = 500 and m = 50. The values for different service time distributions are indistinguishable.

3.2. Analysis of primary load balancing system

Using the results from the secondary load balancing system, we can solve for the response time of the primary system. Let 

\[ s = (1 - \rho), \]

where \( \rho \) is the proportion of occupied I-queues computed in Theorem 1.

Theorem 2 (Queue Size Distribution). Let the random variable \( Q_n \) denote the queue size of the servers of an n-system in equilibrium. Let \( Q_s \) denote the queue size of a M\( G \)1 server at the reduced load s with the same service time distribution and service discipline, then

\[ P(Q_n = k) \sim P(Q_s = k) \sim \frac{1}{k} \quad \text{as} \quad n \to \infty \]

(3)

Corollary 3 (Mean Response Time). Let the mean queue size at the servers in the large system limit be \( \bar{q} \). It is given by

\[ \bar{q} = q_s \sim q_s \rho, \]

(4)

with \( q_s \) being the mean queue size of the M\( G \)1 server with the same service time distribution and service discipline.

The mean response time \( \bar{T} = \bar{q} = q_s \rho \), assuming a mean service time of 1.

Corollary 4 (Insensitivity). The queue size distribution of the JIQ algorithm with PS service discipline in the large system limit depends on the service time distributions only through its mean.

We defer the proof of Theorem 2 to the Appendix and provide an outline below. Recall that there are two types of arrivals to the processors: one arrival process occurs through the I-queues and only when the processor is idle. The other arrival process occurs regardless of the processor queue length, when a job is randomly dispatched. The rate of each type of arrivals depends on the proportion of occupied I-queues, \( \rho \). With probability \( \rho \), an incoming job finds an occupied I-queue, and with probability \( 1 - \rho \), it finds an empty I-queue. Hence the arrival process due to random distribution is Poisson with rate \( s = (1 - \rho) \).

Let the arrivals at empty I-queues be colored green, and the arrivals at occupied I-queues colored red. For an idle processor, the first arrival can be red or green, but the subsequent arrivals can only be green. The arrival process of the green jobs is Poisson with rate \( s \), but the arrival process of the red jobs is not Poisson. However, observe that a busy period of a server in the JIQ system is indistinguishable from a busy period of an M\( G \)1 server with load \( s \). Hence, the queue size distribution differs from that of an M\( G \)1 server with load \( s \) only by a constant factor \( s \).

Fig. 5 (a) shows the mean queue size for general service disciplines with exponential service time distributions and Fig. 5 (b) shows the mean queue size for FIFO service discipline with Weibull service time distribution with mean 2 and...
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

- $r = 10, \text{exp}(1)$
- Similar perf. for JIQ-Random and JIQ-SQ(d)

![Graph showing mean response time for JIQ-Random and JIQ-SQ(2) algorithms with $r = 10$. Fig. (a) has an exponential service time distribution with mean 1 (this makes the minimum possible mean response time 1), and Fig. (b) has a Weibull service time distribution with mean 2 and variance 20 (minimum possible mean response time is 2) and FIFO service discipline. Line curves are obtained from Theorem 2. Markers are from simulations with $n = 500$ and $m = 50$. Contrary to the improvement in $\lambda$ observed in Fig. 4, the performance gain of JIQ-SQ(2) over JIQ-Random is not significant at moderate load, and the magnitude of reduction in response time remains small even at higher load with $r = 10$. For instance, the reduction in response time is only 17% at $\lambda = 0$ (9 in Fig. 5(a). This is because improvement of JIQ-SQ(2) over JIQ-Random is most conspicuous when the random arrivals incur large queue sizes and there is significant improvement in $\lambda$. This is expected to happen at high load with a big processor to I-queue ratio, $r$. With $r = 10$, the queue sizes incurred by random arrivals is small at low loads, and the number of idle processors per I-queue is too small at high loads, resulting in similar mean response times for JIQ-Random and JIQ-SQ(2).

4. Design and extension

In this section, we seek to provide understanding of the relationship between the analytical results and system parameters, and to discuss extensions with local estimation of load at the servers.

4.1. Reduction of queueing overhead

We can compute the reduction of queueing overhead by JIQ-Random as Eq. (1) gives an explicit expression for $\lambda$. We compute $q_s$ for exponential service time distributions and general service discipline, $q_s = \frac{1}{s} \left(1 + \frac{r}{1 + r}\right)$. Eq. (5) also holds for PS service discipline and general service time distributions in the large system limit due to insensitivity. The mean response time for JIQ-Random is hence $\frac{1}{1 + \frac{r}{1 + r}}$, since the mean service time of a job is 1, the queueing overhead is $\frac{1}{1 + \frac{r}{1 + r}}$ for Random and $\frac{1}{1 + \frac{r}{1 + r}}$ for JIQ-Random. This is a $\frac{r}{1 + r}$-fold reduction.

For the FIFO service discipline, by the Pollaczek–Khinchin formula, $q_s = \frac{1}{s} \left(1 + \frac{r}{1 + r}\right)$. The ratio of the variance of the particular service time distribution to its mean squared is $C_2 = \frac{2}{\bar{x}^2}$, where $\bar{x}$ is the mean of the service time distribution.
Mean response time

- PS, General Service
- JIQ-Random superior if load not too high
- 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>Service Discipline</th>
<th>PS FIFO</th>
<th>R10</th>
<th>R20</th>
<th>R40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Det</td>
<td>0.5</td>
<td>23</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>Er    2</td>
<td>0.9</td>
<td>55</td>
<td>54</td>
<td>53</td>
</tr>
<tr>
<td>Exp</td>
<td>0.5</td>
<td>23</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>Bim-1</td>
<td>0.9</td>
<td>55</td>
<td>54</td>
<td>53</td>
</tr>
<tr>
<td>Web-1</td>
<td>0.5</td>
<td>23</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>Bim-2</td>
<td>0.9</td>
<td>55</td>
<td>54</td>
<td>53</td>
</tr>
<tr>
<td>Web-2</td>
<td>0.5</td>
<td>23</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>Bim-3</td>
<td>0.9</td>
<td>55</td>
<td>54</td>
<td>53</td>
</tr>
</tbody>
</table>

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 (30 for PS and 30 for FIFO) 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 variance.
Observations

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