Announcements

• No class next week (Tech. Series Lecture...)
• Party at Ted Romer’s on 2/21 (romer@cs)
• There’s only one typo in tonight’s slides

A Clarification

Multiple Class QNMs
• Introduction; Fundamental Laws
• Bottleneck Analysis
• Exact and Approximate Solution Techniques
• Approximate Techniques for Other Model Features
Single Class MVA

We solve for populations < N ONLY because that gives us the arrival instant queue lengths we need.
There is no assertion that we are trying to model the system as it operates with smaller populations.
Where’s the randomness?

Multiple Class QNMs

\[ N_c \quad Z_c \]

\[ \lambda_c \]

D_{1,2} \quad D_{2,2}

D_{1,3} \quad D_{2,3}

D_{1,4} \quad D_{2,4}
Why Have Multiple Classes?

- Sometimes distinct (sets of) customers have different behaviors
  - The model is more accurate when these distinctions are reflected than when they are averaged into a single representative class
    - By this I mean a multiple class model will give different results than “the equivalent” single class model
    - In general, the multiple class model will tend to exhibit lower performance

Why Have Multiple Classes?

- Equally important, multiple class models have input parameters and output measures that are per class
- This is convenient when the system has a set of naturally defined classes
  - **Inputs:** Can adjust arrival rates, for instance, for each class separately to project future performance
  - **Outputs:** Obtain response time (and other) measures for each class individually
    - (In general, it’s difficult to translate the outputs of a single class model into per class measures)
Example Multiclass Network

\[ \begin{align*}
\lambda_{\text{slow}} &= 0.5 \\
\lambda_{\text{fast}} &= 0.25
\end{align*} \]

In general, the classes can have completely distinct sets of demands, including demands of zero at some centers.

Fundamental Laws

\[ \begin{align*}
U_{c,k} &= X_c D_{c,k} \\
N_c &= X_c R_c
\end{align*} \]

\[ \begin{align*}
U_{\text{slow},2} &= 0.5 \times 1 = 50\% \\
U_{\text{fast},2} &= 0.25 \times 1 = 25\% \\
Q_{\text{slow},1} &= 0.5 \times 100 = 50
\end{align*} \]
Asymptotic Bounds

Which center is the bottleneck?

What does "bottleneck" even mean in a multiple class model?

\[ \sum_{c=1}^{C} \lambda_c D_{c,k} < 1 \]

Asymptotic Bounds

Basic observation is just as with single class models: the center utilization must be less than 1.0

\[ \lambda_A * 0.5 + \lambda_B * 1 < 1 \]
Asymptotic Bounds

D_{slow,1} = 100
D_{fast,1} = 5
D_{slow,2} = 4
D_{fast,2} = 2
D_{slow,3} = 3
D_{fast,3} = 3
D_{slow,4} = 8
D_{fast,4} = 0

Exact Analysis

\lambda_A \rightarrow D_{A,k}(S_{A,k}) \rightarrow \lambda_B

Residence Time = Service Time + Queueing Delay

The queueing delay depends on the scheduling discipline used:
- FCFS - we require \( S_{A,k} = S_{B,k} \)
- Processor Sharing (PS) - limiting case of Round-Robin (no service time restrictions)
- Last Come First Served Preemptive Resume (LCFS-PR) - no restrictions
Computing Residence Time

For FCFS scheduling, the analysis is similar to the single class case:

\[ R_{c,k}[\tilde{\lambda}] = V_{c,k} \left( S_{c,k} + \sum_{r=1}^{C} A_{r,k}^{(c)}[\tilde{\lambda}] S_{r,k} \right) \]

where

- \( \tilde{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_C) \) is the vector of arrival rates
- \( A_{r,k}^{(c)}[\tilde{\lambda}] \) average number of class \( r \) customers seen by an arriving class \( c \) customer
- \( S_{c,k} = S_{r,k} \) by assumption (for FCFS centers)

The Arrival Instant Theorem

\[ A_{r,k}^{(c)}[\tilde{\lambda}] = Q_{r,k}[\tilde{\lambda}] \]

The arrivals are random, so the “state” seen on arrival is the equilibrium state.
The Result

\[ R_{c,k}[\tilde{\lambda}] = V_{c,k} \left( S_{c,k} + \sum_{r=1}^{C} Q_{r,k}[\tilde{\lambda}] S_{r,k} \right) \]

\[ = D_{c,k} \left( 1 + \sum_{r=1}^{C} Q_{r,k}[\tilde{\lambda}] \right) \quad \text{(since service times are equal)} \]

\[ = D_{c,k} \left( 1 + \sum_{r=1}^{C} \lambda_{r} R_{r,k}[\tilde{\lambda}] \right) \]

So

\[ \frac{R_{r,k}[\tilde{\lambda}]}{R_{c,k}[\tilde{\lambda}]} = \frac{D_{r,k}}{D_{c,k}} \]

Simplifying the Result

\[ R_{c,k}[\tilde{\lambda}] = D_{c,k} \left( 1 + \sum_{r=1}^{C} \frac{D_{r,k}}{D_{c,k}} \lambda_{r} R_{c,k}[\tilde{\lambda}] \right) \]

\[ = \frac{D_{c,k}}{1 - \sum_{r=1}^{C} D_{r,k} \lambda_{r}} \]

\[ = \frac{D_{c,k}}{1 - \sum_{r=1}^{C} U_{r,k}[\tilde{\lambda}]} \]

\[ = \frac{D_{c,k}}{1 - U_{c}[\tilde{\lambda}]} \]

“service time inflation”
Example

\[ \lambda_{\text{slow}} = .1 \quad \lambda_{\text{fast}} = .15 \]

<table>
<thead>
<tr>
<th></th>
<th>slow</th>
<th>fast</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>10</td>
<td>0.75</td>
<td>10.75</td>
</tr>
<tr>
<td>R1</td>
<td>100</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>0.4</td>
<td>0.3</td>
<td>0.7</td>
</tr>
<tr>
<td>R2</td>
<td>13.33333</td>
<td>6.66667</td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>0.3</td>
<td>0.45</td>
<td>0.75</td>
</tr>
<tr>
<td>R3</td>
<td>12</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>0.8</td>
<td>0</td>
<td>0.8</td>
</tr>
<tr>
<td>R4</td>
<td>40</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Service Time Inflation

\[ D_{c,k} = \frac{D_{c,k}}{1 - \sum_{i=1}^{c} U_{i,k}[\lambda_i]} \]

\[ \lambda_{\text{slow}} = .1 \]

<table>
<thead>
<tr>
<th></th>
<th>fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>0.75</td>
</tr>
<tr>
<td>R1</td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td>0.571429</td>
</tr>
<tr>
<td>R2</td>
<td>13.333333</td>
</tr>
<tr>
<td>U3</td>
<td>0.545455</td>
</tr>
<tr>
<td>R3</td>
<td>12</td>
</tr>
<tr>
<td>U4</td>
<td>0.8</td>
</tr>
<tr>
<td>R4</td>
<td>40</td>
</tr>
</tbody>
</table>
Processor Sharing Scheduling

\[ R_{c,k} [\bar{\lambda}] = V_{c,k} \left( S_{c,k} + \sum_{r=1}^{C} A_{r,k}^{(c)} [\bar{\lambda}] S_{r,k} \right) \]

\[ = \ldots \]

\[ = D_{c,k} \left( 1 + \sum_{r=1}^{C} Q_{r,k} [\bar{\lambda}] \right) \]

Intuitive explanation: Arriving customer received only (the reciprocal) of this fraction of the processor’s cycles

LCFS-PR Scheduling

\[ R_{c,k} [\bar{\lambda}] = D_{c,k} \left( 1 + \sum_{r=1}^{C} Q_{r,k} [\bar{\lambda}] \right) \]

\[ = D_{c,k} \left( 1 + \sum_{r=1}^{C} \lambda_r R_{r,k} [\bar{\lambda}] \right) \]

\[ = D_{c,k} + \sum_{r=1}^{C} \lambda_r R_{c,k} [\bar{\lambda}] D_{r,k} \]

Intuitive explanation: Number of interruptions * time/interruption
Scheduling Summary

- FCFS, PS, and LCFS all give the same results for the performance measures we're considering:
  - utilization
  - mean residence/response time
  - mean queue length
  - (queue length distribution)
- FCFS requires that $S_{ck} = S_{r,k}$ for all classes $c$ and $r$ (But NOT that service demands be equal.)

Open QNMs

- As with single class models
  - utilizations can be computed from fundamental laws
  - compute residence times on a “per service center” basis
  - system response time is the sum of the residence times
- Time $O(CK)$
Closed QNMs

• Each class has its own set of loads, number of customers, and think time
• As always, the basic relationship is

\[ R_{c,k}[\vec{N}] = V_{c,k}\left(S_{c,k} + \sum_{i=1}^{C} A_{r,k}^{(c)}[\vec{N}]S_{r,k}\right) \]

where \( \vec{N} \equiv (N_1, N_2, \cdots, N_C) \)
• The key question is what is the value of \( A_{r,k}^{(c)} \)?

Multiclass Arrival Instant Theorem

\[ A_{r,k}^{(c)}[\vec{N}] = Q_{r,k}[\vec{N} - \vec{e}_c] \]

where \( \vec{e}_c = (0,0,\cdots,0,1,0,\cdots,0) \) is the unit vector in the \( c \) direction

As in the single class case, the intuition is that the arriving customer cannot see itself at the service center.
**Exact MVA**

\[ R_{ck}[\tilde{N}] = V_{ck}\left( S_{ck} + \sum_{r=1}^{c} Q_{rk}[\tilde{N} - \tilde{e}_{c}] S_{rk} \right) \]

\[ = D_{ck}\left( 1 + \sum_{r=1}^{c} Q_{rk}[\tilde{N} - \tilde{e}_{c}] \right) \]

Leading to the MVA approach

\[ R_{c}[\tilde{N}] = \sum_{k=1}^{K} R_{ck}[\tilde{N}] \]
\[ X_{c}[\tilde{N}] = \frac{N_{c}}{Z_{c} + R_{c}[\tilde{N}]} \]
\[ Q_{c,k}[\tilde{N}] = X_{c}[\tilde{N}] R_{c,k}[\tilde{N}] \]

---

**Example Structure**

\[ \tilde{N} = (2, 3) \]

\[ R_{c}[\tilde{N}] = \sum_{k=1}^{K} R_{c,k}[\tilde{N}] \]
\[ X_{c}[\tilde{N}] = \frac{N_{c}}{Z_{c} + R_{c}[\tilde{N}]} \]
\[ Q_{c,k}[\tilde{N}] = X_{c}[\tilde{N}] R_{c,k}[\tilde{N}] \]
Example Solution

\[ D_{A,1} = 1 \]
\[ D_{A,2} = 2 \]
\[ D_{B,1} = 3 \]
\[ D_{B,2} = 4 \]

<table>
<thead>
<tr>
<th>Population</th>
<th>(0,0)</th>
<th>(1,0)</th>
<th>(0,1)</th>
<th>(1,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R A,1</td>
<td>1</td>
<td></td>
<td>4/3</td>
<td></td>
</tr>
<tr>
<td>R A,2</td>
<td>3</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>R B,1</td>
<td></td>
<td>2</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>R B,2</td>
<td></td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>X A</td>
<td>&quot;1/4&quot;</td>
<td></td>
<td>&quot;3/19&quot;</td>
<td></td>
</tr>
<tr>
<td>X B</td>
<td>&quot;1/6&quot;</td>
<td></td>
<td>&quot;2/19&quot;</td>
<td></td>
</tr>
<tr>
<td>Q A,1</td>
<td>0</td>
<td>&quot;1/4&quot;</td>
<td>&quot;4/19&quot;</td>
<td></td>
</tr>
<tr>
<td>Q A,2</td>
<td>0</td>
<td>&quot;3/4&quot;</td>
<td>&quot;15/19&quot;</td>
<td></td>
</tr>
<tr>
<td>Q B,1</td>
<td>0</td>
<td>&quot;1/3&quot;</td>
<td>&quot;5/19&quot;</td>
<td></td>
</tr>
<tr>
<td>Q B,2</td>
<td>0</td>
<td>&quot;2/3&quot;</td>
<td>&quot;14/19&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Time (Space) Complexity of MVA for Closed Networks

\[ \prod_{c=1}^{C} (N_c + 1) \text{ populations total} \]
\[ O(C \cdot K) \text{ work per population} \]

<table>
<thead>
<tr>
<th>Customers/Class</th>
<th>Number of Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
</tr>
</tbody>
</table>
### Approximate MVA

- **Goal:** be faster!
- **How?** Iterate.

\[ Q_{i,k}[	ilde{N} - \tilde{e}_c] = \begin{cases} 
  Q_{i,k}[\tilde{N}] & r \neq c \\
  \frac{N_c - 1}{N_c} Q_{i,k}[\tilde{N}] & r = c 
\end{cases} \]

### Example Solution

#### Populations

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Class</th>
<th>Q 1</th>
<th>Q 2</th>
<th>X</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>0.25</td>
<td>0.75</td>
<td>0.166</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.333</td>
<td>0.667</td>
<td>0.111</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>0.211</td>
<td>0.79</td>
<td>0.158</td>
<td>6.333</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.263</td>
<td>0.737</td>
<td>0.105</td>
<td>9.5</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>0.195</td>
<td>0.805</td>
<td>0.154</td>
<td>6.474</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.253</td>
<td>0.747</td>
<td>0.104</td>
<td>9.579</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>0.193</td>
<td>0.807</td>
<td>0.154</td>
<td>6.495</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.249</td>
<td>0.751</td>
<td>0.104</td>
<td>9.610</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>0.192</td>
<td>0.808</td>
<td>0.154</td>
<td>6.508</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.248</td>
<td>0.752</td>
<td>0.104</td>
<td>9.614</td>
</tr>
<tr>
<td><strong>Exact</strong></td>
<td>A</td>
<td>0.211</td>
<td>0.789</td>
<td>0.158</td>
<td>6.333</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.263</td>
<td>0.737</td>
<td>0.105</td>
<td>9.500</td>
</tr>
</tbody>
</table>
Accuracy?

• In general, less accurate than the equivalent single class approximation
• Response time errors 5-30% are typical
• The problem is particularly acute when there are only a small number of customers in each class

Linearizer

• An improved, but more expensive, solution is obtained by estimating the change in the fraction of each class’s customers at each center caused by removing a single customer of some class

\[
F_{c,k}[^n] = \frac{Q_{c,k}[^n]}{n_c} \quad \delta_{c,k}[^n] = F_{c,k}[^n - \tilde{c}_r, r] - F_{c,k}[^n] \\
Q_{c,k}[^n - \tilde{c}_r] = (\tilde{n} - \tilde{c}_r) \left( F_{c,k}[^n] + \delta_{c,k}[^n] \right)
\]
Linearizer

- **Okay, how does it work?**
  - Guess some deltas
  - Iterate at all populations with one customer removed, using the current deltas
  - Take one step up (to the full population), using the exact MVA equations
  - Compute new deltas using the C+1 solutions just computed
  - Do this three times and stop. (I swear I didn't make this up.)

\[
\delta_{c,k}^{(i)}[\tilde{n}] \equiv F_{c,k}[\tilde{n} - \tilde{c}_c] - F_{c,k}[\tilde{n}]
\]

\[
Q_{c,k}[\tilde{n} - \tilde{c}_c] = (\tilde{n} - \tilde{c}_c) \cdot \left( F_{c,k}[\tilde{n}] + \delta_{c,k}^{(i)}[\tilde{n}] \right)
\]

*What could be simpler?*
AMVA vs. Linearizer

- Linearizer is about 3C times as slow as AMVA
- Accuracy is <5% (vs. <30% or so)
- AMVA is more easily adapted to handle characteristics that do not meet our assumptions
  - e.g., deterministic service times

Mixed Networks

- Hard to believe, but there’s still one more kind of network to solve: those with some open and some closed classes
- The solution is simple, actually:
  - First, use fundamental laws to compute the utilizations of all the (queueing) centers by the open classes
  - Now inflate the service times of the closed classes by $1 - U_{open,k}$
  - Remove the open classes and solve to get closed class residence times, throughputs, and queue lengths
  - Use the closed class queue lengths to compute open class residence times
Example

$\lambda_A = .25$

$N_B = 1$

$D_A,1 = 1 \rightarrow U_A,1 = .25$

$D_B,1 = 2$

$D_A,2 = 3 \rightarrow U_A,2 = .75$

$D_B,2 = 4$

$N_B = 1$

$D_B,1 = 8/3$

$Q_B,1 = 8/54$

$D_B,2 = 16$

$Q_B,2 = 48/54$

$R_A,1 = 1(1 + Q_A,1 + 8/54)/.75$

$R_A,2 = 3(1 + Q_A,2 + 48/54)/.25$

What Can We Model?

- The standard QNMs impose restrictions that leave out features that might be vital in some situations
  - effects of memory contention
  - software parallelism
  - priority scheduling
  - I/O path components (e.g., controllers)
  - locking
How Can We Address These?

- There are only a few options open to us
  - Reflect the features in the model structure (e.g., adjust the service demands)
  - Develop an approximate analysis technique (e.g., deterministic service times)
  - Resort to another approach (e.g., simulation or lower level analytic models)

Adjusting the Model Structure

File Cache: Imagine you want to estimate the impact of adding memory that will be used primarily to increase the size of the file cache (→ higher hit rates)

\[ D'_{c,Disk} = \frac{(\text{new miss prob})/(\text{old miss prob})}{D_{c,Disk}} \]

\[ D_{c,Disk} = (\text{#block requests/customer}) \times (\text{cache miss probability}) \times (\text{disk service time}) \]

So, we must estimate the miss probability with the increased cache size
Embellishments

- Notice that if we're modeling an existing system, our measurements of $D_{c, Disk}$ capture the current miss probability (so all we really need to get right is the relative change in the miss prob).
- How do we make the estimate? It's ad hoc.

$$\frac{\text{new prob}}{\text{old prob}} = \left( \frac{\text{old size}}{\text{new size}} \right)^{\frac{1}{\alpha}}$$

- Note that doing something “better” (e.g., using simulation) requires a lot more data/work.
- Note also that you can combine simulation (to get the miss rate) and the QNM (to evaluate overall system performance).

Example II

- Imagine the Webcrawler spawns a thread on search queries to update tracking information.

\[ \lambda_{\text{search}} = 0.25 \]

\[ \lambda_{\text{fetch}} = 0.3 \]

\[ \lambda_{\text{track}} = \lambda_{\text{search}} \]
Adjusting the Solution Technique: FCFS w/ Different Means

- Imagine that the system supports both data-oriented and multimedia apps.
- The data-oriented fetch single disk blocks each I/O, while the multimedia fetch multiple blocks.

\[ R_{c,k}[^\lambda] = V_{c,k} \left( S_{c,k} + \sum_{r=1}^{C} Q_{r,k}[^\lambda]S_{r,k} \right) \]

*Can’t reduce this, but can solve it (iteratively or directly).*

“Odd” Service Time Distributions

- Problem: Customer in service has an expected remaining service time not equal to the mean time.

\[ R_{c,k}[^\lambda] = V_{c,k} \left( S_{c,k} + \sum_{r=1}^{C} Q_{r,k}[^\lambda] - U_{r,k}[^\lambda] \right)S_{r,k} + U_{r,k}[^\lambda]F_{r,k}[^\lambda] \]

- \( F_{r,k}[^\lambda] = \frac{S_{r,k}}{2} \) deterministic
- \( F_{r,k}[^\lambda] = 4S_{r,k} \) something else
Priority Scheduling

• Imagine class A has (preemptive) priority over B, B over C, etc.

\[ \lambda_A \rightarrow \lambda_B \rightarrow \lambda_C \]

Class Exercise: Model Structure
Class Exercise: Solution Technique

"Simultaneous Resource Possession"

- Suppose a device controller is shared and is a potential bottleneck
- The devices cache fetched data, so they can use the controller as soon as it becomes free
- (Assume SIOs have negligible delay)
SRP

\[ R_{c,k}[\tilde{N}] = V_{c,k}(S_{c,k} + \text{controller delay}) \left( 1 + \sum_{r=1}^{C} Q_{r,k}^{(c)}[\tilde{N}] \right) \]

\( Z_c \) chosen to match \( X_k \)

\( N_c = 1 \)

\( C = \# \) disks

Software Locks

Looks familiar, eh?