Two-Level Iterative Queuing Modeling of Software Contention

Daniel A. Menascé
Dept. of Computer Science
George Mason University

www.cs.gmu.edu/faculty/menasce.html
Motivation

Software View

Hardware View
% Software Contention Time vs. Multithreading Level
%Software Contention Time vs Non-CS/CS Ratio

© 2002 D. Menascé. All Rights Reserved.
Software Contention

example: multithreaded software server
QN Models that Capture SW Contention

1. Carleton University’s Layered Queuing Networks (LQN)

2. Menascé’s two-level iterative SQN-HQN model
QN Models that Capture SW Contention

1. Carleton University’s Layered Queuing Networks (LQN)

2. Menascé’s two-level iterative SQN-HQN model
If we just consider the SQN we are ignoring time spent at the software resources due to contention for hardware resources.
If we just consider the SQN we are ignoring time spent at the software resources due to contention for hardware resources.

The HQN model must consider that some processes are not using hardware resources because they are blocked for software resources.
SQN-HQN Scheme

\[ \sum_j s_j D_{sh,i}^j \]

Software QN

Adjust demands

Hardware QN

\[ D_{j}^s \]

\[ B \]

\[ R'_i(N^h) \]

\[ N^h \]

\[ N \]
Input Service Demands

<table>
<thead>
<tr>
<th>Hardware Devices (↓)</th>
<th>NCS</th>
<th>CS 1</th>
<th>CS 2</th>
<th>Hardware Demands</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>0.2000</td>
<td>0.0600</td>
<td>0.0808</td>
<td>0.3408</td>
</tr>
<tr>
<td>Disk 1</td>
<td>0.0560</td>
<td>0.0576</td>
<td>0.0000</td>
<td>0.1136</td>
</tr>
<tr>
<td>Disk 2</td>
<td>0.0360</td>
<td>0.0000</td>
<td>0.1212</td>
<td>0.1572</td>
</tr>
<tr>
<td>Disk 3</td>
<td>0.0360</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0360</td>
</tr>
<tr>
<td>Software Demands</td>
<td>0.3280</td>
<td>0.1176</td>
<td>0.2020</td>
<td></td>
</tr>
</tbody>
</table>

- sum across a column
- sum across a row
# Service Demands

<table>
<thead>
<tr>
<th>Hardware Devices (↓)</th>
<th>NCS</th>
<th>CS 1</th>
<th>CS 2</th>
<th>Hardware Demands</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>0.2000</td>
<td>0.0600</td>
<td>0.0808</td>
<td>0.3408</td>
</tr>
<tr>
<td>Disk 1</td>
<td>0.0560</td>
<td><strong>0.0576</strong></td>
<td>0.0000</td>
<td><strong>0.1136</strong></td>
</tr>
<tr>
<td>Disk 2</td>
<td>0.0360</td>
<td>0.0000</td>
<td>0.1212</td>
<td>0.1572</td>
</tr>
<tr>
<td>Disk 3</td>
<td>0.0360</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0360</td>
</tr>
<tr>
<td>Software Demands</td>
<td>0.3280</td>
<td><strong>0.1176</strong></td>
<td>0.2020</td>
<td></td>
</tr>
</tbody>
</table>

- $D_{cs1}^s$ (total service time of a software module)
- $D_{cs1,disk1}^{sh}$ (total service time of a software module at a physical device)
- $D_{disk1}^h$ (total service time of the application at a given device)
SQN-HQN Scheme

\[ D_{sh}^{j,i} \]

\[ D^s_j \]

\[ D^h_i \]

\[ \sum_j \]

\[ R'_i (N^h) \]

\[ B \]

\[ N^h \]

Software QN

Hardware QN

Adjust demands

Avg. number of processes blocked due to software contention.
SQN-HQN Scheme

$$D_{sh}^{j,i}$$

Software QN

$$\sum_j D_j^s$$

Adjust demands

$$R_i'(N^h)$$

Hardware QN

$$\sum_j D_j^h$$

Avg. number of processes blocked due to software contention.

Adjusted population for HQN

$$N$$
SQN-HQN Scheme

Adjusted demands for SQN.

Adjusted demands for HQN.

Avg. number of processes blocked due to software contention.

Adjusted population for HQN.
Basic Idea

• Iteration between solving the SQN and the HQN.
• Number B of processes blocked due to software contention computed through the SQN.
• Population at HQN is reduced by B.
• Service demands at SQN are adjusted to account for physical contention.
SQN-HQN: Initialization

\[ \sum_{j} s_j D_{sh}^j D_{sh}^i \]

Software QN

Adjust demands

Hardware QN

No adjustment at initialization

\[ D^s_{sh} \]

\[ D^s_j \]

\[ D^h_i \]

\[ R'_i(N^h) \]

\[ B \]

\[ N^h \]

\[ N \]
SQN-HQN: Solve SQN – No Hardware Contention

\[ D_{sh}^{j,i} \]

\[ \sum_j D_j^s \]

\[ R_i'(N^h) \]

\[ N^h = N - B \]

\[ D_i^h \]

\[ N \]
SQN-HQN: Solve HQN

\[ \sum_{j} s_j D_{sh}^{j,i} = D_j^s - B \]

\[ N^h = N - B \]
SQN-HQN: Adjust demands for SQN

\[ D_{sh,j,i}^{s} \]

\[ \sum_j D_{j}^{s} \rightarrow \text{Adjust demands} \]

\[ D_{j}^{s} \rightarrow R_i'(N^h) \]

\[ B \rightarrow N^h = N - B \]

Software QN

Hardware QN
SQN-HQN: Solve SQN Again

\[ \sum_j s_j D_{j,i}^{sh} \]

Software QN

Adjust demands

Hardware QN

\[ N^h = N - B \]
SQN-HQN: Solve HQN

\[ D_{j,i}^{sh} \]

\[ D_j^s \]

Adjust demands

\[ R_i'(N^h) \]

\[ N^h = N - B \]

Software QN

Hardware QN

\[ \sum_j D_j^h \]

\[ D_i^h \]
SQN-HQN: Adjust demands for SQN

\[ D_{j,i}^{sh} \]

\[ D_{j}^{s} \]

\[ D_{i}^{h} \]

\[ \sum_{j} \]

\[ R_{i}^{'}(N_{h}^{h}) \]

\[ N_{h}^{h} = N - B \]

Software QN

Hardware QN

Adjust demands

\[ D_{sh,i,j} \]

\[ B \]

\[ N \]
Convergence is checked on absolute relative error on the number of blocked processes in the SQN.
Adjustment of SQN Demands

- Single class case:
  \[ D^s_j \leftarrow \sum_i \left( \frac{D^{sh}_{j,i}}{D^h_i} \times R'_i(N^h) \right) \]

- Multiple class case:
  \[ D^s_{j;r} \leftarrow \sum_i \left( \frac{D^{sh}_{j;i,r}}{D^h_{i;r}} \times R'_{i;r}(\vec{N}^h) \right) \]
Comparison with other approaches

<table>
<thead>
<tr>
<th>N</th>
<th>SQN-HQN</th>
<th>GB</th>
<th>SQN_HQN</th>
<th>ASM</th>
<th>ASPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.544</td>
<td>1.54</td>
<td>0.27</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.088</td>
<td>2.11</td>
<td>1.06</td>
<td>4.60</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.317</td>
<td>2.37</td>
<td>2.22</td>
<td>5.89</td>
<td>4.2</td>
</tr>
<tr>
<td>4</td>
<td>2.428</td>
<td>2.49</td>
<td>2.49</td>
<td>5.99</td>
<td>2.8</td>
</tr>
<tr>
<td>5</td>
<td>2.487</td>
<td>2.56</td>
<td>2.86</td>
<td>5.87</td>
<td>2.0</td>
</tr>
<tr>
<td>6</td>
<td>2.521</td>
<td>2.60</td>
<td>3.05</td>
<td>5.78</td>
<td>1.5</td>
</tr>
<tr>
<td>7</td>
<td>2.541</td>
<td>2.62</td>
<td>3.00</td>
<td>5.75</td>
<td>1.5</td>
</tr>
<tr>
<td>8</td>
<td>2.555</td>
<td>2.63</td>
<td>2.86</td>
<td>5.77</td>
<td>1.1</td>
</tr>
</tbody>
</table>

GB: global balance equations

SQN is consistently pessimistic.
ASPA is much more complex to implement.
ASM only works for one class.
Modeling Non-Software Resources

**Software QN**

- Client think time
- **ncs**
- **cs1**
- **cs2**
- Network

**Hardware QN**

- Client think time
- **CPU**
- **disk 1**
- **disk 2**
- **disk 3**
- Network

B is the avg. no. of processes in the software resource waiting lines.
Open QN at the Software Level

Software QN

Hardware QN
SQN-HQN Scheme: Open SQN

\[ \sum_j s_j D_{sh}^{j,i} \]

Adjust demands

\[ D_j^s \]

Software QN

\[ R_i'(N^h) \]

Hardware QN

\[ N^h = \overline{N} - B \]

\[ \overline{N}, B \]

\[ \lambda \]

\[ D_i^h \]

\[ D_{j}^{sh} \]
### Results of Iterations for Open SQN Case

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Nh</th>
<th>Resp. Time</th>
<th>Ns</th>
<th>B</th>
<th>NCS</th>
<th>CS1</th>
<th>CS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>0.3280</td>
<td>0.1176</td>
<td>0.2020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.295</td>
<td>0.821</td>
<td>1.641</td>
<td>0.346</td>
<td>0.3662</td>
<td>0.1302</td>
<td>0.2235</td>
</tr>
<tr>
<td>2</td>
<td>1.440</td>
<td>0.946</td>
<td>1.893</td>
<td>0.453</td>
<td>0.3858</td>
<td>0.1365</td>
<td>0.2342</td>
</tr>
<tr>
<td>3</td>
<td>1.513</td>
<td>1.014</td>
<td>2.028</td>
<td>0.515</td>
<td>0.3958</td>
<td>0.1397</td>
<td>0.2396</td>
</tr>
<tr>
<td>4</td>
<td>1.550</td>
<td>1.050</td>
<td>2.100</td>
<td>0.549</td>
<td>0.4010</td>
<td>0.1414</td>
<td>0.2424</td>
</tr>
<tr>
<td>5</td>
<td>1.570</td>
<td>1.069</td>
<td>2.137</td>
<td>0.568</td>
<td>0.4037</td>
<td>0.1423</td>
<td>0.2438</td>
</tr>
<tr>
<td>6</td>
<td>1.580</td>
<td>1.079</td>
<td>2.157</td>
<td>0.577</td>
<td>0.4051</td>
<td>0.1427</td>
<td>0.2446</td>
</tr>
<tr>
<td>7</td>
<td>1.585</td>
<td>1.084</td>
<td>2.167</td>
<td>0.582</td>
<td>0.4059</td>
<td>0.1430</td>
<td>0.2450</td>
</tr>
<tr>
<td>8</td>
<td>1.588</td>
<td>1.086</td>
<td>2.173</td>
<td>0.585</td>
<td>0.4062</td>
<td>0.1431</td>
<td>0.2452</td>
</tr>
<tr>
<td>9</td>
<td>1.589</td>
<td>1.088</td>
<td>2.175</td>
<td>0.587</td>
<td>0.4064</td>
<td>0.1432</td>
<td>0.2453</td>
</tr>
<tr>
<td>10</td>
<td>1.590</td>
<td>1.088</td>
<td>2.177</td>
<td>0.587</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Adjusted SQN Demands
Modeling and Optimization of Multitiered Server-based Systems
The SQN-HQN Model

- **SQN**
  - the software queueing network
  - models the software application using a software network
- **HQN**
  - the hardware queueing network
  - models the hardware infrastructure on top of which the software runs

- Software modules blocked at the SQN are not counted as active in the HQN.
- Contention computed by the HQN is used to elongate the execution time of software modules at the SQN.
The SQN-HQN Model

\[ D_j^s = \sum_i D_{j,i}^{sh} \times R_i'(N^h) \quad (1) \]

while \(|N_k^b - N_{k-1}^b| < \epsilon\)
A 3-tiered Server-based System: the SQN

single-queue multiple-servers / Open
Seidmann’s Approximation to Model

Multi-server Queues

- \( m \) demand = \( D/m \)
- \( (m-1) \) demand = \( D(m-1)/m \)

Delay device

Load independent
Residence time at SQN

\[ R = \begin{cases} \frac{D}{1 - \lambda D} & \text{load independent} \\ D & \text{delay} \end{cases} \]

\[
R_{SQN} = \frac{D_{ws}/m}{1 - \lambda D_{ws}/m} + \frac{D_{ws}(m - 1)}{m} + \\
\frac{D_{as}/n}{1 - \lambda D_{as}/n} + \frac{D_{as}(n - 1)}{n} + \\
\frac{D_{as}/p}{1 - \lambda D_{as}/p} + \frac{D_{as}(p - 1)}{p}
\]
A 3-tiered Server-based System: the **HQN**

**A Closed QN**

- Models the hardware infrastructure (processors, storage devices, load balancers, etc.) on top of which the software runs.
Mapping the SQN to the HQN

- Mapping the service demand at each SW tier to the HW it uses.
- Example:
Mapping the SQN into the HQN

- Mapping the service demand at each SW tier to the HW it uses.
- Example:

<table>
<thead>
<tr>
<th>Hardware Component</th>
<th>Software server</th>
<th>HQN Demands $(D^h_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WS</td>
<td>AS</td>
</tr>
<tr>
<td>CPU 1</td>
<td>$f_1 D_{ws}$</td>
<td>$f_2 D_{as}$</td>
</tr>
<tr>
<td>Disk 1.1</td>
<td>$(1 - f_1)D_{ws}$</td>
<td>0.00</td>
</tr>
<tr>
<td>Disk 1.2</td>
<td>0.00</td>
<td>$(1 - f_2)D_{as}$</td>
</tr>
<tr>
<td>CPU 2</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Disk 2.1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQN Demands $(D^s_j)$</td>
<td>$D_{ws}$</td>
<td>$D_{as}$</td>
</tr>
</tbody>
</table>

- $f_1$, $f_2$ and $f_3 \in [0,1]$ indicate the fraction of the demand of a software module spent at CPU.
The SQN-HQN Model

\[ D_j^s = \sum_{i} D_{j,i}^{sh} \times R_i'(N^h) \quad (1) \]

while \( |N_k^b - N_{k-1}^b| < \epsilon \)

AND \( U_{ws} = \lambda D_{ws}/m < 1 \)

AND \( U_{as} = \lambda D_{as}/n < 1 \)

AND \( U_{ds} = \lambda D_{ds}/p < 1 \)
Modeling the Multilayered System using SQN-HQN

- $N^b = N^{b}_{ws} + N^{b}_{as} + N^{b}_{ds}$

- $N^{b}_{ws} = \text{avg number in the queue} - \text{utilization}$
  
  \[
  = \frac{\lambda D_{ws}/m}{1 - \lambda D_{ws}/m} - \frac{\lambda D_{ws}}{m}
  \]
  
  \[
  = \frac{(\lambda D_{ws}/m)^2}{1 - \lambda D_{ws}/m}
  \]
Modeling the Multitiered System using SQN-HQN

- \( N^{b} = N^{b}_{ws} + N^{b}_{as} + N^{b}_{ds} \)

- \( N^{b}_{ws} = \frac{(\lambda D_{ws}/m)^2}{1-\lambda D_{ws}/m} \)

- \( N^{b}_{as} = \frac{(\lambda D_{as}/n)^2}{1-\lambda D_{as}/n} \)

- \( N^{b}_{ds} = \frac{(\lambda D_{ds}/p)^2}{1-\lambda D_{ds}/p} \)
Example – used parameters

\[
\lambda = 2.5 \text{ tps}
\]

number of threads in each tier
\[
\begin{aligned}
m &= 10 \\
n &= 20 \\
p &= 25
\end{aligned}
\]

Demands
\[
\begin{aligned}
D_{ws} &= 0.340 \text{ sec} & f_1 &= 0.60 \\
D_{as} &= 0.200 \text{ sec} & f_2 &= 0.40 \\
D_{ds} &= 0.300 \text{ sec} & f_3 &= 0.75
\end{aligned}
\]
Example

• First iterations of the SQN-HQN method -

<table>
<thead>
<tr>
<th>Iter. No.</th>
<th>$D_{ws}$</th>
<th>$D_{as}$</th>
<th>$D_{ds}$</th>
<th>$N^b_{ws}$</th>
<th>$N^b_{as}$</th>
<th>$N^b_{ds}$</th>
<th>$N^b$</th>
<th>Abs. Error ($\Delta$)</th>
<th>$R_{SQN}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.340</td>
<td>0.200</td>
<td>0.300</td>
<td>0.008</td>
<td>0.001</td>
<td>0.001</td>
<td>0.009</td>
<td>1.000</td>
<td>0.844</td>
</tr>
<tr>
<td>1</td>
<td>0.446</td>
<td>0.250</td>
<td>0.374</td>
<td>0.014</td>
<td>0.001</td>
<td>0.001</td>
<td>0.016</td>
<td>0.424</td>
<td>1.076</td>
</tr>
<tr>
<td>2</td>
<td>0.505</td>
<td>0.277</td>
<td>0.413</td>
<td>0.018</td>
<td>0.001</td>
<td>0.002</td>
<td>0.021</td>
<td>0.227</td>
<td>1.203</td>
</tr>
<tr>
<td>3</td>
<td>0.540</td>
<td>0.292</td>
<td>0.433</td>
<td>0.021</td>
<td>0.001</td>
<td>0.002</td>
<td>0.024</td>
<td>0.129</td>
<td>1.275</td>
</tr>
<tr>
<td>4</td>
<td>0.559</td>
<td>0.300</td>
<td>0.445</td>
<td>0.023</td>
<td>0.001</td>
<td>0.002</td>
<td>0.026</td>
<td>0.071</td>
<td>1.316</td>
</tr>
<tr>
<td>5</td>
<td>0.571</td>
<td>0.305</td>
<td>0.452</td>
<td>0.024</td>
<td>0.002</td>
<td>0.002</td>
<td>0.027</td>
<td>0.040</td>
<td>1.339</td>
</tr>
<tr>
<td>6</td>
<td>0.577</td>
<td>0.308</td>
<td>0.456</td>
<td>0.024</td>
<td>0.002</td>
<td>0.002</td>
<td>0.028</td>
<td>0.023</td>
<td>1.352</td>
</tr>
<tr>
<td>7</td>
<td>0.580</td>
<td>0.310</td>
<td>0.458</td>
<td>0.025</td>
<td>0.002</td>
<td>0.002</td>
<td>0.028</td>
<td>0.013</td>
<td>1.359</td>
</tr>
<tr>
<td>8</td>
<td>0.582</td>
<td>0.310</td>
<td>0.459</td>
<td>0.025</td>
<td>0.002</td>
<td>0.002</td>
<td>0.029</td>
<td>0.007</td>
<td>1.363</td>
</tr>
</tbody>
</table>

$$\Delta = \left| \frac{N^b(k) - N^b(k-1)}{N^b(k)} \right| .$$

Daniel A. Menascé
Example

• First iterations of the SQN-HQN method - the HQN:

<table>
<thead>
<tr>
<th>Iter. No.</th>
<th>$R^\epsilon_{CPU1}$</th>
<th>$R^\epsilon_{Disk1,1}$</th>
<th>$R^\epsilon_{Disk1,2}$</th>
<th>$R^\epsilon_{CPU2}$</th>
<th>$R^\epsilon_{Disk2,1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.401</td>
<td>0.158</td>
<td>0.137</td>
<td>0.293</td>
<td>0.081</td>
</tr>
<tr>
<td>1</td>
<td>0.469</td>
<td>0.168</td>
<td>0.145</td>
<td>0.329</td>
<td>0.084</td>
</tr>
<tr>
<td>2</td>
<td>0.511</td>
<td>0.173</td>
<td>0.148</td>
<td>0.348</td>
<td>0.085</td>
</tr>
<tr>
<td>3</td>
<td>0.534</td>
<td>0.176</td>
<td>0.150</td>
<td>0.360</td>
<td>0.086</td>
</tr>
<tr>
<td>4</td>
<td>0.547</td>
<td>0.177</td>
<td>0.151</td>
<td>0.366</td>
<td>0.086</td>
</tr>
<tr>
<td>5</td>
<td>0.555</td>
<td>0.178</td>
<td>0.152</td>
<td>0.369</td>
<td>0.086</td>
</tr>
<tr>
<td>6</td>
<td>0.559</td>
<td>0.179</td>
<td>0.152</td>
<td>0.371</td>
<td>0.086</td>
</tr>
<tr>
<td>7</td>
<td>0.562</td>
<td>0.179</td>
<td>0.152</td>
<td>0.372</td>
<td>0.086</td>
</tr>
<tr>
<td>8</td>
<td>0.563</td>
<td>0.179</td>
<td>0.152</td>
<td>0.373</td>
<td>0.086</td>
</tr>
</tbody>
</table>
Example

- $R_{SQN}$ as a function of $n$ and $p$
- $m=30$
- $\lambda = 3.6$ tps
Near Optimal Number of Threads

- Problem: what is the optimal number of web servers, application servers and database servers that minimizes the response time?

- More precisely, find

\[(m^*, n^*, p^*) = \arg\min_{(m,n,p)} \{ R_{SQN}(m,n,p) \}\]

- an exact optimization solver is not an option because there is no closed form expression for \( R_{SQN} \). Instead, there is an iterative algorithm.

Daniel A. Menascé
Near Optimal Number of Threads

- A combinatorial search technique can be used to search the solution space for the near optimal tuple.
- We used hill-climbing, which starts from an arbitrary point in the search space, and builds a relatively small neighborhood, $\mathcal{N}$, by perturbing the values of each element of $(m, n, p)$

$$
\mathcal{N} = \{(m', n', p') = (m + \delta, n + \delta, p + \delta) \mid 
$$

- For example,

$$
\delta \in \{-1, 0, 1\}, \quad (m', n', p') \neq (m, n, p),
$$

$$
\frac{\lambda D_{ws}}{m'} < 1, \quad \frac{\lambda D_{as}}{n'} < 1, \quad \frac{\lambda D_{as}}{p'} < 1 \text{ and } \\
m' \geq 1, n' \geq 1, p' \geq 1
$$
Near Optimal Number of Threads

• Additionally, the number of threads of each type in each point in the neighborhood also has to satisfy the memory constraint in each physical machine.

• In other words,

\[
m \times M_{ws} + n \times M_{as} \leq AM_{M1} \\
p \times M_{ds} \leq AM_{M2}
\]

where \( M_{ws}, M_{as} \) and \( M_{ds} \) are the memory footprint for each thread type, and \( AM_{M1} \) and \( AM_{M2} \) stand for the available memory on machines 1 and 2.
Hill Climbing Algorithm

Algorithm 1 Hill-climbing algorithm

\[(m, n, p) \leftarrow \text{StartConfig};\]
\[\text{MinRespTime} \leftarrow \text{RespTime}\left((m, n, p)\right);\]
\[\text{for} \quad \text{NumRestarts} = 1 \text{ to } \text{MaxRestarts} \text{ do}\]
\[\text{NumIter} \leftarrow 1;\]
\[\text{// Do one start/restart */}\]
\[\text{while} \quad \text{NumIter} \leq \text{MaxIter} \text{ do}\]
\[\text{// Build neighborhood of } (m, n, p). \text{ See Eq. 10. */}\]
\[\mathcal{N} \leftarrow \text{Neighborhood}\left((m, n, p)\right);\]
\[\text{// Find config. with smallest resp. time in } \mathcal{N}. */\]
\[(m', n', p') \leftarrow \text{argmin}_{(m, n, p) \in \mathcal{N}} \text{RespTime}\left((m, n, p)\right);\]
\[\text{// Find resp. time of } (m', n', p'). \text{ See Section IV */}\]
\[\text{NewRespT} \leftarrow \text{RespTime}\left((m', n', p')\right);\]
\[\text{if} \quad \text{NewRespT} < \text{MinRespTime} \text{ then}\]
\[\text{// Resp. time of } (m', n', p') < \text{that of } (m, n, p) */\]
\[\text{MinRespTime} \leftarrow \text{NewRespT};\]
\[\text{// move the neighborhood center to } (m', n', p'). */\]
\[(m, n, p) \leftarrow (m', n', p');\]
\[\text{// increment number of iterations */}\]
\[\text{NumIter} \leftarrow \text{NumIter} + 1;\]
\[\text{end if}\]
\[\text{end while}\]
\[\text{// Record local optimum in LocalOpt array */}\]
\[\text{LocalOpt[NumRestarts].Config} \leftarrow (m, n, p);\]
\[\text{LocalOpt[NumRestarts].RespT} \leftarrow \text{MinRespTime};\]
\[\text{// set a new point for a random restart */}\]
\[(m, n, p) \leftarrow \text{RandomConfig};\]
\[\text{end for}\]
\[\text{// Find index of best local optimum */}\]
\[\text{OptConfigIndex} \leftarrow \text{argmin}_i \text{LocalOpt[i].RespT};\]
\[\text{// Return best local optimum configuration */}\]
\[\text{Return } \text{LocalOpt[OptConfigIndex].Config}\]
Near Optimal Results for Varying Arrival Rates – used parameters

Demands
\[
\begin{align*}
D_{ws} &= 0.340 \text{ sec} & f_1 &= 0.60 \\
D_{as} &= 0.200 \text{ sec} & f_2 &= 0.40 \\
D_{ds} &= 0.300 \text{ sec} & f_3 &= 0.75
\end{align*}
\]

Memory footprint
\[
\begin{align*}
M_{ws} &= 100 \text{ MB} \\
M_{as} &= 150 \text{ MB} \\
M_{ds} &= 100 \text{ MB}
\end{align*}
\]

Available memory
\[
\begin{align*}
AM_{\text{machine1}} &= 8000 \text{ MB} \\
AM_{\text{machine2}} &= 4000 \text{ MB}
\end{align*}
\]

Hill climbing
\[
\begin{align*}
\text{MaxIter} &= 200 \\
\text{MaxRestarts} &= 50
\end{align*}
\]
Near Optimal Results for Varying Arrival Rates

<table>
<thead>
<tr>
<th>Arrival rate (tps)</th>
<th>m</th>
<th>n</th>
<th>p</th>
<th>$R_{SQN}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>41</td>
<td>26</td>
<td>39</td>
<td>1.084</td>
</tr>
<tr>
<td>2.5</td>
<td>44</td>
<td>24</td>
<td>39</td>
<td>1.355</td>
</tr>
<tr>
<td>3.0</td>
<td>44</td>
<td>24</td>
<td>39</td>
<td>1.915</td>
</tr>
<tr>
<td>3.5</td>
<td>78</td>
<td>1</td>
<td>39</td>
<td>1.307</td>
</tr>
<tr>
<td>4.0</td>
<td>2</td>
<td>52</td>
<td>39</td>
<td>1.202</td>
</tr>
<tr>
<td>4.5</td>
<td>44</td>
<td>24</td>
<td>2</td>
<td>1.152</td>
</tr>
<tr>
<td>5.0</td>
<td>3</td>
<td>51</td>
<td>39</td>
<td>0.989</td>
</tr>
<tr>
<td>5.5</td>
<td>4</td>
<td>50</td>
<td>39</td>
<td>0.915</td>
</tr>
<tr>
<td>6.0</td>
<td>4</td>
<td>50</td>
<td>39</td>
<td>0.929</td>
</tr>
<tr>
<td>6.5</td>
<td>5</td>
<td>50</td>
<td>39</td>
<td>0.894</td>
</tr>
<tr>
<td>7.0</td>
<td>6</td>
<td>49</td>
<td>39</td>
<td>0.878</td>
</tr>
</tbody>
</table>
Concluding Remarks

• Simple approach.
• Open, closed, and multiclass QNs can be used at the SQN.
• SQNs can include non-software resources that are not mapped to hardware resources.
• HQNs are closed and can be multiclass.
• Any technique can be used to solve the SQN and HQN. This includes any known approximation to multiple-server devices, priorities, simultaneous resource possession, etc.