The queueing Package
Queueing Networks analysis with GNU Octave

Moreno Marzolla
marzolla@cs.unibo.it
http://www.moreno.marzolla.name/

Università di Bologna
december 4, 2012

Outline

Introduction
Markov chains
Single-station Queueing Systems
Closed Queueing Network Example
Open Queueing Network Example
Conclusions
What is queueing?

queueing is a software package for Queueing Network and Markov chain analysis written in GNU Octave, an interpreted language for numerical computations.

- queueing implements the most important numerical algorithms for QN analysis
  - Mean Value Analysis (MVA) for single and multiclass networks; convolution algorithm; performance bounds computation; ...
- queueing provides numerical algorithms for Markov chain analysis
  - Steady-state and transient state occupancy probabilities; mean time to absorption; mean residence times; time-averaged sojourn times; ...
- Free software (GPLv3+)

http://www.moreno.marzolla.name/software/queueing/

Installation

- Install from Octave-Forge
  octave> pkg install -local -forge queueing
  - If it doesn’t work, get the tarball from
    http://octave.sourceforge.net/queueing/ and install it
    octave> pkg install -local queueing-1.2.0.tar.gz
    - If it still doesn’t work, get the tarball from the URL above
      tar xfz queueing-1.2.0.tar.gz
      octave -p queueing/inst/
- Load the package at the Octave prompt
  octave> pkg load queueing
- Check if everything works
  octave> test qncsmva
  PASSES 9 out of 9 tests
Reliability Analysis of Multiprocessor Systems

There are $N = 2$ processors with individual Mean Time To Failure (MTTF) $1/\gamma$. States $n \in \{0, 1, 2\}$ denote that there are $n$ working processors. If one processor fails, it can be recovered (state $RC$) with probability $c$; recovery takes time $1/\beta$. When the system can not be recovered, a reboot is required (state $RB$), which brings down the entire system for time $1/\alpha > 1/\beta$. The mean time to repair a failed processor is $1/\delta$.
Reliability Analysis of Multiprocessor Systems

Mean Time Between Failures (MTBF)
The MTBF is the mean duration of continuous operation. The system starts in state 2; state RC is considered operational.

If we make states 0 and RB absorbing by removing all their outgoing transitions, the MTBF is the mean time to absorption of the new chain.

\[
\begin{align*}
& \text{RC} \\
& \beta \\
& \gamma \\
& \delta \\
& 2(1-c)\gamma \\
& \end{align*}
\]

\[
\text{# state space enumeration \{2, RC, RB, 1, 0\}}
\]

\[
Q(3,:)=Q(5,:)=0; \quad \# \text{make states \{0, RB\} absorbing}
\]

\[
p0=[1 \ 0 \ 0 \ 0 \ 0]; \quad \# \text{initial state occupancy prob.}
\]

\[
\text{MTBF = ctmcmtta(Q, p0)/60/60/24/365 \# MTBF (years)}
\]

\[
\Rightarrow 2.8376
\]
Example
Single Station Queueing Systems

Let us consider a $M/M/1$ system

- Arrival rate $\lambda = 0.3$ jobs/second
- Service rate $\mu = 0.4$ jobs/sec

We can compute the utilization $U$, response time $R$, average number of requests in the system $Q$ and throughput $X$ as follows:

```matlab
lambda = 0.3;
mu = 0.4;
[U R Q X] = qsmm1(lambda, mu);
=> U = 0.7500
    R = 10.0000
    Q = 3.0000
    X = 0.30000
```

Example
Single Station Queueing Systems

We can examine how the response time $R$ grows as the arrival rate $\lambda$ approaches the service rate $\mu = 0.4$

```matlab
mu = 0.4;
lambda = linspace(0.1,0.39,50);
R = zeros(size(lambda));
for i=1:length(lambda)
    [nc R(i)] = qsmm1(lambda(i),mu);
endfor
plot(lambda,R);
```
Example
Single Station Queueing Systems

M/M/1 queue (μ = 0.4)

Compute Farm

Simple closed model of a scientific computing cluster

- N independent jobs process data stored in a tape library.
- A disk-based cache is used to limit access to the (slow) tape library.
- A job reads data from disk; a cache miss happens with probability 1 – p and requires data to be copied from tape to disk before the job is allowed to proceed.
Closed Queueing Network Example

Compute Farm
Closed QN Model

Model Parameters

Mean CPU burst $Z = 1000\text{s}$, mean tape service time $S_2 = 200\text{s}$.

For the same amount of money we can buy:

- fast disks (costly, less disk space, lower cache hit ratio), or
- slow disks (cheap, more disk space, higher cache hit ratio).

<table>
<thead>
<tr>
<th>Case A: Slow disks</th>
<th>Case B: Fast disks</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Disk service time $S_1 = 1\text{s}$</td>
<td>▶ Disk service time $S_1 = 0.9\text{s}$</td>
</tr>
<tr>
<td>▶ Cache hit ratio $p = 0.9$</td>
<td>▶ Cache hit ratio $p = 0.8$</td>
</tr>
</tbody>
</table>
Compute Farm

Octave code

```octave
### Case A: slow disks ###
SA = [1 200]; pA = .9; PA = [pA 1-pA; 1 0];
VA = qnvisits(PA);
### Case B: fast disks ###
SB = [0.9 200]; pB = .8; PB = [pB 1-pB; 1 0];
VB = qnvisits(PB);
### Compute performance ###
Z = 1000;   # CPU burst length
NN = 1:100; # number of concurrent jobs
XA = XB = zeros(size(NN));
for n=NN
    [U R Q X] = qncsmva(n, SA, qnvisits(PA), 1, Z);
    XA(n) = X(1)/VA(1);
    [U R Q X] = qncsmva(n, SB, qnvisits(PB), 1, Z);
    XB(n) = X(1)/VB(1);
endfor
```

Compute Farm

Results

![Graph showing system throughput vs. number of concurrent jobs for slow and fast disks](image)
E-Commerce site

Open Queueing Network Model

- Dispatcher (center 1, service time $S_1 = 0.5$);
- Web Servers (centers 2–4, service time $S_2 = S_3 = S_4 = 0.8$);
- Database Servers (centers 5–6, service time $S_5 = S_6 = 1.8$);
- External Arrival at center 1 with rate $\lambda = 0.9\text{jobs/s}$
- Uniform routing of requests, $p_{\text{exit}} = 0.5$

Routing Matrix

```
P = \begin{pmatrix}
0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 \\
0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\
0 & 1/3 & 1/3 & 1/3 & 0 & 0 
\end{pmatrix}
```
E-commerce example

Model Definition

```matlab
ws = 2:4; db = 5:6; p_exit = 0.5;
K = 1 + length(ws) + length(db); ## n. of service centers
S = lambda = zeros(1,K); P = zeros(K,K);
S(1) = 0.5; ## Service time at the dispatcher
S(ws) = 0.8; ## Service time at the Web Servers
S(db) = 1.8; ## Service time at the DB servers
lambda(1) = 0.9; ## Arrival rate
P(1,ws) = 1/length(ws);
P(ws,db) = (1-p_exit)/length(db);
P(db,ws) = 1/length(ws);
V = qnvisits(P,lambda);
[U R Q X] = qnos(sum(lambda),S,V)
printf("System Throughput..... %f\n", X(1) / V(1));
printf("System Response Time.. %f\n", dot(R,V));
```

E-commerce example

Results

\[
U = \\
\begin{bmatrix}
0.45000 & 0.48000 & 0.48000 & 0.48000 & 0.81000 & 0.81000 \\
\end{bmatrix}
\]

\[
R = \\
\begin{bmatrix}
0.90909 & 1.53846 & 1.53846 & 1.53846 & 9.47368 & 9.47368 \\
\end{bmatrix}
\]

\[
Q = \\
\begin{bmatrix}
0.81818 & 0.92308 & 0.92308 & 0.92308 & 4.26316 & 4.26316 \\
\end{bmatrix}
\]

\[
X = \\
\begin{bmatrix}
0.90000 & 0.60000 & 0.60000 & 0.60000 & 0.45000 & 0.45000 \\
\end{bmatrix}
\]

System Throughput..... 0.900000
System Response Time.. 13.459698
Conclusions

Queueing networks are simple yet powerful tools for performance modeling and capacity planning. Useful insights can be obtained with very little effort.

Suggested reading:

http://www.moreno.marzolla.name/software/queueing/