Formal Methods Project for System Verification Project

A dynamic server allocation for energy efficiency

Fabio Dainese (857661), Martina Donadi (865737)

Ca' Foscari, Univeristy of Venice

December 3, 2019

Outline

Introduction

Power consumption in data centres receive a huge concern by data centre providers.

- Find a trade-off between performance cost and saving energy;
- Model a (high/low) policy to control dynamically the powering on and off of the servers.

The system can basically be regarded as a multi-server queue.

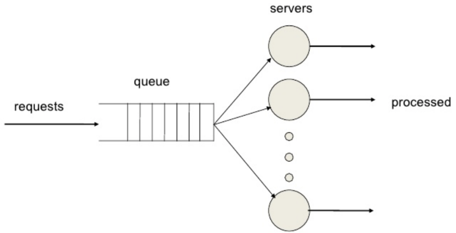

Figure: Multi server queue

- N servers of which M < N static servers (always on);
- Alternating high/low job arrival rate.

Components and Activities of the System

- Q_i: Queue's current state, with $0 \leq i \leq n$, which represents the number of jobs in the system at the time i;
- $Arrival_{high}$, $Arrival_{low}$: Represent the two possible jobs' arrival stream states (high or low);

- All the possible server's state are represented by:
  - $Server_{on}$: State in which the server is either active or idle;
  - $ServerPowering_{on}$: State in which the server is turning on;
  - $Server_{off}$: State in which the server is fully turned off;
  - $ServerPowering_{off}$: State in which the server is turning off;
  - $Server_{failOn}$: State in which the server encountered an error performing the power-up activity;
  - $Server_{failOff}$: State in which the server encountered an error performing the shutdown activity;
  - $Server_{static}$: State representing a server that is always active (M servers).

The set of possible actions that the *queue* component can perform are:

- **Service**: When a request is successfully elaborated and the job leaves the system (at a fixed rate of μ);
- **arrivalH**: When the arrival into the system occurs at high rate λ;
- **arrivalL**: When the arrival into the system occurs at low rate ϵ.

The activities of the *arrival stream*, identified by the components $Arrival_{high}$ and $Arrival_{low}$, are modeled by the previously defined queue activities, i.e. **arrivalH** and **arrivalL**, and also by the switching actions between the high and low rate defined through the activities:

- **highPeriodEnd** at rate β;
- **lowPeriodEnd** at rate γ.

The activities of the server are modeled by the previously defined activities **highPeriodEnd**, **lowPeriodEnd**, **service** and by new ones, such as:

- **powerup**: Used to turn on a server (η rate);
- **poweroff**: Used to turn off a server (ξ rate);
- **repair**: Used to fix the server in case of error/fault ($\sigma rate$);

- **Queue** will accept the incoming jobs, regardless the type of arrival, up until the queue if full ($i \leq n$), plus every time a job is served the queue index counter is decreased by 1;
- **Arrival Stream** will be in charge of detecting and switching accordingly to the appropriate arrival rate stream situation (high or low);
- **Server** will power-up or down accordingly to the high or low jobs arrival. To keep in mind that we have considered also M static servers.

PEPA Components of the System

$$
\begin{aligned}
Q_0 &\stackrel{\text{def}}{=} (arrivalH, \lambda).Q_1 + (arrivalL, \epsilon).Q_1 \\
Q_1 &\stackrel{\text{def}}{=} (arrivalH, \lambda).Q_2 + (arrivalL, \epsilon).Q_2 + (service, \mu).Q_1 \\
&\vdots \\
Q_i &\stackrel{\text{def}}{=} (arrivalH, \lambda).Q_{i+1} + (arrivalL, \epsilon).Q_{i+1} + (service, \mu).Q_{i-1} \\
&\vdots \\
Q_n &\stackrel{\text{def}}{=} (service, \mu).Q_{n-1}
\end{aligned}
$$

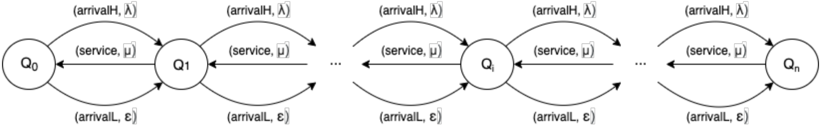

Figure: Queue PEPA derivation graph

$$Arrival_{high} \stackrel{\text{def}}{=} (arrivalH, \lambda).Arrival_{high} + (highPeriodEnd, \beta).Arrival_{low}$$
$$Arrival_{low} \stackrel{\text{def}}{=} (arrivalL, \epsilon).Arrival_{low} + (lowPeriodEnd, \gamma).Arrival_{high}$$

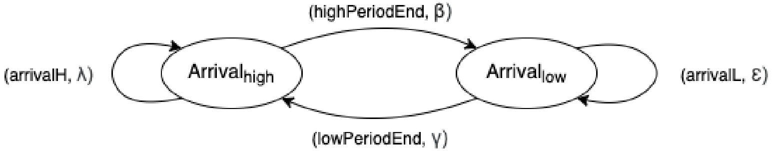

Figure: Arrival Stream PEPA derivation graph

$$
\begin{aligned}
ServerPowering_{on} &\stackrel{\text{def}}{=} (powerup, \eta * (1-\rho)).Server_{on} + (powerup, \eta * \rho).Server_{failOn} \\
Server_{on} &\stackrel{\text{def}}{=} (service, \mu).Server_{on} + (highPeriodEnd, \beta).ServerPowering_{off} \\
ServerPowering_{off} &\stackrel{\text{def}}{=} (poweroff, \xi * (1-\rho)).Server_{off} + (poweroff, \xi * \rho).Server_{failOff} \\
Server_{off} &\stackrel{\text{def}}{=} (lowPeriodEnd, \gamma).ServerPowering_{on}
\end{aligned}
$$

$$
\begin{aligned}
Server_{static} &\stackrel{\text{def}}{=} (service, \mu).Server_{static} \\
Server_{failOn} &\stackrel{\text{def}}{=} (repair, \sigma).Server_{on} \\
Server_{failOff} &\stackrel{\text{def}}{=} (repair, \sigma).Server_{off}
\end{aligned}
$$

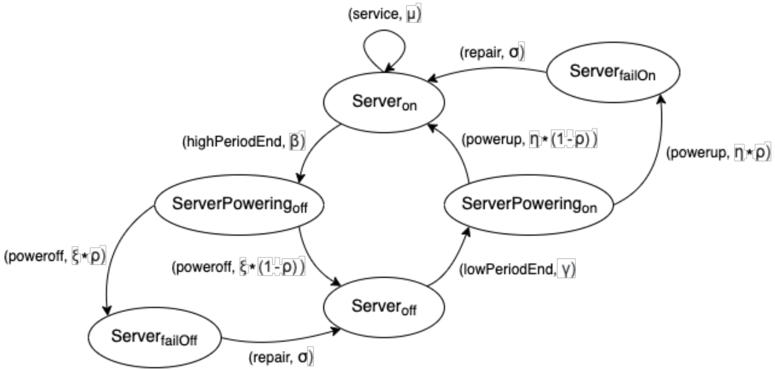

Figure: Server PEPA derivation graph

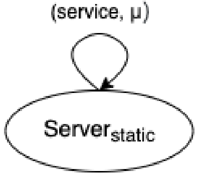

Figure: Static server PEPA derivation graph

Entire System Expressed in PEPA

Entire System Expressed in PEPA

$$System \stackrel{\text{def}}{=} ((Arrival_{high} \underset{K}{\bowtie} Server_{on} \underset{K}{\bowtie} \ldots \underset{K}{\bowtie} Server_{on}) \underset{Z}{\bowtie} Server_{static}[M]) \underset{L}{\bowtie} Q_0$$

Where respectively the sets K, Z, L are defined as follows:

- $K = \{highPeriodEnd\}$;
- $Z = \emptyset$;
- $L = \emptyset$.

Derivation Graph of the System

$$Arrival_{high} \underset{\{highPeriodEnd\}}{\bowtie} Server_{on}$$

12 states

1	Arrival_high	Server_on	0.7511560761484473
2	Arrival_low	ServerPowering_off	0.03755780380742236
3	Arrival_high	ServerPowering_off	0.037557803807422366
4	Arrival_low	Server_off	0.017839956808525625
5	Arrival_low	Server_failOff	0.0018778901903711179
6	Arrival_high	Server_off	0.0572756508063191
7	Arrival_high	Server_failOff	0.005633670571113353
8	Arrival_low	ServerPowering_on	0.00891997840426281
9	Arrival_high	ServerPowering_on	0.0661956292105819
10	Arrival_low	Server_on	0.00847397948404967
11	Arrival_low	Server_failOn	4.459989202131406E-4
12	Arrival_high	Server_failOn	0.007065561841271393

Figure: Space State View for the system equation - PEPA Eclipse plug-in

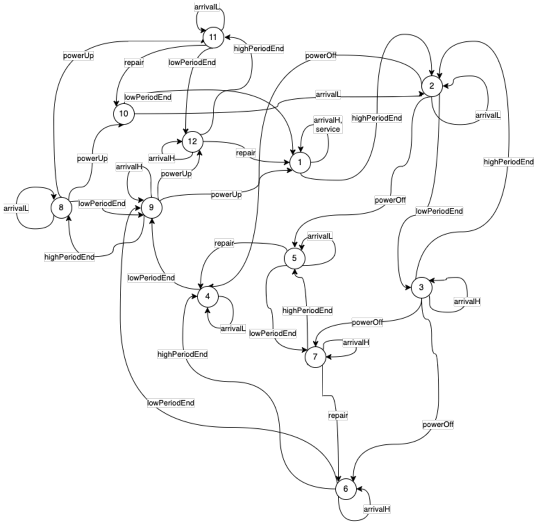

Figure: System derivation graph

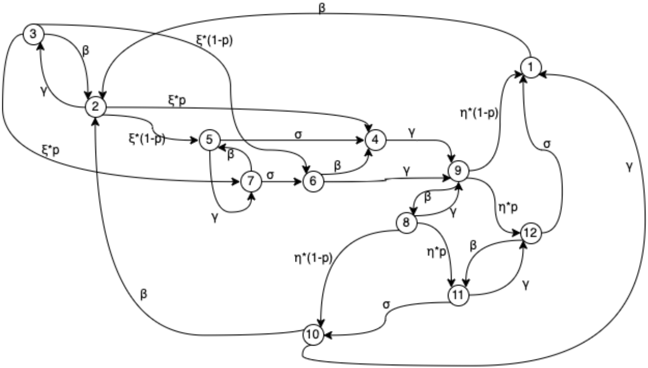

Figure: CTMC representation - State transition diagram

Evaluation

Different types of performance measures can be derived from the steady state distribution of a Markov Process:

- State-based measures : correspond to the probability that a model is in a state (e.g. utilisation);
- Rate-based measures : those which correspond to the predicted rate at which some event occur (e.g. throughput);
- Response-time.

When the model is in steady state we have:

- the total probability flux out of each state is equal to the total probability flux into the state;
- π_i is the probability that the model is in state x_i
- $\sum_{x_1 \in S} \pi_i = 1$ for π_i is a probability distribution.

The collection of the equations expressing the steady state condition for each state x_i of the model is resumed in the Global Balance equation:

$$\pi Q = 0$$

where Q is the Infinitesimal Generator matrix.

$$
\begin{cases}
\pi_1 \cdot \beta = \pi_9 \cdot (\eta \cdot (1-\rho)) + \pi_{10} \cdot \gamma + \pi_{12} \cdot \sigma \\
\pi_2 \cdot (\gamma + \xi) = \pi_1 \cdot \beta + \pi_3 \cdot \beta + \pi_{10} \cdot \beta \\
\pi_3 \cdot (\beta + \xi) = \pi_2 \cdot \gamma \\
\pi_4 \cdot \gamma = \pi_2 \cdot (\xi \cdot (1-p)) + \pi_5 \cdot \sigma + \pi_6 \cdot \beta \\
\pi_5 \cdot (\sigma + \gamma) = \pi_2 \cdot (\xi \cdot \rho) + \pi_7 \cdot \beta \\
\pi_6 \cdot (\beta + \gamma) = \pi_3 \cdot (\xi \cdot (1-\rho)) + \pi_7 \cdot \sigma \\
\pi_7 \cdot (\beta + \sigma) = \pi_3 \cdot (\xi \cdot \rho) + \pi_5 \cdot \gamma \\
\pi_8 \cdot (\gamma + \eta) = \pi_9 \cdot \beta \\
\pi_9 \cdot (\beta + \eta) = \pi_4 \cdot \gamma + \pi_6 \cdot \gamma + \pi_8 \cdot \gamma \\
\pi_{10} \cdot (\beta + \gamma) = \pi_8 \cdot (\xi \cdot (1-\rho)) + \pi_{11} \cdot \sigma \\
\pi_{11} \cdot (\sigma + \gamma) = \pi_8 \cdot (\xi \cdot \rho) + \pi_{12} \cdot \beta \\
\pi_{12} \cdot (\sigma + \beta) = \pi_9 \cdot (\xi \cdot \rho) + \pi_{11} \cdot \gamma \\
\pi_1 + \pi_2 + \pi_3 + \pi_4 + \pi_5 + \pi_6 + \pi_7 + \pi_8 + \pi_9 + \pi_{10} + \pi_{11} + \pi_{12} = 1
\end{cases}
$$

Figure: Global balance system equations

$$Q = \begin{bmatrix}
-\beta & \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -(\gamma+\xi) & \gamma & \xi*(1-\rho) & \xi*\rho & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \beta & -(\beta+\xi) & 0 & 0 & \xi*(1-\rho) & \xi*\rho & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\gamma & 0 & 0 & 0 & 0 & \gamma & 0 & 0 & 0 \\
0 & 0 & 0 & \sigma & -(\sigma+\gamma) & 0 & \gamma & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \beta & 0 & -(\beta+\gamma) & 0 & 0 & \gamma & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \beta & \sigma & -(\beta+\sigma) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -(\gamma+\eta) & \gamma & \eta*(1-\rho) & \eta*\rho & 0 \\
\eta*(1-\rho) & 0 & 0 & 0 & 0 & 0 & 0 & \beta & -(\beta+\eta) & 0 & 0 & \eta*\rho \\
\gamma & \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -(\beta+\gamma) & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma & -(\sigma+\gamma) & \gamma \\
\sigma & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta & -(\sigma+\beta)
\end{bmatrix}$$

Figure: Infinitesimal generator matrix

Exit Rate and Sojourn Time

Exit rate and sojourn time are two related measures:

- Exit rate: $q_i = \sum_{x_j \in S, j \neq i} q_{ij}$
- Sojourn time: $1/q_i$.

The Sojourn time in a state x_i is exponentially distributed with parameter q_i

$$
\begin{aligned}
q(1) &= \beta \\
q(2) &= \gamma + \xi \\
q(3) &= \beta + \xi \\
q(4) &= \gamma \\
q(5) &= \sigma + \gamma \\
q(6) &= \beta + \gamma \\
q(7) &= \beta + \sigma \\
q(8) &= \sigma + \eta \\
q(9) &= \beta + \eta \\
q(10) &= \beta + \gamma \\
q(11) &= \sigma + \gamma \\
q(12) &= \sigma + \beta
\end{aligned}
$$

Figure: Exit rates

$$ST(1) = \frac{1}{\beta}$$
$$ST(2) = \frac{1}{\gamma + \xi}$$
$$ST(3) = \frac{1}{\beta + \xi}$$
$$ST(4) = \frac{1}{\gamma}$$
$$ST(5) = \frac{1}{\sigma + \gamma}$$
$$ST(6) = \frac{1}{\beta + \gamma}$$
$$ST(7) = \frac{1}{\beta + \sigma}$$
$$ST(8) = \frac{1}{\sigma + \eta}$$
$$ST(9) = \frac{1}{\beta + \eta}$$
$$ST(10) = \frac{1}{\beta + \gamma}$$
$$ST(11) = \frac{1}{\sigma + \gamma}$$
$$ST(12) = \frac{1}{\sigma + \beta}$$

Figure: Sojourn times

Utilization

The *utilisation* measure is the total probability that the model is in one of the states in which the resource is in use.

$$U_{Arrival_{high}} = \pi_1 + \pi_3 + \pi_6 + \pi_7 + \pi_9 + \pi_{12}$$

$$U_{Server_{on}} = \pi_1 + \pi_{10}$$

Evaluation of the System Using the PEPA Eclipse Plug-In

```
l = 50.0;
e = 10.0;
m = 10.0;
b = 10.0;
g = 10.0;
n = 100.0;
r = 0.1;
x = 100.0;
s = 1.0;

Q0 = (arrivalH,l).Q1 + (arrivalL,e).Q1;
Q1 = (arrivalH,l).Q2 + (arrivalL,e).Q2 + (service,m).Q0;
Q2 = (service,m).Q1;

Arrival_high = (arrivalH,l).Arrival_high + (highPeriodEnd,b).Arrival_low;
Arrival_low = (arrivalL,e).Arrival_low + (lowPeriodEnd,g).Arrival_high;

ServerPowering_on = (powerup, n * (1-r)).Server_on + (powerup, n*r).Server_failOn;
Server_on = (service,m).Server_on + (highPeriodEnd,b).ServerPowering_off;
ServerPowering_off = (poweroff,x*(1-r)).Server_off + (poweroff,x*r).Server_failOff;
Server_off = (lowPeriodEnd,g).ServerPowering_on;

Server_static = (service,m).Server_static;
Server_failOn = (repair,s).Server_on;
Server_failOff = (repair,s).Server_off;

((Arrival_high <highPeriodEnd> Server_on <highPeriodEnd> Server_on <highPeriodEnd> Server_on) <>
    Server_static <> Server_static <> Server_static ) <> Q0
```

Figure: Code of the system expressed using PEPA Eclipse plug-in

To point out that we have modeled a system having $N = 6$ servers, which 3 of them are *dynamic* and the rest are *static* (i.e. always on).
For complexity reasons the queue capacity has been bounded at 2.

Finally the various rates were set as follows:

- **High arrival rate**: 50;
- **Low arrival rate**: 10;
- **Powering-up and down**: 100 in equally *high* and *low* period of job arrivals.

Utilisation | **Throughput** | Population

Action	Throughput
arrivalH	52.08186743082742
arrivalL	2.839440467322871 3
highPeriodEnd	1.2115334905786868
lowPeriodEnd	4.84613396231475
poweroff	3.634600471736062
powerup	3.6346004717360594
repair	0.7269200943472127
service	58.13672035090973 4

Figure: Throughput evaluation of the system

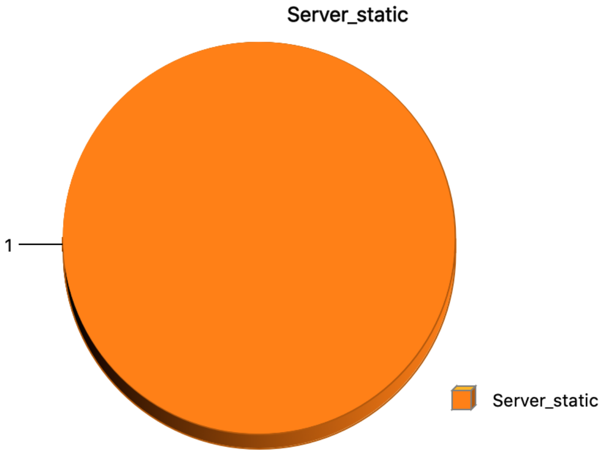

Figure: Utilization of the Static Server component

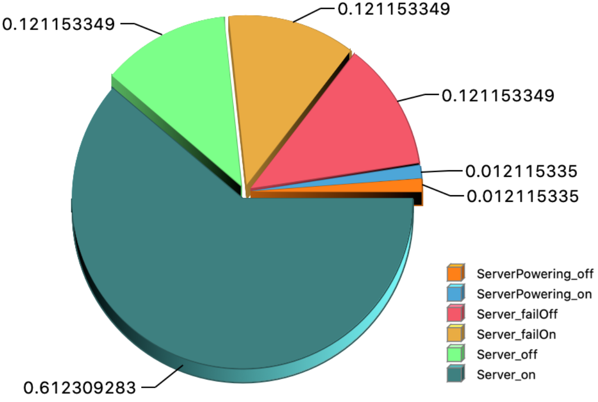

Figure: Utilisation of the Dynamic Server component

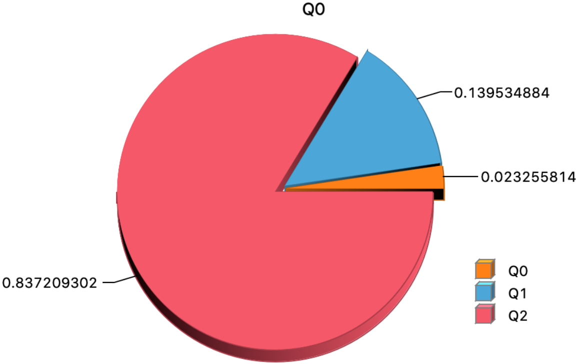

Figure: Utilisation of the Queue component