Computer Science and Information Technology 2(4): 211-218, 2014 http://www.hrpub.org
DOI: 10.13189/csit.2014.020405

Proposed Pricing Model for Cloud Computing

Muhammad Adeel Javaid

Member Vendor Advisory Council, CompTIA
Corresponding Author: ajaviad@gmail.com

Abstract Cloud computing is an emerging technology of business computing and it is becoming a development trend. The process of entering into the cloud is generally in the form of queue, so that each user needs to wait until the current user is being served. In the system, each Cloud Computing User (CCU) requests Cloud Computing Service Provider (CCSP) to use the resources, if CCU(cloud computing user) finds that the server is busy then the user has to wait till the current user completes the job which leads to more queue length and increased waiting time. So to solve this problem, it is the work of CCSP's to provide service to users with less waiting time otherwise there is a chance that the user might be leaving from queue. CCSP's can use multiple servers for reducing queue length and waiting time. In this paper, we have shown how the multiple servers can reduce the mean queue length and waiting time. Our approach is to treat a multiserver system as an M/M/m queuing model, such that a profit maximization model could be worked out.

Keywords Cloud Pricing, Cloud Pricing Model, Cloud Multi-Server Model

1. Scope

Two server speed and power consumption models are considered, namely, the idle-speed model and the constant-speed model. The probability density function of the waiting time of a newly arrived service request is derived. The expected service charge to a service request is calculated. To the best of our knowledge, there has been no similar investigation in the literature, although the method of optimal multicore server processor configuration has been employed for other purposes, such as managing the power and performance tradeoff.

2. Existing System

To increase the revenue of business, a service provider can construct and configure a multiserver system with many servers of high speed. Since the actual service time (i.e., the task response time) contains task waiting time and task execution time so more servers reduce the waiting time and faster servers reduce both waiting time and execution time.

2.1. Problems on Existing System

1. In single Sever System if doing one job another process waiting for another completion of server service, So it take time to late (See Figure-1).
2. Due to increase the Service cost of cloud.

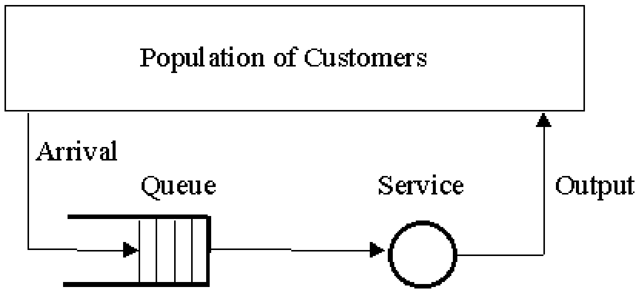


Figure 1. Single Server Queuing System

3. Proposed System

We study the problem of optimal multiserver configuration for profit maximization in a cloud computing environment. Our approach is to treat a multiserver system as an M/M/m queuing model, such that our optimization problem can be formulated and solved analytically (See Figure-2).

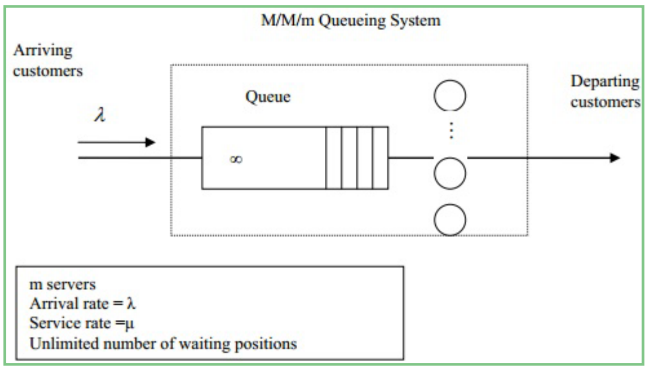


Figure 2. M/M/m Queuing System

Figure 3 and 4 below explain the basic queuing system design and examples of the negative exponential distribution for service times. The estimate of average service time for different customers and utilization of system could be easily worked out by the formulas given below.

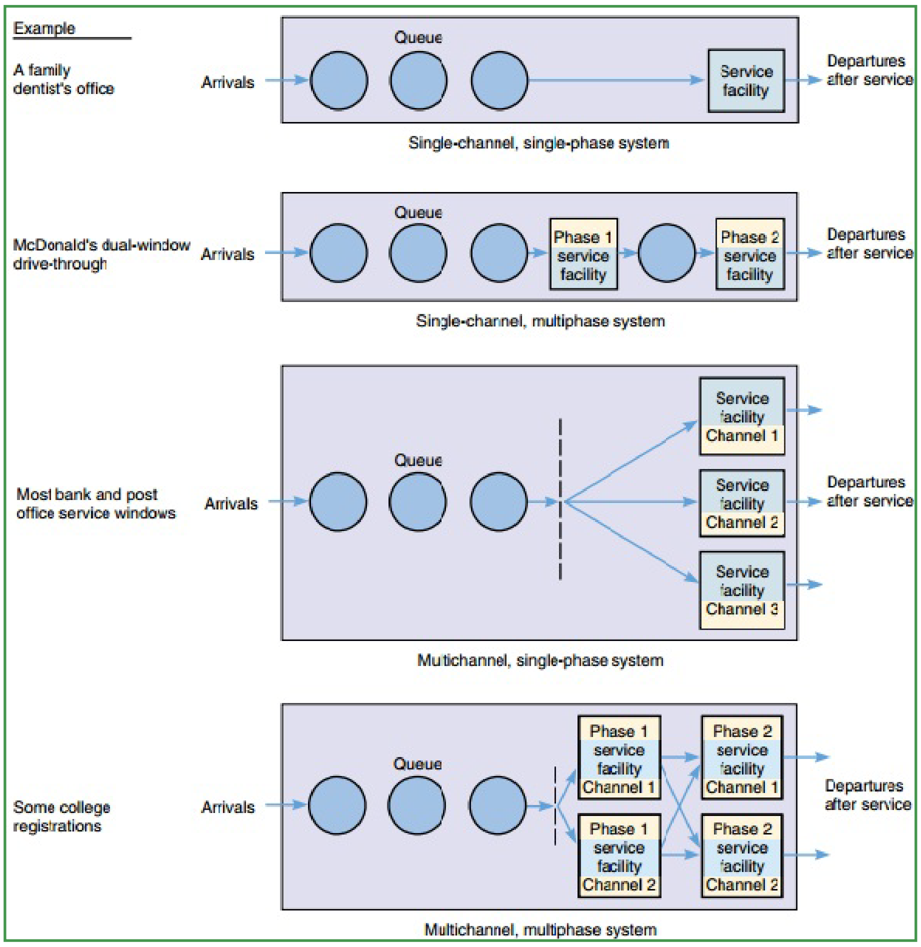

Example
A family dentist's office
Arrivals
Queue
Service facility
Departures after service
Single-channel, single-phase system
McDonald's dual-window drive-through
Arrivals
Queue
Phase 1 service facility
Phase 2 service facility
Departures after service
Single-channel, multiphase system
Most bank and post office service windows
Arrivals
Queue
Service facility Channel 1
Service facility Channel 2
Service facility Channel 3
Departures after service
Multichannel, single-phase system
Some college registrations
Arrivals
Queue
Phase 1 service facility Channel 1
Phase 1 service facility Channel 2
Phase 2 service facility Channel 1
Phase 2 service facility Channel 2
Departures after service
Multichannel, multiphase system

Figure 3. Basic Queuing System Designs

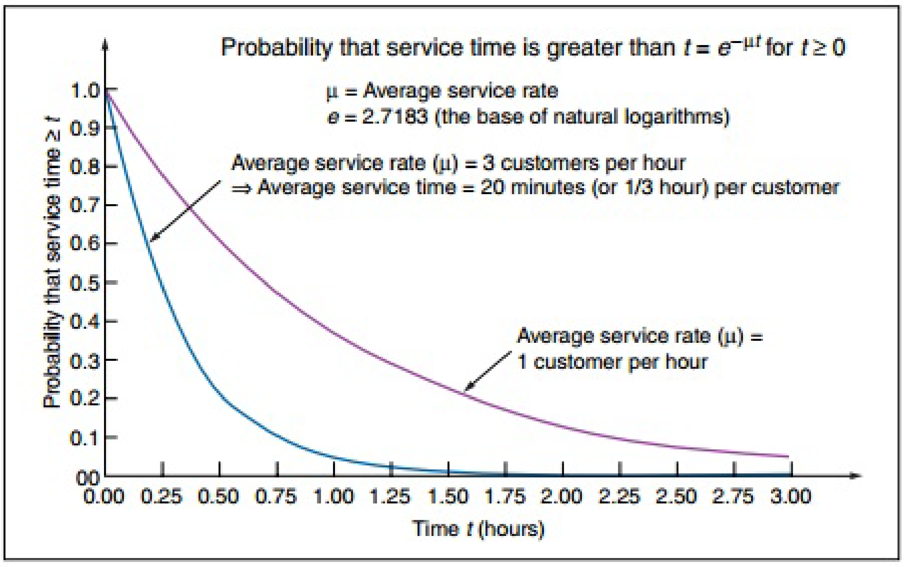


Figure 4. Two Examples of the Negative Exponential Distribution for Service Times

By using the following formula the utilization of system and waiting time spent in the queue by customer could be easily worked out.

$$N = N_q + N_s$$

$$T = W + \overline{x}$$

$$\overline{x} = \frac{1}{\mu} \quad \rho = \frac{\lambda}{\mu}$$, offered load(offered traffic)

$$P_k = \begin{cases} \frac{\rho^k}{k!} \cdot P_0 & k < m \\ \frac{\rho^m}{m!} (\frac{\rho}{m})^{k-m} \cdot P_0 & k \geq m \end{cases}$$

$$p_0 = \frac{1}{\sum_{k=0}^{m-1} \frac{\rho^k}{k!} + \frac{\rho^m}{m!} \cdot \frac{m}{m-\rho}}$$

$$D_m(\rho) = \frac{\rho^m}{m!} \cdot \frac{m}{m-\rho} \cdot p_0$$

(Erlangs second formula=the probability that all servers are busy)

$$N = \rho + D_m(\rho) \cdot \frac{\rho}{m-\rho}$$

$$N_q = D_m(\rho) \cdot \frac{\rho}{m-\rho}$$

Where the notation of the above formula is as given below:

Notation	Description
N	Average number of customers in the system , $N = N_q + N_s$
N_q	Average number of customers in the queue
N_s	Average number of customers in the service facilities
$\tilde{x}$	Random variable which describes time spent in the service facility by a customer
$\overline{x}$	Average service time for a customer, $\overline{x} = E(\tilde{x})$
$\tilde{w}$	Random variable which describes time spent in the waiting queue by a customer
W	Average waiting time spent in the queue by a customer $W = E(\tilde{w})$
$\tilde{s}$	Random variable which describes time spent in the system by a customer; $\tilde{s} = \tilde{x} + \tilde{w}$
T	Average time spent in the system by a customer $T = E(\tilde{s})$, $T = W + \overline{x}$
λ	Arrival rate
λ_{eff}	The effective arrival rate
μ	Service rate
ρ	$\rho = \frac{\lambda}{\mu}$, offered load (offered traffic)
p_k	Stationary probabilities; p_k is the probability that there are k customers in the system

The average number of customers in a queue is given by:

$$N_q = 0 \cdot p_m + 1 \cdot p_{m+1} + 2 \cdot p_{m+2} + 3 \cdot p_{m+3} + ..$$

$$N_q = 0 \cdot p_m + 1 \cdot p_m \frac{\rho}{m} + 2 \cdot p_m \left(\frac{\rho}{m}\right)^2 + 3 \cdot p_m \left(\frac{\rho}{m}\right)^3 + 4 \cdot p_m \left(\frac{\rho}{m}\right)^4 + \ldots$$

$$N_q = p_m \sum_{k=0}^{\infty} k \left(\frac{\rho}{m}\right)^k \overset{(*)}{=} p_m \frac{\frac{\rho}{m}}{\left(1-\frac{\rho}{m}\right)^2} = p_0 \frac{\rho^m}{m!} \frac{\frac{\rho}{m}}{\left(1-\frac{\rho}{m}\right)^2}$$

To get (*) we can denote $\frac{\rho}{m} = x$ and differentiate the geometric series:

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \Rightarrow \quad (\textit{differentiate})$$

$$\sum_{k=0}^{\infty} k x^{k-1} = \frac{1}{(1-x)^2} \Rightarrow \quad (\text{multiply by x})$$

$$\sum_{k=0}^{\infty} k x^k = \frac{x}{(1-x)^2}$$

We can even rewrite the last expression using Erlang's second formula:

$$D_m = p_0 \frac{\rho^m}{m!} \frac{1}{\left(1-\frac{\rho}{m}\right)}$$

$$N_q = D_m \frac{\frac{\rho}{m}}{1-\frac{\rho}{m}}$$

The average number of customers in the system is the sum:

$$N = N_s + N_q$$

The average number of customers in the service facilities for an M/M/m system is given by:

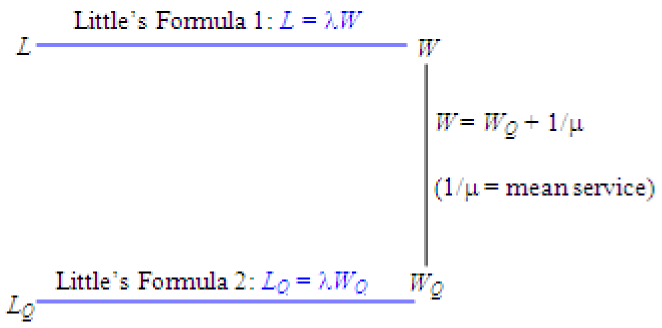
$$N_s = \lambda \cdot \bar{x} = \frac{\lambda}{\mu} = \rho \qquad (*)$$

3.1. Fundamental Measures of Cost

Each of the following fundamental quantities represents a way to measure the "cost" of queuing in the long run. Their interrelationship will be spelled out later on.
L_Q = Average **q**ueue **l**ength (not including customers that are being served)
L= Average population
= Average number of customers in the system
// L_Q and L are time-averages, i.e.,
// averages over all nanoseconds until ∞.
W_Q = Average **q**ueuing time of a customer
W = Average delay of a customer
// W_Q and W are averages over customers
= W_Q + 1/μ // Delay = queuing + a service time
// 1/μ = mean service time in the formula $W = W_Q + 1/\mu$.

// Later when μ sometimes stands for ave. output rate,
// then $W = W_Q + 1/\mu$ is only for a single-server system.

We now establish two more relations among these four measures of cost so that any of them determines all others:


Little's Formula 1: $L = \lambda W$
L
W
$W = W_Q + 1/\mu$
$(1/\mu = \text{mean service})$
Little's Formula 2: $L_Q = \lambda W_Q$
L_Q
W_Q

Little's Formula 1. $L = \lambda W$ for **all** stationary queuing models.
// **All** stationary models regardless of the arrival process, service-
// time distribution, number of servers, and queuing discipline

Proof. Think of a measure of the cost as **money that customers pay to the system**. In this proof, let each customer **in the system** pay to the system at the rate of \$1 per unit time.

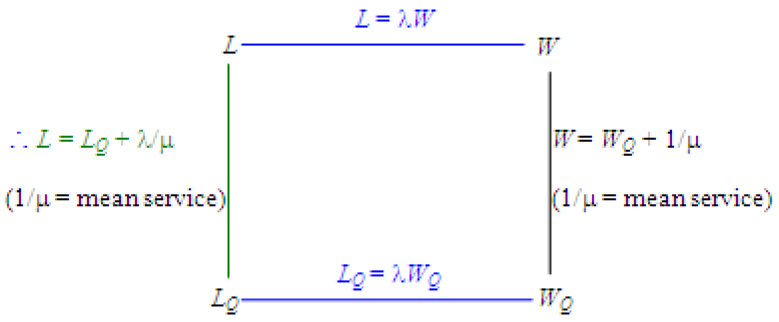
L = Time-average "rate" at which the system earns
// Unit of this "rate" = \$/unit time
= $\lambda \cdot$ Average payment per customer when **in the system**
// Unit of this amount = \$/person
// time-ave $\rightarrow$ ave over customers
= λW // Unit = (person/unit time)*(\$/person) = \$/unit time

Little's Formula 2. $L_Q = \lambda W_Q$ for **all** stationary queuing models.

Proof. Let each customer **in queue** pay \$1 per unit time to the system.

L_Q = Time-average "rate" at which the system earns
= $\lambda \cdot$ Average amount a customer pays **in queue**
= λW_Q

In conclusion,


$L = \lambda W$
L
W
$\therefore L = L_Q + \lambda/\mu$
$(1/\mu = \text{mean service})$
$W = W_Q + 1/\mu$
$(1/\mu = \text{mean service})$
$L_Q = \lambda W_Q$
L_Q
W_Q

Now we can calculate the expected service charge to a service request. Based on these results, we get the expected net business gain in given unit of time.

3.2. Mechanisms

Multiple sporadic servers as a mechanism for rescheduling aperiodic tasks are applicable to today's computer environments. A developed simulation tool enables evaluation of its performance for various task sets and server parameters (See Figure-5 below). By increasing the number of servers, aperiodic task response time is reduced; system utilization and the number of reschedulings are increased whereas periodic task execution is disrupted insignificantly. Proper selection of server parameters improves task response time, and decreases the number of unnecessary reschedulings. Simulation results prove model correctness and simulation accuracy. The simulator is applicable to developing rescheduling algorithms and their implementation into real environments.

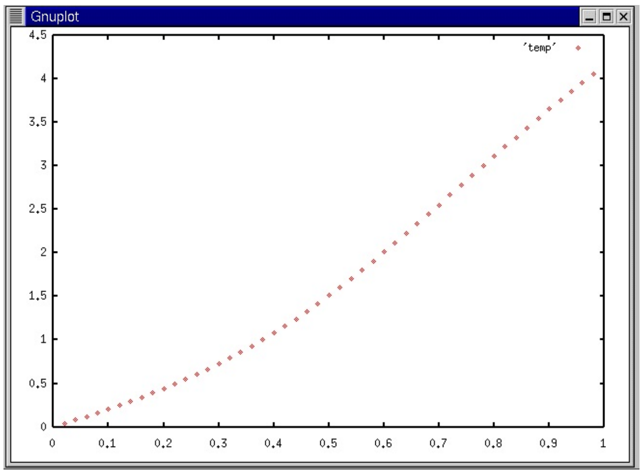


Figure 5. Simulation Result Obtained

3.3. Simulation Code

The following source code will provide you a tool to perform simulation in an M/M/m environment with finite number of customers.

```
#include <stdio.h>
#include <stdlib.h> // Needed for rand() and RAND_MAX
#include <math.h> // Needed for log()
#include <iostream.h>

//----- Constants
------------------------------------------------------------
#define SIM_TIME 1.0e7 // Simulation time

//----- Function prototypes
--------------------------------------------------
double expntl(double x); // Generate exponential RV with mean x
double general();

/********************** Main program*****************************/
void main(void)
{
for(int i=1;i<=49;i++)
{
double end_time = SIM_TIME; // Total time to simulate
double Ta ; // Mean time between arrivals
double Ts = 2.0; // Mean service time
int m=2; //no of servers
double time = 0.0; // Simulation time
double t1 = 0.0; // Time for next event #1 (arrival)
double t2 = SIM_TIME; // Time for next event #2 (departure)
long int n = 0; // Number of customers in the system
long int k=8; //buffer space
float p;
double c = 0.0; // Number of service completions
double s = 0.0; // Area of number of customers in system
double tn = time; // Variable for "last event time"
double x; // Throughput
double l; // Mean number in the system
double w; // Mean residence time

p=0.02*i; Ta=Ts/(m*p);

// Main simulation loop
while (time < end_time)
{
if (t1 < t2) // *** Event #1 (arrival)
{
time = t1;
s = s + n * (time - tn); // Update area under "s" curve
if(n<k) n++;
tn = time; // tn = "last event time" for next event
t1 = time + expntl(Ta);
if (n == 1)
t2 = time + expntl(Ts);

}
else // *** Event #2 (departure)
{
time = t2;
s = s + n * (time - tn); // Update area under "s" curve
if(n>m)
n-=m; else n=0;
tn = time; // tn = "last event time" for next event
c++; // Increment number of completions
if (n > 0)
t2 = time + expntl(Ts);
else
t2 = end_time;

}
}

x = c / time; // Compute throughput rate
l = s / time; // Compute mean number in system
w = l / x; // Compute mean residence or system time
if(l>0)
cout<<p<<"\t"<<l<<endl;
}


}

/*********************************************
************************
Function to generate exponentially distributed RVs
Input: x (mean value of distribution)
Output: Returns with exponential RV
**********************************************
************************/
double expntl(double x)
{
double z; // Uniform random number from 0 to 1

// Pull a uniform RV (0 < z < 1)
do
{
z = ((double) rand() / RAND_MAX);
}
while ((z == 0) || (z == 1));

return(-x * log(z));
}
```

3.4. Implementation

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective (See Figure-6).

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

4. Conclusion

This paper proposes a novel pricing demand scheme designed for a cloud based environment that offers querying services and aims at the maximization of the cloud profit with predictive demand price solution on economic way of user profit. The proposed solution allows: on one hand, long-term profit maximization with price minimization on request of same demand, and, on the other, dynamic calibration to the actual behavior of the cloud application.

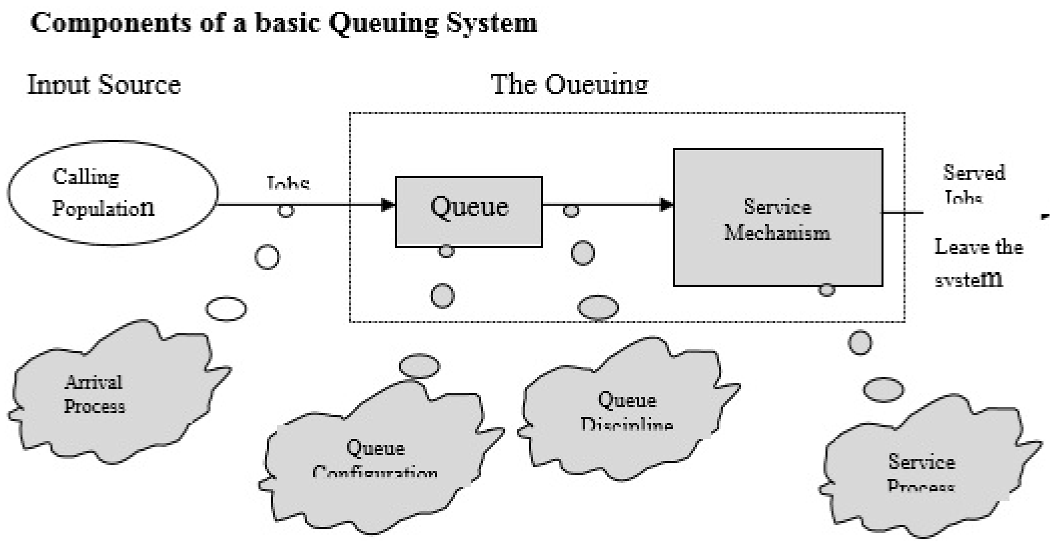


Figure 6. Components of a Queuing System

REFERENCES

[1] M. Armbrust, et al., “Above the clouds: a Berkeley view of cloud computing,” Technical Report No. UCB/EECS-2009-28, February 2009

[2] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, “Economic models for resource management and scheduling in grid computing,” Concurrency and Computation: Practice and Experience, vol. 14, pp. 1507-1542, 2007.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, 2009.

[4] P. Chandrakasan, S. Sheng, and R. W. Brodersen, “Low-power CMOS digital design,” IEEE Journal on Solid-State Cir- cuits, vol. 27, no. 4, pp. 473-484, 1992.

[5] N. Chun and D. E. Culler, “User-centric performance analysis of market-based cluster batch schedulers,” Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002.

[6] Durkee, “Why cloud computing will never be free,” Communications of the ACM, vol. 53, no. 5, pp. 62-69, 2010.

[7] K. Hwang, G. C. Fox, and J. J. Dongarra, Distributed and Cloud Computing, Morgan Kaufmann, 2012.

[8] Intel, Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor – White Paper, March 2004.

[9] P. Mell and T. Grance, “The NIST definition of cloud comput- ing,” National Institute of Standards and Technology, 2009.