Message Queuing Model for a Healthcare Hybrid Cloud Computing Platform

Roxana MARCU, Iulian DANILA, Dan POPESCU, Oana CHENARU, Loretta ICHIM

University Politehnica Bucharest, Splaiul Independentei no.313, Bucharest, 060042, Bucharest, Romania.
roxana.marcu@cti.pub.ro; iulian.danila@gmail.com; dan_popescu_2002@yahoo.com

Abstract: This paper sets forth a cloud platform for implementing a completely integrated system for healthcare that fulfils identified field specific requirements. To achieve a high level of performance at a low cost, the paper presents an analytical model for cloud computing scheduling. The proposed model takes into consideration classification based on message priority, separation of classification and processing from the message output servers, and random load of the incoming requests. Performance is measured in terms of the number of requests, waiting time, response time, and requests drop rate for each priority class defined. Experimental results indicate that proposed model is able to support a great number of arrival requests providing short response time related to priority classes.

Keywords: Cloud computing, queuing model, performance analysis.

1. Introduction

Information technology and management systems dedicated to healthcare services require a high level of quality, safety, cost effectiveness and access. Medical data should be available to internal organizations and authorized third parties, at any time, at low cost, at high quality and via secure channels.

Cloud computing paradigm is defined as a set of network enabled services providing scalable, on demand resources [7]. Moving data to the cloud enables data consolidation, aggregation and reduces financial expense through minimised redundancy and cheaper operation costs.

There are four types of cloud models: public, private, community and hybrid. Private cloud infrastructure is provisioned for exclusive use of a single organization. Public infrastructure is provisioned for open use of general public. Community cloud infrastructure is provisioned for exclusive use of specific community consumers from organizations that have shared concerns. Hybrid cloud infrastructure is a composition of two or more distinct deployment models.

In the healthcare domain, the chosen approach must combine the secured access of individual organisations with the availability needed for public information. This can be achieved by developing a hybrid cloud application that allows a company to use public IT resources while keeping sensitive information private. In such applications, lack of compatibility might limit the platform functionality.

DICOM (Digital Imaging and Communications in Medicine) andHL7 (Health Level 7) standards provide specifications on how to implement communication and exchange of data between medical software in order to eliminate or reduce incompatibility among different applications.

Cloud infrastructure provides access to a large pool of virtualized resources (servers, storage, applications, services, etc.) available for dynamic usage by all customers. Sharing takes place per a customized service level agreement (SLA) between customers and provider. SLA management consists of two phases: negotiation and monitoring Quality of Service (QoS) for provided infrastructure. Defined QoS performance parameters for cloud platforms are: time spent, waiting time and mean number of tasks in the system, rejection probability and total response time of service [12]. SLA QoS parameters can be determined using queuing theory [10] and can be used to improve the performance and reduce delays in the transmission of information.

This paper provides a quantitative method for modelling message processing considering three connected queuing systems: a scheduling queue, a primary queue and a secondary queue. The method was designed for a hybrid cloud in the healthcare domain, but can be applied to any cloud model. The paper continues the work presented in [7] where the authors provided a detailed architecture for such a system, comprising of two queues, and evaluated the resources needed for such an application.

This paper is organized as follows. Section 2 describes general background of the research domain and related work. Section 3 presents the preliminaries for computing mean performance measures of the cloud platform. In section 4 experimental results are presented and section 5 concludes the paper with a summary of achievements.

2. Background and related work

The benefits of adopting hybrid cloud technologies in the healthcare domain have been analysed in several papers [7][2][4]. They combine public cloud benefits like high accessibility, easy maintenance and scalability with private cloud advantages, more focused on data privacy and controllability. In healthcare, such cloud models can be considered as a solution for complex data mining algorithms.

Related work in this domain analysed different methods for evaluating cloud systems performance: different queuing models, Petri nets, stochastic analysis or genetic algorithms [6]. In [88] authors use load balancing and two schedule entities to determine the resource that will process the request globally and to define the order in which requests are processed. Results indicate that proposed model provides significant improved performance compared to M/M/1 and M/M/S models. Several papers evaluate performance of different queuing models in particular resource configurations. In [5] authors provide a data centred model based on a M/M/m queuing system that showed improved message processing speed compared to classic method of short service time first-in first-out (FIFO). In [2] a priority M/M/C/C queue framework is proposed to support cloud provider service level agreement (SLA) in terms of data centre dimensions, needed resource capacity needed and number of supported requests. In [10] authors propose a dynamic scheduling algorithm to be used in priority queue systems that showed improved throughput, low drop rate and high access to resources considering different class priority levels. In [9] resource allocation problem is studied and the authors provided a network of M/M/1 queues service scheme to minimize resource cost and service response time for cloud providers. In [4] authors presented an algorithm for simulation of waiting systems. In all these papers queuing algorithms are presented as a tool for analysing and predicting client-server application behaviour and support decision making for resource.

Related work mainly addresses general queuing models that mainly address a single aspect of the request processing. This paper extends the research in cloud queuing theory by designing a new model that is better suited to the needs of healthcare applications. The proposed model takes into consideration classification based on message priority, separation of classification and processing from the message output servers, and random load of the incoming requests. This is accomplished by extending the priority-based model presented in [11].

3. Preliminaries and notations

Queuing theory provides an abstract representation of mathematical systems. It is able to isolate factors that affect system stability to be able to measure system performance: server utilization, waiting time, number of requests in the system, probability to exceed buffer of the waiting queue, response time [3].

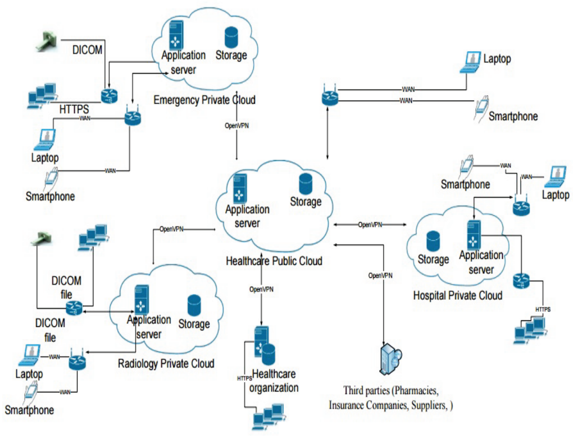

Figure 1. Healthcare hybrid cloud architecture [7]

A queue can be represented as $A/S/c/K/N/D$, where A is the arrival process, S is service time distribution defined as exponential service time, c is the number of servers, K is capacity of queue, N represents the size of population (jobs) to be served, and D defines the queue discipline. K and N are considered infinite in case they are not specified. By default, D is FIFO.

Queuing theory focuses mainly on steady state condition. It is defined as the state in which the system becomes essentially independent of the initial state and elapsed time. A system in steady state condition is characterized by the following parameters [8]:

- use ratio ρ ,defined in (3.1), must fulfil the steady state condition $\rho < 1$. Therefore λ, arrival

rate (requests per unit of time), must be strictly less than service rate μ. For $\rho \geq 1$ queue grows without bound;

$$\rho = \frac{\lambda}{\mu}, \rho < 1 \tag{3.1}$$

- probability of exact k number of requests in system, $P(k)$;

- expected queue length (excluding number of processed requests) for c number of resources available in the system;

$$E(Q) = \sum_{k=c}^{\infty} (k-c)P(k) \tag{3.2}$$

- expected number of customers in the system;

$$E(N) = \sum_{k=0}^{\infty} kP(k) \tag{3.3}$$

- expected waiting time in the queue (excluding service time) for each request;

$$E(W_Q) = \frac{E(Q)}{\sum_{k=0}^{\infty} \lambda_k p(k)} \tag{3.4}$$

-total waiting time (including service time) for each request, assuming that mean service time $1/\mu$ is constant.

$$E(W) = E(W_Q) + \frac{1}{\mu} \tag{3.5}$$

Let N denote the number of requests in the system and T request response time, Little's law gives the relation between mean number of requests in the system, mean response time in the system and average arrival rate of the requests [8]:

$$E(N) = \lambda E(T) \tag{3.6}$$

4. Analytical Hybrid Cloud Queuing Model

Mathematical analysis can measure the system performance in terms of queue length, response time including average waiting time, mean queue size, delay and resource utilization. It can demonstrate effect of priority classes on overall system performance and it can determine the number of resources needed, as function of arrival rates.

We define of arrival requests process in the proposed healthcare cloud application as follows:

- all user requests are gathered in a service centre as single point of access for all users. Arrival time between two client requests is independent of each other and is considered to follow a Poisson distribution: $\lambda = \lambda_H + \lambda_M + \lambda_L$, where λ_i is the arrival rate of a High, Medium or, respectively, a Low priority request;

- client requests will be processed by the service in a first come first serve (FCFS) order. Client requests will receive response after they are processed with a response time that must be lower than the agreed service level agreement response time;

- service response time is defined as $T_{srt} = T_{sub} + T_W + T_S + T_E + T_R$, where T_{sub} is the submission time, T_W is the waiting time, T_S is the service time, T_E is the execution time and T_R is the return time. T_{sub} and T_R are considered negligible [11].

- application buffer is limited, therefore the number of waiting requests is limited.

Secure data access for healthcare hybrid cloud platform is to be granted using RBAC (Role Based Access Control) model due to his granularity. Data will be secured using DICOM anonymization and identification services for specific medical area data (DICOM files). In case data is required outside private cloud it is transferred using the same encryption rules via DICOM anonymize/identify service, thus protecting confidential data. Healthcare cloud data integrity will be granted by the use of DICOM integrity validation services (DICOM object encoding) and digital signature.

We applied queuing theory to analyse the platform. We considered a model of cloud queue system composed by three queues: M/M/1 schedule queue, M/M/k/N primary queue and M/M/1 secondary queue. The FCFS queue is processed by a dispatcher that forwards the requests to a scheduler. Schedule queue system receives a FCFS queue and classifies each request building three input queues for primary system. Primary server will schedule the highest priority requests first and place the result in three priority queues. Load balancing can be applied at this level, by defining several channels for the input requests. This will increase the speed of the request classification system. In a secondary queue system, results are sent to the requester, sending the highest priority requests first.

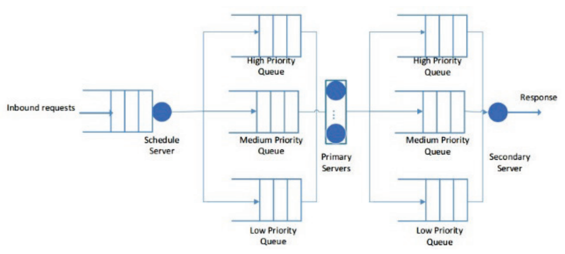

Figure 2. Analytical queue model

Observation 1: Priority class defined for a request has the following values: high, medium and low, which define the priority rule.

Based on Burke's theorem and Jackson's theorem the system can be modelled as an open Jackson network in which the queues can be analyzed independently.

4.1 Requests scheduling queue model

We consider that n requests are received randomly from all system users according to an independent Poisson process. Receiving rates are λ_H, λ_M and λ_L, and requests are processed by a single server. Service times of all jobs are exponentially distributed with the same mean $1/\mu$ [11]. According to (3.1) and Observation 1:

$$\rho_H + \rho_M + \rho_L < 1 \qquad (4.1)$$

where $\rho_i = \lambda_i/\mu$ occupation rate due to type i jobs and each type i request is treated according to its high, medium or low priority.

We modelled the system as a pre-emptive resume priority M/M/1 queuing system, satisfying stability condition $\rho < 1$. The mean number of requests in the system at time t can be computed as sum of all mean number of priority class requests in the system:

$$E(N) = E(N)_H + E(N)_M + E(N)_L = \frac{\rho}{1-\rho} \qquad (4.2)$$

Using (2.6), the mean service response time spent in the scheduling queue at time t is:

$$E(T) = \frac{E(N)}{\lambda} = \frac{\mu}{1-\rho} \qquad (4.3)$$

Mean waiting time is defined by the mean time spent in the system (3.3) subtracting service time:

$$E(W) = E(T) - \frac{1}{\mu} = \frac{\mu}{\rho(1-\rho)} \qquad (4.4)$$

According to the behavior of an M/M/1 queue, each priority class depends only on priority classes higher than his, and has no dependence with lower priorities. This means the mean number of high priority requests can be defined as:

$$E(N)_H = \frac{\rho_H}{1-\rho_H}; E(T)_H = \frac{\mu}{1-\rho_H}$$
$$E(W)_H = \frac{\mu}{\rho_H(1-\rho_H)} \qquad (4.5)$$

Using (3.2) and (3.5) mean performance measures $E(N)_M$, $E(T)_M$ and $E(W)_M$ for medium priority requests become:

$$E(N)_M = \frac{\rho_M}{(1-\rho_H)(1-\rho_H-\rho_M)} \qquad (4.6)$$
$$E(T)_M = \frac{1}{\mu(1-\rho_H)(1-\rho_H-\rho_M)}$$
$$E(W)_M = E(T)_M - \frac{1}{\mu}$$

Mean performance measures $E(N)_L$, $E(T)_L$ and $E(W)_L$ for low priority class are determined as:

$$E(N)_L = \frac{\rho_L(1+\rho_H-\rho_M)}{(1-\rho_H-\rho_M)(1-\rho_H-\rho_M-\rho_L)} \qquad (4.7)$$
$$E(T)_L = \frac{1/\mu\,(1+\rho_H-\rho_M)}{(1-\rho_H-\rho_M)(1-\rho_H-\rho_M-\rho_L)}$$
$$E(W)_L = E(T)_L - \frac{1}{\mu}$$

4.2. Primary system for requests classification

Primary system is modelled as a homogeneous $M/M/c/K$ queuing system (equal service rate for all servers). Three queues are used to hold priority requests according to the predefined classes (high, medium and low). Assuming that primary queue system has a total number of resources $C_{total} = \sum_{i=1}^{s} C_i, \forall i$ and a finite size, queue capacity is $K - C_{total}$, where K is the current number of customers. This limitation means that any request *n* that arrives at time *t* has an arrival rate of $\lambda_n = \lambda, \forall n < K$. Any request arrived after the maximum queue size limit has an input rate of 0: $\lambda_n = 0, \forall n \geq K$.

Priority class rejection probability is the main quality parameter of the primary queue system. In order to minimize rejection probability total resource number must be split into shared resources C_{shared} (C$_S$) and reserved resources $C_{rezerved} (C_R = C_{HR} + C_{MR})$ having $C_{total} = C_R + C_S$. High reserved resources $C_{HR} = \sum_{i=1}^{j_1} C_i, \forall i < j_1$ are used to serve only high priority requests. $C_{MR} = \sum_{i=1}^{j_2} C_i, \forall j_1 \leq i < j_2$ serve high and medium priority requests and $C_S = \sum_{i=j+1}^{C_{total}} C_i, \forall j_2 < i < C_{total}$ resources will process all types of requests.

The number of servers assigned for high, medium or low priority requests must be updated periodically according to a predefined dynamic scheduling algorithm. Algorithm 4.1 calculates the number of reserved and shared resources based on arrival rates, priority classes and total number of resources allocated for primary system. Assuming that each priority class is indicated by a priority parameter $\beta_i, i \in \{H, M, L\}$ $\beta_H \lambda_H \geq \beta_M \lambda_M \geq \beta_L \lambda_L$ and $\beta_M \lambda_M, \beta_L \lambda_L \neq 0$, algorithm 4.1 guarantees $C_{shared} \neq 0$.

Algorithm 4.1 Dynamic scheduling algorithm

Input: priority class arrival rates $\lambda_H, \lambda_M, \lambda_L$, priority levels $\beta_H, \beta_M, \beta_L$ and total number of resources C_{total}.

Output: Number of servers allocated for each priority class reserved resources (C_{HR}, C_{MR}) and shared resources C_S, computes as:

$$C_{HR} = \frac{C_{total} \beta_H \lambda_H}{\beta_H \lambda_H + \beta_M \lambda_M + \beta_L \lambda_L}$$

$$C_{MR} = \frac{k_1 \beta_M \lambda_M}{\beta_M \lambda_M + \beta_L \lambda_L}; k_1 = C_{total} - C_{HR}$$

$$C_S = k - C_{HR} - C_{MR}$$

Each server is able to serve any type of priority requests. When the number of shared and reserved resources is changed, all free resources are assigned according to the new defined pool. In case there are not enough free resources, when a busy resource will finish the current request, it will take a new request from the pool according to the new classification (shared or reserved).

Observation 2. High priority requests will be accepted in the system as long as:

$$C_S + \sum_{i=1}^{k} C_R = C_{total} - \sum_{i=k+1}^{j} C_R \qquad (4.8)$$

We assume all requests are queued in the priority queues, having service rate dependent of time *t*, with *n* number of requests being processed. A request leaves the system in a time $k\mu$. Service rate of the system increases until all servers are busy, C_R are busy with high and medium priority requests and C_S proceses low prority requests. Analytical analysis for primary system defines mean performance measures using M/M/c/K state probability equations [5] for a total number of $C_{total} = C_R + C_S$ resources and a maximum capacity *K*. *K* will be split in priority queues according to the number of priority classes.

System stationary probability of exact *n* customers in primary queueing system $p(n)$ for state *n* caracterized by arrival rate $\lambda(i)$ and service time $\mu(i)$ is determined based on balance equations. Balance equations define a stationary system when input flux is equal with output flux in a given state $n, n > 0$ for arrival rate λ_H and state dependent service rate $C_{rezerved} \mu$.

For a total request arrival rate λ and q proportion of high priority requests, we have $\lambda_H = q\lambda$ and $\lambda_M + \lambda_L = (1-q)\lambda$. Arrival rate is defined as: $\lambda(k) = \begin{cases} \lambda_H + \lambda_M + \lambda_L, k < C_{MR}, \\ \lambda_H + \lambda_M, C_{MR} \leq k < C_S, \\ \lambda_H, k \geq C_S. \end{cases}$

State transition diagram defines probability $p(n)$ of n requests in the system, using balance equations according to birth-death process:

$$\begin{cases}\lambda(0)p(0)=\mu(1)p(1)\\ \lambda(n)p(n)+\mu(n)p(n)=\lambda(n-1)p(n-1)+\mu(n+1)p(n+1)\end{cases} \tag{4.9}$$

Solving this system of linear equation, stationary probability $p(n)$ for state n is:

$$p(n)=\frac{\prod_{n=0}^{n=k-1}\lambda(n)}{\prod_{n=1}^{n=k}\mu(n)}p(0); p(0)=\left(1+\sum_{n>0}\frac{\prod_{n=0}^{n=k-1}\lambda(n)}{\prod_{n=1}^{n=k}\mu(n)}\right)^{-1} \tag{4.10}$$

Observation 3: Sum of all probabilities equals 1

$$\sum_{n=0}^{\infty}P_n=1 \tag{4.11}$$

Stationary probability of k class priority requests in primary queue is defined as $p(k,C)$ using (4.10) and (4.11) for a number of C allocated resources $C\in\{C_{HR},C_{MR},C_S\}$.

$$p(k,C)=\begin{cases}\dfrac{(\lambda_C/\mu)^k}{k!}p_0, 1\le k\le C\\ \dfrac{C^C}{C!}(\lambda_C/C\mu)^k p_0, C<k\le C+K\end{cases} \tag{4.12}$$

$$p(0,C)^{-1}=\begin{cases}1+\sum_{k=1}^{C}\dfrac{(\lambda_C/\mu)^k}{k!}+\dfrac{C^C}{C!}\sum_{k=C+1}^{C+K}\left(\dfrac{\lambda_C}{C\mu}\right)^k \lambda_C\ne C\mu\\ 1+\sum_{k=1}^{C}\dfrac{(\lambda_C/\mu)^k}{k!}+K\dfrac{C^C}{C!},\lambda_C=C\mu\end{cases} \tag{4.13}$$

Rejection probabilities for primary system, using final state probability of the system $p(C+K)$ for each number of allocated resources ($k=C+K$ and $C\in\{C_{HR},C_{MR},C_S\}$)is:

$$p_C(C)=\frac{C^C}{C!}(\lambda_C/C\mu)^{C+K}p(0,C) \tag{4.14}$$

Queue length is defined using (3.2) and (4.12) for $C<k\le C+K$ as:

$$E(Q)=\frac{C^C}{C!}\sum_{k=C+1}^{C+K}(k-C)(\lambda_C/C\mu)^k p(0,C) \tag{4.15}$$

Mean performance measures for all priority requests having $C\in\{C_{HR},C_{MR},C_S\}$ resources and arrival process $\lambda_C\in\{\lambda_H,\lambda_M,\lambda_L\}$, according to (3.3), (3.5) and (3.6), are defined as:

$$E(N)=E(Q)+\frac{\lambda_C}{C\mu}(1-p(K)); \tag{4.16}$$

$$E(T)=\frac{E(Q)}{\lambda_C(1-p_C(C))}+\frac{1}{C\mu};$$

$$E(W)=\frac{E(Q)}{\lambda_C(1-p_C(C))}.$$

4.3 Secondary system for response transmission

Secondary queue server is modelled with a finite capacity, with $K-1$ waiting positions, $M/M/1/K$ queuing system and mean performance measures $E(N)_i$, $E(T)_i$ and $E(W)_i$ $\forall i\in\{H,M,L\}$.

Similar as for primary queue, balance equations according to birth-death process are defined as:

$$\begin{cases}\lambda(0)p(0)=\mu(1)p(1)\\ \lambda(K)p(K)=\lambda(K-1)p(K-1)\end{cases} \tag{4.17}$$

Stationary probability is defined by solving system of linear equations:

$$p(k)=\begin{cases}\left(\lambda/\mu\right)^k p(0), k\le K\\ 0, k>K\end{cases} \tag{4.18}$$

$$p(0)=\begin{cases}(K+1)^{-1};\rho=1\\ \dfrac{1-\rho}{1-\rho^{K+1}};\rho\ne 1\end{cases} \tag{4.19}$$

Similar to (4.15), defined queue length at steady state, independently of request priority is

$$E(Q)=\sum_{k=1}^{K}(k-1)\left(\frac{\lambda}{\mu}\right)^k\frac{1-\rho}{1-\rho^{K+1}}, \rho\ne 1 \tag{4.20}$$

Mean performance measures according to (3.3), (3.5) and (3.6) are defined as:

$$\begin{gathered}E(N)=\frac{\rho}{1-\rho}-\frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}}\\ E(T)=\frac{E(N)}{\lambda(1-p(K))};E(W)=E(T)-\frac{1}{\mu}\end{gathered} \tag{4.21}$$

Considering $E(N)=E(N)_H+E(N)_M+E(N)_L$, (4.21) and Observation 1, performance measures high priority requests are:

$$\begin{gathered}E(N)_H=\frac{\rho_H}{1-\rho_H}-\frac{(K+1)\rho_H{}^{K+1}}{1-\rho_H{}^{K+1}}\\ E(T)_H=\frac{E(N)_H}{\lambda_H(1-p(K))}\\ E(W)_H=\frac{E(N)_H}{\lambda_H(1-p(K))}-\frac{1}{\mu}\end{gathered} \tag{4.22}$$

Medium priority requests mean performance measures are expressed as:

$$E(N)_M = \frac{\rho_H + \rho_M}{1-(\rho_H + \rho_M)} - \frac{(K+1)(\rho_H + \rho_M)^{K+1}}{1-(\rho_H + \rho_M)^{K+1}}$$
$$- E(N)_H \tag{4.23}$$
$$E(T)_M = \frac{E(N)_M}{\lambda_M (1-p(K))}$$
$$E(W)_M = \frac{E(N)_M}{\lambda_M (1-p(K))} - \frac{1}{\mu}$$

According to the queue discipline expressed in Observation 1, low priority requests will be processed after high and medium priority requests with following performance measures:

$$E(N)_L = \frac{\rho_H + \rho_M + \rho_L}{1-(\rho_H + \rho_M + \rho_L)}$$
$$-\frac{(K+1)(\rho_H + \rho_M + \rho_L)^{K+1}}{1-(\rho_H + \rho_M + \rho_L)^{K+1}} - E(N)_H - E(N)_M \tag{4.24}$$
$$E(T)_L = \frac{E(N)_L}{\lambda_L (1-p(K))}$$
$$E(W)_L = \frac{E(N)_L}{\lambda_L (1-p(K))} - \frac{1}{\mu}$$

Rejection probability for secondary subsystem is defined by probability $p(k)$ in final state:

$$p(K) = \rho^K \frac{1-\rho}{1-\rho^{K+1}} \tag{4.25}$$

5. Experimental results

We conducted a numerical analysis to evaluate the performance of the proposed queuing model. The analysis was made considering request handling at public cloud level of the following two aspects: level of QoS services and number of service resources. Level of QoS services can be guaranteed by a given number of service resources. The number of service resources refer to the ones that are required for a given number of customers to ensure that customer services can be guaranteed in terms of percentage of response time and ensuring reliable message delivery.

For simulation we used Matlab, considering messages enter randomly in the system, following a Poisson distribution. We defined a range between [145 – 3550] input requests per hour, having class priority distribution 30% high priority, 50% medium priority and 20% low priority (Figure 3).

Requests are processed by 5 servers having a service response time $\mu = 1$ for schedule and secondary resources and $\mu = 5$ for primary resources.

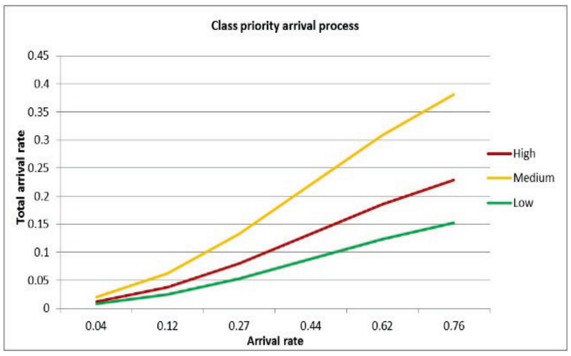

Figure 3. Arrival rate class priority requests distribution in requests per second

Based on theoretical analysis, this number is considered the minimum that could be used for this kind of architecture. Each server has a capacity buffer that can hold 50 requests per unit of time. Mean performance metrics $E(N)_i$, $E(T)_i$ and $E(W)_i$ $\forall i \in \{H, M, L\}$ are simulated for all three systems independently and concluded for the overall system in terms of response time for all priority classes.

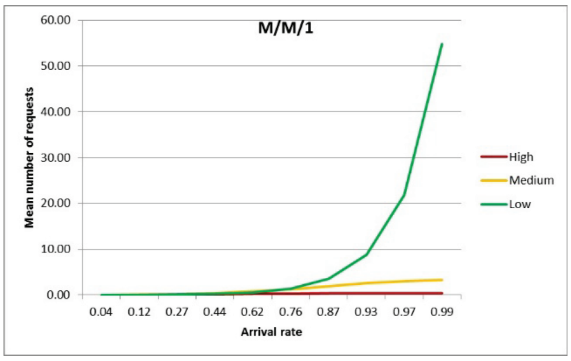

Figure 4. Mean number of requests in schedule queue

Schedule queue average load is simulated based on the mean number of requests in the schedule system calculated with (4.2). Results show that the number of high priority requests in the system decreases once arrival rate is significantly increased ($\lambda > 2743$ requests/hour). Class priority requests is respected, highest priority requests being processed first $E(N)_H < E(N)_M < E(N)_L$ (Figure 4).

Mean time spent in schedule queue is simulated using (4.3). Results show negligible values for high and medium priority requests independent of arrival rate. Low priority requests performance is increasing significantly for a single server in schedule queue when arrival rate $\lambda_L > 1585$ requests/hour (Figure 5).

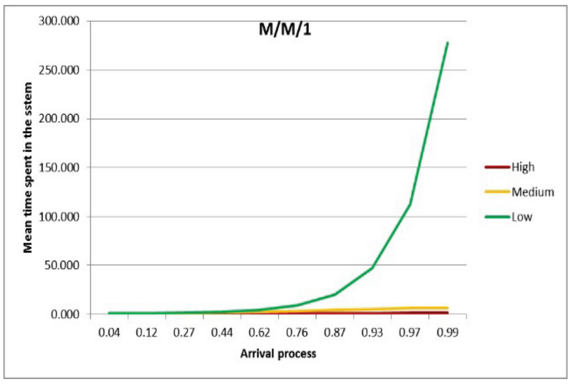

Figure 5. Mean response time in schedule queue

Results of simulating mean waiting time in schedule system, using (4.4), show that for an arrival rate $\lambda_L > 1585$ requests/hour performance for lowest priority class is bad (Figure 6). Therefore, the resource number should be increased.

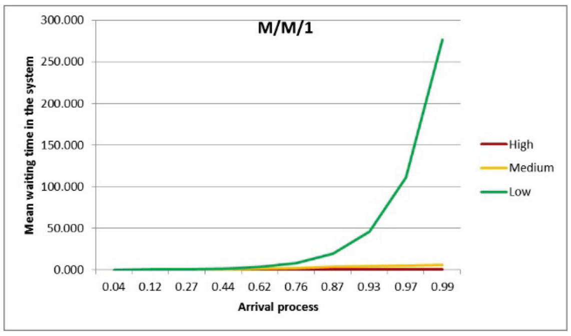

Figure 6. Mean waiting time in schedule queue

Figures 4, 5 and 6 show that schedule queue systems follows an exponential distribution for all performance metrics defined in section 4: $E(N)_i$, $E(T)_i$ and $E(W)_i$ $\forall i \in \{H, M, L\}$. Highest priority requests have good performance, independent of the arrival rate. Lowest priority class performance decreases significantly for an arrival rate of 1500-2000 requests/hour processed by a single server.

Primary queue system needs to process class priority queues received from the schedule queue and send same queues to the secondary queue that will only transmit to the requestor. $E(N)_i$, $E(T)_i$ and $E(W)_i$ performance metrics are analysed based on the dynamic resource allocation algorithm and relationships defined in section 4.2. With a service time greater than schedule system and a limited capacity of the buffers, primary system performance metrics are analysed relative to priority level for each class and arrival rate.

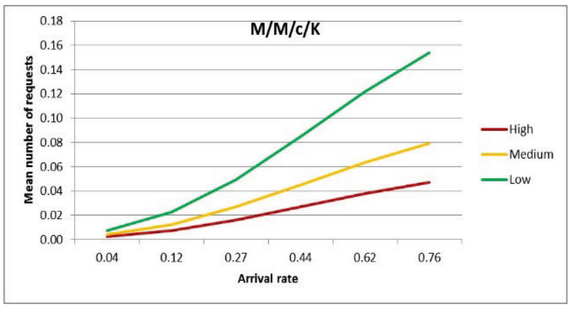

Figure 7. Mean number of requests in primary queue

Results show that highest priority requests are processed first keeping their number in the system to minimum (Figure 7). Algorithm 4.1 keeps a constant mean response time for high and medium priority request.

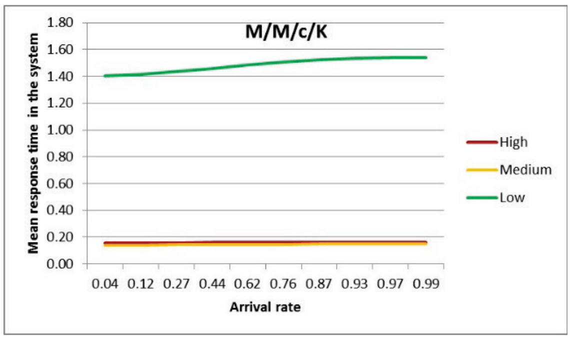

Figure 8. Response time in primary queue

For low priority requests response time is increasing very slow, even for high total arrival rate (Figure 8).

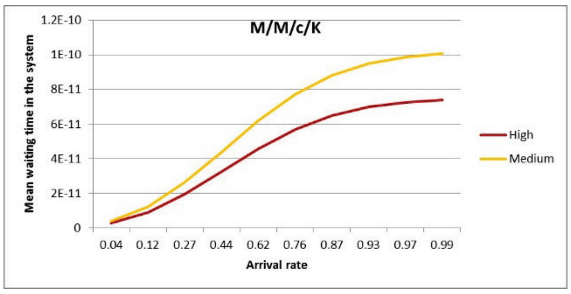

Figure 9. Mean waiting time in primary queue

Using (4.21), mean waiting time simulation show negligible values for highest priority classes and increases to accepted values between [0.7, 0.9] for the low priority classes (Figure 9).

Secondary queue system receives as input three priority queues from the primary queue system, having a single server with limited buffer capacity. Average occupancy is simulated based on mean number of requests in the system (4.19) and priority discipline as function of arrival rate (Figure 10).

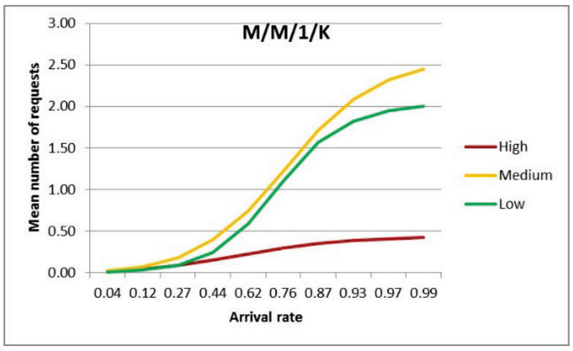

Figure 10. Mean number of requests in queue

Even though priority classes are not respected for the mean number of requests in the system, mean response time results follows priority discipline (Figure 11).

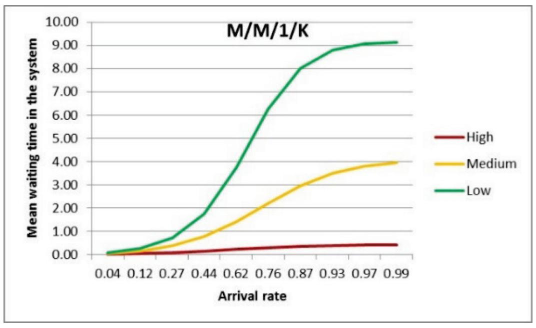

Figure 11. Mean waiting time in secondary queue

Figure 12 shows the result of the secondary queue expressed for the same queue discipline and increased number of servers. Performance is improved significantly, keeping priority discipline decreasing low priority waiting time from 9s to 1s.

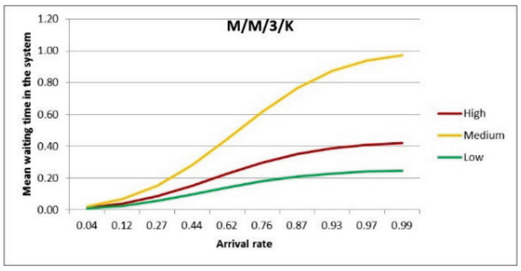

Figure 12. Mean waiting time in secondary queue

Ensuring reliable message delivery is dependent on the request drop rate. Rejection probabilities (drop rate) for all class priority requests are represented as function of allocated priority class resources in queues that have limited storage capacity: primary and secondary. Using (4.14), Figure 13 shows drop rate for all three priority classes. Results show that drop rate for high priority request are negligible $[10^{-46}:10^{-20}]$ for Poisson arrival process defined in Figure 1. Medium priority drop rates increases $[10^{-45}:10^{-17}]$ and also negligible low priority requests drop rate $[10^{-31}:10^{-11}]$.

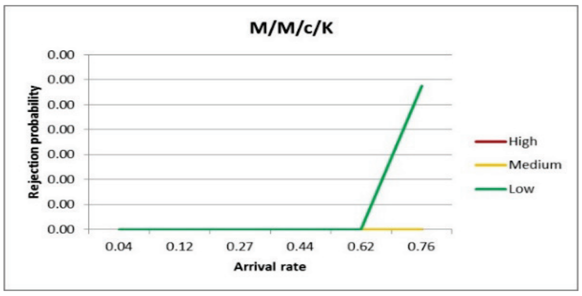

Figure 13. Drop rate for M/M/c/K queue system.

Secondary queue drop rate results are displayed in Figure 14 using (4.23). High and medium priority requests rejection probability is 0 or negligible (10^{-46}). Low priority requests drop rate is bad for $\lambda > 2217$ requests/hour for a single server system with limited buffer capacity.

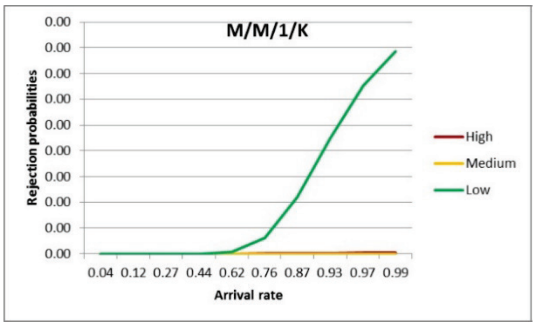

Figure 14. Drop rate for M/M/1/K queue system.

We considered processing class i priority requests for all three servers. Analysis of mean service response time showed good performance for high and medium priority, independent of the simulated arrival rate (between [145, 3550]). Low priority requests performance is low for $\lambda > 2743$ requests/hour but is improved by increasing the number of resources. Results showed that class priority driven solution ensure good performance all requests (Figure 15).

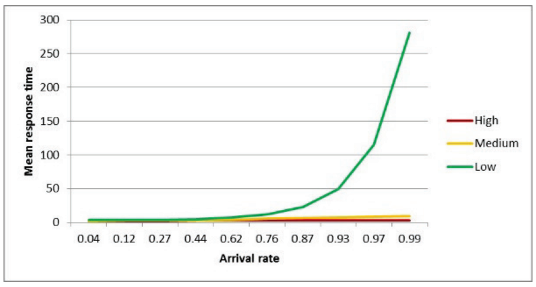

Figure 15. Queue system response time

6. Conclusions

This paper details the implementation of a queuing model that can be used for performance analysis in a hybrid cloud architecture. In order to achieve performance, priority classes have been introduced for arrival requests and have been considered in the analytical model. Model was build up using three linked queues. First one classifies requests based on their priority (schedule queue). The second one has a dynamic algorithm of resource allocation based on priority classes that concentrates the available resources (primary queue). The third one sends the requests received from the preceding queue to the requestor. Numerical analysis has been conducted for a Poisson distribution of arrival rates analysing three main performance metrics of the system: mean number of the requests, mean response time and mean waiting time in the system. Based on the obtained results we conclude the following: critical performance state for a queue system with 5 servers and limited buffer capacity is given by arrival rates lower than 70 times buffer

capacity; split of requests into different priority classes does not affect overall system performance for low arrival rates; dynamic resource algorithm for primary queue is needed in order to keep good performance for highest priority classes independent of arrival rate, service time and buffer capacity; low priority classes performance can be improved by increasing resource number in all processing points.

While this method can be applied on either public, private or hybrid cloud structure, the authors proposed a hybrid cloud architecture for a healthcare application because of the characteristics of all involved application users. Future work will analyse security and privacy over the data stored in the cloud.

Acknowledgements

This work was supported by the Romanian National Research Programme PNII, project: Cloud Architecture for an open Library of Complex re-Usable Logical function blocks for Optimized Systems – CALCULOS.

REFERENCES

1. Daskin, M. (2010). Service Science, *John Wiley&Sons*, Inc.
2. Ellens, W., Zivkovnic, M., Akkerboom, J., Litjens, R., Van Den Berg, H. (2012) Performance of cloud computing centres with multiple priority classes, *Proceedings of the 5th IEEE International Conference on Cloud Computing*, pp.245-252.
3. EMC (2014). Hybrid Cloud Powers Next-Generation Health IT at the Point-Of-Care, *Whitepaper EMC Healthcare Solutions*.
4. Florea, I. Sasu L. (2012). An Algorithm for Simulation of Waiting Systems with Different Types and Variable Number of Parallel Working Stations Each Having its Own Queue, *Studies in Informatics and Control, vol.* 21 (3), pp.333-340.
5. Guo, L., Yan, T., Zhao, S., Jiang, C. (2014). Dynamic performance optimization for cloud computing using M/M/m queuing system, *Hindawi Publishing Corporation Journal of Applied Mathematics*.
6. HASSAN, M. A., Kacem, I., Martin, S.,& Osman, I. M. (2015). Genetic Algorithms for Job Scheduling in Cloud Computing, *Studies in Informatics and Control*, *vol.* 24 (4), pp.387-400.
7. Marcu, R., Popescu, D., Danila I. (2014). Healthcare integration based on cloud computing. *U.P.B. Sci. Bull.*, *Series C, Vol. 77*, Issue 2, pp.31-42.
8. Mohamend, E., Esedimy, E.I., Rashad M.Z. (2014). Enhancing cloud computing scheduling based on queuing models, *International Journal of Computer Applications (0975-8887), volume 85* - no. 2.
9. Nan, X., He, Y., Guan, L. (2011). Optimal resource allocation for multimedia cloud in priority service scheme, *Proceedings of the 13th IEEE International Workshop on Multimedia Signal Processing*, pp.1-6.
10. Narman, H., Hossain, S., Atiquzzaman, M. (2013). Dynamic Dedicated Servers Scheduling for Multi Priority Level Classes in Cloud Computing, *IEEE Int. Conf. on Communications.*
11. Xiong, K., Perros, H. (2009). Service Performance and Analysis in Cloud Computing, *IEEE Congress on Services (SERVICES 2009).*
12. Wang, L., von Laszewski, G., Younge, A. et al. (2010). Cloud computing: a perspective study. *New Generation Computing, vol. 28*, no. 2, pp.137–146.