Synchronization Server Infrastructure: A relationship between System Downtime and Deployment Cost

Carlos Melo*, Jamilson Dantas*, Iure Fé*, Andre Oliveira*, Paulo Maciel*
*Informatics Center, Federal University of Pernambuco, Recife, Brazil
{casm3, jrd, isf2, apbo, prmm}@cin.ufpe.br*

***Abstract*—The perfect relationship between deployment costs and systems availability is one of the primary goals of companies that wish to provide some computer environment or service through the Internet. The question that everyone wants to know the answer is: How much may I save and still improve the availability of my system avoiding financial losses with an SLA contract breach? This paper attempts to respond to this question by combining some techniques used at dependability evaluation field, such like hierarchical models, sensitive analysis techniques, and analysis costs. The models shown in this paper are artifacts and scenarios for a data synchronization server hosted at a private cloud computing platform based on Eucalyptus. Each scenario presented here was generated to study the impact of redundant techniques in the system availability and deployment cost. By analyzing each architecture separately and comparing them, we can indicate the one that has the best cost-benefit relationship and attends the providers, as well as the client's needs.**

Keywords*—*Data Synchronization, Availability Models, Sensitive Analysis, Cloud Computing, Hierarchical Modelling, Deployment Cost.

I. Introduction

Mobile devices usage had an enormous growth since the beginning of the 21st century, thanks mainly to the advances in wireless communications and the widespread access to the Internet. The number of mobile devices activation grows five times faster than the human population [1]. Many people use more than one computational device throughout their daily routine, and in return to the continued growth in the design of these devices the complexity of their interconnection arises and the probability of conflicts and failures of the respective services. Seeking to reduce this kind of problem, many companies have migrated to cloud computing.

Data synchronization is one of the most prominent services offered through cloud computing environments. Google Drive, Dropbox, OneDrive, bank transactions, video and music on demand had become essential for users that want to share and access personal and non-sensitive enterprise data anywhere, from any terminal and available seven days a week and 24 hours a day [2]. However, establish availability parameters for this type of service is a significant challenge that companies are facing to offer data synchronization applications, or even host this kind of service.

This paper is an extension from [3] and evaluates the availability of distinct redundancies techniques applied to synchronization server deployed on a set of cloud infrastructures. We describe and evaluate each environment providing their deployment cost and annual downtime, by doing it we can point which one has the best cost $\times$ benefit values.

The remainder of this paper is organized as follows. Section II shows the background used as a basis for the development of this study. Section III shows the proposed architecture that is evaluated in our study. Section V provides the case studies considering the metrics computed for each distinct scenario. Finally, Section VI presents the final remarks and future works.

II. Background

This section presents concepts regarding data synchronization, dependability, replication techniques, stochastic modeling, and sensitivity analysis, which are essential for a proper understanding of this paper.

A. Data Synchronization

Data synchronization is the process that allows data consistency between a source and a target storage mechanism [4]. Synchronization Markup Language (SyncML) is an open standard for data synchronization between Internet-connected devices, which is designed to work as a client-server infrastructure.

The main components of a SyncML based framework are: the web server, which permits the connection between two distinct applications, a client and a server one; the database, responsible for storing the path to a user's data; and the actual synchronization server, which is in charge of the security and data exchange between devices.

B. Dependability and Replication Techniques

Dependability is the ability of a computer system to deliver a service that can be justifiably trusted [5]. This definition considers the user's perception and evaluation of the system behavior. Evaluating the dependability of a system is usually the result of evaluating at least one of its five attributes: reliability, availability, maintainability, integrity and safety [6].

Steady state availability, also called the long-run availability, may be represented by a ratio between the Mean Time To Failure (MTTF) and Mean Time To Repair (MTTR) of the system (Equation 1).

$$A = \frac{MTTF}{MTTF + MTTR} \tag{1}$$

The MTTF for a system may be computed from Equation 2, where **R(t)** is the reliability of that system as a function of elapsed time. The Equation 3 provides a way of computing the MTTR from the values of MTTF, availability (A), and unavailability (UA = $1 - A$).

$$MTTF = \int_0^\infty R(t)\partial t \tag{2}$$

$$MTTR = MTTF \times (\frac{UA}{A}), \tag{3}$$

System downtime is the period when the service has been unavailable; usually, it is given in hours per year: $Downtime = UA \times 8760h$, whereas the system's annual uptime, the period when the system has been available; therefore $Uptime = 8760h - Downtime$.

Replication consists in deploying redundant components, such as spare virtual machines, computers or servers that may assume the service provisioning role if the main component that was responsible for it enters into a failure state [7].

C. Stochastic Models

Stochastic models for reliability and availability evaluation may be broadly classified into combinatorial (or non-state space) and state-space models. Combinatorial models capture conditions that make a system fail (or to be working) regarding structural relationships between the system components. State-space models represent the system behavior (failures and repair activities) by its states and event occurrence expressed as rates or distribution functions. These models allow representing more complex relations between components of the system than combinatorial models do.

Reliability Block Diagram (RBD) and Fault Tree are among the most prominent combinatorial modeling formalisms, whereas Stochastic Petri Nets (SPN), Continuous Time Markov Chain (CTMC), and Stochastic Automata Networks are widely used state-space modeling formalisms [8].

D. Sensitivity Analysis

Sensitivity analysis techniques aim at determining the most influential factors in a system [9], [10]. With sensitivity analysis techniques we can detect the most critical system's components where the manager or administrator should invest with the highest priority to improve the results of metrics of interest.

In this paper we employ a percentage difference technique, which computes the sensitivity index $S_\theta(Y)$, which indicates that impact of a given measure Y with respect to a parameter θ. Equation 4 presents how the percentage difference index is computed for a metric $Y(\theta)$, where $max\,\{Y(\theta)\}$ and $min\,\{Y(\theta)\}$ are the maximum and minimum output values, respectively, computed when varying the parameter θ over the range of its *n* possible values of interest. If $Y(\theta)$ is known to vary monotonically, so only the extreme values of θ (i.e., θ_1 and θ_n) may be used to compute $max\,\{Y(\theta)\}$; $min\,\{Y(\theta)\}$, and subsequently $S_\theta(Y(\theta))$ [10].

$$S_\theta(Y) = \frac{max\,\{Y(\theta)\} - min\,\{Y(\theta)\}}{max\,\{Y(\theta)\}} \tag{4}$$

While computing $S_\theta(Y)$, we fix the values of the other model parameters. The same process is carried out for every input parameter of the target model until all parameters have been analyzed and a sensitivity ranking is built. The parameters with largest index on that ranking have the most significant impact on model results.

III. System Architecture

Some cloud computing platforms can be used to deploy the infrastructure required for the data synchronization service evaluated in this paper. OpenStack, Open Nebula, CloudStack, and Eucalyptus are among the most commonly used platforms for building cloud infrastructures [11].

In general the architectures are divided in machines for control of access to the infrastructures (frontend) and for service provisioning (Nodes), the frontend usually have a hardware (HW), operating system (OS), Platform Manager (PM), Cluster Manager (CM), Block-based Storage (BBS), and File-based Storage (FBS). HW and OS components are also present on the nodes, which also contain the hypervisor responsible for management and allocation of resources for Virtual Machines (VM), the Instance Manager (IM), and the end-user service, which is comprises the Apache Web Server, Synchronization Server, and Database, running on the VM. This paper proposes two groups of architectures variations: the baseline and the ones with redundancy in the nodes.

A. Baseline Architecture

The baseline environment consists of two computers, a Frontend and the Node; this is the minimum amount of resources required when you want to have a machine unique to the management of resources and provision of the service in the Eucalyptus platform. Figure 1 shows a high-level vision of this architecture, which focuses on the area demarcated by the dotted square, *ie* the side of the provider of service, meaning that our models will not cover everything that occurs between the network and the client side.

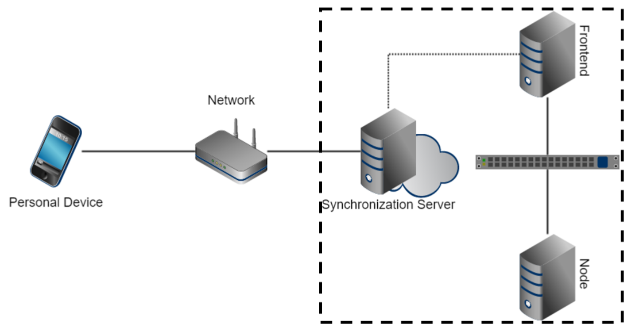


Fig. 1: High-level vision of Baseline Architecture

B. Node Redundant Architectures

The architectures with redundant nodes consists of three computers: a Frontend and two Nodes, one as spare or standby and another one receiving the load. Figure 2 shows a high-level vision of these architectures.

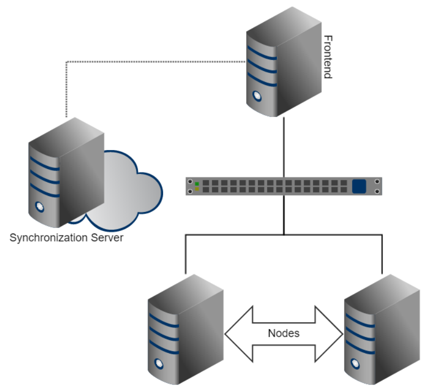


Fig. 2: High level vision of node redundant Architectures

IV. Availability Models

This section describes the models used for the availability system evaluation of our two scenarios.

A. Baseline Architecture

RBDs based in the ones proposed in [12], [7] are used to represent the relationship between the Eucalyptus cloud components. The choice of RBDs was due to the absence of priority among the elements of this architecture, which implies that any component in a faulty state will affect the availability of the proposed service. Figures 3 and 4 shown the RBDs for the Frontend and Nodes respectively.

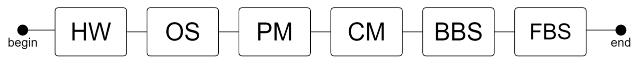


Fig. 3: RBD Frontend

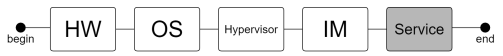


Fig. 4: RBD Node

It is important to mention that the service components in node's RBDs are a subsystem, which is described by the CTMC in Figure 5. We have made the choice for a CTMC with the purpose of using an immediate repair strategy that consists of the rebooting of the virtual machine.

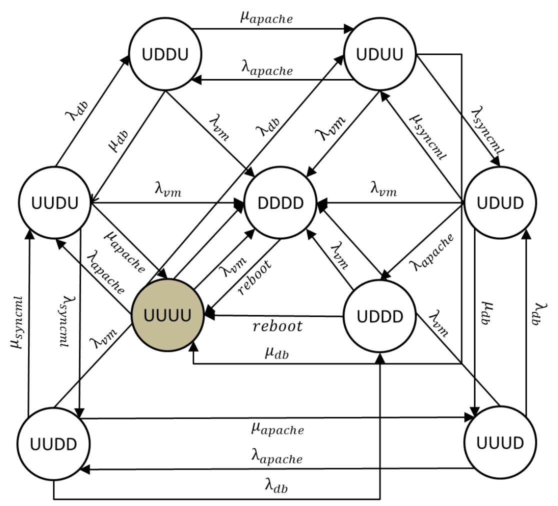


Fig. 5: Service CTMC

The service CTMC are described in Table I, the only place where the system is available is represented by **UUUU**, where each components is in UP state. The acronym DB is for database.

TABLE I: Status of the service

State	Meaning	Service
UUUU	VM UP, Apache UP, DB UP, SyncML UP	UP
UUUD	VM UP, Apache UP, DB UP, SyncML DOWN	DOWN
UUDU	VM UP, Apache UP, DB DOWN, SyncML UP	DOWN
UDUU	VM UP, Apache DOWN, DB UP, SyncML UP	DOWN
UDUD	VM UP, Apache DOWN, DB UP, SyncML DOWN	DOWN
UUDD	VM UP, Apache UP, DB DOWN, SyncML DOWN	DOWN
UDDU	VM UP, Apache DOWN, DB DOWN, SyncML UP	DOWN
UDDD	VM UP, Apache DOWN, DB DOWN, SyncML DOWN	DOWN
DDDD	VM DOWN, Apache DOWN, DB DOWN, SyncML DOWN	DOWN

B. Active-active Architecture

To turn it in a Active-active redundant system we need to put the redundant components to work together, as can be seen in Figure 6.

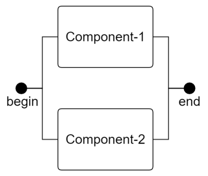


Fig. 6: Active-active component

Both components are activated, however only one is necessary to keep the service operational, as described in RBD's subsection. This RBD may describe nodes and frontend redundancy.

C. Node Warm and Cold Standby Standby Architecture

Already for a warm and cold standby redundancy be applied to the nodes, we need to represent the interaction between these components with another CTMC, Figure 7 is based on the one presented in [13].

The notation adopted to define each CTMC component resumes to (**U**) for when its **UP** and (**D**) for when its **DOWN**; the first character represents the service, the second and the third may represent the redundant node or the node where the service is running. The states are UWU, DWD, DWU, UDU, DDU, DDD, DUD, DUW, DDW, UUD and UUW. The four states where the system is available are UWU, UDU, UUD, and UUW, which is the only one where at least one node and the service are in UP state.

The subsystem's availability is equal to the sum of the probabilities of the system being on UWU, UDU, UUD and UUW states. In theses states, at least one node is functional, the CTMC that represents the system is explained by Table II. The main difference between cold and warm standby is the activation time *sa*.

V. Case Study

Each scenario was chosen to represent the closest approach to what happens in the real world; which can be seen in the next subsections.

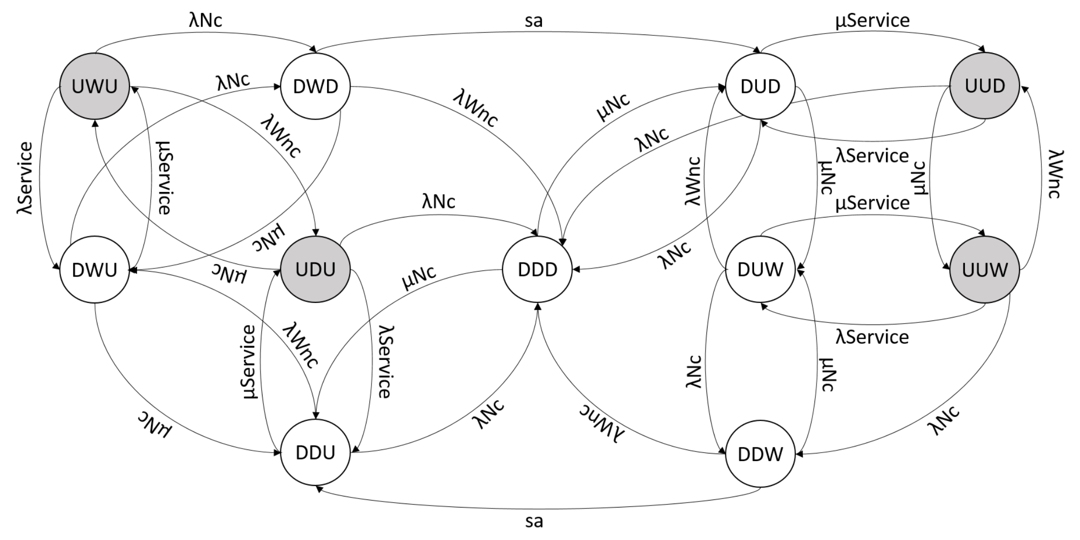


Fig. 7: CTMC node with warm standby redundancy

TABLE II: Status of Nodes in warm standby redundancy

State	Meaning	Status
UWU	Service UP, Standby Node WAITING, Node UP	Available
UUD	Service UP, Standby Node UP, Node DOWN	Available
UUW	Service UP, Standby Node UP, Node WAITING	Available
UDU	Service UP, Standby Node DOWN, Node UP	Available
DUW	Service DOWN, Standby Node UP, Node WAITING	Unavailable
DUD	Service DOWN, Standby Node UP, Node DOWN	Unavailable
DDD	Service DOWN, Standby Node DOWN, Node DOWN	Unavailable
DDW	Service DOWN, Standby Node DOWN, Node WAITING	Unavailable
DDU	Service DOWN, Standby Node DOWN, Node UP	Unavailable
DWU	Service DOWN, Standby Node WAITING, Node UP	Unavailable
DWD	Service DOWN, Standby Node WAITING, Node DOWN	Unavailable

A. Case of Study I - Baseline Architectures and Variations

In this paper 4 architectures are presented, 3 of them being variations of primary or baseline architecture. Table III describes each of these environments, the number of machines and the kind of redundancy applied.

TABLE III: Architectures List

Acronym	Architecture Description	No. of Machines
A1	Baseline Architecture	2
A2	Node in Active-active	3
A3	Node in Warm Standby	3
A4	Node in Cold Standby	3

All proposed architectures were arranged hierarchically, aiming to obtain the general availability values for each one. These are shown in Table IV.

TABLE IV: Comparison of availability results for each architecture

Architecture	Av. (%)	Uptime (h/year)	Downtime (h/year)
A1	98.14	8597.75	162.35
A2	99.46	8713.26	46.74
A3	99.44	8711.37	48.63
A4	99.40	8707.9	52.10

To performs the models evaluation, which defines each of the 4 architectures, some input values become necessary. These values are presented in Table V, some of these values were extracted from the work done by [12], [14], [15] others calculated from the models proposed for the baseline architecture. The evaluation period happened in a pre-determined period of one year or 8760 hours.

TABLE V: Input parameters for each architecture

Component	MTTF	MTTR
HW	8760 h	1.66 h
SO	2893 h	0.25 h
CLC	788.4 h	1 h
CC	788.4 h	1 h
SC	788.4 h	1 h
Walrus	788.4 h	1 h
Hypervisor	2990 h	1 h
NC	788.4 h	1 h
SyncML	788.4 h	1 h
Apache	788.4 h	1 h
Database	1440 h	13 h
VM	2880 h	reboot
Frontend subsystem	180.72 h	0,9687 h
Node subsystem	484.81 h	0.911 h
Node subsystem in standby	581.77 h	0.911 h
Service subsystem	279.365	0.027 h

where **reboot** is the rate to restart or re-instantiate a virtual machine, which is equivalent to 52.17 or 69 seconds.

With the obtained results, it is possible to find the value of 98.14% of availability for the baseline architecture (A1), which implies in 162.25 hours of downtime per year, corresponding to approximately to a week where users can not perform data synchronization.

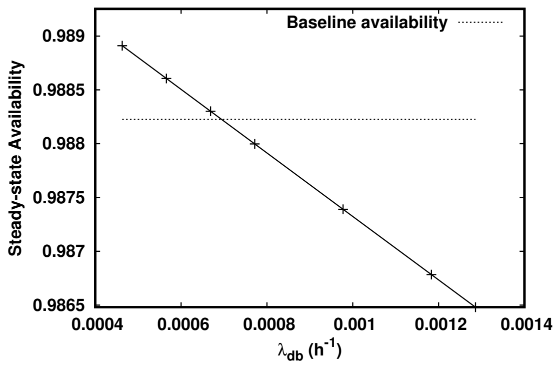
These results raise the question: What components are affecting the availability of the system in this way? To answer this question, we performed a sensitivity analysis and detect the primary elements responsible for these values, so that can be improved to provide increased availability of the system as a whole. The input parameters for sensitivity analysis are the same shown in Table V.

The technique used for the sensitivity analysis was the percentage difference, which is the variation of a parameter at a time and fix the others. The results of the sensitivity analysis are shown in Table VI.

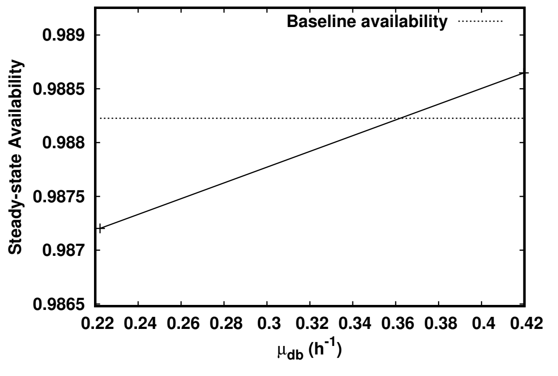
TABLE VI: Sensitivity Ranking for baseline architecture

Component	Position	Value
λ_{db}	1st	1.05E-02
μ_{db}	2nd	8.47E-03
$MTTF_{nc,walrus,clc,sc,cc}$	3rd	1.69E-03
λ_{apache}	8th	1.50E-03
λ_{syncml}	9th	1.50E-03
$MTTR_{nc,walrus,clc,sc,cc}$	10th	1.13E-03
$MTTF_{hw}$	15th	4.85E-04
$MTTF_{hypervisor}$	16th	4.46E-04
$MTTR_{hypervisor}$	17th	2.97E-04
$MTTF_{os}$	18th	2.30E-04
$MTTR_{hw}$	19th	2.03E-04
$MTTR_{os}$	20th	1.54E-04
λ_{vm}	21th	4.05E-05
$reboot$	22th	6.66E-06

Some of the components ranked the highest in the sensitivity analysis are related directly or indirectly to the service and the node where the service is running. The range of values and the rates of these components can significantly affect the availability of the service as can be seen in λ_{db} and μ_{db}, failure and repair of the database, which are service components. Figures 8a and 8b shows the variation of their values, where the continuous line represents the availability of the system and the dotted part represents what the system's availability should be if we improve the rates.



(a) λ_{db}



(b) μ_{db}

Fig. 8: Variation of λ_{db} and μ_{db}

If we manage to increase the λ_{db}, we may significantly improve the availability of the system, but in the second case, if μ_{db} value is decreased, we will increase the metric value in the same way. With this in mind, we applied redundancy techniques in the node where the database is configured. Architectures A2, A3 and A4 shown downtime results closer to 2 days, **A2** with nodes in Active-active deserves noteworthy for presenting the highest availability since both machines are powered up.

B. Case of Study II - Cost and Power Consumption Analysis

To performs the cost analysis we lifted a series of data; This survey consisted of three stages.

The first step was the definition of the components necessary for the provision of the service, much of this process was carried out while setting the architectures in the first case study. Thus, we can define the components of each infrastructure as servers (variable number for architecture), Switch, rack, keyboard, monitor, mouse and personal computer and air conditioning (fixed amount).

In the second step was the examination of acquisition cost for each component individually, we consulted shopping sites, some Brazilian and North American ones, the lowest value of the components on April 13th, 2016, was chosen.

The third step is characterized by the analysis of the technical specification of each component in search of energy power values, aiming to calculate the energy cost per unit for the period of one year, where these components would be functional 24 hours a day seven days a week. The cost of kilowatt time in Recife, Brazil, in March 2016 was 58 cents of real, converted to 16 cents of a dollar. In possession of these data, we can calculate the amount of consumed kWh by a device in the course of one year by Equation 5, and multiply the result by the value of kWh.

$$kWh = \frac{Eq.Power(W) \times No.HoursDay \times No.ofDaysYear}{1000} \quad (5)$$

Table VII shows the relationship between components and their acquisition costs, potency and annual energy cost.

TABLE VII: Relationship between components, power and cost

Component	Purchase Cost (USD)	Watt	Energetic Cost (USD)
Server	1209.82	100	140.16
Switch	119.00	6.9	9.67
Air conditioning	490.83	1080	1513.73
Monitor	191.95	70	98.11
Keyboard	5.71	-	-
Mouse	2.55	-	-
Personal Computer	313.27	100	140.16
Rack	416.73	-	-

TABLE VIII: Estimated Cost for each architecture

Architecture	Watt Cost (USD)	Purchase Cost (USD)	Total Cost (USD)
A1	2041.99	3959.68	6001.67
A2	2182.15	5169.50	7351.65
A3	2110.24	5169.50	7279.74
A4	2047.24	5169.50	7216.74

Table VIII presents the overall acquisition costs for each proposed architecture where the column that is the total value it is the sum of acquisition costs and costs of electricity for each one.

There are some fixed expenses for each architecture; that expenditure corresponds to the switch, air conditioning, monitor, keyboard, mouse, and the tower of the personal computer needed to mount and access to infrastructure, a cost of 1540.04 dollars. Another fixed expense is the energy consumption of these components; this one is 1761.67 dollars.

The architecture with the largest cost is A2, which for its implementation and operation over a year presents a cost of 7351.65 dollars, a value 1349.98 dollars greater than the architecture with lower cost: A1. It is also important to report the values in watts for architectures with the warm type of redundancy and cold standby, which were 48.7 and 3.75 watts, respectively. We have found these values through experiment, which consisted of measuring values in watts per machine in standby mode and off through the WattsUP device.

To determine the architectures with the best cost $\times$ benefit relationship we must define what is the benefit we are seeking. For this work, we expect the ideal value between downtime and cost. But they are different greatness, so wee need to normalize and put them in the same interval: 0 and 1. Equation 6 presents the normalization process conducted.

$$NormalizeXY = \frac{NumX - MinNumX}{MaxNumX - MinNumX} \quad (6)$$

where $X = \{$Cost, and Downtime$\}$, $Y = \{$Architectures: A1, A2, A3, A4$\}$, $MinNumX = \{$Minimum value for architectures$\}$ and $MaxNumX$ = $\{$Maximum value for architectures$\}$

Now in possession of the normalized cost and downtime values, we must relate them somehow, so we use the Euclidean distance, the architecture with less distance from the origin tends to be the best one. Equation 7 shows how we perform the calculation of distances.

$$DistanceZ = \sqrt{MN^2 + MN^2} \quad (7)$$

where $Z = Architectures : A1, A2, A3 and A4$, $MN =$ Normalized Cost, Normalized Downtime.

The Ranking for the relationship between cost and downtime appears in Table IX.

TABLE IX: Architecture Ranking

Rank	Architecture	N. Cost	N. Downtime	Distance
1st	A4	0.45003	0.31974	0.52548
2nd	A3	0.47337	0.29833	0.53027
3rd	A2	0.50000	0.28667	0.53953
4th	A1	0.00000	1.00000	0.70711

The ranking shown that architecture A4, with node in cold standby redundancy, has the best relationship between cost and downtime. This architecture don't have the higher availability value or the worst price, but thanks to the normalization and distance methods we are able to define it as the best architecture for our goal. The last place goes for architecture A1, with the worse downtime.

VI. Final Remarks

In this paper hierarchical models, RBDs, SPN, and CTMCs were presented as artifacts for the providers who wants to offer a synchronization service through the Internet, the behavior and availability evaluation of 4 architectures were used to demonstrate how feasible are the models proposed. After that, a cost analysis and energy consumption study were applied to all environments aiming to find the one with the best relationship between cost and downtime.

The results shown that the architecture with higher availability was the one with Node in Active-active redundancy, while the one with the lower deployment cost was the baseline. The architecture with the best cost $\times$ benefit was the one with nodes in warm standby redundancy. As a future work, we intend to implement a variable relationship between metrics, which consists in set the downtime with a greater weight than the cost, and vice versa.

References

[1] Z. D. Boren, "There are officially more mobile devices than people in the world," http://ind.pn/1xlKiif, 2014, [Online; accessed 19-December-2015].

[2] A. Traud, J. Nagler-Ihlein, F. Kargl, and M. Weber, "Cyclic data synchronization through reusing syncml," *2008 The Ninth International Conference on Mobile Data Management*, pp. 165–172, 2008.

[3] C. Melo, J. Dantas, J. Araujo, and P. Maciel, "Availability models for synchronization server infrastructure," in *Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'16)*, Budapest, Hungary, 2016.

[4] S. Agarwal, D. Starobinski, and A. Trachtenberg, "On the scalability of data synchronization protocols for pdas and mobile devices," *Network, IEEE*, vol. 16, no. 4, pp. 22–28, Jul 2002.

[5] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, 2004.

[6] A. Avižienis, J. Laprie, B. Randell, and U. of Newcastle upon Tyne. Computing Science, *Fundamental Concepts of Dependability*, ser. Technical report series. University of Newcastle upon Tyne, Computing Science, 2001. [Online]. Available: https://books.google.com.br/books?id=cDkmGwAACAAJ

[7] J. Dantas, R. Matos, J. Araujo, and P. Maciel, "Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud," *Computing*, vol. 97, no. 11, pp. 1121–1140, 2015. [Online]. Available: http://dx.doi.org/10.1007/s00607-015-0447-8

[8] M. Malhotra and K. Trivedi, "Power-hierarchy of dependability-model types," *Reliability, IEEE Transactions on*, vol. 43, no. 3, pp. 493–502, Sep 1994.

[9] P. M. Frank, *Introduction to Sensitivity Analysis*. Academic, 1978.

[10] R. Matos, J. Araujo, D. Oliveira, P. Maciel, and K. Trivedi, "Sensitivity analysis of a hierarchical model of mobile cloud computing," *Elsevier Journal Simulation Modelling Practice and Theory*, vol. 50, pp. 151–164, Jan. 2015.

[11] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," in *Proc. of 2nd Int. Symp. on Information Science and Engineering (ISISE)*. Shanghai: IEEE Press, 2009, pp. 23–27.

[12] J. Dantas, R. Matos, J. Araujo, and P. Maciel, "Models for dependability analysis of cloud computing architectures for eucalyptus platform," *International Transactions on Systems Science and Applications*, vol. 8, pp. 13–25, 2012.

[13] R. M. D. Melo, M. C. Bezerra, R. M. Jamilson Dantas, I. J. D. M. Filho, and P. Maciel, "Redundant vod streaming service in a private cloud: Availability modeling and sensitivity analysis," *Mathematical Problems in Engineering*, vol. 2014, p. 14, 2014.

[14] I. Costa, J. Araujo, J. Dantas, E. Campos, F. A. Silva, and P. Maciel, "Availability evaluation and sensitivity analysis of a mobile backend-as-a-service platform," *Journal Quality and Reliability Engineering International*, 2015.

[15] M. C. Bezerra, R. Melo, J. Dantas, P. Maciel, and F. Vieira, "Availability modeling and analysis of a vod service for eucalyptus platform," in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, Oct 2014, pp. 3779–3784.