Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 764010, 14 pages
http://dx.doi.org/10.1155/2014/764010

Research Article

Redundant VoD Streaming Service in a Private Cloud: Availability Modeling and Sensitivity Analysis

Rosangela Maria De Melo,[1,2] Maria Clara Bezerra,[1] Jamilson Dantas,[1] Rubens Matos,[1] Ivanildo José De Melo Filho,[1,2] and Paulo Maciel[1]

[1] *Informatics Center, Federal University of Pernambuco (UFPE), 50740-560 Recife, PE, Brazil*
[2] *Federal Institute of Education, Science, and Technology of Pernambuco (IFPE), Belo Jardim Campus, 55155-730 Belo Jardim, PE, Brazil*

Correspondence should be addressed to Rosangela Maria De Melo; rmm3@cin.ufpe.br

Received 25 April 2014; Revised 31 July 2014; Accepted 31 July 2014; Published 30 September 2014

Academic Editor: Oluwole Daniel Makinde

For several years cloud computing has been generating considerable debate and interest within IT corporations. Since cloud computing environments provide storage and processing systems that are adaptable, efficient, and straightforward, thereby enabling rapid infrastructure modifications to be made according to constantly varying workloads, organizations of every size and type are migrating to web-based cloud supported solutions. Due to the advantages of the pay-per-use model and scalability factors, current video on demand (VoD) streaming services rely heavily on cloud infrastructures to offer a large variety of multimedia content. Recent well documented failure events in commercial VoD services have demonstrated the fundamental importance of maintaining high availability in cloud computing infrastructures, and hierarchical modeling has proved to be a useful tool for evaluating the availability of complex systems and services. This paper presents an availability model for a video streaming service deployed in a private cloud environment which includes redundancy mechanisms in the infrastructure. Differential sensitivity analysis was applied to identify and rank the critical components of the system with respect to service availability. The results demonstrate that such a modeling strategy combined with differential sensitivity analysis can be an attractive methodology for identifying which components should be supported with redundancy in order to consciously increase system dependability.

1. Introduction

Cloud computing environments provide adjustable storage capacity and processing power, as well as other computational resources, which enable the fast provision of varying workloads [1]. Most popular online services use cloud computing to ensure availability and proper service delivery to users [2]. And current video on demand (VoD) streaming services rely on such cloud computing benefits as the pay-per-use model and scalability [1]. Cloud infrastructures, however, are not failure free; recent outages at Amazon [3] disrupted the service provision of several key companies, including one of the major players in the VoD market. The design of video streaming services based on private clouds must therefore consider the deployment of high availability techniques such as redundancy [4] in an attempt to avoid the service interruptions that result from component failure.

This paper proposes an analytical availability model to support the evaluation of a VoD streaming service running in a Eucalyptus private cloud. The modeling strategy considers reliability block diagrams (RBDs) and continuous time Markov chains (CTMCs), as well as a parametric sensitivity analysis of the proposed model which can identify the bottlenecks in system availability and thereby guide the implementation of system improvements.

The remainder of the paper is organized as follows: Section 2 presents related works on system availability and sensitivity analysis; Section 3 discusses basic concepts of cloud computing technologies, video streaming, dependability model analysis, and sensitivity analysis; Section 4 describes the system architectures analyzed here; Section 5 presents the availability models designed for the architectures; Section 6 is a case study which proves the applicability of the proposed model; and finally Section 7 offers conclusions and also suggests the direction that future work will take.

2. Related Works

Analytical modeling techniques predict certain behaviors of systems. Such information can assist the decision making process concerning the design of system infrastructure and the achievement of required availability levels. Since several recently published works have employed hierarchical modeling to represent cloud computing architectures, comparisons of the various solutions and appraisals of dependability metrics can be made [5, 6]. Dantas et al. [6] investigated the benefits of a warm standby redundancy mechanism in a Eucalyptus cloud computing environment. A hierarchical modeling approach was employed to represent a redundant architecture and compare its availability to that of a nonredundant architecture. Chuob et al. [5] proposed a private cloud solution, selected from among the suitable cloud environments, for an e-government data center, basing the cloud model on the Ubuntu Enterprise Cloud (UEC) architecture. The availability of each component of the cloud was represented by a Markov chain. Matos et al. [7] investigated the availability of data networks including redundancy mechanisms. Several scenarios were evaluated through analytic-numeric solution of Markov chains. Furthermore, the impact of different component parameters on the overall system availability was evaluated with differential sensitivity analysis.

In another study, Bezerra et al. [8] investigated hierarchical modeling techniques to evaluate a nonredundant VoD service. The authors employed sensitivity analysis to identify bottlenecks in the model and by this means propose improvements. Meanwhile, Longo et al. [9] established stochastic analytic models that can be used for cloud service availability analysis. They developed a one-level monolithic model despite the fact that such models tend to become unmanageable when cloud size increases beyond a certain point. In fact, the results demonstrated that errors introduced by model decomposition are negligible. The authors developed closed-form solutions for the submodels and showed that the current approach is capable of being scaled up for large size clouds.

In [10], Bruneo et al. described GridVideo, a Grid-based multimedia application for the distributed tailoring and streaming of media files. The goal of this paper was to demonstrate through a real experience how Grid technologies can be used for the development of nonscientific applications. Relevant performance aspects were analyzed for responsiveness and system efficiency. Different multimedia data dissemination strategies were analyzed and an innovative technique based on the Fibonacci series was proposed. Alternatively, Ghosh et al. presented in [11] a scalable, stochastic model-driven approach to quantify availability in a large-scale IaaS cloud, where failures were normally treated through migration to physical machines divided into three redundancy pools: hot (fully running), warm (turned on, but not fully running), and cold (turned off). The researchers showed how scalability issues for a monolithic model can be solved by submodel interaction or simulation.

Virtualization refers to the technique of instantiating one or several virtual machines (VMs) on top of a single physical machine, administered by a virtual machine monitor (VMM) [12]. In [12], Bruneo et al. proposed a technique to model the VMM aging process and investigate the optimal rejuvenation policy for maximizing VMM availability under variable workload conditions. The authors started out with dynamic reliability theory and by adopting symbolic algebraic techniques investigated and compared existing time-based VMM rejuvenation policies. The paper proposed a policy that adapted the rejuvenation timer to the VMM workload condition, thereby improving system availability. From a different perspective, in Matos et al. [13], the authors used hierarchical modeling and differential sensitivity analysis techniques to determine which parameters cause the greatest impact on mobile cloud availability. The results proved that aside of specific exceptions, distinct approaches can deliver similar results in terms of sensitivity rankings.

In this paper, the authors analyze an analytical availability model to support the evaluation of a VoD streaming service running in a private cloud. A parametric sensitivity analysis of the proposed model is also presented, enabling the identification of availability bottlenecks and providing guidelines for the implementation of system improvements.

3. Fundamental Concepts

Certain concepts, related to cloud computing technologies, video streaming, dependability modeling, and sensitivity analysis, are fundamental to this paper and need to be understood.

3.1. Cloud Computing and the Eucalyptus Platform. A cloud computing system is a bundle of resources comprising hardware, software, development platforms, and services that are readily usable and accessible through the Internet [1]. Cloud computing providers supply these services at different levels, including infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). In these environments virtual resources can be dynamically allocated and resized to handle varying workloads, thereby optimizing the physical resources. Services are typically accessed through a pay-per-use model [14] whilst the degree of provision is determined and guaranteed through service level agreements.

Eucalyptus is a Linux-based software architecture that facilitates the implementation of private and hybrid IaaS clouds. Clients can utilize their own system resources together with cloud services through a self-service interface according to their needs at any particular time. The Eucalyptus software framework is modular [15] and consists of five high level components, each with its own web service. These are the cloud controller (CLC), cluster controller (CC), node controller (NC), storage controller (SC), and Walrus [15].

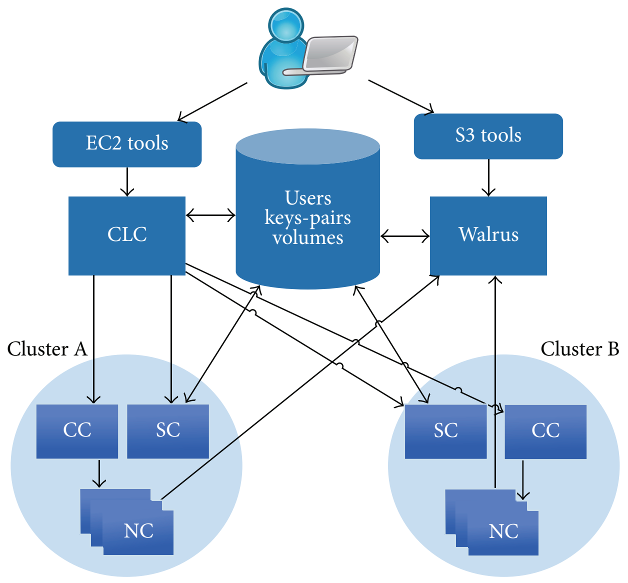

Figure 1: Eucalyptus platform architecture [15].

Figure 1 illustrates an example on Eucalyptus cloud computing environment comprising two clusters (A and B). Each cluster has a single CC and SC and various NCs.

The CLC is the frontend of the cloud infrastructure. It employs web service interfaces to receive client tool requests on one side and interact with the remaining Eucalyptus components on the other side [6, 16]. The CC usually executes on a cluster frontend machine or on any machine that has network connectivity to both the nodes running NCs and to the machine running the CLC [6]. The NC runs on each node and controls the life cycle of instances running on the node [6]. The NC interacts with the operating system and with the hypervisor running on the node. The SC provides persistent block storage for use by the virtual machine instances. Walrus is a file-based data storage service which is interface compatible with Amazon's Simple Storage Service (S3) [6].

3.2. Video Streaming. Video streaming is a technology employed in the instantaneous transmission of digital multimedia over the Internet [17]. Streaming enables data to be delivered and viewed without having to wait until it is fully downloaded and stored on the client system. This solution to multimedia access is fast whilst alleviating pressure on network bandwidth and client storage space. As the multimedia data downloads it is stored in a fast buffer for immediate execution. The effectiveness of video streaming transmission is heavily dependent on digital encoding, communication protocols, and buffering mechanisms [18].

Video streaming services are implemented according to the Real Time Streaming Protocol (RTSP) and designed to manage the transfer of audio and video data in real time from streaming servers [19]. The protocol establishes and controls the various synchronized streams of data involved in a multimedia transmission. Despite the bandwidth limitations, delay, and packet loss that is inherent in real time streaming applications, no quality of service (QoS) guarantee is offered by the Internet for this service [19, 20].

Besides the RTSP protocol, several additional elements comprise the VoD architectural environment. These include video compression, application-layer QoS control, streaming servers, media synchronization mechanisms, and further protocols for actually transmitting the data [20]. Initially, a compression algorithm condenses the data before being saved in a storage device. Upon request the streaming server retrieves the compressed data from storage and packages the compressed bit fluxes, before sending them off to the Internet to be viewed by the user [20].

From a commercial viewpoint VoD technology is closely associated with cloud computing. The commercial interest allows clients to access the hosted media files and distributes them when requested. This type of service is relatively of low cost; the user pays a fixed fee for uninterrupted service and gains freedom and flexibility in terms of what they watch and when they watch it [18].

3.3. Dependability Analysis Models. Dependability is closely related to the disciplines of fault tolerance and reliability. The concept of dependable computing in fact dates back to the 1820s when Charles Babbage undertook the mission to conceive and construct a mechanical calculating engine that would eliminate the risk of human errors [21–23]. In the early 1980s Laprie coined the term dependability for encompassing a set of concepts that included reliability, availability, safety, confidentiality, maintainability, security, and integrity [21, 24]. Although the concepts of availability and reliability are tightly linked there is a subtle distinction between them. Whereas the reliability of a system at time t is the probability that the system performs without failing up to time t, availability is expressed as the ratio of the expected system uptime to the expected total time (the sum of up and downtime combined):

$$A = \frac{E[\text{Uptime}]}{(E[\text{Uptime}] + E[\text{Downtime}])}. \quad (1)$$

It may also be represented by

$$A = \frac{\text{MTTF}}{(\text{MTTF} + \text{MTTR})}, \quad (2)$$

where MTTF and MTTR are the mean time to failure and recovery, respectively.

Several types of models can be used for the analytical evaluation of dependability. Reliability block diagrams (RBDs), fault trees, stochastic Petri nets (SPN), and continuous Time Markov chains (CTMCs) have all been employed to model fault-tolerant systems and evaluate various dependability measures. These types of models differ from one another not only in their suitability for a specific application but also in terms of modeling power [23]. Therefore, an evaluation which combines distinct model types may often provide the best solution. In the current situation under consideration, for example, the relationships [25] between independent subsystems are best modeled with RBDs, whereas detailed or more complex mechanisms, such as active redundancy mechanisms [26] and resource constraints, are better modeled with SPNs or CTMCs. Such a combined approach allows for the representation of the different types of dependency that exist between components and avoids the common problem of state explosion [27] which can occur when dealing with large systems.

3.4. Sensitivity Analysis. In general the various components that constitute a computer system do not necessarily contribute equally to system performance. The identification of the more essential elements is critical to the assessment of system dependability. This must be taken into consideration when designing a system so that resources may be allocated according to the levels of importance of each component. Since sensitivity analysis is a technique employed to determine factors that are most relevant regarding the measures or output of a model it can be of great assistance in establishing the critical components in a system, by identifying them in an analytical model [28].

Sensitivity analysis can be performed in several ways. The simplest technique is to select a parameter to vary while keeping the others fixed. The corresponding changes in model output are recorded for each input parameter variation and in this manner a sensitivity ranking is obtained. Further techniques of sensitivity analysis are experimental factorial design, correlation analysis, regression analysis, perturbation analysis (PA), and differential analysis, also known as parametric sensitivity analysis or direct method [7, 28, 29].

Differential analysis was chosen for this work because it can be efficiently performed in the type of analytical models usually employed in availability and performance studies. It is accomplished by calculating the partial derivatives for the measures of interest of the respective parameters. For example, considering a metric Y which depends on a parameter (λ), the sensitivity of Y with respect to λ is computed with (3) or (4) when adopting scaled sensitivity [25]. Consider the following:

$$S_\lambda(Y) = \frac{\partial Y}{\partial \lambda}, \tag{3}$$

$$S_\lambda^*(Y) = \frac{\partial Y}{\partial \lambda}\left(\frac{\lambda}{Y}\right). \tag{4}$$

Other scaling methods may be used, depending on the nature of the parameters, the measure of interest, and the requirement to remove the effects of units [29]. $S_\lambda(Y)$ and $S_\lambda^*(Y)$ are also referred to as sensitivity coefficients [28], whose ordered values produce the ranking used to compare the degree of influence among all parameters.

The Mercury tool [30, 31] assisted in the analysis presented in this work by providing the sensitivity indices of the CTMCs. These indices were employed in the sensitivity functions of the top-level RBD models.

4. VoD Service Architecture

The proposed VoD service architecture is based on the Eucalyptus platform. As shown in Figure 2, it is divided into

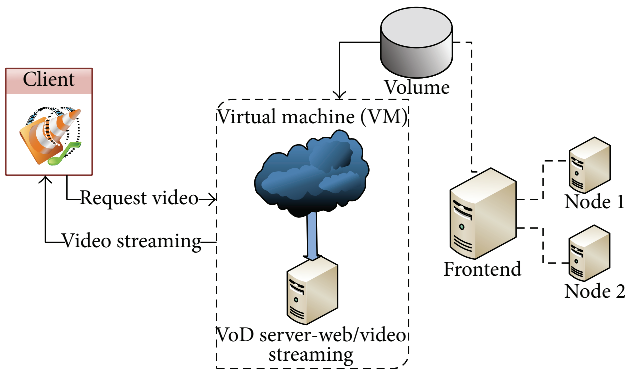

Figure 2: Video streaming service architecture.

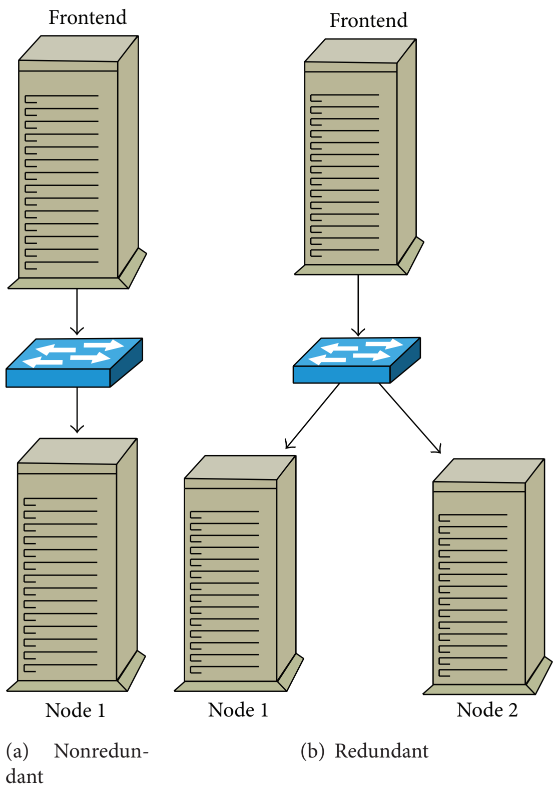

Figure 3: Nonredundant and redundant architectures.

client and Eucalyptus infrastructures. The client connects to the video streaming server via Internet with the VLC software that is installed on their device. The volume component, which stores the videos, is created by the physical resources of the frontend machine. A virtual machine (VM) is instantiated by the physical resources of the NC where the Apache and VLC applications are installed. Apache is a web service which allows visualization of videos over the Internet, and VLC's function in this context is to enable video streaming to the client. The VLC application was chosen because it supports a great variety of video formats.

This VoD service architecture was initially proposed by [8]. Figure 3 illustrates both a nonredundant and a redundant VoD service architecture. The nonredundant architecture, in

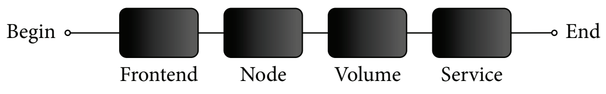

Figure 4: Nonredundant architecture RBD.

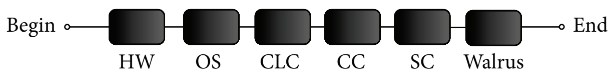

Figure 5: Nonredundant frontend RBD.

Figure 3(a), comprises one frontend and one NC, whereas the redundant architecture, Figure 3(b), includes an additional NC.

The configuration of the frontend machine consists of an i3 Intel processor, 500 GB Sata Hard Disk, 6 GB RAM, and Frontend version of the CentOS operating system. The NC configuration is an i5 Intel processor, 500 GB Sata Hard Disk, 6 GB RAM, and Node version of the CentOS operating system. The system platform is based on the Eucalyptus cloud framework (version 3.2.2) and employs the CentOS operating system (version 6.4).

5. Availability Models

This section describes the availability models designed to represent the redundant and nonredundant systems that are the subject of this work. Quantification of availability for complex IT systems can be achieved by the representation of system states with the hierarchical modeling of RBDs and Markov chains [32].

5.1. Nonredundant Architecture Model. The architecture subsystems in Figure 3 are represented with RBDs and CTMCs. These models are then combined, which constitute a hierarchical model. The nonredundant architecture, as depicted in the top-level RBD of Figure 4, is divided into four parts; frontend, node, volume, and service.

Within the nonredundant architecture, the frontend subsystem is represented by the pure series RBD illustrated in Figure 5. This subsystem consists of hardware (HW) and operating system (OS) and the following Eucalyptus components: CLC (cloud controller), CC (cluster controller), SC (storage controller), and Walrus.

Figure 6 is the RBD model of the node subsystem. Besides the hardware and operating system also present in the frontend, each node requires a hypervisor and a Eucalyptus node controller in order to be available in the cloud [6]. The volume subsystem for video storage is allocated to the frontend.

The greater complexity of the service subsystem requires further refinement with a CTMC (Figure 7). This allows for the calculation of the availability values which will later be considered in the top-level RBD. A CTMC is necessary due to the interdependency between the subsystem components and also means that a closed-form equation of steady-state availability can be obtained, which is useful for sensitivity analysis purposes and for computing the desired measures without incurring a numerical solution.

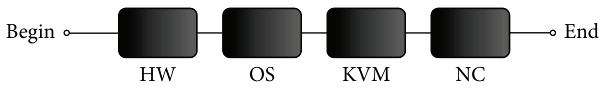

Figure 6: Node RDB.

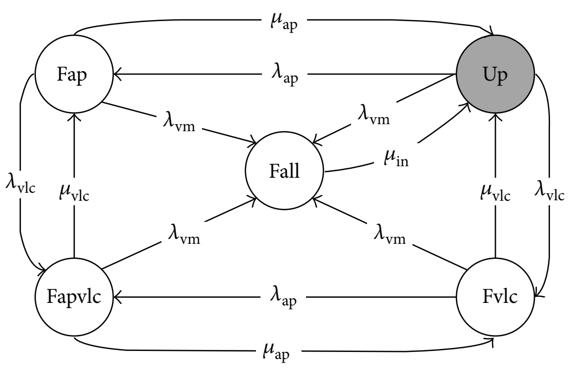

Figure 7: Service module CTMC.

Table 1: Nomenclature of states.

State	Description
Fap	Apache failure, the service is unavailable
Fapvlc	Apache and VLC failure, the service is unavailable
Fall	Failure of all components (VLC, Apache, and VM)
UP	Service available
Fvlc	Failure of the VLC, the service is unavailable

The CTMC in Figure 7 is composed of the following software components: Apache, VLC, and VM. The CTMC has five states: *UP*, *Fap*, *Fapvlc*, *Fvlc,* and *Fall*. The white circles indicate down states (where the service is not available due to failure), and the gray circle indicates the operational state. Only the UP state represents service availability, where all applications (Apache and VLC) are working. From UP the following states can be reached: Apache application failure (Fap), VLC application failure (Fvlc), or virtual machine failure (Fall). At Fap, service is no longer provided, and from this location the other three states can be achieved; VLC application failure (Fvlc), Apache application repair (UP), or virtual machine failure (Fall). The Apache and VLC failure state (Fapvlc) also indicate that the service is unavailable. From Fapvlc the other three states, Fap, Fall, and Fvlc, can be reached. In Fvlc the service is unavailable due the failure of the VLC application, and from this location it is similarly possible to achieve the three other states. The Fall state represents the failure of all components in the subsystem (Apache, VLC, and virtual machine), making it possible for a new VM instantiation to occur, including all necessary applications, which returns the system to the UP state. The nomenclature of all states is compiled in Table 1.

The rates λ_{ap}, λ_{vlc}, and λ_{vm} denote the failure rate of the Apache, VLC, and VM, respectively. The failure and repair rates are defined as $\text{MTTF}_i = 1/\lambda_i$ and $\text{MTTR}_i = 1/\mu_i$ if λ_i and μ_i are constants. The repair rates for Apache and VLC are μ_{ap} and μ_{vlc}. The value μ_{in} is the instantiation rate of a new VM, that is, the reciprocal of the mean time to ativate a VM after being requested. The closed-form equation (5) which is obtained from the CTMC computes the availability of the service block (Figure 4):

$$\begin{aligned} A_s = {} & \left(\mu_{\text{in}}\left(\lambda_{\text{ap}}\lambda_{\text{vm}}(\beta) + \lambda_{\text{ap}}(\beta_1)\mu_{\text{vlc}} + (\beta_1)(\beta_2)(\beta + \mu_{\text{vlc}})\right)\right) \\ & \times \left(\left(\lambda_{\text{ap}} + \beta_1\right)(\lambda_{\text{vm}} + \mu_{\text{in}}) \times (\beta)\left(\lambda_{\text{ap}} + \beta + \mu_{\text{vlc}}\right)\right)^{-1}, \end{aligned} \tag{5}$$

where

$$\begin{gathered} \beta = \lambda_{\text{vlc}} + \lambda_{\text{vm}} + \mu_{\text{ap}}, \\ \beta_1 = \lambda_{\text{vm}} + \mu_{\text{ap}}, \qquad \beta_2 = \lambda_{\text{vm}} + \mu_{\text{vlc}}. \end{gathered} \tag{6}$$

The result is then inserted into (7), with which the availability of the entire system (A) can be computed. In this equation, (A_f), (A_n), (A_v), and (A_s) correspond to the availability of the frontend, node, volume, and service, respectively. Equation (8) uses a notation that is similar to (7), although here A_f, A_n, and A_v are expressed as mean failure (λ_i) and repair (μ_i) rates of the respective subsystems derived from the RBD models. Consider the following:

$$A = A_f \times A_n \times A_v \times A_s, \tag{7}$$

$$A = \left(\prod_{i \in \{f,n,v\}} \frac{\mu_i}{\mu_i + \lambda_i}\right) \times A_s. \tag{8}$$

5.2. Redundant Architecture Model. The warm standby redundant design employs a primary and secondary node. Both have the same hardware and software specifications, but only the primary node is active and receiving workload. The secondary node is switched on but is not receiving or processing any workload. Since this secondary node is in an idle state, it is considered less likely to fail than the active one. When failure occurs in the primary node, the secondary one takes over the service with little or no perceptible interruption to the user.

Figure 8 is the RBD model of the redundant system. In the top-level model the service as well as the node subsystem infrastructure is represented by the service RBD block. However the availability of such a subsystem (service + node subsystem infrastructure) cannot be properly represented by an RBD since the node subsystem implements an active redundant mechanism.

Therefore, the Service RBD block is refined by the CTMC depicted in Figure 9, which represents the service availability of the node subsystem infrastructure. The CTMC comprises the shaded states UUW, UDU, UUD, and UWU (service

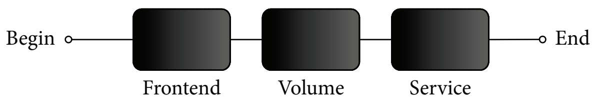

Figure 8: Redundant system top-level RBD.

available) and the white states DDW, DUW, DDD, DWU, DDU, DWD, and DUD (service unavailable).

The notation for the states is based on the current condition of each component. The three letters represent initialisms of the operating condition of the three components, respectively, the service, the first node, and the second node. The service may be up (U) or down (D). The NCs work by being alternately in warm standby mode, and only one of them should be up (U) at any one time, whilst the other either is in warm standby (W) or is down (D). In this model the initial service is represented by UWU, where the service is available, the first node is in warm standby, and the second node is running. From this state it is possible to move to DWU (service failure), DWD (second node failure), or UDU (first node failure). From the DWU state (service down, first node in warm standby, and second node up), UWU (representing service repair), DWD (second node failure), or DDU (first node failure) may be reached.

From state DWD three outcomes are possible; either the failure of all system components (DDD), the initialization of the first node (DUD), or the repair of the second node (DWU). From state UDU (service and second node running), the possible outcomes are DDD (failure of all components), DDU, or UWU. The state UDU can lead to either the failure of all system components (DDD), the repair to waiting state of the first node (DWU), or the instantiation of a new virtual machine with all system applications, making the service available again (UDU).

In state DDD all system components are down; the service is unavailable and the two nodes are unavailable. From this state it is possible to reach two other states; the repair of the first node (DUD) and the repair of the second node (DDU). State DUD represents service unavailability, where service and second node are down, but the first node is up. From this state, three other states can be achieved; failure of all components of the system (DDD), repair of the service (UUD), or repair of the warm standby mode of the second node (DUW). Conversely, state UUD indicates system availability, where the service and the first node are up, but the second node is faulty. From here, the following three states can be reached; DUD (service failure), UUW (repair of the node to warm standby mode), and DDD (since failure of the only functional mode will automatically cause service failure too).

In state DUW the system is unavailable due to service application failure, although the first node is up and the second node is in warm standby mode. From DUW the following states can be reached; DUD (failure of the warm standby node), DDW (first node failure), or UUW (service repair). With service and first node being operational, and the second node being in warm standby, UUW indicates system availability. From this position in the model,

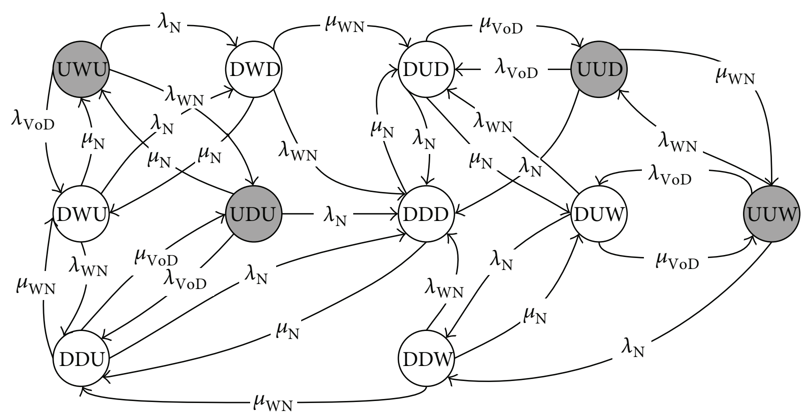

Figure 9: Redundant system CTMC with two nodes.

Table 2: Nomenclature of states.

State	Description
UUD	Service is up, N1 is up, and N2 is down
DDW	Service is down, N1 is down, and N2 is waiting
DUW	Service is down, N1 is up, and N2 is waiting
UDU	Service is up, N1 is down, and N2 is running
UUW	Service is up, N1 is up, and N2 is waiting
DDD	Service is down, N1 is down, and N2 is down
UWU	Service is up, N1 is waiting, and N2 is up
DWU	Service is down, N1 is waiting, and N2 is up
DDU	Service is down, N1 is down, and N2 is up
DWD	Service is down, N1 is waiting, and N2 is down
DUD	Service is down, N1 is up, and N2 is down

the possibilities are warm standby failure (UUD), service failure (DUW), or first node failure, which would cause the service to become unavailable (DDW). The state DDW indicates system unavailability, with service and first node down and second node in warm standby. From DDW, it is possible to reach three other states: failure of all the components of the system (DDD), initialization of the second node (DDU), or repair of the first node (DUW).

The nomenclature for the all states is compiled in Table 2, and the notation of states represents the initial letter of the current condition of each component. The first character denotes the state of the service module, either up (U) or down (D). The second character represents the status of the first node (N1): up (U), down (D), or waiting (W). This waiting condition signifies that the component is in warm standby. The third character refers to the status of the second node (N2): up (U), down (D), or waiting (W).

System failure is an event that occurs when the provided service deviates from the intended service [30, 31]. The failure rates of the two nodes are represented by λ_N, whilst μ_N represents the rate of node repair. A node in warm standby has the failure rate of λ_{WN}, and the repair rate to return it to standby is μ_N. A warm standby node is transformed to available mode at the rate of μ_{WN}. The failure rate of the service application is λ_{VoD}, while the repair rate is μ_{VoD}. The λ_{VoD} was obtained from the inverse of the time to failure of the service module. To calculate this result, we used the CTMC model of Figure 7 and the Mercury tool [30, 31]. The repair rate of the service is considered as the instantiation of a new virtual machine, including all the applications necessary to its operation (Apache and VLC).

Through the CTMC of Figure 9 it is possible to obtain the closed-form equation for calculating the availability of the redundant model (A_{VoD}) as shown below in (9). The equation for calculating the availability of the whole infrastructure of the service (A_{VoD}) can also be obtained from its corresponding RBD model [6]. Consider the following:

$$A_{VoD} = \frac{\alpha\left(2\beta^2\alpha_1 + \beta\alpha_2\beta_1 + \alpha_3\right)\beta_1^2 + \alpha_4\beta_1^3}{\alpha_5\left(\beta^2\alpha_1\left(\lambda_N + 2\mu_N\right) + \beta\varphi\beta_1 + \varphi_1\beta_1^2 + \beta_7\beta^3\right)}, \quad (9)$$

where

$$\begin{aligned}
\beta &= \lambda_{WN},\\
\beta_1 &= \mu_{WN},\\
\alpha &= \mu_N\mu_{VoD},\\
\alpha_1 &= \left(\lambda_N + \beta + \mu_N\right) 2\left(\lambda_N + \beta + \mu_N\right),\\
\alpha_2 &= 4\lambda_{N^2} + 17\lambda_N\beta + 13\beta + 5\lambda\mu_N + 12\beta\mu_N + 2\mu_{N^2},\\
\alpha_3 &= \left(8\beta\left(2\beta + \mu_N\right) + \lambda_N\left(11\beta + \mu_N\right)\right),\\
\alpha_4 &= \left(7\beta + 2\mu_N\right)\left(\mu_{WN^3}\right),\\
\alpha_5 &= \lambda_N + \lambda_{VoD} + \mu_{VoD},\\
\alpha_6 &= 2\lambda_N\left(\lambda_N + \beta\right)\left(\lambda_N + 3\beta\right),\\
\beta_2 &= \left(8\lambda_{N^2} + 24\lambda_N\lambda_{WN} + 13\beta^2\right)\mu_N,\\
\beta_3 &= \left(7\lambda_N + 12\beta\right)\mu_{N^2} + 2\mu_{N^3}\beta_1,\\
\beta_4 &= 2\lambda_N\beta\left(2\lambda_N + 3\beta\right),
\end{aligned}$$

$$\beta_5 = (\lambda_N + \beta)(\lambda_N + 16\beta)\mu_N,$$
$$\beta_6 = (\lambda_N + 8\beta)\mu_{N^2},$$
$$\beta_7 = (\lambda_N(\beta + 2\mu_N) + \mu_N(7\beta + 2\mu_N)),$$
$$\varphi = \alpha_6 + \beta_2 + \beta_3,$$
$$\varphi_1 = \beta_4 + \beta_5 + \beta_6. \quad (10)$$

A closed-form equation for computing the availability of the complete redundant service, $A_{\text{service}R}$, can also be obtained, as demonstrated by (11). A_f and A_v can be computed from the RBD of Figure 8, whilst A_{VoD} is calculated from (9). In this equation, A_f, A_v, and A_{VoD} correspond to the availability of the frontend, volume, and service, respectively. Consider the following:

$$A_{\text{service}R} = A_f \times A_v \times A_{\text{VoD}}. \quad (11)$$

6. Case Study

The case study focused on the analysis of system availability and the identification of components which most affect the streaming service. Initially an availability analysis of a nonredundant architecture (Figure 3(a)) was performed. This was followed by a sensitivity analysis to establish a ranking of the most important parameters of the architecture. This methodology was then applied to a redundant architecture that included an additional node and its service elements (Figure 3(b)). Consequently the availability analysis of the VoD service was undertaken again but this time was implemented in a model of a redundant infrastructure. Finally, a sensitivity analysis of this system was performed and the critical parameters ranked accordingly.

6.1. Evaluation of the Nonredundant Architecture.

6.1. Evaluation of the Nonredundant Architecture. Figure 3(a) depicts the nonredundant architecture, which includes a dedicated frontend machine and another machine for the node. Table 3 compiles the MTTF and MTTR rates for the frontend, node, and volume elements of this nonredundant architecture. These values were obtained from [6, 8, 32]. The computation of dependability metrics for the frontend module produced an MTTF of 180.72 hours and an MTTR of 0.96 hours.

Since the same values of MTTF and MTTR [6, 8] are assumed for the HW and OS elements of the nodes subsystem, Table 3 only includes the parameter values for the KVM and NC blocks [32, 33]. Model analysis of this subsystem produced an MTTF of 481.82 hours and an MTTR of 0.91 hours.

Table 4 contains the parameters for the RBD block model of Figure 4. The availability of the service subsystem is calculated from the CTMC illustrated in Figure 7, and all the values required to solve the CTMC are given in Table 5.

These figures are derived from the analyses done in [6, 32]. The instantiation rate, μ_{in}, is calculated as the sum of the mean time for starting the VM (MTVM) and the mean time for starting the service (MTSS):

$$\mu_{\text{in}} = \text{MTVM} + \text{MTSS}. \quad (12)$$

Monitoring scripts were created for this experiment with Linux utilities such as date and mpstat [34]. The scripts monitor the service initialization. A hundred measurements for MTSS were taken before a normality test was performed, demonstrating with a 95% confidence level that the results do not conflict with variance and standard deviation.

According to the selected parameters a value of 0.9885713 was obtained for the availability of the nonredundant streaming system. This equates to about 100 hours of downtime a year, a figure which highlights the importance of identifying effective solutions to improve the system.

Table 3: Nonredundant system RBD input parameters [6, 8, 32].

Module	Component	MTTF	MTTR
Frontend	HW	8760 h	100 min
	OS	2895 h	1 h
	CLC	788.4 h	1 h
	CC	2788.4 h	1 h
	SC	2788.4 h	1 h
	Walrus	2788.4 h	1 h
Node	KVM	2990 h	1 h
	NC	788.4 h	1 h
Volume	Volume	100000 h	1 h

Table 4: Nonredundant system RBD parameters.

Component	MTTF	MTTR
Frontend	180.72 h	0.96999 h
Node	481.83 h	0.91000 h
Volume	100000 h	1 h
Service	217.77 h	0.92633 h

6.2. Sensitivity Analysis of Nonredundant Architecture. Potential bottlenecks in the system are identified by calculating the relative importance of each component in the system in terms of reliability. By this means it is be possible to determine which components merit additional research and development to improve system reliability [35]. One technique for achieving this was introduced by Birnbaum [36], which measures the importance of the ith component at time t:

$$I_B^i(t) = \frac{\partial h(p(t))}{\partial p_i}, \quad (13)$$

where $I_B^i(t)$ is the index of reliability importance (or Birnbaum Importance) of the component i; p_i is the component reliability of i; and $h(p(t))$ is the reliability of the whole system. Based on this definition, whilst observing that $0 < p_i < 1$, the reliability importance of a component i may be written as follows:

$$I_B^i(t) = h(1_i, p(t)) - h(0_i, p(t)), \quad i = 1, \ldots, n, \quad (14)$$

Table 5: Service module CTMC parameters.

Parameters	Description	Values (h^{-1})
λ_{ap}	Apache failure rate	1/788.4
λ_{vlc}	VLC failure rate	1/336
λ_{vm}	VM failure rate	1/2880
μ_{ap}	Apache repair rate	1
μ_{vlc}	VLC repair rate	1
μ_{in}	Instantiation rate for a new VM	1/0.019166

Table 6: Reliability importance.

Component	Importance value
Frontend	1
Service	0.01618897
Node	$2.64336026 \times 10^{-7}$
Volume	$3.11381368 \times 10^{-11}$

where $p(t)$ represents a vector of component reliability with the removed component ith; 0_i represents the failure condition of component i; and 1_i represents the component in constant operating mode [37]. It follows that $I_B^i(t)$ equates to the probability that component i is critical at time t. In systems with dependent components this observation provides a generalization of the Birnbaum measure [36].

As shown in (14), instead of calculating the partial derivative given in the standard definition, the system reliability is calculated when the component is working and calculated again when the component is in a fail state. The importance measure is then calculated by a simple subtraction.

Table 6 gives the reliability importance of all system components calculated for a time period of 4380 hours or 6 months. These results were calculated from the input parameter values given in Table 4.

The elements with greatest reliability importance for the system are the frontend and service subsystems. Accordingly, a detailed parametric sensitivity analysis of the service subsystem was performed using the proposed CTMC model (Figure 7). The sensitivity indices, for $S_\theta^*(A)$, were obtained with Mercury [30, 31], where (A) is the steady-state availability and (θ) is each system parameter (the MTTF and MTTR of each component). The indices can also be computed with

$$\begin{aligned} S_\theta(A) = {} & \frac{\partial A_f}{\partial \theta} \times A_n \times A_v \times A_s \\ & + A_f \times \frac{\partial A_n}{\partial \theta} \times A_v \times A_s \\ & + A_f \times A_n \times \frac{\partial A_v}{\partial \theta} \times A_s \\ & + A_f \times A_n \times A_v \times \frac{\partial A_s}{\partial \theta}. \end{aligned} \tag{15}$$

The failure and repair rates of the frontend module are represented by λ_f and μ_f, respectively, and the corresponding derivative expressions are given in (16). The derivative expressions for A_n and A_v are similar to those of A_f since those modules are also represented with RBDs. Consider the following:

$$\begin{aligned} \frac{\partial A_f}{\partial \lambda_f} &= -\frac{\mu_f}{\left(\mu_f + \lambda_f\right)^2}, \\ \frac{\partial A_f}{\partial \mu_f} &= -\frac{\mu_f}{\left(\lambda_f + \mu_f\right)^2} + \frac{1}{\lambda_f + \mu_f}. \end{aligned} \tag{16}$$

The corresponding derivative expressions for A_s are given in

$$\begin{aligned} \frac{\partial A_s}{\partial \lambda_{\text{vm}}} = {} & -\left((\beta)\left(\lambda_{\text{vm}} + \mu_{\text{vlc}}(\beta_3) + (\beta_1)\lambda_{\text{ap}}\lambda_{\text{vm}} + \lambda_{\text{ap}}(\beta_3)\mu_{\text{vlc}}\right)\mu_{\text{in}}\right) \\ & \times \left((\beta_1)^2(\beta_2)(\gamma)\left(\lambda_{\text{ap}} + \beta\right)\right)^{-1} \\ & - \left(\left((\beta + \lambda_{\text{vlc}})(\alpha)(\beta_3) + (\beta_1)\lambda_{\text{ap}}\lambda_{\text{vm}} + \lambda_{\text{ap}}(\beta_3)\mu_{\text{vlc}}\right)\mu_{\text{in}}\right) \\ & \times \left((\beta_1)(\beta_2)(\gamma)^2\left(\lambda_{\text{ap}} + \beta\right)\right)^{-1} \\ & - \left(\left((\beta)(\alpha)(\beta_1) + (\beta_1)\lambda_{\text{ap}}\lambda_{\text{vm}} + \lambda_{\text{ap}}(\beta_3)\mu_{\text{vlc}}\right)\mu_{\text{in}}\right) \\ & \times \left((\beta_2)(\beta_2)(\alpha)\left(\lambda_{\text{ap}} + \beta\right)^2\right)^{-1} \\ & + \left(\mu_{\text{in}}(\beta)(\beta_3) + (\beta_3)^2 + \left(\beta(\beta_3) + (\beta_1)\lambda_{\text{ap}} + \vartheta\right)\right) \\ & \times \left((\beta_1)(\beta_2)(\gamma)(\lambda_{\text{ap}} + \beta)\right)^{-1} \\ & - \left((\beta)(\alpha)(\beta_3) + \left(\beta_1\lambda_{\text{ap}}\lambda_{\text{vm}} + \lambda_{\text{ap}}(\beta_3)\mu_{\text{vlc}}\right)\mu_{\text{in}}\right) \\ & \times \left((\beta_1)(\beta_2)^2(\gamma)\left(\lambda_{\text{ap}} + \beta\right)\right)^{-1}, \end{aligned}$$

$$
\begin{aligned}
\frac{\partial A_s}{\partial \lambda_{\text{vlc}}}
&= -\Big(\big(\lambda_{\text{ap}}\mu_{\text{vlc}}(\beta_3) \\
&\qquad +\lambda_{\text{ap}}\lambda_{\text{vm}}(\beta_1)+(\beta)(\alpha)(\beta_3)\big)\mu_{\text{in}}\Big) \\
&\quad \times\Big((\gamma)(\beta_1)^2\big(\lambda_{\text{ap}}+\beta\big)(\beta_2)\Big)^{-1} \\
&\quad + \frac{\big(\lambda_{\text{ap}}\lambda_{\text{vm}}+(\alpha)(\beta_3)\big)\mu_{\text{in}}}{(\gamma)(\beta_1)\big(\lambda_{\text{ap}}+\beta\big)(\beta_2)} \\
&\quad - \Big(\big(\lambda_{\text{ap}}\mu_{\text{vlc}}(\beta_3) + \lambda_{\text{ap}}\lambda_{\text{vm}}(\beta_1) \\
&\qquad + (\beta)(\alpha)\beta_3\big)\mu_{\text{in}}\Big) \\
&\quad \times\Big((\alpha)(\beta_1)\big(\lambda_{\text{ap}}+\beta\big)^2(\beta_2)\Big)^{-1},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial A_s}{\partial \lambda_{\text{ap}}}
&= -\frac{\big((\alpha)(\beta)(\beta_3)+(\beta_1)\lambda_{\text{ap}}\lambda_{\text{vm}}+\lambda_{\text{ap}}\mu_{\text{vlc}}(\beta_3)\big)\mu_{\text{in}}}{(\beta_1)(\gamma)(\beta_2)^2\big(\beta+\lambda_{\text{ap}}\big)} \\
&\quad - \frac{\big((\alpha)(\beta)(\beta_3)+(\beta_1)\lambda_{\text{ap}}\lambda_{\text{vm}}+\kappa(\beta_3)\big)\mu_{\text{in}}}{(\beta_1)(\gamma)(\beta_2)\big(\beta+\lambda_{\text{ap}}\big)^2} \\
&\quad + \frac{\big(\mu_{\text{vlc}}(\beta_3)+(\beta_1)\lambda_{\text{vm}}\big)\mu_{\text{in}}}{(\beta_1)(\gamma)(\beta_2)\big(\beta+\lambda_{\text{ap}}\big)},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial A_s}{\partial \mu_{\text{vlc}}}
&= -\frac{\big((\beta_3)\lambda_{\text{ap}}\mu_{\text{vlc}}+\lambda_{\text{vm}}(\beta_1)\lambda_{\text{ap}}+(\beta_3)(\beta)(\alpha)\big)\mu_{\text{in}}}{(\gamma)(\beta_1)\big(\beta_1+\lambda_{\text{ap}}+\mu_{\text{vlc}}\big)^2(\beta_2)} \\
&\quad + \frac{\mu_{\text{in}}\big((\beta_3)(\alpha)+(\beta_3)(\beta)+(\beta_3)\lambda_{\text{ap}}\big)}{(\gamma)(\beta_1)\big(\beta+\lambda_{\text{ap}}\big)(\beta_2)},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial A_s}{\partial \mu_{\text{in}}}
&= -\frac{\mu_{\text{in}}\big((\beta_3)(\alpha)(\beta)+(\beta_3)\lambda_{\text{ap}}\mu_{\text{vlc}}+(\beta_1)\lambda_{\text{ap}}\lambda_{\text{vm}}\big)}{(\beta_1)(\beta_1)\big(\lambda_{\text{ap}}+\beta\big)(\gamma)^2} \\
&\quad + \frac{(\beta_3)(\beta_3)(\beta)+(\beta_3)\lambda_{\text{ap}}\mu_{\text{vlc}}+(\beta_2)\lambda_{\text{ap}}\lambda_{\text{vm}}}{(\beta_1)(\beta_2)\big(\beta+\lambda_{\text{ap}}\big)(\gamma)},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial A_s}{\partial \mu_{\text{ap}}}
&= -\frac{\mu_{\text{in}}\big((\alpha)(\beta_3)+\lambda_{\text{vm}}\lambda_{\text{ap}}+\mu_{\text{vlc}}\lambda_{\text{ap}}+(\alpha)(\beta)\big)}{(\gamma)(\beta_2)(\beta_1)\big(\beta+\lambda_{\text{ap}}\big)} \\
&\quad - \frac{\big(\lambda_{\text{vm}}(\beta_2)\lambda_{\text{ap}}+\mu_{\text{vlc}}(\beta_3)\lambda_{\text{ap}}+(\alpha)(\beta)(\beta_3)\big)\mu_{\text{in}}}{(\gamma)(\beta_2)(\beta_1)^2\big(\beta+\lambda_{\text{ap}}\big)} \\
&\quad - \frac{\big(\lambda_{\text{vm}}(\beta_1)\lambda_{\text{ap}}+\mu_{\text{vlc}}(\beta_3)\lambda_{\text{ap}}+(\alpha)(\beta)(\beta_3)\big)\mu_{\text{in}}}{(\gamma)(\beta_2)(\beta_1)\big(\beta+\lambda_{\text{ap}}\big)^2} \\
&\quad - \frac{\big(\lambda_{\text{vm}}(\beta_1)\lambda_{\text{ap}}+\mu_{\text{vlc}}(\beta_3)\lambda_{\text{ap}}+(\alpha)(\beta)(\beta_3)\big)\mu_{\text{in}}}{(\gamma)(\beta_2)^2(\beta_1)\big(\beta+\lambda_{\text{ap}}\big)},
\end{aligned}
\tag{17}
$$

where

$$
\begin{aligned}
\beta &= \mu_{\text{ap}}+\mu_{\text{vlc}}+\lambda_{\text{vm}}+\lambda_{\text{vlc}}; \\
\beta_1 &= \mu_{\text{ap}}+\lambda_{\text{vlc}}; \\
\beta_2 &= \mu_{\text{ap}}+\lambda_{\text{ap}}+\lambda_{\text{vm}}; \\
\beta_3 &= \mu_{\text{ap}}+\lambda_{\text{vm}}; \\
\alpha &= \mu_{\text{vlc}}+\lambda_{\text{vm}}; \\
\gamma &= \lambda_{\text{vm}}+\mu_{\text{in}}; \\
\kappa &= \lambda_{\text{ap}}+\mu_{\text{vlc}}; \\
\vartheta &= \lambda_{\text{ap}}\lambda_{\text{vm}}+\kappa.
\end{aligned}
\tag{18}
$$

The measures of interest were calculated from the values given in Table 5. Table 7 shows the sensitivity ranking with respect to the CTMC parameters.

The results are ranked according to the absolute values. Negative values indicate that there is an inverse relationship between the parameters and the system availability. This is the case with MTTF values, where an increase in the parameter value results in a decrease in the availability measure. The table indicates that the VLC failure rate (λ_{vlc}) is the most critical availability parameter, whilst the VLC repair rate (μ_{vlc}) is the second most important, and the first related to system recovery. Clearly it is the VLC component that should be prioritized when considering improvements in the service subsystem. To demonstrate this further, system availability was computed for varying VLC failure rates whilst keeping all other parameters fixed. Figure 10 plots the changes in availability against the steadily increasing MTTF rates. As expected, the availability improves as the time to failure increases. MTTR is of course a critical factor for system availability since it defines component downtime.

Figure 11 depicts system availability and downtime as a function of the MTTR for the VLC application component. As the MTTR increases the availability decreases. Over the range indicated in the graph, there is a reduction in downtime of 23.96 hours a year.

Figures 12 and 13 combine the subsystem results of parameter variation for MTTF and MTTR, respectively. Note the difference in scale between the graphs, where failure rates are in hours whilst recovery rates are in minutes. The position of the plot lines confirms the sensitivity ranking of Table 7. The strong effect that changes in VLC and Apache repair time

Table 7: Sensitivity ranking of availability.

Parameter	SS (A_s)
λ_{vlc}	−0.00296633
μ_{vlc}	0.00296530
λ_{ap}	−0.00126634
μ_{ap}	0.00126590
μ_{in}	0.00005786
λ_{vm}	−0.00005639

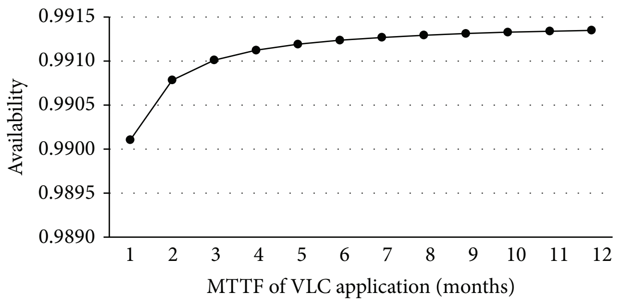

Figure 10: Effect of VLC MTTF changes on system availability.

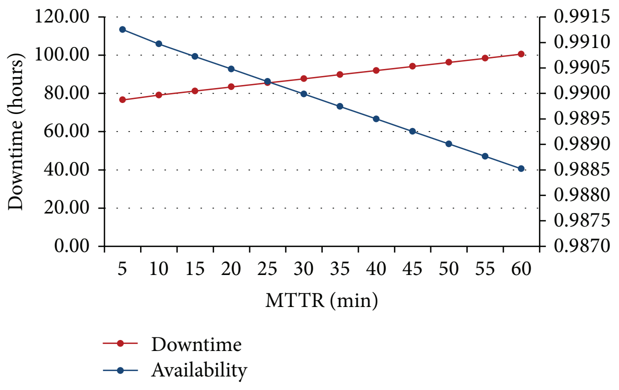

Figure 11: Effect of VLC MTTR changes on system availability and downtime.

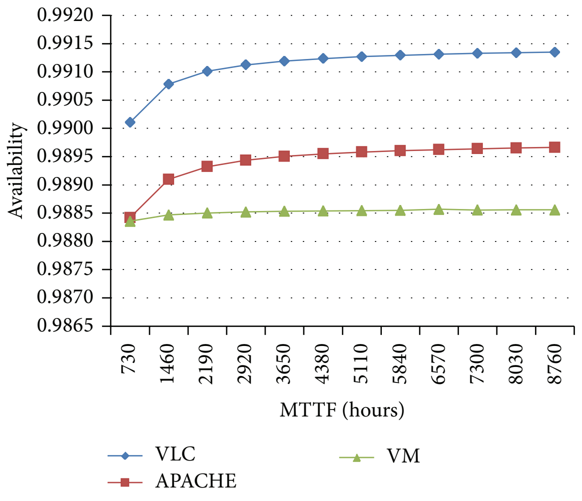

Figure 12: Effect of MTTF changes on system availability.

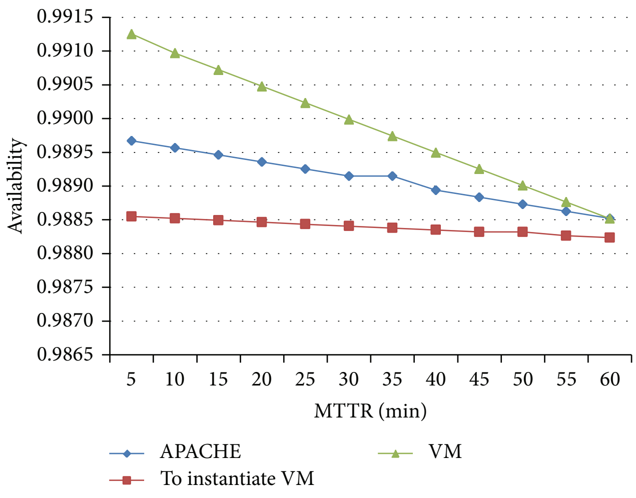

Figure 13: Effect of MTTR changes on system availability.

have on availability should be noted from Figure 13, whereas changes in the instantiation time of a new VM (ie., the repair time from VM failure) have little impact.

Equation (12) is employed to calculate the sensitivity of the system to the frontend and service modules, which were previously identified by the Birnbaum method as the most critical components of the video streaming system. Table 8 compiles the consequent ranking and identifies that frontend failure (λ_f) and repair (μ_f) rates assume the greatest importance in system steady-state availability, since they have the highest sensitivity values. Any change in these parameters will have a major impact on system availability, although in opposite directions. This confirms the results from the calculation of importance indices for the system. Sensitivity with respect to λ_f is negative, since availability increases as its value decreases. In contrast $S^*_{\mu_f}(A)$ is positive with respect to μ_f, since both values increase or decrease together.

Table 8: Sensitivity ranking of frontend and service module availability.

Parameter	SS (A)
λ_f	−0.0000158662
μ_f	0.0000158662
λ_{vlc}	−0.0000088157
μ_{vlc}	0.0000088126
λ_{ap}	−0.0000037634
μ_{ap}	0.0000037621
μ_{in}	0.0000000197
λ_{vm}	−0.0000000154

Table 8 also shows that μ_{in}, which is the instantiation rate of a new VM, has minimal impact on system availability, with only λ_{vm} having less influence.

Table 9: Redundant system RBD parameters [8, 32].

Parameter	MTTF	MTTR
Frontend	180.72	0.969999 h
Volume	100000	1 h
Service	149.98	0.037903 h

Table 10: Redundant system CTMC parameters.

Parameter	Description	Value (h^{-1})
λ_{N}	Mean time to node failure	1/481.83
λ_{WN}	Mean time to standby node failure	1/578.196
μ_{N}	Mean time to node repair	1/0.91
μ_{WN}	Mean time to standby node repair	1/0.0333
λ_{VoD}	Mean time to service failure	1/217.779
μ_{VoD}	Mean time to service instantiate	1/0.0275

6.3. Evaluation of the Redundant Architecture. Analysis of the nonredundant system suggested the addition of a redundant node. Should the primary node fail, the streaming service would continue to run on a VM in the secondary node, and thus the resources of the cloud environment are expanded. Figure 3(b) illustrates the proposed architecture. An availability analysis was performed on this system.

The input parameters are given in Tables 9 and 10 and were obtained from [8, 32].

The parameter λ_{VoD} was obtained from the inverse of the time to failure of the service module with the CTMC model of Figure 7. Since the redundant node is in warm standby the failure rate was assumed to be 20% less than the mean failure rate of an active node [6]. The calculation of μ_{VoD} is achieved with

$$\mu_{\mathrm{VoD}} = T_{\mathrm{NODES}} + \mu_{\mathrm{in}}, \tag{19}$$

where T_{NODES} is the time it takes to activate the node from warm standby. This time is derived from the configuration files of the Hearbeat monitoring software [38]. Heartbeat sends messages from one node to the other; when it detects that host 1 is offline the service is initiated in host 2. All resources and applications are activated in the redundant machine and there is no perceptible interruption or delay to the end user. Rate μ_{in} is derived from (12).

The availability of the redundant system was computed with the parameters compiled in Table 10. Figure 14 is a summary of steady-state availability and downtime for architecture A (nonredundant) and architecture B (redundant). Availability increases from 0.9885713 for A to 0.994401 for B. This clearly indicates that significant system improvement was achieved after the implementation of warm standby redundancy. When discussed in terms of downtime this improvement is even more evident: downtime is 100.11 hours for A and 49.04 hours for B, which equates to a decrease of approximately 51.01%. Therefore it can be concluded that the inclusion of redundancy successfully enhanced the availability, resource level, and reliability of the system.

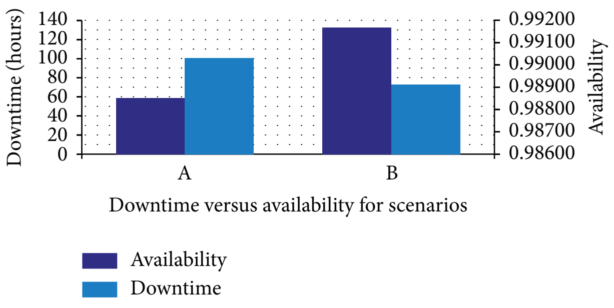

Figure 14: Availability and downtime for nonredundant and redundant architectures.

Table 11: Redundant architecture sensitivity analysis.

Parameter	SS (A)
μ_f	0.005348791537
λ_f	0.005348791537
μ_{VoD}	0.000184041927
λ_{N}	0.000128199816
λ_{VoD}	0.000126991728
μ_{WN}	0.000065313366
μ_{Vol}	−0.00001001878
λ_{Vol}	0.000010018748
μ_{N}	−0.00000748384
λ_{WN}	0.000001588841

6.4. Sensitivity Analysis of Redundant Architecture. In Figure 3(b) there is one frontend and two nodes, representing a redundant system employing a warm standby mechanism. This system's architecture is modeled in Figures 8 and 9. Parametric sensitivity analysis was performed on these models to identify the most critical components of this VoD redundant architecture. The sensitivity indices were computed with the following equation:

$$\begin{aligned} S_\theta(A) = {} & \frac{\partial A_f}{\partial \theta} \times A_v \times A_s \\ & + A_f \times \frac{\partial A_v}{\partial \theta} \times A_s \\ & + A_f \times A_v \times \frac{\partial A_s}{\partial \theta}. \end{aligned} \tag{20}$$

The sensitivity ranking in Table 11 indicates that frontend repair rate μ_f and failure rate λ_f are the two most critical availability parameters, proving that this model is the most critical point for improving system availability. The table also shows that activation of the waiting node λ_{WN} has the second lowest impact on system availability: only μ_{N} has less impact. The sensitivity analysis therefore suggests applying redundancy to the frontend. The graph of Figure 15 illustrates the changing behavior of availability produced by varying frontend failure rates. Figure 16 does the same for repair rates, showing system availability as a function of varying repair rates. This graph also includes the downtime rates, and over

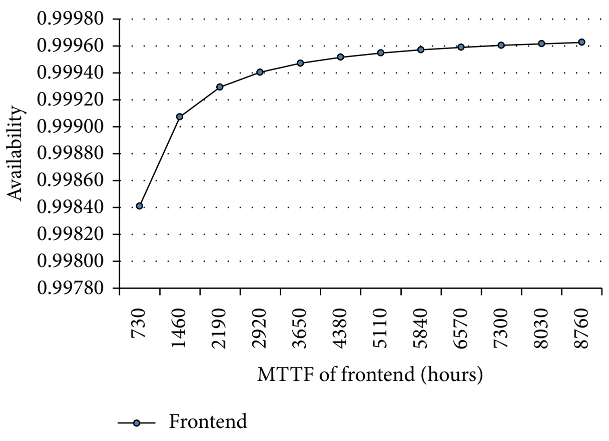

FIGURE 15: Effect of failure times on system availability.

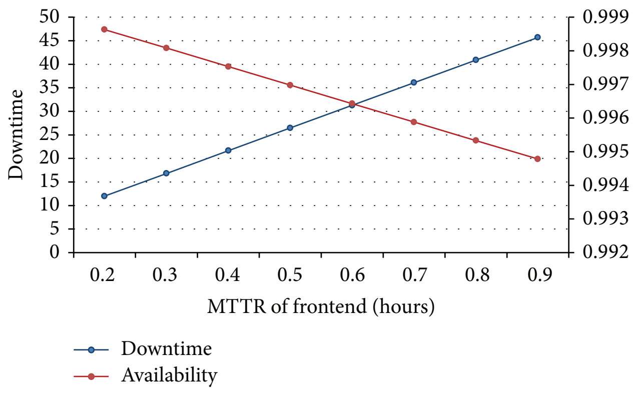

FIGURE 16: Effect of recovery times on system availability.

the range of the graph the decreasing MTTR rates cause a reduction in downtime of 33.71 hours a year.

7. Conclusion

This paper assessed the benefits of employing redundancy in a cloud-based video streaming service. A hierarchical analytical model strategy was developed with reliability block diagrams (RBD) and continuous time Markov chains (CTMCs) to calculate the availability of the system. Furthermore, a parametric sensitivity analysis was proposed to identify which system parameters had the most influence on availability measures. By this means weaknesses were revealed in the system and guidelines for system improvements were developed. A significant increase in service availability was observed from the inclusion of a redundant node. The increase in availability from 0.988571 to 0.994401 equates to a reduced downtime of 51.01%.

The parametric sensitivity analysis performed for the service subsystem identified the failure rate of the VLC application as the most important system parameter when availability is the measure of interest. Conversely, parametric sensitivity analysis performed on the frontend and service subsystems identified three important rates; frontend failure and repair rates and VLC application failure. Analysis of the complete redundant architecture placed frontend repair and failure rates at the top of the sensitivity ranking, providing further proof that this is a key component to be considered for system improvements. The results of analyses were validated by varying the rates of each parameter whilst keeping the others fixed and producing graphs which illustrated the corresponding change in the measure of interest.

Future work will consider the extension of sensitivity analysis to all VoD service subsystems. Other study scenarios will be conceived, such as the analysis of performability issues relating to a video streaming service that runs in parallel on a number of VMs.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] M. Armbrust, A. Fox, R. Griffith et al., “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] W. Zhu, C. Luo, J. Wang, and S. Li, “Multimedia cloud computing,” *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011.

[3] Netflix, “Lessons netflix learned from the aws outage,” The Netflix Tech Blog, 2014, http://techblog.netflix.com/2011/04/lessonsnetflix-learned-from-aws-outage.html.

[4] R. Chen and F. B. Bastani, “Warm standby in hierarchically structured process-control programs,” *IEEE Transactions on Software Engineering*, vol. 20, no. 8, pp. 658–663, 1994.

[5] S. Chuob, M. Pokharel, and J. S. Park, “Modeling and analysis of cloud computing availability based on eucalyptus platform for e-government data center,” in *Proceedings of the 5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS ’11)*, pp. 289–296, 2011.

[6] J. Dantas, R. Matos, J. Araujo, and P. Maciel, “An availability model for eucalyptus platform: an analysis of warm-standy replication mechanism,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC ’)*, pp. 1664–1669, 2012.

[7] R. Matos, P. R. M. Maciel, F. Machida, D. S. Kim, and K. S. Trivedi, “Sensitivity analysis of server virtualized system availability,” *IEEE Transactions on Reliability*, vol. 61, no. 4, pp. 994–1006, 2012.

[8] M. C. Bezerra, R. Melo, J. Dantas, P. Maciel, and F. Vieira, “Availability modeling and analysis of a vod service for eucalyptus platform,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2014.

[9] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi, “A scalable availability model for infrastructure-as-a-service cloud,” in *Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems & Networks*, pp. 335–346, 2011.

[10] D. Bruneo, G. Iellamo, G. Minutoli, and A. Puliafito, “Gridvideo: a practical example of nonscientific application on the grid,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 5, pp. 666–680, 2009.

[11] R. Ghosh, F. Longo, F. Frattini, S. Russo, and K. S. Trivedi, "Scalable analytics for iaas cloud availability," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 57–70, 2014.

[12] D. Bruneo, S. Distefano, F. Longo, A. Puliafito, and M. Scarpa, "Workload-based software rejuvenation in cloud systems," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1072–1085, 2013.

[13] R. Matos, J. Araujo, D. Oliveira, P. Maciel, and K. Trivedi, "Sensitivity analysis of a hierarchical model of mobile cloud computing," *Simulation Modelling Practice and Theory*, 2014.

[14] F. C. Cardoso, "Concepts of virtual private network for secure streaming video," 2010.

[15] Version 2.0., "Eucalyptus cloud computing platform - administrator guide," Tech. Rep., Eucalyptus Systems, 2010.

[16] *Introduction to Cloud Computing Architecture*, Sun Microsystems, 2009.

[17] G. D. Delgado, V. C. Frías, and M. A. Igartua, "Video-streaming transmission with qos over cross-layered ad hoc networks," in *International Conference on Software in Telecommunications and Computer Networks (SoftCOM '06)*, pp. 102–106, IEEE, 2006.

[18] D. Diaz-Sanchez, F. Almenarez, A. Marín, D. Proserpio, and P. Arias Cabarcos, "Media cloud: an open cloud computing middleware for content management," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 970–978, 2011.

[19] A. M. A. Valeriana and R. Marcelo, *Monitoramento do Protocolo Rtsp (Real Time Streaming Protocol) Utilizando Ntop (Network Top)*, Centro Brasileiro de Pesquisas Físicas, 2008.

[20] D. Wu, Y. T. Hou, W. Zhu, Y. Q. Zhang, and J. M. Peha, "Streaming video over the internet: approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, 2001.

[21] J. C. Laprie, *Dependable Computing and Fault Tolerance: Concepts Terminology*, IEEE, 1995.

[22] S. Schaffer, *Babbage's Intelligence: Calculating Engines and the Factory System*, Critical Inquiry—The University of Chicago Press, 1994.

[23] P. Maciel, K. S. Trivedi, R. Matias, and D. S. Kim, *Dependability Modeling*, IGI Global, Hershey, Pa, USA, 2011.

[24] J. Laprie, *Dependability: Basic Concepts and Terminology*, Springer, 1992.

[25] P. M. Frank, *Introduction to System Sensitivity Theory*, Academic Press, 1978.

[26] D. Johnson, K. Murari, M. Rju, R. B. Suseendran, and Y. Andgirikumar, *Eucalyptus Beginner's Guide. Uec Edition for Ubuntu Server 10.04—Lucid Lynx, V1.0*, 2010.

[27] A. Valmari, *The State Explosion Problem*, Springer, 1998.

[28] D. M. Hamby, "A review of techniques for parameter sensitivity analysis of environmental models," *Environmental Monitoring and Assessment*, vol. 32, no. 2, pp. 135–154, 1994.

[29] R. Matos Jr., *An automated approach for systems performance and dependability improvement through sensitivity analysis of markov chains [Ph.D. dissertation]*, 2011.

[30] B. Silva, G. Callou, E. A. G. Tavares et al., "Astro: an integrated environment for dependability and sustainability evaluation," *Sustainable Computing: Informatics and Systems*, vol. 2, pp. 1–31, 2012.

[31] Mercury tool developed by modcs group, https://sites.google.com/site/mercurytooldownload/.

[32] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability modeling and analysis of a virtualized system," in *Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC '09)*, pp. 365–371, 2009.

[33] T. Hu, M. Guo, S. Guo et al., "Mttf of composite web services," in *Proceedings of the International Symposium on Parallel and Distributed Processing with Applications (ISPA '10)*, pp. 130–137, IEEE, 2010.

[34] R. Blum, *Linux Command Line and Shell Scripting Bible*, 2000.

[35] Z. Birnbaum, "On the importance of different components in a multicomponent system," in *Multivariate Analysis—II*, pp. 581–592, 1968.

[36] A. B. Huseby, "Importance measures for multicomponent binary systems," Statistical Research Report, 2004.

[37] J. J. C. de Figueirêdo, *Análise de dependabilidade de sistemas data center baseada em indices de importância [Dissertações de Mestrado]*, 2011.

[38] Heartbeat, linux-HA Project, 2014, http://www.linuxha.org/.