The Journal of Supercomputing (2022) 78:9985–10024
https://doi.org/10.1007/s11227-021-04217-1

Availability evaluation of system service hosted in private cloud computing through hierarchical modeling process

Danilo Clemente[1] · **Paulo Pereira**[1] · **Jamilson Dantas**[1] · **Paulo Maciel**[1]

Accepted: 16 November 2021 / Published online: 20 January 2022

Abstract

Cloud computing provides an abstraction of the physical tiers, allowing a sense of infinite resources. However, the physical resources are not unlimited and need to be used more assertively. The challenge of cloud computing is to improve the use of resources without jeopardizing the availability of environments. Stochastic models can efficiently evaluate cloud computing systems, which is needed for proper capacity planning. This paper proposes an availability evaluation from a system hosted on a private cloud. To achieve this goal, we created hierarchical models to represent the studied environment. Sensitivity analysis is performed to identify the most influential parameters and components that must be compatible with improving system availability. A case study supports the demonstration of the accuracy and utility of our methodology. We propose structural changes in the environment using different redundancies in the components to obtain satisfactory results. Finally, we analyze scenarios regarding DC's redundancy.

Keywords Availability evaluation · Stochastic petri net · Cloud computing · Stochastic models · Analytical models · Sensitivity analysis · Analytical models · Data center redundancy

✉ Danilo Clemente
danilo.clemente@ufpe.br

Paulo Pereira
prps@cin.ufpe.br

Jamilson Dantas
jrd@cin.ufpe.br

Paulo Maciel
prmm@cin.ufpe.br

[1] Centro de informática, Universidade Federal de Pernambuco, Recife, PE, Brazil

1 Introduction

Cloud computing has brought several benefits to its users, such as reliability and high availability. The data centers (DC) responsible for hosting cloud computing have many features of redundancy and fault tolerance of hardware and software, offering more capacity and better use of resources. According to [11], there are four cloud computing models: private cloud, community cloud, public cloud, and hybrid cloud. Each model has particular characteristics with its advantages and disadvantages.

To improve the system's availability, the replication of components and subsystems is a necessary action [6]. These components can be analyzed and replicated in a macro way, such as DC replication, electrical structure, a system of cooling and also micro way, representing internal redundancies in the DC such as servers, virtual clusters, and applications [31].

The private cloud is a model widely adopted in companies that intend to use facilities such as scalability and availability in their domains. In this way, they can offer their internal customers an agile and flexible infrastructure while still having complete control over customer data [24]. Furthermore, private clouds can also complement the local infrastructure interacting with a public cloud. Getting the hybrid cloud rating.

Reliability and availability are widely used metrics to assess the degree of operability of a system or component [5]. Combined with modeling, it has a high capacity to improve operational costs and infrastructure planning. Finally, hierarchical modeling provides the ability to model a wide range of systems and subsystems.

The related works considered in this study address stochastic models to evaluate the availability of critical systems hosted in private cloud environments. Therefore, we can identify different modeling techniques that could be used in our scenarios.

Sousa et al. [39] proposed models able to represent cloud infrastructures with different redundancy mechanisms, such as cold standby, hot standby, warm standby, and active-active redundancy mechanism, as well as allowing the assessment of the respective impact on availability and downtime. In addition, they built hierarchical modeling representing the Eucalyptus platform as the cloud computing framework.

However, although the authors proposed models representing physical and virtual machines (VMs) and management modules of the cloud infrastructure. They did not represent the VM's migration, did not identify which server the virtual machine is hosted on, nor did perform sensitivity analysis.

This paper proposes an availability evaluation from a system hosted on a private cloud. We created hierarchical models with the goal improvement the system's availability. Specifically, our contributions are:

- Evaluation methodology that systematically conducts the study
- We proposed parametric sensitivity analysis on a hierarchical analytical model and used the Bootstrapping technique to calculate the confidence interval (CI)

- SPN models contemplating different types of component redundancy (hot and cold standby), hot and cold VM migrations, DC redundancy (active/active and active/standby) and active DC control
- Hierarchical models to estimate and plan the availability of system hosted in a private cloud with DC redundancy

The remainder of the paper is organized as follows: Sect. 2 explains the main concepts related to Availability measures and Models, sensitivity analysis, Data Center Configurations, and Data Center redundancy. Section 3 summarizes the related works found in the literature review. Section 4 presents the methodology used by work. Section 5 presents the physical and virtual architecture of the system, and Data Center Structure. Section 6 builds the models that represent the current environment studied. Section7 shows the case studies where we perform the sensitivity analysis, availability evaluation models and propose a better environment structure to increase the availability metric, including DC's redundancy. Section 8 contains the final remarks on this work.

2 Background

In this section, we review the basic concepts of cloud computing and the availability assessment paradigm. These concepts are necessary to understand our proposal, including the aspects that involve the case studies.

2.1 Dependability and redundancy in high availability

Systems dependability can be understood as the ability to deliver a specified functionality that can be justifiably trusted [3]. An alternate definition of dependability is "the ability of a system to avoid failures that are more frequent or more severe, and outage durations that are longer than is acceptable to the user" [3]. Dependability encompasses measures such as reliability, availability, and safety. Due to the ubiquitous provision of services on the Internet and on cloud systems, dependability has become an attribute of prime concern in hardware/software development, deployment, and operation [21], since such services require high availability, stability, fault tolerance, and dynamical extensibility.

Many techniques have been proposed and adopted to build failover clusters [19] as well as to leverage virtualization and cloud systems for addressing service dependability issues. Many of those techniques are based on redundancy, i.e., the replication of components so that they work for a common purpose, ensuring data security and availability even in the event of some component failure.

Three replication techniques deserve special attention due to its extensive use in clustered server infrastructures [21]: *Cold Standby, Hot Standby, and Warm Standby*. In the *Cold Standby* technique, the backup nodes are turned off on standby and will only be activated if the primary node fails. The positive point for this technique is that the secondary node has low consumption of energy and do not wear

the system. On the other hand, the secondary node needs significant time to be activated, incurring in data loss, or long delays, in active user sessions, as well as rejection of new user requests. The *Hot Standby* may be considered the most transparent of the replication modes. The replicated modules are synchronized with the operating module, thereby, the active and standby cluster participants are seen by the end user as a single resource. The change of equipment is not noticed when the primary node breaks. The *Warm Standby* technique tries to balance the costs and the recovery time delay of Cold and Hot Standby techniques. The secondary node is on standby, but not completely turned off, so it can be activated faster than in the Cold Standby technique. The replicated node is partially synchronized with the operating node, so users may lose some information in the exact moment of the switchover to the primary node.

2.2 Availability measures and models

Availability (A) as a measure of assessing operability in computing service has been studied for a long time [4, 14]. Device improvements have increased computer system availability at the pass of time. Previously, by 1980, well-run computer systems offered 99 percent availability. This sounds good but corresponds a proximately 100 minutes of downtime per week. Such outages may be acceptable for systems that are classified as not critical. On the other hand, mission-critical and online applications cannot tolerate this downtime. They require high-availability systems that deliver 99.999 percent availability: at most five minutes of service interruption per year.

The system's availability may be represented by a ratio between the mean time to failure (MTTF) and mean time to repair (MTTR) of the system (Eq.1) [43].

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

A variation of this equation can be used to find the MTTF value

$$MTTF = \frac{A \times MTTR}{1 - A}. \tag{2}$$

The system's MTTF may be computed by Eq.3, where **R(t)** is the reliability of that system as a function of elapsed time. Equation.4 provides a way of computing the MTTR from the values of MTTF, availability, and unavailability (UA = 1 − *A*) [43].

$$MTTF = \int_0^\infty R(t)dt \tag{3}$$

$$MTTR = MTTF \times \frac{UA}{A} \tag{4}$$

The Reliability Block Diagram (RBD) [8, 17] is a graphical analysis technique representing systems or components as blocks and their functional relationships as connections between these blocks. It is considered a diagram oriented to success, as

it does not contemplate the capacity or quantity of the repair team. It only considers when a component takes back to a functional state [6]. The blocks within the block diagram are linked depending on their effects on the system. There may be serial connections, with logical representation *AND*, parallel connections, with logical representation *OR* and k-out-of-n (KooN) configuration represents the number of components that must be in the operational state for the block to be operational. All of these conditions directly affect the system availability and can be calculated according to probabilistic principles. The availability calculation of a system or subsystem depends on the redundancy of each component. The representation of the availability calculation for components in serial mode is shown in Eq. 5.

$$A_{(s)} = \prod_{i=1}^{n} A_{i(s)} \tag{5}$$

The Petri Net (PN) is a concept introduced in [34]. According to [29], it is a visual paradigm for the formal description of logical interactions between the parts or the flow of activities in complex systems. Some works can be found at [32, 33]. Initially, the PN does not use timing in its modulation. However, we need this feature to be able to perform reliability and availability analysis. Therefore, this work uses an extension knowns as the Stochastic Petri Net (SPN) that has been studied extensively [22, 28], where times can be associated with transactions.

According to [1], the SPNs are obtained by associating with each transition in a PN an exponentially distributed firing time. The authors have shown that SPNs correspond to continuous-time Markov chains (CTMC) due to the memoryless property of the exponential distribution of firing times. Hence, SPN markings correspond to CTMC states.

The SPNs are extensively used in probabilistic models for performance analysts. It is a handy tool for analyzing computer systems since they allow the system operations to be precisely described through a graph that translates into a Markovian model useful for obtaining performance estimates. The SPN model permits the calculation of the steady-state probabilities and analysis of performance measures as average delay and average throughput. All of this analysis is performed using the equivalent Markov model.

The SPN model works with places and transitions. Places represent states or conditions of the system, while transitions represent events, which may or may not cause changes in the system. Other components are bows and tokens. Whenever tokens exist in a location represent an active state.

2.3 Sensitivity analysis

The objective of sensitivity analysis (SA) is to quantify parameter variations on calculated results. Terms as influence, importance, ranking by significance, and dominance are all related to sensitivity analysis. It can be considered as a formal method for evaluating data and models to determine which factors are most influential in a system [15].

A typical approach to model evaluation involves performing computations with specific input parameter values to produce output values and scatter plots. Thus, the scientific goal of sensitivity analysis is not to confirm preconceived notions, such as about the relative importance of specific inputs, but to discover and quantify the most important features of the models under investigation [23].

The systematic methodology for performing sensitivity analysis utilized in this work analyzes changes in data distribution and their impact on the system. First, identify which component of the system has the most significant interference in the final metric [24]. When a slight change in a component of the system results in a significant variation in the final metric, it is knowns that the system is susceptible to this parameter. Some sensitivity analysis techniques have been developed and reported in the literature [23, 26]. In this paper, we employ a percentage difference technique for computing the sensitivity index $S_y(A)$, which indicates the impact on a given in availability caused by variations in an input parameter y. Equation 6 shows how the index of the sensitivity analysis is calculated for the y metric, where max_y and min_y represent the maximum and minimum output values, respectively, of the calculation varying the parameter y over the value the maximum value max_y.

$$S_y(A) = \frac{max_y - min_y}{max_y} \tag{6}$$

While calculating $S_y(A)$, the model's other parameters need to be fixed. Thus, it is performed for all parameters to be calculated and to build the sensitivity analysis classification. This classification improves the predictability of increased availability.

2.4 Data center configurations

The DC is a harmonious functioning set of interconnected systems. It uses technologies that serve only one purpose: provide operating conditions for the hosted system. Currently, DCs are designed to serve mainly two kinds of services; cloud computing and big data. They are planned following various design methodologies. There have three main components [36]: *Equipaments Information Technology (IT), Power Supply, and Cooling system*.

The structure of the DC needs to be carefully crafted to best address its principal goal. According to [13], some international guidelines guide the physical structure of the DC. The most renowned are Telecommunications Infrastructure Standard for Data Center (ANSI/TIA-942-A). It covers DC infrastructure considering redundancy, telecommunication, architectural, electrical, and mechanical.

Based on these, an Uptime Institute[1], a professional services organization specializing in DCs, and the Telecommunications Industry Association (TIA), advocate a 4-tier classification loosely based on the power supply, UPS, cooling system, and redundancy of the DC [42].

[1] https://uptimeinstitute.com.

2.4.1 Tier I data center: basic system

Have a single path for power supply, UPS, and cooling system. It does not have redundant components (neither physical nor logical). It provides for a minimal level of load distribution with little or no redundancy. A failure or a maintenance stoppage can lead to service interruption. The DC's project must contain single or many cooling systems but without redundancy. The DC's availability is 99.771%, with 28,8 hours of downtime per year [20].

2.4.2 Tier II data center: redundant components

Increase availability by adding redundant components to the previous tier. It provides for Partial redundancy in power, cooling, and networking (LAN and SAN). The possible point of failure of this tier is related to the cooling and power system. According to [20], availability is 99,741%, with an experience of 22 hours of downtime per year.

2.4.3 Tier III Data center: concurrently maintainable

Known as the Self Sustained System. It has two paths to utilities, and each has redundant components. It provides redundancy even during maintenance. The only point of failure is the distribution room, where Core, LAN, and SAN switches are installed. Also, according to the study, the DC's availability is 99.982 %, with a maximum of 1.6 hours of downtime per year.

2.4.4 Tier IV data center: fault-tolerant

known as High Fault Tolerance. It has two simultaneously active power and cooling distribution paths with redundant components. It is supposed to tolerate any single equipment failure without impacting the load. Tier IV DC typically serves large corporations (cloud hosting) and, according to [20], provides 99.995% availability and 26.3 minutes of annual downtime.

2.5 Cloud data center configurations

Cloud computing has brought about a geographic shift in computing. It has become known as computing on-demand, software as a service, infrastructure as a service, among others [16]. The physical location of information and resources is no longer paramount. The most important is that the data must be accessible in a reliable way at the desired time. To meet the demands of many users, according to [11], it is necessary to share computing resources, allowing rapid provisioning and staggering. The cloud computing DC needs some features [9, 13]; *Agility, Resiliency, Modularity, Scalability, Reliability, Availability, Sustainability, and Low cost*. There are four cloud computing models: private cloud, community cloud, public cloud, and hybrid

cloud. Each has particular characteristics with its advantages and disadvantages. The most adopted models in cloud computing are a software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Data storage as a service (DaaS). Cloud computing is an up-and-coming trend that allows elasticity, high performance, low cost, and high availability regardless of the model adopted.

3 Related works

Hierarchical modeling to increase availability is studied extensively. In this section, we summarize some studies related to modeling and availability evaluation in cloud computing environments. Mesbahi et al. [27] present solutions on high availability in a cloud environment by proposing a roadmap of all the studies necessary to achieve good reliability and availability. The work presents the importance of combinatorial models, models of state space, and hierarchical models. Torquato et al., in the article [41] exposes the problem of software aging and the impact on the availability of the environment. It presents SPN availability models and uses redundancy techniques such as Warm-Standby and Cold-Standby to defend that virtual machines' hot migration contributes to the software's rejuvenation.

The work by Callou et al. [7] is based on cost, sustainability, and availability analysis of DC in information technology (IT). They developed the study aiming at energy savings and minimizing high costs from DC. The authors proposed availability models in RBD and SPN to represent the DC combined with the energy flow model to increase energy savings. Applying their proposal, the authors improved the availability with a slight increase in cost and sustainability.

Melo et al. [26], assessed the capacity-oriented availability (COA) of a private cloud. The focus of the study was to make better use of the physical resources of the infrastructure. It performs modeling in RBD and SPN to support its studies. The results were satisfactory; however, they did not consider the uses of virtual and physical clusters or the migration of virtual machines to other physical resources. Torquato et al. [40] also did not consider virtual machine migrations in theirs work. However, they did present models elaborated and mathematically proven in RBD and stochastic reward networks (SRN). The main goal was to evaluate a virtual data center (VDC) availability, and they have exposed availability and COA calculations. The study results identify the limit for the increase in availability caused by the addition of VMs. The addition of physical resources and VMs from this point on becomes a waste of resources. On the other hand, it significantly increases the COA.

Matos et al. [24] propose a hierarchical availability evaluation model, represented by RBD, CTMC, and SPN that correspond to a private cloud of the architecture of Eucalyptus-based environments. These models contemplate fault-tolerance solutions like warm-standby redundant hosts for some of its main components. They elaborate mathematically on two execution forms of sensitivity analysis. According to the authors, the differential sensitivity analysis may also be used for the availability and performance evaluation of different kinds of systems. The technique is instrumental in analyzing systems with many components and events when other sensitivity analysis methods only provide a partial view of

the influence of each parameter. Andrade et al. in work [2] also presents a study of availability and sensitivity analysis. The focus is on evaluating a disaster recovery as a service (DRaaS) solution addressing downtime and costs. A sensitivity study was also carried out to identify which parameters have the most significant impact on availability. The work presents several well-structured and segmented SPN models of the service infrastructure and the DC using a real scenario.

Rosando et al. [37] indicate that 25% of downtime on DCs is caused by power outages. The study presents stochastic models in RBD and SPN, using the technique of estimating availability based on energy supply and the IT subsystem. The study uses DCs classified by TIA-942 (Tier I to IV). The authors argue that the use of redundant components in energy and IT systems drastically reduces service downtime. The improvement in availability depends on the classification of the DC. In addition, there is a 36.28% improvement in the availability of a Level IV DC relative to a Level I DC. This gain is equal to approximately 19.65 h less in the environment's downtime. The work presents significant conclusions on improving the availability of IT services on DCs and presents well-designed foundations and models.

According to Santos et al. [38], a DC can be divided into three main subsystems; cooling, power supply, and information technology (IT). The authors explain that these systems are independent, but they can interfere with each other's availability. The study proposes RBD and SPN models to evaluate the service's availability that is hosted on a DC that uses cloud configuration. The authors compare the availability of the IT subsystem hosted on DCs classified as Tier I and Tier IV. The conclusion of the work exposes that on DC Tier I, the MTTR of the edge router is the metric that has the greatest impact on availability. In DC Tier IV, the server's MTTR and MTTF are the metrics that most interfere in availability, followed by the edge router's MTTR. The study ends by stating that the system hosted in a DC Tier I has availability of 99.78% and in a DC Tier IV has 99.90%.

In [10] Dhanujati et al. present a study on disaster recovery using a real company. The company operates in the electricity business in Indonesia, with approximately 37 million users. The study provides a foundation and an elaboration of the necessary infrastructure so that the service can always be operational. Furthermore, the authors consider preserving current transitions even in the event of failure of the primary DC. Thus, the paper has elaborate foundations. But, on the other hand, it does not model the environment, calculate availability, or a strategy to be used in a disaster.

The main difference between our work and previous ones is that we performed a complete availability analysis of an application hosted in a private cloud. We started with the study of the availability of physical and logical components and evolved to the hosting level in a redundant DC. We propose analytic availability models, verifying availability impact by adding physical nodes, application instances to the system, and a redundant DC. Other features of our proposed model are live and reactive migrations of virtual machines (respecting affinity rules); use of hot and cold standby redundancy; component synthesis, and active DC control.

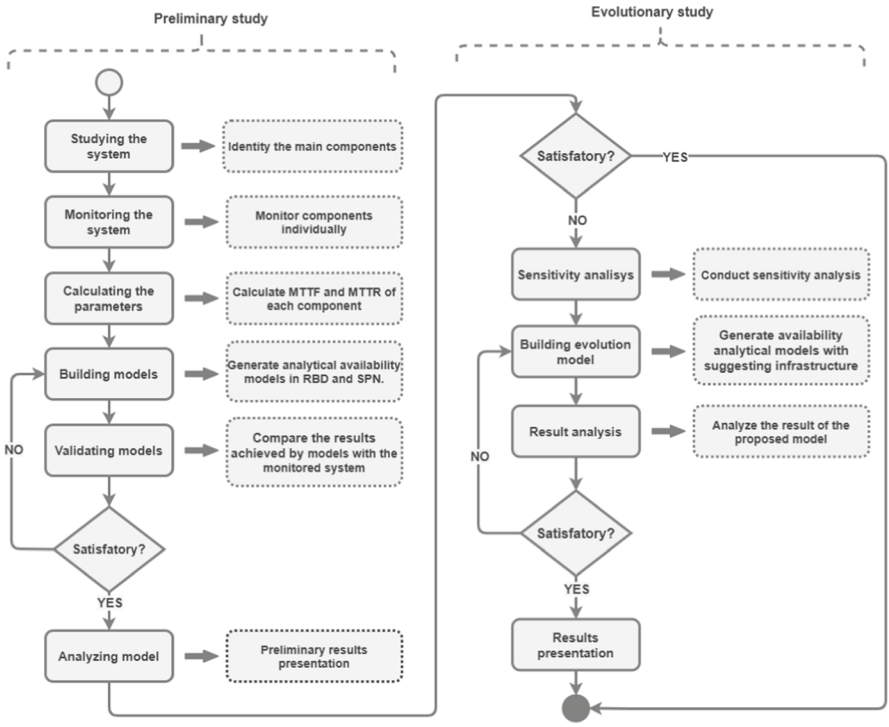

Fig. 1 Supporting methodology

4 Evaluation methodology: an overview

We introduce the followed methodology in this work, applying a logical consistency to reach our main objective and how this work can be replicated [18]. The methodology is shown in Fig. 1. It is divided into two major groups, Preliminary study, and Evolutionary study. The Preliminary study analyzes the current system and its functioning, builds models representing it, and validates if it is consistent with reality [25, 30]. In the Evolutionary study, we identified points that could be improved and evolved the model by applying changes to the environment's infrastructure. The flowchart contains two types of activities, macro-task, represented by boxes, and micro-task, represented by dashed boxes. We only go to the subsequent step after the conclusion of the current one. The rhombus represents a step that can lead to two different pathways. This decision varies depending on the timing of the methodology. The first decision is satisfactory whether the result achieved by the model is within the confidence interval. It is not the case; it returns to the building models task to adjust in the model. The second and third decisions analyze the availability results to identify if it is good enough for the system's responsibility. If it is satisfactory, we can results presentation and finish the methodology. If it is not, we proceed to the next box or return to the building evolution model task.

4.1 Preliminary study

The preliminary study covers the first six macro-tasks of the supporting methodology: (1) studying the system; (2) monitoring the system; (3) calculating the parameters; (4) building models, (5) validating models, and (6) Analyzing mode.

- *Studying the system:* This step consists of understanding the system's structure and the leading hardware and software components. The result of this step is a components list that needs to be monitored, making possible a most efficient system's monitoring;
- *Monitoring the system:* Configure the monitor tool to identify malfunctions of the main components identified in the previous step. We also determined a deadline for data collection and wait for the desired time to obtain the most realistic scenario possible. The results are the records of the failures and repairs of the main components.
- *Calculating the parameters:* We utilized the records of the previous task to calculate the MTTR and MTTF parameters of each component. Thus, we did not obtain this information from the literature, permitting more realistic data for our environment. This data is utilized to step of build models.
- *Building models:* We build hierarchical models representing the initial environment in this step, know as the baseline. We use the results of the calculations obtained in the previous step as input for the RBD models representing the consolidation of components operating in series. With a result of RBD models, we obtained the MTTF and MTTR values' of components. Finally, with these values, we created the availability model in SPN.
- *Validating models:* The statistic will be applied to compare the experimental results with the model results. Case the result is satisfactory, we proceed to the next box. Case it is not, we return to the previous step to edit models.
- *Analyzing model:* Presentation of preliminary results. In this step, we present the results obtained to those responsible for the system. This step is essential for identifying the current availability of the environment and validating whether improvements are needed. If the values are satisfactory for them, we can finish the workflow. If not, we can proceed to the evolutionary study.

4.2 Evolutionary study

The evolutionary study has four macro-tasks: (6) sensitivity analysis; (7) building evolution model; (8) result analysis; and (9) result presentation.

- *Sensitivity analysis:* Conduct sensitivity analysis to identify components that can be adjusted. The result of this step is a list of the parameter has the most significant interference on availability.
- *Building evolution model:* In this step, we analyze the results obtained in the sensitivity analysis, identifying the components that can be applied to redundancies

and adjust the previous model to improve the availability of the environment. The result is a new availability analytical model in SPN with suggesting infrastructure.

- *Result analysis:* We performed the analysis of the availability metric generated by the model in the previous step. As a result, we identified improvements in the availability. In case it was significant, we validated the results with those responsible for the system. If the results are satisfactory, we proceed to the next step. If not, we go back to the previous step to make further adjustments to the redundancies of the components. This process repeats until we obtain an acceptable value for the availability of the environment.
- *Result presentation:* This step is characterized by data representation through graphs and tables. The results will include the proposed models, the assessment of the availability of these models, and an assessment of sensitivity analysis.

5 The system architecture

This paper uses an academic system of a Brazilian university to analyze and present proposals for infrastructure changes, aiming to improve its availability. The system is hosted in a DC inside of the university, using a private cloud system. The system has some peculiar characteristics, but this study can be applied to any application in a similar environment. Our approach analyzes IT infrastructure. We do not cover other segmentations of a DC in our discussions, such as energy and cooling, even though we know their importance for the complex IT system's availability. Instead, our focus is to identify the sensitive points of the environment's logical and physical infrastructure and perform modeling, proposing a better structure for the application.

Even though we know the advantages of modular software architecture, we won't cover software development in our study. We focus only on the layout and quantities of virtual tiers and physical infrastructure resources. We segment the environment into three large groups for better understanding and exposes the models separately. They are Computational Structure, explained in the Sect. 5.1, Logical Structure, explained in the Sect. 5.2, and Data Center structure, explained in the Sect. 5.3.

5.1 Computational structure

The computational structure represents the physical environment, and it is presented in Fig. 2. It consists of three tiers: **connectivity, virtualization, and storage**. The connectivity tier consists of two ethernet core switches (NET) operating in high availability, active-active mode. There is also a cluster of physical servers (SRV), working in high availability. The SRV's cluster is part of the virtualization tier. The last physical tier in our study is the storage tier. It consists of a Fiber Channel Switch cluster (SAN), working in high availability with two active-active components and the storage (STG). The internal components of the storage already have several redundancy features (physical and logical).

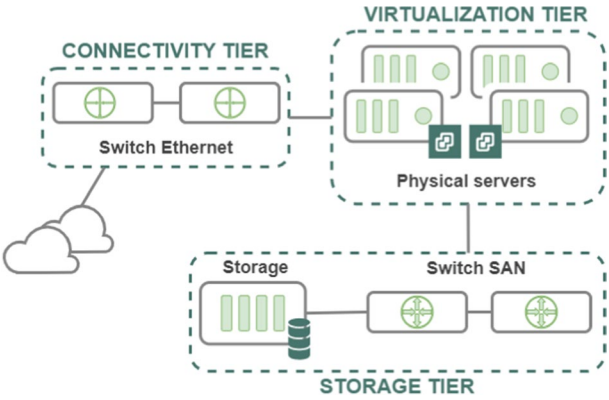

Fig. 2 Computational structure

The virtualization tier comprises the SRV and its operation system (OS). In this study, we use a virtualization operating system owned by VMware[2], knows as ESXi. We use three different physical server configurations: small, midsize, and large servers. All servers have the same processing capacity. The difference between them is the amount of RAM. The small server has 12 GB of RAM, the midsize server has 24 GB of RAM, and the large server has 32 GB of RAM.

5.2 Logical Structure

The logical structure represents the software responsible for keeping the system in an operational state. In this study, we know it as a software tier. It consists of four tiers: **Orchestration (VC), Load balancer (LB), Application (APP), and Database (DB)**.

The software tiers are virtual machines that are hosted inside of the virtualization tier. Each software tier has an OS (CentOS Linux Operating System) and the application responsible for a specific service, as shown in Fig. 3.

Orchestration tier - This tier is not directly part of the system but is very important to maintain the system's high availability. The software used is knowns as VCenter[3]. It is responsible for managing all physical resources of the server cluster, the environment's performance, and virtual machines' availability, among other functions. The VCenter uses a resource known as *High Availability (HA)* that allows the identification of malfunctions of the SRV and runs the migration of hosted virtual machines to another SRV automatically. In this process, VCenter identifies which SRV has the most resource availability and then performs the virtual machine's initialization on the chosen server.

[2] www.vmware.com

[3] https://www.vmware.com/br/products/vcenter-server.html

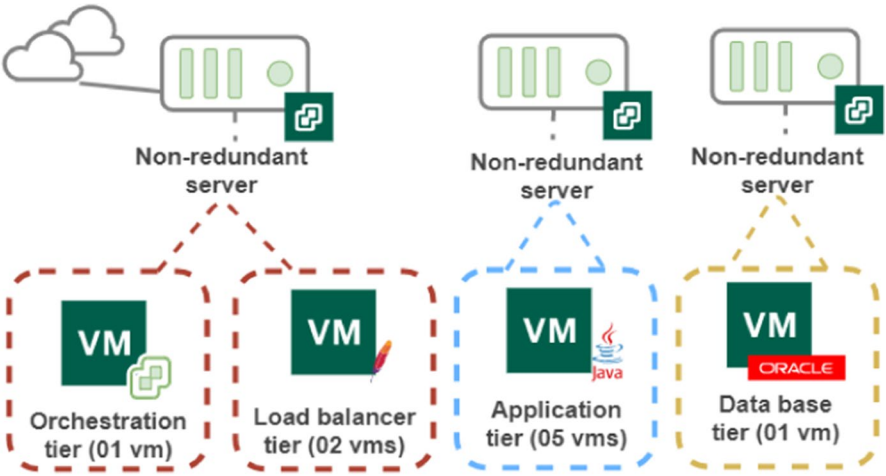

Fig. 3 Baseline architecture

Even with the ability to identify which server has the better physical resources, the HA functionality is a reactive action, as it is only performed after the SRV's failure. So, the virtual machine also suffers an interruption in the provision of services. VCenter has several other features, but we do not use them in this study. This skill set means that VCenter is classified as a virtual environment orchestrator. The Orchestration tier is hosted on a small server configuration, and the amount of RAM is 6GB. In Fig. 3, it corresponds to the red dashed box.

Load balancer tier - This tier is responsible for the application's interaction with the client. It receives requests from customers and forwards them to the application tier. One of its functions is identifying which component of the application tier has the most significant availability of resources and redirect requests to it. This resource is known as load balancing. The load balancer used in this solution is a free software known as Apache. [4]In Fig. 3, it corresponds to the red dashed box that is hosted on a small server configuration. The amount of RAM is 2GB.

Application tier - The tier is the application itself. Responsible for all business logic and information processing. The language used is a Java,[5] the free version owned by Oracle. The application is developed in a monolithic way, and as already said, we do not propose changes in the application's construction structure. We used the free container platform knowns as Apache Tomcat.[6] In Fig. 3, it corresponds to the blue dashed box that is hosted on a middle server configuration, and the amount of RAM is 4GB.

[4] https://www.apache.org.
[5] https://www.oracle.com/br/java.
[6] https://tomcat.apache.org.

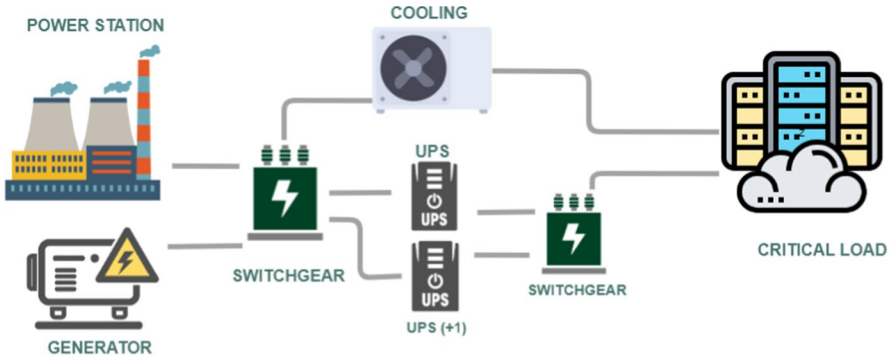

Fig. 4 Data center structure (DCS)

Database tier -This is the last virtual tier and is responsible for data storage. It is composed of proprietary software knowns as Oracle Enterprise by Oracle.[7] In Fig. 3, it corresponds to the yellow dashed box that is hosted on an extensive server configuration, and the amount of RAM is 30GB.

The software's tier data (VMs) are stored in the storage tier and accessible by all physical servers. In failure cases of the physical server, the virtual machines could be started (manually or automatically) in any physical server. We decided to use the physical servers' total capacity, always hosting the maximum number of supported virtual machines. We do not put together VMs of different tiers in the same physical server, except for the Orchestration tier and Load balancer tier. Thus, a small server configuration can host a maximum of one orchestrator and two load balancers. A midsize server configuration can host a maximum of five applications, and an extensive server configuration can host a maximum of one database.

5.3 Data center structure

The data center structure (DCS) represents the components responsible for cooling and energy distribution. The DCS is represented by Fig. 4. Two UPS are operating in high availability, an electric generator, and eight cooling devices, also operating at high availability. In circumstances of power supply failure by the local operator, the energy autonomy of the environment is equivalent to seven uninterrupted hours. They are the six hours of the generator (without supply) and an hour of the no-break. The autonomy of the UPS is two hours each, totaling four hours. However, suppose the DC is without electrical supply for more than an hour (from the generator or the local power station). In that case, the environment needs to be turned off to avoid damage to the equipment due to overheating. It occurs because all energy supply is carried out by the UPS, not including the refrigeration system.

[7] https://www.oracle.com

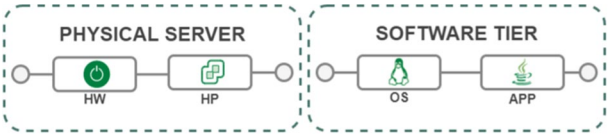

Fig. 5 RBD model representing physical server and software tier

6 Proposed availability model

In this section, we present the availability evaluation models that represent the studied environment. We used the Mercury tool[8] [35] to perform models, sensitivity analysis, and availability measures.

We propose a hierarchical availability modeling using RBD and SPN. This approach is beneficial for analyzing redundant cloud systems. In this way, we can combine strategic components of the system. Furthermore, the SPN allows us to include temporized actions for the proposed model, component simplification, represent high availability actions, like live migration and reactive migrations of virtual machines. Thus, we were able to propose high-level models more faithful to the studied scenario.

6.1 RBD models

We use the RBD models to synthesize some components as SRV and Software tier, Fig. 5. We utilized the serial RBD because they have no redundancy. This strategy allows extraction of the MTTF and MTTR values of them, utilizing Equation 5. Then, we use this data as input for the SPN models, and thus, we are able to calculate the system's availability.

We verify that all physical components (motherboard, memory, controllers, and other hardware components) are combined in the same box, represented by (HW). After that, put this box in line with the operating system installed on the server, identified by (HP). The model represents the physical server or virtualization tier. This approach facilitates the modeling of the software tiers, which also is exposed in Fig. 5. The box (OS) represents the Linux operation system installed in the virtual machine. The software responsible by service is represented by (APP). This software is different for each software tier. The logical representation of the physical server is $(HW)\,AND\,(HP)$, and of the software tier it is $(OS)\,AND\,(APP)$.

6.2 SPN models – computational structure

This time we present the SPN models for the following tiers: *connectivity, storage, and virtualization*. We created the last tier with data extracted by the RBD model (SRV). The representation of the physical component's models (computational

[8] http://www.modcs.org/.

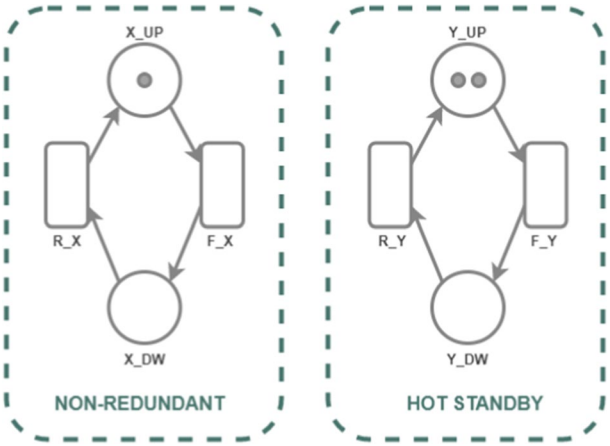

Fig. 6 SPN model representing connectivity, storage and virtualization tiers

structure) in SPN follows three different configurations: *non-redundant, hot standby, and cold standby*. The non-redundant and hot standby configurations are exposed in Fig. 6.

The non-redundant configuration represents tiers that do not have components with a mechanism for high availability. Case this component has been in a failure state, the system turns inaccessible. The hot standby configuration represents tiers that have high availability mechanisms. In these configurations, the tier has more than one component being executed simultaneously in active-active mode. The cold standby is also a mechanism for high availability, although the components perform in active-passive. Just one element does turn on at a time.

The non-redundant model represents the storage and virtualization tiers (operating with just one SRV). It is exposed on Fig. 6 in group classified by non-redundant. However, we adopted the label (X) as a generic name. Where it was exposed, represents STG or SRV. The hot standby represents the connectivity and virtualization tiers (with more than one SRV). It is exposed on Fig. 6 in the group classified by hot standby. The label (Y) also is a generic name, can be replaced by SAN or SRV.

The places X_UP and Y_UP represent the tiers in the operational state. X_DW and Y_DW symbolize tiers in the failed state. The number of tokens represents the quantity of the components in the tier. The model representing the non-redundant tier has a token that represents one component. In contrast to hot standby, which has more than one token that represents the number of tier components. The token's locations signalize the component status.

In non-redundant models, the tier is operational when X_UP has a token and, at failure, if a token is in X_DW. In hot standby models, the tier is operational when Y_UP has at least one token and is a failure state when all tokens are in Y_DW.

Transitions represent the model's actions. When these actions have a fixed time, they are known as timed transactions. The transitions F_X and F_Y represent the MTTFs, and the transitions R_X and R_Y represent the MTTRs of the tiers. In non-redundant models, the timed transitions F_X and R_X are configured

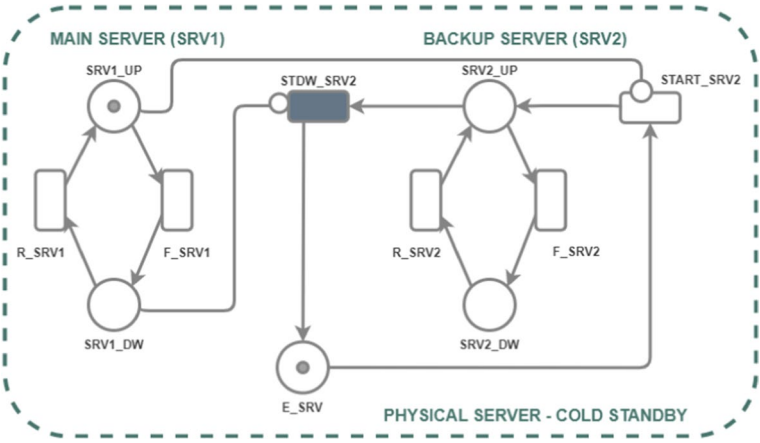

Fig. 7 SPN model representing cold standby server

with server semantics *Single Server*. However, in hot standby models, the F_Y and R_Y are configured with server semantics *Infinite Server* to represent parallel operation (for failure and repair).

The arrow is known as an arc transition. It is responsible for connecting places using transitions. It consumes and creates tokens depending on how it is connected.

In the *non-redundant model*, when there is a token in place X_UP, the only active transition is F_X because it is connected to the only possible location for token removal. When the timed transition F_X is triggered, the token is consumed by the arc transition and generated in place X_DW, identifying the system in the failed state. As of this moment, the only active transition is R_X. When fired, the token is consumed by the arc transition and generated in place of X_UP.

In the initial state of the *hot standby model*, the place Y_UP has all the tokens, making F_Y the only active transition. When triggered, one token is consumed by the arc transition and generated in place Y_DW. At this time, Y_UP and Y_DW have tokens, signaling that there are components on failure and operational states. Then the two timed transitions are active. If the R_Y transition is triggered, the system returns to the initial state with all tokens in the place Y_UP, but if the F_Y is triggered, it will generate one more token place Y_DW. If all tokens are in place Y_DW, it signals that the tier is in a failure state.

Figure 7 shows the availability model for the virtualization tier operating in *Cold Standby*. We identify the main server as (SRV1) and the backup server as (SRV2). This modeling requires strategies to manage and identify which server is in an operational state. Thus, we created a place presented as (E_SRV). When there has a token, the main server is operational, and the backup server is down.

It is also necessary to control the startup and shutdown of the backup server, as it can only be in a functional state if the principal server is in a failed state. These actions are performed by the timed transition START_SRV2 and by the immediate transition STDW_SRV2, respectively.

As there is only one primary server and one backup server, timed transitions are configured with server semantics *Single Server*. In addition, the presented model includes inhibiting arcs and guard expressions to restrict the triggering moments of the transitions. The immediate transition STDW_SRV2 suffers actions from the inhibiting arc, the arc transition, and the guard expression. The Inhibiting Arc does not allow it to be fired if there are tokens in the place SRV1_DW. The arc transition only allows it to be triggered if there was a token in the place SRV2_UP. The guard expression $(\#STF\_UP = 0)\, AND\, (\#SFT\_DW = 0)$ restricts the trigger if the number of tokens in the places STF_UP and STF_DW are equal to 0. Thus, the transition will only be triggered if all restrictions are met. Places with STF labels represent the software tiers presented in the Sect. 5.2. The timed transition START_SRV2 is also influenced by the inhibiting arc and cannot be triggered if there are tokens in place SRV1_UP.

In the initial state of the cold standby model, the primary server is operational, and the backup server is shut down, so the places SRV1_UP and E_SRV have one token each. The only transition active is F_SRV1. When triggered, a token is consumed by the arc transaction and generated in place SRV1_DW. At this time, the transitions R_SRV1 and START_SRV2 are actives. If the R_SRV1 is triggered, the system returns to the initial state.

Nevertheless, if START_SRV2 is triggered, the token from E_SRV is consumed and generated in place SRV2_UP. Currently, the backup server is in an operational state, and the primary server is in a failure state. From now on, an external process is beginning. The orchestration tier migrates the virtual machines to the backup server. The active transitions are the F_SRV2 and R_SRV1. If the F_SRV2 is triggered, the token is consumed and generated on SRV2_DW. So, all servers are down, and the virtualization tier is in a failure state.

However, if R_SRV1 is triggered, the place SRV1_UP receives a token informing that all servers are operational. At this time, the immediate transition STDW_SRV2 and the timed transition F_SRV2 are active. If the transition STDW_SRV2 is triggered, it generates a token on E_SRV, backing to the initial state of the tier. However, if F_SRV2 is triggered, the backup server is in a failure state, and all virtual machines are migrated to the central server by the orchestration tier.

6.3 SPN models – logical structure

In this subsection, we present the SPN models for software tiers: *orchestration, load balancer, application, and database*. To create these models, we extracted data from the RBD model, exposed in Fig. 5. The software tier is virtual; therefore, it suffers external interference from the virtualization and storage tiers. If any of these tiers are in the fault state, the software tier is directly affected and is forced to shut down. Because of this, it is classified as sensitive to external changes.

The representation of the software tiers' models in SPN follows three different configurations: *non-redundant, hot standby, and cold standby*. Similar to the computational structure models, the non-redundant configuration represents tiers that do not have high availability.

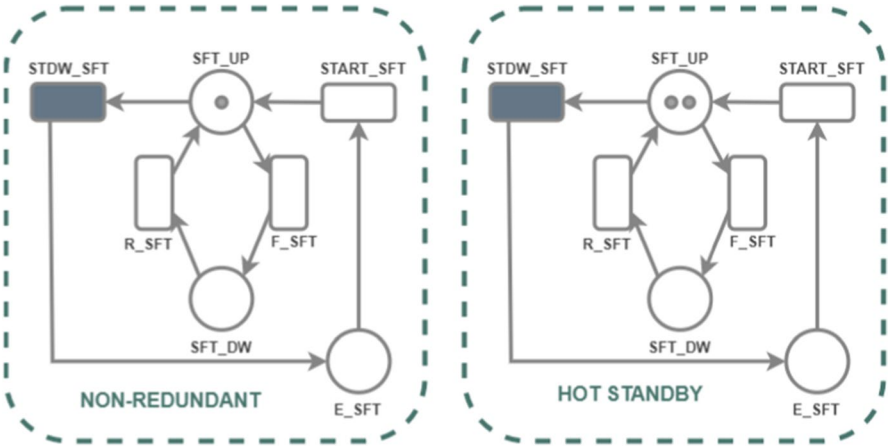

Fig. 8 SPN model representing software tier

Table 1 Guard expression of Software tier

Transition	Guard expression
R_SFT	(#SAN_UP > 0) and (#STG_UP > 0) and (#SRV_UP > 0)
STDW_SFT	(#STG_UP = 0) or (#STG_UP = 0) or (#SRV_UP = 0)
START_SFT	(#SAN_UP > 0) and (#STG_UP > 0) and (#SRV_UP > 0)

On the other hand, the hot standby configuration represents tiers with high availability mechanisms in active-active mode. The cold standby configuration represents tiers that are hosted in the physical server with a cold standby configuration. Figure 8 represents them. This figure is adopted in the label (SFT) to represent a generic name. In the place that appears, you can read (VC, LB, APP, or DB) representing labels used to orchestration, load balancer, application e database tiers, respectively. The non-redundant configuration represents the orchestration and the database tiers when operating with one component. The hot standby configuration represents the load balancer and application tiers.

The places SFT_UP and SFT_DW represent the tier in the operational and failure status, respectively. The timed transitions F_SFT and R_SFT represent the MTTFs and MTTRs of the tiers. For non-redundant models, these transitions are configurated with server semantics *Single Server* and for hot standby models, is *Infinite Server*. The number of tokens represents the number of components.

Because these tiers are sensitive to external changes, we create a control place (E_SFT) and two transitions to determine the entry and exit of this state (STDW_SFT and START_SFT). The transition STDW_SFT is of type immediate, and the START_SFT is of type timed. These transitions suffer actions from the arc

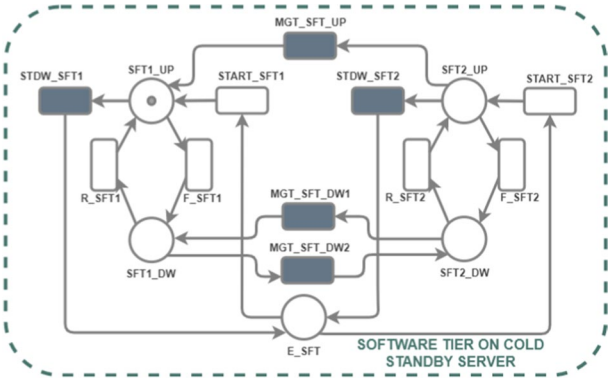

Fig. 9 SPN model representing virtual tier hosted in cluster cold standby

transition and the guard expression. Another transition that has guard expression is R_SFT. It is necessary because the tier only can back to the operational state if the physical components are functional. Information about guard expressions is presented in Table 1. The guard expressions have references to the physical server and storage tier because there is a direct dependence on these resources.

The operation of both models (non-redundant and hot standby) is similar to computational structure models. On the initial state, the only place that has tokens is SFT_UP. The only transition active is F_SFT. When triggered, a token is consumed from SFT_UP and generated a token in SFT_DW. In the non-redundant model, the tier is in a failure state, and the only transition active is the R_SFT (case the guardian expression is met). If the transition is triggered, the token is consumed e generated on SFT_UP restoring to the initial state. SFT_UP and SFT_DW have tokens in the hot standby models, signaling that have components on failure and operational states. Then the two timed transitions are active (case the guardian expression is met for R_SFT transition). If the R_SFT is triggered, the system returns to the initial state (all tokens in SFT_UP). But if F_SFT is triggered, a token is generated in SFT_DW. The tier is in a failed state.

At any time, if the guardian expression for immediate transition STDW_SFT is met, the transition will be immediately triggered (case the place SFT_UP has a token). So, the token is consumed and generated in E_SFT. This time, the tier is in a failure state but provoked by external changes. From now on, the only transition that can trigger is START_SFT if external interferences have been remedied (guard expression). If triggered, a token is generated in SFT_UP, and the system returns to a functional state.

The last configuration that represents software tier models is the cold standby. This configuration needs to be hosted in a server that has a cold standby configuration. The model allows identifying which physical server the virtual machine is hosted on. Figure 9 shows the availability model for this configuration. The places SFT1_UP and SFT1_DW represent the service hosted on the main server. The places SFT2_UP and SFT2_DW represent the service hosted on the backup server. The Basic features are similar to the other models already exposed. SFT1_UP and

Table 2 Guard expression of software tier on cold standby

Transition	Guard expression
STDW_SFT1	(#SAN_UP=0) or (#STG_UP=0) or (#SRV1_UP=0)
STDW_SFT2	(#SAN_UP=0) or (#STG_UP=0) or (#SRV2_UP=0)
START_SFT1/ R_SFT1	(#SAN_UP>0) and (#STG_UP> 0) and (#SRV1_UP>0)
START_SFT2/ R_SFT2	(#SAN_UP>0) and (#STG_UP> 0) and (#SRV2_UP>0)
MGT_SFTC	(#SRV1_UP > 0)
MGT_VC_SFT1	(#SRV1_UP > 0)
MGT_VC_SFT2	(#SRV1_UP=0) and (#SRV2_UP > 0)

SFT2_UP represent tiers in the operational and SFT1_DW and SFT2_DW in the failure. The timed transitions F_SFT1 and F_SFT2 represent the MTTFs, and the timed transitions R_SFT1 and R_SFT2 represent the MTTRs of the tier. It is possible to two different configurations for these timed transitions. Case the tier does operate in a non-redundant mode, the transitions are configurated with server semantics *Single Server*. Case the tier does operate in a hot standby mode, the transitions are configurated with server semantics *Infinite Server*. The number of tokens represents the number of components.

The control place E_SFT represents the external interferences suffered by the tier. The immediate transitions STDW_SFT1 and STDW_SFT2 represent the shutdown of the tier when suffering external interferences. The timed transitions START_SFT1 and START_SFT2 represent the startup of the tier after external interferences. These transitions suffer actions from the arc transition and the guard expression. Other transitions with guard expression are R_SFT1, R_SFT1, MGT_ SFT_UP, MGT_SFT_DW1, and MGT_SFT_DW2 because the tier only can back to the operational state if the physical components are operational. Information about guard expressions is presented in Table 2. The guard expressions have references to the physical server and storage tier because there is a direct dependence on these resources.

The model also represents live and reactive virtual machine migrations through MGT_SFT_UP, MGT_ SFT_DW1, and MGT_SFT_DW2 immediate transitions. These actions are the responsibility of the orchestration tier. The transition MGT_ SFT_UP performs the live migration (without loss of connectivity) of the software tier hosted on the backup server to the main server, case the primary server is operational. The transitions MGT_SFT_DW1 and MGT_SFT_DW2 represent the reactive migrations. In these actions, the tier's component is in a failed state. These actions are necessary to ensure that the service does not attempt to use physical resources from an inoperable server upon returning to the operational state.

This model always has two possible paths: the internal path and the external interference path. We will distinguish this way in explaining the model's functionality.

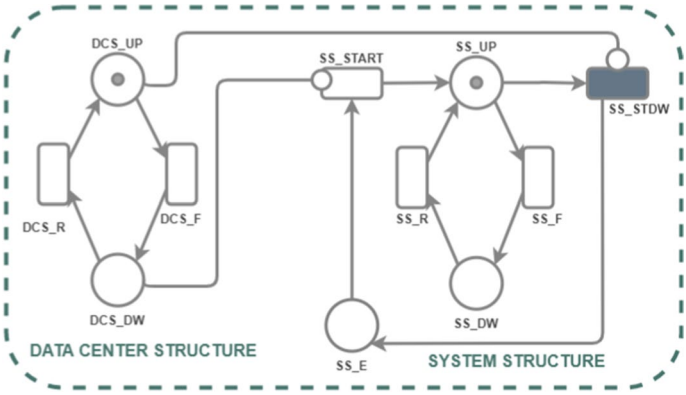

Fig. 10 SPN model representing Data Center and System structure

Fails of the virtualization or storage tiers can trigger the external interference path at any time.

We assume that the primary server is operational, and the virtual tier is hosted on it representing the initial state. So, the place SFT1_UP has a token, and the only active transition is F_SFT1. If triggered, a token is generated at SFT1_DW. At this point, the only possible transition is R_SFT1 which, when fired, a token is generated at SFT1_UP, returning to the initial state of the model.

Now, we explain the possible paths for external interference. For example, if there is a token in place SFT1_UP, the immediate STDW_SFT1 transition can be triggered and generate a token in place E_SFT. At this time, the component is in a state of failure due to external actions. When external interferences are remedied, the transitions START_SFT1 or START_SFT2 are triggered. The START_SFT1 is triggered if the central server is operational, returning to the model's initial state. And the START_SFT2 is fired if the backup server is functional, generating a token SFT2_UP, signaling that the backup server hosts the component. At this point, The transitions F_SFT2 and R_SFT2 can execute the internal path representing failure and restoration of the components, respectively. The immediate transition STDW_SFT2 can also perform the external interference path, case the backup server or storage fails, returning the token to the special place E_SFT.

6.4 SPN models – data center structure

At this time, We propose the analytical models representing the availability metrics of DCS and system structure (SS). The DCS was described in Sect. 5.3. The SS is a synthesis of computational and logical structures. This synthesis allows the construction of high-level models in a more simplified form.

Figure 10, is a SPN model representing DCS and SS. We create the place (SS_E) to identify the moment when the SS is in a state of failure caused by DCS failure. This strategy is necessary to control the startup and shutdown of the SS, as it can only be in a functional state if the DCS is in an operational state. These actions are

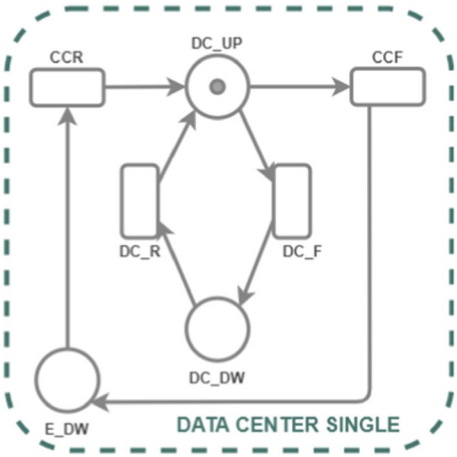

Fig. 11 SPN model representing Data Center single

performed by the timed transition SS_START and by the immediate transition SS_STDW, respectively. The timed transitions have the server semantics *Single Server*. In addition, the presented model includes inhibiting arcs restricting the trigger.

The transitions SS_STDW and SS_START suffer the actions of the inhibiting and transition arcs. Inhibit does not allow firing as long as there are tokens in the location that is connected. On the other hand, arc transition only allows firing when there are tokens in the place that is connected. Thus, the trigger is only triggered when all constraints are met.

In the initial state, the places DCS_UP and SS_UP have one token. The active transitions are DCS_F and SS_F. If SS_F is trigged, a token is consumed and generated in SS_DW. At this time, the SS_R and DCS_F are actives. If the SS_R is triggered, the system returns to the initial state. Nevertheless, if DCS_F is fired, DCS_DW receives a token. Currently, the DCS and SS are in a failure state. From now on, the only active transition is DCS_R. Even with a token in SS_DW, the transition SS_R cannot be triggered because it has a guardian expression ($\#DCS\_UP > 0$) that only permits the trigger if DCS is in the operational state. When DCS_R is triggered, the DCS comes back to a functional state. The transition SS_R can be fired coming back to the model's initial state.

The second model used to represent the DCS is shown in Fig. 11. We simplified the previous model (Fig. 10) and added the ability to interact with external faults to the DC. In our study, this fault was classified as *Common Cause Failure (CCF)*. The CCF can be an interruption of external access, power outages, natural disasters, among other problems that interrupt external access to the environment. In that model, we consolidate the DCS and SS groups. From this moment on is called DC. Thus, when we say that the DC is in an operational state is the same that DCS and SS are functional.

The places E_DW and DC_DW represent the DC in the failed state. DC_DW means the DC failed due to internal problems, while E_DW is due to external faults. The transitions are configured with server semantics *Single Server*. A token is generated in E_DW whenever a CCF occurs, indicating that the DC is

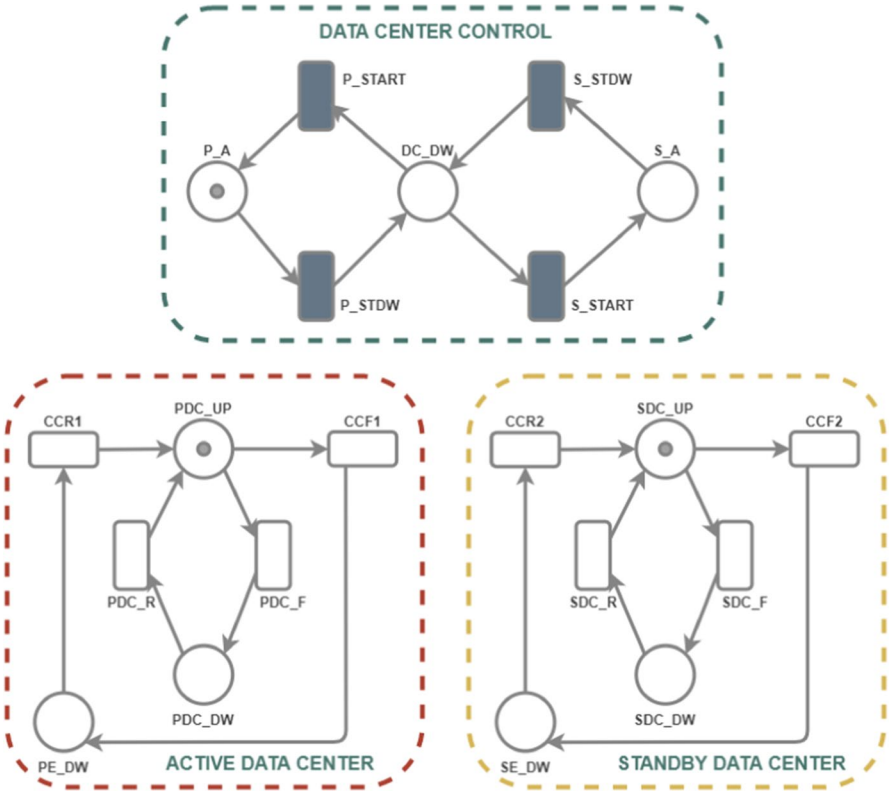

Fig. 12 SPN model representing Data Center redundanty

unavailable due to an external problem. In the initial state, the only token is in DC_UP. The transitions active are CCF and DC_F. When triggered, a token is consumed from DC_UP and generated in E_DW or DC_DW. From now on, the active transitions are the CCR or DC_R. When fired, DC_UP receives a token, returning to the initial state.

The last model used to represent the DCS is shown in Fig. 12. In this model, the composition that designates the DC was duplicate. And we appended a new structure for an active DC control. The Principal DC (PDC) corresponds to the red dashed box. The Secondary DC (SDC) corresponds to the yellow dashed box. The active DC control is the green dashed box. The active DC receives the access traffic. The DCs are independent and can be operational at the same time, yet, only the actives DC receives access requests.

When there is a token in P_A, it signals that the PDC is active. If the token is in S_A, it represents that the SDC is active. The DC_DW represents that has DC on failure state. The P_START and S_START immediate transitions are responsible for redirecting access traffic to the active DC, already the transitions P_STDW and S_STDW stop the access traffic of active DC. To ensure proper functioning

has guard expressions in the transitions. The P_STDW's guard expression is ($\#PDC\_UP < \#P\_A$). The S_STDW's guardian expression is ($\#SDC\_UP = 0$). Both ensure that transactions are triggered only when a DC active is no longer operational. The guardian expression of P_START and S_START are, respectively ($\#PDC\_UP > \#P\_A$) and (($\#SDC\_UP = 1$) AND ($\#S\_A = 0$)).

In the initial state, there are tokens in P_A, PDC_UP, and SDC_UP. The fault transitions active are CCF1, PDC_F, CCF2, and SDC_F. When triggered, a token is added to the failures' state place PE_DW, PDC_DW, SE_DW, or SDC_DW. If the CCF1 is fired, a CCF occurred in the PDC, making it unavailable. The immediate transition P_STDW is triggered generating a token on DC_DW. After that, another transition is fired, S_START, making the SDC active and receiving the access traffic. Right now, there are tokens in places PE_DW, SDC_UP, and S_A, and the active transitions are CCR1, SDC_F, and CCF2. Case SDC_F or CCF2 are triggered, is generated a token at the SDC_DW or SE_DW. Regardless of the triggered transition, the SDC is in a failed state and the system is completely inaccessible. However, if the CCR1 is fired, generates a token in PDC_UP. Indicating that both DCs are operational, but the SDC is still active. This way, if there are problems in the SDC, the immediate transition S_STDW and after P_START is triggered, making the PDC active and accessible externally.

7 Case studies

In this section, we carried out four case studies: baseline infrastructure, System Availability Evaluation, Data Center Availability Evaluation and Data Center Redundancy Availability. The baseline infrastructure represents the studied system's current environment. We present the baseline infrastructure, the availability model in SPN, and sensitivity analysis. The System availability evaluation represents the proposed infrastructure implementing changes guided by the sensitivity study. We built a new model depicting the proposed environment and presenting the proposed model's availability values. In the Data Center availability evaluation, we introduced the DC structure in the previous model and generated the environment availability. Afterward, in Data Center redundancy availability, we implement a new model applying for the DC redundancy and present the proposed model's availability values.

In order to approximate the results of the model with a real system, we performed an experiment monitoring the system for six months using Zabbix[9]. Each component of the virtual tier was monitored individually using specific techniques that varied depending on the characteristics of each one. We perform network connectivity monitoring for OS. HTTP and HTTPS protocols' accessibility for load balancer and application. And specific monitoring functions for the database. Data from physical servers, SAN switches, Ethernet switches, and storage were combined between the literature (MTTF) and studied environment (MTTR). This distinction was necessary

[9] https://www.zabbix.com/.

Table 3 Input values to RBD model

Component	MTTF(h)	MTTR(h)
Switch Ethernet	87600	24
Switch SAN	87600	48
Disk Array	131400	4
Pysical server	26280	24
Hypervisor (HP)	8760	1.67
Orchestration Service	25535	8
Ballancer VM	98.11	0.12
Application VM	338.54	0.13
Database VM	99.84	0.05
Ballancer service	1033.46	0.43
Application service	34.71	0.13
Database service	271.43	0.91

due to a maintenance contract directly with the manufacturer for some equipment. The contract has a service level agreement (SLA) for a 48-hour defective parts exchange for the switches SAN and the storage and 24-hour for the switches ethernet and physical server.

In the monitoring used in this study, we identified only the moments of failures and repairs of each component. Whenever there was a change in the component's state, we collect time and date for analysis. In this way, we were able to identify and isolate each component's failure and repair times, allowing the identification of MTTF and MTTR. The values are shown in Table 3.

7.1 Case study I – baseline infrastructure

We created the availability model in SPN for the scenario and discovered the system's total availability metric. After that, we created a confidence interval to validate the model and perform a sensitivity analysis to identify which component is more feasible to improve the studied metric.

7.1.1 Infrastructure

The baseline scenario of the study was presented in Fig. 3 (see Sect. 3). It consists of connectivity, storage, virtualization, load balancer, orchestration, application, and database tiers. The storage component and the tiers virtualization, orchestration, and database have a non-redundant component. The SAN component, the connectivity tier, and the load balancer tier have two components each. The application tier has five components. All tiers that have mode then one component running on active-active mode.

The connectivity tier is directly responsible for the application's availability because when it presents failure, the service becomes inaccessible even though it is operational. The storage tier is also directly responsible, but differently. The

Table 4 Input values to SPN model

Component	MTTF(h)	MTTR(h)
Switch ethernet	87600	24
Switch SAN	87600	28
Storage	131400	4
Physical server	6502.26	7.19
Orchestration tier	25535	8
Load balancer tier	89.60	0.15
Application tier	31.48	0.13
Database tier	72.99	0.28

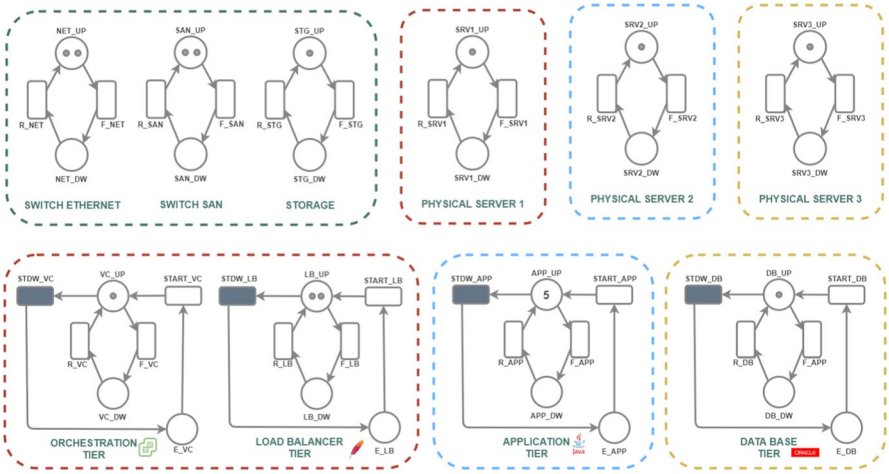

Fig. 13 SPN model representing case study I

complete logical structure is stored in the storage tier, and the physical servers are connected directly to it. Thus, in the case of a problem in the storage tier, the entire logical structure is inaccessible and result in a failure state for all virtual machines simultaneously. If the failure occurs when connecting a non-redundant physical server to the storage tier, the virtual machines hosted on that server become in a failed state. If the failure occurs with physical servers operating with redundancy mechanisms, it may or may not generate a failure state for all systems.

In this study, we illustrate the hosting of VM on physical servers by their outline color, informing that a VM usage the physical resources *memory, processing, network, etc.* of that server or group of servers. The VM cannot start on another group of physical servers. The virtual tiers are in an operational state if one or more servers in that group are in the same state. We always respect each physical server's hosting capacity. For example, in Fig. 3, the application software tier, made up of five virtual servers, uses the application server resource *with the same outline color*. If the application server has a failure state, all virtual machines have a failure state

Table 5 Availability's value of the Case study I

Metric	A	#9's	dt(h)
A_t	0.992698	2.1365	63.96

and cannot be started on another server or group of physical servers. The same idea occurs for the entire environment.

7.1.2 Availability model

We perceive that the virtual machine's availability and its operating system are directly linked to the type of virtual tier hosted. Operating systems that host the load balancer tier, for example, have different availabilities than the operating system that hosts the application tier. For this reason, we monitor and differentiate the data for each type of operating system. Therefore, we do not use in this study unique data for all operating systems. The values presented in Table 3 are individual for each component, and they were used as the input of the RBD model to synthesize each tier's values. The RBD models are previously exposed in Sect. 6. Table 4 shows the result obtained by the Mercury tool for the RBD model. We use these data as input of the SPN models.

The availability model in SPN is exposed in Fig. 13. To represent the components of physical servers, we used the structure shown in Fig. 6 *(non-redundant server)*. The representation of the orchestration and database tiers are shown in Fig. 8 *(Software tier non-redundant)*. The SAN switch and the tiers of the load balancer, application, connectivity are represented by group *(software tier hot standby)*, shown in Fig. 8.

For the studied system to be operational, we need to have at least one component of the following tiers in the functional state *connectivity, storage, virtualization, load balancer, application, and database*. The only tier that interferes with the others, but can be in the failure state, allowing the system to remain operating, is the orchestration tier.

The availability is the probability that these tiers are in the operational state at the same time [6]. Thus, the total availability is calculated as the probability of the following tiers are operational: connectivity AND storage AND virtualization AND load balancer AND application AND database. The expression for calculating a total availability in the Mercury tool is shown in

$$A_{(t)} = P\{((\#NET\_UP > 0)\ and\ (\#LB\_UP > 0) \\ and\ (\#APP\_UP > 0)\ and\ (\#DB\_UP > 0))\}.$$

We do not use all tiers because of the guard expressions of the components. They allow the load balancer, application, and database tiers to be in the operational state only if the storage and virtualization tiers are in the operational state. After performing a stationary analysis in the Mercury tool, we obtained the availability shown in Table 5.

Table 6 Model result validation

Metric	Real System	Model	CI 95%
A_t	0.9925	0.9927	$0.9888 < \theta < 0.9932$

To verify whether the calculated availabilities are realistic, we need to calculate the confidence interval *CI*. The monitoring identified only the failures and repairs of the components, collecting the time and date for analysis. The amount of information collected varied between components depending on your stability. More stable components presented few variations and consequently little data.

The application tier had the most variation, with 410 records, including virtual machine and application records. On the other hand, the most stable was the database tier, with 198 records. Thus, the information collected was sufficient to perform calculations and build the model but insufficient to generate a statistically acceptable population and generate a *CI*. For this, we chose the Bootstrapping technique to calculate *CI*. According to [12], Bootstrapping is a resampling mechanism capable of generating population statistics by sampling a data set.

First, we collected samples by the actual system monitoring. After that, we generated 1000 MTTF and MTTR samples for each component based on monitoring data. Then we calculate the availability of the bootstrap resampling. From this point, we can calculate the confidence interval. The twenty-fifth smallest and twenty-fifth largest of these 1000 bootstrap samples are the percentiles for the 95 % confidence interval. Now, we can identify whether our proposed model matches the actual system. We calculated the real system availability using the techniques presented in Sect. 2. The availability obtained by the model is exposed in Table 5 with metric A_t.

As we can see in Table 6, the confidence interval contains the availability metric of the proposed model. So, we cannot refute that the model does not represent the actual system.

The result of the total availability (A_t) presented in Table 5 shows the availability of the studied system with at least one component of each tier active. The total availability of baseline is $A_t = 0.992698$, with an annual downtime in hours of 63.94 h. This metric has a low value and a high annual downtime compared to other critical cloud-hosted systems. Aiming to improve this metric, we performed a sensitivity analysis to identify which component has the greatest sensitivity to change to improve system availability more accurately.

7.1.3 Sensitivity analysis

Sensitivity analysis identifies which parameter has the most significant interference on availability. First, we performed experiments individually, varying the values of each parameter. Then, we analyzed the impact on the metrics studied to identify them: the more significant influence, the greater the ability to interfere with availability. Thus, enabling more proactive adjustments to the environment.

This analysis and the experiments were performed using the Mercury tool, assuming a variation of 50 % for more and less than the values presented in

Table 7 Sensitivity analysis

Metric	Variation	Metric	Variation
MTTF_DB	0.004970	MTTF_LB	0.000017
MTTF_SRV	0.004459	MTTF_APP	0.000012
MTTR_DB	0.003849	MTTF_SAN	0.000008
MTTR_SRV	0.002971	MTTR_SAN	0.000007
MTTF_STG	0.000076	MTTR_LB	0.000006
MTTR_VC	0.000071	MTTF_NET	0.000000
MTTR_STG	0.000055	MTTR_NET	0.000000
MTTF_VC	0.000048	MTTR_APP	0.000000

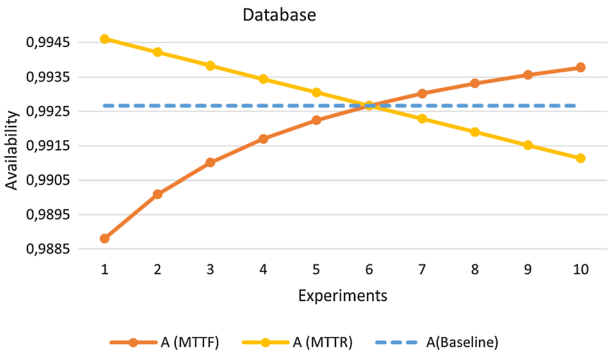

Fig. 14 Sensitivity analysis - database

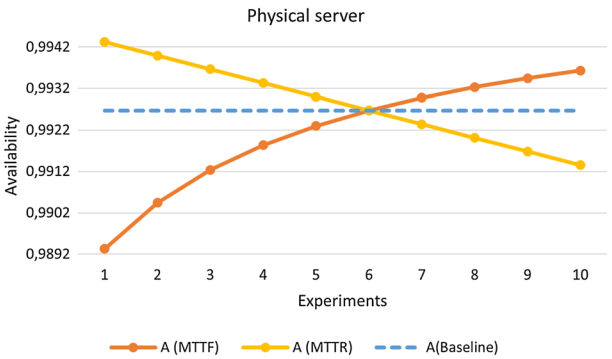

Fig. 15 Sensitivity analysis - physical server

Table 4. Within this minimum and maximum range, we configured a variation of the values of 10 %. Thus, it was possible to carry out ten experiments for each parameter. Having obtained the availability values for each experiment, we compared them with the value obtained in the presented model, identifying the

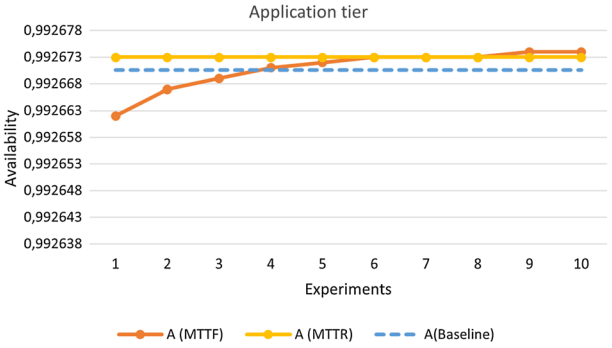

Fig. 16 Sensitivity analysis - application tier

variation of the final metric. Next, we check the highest and slightest variation of each parameter, as explained in Sect. 2. In this way, we were able to classify and identify which parameter has the most significant variation in the final metric. For example, values are shown in Table 7; the highest value means it has the most significant impact on final availability.

The studied environment has many parameters. Thus, we visually present the four that had the broad variations in the availability metric. As a comparison, we also demonstrate two parameters that obtained low levels of sensitivity. We detected that DB and SRV parameters had the highest rates, in contrast to APP. In the visual presentation, we combine the MTTF and MTTR of each component on the same graph, comparing against baseline availability for better understanding.

Figure 14 shows the DB parameters, and Fig. 15 of the SRV. The graphs show the experiment's number on the X-axis and the availability value on the Y-axis. Subtitle A(MTTF) represents the availabilities obtained in the experiments by changing the MTTF values. Subtitle A(MTTR) represents the availability values obtained when changing the component's MTTR.

As informed, we carry out experiments by changing parameter values. Thus, the experiments that used the smallest values are represented by the first point on the X-axis, followed by the second-smallest values on the second point. And so on until we reach the highest values of the experiments at point 10.

Utilizing Fig. 14 as an example, the lowest value used for the MTTF-_DB was 36.5. With this value, the availability obtained in the experiment was 0.988812. The second experiment used the MTTF_DB of 43.8, obtaining availability of 0.990095. Thus, the experiments continued until we reached the highest value for the MTTF_DB represented in point 10, which was 102.19 with the result of 0.993782. We perform that the higher the MTTF_DB value, the higher the availability value.

The same technique was used for MTTR_DB. The lowest value was 0.14 and had an availability of 0.994601. The second experiment used the value of 0.17 and obtained 0.994215 availability. The last experiment was with 0.4 of MTTR_DB, with the result of 0.991135. At this moment, we perceive a reverse direction. The higher the MTTR_DB value, the lower the availability value.

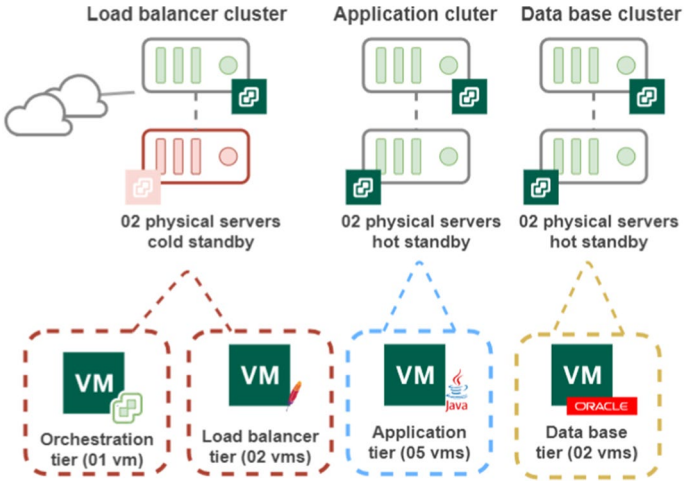

Fig. 17 The architecture of case study II

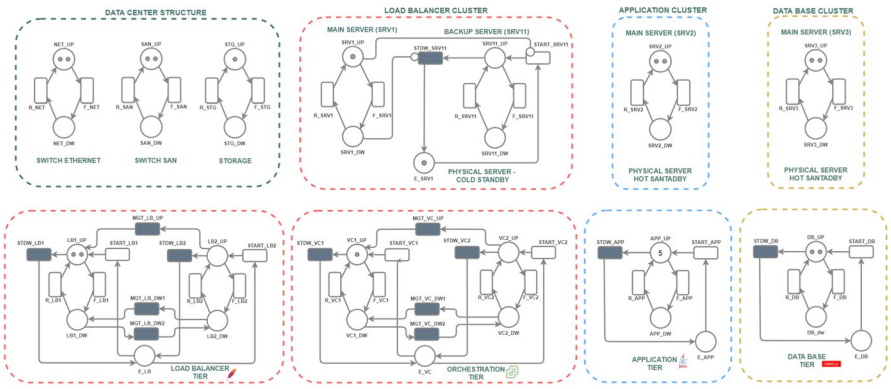

Fig. 18 SPN model representing case study II

The results obtained for the DB and SRV components are understandable. The higher the MTTF value, the longer the component uptime, resulting in better availability. On the other hand, the higher the MTTR value, the longer the failure time, resulting in lower availability. However, in our environment, not all components have such an expressive variation in availability to changes in parameters. The application tier, for example, had one of the lowest sensitivity rates. We can identify, in Fig. 16, that changing the MTTF_APP and MTTR_APP parameters had practically no effect on the availability value of the environment.

With this sensitivity analysis, we were able to identify which parameters could be improved to have a more assertive effect on the environment's availability value.

Table 8 Availability's value of the Case study II

Metric	A	#9's	dt(h)
A_t	0.999890	3.9590	0.96

Table 9 Input for the SPN model - DC Values

Component	MTTF(h)	MTTR(h)
Data Center Structure	2365.25	5.50
System Structure	183227.80	20.14
CCF	8760.00	24.00

7.2 Case study II – system availability evaluation

From this point of work onwards, the models are suggestions for improving availability. We do not have validations with the real data extracted by monitoring.

In this second case study, we propose changes in the database and virtualization tiers. Seeking to improve availability measures. Several approaches can decrease the average recovery time for a service, resulting in the longest service time in the operational state. In addition, we apply redundancy to the desired tiers. The proposed architecture of case study II is exposed in Fig. 17.

To improve the virtualization tier, we doubled the number of components. There are two possible configurations, cold standby, and hot standby. The physical servers responsible for hosting the orchestration and load balancer tiers operate in cold standby. Those responsible for the application tier run in hot standby. For the database tier, we propose to double the number of components working in hot standby. As previously studied, a non-redundant physical server does not have the physical resources to support two virtual database components. Therefore, to support this configuration, the cluster of physical servers works on hot standby.

The SPN model is presented in Fig. 18. When using the cold standby configuration for the physical server, exposed in Fig. 7, we also need to change the SPN model for the virtual tiers hosted on it. Therefore, we used the configuration shown in Fig. 9.

The availability expression is exposed in Expression

$$
\begin{aligned}
&P\{((\#NET\_UP > 0)\, and \\
&A_{(t)} = (((\#LB1\_UP) + (\#LB2\_UP)) > 0)\, and \\
&(\#APP\_UP \geq 1)\, and\, (\#DB\_UP \geq 1)\}.
\end{aligned}
$$

After stationary analysis, we achieve the availability of the environment. The results are in Table 8.

The value of availability metric for this study is $A_t = 0,999890$ with an annual downtime in hours of $0.96\, h$. These results are considered satisfactory for the application characteristics.

Table 10 Availability's value - Model simplification

Metric	A	#9's	dt(h)
A_t	0.997360	2.5783	23.13

Table 11 Availability's value - DC availability model

Metric	A	#9's	dt(h)
A_t	0.994604	2.2679	47.26

7.3 Case Study III - Data Center Availability Evaluation

After we were able to propose an availability model classified as acceptable for the system, we concatenate the DCS's values. The studied DC fits into the Tier II classification. In this case study, we simplified the previous model 7.2, added the CCF values, and finally created a model regarding DC redundancy. Data referring to DCS were obtained from the studied environment, through manual monitoring. We identify the moments of failure of each component and record the referred time. So we have your MTTF and MTTR. The values used in this case study are exposed in Table 9. In calculating the SS we use Eq. 2 to find the MTTF and MTTR.

7.3.1 Model simplification

In this study, we created the SPN's availability model that represents the DCS and SS. The goal is to discover the system's total availability. For the SS structure, it was necessary to simplify the model presented in the Sect. 7.2.

The availability model in SPN is exposed in Fig. 10 and its operation was showed in the Sect. 6.4. As already explained in sect. 6, there is a relationship between the systems, because, in case the DCS presents failures, the entire SS is also in the failure state. Consequently, the SS can only return to operational status if the DCS is also operational. In the model, the total availability is calculated as the probability of all groups are in the operational state: DCS AND SS. To calculate a total availability in the Mercury tool, we used the Expression

$$A_{(t)} = P\{(\#SS\_UP > 0)\}.$$

We do not use all groups in the expression because of the inhibitor arc. It allows the SS to be operational only if the DCS is functional. After performing a stationary analysis in the Mercury tool, we obtained the availability shown in Table 10.

The result of total availability (A_t) is presented in Table 10. It shows the availability of the system hosted in a private cloud considering the structure of the studied DC. Our next step is to add the CCF values.

7.3.2 DC availability model

In this case study, we created an availability model representing the DC, contemplating outages caused by external interference. Initially, we performed the simplification of the model presented in the Previous Case Study 7.3.1, also using Equation 2 to find the MTTF of the DC. This simplification generates a simple model that represents the DC. Enabling the implementation of CCF and redundancies. The values used as input to the model are $MTTR$: 25.636946821 and $MTTF$: 9683.900834, and the representation is shown in Fig. 11. The availability expression is exposed in Expression $A_{(t)} = P\{(\#DC\_UP > 0)\}$.

After stationary analysis, we achieve the availability of the environment. The results are in Table 11.

The value of availability metric for this study is $A_t = 0.994604$ with an annual downtime in hours of 47.26 h. We can see that availability has deteriorated compared to the previous model. This fact occurs because we have added the CCF values. The availability value is closer to the reality of the studied environment, yet, it has a low value and a high annual downtime compared to other critical cloud-hosted systems. Aiming to improve this metric, we performed another case study implementing DC replication.

7.4 Case Study IV – data center redundancy availability model

A DC's structure is directly proportional to the availability of hosted services. The studied DC does not fit perfectly into the descriptions of a DC tier II as it does not have two telecommunications companies nor two generators. However, it is very close in its features and availability. There are some possibilities to improve this metric: improving the DC's classification or deploy DC's redundancy. Each of the configurations has positive and negative points. In this study, we apply for DC redundancy because we understand that it is more beneficial in the long run. We can perform high availability, load balance, improve backup, and prevent higher outages due to disasters. So, we propose an availability model for DC's replication that permits identify the active DC.

The studied environment does not have a redundant DC. Thus, we replicate the data from the primary to the secondary DC. Another feature is that DCs operate separately from each other. We are aware of the interference in availability and the importance of some points such as data transfer, network throughput, and geographic studies. However, these points are not considered in this paper. We assume that data is constantly being replicated and there is no replication loss. We carry out comparative studies of availability in different DC's redundancy scenarios:

- Two DCs operating in Active/Standby;
- Two DCs operating in Active/Active;
- Three DCs operating in Active/Standby. Being two Active DCs and one Standby.

Table 12 Availability's value of the Case Study IV

DC	A	#9's	dt(h)
02 DCs (Active/Standby)	0.999971	4.5419	0.25
02 DCs (Active/Active)	0.999978	4.6645	0.19
03 DCs (02 Active/01 Standby)	0.999999	6.9360	0.0001

The availability model is exhibited in Fig. 12. There is a control unit to check the status of the DCs and redirecting external requests to the active DC. Places P_A and S_A identify the active DC. PDC receives active status if a token exists in place PDC_UP. On the other hand, the SDC receives the active status if it is operational and the PDC fails. A similar process takes place to return the PDC as active. It is necessary that the SDC is in the fault state and the PDC is operational. The number of tokens in place PDC_UP represents the number of active DCs. In our study, the maximum amount at this place is two tokens. However, we can quantify the number of active DCs we want. Distinct from the SDC that always has a token as represents a secondary DC (backup DC).

The availability expressions are different depending on the scenarios. In scenarios that have a standby DC, the expression is:

$$\begin{aligned} A_{(t)} = P\{&((\#P\_A > 0)\, AND\, (\#PDC\_UP > 0)) \\ &OR\, ((\#S\_A > 0)\, AND\, (\#SDC\_UP > 0))\}\,. \end{aligned} \tag{7}$$

In the scenario that all DCs are active, the expression is as follows:

$$A_{(t)} = P(\#PDC\_UP > 0)\,. \tag{8}$$

After stationary analysis, we achieve the availability of the environment. The results are in Table 12.

We can see that the availability difference using a single DC, Table 11, for using redundant DCs, Table 12 is high. However, the difference between using two Active/Active DCs for Active/Standby is small. When using three DCs, two active and one standby, the availability value increases dramatically.

8 Conclusions and future work

In this study, we proposed analytical models to assess the availability of a system hosted in a private cloud. We had two principal goals, identify and propose changes in the physical and logical infrastructure used by the system. And classify and recommend the best DC redundancy strategy.

We conducted four case studies showing how our models can be used to plan the system's environment. First, the built model represents the physical and virtual environment, contemplating that all virtual machines move between physical servers in

the same cluster. Second, we made a study to better take advantage of these physical servers' resources, always using each physical machine's maximum capacity. Third, we added the DC structure to the proposed model. Fourth, we propose an availability model for DC's replication that permits identify the active DC.

The first study, classified as the baseline, represents the actual environment that is currently used. We created the availability model, discovered the confidence interval using bootstrap, and conducted a sensitivity analysis to identify the components with the most significant tendency to change availability. The second case study was the proposal for changes in the environment. We discovered that two components have a high availability sensitivity index, *database tier* and *physical server* and applied the necessary redundancies to improve availability. The case studies point to a significant improvement in the metric of availability of the environment. The availability metric was increased from $A_{CaseI} = 0.992673$, with an annual downtime in hours of $63.94h$, to $A_{CaseII} = 0.999890$ with an annual downtime in hours of $0.96\ h$. The tiers that most affected the availability were the database and physical server.

The third study added DC's data to the model obtained in the previous case study. We applied simplification techniques until we achieved a simple model that represented the DC and the system. The fourth case study was the analysis of three DC redundancy scenarios. We created an availability model that enables the identification of the DC status (active and standby). The scenario recommended by us was not the one that obtained the best availability, but the one that possibly has the lowest cost-benefit ratio. The difference in availability when using two DCs acting in an active/active and active/standby way is tiny. Thus, we concluded that two DCs operating in an active/standby way would reach an acceptable availability, 0.999971, with an annual downtime in hours of $0.25\ h$, and would have a more accessible structure.

The approach presented in this paper can be applied to services hosted in a private cloud where the application has a similar infrastructure. As future work, we intend to implement DC replication and disaster recovery techniques and costs.

Acknowledgements We would like to thank the Coordination of Improvement of Higher Education Personnel – CAPES, National Council for Scientific and Technological Development – CNPq, Fundação de Amparo à Ciência e Tecnologia de Pernambuco – FACEPE, MoDCS and UNAME Research Groups for their support.

References

1. Ajmone Marsan M, Conte G, Balbo G (1984) A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. ACM Trans Comp Sys (TOCS) 2(2):93–122
2. Andrade E, Nogueira B, Matos R, Callou G, Maciel P (2017) Availability modeling and analysis of a disaster-recovery-as-a-service solution. Computing 99(10):929–954
3. Avizienis A, Laprie JC, Randell B, Landwehr C (2004) Basic concepts and taxonomy of dependable and secure computing. IEEE trans Depend Secure Comput 1(1):11–33
4. Avizienis A, Laprie JC (1986) Dependable computing: From concepts to design diversity. Proceedings of the IEEE 74:629–638. https://doi.org/10.1109/proc.1986.13527
5. Bauer E (2011) Design for reliability: information and computer-based systems. Wiley, New Jersey
6. Bauer E, Adams R (2012) Reliability and availability of cloud computing. Wiley, New Jersey

7. Callou G, Andrade E, Ferreira J (2019) Modeling and analyzing availability, cost and sustainability of it data center systems. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 2127–2132. IEEE
8. Čepin M (2011) Reliability block diagram. In: Assessment of Power System Reliability, pp. 119–123. Springer, Berlin https://doi.org/10.1007/978-0-85729-688-7_9
9. Communications AG (2021) Ansi/tia-942 (Telecommunications infrastructure standard for data centers). https://www.tic.ir/Content/media/article/TIA%20942%20-A(2012)_0.PDF. Last accessed 09 Set 2021
10. Dhanujati N, Girsang AS (2018) Data center-disaster recovery center (dc-drc) for high availability it service. In: 2018 International Conference on Information Management and Technology (ICIMTech), pp. 55–60. IEEE
11. Dillon T, Wu C, Chang E (2010) Cloud computing: issues and challenges. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 27–33. IEEE
12. Dixon PM (2006) Bootstrap resampling. Encyclopedia of environmetrics **1**
13. Geng H (2014) Data centers-strategic planning, design, construction, and operations. In: Data Center Handbook, pp. 1–14. Wiley, New Jersey Inc. https://doi.org/10.1002/9781118937563.ch1
14. Gray J, Siewiorek D (1991) High-availability computer systems. Computer 24:39–48. https://doi.org/10.1109/2.84898
15. Hamby DM (1994) A review of techniques for parameter sensitivity analysis of environmental models. Environ Monitor Assess 32(2):135–154
16. Hayes B (2008). Cloud computing
17. Kuo W, Zuo MJ (2003) Optimal reliability modeling: principles and applications. Wiley, New Jersey
18. Lee AS (1989) A scientific methodology for mis case studies. MIS quarterly pp. 33–50
19. Liu T, Song H (2003) Dependability prediction of high availability oscar cluster server. In: Proceedings of the 2003 International Conference on Parallel and Distributed Processing Techniques and Applications. Citeseer
20. LLC UI (2021) Uptime institute. https://uptimeinstitute.com/. Last accessed 09 Set 2021
21. Maciel P, Trivedi K, Matias Jr R, Kim D (2012) Performance and dependability in service computing, p. 45. IGI Global. https://doi.org/10.4018/978-1-60960-794-4.ch003
22. Marsan MA, Chiola G (1986) On petri nets with deterministic and exponentially distributed firing times. In: European Workshop on Applications and Theory in Petri Nets, pp. 132–145. Springer
23. Matos R, Araujo J, Oliveira D, Maciel P, Trivedi K (2015) Sensitivity analysis of a hierarchical model of mobile cloud computing. Simulat Modell Practice Theory 50:151–164
24. Matos R, Dantas J, Araujo J, Trivedi KS, Maciel P (2017) Redundant eucalyptus private clouds: availability modeling and sensitivity analysis. J Grid Comput 15(1):1–22
25. Melo C, Dantas J, Pereira P, Maciel P (2021) Distributed application provisioning over ethereum-based private and permissioned blockchain: availability modeling, capacity, and costs planning. J Supercomput. https://doi.org/10.1007/s11227-020-03617-z
26. Melo C, Matos R, Dantas J, Maciel P (2017) Capacity-oriented availability model for resources estimation on private cloud infrastructure. In: 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 255–260. IEEE
27. Mesbahi MR, Rahmani AM, Hosseinzadeh M (2018) Reliability and high availability in cloud computing environments: a reference roadmap. Human-centric Comput Info Sci 8(1):1–31
28. Molloy MK (1982) Performance analysis using stochastic petri nets. IEEE Comp Arch Letter 31(09):913–917
29. Murata T (1989) Petri nets: Properties, analysis and applications. Proceed IEEE 77(4):541–580
30. Pereira P, Araujo J, Melo C, Santos V, Maciel P (2021) Analytical models for availability evaluation of edge and fog computing nodes. J Supercomput. https://doi.org/10.1007/s11227-021-03672-0
31. Pereira P, Araujo J, Torquato M, Dantas J, Melo C, Maciel P (2020) Stochastic performance model for web server capacity planning in fog computing. J Supercomput 76(12):9533–9557
32. Peterson JL (1977) Petri nets. ACM Comput Surv (CSUR) 9(3):223–252
33. Peterson JL (1981) Petri net theory and the modeling of systems. Prentice Hall PTR, New Jersey
34. Petri CA (1966) Communication with automata. Ph.D. thesis, Universität Hamburg
35. Pinheiro T, Oliveira D, Matos R, Silva B, Pereira P, Melo C, Oliveira F, Tavares E, Dantas J, Maciel P (2021) The mercury environment: A modeling tool for performance and dependability evaluation. In: Intelligent Environments 2021, pp. 16–25. IOS Press

36. Rahman A, Liu X, Kong F (2013) A survey on geographic load balancing based data center power management in the smart grid environment. IEEE Commun Surv Tutorial 16(1):214–233
37. Rosendo D, Leoni G, Gomes D, Moreira A, Gonçalves G, Endo P, Kelner J, Sadok D, Mahloo M (2018) How to improve cloud services availability? investigating the impact of power and it subsystems failures. In: Proceedings of the 51st Hawaii International Conference on System Sciences
38. Santos GL, Endo PT, Gonçalves G, Rosendo D, Gomes D, Kelner J, Sadok D, Mahloo M (2017) Analyzing the it subsystem failure impact on availability of cloud services. In: 2017 IEEE Symposium on Computers and Communications (ISCC), pp. 717–723. IEEE
39. Sousa E, Lins F, Tavares E, Maciel P (2017) Cloud infrastructure planning considering different redundancy mechanisms. Computing 99(9):841–864
40. Torquato M, Guedes E, Maciel P, Vieira M (2019) A hierarchical model for virtualized data center availability evaluation. In: 2019 15th European Dependable Computing Conference (EDCC), pp. 103–110. IEEE
41. Torquato M, Umesh I, Maciel P (2018) Models for availability and power consumption evaluation of a private cloud with vmm rejuvenation enabled by vm live migration. J Supercomput 74(9):4817–4841
42. Turner IV WP, PE J, Seader P, Brill K (2006) Tier classification define site infrastructure performance. Uptime Institute **17**
43. Wang D, Trivedi KS (2005) Computing steady-state mean time to failure for non-coherent repairable systems. IEEE Trans reliability 54(3):506–516