UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

PROGRAMA DE PÓS–GRADUAÇÃO EM INFORMÁTICA APLICADA

IVSON GALDINO BORGES

AVALIAÇÃO DE DESEMPENHO, DISPONIBILIDADE E CONSUMO DE ENERGIA DE UM AMBIENTE DE VIDEOMONITORAMENTO COM ARMAZENAMENTO DISTRIBUÍDO

RECIFE - PE

2023

IVSON GALDINO BORGES

AVALIAÇÃO DE DESEMPENHO, DISPONIBILIDADE E CONSUMO DE ENERGIA DE UM AMBIENTE DE VIDEOMONITORAMENTO COM ARMAZENAMENTO DISTRIBUÍDO

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

ORIENTADOR: Prof. Dr. Gustavo Rau de Almeida Callou

RECIFE - PE

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

B732a Borges, Ivson Galdino
Avaliação de desempenho, disponibilidade e consumo de energia de um ambiente de videomonitoramento com armazenamento distribuído / Ivson Galdino Borges. - 2023.
121 f. : il.

Orientador: Gustavo Rau de Almeida Callou.
Inclui referências e apêndice(s).

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2024.

1. Armazenamento distribuído. 2. Avaliação de desempenho. 3. Disponibilidade. 4. Modelagem. 5. Consumo de energia. I. Callou, Gustavo Rau de Almeida, orient. II. Título

CDD 004

IVSON GALDINO BORGES

AVALIAÇÃO DE DESEMPENHO, DISPONIBILIDADE E CONSUMO DE ENERGIA DE UM AMBIENTE DE VIDEOMONITORAMENTO COM ARMAZENAMENTO DISTRIBUÍDO

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

Aprovada em: 29 de Agosto de 2023.

BANCA EXAMINADORA

Prof. Dr. Gustavo Rau de Almeida Callou (Orientador)
Universidade Federal Rural de Pernambuco
Departamento de Computação

Prof. Dr. Ermeson Carneiro de Andrade
Universidade Federal Rural de Pernambuco
Departamento de Computação

Prof. Dr. Francisco Airton Pereira da Silva
Universidade Federal do Piauí

Dedico este trabalho à minha família, com imensa gratidão e amor. O encorajamento e o carinho de vocês foram fundamentais para superar os desafios.

Agradecimentos

Sou grato a Deus, por Sua Bondade e Graça, creio que tudo é Dele, para Ele e Nele tudo subsiste.

Agradeço a minha esposa Jéssica por acreditar em mim e ser minha maior incentivadora nos momentos de desânimo e celebrar comigo cada passo dado em direção aos meus objetivos.

Agradeço a meu filho Samuel, que com seu sorriso e sua presença ilumina meus dias e me motivam a perseverar em busca dos meus objetivos.

Aos meus pais, irmãos, sogros e cunhados, vocês foram minha motivação e força nas horas mais difíceis. Cada conquista é também de vocês, pois cada passo que dei foi sustentado pelo amor e confiança que depositaram em mim.

Agradeço ao Prof. Dr. Gustavo Callou pela acolhida e orientação durante toda a pesquisa, sempre incentivando e direcionando para o desenvolvimento deste trabalho, além de ser grande inspiração como ser humano e profissional para todos nós.

Agradeço o apoio e colaboração dos colegas do PeDAL (*Performance Evaluation and Data Analysis Lab*) sempre dispostos a ajudar e trocar conhecimentos.

Aos colegas de trabalho da Escola Técnica Maria Eduarda Ramos de Barros e da Secretaria de Tecnologias Digitais da UFRPE, pelas palavras de incentivo para seguir adiante.

... persistence is a friend of conquest.

(Bill Gates)

Resumo

Sistemas de videomonitoramento têm sido amplamente utilizados como ferramenta para controle de acesso, inibição de crimes, prevenção de acidentes e segurança interna em diversos estabelecimentos. O armazenamento dos dados e imagens gerados por esses sistemas é um grande desafio, pois a quantidade de dispositivos, a qualidade das imagens e o tempo necessário de armazenamento vêm impactando na utilização de recursos de hardware e, consequentemente, no consumo elétrico. Avaliar a relação entre a quantidade de câmeras e as formas de armazenamento é crucial para propor estratégias que garantam confiabilidade e disponibilidade com o menor custo possível. É importante verificar os níveis de criticidade das imagens a serem armazenadas para definir qual a configuração de armazenamento será utilizada. A avaliação de aspectos de disponibilidade e desempenho é uma grande preocupação nesses ambientes computacionais, muitos desafios são enfrentados para garantir o pleno funcionamento desses sistemas. Para sistemas de videomonitoramento a disponibilidade é essencial, pois falhas podem comprometer a eficácia do monitoramento. A partir de uma arquitetura base, este trabalho fez a proposição de novas arquiteturas após a análise e adoção de técnicas de redundância baseadas em modelos em diagrama de blocos de confiabilidade (RBD) e redes de Petri estocásticas (SPN). Com a arquitetura base, foi obtida a disponibilidade de 99,66%, o que corresponde a 29,63 horas de indisponibilidade por ano. Em uma das arquiteturas propostas, com estratégias de redundância, a disponibilidade alcançou 99,999%, com 3,85 minutos de indisponibilidade anual, o que indica uma redução de 99,8%. Os modelos analíticos propostos neste trabalho podem dar suporte ao planejamento de infraestruturas computacionais para sistema de videomonitoramento. Em relação ao consumo elétrico e de desempenho, os resultados obtidos nesse trabalho indicam que a forma de armazenamento (ex., com volumes distribuídos ou replicados) geram impactos distintos tanto no nível de utilização de hardware como no consumo elétrico. Por exemplo, foi possível observar que o uso de volumes replicados aumenta o consumo elétrico em cerca de 6,17% com 100 câmeras, em relação ao modo distribuído, e que o maior impacto no desempenho é na utilização de CPU do sistema de gerenciamento de vídeo independente, do modo de armazenamento.

Palavras-chave: Armazenamento Distribuído. Avaliação de Desempenho. Disponibilidade. Modelagem. Consumo de energia.

Abstract

Video surveillance systems have been widely used as a tool for access control, crime inhibition, accident prevention, and internal security in several establishments. The storage of data and images generated by these systems is a major challenge since the number of devices, the quality of the images, and the time required for storage have impacted the use of hardware resources and, consequently, electrical consumption. Evaluating the relationship between the number of cameras and the forms of storage is crucial to propose strategies that ensure reliability and availability at the lowest possible cost. It is important to verify the criticality levels of the images to be stored to define which storage configuration will be used. The evaluation of availability and performance aspects is a major concern in these computing environments, and many challenges are faced to ensure the full operation of these systems. For video monitoring systems, availability is essential, as failures can compromise the monitoring effectiveness. This work proposed new architectures from a baseline architecture after analyzing and adopting redundancy techniques based on reliability block diagram models (RBD) and stochastic Petri nets (SPN). Assuming the baseline architecture, we obtained an availability of 99.66%, corresponding to 29.63 hours of unavailability per year. In one of the proposed architectures, the achieved availability results reached 99.999%, corresponding to 3.85 minutes of downtime, which indicates a 99.8% reduction in the annual downtime values. The analytical models proposed in this work can support planning computational infrastructures for video surveillance systems. For electrical consumption and performance, the results obtained in this work indicate that the form of storage (e.g., with distributed or replicated volumes) generate different impacts on both hardware utilization and electrical consumption. For example, it was possible to observe that the use of replicated volumes increases electrical consumption by 6.17% with 100 cameras in comparison to the distributed mode. In addition, the greatest impact on performance is on the CPU utilization of the video management system independent of the storage mode.

Keywords: Distributed Storage. Performance evaluation. Availability. Modeling. Energy consumption.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AI	*Availability Importance*
API	*Application Programming Interface*
AVC	*Advanced Video Coding*
CCTV	*Closed-Circuit Television*
COA	*Capacity-Oriented Availability*
CPU	*Central Processing Unit*
CTMC	*Continuous Time Markov Chains*
EPS-RAID	*Extra-Parity S-RAID*
FUSE	*Filesystem in Userspace*
GEC	*Geoaware Erasure Coding*
GFS	*Google File System*
HDFS	*Hadoop Distributed File System*
HTTPS	*Hyper Text Transfer Protocol Secure*
IOPS	*Input/Output per Second*
IoT	*Internet of Things*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
kWh	*kilowatt-hours*
MTTF	*Mean Time to Failure*
MTTR	*Mean Time to Repare*
NIC	*Network Interface Card*
PG	*Placement Groups*
PIR	*Passive Infrared*
PTZ	*Pan, Tilt e Zoom*
RAID	*Redundant Array of Inexpensive Drives*
HD	*Hard Disk*

RAM	*Random Access Memory*
RBD	*Reliability Block Diagram*
RPM	Rotações por minuto
RTSP	*Real Time Streaming Protocol*
SAD	Sistema de Arquivos Distribuídos
OS	*Operating System*
SPN	*Stochastic Petri Model*
SSD	*State Solid Drive*
TTF	*Time to Failure*
TTR	*Time to Repair*
UTP	*Unshielded Twisted Pair*
VFS	*Virtual File System*
VMS	*Video Management System*

Sumário

1 Introdução

Nos últimos anos, o desenvolvimento em diversos setores socioeconômicos tem impactado significativamente a vida das pessoas em várias áreas, incluindo a segurança e proteção individual. Um exemplo disso é a crescente instalação de câmeras de segurança em espaços públicos e privados, com o objetivo de melhorar a segurança por meio do monitoramento contínuo desses ambientes. O videomonitoramento oferece benefícios como controle de acesso, vigilância remota, prevenção de crimes e emissão de alertas, além de ser utilizado em fins comerciais, permitindo identificar o perfil e o comportamento dos clientes e, assim, otimizar a disposição de produtos para uma experiência agradável de consumo. O mercado global de segurança eletrônica foi avaliado em US$ 40,93 bilhões em 2020 e espera-se que atinja US$ 68,57 bilhões até 2026, de acordo com a consultoria Mordor Intelligence (INTELLIGENCE, 2020).

Durante a pandemia de COVID-19, a vigilância por vídeo IP (Internet Protocol) foi amplamente utilizada em diversas localidades, principalmente para fins de segurança, mas também no monitoramento do cumprimento de normas sanitárias. As câmeras de última geração empregadas nesse contexto possuem alto poder de computação e recursos avançados de análise de vídeo, o que permite aos usuários transformar imagens em tempo real em análises de *big data*. Em conformidade com as práticas e normas de distanciamento social, várias técnicas de análise de vídeo foram amplamente implementadas para adotar medidas de precaução, como o gerenciamento de filas, a contagem de pessoas, a detecção de multidões e a verificação do uso de equipamento de proteção individual.

Nesse sentido, os sistemas de vigilância por vídeo surgem como ferramenta de apoio para realização de tarefas como, por exemplo, controle de tráfego, prevenção de acidentes e crimes, detecção de movimento e segurança interna em aeroportos, estações de trem, estacionamentos, rodovias, shoppings, escritórios, hospitais, dentre outros. Atualmente, os sistemas de monitoramento por vídeo são digitais, demandam uma alta definição e fazem uso da rede local (e até da Internet) para transmissão, armazenamento e acesso às imagens. Importante ressaltar que, assim como, a quantidade de dispositivos de vigilância tem aumentado, a demanda por uma maior qualidade da imagem passou a ser exigida, impactando significativamente no tempo necessário para transmissão, na utilização de recursos computacionais e, consequentemente, no consumo elétrico. Sendo

assim, estudos que visam encontrar uma forma de armazenamento desses dados é de fundamental importância, uma vez que deve possibilitar acesso rápido, e garantia da disponibilidade e confiabilidade com eficiência energética.

As falhas em sistemas de videomonitoramento podem acarretar uma série de problemas, incluindo o aumento do risco de atividades criminosas não detectadas, a impossibilidade de coletar evidências essenciais, causando impactos na segurança pública, por exemplo. Portanto, manter a operação confiável e a alta disponibilidade desses sistemas é imperativo para evitar tais consequências negativas. A alta disponibilidade é necessária para aplicações de videomonitoramento, e diferentes técnicas podem ser aplicadas para atingir uma disponibilidade que seja considerada aceitável para o sistema. O planejamento de um sistema de monitoramento com alta disponibilidade é uma tarefa desafiadora para garantir a qualidade e o funcionamento contínuo. É comum ocorrerem falhas parciais nos equipamentos de vigilância, o que pode comprometer a eficácia do monitoramento ou até mesmo levar a uma falha completa do sistema. Portanto, é fundamental realizar estudos e análises detalhadas para garantir a máxima disponibilidade desses sistemas. Além disso, os sistemas de vigilância devem ser altamente confiáveis, uma vez que estão, muitas vezes, intimamente relacionados à segurança das pessoas.

Este trabalho visa avaliar o consumo de energia, o desempenho e a disponibilidade de um ambiente de videomonitoramento com armazenamento distribuído. Sendo assim, este trabalho faz a proposição de uma estratégia computacional para avaliar, comparar e estimar o consumo elétrico, o desempenho e a disponibilidade de um sistema de videomonitoramento utilizando modos de armazenamento distribuído com e sem réplicas. Experimentos foram projetados para monitorar e avaliar métricas como utilização de CPU, memória, rede, disco e consumo elétrico. Também foram realizadas projeções do comportamento do sistema com uma quantidade maior de câmeras. No contexto da disponibilidade, este trabalho propõe modelos em RBD e SPN para a análise da disponibilidade de um sistema de videomonitoramento.

1.1 Motivação e Justificativa

É notório que um sistema de videomonitoramento pode ser aplicado em várias situações, como em segurança pública, monitorando infraestruturas críticas e até mesmo ambientes hospitalares (ELHARROUSS; ALMAADEED; AL-MAADEED, 2021; KARIMAA, 2011). Nesse sentido, aspectos como consumo elétrico, desempenho dos recursos computacionais e disponibilidade são três fatores importantes que precisam ser considerados ao projetar, implantar e operar um ambiente de videomonitoramento. Os sistemas de videomonitoramento podem consumir muita energia, especialmente se utilizarem grande quantidade de câmeras. Além disso, o processamento e o armazenamento das imagens, geram custos relacionados ao consumo elétrico (ANDREI et al., 2019). É importante considerar o consumo elétrico dos sistemas de videomonitoramento ao projetar a infraestrutura de energia do ambiente. A análise permite compreender e otimizar o uso da energia, garantindo a eficiência energética e reduzindo custos operacionais. Além disso, os sistemas de videomonitoramento também podem ser exigentes em termos de recursos computacionais, como utilização de CPU e memória, por exemplo. Eles precisam ser capazes de armazenar e processar grandes quantidades de dados de vídeo. Portanto, é importante garantir que os sistemas de videomonitoramento tenham os recursos computacionais necessários para seu pleno funcionamento (BANU et al., 2017). Em relação à disponibilidade, os sistemas de videomonitoramento precisam estar disponíveis 24 horas por dia, 7 dias por semana e podem estar lidando com informações sensíveis e vitais, como em sistemas de segurança pública, vigilância de infraestruturas críticas e até mesmo monitoramento em ambientes hospitalares (ELHARROUSS; ALMAADEED; AL-MAADEED, 2021; SRINIVASAN et al., 2021). A análise de disponibilidade permite identificar falhas, pontos de vulnerabilidade e possibilita a adoção de estratégias para garantir que o sistema esteja operacional de forma confiável e contínua, evitando lacunas na cobertura de monitoramento (KARIMAA, 2011).

1.2 Objetivos

A utilização de sistemas de videomonitoramento requer alta disponibilidade e o armazenamento das imagens por determinado período, para isso é necessário prover acesso a recursos computacionais suficientes à demanda visando garantir a confiabilidade e a disponibilidade para o armazenamento e acesso às imagens, aliados ao consumo eficiente de energia elétrica. Dessa forma, a presente pesquisa tem como questão principal: **Qual estratégia pode ser utilizada para garantir o funcionamento de um sistema de videomonitoramento com níveis satisfatórios de desempenho, disponibilidade e consumo de energia?**

Esta pesquisa tem como objetivo a proposição de uma estratégia de avaliação de desempenho, disponibilidade e consumo de energia de um sistema de videomonitoramento configurado com armazenamento distribuído. Dessa forma, o armazenamento configurado com volumes distribuídos e replicados serão analisados e os resultados comparados quanto a métricas de desempenho e de consumo de energia, além de avaliar a disponibilidade do sistema. Para atingir tal objetivo, a metodologia proposta irá utilizar técnicas de avaliação de desempenho e disponibilidade. A técnica de medição será utilizada para analisar os recursos computacionais (CPU, memória e rede) e o consumo elétrico e a modelagem analítica será utilizada para avaliar a disponibilidade do ambiente. Os resultados obtidos servirão de subsídios para realização de projetos de videomonitoramento, visando a otimização dos recursos computacionais e da disponibilidade do ambiente projetado.

1.2.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Propor abordagem de medições em um ambiente real com armazenamento com volumes replicados e distribuídos.;
- Propor metodologia para avaliar de forma integrada a disponibilidade, o desempenho e o consumo elétrico do ambiente avaliado;
- Propor modelos para representar o ambiente com a finalidade de avaliar a disponibilidade;
- Propor estratégias para maximizar a disponibilidade do ambiente.

1.3 Estrutura do Trabalho

Além desta introdução, esta dissertação é composta e organizada da seguinte forma: O Capítulo 2 apresenta os trabalhos relacionados a esta pesquisa, mostrando o que já foi estudado sobre videomonitoramento e estratégia de armazenamento de imagens. O Capítulo 3 mostra os conceitos necessários para o entendimento deste trabalho, tais como, sistemas de armazenamento distribuídos, sistema de gerenciamento de vídeo, avaliação de desempenho, disponibilidade, entre outros. Em seguida, o Capítulo 4 apresenta a metodologia aplicada para o desenvolvimento dessa pesquisa. Posteriormente, o Capítulo 5 apresenta os modelos propostos para representar a arquitetura base do sistema de videomonitoramento com armazenamento distribuído. Em sequência, o Capítulo 6 ilustra a aplicabilidade da estratégia proposta em estudos de caso aplicados. Por fim, o Capítulo 7 apresenta as conclusões, limitações sobre todo o trabalho desenvolvido e os direcionamentos dos trabalhos futuros.

2 Trabalhos Relacionados

A tecnologia por videomonitoramento vem avançando recentemente em vários aspectos, e nos últimos anos, alguns estudos propuseram alternativas à forma de armazenamento das imagens obtidas, considerando aspectos como segurança, eficiência e disponibilidade dos dados armazenados. Inicialmente são apresentados trabalhos relacionados a sistemas de arquivos distribuídos e, em seguida, os trabalhos com ênfase em videomonitoramento. Ao final é realizada uma comparação da literatura apresentada com o proposto nesta dissertação, deixando mais claras as contribuições desta pesquisa.

2.1 Sistemas de Arquivos Distribuídos

Os autores em (ACQUAVIVA et al., 2018) fizeram um estudo comparativo em termos de desempenho e confiabilidade das arquiteturas (e dos seus componentes) para os sistemas de armazenamento distribuído HDFS, Ceph, GlusterFS e o XtremeFS. Constatou-se que o GlusterFS e o XtremeFS levam menos tempo para implantação em relação ao HDFS e ao Ceph. O tempo de gravação e de replicação também foram avaliados, observando o comportamento de cada sistema a medida em que cada carga de trabalho é gerada. Percebe-se que o GlusterFS apresenta o melhor tempo de propagação no cenário com replicação. Em relação ao tempo de gravação, o Ceph leva mais tempo do que o HDFS e o GlusterFS. Isso pode ter ocorrido devido à estrutura do Ceph, que tem a presença de *Placement Groups* (PGs) que precisam ser atualizados e consomem tempo. O HDFS possui um ponto centralizado, onde os metadados são armazenados. Enquanto Ceph, GlusterFS e XtremeFS possuem diferentes pontos onde são armazenados os metadados. Os autores não tiveram como objetivo avaliar os sistemas considerando suas possibilidades de configuração, recursos de hardware e nem tiveram como foco analisar o consumo elétrico.

Os autores em (YU et al., 2020) apresentaram um sistema de armazenamento distribuído para câmeras de vigilância com características para preservação da privacidade com robustez e de baixo custo. O objetivo da estratégia é a defesa contra ataques que podem capturar/danificar uma ou mais câmeras de determinada localidade. Com a utilização de um algoritmo denominado *Geoaware Erasure Coding* (GEC) o sistema gera

blocos de dados codificados e entre câmeras de vigilância geograficamente dispersas. Dessa forma, garante robustez de todo o sistema, diante ataques de alcance geográfico. Os dados apresentados pelos autores compara, através de experimentos, o número de dispositivos sob ataque em relação ao número de dispositivos onde os dados não podem ser recuperados. Esse estudo não apresentou os impactos no desempenho dos recursos de hardware, nem teve como foco a análise do consumo de energia na utilização da estratégia.

Em (KIRBY et al., 2019) os autores avaliaram o GlusterFS em máquinas virtuais, as métricas avaliadas foram: velocidades de leitura e gravação considerando dados de 4KB e 512KB, teste de latência e Operações de entrada e saída por segundo (IOPS). Segundo os autores o grande fator limitante ao GlusterFS é a estabilidade e disponibilidade da rede, pois o GlusterFS não requer um hardware excepcional, em cargas de trabalho intensas o hardware não prejudica o funcionamento do sistema. O cenário avaliado não considerou métricas como utilização de CPU, nem uso da rede ou de recursos energéticos necessários para garantir o funcionamento pleno do GlusterFS.

Os autores em (ABUD; GOFF; AVOLIO, 2020) apresentam uma avaliação de desempenho de três sistemas de arquivos distribuídos: Gluster, Hadoop e Ceph. Os experimentos realizados demonstraram, por meio dos resultados, que a taxa de transferência para leitura sequencial é a semelhante nos três sistemas avaliados. Por outro lado, o sistema de arquivos Gluster apresentou um desempenho excepcionalmente bom para escrita sequencial em comparação com o Hadoop e o Ceph. Esse resultado é promissor, pois sugere que a arquitetura adotada pelo Gluster, que não utiliza servidores de metadados dedicados, pode ser vantajosa para aplicações que demandam alto desempenho em operações de escrita. Concluem que a escolha de tecnologias que não dependem de servidores de metadados dedicados, como os bancos de dados chave-valor, pode ser um diferencial significativo para o desempenho geral do sistema de armazenamento distribuído. Os autores não tiveram como objetivo avaliar a disponibilidade nem o consumo elétrico dos sistemas analisados.

Os autores em (PAREKH; GAUR; GARG, 2020) propuseram modelos em SPN para o GlusterFS e CephFS. A análise de desempenho orientada por modelos é uma técnica importante para analisar o desempenho de qualquer sistema. Portanto, os detalhes internos dos sistemas de arquivos distribuídos foram estudados em profundidade e as operações de arquivos foram modeladas. Modelos GSPN foram projetados e simulações foram executadas usando a ferramenta *Pipe* para analisar a disponibilidade do GlusterFS e

CephFS em relação ao aumento do número de servidores. Observou-se que, com o mesmo número de servidores e fator de replicação, a disponibilidade do GlusterFS foi melhor do que a do CephFS. O trabalho teve como objetivo avaliar a disponibilidade em dois sistemas de arquivos distribuídos, mas não houve análise do desempenho ou consumo elétrico.

2.2 Videomonitoramento

Em (ZHAO; CUI; ZHANG, 2012) foi proposto um modelo estrutural para armazenar as imagens obtidas por um sistema de videomonitoramento, destacando a importância de garantir escalabilidade, acesso aos dados pela internet, alta disponibilidade (tolerância a falhas) e com baixo custo. O estudo concluiu que o armazenamento em nuvem mitigou os gargalos de desempenho e as limitações do armazenamento tradicional (local). Embora indique os benefícios e vários aspectos, o estudo não realizou medições para avaliar o desempenho ou o impacto do consumo elétrico para manter o modelo estrutural proposto.

Em (DAI; LIU, 2020) foi proposto um ambiente de larga escala destinado a sistemas de videovigilância. Os dados dos vídeos são organizados por distribuição com a fragmentação do índice combinado com operações de banco de dados distribuídos. O cenário proposto apresenta uma estratégia que se adapta a crescente demanda dos sistemas de videomonitoramento adotando o armazenamento distribuído das imagens capturadas, de forma a garantir disponibilidade da infraestrutura. O trabalho não teve como foco a avaliação do desempenho e nem do consumo de energia.

Os autores em (XIAO et al., 2018) propuseram uma arquitetura RAID (*Redundant Array of Inexpensive Disks*), denominada EPS-RAID (*extra parity S-RAID*) que adiciona um disco de paridade e um SSD (*Solid State Disk*) visando otimizar as leituras e gravações aleatórias realizadas na estrutura de armazenamento, são avaliados o desempenho nas operações de leitura e gravação, e de consumo elétrico. Concluíram que as operações realizadas no ambiente proposto torna-se mais eficiente, considerando o modo de trabalho e o modo ocioso, possibilitando a economia de energia, pois os discos estarão ativos nos momentos específicos de uso. O sistema proposto busca garantir a disponibilidade dos dados armazenados por meio da redundância de discos. Entretanto, o trabalho não teve como foco avaliar a disponibilidade em busca de propor melhorias que pudessem maximizar os valores da disponibilidade.

O trabalho (RAJAVEL et al., 2022) propôs aprimoramentos na capacidade de computação de borda para otimizar a eficiência e a inteligência do sistema de vigilância por vídeo baseado em IoT aplicado a saúde. Essas melhorias visam reduzir a largura de banda da rede e o tempo de resposta em comparação com os sistemas existentes. Além disso, o sistema proposto apresenta um tempo de resposta significativamente menor devido à implementação de mecanismos robustos de detecção de objetos e reconhecimento de quedas no *datacenter* em nuvem, utilizando técnicas como subtração de fundo e algoritmos de rede neural. O trabalho não explora aspectos de disponibilidade ou de consumo energético.

Os autores em (ADVIRKAR et al., 2020) propuseram uma estratégia de funcionamento de um ambiente de videomonitoramento com a utilização de sensores PIR(*Passive Infrared*) visando otimizar a obtenção de imagens e seu armazenamento. O sistema apresentado pode detectar, reconhecer e captar imagens de pessoas externas e, por meio de pesquisa, pode concluir se a pessoa poderá ter acesso ou não. Há uma preocupação em melhorar a eficiência do sistema acelerando o processo de correspondência de padrões, para fornecer resultados mais precisos e aumentando a segurança. No entanto, não foi objetivo do trabalho apresentar aspectos como: desempenho dos recursos de hardware, consumo de energia e disponibilidade.

Em (KAVALIONAK et al., 2019) foi apresentado um novo protocolo distribuído para um sistema de reconhecimento facial que explora as capacidades computacionais dos dispositivos de vigilância (ou seja, câmeras) para realizar o reconhecimento de uma pessoa. As câmeras recorrem ao servidor centralizado se seus recursos de hardware não forem suficientes para realizar o reconhecimento. Para avaliar o algoritmo proposto, simularam e testaram os algoritmos de classificação 1NN e kNN ponderada por meio de experimentos extensivos em um conjunto de dados. Por meio de simulações, os autores mostraram que o algoritmo é capaz de reduzir em até 50% a carga do servidor sem nenhum impacto negativo na qualidade do serviço de vigilância. Embora aborde aspectos de desempenho, algumas métricas não são avaliadas, por exemplo, CPU, Memória e Uso de disco. Aspectos como consumo de energia e disponibilidade não são avaliados.

Os autores em (SULTANA; WAHID, 2019), apresentaram o design, a implantação e a avaliação de desempenho de um sistema de vigilância inteligente baseado em *Fog Computing* e orientado a eventos para detecção de crimes em tempo real e gerenciamento de segurança. O sistema superou outros em eficiência energética, largura de banda e uso da

CPU. Concluíram após uma análise comparativa geral, a preeminência do sistema proposto sobre outros, dados os requisitos de vigilância por vídeo. O cenário avaliado considera apenas a detecção de objetos, como armas de fogo e facas. Embora tenha analisado métricas relacionadas ao desempenho e a energia, o trabalho não avalia a disponibilidade do sistema.

Os autores em (SRINIVASAN et al., 2021) diante da necessidade de um sistema de videomonitoramento leve, robusto que deve funcionar 24 horas por dia, 7 dias da semana, visando detecção de pessoas, detecção de violação de distanciamento social, detecção de rosto e de máscara facial, para isso faz uso de diferentes modelos de detecção, avaliando o desempenho dos algoritmos de detecção, a partir da aplicação de técnicas de aumento de dados para lidar com a falta de conjunto de dados disponíveis. Embora o desempenho seja objeto do trabalho, não é apresentado resultados referentes a utilização de recursos computacionais como CPU e memória, por exemplo. Não há também avaliação do impacto do consumo de energia, nem análise de disponibilidade.

Em (ANDREI et al., 2019) os autores analisaram um sistema de segurança composto por um sistema de CCTV - *Close Circuit Television* e um sistema de alarme anti-furto instalado em um edifício residencial. A análise considerou dois aspectos: (1) o impacto do consumo de energia do sistema de segurança e (2) os custos refletidos na conta de luz. O estudo comparou ainda o consumo do sistema de segurança projetado e instalado em um edifício residencial em relação ao consumo médio anual de um imóvel. O sistema avaliado não possui armazenamento com réplicas e/ou estratégias que garantam o armazenamento redundante das imagens e nem tiveram o objetivo de analisar a disponibilidade.

Em (AHMAD et al., 2019) foi proposto um modelo híbrido de videomonitoramento, com a utilização de uma câmera aérea centralizada que orienta o funcionamento das demais câmeras. Os custos de instalação e o consumo de energia são avaliadas e concluem que o modelo proposto torna o consumo de energia eficiente, embora não tenha abordado questões como a disponibilidade e desempenho dos recursos computacionais de hardware nos elementos que formam o sistema.

Os autores em (MA et al., 2018) apresentaram um modelo de armazenamento de vídeo para lidar com a baixa eficiência existente na recuperação de arquivos de vídeo. Além disso, para se atingir altas demandadas por escalabilidade e disponibilidade de dados foi proposto um índice aos quadros-chave, utilizado para localizar e recuperar o vídeo de

maneira mais rápida. O sistema proposto foi representado em SPN a fim de avaliar a eficiência da estratégia proposta. Os experimentos realizados pelos autores não tiveram como objetivo avaliar a utilização de recursos de hardware ou de energia, limitando-se a verificar o impacto dos quadros-chave e da eficiência de recuperação do conteúdo.

A Tabela 1 resume uma comparação dos trabalhos relacionados em cinco aspectos: sistema de armazenamento distribuído (SAD), consumo elétrico, videomonitoramento, desempenho e disponibilidade. Todos os trabalhos estão relacionados no contexto de videomonitoramento e/ou armazenamento. Os trabalhos Acquaviva et al. (2018), Xiao et al. (2018), Yu et al. (2020), Kirby et al. (2019), Rajavel et al. (2022), Abud, Goff e Avolio (2020), Kavalionak et al. (2019) e Sultana e Wahid (2019) abordaram aspectos relacionados ao desempenho, já Dai e Liu (2020) propuseram uma estratégia de armazenamento distribuído para garantir a disponibilidade da infraestrutura do sistema e Parekh, Gaur e Garg (2020) aplicaram a técnica de modelagem para avaliar a disponibilidade de sistemas de arquivos distribuídos. O consumo elétrico foi objeto de análise em Andrei et al. (2019), Xiao et al. (2018) e Ahmad et al. (2019).

Diferentemente dos estudos citados, este trabalho propõe uma estratégia para avaliação de desempenho, consumo de energia e disponibilidade de sistemas de videomonitoramento com armazenamento distribuído a partir da utilização de técnicas como regressão linear e validação da disponibilidade. A técnica de regressão linear aplicada ao desempenho e ao consumo elétrico possibilita estimar a utilização de recursos como CPU, memória e os custos de energia elétrica de ambientes com diferentes configurações e número de câmeras em funcionamento. Para a disponibilidade, esta pesquisa aplicou a técnica de validação que visa garantir a confiabilidade dos resultados obtidos a partir dos modelos propostos, verificando a consistência entre esses modelos e o sistema real.

Dessa forma essa pesquisa torna-se relevante, pois reúne características e aspectos desses trabalhos, possibilitando uma evolução para o planejamento de infraestruturas de vigilância por vídeo. Essa pesquisa apresenta uma estratégia de avaliação que auxilia no processo de tomada de decisão na elaboração de projetos para sistemas de videomonitoramento visando garantir recursos computacionais suficientes para proporcionar alta disponibilidade com eficiência energética.

Tabela 1 – Trabalhos Relacionados

Autores	SAD	Energia	Video monitoramento	Desempenho	Disponibilidade
ZHAO; CUI; ZHANG (2012)	X		X		
MA et al.(2018)	X		X		
ACQUAVIVA et al.(2018)	X			X	
ANDREI et al.(2019)		X	X		
DAI; LIU(2020)	X		X		X
XIAO et al.(2018)		X		X	
YU et al.(2020)	X		X	X	
KIRBY et al.(2019)	X			X	
RAJAVEL et al.(2022)			X	X	
ABUD; GOFF; AVOLIO(2020)	X			X	
AHMAD et al.(2019)		X	X		
ADVIRKAR et al.(2020)			X		
KAVALIONAK et al.(2019)			X	X	
SULTANA; WAHID(2019)			X	X	
SRINIVASAN et al.(2021)			X		
PAREKH; GAUR; GARG(2020)	X				X
Esta Pesquisa	X	X	X	X	X

Fonte: O autor

3 Fundamentação Teórica

Este capítulo apresenta um background conceitual sobre os temas abordados para uma melhor compreensão deste trabalho. Inicialmente, são abordados conceitos relacionados a sistemas de armazenamento distribuído e sistemas de gerenciamento de vídeo. Em seguida, são apresentados os conceitos básicos de avaliação de desempenho, regressão linear, disponibilidade, estratégias de redundância, *Reliability Block Diagram* (RBD), *availability importance*, validação de disponibilidade e injeção de falhas.

3.1 Sistemas de Arquivos Distribuídos

Um Sistema de Arquivos Distribuídos (SAD) é um sistema no qual os arquivos estão armazenados e distribuídos em diferentes computadores, interligados por meio de uma rede de comunicação (NETO et al., 2017). Um SAD apresenta o mesmo conceito que um sistema de arquivos local, permitindo que usuários ou programas leiam e escrevam dados que estão aparentemente armazenados na estação local do usuário, contudo os dados se encontram distribuídos em várias máquinas, sendo acessados por uma rede. Essa característica do SAD possibilita que usuários acessem arquivos de qualquer computador da rede de forma transparente para o usuário final ou programa.

Um SAD consiste em um *cluster* de armazenamento de dados responsável por fornecer meios para a leitura e gravação dos diversos tipos de dados. Então, esse *cluster* é formado por vários volumes de armazenamento e nós de serviço de armazenamento. A replicação dos dados consiste em armazenar uma réplica dos dados em diferentes nós, garantindo uma forma de *backup* que possibilita a recuperação automática quando houver falha em algum nó. Além das vantagens inerentes de rápida escalabilidade e maior redundância, sistemas de arquivos distribuídos também podem oferecer a vantagem de fornecer acesso a uma base unificada de dados aos usuários, evitando réplicas desnecessárias de arquivos em máquinas locais. Entretanto, uma desvantagem, que merece destaque, na utilização de sistemas de arquivos distribuídos é a necessidade de uma boa infraestrutura de rede para suportar o constante tráfego de arquivos, o que pode ser ineficiente quando se possui uma rede precária, com baixas taxas de transmissão. Alguns exemplos de sistemas de arquivos distribuídos são o Ceph, StorageOS, *Google File System* (GFS), Lustre e

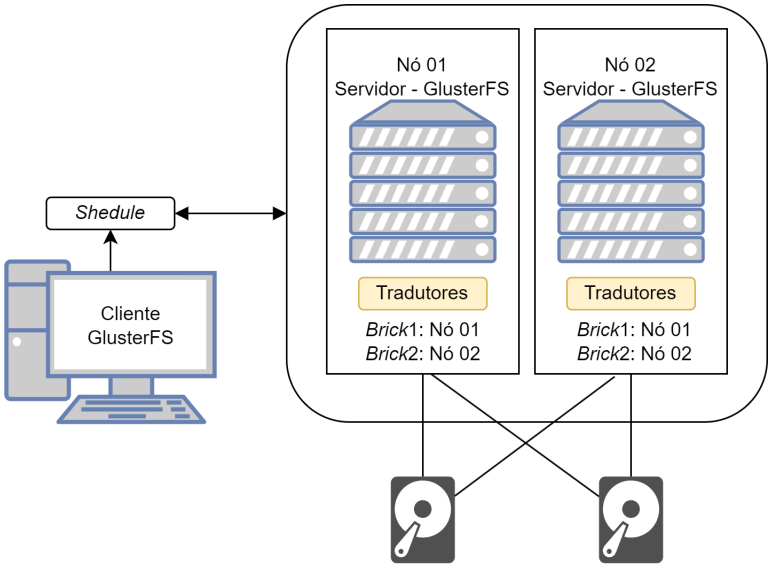
GlusterFS. Esse trabalho utiliza o sistema de arquivos distribuídos GlusterFS.

A implementação dos sistemas de arquivos distribuídos apresentam diversas vantagens, tais como: flexibilidade, o acesso aos arquivos é independente da localização do cliente; transparência, os usuários não percebem a complexidade dos processos envolvidos; escalabilidade, o sistema suporta o aumento de recursos; confiabilidade, possibilita a redução de danos ou perdas de arquivos; integridade, gerencia simultaneidade de acessos a um mesmo arquivo.

3.1.1 GlusterFS

O GlusterFS é utilizado em milhares de organizações que abrangem mídia, saúde, governo, educação e serviços financeiros. O GlusterFS é um SAD caracterizado pela alta escalabilidade, agregando recursos de armazenamento em discos de vários servidores por meio de um *namespace global* (SELVAGANESAN; LIAZUDEEN, 2016). As empresas podem dimensionar a capacidade, o desempenho e a disponibilidade sob demanda, sem dependência de fornecedor, em ambientes locais, de nuvem pública ou híbridas. O GlusterFS pode ser executado sobre vários tipos de sistemas operacionais como o Linux, FreeBSD, OpenSolaris e Mac OS. A arquitetura do GlusterFS é composta pelos seguintes componentes: servidor, cliente e volumes, como mostra a Figura 1.

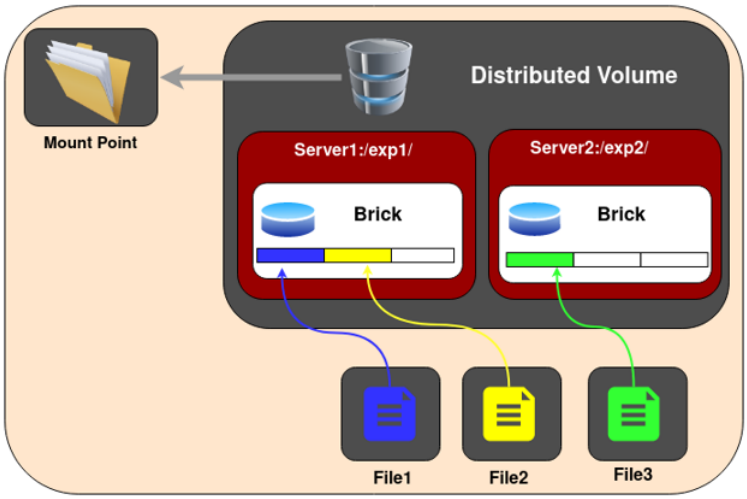
Figura 1 – Arquitetura do GlusterFS



Fonte: O autor.

Nesta arquitetura, o cliente define o ponto de montagem executando o *Glusterfs-client*, que age como um centralizador para *bricks*, que são unidades básicas de armazenamento remotas atribuídas ao volume. Os servidores (*Nodes*) constituem pontos de armazenamento local que abrigam os locais físicos dos *bricks* para que estes sejam disponibilizados aos clientes em volumes lógicos. E, por fim, os volumes, caracterizados por unidades finais, as quais são mapeadas pelos clientes após passarem pela configuração dos tradutores (*translators*) com as especificações neles definidas. Os tradutores são objetos que conectam um ou mais subvolumes e são carregados em tempo de execução por meio de uma interface muito bem definida, referenciadas tanto pelos clientes quanto pelos servidores, caracterizando as funções básicas de cada tipo de volume e suas operações específicas. Outro componente importante do GlusterFS é o *schedule* responsável pela definição da maneira como os arquivos serão disponibilizados entre os componentes do sistema e em conjunto com os tradutores possibilitam a classificação dos tipos de volumes existentes, que são: distribuído (Figura 2), replicado (Figura 3) e disperso (Figura 4). Estes volumes possuem características peculiares e apresentam comportamentos distintos em função dos tradutores.

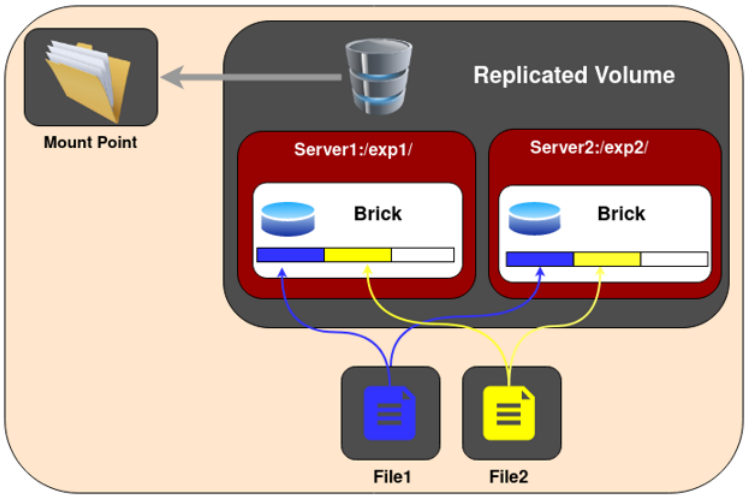
Figura 2 – Gluster com Volumes Distribuídos



Fonte: Página do GlusterFS na internet.[1]

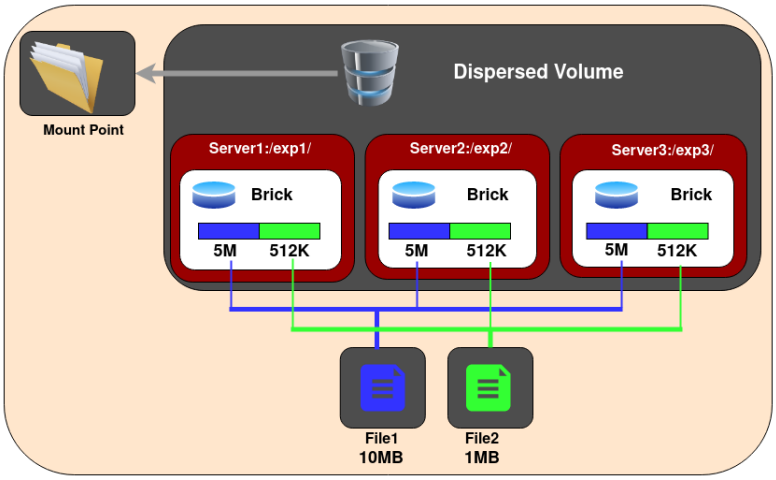
[1] Disponível em: <https://docs.gluster.org/en/latest/Quick-Start-Guide/Architecture/>. Acesso em 20 ago 2023.

Figura 3 – Gluster com Volumes Replicados



Fonte: Página do GlusterFS na internet.[2]

Figura 4 – Gluster com Volumes Dispersos



Fonte: Página do GlusterFS na internet.[3]

No volume distribuído, os arquivos são distribuídos em vários *bricks*. Conforme a Figura 2, o *file1* é armazenado no *brick1* ou *brick2*, mas não em ambos. Nesse tipo de volume, o objetivo é ampliar a capacidade de forma rápida e econômica, sem redundância de dados. Com isso, a falha em um *brick* ocasionará na perda total dos dados ali armazenados. O risco de perder dados devido à falha em algum *brick* é solucionado com a utilização do volume replicado, onde os dados são armazenados e cópias são mantidas em todos os *bricks*

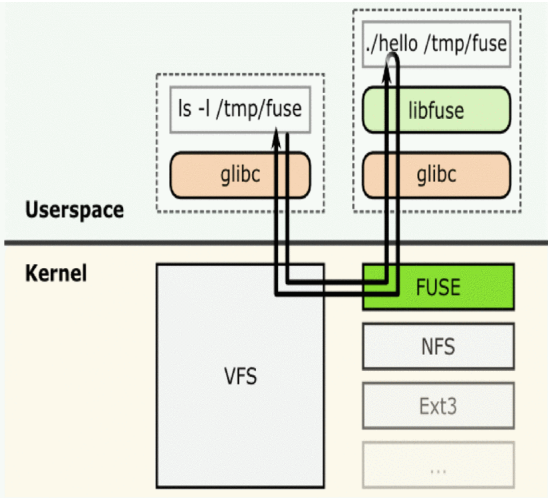
[2] Disponível em: <https://docs.gluster.org/en/latest/Quick-Start-Guide/Architecture/>. Acesso em 20 ago 2023.

[3] Disponível em: <https://docs.gluster.org/en/latest/Quick-Start-Guide/Architecture/>. Acesso em 20 ago 2023.

do cluster, conforme Figura 3. Esse volume é utilizado para garantir maior confiabilidade, pois mesmo com a falha de alguma unidade os dados estarão disponíveis a partir de outros *bricks*. Os volumes dispersos são baseados em códigos de eliminação. Ele distribui os dados codificados dos arquivos, com alguma redundância adicionada, em vários *bricks* no volume, conforme Figura 4. O número de *bricks* redundantes no volume pode ser definido durante a criação do volume e determinam quantos *bricks* podem ser perdidos sem interromper a operação do volume.

O GlusterFS é um sistema de arquivos em espaço de usuário (*userspace*). Essa foi uma decisão tomada pelos desenvolvedores do GlusterFS, uma vez que incluir os módulos no kernel do Linux é um processo longo e difícil. Sendo um sistema de arquivos em espaço de usuário, para interagir com o VFS (*Virtual File System*) do kernel, o GlusterFS utiliza o FUSE (*Filesystem in Userspace*). Por muito tempo, a implementação de um sistema de arquivos em espaço de usuário era considerada impossível. O FUSE foi desenvolvido como uma solução para isso. O FUSE é um módulo do kernel que suporta a interação entre o VFS do kernel e aplicativos de usuário não privilegiados, e possui uma API que pode ser acessada a partir do espaço do usuário. Utilizando essa API, é possível escrever em qualquer tipo de sistema de arquivos, pois existem muitas interfaces entre o FUSE e outras linguagens.

Figura 5 – Diagrama Estrutural do FUSE



Fonte: Página de documentação do GlusterFS.[4]

[4] Disponível em: <https://glusterdocs-beta.readthedocs.io/en/latest/overview-concepts/fuse.html>. Acesso em 20 ago 2023.

O diagrama de sistema de arquivos "*hello world*" apresentado na Figura 5, é compilado para criar um binário "*hello*". Ele é executado com um ponto de montagem do sistema de arquivos /tmp/fuse. Em seguida, o usuário emite um comando *ls-l* no ponto de montagem /tmp/fuse. Esse comando alcança o VFS por meio do *glibc*, já que o ponto de montagem /tmp/fuse corresponde a um sistema de arquivos baseado em FUSE, o VFS o passa para o módulo FUSE. O módulo do kernel FUSE entra em contato com o binário real do sistema de arquivos "*hello*" após passar pelo *glibc* e pela biblioteca FUSE no espaço do usuário (*libfuse*). O resultado é retornado pelo "*hello*" pelo mesmo caminho e chega ao comando ls-1. A comunicação entre o módulo do kernel FUSE e a biblioteca FUSE (*libfuse*) ocorre por meio de um descritor de arquivo especial obtido ao abrir /dev/fuse. Esse arquivo pode ser aberto várias vezes e o descritor de arquivo obtido é passado para a chamada do sistema de montagem, para fazer a correspondência com o descritor e com o sistema de arquivos montado. Neste trabalho, são utilizados volumes distribuídos e replicados para poder atender às necessidades que um ambiente de videomonitoramento requer, como grande capacidade de armazenamento e garantia de confiabilidade e disponibilidade das imagens armazenadas.

3.2 Sistema de Gerenciamento de Vídeo

Um Sistema de Gerenciamento de Vídeo (do inglês *Vídeo Management System - VMS*) é uma plataforma baseada em software utilizada para controlar e gerenciar câmeras de vigilância por vídeo, dispositivos de gravação e outros componentes de segurança. Empresas, governos e outras organizações que precisam de monitoramento e segurança em grande escala geralmente utilizam os sistemas VMS (ISARSOFT, 2023; NEAL; RAHMAN, 2015).

As principais funções de um VMS são:

- Monitoramento de vídeo em tempo real: permite que o pessoal de segurança visualize as transmissões de vídeo das câmeras de vigilância em tempo real, tanto no local quanto remotamente;
- Gravação e armazenamento de vídeo: possibilita a gravação de vídeo de várias câmeras, armazenando os vídeos em servidores ou dispositivos de armazenamento em rede. Além disso, os sistemas VMS oferecem recursos de pesquisa e recuperação

dos vídeos gravados;

- Análise de vídeo: os sistemas VMS podem incorporar análise de vídeo para detectar automaticamente comportamentos suspeitos ou anormais e alertar o pessoal de segurança;
- Integração com outros sistemas de segurança: podem ser integrados a outros sistemas de segurança, como controle de acesso e sistemas de alarme, proporcionando uma plataforma centralizada de gerenciamento de segurança;
- Gerenciamento de usuários: permitem o gerenciamento de usuários, possibilitando que administradores atribuam diferentes níveis de acesso a diferentes usuários.

Ao optar por um sistema de gerenciamento de vídeo, é crucial considerar vários fatores para garantir a seleção da solução adequada (ISARSOFT, 2023). Alguns pontos importantes a serem ponderados incluem:

- Escalabilidade: deve-se levar em conta o tamanho da instalação e o número de câmeras que precisam ser gerenciadas. O VMS deve ter capacidade de expansão ou redução conforme necessário;
- Integração: deve ser capaz de se integrar com outros sistemas de segurança já existentes, como controle de acesso ou sistemas de detecção de intrusão;
- Usabilidade: deve ser intuitivo e de fácil navegação, com uma interface simples que possa ser personalizada para atender às necessidades específicas;
- Análise de vídeo: deve oferecer funcionalidades de análise de vídeo necessárias diretamente ou por meio de complementos de parceiros, ou ser fácil de integrar com software de análise de vídeo;
- Compatibilidade: é possível verificar se o VMS é compatível com as câmeras e dispositivos de gravação que você já possui ou pretende instalar no futuro;
- Confiabilidade: o VMS deve ser confiável e ter um bom histórico de tempo de atividade e desempenho;
- Suporte e Manutenção: o nível de suporte e manutenção oferecido pelo fornecedor do VMS, incluindo treinamento, suporte técnico e atualizações de software;
- Segurança: o VMS deve possuir recursos de segurança robustos, como criptografia e autenticação de usuários, para proteger seus dados de vídeo contra acesso não autorizado.

Em síntese, um VMS oferece uma solução abrangente para vigilância por vídeo e gerenciamento de segurança, permitindo que as organizações monitorem efetivamente suas instalações, ativos e pessoas em tempo real. A escolha do VMS adequado requer uma análise cuidadosa das necessidades e requisitos específicos, bem como a avaliação dos fornecedores potenciais com base em suas capacidades, confiabilidade e suporte oferecido. Neste trabalho, será adotado um sistema de gerenciamento de vídeo, uma vez que sua utilização é considerada crucial para aprimorar a eficácia do monitoramento, permitindo a organização, armazenamento seguro e acesso rápido às imagens, além de facilitar a busca e pesquisa de eventos relevantes, assegurando, desse modo, uma resposta mais rápida e eficiente diante das necessidades. Alguns exemplos de VMS são: shinobi, iSpy e ZoneMinder. Este trabalho utiliza o ZoneMinder.

3.2.1 ZoneMinder

O ZoneMinder é um sistema de gerenciamento de vídeo gratuito, de código aberto, desenvolvido para SO Linux. O ZoneMinder fornece uma interface web para o usuário acrescentar câmeras, definir parâmetros de gravação, visualizar ao vivo e armazenar os dados obtidos. O ZoneMinder utiliza o FFmpeg (um framework de vídeo de código aberto) (ZONEMINDER, 2023). É importante destacar que o ZoneMinder decodifica um fluxo de vídeo de entrada em MJPEG (*Motion Joint Photographic Experts Group*), independentemente do formato de codificação de vídeo de origem, e os quadros de vídeo resultantes são armazenando no formato de imagens JPEG (*Joint Photographic Experts Group*), que podem ser exibidas em interfaces de monitor em tempo real e transcodificadas para o formato H.264 ou H.265, que são padrões de compressão de vídeo e permitem reduzir o tamanho dos arquivos de vídeo mantendo a qualidade. Além disso, o ZoneMinder fornece um mecanismo interno de detecção de movimento analisando as sequências de imagens JPEG armazenadas.

Existem múltiplas formas de acesso ao ZoneMinder, possibilitando que ele seja utilizado a partir de qualquer dispositivo conectado à Internet que possua um navegador web ou ainda por meio de aplicativos para Android e IOS. Câmeras IP com protocolos Onvif, RTSP, HTTPS, com ou sem função PTZ, além de câmeras USB e analógicas são compatíveis com o ZoneMinder. Um fator determinante para a definição do ZoneMinder

como VMS utilizado nos experimentos foi a versatilidade, visto que é possível a implantação de um pequeno sistema, para uso em uma residência e até mesmo sistemas com um número expressivo de câmeras, possibilitando o monitoramento remoto com a utilização de gatilhos ou detecção de movimento que acionam a captura e transmissão da imagem, possibilitando melhor utilização dos espaços de armazenamento e uso de recursos de rede. Neste trabalho, o ZoneMinder será utilizado para processar as imagens capturadas pelas câmeras. Uma vez processadas, as imagens são direcionadas ao armazenamento para fins de preservação e acesso futuro.

3.3 Avaliação de desempenho

A avaliação de desempenho de sistemas pode ser abordada por meio de diferentes técnicas, que podem ser categorizadas em duas vertentes: técnicas baseadas em medição e técnicas baseadas em modelagem (LILJA, 2005).

Na medição é realizada a análise do sistema real, executando cargas de trabalho para obtenção de métricas, possibilitando aferir o comportamento do sistema em determinadas condições de trabalho. Antes de aplicar a carga de trabalho em um sistema, é necessário realizar um estudo preliminar sobre a carga que será aplicada. A definição da carga de trabalho é tão importante quanto a definição da estratégia de medição a ser seguida, pois é a partir dela que serão selecionadas as ferramentas e estratégias de medição adequadas. O PERFMOM, IOSTAT, PROMETHEUS e SYSTAT (BORGES; CALLOU; SILVA, 2023) são algumas ferramentas que podem ser utilizadas para obtenção de métricas como: vazão, taxa de utilização e tempo de resposta (BORGES; CALLOU; SILVA, 2023). É importante ressaltar que ferramentas utilizadas para avaliação de desempenho de sistemas podem afetar o comportamento do sistema avaliado. Quanto mais informações uma ferramenta de medição oferece, maior é a interferência introduzida por ela, isso é minimizado quando é utilizada uma ferramenta externa. Essa alteração pode tornar os dados coletados menos confiáveis (LILJA, 2005). Nesse sentido, ferramentas de medição direcionadas a eventos são consideradas mais confiáveis, pois causam menor interferência nos dados medidos, ocorrendo apenas quando eventos específicos acontecem. No entanto, a frequência dos eventos pode influenciar.

A modelagem analítica é uma abordagem utilizada para abstrair o sistema em estudo. O modelo deve conter os detalhes essenciais do sistema que afetam o seu comportamento. Ao executar o modelo, é possível obter resultados que podem não ser tão precisos quanto os obtidos por meio de medições, mas ainda assim é possível calcular estimativas. Existem diversos tipos de modelos formais que podem ser utilizados para quantificar as métricas de desempenho desejadas. Na modelagem são utilizados modelos matemáticos para representar sistemas reais, podendo, por exemplo, simular o recebimento das requisições de cargas de trabalho em um determinado ambiente. Redes de Petri, cadeias de Markov e teoria de fila são algumas formas de demonstrar tais modelos (MACIEL, 2023b).

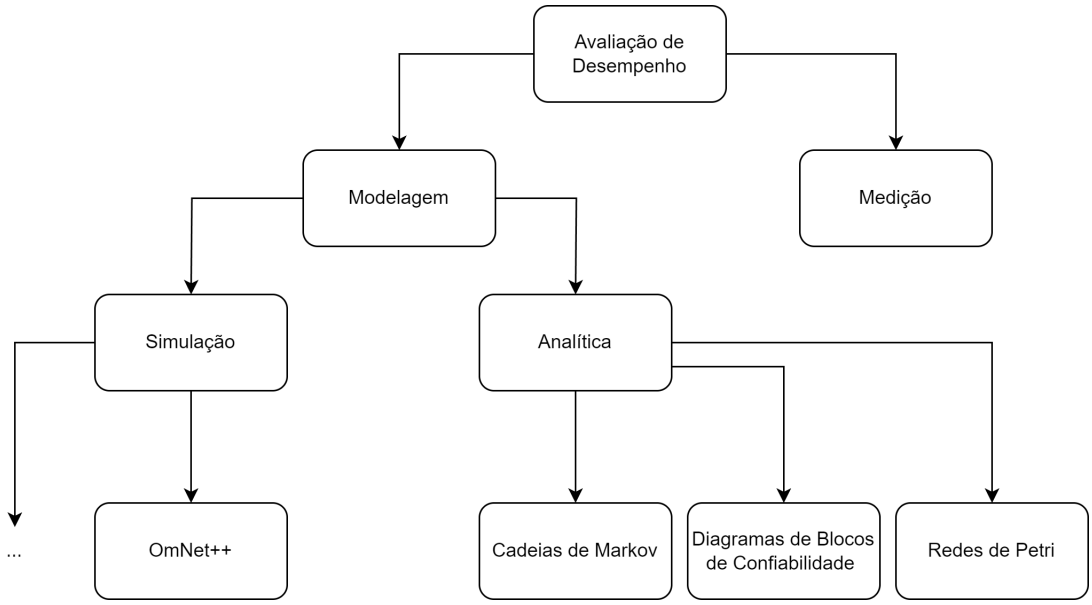
A simulação pode ser empregada tanto para a avaliação de desempenho quanto para a verificação de modelos analíticos. Ao contrário das medições, as simulações são baseadas em modelos abstratos do sistema, o que permite sua aplicação sem a necessidade de implementar completamente o sistema real. Dessa forma, o modelo utilizado no processo de simulação é construído com base nas características fundamentais do sistema e o nível de complexidade e abstração adotado pode variar de um sistema para outro. Durante o processo de simulação, é possível controlar os valores dos parâmetros do sistema, facilitando a obtenção de informações relevantes para a avaliação de desempenho (LILJA, 2005). Inclusive, na simulação é possível verificar o comportamento de um sistema real a partir de experimentos realizados em um modelo computacional. Essa é uma técnica de baixo custo e com ampla aceitação. O OMNET++, GPSS, OPNET são algumas das aplicações que permitem a criação de um ambiente de simulação. Vale ressaltar que a precisão nos resultados é moderada.

A análise de desempenho pode ser realizada de diversas maneiras, como ilustrado na Figura 6 (baseada em CALLOU et al., 2012). Dependendo das restrições de tempo e recursos, a combinação de várias técnicas pode proporcionar uma compreensão mais completa do sistema e fornecer informações mais precisas para auxiliar na tomada de decisões. Por meio de ferramentas de modelagem, simulação e medição, é possível obter informações detalhadas sobre o sistema em análise.

A utilização de técnicas de medição permite obter resultados mais precisos, uma vez que está diretamente relacionada ao ambiente real sendo analisado. No entanto, a modelagem oferece uma variedade de métodos alternativos de avaliação de desempenho, que podem ser obtidos por meio de simulação ou modelagem analítica. Os modelos

analíticos incluem métodos probabilísticos, teoria de filas, modelos de Markov ou redes de Petri (CALLOU et al., 2012). Para as simulações podem ser utilizadas ferramentas como OMNET++, entre outras.

Figura 6 – Técnicas de avaliação de desempenho



Fonte: O autor.

É importante destacar que a modelagem e a simulação permitem a avaliação do sistema em diferentes cenários, mesmo aqueles que ainda não estão disponíveis no sistema real em análise. Isso proporciona uma visão abrangente do desempenho do sistema e auxilia na identificação de possíveis melhorias ou ajustes necessários.

Nesse trabalho são aplicadas técnicas de medição obtendo métricas, por meio do software Prometheus(RABENSTEIN; VOLZ, 2015), referentes a uso de CPU, memória, dados enviados e recebidos nas interfaces de rede, e escrita nos discos. Para a disponibilidade são aplicadas técnicas para calcular o MTTF (*Mean Time To Failure*) e o MTTR (*Mean Time To Repair*) do sistema, e a técnica de modelagem para propor modelos analíticos que possam representar o sistema e suas diferentes configurações.

3.4 Avaliação do consumo de energia

A avaliação do consumo elétrico é um processo crucial para compreender, controlar e aprimorar o uso eficiente da energia elétrica em diversos ambientes. Ela envolve a medição precisa do consumo de eletricidade, a identificação das principais fontes de consumo e a

análise dos padrões de uso ao longo do tempo. Por meio de medições detalhadas e registros históricos, é possível identificar áreas de ineficiência e implementar estratégias para reduzir o consumo, como a adoção de equipamentos mais eficientes.

Nesse trabalho utilizaremos o Watts Up Pro Portable Power Meter para a obtenção de métricas de consumo de energia dos componentes do sistema de videomonitoramento. As medições serão realizadas no sistema a partir de várias cargas de trabalho, obtendo assim informações do consumo elétrico, servindo para avaliar a eficiência energética do sistema. A análise dos dados coletados permitirá a identificação de oportunidades para reduzir o consumo de energia e, consequentemente, os custos associados ao funcionamento do sistema.

3.5 Regressão Linear

A regressão linear é uma técnica estatística utilizada na tentativa de modelar uma equação matemática linear que descreva o relacionamento entre duas variáveis (CURRAL, 1994; MATOS, 1995). A análise de regressão linear é usada para prever o valor de uma variável com base no valor de outra. A variável que deseja prever é chamada de variável dependente. A variável usada para prever o valor de outra variável é chamada de variável independente.

A regressão linear simples é dada pela seguinte Equação 3.1:

$$y = a + bx \tag{3.1}$$

onde y é o resultado da saída prevista, x é a variável de entrada e a e b são os parâmetros de regressão.

Para obter o parâmetro de regressão b é utilizada a Equação 3.2, onde n é a quantidade de pontos $(x_i; y_i)$.

$$b = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n\sum_{i=1}^{n} \mathrm{x}_i^2 - (\sum_{i=1}^{n} x_i)^2} \tag{3.2}$$

O cálculo do parâmetro de regressão a é dada pela Equação 3.3, onde $\overline{y}$ e $\overline{x}$ corresponde a média aritmética da variável de saída e de entrada, respectivamente.

$$a = \overline{y} - b\overline{x} \tag{3.3}$$

Vinculado a regressão linear existe um coeficiente de determinação que expressa um grau de erro, representado por R^2 (LILJA, 2005). Esse coeficiente varia entre 0 e 1. Quanto mais próximo o valor do coeficiente de determinação estiver de 1, maior será a confiabilidade da regressão. O cálculo do coeficiente de determinação é obtido a partir da Equação 3.4.

$$R^2 = \frac{b\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{3.4}$$

A regressão linear pode ser utilizada para estimar ou prever a demanda pela utilização de recursos. Nesse trabalho, a regressão será aplicada para elaborar projeções do consumo elétrico, taxa de transmissão de dados via rede e utilização de memória para cenários com um maior número de câmeras em funcionamento.

3.6 Disponibilidade

A disponibilidade é uma medida fundamental na avaliação da operabilidade de serviços de computação e tem sido objeto de estudo há bastante tempo. Pesquisadores e trabalhos como (AVIZIENIS; LAPRIE, 1986; GRAY; SIEWIOREK, 1991; BAUER, 2011b; BAUER; ADAMS, 2012; MACIEL et al., 2012 e MACIEL, 2023b) têm contribuído significativamente nesse campo.

A disponibilidade refere-se à capacidade de um sistema estar operacional e pronto para fornecer serviços aos usuários. É uma métrica que quantifica o tempo em que um sistema está funcionando corretamente e disponível para uso. Em outras palavras, é a medida da probabilidade de que um serviço esteja disponível e operacional em um determinado momento. A importância da disponibilidade reside no fato de que muitos serviços de computação são essenciais para o funcionamento de empresas, organizações e até mesmo para o dia a dia das pessoas. Falhas ou interrupções em serviços críticos

podem ter consequências significativas, como perda de produtividade, prejuízos financeiros, insatisfação dos clientes e até mesmo riscos à segurança.

As pesquisas focadas em disponibilidade visam desenvolver técnicas e estratégias para aumentar a disponibilidade dos sistemas, minimizando o tempo de inatividade e garantindo a continuidade dos serviços. Isso envolve a identificação e mitigação de potenciais pontos de falha, a implementação de mecanismos de recuperação de falhas, o planejamento adequado da infraestrutura de rede, entre outras medidas. Além disso, a disponibilidade não se limita apenas à detecção e correção de falhas. Ela também considera o tempo necessário para recuperar um sistema após uma falha, ou seja, o tempo de recuperação. A rapidez e eficiência na recuperação são fatores cruciais para minimizar o impacto das falhas e restaurar o serviço o mais rápido possível.

Com o avanço tecnológico, as melhorias nos dispositivos ao longo do tempo têm contribuído para o aumento da disponibilidade dos sistemas computacionais. Em comparação a décadas passadas, onde sistemas bem executados ofereciam cerca de 99% de disponibilidade, houve uma busca por níveis de disponibilidade cada vez mais altos. Alguns anos atrás, uma disponibilidade de 99% parecia satisfatória, mas isso significava aproximadamente 100 minutos de indisponibilidade por semana. Essas interrupções poderiam ser toleráveis para sistemas não críticos. No entanto, para aplicativos de missão crítica e serviços online, esse tempo de indisponibilidade é inaceitável. Esses sistemas exigem uma alta disponibilidade, geralmente referida como "cinco 9's", o que significa que o sistema deve ter 99,999% de disponibilidade, permitindo no máximo cinco minutos de interrupção de serviço por ano.

A disponibilidade é uma métrica adimensional expressa em forma de porcentagem, variando de 0% a 100%, ou por meio de uma escala de 0 a 1. Um sistema que possui uma disponibilidade de 99,999%, ou 0,99999 na escala, é considerado um sistema de cinco 9's, o que representa um nível extremamente alto de confiabilidade e disponibilidade. A Tabela 2, baseada em (BAUER, 2011b; MACIEL, 2023b), apresenta a relação entre a quantidade de 9's e o tempo de indisponibilidade anual.

Tabela 2 – Relação entre quantidade de 9's, disponibilidade e indisponibilidade

Quantidade de 9's	**Disponibilidade**	**Disponibilidade (%)**	**Tempo anual indisponível**
1	0,9	90%	36,5 dias
2	0,99	99%	3,65 dias
3	0,999	99,9%	8,76 horas
4	0,9999	99,99%	52 minutos
5	0,99999	99,999%	5 minutos
6	0,999999	99,9999%	30 segundos
7	0,9999999	99,99999%	3 segundos

Fonte: O autor

O tempo de atividade de um sistema é frequentemente medido pelo Tempo Médio de Falha (MTTF), que representa a quantidade média de tempo que o sistema opera sem falhas. Por outro lado, o tempo de indisponibilidade do sistema é medido pelo Tempo Médio de Reparo (MTTR), que representa o tempo médio necessário para reparar o sistema em caso de falha (WANG; TRIVEDI, 2005; SCHNEEWEISS, 2012; MACIEL et al., 2012). A disponibilidade do sistema pode ser calculada usando a relação entre o MTTF e o MTTR. Essa relação pode ser expressa pela Equação 3.5:

$$A = \frac{MTTF}{MTTR + MTTF} \tag{3.5}$$

Ao usar os valores MTTF e MTTR para cálculos de disponibilidade, o resultado será sempre um número entre 0 e 1. O MTTF de um sistema pode ser calculado pela equação 3.6, onde *R(t)* é a confiabilidade do sistema em função do tempo decorrido.

$$MTTF = \int_0^\infty R(t) \tag{3.6}$$

Com (γ_i) inversamente proporcional ao valor MTTF, temos:

$$\gamma_i = \frac{1}{MTTF} \tag{3.7}$$

Em sistemas que possuem taxas (γ_i) constantes e iguais, que operem em série, o MTTF pode ser calculado a partir da Equação 3.8,

$$MTTF = \frac{1}{\sum_{i=1}^{n} \gamma_i} \tag{3.8}$$

onde n é o número de componentes e γ_i a taxa (MACIEL, 2023a). Em situações em que todos os γ_i são iguais, pode-se chamar de γ, então temos:

$$MTTF = \frac{1}{n \times \gamma} \tag{3.9}$$

Os sistemas que atuam de forma paralela e possuam taxas de falhas (γ) constantes, temos a Equação 3.10.

$$MTTF = \frac{1}{\gamma} \times \sum_{i=1}^{n} \times \frac{1}{i} \tag{3.10}$$

A disponibilidade pode ser computada pela Equação 3.11, para sistemas com configuração *K-out-of-N*, que detenham taxas de falha (γ) iguais e constantes.

$$MTTF = \frac{1}{\gamma} \times \sum_{i=n}^{n} \times \frac{1}{i} \tag{3.11}$$

onde n que representa a quantidade de componentes do sistema e k se refere ao número de componentes que devem estar em estado ativo para que o sistema seja considerado disponível, ou seja, indica que o sistema está em pleno e adequado funcionamento.

As Equações 3.12 e 3.13 são utilizadas para obter o valor estimado do MTTF e do MTTR, respectivamente. Nessas equações aplicam-se a razão entre o somatório dos tempos de falha (TTF) e a quantidade de falhas (n) para a Equação 3.12 e para a Equação 3.13 aplica-se a relação entre o somatório dos tempos de reparo e a quantidade de reparos realizados (MACIEL, 2023a).

$$\widehat{MTTF} = \frac{\sum_{i-1}^{n} ttf_i}{n} \tag{3.12}$$

$$\widehat{MTTR} = \frac{\sum_{i-1}^{n} ttr_i}{n} \tag{3.13}$$

3.6.1 Estratégias de Redundância

A redundância se refere à duplicação ou à presença de componentes, ou recursos adicionais em um sistema, projetados para entrar em operação caso um componente principal falhe. Esta técnica desempenha um papel crucial na garantia de que sistemas essenciais possam manter seu funcionamento, mesmo diante de falhas inesperadas.

Atualmente, há uma ampla gama de técnicas disponíveis para melhorar a disponibilidade de ambientes computacionais, visando garantir a continuidade dos serviços e minimizar interrupções indesejadas. Dentre essas técnicas, destaca-se a criação de clusters e a utilização da virtualização (MACIEL, 2023a; HAIM; ROTTENSTREICH, 2020; WAGDY; BABULAK; AL-DABASS, 2021; RIABKO et al., 2023), que têm se mostrado eficazes na busca por maior confiabilidade e desempenho.

A criação de *clusters* é uma estratégia comum na construção de ambientes altamente disponíveis. Consiste em agrupar um conjunto de servidores, também conhecidos como nós, de forma a compartilhar a carga de trabalho e prover redundância. Com isso, caso um nó falhe, outros estão prontos para assumir as tarefas e garantir que os serviços continuem funcionando sem interrupções significativas. Além disso, a utilização de *clusters* permite a escalabilidade dos recursos, facilitando a distribuição do processamento e o balanceamento de carga conforme a demanda.

Outra técnica amplamente adotada é a virtualização, que proporciona maior flexibilidade e eficiência na alocação de recursos. Por meio da virtualização, é possível criar múltiplas máquinas virtuais em um único hardware físico, permitindo a execução de diversos sistemas operacionais e aplicativos independentes. Dessa forma, em caso de falha em uma máquina virtual, as demais continuam operacionais, garantindo a disponibilidade dos serviços. Além disso, a virtualização possibilita o gerenciamento centralizado dos recursos, facilitando a manutenção, o monitoramento e a recuperação em situações de falha.

As técnicas mencionadas, anteriormente, baseiam-se no princípio da redundância, ou seja, na replicação de componentes essenciais do sistema. Essa abordagem consiste em ter cópias de elementos críticos, como servidores, armazenamento e rede, de modo que, se um componente falhar, outros estejam prontos para assumir suas funções imediatamente. Isso garante que os serviços continuem disponíveis, minimizando o impacto de falhas e defeitos.

Existem dois tipos principais de mecanismos de redundância em termos de componentes: ativo/ativo e ativo/passivo. No caso do mecanismo de redundância ativo/ativo, dois componentes - primário e secundário - trabalham em conjunto para realizar o balanceamento de carga. Ambos os componentes estão ativos e recebem solicitações de acesso. Se um dos componentes apresentar defeito, o outro assume todas as responsabilidades do serviço, garantindo a continuidade das operações (BAUER, 2011a).

Por outro lado, o mecanismo de redundância ativo/passivo também envolve os mesmos componentes - primário e secundário -, mas apenas o componente primário é responsável por fornecer o serviço, enquanto o componente secundário permanece em estado de espera, pronto para entrar em ação quando necessário. Caso o componente primário apresente falhas ou defeitos, o componente secundário assume o papel de fornecer o serviço, garantindo a continuidade do mesmo (BAUER, 2011a).

O mecanismo de redundância ativo/passivo é composto por três técnicas amplamente utilizadas em infraestruturas em nuvem: *cold standby*, *hot standby* e *warm standby* (MACIEL; DANTAS; JÚNIOR, 2019; BAUER, 2011a; MACIEL et al., 2012; MELO et al., 2018;). Na técnica *cold standby*, o componente secundário permanece desligado em *standby*, sendo acionado somente se o componente principal apresentar defeito. Uma vantagem dessa abordagem é que o nó secundário não sofre desgaste dos componentes durante a inatividade. No entanto, essa técnica requer um tempo significativo para que o nó secundário alcance o estado funcional, o que pode resultar em um aumento da indisponibilidade do sistema.

A técnica *hot standby* é caracterizada pela ativação imediata do componente secundário, que está sempre pronto para assumir o serviço assim que necessário. Nesse caso, ambos os componentes estão operacionais e prontos para atender às solicitações. Essa abordagem oferece uma rápida recuperação em caso de falha, minimizando o tempo de indisponibilidade. No entanto, é importante considerar os recursos extras necessários para manter o componente secundário em estado ativo o tempo todo. Por fim, a técnica *warm standby* que combina elementos das técnicas *cold* e *hot standby*. Nesse cenário, o componente secundário fica em um estado parcialmente ativo, com recursos pré-carregados e pronto para ser acionado rapidamente. Embora possa levar um pouco mais de tempo para que o nó secundário esteja completamente funcional, essa técnica oferece um equilíbrio entre disponibilidade imediata e economia de recursos. Cada uma dessas técnicas possui

suas características específicas e a escolha adequada depende das necessidades do sistema, dos requisitos de disponibilidade e dos recursos disponíveis.

3.7 Estratégias de Modelagem

Esta seção apresenta duas estratégias de modelagem utilizadas para avaliar sistemas. Inicialmente, são apresentados os conceitos básicos do modelo de diagrama de blocos de confiabilidade (RBD, do inglês *Reliability Block Diagram*), seguidos dos fundamentos de redes de Petri estocásticas (SPN, do inglês *Stochastic Petri Nets*). Nesta pesquisa, a modelagem RBD é empregada para obter os tempos médios de falha e os tempos médios de reparo, além de ser utilizada para analisar o índice de importância para a disponibilidade (*Availability Importance*) nos cenários com redundância *hot standby*. Por outro lado, os modelos SPN são aplicados para avaliar a disponibilidade em ambientes com redundância *cold standby*.

3.7.1 Reliability Block Diagram - RBD

O *Reliability Block Diagram* ou diagrama de blocos de confiabilidade é um modelo orientado que vincula blocos funcionais aos efeitos das falhas dos componentes na confiabilidade do sistema. Os blocos que representam os componentes podem ser combinados, em série, paralelo ou em configurações *K-out-of-n* (MACIEL, 2023b; TRIVEDI et al., 1996)

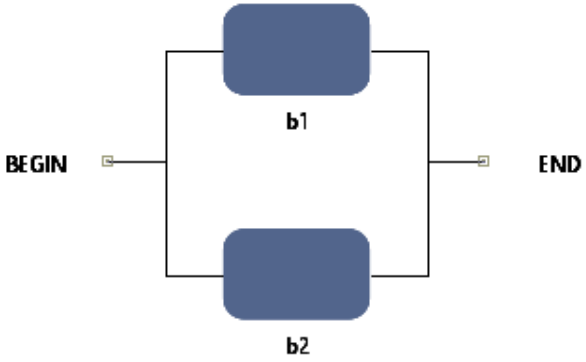
Os RBDs são compostos por dois nós especiais, chamados de vértice de origem e vértice de destino, blocos (geralmente retângulos) e arcos que conectam os blocos e os vértices. O nó de origem é geralmente colocado no lado esquerdo do diagrama, onde o vértice de destino é posicionado à direita. Dessa forma, os RBDs permitem a modelagem de sistemas, fornecendo uma representação visual dos componentes dos mesmos. A modelagem RBD estabelece uma relação lógica entre os componentes, definindo quais combinações são necessárias para garantir a funcionalidade do sistema. Além disso, por meio dessa técnica, é possível calcular métricas de confiabilidade, como a disponibilidade, por exemplo.

A Figura 7 mostra os blocos organizados em série e a Figura 8 apresenta um modelo RBD paralelo. No modelo em série, se um único componente falhar, o sistema interrompe o serviço disponibilizado, entrando num modo de falha.

Figura 7 – RBD em série

Fonte: O autor.

Figura 8 – RBD paralelo



Fonte: O autor.

A disponibilidade de um sistema de dois blocos em série (Figura 7) pode ser calculada a partir da Equação 3.14.

$$A_s(t) = A_{b1}(t) \times A_{b2}(t) \tag{3.14}$$

onde, A_{b1} é a disponibilidade do bloco $b1$ e A_{b2} é a disponibilidade do bloco $b2$.

Considerando um sistema com n elementos em série, a disponibilidade pode ser obtida com a Equação 3.15.

$$A_s(t) = \prod_{i=1}^{n} A_i(t) \tag{3.15}$$

Se os TTFs (*Time to Failure*) e os TTRs (*Time to Repair*) de todos os componentes forem distribuídos de forma idêntica, a disponibilidade de estado estacionário do sistema é obtida por (MACIEL, 2023b):

$$A_s = \left(\frac{MTTF}{MTTF + MTTR} \right)^n \tag{3.16}$$

A disponibilidade de um sistema RBD em paralelo (Figura 8) pode ser obtida com a Equação 3.17.

$$A_p(t) = 1 - \prod_{i=1}^{2} (1 - A(t)) \tag{3.17}$$

Os blocos RBDs podem ser representados na forma *k-out-of-n*, que representam estruturas em que o subsistema pode funcionar se k componentes estiverem em operação (MACIEL, 2023b). Por exemplo, considere uma infraestrutura com oito componentes, onde é necessário que pelo menos três estejam ativos para fornecer o serviço esperado. Nesse caso, temos uma estrutura *3-out-of-8* (ou 3 de 8). As estruturas em série-paralelo são casos especiais das estruturas *k-out-of-n*, ou seja, uma estrutura em série é uma *n-out-of-n*, enquanto uma estrutura em paralelo é uma *1-out-of-n* (SAHNER; TRIVEDI, 1987).

Para a definição matemática da disponibilidade dessa representação, é necessário estabelecer a variável aleatória discreta X, que representa o número de blocos que não apresentam falhas em um determinado intervalo de tempo, conforme Equação 3.18. Os eventos probabilísticos de disponibilidade são independentes para cada bloco na configuração *k-out-of-n*, e todos os n blocos possuem a mesma taxa de falha (SAHNER; TRIVEDI, 1987; MACIEL, 2023b).

$$A_{k-out-of-n}(t) = \sum_{i=k}^{n} P(X = i) \tag{3.18}$$

Os modelos RBD são amplamente utilizados em sistemas que possuem módulos independentes, em que cada um pode ser representado de forma individual por um bloco de disponibilidade. Essa abordagem facilita a análise e o entendimento da disponibilidade do sistema como um todo. No entanto, quando se trata de sistemas complexos que requerem

a adição de redundância em seus módulos, é necessário recorrer a técnicas de modelagem hierárquica.

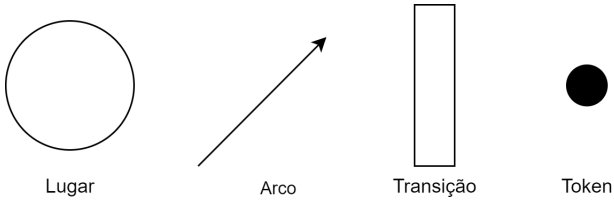
A modelagem hierárquica é uma técnica que permite a criação de representações mais abrangentes e detalhadas de sistemas complexos. Nesse contexto, é comum combinar modelos RBD com outros modelos, como SPN (redes de Petri estocásticas), CTMC (cadeias de Markov de tempo contínuo) e outros, para obter resultados mais expressivos. No entanto, um modelo RBD é especialmente útil para sistemas em que a disponibilidade depende da interação entre componentes independentes.

O RBD permite visualizar facilmente a estrutura do sistema e identificar caminhos críticos de falha, tornando-o eficaz para avaliar a disponibilidade do sistema. Além disso, o RBD é mais adequado para modelar sistemas com ênfase na falha e na recuperação, onde a análise da confiabilidade é a principal preocupação.

3.7.2 Stochastic Petri Nets - SPN

Em 1962, Carl Adam Petri introduziu o conceito de redes de Petri (*PN - Petri Net*) como uma técnica gráfica e formal para representar e simular o comportamento de sistemas (PETRI, 1962). Desde então, essa abordagem tem sido amplamente utilizada na modelagem e análise de sistemas em diversas áreas do conhecimento. As redes de Petri são especialmente adequadas para descrever e analisar sistemas de processamento de informações caracterizados como concorrentes, assíncronos, distribuídos, paralelos, não determinísticos e/ou estocásticos. Por meio do uso de *tokens*, essas redes permitem simular a dinâmica e as atividades concorrentes do sistema (MURATA, 1989). As redes de Petri básicas são compostas por lugares, arcos, transições e *tokens* (Figura 9). Lugares descrevem os estados do sistema, enquanto as transições representam as ações necessárias para modificar esses estados. A habilitação de uma transição requer a satisfação de pré-condições específicas. Arcos servem para indicar o fluxo de tokens na rede, e os tokens representam o estado atual do sistema.

Figura 9 – Elementos de uma *Petri Net*

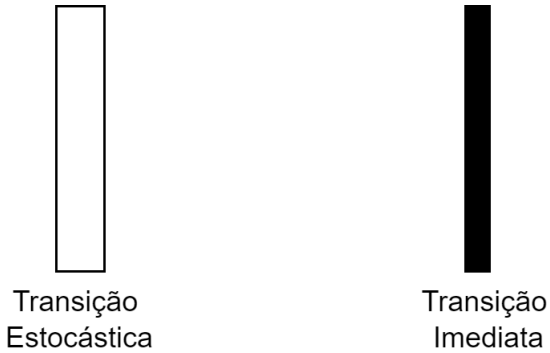


Fonte: O autor.

Inicialmente, as redes de Petri não levavam em consideração o tempo em sua modelagem. No entanto, o tempo é necessário para a realização de análises de desempenho e dependabilidade. Neste trabalho, é utilizada uma extensão conhecida como redes de Petri estocásticas, essa extensão possibilita associar tempos às transições do modelo, permitindo uma abordagem mais detalhada e realista das dinâmicas do sistema em estudo (MACIEL, 2023a).

As redes de Petri estocásticas podem ser usadas para modelar o desempenho e a confiabilidade de sistemas complexos. As SPNs têm duas representações distintas para as transições: transições estocásticas e transições imediatas (Figura 10). As transições estocásticas são associadas a um tempo que representa o período de execução da atividade. As transições imediatas não demandam tempo para disparar, pois disparam instantaneamente.

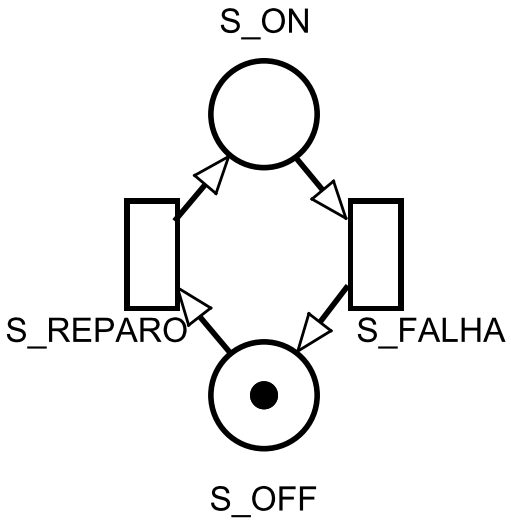
Figura 10 – Transições em SPN



Fonte: O autor.

A Figura 11 apresenta a modelagem SPN. Capaz de representar qualquer sistema ou componente com dois estados possíveis: operacional ou em falha. Utilizando esse modelo, é possível calcular a disponibilidade dos componentes ou sistemas desejados.

Figura 11 – Exemplo SPN



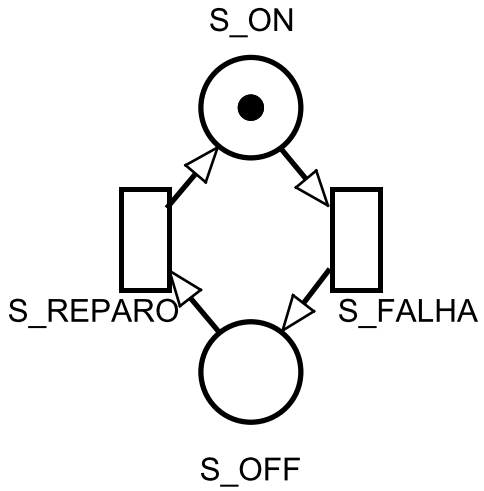
Fonte: O autor.

No modelo SPN apresentado na Figura 11, S identifica o subsistema modelado. Exemplificando, existem lugares identificados como S_ON e S_OFF, que representam, respectivamente, o estado ativo (ON) e de falha (OFF). O que define o estado do modelo é a localização do *token*. As transições temporizadas identificadas no modelo como S_FALHA e S_REPARO representam as ações que levam a alteração de estado do sistema. Elas possuem associados os valores do MTTF e MTTR, respectivamente. Essa representação permite realizar cálculos para identificar a disponibilidade do sistema, considerando a Equação 3.19, onde termo A_s representa a disponibilidade do sistema. O termo P representa a probabilidade de ocorrência. O termo # representa a quantidade de *tokens* em um determinado lugar. A representação S_ON informa qual lugar está sendo referenciado e o termo (>0) representa a condição da equação. Assim a disponibilidade do nosso modelo pode ser obtida, identificando a probabilidade de existir mais de um *token* no estado/lugar S_ON.

$$A_s = P\{\#S\_ON > 0\} \tag{3.19}$$

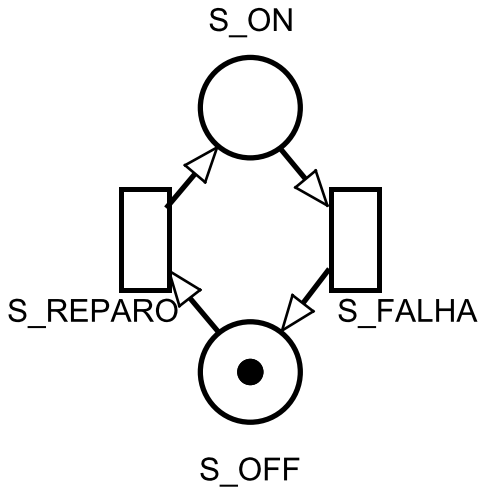
O comportamento do modelo é apresentado nas Figuras 12 e 13. A Figura 12 apresenta um modelo com o sistema ativo, pois possui um *token* no lugar S_ON. A única transição ativa é S_FALHA pois está conectada ao único lugar possível de retirada de *tokens*. Quando a transição S_FALHA é disparada, o *token* é consumido do lugar S_ON e um *token* é gerado no lugar S_OFF, indicando que o sistema está indisponível. A transição temporizada S_FALHA possui um valor referente ao MTTF do sistema ou componente representado.

Figura 12 – SPN - Sistema Ativo



Fonte: O autor.

Figura 13 – SPN - Sistema Indisponível



Fonte: O autor.

Na Figura 13 o sistema está inoperante, pois existe um *token* no lugar S_OFF. A transição ativa é S_REPARO por estar conectada ao único lugar possível de retirada de *tokens*. Quando a transição S_REPARO é acionada, o *token* é levado do lugar S_OFF para o lugar S_ON. Com isso, o modelo retorna ao estado ativo. O valor da transição S_REPARO se refere ao MTTR do componente ou sistema.

3.8 Availability Importance

A *Availability Importance*, ou índice de importância para a disponibilidade é uma métrica que indica o impacto de um componente específico na disponibilidade geral do sistema. Essa técnica será utilizada para identificar os componentes mais críticos para a disponibilidade do sistema.

A *AI* (ou *Availability Importance*) de um componente i corresponde à melhoria na disponibilidade do sistema quando a disponibilidade do componente i é aumentada em uma unidade (KUO; ZUO, 2003). A importância para a disponibilidade do componente i pode ser calculada com a Equação 3.20,

$$AI_i = A_s(1_i, p^i) - A_s(0_i, p^i) \tag{3.20}$$

onde, AI_i é a importância do componente i, A_s é a disponibilidade do sistema, p^i representa o vetor de disponibilidade do componente com i -ésimo componente removido, 0_i representa a falha do componente i e 1_i denota o componente i funcionando adequadamente.

Os resultados podem ser normalizados conforme Equação 3.21,

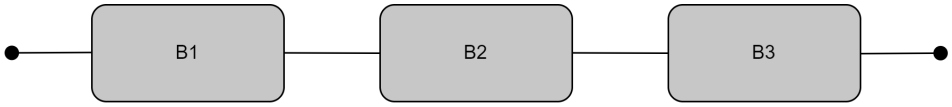
$$I_{ni} = \frac{I_i}{I_x} \tag{3.21}$$

onde I_{ni} é o índice normalizado para o componente i; I_i é o valor do índice não normalizado para o componente i e, I_x é o valor do maior índice não normalizado entre os componentes.

Para exemplificar temos a Figura 14, onde se deseja saber qual dos componentes é mais sensível para a disponibilidade do sistema. Os valores obtidos estão na Tabela 3, que apresenta os valores para disponibilidade e os valores da importância para a disponibilidade, permite identificar que o componente B3 é o mais importante. A importância do componente *B1* é calculado com a Equação 3.22.

$$AI_i = 1_1 \times p_2 \times p_3 - 0_1 \times p_2 \times p_3 \tag{3.22}$$

Figura 14 – RBD com 3 componentes em série



Fonte: O autor.

É importante ressaltar que em uma estrutura em série, o componente menos confiável tem o maior *AI*. Em uma estrutura paralela, o componente mais confiável tem o maior *AI* (KUO; ZUO, 2003).

Tabela 3 – Importância da disponibilidade

Métrica	**Componente B1**	**Componente B2**	**Componente B3**
Disponibilidade	0,90	0,85	0,80
Availability Importance	0,68	0,72	**0,76**

Fonte: O autor.

3.9 Injeção de falhas

Ao avaliar a confiabilidade e a robustez de sistemas críticos, é fundamental compreender os dois principais métodos de injeção de falhas: por hardware e por software. Essas abordagens desempenham um papel vital no teste e na validação de sistemas para garantir que possam resistir a condições adversas e continuar operando de maneira segura e eficaz. Vamos explorar cada um desses métodos em detalhes a seguir.

A injeção de falhas baseada em hardware envolve o uso de hardware adicional para introduzir falhas em um sistema-alvo. Existem dois tipos principais de falhas de hardware: com contato e sem contato (HSUEH; TSAI; IYER, 1997; WANG et al., 2019). No caso da injeção de falhas com contato, o injetor tem um contato físico direto com o sistema-alvo e pode gerar variações de corrente ou tensão no sistema sendo testado. Já na injeção de falhas sem contato, o injetor gera algum evento externo que leva à falha do componente desejado.

Por outro lado, a injeção de falhas baseada em software envolve a modificação do estado de funcionamento do software em execução no sistema em teste. Essa técnica é dividida em duas categorias: tempo de compilação e tempo de execução (HSUEH; TSAI; IYER, 1997; ZIADE et al., 2004).

Injetores de falhas em tempo de compilação fazem alterações no código-fonte do sistema original para emular o efeito de falhas de hardware e software no sistema. Já os injetores de falhas em tempo de execução não exigem a modificação do código-fonte do sistema. Em vez disso, utilizam algum mecanismo que dispara a injeção de falhas. Existem três técnicas para implementar injetores de falha em tempo de execução: *Time-out* (temporizador), *Exception/trap* (exceção/armadilha) e *Code insertion* (inserção de código).

As técnicas de injeção de falhas têm sido amplamente aceitas há bastante tempo para a validação da confiabilidade de sistemas, por meio da análise do comportamento dos dispositivos quando ocorre uma falha (ZIADE et al., 2004). A escolha entre os métodos

de injeção de falhas depende do tipo de falhas que se deseja injetar e do tempo necessário para criar essas falhas. Nesta pesquisa, foram utilizados métodos de injeção de falhas de software com a técnica *Time-out*, que consiste na injeção de falhas após um tempo determinado.

4 Metodologia

Nesse capítulo, é apresentada a metodologia utilizada para a avaliação de desempenho e consumo de energia do sistema de videomonitoramento, bem como as atividades realizadas para a avaliação da disponibilidade. A metodologia desempenha um papel fundamental em qualquer estudo científico, pois é por meio dela que são estabelecidos os caminhos e procedimentos para alcançar os objetivos da pesquisa. Também são contextualizadas as atividades realizadas e os objetos gerados em cada uma das atividades dessa metodologia. Este capítulo, está dividido nas seguintes seções: visão geral, avaliação de desempenho e consumo de elétrico, e avaliação da disponibilidade.

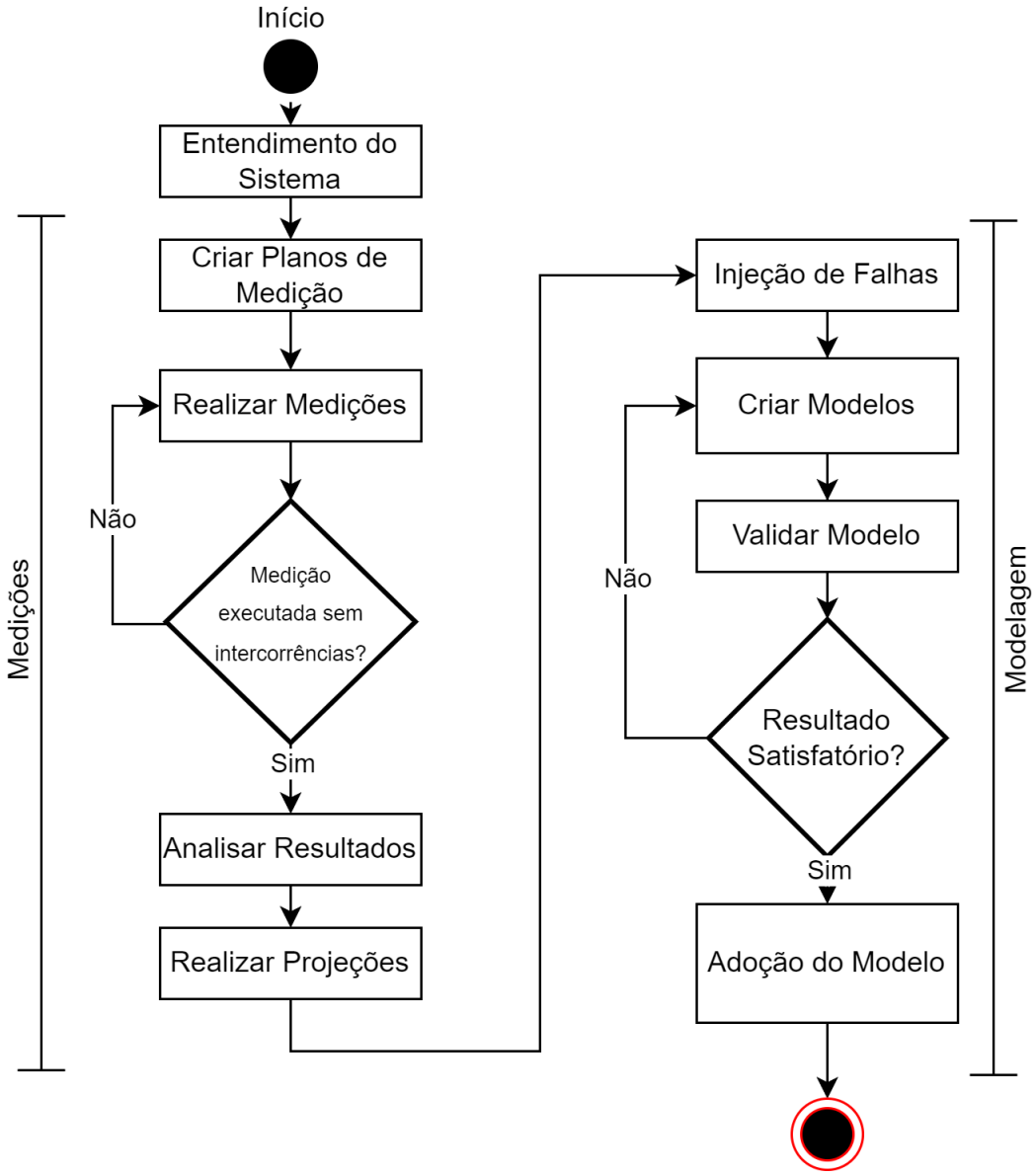
4.1 Visão Geral

A metodologia está dividida em três etapas: (i) entendimento do sistema, (ii) medições e (iii) modelagem. A Figura 15 apresenta a metodologia proposta.

A primeira etapa corresponde ao entendimento do sistema, onde foi feita uma revisão da literatura e foi estudado o sistema a fim de compreender a infraestrutura de armazenamento distribuído, analisando seus componentes e requisitos de funcionamento. Ainda nessa atividade foi estudada uma solução de gerenciamento de video identificando os métodos de funcionamento do VMS. Ao final, um ambiente foi configurado para a realização dos experimentos.

A etapa seguinte corresponde às atividades de medições no ambiente em análise. Sendo assim, nessa etapa se define um plano de medição para realizar os experimentos no ambiente montado em laboratório. Nessa etapa, são planejadas as atividades de medição para a execução dos experimentos, dimensionando as cargas de trabalho que serão utilizadas, por exemplo. Em seguida, as medições foram realizadas seguindo o plano de medição criado anteriormente. Para obter os dados referentes ao uso dos recursos computacionais foram utilizados os softwares: *Prometheus* e *Grafana*. O *Prometheus* é um software de monitoramento e alerta robusto e altamente configurável, com capacidade de coletar, armazenar, consultar e visualizar métricas de sistemas de forma eficiente e confiável. O *Grafana* é uma plataforma de visualização de métricas e análise de dados, onde é possível criar dashboards personalizados e painéis interativos, fornecendo uma visão

Figura 15 – Metodologia



Fonte: O autor.

abrangente do ambiente monitorado. Para a obtenção dos dados referentes ao consumo elétrico foi utilizado o *Watts Up Pro Portable Power Meter*, equipamento que fornece informações precisas sobre o consumo de energia em tempo real, permitindo que seja monitorado e avaliado o uso de energia.

Após as medições, os dados são analisados e realizam-se os cálculos estatísticos da média de consumo, conforme as diferentes cargas de trabalho. Importante destacar que essa atividade também envolve a geração de gráficos de comparação. Esses gráficos permitem visualizar e comparar diferentes conjuntos de dados coletados em diferentes situações ou períodos. Por exemplo, é possível criar gráficos que mostrem a evolução do

consumo de recursos ao longo de uma semana, mês ou trimestre. Essa visualização facilita a identificação de tendências, padrões sazonais ou anomalias que possam exigir intervenção ou ajustes no ambiente.

Em seguida, projeções são realizadas utilizando a técnica estatística de regressão linear para elaborar projeções de consumo de energia e outras métricas em diferentes cenários. A regressão linear estabelece uma relação entre a variável dependente (consumo elétrico e taxa de transmissão de dados) e as variáveis independentes (número de câmeras, forma de armazenamento, etc.). Com base em dados históricos de consumo e informações sobre as câmeras e formas de armazenamento, é construído um modelo de regressão para realizar as projeções de consumo em diversas configurações. Essas projeções são baseadas em conjuntos de dados históricos, incluindo informações sobre o número de câmeras em operação e o tipo de armazenamento utilizado.

A etapa seguinte corresponde as atividades de modelagem. Nessa etapa, falhas são injetadas no sistema construído para os experimentos. Ao monitorar o funcionamento do sistema real, podemos obter os valores de entrada utilizados nos modelos propostos. Nessa atividade, o sistema é monitorado para capturar o tempo de falha (TTF) e o tempo de reparo (TTR). Assim, a disponibilidade do sistema real é calculada.

Em seguida, são criados modelos de dependabilidade para representar o ambiente e a partir dele diversos cenários são avaliados, visando identificar quais configurações melhoram disponibilidade e desempenho. Na atividade de validação do modelo, são comparados os valores de saída dos modelos analíticos com o ambiente monitorado; ou seja, verifica-se estatisticamente se os modelos propostos representam o sistema real ou não.

Por fim, a partir da confirmação que os valores dos modelos são satisfatórios, podemos utilizá-los para avaliar a disponibilidade e o desempenho do sistema em diferentes configurações e cenários e, assim, planejar novos ambientes.

4.1.1 Entendimento do sistema

Para planejar infraestruturas de videomonitoramento é necessário entender como elas funcionam, identificando seus principais componentes e avaliar as soluções existentes. Essa etapa requer atenção, pois erros podem comprometer as próximas etapas. Essa

atividade é essencial porque permite aprender técnicas que podem ser adotadas e adaptadas.

- Entrada: leitura de artigos e documentações sobre as soluções de videomonitoramento e armazenamento;
- Ação: desenvolver um ambiente para realização de experimentos;
- Saída: Compreensão de funcionamento do sistema e ambiente de experimentos instalado e configurado.

4.2 Medições

A etapa de medições é composta por quatro atividades: criar plano de medição, realizar medições, analisar resultados e realizar projeções. O principal objetivo dessa etapa é monitorar o ambiente, obtendo resultados da utilização de recursos computacionais e de consumo elétrico para elaborar projeções para diferentes configurações e demandas.

4.2.1 Criar plano de medição

A elaboração do plano de medição desempenha um papel fundamental na condução de experimentos, pois fornece um guia estruturado para a realização dos testes. Esse plano define as cargas de trabalho que serão aplicadas, as métricas a serem obtidas e as diferentes configurações dos componentes do sistema que serão investigadas. Ao elaborar um plano de medição abrangente e bem definido, é possível obter resultados consistentes e confiáveis dos experimentos. Isso ajuda a compreender melhor o comportamento do sistema em diferentes condições e a tomar decisões para aprimorar o desempenho, a eficiência e a escalabilidade do ambiente de videomonitoramento.

- Entrada: métricas relevantes e modos de configuração do sistema;
- Ação: elaborar roteiro para os experimentos;
- Saída: plano estruturado com as métricas, cargas de trabalho e modos de configuração a serem avaliados.

4.2.2 Realizar medições

A partir do plano de medição elaborado, os experimentos são executados para testar o sistema sob diferentes cargas de trabalho. Esses experimentos visam avaliar o desempenho, a eficiência e a capacidade do sistema em lidar com diferentes demandas e condições operacionais. Durante a execução dos experimentos, as cargas de trabalho definidas no plano de medição são aplicadas ao sistema e várias métricas são coletadas e registradas conforme o plano de medição. Essas métricas podem incluir informações sobre a utilização de CPU, uso de memória RAM, taxa de transmissão de dados, taxa de escrita no disco e consumo elétrico.

- Entrada: informações do plano de medição;
- Ação: monitorar o sistema utilizando o *Prometheus*, *Grafana* e *Watts Up Pro Portable Power Meter*;
- Saída: dados coletados do sistema avaliado.

4.2.3 Analisar resultados

Nesta etapa, são realizados cálculos estatísticos da média de consumo, conforme as diferentes cargas de trabalho, além de atividades que envolve a geração de gráficos de comparação. Esses gráficos permitem visualizar e comparar diferentes conjuntos de dados coletados em diferentes situações ou períodos. Por exemplo, é possível criar gráficos que mostrem a evolução do consumo de recursos ao longo de uma semana, mês ou trimestre. Essa visualização facilita a identificação de tendências, padrões sazonais ou anomalias que possam exigir intervenção ou ajustes no ambiente.

- Entrada: dados coletados;
- Ação: aplicar técnicas e inferências estatísticas calculando médias e gerando gráficos comparativos;
- Saída: gráficos e dados consolidados.

4.2.4 Realizar projeções

Nessa fase, serão elaboradas projeções que visam estimar o consumo de energia e outras métricas em diferentes cenários, levando em consideração variações no número de câmeras e nas formas de armazenamento, como distribuído e replicado. Para realizar essas projeções, uma técnica comumente utilizada é a Regressão Linear. A Regressão Linear é uma técnica estatística que busca estabelecer uma relação linear entre uma variável dependente (ex.: consumo elétrico e taxa de transmissão de dados) e uma ou mais variáveis independentes (ex.: número de câmeras, forma de armazenamento, entre outras). Com base nos dados históricos de consumo e nas informações sobre as câmeras e formas de armazenamento utilizadas, é possível construir um modelo de regressão que permita realizar projeções de consumo para diferentes configurações. Para aplicar a Regressão Linear, é necessário ter um conjunto de dados históricos, juntamente com as variáveis independentes relevantes. Nesse caso, são coletados dados de consumo de energia em diferentes momentos, registrando informações sobre o número de câmeras em operação e o tipo de armazenamento utilizado (distribuído ou replicado).

- Entrada: dados consolidados;
- Ação: aplicação da técnica estatística de Regressão Linear;
- Saída: equações que permitem realizar projeções considerando mais câmeras e diferentes formas de armazenamento.

4.3 Modelagem

A etapa de modelagem é formada pelas seguintes atividades: injeção de falhas, criar modelos, validar modelo e adotar modelo. O objetivo dessa etapa é avaliar a disponibilidade, possibilitando a elaboração de modelos analíticos que após processo de validação podem ser utilizados para dimensionar e fornecer informações precisas para o planejamento de sistemas de videomonitoramento.

4.3.1 Injeção de falhas

A disponibilidade do sistema é um aspecto crítico, uma vez que as falhas podem ser imprevisíveis ou levar muito tempo para ocorrer (JAMMAL et al., 2018 PEREIRA et al., 2021). Quando uma falha ocorre, ela provoca alterações no sistema, levando o ambiente a um estado de falha. Além disso, a falha pode se propagar pelo sistema e afetar a disponibilidade do serviço fornecido. Portanto, desenvolvemos um injetor de falhas para acelerar, controlar e monitorar o ambiente durante eventos de falhas (AGARWAL et al., 2020.

O injetor de falhas utilizado é capaz de simular falhas, aplicando estresse ao sistema. Para isso, é utilizada uma distribuição exponencial para gerar tempos até a falha (TTF) e tempos de recuperação total (TTR) aleatórios, levando em consideração uma aceleração no tempo médio até a falha (MTTF) e no tempo médio para recuperação (MTTR) com base em artigos científicos e relatórios técnicos.

- Entrada: MTTF e MTTR dos componentes do sistema;
- Ação: desenvolver um injetor de falhas e monitorar o ambiente enquanto o injetor de falhas está em execução;
- Saída: dados coletados considerando o sistema de base.

4.3.2 Criar Modelos

Após a realização do experimento e coleta dos dados do ambiente de base, são propostos modelos. Esses modelos desempenham um papel crucial, pois permitem avaliar conceitos de sistemas que envolvem diversos componentes, sem a necessidade de construir esses sistemas, o que seria extremamente custoso. Em vez disso, os modelos são utilizados para examinar diferentes cenários, realizando alterações apenas nos parâmetros de entrada.

Os modelos são especialmente valiosos devido à sua escalabilidade. Eles permitem ajustar o número de componentes em um cenário simplesmente alterando os parâmetros de entrada. Com isso, é possível observar como um sistema se comportaria em diferentes condições e configurações.

É importante ressaltar que os modelos geralmente abstraem os detalhes específicos de hardware e aplicação, buscando encontrar equações em forma fechada (SIVASUBRAMANIAM; RAMACHANDRAN; VENKATESWARAN, 1994). Em outras

palavras, para serem viáveis, esses modelos precisam simplificar alguns componentes do sistema (JAIN, 1990). Essa abstração é necessária para facilitar a análise e permitir o estudo de sistemas complexos de forma mais eficiente.

Os modelos fornecem uma base teórica para a compreensão e previsão do comportamento dos sistemas, permitindo avaliar seu desempenho, capacidade e outros aspectos relevantes. Embora simplifiquem certos aspectos do sistema, eles ainda são valiosos para obter percepções sobre seu funcionamento e tomar decisões informadas em relação ao projeto, dimensionamento e otimização dos sistemas.

- Entrada: compreensão dos componentes essenciais do sistema;
- Ação: criação de modelo com base nos componentes do sistema;
- Saída: modelos SPN e RBD.

4.3.3 Validar modelo

Nessa atividade, é realizada a validação do modelo proposto. Os dados obtidos com o modelo são comparados estatisticamente como os obtidos no monitoramento do sistema real. Se os resultados forem equivalentes, assumimos que os modelos são válidos. Na validação há interesse na média do modelo e na média do sistema real.

Como os dados são apenas uma amostra de uma população, é preciso definir intervalos de confiança para mostrar o intervalo provável dos valores médios da população. Assim, exploramos os dados e calcula-se um intervalo de confiança da média. O método de obtenção do intervalo de confiança é apresentado no capítulo 6.

- Entrada: modelo analítico;
- Ação: análises estatísticas;
- Saída: modelos validados.

4.3.4 Adotar modelo

Após a validação do modelo proposto, é possível adotar esses modelos para avaliar diversas composições do sistema. Isso nos permite investigar e prever como um sistema se comportaria quando seus componentes possuem parâmetros específicos. Além disso, é possível avaliar o número de componentes redundantes necessários para manter um

determinado nível de disponibilidade e assim por diante.

Ao adotar esses modelos validados, ganhamos a capacidade de realizar análises detalhadas e simulações em diferentes cenários. Podemos ajustar os parâmetros dos componentes, como taxa de falhas e tempos de recuperação, para avaliar seu impacto na disponibilidade.

Por exemplo, é possível investigar como uma alteração de um componente afeta a disponibilidade geral do sistema. Também é possível analisar a adição de componentes redundantes e determinar quantos são necessários para manter um determinado nível de disponibilidade, garantindo que o sistema possa lidar com falhas sem interrupção significativa dos serviços.

- Entrada: parâmetros dos componentes do sistema;
- Ação: parametrização dos modelos e resolução de equações analíticas;
- Saída: modelos adequados a cenários distintos.

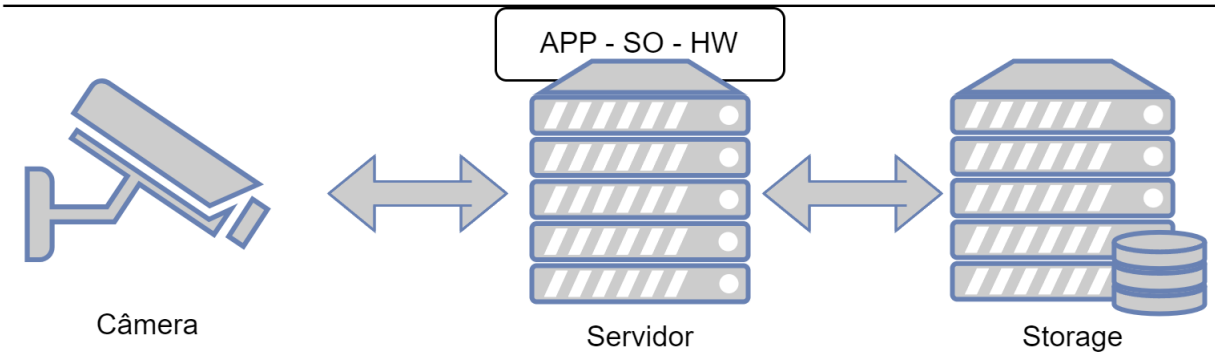
5 Modelos

Neste capítulo, são apresentados os modelos propostos para a avaliação da disponibilidade. Inicialmente é apresentada uma modelagem hierárquica heterogênea que representa uma arquitetura base e os seus componentes. Esse modelo, em RBD, será utilizado para, por meio da técnica de validação da disponibilidade, obter os tempos médios de falha e reparo de cada componente do sistema, além de avaliar os índices de importância para a disponibilidade (*Availability Importance*) e avaliação da disponibilidade com redundância *hot standby*. Em seguida, é apresentado uma modelagem em SPN utilizada para representar o sistema com componentes em redundância *cold standby*.

5.1 Modelo de disponibilidade da arquitetura base

Esta seção apresenta o modelo proposto para a arquitetura base (Figura 16), composta por câmera, servidor (aplicação, Sistema Operacional e *hardware*) e *Storage*. Uma abordagem de modelagem hierárquica heterogênea foi adotada para representar a arquitetura e avaliar a disponibilidade do serviço. A representação inicial ocorreu por meio de modelo RBD.

Figura 16 – Arquitetura base

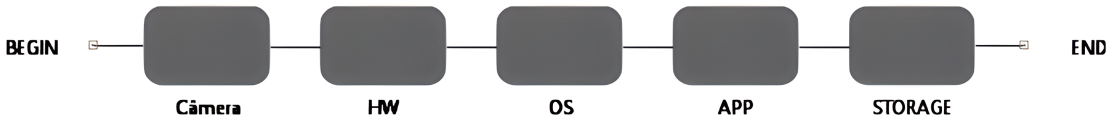


Fonte: O autor.

Modelos hierárquicos têm sido amplamente utilizados para o planejamento e avaliação de dependabilidade de vários tipos de sistemas, algumas aplicações de modelagem utilizadas são: computação em nuvem (PEREIRA et al., 2022; SOUSA et al., 2014; DANTAS et al., 2012; ANDRADE et al., 2017; MATOS et al., 2015), consumo energético (GONÇALVES et al., 2022; SILVA et al., 2014), *Vod Streaming* na nuvem (MELO et al., 2014) e outros. O modelo RBD foi adotado como uma representação de alto nível para ilustrar o relacionamento entre os componentes que compõem a arquitetura proposta. O RBD oferece percepções sobre como o funcionamento dos componentes de um sistema impacta a operação do sistema na totalidade. Além disso, o RBD permite o cálculo de métricas de confiabilidade, como a disponibilidade (MACIEL, 2023b), que é um dos objetivos desta pesquisa.

A Figura 17 representa o modelo de disponibilidade para a arquitetura base. Esse modelo será utilizado para avaliar a disponibilidade e os índices de importância para a disponibilidade. Além disso, esse modelo será readequado para representar cenários com redundância *hot standby*.

Figura 17 – RBD - Arquitetura base



Fonte: O autor.

O modelo descrito na Figura 17 é caracterizado pela composição de cinco blocos em série dos seguintes subsistemas: *câmera*, *hardware(hw)*, *sistema operacional(os)*, *aplicação(app)* e *storage*. Esses subsistemas são individualmente importantes para a disponibilidade, considerando que para que o sistema esteja disponível todos esses subsistemas devem estar funcionando adequadamente, sendo o modo operacional representado pela Expressão 5.1:

$$OM_{sys} = camera \wedge hw \wedge os \wedge app \wedge storage \tag{5.1}$$

onde, *camera*, *hw*, *os*, *app* e *storage* são probabilidades que representam o estado de operação do componente do sistema.

O resultado da multiplicação da disponibilidade de cada sistema resulta na disponibilidade total do ambiente, considerando a arquitetura base. Sendo assim, a disponibilidade total pode ser obtida pela Equação 5.2.

$$A_{sys} = A_{camera} \times A_{hw} \times A_{os} \times A_{app} \times A_{storage} \tag{5.2}$$

onde, A_{sys} é a disponibilidade total do sistema, A_{hw} é a disponibilidade do hardware, A_{os} é a disponibilidade do sistema operacional, A_{app} é a disponibilidade da aplicação e $A_{storage}$ é a disponibilidade do componente armazenamento.

A indisponibilidade do sistema pode ser obtida com a Equação 5.3 e o *downtime* anual(horas) pode ser obtido com a Equação 5.4.

$$UA = 1 - A_{sys} \tag{5.3}$$

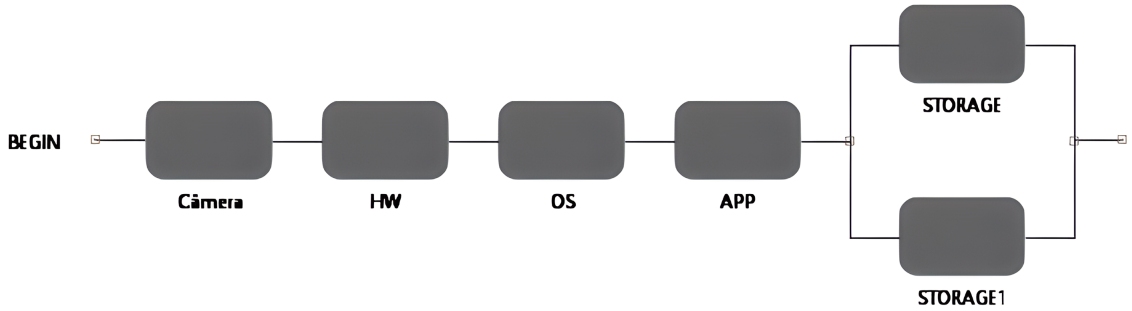
$$Dty_{sys} = UA \times 8760h \tag{5.4}$$

A Figura 18 apresenta uma alteração na infraestrutura que agora é formada pelos seguintes componentes: câmera, *hardware*, sistema operacional, aplicação, *storage* e *storage1*. O componente *storage* está redundante com *storage1*, a redundância deste componente é do tipo *hot standby*. Logo, o sistema estará funcionando se pelo menos um componente esteja ativo e se os demais componentes do modelo também estiverem em funcionamento. Em outras palavras, para ocorrer a falha do sistema é necessário que os componentes *storage* e *storage1* falhem, ou ocorra a falha de qualquer um dos demais componentes (câmera, *hardware*, sistema operacional ou aplicação) do modelo que se encontram conectados em série. Sendo assim, a disponibilidade pode ser computada através da Equação 5.5,

$$A_{sys} = (1 - ((1 - A_{storage}) \times (1 - A_{storage1}))) \times A_{camera} \times A_{hw} \times A_{os} \times A_{app} \tag{5.5}$$

Os resultados obtidos com o modelo em RBD servirão de insumos para a modelagem SPN.

Figura 18 – RBD - Arquitetura base modificada

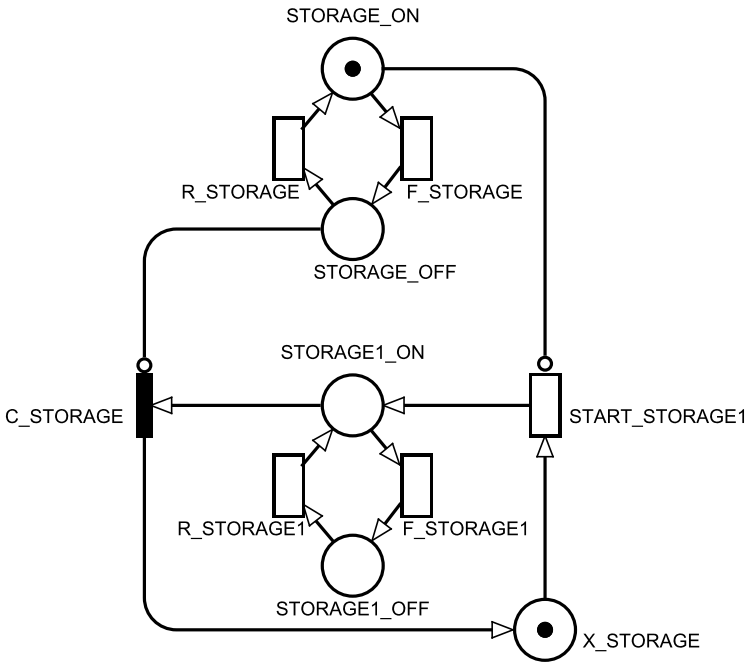


Fonte: O autor.

5.2 Modelos para disponibilidade em *cold standby*

A Figura 19 apresenta, como um exemplo, um modelo em SPN que possui o componente *storage* em redundância *cold standby*. Esse tipo de redundância é uma estratégia do tipo ativo/passivo, ou seja, possui dois componentes redundantes, sendo que apenas um componente está funcional por vez.

Figura 19 – SPN - *Storage* em *cold standby*



Fonte: O autor.

O *storage* principal é identificado pelo lugar STORAGE e o secundário por STORAGE1. Para gerenciar e identificar qual dos componentes está ativo existe um lugar apresentado por X_STORAGE, quando há um *token* nesse local, sinaliza que o *storage* principal está ativo e o secundário inoperante. É preciso controlar a inicialização e o desligamento do *storage* secundário (STORAGE1), pois ele só poderá estar ativo se o *storage* principal estive em estado de falha. Essas ações são gerenciadas pela transição temporizada START_STORAGE1 e pela transição imediata C_STORAGE. A transição imediata representa o desligamento do *storage* secundário que ocorre apenas quando o *storage* principal estiver ativo. O modelo possui arcos de inibição utilizados para limitar os momentos de disparo das transições. A transição imediata C_STORAGE sofre ações do arco inibidor. O arco inibidor não permite que a transição seja disparada se houver *token* no lugar STORAGE_OFF.

Inicialmente, o *storage* principal está ativo e o *storage* secundário está desligado, de forma que os lugares STORAGE_ON e X_STORAGE possuem um *token* cada. Assim, a única transição ativa é F_STORAGE, que representa a falha do *storage* principal. Quando disparada, um *token* é consumido e gerado no lugar STORAGE_OFF. Neste momento há uma indisponibilidade no serviço, pois todos os *storage* estão inoperantes. Agora as transições R_STORAGE e START_STORAGE1 estão habilitadas para dispararem. Se R_STORAGE for acionada, o *storage* principal retornará ao estado ativo. No entanto, se a transição START_STORAGE1 for acionada, o *token* de X_STORAGE será consumido e gerado no lugar STORAGE1_ON, que levará o *storage* secundário a um estado ativo. Se R_STORAGE for disparada, o lugar STORAGE_ON receberá um *token* indicando que o *storage* principal retornou ao estado ativo. Após este processo a transição C_STORAGE é acionada imediatamente desligando o servidor secundário, levando o modelo de volta ao estado inicial. Nos estudos de caso desenvolvidos neste trabalho está estratégia será aplicada também em outros componentes.

6 Estudos de Caso

Este capítulo apresenta cinco estudos de caso. O primeiro visa apresentar os dados de desempenho e consumo elétrico obtidos nas medições. O segundo estudo aplica a técnica de regressão linear para estimar a utilização de recursos computacionais e de consumo de energia. O terceiro estudo faz uso da técnica de validação do modelo da arquitetura base. O quarto estudo avalia a arquitetura base, de modo a aplicar e avaliar as redundâncias *cold standby* e *hot standby* em alguns cenários. O quinto estudo adota o modelo validado para cenários com componentes redundantes, considerando o índice de importância para a disponibilidade (*Availability Importance*).

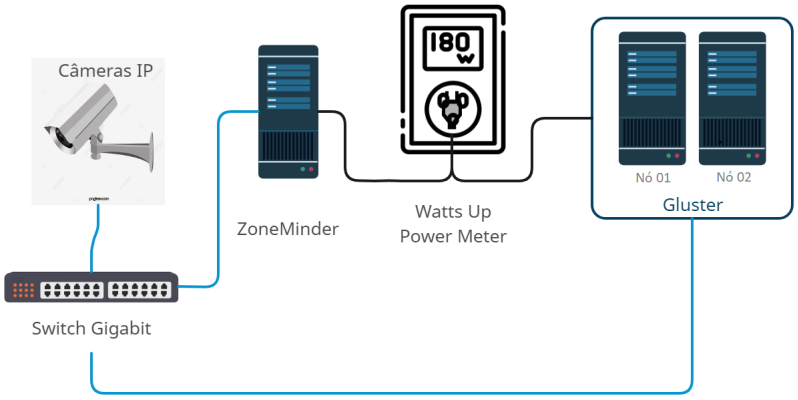
6.1 Estudo de Caso I

Esse estudo visa comparar o desempenho e o consumo de energia de um sistema de monitoramento de vídeo com diferentes configurações para a forma de armazenamento das imagens. São avaliadas métricas como utilização de CPU, memória, rede e o consumo elétrico a partir da adição de novas câmeras. Os cenários são avaliados com o armazenamento em volumes distribuídos e replicados. A utilização de volumes distribuídos (Figura 2) permite a expansão da capacidade de armazenamento com a adição de novos elementos ao cluster, de forma rápida e de baixo custo. Para sistemas de videomonitoramento é importante ter disponível uma grande capacidade de armazenamento, considerando que as imagens necessitam ser guardadas por certo período. No estado do Rio de Janeiro, por exemplo, as imagens de câmeras em estabelecimentos bancários deve ser armazenadas por, no mínimo, dois anos (ALERJ, 2016). Os volumes podem estar geograficamente separados ou na mesma sala. Vale salientar que na forma distribuída não há redundância de dados, ou seja, a falha de um dos elementos pode comprometer os dados armazenados. Quando há utilização de volumes replicados (Figura 3), os dados armazenados estão disponíveis em número de réplicas definidos no sistema, com isso é garantido a confiabilidade no armazenamento, pois com a falha de alguma unidade de armazenamento, os dados seguem acessíveis por meio das réplicas. Assim como na configuração distribuída, os volumes podem estar em salas ou espaços físicos distintos.

6.1.1 Ambiente de experimentos

A Figura 20 ilustra a configuração do ambiente adotado na realização dos experimentos. Esse ambiente é composto por três máquinas físicas, sendo uma com o ZoneMinder, utilizado para capturar o fluxo de vídeo das câmeras e direcionar as imagens para o cluster de armazenamento. As outras duas máquinas físicas são utilizadas para realizar o armazenamento em volumes distribuídos e em volumes replicados. É importante destacar que a estrutura do GlusterFS não necessita de um servidor de metadados, e que a comunicação entre os dispositivos é realizada por cabos UTP de CAT6 através de um *switch* Gigabit. A Tabela 4 apresenta as características técnicas de cada uma das máquinas utilizadas no ambiente de experimentos.

Figura 20 – Ambiente de experimentos



Fonte: O autor.

Tabela 4 – Características técnicas das máquinas

Componente	Sistema Operacional	CPU	RAM	HD	NIC
Máquinas do Cluster	Ubuntu Server 20.04	AMD A855B 3.2Ghz	8GB	500GB 7200RPM	1 Gigabit Ethernet
Servidor ZoneMinder	Ubuntu Server 20.04	Xeon E3-1220 v6 3Ghz	16GB	1TB 7200RPM	1 Gigabit Ethernet

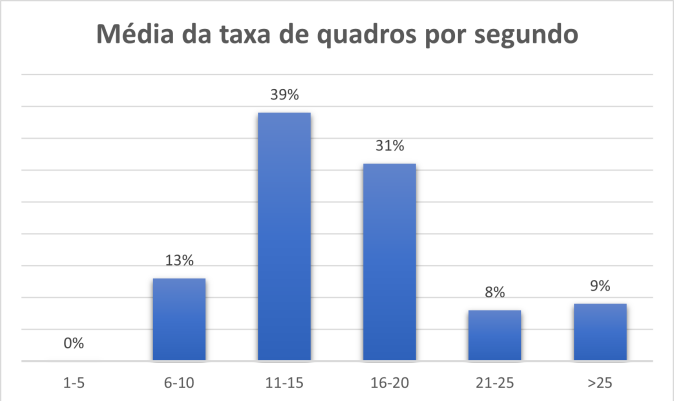
Fonte: o autor

O consumo de energia está sendo obtido a partir do Watts Up Pro Portable Power Meter As demais métricas (ex., utilização de CPU e memória, dados enviados e recebidos via rede) foram obtidas com os softwares *Prometheus* e *Grafana*. Os experimentos levaram

em consideração vários cenários incrementais, onde inicialmente se adotou uma única câmera. Posteriormente, duas câmeras e assim sucessivamente, até se obter um total de 11 câmeras gerando dados simultaneamente.

As configurações das câmeras, nos aspectos de resolução e quadros por segundo, garantem que o fluxo de vídeo enviado aos componentes do sistema é semelhante para manter um ambiente de experimentos padronizado. Os fluxos de vídeos gerados pelas câmeras foram padronizados considerando o guia de taxa de quadros para vigilância por vídeo elaborado pelo IPVM (IPVM, 2023) instituição reconhecida mundialmente em segurança física, incluindo a vigilância por vídeo. O guia elaborado pelo IPVM (IPVM, 2021) expõe um comparativo entre as taxas de quadros por segundo e o impacto na obtenção das imagens, apresentando graficamente a média da taxa de quadros utilizadas pelo setor de videomonitoramento (Figura 21), chegando à conclusão de que 15 quadros por segundo, fornece quadros suficientes para a captura de imagens para fins de monitoramento de segurança. Taxas de 30 quadros por segundo aumentaria consideravelmente os custos de armazenamento, com uma melhoria mínima de detalhes capturados. O *codec* de vídeo utilizado foi o H.264, padrão para gravação, compactação e distribuição de conteúdo de vídeo online. Apelidado de "o codec mais utilizado no mundo" (STANIMIROVIC, 2021), e também conhecido como *Advanced Video Coding (AVC)*. O H.264 foi desenvolvido em conjunto pela International Telecommunications Union e como MPEG-4 pela *International Organization for Standardization (ISO)*. Assim, as configurações aplicadas as câmeras de segurança utilizadas no experimento foram: resolução de 1280x720 pixels, 15 Quadros por segundo e codec de vídeo: H.264.

Figura 21 – Média de quadros por segundo



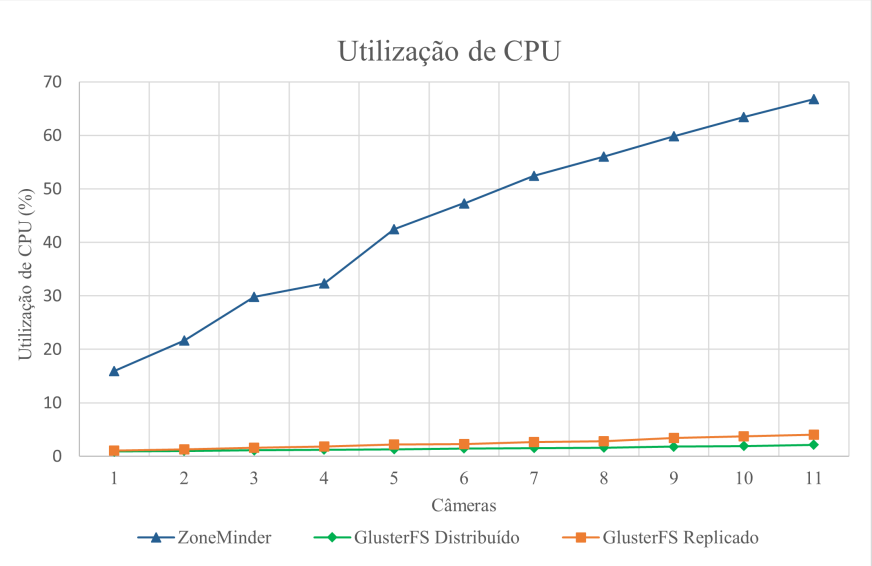
Fonte: O autor

6.1.2 Resultados

Para a obtenção dos resultados, os experimentos foram realizados com duração de 24 horas cada. A partir dos dados coletados foram calculadas as médias com cada cenário avaliado, ou seja, considerando a variação de cargas de trabalho em formas de armazenamento diferentes (distribuído e replicado).

A partir dos dados das medições realizadas, foi possível extrair o consumo médio de energia elétrica, a utilização da CPU, utilização de memória, tráfego de rede e escrita no disco para os cenários assumindo o armazenamento do GlusterFS com volumes distribuídos e replicados. A Figura 22 mostra os resultados da utilização de CPU nos cenários considerados, com número crescente de câmeras. É possível identificar que o incremento no número de câmeras no sistema faz com que o nível de CPU da infraestrutura de armazenamento sofra um leve e proporcional crescimento. Esse comportamento é observado tanto assumindo o ambiente distribuído como no replicado. Todavia, é importante destacar que o aumento é expressivo na infraestrutura física que fornece suporte ao VMS (CPU - ZoneMinder).

Figura 22 – Utilização de CPU



Fonte: O autor.

Com a finalidade de garantir que a diferença da utilização de CPU dos cenários distribuído e replicado é significativa, aplicou-se a abordagem do intervalo de confiança para o teste t pareado (CHUNG, 2003). Nessa abordagem, primeiro calcula-se uma nova

variável com base na diferença das médias entre os cenários avaliados, conforme Equação 6.2, onde X_{1i} é a média do primeiro cenário (distribuído), X_{2i} é a média do segundo cenário (replicado) e Z_i é a diferença entre as médias. Em seguida, calcula-se a média e o desvio padrão da nova variável Z. Por fim, utiliza-se a Equação 6.3 para calcular o intervalo de confiança, onde $\overline{Z}$ é a média dos valores de Z, $t_{\alpha/2,n-1}$ é o valor da distribuição t para $\alpha/2$ e $n-1$ são os graus de liberdade, S é o desvio padrão dos valores de Z e n é o número de pares de médias avaliadas. A Tabela 5 apresenta o resumo estatístico desse teste. Como o intervalo gerado [-0.21758446355935115, -0.12067255184064882] não inclui o valor 0, existe evidência estatística para afirmar que a diferença entre as amostras é estatisticamente significativa com 95% de confiança.

$$\overline{X}_{1i} - \overline{X}_{2i} = Z_i \tag{6.2}$$

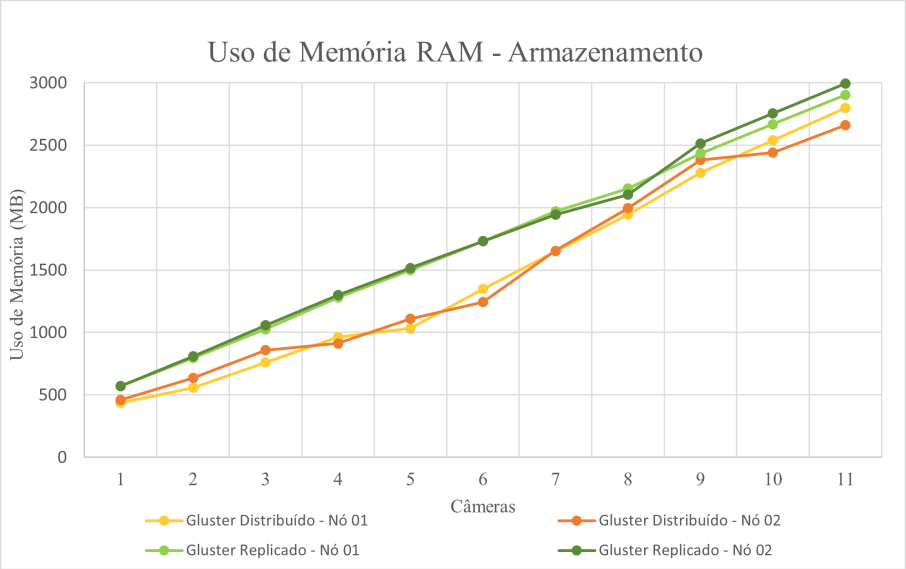
$$\overline{Z} \pm t_{\alpha/2,n-1}\frac{s}{\sqrt{n}} \tag{6.3}$$

Tabela 5 – Teste t pareado - CPU - 8 câmeras

Estatística	**Resultado**
Quantidade	30
Desvio Padrão	0.023064137998092323
Nível de Confiança	95%
Limite Inferior	-0.21758446355935115
Limite Superior	-0.12067255184064882

A Figura 23 apresenta os dados obtidos referente a utilização de memória pelos componentes responsáveis pelo armazenamento das imagens, conforme a estratégia de armazenamento, volumes distribuídos e replicados, respectivamente. Percebe-se que o uso de memória RAM é semelhante para os nós quando configurados com o mesmo tipo de volume. O uso de memória quando os componentes operam com o armazenamento replicado tende a ser ligeiramente maior em relação aos volumes distribuídos.

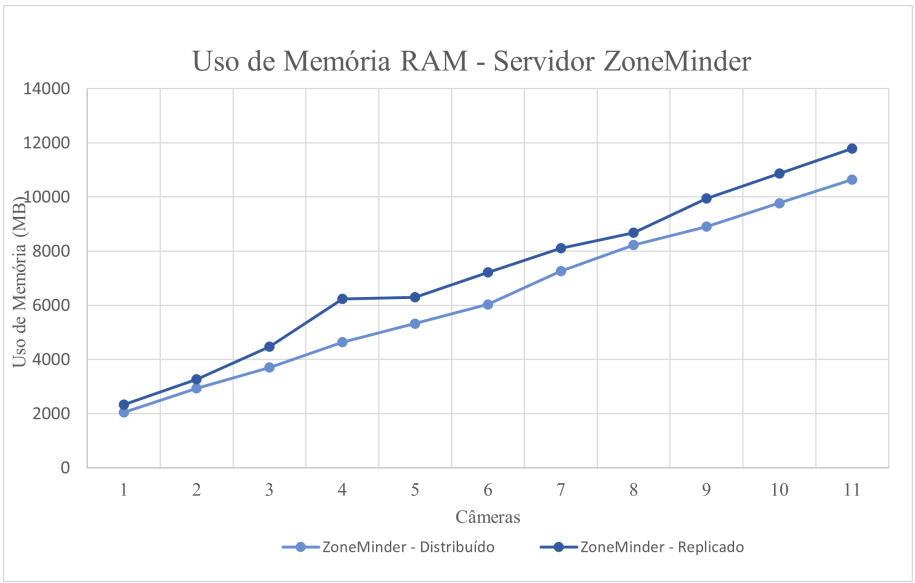
Figura 23 – Utilização de Memória pelos componentes de armazenamento



Fonte: O autor.

Os dados referentes a utilização de memória RAM pelo servidor ZoneMinder podem ser visualizados na Figura 24, os resultados apresentados consideram a utilização de memória pelo servidor com as diferentes configurações de armazenamento (distribuído e replicado) percebe-se que o consumo de memória pelo servidor é levemente afetado com a alteração nas configurações dos volumes de armazenamento.

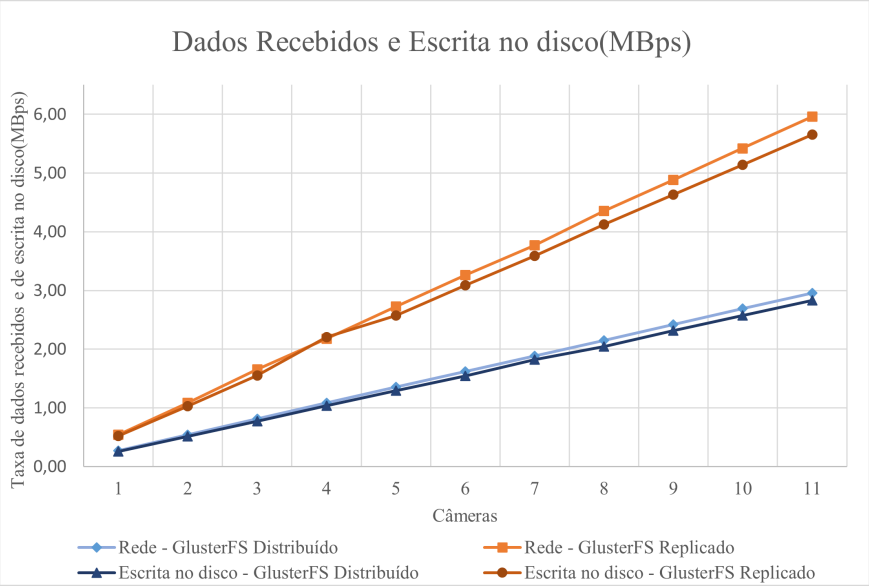
Figura 24 – Utilização de memória pelo gerenciador de vídeo



Fonte: O autor.

A Figura 25 apresenta os resultados referentes às taxas de recebimento dos dados via rede e escrita no disco, ambos expressos em MBps. É possível observar que os volumes replicados acabam por demandar o dobro das taxas em relação aos volumes distribuídos. Isso ocorre porque no modo replicado os dados são enviados, simultaneamente, para os computadores que formam o SAD. Dessa forma, pode-se compreender que haverá um maior comprometimento dos recursos com a adição de mais câmeras.

Figura 25 – Taxa de dados recebidos e taxa de escrita no disco



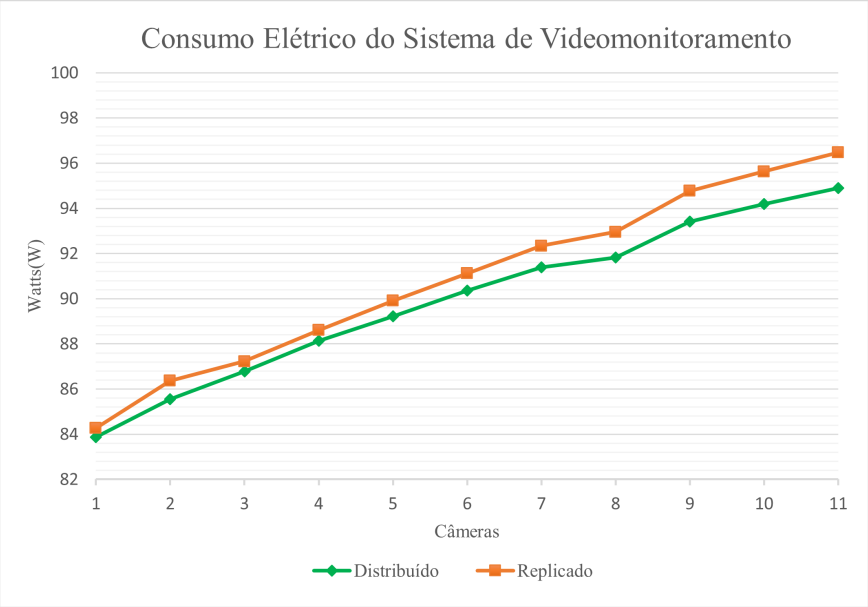
Fonte: O autor.

A Figura 26 ilustra o resultado comparativo do consumo de energia do sistema de videomonitoramento (ZoneMinder e nós do GlusterFS) com a utilização das diferentes formas de armazenamento (distribuído e replicado). É possível perceber que o consumo de energia tem um crescimento gradual e o consumo do armazenamento com volumes replicados é discretamente superior ao modo com volumes distribuídos.

A partir dos dados obtidos nas medições é possível calcular o consumo e o custo mensal. A Equação 6.4 pode ser utilizada para obter o consumo mensal em kWh, onde C é o consumo mensal do sistema, P é a potência elétrica do sistema, T a quantidade de horas de funcionamento por dia e D o número de dias.

$$C(\text{kWh}) = P(\text{W}) \times T(\text{horas/dia}) \times D(\text{dias/mês}) \tag{6.4}$$

Figura 26 – Consumo elétrico do sistema de videomonitoramento



Fonte: O autor.

A Tabela 6 apresenta as informações sobre o consumo elétrico e o custo mensal, os dados são apresentados em kWh (Quilowatt-hora) e considera o kWh custando R$ 0,74. Nota-se que os custos com o armazenamento replicado são discretamente maiores. Por exemplo, assumindo o custo mensal, o valor considerando para o armazenamento distribuído saiu de R$44,68 para R$50,56; considerando o replicado, partiu de R$44,90 para R$51,40. Essa diferença, no entanto, pode ser expressiva ao se considerar um número superior de câmeras. O consumo elétrico é afetado de maneira progressiva, impactando no custo. Percebe-se que o modo de armazenamento replicado tende a ser mais oneroso à medida que novas câmeras entram em funcionamento.

Tabela 6 – Consumo Elétrico e Custo mensal

Câmeras	kWh		Custo(R$)	
	Distribuído	Replicado	Distribuído	Replicado
01	60,38	60,67	44,68	44,90
02	61,60	62,18	45,58	46,02
03	62,49	62,80	46,24	46,47
04	63,46	63,80	46,96	47,21
05	64,24	64,73	47,54	47,90
06	65,06	65,61	48,14	48,55
07	65,80	66,48	48,69	49,20
08	66,11	66,93	48,92	49,53
09	67,25	68,23	49,77	50,49
10	67,82	68,85	50,19	50,95
11	68,32	69,45	50,56	51,40

Fonte: o autor

6.2 Estudo de Caso II

O objetivo desse segundo estudo é propor projeções futuras do comportamento do sistema para cenários que não foram possíveis de serem medidos. Isso foi possível ao se observar dados e gráficos obtidos nos experimentos apresentados no Estudo de caso I. Ao observar a Figura 26, foi possível identificar uma relação linear entre o consumo de energia e o número de câmeras em funcionamento. A partir de então, regressões lineares foram realizadas para obter equações que possam ser utilizadas para realizar projeções em cenários diversos, auxiliando na elaboração do planejamento de infraestruturas de videomonitoramento.

Ao dimensionar corretamente o consumo elétrico do sistema, é possível otimizar o uso de energia e reduzir os custos operacionais. Isso inclui escolher componentes com eficiência energética, como dispositivos de armazenamento. Conhecer o consumo elétrico do sistema auxilia, inclusive, no planejamento adequado da infraestrutura elétrica.

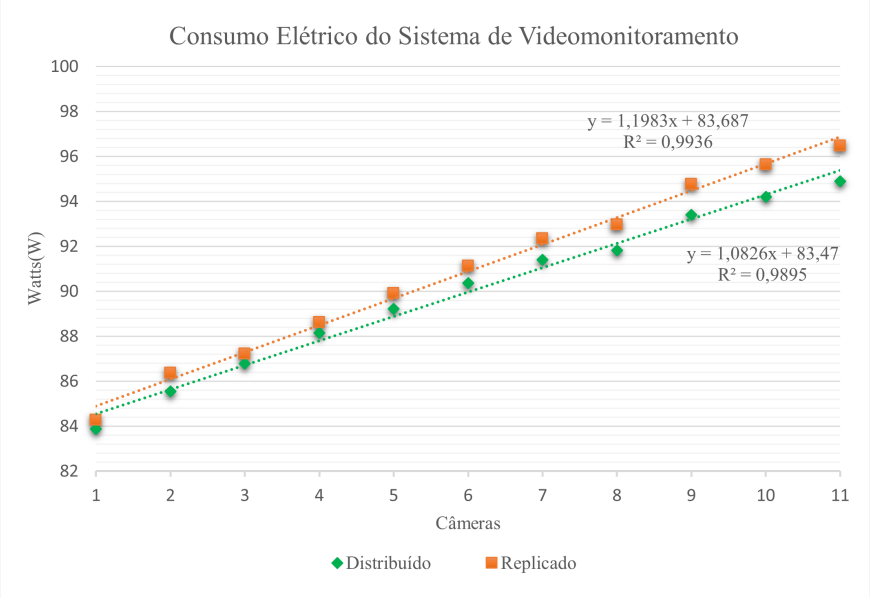
Para o consumo elétrico, a Equação 6.5 foi obtida para o sistema de videomonitoramento com armazenamento distribuído e a Equação 6.6 para o cenário com armazenamento replicado, onde y é a previsão do consumo de energia em $Watts$ e x é o número de câmeras utilizadas no sistema.

$$y = 1,0826x + 83,47 \tag{6.5}$$

$$y = 1,1983x + 83,687 \tag{6.6}$$

A partir dessas equações é possível estimar o consumo de energia do sistema. Os coeficientes de determinação R foram de 0,9895 e 0,9936, respectivamente, indicando a validade da regressão. A Figura 27 expõe o gráfico utilizado para obter as equações da regressão linear aplicada à projeção do consumo elétrico, a partir de um número maior de câmeras em utilização.

Figura 27 – Regressão Linear - Consumo elétrico



Fonte: O autor.

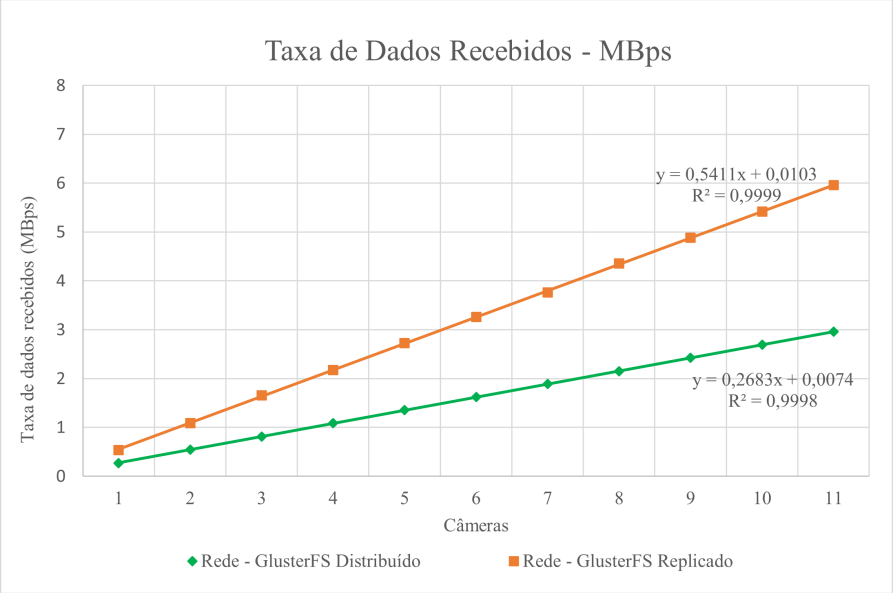
O tráfego de rede desempenha um papel crítico em sistemas de videomonitoramento IP, permitindo a transmissão eficiente e confiável de fluxos de vídeo em tempo real. Uma rede com largura de banda adequada, baixa latência, confiabilidade e segurança adequada é essencial para garantir um sistema de videomonitoramento eficaz, oferecendo visualização

em tempo real, gravação contínua e acesso remoto aos fluxos de vídeo. As Equações 6.7 e 6.8 foram obtidas com as taxas de tráfego de rede referentes a dados recebidos (Figura 28) pelo cluster de armazenamento distribuído e replicado, respectivamente. Onde y é a projeção da taxa de recebimento em $MBps$ e x é o número de câmeras em funcionamento. As equações tiveram coeficiente de determinação 0,9998 e 0,9999, respectivamente.

$$y = 0,2683x + 0,0074 \tag{6.7}$$

$$y = 0,5411x + 0,0103 \tag{6.8}$$

Figura 28 – Regressão Linear - Dados recebidos



Fonte: O autor.

A memória RAM desempenha um papel crucial em sistemas de videomonitoramento, garantindo o desempenho e a eficiência na captura, processamento e armazenamento de dados de vídeo em tempo real. Em um sistema de videomonitoramento, várias câmeras podem estar capturando simultaneamente fluxos de vídeo de alta resolução. Esses dados de vídeo precisam ser processados e armazenados de forma rápida e eficiente para permitir a visualização ao vivo e a recuperação de gravações. É aí que a memória RAM desempenha um papel fundamental.

A memória RAM atua como uma área de armazenamento temporário de dados em tempo real. Ela permite que o sistema capture, armazene temporariamente e processe os fluxos de vídeo sem interrupções, evitando a perda de quadros ou atrasos na transmissão. Quanto maior a capacidade da memória RAM, mais fluxos de vídeo simultâneos o sistema é capaz de manipular sem comprometer o desempenho.

Para projeções de utilização de memória RAM obtivemos as Equações 6.9 e 6.10 para o componente com gerenciador de vídeo e as Equações 6.11 e 6.12 para cada nó do cluster de armazenamento, considerando em ambos os casos as estratégias de armazenamento distribuído e replicado, respectivamente.

$$y = 864,26x + 1131,7 \tag{6.9}$$

$$y = 916x + 1705,9 \tag{6.10}$$

$$y = 244,31x + 16,552 \tag{6.11}$$

$$y = 239,24x + 318,66 \tag{6.12}$$

A Tabela 7 mostra as estimativas de consumo de energia (W), custo (R$), taxa de transferência (MBps) e Utilização de memória RAM calculadas a partir das equações anteriores considerando 20, 40, 60, 80 e 100 câmeras. Percebemos que a diferença do consumo e custo mensal entre a utilização de volumes distribuídos e replicados tende a aumentar expressivamente com o aumento da quantidade de câmeras e consequente necessidade de armazenar mais dados. Por exemplo, a diferença de custo com 20 câmeras é de 2,42% e chega a 10% com o uso de 1000 câmeras. Os valores estimados para as taxas de recebimento de dados são proporcionais, ou seja, na configuração de volumes replicados a taxa de dados chega a ser duas vezes maior em comparação ao volume distribuído, isso ocorre porque nosso cenário está configurado com duas réplicas. A utilização de memória RAM é proporcional ao número de câmeras em funcionamento, percebe-se que a medida que o número de câmeras é ampliado em dobro, a demanda por memória também

dobra. Essa tendência se aplica tanto no servidor com o ZoneMinder como também nos componentes do armazenamento.

Tabela 7 – Projeções obtidas com regressão linear

	Consumo de Energia Estimado(W)		Custo Mensal (em Reais)			Taxa de Dados Recebidos(MBps)		Memória RAM(MB)			
Câmeras	Armazenamento Distribuído	Armazenamento Replicado	Armazenamento Distribuído	Armazenamento Replicado	Diferença (%)	Armazenamento Distribuído	Armazenamento Replicado	ZoneMinder Distribuído	Zoneminder Replicado	Gluster Distribuído	Gluster Replicado
20	105,11	107,65	56,00	57,36	2,42%	5,37	10,83	18417	20026	4903	5103
40	126,75	131,61	67,53	70,13	3,85%	10,74	21,65	35702	38346	9789	9888
60	148,39	155,58	79,06	82,90	4,85%	16,11	32,48	52987	56666	14675	14673
80	170,03	179,55	90,59	95,66	5,59%	21,47	43,30	70273	74986	19561	19458
100	191,67	203,51	102,12	108,43	6,17%	26,84	54,12	87558	93306	24448	24243
1000	1165,47	1281,987	620,96	683,04	10%	268,31	541,11	865392	917706	244327	239559

Fonte: o autor

6.3 Estudo de Caso III

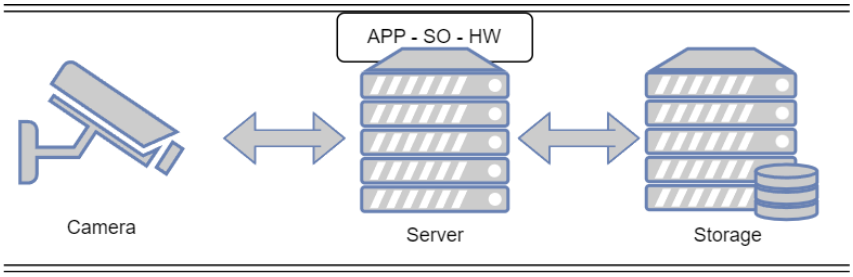
O objetivo desse estudo é demonstrar os procedimentos utilizados para validar a disponibilidade obtida a partir do modelo RBD proposto que representa nossa arquitetura de base (Figura 17). Para garantir a confiabilidade dos resultados obtidos a partir dos modelos propostos, é necessário realizar técnicas de validação que verifiquem a consistência entre esses modelos e o sistema real. Essas técnicas de validação visam validar os principais aspectos do modelo, incluindo suposições, valores e distribuições dos parâmetros de entrada, bem como os valores de saída gerados (JAIN, 1991).

Uma abordagem comum para validar modelos de dependabilidade é comparar os resultados das medidas coletadas em um ambiente real com os valores gerados pelo modelo (LAVENBERG, 1983). Essa comparação permite verificar se os resultados do modelo estão em concordância com o comportamento observado no sistema real. A validação por comparação ajuda a identificar eventuais discrepâncias entre o modelo e o sistema real, permitindo ajustes e refinamentos necessários.

Ao realizar a validação, é importante considerar que os modelos são simplificações do sistema real e, portanto, podem não capturar todos os detalhes e nuances. No entanto, a validação ajuda a garantir que os modelos sejam representações razoavelmente precisas e confiáveis do sistema, permitindo que sejam utilizados para fazer previsões e tomar decisões com base nos resultados gerados.

Para fazer essa validação foi considerado um ambiente base para a realização dos experimentos formado por uma câmera, um servidor (*hardware*, sistema operacional e aplicação) e um componente de armazenamento, conforme Figura 29.

Figura 29 – Ambiente base



Fonte: O autor

Os tempos médios de falha (MTTF) e de reparo (MTTR) utilizados para os componentes da arquitetura base foram obtidos a partir de (PEREIRA et al., 2022). Assumimos que os tempos médio de falha e reparo são exponencialmente distribuídos e, assim, as taxas de falha e de reparo são o inverso do MTTF e MTTR. A Tabela 8 mostra os valores adotados para os MTTFs e MTTRs em horas.

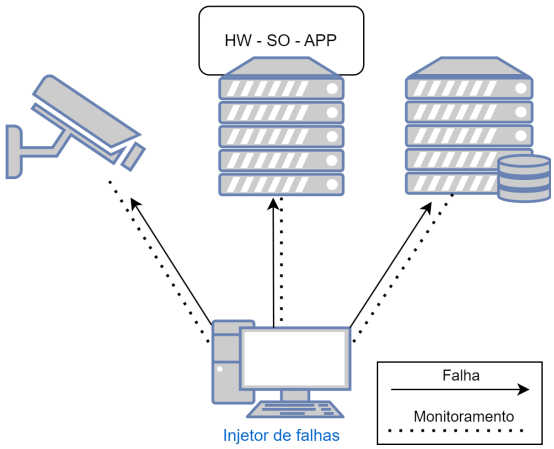
Tabela 8 – MTTF e MTTR

	MTTF(h)	MTTR(h)
Câmera	8760	1,67
Hardware	8760	1,67
OS	2880	1
App	217,8	0,46
Storage	43800	24

Fonte: o autor

Para conduzir a validação do modelo RBD proposto, foi necessário estressar o sistema, acelerando o MTTF dos componentes utilizando como fator de aceleração o valor 1.000. Também foi criado um injetor de falhas (Apêndice A) que é o responsável por inserir falhas em cada um dos componentes, listar os componentes que irão falhar e também monitorar esses dispositivos. O injetor de falha cria uma thread para cada um dos componentes presente no sistema, e são essas as responsáveis por injetar as falhas. Esse injetor foi executado em um microcomputador com CPU AMD de 3.2Ghz, 8 GB de RAM, 500 GB de armazenamento. A Figura 30 mostra como nosso ambiente de validação foi estruturado.

Figura 30 – Ambiente de Injeção de Falhas



Fonte: o autor

A Tabela 9 mostra os valores, em horas, utilizados no injetor de falha. Os tempos de falha e reparo da câmera são assumidos como distribuído exponencialmente.

Tabela 9 – MTTF e MTTR utilizados no injetor de falhas

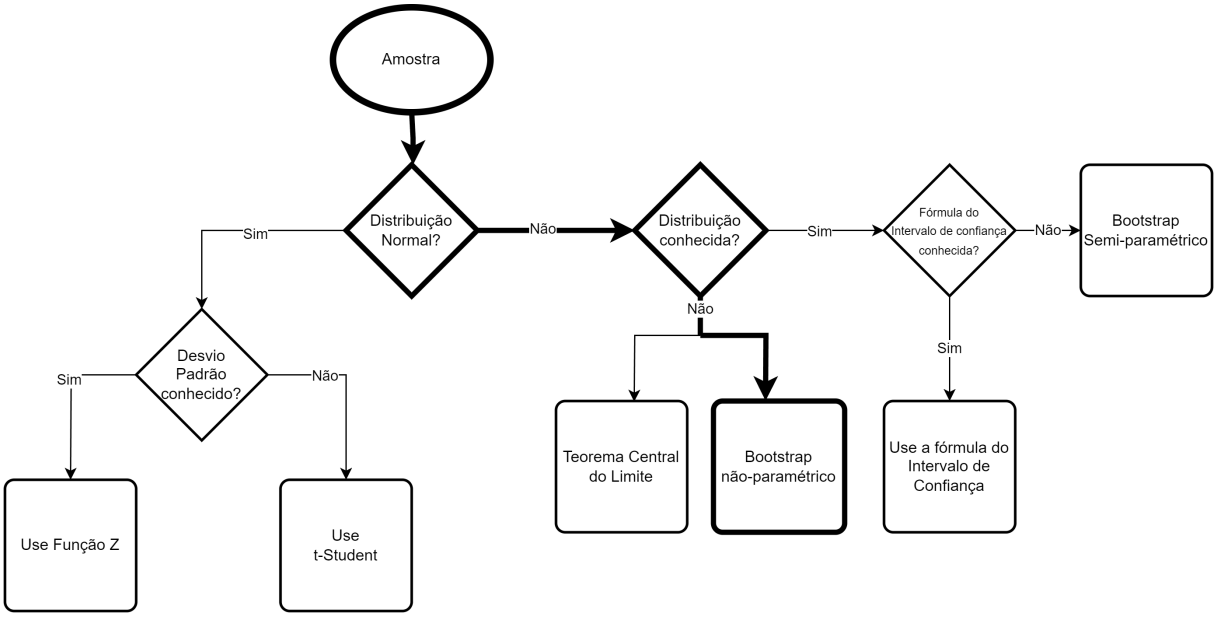
	MTTF(h)	MTTR(h)
Câmera	8,76	1,67
Hardware	8,76	1,67
OS	2,88	1
App	0,21	0,46
Storage	43,8	24

Fonte: o autor

Dessa forma, as threads geram um número aleatório seguindo a distribuição exponencial, os números gerados são utilizados como tempos para falhas e reparos do sistema. Outra thread monitora se o sistema está disponível ou indisponível. Se o sistema estiver ativo o injetor salva o status como “UP”, em um arquivo de texto, e se o sistema estiver inativo salva os status como “DOWN”. Esse monitoramento de checagem do status do sistema é feito a cada intervalo de 10 segundos. Foi realizado um experimento com duração de 168h, sendo obtido um documento com os períodos em que os componentes estavam funcionando ("UPs") ou não estavam respondendo ("DOWNs"). Sendo assim, foi possível então calcular os tempos de falha (MTTF) e de reparo (MTTR). Por exemplo, se antes de um "DOWN"existirem 70 "UPs", então temos que o tempo de atividade foi de 700 segundos, 70x10s (intervalo de obtenção das amostras). Portanto, transformando os "UPs"e "DOWNs"em MTTFs e MTTRs, pode-se calcular a disponibilidade real do sistema e computar também o intervalo de confiança de 95%.

Na amostra dos dados coletados não foi possível determinar a distribuição teórica, e então foram adotadas as inferências estatísticas conforme apresentado na Figura 31. A Figura 31, baseada em (MACIEL, 2023a), apresenta as etapas seguidas para definir a técnica a ser utilizada para o cálculo do intervalo de confiança. O primeiro passo corresponde a analisar se a amostra segue uma distribuição normal. Caso seja e o desvio-padrão da amostra seja conhecido, pode-se usar a expressão analítica da distribuição normal. Caso contrário, deve-se utilizar a distribuição t-student. Assumindo que a amostra não é normalmente distribuída e se tem conhecimento da distribuição da amostra, deve-se utilizar a fórmula analítica do intervalo de confiança apropriado. Caso contrário, emprega-se o bootstrap semi-paramétrico. Se a amostra não segue uma distribuição normal e não se conhece a distribuição adequada, pode-se usar o bootstrap n-paramétrico ou o teorema central do limite.

Figura 31 – Inferência Estatística

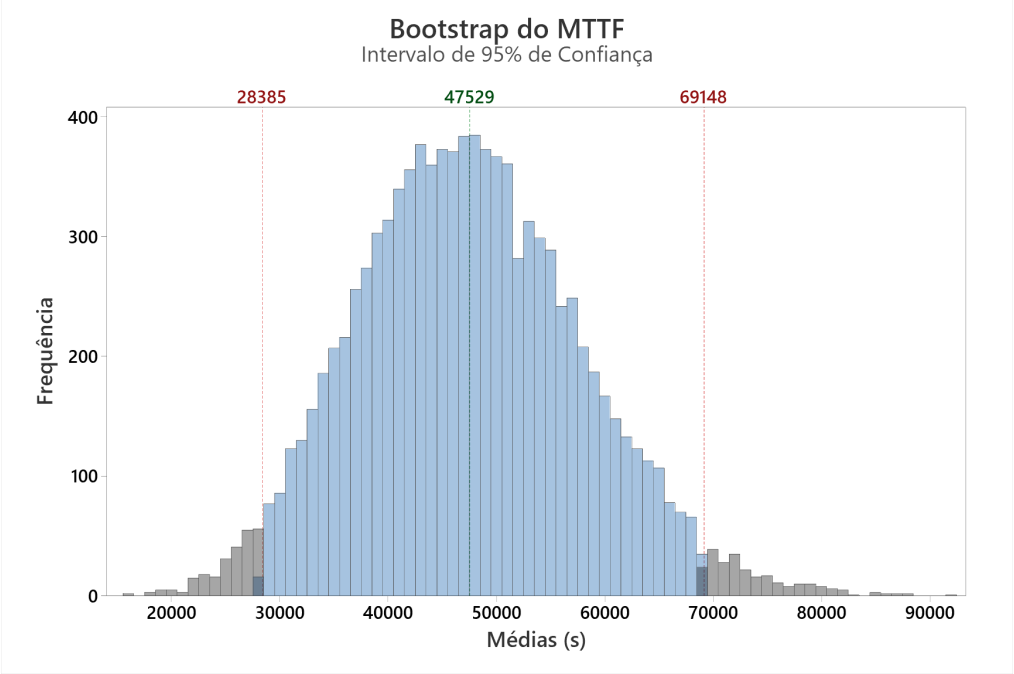


Fonte: o autor

A técnica utilizada foi a de bootstrap n-paramétrico para calcular o intervalo de confiança e gerar as n reamostragens de tamanho m (MACIEL, 2023a). Essas reamostras são obtidas por meio de seleções aleatórias com reposição da amostra original. Portanto, temos n valores das estatísticas de interesse, cada um vindo de uma reamostragem. Esses valores são, então, ordenados em ordem crescente $\widehat{\theta}_1, \widehat{\theta}_2, ..., \widehat{\theta}_i, ..., \widehat{\theta}_{n-1}, \widehat{\theta}_n$, onde $\widehat{\theta}_1$ é o menor valor e $\widehat{\theta}_n$ é o maior valor. Depois disso, o valor $\widehat{\theta}_{\alpha/2\times100\%}$ no percentil $\alpha/2\times100\%$ corresponde ao limite inferior, e o valor $\widehat{\theta}_{1-\alpha/2\times100\%}$ no percentil $(1-\alpha/2)\times100\%$ corresponde ao limite superior (MACIEL, 2023a).

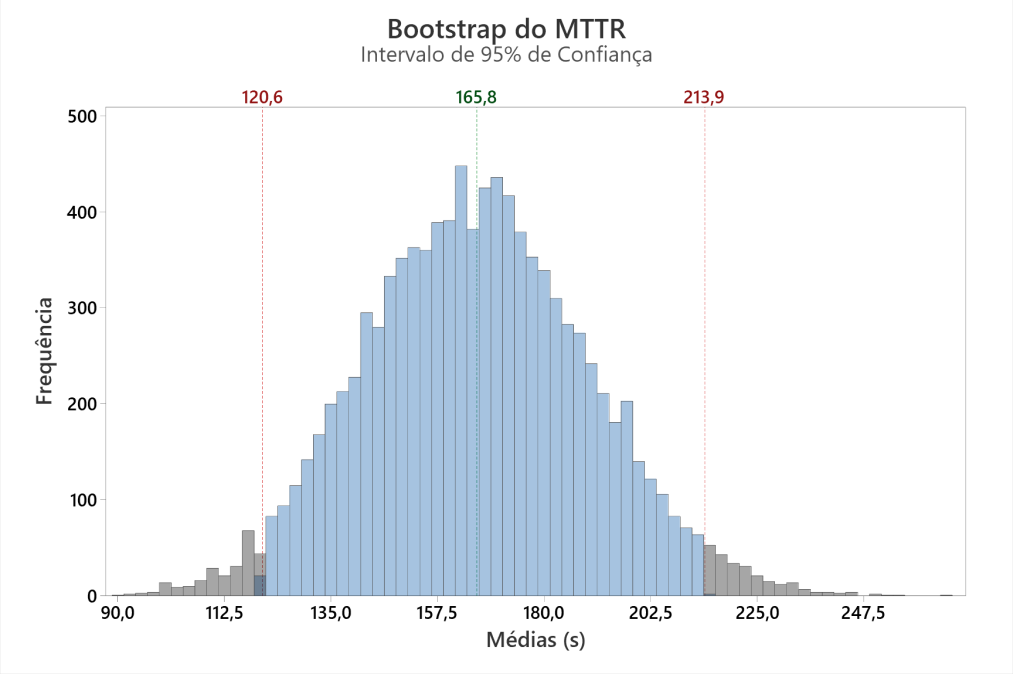
A partir da técnica bootstrap foram geradas 10.000 novas amostras de blocos de "UPs" e "DOWNs". Para cada reamostragem a média foi calculada, e essas médias foram ordenadas de forma crescente, onde 250ª menor e a 250ª maior dessas 10.000 amostras são os "percentis" para o intervalo de confiança de 95%. As Figuras 32 e 33 apresentam os intervalos de confiança para o MTTF e MTTR da nossa arquitetura base, ou seja, considerando todos os componentes. A disponibilidade foi computada a partir da Equação (3.5).

Figura 32 – Bootstrap MTTF



Fonte: O autor.

Figura 33 – Bootstrap MTTR



Fonte: O autor.

Vale ressaltar que a interpretação correta de 95% de confiança depende da compreensão da frequência da probabilidade. Isso significa que, se nosso experimento for realizado repetidamente, a longo prazo, a disponibilidade do sistema será de 95% do tempo nesses intervalos de confiança calculados.

A Tabela 10 mostra uma comparação entre os resultados obtidos a partir do modelo proposto e os resultados quantificados no sistema real. Para os resultados obtidos no sistema real, também foram computados os respectivos intervalos de confiança. Também são apresentados resultados para cada componente de forma individual, os resultados foram obtidos de forma similar.

Tabela 10 – Resultados para disponibilidade

Componente	Modelo	Sistema Real	Intervalo de Confiança
Arquitetura base	0.9966197245593511	0.9965410978496668	[0.9957695385990625; 0.9969163407243262]
Câmera	0.9998093970669976	0.9998244161838111	[0.9997967678164672; 0.999838504189919]
Hardware	0.9998093970669976	0.9998189134880495	[0.9997972494384852; 0.9998405775376139]
OS	0.9996528982992018	0.9996451086731251	[0.9995909815607783; 0.9996992357854718]
APP	0.9978924218821589	0.9980948656093451	[0.9977876578891643; 0.998365172512243]
Storage	0.9994523548740417	0.9993735001647448	[0.9991699047172822; 0.9995770956122074]

Fonte: o autor.

Como podemos observar, os intervalos de confiança contêm a estimativa do modelo para o sistema e seus componentes. Assim, podemos assumir que nosso modelo representa de forma confiável o sistema de videomonitoramento. Consequentemente, podemos considerar que nosso modelo está validado. A partir de agora, podemos usar o modelo para calcular a disponibilidade do sistema. Além de utilizá-lo com o intuito de preparar estratégias para alcançar níveis mais altos de disponibilidade para diversos cenários.

6.4 Estudo de Caso IV

Este estudo de caso mostra como o modelo pode ser utilizado para investigar os problemas de disponibilidade em um sistema (ver Figura 17) e propor estratégias de redundância visando maximizar a disponibilidade.

Vale ressaltar que para as avaliações a seguir, foram utilizados os valores de MTTF e MTTR validados e disponíveis na Tabela 8. O *Mercury tool* (SILVA et al., 2015; MODCS, 2023) foi utilizado para calcular os respectivos valores de disponibilidade.

6.4.1 Arquitetura base

Realizando a avaliação da arquitetura base (Figura 17), foi obtido um tempo de inatividade de 29,63 horas para o sistema, o que significa que durante um ano o sistema ficaria fora do ar por cerca de 30 horas; consequentemente, o *uptime* seria de 8.736,18 horas. Se a solução for adotada, o número de noves é 2,4710. A Tabela 11 apresenta os resultados obtidos no modelo de arquitetura base.

Tabela 11 – Resultados da disponibilidade da arquitetura base

Métrica	**Valores**
Disponibilidade	0.9966197245593511
Indisponibilidade	0.00338027544
# de 9's	2.471047909935344
Uptime(h)	8736.181908376244
Downtime(h)	29.630861623757458

Fonte: o autor

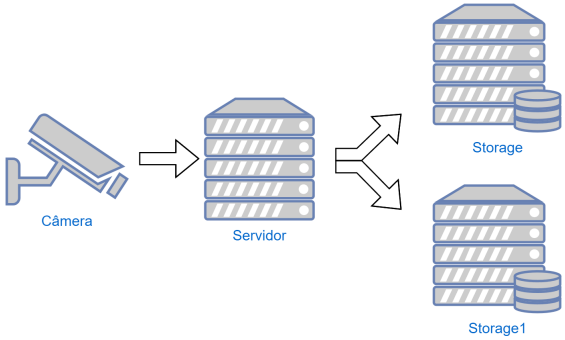
Para melhoria da disponibilidade, existe um mecanismo bem conhecido chamado redundância. Conforme descrito no Capítulo 3, existem três tipos de redundâncias: *cold*, *warm* e *hot standby*. A escolha da estratégia de redundância depende de quão crítico é o processo e as consequências de falhas dos componentes (MELO et al., 2018). A redundância *cold standby* é aplicada a serviços que podem tolerar atrasos na restauração, o que significa que a intervenção humana é aceitável e o tempo não é uma alta prioridade.

Por outro lado, a redundância *warm standby* ocorre quando a resposta à falha e ao tempo é importante, mas não crítica, portanto, uma interrupção temporária é aceitável. Finalmente, redundância *hot-standby* oferece correção instantânea do processo quando há uma falha; em outras palavras, tempo e segurança têm alta prioridade e o sistema não pode parar por nenhum momento.

Nesse sentido, são propostos modelos adotando redundâncias *hot standby* e *cold standby* para os componentes do sistema. Assim, avaliaremos os seguintes cenários:

- Cenário 01: redundância aplicada apenas no componente *storage* (Figura 34);

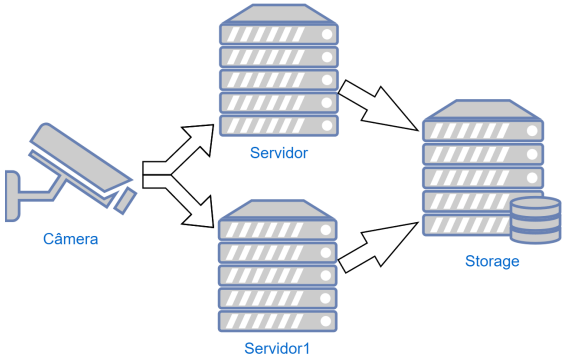
Figura 34 – Cenário 01



Fonte: O autor

- Cenário 02: redundância no componente servidor (*HW*, *OS* e *APP*) (Figura 35);

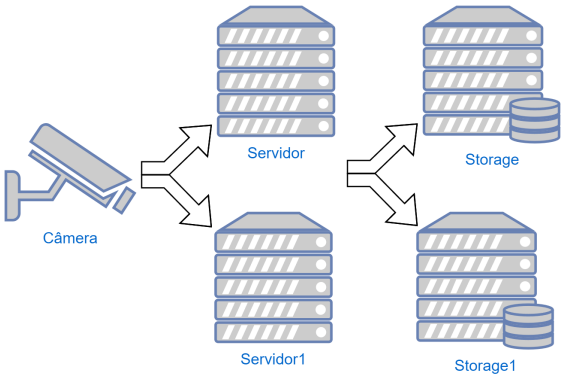
Figura 35 – Cenário 02



Fonte: O autor

- Cenário 03: redundância no *storage* e no servidor (Figura 36);

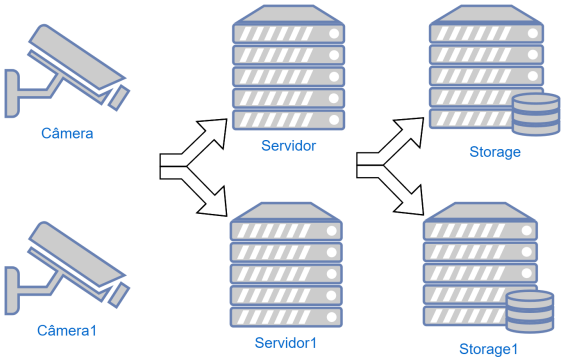
Figura 36 – Cenário 03



Fonte: O autor

- Cenário 04: redundância em todos os componentes do sistema (Figura 37.

Figura 37 – Cenário 04

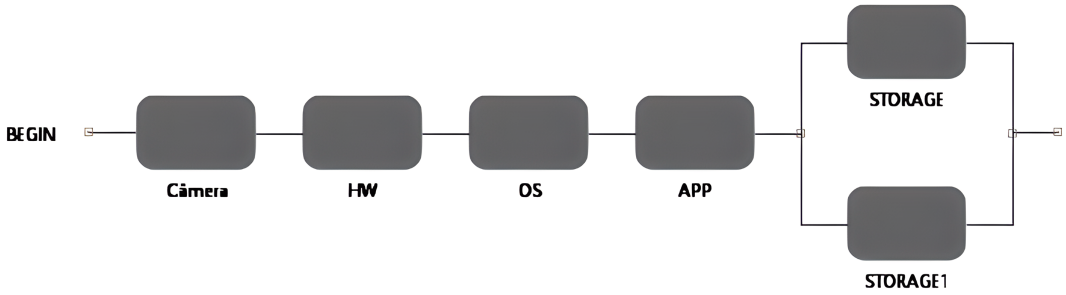


Fonte: O autor

6.4.2 Disponibilidade com *hot standby*

O cenário 01 conta com a redundância *hot standby* no componente *storage*. A Figura 38 representa um modelo RBD para esse cenário, onde os blocos *STORAGE* e *STORAGE1* estão em paralelo. Nesse cenário, para que o sistema esteja funcionando é necessário que pelo menos um dos *storage* esteja ativo e os demais componentes do sistema também estejam em funcionamento.

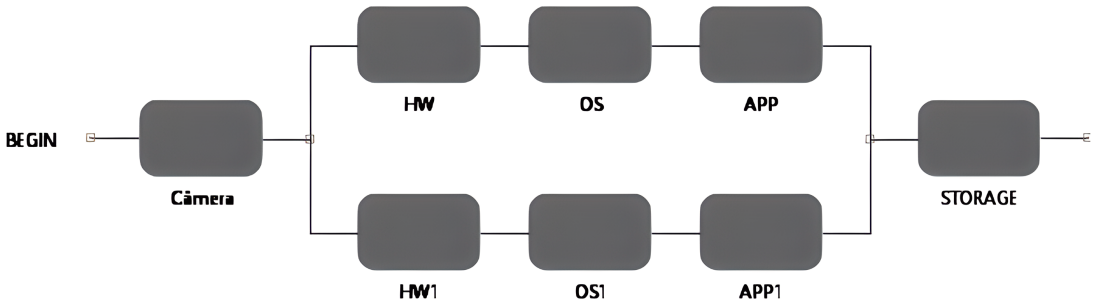
Figura 38 – Arquitetura com redundância *hot standby* no *storage*



Fonte: o autor

O cenário 02, considera a redundância aplicada apenas no servidor (Figura 39). Nesse caso, um servidor é representado pelos componentes *HW*, *OS* e *APP* agrupados em série. Dessa forma, o modelo RBD para representar esse ambiente tem os 2 conjuntos de blocos (*HW*, *OS*, e *APP*), que representam cada servidor, em paralelo, indicando que o sistema estará funcional mesmo na ocorrência de uma falha de um dos componentes de um servidor.

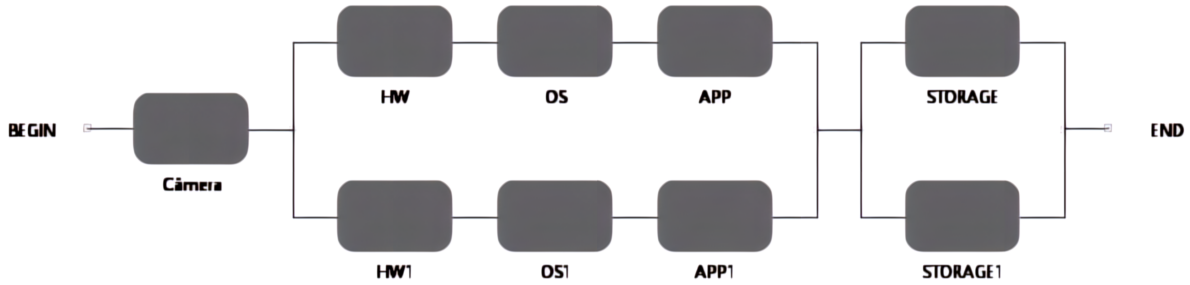
Figura 39 – Arquitetura com redundância *hot standby* no servidor



Fonte: o autor

No cenário 03, a redundância *hot-standby* é aplicada nos componentes do servidor (*HW*, *OS*, *APP*) e no *storage* simultaneamente. A Figura 40 representa o modelo desse cenário. Nesse cenário 03 foi possível diminuir o tempo de inatividade do ambiente para 1,73 horas. Isso corresponde a uma disponibilidade em número de noves do sistema é de 3,7035.

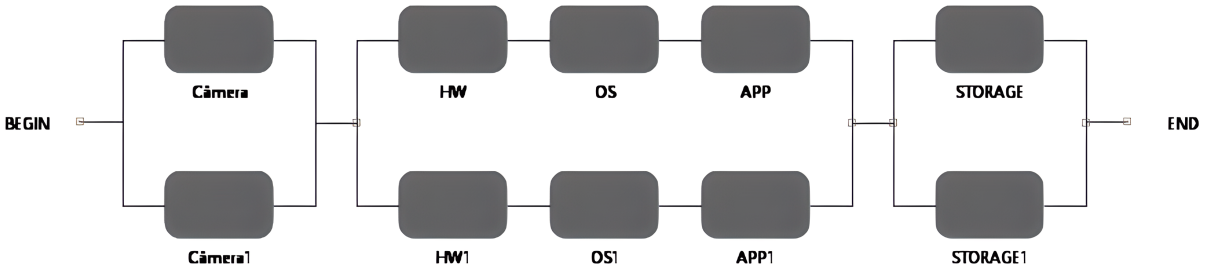
Figura 40 – Arquitetura com redundância *hot standby* no servidor e no *storage*



Fonte: o autor

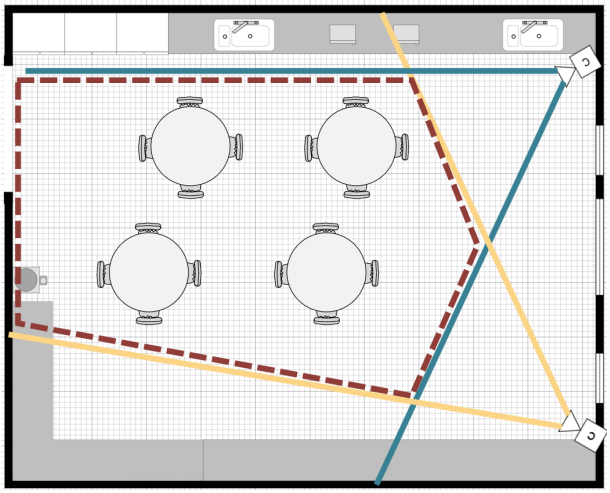
Para o cenário 04, temos a redundância *hot standby* aplicada em todos os componentes, incluindo a câmera, conforme pode ser visto na Figura 41. Nesse cenário, leva-se em consideração o fato que as câmeras também são redundantes, pois apesar de não estarem gravando o ambiente a partir do mesmo ângulo, o posicionamento pode captar imagens bastantes semelhantes, como podemos ver na Figura 42, onde a linha tracejada indica o perímetro monitorado em comum pelas duas câmeras.

Figura 41 – Arquitetura com redundância *hot standby* em todos os componentes



Fonte: o autor

Figura 42 – Exemplo de área monitorada por duas câmeras



Fonte: o autor

A Tabela 12 apresenta os resultados obtidos com os modelos e seus respectivos cenários. Ao analisar os resultados obtidos a partir da avaliação dos modelos RBDs, pode-se perceber que há mais ganho na disponibilidade quando há redundância em todos os componentes. Nesse caso, a disponibilidade alcançou o valor de 99,9992%, o que corresponde a uma indisponibilidade anual de apenas 3,85 minutos. Se for comparado os resultados obtidos com o tempo de indisponibilidade da arquitetura base temos uma redução de 99,8%. Assumindo o cenário 3 (redundância no servidor e no *storage*), pode-se obter o valor de 99,98% de disponibilidade e 1,73 horas de indisponibilidade, o que corresponde a uma melhora de 94,16% em relação à arquitetura base.

Tabela 12 – Resultados obtidos com estratégia de redundância *hot standby*

Cenário	Disponibilidade	Indisponibilidade	# de 9's	*uptime (h)*	*downtime (h)*
01	0.99716551849394	0.0028344815	2.5475263722740933	8740.96623581785	24.846534182149817
02	0.9992548703068616	0.00074512969	3.127768129785573	8759.281102620582	6.531667379419177
03	0.9998021073661751	0.00019789263	3.7035703712748003	8764.07808022333	1.734689776670881
04	0.9999926725802611	0.00000732741	5.135048929919577	8765.748539210483	0.06423078951801575

Fonte: O autor

6.4.3 Disponibilidade com *cold standby*

Essa subseção avalia, por meio de modelos SPN, a utilização da estratégia ativo-passivo onde um único componente ativo opera e fornece o serviço, e paralelamente, há um ou mais componentes passivos configurados para replicar o ambiente do componente ativo. Esses componentes passivos permanecem monitorando constantemente o componente principal, prontos para assumir o controle do serviço no momento em que for detectada uma falha. A Figura 43 apresenta o modelo SPN proposto para representar esse ambiente com redundância *cold standby*. Esse modelo faz uso de determinadas expressões, com as quais é possível obter os resultados considerando os cenários propostos em 6.4.1. As expressões, aplicadas no *Mercury tool* (MODCS, 2023; SILVA et al., 2015) para obtenção dos valores de disponibilidade conforme os cenários, estão apresentadas na Tabela 14.

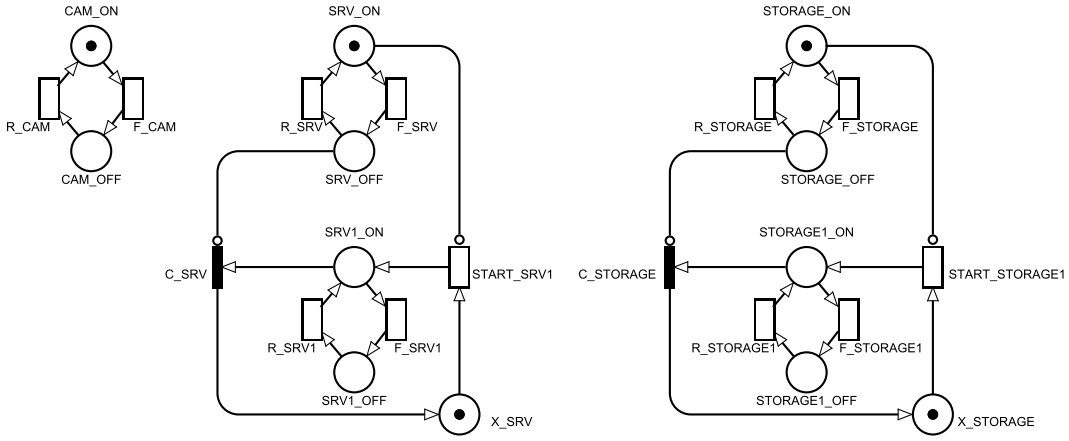
A Tabela 13 mostra os tempos utilizados nas transições temporizadas nos modelos SPN avaliados nesse estudo. Os tempos foram obtidos a partir do modelo RBD da arquitetura base após o processo de validação. Os valores aplicados no subsistema servidor, foram obtidos com a redução realizada nos blocos *HW*, *OS* e *APP*. Os tempos aplicados nas transições temporizadas START_STORAGE1 e START_SRV1 indicam o tempo necessário para a inicialização dos componentes redundantes.

Tabela 13 – MTTF e MTTR utilizados nas transições temporizadas nos modelos SPN

Transição temporizada	Tempo(h)
F_CAM	8760
F_SRV, F_SRV1	197,9
F_STORAGE, F_STORAGE1	43800
R_CAM	1,67
R_SRV, R_SRV1	0,52
R_STORAGE, R_STORAGE1	24
START_STORAGE1	0,025
START_SRV1	0,0166667

Fonte: o autor

Figura 43 – Modelo SPN - *Cold Standby*



Fonte: O autor

Tabela 14 – Expressões utilizadas para obtenção da disponibilidade em cada cenário

Cenário	Expressão
01	P{(#Cam_ON=1)AND(#SRV_ON=1)AND((#STORAGE_ON=1)OR(#STORAGE1_ON=1))}
02	P{(#Cam_ON=1)AND((#SRV_ON=1)OR(SRV1_ON=1))AND(#STORAGE_ON=1)}
03	P{(#Cam_ON=1)AND((#SRV_ON=1)OR(#SRV1_ON=1))AND((#STORAGE_ON=1)OR(#STORAGE1_ON=1))}
04	P{((#Cam_ON=1)OR(#Cam1_ON=1))AND((#SRV_ON=1)OR(#SRV1_ON=1))AND((#STORAGE_ON=1)OR(#STORAGE1_ON=1))}

Fonte: O autor

A Tabela 15 mostra os resultados obtidos nos cenários em estudo a partir da análise estacionária feita no *Mercury tool*. Além disso, essa tabela também apresenta uma comparação dos resultados obtidos com as estratégias de redundância *hot standby* e *cold standby*.

Tabela 15 – Comparação dos valores de disponibilidade entre *cold* e *hot standby*

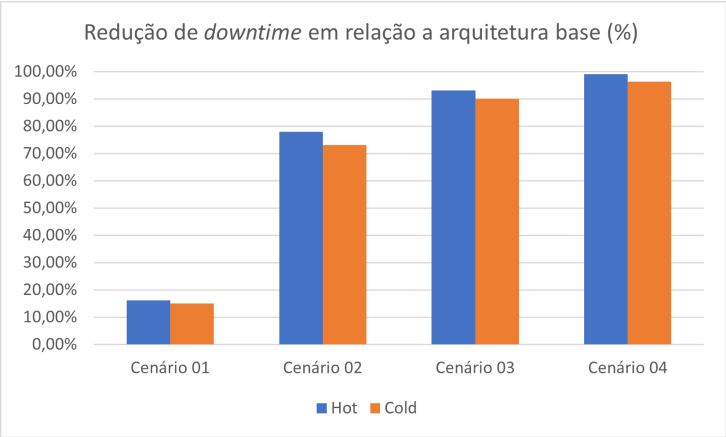
Cenário	hot standby		cold standby	
	Disponibilidade	Downtime anual(h)	Disponibilidade	Downtime anual(h)
01	0.99716551849394	24.84	0.9971045444653741	25.17
02	0.9992548703068616	6.53	0.9990846163027558	7.95
03	0.9998021073661751	1.73	0.9996579991741666	2.97
04	0.9999926725802611	0.06	0.9998736408649267	1.09

Fonte: O autor

A estratégia *cold standby*, aplicada no cenário 03, reduz em 88,20% e 62,64% o *downtime* anual em relação ao cenário 01 e 02, respectivamente. Já assumindo o cenário 04, onde todos os componentes se encontram redundantes, foi possível obter uma redução de 96,32% na indisponibilidade anual se comparado ao cenário da arquitetura base (6.4.1). Ao analisar os dados foi possível perceber uma redução de 94,50% no *downtime* quando aplicada a técnica *hot standby* no cenário 04, em relação ao mesmo cenário com a estratégia *cold standby*. Para os cenários 01, 02 e 03 a diferença percentual no *downtime* entre as estratégias de redundância são de 1,31%, 17,86% e 41.75%, respectivamente.

A Figura 44 apresenta os percentuais de redução de *downtime* quando os resultados são comparados aos obtidos na arquitetura base (Tabela 11). Com o cenário 04, em redundância *hot standby*, a redução de *downtime* é de 99,08% e com *cold standby* é de 96,32%. No cenário 01, a redução é de 16,17% e 15,05%, com redundância *cold* e *hot standby*, respectivamente.

Figura 44 – Percentuais de redução de *downtime* em relação à arquitetura base



Fonte: o autor

É notório que a disponibilidade é maior quando a estratégia *hot standby* é utilizada. No entanto, os custos envolvidos tendem a serem maior, pois nesta estratégia há um número maior de componentes em funcionamento. A definição pela estratégia a ser adotada deve considerar que o uso de *hot standby* é preferível quando a alta disponibilidade e a rápida recuperação são cruciais, independentemente do custo adicional associado a um segundo servidor ativo. Por outro lado, o *cold standby* pode ser uma escolha mais adequada para sistemas em que se deseja economizar recursos. Os resultados aqui apresentados visam auxiliar os projetistas de ambientes de videomonitoramento para a definição de

estratégias que garantam a alta disponibilidade de seus ambientes considerandos os níveis e componentes redundantes.

6.4.4 Redundância híbrida *cold* e *hot*

Essa seção considera cenários com a aplicação de estratégia de redundância híbrida, ou seja, alguns componentes com redundância *hot standby* e outros componentes em *cold standby*. Os cenários avaliados agora possuem as seguintes características:

- Cenário híbrido 01 (Figura 45): esse cenário tem um servidor redundante *cold* e um *storage* redundante *hot*;
- Cenário híbrido 02 (Figura 46): um servidor tem redundância *hot* e o *storage* tem redundância em *cold*.

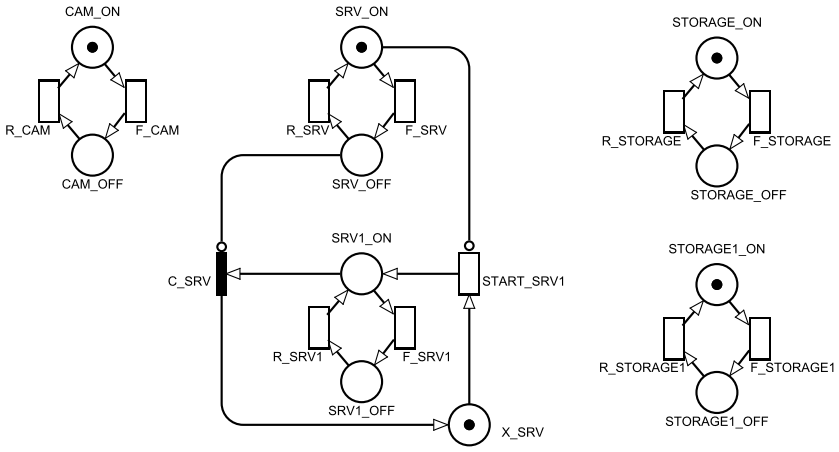
Os modelos SPN propostos estão representados nas Figuras 45 e 46, e as expressões utilizadas para obter os valores de disponibilidade são apresentadas na Tabela 16. Os tempos aplicados as transições estão na Tabela 13.

Tabela 16 – Expressões para os cenários híbridos

Cenário Híbrido	Expressão
01	P{(#Cam_ON=1)AND((#SRV_ON=1)OR(#SRV1_ON=1))AND((#STORAGE_ON=1)OR(#STORAGE1_ON=1))}
02	P{(#Cam_ON=1)AND((#SRV_ON=1)OR(#SRV1_ON=1))AND((#STORAGE_ON=1)OR(#STORAGE1_ON=1))}

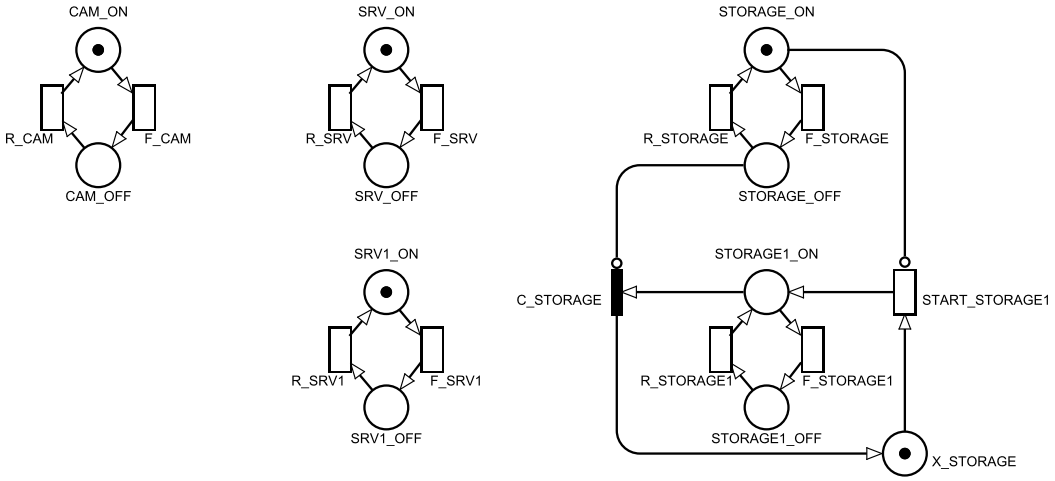
Fonte: O autor

Figura 45 – Modelo SPN - Cenário Híbrido 01



Fonte: O autor

Figura 46 – Modelo SPN - Cenário Híbrido 02



Fonte: O autor

Após realizar as análises estacionárias nos modelos com arquitetura com redundância híbrida temos as disponibilidades obtidas na Tabela 17. Os resultados obtidos indicam que há uma maior disponibilidade no cenário híbrido 02, quando para o servidor é aplicada a estratégia de redundância *hot standby* e para o *storage* em *cold standby*. A estratégia de aplicar redundâncias de diferentes tipos nos componentes do sistema, apresenta resultados expressivos em disponibilidade, tornando alternativas para implementação em sistemas de videomonitoramento.

Tabela 17 – Valores da disponibilidade nos cenários com redundância híbrida

Cenário híbrido	Disponibilidade	*Downtime* anual(h)
01	0.9996792887605509	2.78
02	0.9997988723315895	1.74

Fonte: O autor

6.5 Estudo de caso V

Esse estudo de caso tem duas finalidades: (i) identificar a importância de cada componente do sistema em relação à disponibilidade e (ii) propor e avaliar estratégias que aumentem a disponibilidade do ambiente. A análise de importância de componentes é uma abordagem valiosa para projetistas, fabricantes e especialistas em manutenção, pois permite identificar como a falha de um componente pode afetar o funcionamento do sistema. Isso auxilia na identificação dos pontos fracos do sistema e aponta quais componentes precisam de atenção ou melhorias(SI; LIU; CAI, 2009) .

Para estimar a importância dos componentes, será utilizado o método de importância para a disponibilidade (*Availability Importance*), conforme descrito no Capítulo 3. Por meio dessa abordagem, cada componente que compõe a infraestrutura do sistema será avaliado, destacando aqueles que têm maior impacto na disponibilidade. Além disso, será identificado quais os componentes, que ao terem sua disponibilidade aumentada, contribuirão significativamente para a disponibilidade do serviço.

A análise dos componentes foi realizada considerando a perspectiva da disponibilidade. Utilizando o MTTF e o MTTR de cada componente, utilizamos o *Mercury tool* para obter os índices de importância dos componentes em relação à disponibilidade. A Tabela 18 apresenta os índices de importância dos componentes, considerando os tempos de falha e reparo de cada componente analisado no modelo RBD da arquitetura base (Figura 17).

Tabela 18 – Importância dos componentes na disponibilidade da arquitetura base

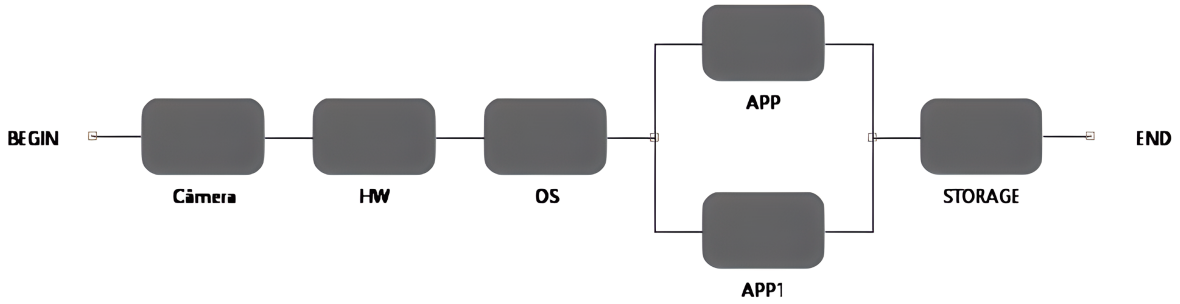
Componente	Índice de importância	Índice de Importância normalizado
APP	0.9987246147030486	1.0
STORAGE	0.9971658175591096	0.9984392122503134
OS	0.9969657730748231	0.9982389123064235
HW	0.9968097194155172	0.9980826593644128
Câmera	0.9968097194155172	0.9980826593644128

Fonte: o autor

Como podemos observar, o *hardware(HW)*, câmera e o sistema operacional (*OS*), possuem valores de índice de importância semelhantes. O componente que exerce maior impacto na disponibilidade do sistema é a aplicação *(APP)* (99,87%). Considerando os índices de importância para disponibilidade obtidos, serão aplicadas técnicas de redundância visando avaliar o impacto dessas estratégias para o aumento da disponibilidade do sistema. Dessa forma, foi possível obter os seguintes cenários.

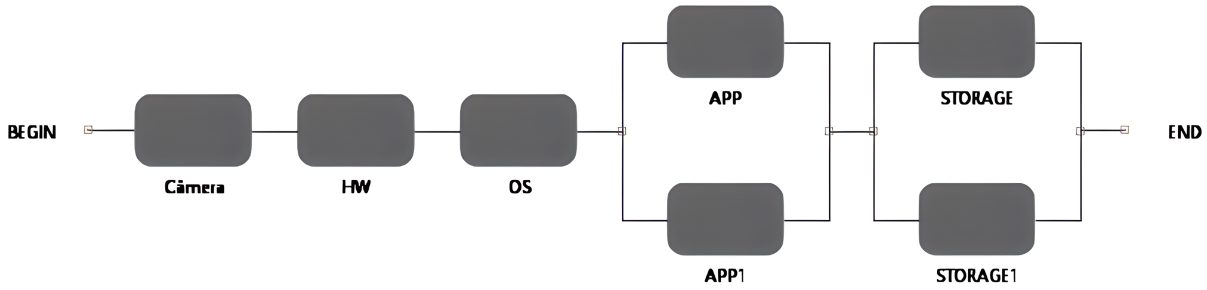
- Cenário 01: O sistema será formado por câmera, (*HW*), (*OS*), (*APP*), (*APP1*) e *storage* (Figura 47);
- Cenário 02: O sistema terá os seguintes componentes: câmera, (*HW*), (*OS*), (*APP*), (*APP1*), *storage* e *storage1* (Figura 48);
- Cenário 03: O sistema terá: câmera, (*HW*), (*OS*), (*OS1*), (*APP*), (*APP1*), *storage* e *storage1* (Figura 49);
- Cenário 04: O sistema formado por câmera, (*HW*), (*HW1*), (*OS*), (*OS1*), (*APP*), (*APP1*), *storage* e *storage1* (Figura 50);
- Cenário 05: O sistema composto por: câmera, câmera1, (*HW*), (*HW1*), (*OS*), (*OS1*), (*APP*), (*APP1*), *storage* e *storage1* (Figura 51);

Figura 47 – Arquitetura - Cenário 01



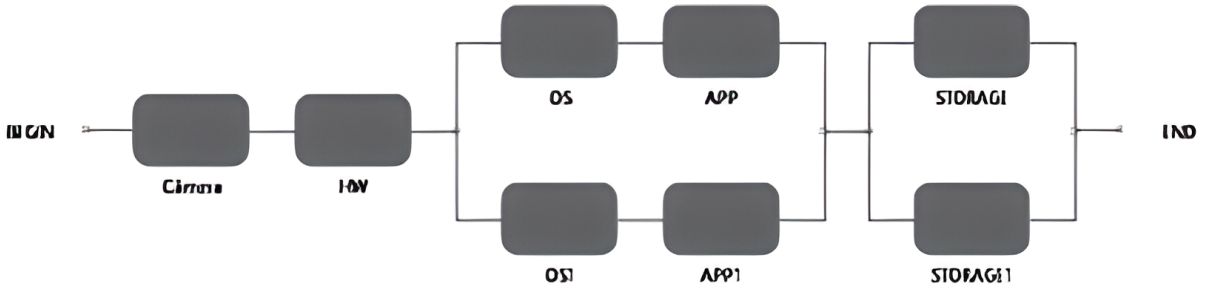
Fonte: o autor

Figura 48 – Arquitetura - Cenário 02



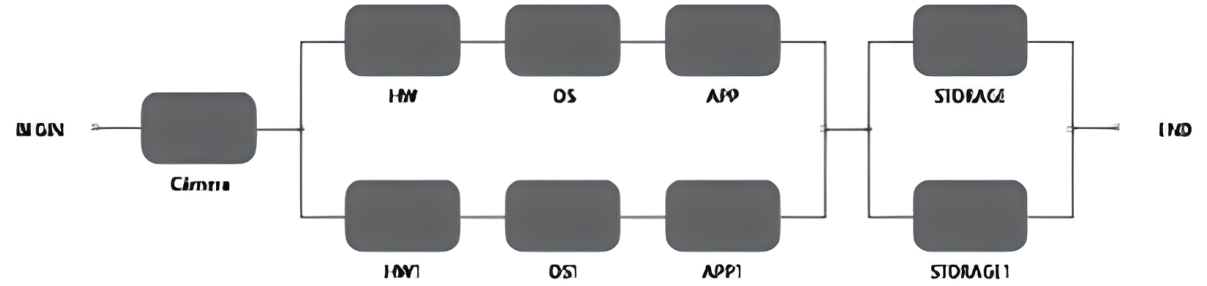
Fonte: o autor

Figura 49 – Arquitetura - Cenário 03



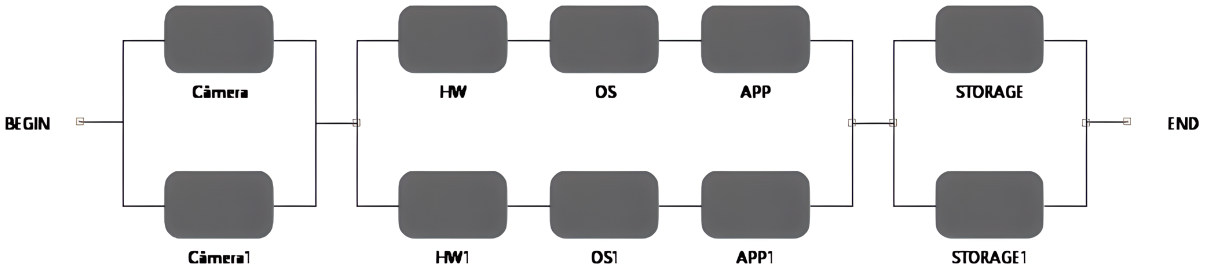
Fonte: o autor

Figura 50 – Arquitetura - Cenário 04



Fonte: o autor

Figura 51 – Arquitetura - Cenário 05



Fonte: o autor

A Tabela 19 apresenta o resultado da disponibilidade de cada cenário em comparação com os resultados obtidos na arquitetura base. Analisando os resultados de todos os cenários, percebe-se que, exceto o cenário da arquitetura base, o cenário 1 apresenta o pior resultado com 99,87% totalizando uma indisponibilidade de 11 horas por ano. Quando adicionada redundância nos componentes *storage* e *(APP)* (Cenário 02) a disponibilidade vai a 99,92% com indisponibilidade de 6,42 horas por ano. Os cenários 03 e 04 têm disponibilidade de 99,96% e 99,98%, respectivamente, com *downtime* de 3,39 horas e 1,73 horas. O cenário 05 é o que apresenta a maior disponibilidade (99,999%) e indisponibilidade de 3,6 minutos por ano, contudo necessita de redundância do componente câmera. Ainda assim, dependendo da criticidade do ambiente monitorado, a solução pode garantir alta disponibilidade.

Tabela 19 – Resultado da disponibilidade dos cenários avaliados

Cenário	Disponibilidade	# de 9's	*uptime(h)*	*downtime(h)*
Base	0.9966197245593511	2.471047909935344	8736.181908376244	29.630861623757458
1	0.9987201784826413	2.8928505924075685	8754.594094199816	11.218675800183895
2	0.9992671227205835	3.1349687420495744	8759.388504985249	6.424265014752107
3	0.9996125110976286	3.411740731113439	8762.41611483136	3.3966551686404247
4	0.9998021073661751	3.7035703712748003	8764.07808022333	1.734689776670881
5	0.9999926725802611	5.135048929919577	8765.748539210483	0.06423078951801575

Fonte: o autor

7 Conclusão

O avanço tecnológico tem impactado no cotidiano das pessoas em vários aspectos. Quando falamos de segurança, por exemplo, a tecnologia está presente por meio de sistemas de videomonitoramento, que aliado a crescente instalação de câmeras em ambiente públicos e privados vem auxiliando na prevenção de crimes, controle de acesso, identificação de comportamentos e objetos. Os atuais sistemas de videomonitoramento são digitais, demandam recursos computacionais para funcionarem adequadamente, com garantia de disponibilidade e confiabilidade dos dados armazenados. A falta de uma infraestrutura adequada poderá comprometer o desempenho.

Este trabalho apresenta cinco estudos de caso que tratam de desempenho, consumo elétrico e disponibilidade em um ambiente de videomonitoramento. No primeiro estudo, avaliamos a utilização de recursos computacionais e o consumo de energia do sistema com a alternância nos tipos de armazenamento, os dados são comparados visando obter resultados que possam nortear a escolha da melhor estratégia de captura e armazenamento, considerando os níveis de utilização de CPU, memória RAM, taxas de transmissão de dados na rede, além dos custos de energia elétrica. No estudo de caso II apresentamos equações de regressão linear que possibilitam estimar a quantidade de recursos computacionais e de custos com um número maior de câmeras em operação. As expressões foram obtidas a partir dos dados coletados durante a medição com várias cargas de trabalho. Os dois primeiros estudos exploram cenários por meio de experimentos sob diferentes cargas de trabalho. As análises tiveram como objetivo obter informações que auxiliem no dimensionamento para que o sistema atenda as demandas satisfatoriamente. Os resultados das medições e as projeções realizadas com regressão linear revelaram que o armazenamento replicado, tolerante a falhas, utilizará mais recursos em relação ao armazenamento distribuído, sendo o dobro em uso de rede e disco. Esse comportamento é explicado pelo fato das imagens serem enviadas a todos os elementos que formam o cluster de armazenamento. Com 100 câmeras em uso, a diferença nos custos de energia será de 6,17%, percebe-se que esse percentual se torna maior com a ampliação do número de fluxos de vídeo.

Os próximos estudos de caso, abordaram a disponibilidade, propondo mediante modelos RDB e SPN a busca por um nível satisfatório de disponibilidade do sistema. Inicialmente, temos um modelo RBD para calcular a disponibilidade da nossa arquitetura

base, formada por uma câmera, hardware, sistema operacional, aplicação e *storage*. Ainda nesse estudo aplicamos uma técnica de validação que fez uso de um injetor de falhas que possibilitou aferir, no ambiente real, os tempos de MTTF e de MTTR que após análises estatísticas foram comparados ao resultado obtido com o modelo de diagrama de blocos de confiabilidade (RBD) concluindo que o modelo proposto é equivalente ao sistema real. No estudo de caso IV, com o modelo da arquitetura base já validado é avaliado obtendo valores de disponibilidade de 99,66% com *downtime* de 29,63 horas por ano, diante desse resultado estratégias de redundância foram adotadas visando a melhoria da disponibilidade. Nesse sentido, avaliamos redundâncias do tipo *hot standby* e *cold standby*, para essa última utilizamos modelos estocásticos para obtenção dos valores de disponibilidade. Percebemos que com a aplicação de técnicas de redundância poderemos obter disponibilidade de até 99,999% e *downtime* anual de 0,06h, o que indica uma melhoria extremamente relevante se comparado aos valores obtidos na arquitetura base. No último estudo de caso, consideramos o índice de importância para a disponibilidade para propor por meio de melhorias pontuais o aumento da disponibilidade.

A abordagem utilizada, os modelos propostos, os resultados obtidos nessa pesquisa certamente contribuem para o planejamento de ambientes de videomonitoramento que utilizem recursos semelhantes aos avaliados neste trabalho.

7.1 Contribuições

A seguir são elencadas algumas contribuições deste trabalho:

- Proposição de uma estratégia para monitorar um sistema de videomonitoramento com armazenamento distribuído, possibilitando escalabilidade e disponibilidade dos dados armazenados;
- Avaliação de desempenho do sistema de videomonitoramento em diversos cenários de armazenamento, com estratégias de tolerância a falhas visando garantir a integridade dos dados armazenados.
- Avaliação do consumo elétrico do sistema a partir das diversas configurações de armazenamento;

- Estratégia de regressão linear para estimar recursos computacionais necessários e para dimensionar o consumo e custo de energia elétrica para funcionamento do sistema;
- Metodologia de avaliação da disponibilidade em modelos RBD e SPN com configurações com e sem redundâncias *hot standby* ou *cold standby*. Através desses modelos, foi possível avaliar métricas de disponibilidade e *downtime*;
- Avaliação do ambiente ao observar o índice de importância para a disponibilidade, sendo possível destacar quais componentes do sistema mais impactam a disponibilidade. Os resultados obtidos podem ser utilizados para a elaboração de técnicas que possam ampliar a disponibilidade dos componentes;
- Aplicação de metodologia de validação do modelo proposto, onde um ambiente real foi monitorado, dados coletados e analisados estatisticamente, com o intuito de corroborar o modelo elaborado, possibilitando confirmar que o mesmo representa um modelo real.

Dessa forma, os objetivos relacionados na Seção 1.2.1 foram alcançados, além das contribuições mencionadas acima, um artigo foi publicado com base nos resultados dessa pesquisa.

- Borges.I, Callou,G., Airton.F. **Performance evaluation and energy consumption of a video surveillance system with distributed storage**. 18th Iberian Conference on Information Systems and Technologies (CISTI). 2023. (QUALIS A4).

7.2 Limitações

As limitações deste trabalho são apresentadas abaixo:

- O ambiente computacional utilizado no desenvolvimento da pesquisa possui restrições nas configurações de hardware: CPU, memória RAM, unidades de armazenamento (HD) e número de câmeras;
- As regressões lineares propostas poderiam ter sido feitas com um número maior de amostras. No entanto, a estratégia proposta fornece uma estimativa relevante aos projetistas das infraestruturas dos ambientes de videomonitoramento, pois cada ponto da regressão linear representa a média de uma amostra de 30 experimentos.

7.3 Trabalhos Futuros

Nesta seção, listamos alguns aspectos que podem ser investigados em trabalhos futuros, como: avaliar outros itens de dependabilidade, considerando o ambiente proposto, tais como segurança, integridade e manutenibilidade; propor modelos para representar o desempenho e calcular e apresentar resultados na relação entre consumo energético, disponibilidade e desempenho; elaborar estudos que possibilitem computar os valores de probabilidade de falha das câmeras que possam impactar na disponibilidade; realizar análises de custos para fornecer informações e, com os índices de disponibilidade, permitir a escolha da melhor estrutura do ambiente; e construir um modelo de disponibilidade orientada à capacidade (COA).

Referências

ABUD, A. A.; GOFF, F. L.; AVOLIO, G. Performance evaluation of distributed file systems for the phase-ii upgrade of the atlas experiment at cern. *Journal of Physics: Conference Series*, IOP Publishing, v. 1525, n. 1, p. 012028, apr 2020. Disponível em: <https://dx.doi.org/10.1088/1742-6596/1525/1/012028>. 22, 26, 27

ACQUAVIVA, L. et al. Cloud distributed file systems: A benchmark of hdfs, ceph, glusterfs, and xtremefs. In: IEEE. *2018 IEEE Global Communications Conference (GLOBECOM)*. [S.l.], 2018. p. 1–6. 21, 26, 27

ADVIRKAR, S. et al. Smart surveillance system. *International Journal of Research in Engineering, Science and Management (IJERSM)*, v. 3, n. 3, p. 70–72, 2020. 24, 27

AGARWAL, P. K. et al. Application health monitoring for extreme-scale resiliency using cooperative fault management. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 32, n. 2, p. e5449, 2020. 62

AHMAD, M. et al. Energy efficient camera solution for video surveillance. *International Journal of Advanced Computer Science and Applications*, Science and Information (SAI) Organization Limited, v. 10, n. 3, 2019. 25, 26, 27

ALERJ. *Lei nº 7209, de 14 de janeiro de 2016*. 2016. Disponível em: <http://alerjln1.alerj.rj.gov.br/CONTLEI.NSF/c8aa0900025feef6032564ec0060dfff/0846c5d8ce1b399383257f3f0054e513?OpenDocument#:~:text=ALTERA%20A%20LEI%20N%C2%BA%203.162,DE%20IMAGENS%20NOS%20ESTABELECIMENTOS%20BANC%C3%81RIOS>. 70

ANDRADE, E. et al. Availability modeling and analysis of a disaster-recovery-as-a-service solution. *Computing*, Springer, v. 99, p. 929–954, 2017. 66

ANDREI, H. et al. Energy consumption analysis of security systems for a residential consumer. In: IEEE. *2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. [S.l.], 2019. p. 1–4. 18, 25, 26, 27

AVIZIENIS, A.; LAPRIE, J.-C. Dependable computing: From concepts to design diversity. *Proceedings of the IEEE*, IEEE, v. 74, n. 5, p. 629–638, 1986. 40

BANU, V. C. et al. Intelligent video surveillance system. In: *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)*. [S.l.: s.n.], 2017. p. 208–212. 18

BAUER, E. *Beyond redundancy how geographic redundancy can improve service availability and reliability of computer-based systems*. [S.l.]: Wiley-IEEE Press, 2011. 45

BAUER, E. *Design for reliability: information and computer-based systems*. [S.l.]: John Wiley & Sons, 2011. 40, 41

BAUER, E.; ADAMS, R. *Reliability and availability of cloud computing*. [S.l.]: John Wiley & Sons, 2012. 40

BORGES, I.; CALLOU, G.; SILVA, F. A. Performance evaluation and energy consumption of a video surveillance system with distributed storage. In: *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2023. p. 1–6. 36

CALLOU, G. et al. Models for dependability and sustainability analysis of data center cooling architectures. In: IEEE. *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*. [S.l.], 2012. p. 1–6. 37, 38

CHUNG, C. A. *Simulation modeling handbook: a practical approach*. [S.l.]: CRC press, 2003. 73

CURRAL, J. Statistics packages: A general overview. *Universidade de Glasgow*, p. 32, 1994. 39

DAI, J.; LIU, N. The data platform for large-scale video surveillance systems. In: SPRINGER. *International Conference on Applications and Techniques in Cyber Intelligence ATCI 2019: Applications and Techniques in Cyber Intelligence 7*. [S.l.], 2020. p. 1556–1561. 23, 26, 27

DANTAS, J. et al. An availability model for eucalyptus platform: An analysis of warm-standy replication mechanism. In: IEEE. *2012 IEEE international conference on Systems, man, and cybernetics (SMC)*. [S.l.], 2012. p. 1664–1669. 66

ELHARROUSS, O.; ALMAADEED, N.; AL-MAADEED, S. A review of video surveillance systems. *Journal of Visual Communication and Image Representation*, Elsevier, v. 77, p. 103116, 2021. 18

GONÇALVES, C. F. et al. Stochastic models for planning vle moodle environments based on containers and virtual machines. *Revista de Informática Teórica e Aplicada*, v. 29, n. 2, p. 63–83, 2022. 66

GRAY, J.; SIEWIOREK, D. P. High-availability computer systems. *Computer*, IEEE, v. 24, n. 9, p. 39–48, 1991. 40

HAIM, R. B.; ROTTENSTREICH, O. Reliable and time-efficient virtualized function placement. In: IEEE. *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*. [S.l.], 2020. p. 71–78. 44

HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. *Computer*, IEEE, v. 30, n. 4, p. 75–82, 1997. 54

INTELLIGENCE, M. *MERCADO DE SEGURANÇA ELETRÔNICA - CRESCIMENTO, TENDÊNCIAS, IMPACTO DO COVID-19 E PREVISÕES (2023-2028)*. 2020. Disponível em: <https://www.mordorintelligence.com/pt/industry-reports/electronic-security-market>. 16

IPVM. *Frame Rate Guide for Video Surveillance*. 2021. Disponível em: <https://ipvm.com/reports/frame-rate-surveillance-guide>. 72

IPVM. *The Authority On Physical Security Technology*. 2023. Disponível em: <https://ipvm.com/about?from=quick-links>. 72

ISARSOFT. *What is a Video Management System (VMS)?* 2023. Disponível em: <https://www.isarsoft.com/knowledge-hub/vms>. 33, 34

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: john wiley & sons, 1990. 63

JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling, NY: Wiley*. [S.l.: s.n.], 1991. ISBN 0471503361. 82

JAMMAL, M. et al. Ace: Availability-aware cloudsim extension. *IEEE Transactions on Network and Service Management*, IEEE, v. 15, n. 4, p. 1586–1599, 2018. 62

KARIMAA, A. Video surveillance in the cloud: Dependability analysis. In: *Proceedings of the fourth international conference on dependability, DEPEND*. [S.l.: s.n.], 2011. p. 92–95. 18

KAVALIONAK, H. et al. Distributed video surveillance using smart cameras. *Journal of Grid Computing*, Springer, v. 17, p. 59–77, 2019. 24, 26, 27

KIRBY, A. et al. Storage and file structure of a bioinformatics cloud architecture. In: IEEE. *2019 IEEE Cloud Summit*. [S.l.], 2019. p. 110–115. 22, 26, 27

KUO, W.; ZUO, M. J. *Optimal reliability modeling: principles and applications*. [S.l.]: John Wiley & Sons, 2003. 53

LAVENBERG, S. S. *Computer Performance Modeling Handbook*. USA: Academic Press, Inc., 1983. ISBN 0124387209. 82

LILJA, D. J. *Measuring computer performance: a practitioner's guide*. [S.l.]: Cambridge university press, 2005. 36, 37, 40

MA, S. et al. A retrieval optimized surveillance video storage system for campus application scenarios. *Journal of Electrical and Computer Engineering*, Hindawi, v. 2018, 2018. 25, 27

MACIEL, P. R. et al. Dependability modeling. In: *Performance and dependability in service computing: concepts, techniques and research directions*. [S.l.]: IGI Global, 2012. p. 53–97. 40, 42, 45

MACIEL, P. R. M. *Performance, reliability, and availability evaluation of computational systems, Volume 1: Reliability, availability modeling, measuring, and data analysis*. [S.l.]: CRC Press, 2023. 43, 44, 50, 85, 86

MACIEL, P. R. M. *Performance, reliability, and availability evaluation of computational systems, Volume 2: Reliability, availability modeling, measuring, and data analysis*. [S.l.]: CRC Press, 2023. 37, 40, 41, 46, 48, 66

MACIEL, P. R. M.; DANTAS, J. R.; JÚNIOR, R. d. S. M. Markov chains and stochastic petri nets for availability and reliability modeling. In: *Reliability Engineering*. [S.l.]: CRC Press, 2019. p. 127–151. 45

MATOS, M. A. Manual operacional para a regressão linear. *Faculdade de Engenharia da Universidade do Porto*, v. 63, 1995. 39

MATOS, R. et al. Sensitivity analysis of a hierarchical model of mobile cloud computing. *Simulation Modelling Practice and Theory*, Elsevier, v. 50, p. 151–164, 2015. 66

MELO, R. et al. Redundancy mechanisms applied in cloud computing infrastructures. In: IEEE. *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.], 2018. p. 1–6. 45, 89

MELO, R. M. D. et al. Redundant vod streaming service in a private cloud: Availability modeling and sensitivity analysis. *Mathematical Problems in Engineering*, Hindawi, v. 2014, 2014. 66

MODCS. *Mercury tool*. 2023. Disponível em: <https://www.modcs.org/?page_id=2330>. 89, 95

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989. 49

NEAL, D.; RAHMAN, S. Video surveillance in the cloud? *arXiv preprint arXiv:1512.00070*, 2015. 33

NETO, M. C. et al. Authentication and availability in a flexible and adaptable distributed file system. In: IEEE. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.], 2017. p. 1–6. 28

PAREKH, A.; GAUR, U. K.; GARG, V. Analytical modelling of distributed file systems (glusterfs and cephfs). In: SPRINGER. *Reliability, Safety and Hazard Assessment for Risk-Based Technologies: Proceedings of ICRESH 2019*. [S.l.], 2020. p. 213–222. 22, 26, 27

PEREIRA, P. et al. Analytical models for availability evaluation of edge and fog computing nodes. *The Journal of Supercomputing*, Springer, v. 77, p. 9905–9933, 2021. 62

PEREIRA, P. et al. Availability model for edge-fog-cloud continuum: an evaluation of an end-to-end infrastructure of intelligent traffic management service. *The Journal of Supercomputing*, v. 78, 02 2022. 66, 83

PETRI, C. A. *Kommunikation mit Automaten. Darmstadt, Technische Universität*. Tese (Doutorado) — Dissertation, 1962. 49

RABENSTEIN, B.; VOLZ, J. Prometheus: A {Next-Generation} monitoring system (talk). 2015. 38

RAJAVEL, R. et al. Iot-based smart healthcare video surveillance system using edge computing. *Journal of ambient intelligence and humanized computing*, Springer, p. 1–13, 2022. 24, 26, 27

RIABKO, A. V. et al. Cluster fault tolerance model with migration of virtual machines. 2023. 44

SAHNER, R. A.; TRIVEDI, K. S. Reliability modeling using sharpe. *IEEE Transactions on Reliability*, IEEE, v. 36, n. 2, p. 186–193, 1987. 48

SCHNEEWEISS, W. G. *Boolean functions: with engineering applications and computer programs*. [S.l.]: Springer Science & Business Media, 2012. 42

SELVAGANESAN, M.; LIAZUDEEN, M. A. An insight about glusterfs and its enforcement techniques. In: IEEE. *2016 International Conference on Cloud Computing Research and Innovations (ICCCRI)*. [S.l.], 2016. p. 120–127. 29

SI, S.; LIU, F.; CAI, Z. Failure importance analysis models based on bayesian network. *2009 16th International Conference on Industrial Engineering and Engineering Management*, p. 151–154, 2009. 100

SILVA, B. et al. Mercury: An integrated environment for performance and dependability evaluation of general systems. In: *Proceedings of industrial track at 45th dependable systems and networks conference, DSN*. [S.l.: s.n.], 2015. p. 1–4. 89, 95

SILVA, V. C. O. da et al. Energy consumption in mobile devices considering communication protocols. *Advances in Information Sciences and Service Sciences*, Citeseer, v. 6, n. 5, p. 1, 2014. 66

SIVASUBRAMANIAM, A.; RAMACHANDRAN, U.; VENKATESWARAN, H. A comparative evaluation of techniques for studying parallel system performance. *Georgia Institute of Technology*, p. 1–24, 1994. 62

SOUSA, E. et al. Dependability evaluation of cloud infrastructures. In: IEEE. *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.], 2014. p. 1282–1287. 66

SRINIVASAN, S. et al. Covid-19 monitoring system using social distancing and face mask detection on surveillance video datasets. In: *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. [S.l.: s.n.], 2021. p. 449–455. 18, 25, 27

STANIMIROVIC, U. *What is H.264 video codec and how it's shaping the online video world*. 2021. Disponível em: <https://www.brid.tv/what-is-h264-video-codec/>. 72

SULTANA, T.; WAHID, K. A. Iot-guard: Event-driven fog-based video surveillance system for real-time security management. *IEEE Access*, v. 7, p. 134881–134894, 2019. 24, 26, 27

TRIVEDI, K. S. et al. Reliability analysis techniques explored through a communication network example. North Carolina State University. Center for Advanced Computing and Communication, 1996. 46

WAGDY, M.; BABULAK, E.; AL-DABASS, D. Network function virtualization over cloud-cloud computing as business continuity solution. *Intechopen, Published: July 14th*, 2021. 44

WANG, D.; TRIVEDI, K. S. Computing steady-state mean time to failure for non-coherent repairable systems. *IEEE Transactions on reliability*, IEEE, v. 54, n. 3, p. 506–516, 2005. 42

WANG, L. et al. Design and implementation of fault injector for hybrid heterogeneous storage system. In: *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. [S.l.: s.n.], 2019. p. 10–15. 54

XIAO, Y. et al. An extra-parity energy saving data layout for video surveillance. *Multimedia Tools and Applications*, Springer, v. 77, p. 4563–4583, 2018. 23, 26, 27

YU, J. et al. A distributed storage system for robust, privacy-preserving surveillance cameras. In: IEEE. *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.], 2020. p. 1195–1196. 21, 26, 27

ZHAO, Z. F.; CUI, X. J.; ZHANG, H. Q. Cloud storage technology in video surveillance. In: TRANS TECH PUBL. *Advanced Materials Research*. [S.l.], 2012. v. 532, p. 1334–1338. 23, 27

ZIADE, H. et al. A survey on fault injection techniques. *Int. Arab J. Inf. Technol.*, v. 1, n. 2, p. 171–186, 2004. 54

ZONEMINDER. *ZoneMinder - Documentation/Features*. 2023. Disponível em: <https://zoneminder.com/features/>. 35

APÊNDICE

APÊNDICE A – Injetor de Falhas

Este apêndice contém informações sobre o injetor de falhas utilizado para acelerar o processo de falha em componentes, com ele também é possível coletar dados para avaliar a disponibilidade de um serviço. O injetor de falhas foi desenvolvido em Python e é formado por *main.py* (A.1), *component.py* (A.2) e *monitor.py* (A.3).

Para executar é necessário criar um ambiente virtual Python. Esta etapa é importante porque é preciso instalar alguns pacotes. Com o ambiente virtual disponível, deve ser instalado os pacotes usados pelo injetor de falhas. Em seguida, deve ser editado o arquivo *main.py* para definir as seguintes variáveis:

- IP_ADDRESS é a variável que irá armazenar o endereço IP do servidor onde as falhas serão injetadas. Usamos esse endereço para conectar via SSH;
- PASSWORD é a variável que vai armazenar a senha do servidor, para poder conectar via SSH;
- NETWORK_INTERFACE é a variável que irá armazenar a interface de rede que estará "up" e "down" durante a injeção de falhas;
- MONITORING_INTERVAL é a variável que armazenará o intervalo de monitoramento. O monitor coletará dados sobre o serviço a cada quantidade de segundos definida nesta variável;
- MONITORING_FILE_NAME é a variável que armazenará o nome do arquivo. Este arquivo armazenará os dados coletados.

Depois dessas etapas o injetor pode ser executar a partir do comando:

```
1   (fault-injection) root@root:~$ python simulator/main.py
```

Com isso, serão solicitados o número de componentes, em seguida o nome do componente 01, posteriormente dados de MTTR e MTTF do componente 01, e assim por diante, até o número de componentes informado.

Os valores de MTTF e o MTTR devem ser informados em segundos.

O injetor, com base nos valores de MTTR e MTTF, gerará valores aleatórios considerando a distribuição exponencial, e "injetará" a falha e o reparo, nos tempos definidos.

Para interromper a execução do injetor de falhas, pressione CTRL + C.

A.1 main.py

```
import signal
from time import sleep
from utils.monitor import Monitor
from utils.component import Component

# Fault injection
IP_ADDRESS = "10.0.0.230"
PASSWORD = "ubuntu"
NETWORK_INTERFACE = "eno1"

# Monitor
MONITORING_INTERVAL = 10
MONITORING_FILE_NAME = "dados_monitoramento.txt"

def handler(signum, frame):
    res = input("\nCTRL + C was pressed. Do you really want
        to exit? y/n ")
    if res == 'y':
        exit(1)

signal.signal(signal.SIGINT, handler)

if __name__ == '__main__':
    monitor = Monitor(MONITORING_INTERVAL,
       MONITORING_FILE_NAME, IP_ADDRESS)
    components = []

    num_components = input("How many components? ")

    for i in range(int(num_components)):
```

```
29          name = input("\nEnter the name of the component ["
               + str(i+1) + "]: ")
30          mttf = input("Enter MTTF (FAIL) in seconds of the
               component [" + str(i+1) + "]: ")
31          mttr = input("Enter MTTR (REPAIR) in seconds of the
                component [" + str(i+1) + "]: ")
32
33          components.append(
34              Component(
35                  name,
36                  mttf,
37                  mttr,
38                  IP_ADDRESS,
39                  PASSWORD,
40                  NETWORK_INTERFACE
41              )
42          )
43
44      for component in components:
45          component.run()
46
47      monitor.start_monitor()
48
49      print("Monitor and Fault injection RUNNING...")
```

A.2 component.py

```
1  import multiprocessing as mp
2  import subprocess
3  from time import sleep
4  import numpy as np
5
6  class Component:
7      def __init__(self, name, mttf, mttr, ip_address,
           password, network_interface):
8          self.name = name
9          self.mttf = mttf
10         self.mttr = mttr
11         self.ip_address = ip_address
12         self.password = password
13         self.network_interface = network_interface
14
15     def inject_fault(self):
16         while True:
17             random_mttf = np.random.exponential(float(self.
                   mttf))
18             # time is in second
19             print("\n" + self.name, "is  UP, next FAULT
                   will happen in:", random_mttf, "seconds")
20             sleep(random_mttf)
21
22             random_mttr = np.random.exponential(float(self.
                   mttr))
23             # to check the interface in ubuntu, type on
                   terminal: ip addr
24             ssh_command = "sshpass -p " \
25                 + self.password \
```

```
26                  + " ssh -f -o StrictHostKeyChecking=no " \
27                  + self.ip_address \
28                  + " -l root \"ip link set " + self.
                        network_interface + " down && sleep " \
29                  + str(int(random_mttr)) + " && ip link set
                        " + self.network_interface + " up &>/dev
                        /null &\""
30
31              subprocess.Popen(ssh_command, shell=True,
                    stdout=subprocess.PIPE)
32
33              print("\n" + self.name, "is DOWN, next REPAIR
                    will finish in:", random_mttr, "seconds")
34              sleep(random_mttr)
35
36      def run(self):
37          self.process = mp.Process(target=self.inject_fault)
38          self.process.daemon = True
39          self.process.start()
40
41      def terminate(self):
42          self.process.kill()
```

A.3 monitor.py

```
from time import sleep
import subprocess

class Monitor:
    def __init__(self, interval, file_name, ip_address):
        self.interval = interval
        self.file_name = file_name
        self.ip_address = ip_address
    def start_monitor(self):
        while True:
            command = "ping -w 1 " \
                +  self.ip_address \
                + " | grep received | awk '{print $4}'"
            output = subprocess.Popen(command, shell=True,
               stdout=subprocess.PIPE).stdout.read().rstrip
               ()

            if output.decode('UTF-8')=='0':
                with open(self.file_name, "a") as myfile:
                    print("\n[MONITOR] system is DOWN")
                    myfile.write("DOWN" + "\n")

            elif output.decode('UTF-8')=='1':
                with open(self.file_name, "a") as myfile:
                    print("\n[MONITOR] system is UP")
                    myfile.write("UP" + "\n")
            else:
                print("output is not known")
            sleep(self.interval)
```