Pós-Graduação em Ciência da Computação

Eric Rodrigues Borba

MODELAGEM DE DESEMPENHO E DISPONIBILIDADE PARA SISTEMAS DE ARMAZENAMENTO HÍBRIDOS

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE

2017

Eric Rodrigues Borba

MODELAGEM DE DESEMPENHO E DISPONIBILIDADE PARA SISTEMAS DE ARMAZENAMENTO HÍBRIDOS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Prof. Dr. Eduardo Antônio Guimarães Tavares*

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

B726m Borba, Eric Rodrigues
Modelagem de desempenho e disponibilidade para sistemas de armazenamento híbridos / Eric Rodrigues Borba. – 2017.
105 f.:il, fig., tab.

Orientador: Eduardo Antônio Guimarães Tavares.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Avaliação de desempenho. 2. Computação em nuvem. I. Tavares, Eduardo Antônio Guimarães (orientador). II. Título.

004.029 CDD (23. ed.) UFPE- MEI 2017-155

Eric Rodrigues Borba

Modelagem de Desempenho e Disponibilidade para Sistemas de Armazenamento Híbridos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciência da Computação

Aprovado em: 14/06/2017

BANCA EXAMINADORA

Prof. Dr. Paulo Romero Martins Maciel
Centro de Informática/UFPE

Prof. Dra. Erica Teixeira Gomes de Sousa
Departamento de Estatística e Informática/UFRPE

Prof. Dr. Eduardo Antônio Guimarães Tavares
Centro de Informática/UFPE
(**Orientador**)

À minha família.

Agradecimentos

Agradeço primeiramente a Deus por me permitir vivenciar e superar os obstáculos encontrados durante este percurso.

Agradeço ao meu orientador, Prof. Eduardo Tavares, que acreditou em minha capacidade e me ajudou a superar minhas deficiências. Agradeço pela sua orientação e paciência, compartilhando seu conhecimento, tanto em sala de aula, quanto no decorrer do trabalho.

Agradeço aos meus pais, Esdras e Marily, pois foi a sua dedicação sem limites, o esforço incessante, o investimento em minha educação, e credibilidade cega em minha capacidade, além do apoio constante, que me permitiram chegar até aqui.

Agradeço à minha esposa, Rosiely Borba, a qual desde o momento que conheci transformou minha vida me dando o apoio emocional necessário para alcançar mais este objetivo; a conclusão de mais esta etapa em minha vida (do curso e deste trabalho) só foi possível graças a sua compreensão e carinho em todos os momentos, sem exceção (principalmente os mais dolorosos e duradouros).

Agradeço aos meus irmãos, Cindel e Ezinho, que vibraram comigo a cada etapa conquistada como se fossem deles (e no final das contas, é deles também, afinal sempre fizeram parte disto).

Agradeço aos meus familiares, tios, primos, avós, sogros, cunhados, que acompanharam minha luta desde o início (mesmo que as vezes de longe), acreditaram em mim, e me incentivaram até o final.

Agradeço ao Prof. Paulo Maciel pelos ensinamentos em sala de aula e aos colegas do MoDCS pela troca de conhecimento nos WMoDCS.

Agradeço ao meu amigo, Rômulo Araújo, pelos conselhos valiosos que me foram dados para atingir este objetivo. Também agradeço aos amigos realizados durante o curso, e que foram fundamentais para troca de conhecimento, bem como em momentos (breves) de lazer: Érico, Iúre, Jonas, Djalma. Aos alunos de iniciação que também me incentivaram: Ihago e Nilson.

À Universidade Federal de Pernambuco (UFPE), pelo suporte acadêmico durante esses anos.

Aos meus colegas de escala da CBTU/METROREC, do Centro de Controle Operacional, por compreenderem a situação e me apoiarem várias vezes.

Por fim, agradeço aos demais colegas, amigos e parentes que contribuíram direta ou indiretamente para a realização deste objetivo.

O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.

—JOSÉ DE ALENCAR

Resumo

O aperfeiçoamento de sistemas computacionais pode vir a ser limitado de acordo com a eficiência dos dispositivos de armazenamento. Deste modo, a substituição de dispositivos de disco rígido (*hard disk drives*) por dispositivos de estado sólido (*solid-state drives*) pode ser uma forma efetiva para melhorar o desempenho dos sistemas, tanto para computadores pessoais, quanto para *data centers*. Entretanto, o alto custo por *gigabyte* e um reduzido tempo de vida, dificulta a substituição por completo dos *hard disk drives* (HDDs) por *solid-state drives* (SSDs). Para mitigar estas questões, diversas arquiteturas e políticas de armazenamento têm sido concebidas baseadas em sistemas de armazenamento híbridos, todavia, não propõem modelos de desempenho e disponibilidade para melhor avaliar diferentes arquiteturas. Esta dissertação apresenta uma abordagem baseada em modelos estocásticos para a modelagem de desempenho e disponibilidade de sistemas de armazenamento híbridos, usando redes de *Petri* estocásticas (SPN) e diagrama de blocos de confiabilidade (RBD). Os modelos propostos possibilitam representar operações de escrita, leitura e *mixed* (escrita e leitura), e eles podem estimar tempo médio de resposta, vazão e disponibilidade. Inicialmente, análises estatísticas (teste t emparelhado) validam os modelos SPN concebidos, para um cenário com quatro diferentes dispositivos de armazenamento (HDDs) e cinco tamanhos diferentes de objetos, com o auxílio do *DiskSim*, que é uma ferramenta de simulação de sistemas de armazenamento. Em seguida, um planejamento de experimento (DoE) define os fatores relevantes para uma arquitetura baseada na plataforma de computação em nuvem adotada (*OpenStack Swift*). Posteriormente, experimentos de desempenho e disponibilidade, considerando esta plataforma de computação em nuvem, proporcionam uma comparação entre tecnologias de armazenamento tradicionais (HDD e SSD) e diferentes soluções de armazenamento híbrido. Além disso, este trabalho propõe a avaliação conjunta do desempenho (tempo médio de resposta), disponibilidade (para o cálculo do *downtime*) e custo, com o intuito de fornecer uma análise a respeito de diferentes tecnologias e arquiteturas de armazenamento, bem como políticas de leitura e escrita. Os resultados demonstram a viabilidade da abordagem proposta, bem como os benefícios por adotar dispositivos de armazenamento híbridos associados a uma política de armazenamento adequada.

Palavras-chave: Avaliação de Desempenho. Disponibilidade. Diagrama de Blocos de Confiabilidade. Armazenamento Híbrido. Redes de Petri Estocásticas. Computação em Nuvem.

Abstract

Improvements in computational systems may be constrained by the efficiency of storage drives. Therefore, replacing hard disk drives (HDD) with solid-state drives (SSD) may also be an effective way to improve system performance, both in personal computers and data centers. However, the higher cost per gigabyte and reduced lifetime of SSDs hinder a thorough replacement. To mitigate these issues, several architectures and storage policies have been conceived based on hybrid storage systems, but performance and availability models have not been proposed to better assess such different architectures. This dissertation presents an approach based on stochastic models for performance and availability modeling of hybrid storage systems, using stochastic Petri nets (SPN) and reliability block diagrams (RBD). The proposed models can represent write, read and mixed operations, and they may estimate response time, throughput and availability. Initially, statistical analyses (paired t-test) validate SPN models conceived, for a scenario with four different storage devices (HDDs) and five different object sizes, with the DiskSim support, which is a storage system simulation tool. Next, a design of experiment (DoE) defines the relevant factors for a storage architecture based on the cloud computing platform adopted (OpenStack Swift). Hereafter, performance and availability experiments, considering this cloud computing platform, provide a comparison between traditional storage technologies (HDD and SSD) and different hybrid storage solutions. Also, this work proposes a joint evaluation of both performance (response time), availability (for calculating downtime), and cost, in order to provide an analysis of different technologies and storage architectures as well as read/write policies. The results show the feasibility of the proposed approach as well as the improvements by adopting hybrid storage devices associated with an adequate storage policy.

Keywords: Performance Evaluation. Availability. Reliability Block Diagrams. Hybrid Storage. Stochastic Petri Nets. Cloud Computing.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

Sumário

1 INTRODUÇÃO

Este capítulo apresenta inicialmente o contexto (Seção 1.1) no qual está situada a abordagem proposta neste trabalho. Em seguida, discorre a respeito da motivação (Seção 1.2) acerca deste campo de estudo. Logo após, a Seção 1.3 exibe os objetivos, geral e específicos. Posteriormente, a Seção 1.4 explana e compara possíveis soluções encontradas em trabalhos relacionados com esta proposta. Por fim, este capítulo denota a estrutura desta dissertação (Seção 1.5).

1.1 CONTEXTO

*Solid-State Drive*s (SSDs) têm superado *Hard Disk Drive*s (HDDs) em proeminentes aspectos, como por exemplo, operações de leitura mais rápidas (LEE; MIN; EOM, 2015). HDDs provêm alta latência para acessos aleatórios devido à necessidade de movimentação de alguns de seus componentes mecânicos, e esta questão têm se tornado um gargalo em vários *Data Center*s (DCs) (YAMATO, 2015). Como consequência, SSDs também vêm sendo adotados em sistemas de alto desempenho nas últimas décadas, e o alto consumo de energia têm motivado a substituição de HDDs por SSDs (ZENG et al., 2012). Resultados experimentais mostram uma economia de 27% com SSDs, podendo este valor chegar a 60% dependendo da carga de trabalho (*workload*) (ZENG et al., 2012).

Por exemplo, provedores de serviço representativos (*Amazon*, *Dropbox*, *Facebook*) têm substituído HDDs em seus *data centers* por SSDs (WU et al., 2013) para obter melhor desempenho no que diz respeito as taxas de *Input/Output per Second* (IOPS), além de uma redução no consumo de energia. O *Myspace* anunciou a substituição de HDDs em todos os seus servidores por SSDs para economizar 99% de energia, e o *Facebook* realizou aperfeiçoamentos no armazenando de dados provenientes do *MySQL* dentro de SSDs (XIAO et al., 2012). Entretanto, existem alguns obstáculos para uma substituição completa de tecnologia já que SSDs usualmente não são capazes de suprir todos os requisitos no que diz respeito a desempenho, capacidade e confiabilidade. Para alguns *workloads*, SSDs podem não prover melhores resultados que HDDs no que diz respeito a operações de escrita. Além do mais, a baixa capacidade e o tempo de vida curto de SSDs, comparado a dispositivos de discos magnéticos, não permitem

lidar com a grande demanda de dados persistentes em *data centers* (YAMATO, 2015). Ademais, embora o custo por *gigabyte* de SSDs tenha diminuído ao longo dos anos, ainda são significativamente mais custosos do que HDDs (KIM; NO, 2014), aproximadamente 10 vezes mais (LEE; MIN; EOM, 2015). As desvantagens citadas anteriormente tornam difícil considerar apenas SSDs como um mecanismo de armazenamento para um alto volume de dados.

Alternativamente, abordagens híbridas vêm sendo propostas. *Hybrid Storage Drive*s (HSDs) possuem um desempenho maior que HDDs e um custo menor que SSDs, tornando-se uma solução promissora para vários sistemas, como os que são baseados em *Cloud Computing* (CC) (YAMATO, 2015); por exemplo, SSDs têm sido adotados como *cache* para HDDs. Outra tendência é a avaliação da frequência de acesso de dados e seu padrão (aleatório ou sequencial), armazenando dessa forma os dados no *drive* mais adequado. Todavia, a arquitetura de sistemas de armazenamento híbridos com SSDs ainda é um desafio. Muitas pesquisas já realizadas analisam principalmente a utilização do espaço de armazenamento, negligenciando questões como a confiabilidade (WAN et al., 2014). Modelos de dependabilidade e desempenho (MACIEL et al., 2011) são muito importantes, pois diferentes arquiteturas podem ser avaliadas antes da implementação real do sistema modelado.

Este trabalho apresenta uma abordagem baseada em modelos estocásticos para desempenho e disponibilidade através da modelagem de sistemas de armazenamento híbridos usando *Stochastic Petri Net*s (SPNs) e *Reliability Block Diagram*s (RBDs). Os modelos propostos podem representar operações de escrita, leitura, e *mixed* (escrita e leitura) além de estimar métricas como: tempo médio de resposta (*response time*), vazão (*throughput*), e disponibilidade (*availability*). Adicionalmente, este trabalho executa uma validação usando o simulador *DiskSim*, e um caso de estudo baseado na plataforma *OpenStack Swift* (OS) é adotado. Resultados demonstram a viabilidade da abordagem proposta e, comparações relacionadas a diferentes soluções de armazenamento híbrido e políticas de escrita e leitura.

1.2 MOTIVAÇÃO

O desempenho obtido por HDDs não é suficiente para a atual demanda exigida por sistemas que necessitam de altas vazões, baixa latência e consumo de energia reduzido. Por exemplo, em *data centers* que fornecem *Infrastructure as a Service* (IaaS), a utilização de HDDs na infraestrutura de armazenamento de *clouds* tornou-se um gargalo para aplicações que cada vez mais exigem níveis de desempenho superiores (YAMATO, 2015). Características inerentes a estes dispositivos, como a necessidade de movimentação mecânica de alguns de seus componentes, por exemplo, *platter*, *spindle* e *actuator arm*, dificultam um progresso mais significativo (LEE; MIN; EOM, 2015; MAO et al., 2012; LAGA et al., 2016). Esta observação torna-se mais perceptível quando se leva em consideração *workloads* compostos por requisições aleatórias, o que demanda por consequência o acesso a dados existentes em diferentes setores, provocando dessa forma o deslocamento físico para pontos extremos do disco.

Apesar dos fatores mencionados, o tempo médio de resposta para acesso a dados em discos magnéticos tem atingido um percentual de redução de 15% ao ano (PARK; SUH; BAEK, 2016); a diminuição do tempo de busca (*seek time*) em 8% e o aumento da velocidade de rotação (9%) são exemplos de fatores responsáveis por esse aperfeiçoamento, o qual foi obtido devido ao aprimoramento de técnicas como: *caching*, *write buffering*, *prefetching*, *request scheduling* e *parallel I/O* (HSU; SMITH, 2004; PARK; SUH; BAEK, 2016). No entanto, este ritmo de evolução da tecnologia de HDDs, no que diz respeito ao desempenho, não condiz com a de outros componentes, limitando, dessa forma, a velocidade de acesso e distribuição de dados ao desempenho do componente de armazenamento do sistema. Como exemplo, é possível citar o aumento da velocidade de processadores, os quais possuem uma evolução de até 60% anual, além da diminuição do tempo de acesso em *Random Access Memory* (RAM) em uma proporção 50% mais rápida que discos magnéticos (PARK; SUH; BAEK, 2016; HSU; SMITH, 2004).

Por não possuir componentes mecânicos e serem baseados em memórias *flash* (ZENG et al., 2012), SSDs a princípio seriam substitutos adequados, tendo em vista os significativos valores de vazão e tempo médio de resposta apresentados por esta tecnologia. Porém, sua adoção em larga escala não é possível devido a limitações ainda não solucionadas. Por exemplo, a baixa capacidade dos discos de estado sólido é um grande empecilho, especialmente no que diz respeito à *data centers*, nos quais um grande volume de dados são armazenados. Além disso, quando submetidos à mesma carga de trabalho, o tempo de vida de SSDs é bastante reduzido quando comparado a HDDs, devido à característica de seus blocos de memória, os quais possuem um número limitado de ciclos de programação e deleção (*program/erase cycles*), usualmente menor que 100000 (BU et al., 2012; LI; LEE; LUI, 2013; KIM; NO, 2014; WAN et al., 2014).

A combinação de ambas tecnologias citadas representa, dentro da literatura, uma possível solução no que diz respeito à *Quality of Service* (QoS) provida por *clouds*. Aspectos como disponibilidade e desempenho devem ser balanceados para a prestação dos serviços ofertados, caso contrário, falhas ou longos tempos de resposta podem acarretar na perda de usuários (YANG; TAN; DAI, 2013). Portanto, se faz necessário uma solução que permita a análise e exploração das características individuais dos *drives* mencionados, de acordo com o custo-benefício (KIM; NO, 2014), proporcionando dessa forma uma integração efetiva (WAN et al., 2014) através da arquitetura e políticas de armazenamento adequadas.

1.3 OBJETIVOS

O objetivo principal deste trabalho é a concepção de modelos analíticos para possibilitar a estimação e comparação das métricas de desempenho e disponibilidade, além do custo de sistemas de armazenamento híbridos, em ambientes cuja infraestrutura ofereça escalabilidade e mecanismos de redundância.

A fim de demonstrar a viabilidade da proposta mencionada, os objetivos específicos

deste trabalho são:

- elaborar modelos analíticos, baseados em um formalismo matemático de espaço de estado, especificamente redes de *Petri* estocásticas (*Stochastic Petri Net*), com os quais é possível avaliar o desempenho de operações realizadas em sistemas de armazenamento híbridos (leitura, escrita e *mixed*);
- conceber um modelo combinatorial, baseado em diagrama de blocos de confiabilidade (*Reliability Block Diagram*), para avaliar a disponibilidade de sistemas de armazenamento híbridos;
- validar os modelos propostos, considerando a técnica estatística teste t emparelhado, através da comparação entre valores obtidos da análise estacionária dos modelos de redes de *Petri* estocásticas concebidos, com os resultados do simulador de sistemas de armazenamento, *DiskSim*;
- demonstrar a viabilidade dos modelos propostos através de um estudo de caso que envolve a plataforma *OpenStack Swift*, com base na técnica *Design of Experiment* (DoE), para ponderar fatores que usualmente influenciam as métricas de desempenho e a disponibilidade dos dispositivos de armazenamento neste ambiente;
- comparar diferentes arquiteturas, tecnologias e políticas de armazenamento, no que diz respeito às métricas de desempenho e dependabilidade adotadas, além do custo de aquisição dos equipamentos envolvidos.

1.4 TRABALHOS RELACIONADOS

SSDs têm demonstrado ser uma tecnologia viável para sistemas de armazenamento, uma vez que contribuem significativamente para um melhor desempenho do sistema. Neste contexto, armazenagem híbrida é um proeminente campo de pesquisa que tem motivado diversos estudos. Vários trabalhos propõem arquiteturas para integração de HDDs e SSDs. Além disso, a disposição dos dados também é um fator a ser considerado para a diminuição do tempo médio de resposta de requisições, através da alocação específica dos dados no dispositivo de armazenamento mais adequado, de acordo com, por exemplo, o padrão de acesso.

PARK; SUH; BAEK (2016) propõem uma técnica de otimização para o acesso à *cache* de discos magnéticos, através da qual é possível identificar quando manter um metadado específico na *cache* proverá um ganho de desempenho. Experimentos são efetuados através do simulador *DiskSim*, nos quais o tempo médio de resposta às requisições sofre uma redução entre 15-20%. Em KIM; NO (2014), os autores consideram um mecanismo para prover a integração entre SSDs e HDDs através de um sistema de arquivos híbrido. Entretanto, este trabalho apenas investiga o desempenho dos dispositivos de armazenamento individualmente, e baseado nos resultados, uma estrutura de super bloco ideal é apresentada, sem experimentação ou simulação.

Com o intuito de avaliar diferentes políticas de armazenamento de dados, STRUNK (2012) efetua diversos experimentos a partir de um protótipo de armazenamento híbrido. Especificamente, um SSD é submetido a vários tipos de *workloads* com o propósito de encontrar um limiar de capacidade a partir do qual o ganho de desempenho não é mais significativo. De forma similar, CHEN; KOUFATY; ZHANG (2011) sugerem um módulo do *Kernel* para a identificação de blocos críticos, os quais são melhores processados quando direcionados para SSDs.

KIM et al. (2015) avaliam o impacto do armazenamento e processamento de metadados em SSDs, para isso, um modelo analítico é adotado com o objetivo de assistir a redução do *overhead* causado por acessos desnecessários à SSDs. Já em LEE; MIN; EOM (2015), resultados experimentais demonstram o ganho de desempenho obtido pela adoção de um *Virtual Disk Layer* (VDL) modificado, o qual é proposto para mitigar os danos causados por escritas aleatórias a discos de estado sólido; neste trabalho, SSDs são adotados como *cache*, tanto para operações de leitura, quanto para requisições de escrita, sendo as últimas convertidas em sequenciais pela camada de disco virtual (VDL). Dessa forma, o número de blocos apagados previamente às operações de escrita - *Write Amplification* (WA) - é reduzido. Para cumprir os altos requisitos de desempenho relacionados à IOPS, XU et al. (2015) propõem uma arquitetura de armazenamento híbrido, na qual *Raw Data* (RD) são processados em SSDs, enquanto que HDDs armazenam os dados já processados. Embora a abordagem supere soluções constituídas apenas por discos rígidos, simulações realizadas no *DiskSim* indicam a degradação do tempo médio de resposta, quando considerado somente SSDs.

Em YAMATO (2015), o autor apresenta uma comparação entre dispositivos híbridos (HDD+SSD), armazenamento distribuído, e um único HDD, fazendo uso do *OpenStack Cinder* (OC). Nesta experimentação, o sistema de armazenamento híbrido obteve os melhores índices de desempenho, levando em consideração o *workload* adotado. TAN; FONG; LIU (2014) investigam a efetividade da adoção do *Hadoop Distributed File System* (HDFS) em um armazenamento HDD-SSD. Os experimentos utilizam diferentes arquiteturas, as quais são submetidas a três tipos de *workloads* encontrados em ambientes de *Big Data* (BD). Embora os resultados indiquem os benefícios da adoção de SSDs, os autores sugerem que arquiteturas baseadas em *workload-awareness* podem obter melhores resultados. WAN et al. (2014) apresenta um *framework* para a predição do uso de dados baseada no padrão de acesso, frequência e *workloads* específicos; a estimação futura do uso de dados é realizada através de uma *Continuous-Time Markov Chain* (CTMC). Resultados experimentais baseados em vestígios de dados (*data traces*) demonstram que uma alta vazão de leitura pode ser obtida com a atenuação de acessos aleatórios.

A Tabela 1.1 reúne as propostas dos trabalhos relacionados mencionados, bem como as abordagens desta dissertação. Diferente de pesquisas anteriores, o presente trabalho propõe modelos analíticos baseados em SPNs e RBDs para avaliar o desempenho e disponibilidade de *Hybrid Storage Systems* (HSSs). Também é levado em consideração o sistema de armazenamento de uma plataforma de computação em nuvem (*OpenStack Swift*) para demonstrar a

usabilidade prática dos modelos concebidos. Ademais, estes modelos são validados através do simulador *DiskSim*, uma ferramenta largamente adotada para avaliação de arquiteturas de armazenamento. Este trabalho compreende diversos experimentos através dos modelos propostos, adotando tecnologias tradicionais (HDD e SSD), bem como as políticas de armazenamento para dispositivos híbridos que serão apresentadas a seguir (Seção 2.3). Adicionalmente, diferente de outros estudos, as métricas avaliadas são analisadas concomitantemente, com o intuito de prover resultados que levem em consideração aspectos de desempenho e dependabilidade.

Tabela 1.1: Comparação entre esta dissertação e trabalhos relacionados

	Modelagem	**Avaliação de Desempenho**	**Avaliação de Dependabilidade**	**Validação**	**Avaliação de Desempenho, Dependabilidade e Custo**	**Armazenamento Híbrido**
Este trabalho	✔	✔	✔	✔	✔	✔
(PARK; SUH; BAEK, 2016)		✔				✔
(KIM; NO, 2014)		✔				✔
(STRUNK, 2012)		✔				✔
(CHEN; KOUFATY; ZHANG, 2011)		✔				✔
(KIM et al., 2015)	✔	✔				✔
(LEE; MIN; EOM, 2015)		✔				✔
(XU et al., 2015)		✔				✔
(YAMATO, 2015)		✔				✔
(TAN; FONG; LIU, 2014)		✔				✔
(WAN et al., 2014)	✔	✔				✔

1.5 ESTRUTURA DA DISSERTAÇÃO

Inicialmente, este capítulo contextualiza a temática na qual o objeto de pesquisa está inserido. Em seguida, aprofunda o tema apresentado tendo em vista explanar a motivação acerca dos problemas existentes no campo de pesquisa avaliado. Posteriormente, define os objetivos (geral e específicos) e, apresenta trabalhos relacionados a este estudo para expor as soluções existentes na literatura, bem como as lacunas a serem preenchidas.

O Capítulo 2 apresenta o referencial teórico necessário para compreensão deste trabalho. Primeiramente, são denotados os conceitos acerca dos dispositivos de armazenamento estudados (HDD e SSD). Em seguida, demonstra as políticas de armazenamento abordadas neste estudo, as quais são destinadas a dispositivos híbridos. Posteriormente, introduz as métricas

utilizadas para avaliação das arquiteturas abordadas, as quais estão relacionadas aos conceitos de desempenho e dependabilidade. Os formalismos matemáticos, SPN e RBD, também são abordados para apresentar os conceitos necessários à modelagem realizada.

Já o Capítulo 3 apresenta a metodologia adotada. A princípio, expõe as etapas aplicadas para a compreensão do ambiente estudado, concepção dos modelos analíticos, e planejamento dos experimentos. Além disso, é feito um esclarecimento a respeito das arquiteturas assumidas para o desenvolvimento das propostas desta dissertação. Também são denotados os métodos e técnicas estatísticas para a análise dos resultados obtidos nos experimentos realizados.

O Capítulo 4 introduz os modelos SPN e RBD concebidos neste trabalho. Inicialmente, este capítulo explana algumas considerações e conceitos necessários para a compressão das métricas de desempenho e disponibilidade assumidas neste trabalho. Ademais, também define o propósito de cada um dos modelos. Posteriormente, apresenta e explana detalhadamente os modelos analíticos a fim de prover um melhor esclarecimento.

O Capítulo 5 apresenta os resultados obtidos dos experimentos realizados. Inicialmente, expõe os parâmetros adotados de acordo com a arquitetura de computação em nuvem assumida (*OpenStack Swift*). Em seguida, demonstra a validação dos modelos de desempenho concebidos (SPN). Posteriormente, exibe os planejamentos para os experimentos (DoE) de desempenho e disponibilidade. Em seguida, apresenta diversos experimentos com o intuito de prover uma comparação entre diferentes tecnologias e políticas de armazenamento. Por fim, exibe uma análise de diferentes configurações de um sistema de armazenamento, através do cálculo da *Euclidian Distance* (ED), na qual são levadas em consideração, concomitantemente, métricas de desempenho, dependabilidade e custo.

Finalmente, o Capítulo 6 apresenta observações acerca dos resultados obtidos, bem como as contribuições alcançadas. Este capítulo também menciona trabalhos futuros, com o intuito de propor possíveis pesquisas a serem realizadas dentro do contexto no qual o trabalho está inserido.

2

REFERENCIAL TEÓRICO

Este capítulo introduz os conceitos básicos necessários para a compreensão do trabalho proposto. As Seções 2.1 e 2.2 apresentam os dispositivos que compõe o objeto de estudo deste trabalho. Em seguida, a Seção 2.3 explana as políticas adotadas para dispositivos de armazenamento híbridos. Posteriormente, esse capítulo define os conceitos de desempenho e dependabilidade (Seções 2.4 e 2.5), os quais são necessários para experimentação dos modelos formais concebidos. Logo após, a Seção 2.6 denota os conceitos a respeito de redes de Petri (PN) e suas extensões. Finalmente, a Seção 2.7 apresenta o formalismo diagrama de blocos de confiabilidade (RBD).

2.1 *HARD DISK DRIVE*

Hard Disk Drive (HDD) é um componente essencial para computadores pessoais e grandes sistemas de processamento de dados (CHEN et al., 2006). Desde o início de sua produção, em 1956, a indústria tem fomentado excelentes inovações relacionadas ao design e fabricação dos mesmos, atingindo níveis de evolução semelhantes ao da revolução dos semicondutores (AL MAMUN; GUO; BI, 2006). Além disso, em meio a sistemas de armazenamento magnético, HDD é o dispositivo secundário de armazenagem em massa dominante, no que diz respeito à produção industrial, devido aos seguintes fatores: grande capacidade, baixo custo por *gigabyte* e uma ampla infraestrutura de produção (WANG; TARATORIN, 1999).

2.1.1 Arquitetura

Com relação à arquitetura de computadores, HDDs estão situados entre as RAMs e dispositivos (*drives*) removíveis (AL MAMUN; GUO; BI, 2006). Dessa forma, HDDs proveem acesso direto a grandes quantidades de dados não-voláteis, ou seja, não é requerido energia para preservar os dados. Além do mais, possui atributos considerados relevantes por usuários como: baixo custo, razoável vazão de dados e tempo de acesso, e uma considerável confiabilidade, sendo também conhecido como *Direct Access Storage Device* (DASD).

Basicamente, os componentes (Figura 2.1) dos discos magnéticos são classificados em quatro categorias (AL MAMUN; GUO; BI, 2006):

- **Componentes magnéticos:** estes componentes são responsáveis pelo armazenamento, gravação e leitura dos dados;
    - Disco (*disk*): usualmente os discos são feitos de alumínio e cobertos em ambos os lados por uma pequena camada de material magnético. O mesmo é fixado através de um eixo (*spindle*) o qual possui um motor capaz de atingir velocidades acima de 6 mil RPM (rotações por minutos), permitindo dessa forma o acesso às faixas (*tracks*) para escrita ou leitura dos dados;
    - Cabeça de leitura/escrita: estes elementos são utilizados para a leitura e escrita de dados nos discos magnéticos. Ambos os componentes são integrados à uma larga estrutura (*slider*) que provê conectividade elétrica para as bobinas das "cabeças". Dessa forma, a proximidade e polarização das bobinas geram um campo magnético de forma a induzir a magnetização da mídia, permitindo dessa maneira a escrita e leitura de bits;
- **Componentes mecânicos:** os componentes estritamente mecânicos se restringem ao revestimento externo do drive. Tanto a tampa superior (*top clover*), quanto a placa da base (*base plate*) criam um invólucro para proteção do dispositivo. Além disso, a placa da base também é usada como suporte para fixação dos componentes internos;
- **Componentes eletromecânicos:**
    - *Spindle motor:* trata-se de um motor de corrente contínua com o propósito de girar e posicionar o disco magnético na faixa onde será escrita/lida a informação. A velocidade do motor deve ser precisamente controlada a fim de evitar perturbações à performance do *drive*. Por exemplo, a velocidade é um fator influenciável à proximidade do *slider* ao disco; processo esse que pode vir a afetar a densidade dos bits armazenados;
    - Atuador (*actuator*): a movimentação da cabeça de leitura/escrita através de diferentes raios do disco é efetuada por este componente (Figura 2.2). Este deslocamento é possível devido ao *Voice Coil Motor* (VCM), o qual de acordo com a lei de *Faraday* [1] provoca a aproximação ou distanciamento do *slider* ao *spindle*, quando da passagem de corrente pela bobina

[1] Também denominada como lei da indução magnética. Ela prevê como um campo magnético interage com um circuito elétrico para produzir uma força eletromotriz

do mesmo. Atualmente, a quantidade de *sliders* é proporcional ao número de discos do *drive*, no entanto, todos são conectados a um único atuador;

- **Eletrônicos:** os componentes eletrônicos usualmente constituem um único *chip*, o qual é situado na placa de circuitos impresso (*printed circuit board* - PCB).
    - *Interface:* este componente possibilita a comunicação entre o driver e o sistema hospedeiro. São alguns exemplos: *Integrated Drive Electronics* (IDE), *Advanced Technology Attachment* (ATA), *Small Computer Systems Interface* (SCSI) e *Serial Advanced Technology Attachment* (SATA);
    - Circuitos Integrados: possuem funções como o caminho físico para a transmissão dos dados (*channel electronics*), processamento digital dos sinais (*digital signal processing* - DSP) e controle da *interface*. Também estão incluídos nessa classificação a *cache* (RAM) e o controlador de disco.

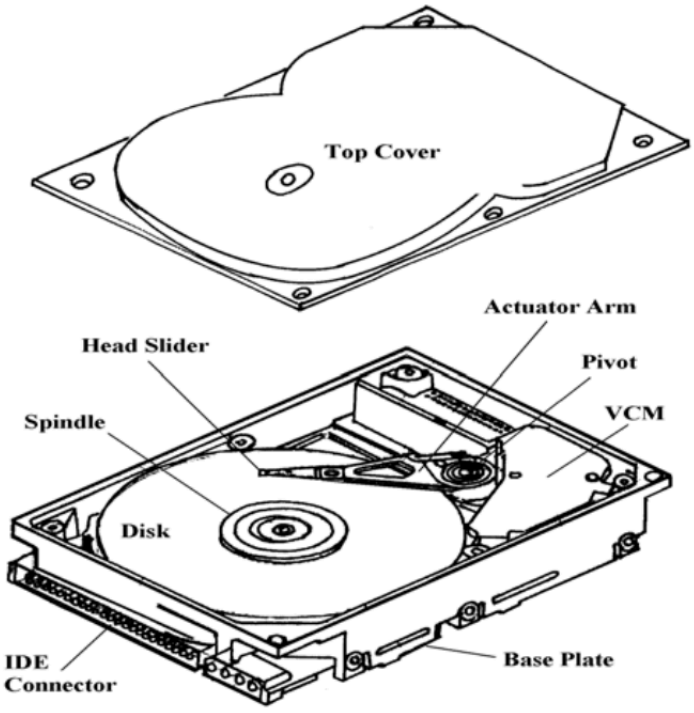


Figura 2.1: Componentes de um *hard disk drive* (HDD) (AL MAMUN; GUO; BI, 2006)

2.1.2 Desempenho

Sistemas que demandam um grande número de acessos (por exemplo, *clouds*) necessitam de vazões significantes, além de um baixo tempo médio de resposta (*response time*). Um pequeno atraso na entrega das requisições pode acarretar em consideráveis atrasos (*delays*) no

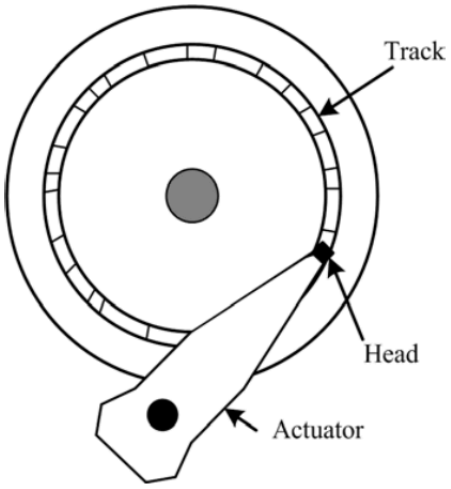


Figura 2.2: Atuador (AL MAMUN; GUO; BI, 2006)

processamento de dados e execução de programas (AL MAMUN; GUO; BI, 2006). Portanto, quando da adoção de HDDs, os seguintes parâmetros de desempenho (WANG; TARATORIN, 1999) dos discos rígidos são observados:

- **Capacidade:** trata-se do espaço de armazenamento existente no dispositivo;
- **Latência rotacional:** tempo para deslocamento da cabeça de leitura de um determinado setor para outro, dentro de uma mesma faixa;
- **Tempo de acesso:** diz respeito à soma do tempo de busca (*seek time*), tempo para cessar as vibrações (*head setting time*) e o tempo da latência rotacional;
- **Tempo médio de resposta:** está diretamente relacionado ao tempo de acesso, além da execução da operação requisitada;
- **Vazão:** representa o número de bits por unidade de tempo que a cabeça de leitura/escrita é capaz de processar.

2.1.3 Confiabilidade

A confiabilidade de dispositivos de armazenamento é uma grande preocupação de sistemas computacionais de alta performance e provedores de serviço na Internet (SCHROEDER; GIBSON, 2007). Grandes *clusters* de servidores enfrentam falhas de *storages* que podem vir a provocar a indisponibilidade temporária de dados, e em alguns casos sendo até mesmo permanente. O aumento na geração de dados conduz à necessidade de expansão dos sistemas de armazenamento, o que consequentemente acarreta no acréscimo de *drives* com falha.

É importante ressaltar que o comportamento da falha de um HDD depende das condições de operação. Portanto, fatores ambientais como temperatura e umidade, além de diferentes

cargas de trabalho (*workloads*) e horas de funcionamento, podem afetar de formas diferentes um mesmo modelo (SCHROEDER; GIBSON, 2007). Ademais, fatores internos também são fontes de distúrbios e erros. Neste caso, os sistemas de controle necessitam alcançar um nível de regulação extremamente preciso no que diz respeito aos servomecanismos. Estas fontes de erro estão listadas abaixo por ordem de impacto (CHEN et al., 2006):

1. vibrações e choques externos;
2. baixa velocidade do disco e histerese [2];
3. imprecisão causada por efeitos nas cabeças de leitura e escrita;
4. ressonância mecânica no atuador e disco;
5. ruídos eletrônicos nos canais.

2.2 *SOLID-STATE DRIVE*

Solid-State Drive (SSD) é um dispositivo que incorpora memórias de estado sólido e emula HDDs para o armazenamento de dados (MICHELONI; MARELLI; ESHGHI, 2012). O advento de SSDs representa certamente uma mudança significativa no que diz respeito à sistemas de armazenamento. Capaz de atingir altos valores de vazão, obtém níveis de desempenho bem maiores do que dispositivos de discos magnéticos, especialmente com relação às requisições aleatórias (AGRAWAL et al., 2008).

Acessos aleatórios representam grande parte das transferências *online* realizadas por grandes companhias. Com a migração de serviços para os centros de dados (*datacenters*), a diversidade de aplicações existentes em um único sistema de armazenamento foi incrementada de forma que cada microssegundo de latência a mais representa significante perda de dinheiro, eficiência e energia. Portanto, no que concerne ao ambiente corporativo, SSDs tornaram-se essenciais para o incremento do desempenho de sistemas (MICHELONI; MARELLI; ESHGHI, 2012). A adoção desta tecnologia se faz necessária devido à vazão e tempo de acesso demandados por aplicações como: processamento de transações *online*, mineração de dados, e computação em nuvem.

O tempo de acesso de HDDs depende do quão rápido é possível mover a cabeça de leitura para a faixa onde está o dado requerido. Além disso, a vazão máxima é ditada pela velocidade de rotação do disco. Dessa forma, mesmo obtendo valores razoáveis de desempenho para requisições sequenciais, sua limitação mecânica provoca incrementos significativos da latência de acesso a dados aleatórios.

[2] Este termo refere-se à capacidade de um sistema conservar suas propriedades quando da ausência do estímulo que o levou a um determinado estado

Comparativamente, dispositivos de estado sólido apresentam melhores resultados com relação ao consumo de energia, resistência a impactos, e vazões (IOPS), para requisições sequenciais e aleatórias (MICHELONI; MARELLI; ESHGHI, 2012). Por exemplo, o tempo médio de resposta em SSDs pode ser da ordem de microssegundos, enquanto que milissegundos é a duração necessária em HDDs. No entanto, apesar do aumento da produção e redução do preço da *Flash Memory* (FM), o alto custo por *gigabyte* ainda torna inviável a adoção irrestrita desses dispositivos de estado sólido.

2.2.1 Arquitetura

Basicamente, SSDs possuem como componente principal conjuntos de memória *flash*. No entanto, cada SSD contém também uma *interface* lógica para suportar a conexão física com o hospedeiro (*interface* física) e a emulação lógica de um HDD, a qual é realizada pelo controlador de memória (controlador do SSD) (AGRAWAL et al., 2008). A emissão de comandos e transporte de dados são processados por um multiplexador (*demux/mux*), através de uma conexão serial para os pacotes de memória *flash*. O gerenciamento desse fluxo de dados é realizado por um processador, o qual conta com um gerenciador de *buffer* para lidar com requisições pendentes, além de mapear os endereços dos blocos lógicos para suas localizações físicas. Esse mapeamento (*logical block adrress*) é armazenado em uma memória RAM no momento que o *drive* é energizado. O propósito é não desgastar as memórias NAND (*Not And*) com a frequente atualização de informação dos dados já existentes. Este componente (RAM) também é utilizado como *cache* para armazenar os dados antes de transferi-los para as memórias *flash*. A disposição dos componentes é mostrada na Figura 2.3.

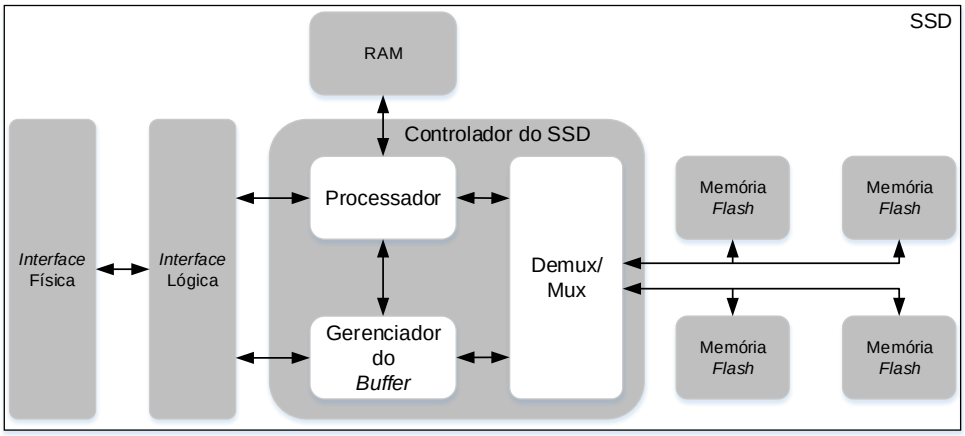


Figura 2.3: Arquitetura de um *solid-state drive* (SSD)

2.2.2 Controlador da memória

Como dito anteriormente, SSDs possuem um controlador integrado para efetuar a conexão com o hospedeiro (*interface* física - *PCI-Express, Serial Attached SCSI, Serial Advanced Technology*) e memórias NAND. Adicionalmente, atividades como nivelamento de desgaste (*wear leveling*), coletor de lixo (*garbage collection*), gerenciamento de blocos inutilizados (*bad block management*) e o mapeamento de blocos lógicos para blocos físicos, constituem a camada de tradução *flash* (*flash translation layer*) (EL MAGHRAOUI et al., 2010). Para identificação e reparo de erros, um *hardware* específico executa um código de correção (*error correction code* - ECC), o qual usualmente é compartilhado entre os múltiplos canais *flash* (*flash channels*) (MICHELONI; MARELLI; ESHGHI, 2012; WOO; KIM, 2013). A Figura 2.4 demonstra a organização em blocos das funcionalidades mencionadas a respeito do controlador de memória.

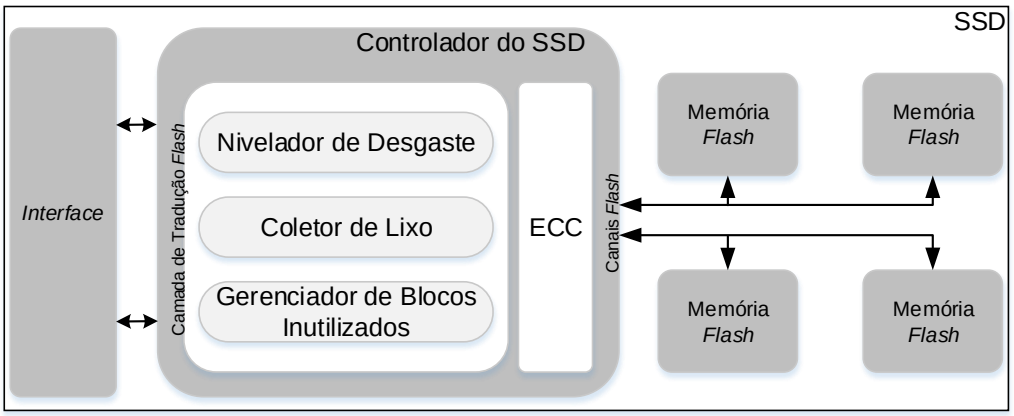


Figura 2.4: Funcionalidades do controlador da memória

O mecanismo de nivelamento de desgaste possui como propósito a utilização mínima e uniforme dos blocos das memórias NAND. Portanto, um número máximo de ciclos (escrita/deleção) estimado para um bloco é levado em consideração para a execução desta técnica. Essa distribuição de dados acarreta na execução do coletor de lixo, o qual a partir de um valor limite (*threshold*) de blocos livres inicia a verificação de cópias existentes de um mesmo arquivo, mantendo dessa forma a última versão, executando em seguida a deleção das duplicatas. Esta atividade pode vir a ser prejudicial ao desempenho da memória, dessa forma, o coletor de lixo usualmente é executado em segundo plano (MICHELONI; MARELLI; ESHGHI, 2012).

O equilíbrio entre essas duas técnicas citadas é capaz de postergar o tempo de vida de SSDs, no entanto, o surgimento de blocos inutilizados é inevitável. A identificação e mapeamento de blocos inutilizados são realizados pelo módulo gerenciador de blocos inutilizados (*bad block management*). Para isto, uma tabela de blocos inutilizados (*bad blocks table*) é criada no momento da primeira inicialização da memória (MICHELONI; MARELLI; ESHGHI,

2012), contendo uma lista dos *bad blocks* presentes no teste de fábrica, sendo atualizada durante sua utilização e surgimento dos mesmos.

2.2.3 Memória *Flash* NAND

Memórias *flash* NAND são usualmente utilizadas em SSDs; trata-se de um dispositivo eletrônico, sem partes mecânicas, composto por conjuntos de células de porta flutuante (*floating gate cell*). Embora forneçam excelente velocidade de acesso aleatório (quando comparada a discos magnéticos), possui limitações, como por exemplo, a necessidade de deleção de blocos antes da escrita de dados, e limite de ciclos de programação/deleção (*program/erase cycles*) para inutilização dos mesmos (WOO; KIM, 2013). Consequentemente, a velocidade de leitura e escrita é assimétrica, ou seja, é necessário mais tempo para escrever do que para ler um determinado dado.

Um chip de memória NAND é constituído por vários blocos, os quais usualmente possuem de 64 a 128 páginas (MAO et al., 2012); esta estrutura (*page*) é a unidade para leitura ou escrita (granularidade), a qual geralmente possui 4 KB. Em contraste, a operação de deleção apenas pode ser executada em um bloco inteiro. Portanto, uma vez que a página tenha sido programada a mesma não pode ser atualizada, ou seja, todo o bloco é deletado e os dados ainda relevantes devem ser reescritos em um espaço livre (*erase-before-write*) (WOO; KIM, 2013). O excesso de operações de deleção é um gargalo para o desempenho de memórias *flash* (RICHTER, 2013), além de causar excessos de fragmentação interna (BREWER; GILL, 2011).

A arquitetura das células de porta flutuante pode ser visualizada na Figura 2.5. A definição do estado lógico em que se encontra a célula está diretamente relacionada à transferência de elétrons do substrato para a porta flutuante (*floating gate*), através da porta de óxido (*oxide gate*). Inicialmente, a ausência de elétrons no *floating gate* caracteriza o estado lógico "1". O processo de programação, ou escrita, é obtido através da geração de uma diferença de potencial entre o dreno (*drain*) e à fonte (*source*). Dessa forma, um fluxo de elétrons é constituído em direção ao dreno; no entanto, a aplicação de uma voltagem pré-definida na porta de controle (*control gate*) gera um campo elétrico forte o suficiente, no dielétrico interpolar, para atrair os elétrons em direção ao *floating gate* (WOO; KIM, 2013). A energia cinética adquirida pelos elétrons é forte o suficiente para atravessarem o dielétrico (*oxide gate*) e serem armazenados na porta flutuante, obtendo assim o estado lógico "0". Este processo é reversível, através da inversão da polaridade aplicada na porta de controle, caracterizando dessa forma a operação de deleção (KULKARNI; JISHA, 2013). O processo de leitura é realizado através da aplicação de uma voltagem pré-definida na porta de controle e no dreno. Esta operação induz à geração de um valor específico de corrente, o qual é dependente da quantidade de carga existente no *floating gate*. Em seguida, seu valor é convertido para o estado lógico correspondente.

A execução de diversas operações de escrita e reescrita durante o tempo de vida de uma memória *flash* naturalmente danifica os componentes das células não-voláteis. Inevitavelmente,

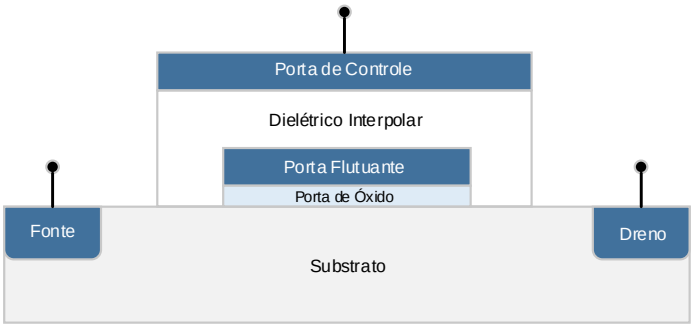


Figura 2.5: Célula de porta flutuante

a injeção e remoção de elétrons, através da barreira de óxido, provocará um dano irreversível, como por exemplo o vazamento de elétrons. A degradação das células é estimada de acordo com a quantidade de ciclos de escrita/deleção, atingindo em média 100 K ciclos para células de nível único (RICHTER, 2013).

Para estender sua capacidade, SSDs modernos têm adotado células com níveis múltiplos (*multi-level cell* - MLC), ou seja, possibilita o armazenamento de 2 *bits*, enquanto que a abordagem de nível único (*single-level cell* - SLC) permite apenas um único *bit* (LEE; MIN; EOM, 2015). No entanto, a quantidade de ciclos mínima, para o surgimento de falhas em um determinado bloco, é reduzida de 100 K (SLC) para 10 K (MLC) ciclos (LI; LEE; LUI, 2013; MAO et al., 2012).

2.3 SISTEMAS DE ARMAZENAMENTO HÍBRIDO

Dispositivos de estado sólido possuem uma interface de entrada e saída similar a HDDs. Portanto, diversas pesquisas têm concebido técnicas para o desenvolvimento de dispositivos de armazenamento híbrido (WOO; KIM, 2013). Usualmente, as tentativas de aperfeiçoamento se concentram na camada de *software*, isto é, o controlador de armazenagem, na qual é possível prover melhorias de forma a aumentar o desempenho dos dispositivos. Este trabalho aborda técnicas que envolvem um melhor aproveitamento da *cache* além de um mecanismo de classificação e direcionamento dos dados.

Nesta seção são descritas duas arquiteturas representativas para armazenamento híbrido, as quais são posteriormente adotadas para demonstrar os modelos propostos neste trabalho, além da estimação de métricas de desempenho e disponibilidade.

2.3.1 SSD como *Cache* (*SSDasCache*)

Devido ao baixo desempenho de discos magnéticos para lidar com operações de escrita e leitura aleatórias, e o alto custo de memórias *cache* tradicionais, baseadas em RAM, SSDs

tornaram-se a princípio uma solução adequada para a vazão (*throughput*), e o tempo médio de resposta (*response time*), de sistemas computacionais. Considerando que as modificações para esta abordagem usualmente são mínimas, diversos trabalhos têm adotado SSD como um mecanismo de *cache* (Figura 2.6) (LEE; MIN; EOM, 2015; WU et al., 2015; BU et al., 2012; APPUSWAMY; MOOLENBROEK; TANENBAUM, 2012; LEE et al., 2011).

Além de considerar o dispositivo SSD como *cache*, este trabalho adota a política de armazenamento escrever de volta (*write-back*), a qual é uma abordagem bastante comum (LEE; MIN; EOM, 2015; APPUSWAMY; MOOLENBROEK; TANENBAUM, 2012). Esta política é caracterizada por direcionar todas as operações de escrita para a *cache*, e apenas periodicamente para o disco primário.

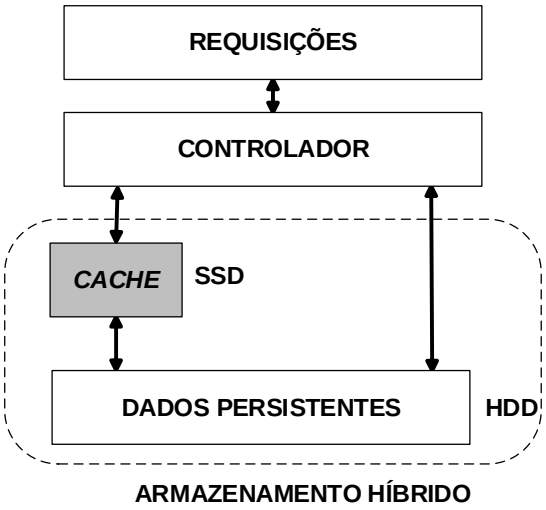


Figura 2.6: Armazenamento híbrido com o *solid-state drive* (SSD) como *cache* para um *hard disk drive* (HDD)

2.3.2 SSD para Requisições Aleatórias (*SSDRandom*)

Requisições de pequenos objetos em um sistema de armazenamento podem vir a causar uma significativa degradação do desempenho (CHEN; KOUFATY; ZHANG, 2011), e neste contexto, metadados (*metadata*) possuem uma contribuição considerável. Mais especificamente, blocos de metadados contêm atributos relacionados a cada arquivo armazenado, como por exemplo sua localização no *drive* e tamanho. Além disso, metadados devem ser armazenados na memória antes que um arquivo possa ser manipulado, aumentando de forma considerável a quantidade de requisições de entrada e saída (MAO; WU; JIANG, 2015). Estes blocos, apesar de geralmente serem pequenos (STRUNK, 2012), são responsáveis por 99% do tempo de operação de I/O (CARNS et al., 2011). Em HDDs, a manipulação de metadados afeta significativamente o desempenho devido às rotações necessárias para acesso, tanto do metadado, quanto do dado a que se refere, pois ambos usualmente estão em diferentes trechos.

Tendo em vista que usualmente as buscas de dados seguem um padrão de acesso aleatório, diversos autores sugerem o armazenamento dos blocos de metadados em SSDs (CHEN; KOUFATY; ZHANG, 2011; APPUSWAMY; MOOLENBROEK; TANENBAUM, 2012; WU et al., 2015). De acordo com os resultados desses experimentos, SSDs demonstraram ser um mecanismo adequado para reduzir o atraso dos acessos aleatórios a esses blocos. Entretanto, a utilização de SSDs exclusivamente para esse propósito (metadados) pode prejudicar possíveis benefícios que poderiam ser alcançados quando do armazenamento de outros tipos de dados neste dispositivo. Portanto, o aperfeiçoamento do desempenho deve levar em consideração fatores como: a quantidade de blocos de metadados, o tipo da operação (escrita ou leitura), e o padrão de acesso (sequencial ou aleatório). Adicionalmente, a capacidade e a resistência do SSD não devem ser negligenciadas.

Este trabalho considera SSDs como um mecanismo para armazenamento de blocos de metadados e outros acessos aleatórios de dados (Figura 2.7); em contraste, HDDs são responsáveis pelo armazenamento de dados sequenciais. O reconhecimento de padrão, isto é, se o *workload* é aleatório ou sequencial, é um procedimento possível através do uso de abordagens baseadas em *software* e *hardware* (NIJIM et al., 2011; JOO et al., 2014). Por exemplo, chamadas do sistema (*system calls*) no *kernel* do sistema operacional *Linux*, e o *firmware* de *drives* (exemplo, HDD, SDD e híbrido) (JOO et al., 2014; NIJIM et al., 2011; CHEN; DING; JIANG, 2009), podem detectar se uma determinada operação, leitura ou escrita, possui um padrão sequencial observando o tamanho da requisição, frequência e distância entre os blocos. Adicionalmente, alguns autores, como RAMASAMY; KARANTHARAJ (2015), têm demonstrado os benefícios por dividir um *workload* em requisições aleatórias e sequenciais.

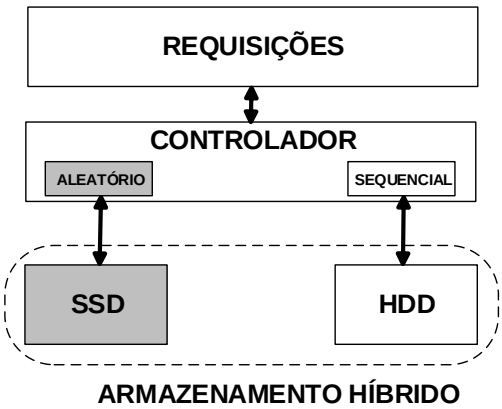


Figura 2.7: Política de armazenamento de dados aleatórios em um dispositivo híbrido

2.4 AVALIAÇÃO DE DESEMPENHO

A avaliação de desempenho tornou-se um pré-requisito para cada estágio da vida de um sistema computacional, desde a criação do projeto até sua fabricação, além de um possível aprimoramento futuro (JAIN, 1990). No entanto, sistemas computacionais não ocupam mais apenas áreas exclusivas; ou seja, possuem um escopo que envolve desde a substituição de alguns controles mecânicos dos carros, até os celulares disponíveis para o público em geral. Portanto, essa proliferação e mudança demanda que tanto desenvolvedores de sistemas quanto usuários atenham-se ao desempenho dos equipamentos adotados (LILJA, 2005).

Quando aplicada, por exemplo, à experimentos da ciência da computação e engenharia, a análise de desempenho deve ser realizada de acordo com a combinação da medição, interpretação e comunicação das métricas estudadas de um determinado sistema computacional. Entretanto, frequentemente é necessária a análise de uma pequena porção independente do sistema, por exemplo, dispositivos de armazenamento. Infelizmente, alguns componentes podem possuir uma interação bastante complexa, fazendo com que seja imprescindível uma cuidadosa decisão a respeito das técnicas, carga de trabalho e ferramentas a serem utilizadas (LILJA, 2005).

Devido à existência de diversos tipos de métricas de desempenho, diferentes estratégias devem ser concebidas em torno da ideia da medição de um evento; a mudança de estado de um sistema pode ser classificada como um evento (LILJA, 2005). Por exemplo, um evento pode ser definido como o acesso a um disco, uma operação de comunicação entre componentes ou até mesmo a combinação de ambos. O temporizador de intervalo é uma ferramenta de medição fundamental na análise de desempenho de sistemas computacionais. Basicamente, sua utilização está relacionada à ideia de contabilizar o tempo entre dois eventos pré-definidos. Já a ferramenta "perfil"(*profile*) fornece uma visão geral a respeito do comportamento de um sistema durante sua execução. Especificamente, é uma medição do tempo despendido em um específico estado.

Entretanto, as técnicas citadas não levam em consideração a ordem de execução dos eventos observados. Para isso, uma lista de eventos dinâmica (*trace*) é gerada por um programa enquanto é executado. No entanto, o distúrbio de comportamento causado pela instrumentação de um sistema é uma situação quase que inevitável. Dessa forma, o excesso de instrumentação, muitas vezes adotado pela necessidade de mais dados, leva à obtenção de dados menos confiáveis. Como resultado, usualmente a inferência a respeito do comportamento de sistemas é realizada com pouca informação, a fim de não alterar o comportamento dos sistemas.

Qualquer ferramenta para medição possui três importantes características as quais determinam a qualidade de seus resultados (LILJA, 2005). Por exemplo, a acurácia (*accuracy*) é uma indicação do quão próximo um determinado valor medido (*true value*) está da média da amostra (*mean value*); portanto, trata-se da diferença entre um valor medido e seu valor de referência. A segunda característica a ser considerada é a precisão (*precision*); geralmente está relacionada à extensão de medições ao redor do valor médio, sendo a distribuição resultante uma indicação

da precisão desse processo. Por fim, a resolução (*resolution*) mensura a menor mudança de um sistema que pode ser detectada. A Figura 2.8 ilustra a diferença entre acurácia e precisão.

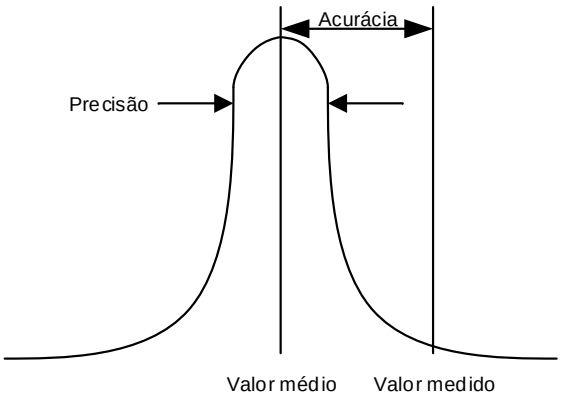


Figura 2.8: Medições hipotéticas demonstrando a diferença entre acurácia e precisão

A princípio, para a avaliação de desempenho se faz necessária uma referência que represente as características das aplicações a serem executadas no sistema a ser avaliado. Uma carga de trabalho real (*real workload*) é a observação de um sistema em sua condição normal de operação (JAIN, 1990). No entanto, dificilmente as condições observadas de um sistema real serão repetidas. Portanto, uma carga de trabalho sintética (*synthetic workload*) é mais adequada para ser aplicada em experimentos. Além de ser similar a cargas de trabalho reais, possibilitam a repetição de forma controlada do experimento, permitindo assim uma análise mais precisa dos parâmetros dos sistemas. Os seguintes tipos de carga de trabalho são usados para comparação de sistemas computacionais: adição de instrução, mistura de instruções, *kernels*, programas sintéticos e aplicação comparativa.

A medição de um sistema real, simulação, e modelagem analítica, são as três técnicas fundamentais existentes para a avaliação de desempenho de sistemas (JAIN, 1990). Usualmente, a medição de um sistema real não é uma técnica flexível com relação à mudança de parâmetros, porém, fornece informações confiáveis a respeito do sistema com um todo. No entanto, uma das características desejáveis para avaliação diz respeito ao acompanhamento da mudança de comportamento à medida que as configurações são alteradas. Por exemplo, avaliar o impacto no desempenho total, após a mudança de velocidade de apenas um componente, pode se demonstrar uma tarefa muito difícil, senão impossível de ser realizada. Medições em sistemas reais podem vir a demandar muito tempo para serem efetuadas, além de serem mais custosas devido à necessidade da compra de equipamentos (LILJA, 2005).

As técnicas de modelagem analítica, e simulação, não são significativamente afetadas pelas desvantagens mencionadas a respeito da medição de sistemas reais (JAIN, 1990). A simulação de um sistema computacional é realizada por um programa feito para modelar importantes características a serem analisadas (LILJA, 2005). Por se tratar de um programa, o mesmo pode

ser modificado para o estudo do impacto de mudanças nos componentes simulados. Dependendo do nível de detalhes do sistema simulado, o custo e tempo demandados para a análise podem ser significativamente reduzidos quando comparado a experimentos em máquinas reais. No entanto, a dificuldade em abranger todos os detalhes, e a exigência de um tempo reduzido, tanto para desenvolvimento do simulador quanto para execução da simulação, pode vir a limitar a acurácia dos resultados obtidos (LILJA, 2005).

Em geral, modelos analíticos podem vir a ser uma solução mais simplificada e precisa, além de menos custosa para a avaliação de desempenho de sistemas (JAIN, 1990). Um modelo analítico nada mais é que a descrição matemática de um sistema (LILJA, 2005). Usualmente, a modelagem analítica fornece uma melhor compreensão a respeito dos efeitos dos parâmetros do sistema e suas interações. Em adição, também pode auxiliar na validação de resultados produzidos por um simulador, ou de valores medidos em um sistema real.

A teoria das filas (*queueing theory*) é uma importante técnica de modelagem analítica para sistemas. Em sistemas de computadores, muitas tarefas não compartilham os recursos existentes, como o núcleo de um processador, discos e interfaces de rede, por exemplo (Figura 2.9). No entanto, considerando um sistema com apenas um recurso para cada equipamento, as tarefas devem ser executadas uma por vez, gerando dessa forma filas (*queues*). Um dos propósitos da teoria das filas é justamente determinar o tempo despendido pelas tarefas, desde o tempo na fila até o seu processamento, ou seja, o tempo dentro do sistema.

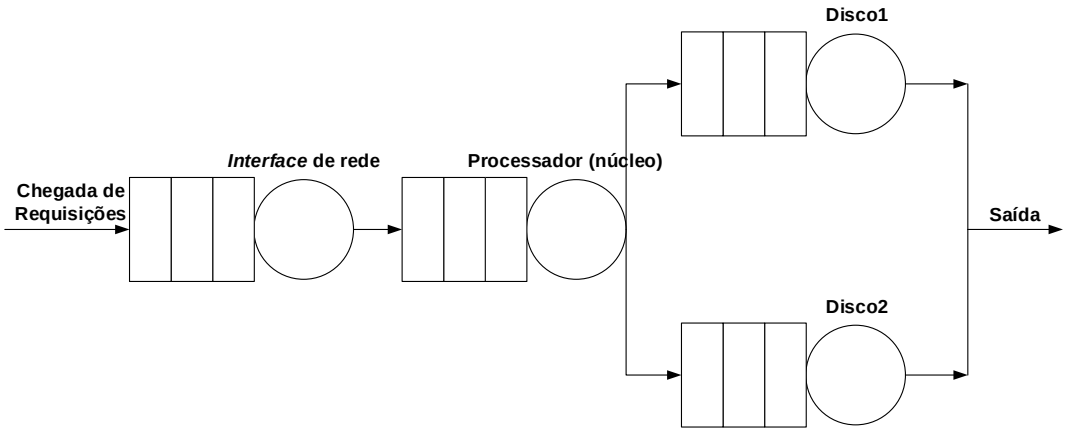


Figura 2.9: Um modelo de fila representando um servidor

Um dos teoremas mais comuns utilizados dentro do contexto da teoria das filas é a lei de *Little* (*Little's law*), a qual é adotada neste trabalho. Basicamente, o tempo médio de resposta de sistemas pode ser obtido através da Equação 2.1 (JAIN, 1990)

$$R = \frac{L}{\lambda} \tag{2.1}$$

onde R representa o tempo médio de resposta, L é o número médio de requisições e λ a taxa de chegada de tarefas. Esta relação pode ser aplicada em um sistema no qual o número de

tarefas de entrada é igual à quantidade de tarefas completadas.

2.5 AVALIAÇÃO DE DEPENDABILIDADE

Dependabilidade é caracterizada como a capacidade de um sistema de prover serviços de forma confiável (MACIEL et al., 2011). Usualmente, o conceito de dependabilidade abrange as seguintes métricas: disponibilidade, confiabilidade, segurança, confidencialidade, integridade e manutenabilidade. Os critérios estabelecidos para esses atributos (*attributes*) podem vir a implicar qualitativamente na avaliação dos sistemas (BERNARDI; MERSEGUER; PETRIU, 2012). A Figura 2.10 mostra uma organização sistemática dos três conceitos que envolvem esta propriedade.

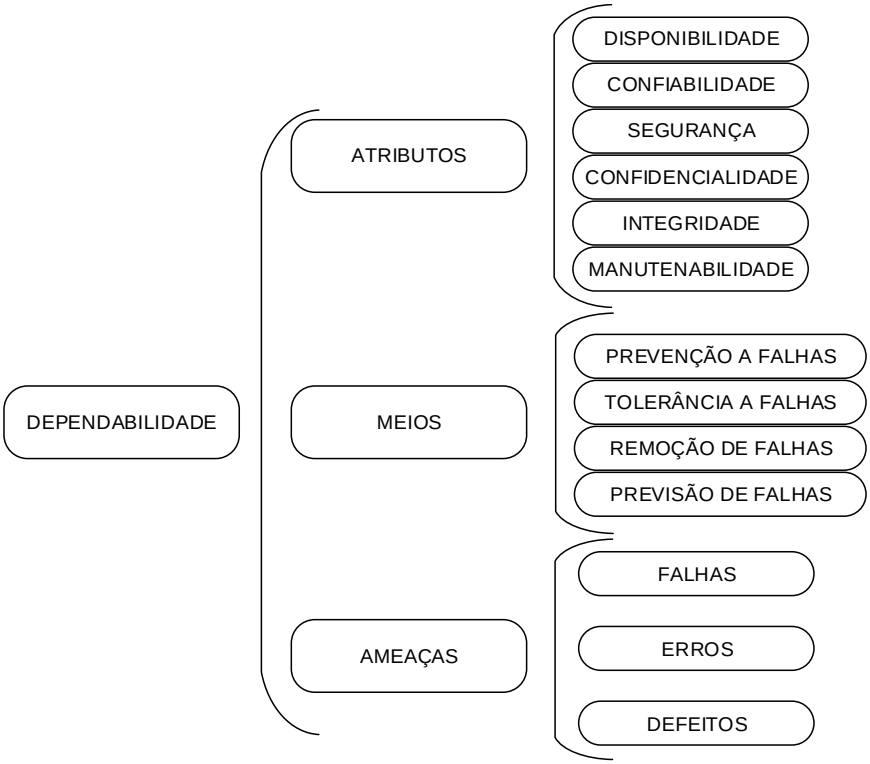


Figura 2.10: Conceitos de dependabilidade

Os atributos de dependabilidade definem a capacidade de um sistema em prover as funcionalidades específicas para as quais foi projetado. No entanto, algumas ameaças (*threats*) podem levar um sistema a apresentar um comportamento diferente do esperado. Especificamente, uma falta pode ser definida como a causa de um erro, o qual é parte de um estado do sistema e que pode levar à uma falha do sistema. Portanto, um erro pode ser definido como um estágio intermediário entre uma falta e uma falha (BERNARDI; MERSEGUER; PETRIU, 2012). Uma falta pode vir a ser considerada uma falha desde que se refira a um componente específico de um sistema (MACIEL et al., 2011).

Quatro técnicas - meios (*means*) - podem ser consideradas para definir o quão confiável é um sistema dentro do contexto de dependabilidade (BERNARDI; MERSEGUER; PETRIU, 2012). A prevenção de falhas (*fault prevention*) se atém a técnicas empregadas durante a fase de projeto e produção de um sistema (AVIZIENIS et al., 2001), evitando dessa forma futuras ocorrências indesejadas. Já a remoção de falhas (*fault removal*) ocorre durante as fases de desenvolvimento e operação, portanto, existem três etapas: verificação, diagnóstico e correção. No entanto, apesar do planejamento inicial, técnicas de tolerância (*fault tolerance*) devem ser aplicadas para preservar o serviço ofertado, mesmo na presença de falhas. Ainda considerando o aspecto de planejamento, isto é, prever (*fault forecasting*) um possível comportamento indesejado, a avaliação durante a operação do sistema busca em termos de probabilidade identificar se os atributos de dependabilidade são satisfeitos. Considerando o contexto deste trabalho, alguns desses atributos são explanados a seguir.

O atributo **confiabilidade** (*reliability*) representa a probabilidade de um sistema executar as funções para as quais foi projetado, dentro de um tempo específico, sem a ocorrência de falhas (MACIEL et al., 2011). Esta relação é expressa matematicamente através da Equação 2.2, na qual T é uma variável randômica e contínua, que representa o tempo para falha de um sistema. Para um dado valor de t, $R(t)$ é a probabilidade do tempo para falha ser maior ou igual a t (EBELING, 2004).

$$R(t) = P\{T \geq t\}, T \geq 0 \tag{2.2}$$

Portanto, se considerarmos $P\{T < t\}$, a probabilidade da ocorrência de falhas, até um instante t, pode ser obtida. De fato, a Equação 2.3 mostra essa relação, de forma que $F(t)$ representa a função de distribuição acumulada da distribuição de falhas (EBELING, 2004).

$$F(t) = 1 - R(t) = P\{T < t\}, T \geq 0 \tag{2.3}$$

A **disponibilidade** (*availability*) é descrita como a probabilidade de um determinado sistema estar em condições de funcionamento (MACIEL et al., 2011). Em particular, a disponibilidade de estado estacionário (*steady-state availability*) pode ser expressa em função do tempo médio para falha (*mean time to failure* - MTTF) e do tempo médio para reparo (*mean time to repair* - MTTR) (KANOUN; SPAINHOWER, 2008)

$$Availability = \frac{MTTF}{MTTF + MTTR} = \frac{uptime}{uptime + downtime} \tag{2.4}$$

onde $uptime$ representa o tempo operacional do sistema, enquanto que o $downtime$ está relacionado ao período de inatividade.

O tempo médio para falha especifica em quanto tempo um determinado sistema, ou subsistema, irá funcionar corretamente (MODARRES; KAMINSKIY; KRIVTSOV, 2009), ou seja, representa a expectativa de tempo para uma falha ser observada (BERNARDI; MERSEGUER;

PETRIU, 2012). Para estimar o MTTF se faz necessário o conhecimento da distribuição estatística dos valores de tempo para as falhas (*failure distribution function*) (KAPUR; PECHT, 2014). Por exemplo, para o caso da distribuição exponencial (*exponential probability distribution*), a qual possui uma taxa de falha constante (EBELING, 2004), o cálculo do MTTF segue o disposto na Equação 2.5, onde $f(t)$ representa a função densidade de probabilidade, e λ a taxa de falhas (*hazard rate*).

$$MTTF = \int_0^\infty R(t)dt = \int_0^\infty tf(t)dt = \int_0^\infty e^{-\lambda t}dt = \frac{1}{\lambda} \tag{2.5}$$

Manutenabilidade (*maintainability*) é definida como a probabilidade de que um sistema em falha ou componente vai ser restaurado ou reparado para uma específica condição dentro de um período de tempo (EBELING, 2004). De forma similar a confiabilidade, manutenabilidade é caracterizada por uma distribuição de probabilidade, no entanto, neste caso leva em consideração o tempo para reparo. A manutenabilidade é descrita pela Equação 2.6, a qual representa a probabilidade de que o reparo vai ser concluído dentro do tempo t, onde $h(t)$ é a função densidade de probabilidade.

$$P(T \leq t) = \int_0^t h(t)dt \tag{2.6}$$

O tempo médio para reparo (MTTR) usualmente é utilizado para quantificar a manutenabilidade, no entanto, assim como para o cálculo do MTTF, a distribuição estatística também deve ser levada em consideração (EBELING, 2004). A equação 2.7 representa o cálculo para obtenção do mesmo, onde $H(t)$ é a função de distribuição acumulada.

$$MTTR = \int_0^\infty th(t)dt = \int_0^\infty (1 - H(t))dt \tag{2.7}$$

Para a avaliação de dependabilidade o uso de modelos é amplamente adotado (BERNARDI; MERSEGUER; PETRIU, 2012). Basicamente, um modelo é a abstração de um sistema e tem como propósito possibilitar a compreensão antes da concepção de fato do mesmo. Um modelo de dependabilidade considera as abstrações necessárias para representar as falhas do sistema e suas consequências. A modelagem então pode ser definida de acordo com a interação ou a estrutura dos componentes de um sistema.

Para interações mais complexas, e que representem as dependências entre os sistemas, modelos de espaço de estado podem realizar uma representação mais precisa através da análise do comportamento dinâmico quando da ocorrência de eventos (MACIEL et al., 2011). Modelos combinatoriais estão relacionados às relações estruturais entre os elementos de um sistema, no entanto, levam em consideração a premissa de que a falha ou recuperação de um componente não é afetada pelo comportamento de outro elemento.

Diagramas de bloco de confiabilidade (RBD), árvores de falha (*fault tree*) e grafos de confiabilidade (*reliability graphs*) são modelos combinatoriais representativos, enquanto que cadeias de Markov (CTMC) e redes de *Petri* estocásticas (SPN) são os modelos baseados em

espaço de estados mais utilizados (MACIEL et al., 2011).

2.6 REDES DE *PETRI* ESTOCÁSTICAS

Nesta seção, será abordada e explanada uma das técnicas de análise de espaço de estado mencionada na Seção 2.5, a qual foi utilizada neste trabalho dentro do contexto de avaliação de performance. Inicialmente, o conceito original de redes de *Petri* é introduzido de forma a possibilitar a compreensão de seus fundamentos. Em seguida, a extensão redes de *Petri* estocástica generalizadas é apresentada. É importante ressaltar que, nos capítulos seguintes é usado o termo SPN para representar qualquer extensão deste formalismo.

2.6.1 Redes de *Petri*

Petri Nets (PN) são uma família de formalismos adequados para modelagem de diversos tipos de sistemas, uma vez que características como concorrência, sincronização, assincronismo, distribuição e não-determinismo são bem representadas (MURATA, 1989). Introduzida por Carl Adam Petri em 1962 (PETRI, 1962; BOLCH et al., 2006), redes de *Petri* não foram desenvolvidas originalmente para prover avaliação de desempenho, apesar de toda a sua potencialidade para representar sistemas complexos (FRANCÊS, 2003). Por ser uma ferramenta gráfica, redes de *Petri* pode ser utilizada como um auxílio de comunicação visual similar a gráficos de fluxo e diagramas de bloco (MURATA, 1989), possibilitando então a descrição das relações existentes entre condições e eventos (O'CONNOR; O'CONNOR; KLEYNER, 2012). Os elementos que constituem uma rede de *Petri* são demonstrados na Figura 2.11, os quais serão explanados a seguir.

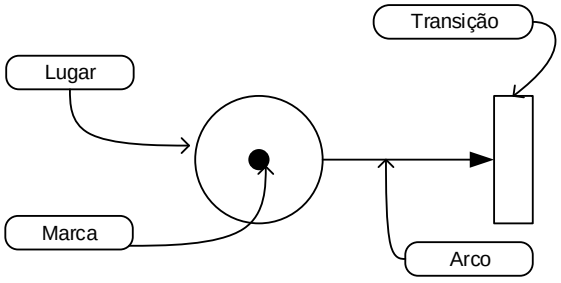


Figura 2.11: Elementos de uma rede de *Petri*

Em geral, uma rede de *Petri* é um grafo bipartido dirigido, o qual consiste em dois tipos de nós (*nodes*), denominados por lugar (*place*), e transição (*transition*). Graficamente, lugares são representados por um círculo ou elipse (REISIG, 2013), e são associados a um componente passivo destinado a retratar uma condição ou armazenar objetos (BAUSE; KRITZINGER, 2002). A mudança das condições de um sistema, o que pode ser visto também como

uma mudança de valores, é representada pelas transições, as quais são simbolizadas por um retângulo e são caracterizadas como o componente ativo das redes de *Petri* (REISIG, 2013).

Por ser um grafo bipartido, a conexão entre os elementos deve ser feita considerando os dois tipos de nós, isto é, um lugar pode apenas se conectar com uma transição e vice-versa (BAUSE; KRITZINGER, 2002). Lugares e transições são conectados diretamente através de arcos (*arcs*). Graficamente, um arco é representado por uma seta e não constitui um componente do sistema, mas apenas uma abstrata relação, como por exemplo, conexões lógicas (REISIG, 2013). Existem dois tipos de arcos: arcos de entrada (*input arcs*) e arcos de saída (*output arcs*). Arcos de entrada efetuam a conexão de um lugar de entrada (*input place*) para uma transição, enquanto que arcos de saída conectam uma transição para um lugar de saída (*output place*) (BOLCH et al., 2006).

Para uma transição ser executada se faz necessário que a mesma esteja habilitada. Uma transição está habilitada se todos os seus lugares de entrada estiverem com pelo menos uma marca (*token*). Representado graficamente por um ponto preto, o *token* retrata o estado em que o sistema se encontra em um determinado momento (O'CONNOR; O'CONNOR; KLEYNER, 2012). O disparo (*firing*) de uma transição habilitada representa a execução de uma ação, provocando então a absorção e geração de *tokens* nos lugares de entrada e saída (Figura 2.12), respectivamente, levando dessa forma o modelo para um novo estado de marcação (O'CONNOR; O'CONNOR; KLEYNER, 2012).

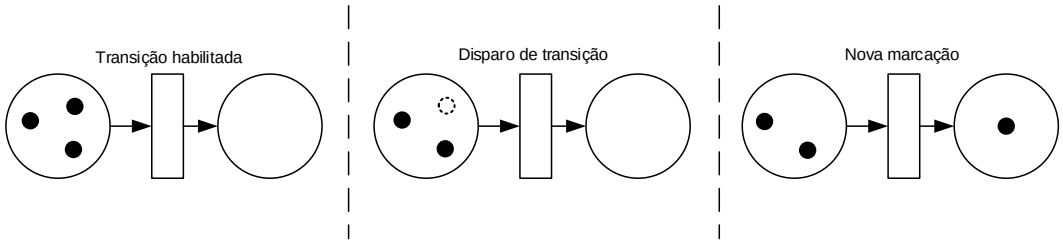


Figura 2.12: Disparo de uma transição

A definição formal das redes de *Petri* é demonstrada a seguir (MURATA, 1989):

Definição 2.1. *Uma rede de Petri é uma 5-tupla,* $PN = (P, T, F, M, M_0)$*, onde:*

- $P = \{p_1, p_2, ..., p_m\}$ *é o conjunto finito de lugares*;
- $T = \{t_1, t_2, ..., t_n\}$ *é o conjunto finito de transições*;
- $F \subseteq (P \times T) \cup (T \times P)$ *é o conjunto de arcos*;
- $M : F \to \aleph$ *é a função que atribui peso aos arcos*;
- $M_0 : P \to \aleph$ *é a marcação inicial, onde* $P \cap T = \emptyset$ e $P \cup T \neq \emptyset$.

As marcações alcançáveis e os passos de uma rede de *Petri* podem ser todos compilados em um grafo de alcançabilidade (*reachability graph* - RG). Neste grafo direcionado os nós representam as marcações alcançáveis [3], enquanto que as arestas refletem as transições entre as mesmas. A princípio, o grafo de alcançabilidade é um ponto de partida adequado para a análise da rede e verificação de algumas propriedades (comportamentais e estruturais), desde que apresente um número finito de marcações alcançáveis (REISIG, 2013). A Figura 2.13 ilustra uma rede de *Petri* hipotética e seu correspondente grafo de alcançabilidade.

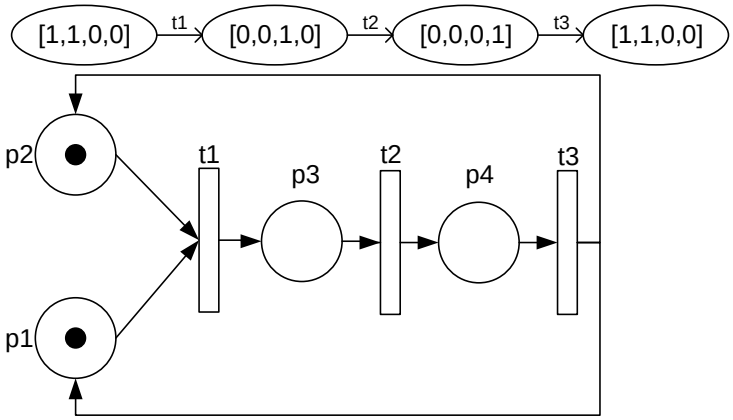


Figura 2.13: Rede de *Petri* e correspondente grafo de alcançabilidade

2.6.2 Redes de *Petri* Estocásticas Generalizadas

Redes de *Petri* demonstram ser adequadas para modelagem de sistemas computacionais devido a sua capacidade de representação de concorrência e sincronização. Por possibilitar a representação das relações entre eventos e condições, redes de *Petri* se mostraram um recurso bastante utilizado para avaliação das propriedades dos sistemas, como por exemplo, vivacidade [4] (*liveness*), *deadlock*[5] e consistência [6]. No entanto, como o tempo não é considerado em sua definição, análises de desempenho e dependabilidade tornam-se inviáveis. Esta demanda levou diversos autores a propor modificações na definição básica de forma a obter uma ferramenta de modelagem melhor aplicável à representação de sistemas reais (AJMONE MARSAN; CONTE; BALBO, 1984a).

Em particular, a inclusão de um tempo associado a uma transição especifica o atraso (*delay*) entre a habilitação e o disparo da mesma. Dentro deste contexto, ZUBEREK (1980)

[3] Alcançabilidade indica a possibilidade de uma determinada marcação ser atingida pelo disparo de um número finito de transições a partir de uma marcação inicial

[4] Uma rede de *Petri* é considerada *live* se, independente das marcações alcançáveis a partir de uma marcação inicial, for sempre possível disparar qualquer transição através de uma sequência de transições

[5] Uma rede de *Petri* possui *deadlock* quando a partir de uma determinada marcação, nenhuma transição está habilitada

[6] Uma rede de *Petri* é considerada consistente se a sequência de disparo de transições a partir de uma marcação inicial, retornar a esta mesma marcação inicial com todas as transições sendo disparadas ao menos uma vez

estabeleceu um tempo fixo de forma a modelar o desempenho de um sistema computacional em um nível específico, enquanto que MERLIN; FARBER (1976) introduziram as redes de *Petri* temporizadas (*timed Petri nets*) associando a cada transição um tempo máximo e mínimo para disparo. Já MOLLOY (1982) propôs um modelo de redes de *Petri* estocástica (SPN) no qual o tempo de disparo das transições é aleatório e exponencialmente distribuído.

A utilização da distribuição exponencial para a definição das especificações temporais torna esta extensão (SPN) bastante atrativa, principalmente devido a propriedade de perda de memória (*memoryless*). Esta propriedade torna desnecessária a distinção entre as distribuições do atraso atual e os que ainda estão por acontecer (BALBO, 2001), tornando dessa forma o gráfico de alcançabilidade de uma SPN limitada (*bounded*[7]) isomórfico a uma cadeia de *Markov* de tempo contínuo (CTMC) (MURATA, 1989; TRIVEDI, 2008). Este recurso permite, por exemplo, o cálculo da probabilidade de estado estacionário (*steady-state probabilities*) de uma marcação.

A introdução de SPNs possibilita então a junção dos conceitos de modelos gráficos e probabilísticos, tornando-se uma ferramenta bastante útil para a estimação de desempenho de um sistema, bem como uma alternativa à geração de cadeias de *Markov*, através da adoção de técnicas de simulação (TRIVEDI, 2008). Entretanto, sua limitação está associada ao fato de que o tamanho de um sistema pode aumentar a complexidade de sua representação gráfica. Consequentemente, o número de estados da cadeia de *Markov* associada também cresce indiscriminadamente. Portanto, SPNs devem ser usadas apenas para modelar sistemas com o espaço de estados limitado (AJMONE MARSAN; CONTE; BALBO, 1984a).

A seguir, a definição formal de redes de *Petri* estocásticas é apresentada (BAUSE; KRITZINGER, 2002):

Definição 2.2. *Formalmente, uma SPN é uma 2-tupla,* $SPN = (PN, \Lambda)$*, onde:*

- $PN = (P, T, F, M, M_0)$ *é formado pela rede de Petri abordada na Definição 2.1*;
- $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$ *é o conjunto de taxas, onde a taxa* λ_i *é associada à transição* t_i.

Porém, nem sempre é desejável associar um valor de tempo aleatório para cada transição. Ademais, a representação de atividades breves apenas do ponto de vista lógico pode ser particularmente conveniente se o número de estados da cadeia de *Markov* gerada for reduzido. Dito isso, AJMONE MARSAN; CONTE; BALBO (1984a) introduziu as redes de *Petri* generalizadas (*generalized stochastic Petri nets* - GSPNs), as quais possuem dois tipos de transições: temporizada, representada por um retângulo branco, e imediata, simbolizada por um retângulo preto. Transições imediatas, por definição, não possuem atrasos (*delay*), enquanto que as temporizadas são associadas a um tempo de distribuição exponencial, já mencionado na definição

[7] Uma rede é limitada (*bounded*) se em cada lugar da rede, o número de *tokens* nunca excede um valor k. Dessa forma, a rede é dita k-limitada

de SPNs. Vale ressaltar também que, transições imediatas, quando habilitadas, possuem precedência com relação à temporizadas (MARSAN et al., 1994).

Outras extensões foram propostas para GSPNs, das quais as mais relevantes para este trabalho são explanadas a seguir (BOLCH et al., 2006):

- **Arco inibidor:** um arco inibidor conecta um lugar a uma transição e é representado graficamente por uma linha com um círculo branco na extremidade oposta ao lugar. Quando o número de marcas existentes no lugar é igual ou maior que a multiplicidade condicionada ao arco, a transição é desabilitada;

- **Prioridades:** embora arcos inibidores possam ser usados para especificar relações de prioridade, tais atribuições são melhores definidas se introduzidas explicitamente no paradigma. Prioridades são especificadas por números inteiros associados às transições. Dessa forma, uma transição t_i pode ser habilitada apenas se a sua prioridade é maior que outras transições da rede, ou seja $t_i > t_n$;

- **Peso:** se são associados os pesos w_i e w_j para as respectivas transições imediatas, t_i e t_j, e apenas ambas estão habilitadas, a probabilidade de disparo de t_i é dada por $w_i/(w_i + w_j)$ (BAUSE; KRITZINGER, 2002);

- **Semântica de servidor:** este trabalho aborda as semânticas *single-server* e *infinite-server*. A habilitação de uma transição imediata *single-server* realiza o processamento (*delay*) de marcas de forma individual, ou seja, a ocorrência de um novo disparo está condicionada a finalização do atraso anterior. Já no que diz respeito à semântica *infinite-server*, a absorção de *tokens* é feita de forma paralela, portanto, à medida que a respectiva transição é habilitada a mesma pode ser disparada (MARSAN et al., 1994).

Considerando as extensões para GSPNs apresentadas, é possível então definir seu formalismo como se segue:

Definição 2.3. *Formalmente, uma GSPN é uma 8-tupla,* $GSPN = (P, T, I, O, H, \Pi, W, M_0)$*, onde:*

- $P = \{p_1, p_2, ..., p_m\}$ *é o conjunto finito de lugares*;

- $T = \{t_1, t_2, ..., t_n\}$ *é o conjunto finito de transições imediatas* (T_{im}) *e temporizadas* (T_{timed})*,* $P \cap T = \emptyset$ *e* $T = T_{im} \cup T_{timed}$;

- $\Pi : T \to \aleph$ *é a função de prioridade, onde* $\Pi(t) \geq 1$*, se* $t \in T_{im}$*, ou* $\Pi(t) = 0$*, se* $t \in T_{timed}$;

- $I, O, H : T \to Bag(P)$ *são as funções de entrada, saída e inibição, respectivamente, onde* $Bag(P)$ *é o multiconjunto de* $P(Bag(P) : P \to \aleph)$;

- $W : T \rightarrow \Re$ *é a função de atribuição de peso ou taxa, onde* $W(t) = w_t$, *se* $t \in T_{im}$, *ou* $W(t) = \lambda_t$, *se* $t \in T_{timed}$;
- $M_0 : P \rightarrow \aleph$ *é a marcação inicial, onde* $P \cap T = \emptyset$ *e* $P \cup T \neq \emptyset$.

A adição do conceito de transições imediatas através das GSPNs aumentou a capacidade de modelagem de sistemas reais, no entanto, sua análise é caracterizada por ser mais complexa do que em SPNs. De fato, o isomorfismo existente entre SPNs e CTMCs não acontece de forma similar para GSPNs devido à existência de dois tipos de marcações: voláteis (*vanishing*) e tangíveis (*tangible*) (MARSAN et al., 1994). Marcações voláteis são caracterizadas por representar estados quando da habilitação de ao menos uma transição imediata, enquanto que as tangíveis estão associadas às transições temporizadas.

Para uma dada GSPN, um grafo de alcançabilidade extendido (*extended reachability graph* - ERG) é gerado contendo as informações de ambos os tipos de marcações. Porém, para evitar uma descontinuidade estocástica as marcações voláteis devem ser eliminadas, a fim de obter o grafo de alcançabilidade isomórfico a uma cadeia de *Markov* correspondente (MARSAN et al., 1994). A eliminação das marcações *vanish* é um importante passo para geração da CTMC equivalente. Duas técnicas podem ser utilizadas: eliminação *on the fly* e pós-eliminação (*post-elimination*).

Para fins didáticos iremos abordar neste trabalho apenas a eliminação *on the fly*, a qual é demonstrada na Figura 2.14. Basicamente, a geração de marcações voláteis é evitada através do redirecionamento dos arcos para a marcação seguinte do grafo de alcançabilidade (BOLCH et al., 2006). É importante ressaltar que esta divisão (*splitting*) leva em consideração as probabilidades das transições imediatas envolvidas no processo, associando as mesmas às taxas dos arcos resultantes.

2.7 DIAGRAMA DE BLOCOS DE CONFIABILIDADE

Diagrama de blocos de confiabilidade (*reliability block diagram* - RBD) é uma representação gráfica que demonstra a combinação de sucessos ou falhas dos componentes de um sistema. Basicamente, representa a relação lógica de um sistema, subsistemas, e componentes, considerando seus valores de confiabilidade individualmente (KAPUR; PECHT, 2014; RAUSAND; ARNLJOT et al., 2004; KUO; ZUO, 2003). Ademais, embora tenha sido proposto inicialmente para o cálculo de confiabilidade, RBDs podem ser também utilizados para estimar outras métricas de dependabilidade, como disponibilidade e manutenabilidade (MACIEL et al., 2011).

Basicamente, RBDs consistem de componentes e suas relações lógicas, os quais são graficamente representados por retângulos e arcos, respectivamente. Além de conectar os componentes entre si, os arcos também são vinculados aos vértices de início (*begin*) e fim (*end*). O

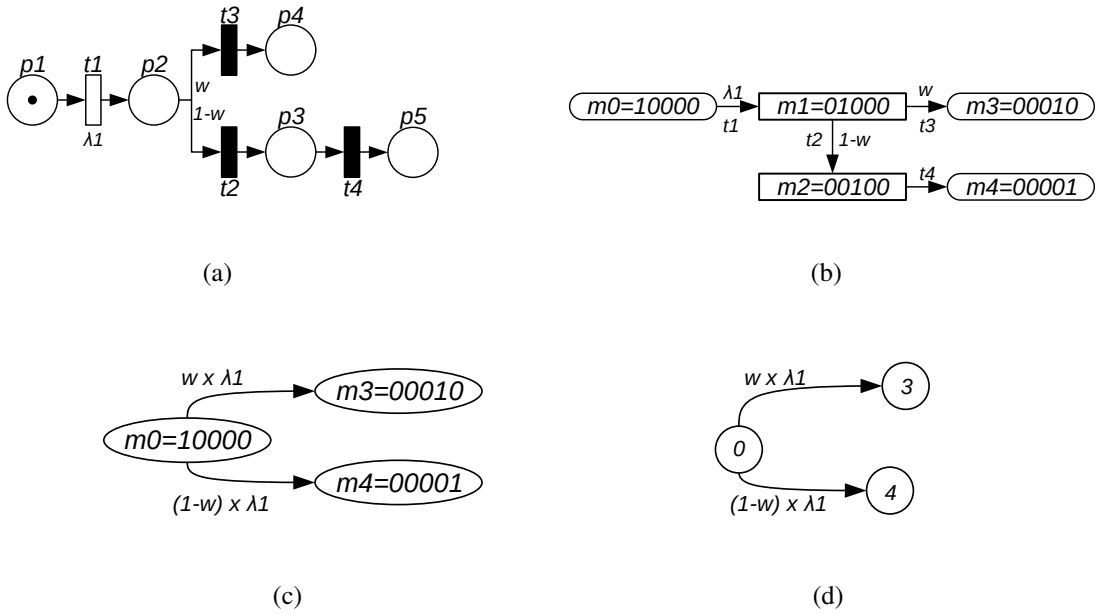


Figura 2.14: Remoção de marcações *vanishing* demonstrada por (a) uma GSPN, (b) ERG equivalente, (c) RG resultante, e (d) a CTMC correspondente (adaptada de BOLCH et al. (2006))

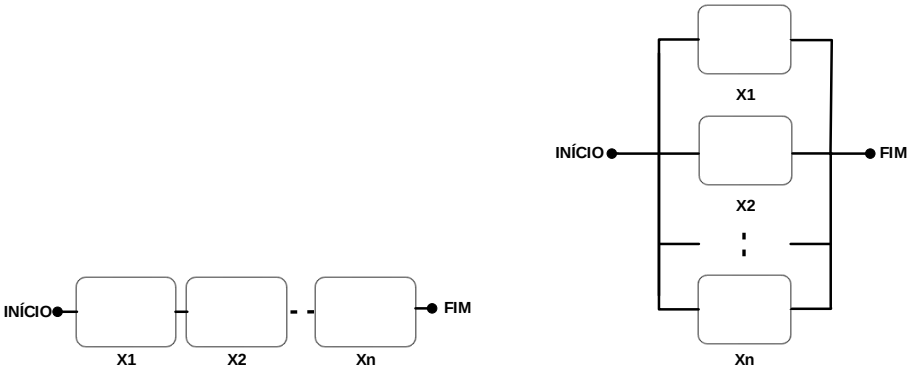
vértice de início é usualmente posicionado no lado esquerdo do modelo, enquanto que o vértice fim é disposto do lado oposto (MACIEL et al., 2011).

Usualmente, a disposição dos componentes no modelo corresponde ao arranjo físico dos itens de um sistema. No entanto, alguns casos fogem à regra, como por exemplo, quando a falha de um de dois resistores dispostos fisicamente em paralelo causa a falha de um sistema. Neste caso em específico, a modelagem adequada seria dois blocos em série.

RBDs são frequentemente utilizados para modelar o efeito de falhas em um sistema (MODARRES; KAMINSKIY; KRIVTSOV, 2009). Deve-se considerar um arranjo em série de um RBD (Figura 2.15(a)) quando a falha de um bloco resulta na falha de todo o sistema. Isso significa que todos os subsistemas devem estar operacionais para permitir a operação do sistema (KAPUR; PECHT, 2014). Já um sistema que mantém sua operacionalidade enquanto ao menos um de seus n componentes estiver funcional, é representado por arranjos paralelos (RAUSAND; ARNLJOT et al., 2004), o qual é demonstrado na Figura 2.15(b).

Diante da ocorrência de falha de algum componente, quando da existência de redundância, o comportamento redundante pode ser apresentado das seguintes maneiras (KAPUR; PECHT, 2014):

- ***Hot standby*:** equipamentos redundantes regidos por esse conceito são caracterizados por estarem ativados mesmo quando não solicitados. Portanto, ele possui a mesma taxa de falha que o componente que está de fato sendo utilizado. Arranjos paralelos usualmente consideram esse tipo de redundância;



(a) Série (b) Paralelo

Figura 2.15: Arranjos do *reliability block diagram* (RBD)

- ***Cold standby*:** neste caso em específico o componente em espera (*standby*) não falha enquanto não for solicitado, pois o mesmo está inativo até o momento de falha do componente principal;
- ***Warm standby*:** o componente redundante possui uma baixa taxa de falha comparado ao principal devido a fato de ser solicitado apenas em intervalos regulares (por exemplo, cópia de segurança de um sistema).

Assumindo n componentes (blocos) em série, a disponibilidade (A_s) ou confiabilidade (R_s) de um sistema pode ser estimada como

$$D_s = \prod_{i=1}^{n} d_i \tag{2.8}$$

onde d_i pode representar a disponibilidade (A_i) ou confiabilidade (R_i) do componente i (EBELING, 2004).

Já em um sistema com n componentes paralelos, a disponibilidade ou confiabilidade é calculada como

$$D_s = 1 - \prod_{i=1}^{n} [1 - d_i] \tag{2.9}$$

onde d_i corresponde a disponibilidade (A_i) ou confiabilidade (R_i) do componente i (EBELING, 2004).

2.8 CONSIDERAÇÕES FINAIS

Este capítulo apresentou alguns dos conceitos fundamentais necessários para o entendimento deste trabalho. Inicialmente, explorou detalhadamente os dispositivos de armazenamento abordados de forma a permitir a compreensão da função dos seus componentes. Dessa

forma, foi explanada a influência que esses elementos possuem no que diz respeito ao desempenho e disponibilidade de HDDs e SSDs. Em seguida, apresentou as políticas para dispositivos de armazenamento híbridos (SSDasCache e SSDRandom) adotadas para os experimentos demonstrados no Capítulo 5. Posteriormente, este capítulo abordou os aspectos de desempenho e dependabilidade, os quais são fundamentais para as avaliações a serem efetuadas. Especificamente, no que concerne a desempenho, a lei de *Little*, os conceitos de carga real e sintética, e os benefícios da modelagem analítica, são fundamentais para o entendimento deste trabalho. O atributo disponibilidade, junto com o conceito de tempo médio para falha (MTTF), e tempo médio para reparo (MTTR), são os fundamentos de dependabilidade abordados nos capítulos a seguir. Finalmente, os formalismos matemáticos SPNs e RBDs foram apresentados para compreensão da modelagem proposta.

3

METODOLOGIA E ARQUITETURAS

Este capítulo apresenta a metodologia adotada para a modelagem de desempenho e disponibilidade de sistemas de armazenamento híbridos, detalhando os processos e técnicas para a experimentação de diferentes políticas de armazenamento híbrido, para demonstrar a viabilidade da proposta deste trabalho. Inicialmente (Seção 3.1), apresenta a metodologia geral adotada, na qual são especificadas as etapas (processos) necessárias para atingir o objetivo desta abordagem. Posteriormente, introduz a técnica de planejamento de experimentos utilizada e os experimentos realizados neste trabalho (Seção 3.1.1). Em seguida, as Seções 3.2 e 3.3 denotam respectivamente os requisitos adotados para plataforma de computação em nuvem assumida (*OpenStack Swift*), e o simulador de arquiteturas de armazenamento *DiskSim*, o qual é utilizado para a validação dos modelos analíticos (SPN), bem como para os experimentos de desempenho.

A Figura 3.1 apresenta os elementos utilizados para a representação em alto nível (fluxogramas) da metodologia proposta. Mais especificamente, os processos representam as etapas que são executadas (Figura 3.1(a)). Etapas que se combinam, mas que são apresentadas em outro fluxograma, constituem um subprocesso (Figura 3.1(b)). A Figura 3.1(c) ilustra quais informações (dados) são utilizadas para a execução de um determinado processo. Já a Figura 3.1(d) retrata uma decisão, a qual ditará a próxima etapa. O elemento denotado pela Figura 3.1(e) representa os resultados (documento) de um determinado processo. A Figura 3.1(f) ilustra o elemento que demonstra o início e fim do fluxograma.

(a) (b) (c) (d) (e) (f)

Figura 3.1: Elementos de um fluxograma

3.1 METODOLOGIA DE MODELAGEM E AVALIAÇÃO

Esta seção apresenta a metodologia adotada para modelagem de desempenho e disponibilidade de sistemas de armazenamento híbridos e para avaliação dos experimentos realizados neste trabalho. A metodologia proposta consiste no levantamento de requisitos para a concepção de modelos formais de espaço de estados e combinatoriais. Propõe ainda a validação dos modelos de desempenho para em seguida realizar um planejamento de experimento que demonstre a viabilidade da abordagem. Um experimento também é proposto para a avaliação de dependabilidade de sistemas de armazenamento híbridos, através do modelo RBD concebido. Em seguida, é adotada uma avaliação que considera todos os aspectos para análise mencionados.

A Figura 3.2 demonstra um fluxograma detalhado das etapas, as quais são melhores explanadas a seguir:

- **Entendimento do sistema de armazenamento adotado:** a concepção dos modelos analíticos concebidos neste trabalho segue a arquitetura de armazenamento de uma plataforma de computação em nuvem. Este processo diz respeito à análise dos requisitos relacionados à redundância, distribuição e sincronização dos dados submetidos e requisitados a este sistema de computação em nuvem. A Seção 3.2 descreve com mais detalhes as características da plataforma adotada;

- **Definição de métricas de interesse:** neste processo são definidas as métricas adotadas para validação e experimentação dos modelos analíticos. Tempo médio de resposta, vazão, disponibilidade e *downtime* são as métricas adotadas de acordo com a característica a ser avaliada (desempenho ou dependabilidade). A escolha das métricas leva em consideração o objeto de estudo deste trabalho (dispositivos de armazenamento), bem como requisitos da nuvem assumida, por exemplo, redundância;

- **Geração de modelos de desempenho e dependabilidade:** estes processos denotam os processos de modelagem, os quais levam em consideração o levantamento de requisitos da plataforma de computação em nuvem e a definição das métricas a serem analisadas. Para os modelos de desempenho foi adotada a modelagem estocástica com SPN, enquanto que RBDs são adotados para a análise de disponibilidade. Os modelos de desempenho representam a requisição e execução de operações de leitura, escrita e *mixed* (escrita e leitura). Já o modelo combinatorial concebido é constituído pelos componentes da arquitetura de armazenamento assumida, dos quais é levado em consideração o tempo médio para falha (MTTF) e o tempo médio para reparo (MTTR). A concepção dos modelos, considerações e ferramentas adotadas são detalhadas no Capítulo 4;

- **Análise quantitativa:** este processo analisa se os resultados obtidos das métricas

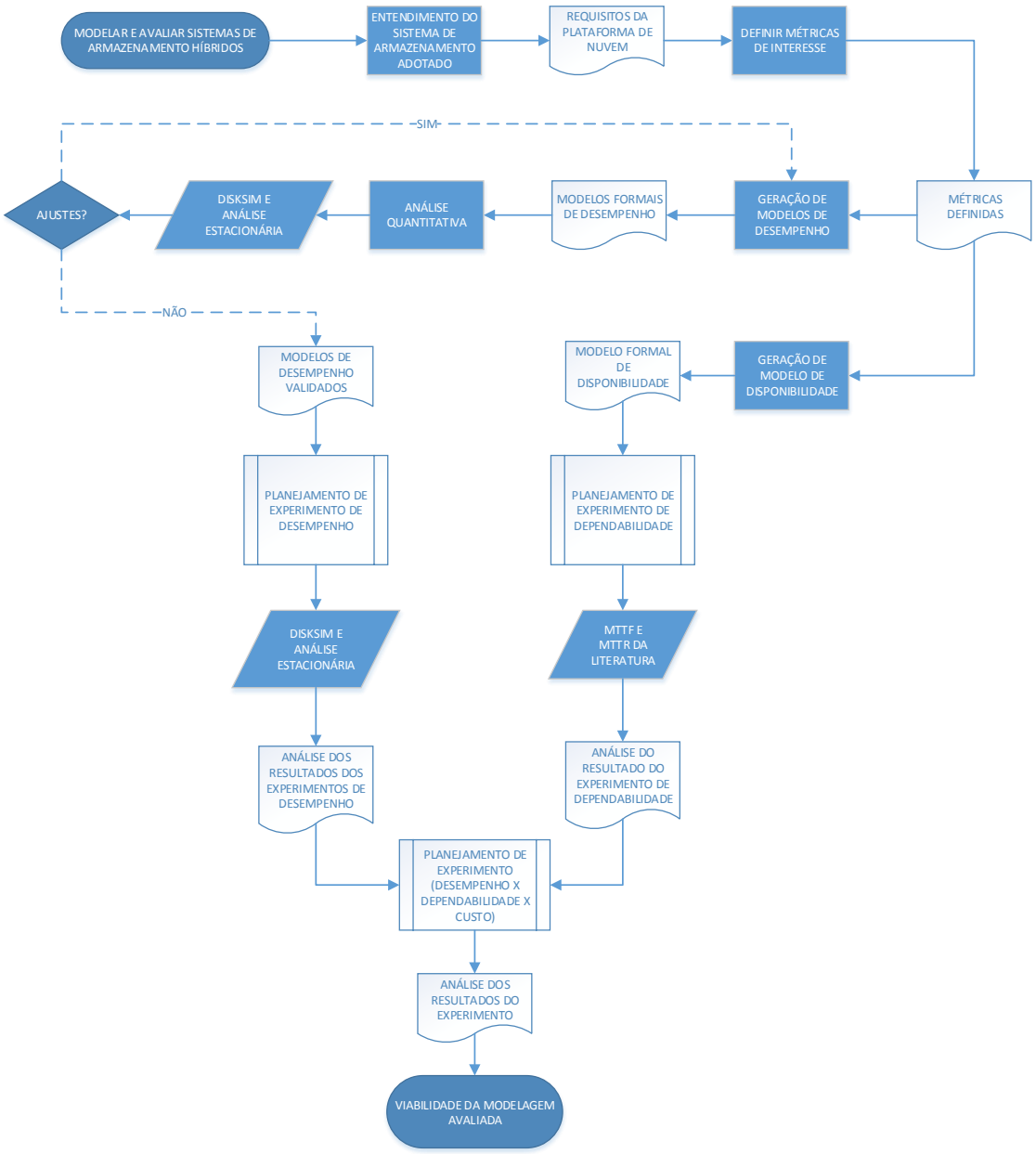


Figura 3.2: Metodologia da modelagem e experimentos

de desempenho estimadas através dos modelos SPN são condizentes com os valores de simulações de dispositivos de armazenamento. Para isso, os mesmos valores adotados em um simulador, para a execução de operações de escrita e leitura, para cada componente de armazenamento são também assumidos nos modelos de desempenho. Da mesma forma, o tempo entre requisições também é similar. Parâmetros como: modelo do dispositivo de armazenamento, tamanho dos objetos submetidos, capacidade e padrão das requisições (sequencial ou aleatória), também são levados em consideração. A comparação então é realizada entre o valores das análises estacionárias dos modelos SPN e das simulações de dispositivos de armazenamento, levando em consideração a redundância de uma plataforma de computação em nuvem. Testes t emparelhados são realizados para validar os modelos concebidos para o cenário descrito. A invalidação conduz novamente para a geração dos modelos de desempenho, onde são realizadas as modificações necessárias. A Seção 3.3 denota o simulador adotado neste trabalho;

- **Planejamento de experimentos de desempenho:** esta etapa considera validados os modelos formais baseados em espaço de estados (SPN) concebidos. Similar a etapa de validação, este processo leva em consideração valores advindos de um simulador de dispositivos de armazenamento para a execução de operações de escrita e leitura, de forma a possibilitar a realização de experimentos envolvendo tecnologias tradicionais (HDD e SSD) e dispositivos híbridos com diferentes políticas de armazenamento. Os experimentos de desempenho são realizados de acordo com o tipo de operação requisitada (leitura, escrita ou *mixed*) e levam em consideração um primeiro experimento, o qual é realizado para identificar os fatores mais significativos para as métricas adotadas. Os fatores analisados são: tamanho do objeto, tecnologia de armazenamento, capacidade de armazenamento, tipo da operação e o número de nós. A Subseção 3.1.1.1 detalha a metodologia abordada para a experimentação mencionada;

- **Planejamento de experimento de dependabilidade:** corresponde ao processo de estimar a disponibilidade e o *downtime* de diferentes arquiteturas e tecnologias de armazenamento. Além disso, também proveem os efeitos principais dos fatores considerados para esse experimento. Os fatores analisados são: número de nós, número de proxies e tecnologia de armazenamento. Valores de MTTF e MTTR obtidos na literatura são adotados para os componentes dos diagramas de blocos de confiabilidade gerados. Os subprocessos relacionados a esta etapa são descritos na Subseção 3.1.1.2;

- **Planejamento de experimento (desempenho x dependabilidade x custo):** este processo proporciona uma análise que envolve três aspectos: desempenho, dependabilidade e custo. Para isso, é estimado um único valor para cada configuração

(tratamento), a qual é correspondente a uma determinada arquitetura de armazenamento. Isto é possível através da normatização das métricas e cálculo da distância Euclidiana para cada tratamento. Os valores são obtidos através dos modelos SPN e RBD. O custo dos equipamentos também é considerado neste experimento, o qual é explanado na Subseção 3.1.1.3.

É importante ressaltar que considerando os modelos concebidos neste trabalho, essa metodologia de planejamento de experimentos pode ser utilizada para a avaliação de políticas de armazenamento híbridos, além de auxiliar em decisões de projeto, como por exemplo, a configuração da arquitetura de armazenamento a ser adotada por uma determinada plataforma. Esta abordagem é limitada pela necessidade de um embasamento suficiente para utilização dos modelos analíticos concebidos.

3.1.1 Metodologia de Avaliação

Esta seção explana a metodologia aplicada para os experimentos de desempenho e dependabilididade efetuados neste trabalho. Para isto, adotamos uma técnica de planejamento (*design of experiment* - DoE) que possibilita o estudo simultâneo dos fatores considerados nos experimentos (JOHANNESSON; PERJONS, 2014). Dessa forma, abordamos o escopo inteiro de experimentação através da combinação das variáveis adotadas em um único estudo. Portanto, diminuímos drasticamente a quantidade de testes requeridos para as análises realizadas.

Este trabalho adota uma abordagem fatorial, para cada um dos experimentos, onde cada tratamento é a combinação única dos níveis dos fatores (OEHLERT, 2010). Fatores são variáveis (parâmetros) que podem ser controladas durante o experimento, enquanto que os níveis representam os valores que cada fator pode assumir (JOHANNESSON; PERJONS, 2014). Para uma abordagem fatorial, a quantidade de tratamentos (T) para um determinado experimento é definida pela Equação 3.1, onde l representa a quantidade de níveis de um fator (i) específico, dentre um total de k fatores.

$$T = \prod_{i=1}^{k} l_i \tag{3.1}$$

Ambos os experimentos, desempenho e dependabilidade, avaliam a magnitude (efeito) de um fator com respeito às métricas adotadas. O efeito de um fator é definido como a alteração produzida nas respostas devido à mudança de seus respectivos níveis (MONTGOMERY; RUNGER, 2014), também chamado como *main effect*. Por exemplo, considerando um experimento com dois fatores, com dois níveis cada (A_{baixo}, A_{alto} e B_{baixo}, B_{alto}), o efeito principal de um determinado fator A é a diferença entre a média dos resultados dos tratamentos que envolvem os níveis *alto* e *baixo* de A (Equação 3.2).

$$A = \frac{A_{alto}B_{baixo} + A_{alto}B_{alto}}{2} - \frac{A_{baixo}B_{baixo} + A_{baixo}B_{alto}}{2} \tag{3.2}$$

Este trabalho também estima a interação entre os fatores. Para isso, os efeitos de um fator A sobre os diferentes níveis de um fator B é calculado de acordo com a Equação 3.3.

$$AB = \frac{A_{alto}B_{baixo} + A_{baixo}B_{alto}}{2} - \frac{A_{baixo}B_{baixo} + A_{alto}B_{alto}}{2} \tag{3.3}$$

3.1.1.1 Experimentos de Desempenho

Esta subseção apresenta a metodologia adotada para a realização dos experimentos de desempenho. A Figura 3.3 ilustra as etapas e processos necessários, bem como seus respectivos resultados (documentos), os quais são explanados a seguir:

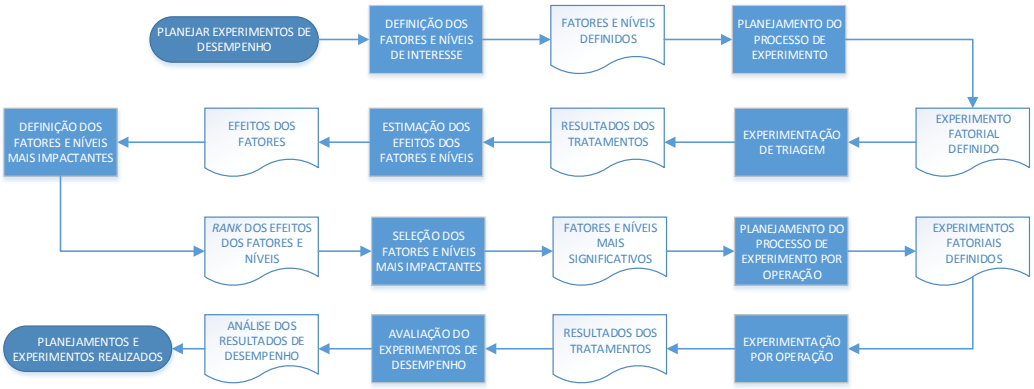


Figura 3.3: Planejamento dos experimentos de desempenho

- **Definição de fatores e níveis de interesse:** este processo define os fatores e níveis considerados significativos para a experimentação de sistemas de armazenamento. Logo, a capacidade de armazenamento, tamanho do objeto, tipo da operação, tecnologia e quantidade de nós são os fatores avaliados, assim como seus respectivos níveis;

- **Planejamento do processo de experimento:** este processo define a abordagem de experimentação utilizada. Para os experimentos de desempenho adotamos um planejamento fatorial, o qual proporciona resultados de acordo com a combinação de todos os níveis e fatores considerados na etapa anterior;

- **Experimentação de triagem, estimação, definição e seleção de fatores e níveis:** estas etapas correspondem ao experimento realizado para estimar a magnitude e direção dos efeitos dos fatores (MONTGOMERY; RUNGER, 2014). Inicialmente, estima os resultados dos tratamentos de acordo com as combinações definidas pela abordagem fatorial (*experimentação de triagem*), através dos resultados da análise estacionária dos modelos SPN concebidos. Em seguida, calcula os efeitos principais

e interações dos fatores (*estimação dos efeitos dos fatores*). Este estudo de caracterização (*screening experiment*) auxilia a definir as variáveis críticas do processo (*definição dos fatores e níveis mais impactantes*), bem como a direção de ajustes necessários para o objetivo do trabalho, através de um *rank* ordenado pelo níveis de efeito. Portanto, esta abordagem auxilia a selecionar, de acordo com os efeitos principais e interações, quais fatores e respectivos níveis são significantes ao objeto de estudo (*seleção dos fatores e níveis mais impactantes*);

- **Planejamento do processo de experimento por operação:** este processo define a abordagem adotada para os demais experimentos de desempenho realizados neste trabalho. Ademais, considera um planejamento fatorial, no qual os níveis dos fatores são definidos de acordo com os resultados do experimento de triagem (*screening experiment*). Além disso, realiza os experimentos de acordo com o tipo de operação requisitada (escrita, leitura ou *mixed*), e avalia dispositivos híbridos com diferentes políticas de armazenamento;

- **Experimentação por operação e avaliação:** estas etapas estimam os tratamentos definidos de acordo com o planejamento fatorial, e um experimento por cada tipo de operação é realizado (*experimentação por operação*). Além disso, analisa as diferentes tecnologias e políticas de armazenamento, para dispositivos híbridos, de acordo com os resultados obtidos (*análise dos resultados de desempenho*). Esta avaliação considera os parâmetros definidos para as cargas de trabalho submetidas, de maneira a identificar os pontos fortes e fracos de cada tecnologia de armazenamento. Os valores de tempo médio de resposta e IOPS são obtidos através dos modelos SPN propostos neste trabalho.

3.1.1.2 Experimento de Dependabilidade

Esta subseção denota a metodologia para a experimentação de sistemas de armazenamento híbridos dentro do contexto de dependabilidade. A Figura 3.4 demonstra os processos necessários para o planejamento do experimento, bem como para a análise dos resultados. A seguir, abordamos processos e etapas necessárias para a execução da metodologia proposta:

- **Definição de fatores e níveis de interesse:** este processo define os fatores e níveis adotados para avaliar suas respectivas influências no que concerne à dependabilidade. Para este experimento, o foco deste trabalho são os efeitos ocasionados nas métricas devido a quantidade de elementos redundantes e o tempo de vida dos mesmos. Logo, os níveis dos fatores *proxy* e nós de armazenamento são referentes à quantidade de componentes em paralelo, considerando os requisitos da plataforma de computação em nuvem adotada. A tecnologia dos dispositivos de armazenamento

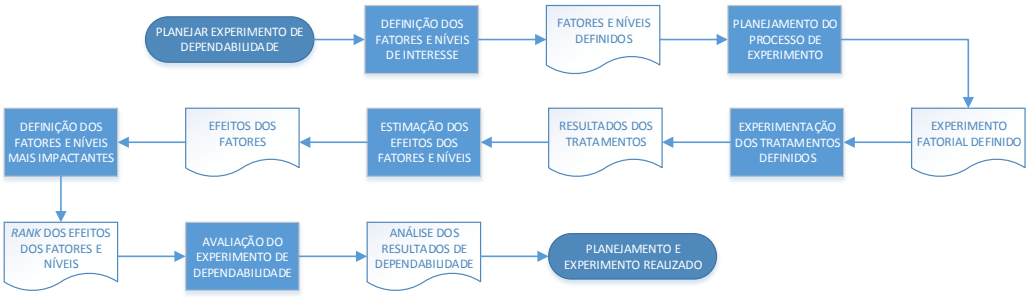


Figura 3.4: Planejamento do experimento de dependabilidade

também é um fator analisado devido à diferença do tempo de vida entre HDDs e SSDs, além da redundância obtida quando da adoção de dispositivos híbridos;

- **Planejamento do processo de experimento:** este processo define a abordagem de experimentação a ser assumida. Similar aos experimentos de desempenho, um planejamento fatorial é adotado para dispor todas as combinações possíveis entre os níveis dos fatores;

- **Experimentação dos tratamentos definidos:** este processo estima os resultados dos tratamentos definidos pelo planejamento de experimento fatorial. Para isso, são considerados o tempo médio para a falha (MTTF) e o tempo médio para reparo (MTTR) de cada componente. Os resultados para as métricas adotadas (disponibilidade e *downtime*) são obtidos através da análise do modelo RBD concebido neste trabalho;

- **Estimação, definição e avaliação dos efeitos dos fatores:** estes processos consistem em prover mecanismos para a avaliação da influência dos fatores no que diz respeito aos resultados de dependabilidade referentes aos tratamentos do processo anterior. Mais especificamente, os efeitos principais e interações dos fatores são calculados (*estimação dos efeitos dos fatores*) e ordenados através de um *rank* (*definição dos fatores e níveis mais impactantes*). Dessa forma, possibilita identificar a diferença de impacto entre fatores e interações, levando em consideração a redundância e as tecnologias de armazenamento adotadas (*avaliação do experimento de dependabilidade*).

3.1.1.3 Experimento (Desempenho x Dependabilidade x Custo)

Esta subseção aborda a metodologia proposta para a avaliação concomitante de desempenho, dependabilidade e custo de sistemas de armazenamento híbridos. A Figura 3.5 ilustra a metodologia adotada. Os processos e etapas são descritos a seguir:

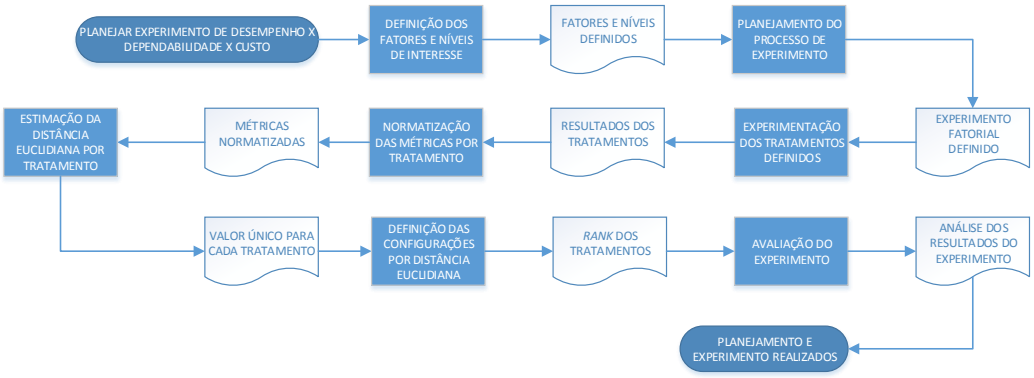


Figura 3.5: Planejamento do experimento (desempenho, dependabilidade e custo)

- **Definição de fatores e níveis de interesse:** corresponde ao processo que define os fatores e níveis significativos para a análise de desempenho, dependabilidade e custo. Portanto, é considerado para este experimento os fatores tecnologia de armazenamento, com a política de armazenamento especificada, quantidade de nós de armazenamento e o número de *proxies*;

- **Planejamento do processo de experimento:** similar aos experimentos exclusivos de desempenho e dependabilidade, este processo define um planejamento fatorial como a abordagem de experimentação a ser adotada;

- **Experimentação dos tratamentos definidos:** estima os resultados dos tratamentos para as métricas consideradas. A análise estacionária dos modelos SPN proveem os valores do tempo médio de resposta, enquanto que RBDs estimam o *downtime* de cada um dos tratamentos. O custo dos equipamentos também é levado em consideração;

- **Normatização das métricas por tratamento:** equipara as métricas de forma que estejam na mesma escala. Portanto, para que possuam a mesma magnitude, inicialmente uma normatização é realizada (JAIN, 1990). Para isso, a seguinte equação é adotada:

$$N(X) = \frac{X_i - Min(X)}{Max(X) - Min(X)}, \tag{3.4}$$

  onde X_i representa o valor da métrica X, relacionada ao tratamento i. Ademais, $Max(X)$ e $Min(X)$ denotam respectivamente o maior e o menor valor da métrica X no que concerne a todos os tratamentos do experimento;

- **Estimação da distância *Euclidiana* por tratamento:** adota a técnica de distância *Euclidiana* (ED) para encontrar um valor único para cada tratamento, que representa

a composição das métricas envolvidas no experimento. A distância é calculada entre dois pontos em um espaço de três dimensões (ALSUHIBANY et al., 2016; BOUHMALA, 2016). Um ponto é o resultado (tempo médio de resposta, *downtime*, custo) de um tratamento com os respectivos níveis para cada fator, e o outro ponto é a origem (0,0,0). Dessa forma, esta análise busca identificar os tratamentos com os menores valores de distância devido às métricas que foram adotadas, isto é, tempos e custos menores são mais adequados. O cálculo da ED para cada tratamento é feito através da seguinte equação:

$$D(Z,O) = \sqrt{\sum_{j=1}^{w} (Z_i^j - O^j)^2}, \tag{3.5}$$

onde Z_i^j e O^j representam dois pontos em um espaço de w dimensões. Portanto, para este trabalho, Z é composto por 3 métricas, cujos valores são referentes ao tratamento i. Ademais, j identifica a dimensão (métrica) a qual se refere o cálculo. O ponto O caracteriza a origem;

- **Definição das configurações por distância *Euclidiana*:** corresponde à ordenação dos tratamentos, de acordo com as EDs estimadas, de forma a permitir a interpretação dos resultados obtidos;

- **Avaliação do experimento:** investiga a influência dos níveis dos fatores no que concerne aos resultados obtidos das EDs. Adicionalmente, analisa a influência individual do desempenho, da dependabilidade, e do custo dos elementos abordados, de acordo com os requisitos de um sistema de armazenamento híbridos.

3.2 ARQUITETURA DO SISTEMA DE ARMAZENAMENTO

Esta seção apresenta o sistema de armazenamento adotado neste trabalho, o qual é baseado na arquitetura do módulo de armazenamento (*back-end*) da plataforma *OpenStack Swift*, a qual é amplamente adotada por diversos provedores de serviço (ROSADO; BERNARDINO, 2014; BISWAS; PATWA; SANDHU, 2015). A Figura 3.6 ilustra a arquitetura mencionada, relativa ao armazenamento de objetos, quando da chegada de requisições à *cloud*. A arquitetura padrão do *OpenStack Swift* assume três nós ativos para armazenamento, garantindo dessa forma excelentes resultados no que diz respeito à disponibilidade (KAPADIA; VARMA; RAJANA, 2014). *OpenStack Swift* é capaz de replicar cada objeto, por exemplo, um registro de dados, em seus nós de armazenamento (*storage nodes*). Portanto, quando um nó falha, os outros nós mantêm a operacionalidade do sistema (WANG; LI, 2015).

Em geral, as réplicas geradas pelo *Swift* são armazenadas por padrão em três nós de armazenamento diferentes (KAPADIA; VARMA; RAJANA, 2014). Esta quantidade específica

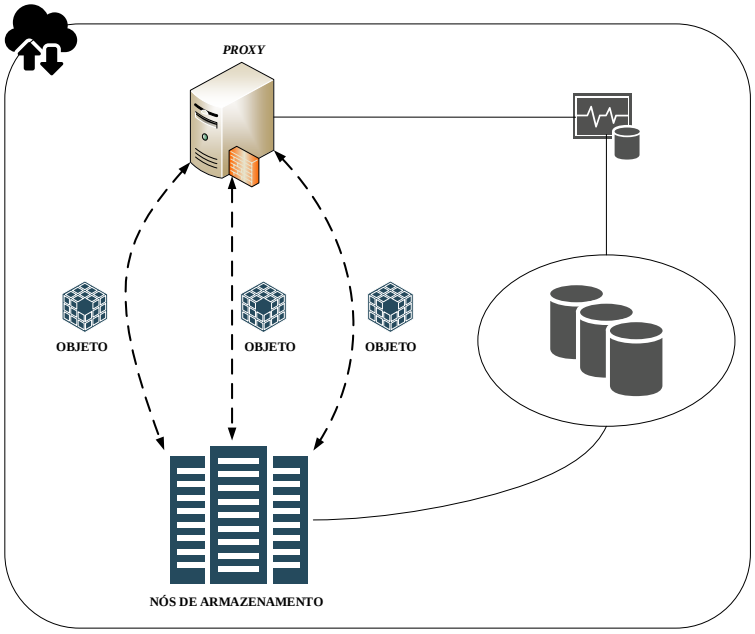


Figura 3.6: Arquitetura do módulo de armazenamento do *OpenStack Swift*

evita o gargalo de sincronização observado em CHEKAM et al. (2016), no qual o processo de sincronização é significativamente atrasado, além de produzir uma expressiva sobrecarga (*overhead*) quando o número de objetos replicados é maior do que 3.

Além disso, sua alta escalabilidade proporciona melhores índices de desempenho, os quais são alcançados através da distribuição da carga de trabalho por todos os nós de armazenamento (KHEDHER, 2015). Estes requisitos são considerados neste trabalho, o qual também leva em consideração a possibilidade da utilização de *hardwares* pouco dispendiosos.

3.3 ARQUITETURA DO SIMULADOR

Esta seção apresenta o ambiente utilizado para simular condições de geração e processamento de requisições similares ao módulo de armazenamento da plataforma de computação em nuvem considerada, com o propósito de avaliar diferentes configurações e políticas de armazenamento.

Para isso, este trabalho adota o simulador *DiskSim* (BUCY et al., 2008), com a extensão para SSDs (AGRAWAL et al., 2008), para obter os tempos individuais de leitura e escrita dos equipamentos de armazenamento, os quais são utilizados nos modelos SPN criados. Este simulador é também adotado para validação dos modelos concebidos (Seção 5.2.1). O *DiskSim* é um configurável simulador de sistemas de armazenamento desenvolvido para suportar pesquisas considerando vários aspectos da arquitetura dos subsistemas desses dispositivos. Em razão disso, possui módulos configuráveis que representam os dispositivos e controladores de armazenamento, as interfaces de comunicação, *drivers*, requisições, *cache*, e políticas de armazenagem. Dessa maneira, *DiskSim* tem sido utilizado em diversos estudos (KIM et al., 2011;

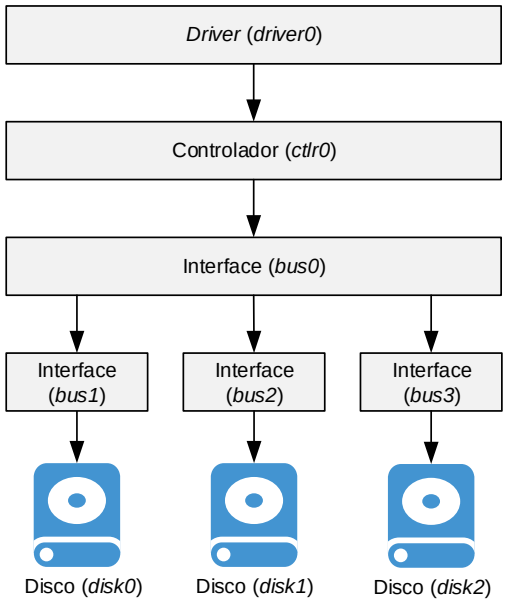


Figura 3.7: Arquitetura de um sistema de armazenamento no *DiskSim*

LI; LEE; LUI, 2013; WU; HE, 2012; EL MAGHRAOUI et al., 2010) para entender a relação de desempenho entre os dispositivos de armazenamento e todo o sistema em que os mesmos estão inseridos, além de avaliar novas arquiteturas.

Para este trabalho, desenvolvemos o Script 3.1, o qual foi implementado utilizando a linguagem de *script* do *DiskSim*, com o propósito de possibilitar simulações adequadas aos experimentos realizados, e recebe como entrada: os dispositivos de armazenamento (*storages*) a serem avaliados, a quantidade e disposição dos controladores e interfaces (*buses*), o *driver* a ser utilizado, além das especificações da carga de trabalho a ser aplicada. Como saída, o algoritmo fornece valores da vazão (IOPS) e tempo médio de resposta, tanto individualmente (módulos e dispositivos de armazenamento), quanto de todo o sistema.

A arquitetura e os componentes definidos no *script* apresentado (linha 2 a 16) caracterizam três nós de armazenamento e são ilustrados na Figura 3.7. É importante ressaltar que, apesar da ilustração conter apenas discos (HDDs), também foram implementados *scripts* para SDDs e dispositivos híbridos (SSD + HDD). No entanto, por limitação do simulador, dispositivos de armazenamento híbridos não podem ser simulados de acordo com arquiteturas de armazenamento redundante, ou seja, a simulação é restrita à apenas uma composição (HDD + SSD). A variável y (linha 21) denota a quantidade de requisições submetidas aos dispositivos de armazenamento, enquanto que x (linha 18) representa a quantidade de clientes concorrentes no sistema. O tempo entre requisições segue uma distribuição exponencial, na qual i (linha 26) define o valor do atraso. Ademais, a variável j (linha 27) representa o tamanho do objeto requisitado (para leitura ou escrita). As variáveis w e z caracterizam a probabilidade de uma operação ser de leitura (linha 25) e sequencial (linha 24), respectivamente.

Script 3.1: ARQUITETURA E CARGA DE TRABALHO NO *DiskSim*

Entrada: *storages*, *interfaces*, controladores, *driver*, carga de trabalho (x, y, x, w, i, j)
Saída: vazão (IOPS), tempo médio de resposta (s)

```
1  início
2      topology disksim_iodriver driver0 [
3          disksim_bus bus0 [
4              disksim_ctrl ctlr0 [
5                  disksim_bus bus1 [
6                      disksim_disk disk0 []
7                  ]
8                  disksim_bus bus2 [
9                      disksim_disk disk1 []
10                 ]
11                 disksim_bus bus3 [
12                     disksim_disk disk2 []
13                 ]
14             ]
15         ]
16     ]
17     disksim_pf Proc {
18         Number of processors = x
19     }
20     disksim_synthio Synthio {
21         Number of IO requests to generate = y,
22         Generators = [
23         disksim_synthgen {
24             Probability of sequential access = z,
25             Probability of read access = w,
26             General interarrival times = [ exponential, 0.0, i ],
27             Size = [j]
28         }
29         ]
30     }
31 fim
```

3.4 CONSIDERAÇÕES FINAIS

Esse capítulo apresentou a metodologia proposta para a modelagem e avaliação de sistemas de armazenamento híbridos. Inicialmente, discorreu a respeito dos processos necessários para a execução da proposta deste trabalho. A técnica adotada para os planejamentos de experimentos também foi explanada, assim como a metodologia e técnicas estatísticas para a avaliação de desempenho, dependabilidade e custo. Em seguida, introduziu os requisitos considerados da plataforma de computação em nuvem assumida. Posteriormente, apresentou o simulador utilizado para a validação dos modelos SPN e experimentações realizadas neste trabalho.

4

MODELAGEM DE DESEMPENHO E DISPONIBILIDADE

Este capítulo apresenta os modelos de desempenho e disponibilidade concebidos para representar o armazenamento com dispositivos híbridos. Os modelos também permitem a representação de componentes redundantes, o que é uma abordagem comumente adotada por centro de dados (*data centers*) e ambientes de computação em nuvem.

Inicialmente, a Seção 4.1 explana algumas considerações, como por exemplo, a arquitetura da plataforma na qual os modelos foram baseados. Além disso, apresenta as métricas adotadas e ferramentas utilizadas. Posteriormente, é introduzida a sintaxe adotada para definição das métricas das SPNs. Em seguida, as Seções 4.2 e 4.3 apresentam os modelos de desempenho e disponibilidade, respectivamente. Ademais, as referidas seções detalham os componentes dos modelos, bem como as equações para obtenção das métricas assumidas.

4.1 CONSIDERAÇÕES INICIAIS

Os modelos propostos são baseados na arquitetura do módulo de armazenamento (*back-end*) da plataforma *OpenStack Swift*, a qual é amplamente adotada por provedores de serviços devido a alta disponibilidade oferecida. Como dito no Capítulo 3, os valores de disponibilidade oferecidos por essa plataforma são decorrentes da réplica das informações (objetos) em 3 nós de armazenamento.

Para este trabalho, os modelos são apresentados levando em consideração esses três nós de armazenamento (primário e reservas). Entretanto, esta não é uma limitação dos modelos, os quais são capazes de representar sistemas de armazenamento híbridos em outras plataformas de computação em nuvem, com uma quantidade diferente de nós; limitados apenas pela dificuldade de avaliação devido à explosão do tamanho do espaço de estados (VALMARI, 1998).

Redes de *Petri* estocásticas (SPN) é adotada para representar operações de escrita, leitura, e uma mistura de ambas as operações (*mixed*). Os modelos desenvolvidos são utilizados para avaliar o desempenho de dispositivos híbridos sob diferentes tipos de requisição, consi-

derando o tempo médio de resposta (*response time*) e a vazão (*throughput*) como as métricas de interesse. A vazão neste trabalho representa a quantidade de entradas e saídas por segundo (*Input/Output per Second* - IOPS) (MEISTER; BRINKMANN, 2010).

Diagrama de blocos de confiabilidade (RBDs) é adotado neste trabalho para estimar a disponibilidade (A) e o tempo de inatividade (*downtime* - D) de sistemas de armazenamento com *drives* híbridos. A disponibilidade do sistema é calculada através da Equação 2.4, a qual seus elementos foram explanados na Seção 2.5. Além disso, a métrica *downtime* representa a indisponibilidade do sistema por um determinado período (T), e é obtida de acordo com a Equação 4.1.

$$D = (1 - A) \times T \tag{4.1}$$

É importante ressaltar que, uma única SPN poderia ser adotada para fins da modelagem de desempenho e disponibilidade; no entanto, a explosão do espaço de estados poderia dificultar a avaliação dos modelos. Os modelos neste trabalho foram avaliados através das ferramentas Mercury (SILVA et al., 2013) e TimeNET (ZIMMERMANN et al., 2006).

Adicionalmente, este trabalho considera as operações de escrita e leitura baseadas em uma distribuição exponencial, o que é uma abordagem similar adotada por outros trabalhos (GUO et al., 2014; SUN; PARKHOMOVSKY, 2013; KHAZAEI; MISIC; MISIC, 2012; VARKI et al., 2004). Entretanto, esta não é uma limitação dos modelos desenvolvidos. Por exemplo, poderia ser adotado para os modelos SPN outras distribuições de probabilidade utilizando distribuições *phase-type* (hipoexponencial, hiperexponencial ou *Erlang*), através da técnica de *moment matching* (WATSON; DESROCHERS, 1991). Estas distribuições são caracterizadas por serem formadas pela composição de fases exponenciais, possibilitando então sua utilização em modelos *markovianos*. Assim como MAO et al. (2012), este trabalho assume que as falhas e reparos de HDDs e SSDs são eventos independentes, que seguem uma distribuição exponencial.

Os seguintes operadores são adotados paras os modelos SPN: $P\{exp\}$ estima a probabilidade da expressão interna (exp); $E\{\#p\}$ representa o valor médio da expressão interna, a qual $\#p$ denota o número de marcas no lugar p; e $W(T)$ representa a taxa de disparo associada a transição T.

Para uma melhor compreensão, os modelos são apresentados com o uso de *building blocks*, isto é, blocos básicos.

4.2 MODELOS DE DESEMPENHO

Esta seção apresenta os modelos baseados em redes de *Petri* estocástica (SPNs) implementados para a avaliação de desempenho de sistemas de armazenamento híbridos. Os modelos em questão levam em consideração as diferentes cargas de trabalho à que são submetidos os dispositivos de armazenamento. Portanto, foram desenvolvidos modelos que contemplam operações de escrita, leitura e *mixed* (escrita e leitura).

4.2.1 Modelo SPN para Operações de Escrita

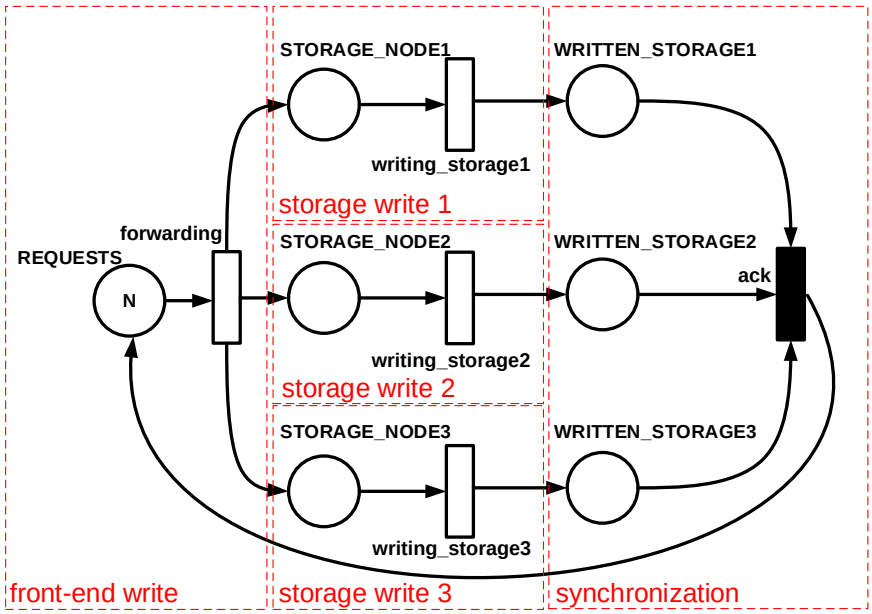


Figura 4.1: Modelagem de requisições de escrita

A Figure 4.1 demonstra o modelo SPN para a representação específica de operações de escrita em nós de armazenamento, o qual é baseado na arquitetura da plataforma *OpenStack Swift* (isto é, 3 nós de armazenamento). Porém, este modelo também permite a análise de desempenho quando adotados dispositivos de armazenamento híbrido em outras plataformas. O impacto ocasionado pela variação da taxa de chegada de requisições, bem como da quantidade de usuários a submetê-las, também são contemplados pelo modelo proposto. Adicionalmente, diferentes tempos para a execução das operações de escrita também podem ser abordados, os quais são dependentes do tipo de tecnologia de armazenamento adotado e do tamanho do objeto que compõe a requisição.

O modelo é composto de 7 lugares: $REQUESTS$, $STORAGE\_NODE_i$ e $WRITTEN\_STORAGE_i$ (um para cada nó de armazenamento). Além disso, 4 transições exponenciais são consideradas ($forwarding$ e $writing\_storage_i$) e uma transição imediata (ack). Os atributos das transições são discriminados na Tabela 4.1. Um bloco *storage write* é adotado para cada nó do sistema, enquanto que os blocos *front-end write* e *synchronization* são únicos, representando o início e finalização do processo de armazenamento de objetos, respectivamente.

O bloco *front-end write* é responsável por representar a chegada de requisições no servidor *proxy*. A marcação do lugar $REQUESTS$ (N) denota o estado inicial, o qual representa a quantidade de requisições concorrentes permitidas. Esta marcação (N) também pode ser interpretada como a quantidade de usuários efetivamente realizando requisições. A transição $forwarding$ indica o encaminhamento das requisições para os nós de armazenamento, onde o respectivo objeto é preparado para ser armazenado (representado por marcas no lugar

Tabela 4.1: Atributos das transições - modelo de escrita

Transição	Tipo	Semântica de Servidor	Peso	Prioridade
forwarding	Temporizada	Servidor Infinito	-	-
writing_storage1	Temporizada	Servidor Único	-	-
writing_storage2	Temporizada	Servidor Único	-	-
writing_storage3	Temporizada	Servidor Único	-	-
ack	Imediata	-	1	1

$STORAGE\_NODE_i$). Ainda com relação à transição $forwarding$, é adotado para a mesma a semântica de servidor infinito (*infinite server*) (BALBO, 2001) com o objetivo de representar chegadas concorrentes.

Já a transição $writing\_storage_i$ (bloco *storage write*) utiliza a semântica de servidor único (*single server*) (BALBO, 2001), pois é assumido que apenas uma operação por vez pode ser realizada no dispositivo de armazenamento. Dessa forma, à medida que novas requisições chegam ao lugar $STORAGE\_NODE_i$, são processadas apenas se o recurso estiver livre, caso contrário um enfileiramento de requisições é formado. Este processo é representado neste modelo pelo acúmulo de marcas no lugar $STORAGE\_NODE_i$. Estas marcas são absorvidas individualmente quando há o disparo da transição $writing\_storage_i$, o que ocorre de acordo com o tempo associado à mesma. Vale ressaltar que este tempo varia de acordo com a tecnologia de armazenamento utilizada e do tamanho do objeto que compõe a requisição.

A finalização do processo de escrita nos nós de armazenamento é representada pelo bloco *synchronization*. A geração de marcas no lugar $WRITTEN\_NODE_i$ denota que o objeto foi armazenado. No entanto, como o modelo segue a arquitetura da plataforma *OpenStack Swift*, se faz necessária a confirmação do encerramento do processo de armazenamento das 3 réplicas. A transição imediata *ack* é responsável por executar essa função. Portanto, a mesma está habilitada para disparo apenas com a existência de ao menos uma marca em cada lugar $WRITTEN\_NODE_i$. É importante ressaltar que este trabalho não considera o tempo de retorno da informação de confirmação da escrita, motivo esse da utilização de uma transição sem atraso, como demonstrado na Seção 2.6.

O tempo médio de resposta é estimado através da lei de *Little* (CASSANDRAS; LAFORTUNE, 2009; TRIVEDI, 2008), segundo a Equação 2.1, na qual R representa o tempo de resposta médio para execução da operação de escrita, L é o número médio de requisições dos usuários e λ representa a taxa de chegada das requisições no servidor *proxy*. Especificamente para este modelo, o número médio e a taxa de chegada de requisições são calculados através das Equações 4.2 e 4.3, respectivamente.

$$L = N - E\{\#REQUESTS\} \tag{4.2}$$

$$\lambda = E\{\#REQUESTS\} \times W(forwarding) \tag{4.3}$$

Com relação à vazão (isto é, IOPS), foi inserido um par *dummy* (*dummy pair*) após a transição *ack* (Figure 4.2) para facilitar o cálculo da mesma (KAMATH; VISWANADHAM, 1986; LEE; PARK, 1993). Para este trabalho, o tempo adotado para a transição *dummy* é muito pequeno ($\varphi = 0.00001s$), ou seja, não possui influência significativa no resultado da estimação da métrica. Dessa forma, a vazão (V) pode ser calculada de acordo com a Equação 4.4.

$$V = E\{\#PLACE\_DUMMY\_X\} \times 1/\varphi \tag{4.4}$$

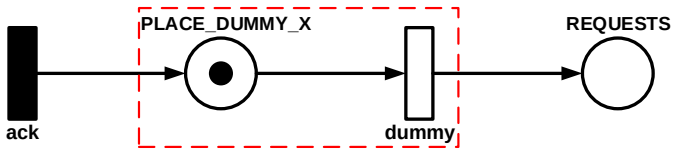


Figura 4.2: Par *dummy*

As equações demonstradas possibilitam a avaliação de desempenho de diferentes tecnologias de armazenamento híbrido quando submetidas a cargas de trabalho compostas apenas por operações de escrita. Diferentes combinações de dispositivos de armazenamento podem então ser analisadas através do modelo proposto, a partir da variação das taxas de escrita atribuídas às transições. Entretanto, esta abordagem é limitada pela necessidade de um embasamento a respeito do formalismo adotado para utilização do modelo analítico concebido. As métricas adotadas para esta abordagem podem ser visualizadas na Tabela 4.2.

Tabela 4.2: Métricas para o modelo SPN - escrita

Equação	Métrica	Sintaxe
2.1	Tempo médio de resposta	$R = (N - E\{\#REQUESTS\})/(E\{\#REQUESTS\} \times W(forwarding))$
3.4	Vazão	$V = E\{\#PLACE\_DUMMY\_X\} \times 1/\varphi$

4.2.2 Modelo SPN para Operações de Leitura

A Figura 4.3 demonstra o modelo baseado em redes de *Petri* estocástica para a representação de operações de leitura. Similar ao modelo apresentado na Subseção 4.2.1, a arquitetura da plataforma *OpenStack Swift* é levada em consideração, apesar de não ser uma limitação do modelo. Portanto, a quantidade de requisições, bem como a taxa de chegada das mesmas também são representadas neste modelo. Como já dito anteriormente, o armazenamento de objetos é feito em três nós, dessa forma, requisições de leitura são realizadas no nó de armazenamento que estiver disponível, dentre os 3 existentes, caso contrário, um enfileiramento é formado. Usualmente, a melhoria de desempenho pela adoção de SSDs em detrimento a HDDs é observado principalmente quando da execução de operações de leitura. Dessa forma, este modelo possibilita a avaliação de desempenho para diferentes arquiteturas de dispositivos de armazenamento

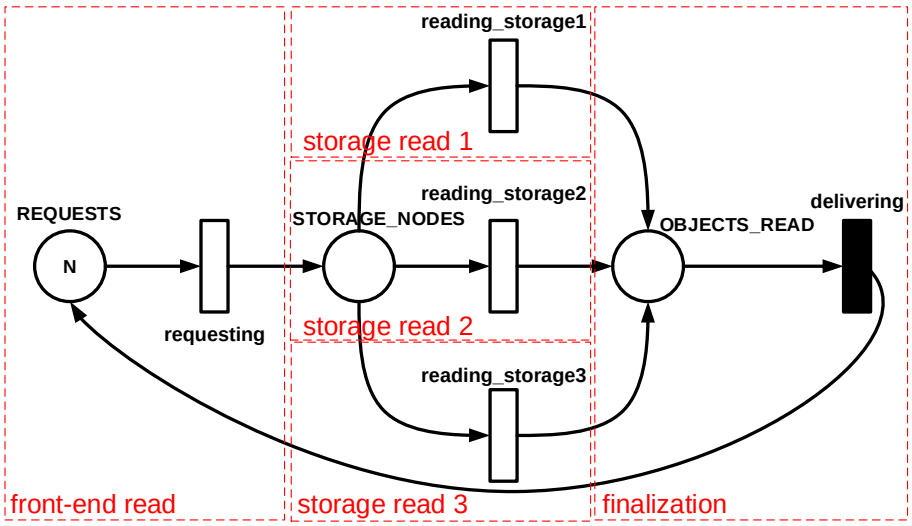


Figura 4.3: Modelagem de requisições de leitura

híbridos, quando submetidos a cargas de trabalho exclusivamente constituídas por operações de leitura.

Este modelo é composto por três lugares (*REQUESTS*, *STORAGE_NODES* e *OBJECTS_READ*), quatro transições exponenciais (*requesting*, $reading\_storage_i$), e uma transição imediata (*delivering*). Os atributos das transições são explanados na Tabela 4.3. Cada nó de armazenamento do sistema é representado por um bloco *storage read*. Nós assumimos que cada nó contém uma cópia dos objetos requisitados. Os blocos *front-end* e *finalization* representam a chegada das solicitações por objetos e a finalização do processo de leitura, respectivamente.

Tabela 4.3: Atributos das transições - modelo de leitura

Transição	Tipo	Semântica	Peso	Prioridade
requesting	Temporizada	Servidor Infinito	-	-
reading_storage1	Temporizada	Servidor Único	-	-
reading_storage2	Temporizada	Servidor Único	-	-
reading_storage3	Temporizada	Servidor Único	-	-
delivering	Imediata	-	1	1

Similar ao modelo SPN para operações de escrita, o bloco *front-end read* representa o servidor *proxy*, o qual é responsável por lidar com a chegada de requisições. A marcação *N*, ilustrada dentro do lugar *REQUESTS* denota o estado inicial do modelo, indica a quantidade de requisições de leitura concorrentes permitidas no sistema. A transição *requesting* adota a semântica de servidor infinito. Portanto, à medida que a respectiva transição dispara, condicionada à existência de marcas no lugar *REQUESTS*, está sendo representado o redirecionamento de requisições para os nós de armazenamento. Esta operação é finalizada com a geração de marcas no lugar *STORAGE_NODES*.

A operação de leitura é representada pelas transições $reading\_storage_i$, contidas nos

blocos *storage read*. Qualquer nó (mas apenas um) pode ser selecionado para a execução da operação, já que como dito anteriormente, consideramos que todos possuem os mesmos dados. Dessa forma, quando há a existência de marcas no lugar $STORAGE\_NODES$, uma das transições que representam o processo de leitura irá disparar. Considerando que os nós de armazenamento são capazes de processar apenas uma operação de leitura por vez, todas as transições $reading\_storage_i$ adotam a semântica de servidor único (*single server*). Em outras palavras, apenas três requisições (o número de nós) podem ser executadas em paralelo. O tempo de atraso entre requisições, associado às transições que representam a operação de leitura, segue uma distribuição exponencial, de acordo com o tamanho do objeto requisitado e da tecnologia utilizada. Após o disparo da transição, a marca referente ao objeto requerido é absorvida do lugar $STORAGE\_NODES$, e em seguida, de acordo com o atraso atribuído à transição, uma marca é gerada no lugar $OBJECTS\_READ$.

O bloco *finalization* representa o encerramento do processo de leitura. A geração de marcas no lugar $OBJECTS\_READ$ indica que uma das cópias do objeto requerido pelo usuário foi obtida de um dos nós de armazenamento. Portanto, esta marcação da rede demonstra que o dado em questão passa a estar disponível para ser enviado ao usuário. Similar ao modelo SPN de escrita, este trabalho não considera o tempo de envio do objeto lido pelo sistema ao *proxy*. Dessa forma, a finalização da requisição é representada pela absorção de marcas do lugar $OBJECTS\_READ$, quando há o disparo da transição imediata *delivering*, isto é, sem atraso algum atribuído.

Para este modelo, o tempo médio de resposta para operações de leitura é também estimado através do uso da lei de *Little*, de acordo com a Equação 2.1. Especificamente, o número médio de requisições de leitura na fila do sistema é representado pela Equação 4.5, enquanto que a taxa de chegada das requisições segue a Equação 4.6.

$$L = N - E\{\#REQUESTS\} \tag{4.5}$$

$$\lambda = E\{\#REQUESTS\} \times W(requesting) \tag{4.6}$$

Assim como no modelo SPN para operações de escrita, um par *dummy* (Figura 4.2) é adotado para facilitar estimar a vazão do sistema. Portanto, o mesmo é inserido após a transição $delivering$, com um tempo insignificante de $\varphi = 0.00001s$, de forma a não causar influência no resultado da métrica. Levando em conta as considerações explanadas, a vazão (V) pode ser calculada de acordo com a Equação 4.7.

$$V = E\{\#PLACE\_DUMMY\_DELIVERING\} \times 1/\varphi \tag{4.7}$$

A Tabela 4.4 reúne as métricas adotadas neste modelo para a avaliação de desempenho de sistemas de armazenamento híbrido quando submetidos a cargas de trabalho constituídas apenas por requisições de leitura. Logo, a análise de diferentes tecnologias de armazenamento

torna-se possível com a mudança dos valores atribuídos aos componentes do modelo proposto. Consideramos que, este formalismo passa a ser uma abordagem bastante útil, principalmente quando da adoção de SSDs, os quais possuem significantes resultados no que diz respeito à leitura de dados. No entanto, esta abordagem requere um embasamento a respeito do formalismo adotado para utilização do modelo analítico concebido.

Tabela 4.4: Métricas para o modelo SPN - leitura

Equação	Métrica	Sintaxe
2.1	Tempo médio de resposta	$R = (N - E\{\#REQUESTS\})/(E\{\#REQUESTS\} \times W(requesting))$
3.7	Vazão	$V = E\{\#PLACE\_DUMMY\_DELIVERING\} \times 1/\varphi$

4.2.3 Modelo SPN para Operações *Mixed*

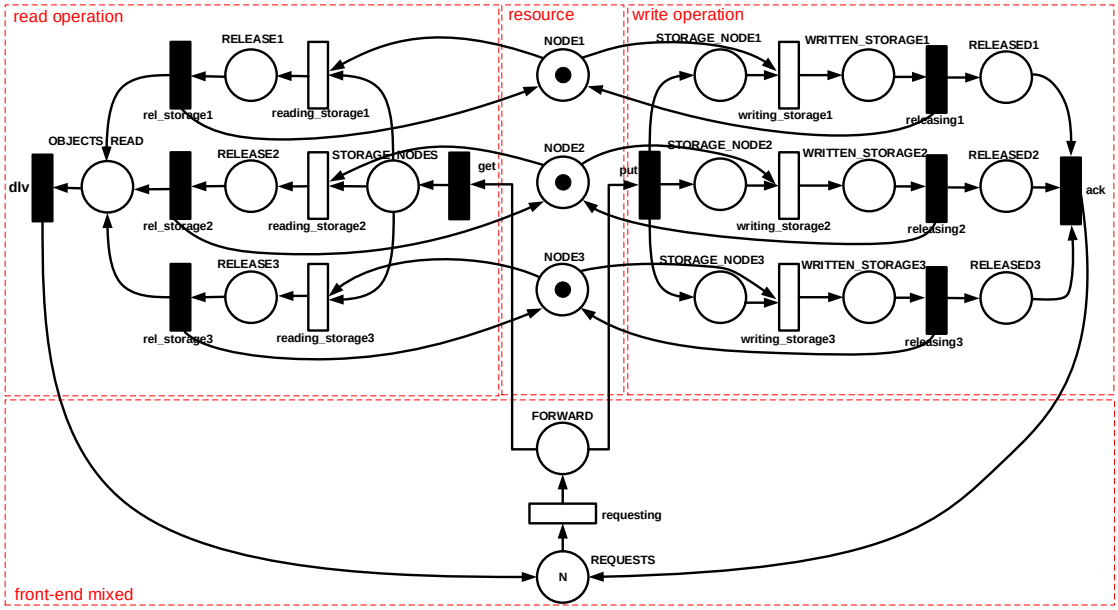


Figura 4.4: Modelagem de requisições *mixed*

A Figura 4.4 demonstra o modelo SPN concebido para representar requisições misturadas (*mixed*) em sistemas de armazenamento híbridos. A composição da carga de trabalho considerada para este modelo é formada por requisições de escrita e leitura. Da mesma forma que os modelos apresentados nas Subseções 4.2.1 e 4.2.2, este modelo leva em consideração a arquitetura de armazenamento da plataforma *OpenStack Swift*, bem como o requisito de três réplicas para cada objeto. No entanto, como dito anteriormente, esta não é uma limitação do modelo, podendo ser adotado para diferentes plataformas de computação em nuvem. Adicionalmente, para lidar com mais de um tipo de requisição assumimos que a escrita e leitura de objetos são efetuadas nos mesmos nós de armazenamento. Portanto, a disponibilidade do recurso (nó de armazenamento) para execução de uma determinada operação independe da característica (leitura ou escrita) da requisição anterior.

Para uma melhor explicação, este modelo é dividido em 4 blocos. Similar aos modelos de escrita e leitura, o bloco *front-end mixed* representa a chegada das solicitações de escrita e leitura. O bloco *resource* denota os recursos existentes no sistema modelado. A execução das operações de leitura e escrita é representada pelos blocos *read operation* e *write operation*, respectivamente. Os atributos das transições que compõem esses blocos são explanados na Tabela 4.5, enquanto que uma explicação mais detalhada de cada bloco é apresentada a seguir.

Tabela 4.5: Atributos das transições - modelo *mixed*

Transição	Tipo	Semântica	Peso	Prioridade
requesting	Temporizada	Servidor infinito	-	-
$writing\_storage_i$	Temporizada	Servidor único	-	-
$reading\_storage_i$	Temporizada	Servidor único	-	-
$releasing_i$	Imediata	-	1	1
$rel\_storage_i$	Imediata	-	1	1
ack	Imediata	-	1	1
dlv	Imediata	-	1	1
put	Imediata	-	0.5	1
get	Imediata	-	0.5	1

Mais especificamente, o bloco *front-end mixed* representa a chegada de requisições no servidor *proxy*, no qual a marcação (N), no lugar $REQUESTS$, indica o número de requisições de leitura e escrita permitidas. A diferenciação do tipo de operação a ser efetuada nos nós de armazenamento é realizada pelas transições imediatas *put* e *get*, as quais representam as operações de escrita e leitura, respectivamente. Para este trabalho, consideramos que requisições de leitura ou escrita possuem a mesma probabilidade (0.5 cada), as quais são modeladas como o peso nas transições imediatas (AJMONE MARSAN; CONTE; BALBO, 1984b).

O bloco *resource* denota a disponibilidade de cada nó de armazenamento para executar operações de escrita ou leitura. Como dito anteriormente, este trabalho assume que apenas uma operação por vez é realizada em cada nó. Além disso, a marcação atribuída ao lugar $NODE_i$ permite que somente um tipo de operação seja executado. Por exemplo, uma operação de leitura apenas pode ser realizada após a obtenção de um recurso de um nó ($NODE_i$). A disponibilidade do recurso é representada pela existência de uma marca no local $NODE_i$, a qual é absorvida quando do disparo de uma transição de leitura ($reading\_storage_i$) ou escrita ($write\_storage_i$). A indisponibilidade de recurso (inexistência de marcas no local $NODE_i$) está associada ao tempo e condições de execução das operações, nos blocos de leitura e escrita, os quais se assemelham aos modelos apresentados anteriormente.

Mais especificamente, o disparo das transições de leitura ou escrita é reproduzido nos blocos *read operation* e *write operation*, respectivamente. Ambas as operações demandam a existência de uma marca no lugar $NODE_i$, e em um dos seguintes lugares: $STORAGE\_NODES$ ou $STORAGE\_NODE_i$. Para qualquer outra operação, no mesmo nó, não será permitida sua execução até a liberação do recurso. A liberação é representada pelo disparo das transições

imediatas $rel\_storage_i$ ou $releasing_i$, após ter sido efetuada a operação de leitura ou escrita, respectivamente. Dessa forma, a marca que representa a disponibilidade do recurso é gerada no lugar $NODE_i$, sem atraso. Além da liberação do recurso, o disparo das transições imediatas citadas geram marcas nos lugares $OBJECTS\_READ$ ou $RELEASED_i$.

Similar ao modelo SPN para leitura, quando da presença de ao menos uma marca, o lugar $OBJECTS\_READ$ denota que uma cópia do objeto requerido foi obtida de um dos nós de armazenamento, em seguida, a finalização da requisição é representada pelo disparo da transição dlv. Já a presença de marcas no lugar $RELEASED_i$ representa o armazenamento do objeto. Idêntico ao modelo SPN para escrita, se faz necessária a confirmação do processo de armazenagem das 3 réplicas (de acordo com o *OpenStack Swift*), a qual é efetuada pela habilitação e disparo da transição imediata *ack*.

O tempo médio de resposta para este modelo é também estimado através da lei de *Little*, da mesma forma que nos modelos anteriores, de acordo com a Equação 2.1. Logo, o número médio e taxas de requisições são calculados através das Equações 4.8 e 4.9.

$$L = N - E\{\#REQUESTS\} \tag{4.8}$$

$$\lambda = E\{\#REQUESTS\} \times W(requesting) \tag{4.9}$$

Para estimar a vazão, dois pares *dummy* foram inseridos após as transições *ack* ($PLACE\_DUMMY\_ACK$) e dlv ($PLACE\_DUMMY\_DLV$). Similar aos modelos anteriores, ambas as transições *dummy* possuem um tempo associado muito pequeno ($\varphi = 0.00001s$). Portanto, a vazão (V) para este modelo em questão é calculada de acordo com a Equação 4.10.

$$V = (E\{\#PLACE\_DUMMY\_DLV\} \times 1/\varphi) + (E\{\#PLACE\_DUMMY\_ACK\} \times 1/\varphi) \tag{4.10}$$

Usualmente, requisições reais possuem ambas as operações de escrita e leitura, mesmo que em porcentagens muito diferentes. Portanto, é importante ressaltar que, a capacidade deste modelo de abranger os dois tipos de operação tornam o mesmo o mais adequado para a avaliação de desempenho de cargas de trabalho reais. Porém, para utilização deste modelo se faz necessário um embasamento a respeito do formalismo adotado para a concepção do mesmo. As métricas adotadas neste trabalho para a avaliação de desempenho de sistemas de armazenamento híbridos quando submetidos a cargas de trabalho *mixed* podem ser visualizadas na Tabela 4.6.

Tabela 4.6: Métricas para o modelo SPN - *mixed*

Equação	Métrica	Sintaxe
2.1	Tempo médio de resposta	$R = (N - E\{\#REQUESTS\})/(E\{\#REQUESTS\} \times W(requesting))$
3.10	Vazão	$V = (E\{\#PLACE\_DUMMY\_DLV\} \times 1/\varphi) + (E\{\#PLACE\_DUMMY\_ACK\} \times 1/\varphi)$

4.3 MODELO RBD PARA NÓS DE ARMAZENAMENTO REDUNDANTE

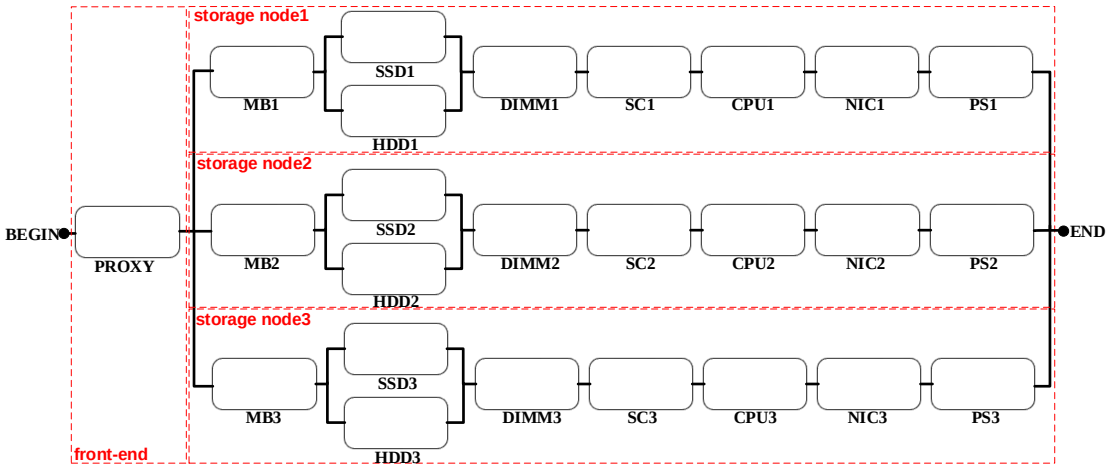


Figura 4.5: Modelo RBD para um armazenamento híbrido redundante

Esta seção apresenta o modelo RBD concebido para avaliação da disponibilidade de sistemas de armazenamento híbridos. Como apresentado anteriormente, nas Seções 2.5 e 2.7, diagrama de blocos de confiabilidade é um modelo combinatorial que relaciona as relações estruturais entre os elementos de um sistema. Além disso, partem da premissa de que a falha ou recuperação de um componente não é afetada pelo comportamento de outro elemento. Para este trabalho, RBD é adotado para estimar a disponibilidade e o *downtime* do sistema em questão. Similar aos formalismos de espaço de estado apresentados previamente, o modelo em questão leva em consideração a arquitetura da plataforma *OpenStack Swift* (isto é, 3 nós de armazenamento). No entanto, esta não é uma limitação do modelo, o qual pode representar dispositivos de armazenamento híbridos em outras plataformas. Portanto, o modo operacional do modelo requer que o servidor *proxy* e, ao menos um nó de armazenamento (com um HDD ou SSD), estejam funcionais.

Para fins explicativos, consideramos o modelo RBD concebido composto por quatro blocos básicos. O bloco *front-end* é constituído apenas pelo servidor *proxy*, o qual é responsável pelo recebimento e distribuição das requisições, como explanado em seções anteriores. Já os nós de armazenamento, os quais contêm os dispositivos híbridos arranjados em paralelo, são representados nos blocos *storage node*.

A Figura 4.5 demonstra o modelo RBD para 3 nós de armazenamento e um servidor *proxy*. O servidor *proxy* é representado por um único bloco (*PROXY*), enquanto que os nós de armazenamento são detalhados, pois os mesmos são o foco da abordagem de modelagem proposta. Particularmente, o servidor *proxy* não contém qualquer componente cuja representação individual faça parte do objetivo deste trabalho. Portanto, a composição de todos os *hardwares*, representando uma única máquina, é considerada para o servidor *proxy*. Ainda com relação ao bloco *PROXY*, os respectivos MTTFs e MTTRs são especificados, e uma técnica de redução

baseada no arranjo em série é utilizada no modelo para a obtenção de um único bloco (KUO; ZUO, 2003). A Seção 5.3 provê maiores detalhes sobre os valores de MTTF e MTTR adotados.

Assim como na plataforma *OpenStack Swift*, os nós de armazenamento estão dispostos em paralelo entre si. Esta composição ainda é arranjada em série com o *PROXY*. Cada bloco *storage node* contém um dispositivo de disco rígido (*HDD*) e de estado sólido (*SSD*), os quais também são considerados em paralelo, ao menos um dispositivo de armazenamento deve estar funcional. O restante dos componentes que compõem os nós de armazenamento estão arranjados em série, e cada componente deve estar funcionando apropriadamente, para que o nó como um todo possa ser considerado disponível. A Tabela 4.7 demonstra os componentes do RBD e sua descrição.

Tabela 4.7: Descrição dos componentes do modelo RBD

Componente	Descrição
PROXY	Servidor *proxy*
SSD	Dispositivo de estado sólido
HDD	Dispositivo de disco rígido
MB	Placa mãe
DIMM	Memória RAM
SC	Controlador de armazenamento
CPU	Unidade central de processamento
NIC	Controlador da interface de rede
PS	Fonte de energia

Como já mencionado, este modelo contém componentes relacionados através de arranjos em série e paralelo. Logo, para estimar as métricas adotadas neste trabalho (disponibilidade e *downtime*) se faz necessária a análise dos subsistemas em questão, individualmente (EBELING, 2004). Posteriormente, os valores obtidos devem ser dispostos para o cálculo da métrica de acordo com a relação entre os subsistemas. Para este trabalho, a disponibilidade de todo o sistema (A_s), para 3 nós de armazenamento, é estimada através da Equação 4.11.

$$A_s = A_{proxy} \times (1 - \prod_{j=1}^{3} (1 - A_{sn_j})) \tag{4.11}$$

Especificamente, A_{proxy} indica a disponibilidade do servidor *proxy*. A_{sn_j} representa a disponibilidade de um nó de armazenamento j (que também é composto por arranjos em série e paralelo), e pode ser obtida como segue

$$A_{sn_j} = A_{MB_j} \times A_{Hybrid_j} \times A_{DIMM_j} \times A_{SC_j} \times A_{CPU_j} \times A_{NIC_j} \times A_{PS_j} \tag{4.12}$$

onde, A_{Hybrid_j} denota a disponibilidade do dispositivo de armazenamento híbrido (SSD_j e HDD_j) que pode ser calculada de acordo com a Equação 4.13. Os demais elementos em série, representam a disponibilidade dos componentes do nó de armazenamento, os quais foram explanados na Tabela 4.7.

$$A_{Hybrid_j} = (1 - (1 - A_{SSD_j}) \times (1 - A_{HDD_j})) \tag{4.13}$$

Por fim, o tempo de inatividade do sistema (*downtime*) pode ser obtido através da Equação 4.1, considerando a disponibilidade total (A_s) e um período (T) específico a ser analisado. Dessa forma, considerando as equações demonstradas, o RBD concebido permite a avaliação de sistemas de armazenamento híbridos, no que diz respeito à disponibilidade e *downtime*. Portanto, possibilita a análise dos benefícios ou malefícios provocados ao sistema representado, levando em consideração apenas o tempo médio entre falhas (MTTF), e reparos (MTTR), dos dispositivos de armazenamento que porventura venham a ser avaliados. Entretanto, para adoção desta abordagem se faz necessário um embasamento a respeito do formalismo baseado para a concepção deste modelo analítico.

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os modelos SPN e RBD concebidos, os quais são as principais contribuições deste trabalho. Primeiramente, explanou a arquitetura da plataforma de computação em nuvem (*OpenStack Swift*) na qual os modelos apresentados foram baseados. Basicamente foi especificada sua estrutura de armazenamento, como por exemplo, a redundância de objetos que essa plataforma assume. Além disso, apresentou os conceitos de desempenho e dependabilidade que são considerados neste trabalho. Em seguida, apresentou os modelos baseados em redes de *Petri* estocásticas para operações de escrita, leitura e *mixed*. A descrição dos elementos das SPNs foi feita, bem como das métricas a serem estimadas por tais modelos. Similarmente, especificou os componentes que constituem os blocos básicos, os quais compõem o modelo combinatorial para sistemas de armazenamento híbridos. O significado dos componentes foi explicado, bem como as equações para o cálculo de disponibilidade dos arranjos em série e paralelo.

5

RESULTADOS EXPERIMENTAIS

Este capítulo apresenta os resultados experimentais para demonstrar a viabilidade prática da abordagem de modelagem proposta. Inicialmente, a Seção 5.1 apresenta a configuração do experimento, onde são definidas as cargas de trabalho adotadas, a ferramenta utilizada para validação e experimentação dos modelos, além dos dispositivos de armazenamento adotados. Em seguida, a Seção 5.2 denota os experimentos efetuados para validação dos modelos concebidos, além da avaliação de desempenho de dispositivos de armazenamento híbridos, sob diferentes tipos de operação, arquitetura, e políticas de armazenamento de dados. Posteriormente, a Seção 5.3 exibe os experimentos e resultados de disponibilidade a respeito do sistema de armazenamento híbrido adotado neste trabalho. Finalmente, na Seção 5.4, são avaliadas concomitantemente, as métricas de desempenho, dependabilidade, e o custo, a fim de demonstrar as configurações mais adequadas para os sistemas quando da adoção de dispositivos de armazenamento híbridos.

5.1 CONFIGURAÇÃO DOS EXPERIMENTOS

Esta seção apresenta a configuração adotada para validação e experimentação dos modelos SPN concebidos com o propósito de prover a avaliação de desempenho de sistemas de armazenamento híbridos.

Assim como no Capítulo 4, os experimentos também são baseados na plataforma *OpenStack Swift*, na qual dados estruturados e não-estruturados são armazenados na forma de objetos (KAPADIA; VARMA; RAJANA, 2014; VAIDYA; PUNDKAR, 2015; JIN; LEE, 2015). Como explicado anteriormente, esta plataforma de computação em nuvem usualmente mantém três cópias do mesmo objeto, em diferentes nós de armazenamento (KAPADIA; VARMA; RAJANA, 2014; BISWAS; PATWA; SANDHU, 2015), os quais são submetidos de forma concorrente por diferentes clientes.

Este trabalho assume a característica de cargas de trabalho reais, nas quais mais de 80% das requisições de escrita e leitura são aleatórias (isto é, dados não-relacionados) (CARNS et al., 2011). Portanto, nós adotamos 80% como a percentagem de requisições aleatórias para as si-

Tabela 5.1: Modelos dos dispositivos de armazenamento e respectivas capacidades

Tecnologia	Modelo	Capacidade
HDD	*Hewlett Packard C2247A*	250GB
HDD	*Seagate Barracuda ST32171W*	500GB
HDD	*Seagate Cheetah 4LP ST34501N*	1TB
HDD	*IBM DNES 390170W*	2TB
SSD	*Samsung K9XXG08UXM*	250GB, 500GB, 1TB, 2TB

mulações realizadas pelo *DiskSim*, enquanto que as 20% restantes são sequenciais. Além disso, quatro diferentes tipos de HDD (Tabela 5.1) e 1 SSD (por limitação da ferramenta), com diferentes capacidades (250GB, 500GB, 1TB e 2TB) foram utilizados no *DiskSim*. É importante ressaltar que, os HDDs e os SSDs adotados são modelos definidos como validados por BUCY et al. (2008) e AGRAWAL et al. (2008), respectivamente, isto é, resultados de simulações condizem com o dispositivo de fato. Similar a outros trabalhos (CARNS et al., 2011; ZHENG et al., 2013), cada simulação no *DiskSim* considera a submissão de 10.000 requisições, as quais são executadas por 5 clientes, com um tempo médio de atraso de 1ms, seguindo uma distribuição exponencial. Ademais, consideramos objetos com cinco tamanhos diferentes (256KB, 512KB, 1MB, 2MB e 4MB) e três tipos de operações: escrita (*write-only*), leitura (read-only), e *mixed* (50% escrita e 50% leitura). A Tabela 5.2 reúne os valores assumidos para os parâmetros mencionados.

Tabela 5.2: Parâmetros e valores para simulação

Parâmetro	Valores
Requisições randômicas	80%
Requisições sequenciais	20%
Número de requisições	10.000
Quantidade de clientes	5
Capacidade de armazenamento	250GB, 500GB, 1TB, 2TB
Tamanho do objeto	256KB, 512KB, 1MB, 2MB, 4MB
Tipo de operação	*write-only, read-only, mixed*
Tecnologia	HDD, SSD, Híbrido (*SSDRANDOM* e *SSDasCACHE*)

5.2 EXPERIMENTOS DE DESEMPENHO

Esta seção apresenta resultados obtidos dos modelos de desempenho. Inicialmente, uma validação dos modelos SPN é apresentada, e, em seguida, resultados experimentais são detalhados.

5.2.1 Validação dos Modelos de Desempenho

Para validar os modelos formais de espaço de estado concebidos, realizamos simulações no *DiskSim* e comparamos com os resultados gerados pelos modelos SPN, usando uma análise estacionária. A Tabela 5.3 denota os tempos das transições obtidos para operações de escrita e leitura para cada componente de armazenamento e, assumimos que os mesmos seguem uma distribuição exponencial. Particularmente, discos rígidos (HDDs) são assumidos para esta validação. Quatro diferentes tipos de HDD, com capacidades diferentes (250GB, 500GB, 1TB e 2TB) foram considerados no *DiskSim*. De forma similar à carga de trabalho sintética adotada no *Disksim*, um atraso de 1ms entre requisições é considerado nos modelos SPN para as transições *request* e $forwarding$. Além disso, cinco marcas são adotadas como marcação inicial no lugar $REQUESTS$ para denotar o número de requisições concorrentes. A Tabela 5.4 demonstra os resultados obtidos do simulador e dos modelos de escrita, leitura, e *mixed*, respectivamente.

Tabela 5.3: Tempos das transições para *hard disk drives* (HDDs)

Capacidade	Tamanho do Objeto	Escrita (s)	Leitura (s)
250 GB	256 KB	0.018976496	0.01990558
250 GB	512 KB	0.019309865	0.02217471
250 GB	1 MB	0.021278833	0.026644063
250 GB	2 MB	0.030662351	0.035454993
250 GB	4 MB	0.054161631	0.053137919
500 GB	256 KB	0.013369235	0.01603979
500 GB	512 KB	0.013897936	0.01800167
500 GB	1 MB	0.018443708	0.021996261
500 GB	2 MB	0.032511969	0.029815448
500 GB	4 MB	0.060809326	0.045450513
1 TB	256 KB	0.011006907	0.013493884
1 TB	512 KB	0.011834344	0.015501674
1 TB	1 MB	0.017183868	0.019416786
1 TB	2 MB	0.031300076	0.027273924
1 TB	4 MB	0.059583922	0.043025287
2 TB	256 KB	0.011549144	0.013674571
2 TB	512 KB	0.012401185	0.015594799
2 TB	1 MB	0.018368684	0.01960359
2 TB	2 MB	0.032442889	0.027405343
2 TB	4 MB	0.060789634	0.043106382

Testes t emparelhado (MONTGOMERY; RUNGER, 2014) foram conduzidos para os modelos de leitura, escrita e *mixed*, considerando IOPS^{-1} e o tempo médio de resposta (RT) como as métricas a serem avaliadas. Assumindo um nível de significância de $\alpha = 0.05$, os intervalos obtidos são IOPS^{-1}:(-0,00015;0,00502), RT:(-0,00029;0,02659) para operações de escrita, IOPS^{-1}:(-0,000901;0,000026), RT: (-0,00011;0,00502) para operações de leitura, e IOPS^{-1}:(-0,001455;0,001107), RT:(-0,00165;0,00834) para operações *mixed*. Como os intervalos contêm 0, a hipótese nula é verdadeira, o que indica que as diferenças entre as simulações

Tabela 5.4: Resultados da validação

Capacidade	**Tamanho do Objeto**	**Escrita**				**Leitura**				***Mixed* (50% E / 50% L)**			
		IOPS^{-1}		**Tempo Médio de Resposta (s)**		**IOPS^{-1}**		**Tempo Médio de Resposta (s)**		**IOPS^{-1}**		**Tempo Médio de Resposta (s)**	
		DiskSim	**SPN**	***DiskSim***	**SPN**	***DiskSim***	**SPN**	***DiskSim***	**SPN**	***DiskSim***	**SPN**	***DiskSim***	**SPN**
250 GB	256 KB	0.0189843	0.0222882	0.093907	0.1063190	0.00682823	0.0066571	0.0330421	0.0273444	0.0127291	0.0113429	0.062589	0.0535380
250 GB	512 KB	0.0199265	0.0227247	0.098618	0.1085030	0.00758418	0.0073977	0.0368149	0.0310445	0.0132166	0.0115947	0.065035	0.0547930
250 GB	1 MB	0.0283555	0.025192	0.140755	0.1208430	0.00903777	0.0089991	0.0441026	0.0390487	0.0150445	0.0128946	0.074191	0.0612900
250 GB	2 MB	0.0472591	0.0362025	0.23525	0.1759060	0.01201108	0.0118948	0.0589898	0.0535251	0.0211288	0.01846	0.104601	0.0891190
250 GB	4 MB	0.0654975	0.0643413	0.326425	0.3166120	0.01786987	0.0185085	0.0882531	0.0865928	0.0357627	0.032663	0.177753	0.1601390
500 GB	256 KB	0.0134235	0.01567	0.066106	0.0732130	0.00544179	0.0053673	0.0260737	0.0209057	0.0094209	0.0080195	0.046039	0.0369180
500 GB	512 KB	0.0151241	0.0158843	0.074609	0.0742850	0.00600923	0.0060512	0.0289223	0.0243186	0.0098129	0.008384	0.048013	0.0387380
500 GB	1 MB	0.0242827	0.01932	0.12039	0.0914720	0.00711871	0.0073977	0.034486	0.0310445	0.0118636	0.0193442	0.058278	0.0516950
500 GB	2 MB	0.0443224	0.0321827	0.22057	0.1558040	0.00940887	0.0100911	0.0459622	0.0445074	0.018929	0.0195707	0.09361	0.0946820
500 GB	4 MB	0.0628983	0.0609561	0.313389	0.2996860	0.01272267	0.0151415	0.0625267	0.0697581	0.0330826	0.0365332	0.164353	0.1795060
1 TB	256 KB	0.0101954	0.012748	0.05397	0.0585930	0.00457816	0.0044964	0.021732	0.0165670	0.0076348	0.0066072	0.037104	0.0298570
1 TB	512 KB	0.0131226	0.0133328	0.064701	0.0615190	0.00514952	0.0051994	0.0246052	0.0200685	0.0082664	0.0070996	0.040281	0.0323170
1 TB	1 MB	0.0220452	0.0175661	0.104207	0.0826990	0.00627397	0.0065264	0.0302366	0.0266916	0.0107419	0.0102016	0.052668	0.0478290
1 TB	2 MB	0.0430616	0.0313136	0.204266	0.1514580	0.00854432	0.0092494	0.0416374	0.0402996	0.0176748	0.018921	0.087321	0.0914350
1 TB	4 MB	0.0594542	0.0609561	0.286206	0.2996860	0.01323983	0.0144828	0.0628127	0.0664644	0.0319115	0.0365132	0.158512	0.1794080
2 TB	256 KB	0.010301	0.0134875	0.056498	0.0622940	0.00468653	0.004558	0.0222764	0.0168735	0.0079612	0.0069037	0.03875	0.0313400
2 TB	512 KB	0.0143515	0.0139739	0.066743	0.0647270	0.00534735	0.0051994	0.0255921	0.0200685	0.0085648	0.0074355	0.041787	0.0339970
2 TB	1 MB	0.0241827	0.0186976	0.119891	0.0883590	0.00675963	0.0065264	0.0351848	0.0266916	0.0113319	0.0109298	0.05561	0.0514720
2 TB	2 MB	0.0442851	0.0321827	0.210386	0.1558040	0.00929441	0.0092494	0.0433794	0.0402996	0.0182892	0.0195393	0.090398	0.0945290
2 TB	4 MB	0.05766	0.0609561	0.253635	0.299686	0.01082826	0.0144828	0.0510553	0.0664644	0.0326187	0.0365133	0.162034	0.179408

do *DiskSim* e a análise dos modelos não são estatisticamente significantes.

5.2.2 Resultados

Esta seção apresenta os resultados da avaliação dos sistemas de armazenamento híbridos. Inicialmente, um experimento *screening* é apresentado avaliando os fatores e interações que proveem grandes impactos nos sistemas adotados. Em seguida, experimentos adicionais são realizados para cada tipo de operação: escrita, leitura, e *mixed*.

5.2.2.1 Experimento I: *Screening*

Este experimento avalia a magnitude e interação de cada fator. Um planejamento de experimentos (*design of experiment* - DoE) baseado na abordagem fatorial (*factorial*) é levado em consideração ($\prod_{i=1}^{k} l_i$). Como foi apresentado no Capítulo 3, um experimento fatorial significa que todas as possíveis combinações dos níveis dos fatores são investigadas (MONTGOMERY; RUNGER, 2014).

Portanto, este trabalho assume 5 fatores significativos para os sistemas de armazenamento adotados ($k = 5$): tecnologia de armazenamento (*technology*), capacidade de armazenamento (*capacity*), tamanho do objeto (*object*), tipo da operação (*operation*), e o número de nós (*node*). O fator *operation* contempla 3 níveis ($l_{op} = 3$), e os outros fatores levam em consideração apenas 2 níveis: *technology* - HDD e SSD; *capacity* - 250GB e 2TB; *object* - 256KB e 4MB; e *node* - 1 e 3. Dessa forma, $\prod_{i=1}^{5} l_i = 48$ tratamentos são arranjados e avaliados (Tabela A.1). A Tabela 5.5 reúne os fatores e níveis mencionados.

Tabela 5.5: Experimento I - Fatores e níveis

technology	***capacity***	***object***	***operation***	***node***
HDD	250GB	256KB	Escrita	1
SSD	2TB	4MB	Leitura	3
			Mixed	

Para cada tratamento, modelos SPN são gerados, e, então, o tempo médio de resposta e IOPS são estimados através de uma análise estacionária. Os tempos de transição de acordo com o tamanho do objeto são demonstrados nas Tabelas 5.3, para HDD, e 5.6, para SSD e dispositivos híbridos, os quais assumimos que são exponencialmente distribuídos. Estes valores e os resultados da análise estacionária são adotados em todos os experimentos a seguir.

Os resultados foram utilizados para estimar os efeitos de cada fator e interações. O efeito de um fator é definido como a mudança nos valores de resposta gerados quando da alternância do nível de um fator. Este trabalho considera efeitos principais (*main effects*) e interações de segunda ordem (*second-order interactions*), pois as interações de ordem superior são geralmente insignificantes (MONTGOMERY; RUNGER, 2014). As Figuras 5.1 e 5.2 ilustram os efeitos causados quando da variação dos níveis. A Tabela 5.7 demonstra o *rank* para os efeitos,

Tabela 5.6: Tempos das transições para solid-state drives (SSDs) e dispositivos híbridos

	Escrita (s)			Leitura (s)		
	SSD	**SSDasCache**	**SSDRandom**	**SSD**	**SSDasCache**	**SSDRandom**
256KB	0.002341591	0.007413389	0.001710082	0.001437625	0.00203086	0.001283695
512KB	0.004112514	0.008254444	0.002587779	0.001778126	0.002330238	0.001483562
1MB	0.005015605	0.012122439	0.003371909	0.002598002	0.002921418	0.001986167
2MB	0.005610467	0.020084268	0.004584244	0.004026666	0.003574775	0.003299692
4MB	0.011225273	0.026371999	0.008308976	0.008053261	0.006550105	0.006511148

o qual denota *technology*, *object_size* e algumas interações do fator *operation* (por exemplo, $operation(write - read) * node$), como os fatores com o maior impacto nas métricas adotadas. Como *operation* possui 3 níveis, os que são considerados para o cálculo do efeito estão indicados em parênteses (por exemplo, $operation(write - read)$). Outros fatores e interações não impactam significativamente as métricas adotadas, e por isso, não são mostrados.

Analisando os dois fatores com os maiores efeitos (*technology* e *object_size*) observamos que, SSD é o melhor nível para tecnologia, pois reduz significativamente o tempo médio de resposta (84.66%) e IOPS^{-1} (81.41%). Já para o fator *object_size*, o nível 256KB diminui o tempo médio de resposta (73.61%), bem como IOPS^{-1} (70.29%).

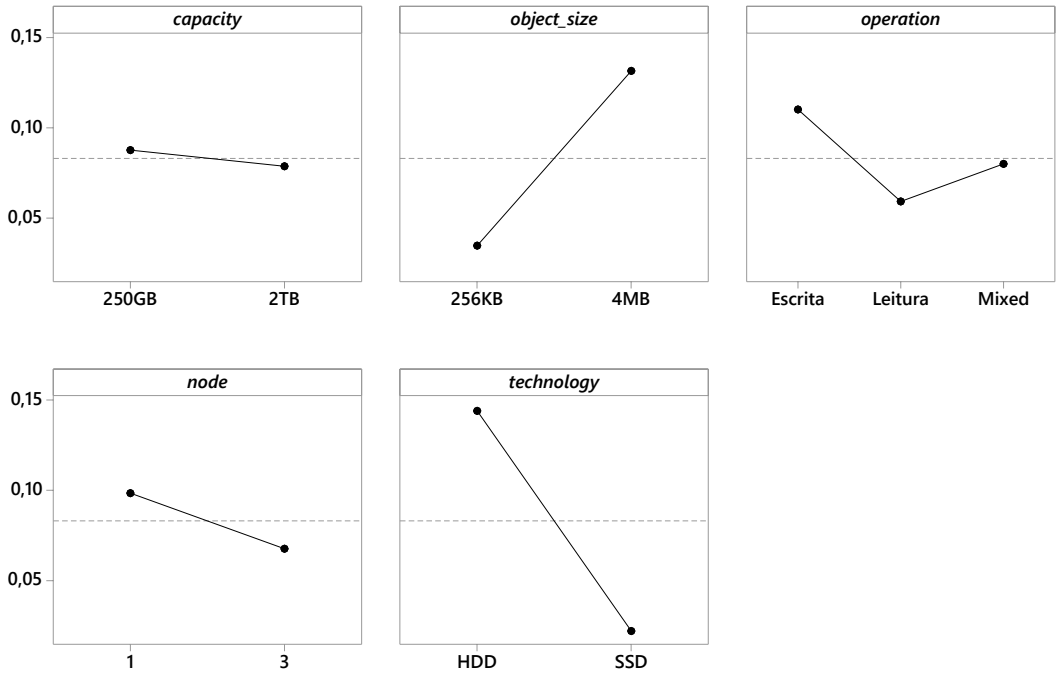


Figura 5.1: Experimento I - gráfico de efeitos para o tempo médio de resposta

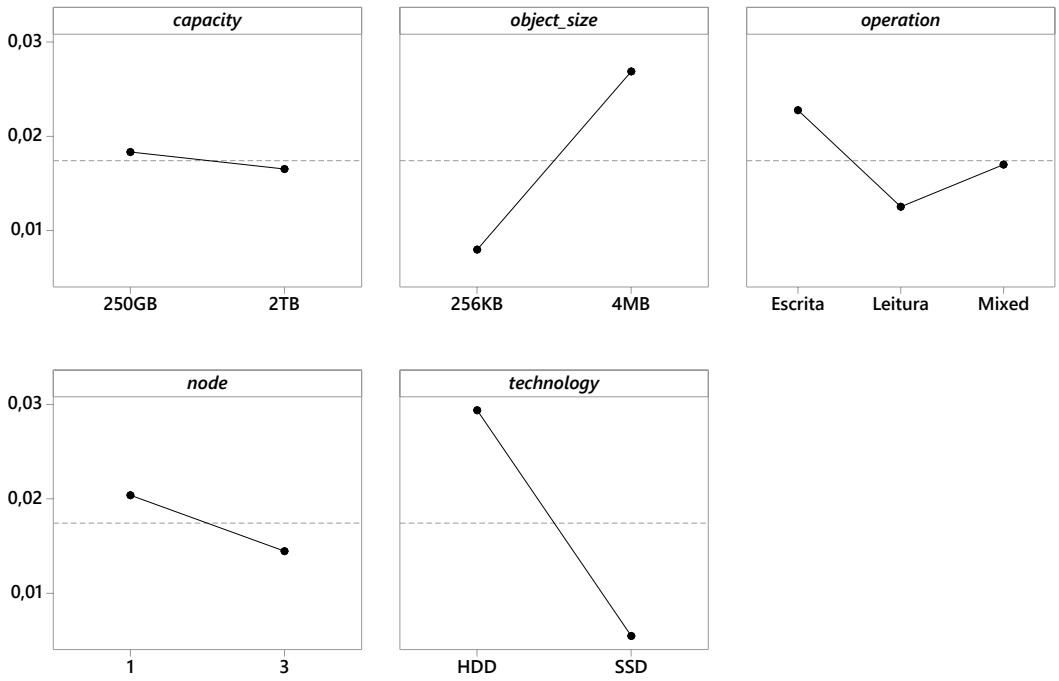


Figura 5.2: Experimento I - gráfico de efeitos para *input/output per second* (IOPS^{-1})

5.2.2.2 Experimento II: Operações de Escrita

Este experimento avalia o desempenho de dispositivos híbridos com diferentes políticas para armazenamento de dados (Seção 2.3), assumindo operações de escrita. Um DoE baseado em um planejamento fracional é adotado, levando em consideração o *rank* obtido no experimento anterior.

Tabela 5.7: Experimento I - *Rank* de efeitos dos fatores e interações

Tempo Médio de Resposta		**IOPS^{-1}**	
Fator / Interação	**Efeito**	**Fator / Interação**	**Efeito**
object_size	0.0967681	*technology*	0.02392785
technology	0.0919669	*object_size*	0.01888748
*object_size*technology*	0.06268113	*object_size*technology*	0.013005015
operation(write-read)	0.0509491	*operation(write-read)*	0.0102281
*operation(write-read)*node*	0.03729775	*operation(write-read)*node*	0.00732548
*object_size*operation(write-read)*	0.0347153	*object_size*operation(write-read)*	0.006831735
*operation(write-read)*technology*	0.0322863	*operation*technology(write-read)*	0.0063368
node	0.0307925	*node*	0.0059244
capacity	0.0089534	*capacity*	0.0017981

A Tabela 5.7 mostra uma variação considerável para os fatores $object\_size$ e $technology$, portanto ambos são considerados para esse experimento. Alguns níveis adicionais foram acrescentados para o fator $object\_size$ (256KB, 512KB, 1MB, 2MB , 4MB), e duas abordagens híbri-

das (HDD + SSD) são levadas em consideração no fator *technology* (HDD, SSD, SSDRandom e SSDasCache). Já os fatores *capacity* e *node* são fixados em 2TB e 3 nós, respectivamente, por serem os níveis desses fatores com os melhores resultados de tempo médio de resposta e IOPS. Porém, para a abordagem híbrida, adotamos o HDD com capacidade de 2TB e o SSD de 250GB, uma capacidade razoável devido ao custo do SSD. Portanto, o planejamento fracional desse experimento é constituído de $\prod_{i=1}^{2} l_i = 20$ tratamentos (Tabela B.1).

As Figuras 5.3 e 5.4 demonstram os resultados para IOPS e tempo médio de resposta, respectivamente. As abordagens híbridas proveem melhores resultados que o HDD, e assumindo dados randômicos, a política de armazenamento SSDRandom obtêm os melhores valores. Limitada pela política *write-back*, ou seja, escritas periódicas no HDD, SSDasCache alcança um desempenho melhor apenas que o HDD. De fato, a diminuição do tempo médio de resposta quando da adoção da política de armazenamento SSDasCache, comparada com HDD, segue: 38.72% (256KB), 33.85% (512KB), 25.78% (1MB), 27.38% (2MB), e 49.46% (4MB). Entretanto, SSDasCache obtém um IOPS menor que o SSD como segue: 68.13% (256KB), 49.99% (512KB), 58.64% (1MB), 72.37% (2MB) e 58.37% (4MB). HDDs possuem algumas limitações para lidar com requisições aleatórias (HSU; SMITH, 2004), o que é destacado pela composição (80% aleatória) da carga de trabalho adotada.

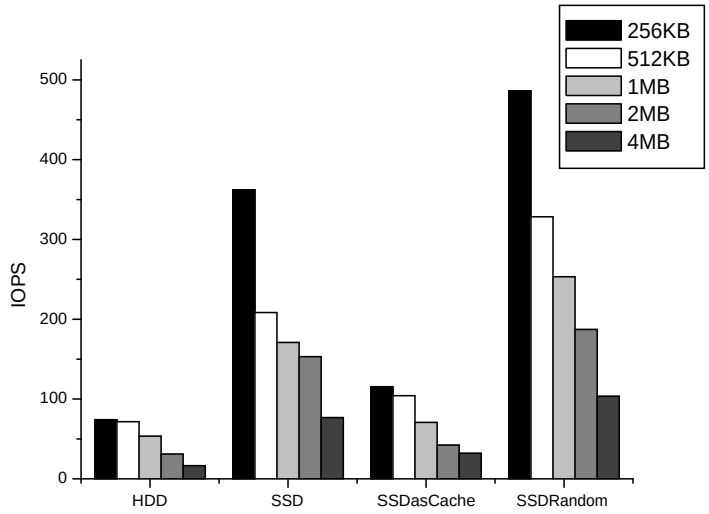


Figura 5.3: Experimento II - *input/output per second* (IOPS) por tamanho do objeto

Já a política de armazenamento SSDRandom é útil para pequenos e grandes objetos. Por exemplo, SSDRandom diminui o tempo médio de resposta em 39.98% e 28.0%, comparado com o SSD, para 256KB e 4MB $object\_size$, respectivamente. Além disso, SSDRandom apresenta um IOPS 34.30% e 34.72% maior que o SSD, para os níveis extremos de $object\_size$.

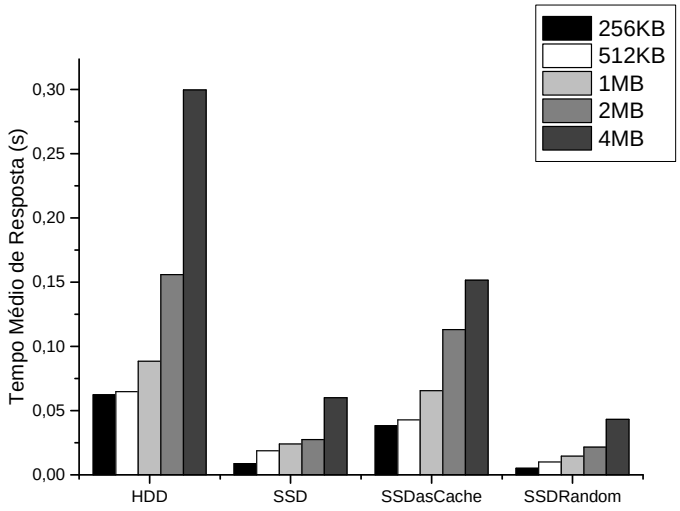


Figura 5.4: Experimento II - tempo médio de resposta por tamanho do objeto

5.2.2.3 Experimento III: Operações de Leitura

Similar ao Experimento II, o desempenho de dispositivos de armazenamento híbridos foram avaliados, levando em consideração o mesmo planejamento de experimento (DoE), porém apenas operações de leitura são consideradas para esse experimento.

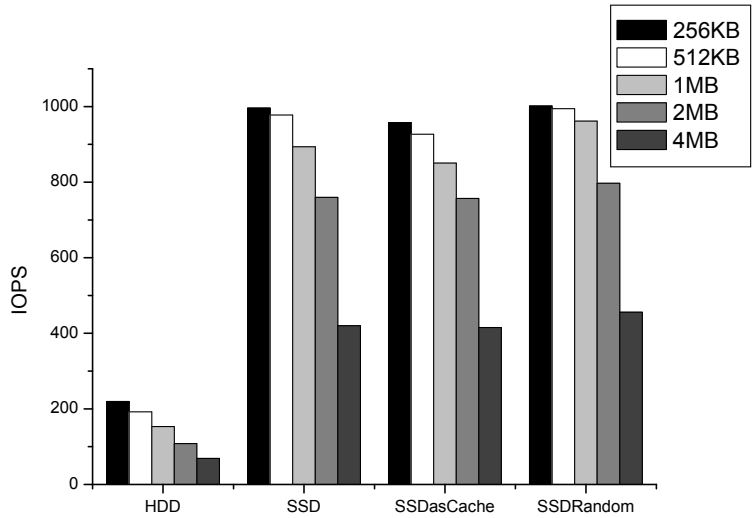


Figura 5.5: Experimento III - *input/output per second* (IOPS) por tamanho do objeto

As Figuras 5.5 e 5.6 mostram o resultado do experimento (Table C.1). Quando comparado com os resultados da operação de escrita (Experimento II), SSDs obtêm melhores resultados com relação às métricas adotadas, e SSDasCache possui um comportamento similar. Entretanto, acessos esporádicos à memória de armazenamento permanente (HDD), quando da inexistência do objeto solicitado na *cache* (*cache miss*), provocam um desempenho menor de

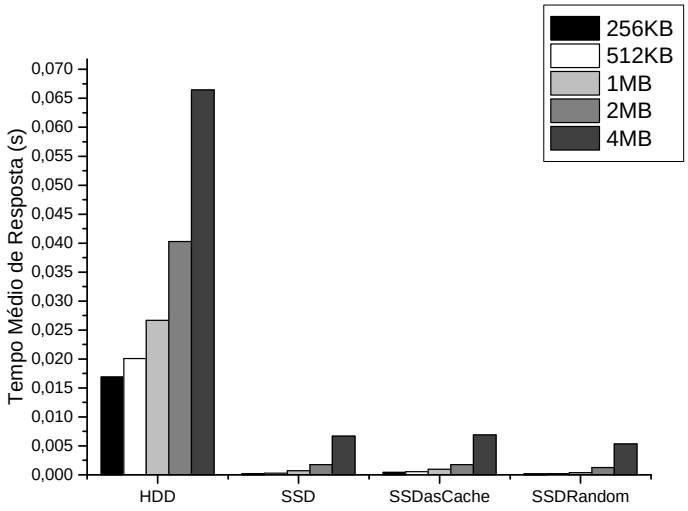


Figura 5.6: Experimento III - tempo médio de resposta por tamanho do objeto

SSDasCache. De fato, os valores a seguir demonstram percentualmente a redução de IOPS quando comparados os resultados de SSD e SSdasCache: 3.8% (256KB), 5.2% (512KB), 4.8% (1MB), 0.37% (2MB) e 1.1% (4MB).

Todavia, a abordagem proposta para lidar com requisições aleatórias (SSDRandom) possui os melhores resultados. Por exemplo, SSDRandom reduz o tempo médio de resposta em 18.61% e 20.35% quando comparado com o SSD, para objetos de 256KB e 4MB (*object_size*), respectivamente. Esta comparação destaca a melhoria obtida quando da adoção de SSDRandom. De fato, SSDRandom apresenta um aumento de 0.5% e 8.5% no IOPS em comparação com SSD, levando em consideração estes mesmos níveis (256KB e 4MB).

5.2.2.4 Experimento IV: Operações *Mixed*

Similar aos experimentos anteriores, operações *mixed* foram avaliadas para diferentes tamanhos de objeto, tecnologias, e políticas de armazenamento. O planejamento do Experimento II (DoE) é mantido, contudo, a carga de trabalho é constituída por operações de escrita e leitura (50% cada).

As Figuras 5.7 e 5.8 descrevem os resultados (Table D.1) de desempenho com relação a *object_size*. Similar ao Experimento II, SSDasCache possui um desempenho maior apenas que HDD devido a existência de requisições de escrita nas operações *mixed*, ou seja, mais acessos ao HDD da composição híbrida. De fato, o tempo médio de resposta de SSDasCache é menor percentualmente que HDD como segue: 40.35% (256KB), 38.17% (512KB), 37.03% (1MB), 40.54% (2MB) e 57.95% (4MB). Entretanto, apresenta os seguintes valores percentuais de redução quando comparado ao SSD, com relação a IOPS: 67.05% (256KB), 49.52% (512KB), 58.24% (1MB), 71.98% (2MB) e 58.01% (4MB).

SSDRandom também provê os melhores resultados para esse experimento. Por exem-

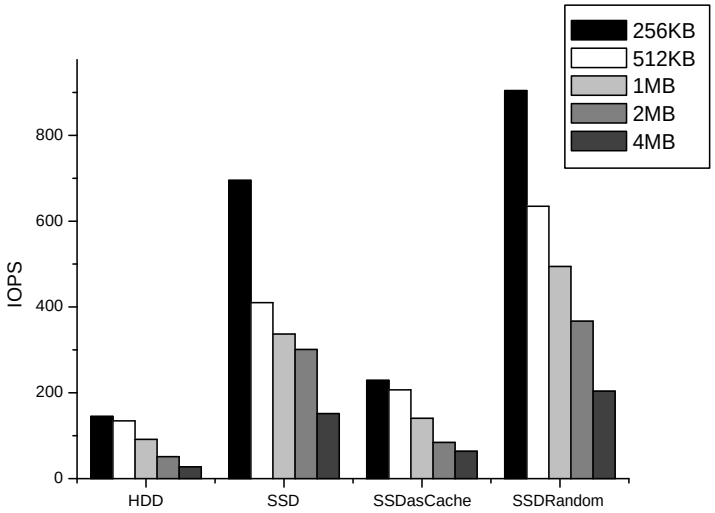


Figura 5.7: Experimento IV - *input/output per second* (IOPS) por tamanho do objeto

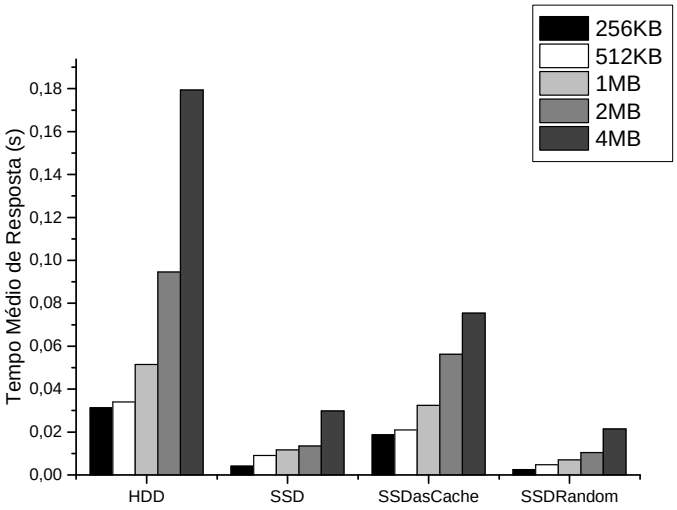


Figura 5.8: Experimento IV - tempo médio de resposta por tamanho do objeto

plo, SSDRandom tem um tempo médio de resposta 39.19% e 28.27% menor que o SSD, e também possui um IOPS 30.07% e 34.34% maior que o SSD, para objetos de 256KB e 4MB ($objects\_size$).

5.3 EXPERIMENTO DE DEPENDABILIDADE

Para avaliar a disponibilidade de sistemas de armazenamento híbridos, este experimento contempla 3 fatores (k = 3): número de nós (*node*); número de *proxies* (*proxy*); e a tecnologia de armazenamento ($technology$). Para o fator $technology$, os níveis são HDD, SSD, e Hybrid (HDD + SSD). Já para o fator *proxy* os níveis são 1 e 2, enquanto que para *node*, 1 e 3 são os

níveis considerados.

A Tabela 5.8 detalha os MTTRs e MTTFs adotados para cada componente, e as respectivas disponibilidades estimadas, as quais foram obtidas de (DARWISH et al., 2015; KIM; MACHIDA; TRIVEDI, 2009; DANTAS et al., 2016). É importante ressaltar que, para o servidor *proxy*, é considerada a disponibilidade de toda a máquina, e não apenas um componente básico específico.

Tabela 5.8: Parâmetros para o *reliability block diagram* (RBD)

Componente	MTTF (horas)	MTTR (horas)	Disponibilidade
PROXY	788.4	1.0	0.998733
NIC	100000.0	0.25	0.9999975
SC	300000.0	9.0	0.99997
MB	5000000.0	0.083	0.99999
PS	50000.0	46.0	0.999080
CPU	10000.0	2.0	0.9998
DIMM	4000000.0	8.0	0.999998
HDD	1400000.0	5.0	0.999996
SSD	500000.0	5.0	0.999990

Similar a 5.2, um planejamento fatorial é levado em consideração, e, para cada tratamento, um modelo RBD é gerado (Seção 4.3). A combinação dos níveis dos fatores adotados para este experimento resulta em 12 tratamentos ($\prod_{i=1}^{3} l_i$) a serem avaliados. A Tabela 5.9 descreve a disponibilidade e o *downtime* para cada tratamento, e a Tabela 5.10 demonstra um *rank* para os efeitos estimados.

Tabela 5.9: Tratamentos e resultados - dependabilidade

proxy	*node*	*technology*	Disponibilidade	*Downtime*
1	1	HDD	0.997577705	21.23337695
2	1	HDD	0.998841422	10.15587582
1	3	HDD	0.998733213	11.10441281
2	3	HDD	0.999998393	0.01408046
1	1	SSD	0.997571293	21.28959154
2	1	SSD	0.998835001	10.21216162
1	3	SSD	0.998733214	11.10441304
2	3	SSD	0.999998393	0.014080688
1	1	Hybrid	0.997581269	21.20214662
2	1	Hybrid	0.998844989	10.12460593
1	3	Hybrid	0.998733214	11.10441269
2	3	Hybrid	0.999998394	0.014080335

Os resultados indicam que *proxy* é o fator mais influente, o qual proporciona um efeito de 0.12% no que diz respeito a disponibilidade, e 11.08 horas/ano com relação a *downtime*. Adicionalmente, o nível Hybrid provoca uma diminuição no *downtime* médio em 0.0282 e 0.015 horas/ano, comparado ao SSD e HDD, respectivamente. Interações também provocam

melhorias. $node * technology$ diminui o *downtime* em 0.043 horas/ano quando considerados os níveis Hybrid e SSD.

Tabela 5.10: *Rank* de efeitos dos fatores e interações - dependabilidade

Disponibilidade		***Downtime***	
Fator / Interação	**Efeito**	**Fator / Interação**	**Efeito**
proxy	0.0012639	*proxy*	11.08395
node	0.0011569	*node*	10.14375
*node*technology(ssd-hybrid)*	0.000005	*node*technology(ssd-hybrid)*	0.04378
technology(ssd-hybrid)	0.000005	*technology(ssd-hybrid)*	0.0282
technology(ssd-hdd)	0.000004	*technology(ssd-hdd)*	0.0282
*node*technology(ssd-hdd)*	0.0000035	*node*technology(ssd-hdd)*	0.02815
*proxy*node*	0.000002	*node*technology(hdd-hybrid)*	0.0156
*node*technology(hdd-hybrid)*	0.000001500	*technology(hdd-hybrid)*	0.0156

5.4 DESEMPENHO X DEPENDABILIDADE X CUSTO

As seções anteriores apresentaram análises individuais para desempenho e disponibilidade. Nesta seção, o impacto dos fatores adotados e os níveis são investigados concomitantemente com relação ao tempo médio de resposta, *downtime* e o custo dos equipamentos.

Particularmente, 3 fatores ($k = 3$) são levados em conta ($node$, $proxy$, $technology$) com os mesmos níveis dos experimentos anteriores. Especificamente, os níveis dos fatores $object\_size$ e $capacity$ são fixados em 256 KB e 2 TB, respectivamente. Apesar da capacidade não possuir uma grande influência nas métricas, este nível (2TB) é assumido por ter apresentado melhores resultados de desempenho. Similarmente, $object\_size$ possui proeminentes resultados nos experimentos de desempenho quando da adoção de 256KB. Além disso, o nível *mixed* ($operation$) é fixado com o intuito de abranger os resultados providos por ambas as operações (escrita e leitura). Com relação ao custo, são considerados os seguintes valores para o HDD, SSD, e servidor *proxy*: US$0.075/GB, US$1/GB (WU et al., 2015) e US$500 (SOUSA et al., 2016). Os tratamentos assumidos, bem como os respectivos resultados associados às métricas consideradas nesta seção são demonstrados na Tabela 5.11.

No entanto, para fins de comparação, se faz necessário que as métricas estejam na mesma escala. Portanto, para que possuam a mesma magnitude, inicialmente uma normatização é realizada de acordo com a Equação 3.4. Além disso, a técnica de distância *Euclidiana* (ED) é adotada (Equação 3.5) para identificar os menores tempos e custo através de um valor único para cada tratamento. Dessa forma, um ponto é o resultado (tempo médio de resposta, *downtime*, custo) de um tratamento com os respectivos níveis para cada fator, e o outro ponto é a origem (0,0,0).

A Tabela 5.12 destaca de forma ordenada, da menor para a maior distância *Euclidiana*, as melhorias obtidas por adotar dispositivos de armazenamento híbridos. Note que os valores

Tabela 5.11: Desempenho x Dependabilidade x Custo - tratamentos e resultados

Tratamento	*technology*	*proxy*	*node*	Tempo Médio de Resposta (s)	*Downtime* (horas/ano)	Custo (US$)
1	HDD	1	3	0.0313399	11.10441281	960.8
2	HDD	1	1	0.0616748	21.23337695	653.6
3	SSD	1	3	0.004096121	11.10441304	6644
4	SSD	1	1	0.005066902	21.28959154	2548
5	SSDasCache	1	3	0.018691634	11.10441269	1710.8
6	SSDasCache	1	1	0.017820183	21.20214662	903.6
7	SSDRandom	1	3	0.002490499	11.10441269	1710.8
8	SSDRandom	1	1	0.004609821	21.20214662	903.6
9	HDD	2	3	0.0313399	0.014080461	1460.8
10	HDD	2	1	0.0616748	10.15587582	1153.6
11	SSD	2	3	0.004096121	0.014080688	6144
12	SSD	2	1	0.005066902	10.21216162	3048
13	SSDasCache	2	3	0.018691634	0.014080335	2210.8
14	SSDasCache	2	1	0.017820183	10.12460593	1403.6
15	SSDRandom	2	3	0.002490499	0.014080335	2210.8
16	SSDRandom	2	1	0.004609821	10.12460593	1403.6

Tabela 5.12: Desempenho x Dependabilidade x Custo - distância *Euclidiana*

Tratamento	Tempo Médio de Resposta (s)	*Downtime* (horas/ano)	Custo (US$)	Distância Euclidiana
15	0	0	0.259949252	0.259949252
13	0.273740403	0	0.259949252	0.377501552
16	0.035808851	0.475218926	0.125200321	0.492737681
9	0.4874503	0.000000006	0.134748932	0.50573223
7	0	0.521272191	0.176482372	0.55033692
14	0.259016046	0.475218926	0.125200321	0.555515491
5	0.273740403	0.521272191	0.176482372	0.614658062
12	0.043531862	0.479334254	0.399706197	0.62563679
1	0.4874503	0.521272197	0.051282051	0.715515455
11	0.027129188	0.000000017	0.91653312	0.916934541
8	0.035808851	0.995889879	0.04173344	0.997406941
6	0.259016046	0.995889879	0.04173344	1.02986778
4	0.043531862	1	0.316239316	1.049715356
10	1	0.476688686	0.08346688	1.110945014
3	0.027129188	0.521272208	1	1.128034001
2	1	0.99735778	0	1.412346466

referentes às métricas estão normatizados segundo a Equação 3.4. A abordagem SSDRandom provê os resultados mais significativos, e a melhor configuração (tratamento 15) leva em consideração 2 *proxies* e 3 nós de armazenamento. Já a política de armazenamento SSDasCache também demonstra melhores combinações que os *drives* HDD e SSD, e neste caso, o arranjo com 2 *proxies* e 3 nós também denota a menor distância *Euclidiana* (tratamento 13). Adicionalmente, estes resultados destacam a influência dos níveis; por exemplo, com exceção para SSDs (devido ao alto custo), configurações com três nós e um *proxy* (tratamentos 1, 5 e 7) proveem menores valores (distância *Euclidiana*) do que com apenas um nó de armazenamento (tratamentos 2, 6 e 8). Um comportamento similar é observado quando dois *proxies* são adotados, mesmo com um aumento substancial do custo, ou seja, com exceção para SSDs (devido ao alto custo por *gigabyte*), os tratamentos com as menores distâncias *Euclidianas* são constituídos por 3 nós de armazenamento.

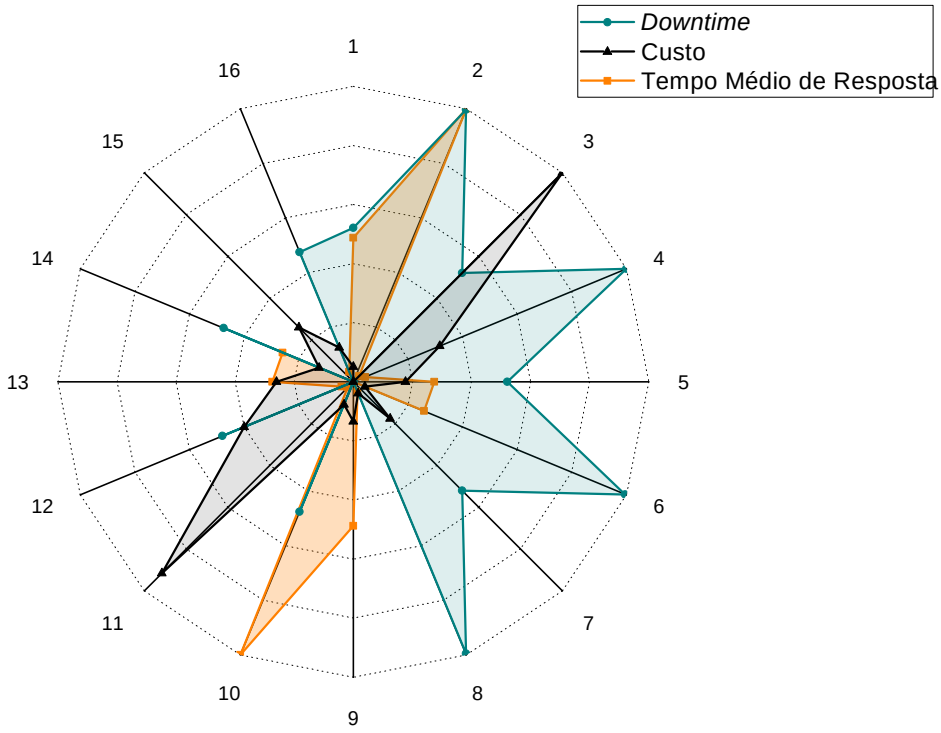


Figura 5.9: Ilustração da distância das métricas dos tratamentos para a origem

Para fins de uma melhor explanação, a Figura 5.9 ilustra os resultados obtidos para as métricas de cada tratamento após a normatização. Fica evidente neste caso os altos valores de tempo médio de resposta quando da adoção de HDDs (tratamentos 2 e 10), justificando dessa forma as elevadas distâncias Euclidiana estimadas para essa tecnologia no *rank* apresentado (Tabela 5.12). É possível destacar também o alto custo apresentado pelos tratamentos 3 e 11, o qual é proveniente do oneroso valor por *gigabyte* dos dispositivos de armazenamento sólido (SSD). Ainda com relação a quantidade de nós de armazenamento, é perceptível a influência negativa nos resultados (distância *Euclidiana*) da quantidade de horas por ano (*downtime*) ao adotar apenas um dispositivo de armazenamento (tratamentos 2, 4, 6 e 8).

É importante enfatizar que os resultados supracitados contemplam as métricas concomitantemente. Entretanto, se uma métrica específica é mais importante do que outras, este *rank* não se sustenta. Por exemplo, considere a infraestrutura de armazenamento de um *data center*, onde o tempo de inatividade é o fator mais crítico para os usuários do serviço. Neste caso, o custo para ressarcimento de possíveis prejuízos decorrentes da indisponibilidade do sistema, possivelmente supera os valores de aquisição dos equipamentos assumidos neste trabalho.

5.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou experimentos com o objetivo de demonstrar a viabilidade dos modelos formais concebidos, bem como da política para dispositivos de armazenamento híbri-

dos sugerida. Inicialmente, detalhou a configuração para os experimentos, como por exemplo, o algoritmo a ser utilizado no simulador *DiskSim*, a composição das cargas de trabalho, e os dispositivos de armazenamento adotados. Em seguida, denotou a validação dos modelos SPNs, para os parâmetros adotados neste trabalho, o qual foi realizado com valores de análises estacionárias (SPN) e do *DiskSim*. Posteriormente, explanou um planejamento de experimento a fim de obter os fatores mais significativos para os dispositivos de armazenamento adotados, quando aplicados a esta arquitetura de armazenamento. Os experimentos de desempenho e disponibilidade demonstraram melhores resultados quando da adoção de dispositivos de armazenamento híbridos. Finalmente, este capítulo abordou uma avaliação concomitante das métricas de desempenho (tempo médio de resposta), dependabilidade (*downtime*), e custo. Esta abordagem resultou em análises que abrangem diversos aspectos e indicou os tratamentos com dispositivos de armazenamento híbridos mais adequados para os cenários considerados neste trabalho.

6

CONCLUSÕES E TRABALHOS FUTUROS

Em sistemas empresariais, uma alta vazão aliada à baixa latência de acesso são essenciais para prover com qualidade os serviços ofertados (MICHELONI; MARELLI; ESHGHI, 2012). O processamento e acesso em tempo real de dados é uma demanda de grande parte das transações atuais. Essas transações são usualmente constituídas por pequenos arquivos aleatórios que carecem de alta confiabilidade e IOPS como pré-requisito. Portanto, diversas pesquisas têm sido feitas com o intuito de aperfeiçoar a disponibilidade e o desempenho dos dispositivos de armazenamento, os quais são frequentemente o gargalo em sistemas computacionais que executam aplicações com intensa carga de operações de entrada e saída (VARKI et al., 2004).

Para diminuir essas adversidades, diversas técnicas são empregadas nos tradicionais *hard disk drives*, como por exemplo, a organização em RAID (*redundant array of inexpensive drives*) e distribuição dos dados em múltiplos discos (MICHELONI; MARELLI; ESHGHI, 2012). Porém, apesar de melhorias nos resultados, os mesmos não são suficientes para atender por completo as demandas de plataformas de computação em nuvem (YAMATO, 2015). Por possuir melhor desempenho e baixo consumo de energia (ZENG et al., 2012), a substituição de HDDs por SSDs é uma alternativa (XIAO et al., 2012) adotada, por exemplo, em algumas instâncias da *Amazon Web Service* (AWS) (TAN; FONG; LIU, 2014). Entretanto, o alto custo por *gigabyte* e baixa durabilidade (comparado ao HDD) torna ainda inviável a utilização exclusiva deste dispositivo em *data centers*.

Dito isto, pesquisas relacionadas a dispositivos híbridos tem atraído a atenção da indústria e academia. Como exemplo deste fato, o sistema operacional *Windows* passou a suportar nativamente dispositivos híbridos a partir da sua versão 7 (MICHELONI; MARELLI; ESHGHI, 2012). Ademais, devido ao seu potencial, uma larga variedade de estudos propõem diversas formas de gerenciamento dos dispositivos. No entanto, a maioria dos trabalhos relacionados levam em consideração apenas um aspecto (desempenho). Este trabalho apresentou uma abordagem baseada em modelos estocásticos para modelagem de desempenho e disponibilidade de sistemas de armazenamento híbridos.

Os modelos de desempenho concebidos são baseados em redes de *Petri* estocásticas e permitem a avaliação de diferentes arquiteturas de armazenamento quando submetidas à requi-

sições de escrita, leitura, e *mixed*. Estes modelos foram validados, para o ambiente adotado, por meio da ferramenta de simulação de arquiteturas de armazenamento *DiskSim*. Além disso, planejamentos de experimento (DoE) foram realizados para prover a análise de tecnologias tradicionais (HDD e SSD) e das políticas de armazenamento abordadas para sistemas de armazenamento híbridos (SSDasCache e SSDRandom). De fato, a abordagem híbrida SSDRandom apresentou os melhores índices de desempenho, alcançando valores de IOPS maiores do que SSDs: 34.30%, 0.5%, e 30.07% para objetos de 256KB; e 34.72%, 8.5%, e 34.34%, para objetos de 4MB, para operações de escrita, leitura e *mixed*, respectivamente.

Este trabalho também apresentou um experimento de disponibilidade utilizando RBDs com o intuito de estimar os efeitos da adoção de dispositivos híbridos na arquitetura de armazenamento do sistema assumido (*OpenStack Swift*). Dentro desse contexto, a quantidade de *proxies* foi o fator mais influente, com um efeito de 0.12% e 11.08 horas/ano no que diz respeito à disponibilidade e *downtime*, respectivamente. Com relação à tecnologia de armazenamento, a abordagem híbrida obteve os melhores valores de disponibilidade (0.999998394) e *downtime* (0.01408033 horas/ano), mesmo com a utilização de SSDs.

Para avaliar concomitantemente os resultados de desempenho, disponibilidade, e custo, este trabalho adotou a técnica de distância *Euclidiana*. Levando em consideração esses três aspectos, os resultados confirmam a influência negativa do custo e *downtime* de SSDs puros, acarretando em configurações menos indicadas (distância *Euclidiana* grande). Já para HDDs, por possuírem um custo por *gigabyte* bem menor que SSDs, constatamos melhores resultados, com exceção quando da adoção de apenas um nó, devido ao seu alto tempo médio de resposta. As políticas de armazenamento para sistemas de armazenamento híbridos apresentaram os melhores tratamentos. Como o custo e *downtime* para os dispositivos híbridos possuem os mesmos valores, a diferença da distância observada entre as duas políticas de armazenamento (SSDasCache e SSDRandom) é devida ao desempenho (tempo médio de resposta) obtido pela abordagem SSDRandom, compondo esta a configuração mais adequada, dentro dos cenários assumidos neste trabalho.

6.1 CONTRIBUIÇÕES

Esta seção apresenta as principais contribuições desenvolvidas neste trabalho, as quais são listadas a seguir:

- **Modelos formais de desempenho:** este trabalho propôs modelos baseados em espaço de estados (redes de *Petri* estocásticas) capazes de expressar operações de escrita, leitura, e *mixed* (50% escrita e 50% leitura). Estes modelos permitem estimar a vazão e o tempo médio de resposta de sistemas de armazenamentos híbridos aplicados em arquiteturas com nós de armazenamento redundantes. Além disso, os modelos são validados, para os parâmetros adotados neste trabalho, com o auxílio da ferramenta de representação de sistemas de armazenamento *DiskSim*;

- **Modelo formal de disponibilidade:** esta dissertação apresentou o modelo RBD (diagrama de blocos de confiabilidade) concebido para estimar a disponibilidade de sistemas de armazenamento híbridos. Este modelo representa detalhadamente os componentes existentes em um nó de armazenamento de forma a levar em consideração o efeito de cada elemento na disponibilidade do sistema;

- **Metodologia para avaliação de sistemas de armazenamento híbridos:** a metodologia proposta neste trabalho proporciona um auxílio na tomada de decisões a respeito da arquitetura de armazenamento a ser adotada em um determinado *data center*. A adoção dos modelos formais propostos em conjunto com os planejamentos de experimentos de desempenho e dependabilidade permite a avaliação de arquiteturas e diferentes políticas de armazenamento para dispositivos híbridos, ainda na fase de concepção. É importante ressaltar que a utilização dos modelos concebidos está condicionada a um embasamento prévio a respeito dos formalismos baseados;

- **Planejamento de tecnologia e política de armazenamento conforme requisitos de desempenho:** esta dissertação adotou um planejamento de experimentos (DoE) para a avaliação de desempenho de sistemas de armazenamento híbridos ainda não relatado em trabalhos passados. Através desta metodologia é possível identificar e eliminar os fatores menos significativos a fim de permitir uma análise mais precisa, sem ruídos, em experimentos que avaliem a vazão e o tempo médio de resposta quando da requisição de operações de escrita, leitura e *mixed*;

- **Planejamento de adoção de tecnologia conforme requisitos de disponibilidade:** este planejamento proposto permite a avaliação da influência do MTTF e MTTR dos componentes de um sistema de armazenamento híbrido na disponibilidade e *downtime* do sistema como um todo. Para isto, é levado em consideração fatores como a tecnologia e a redundância adotada;

- **Planejamento de configuração de armazenamento conforme aspectos de desempenho, disponibilidade e custo:** o trabalho apresentado nesta dissertação propõe a avaliação simultânea de aspectos que afetem o desempenho, dependabilidade, e o custo de sistemas de armazenamento híbrido. Dessa forma, com o auxílio da técnica de distância *Euclidiana*, é possível identificar os tratamentos (configurações) mais adequadas para o sistema de armazenamento concebido.

6.2 TRABALHOS FUTUROS

Esta seção lista algumas oportunidades para trabalhos futuros, as quais podem ser vistos como extensões dentro do contexto abordado nessa dissertação:

- **Consumo de energia:** este trabalho levou em consideração apenas o custo dos equipamentos adotados para uma determinada configuração. No entanto, é possível aferir o custo devido à quantidade de *joules* [1] consumidos quando da execução de cada operação de leitura, escrita ou *mixed* (PARK et al., 2011). Dessa forma, é possível estender os modelos analíticos de desempenho para estimar também esta métrica;

- **Performabilidade:** as métricas de desempenho e a disponibilidade são estimadas neste trabalho através de modelos independentes (SPNs e RBD). No entanto, a integração desses aspectos pode ser feita através de modelagem hierárquica para combinar um modelo de dependabilidade de alto nível com modelos de desempenho de baixo nível. Dessa forma, a performabilidade irá descrever o efeito da indisponibilidade na degradação do desempenho dos sistemas de armazenamento híbridos (XIE; DAI; POH, 2004);

- **GRASP:** a metodologia apresentada neste trabalho para definir a melhor configuração para os sistemas de armazenamento pode ser otimizada através do Procedimento de Busca Adaptativa Aleatória Gulosa (Greedy Randomized Adaptive Search Procedure - GRASP). Esta metaheurística é caracterizada por prover uma solução de boa qualidade (subconjunto) dentro de um conjunto finito de elementos de acordo com um número máximo de interações definido (FEO; RESENDE, 1995);

- **Carga de trabalho e *storages*:** uma possível extensão a este trabalho é submeter os dispositivos de armazenamento a cargas de trabalho com uma variedade maior de parâmetros, a fim de investigar possíveis aspectos não observados nos resultados apresentados nesta dissertação. Por exemplo, submeter objetos menores do que 256KB e maiores que 4MB, além de diversificar a porcentagem de dados aleatórios e sequenciais. Além disso, incrementar a gama de *storages* avaliados pode vir a fornecer dados não obtidos nos experimentos realizados;

- **Plataformas de computação em nuvem:** a modelagem formal apresentada leva em consideração a arquitetura de armazenamento da plataforma *OpenStack Swift*. Uma possível extensão deste trabalho é avaliar os modelos concebidos e alterá-los, se necessário, para se adaptarem a diferentes cenários, como por exemplo as plataformas *Eucalyptus* (EUCALIPTUS, 2017) e *OpenNebula* (OPENNEBULA, 2017).

[1] 1 quilowatt-hora corresponde a 3.6MJ

REFERÊNCIAS

AGRAWAL, N. et al. Design Tradeoffs for SSD Performance. In: **USENIX Annual Technical Conference**. [S.l.: s.n.], 2008. p.57–70.

AJMONE MARSAN, M.; CONTE, G.; BALBO, G. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. **ACM Transactions on Computer Systems (TOCS)**, [S.l.], v.2, n.2, p.93–122, 1984.

AJMONE MARSAN, M.; CONTE, G.; BALBO, G. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. **ACM Transactions on Computer Systems (TOCS)**, [S.l.], v.2, n.2, p.93–122, 1984.

AL MAMUN, A.; GUO, G.; BI, C. **Hard disk drive**: mechatronics and control. [S.l.]: CRC press, 2006. v.23.

ALSUHIBANY, S. A. et al. Analysis of free-text keystroke dynamics for Arabic language using Euclidean distance. In: **Innovations in Information Technology (IIT), 2016 12th International Conference on**. [S.l.: s.n.], 2016. p.1–6.

APPUSWAMY, R.; MOOLENBROEK, D. C. van; TANENBAUM, A. S. Integrating flash-based SSDs into the storage stack. In: **Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on**. [S.l.: s.n.], 2012. p.1–12.

AVIZIENIS, A. et al. **Fundamental concepts of dependability**. [S.l.]: University of Newcastle upon Tyne, Computing Science, 2001.

BALBO, G. Introduction to stochastic Petri nets. In: **Lectures on Formal Methods and PerformanceAnalysis**. [S.l.]: Springer, 2001. p.84–155.

BAUSE, F.; KRITZINGER, P. S. **Stochastic Petri Nets**. [S.l.]: Vieweg Wiesbaden, 2002. v.1.

BERNARDI, S.; MERSEGUER, J.; PETRIU, D. C. Dependability modeling and analysis of software systems specified with UML. **ACM Computing Surveys (CSUR)**, [S.l.], v.45, n.1, p.2, 2012.

BISWAS, P.; PATWA, F.; SANDHU, R. Content level access control for openstack swift storage. In: **Proceedings of the 5th ACM Conference on Data and Application Security and Privacy**. [S.l.: s.n.], 2015. p.123–126.

BOLCH, G. et al. **Queueing networks and Markov chains**: modeling and performance evaluation with computer science applications. [S.l.]: John Wiley & Sons, 2006.

BOUHMALA, N. How Good is the Euclidean Distance Metric for the Clustering Problem. In: **Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress on**. [S.l.: s.n.], 2016. p.312–315.

BREWER, J.; GILL, M. **Nonvolatile Memory Technologies with Emphasis on Flash**: a comprehensive guide to understanding and using flash memory devices. [S.l.]: John Wiley & Sons, 2011. v.8.

BU, K. et al. The optimization of the hierarchical storage system based on the hybrid ssd technology. In: **Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on**. [S.l.: s.n.], 2012. p.1323–1326.

BUCY, J. S. et al. The disksim simulation environment version 4.0 reference manual (cmu-pdl-08-101). **Parallel Data Laboratory**, [S.l.], p.26, 2008.

CARNS, P. et al. Understanding and improving computational science storage access through continuous characterization. **ACM Transactions on Storage (TOS)**, [S.l.], v.7, n.3, p.8, 2011.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to discrete event systems**. [S.l.]: Springer Science & Business Media, 2009.

CHEKAM, T. T. et al. On the synchronization bottleneck of OpenStack Swift-like cloud storage systems. In: **Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on**. [S.l.: s.n.], 2016. p.1–9.

CHEN, B. M. et al. Hard Disk Drive Servo Systems. **Advances in Industrial Control** (, [S.l.], 2006.

CHEN, F.; DING, X.; JIANG, S. Exploiting disk layout and block access history for i/o prefetch. **Advanced Operating Systems and Kernel Applications: Techniques and Technologies: Techniques and Technologies**, [S.l.], p.201, 2009.

CHEN, F.; KOUFATY, D. A.; ZHANG, X. Hystor: making the best use of solid state drives in high performance storage systems. In: **Proceedings of the international conference on Supercomputing**. [S.l.: s.n.], 2011. p.22–32.

DANTAS, J. et al. Hierarchical model and sensitivity analysis for a cloud-based VoD streaming service. In: **Dependable Systems and Networks Workshop, 2016 46th Annual IEEE/IFIP International Conference on**. [S.l.: s.n.], 2016. p.10–16.

DARWISH, K. O. et al. Towards Reliable Mobile Cloud Computing. In: **P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015 10th International Conference on**. [S.l.: s.n.], 2015. p.127–133.

EBELING, C. E. **An introduction to reliability and maintainability engineering**. [S.l.]: Tata McGraw-Hill Education, 2004.

EL MAGHRAOUI, K. et al. Modeling and simulating flash based solid-state disks for operating systems. In: **Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering**. [S.l.: s.n.], 2010. p.15–26.

EUCALIPTUS. **Site Oficial**. Acessado em: 16/05/2017, Disponível em: `http://www.dxc.technology`.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. **Journal of global optimization**, [S.l.], v.6, n.2, p.109–133, 1995.

FRANCÊS, C. R. L. Introdução às redes de petri. **Laboratório de Computação Aplicada, Universidade Federal do Pará**, [S.l.], 2003.

GUO, L. et al. Dynamic performance optimization for cloud computing using m/m/m queueing system. **Journal of Applied Mathematics**, [S.l.], v.2014, 2014.

HSU, W. W.; SMITH, A. J. The performance impact of I/O optimizations and disk improvements. **IBM Journal of Research and Development**, [S.l.], v.48, n.2, p.255–289, 2004.

JAIN, R. **The art of computer systems performance analysis**: techniques for experimental design, measurement, simulation, and modeling. [S.l.]: John Wiley & Sons, 1990.

JIN, J.-H.; LEE, M.-J. On supporting user-defined collaborative workspaces over cloud storage. In: **Information and Communication Technology Convergence (ICTC), 2015 International Conference on**. [S.l.: s.n.], 2015. p.1003–1008.

JOHANNESSON, P.; PERJONS, E. **An introduction to design science**. [S.l.]: Springer, 2014.

JOO, Y. et al. Rapid prototyping and evaluation of intelligence functions of active storage devices. **IEEE Transactions on Computers**, [S.l.], v.63, n.9, p.2356–2368, 2014.

KAMATH, M.; VISWANADHAM, N. Applications of Petri net based models in the modelling and analysis of flexible manufacturing systems. In: **Robotics and Automation. Proceedings. 1986 IEEE International Conference on**. [S.l.: s.n.], 1986. v.3, p.312–317.

KANOUN, K.; SPAINHOWER, L. **Dependability benchmarking for computer systems**. [S.l.]: John Wiley & Sons, 2008. v.72.

KAPADIA, A.; VARMA, S.; RAJANA, K. **Implementing Cloud Storage with OpenStack Swift**. [S.l.]: Packt Publishing Ltd, 2014.

KAPUR, K. C.; PECHT, M. **Reliability engineering**. [S.l.]: John Wiley & Sons, 2014.

KHAZAEI, H.; MISIC, J.; MISIC, V. B. Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems. **IEEE Transactions on parallel and distributed systems**, [S.l.], v.23, n.5, p.936–943, 2012.

KHEDHER, O. **Mastering OpenStack**. [S.l.]: Packt Publishing Ltd, 2015.

KIM, D. S.; MACHIDA, F.; TRIVEDI, K. S. Availability modeling and analysis of a virtualized system. In: **Dependable Computing, 2009. PRDC'09. 15th IEEE Pacific Rim International Symposium on**. [S.l.: s.n.], 2009. p.365–371.

KIM, M. et al. Enhanced dual Bloom filter based on SSD for efficient directory parsing in cloud storage system. In: **Computing, Networking and Communications (ICNC), 2015 International Conference on**. [S.l.: s.n.], 2015. p.413–417.

KIM, T.; NO, J. Utilizing flash-memory SSD for developing hybrid filesystem. In: **System Integration (SII), 2014 IEEE/SICE International Symposium on**. [S.l.: s.n.], 2014. p.700–705.

KIM, Y. et al. HybridStore: a cost-efficient, high-performance storage system combining ssds and hdds. In: **Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on**. [S.l.: s.n.], 2011. p.227–236.

KULKARNI, S.; JISHA, P. Study of Bad block management and Wear leveling in NAND flash memories. , [S.l.], 2013.

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

LAGA, A. et al. Lynx: a learning linux prefetching mechanism for ssd performance model. In: **Non-Volatile Memory Systems and Applications Symposium (NVMSA), 2016 5th**. [S.l.: s.n.], 2016. p.1–6.

LEE, D.; MIN, C.; EOM, Y. I. Effective flash-based SSD caching for high performance home cloud server. **IEEE Transactions on Consumer Electronics**, [S.l.], v.61, n.2, p.215–221, 2015.

LEE, S. et al. Performance analysis of ssd/hdd hybrid storage manager. In: **Nano, Information Technology and Reliability (NASNIT), 2011 15th North-East Asia Symposium on**. [S.l.: s.n.], 2011. p.136–139.

LEE, Y. K.; PARK, S. J. OPNets: an object-oriented high-level petri net model for real-time system modeling. **Journal of Systems and Software**, [S.l.], v.20, n.1, p.69–86, 1993.

LI, Y.; LEE, P. P.; LUI, J. Stochastic modeling of large-scale solid-state storage systems: analysis, design tradeoffs and optimization. In: **ACM SIGMETRICS Performance Evaluation Review**. [S.l.: s.n.], 2013. v.41, n.1, p.179–190.

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

MACIEL, P. et al. Performance and dependability in service computing: concepts, techniques and research directions, ser. **Premier Reference Source. Igi Global**, [S.l.], 2011.

MAO, B. et al. HPDA: a hybrid parity-based disk array for enhanced performance and reliability. **ACM Transactions on Storage (TOS)**, [S.l.], v.8, n.1, p.4, 2012.

MAO, B.; WU, S.; JIANG, H. Improving storage availability in cloud-of-clouds with hybrid redundant data distribution. In: **Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International**. [S.l.: s.n.], 2015. p.633–642.

MARSAN, M. A. et al. **Modelling with generalized stochastic Petri nets**. [S.l.]: John Wiley & Sons, Inc., 1994.

MEISTER, D.; BRINKMANN, A. dedupv1: improving deduplication throughput using solid state drives (ssd). In: **Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on**. [S.l.: s.n.], 2010. p.1–6.

MERLIN, P.; FARBER, D. Recoverability of communication protocols–implications of a theoretical study. **IEEE transactions on Communications**, [S.l.], v.24, n.9, p.1036–1043, 1976.

MICHELONI, R.; MARELLI, A.; ESHGHI, K. **Inside solid state drives (SSDs)**. [S.l.]: Springer Science & Business Media, 2012. v.37.

MODARRES, M.; KAMINSKIY, M. P.; KRIVTSOV, V. **Reliability engineering and risk analysis**: a practical guide. [S.l.]: CRC press, 2009.

MOLLOY, M. K. Performance analysis using stochastic Petri nets. **IEEE Transactions on computers**, [S.l.], v.31, n.9, p.913–917, 1982.

MONTGOMERY, D. C.; RUNGER, G. C. **Applied statistics and probability for engineers**. [S.l.]: John Wiley & Sons, 2014.

MURATA, T. Petri nets: properties, analysis and applications. **Proceedings of the IEEE**, [S.l.], v.77, n.4, p.541–580, 1989.

NIJIM, M. et al. DM-pas: a data mining prefetching algorithm for storage system. In: **High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on**. [S.l.: s.n.], 2011. p.500–505.

O'CONNOR, P. D.; O'CONNOR, P.; KLEYNER, A. **Practical reliability engineering**. [S.l.]: John Wiley & Sons, 2012.

OEHLERT, G. W. **A first course in design and analysis of experiments**. [S.l.: s.n.], 2010.

OPENNEBULA. **Site Oficial**. Acessado em: 16/05/2017, Disponível em: `https://opennebula.org`.

PARK, J. K.; SUH, D. H.; BAEK, S. A study of divided disk cache performance using DiskSim. In: **Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on**. [S.l.: s.n.], 2016. p.1867–1872.

PARK, S. et al. A comprehensive study of energy efficiency and performance of flash-based SSD. **Journal of Systems Architecture**, [S.l.], v.57, n.4, p.354–365, 2011.

PETRI, C. A. Kommunikation mit automaten. , [S.l.], 1962.

RAMASAMY, A. S.; KARANTHARAJ, P. RFFE: a buffer cache management algorithm for flash-memory-based ssd to improve write performance. **Canadian Journal of Electrical and Computer Engineering**, [S.l.], v.38, n.3, p.219–231, 2015.

RAUSAND, M.; ARNLJOT, H. et al. **System reliability theory**: models, statistical methods, and applications. [S.l.]: John Wiley & Sons, 2004. v.396.

REISIG, W. **Understanding petri nets**. [S.l.]: Springer, 2013.

RICHTER, D. **Flash memories**: economic principles of performance, cost and reliability optimization. [S.l.]: Springer Science & Business Media, 2013.

ROSADO, T.; BERNARDINO, J. An overview of openstack architecture. In: **Proceedings of the 18th International Database Engineering & Applications Symposium**. [S.l.: s.n.], 2014. p.366–367.

SCHROEDER, B.; GIBSON, G. A. Understanding disk failure rates: what does an mttf of 1,000,000 hours mean to you? **ACM Transactions on Storage (TOS)**, [S.l.], v.3, n.3, p.8, 2007.

SILVA, B. et al. Astro: an integrated environment for dependability and sustainability evaluation. **Sustainable computing: informatics and systems**, [S.l.], v.3, n.1, p.1–17, 2013.

SOUSA, E. et al. A modeling strategy for cloud infrastructure planning considering performance and cost requirements. **Services Transactions on Cloud Computing (STCC)**, [S.l.], v.4(1), n.June, p.32–45, 2016.

STRUNK, J. D. Hybrid Aggregates: combining ssds and hdds in a single storage pool. **ACM SIGOPS Operating Systems Review**, [S.l.], v.46, n.3, p.50–56, 2012.

SUN, F.-B.; PARKHOMOVSKY, A. Physics-based life distribution and reliability modeling of SSD. In: **Reliability and Maintainability Symposium (RAMS), 2013 Proceedings-Annual**. [S.l.: s.n.], 2013. p.1–6.

TAN, W.; FONG, L.; LIU, Y. Effectiveness assessment of solid-state drive used in big data services. In: **Web Services (ICWS), 2014 IEEE International Conference on**. [S.l.: s.n.], 2014. p.393–400.

TRIVEDI, K. S. **Probability & statistics with reliability, queuing and computer science applications**. [S.l.]: John Wiley & Sons, 2008.

VAIDYA, R.; PUNDKAR, S. Optimized data migration within OpenStack cloud. In: **Bombay Section Symposium (IBSS), 2015 IEEE**. [S.l.: s.n.], 2015. p.1–5.

VALMARI, A. The state explosion problem. **Lectures on Petri nets I: Basic models**, [S.l.], p.429–528, 1998.

VARKI, E. et al. Issues and challenges in the performance analysis of real disk arrays. **IEEE Transactions on Parallel and Distributed Systems**, [S.l.], v.15, n.6, p.559–574, 2004.

WAN, L. et al. SSD-optimized workload placement with adaptive learning and classification in HPC environments. In: **Mass Storage Systems and Technologies (MSST), 2014 30th Symposium on**. [S.l.: s.n.], 2014. p.1–6.

WANG, S. X.; TARATORIN, A. M. **Magnetic Information Storage Technology**: a volume in the electromagnetism series. [S.l.]: Academic press, 1999.

WANG, Y.; LI, X. Achieve high availability about point-single failures in OpenStack. In: **Computer Science and Network Technology (ICCSNT), 2015 4th International Conference on**. [S.l.: s.n.], 2015. v.1, p.45–48.

WATSON, J.; DESROCHERS, A. A. Applying generalized stochastic petri nets to manufacturing systems containing nonexponential transition functions. **IEEE transactions on systems, man, and cybernetics**, [S.l.], v.21, n.5, p.1008–1017, 1991.

WOO, Y.-J.; KIM, J.-S. Diversifying wear index for MLC NAND flash memory to extend the lifetime of SSDs. In: **Embedded Software (EMSOFT), 2013 Proceedings of the International Conference on**. [S.l.: s.n.], 2013. p.1–10.

WU, D. et al. Understanding the impacts of solid-state storage on the Hadoop performance. In: **Advanced Cloud and Big Data (CBD), 2013 International Conference on**. [S.l.: s.n.], 2013. p.125–130.

WU, G.; HE, X. Delta-FTL: improving ssd lifetime via exploiting content locality. In: **Proceedings of the 7th ACM european conference on Computer Systems**. [S.l.: s.n.], 2012. p.253–266.

WU, S. et al. HM: a column-oriented mapreduce system on hybrid storage. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], v.27, n.12, p.3304–3317, 2015.

XIAO, W. et al. PASS: a hybrid storage system for performance-synchronization tradeoffs using ssds. In: **Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on**. [S.l.: s.n.], 2012. p.403–410.

XIE, M.; DAI, Y.-S.; POH, K.-L. **Computing system reliability**: models and analysis. [S.l.]: Springer Science & Business Media, 2004.

XU, C. et al. An SSD-HDD Integrated Storage Architecture for Write-Once-Read-Once Applications on Clusters. In: **Cluster Computing (CLUSTER), 2015 IEEE International Conference on**. [S.l.: s.n.], 2015. p.74–77.

YAMATO, Y. Automatic verification technology of software patches for user virtual environments on IaaS cloud. **Journal of Cloud Computing**, [S.l.], v.4, n.1, p.4, 2015.

YANG, B.; TAN, F.; DAI, Y.-S. Performance evaluation of cloud service considering fault recovery. **The Journal of Supercomputing**, [S.l.], v.65, n.1, p.426–444, 2013.

ZENG, L. et al. HRAID6ML: a hybrid raid6 storage architecture with mirrored logging. In: **Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on**. [S.l.: s.n.], 2012. p.1–6.

ZHENG, Q. et al. COSBench: cloud object storage benchmark. In: **Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering**. [S.l.: s.n.], 2013. p.199–210.

ZIMMERMANN, A. et al. Towards version 4.0 of TimeNET. In: **Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2006 13th GI/ITG Conference**. [S.l.: s.n.], 2006. p.1–4.

ZUBEREK, W. M. Timed Petri nets and preliminary performance evaluation. In: **Proceedings of the 7th annual symposium on Computer Architecture**. [S.l.: s.n.], 1980. p.88–96.

APÊNDICE

A

EXPERIMENTO *SCREENING* - TRATAMENTOS

Tabela A.1: Tratamentos e resultados - experimento screening

capacity	*object*	*operation*	*node*	*technology*	$\mathbf{IOPS^{-1}}$	**Tempo Médio de Resposta (s)**
250GB	256KB	Escrita	3	HDD	0.0222882	0.106319
250GB	4MB	Escrita	3	HDD	0.0643413	0.316612
250GB	256KB	Leitura	3	HDD	0.0066571	0.0273444
250GB	4MB	Leitura	3	HDD	0.0185085	0.0865928
250GB	256KB	*Mixed*	3	HDD	0.0113429	0.053538
250GB	4MB	*Mixed*	3	HDD	0.032663	0.160139
250GB	256KB	Escrita	3	SSD	0.0029419	0.009527724
250GB	4MB	Escrita	3	SSD	0.0122125	0.055913556
250GB	256KB	Leitura	3	SSD	0.0010035	0.0002047
250GB	4MB	Leitura	3	SSD	0.0026904	0.00767544
250GB	256KB	*Mixed*	3	SSD	0.0014385	0.004096137
250GB	4MB	*Mixed*	3	SSD	0.0065967	0.029821095
250GB	256KB	Escrita	1	HDD	0.018976496	0.093868232
250GB	4MB	Escrita	1	HDD	0.054161631	0.269757579
250GB	256KB	Leitura	1	HDD	0.01990558	0.098513658
250GB	4MB	Leitura	1	HDD	0.053137919	0.264642237
250GB	256KB	*Mixed*	1	HDD	0.0194232	0.096102341
250GB	4MB	*Mixed*	1	HDD	0.0535083	0.266492714
250GB	256KB	Escrita	1	SSD	0.0024939	0.006591855
250GB	4MB	Escrita	1	SSD	0.0104815	0.051309263
250GB	256KB	Leitura	1	SSD	0.0014297	0.0051634
250GB	4MB	Leitura	1	SSD	0.0080533	0.039195858
250GB	256KB	*Mixed*	1	SSD	0.0059588	0.005078162
250GB	4MB	*Mixed*	1	SSD	0.0095666	0.046760257
2TB	256KB	Escrita	3	HDD	0.0134875	0.0622935
2TB	4MB	Escrita	3	HDD	0.060956129	0.299685566
2TB	256KB	Leitura	3	HDD	0.0045580	0.0168735
2TB	4MB	Leitura	3	HDD	0.014482775	0.066464362
2TB	256KB	*Mixed*	3	HDD	0.0069037	0.0313399
2TB	4MB	*Mixed*	3	HDD	0.0365133	0.179408391
2TB	256KB	Escrita	3	SSD	0.0027620	0.008636495
2TB	4MB	Escrita	3	SSD	0.0130338	0.060023509
2TB	256KB	Leitura	3	SSD	0.001003861	0.000206891
2TB	4MB	Leitura	3	SSD	0.00269044	0.00767544
2TB	256KB	*Mixed*	3	SSD	0.0014385	0.004096121
2TB	4MB	*Mixed*	3	SSD	0.006596749	0.029821095
2TB	256KB	Escrita	1	HDD	0.011549144	0.056739263
2TB	4MB	Escrita	1	HDD	0.060789634	0.302891945
2TB	256KB	Leitura	1	HDD	0.013674571	0.067362961
2TB	4MB	Leitura	1	HDD	0.043106382	0.214490352
2TB	256KB	*Mixed*	1	HDD	0.012547696	0.061726183
2TB	4MB	*Mixed*	1	HDD	0.051855017	0.258229749
2TB	256KB	Escrita	1	SSD	0.002341591	0.006738864
2TB	4MB	Escrita	1	SSD	0.011225273	0.055088038
2TB	256KB	Leitura	1	SSD	0.001437625	0.005177908
2TB	4MB	Leitura	1	SSD	0.008053261	0.039230848
2TB	256KB	*Mixed*	1	SSD	0.005990096	0.005066902
2TB	4MB	*Mixed*	1	SSD	0.009629171	0.047110279

B

EXPERIMENTO DE ESCRITA - TRATAMENTOS

Tabela B.1: Tratamentos e resultados - experimento de escrita

capacity	*operation*	*node*	*technology*	*object*	IOPS	Tempo Médio de Resposta (s)
2TB	Escrita	3	HDD	256KB	74.1428796	0.0622935
2TB	Escrita	3	SSD	256KB	362.0548908	0.008636495
2TB	Escrita	3	SSDRandom	256KB	486.2502795	0.005182967
2TB	Escrita	3	SSDCache	256KB	115.3748959	0.038169276
2TB	Escrita	3	HDD	512 KB	71.5618526	0.0647275
2TB	Escrita	3	SSD	512 KB	208.4158932	0.018794879
2TB	Escrita	3	SSDRandom	512 KB	328.3706565	0.010041348
2TB	Escrita	3	SSDCache	512 KB	104.2205944	0.042813129
2TB	Escrita	3	HDD	1 MB	53.4828064	0.0883593
2TB	Escrita	3	SSD	1 MB	170.9761491	0.024055495
2TB	Escrita	3	SSDRandom	1 MB	253.2339199	0.014546913
2TB	Escrita	3	SSDCache	1 MB	70.70141255	0.065578438
2TB	Escrita	3	HDD	2 MB	31.0725688	0.1558042
2TB	Escrita	3	SSD	2 MB	153.0467094	0.02748674
2TB	Escrita	3	SSDRandom	2 MB	187.1658195	0.021522159
2TB	Escrita	3	SSDCache	2 MB	42.28118757	0.113136411
2TB	Escrita	3	HDD	4MB	16.40524123	0.299685566
2TB	Escrita	3	SSD	4MB	76.7234657	0.060023509
2TB	Escrita	3	SSDRandom	4MB	103.3621592	0.043212028
2TB	Escrita	3	SSDCache	4MB	31.93500953	0.15145771

C

EXPERIMENTO DE LEITURA - TRATAMENTOS

Tabela C.1: Tratamentos e resultados - experimento de leitura

capacity	*operation*	*node*	*technology*	*object*	IOPS	Tempo Médio de Resposta (s)
2TB	Read	3	HDD	256KB	219.3944713	0.0168735
2TB	Read	3	SSD	256KB	996.15385	0.000206891
2TB	Read	3	SSDRandom	256KB	1001.443079	0.000168372
2TB	Read	3	SSDCache	256KB	957.5719041	0.000414903
2TB	Read	3	HDD	512 KB	192.3298842	0.0200685
2TB	Read	3	SSD	512 KB	977.7847309	0.000313435
2TB	Read	3	SSDRandom	512 KB	994.2897937	0.000219121
2TB	Read	3	SSDCache	512 KB	926.933537	0.000561587
2TB	Read	3	HDD	1 MB	153.2238294	0.0266916
2TB	Read	3	SSD	1 MB	893.6406742	0.000722893
2TB	Read	3	SSDRandom	1 MB	961.5236689	0.000395611
2TB	Read	3	SSDCache	1 MB	850.4370396	0.000945236
2TB	Read	3	HDD	2 MB	108.115121	0.0402996
2TB	Read	3	SSD	2 MB	759.665413	0.001777914
2TB	Read	3	SSDRandom	2 MB	797.0235951	0.001252278
2TB	Read	3	SSDCache	2 MB	756.8412775	0.0017847
2TB	Read	3	HDD	4MB	69.04754096	0.066464362
2TB	Read	3	SSD	4MB	420.0904035	0.00667544
2TB	Read	3	SSDRandom	4MB	455.9057114	0.005316527
2TB	Read	3	SSDCache	4MB	415.4124672	0.006880867

D

EXPERIMENTO *MIXED* - TRATAMENTOS

Tabela D.1: Tratamentos e resultados - experimento *mixed*

capacity	*operation*	*node*	*technology*	*object*	IOPS	Tempo Médio de Resposta (s)
2TB	*Mixed*	3	HDD	256KB	144.8506801	0.0313399
2TB	*Mixed*	3	SSD	256KB	695.1723061	0.004096121
2TB	*Mixed*	3	SSDRandom	256KB	904.2178412	0.002490499
2TB	*Mixed*	3	SSDCache	256KB	229.0486615	0.018691634
2TB	*Mixed*	3	HDD	512 KB	134.4899365	0.0339973
2TB	*Mixed*	3	SSD	512 KB	410.1362101	0.00904831
2TB	*Mixed*	3	SSDRandom	512 KB	634.8420285	0.004765842
2TB	*Mixed*	3	SSDCache	512 KB	206.9979062	0.021018075
2TB	*Mixed*	3	HDD	1 MB	91.4926804	0.0514715
2TB	*Mixed*	3	SSD	1 MB	337.0069638	0.011686793
2TB	*Mixed*	3	SSDRandom	1 MB	494.5499661	0.006975064
2TB	*Mixed*	3	SSDCache	1 MB	140.7108207	0.03240822
2TB	*Mixed*	3	HDD	2 MB	51.1788773	0.0945286
2TB	*Mixed*	3	SSD	2 MB	300.9286112	0.013457611
2TB	*Mixed*	3	SSDRandom	2 MB	366.9692854	0.010472927
2TB	*Mixed*	3	SSDCache	2 MB	84.30183475	0.056203385
2TB	*Mixed*	3	HDD	4MB	27.3873129	0.179408391
2TB	*Mixed*	3	SSD	4MB	151.5898206	0.029821095
2TB	*Mixed*	3	SSDRandom	4MB	203.6424628	0.021388124
2TB	*Mixed*	3	SSDCache	4MB	63.65151693	0.0754401