Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Avaliação de Desempenho de Migração de Contêineres com Redes de Petri Estocásticas

Leonel Feitosa Correia

Picos-PI, Fevereiro de 2024

Leonel Feitosa Correia

Avaliação de Desempenho de Migração de Contêineres com Redes de Petri Estocásticas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Redes de Computadores), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Prof. Dr. Francisco Airton Pereira da Silva

Coorientador: Prof. Dr. Paulo Antônio Leal Rego

Picos-PI

Fevereiro de 2024

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Sistema de Bibliotecas UFPI - SIBi/UFPI
Biblioteca Setorial do CCN

C583a Correia, Leonel Feitosa.
Avaliação de desempenho de migração de contêineres com redes de Petri estocásticas / Leonel Feitosa Correia. -- 2024.
83 f.

Dissertação (Mestrado) - Universidade Federal do Piauí. Centro de Ciências da Natureza. Programa de Pós-Graduação em Ciências da Computação, Picos, 2024.
"Orientador: Prof. Dr. Francisco Airton Pereira da Silva. Coorientador: Prof. Dr. Paulo Antônio Leal Rego."

1. Contêiner. 2. Análise de sensibilidade. 3. Rede Petri. 4. Migração. I. Silva, Francisco Airton Pereira da. II. Rego, Paulo Antonio Leal. III. Titulo.

CDD 004.69

Bibliotecária: Caryne Maria da Silva Gomes - CRB3/1461

Leonel Feitosa Correia

Avaliação de Desempenho de Migração de Contêineres com Redes de Petri Estocásticas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Redes de Computadores), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Prof. Dr. Francisco Airton Pereira da Silva Orientador	**Prof. Dr. Gustavo Rau de Almeida Callou** Membro da Banca
Prof. Dr. Paulo Antônio Leal Rego Coorientador	**Profa. Dra. Juliana Oliveira de Carvalho** Membro da Banca

Prof. Dr. Erico Meneses Leão
Membro da Banca

Picos-PI
Fevereiro de 2024

Resumo

As tecnologias de contêineres estão presentes na maioria dos data centers ao redor do mundo. Esses contêineres operam normalmente em um único *host* controlador, interagindo com um único *kernel*, sendo mais leves que as máquinas virtuais. Para garantir o funcionamento adequado das aplicações distribuídas, é fundamental que os contêineres possam ser realocados de forma simples, mantendo a continuidade do sistema. Por exemplo, o Kubernetes pode encerrar a execução de um contêiner sem estado e, em seguida, iniciar um equivalente a ele em outro *host*. No entanto, a migração de contêineres com estado ainda não é nativamente aplicável ao Kubernetes. A motivação por trás da realização da migração de contêineres é preservar a disponibilidade dos dados contidos em contêineres com estado e otimizar a eficiência operacional dos próprios data centers. Para facilitar essas migrações, existem ferramentas como o *Checkpoint Restoration In Userspace* (CRIU), adaptadas para o processo de migração de contêineres. O problema da pesquisa está no alto custo envolvido ao fazer uma predição do desempenho de migração de contêiner em ambientes reais. Esta dissertação propõe dois modelos de redes de Petri estocásticas (SPN) com e sem estado absorvente. Os modelos avaliam as políticas de migração Cold, PreCopy, PostCopy e Hybrid. O foco desses modelos está nas métricas de Tempo Total de Migração (MTT), Tempo Médio de Migração (MMT), utilização, probabilidade de descarte e taxa de migração. O foco principal é avaliar os fatores que influenciam as métricas, levando em consideração a quantidade de elementos migrados simultaneamente e a capacidade paralela do sistema de migração. O modelo com estado absorvente também permite calcular a função de distribuição de probabilidade acumulada (CDF). Por fim, foi também realizada uma Análise de Sensibilidade (DoE) conduzido para a política Hybrid no modelo absorvente.

Palavras-chaves: contêiner, migração ao vivo, desempenho, rede de Petri, análise de sensibilidade;

Abstract

Container technologies are present in most data centers around the world. These containers typically operate on a single controller host, interacting with a single kernel, and are lighter than virtual machines. To ensure the proper functioning of distributed applications, it is essential that containers can be relocated simply, maintaining system continuity. For example, Kubernetes can terminate the execution of a stateless container and then launch its equivalent on another host. However, stateful container migration is not yet natively applicable to Kubernetes. The motivation behind performing container migration is to preserve the availability of data contained in stateful containers and to optimize the operational efficiency of the data centers themselves. To facilitate these migrations, there are tools such as Checkpoint Restoration In Userspace (CRIU), adapted for the container migration process. The research problem is the high cost involved when making a prediction of container migration performance in real environments. This dissertation proposes two models of stochastic Petri nets (SPN) with and without absorbing state. The models evaluate Cold, PreCopy, PostCopy, and Hybrid migration policies. The focus of these models is on the metrics of Total Migration Time (MTT), Average Migration Time (MMT), utilization, discard probability and migration rate. The main focus is to evaluate the factors that influence the metrics, taking into account the number of elements migrated simultaneously and the parallel capacity of the migration system. The model with absorbing state also allows calculating the cumulative probability distribution function (CDF). Finally, a Sensitivity Analysis (DoE) was also carried out for the Hybrid policy in the absorbent model.

Keywords: container, live migration, performance, petri net, sensitivity analysis.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AR	Realidade Aumentada
AR	Taxa de Chegada
AWS	*Amazon Web Services*
CDF	Função de Distribuição Cumulativa
CRIU	*Checkpoint/Restore In Userspace*
CTMC	Cadeias de Markov de Tempo Contínuo
CPU	Unidade Central de Processamento
DoE	*Design of Experiments*
IoT	Internet das Coisas
KVM	Máquina Virtual do Kernel
LXC	Contêineres Linux
MMT	Tempo Médio de Migração
MR	Taxa de Migração
MRT	Tempo Médio de Resposta
MTT	Tempo Total de Migração
NFV	Virtualização de Funções de Rede
OCI	*Open Container Initiative*
PD	Probabilidade de Descarte
SPN	Redes de Petri Estocásticas
VM	Máquinas Virtuais

Sumário

1 Introdução

A containerização é uma forma de virtualização do sistema operacional que isola os aplicativos de outros processos, permitindo que sejam executados em diferentes ambientes. Atualmente, a tecnologia de contêiner está principalmente associada ao Docker e ao Kubernetes, mas existe desde 2008 (TURNBULL, 2014). Com a crescente adoção da tecnologia de contêineres, , os principais provedores de nuvem pública, como Amazon Web Services (AWS), Microsoft Azure e Google Cloud, reconheceram a necessidade de suportar essa tecnologia. Como resultado, eles agora oferecem suporte para contêineres em suas plataformas para gerenciar eficientemente as cargas de trabalho (STATISTA, 2023). Portanto, os contêineres se tornou uma ferramenta indispensável para qualquer aplicação web moderna. A migração de recursos de *software* é essencial em plataformas de computação virtualizadas de larga escala, como *data centers* (VARASTEH; GOUDARZI, 2015). A migração de contêiner é usada por vários motivos, como manter a proximidade entre os serviços de computação móvel de ponta e os usuários (CONFORTI et al., 2021). A migração é comumente usada por gerentes de *data center* como um facilitador para programar períodos de inatividade do servidor, consolidação de recursos e recuperação de desastres(JUNIOR; MIORANDI; PIERRE, 2020).

Ao considerarmos a interseção entre a versatilidade da tecnologia do contêiner e a eficiência da migração, fica evidente como os contêineres se alinham bem ao processo de realocação de recursos em ambientes dinâmicos e escaláveis. Os contêineres de memória sem estado são particularmente adequados para esse processo de realocação. O Kubernetes, por exemplo, pode desligar um contêiner sem estado e iniciar um equivalente a ele para concluir a realocação entre os nós físicos de maneira eficiente. Isso possibilita a transferência desses contêineres para diferentes servidores conforme a demanda, garantindo a otimização dos recursos e atendendo às necessidades dos usuários de forma dinâmica (BURNS et al., 2022). Até agora, não é aplicável nativamente ao Kubernetes para migrar contêineres com estado, como aqueles com bancos de dados ou servidores de jogos online. Os estados de um contêiner devem ser salvos em volumes e, em seguida, sincronizados com o contêiner replicado(KOTIKALAPUDI, 2017).

O *Checkpoint/Restore In Userspace* (CRIU) é uma solução que complementa as funcionalidades do Kubernates. O CRIU é uma ferramenta de migração responsável por congelar um contêiner em execução (ou um aplicativo individual) e salvar seu estado atual em arquivos. Os dados salvos podem ser usados para restaurar o aplicativo e executá-lo exatamente como estava ao criar o ponto de restauração (TORRE et al., 2019). O CRIU é uma opção para migração de contêineres e possui uma comunidade de desenvolvimento ativa. Esta dissertação se concentra em contêineres computacionais e o CRIU como a

ferramenta de migração devido à sua popularidade e versatilidade. Com o CRIU, é possível congelar uma aplicação em execução, salvar seu estado atual em um ou mais arquivos e posteriormente restaurar os arquivos para retomar a execução a partir do ponto de congelamento anterior, também conhecido como checkpoint (MAHESHWARI et al., 2018). O CRIU oferece diferentes políticas de migração, incluindo Cold, PreCopy, PostCopy e HybridCopy. Cada uma das políticas de migração apresenta uma abordagem distinta para a transferência de contêineres entre nós, considerando, aspectos como o tempo de inatividade do serviço e a eficiência na migração (PICKARTZ et al., 2016).

1.1 Motivação e Casos de Uso Ilustrativos

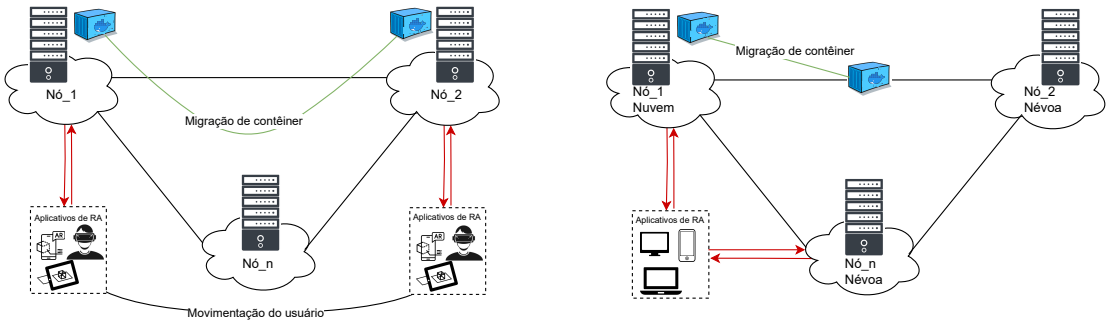
A migração de contêineres entre nós na névoa desempenha um papel crucial em diversas aplicações distintas. Neste contexto, dois casos de uso se destacam: o Caso de Uso de Mobilidade e o Caso de Uso de Orquestração. No Caso de Uso de Mobilidade, a migração de contêineres assegura respostas rápidas e baixa latência ao transferir serviços entre nós de névoa para apoiar a mobilidade de usuários e dispositivos IoT. No Caso de Uso de Orquestração, a migração permite a gestão dinâmica de recursos, migrando serviços entre névoa e Centros de Dados de Borda para otimizar a eficiência. Ambos os casos demandam a migração para manter o desempenho, latência e recursos adequados à medida que as necessidades dos aplicativos evoluem.

Essa necessidade de migração se torna ainda mais evidente quando consideramos a infraestrutura de Internet das Coisas (IoT). Sensores e atuadores podem ser conectados à Internet diretamente ou por uma rede de sensores. De qualquer forma, os sensores e atuadores físicos exploram o conceito de *gateway*, um dispositivo que de um lado se comunica com eles e do outro com a internet e os serviços disponíveis. Esse gateway é fornecido com recursos de computação e armazenamento que permitem a execução do código (PULIAFITO; MINGOZZI; ANASTASI, 2017; JIANG; HUANG; TSANG, 2017).

Em casos de uso específicos, a migração representará uma função obrigatória, necessária para suportar adequadamente a mobilidade de usuários ou dispositivos IoT (TANG et al., 2018). Em particular, a migração será crucial para lidar com cenários em que sensores e atuadores são caracterizados nativamente pela mobilidade, como quando usados ou trazidos por usuários móveis ou transportados por veículos (ZHU et al., 2018). Por exemplo, vamos considerar um veículo de direção autônoma e um serviço de apoio que coleta dados de contexto do ambiente, e outros veículos para notificar possíveis situações perigosas aplicando técnicas de mineração nos dados coletados. Neste caso, a migração do serviço é obrigatória para manter a latência de comunicação entre o veículo e o serviço de suporte o mais baixo possível e permitir que o veículo reaja prontamente. Da mesma forma, em um ambiente industrial, dispositivos Realidade Aumentada (AR) portáteis podem

ser usados pelos operadores para obter informações em tempo real sobre aspectos como: produtos, procedimentos de produção, instruções. Os sistemas AR em tempo real exploram os serviços de computação em névoa para analisar e aumentar imagens com baixa latência limitada e sem a necessidade de descarregá-las na nuvem (FERNÁNDEZ-CARAMÉS et al., 2018). Neste caso, a migração pode ser explorada para suportar funcionalidades AR mesmo quando a mobilidade está envolvida, conforme mostrado.

A Figura 1 apresenta a arquitetura dos casos de uso: o Caso de Uso de Mobilidade e o Caso de Uso de Orquestração. Ambos os casos apresentam a importância da migração para suportar adequadamente a mobilidade e a funcionalidade dos serviços em cenários de uso intensivo de dados e de tempo crítico. Na Figura 1a, é possível observar a arquitetura em um ambiente industrial, onde a migração desempenha um papel crucial. Consideremos um operador movendo-se pelo o ambiente de produção para inspeção equipado com óculos inteligentes que mostram informações de status nos vários equipamentos. Neste caso, a migração do serviço AR é obrigatória para garantir que as imagens sejam sempre recolhidas e analisadas em tempo útil.



(a) Suporte à Mobilidade (b) Orquestração da Plataforma

Figura 1 – Casos de uso de migração de contêineres

A Figura 1b apresenta a arquitetura geral do Caso de Uso Orquestração que mostrando a necessidade de uma gestão eficiente dos recursos em uma infraestrutura multicamadas que integra a nuvem e a névoa. Essa necessidade de migração de serviços é ainda mais evidente quando consideramos uma infraestrutura em névoa que está integrada à infraestrutura de nuvem. Essa infraestrutura multicamadas exigirá suporte para migrar serviços da nuvem para a névoa e vice-versa, com a finalidade de acomodar os requisitos de aplicativos que variam ao longo do tempo (DU et al., 2017).

Por exemplo, um serviço na nuvem, que precisa de interações rápidas com sistemas ciberfísicos, pode ser transferido para um nó de névoa. Da mesma forma, um serviço em um nó de névoa que demanda mais recursos pode ser migrado para a nuvem. Além dessa funcionalidade, a migração de serviços dentro da névoa representará um recurso importante para garantir o bom funcionamento da infraestrutura da névoa. Isso permitirá uma melhor

distribuição de carga, otimização de recursos e maior resiliência em caso de falhas ou alterações na demanda. A migração entre diferentes nós de névoa pode ser explorada para implementar políticas de balanceamento de carga que visam melhorar a utilização de recursos de névoa e garantir que os requisitos do aplicativo sejam constantemente atendidos (PUTHAL et al., 2018). Um serviço de orquestração em infraestrutura de névoa pode redistribuir cargas para evitar degradação do serviço ou economizar energia. O descarregamento de computação adaptável pode reduzir o consumo de energia desses sistemas. Entretanto, em ambos os casos de usos, deparamo-nos com desafios que demandam soluções para melhorar o processo de migração de contêiner. Esses desafios incluem determinar qual política de migração utilizar e a quantidade máxima de elementos a serem migrados de forma simultânea, considerando as limitações da infraestrutura disponível.

1.2 Proposta e Contribuições

Esta dissertação propõe como contribuição geral **uma forma prática de predizer o desempenho quanto à migração de contêineres entre computadores**. Tal predição é feita mediante modelos de redes de Petri estocásticas. De forma mais específica, os dois modelos são descritos sucintamente abaixo.

1. Um modelo SPN **com** estado absorvente validado, proporcionando calcular o Tempo Total de Migração (MTT) para vários contêineres em *batch* e a função de distribuição cumulativa (CDF) que calcula a probabilidade da migração finalizar em determinada janela de tempo. Uma análise de sensibilidade foi realizada sobre este modelo, permitindo examinar diferentes fatores e entender como as variações em seus níveis afetam o desempenho de um sistema de migração de contêiner.

2. Um modelo SPN **sem** estado absorvente capaz de calcular as métricas clássicas de desempenho relacionadas à teoria de filas (com nomes adaptados): Tempo Médio de Migração (MMT), Utilização de Recursos, Probabilidade de Descarte e Taxa de Migração. Estudos de caso são apresentados para servir de guia de uso da proposta em questão. O modelo é bastante adaptável para variar parâmetros de tempo de serviço, taxa de chegada e capacidade paralela de processamento.

1.3 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma. O Capítulo 2 apresenta referencial teórico, com os principais conceitos necessários para a compreensão deste trabalho. O Capítulo 3 apresenta os trabalhos relacionados, comparando-os com nossa proposta. O Capítulo 4 apresenta a metodologia utilizada nesta pesquisa. O Capítulo 5 apresenta o modelo SPN com estado absorvente, validação do modelo proposto, estudo

de caso e análise de sensibilidade. O Capítulo 6 apresenta o modelo SPN sem estado absorvente e seu respectivo estudo de caso. O Capítulo 7 apresenta as principais conclusões obtidas acerca dos estudos realizados e trabalhos futuros. Finalmente, o Capítulo 8 contém as publicações realizadas ao longo do mestrado.

2 Referencial Teórico

Este Capítulo apresenta conceitos essenciais para a compreensão da proposta e trabalhos relacionados. Inicialmente, são apresentadas informações importantes sobre o funcionamento dos contêineres e sua comparação com máquinas virtuais. Em seguida, discutimos sobre Docker e LXC, duas tecnologias amplamente utilizadas para a criação e gerenciamento de contêineres. O Capítulo também aborda as Políticas de Migração. Além disso, é apresentado o *Checkpoint Restore In Userspace* (CRIU), uma ferramenta que permite pausar e retomar a execução de contêineres ou aplicação. Posteriormente, são introduzidos conceitos sobre Redes de Petri Estocásticas. Finalmente, é apresentado a Análise de Sensibilidade usando *Design of Experiments* (DoE), uma metodologia estatística que permite a análise de vários parâmetros de entrada em um sistema.

2.1 Contêiner vs Máquina Virtual

A virtualização baseada em contêiner fornece um nível diferente de abstração em termos de virtualização e isolamento quando comparada com *hipervisores*. Em particular, pode ser considerado uma alternativa leve à virtualização baseada em *hipervisor*. Os *hipervisores* abstraem *hardware*, o que resulta em sobrecarga em termos de virtualização de *hardware* e *drivers* de dispositivos virtuais. Um sistema operacional completo (por exemplo, Linux) é normalmente executado sobre esse *hardware* virtualizado em cada instância de máquina virtual. Por outro lado, os contêineres implementam o isolamento de processos no nível do sistema operacional, evitando assim essa sobrecarga. Esses contêineres são executados no mesmo kernel do sistema operacional compartilhado da máquina *host* subjacente e um ou mais processos podem ser executados em cada contêiner. A arquitetura de virtualização baseada em contêiner é visualizada na Figura 2.

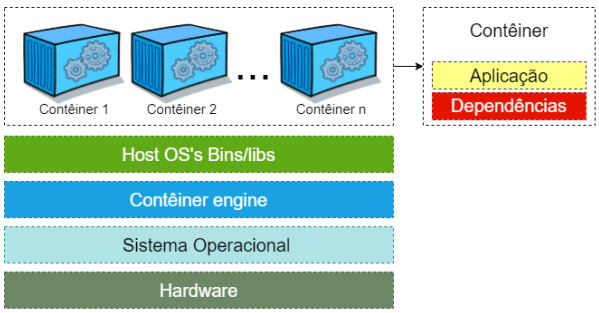


Figura 2 – Componentes principais de uma arquitetura de migração de contêiner.

Devido ao kernel compartilhado (assim como às bibliotecas do sistema operacional), uma vantagem das soluções baseadas em contêiner é que elas podem atingir uma densidade

maior de instâncias virtualizadas e as imagens de disco são menores em comparação com as soluções baseadas em *hipervisor*. A abordagem de kernel compartilhado também tem algumas desvantagens. Uma limitação dos contêineres é que, por exemplo, os contêineres do Windows não podem ser executados em um *host* Linux. Uma outra desvantagem notável é que os contêineres não conseguem isolar os recursos com a mesma eficiência dos *hipervisores* (BUI, 2015). Essa diferença pode ser claramente observada ao comparar as máquinas virtuais (VM) com os contêineres, como ilustrado na Tabela 1.

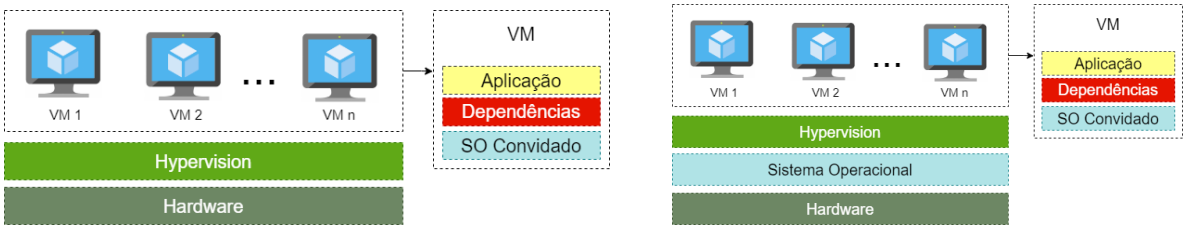
Tabela 1 – Comparação da migração de VM e contêiner (KAUR; GUILLEMIN; SAILHAN, 2022).

Migração de VM	**Migração de contêiner**
As VM são migradas considerando as dependências como pro exemplo: rede, SO, armazenamento.	A migração dos aplicativos implantadas em contêineres.
Dado o grande tamanho da VM, a migração/backup da VM é demorada.	As imagens de contêiner são leves e comparativamente menores em tamanho e são mais rápidas de realocar ou migrar.
A migração de VM inclui a migração/transferência do estado da Unidade Central de Processamento (CPU), conteúdo da memória, conexões/configuração de rede e imagem do disco	A migração de contêiner envolve a transferência de memória, sistema de arquivos e conectividade de rede
Demanda mais tempo que o contêiner.	O tempo de iniciar leva segundos
Cada VM integra seu próprio sistema operacional, que funciona como um servidor virtual e requer um uso significativo de recursos. Porém, menos VMs podem ser implantadas em servidores físicos.	Vários contêineres podem ser implantados em um único sistema operacional, resultando em menos recursos consumidos para implantar e executar contêineres em um servidor físico.
Durante a migração, a transferência de dados leva tempo.	Ao migrar o contêiner, uma quantidade menor de dados é transferido.

Nesta dissertação se concentrou em contêineres no Linux, que são implementados principalmente por meio de grupos de controle (MORABITO; KJÄLLMAN; KOMU, 2015). O primeiro mecanismo fornece gerenciamento de recursos para grupos de processos, e o último fornece uma visão privada e restrita de certos recursos do sistema dentro de um contêiner. Além disso, mecanismos de segurança adicionais podem ser usados para obter um isolamento mais seguro entre contêineres e entre contêineres e o sistema *host*. Esses recursos de nível de kernel mencionados acima são normalmente usados por meio de ferramentas de nível de usuário. Existem várias soluções baseadas em contêineres, por exemplo o OpenVZ , LXC, Docker e Podman.

O Docker, por outro lado, é uma plataforma de alto nível que tornou os contêineres muito populares em um curto espaço de tempo. O Docker combina contêineres Linux

com um sistema de arquivos sobreposto e fornece ferramentas para construir e empacotar aplicativos em ambientes portáteis, ou seja, imagens de contêineres (ANDERSON, 2015). Por outro lado, a virtualização baseada em *hipervisor* tem sido amplamente utilizada durante a última década para implementar virtualização e isolamento. Ao contrário dos contêineres, os *hipervisores* operam no nível do *hardware*, suportando assim máquinas virtuais independentes e isoladas do sistema *host*. Como o *hipervisor* isola a VM do sistema *host* subjacente, é possível executar, por exemplo, VMs baseadas em Windows no Linux. A desvantagem aqui é que um sistema operacional completo é instalado na máquina virtual, o que significa que a imagem será substancialmente maior. Além disso, a emulação do dispositivo de *hardware* virtual incorre em mais sobrecarga (BLENK et al., 2015). Os *hipervisores* são classificados em dois tipos diferentes na Figura 3: *Hipervisores* nativos ou bare-metal e *hipervisores* hospedados.



(a) *Hipervisores* nativos ou bare-metal (b) *Hipervisores* hospedados

Figura 3 – Tipos de *hipervisores*.

No entanto, a distinção entre os dois tipos de *hipervisores* nem sempre é tão clara. Por exemplo, a máquina virtual baseada em kernel do Linux tem características de ambos os tipos. Embora existam várias soluções de *hipervisor* comerciais e de código aberto, focamos aqui no KVM, que é uma solução de virtualização para o Kernel do Linux. O KVM é usado em conjunto com um emulador de *hardware* e virtualizador como o QEMU.

Além de contêineres e *hipervisores*, surgiram várias soluções de virtualização híbrida. Uma dessas soluções é o ZeroVm (MORABITO; KJÄLLMAN; KOMU, 2015), que fornece um ambiente de aplicativo único rodando dentro da sandbox NaCL do Google. Como o ZeroVm opera inteiramente como um processo leve e de espaço de usuário restrito, é fácil iniciar aplicativos ZeroVm para processar, por exemplo, dados armazenados em um servidor de banco de dados. Ou seja, a computação é trazida para os dados, e não vice-versa. Uma limitação do ZeroVm é que é muito desafiador portar o *software* Linux existente para ele porque ele implementa um subconjunto da funcionalidade do Linux.

2.1.1 Docker e LXC

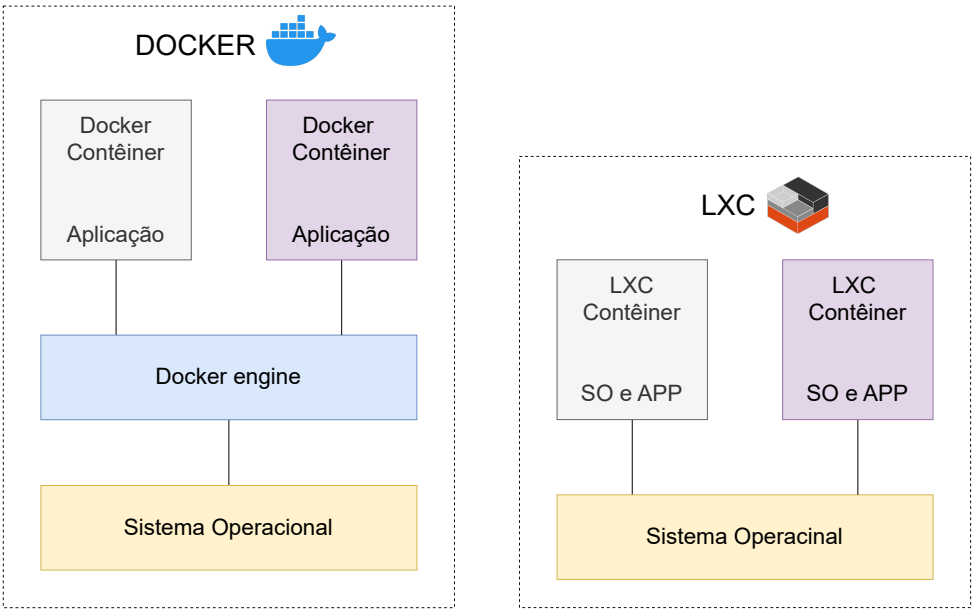
LXC e Docker transformaram a gestão de ambientes virtuais e aplicativos. LXC gira em torno de ambientes virtuais Linux isolados, enquanto o Docker, construído sobre LXC,

concentra-se na contenção de aplicativos, tornando o desenvolvimento e gerenciamento mais simples e portátil em várias plataformas. Essas abordagens alteraram profundamente como lidamos com virtualização e empacotamento de aplicações. O LXC é um sucessor do Open Vz e baseia-se nos princípios estabelecidos neste projeto. É uma tecnologia que fornece virtualização no nível do sistema operacional (BAUKES, 2019). Ele permite que você crie vários ambientes virtuais Linux isolados em uma única máquina convidada. Esses ambientes virtuais podem ser pensados como contêineres que podem ser usados para empacotar aplicativos ou para emular novas máquinas com seu próprio sistema operacional (BERNSTEIN, 2014).

Os ambientes virtuais mencionados são diferentes das máquinas virtuais mais conhecidas. A principal diferença é que não há *software* de emulação em um ambiente virtual como máquinas virtuais. A única emulação que ocorre aqui é a emulação do sistema operacional que roda no contêiner (XAVIER et al., 2013). A emulação de *hardware* não acontece em ambientes virtuais, então criar um ambiente muito mais rápido do que iniciar uma máquina virtual. O lançamento do aplicativo também é quase instantâneo. As mesmas condições que envolvem um início rápido se aplicam mesmo se uma nova máquina for emulada no contêiner com o sistema operacional, porque também é essencialmente um aplicativo empacotado. No caso de máquinas virtuais, outros recursos da máquina, como configurar e emular um disco rígido, processador virtual ou interfaces de rede virtual, são necessários, portanto, demoram muito mais para iniciar do que iniciar um contêiner LXC e também têm CPU mais alta e consumo de RAM da máquina convidada (BERNSTEIN, 2014). Uma pequena desvantagem dos ambientes virtuais é sua implantação e migração reais para o ambiente de produção, esses recursos têm máquinas virtuais melhor resolvidas. Controlar ambientes virtuais por meio de uma interface gráfica também tende a ser bastante complicado (MA; KIM; KIM, 2016; MUKUTE et al., 2019).

O conceito básico do Docker é baseado na tecnologia LXC, mas com o tempo expandiu as propriedades dessa tecnologia e a enriqueceu com novas perspectivas sobre a questão da containerização. Uma das extensões e inovações básicas é o uso de uma Interface de Programação de Aplicação (API) de alto nível que fornece uma melhor solução de virtualização, de modo que, nesse caso, processos individuais sejam executados isoladamente em vez de sistemas operacionais inteiros. Portanto, os contêineres do Docker não possuem seu próprio sistema operacional. Em vez disso, eles dependem do sistema operacional convidado e de seus recursos, bem como da infraestrutura geral da máquina convidada. O Docker pode assim ser caracterizado como uma plataforma de contêineres com boa portabilidade, cuja principal tarefa é empacotar a aplicação e todos os seus componentes, dos quais depende de um contêiner virtual. Esse contêiner pode ser executado em qualquer servidor Linux. Com a tecnologia LXC, o Docker compartilha cgroups do Linux, namespaces e o kernel do Linux. No entanto, os contêineres Docker são muito mais avançados do que os contêineres LXC.

A principal característica do Docker é a sua abordagem de abandonar a tradicional virtualização e isolamento de todo o sistema operacional, como foi feito pela tecnologia LXC. Em vez disso, o Docker começou a se concentrar mais nos aplicativos, nos próprios serviços e em seu isolamento (BAUKES, 2019). Embora o Docker tenha sido originalmente desenvolvido sobre o LXC, com o tempo, o próprio ambiente do Docker, também conhecido como *libcontainer*, evoluiu. Ao contrário do LXC apresentado na Figura 4b, onde cada contêiner contém seu próprio sistema operacional, o Docker fornece um ambiente que consiste em apenas um sistema operacional convidado como apresentado na Figura 4a, no qual todos os aplicativos são executados compactados em contêineres, e cada aplicativo possui seu próprio ambiente isolado (ANDERSON, 2015). Esse ambiente é geralmente especificado e criado usando a imagem do Docker. A criação desse ambiente pode ser automatizada usando o chamado Dockerfile.



(a) Contêiner Docker (b) Contêiner LXC

Figura 4 – Comparação entre os contêiner Docker e os contêiner LXC.

Outro recurso importante que diferencia o Docker da tecnologia LXC é o número limitado de aplicativos que podem ser empacotados em um contêiner. O Docker surgiu com o conceito de empacotar apenas um único aplicativo em um contêiner, ao contrário do LXC, onde vários aplicativos ou todo o sistema operacional são embalados em um contêiner (BAUKES, 2019). A introdução dessa inovação trouxe menos dificuldades para os desenvolvedores.

2.1.2 Políticas de Migração

Plataformas para gerenciar cargas de trabalho e serviços conteinerizados, como Kubernetes, podem utilizar o runC [1] para gerar e executar contêineres. O runC é um *software* de execução de contêiner leve e portátil, iniciado pela *Open Container Initiative* (OCI) e utilizado por muitas soluções de virtualização baseadas em contêiner, como Docker e Containerd. Para implementar a migração de contêiner em sistemas operacionais Linux, o runC recorre ao *Checkpoint Restoration In Userspace* (CRIU)[2], que é um componente de *software* que oferece funcionalidade de pontos de verificação/restauração para aplicativo Linux . O CRIU apresenta vários métodos para migração de contêiner, incluindo Cold, PreCopy, PostCopy e Hybrid. Nesta dissertação modelamos os quatro métodos com redes de Petri estocásticas, o Cold (Figura 5), PreCopy (Figura 6), PostCopy (Figura 7) e Hybrid (Figura 8) visando avaliar o desempenho no processo de migração.

Arquitetura da Política de Migração Cold - A Figura 5 mostra as diferentes fases da migração do contêiner com base na política de migração Cold: pontos de verificação, transferência e restauração. Durante o checkpoint, o CRIU congela o contêiner em execução no nó de origem e coleta metadados sobre o estado da CPU, conteúdo da memória e informações sobre a árvore de processos associada ao serviço de contêiner em execução. Durante a fase de transferência, as informações de metadados coletadas são transferidas para o nó de destino. A fase de restauração retoma o serviço de contêiner do ponto congelado com os metadados transferidos no nó de destino (DESHPANDE; LIU, 2017; SINDI; WILLIAMS, 2019). O tempo de migração do contêiner pode ser expresso como:

$$T_{cm} = T_sc + T_t + T_r \tag{2.1}$$

onde:

- T_{cm} é o tempo total de migração do contêiner,
- T_{sc} é o tempo do ponto de verificação,
- T_t é o tempo de transferência de metadados da origem para o nó de destino e
- T_r é o tempo de restauração.

Para a política de migração Cold, T_{cm} também corresponde ao tempo de inatividade do serviço, pois o serviço de contêiner em execução está interrompido no início da fase de checkpoint, e o serviço é restaurado somente após a restauração bem sucedida.

[1] runC: <https://www.docker.com/blog/runc>
[2] CRIU: <https://criu.org/Main_Page>

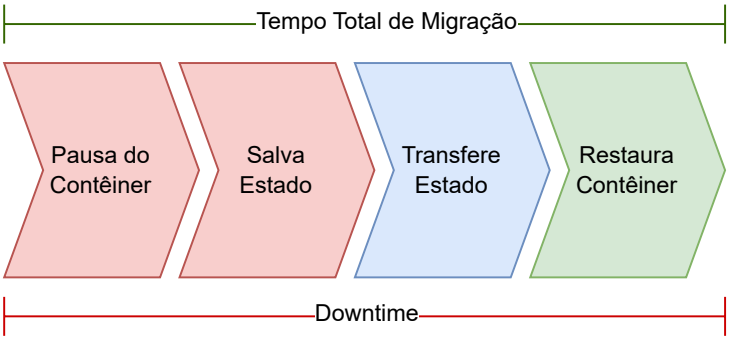


Figura 5 – Método de migração Cold.

Arquitetura da Política de Migração PreCopy - O método mostrado na Figura 6 é a segunda abordagem migratória, consistindo em fases de Salva Estado Pre-Dump, Transfere PreDump, Pausa do Contêiner e Salva Modificações, Transfere Estado e Restauração do contêiner. O ponto de verificação do contêiner ou aplicativo no nó de origem, acontece no PreDump e Salva Modificações. A fase de PreDump coleta todas as informações de estado e memória do contêiner, e a fase de Salva Modificações coleta apenas as informações das páginas de memória modificadas. Na fase de transferência, os metadados coletados são transferidos para o nó de destino, e na fase de restauração o serviço de aplicativo é restaurado (NIE et al., 2017).

O tempo de migração do contêiner na abordagem PreCopy pode ser expresso como:

$$T_{prm} = T_{pd} + T_{pdt} + T_d + T_{dt} + T_r \tag{2.2}$$

onde:

- T_{prm} é a migração de contêiner total baseada em PreCopy,
- T_{pd} é o tempo de PreDump,
- T_{pdt} é o tempo de transferência de dados PreDump,
- T_d é o tempo de Dump,
- T_{dt} é o tempo de transferência de dados de Dump, e
- T_r é o tempo de restauração.

A vantagem do método PreCopy é que o serviço permanece responsivo durante a fase de PreDump de coleta do ID do processo, das páginas de memória e do estado de execução. Para a política de migração PreCopy.

Arquitetura da Política de Migração PostCopy - O método mostrado na Figura 7 é terceira abordagem migratória apresentada, consistindo em fases de Pausa

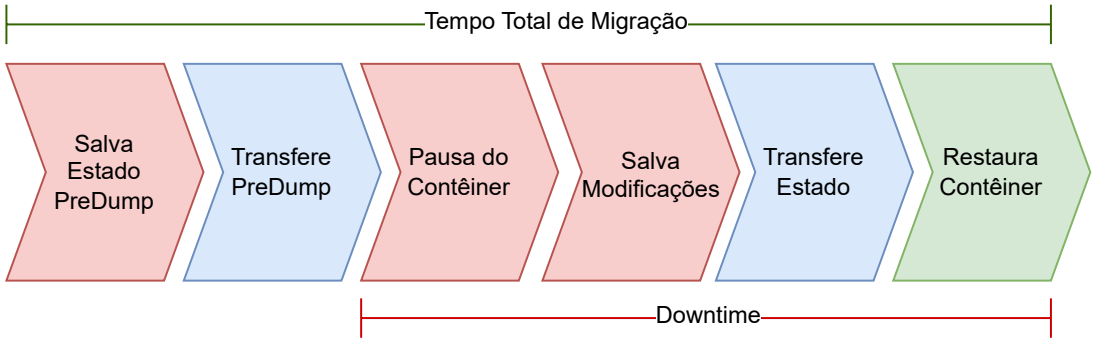


Figura 6 – Método de migração PreCopy.

Contêiner, Salva Estado, Transfere Estado, Restorauração do Contêiner e Transferência de páginas solicitadas. A abordagem PostCopy de migração é o inverso da estratégia PreCopy de migração. Inicialmente, ela suspende a operação do contêiner no nó de partida e replica o estado de execução para o nó de destino, possibilitando que o contêiner prossiga com sua execução no novo ambiente. Somente após essa etapa é que ocorre a transferência das demais informações, ou seja, todas as páginas de memória associadas. Neste estudo, apresentamos a variante de migração PostCopy que se baseia na estratégia de paginação por demanda, popularmente conhecida como migração *lazy* (PULIAFITO et al., 2019). Com a migração *lazy*, o contêiner retomado tenta acessar as páginas de memória no nó de destino, ao não encontrar essas páginas, são gerados erros de página. Isso leva o Daemon de páginas em modo *lazy* do nó de destino a entrar em comunicação com o servidor de páginas localizado no nó de partida. Esse servidor, apenas mediante solicitação, encaminha as páginas com falhas para o nó de destino.

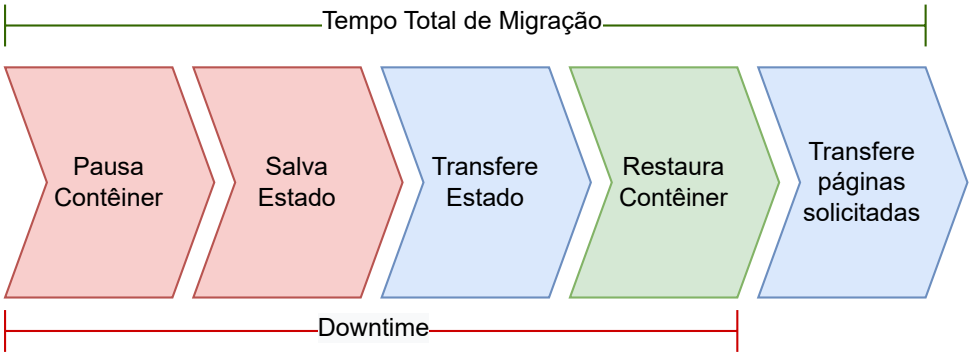


Figura 7 – Método de migração PostCopy.

O tempo de migração do contêiner na abordagem PostCopy pode ser expresso como:

$$T_{ptm} = T_{sc} + T_{td} + T_r + T_{tfp} \tag{2.3}$$

onde:

- T_{ptm} é a migração de contêiner total baseada em PostCopy,
- T_{sc} é o tempo do ponto de verificação,
- T_{td} é o tempo de transferência de dados do dump,
- T_r é o tempo de restauração e
- T_{tfp} é o tempo de transferência das páginas com falha.

A política de migração PostCopy é realiza a cópia de cada página de memória apenas uma vez, tornando o volume de dados comparável à migração e à fase PreDump da migração PreCopy.

Arquitetura da Política de Migração Hybrid - O método mostrado na Figura 8 é a quarta abordagem migratória de contêiner, consistindo em fases de: PreDump, Transfere PreDump, Pausa Contêiner, Salva Modificações, Transfere Estado, Restauração do Contêiner e Transferência de Páginas Solicitadas. A migração híbrida, combina PreCopy e PostCopy para suprir as limitações e aprimorar seus pontos fortes. As duas primeiras etapas da migração híbrida coincidem com as da migração PreCopy, um PreDump de todo o estado e sua transmissão no destino enquanto o contêiner continua em execução no nó de origem. Em seguida, o contêiner é interrompido e seu estado é despejado de uma forma que combina as políticas PreCopy e PostCopy. O Salva Modificações na migração híbrida é representado pelas modificações no estado de execução que ocorreram durante a fase de PreCopy. Depois que os arquivos modificados juntos com os coletados na etapa PreDump é transferido para o nó de destino, o contêiner pode ser restaurado. Nesta etapa, o nó de destino tem o estado de execução do contêiner atualizado. No entanto, algumas páginas de memória ficaram sujas durante a fase de PreCopy. A última etapa da política de migração híbrida consiste no servidor de páginas na origem transmitindo lentamente as páginas sujas para o Daemon de páginas *lazy* no nó de destino (PULIAFITO et al., 2021).

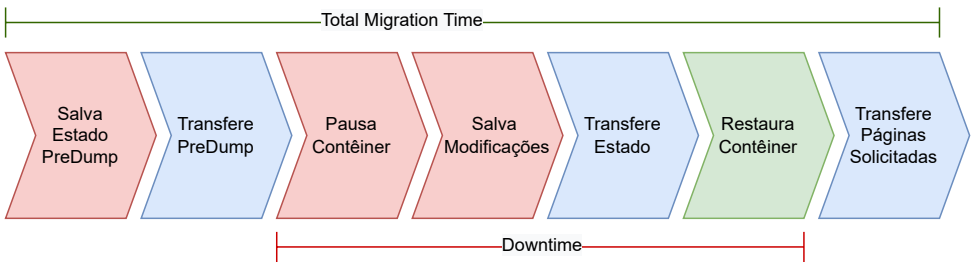


Figura 8 – Método de migração Hybrid.

O tempo de migração do contêiner na abordagem Hybrid pode ser expresso como:

$$T_{hcm} = T_{pd} + T_{pdt} + T_d + T_{dt} + T_r + T_{tfp} \tag{2.4}$$

onde:

- T_{hcm} é a migração de contêiner total baseada em Hybrid,
- T_{pd} é o tempo de PreDump,
- T_{pdt} é o tempo de transferência de dados PreDump,
- T_d é o tempo de Dump,
- T_{dt} é o tempo de transferência de dados de Dump,
- T_r é o tempo de restauração, e
- T_{tfp} é o tempo de transferência das páginas com falha.

2.2 CRIU - Checkpoint Restore In Userspace

CRIU é uma ferramenta poderosa que possibilita a migração de aplicativos de um *host* para outro, enquanto preserva os seus estados. Esta ferramenta tem um potencial significativo para uso em sistemas de computação de borda, onde a migração de aplicativos em execução pode ser necessária devido a restrições de recursos ou outros fatores. A capacidade de migrar uma aplicação em execução preservando seu estado, pode trazer melhorias significativas em termos de confiabilidade, disponibilidade e tolerância a falha(TOŠIĆ, 2023). Tecnologias que suportam o CRIU incluem Docker, LXC, Podman e rkt. Alguns casos de uso em potencial pertencentes ao caso de uso mencionado acima são:

1. **Tolerância a Falhas e Recuperação de Desastres:** CRIU pode ser usado para fornecer tolerância a falhas e recursos de recuperação de desastres em sistemas de computação de ponta. Ao migrar em execução aplicativos para um *host* diferente em caso de falha ou desastre, a CRIU pode ajudar garantir que os serviços críticos permaneçam disponíveis e reduzir o impacto do tempo de inatividade.

2. **Mobilidade:** O CRIU pode ser usado para fornecer recursos de mobilidade em sistemas de computação de ponta. Ao permitir que aplicativos em execução sejam migrados entre diferentes *hosts*, o CRIU pode permitem novos casos de uso, como computação de borda móvel, em que os serviços seguem os usuários conforme eles mover-se.

3. **Dimensionamento dinâmico:** O CRIU pode ser usado para permitir o dimensionamento dinâmico de serviços na borda sistemas de computação. Ao migrar aplicativos em execução entre *hosts* com base nas alterações sob demanda, a CRIU pode ajudar a garantir que o sistema permaneça responsivo e que os recursos sejam usados de forma eficiente.

4. **Eficiência Energética:** CRIU pode ser usado para habilitar a computação com eficiência energética na borda sistemas de computação. Ao migrar aplicativos em execução para *hosts* atualmente ociosos, A CRIU pode ajudar a reduzir o consumo geral de energia e melhorar a sustentabilidade do sistema.

Um dos principais pontos de interesse é o advento de protocolos descentralizados para gerenciamento e orquestração de contêineres. Protocolos como Caravela (PIRES; SIMÃO; VEIGA, 2021) e Nion Network (TOŠIĆ et al., 2023) propuseram uma rede descentralizada de nós que realizam migrações de contêineres em um esforço para equilibrar a utilização de recursos em toda a rede. A rede Nion protege a decisão fazendo o processo através de um mecanismo de consenso de blockchain eficiente e escalável. A adição de novos blocos à cadeia pode ser considerada como a transição de estado, enquanto a função de transição de estado são os nós que executam migrações planejadas de contêineres formulários. Claramente, essas redes estão sujeitas ao comportamento bizantino e, embora os mecanismos de consenso que os blockchains normalmente usam tolerem essas falhas, os aplicativos não essas falhas são tradicionalmente resolvidas com a introdução de redundância sempre que possível e/ou backup. Nesso caso de uso, a redundância exigiria mais nós para executar o aplicativo em contêiner e servir como backups prontos. Evidentemente, isso aumenta a preocupação com a eficiência de toda a rede devido à utilização adicional de recursos de computação. Por outro lado, os backups apenas impõem requisitos adicionais de armazenamento, mas devem ser frequentes para evitar estados de perda de perda. O CRIU pode ser usado para capturar aplicativos regularmente e enviar esses instantâneos para outros nós. Em caso de falha na máquina *host* (nó), o protocolo pode detectá-la e restaurar o aplicativo em outro nó.

2.3 Redes de Petri Estocásticas

As Redes de Petri (PNs) são uma ferramenta de modelagem gráfica e matemática aplicável a muitos sistemas. São uma ferramenta para descrever e estudar sistemas de processamento de informações que são caracterizados como sendo concorrente, assíncrono, distribuído, paralelo, não determinístico e/ou estocástico. Como uma ferramenta gráfica, PNs podem ser usa como um auxílio de comunicação visual semelhante ao fluxo gráficos, diagramas de blocos e redes. Além disso, *tokens* são usados nessas redes para simular as atividades dinâmicas e concorrentes de sistemas. Como uma ferramenta matemática, é

possível configurar equações de estado, equações algébricas e outros modelos matemáticos que regem o comportamento de sistemas. Desde o trabalho seminal de Petri, muitas representações e extensões foram propostas permitindo descrições mais concisas e representando características de sistemas não observadas no início modelos (MURATA, 1989).

Redes de Petri Estocásticas (SPNs) são casos especiais de PNs. Modelos SPN foram propostos com o objetivo de desenvolver uma ferramenta que permitiu a integração de descrição formal, prova de correção e avaliação de desempenho. As propostas de avaliação de desempenho visavam uma equivalência entre SPN e Cadeias de Markov de Tempo Contínuo (CTMC) (GERMAN, 2000). A fim de obter uma equivalência entre um PN e um CTMC, foi necessário introduzir especificações temporais de modo que a evolução futura do modelo, dada a marcação atual, seja independente da marcando a história. Portanto, os SPNs podem ser traduzidos para CTMC, que podem então ser resolvidos para alcançar os resultados desejados de desempenho ou confiabilidade (MOLLOY, 1982; MARSAN et al., 1998; TRIVEDI, 2008; MARSAN, 1990)

A Figura 9 exibe os componentes usados para modelar um SPN e a Figura 10 apresenta dois exemplos de SPNs, uma sem estado absorvente e outra com estado absorvente. Os lugares são representados por círculos, enquanto as transições são representadas como retângulos vazios (transições temporizadas). As transições determinísticas têm um tempo de disparo fixo e as transições imediatas não têm tempo associado. Arcos (arestas) conectam lugares a transições e vice-versa. *Tokens* (pequenos círculos preenchidos) podem residir em lugares, que denotam o estado (ou seja, marcação) de SPN. Um arco inibidor é um arco especial que representa um pequeno círculo branco em uma borda, em vez de uma seta, e são geralmente usados para desative as transições se houver *tokens* presentes em um Lugar.

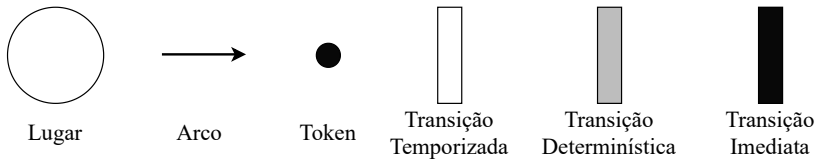


Figura 9 – Componentes principais de uma SPN.

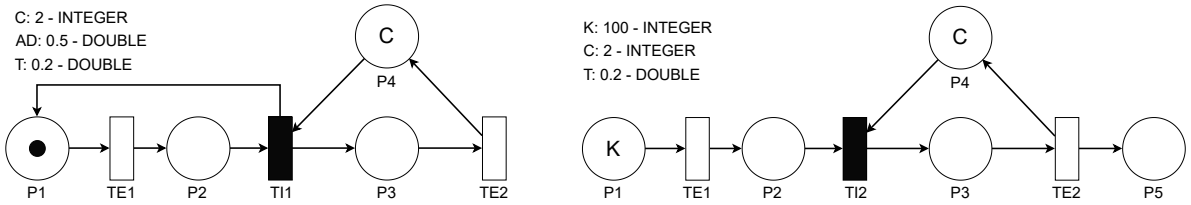
A Figura 10a apresenta o modelo SPN de desempenho que não possui estado absorvente. Isso significa que a movimentação dos *tokens* nunca termina, tende ao infinito, pois novos *tokens* sempre são gerados e a capacidade é sempre reposta. Para descrever as métricas é necessário detalhar como representamos probabilidade e esperança. Calculamos a probabilidade de ter 0 *tokens* no lugar P4 da seguinte maneira P#{P4=0}. Esperança matemática significa a soma de probabilidades de diferentes possibilidades. Tal esperança nos retorna o número de *tokens* em um determinado lugar de maneira estacionária, ou seja, tendendo ao infinito qual valor terá aquele lugar geralmente (MACIEL et al., 2017).

Em relação ao modelo, primeiramente é importante notar os três parâmetros criados como variáveis. O AD é o tempo entre chegadas atribuído a TE1. O C é a capacidade de processamento paralelo atribuído a P3. O tempo T é o tempo de processamento de um token atribuído à transição TE2.

Quanto às métricas temos as seguintes:

- MRT = E#{P1}*AD
- U = (E#{P1}/C)*100
- DP = (P#{P0=0})*100
- TP = E#{P1}/T

O modelo da Figura 10b é do tipo absorvente, ou seja, tendendo ao infinito, a movimentação dos *tokens* sessará. Existem neste caso, 100 *tokens* (requisições) a serem processadas, quando todas passarem por TE1, o modelo não terá mais movimentação. Para modelos SPN de desempenho absorvente calculamos basicamente dois itens, o tempo médio de absorção (MTTA) e a probabilidade de absorção em função do tempo através da função de *cumulative distribution function* (CDF). O MTTA, que para este caso significa quanto tempo leva para processar os 100 *tokens* da entrada (BAUSE; KRITZINGER, 2002).



(a) Modelo SPN de desempenho não absorvente.

(b) Modelo SPN absorvente.

Figura 10 – Exemplos de SPNs

O comportamento de SPN é definida em termos de um fluxo de token. Os *tokens* são criados e destruídos de acordo com os disparos de transição (GERMAN, 2000). As transições imediatas representam atividades instantâneas e têm maior prioridade de disparo do que transições temporizadas. Tais transições também podem conter uma condição de guarda e uma o usuário pode especificar uma prioridade de disparo diferente entre outras transições imediatas. Há também funções de guarda em SPNs. As funções de guarda são expressões booleanas que controlam o disparo de uma transição, declarando alguma condição em relação à marcação da rede. Se a função de guarda de uma transição produz um valor verdadeiro, é capaz de disparar, caso contrário, a transição é desabilitada (MARSAN et al., 1998).

2.4 Análise de Sensibilidade usando DoE

A análise de sensibilidade é uma medida do efeito de um determinado dado de entrada sobre os dados de saída, visando os componentes mais relevantes do sistema, e a partir daí, buscar adotar um conjunto de técnicas que visam melhorar esses sistemas em diferentes cenários (CAMPOLONGO; TARANTOLA; SALTELLI, 1999). Neste trabalho, aplicamos uma análise de sensibilidade com DoE.

O *Design of Experiments* (DoE) corresponde a um conjunto de técnicas estatísticas que aprofundam o conhecimento sobre o produto ou processo em estudo (KLEIJNEN, 1995). Também pode ser definido por uma série de testes nos quais o pesquisador altera o conjunto de variáveis ou fatores de entrada para observar e identificar os motivos das mudanças na resposta de saída.

Os projetistas de sistemas costumam adotar análises de sensibilidade para avaliar o quão "sensível" uma métrica é a mudanças no modelo (SANTOS et al., 2021b). Os parâmetros a serem alterados são definidos por meio de um plano de experimento. O objetivo é gerar a quantidade mais significativa de informações com o mínimo de experimentos possíveis. O comportamento do sistema com base nas mudanças de parâmetro pode ser observado usando conjuntos de saídas. Na literatura, existem três categorias de gráficos geralmente adotados para experimentos com DoE.

O gráfico de Pareto permite detectar qual o efeito da interação de fatores é mais importante para o processo ou estudo de otimização de projeto. Exibe os valores absolutos dos efeitos e desenha uma linha de referência. Qualquer efeito que ultrapasse essa linha de referência é potencialmente importante. Um gráfico de Pareto é construído como na Figura 11, por exemplo. O gráfico mostra que os fatores B (*tool geometry*) e C (*cutting angle*) e a interação AC possuem maior impacto.

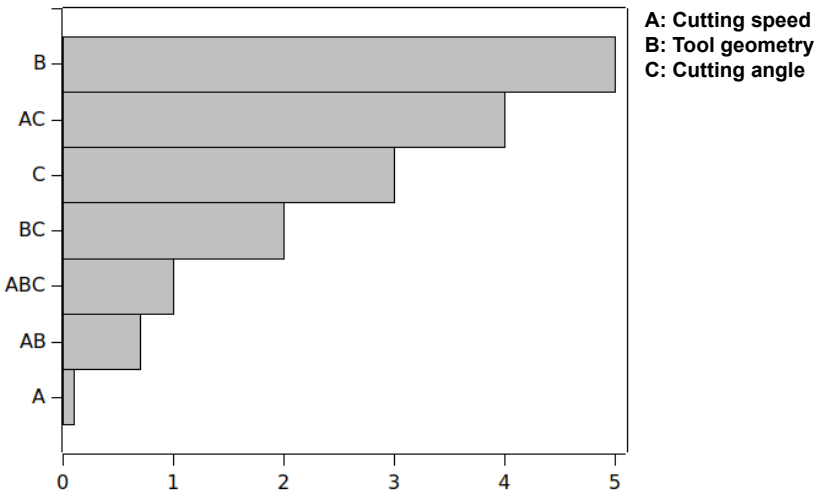


Figura 11 – Exemplo de gráfico de Pareto com efeitos padronizados.

O processo de "Interações", identifica efeitos importantes e determina sua magnitude, logo, as interações entre os efeitos são cruciais. As interações ocorrem quando o efeito de um fator depende do nível de outro fator. Uma medida de *design* sempre aborda vários fatores. Entender como esses fatores interagem em que magnitude permite para escolher a melhor combinação de medidas, revelando combinações de fatores com efeito cumulativo ou degradante. A interação entre os fatores A e B pode ser calculada usando a Equação 2.5.

$$I_{A,B} = \frac{1}{2}(E_{A,B(+1)} - E_{A,B(-1)}) \tag{2.5}$$

O $E_{A,B(+1)}$ é o efeito do fator A no nível alto do fator B e $E_{A,B(-1)}$ é o efeito do fator A no nível baixo do fator B.

Para determinar se dois parâmetros de processo estão interagindo ou não, pode-se usar uma ferramenta gráfica simples, porém poderosa, chamada de gráficos de interação. Se as linhas no gráfico de interação forem paralelas, não haverá interação entre os parâmetros do processo. Isso implica que a mudança na resposta média do fator A não depende dos níveis do fator B. Por outro lado, se as linhas não são paralelas, existe uma interação entre os fatores. Quanto maior o grau de afastamento de ser paralelo, mais forte o efeito de interação. Para interação sinérgica, as linhas no gráfico não se cruzam. Por exemplo, a Figura 12a é um exemplo de interação sinérgica.

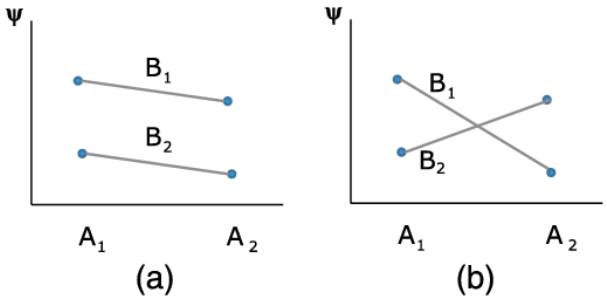


Figura 12 – Gráficos de interações sinérgica e antagônica.

Na interação antagônica, as linhas no gráfico se cruzam. Isso pode ser ilustrado na Figura 12b. Neste caso, a mudança na resposta média para o fator A no nível A_1 é alta em comparação ao nível A_2. As mudanças nos níveis do fator A para a resposta média, indica uma dependência do fator A em relação aos níveis do fator B.

2.5 Conclusão do Capítulo

Este Capítulo forneceu uma visão abrangente dos conceitos fundamentais e das técnicas utilizadas na proposta de pesquisa. Discutiu-se a virtualização baseada em contêineres, destacando suas vantagens em relação à virtualização baseada em *hipervisores*.

Foram apresentadas as políticas de migração Cold, PreCopy, PostCopy e Hybrid, juntamente com a ferramenta CRIU, que desempenha um papel importante na migração de contêineres. Além disso, foram discutidos os conceitos de Redes de Petri Estocásticas, que é uma ferramenta valiosa para modelar sistemas complexos.

Por fim, a análise de sensibilidade usando DoE foi introduzida como uma técnica eficaz para identificar os fatores relevante do sistema. A compreensão desses conceitos e técnicas é essencial para a implementação e análise do modelo proposto. Após a exploração abrangente dos conceitos fundamentais e técnicas, o próximo Capítulo se concentra em apresentar os trabalhos relacionados a este estudo.

3 Trabalhos Relacionados

Este Capítulo apresenta os trabalhos relacionados com abordagens ou contextos a este estudo. A seleção dos artigos considerou critérios como a motivação, as políticas de migração, as métricas, os métodos de avaliação e a ferramenta empregada. Os trabalhos relacionados foram agrupados com base nas políticas de migração, uma vez que, ao analisar a natureza da aplicação conteinerizada é essencial considerar a previsão do tempo total de migração, o número de contêineres descartados e o tempo de inatividade A Tabela 2 sintetiza as contribuições dos trabalhos mais proeminentes relacionados a esta dissertação.

3.1 Trabalhos que Utilizam a Política **PreCopy**

A política de migração PreCopy é foco de muitas pesquisas devido à sua alta performance em migrações ao vivo. Além disso, a política PreCopy é conhecida por contribuir para a redução do tempo de inatividade e oferecer previsibilidade, o que é de grande interesse em ambientes que priorizam a disponibilidade contínua dos serviços. Xu et al. (2020) apresenta um sistema de migração ao vivo chamado Sledge. O sistema é projetado para assegurar que os componentes permaneçam íntegros durante o processo de migração, integrando tanto as imagens quanto o contexto de gerenciamento.

Benjaponpitak, Karakate e Sripanidkulchai (2020) desenvolveu o CloudHopper, capaz de realizar a migração ao vivo de contêineres entre múltiplos provedores de nuvem. Fan et al. (2019) apresenta um modelo de migração ao vivo de localidade para o *docker container*, incorporando a política de migração PreCopy. Eles consideram fatores como a distância entre os contêineres, a largura de banda disponível e os custos associados. Smimite e Afdel (2019) aborda a migração ao vivo de uma aplicação monolítica, mais especificamente o streaming de vídeo instalado em um contêiner aninhado em uma máquina virtual. Os autores exploraram as vantagens da virtualização híbrida para melhorar o desempenho da migração, analisando aspectos como a utilização do processador, o uso do disco, o tempo total de migração e o tempo de inatividade.

Al-Dhuraibi et al. (2017) propõe o ELASTICDOCKER, um sistema que possibilita a elasticidade vertical de contêineres Docker de forma autônoma. Como a elasticidade vertical é limitada à capacidade da máquina hospedeira, o sistema realiza a migração ao vivo dos contêineres quando não há recursos suficientes na máquina de hospedagem. Os autores Bhardwaj e Krishna (2022) desenvolveram uma configuração experimental para implementar e avaliar o desempenho da técnica de migração baseada em contêiner LXD/CR e compará-la com o esquema de migração de VM PreCopy. Ramanathan et al. (2021b) fornece uma análise baseada em experimentos da migração ao vivo em duas

tecnologias de virtualização, VM e contêiner, com um foco detalhado na abordagem de migração em contêineres. O objetivo é realizar a migração de contêineres para atender às demandas da Virtualização de Funções de Rede (NFV).

3.2 Trabalhos que Utilizam **Múltiplas Políticas**

A escolha de múltiplas políticas de migração decorre do contexto específico dos trabalhos, nos quais uma das propostas é analisar o comportamento das migrações de contêineres e entender como diferentes situações podem afetar a migração e, possivelmente, identificar qual política é mais adequada para cada contexto. Os autores Stoyanov e Kollingbaum (2018) utilizaram o recurso "cache/proxy" de imagem do CRIU para a migração ao vivo de contêineres. O objetivo é melhorar o tempo total de migração e o tempo de inatividade das aplicações de contêineres, evitando o uso de armazenamento secundário, que pode resultar em atrasos significativos devido a modificações rápidas de memória durante a transferência. Govindaraj e Artemenko (2018) apresentam um esquema inovador de migração de contêineres ao vivo, intitulada "migração de redundância", utilizando a ferramenta CRIU. O objetivo é minimizar o tempo de inatividade em comparação com a migração tradicional em contêineres Linux.

Chou et al. (2019) apresentou uma abordagem de migração baseada em checkpoints, utilizando um pool global de memória não volátil compartilhada. Os autores implementam o protótipo no CRIU e avaliam o tempo de inatividade, MTT e envelhecimento da aplicação durante o processo de migração. Ramanathan et al. (2021a) propõe um *framework* para migrar componentes virtuais EPC em contêineres usando migração ao vivo, com análise experimental comparando VMs e contêineres, visando melhorar eficiência e reduzir interrupções de serviço. A análise considera diversos fatores do sistema, incluindo o tipo das instâncias de modelo de arquitetura de hardware do processador e taxa de dados da interface de rede. Pecholt, Huber e Wessel (2021) apresentam requisitos para um sistema, propondo um conceito de instância e migração de contêineres OS ao vivo. O sistema assegura integridade e confidencialidade durante implantação, migração e repouso. Além disso, os autores utilizam VMs criptografadas para remover componentes do TCB, mantendo a migração transparente e a execução contínua dos contêineres.

3.3 Trabalhos que não Explicaram a Política Utilizada

Ma et al. (2018) introduz uma plataforma de borda para a migração contínua de serviços móveis, assegurando a conexão com o servidor mais próximo. Os pesquisadores enfrentam o desafio da migração de contêineres Docker e propõem a aplicação da estrutura de armazenamento em camadas para minimizar a sobrecarga, dispensando a dependência de sistemas distribuídos complexos. Di, Shao e Tan (2021) propõe uma ferramenta para a

migração em tempo real de contêineres Docker entre diferentes nós, mesmo em *workflows* que contenham diversos contêineres. O método de migração da ferramenta otimiza o início de contêineres a partir de pontos de verificação e propõe uma estratégia de migração de múltiplos contêineres, reduzindo o tempo total de migração em comparação com a migração sequencial.

Tay, Gaurav e Karkun (2017) propõe fornecer um quadro matemático para a tomada de decisão ao colocar e migrar cargas de trabalho em um data center, onde as aplicações são empacotadas como contêineres de sistema operacional em execução em máquinas virtuais. Os autores comparam o desempenho de contêineres e máquinas virtuais em diferentes cenários de migração de carga de trabalho. González e Arzuaga (2020) apresenta o Herd-Monitor, um sistema de monitoramento de recursos para coletar métricas de desempenho de contêineres e seus respectivos hospedeiros de computação. Abdullah, Hadeed et al. (2022) propõe um algoritmo para gerenciar a execução de contêineres. O algoritmo considera um conjunto de restrições ao migrar contêineres entre nós, como disponibilidade de recursos, prazo limite e localização dos nós. Quando um evento ocorre, o contêiner é migrado de um nó, para o nó mais próximo ou mais adequado possível. Machen et al. (2017) apresenta um *framework* em camadas para a migração de aplicações de serviço ativas que estão encapsuladas em VMs ou contêineres. A abordagem em camadas utilizada reduz significativamente o tempo durante o qual o serviço fica fora do ar.

Das e Sidhanta (2023) propõem um algoritmo leve para a execução de migração ao vivo de contêineres em clusters de borda com recursos limitados, abrangendo cenários que envolvem partições de rede e falhas de dispositivos. Majeed et al. (2020) propõe modelos de predição que possam prever o tempo de inatividade de sistemas com base em informações métricas coletadas da infraestrutura subjacente e das áreas circundantes. A previsão seria extremamente útil para automatizar o processo de migração de tarefas usando contêineres na computação em borda. Kakakhel et al. (2018) examina a migração ao vivo de aplicações com estado por meio de contêineres. Os experimentos realizados pelos autores demonstram que a migração ao vivo de aplicações com estado pode resultar em três tipos distintos de erros, como erros de reenvio, erros de reprocessamento e erros de ordem incorreta. Kotikalapudi (2017) propõe a realização da migração ao vivo de tecnologias de virtualização baseadas em hipervisor e contêineres, especificamente a Máquina Virtual do Kernel (KVM) e os contêineres Linux (LXC).

3.4 Trabalhos que Utilizaram a Política **Cold**

A última classificação trata-se dos trabalhos que utilizam a política de migração Cold. A política de migração Cold difere das abordagens ao vivo, uma vez que não envolve a transferência em tempo real. O que implica que o impacto imediato na disponibilidade é

menor em comparação com migrações ao vivo. Como resultado, a investigação aprofundada de migrações cold pode não ter sido tão explorada, o que consequentemente resultou na limitada presença de trabalhos nessa categoria específica.

Torre et al. (2019) apresenta um teste que analisa a migração de contêineres considerando diversas condições ajustáveis, como o tamanho da imagem, a capacidade de rede e a carga de CPU e RAM. A abordagem visa avaliar e quantificar o desempenho da migração em cenários de condições variáveis e medir o tempo necessário para a migração. O foco dos autores é aprimorar a eficiência da migração de serviços em tempo real. Karhula, Janak e Schulzrinne (2019) utiliza o Docker e a ferramenta CRIU para a realização de checkpoints e a suspensão de funções de longa duração. A principal motivação por trás desse estudo é aprimorar a eficiência no uso de recursos para a execução de funções de longa duração em dispositivos IoT. Além disso, uma das preocupações abordadas é a possibilidade de reduzir a utilização de recursos durante a execução dessas funções.

3.5 Discussão Sobre as Contribuições

Este trabalho propõe dois modelos SPN para a modelagem de algumas políticas de migração de contêineres encontradas na literatura. Para uma melhor explanação das contribuições deste trabalho em relação ao estado da arte apresentado no Capítulo 2, a seguir destacamos os cinco pontos que julgamos merecer atenção.

1. **Políticas:** Nossa proposta explorou mais políticas de migração do que os demais trabalhos levantados. Mais importante do que tal abrangência, destacamos o uso de modelagem em si que permite analisar diversas configurações destas mesmas políticas. Cada política possui suas vantagens e desvantagens e são muito dependentes da infraestrutura computacional do avaliador. Também deve ser levado em conta níveis de acordo de serviço caso seja um serviço computacional público. A combinação de avaliação de desempenho das quatro políticas em questão junto à modelagem analítica é um diferencial em relação a trabalhos anteriores.

2. **Métricas:** A primeira métrica selecionada neste trabalhou foi o MTT, utilizada para medir o tempo total de migração, onde observamos o tempo que o contêiner leva para iniciar o processo de migração até o momento onde o contêiner está disponível no host de destino. A segunda métrica selecionada foi o MMT, utilizada para medir o tempo médio para concluir a migração de todos os contêineres. A terceira métrica escolhida foi a probabilidade de descarte, que calcula a probabilidade de um contêiner não finalizar o processo de migração. A quarta métrica escolhida foi a utilização, a qual observa a utilização dos recursos computacionais do sistema de migração. A quinta métrica selecionada foi a taxa de migração, a que observa a frequência com

que contêineres são movidos entre o ambiente de computacional. A combinação das cinco métricas fazem, portanto, com que esta dissertação agregue valor ao trabalho de planejamento de migração de contêineres.

3. **Método de avaliação:** A modelagem proposta neste trabalho permite representar e computar características de desempenho de diferentes estratégias de migração de contêiner. As representações abstratas possibilitam prever e entender o comportamento de sistemas sob diferentes cenários, mesmo em situações não previamente observadas. Modelos analíticos possuem um alto poder de representatividade, que possuem como um dos principais benefício seu baixo custo e relativamente alto nível de aceitação dos resultados. A modelagem oferece uma visão preditiva e uma compreensão mais profunda em comparação à mensuração. Embora a mensuração seja o método mais usado na literatura, se concentra principalmente na execução real do sistema, proporcionando alto custo e significativa dificuldade de generalização dos resultados. Ademais, o modelo estando pronto basta reexecutar o modelo usando uma ferramenta de modelagem, que no caso deste trabalho foi o Mercury (MACIEL et al., 2017). Sem dúvida, o uso de modelos é um diferencial em comparação a estudos prévios da literatura.

4. **Análise de sensibilidade:** A análise de sensibilidade com DoE foi empregada neste trabalho para identificar os fatores que mais influenciam a migração de contêineres. O objetivo foi obter a maior quantidade de informações com o mínimo de experimentos. Essa análise não foi identificada em nenhum trabalho da literatura e por isso não foi incluído na tabela comparativa. O principal objetivo deste estudo é utilizar o DoE para identificar qual componente impacta mais no MTT. O DoE permitiu aprofundar o processo em estudo e identificar os fatores críticos do processo analisado. DoE possui um grande poder de otimização de um sistema e é algo novo em relação ao o que já foi feito na área.

5. **Migração paralela:** A principal contribuição deste trabalho está atrelada à migração paralela de contêineres. A migração paralela de contêineres é um processo no qual vários contêineres são movidos de um ambiente de execução para outro. Apesar de não ser uma atividade muito explorada em trabalhos da literatura, migração paralela é essencial em caso de falhas de sistema repentina. O modelo SPN proposto calcula facilmente as métricas de interesse considerando qualquer quantidade de contêineres migráveis.

Tabela 2 – Trabalhos relacionados.

Trabalhos	**Políticas**	**Métricas**	**Uso do CRIU**	**Análise de Sensibilidade**	**Método de Avaliação**
(TORRE et al., 2019)	Cold	MTT	✓	✗	Mensuração
(STOYANOV; KOLLINGBAUM, 2018)	PreCopy, PostCopy, Hybrid	MTT	✓	✗	Mensuração
(MA et al., 2018)	Não explícito	MTT, Largura de Banda de Rede, Latência de Rede	✓	✗	Mensuração
(DI; SHAO; TAN, 2021)	Não explícito	MTT	✓	✗	Mensuração
(KOTIKALAPUDI, 2017)	Não explícito	MTT, Downtime, Utilização	✓	✗	Mensuração
(TAY; GAURAV; KARKUN, 2017)	Não explícito	MTT	✗	✗	Mensuração e Modelagem
(XU et al., 2020)	PreCopy	MTT, Downtime, Tempo de Extração de Imagens	✓	✗	Mensuração
(GONZÁLEZ; ARZUAGA, 2020)	Não explícito	Utilização	✓	✗	Mensuração
(GOVINDARAJ; ARTEMENKO, 2018)	PreCopy, PostCopy, Hybrid	MTT, Downtime	✓	✗	Mensuração
(BENJAPONPITAK; KARAKATE; SRIPANIDKULCHAI, 2020)	PreCopy	MRT, Throughput	✓	✗	Mensuração
(ABDULLAH; HADEED et al., 2022)	Não explícito	Utilização, Tempo de Resposta	✓	✗	Mensuração
(RAMANATHAN et al., 2021a)	Cold, PreCopy	MTT, Downtime	✓	✗	Mensuração
(FAN et al., 2019)	PreCopy	MTT, Utilização	✗	✗	Mensuração
(MACHEN et al., 2017)	Não explícito	MTT, Downtime	✗	✗	Mensuração
(PECHOLT; HUBER; WESSEL, 2021)	Cold, PreCopy, PostCopy	MTT, Downtime, Confidêncialidade, Integridade	✗	✗	Mensuração
(SMIMITE; AFDEL, 2019)	PreCopy	MTT, Consumo de Mémoria, Tráfego de Rede	✓	✗	Mensuração
(AL-DHURAIBI et al., 2017)	PreCopy	Utilização, Tempo de Resposta	✓	✗	Mensuração
(CHOU et al., 2019)	PreCopy, PostCopy	MTT, Downtime, Envelhecimento	✓	✗	Mensuração
(DAS; SIDHANTA, 2023)	Não explícito	Throughput, Latência	✓	✗	Mensuração
(MAJEED et al., 2020)	Não explícito	R2, MAPE, MAE	✓	✗	Modelagem
(KARHULA; JANAK; SCHULZRINNE, 2019)	Cold	Utilização	✓	✗	Mensuração
(BHARDWAJ; KRISHNA, 2022)	PreCopy	MTT, Downtime, Utilização	✓	✗	Mensuração
(RAMANATHAN et al., 2021b)	PreCopy	MTT, Downtime	✓	✗	Mensuração
(KAKAKHEL et al., 2018)	Não explícito	MTT, Downtime	✓	✗	Mensuração
(BACCARELLI; SCARPINITI; MOMENZADEH, 2018)	Cold, PreCopy, PostCopy	Taxa de Migração, Energia	✓	✗	Mensuração
Este trabalho	Cold, PreCopy, PostCopy, Hybrid	MTT, MMT, Probabilidade de Descarte, Utilização, Taxa de Migração	✓	✓	Mensuração e Modelagem

3.6 Conclusão do Capítulo

Este Capítulo apresentou uma análise detalhada de 25 trabalhos relacionados à migração, destacando-se pela abrangência das políticas de migração exploradas e pela utilização de modelagem para analisar diversas configurações dessas políticas. As métricas selecionadas neste estudo, incluindo o Tempo Total de Migração (MTT) para todos os contêineres a serem migrados, o Tempo Médio de Migração (MMT) de todos os contêineres, a probabilidade de descarte, a utilização dos recursos computacionais e a taxa de migração, proporcionaram uma avaliação abrangente do processo de migração de contêineres.

A modelagem analítica utilizada permitiu representar e computar características de desempenho de diferentes estratégias de migração de contêineres, oferecendo uma visão preditiva e uma compreensão mais profunda em comparação à mensuração. A análise de sensibilidade com DoE foi empregada para identificar os fatores que mais influenciam a migração de contêineres. A principal contribuição deste trabalho está na migração paralela de contêineres, um processo no qual vários contêineres são movidos simultaneamente de um ambiente de execução para outro. Após explorar os trabalhos relacionados a este estudo, o próximo Capítulo se concentra em apresentar a metodologia deste estudo.

4 Metodologia

O objetivo principal deste trabalho é desenvolver um modelo para avaliar a migração de contêineres utilizando quatro políticas de migração. A Figura 13 apresenta um fluxograma que resume a estratégia usada neste trabalho.

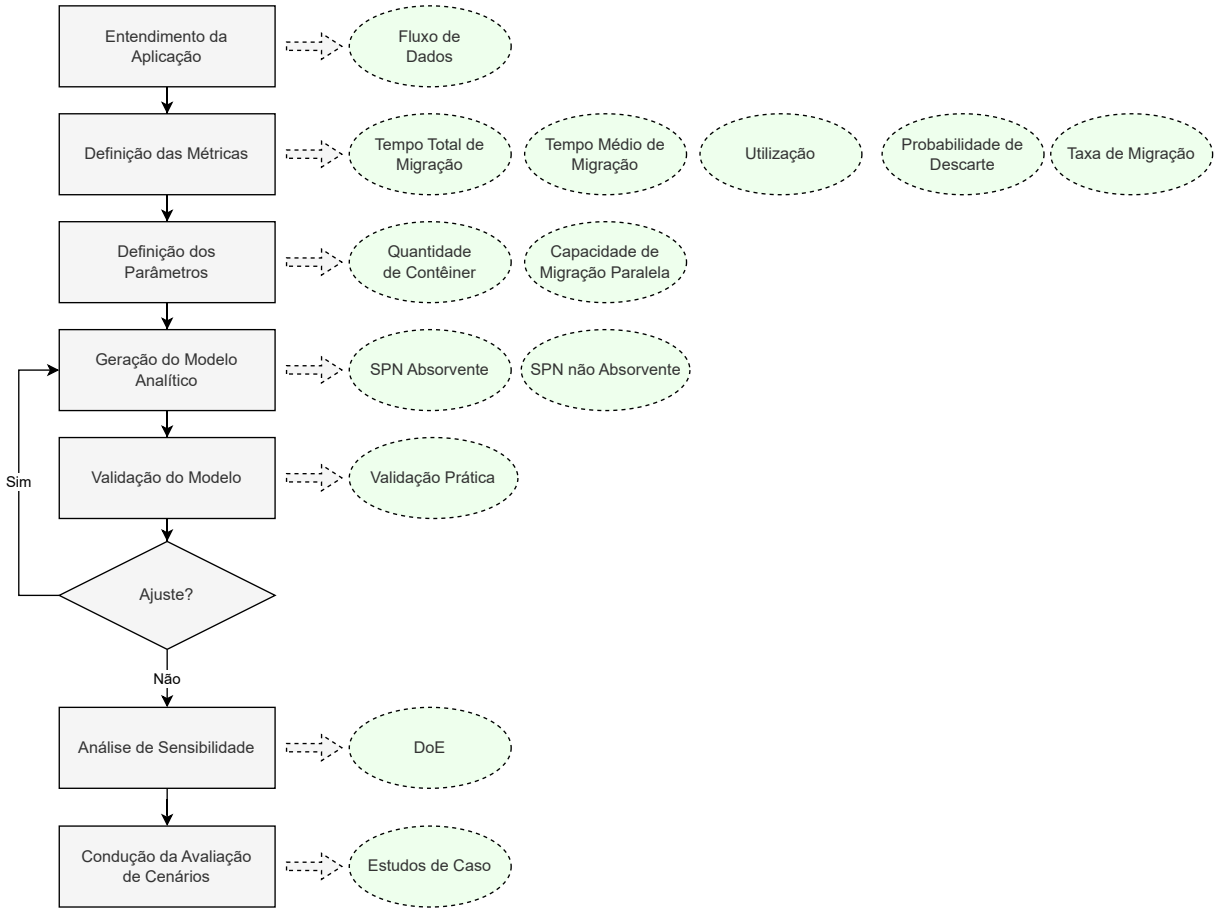


Figura 13 – Metodologia de desenvolvimento do modelo analítico.

- **Entendimento da Aplicação:** É fundamental entender o funcionamento da aplicação, identificar a quantidade e os tipos de componentes envolvidos no processo de migração de contêiner. Isso inclui detalhar o fluxo de dados do sistema, especificando para onde os dados serão enviados após o início do processo de migração. É importante considerar quais metadados são enviados para o host de destino. Além de vários fatores relevantes a serem levados em consideração no processo de migração. Por fim, é essencial estar ciente das limitações das ferramentas utilizadas no processo de migração.

- **Definição das Métricas:** As métricas de interesse são identificadas, levando em consideração as informações que o modelo pode fornecer para diagnosticar o desempenho do sistema. Nesta dissertação, as métricas selecionadas podem ser importantes na percepção do usuário final e úteis para administradores de sistema, são elas: Tempo Total de Migração (MTT), Tempo Médio de Migração (MMT), Probabilidade de Descarte, Utilização e Taxa de Migração.

- **Definição dos Parâmetros:** Os parâmetros que serão inseridos no modelo são definidos. Esses parâmetros definem o comportamento e capacidade de recursos para os componentes no sistema de migração de contêineres. Neste trabalho, os parâmetros adicionados foram a quantidade de elementos a serem migrados (K) e capacidade associada aos contêineres de migração (C), peso da falha no processo de migração e a taxa de migração.

- **Geração dos Modelos Analíticos:** Dois modelo de desempenho usando Rede de Petri Estocásticas é desenvolvido aqui, um com estado absorvente e outro sem estado absorvente. Nesta parte, o modelo é construído considerando as métricas e parâmetros definidos, e os resultados esperados. A escolha da Rede de Petri Estocásticas se dá porque o cenário de migração de contêiner considerado possui poucos componentes e o modelo SPN atende satisfatoriamente a sistemas de baixa complexidade.

- **Validação do Modelo:** Foi realizada uma validação em um ambiente real que contempla a Migração de Contêiner. Os resultados serão comparados com os valores retornados pelo modelo através do intervalo de confiança e teste T. Se os valores forem semelhantes, o modelo estará validado, caso contrário, haverá a necessidade de ajustes no modelo. Se após a validação for detectada a necessidade de realizar ajustes no modelo, deve-se retornar à etapa de geração do modelo analítico.

- **Análise de Sensibilidade:** Usando DoE, a análise apresenta resultados considerando fator e níveis predefinidos. A partir disso, é possível identificar os fatores mais relevantes para a migração de contêiner, e o modo como a interação entre os fatores e as variações dos seus níveis impactam no desempenho.

- **Condução da Avaliação de Cenários:** Dois cenários construídos são avaliados usando SPN por meio de simulação. Um cenário se propõe a avaliar a migração de contêineres, porém com uma limitação na quantidade total que pode ser migrada, utilizando um modelo com estado absorvente. O outro cenário avalia a migração de contêineres com base na taxa de migração, utilizando um modelo sem estado absorvente. A simulação nos permite modelar o comportamento do sistema em diversas situações, ajustando os níveis dos fatores relevantes para cada cenário. Estes incluem a quantidade de contêineres a serem migrados, a migração paralela, a probabilidade de descarte e a taxa de migração. Por fim, as métricas escolhidas serão

analisadas, permitindo identificar quais configurações proporcionam um desempenho satisfatório na migração de contêiner.

4.1 Conclusão do Capítulo

Este capitulo apresentou uma metodologia onde foram criados dois modelos para avaliar a migração de contêineres, considerando métricas e parâmetros específicos. Através da validação do modelo e da análise de sensibilidade, foram indentificados os fatores mais relevantes para a migração de contêineres. A avaliação de cenários irá permitir modelar o comportamento do sistema em diversas situações, destacando as configurações que proporcionam um desempenho satisfatório na migração de contêineres. O estudo contribui para a compreensão e otimização do processo de migração de contêineres. Após apresentar a metodologia utilizada neste estudo, o próximo Capítulo se concentra em mostrar o Modelo SPN com estado Absorvente, validação, estudo de caso e DoE.

5 Modelo SPN Absorvente

Este trabalho adotou a estratégia de modelagem e mensuração. Ao contrário do uso de modelagem, a técnica de mensuração é atrelada totalmente ao ambiente de execução e aos parâmetros adotados. Por exemplo, caso o avaliador execute a migração de dois elementos, não é possível com este experimento predizer como será o comportamento da migração de cem elementos paralelos sem ter uma infraestrutura compatível para executar tão atividade. Por outro lado, a modelagem permite isso.

Portanto, este Capítulo apresenta um modelo SPN para representar e computar características de desempenho de uma rede com diferentes estratégias de migração de contêiner. O objetivo do modelo proposto é auxiliar administradores de sistemas baseados nessa arquitetura na tarefa complexa de ajustar vários parâmetros adequadamente para alcançar níveis de desempenho desejáveis. Logo, o modelo deve ser útil para checar o efeito de mudanças no sistema, antes mesmo que elas sejam implementadas.

A Figura 14 mostra o modelo do SPN absorvente proposto. No Capítulo 2 já foi explicado o funcionamento de cada política de migração, bem como apresentado o detalhamento de cada componente e suas funcionalidades. Assim, este Capítulo destaca o funcionamento do modelo de forma geral. Para o processo de migração, os *tokens* representam um contêiner a ser migrado. Existe uma quantidade de elementos migráveis correspondentes à marcação K no lugar Node_A. Assim, os *tokens* se encontram inicialmente no Node_A e serão movidos para o Node_B. A migração somente ocorrerá caso tenha capacidade de migração. Tal capacidade é dada pela quantidade de agentes de migração paralela, dado pela marcação C no lugar Cap. O lugar Node_B é do tipo absorvente, como explicado na seção anterior. Quando os *tokens* chegam ao destino Node_B, a migração está finalizada. O cálculo é feito a partir da probabilidade de todos os *tokens* presentes no Node_A chegarem ao Node_B. A métrica CDF é calculada com base no lugar Node_B.

A escolha de qual caminho o *token* vai seguir se dá por condições de guarda presentes nas seguintes transições: Dump1, PDump2, Dump3 e PDump4. As transições são ativadas conforme a variação da variável POLÍTICA. Se a variável POLÍTICA for igual a 1, ela seguirá pela política de migração Cold. Se a variável POLÍTICA for igual 2, o *token* seguirá pela política PreCopy. Se POLÍTICA for igual 3, o *token* seguirá pela política PostCopy. Por fim, se a variável POLÍTICA for igual a 4, a migração será referente a política Hybrid. Vale ressaltar que podem existir N tipos de políticas de migração. Todas as transições possuem distribuição de probabilidade exponencial que foi condizente com o experimento real de validação. Todas as transições possuem semântica *infinite*

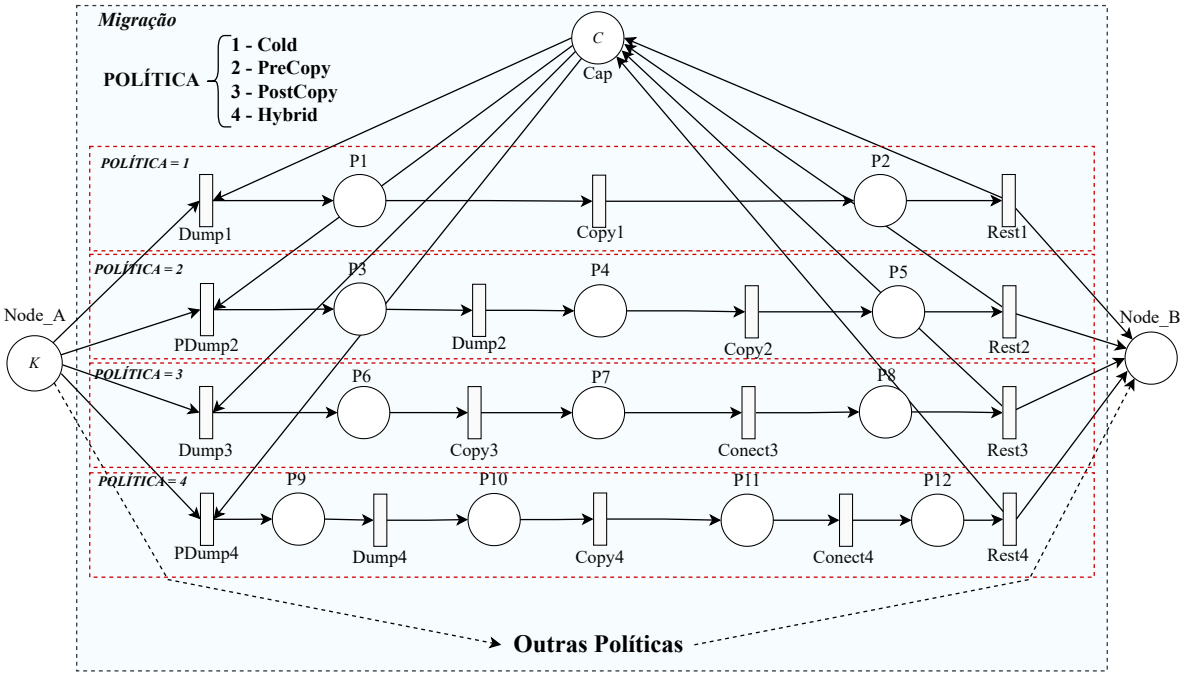


Figura 14 – Modelo SPN para calcular o tempo total de migração por diferentes políticas de migração.

server, indicando que as transições podem ser disparadas com múltiplos *tokens* em paralelo (MARSAN et al., 1998). A Tabela 3 apresenta os principais elementos do modelo.

5.1 Métrica

O Migration Total Time (MTT) é o tempo médio para terminar a migração de todos os elementos a serem migrados. MTT é o tempo esperado para chegar a um impasse de marcação. Em teoria de redes de Petri e cadeias de Markov, quando um *token* não possui outro lugar para ser consumido, o estado absorvente pode ser calculado gerando uma Continuous-time Markov Chain (CTMC) associada ou por análise numérica (NELSON, 2013). No presente trabalho, utilizamos análise transiente, que a partir de uma SPN gera uma CTMC e faz o cálculo do estado absorvente.

O comportamento da CTMC pode ser descrita pela equação de Kolmogorov dado o vetor de probabilidade inicial $\pi(0)$ (ver Equação 5.1). A Equação 5.2 fornece o tempo total esperado que o CTMC gasta no estado i durante o intervalo [0, t). O tempo gasto antes da absorção pode ser calculado restrito aos estados do conjunto de estados não absorventes (N) por $lim_{t\to\infty} L_N(t)$. Assim, $L(t)$ satisfaz a Equação 5.3 onde $\pi_N(0)$ é o vetor $\pi(0)$ restrito aos estados do conjunto N. Q_N é a matriz geradora infinitesimal restrita aos estados não absorventes (PINHEIRO et al., 2018). Finalmente, o MTT pode ser descrito como a Equação 5.4 (BOLCH et al., 2006).

Tabela 3 – Descrição dos elementos principais do modelo.

Tipo	**Elemento**	**Descrição**
Lugares	Node_A	Local onde se encontra o elemento de migração a ser migrado
	Node_B	Local para o será migrado o elemento de migração
	P11	Capacidade associada ao processamento de uma migração
Transições Temporizadas	Dump1	Tempo associado a criação do ponto de restauração
	Copy1	Tempo associado para transferir os metadados do elemento a ser migrado
	Rest1	Tempo associado para a restauração do contêiner para a política de migração Cold
	PDump2	Tempo associado a verificação do aplicativo no nó de origem
	Dump2	Tempo associado a coleta das informações das páginas de memória moficidadas
	Copy2	Tempo associado à transferência das informações das páginas de memória moficidadas
	Rest2	Tempo associado para a restauração do contêiner para a política de migração PreCopy
	Dump3	Tempo associado a criação do ponto de restauração
	Copy3	Tempo associado à transferência das informações das páginas de memória
	Conect3	Tempo associado para conectar o Node_A ao Node_B
	Rest3	Tempo associado para a restauração do contêiner para a política de migração PostCopy e transferência das páginas com falha
	PDump4	Tempo associado a verificação do aplicativo no nó de origem
	Dump4	Tempo associado a criação do ponto de restauração
	Copy4	Tempo associado à transferência das informações das páginas de memória moficidadas
	Conect4	Tempo associado para conectar o Node_A ao Node_B
	Rest4	Tempo associado para a restauração do contêiner para a política de migração Hybrid e transferência das páginas com falha
Marcações dos Lugares	K	contêiner a ser migrado
	C	Capacidade associada aos contêiner de migração

$$\frac{d\pi(t)}{dt} = \pi(t)Q \tag{5.1}$$

$$L_i(t) = \int_0^t \pi_i(x)dx \tag{5.2}$$

$$L_N(\infty)Q_N = -\pi_N(0) \tag{5.3}$$

$$MTT = \sum_{i \in N} L_i(\infty) \tag{5.4}$$

Usando a Figura 15 como exemplo, o MTT é baseado em um conjunto de probabilidades para quando um *token* sai do lugar Início para o Fim. Nesse exemplo, a migração é dividida em dois tempos divididos nas transições T1 e T2. A migração de K elementos pode ser feita em paralelo dependendo da quantidade de agentes de migração disponíveis (marcação C) (GERMAN, 2000; MARSAN et al., 1998).

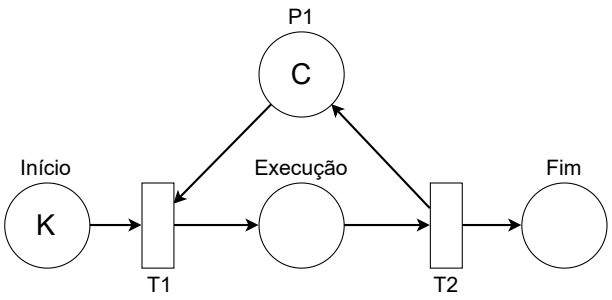


Figura 15 – Exemplo de Modelo SPN.

5.2 Validação Prática

Para utilizar um modelo analítico (e.g., PN, CTMC, redes de filas, etc.) é essencial validar tal modelo para verificar sua corretude. A validação se trata de comparar os resultados gerados pelo modelo com os resultados advindos de uma simulação ou mensuração real (JAIN, 1990). Esta Seção apresenta a validação do modelo absorvente proposto. Começamos por descrever o protótipo sintético e depois apresentamos a metodologia e os resultados. Para validar nosso modelo SPN proposto, desenvolvemos um protótipo e conduzimos experimentos em um cenário real. O objetivo foi comparar o Tempo Total de Migração (MTT) calculado pelo modelo com o MTT observado nos experimentos. A configuração do experimento é apresentada na Figura 16. Durante os experimentos, testamos as quatro políticas detalhadas no Capítulo 2.1.2 (Cold, PreCopy, PostCopy e Hybrid), e os dados coletados foram usados para a análise.

Nosso ambiente de testes composto por duas máquinas virtuais, ambas hospedadas no mesmo Hypervisor. Cada máquina virtual possui 5000 MB de RAM e 2 núcleos de

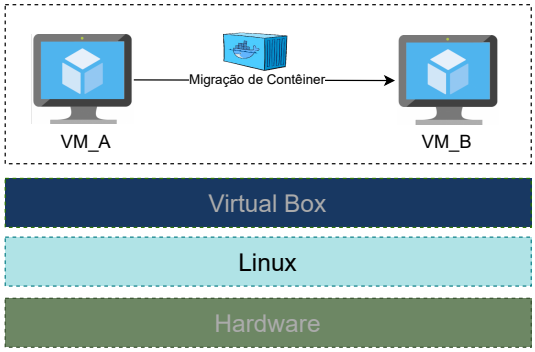


Figura 16 – Configuração do Experimento.

processamento, e opera com Ubuntu 22.04 e Kernel Linux. Elas utilizam o CRIU 3.17.1 para as funcionalidades de *checkpoint* e *restore*, o OpenSSL 3.0.2 para transferência de arquivos e o runC 1.1.9 como runtime de contêiner. A aplicação em uso é projetada para ser intensiva em termos de consumo de memória RAM e processamento. Utilizamos uma aplicação em Python que calcula uma sequência de Fibonacci e soma cada número ao seu antecessor na sequência. Para garantir um alto uso da memória RAM, a aplicação gera uma imagem correspondente a cada número calculado na sequência de Fibonacci, armazenada diretamente na memória RAM, em vez de ser salva em um disco rígido. Isso resulta em um constante preenchimento da memória RAM com novas imagens. A adoção da sequência de Fibonacci foi inspirada pelo trabalho de (DAYO, 2021) que fez algo semelhante com contêineres Docker.

A Figura 17 apresenta o fluxo de execução do experimento. O fluxograma mostra o processo de migração de contêineres em um ambiente de teste. O processo começa com a seleção da política de migração, que pode ser Cold, PreCopy, PostCopy ou Hybrid. Na política Cold, o fluxo segue diretamente para as etapas Dump1 e Copy1 antes do Restart1. As políticas PreCopy e PostCopy passam por etapas adicionais. A política PreCopy inclui Pdump2 e Dump2, enquanto a política PostCopy inclui Dump3 e Copy3. Na política Hybrid, o processo se inicia com Pdump4 seguido por Dump4 e Copy4 antes do Restart4. Em cada uma dessas etapas intermediárias, os tempos são registrados para análise posterior. Após cada reinicialização, o log MTT é registrado. Para registrar os tempos de execução, instrumentamos o início e o fim de cada etapa de migração. Foram utilizadas as instruções da documentação oficial do CRIU[1] para garantir a execução adequada das políticas de migração exploradas. Os tempos de serviço obtidos com este experimento foram utilizados para alimentar as transições estendidas do modelo SPN. A partir desses valores, todas as transições foram preenchidas com os tempos de suas respectivas etapas. Para tornar o resultado mais confiável, repetimos a execução do experimento 50 vezes para cada política de migração.

[1] Documentação CRIU: <https://criu.org/>

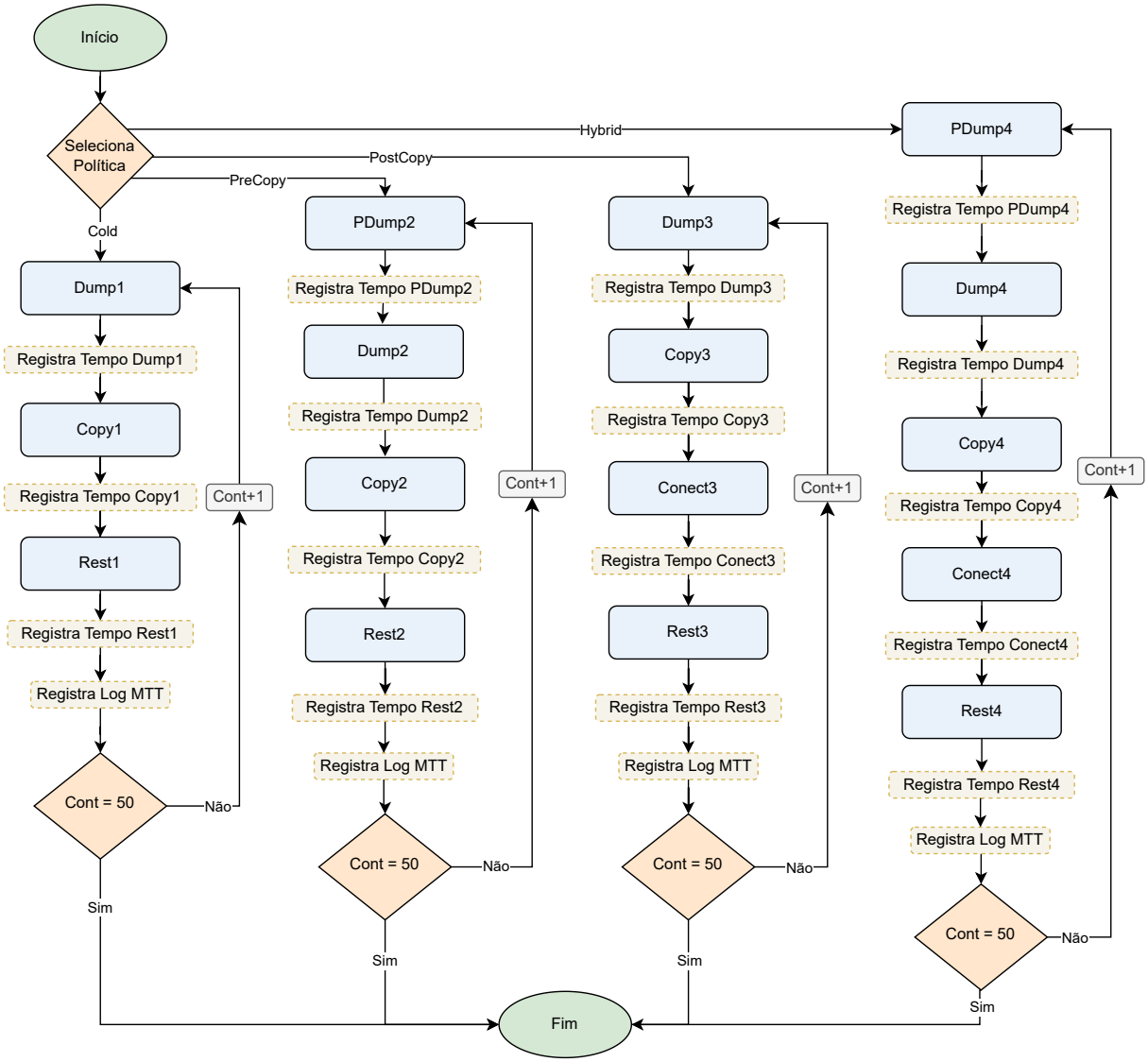


Figura 17 – Fluxograma do Experimento.

A Tabela 4 mostra os parâmetros de entrada utilizados para alimentar o modelo advindos do experimento. A Tabela 5 apresenta o resultado da validação. O modelo SPN pode ser solucionado de duas formas, por análise estacionária ou simulação estacionária. O modelo gerou os resultados por meio de simulação, condicionados a uma margem de erro de 2%. Aplicamos o teste T de duas amostra[2] para comparar o MTT gerado pelo modelo com o MTT obtido nos experimentos. Todas as amostras apresentaram distribuição normal. Para verificar a significância do Teste T, observamos o valor p. O valor p é superior a 0,05 em todos os casos. Portanto, não podemos refutar a hipótese nula de igualdade em todos os casos, com 95% de confiança. Portanto, podemos concluir que os resultados gerados pelo modelo são estatisticamente equivalentes ao experimento. O modelo reflete o ambiente real.

[2] Sample T-Test https://tinyurl.com/yanthw4e

Tabela 4 – Parâmetros de configuração de tempo usados no estudo de caso.

POLÍTICA	Name	Time(s)
Cold	Dump1	0.84352
	Copy1	4.55536
	Rest1	0.95682
PreCopy	PDump2	0.77930
	Dump2	0.75376
	Copy2	9.35288
	Rest2	0.90658
PostCopy	Dump3	4.98894
	Copy3	0.54768
	Conect3	3.70338
	Rest3	0.63900
Hybrid	PDump4	0.69168
	Dump4	5.48088
	Copy4	4.06736
	Conect4	4.11074
	Rest4	0.71172

Tabela 5 – Comparação dos resultados da mensuração com o modelo proposto.

Politica	MTT(s)		Intervalo de Confiança		Desvio Padrão		P-Value
	Exp	*Mod*	*Exp*	*Mod*	*Exp*	*Mod*	
Cold	6,355	6,308	[6,405 - 6,305]	[6,019 - 6,598]	2,161	4,659	0,949
PreCopy	11,792	11,847	[11,742 - 11,842]	[11,269 - 12,425]	2,733	9,317	0,968
PostCopy	9,879	9,749	[9,829 - 9,929]	[9,369 - 10,128]	0,902	6,121	0,883
Hybrid	15,062	15,269	[9,829 - 9,929]	[14,743 - 15,795]	1,260	8,471	0,865

5.3 Estudo de Caso

Esta Seção apresenta os resultados das simulações com o modelo SPN absorvente proposto. O objetivo desta seção é mostrar a amplitude de utilização do modelo. Focamos em dois parâmetros em particular: a quantidade de elementos a serem migrados e a capacidade de migração paralela. O estudo de caso é útil tanto para obter novas descobertas sobre o funcionamento do modelo e consequentemente do sistema real, como também ilustrar como o modelo pode ser explorado. Os tempos utilizados nas transições do modelo foram os mesmos extraídos da validação. A representação do modelo e a computação dos resultados da análise numérica foram obtidos com a ferramenta Mercury (MACIEL et al., 2017). Nos estudos de caso, utilizamos a mesma quantidade de políticas de migração representadas no modelo apresentado no Capítulo 2.1.2. Os parâmetros foram alimentados baseados em tempos colhidos no processo de validação, que estão presentes na Tabela 4.

A Figura 18 apresenta o resultado para o MTT em função da variação da quantidade de elementos a serem migrados. Foram testadas as seguintes quantidades de elementos migráveis: K = [10, 20, ... , 90, 100]. Ao contrário da validação, aqui foi configurado o uso de dois agentes de migração paralela (marcação C = 2). O tempo de migração aumenta

proporcionalmente à quantidade de elementos migrados, pois há um limite paralelo de migrações igual a 2. A política Hybrid teve o maior de todos MTT, chegando a ≈ 750s. A política Cold teve um MTT menor do em relação as outras políticas apresentadas. Comparando as quatro políticas vemos que com menos elementos a serem migrados, os MTTs são semelhantes e à medida que a quantidade aumenta, estas linhas se distanciam. A política Hybrid que possui o pior desempenho, tem um ângulo de piora maior do que os demais conforme aumenta a quantidade de elementos a serem migrados. Por exemplo, comparando o intervalo de K=[50-100], temos que o Cold possui um aumento de MTT=158s enquanto que o Hybrid possui um aumento de MTT=247s. Portanto, para grandes quantidades de elementos a serem migrados a escolha da política se torna cada vez mais criteriosa, pois isso impacta de forma expressiva no valor final de MTT.

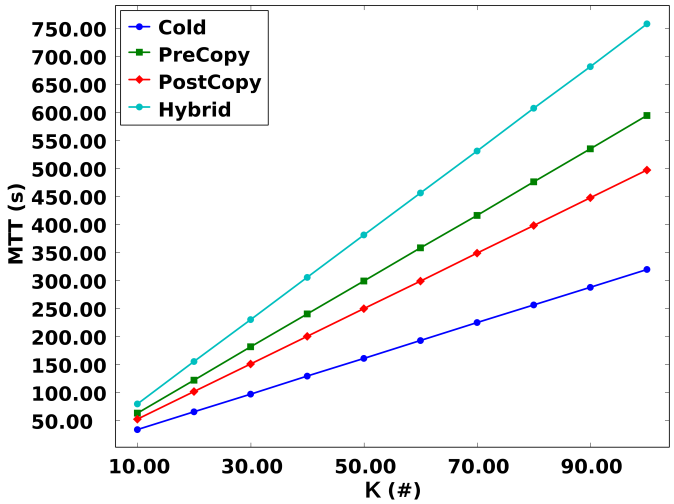


Figura 18 – MTT em função do número de contêineres a serem migrados (K).

A Figura 19 apresenta o MTT em função do aumento da capacidade de migrações paralelas (C). O C foi variado com os seguintes valores: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 e 15. A análise foi feita considerando 50 elementos a serem migrados (K=50). Novamente, foram comparadas as quatro políticas. Durante o começo da análise, as quatro políticas de migração começam com um tempo consideravelmente alto, onde a Hybrid tem o pior desempenho. Desde o início da variação de C até o final, o tempo MTT segue diminuindo até se estabilizar. A estabilização acontece para as quatro políticas para C >= 13. Isso implica que há uma baixa alteração a partir desse ponto, pois o MTT tem uma variação muito baixa em relação a um C >= 3. Apesar das linhas se aproximarem com o aumento do C, observa-se que elas não se tocam (pelo menos não até este valor de C=15). No gráfico anterior, para a variação do K, no início temos que o MTT para as duas estratégias de contêineres é praticamente igual. Aqui, para a variação de C, essa igualdade nunca existe. Portanto, o impacto da variação de C foi maior do que o impacto da variação de K. A principal contribuição deste estudo de caso é mostrar que o avaliador

pode ter uma grande economia de recursos conhecendo o limiar de C que fará com que não melhore mais o desempenho mesmo o aumentando.

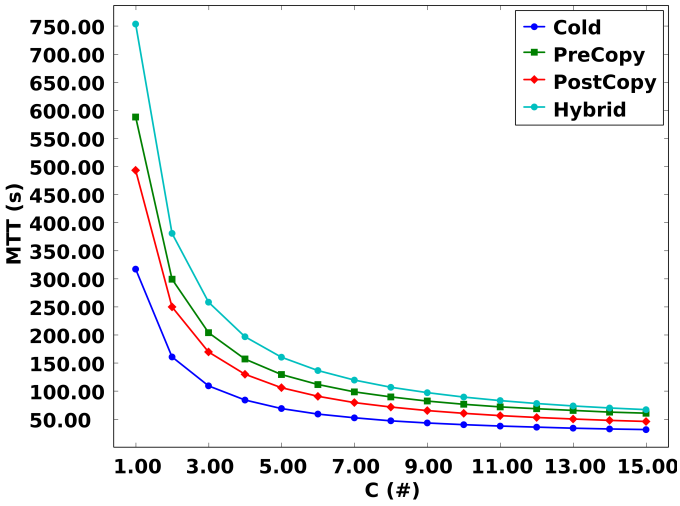


Figura 19 – MTT em função da capacidade de migrações paralelas (C).

O modelo proposto neste trabalho também calcula a função de distribuição cumulativa (CDF). Com a CDF é possível estimar a probabilidade (entre 0 e 1) de terminar a execução antes de um tempo específico $[P(T < t)]$ e a probabilidade de terminar a execução em um intervalo de tempo $[P(t1 < T < t2) = P(T < t2) - P(T \leq t1)]$. Imagine que existe uma tarefa a ser executada, e esta tarefa pode ser realizada por uma, duas ou três máquinas virtuais em paralelo. Quanto mais "trabalhadores" maior é a probabilidade de terminar a execução na maior parte do tempo (SILVA et al., 2017).

A Figura 20 apresenta a CDF, que permite determinar a probabilidade de toda a migração finalizar em determinada janela de tempo. Essa janela de tempo refere-se ao MTT. A análise (do tipo transiente) foi feita considerando 50 elementos a serem migrados (K=50) e com capacidade paralela de migração igual a 2 (C=2). A janela total de MTT variou entre 0 e 459s. No geral, a probabilidade de finalização da migração é maior para Cold, que é maior do que PostCopy, que é maior do que a PreCopy, por fim, maior que a Hybrid. A partir daqui iremos considerar a notação P[política] para denotar a probabilidade de uma política específica. P[Cold] = P[PreCopy] = P[PostCopy] = P[Hybrid]= 0 para os MTTs iguais a 130s, respectivamente. Um ponto interessante é analisar o MTT = 250s, onde temos que P[Cold] = 100%, P[PreCopy] = 52%, P[PostCopy] = 76% e P[Hybrid] = 0%. Consideremos para este exemplo que MTT = 250ms é um requisito importante de Acordo de Nível de Serviço a ser atendido. Se o analista não quiser assumir riscos, obviamente irá adotar a política Cold, porém, se este requisito não for tão crítico, ele pode usar o PostCopy assumindo o risco que há apenas 76% de chance de a migração terminar nesse requerido tempo. Finalmente, vale ressaltar que no tempo MTT=459s, a probabilidade de término da migração é de 100% para todas as políticas. Novamente, se o

requisito demanda que se tenha um MTT igual a 459s, qualquer uma das quatro políticas irá atendê-lo.

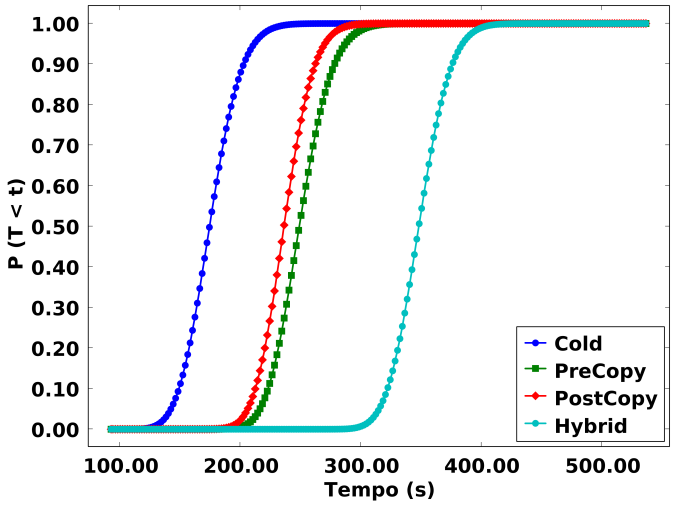


Figura 20 – CDF considerando o tempo total de migração.

5.4 Análise de Sensibilidade Usando DoE

Esta seção apresenta uma análise de sensibilidade usando o método DoE (FEITOSA et al., 2021; GONÇALVES et al., 2021; SANTOS et al., 2021b; CARVALHO et al., 2020; SANTOS et al., 2021a; VICTOR et al., 2021). O objetivo foi identificar qual componente impacta mais no MTT (tempo total de migração), a fim de permitir aprofundar o conhecimento sobre o processo em estudo. O DoE permitiu identificar os fatores críticos que afetam a migração de contêiner. A Tabela 6 mostra os fatores e níveis usados na execução do DoE.

A alteração dos tempos e valores ligados aos níveis é feita com a elevação e diminuição de 50% do valor original. A Tabela 7 mostra os resultados das 32 execuções relacionadas às combinações usadas na criação do DoE. A Figura 21 apresenta o efeito de influência de cada fator individualmente e os efeitos das interações entre fatores. O fator K teve o maior impacto sobre o MTT, seguido pelo C e a interação entre C e K. As outras interações que tiveram efeito foram K e Dump4, C e Dump4 essas interações tiveram um efeito de ≈ 16.0, mostrando impacto no MTT. As interações de K e Conect4, C e Conect4, k e Copy4, C e Copy4 tiveram um efeito pequeno em relação as outras interações apresentadas, chegando a ≈ 12. A interação que teve o menor efeito foram Dump4 e Copy4, Dump4 e Conect4, Copy4 e Conect4, chegando a um efeito de quase 0. Esses resultados são relevantes para o desenvolvimento de estratégias na melhoria do MTT. Os dados apresentados permitem que o arquiteto de sistema identifique e priorize os fatores mais críticos. Proporcionando uma melhor eficiência no processo de migração de contêiner para a política de migração Hybrid.

Tabela 6 – Tabela de Fatores e Níveis para a Política de migração Hybrid

Fatores	**Configuração Baixa**	**Configuração Alta**
K	5,0	15,0
C	1,0	3,0
Dump4	2,74(s)	8,221(s)
Copy4	2,033(s)	6,101(s)
Conect4	2,055(s)	6,166(s)

Tabela 7 – Tabela de Combinação para a política de migração Hybrid

K	**C**	**Dump4(s)**	**Copy4(s)**	**Conect4(s)**	**MTT(s)**
5,00	1,00	2,74	2,03	2,05	41,14
5,00	1,00	2,74	2,03	6,16	61,72
5,00	1,00	2,74	6,10	2,05	61,47
5,00	1,00	2,74	6,10	6,16	82,02
5,00	1,00	8,22	2,03	2,05	68,58
5,00	1,00	8,22	2,03	6,16	89,11
5,00	1,00	8,22	6,10	2,05	88,91
5,00	1,00	8,22	6,10	6,16	109,47
5,00	3,00	2,74	2,03	2,05	17,62
5,00	3,00	2,74	2,03	6,16	26,96
5,00	3,00	2,74	6,10	2,05	26,83
5,00	3,00	2,74	6,10	6,16	35,62
5,00	3,00	8,22	2,03	2,05	30,49
5,00	3,00	8,22	2,03	6,16	39,03
5,00	3,00	8,22	6,10	2,05	38,93
5,00	3,00	8,22	6,10	6,16	47,36
15,00	1,00	2,74	2,03	2,05	123,46
15,00	1,00	2,74	2,03	6,16	185,09
15,00	1,00	2,74	6,10	2,05	184,53
15,00	1,00	2,74	6,10	6,16	246,17
15,00	1,00	8,22	2,03	2,05	205,68
15,00	1,00	8,22	2,03	6,16	267,31
15,00	1,00	8,22	6,10	2,05	266,73
15,00	1,00	8,22	6,10	6,16	328,41
15,00	3,00	2,74	2,03	2,05	45,01
15,00	3,00	2,74	2,03	6,16	68,08
15,00	3,00	2,74	6,10	2,05	67,81
15,00	3,00	2,74	6,10	6,16	90,29
15,00	3,00	8,22	2,03	2,05	76,21
15,00	3,00	8,22	2,03	6,16	98,41
15,00	3,00	8,22	6,10	2,05	98,19
15,00	3,00	8,22	6,10	6,16	120,34

A Figura 22a mostra a interação entre K e C. Existe interação entre os níveis 1 e 3 de C em relação ao K. Caso o avaliador tenha que escolher migração única, o resultado sugere adotar o mínimo possível de elementos a serem migrados. Com a migração de 3

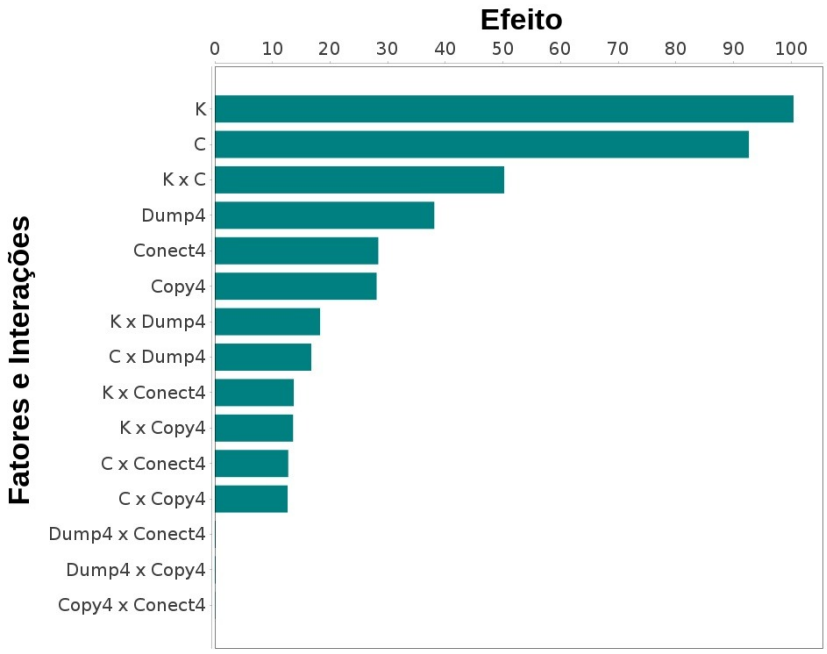


Figura 21 – Fatores e Interações e seus respectivos efeitos para a política de migração Hybrid

contêineres simultaneamente, obtemos um pequeno crescimento do MTT ao aumentar a quantidade de elementos a serem migrados. A Figura 22b apresenta a interação entre K e Dump4. Há uma interação significativa entre os fatores, sendo a combinação mais eficiente para reduzir o MTT obtida com Dump4 = 2,74s e K = 5,0, resultando em um MTT aproximado de ≈44,0s. No entanto, é importante destacar que a variação de Dump4 = 8,221s tem um impacto ligeiramente maior chegando a um MTT aproximado de ≈63,0s, considerando o K = 5.

A Figura 22c apresenta a interação entre K e Copy4. A fim de minimizar o MTT, a combinação ideal é alcançada com Copy4 = 2,033s e K = 5. Essa configuração resulta em um MTT de ≈46,0s. No entanto, é relevante observar que o impacto da variação em Copy4 = 6,101s é ligeiramente maior em relação a Copy4 = 2,033s, considerando o K = 15. A Figura 22d apresenta a interação entre os fatores K e Conect4. Caso o avaliador configure a Conect4 com um tempo de 2,055s, o nível ideal para K será 5 resultando na queda do MTT. De modo geral, caso o avaliador configure o Conect4 para 6,166s, o nível ideal de K continuará sendo 5.

A Figura 22e mostra a interação entre os fatores C e Dump4. Se o avaliador definir o tempo de Dump4 como 8. 221s, o nível ideal para C será 3. O avaliador deve considerar o C ao definir o tempo atribuído a Dump4. Se houver mais C, o tempo alocado para realizar o Dump4 pode ser maior, resultando em um impacto mínimo no MTT. A Figura 22f apresenta a interação entre C e Copy4. Existe pouca interação entre os fatores. A combinação que leva ao menor MTT é alcançada com um Copy4 de 2,033s e um C de 3,

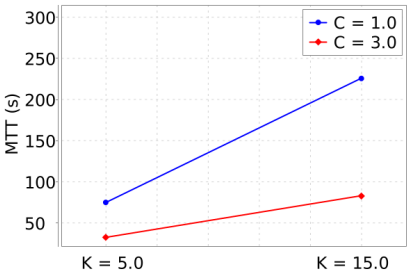
resultando em um MTT de ≈ 51,0s. No entanto, é importante ressaltar que a variação de Copy4 = 6,101s tem um impacto ligeiramente maior em relação a Copy4 = 2,033s em relação ao C. Caso o avaliador escolha o C = 3, a diferença de tempo é de apenas 6 s, diferente do C = 1, em que o tempo de diferença chega a 23s.

A Figura 22g detalha a interação entre os fatores C e Conect4. Se o avaliador definir C = 1, o nível ideal para Conect4 será 2,055s. No entanto, se o C = 3, a diferença no MTT não é significativa, e o avaliador pode escolher qualquer uma das configurações. A Figura 22h mostra a interação entre Dump4 e Copy4. Considere os níveis 2,033s e 6,101s de Copy4 em relação ao Dump4. Não existe interação entre estes dois fatores. O resultado sugere que se o avaliador adotar um C = 3, o MTT será o mesmo caso o Copy4 seja de 2,033s e 6,101s. A mesma situação acontece quando o avaliador adota um C = 1. Os fatores K e Dump4, Copy4 e Conect4 apresentados nas Figuras 22i, j) também não tiveram interação e assim podem receber menos atenção do avaliador.

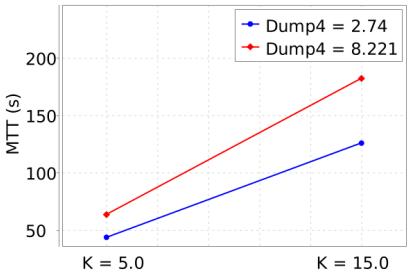
5.5 Conclusão do Capítulo

Este Capítulo apresentou um modelo de Rede de Petri Estocástica (SPN) com estado absorvente para analisar o desempenho de diferentes estratégias de migração de contêineres, incluindo Cold, PreCopy, PostCopy e Hybrid. O modelo foi validado através de experimentos em um cenário real, onde o Tempo Total de Migração (MTT) calculado pelo modelo foi comparado com o MTT observado nos experimentos. Os experimentos focaram em dois parâmetros principais: a quantidade de elementos a serem migrados e a capacidade de migração paralela. Os resultados mostraram que o tempo de migração aumenta proporcionalmente à quantidade de elementos migrados, especialmente quando há um limite paralelo de migrações igual a 2. Além disso, a política Hybrid apresentou o maior MTT, enquanto a política Cold teve um MTT menor em relação às outras políticas. A análise também considerou o aumento da capacidade de migrações paralelas. Foi observado que o MTT diminui até se estabilizar para as quatro políticas quando a capacidade de migração paralela é maior ou igual a 13. Isso sugere que aumentar a capacidade de migração paralela além de 13 tem um impacto mínimo no MTT.

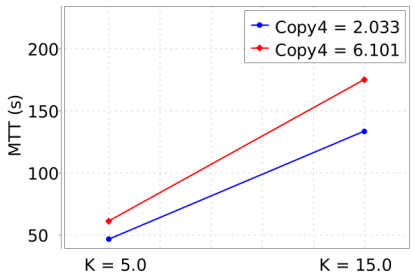
Além disso, a análise da Cumulative Distribution Function (CDF) revelou que a probabilidade de finalização da migração é maior para a política Cold, seguida pela PostCopy, PreCopy e, por último, a Hybrid. Isso sugere que a política Cold pode ser a mais eficiente em termos de tempo de migração. Finalmente, a análise do Design of Experiments (DoE) mostrou que o fator K (quantidade de elementos a serem migrados) e a interação entre C (capacidade de migração paralela) e K têm um impacto significativo no MTT. Esses resultados podem orientar a escolha de estratégias de migração eficientes em cenários reais, levando em consideração tanto a quantidade de elementos a serem migrados quanto



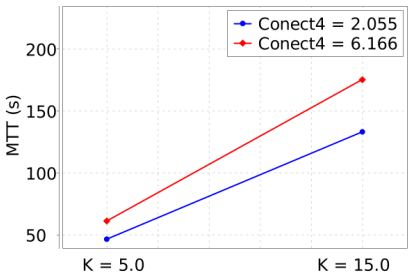
(a) K vs. C



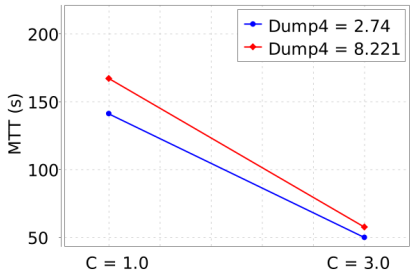
(b) K vs. Dump4



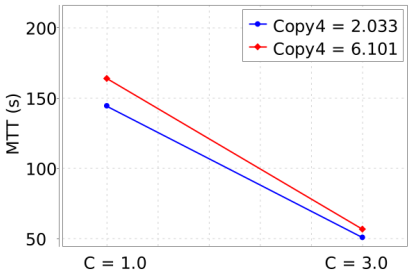
(c) K vs. Copy4



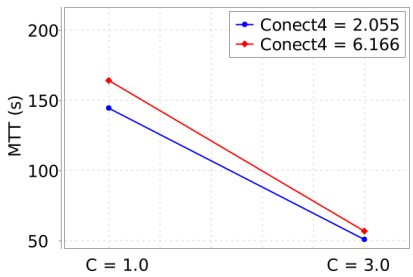
(d) K vs. Conect4



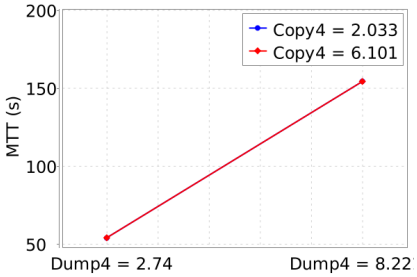
(e) C vs. Dump4



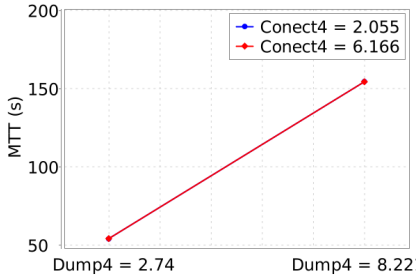
(f) C vs. Copy4



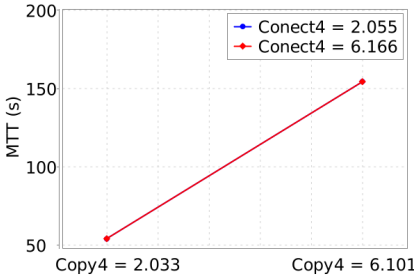
(g) C vs. Conect4



(h) Dump4 vs. Copy4



(i) Dump4 vs. Conect4



(j) Copy4 vs. Conect4

Figura 22 – Análise de sensibilidade em relação à interação entre fatores.

a capacidade de migração paralela. Após apresentar o Modelo SPN com estado Absorvente, validação, estudo de caso e DoE, o próximo Capítulo se concentra em apresentar o Modelo SPN sem estado Absorvente e seu respectivo estudo de caso.

6 Modelo SPN Não Absorvente

Cada política de migração de contêiner tem suas próprias regras, etapas de cópia e processamento. A escolha da política de migração de contêiner é complexa, pois diversos fatores podem influenciar a migração. Esse Capítulo propõe um modelo de redes de Petri estocásticas (SPN) sem estado absorvente para representar e calcular quatro métricas: o tempo médio de migração (MMT), a probabilidade de descarte, utilização e taxa de migração. O modelo representam e avaliam quatro políticas de migração de contêiner apresentadas no Capítulo 2, variando o *arrival delay* (AD). Dessa forma, administradores de *data centers* poderão planejar melhor o processo de migração mesmo em estágios iniciais de design da arquitetura computacional a ser implantada.

A Figura 23 representa o modelo SPN sem estado absorvente que permite calcular mais métricas de interesse. Um modelo sem estado absorvente é aquele que recebe novas requisições de forma constante de forma ininterrupta obedecendo um tempo entre chegadas AD. O Nó de origem é representada por uma transição de cor cinza de distribuição determinística, ou seja, a cada intervalo de tempo é iniciada a migração de um novo contêiner. O comportamento do modelo segue o mesmo fluxo do modelo com estado absorvente.

Um ponto que deve ser observado e que inclusive difere do modelo absorvente é que nos lugares P2, P6, P8 e P14, o token pode seguir por dois caminhos, no qual se o token for pelas transições imediatas TI1, TI2, TI3 ou TI4, indica que o processo de migração do contêiner falhou ao ser copiado do Nó_A para o Nó_B. Porém, se o token seguir por TI5, TI6, TI7 ou TI8, isso indica que o processo de migração prosseguiu sem falha na etapa de cópia.

6.1 Métricas

O tempo médio de resposta (MRT) pode ser obtido a partir da Lei de Little (JAIN, 1990). Esta lei requer um sistema estável, ou seja, que possua uma taxa de requisições menor que a taxa de processamento dos servidores. A lei indica que o MRT é dado pela multiplicação da quantidade de requisições dentro do sistema pelo tempo entre chegadas (AD). No modelo o tempo entre chegadas reside na transição de nome AD, localizada na extremidade esquerda do modelo. Nesta dissertação as requisições são os contêineres e chamados o MRT de *Mean Migration Time* (MMT). Assim, nesta dissertação, o número de elementos no sistema é a soma de tokens em todos os lugares por onde passa o contêiner. Por ter um único lugar de capacidade (Cap), para nosso modelo esta soma pode ser facilmente obtida por $C - Esp\{Cap\}$, onde C é a quantidade total máxima de migração

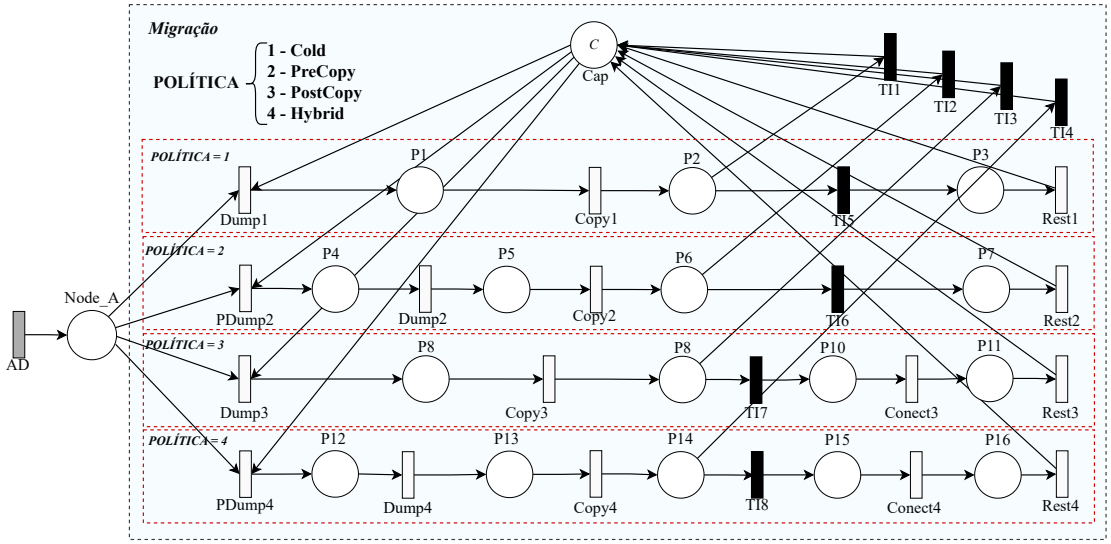


Figura 23 – Modelo não absorvente

paralela e $Esp\{Cap\}$ representa o número de tokens no lugar Cap naquele momento. $Esp\{Nomedolocal\}$ representa a esperança estatística de existir tokens em "nomedolocal", onde $Esp\{Nomedolocal\} = (\sum_{i=1}^{n} P(m(Local) = i) \times i)$, sendo n o maior número de tokens que o $Local$ pode conter. Em outras palavras, $Esp\{Nomedolocal\}$ indica o valor esperado de tokens naquele Local em determinado momento ou estacionariamente. Portanto, a equação correspondente à Lei de Little para MMT utilizada no nosso modelo é expressa na Equação 6.1.

$$\mathbf{MMT} = (C - Esp\{Cap\}) \times AD \tag{6.1}$$

A Equação 6.2 define a probabilidade de haver perdas de requisições (DP_PROB). Para calcular o descarte é necessário não restar mais nenhuma capacidade de enfileiramento na entrada do sistema. $P(Local = n)$ calcula a probabilidade de existirem n tokens em "Local". Portanto, DP_PROB considera não ter mais capacidade disponível ($\#Cap = 0$) e ter tokens aptos a entrar no processo de migração ($\#Node\_A > 0$).

$$\mathbf{DP\_PROB} = P\{(\#Cap = 0) AND (\#Node\_A > 0)\} \tag{6.2}$$

A utilização é a divisão do número esperado de tokens em um local (por onde passam os tokens executados) pela respectiva capacidade total. A utilização média do processamento da etapa de migração é dada pela Equação 6.3, lembrando que $C - Esp\{Cap\}$ retorna para este modelo o número de elementos dentro do sistema.

$$\mathbf{U} = \frac{C - Esp\{Cap\}}{C} \tag{6.3}$$

A taxa de migração (MR) é dada pela soma das vazões de cada política. Assim, quando uma política for habilitada, a vazão das demais é nula. O cálculo da vazão de um par lugar+transição é dado pela divisão do número de tokens naquele lugar ($Esp\{lugar\}$) pelo tempo daquela transição ($T\{transicao\}$). Assim MR é dada pela equação 6.4.

$$\mathbf{MR} = \frac{Esp\{P3\}}{T\{Rest1\}} + \frac{Esp\{P7\}}{T\{Rest2\}} + \frac{Esp\{P11\}}{T\{Rest3\}} + \frac{Esp\{P16\}}{T\{Rest6\}} \tag{6.4}$$

6.2 Estudo de Caso

A Figura 24 apresenta o resultado do Tempo Médio de Migração (MMT) em função da variação da Taxa de Chegada (AR). Foram testadas os seguintes valores: AR = [0,005, 0,043, 0,082, ... , 0,350] e capacidade paralela de migração igual a 2 (C=2). Para a condução desse caso de uso, a Probabilidade de Falha está igual a 0,0001.

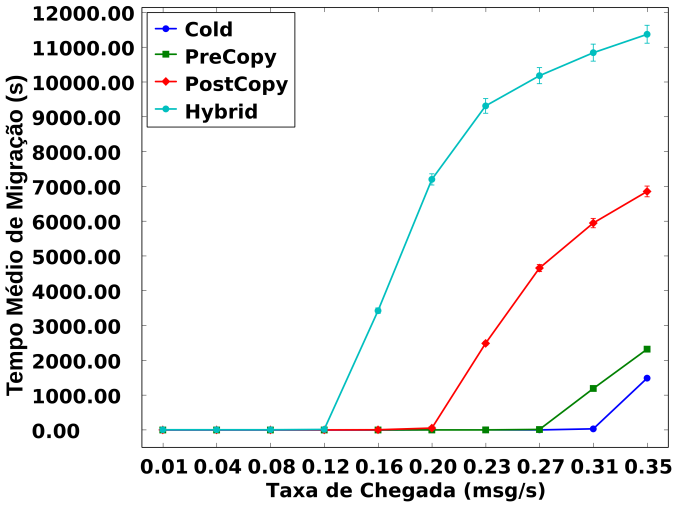


Figura 24 – Tempo Médio de Migração.

Durante o começo do experimento, onde o AR = 0,005, todas as políticas possuem um valor de MTT próximo, para política Cold e PreCopy ≈ 6,9s, seguido por PostCopy com ≈ 9,8s e Hybrid com ≈ 15,0s. O MMT segue com esses valores até AR = 0,158, nesse ponto a política Hybrida tem um aumento significativo do MMT chegando a ≈ 3434,0s. As outras políticas seguem sem grandes alterações no MMT até AR = 0,235, quando a política PostCopy chega a um MMT de ≈ 2496,0s. No final do experimento, quando o AR = 0,35 a política Cold possui um MMT menor em relação as demais, chegando a ≈ 1496,0s, comparado com a política Hybrida a que teve um MMT maior chegando a ≈ 11379,0s uma diferença de 9883,0s. Portanto, caso o AR seja inferior a AR = 0,158 a escolha da política não é relevante em relação ao MMT, porém quando o AR é superior a 0,35, vale considerar a escolha da política Cold para o processo de migração.

A Figura 25 apresenta o resultado da Probabilidade de Descarte (PD) em função da variação da Taxa de Chegada (AR). Durante o início do experimento, onde AR = 0,005, todas as políticas começam com um valor de PD próximo a zero. O PD permanece próximo a esses valores até AR = 0,081, quando a política Hybrid tem um aumento significativo da probabilidade de descarte, chegando a aproximadamente 29,9%. As outras políticas seguem sem grandes alterações do PD até AR = 0,158, quando todas as políticas têm um aumento significativo da probabilidade de descarte, chegando a ≈ 18%. Nesse ponto, a política Hybrid já está com 90% de PD. Um ponto interessante é que quando o AR é igual e superior a 0,196, a política PostCopy tem a probabilidade de descarte estabilizada em ≈ 23%, seguindo até o final do experimento quando o AR = 0,35. No final do experimento a política PostCopy possui um PD menor em relação às demais políticas, chegando a 24%, comparado com a política Hybrid que teve o maior probabilidade de descarte, chegando a 91%, uma diferença de 64% em relação à política PostCopy. Portanto, caso o AR seja inferior a AR = 0,005, a escolha da política não é relevante, pois todas as políticas têm a probabilidade de descarte igual a zero. No entanto, quando o AR é superior a 0,120, vale a pena considerar a escolha das demais políticas. Caso o AR seja igual a 0,350, a política de migração com a menor probabilidade de descarte é a PostCopy.

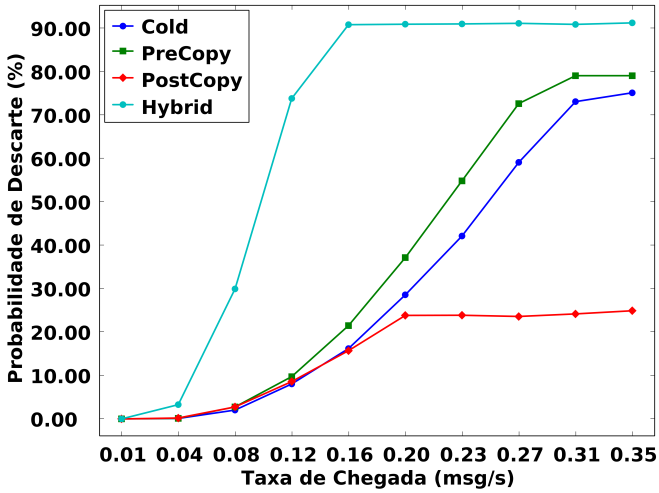


Figura 25 – Probabilidade de Descarte.

A Figura 26 apresenta o resultado da utilização do sistema em função do AR. No início do experimento, onde AR = 0,005, todas as políticas começam com a utilização superior a 25%. A utilização da política Hybrid mostra um aumento rápido na utilização e atinge rapidamente 90% em AR = 0,12. As políticas Cold e PreCopy tiveram um aumento mais gradual na utilização durante o experimento. A PostCopy mostra uma estabilização em torno de ≈ 59% de utilização para AR superior a 0,196. No final do experimento, onde o AR = 0,35, a política PostCopy tem a menor utilização, chegando a 58,7%, enquanto a política Hybrid tem a maior utilização, chegando a 90,4%. Portanto, caso a Taxa de Chegada seja igual a 0,158, o avaliador pode considerar a escolha das

políticas Cold, PreCopy e PostCopy, pois a utilização das três políticas é ≈ 50%. Caso o avaliador considere um AR = 0,35, a política PostCopy demonstrou uma utilização de 58% dos recursos, 32% a menos que a política Hybrid.

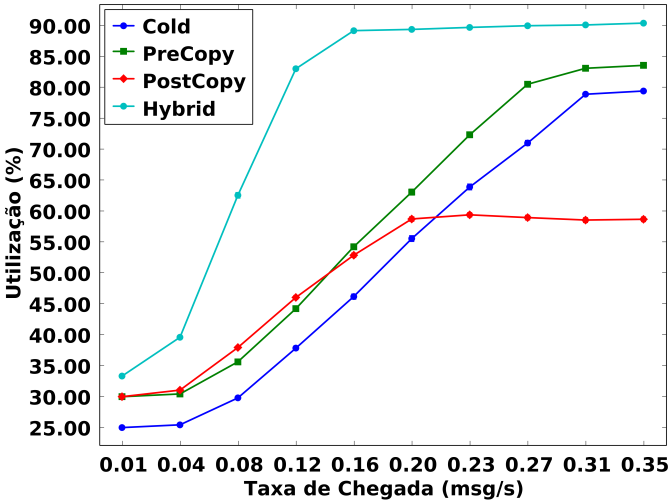


Figura 26 – Utilização do Sistema.

A Figura 27 apresenta o resultado da Taxa de Migração (MR) em função da variação da probabilidade de falha durante o processo de cópia, incluindo cenários extremos com 0,9 de probabilidade de falha. Foram testadas os seguintes valores: Probabilidade = [0.1, 0.2, 0.3, ..., 0.9] e capacidade paralela de migração igual a 2 (C=2). Durante o começo do experimento, onde a Probabilidade = 0,1, a política Cold possui a maior taxa de migração, chegando a ≈ 2,80s, seguido por PostCopy com ≈ 1,80s, PreCopy com ≈ 1,60s e Hybrid com ≈ 1,20s. A taxa diminui conforme o Probabilidade de Falha aumenta, podemos observar quando a Probabilidade de Falha chega a 0,5 que as políticas Hybrid e PreCopy estão com a Taxa de Migração bem próximas, tendo apenas uma diferença de 0,10s. Quando a Probabilidade de Falha chega a 0,7, a Taxa de migração chega a ser a mesma para as políticas de migração Cold e PostCopy, com uma Taxa de 0,95s. Por fim, quando a Probabilidade de Falha chega a 0,9, a Taxa de Migração das quatro políticas estão bem próximas, chegando a ≈ 0,20s. Portanto, caso a Probabilidade de Falha seja baixo, a escolha da política influência diretamente na Taxa de Migração, porém se o Probabilidade de Falha for alto, a escolha da política não se torna um fator relevante, pois a Taxa de Migração será semelhante nas quatro políticas do estudo de caso.

6.3 Conclusão do Capítulo

Em conclusão, o modelo de redes de Petri estocásticas (SPN) sem estado absorvente proposto neste Capítulo proporciona calcular métricas fundamentais no processos de migração. As políticas de migração Cold, PreCopy, PostCopy e Hybrid foram avaliadas

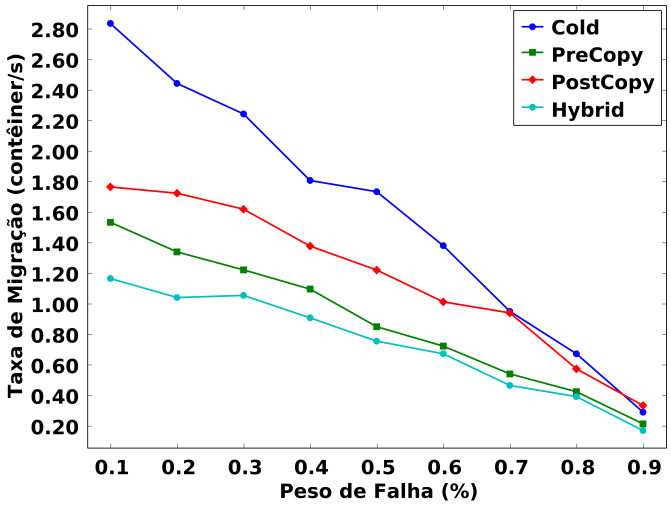


Figura 27 – Taxa de Migração.

em diferentes cenários, variando a taxa de chegada (AR) e a probabilidade de falha. Os resultados indicam que a escolha da política de migração depende fortemente do AR. Para AR inferior a 0,158, a escolha da política não é relevante em relação ao tempo médio de migração (MMT). No entanto, para AR superior a 0,35, a política Cold é a mais eficiente.

Em relação à probabilidade de descarte (PD), a política PostCopy se destaca quando o AR é igual ou superior a 0,196, mantendo a PD estável em aproximadamente 23%. Para AR inferior a 0,005, todas as políticas apresentam PD igual a zero, tornando a escolha da política irrelevante. A utilização do sistema também é um fator importante na escolha da política de migração. A política Hybrid atinge rapidamente 90% de utilização em AR = 0,12, enquanto as políticas Cold e PreCopy mostram um aumento mais gradual. A política PostCopy, por outro lado, estabiliza em torno de 59% de utilização para AR superior a 0,196. Finalmente, a taxa de migração (MR) é fortemente influenciada pela probabilidade de falha durante o processo de cópia. A política Cold apresenta a maior taxa de migração quando a probabilidade de falha é baixa (0,1), mas todas as políticas convergem para uma taxa de migração próxima quando a probabilidade de falha é alta (0,9). Após apresentar o Modelo SPN sem Estado Absorvente e seu estudo de caso, o próximo Capítulo mostra as conclusões do trabalho e as propostas de trabalhos futuros.

7 Conclusão

Esta dissertação propõe dois modelos de migração de contêiner entre dois hosts para avaliar as distintas políticas de migração para auxiliar administradores de infraestruturas computacionais na escolha de políticas de migração de contêineres que atendam aos seus requisitos não funcionais, incluindo o tempo total de migração (MTT), tempo médio de migração (MMT), probabilidade de descarte, utilização e taxa de migração. Os modelos exploram a variação da quantidade de elementos a serem migrados, capacidade de migração paralela do sistema, probabilidade de falha e taxa de chegada. O modelo absorvente permite observar o CDF e o MTT. O modelo sem estado absorvente permite observar o MMT, probabilidade de descarte, utilização e taxa de migração. A validação permite ter uma confiança maior que o modelo está consistente e que pode calcular corretamente as métricas de interesse. A validação permitiu observar com um certo grau de confiança estatística que o resultado do modelo é semelhante ao resultado do experimento e, portanto, o modelo pode ser utilizado para os devidos fins propostos. Os modelos propostos são essenciais para analistas de sistemas preverem o tempo de migração em vários cenários, especialmente quando a infraestrutura para testes reais não está disponível.

Contudo, o modelo absorvente permitiu observar que o MTT aumenta conforme a quantidade de elementos migrados aumenta. A política Cold apresentou o menor MTT em comparação com as outras políticas testadas, enquanto a Hybrid teve o maior MTT. À medida que a quantidade de elementos migrados aumenta, as diferenças entre os MTTs das políticas se tornam mais pronunciadas. Portanto, a escolha da política de migração torna-se crucial, pois a quantidade de elementos migrados impacta significativamente o MTT final. Em relação ao aumento da capacidade de migrações paralelas (C) revela que todas as quatro políticas de migração inicialmente apresentam tempos consideráveis, com a Hybrid mostrando o pior desempenho. Conforme C aumenta, o MTT diminui e eventualmente se estabiliza para valores de C maiores ou iguais a 13, indicando uma baixa alteração adicional no desempenho. Embora as linhas dos gráficos se aproximem com o aumento de C, elas nunca se tocam, ao contrário do que acontece com a variação de K. Isso sugere que o impacto da variação de C é maior do que o impacto da variação de K. Assim, conhecer o limite de C que não melhora mais o desempenho pode resultar em uma economia significativa de recursos. A análise do CDF em relação ao MTT mostra que a probabilidade de conclusão da migração varia entre as políticas de migração testadas. Em geral, a probabilidade de conclusão é maior para Cold, seguida por PostCopy, PreCopy e, por último, Hybrid. A escolha da política de migração depende da importância do requisito de tempo de migração. Para requisitos críticos a política Cold é a escolha mais segura, enquanto para requisitos menos críticos, PostCopy pode ser considerada, mesmo com uma

probabilidade de conclusão menor.

Com o modelo não absorvente se pode observar que a escolha da política de migração tem um impacto direto no MMT, especialmente em ambientes com Taxas de Chegada mais altas. A política Cold demonstrou consistentemente o menor MMT em comparação com as outras políticas, enquanto a Hybrid exibiu os valores mais altos de MMT. Portanto, ao selecionar a política de migração, é crucial considerar a relação entre AR e MMT para otimizar o desempenho do sistema. A escolha da política de migração tem um impacto direto na Probabilidade de Descarte em relação à Taxa de Chegada (AR). A política PostCopy apresenta consistentemente a menor Probabilidade de Descarte, tornando-se a escolha mais eficiente, especialmente em situações de alta demanda. Assim, a seleção da política de migração é crucial para otimizar o desempenho do sistema. A política Hybrid demonstrou um rápido aumento na utilização, atingindo 90% em um AR de 0,12. Outro fato interessante foi a política PostCopy que estabilizou em torno de 59% de utilização para AR superior a 0,196. Portanto, caso a Probabilidade de Falha seja baixo, a escolha da política influência diretamente na Taxa de Migração, porém se o Probabilidade de Falha for alto, a escolha da política não se torna um fator relevante, pois a Taxa de Migração será semelhante nas quatro políticas do estudo de caso. Para a análise de sensibilidade, dois fatores importantes foram identificados: a quantidade de contêineres a serem migrados e a capacidade de migração paralela. Caso o avaliador tenha que escolher a migração única, onde um contêiner é migrado por vez, o resultado sugere adotar o mínimo possível de elementos para atingir um menor MTT. Com a migração de três contêineres simultaneamente, observa-se um pequeno aumento do MTT ao aumentar a quantidade de elementos a serem migrados. Este trabalho pode ser útil para projetistas de sistemas neste contexto para definir melhor as configurações do sistema de migração de contêineres. Finalmente, esta investigação proporcionou mais um passo no amadurecimento da migração de contêineres.

7.1 Trabalhos Futuros

A seguir listamos alguns possíveis trabalhos futuros:

- *Estudo de políticas de migração adicionais:* Incluir novas políticas de migração além das já estudadas (Cold, PostCopy, PreCopy e Hybrid).
- *Análise de desempenho em diferentes infraestruturas:* Avaliar o desempenho das políticas de migração em diferentes tipos de infraestruturas, como nuvens privadas, públicas e híbridas.
- *Impacto da latência da rede:* Avaliar a latência da rede afeta o tempo de migração e a probabilidade de descarte.

- *Migração de contêineres em larga escala:* Apresentar uma análise da migração de contêineres em larga escala, envolvendo centenas ou milhares de contêineres.
- *Impacto do tamanho do contêiner:* Avaliar a influência do tamanho do contêiner no processo migração.

8 Publicações Durante o Mestrado

No início do mestrado, a pesquisa estava concentrada em blockchain, com várias investigações sendo realizadas nessa área. No entanto, ao longo do mestrado, houve uma mudança significativa no foco da pesquisa. O tema passou a ser a migração de contêiner, que se tornou o principal objeto de estudo.

FEITOSA, Leonel; REGO, Paulo AL; SILVA, Francisco Airton. Avaliação de Desempenho de Migração ao Vivo de Contêineres com Redes de Petri Estocásticas. In: **Anais do XXIV Workshop de Testes e Tolerância a Falhas.** SBC, 2023. p. 94-107.

FEITOSA, Leonel ; DANTAS, Jamilson Ramalho; SILVA, Francisco Airton. Blockchain as a service environment: **a dependability evaluation. The Journal of Supercomputing**, p. 1-25, 2023.

SILVA, Francisco A.; GONÇALVES, Glauber D.; FÉ, Iure; FEITOSA, Leonel; SOARES, André. Avaliação de Desempenho de Blockchains Permissionadas Hyperledger Orientada ao Planejamento de Capacidade de Recursos Computacionais. In: **Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. SBC**, 2023. p. 71-84.

SOUSA, Rubenilson.; CRISTIAN, Leonardo.; FEITOSA, Leonel.; CHOI, Eunmi.; NGUYEN, Tuan Anh.; MIN, Dugki.; SILVA, Francisco Airton. Performability Evaluation and Sensitivity Analysis of a Video Streaming on Demand Architecture. **Applied Sciences**, v. 13, n. 2, p. 998, 2023.

SILVA, Francisco Airton.; FÉ, Iure.; BRITO, Carlos.; ARAÚJO, Gabriel.; FEITOSA, Leonel.; CHOI, Eunmi.; NGUYEN, Tuan Anh.; MIN, Dugki. Supporting availability evaluation of a smart building monitoring system aided by fog computing. **Electronics Letters**, v. 58, n. 12, p. 471-473, 2022.

GOMES, Rayner.; VIEIRA, Dario.; CASTRO, Miguel.; SILVA, Francisco Airton .; FEITOSA, Leonel. Parallel Differential Evolution Meta-Heuristics and Modelling for Network Slicing in 5G Scenarios. In: **The Eighteenth International Conference on Networking and Services-ICNS**. 2022.

ARAÚJO, Gabriel.; BRITO, Carlos.; FEITOSA, Leonel.; NGUYEN, Tuan Anh.; LEE, Jae Woo.; SILVA, Francisco Airton. Mobile Games at the Edge: A Performance Evaluation to Guide Resource Capacity Planning. In: **PROCEEDINGS OF THE 12TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING AND SERVICES SCIENCE (CLOSER)**. 2022. p. 238-245.

Referências

ABDULLAH, D. B.; HADEED, W. et al. Container live migration in edge computing: a real-time performance amelioration. *International Journal of Applied Science and Engineering*, Chaoyang University of Technology, v. 19, n. 3, p. 1–8, 2022. 27, 30

AL-DHURAIBI, Y. et al. Autonomic vertical elasticity of docker containers with elasticdocker. In: IEEE. *2017 IEEE 10th international conference on cloud computing (CLOUD)*. [S.l.], 2017. p. 472–479. 25, 30

ANDERSON, C. Docker [software engineering]. *Ieee Software*, IEEE, v. 32, n. 3, p. 102–c3, 2015. 11, 13

BACCARELLI, E.; SCARPINITI, M.; MOMENZADEH, A. Fog-supported delay-constrained energy-saving live migration of vms over multipath tcp/ip 5g connections. *IEEE Access*, IEEE, v. 6, p. 42327–42354, 2018. 30

BAUKES, M. *LXC vs Docker: Why Docker is Better*. [S.l.]: October, 2019. 12, 13

BAUSE, F.; KRITZINGER, P. S. *Stochastic petri nets*. [S.l.]: Vieweg Wiesbaden, 2002. v. 1. 21

BENJAPONPITAK, T.; KARAKATE, M.; SRIPANIDKULCHAI, K. Enabling live migration of containerized applications across clouds. In: IEEE. *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. [S.l.], 2020. p. 2529–2538. 25, 30

BERNSTEIN, D. Containers and cloud: From lxc to docker to kubernetes. *IEEE cloud computing*, IEEE, v. 1, n. 3, p. 81–84, 2014. 12

BHARDWAJ, A.; KRISHNA, C. R. A container-based technique to improve virtual machine migration in cloud computing. *IETE Journal of Research*, Taylor & Francis, v. 68, n. 1, p. 401–416, 2022. 25, 30

BLENK, A. et al. Survey on network virtualization hypervisors for software defined networking. *IEEE Communications Surveys & Tutorials*, IEEE, v. 18, n. 1, p. 655–685, 2015. 11

BOLCH, G. et al. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. [S.l.]: John Wiley & Sons, 2006. 38

BUI, T. Analysis of docker security. *arXiv preprint arXiv:1501.02967*, 2015. 10

BURNS, B. et al. *Kubernetes: up and running*. [S.l.]: "O'Reilly Media, Inc.", 2022. 3

CAMPOLONGO, F.; TARANTOLA, S.; SALTELLI, A. Tackling quantitatively large dimensionality problems. *Computer Physics Communication*, Institute Jbr Systems, Informatics and Safety. Joint Research Centre of the European Commission, v. 117, n. 1, p. 75–85, 1999. 22

CARVALHO, D. et al. Mobile edge computing performance evaluation using stochastic petri nets. In: IEEE. *2020 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.], 2020. p. 1–6. 46

CHOU, C. C. et al. Optimizing post-copy live migration with system-level checkpoint using fabric-attached memory. In: IEEE. *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC)*. [S.l.], 2019. p. 16–24. 26, 30

CONFORTI, L. et al. Extending the quic protocol to support live container migration at the edge. In: IEEE. *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. [S.l.], 2021. p. 61–70. 3

DAS, R.; SIDHANTA, S. Live migration of containers in the edge. *SN Computer Science*, Springer, v. 4, n. 5, p. 479, 2023. 27, 30

DAYO, A. O. A multi-containerized application using docker containers and kubernetes clusters. *Int J Comput Appl*, v. 183, n. 44, p. 55–60, 2021. 41

DESHPANDE, L.; LIU, K. Edge computing embedded platform with container migration. In: IEEE. *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. [S.l.], 2017. p. 1–6. 14

DI, Z.; SHAO, E.; TAN, G. High-performance migration tool for live container in a workflow. *International Journal of Parallel Programming*, Springer, v. 49, p. 658–670, 2021. 26, 30

DU, J. et al. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, IEEE, v. 66, n. 4, p. 1594–1608, 2017. 5

FAN, W. et al. A live migration algorithm for containers based on resource locality. *Journal of Signal Processing Systems*, Springer, v. 91, p. 1077–1089, 2019. 25, 30

FEITOSA, L. et al. Performance evaluation of message routing strategies in the internet of robotic things using the d/m/c/k/fcfs queuing network. *Electronics*, MDPI, v. 10, n. 21, p. 2626, 2021. 46

FERNÁNDEZ-CARAMÉS, T. M. et al. A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard. *Sensors*, MDPI, v. 18, n. 6, p. 1798, 2018. 5

GERMAN, R. *Performance analysis of communication systems with non-Markovian stochastic Petri nets*. [S.l.]: John Wiley & Sons, Inc., 2000. 20, 21, 40

GONÇALVES, I. et al. Surveillance system in smart cities: a dependability evaluation based on stochastic models. *Electronics*, MDPI, v. 10, n. 8, p. 876, 2021. 46

GONZÁLEZ, A. E.; ARZUAGA, E. Herdmonitor: monitoring live migrating containers in cloud environments. In: IEEE. *2020 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2020. p. 2180–2189. 27, 30

GOVINDARAJ, K.; ARTEMENKO, A. Container live migration for latency critical industrial applications on edge computing. In: IEEE. *2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA)*. [S.l.], 2018. v. 1, p. 83–90. 26, 30

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: John Wiley & Sons, 1990. 40, 53

JIANG, Y.; HUANG, Z.; TSANG, D. H. Challenges and solutions in fog computing orchestration. *IEEE Network*, IEEE, v. 32, n. 3, p. 122–129, 2017. 4

JUNIOR, P. S.; MIORANDI, D.; PIERRE, G. Stateful container migration in geo-distributed environments. In: IEEE. *2020 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. [S.l.], 2020. p. 49–56. 3

KAKAKHEL, S. R. U. et al. Virtualization at the network edge: A technology perspective. In: IEEE. *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. [S.l.], 2018. p. 87–92. 27, 30

KARHULA, P.; JANAK, J.; SCHULZRINNE, H. Checkpointing and migration of iot edge functions. In: *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking*. [S.l.: s.n.], 2019. p. 60–65. 28, 30

KAUR, K.; GUILLEMIN, F.; SAILHAN, F. Container placement and migration strategies for cloud, fog, and edge data centers: A survey. *International Journal of Network Management*, Wiley Online Library, v. 32, n. 6, p. e2212, 2022. 8, 10

KLEIJNEN, J. P. Sensitivity analysis and optimization in simulation: design of experiments and case studies. In: IEEE. *Winter Simulation Conference Proceedings, 1995.* [S.l.], 1995. p. 133–140. 22

KOTIKALAPUDI, S. V. N. *Comparing live migration between linux containers and kernel virtual machine: investigation study in terms of parameters*. 2017. 3, 27, 30

MA, J.-S.; KIM, H.-Y.; KIM, Y.-W. The virtualization and performance comparison with lxc-lxd in arm64bit server. In: IEEE. *2016 6th International Conference on IT Convergence and Security (ICITCS)*. [S.l.], 2016. p. 1–4. 12

MA, L. et al. Efficient live migration of edge services leveraging container layered storage. *IEEE Transactions on Mobile Computing*, IEEE, v. 18, n. 9, p. 2020–2033, 2018. 26, 30

MACHEN, A. et al. Live service migration in mobile edge clouds. *IEEE Wireless Communications*, IEEE, v. 25, n. 1, p. 140–147, 2017. 27, 30

MACIEL, P. et al. Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In: IEEE. *2017 IEEE 22nd Pacific Rim international symposium on dependable computing (PRDC)*. [S.l.], 2017. p. 50–57. 20, 29, 43

MAHESHWARI, S. et al. Traffic-aware dynamic container migration for real-time support in mobile edge clouds. In: IEEE. *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. [S.l.], 2018. p. 1–6. 4

MAJEED, A. A. et al. Modelling fog offloading performance. In: IEEE. *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*. [S.l.], 2020. p. 29–38. 27, 30

MARSAN, M. A. Stochastic petri nets: an elementary introduction. In: SPRINGER. *Advances in Petri Nets 1989 9*. [S.l.], 1990. p. 1–29. 20

MARSAN, M. A. et al. Modelling with generalized stochastic petri nets. *ACM SIGMETRICS performance evaluation review*, ACM New York, NY, USA, v. 26, n. 2, p. 2, 1998. 20, 21, 38, 40

MOLLOY. Performance analysis using stochastic petri nets. *IEEE Transactions on computers*, IEEE, v. 100, n. 9, p. 913–917, 1982. 20

MORABITO, R.; KJÄLLMAN, J.; KOMU, M. Hypervisors vs. lightweight virtualization: a performance comparison. In: IEEE. *2015 IEEE International Conference on cloud engineering*. [S.l.], 2015. p. 386–393. 10, 11

MUKUTE, T. et al. Design and implementation of multi-cloud vnfs deployment utilizing lightweight lxc virtualization. In: IEEE. *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. [S.l.], 2019. p. 1–5. 12

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989. 20

NELSON, R. *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. [S.l.]: Springer Science & Business Media, 2013. 38

NIE, H. et al. Research on optimized pre-copy algorithm of live container migration in cloud environment. In: SPRINGER. *Parallel Architecture, Algorithm and Programming: 8th International Symposium, PAAP 2017, Haikou, China, June 17–18, 2017, Proceedings 8*. [S.l.], 2017. p. 554–565. 15

PECHOLT, J.; HUBER, M.; WESSEL, S. Live migration of operating system containers in encrypted virtual machines. In: *Proceedings of the 2021 on Cloud Computing Security Workshop*. [S.l.: s.n.], 2021. p. 125–137. 26, 30

PICKARTZ, S. et al. Migrating linux containers using criu. In: SPRINGER. *High Performance Computing: ISC High Performance 2016 International Workshops, ExaComm, E-MuCoCoS, HPC-IODC, IXPUG, IWOPH, P^ 3MA, VHPC, WOPSSS, Frankfurt, Germany, June 19–23, 2016, Revised Selected Papers 31*. [S.l.], 2016. p. 674–684. 4

PINHEIRO, T. et al. Performance and data traffic analysis of mobile cloud environments. In: IEEE. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.], 2018. p. 4100–4105. 38

PIRES, A.; SIMÃO, J.; VEIGA, L. Distributed and decentralized orchestration of containers on edge clouds. *Journal of Grid Computing*, Springer, v. 19, p. 1–20, 2021. 19

PULIAFITO, C.; MINGOZZI, E.; ANASTASI, G. Fog computing for the internet of mobile things: issues and challenges. In: IEEE. *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. [S.l.], 2017. p. 1–6. 4

PULIAFITO, C. et al. Container migration in the fog: A performance evaluation. *Sensors*, MDPI, v. 19, n. 7, p. 1488, 2019. 16

PULIAFITO, C. et al. Design and evaluation of a fog platform supporting device mobility through container migration. *Pervasive and Mobile Computing*, Elsevier, v. 74, p. 101415, 2021. 17

PUTHAL, D. et al. Secure and sustainable load balancing of edge data centers in fog computing. *IEEE Communications Magazine*, IEEE, v. 56, n. 5, p. 60–65, 2018. 6

RAMANATHAN, S. et al. Live migration of virtual machine and container based mobile core network components: A comprehensive study. *IEEE Access*, IEEE, v. 9, p. 105082–105100, 2021. 26, 30

RAMANATHAN, S. et al. A comprehensive study of virtual machine and container based core network components migration in openroadm sdn-enabled network. *arXiv preprint arXiv:2108.12509*, 2021. 25, 30

SANTOS, B. et al. Iot sensor networks in smart buildings: A performance assessment using queuing models. *Sensors*, MDPI, v. 21, n. 16, p. 5660, 2021. 46

SANTOS, L. et al. Data processing on edge and cloud: A performability evaluation and sensitivity analysis. *Journal of Network and Systems Management*, Springer, v. 29, n. 3, p. 1–24, 2021. 22, 46

SILVA, F. A. et al. Mobile cloud performance evaluation using stochastic models. *IEEE Transactions on Mobile Computing*, IEEE, v. 17, n. 5, p. 1134–1147, 2017. 45

SINDI, M.; WILLIAMS, J. R. Using container migration for hpc workloads resilience. In: IEEE. *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. [S.l.], 2019. p. 1–10. 14

SMIMITE, O.; AFDEL, K. Impact of hybrid virtualization using vm and container on live migration and cloud performance. In: SPRINGER. *Lecture Notes in Real-Time Intelligent Systems*. [S.l.], 2019. p. 196–208. 25, 30

STATISTA. Container technology - statistics and facts. 2023. 3

STOYANOV, R.; KOLLINGBAUM, M. J. Efficient live migration of linux containers. In: SPRINGER. *High Performance Computing: ISC High Performance 2018 International Workshops, Frankfurt/Main, Germany, June 28, 2018, Revised Selected Papers 33*. [S.l.], 2018. p. 184–193. 26, 30

TANG, Z. et al. Migration modeling and learning algorithms for containers in fog computing. *IEEE Transactions on Services Computing*, IEEE, v. 12, n. 5, p. 712–725, 2018. 4

TAY, Y.; GAURAV, K.; KARKUN, P. A performance comparison of containers and virtual machines in workload migration context. In: IEEE. *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. [S.l.], 2017. p. 61–66. 27, 30

TORRE, R. et al. Towards a better understanding of live migration performance with docker containers. In: VDE. *European Wireless 2019; 25th European Wireless Conference*. [S.l.], 2019. p. 1–6. 3, 28, 30

TOŠIĆ, A. Run-time application migration using checkpoint/restore in userspace. *arXiv preprint arXiv:2307.12113*, 2023. 18

TOŠIĆ, A. et al. A blockchain protocol for real-time application migration on the edge. *Sensors*, MDPI, v. 23, n. 9, p. 4448, 2023. 19

TRIVEDI, K. S. *Probability & statistics with reliability, queuing and computer science applications*. [S.l.]: John Wiley & Sons, 2008. 20

TURNBULL, J. *The Docker Book: Containerization is the new virtualization*. [S.l.]: James Turnbull, 2014. 3

VARASTEH, A.; GOUDARZI, M. Server consolidation techniques in virtualized data centers: A survey. *IEEE Systems Journal*, IEEE, v. 11, n. 2, p. 772–783, 2015. 3

VICTOR, C. et al. Performability assessment and sensitivity analysis of a home automation system. In: IEEE. *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. [S.l.], 2021. p. 1–4. 46

XAVIER, M. G. et al. Performance evaluation of container-based virtualization for high performance computing environments. In: IEEE. *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. [S.l.], 2013. p. 233–240. 12

XU, B. et al. Sledge: Towards efficient live migration of docker containers. In: IEEE. *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. [S.l.], 2020. p. 321–328. 25, 30

ZHU, C. et al. Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 3, p. 4150–4161, 2018. 4