**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL**
**FACULDADE DE INFORMÁTICA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

# UM ESTUDO SOBRE A CONTENÇÃO DE DISCO EM AMBIENTES VIRTUALIZADOS UTILIZANDO CONTÊINERES E SEU IMPACTO SOBRE APLICAÇÕES MAPREDUCE

**KASSIANO JOSÉ MATTEUSSI**

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. César Augusto F. De Rose

**Porto Alegre**
**2016**

# Ficha Catalográfica

M435e Matteussi, Kassiano José

Um Estudo Sobre a Contenção de Disco em Ambientes Virtualizados Utilizando Contêineres e Seu Impacto Sobre Aplicações MapReduce / Kassiano José Matteussi . – 2016.

94 f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. César Augusto Fonticella De Rose.

1. Interferência em Ambientes Virtualizados. 2. Contenção de Disco. 3. Contêineres. 4. MapReduce. 5. Big Data. I. De Rose, César Augusto Fonticella. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Um Estudo Sobre a Contenção de Disco em Ambientes Virtualizados Utilizando Contêineres e Seu Impacto Sobre Aplicações *Mapreduce*" apresentada por Kassiano José Matteussi como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 17 de março de 2016 pela Comissão Examinadora:

Prof. Dr. César Augusto Fonticielha De Rose– Orientador — PPGCC/PUCRS

Prof. Dr. Avelino Francisco Zorzo – PPGCC/PUCRS

Prof. Dr. Claudio Fernando Resin Geyer - UFRGS

Homologada em 15/09/2016, conforme Ata No. 018 pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.


**Campus Central**
Av. Ipiranga, 6681 – P. 32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

## DEDICATÓRIA

Dedico este trabalho a meu falecido pai, Ademir Matteussi!

“Without ambition one starts nothing. Without work one finishes nothing. The prize will not be sent to you. You have to win it.”
(Ralph Waldo Emerson)

## AGRADECIMENTOS

À Deus. À minha família, grato pelo imenso apoio e por sempre acreditarem em mim. Ao Professor César Augusto F. De. Rose, muito obrigado pela oportunidade que tu me proporcionaste, foi sem dúvidas a melhor experiência da minha vida até o momento. Aos amigos do PPGCC e a toda galera do LAD/PUCRS, em especial aos colegas Miguel Xavier e Marcelo Neves, obrigado pelos ensinamentos e a toda a paciência que tiveram comigo. À DELL e a PUCRS pela concessão da bolsa de estudo.

# UM ESTUDO SOBRE A CONTENÇÃO DE DISCO EM AMBIENTES VIRTUALIZADOS UTILIZANDO CONTÊINERES E SEU IMPACTO SOBRE APLICAÇÕES MAPREDUCE


## RESUMO

Aplicações Big Data precisam analisar e processar grandes quantidades de dados em paralelo, como é o caso das aplicações MapReduce que são conhecidas pela utilização intensiva de disco. Nesse contexto, observou-se a tendência em utilizar a virtualização baseada em contêineres para o processamento de aplicações de alto desempenho, como o MapReduce. Entretanto, o uso de contêineres para o processamento intensivo de dados trouxe à tona diversos problemas de interferência já resolvidos para as tecnologias tradicionais de virtualização, como por exemplo a interferência causada pela uso concorrente dos recursos de disco, problema esse também conhecido como contenção de disco. Esse trabalho apresenta um estudo experimental que investiga o uso intensivo e concorrente dos recursos de escrita em disco: em que se concentram os principais problemas de interferência de disco, responsáveis por conduzir os ambientes virtualizados que utilizem contêineres para o processamento de aplicações MapReduce à contenção de disco. Os resultados demonstram que ao utilizar políticas tanto estáticas quanto dinâmicas para o gerenciamento de recursos é possível minimizar os problemas de contenção de disco e acelerar as aplicações MapReduce processadas sobre contêineres. A métrica Makespan foi avaliada e os ganhos sobre as estratégias são de: estática 31% e a dinâmica 26%.



**Palavras-Chave:** Interferência em Ambientes Virtualizados, Contenção de Disco, Contêineres, MapReduce, Big Data.

# A STUDY OF DISK CONTENTION IN VIRTUALIZED ENVIRONMENTS USING CONTAINERS AND ITS IMPACT ON MAPREDUCE APPLICATIONS


## ABSTRACT

Big Data applications need to process and analyze large data set in parallel, such as MR applications that perform intensive disk operations. In such context, we observed a trend on use container based virtualization to process high-performance applications as MapReduce. However, the use of containers to data-intensive processing brought back many already solved interference problems in the traditional virtualized environments, such as example the interference problem caused by concurrently disk resources usage, also known as disk contention. This work shows an experimental study to investigate the intensive and concurrent usage of write resources in disk: where are concentrated the main disk interference problems, being responsible for leading the virtualized environment that uses containers to process MapReduce application until disk contention. Our result presents a static and dynamic policy for disk resource management in order to minimize the contention between MapReduce applications. Finally, the Makespan metric was evaluated and the gains over the strategies are static 31% and dynamic 26%.



**Keywords:** Interference in Virtualized Environments, Disk Contention, Containers, MapReduce, Big Data.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ALGORITMOS

# LISTA DE SIGLAS

ANS – Acordo de Nível de Serviço
BDS – Big Data Systems
BLKIO – Block I/O Controller
CGROUPS – Grupos de Controle
DFS – Distributed File System
E-COMMERCE – Comércio Eletrônico
EMR – Elastic MapReduce
E/S – Entrada/Saída
ET – Execution Time
FIO – Flexible I/O
GFS – Google File System
HD – Hard Disk
HDFS – Hadoop Distributed File System
HMR – Hadoop MapReduce
HPC – High Performance Computing
IOPS – Operações de Entrada e Saída por Segundo
LXC – Linux Containers
MBS – Mega Bytes por segundo
MS – Makespan
MR – MapReduce
MVS – Máquinas Virtuais
QOS – Qualidade de Serviço
SAS – Serial Attached SCSI
SLA – Service Level Agreement
SQL – Structured Query Language
SO – Sistema Operacional
TDB – Total Disk Bandwidth
UCP – Unidade Central de Processamento
VMM – Virtual Machine Monitor
YARN – Yet Another Resource Negotiator

# SUMÁRIO

# 1. INTRODUÇÃO

A evolução das redes sociais, redes de sensores, dispositivos móveis, e todas as recentes tecnologias da informação e comunicação têm gerado informação de forma muito veloz e em grandes quantidades, na ordem dos Terabytes até os Zettabytes! A riqueza contida nas informações obtidas através da análise e processamento Big Data potencializa a tomada de decisões estratégicas, de maneira a produzir respostas muito valiosas [30]. Por esse motivo, muitas organizações estão em busca de soluções eficientes, inteligentes, confiáveis e que ao mesmo tempo possuam baixo custo para lidar com processamento de larga escala exigidos por aplicações Big Data.

Nesse contexto, o modelo de programação paralelo e distribuído MapReduce (MR) [52] - e sua implementação mais popular o *framework open-source* Hadoop MR [52] é geralmente considerado uma boa alternativa para a execução de aplicações que efetuem a análise e processamento de dados em larga escala. Isso porque, o mesmo fornece facilidade de programação, tolerância a falhas e escalonamento de tarefas em arquiteturas escaláveis de alto desempenho, tal como *cluster* de computadores. *Clusters* permitem que múltiplas aplicações sejam executadas de forma paralela e distribuída, garantindo assim a escalabilidade, disponibilidade e sustentabilidade, além de fornecer alto poder de processamento [38].

Inicialmente aplicações MR eram executadas sequencialmente em *clusters*, ou seja, uma após a outra e sem o compartilhamento de recursos, o que propiciava a má utilização (subutilização) dos mesmos. Tal cenário, muitas vezes pode se tornar inviável devido a demanda de processamento existente, como por exemplo, a formação de filas de aplicações para o processamento em um *cluster*. Nesse contexto, o uso de virtualização emergiu como uma solução inteligente e de baixo custo, possibilitando a consolidação de máquinas físicas de forma virtual (*clusters* virtualizados) e, por conta dessa característica, torna possível o compartilhamento dos recursos virtualizados entre diversas aplicações [44].

Muito comum na computação em nuvem, os *clusters* virtuais fazem uso de serviços e aplicações sobre os tradicionais virtualizadores, tais como: ESXi da VMware [49], XEN [36] e entre outros, que possuem como finalidade gerenciar o ambiente virtual criado. Ainda, como exemplo, pode-se citar o serviço Elastic Map Reduce (EMR) [2] disponibilizado pela Amazon [1], o qual permite que empresas, pesquisadores e desenvolvedores, de forma descomplicada e econômica, processem uma grande quantidade de dados via aplicação web utilizando MR. Infelizmente, aplicações Hadoop MR executadas sobre ambientes de *cluster* virtuais são normalmente incapazes de atingir o mesmo desempenho obtido a partir de plataformas não-virtualizadas [24, 55, 54]. Isso ocorre devido à sobrecarga imposta pelos próprios virtualizadores, que possuem o Virtual Machine Monitor (VMM) ou monitor de máquina virtual sobrecarregados pela comunicação intensiva entre as máquinas virtuais

e o hardware físico. E por conta desse problema, a utilização da virtualização tem sido tradicionalmente evitada pela Computação de Alto Desempenho (CAD) [24, 54].

Não muito distante, o modelo de virtualização baseado em contêineres também conhecido como *OS-level Virtualization*, virtualização em nível de Sistema Operacional, representa a versão mais leve de virtualização, oferecendo performance semelhante a nativa, isolamento, tolerância a falhas, melhor capacidade de gerenciamento de recursos [53]. Ainda, por conta dessas características, atualmente existe a tendência natural de se utilizar contêineres para a CAD. Como exemplo dessa tendência, cita-se os atuais sistemas para o processamento e análise Big Data, são eles: YARN [17], Flink [13] e Mesos [14]. Esses sistemas geralmente utilizam contêineres como base para oferecer um ambiente com alta disponibilidade, desempenho e compartilhamento de recursos para o processamento de larga escala utilizando *clusters* virtualizados.

Adicionalmente, os *frameworks* para o processamento Big Data que suportam esses sistemas, tais como: Hadoop [52], Storm [16], Spark [15] e suas aplicações: de consultas SQL, processamento de eventos em tempo real, processamento em lote (*batch jobs*), aprendizagem de máquina e, entre outras possuem diferentes complexidades e utilização de recursos computacionais. Nesse contexto, essas aplicações compartilham o mesmo ambiente e são executadas de forma paralela, propiciando inúmeras solicitações para alocação de recursos. Esse comportamento pode causar problemas de interferência independentemente do modelo de virtualização utilizado.

A interferência diz respeito a disputa pela utilização de recursos como por exemplo: a utilização de recursos de CPU, caches compartilhados, barramento de memória, controladores de memória, rede e de escrita e leitura (E/S) em disco levando o ambiente virtualizado a um estado de contenção de recursos. Isso acarreta em falhas de execução, variações de performance entre os *frameworks* de processamento e as aplicações MR em execução sobre o mesmo *cluster* virtualizado, resultando na degradação de desempenho [62].

Muitos esforços já foram realizados para solucionar problemas de interferência relativos a contenção de recursos de CPU, memória, disco e rede em ambientes tradicionais de virtualização [63, 35, 53, 58, 37, 27]. Entretanto, preocupações relativas a interferência de recursos entre *frameworks* de processamentos e suas aplicações em ambientes compartilhados que anteriormente já estavam resolvidas para esse modelo, voltaram à tona devido a tendência quanto ao uso de contêineres para HPC. Principalmente os problemas relativos a contenção de disco, uma vez que aplicações *Big Data* são tipicamente categorizadas como intensivas de disco[21].

Dessa forma, o presente trabalho aborda essa oportunidade de pesquisa, apresentando um estudo experimental que, investigue o impacto da contenção de disco - causada principalmente pelo uso intensivo e concorrente dos recursos de escrita realizados por aplicações MR executadas sobre ambientes virtualizados que utilizem contêineres. Os resultados obtidos apresentam que ao evitar os problemas de contenção e subutilização

de recursos utilizando simples estratégias para a alocação de recursos, o desempenho das aplicações é aprimorado. Em seguida, será apresentada a motivação para essa pesquisa junto a um exemplo conceitual do problema.

## 1.1 Motivação

Aplicações Big Data precisam analisar e processar grandes quantidades de dados em paralelo, como é o caso das aplicações MapReduce que são comumente conhecidas por efetuarem operações de Leitura/Entrada e Escrita/Saída (E/S) em disco de forma intensiva sobre *clusters* virtualizados. Além disso, o processamento de Big Data geralmente é realizado sobre nuvens públicas ou privadas que devem assegurar o desempenho das aplicações e garantir os Acordos de Nível de Serviço (ANS) contratados pelos usuários [29, 59]. Ainda, embora o uso da virtualização tradicional permita o processamento de aplicações em modo compartilhado, seu desempenho não é eficaz quando comparado à virtualização baseada em contêineres. Isso porquê contêineres possuem o desempenho similar ao de um Sistema Operacional (SO) nativo e, por conta disso, pode-se visualizar uma tendência no seu uso para a CAD.

Junto a essa tendência, diversos problemas de interferência relacionados aos modelos tradicionais de virtualização anteriormente já estudados voltaram à tona. Um desses problemas é relacionado a contenção de disco, especialmente para com o recurso de escrita em disco, utilizando intensamente durante o processamento de aplicações Big Data, como por exemplo, o processamento de aplicações MapReduce. Também é sabido que, nem mesmo os atuais sistemas Big Data como Mesos, Flink e YARN que utilizam como base os contêineres, efetuam adequadamente o gerenciamento desse recurso, deixando o controle relativo a operações de escrita em disco como responsabilidade do SO [50].

Em consequência ao gerenciamento inadequado dos recursos de escrita em disco, um *cluster* virtualizado pode ser conduzido a problemas de interferência, levando o mesmo a um estado de contenção de disco. Como consequência, pode-se ocorrer perda de desempenho das aplicações em execução. Por exemplo, ao executar aplicações MR simultaneamente, como pode ser visto na Figura 1.1, inúmeras operações de escrita sobre o disco podem ser realizadas de forma concorrente, elevando a utilização desse recurso. Dessa forma, quando a capacidade do sistema for atingida devido ao uso intenso do mesmo, a (interferência) contenção de disco ocorre, levando as aplicações a demandarem maior tempo para sua execução.

Trabalhos existentes [41, 60, 61, 42, 50] buscam minimizar a contenção de disco, apresentando soluções baseadas em algoritmos que efetuam particionamento e redistribuição dos trabalhos MR, predição de desempenho, estratégias para o gerenciamento de recursos (leitura) de forma estática e limitação/priorização de trabalhos MR. Entretanto,

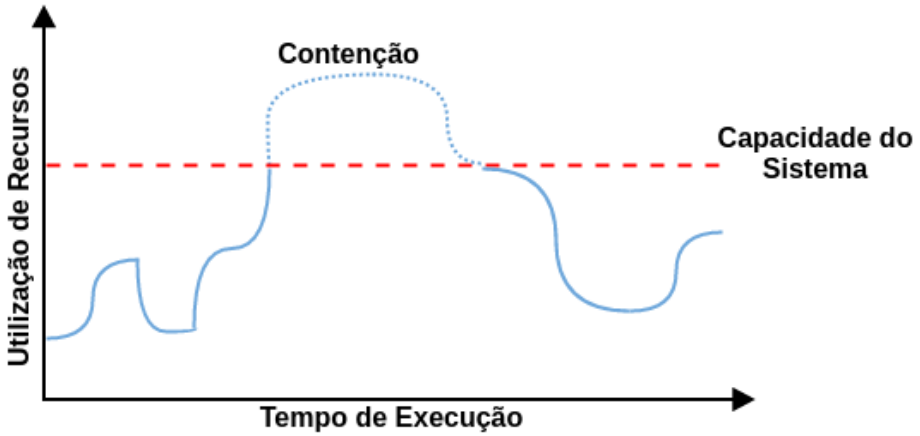



Figura 1.1 – Contenção de Recursos

essas soluções geralmente alteram o *framework* de processamento, tornando sua portabilidade/aplicabilidade complexa. Além disso, em ambientes compartilhados, as aplicações Big Data em execução desconhecem a utilização de recursos de escrita em disco uma das outras, uma vez que não existe nenhum controle sobre os processos que chegam até o SO, ou seja, atualmente o gerenciamento dos recursos de disco para com essas aplicações é inadequado.

Desse modo, observou-se uma oportunidade de pesquisa que busque oferecer soluções para o gerenciamento de recursos de escrita em disco em meio ao processamento compartilhado de aplicações Big Data sobre contêineres. Motivando assim, a proposta desse trabalho, à qual apresenta um estudo experimental que, investigue o impacto da contenção de disco - causada principalmente pelo uso intensivo e concorrente dos recursos de escrita realizados por aplicações MR executadas sobre ambientes virtualizados que utilizem contêineres.

## 1.2 Hipótese e Questões de Pesquisa

O objetivo do presente trabalho é de investigar a seguinte hipótese: É possível acelerar aplicações MR minimizando a contenção de disco em contêineres. Para conduzir essa investigação as seguintes questões de pesquisa foram definidas:

- **Q1**. Como os recursos computacionais são consumidos pelas aplicações MapReduce? O objetivo dessa questão de pesquisa é compreender e identificar as características inerentes ao consumo de recursos das aplicações MapReduce, verificando a existência de padrões e possíveis causas que podem gerar gargalos de desempenho relacionados a contenção de disco.

- **Q2**. A consolidação de aplicações MR em contêineres conduz a perda de desempenho devido a contenção de disco? O principal objetivo dessa pergunta é identificar quando os problemas de contenção em disco vão ocorrer ao executar aplicações MR sobre um *cluster* virtual utilizando contêineres.
- **Q3**. É possível efetuar restrições em disco a fim de limitar o uso desses recursos para aplicações MR em execução sobre contêineres? O principal objetivo dessa questão de pesquisa é verificar se o modelo de virtualização baseado em contêineres é capaz de lidar com o modo de escrita efetuado pelo MR. Ao responder essa questão de pesquisa será possível elaborar/aplicar políticas para efetuar o gerenciamento de recursos de escrita em disco entre as múltiplas aplicações MR em execução;
- **Q4**. É possível minimizar a contenção de disco ao efetuar a restrição de recursos por contêiner? Essa questão de pesquisa tem como objetivo demonstrar que, com o auxílio de simples políticas de gerenciamento de recursos é possível diminuir a contenção de disco e, consequentemente melhorar o desempenho de aplicações MR sobre um *cluster* virtual utilizando contêineres.

A seguir será apresentada a organização do presente trabalho.

## 1.3 Organização do Trabalho

O presente capítulo apresentou a introdução desse trabalho e sua motivação. Os próximos capítulos estão organizados da seguinte forma: Capítulo 2 e 3 apresentam o embasamento teórico deste trabalho, concentrando-se nas áreas de Big Data, virtualização e suas políticas de gerenciamento de recursos, a interferência presente sobre os ambientes virtualizados (virtualização tradicional e contêineres) e em especial os problemas de contenção de disco.

No decorrer do Capítulo 4 serão contextualizados o problema e a proposta desse trabalho, bem como será efetuado um estudo experimental que busque responder as questões pesquisa relacionadas ao mesmo. O Capítulo 5 irá apresentar as considerações finais, contribuições e os trabalhos futuros.

# 2. FUNDAMENTAÇÃO TEÓRICA

Um dos grandes desafios computacionais da atualidade se concentra em armazenar, manipular e analisar de forma veloz e à um baixo custo um volume muito grande de dados gerados (petabytes diários) por sistemas corporativos, serviços e sistemas Web, mídias sociais [52].

Esses conjuntos de dados têm se tornado uma valiosa fonte de informação que desperta o interesse pela potencialidade que podem trazer aos negócios, como exemplos podem ser citadas as corporações Google e Facebook, responsáveis por gerar e manter quantidades inimagináveis de dados oriundos de seus usuários [34]. Além disso, dados gerados e caracterizados pelo crescimento, o uso e a disponibilidade das informações, sejam elas estruturadas ou não, denomina-se Big Data [7].

No decorrer das próximas seções busca-se abordar todos os assuntos inerentes ao presente trabalho. Vão ser apresentados assuntos como o ecossistema Big Data, as tecnologias de virtualização e seus problemas de interferência, bem como as políticas de gerenciamento de recursos utilizados por esses ambientes com o objetivo de contornar possíveis problemas de desempenho.

## 2.1 Big Data e Sua Evolução

Big Data trata-se de um termo empregado para descrever o crescimento, o uso e a disponibilidade das informações, sejam elas estruturadas ou não. Big Data pode ser caracterizado por diferentes aspectos e é comumente conhecido por 5 características ou dimensões, são elas: volume, velocidade, variedade, veracidade e valor [7].

- **Volume**: com as infraestruturas de armazenamento tornando-se cada vez mais acessíveis, os dados gerados por diferentes fontes possuem grandezas em ordem de petabytes ou zettabytes;
- **Velocidade**: refere-se a quão rápido os dados estão sendo produzidos, bem como o quão rápido os mesmos devem ser tratados para atender a demanda;
- **Variedade**: os dados produzidos possuem naturezas diversas, por exemplo, sites de comércio eletrônico web utilizam dados estruturados, *logs* de um servidor web são conhecidos como dados semi-estruturada e redes sociais lidam com dados não estruturados como áudio, vídeo, imagens e etc;
- **Veracidade**: diz respeito a confiabilidade sobre a fonte de dados, como por exemplo, mensurar os sentimentos dos clientes em redes sociais são incertos por natureza, já

que implicam uso do juízo humano. Ainda, esses dados contêm valiosas informações que mesmo imprecisas e incertas podem ser processados, representando uma das características do Big Data;

- **Valor**: os dados recebidos na forma original, geralmente possuem baixo valor em relação ao seu volume. Entretanto, pode ser obtido um valor elevado ao se efetuar a análise destes mesmos dados.

A riqueza contida nas informações obtidas pela a análise e processamento de Big Data potencializam a tomada de decisões estratégicas, de maneira a produzir respostas muito valiosas em tempo de execução. Por esse motivo, muitas organizações buscam por soluções eficientes, inteligentes, confiáveis e de baixo custo para lidar com processamento e análise de Big Data que, geralmente são realizadas utilizando o modelo de programação paralelo e distribuído MR.

### 2.1.1 Modelo MapReduce

O MR foi introduzido pela primeira vez na linguagem de programação Lisp e mais tarde popularizado pelo Google [11]. MR visa o processamento de grandes volumes de dados através da execução de várias operações de *map* e *reduce*, ambas as funções são implementadas pelo programador. A função de *map* recebe uma porção de dados do arquivo de entrada a ser processado, gerando um conjunto de tuplas intermediárias no formato chave-valor.

As tuplas são automaticamente agrupadas com base em suas chaves e em seguida, cada função de *reduce* recebe uma chave como entrada, bem como uma lista com todos os valores gerados pelas fases de *map* para cada chave recebida. Por fim, a lista de valores é ordenada ou combinada para produzir o arquivo de saída com os resultados. Ambas as funções são interligadas pela fase de comunicação e sincronização, denominada *shuffle* [11]. A Figura 2.1 demonstra o exemplo do fluxo de dados de uma aplicação MR em execução utilizando três funções de *map* e duas funções *reduce*.

Em primeiro lugar, os dados de entrada são divididos e pré-carregados nos discos locais de cada nodo. Em seguida, cada fase *map* processa os dados de entrada gerando dados intermediários. Os dados são transferidos das fases de *map* para as fases de *reduce* pela fase de *shuffle* que, de forma transparente efetua a cópia das saídas dos *maps* para os *reduces* apropriados. A fase de *map* também possui uma etapa de ordenação interna, que prepara os dados de saída dos *maps* para o *reduce*, cada *reduce* possui uma fase de agrupamento interna que, prepara os dados para o processamento que em seguida gera os resultados mediante a função utilizada.

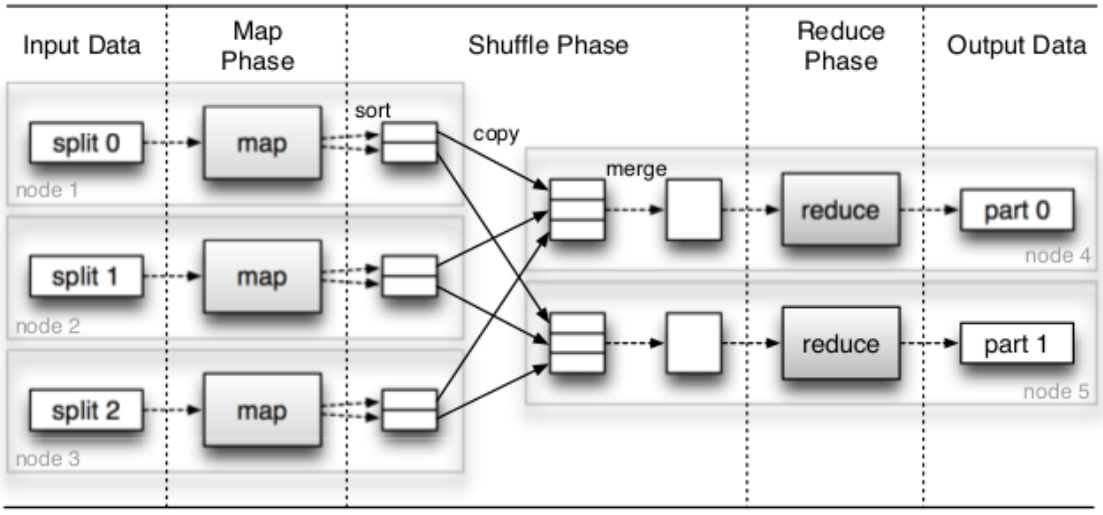



Figura 2.1 – Fluxo de Execução MapReduce [32]

Implementações MR são tipicamente acopladas com um Sistema de Arquivos Distribuídos (DFS) como Google File System (GFS) ou Hadoop Distributed File System (HDFS). O DFS é responsável por efetuar o armazenamento dos dados de todas as aplicações em blocos de 64(Padrão), 128 ou 256 MB distribuídos entre os múltiplos nodos do *cluster*. A localização dos dados é levada em conta durante o escalonamento das tarefas de MR, por exemplo, as implementações MR irão tentar escalonar uma tarefa de *map* em um nó que contém uma réplica dos dados de entrada. Isso se torna mais eficaz ao lidar com um grande conjunto de dados, pois o processamento local irá evitar a transferência via rede. Ainda funcionalidades como: paralelismo, tolerância a falhas, distribuição dos dados e balanceamento de carga são deixadas a cargo do MR, simplificando o processo de desenvolvimento. Do ponto de vista de sistemas distribuídos, o MR oferece transparência de replicação, distribuição e sincronização [43]. Ainda, o modelo MR possui diversos *frameworks* utilizados como base para efetuar o processamento Big Data, tais como Storm [16], Spark [15] e Hadoop [52]. Contudo, este trabalho irá se concentrar no Hadoop, pois é uma das implementações mais populares do MR.

### 2.1.2 Framework Hadoop

O *Hadoop MapReduce* (HMR) é um *framework Open-Source* desenvolvido em Java que implementa MR e inclui um escalonador para efetuar o gerenciamento de múltiplas aplicações MR de forma simultânea, semelhante ao uso de *batch jobs* em um *cluster*. Basicamente, um *job* (trabalho) MR é constituído por múltiplas tarefas de *map/reduce* escalonadas sobre um *cluster* Hadoop, onde várias tarefas podem ser executados simultaneamente no mesmo *cluster*.

HMR é composto pelo sistema de arquivos *Hadoop Distributed File System* (HDFS) que fornece resiliência, e alta taxa de transferência durante o acesso aos dados [43, 6]. O HDFS compõe a arquitetura HMR e é responsável por efetuar o armazenamento dos dados de todas as aplicações, utilizando blocos de 64(Padrão), 128 ou 256 MB distribuídos entre os múltiplos nodos do *cluster*. Ainda, diferentemente de outras abordagens, o armazenamento e processamento do HDFS é feito em cada nodo do sistema.

O HMR é composto por dois componentes principais: um JobTracker e alguns TaskTrackers [52]. Inicialmente, o cliente envia um trabalho para o JobTracker e em seguida, o JobTracker será o responsável pela coordenação da execução de todos os trabalhos no sistema, agendando tarefas *map/reduce* para serem executadas pelos TaskTrackers sobre um número fixo de *slots*[1]. Também é função de um TaskTracker relatar o seu progresso da execução para o JobTracker que, por sua vez mantém um registro sobre a evolução geral de cada trabalho. O JobTracker atribui as tarefas para os TaskTrackers mais próximos dos dados de entrada.

Os dados manipulados pelo HMR são dispostos em blocos e estão localizados sobre o HDFS, normalmente com o tamanho de 64MB (por bloco de dados). A arquitetura de um *cluster* HDFS é composta por dois tipos de nodos: um NameNode e um número de DataNodes. O NameNode mantém os meta-dados do sistema de arquivos, incluindo informações sobre a árvore de diretórios e de arquivos, bem como a localização física de cada bloco de dados, já os DataNodes são responsáveis por armazenar os blocos de dados. Há também um NameNode secundário que funciona como um apoio para o NameNode primário, porém é apenas usado em caso de falhas, como por exemplo, quando uma aplicação MR precisa ler um arquivo do HDFS, ela primeiramente se comunica com o NameNode para descobrir onde estão todos os DataNodes com os blocos de dados do arquivo a ser lido e, em seguida a aplicação inicia a leitura dos mesmos.

Os blocos de dados são tipicamente replicados (3 réplicas por bloco) sobre múltiplos DataNones, basicamente duas réplicas em diferentes nodos sobre o mesmo *rack* do *cluster* e uma sobre um *rack* diferente. A distribuição das réplicas seguem um algoritmo bem definido que contém as informações sobre a localização dos blocos e dos DataNodes. A replicação além de prover a tolerância a falhas, favorece o escalonamento de trabalhos MR, alocando o trabalho próximo aos dados de entrada, por exemplo, se um nodo que armazena um determinado bloco de dados já está executando muitas tarefas, é possível processar esse mesmo bloco de dados sobre outro nodo que possui os mesmos blocos de dados. Caso contrário, a tarefa é processada localmente utilizando blocos de dados remotos. Esse comportamento pode degradar o desempenho da aplicação devido à alta utilização de recursos de disco e rede.

[1] Um *slot* representa um núcleo do processador, aplicações MR são configuradas com base nos *slots* disponíveis.

**Versões do Hadoop**. Atualmente existem duas versões diferentes do Hadoop: 1.x, 2.x. A primeira representa a implementação original e mais utilizada do HMR e, por isso foi escolhida para ser utilizada neste trabalho. Já a segunda destina-se a ser a próxima geração do HMR, MapReduce 2.0 (MRv2) ou YARN (Yet Another Resource Negotiator) representa a nova geração de sistemas para o processamento Big Data. Além disso, todas as observações feitas para Hadoop 1.x neste trabalho, podem ser utilizadas no YARN. Do mesmo modo, as contribuições deste trabalho pode ser facilmente portadas para YARN no futuro. Por fim, computação MR normalmente é realizada sobre arquiteturas paralelas de processamento, tais como aglomerados (*clusters*) que vão ser visto a seguir.

### 2.1.3 Arquiteturas Paralelas para Processamento Big Data

As aplicações “Big Data” fazem da computação o mecanismo para criar soluções capazes de analisar grandes bases de dados, processar aplicações que utilizam cálculos complexos, identificar comportamentos e disponibilizar serviços especializados em seus domínios sobre a nuvem [34]. Ainda, muitos problemas são complexos e podem consumir horas ou dias de processamento quando processados nas arquiteturas convencionais que, mesmo em constante evolução são insuficientes para acompanhar a crescente complexidade das novas aplicações.

Dessa maneira, a computação paralela e distribuída acena como um papel fundamental no processamento e na extração de informação relevante as aplicações “Big Data”. Essa computação é normalmente realizada em aglomerados (*clusters*) que, com um conjunto de computadores comuns, conseguem agregar alto poder de processamento a um custo associado relativamente baixo [56].

*Clusters* permitem o compartilhamento de recursos entre vários usuários visando o melhor aproveitamento global dos recursos, como exemplo, um único nodo pode ser compartilhado por vários usuários e diferentes aplicações [55]. Nesse contexto, por mais que os usuários compartilhem o mesmo *cluster* e seus recursos entre múltiplas aplicações, os recursos vão ser utilizados conforme as solicitações das aplicações, não existindo controle sobre o compartilhamento de recursos entre às aplicações em execução.

Complementar a isso, a virtualização se torna uma boa opção para suprir tal deficiência, sendo a base da computação em nuvem, possibilitando o compartilhamento de recursos entre múltiplas aplicações em execução sobre um *cluster*. Adicionalmente, com a possibilidade de se utilizar a virtualização em CAD, o termo *clusters* virtualizados emergiu. *Clusters* virtuais são constituídos por máquinas físicas ou uma VM hospedada por vários *clusters* físicos [44, 22], que permitem maximizar a utilização dos recursos do *cluster* e ao mesmo tempo ajudam no quesito economia, uma vez que os gastos com infraestrutura e manutenção são minimizados.

Sendo assim, a utilização de *Clusters* aliada ao uso da tecnologia de virtualização se torna ainda mais eficaz. A próxima seção irá apresentar os conceitos que envolvem a tecnologia de virtualização, o compartilhamento de recursos e os problemas de interferências provenientes do uso dessa tecnologia.

## 2.2 Virtualização

A virtualização pode ser definida como um ambiente virtual que simula outro ambiente real, propiciando a utilização de diversos sistemas e aplicativos sem a necessidade de acesso físico à máquina na qual estão hospedados[49, 25, 20]. Alguns dos benefícios dessa tecnologia são: a alta disponibilidade, segurança, tolerância a falhas, o compartilhamento de recursos e a escalabilidade.

A utilização da virtualização diminui os custos operacionais com aquisição de infraestrutura, consolidando seus servidores como máquinas virtuais em um único servidor físico [20]. Para que isso seja possível é necessário a utilização de técnicas de virtualização que, por sua vez utilizam virtualizadores para emular de forma virtual os componentes físicos de um computador, tornando possível o compartilhamento de recursos entre as máquinas virtuais. As principais técnicas de virtualização são descritas a seguir.

### 2.2.1 Virtualização Tradicional

A Virtualização tradicional é representa pela virtualização total e pela para-virtualização, que possuem como principais representantes os seguintes virtualizadores. vSphere [49], Xen [36], KVM [28]. Esses virtualizadores são constituídos por um monitor da máquina virtual (VMM), localizado sob uma máquina física que fornece abstração total aos recursos do *hardware*. Veja Figura 2.2.

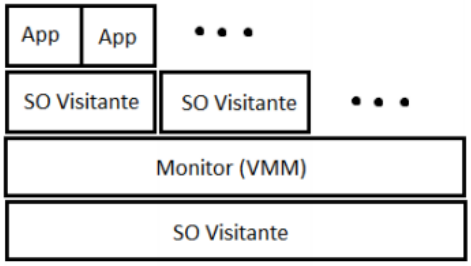



Figura 2.2 – Tecnologia Tradicional de Virtualização

Nesse exemplo, cada máquina virtual possuirá seu próprio SO executado de forma completamente isolada, em que o VMM irá possuir o completo controle dos recursos do

sistema. Uma máquina virtual em execução no sistema não tem acesso direto a qualquer dispositivo físico do *host*, ou seja, quaisquer solicitações devem passar pelo VMM. Ainda, esse deve fornecer e controlar as versões virtualizadas de dispositivos do sistema tais como: dispositivos de E/S, armazenamento, memória e entre outros.

Embora as técnicas tradicionais de virtualização forneçam uma série de vantagens, são tradicionalmente evitadas pela computação de alto desempenho (CAD), como exemplo *Clusters*. Isso ocorrer poque os *Frameworks* que executam aplicações MR sob o modelo tradicional de virtualização, são incapazes de atingir o mesmo desempenho a partir de plataformas não virtualizadas devido a sobrecarga presente nessas tecnologias [55]. Por esse motivo, existe a tendência de se utilizar a virtualização baseada em contêineres para o processamento que exija o alto desempenho, como é o caso do MR.

### 2.2.2 Virtualização Baseada em Contêineres

Virtualização em Nível de Sistema Operacional, comumente chamada de Virtualização baseada em Contêineres, é a forma menos intrusiva de virtualização (a camada de virtualização afeta minimamente as aplicações, com baixa ou nenhuma interferência). Ao contrário das técnicas tradicionais de virtualização, esse modelo não depende de um virtualizador. Em vez disso, o SO é modificado de forma a isolar múltiplas áreas de usuários (User-Spaces), em mais alto nível, intituladas contêineres [33].

Os contêineres são viabilizados através de um conjunto de *namespaces* que por sua vez possuem a função de abstrair e fornecer recursos computacionais de forma isolada, como por exemplo:

- ***Namespace* de Sistema de Arquivos**: limita o escopo do sistema de arquivos para um processo ou um grupo de processos [53];

- ***Namespace PID***: assegura que todos os processos em um contêiner possuam seus identificadores únicos [53];

- ***Namespace de Rede***: garante que cada contêiner possua seus próprios dispositivos de rede, endereços IP, roteamento, regras de *firewall*, caches de rede, entre outros [53];

- ***Namespace /proc e /sys***: assegura que cada contêiner possua sua própria representação do proc e do sys (sistemas de arquivos especiais usados para exportar algumas informações do *kernel* para aplicações. Em poucas palavras, são os subconjuntos de um sistema hospedeiro Linux [53].

- ***Namespace de Usuário***: consiste de tabelas de identificadores de usuários (uid), de forma que, o mesmo uid em diferentes contêineres possa ser referido a diferentes usuários [53];

- ***Namespace UTS***: garante que cada contêiner possua seu próprio *hostname* [53].

Com a utilização de *namespaces* assegura-se que diferentes usuários utilizem diferentes áreas sobre o mesmo sistema operacional, ou seja, o processo de um usuário torna-se incapaz de se comunicar com os processos de outros usuários, garantindo o isolamento do sistema como um todo e fazendo com que os usuários tenham a impressão de que estão trabalhando em sistemas independentes [56]. Em suma, um contêiner representa os *namespaces* sobre um único *kernel*, o do próprio SO, compartilhado de forma isolada sobre *contêineres*. A Figura 2.3 ilustra a comparação entre as tecnologias de virtualização tradicional e baseado em contêineres.

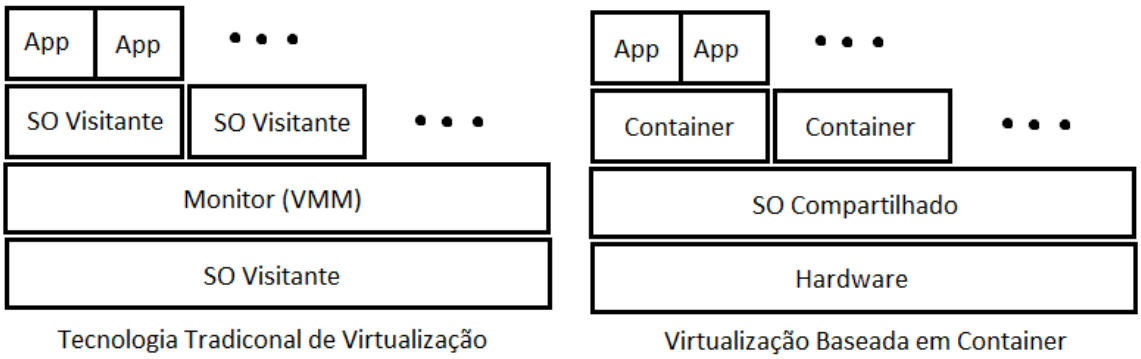



Figura 2.3 – Comparação Entre os Modelos de Virtualização

Como pode ser visto na Figura 2.3, enquanto a virtualização baseada em contêineres fornece abstração de *software* para cada contêiner sob o mesmo SO compartilhado, os modelos tradicionais de virtualização fornecem abstração (*hardware*) completa a cada SO visitante. A vantagem deste tipo de virtualização reside principalmente no desempenho, pois não é necessário o uso de um virtualizador para tradução das instruções entre *hardware/software*, resultando em ganhos de desempenho para os sistemas. Por esse motivo, a virtualização baseada em contêineres é uma alternativa mais leve quando comparada com os virtualizadores tradicionais como KVM, VMware, XEN.

Atualmente, Linux-VServer [45], OpenVZ [48] e *Linux Containers* (LXC) [8] são os principais representantes da virtualização baseada em contêineres. Ainda, o LXC (modelo de virtualização escolhido para esse trabalho) é o representante mais popular dentre a virtualização baseada em contêineres por ter funcionalidades implementadas no Kernel do linux. Ainda, o mesmo utiliza *namespaces* do *kernel* do Linux para fornecer isolamento de recursos entre todos os contêineres [55], ou seja, durante a inicialização de um contêiner LXC, por padrão, PIDs, IPCs e pontos de montagem são virtualizados e isolados por meio

da utilização de *namespace* de PID, de IPC e de sistema de arquivos, como visto anteriormente. Sendo essas características que possibilitam esse modelo de virtualização fornecer o compartilhamento de recursos.

### 2.2.3 Compartilhamento de Recursos em Clusters Virtualizados

Os modelos de virtualização fornecem o compartilhamento dos recursos de um *cluster* para *frameworks* e aplicações que efetuam o processamento em larga escala, como exemplo cita-se o HMR. Tipicamente as aplicação MR executam seus trabalhos em várias ondas (*waves*) de tarefas curtas sobre um *cluster*, em que o número de tarefas corresponde ao número de *slots* disponíveis e, tem como objetivo o rápido processamento em resposta a alta utilização de recursos.

Entretanto, ao utilizar *clusters* virtualizados e consolidar múltiplos *frameworks* de processamento e suas aplicações, como é o caso do Hadoop, problemas com o gerenciamento de recursos podem ocorrer, levando as aplicações em execução a resultados inconsistentes e pouco confiáveis. Isso ocorre porque o gerenciador de recursos de um *framework* não tem conhecimento sobre os recursos alocados para o processamento do outro, acarretando em problemas de interferência entre as aplicações em execução. Além disso, a sobrecarga presente os modelos tradicionais de virtualização podem levar à perda de desempenho devido a camada de comunicação e abstração existente entre *hardware/software*. Por esses motivos, a utilização dos modelos tradicionais de virtualização são comumente evitadas pela CAD [55].

Adicionalmente, o surgimento de novos *frameworks* que visem o processamento de aplicações Big Data é uma tendência. Contudo, gerenciar e compartilhar recursos a fim de isolar aplicações e obter o alto desempenho é um desafio. Essa necessidade aliada as vantagens que a virtualização baseada em contêineres oferece impulsionaram a criação de sistemas específicos para esse fim, como é o caso do Mesos [19] e o YARN [17].

Big Data Systems (BDS) como YARN (*Hadoop*, Versão 2.0) e Mesos foram desenvolvidos para serem gerenciadores de recursos, mas atualmente foram redesenhados e atualmente são conhecidos como Sistemas Operacionais distribuídos de larga escala [46, 40]. Esses sistemas oferecem compartilhamento e gerenciamento de recursos entre diferentes *frameworks* de processamento e suas aplicações, bem como oferecem compatibilidades entre as aplicações. Por exemplo, aplicações MR executadas no HMR também podem ser executadas no YARN/Mesos. A Figura 2.4 apresenta a arquitetura do sistema YARN como exemplo.

No entanto, esses sistemas gerenciam somente recursos de CPU e memória, deixando o controle relativo a operações de E/S em disco e rede como responsabilidade do Sistema Operacional (SO) [50]. Ainda, o inadequado gerenciamento de recursos entre

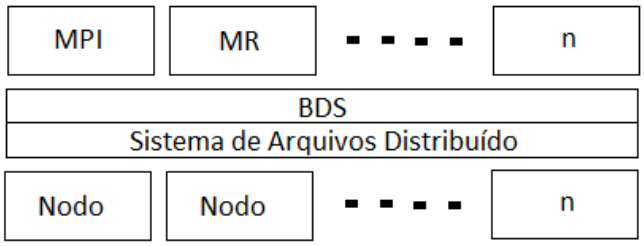



Figura 2.4 – Arquitetura de Sistemas Big Data

aplicações em execução de forma simultânea sobre o mesmo *cluster* podem resultar em problemas de interferência, que são causados pelas inúmeras solicitações de recursos efetuadas pelas aplicações em processamento [55, 12].

## 2.3 Interferência em Ambientes Virtualizados

Diversos problemas de interferência relacionados aos modelos tradicionais de virtualização anteriormente já estudados voltaram a ser estudados sobre contêineres. Problemas de interferência, são tipicamente encontrados quando ocorre a concorrência descontrolada pela utilização de recursos, entre aplicações em execução de forma simultânea sobre o mesmo ambiente virtualizado (compartilhado), problema esse também conhecido como contenção de recursos.

### 2.3.1 Contenção de Recursos

Aplicações executadas de forma simultânea compartilham o mesmo ambiente e podem competir pela utilização de recursos, por exemplo: utilização de CPU, caches compartilhados, barramento de memória, controladores de memória, utilização da rede e operações de leitura/escrita (E/S) em disco, causando assim a contenção de recursos e afetando diretamente as aplicações consolidadas sobre o mesmo ambiente virtualizado [5].

Além disso, quando a demanda pela utilização de recursos é muito alta e não é gerenciada, ou seja, as solicitações são maiores que os recursos disponíveis, as aplicações acabam interferindo entre si, impactando diretamente no desempenho das aplicações em execução.

A contenção de recursos pode causar falhas de isolamento na camada de virtualização, uma vez que a aplicação em execução excede a quantidade de recursos para si alocados. Mesmo que uma VM recebe uma fatia de recursos, isso não garante que as falhas de isolamento deixem de ocorrer [55, 54]. Por esse motivo, interferências de desempenho podem acontecer a partir de qualquer contenção de recursos ou problemas de isolamento.

Ainda, o crescimento constante de aplicações para o processamento Big Data, conhecidas pelo uso intensivo de disco (I/O Bound) trouxe à tona preocupações relativas a contenção de disco e o impacto que a mesma pode causar em ambiente em que o desempenho é crucial e o Service Level Agreement (SLA) ou Acordo de Nível de Serviço (ANS) que não deve ser violado, como em ambientes de nuvem computacional.

### 2.3.2 Contenção de Disco

A contenção de disco é conhecida por apresentar elevadas taxas de latência devido a concorrência pela utilização dos recursos de banda de disco entre aplicações MR em execução simultaneamente [29, 57].

A contenção de disco em aplicações MR ocorre devido às múltiplas tarefas de *map* e *reduce* iniciadas de forma simultânea e concorrente. Essas tarefas competem pelo barramento de disco, ou seja, por parte da largura de banda de disco (Escrita máxima suporta em disco por segundo), criando um cenário em que a solicitação por recursos de banda ultrapassa a banda disponível, causando a contenção.

Quando o limite da largura da banda é atingido, os problemas de contenção elevam as taxas de latência e transferência dos dados em processamento, fazendo com que os processos parem e forçando os mesmos a aguardarem até o disco ser liberado para escrita novamente. Tal comportamento é responsável por impactar diretamente no desempenho e no tempo de execução das aplicações MR em execução [26, 40].

Além disso, mesmo em número muito pequeno, algumas tarefas de *map/reduce* podem competir por recursos durante toda a execução da aplicação, até mesmo se os recursos do *cluster* já estiverem sobrecarregados. Por fim, as tarefas geram interferência entre si, necessitando maior tempo para processamento devido as variações de desempenho, que são inesperadas e podem prejudicar tanto a funcionalidade dos componentes internos do *framework* de processamento, bem como as aplicações em execução.

Nesse contexto, os escalonadores de disco fornecidos pelo SO possuem a tarefa de minimizar possíveis problemas de contenção, efetuando o escalonamento dos processos que escrevem em disco afim de evitar a interferência e com o objetivo de manter o desempenho das aplicações em processamento.

### 2.3.3 Escalonadores de Disco

Os escalonadores que atuam em nível de sistema operacional, como CFQ [9], noop [9] e deadline [9], ajudam a detectar gargalos de utilização de recursos e tentam

dividir os processos de forma equilibrada, reordenando e priorizando os mesmos ao tentar assegurar a divisão equilibrada dos recursos de disco.

Como resultado, os recursos são distribuídos igualmente entre os processos que estão em execução, porém, sem garantias quanto a performance. Isso ocorre porque os escalonadores são incapazes de prever ou tomar decisões relativas a carga de trabalho que está sendo processada, bem como qual é o nível de utilização de recursos necessário para o processamento das aplicações.

Contudo, ao utilizar a virtualização, esses escalonadores são sobrecarregados e dessa forma, nem sempre ao dividir igualmente os recursos as aplicações vão ser favorecidas, isso porquê o compartilhamento de recursos não leva em conta a utilização dos mesmos perante as aplicações que estão em processamento em ambientes virtualizados. Além dos escalonadores de disco, administradores de centro de dados podem agregar o uso de estratégias de alocação de recursos com o intuito de evitar problemas de interferência e isolamento.

## 2.4 Políticas Para a Alocação de Recursos em Ambientes Virtualizados

A alocação de recursos possibilita o provisionamento de ambientes elásticos, em que aplicações que estão em execução sobre VMs na nuvem possam ter seus recursos ajustados conforme a flutuação da carga de trabalho que está em processamento [10].

Dessa forma, podemos concluir que ambientes de nuvem permitem o ajuste adaptativo de recursos, para mais ou menos, a qualquer momento. Ainda, Galante e Bona [18] apresentam duas abordagens para a alocação de recursos: manual (estática) ou automática (dinâmica).

### 2.4.1 Alocação Estática de Recursos

A política de alocação estática de recursos exige a intervenção manual, ou seja, pode-se monitorar as aplicações em execução e se efetuar as alocações de recursos manualmente ou, simplesmente restringir todo o ambiente virtual. Como é demonstrado na Figura 2.5.

O exemplo apresentado na Figura 2.5 demonstra que alocações estáticas de recursos nem sempre podem atender a todos os contêineres em execução, isso porque em alguns casos será necessário a alocação de mais ou menos recursos para atender o processamento das aplicações em execução.

Como implicações da alocação estática de recurso pode-se citar:

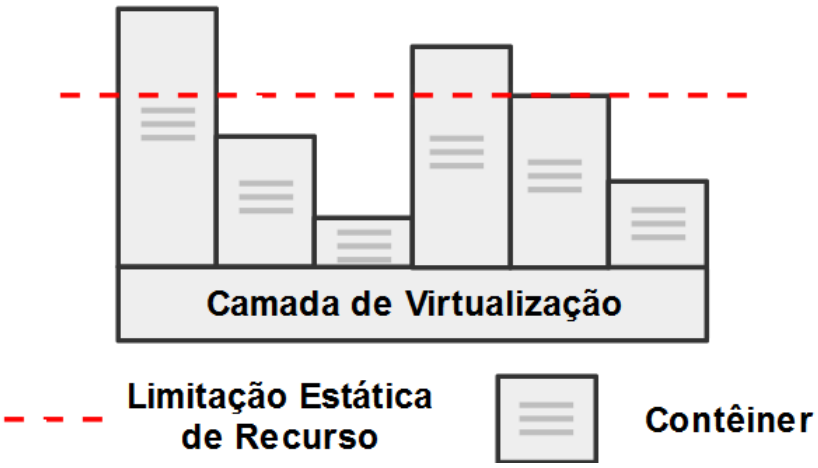



Figura 2.5 – Alocação Estática de Recursos

- **Sobre-utilização de Recursos**: Também conhecida como *Overutilization* defini-se como a situação onde são alocados recursos em excesso. Como exemplo, cita-se duas aplicações em execução sobre diferentes contêineres, um deles é limitado estaticamente com a maior fatia de recursos enquanto o contêiner com a maior necessidade de recursos recebe a menor fatia de recursos. Essa situação é tipica de um ambiente desbalanceado e pode potencialmente causar problemas de contenção se houver concorrência pela utilização dos mesmos;
- **Subutilização de Recursos**: Também conhecida como *Underutilization* ocorre sempre que um contêiner não utilizar na totalidade os recursos para si destinados. Esse problema ocorre comumente em infraestruturas de *cluster* e nuvens onde é comum o uso de alocações estáticas de recursos.

### 2.4.2 Alocação Dinâmica de Recursos

Ao utilizar a política de alocação dinâmica de recursos, cada aplicação em execução irá possuir seu próprio perfil de utilização de recursos. Desse modo, para alcançar o alto desempenho em meio a essa diversidade de aplicações é ideal efetuar o gerenciamento justo de recursos, ou seja, as alocações devem ser realizadas de forma dinâmica e de acordo com as necessidades de cada aplicações. Observar Figura 2.6.

A Figura 2.6 demonstra a alocação dinâmica de recursos, nesse exemplo cada contêiner receberá os recursos de acordo com as necessidades de sua aplicação. Além disso, essa política pode ser dividida em duas abordagens: reativa e proativa que, são listadas a seguir.

- **Proativas**: a abordagem proativa usa técnicas de predição para antecipar o comportamento de carga da aplicação e, assim, decidir pelas ações de elasticidade [33];

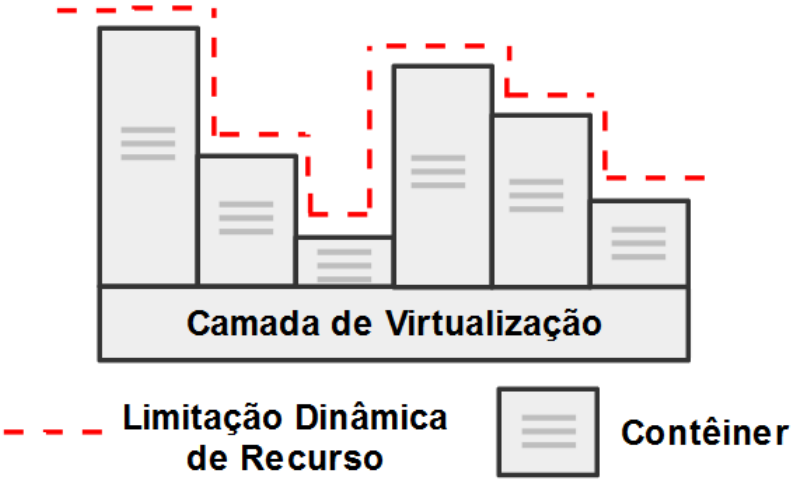



Figura 2.6 – Alocação Dinâmica de Recursos

- **Reativas**: a abordagem reativa é marcada pelo emprego do mecanismo regra-condição-ação [33].

Adicionalmente, uma regra é composta por um conjunto de condições que quando satisfeitas, disparam ações sobre a infraestrutura. Cada condição considera um evento ou uma métrica do sistema que é comparado com um limiar que, geralmente é fornecido por um sistema de monitoramento da infraestrutura, ou pela própria aplicação em execução [3, 39].

### 2.4.3 Alocação de Recursos Utilizando Grupos de Controle

Grupos de Controle (Cgroups) disponibiliza mecanismos para o agrupamento, rastreamento e limitação de processos sobre o *Kernel* de um SO, fornecendo a opção de controle sobre os recursos de CPU, memória, disco e rede, auxiliando os administradores a efetuar o controle total sobre o gerenciamento de recursos.

Organizado hierarquicamente, o Cgroup possibilita a criação de inúmeras hierarquias simultaneamente, representando uma ou mais árvores desconectadas e separadas de tarefas (processos), ou seja, seus subgrupos filhos podem herdar atributos (subsistemas) de seus pais. Um subsistema representa um recurso único, tal como a utilização de CPU ou memória e, assim por diante. Abaixo são listados alguns dos principais subsistemas que fazem parte do *cgroup*.

- cpu, este subsistema usa o escalonador para fornecer acesso às tarefas de *cgroup* para o CPU;
- cpuset, este subsistema atribui CPUs individuais (em sistemas *multicore*) e nós de memória para atribuir em um *cgroup*;

- *devices*, este subsistema permite ou nega acesso aos dispositivos por tarefas em um *cgroup*;
- *memory*, este subsistema define limites no uso de memória pelas tarefas em um *cgroup* e gera relatórios automáticos nos recursos de memória usados por essas tarefas;
- net_cls — este subsistema marca os pacotes de rede com um identificador de classe (classid) que permite ao controlador de tráfego do Linux (tc) identificar os pacotes que originam um cgroup em particular;
- *blkio*, define limites de acesso de entrada/saída para e a partir de dispositivos de bloco tais como drives físicos (discos, USB, etc);

A seguir vão ser descritas as considerações inerentes ao presente capítulo.

## 2.5 Considerações

Esse capítulo apresentou a fundamentação teórica necessária para a compreensão desse trabalho. Além disso, algumas observações podem ser realizadas, são elas: problemas como a contenção de disco continuam presentes, independentemente do modelo de virtualização ou sistema (*framework*) utilizado. Desse modo, o SO fica encarregado de solucionar os problemas de interferência, sendo necessário o uso de escalonadores e políticas de alocação de recursos para minimizar os problemas de contenção. Nesse sentido, visando investigar os problemas de contenção em disco, o próximo capítulo busca compreender o estado da arte.

# 3. TRABALHOS RELACIONADOS

Esse capítulo apresenta o estado da arte, os problemas e as soluções relativas a interferência de desempenho causada pela contenção em disco durante o processamento de aplicações MR.

As seções desse capítulo são distribuídas da seguinte forma: uso nativo, trabalhos que não fazem uso da virtualização; Virtualização tradicional, trabalhos baseados e que apresentam soluções referentes aos modelos tradicionais de virtualização e por fim, os trabalhos e soluções baseados em Contêineres.

## 3.1 Uso Nativo

YongChul *et al.* [60, 61] demonstra a variação de desempenho existente durante o processamento das tarefas *map* e *reduce*. Os autores relatam que o processamento dessas tarefas pode ser desequilibrado, ou seja, o tempo de processamento pode variar independente da aplicação que esteja em execução. Isso ocorre porque algumas tarefas durante a execução das fases de *map/reduce* podem exigir mais recursos de um determinado tipo, como por exemplo, mais recursos de CPU, memória e E/S em disco. Dessa forma, quanto maior for o desiquilíbrio existente durante o processamento maior será o tempo de execução das fases e consequentemente da aplicação. Por fim. esse comportamento pode levar o *cluster* a um rendimento significativamente menor devido à contenção de recursos.

Para solucionar o problema de desequilíbrio os autores desenvolveram a ferramenta *SkewTune*. *SkewTune* é uma extensão customizada do *Hadoop* que possui como finalidade monitorar a execução das tarefas *map/reduce* e informar ao sistema quais são as fases mais "lentas"e em seguida parar a execução das mesmas. Essas tarefas possuem seu bloco de dados particionado e redistribuído sobre os nós do *cluster*. Os resultados apontam melhora no desempenho das aplicações em até 4 vezes perante a execução padrão do *Hadoop*.

## 3.2 Virtualização Tradicional

Song *et al.* [42] menciona que muitas corporações dependem da análise de dados em larga escala para efetuar o processamento e a análise de logs, extração de características estratégicas sobre os dados. Nesse caso, o processamento das aplicações pode variar constantemente, tornando a demanda pela utilização de recursos como, CPU, memó-

ria disco e rede flutuantes. Nesse contexto, esse trabalho propõe um modelo para predizer o consumo de recursos das tarefas MR que compõem as aplicações MR.

O modelo é baseado em regressão linear múltipla e implementa um monitor para as tarefas MR e um módulo de predição. O monitor é encarregado de recolher informações sobre as propriedades relacionadas a cada tarefa MR, em seguida essas informações devem ser entregues para o módulo de predição que treina localmente o modelo de regressão linear múltiplo. Como resultado o modelo pode prever a utilização de recursos com precisão de aproximadamente 12%.

Siyuan *et al.* [41] observa o problema de contenção em disco gerado pela execução de aplicações MR. Geralmente o problema de desempenho nas aplicações MR é causado pela simultaneidade de tarefas de *map/reduce* em execução sobre o mesmo nó. Caso os recursos estejam sendo utilizados no limite, até mesmo um pequeno grupo de tarefas pode saturar os recursos disponíveis para o processamento, causando assim a contenção de disco.

Como estratégia para resolver esse problema os autores propõe duas modificações para o *Hadoop*, *I/O Throttling* e *Coordination*. A primeira é responsável por limitar o número de solicitações de E/S em disco, evitando a queda repentina de desempenho e consequentemente reduzindo a contenção de disco localmente. Já a segunda atribui prioridades para todas as tarefas MR, ou seja, no momento que uma tarefa estiver sendo executada as demais tarefas presentes no mesmo nó vão estar paradas o que torna a tarefa em execução detentora da totalidade do recurso de E/S durante sua execução. Com a utilização de ambas as técnicas é possível obter Qualidade de Serviço (QoS) para o ambiente de execução. Ainda os resultados demonstram que é possível reduzir o tempo médio de execução de uma aplicação MR em até 34%.

Ibrahim *et al.* [23] cita que problemas de interferência como a contenção de recursos são responsáveis pela degradação de performance. Ainda, os autores menciona que esse cenário se tornou ainda mais importante pelo fato de que sistemas de processamento de larga escala, como por exemplo aplicações MR que são executadas sobre ambientes de nuvem, exigindo o adequado gerenciamento de recursos. Além disso, esse trabalho identifica a contenção de disco como um dos principais desafios a serem resolvidos.

Para solucionar o problema esse trabalho propõe criar uma nova abordagem que efetue o processamento de aplicações MR utilizando pares de escalonadores, ou seja, a VM irá possuir um escalonador e o virtualizador outro. Desse modo, a cada execução esses pares podem ser variados a fim de encontrar o melhor caso. Os resultados demonstram que ao utilizar pares de escalonadores é possível obter até 25% de performance.

Matthew *et al.* [29] cita que atualmente conjuntos imensos de dados são processados utilizando a nuvem e que por esse motivo problemas de interferência tem chamado muita atenção, principalmente os ligados a contenção de disco que para esse caso decorrem da execução de aplicações MR.

Como solução esse trabalho propõe o *framework* Fennel. Fennel efetua o escalonamento de recursos entre as aplicações MR em execução através do monitoramento dos processos em execução e do *hardware*. As informações obtidas alimentam uma fórmula que é responsável por priorizar as aplicações que ofereçam os menores níveis de contenção de disco, garantindo assim a alta performance durante a execução das mesmas. Os resultados demonstram que é possível melhorar o tempo de execução em até 25%.

Xu *et al.* [57] comenta que aplicações para o processamento intensivo de dados continuam a crescer em vários contextos, sendo muito comum que isso ocorra em meio a ambientes compartilhados. Infelizmente os *frameworks* existentes de processamento não efetuam o gerenciamento dos recursos de disco e, por essa razão a performance das aplicações pode ser degradada consideravelmente devido a contenção de disco.

A solução proposta identifica os processos que estão gerando a contenção de disco e os distribui entre os nodos com base no uso e demanda de banda das aplicações. Os resultados demonstram que o uso do IBIS a performance comparada a um cenário com contenção de disco pode apresentar ganho até de 18%.

## 3.3 Contêineres

Yan *et al.* [50] elaborou um relatório técnico que estuda as técnicas de gerenciamento de recursos existentes no YARN. No presente trabalho o autor relata que esse sistema não efetua o gerenciamento de recursos de E/S em disco e rede, bem como relata que os *frameworks* suportados pelo YARN podem somente ser limitados quanto ao número de núcleos de CPU (vcpus) e quantidade de memória a ser utilizada.

Dessa forma, esse trabalho apresenta uma estratégia para o gerenciamento estático de recursos de leitura de disco. Já operações de escrita, são deixadas como trabalho futuro uma vez que o *kernel* não suporta escrita *bufferizada*, padrão utilizado pelo *Hadoop*. Essa solução particiona o disco em fatias, nomeadas (vdisks) que representam o percentual máximo de recursos de leitura que cada aplicação irá possuir, ou seja, o número de fatias será igual ao número de aplicações em execução que por sua vez correspondem ao percentual máximo de recursos que poderá ser utilizado. Contudo, esse trabalho ainda não foi concluído. Pois, oferece uma solução parcial para a resolução do problema.

## 3.4 Considerações

Para uma melhor interpretação sobre os trabalhos estudados a Tabela 3.1 foi criada. Sua finalidade é de apontar as principais similaridades/diferenças entre esses trabalhos e a presente proposta dessa dissertação. Além disso, essa tabela busca informar

algumas características sobre esses trabalhos, como por exemplo: qual o modelo de virtualização utilizado, se a estratégia altera o *framework* de processamento e também qual foi a solução empregada. Com essas informações em mãos é possível compreender o estado da arte e identificar possíveis oportunidades de pesquisa. Finalmente, com base nessa tabela, também foi classificado o presente trabalho.

Tabela 3.1 – Estado da Arte

| **Autor** | **# Ref.** | **Medelo de Virtualização** | **Altera o Framework** | **Solução Utilizada** |
|---|---|---|---|---|
| YongChul *et al* | [60, 61] | Não Utiliza | Sim | Particionamento e Redistribuição de Tarefas |
| Song *et al* | [42] | Virtualizador | Sim | Modelo preditivo para predizer desempenho |
| Siyuan *et al* | [41] | Virtualizador | Sim | Limita e prioriza tarefas de map/reduce |
| Ibrahim *et al* | [23] | Virtualizador | Sim | Manipulação Sobre os Escalonadores |
| Matthew *et al* | [29] | Virtualizador | Não | Escalonamento de Recursos e Limitação de Banda |
| Xu *et al* | [57] | Virtualizador | Não | Monitoramento de Processos e Limitação Sobre Banda de Disco |
| Yan *et al* | [50] | Contêineres | Sim | Alocação Estática de Recursos de Leitura em Disco |
| **Matteussi *et al*** | **-** | **Contêineres** | **Não** | **Um estudo experimental utilizando políticas para o gerenciamento de escrita em disco** |

Os trabalhos acima descritos demonstram a importância que a interferência em ambientes virtualizados possui nos dias de hoje. Isso se agrava devido ao fato de que *frameworks* para o processamento de dados em larga escala continuam a crescer sem que exista uma real preocupação quanto ao gerenciamento de recursos por parte dos mesmos. A tarefa de gerenciar recursos geralmente é função dos sistemas que suportam esses *fra-*

*meworks* ou dos virtualizadores responsáveis por hospedar toda essa infraestrutura. No entanto, muitas vezes esses sistemas somente tem capacidade ou, efetuam o controle sobre CPU e Memória.

Esses trabalhos também expõem que as soluções existentes são fortemente acopladas aos modelos de virtualização tradicional e que em muitas vezes suas soluções envolvem mudanças internas no *framework* de processamento, o que acaba tornando muitas vezes inviável a portabilidade dessas soluções para outras versões do MR ou para diferentes modelos de virtualização. Desse modo, acredita-se que o estudo sobre a interferência (contenção de disco) oriunda da execução de aplicações MR em ambientes virtualizados utilizando contêineres seria uma boa oportunidade de pesquisa, por ser um tópico com poucos estudos até o momento e também por ser um modelo de virtualização que oferece desempenho próximo ao nativo.

Finalmente, com base no estudo efetuado e nas observações acima apresentadas o presente trabalho também foi classificado. Sendo assim, esse trabalho aborda um estudo experimental que busca através do uso de políticas para o gerenciamento de recurso de escrita minimizar a contenção de disco a fim de acelerar as aplicações MR que sejam executadas sobre contêineres. Ainda, a solução proposta deve ser transparente, ou seja, a mesma deve ser independente da versão da MR, do ambiente de virtualização ou *hardware* para o processamento utilizado. Além disso, essa solução irá levar em conta somente o gerenciamento dos recursos de escrita, pois geralmente é sobre recurso que a interferência decai. O próximo capítulo irá apresentar de forma detalhada a solução proposta.

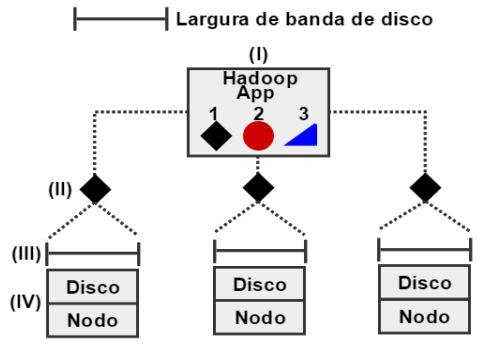
# 4. UM ESTUDO EXPERIMENTAL PARA MINIMIZAR O IMPACTO DA CONTENÇÃO DE DISCO SOBRE AMBIENTES VIRTUALIZADOS UTILIZANDO CONTÊINERES

No decorrer desse capítulo serão contextualizados o problema e a proposta desse trabalho, bem como será efetuado um estudo experimental que busque responder as questões pesquisa relacionadas ao mesmo.

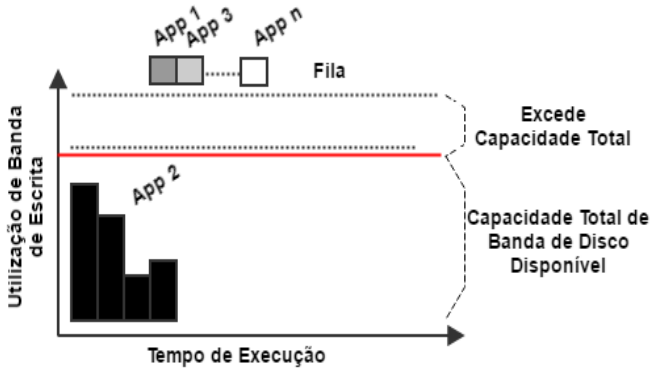
## 4.1 Problemática

Aplicações MR foram inicialmente concebidas para serem executadas sequencialmente em arquiteturas distribuídas de larga escala, tais como *cluster* de computadores, ou seja, uma após a outra, sem o compartilhamento de recursos. Durante a execução dessas aplicações, os recursos disponíveis pelo *cluster* são utilizados em sua totalidade (CPU, memória, rede e disco) com o objetivo intrínseco de potencializar o alto desempenho. Além disso, às inúmeras tarefas de *map* e *reduce* geradas pelas aplicações MR são compostas por pequenos blocos de dados.

Essa característica, possibilita o processamento veloz e concorrente das tarefas de *map* e *reduce* sobre um *cluster*, observe as aplicações 1, 2 e 3 na Figura 4.1(a) - I. Por fim, ao final da execução de cada tarefas, os dados gerados pelo processamento das mesmas é consolidados em disco.




(a) Nesse exemplo, as aplicações são executadas de forma sequencial, ou seja, uma após a outra sobre todos os nodos do *cluster*. Durante a execução de cada aplicação (1, 2 e 3), os recursos como à largura de banda de escrita em disco são utilizados até seu limite.

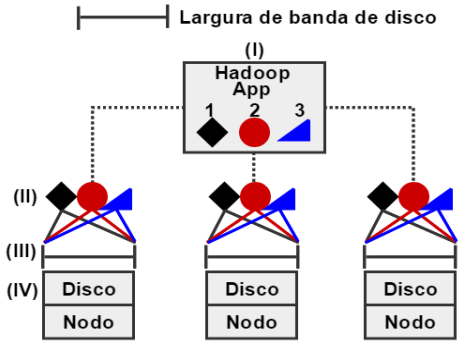



(b) Ao executar aplicações em modo sequencial, o tempo de execução obtido será menor em comparação com aplicações executadas em modo compartilhado (Figura 4.2). Isso ocorre porque não existe compartilhamento de recursos de disco entre aplicações, e consequentemente a contenção para com o mesmo não ocorre.

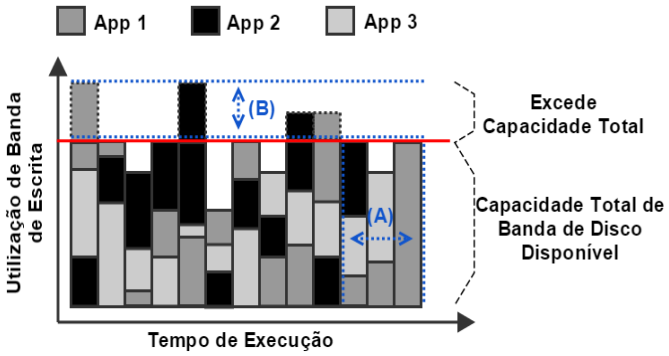
Figura 4.1 – Executando Sequencial de Aplicações MR

Ao final do processamento das tarefas de *map* e *reduce*, os dados obtidos (Figura 4.1(a) - II) são consolidados em disco, utilizando o recurso de escrita (banda) até o limite suportado pelo hardware (Figura 4.1(a) - III/IV). Esse ciclo se repete para cada aplicação que estiverem esperando para ser processada.

Entretanto, manter o *cluster* processando apenas aplicações em modo sequencial potencializa problemas como a subutilização de recursos, ou seja, existem recursos ociosos que poderiam estar sendo utilizados para o processamento de outras aplicações. Nesse contexto, a utilização de *clusters* virtualizados permite a execução de inúmeras aplicações MR de forma compartilhada e simultânea, como pode ser observado na Figura 4.2 - (a) e (b).




(a) Durante o processamento de aplicações MR em modo compartilhado a competição pela utilização de recursos de escrita em disco (banda) ocorre de forma intensa (I/II), ou seja, as aplicações buscam utilizar esse recurso até o limite disponível, para assim consolidar os dados processados (III/IV).




(b) O consumo desordenado de recursos é consequência da falta de gerenciamento de disco que, devido a intensa e simultânea utilização dos recursos de escrita (banda), excede a capacidade total de banda disponível (B). Esse comportamento pode levar a contenção de disco e também ao baixo desempenho, afetando diretamente o tempo de execução das aplicações. (A)

Figura 4.2 – Execução Simultânea de Aplicações MR

Também é possível observar na Figura 4.2(a), que ao executar múltiplas aplicações sobre um ambiente compartilhado que não efetue o adequado gerenciamento de recursos de disco, poderá ser encontrada a seguinte situação: todas as aplicações em execução (4.2(a) - II) vão concorrer pela largura total de banda de disco disponível no momento em que forem consolidar os dados processados (4.2(a) - III). Ainda, a concorrência desordenada pela utilização de recursos de disco observada na Figura 4.2(b) irá resultar em problemas de interferência, nesse caso, a contenção de disco.

A contenção de disco ocorre no momento em que a largura total de banda for saturada, ou seja, caso a aplicação MR possua inúmeros tarefas de *map*/*reduce* escrevendo em disco simultaneamente, mesmo em um número muito pequeno ou consolidando poucos dados, a contenção de disco pode ocorrer. Isso porque para cada tarefa processada simultaneamente, a largura de banda disponível é utilizada até seu limite, como pode ser observado na Figura 4.2(b). As implicações da contenção de disco podem levar tanto

o *frameworks* quanto as aplicações por esses executadas a apresentarem instabilidades, tempos de execução alternados, baixo desempenho, erros de execução durante o processamento das aplicações, resultados pouco confiáveis e sobrecarga sobre outros recursos.

Infelizmente, devido aos problemas de interferência ou subutilização de recursos, as soluções e trabalhos relacionados apresentadas anteriormente (Capítulo 3) podem não alcançar o alto desempenho e o eficiente compartilhamento de recursos entre as aplicações MR em execução. Além disso, nem mesmo os atuais sistemas Big Data como Mesos/Flink e YARN que utilizam a virtualização baseada em contêineres, efetuam adequadamente o gerenciamento dos recursos de disco. Esse comportamento possibilita que problemas de interferência ocorram ao deixar o controle sobre as operações de disco como responsabilidade do SO. A seguir será contextualizada a proposta desse trabalho.

## 4.2 Proposta

Além da tendência em utilizar contêineres para aplicações CAD em execução sobre ambientes compartilhados, o uso desse modelo de virtualização traz a tona diversos problemas já resolvidos para as tecnologias tradicionais de virtualização. Nesse sentido, esse trabalho aborda um estudo experimental que busca através do uso de políticas para o gerenciamento de recurso de escrita minimizar a contenção de disco a fim de acelerar as aplicações MR que sejam executadas sobre contêineres.

Para chegar ao objetivo proposto, apresenta-se nesse trabalho, de forma simples e eficiente, políticas para a alocação estática (ajuste manual) de recursos de escrita (banda) em disco. O principal objetivo é controlar a disputa não coordenada pela utilização de recursos de escrita por contêiner. Evitando assim, problemas de interferência (contenção de disco) entre aplicações MR executadas concorrentemente sobre *clusters* virtualizados utilizando contêineres. Ainda, a solução proposta deve ser transparente, ou seja, deve ser independente da versão da MR, do ambiente de virtualização, *hardware* ou, *framework* utilizado para o processamento. A Figura 4.3 apresenta de forma conceitual essa proposta.

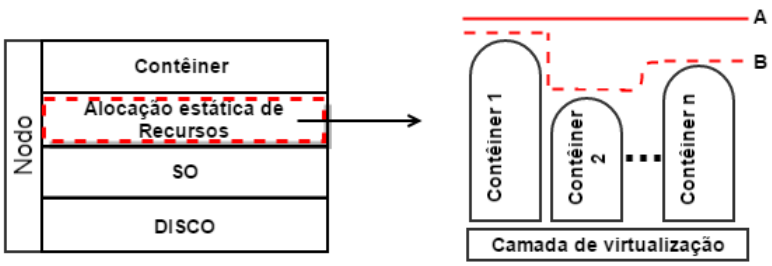
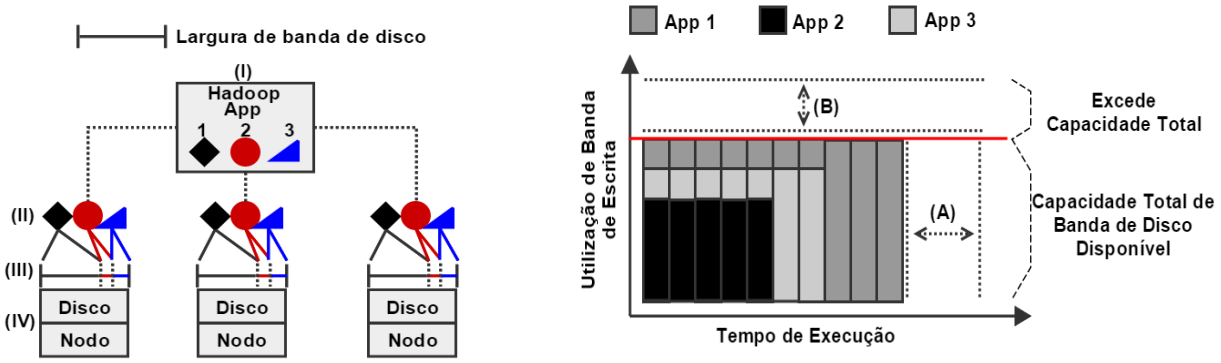



Figura 4.3 – Estratégia Proposta: A) Recursos Disponíveis; B) Alocação Estática de Recursos por Contêiner

A política de alocação estática de recursos possibilita o compartilhamento dos mesmos em *clusters* [19] virtualizados, isso permite o controle e a utilização dos recursos de disco para cada contêiner, ou seja, é possível maximizar a utilização dos mesmos, aumentando a performance e garantindo que os mesmos sejam reservados para as aplicações em execução sem interferência e, minimizando assim a contenção de disco. Por fim, com os ajustes manuais, espera-se que as aplicações restringidas evitem o cenário de interferência (contenção de disco). Ainda, as aplicações detentoras da maior fatia dos recursos de escrita em disco (banda) vão ser aceleradas em resposta a estratégia proposta, que fornece um cenário livre de interferência. Conforme pode ser observado conceitualmente na Figura 4.4 - (a) e (b).




(a) A utilização da alocação estática de recursos permite efetuar restrições (alocar fatias de recursos) de banda de disco (III) de forma isolada e para cada contêiner que conter uma aplicação MR (II).

(b) Ao utilizar a estratégia de alocação estática de recursos os mesmos vão ser coordenados, não excedendo o limite imposto (B), evitando assim a interferência (contenção de disco) e resultando no alto desempenho das aplicações MR quando comparadas a execuções efetuadas sobre ambientes compartilhados sem o adequado gerenciamento de recursos.

Figura 4.4 – Executando Aplicações MR sobre Contêineres Utilizando a Estratégia de Alocação Estática de Recursos

A estratégia pode ser vista na Figura 4.4(a) em que múltiplas aplicações estão em execução utilizando a alocação estática de recursos (Figura 4.4(a) I e II). Essa proposta garante que cada contêiner receba uma fatia dos recursos de banda de disco de forma isolada, garantindo a largura de banda de escrita para cada aplicação em execução. Além disso, essa estratégia deve ser replicada para cada nodo do *cluster*, garantindo assim que os recursos sejam gerenciados adequadamente.

No exemplo demonstrado na Figura 4.4(b), três contêineres possuem seus recursos de banda de disco restringidos estaticamente, mantendo o isolamento, gerenciamento ordenado de recursos e também evitando a contenção de disco, como pode ser visto na Figura 4.4(a). As alocações são efetuadas estaticamente para cada contêiner de forma aleatória em um primeiro momento e, depois realocadas estaticamente com base no término das aplicações em execução. Dessa forma, essa estratégia busca demonstrar que as

restrições de banda de escrita ajudam a obter desempenho (Figura 4.4(b) - (A)) ao evitar problemas de interferência (B).

A seguir, serão demonstrados os experimentos e avaliações realizados afim de verificar a eficiência e eficácia da proposta de alocação da presente proposta.

### 4.2.1 Ambiente Experimental

Os experimentos a serem descritos nessa seção possuem o comum objetivo de responderem as questões de pesquisa desse trabalho. A seguir é apresentado o ambiente operacional utilizado para a execução dos experimentos.

A estrutura utilizada para os experimentos é composta por 1 nodo Dell PowerEdge M610 com dois processadores Intel Xeon Six-Core E5645 2.4GHz Hyper-Threading, totalizando 12 núcleos (24 threads) por nó, 24GB de memória e 1 disco Serial Attached SCSI (SAS) de 300GB com 10K (mil) RPM. Neste nó foram criados 3 *cluster* virtuais, compostos por 1 contêiner e 1/3 dos recursos de memória e cpu disponíveis do nodo.

A pilha de *software* utilizada é composta pelo sistema operacional Ubuntu Server 14.04.1 LTS com *patch* para o Kernel (version 3.3), para assim suportar as operações de bufferização de E/S efetuadas pelo Hadoop[1]. Isso porque versões tradicionais do Kernel não suportam restrições de escrita em *buffer*. O modelo de virtualização de contêineres utilizado sobre o Ubuntu foi o LXC (version 1.0.7). Ainda, para cada contêiner, o Hadoop (version 1.1.2) foi instalado e, configurado com 6 tarefas de *map* e 1 de *reduce*, reservando 3 cores para o processo de monitoramento.

O monitoramento dos recursos de disco foi feito por contêiner e também para o nodo que hospeda os *cluster* virtuais, auxiliando na coleta de informações, como por exemplo a utilização de banda de disco. Para essa tarefa, utilizou-se uma versão modificada do software IOtop[2].

### 4.2.2 Caracterizando a Utilização de Recursos de Aplicações MapReduce

A transição para a computação em nuvem é um grande desafio atualmente, pois combina a necessidade de serviços altamente escaláveis com o suporte para processar uma diversidade de *framewors* e suas cargas de trabalho de forma simultânea [21]. Nesse contexto, o MR se tornou muito atrativo para o processamento de dados em larga escala, uma vez que é capaz de efetuar a computação de inúmeras tarefas de *map/reduce* de

[1] Custom Kernel: git://git.kernel.org/pub/scm/linux/kernel/git/wfg/linux.git buffered-write-io-controller
[2] Custom Iotop: https://github.com/mvneves/iotop-cgroups

aplicações MR simultaneamente. Essas tarefas mesmo em um número pequeno, podem levar a utilização dos recursos do *cluster* até o seu limite, ocasionando assim problemas de contenção.

Por essa razão, esse trabalho busca minimizar a contenção de disco sobre ambientes virtualizados que utilizem contêineres, mas para que isso seja possível inicialmente é necessário compreender como os recursos de disco são utilizados e quais são os problemas performáticos que podem ser encontrados, sendo esse o objetivo desse experimento. Com base no monitoramento da execução de aplicações MR vão ser capturados os rastros de execução inerentes a utilização dos recursos computacionais, tais como: disco, rede, CPU e memória.

Para simular um ambiente com características reais, optou-se pelo uso do Hibench Benchmark Suite [21]. Esse *benchmark* oferece uma série de aplicações utilizadas no mundo real, são elas: Sort, Terasort, WordCount, Nutch, PageRank, Bayes, K-means e Hive. Abaixo uma breve descrição de cada aplicação e suas respectivas aplicações.

- **Sort, Terasort e WordCount**: Sort e Terasort efetuam operações de ordenação de documentos, registros páginas e etc. WordCount conta o número de ocorrências de palavras em arquivos de texto. Essas aplicações efetuam operações básicas e são muito utilizadas no campo da análise de desempenho, comércio eletrônico, mecanismos para busca na Web e redes sociais;
- **Nutch e PageRank**: são aplicações que utilizam algoritmos para calcular a classificação de páginas web, ou seja, a avaliação da relevância de uma determinada página, são muito utilizadas em mecanismos para busca na Web;
- **Bayes e K-means**: Bayes é uma biblioteca de aprendizagem de máquina de código aberto construído em cima do Hadoop que implementa a parte de treino do algoritmo de classificação Naive Bayesian. K-means implementa o algoritmo para descoberta de conhecimento e é responsável por fornecer uma classificação de informações de acordo com os próprios dados. Ambos fazem parte do Apache Mahout e são considerados algoritmos de classificação e clusterização muito utilizados por redes sociais, comércio eletrônico, mineração de dados, processamento de imagens, reconhecimento textual e etc;
- **Hive**: é um sistema de armazenamento de dados de código aberto para consulta e análise de grandes conjuntos de dados armazenados pelo Hadoop. Muito utilizado em redes sociais, mecanismos de busca, comércio eletrônico, para o processamento de logs, extração de informações Web.

Para cada aplicação acima mencionada foi efetuado o monitoramento da utilização dos recursos com o auxílio da ferramenta Dstat Monitor [31]. O arquivo com os *logs* gerados pelo Dstat foram utilizados como entrada para gerar os gráficos relativos ao consumo de

recursos para cada uma das aplicações. Particularmente, para a realização desse experimento foram utilizados 4 (quatro) nós de um *cluster*, sua configuração de *hardware/software* pode ser visto na Tabela 4.1.

Tabela 4.1 – Configuração de Software e Hardware

| | |
|---|---|
| Processador | Dual-socket quad-core Intel(R) Xeon(R) E5520 @ 2.27GHz (with HyperThread) |
| Memória | 16GB ECC DRAM |
| Disco Rígido | x 1 SCSI 10000RPM HDD of 300GB |
| Rede | 1 Gigabit Ethernet NIC |
| Sistema Operacional | Ubuntu Linux 12.04 LTS (kernel 3.2.0-29-generic) |
| Versão Hadoop | Hadoop 1.1.2 |
| Número de *maps* | 8 |
| Número de *reduces* | 8 |
| JVM | jdk1.7.0_45 |

Os resultado obtidos foram sumarizados na Tabela 4.2 e apresentam informações relativas a carga de entrada, saída e tempo total de execução por aplicação. É interessante observar a variação que as cargas de trabalho possuem (carga de entrada/saída), esse comportamento irá ajudar na classificação dessas aplicações ao final desta seção. Ainda, ao compreender a utilização de recursos de aplicações MR se pode maximizar o desempenho das mesmas uma vez que a utilização de recursos estará mapeada.

Tabela 4.2 – Sumário de Execução por Aplicação

| **Aplicação** | **Carga de Entrada** | **Carga de Saída** | **Tempo de Execução (s)** |
|---|---|---|---|
| Sort | 15,9Gb | 15,1Gb | 2225 |
| Terasort | 37,2Gb | 37,2Gb | 1265 |
| WordCount | 11,4Gb | 1Mb | 2286 |
| Nutch | 509Mb | 208,4Mb | 266 |
| Pagerank | 1,3Gb | 476Mb | 160 |
| K-means | 46,4Gb | 57Gb | 3038 |
| Bayes | 352Mb | 1Mb | 401 |
| Hive | 5Gb | 1,7Gb | 788 |

As Figuras 4.6 e 4.7 apresentam o rastro de execução de todas as aplicações mencionadas na Tabela 4.2. É importante ressaltar que as aplicações MR são compostas por n *jobs* que correspondem a fases particulares de processamento, sendo enumeradas e demonstrados nessas figuras. Ainda, como esse trabalho é relacionado com operações de disco, as observações inerentes ao uso dos recursos de CPU, memória e rede se comentadas, serão de forma breve.

Ao verificar o comportamento das operações de disco, primeiramente pode ser observado que o disco é fortemente atrelado a CPU, isso ocorre pois cada tarefa de um *job* MR é ligada a um núcleo do processador do nodo. Além disso, as operações de escrita possuem relação direta com o CPU *wait* e os recursos de leitura com o CPU *usr*. Essas variáveis indicam quanto tempo as tarefas de *map/reduce* esperaram na fila até serem processadas, ou seja, a subutilização dos recursos de CPU. O processamento MR ocorre em rajadas (*waves*) e por um curto período de tempo, levando o *cluster* a alta utilização de recursos de disco. Entretanto, se o recurso de CPU *wait/usr* apresentar o uso de recursos constante e elevado pode representar tarefas MR muito ineficiente ou, problemas com o disco rígido. Esse comportamento pode ser visto claramente nas Figuras 4.6(a), 4.6(b) e 4.6(c).

Também foi observado que aplicações que fazem uso de disco de forma intensiva também utilizam mais recursos de rede, como pode ser visto nas Figuras 4.6(a), 4.6(b), 4.6(c) e 4.7(d). Isso porque a consolidação de dados é mais intensa para algumas aplicações, as fases de escrita no MR são sempre efetuadas ao final do processamento de cada tarefa de *map*, após a transferência dos dados via *shuffle* e ao final do processamento de cada *reduce*. Esse comportamento pode ser visto com maior nitidez na Figura 4.5, em que ao fim de uma rajada de tarefas *map* (I) ocorre a consolidação dos dados em disco (II). Geralmente a quantidade de tarefas de *map/reduce* é maior que a quantidade de *slots* (cores do processador) disponíveis para processamento e, por conta disso a computação dos dados é efetuada em *waves*/blocos ou rajadas.

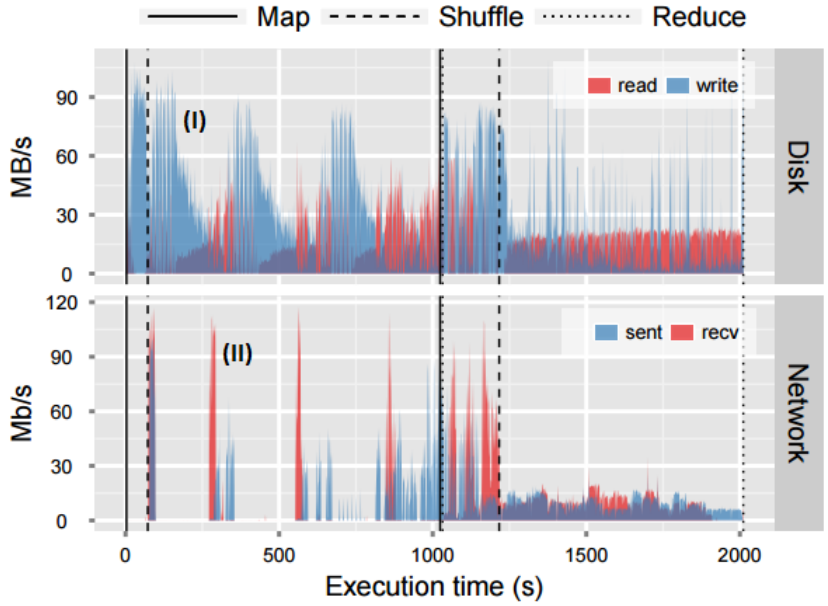
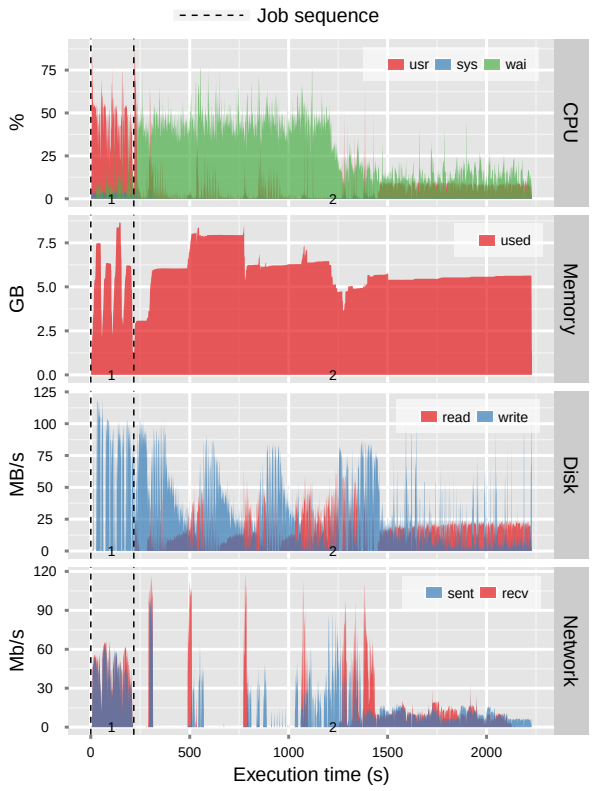


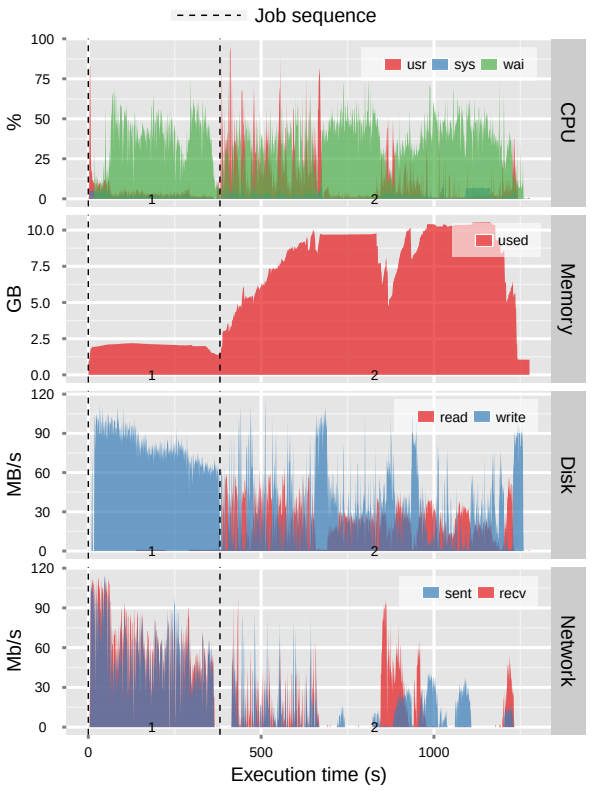

Figura 4.5 – Job 2, Aplicação Sort: Relação ao Uso de Disco e Rede

Cada aplicação MR possui um ciclo de processamento que se repete até que todos os dados sejam processados e, como os ciclos são idênticos é muito comum que as aplicações em execução formem padrões bem definidos de utilização de recursos, alguns

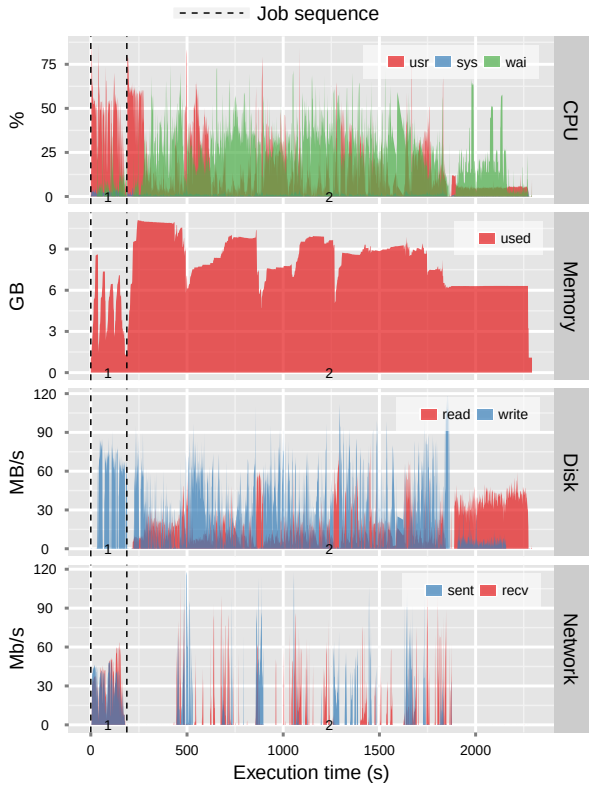
exemplos podem ser vistos nas Figuras 4.6(a), 4.7(a), 4.7(b), 4.7(c), 4.7(d). Contudo, ao observar os padrões de processamento é muito claro que ocorre a subutilização de recursos durante a execução de quaisquer aplicações MR, como pode ser visto em ambas as Figuras 4.6, 4.7. A subutilização representa o baixo consumo de recursos ou também a sequer utilização dos mesmo. Esse comportamento pode formar vales sem a utilização de recursos que podem ser vistos claramente na Figura 4.6(d), 4.7(a) e 4.7(b). Ainda, ao utilizar ambientes virtualizados é muito comum que a subutilização cause problemas de interferência, pois representa a má distribuição de recursos.



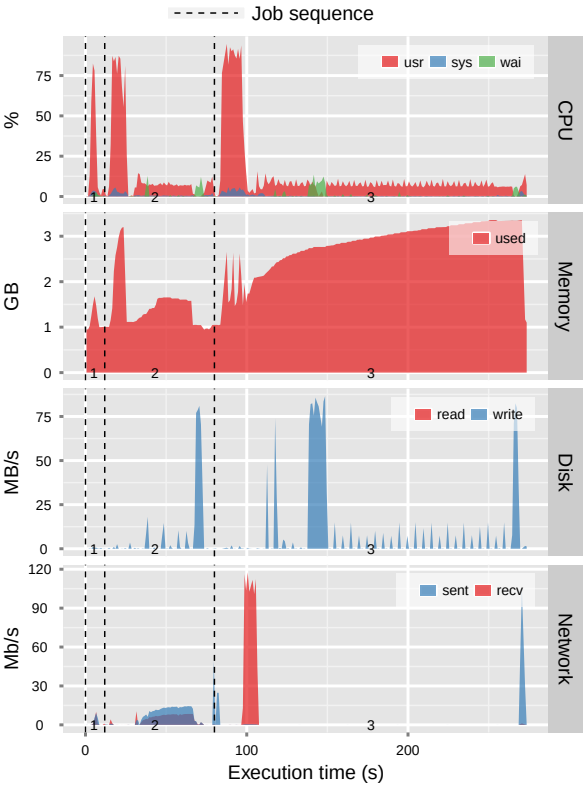

(a) Execução de Todos os Jobs da Aplicação Sort. Jobs: (1) random text writer, (2) sorter Sort




(b) Execução de Todos os Jobs da Aplicação Terasort. Jobs: (1) teraGen, (2) terasort

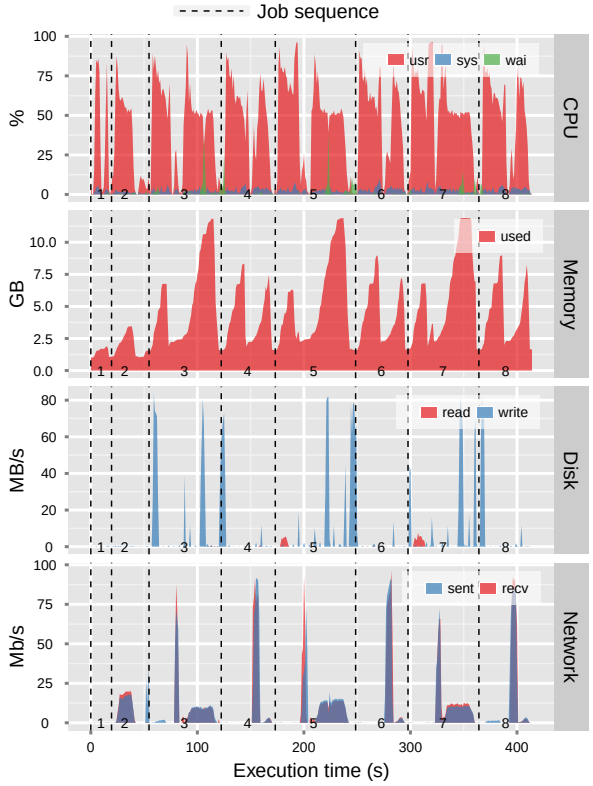



(c) Execução de Todos os Jobs da Aplicação Wordcount. Jobs: (1) random text writer, (2) word count

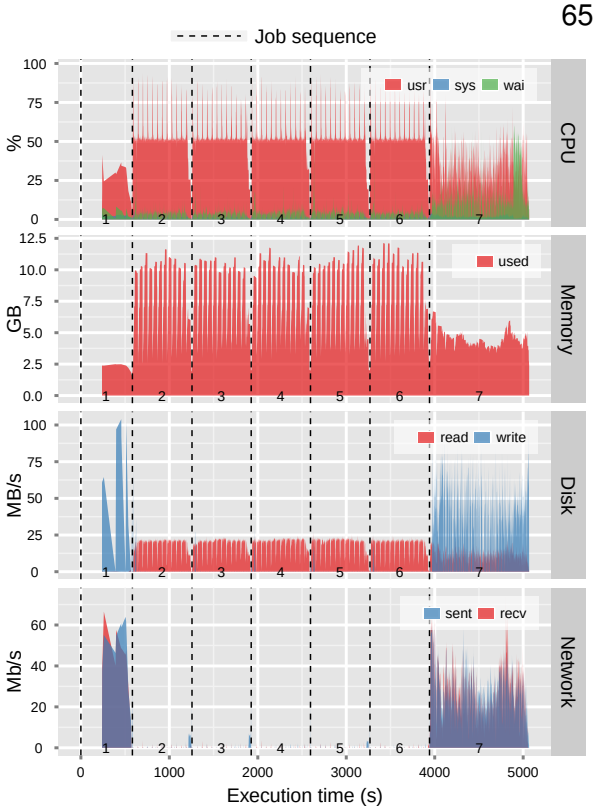



(d) Execução de Todos os Jobs da Aplicação Nutch. Jobs: (1) create nutch urls, (2) create nutch index data, (3) create index lucene

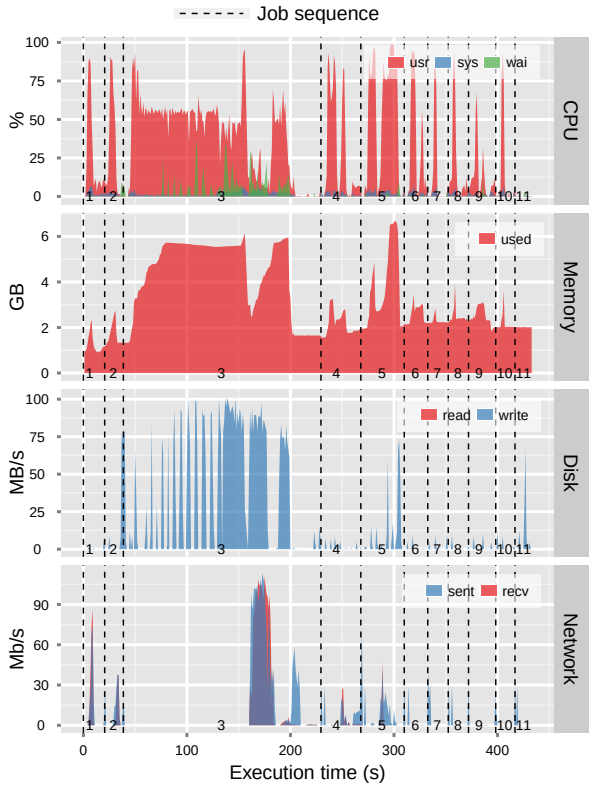
Figura 4.6 – Rastro de utilização de Recursos, Aplicações: Sort, Terasort, Wordcount, Nutch



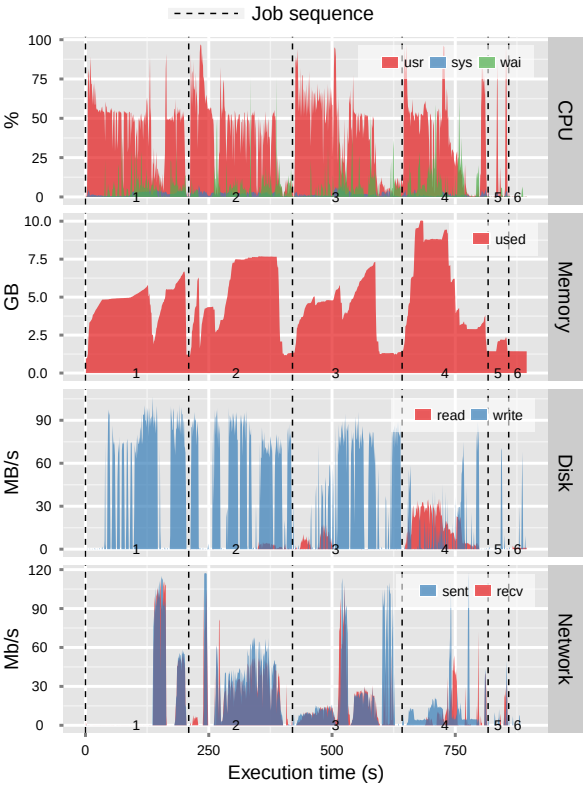

(a) Execução de Todos os Jobs da Aplicação Pagerank. Jobs: (1) create pagerank nodes, (2) create pagerank links, (3-8) estágios e iterações




(b) Execução de Todos os Jobs da Aplicação K-means. Jobs: (1) datatools, (2-6) estágios de iterações no cluster, (7) cluster classification




(c) Execução de Todos os Jobs da Aplicação Bayes.Jobs: (1) create bayes data, (2) document tokenizer, (3) generate collocations, (4) compute NGrams, (5) dictionary vectorizer, (6,9) partial vector merger, (7) vector TfIdf, (8) make partial vectors, (10,11) train index mapper-reducer




(d) Execução de Todos os Jobs da Aplicação Hive. Jobs: (1) create rankings, (2) create uservisits, (3) Job INSERT uservisits, (4-6) estágios de Job INSERT rankings

Figura 4.7 – Rastro de utilização de Recursos, Aplicações: Pagerank, K-means, Bayes, Hive

Outra observação interessante diz respeito ao consumo intensivo de recursos em momentos alternados, como por exemplo a aplicação K-means (Figura 4.7(b)) faz uso intensivo dos recursos de escrita durante a parte final de seu processamento enquanto a aplicação Bayes (Figura 4.7(c)) utiliza intensamente recursos de escrita nas fases iniciais. Isso indica que mais aplicações poderiam ser executadas simultaneamente, ou seja, os recursos do *cluster* poderiam ser distribuídos entre as aplicações de tal forma que sua utilização fosse maximizada. Nas Figuras 4.6 e 4.7 podem ser observados os perfis de utilização de recursos das aplicações. Além disso, ao verificar que as aplicações seguem padrões e que possuem vales sem a utilização de recurso pode-se sugerir a combinação de *workloads* (cargas de trabalho) a fim de obter a melhor utilização dos recursos do *cluster*.

Caracterizando Aplicações MR

Nessa seção vamos abordar de forma coesa a caracterização de aplicações MR, vão ser avaliados o consumo de recurso em nível de CPU, memória, disco e rede, bem como a Tabela 4.2 que apresenta os dados referentes a carga de trabalho das aplicações. Por fim, os rastros de execução (*traces*) demonstrados nas Figuras 4.6 e 4.7.

As aplicações Sort/Terasort transformam dados de uma representação para a outra, por essa razão possuem a mesma quantidade de dados no início e no final de sua execução. Ambas as aplicações efetuam a utilização de recursos de forma intensiva e intermitente, caracterizando as mesmas como aplicações de disco intensiva. Ainda, durante a fase de *shuffle* a aplicação Sort faz uso intensivo de rede por conta da manipulação de uma grande quantidade de dados, podendo até mesmo apresentar problemas de contenção de rede. Os recursos de CPU e memória são caracterizados como moderado, uma vez que os recursos são utilizados intensivamentes somente por alguns momentos.

A aplicação Wordcount extrai uma pequena quantidade de dados de um grande conjunto, possuindo assim alta utilização de disco, CPU e memória nas fases de processamento iniciais, caracterizando esses recursos como uso moderado. Já o consumo de rede na fase de *shuffle* é baixo devido a pequena quantidade de dados serem transmitidos para os *reduces*.

A aplicação Nutch apresenta picos alternados de utilização de recursos, utilizando mais CPU nas fases iniciais de processamento pois descompacta uma série de urls inseridas pelos usuários dos blocos de dados (crawl data) e mais disco nas fases de *reduce* em que são efetuados cálculos sobre esses links e reconsolidados em disco (re-crawl data). Contudo, acreditasse que o teste tenha sido criado com um conjunto de dados muito pequeno e, por conta disso essa aplicação foi categorizada como moderada para os recursos de CPU, memória e disco, já os recursos de rede possuem baixa utilização.

A aplicação Pagerank é composta por inúmeras iterações para calcular o ranking de uma página, nesse cotexto são utilizados intensamente os recursos de CPU, memória de forma moderada, seguida pela baixa utilização de recursos de disco e rede.

A aplicação K-means passa o maior tempo efetuando iterações para o processamento do *centroid*, utilizando os recursos de CPU de forma moderada, memória de forma intensiva, seguidos pelo baixo consumo de recursos de rede e disco que somente são utilizados no início e no final da execução dessa aplicação.

A aplicação Bayes é formada por vários ciclos de processamento, a maior parte do tempo de execução é utilizado para efetuar o cálculo do algoritmo Naive Bayesian, que por sua vez, utiliza os recursos de CPU de forma moderada, seguido pelos recursos de memória, rede e disco que apresentam atividade constante mas com baixo consumo de recursos.

Finalmente, a aplicação Hive efetua diversas operações que simulam banco de dados utilizando todos os recursos de forma moderada. Por fim, a caracterização efetuada foi sumarizada e pode ser visto na Tabela 4.3. Ao observa-lá é evidente a relação entre os recursos, se grandes quantidades de dados forem processadas nas tarefas de *map/reduce*, consequentemente irão existir um número elevado de transferências na rede e, caso exija alto necessidade de processamento, são necessários mais recursos de memória.

Tabela 4.3 – Caracterização da Utilização de Recursos de Aplicações MapReduce

| Aplicação | Disco | Rede | Memória | CPU |
|---|---|---|---|---|
| **Sort** | Intenso | Intenso | Moderado | Moderado |
| **Terasort** | Intenso | Intenso | Moderado | Moderado |
| **Wordcount** | Moderado | Baixo | Moderado | Moderado |
| **Nutch** | Moderado | Baixo | Moderado | Moderado |
| **Pagerank** | Baixo | Baixo | Moderado | Intenso |
| **K-means** | Baixo | Baixo | Intenso | Moderado |
| **Bayes** | Baixo | Baixo | Baixo | Moderado |
| **Hive** | Moderado | Moderado | Moderado | Moderado |

Essas observações são suficientes para responder a questão Q1 desse trabalho. **Como os recursos computacionais são consumidos pelas aplicações MapReduce?** Os experimentos realizados até o presente momento revelaram que as aplicações possuem diferentes perfis de utilização de recursos de disco e, que o adequado gerenciamento dos mesmos é ideal para maximizar o uso dos mesmo em um *cluster*. Ainda, ao consolidar aplicações MR em um ambiente com a utilização dos recursos de disco controlados será possível executar mais de uma aplicação simultaneamente com baixa ou nenhuma interferência.

A seguir será apresentado um simples experimento que serve como base para calibrar o percentual máximo de utilização de disco utilizados no restante desse trabalho.

### 4.2.3 Definindo a Largura de Banda de Escrita

Este experimento tem como finalidade descobrir a largura de banda máxima disponível para escrita em disco. Ainda, garantir a taxa de banda de escrita com precisão é essencial para a obtenção de resultados confiáveis, isso porque todos os experimentos são calibrados com essa taxa, ou seja, as aplicações são limitadas e devem respeitar o limite impost.

A métrica utilizada para o presente teste é a largura de banda em MBps. O experimento foi conduzido da seguinte maneira: operações de escrita bufferizadas e não bufferizadas vão ser efetuadas pelo *benchmark* Flexible I/O (FIO) durante dois minutos, a fim de utilizar a largura total de banda de escrita disponível. Os resultados apresentam um intervalo de confiança de 95%.

Os resultados obtidos demonstram que a taxa máxima de escrita em disco foi de 185MBps. A partir desse momento e para quaisquer limitações de disco efetuadas durante esse trabalho, as limitações devem ser calculadas com base nesse valor. Além disso, também pode-se ver na Tabela 4.4 uma sequência de testes com limitações efetuadas em intervalos de 10%. Esses testes foram executados durante a execução desse trabalho, mas em um segundo momento. Sua finalidade é de apresentar os valores referentes a banda máxima de escrita calculada sobre a taxa obtida de 185Mbps, conforme pode ser visto na Tabela 4.4.

Tabela 4.4 – Alocações Estáticas de Recursos x Largura de Banda de Escrita em Disco

| **Utilização do disco (%)** | **Banda de Escrita Utilizada (MBps)** |
|---|---|
| 100 | 185 |
| 90 | 166,5 |
| 80 | 148 |
| 70 | 129,5 |
| 60 | 111 |
| 50 | 92,5 |
| 40 | 74 |
| 30 | 55,5 |
| 20 | 37 |
| 10 | 18,5 |

O próximo experimento irá observar a existência da contenção de disco ao executar múltiplas aplicações MR sobre contêineres.

### 4.2.4 Avaliando a Contenção de Disco Sobre Contêineres

Esse experimento foi realizado com o intuito de demonstrar que o problema de contenção de disco existe e pode afetar aplicações em execução sobre o mesmo ambiente virtualizado utilizando contêineres. A métrica utilizada para a realização desse teste foi o tempo de execução da aplicação de ordenação MR Sort, os resultados respeitam o intervalo de confiança de 95%.

O ambiente foi configurado de tal forma que somente os recursos de disco não fossem controlados, isolando assim outros possíveis problemas de performance. Além disso, para cada contêiner, os recursos de CPU e memória foram restringidos em 50%. Já o recurso de disco recebeu a restrição de 185MBps por contêiner - taxa definida no experimento anterior.

O experimento foi conduzido da seguinte maneira: foram criados dois contêineres, em um deles a aplicação MR Sort foi executada com carga de entrada de 5GB, já no segundo contêiner foi utilizado o *benchmark* IOzone [4]. Nesse contexto, enquanto a aplicação MR estiver sendo executada, o IOzone irá produzir operações de E/S intensamente e com objetivo de levar o disco a utilização máxima de banda de escrita, podendo conduzir o teste a um cenário de contenção de disco.

A Figura 4.8 demonstra dois cenários distintos, em que o tempo de execução da aplicação MR Sort é consideravelmente afetado quando o ambiente virtualizado é levado a um estado de contenção de disco. Isso ocorre devido ao ineficiente gerenciamento de recursos entre as aplicações Sort e o IOzone que, são responsáveis por efetuar múltiplas operações de E/S em disco de forma simultânea.

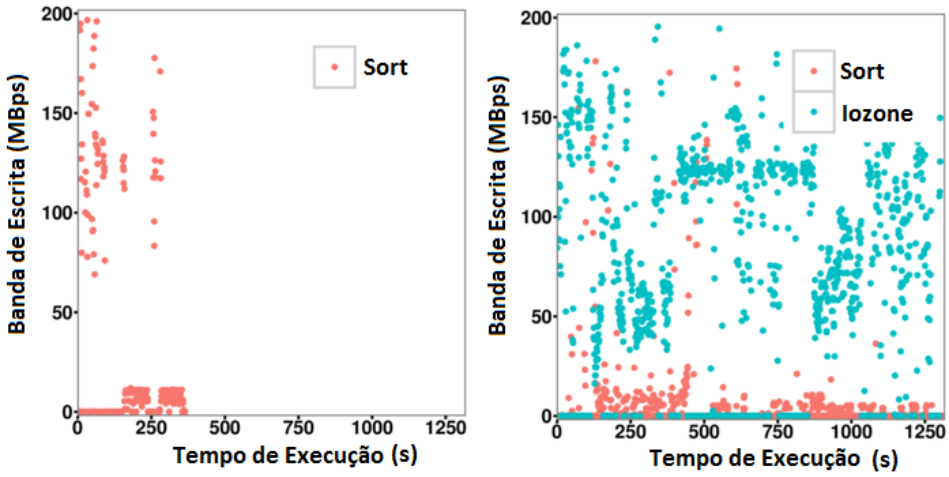



Figura 4.8 – Interferência de Desempenho Causada Pela Contenção de Disco

Além disso, a contenção de disco ocorre devido a largura de banda não ser suficiente para ambas as aplicações em execução, tornando o desempenho dessas, flutuantes, ou seja, o tempo de execução e a utilização podem variar entre as execuções.

Complementando o presente experimento, buscou-se realizar casos de testes reais, em que aplicações MR são executadas simultaneamente. Os testes foram executados seguindo as mesmas restrições, configurações e métrica do caso de teste anteriormente demonstrado. Os tempos de execução podem ser vistos na Tabela 4.5, representando a execução dessas aplicações no melhor caso, ou seja, as aplicações foram executadas utilizando 100% dos recursos de disco.

Tabela 4.5 – Tempo Sequencial de Execução de Aplicações MR

| **Aplicação** | **Carga (GB)** | **Tempo de execução (s)** |
|---|---|---|
| Sort (A) | 5GB | 289 |
| Sort (B) | 10GB | 992 |
| Terasort (A) | 10GB | 666 |
| Terasort (B) | 20GB | 1562 |
| WordCount (A) | 2,5GB | 399 |

A Tabela 4.6, apresenta um conjunto de testes e os tempos de execuções obtidos em um cenário compartilhado entre duas aplicações executadas simultaneamente.

Tabela 4.6 – Tempo de Execução Sequencial X Tempo de Execução em Contenção de Disco

| **Teste** | **Aplicação** | **Tempo de execução (s)** | **Aplicação** | **Tempo de execução (s)** |
|---|---|---|---|---|
| + 1 | Sort (A) | 464 | WordCount (A) | 646 |
| 2 | Sort (B) | 1919 | Terasort (A) | 1489 |
| 3 | Terasort (B) | 4348 | Terasort (B) | 4372 |

Pode-se observar que ao compartilhar um ambiente para a execução de múltiplas aplicações MR, os problemas de interferência, como a contenção de disco podem ocorrer. Quando as aplicações estão em execução simultaneamente, elas concorrem pela utilização total da banda de disco disponível, ou seja, durante a execução de aplicações MR podem existir momentos em que a banda de disco disponível não irá atender a demanda das aplicações, causando assim a contenção de disco sobre os contêineres.

Os resultado obtidos são suficientes para responder a questão de pesquisa Q2. **A consolidação de aplicações MR em contêineres conduz a perda de desempenho devido a contenção de disco?** Como foi observado nesse experimento, a contenção de disco influência no desempenho de aplicações MR executadas simultaneamente sobre contêineres. Os resultados fortalecem a ideia de que o adequado gerenciamento de recursos é essencial para manter o desempenho de múltiplas aplicações em um ambiente compartilhado com baixa ou nenhuma interferência.

Nesse sentido, o próximo experimento investiga a possibilidade de se restringir os recursos de disco sobre contêineres, uma vez que, é evidente a presença de interferência e também a necessidade de gerenciamento para com esse recurso.

### 4.2.5 Garantindo o Isolamento Entre Contêineres

Contêineres se tornaram uma tendência para o processamento de aplicações MR e, por conta disso atuais sistemas como YARN, Flink e Mesos fazem uso desse modelo de virtualização para executarem suas aplicações. Entretanto, nem mesmo os atuais sistemas fazem a gestão adequada de recursos de disco, ou seja, ao utilizar contêineres para o processamento MR as aplicações vão se comunicar diretamente com SO, através dos Grupos de Controle (CGroups) fornecido pelo Linux, porém, sem qualquer gerenciamento de recursos.

Dessa maneira, esse experimento busca avaliar o isolamento de um dos subsistemas pertencentes ao CGroups, o Block I/O Controller (*blkio*). Ainda, como uma de suas características, o *blkio* possibilita a restrição de largura de banda, permitindo priorizar a execução de aplicações MR em meio a um ambiente compartilhado, garantindo assim um melhor controle sobre o uso dos recursos.

A limitação da largura de banda de disco define o limite máximo de banda para operações de leitura/escrita sobre um dispositivo específico, ou seja, um dispositivo pode ter sua taxa de leitura e escrita limitada da seguinte maneira:

*blkio.throttle.read_bps_device e blkio.throttle.write_bps_device* especificam a taxa máxima para operações de leitura e escrita de um dispositivo, onde essas taxas são expressadas em bytes por segundo. O formato abaixo corresponde ao dispositivo (disco) da máquina que terá as restrições aplicadas (maior: menor) seguido pela taxa máxima para leitura e escrita respectivamente escolhida em bytes. Abaixo ambos os exemplos onde o dispositivo está apto a ler e escrever a uma taxa de 15 Mbps.

Ex, *echo “8:0 15728640” >blkio.throttle.read_bps_device*

E por fim.

Ex, *echo “8:0 15728640” >blkio.throttle.write_bps_device*.

A métrica utilizada para a realização desse teste foi o percentual de largura de banda de escrita alocado para a aplicação MR Sort, ou seja, a aplicação não deve utilizar mais recursos dos quais foram alocados. Os testes levam em conta o intervalo de confiança de 95%.

O ambiente foi configurado de tal forma que somente os recursos de disco não fossem controlados, isolando assim possíveis problemas de performance. Ainda, para esse experimento foram criados dois contêineres, em que os recursos de CPU e memória foram

setados em 50%. Entretanto, para esse teste apenas um contêiner foi utilizado com o intuito de ajustar a largura de banda de escrita livre de interferência e de forma crescente, em intervalos de 20% até o limite de 100% (185MBps). O valor de banda de disco utilizado para as alocações durante os testes pode ser visto na Tabela 4.4.

Ao observar os resultados obtidos e apresentados na Figura 4.9 - (A) Kernel Padrão, pode-se notar que nenhuma das limitações foram respeitas. Após pesquisa, se descobriu que isso ocorre pode dois motivos interligados.

1. Inicialmente o Hadoop consolida os dados em disco de forma bufferizada, ou seja, assincronamente através do uso de uma fatia da memória (buffer) para o armazenamento temporário de dados antes de gravar os mesmos em disco. Esse comportamento ajuda o *framework* Hadoop a obter desempenho, uma vez que minimiza a contenção de disco ao evitar que processos lentos sejam consolidados junto com processos rápidos.

2. Entretanto, ao executar aplicações MR sobre contêineres, ou outros ambientes virtualizados, os dados a serem consolidados em disco não respeitam os limites, uma vez que o subsistema *blkio* pode restringir somente operações síncronas de escrita, não atendendo ao padrão de escrita utilizado pelo hadoop [9, 50].

A fim de resolver esse problema, esforços foram realizados para encontrar soluções que permitissem a restrição sobre as operações de escrita síncronas e assíncronas. Foram testadas inúmeras versões de Kernel, desde versões anteriores à 3.3, até atuais versões, como a 4.x. Infelizmente, os resultados continuaram sendo insatisfatórios e devido a isso, intensificou-se a procura por *patches* para correção desse problema diretamente no Kernel.

O problema só foi solucionado ao utilizar uma versão customizada (patch) para o Kernel, disponibilizada e desenvolvida por Wu, Fengguang [51]. Essa versão do *Kernel* monitora de forma constante todos os processos de escrita que estão utilizando a *cache*, repassando a informação desses processos para o sistema responsável, o CGroups.

Conforme pode ser visto na Figura 4.9 - (B) Kernel com Patch, com a utilização do *patch* é possível restringir o uso dos recursos de banda de escrita em disco para aplicações MR em execução sobre contêineres. Entretanto, nem sempre a solução funciona perfeitamente, podendo estar sujeita a pequenas falhas de isolamento e que podem ser vistas em todos os testes apresentados na Figura 4.9. Os pequenos deslizes (falhas) no isolamento ocorrem devido a sobrecarga criada pelas diversas operações de E/S em disco que, mesmo em um número muito pequeno podem prejudicar o desempenho das aplicações em execução.

Adicionalmente, conforme os limites são alterados o tempo de execução da aplicação pode aumentar ou diminuir. Com a banda limitada a 20% o tempo de execução aumenta

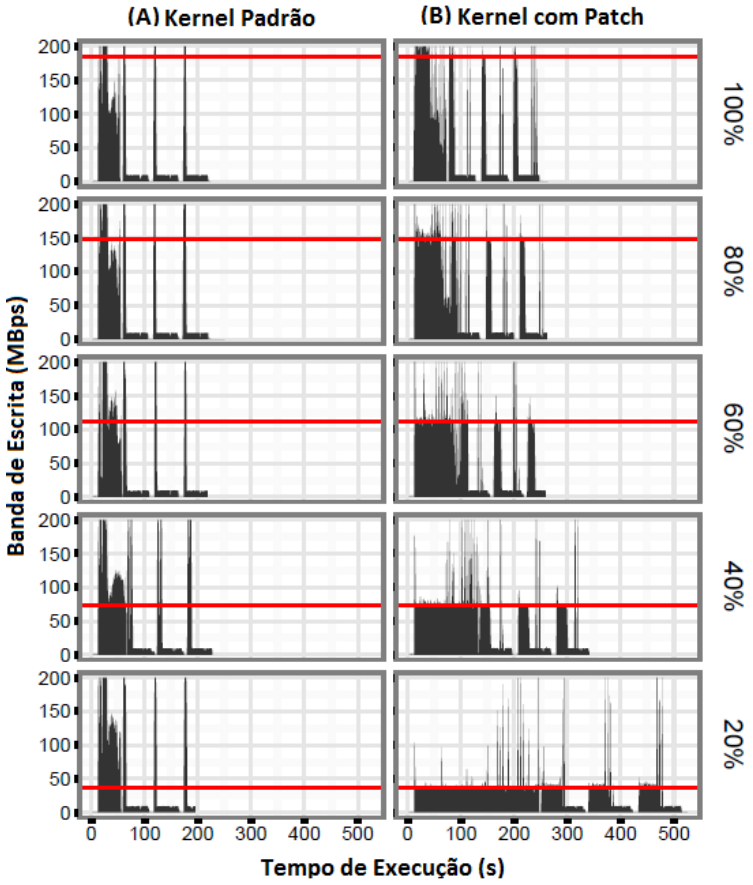



Figura 4.9 – Restrição de Disco x Isolamento: (A) Kernel Padrão e (B) Kernel Com Patch

uma vez que a aplicação Sort necessita utilizar mais recursos que o limite estabelecido para si. Ainda, em determinado momento os limites estabelecidos são satisfatórios, ou seja, o desempenho da aplicação não é alterado quando a banda é aumentada.

A questão Q3 elaborada diz respeito ao seguinte questionamento. **É possível efetuar restrições em disco a fim de limitar o uso desses recursos para aplicações MR em execução sobre contêineres?** Dessa forma, os resultados demonstram que as restrições de banda de disco são respeitadas, bem como é possível se obter isolamento ao executar uma aplicação MR utilizando contêineres, respondendo assim a questão Q3 desse trabalho.

O próximo experimento tem como objetivo demonstrar que é possível se obter ganho de desempenho ao controlar a contenção de disco, para esse fim será utilizada a estratégia de alocação estática de recursos.

### 4.2.6 Alocação Estática de Recursos

O compartilhamento de recursos entre diferentes *frameworks* de processamento e suas aplicações em um *cluster* requerem uma gestão de recursos eficaz e justa, para assim evitar possíveis problemas de interferência e desempenho causados pela contenção de disco. Ainda, embora o gerenciamento de CPU e memória já estejam bem resolvidos, questões relacionadas ao gerenciamento de disco permanecem em aberto. Nesse sentido, se torna necessário o uso de estratégias que ofereçam o adequado gerenciamento dos mesmos em meio a ambientes que apresentem a utilização descoordenada dos recurso de disco, principalmente os recursos de escrita.

Este experimento representa o primeiro passo em busca de uma estratégia dinâmica para o gerenciamento de recursos de disco. Ainda, o mesmo possui como finalidade minimizar a contenção desse recurso entre as aplicações MR executadas sobre um *cluster* utilizando contêineres. Nesse contexto, o presente experimento aborda o uso da alocação estática de recursos para efetuar o gerenciamento dos recursos de banda de escrita em disco, ou seja, serão efetuados ajustes manuais por aplicação, visando controlar a concorrência simultânea e desordenada de recursos em meio a um ambiente compartilhado. Desse modo, se espera obter ganho de desempenho ao evitar/controlar a contenção de disco.

A aplicação Teragen foi escolhida para esse experimento por efetuar operações de escrita em disco de forma linear e intensiva, propiciando assim a alta utilização do dispositivo de armazenamento. O Teragen será executado sobre três contêineres utilizando cargas variáveis (dados de entrada) de 10, 20 e 30GB, respectivamente. Cada contêiner irá possuir os recursos de escrita restringidos com o intuito de controlar o uso de banda de escrita, minimizar a contenção de disco e por fim, de acelerar as aplicações MR em execução. Para que isso ocorra, as restrições de disco vão ser definidas com base nas observações obtidas sobre a realização de alguns experimentos preliminares, abaixo apresentados.

Os testes vão ser reproduzidos sobre os escalonadores de disco fornecidos pelo Kernel, são eles: CFQ, noop e deadline. Ainda, a métrica utilizada para esse experimento é o Makespan (MS) que representa a diferença de tempo entre o início e fim de uma sequência de trabalhos ou tarefas. Por fim, os resultados devem apresentar um intervalo de confiança de 95%. Abaixo são descritos os experimentos.

#### Experimentos Preliminares

Os experimentos preliminares possuem como finalidade responder a dúvidas inerentes ao comportamento da aplicação Teragen. Inicialmente, acreditasse que o que o tempo de execução de uma aplicação MR é diretamente ligado a sua carga de processa-

mento, como pode ser visto na Figura 4.10, dessa forma quanto maior a carga processada, maior será o tempo de execução.

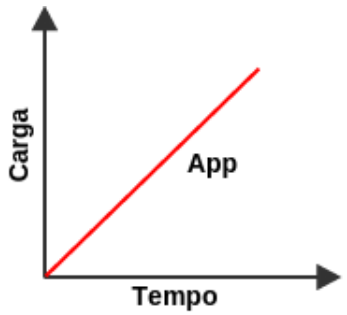



Figura 4.10 – Tempo de Execução x Carga de Trabalho

O comportamento observado na Figura 4.10 ocorre devido as fases de processamento e, o uso de recursos das aplicações MR seguirem padrões bem definidos, como pode ser visto na seção inerente a caracterização de recursos. Para demonstrar esse caso, foi conduzido um experimento utilizando a aplicação Teragen com cargas de trabalho alternadas, as execuções foram feitas de forma individual, sem a presença de contenção ou limitações, garantindo assim a utilização dos recursos em sua totalidade. Os resultados podem ser vistos na Tabela 4.7 e seguiram o resultado esperado.

Tabela 4.7 – Aplicação Sort: Carga de Trabalho x Tempo de Execução

| **Carga (GB)** | **Tempo (sec)** |
|---|---|
| 10 | 75 |
| 20 | 151 |
| 30 | 228 |

O experimento a seguir investiga se o número de tarefas de *map* influenciam no tempo de execução das aplicações MR. Para conduzir esse teste utilizou-se a aplicação Sort com carga de trabalho de 10GB. Ainda, a cada nova execução do Teragen o número de tarefas de *map* foi variado. Os resultados podem ser observados na Tabela 4.8. Nesse caso, independentemente da configuração de *maps* utilizada, se a carga de trabalho não for alterada, o tempo de execução será o mesmo.

Tabela 4.8 – # de *Maps* x Tempo de Execução

| **Maps** | 4 | 8 | 16 | 21 |
|---|---|---|---|---|
| **Tempo (sec)** | 86 | 87 | 83 | 88 |

Finalmente, sabido o comportamento da aplicação é possível seguir para o experimento principal, que abordará o uso de alocações estáticas de recursos.

Experimento Principal

Com base nos resultados preliminares, acredita-se que a aplicação que possuir a menor carga de trabalho deve ser priorizada, ou seja, a mesma deve receber uma fatia generosa dos recursos, para que assim sua execução seja acelerada. Desse modo, a contenção de disco será minimizada pelos seguintes motivos:

1. Quando a aplicação priorizada for concluída a mesma deixará de competir pelos recursos de banda de escrita em disco, minimizando assim a concorrência e possíveis problemas de interferência como a contenção de disco;

2. Efetuando o adequado gerenciamento dos recursos de disco, as aplicações em execução irão respeitar a limites pré-estabelecidos, evitando a concorrência de recursos e consequentemente a contenção de disco;

A Tabela 4.9 demonstra os valores de banda de escrita utilizados por contêiner durante a execução desse experimento. A rodada 1 representa o valor das limitações enquanto as 3 (três) aplicações estão em execução, as rodadas subsequentes são alternadas e os recursos realocados ao final da execução de cada aplicação.

Tabela 4.9 – Tabela Referência: Alocação de Recursos de Banda de Escrita

| **Aplicação** | **Carga (GB)** | **Banda de Escr. (%) Rodada 1** | **Banda de Escr. (%) Rodada 2** | **Banda de Escr. (%) Rodada 3** |
|---|---|---|---|---|
| Teragen | 10 | 70 | 0 | 0 |
| Teragen | 20 | 20 | 90 | 0 |
| Teragen | 30 | 10 | 10 | 100 |

As Figuras 4.11, 4.12, 4.13 apresentam a utilização de recursos com e sem o uso da estratégia proposta, bem como demonstram que em um ambiente livre de restrições os escalonadores tentam manter a divisão justa de recursos. Além disso, esse cenário só é alterado quando as aplicações forem concluídas, resultando na realocação de recursos entre as aplicações que ainda estiverem em execução. Já em um ambiente restrito, os resultados demonstram que ao priorizar a alocação de recursos de banda de escrita em disco é possível reduzir o MS significativamente, independente do escalonador utilizado. Isso ocorre porque ao priorizar uma aplicação os recursos são reservados ao longo do seu ciclo de vida e, durante esse ciclo os recursos podem ser consumidos em sua totalidade, livres de interferência e consequentemente de contenção. Os resultados obtidos podem ser vistos na Tabela 4.10.

Como pode ser visto na Tabela 4.10, o *noop* obteve o melhor desempenho por ser o escalonador de disco mais simples oferecido pelo Kernel, o mesmo não executa operações para classificação/ordenação de processos ou qualquer outra forma de prevenção que

Tabela 4.10 – Alocação Estática x Makespan Por Escalonador

| Estado | CFQ | deadline | noop |
|---|---|---|---|
| **Sem Restrição (s)** | 382 | 395 | 422 |
| **Com Restrição (s)** | 307 | 297 | 293 |
| **Ganho** (%) | 19,6 | 24,8 | 30,5 |

busque minimizar a latência em disco. Quando enviadas novas solicitações para esse escalonador, as mesmas são apenas inseridas em uma fila para o posterior processamento. Além disso, seu uso é ideal para aplicações MR em execução sobre contêineres, pois o escalonador *noop* foi construído para atender aplicações que efetuem acesso a disco de forma aleatória e que utilizem a bufferização e, neste caso o mesmo modo de escrita utilizado pelo Hadoop. Também pode ser observado na Tabela 4.10 que ao utilizar a estratégia de alocação estática de recursos o ambiente será controlado e consequentemente o tempo de execução das aplicações também. Na Figura 4.11 pode ser observada a utilização de recursos durante a execução do presente experimento com o escalonador *noop*.

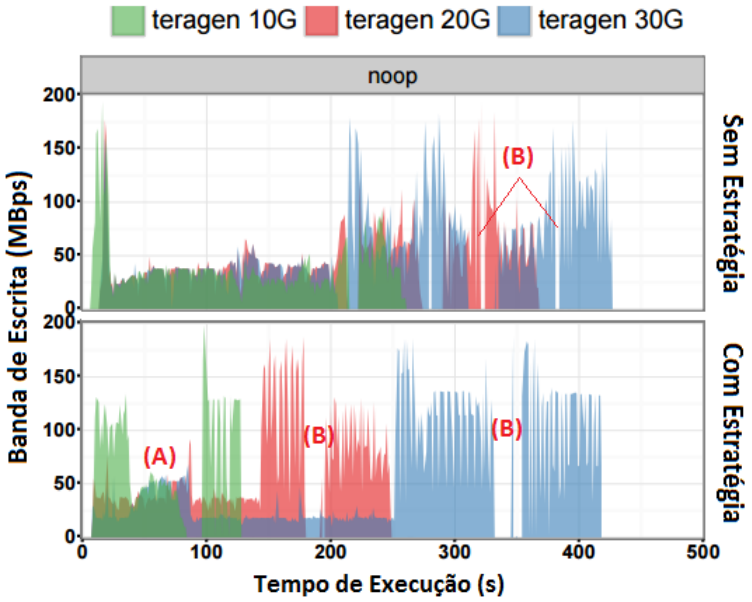



Figura 4.11 – Noop: Alocação Estática x Utilização de Recursos. (A) Representa Falhas de Isolamento e (B) Vales Sem a Utilização de Recursos de Disco

Ainda, na Figura 4.11 - (A) pode-se verificar pequenas falhas de isolamento durante a execução da aplicação Teragen 20GB. Como consequência a isso, a banda da aplicação Teragen 10GB sofre interferência, prejudicando o seu desempenho e das demais aplicações em execução. Já na Figura 4.11 - (B), apresenta-se vales sem a utilização de recursos que representam características de processamento das aplicações, possivelmente nesses momentos sejam utilizados outros recursos, como CPU ou memória. Esses vales podem ser maiores ou menores, entretanto isso depende de fatores como a utilização de recursos das aplicações em processamento, ou até quantas aplicações estão em execução. Essas observações podem ser vista nas Figuras 4.11, 4.12, 4.13.

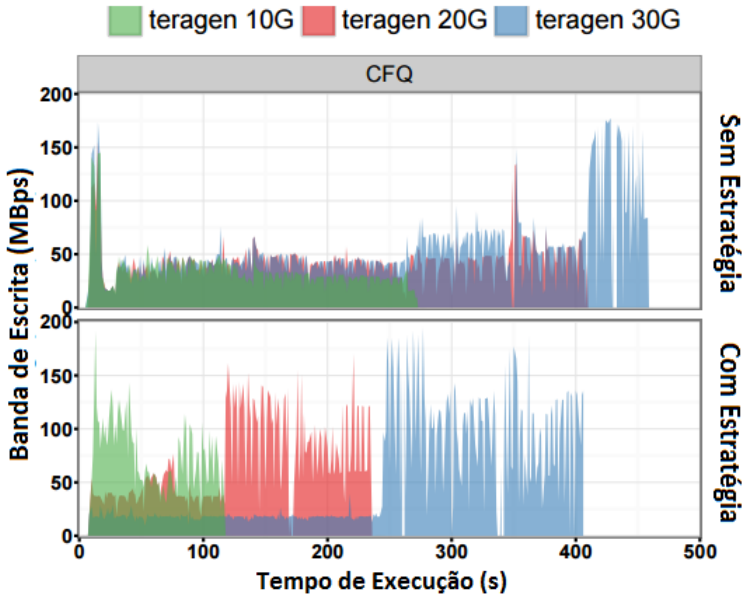



Figura 4.12 – CFQ: Alocação Estática x Utilização de Recursos.

O CFQ é o escalonador de disco mais utilizados nos dias de hoje, sendo desenvolvido para manter o compartilhamento justo de recursos através do uso de filas de processos, formadas pelo algoritmo Round Robin. No entanto, esse escalonador obteve resultados inferiores em comparação com os outros escalonadores, acredita-se que isso ocorre devido ao modelo de escrita assíncrono e bufferizado do Hadoop. Além disso, a Figura 4.12 apresenta a utilização de recursos durante a execução do presente experimento com o escalonador CFQ. As mesmas observações efetuadas para o CFQ podem ser feitas para o escalonador deadline, que por sua vez busca efetuar a divisão justa de recursos com o objetivo de evitar a contenção de disco e possíveis problemas de latência, como pode ser visto na Figura 4.13.

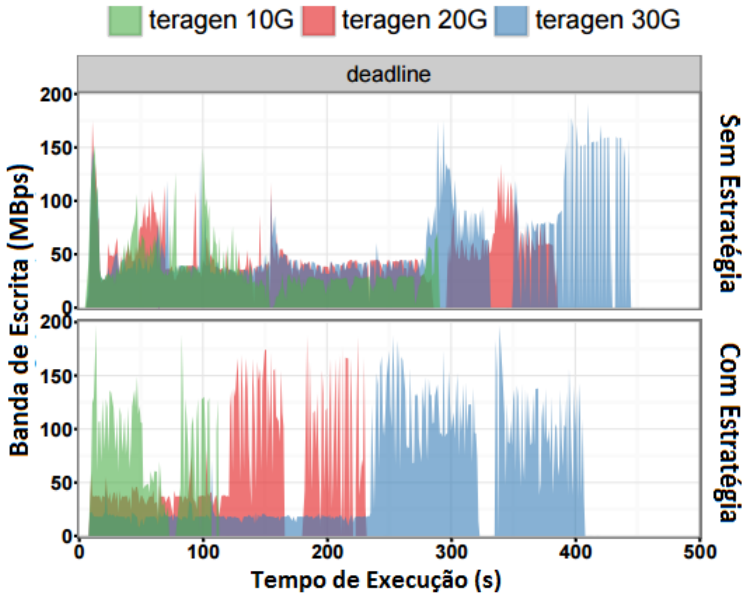



Figura 4.13 – Deadline: Alocação Estática x Utilização de Recursos

Na Tabela 4.11 é possível notar que aplicações idênticas seguem uma ordem crescente de desempenho, consequentemente os resultando seguiram a mesma ordem. No entanto, quando comparados os resultados por escalonador pode-se verificar que o tempo de execução das aplicações ficaram muito próximos. Isso possivelmente ocorre devido as cargas de trabalho possuírem o mesmo padrão de processamento. Ainda, pode ser observado que que o ganho de desempenho sobre os escalonadores CFQ/Deadline para com as aplicações que possuem carga de trabalho de 30GB não foi significativo comparado ao resultado fornecido pelo escalonador *noop*, essa diferença pode ocorrer conforme o escalonador lida com as aplicações, junto ao fato de que ao alocar somente 10% dos recursos para a aplicação pode ter penalizado a mesma com o baixo desempenho.

Tabela 4.11 – Análise Detalhada de Desempenho: Estratégia de Alocação Estática de Recursos

| **Escalonador** | **CFQ** | | | **Noop** | | | **Deadline** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Teragen** | **10GB** | **20GB** | **30GB** | **10GB** | **20GB** | **30GB** | **10GB** | **20GB** | **30GB** |
| **Contenção (s)** | 281 | 404 | 462 | 313 | 444 | 540 | 303 | 406 | 478 |
| **Alocação Estática (s)** | 175 | 291 | 456 | 155 | 276 | 448 | 148 | 299 | 445 |
| **Ganho (%)** | 37,7 | 27,9 | 1,2 | 50,4 | 37,8 | 17 | 51,1 | 26,3 | 6,9 |

Por fim, os resultados são sumarizados de forma detalhada na Tabela 4.11. Pode-se verificar ainda a eficiência da estratégia de alocação estática de recursos que obteve ganho de desempenho sobre todos os escalonadores de disco, bem como sobre a métrica de MS.

A questão de pesquisa Q4 definida como: **É possível minimizar a contenção de disco ao efetuar a restrição de recursos por contêiner?** Esse experimento abordou o uso da política de alocação estática de recursos e demonstrou que é possível acelerar as aplicações MR ao controlar o gerenciamento de recursos de disco, minimizando assim os problemas de contenção sobre ambientes virtualizados utilizando contêineres. Considera-se a questão de pesquisa respondida. Adicionalmente, o presente trabalho foi brevemente estendido e irá apresentar o uso de uma política simples para a alocação dinâmica de recursos na próxima seção.

## 4.3 Uma Estratégia Dinâmica Para a Alocação de Recursos

Buscando complementar esse trabalho, apresenta-se nessa seção uma estratégia simples, porem eficiente para a alocação dinâmica de recursos de disco, também classificamos a presente estratégia como uma solução reativa. As políticas de alocação de recursos são fundamentais para se manter o adequado gerenciamento de recursos entre aplicações

MR em execução sobre ambientes virtualizados utilizando contêineres. Nesse contexto, a presente estratégia é composta por um simples gerenciador de recursos, que irá efetuar a alocação dinâmica de banda de escrita de disco por contêiner, evitando problemas de interferência como a contenção de disco. A arquitetura do gerenciador pode ser vista na Figura 4.14.

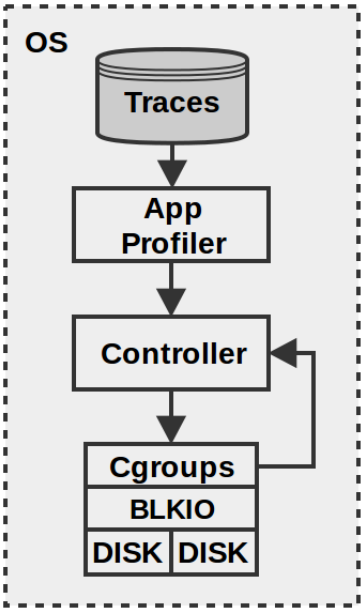



Figura 4.14 – Visão Geral: Estratégia para o Gerenciador Dinâmico de Recursos

A caixa pontilhada representa o nodo de um *cluster* e, enquanto existir VMs em execução a estratégia estará disponível. O sistema de gerenciamento é composto por:

- **Traces**: ou traços de execução fornecem informações e demonstram aspectos relevantes inerentes ao processamento e a utilização de recurso de aplicações executadas em ambientes computacionais. Ainda, os *traces* são úteis porque aplicações MR possuem padrões de processamento bem conhecidos e geralmente são executadas inúmeras vezes, o que colabora com a coleta de informações confiáveis [42];

- **Application Profiler**: para cada aplicação submetida ao *cluster*, esse módulo deverá buscar informações contidas nos *traces* (rastros de execução de aplicações) - são informações sobre a utilização de recursos, tempo de execução de aplicações já concluída e banda de escrita utilizada. Por fim, essas informações devem ser repassadas ao *controller*.

- **CGroups**: ajuda a obter informações do sistema, como o número de VMs em execução, uso de CPU, memória, rede e disco, no que diz respeito ao total da largura de banda de escrita disponível por dispositivo, Total Disk Bandwidth (TDB). Estas informações são utilizadas pelo controlador para alimentar o Algoritmo 4.3;

- **Blkio and Disk**: é um subsistema do CGroups capaz de controlar a largura de banda de escrita em disco, utilizando alocações realizadas pelo controlador;
- **Controller**: o controlador desenvolvido é composto pelo Algoritmo 4.3 que indica como as aplicações em execução devem ser priorizados, auxiliando na tarefa de evitar a contenção de disco e acelerando as aplicações em execução.

Quando iniciado o Algoritmo 4.3, alguns cálculos devem ser realizados, como por exemplo o *threshold* (linha 2) que representa a divisão justa de recursos de banda de escrita - divisão justa da banda máxima de escrita em disco pelo número de aplicações em execução. Contudo, ainda é necessário saber qual das aplicações pode gerar o maior nível de contenção. Para realizar tal tarefa a função Bdw_UsageAvg foi criada. O cálculo visto na Fórmula 4.1 obtém a média de uso de banda de escrita para cada aplicação em execução no *cluster* com base em um *trace* previamente criado e o tempo de execução (ET).

$$Bdw\_UsageAvg_{App} = \frac{\sum_{i=1}^{ET} Bandwidth}{ET} \tag{4.1}$$

As bandas médias de todas as aplicações são obtidas e ordenadas em forma crescente, caso a média obtida ( Bdw_UsageAvg) seja maior que o *threshold* a aplicação será penalizada e receberá somente 10% dos recursos, o suficiente para evitar erros de execução e não parar totalmente o processamento. Caso existam mais aplicações nesse contexto, as mesmas vão receber parcelas igualitárias sobre a fatia dos 10%. Além disso, aplicações com alta utilização de recursos se priorizadas em meio a um ambiente compartilhado podem monopolizar os recursos por muito tempo, prejudicando a execução das demais aplicações.

**Algorithm 4.1** Arquitetura Conceitual: Estratégia de Alocação Dinâmica de Recursos

**Data:** # Aplicações em execução; Traces; Informações do sistema (Banda disco)
**Result:** Percentual e alocação de recursos por contêiner

```
1:  Main_App_priority ← NULL;
2:  Threshold ← TDB/NumApp;
3:  SortApp(Bdw_UsageAvg(get_trace(App)));
4:  for cada aplicação do
5:      if UsageAvg < Threshold then
6:          if existApp > Threshold then
7:              Second_App_priority ← App;
8:              AppAllocation ← Min_band;
9:          else
10:             Second_App_priority ← App;
11:             AppAllocation ← Threshold;
12:     else
13:         if Main_App_priority == NULL then
14:             Main_App_priority ← App;
15:             AppAllocation ← Max_band;
16:         else
17:             AppAllocation ← Min_band;
```

A aplicação com a menor média e que, seja inferior ao *threshold* irá receber a fatia de 90% dos recursos de banda de escrita em disco. Desse modo, a aplicação irá deter a maior fatia de recursos e será concluída brevemente, minimizando a contenção de disco ao descongestionar o ambiente. Esse ciclo se repete até que não existam mais aplicações em execução.

### 4.3.1 Alocação Dinâmica de Recursos

Diferentemente do experimento da alocação estática de recursos que utilizou aplicações idênticas com cargas de trabalho intensivas e lineares, no presente experimento simulou-se um ambiente real com aplicações diversificadas e com cargas de trabalho alternadas. O experimento irá utilizar a estratégias acima descrita, utilizando a métrica MS e obedecendo o intervalo de confiança de 95%. As aplicações foram escolhidas com base na caracterização efetuada na Seção 4.2.2, e podem ser observadas na Tabela 4.12.

Tabela 4.12 – Aplicações Utilizadas no Experimento

| Aplicação | Carga | Utilização do Disco |
|---|---|---|
| **Terasort** | 20GB | Intensiva |
| **Sort** | 1.1GB | Moderada/Intensiva |
| **K-means** | 3GB | Baixa |

Os resultados obtidos no presente experimento são demonstrados na Tabela 4.13. Os mesmos indicam novamente que o escalonador *noop* é a melhor opção para o proces-

samento das aplicações MR, esse apresentou um ganho no MS de até 26%. Também pode ser observada pequena queda desempenho quando comparada a estratégia de alocação estática de recursos que obteve ganho no MS de aproximadamente 31%.

A diferença de desempenho possivelmente ocorre por utilizarmos aplicações com diferentes padrões de utilização de recursos e de processamento. Ainda, ao lidar com um ambiente heterogêneo deve-se levar em conta que a utilização de recursos é aleatória, ou seja, as aplicações geralmente podem fazer uso de recursos em diferentes momentos e, mesmo que o ambiente esteja totalmente restrito e livre de interferências os problemas de subutilização/sobre-utilização de recursos podem ocorrer, levando o *cluster* ao estado de contenção.

Tabela 4.13 – Escalonador de disco x Makespan

| **Estado** | **CFQ** | **deadline** | **noop** |
|---|---|---|---|
| **Sem Restrição** (s) | 1614 | 1381 | 1525 |
| **Com Restrição** (s) | 1448 | 1131 | 1123 |
| **Ganho** (%) | 10 | 18 | 26 |

As Tabelas 4.14, 4.15 apresentam resultados detalhados inerentes ao experimento realizado, a métrica a ser observada para esses casos é o tempo de execução. Diferentemente da Tabela 4.11 os resultados aqui apresentam algumas particularidades. Nota-se que a mesma aplicação possui seu desempenho alternado conforme o escalonador utilizado, por exemplo a aplicação K-means obteve melhor desempenho em um cenário de contenção ao utilizar o CFQ, possivelmente pelo fato de que esse escalonador utiliza a política de divisão justa de recursos. Isso indica que a carga de trabalho está fortemente ligada ao consumo dos recursos e, também sugere uma nova frente de pesquisa em que se investigue a implicação das aplicações e suas cargas de trabalho sobre os escalonadores de disco.

Tabela 4.14 – Análise Detalhada de Desempenho: Estratégia de Alocação Dinâmica de Recursos (A)

| **Escalonador** | | **CFQ** | | | **Noop** | |
|---|---|---|---|---|---|---|
| **Teragen (s)** | **K-means** | **Sort** | **Terasort** | **K-means** | **Sort** | **Terasort** |
| **Contenção (s)** | 877 | 1252 | 2713 | 1417 | 1158 | 2000 |
| **Alocação Dinâmica(s)** | 737 | 859 | 2749 | 676 | 564 | 2131 |
| **Ganho (%)** | 15,9 | 31,3 | -1,6 | 52,2 | 51,2 | -6 |

Com exceção da aplicação Terasort, as demais aplicações apresentam tempos de execução semelhantes, indicando que ao gerenciar recursos dinamicamente pode-se possuir um ambiente de processamento controlado. Já os resultados obtidos pela aplicação Terasort indicam um cenário de depreciação que decorre da possível sobre-utilização/subutilização

de recursos para com as aplicações Sort e K-means, ou seja, mesmo utilizando a presente estratégia esse comportamento indica que a alocação dinâmica deve ocorrer conforme a demanda (flutuação) das aplicações e não sobre limites variáveis, como os utilizados na presente estratégia. Ainda, ficou claro que, mesmo controlando a contenção ao penalizar a aplicação Terasort com apenas 10% de recursos, esse percentual é insuficiente para o processamento da mesma, indicando mais uma vez a necessidade de efetuar o gerenciamento de recursos com base na demanda das aplicações.

Tabela 4.15 – Análise Detalhada de Desempenho: Estratégia de Alocação Dinâmica de Recursos (B)

| **Escalonador** / **Aplicação** | **Deadline** | | |
|---|---|---|---|
| | **K-means** | **Sort** | **Terasort** |
| **Contenção (s)** | 1287 | 987 | 1546 |
| **Alocação Dinâmica(s)** | 699 | 522 | 1871 |
| **Ganho (%)** | 45,6 | 47,1 | -20 |

Finalmente, se pode observar que a estratégia obteve ganho de desempenho tanto ao avaliar o MS quanto avaliando as aplicações individualmente. Por conta disso, os resultados apresentados para esse trabalho são satisfatórios e respondem efetivamente a questão de pesquisa Q4.

## 4.4 Considerações

Esse capítulo apresentou um estudo experimental que busque a utilização de estratégias de alocação de recursos para minimizar a contenção de disco e ao mesmo tempo obter melhor desempenho, para que isso fosse possível diversos experimentos foram realizados, colocando duas estratégias à prova. Tanto a alocação estática, quanto a alocação dinâmica de recursos se mostraram eficientes e satisfatórias.

A primeira estratégia abordou a alocação estática de recursos, a qual foi aplicada sobre um cenário utilizando aplicações semelhantes com cargas variáveis, os resultados se mostraram satisfatórios e serviram como base para a estratégia de alocação dinâmica, que por sua vez foi aplicada sobre um cenário real composto por aplicações e cargas diversificadas. Como esperado, resultados satisfatórios também foram atingidos.

Acredita-se ainda que a estratégia possa ser aperfeiçoada para gerenciar os recursos de banda de escrita em disco conforma a demanda das aplicações e para atingir um maior número de nós. Por fim, o presente trabalho conseguiu chegar ao seu objetivo, ao minimizar a contenção de disco foi possível acelerar as aplicações MR em execução sobre ambientes virtualizados utilizando contêineres.

# 5. CONCLUSÕES E PERSPECTIVAS

A evolução tecnológica é responsável pela criação de enormes conjuntos de dados, repletos de informações valiosas e que exigem na mesma velocidade soluções de processamento rápidas e eficazes, como é o caso do modelo MR que oferece o processamento paralelo e distribuído em larga escala. Nesse contexto, a literatura aponta a tendência em utilizar contêineres para aplicações HPC em execução sobre ambientes virtualizados. Contudo, o uso desse modelo de virtualização trouxe à tona diversos problemas já resolvidos para as tecnologias tradicionais de virtualização, como por exemplo, o tópico relacionado a interferência (contenção de disco).

Nesse sentido, esse trabalho teve como objetivo responder a seguinte hipótese de pesquisa; é possível acelerar aplicações MR minimizando a contenção de disco em contêineres. Para responder a tal questionamento, quarto questões de pesquisa foram criadas e posteriormente respondidas com base em um estudo experimental que, com o uso de políticas para o gerenciamento do recurso de escrita demonstrou que é possível minimizar a contenção de disco e também de acelerar as aplicações MR executadas sobre contêineres.

Desta maneira, as seguintes questões de pesquisa foram respondidas:

1. **Como os recursos computacionais são consumidos pelas aplicações MapReduce?**

   Os experimentos demonstram que aplicações MR não possuem o mesmo comportamento em termos de utilização de recursos de disco.

2. **A consolidação de aplicações MR em contêineres conduz a perda de desempenho devido a contenção de disco?**

   Com base nos experimentos realizados foi comprovado que a contenção de disco influência no desempenho de aplicações MR executadas simultaneamente sobre contêineres.

3. **É possível efetuar restrições em disco a fim de limitar o uso desses recursos para aplicações MR em execução sobre contêineres?**

   Apesar das dificuldades encontradas para prover um ambiente confiável para a execução dos testes, os resultados demonstram que as restrições de banda de disco são respeitadas, bem como é possível se obter isolamento ao executar uma aplicação MR utilizando contêineres.

4. **É possível minimizar a contenção de disco ao efetuar a restrição de recursos por contêiner?**

O uso de políticas para a alocação de recursos demonstrou que é possível acelerar as aplicações MR ao controlar o gerenciamento de recursos de disco, minimizando assim os problemas de contenção sobre ambientes virtualizados utilizando contêineres.

## 5.1 Contribuições

As principais contribuições deste trabalho são:

- Caracterização de utilização de recursos de disco: demonstra o rastro de execução detalhado da utilização de cada recurso e de inúmeras aplicações Big Data comumente utilizadas. A caracterização pode ser muito útil para trabalhos futuros que necessitem compreender a utilização de recursos;
- Solução transparente: essa proposta busca se diferenciar das existentes por empregar de forma transparente o gerenciamento dos recursos de disco, ou seja, a solução não depende de alterações em *hardware/software* nos *frameworks* ou nas aplicações;
- Por ser uma solução transparente, a presente proposta pode ser utilizada facilmente por qualquer modelo de virtualização, ou seja, todos as ideias apresentadas nessa proposta podem ser representadas nos modelos tradicionais de virtualização. Esse é um diferencial nesse trabalho, pois a maioria das soluções existentes necessitam de alterações nas aplicações e *frameworks* de processamento;
- Esse trabalho demonstrou através de um estudo experimental que ao utilizar políticas de gerenciamento de recursos é possível obter ganho de desempenho e ao mesmo tempo diminuir a contenção de disco, **comprovando a hipótese de pesquisa**. Ainda, esse trabalho pode ser utilizado como parte inicial de uma nova estratégia para o gerenciamento dinâmico de recursos de disco;

## 5.2 Trabalhos Futuros

Considerando os benefícios proporcionados neste trabalho, os seguintes tópicos podem ser candidatos de um estudo em maior profundidade e de possíveis trabalhos futuros:

- Foi observado durante o desenvolvimento desse trabalho que o recurso de rede também não é adequadamente gerenciado, dessa forma uma extensão desse trabalho poderia ser desenvolvida para aprimorar a utilização desse recurso. Ainda, se faz muito importante a gerência do mesmo devido o surgimento da tecnologia SSD, com

o uso dessa tecnologia é possível efetuar a consolidação de um grande conjunto de dados em uma velocidade altíssima, porém ao consolidar os dados de modo tão rápido o gargalo nas aplicações MR passa a ser a rede;

- Efetuar uma extensão do trabalho para lidar com a abordagem de acordo de níveis de serviço (SLA);
- Existem trabalhos que propõem métodos para prever a utilização da rede em aplicações MR, essa mesma abordagem poderia ser utilizada para prever a utilização de disco [47];
- Os logs dos *frameworks* Big Data possuem muita informação que pode ser utilizada para prever a utilização de disco, seja através de modelos ou pela instrumentação do código fonte. Uma vez que sabe-se prever o futuro da utilização de recursos, pode-se fazer a alocação dinâmica com precisão;
- Os experimentos também revelaram a necessidade de um novo escalonador disco, ou de uma nova forma para escalonar recursos e aplicações. Isso porque foi identificado que aplicações MR podem possuir tempos de execução alternado ao variar o escalonador utilizado, sugere-se então o estudo aprofundados da relação das aplicações MR com os escalonadores de disco.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] Amazon. “Amazon Web Services”. 2015. [Online]. Acessado: 24/11/2015, Disponível em: https://aws.amazon.com.

[2] Amazon. “Elastic Map Reduce”. 2015. [Online]. Acessado: 24/11/2015, Disponível em: https://aws.amazon.com/pt/elasticmapreduce/.

[3] Beernaert, L.; Matos, M.; Vilaça, R.; Oliveira, R. “Automatic Elasticity in Openstack”. In: Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management, SDMCMM’12, 2012, pp. 2:1–2:6.

[4] Benchmark, I. F. “IOzone Filesystem Benchmark”. 2015. [Online]. Acessado em: 01-01-2016, Disponível em: http://www.iozone.org/.

[5] Blagodurov, S.; Fedorova, A. “Optimizing Shared Resource Contention in HPC Clusters”, *Super Computing*, 2013, pp. 1–25.

[6] Borthakur, D. “HDFS Architecture Guide”, Relatório Técnico, The Apache Software Fundation, 2014.

[7] Chandarana, P.; Vijayalakshmi, M. “Big Data Analytics Frameworks”. In: Proceedings of the International Conference on Circuits, Systems, Communication and Information Technology Applications, CSCITA’14, 2014, pp. 430–434.

[8] Containers, L. “Linux Containers (LXC)”. 2015. [Online]. Acessado em: 30-06-2015, Disponível em: https://linuxcontainers.org/.

[9] Controller, B. “Manual Blkio”. 2015. [Offline]. Acessado em: 15-12-2015, Disponível em: https://www.kernel.org/doc/Documentation/cgroups/blkio-controller.txt.

[10] Coutinho, E.; Sousa, F. R.; Gomes, D. G.; Souza, J. “Elasticidade em Computação na Nuvem: Uma Abordagem Sistemática”, *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos SBRC’13*, May 2013, pp. 1–44.

[11] Dean, J.; Ghemawat, S. “MapReduce: Simplified Data Processing on Large Clusters”. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation, OSDI’04, 2004, pp. 10–10.

[12] Des Ligneris, B. “Virtualization of Linux Based Computers: The Linux-Vserver Project”. In: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications, HPCS’05, 2005, pp. 340–346.

[13] Foundation, A. S. “Apache Flink”. 2016. [Online]. Acessado em: 03-05-2016, Disponível em: https://flink.apache.org/.

[14] Foundation, A. S. “Apache Mesos”. 2015. [Online]. Acessado em: 30-06-2015, Disponível em: http://mesos.apache.org/.

[15] Foundation, A. S. “Apache Spark”. 2015. [Online]. Acessado em: 30-06-2015, Disponível em: http://spark.apache.org/.

[16] Foundation, A. S. “Apache Storm”. 2015. [Online]. Acessado em: 26-11-2015, Disponível em: http://storm.apache.org/.

[17] Foundation, A. S. “Apache YARN”. 2015. [Online]. Acessado em: 30-06-2015, Disponível em: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

[18] Galante, G.; de Bona, L. “A Survey on Cloud Computing Elasticity”. In: Proceedings of the IEEE 50th International Conference on Utility and Cloud Computing, UCC’12, 2012, pp. 263–270.

[19] Hindman, B.; Konwinski, A.; Zaharia, M.; Ghodsi, A.; Joseph, A. D.; Katz, R. H.; Shenker, S.; Stoica, I. “Mesos: A platform for fine-grained resource sharing in the data center.” In: Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation, NSDI’11, 2011, pp. 22–22.

[20] Hortonworks. “Virtualização com Hadoop”. 2015. [Offline]. Acessado em: 15-12-2015, Disponível em: http://br.hortonworks.com/partners/virtualization/.

[21] Huang, S.; Huang, J.; Dai, J.; Xie, T.; Huang, B. “The HiBench Benchmark Suite: Characterization of The MapReduce-Based Data Analysis”. In: Proceedings of the IEEE 26th International Conference on Data Engineering Workshops ICDEW’10, 2010, pp. 41–51.

[22] Hwang, k.; Dongarra, J.; Fox, G. “Virtualização: Clusters Físicos vs. Clusters Virtuais”, Relatório Técnico, Microsoft Inc, 2012.

[23] Ibrahim, S.; Jin, H.; Lu, L.; He, B.; Wu, S. “Adaptive Disk I/O Scheduling for MapReduce in Virtualized Environment”. In: Proceedings of the International Conference on Parallel Processing. ICPP’11., 2011, pp. 335–344.

[24] Ibrahim, S.; Jin, H.; Lu, L.; Qi, L.; Wu, S.; Shi, X. “Evaluating MapReduce on Virtual Machines: The Hadoop Case”. In: Proceedings of the 1st International Conference on Cloud Computing. CloudCom’09, 2009, pp. 519–528.

[25] Jersak, L. C. “Mapeamento de Máquinas Virtuais em Datacenters Privados Visando Minimizar a Interferência de Desempenho”. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS. 2014. pp, 65.

[26] Kambatla, K.; Pathak, A.; Pucha, H. “Towards Optimizing Hadoop Provisioning in The Cloud”. In: Proceedings of the USENIX Association on 2009 Conference on Hot Topics in Cloud Computing, HotCloud’09, 2009, pp. 5.

[27] Koh, Y.; Knauerhase, R.; Brett, P.; Bowman, M.; Wen, Z.; Pu, C. “An Analysis of Performance Interference Effects in Virtual Environments”. In: Proceedings of the IEEE International Symposium on Performance Analysis of Systems Software, ISPASS’07., 2007, pp. 200–209.

[28] Machine, K. V. “Kernel Virtual Machine (KVM)”. 2015. [Online]. Acessado em: 30-06-2015, Disponível em: http://www.linux-kvm.org/page/Main_Page.

[29] Malensek, M.; Pallickara, S.; Pallickara, S. “Alleviation of Disk I/O Contention in Virtualized Settings for Data-Intensive Computing”, Relatório Técnico, The Galileo Project: Department of Computer Science Colorado State University, 2015.

[30] Marcos D, A.; Rodrigo N, C.; Bianchi, S.; Marco A, N.; Buyya, R. “Big Data Computing and Clouds: Trends and Future Directions”, *Parallel and Distributed Computing. Science Direct*, vol. 79–80, May 2015, pp. 3–15.

[31] Neves, M. V. “Dstat Monitor”. 2015. [Online]. Acessado em: 15-12-2015, Disponível em: https://github.com/mvneves/dstat-monitor.

[32] Neves, M. V. “Application-aware Software-Defined networking to Accelerate Mapreduce Applications”. Tese de Doutorado, Programa de Pós-Graduação em Ciência da Computação, PUCRS. 2015. pp, 121.

[33] Oliveira. “Aprimorando a Elasticidade de Aplicações de Bando de Dados Utilizando Virtualização em Nível de Sistema Operacional”. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS. 2014. pp, 95.

[34] Polato, I.; Ré, R.; Goldman, A.; Kon, F. “A Comprehensive View of Hadoop Research—A Systematic Literature Review”, *Journal of Network and Computer Applications*, vol. 46, 2014, pp. 1–25.

[35] Potter; Kenneth, H. “Dynamic Addressing Mapping to Eliminate Memory Resource Contention in A Symmetric Multiprocessor System”. Google Patents. US Patent 6,505,269, Jan 7 2003.

[36] Project, X. “XEN Project”. 2015. [Online]. Acessado em: 30-06-2015, Disponível em: http://xenserver.org/.

[37] Pu, X.; Liu, L.; Mei, Y.; Sivathanu, S.; Koh, Y.; Pu, C. “Understanding Performance Interference of I/O Workload in Virtualized Cloud Environments”. In: Proceedings of

the IEEE 3rd International Conference on Cloud Computing, CLOUD'10, 2010, pp. 51–58.

[38] Raj, P.; Raman, A.; Nagaraj, D.; Duggirala, S. "High-Performance Big-Data Analytics: Computing Systems and Approaches". Springer. 2015. pp. 428.

[39] Raveendran, A.; Bicer, T.; Agrawal, G. "A Framework for Elastic Execution of Existing MPI Programs". In: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, IPDPSW'11, 2011, pp. 940–947.

[40] Sharma, B.; Prabhakar, R.; Lim, S.; Kandemir, M.; Das, C. "MROrchestrator: A Fine-Grained Resource Orchestration Framework for MapReduce Clusters". In: Proceedings of the IEEE 5th International Conference on Cloud Computing, CLOUD'12, 2012, pp. 1–8.

[41] Siyuan, M.; Sun, X.-H.; Raicu, I. "I/O Throttling and Coordination for MapReduce", Relatório Técnico, Illinois Institute of Technology, 2012.

[42] Song, G.; Meng, Z.; Huet, F.; Magoules, F.; Yu, L.; Lin, X. "A Hadoop MapReduce Performance Prediction Method". In: Proceedings of the 2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing. HPCC_EUC'13, 2013, pp. 820–825.

[43] Sousa, F. R.; Moreira, L. O.; Machado, J. C. "Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios", *II Escola Regional de Computação Ceará, Maranhão e Piauí ERCEMAPI'09*, 2009, pp. 150–175.

[44] Stillwell, M.; Schanzenbach, D.; Vivien, F.; Casanova, H. "Resource Allocation Using Virtual Clusters". In: Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. CCGRID'09, 2009, pp. 260–267.

[45] V-Server, L. "Linux V-Server". 2015. [Online]. Acessado em: 15-12-2015, Disponível em: linux-vserver.org/.

[46] Vavilapalli, V. K.; Murthy, A. C.; Douglas, C.; Agarwal, S.; Konar, M.; Evans, R.; Graves, T.; Lowe, J.; Shah, H.; Seth, S.; Saha, B.; Curino, C.; O'Malley, O.; Radia, S.; Reed, B.; Baldeschwieler, E. "Apache Hadoop YARN: Yet Another Resource Negotiator". In: Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC'13, 2013, pp. 1–16.

[47] Veiga Neves, M.; De Rose, C.; Katrinis, K.; Franke, H. "Pythia: Faster Big Data in Motion Through Predictive Software-Defined Network Optimization at Runtime". In: Proceedings of the IEEE 28th International on Parallel and Distributed Processing Symposium, IPDPS'14, 2014, pp. 82–90.

[48] Virtuozzo, O. “OpenVZ”. 2015. [Online]. Acessado em: 30-06-2015, Disponível em: https://openvz.org/.

[49] VMware. “VMware”. 2015. [Online]. Acessado em: 25-07-2015, Disponível em: www.vmware.com/.

[50] Wei, Y.; Karthik, K.; Bc, W.; Todd, L. “Disk I/O Isolation Scheduling 3”, Relatório Técnico, The Apache Software Fundation, 2014.

[51] Whitcroft, A. “Buffered Write IO Controller in Balance Dirty Pages”. 2015. [Online]. Acessado em: 15-12-2015, Disponível em: http://comments.gmane.org/gmane.linux.kernel.mm/76183.

[52] White, T. “Hadoop: The Definitive Guide”. O’Reilly Media, Inc. 2009. 3rd. pp. 657.

[53] Xavier, M.; De Oliveira, I.; Rossi, F.; Dos Passos, R.; Matteussi, K.; De Rose, C. “A Performance Isolation Analysis of Disk-Intensive Workloads on Container-Based Clouds”. In: Proceedings of The 23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP’15., 2015, pp. 253–260.

[54] Xavier, M.; Neves, M.; De Rose, C. “A Performance Comparison of Container-Based Virtualization Systems for Mapreduce Clusters”. In: Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP’14., 2014, pp. 299–306.

[55] Xavier, M.; Neves, M.; Rossi, F.; Ferreto, T.; Lange, T.; De Rose, C. “Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments”. In: Proceedings of the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP’13., 2013, pp. 233–240.

[56] Xavier, M. G. “Uma Arquitetura Para Provisionamento de Ambientes de Alto desempenho customizados”. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS. 2013. pp, 103.

[57] Xu, Y.; Suarez, A.; Zhao, M. “IBIS: Interposed Big-Data I/O Scheduler”. In: Proceedings of the 22Nd International Symposium on High-performance Parallel and Distributed Computing. HPDC’13, 2013, pp. 109–110.

[58] Yang, Z.; Fang, H.; Wu, Y.; Li, C.; Zhao, B.; Huang, H. “Understanding the Effects of Hypervisor I/O Sheduling for Virtual Machine Performance Interference”. In: Proceedings of the IEEE 4th International Conference on Cloud Computing Technology and Science CloudCom’12., 2012, pp. 34–41.

[59] Yazdanov, L.; Gorbunov, M.; Fetzer, C. “EHadoop: Network I/O Aware Scheduler for Elastic MapReduce Cluster”. In: Proceedings of the IEEE 8th International Conference onCloud Computing CLOUD’15., 2015, pp. 821–828.

[60] YongChul, K.; Balazinska, M.; Howe, B.; Rolia, J. “A Study of Skew in Mapreduce Applications”. In: Proceedings of the 5th Open Cirrus Summit. OCS’11 on, 2011, pp. 11.

[61] YongChul, K.; Balazinska, M.; Howe, B.; Rolia, J. “SkewTune: Mitigating Skew in MapReduce Applications”. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. SIGMOD’12, 2012, pp. 25–36.

[62] Zaharia, M.; Konwinski, A.; Joseph, A. D.; Katz, R. H.; Stoica, I. “Improving MapReduce Performance in Heterogeneous Environments.” In: Proceedings of the 4th USENIX Symposium on Operating Systems Design and Implementation. OSDI’04 on, 2008, pp. 7.

[63] Zhuravlev, S.; Blagodurov, S.; Fedorova, A. “Addressing Shared Resource Contention in Multicore Processors via Scheduling”. In: Proceedings of the 15th Architectural Support for Programming Languages and Operating Systems ASPLOS’10 on, 2010, pp. 129–142.