PUCRS

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

FÁBIO MIGUEL BLASAK DA FONSECA

OTIMIZANDO A EXECUÇÃO DE APLICAÇÕES DE BANCO DE DADOS ATRAVÉS DE UMA MELHOR ALOCAÇÃO DE RECURSOS DE DISCO EM AMBIENTES VIRTUALIZADOS

Porto Alegre
2017

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

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

OTIMIZANDO A EXECUÇÃO DE APLICAÇÕES DE BANCO DE DADOS ATRAVÉS DE UMA MELHOR ALOCAÇÃO DE RECURSOS DE DISCO EM AMBIENTES VIRTUALIZADOS

FÁBIO MIGUEL BLASAK DA FONSECA

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. Dr. César Augusto Fonticielha De Rose

Porto Alegre
2017

Ficha Catalográfica

B644o Blasak da Fonseca, Fábio Miguel

Otimizando a execução de aplicações de banco de dados através de uma melhor alocação de recursos de disco em ambientes virtualizados / Fábio Miguel Blasak da Fonseca . – 2017.

123 f.

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

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

1. alocação de recursos. 2. virtualização. 3. contenção de disco. 4. banco de dados. I. Fonticielha De Rose, César Augusto. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecário responsável: Marcelo Votto Texeira CRB-10/1974

Fábio Miguel Blasak da Fonseca

Otimizando a execução de aplicações de banco de dados através de uma melhor alocação de recursos de disco em ambientes virtualizados

Tese/Dissertação apresentada como requisito parcial para obtenção do grau de Doutor/Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciencia da Computação, Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 11 de Agosto de 2017.

BANCA EXAMINADORA:

Prof. Dr. Lucas Mello Schnorr (PPGC/UFRGS)

Prof. Dr. Tiago Coelho Ferreto (PPGCC/PUCRS)

Prof. Dr. César Augusto Fonticielha de Rose (PPGCC/PUCRS - Orientador)

DEDICATÓRIA

Dedico este trabalho aos meus pais: Vilson Miguel da Fonseca e Teresinha da Fonseca. Também gostaria de dedicar a minha esposa Tatiane Schwanck Schardosim a qual sem seu suporte, nada seria possível.

“To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science.”
(Albert Einstein)

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, a oportunidade que tu me proporcionaste, foi e está sendo a melhor experiência da minha vida até o momento. Aos amigos do PPGCC e a toda galera do LAD/PUCRS, em especial à Kassiano José Matteussi ao qual me ajudou bastante nas fases iniciais da implementação do meu projeto. À DELL e a PUCRS pela concessão da bolsa de estudo.

OTIMIZANDO A EXECUÇÃO DE APLICAÇÕES DE BANCO DE DADOS ATRAVÉS DE UMA MELHOR ALOCAÇÃO DE RECURSOS DE DISCO EM AMBIENTES VIRTUALIZADOS

RESUMO

A crescente necessidade de extensão dos recursos de TI (Tecnologia da Informação) para atender as demandas do negócio, geraram uma preocupação de como aumentar a capacidade com menor custo e maior aproveitamento do *data center*. Portanto, a fim de evitar a subutilização de recursos de infraestrutura a virtualização é uma tendência para redução de custos e consolidar a infraestrutura de servidores, aproveitando assim os ativos existentes. Entretanto, com o crescimento da virtualização, surge um problema relacionado a concorrência por recursos em ambientes consolidados, onde aplicações com uso intensivo de disco, como bancos de dados, podem ser impactados neste tipo de ambiente, caso não tenham os seus recursos gerenciados apropriadamente, podendo gerar degradação no desempenho e consequentemente aumentando o tempo de execução.

A fim de otimizar performance e reduzir a contenção de E/S (Entrada/Saída), Kassiano J. M. [19] apresentou um estudo sobre a aceleração de aplicações *Hadoop* através de ajuste manual na alocação de recursos de disco, mostrando que é possível obter ganhos de performance. Logo, o trabalho proposto, segue esta linha de estudo, entretanto, com o objetivo de otimizar a execução de aplicações de banco de dados em ambientes virtualizados com recursos compartilhados, aplicando uma política de ajuste dinâmico de alocação de recursos de disco, a qual visa acelerar ainda mais os ganhos de performance. Essa política tem por objetivo distribuir os recursos de disco de forma otimizada, conforme algoritmo aplicado, evitando que um ou mais processos consumam todos os recursos de disco, enquanto outros aguardam para serem executados ou executam com o mínimo de recursos de disco apropriados, por isso, levando maior tempo para concluir o processamento. Para evidenciar esta situação, foram avaliados *workloads* de banco de dados do tipo OLTP (*Online Transac-*

tion Processing) e DW (*Data Warehouse*), utilizando o simulador de cargas de dados Orion [24] e com dados reais capturados de um teste de carga cedidos por uma empresa de TI de grande porte, em parceria com a universidade PUCRS (Pontifícia Universidade Católica do Rio Grande do Sul), através do recurso *Oracle RAT (Real Application Testing)* [25].

Foram realizados testes em laboratório utilizando os seguintes cenários de teste: sem ajuste de recursos de disco, com ajuste estático de recursos de disco e através de uma política de ajuste dinâmico de recursos de disco com base em métricas de performance. A partir disso, pode-se observar que a política dinâmica obteve o melhor resultado entre os demais grupos de teste, gerando um ganho de 23% para a execução de *workloads* de banco de dados OLTP, 21% para *workloads* de banco de dados DW e 18% durante a execução de ambientes com *workloads* de tipos diferentes em concorrência, exemplo: DW e OLTP.

Palavras-Chave: alocação de recursos, virtualização, contenção de disco, banco de dados.

OTIMIZANDO A EXECUÇÃO DE APLICAÇÕES DE BANCO DE DADOS ATRAVÉS DE UMA MELHOR ALOCAÇÃO DE RECURSOS DE DISCO EM AMBIENTES VIRTUALIZADOS

ABSTRACT

The growing need to extend IT (Information Technology) resources to meet business needs has raised concerns about how to increase capacity with lower cost and greater use of data center. Therefore, in order to avoid underutilization of infrastructure resources virtualization is a trend towards cost reduction and consolidation of the server infrastructure, thus taking advantage of existing assets. However, with virtualization growth, there is a problem related to resources concurrence in consolidated environments, where disk-intensive applications such as databases can be impacted in this type of environment, if they do not have their resources managed properly, can generate performance degradation and increasing execution time respectively.

In order to optimize performance and reduce I/O contention, Kassiano J.M. [19] presented a study on the acceleration of Hadoop applications through manual adjustment of disk resource allocation, showing that it is possible to get performance gains. Therefore, proposed work follows this line of study, however, with objective of optimizing the execution of database applications in virtualized environments with shared resources, applying a dynamic adjustment policy of disk resources allocation. It aims to distribute disk resources optimally through an algorithm, avoiding that one or more processes consume all disk resources, while others wait to be executed or are being executed without minimum of appropriate disk resources, thus, taking more time to complete their execution. In order to demonstrate this scenario, workloads of OLTP (Online Transaction Processing) and DW (Data Warehouse) databases have been evaluated using the Orion data load simulator [24] and real captured data from a loading test provided by a large IT company in partnership with PUCRS Uni-

versity (Pontifical Catholic University of Rio Grande do Sul), through the Oracle RAT (Real Application Testing) [25].

Laboratory tests have been performed using the following test scenarios: without adjustment of disk resources, with static adjustment of disk resources and through a dynamic adjustment policy of disk resources based on performance metrics. In this case, it can be observed that dynamic policy obtained the best result among the other test groups, generating a gain of 23% for OLTP database workloads, 21% for DW database workloads and 18% for environments with different types of workloads in concurrency like DW and OLTP.

Keywords: resource allocation, virtualization, disk contention, databases.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

BPS – Bits per Second

CPU – Central Processing Unit

ERP – Enterprise Resource Planning

ETL – Extract, Transform and Load

HPC – High-Performance Computing Clusters

IAAS – Infrastructure as a Service

NIST – National Institute of Standards and Technology

OLTP – Online Transaction Processing

PAAS – Platform as a Service

PC – Personal Computers

PUCRS – Pontificia Universidade Católica do Rio Grande do Sul

QOS – Qualify of Service

RAT – Real Application Testing

SAAS – Software as a Service

SGBD – Sistema de Gerenciamento de Banco de Dados

SLA – Service Level Agreement

SSD – Solidate-state Drive

TI – Tecnologia da Informação

TPS – Transaction per Second

VMM – Virtual Machine Manager

LISTA DE ABREVIATURAS

CGroups. – Control Groups

DW. – Data Warehouse

E/S. – Entrada/Saída

etc.. – Et Cetera

LXC. – Linux Contêiner

N/A. – Não se Aplica

SO. – Sistema Operacional

TC. – Traffic Controller

TPC. – Transaction Processing Performance Council

VM. – Virtual Machine

SUMÁRIO

1. INTRODUÇÃO

Com o crescente aumento dos recursos computacionais e a disseminação da internet a necessidade de acesso a informação de forma rápida e a melhor utilização dos recursos de *hardware* e software tornaram-se uma grande preocupação. Logo, o conceito de computação em nuvem surgiu para proporcionar flexibilidade no acesso a informação, através do provisionamento de recursos de forma facilitada, possibilitando a melhor utilização dos recursos de *hardware* e *software* com o seu custo medido sob demanda e oferecendo flexibilidade ao usuário para adição/remoção de recursos computacionais [35]. Com a utilização da computação em nuvem a virtualização tornou-se parte integrante a esta arquitetura, provisionando uma abstração lógica sobre a infraestrutura e de seus recursos físicos, bem como reduzindo a complexidade na administração do ambiente. Entretanto, o uso demasiado de aplicações com grande volume de dados exigem configurações robustas de *hardware* e podem impactar outras aplicações na mesma infraestrutura, como por exemplo, *Big Data*, DW, ERP (*Enterprise Resource Planning*), dentre outras. Para sanar este impacto, o isolamento de recursos tornou-se um elemento importante para melhorar a administração dos recursos de TI.

O conceito de virtualização é uma ideia estabelecida a mais de 30 anos [12] e sua utilização foi considerada um sinônimo de perda de performance, entretanto, isso vem se modificando [21] devido a processadores mais rápidos, onde soluções de virtualização com melhor performance tornaram-se viáveis em máquinas menos robustas, como por exemplo, em computadores pessoais (PC). Com isso, o próximo passo foi a simplificação do ambiente de TI, a partir da consolidação de vários servidores *standalone* para poucos servidores utilizando configurações de *hardware* mais robustas que compartilharão estes recursos entre as diversas máquinas virtuais para diversos tipos de aplicações. Entretanto, existe um desafio para a utilização da virtualização de aplicações de banco de dados devido a performance. A partir de um estudo realizado por Xavier *et al.* [36] relacionado a uma análise de aplicações com uso intensivo de disco e seu impacto em ambientes consolidados e virtualizados, pode-se observar contenção de recursos quando existe múltiplos ambientes em execução concorrente, disputando E/S de disco para desempenhar uma atividade. Desta maneira, acredita-se que a contenção de disco pode ser mitigada com a alocação de recursos de disco mais apropriada, o que vai permitir ganhos na performance da aplicação de banco de dados sob o modelo de virtualização baseado em contêiner.

De acordo com Matteussi *et al.* [19] através de um estudo com foco em aplicações *Hadoop* em ambientes virtualizados, pode-se definir um ajuste na alocação de recursos de disco de forma estática, priorizando os recursos e possibilitando um aumento de performance dentre as diversas máquinas que compõe esta aplicação. Além deste estudo, existem outros que também tentam realizar um ajuste de alocação de disco, como forma

de gerar ganhos de performance, entretanto, nenhum deles realiza este ajuste de forma dinâmica para diversos tipos de bancos de dados, sem gerar impacto no ambiente para manutenções ou sem ter a necessidade de um *software* proprietário.

A partir disso, a proposta deste trabalho é uma política de ajuste dinâmico de recursos de disco em um ambiente consolidado composto por várias máquinas virtuais e seus respectivos banco de dados. Esta política foi baseada no modelo de Dawoud *et al.* [7] que utiliza a plataforma de virtualização VMWare permitindo uma alocação de recursos de disco, entretanto, a mesma apresentava problemas durante esta alocação, pois o ajuste era realizado apenas na inicialização das máquinas virtuais, não permitindo a adaptação do ambiente em tempo de execução, logo gerando *downtime* para a aplicação, em caso de alterações. Já a política proposta por este trabalho é baseada na plataforma LXC (Linux Contêiner) que proporciona maior flexibilidade do ajuste de recursos, não gerando impacto aos processos já em execução. Além disso, foi utilizado o recurso *Control Groups* (*cgroups*) que suporta as interações entre o *hardware* e o sistema operacional entre diferentes máquinas virtuais para definição de restrição de recursos de disco. A política tem como principais componentes de sua arquitetura: o controlador de recursos que recebe os dados dos agentes de monitoração de recursos e de performance entre as diferentes máquinas virtuais controladas por ela; o monitor de recursos que atua como um coletor das estatísticas atuais do que está sendo utilizado e o que está ainda disponível a nível de sistema operacional; monitor de performance que avalia de forma similar ao monitor de recursos, realiza a coleta de dados de utilização de recursos na camada do banco de dados, se o mesmo está ativo e se necessita uma priorização de recursos ou não para que a política seja aplicada. Assim, no componente de gerenciamento de recursos, controla os limites dos recursos do sistema operacional de disco. Mais detalhes sobre a arquitetura da política serão apresentados no decorrer deste trabalho.

Para evidenciar os ganhos do ajuste dinâmico realizado por esta política foram realizados testes em laboratório para o monitoramento dos recursos de disco durante a execução de transações no banco de dados nos seguintes cenários de ajuste de alocação de recursos de disco: sem ajuste, com ajuste estático e com ajuste dinâmico aplicando a política.

A seguir será apresentada a motivação, bem como a organização deste trabalho.

1.1 Motivação

O compartilhamento de recursos, especialmente de disco, entre diferentes aplicações em um ambiente consolidado e virtualizado, pode resultar em uma interferência na performance, devido a concorrência por E/S sem um controle apropriado destas solicitações de disco para finalizar uma tarefa ou processo na aplicação. Isso pode ser mitigado com

um controle no acesso destes recursos mais sensível. A figura 1.1 não reflete um *workload* real, mas sim um fictício sobre a demanda de utilização de recursos por diferentes aplicações para exemplificar o problema de concorrência de recursos de disco. Foi utilizado o seguinte cenário de exemplo: três aplicações em execução concorrente e com uso intensivo de disco, sendo que no gráfico à esquerda, não há nenhum controle sobre os recursos de disco, percebendo em algumas áreas do gráfico, lacunas que significam ociosidade aonde poderia ser aproveitado para execução de alguma operação de E/S, assim como em alguns momentos do gráfico aonde limite da largura de banda do disco é extrapolado devido a alta utilização e demanda por estas aplicações, gerando também atraso para que as mesmas sejam finalizadas. Em contrapartida, no lado direito do gráfico, existe um controle mais fino sobre os recursos de E/S de disco, permitindo que seja melhor aproveitada a capacidade de E/S do disco, mitigando ociosidade de recursos e permitindo que estas aplicações possam finalizar mais cedo do que o esperado devido a um melhor aproveitando de E/S do disco.

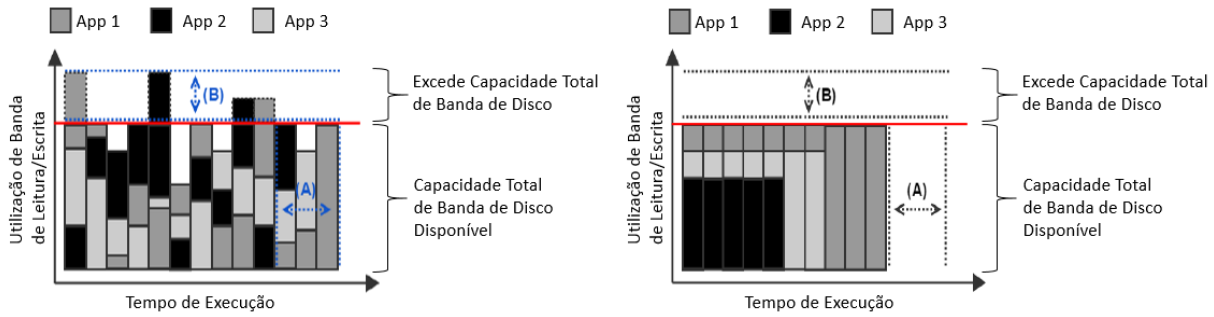

Figura 1.1 – Utilização dos recursos de disco entre aplicações em execução concorrente [19]

Um ajuste estático traz ganhos distribuindo os recursos de disco entre as aplicações conforme uma configuração, evitando que algumas utilizem 100% destes recursos, enquanto outras não recebam nenhum. Entretanto, o ajuste estático pode gerar alguns momentos de ociosidade de recursos de disco, pois não repassa estes recursos para processos ainda em execução. Um exemplo disso pode ser demonstrado através do seguinte cenário de teste: duas aplicações de mesmo tipo e que são caracterizadas com uso intensivo de disco, em execução concorrente e que estão disputando recursos de um disco compartilhado. Foi alocado de modo estático (não irá ser alterado ao longo do fluxo de execução destas aplicações), através de um recurso do sistema operacional chamado CGroups (*Control Groups*) [29], 10% dos recursos de E/S de disco para a aplicação A e 90% para a aplicação B. Como a segunda aplicação possui mais recursos de disco para ser utilizado em comparação à primeira, a mesma irá executar mais rápido e irá finalizar em um menor tempo comparado a aplicação A. Assim, por mais que a aplicação A necessite de mais recursos de E/S acima destes 10% previamente alocados, a mesma estará limitada a este percentual. Após a aplicação B finalizar, a aplicação A continuará utilizando os mesmos 10%, neste caso, haverá 90% dos recursos que estarão ociosos no servidor e não poderão

ser utilizados pela aplicação devido a esta definição fixada de recursos e que não se adapta conforme a execução destes *workloads*. Sendo assim, uma política de alocação dinâmica de recursos de disco para leitura e escrita é ideal para tratar este tipo de situação. Este trabalho tenta minimizar a contenção de disco e otimizar a performance de aplicações de banco de dados para que sejam executadas concorrentemente em ambientes virtualizados utilizando contêineres. Para isso, pretende-se ajustar os recursos de disco das aplicações aplicando uma política dinâmica. Este caso de estudo é focado em ambientes de banco de dados estruturados tradicionais com recursos de infraestrutura (CPU (*Central Processing Unit*), memória e disco) compartilhados, como por exemplo: MySQL, Oracle ou Postgres. Este cenário é uma das realidades em grandes empresas, onde existem muitos servidores em *cluster* com recursos compartilhados, sendo o banco de dados uma de suas principais aplicações para suporte ao negócio da empresa. Desta forma, o melhor uso dos recursos e a otimização da performance são critérios muito importantes para aplicações críticas. Baseado nisso, o foco deste trabalho é bancos de dados tradicionais com compartilhamento de recursos, sendo a contenção de disco o problema a ser avaliado.

1.2 Organização do Trabalho

Este trabalho está estruturado da seguinte forma: o capítulo 2 apresenta a fundamentação teórica, onde é feita a contextualização de conceitos básicos dos assuntos deste trabalho; o capítulo 3 apresenta um comparativo entre este trabalho e os trabalhos relacionados encontrados na literatura, além disso, são apresentados os trabalhos do grupo de pesquisa da PUCRS relacionados com este assunto; o capítulo 4 descreve as hipóteses e as questões de pesquisa que norteiam este trabalho, além disso, demonstra o primeiro ciclo de testes com o ajuste de recursos de disco de forma estática para responder a questão de pesquisa Q1, além disso são apresentados os testes realizados em relação aos ajustes estáticos; o capítulo 5 apresenta a política de alocação de recursos de disco dinâmica proposta, que é a contribuição deste trabalho, descrevendo os princípios gerais, a arquitetura genérica e seu funcionamento; o capítulo 6 apresenta a validação dos resultados, demonstrando a implementação da política proposta através de um algoritmo, bem como os resultados obtidos com a aplicação da política de alocação de recursos de disco proposta por este trabalho, respondendo assim a questão de pesquisa Q2; o capítulo 7 apresenta as considerações finais, contribuições e sugestões para trabalhos futuros; e por fim, o capítulo 8 apresenta os anexos utilizados ao longo deste trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Como apresentado no capítulo introdutório deste trabalho, a virtualização tornou-se uma grande tendência por diversas empresas para a simplificação da infraestrutura de TI, a fim de consolidar ambientes e apoiar na redução de custos relacionados a manutenção de servidores físicos dedicados para as aplicações de negócio. Entretanto, isso traz uma preocupação quanto a performance devido a contenção de recursos de disco com múltiplas aplicações em execução concorrente em um ambiente virtualizado e consolidado. Com o aprimoramento desta tecnologia e servidores com configurações de *hardware* mais robustos, a preocupação quanto a performance pode ser minimizada, entretanto, ainda existe um paradigma na área de banco de dados devido a resistência de empresas em utilizar a virtualização nesta área. Baseado nisso, um dos fatores que gera muita preocupação é a contenção de disco em ambientes virtualizados e com compartilhamento de recursos.

Nas próximas seções são abordados os assuntos relacionados a ambientes virtualizados e aplicações de banco de dados.

2.1 Ambientes Virtualizados

Esta seção apresenta uma contextualização referente a computação em nuvem, virtualização, isolamento de recursos, banco de dados e políticas de alocação de recursos em ambientes virtualizados.

2.1.1 Computação em Nuvem

De acordo com a definição do NIST (*National Institute of Standards and Technology*) [20], o termo computação em nuvem é um modelo que permite acesso a rede de forma conveniente e sob demanda a um conjunto compartilhado de recursos computacionais (rede, servidores, *storage*, aplicações, etc.) que podem ser configuráveis, rapidamente provisionados e habilitados com o mínimo de esforço de administração ou interação por parte do provedor de serviço. Este modelo possui cinco características essenciais: (i) Provisionamento de recursos sob demanda; (ii) Acesso através da rede por diferentes tipos de dispositivos como computadores, *tablets*, *smartphones*, etc.; (iii) Conjunto de recursos que podem ser provisionados e compartilhados entre diferentes usuários; (iv) Rápida Elasticidade - habilidade de escalar recursos computacionais – adicionar ou reduzir, de forma facilitada com um mínimo de impacto na arquitetura e; (v) Medição de serviço – ambientes de computação em nuvem automaticamente realizam o controle e otimização de recursos,

assim como, a utilização destes recursos podem ser monitorados, controlados e reportados, fornecendo transparência para ambos – provedor e cliente do serviço utilizado.

Desta forma, organizações podem se beneficiar com esta flexibilidade de alocação ou redução de recursos de infraestrutura sob demanda, aliando um suporte integrado sobre um SLA (*Service Level Agreement*) definido entre cliente e prestador de serviços. A partir disso, em uma plataforma de computação em nuvem pode haver diferentes arquiteturas para provisionamento de recursos como IAAS (*Infrastructure as a Service*), PAAS (*Platform as a Service*), SAAS (*Software as a Service*), etc., sendo consideradas como um conjunto de recursos baseados em uma infraestrutura virtualizada para fornecer flexibilidade e eficiência [33]. Estes recursos computacionais (memória, *storage*, rede, etc.) estão distribuídos em servidores físicos que são virtualizados e apresentados como múltiplas máquinas virtuais, para atender diferentes demandas do ambiente. Para alcançar um bom aproveitamento de recursos é possível utilizar técnicas como o balanceamento de carga em torno dos servidores físicos e dos *storages*. Uma das formas para se alcançar este balanceamento é através do recurso migração *online* de máquinas virtuais. Desta forma, as aplicações virtualizadas a partir de recursos físicos de infraestrutura são distribuídos entre servidores de forma transparente ao usuário, sem interrupção do serviço fornecido pela nuvem [33]. Baseado nisso, a virtualização torna-se um componente primordial para proporcionar flexibilidade na criação de novos ambientes, assim como adição e remoção de recursos de infraestrutura (CPU, memória, etc.) sem a percepção do usuário.

2.1.2 Tipos de Virtualização

A virtualização é 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 [34] [16] [14]. Alguns dos benefícios dessa tecnologia são: alta disponibilidade, segurança, tolerância a falhas, compartilhamento de recursos e escalabilidade. A utilização da virtualização diminui os custos operacionais com aquisição de infraestrutura, consolidando seus servidores como máquinas virtuais em um único servidor físico [14]. 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, possibilitando o compartilhamento de recursos entre as máquinas virtuais. As principais técnicas de virtualização são descritas a seguir.

2.1.2.1 Virtualização Tradicional

A Virtualização tradicional é representada pela virtualização total e pela paravirtualização, tendo como principais representantes os seguintes virtualizadores: vSphere

[34], Xen [39], KVM [18]. Esses virtualizadores são constituídos por um monitor da máquina virtual (VMM), hospedado sob uma máquina física que fornece abstração total aos recursos do *hardware*, conforme pode ser visto na figura 2.1.

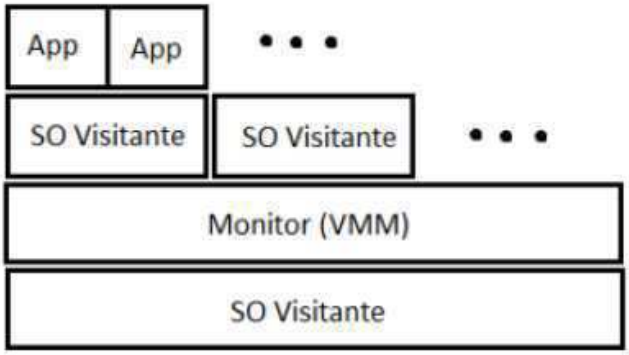

Figura 2.1 – Tecnologia tradicional de virtualização (adaptada de Xavier *et al.* [38]).

2.1.2.2 Virtualização baseada em contêineres

A virtualização baseada em contêineres é uma abordagem também conhecida como virtualização a nível de sistema operacional. Nesta abordagem, a camada de virtualização executa como uma aplicação dentro do sistema operacional. As máquinas virtuais são executadas como *guests* do *kernel* do sistema operacional, sendo chamados de contêineres. De acordo com Xavier *et al.* [36] [38], a virtualização baseada em contêineres é uma versão *lightweigtht* para *Hypervisors*. Esta abordagem de virtualização particiona os recursos das máquinas físicas, criando múltiplas instâncias de *userspaces* no mesmo sistema operacional. Usuários tem a ilusão que estão trabalhando em um ambiente proprietário. A figura 2.2 apresenta esta abordagem, onde a virtualização baseada em contêineres trabalha a nível de sistema operacional, providenciando abstrações diretamente para as aplicações que estão em execução neste ambiente. Por esta razão, todos os contêineres, compartilham um mesmo *kernel* de sistema operacional e supostamente tem um isolamento mais fraco comparado aos sistemas tradicionais de *Hypervisors*. Entretanto, do ponto de vista dos usuários, cada contêiner executa exatamente como um sistema operacional individual.

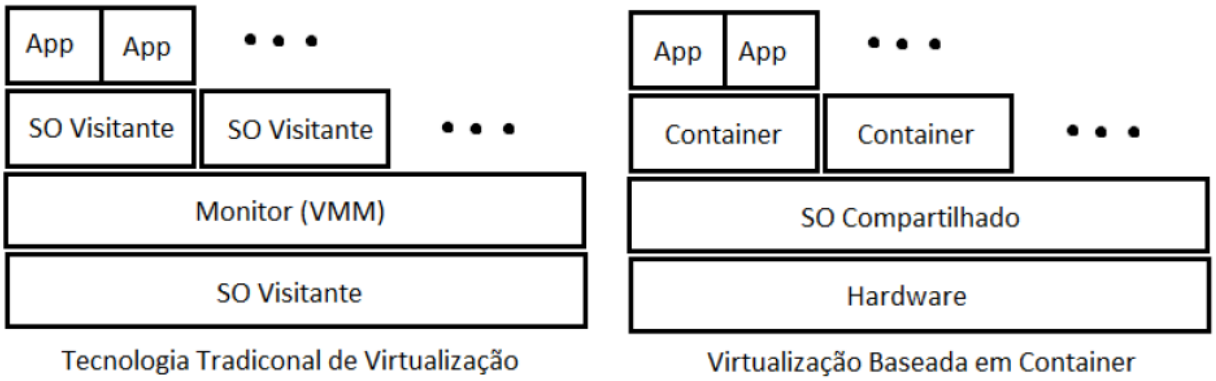

Figura 2.2 – Comparação entre os modelos de virtualização (adaptada de Xavier *et al.* [38]).

Atualmente, o LXC é uma tecnologia baseada em contêiner que tem ganhado espaço devido a sua inclusão no *kernel* Linux. O LXC utiliza principalmente dois recursos de *kernel* para conter processos [36]: *namespaces* e *cgroups*. Os subsistemas *namespaces* fornecem ambientes isolados de espaços para o usuário na forma de contêineres. Além disso, o subsistema *cgroups* fornece o gerenciamento de recursos, em outras palavras, o objetivo dos subsistemas LXC é criar um ambiente tão próximo quanto possível para uma instalação padrão Linux, mas sem a necessidade de um *kernel* separado. Portanto, optou-se pela utilização do LXC como plataforma de virtualização deste trabalho.

O *cgroups* é o recurso que garante o isolamento de recursos, evitando que diferentes máquinas virtuais, utilizem mais recursos do que os alocados e definidos para cada ambiente, restringindo o que está sendo utilizado e o que ainda está disponível para ser alocado entre os ambientes em execução. A próxima seção vai contextualizar o termo isolamento, bem como, apresentar os outros tipos de isolamento existentes.

2.1.3 Isolamento de Recursos

Com a popularização da computação em nuvem nas organizações, pode-se identificar algumas preocupações com a sua adoção em massa, tais como: (i) número de usuários em busca de soluções que oferecem escalabilidade e elasticidade na alocação de recursos as suas aplicações no modelo *pay-per-use* e; (ii) preocupação com o melhor aproveitamento dos recursos de infraestrutura da nuvem. Um fator bastante decisivo sobre o modelo de gerenciamento de recursos deve-se ao nível de isolamento que o ambiente de TI foi estruturado, de acordo com Rouven *et al.* [17] [31] o isolamento é entendido como um serviço provisionado que não deve interferir nos demais serviços do ambiente, por exemplo, a aplicação não interfere no banco de dados e vice-versa.

Outra forma de isolamento seria o de infraestrutura, como por exemplo, um cenário com múltiplas máquinas virtuais as quais concorrem com os mesmos recursos compartilhados, a aplicação adequada do isolamento auxiliaria a evitar a interferência de máquina virtual na execução de outras máquinas no mesmo ambiente compartilhado. Isso é bastante

comum, quando o usuário tem cotas de utilização e o mesmo excede, sendo degradada a performance destes recursos para equilibrar sua utilização pelos demais usuários, respeitando suas cotas e reduzindo contenção. Desta forma, pode-se classificar o isolamento em três tipos: isolamento de falhas, isolamento de recursos e isolamento de segurança [30]. Neste caso, o isolamento de recursos é o mecanismo pelo qual será garantido que as restrições de recursos de disco definidos pela política proposta por este trabalho sejam seguidos, sem o uso fora dos limites máximos de leitura e escrita durante os testes para validação deste política. Desta forma, o isolamento de recursos é descrito em mais detalhes abaixo, além de uma breve contextualização sobre os demais tipos de alocação: segurança e falhas.

2.1.3.1 Visão Geral sobre Tipos de Isolamento

Esta subseção apresenta uma breve definição sobre cada tipo de isolamento de recursos, os serão apresentados a seguir.

2.1.3.1.1 Isolamento de Segurança

Trata-se de minimizar o acesso de uma máquina virtual (VM) a objetos lógicos como arquivos, endereços virtuais de memória, Ids de usuários, Ids de processos, etc., entre outras máquinas. Assim, reduzindo falhas de segurança nos dados das aplicações em execução. O isolamento de segurança promove as seguintes características: (i) Independência de configuração, assim *Global Names* (exemplo: arquivos) selecionados por uma VM não podem conflitar com os nomes selecionados por outra VM e; (ii) Segurança, quando *global namespaces* são compartilhados, uma VM não estará habilitada a modificar dados e código que pertençam a outra VM [30].

2.1.3.1.2 Isolamento de Falhas

É a habilidade de limitar uma VM que esteja com algum tipo de falha que possa impactar outras VMs no mesmo servidor físico ou grupo de máquinas virtuais. Um isolamento de falhas completo entre vários ambientes requer que não haja compartilhamento de código ou dados entre eles. Em sistemas baseados em *Hypervisor*, as VMs são completamente isoladas das outras usando espaços de endereçamento únicos para evitar compartilhamento de dados [30].

2.1.3.1.3 Isolamento de Recursos

Em um cenário aonde se tem várias aplicações críticas em execução, em um mesmo servidor físico, pode haver riscos relacionados a carga de trabalho de alguma aplicação com uma alta atividade que impactará na performance geral do servidor físico ou

mesmo consumindo mais recursos do que foi previamente provisionado a ela. O isolamento neste caso deve-se a restrição de recursos físicos como ciclos de CPU, memória, *link* de rede, espaço em disco, etc.), mas também com recursos lógicos como portas de rede, *buffers* de memória, etc. [30]. Desta forma, uma máquina virtual utilizará apenas os recursos de infraestrutura provisionados, não havendo interferência com outros ambientes e os recursos designados a eles. Com exceção se o recurso está explicitamente compartilhado, como por exemplo, um disco compartilhado entre dois ou mais ambientes. Neste caso, haverá compartilhamento de dados e uso de recursos e consequentemente, concorrência entre estes ambientes.

A próxima seção descreve o isolamento de recursos em um cenário de ambientes virtualizados, relacionado ao problema de contenção de recursos devido a concorrência por múltiplas máquinas virtuais com recursos compartilhados, como o disco. Sendo o foco deste trabalho, assim como o objetivo de acelerar a execução destas aplicações de banco de dados neste tipo de cenário.

2.1.3.2 Contenção de Recursos em Ambientes Consolidados

Com a crescente utilização da virtualização pelas organizações em busca de redução de custos operacionais com servidores físicos no *data center* e a consolidação de aplicações, surge a preocupação quanto a concorrência de recursos em ambientes virtualizados consolidados. Desta forma, aplicações críticas são mais sucessíveis a gargalos de performance, tais como, transações em um banco de dados. Este fato é evidenciado através de um estudo comparativo realizado pela universidade *Technische Universität Darmstadt* na Alemanha [15], o qual demonstra que a performance em banco de dados tem degradação em ambientes virtualizados. Para avaliar este problema, foi configurado um laboratório de teste com dois servidores, um físico e um virtualizado com as mesmas configurações de hardware, mesma versão de sistema operacional (Ubuntu) e de banco de dados MySQL Server.

Os critérios de avaliação utilizados foram os seguintes: a) Utilização de uma mesma configuração de *hardware* para garantir que os resultados sejam comparáveis; b) A virtualização permite múltiplas VMs em execução de forma paralela no mesmo servidor, o que permite avaliar o efeito de operações com paralelismo de instâncias de banco de dados em ambos os ambientes – físico e virtual. A partir dos testes realizados, é possível identificar que o compartilhamento de recursos através de um ambiente consolidado composto por várias máquinas virtuais e com diferentes *workloads*, pode-se gerar contenção de recursos e consequentemente, impacto na performance. Desta forma, uma melhor alocação de recursos, torna-se imprescindível para uma performance adequada em aplicações críticas com requisitos de performance, como banco de dados. Surgindo uma oportunidade para o trabalho proposto com a definição de uma política dinâmica de alocação de recursos. Na

próxima seção é contextualizado os tipos de políticas de alocação de recursos e alguns conceitos relacionados a banco de dados, antes de ser apresentado mais detalhes sobre a política proposta nos próximos capítulos.

2.1.4 Políticas para Alocação de Recursos

A alocação de recursos em ambientes virtualizados 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 [5]. 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 [10] apresentam duas abordagens para a alocação de recursos: manual (estática) ou automática (dinâmica).

2.1.4.1 Alocação Estática

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 implicações da alocação estática de recurso pode-se citar:

- 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.

2.1.4.2 Alocação Dinâmica

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 a alocação apropriada de recursos, ou seja, devem ser realizadas de forma dinâmica e de acordo com

as necessidades de cada 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 [23];
- Reativas: a abordagem reativa é marcada pelo emprego do mecanismo regra-condição-ação [23].

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] [28].

2.1.4.3 Alocação utilizando Grupos de Controle

Grupos de Controle (*cgroups*) disponibilizam 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* [29].

- 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 política de alocação de recursos a ser proposta neste trabalho (mais detalhes no capítulo 5) utiliza a alocação de recursos dinâmica, desta forma, o uso do recurso *cgroups* é essencial para que o controle da alocação dos recursos de disco seja executada de forma apropriada e possa ser tratada a contenção de disco devido a requisições de E/S de múltiplas aplicações de banco de dados em execução compartilhando a mesma área de disco. A partir disso, os seguintes conceitos necessitam de uma contextualização adicional: tipos de *workload* de banco de dados, contenção de disco e métricas de performance (como será medido a necessidade de uma maior ou menor alocação de recursos). As próximas seções vão abordar estes temas em detalhes.

2.2 Aplicações de Banco de Dados

Esta seção apresenta uma breve contextualização referente banco de dados, contenção de disco e sobre métricas de performance.

2.2.1 Definição de Banco de Dados

Banco de dados é um recurso que pode ser interpretado como um sistema computadorizado de armazenamento de registros, porém, não se resume apenas ao armazenamento dos registros, pois oferece também a possibilidade de manipulação e gerenciamento destes registros. A manipulação dos registros poderá acontecer sob a forma de alterações, inserções e exclusões de dados [6]. Segundo Navathe [9], um banco de dados, possui as seguintes características:

- Representa fatos do mundo real;
- Dados organizados: O banco de dados é uma coleção lógica e coerente de dados com algum significado inerente;
- Dados coerentes: O banco de dados é projetado, construído e populado por dados, atendendo a um assunto em comum. Possui um grupo de usuários definidos que acessam esses dados por meio de algumas aplicações previamente geradas, administrando os dados de acordo com o interesse desse grupo de usuários;
- Tamanho flexível: O banco de dados pode ser de qualquer tamanho e de complexidade ao longo de sua existência. Dessa forma, um conjunto de dados pode conter de centenas até milhões de registros de diversos formatos e complexidades.

Os banco de dados tem como uma de suas características a independência entre estruturas lógicas de dados como tabelas, *views* e índices das estruturas de armazenamento físicas (estrutura de arquivos). Como as estruturas lógicas e físicas são separadas, a camada física de dados pode ser armazenada sem influenciar o acesso às estruturas lógicas. Uma analogia é renomear uma estrutura lógica, por exemplo uma tabela, e não renomear o arquivo físico que armazena essa tabela. Normalmente, estes arquivos físicos são criados no momento em que uma estrutura de dados lógica é definida. Um arquivo de dados existe fisicamente no disco e é criado pelo SGBD (Sistema de Gerenciamento de Banco de Dados). Nele são armazenadas as estruturas de dados, tais como as tabelas, índices e os registros. Estes arquivos que são em um formato proprietário, normalmente não são acessados por outros programas. Os arquivos de dados podem estar localizados em um sistema de arquivos local do sistema operacional ou em um grupo de discos, como por exemplo, em um *storage* [9]. A próxima seção apresenta uma breve descrição sobre os tipos de *workload* de banco de dados abordados durante todo este trabalho.

2.2.1.1 Tipos de *Workloads* de Banco de Dados: DW e OLTP

Este trabalho utiliza dois tipos *workloads*: DW e OLTP. Ambos tem características de acesso ao disco diferentes, pois o DW possui apenas leitura e o OLTP leitura e escrita. A figura 2.3 apresenta um exemplo de comportamento entre os dois tipos de *workloads*:

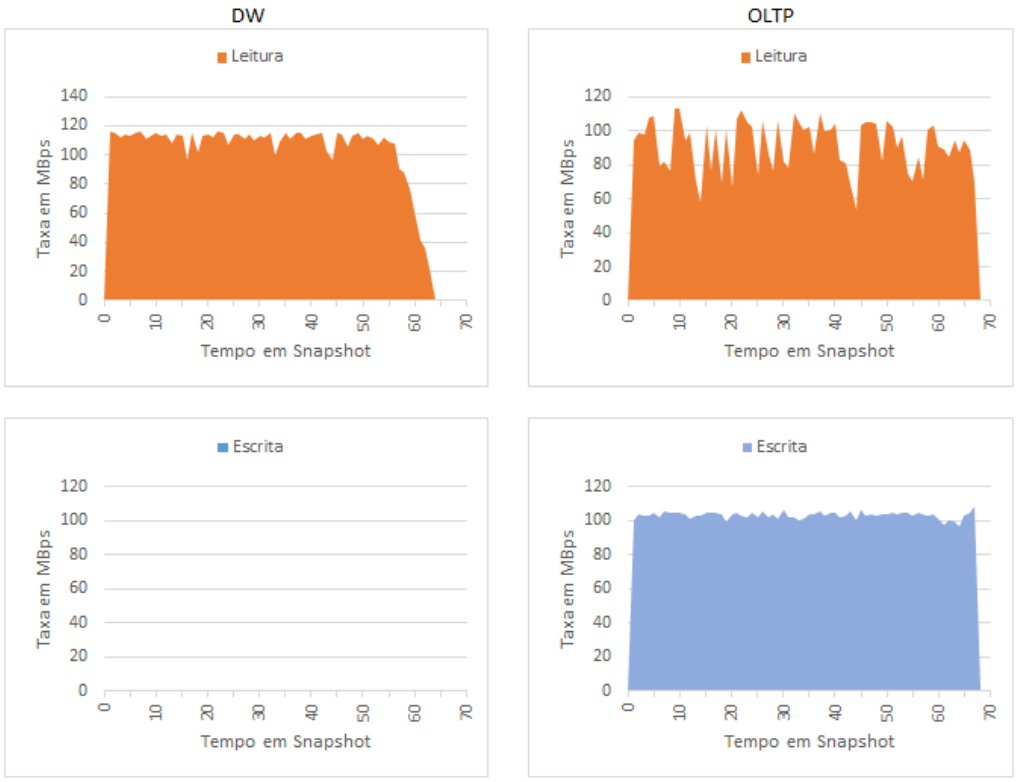

Figura 2.3 – Gráfico comparativo entre *workloads* DW x OLTP

2.2.1.1.1 Tipo de Workload DW

De acordo com Anzanello *et al.* [2] um DW é um repositório de informações de fontes autônomas, distribuídas e possivelmente heterogêneas de dados. Esses dados são transformados em informações úteis, oferecendo um enfoque histórico para permitir um suporte efetivo à decisão. O grande benefício que esta transformação gera é a possibilidade de prover múltiplas visões das informações para um conjunto de usuários diferentes.

De acordo com Srivastava *et al.* [32], um DW possui geralmente quatro características:

- Organização por assunto: Os dados são organizados por assunto, ao invés de serem organizados por aplicação;
- Integração entre dados: Quando os dados residem em muitas aplicações distintas em um ambiente operacional, a codificação dos dados é frequentemente inconsistente. Por exemplo, se uma aplicação representa o dado Sexo por "m"ou "f", outra aplicação pode representar por "M"ou "F";
- Presença de dados históricos: O DW deve manter um espaço para armazenamento de dados históricos de cinco a dez anos ou uma retenção maior, uma vez que estes dados serão usados para comparações, tendências, previsões, etc. Além disso, não há alteração nestes dados;
- Ausência de volatilidade: Os dados em um DW não são voláteis, ou seja, uma vez que sejam armazenados, os dados não são alterados ou atualizados de forma alguma, podendo somente ser acessados e carregados novos dados.

2.2.1.1.2 Tipo de Workload OLTP

De acordo com Dias *et al.* [8], a evolução tecnológica propiciou às empresas armazenar uma grande massa de dados oriundos dos dados transacionais. Tais sistemas utilizam tipicamente tecnologia OLTP e baseia-se em consultas a dados armazenados de forma tabular. As aplicações que a utilizam necessitam de suporte a decisões simples. Não são adequados a estruturas complexas, tais como metaconhecimento, transações de longa duração ou dados comportamentais. É principalmente utilizada em sistemas de banco de dados relacionais devido às suas características.

As maiorias das transações OLTP possuem informações que devem ser manipuladas em tempo real. Nelas, o repositório de dados é volátil, isto é, não há interesse em dados históricos não consolidados. Além disso, possui como padrão de acesso aos dados

pelas transações em execução bastante atividade de escrita e muitas vezes, pouca atividade de leitura, ao contrário de um banco de dados de DW que prevalece operações de leitura. Após a contextualização dos conceitos básicos relacionados a banco de dados, a próxima seção apresenta as aplicações de banco de dados em ambientes virtualizados e sobre dois tópicos adicionais: contenção de disco e métricas de performance.

2.2.2 Banco de dados em ambientes virtualizados

A virtualização pode ser uma ótima estratégia para auxiliar a atividade de administração dos recursos existentes, consolidação de ambientes e redução de custos com manutenção de servidores, assim como, flexibilidade na manutenção de ambientes virtualizados comparados aos físicos [1]. Neste caso, existem vários desafios na implementação de banco de dados em máquinas virtuais, tais como, desempenho das máquinas virtuais em comparação a utilização de servidores físicos e a dificuldade de consolidação dos servidores virtuais sobre uma máquina física [1] devido a contenção de recursos. No cenário de contenção de recursos, um fator que pode impactar bastante o desempenho o banco de dados e que é o foco deste trabalho é a contenção de disco.

2.2.2.1 Contenção de Disco

É muito comum efetuar a consolidação de múltiplas aplicações em HPC Clusters (*High-Performance Computing Clusters*), estas aplicações quando executadas de forma simultânea, compartilham o mesmo ambiente e possibilitando competição pela utilização de recursos, por exemplo: utilização de CPU, *caches* compartilhados, barramento de memória, controladores de memória e operações em rede e de escrita e leitura (E/S) em disco, causando a contenção de recursos e afetando diretamente as aplicações consolidadas [4]. Quando a demanda pela utilização de recursos é muito alta e não é gerenciada, as aplicações acabam interferindo entre si. O problema de interferência impacta diretamente no desempenho das aplicações.

A Figura 2.4 ilustra a execução de três aplicações, em três cenários: o primeiro cenário (a) ilustra a execução de aplicações em modo exclusivo, sem o compartilhamento de recursos. O tempo de execução para cada aplicação não sofre alteração. Isso ocorre porque para cada aplicação os recursos do *cluster* são utilizados em sua totalidade. No entanto, para executar uma nova aplicação é preciso aguardar o término da anterior. Já o segundo cenário (b) ilustra a execução de aplicações em um ambiente virtualizado, com compartilhamento de recursos, apresentando baixa contenção. Como pode ser visto, as aplicações são executadas de forma paralela, aproveitando melhor os recursos do *cluster*. De fato, esse é o cenário ideal. Entretanto, problemas como contenção de recursos ocorrem, causando impacto entre as aplicações e consequentemente aumentam o tempo de

execução. Conforme observado na segunda aplicação do cenário (b). Por fim, o terceiro cenário (c) ilustra a execução de aplicações em um ambiente virtualizado, com compartilhamento de recursos e contenção. Nesse caso, as aplicações compartilham e competem por recursos durante toda a execução. Sem dúvida, é o pior caso. Pois a contenção compromete totalmente o desempenho das aplicações.

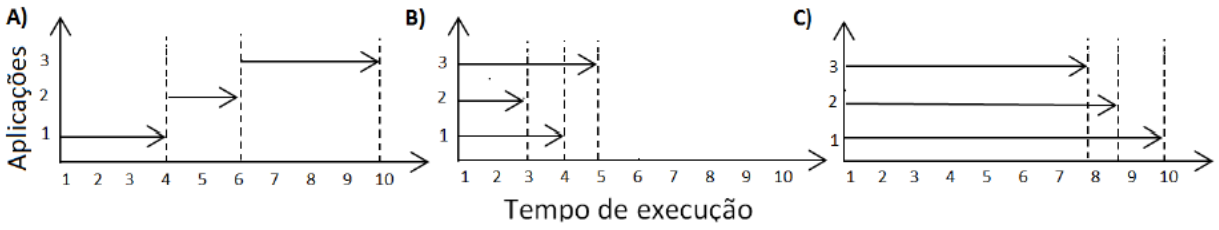

Figura 2.4 – Cenários de contenção de disco

Após a apresentação dos principais conceitos envolvidos neste trabalho relacionados a virtualização, isolamento, política para alocação de recursos e banco de dados, a próxima seção descreve sobre métricas de performance, relacionando como é medido a atividade do banco de dados através do TPS que é utilizado pela política dinâmica de alocação de recursos proposta neste trabalho.

2.2.2.2 Métricas de Performance

Segundo Hagmann *et al.* [13], existem diversas métricas de performance, tais como utilização de disco, cache (utilização da memória interna alocada ao banco de dados, ao invés de operações diretas no disco), paginação do sistema operacional. Neste trabalho, a métrica utilizada é o número de transações por segundo (TPS).

A velocidade de uma aplicação de banco de dados é medida por *Transaction Throughput* (a performance do banco de dados) que é expressada como um número de transações por segundo (TPS). [27]

Em um banco de dados transacional, geralmente pode-se encontrar os seguintes componentes: arquivos do banco de dados e os arquivos de log. Ambos são fatores que requirem E/S de disco, o qual é um recurso de infraestrutura, geralmente mais lento que outros recursos como CPU e memória.

Em um cenário com uma baixa performance, pode-se definir [27]:

- Acesso aos dados do banco de dados, se apresentar bastante randômico, por exemplo, em um *workload* do tipo OLTP, ao qual, pode apresentar transações alternadas de leitura e escrita, aliado a uma grande quantidade de dados, não possibilitando colocar todos os dados necessários em cache, resultando em uma requisição de E/S de disco por cada dado solicitado;
- Ambos os arquivos de banco de dados e os logs estarem em um mesmo local do disco, também gerando concorrência de recursos de E/S;

- Uma área compartilhada para diversos ambientes de banco de dados, também gerando contingência de recursos.

Isto significa que para cada transação de banco de dados é desempenhada várias operações no sistema de arquivos, tais como: (i) Operações de leitura e escrita em arquivos do banco de dados; (ii) Operações de leitura e escrita em logs do banco de dados; (iii) Escrita de dados da área de *cache* do banco para o disco para fins de consistência de dados, esta operação pode ocorrer tanto nos arquivos do banco de dados, como nos logs do mesmo, e por fim; (iv) Operações de leitura e escrita de metadados nos arquivos de log, que são os dados internos do banco de dados não relacionados a aplicação.

Existem várias maneiras de se aumentar o *Transaction Throughput* do banco de dados, dentre elas: (i) Redução da frequência de *commits* (ação relacionada a tornar permanente uma alteração de dados no banco de dados) de modo sequencial para grupo de operações em uma transação de banco de dados; (ii) Ajuste nos parâmetros de cache do banco de dados para que possa ser armazenado, uma maior quantidade de dados em memória, ao invés de se utilizar E/S de disco; (iii) Distribuir os arquivos e logs do banco de dados em discos distintos, a fim de reduzir a contenção de E/S de disco e; (iv) Para ambientes com uma infraestrutura mais antiga, deve-se revisar e atualizar o *hardware* do servidor, a fim de proporcionar uma melhor performance ao banco de dados.

Desta forma, pode-se avaliar a performance do banco de dados, com base neste índice de TPS, a fim de avaliar não o tempo que uma transação pode levar, mas sim, a quantidade de operações realizadas a cada segundo. Este índice é calculado da seguinte forma: delta da quantidade de *commits* + delta da quantidade de *rollbacks* (ação relacionada para desfazer uma operação no banco dados) [26].

Baseado nisso, o presente trabalho tem como foco, ambientes de banco de dados com recursos compartilhados, como disco, tendo como impacto direto, a contenção de recursos. Assim, ao longo deste trabalho, será abordado uma política para alocação de recursos de disco neste cenário que se baseia no índice de TPS para que sejam avaliados diferentes ambientes de banco de dados para priorização de recursos de disco.

3. TRABALHOS RELACIONADOS

Este capítulo tem por objetivo apresentar os trabalhos encontrados na literatura e do grupo de pesquisa da PUCRS que estão relacionados com a proposta desta dissertação, listando um comparativo entre eles. Reforçando o diferencial que gerou a motivação para este trabalho.

As seções deste capítulo estão organizadas da seguinte forma: trabalhos encontrados na literatura e trabalhos do grupo de pesquisa.

3.1 Trabalhos Encontrados na Literatura

Para maior embasamento foi realizada uma avaliação das soluções encontradas na literatura de alguns autores, e a seguir é apresentada uma breve explicação sobre cada trabalho analisado:

Barzan *et al.* [22] desenvolveu uma política de alocação de recursos que realiza uma predição dos recursos utilizados por transações de banco de dados com concorrência de recursos, entretanto, o mesmo era baseado no modelo tradicional de *Hypervisor* e apenas trabalhava com o *workload* de banco de dados OLTP. Além disso, a solução apenas tratava de operações de escrita, desconsiderando leitura, também foi utilizado apenas simulações e nenhum *workload* real para avaliação dos benefícios desta política, apresentando oportunidades de melhoria;

Zhu *et al.* [40] elaborou um modelo matemático com o objetivo de garantir que uma máquina virtual particular recebesse os recursos suficientes para manter o serviço dentro dos níveis desejados de eficiência. Este modelo foi usado por Dawoud *et al.* [7] na construção de uma arquitetura elástica. Entretanto, a arquitetura foi desenvolvida baseada em um sistema de virtualização tradicional, permitindo que recursos possam ser alocados somente na inicialização das máquinas virtuais. Em outras palavras, esta limitação não permite um serviço para alocação de recursos adicionais. A proposta acadêmica de Dawoud *et al.* [7] foi considerada promissora, apesar das dificuldades apresentadas quanto a alocação dinâmica de recursos. Devido a esta razão, este trabalho foi categorizado como alocação de recursos de forma estática na tabela 3.1;

Giri *et al.* [11] elaborou uma apresentação sobre um recurso no banco de dados Oracle chamado *Resource Manager*. Ele fornece a habilidade de otimizar a alocação de recursos entre conexões concorrentes de banco de dados. Esta solução apresenta as seguintes funcionalidades: (i) Priorização de recursos por parte das sessões do banco de dados através do *Resource Manager*; (ii) Definição de grupos de recursos para associar tipos específicos de usuários, por exemplo, processos ETL (*Extract, Transform and Load*) relacionados a uma aplicação de DW, pode ser associada a um grupo com uma alta porcentagem

de recursos da máquina; (iii) Limitação do nível de paralelismo entre os processos executados no banco de dados e; (iv) Limitação do uso de E/S, CPU e memória. O *Resource Manager* possui algumas similaridades com o *cgroups* (já abordado na seção 2.1.4.3), a configuração do ambiente para uso do *Resource Manager* é bastante simples, entretanto, é necessário dimensionar os grupos de alocação de recurso apropriadamente para evitar impactos na performance devido a um planejamento falho. Neste caso, esta solução é restrita ao banco de dados Oracle, e exige uma parada no ambiente para configuração do recurso, logo, não é possível realizar alterações na política de forma *online*. Além disso, esta solução é parte do *software* proprietário do banco de dados Oracle, não permitindo a sua utilização em outras tecnologias de banco de dados;

Matteussi *et al.* [19] utilizou uma política de alocação de recursos de forma estática. Neste caso, a mesma foi baseada em testes com ajustes manuais na alocação dos recursos de disco em máquinas virtuais, a fim de obter uma performance aprimorada para a execução de *jobs* de uma aplicação *Hadoop*. O presente trabalho tem como o objetivo de estender esta linha de pesquisa, propondo uma política de alocação de recursos de disco de forma dinâmica, mas com foco em aplicações de banco de dados.

As soluções analisadas possuem algumas características diferentes entre elas, inclusive algumas delas não atendem requisitos que a política deste trabalho propõe-se a melhorar, sendo assim, foi realizada uma avaliação destas soluções em relação aos seguintes pontos:

- Autor: Apresenta o nome do autor encontrado na literatura. É realizado um comparativo entre os trabalhos listados para destacar as diferenças entre a proposta deste trabalho e os demais;
- Plataforma: Define qual plataforma de virtualização foi utilizada, tais como *Hypervisor*, Contêiner ou N/A (Não se Aplica);
- Alocação: Define o tipo de alocação de recursos utilizado pelo trabalho do autor. A alocação de recursos pode ser Dinâmica ou Estática;
- Operação E/S: Define o tipo de operação de E/S suportada pelo trabalho do autor;
- Proprietário: Define se o trabalho é parte de algum *software* proprietário ou não. Neste caso, se o mesmo é proprietário, pode limitar a aplicabilidade da solução proposta, uma vez que pode requisitar a aquisição de um *software* específico para utilizar tal recurso;
- *Workload*: Define o tipo de *workload* suportado por cada trabalho encontrado na literatura. Neste caso, pode ser OLTP, DW ou *Hadoop*.

Na tabela 3.1 apresenta um resumo comparativo dos trabalhos analisados na literatura, que possuem similaridade com a proposta deste trabalho:

Tabela 3.1 – Lista de trabalhos encontrados na literatura

Autor	Plataforma	Alocação	Operação E/S	Proprietário	Workload
Barzan *et al*	*Hypervisor*	Dinâmico	Escrita	Não	OLTP
Dawoud *et al*	*Hypervisor*	Estático	Leitura/Escrita	Não	OLTP
Giri *et al*	N/A	Dinâmico	Leitura/Escrita	Sim	OLTP/DW
Kassiano *et al*	Contêiner	Estático	Escrita	Não	Hadoop

Sendo assim, os trabalhos analisados não apresentam soluções efetivas para minimizar a contenção em recursos de disco de banco de dados, uma vez que possuem restrições em sua solução, tais como: necessidade de *software* proprietário, alocação estática de recursos de disco, limitação de um tipo de *workload* de banco de dados e apenas o modo de escrita com simulações de dados, sem a utilização de cargas de dados com transações de dados reais, conforme exibido na tabela 3.1. Logo, este trabalho se diferencia dos demais, por tratar os problemas identificados, e propondo a alocação dinâmica de recursos de disco, de forma mais transparente para a este tipo de aplicação. Isso se dará através do uso de uma política que será apresentada no capítulo 5, onde utilizará o recurso de *cgroups* como base para a priorização de recursos de disco na plataforma de virtualização – LXC (Linux Contêiner), sendo independente do tipo de banco de dados (basta adaptar o driver de conexão), além disso permite o ajuste de forma dinâmica sem requisitar *downtime* na aplicação.

3.2 Trabalhos do Grupo de Pesquisa

Além dos trabalhos relacionados na literatura foram identificados alguns artigos criados pelo grupo de pesquisa do Prof. Dr. César Augusto Fonticielha De Rose que também serviram como motivação para este trabalho. Os trabalhos de pesquisa tiveram como foco o estudo dos efeitos de contenção de recursos em ambientes virtualizados, os quais estão relacionados ao assunto abordado no presente trabalho.

A tabela 3.2 apresenta os trabalhos que tem como enfoque o estudo dos efeitos de contenção de recursos de disco em ambientes virtualizados.

Além do trabalho realizado por Matteussi *et al.* [19] já visto na seção anterior, a tabela 3.2 descreve outros dois trabalhos nesta área.

O primeiro deles realizado por Xavier *et al.* [37] referente a um estudo comparativo sobre o isolamento de performance entre diferentes tipos de processamento: paralelo, distribuído e baseado em rede, apoiou este trabalho a avaliar os diferentes tipos de sistemas de virtualização baseados em contêineres, sendo utilizado o Linux Contêiner (LXC) nos experimentos realizados neste trabalho devido a sua flexibilidade e facilidade de configuração de ambientes e aplicação de restrições de recursos entre as máquinas virtuais em execução nesta plataforma de virtualização.

Tabela 3.2 – Lista de trabalhos relacionados do grupo de pesquisa

Título	**Foco do Trabalho**
Minimizando a contenção de disco em contêineres como estratégia para acelerar aplicações *Map Reduce* [19]	Isolamento de performance com foco em aplicações de *Map Reduce* utilizando sistema de arquivos compartilhados.
A Performance Comparison of Container-Based Virtualization Systems for Map Reduce Clusters. In: Proceedings of the parallel, Distributed and Network-Based Processing [37]	Estudo comparativo sobre isolamento de performance em ambientes virtualizados e em ambientes clusterizados com *Map Reduce*.
A Performance Isolation Analysis of Disk-Intensive Workloads on Container-BasedClouds [36]	Estudo comparativo sobre os diversos efeitos em aplicações com uso intensivo de disco em ambientes virtualizados.

Por fim, o outro trabalho também realizado por Xavier *et al.* [36] é referente a uma análise de isolamento de performance em banco de dados em ambientes virtualizados, serviu como motivação para a execução deste estudo devido a oportunidade de utilização de banco de dados em máquinas virtuais, assim como, avaliar oportunidade de otimizar sua performance e incentivar sua utilização neste meio.

4. HIPÓTESES E QUESTÕES DE PESQUISA

O Objetivo geral é continuar o trabalho realizado por Matteussi *et al.* [19] que teve como foco o ajuste estático de recursos de disco para acelerar aplicações *Hadoop*. A partir disso, esta proposta propõe uma política de alocação dinâmica de recursos de disco para aplicações de banco de dados tradicionais, a fim de acelerar sua execução, minimizando a contenção de disco em contêineres. Bem como avaliar os resultados obtidos com a política proposta.

Motivado pelo objetivo geral deste trabalho foram definidos os seguintes objetivos específicos:

i) estudar as arquiteturas e tecnologias de virtualização auxiliando o embasamento da fundamentação teórica da pesquisa;

ii) estudar as técnicas de virtualização existentes no estado da arte, assim como as atividades realizadas pelo trabalho de Matteussi *et al.* [19] com o intuito de auxiliar no alcance do objetivo geral;

iii) avaliar a proposta de ajuste estático do trabalho realizado por Matteussi *et al.* [19] no cenário de banco de dados referente a sua aplicabilidade neste tipo de ambiente;

iv) propor uma política de ajuste dinâmica de recursos de disco para banco de dados tradicionais;

v) avaliar a política proposta por meio de experimentos, observando o tempo de execução nos ambientes e o desempenho obtido neles.

Para conduzir essa investigação as seguintes questões de pesquisa foram definidas:

Q1: Qual o impacto do ajuste estático de recursos de disco proposto pelo trabalho de Matteussi *et al.* [19], levando em consideração o *workload* de banco de dados?

Q2: Qual o impacto de uma política de ajuste dinâmico de recursos de disco em relação a um cenário com múltiplos bancos de dados em execução em um ambiente virtualizado, levando em consideração o desempenho destes ambientes e o tempo de execução?

Formulou-se duas hipóteses neste trabalho, a hipótese H1 tem como objetivo contribuir na resolução da primeira questão de pesquisa e a H2 está associada com a segunda questão de pesquisa respectivamente:

H1: Os ganhos obtidos pelo estudo realizado por Matteussi *et al.* [19] em um *workload* do tipo *Hadoop*, aplicam-se também em um *workload* de banco de dados;

H2: A aplicação de uma única política dinâmica de priorização dos recursos de disco pode apresentar ganhos de performance para qualquer combinação dos *workloads* de banco de dados estudados.

A seguir serão apresentadas os experimentos que permitem responder a questão de pesquisa Q1, bem como o embasamento para a resposta da questão de pesquisa Q2. Além disso, será apresentado o detalhamento da aplicação de ajuste estático de recursos de disco.

4.1 Aplicação de Ajuste Estático de Recursos de Disco

O trabalho realizado por Matteussi *et al.* [19] teve como objetivo a realização de testes com o ajuste estático nos recursos de disco em aplicações *Hadoop* utilizando ambientes virtualizados, avaliando os benefícios na otimização da performance e redução da contenção de disco para o tipo de *workload Hadoop*. O mecanismo utilizado para garantir que os ajustes aplicados na alocação de recursos respeitem os limites definidos foi o isolamento de recursos. Foi através dele que o recurso *cgroups*, apresentado na seção 2.1.4.3, pode aplicar as restrições para as operações de leitura e escrita no disco e garantir que as mesmas não ultrapassem estes limites. Esta afirmação é válida tanto para o ajuste estático, quanto para o dinâmico que será apresentado na seção 6.3.

A questão de pesquisa Q1 tem como objetivo avaliar se o ajuste estático nos recursos de disco podem também otimizar aplicações de banco de dados. Para responder esta questão de pesquisa e a questão Q2 foram realizados testes com a mesma abordagem realizada pelo trabalho de Matteussi *et al.* [19], a fim de verificar se são válidas para aplicações de banco de dados.

Nas subseções a seguir são apresentados todos os pontos abordados nos testes realizados, como: configuração de ambiente, identificação da largura de banda de E/S, execução dos testes, bem como as considerações dos resultados obtidos.

4.1.1 Configuração do ambiente de testes

A identificação de problemas relacionados à performance no banco de dados muitas vezes é difícil, isso ocorre devido à complexidade do ambiente, entretanto, para este trabalho foi restringido ao problema de contenção de disco. Para avaliar esta contenção foi utilizada a monitoração do ambiente para a medição do uso dos recursos de disco durante a execução dos *workloads* utilizados neste trabalho. Esta medição dos recursos computacionais é chamada de *profiling*. Esta atividade tem como objetivo, a compreensão sobre o modo de execução de uma determinada aplicação.

O ambiente utilizado para os testes foram compostos pela seguinte configuração de *hardware*: servidor Dell Poweredge R610, CPU com 16 núcleos, memória de 16 GB de RAM, disco rígido de 150GB, sistema operacional do host Ubuntu Server 16.04, plataforma

de virtualização LXC, sistema operacional dos contêineres - *Oracle Enterprise Linux* 6.5 e banco de dados *Oracle Enterprise Database* 12c.

Foram criadas duas máquinas virtuais com o LXC, as quais foram nomeadas respectivamente: lxc-ora01 e lxc-ora02. Ambos configurados com o sistema operacional *Oracle Enterprise Linux* 6.5 (durante a criação do contêiner, foi utilizada a opção *template* = oracle do LXC) e instalado uma instância de banco de dados *Oracle* 12c em cada contêiner. Além disso, para definição das restrições de recursos foi utilizado o recurso *cgroups* (descrito em detalhes na seção 2.1.4.3) presente no LXC, onde utilizou-se os seguintes elementos: `blkio.throttle.read_bps_device` é responsável pela definição de um limite para operações de leitura em BPS (*Bits per Second*) em um disco específico; `blkio.throttle.write_bps_device` é responsável pela definição de um limite para operações de escrita em BPS em um disco específico; `cpuset.cpus` é responsável por definir a quantidade de CPU cores que cada contêiner poderá utilizar; `memory.limit_in_bytes` é responsável por definir a quantidade de memória em *bytes* que cada contêiner poderá utilizar.

Para evitar a concorrência de recursos em outras áreas do servidor que possam interferir nos testes realizados com o disco, como por exemplo, CPU e memória, estes recursos foram alocados 50% de recurso para cada ambiente utilizado nos testes. Desta forma, identificou-se o problema de contenção de recursos exclusivamente de disco que é o foco deste trabalho.

Além disso, o foco dos testes realizados foi exemplificar o pior caso de contenção de recursos de disco, onde todos os contêineres estão sendo executados ao mesmo tempo, com suas respectivas cargas de trabalho, com o objetivo de avaliar o comportamento dos cenários de teste com diferentes configurações de restrição de recursos.

4.1.2 Identificação da largura de banda de leitura e escrita

Primeiramente, foi necessário identificar qual era a taxa máxima de leitura e escrita que o disco do *host* poderia alcançar para a realização dos testes iniciais quanto a restrições de recursos de disco. Este passo é importante para identificar os limites de E/S para avaliar os valores máximos que podem ser configurados nas restrições de disco.

Para isso, foi utilizado o programa HDPARM do Linux para avaliar a taxa de leitura e o DD para avaliar a taxa de escrita respectivamente. Foram realizadas cinquenta execuções consecutivas de cada ferramenta para identificar uma média confiável, a qual foi utilizada como base para os próximos testes. Conforme apresentado na figura 4.1 a seguir, obteve-se uma taxa média de leitura de 118 Mbps e 108 Mbps para escrita. Estes dados de taxa de leitura e escrita foram utilizados em todos os cenários de teste deste trabalho, como métrica para o limite máximo que pode ser alcançado no ambiente de teste utilizado.

Figura 4.1 – Coleta de dados para identificar taxa de leitura/escrita no disco

4.1.3 Execução dos Testes

Para os testes de performance foi utilizada a ferramenta IOTOP, para capturar os índices de leitura/escrita dos *workloads*, bem como o tempo de execução da aplicação. A seguir são apresentados as simulações realizadas para cada cenário, com o objetivo de evidenciar os índices coletados com os diferentes tipos de *workloads*, bem como diversas situações de restrições aplicadas. As simulações realizadas nesta seção são baseadas na ferramenta de *benchmark* Orion [24] da fabricante Oracle, para cada tipo *workload*, DW e OLTP. Cada ciclo de simulação foi composta por dez ciclos de execução para cada cenário de teste, capturando uma amostra dos *snapshots* a cada cinco segundos, os quais foram considerados na avaliação dos resultados para cada *workload*.

Foram realizadas simulações de execução de forma isolada, a qual não foi realizado nenhum ajuste de restrição para alocação de disco, e execução com contêineres concorrentes, como forma evidenciar o comportamento em ambas as situações. Para as execuções concorrentes foram aplicadas as seguintes restrições: sem ajuste de restrição nos limites de E/S, com o objetivo de avaliar o comportamento do disco sem intervenção e com ajuste estático da restrição dos limites de E/S, onde foram aplicados os seguintes percentuais, 50% x 50% e 80% x 20%, com o objetivo de verificar o comportamento do disco com as restrições definidas nas taxas de leitura e escrita utilizando os limites máximos encontrados na seção 4.1.2. Os percentuais de ajuste estático de restrição de leitura e escrita foram escolhidos com o seguinte objetivo: (i) o percentual 50% x 50% para podermos comparar com o cenário sem ajuste e; (ii) o percentual 80% x 20% para mostrar um dos piores casos com restrição estática, proporcionando um comparativo no gráfico de forma visual que o menor percentual continua sua alocação por mais tempo, até o final de sua execução. Para cada cenário executado, coletou-se os dados de Taxa em Mbps e tempo, os quais

serão apresentados nos próximos itens, em formato de gráfico. Além disso, utilizou-se para o processamento uma carga de dados de 20GB para cada cenário de teste.

Os cenários de teste foram divididos em três grupos, para facilitar a análise dos resultados e possibilitar uma comparação dos dados, onde: o grupo A é composto por testes realizados com o *workload* do tipo DW; o grupo B é composto por testes realizados com o *workload* do tipo OLTP; e o grupo C é composto por testes realizados com *workloads* do tipo DW e OLTP. Ambos em execução paralela, forçando o pior cenário que é a concorrência por recursos de disco.

Os cenários A1 e B1 com *workloads* isolados foram configurados para utilizar apenas uma máquina virtual, ou seja, sem concorrência, e os recursos de CPU e memória do servidor foram configurados com 100%. Já os demais cenários foram configurados para utilizar duas máquinas virtuais e os recursos de CPU e memória do servidor foram configurados com 50% dos recursos em cada uma destas máquinas. Além disso, os limites de E/S do disco para leitura e escrita foram utilizados os valores médios encontrados na seção 4.1.2 como taxa máxima de leitura e escrita para todos os cenários. Cada máquina virtual possuía um banco de dados.

O ajuste estático de recursos de disco é realizado de forma manual e sem alteração das restrições aplicadas durante toda a execução. A tabela 4.1 a seguir, apresenta o resumo de todos os cenários de testes realizados, sem aplicação de ajuste de restrições de disco e com a aplicação de ajuste estático de restrições de disco.

Tabela 4.1 – Cenários de teste para o ajuste manual das restrições de disco

Cenário de Teste	**Restrição**	**Workload**	**Operação E/S**
A1	-	DW	Leitura
A2	-	DW x DW	Leitura
A3	50% x 50%	DW x DW	Leitura
A4	80% x 20%	DW x DW	Leitura
B1	-	OLTP	Leitura e Escrita
B2	-	OLTP x OLTP	Leitura e Escrita
B3	50% x 50%	OLTP x OLTP	Leitura e Escrita
B4	80% x 20%	OLTP x OLTP	Leitura e Escrita
C1	-	DW x OLTP	Leitura e Escrita
C2	50% x 50%	DW x OLTP	Leitura e Escrita
C3	80% x 20%	DW x OLTP	Leitura e Escrita

As subseções a seguir apresentam os detalhes dos testes realizados para os grupos A, B e C apresentados na tabela 4.1, com o objetivo de detalhar o comportamento dos mesmos.

4.1.3.1 Testes do grupo A - *Workload* DW

O *workload* do tipo DW tem como característica, a execução de um processo de leitura para demonstrar a execução de um relatório em cima de uma carga de dados. Com

o método de ajuste estático, foram aplicados os limites para as operações de E/S, apresentados na tabela 4.2:

Tabela 4.2 – Limites de E/S para os testes com DW

Cenário de Teste	Restrição	Taxa Limite de Leitura	Taxa Limite de Escrita
A1	-	118	108
A2	-	118	108
A3	50% x 50%	59	54
A4	80% x 20%	94.4	86.4

Baseado na tabela 4.2, pode-se observar que os valores para as taxas de limite de leitura e escrita foram definidas a partir da restrição utilizada, como por exemplo, no cenário A1 sem restrição, utiliza-se 100% dos recursos disponíveis de disco, já no cenário A3, é aplicado a restrição de recursos de disco de 50% para cada contêiner da taxa limite identificada na seção 4.1.2.

A figura 4.2 apresenta os resultados obtidos em cada cenário de teste, onde: (A1) Isolado sem ajuste estático de restrição de disco; (A2) Concorrente sem ajuste estático de restrição de disco; e (A3) e (A4) Concorrentes com ajuste estático de restrição de disco, conforme os percentuais definidos durante toda a execução.

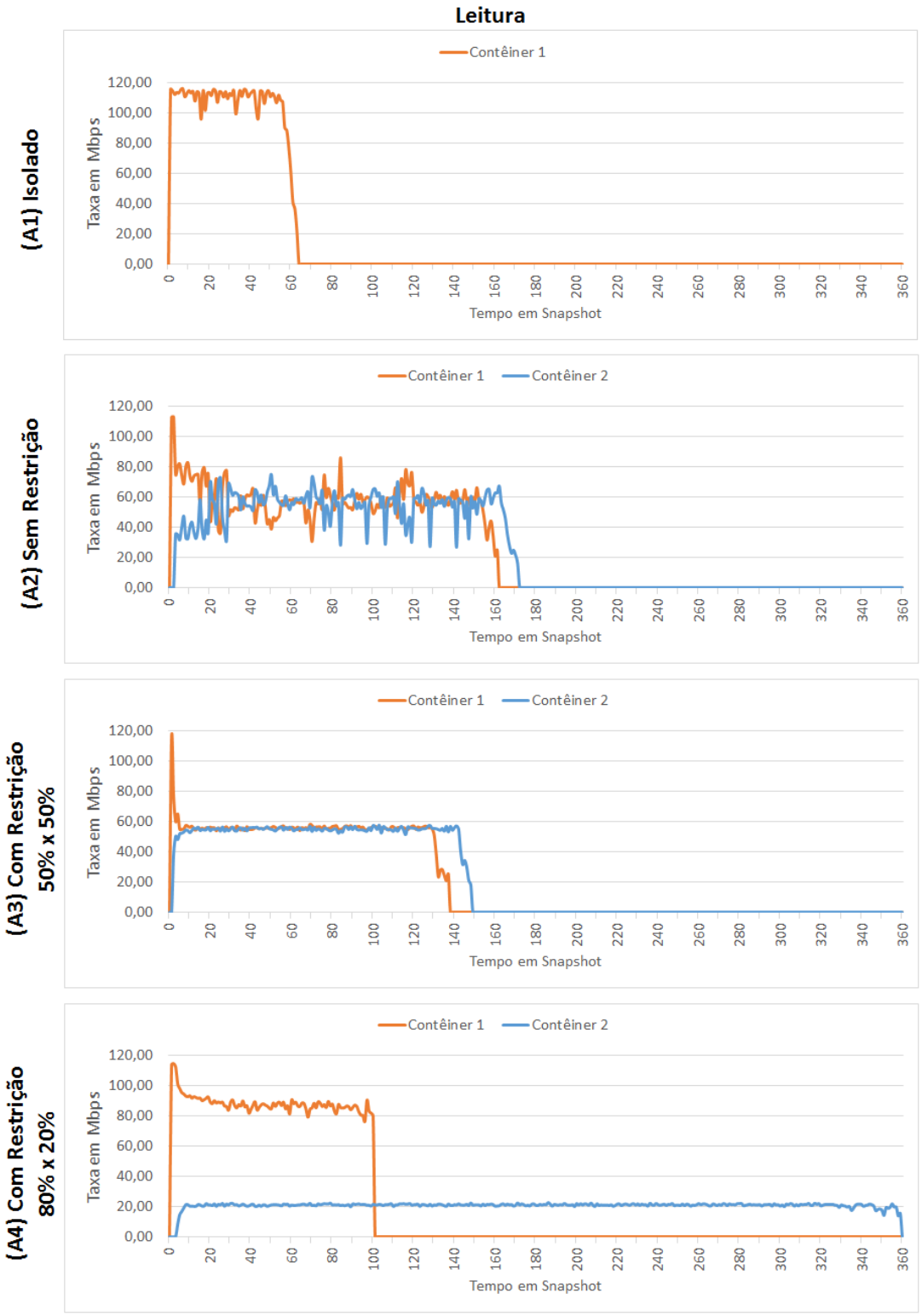

Figura 4.2 – Experimentos realizados com ajuste estático em *workloads* DW

A seguir serão detalhados os cenários de teste A1, A2, A3 e A4 do grupo A.

Cenário A1: Execução de *Workload* DW de Forma Isolada

O teste com o cenário isolado foi realizado para demonstrar o comportamento de um ambiente de banco de dados com um *workload* do tipo DW sem a concorrência com

outras máquinas virtuais, para exibir sua execução de forma dedicada. O gráfico A1 apresentado anteriormente na figura 4.2, representa a execução isolada deste *workload*. Como não ocorreu operações de escrita devido ao tipo de transação em execução (DW), o gráfico mostra apenas as operações de leitura. Este *workload* teve como tempo de execução de 315 segundos.

Cenário A2: Execução de Workload DW x DW Sem Restrição

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo DW sem restrição de recursos de disco, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico A2 apresentado anteriormente na figura 4.2, demonstra a execução de múltiplas máquinas virtuais com o *workload* de DW. Sendo o tempo de duração total com os dois contêineres de 855 segundos.

Cenário A3: Execução de Workload DW x DW Com Restrição 50% x 50%

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo DW com restrição de 50% de recursos de disco para cada contêiner, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico A3 apresentado anteriormente na figura 4.2, demonstra a execução de múltiplas máquinas virtuais com o *workload* de DW. Sendo o tempo de duração total com os dois contêineres de 740 segundos.

Cenário A4: Execução de Workload DW x DW Com Restrição 80% x 20%

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo DW com restrição de 80% de recursos de disco o contêiner 1 e 20% para o contêiner 2, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico A4 apresentado anteriormente na figura 4.2, demonstra a execução de múltiplas máquinas virtuais com o *workload* de DW. Sendo o tempo de duração total com os dois contêineres de 1795 segundos.

4.1.3.2 Testes do grupo B - *Workload* OLTP

Nestes testes foi utilizado o *workload* OLTP, cuja característica foi a execução de um processo de escrita, e que também apresentava operações de leitura, para demonstrar a execução de uma transação de banco de dados para carga e modificação de dados. A tabela 4.3 apresenta os limites para as operações de E/S, utilizados como ajuste manual.

Tabela 4.3 – Limites de E/S para os testes com OLTP

Cenário de Teste	Restrição	Taxa Limite de Leitura	Taxa Limite de Escrita
B1	-	118	108
B2	-	118	108
B3	50% x 50%	59	54
B4	80% x 20%	94.4	86.4

Baseado na tabela 4.3, pode-se observar que os valores para as taxas de limite de leitura e escrita foram definidas a partir da restrição utilizada, como por exemplo, no cenário B1 sem restrição, utiliza-se 100% dos recursos disponíveis de disco, já no cenário B3, é aplicado a restrição de recursos de disco de 50% para cada contêiner da taxa limite identificada na seção 4.1.2.

A figura 4.3 apresenta os resultados obtidos em cada cenário de teste, onde: (B1) Isolado sem ajuste estático de restrição de disco; (B2) Concorrente sem ajuste estático de restrição de disco; e (B3) e (B4) Concorrentes com ajuste estático de restrição de disco.

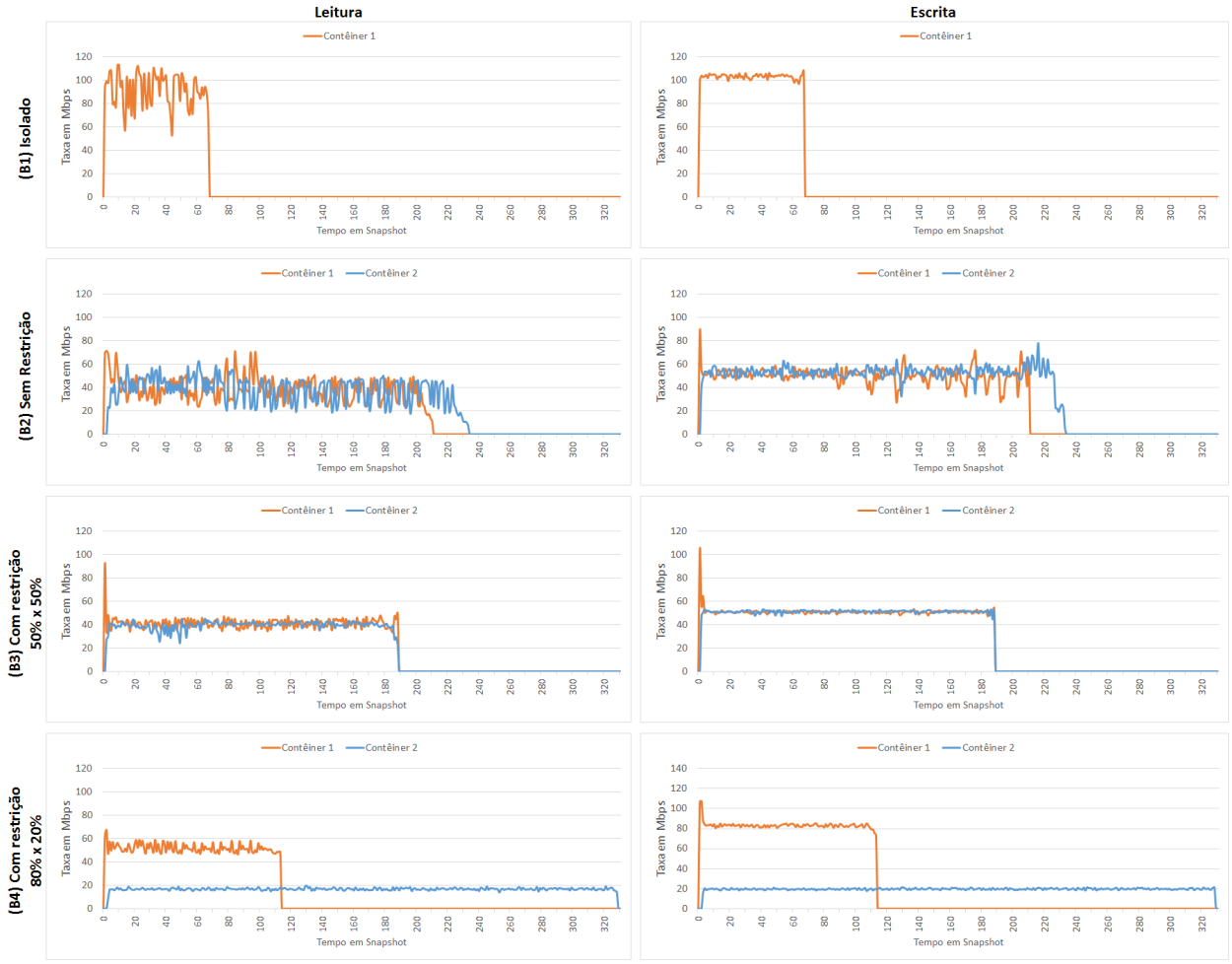

Figura 4.3 – Experimentos realizados com ajuste estático em *workloads* OLTP

A seguir serão detalhados os cenários de teste B1, B2, B3 e B4 do grupo B.

Cenário B1: Execução de *Workload* OLTP de Forma Isolada

O teste com o cenário isolado foi realizado para demonstrar o comportamento de um ambiente de banco de dados com um *workload* do tipo OLTP sem a concorrência com outras máquinas virtuais, para exibir sua execução de forma dedicada. Este *workload* possui operações de leitura e escrita, diferente do *workload* do tipo DW, onde haviam apenas operações de leitura. O gráfico B1 apresentado anteriormente na figura 4.3, demonstra a execução de uma máquina virtual com o *workload* do tipo OLTP de forma isolada e sem restrição. O tempo de execução foi de 335 segundos.

Cenário B2: Execução de *Workload* OLTP x OLTP Sem Restrição

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo OLTP sem restrição de recursos de disco, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico B2 apresentado anteriormente na figura 4.3, demonstra a execução de duas máquinas virtuais com o *workload* de OLTP. O tempo de execução foi de 1165 segundos.

Cenário B3: Execução de *Workload* OLTP x OLTP Com Restrição 50% x 50%

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo OLTP com restrição de 50% de recursos de disco para cada contêiner, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico B3 apresentado anteriormente na figura 4.3, demonstra a execução de duas máquinas virtuais com o *workload* de OLTP. O tempo de execução foi de 940 segundos.

Cenário B4: Execução de *Workload* OLTP x OLTP Com Restrição 80% x 20%

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo OLTP com restrição de 80% de recursos de disco o contêiner 1 e 20% para o contêiner 2, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico B4 apresentado anteriormente na figura 4.3, demonstra a execução de duas máquinas virtuais com o *workload* de OLTP. O tempo de execução foi de 1640 segundos.

4.1.3.3 Testes do Grupo C - *Workloads* DW x OLTP

Nos testes do grupo C foram utilizados dois tipos de *workloads*, DW e OLTP. O *workload* DW tem como característica a execução de um processo de leitura, já o *workload*

OLTP possui característica de leitura e escrita. Com o método de ajuste estático, foram aplicados os limites apresentados na tabela 4.4 para os testes:

Tabela 4.4 – Limites de E/S para os testes com DW x OLTP

Cenário de Teste	Restrição	Taxa Limite de Leitura	Taxa Limite de Escrita
C1	-	118	108
C2	50% x 50%	59	54
C3	80% x 20%	94.4	86.4

Baseado na tabela 4.4, pode-se observar que os valores para as taxas de limite de leitura e escrita foram definidas a partir da restrição utilizada, como por exemplo, no cenário C1 sem restrição, utiliza-se 100% dos recursos disponíveis de disco, já no cenário C3, é aplicado a restrição de recursos de disco de 50% para cada contêiner da taxa limite identificada na seção 4.1.2.

A figura 4.4 apresenta os resultados obtidos em cada cenário de teste, onde: (C1) Concorrente sem ajuste estático de restrição de disco; e (C2) e (C3) Concorrentes com ajuste estático de restrição de disco:

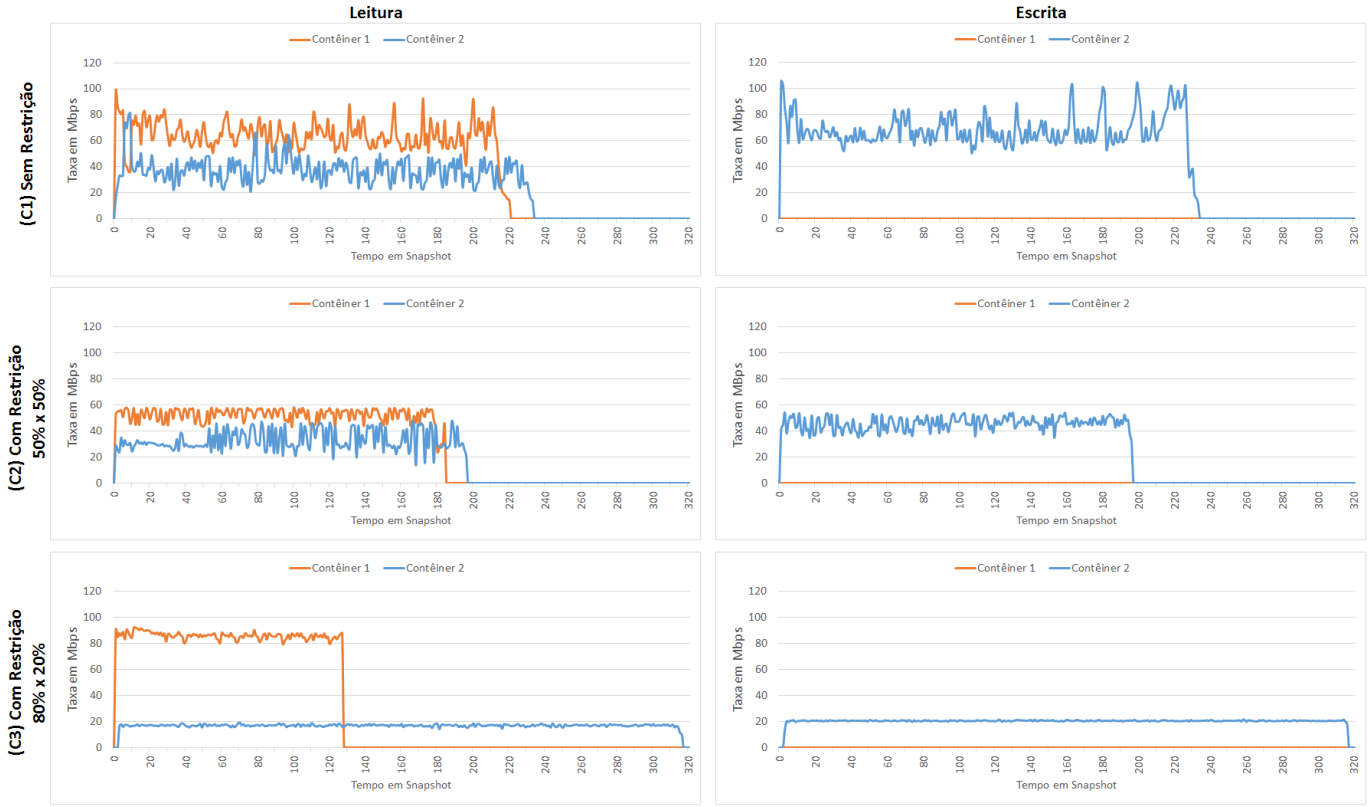

Figura 4.4 – Experimentos realizados com ajuste estático em *workloads* DW e OLTP

A seguir serão detalhados os cenários de teste C1, C2 e C3 do grupo C.

Cenário C1: Execução de *Workload* DW x OLTP Sem Restrição

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* de tipos diferentes, sem restrição de recursos de disco, onde o contêiner 1 é do tipo DW e o contêiner 2 é do tipo OLTP, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho, cuja característica é a presença de operações de leitura e escrita para o OLTP, e apenas para leitura para o DW. O gráfico C1 apresentado anteriormente na figura 4.4, demonstra a execução de duas máquinas virtuais com o *workload* de DW e OLTP. O tempo de execução foi de 1165 segundos.

Cenário C2: Execução de *Workload* DW x OLTP Com Restrição: 50% e 50%

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* de tipos diferentes, com restrição de 50% para cada contêiner, onde o contêiner 1 é do tipo DW e o contêiner 2 é do tipo OLTP, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico C2 apresentado anteriormente na figura 4.4, demonstra a execução de duas máquinas virtuais com o *workload* de DW e OLTP. O tempo de execução foi de 980 segundos.

Cenário C3: Execução de *Workload* DW x OLTP Com Restrição: 80% e 20%

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* de tipos diferentes, onde cada contêiner possui uma restrição diferente. O contêiner 1 é do tipo DW e foi aplicado uma restrição de recursos de disco de 80%, o contêiner 2 é do tipo OLTP e foi aplicado uma restrição de 20% de recursos de disco. Ambos sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico C3 apresentado anteriormente na figura 4.4, demonstra a execução de duas máquinas virtuais com o *workload* de DW e OLTP. O tempo de execução foi de 1580 segundos.

4.1.4 Considerações dos Resultados Obtidos

Com o objetivo de identificar o comportamento de execuções de forma isolada sem restrição, com dois contêineres em concorrência e sem restrição, e por fim, com dois contêineres em concorrência com restrição foram realizados os cenários de teste dos grupos A, B e C com seus respectivos ajustes estáticos para os recursos de disco conforme apresentado na seção anterior 4.1.3.

A tabela 4.5 apresenta para cada execução isolada foram realizados os testes em apenas uma máquina virtual, executando o contêiner sem aplicar restrição de recursos para

os cenários A1 e B1. Após isso, foram realizadas testes com execuções de duas máquinas virtuais sem aplicar restrição de recursos de disco, a fim de avaliar o comportamento sem nenhum ajuste de E/S para os cenários A2, B2 e C1. Finalmente, foram realizados testes com duas máquinas virtuais aplicando a restrição de forma estática, ou seja, sem alterações durante toda a execução dos *workloads* avaliados, seguindo as restrições de recursos definidas respectivamente 50% x 50% nos cenários A3, B3 e C2 e 80% e 20% nos cenários A4, B4 e C3. Estes percentuais estão relacionados a taxa máxima de operações de leitura e escrita que o disco pode alcançar, sendo aplicados respectivamente para 50% x 50% - 50% para o contêiner 1 e 50% para o contêiner 2. E para a restrição de 80% x 20% - 80% para o contêiner 1 e 20% para o contêiner 2. Em todos os cenários realizados, foram aplicadas 10 execuções em cada cenário, com o objetivo de identificar a duração, desvio padrão, por fim o *speedup* para mostrar os ganhos e perdas obtidos nos cenários com ajuste de restrição de disco.

Tabela 4.5 – Tempos em segundos de execução para os cenários de teste

Cenário	Restrição	Workload	Duração (s)	Desvio Padrão	Speedup
A1	-	DW	315	0,94	-
A2	-	DW x DW	855	1,80	1
A3	50% x 50%	DW x DW	740	1,43	1,15
A4	80% x 20%	DW x DW	1795	5,11	0,47
B1	-	OLTP	335	2,35	-
B2	-	OLTP x OLTP	1165	1,84	1
B3	50% x 50%	OLTP x OLTP	940	2,63	1,23
B4	80% x 20%	OLTP x OLTP	1640	1,61	0,71
C1	-	DW x OLTP	1165	2,99	1
C2	50% x 50%	DW x OLTP	980	2,55	1,18
C3	80% x 20%	DW x OLTP	1580	1,70	0,73

No grupo A, no cenário A1 foi utilizado um *workload* DW com apenas uma máquina virtual, sem restrição de recursos de disco e teve um tempo de duração de 315 segundos. Já o cenário A2 foram utilizadas duas máquinas virtuais com concorrência de recursos e assim como o cenário A1, também não foi aplicado nenhuma restrição nos recursos de disco. Este cenário A2 teve como tempo de duração de 855 segundos, gerando um tempo em torno de três vezes mais que a execução isolada apresentada no cenário A1. Já o cenário A3, similar ao cenário A2, também foram utilizadas duas máquinas virtuais, entretanto, foi aplicado uma restrição de recursos de 50% x 50%. Neste cenário teve um tempo de execução de 740 segundos e um ganho de 13%, comparado a execução do cenário A2 sem nenhuma restrição de recursos de disco. E no cenário A4, similar ao cenário A2, também foram utilizadas duas máquinas virtuais, entretanto, foi aplicada uma restrição de recursos de 80% x 20%. Neste cenário teve-se um tempo de execução de 1795 segundos, gerando uma perda de 110%, comparado a execução do cenário A2 sem nenhuma restrição de recursos de disco. O cenário A4, o contêiner 1 foi finalizado bem antes que o contêiner 1 do cenário A2 devido a priorização de recursos de 80%, entretanto, o contêiner 2 ficou estati-

camente com apenas 20%, gerando um tempo de execução maior do que o comparado no cenário A3 com 50% dos recursos, logo tendo uma perda significativa por não ter um ajuste dinâmico.

No grupo B, no cenário B1 foi utilizado um *workload* OLTP com apenas uma máquina virtual, sem restrição de recursos de disco e teve um tempo de duração de 335 segundos. Já o cenário B2 foram utilizadas duas máquinas virtuais com concorrência de recursos e assim como o cenário B1, também não foi aplicado nenhuma restrição nos recursos de disco. Este cenário B2 teve como tempo de duração de 1165 segundos, gerando um tempo em torno de quatro vezes mais que a execução isolada apresentada no cenário B1. Já o cenário B3, similar ao cenário B2, também foram utilizadas duas máquinas virtuais, entretanto, foi aplicado uma restrição de recursos de 50% x 50%. Neste cenário teve um tempo de execução de 940 segundos e um ganho de 19%, comparado a execução do cenário B2 sem nenhuma restrição de recursos de disco. E no cenário B4, similar ao cenário B2, também foram utilizadas duas máquinas virtuais, entretanto, foi aplicada uma restrição de recursos de 80% x 20%. Neste cenário teve-se um tempo de execução de 1640 segundos e uma perda de 41%, comparado a execução do cenário A2 sem nenhuma restrição de recursos de disco. O cenário B4, o contêiner 1 foi finalizado bem antes que o contêiner 1 do cenário B2 devido a priorização de recursos de 80%, entretanto, o contêiner 2 ficou estaticamente com apenas 20%, gerando um tempo de execução maior do que o comparado no cenário B3 com 50% dos recursos, logo tendo uma perda significativa por não ter um ajuste dinâmico.

No grupo C, no cenário C1 foram utilizadas duas máquinas virtuais, o contêiner 1 foi utilizado um *workload* DW e no contêiner 2 foi utilizado um *workload* OLTP, sem restrição de recursos de disco e teve um tempo de duração de 1165 segundos. Já o cenário C2, similar ao cenário C1, também foram utilizadas duas máquinas virtuais, entretanto, foi aplicada uma restrição de recursos de 50% x 50%. Neste cenário teve-se um tempo de execução de 980 segundos, gerando um ganho de 16%, comparado a execução do cenário C1 sem nenhuma restrição de recursos de disco. E no cenário C3, similar ao cenário C2, também foram utilizadas duas máquinas virtuais, entretanto, foi aplicada uma restrição de recursos de 80% x 20%. Neste cenário teve-se um tempo de execução de 1580 segundos, gerando uma perda de 36%, comparado a execução do cenário C1 sem nenhuma restrição de recursos de disco. O cenário C3, o contêiner 1 foi finalizado bem antes que o contêiner 1 do cenário C2 devido a priorização de recursos de 80%, entretanto, o contêiner 2 ficou estaticamente com apenas 20%, gerando um tempo de execução maior do que o comparado no cenário C2 com 50% dos recursos, logo tendo uma perda significativa por não ter um ajuste dinâmico.

Em relação ao *Speedup* dos cenários estáticos podemos dividir a análise em dois grupos, o que obtiveram ganho de performance e os que obtiveram perda de performance. Os cenários A3, B3 e C2 obtiveram os melhores resultados em relação aos demais cenários, comprovando que um ajuste estático pode gerar um ganhos de performance, onde: (i) A3

teve um *Speedup* de 1,15 e ganho de 13% em relação ao A2; (ii) B3 teve um *Speedup* de 1,23 e ganho de 19% em relação ao B2 e; (iii) C2 teve um *Speedup* de 1,18 e ganho de 16% em relação ao C1. Já os cenários A4, B4 e C3 obtiveram perda de performance, e o motivo desta perda é devido uma ineficiência no ajuste estático que não permite a realocação de recursos ociosos em processos em andamento, comprovando que um ajuste dinâmico poderia trazer ganhos para estes cenários, onde: (i) A4 teve um *Speedup* de 0,47 em relação ao A2; (ii) B4 teve um *Speedup* de 0,71 em relação ao B2 e; (iii) C3 teve um *Speedup* de 0,73 em relação ao C1.

Com base nos dados abordados nesta seção pode-se responder a questão de pesquisa Q1 - Qual o impacto do ajuste estático de recursos de disco proposto pelo trabalho de Matteussi *et al.* [19], levando em consideração o *workload* de banco de dados? Sim. Neste caso, pode-se observar ganhos no ajuste estático de recursos de disco, assim como no trabalho realizado por Matteussi, mas com o *workload* de banco de dados. Isso pode ser comprovado através do tempo de execução entre os cenários sem ajuste de recursos de disco (A2, B2 e C1) com tempos de execução em segundos de 855, 1165 e 1165 respectivamente e com ajuste estático (A3, B3, C2) que executaram em 740, 940 e 980 respectivamente. Assim, pode-se identificar que os testes realizados com o ajuste estático executaram mais rapidamente que os cenários sem ajuste, como foi observado pelo trabalho do Kassiano [19] em seu trabalho com o *workload* Hadoop. Os ganhos observados nos experimentos com ajuste estático em relação ao testes sem ajuste de recursos foram de 13% para o grupo de testes A3 (DW x DW), 19% para o grupo de testes B3 (OLTP x OLTP) e 16% para o grupo de testes C2 (DW x OLTP). Além disso, também pode ser avaliado a hipótese de pesquisa H1 - Os ganhos obtidos pelo estudo realizado por Matteussi *et al.* [19] em um *workload* do tipo *Hadoop*, aplicam-se também em um *workload* de banco de dados. Baseado na resposta a questão de pesquisa Q1, esta hipótese foi comprovada, uma vez que também se observou ganhos de performance em aplicações de banco de dados, assim como observado nos experimentos realizados por Matteussi *et al.* [19] em *Hadoop*.

Entretanto, uma política dinâmica de alocação pode trazer maiores benefícios, uma vez que o ajuste de recursos de E/S irá refletir durante todo o ciclo de execução, podendo se adaptar conforme a demanda de recursos necessários para processar as transações de banco de dados. Um exemplo disso pode ser observado no cenário de teste A4. Neste experimento foram utilizadas duas máquinas virtuais, uma com restrição de 80% dos recursos de disco e outra com 20%. Como observado no gráfico 4.2, o ambiente com maior alocação de recursos de disco (80%) executou mais rapidamente e após seu término, o outro ambiente com alocação de 20% manteve-se com esta alocação, mesmo ainda havendo processamento pendente a ser executado, não houve alteração na sua alocação de recursos, devido ao método de ajuste estático. Neste momento com apenas esta aplicação com a alocação reduzida e estática de 20%, ainda haviam 80% destes recursos disponíveis e que poderiam ser reaproveitados para que o processamento pendente neste ambiente fosse acelerado. Com a utilização de uma política dinâmica, o ajuste de recursos estaria

sendo ajustado de forma flexível, com um melhor aproveitamento dos mesmos e com o objetivo de acelerar as aplicações de banco de dados. Desta forma, necessita-se responder a questão Q2 e sua hipótese de H2, referente a aplicação de uma política dinâmica de alocações de recursos de disco para otimizar a performance de aplicações de banco de dados. O capítulo 5 a seguir descreve a proposta para uma política dinâmica de alocação de recursos de disco, assim como no capítulo 6 apresentará os testes realizados para responder a esta questão de pesquisa Q2.

5. DDPM: UMA POLÍTICA PARA OTIMIZAR A EXECUÇÃO DE TRANSAÇÕES DE BANCO DE DADOS

Conforme os resultados obtidos no capítulo anterior 4 uma política dinâmica gera benefícios de performance em aplicações de banco de dados em um ambiente compartilhado, que é a motivação deste trabalho. Com isso, a seguir é apresentada a proposta de uma política dinâmica de ajuste de recursos de disco.

A seguir são apresentados as seções princípios gerais, arquitetura genérica e funcionamento da política.

5.1 Princípios Gerais

O trabalho realizado por Matteussi *et al.* [19] teve como foco o problema de contenção de disco ocorrido em aplicações de *Hadoop*. Estas aplicações precisam analisar e processar grandes quantidades de dados em paralelo, como é o caso das aplicações *Hadoop* que são comumente conhecidas por efetuarem operações de E/S de forma intensiva sobre os nós de um *cluster*. Baseado nisso, Kassiano realizou um estudo para aplicar um ajuste manual, com o objetivo de acelerar os processos de MapReduce do *Hadoop*, a fim de otimizar a alocação de recursos e acelerar a execução destas aplicações. Conforme os resultados obtidos no trabalho de Matteussi *et al.* [19] foi comprovado que este objetivo é verdadeiro e se aplica a este cenário. Além disso, esta política proposta caracteriza-se como reativa, uma vez que ela estaria avaliando a utilização dos recursos de disco do ambiente em tempo de execução.

A partir deste estudo foram definidas hipóteses e questões de pesquisa descritas no capítulo 4 para avaliar se o estudo realizado por Matteussi *et al.* [19] pode ser aplicado em um cenário de banco de dados transacional em ambientes virtualizados. Conforme demonstrado nos experimentos realizados no capítulo anterior, foi observado que através do ajuste estatístico de recursos, pode-se obter ganhos de performance comparado a execução concorrente de ambientes sem qualquer controle nos recursos de E/S. Entretanto, durante os testes realizados ainda observa-se ociosidade de recursos neste método, uma vez que não ocorre nenhuma adaptação nesta alocação durante o ciclo de execução destes ambientes. Como exemplo deste comportamento, os experimentos de teste A4, B4 e C3 realizados no capítulo anterior demonstram a utilizam de um ambiente com priorização de recursos (80%) e outro com uma menor alocação (20%), após a aplicação com maior alocação é finalizada, a outra se mantém com a mesma alocação devido ao ajuste estático utilizado. A partir disso, pode-se identificar uma oportunidade de melhoria nesta alocação de recursos de disco, uma vez que isso não pode ser definida de forma estática, mas se adaptar conforme a demanda pelas aplicações em execução. Baseado nisso, foi definida

a política chamada de DDPM (*Database Disk Performance Manager*) com o objetivo de otimizar a execução de aplicações de banco de dados através de uma melhor alocação de recursos em ambientes virtualizados. Esta política pretende distribuir as capacidades de E/S entre as máquinas virtuais que estão concorrendo por seus recursos de disco de forma gerenciada, a partir disso, será calculada as capacidades para cada ambiente com base em uma métrica de performance chamada TPS, que foi apresentado na seção 2.2.2.2. Logo, o tempo de execução menor, traz maior benefício para o ambiente como um todo, pois ao concluir o processamento será liberado os recursos de disco que estavam sendo consumidos, para que a política possa redistribuir estes recursos de disco para as demais aplicações que ainda estão em execução naquele momento.

5.2 Arquitetura Genérica

A execução de múltiplas máquinas virtuais em um ambiente consolidado tem como um de seus principais problemas a contenção de recursos de disco. Afetando diretamente o desempenho e a capacidade de concluir uma operação (transação de banco de dados, processo de sistema operacional, etc.) em tempo hábil. Desta forma, uma alocação de recursos de disco mais apropriada, tem como objetivo uma distribuição otimizada entre os ambientes solicitantes, consequentemente melhorando a performance geral.

Este problema de contenção de disco impacta diretamente na performance das aplicações, gerando um aumento no tempo de execução das mesmas. O capítulo 3 apresentou o estado da arte e algumas soluções que buscam amenizar o impacto desta contenção, entretanto, estas soluções não atendem de forma satisfatória este problema, devido aos pontos já identificados, tais como, necessidade de *software* proprietário, limitação de tecnologia de banco de dados, por exemplo, banco de dados Oracle, arquitetura tradicional de virtualização que não permite o ajuste dinâmico dos recursos sem impacto nas aplicações já em execução, aplicação de restrição de recursos em apenas leitura ou escrita, assim como, falta de testes efetivos com dados reais para avaliar estas soluções encontradas na literatura. Logo, não sendo factível a sua aplicabilidade em um cenário corporativo devido a estes pontos negativos.

Com isso, este trabalho, apresenta uma proposta que pode ser aplicada em vários banco de dados, tais como: MySQL, Oracle e PostgreSQL. Uma vez que bastará utilizar um *driver* de conexão apropriado para o banco de dados desejado, não dependendo de qual banco de dados foi utilizado, já que este ajuste fino é realizado a nível de sistema operacional.

A partir disso, foi definida uma arquitetura genérica para atender a política proposta, baseada no modelo criado por Dawoud *et al.* [7], a figura 5.1 apresenta esta arquitetura que possui os seguintes elementos: Controlador QoS (*Quality of Service*), Controlador de Recursos, Monitor de Recursos, Gerenciador de Recursos, Monitor de Performance, um

conjunto de diversas máquinas virtuais e seus respectivos banco de dados com um gerenciador de recursos para cada banco de dados:

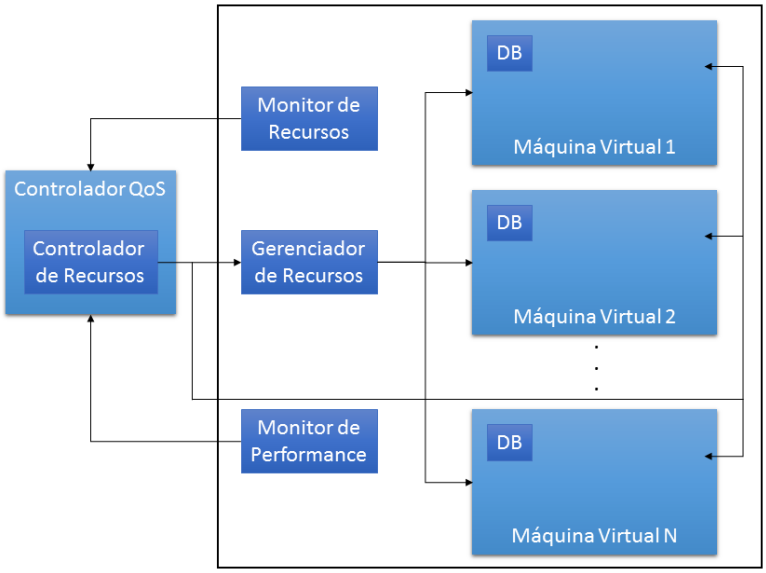

Figura 5.1 – Arquitetura genérica da política baseada no modelo de Dawoud *et al.* [7]

A seguir são detalhados os elementos da arquitetura genérica.

i) Controlador QoS

Como o recurso de disco de um servidor possui um limite máximo para sua capacidade de E/S, isto define uma taxa de limitação para as operações de Leitura e para Escrita desempenhadas por um processo no sistema operacional. O controlador QoS define um nível de serviço para E/S baseado na capacidade máxima para leitura e para escrita, assim, definindo os parâmetros de controle para os demais elementos da arquitetura.

ii) Controlador de Recursos

O controlador de recursos recebe dados do Monitor de Recursos e do Monitor de Performance, a fim de avaliar a taxa atual de utilização de recursos e evitar gargalos de performance. Ele realiza a alocação e desalocação de recursos conforme a demanda pelos ambientes de banco de dados e as transações em execução.

iii) Monitor de Recursos

O monitor de recursos dinamicamente mede o consumo de recursos e atualiza o controlador QoS com as novas medições. O módulo depende da plataforma de virtualização (LXC, Xen, VMWare, etc.) para medir o uso de recursos de disco para cada máquina virtual.

iv) Monitor de Performance

O monitor de performance mantém o controlador QoS atualizado, baseado em métricas de performance, principalmente sobre TPS. Esta métrica é apresentada em detalhes na seção 2.2.2.2.

v) Gerenciador de Recursos

O gerenciador de recursos depende do recurso CGroups ou *Control Groups* que é um recurso do *kernel do Linux* para controlar limites dos recursos do sistema operacional, como disco, memória, CPU, etc. O *CGroups* não permite que a máquina virtual exceda os valores predefinidos no recurso controlado, neste caso, o foco é na utilização do disco por parte de cada máquina virtual.

5.3 Funcionamento da Política

A política proposta possui um conjunto de processos que garantem uma melhor alocação dos recursos de disco proporcionando otimização do processamento em aplicações de banco de dados em máquinas virtuais, logo este tipo de alocação é mais eficaz que a alocação sem ajuste ou com ajuste estático. Enquanto a alocação sem ajuste permite uma sobrecarga de alocação para recursos de E/S, a alocação estática mitiga este problema, mas permite que os recursos fiquem ociosos, pois não redistribui de forma dinâmica, já a alocação dinâmica trata todos estes pontos.

Na figura 5.2 é demonstrado um exemplo do funcionamento da política dinâmica proposta. No *snapshot* 0, pode ser visto um pico de utilização do contêiner 1, após a entrada dos contêineres 2 e 3, todos disputam recursos de E/S (*snapshots* de 0 a 5), neste caso, ainda não houve uma utilização de recursos igual ou acima da taxa máxima de E/S definidos na seção 4.1.2. Após a disputa dos contêineres por recursos de disco (no *snapshot* 5), é iniciada a política para aplicar uma restrição dinâmica de recursos em todos os contêineres em execução. Como será descrito na próxima seção sobre os processos envolvidos nesta política, a mesma realiza a avaliação de recursos de disco a cada 1 segundo, recalculando a restrição a ser aplicada a cada contêiner com base na métrica de TPS. Isto ocorre durante todo o ciclo de execução e não apenas quando um contêiner é finalizado. O fluxo de funcionamento da política é descrito na figura 5.3. Como a política realiza a monitoração de transações de banco de dados, as mesmas muitas vezes podem ter duração de milissegundos dependendo da duração de uma execução de uma operação de leitura ou escrita de dados no banco de dados. Desta forma, foi definido a coleta do status atual do ambiente a cada um segundo para que possa ser avaliado alguma mudança no ambiente, uma vez que um tempo menor que isso, pode não haver mudanças perceptíveis no banco de dados. Foram realizados alguns testes iniciais durante o desenvolvimento desta política e com base dos *workloads* avaliados, a partir de um segundo, começou a ser avaliado

mudanças no ambiente. O mesmo princípio pode ser válido para execuções de durações mais longas. Como será demostrado no capítulo 6, a execução dos *workloads* com e sem a utilização da política não tiveram impacto na execução da transação, pelo contrário, com a sua utilização obteve-se melhorias na sua performance. Baseado nisso, o exemplo descrito na figura 5.2, o contêiner 1, tem utilização maior de recursos, pois tem a maior taxa de TPS no banco de dados, portanto, a política prioriza este contêiner em relação ao contêiner 2 e o 3, definindo menos recursos a eles. Uma vez que o contêiner 1 é finalizado, a política vai alocando gradualmente todos os recursos ao contêiner 2 devido a ter uma maior taxa de TPS em comparação ao contêiner 3 neste exemplo de funcionamento, como pode ser observado entre os *snapshots* 75 e 120. O restante dos recursos são alocados ao contêiner 3. Uma vez, o contêiner 2 tenha finalizado sua execução (*snapshot* 120), o contêiner 3 obteve mais recursos de disco, aumentando sua taxa de utilização e finalizando no *snapshot* 145.

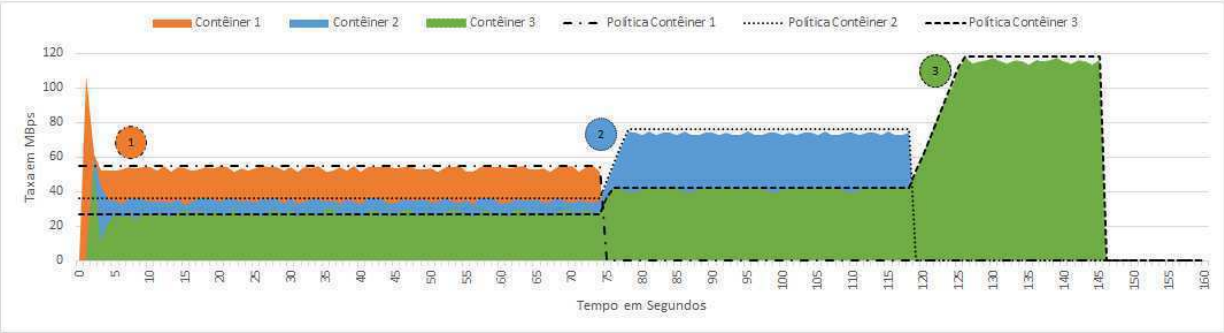

Figura 5.2 – Exemplo de funcionamento da política dinâmica

A partir disso, pode-se definir um fluxograma para ilustrar os principais processos envolvidos nesta política, conforme representado a seguir na figura 5.3:

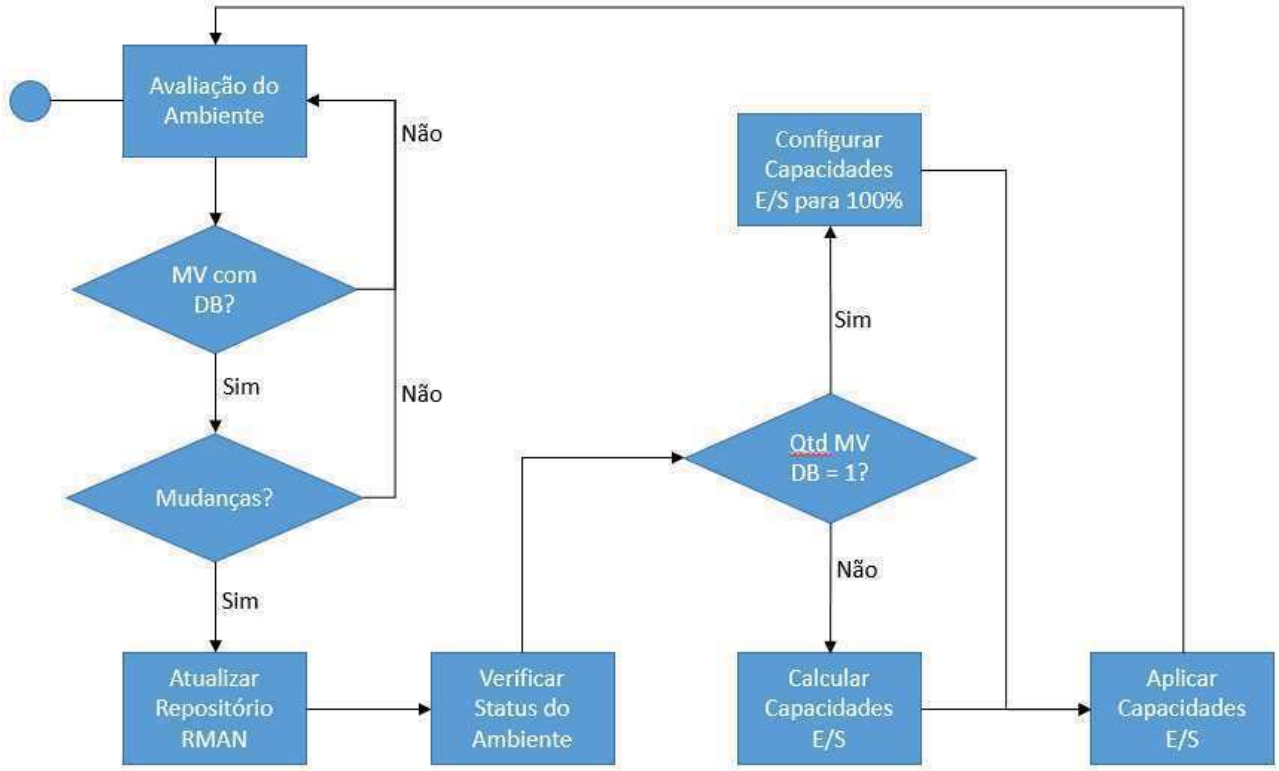

Figura 5.3 – Fluxograma da política de alocação de recursos dinâmica

A seguir são detalhados cada um dos processos da política proposta apresentados na figura 5.3.

i) Processo Avaliação do Ambiente

Esta etapa compreende a verificação do ambiente para identificar a existência de alguma máquina virtual em execução no momento. Além disso, se ocorrer alguma mudança no ambiente, como o desligamento de uma máquina virtual ou banco de dados, bem como a inclusão de novas máquinas virtuais. Além disso, é responsável por coletar métricas de performance referente a taxa de TPS do banco de dados e as atuais definições de capacidades de E/S desta máquina virtual. É com base nestas informações, que é avaliada a atual utilização dos recursos de disco do ambiente e o que ainda está disponível, para que o gerenciador de recursos aplique a política.

A Avaliação do Ambiente é um elemento em constante execução, por isso estas informações são coletadas a cada 1 segundo, para auxiliar na identificação de qualquer mudança no ambiente em que esta política está aplicada, a fim de minimizar o tempo de ociosidade no caso de conclusões de transações, e/ou maior otimização no caso de re-distribuição dos recursos de disco devido a mudanças das métricas ou entrada de novas transações. Como o comportamento das aplicações de banco de dados, possuem uma oscilação na frequência de transações muito alta, o tempo de busca de mudanças também deve ser baixo.

Estas informações irá apoiar as demais fases desta política, a fim de avaliar a performance e obter dados relevantes referente aos recursos disponíveis no ambiente para que sejam gerenciados pelo DDPM, durante todo o fluxo.

ii) Processo Atualização do Repositório DDPM

Este processo é responsável por receber dados referentes a etapa Avaliação do Ambiente, sempre quando houverem mudanças no ambiente. Ou seja, qualquer mudança de status (ativo/inativo) de uma máquina virtual ou mesmo do banco de dados será atualizado no repositório DDPM, a fim de auxiliar no controle do ambiente monitorado e o cálculo das capacidades do ambiente para distribuir os recursos entre as máquinas virtuais em execução. Além disso, sempre quando uma nova máquina virtual for detectada, será avaliado se a mesma possui um banco de dados ativo ou não. Caso exista um banco de dados em uma nova máquina virtual detectada será coletado dados sobre o mesmo, tais como, nome do banco de dados, se existem transações em execução ou não, se está ativo ou não (desligado) e a atual taxa de TPS. Também são coletados dados da máquina virtual como definições atuais de capacidade de E/S com o objetivo de documentar o uso atual de recursos do ambiente.

Caso alguma máquina virtual nova seja detectada durante a etapa de Avaliação do Ambiente, mas não possua um banco de dados ativo, a mesma não será adicionada no repositório DDPM. Uma vez que esta política tem como foco trabalhar com aplicações de banco de dados.

iii) Processo Verificação de Status do Ambiente

Uma vez que o repositório DDPM está atualizado com os dados de todos os ambientes a serem gerenciados por esta política, o próximo passo é a validação do status de cada máquina virtual e seus respectivos banco de dados. Isso tem como objetivo avaliar quantos ambientes estão atualmente concorrendo por recursos de disco no momento. É importante para que a política possa avaliar os recursos disponíveis de disco e distribuir as alocações de recursos de forma otimizada na execução das aplicações de banco de dados.

Nesta etapa são avaliados os seguintes critérios, tais como, status da máquina virtual (ativa ou não), status do banco de dados (ativo ou não). A partir disso é avaliado a quantidade de máquinas virtuais e banco de dados. Se existir apenas um ambiente atualmente monitorado, o próximo a ser executado será a configuração da capacidade de E/S para 100%, caso contrário deverá ser calculado as capacidades E/S.

iv) Processo Configuração das Capacidades E/S para 100%

Esta etapa é responsável por alocar 100% dos recursos de leitura e escrita de disco para a máquina virtual ter total prioridade sobre as capacidades de E/S do ambiente. Isso será realizado quando as seguintes condições forem atendidas: 1) Existir apenas uma máquina virtual atualmente no ambiente; 2) Existir apenas um banco de dados em execução com alguma transação de dados em execução. Caso ocorra de haver apenas uma máquina virtual, mas com um banco de dados sem nenhuma atividade, o mesmo será considerado em ociosidade, sendo disponibilizados todos os recursos de disco novamente ao ambiente.

v) Processo Cálculo das Capacidades E/S

Como foi descrito na subseção anterior, caso exista apenas uma máquina virtual com base nos critérios definidos acima, as capacidades de E/S serão configurados para 100%. Entretanto, se existir mais de uma máquina virtual com aplicações de banco de dados, será necessário calcular os recursos a serem associados para cada ambiente. Os seguintes critérios são avaliados: 1) Quantidade de máquinas virtuais; 2) Quantidade de banco de dados ativos; 3) Taxa de TPS (*Transaction per Second*).

Como pode ser visto na tabela 5.1, as seguintes variáveis são utilizadas nas fórmulas para cálculo de capacidade de E/S para as máquinas virtuais controladas pela política:

Tabela 5.1 – Variáveis utilizadas nas fórmulas da política

Váriavel	**Descrição**
X	Taxa de leitura
Y	Taxa de escrita
α	Taxa máxima de leitura
β	Taxa máxima de escrita
Q	Quantidade de máquinas virtuais ativas
Z	Valor atual da taxa de TPS capturada da máquina virtual
C	Conjunto de todas as máquinas virtuais monitoradas pela política DDPM
X’	Taxa de leitura
Y’	Taxa de escrita
defineMR	Função para definir a taxa máxima de leitura para uma máquina virtual
defineMW	Função para definir a taxa máxima de escrita para uma máquina virtual
defineRR	Função para definir a taxa de leitura para uma máquina virtual
defineWR	Função para definir a taxa de escrita para uma máquina virtual
defineIR	Função para definir a taxa média de leitura para uma máquina virtual
defineIW	Função para definir a taxa média de escrita para uma máquina virtual

A partir disso, foram definidas as seguintes condições e fórmulas em notação de pseudo-código para simplificar o entendimento:

```
1  int sum(int array[], int n) {
2     if (n < 0)
3       return sum;
4
5     return array[n] + 10 + sum(array, n - 1);
6  }
7
8  int defineMR (int ALFA, int Q, int C[]) {
9     int idx = ALFA/Q;
10    int idx2 = Q/100;
11
12    return idx + (idx * (sum(C, Q)/idx2);
13 }
14
15 int defineMW (int BETA, int Q, int C[]) {
16    int idx = BETA/Q;
17    int idx2 = Q/100;
18
19    return idx + (idx * (sum(C, Q)/idx2);
20 }
21
22 int defineRR (int ALFA, int Q, int C[]) {
23    int idx = ALFA/Q;
24    int idx2 = Q/100;
25
26    return idx - ((idx * (sum(C, Q)/idx2))/Q;
27 }
28
29 int defineWR (int BETA, int Q, int C[]) {
30    int idx = BETA/Q;
31    int idx2 = Q/100;
32
33    return idx - ((idx * (sum(C, Q)/idx2))/Q;
34 }
35
36 int defineIR (int X[], int Q, int C[]) {
37
38    return sum(X,Q)/Q;
39 }
40
41 int defineIW (int Y[], int Q) {
42
43    return sum(Y,Q)/Q;
44 }
```

Listing 5.1 – Funções utilizadas para o cálculo das fórmulas

Sendo X, o ajuste referente as operações de leitura para uma máquina virtual ativa que esteja no domínio de todos os ambientes monitorados pela política. Já a variável Y é a definição do ajuste referente as operações de escrita para uma máquina virtual ativa. A variável Z representa a taxa atual de TPS capturada de um ambiente de banco de dados monitorado pela política. Assim como X' e Y' representam as operações de leitura e escrita de outras máquinas virtuais concorrentes respectivamente, já a variável Q representa todos os bancos de dados ativos detectados pela política, a fim de apoiar no cálculo das capacidades de E/S para as máquinas virtuais.

As variáveis α e β definem as taxas máximas de leitura e escrita respectivamente. Estas taxas são referentes ao servidor que hospeda todas as máquinas virtuais atualmente em execução no ambiente que a política está em execução. Para obtenção destes limites das capacidades de E/S são realizados testes de carga, coletando os dados de cada teste referente as operações de leitura e escrita, a fim de avaliar a capacidade máxima do servidor, coletando uma média entre 50 execuções consecutivas para definir estes valores. Mais detalhes sobre esta validação pode ser verificada na seção 4.1.2.

Desta forma, pode-se identificar as seguintes regras de cálculo para o ajuste de recursos:

Regra 1: Apenas uma máquina virtual ativa

Durante a monitoração realizada pela política, caso seja detectada apenas uma máquina virtual, serão atribuídos as taxas máximas de leitura e escrita para esta máquina virtual, com o objetivo de permitir o uso de todos os recursos disponíveis do ambiente. Isso é definido na primeira linha das condições aonde X = α (taxa máquina de leitura) e Y = β (taxa máxima de escrita) quando Q (quantidade de máquinas virtuais ativas) seja igual a um. Isto é definido na linha um das condições matemáticas apresentadas acima.

Regra 2: Uma máquina virtual ou banco de dados inativo

Caso uma máquina virtual esteja inativa ou mesmo o banco de dados, não possua nenhuma transação em execução no momento ou mesmo sua taxa de TPS esteja igual a zero (linha dois das condições matemáticas apresentadas acima), deverá ser configurado os recursos de E/S para este ambiente igual a zero. Desta forma, liberando recursos que não estão sendo utilizados pela política para serem reutilizados pelas demais máquinas virtuais.

Regra 3: Mais de uma máquina virtual ativa com maior taxa de TPS

Quando a política detecta que existe mais de uma máquina virtual ativa, também é verificado se a mesma possui um banco de dados em execução e ativo (com alguma transação em execução, definido pela métrica de TPS - variável Z), deverão ser utilizadas as demais condições matemáticas definidas acima para este tipo de caso.

Para este cenário, a política verifica se existe uma máquina virtual, caso exista um ambiente com a taxa de TPS (variável Z) maior que todos os demais e a soma da utilização de todas as taxas de leitura ou escrita destas máquinas virtuais sejam maior ou igual a taxa máxima de leitura (α) ou escrita (β) respectivamente. Além de verificar se o ambiente pertence ao conjunto das máquinas virtuais atualmente monitoradas (váriavel C). Sendo definidos pelas linhas três a seis das condições matemáticas definidas acima. A verificação quanto a soma da utilização das taxas de leitura e escrita definem se existem recursos ainda disponíveis de E/S no servidor, ou seja, ainda não utilizados pela política proposta, o ajuste de recursos é gerenciado pelo próprio escalador de E/S do sistema operacional do servidor, caso contrário, a política proposta irá controlar isso. Assim, são definidas quatro funções para este cálculo: defineMR, defineMW, defineRR e defineWR.

a) defineMR

Esta fórmula define o valor da taxa de leitura a ser utilizada quando a regra três acima for satisfatória. Neste caso, primeiramente é calculado a taxa média de leitura para todas as máquinas virtuais com banco de dados ativos que são monitorados pela política, além disso será somado um percentual adicional de 10% quando existir um ambiente com maior taxa de TPS com o objetivo de priorizar um banco de dados com maior processamento, a fim de acelerar seu desempenho. Caso o mesmo em uma verificação posterior, permaneça ainda com a maior taxa de TPS, será adicionado consecutivamente mais 10% até um limite de 90% para priorização deste ambiente com demanda por mais processamento.

b) defineMW

Esta fórmula define o valor da taxa de escrita a ser utilizada quando a regra três acima for satisfatória. Neste caso, primeiramente é calculado a taxa média de escrita para todas as máquinas virtuais com banco de dados ativos que são monitorados pela política, além disso será somado um percentual adicional de 10% quando existir um ambiente com maior taxa de TPS com o objetivo de priorizar um banco de dados com maior processamento, a fim de acelerar seu desempenho. Caso o mesmo em uma verificação posterior, permaneça ainda com a maior taxa de TPS, será adicionado consecutivamente mais 10% até um limite de 90% para priorização deste ambiente com demanda por mais processamento.

c) defineRR

Similar à função defineMR, também realiza o cálculo da taxa de leitura, entretanto, esta função define a taxa a ser reduzida dos demais ambientes que não tem uma taxa TPS maior que um determinado banco de dados. Por exemplo, se existe dois ambientes atualmente em execução, um deles possui a maior taxa de TPS por duas ou mais execuções executivas, o mesmo será acrescido incrementos de 10% e o outro ambiente com menor TPS será decrescido 10%. Reduzindo um percentual dos ambientes com menor TPS e adicionando isso ao ambiente com maior para priorizar o mesmo.

d) defineWR

Similar à função defineMW, também realiza o cálculo da taxa de escrita, entretanto, esta função define a taxa a ser reduzida dos demais ambientes que não tem uma taxa TPS maior que um determinado banco de dados. Por exemplo, se existe dois ambientes atualmente em execução, um deles possui a maior taxa de TPS por duas ou mais execuções executivas, o mesmo será acrescido incrementos de 10% e o outro ambiente com menor TPS será decrescido 10%. Reduzindo um percentual dos ambientes com menor TPS e adicionando isso ao ambiente com maior para priorizar o mesmo.

Regra 4: Mais de uma máquina virtual ativa sem taxa de maior TPS

Similar à a regra três aonde se tem mais de uma máquina virtual atual e um banco de dados com transações em execução, mas com o seguinte cenário: não existe um ambiente com uma maior taxa de TPS, no caso, ambientes em um determinado período com taxa TPS igual. Neste caso, será calculo apenas a soma das taxas de E/S sendo utilizadas para definir um valor médio para ser atribuído para as operações de leitura e escrita. Sendo definidas pelas funções defineIR e defineIW respectivamente.

Esta política não tem o objetivo de calcular a taxa exata de recurso de disco para cada contêiner, mas sim, otimizar a taxa total de leitura e escrita, onde cada contêiner receba recursos adequados para o seu processamento conforme sua necessidade, por isso, é utilizado o percentual de 10% no cálculo da capacidade de E/S. Durante os testes realizados na validação desta política, assim o uso de um percentual inferior a 10% neste cálculo, não gerou benefícios na performance durante o testes, desta forma, chegou-se no valor de 10% como valor inicial, assim como facilitando no cálculo da priorização dos recursos de disco durante a monitoração realizada pela política. Além disso, não está previsto no cálculo a limitação ou tratamento diferenciado para contêineres que fiquem executando infinitamente, já que as aplicações de banco de dados normalmente tem um término para o seu processamento, deferente de outros tipos de aplicações que podem ter este tipo de comportamento.

vi) Processo Aplicação das Capacidades E/S

Conforme descrito na seção anterior, dependendo da quantidade de máquinas virtuais com banco de dados em atividade, será adotado uma regra. Neste caso, se existir apenas um ambiente atualmente ativo, será definido 100% dos recursos de disco para leitura e escrita para este ambiente. Caso contrário será calculado conforme fórmula de cálculo de distribuição mencionada acima para o caso de haver mais de um ambiente em execução. Uma vez que as capacidades tenham já sido calculadas apropriadamente, este processo tem como objetivo de efetivamente aplicar as capacidades definidas para cada máquina virtual controlada pela política.

Após aplicar as definições das capacidades de E/S em todas as máquinas virtuais, o processo retorna a primeira etapa de Avaliação do Ambiente, configurando um processo cíclico que continuamente estará avaliando o ambiente e aplicando as capacidades de E/S com base na taxa atual de TPS de cada banco de dados.

No próximo capítulo será demonstrado o uso desta política dinâmica de alocação de recursos através da implementação de um algoritmo, assim como, a discussão sobre os resultados obtidos.

6. VALIDAÇÃO DOS RESULTADOS

Conforme descrito nos capítulos anteriores, o compartilhamento de recursos por várias máquinas virtuais em um ambiente consolidado pode gerar problemas de contenção de recursos. Em aplicações de banco de dados a contenção de disco é um dos problemas que impacta na performance destas aplicações, o qual foi abordado neste trabalho. Como forma de minimizar este impacto e melhorar a performance este trabalho propôs uma política dinâmica de ajuste de recursos de disco. A partir disso, foram realizados testes com ajuste estático de alocação de recursos de disco para responder a questão de pesquisa Q1, a qual foi respondida no capítulo 4, baseado nestes resultados obtidos foi confirmado que o trabalho realizado por Matteussi *et al.* [19] pode ser aplicado em um cenário de banco de dados tradicionais. Entretanto, a pergunta Q2 ainda está pendente para ser respondida, referente a adoção de uma política dinâmica de alocação de recursos. No capítulo 5 foi apresentada a proposta de uma política chamada DDPM para aplicar o ajuste dinâmico de recursos de disco para aplicações de banco de dados em ambientes virtualizados consolidados.

Este capítulo tem por objetivo apresentar a utilização desta política através da implementação de um algoritmo, bem como a aplicação da mesma nos cenários de testes e avaliar os resultados obtidos, com o objetivo final de responder a pergunta de pesquisa Q2 e a hipótese de pesquisa H2.

6.1 Arquitetura da Política

Com base na arquitetura genérica apresentada na seção 5.2 foi instanciada uma arquitetura da política proposta utilizando como plataforma de virtualização o LXC, com o objetivo de avaliar os resultados obtidos a partir da aplicação da mesma.

Como pode ser visto na figura 6.1 a seguir, a arquitetura é composta pelos seguintes elementos: (i) Monitor de Recursos é o agente responsável pela verificação dos recursos atuais de disco para que sejam disponibilizados ao Gerenciador de Recursos. É dependente do LXC para verificar o status atual dos recursos utilizados no contêiner; (ii) Monitor de Performance é o agente que realiza a interface entre o Gerenciador de Recursos e os bancos de dados presentes nos contêineres. Ele conecta nos bancos de dados remotamente e verifica as transações referente a taxa TPS, baseado na carga de dados que está ocorrendo no momento, para que seja definida uma nova configuração de recursos de E/S para o disco. A partir disso, esta informação é repassada ao Gerenciador de Recursos; (iii) Gerenciador de Recursos recebe os dados do Monitor de Recursos e do

Monitor Performance; (iv) Controlador de Disco realiza a modificação dos limites de disco para cada contêiner envolvido, com o objetivo de proporcionar uma melhor performance aos contêineres com *workloads* em processamento concorrente.

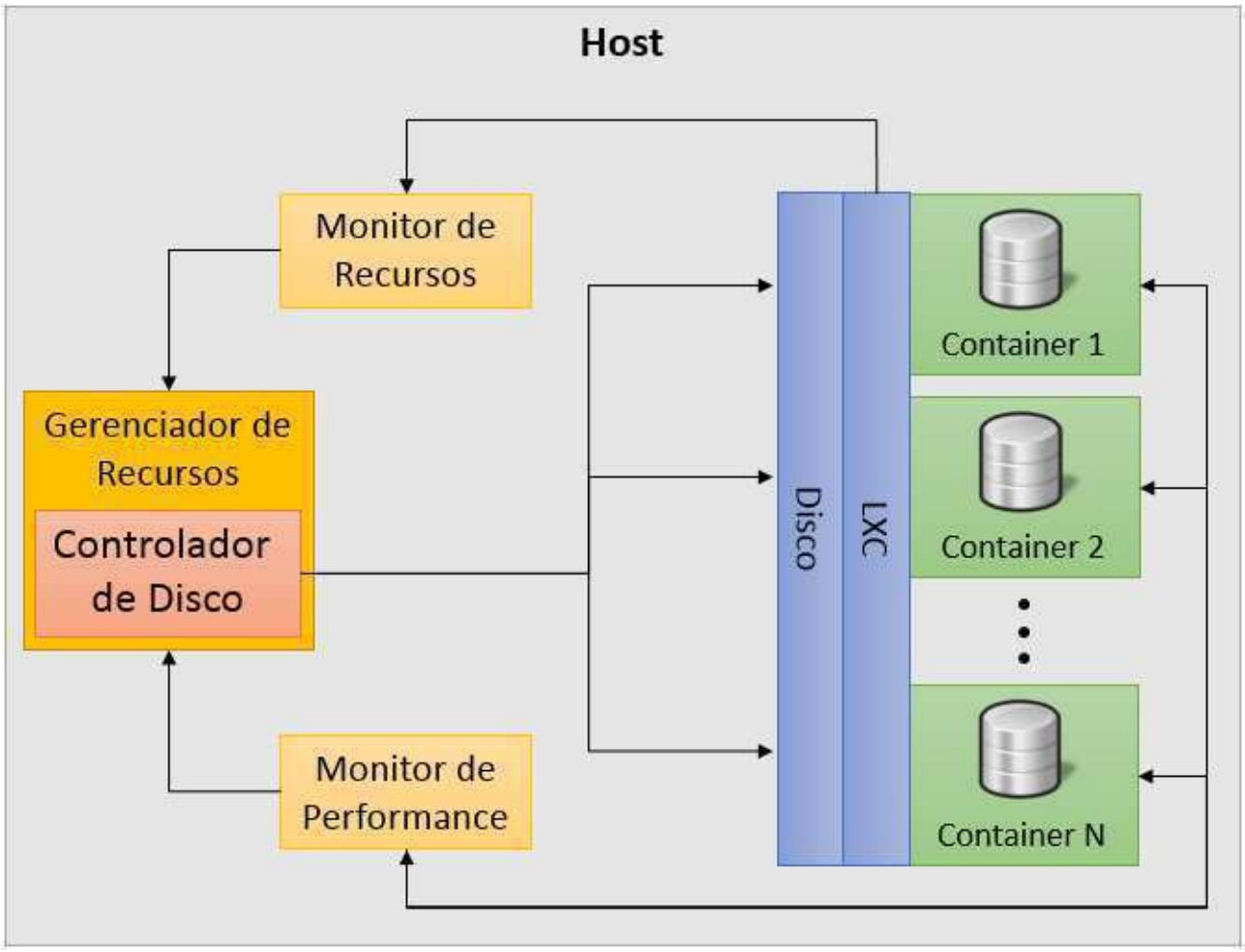

Figura 6.1 – Arquitetura para alocação dinâmica de recursos de disco

A política de ajuste dinâmico de recursos de disco é ativada através do comando DDPM, ele é o responsável por iniciar ou parar a política. Isso ocorre a partir de um parâmetro de controle para iniciar ou parar respectivamente os demais elementos da arquitetura. Além disso, é realizada uma verificação para identificar se o mesmo já está em execução ou não, evitando múltiplas chamadas ao mesmo algoritmo. Para mais detalhes o código do algoritmo pode ser visto no Anexo A.

A seguir são apresentados os elementos que compõem a política DDPM que realiza o ajuste dinâmico de alocação de recursos de disco.

6.1.1 Controlador de Disco

O Controlador de disco tem o objetivo de aplicar uma capacidade de recursos de disco para as operações de leitura e escrita. Isto é realizado através do recurso *CGROUPS*

(descrito em detalhes na seção 2.1.2.2). O mesmo recebe três parâmetros: (i) O contêiner a ser modificado; (ii) Taxa de Leitura que especifica a capacidade para a taxa de leitura a ser aplicada ao contêiner; (iii) Taxa de Escrita que especifica a capacidade para a taxa de escrita a ser aplicada ao contêiner.

A partir disso, o algoritmo verifica se os arquivos de configuração do *CGROUPS* do contêiner especificado existem para que sejam aplicadas as restrições informadas para leitura e escrita. Para mais detalhes o código do algoritmo pode ser visto no Anexo D.

6.1.2 Monitor de Performance

O Monitor de Performance tem como objetivo de verificar em todos os contêineres ativos, se existe algum banco de dados em execução, em caso positivo, o mesmo coletará os seguintes dados:

- TPS: Métrica utilizada para mensurar a atividade do banco de dados, que possibilita a avaliação nos demais módulos da proposta deste trabalho. Neste caso, é capturado a taxa TPS, após isso, é realizada uma nova coleta após um segundo, sendo realizada a diferença entre as duas coletas para definir o valor atual desta taxa. Mais detalhes sobre este tópico, são apresentados na seção 2.2.2.2;

- Status do Banco de Dados: Status que indica se existe processamento em execução no banco de dados ou não, esta informação também é armazenada, juntamente com a taxa TPS. Nessa verificação excluí-se processos internos relacionados ao próprio banco de dados.

As informações capturadas pelo Monitor de Performance são utilizados pelos elementos Monitor de Recursos e Gerenciador de Recursos para referência ao status atual de cada banco de dados em cada contêiner ativo. Para mais detalhes o código do algoritmo pode ser visto no Anexo B.

6.1.3 Monitor de Recursos

O Monitor de Recursos tem como objetivo realizar a coleta dos dados referente aos limites atuais de leitura/escrita do *CGROUPS* de todos os contêineres em execução, assim como, verifica quais contêineres ainda estão em execução, em caso de desligamento devido a alguma parada programada ou mesmo devido a algum erro crítico. Para mais detalhes o código do algoritmo pode ser visto no Anexo C.

6.1.4 Gerenciador de Recursos

O Gerenciador de Recursos tem como objetivo avaliar os dados capturados pelos elementos Monitor de Performance e Monitor de Recursos, para que sejam aplicadas as restrições de disco e priorização da utilização de recursos. O mesmo está dividido nos seguintes componentes:

- verificaDBStatus: Este componente carrega a lista dos bancos de dados ativos, baseado na lista de contêineres em execução que o Monitor de Recursos capturou previamente. Baseado nesta lista, é realizada uma verificação dos seguintes dados: contêiner, status do contêiner (ativo/inativo), status do banco de dados (ativo/inativo) e a taxa TPS. Se o contêiner estiver inativo, é executado o Controlador de Disco para desativar as restrições de disco neste contêiner. Caso contrário, será verificado nos bancos de dados que estão ativos, qual possui a maior taxa de TPS e de qual contêiner, este banco de dados está hospedado. A partir desta verificação em todos os contêineres e os bancos de dados, será chamado o componente atualizarLimitesIO;

- atualizarLimitesIO: Este componente verifica a lista dos bancos de dados ativos, caso haja apenas um banco de dados ativo, será alocado 100% dos recursos a este contêiner. Caso contrário, será dividido a taxa máxima de leitura/escrita previamente capturados entre a quantidade de banco de dados ativos. Após isso, o contêiner que possui maior taxa TPS, receberá o valor de capacidade de disco de leitura/escrita + um índice de incremento, definido inicialmente como 10% do valor calculado da taxa de leitura/escrita. O objetivo é que processos concorrentes com maior taxa TPS possam ser finalizados antecipadamente, liberando mais recursos e possa ser alocado mais recursos aos contêineres restantes. Caso ocorra recorrência do mesmo contêiner que necessite ser priorizado, esta taxa será incrementada em 10% sucessivamente para priorizar os recursos deste contêiner. Além disso, os demais contêineres com a taxa TPS menor, serão decrescidos pelo valor atual desta taxa de inicial em 10%.

Para mais detalhes o código do algoritmo está disponível no Anexo E.

6.2 Configuração do ambiente de testes

O ambiente utilizado para os testes é composto pela seguinte configuração de *hardware*: Dell Poweredge R610, CPU com 16 núcleos, memória de 16 GB de RAM, disco rígido de 150GB, sistema operacional do host Ubuntu Server 16.04, plataforma de virtualização LXC, sistema operacional dos contêineres *Oracle Enterprise Linux* 6.5 e banco de dados *Oracle Enterprise Database* 12c.

Foram criados dois contêineres chamados lxc-ora01 e lxc-ora02, todos utilizando o sistema operacional *Oracle Enterprise* Linux 6.5 (durante a criação do contêiner, foi utilizada a opção *template* = oracle do LXC) e instalado uma instância de banco de dados Oracle em cada contêiner. A seguir serão descritos os passos realizados nos testes para responder a questão de pesquisa Q2 (seção 4.1.4). Além disso, para definição das capacidades dos recursos, foi utilizado o recurso *cgroups* presente no LXC, utilizando os seguintes processos: 1) `blkio.throttle.read_bps_device` / `blkio.throttle.write_bps_device` – disco; 2) `cpuset.cpus` – CPU; 3) `memory.limit_in_bytes` – memória.

Para os testes realizados com a política de ajuste dinâmico de recursos de disco foram utilizados a mesma largura de banda para as operações de leitura e escrita identificados na seção 4.1.2, onde a taxa média de leitura é de 118 Mbps e de 108 Mbps para escrita.

6.3 Ajuste Dinâmico das Restrições de Disco

Conforme foi descrito nos testes realizados na seção 4.1.3, foi aplicado um ajuste estático definindo algumas configurações de restrição de disco: isolado sem restrição, sem restrição com concorrência, com restrição estática com concorrência de 50% x 50%, em ambos tipos de *workloads*, assim como evidenciado pelo trabalho de Matteussi *et al.* [19] que existem ganhos ao se restringir os recursos de disco, em comparação, a execução sem qualquer controle para processos com concorrência de recursos. Com base nisso, foram definidos alguns cenários de testes para demonstrar a política proposta para alocação de recursos de disco de forma dinâmica, a fim de, realizar o mesmo procedimento sem a intervenção manual e estática.

Conforme descrito na seção 6.2, o servidor onde as máquinas virtuais foram executadas tem 16GB de memória RAM e 16 CPU cores. Para garantir que não ocorresse intervenção de outro tipo de contenção nos testes realizados, como por exemplo, CPU e memória, os recursos foram divididos de forma igual entre os dois contêineres em execução permitindo que apenas a contenção de disco pudesse ser avaliada durante a execução dos testes. Além disso, as taxas máximas de E/S são baseados nos dados obtidos na seção 4.1.2, como limites para a alocação dos recursos de disco que são utilizados pela política de ajuste dinâmico e para as demais alocações neste grupo.

Desta maneira foram realizados testes com dois contêineres com *workloads* semelhantes, utilizando as seguintes ferramentas: Oracle Orion [24] e Oracle RAT [25]. Sendo o Orion [24], uma ferramenta de *benchmark* de disco e o Oracle RAT uma ferramenta de captura de *workload* de um ambiente real, para que seja reproduzido em outro de banco de dados transacional. Neste caso, foram utilizados *workloads* de dados reais de uma empresa de grande porte de TI. Neste teste, a mesma carga de dados foi utilizada para efeito de reprodução de uma situação real encontrada no dia a dia das empresas. Neste caso,

foram realizados dois cenários de teste utilizando um *workload* do tipo DW e outro do tipo OLTP.

Os cenários de teste foram agrupados da seguinte forma: o grupo D é composto por testes realizados com o *workload* do tipo OLTP, o grupo E é composto por testes realizados com o *workload* do tipo DW e, por fim, o grupo F é composto por testes realizados com o *workload* do tipo DW e OLTP. A tabela 6.1 a seguir apresenta um resumo de todos os cenários de testes realizados para estes grupos:

Tabela 6.1 – Cenários de teste para o ajuste dinâmico das restrições de disco

Cenário de Teste	**Restrição**	**Workload**	**Operação E/S**
D1	-	OLTP x OLTP	Leitura e Escrita
D2	Estática - 50% x 50%	OLTP x OLTP	Leitura e Escrita
D3	Dinâmica	OLTP x OLTP	Leitura e Escrita
E1	-	DW x DW	Leitura
E2	Estática - 50% x 50%	DW x DW	Leitura
E3	Dinâmica	DW x DW	Leitura
F1	-	DW x OLTP	Leitura e Escrita
F2	Estática - 50% x 50%	DW x OLTP	Leitura e Escrita
F3	Dinâmica	DW x OLTP	Leitura e Escrita

Além disso, o foco dos testes realizados foi exemplificar o pior caso de contenção de recursos de disco, onde todos os contêineres estão sendo executados ao mesmo tempo, com suas respectivas cargas de trabalho para avaliar o comportamento dos cenários de teste com as diferentes configurações de restrição de recursos. A seguir serão apresentados os detalhes dos cenários de teste dos grupos D, E e F.

6.3.1 Testes do Grupo D - *Workloads* OLTP

Foram utilizados *workloads* do tipo OLTP nos testes do grupo D, onde existem processos de leitura e escrita que é uma característica deste tipo de *workload*.

A figura 6.2 apresenta um gráfico com os resultados de execução de cada cenário de teste realizado, onde: D1 está com processos em concorrência e sem restrição; D2 está com processos em concorrência e com restrição estática; e D3 está com processos em concorrência e com restrição dinâmica a partir da política DDPM.

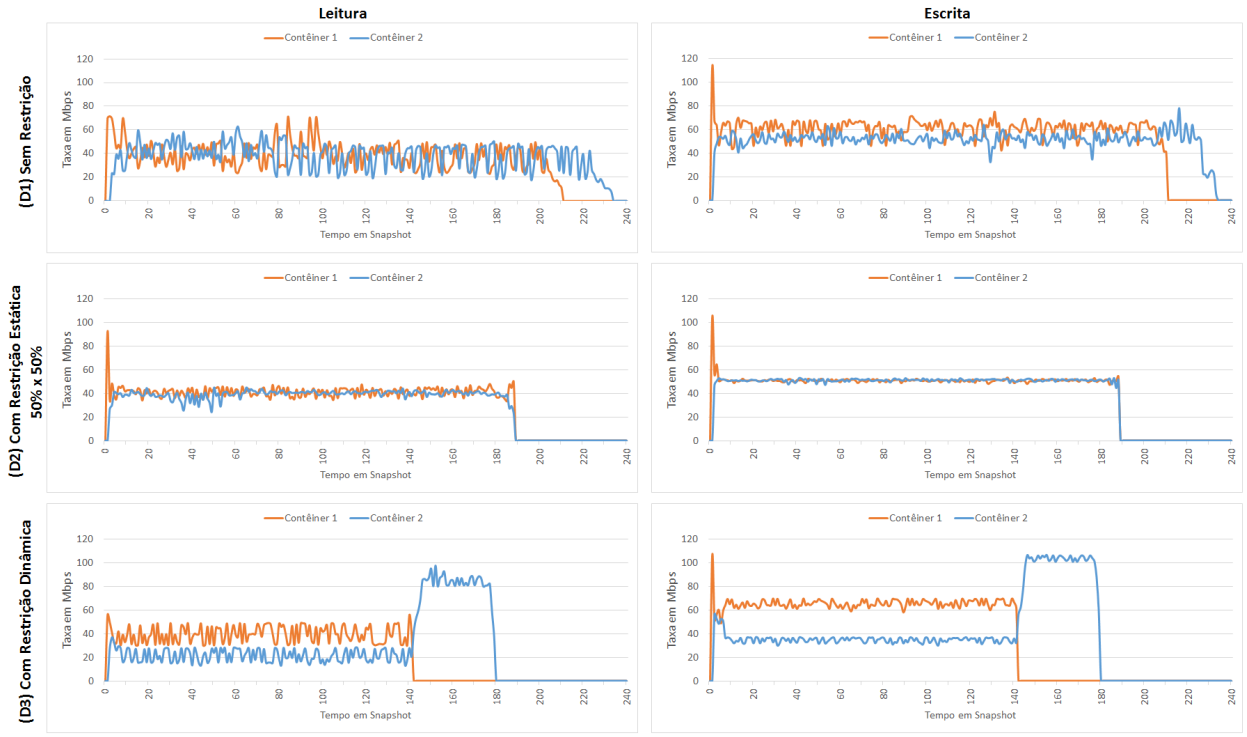

Figura 6.2 – Experimentos realizados com ajuste estático e dinâmico em *workloads* OLTP

A seguir serão detalhados os cenários de teste D1, D2 e D3 do grupo D.

Cenário D1: Execução de *Workload* OLTP x OLTP Sem Restrição

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo OLTP sem restrição de recursos de disco, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico D1 apresentado anteriormente na figura 6.2, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo OLTP. O tempo de execução foi de 1165 segundos.

Cenário D2: Execução de *Workload* OLTP x OLTP Com Restrição Estática

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo OLTP com restrição de 50% de recursos de disco para cada contêiner, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico D2 apresentado anteriormente na figura 6.2, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo OLTP. O tempo de execução foi de 940 segundos.

Cenário D3: Execução de *Workload* OLTP x OLTP Com Restrição Dinâmica

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo OLTP com restrição dinâmica através da utilização da política DDPM para

os recursos de disco para cada contêiner, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico D3 apresentado anteriormente na figura 6.2, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo OLTP. O tempo de execução foi de 895 segundos.

6.3.2 Testes do Grupo E - *Workloads* DW

Foram utilizados *workloads* do tipo DW nos testes do grupo E, onde existem apenas processos de leitura, cujo é uma característica deste tipo de *workload*.

A figura 6.3 a seguir apresenta um gráfico com os resultados de execução de cada cenário de teste realizado, onde: E1 está com processos em concorrência e sem restrição; E2 está com processos em concorrência e com restrição estática; e E3 está com processos em concorrência e com restrição dinâmica a partir da política DDPM.

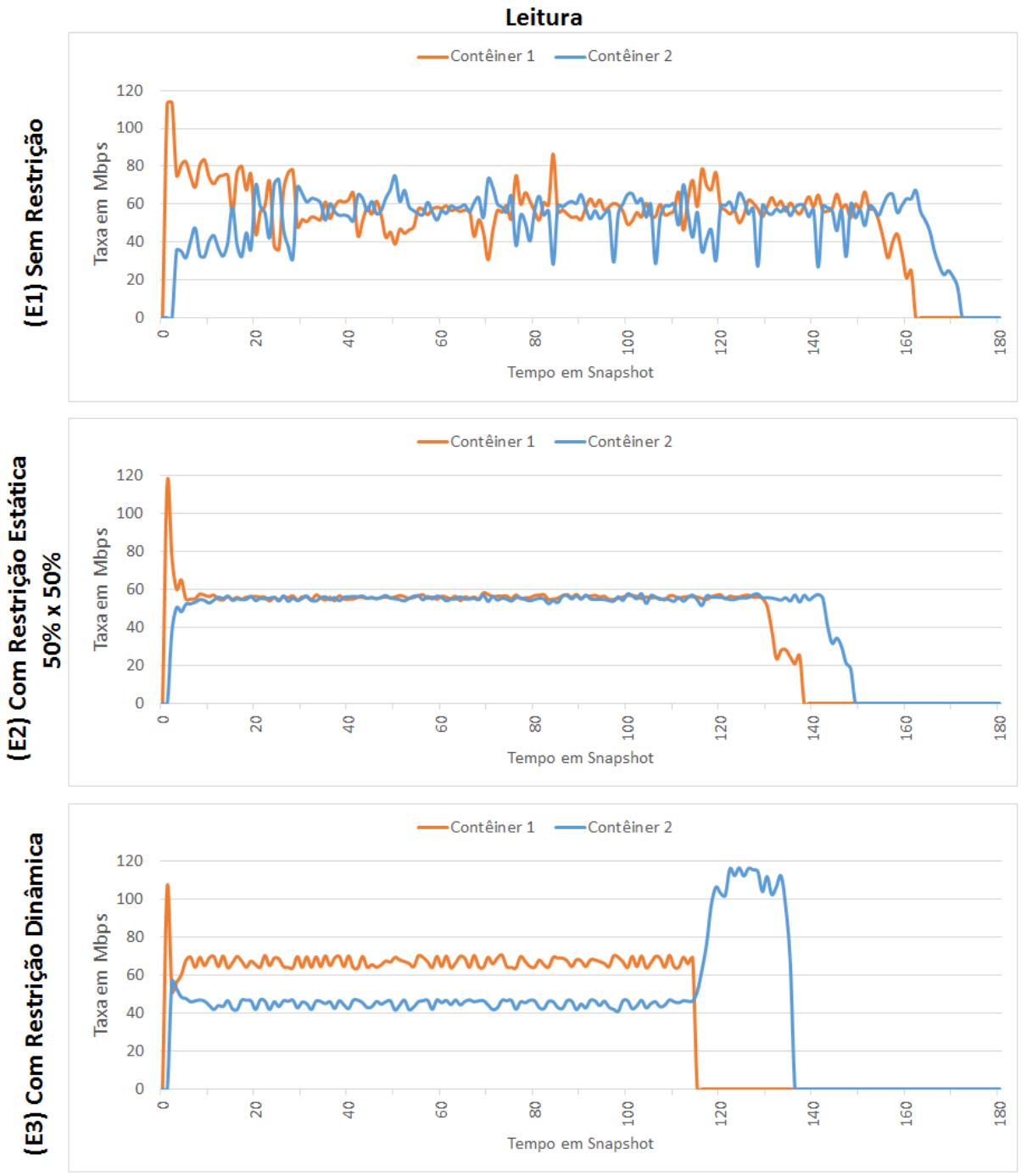

Figura 6.3 – Experimentos realizados com ajuste estático e dinâmico em *workloads* DW

A seguir serão detalhados os cenários de teste E1, E2 e E3 do grupo E.

Cenário E1: Execução de *Workload* DW x DW - Sem Restrição

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo DW sem restrição de recursos de disco, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico E1 apresentado anteriormente na

figura 6.3, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo DW. O tempo de execução foi de 855 segundos.

Cenário E2: Execução de *Workload* DW x DW Com Restrição Estática

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo DW com restrição de 50% de recursos de disco para cada contêiner, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico E2 apresentado anteriormente na figura 6.3, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo DW. O tempo de execução foi de 740 segundos.

Cenário E3: Execução de *Workload* DW x DW Com Restrição Dinâmica

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads* do tipo DW com restrição dinâmica através da utilização da política DDPM para os recursos de disco para cada contêiner, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho O gráfico E3 apresentado anteriormente na figura 6.3, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo DW. O tempo de execução foi de 675 segundos.

6.3.3 Testes do Grupo F - *Workloads* DW x OLTP

Foram utilizados *workloads* do tipo DW e OLTP nos testes do grupo F, onde existem processos de leitura e escrita em concorrência, devido a dois tipos diferentes de *workload* em execução.

A figura 6.4 a seguir apresenta um gráfico com os resultados de execução de cada cenário de teste realizado, onde: F1 está com processos em concorrência e sem restrição; F2 está com processos em concorrência e com restrição estática; e F3 está com processos em concorrência e com restrição dinâmica a partir da política DDPM:

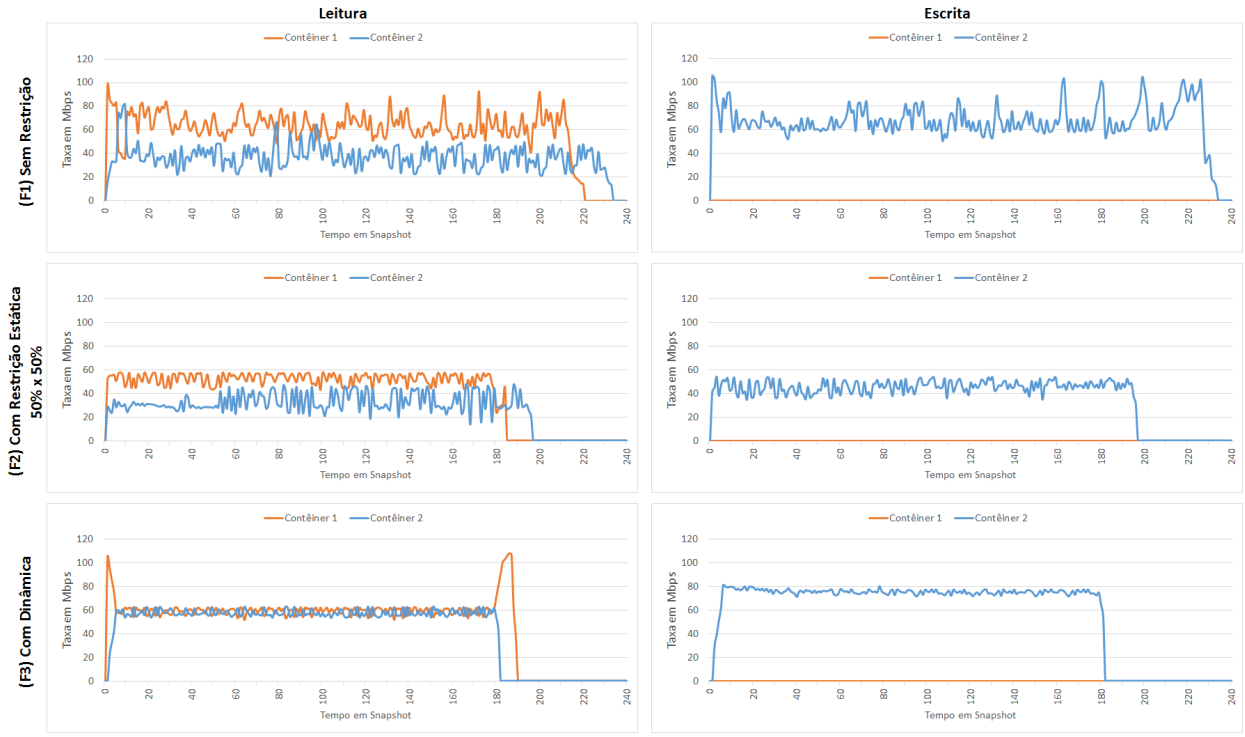

Figura 6.4 – Experimentos realizados com ajuste estático e dinâmico em *workloads* DW e OLTP

A seguir serão detalhados os cenários de teste F1, F2 e F3 do grupo F.

Cenário F1: Execução de *Workload* DW x OLTP - Sem Restrição

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads*, um do tipo DW e outro do tipo OLTP, sem restrição de recursos de disco, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico F1 apresentado anteriormente na figura 6.4, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo DW e OLTP. O tempo de execução foi de 1165 segundos.

Cenário F2: Execução de *Workload* DW x OLTP Com Restrição Estática

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads*, um do tipo DW e outro do tipo OLTP, com restrição de 50% de recursos de disco para cada contêiner, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico F2 apresentado anteriormente na figura 6.4, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo DW e OLTP. O tempo de execução foi de 980 segundos.

Cenário F3: Execução de *Workload* DW x OLTP Com Restrição Dinâmica

Este cenário de teste foi realizado para demonstrar o comportamento de dois *workloads*, um do tipo DW e outro do tipo OLTP, com restrição dinâmica através da utilização da política DDPM para os recursos de disco para cada contêiner, os quais sofrem concorrência, pois cada um está em uma máquina virtual em um ambiente compartilhado para avaliar o seu comportamento, bem como seu desempenho. O gráfico F3 apresentado anteriormente na figura 6.4, demonstra a execução de múltiplas máquinas virtuais com o *workload* do tipo DW e OLTP. O tempo de execução foi de 950 segundos.

6.4 Considerações da Aplicação de uma Política Dinâmica

Como observado nos testes realizados na seção 6.3 para os cenários com o ajuste dinâmico de recursos de disco, a política apresentou ganhos de performance em comparação ao ajuste estático. Isto pode ser observado através da tabela 6.2 que apresenta os tempos de execução e o *speedup* em cada experimento:

Tabela 6.2 – Tempos em segundos de execução para os cenários de teste

Cenário	Restrição	Workload	Duração (s)	Desvio Padrão	Speedup
D1	-	OLTP x OLTP	1165	2,70	1
D2	Estática	OLTP x OLTP	940	1,86	1,23
D3	Dinâmica	OLTP x OLTP	895	1,54	1,30
E1	-	DW x DW	855	2,13	1
E2	Estática	DW x DW	740	1,92	1,15
E3	Dinâmica	DW x DW	675	1,17	1,26
F1	-	DW x OLTP	1165	2,99	1
F2	Estática	DW x OLTP	980	2,55	1,18
F3	Dinâmica	DW x OLTP	950	2,79	1,22

Todos os testes realizados com a política (cenário de teste D3, E3 e F3) apresentaram ganhos, sendo possível responder a questão de pesquisa Q2 - Qual o impacto de uma política de ajuste dinâmico de recursos de disco em relação a um cenário com múltiplos bancos de dados em execução em um ambiente virtualizado, levando em consideração o desempenho destes ambientes e o tempo de execução? Durante os testes, uma política dinâmica de ajuste de recursos de disco pode ser aplicada a *workloads* concorrentes do mesmo tipo, DW ou OLTP. Nos *workloads* do tipo DW que envolvem apenas características de leitura, obteve-se redução na contenção de disco e melhorando a performance de leitura. Nos *workloads* do tipo OLTP, que envolvem características de leitura e escrita, observou-se a contenção de disco, aplicando a restrição maior na escrita e consequentemente na sua leitura, visto que este tipo de *workload* exige mais escrita do que leitura.

A tabela 6.2 apresenta os cenários de testes realizados com execuções de duas máquinas virtuais sem aplicar restrição de recursos de disco, a fim de avaliar o comportamento sem nenhum ajuste de E/S para os cenários D1, E1 e F1. Após isso, foram reali-

zados testes com duas máquinas virtuais aplicando a restrição de forma estática, ou seja, sem alterações durante toda a execução dos *workloads* avaliados, seguindo as restrições de recursos definidas respectivamente 50% x 50% nos cenários D2, E2 e F2. Estes percentuais estão relacionados a taxa máxima de operações de leitura e escrita que o disco pode alcançar, sendo aplicados respectivamente para 50% x 50% - 50% para o contêiner 1 e 50% para o contêiner 2. E por fim, foram realizados os testes com ajuste dinâmico de recursos de disco, aplicando a política DDPM nos cenários de teste D3, E3 e F3. Em todos os cenários realizados foram aplicadas 10 execuções em cada cenário, com o objetivo de identificar a duração, desvio padrão e, por fim o *speedup* para mostrar os ganhos e perdas obtidos nos cenários com ajuste de restrição de disco.

Para os cenários apresentados na tabela 6.2 acima, ambos os tipos de *workloads*, DW e OLTP, apresentaram ganhos no tempo de execução para a restrição com ajuste estático e com ajuste dinâmico, isso se comparando ao cenário sem ajuste de restrição. Isso ocorreu devido ao melhor aproveitando da contenção de disco. Entretanto, o ajuste dinâmico se destacou (cenários D3 e E3) em relação ao ajuste estático, pois o mesmo distribui a alocação conforme o TPS utilizado no banco de dados, dando mais recursos para quem está necessitando, evitando assim ociosidade de recursos enquanto houver contêineres em processamento. Isso pode ser comprovado pelo *speedup* observado nos cenários de teste, onde todos os cenários apresentaram ganhos de performance tanto com o ajuste de recursos de disco estático quanto o dinâmico. Entretanto, o cenário dinâmico apresentou melhores resultados do que o estático devido a melhor utilização dos recursos ociosos durante todo o ciclo de execução. Segue abaixo uma análise mais detalhada sobre os resultados obtidos com ajuste estático e dinâmico:

a) Comparativo dos experimentos D2 e D3 - com *workload* OLTP

Como pode ser observado no gráfico 6.2 apresentado anteriormente, o cenário de teste D2 (ajuste estático) apresentou um pico de execução no momento zero do ciclo de execução do contêiner 1 e logo após o segundo contêiner iniciar sua execução, ambos se mantiveram com o ajuste estático definido de 50% para ambas as operações de leitura e escrita, sem nenhuma priorização de recursos, acabando sua execução no *snapshot* 190. No gráfico relacionado as operações de leitura, pode-se identificar que a taxa de Mbps para as operações de leitura estavam em torno de 40 Mbps (descartando os picos de execução de ambos contêineres acima deste limite) e de escrita em torno de 50 Mbps (descartando os picos de execução de ambos contêineres acima deste limite) para cada contêiner, havendo ainda ociosidade de recursos, uma vez que neste experimento existe maior atividade de escrita do que leitura, havendo uma oportunidade de priorizar esta atividade e acelerar a execução destes contêineres. Pode ser observado no cenário de teste D3 com a utilização do ajuste dinâmico que a política priorizou o contêiner 1 em relação ao contêiner 2, alocando mais recursos para o mesmo. A partir disso, o contêiner 1 teve sua execução acelerada e

finalizou no *snapshot* 140, após isso, como o contêner 2 era o único em execução, a política, alocou todos os recursos de disco para ele e o mesmo finalizou no *snapshot* 180. Pode-se observar que o experimento D3 finalizou mais rápido que D2, apresentando um melhor tempo de execução e *speedup*. Os resultados obtidos nos experimentos D2 e D3 foram os seguintes: i) *Speedup*: D2 - 1,23 e D3 - 1,30; ii) Ganho comparado a execução sem qualquer ajuste (estático ou dinâmico) aplicado: D2 - 19% e D3 - 23%.

b) Comparativo dos experimentos E2 e E3 - com *workload* DW

Como pode ser observado no gráfico 6.3 apresentado anteriormente, o cenário de teste E2 (ajuste estático) apresentou dois contêineres em execução concorrente com *workloads* DW. Neste experimento, nota-se a operação de apenas leitura, uma vez que foi executado um relatório de BI com uma carga massiva de dados para que o banco de dados realize sua leitura e execução do mesmo. Como pode ser observado ao longo do gráfico, houve um pico de execução acima da taxa de 100 Mbps para o contêiner 1 e no momento que o contêiner 2 inicia sua execução, logo após, o ajuste estático de 50% dos recursos de disco, começa a ser aplicado, dividindo os recursos disponíveis entre os contêineres, sendo os mesmos finalizados no *snapshot* 150. Durante todo o ciclo de execução o contêiner 1 tem picos de execução acima do contêiner 2, demonstrando que o mesmo tem um maior processamento a ser executado, necessitando mais recursos de disco, mas devido a política de ajuste estático, a mesma reduz sua alocação e consequentemente impactando o tempo total de execução de ambos os contêineres, uma vez que poderia ser priorizado o contêiner com maior execução a ser processado. Já o cenário E3 com o ajuste dinâmico, pode-se notar que o contêiner teve uma maior alocação de recursos a partir do *snapshot* 2 e finalizando sua execução do *snapshot* 115. Logo após, a política identificando apenas o contêiner 2 em execução, aloca todos os recursos para a operação de leitura no *snapshot* 115 e o mesmo é finalizado em torno do *snapshot* 136. Neste exemplo com apenas uma operação de E/S em execução (leitura), o ajuste dinâmico permitiu a execução acelerada dos contêineres em execução. Os resultados obtidos nos experimentos E2 e E3 foram os seguintes: i) *Speedup*: E2 - 1,15 e E3 - 1,26; ii) Ganho comparado a execução sem qualquer ajuste (estático ou dinâmico) aplicado: E2 - 13% e E3 - 21%.

c) Comparativo dos experimentos F2 e F3 - com *workload* DW e OLTP

Como pode ser observado no gráfico 6.4 apresentado anteriormente, o cenário de teste F2 apresenta a execução concorrente de dois contêineres, sendo o contêiner 1 com um *workload* DW e o contêiner 2 com um *workload* OLTP. Neste caso, pode-se avaliar concorrência de tipos diferentes de *workload* e operações de E/S: apenas leitura e leitura e escrita respectivamente. Com o ajuste estático aplicado neste experimento de 50% dos recursos de disco, ambas as operações de E/S para ambos os contêineres foi aplicado

este limite. Por se tratar de uma execução de tipos de *workload* diferentes, existe muita concorrência por ambos os recursos de E/S, não havendo muita ociosidade de recursos, uma vez que ambas as operações de leitura e escrita estão sendo altamente requisitadas por ambos os contêineres, sendo o contêiner 1 finalizando em torno do *snapshot* 185 e o segundo no *snapshot* 195. No cenário de teste F3 com o ajuste dinâmico, o gráfico de leitura teve uma distribuição em torno de 60 Mbps para ambos os contêineres do *snapshot* 1 ao 180, pois a política identificou que ambos necessitariam de recursos para as operações de leitura. Entretanto, como o contêiner 2 também tinha operações de escrita, o mesmo teve uma alocação maior de recursos, mas não utilizou 100% dos recursos, uma vez que eles dependiam das operações de leitura - o contêiner 2 estava com a execução das operações de leitura e escrita em paralelo, então, não se obteve um maior ganho, como observado nos experimentos anteriores com apenas um tipo de *workload* de banco de dados. Neste caso, ocorreu aceleração nas aplicações, pois o contêiner 1 e 2 finalizaram antes que o esperado, em comparação ao cenário de teste F2. Os resultados obtidos nos experimentos F2 e F3 foram os seguintes: i) *Speedup*: F2 - 1,18 e F3 - 1,22; ii) Ganho comparado a execução sem qualquer ajuste (estático ou dinâmico) aplicado: F2 - 16% e F3 - 18%.

Conforme descrito nos comparativos a, b e c, um ajuste estático de recursos pode gerar ganhos de performance, dependendo da configuração de restrição de recursos adotados em cada contêiner, mas possui uma desvantagem que estas restrições não se adaptam ao longo do ciclo de execução, conforme a demanda por recursos pelos processos dos bancos de dados do ambiente, gerando ociosidade e mau uso dos recursos de disco. Por outro lado, a política proposta tem a característica de se adaptar dinamicamente e baseado na demanda por recursos, minimizando a contenção de disco devido a um melhor aproveitamento dos recursos de disco. Entretanto, como observado o *speedup* entre os cenários do grupo D, E e F; a execução de *workloads* de mesmo tipo (DW ou OLTP) tiveram um *speedup* melhor, se comparado com o resultado do grupo F com *workloads* diferentes em execução concorrente. Isso se deve com a execução de *workloads* de mesmo tipo, havendo maior concorrência em uma mesma operação de E/S (leitura ou escrita), podendo gerar maiores oportunidades para a aplicação da política. Neste caso, pode-se observar que a política teve melhor performance com contêineres com transações de banco de dados com operações de E/S de mesmo tipo (leitura ou escrita).

Com isso, pode-se avaliar a hipótese de pesquisa H2 - A aplicação de uma única política dinâmica de priorização dos recursos de disco pode apresentar ganhos de performance para qualquer combinação dos *workloads* de banco de dados estudados. Baseado nos resultados obtidos, esta hipótese foi comprovada. Pode-se observar um melhor uso dos recursos de disco, bem como ganhos de performance, reduzindo consequentemente a contenção de disco conforme proposto por este trabalho.

7. CONCLUSÃO

O presente trabalho como já apresentado nos capítulos anteriores, tem como objetivo a otimização dos recursos de disco para ambientes de banco de dados tradicionais, executados em ambientes virtualizados consolidados. Em um primeiro momento foram realizados alguns testes, como forma de avaliar a viabilidade, onde constatou-se que com um ajuste estático dos recursos de disco obteve-se o comportamento esperado de contenção de disco, e por sua vez uma melhora na performance dos bancos de dados para o contêiner priorizado.

A partir disso, foi possível responder a questão de pesquisa Q1 - Qual o impacto do ajuste estático de recursos de disco proposto pelo trabalho de Matteussi *et al.* [19], levando em consideração o *workload* de banco de dados? Sim. Neste caso, pode-se observar ganhos no ajuste estático de recursos de disco, assim como no trabalho realizado por Kassiano, mas com o *workload* de banco de dados. Além disso, também pode ser avaliado a hipótese de pesquisa H1 - Os ganhos obtidos pelo estudo realizado por Matteussi *et al.* [19] em um *workload* do tipo *Hadoop*, aplicam-se também em um *workload* de banco de dados. Baseado na resposta a questão de pesquisa Q1, esta hipótese foi comprovada, uma vez que também se observou ganhos de performance em aplicações de banco de dados, assim como observado nos experimentos realizados por Matteussi *et al.* [19] em *Hadoop*. Possibilitando cumprir o primeiro objetivo esperado, que era aplicar os testes de ajuste estático de recursos de disco em um cenário de múltiplos contêineres com bancos de dados em execução.

Após isso, foram realizados testes com múltiplos contêineres aplicando a política dinâmica proposta com o objetivo de avaliar o desempenho da mesma. Como apresentado no capítulo 6, os testes realizados na seção 6.3 para os cenários com o ajuste dinâmico de recursos de disco, a política funcionou de forma satisfatória, sendo possível responder a questão de pesquisa Q2 - Qual o impacto de uma política de ajuste dinâmico de recursos de disco em relação a um cenário com múltiplos bancos de dados em execução em um ambiente virtualizado, levando em consideração o desempenho destes ambientes e o tempo de execução? Durante os testes, uma política dinâmica de ajuste de recursos de disco pode ser aplicada a *workloads* concorrentes do mesmo tipo, DW ou OLTP. Com isso, pode-se avaliar a hipótese de pesquisa H2 - A aplicação de uma única política dinâmica de priorização dos recursos de disco pode ser aplicada em um *workload* de banco de dados do tipo DW e OLTP. Baseado nos resultados obtidos, esta hipótese foi comprovada. Os ganhos obtidos através da política dinâmica em comparação a um cenário sem qualquer ajuste nos recursos de disco foram de: i) Testes com contêineres com mesmo *workload* de banco de dados: OLTP - 23% e DW - 21%; ii) Testes com contêineres com tipos diferentes

de *workload* de banco de dados foi de 18% para DW e OLTP. Baseado nisso, pode-se observar um melhor uso dos recursos de disco, bem como ganhos de performance, reduzindo consequentemente a contenção de disco conforme proposto por este trabalho. Além disso, foi utilizada uma carga de dados real, vinda de um ambiente de performance de uma empresa TI de grande porte, demonstrando que a proposta funciona com dados reais de um ambiente tradicional de banco de dados.

Diversos desafios foram superados para que este trabalho fosse finalizado, como por exemplo, a forma mais efetiva de mensurar o uso da atividade de E/S dos diferentes contêineres. A maioria das ferramentas de monitoração de disco, não oferecem uma visão clara e agrupada por contêiner, e por sua vez, organizada por usuário e processo. Após vários testes e medições, constatou-se que a ferramenta IOTOP era a mais adequada para cumprir este fim. Entretanto, foram necessários algumas adequações para possibilitar a distinção de cada contêiner. Esta adequação consiste em utilizar um ID de usuário diferente para cada contêiner, onde foi alterado para um padrão conhecido, como por exemplo, 601 e 602. O primeiro dígito está relacionado a conta do usuário de banco de dados Oracle e o último dígito corresponde ao contêiner. Desta forma, alterou-se o usuário que seria dono dos binários do banco de dados para estes IDs predefinidos, para visualizar os processos conforme estes códigos, e por fim, possibilitando a identificação de cada contêiner. Sem este processo, não seria possível mensurar as execuções por contêiner.

Finalmente, esta solução poderia ser aplicada em um cenário corporativo utilizando a plataforma de virtualização LXC como utilizado neste trabalho para possibilitar a flexibilidade no ajuste de recursos de disco sem a necessidade de indisponibilidade nas máquinas virtuais. Este trabalho utilizou o LXC (Linux Contêiner) como plataforma de virtualização, mas também poderia ter sido utilizado VMWare, uma vez que o mesmo é bastante utilizado no meio corporativo. Além disso, uma vez definido a tecnologia de banco de dados a ser utilizada como Oracle, MySQL ou Postgres, bastaria adaptar o *driver* de conexão nos scripts utilizadas pela política e avaliar os desafios descritos acima para que seja aplicado em um ambiente corporativo.

A seguir serão apresentadas as contribuições deste trabalho, bem como os possíveis trabalhos futuros.

7.1 Contribuições

As principais contribuições deste trabalho são:

- Implementação de um algoritmo para utilização de uma política de ajuste dinâmico de recursos: Foi modelado e construído um protótipo para utilização desta política, a fim

de demonstrar sua utilização em vários cenários de teste para verificar a contenção de disco e benefícios em ganhos de performance;

- Avaliação da arquitetura da política dinâmica proposta: A política foi avaliada, levando em consideração o desempenho das aplicações de banco de dados, levando em consideração dados simulados através de uma ferramenta de *benchmarking* e dados reais capturados de uma empresa de TI de grande porte;

- A hipótese H1 que afirma que os ganhos obtidos pelo estudo realizado por Matteussi *et al.* [19] em um *workload* do tipo *Hadoop*, aplicam-se também em um *workload* de banco de dados. A hipótese foi comprovada;

- A hipótese H2 que afirma que a aplicação de uma única política dinâmica de priorização dos recursos de disco pode obter ganhos de performance para qualquer combinação de *workloads* de banco de dados estudados. A hipótese foi comprovada;

- Esse trabalho demonstrou através de um estudo experimental que ao utilizar uma política dinâmica de alocação de recursos de disco é possível obter ganho de desempenho, e ao mesmo tempo diminuir a contenção de disco. Além disso, oferecendo uma solução transparente, uma vez que a proposta diferencia-se das existentes, pois a solução não depende de alterações em *hardware/software*, *frameworks* ou nas aplicações. Adicionalmente, esta proposta tem independência de banco de dados, apesar deste trabalho ter utilizado o banco de dados Oracle, pode-se aplicar o mesmo em MySQL e PostgreSQL sem alterações na solução proposta. Uma vez que não há utilização de objetos ou recursos específicos do banco de dados, pois o controle das restrições, são realizadas a nível de sistema operacional.

7.2 Trabalhos Futuros

Considerando os benefícios proporcionados neste trabalho, os seguintes tópicos podem ser avaliados para possíveis trabalhos futuros:

- Este trabalho teve como objetivo a restrição dos recursos apenas de disco, entretanto, pode-se avaliar novas propostas para a aplicação de uma política dinâmica de restrição de recursos que incluíssem memória e CPU, uma vez, que isso possa contribuir ainda mais no desempenho das transações de banco de dados;

- Com o advento de discos mais rápidos, como os SSDs (*Solid-State Drives*), surge uma oportunidade de avaliar o impacto de performance com a utilização destes discos, e se uma política dinâmica de restrição de recursos seria factível para qualquer aplicação de banco de dados;

- Este trabalho teve como foco os bancos de dados transacionais, surgindo a oportunidade de avaliar os bancos de dados não transacionais, ou noSQL;
- Efetuar uma extensão do trabalho para lidar com uma abordagem de acordo de níveis de serviço (SLA);
- Por fim, é possível avaliar um método de predição de restrição de disco com base em um histórico de utilização. Esta poderia ser uma sugestão para um nova política, a fim de monitorar um ambiente de banco de dados por um período de tempo para então, avaliar predições de restrições, a fim de, aprimorar a política de restrições atualmente proposta.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Aboulnaga, A.; Amza, C.; Salem, K. “Virtualization and databases: State of the art and research challenges”. In: 11th International Conference on Extending Database Technology: Advances in Database Technology, 2008, pp. 746–747.

[2] Anzanello, C. A. “Olap conceitos e utilização”, Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, UFRGS, 2005, 104p.

[3] Beernaert, L.; Matos, M.; Vilaça, R.; Oliveira, R. “Automatic elasticity in openstack”. In: Secure and Dependable Middleware for Cloud Monitoring and Management, 2012, pp. 1–6.

[4] Blagodurov, S.; Fedorova, A. "Optimizing Shared Resource Contention in HPC Clusters". Capturado em: http://sc13.supercomputing.org/sites/default/files/PostersArchive/tech_posters/post122s2-file2.pdf. Novembro 2016.

[5] Coutinho, E.; Sousa, F. R.; Gomes, D. G.; Souza, J. “Elasticidade em computaçao na nuvem: Uma abordagem sistemática”. In: XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC 2013)-Minicursos, 2013, pp. 1–44.

[6] Date, C. J. “Introdução a sistemas de bancos de dados”. Rio de Janeiro: Elsevier Editora Ltda, 2004, 803p.

[7] Dawoud, W.; Takouna, I.; Meinel, C. “Elastic virtual machine for fine-grained cloud resource provisioning”. In: Global Trends in Computing and Communication Systems: 4th International Conference, 2012, pp. 11–25.

[8] Dias, C. A. “Descoberta de conhecimento em banco de dados para apoio a tomada de decisão”, Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, UNESP, 2006, 100p.

[9] Elmasri, R.; Navathe, S. B.; Pinheiro, M. G.; Canhette, C. C.; Melo, G. C. V.; Amadeu, C. V.; de Oliveira Morais, R. “Sistemas de banco de dados”. Rio de Janeiro: Elsevier Editora Ltda, 2005, 513p.

[10] Galante, G.; de Bona, L. C. E. “A survey on cloud computing elasticity”. In: Fifth IEEE Conference on Utility and Cloud Computing (UCC), 2012, pp. 263–270.

[11] Giri, M. "Effective Resource Management Using Oracle Database Resource Manager". Capturado em: http://www.oracle.com/technetwork/articles/servers-storage-admin/o11-056-oracledb-rm-419380.pdf. Junho 2016.

[12] Goldberg, R. P. “Survey of virtual machine research”. In: Computer, vol. 7, no. 6, IEEE, 1974, pp. 34–45.

[13] Hagmann, R. B. “An observation on database buffering performance metrics.” In: 12th International Conference on Very Large Data Bases, 1986, pp. 289–293.

[14] HORTONWORKS. "Virtualização com Hadoop". Capturado em: http://br.hortonworks.com/partners/virtualization/. Outubro 2016.

[15] Ivanov, T.; Petrov, I.; Buchmann, A. “A survey on database performance in virtualized cloud environments”. In: International Journal of Data Warehousing and Mining (IJDWM), 2012, pp. 1–26.

[16] 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, 110p.

[17] Krebs, R.; Momm, C.; Kounev, S. "Metrics and techniques for quantifying performance isolation in cloud environments". Capturado em: https://sdqweb.ipd.kit.edu/publications/pdfs/KrMoKo2012-QoSA-QuantifyingPerfIsoMetrics.pdf. Novembro 2016.

[18] KVM. "Main Page". Capturado em: http://www.linux-kvm.org. Outubro 2016.

[19] Matteussi, K. J. “Minimizando a contenção de disco em contêineres como estratégia para acelerar aplicações map reduce”, Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2016, 94p.

[20] Mell, P.; Grance, T. "The NIST definition of cloud computing". Capturado em: https://www.nist.gov/sites/default/files/documents/itl/cloud/cloud-def-v15.pdf. Setembro 2016.

[21] Mergen, M. F.; Uhlig, V.; Krieger, O.; Xenidis, J. "Virtualization for High-performance Computing". Capturado em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.3214&rep=rep1&type=pdf. Setembro 2016.

[22] Mozafari, B.; Curino, C.; Madden, S. “Dbseer: Resource and performance prediction for building a next generation database cloud”. In: Biennial Conference on Innovative Data Systems Research (CIDR), 2013, pp. 10–95.

[23] Oliveira, I. C. “Aprimorando a elasticidade de aplicações de banco de dados utilizando virtualização em nível de sistema operacional”, Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2015, 89p.

[24] ORACLE. "Oracle Database Performance Tuning Guide - I/O Configuration and Design". Capturado em: https://docs.oracle.com/cd/E18283_01/server.112/e16638/iodesign.htm. Junho 2016.

[25] ORACLE. "Real Application Testing". Capturado em: https://www.oracle.com/database/real-application-testing/index.html. Junho 2016.

[26] ORACLE. "Transaction Per Second". Capturado em: https://docs.oracle.com/cd/B16240_01/doc/doc.102/e16282/oracle_database_help/oracle_database_instance_throughput_transactions_ps.html. Maio 2016.

[27] ORACLE. "Transaction Throughput". Capturado em: http://docs.oracle.com/cd/E17076_02/html/programmer_reference/transapp_throughput.html. Maio 2016.

[28] Raveendran, A.; Bicer, T.; Agrawal, G. “A framework for elastic execution of existing mpi programs”. In: IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011, pp. 940–947.

[29] REDHAT. "Using Control Groups". Capturado em: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch-Using_Control_Groups.html. Maio 2016.

[30] Soltesz, S.; Pötzl, H.; Fiuczynski, M. E.; Bavier, A.; Peterson, L. “Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors”. In: 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, 2007, pp. 275–287.

[31] Sonehara, N.; Echizen, I.; Wohlgemuth, S. “Isolation in cloud computing and privacy-enhancing technologies”. In: Business & Information Systems Engineering Conference, 2011, pp. 155–165.

[32] Srivastava, J.; Chen, P.-Y. “Warehouse creation-a potential roadblock to data warehousing”. In: IEEE Transactions on Knowledge and Data Engineering Conference, 1999, pp. 118–126.

[33] Tianfield, H. “Cloud computing architectures”. In: International Conference on Systems, Man and Cybernetics (SMC), 2011, pp. 1394–1399.

[34] VMWARE. "VMware Virtualization for Desktop Server, Application, Public & Hybrid Clouds". Capturado em: http://www.vmware.com. Abril 2016.

[35] Weitzen, H. S. “O poder da informação: como transformar a informação que você domina em um negócio lucrativo”. Rio de Janeiro: Makron Books/McGraw-Hill, 1991, 215p.

[36] Xavier, M. G.; De Oliveira, I.; Rossi, F.; Dos Passos, R.; Matteussi, K. J.; De Rose, C. A. F. “A performance isolation analysis of disk-intensive workloads on container-based clouds”. In: 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2015, pp. 253–260.

[37] Xavier, M. G.; Neves, M. V.; De Rose, C. A. F. “A performance comparison of container-based virtualization systems for mapreduce clusters”. In: 22nd Euromicro International

Conference on Parallel, Distributed, and Network-Based Processing, 2014, pp. 299–306.

[38] Xavier, M. G.; Neves, M. V.; Rossi, F.; Ferreto, T.; Lange, T.; De Rose, C. A. F. “Performance evaluation of container-based virtualization for high performance computing environments”. In: 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2013, pp. 233–240.

[39] XEN. "Xen Project". Capturado em: http://xenserver.org/. Junho 2016.

[40] Zhu, X.; Wang, Z.; Singhal, S. “Utility-driven workload management using nested control design”. In: American Control Conference, 2006, pp. 6–7.

APÊNDICE A – CÓDIGO FONTE: DDPM.SH

```
1  ########################################
2  # DDPM
3  ########################################
4  # Parametros:
5  # start - Inicia o framework de monitoracao
6  # stop - Finaliza o framework de monitoracao
7  ########################################
8
9  # Variabeis de ambiente
10 CONFIG_PATH=/root/DDPM
11 CHECK_MONITOR_PERFORMANCE=0
12 CHECK_MONITOR_RECURSOS=0
13 CHECK_GERENCIADOR_RECURSOS=0
14 GERENCIADOR_RECURSOS_ENC_FILE=$CONFIG_PATH/lock/gerenciador_recursos_encerrar.
       txt
15
16 # ----------------------------------------------
17 # Sinaliza que o DDPM vai iniciar uma execucao
18 echo "==== DDPM - `date +"%d-%m-%y %T"`: Inicio"
19 echo "1" > $CONFIG_PATH/lock/DDPM_lock.txt
20 # ----------------------------------------------
21
22 # Verifica se foi especificado o parametro de incializacao do framework: start/
       stop
23 echo "-- Parametro de Inicializacao especificado: gerenciador_recursos.sh $1"
24 if [ "$1" = "start" ]
25 then
26    CHECK_MONITOR_PERFORMANCE=`ps -ef | grep monitor_performance.sh | grep -v
       grep | wc -l`
27    CHECK_MONITOR_RECURSOS=`ps -ef | grep monitor_recursos.sh | grep -v grep | wc
       -l`
28    CHECK_GERENCIADOR_RECURSOS=`ps -ef | grep gerenciador_recursos.sh | grep -v
       grep | wc -l`
29
30    if [ $CHECK_MONITOR_PERFORMANCE -eq 1 ] || [ $CHECK_MONITOR_RECURSOS -eq 1 ]
       || [ $CHECK_GERENCIADOR_RECURSOS -eq 1 ]
31    then
32       echo "-- Erro: Scripts ja estao em execucao."
33       exit
34    else
35       echo "-- Iniciando gerenciador de recursos..."
36       echo "-- Log: $CONFIG_PATH/logs/gerenciador_recursos.log"
37       nohup $CONFIG_PATH/gerenciador_recursos.sh > $CONFIG_PATH/logs/
       gerenciador_recursos.log &
38    fi
```

```
39 else if [ "$1" = "stop" ]
40      then
41         # Sinaliza para o gerenciador de recursos ser encerrado
42         echo "-- Sinaliza para o gerenciador de recursos ser encerrado"
43         echo "-- Log: $CONFIG_PATH/logs/gerenciador_recursos.log"
44         echo "1" > $GERENCIADOR_RECURSOS_ENC_FILE
45      else
46        echo "-- Erro: Parametro nao especificado. Deve ser utilizado:
    gerenciador_recursos.sh [start/stop]"
47         exit
48      fi
49 fi
50
51 # ------------------------------------------------
52 # Sinaliza que o DDPM finalizou uma execucao
53 echo "==== DDPM - `date +"%d-%m-%y %T"`: Fim"
54 echo "0" > $CONFIG_PATH/lock/DDPM_lock.txt
55 # ------------------------------------------------
```

- code/DDPM.sh

APÊNDICE B – CÓDIGO FONTE: MONITOR_PERFORMANCE.SH

```
1  ########################################
2  # Monitor de Performance
3  ########################################
4  # Requisitos:
5  # Deve-se criar uma view nos bancos de dados para facilitar a captura da
       informacao sobre TPS
6  # create or replace view V_DATABASE_TPS as select t1.value "COMMITS",t2.value "
       ROLLBACKS",
7  #                                            t1.value+t2.value "CURRENT_TPS
       ",t3.qty "QTY_ACTIVE_SESSIONS",
8  #                                            decode(t3.qty,0,'INACTIVE','
       ACTIVE') "DB_STATUS"
9  # from (select value from v$sysstat where name='user commits') t1,
10 #      (select value from v$sysstat where name='user rollbacks') t2,
11 #      (select count(1) "QTY" from gv$session where username is not null and sid
        <> sys_context('USERENV','SID') and status='ACTIVE') t3;
12 # grant select on v_database_tps to system;
13 ########################################
14
15 # Variabeis de ambiente
16 LXC_CONTAINER=""
17 LXC_CONTAINER_ALIVE=""
18 BLKIO_READ_VALUE=""
19 BLKIO_WRITE_VALUE=""
20 LXC_DB_ALIVE=0
21 LXC_DB_TNS=""
22 CONFIG_PATH=/root/DDPM
23 MONITOR_RECURSOS_LOCK=0
24 MONITOR_RECURSOS_FILE=$CONFIG_PATH/repository/monitor_recursos_status.txt
25 TPS1=0
26 TPS2=0
27 TPS_DIF=0
28 DB_USERID=system
29 DB_PASS=manager
30 DB_STATUS=""
31
32 export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib/${LD_LIBRARY_PATH:+:
       $LD_LIBRARY_PATH}
33 export ORACLE_HOME=/usr/lib/oracle/12.1/client64
34 export PATH=$ORACLE_HOME/bin:$PATH
35 export TNS_ADMIN=/root
36 export ORACLE_BASE=/u01/app/oracle
37 export ORACLE_LIBRARY=$ORACLE_HOME/lib/libclntsh.so
38
39 # ----------------------------------------------
```

```
40 # Sinaliza que o monitor de performance vai iniciar uma execucao
41 echo "============================================================"
42 echo "==== Monitor de Performance - `date +"%d-%m-%y %T"`: Inicio"
43 echo "============================================================"
44 echo "1" > $CONFIG_PATH/lock/monitor_performance_lock.txt
45 # ------------------------------------------
46
47 # Verifica se o monitor de recursos nao esta em execucao
48 MONITOR_RECURSOS_LOCK=`cat $CONFIG_PATH/lock/monitor_recursos_lock.txt`
49
50 # Se o monitor de recursos nao esta em execucao, sera criado uma copia do status
       atual dos recursos para garantir uma execucao com sucesso do monitor de
      performance
51 if [ $MONITOR_RECURSOS_LOCK -eq 0 ]
52 then
53    echo "-- Criar arquivo temporario do monitor_recursos_stats.txt"
54    cp $CONFIG_PATH/repository/monitor_recursos_status.txt $CONFIG_PATH/tmp/
       monitor_recursos_status.tmp.txt
55    MONITOR_RECURSOS_FILE=$CONFIG_PATH/tmp/monitor_recursos_status.tmp.txt
56 else
57    echo "-- Monitor de Recursos em execucao, aguardando liberacao de lock..."
58    while (`cat $CONFIG_PATH/lock/monitor_recursos_lock.txt` -eq 1)
59    do
60     sleep 1
61    done
62    echo "-- Criar arquivo temporario do monitor_recursos_stats.txt"
63    cp $CONFIG_PATH/repository/monitor_recursos_status.txt $CONFIG_PATH/tmp/
       monitor_recursos_status.tmp.txt
64    MONITOR_RECURSOS_FILE=$CONFIG_PATH/tmp/monitor_recursos_status.tmp.txt
65 fi
66
67 # Gerar lista do status de performance dos banco de dados
68 echo "-- Gerar lista do status de performance"
69 cat /dev/null > $CONFIG_PATH/repository/monitor_performance_status.txt
70
71 # Le o arquivo de status do monitor de recursos ($CONFIG_PATH/repository/
      monitor_recursos_status.txt)
72 while IFS= read -r var
73 do
74   LXC_CONTAINER=`echo $var | cut -d: -f1`
75   LXC_CONTAINER_ALIVE=`echo $var | cut -d: -f2`
76   LXC_DB_TNS="`echo $var | cut -d: -f3`"
77   LXC_DB_ALIVE=`echo $var | cut -d: -f4`
78   BLKIO_READ_VALUE="`echo $var | cut -d: -f5`"
79   BLKIO_WRITE_VALUE="`echo $var | cut -d: -f6`"
80
81   echo " "
```

```
82    echo "-- LXC_CONTAINER=$LXC_CONTAINER; LXC_CONTAINER_ALIVE=
        $LXC_CONTAINER_ALIVE; LXC_DB_TNS=$LXC_DB_TNS; LXC_DB_ALIVE=$LXC_DB_ALIVE;
        BLKIO_READ_VALUE=$BLKIO_READ_VALUE; BLKIO_WRITE_VALUE=$BLKIO_WRITE_VALUE"
83    echo "-- Obter taxa TPS - $LXC_CONTAINER->$LXC_DB_TNS..."
84
85    # A partir da lista de containers, sera verificado em todos os containers com
        banco de dados ativos quanto a taxa TPS
86    if [ $LXC_CONTAINER_ALIVE -eq 1 ] && [ $LXC_DB_ALIVE -eq 1 ]
87    then
88       # Coleta o valor atual de TPS
89       TPS1=`sqlplus -s $DB_USERID/$DB_PASS@$LXC_DB_TNS <<+
90  set pagesize 0 feedback off verify off heading off echo off;
91  select current_tps from sys.v_database_tps;
92  exit;
93  +`
94       echo "-- $LXC_CONTAINER->$LXC_DB_TNS - TPS1=$TPS1"
95
96       # Gera um delay de 1 segundo para realizar uma proxima verificacao do TPS
97       sleep 1
98
99       # Realiza mais uma coleta do valor atual de TPS
100      TPS2=`sqlplus -s $DB_USERID/$DB_PASS@$LXC_DB_TNS <<+
101 set pagesize 0 feedback off verify off heading off echo off;
102 select current_tps from sys.v_database_tps;
103 exit;
104 +`
105      echo "-- $LXC_CONTAINER->$LXC_DB_TNS - TPS2=$TPS2"
106
107      # Calcula taxa TPS: TPS2-TPS1
108      TPS_DIF=$((TPS2-TPS1))
109
110      # Verifica se o banco esta ativo/inativo devido as suas sessoes ativas
111      DB_STATUS="`sqlplus -s $DB_USERID/$DB_PASS@$LXC_DB_TNS <<+
112 set pagesize 0 feedback off verify off heading off echo off;
113 select db_status from sys.v_database_tps;
114 exit;
115 +`"
116
117   else
118      # Se o container nao estiver ativo ou o banco de dados nao estiver ativo,
       sera definido um valor default de -100 para taxa TPS
119      TPS_DIF=-100
120      # Se o container ou o bancod e dados estiverem down, estara sendo setado o
       valor de INACTIVE a variavel DB_STATUS
121      DB_STATUS="INACTIVE"
122   fi
123
124   echo "-- Taxa TPS - $LXC_CONTAINER->$LXC_DB_TNS:$TPS_DIF"
```

```
125
126     echo "-- Salvar status de performance: $LXC_CONTAINER:$LXC_CONTAINER_ALIVE:
          $LXC_DB_TNS:$LXC_DB_ALIVE:$TPS_DIF:$DB_STATUS:$BLKIO_READ_VALUE:
          $BLKIO_WRITE_VALUE"
127     echo "$LXC_CONTAINER:$LXC_CONTAINER_ALIVE:$LXC_DB_TNS:$LXC_DB_ALIVE:$TPS_DIF:
          $DB_STATUS:$BLKIO_READ_VALUE:$BLKIO_WRITE_VALUE" >> $CONFIG_PATH/repository/
          monitor_performance_status.txt
128
129 done < $MONITOR_RECURSOS_FILE
130
131 # ------------------------------------------------------
132 # Sinaliza que o monitor de performance finalizou uma execucao
133 echo "============================================================"
134 echo "==== Monitor de Performance - `date +"%d-%m-%y %T"`: Fim"
135 echo "============================================================"
136 echo "0" > $CONFIG_PATH/lock/monitor_performance_lock.txt
137 # ------------------------------------------------------
```

- code/monitor_performance.sh

APÊNDICE C – CÓDIGO FONTE: MONITOR_RECURSOS.SH

```
1  #!/bin/sh
2
3  ########################################
4  # Monitor de Recursos
5  ########################################
6
7  # Variabeis de ambiente
8  LXC_CONTAINER=""
9  LXC_CONTAINER_ALIVE=""
10 BLKIO_READ_FILE=""
11 BLKIO_WRITE_FILE=""
12 BLKIO_READ_VALUE=""
13 BLKIO_WRITE_VALUE=""
14 LXC_DB_ALIVE=0
15 CONFIG_PATH=/root/DDPM
16 CONT=2
17 export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib/${LD_LIBRARY_PATH:+:
       $LD_LIBRARY_PATH}
18 export ORACLE_HOME=/usr/lib/oracle/12.1/client64
19 export PATH=$ORACLE_HOME/bin:$PATH
20 export TNS_ADMIN=/root
21 export ORACLE_BASE=/u01/app/oracle
22 export ORACLE_LIBRARY=$ORACLE_HOME/lib/libclntsh.so
23
24 # ------------------------------------------------
25 # Sinaliza que o monitor de recursos vai iniciar uma execucao
26 echo "============================================================"
27 echo "==== Monitor de Recursos - `date +"%d-%m-%y %T"`: Inicio"
28 echo "============================================================"
29 echo "1" > $CONFIG_PATH/lock/monitor_recursos_lock.txt
30 # ------------------------------------------------
31
32 # Gerar lista dos containers disponiveis
33 echo "-- Gerar lista de containers disponiveis..."
34 echo "-- Arquivo: $CONFIG_PATH/repository/lxc_servers_list.txt"
35 lxc-ls -1 > $CONFIG_PATH/repository/lxc_servers_list.txt
36 echo "-- Limpar arquivo atual de status dos containers ($CONFIG_PATH/repository/
       monitor_recursos_status.txt)..."
37 # Gerar lista do status do recursos atuais dos servidores
38 cat /dev/null > $CONFIG_PATH/repository/monitor_recursos_status.txt
39
40 # Le lista dos containers disponiveis para verificar configuracao atual dos
       CGROUPS
41 echo "-- Carregar lista de containers disponiveis..."
42 echo "-- Arquivo: $CONFIG_PATH/repository/lxc_servers_list.txt"
```

```
43 while IFS= read -r var
44 do
45   # Inicializa as variaveis do CGROUPS com o container atual
46   LXC_CONTAINER=$var
47   BLKIO_READ_FILE=/sys/fs/cgroup/blkio/lxc/$LXC_CONTAINER/blkio.throttle.
       read_bps_device
48   BLKIO_WRITE_FILE=/sys/fs/cgroup/blkio/lxc/$LXC_CONTAINER/blkio.throttle.
       write_bps_device
49   echo " "
50   echo "-- LXC: $LXC_CONTAINER"
51   echo "-- BLKIO_READ_FILE=/sys/fs/cgroup/blkio/lxc/$LXC_CONTAINER/blkio.
       throttle.read_bps_device"
52   echo "-- BLKIO_WRITE_FILE=/sys/fs/cgroup/blkio/lxc/$LXC_CONTAINER/blkio.
       throttle.write_bps_device"
53
54   # Verifica se os arquivos de configuracao do CGROUPS existem para o container
       atual
55   # Caso nao existam, sera atribuido o valor de -1 as variaveis
56   echo "-- Verifica se os arquivos BLKIO_READ_FILE e BLKIO_WRITE_FILE existem...
       "
57   if [ -f $BLKIO_READ_FILE ] && [ -f $BLKIO_WRITE_FILE ]
58   then
59           BLKIO_READ_VALUE=`cat $BLKIO_READ_FILE`
60           BLKIO_WRITE_VALUE=`cat $BLKIO_WRITE_FILE`
61
62           # Verifica se a variavel esta vazia, se sim, atribui o valor padrao de
       -1
63           if [ -z $BLKIO_READ_VALUE ]
64             then
65                 BLKIO_READ_VALUE=-1
66           fi
67
68           # Verifica se a variavel esta vazia, se sim, atribui o valor padrao de
       -1
69           if [ -z $BLKIO_WRITE_VALUE ]
70             then
71                 BLKIO_WRITE_VALUE=-1
72           fi
73   else
74      BLKIO_READ_VALUE=-1
75      BLKIO_WRITE_VALUE=-1
76   fi
77
78   echo "-- BLKIO_READ_VALUE=$BLKIO_READ_VALUE"
79   echo "-- BLKIO_WRITE_VALUE=$BLKIO_WRITE_VALUE"
80
81   # Verifica se o container atual esta up ou down
```

```
82    LXC_CONTAINER_ALIVE=`lxc-info -n $LXC_CONTAINER | grep "RUNNING" | grep -v
       grep | wc -l`
83
84    echo "-- LXC Status (0 - Down, 1 - Up): $LXC_CONTAINER_ALIVE"
85
86    # Verifica se o container esta up, verifica o status do banco, se esta down,
       nao sera necessario verificar
87    if [ $LXC_CONTAINER_ALIVE -gt 0 ]
88    then
89       LXC_DB_ALIVE=`lxc-attach -n $LXC_CONTAINER -- ps -ef | grep pmon | grep -v
       "+ASM" | grep -v grep | wc -l`
90    else
91       LXC_DB_ALIVE=0
92    fi
93
94    echo "-- LXC DB Status (0 - Down, 1 - Up): $LXC_DB_ALIVE"
95
96    # Salva estatisticas atuais do container
97    echo "-- Salvar estatisticas atuais do container: $LXC_CONTAINER:
       $LXC_CONTAINER_ALIVE:oradb`echo $((CONT))`:$LXC_DB_ALIVE:$BLKIO_READ_VALUE:
       $BLKIO_WRITE_VALUE"
98    echo "$LXC_CONTAINER:$LXC_CONTAINER_ALIVE:oradb`echo $((CONT))`:$LXC_DB_ALIVE:
       $BLKIO_READ_VALUE:$BLKIO_WRITE_VALUE" >> $CONFIG_PATH/repository/
       monitor_recursos_status.txt
99
100   CONT=$((CONT+1))
101
102 done < "$CONFIG_PATH/repository/lxc_servers_list.txt"
103
104 # ------------------------------------------------
105 # Sinaliza que o monitor de recursos finalizou uma execucao
106 echo "============================================================"
107 echo "==== Monitor de Recursos - `date +"%d-%m-%y %T"`: Fim"
108 echo "============================================================"
109 echo "0" > $CONFIG_PATH/lock/monitor_recursos_lock.txt
110 # ------------------------------------------------
```

- code/monitor_recursos.sh

APÊNDICE D – CÓDIGO FONTE: CONTROLADOR_DISCO.SH

```
1  ########################################
2  # Controlador de Disco
3  ########################################
4
5  # Variabeis de ambiente
6  BLKIO_READ_FILE=""
7  BLKIO_WRITE_FILE=""
8  BLKIO_READ_VALUE=""
9  BLKIO_WRITE_VALUE=""
10 CONFIG_PATH=/root/DDPM
11 # Variaveis de Configuracao
12 # Valores obtidos atraves um exercicio de calibration para verificar a taxa
       maxima de leitura/escrita
13 MAX_IO_READ=118
14 MAX_IO_WRITE=108
15
16 # ------------------------------------------------
17 # Sinaliza que o controlador de disco vai iniciar uma execucao
18 echo "============================================================"
19 echo "==== Controlador de Disco - `date +"%d-%m-%y %T"`: Inicio"
20 echo "============================================================"
21 echo "1" > $CONFIG_PATH/lock/controlador_disco_lock.txt
22 # ------------------------------------------------
23
24 # Recebe o valor de entrada para converter para bytes
25 echo "-- Obtem os valores de BPS para leitura/escrita e converte de MB -> Bytes"
26 BLKIO_READ_VALUE=`echo $2 | awk '{ foo=$1*1024*1024; print foo}'`
27 BLKIO_WRITE_VALUE=`echo $3 | awk '{ foo=$1*1024*1024; print foo}'`
28
29 # Recurso de disco a ser limitado
30 BLKIO_READ_FILE=/sys/fs/cgroup/blkio/lxc/$1/blkio.throttle.read_bps_device
31 BLKIO_WRITE_FILE=/sys/fs/cgroup/blkio/lxc/$1/blkio.throttle.write_bps_device
32
33   # Verifica se os arquivos de configuracao do CGROUPS existem para o container
       atual
34   # Caso nao existam, sera atribuido o valor de -1 as variaveis
35   echo "-- Verifica se os diretorios CGROUPS existem para o container: $1"
36   echo "-- BLKIO_READ_FILE=$BLKIO_READ_FILE"
37   echo "-- BLKIO_WRITE_FILE=$BLKIO_WRITE_FILE"
38   if [ -f $BLKIO_READ_FILE ] && [ -f $BLKIO_WRITE_FILE ]
39   then
40      # Se foi passado o valor 0 nos parametros para READ/WRITE, o mesmo estara
       desabilitando os limites
41      if [ $BLKIO_READ_VALUE -eq 0 ] && [ $BLKIO_WRITE_VALUE -eq 0 ]
42      then
```

```
43          echo " "
44          echo "-- Container: $1"
45          echo "-- Desabilitando limites de IO..."
46          echo "8:0 `echo $MAX_IO_READ | awk '{ foo=$1*1024*1024; print foo}'`" >
    $BLKIO_READ_FILE
47          echo "8:0 `echo $MAX_IO_WRITE | awk '{ foo=$1*1024*1024; print foo}'`" >
     $BLKIO_WRITE_FILE
48          echo " "
49        # Se os valores de READ/WRITE forem diferentes de zero, estara sendo
    definido o limite com estes valores
50        else
51          echo " "
52          echo "-- Container: $1"
53          echo "-- Limite: READ - $2 MB -> $BLKIO_READ_VALUE; WRITE - $3 MB ->
    $BLKIO_WRITE_VALUE"
54          echo "-- Gravando novos limites de leitura/escrita no CGROUPS..."
55          # Nota: O CGROUPS aceita apenas ID de um disco, particoes nao sao
    suportadas!
56          # Referencia: https://github.com/docker/docker/issues/10004
57          echo "8:0 $BLKIO_READ_VALUE" > $BLKIO_READ_FILE
58          echo "8:0 $BLKIO_WRITE_VALUE" > $BLKIO_WRITE_FILE
59          echo " "
60        fi
61      else
62         echo " "
63         echo "-- Container: $1"
64         echo "-- Erro: Nao foi encontrado os diretorios respectivos do cgroups
    para o container: $1"
65         echo "-- BLKIO_READ_FILE=$BLKIO_READ_FILE"
66         echo "-- BLKIO_WRITE_FILE=$BLKIO_WRITE_FILE"
67         echo " "
68      fi
69
70  # ------------------------------------------------------
71  # Sinaliza que o controlador de disco finalizou uma execucao
72  echo "============================================================"
73  echo "==== Controlador de Disco - `date +"%d-%m-%y %T"`: Fim"
74  echo "============================================================"
75  echo "0" > $CONFIG_PATH/lock/controlador_disco_lock.txt
76  # ------------------------------------------------------
```

- code/controlador_disco.sh

APÊNDICE E – CÓDIGO FONTE: GERENCIADOR_RECURSOS.SH

```
1  ########################################
2  # Gerenciador de Recursos
3  ########################################
4
5  # Variabeis de ambiente
6  CONFIG_PATH=/root/DDPM
7  declare -a ARRAY_LXC_DBS
8  declare -a ARRAY_LXC_ACTIVE_DBS
9  POS_ARRAY=0
10 ARRAY_LXC_DBS_SIZE=0
11 ARRAY_LXC_ACTIVE_DBS_SIZE=0
12 MONITOR_PERFORMANCE_FILE=$CONFIG_PATH/repository/monitor_performance_status.txt
13 LXC_CONTAINER=""
14 LXC_CONTAINER_STATUS=0
15 LXC_DB_STATUS=""
16 LXC_DB_TPS=""
17 LXC_DB_MAX_TPS=0
18 LXC_CONTAINER_DB_MAX_TPS=""
19 IDX_DB_READ_IO=0
20 IDX_DB_WRITE_IO=0
21 IDX_INC_DB_IO=0
22 LXC_CONTAINER_ANT=""
23 IDX_INC_DB_READ_IO=0
24 IDX_INC_DB_WRITE_IO=0
25 FLAG_ENCERRAR_SESSAO=0
26 GERENCIADOR_RECURSOS_ENC_FILE=$CONFIG_PATH/lock/gerenciador_recursos_encerrar.
       txt
27 LXCSTATUS_REPORT_FILE=$CONFIG_PATH/repository/report_lxc_status_`date +"%d%m%s-%
       h%m%s"`.csv
28 BPS_IO_READ_LIMIT=0
29 BPS_IO_WRITE_LIMIT=0
30 LXC_MAX_USAGE=0
31 QTYDBS=0
32 TOTAL_READIO=0
33 TOTAL_WRITEIO=0
34
35 # ----------------------------------------
36 # Variaveis de Configuracao
37 # Valores obtidos atraves um exercicio de calibration para verificar a taxa
       maxima de leitura/escrita
38 MAX_IO_READ=118
39 MAX_IO_WRITE=108
40 # Percentual a ser usado para reduzir do processo com menor TPS para distribuir
       dentre os demais processos
41 IDX_RED_MAX_DB_IO=10
```

```
42 # Variavel auxiliar para incremento do percentual a ser usado
43 IDX_RED_MAX_DB_IO_ADD=0
44 # ----------------------------------------
45
46 # Funcoes auxiliares
47 # Apos a execucao dos monitores, o mesmo ira ler o arquivo de configuracao para
       criacao de um array
48 carregarLista() {
49     # Inicializa a variavel com a posicao inicial do ARRAY
50     POS_ARRAY=-1
51     unset ARRAY_LXC_DBS
52
53     # Le o arquivo de status do monitor de performance ($CONFIG_PATH/repository
       /monitor_performance_status.txt)
54     while IFS= read -r var
55     do
56       POS_ARRAY=$((POS_ARRAY+1))
57       ARRAY_LXC_DBS[$POS_ARRAY]=$var
58     done < $MONITOR_PERFORMANCE_FILE
59
60     # Salva a quantidade de elementos do array ARRAY_LXC_DBS
61     ARRAY_LXC_DBS_SIZE=$((POS_ARRAY+1))
62 }
63
64 # Verificar a utilizacao de IO total dos conteineres
65 verificarUtilizacaoMaximaIO() {
66     POS_ARRAY=0
67     QTYDBS=$ARRAY_LXC_ACTIVE_DBS_SIZE
68         TOTAL_READIO=0
69         TOTAL_WRITEIO=0
70         CURREADIO=0
71         CURWRITE=0
72         LXC_MAX_USAGE=0
73
74     while [ $POS_ARRAY -lt $QTYDBS ]; do
75             CURREADIO=`echo ${ARRAY_LXC_ACTIVE_DBS[$POS_ARRAY]} | cut -d: -f7 `
76                 CURWRITEIO=`echo ${ARRAY_LXC_ACTIVE_DBS[$POS_ARRAY]} | cut -d:
       -f8 `
77                 TOTAL_READIO=$((TOTAL_READIO+CURREADIO))
78                 TOTAL_WRITEIO=$((TOTAL_READIO+CURWRITEIO))
79
80                 POS_ARRAY=$((POS_ARRAY+1))
81         done
82
83         if [ $TOTAL_READIO -ge $MAX_IO_READ ] || [ $TOTAL_WRITEIO -ge
       $MAX_IO_WRITE ]
84         then
85             LXC_MAX_USAGE=1
```

```
86            else
87                LXC_MAX_USAGE=0
88            fi
89  }
90
91  # $1 - Parametro referente ao percentual a ser utilizado no script
92  calcularRestricoesIO(){
93          # Calcula o limite de IO a ser distribuido entre os containers
94          IDX_DB_READ_IO=`awk 'BEGIN { rounded = sprintf("%.0f", $MAX_READ_IO/
        $ARRAY_LXC_ACTIVE_DBS_SIZE); print rounded }'`
95          IDX_DB_WRITE_IO=`awk 'BEGIN { rounded = sprintf("%.0f", $MAX_WRITE_IO/
        $ARRAY_LXC_ACTIVE_DBS_SIZE); print rounded }'`
96          # Calcula o indice a ser reduzido do banco de dados com maior atividade
        entre os demais
97          IDX_INC_DB_READ_IO=`awk 'BEGIN { rounded = sprintf("%.0f",
        $IDX_DB_READ_IO*($1/($ARRAY_LXC_ACTIVE_DBS_SIZE/100)); print rounded }'`
98          IDX_INC_DB_WRITE_IO=`awk 'BEGIN { rounded = sprintf("%.0f",
        $IDX_DB_WRITE_IO*($1/($ARRAY_LXC_ACTIVE_DBS_SIZE/100)); print rounded }'`
99  }
100
101 # Ira calcular os limites de IO com base nos dados obtidos da funcao
        verificarDBStatus
102 # Desempenha as seguintes atividades:
103 # - Se haver apenas um banco de dados ativo, o mesmo ficara com 100% dos
        recursos de IO
104 # - Se haver mais de um banco de dados ativo, sera calculado o limite de IO a
        ser atribuido para cada banco de dados
105 atualizarLimitesIO() {
106       POS_ARRAY=0
107
108       while [ $POS_ARRAY -lt $ARRAY_LXC_ACTIVE_DBS_SIZE ]; do
109             LXC_CONTAINER="`echo ${ARRAY_LXC_ACTIVE_DBS[$POS_ARRAY]} | cut -d: -f1
        `"
110             LXC_DB_TPS="`echo ${ARRAY_LXC_ACTIVE_DBS[$POS_ARRAY]} | cut -d: -f2`"
111
112                     verificarUtilizacaoMaximaIO
113
114             # Se haver apenas um banco ativo, o mesmo tera 100% dos recursos de IO
115                     if [ $ARRAY_LXC_ACTIVE_DBS_SIZE -eq 1 ]
116               then
117                   echo "`date +"%d/%m/%y %h:%m:%s"`;$LXC_CONTAINER;$LXC_DB_TPS;
        $MAX_IO_READ;$MAX_IO_WRITE" >> $LXCSTATUS_REPORT_FILE
118                   $CONFIG_PATH/controlador_disco.sh $LXC_CONTAINER $MAX_IO_READ
        $MAX_IO_WRITE
119                   break
120               # Caso contrario, sera calculado o valor do limite de IO
121               else if [ $ARRAY_LXC_ACTIVE_DBS_SIZE -gt 1 ]
122                     then
```

```
123                     # Define os limites de IO para os containers
124                     # Para os containers com maior TPS sera acrescentado um valor a mais - IDX_INC_DB_READ_IO e IDX_INC_DB_WRITE_IO
125                     # Para o container com menor TPS sera reduzido um valor de IDX_INC_DB_READ_IO e IDX_INC_DB_WRITE_IO
126                     if [ "$LXC_CONTAINER" == "$LXC_CONTAINER_DB_MAX_TPS" ]
127                     then
128                         if [ "$LXC_CONTAINER" != "$LXC_CONTAINER_ANT" ]
129                                         then
130                                             LXC_CONTAINER_ANT=$LXC_CONTAINER
131                                                 IDX_RED_MAX_DB_IO_ADD=0
132                                                 calcularRestricoesIO $IDX_RED_MAX_DB_IO
133                                         else
134                                 if [ $LXC_MAX_USAGE -eq 1 ]
135                                                 then
136                                                     if [ $IDX_RED_MAX_DB_IO_ADD -lt 90 ]
137                                                     then
138                                                         IDX_RED_MAX_DB_IO_ADD=$((IDX_RED_MAX_DB_IO_ADD+IDX_RED_MAX_DB_IO))
139                                                     else
140                                                         IDX_RED_MAX_DB_IO_ADD=90
141                                                     fi
142                                 else
143                                                     IDX_RED_MAX_DB_IO_ADD=0
144                                                 fi
145                                                 calcularRestricoesIO $IDX_RED_MAX_DB_IO_ADD
146                                         fi
147                                             echo "`date +"%d/%m/%y %h:%m:%s"` ;$LXC_CONTAINER;$LXC_DB_TPS;$((IDX_DB_READ_IO+IDX_INC_DB_READ_IO));$((IDX_DB_WRITE_IO+IDX_INC_DB_WRITE_IO))" >> $LXCSTATUS_REPORT_FILE
148                                 $CONFIG_PATH/controlador_disco.sh $LXC_CONTAINER_DB_MAX_TPS $((IDX_DB_READ_IO+IDX_INC_DB_READ_IO)) $((IDX_DB_WRITE_IO+IDX_INC_DB_WRITE_IO))
149                                         fi
150                     else
151                         calcularRestricoesIO $IDX_RED_MAX_DB_IO
152
153                                         echo "`date +"%d/%m/%y %h:%m:%s"` ;$LXC_CONTAINER;$LXC_DB_TPS;$((IDX_DB_READ_IO-IDX_INC_DB_READ_IO));$((IDX_DB_WRITE_IO-IDX_INC_DB_WRITE_IO))" >> $LXCSTATUS_REPORT_FILE
154                         $CONFIG_PATH/controlador_disco.sh $LXC_CONTAINER $((IDX_DB_READ_IO-IDX_INC_DB_READ_IO)) $((IDX_DB_WRITE_IO-IDX_INC_DB_WRITE_IO))
155                     fi
156                 fi
157             fi
```

```
158             POS_ARRAY=$((POS_ARRAY+1))
159         done
160 }
161
162 # Desempenha as seguintes atividades:
163 # - Desabilita os limites de IO para as LXC com bancos inativos
164 # - Recalcular os limites de IO para os bancos ativos
165 verificarDBStatus() {
166     # Identifica os bancos que estao inativos
167     POS_ARRAY=0
168
169     # Inicializa o array que ira guardar apenas a lista dos bancos ativos
170     ARRAY_LXC_ACTIVE_DBS_SIZE=0
171     unset ARRAY_ACTIVE_LXC_DBS
172     LXC_DB_MAX_TPS=0
173     LXC_CONTAINER_DB_MAX_TPS=""
174
175     # Carrega a lista de banco de dados para um array
176     carregarLista
177
178     while [ $POS_ARRAY -lt $ARRAY_LXC_DBS_SIZE ]; do
179         LXC_CONTAINER="`echo ${ARRAY_LXC_DBS[$POS_ARRAY]} | cut -d: -f1`"
180         LXC_CONTAINER_STATUS=`echo ${ARRAY_LXC_DBS[$POS_ARRAY]} | cut -d: -f2`
181         LXC_DB_TPS=`echo ${ARRAY_LXC_DBS[$POS_ARRAY]} | cut -d: -f5`
182         LXC_DB_STATUS="`echo ${ARRAY_LXC_DBS[$POS_ARRAY]} | cut -d: -f6`"
183
184         # Verifica os bancos que estao inativos ou o container esta down
185         # Se esta down, desabilita os limites de IO
186         if [ $LXC_CONTAINER_STATUS -eq 0 ] || [ $LXC_DB_STATUS = "INACTIVE" ]
187         then
188             $CONFIG_PATH/controlador_disco.sh $LXC_CONTAINER 0 0
189         fi
190
191         # Se o container esta ativo e o banco de dados tambem, realiza as seguintes atividades:
192         # - Contabiliza a quantidade de banco de dados ativos
193         # - Verifica o banco de dados que tem o maior indice de TPS
194         if [ $LXC_CONTAINER_STATUS -eq 1 ] && [ $LXC_DB_STATUS = "ACTIVE" ]
195         then
196             ARRAY_LXC_ACTIVE_DBS_SIZE=$((ARRAY_LXC_ACTIVE_DBS_SIZE+1))
197
198             ARRAY_LXC_ACTIVE_DBS[$ARRAY_LXC_ACTIVE_DBS_SIZE-1]="$LXC_CONTAINER:$LXC_DB_TPS"
199
200             if [ $LXC_DB_TPS -gt $LXC_DB_MAX_TPS ]
201             then
202                 LXC_DB_MAX_TPS=$LXC_DB_TPS
203                 LXC_CONTAINER_DB_MAX_TPS=$LXC_CONTAINER
```

```
204                                 LXC_CONTAINER_ANT=""
205                 fi
206             fi
207
208             POS_ARRAY=$((POS_ARRAY+1))
209         done
210
211         # Apos verificar o status dos bancos, sera atualizado os limites de IO
212         atualizarLimitesIO
213 }
214
215 # Funcao para desabilitar em todos os containers os limites de IO
216 desabilitarLimitesIO(){
217
218         POS_ARRAY=0
219
220         while [ $POS_ARRAY -lt $ARRAY_LXC_DBS_SIZE ]; do
221             LXC_CONTAINER="`echo ${ARRAY_LXC_DBS[$POS_ARRAY]} | cut -d: -f1`"
222
223             echo "-- Desabilitar limites de IO: LXC_CONTAINER"
224             $CONFIG_PATH/controlador_disco.sh $LXC_CONTAINER 0 0
225
226             POS_ARRAY=$((POS_ARRAY+1))
227         done
228 }
229
230 # ------------------------------------------------
231 # Sinaliza que o gerenciador de recursos vai iniciar uma execucao
232 echo "============================================================"
233 echo "==== Gerenciador de Recursos - `date +"%d-%m-%y %T"`: Inicio"
234 echo "============================================================"
235 echo "1" > $CONFIG_PATH/lock/gerenciador_recursos_lock.txt
236 # ------------------------------------------------
237
238 # Variavel de controle para sinalizar o final da execucao do gerenciador de
    recursos
239 FLAG_ENCERRAR_SESSAO=0
240 echo "0" > $GERENCIADOR_RECURSOS_ENC_FILE
241
242 # Zerar o arquivo de output e cria o cabecalho do arquivo
243 echo "DATA_HORA;LXC_CONTAINER;LXC_DB;LIMITE_READ_MB;LIMITE_WRITE_MB" >
    $LXCSTATUS_REPORT_FILE
244
245 while [ $FLAG_ENCERRAR_SESSAO -eq 0 ]; do
246
247     # Verifica se existe o arquivo de controle
248     echo "-- Verifica o arquivo de controle..."
249     if [ -f $GERENCIADOR_RECURSOS_ENC_FILE ]
```

```
250     then
251         FLAG_ENCERRAR_SESSAO=`cat $GERENCIADOR_RECURSOS_ENC_FILE`
252 
253         if [ $FLAG_ENCERRAR_SESSAO -eq 1 ]
254         then
255             echo "-- Foi solicitado que o gerenciador de recursos seja encerrado.
    FLAG_ENCERRAR_SESSAO = 1"
256             break
257         fi
258     fi
259 
260     # Executa o monitor de recursos e o monitor de performance
261     echo "-- Iniciar monitores de recursos e de performance..."
262     CHECK_MONITOR_RECURSOS = `ps -ef | monitor_recursos.sh | grep -v grep | wc -
    l`
263         if [ $CHECK_MONITOR_RECURSOS -eq 0 ]
264         then
265             $CONFIG_PATH/monitor_recursos.sh
266         fi
267         CHECK_MONITOR_PERFORMANCE = `ps -ef | monitor_performance.sh | grep -v
    grep | wc -l`
268         if [ $CHECK_MONITOR_PERFORMANCE -eq 0 ]
269         then
270         $CONFIG_PATH/monitor_performance.sh
271         fi
272 
273     # Verifica o status dos banco de dados
274     echo "-- Verificar status dos bancos de dados..."
275     verificarDBStatus
276 done
277 
278 # Desabilita os limites de IO antes de fechar o programa
279 echo "-- Desabilitando limites de IO para os containers..."
280 desabilitarLimitesIO
281 
282 # ------------------------------------------------------
283 # Sinaliza que o gerenciador de recursos finalizou uma execucao
284 echo "============================================================"
285 echo "==== Gerenciador de Recursos - `date +"%d-%m-%y %T"`: Fim"
286 echo "============================================================"
287 echo "0" > $CONFIG_PATH/lock/gerenciador_recursos_lock.txt
288 # ------------------------------------------------------
```

- code/gerenciador_recursos.sh