Pontifícia Universidade Católica
do Rio de Janeiro

Liester Cruz Castro

Sintonia fina de Sistemas de Gerenciamento de Banco de Dados em ambientes virtualizados

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio.

Orientador: Prof. Sérgio Lifschitz

Rio de Janeiro
Abril de 2017

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Liester Cruz Castro

Sintonia fina de Sistemas de Gerenciamento de Banco de Dados em ambientes virtualizados

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Sérgio Lifschitz
Orientador
Departamento de Informática – PUC-Rio

Prof. Sérgio Colcher
Departamento de Informática – PUC-Rio

Profª. Noemi de La Rocque Rodriguez
Departamento de Informática – PUC-Rio

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 26 de abril de 2017

Liester Cruz Castro

Graduado em Engenharia em Ciências Informáticas (2008) pela Universidad de Ciencias Informáticas, na Havana. Cuba. Atua principalmente nos seguintes temas: sistemas de banco de dados, sintonia-fina de banco de dados.

Ficha Catalográfica

Cruz Castro, Liester

Sintonia fina de Sistemas de Gerenciamento de Banco de Dados em ambientes virtualizados / Liester Cruz Castro; orientador: Sérgio Lifschitz. – 2017.
84 f. : il. color. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2017.
Inclui bibliografia

1. Informática – Teses. 2. Bancos de dados. 3. Sintonia fina. 4. Ambientes virtualizados. I. Lifschitz, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

A minha esposa Elvismary Molina de Armas, companheira de jornada, incentivadora e alicerce.

Ao meu orientador que como um verdadeiro mestre me mostrou o caminho, motivou a melhorar, apontou as falhas sempre de forma construtiva, deu o suporte necessário para a pesquisa, e foi um grande amigo.

A minha mãe Fidelina Castro Osorio e meu pai Rogelio Cruz Álvarez, que me ensinaram a vencer as dificuldades da vida e ser uma pessoa honesta. A minha irmã Liyenni Cruz Castro por sempre seguir o meu exemplo. Aos pais de minha esposa, que tanto quero e admiro. Ao resto de minha família. Aos meus vizinhos em Jarahueca, que também contribuíram na minha educação.

Aos meus amigos da Universidad de Ciencias Informáticas, especialmente a: Eddy Dangel Quesada Rodríguez, Dagoberto Antonio Suarez e Victor Frank Molina Lopez, por tantos anos de amizade inquebrantável.

Aos amigos Yania Molina Souto Padron, Gil Capote Mastrapa por ser como família para mim. Aos amigos colombianos Neileth J. Stand Figueroa e Cesar A. Diaz Mendoza.

A todos os amigos que conquistei na PUC-Rio, em especial: Julio Omar Prieto Entenza, Alain Dominguez Fuentes, Adriel Gárcia Hernández, Sonia Fiol González, Jefry Sastre, Liander Millan, Rafael Pereira, Alexander Chávez López, Alejandro Mustelier Menes e Grettel Monteagudo García.

A Capes, ao CNPq e a PUC-Rio, pelo apoio financeiro concedido ao longo do curso.

A todos os funcionários e professores do departamento de Informática da PUC-Rio, meu agradecimento e admiração.

Resumo

Castro, Liester Cruz; Lifschitz, Sérgio. **Sintonia fina de Sistemas de Gerenciamento de Banco de Dados em ambientes virtualizados**. Rio de Janeiro, 2017. 84p. Dissertação de Mestrado. — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Devido à enorme quantidade de dados nas aplicações atuais, observa-se o uso crescente dos Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR) em ambientes virtualizados. Isto contribui para aumentar os requisitos das operações de entrada e saída (E/S) das cargas de trabalho relacionadas. É introduzida uma grande sobrecarga para aplicações intensivas em operações de E/S, devida à virtualização dos dispositivos e ao escalonamento das máquinas virtuais. Este trabalho tem por objetivo propor estratégias que permitam aumentar o rendimento das operações de E/S gerenciadas pelos SGBDR em ambientes virtualizados. Por meio da alocação de recursos computacionais, realizamos uma sintonia fina nas ações do escalonador do ambiente virtualizado e também nos parâmetros dos bancos de dados envolvidos. Para isso, foi desenvolvido um sistema que trabalha de maneira coordenada com as diferentes camadas de virtualização. Foram realizados experimentos que permitem avaliar e medir o impacto da abordagem aqui proposta.

Palavras-chave

bancos de dados; sintonia fina de parâmetros; virtualização

Abstract

Castro, Liester Cruz; Lifschitz, Sérgio (Advisor). **Tuning of Database Management Systems in virtualized environments**. Rio de Janeiro, 2017. 84p. Dissertação de Mestrado. — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Due to the huge amount of data present in current applications there is a growing use of Relational Database Management Systems (RDBMS) in virtualized environments. This fact increases the workloads' input/output (I/O) requirements with respect to the corresponding workloads. This is due to resources virtualization and virtual machines scheduling. Our work's goal is to propose strategies that enable better performances for the I/O operations managed by the RDBMS. Considering an intelligent assignment of computational resources, we have executed fine tuning actions at the virtualized environment and on database parameters. We consider a system that works coordinately with distinct virtualization layers. We show some experimental results that evaluate and measure the impact of our proposed approach.

Keywords

databases; parameters tuning; virtualization

Sumário

1 Introdução 11

1.1. Motivação 11

1.2. Contexto 12

1.3. Objetivos 13

1.4. Estrutura da dissertação 14

2 Conceitos e fundamentos 16

2.1. Virtualização 16

2.2. Máquina virtual 17

2.3. Monitor de máquinas virtuais (VMM) 17

2.3.1. Escalonador do VMM 18

2.4. Parâmetros de configuração nos SGBDs 19

2.4.1. Parâmetros de configuração associados ao uso de memória no SGBD 22

2.4.2. Parâmetros associados ao subsistema de recuperação do SGBD 24

2.4.3. Fragmentação e parâmetros de configuração do SGBD 25

2.5. Conclusões 26

3 Sintonia fina de SGBDs em ambientes virtualizados 28

3.1. Estado da arte 28

3.2. Cenários de pesquisa 31

3.3. Ajuste de parâmetros de configuração do SGBD 32

3.3.1. Seleção e ajustes de parâmetros de configuração do SGBD 32

3.3.2. Heurística de ajuste de parâmetros 35

3.4. Ajuste de recursos do ambiente de virtualização privado 38

3.4.1. Heurística de ajuste de recursos 40

3.4.2. Modelo de custo e calculo de recursos necessários para uma VM 43

3.4.3. Modelo de distribuição de recursos para cada VM 45

3.5. Conclusões 48

4 Avaliação da sintonia fina de ambientes virtualizados 50
4.1. Arquitetura da solução proposta 50
4.1.1. Arquitetura para o cenário 1 50
4.1.2. Arquitetura para o cenário 2 52
4.2. Implementação da abordagem de sintonia fina 54
4.3. Análise experimental 57
4.4. Análise dos resultados e discussão 62
4.4.1. Análise das configurações de parâmetros 63
4.4.2. Análise das ajustes de recursos 66
4.5. Conclusões 70

5 Considerações Finais 72
5.1. Contribuições 72
5.2. Trabalhos futuros 74

6 Referências Bibliográficas 75

7 Anexos 80
Anexo 1: Detalhes das cargas de trabalho utilizadas. 80
Anexo 2: Configurações encontradas em ajuste de parâmetros. 80
Anexo 3: Configurações encontradas em ajuste de recursos. 81

Lista de figuras

Figura 3-1: Heurística de ajuste de parâmetros. 35

Figura 3-2: Heurística de ajuste de recursos. 41

Figura 4-1: Arquitetura do cenário 1 51

Figura 4-2: Arquitetura do cenário 2 52

Figura 4-3: Média dos resultados do conjunto de experimentos realizados 57

Figura 4-4: Média dos resultados dos experimentos realizados com a CT3 59

Figura 4-5: Principais componentes e sua relação. Experimento para medir o acesso a disco 60

Figura 4-6: Resultados do experimento para medir o acesso a disco 61

Figura 4-7: Resultados do experimento execução VMs sequencial vs VMs juntas 62

Figura 4-8: Resultados obtidos para o cenário 1 63

Figura 4-9: Resultado obtidos para o cenário 2 67

Lista de tabelas

Tabela 7-1: Informação dos dados das tabelas do esquema TCP-H (10GB). 80

Tabela 7-2 Configuração de parâmetros com menor tempo de execução encontrada. 81

Tabela 7-3: Alocação de recursos de cada VM. 82

Tabela 7-4 Configuração de parâmetros com menor tempo de execução encontrada. 82

Tabela 7-5: Alocação de recursos de cada VM. 83

Tabela 7-6: Configuração de parâmetros com menor tempo de execução encontrada. 83

1
Introdução

Os Sistemas Gerenciadores de Banco de Dados Relacionais (SGBD) estão sendo cada vez mais usados em ambientes virtualizados [1], [2]. Estes ambientes permitem que diversos sistemas operacionais operem sobre um mesmo hardware, isoladamente, como computadores independentes. Entretanto, devido ao compartilhamento, o acesso aos recursos existentes como processadores e discos, constitui um dos principais gargalos da virtualização [3], [4].

O incremento no volume e na demanda de dados tem impactado o tempo de execução dos SGBDs [5], [6], evidenciando-se, em muitos casos, um incremento significativo nas operações de leitura e escrita em disco [6]. O gargalo no desempenho dos ambientes virtualizados é mais claro quando acontecem operações intensivas de leitura e escrita em disco, que têm maior custo de processamento do que as operações feitas inteiramente na memória principal [6], [7]. Visando diminuir os efeitos gerados pela virtualização no desempenho dos SGBD, neste trabalho é proposto realizar e executar ações de sintonia fina que permitam obter melhores tempos de resposta ou maior vazão (throughput).

1.1.
Motivação

Seja um ambiente de virtualização privado, presente em diversas empresas e laboratórios de pesquisa como o existente no laboratório BioBD da PUC-Rio. Existe uma máquina servidora, virtualizada com várias Maquinas Virtuais (VMs), e cada uma destas VMs conta com uma instalação de um SGBD com um banco de dados, onde se fazem testes na área de sintonia fina de banco de dados relacionais e aplicações de sistemas de banco de dados na Bioinformática.

Por conta do baixo desempenho observado em experimentos diversos, foi decidido realizar alguns testes para permitir a compreensão do problema. Os testes mostraram que no ambiente de virtualização privado, com várias VMs executando-se simultaneamente, o tempo de execução de um experimento aumentou devido ao fato que os recursos alocados à VM do teste foram acessados de forma compartilhada. O teste foi projetado para fazer muitas operações de E/S em disco. Cada VM teve que aguardar suas requisições de E/S ao SGBD serem efetivamente realizadas. Os tempos de espera no acesso ao disco somados constituíram o atraso observado com respeito à execução em um ambiente não virtualizado. No experimento feito com apenas uma VM executando o teste ligada, o comportamento é mais parecido com um PC independente, pois não há competição pelo acesso aos recursos alocados.

1.2. Contexto

Esta pesquisa se desenvolve baseada na existência de um ambiente de virtualização privado. Trata-se de um ambiente computacional bastante presente em outros centros de pesquisa.

Um ambiente de virtualização privado é definido nesta pesquisa por:

- Uma máquina servidora com várias VMs criadas e sistemas executando concorrentemente em cada uma delas.
- Cada VM tem um SGBD e uma carga de trabalho com alta demanda de operações de E/S.
- Os usuários que interagem com este ambiente são:
    - O Administrador de banco de dados (DBA): Responsável pela configuração e manutenção dos parâmetros de um SGBD.
    - O Administrador do sistema virtualizado: Responsável pela administração, configuração e orquestração do ambiente virtualizado.

Os papéis destes usuários não são mutuamente exclusivos e podem ser atribuídos a um mesmo usuário, caso este tenha os conhecimentos técnicos necessários.

Nesta dissertação não são estudados os ambientes de virtualização públicos porque, diferentemente dos ambientes de virtualização privados, estes ambientes são gerenciados por terceiros e o serviço oferecido a uma organização especifica (como o laboratório BioBD) pode não ser exclusivo ou dedicado [8].

Nesta pesquisa serão estudados os ambientes virtualizados para usuários ou clientes, que tenham disponíveis todos os recursos computacionais, tanto de software como de hardware.

1.3. Objetivos

Os objetivos deste trabalho são:

- Melhorar o desempenho de um SGBD em uma dada VM.
- Melhorar o desempenho do conjunto de SGBDs no ambiente de virtualização privado.

São propostas nesta pesquisa as seguintes abordagens:

- Busca da configuração adequada dos parâmetros do SGBD para se obter um melhor aproveitamento dos recursos alocados em uma VM particular no ambiente de virtualização privado, levando em consideração as necessidades de operações de E/S.
- Melhora da distribuição de tempo de CPU (tCPU) e da alocação de memória (RAM) para cada VM no sistema de virtualização privado, levando em consideração as necessidades de operações de E/S em cada uma delas.

O primeiro objetivo poderá ser cumprido com os resultados obtidos pela primeira abordagem. Já o segundo objetivo poderá ser atendido tendo em conta ambas abordagens.

Como objetivos específicos desta dissertação são listadas a seguir as principais atividades realizadas:

- Estudo das principais regras no ajuste de parâmetros de configuração de um SGBD estabelecidas por desenvolvedores de sistemas de banco de dados, além de critérios utilizados por especialistas.
- Proposta de uma heurística que permita fazer ajustes de parâmetros de configuração do SGBD baseados na informação proveniente da execução da carga de trabalho. Em particular, incluir a capacidade de calibrar esses parâmetros a partir dos recursos alocados à VM.
- Proposta de uma heurística que permita a alocação de recursos mais adequada a cada VM, utilizando como critério o custo de operações de E/S de cada VM.
- Implementação das ideias elaboradas para conseguir fazer os ajustes de sintonia fina necessários nos parâmetros de configuração do SGBD e na alocação de recursos do ambiente virtualizado.
- Avaliação com experimentos práticos das propostas sugeridas e implementadas de forma a atender os objetivos propostos para esta pesquisa.

Cabe ressaltar que este trabalho não procura obter a configuração ótima de parâmetros do SGBD para a carga de trabalho em execução, nem a melhor alocação possível de recursos para cada VM. A ideia é, criar as condições para configurações alternativas que permitam a diminuição dos tempos de execução de sistemas de banco de dados em ambientes virtualizados.

1.4. Estrutura da dissertação

O capítulo 2 apresenta as definições e os conceitos necessários para o entendimento deste trabalho de dissertação e do contexto onde está

inserido. Especialmente, são detalhadas funcionalidades de interesse no SGBD e seus parâmetros de configuração associados.

O capítulo 3 mostra as principais ideias presentes em trabalhos relacionados ao contexto desta pesquisa. Além disso, se descrevem as abordagens do problema, detalhando-se as heurísticas para o ajuste de parâmetros e recursos do ambiente de virtualização privado.

O capítulo 4 detalha o desenho arquitetônico e a implementação das heurísticas, descrevendo os resultados experimentais dos testes realizados e as conclusões alcançadas.

Finalmente, o capítulo 5 apresenta as conclusões finais, explicita as contribuições desta dissertação e lista possíveis trabalhos futuros.

2
Conceitos e fundamentos

Neste capitulo são apresentados os principais conceitos e fundamentos necessários para conseguir um melhor entendimento do contexto onde está inserida a presente pesquisa de dissertação de mestrado.

2.1.
Virtualização

A virtualização descreve uma tecnologia na qual uma aplicação, o sistema operacional ou o armazenamento de dados são abstraídos do hardware subjacente. A virtualização de servidores usa uma camada de software chamada Monitor de Máquinas Virtuais (VMM da sigla em inglês) para emular o hardware subjacente, incluindo-se geralmente processadores, memória principal, memória secundária e rede. O sistema operacional que normalmente interage diretamente com o hardware, faz sua interação com a emulação deste hardware. Muitas vezes o sistema operacional não está conciente de que está em um ambiente virtualizado [9], [10].

Embora o desempenho não seja igual ao desempenho do sistema operacional em execução diretamente sobre o hardware existente, o conceito de virtualização funciona porque a maioria dos sistemas operacionais e aplicações não precisam do uso completo dos recursos computacionais o tempo todo. Além disso, a virtualização permite economia tanto na aquisição de recursos como na administração de varias máquinas.

2.2.
Máquina virtual

Uma máquina virtual é um ambiente de execução de aplicações que pode ser instalado e executado sobre uma camada de software com função de hospedeira. Seu funcionamento tem base no paradigma hospedeiro – hóspede (em inglês host), onde os hóspedes são as próprias máquinas virtuais.

Cada máquina virtual hóspede funciona com uma imitação virtual da camada de hardware. Essa abordagem permite que o sistema operacional instalado no hóspede seja executado sem modificações. Também permite-se criar hóspedes que usem diferentes sistemas operacionais. O hóspede não está ciente de que ele está sendo executado em hardware virtualizado. No entanto, exige recursos de computação supostamente reais ao hospedeiro [11, 12].

2.3.
Monitor de máquinas virtuais (VMM)

O Monitor de Máquinas Virtuais (VMM) [13], [14] também conhecido como hypervisor (em inglês), é uma camada intermediária de software que mapeia os recursos virtuais demandados pelas aplicações no ambiente virtualizado para recursos físicos reais. Ao gerenciar esse mapeamento e alterá-lo, um VMM pode ser usado para permitir que várias aplicações compartilhem recursos e alterem sua alocação conforme seja necessário. Um VMM também é a camada que permite aos administradores da virtualização fazer ações de gerenciamento e monitoramento das máquinas virtuais [12].

Existem dois tipos de VMM, os de tipo I e os de tipo II [15]. Um VMM de tipo II é aquele que funciona tendo como base de hospedagem um sistema operacional. Já um VMM de tipo I é aquele que também pode ser executado diretamente no hardware sem a necessidade de um sistema operacional hospedeiro. Exemplos de VMMs tipo I incluem as soluções de virtualização de mainframe oferecidas por empresas como IBM [16] e

soluções como VMware ESX [17] e Xen [18], [19]. O Xen tem sido amplamente utilizado na área acadêmica [3], [19], [21], [22], [23] para a experimentação e pesquisa devido a sua robustez como framework aberto e as facilidades que provê no gerenciamento das VMs e configuração dos escalonadores do VMM.

2.3.1. Escalonador do VMM

Os escalonadores (schedulers) têm jogado um papel importante no sucesso da virtualização. Isto porque é ele que orquestra o acesso aos recursos compartilhados na virtualização, estabelecendo prioridades e alocações sobre o uso dos mesmos para cada VM do sistema.

Os escalonadores podem ser classificados como escalonadores de compartilhamento proporcional (Proportional Sharing (PS) em inglês) ou de compartilhamento justo (Fair-Share em inglês), de acordo com o intervalo de tempo durante o qual o escalonador fornece uma alocação de CPU para cada VM ativa.

Os escalonadores PS alocam tempo de CPU para cada VM a partir de suas necessidades de processamento. Isso permite aos usuários uma maneira natural de fazer as alocações de tempo de CPU para grandes números de VMs e processadores. Os escalonadores PS procuram fornecer de forma instantânea o compartilhamento entre as VMs ativas de acordo com o tempo de CPU alocado a cada uma delas. Em contrapartida, os escalonadores de compartilhamento equitativo tentam fornecer uma forma de compartilhamento de tempo de CPU equitativa para cada VM, com base no tempo e no uso real medido de cada VMs em longos períodos de tempo [23].

2.4.
Parâmetros de configuração nos SGBDs

O desempenho dos sistemas de banco de dados depende em grande medida do ajuste apropriado dos parâmetros de configuração dos SGBDs. Para se conseguir um ajuste adequado desses parâmetros, as características do hardware subjacente devem ser consideradas.

Tradicionalmente, os SGBD contêm muitos parâmetros de configuração, gerando-se assim uma alta combinatória para se obter a melhor configuração para um desempenho desejado. É por isso que os DBAs terminam consumindo muito esforço e tempo para encontrar os melhores valores de parâmetros com o objetivo de melhorar o desempenho das aplicações de interesse. Acrescenta-se a isso que alguns dos parâmetros de configuração têm dependências não muito conhecidas ou não são parametrizáveis no SGBD [24]. Dessa forma uma pequena modificação em um parâmetro de configuração pode gerar um impacto negativo importante em um comportamento do SGBD não relacionado diretamente ou, ao menos, não evidentemente com o parâmetro modificado. Muito tempo e esforço podem ser desperdiçados ajustando-se os parâmetros e não se obter os efeitos desejados ou, ainda pior, gerar efeitos marginais negativos.

As propostas de ajuste de sintonia fina também variam dependendo da experiência dos DBAs [25]. Escolher os parâmetros de configuração do SGBD a ajustar, baseado no efeito histórico de modificações no rendimento do sistema de banco de dados, é uma opção a seguir nas ações de sintonia fina de SGBD. Assim, são eliminadas ações de sintonia fina para modificar parâmetros de configuração com pouco ou nenhum efeito sobre o rendimento do SGBD.

Ajustar um sistema de banco de dados para atender objetivos de desempenho desejados pode ser uma tarefa desafiadora. Configurações ruins podem gerar resultados de desempenho significativamente piores que aquelas geradas pelas boas configurações. Alterações em alguns parâmetros causam efeitos locais e incrementais no uso dos recursos, enquanto outros causam efeitos drásticos, como alterar planos de consulta

ou levar o gargalo presente em um recurso para outro recurso usado pelo SGBD. Esses efeitos variam dependendo das plataformas de hardware, carga de trabalho e propriedades dos dados [24], [25].

Grupos de parâmetros podem ter efeitos não independentes. O impacto no desempenho, após alterar um parâmetro, pode variar com base em configurações diferentes de outro parâmetro [24]. Um exemplo se manifesta no grupo de parâmetros associados ao uso de memória do SGBD. Quando são testados diferentes valores do parâmetro para a atualização do otimizador do SGBD sobre o tamanho de cache de disco do sistema operacional, se podem ver diferentes respostas de desempenho do SBD. Se o parâmetro para o ajuste do tamanho do buffer do SGBD é alterado fora de seu valor padrão, e se repete o teste anterior, vão se obter valores de desempenho diferentes para o mesmo ajuste de tamanho de buffer. A mesma coisa pode acontecer para outros ajustes do tamanho de buffer, mostrando-se a dependência que existe no ajuste de parâmetros deste grupo [24].

Existem ferramentas focadas no ajuste de parâmetros encontradas na literatura, tendo como exemplos: IBM DB2's Configuration Advisor [26]**,** PostgreSQL's pgtune [27] e iTuned [28]**.** As duas primeiras ferramentas recomendam configurações padrões baseando-se em perguntas previamente definidas e respostas fornecidas pelos DBAs. A ultima ferramenta utiliza técnicas estatísticas para inferir possíveis configurações de parâmetros de SGBD.

Estas técnicas se aplicam sobre os resultados de um conjunto de experimentos que tiveram como entrada uma carga de trabalho. O principal objetivo é obter boas configurações de parâmetros a partir dos efeitos que tem o ajuste de um parâmetro associado a um recurso determinado (CPU, RAM); este recurso também pode ser ajustado em outros parâmetros do SGBD. Não foi estudado como a informação que brindam os planos de execução da carga de trabalho pode influenciar nos resultados de desempenho do SGBD fazendo-se ajuste de parâmetros. Alem disso, não foi estudado como atualizar o SGBD sobre o uso dos recursos alocados ao meio onde se executa. Ate onde é mostrado, as ferramentas citadas fazem

ajuste de parâmetros de configuração de SGBDs executando-se em ambientes não virtualizados.

Dentro dos parâmetros normalmente disponíveis para configuração nos SGBD podemos citar os que afetam de forma global a configuração do SGBD:

- Tamanho de buffers.
- Tempo de execução dos checkpoints.
- Tempo de execução da desfragmentação no SGBD.

Entretanto, existem outros parâmetros mais específicos, que afetam as decisões do otimizador de consultas durante a seleção dos planos de execução:

- Custo de uma busca de página aleatória.
- Custo da CPU processar uma tupla.
- Tamanho do cache do buffer de disco do sistema operacional.

As configurações padrão dos parâmetros mencionados são definidas pelos desenvolvedores dos SGBDs para manter ampla compatibilidade com o software e hardware ao invés de ser otimizadas para melhorar desempenho [29].

Os DBAs dependem de tentativa e erro, de regras de manuais de especialistas, ou de suas próprias experiências. Geralmente, executam experimentos para fazer a análise *what-if* durante o ajuste de parâmetros. Em um experimento típico, o DBA executa a seguinte lista de passos [24]:

1. Cria uma réplica do banco de dados de produção em um sistema de testes.
2. Modifica os parâmetros do SGBD no sistema de testes para uma configuração escolhida de recursos e carga de trabalho.
3. Executa a carga de trabalho a ser avaliada.
4. Observa o desempenho resultante e compara com o estado inicial.
5. Se o resultado for bom o suficiente, fica com a configuração atual; senão, pode voltar ao passo 2 e tentar outra configuração de parâmetros.

Portanto, uma abordagem natural para o ajuste de parâmetros é realizar uma série de experimentos cuidadosamente planejados até obter uma configuração desejada. No entanto, a execução de experimentos pode ser um processo demorado e custoso.

2.4.1. Parâmetros de configuração associados ao uso de memória no SGBD

Os SGBDs são ferramentas que têm como objetivo fundamental persistir grandes volumes de dados e permitir o acesso a esses dados da forma mais simples, segura e rápida, sendo uns dos principais gargalos no seu desempenho o tempo de acesso aos dados em memória secundária.

Como o SGBD poderia melhorar seu rendimento conhecendo todos os níveis de armazenamento pelos quais os dados devem transitar, sendo o acesso intensivo aos dados em disco uma de suas principais funções?

Lamentavelmente, os registros de CPU e a cache da CPU não podem ser efetivamente ajustados pelo DBA. O SGBD geralmente é apenas mais um programa executando sobre o SO e tem um espaço da RAM definido para seu uso.

Um ajuste eficaz do banco de dados pode propor como estratégia o aumento da quantidade de dados úteis no espaço disponível da RAM, impedindo assim o acesso ao disco, sem afetar adversamente outras áreas do SO. Felizmente, o SGBD tem um gerenciador específico que permite otimizar as operações de leitura e escrita em disco, mantendo uma cache própria de dados no seu espaço de RAM chamada de *buffer*. Este é usado para guardar os dados das principais operações feitas pelo SGBD. Existem diferentes tipos de buffers de acordo com o nível de especialização da implementação usado pelo SGBD para gerenciar os dados em memória principal.

O *tamanho de buffer* do SGBD pode ser ajustado mediante parâmetros de configuração. O SGBD aloca a memória definida para buffer no parâmetro de configuração quando ele é instalado e iniciado. Esta área

permanece inalterada durante toda a execução, mesmo ninguém esteja acessando ao banco de dados.

Se houver memória suficiente para armazenar todos os programas e dados, é necessário pouco gerenciamento de memória. No entanto, se todos os dados não cabem na RAM, o sistema operacional começa a mover páginas de memória para uma área de disco chamada swap. São movimentadas páginas que não foram usadas recentemente de acordo com a política implementada pelo SO. Essa operação é chamada de *swap pageout*. Esta operação normalmente não é um problema pois acontece durante períodos de inatividade. O problema aparece quando essas páginas têm que ser trazidas de volta a partir do swap (*swap pagein*). Esta operação, por sua vez, afeta o desempenho do SGBD já que, enquanto a página é movida do swap, o SGBD fica em espera até que esse processo seja concluído [30], [31].

Para grandes conjuntos de dados como não cabe tudo em memória, o SGBD irá ordená-los em partes, colocando resultados intermediários em arquivos temporários. Aumentar o tamanho memória usada por lotes de ordenação cria menos arquivos temporários e, geralmente, permite uma ordenação mais rápida. No entanto, se os lotes de ordenação são muito grandes, eles causam *swap pageins* porque partes do lote de ordenação fazem a operação de *pageout* durante a ordenação. Nesses casos, é muito melhor usar lotes de ordenação menores e mais arquivos temporários.

A quantidade de *memória usada para lotes de ordenação* alocada em *buffers temporais* nos SGBDs se pode ajustar num parâmetro, sendo comumente usado em cada conexão ativa (seção) no SGBD que esteja executando uma ordenação. Cada seção de acesso simultâneo ao banco de dados usará a quantidade de memória máxima estabelecida pelo parâmetro previamente configurado. A memória total usada, associada a esse parâmetro, será a soma de cada seção.

Tanto o tamanho do buffer como a quantidade de memória usada para lotes de ordenação atuam sobre o mesmo recurso: a memória principal. Portanto, supondo uma quantidade de memória máxima a usar para qualquer operação pelo SGBD, dado que a RAM é finita, não se pode maximizar um desses parâmetros sem que se afete o outro.

O sistema operacional, por sua vez, também tem uma cache para dados que são lidos ou escritos no disco (cache de disco). Os blocos de dados persistentes em disco podem ser carregados no buffer do SGBD a partir do cache de disco do SO, correspondendo a uma simples operação de movimentação de dados dentro da RAM. Se os dados não estiverem no cache de disco do SO, estes têm que ser lidos do disco, o que é uma operação mais custosa do que ler da memória principal.

Existe um parâmetro de configuração que é utilizado para dar como entrada ao otimizador do SGBD um possível *tamanho de cache de disco do SO,* assim o otimizador pode utilizar esta informação para fazer uma melhor seleção dos planos de execução das consultas.

2.4.2. Parâmetros associados ao subsistema de recuperação do SGBD

Existem vários critérios que podem ser combinados para ajustar o subsistema de recuperação do SGBD [7]:

1. Colocar a área de armazenamento dos registros de atualizações em disco (s) dedicado (s) para otimizar o tempo de E/S.

2. Atrasar a escrita das atualizações no banco de dados o maior tempo possível.

3. Sacrificar o mecanismo de recuperação após falhas por um melhor desempenho do banco de dados, alterando os intervalos definidos para executar os checkpoints e os backups do banco de dados.

4. Reduzir o tamanho daquelas transações de atualização que sejam muito grandes.

Os SGBDs têm parâmetros que permitem o ajuste do seu subsistema de recuperação.

A maioria dos SGBDs conta com parâmetros de configuração que permitem definir o *tempo com que os checkpoints são executados*. Esses parâmetros permitem definir o espaço de tempo entre os checkpoints. Levando em conta a quantidade de operações de E/S para executar os checkpoints, pode-se afirmar que aumentando-se o intervalo de tempo entre eles é possível melhorar o desempenho dos bancos de dados, mesmo que em detrimento do mecanismo de recuperação [7], [32].

Existem parâmetros de configuração para ajustar o protocolo *Write Ahead Logging* (WAL) implementado nos SGBDs. Esses parâmetros permitem configurar o *tamanho do buffer* destinado à atividade de WAL dentro do SGBD. Incrementando-se seu tamanho, é possível orientar o banco de dados a fazer mais operações de WAL na memória principal [31], [32], [33], o que permete maior rapidez na atividade de WAL.

Através dos parâmetros de configuração relacionados com os checkpoints, assim como os relacionados com WAL, é possível atrasar a escrita no banco de dados em disco e, assim, melhorar o desempenho do SGBD com cargas de trabalho que façam muitas operações de E/S [32], [34].

2.4.3. Fragmentação e parâmetros de configuração do SGBD

Para fazer desfragmentação os SGBDs copiam os dados dos arquivos de um banco de dados para um banco de dados temporário. Esta operação desfragmenta os objetos do banco de dados, ignora os espaços livres e reagrupa as páginas individuais. Em seguida, o SGBD copia o conteúdo do arquivo de banco de dados temporário de volta para o arquivo de banco de dados original. Uma vez feito isto, o arquivo de banco de dados original é substituído.

Várias operações de cópia são realizadas com os dados sobre o disco, além da própria atividade de desfragmentação. Isto ocasiona um aumento substancial nas operações de E/S, o que pode atrapalhar o desempenho para as transações das sessões ativas no SGBD.

Os SGBDs apresentam parâmetros de configuração que permitem ajustar a frequência com que é feita a atividade de desfragmentação. Às vezes é aconselhável usar esses parâmetros de configuração para aumentar o intervalo de tempo com que é feita a atividade de desfragmentação, de forma a dar prioridade às outras operações que estão sendo realizadas sobre o disco. Quando existe muita fragmentação de dados em disco é melhor fazer o contrário: modificar estes parâmetros para desfragmentar em intervalos de tempos mais curtos. Assim, a atividade de E/S gerada é pequena e constitui menos carga para o SGBD do que ter muita atividade de E/S.

2.5. Conclusões

O uso das novas tecnologias de virtualização é cada dia mais popular devido às facilidades que existem para criar e manter diferentes máquinas virtuais que trabalham isoladamente como se fossem máquinas independentes. Contudo, as VMs compartilham os recursos de uma máquina física real, criando gargalos nos SGBDs e outras aplicações. Na exploração dos principais conceitos associados à virtualização, neste capítulo se identificou que o monitor de máquina virtual e o escalonador têm papéis fundamentais na forma com que os recursos são gerenciados, e podem ser usados para melhorar o desempenho dos sistemas de bancos de dados presentes em um sistema de virtualização privado.

Além do gerenciamento dos recursos físicos que impactam no desempenho dos bancos de dados em ambientes de virtualização privados, o uso efetivo dos sistemas de banco de dados depende em grande medida de ajustes apropriados dos parâmetros de configuração às

características de hardware subjacentes. Mas ajustar um sistema de banco de dados para atender objetivos de desempenho desejados pode ser uma tarefa desafiadora devido à grande quantidade de parâmetros. Além disso, há o desconhecimento sobre o impacto que um valor estabelecido para um parâmetro pode ter sobre outro. Os DBAs muitas vezes dedicam muito tempo e esforço ajustando os parâmetros e podem não obter os efeitos desejados ou ainda gerar efeitos marginais negativos.

Com este sentido, se propõe nesta dissertação o desenvolvimento de uma ferramenta que permita o ajuste (semi) automático de parâmetros adequado aos recursos compartilhados para um SBD em um ambiente de virtualização privado. Por isso são descritos aqueles parâmetros de configuração que podem impactar no desempenho do SBD, considerando os que ajustam o recurso memória compartilhada pelo SGBD. Além disso, outros parâmetros que impactam na diminuição das operações de E/S em disco foram detalhados, visando também obter melhoria de desempenho.

3
Sintonia fina de SGBDs em ambientes virtualizados

Neste capitulo é descrito o estado da arte no contexto de sintonia fina em SGBDs executando-se em ambientes virtualizados, detalhando-se os artigos que definem o ponto de partida desta área de pesquisa. Também são propostos dois cenários definidos no contexto do problema desta dissertação para avaliação desta pesquisa. Na abordagem são propostas duas heurísticas com o objetivo de conseguir melhoras no desempenho de SGBDs executando-se nos cenários definidos.

3.1.
Estado da arte

As tecnologias de virtualização estão sendo amplamente adotadas. Existe um interesse crescente em de automatizar a implantação e o controle de aplicações virtualizadas, incluindo os sistemas de banco de dados [1], [35]. No entanto, há poucos trabalhos que estudam o problema de desempenho de sistemas de BD em ambientes de virtualização privados [12], [20], [36].

Os autores em [36] discutem a existência de alguns parâmetros de virtualização que permitem controlar a forma em que os recursos do servidor (tempo de CPU e memória) são compartilhados para suas máquinas virtuais. Eles argumentam que ajustar esses parâmetros de virtualização poderia influenciar em um ajuste melhor dos parâmetros do SGBD, o que significa que tanto esses parâmetros de virtualização junto com os parâmetros de SGBDs poderiam ser ajustados simultaneamente. Com base nessa ideia, o artigo se concentra no problema de obter o melhor desempenho em sistemas de bancos de dados executando-se em VMs, quando estes estão executando sobre diferentes cargas de trabalho.

Para encontrar os parâmetros de SGBD adequados, eles consideraram um processo de calibração experimental. Nesse processo, encontraram parâmetros de ajuste de SGBD baseados em recursos

atribuídos para a VM onde o SGBD estava sendo executado. Essa decisão ajudou na sincronização da execução da carga de trabalho em tempo de execução, dado que o SGBD utilizou parâmetros pré-ajustados do processo de calibração. No caso, os autores usaram o SGBD PostgreSQL para os experimentos. Além disso, o hypervisor Xen foi escolhido como tecnologia de virtualização devido às suas funcionalidades, que ajudam facilmente aos administradores do ambiente de virtualização a alocar recursos para cada máquina virtual em tempo de execução.

Usando o custo estimado que o otimizador dos planos de consulta do SGBD retorna para cada execução de consulta, os autores em [36] propuseram um modelo de custo para tentar minimizar o custo global de todas as cargas de trabalho que podem acontecer sobre o sistema de virtualização, levando-se em conta diferentes alocações de recursos a cada VM. O modelo de custo foi formulado como um problema de otimização combinatória. Para ajustar os resultados de custo estimados, eles propuseram encontrar os valores apropriados para os parâmetros do otimizador, considerando o efeito dos recursos atribuídos na VM na qual o SGBD estava em execução. Tendo em conta as ideias apresentadas, propuseram uma abordagem para enfrentar este desafio baseando-se na utilização do otimizador em modo *"what if"*, atualizado sobre as características da virtualização.

Os experimentos apresentados em [36] foram focados exclusivamente no recurso CPU. Com esse objetivo, o artigo considera os parâmetros de configuração do SGBD para ajuste de CPU que têm influência no custo do plano de consulta calculado pelo otimizador. Executando cada consulta de uma carga de trabalho usando diferentes valores de alocação de CPU, eles conseguiram calibrar o funcionamento do SGBD em termos de desempenho com diferentes valores nos parâmetros relacionados com o uso de CPU pelo SGBD [37].

No processo de calibração eles descobriram qual VM estava precisando de mais tempo de CPU do que as outras no escalonador do hypervisor e atribuíram novas prioridades do tempo de CPU para cada VM. Do mesmo modo, a quantidade de memória foi atribuída igualitariamente para cada VM. O resultado final mostrou melhores tempos de execução

para as novas alocações de tempo de CPU no escalonador do VMM e ajustes de parâmetros de CPU no SGBD. Isso demonstra de que a atualização do SGBD sobre os recursos alocados pode impactar no desempenho dos sistemas implementados sobre o mesmo.

Em trabalhos posteriores [12], [20] foram introduzidas ideias para estender os resultados obtidos anteriormente, apresentando-se a alocação automática de tempo de CPU e memória. É proposto o uso de informações em tempo de execução para refinar as recomendações anteriores para esses recursos. Adicionalmente, foi introduzido um esquema dinâmico de realocação de recursos para lidar com as mudanças nas características da carga de trabalho em tempo de execução, e foi apresentada uma avaliação empírica das técnicas apresentadas.

Recursos como tempo de CPU (tCPU) e memória foram ajustados separadamente em [12], [20], [36]. Os aspectos sobre como diferentes alocações de tCPU e memória combinadas impactam no desempenho dos SGBD não foram aprofundados. Esses trabalhos analisaram o impacto das alocações de tempo de CPU das VMs nos escalonadores, levando em consideração cargas de trabalho intensivas em CPU sobre os SGBD. No entanto, também é necessário analisar o que foi mostrado no trabalho [3] sobre a degradação de desempenho de E/S existente nos escalonadores dos VMM, para levar em conta a sintonia fina nestes escalonadores e diminuir as operações de E/S usando a alocação de tempos de CPU [23], [38].

Não foram analisados os parâmetros de configuração que influenciam no uso da memória pelos SGBDs [12], [20], [36]. Por exemplo, o parâmetro *work_mem*; que se relaciona com a quantidade de memória usada para lotes de ordenação, foi mantido nos experimentos com o seu valor padrão. Ele não foi configurado em correspondência com a quantidade de memória disponível para os SGBDs, nem com as possíveis operações de ordenação de cada uma das consultas das cargas de trabalho [39]. Além disso, para cargas de trabalho intensivas em leituras e escritas em disco, não foi feito análises com a informação que retorna o otimizador dos SGBDs sobre o uso dos buffers [40], com o objetivo de fazer uma melhor configuração de de buffers.

3.2.
Cenários de pesquisa

A partir dos objetivos definidos para essa pesquisa se pretende explorar aqui dois cenários distintos e realistas:

Cenário 1: Quando no ambiente de virtualização privado atua a pessoa que tem o papel de DBA. Esse usuário pode atuar somente sobre a VM que tem acesso, onde está se executando um SGBD. Ele não conhece a configuração da virtualização e não tem acesso a administração do ambiente privado de virtualização. O DBA pode até não conhecer que esta trabalhando num ambiente virtualizado e pensar que esta o SGBD se executando somente numa PC. Também pode ser o usuário encarregado da configuração, administração e sintonia fina de um SGBD.

Cenário 2: Neste caso existe um usuário com o papel de DBA que vai fazer as mesmas atividades descritas no cenário 1. Além disso, teremos um outro usuário com o papel de administrador do ambiente virtualizado. Esse segundo usuário é o encarregado de administrar o VMM no ambiente privado de virtualização. Logo, será uma pessoa com conhecimentos específicos de virtualização, administração e sintonia fina desse tipo de ambiente. Resumindo, tanto o papel de DBA como o papel de administrador do ambiente virtualizado, vão operar ao mesmo tempo, onde as decisões de configuração de uma das partes são comunicadas à outra parte. Com isto espera-se conseguir tanto uma melhor sincronização como uma boa configuração.

O primeiro cenário definido só pode ser implementado com o papel de DBA devido ao fato que o processo de sintonia fina é feito sobre o SBD que funciona sobre uma VM. O segundo cenário precisa dos papéis de DBA e de administrador do ambiente virtualizado, pois abrange todo o contexto do ambiente de virtualização privado.

3.3. Ajuste de parâmetros de configuração do SGBD

No cenário 1 só atua o papel de DBA, que pode operar somente sobre a VM que tem acesso, ficando preso da alocação de recursos feita para essa VM.

O desempenho de um SGBD é medido no âmbito desta pesquisa dado o tempo de execução (*TE*) de uma carga de trabalho (*CT*). Foi decidido modelá-lo através de uma função, *f*:

$$TE = f(CT, RA, CP) \qquad (1)$$

Usa-se uma abstração em que se envolvem três variáveis: o conjunto de recursos alocados à VM onde se executa o SGBD (*RA*), a carga de trabalho (*CT*) que é executada e a configuração (conjunto de valores) dos parâmetros do SGBD (*CP*). Esta função de desempenho deve ser avaliada para se obter o menor TE possível dada uma CT e RA, modificando as CPs.

Mas, como conhecer uma configuração de parâmetros que possa ter impacto positivo no tempo de execução de uma carga de trabalho?

Para isso é definida a seguir uma heurística de ajuste de parâmetros de configuração de SGBD que vai procurar obter o menor tempo de execução de uma carga de trabalho entre possíveis configurações de parâmetros candidatas obtidas com parâmetros previamente selecionados, levando em conta seus possíveis efeitos no desempenho do SBD.

3.3.1. Seleção e ajustes de parâmetros de configuração do SGBD

Uma das atividades feitas nesta pesquisa de mestrado foi identificar aqueles parâmetros de configuração que têm maior probabilidade de impactar positivamente nos objetivos propostos, e descartar aqueles com possíveis efeitos secundários que tiveram impacto negativo sobre os resultados esperados. A seleção de um grupo reduzido de parâmetros também tem como objetivo reduzir o conjunto de parâmetros a analisar e

ajustar, para diminuir a quantidade de possíveis configurações de parâmetros para o SGBD.

Os parâmetros de configuração dos SGBDs diretamente relacionados com os recursos alocados às VMs foram identificados como candidatos para analise, a partir de seus efeitos no desempenho dos SBDs. Dentre desse grupo de parâmetros, se analisaram os que influem no uso da memória principal no SGBD. Essa eleição foi feita a fim de ter atualizado o otimizador do SGBD da quantidade de memória disponível que pode utilizar, isto em um ambiente de virtualização privado onde os recursos podem ser realocados beneficia o desempenho do SBD. Se existe memória disponível no ambiente virtualizado esta poderá ser utilizada e, assim, se pode aumentar a probabilidade de encontrar os dados procurados em memória e não ter que buscá-los no disco. Isso implica que é possível diminuir a quantidade de leituras e escritas em disco e ganhar em desempenho.

Existe um conjunto de parâmetros que influem no uso de memória pelo SGBD. Estes parâmetros, apesar de receber em diferentes nomes nos SGBDs, implementam funcionalidades similares uma vez sendo ajustados.

Analisando-se para cada plano de execução de consulta a quantidade de leituras e escritas em buffer e a quantidade de leituras e escritas em disco, pode-se calcular a quantidade de blocos de memória acessados pela consulta. Como os blocos de memória têm tamanho fixo nos SGBDs, se pode calcular a quantidade de memória que representa a atividade de E/S feita em memória principal e secundária. A partir disso, se pode encontrar a maior quantidade de memória utilizada entre as consultas, que vai representar a maior alocação de memória necessária para poder executar a carga de trabalho sem acessar a disco. Utilizando esta quantidade encontrada mediante o ajuste dos parâmetros de memória do SGBD, pode-se melhorar o desempenho do SBD, devido ao maior aproveitamento desta memória disponível.

Outro resultado importante dado por um plano de execução de uma consulta é a quantidade de blocos de memória acessados em atividades de ordenação. Com esta quantidade de blocos de memória acessados se pode também calcular a quantidade de memória principal necessária para

evitar estas operações de ordenação em disco. Isto se pode fazer obtendo a maior quantidade de memória utilizada para lotes de ordenação entre as consultas da carga de trabalho. Esta maior quantidade representa a medida de memória para lotes de ordenação a alocar para realizar todos os processos de ordenação da carga de trabalho em memória principal.

Além da quantidade possível de memória a alocar nos buffers de um SGBD, é necessário deixar memória disponível para alocar na cache de disco e para o sistema operacional. A quantidade de memória calculada para alocar nos buffers do SGBD representa uma medida de memória necessária para conseguir fazer as operações de leitura e escrita da carga de trabalho em memória principal, mas não se devem fazer os ajustes dos buffers antes de obter um equilíbrio nos ajustes de memória entre o SGBD e o sistema operacional.

A partir da ideia de conseguir melhor desempenho no sistema de banco de dados, foram identificados outros parâmetros que atuam sobre funcionalidades que provê o SGBD em que se pode gerar, em algumas ocasiões, muita leitura e escrita em disco. Controlando-se a forma em que se fazem estas funcionalidades, pode-se melhorar o desempenho de um SBD.

Os parâmetros que foram identificados são os que estão relacionados com as funcionalidades dos *checkpoints*, o *WAL* e a desfragmentação do SGBD. Cada uma destas funcionalidades realiza processos muito importantes dentro do funcionamento dos SGBD, mas se pode afetar parcialmente o funcionamento delas para conseguir melhor desempenho do sistema de banco de dados a partir de diminuição do acesso ao disco.

Os parâmetros que podem ajustar-se para atualizar o SGBD sobre recurso CPU não foram analisados nesta dissertação de mestrado. Isto é devido a que o uso de CPU por parte do SGBD não impacta diretamente nas operações de leitura e escrita feitas em disco, que constituem o principal gargalo de desempenho destes sistemas [7].

3.3.2.
Heurística de ajuste de parâmetros

A heurística de ajuste de parâmetros recebe como entrada um conjunto de valores recomendados para os parâmetros de configuração de ajustes de memória, do subsistema de recuperação e da atividade de desfragmentação. Este conjunto de valores são tomados das recomendações dos principais especialistas dos SGBDs e outros critérios propostos pelo DBA como ajuste.

Outras entradas desta heurística são a própria carga de trabalho a ser executada sobre o SGBD. Obtém-se para cada consulta seu plano de execução retornado pelo otimizador do SGBD, para que seja utilizado nas próximas análises. Além disso, é considerado como entrada o recurso de memória alocado à VM onde funciona o SBD.

Segundo a função 1, a partir das entradas definidas deve-se encontrar com esta heurística uma configuração candidata que sugira o menor tempo de execução entre as possíveis configurações candidatas criadas mediante os valores de entrada para os parâmetros de configuração selecionados. A presente heurística segue o conjunto de ações mostradas na Figura 3.1. São detalhadas a seguir os seus principais passos.

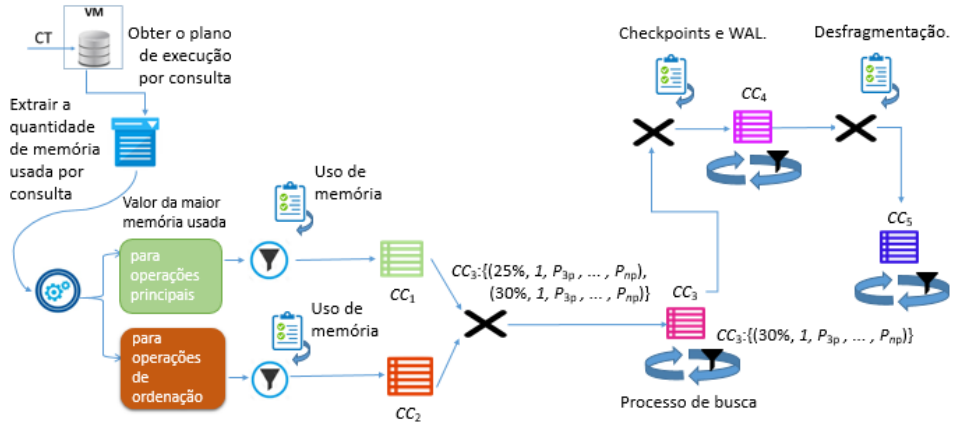

Figura 3-1: Heurística de ajuste de parâmetros.

Passo 1: Sintonia fina de parâmetros para uso de memória

Depois de obter os planos de execução de cada consulta, esta heurística faz a extração da quantidade de memória usada por consulta para operações principais e para operações de ordenação. Determina-se a maior quantidade de memória usada para estas duas operações conforme explicado na seção 3.3.1.

Estas duas quantidades são utilizadas como critério de filtragem. São mantidos os valores do parâmetro de configuração de tamanho de buffer do SGBD que não ultrapassem a maior quantidade de memória calculada para operações principais, estes valores conformam as configurações candidatas a testar para este parâmetro (CC_1). Aqueles valores que não cumprem esse critério são descartados. Também mantem critérios de filtragem similares com os valores do parâmetro para estabelecer a quantidade de memória usada para lotes de ordenação, quando estes não ultrapasam a maior quantidade de memória destinada às operações de ordenação, obtendo-se as configurações candidatas CC_2.

Após do processo de filtragem, a heurística de ajuste de parâmetros combina cada valor em CC_1 com os valores de CC_2. Isto traz como resultado várias configurações candidatas que contem a combinação de um valor de cada um destes parâmetros. Além disso, são combinados com cada configuração candidata os valores do parâmetro de ajuste de memória para a cache de disco. As novas configurações candidatas resultantes (CC_3) desta combinação devem cumprir com que a suma dos valores de cada paramêtro de configuração em CC_3 seja menor que a quantidade de RAM disponível. Se isto não acontecer o valor do parâmetro de ajuste de cache de disco é substituído pelo seu valor padrão.

Cada configuração candidata obtida é materializada nos parâmetros de configuração correspondentes no SGBD. Em seguida, a carga de trabalho é executada com essa configuração candidata e obtém-se o tempo de execução. O processo se repete com todas as configurações candidatas existentes, comparando-se os tempos de execução obtidos. Finalmente, a configuração candidata com menor tempo de execução em CC_3 é mantida

e descartadas as restantes. Este é um processo de busca para encontrar uma boa configuração candidata.

Passo 2: Sintonia fina de parâmetros do subsistema de recuperação

Os novos valores de ajustes a combinar para obter configurações candidatas neste passo são feitos combinando os valores dos parâmetros de controle de tempo com que são feitos os *checkpoints* e os valores para o parâmetro de controle de tempo da realização do *WAL* no SGBD, e a melhor configuração previamente encontrada entre os parâmetros de configuração para o ajuste do uso de memória no SGBD. É feita a combinação anterior devido à inter-relação que existe entre os *checkpoints* e o *WAL* no processo de recuperação de falhas.

As configurações candidatas resultantes são submetidas novamente ao processo de busca da configuração candidata com menor tempo de execução, encontrando-se uma nova configuração candidata em CC_4.

Passo 3: Sintonia fina de parâmetros da desfragmentação

Uma nova combinação é feita com a configuração candidata e os valores de entrada para o parâmetro de controle de frequência com que se faz a desfragmentação no SGBD, obtem-se assim as configurações candidatas CC_5.

Se faz o processo de busca em CC_5 para encontrar a configuração candidata com melhor desempenho, encontrando-se finalmente a configuração candidata com menor tempo de execução esperado do processo de busca atual e de toda a aplicação da heurística de ajuste de parâmetros.

É possível concluir que a heurística de ajuste de parâmetros transita por três fases de criação e revisão de configurações candidatas. Nelas se criam configurações a testar para o contexto de parâmetros que influem no uso de memória do SGBD com o objetivo de diminuir as operações de leitura e escrita em disco e melhorar o desempenho do SGBD a partir da realização de sua atividade de E/S em memória principal. Estas

configurações são criadas mediante a combinatória dos valores de entrada para cada parâmetro a testar, e são discriminadas mediante a análise realizada sobre a informação que oferecem os planos de execução da carga de trabalho. As fases seguintes estão orientadas a encontrar boas configurações de outros parâmetros que também impactam no desempenho do SGBD a partir da diminuição da atividade de leituras e escritas em disco.

3.4. Ajuste de recursos do ambiente de virtualização privado

Com o objetivo de diminuir o maior tempo de execução identificado entre os SGBDs que estão se executando nas VMs do sistema de virtualização privado, será necessária a implementação da abordagem 2.

Uma vez alocada uma configuração de recursos para cada VM, de forma independente em cada uma delas se executará o ajuste de parâmetros para achar uma boa configuração, usando o procedimento descrito na seção 3.3.2.

O processo de ajuste de recursos conjuntamente com o ajuste de parâmetros será só aplicável no cenário 2, onde o papel de administrador do ambiente virtualizado pode alterar as alocações de recursos para cada VM, e o DBA em cada VM tem as permissões para ajustar os parâmetros do SGBD. Dessa forma, no cenário 2 será possível, dado uma nova configuração de recursos das VMs, encontrar uma boa configuração de parâmetros do SGBD.

Por outro lado, o ajuste dos recursos será feito levando em conta as necessidades especificas dos SGBDs no ambiente de virtualização privado, usando uma heurística de ajuste de recursos. A proposta aqui é atualizar o SGBD sobre as caracteristicas da virtualização subjacente, de tal forma que os SGBDs nas diferentes VMs possam aproveitar melhor os recursos disponíveis.

O objetivo do processo de sintonia fina neste cenário é conseguir a melhor configuração de recursos da virtualização e parâmetros de SGBD

possível, encontrada em termos de desempenho dos SGBDs em cada VM, para todo o sistema de virtualização privado.

O desempenho do sistema de virtualização privado vai ser medido no âmbito de nossa pesquisa dado o tempo de execução do sistema (*STE*).

Para isso é necessário definir:

- *SRA*: conjunto de configurações de recursos da virtualização na forma:

  $$SRA = \{RA_1, RA_2,..,RA_n\}, \quad (2)$$

  sendo RA_i os recursos alocados a cada VM_i.

- *SCT:* conjunto de configurações de cargas de trabalho, na forma:

  $$SCT = \{CT_1, CT_2,.., CT_n\}, \quad (3)$$

  sendo CT_i as cargas de trabalho a aplicar sobre cada SGBD em cada VM_i.

- *SCP*: conjunto de configurações de parâmetros de SGBDs , na forma:

  $$SCP = \{CP_1, CP_2,.., CP_n\}, \quad (4)$$

  sendo CP_i configurações de parâmetros a aplicar sobre cada SGBD em cada VM_i.

Finalmente o STE definido para cada VM_i pode ser definido como:

$$STE = g(SCT, SRA, SCP) = máximo(\{TE_1,TE_2,...,TE_n\}). \quad (5)$$

Onde se pode evidenciar que podemos obter STE em função de SCT, SRA e SCP determinado pelo máximo entre os TE obtidos para cada VM.

O objetivo do processo sintonia fina realizado para ajustar os recursos do sistema de virtualização é obter o menor STE possível (minimizar o maior TE entre as VMs) dado uma SCT com CT constantes para cada uma das VMs, com a melhor configuração de parâmetros CP encontrada por cada VM, modificando o SRA da virtualização. O processo de ajuste de recursos inclui o processo de ajuste de parâmetros em cada VM independente, descrito na seção 3.3.

3.4.1.
Heurística de ajuste de recursos

A heurística de ajuste de recursos parte da suposição de que uma distribuição de recursos equitativa às necessidades de cada VM no ambiente de virtualização privado permitirá melhor desempenho do sistema. A heurística parte do pressuposto:

- A necessidades de recursos de cada VM é dada pela quantidade de memória principal e tempo de CPU que o SGBD precisa para ter um melhor desempenho, dada uma carga de trabalho.

Então, alocar a maior quantidade de recursos possível a cada VM segundo suas necessidades máximas permitirá um melhor desempenho do sistema.

Os recursos que foram identificados nesta pesquisa por seu impacto no desempenho dos SGBDs são: a memória principal e o tempo de CPU alocado para uma VM segundo a configuração dos escalonadores do VMM.

A memória é importante pois fazendo maior alocação de memória se pode conseguir maior quantidade de leituras e escritas nos buffers do SGBD, sem ter que acessar ao disco. Assim se pode melhorar a rapidez com que a carga de trabalho é executada.

O ajuste dos tempos de CPU (tCPU) nos escalonadores do VMM é importante porque se aumenta o atendimento às VMs com maior atividade de E/S é possível conseguir maior rapidez para esta atividade de E/S, que constitui um gargalo na virtualização [3].

A heurística proposta para apoiar o processo de sintonia fina no ambiente de virtualização privado tem como objetivo ajustar a alocação de recursos em cada VM. Tomando como base essa ação, deve-se ajustar os parâmetros de configuração dos SGBD em cada VM para conseguir diminuir os maiores tempos de execução obtidos nos SBD nas VMs. Para conseguir isto, se seguem um conjunto de passos. A figura 3.2 mostra a sequência destes passos para maior entendimento.

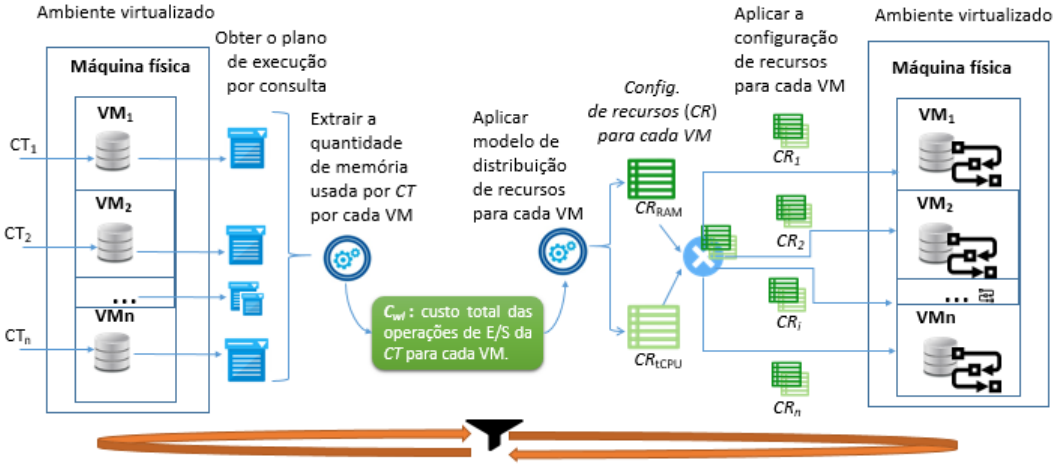

Figura 3-2: Heurística de ajuste de recursos.

Passo 1: Encontrar custo das operações de E/S por VM

A heurística inicia obtendo-se o plano de execução por consulta em cada uma das cargas de trabalho sendo executadas sobre cada SGBD existente em cada VM do ambiente de virtualização privado. Além disso se guarda como configuração global aquela carga de trabalho executando-se sobre certa VM que deu maior tempo de execução.

Com os planos de execução das consultas das diferentes cargas de trabalho executadas, se extrai a quantidade de memória usada por cada carga de trabalho e se calcula o custo das operações de E/S em cada VM. Estas duas atividades estão melhor descritas na seção 3.4.2, que explica a proposta de um modelo de custo para estimar a quantidade de recursos requisitada por uma VM.

Passo 2: Redistribuir e alocar recursos no ambiente de virtualização privado

Para fazer uma boa distribuição dos recursos existentes nas VMs do ambiente de virtualização privado, atendendo suas necessidades de E/S

em disco, é preciso saber como se pode redistribuir os recursos tempo de CPU e RAM em cada VM. Para isso foi criado um modelo de distribuição de recursos. Este modelo (descrito na seção 3.4.3) recebe como entrada o custo das operações de E/S de cada VM, calculado mediante o modelo de custo proposto na seção 3.4.2. Com estas entradas estabelece-se relações entre as VMs a partir dos seus custos e, com essas relações, se chega a uma forma de redistribuição dos recursos entre as VMs.

Uma vez ser obtida a proposta de distribuição de recursos tempo de CPU e RAM para cada VMs, são combinados o valor de ajuste de tempo de CPU e RAM das VM, e criadas configurações candidatas. Em seguida, são aplicadas as configurações candidatas em cada VM.

Passo 3: Encontrar maior tempo de execução no ambiente de virtualização privado

Com as novas alocações de recursos no ambiente de virtualização privado, o passo que deve se fazer em cada VM é atualizar os SGBDs da existência dos novos recursos alocados. Para isso, é executada a heurística de ajuste de parâmetros (descrita na seção 3.3.2) obtendo-se configurações de parâmetros de SGBD com bons tempos de execução por VM.

Os tempos de execução para as diferentes cargas de trabalho executando-se nas VMs são comparados, sendo o resultado deste passo a configuração de parâmetros do SGBD com maior tempo de execução de carga de trabalho correspondente. Este resultado é comparado com o tempo de execução global previamente guardado; se for menor, então se toma como melhor tempo de execução global o resultado atual.

Passo 4: Processo de diminuição do maior tempo de execução global

Os passos 1, 2 e 3 são repetidos até obter a nova configuração candidata de recursos do ambiente virtualizado, e parâmetros do SGBD, que retornem maior tempo de execução entre as cargas de trabalho executadas. O valor de tempo de execução é comparado com o valor de tempo de execução global registrado. Se o valor atual é menor que o valor obtido no passo 3, este valor é guardado como tempo de execução global; senão, mantém-se como tempo de execução global o valor resultante no passo 3.

O processo descrito pode se repetir para encontrar melhores resultados do tempo de execução global. A quantidade de iterações que o sistema deve fazer é definida pelo administrador do ambiente virtualizado. Sempre em cada iteração se compara o tempo de execução atual com o obtido na iteração anterior.

Com os resultados se procura diminuir o pior tempo de execução entre os SBD executando-se nas VMs. Esse tempo representa o pior tempo alcançado no processo de sintonia fina. Se é possível reduzí-lo, então se ganha em desempenho no ambiente de virtualização privado.

3.4.2. Modelo de custo e calculo de recursos necessários para uma VM

Os principais gargalos de desempenho na execução dos SGBD em ambientes virtualizados são as operações de leitura e escrita realizadas em disco. Quanto maior for a carga de operações de leitura e escrita em disco, pior será o desempenho nos SGBD porque estas operações são muito mais custosas do que as que são feitas na memória principal [7]. Também se sabe que os ambientes virtualizados introduzem gargalos de performance quando acontecem operações intensivas de leitura e escrita em disco [3].

Portanto, para melhorar o desempenho de SGBDs em ambientes virtualizados é necessário ter conhecimento das características da atividade de I/O, para assim fazer ações de sintonia fina.

Foi criado um modelo de custo que retorna à quantidade de atividade de E/S feita pelo SGBD, com o objetivo de ser utilizado posteriormente para diminuir a quantidade de operações de leitura e escrita feitas em disco por parte do SGBD.

O modelo de custo para as operações de E/S feitas no SGBD é o seguinte:

Para uma consulta, definimos como seu custo a C_q, onde:

$C_q = SR + SW + TR + TW + SH + TH$. (6)

Sendo:

SR: O número de blocos lidos no disco e copiados no buffer do SGBD (*Shared Reads* em inglês).

SW: O número de blocos que foram levados para o buffer, em seguida modificados e dopois copiados para o disco (*Shared Written* em inglês).

TR: O número de blocos lidos no disco e copiados no buffer temporal do SGBD (*Temporal Reads* em inglês). O buffer temporal é aquele que vai manter a memória usada para lotes de ordenação.

TW: O número de blocos escritos no buffer temporal do SGBD (*Temporal Written* em inglês).

SH: O número de blocos acessados no buffer do SGBD, evitando fazer uma leitura no disco (*Shared Hits* em inglês).

TH: O número de blocos acessados no buffer temporal do SGBD, evitando fazer uma leitura no disco (*Temporal Hits* em inglês em inglês).

Os valores para (SR, SW, TR, TW, SH e TH) são retornados no plano de execução de consulta do otimizador do SGBD para cada uma das consultas, permitindo conhecer as operações de E/S que realiza o SBD. Somando cada uma destas saídas se pode obter o custo da atividade de E/S da execução de uma consulta no SGBD.

Então para uma carga de trabalho que contém várias consultas o custo total de E/S é:

$$C_{wl} = \sum_{q=1}^{N} (C_q) \quad (7)$$

Onde *q* representa uma consulta da carga de trabalho, *N* o número de consultas da carga de trabalho e C_q representa o custo calculado para a consulta *q*.

A formula (7) pode ser utilizada para calcular a *posível maior quantidade de memória que* a VM onde está executando-se essa carga de trabalho precisa. Esse valor compreende, para uma carga de trabalho, toda a atividade de E/S refletida em quantidade de blocos de dados acessados em memória principal e/ou secundária. Como os blocos de memória têm um tamanho fixo nos SGBD, se pode calcular facilmente, em termos de tamanho de memória, a quantidade que representa a atividade de E/S da carga de trabalho (C_{wl} * *tamanho_bloco*).

A posível maior quantidade de memória pode ser utilizada na alocação de recursos atendendo cada VM segundo suas necessidades de E/S, o principal gargalo da virtualização. Alocações de tCPU seguindo este princípio podem ser benéficas no desempenho dos SBDs existentes naquelas VMs com maior quantidade de operações de I/O, já que se prioriza o atendimento dessas VMs que mais competem pelo recurso de disco. Também seguindo o princípio proposto, podem se fazer alocações de memória às VMs a partir de suas necessidades de I/O, diminuindo-se com isto o acesso a disco porque é possível fazer maiores alocações de memória no buffer do SGBD.

3.4.3. Modelo de distribuição de recursos para cada VM

A aplicação do modelo de custo de E/S para uma carga de trabalho pode ser utilizada para fazer alocações de recursos mediante o VMM, atendendo as necessidades de recursos que tem cada VM. Primeiramente é necessário obter a posível maior quantidade de memória necessária para cada VM. Isto se consegue multiplicando-se o C_{wl} que representa a quantidade de blocos de memória usados pelo tamanho em unidades de memória que tem o bloco. Este último valor é definido pelo SGBD,

comumente parametrizado. A partir desse ponto, C_{wl} vai ser usado para referir-se à possível maior quantidade de memória necessária para cada VM em bytes.

O conjunto $SC_{wl} = \{C_{wl1}, C_{wl2}, \ldots, C_{wln}\}$ que contém o C_{wl} para cada VM é a entrada para o segundo passo da distribuição dos recursos na heurística de ajustes de recursos. O modelo de distribuição segue a equação:

$$\alpha_1 x + \alpha_2 x + \alpha_3 x + \ldots + \alpha_n x = T \qquad (8)$$

onde T representa o total de recursos a alocar, e $\alpha_i x$ representa o recurso a alocar à VM_i.

O valor dos coeficientes α_i é dado por $\alpha_i = C_{wli}/C_{wlm}$. Como resultado se tem a razão dos recursos necessários por cada VM com respeito à VM que precisa de menos recursos. Depois é só calcular o valor de x, sendo $x = T/(\alpha_1 + \alpha_2 + \alpha_3 + \ldots + \alpha_n)$.

Usando o modelo de distribuição de recursos apresentado, aplicável tanto ao tempo de CPU como RAM, é possível fazer uma distribuição de recursos equitativa ás necessidades de cada VM.

Para melhor entendimento ilustramos no Exemplo 1 como distribuir recursos de memória.

Exemplo 1: Distribuição do recurso de memória a alocar nas VMs

As VMs neste experimento ilustrativo têm os custos de E/S calculados em 7:

- VM1: 1 GB.
- VM2: 2 GB.
- VM3: 3,5 GB.

E a virtualização tem disponível 16 GB de RAM para essas três VMs, isso é T = 16. Se pretendesse encontrar a alocação de memória principal para cada uma dessas VMs, então a equação se define assim:

$$\alpha_1 x + \alpha_2 x + \alpha_3 x = T$$

onde $\alpha_1 x$ é o tamanho de RAM alocar a VM1, $\alpha_2 x$ é o tamanho de RAM a alocar a VM2 , e $\alpha_3 x$ é o tamanho de RAM a alocar a VM3. $C_{wlm} = \text{minimo}(\{1, 2\ , 3.5\}) = 1$, então $\alpha_1 = 1$, $\alpha_2 = 2$ e $\alpha_3 = 3.5$. Então x = 16/6.5=2.46

Então a alocação resultante é:

VM1 é igual a $\alpha_1 x = 1x = 2{,}46$GB.

VM2 é igual a $\alpha_2 x = 2x = 4{,}92$ GB

VM3 é igual a $\alpha_3 x = 3{,}5x = 8{,}61$ GB

Para ilustrar como, analogamente ao que fizemos para RAM, o modelo de distribuição pode ser usado para alocar o recurso de tempo de CPU às VMs através do escalonador do sistema de virtualização, temos o Exemplo 2.

Exemplo 2: Distribuição do recurso de tCPU a alocar nas VMs

As VMs neste experimento ilustrativo têm os custos de E/S:

- VM1: 1 GB.
- VM2: 2 GB.
- VM3: 3 GB.

Supondo que a quantidade de recurso a alocar é 1768 créditos de CPU (o uso de tCPU durante 20ms consume 200 créditos [41]), isso é T = 1768. Se pretendesse encontrar a alocação de tCPU para cada uma dessas VMs, então a equação é definida por:

$$\alpha_1 x + \alpha_2 x + \alpha_3 x = T$$

Como $C_{wlm} = \text{mínimo}(\{1, 2, 3\}) = 1$, então $\alpha_1 = 1$, $\alpha_2 = 2$ e $\alpha_3 = 3$. Então x = 1768/6=294, 66.

Então alocação resultante é:

VM1 é igual a $\alpha_1 x = 1x = 294$ créditos.

VM2 é igual a $\alpha_2 x = 2x = 589$ créditos.

VM3 é igual a $\alpha_3 x = 3x = 882$ créditos.

3.5. Conclusões

Neste capitulo se mostrou que há poucos trabalhos que estudam o problema de desempenho de SBDs em ambientes de virtualização privados. Nos trabalhos citados é detalhada uma das ideias principais a seguir nesta dissertação, que mostra que os parâmetros da virtualização e os parâmetros de configuração dos SGBDs devem ser ajustados de forma simultanea. Com isto, se podem conseguir melhoras no funcionamento do SBD no ambiente virtualizado, a partir de uma melhor utilização dos recursos alocados.

Também foram detalhados os processos de ajuste de parâmetros e de ajuste de recursos da virtualização. Se apresentou a heurística de ajuste de parâmetros, e a heurística de ajuste de recursos com o modelo de custo associado e o modelo de distribuição de recursos. A proposta leva em conta a sintonia fina nos escalonadores para diminuir as operações de E/S usando a alocação de tempos de CPU e também a alocação da RAM. O ajuste de parâmetros, por sua parte, é baseado na estimativa de recursos necessários para o SGBD na análise da execução da carga de trabalho e os planos de execução gerados.

No próximo capitulo se apresenta uma proposta de sintonia fina de sistema baseada nos cenários definidos: o cenário 1, que permite o ajuste de parâmetros do SGBD de forma individual para cada VM no sistema virtualizado, e o cenário 2, onde se combina o ajuste de recursos da

virtualização com o ajuste de parâmetros dos SGBD de cada VM, com o objetivo de encontrar uma boa configuração que permita melhorar o desempenho do sistema de virtualização privado.

4
Avaliação da sintonia fina de ambientes virtualizados

O presente capítulo detalha as principais características da arquitetura necessária para dar implementar as abordagens e heurísticas explicadas no capítulo anterior. Também são apresentados os ambientes que foram criados para realizar os testes com o sistema desenvolvido. Além disso, são apresentados detalhes sobre a exploração de limites no contexto do problema a partir da experimentação nesse tipo de ambiente. A partir destes limites se encontraram determinadas ideias que foram levadas em conta nas comparações feitas nas análises de resultados. Após a discussão dos resultados obtidos são apresentadas as conclusões do capítulo.

4.1.
Arquitetura da solução proposta

A arquitetura para soluçionar o problema identificado encontra-se dividida em duas partes, que acompanham os cenários 1 e 2 definidos na seção 3.2.

4.1.1.
Arquitetura para o cenário 1

As principais características da arquitetura definida para implementar a solução correspondente ao *cenário 1* são mostradas na Figura 4.1:

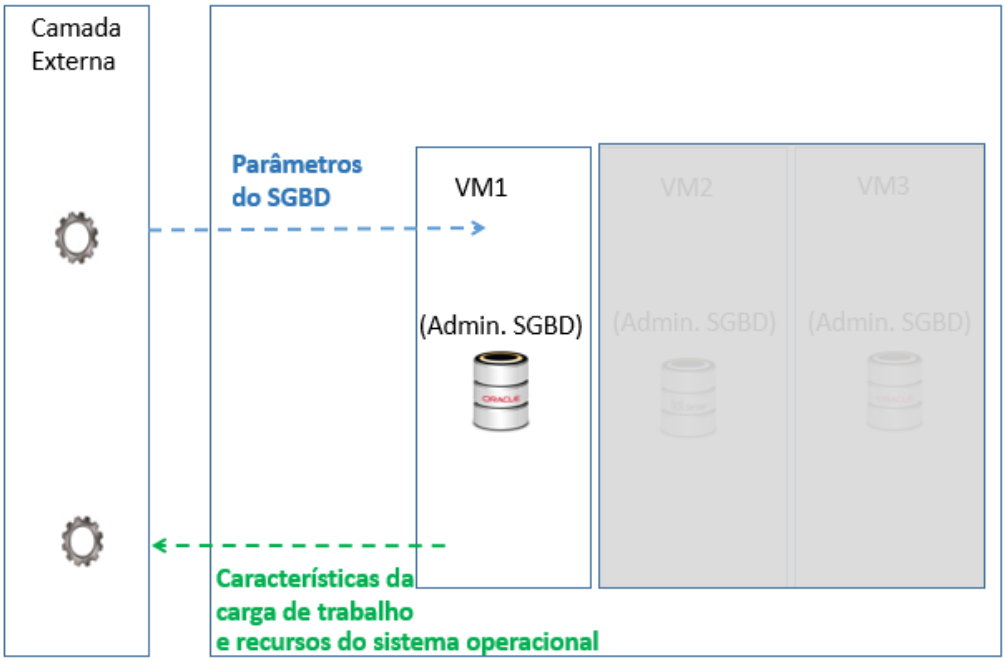

Figura 4-1: Arquitetura do cenário 1

Na figura 4.1 se observam os principais componentes da solução e o fluxo de dados entre eles. Se representa também o ambiente virtualizado com cada uma das VMs. É mostrada só uma VM como abstração das *n* possíveis VMs existentes no sistema. Neste cenário somente interage o papel de administrador de SGBD, ou seja, sobre cada VM se atua de forma individual, e os critérios para o ajuste de parâmetros é independente para cada uma delas. Além do sistema de virtualização (componente geral com as instâncias de cada VM), é representada uma camada externa que vai conter o componente de software que controlará automaticamente o ajuste de parâmetros.

Esta camada externa foi criada com o objetivo de interferir o mínimo possível no processamento e acessos a memória (principal e secundária) de cada VM no sistema de virtualização. O componente de software contido nesta camada tem a responsabilidade de ajustar os parâmetros para um SGBD em uma VM independente, fazendo uso da heurística de ajuste de parâmetros definida na seção 3.3.2. Para conseguir isso, vai capturar as características da carga de trabalho executada sobre o SGBD que funciona dentro da VM. Além disso, este componente vai se recuperar, em intervalos de tempo definidos, os recursos alocados à VM correspondente, a partir

dos mecanismos de consulta de recursos que provê o sistema operacional da VM.

Em seguida, a nova configuração de parâmetros obtida é utilizada para fazer o ajuste de parâmetros no SGBD da VM correspondente, feito pela camada externa remotamente. Todo este processo é repetido no tempo com o objetivo de informar ao SGBD dos recursos alocados do ambiente virtualizado e das características da carga de trabalho.

4.1.2. Arquitetura para o cenário 2

A arquitetura definida para implementar a solução correspondente ao cenário 2 é mostrada na seguinte figura:

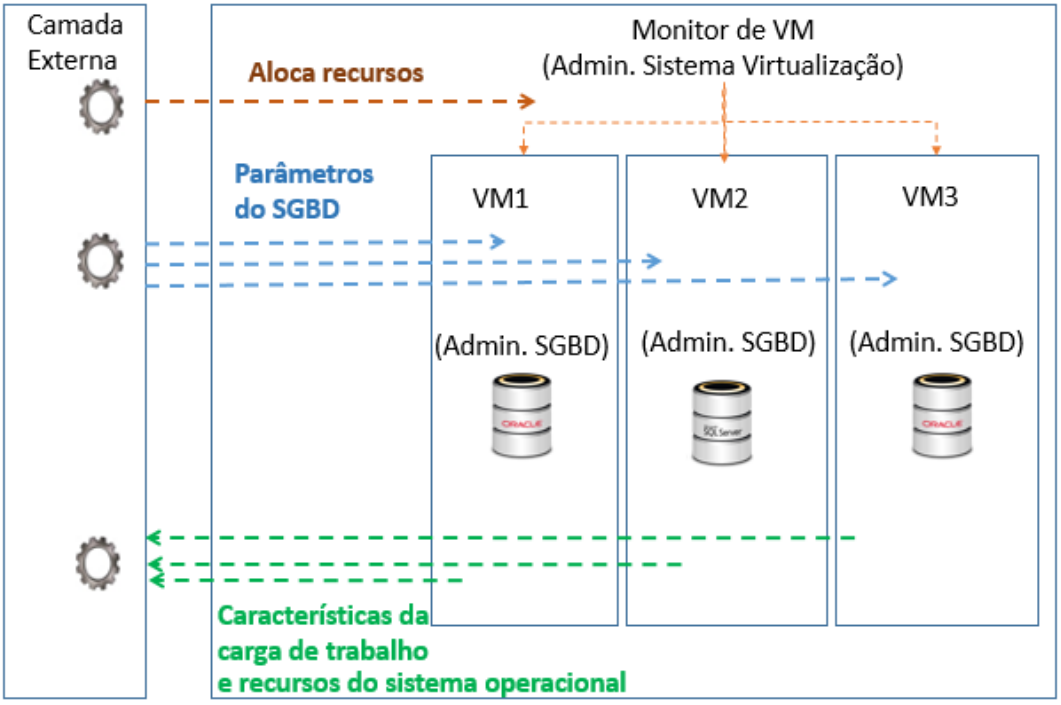

Figura 4-2: Arquitetura do cenário 2

Na figura 4.2 se observam os principais componentes da solução e o fluxo de dados entre eles. Contemplando o ambiente virtualizado com suas VMs, ilustrado com 3 VMs como abstração das *n* possíveis VMs existentes no sistema. Neste cenário interage o papel de administrador de SGBD em

cada VM, que tem um acesso limitado à VM que o controla e tem permissões para realizar o ajuste de parâmetros do SGBD. O papel de administrador do sistema virtualizado também interage neste cenário pois tem uma visão geral de todas as VMs do sistema e tem permissões para ajustar os recursos do ambiente virtualizado.

Também é representada na Figura 4.2 uma camada externa (similar ao cenário 1) que vai conter o componente de software que controlará automaticamente o ajuste de parâmetros e ajuste de recursos para o cenário 2. Esta camada externa foi criada com o objetivo de interferir o mínimo possível no processamento e no acesso às memórias (principal e secundária) desse ambiente.

O componente contido na camada externa tem a responsabilidade de ajustar os parâmetros para cada SGBD em cada VM independente e, de forma global, realizar ajustes na alocação de recursos de cada VM. Também vai capturar as características da execução das cargas de trabalho sobre os SGBDs nas VMs. Além disso, permite recuperar, em intervalos de tempo definidos, detalhes dos recursos alocados em cada VM, a partir dos mecanismos de consulta de recursos que provê o sistema operacional da VM. Os dados recuperados constituem a entrada da heurística de ajuste de recursos definida na seção 3.4.1 que é também executada nesta camada externa.

O processo mencionado é feito pela camada externa de forma remota acessando as VMs onde se encontram os SGBDs e acessando ao VMM onde se define a distribuição dos recursos no sistema virtualizado. Todo este processo é repetido com o objetivo de conseguir obter um bom ajuste na configuração do sistema de virtualização, melhorando seu desempenho. Isto permete ao SGBD ter ciência dos recursos alocados, visando conseguir uma boa configuração de parâmetros ajustada a esses recursos. O mesmo podemos dizer da camada de virtualização (onde reside o VMM) quanto às características das cargas de trabalho, com o objetivo de fazer uma melhor distribuição dos recursos disponíveis em cada VM.

4.2.
Implementação da abordagem de sintonia fina

As restrições propostas na arquitetura descrita (seção 4.1), sugerem a implementação de um sistema que seja não intrusivo sobre o ambiente virtualizado. Para conseguir isto é necessário que o sistema se execute fora do ambiente virtualizado, gerenciando-se remotamente os dados e configurações pertinentes em cada uma das VMs e os SGBDs.

Para conseguir reutilizar e estender funcionalidades de sintonia fina em um SGBD, foi selecionado o framework DBX [42]. Este framework segue uma abordagem não intrusiva para a manutenção automática e em tempo de execução do projeto físico de bancos de dados, sendo completamente desacoplado do código fonte do SGBD. Esta abordagem foi aproveitada e estendida na implementação de sintonia fina de parâmetros de configuração do SGBD e na realização do gerenciamento de recursos de ambientes virtualizados. Além disso, foi reutilizada e estendida a implementação da captura e manipulação dos planos de execução das consultas. A utilização de DBX permitiu um menor esforço e tempo no desenvolvimento, o que permitiu viabilizou dar foco a outros detalhes importantes da implementação do sistema.

As abordagens propostas trouxeram para o DBX novas funcionalidades agregadas que complementam a atividade deste framework.

O DBX segue o ciclo descrito em [43], contendo as fases de Observação, Predição e Reação. Estas fases são executadas conjuntamente para monitorar o sistema, predizer possíveis comportamentos futuros e tomar decisões fazendo alterações sobre as propriedades e o comportamento do SGBD, caso necessário. A arquitetura deste framework propõe realizar, mediante a colaboração de agentes de software, o processo de auto sintonia sobre o funcionamento do SGBD.

As ações implementadas pelo DBX começam com a captura dos dados resultantes nos planos de execução de cada consulta da carga de trabalho. Esta captura é feita pelo agente Observador, que aloca essas informações da carga de trabalho num banco de dados criado para a

utilização exclusiva dos agentes do DBX. O agente Preditor utiliza estes dados na atividade de predição de possíveis ações de sintonia fina sobre o SGBD. A partir dos resultados retornados pelo agente Preditor, o agente Reator pode fazer as ações de sintonia fina pertinentes no sistema de banco de dados.

Na implementação do sistema proposto nesta dissertação foram utilizados todos os agentes expostos. O agente Observador executa a captura e armazenamento dos dados resultantes da execução da carga de trabalho, para ambos cenários definidos no contexto do problema. A heurística de ajuste de parâmetros de configuração do SGBD (seção 3.3.2) é executada pelo agente Preditor. O agente Reator é utilizado na alocação de recursos feita no cenário 2, especificamente após de ser executada a heurística de ajuste de parâmetros e aplicada a heurística de ajuste de recursos (seção 3.4.1). Após alocados estes recursos o processo continua no agente Preditor, que faz novamente o ajuste de parâmetros para esses novos recursos alocados a cada VM. Este processo se repete pelo número de iterações definido para o sistema.

A arquitetura de DBX facilita a criação de novas heurísticas, que podem usar as funcionalidades que provê este framework. Na implementação foi estendido o DBX com as duas heurísticas definidas nas seções 3.3.4 e 3.4.1, a primeira a ser utilizada no processo de ajuste de parâmetros de configuração de SGBDs, e a segunda para utilizar-se no processo de ajuste de recursos no ambiente virtualizado.

Na implementação da heurística de ajuste de parâmetros de configuração dos SGBDs se criaram vários componentes. Um deles propõe configurações candidatas mediante combinação dos valores que recebem os parâmetros de cada fase identificada nesta heurística. Os valores para cada parâmetro são introduzidos mediante um arquivo de configuração no framework DBX, e um destes componentes se encarrega de lê-los e processá-los. Existe mais um componente que é responsável pelas análises da carga de trabalho, estendendo as funcionalidades de DBX de leitura e manipulação da carga de trabalho. As novas funcionalidades adicionadas gerenciam o conteúdo dos buffers existentes nos planos de execução de cada consulta na carga de trabalho. Com estes dados dos

buffers são feitos os cálculos mencionados na seção 3.3.4, para obter ajustes que reduzem a quantidade de configurações candidatas a serem testadas no processo de sintonia fina de ajuste de parâmetros.

Os parâmetros a serem ajustados no processo de sintonia fina em sua maioria apresentam um esquema de ajuste mediante sessões de usuários. Neste esquema um parâmetro pode ter várias configurações para um mesmo usuário ou diferentes usuários (diferentes sessões), sem ter que ser reiniciado o SGBD. Foram utilizados outros parâmetros que somente podem ser alterados após reiniciar o SGBD, para que o valor ajustado faça efeito.

As configurações de parâmetros obtidas com a heurística de parâmetros são ajustadas no SGBD utilizando uma interface de programação de aplicações (API em inglês). Esta API foi criada com o objetivo de interagir com o SGBD para poder ajustar aqueles parâmetros nas sessões de usuário mediante instruções SQL. Também foi criada uma API para a interação com o sistema operacional (SO). Esta API pode reiniciar o SGBD quando for necessário um ajuste de parâmetros específicos. A API para a interação com o SO também foi implementada para obter informação da quantidade de memória alocada às VMs. Isto foi feito porque os valores de ajuste de parâmetros de buffers e de cache de disco estão expressas em porcentagens da quantidade total de RAM utilizada pela VM.

Na implementação da heurística de ajuste de recursos foi criado um componente de alocação de recursos que faz o cálculo para estimar os recursos necessários para cada VM baseando-se no modelo de custo definido na seção 3.4.2. Em seguida, usando o custo a heurística calcula a distribuição das quantidades de recursos (tCPU e RAM) a ser alocados em cada VM.

O componente que faz análises da carga de trabalho também foi implementado para retornar a saída das quantidades de blocos de memória utilizados em cada consulta. Com esta informação, o componente encarregado da alocação de recursos faz os cálculos dos custos para cada uma das VMs.

Para a alocação dos recursos é utilizada a API de interação com o SO. Esta API permite interagir com o VMM mediante sua linha de comandos. Para isso foi criado mais um componente nesta API, com todos os comandos possíveis e suas variáveis de entrada, que vão conter os valores de recursos a alocar.

4.3. Análise experimental

Com objetivo de conhecer melhor o comportamento e funcionamento dos SGBD executando-se em ambientes virtualizados, foi feita uma análise experimental sobre as principais diferenças que têm os SGBDs quando se executam neste tipo de ambiente. Na figura 4.3 é mostrada de forma resumida a média de tempo de execução dos resultados dos experimentos feitos:

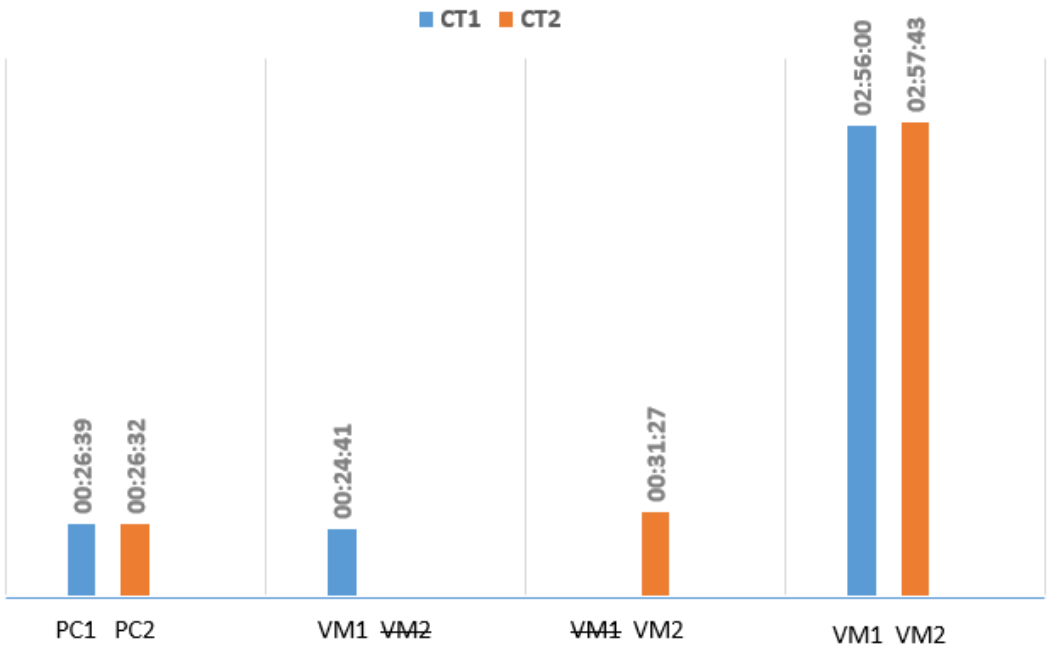

Figura 4-3: Média dos resultados do conjunto de experimentos realizados

Foram criadas duas cargas de trabalho com consultas intensivas em leitura e escritas a disco, de forma que seu tempo de execução numa máquina física seja similar. O conjunto de consultas presentes na carga de trabalho 1 (CT1) ocasionaram maior quantidade de leitura e escritas em

disco que a carga de trabalho 2 (CT2). Esta distribuição de consultas foi realizada para comprovar os efeitos das operações feitas em disco tanto nos ambientes sem virtualização como nos ambientes virtualizados. São mostrados no Anexo 1 outros detalhes destas cargas de trabalho.

O primeiro teste feito com as cargas de trabalho foi realizado em máquinas reais (PC1 e PC2). As máquinas (PCs) foram selecionadas com as mesmas características de recursos disponíveis (RAM, disco, CPU) e os mesmos sistemas operacionais. Primeiramente se executou a CT1 na PC1 e a CT 2 na PC2, os resultados de tempo de execução são mostrados na primeira parte da figura 4.3. Depois foi criada uma virtualização com duas VMs com a mesma configuração de recursos que tinham as PCs reais. Então na VM1 se executou a CT1, tendo a VM2 desligada, após disso se desligou a VM1 e se ligou a VM2 onde se executou a CT2. Os resultados em tempo de execução são mostrados na figura 4.3 no seu segundo e terceiro espaço. O próximo experimento realizado foi executar de forma simultânea na VM1 a CT1 e na VM2 a CT2. Os resultados também são mostrados ao final da figura.

Se pode concluir com estes resultados, que a execução das cargas de trabalho, numa VM isolada, tem um comportamento similar em tempo de execução que executá-las numa máquina real com as mesmas características da VM. O mesmo não quando se executam as duas CT nas duas VMs de forma simultânea, pois o tempo de execução aumenta muito. Devido às características da virtualização onde os recursos são compartilhados entre as máquinas virtuais podem ser esperados atrasos nos tempos de execução das CT em cada VM, mas uma diferença tão notável como a observada precisa de maior análise do que está acontecendo.

Na figura 4.4 é acrescentada mais uma carga de trabalho (CT3) com muitas operações de leitura e escrita em disco com o objetivo de observar os limites dos SGBD executando-se em ambientes virtualizados. Além disso, foi criada uma VM onde se executou esta CT3 sem que existissem outras VMs. Depois foi executada juntamente com as duas VMs utilizadas anteriormente, com suas CTs previamente criadas.

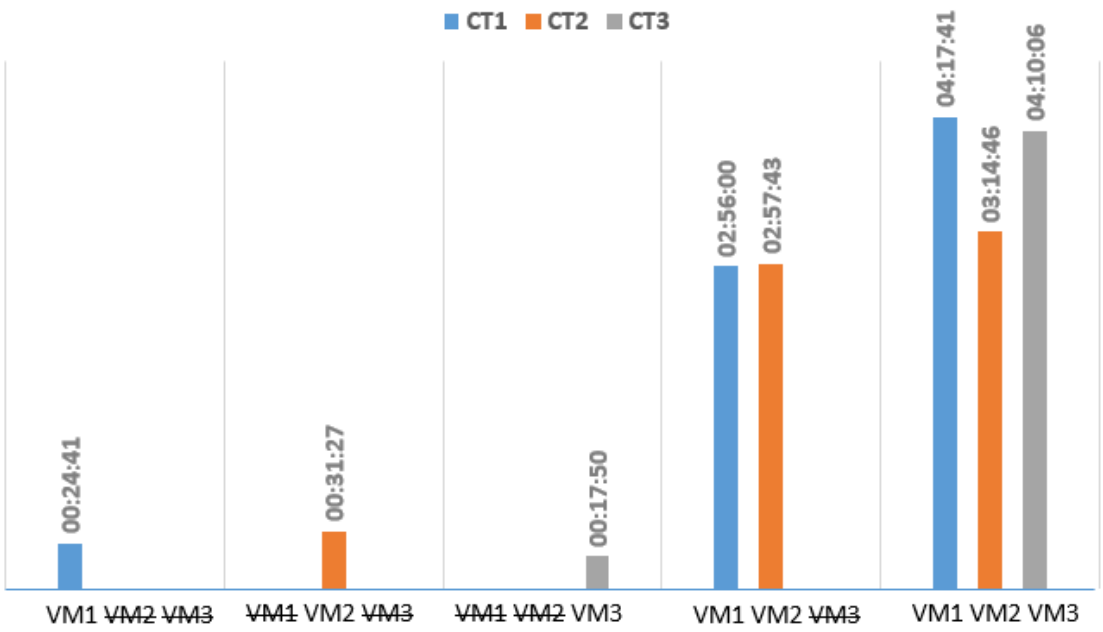

Figura 4-4: Média dos resultados dos experimentos realizados com a CT3

Se pode notar na figura 4.4 que, quando são executadas de forma simultâneas, as três CT nas suas respectivas VMs, o tempo de execução volta a aumentar consideravelmente em comparação com os experimentos anteriores. Um aspecto importante a assinalar no caso das CTs executando-se juntas em suas respectivas VMs, é que o maior gargalo se observa nas CTs das VM1 e VM3, as quais ocasionam maior quantidade de leituras e escritas em disco que a CT2 presente na VM2. Isto evidencia que as operações intensivas de E/S em disco aumentam o gargalo de processamento, devido a competição pelo recurso disco [3].

Conclui-se, a partir das médias dos resultados nos experimentos, que a execução de CTs em VMs isoladas tem um comportamento similar à execução das CT em máquinas reais. Este resultado experimental foi tomado como uma medida base de quanto deve melhorar-se hipoteticamente o rendimento da CT sobre um SGBD numa VM, quando são executadas outras CT sobre outros SGBDs em VMs de forma simultânea. As ideias para melhorar os gargalos de tempo observados, mediante sintonia fina de parâmetros do SGBD e ajuste de recursos da virtualização, seguem esta medida base como comparativa do tempo ideal que se deve obter.

O conjunto de experimentos mostrados anteriormente onde sobre os SGBDs das VM1 e VM2 foram executadas de forma simultânea as CT1 e CT2 respectivamente, foi simulado em um ambiente real utilizando um PC sem virtualização. Como cada carga de trabalho é executada no ambiente virtualizado sobre um banco de dados num SGBD, se pode criar um ambiente real onde existam dois SGBD, cada um deles com um banco de dado, e estejam instalados numa máquina física (PC). A máquina vai ter as mesmas características que a máquina que foi virtualizada para criar as VM1 e VM2.

Então neste novo grupo de experimentos se executaram sobre o primeiro sistema de banco de dados a CT1 e sobre o segundo a CT2, da mesma forma que foi executado no grupo de experimentos anteriores. O principal objetivo foi encontrar as diferenças com respeito ao acesso a disco na virtualização quando comparado aos ambientes reais. Na figura 4.5 se mostram os principais componentes e seus interrelacionamentos no experimento. Na figura 4.6 se mostram os resultados da média dos tempos de execução em cada um dos ambientes.

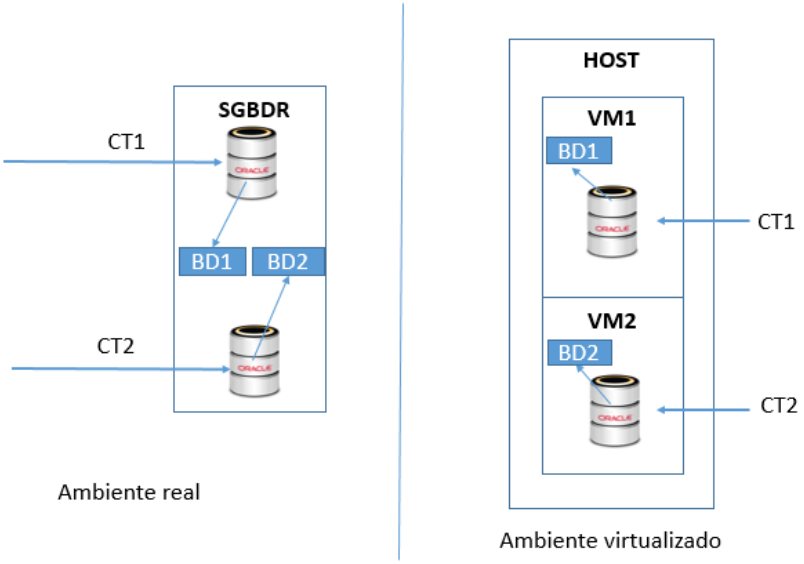

Figura 4-5: Principais componentes e sua relação. Experimento para medir o acesso a disco

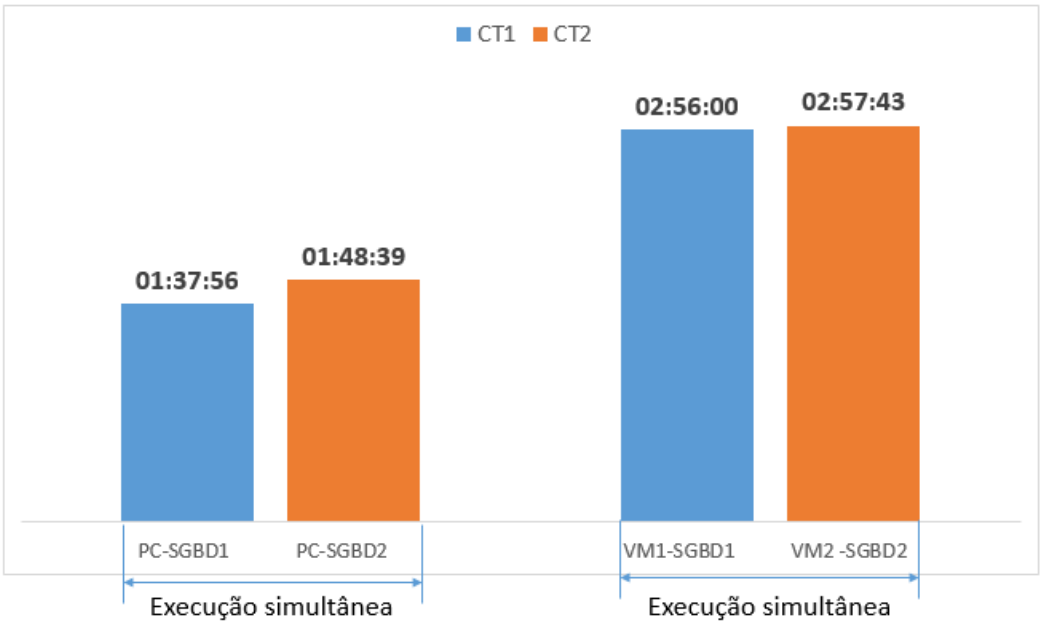

Figura 4-6: Resultados do experimento para medir o acesso a disco

Como se pode ver na figura 4.6 as execuções simultâneas das CTs sobre a PC retornaram menores valores de tempo de execução que as execuções simultâneas das CTs sobre as VM1 e VM2. Estes resultados demonstram mais uma vez que existe gargalo de desempenho na virtualização quando são feitas operações de leitura e escrita em disco, caso que não acontece em um ambiente real não virtualizado.

Se pode notar na figura 4.7 que o maior tempo de execução (TE) das CTs executadas de forma simultânea sobre os respectivos SGBDs nas VMs, tem um valor maior que a soma dos TE das CTs executadas sobre um SGBD numa VM (VM1 ou VM2 ou VM3) sem a interferência de outras VMs. Com o objetivo de obter menor TE total de todas as CT em seu conjunto, uma possível solução é fazer sequencialmente cada execução de CT.

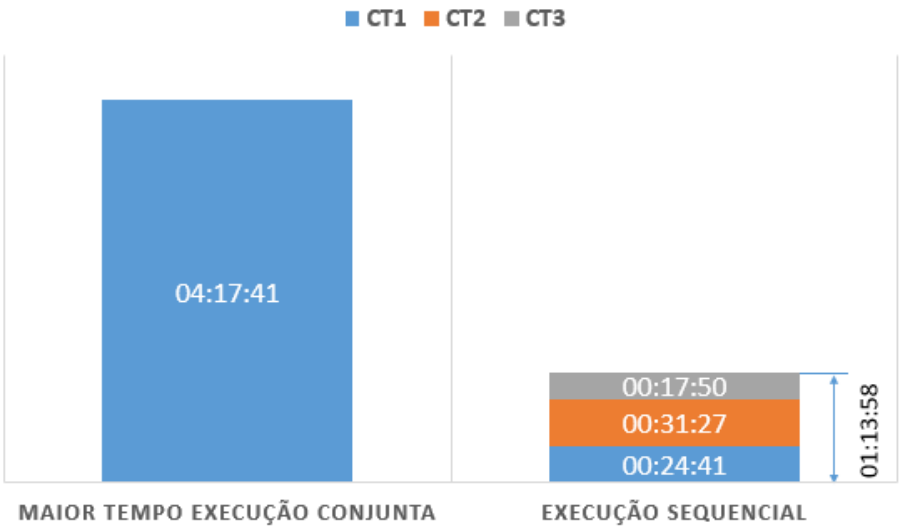

Figura 4-7: Resultados do experimento execução VMs sequencial vs VMs juntas

Esta solução só pode ser feita se os administradores do ambiente de virtualização privado entenderem que é possível executar as VMs uma de cada vez. Finalmente a solução mencionada constitui uma relaxação na execução do ambiente virtualizado, que viola implicitamente o objetivo da virtualização, que é consolidar o funcionamento de várias máquinas num mesmo host mediante VMs.

A presente pesquisa segue o objetivo de melhorar o desempenho do conjunto de SBDs no ambiente de virtualização privado, sempre que não seja desligada alguma das VMs para melhorar o desempenho dos SGBDs dentro delas. Além disso, o segundo objetivo proposto procura melhorar o desempenho de um SGBD numa dada VM sem ter em conta as restantes VMs, mas no contexto em que o administrador de SGBD não tem permissões para desligar as restantes VMs.

4.4. Análise dos resultados e discussão

Os resultados conseguidos com as cargas de trabalho utilizadas na seção 4.3 permitiram explorar os limites no contexto dos sistemas de bancos de dados executando-se em ambientes virtualizados. Isto foi devido à quantidade de operações de E/S que fazem cada uma das consultas das

CTs, que trazem gargalos na virtualização e baixo desempenho na execução dos SGBDs. Conhecendo as características destas cargas de trabalho, se decidiu também utilizá-las no processo de testes sobre o sistema desenvolvido. Mais detalhes destas cargas de trabalho são mostrados no Anexo 1.

Seguindo as abordagens propostas, as análises de resultados e discussão serão detalhadas para cada cenário definido nesta dissertação.

4.4.1. Análise das configurações de parâmetros

Os resultados obtidos nos testes feitos com o sistema sobre as VMs (1, 2 e 3) cada uma com seu SGBD e com suas respetivas CTs (1, 2 e 3), são mostrados no Anexo 2. Estes mostraram configurações com parâmetros ajustados fora do ajuste padrão do SGBD, com menores TEs que as configurações padrões do SGBD. Em estes testes cada VM foi executada tendo desligadas as VMs restantes.

Os resultados obtidos mediante a execução do sistema desenvolvido para cada carga de trabalho neste cenário são mostrados na figura 4.8:

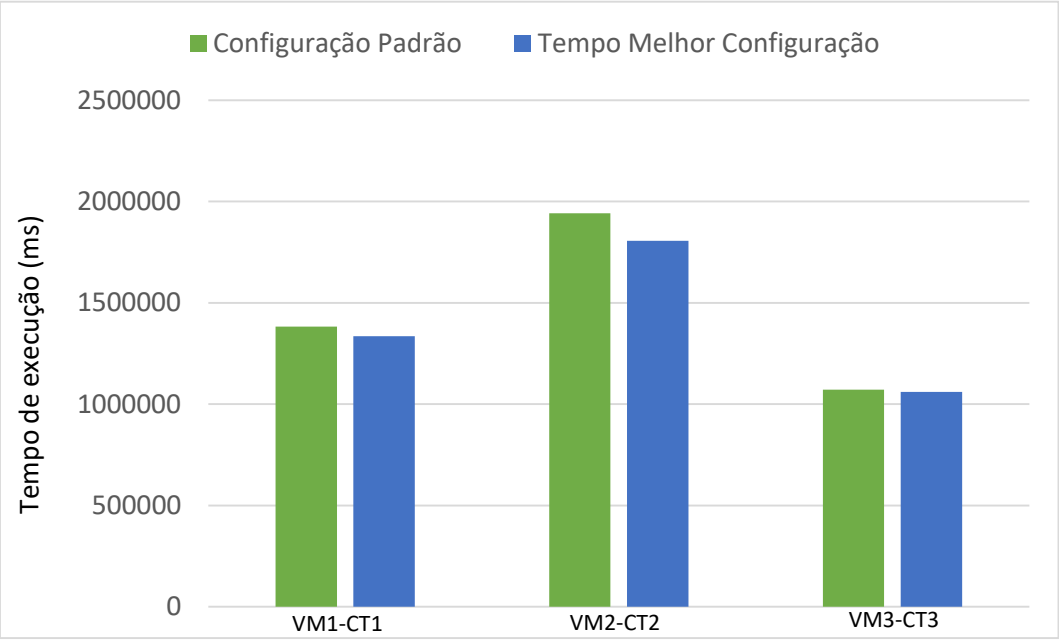

Figura 4-8: Resultados obtidos para o cenário 1

No primeiro gráfico da figura é mostrada para cada VM e a CT executada nela o TE com a configuração padrão existente no SGBD, e ao seu lado o menor TE encontrado (tempo de melhor configuração) utilizando uma configuração de parâmetros recomendada pelo sistema. Como é observado, se comprova que foi encontrada para cada CT uma configuração de parâmetros que diminui o seu TE, o que constitui uma melhora no desempenho do SGBD correspondente. Isto comprova também que o processo de ajuste de parâmetros pode melhorar o rendimento dos SGBD executando-se neste tipo de ambientes. As configurações encontradas podem ser explicadas a partir das ideias que segue a abordagem implementada neste cenário, que procuram entre outros objetivos, um melhor aproveitamento do recurso de memória por parte do SGBD no ambiente virtualizado.

Uma característica comum de todos os resultados obtidos com a heurística de ajuste de parâmetros com as CTs, é que todos os testes mostraram em sua configuração com menor TE, ajustes nos buffers temporais com valores fora do valor padrão. Os planos de execução das consultas das CTs mostraram que 100% das consultas da CT1, 85% das consultas da CT2 e 85% das consultas da CT3, fizeram operações de agrupação e/ou ordenação. Estas operações de ordenação e agrupação usam buffers temporais do SGBD.

Os valores fora do padrão nos parâmetros de configuração dos buffers temporais dos SGBDs, se ajustaram aos critérios definidos na heurística de ajuste de parâmetros. Em todos os casos o sistema calculou a maior quantidade de memória temporária utilizada em cada CT, maior que o valor padrão. Assim, foi criada a configuração candidata que retornou o menor TE no teste com cada CT.

Outras configurações de parâmetros onde se retornaram os menores valores de TE se obtiveram nos testes feitos com a CT3, com resultados de ajuste no parâmetro para a alocação de memória no buffer do SGBD. Neste caso a heurística de ajuste de parâmetros detectou para a CT3 que podiam fazer-se alocações de memória ao buffer com todas as configurações entradas pelo administrador do SGBD. Isto foi devido ao

reportado pela CT3, nos planos de execução, com muitos blocos de memória acessados por consulta (a maior quantidade de todas as CTs). Considerando a análises da CT o sistema testou com as configurações candidatas criadas de todos os valores de entrada. Finalmente a configuração com menor TE teve a maior alocação de memória escolhida entre os valores de entrada para o buffer do SGBD. Se deve assinalar que estas configurações com menor TE se obtiveram junto com as configurações mencionadas anteriormente para os buffers temporários.

Como o ajuste do buffer mencionado para a CT3 levou maior quantidade de memória que a alocação padrão, o sistema só permitiu a criação de configurações com valores adequados de cache de disco. Isto foi feito com o objetivo de não alocar mais memória do que a existente. Finalmente a configuração com menor TE nesta CT teve como ajuste de cache de disco uma porcentagem da memória total que deixou espaço restante em memória para o SO. Esta combinação deu melhores resultados que outras que deixavam menor quantidade hipotética de memória livre a ser usada pelo sistema operacional.

A CT1 também realizou muitas operações de acesso a blocos de memória e o sistema permitiu combinar todos os valores de entrada. Neste caso, a configuração com menor TE foi obtida com maior valor de ajuste no buffer do que a configuração padrão do SGBD. A cache de disco foi ajustada a esse valor de ajuste de buffer obtido. A CT2 fez menor quantidade de acesso a blocos de memória comparado com as restantes CTs. A configuração escolhida com menor TE foi a própria configuração padrão.

Nos testes realizados não se encontraram melhores ajustes que os padrões do SGBD para os parâmetros que impactam nas funcionalidades dos *checkpoints* e da *desfragmentação*. O oposto aconteceu para os ajustes do parâmetro que controlam o tamanho dos arquivos do *WAL* do sistema de banco de dados. Para a CT1 o menor TE encontrado também teve uma configuração que aumentou o tamanho destes arquivos acima do tamanho padrão deste parâmetro. Isto foi devido ao fato que a CT1 é a que mais leituras e escritas ocasionou em disco entre as restantes CTs e ao fazer de maior tamanho os arquivos *WAL,* estes demoraram mais em copiar

para o disco. Esta demora atrapalha menos às restantes operações de leitura e escritas ocasionadas pelo SGBD no disco. Não existiram configurações fora do padrão do SGBD para este parâmetro nos casos das CT 2 e 3. Estas CTs, em sua totalidade, fazem menor quantidade de operações de leitura e escrita em disco onde o tamanho dos arquivos *WAL* é adequado para estas operações.

A natureza da heurística de ajuste de parâmetros implementada ajuda na seleção dos possíveis parâmetros a testar sobre as CTs. Fazendo uso destas facilidades se fizeram várias rodadas sobre várias CTs com outros parâmetros que podiam impactar nas configurações dos buffers do *WAL* e *vacuums*, outros buffers temporários do SGBD, e o processo que escreve os *checkpoints* e *WAL* a disco. Os resultados finais mostraram que as melhores configurações destes foram as suas configurações padrões. Portanto, não foi levado em conta o ajuste dos mesmos na heurística proposta.

O sistema ajudou no processo de refinamento dos valores de entrada para os parâmetros escolhidos. Nas CTs testadas as melhores entradas encontradas nos parâmetros de ajuste de buffers foram até um 40% da memória total alocada à VM, valores acima desta faixa podem dar maiores resultados no TE das consultas. Para ajustar os buffers temporais é aconselhável utilizar pequenos porcentuais da memória total da VM. Nos casos testados, os melhores resultados foram obtidos com até 3% da memória total.

4.4.2. Análise das ajustes de recursos

Os testes do cenário 2 foram feitos com a CT 1 e 2 sobre as VM 1 e 2 respectivamente, e de forma simultânea. Os resultados obtidos mediante a execução do sistema desenvolvido para cada uma das cargas de trabalho e suas VMs correspondentes neste cenário são mostrados na figura 4.9 e se podem ver com mais detalhes no Anexo 3:

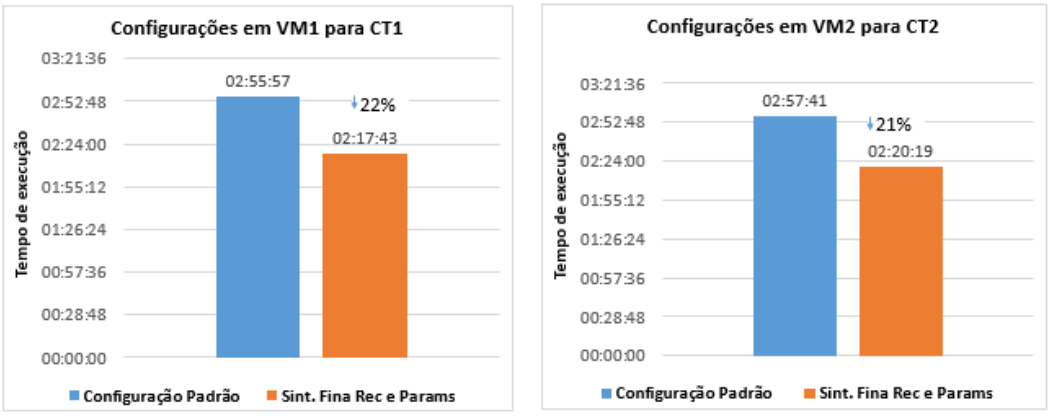

Figura 4-9: Resultado obtidos para o cenário 2

No primeiro gráfico são mostrados os resultados obtidos para execução da CT1 sobre o SGBD dentro da VM1 e no segundo gráfico são mostrados os resultados correspondentes à execução da CT2 sobre o SGBD existente na VM2. Em ambos gráficos se pode observar o TE obtido com a configuração padrão de parâmetros que tem o SGBD correspondente. Ao lado deste resultado se mostra em cada gráfico o menor TE obtido nas CTs (iteração 2), com as configurações de recursos propostas pela heurística de ajuste de recursos e o seu ajuste de parâmetros correspondente.

Observando-se a relação entre os TEs obtidos para cada resultado mostrado, se comprova que foi encontrada para cada CT uma configuração (de recursos e parâmetros) que diminui o seu TE. Isto constitui uma melhora no desempenho dos SGBD correspondentes a cada CT. Também comprova que os processos de ajuste de recursos e parâmetros propostos, podem melhorar o desempenho dos SGBDs executando-se em ambientes virtualizados.

As configurações encontradas podem ser explicadas a partir das ideias que segue a abordagem implementada neste cenário. Busca-se, entre outros objetivos, um melhor aproveitamento do recurso memória por parte de SGBDs executando-se em um ambiente virtualizado. Além disso, o processo de conseguir diminuir o maior tempo de execução entre ambas VMs foi cumprido nestes experimentos. Neste caso o TE da configuração padrão do SGBD na VM2 foi o maior TE entre ambas VMs, onde se pode ver que diminuiu no segundo gráfico da figura 4.9. Também se conseguiu

diminuir os tempos de ambas VMs de forma equilibrada. Este equilíbrio é mostrado na figura 4.9 em forma de porcentagens para cada CT, onde as porcentagens de melhorias conseguidas no segundo resultado com respeito ao primeiro resultado mostrado para ambos gráficos é na média um 21,5 %.

As melhorias obtidas no processo de sintonia fina feito com o sistema são significativas. Se comparadas com os tempos de cada uma das VM executando-se uma sem a intervenção da outra, a diferença continua sendo grande. Esta diferença também é dada pelos problemas mostrados da virtualização [3], que o processo de sintonia fina proposto não pode solucioná-los em sua totalidade, mas como se mostrou pode melhorá-los, o que constitui um de avanço no contexto estudado.

Na primeira iteração da heurística de ajuste de recursos (cenário 2) feita pelo sistema desenvolvido, trocaram-se as configurações de parâmetros encontradas tanto para a CT 1 como para a CT 2. Isto foi devido aos gargalos identificados na competição pelo acesso aos recursos por cada VM (seção 4.3). As respostas dos SGBD ante as CTs trocaram, evidenciando-se outros tempos de respostas nas configurações candidatas. O fato do sistema encontrar outras configurações de parâmetros com menor TE mostra as suas possibilidades de adaptação às mudanças que podem acontecer no ambiente virtualizado.

Comparado com os resultados obtidos no cenário 1, a configuração candidata com menor TE (cenário 2) para a CT1 fez novos ajustes de parâmetros para o seu buffer, a cache de disco e os checkpoints. Esta configuração candidata teve resultados iguais ao cenário 1 para os parâmetros de controle dos buffers temporais e dos tamanhos dos arquivos WAL. Também comparado com os resultados do cenário 1, a configuração candidata com menor TE para a CT2 (cenário 2) fez novos ajustes de parâmetros fora dos padrões do SGBD. Estes ajustes foram feitos em seu buffer e a cache de disco.

Como foram encontrados muitos acessos a blocos de memória e de forma mais demorada no buffer do SGBD, o sistema considerou testar todos os valores para o parâmetro de ajuste de buffer tanto na CT1 como na CT2. Finalmente, em ambos casos, a configuração candidata com

menor TE foi a de maior valor de ajuste de memória. Esta nova configuração é aconselhável pois além de registrar-se muita atividade de I/O, o processamento de dados foi mais lento, obrigando o SGBD a responder melhor com maior alocação de memória. Como esta alocação às leituras e escritas em disco diminuíram, o SGBD teve melhor desempenho e baixou o TE das consultas.

O parâmetro de ajuste da cache de disco para ambas CT foi alocado em correspondência com os valores alocados no buffer dos SGBDs onde se executaram estas CTs. Para o SGBD que executou a CT1 foi ajustado ao parâmetro cache de disco um valor superior a 10% do mesmo parâmetro do SGBD que executou a CT2. Isto foi devido a que a CT2 teve menor acesso a blocos de memória que a CT1. Neste caso, o SGBD que executou a CT2 respondeu com melhor TE ante um ajuste de cache de disco menor que o caso do SGBD que executou a CT1.

A configuração com menor TE (CT1) também teve um ajuste fora do padrão com os parâmetros que controlam o tempo dos *checkpoints* e os *vacuums*. O parâmetro para controle de tempo dos checkpoints, ajustado para aumentar a quantidade de tempo em que se faziam os pontos de recuperação no SGBD, propiciou a minimização da atividade de E/S extra que levava fazer esta atividade. Com esta ação se diminuiu o TE para a CT1. Para o parâmetro de controle dos *vacuums* foi feito um ajuste que atrasou o tempo em fazer-se os *vacuums*. Com esta operação também se propiciou a minimização da atividade extra de E/S no SGBD e, assim, se contribuiu a diminuição do TE da CT1. Na execução da CT2 foram mantidos estes parâmetros com os seus ajustes padrões, devido a que foi feita menor quantidade de operações de E/S que na CT1.

Na segunda iteração dos testes foi aumentada a alocação de memória para ambas VMs mediante a utilização da heurística de recursos. O modelo de distribuição dos recursos, após ser aplicado o modelo de custo proposto nessa heurística, aumentou mais a alocação de memória para cada VM. Isto aconteceu porque tinha-se maior quantidade de memória disponível no host virtualizado. Além disso, a alocação de memória para a VM1 foi maior que a alocação da VM2, o que é natural porque na VM1 houve maior quantidade acesso a blocos de memória que na VM2.

Nesta iteração a percentagem de ajuste de memória no buffer dos SGBDs para as CT 1 e 2 foi mantida, mas devido à maior alocação de memória os resultados em TE foram muito menores do que os obtidos na iteração 1. A mesma coisa aconteceu no uso dos buffers temporários e o ajuste de cache de disco. A configuração para a CT2 deu menor resultado com a troca do valor parâmetro de cache de disco. Este ajuste aumentou em 10% com respeito ao valor ajustado na iteração 1. Esta configuração foi possível porque foi a aumentada a quantidade de memória alocada, obtendo-se o equilíbrio entre a memória alocada para o SGBD e o sistema operacional. A partir disso, o SGBD teve menor TE para esta configuração.

Tanto no cenário 1 como o cenário 2, a execução das cargas de trabalho sobre maior quantidade de dados pode dar melhores resultados que os atuais. Isto é devido a que vai existir maior probabilidade que os SGBDs adotem configurações que propiciem maior uso do recurso memória, em correspondência ao aumento da quantidade de dados a processar. Também outras cargas de trabalho mais intensivas em atualizações e leituras podem ser testadas com o sistema. Neste caso aumentaria a quantidade de operações de escrita em disco, com respeito às CT utilizadas que são mais intensivas nas operações de leitura.

4.5. Conclusões

Neste capítulo foi detalhado para os dois cenários definidos o desenho arquitetônico do sistema, tomando-se como base como se implementaram e estenderam os principais componentes sobre o framework DBX. Além disso, se apresentou as principais peculiaridades do contexto de SGBDs executando-se em ambientes virtualizados através de um processo de análise experimental. Neste processo se identificaram limites do contexto estudado que permitiram conhecer as suas principais características e funcionamento. Nos testes realizados em cada cenário se descreveram os principais resultados obtidos, cumprindo-se com eles os objetivos identificados na pesquisa. Também foi comprovado que se pode

fazer sintonia fina nestes cenários, obtendo-se resultados benéficos no desempenho dos SGBDs.

5
Considerações Finais

No presente capítulo apresentam-se os principais resultados e repercussões obtidos nesta dissertação de mestrado, além dos trabalhos futuros a serem desenvolvidos.

Como principal conclusão este trabalho apresenta uma forma de fazer sintonia fina no contexto de SGBDs executando-se em ambientes de virtualização privados, considerando os principais problemas existentes. Os resultados mostrados provam que se pode fazer sintonia fina, obtendo-se melhoras de desempenho de SBDs nos cenários identificados.

Foi demonstrado que a heurística de ajuste de parâmetros permite encontrar menores tempos de execução (TE) baixo certos parâmetros de SGBD, que os TE que trazem as configurações padrões de parâmetros dos SGBDs. Estes tempos de execução foram obtidos analizando os resultados das cargas de trabalho executadas sobre cada SGBD e os recursos alocados às VM onde se executaram os SGBD.

Foi mostrado também que a atualização do SGBD sobre os recursos alocados à virtualização mediante seus parâmetros de configuração, permitem ao SBD melhorar seu desempenho. Isto foi comprovado com os resultados obtidos com a heurística de ajuste de recursos.

5.1.
Contribuições

Neste trabalho de pesquisa foram criadas duas heurísticas que permitem realizar a sintonia fina de SGBDs em ambientes virtualizados. Estes procedimentos constituem uma ferramenta importante para obter configurações com alocações de recursos de virtualização e ajustes de parâmetros de SGBD, que permitem melhoras de desempenho aos SBD.

A heurística de ajuste de parâmetros cria configurações de parâmetros candidatas a ser testadas com o analises da CT e um conjunto

de valores de entrada para estes parâmetros. Estas configurações candidatas contam com os recursos alocados na virtualização para atualizar ao SGBD da existência deles, conseguindo-se melhoras no desempenho dos SGBDs.

A heurística de ajuste de recursos permite fazer ajustes dos recursos da virtualização tomando as necessidades existente nas VMs, a partir do número de operações de E/S que em estas se fazem. Para conseguir isto utiliza o modelo de custo e o modelo de distribuição de recursos criados nesta pesquisa. O modelo de custo permite calcular custos por CT e VMs a partir das operações de E/S feitas com as CTs. O modelo de distribuição de recursos faz a distribuição dos recursos a partir dos resultados encontrados com o modelo de custo, em benefício de aquelas VMs que os precisem mais.

Este trabalho faz uma exploração experimental para conhecer as principais características do contexto do problema. A documentação destes experimentos pode servir de guia para conhecer melhor no contexto estudado quais são os principais gargalhos encontrados, que conselhos se devem seguir e que limites existem.

Esta pesquisa seleciona quais parâmetros de configuração de SGBDs podem impactar na atualização do SGBD da existência de recursos no ambiente virtualizado, com alvo de diminuir as leituras e escritas ocasionadas em disco pelo SGBD. A descrição destas ideias junto com os resultados experimentais obtidos no ajuste destes parâmetros pode servir de guia no processo de sintonia fina realizado.

Finalmente foi estendido o framework DBX para poder fazer sintonia fina ajuste de recursos da virtualização e de parâmetros do SGBD no contexto do problema estudado. Em DBX se implementaram as heurísticas que apoiam estes processos. Foi estendido neste framework o analises da carga de trabalho em função de identificar funcionalidades que impactem no processo de sintonia fina de ajuste de parâmetros de SGBD.

Outras implementações adicionadas permitem a interação de DBX de forma não intrusiva com o VMM e as VMs do sistema de virtualização privado, seguindo o objetivo de garantir o ajuste de recursos da

virtualização e parâmetros do SGBD necessários no processo de sintonia fina realizado.

5.2.
Trabalhos futuros

São mostrados a continuação possíveis oportunidades de pesquisas e os próximos passos na evolução da pesquisa:

- Testar as funcionalidades do sistema feitas para responder ao cenário 1 num ambiente não virtualizado.
- Estender a interface de DBX para exibir mais informações das configurações candidatas e seu processo de teste sobre a carga de trabalho.
- Estudo do impacto de parâmetros de configuração de forma individual no desempenho de SBD em ambientes de virtualização privados.
- Estender o trabalho de sintonia fina realizado ao contexto de SGBDs que funcionem na nuvem, considerando que os ambientes virtualizados são parte importante entre as tecnologias utilizadas na nuvem.
- Estender as heurísticas criadas a realização de sintonia fina em SGBDs que funcionem em ambientes de virtualização de sistema operacionais [22], [44].

6
Referências bibliográficas

[1] A. Aboulnaga, C. Amza, and K. Salem, "Virtualization and databases: state of the art and research challenges," in *{EDBT} 2008, 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, 2008, Proceedings*, 2008, vol. 261, pp. 746–747.

[2] M. Stonebraker, A. Pavlo, R. Taft, and M. L. Brodie, "Enterprise Database Applications and the Cloud: {A} Difficult Road Ahead," in *2014 {IEEE} International Conference on Cloud Engineering, Boston, MA, USA, March 11-14, 2014*, 2014, pp. 1–6.

[3] S. Gamage, C. Xu, R. R. Kompella, and D. Xu, "vPipe: Piped {I/O} Offloading for Efficient Data Movement in Virtualized Clouds," in *Proceedings of the {ACM} Symposium on Cloud Computing, Seattle, WA, USA, November 03 - 05, 2014*, 2014, p. 27:1--27:13.

[4] C. Xu, S. Gamage, H. Lu, R. R. Kompella, and D. Xu, "vTurbo: Accelerating Virtual Machine {I/O} Processing Using Designated Turbo-Sliced Core," in *2013 {USENIX} Annual Technical Conference, San Jose, CA, USA, June 26-28, 2013*, 2013, pp. 243–254.

[5] E.-E. A. Durham, A. Rosen, and R. W. Harrison, "Optimization of relational database usage involving Big Data a model architecture for Big Data applications," in *2014 {IEEE} Symposium on Computational Intelligence and Data Mining, {CIDM} 2014, Orlando, FL, USA, December 9-12, 2014*, 2014, pp. 454–462.

[6] D. Y. Yoon, N. Niu, and B. Mozafari, "DBSherlock: {A} Performance Diagnostic Tool for Transactional Databases," in *Proceedings of the 2016 International Conference on Management of Data, {SIGMOD} Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, 2016, pp. 1599–1614.

[7] D. Shasha and P. Bonnet, *Database Tuning: Principles, Experiments, and Troubleshooting Techniques*. San Francisco, CA, USA: Morgan Kaufmann

Publishers Inc., 2003.

[8] L. Chunlin and L. Li, “Optimal scheduling across public and private clouds in complex hybrid cloud environment,” *Inf. Syst. Front.*, vol. 19, no. 1, pp. 1–12, 2017.

[9] T. Mühlbauer, W. Rödiger, A. Kipf, A. Kemper, and T. Neumann, “High-Performance Main-Memory Database Systems and Modern Virtualization: Friends or Foes?,” in *Proceedings of the Fourth Workshop on Data analytics in the Cloud, DanaC 2015, Melbourne, VIC, Australia, May 31 - June 4, 2015*, 2015, p. 4:1--4:4.

[10] D. A. Menascé, “Virtualization: Concepts, Applications, and Performance Modeling,” in *31th International Computer Measurement Group Conference, December 4-9, 2005, Orlando, Florida, USA, Proceedings*, 2005, pp. 407–414.

[11] G. J. Popek and R. P. Goldberg, “Formal Requirements for Virtualizable Third Generation Architectures,” *Commun. {ACM}*, vol. 17, no. 7, pp. 412–421, 1974.

[12] A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, and S. Kamath, “Automatic virtual machine configuration for database workloads,” in *Proceedings of the {ACM} {SIGMOD} International Conference on Management of Data, {SIGMOD} 2008, Vancouver, BC, Canada, June 10-12, 2008*, 2008, pp. 953–966.

[13] M. Rosenblum and T. Garfinkel, “Virtual Machine Monitors: Current Technology and Future Trends,” *{IEEE} Comput.*, vol. 38, no. 5, pp. 39–47, 2005.

[14] J. E. Smith and R. Nair, “The Architecture of Virtual Machines,” *{IEEE} Comput.*, vol. 38, no. 5, pp. 32–38, 2005.

[15] V. Timcenko, B. Djordjevic, S. V. B. Rakas, and N. Davidovic, “Performance examination of type-2 hypervisors: case of particular database application in a virtual environment,” in *{ISDOC} 2014 - International Conference on Information Systems and Design of Communication,*

{ISDOC} 2014, Lisbon, Portugal, May 16-17, 2014, 2014, pp. 122–126.

[16] IBM, “IBM,” 2017. [Online]. Available: https://www.ibm.com.

[17] VMWare, “VMWare Web Site,” 2016. [Online]. Available: http://www.vmware.com/.

[18] P. Barham *et al.*, “Xen and the art of virtualization,” in *Proceedings of the 19th {ACM} Symposium on Operating Systems Principles 2003, {SOSP} 2003, Bolton Landing, NY, USA, October 19-22, 2003*, 2003, pp. 164–177.

[19] XenProject, “XenProject Web Site,” 2016. [Online]. Available: https://www.xenproject.org/.

[20] A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, and S. Kamath, “Automatic virtual machine configuration for database workloads,” *{ACM} Trans. Database Syst.*, vol. 35, no. 1, p. 7:1--7:47, 2010.

[21] L. Cherkasova, D. Gupta, and A. Vahdat, “Comparison of the three CPU schedulers in Xen,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 2, pp. 42–51, 2007.

[22] D. Bernstein, “Containers and Cloud: From {LXC} to Docker to Kubernetes,” *{IEEE} Cloud Comput.*, vol. 1, no. 3, pp. 81–84, 2014.

[23] L. Cherkasova, D. Gupta, and A. Vahdat, “Comparison of the three {CPU} schedulers in Xen,” *{SIGMETRICS} Perform. Eval. Rev.*, vol. 35, no. 2, pp. 42–51, 2007.

[24] S. Duan, V. Thummala, and S. Babu, “Tuning Database Configuration Parameters with iTuned,” *PVLDB*, vol. 2, no. 1, pp. 1246–1257, 2009.

[25] B. K. Debnath, D. J. Lilja, and M. F. Mokbel, “SARD: A statistical approach for ranking database tuning parameters,” *Proc. - Int. Conf. Data Eng.*, pp. 11–18, 2008.

[26] DB2-Configurator-Advisor, “DB2 Configurator Advisor,” 2017. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSEPGG_11.1.0/com.ibm.

db2.luw.admin.dbobj.doc/doc/c0052481.html.

[27] PostgreSQL-tuning-Wizard, “PostgreSQL tuning wizard,” 2017. [Online]. Available: http://pgfoundry.org/projects/pgtune.

[28] V. Thummala and S. Babu, “iTuned: a tool for configuring and visualizing database parameters,” *Proc. 2010 Int. Conf. Manag. data - SIGMOD ’10*, p. 1231, 2010.

[29] V. Thummala and S. Babu, “iTuned: a tool for configuring and visualizing database parameters,” in *Proceedings of the {ACM} {SIGMOD} International Conference on Management of Data, {SIGMOD} 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, 2010, pp. 1231–1234.

[30] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts - 6th. ed.*, vol. 4. 2011.

[31] P. B. D. Shasha, *Database Tuning: Principles, Experiments, and Troubleshooting Techniques.*, vol. 2002. 2002.

[32] P. Web, “PostgreSQL checkpoints and WAL,” 2017. [Online]. Available: https://www.postgresql.org/docs/9.5/static/wal-configuration.html.

[33] P. Web, “PostgreSQL Wal Parameters,” 2017. [Online]. Available: https://www.postgresql.org/docs/9.5/static/runtime-config-wal.html.

[34] Sql. Web, “SQLite WAL,” 2017. [Online]. Available: https://sqlite.org/wal.html.

[35] M. Stonebraker, A. Pavlo, R. Taft, and M. L. Brodie, “Enterprise database applications and the cloud: A difficult road ahead,” *Proc. - 2014 IEEE Int. Conf. Cloud Eng. IC2E 2014*, pp. 1–6, 2014.

[36] A. A. Soror, A. Aboulnaga, and K. Salem, “Database virtualization: A new frontier for database tuning and physical design,” in *Proceedings - International Conference on Data Engineering*, 2007, pp. 387–394.

[37] postgresql-cpu-parameters, “PostgreSQL CPU configuration parameters,” 2017. [Online]. Available:

https://www.postgresql.org/docs/9.5/static/runtime-config-query.html.

[38] XenProject, “Xen Wiki Best Practices,” 2016. [Online]. Available: https://wiki.xenproject.org/wiki/Xen_Project_Best_Practices.

[39] PostgreSQL, “Runtime config parameters for PostgreSQL.,” 2016. [Online]. Available: https://www.postgresql.org/docs/9.5/static/runtime-config-resource.html.

[40] PostgreSQL, “How to use PostgreSQL explain analyse buffers,” 2016. [Online]. Available: https://www.postgresql.org/docs/9.6/static/using-explain.html.

[41] H. Guan, R. Ma, and J. Li, “Workload-Aware Credit Scheduler for Improving Network I/O Performance in Virtualization Environment,” *IEEE Trans. Cloud Comput.*, vol. 7161, no. c, pp. 1–1, 2014.

[42] R. P. O. Almeida, Ana Carolina, Angelo Brayner, José Maria Monteiro, Sérgio Lifschitz, “DBX: Um Framework para Auto-sintonia Fina Baseado em Planos Hipotéticos.,” in *Simpósio Brasileiro de Bancos de Dados - Demos and Applications Session*, 2015.

[43] G. Weikum, C. Hasse, A. Moenkeberg, and P. Zabback, “The {COMFORT} Automatic Tuning Project, Invited Project Review,” *Inf. Syst.*, vol. 19, no. 5, pp. 381–432, 1994.

[44] C. Anderson, “Docker,” *{IEEE} Softw.*, vol. 32, no. 3, p. 102, 2015.

7
Anexos

Anexo 1: Detalhes das cargas de trabalho utilizadas.

Bechmarck utilizado: TPC-H.

Instanciado TPC-H para obter banco de dados tamanho em disco 10GB por cada SGBD nas VMs. A distribuição dos dados nas tabelas se pode ver na tabela 4.1.

Tabela 7-1: Informação dos dados das tabelas do esquema TCP-H (10GB).

Nome	Quantidades de tuplas	Tamanho
Customer	450.000	87 MB
Lineitem	17.996.600	2.848 MB
nation	25	8.192 bytes
Orders	4.500.000	646 MB
Part	600.000	96 MB
Partsupp	2.399.940	427 MB
region	5	8.192 bytes
Supplier	30.000	5.512 bytes

Consultas utilizadas de TPC-H por cada carga de trabalho:

- Carga de trabalho 1: {1, 3, 4, 5, 10, 12, 18}
- Carga de trabalho 2: {2, 11, 13, 16, 19, 20}
- Carga de trabalho 3: {1, 3, 5, 11, 13, 16, 19, 22}

Anexo 2: Configurações encontradas em ajuste de parâmetros.

Sistemas de Gerenciamento de Banco de Dados usado no teste: PostgreSQL 9.5.

Hypervisor usado no teste: Xen.

Memória alocada a cada VM:

- VM1: RAM 4GB.
- VM2: RAM 4GB.
- VM3: RAM 4GB.

Memória total máquina virtualizada: 16GB.

Na tabela 4.2 se mostram as configurações encontradas com menor tempo de execução por carga de trabalho nos diferentes SGBDs executando-se nas respetivas VMs. O menor tempo é medido respeito ao tempo de execução da configuração padrão dos SGBDs testados, ambos tempos são mostrados na figura 4.8 da seção 4.4.1. Os valores dos parâmetros mostrados em negrito foram ajustes fora do ajuste padrão do SGBD, no caso contrário são os ajustes padrão do SGBD utilizado.

Tabela 7-2 Configuração de parâmetros com menor tempo de execução encontrada.

Parâmetro	CT1-VM1	CT2-VM2	CT3-VM3
shared_buffers	128MB	128MB	**1.6GB (40% da RAM total)**
effective_cache-size	100%	100%	**2GB (50% da RAM total)**
work_mem	**40MB (1% da RAM total**	**40MB (1% da RAM total**	**40MB (1% da RAM total**
checkpoint_completion_target	0,5	0,5	0,5
min_wal-size	80MB	80MB	80MB
max_wal_size	**2GB**	1GB	1GB
vacuum_cost_delay	0	0	0

Anexo 3: Configurações encontradas em ajuste de recursos.

Sistemas de Gerenciamento de Banco de Dados usado no teste: PostgreSQL 9.5.

Hypervisor usado no teste: Xen.

Memória total máquina virtualizada: 16GB.

Iteração 1:

Na tabela 4.4 são mostradas as alocações de recursos (RAM e tCPU) feitas para cada VM onde foi feito o teste:

Tabela 7-3: Alocação de recursos de cada VM.

Recurso Alocado	VM1	VM2
RAM	**4GB**	**4GB**
tCPU	**1000 (100ms)**	**1000 (100ms)**

Na tabela 4.5 se mostram as configurações encontradas com menor tempo de execução por carga de trabalho nos diferentes SGBDs executando-se nas respetivas VMs de forma simultânea. O menor tempo é medido respeito ao tempo de execução da configuração padrão. Os valores dos parâmetros mostrados em negrito foram ajustes fora do ajuste padrão do SGBD, no caso contrário são os ajustes padrão do SGBD utilizado.

Tabela 7-4 Configuração de parâmetros com menor tempo de execução encontrada.

Parâmetro	CT1-VM1	CT2-VM2
shared_buffers	**40% da RAM total**	**40% da RAM total**
effective_cache-size	**50% da RAM total**	**40% da RAM total**
work_mem	**1% da RAM total**	**1% da RAM total**
checkpoint_completion_target	**0,9**	0,5

min_wal-size	80MB	80MB
max_wal_size	**2GB**	1GB
vacuum_cost_delay	**10ms**	0

Iteração 2:

Na tabela 4.6 são mostradas as alocações de recursos (RAM e tCPU) feitas para cada VM onde foi feito o teste:

Tabela 7-5: Alocação de recursos de cada VM.

Recurso Alocado	**VM1**	**VM2**
RAM	**6140 MB**	**5119 MB**
tCPU	**2000 (200ms)**	**1400 (140ms)**

Na tabela 4.7 se mostram as configurações encontradas com menor tempo de execução por carga de trabalho nos diferentes SGBDs executando-se nas respetivas VMs de forma simultânea. O menor tempo é medido respeito ao tempo de execução da configuração padrão dos SGBDs testados. Tanto o tempo de execução padrão como o tempo de execução desta iteração é mostrado na figura 4.9 da seção 4.4.2. Os valores dos parâmetros mostrados em negrito foram ajustes fora do ajuste padrão do SGBD, no caso contrário são os ajustes padrão do SGBD utilizado.

Tabela 7-6: Configuração de parâmetros com menor tempo de execução encontrada.

Parâmetro	**CT1-VM1**	**CT2-VM2**

shared_buffers	**40% da RAM total**	**40% da RAM total**
effective_cache-size	**50% da RAM total**	**50% da RAM total**
work_mem	**1% da RAM total**	**1% da RAM total**
checkpoint_completion_target	**0,9**	0,5
min_wal-size	80MB	80MB
max_wal_size	**2GB**	1GB
vacuum_cost_delay	**10ms**	0
Menor Tempo/ Tempo configuração padrão	*735.591s / 10534,2s*	***459.838s** / 10644,6s*