Universidade Federal do Ceará
Centro de Ciências
Pró-Reitoria de Pesquisa e Pós-Graduação
Mestrado e Doutorado em Ciência da Computação

ABORDAGEM PARA QUALIDADE DE SERVIÇO EM BANCOS DE DADOS MULTI-INQUILINOS EM NUVEM

Leonardo Oliveira Moreira

TESE DE DOUTORADO

Fortaleza
Julho - 2014

Universidade Federal do Ceará
Centro de Ciências
Pró-Reitoria de Pesquisa e Pós-Graduação

Leonardo Oliveira Moreira

ABORDAGEM PARA QUALIDADE DE SERVIÇO EM BANCOS DE DADOS MULTI-INQUILINOS EM NUVEM

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará como requisito parcial para o título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Javam de Castro Machado

Fortaleza
Julho - 2014

Leonardo Oliveira Moreira

Abordagem para Qualidade de Serviço em Bancos de Dados Multi-Inquilinos em Nuvem

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação, da Universidade Federal do Ceará, como requisito para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 25 de julho 2014

BANCA EXAMINADORA

Prof. Dr. Javam de Castro Machado
Orientador
Universidade Federal do Ceará – UFC

Prof. Dr. José Maria da Silva Monteiro Filho
Universidade Federal do Ceará – UFC

Prof. Dr. Flávio Rubens de Carvalho Sousa
Universidade Federal do Ceará – UFC

Profa. Dra. Fernanda Araújo Baião Amorim
Universidade Federal do Estado do Rio de Janeiro – UNIRIO

Prof. Dr. Sérgio Lifschitz
Potintifícia Universidade Católica do Rio de Janeiro – PUC/RJ

Fortaleza, 25 de julho de 2014

Aos Meus Pais.

AGRADECIMENTOS

A Deus, por estar sempre ao meu lado, dando-me coragem para enfrentar todos os obstáculos da vida.

A minha mãe, Fátima, e ao meu pai, Maurício, exemplos de honestidade e dignidade, que sempre incentivaram e apoiaram a minha educação.

Ao meu irmão e melhor amigo, Maurício Moreira Neto, por compartilhar comigo todos os momentos alegres e difíceis, dando-me força para enfrentar a vida.

Ao meu orientador, Prof. Dr. Javam de Castro Machado, que, com sua competência teórica, auxiliou no delineamento e sistematização do trabalho. Muito obrigado pela amizade, paciência e orientação ao longo de todos estes anos.

Ao meu tio Prof. Dr. Rui Verlaine Oliveira Moreira, exemplo de docente, pesquisador, orientador e pessoa. Grato por me apoiar e ajudar nesta carreira acadêmica. Sem o senhor isso tudo não poderia ser realizado.

Ao Prof. Dr. José Gilvan Rodrigues Maia, pela paciência, amizade, motivação e auxílio, e por ser tão presente para que este trabalho se concretizasse. Obrigado, também, por não me deixar desistir desta pesquisa durante os momentos difíceis.

Ao Prof. Dr. Flávio Rubens de Carvalho Sousa, pelo auxílio constante, pelas recomendações bibliográficas, revisões e sugestões para o aprimoramento desta investigação. Obrigado pela sua valiosa amizade desde a época do mestrado.

Ao amigo Gustavo Adolfo Campos dos Santos, pela amizade e momentos de descontração; amizade que cultivamos desde a época do mestrado. Agradeço também pelas revisões, traduções e sugestões durante toda a fase de estruturação do texto desta tese.

À amiga Natália Ivo, não só pela amizade, mas também por ter haver ajudado, motivado e ter sido uma excelente companheira.

Aos colegas do laboratório BIMO, em especial, Antônio Arleudo, Guilherme Moraes, Moises Rodrigues, Inaciane Monteiro e Tibério Freire, pela amizade, presença e apoio.

Ao amigo Victor Aguiar Evangelista de Farias, pela constante ajuda no desenvolvimento e revisão deste trabalho. Obrigado pela sua paciência e presteza, demonstradas durante

todo o ensaio.

À equipe do OLTPBenchmark, especialmente a Carlo Curino (Microsoft Research), pela colaboração e atenção dedicadas a esta tese.

Aos colegas professores do Instituto Universidade Virtual (IUVI/UFC), pelo apoio, motivação e bom convívio, que propiciaram um ambiente favorável e agradável para o desenvolvimento deste experimento.

Ao diretor do IUVI/UFC, Prof. Dr. Mauro Cavalcante Pequeno, pelo apoio para a realização deste doutorado.

Aos colegas do Mestrado e Doutorado, pela excelente relação pessoal que criamos e espero que não se perca. Em especial, a Ticiana Linhares, Lívia Almada, Paulo Rêgo, Rodrigo Félix, Camila Ferreira, David Araújo, Jeovane Reges, Ricardo Honorato, Vinícius Pires, Regis Pires, Macedo Maia e Antônio Cavalcante, pelo profissionalismo exemplar, pelo apoio nos momentos bons e menos bons e pelas suas amizades.

Aos meus componentes familiares, que sempre me deram amor e força, valorizando meus potenciais.

A todos os que, direta ou indiretamente, contribuíram para a realização deste trabalho.

If I have seen further it is by
standing on the shoulders of Giants
—ISAAC NEWTON

RESUMO

A computação em nuvens é um paradigma bem consolidado de utilização de recursos computacionais, segundo o qual infraestrutura de hardware, software e plataformas para o desenvolvimento de novas aplicações são oferecidos como serviços disponíveis remotamente e em escala global. Os usuários de nuvens computacionais abrem mão de uma infraestrutura computacional própria para dispô-la mediante serviços oferecidos por provedores de nuvem, delegando aspectos de Qualidade de Serviço (QoS) e assumindo custos proporcionais à quantidade de recursos que utilizam modelo de pagamento baseado no uso. Essas garantias de QoS são definidas entre o provedor do serviço e o usuário, e expressas por meio de Acordo de Nível de Serviço (SLA), o qual consiste de contratos que especificam um nível de qualidade a ser atendido, e penalidades em caso de falha. A maioria das aplicações em nuvem é orientada a dados e, por conta disso, Sistemas Gerenciadores de Banco de Dados (SGBDs) são candidatos potenciais para a implantação em nuvem. SGBDs em nuvem devem tratar uma grande quantidade de aplicações ou inquilinos. Os modelos de multi-inquilinatos são utilizados para consolidar vários inquilinos dentro de um só SGBD, favorecendo o compartilhamento eficaz de recursos, além de gerenciar uma grande quantidade de inquilinos com padrões de carga de trabalho irregulares. Por outro lado, os provedores em nuvem devem reduzir os custos operacionais, garantindo a qualidade. Para muitas aplicações, o maior tempo gasto no processamento das requisições está relacionado ao tempo de execução do SGBD. Portanto, torna-se importante que um modelo de qualidade seja aplicado ao SGBD para seu desempenho. Técnicas de provisionamento dinâmico são voltadas para o tratamento de cargas de trabalho irregulares, para que violações de SLA sejam evitadas. Sendo assim, uma estratégia para ajustar a nuvem no momento em que se prevê um comportamento que pode violar o SLA de um dado inquilino (banco de dados) deve ser considerada. As técnicas de alocação são usadas no intuito de aproveitar os recursos do ambiente em detrimento ao provisionamento. Com base nos sistemas de monitoramento e de modelos de otimização, as técnicas de alocação decidem onde será o melhor local para receber um dado inquilino. Para realizar a transferência do inquilino de forma eficiente, técnicas de *Live Migration* são adotadas para ter o mínimo de interrupção do serviço. Acredita-se que a combinação destas três técnicas podem contribuir para o desenvolvimento de um solução robusta de QoS para bancos de dados em nuvem, minimizando violações de SLA. Ante tais desafios,

esta tese apresenta uma abordagem, denominada PMDB, para melhorar QoS em SGBDs multi-inquilinos em nuvem. A abordagem tem como objetivo reduzir o número de violações de SLA e aproveitar os recursos à disposição por meio de técnicas que realizam predição de carga de trabalho, alocação e migração de inquilinos quando necessitam de recursos com maior capacidade. Para isso, uma arquitetura foi proposta e um protótipo implementado com tais técnicas, além de estratégias de monitoramento e QoS voltada para aplicações de banco de dados em nuvem. Ademais, alguns experimentos orientados a desempenho foram especificados para mostrar a eficiência da abordagem a fim de alcançar o objetivo em foco.

Palavras-chave: Computação em Nuvem. Qualidade de Serviço. Multi-Inquilino. Gerenciamento de Dados.

ABSTRACT

Cloud computing is a well-established paradigm of computing resources usage, whereby hardware infrastructure, software and platforms for the development of new applications are offered as services available remotely and globally. Cloud computing users give up their own infrastructure to dispose of it through the services offered by cloud providers, to which they delegate aspects of Quality of Service (QoS) and assume costs proportional to the amount of resources they use, which is based on a payment model. These QoS guarantees are established between the service provider and the user, and are expressed through Service Level Agreements (SLA). This agreement consists of contracts that specify a level of quality that must be met, and penalties in case of failure. The majority of cloud applications are data-driven, and thus Database Management Systems (DBMSs) are potential candidates for cloud deployment. Cloud DBMS should treat a wide range of applications or tenants. Multi-tenant models have been used to consolidate multiple tenants within a single DBMS, favoring the efficient sharing of resources, and to manage a large number of tenants with irregular workload patterns. On the other hand, cloud providers must be able to reduce operational costs while keeping quality levels as agreed. To many applications, the longer time spent in processing requests is related to the DBMS runtime. Therefore, it becomes important to apply a quality model to obtain DBMS performance. Dynamic provisioning techniques are geared to treat irregular workloads so that SLA violations are avoided. Therefore, it is necessary to adopt a strategy to adjust the cloud at the time a behavior that may violate the SLA of a given tenant (database) is predicted. The allocation techniques are applied in order to utilize the resources of the environment to the dentriment of provisioning. Based on both the monitoring and the optimization models systems, the allocation techniques will decide the best place to assign a given tenant to. In order to efficiently perform the transfer of the tenant, minimal service interruption, Live Migration techniques are adopted. It is believed that the combination of these three techniques may contribute to the development of a robust QoS solution to cloud databases which minimizes SLA violations. Faced with these challenges, this thesis proposes an approach, called PMDB, to improve DBMS QoS in multi-tenant cloud. The approach aims to reduce the number of SLA violations and take advantage the resources that are available using techniques that perform workload prediction, allocation and mi-

gration of tenants when greater capacity resources are needed. An architecture was then proposed and a prototype implementing such techniques was developed, besides monitoring strategies and QoS oriented database applications in the cloud. Some performance oriented experiments were then specified to show the effectiveness of our approach.

Keywords: Cloud Computing. Quality of Service. Multi-Tenancy. Data Management.

SUMÁRIO

LISTA DE ABREVIATURAS

ARIMA - Auto Regressive Integrated Moving-Average

AWS - Amazon Web Services

API - Application Programming Interface

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

BD - Banco de Dados

DaaS - Sistemas de Bancos de Dados como um Serviço

EC2 - Elastic Compute Cloud

FEQ - Função de Estimativa da Qualidade

GMR - Greedy Memory Reduction

GTC - Greedy Tenant Consolidation

IaaS - Infrastructure as a Service

IO - Input/Output

JDBC - Java Database Connectivity

MDP - Markov Decision Process

MAPE - Mean Absolute Percentage Error

MRE - Mean Relative Error

MVCC - Multi-Version Concurrency Control

NAS - Network-Attached Storage

OLAP - On-line Analytical Processing

OLTP - On-line Transaction Processing

PaaS - Platform as a Service

PMDB - Performance Multi-tenant Databases

QoS - Qualidade de Serviço

QoSDBC - Quality of Service for Database in the Cloud

RMSE - Root Mean Squared Error

SaaS - Software as a Service

SGBD - Sistema de Gerenciamento de Banco de Dados

SGBDR - Sistema de Gerenciamento de Banco de Dados Relacional

SLA - Acordo de Nível de Serviço

SLADB - Acordo de Nível de Serviço para Banco de Dados

SLO - Objetivos de Nível de Serviço

SQL - Structured Query Language

SV - Support Vector

SVM - Support Vector Machines

SVR - Support Vectors for Regression

TI - Tecnologia da Informação

VM - Máquina Virtual

XML - eXtensible Markup Language

2PC - Two-Phase Commit

2PL - Two-Phase Locking

S2PL - Strict Two-Phase Locking

SO - Sistema Operacional

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

Esta tese apresenta PMDB, uma abordagem para serviços de dados relacionais multi-inquilinos em nuvem, cujo propósito é garantir a qualidade do serviço de dados e a utilização dos recursos de infraestrutura por meio das técnicas de migração, alocação e predição em ambientes multi-inquilinos. A abordagem permite o provisionamento de recursos de forma dinâmica para manter a qualidade do serviço. Além disso, utiliza técnicas de predição para antecipar a migração do inquilino ou provisionamento de recursos a fim de reduzir as violações de SLA.

Neste capítulo são apresentadas a justificativa e a motivação para o desenvolvimento do trabalho, uma descrição do problema tratado, assim como os objetivos e as contribuições. Ao final do capítulo, é descrito como está organizado o restante da tese.

1.1 MOTIVAÇÃO

Computação em nuvem é uma tendência recente de tecnologia, cujo objetivo é proporcionar serviços de Tecnologia da Informação (TI) sob demanda com pagamento baseado no uso [Agrawal et al., 2010]. Características essenciais, como elasticidade e pagamento baseado no uso e economia em larga escala, são as maiores razões pelo sucesso e ampla adoção de infraestruturas em nuvem [Sousa et al., 2009]. Ambientes em nuvem são fundamentados em técnicas de virtualização para melhorar a utilização dos recursos. Neste ambiente, cada máquina física comporta um número variável de máquinas virtuais (VM), de acordo com a capacidade de *hardware* disponível na máquina física. Na nuvem, o usuário do serviço tem algumas garantias, como desempenho e disponibilidade. Essas garantias de qualidade do serviço (QoS) são definidas entre o provedor do serviço e o usuário, e expressas por meio de acordo de nível de serviço (SLA) [Sousa et al., 2010]. Tal acordo consiste de contratos que especificam um nível de qualidade que deve ser atendido, e penalidades em caso de falha. Muitas empresas dependem de SLA, por exemplo, para exibir uma página *web* dentro de um determinado intervalo de tempo. As empresas

esperam que provedores de nuvem forneçam garantias de qualidade utilizando SLAs com base em características de desempenho.

A maioria das aplicações em nuvem é orientada a dados, e por conta disso, Sistemas Gerenciadores de Banco de Dados (SGBDs) são candidatos potenciais para a implantação em nuvem [Elmore et al., 2011b]. Outro motivo é porque, em geral, as instalações destes sistemas são complexas e envolvem uma grande quantidade de dados, ocasionando custo elevado, tanto em *hardware* quanto em *software*. Para muitas empresas, especialmente para *startups* e médias empresas, o pagamento baseado no uso do modelo de computação em nuvem, juntamente com o suporte para manutenção do *hardware*, é muito atraente [Abadi, 2009]. É fundamental que provedores de nuvem ofereçam SLAs com base no desempenho para aplicações de seus clientes [Schad et al., 2010]. Para muitas aplicações, o maior tempo gasto no processamento das requisições está relacionado ao tempo de execução do SGBD. Portanto, torna-se importante que um modelo de qualidade seja aplicado ao SGBD para seu desempenho [Moreira et al., 2012].

Para melhorar o gerenciamento dos recursos e reduzir custos, os provedores implementam o compartilhamento de recursos entre inquilinos [Barker et al., 2012]. O conceito de multi-inquilino é uma técnica para consolidar aplicações de vários inquilinos em sistema único. Esta técnica é frequentemente utilizada para eliminar a necessidade de sistemas separados para cada inquilino. Por exemplo, um inquilino pode ser um banco de dados que está hospedado em um SGBD. Compartilhamento de recursos, em diferentes níveis de abstração e níveis de isolamento distintos, resulta em vários modelos multi-inquilinos; compartilhamento de VM, compartilhamento de SGBD, compartilhamento de banco de dados e compartilhamento de tabela são técnicas bem conhecidas [Lang et al., 2012]. Tais esquemas de compartilhamento têm como objetivo melhorar os lucros dos provedores. SGBDs multi-inquilinos são utilizados para hospedar diversos inquilinos (bancos de dados) dentro de um só sistema, permitindo o compartilhamento eficaz de recursos em variados níveis de abstração e isolamento [Elmore et al., 2011a]. Uma visão geral dos recursos compartilhados e inquilinos, no âmbito de bancos de dados em nuvem, podem ser visualizados na Figura 1.1.

Técnicas de provisionamento dinâmico são voltadas para o tratamento de cargas de trabalho irregulares, para que violações de SLA e suas penalidades contratuais associadas sejam evitadas ou limitadas, reduzindo o custo da perspectiva do provedor [Moreira et al., 2014]. Essas técnicas realizam suas decisões de acordo com a observação da carga de trabalho e podem ser classificadas em reativa e proativa [Santos et al., 2013].

Recurso Compartilhado	Inquilino	Usuário
Máquina Física *Datacenter*	Máquina Virtual (MV)	O usuário da nuvem tem acesso exclusivo a uma máquina virtual, compartilhando a máquina física com outros usuários
Máquina Virtual (MV)	Usuário do Sistema Operacional (SO)	O usuário da nuvem compartilha o sistema operacional, por meio de um conta de usuário do SO
Sistema Operacional (SO)	Instalação de Sistema Gerenciador de Banco de Dados (SGBD)	O usuário da nuvem tem acesso exclusivo a uma instalação de SGBD no SO compartilhado
Instalação de Sistema Gerenciador de Bancos de Dados (SGBD)	Banco de Dados (BD)	O usuário da nuvem tem acesso exclusivo a um Banco de Dados e compartilha a instalação de SGBD com outros usuários da nuvem
Banco de Dados (BD)	Esquema	O usuário da nuvem tem acesso exclusivo a um esquema de BD, compartilhando este BD com outros usuários da nuvem
Tabela	Tupla	O usuário da nuvem compartilha uma tabela com outros usuários. No entanto, tem acesso exclusivo a uma tupla da tabela

Figura 1.1 Níveis de Compartilhamento Multi-Inquilino para Bancos de Dados em Nuvem

Soluções proativas aplicam modelos sofisticados para a previsão, resultando no auxílio à tomada de decisão para possíveis alocações de recursos ou modificações no ambiente, com a antecedência necessária para manter a qualidade. Em contraste, as abordagens reativas não usam previsão, mas sim detectam e reagem à sobrecarga de recursos, existentes por meio de limiares predefinidos para a qualidade do aplicativo ou métricas de sistema. Nos últimos anos, muitos trabalhos se servem dessas técnicas no intuito de resolver problemas de provisionamento de recursos [Sousa et al., 2012] [Sakr and Liu, 2012] [Prevost et al., 2011].

Migração de banco de dados é uma técnica que move um dado inquilino sob observação para um outro ambiente, sempre que forem detectadas alterações em seu desempenho [Barker et al., 2012]. Esta técnica pode ser usada para reduzir a carga em ambientes compartilhados, bem como para consolidar vários inquilinos em um ambiente que pode levá-los a compartilhamento de recursos com interferência reduzida. Assim, torna-se possível estabelecer um cenário de alocação para um conjunto de inquilinos em um conjunto de recursos, cujo objetivo principal é reduzir a interferência entre inquilinos. Na nuvem, a migração traz alguns custos, como custos de SLA e aqueles relacionados

à intervenção humana. A fim de reduzir estes custos, a técnica de migração de banco de dados deve tomar decisões justas que envolvem as seguintes variáveis: "quando" determina o instante de tempo que o inquilino deve ser migrado, "'qual' especifica quais inquilinos são alvos do processo de migração e "como" determinam o procedimento usado para migrar o inquilino [Barker et al., 2012].

De acordo com [Chaudhuri, 2012], um desafio interessante é desenvolver técnicas para garantir o desempenho em SGBD multi-inquilinos. Lidar com padrões imprevisíveis de carga e elasticidade são tarefas desafiadoras, mas é fundamental para garantir que os SLAs dos inquilinos sejam atendidos [Elmore et al., 2011b]. Além disso, antes do desenvolvimento de novas técnicas, é necessário entender como a carga de trabalho de um inquilino influencia no comportamento do outro e como o isolamento fornecido por um SGBD evita a interferência entre inquilinos. O problema é abordado em alguns trabalhos [Moon et al., 2013] [Schiller et al., 2013] [Hatem A. Mahmoud and El-Abbadi, 2012] [Lang et al., 2012] [Xiong et al., 2011] [Ahmad and Bowman, 2011] [Elmore et al., 2011a]. Tais estudos, no entanto, não referenciam aspectos do desempenho. Alguns são orientados a recursos e se concentram em consolidar muitos inquilinos em um mesmo *hardware* ou VM; outros não levam em conta características de SLA ou não consideram aspectos dos ambientes em nuvem.

A tese apresenta uma abordagem, denominada *Performance Multi-tenant Database in the Cloud* (PMDB), cujo objetivo é manter QoS em SGBDs multi-inquilinos e aumentar o reúso dos recursos de infraestrutura em nuvem. Para alcançar o objetivo, a abordagem se serve das técnicas que realizam predição de carga de trabalho, alocação e migração de inquilinos quando necessitam de recursos com maior capacidade. Para isso, uma arquitetura de plataforma como um serviço foi proposta e um protótipo implementado com tais técnicas, além de estratégias de monitoramento e QoS voltada para aplicações de banco de dados. Ademais, experimentos orientados a desempenho e utilização dos recursos foram especificados para mostrar a eficiência da abordagem para alcançar o objetivo em questão.

1.2 DEFINIÇÃO DO PROBLEMA

Em seu sentido mais amplo, o trabalho está relacionado com o problema de melhorar a qualidade de serviço para bancos de dados no ambiente de computação em nuvem. Tal domínio de problema, claramente extenso, é restringido desde um conjunto de carac-

terísticas. A Figura 1.2 ilustra o cenário do problema abordado neste trabalho.

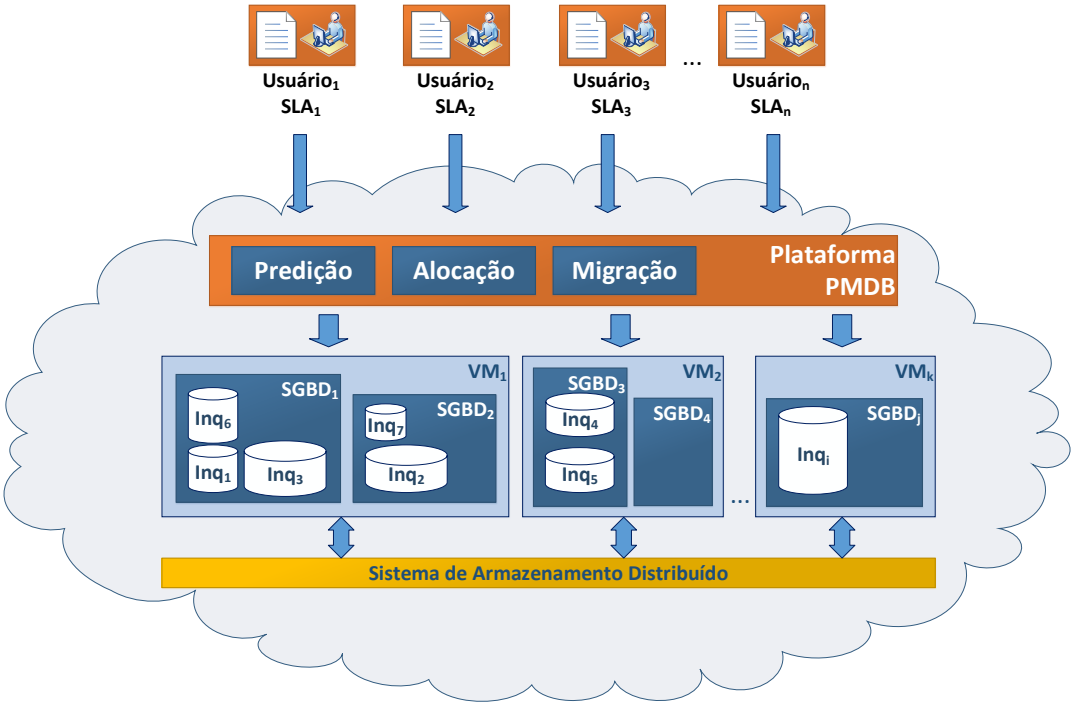

Figura 1.2 Definição do Problema

Considera-se um conjunto de VMs, onde todas as VMs são interligadas por um Sistema de Armazenamento Distribuído. O uso do Sistema de Armazenamento auxilia em uma comunicação e transferência de dados, entre VMs, de forma confiável e tolerante a falhas. Cada VM possui um ou vários SGBDs. Cada SGBD pode possuir um ou mais inquilinos (banco de dados). Cada usuário do PMDB fornece um SLA; esse SLA descreve o modelo de qualidade requerido para cada inquilino. Neste sentido, um SLA pode descrever a qualidade de um ou mais inquilinos.

O principal problema da tese consiste em: como melhorar a qualidade para um conjunto de serviços de banco de dados de acordo com a carga de trabalho corrente, enquanto utiliza recursos eficientemente com redução da violação do SLA?

Hipótese de tese: *Migração de banco de dados multi-inquilino e provisionamento de recursos: ambos, de forma preditiva, podem melhorar a qualidade de serviço e a utilização de recursos em ambientes de nuvem.*

1.3 OBJETIVOS

O objetivo geral da tese é desenvolver uma abordagem para melhorar a QoS para SGBD multi-inquilino, ao mesmo tempo que aproveita os recursos de infraestrutura em nuvem. Desta forma, o aproveitamento dos recursos computacionais diminui os gastos com estes por parte do provedor, aumentando o lucro. O trabalho propõe o uso de técnica de predição de cargas de trabalho, alocação e migração de inquilinos para garantir os SLAs dos inquilinos. As técnicas são implementadas em uma arquitetura, voltada para plataforma como um serviço, que permite sejam monitorados, continuamente, os SGBDs e inquilinos no ambiente, bem como todos os estados das VMs na infraestrutura em nuvem. A abordagem realiza migração conforme os dados de monitoramento, SLA e predição. A migração toma como base a técnica de alocação para melhorar a utilização dos recursos e o desempenho do inquilino. A predição, por sua vez, visa a anteceder o processo de migração ante um eventual decréscimo da qualidade que pode comprometer o SLA de um inquilino.

Para alcançar o objetivo geral, foram definidos os delineados passos na sequência.

1. Investigar a possibilidade da utilização de técnicas de gerenciamento de dados multi-inquilinos em ambientes distribuídos para migração de banco de dados em nuvem, em particular, conduzir o gerenciamento de um grande número de aplicações, cada uma possuindo banco de dados.

2. Projetar e desenvolver uma abordagem para migração de banco de dados em nuvem, objetivando manter a QoS e melhorar a utilização dos recursos em nuvem, por meio de técnicas de alocação de banco de dados e predição de carga de trabalho.

3. Explorar métodos ou técnicas voltadas para predição de cargas de trabalho em ambientes dinâmicos, a fim de realizar decisões inteligentes nas intervenções ou reconfigurações em um ambiente multi-inquilino.

4. Desenvolver um modelo de alocação que objetive melhor compartilhamento de VMs e SGBDs, ao mesmo tempo que não exceda suas capacidades diante de um modelo de qualidade imposto pelos usuários em ambiente de banco de dados multi-inquilino.

5. Analisar o desempenho alcançado pelo uso de técnicas de migração, alocação e predição no gerenciamento de bancos de dados multi-inquilinos. A intenção é verificar se tais técnicas são eficazes para reduzir a ocorrência de violações de SLA, ao

mesmo tempo em que melhora o compartilhamento de recursos em tais ambientes multi-inquilinos.

1.4 CONTRIBUIÇÕES

Principais contribuições da tese

1. *Uma abordagem para a migração de serviços de dados em nuvem*
   Utiliza-se uma arquitetura para banco de dados multi-inquilino que propõe a migração de bancos de dados e o provisionamento de recursos para melhorar a qualidade de serviço de bancos de dados relacional, com base em informações coletadas sobre a carga de trabalho e o ambiente, ou seja, a estratégia de monitoramento. A abordagem é independente do sistema gerenciador de banco de dados e pode ser utilizada em qualquer infraestrutura de nuvem [Moreira et al., 2014] [Moreira et al., 2013].

2. *Uma abordagem para a alocação de serviços de dados em nuvem baseado em SLA*
   Constitui-se de um modelo matemático baseado em desempenho dos inquilinos, ambiente e SLA como critério de decisão para atribuir uma nova localidade para um dado inquilino. O modelo foi implementado como um serviço e utiliza as informações da estratégia de monitoramento. Caso o modelo matemático não encontre um local adequado para o inquilino, ou seja, a atual localização é a melhor, o serviço de alocação sinaliza a necessidade de um provisionamento de nova VM [Moreira et al., 2014] [Moreira et al., 2013].

3. *Uma abordagem para a predição de cargas de trabalho para serviços de dados em nuvem*
   Esta abordagem visa à assistência na tomada de decisão sobre reconfiguração dos inquilinos no ambiente, sob o ponto de vista de sua localização. Com base nas informações de monitoramento do ambiente e das cargas de trabalho, utilizando análise preditiva, antevê-se o comportamento do inquilino em uma futura janela de tempo. Neste sentido e objetivando a diminuição da ocorrência de violações de SLA, o provedor pode se servir desta abordagem e tomar decisão sobre o inquilino [Moreira et al., 2014] [Santos et al., 2013].

4. *A implementação e avaliação experimental de uma arquitetura para o gerenciamento de banco de dados baseado em abordagens de migração, alocação e predição*

Este protótipo fornece suporte à infraestrutura Amazon AWS e pode ser facilmente estendido para outras infraestruturas, uma vez que todas as informações necessárias para sua utilização podem ser extraídas dos SGBDs e das infraestruturas por meio de agentes de monitoramento. A avaliação se serve de uma nuvem real e privilegia vários aspectos, como qualidade do serviço e utilização de recursos [Moreira et al., 2014] [Moreira et al., 2013] [Sousa et al., 2012].

Publicações

A tese é baseada principalmente nas publicações expressas na ordem.

- SOUSA, F. R. C. ; MOREIRA, L. O. ; MACHADO, J. C. "Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios". In: *III Escola Regional de Computação Ceará, Maranhão, Piauí (ERCEMAPI)*, 2009.

  Este capítulo de livro contém uma introdução sobre computação em nuvem, destacando suas características essenciais, modelos de serviços, modelos de implantação e multi-inquilinos. As principais tecnologias e plataformas, utilizadas nestes ambientes, foram exploradas e caracterizadas. Por fim, foram inventariados alguns problemas de pesquisa no novo paradigma computacional.

- SOUSA, F. R. C. ; MOREIRA, L. O. ; MACEDO, J. A. F. ; MACHADO, J. C. "Gerenciamento de Dados em Nuvem: Conceitos, Sistemas e Desafios". In: *25th Brazilian Symposium on Databases (SBBD)*, 2010.

  O segundo capítulo de livro introduz os conceitos de gerenciamento de dados em nuvem. Foram exibidas as formas de implantação de dados em nuvem, seguindo as diversas estruturas voltadas para a manipulação escalável de dados. Os modelos multi-inquilinos orientados a bancos de dados foram explorados, discutidos e caracterizados. Por fim, foram mostradas algumas questões de pesquisa voltadas para bancos de dados em nuvem.

- SOUSA, F. R. C. ; MOREIRA, L. O. ; MACHADO, J. C. "SLADB: Acordo de Nível de Serviço para Banco de Dados em Nuvem". In: *26th Brazilian Symposium on Databases (SBBD)*, 2011.

  O objetivo desse ensaio é conceber uma especificação de SLA para bancos de dados implantados em nuvens computacionais. O modelo de SLA auxilia os provedores

na implementação da qualidade do serviço de banco de dados e beneficia variados aspectos: tempo de resposta, vazão, disponibilidade e consistência. Visando a avaliar o SLA introduzido, alguns experimentos que medem a qualidade do serviço foram apresentados.

- SOUSA, F. R. C. ; MOREIRA, L. O. ; SANTOS, G. A C. ; MACHADO, J. C. "Quality of Service for Database in the Cloud". In: *2st International Conference on Cloud Computing and Services Science (CLOSER)*, 2012.

  Esse trabalho traz a especificação de uma arquitetura geral para prover qualidade de serviço em bancos de dados gerenciados por nuvens computacionais. Para isso, foram reunidos um conjunto de serviços, tecnologias e técnicas para alcançar qualidade em tais ambientes. A avaliação da arquitetura foi realizada utilizando estratégias de provisionamento para aumentar o desempenho e diminuir a violação de SLAs de bancos de dados inquilinos.

- MOREIRA, L. O. ; SOUSA, F. R. C. ; MACHADO, J. C. "Analisando o Desempenho de Banco de Dados Multi-Inquilino em Nuvem". In: *27th Brazilian Symposium on Databases (SBBD)*, 2012.

  Essa pesquisa tem como objetivo expor uma metodologia para analisar a interferência do desempenho entre bancos de dados multi-inquilinos no âmbito das nuvens computacionais. Após a exposição da metodologia, experimentos foram conduzidos para mostrar a eficácia do objetivo do trabalho. Na discussão dos experimentos, diversas lições aprendidas foram exibidas para tratar da interferência de bancos de dados em ambientes multi-inquilinos.

- SANTOS, G. A. C. ; MAIA, J. G. R. ; MOREIRA, L. O. ; SOUSA, F. R. C. ; MACHADO, J. C. "Scale-Space Filtering for Workload Analysis and Forecast". In: *IEEE 6th International Conference on Cloud Computing (IEEE CLOUD)*, 2013.

  Esse ensaio exibe o uso de técnicas de análises preditivas com o objetivo de realizar predições sobre cargas de trabalho em ambientes dinâmicos. Foi mostrada a utilização de métodos baseados em séries temporais e de apredizado de máquina para beneficiar o objetivo do referido trabalho. Por fim, experimentos foram feitos para mostrar a eficácia de tais métodos.

- MOREIRA, L. O. ; SOUSA, F. R. C. ; MAIA, J. G. R. ; FARIAS, V. A. E. ; SANTOS, G. A. C. ; MACHADO, J. C. "A Live Migration Approach for Multi-Tenant RDBMS in the Cloud". In: *28th Brazilian Symposium on Databases (SBBD)*, 2013.

Esse trabalho introduz uma solução parcial de migração, baseada em *Live Migration*, utilizando técnica de alocação com suporte no sistema de monitoramento presente na arquitetura do QoSDBC. Experimentos sobre a técnica de migração combinada com alocação foram feitos com o intuito de diminuir as violações dos SLAs dos inquilinos.

- MOREIRA, L. O. ; FARIAS, V. A. E. ; SOUSA, F. R. C. ; SANTOS, G. A. C. ; MAIA, J. G. R. ; MACHADO, J. C. "Towards Improvements on the Quality of Service for Multi-Tenant RDBMS in the Cloud". In: *6th International Workshop on Cloud Data Management (CloudDB)*, 2014.

  Esse trabalho cobre todos os objetivos descritos na tese. Nele foram mostrados: problemática, motivação, referência teórica e especificação da arquitetura proposta. Trabalhos correlatos foram discutidos e realizadas as avaliações experimentais, enfatizando o uso das técnicas de alocação, migração e predição. Por fim, foram expostas as conclusões e direcionamentos futuros de pesquisa.

Embora as publicações abaixo não estejam diretamente relacionadas a este trabalho, vários conceitos de arquitetura e melhores práticas para aplicações distribuídas foram desenvolvidos e utilizados:

- FARIAS, V. A. E. ; MAIA, J. G. R. ; SOUSA, F. R. C. ; MOREIRA, L. O. ; SANTOS, G. A. C. ; MACHADO, J. C. "A Machine Learning Approach for SQL Queries Response Time Estimation in the Cloud". In: *28th Brazilian Symposium on Databases (SBBD)*, 2013.

  O ensaio introduz uma abordagem para estimar tempo de resposta de consultas em ambientes em nuvem. Para isso, foram utilizados métodos baseados em aprendizagem de máquina, em particular, métodos de regressão, técnica *bag-of-words*, filtro estatístico e algoritmos genéticos. Experimentos foram conduzidos para a medição da qualidade, em particular, taxa de acerto na predição de consultas SQL.

- SOUSA, F. R. C. ; MOREIRA, L. O. ; MACHADO, J. C. "Computação em Nuvem Autônoma: Oportunidades e Desafios". In: *1st Workshop on Autonomic Distributed Systems (WoSiDA), SBRC*, 2011.

  Aqui foram discutidas as características da computação em nuvem e como o gerenciamento autônomo pode ser utilizado neste contexto. Nos três modelos de serviços,

foram destacadas oportunidades da aplicação do gerenciamento autônomo para melhorar a eficácia de tais serviços. Por fim, exibiram-se desafios de pesquisa que ainda podem ser explorados.

- MOREIRA, L. O. ; SOUSA, F. R. C. ; MACHADO, J. C. "A Distributed Concurrency Control Mechanism for XML Data". In: *Journal of Computer and System Sciences (JCSS)*, 2011.

  Esse experimento indica uma solução para controle de concorrência de dados XML em ambientes distribuídos. Foram mostrados e discutidos os algoritmos e técnicas para garantir os critérios de isolamento e consistência de dados em tais ambientes. Para isso, um protótipo foi concebido e discutida sua arquitetura. Experimentos que mostram aspectos de consistência, desempenho e isolamentos foram realizados e discutidos.

1.5 ESTRUTURA DA TESE

O restante da tese está organizada em seis capítulos. As seções seguintes explicam um sumário de cada um.

Capítulo 2: Fundamentação Teórica

O Capítulo 2 introduz a computação em nuvem e descreve o gerenciamento de dados nas nuvens computacionais, destacando diversos aspectos: armazenamento, processamento de consultas, transações, replicação e migração. Além disso, são expostas algumas técnicas para predição de cargas de trabalho voltadas para ambientes dinâmicos, por meio de análise preditiva.

Capítulo 3: Trabalhos Relacionados

O Capítulo 3 contém trabalhos que visam a melhorar qualidade de serviço de dados em nuvem. São indicados os requisitos para a migração no contexto de computação em nuvem, assim como uma análise comparativa entre os trabalhos relacionados.

Capítulo 4: Abordagem para Qualidade de Serviço em Bancos de Dados Multi-Inquilino em Nuvem

O Capitulo 4 traz PMDB, uma abordagem para melhorar a qualidade de serviço de dados em nuvem, destacando o modelo de qualidade e uma camada de *software* para migrar dados. PMDB utiliza técnicas de predição, provisionamento e migração, para garantir a qualidade do serviço em ambientes multi-inquilinos.

Capítulo 5: Arquitetura e Implementação

O Capitulo 5 destaca a arquitetura especificada para o PMDB, destacando os serviços de predição, alocação e migração em ambientes multi-inquilinos. Além disso, é demonstrado e detalhado o funcionamento do PMDB por meio do seu fluxo central. Também são apontados e discutidos os principais algoritmos implementados no PMDB.

Capítulo 6: Avaliação Experimental

Este capítulo mostra detalhes sobre a avaliação experimental realizada na solução expressa nos Capítulos 4 e 5 em uma nuvem computacional real.

Capítulo 7: Conclusões

Finaliza-se o ensaio com as Considerações Finais e com uma discussão sobre direções para possíveis trabalhos futuros.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

2.1 COMPUTAÇÃO EM NUVEM

A computação em nuvem está se tornando uma das expressões-chave da indústria de TI. A nuvem é uma metáfora para a Internet ou infraestrutura de comunicação entre os componentes arquiteturais, baseada em uma abstração que oculta a complexidade da infraestrutura. Cada parte desta infraestrutura é provida de um serviço, e estes são normalmente alocados em centros de dados, utilizando *hardware* compartilhado para computação e armazenamento [Buyya et al., 2009]. A infraestrutura do ambiente de computação em nuvem normalmente é composta por um grande número, centenas ou milhares de máquinas físicas ou nós físicos de baixo custo, conectadas por meio de uma rede. Cada máquina física tem as mesmas configurações de *software*, mas pode ter variação na capacidade de *hardware* em termos de CPU, memória e armazenamento em disco [Soror et al., 2008]. Dentro de cada máquina física existe um número variável de máquinas virtuais (VM) ou nós virtuais em execução, de acordo com a capacidade do *hardware* disponível na máquina física. Os dados persistem, geralmente, em sistemas de armazenamento distribuídos.

Existem diversas propostas para definir o paradigma da computação em nuvem [Vaquero et al., 2008]. O *National Institute of Standards and Technology* (NIST) argumenta que a computação em nuvem é um paradigma em evolução e exprime a seguinte definição: *"Computação em nuvem é um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos e liberados com mínimo esforço gerencial ou interação com o provedor de serviços"* [Sousa et al., 2010]. O NIST descreve que a computação em nuvem é composta por cinco características essenciais, três modelos de serviço e quatro modelos de implantação, detalhados a seguir.

2.1.1 Características Essenciais

As características essenciais são vantagens que as soluções de computação em nuvem oferecem. Algumas destas características, em conjunto, definem exclusivamente a computação em nuvem e fazem a distinção com outros paradigmas. Por exemplo, a elasticidade rápida de recursos, amplo acesso e a medição de serviço são características essenciais para compor uma solução de computação em nuvem.

- *Self-service sob demanda*: o usuário pode adquirir unilateralmente recurso computacional, como tempo de processamento no servidor ou armazenamento na rede, na medida em que necessite e sem precisar de interação humana com os provedores de cada serviço.
- *Amplo acesso*: recursos são disponibilizados por meio da rede e acessados mediante mecanismos padronizados que possibilitem o uso por plataformas do tipo *thin*, tais como telefones celulares, *laptops* e PDAs.
- *Pooling de recursos*: os recursos computacionais do provedor são organizados em um *pool* para servir a múltiplos usuários, empregando um modelo *multi-tenant* ou multi-inquilino, com diferentes recursos físicos e virtuais, dinamicamente atribuídos e ajustados de acordo com a demanda dos usuários. Os usuários não precisam ter conhecimento da localização física dos recursos computacionais, podendo somente especificar a localização em um nível mais alto de abstração, como o país, estado ou centro de dados.
- *Elasticidade rápida*: recursos podem ser adquiridos de forma rápida e elástica, em alguns casos automaticamente, caso haja a necessidade de escalar com o aumento da demanda, e liberados, na retração dessa demanda. Para os usuários, os recursos disponíveis para uso parecem ser ilimitados e podem ser adquiridos em qualquer quantidade e a qualquer momento.
- *Serviço medido*: sistemas em nuvem automaticamente controlam e otimizam o uso de recursos por meio de uma capacidade de medição. A automação é realizada em algum nível de abstração apropriado para o tipo de serviço, como armazenamento, processamento, largura de banda e contas de usuário ativas. O uso de recursos pode ser monitorado e controlado, possibilitando transparência para o provedor e para o usuário do serviço utilizado.

2.1.2 Modelos de Serviços

O ambiente de computação em nuvem é composto de três modelos de serviços. Estes modelos são importantes, pois definem um padrão arquitetural para soluções de computação em nuvem.

- *Software como um Serviço (SaaS)*: o modelo de SaaS proporciona sistemas de *software* com propósitos específicos disponíveis para os usuários por meio da Internet e acessíveis por via de vários dispositivos do usuário mediante uma *interface thin client* como um navegador *web*. Como exemplos de SaaS, pode-se destacar os serviços de *Customer Relationship Management* (CRM) da Salesforce e o Google Docs.
- *Plataforma como um Serviço (PaaS)*: o modelo de PaaS fornece sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de *software*. *Google App Engine* [Ciurana, 2009] e *Microsoft Azure* [Azure, 2014] são exemplos de PaaS.
- *Infraestrutura como um Serviço (IaaS)*: a IaaS torna mais fácil e acessível o fornecimento de recursos, como servidores, rede, armazenamento e outros recursos de computação fundamentais para estabelecer um ambiente de aplicação sob demanda, podendo incluir sistemas operacionais e aplicativos. O Amazon *Elastic Cloud Computing* (EC2) [Robinson, 2008] e o Eucalyptus [Liu et al., 2007] são exemplos de IaaS.

2.1.3 Modelos de Implantação

Quanto ao acesso e à disponibilidade, há variados tipos de modelos de implantação para os ambientes de computação em nuvem. A restrição ou abertura de acesso depende do processo de negócios, do tipo de informação e do nível de visão pretendido.

- *Nuvem privada*: a infraestrutura de nuvem é utilizada exclusivamente por uma organização, sendo esta nuvem local ou remota e administrada pela própria empresa ou por terceiros.
- *Nuvem pública*: a infraestrutura de nuvem é disponibilizada para o público em geral, sendo acessada por qualquer usuário que conheça a localização do serviço.

- *Nuvem comunidade*: fornece uma infraestrutura compartilhada por uma comunidade de organizações com interesses em comum.

- *Nuvem híbrida*: a infraestrutura é uma composição de duas ou mais nuvens, que podem ser do tipo privada, pública ou comunidade, e que continuam a ser entidades singulares, mas conectadas por meio de tecnologia proprietária ou padronizada que permite a portabilidade de dados e aplicações.

2.2 GERENCIAMENTO DE DADOS EM NUVEM

SGBDs em nuvem estão sendo utilizados em nuvem e têm o potencial de atrair clientes de vários setores do mercado, desde pequenas empresas com o objetivo de reduzir o custo total, por meio da utilização de infraestrutura e sistemas de terceiros, até grandes organizações que buscam soluções para gerenciar milhares de máquinas e permitir o atendimento de um aumento inesperado de tráfego [Abadi, 2009]. A infraestrutura de SGBDs em nuvem possui várias vantagens para os usuários: (i) previsibilidade e custos mais baixos, proporcional à qualidade do serviço (QoS) e cargas de trabalho reais; (ii) complexidade técnica reduzida, graças a *interfaces* de acesso unificado e à delegação de *tuning* e administração de SGBDs; e (iii) a elasticidade e escalabilidade, proporcionando a percepção de recursos quase infinitos. Por outro lado, o provedor tem que garantir (i) a ilusão de recursos infinitos, sob cargas de trabalho dinâmicas e (ii) minimizar os custos operacionais associados a cada usuário [Curino et al., 2011].

Diversos sistemas e arquiteturas estão sendo desenvolvidos para suprir as novas demandas de aplicações com vários requisitos de processamento e armazenamento [Abouzeid et al., 2009]. Estes novos sistemas tentam fornecer uma visão de armazenamento e escalabilidade infinitos, mas têm que tratar o problema de provisionar recursos. Este problema, que em SGBDs tradicionais consiste em determinar quais recursos são alocados para um só banco de dados, no ambiente em nuvem torna-se um problema de otimização, onde se tem uma grande quantidade de usuários, múltiplos SGBDs em nuvem e grandes centros de dados. Isso fornece uma oportunidade sem precedentes para explorar a economia em escala, balanceamento dinâmico de carga e gerenciamento de energia. O aumento no número de abordagens disponíveis de SGBDs em nuvem agrava o problema da escolha, implantação e soluções de administração para a gestão de dados. Com isso, os SGBDs em nuvem estão sendo disponibilizados como serviços, que encapsulam a complexidade do gerenciamento por meio de formas de acesso simples e garantias de acordos

de nível de serviço (SLAs).

2.2.1 Requisitos para o Gerenciamento de Dados em Nuvem

A definição dos requisitos é fundamental no gerenciamento de dados como um serviço. [Curino et al., 2011] fornecem uma lista de requisitos de um SGBD como um serviço da perspectiva do usuário, do provedor e requisitos adicionais relacionados à nuvem pública, conforme está na Tabela 2.1.

Requisitos do Usuário	
U1	API simples com pouca configuração e administração (ex. sem *tuning*)
U2	Alto desempenho (ex. vazão, escalabilidade)
U3	Alta disponibilidade e confiança (ex. *hot stand-by*, *backup*)
U4	Acesso fácil a características avançadas (ex. *snapshot*, evolução de esquema, mineração de dados)
Requisitos do Provedor	
P1	Atender o SLA do usuário (ex. potencialmente sob carga de trabalho dinâmica)
P2	Limitar *hardware* e custo de energia (ex. multiplexação intensiva)
P3	Limitar custo de administração (ex. custo com pessoal)
Requisitos extra de Nuvem Pública	
P1	Esquema de preço: barato, previsível e proporcional ao uso (elasticidade)
P2	Garantias de segurança e privacidade
P3	Baixa latência (relevante para OLTP e aplicações *web*)

Tabela 2.1 Requisitos para SGBD como um Serviço [Curino et al., 2011]

Da perspectiva do usuário, a principal necessidade é um serviço de banco de dados com uma *interface* simples, que não necessite de ajuste ou administração. Trata-se de uma melhoria em relação às soluções tradicionais que requerem técnicas para provisionar recursos, seleção de SGBDs em nuvem, instalação, configuração e administração. O usuário quer um desempenho satisfatório, expresso em termos de latência e vazão, independentemente do tamanho da base de dados e das alterações da carga de trabalho. Atualmente, esta é uma tarefa difícil, que exige ampla análise do pessoal de TI, *software* caro e atualizações de *hardware*. Alta disponibilidade é outro requisito fundamental, normalmente oferecido pelos SGBDs tradicionais, mas exige cuidados de configuração e manutenção. Finalmente, as características avançadas de gerenciamento do banco de dados, como *snapshot*, evolução de esquema e mineração de dados devem estar prontamente disponíveis e simples de utilizar.

Da perspectiva do provedor, é necessário atender aos acordos de nível de serviço, apesar da quantidade de dados e alterações na carga de trabalho. O sistema deve ser eficiente na utilização dos recursos de *hardware*. O modelo de serviço proporciona a oportunidade de fazer isso, por multiplexação de cargas de trabalho e ajuste dinâmico da alocação de recursos. Finalmente, a quantidade de tarefas de administração deve ser minimizada. Para tanto, ferramentas sofisticadas de análise de carga de trabalho e centralização do gerenciamento de muitos bancos de dados devem ser utilizadas. Para provedores de serviços em nuvem pública, existem requisitos adicionais, como esquema de preço, segurança, privacidade e latência. Estas questões, entretanto, não são específicas de bancos de dados e podem ser abordadas com técnicas em desenvolvimento pela comunidade de computação em nuvem.

2.2.2 Banco de Dados como um Serviço

Um grande impacto foi causado na indústria de *software*, ao prover um Sistema de Bancos de Dados como um Serviço (DaaS) e diversos desafios de pesquisa foram impostos à comunidade de banco de dados, incluindo, por exemplo, segurança, gerenciamento de recursos compartilhados e a extensibilidade. Em decorrência dessas necessidades, os DaaS estão surgindo como um novo paradigma para a gestão de dados em ambientes corporativos, em que um provedor hospeda um banco de dados e o fornece como um serviço [Agrawal et al., 2010]. Neste novo paradigma da gestão de dados, o usuário utiliza o serviço de dados por meio de várias funcionalidades, como, por exemplo, a configuração das bases de dados, esquemas, carga de dados no serviço de dados e interfaces padronizadas de interação com a base. As atividades de gerenciamento dos aplicativos de banco de dados e dos custos são transferidas do usuário para o provedor de serviços. A Figura 2.1 exibe a organização entre usuários e provedor de DaaS.

De acordo com a Figura 2.1, cada inquilino contrata os serviços fornecidos pelo provedor. Este provedor mantem um conjunto de banco de dados hospedados, em geral, em centros de dados. O provedor deve garantir aspectos de disponibilidade, desempenho e qualidade do serviço definidos em SLA. Do ponto de vista crítico, essa organização oferece uma série de benefícios aos usuários, pois dispõem de um ambiente altamente escalável, disponível e rápido. Além disso, não há preocupação em manter a infraestrutura de *hardware* e *software* para fornecer o serviço de dados. Outro ponto importante é que o provedor de serviços garante a QoS requerida pelo usuário. A própria arquitetura

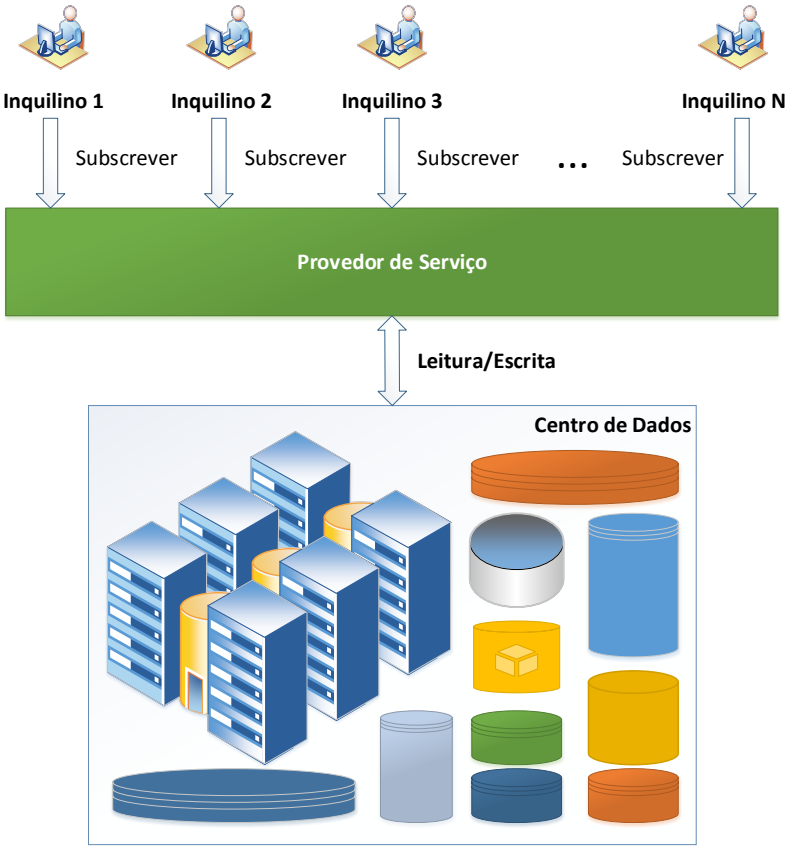

Figura 2.1 Banco de Dados como um Serviço

distribuída fornece confiabilidade no tocante à replicação de recursos.

Um sistema de banco de dados multi-inquilino deve oferecer esquemas que sejam flexíveis em dois aspectos [Aulbach et al., 2008]. Primeiro, deve ser possível estender o esquema-base para suportar múltiplas versões do aplicativo, por exemplo, para regiões geográficas distintas. Uma extensão pode ser privada para um inquilino individualmente ou compartilhada por vários inquilinos. Em segundo lugar, deveria ser possível evoluir de forma dinâmica o esquema-base e suas extensões, enquanto o banco de dados permanece em execução. Evolução de uma extensão deve ser totalmente "self-service": o provedor de serviço não deve ser envolvido; caso contrário, os custos operacionais serão muito altos. Jacobs e Aulbach [Jacobs et al., 2007] garantem que um sistema de banco de dados multi-inquilinos pode compartilhar máquinas, processos e tabelas. De acordo com [Hui et al., 2009], existem três abordagens para elaborar um banco de dados multi-inquilinos de acordo com o tipo de recurso compartilhado. A seguir, cada uma destas

abordagens é mostrada, bem como as vantagens e desvantagens são discutidas.

2.2.3 Modelos Multi-inquilinos para Banco de Dados em Nuvem

O termo multi-inquilino é uma estratégia utilizada para compartilhar recursos. Um inquilino é definido de acordo com o contexto em que se encontra; por exemplo, um inquilino pode ser um banco de dados em relação ao SGBD [Moreira et al., 2012]. Existem vários modelos de multi-inquilino que podem compartilhar desde máquinas físicas até tabelas. Por exemplo, a empresa Salesforce.com [Salesforce, 2014] utiliza o modelo de tabela compartilhada [Weissman and Bobrowski, 2009], enquanto [Soror et al., 2008] utilizam o modelo de VM compartilhada para melhorar a utilização dos recursos. Algumas características do gerenciamento de dados em nuvem aumentam a relevância de outros modelos de SGBDs multi-inquilinos. Para melhorar a compreensão destes modelos, [Elmore et al., 2011a] propõem uma nova classificação, como mostra a Tabela 2.2, ao estabelecer uma relação entre modelos de bancos de dados multi-inquilinos e os modelos de serviço da computação em nuvem, infraestrutura como serviço (IaaS), plataforma como serviço (PaaS) e *software* como serviço (SaaS).

Recurso Compartilhado	**Inquilino**	**IaaS**	**PaaS**	**SaaS**
1. *Hardware*	VM	X		
2. Máquina Virtual (VM)	Usuário do SO		X	
3. Sistema Operacional (SO)	Instância do SGBD		X	
4. Instância do SGBD	BD		X	
5. Banco de Dados	Esquema		X	
6. Tabela	*Tupla*			X

Tabela 2.2 Modelos de Bancos de Dados Multi-inquilinos e a Correspondência com a Computação em Nuvem [Elmore et al., 2011a]

Os modelos correspondentes às linhas 1-3 compartilham recursos nos níveis das mesmas máquinas físicas com variados níveis de abstração; por exemplo, múltiplas VMs, contas de SO de usuários diferentes e diversas instâncias dos SGBDs. Neste caso, não existe compartilhamento de recursos de banco de dados e as instâncias dos SGBDs se mantêm independentes. As linhas 4-6 envolvem o compartilhamento de processos de banco de dados em vários níveis de isolamento, como distintos bancos de dados, esquema ou *tablespace* e *tupla*. Nos vários modelos, os dados dos inquilinos são armazenados de várias formas. O modelo de *hardware* compartilhado utiliza a virtualização para chavear

diversas VMs na mesma máquina. Cada VM possui apenas um processo de banco de dados, e uma VM inteira, em geral, corresponde a um inquilino. Já o modelo de tabela compartilhada armazena dados de vários inquilinos em uma mesma tabela; e algumas *tuplas* de uma tabela correspondem a um inquilino.

Os modelos das linhas 4-6 são os mais amplamente utilizados, pois permitem melhor compartilhamento de recursos. Por outro lado, estes três modelos exprimem interferência maior entre os inquilinos do sistema, o que pode interferir no desempenho do sistema. No modelo da linha 4, um inquilino é um banco de dados separado, que melhora o isolamento. O modelo 4, no entanto, está limitado ao número de estruturas que o SGBD pode manipular. Os modelos 5-6 utilizam um número menor de recursos dentre todos os modelos, mas exprimem algumas desvantagens; por exemplo, o modelo 6 necessita de indexação e otimização, já que os inquilinos compartilham as mesmas tabelas, porém denotam requisitos diferentes.

2.3 MIGRAÇÃO DE BANCO DE DADOS

Nesta seção, encontram-se os conceitos relacionados às técnicas para migração de bancos de dados. Serão mostradas as abordagens para migração de bancos de dados no âmbito da computação em nuvem. Dentre as abordagens, estão as três principais, voltadas para *Live Migration*. Além disso, uma discussão será feita sobre vantagens e desvantagens de cada abordagem [Martinez, 2012].

A migração de bancos de dados é um processo que transforma os dados, de forma que seja possível transmitir por algum mecanismo de comunicação, restaura esses dados e deixa-os disponíveis em um novo destino, ou melhor, SGBD. São os possíveis componentes de dados, alvos de um processo de migração: esquema, dados, consultas, transações e suas respectivas operações. Caso haja alguma alteração da cópia-fonte, durante a migração, será necessária uma sincronização para garantir a coerência da nova cópia do banco de dados. De forma geral, a migração de bancos de dados é composta de dois momentos, que podem ser classificadas em fase *backup* e fase restauração.

Durante a fase de *backup*, existe um processo para recuperar e empacotar os dados da base de origem. Já na fase de restauração, surge a necessidade de um processo para desempacotar, implantar e verificar se os dados foram migrados corretamente. A Figura 2.2 ilustra a aplicação destas fases na migração de bancos de dados. E, claro, a execução

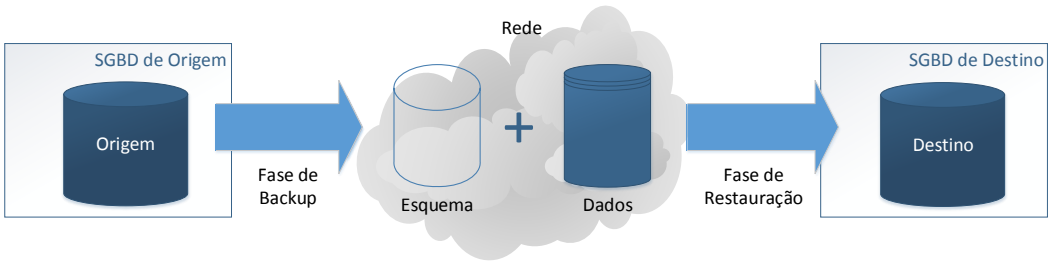

Figura 2.2 Fases do Processo de Migração

da migração de bancos de dados pode, inevitavelmente, comprometer as questões de qualidade de dados, que podem ser conturbadas pela natureza dos dados envolvidos, níveis de isolamento do SGBD, interferências de transações durante a migração etc.

2.3.1 Live Migration para Bancos de Dados Multi-inquilinos

Live Migration permite o balanceamento de carga elástica e escalabilidade sem tempo de inatividade e com interrupção mínima em bancos de dados multi-inquilinos, que residem em um ambiente de computação em nuvem. Há algumas maneiras em que a *Live Migration* pode ser implementada [Martinez, 2012]. Esta subseção exporá três dessas técnicas, mostrando suas características, funcionamento e, por fim, uma discussão sobre a aplicabilidade das técnicas.

Um método para realizar *Live Migration* é baseado no conceito de virtualização na camada de banco de dados. Neste método, o objetivo é separar a representação lógica de uma célula (a célula é um grânulo autocontido de dados de aplicação, metadados e estado que representa um inquilino no banco de dados), a partir do SGBD que o hospeda. Em [Das et al., 2010], é exibida e definida uma técnica, denominada *Iterative copy*, a qual tenta transferir o estado de memória principal da célula, a fim de que a célula-alvo reinicie na fonte de destino.

O estado da memória principal da célula é composto do estado do banco de dados em *cache* e o estado de execução da transação. Em geral, o estado do banco de dados consiste de páginas de banco de dados armazenadas em *cache* ou *buffer pool* ou alguma variante desta. O estado de transação inclui a transação ativa e, possivelmente, um subconjunto de transações efetivadas, necessárias para validar as transações ativas. *Live Migration* tem, em resumo, duas fases principais, cuja ilustração se encontra na Figura

2.3.

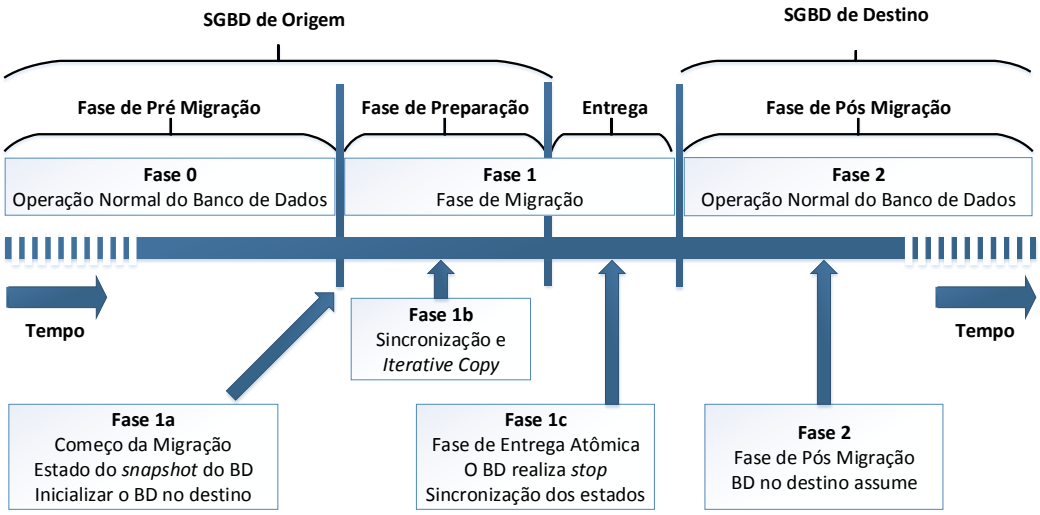

Figura 2.3 Fases do *Live Migration* Baseado em [Das et al., 2010]

A fase 0 pode ser considerada como o funcionamento do banco de dados no modo normal, executando transações. A fase 1 é conhecida como de migração, que notifica a fonte e o destino do processo de migração prestes a começar. Existem múltiplas subfases para a fase 1. A fase 1a começa a migração, adquirindo um *snapshot* do estado de banco de dados existente na fonte. O *snapshot* é então migrado para o novo destino e a migração de célula específica é inicializada. Note-se que a fonte dos dados deve continuar a processar as transações, enquanto a migração está em andamento. A fase 1b, por sua vez, é a fase de *Iterative copy*. Pelo fato de que a fonte de origem continua como servidor da célula que está a migrar, enquanto o *snapshot* está sendo transferido para a fonte de destino, o estado da célula de migração na fonte de destino permanece subjacente ao estado real na fonte de origem. A *Iterative copy* tenta recuperar o atraso e sincronizar os estados entre a fonte de origem e fonte de destino. Já a fase 1c é a real transferência atômica real da propriedade da célula migrada para a fonte de destino. A fonte de origem deixa de servir a célula e copia o estado final não sincronizado para a fonte de destino, liberando todas as alterações decorrentes de transações efetivadas no armazenamento persistente. Finalmente, a fase 2 é a pós-migração e marca o reinício das operações. Essa migração por meio da *Iterative copy* permite mínima interrupção do serviço de banco de dados, ao contrário de outras técnicas, como, por exemplo, *Stop and copy* e *On demand migration*.

Em *Stop and copy*, o serviço de uma célula específica é interrompido, os dados são

copiados e migrados e, em seguida, o serviço é iniciado mais uma vez. Já no *On demand migration*, o mínimo de informação é transferido, durante o que é conhecido como uma paragem rápida e cópia de migração. Assim, é fácil ver, porque esta opção é muito mais atraente do que outras soluções propostas [Das et al., 2011]. Do ponto de vista da implementação, a técnica *Stop and copy* é considerada mais simples, mas possui desempenho bem inferior em razão do longo tempo em que a unidade de transferência é bloqueada pelo processo de cópia, transferência, restauração, atualização e redirecionamento das conexões dos clientes que estão utilizando tal unidade de transferência. Ademais, em muitas soluções desta abordagem de migração, o processo de cópia do banco de dados antes da trasferência é feita mediante operações de *backup* por meio do SGBD. Neste ponto, o *backup* é a união do esquema do banco de dados com a cópia dos dados, ou melhor, o estado do banco de dados. A operação de restauração, por sua vez, é realizada por meio do processo de *restore* do SGBD, executando todo o *backup* transferido.

Existe outro método indicado como opção para realizar *Live Migration*. Esse tira proveito da técnica que extrai dados sob demanda e de forma assíncrona. O método, implementado no sistema Zephyr, é mostrado em [Elmore et al., 2011b] e utiliza um sincronismo dual para permitir que a fonte e a origem de dados possam executar transações. Primeiro, os metadados dos inquilinos são migrados para a fonte destinatária. Nesse ponto do tempo, a fonte destinatária pode começar a servir novas operações, enquanto a fonte de origem está completando todas as transações que estavam ativas quando a migração começou.

Acessos de leitura e escrita, ou melhor, a propriedade das páginas de banco de dados, são divididos entre as fontes de origem e de destino; a fonte de origem, inicialmente, possui todas as páginas e a fonte de destino só adquire propriedade das páginas sob demanda de forma transacional. É também importante ressaltar que as estruturas de índice são replicadas da origem para o destino durante a migração. A sincronização entre a fonte de origem e a de destino é necessária durante um curto espaço de tempo, basicamente, durante a execução de transações que afetam origem e destino. O Zephyr não leva em consideração a camada de banco de dados, o que permite melhor flexibilidade na escolha do destino para a migração, que pode ou não ter réplica do inquilino [Elmore et al., 2011b].

O trabalho proposto por [Curino et al., 2011], implementado no sistema Relational Cloud, aponta a terceira solução de implementação do *Live Migration*, ao mesmo tempo em que privilegia a escalabilidade elástica. Os autores acreditavam que a solução,

em termos de escalabilidade elástica, poderia ser alcançada por meio do particionamento do banco de dados. A estratégia de particionamento, que os autores propõem, é centrada em torno de cargas de trabalho OLTP e *web*, significando que tal estratégia é adequada para operações de curta duração. Por conseguinte, eles devem dividir os dados, de tal maneira que possam minimizar o número de transações em múltiplas fontes de bancos de dados.

Quando o número de fontes de bancos de dados não é minimizado, então existe um aumento no tempo gasto para manter os bloqueios. Portanto, a estratégia analisa rastros de execução para que se possa identificar conjuntos de *tuplas*, acessadas em conjunto dentro de transações individuais. Estes conjuntos de *tuplas* e transações são representados como um grafo onde cada nó é uma *tupla* e uma aresta é desenhada entre dois nós, cujas *tuplas* são tocadas com única transação. O peso na aresta representará quantas vezes a *tupla* é acessada na carga de trabalho. Os autores usam, então, particionamento de grafo para encontrar l partições lógicas balanceadas, durante a tentativa de minimizar o peso total das arestas cortadas.

O objetivo esperado é encontrar um particionamento das *tuplas* do banco de dados, que minimiza o número de transações distribuídas [Martinez, 2012]. O Relational Cloud deve, agora, encontrar uma representação muito concisa das partições, de modo a que as operações SQL podem ser devidamente encaminhadas, o que é feito por meio da definição de um conjunto de predicados sobre os atributos da *tupla*. Portanto, dado um conjunto de *tuplas* e um rótulo de partição para cada *tupla*, o sistema pode extrair um conjunto de atributos candidatos com origem nos predicados usados no rastreamento. Uma vez que isso for concluído, os atributos são dados para um algoritmo de árvore de decisão, juntamente com os rótulos de particionamento.

Se a árvore de decisão pode, com sucesso, generalizar o particionamento com apenas alguns predicados, neste sentido, é considerado que a partição do grafo foi encontrada. Caso contrário, o sistema deve usar tabelas de pesquisa para representar o esquema de particionamento [Curino et al., 2011]. Esta abordagem permite a independência do tipo de esquema, bem como as informações de chave estrangeira. A técnica, ainda, enfrenta dificuldades quando chega a um grafo com N nós e N_2 arestas para um banco de dados de N-*tuplas* [Martinez, 2012].

2.4 ANÁLISE PREDITIVA

Análises preditivas estão se tornando mais populares como resultado dos avanços em aprendizagem de máquina, poder computacional e disponibilidade de grande volumes de dados [Santos, 2013]. As Técnicas para Análises Preditivas fazem uso de mineração de dados, modelagem matemática e análise estatística, para proporcionar previsões viáveis baseadas em tendências, padrões, relações e correlações em dados. Tais técnicas produzem insumos importantes para se conduzir o processo de tomada de decisão. As abordagens e técnicas utilizadas para realizar análises preditivas incluem, dentre muitas outras formas: árvores de decisão e regressão; previsão de séries temporais; redes neurais artificiais; meta-heurísticas como, por exemplo, os algoritmos genéticos; máquinas de vetores de suporte; e outros algoritmos matemáticos [Ayhan et al., 2012] [Debahuti Mishra and Mishra, 2010].

2.4.1 Previsão em Séries Temporais

Tudo o que se observa ou mede tende a ser distinto em variados pontos no tempo [Dorffner, 1996]. Quando coletamos um conjunto de tais medidas com o objetivo de inferir conclusões sobre determinado fenômeno com base nelas, considerando o instante em que foram tomadas as medidas, trata-se de um problema com aspecto temporal. Isso significa que ocorre dependência temporal nos dados. Em outras palavras, uma entrada em uma iteração não expressa informações suficientes para obter a saída apropriada para aquele fenômeno. A saída é, portanto, dependente do número de padrões de entrada [Kok, 2007]. Assim, os dados de séries temporais possuem ordenação natural com relação ao tempo. Isso difere das aplicações típicas de aprendizagem de máquina, em que cada amostra é um exemplo independente do conceito a ser apreendido e a ordem das amostras em um conjunto não importa. Exemplos de aplicações de séries temporais incluem o planejamento de capacidade, reposição de estoques e previsão de vendas.

2.4.1.1 Séries Temporais e Análise Uma série temporal é um conjunto de observações x_t, onde cada observação corresponde a um determinado momento t. Uma série temporal é dita discreta quando o conjunto de T_0, em que são tomadas as observações, é um conjunto discreto, ou seja, finito. Esse é tipicamente o caso quando são feitas observações em intervalos fixos de tempo. Séries de tempo contínuo são obtidas quando as observações são registradas continuamente sobre algum intervalo de tempo, por

exemplo, quando $T_0 = [0, 1]$ [Brockwell and Davis, 2009] e, como tal, as excitações que são impostas ao sistema em observação não estão sob o controle do observador, sendo, em muitos casos, completamente desconhecidos por ele ou ela [Herrera et al., 1999]. Análise de séries temporais é o processo de utilização de técnicas estatísticas para modelar e explicar uma série de amostras dependentes do tempo. Tais amostras são comumente denotadas como pontos de dados. Essa análise permite determinar se os pontos de dados demonstram uma estrutura interna, como autocorrelação, tendência ou variação sazonal, que deve ser contabilizada, e também fornece ferramentas para a seleção de um modelo que pode ser utilizado para prever eventos [Natrella, 2010].

O problema geral de previsão de séries temporais pode ser reformulado como a dificuldade de encontrar um modelo computacionalmente viável e capaz de prever a evolução de um fenômeno, dada sua evolução passada. Na maioria dos casos, o problema de previsão é limitado à predição de uma série temporal de curto prazo, palavras quando se busca modelar a evolução futura de uma série de tempo com o objetivo de se prever qual seria a observação imediatamente adiante. A confiabilidade dos dados previstos é o principal motivo para se utilizar tal abordagem, pois a incerteza dos valores futuros aumenta com o horizonte de tempo [Lendasse et al., 2004]; ou seja, durante a primeira etapa de uma previsão em múltiplos passos, o valor previsto depende inteiramente dos dados observados, de modo ser mais provável que seja mais preciso do que etapas subsequentes, quando as previsões dependem de previsões anteriores e que são, por si, associadas a algum grau de incerteza [Herrera et al., 1999]. [Plummer, 2000] enumera algumas dificuldades que podem surgir durante a realização de séries temporais:

1. quantidade limitada de dados, que pode ser a principal dificuldade;
2. ruído. Qualquer anomalia inesejada nos dados. O ruído pode ser causado por:
   - pontos de dados errôneos;
   - componentes que obscurecem a forma subjacente da série de dados.
3. não estacionariedade dos dados. Os dados não possuem as mesmas propriedades estatísticas em cada ponto do tempo, como, por exemplo, média e variância; seleção de técnica de previsão com uma infinidade de opções de estatísticas para a inteligência artificial. Uma das técnicas mais simples consiste em buscar uma série de dados, descrevendo eventos passados semelhantes, e usar comparações para fazer

uma previsão. Uma das técnicas mais complexas é o treinamento de um modelo a partir da série e aplicar o modelo para obter uma previsão.

2.4.1.2 Auto Regressive Integrated Moving-Average (ARIMA) Modelos ARIMA são amplamente utilizados para estimar previsões sobre séries temporais. Como a abordagem foi proposta por [Box and Jenkins, 1976], os modelos ARIMA são muitas vezes referidos como modelos de Box-Jenkins. Esse modelo autorregressivo integrado de média móvel é geralmente denotado como *ARIMA(p, d, q)*, onde:

- p é o número de termos auto-regressivos,
- d é o número de diferenças não sazonais, e
- q é o número de termos da média móvel.

Como autorregressivo, o modelo explora a possibilidade de os dados expressar autocorrelação, que é uma espécie de memória, com origem na qual se pode estabelecer correlação entre pontos consecutivos da série temporal. Diferenciações são utilizadas para remover variações sazonais das propriedades estatísticas da série temporal, e que a tornam não estacionária. As médias móveis são usadas como suporte para o comportamento geral da série em variados intervalos de tempo. Para identificar o modelo ARIMA apropriado para uma série temporal, começa-se por identificar a ordem de diferenciação necessária para tornar a série estacionária. Assim, são realizadas diferenciações sobre a série original até que esta se torne estacionária. Uma razão forte para o uso de uma sequência fixa de dados, em vez de uma sequência não estacionária, é que as sequências não estacionárias, geralmente, são mais complexas e precisam de mais cálculos para se obter uma previsão. Quando a diferenciação da série é feita com sucesso, supõe-se que a série diferenciada pode ser tratada da mesma maneira como uma série estacionária.

Isto conduz a uma família mais ampla de modelos, que são modelos ARMA, após diferenciação. Como tal, os modelos ARIMA são uma generalização dos modelos ARMA, obtidos por meio da introdução da diferenciação no modelo, com I, indicando *"Integrado"* e faz referência ao processo de diferenciação. Com uma série estacionária ($d = 0$), um modelo básico pode ser identificado. Depois de obter tanto o *AR de ordem p* e *MA de ordem q*, leva-se a um dos três modelos básicos: AR (autorregressivo), MA (média móvel) e um ARMA combinado.

- Quando apenas $p \neq 0$ e $q \neq 0$, há o modelo habitual ARMA, pois $ARIMA(p, 0, q) = ARMA(p, q)$.
- Quando somente $p \neq 0$, há $ARIMA(p, 0, 0) = AR(p)$.
- Finalmente, somente quando $q \neq 0$, $ARIMA(0, 0, q) = MA(q)$.

Em seguida, é necessário estimar os coeficientes do modelo, o que consiste em encontrar φ e θ no primeiro, ou somente φ no segundo caso e θ no terceiro. Estes modelos são dados por:

$$AR(p) : X_t = C + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t \tag{2.1}$$

$$MA(q) : X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{2.2}$$

$$ARMA(p, q) : X_t = C + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{2.3}$$

Onde φ_i e θ_i são parâmetros, a variável aleatória ε é o erro de ruído branco, C é uma constante e μ é a média de X_t (muitas vezes assumido como 0).

Na prática, a estimativa é bastante transparente para o utilizador. Finalmente, o modelo tem que ser verificado. Esta etapa também é chamada de checagem de diagnóstico ou verificação [Anderson, 1976]. São dois importantes elementos de verificação: (1) garantir que os resíduos do modelo sejam aleatórios; e (2) assegurar que os parâmetros estimados sejam estatisticamente significativos. Normalmente, a seleção de parâmetros é guiada pelo princípio de parcimônia, pelo qual o melhor modelo é o mais simples possível, isto é, que possui o menor número de parâmetros, que descrevem adequadamente os dados.

2.4.2 Aprendizagem de Máquina

Em termos gerais, Aprendizagem de Máquina consiste na construção e estudo de programas computacionais capazes de aprender a partir de dados, o que geralmente está associado à tarefa de otimizar determinado critério de desempenho, usando exemplos

de dados ou de experimentos passados [Alpaydin, 2010]. Note-se que tais sistemas são essencialmente inteligentes e portanto aplicáveis em ambientes dinâmicos, visto que não precisam necessariamente ser reprogramados para que lidem adequadamente com novos cenários. Se o sistema pode apreender e se adaptar às mudanças, então, o projeto do sistema não necessitará prever e providenciar soluções para todas as situações possíveis. Como alternativa, o sistema deve ser capaz de inferir hipóteses desde um novo conjunto de dados, o que resulta em alguma forma de representação conveniente para ajustar algoritmos e criar soluções corretas.

2.4.2.1 Aprendizagem e Generalização A aplicação de aprendizagem de máquina envolve duas etapas. A primeira é denominada treinamento e consiste em obter um modelo por meio da observação de amostras, de modo que o modelo obtido seja utilizado na etapa subsequente, denominada ativação. Via de regra, o treinamento deve evitar a replicação dos dados fornecidos como entrada. É comum que os modelos absorvam, *ipsis litteris*, o conjunto de entrada, sendo incapaz de produzir alguma resposta útil para novas amostras em decorrência a um sobreajuste. Por outro lado, seria possível admitir essa limitação desde que o conjunto de treinamento contenha todas as amostras possíveis para o fenômeno estudado, o que seria claramente impossível ou computacionalmente inviável. Assim, é importante que o modelo possua capacidade de *generalização*, ou seja, que lide adequadamente com novos casos que não podem ser encontrados no conjunto de treinamento [Alpaydin, 2010]. Essa abordagem de modelagem que tem a habilidade de aprender com experimentos é muito útil para alguns problemas práticos, desde que se consigam, facilmente, os dados a fim de que se obtenham suposições teoricamente boas sobre as leis básicas que dirigem o sistema com procedência em um dado gerado [Zhang et al., 1998].

2.4.2.2 Support Vector Machines (SVM) As Máquinas de Vetores de Suporte são uma nova classe de algoritmos de aprendizagem, motivada pelo resultado da teoria de aprendizagem estatística. Originalmente desenvolvida para o reconhecimento de padrões, um modelo SVM descreve uma superfície de decisão em termos de um pequeno subconjunto de todos os exemplos de treinamento. Esse subconjunto é denominado de vetores de suporte e produz uma margem maximal de separação entre duas classes de amostras. Vapnik criou a métrica ϵ-*insensitive loss function* para o caso dos *Support Vectors for Regression* (SVR),

$$L_{\varepsilon}\left(f\left(x_i\right), y_i\right) = \begin{cases} 0 & \text{Se } \left|y_i - f\left(x_i\right)\right| \leq \varepsilon \\ \left|y_i - f\left(x_i\right)\right| - \varepsilon & \text{caso contrário} \end{cases} \tag{2.4}$$

a qual não penaliza erros abaixo de um $\epsilon > 0$ escolhido *a priori* [Schölkopf et al., 2000].

Assim, o SVR utiliza *ϵ-loss function* para criar um *ϵ-tube* ao redor das amostras de treinamento; logo, estas amostras internas ao *ϵ-tube* não são penalizadas como erros; enquanto amostras fora do *ϵ-tube* se tornam vetores de suporte (SVs) que serão usados para testes. No mesmo sentido, contudo, vetores de suporte são subconjuntos ideais a serem selecionados [Wang et al., 2012]. O método SVR transforma cada amostra x para um espaço de alta dimensionalidade F por meio de um mapeamento não linear ϕ. Isso porque F permite lidar naturalmente via regressão linear com situações inviáveis no espaço original, em que os dados são amostrados. Na prática, esse mapeamento tornaria o problema inviável, pois a dimensão de F é muito alta, tornando deveras complexo o simples processamento de todas as componentes de uma só amostra. Esse problema é contornado pela utilização do *Kernel Trick*, que consiste em substituir os produtos escalares em F por uma função simples de duas amostras no espaço original [Mukherjee et al., 1997]. Qualquer função que satisfaça o teorema de Mercer pode ser utilizada como uma *kernel function* [Wang and Xu, 2004]. Novos *kernels* estão, entretanto, sendo propostos por pesquisas. Típicos exemplos destas *kernel function* estão demonstradas a seguir.

$$\begin{aligned} &\text{Linear:} && k\left(x_i, x_j\right) = && x_i^T x_j \\ &\text{Polinomial:} && k\left(x_i, x_j\right) = && \left(\gamma x_i^T x_j + r\right)^d, \gamma > 0 \\ &\text{Radial Basis Function (RBF):} && k\left(x_i, x_j\right) = && \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right), \gamma > 0 \\ &\text{Sigmoid:} && k\left(x_i, x_j\right) = && \tanh\left(\gamma x_i^T x_j + r\right) \end{aligned} \tag{2.5}$$

Sendo γ, r e d parâmetros do *kernel* RBF, que geralmente é utilizado pelo seu notório poder de expressão, seus valores devem ser determinados cuidadosamente. Isso porque, implicitamente, esses parâmetros definem a estrutura do espaço de características de alta dimensão $\phi\left(x\right)$ e controlam, assim, a complexidade da solução final [Santos, 2013].

Logo, a *kernel function* se transforma em um produto definido no espaço de alta dimensionalidade, no qual a solução do problema original pode ser representada como sendo um problema linear [Santos, 2013]. Note-se que, em uma configuração não linear, o problema de otimização consiste em encontrar a função mais plana no espaço de característica, e não no espaço de entrada. Note-se, também, que o parâmetro ϵ pode ser utilizado se a precisão desejada da aproximação for especificada antecipadamente. Em alguns casos, contudo, tenciona-se que a estimativa seja a mais precisa possível sem comprometimento a um nível de precisão dada *a priori* [Schölkopf et al., 2000]. Em seu trabalho, os autores descrevem a modificação do algoritmo ϵ-*SVR*, chamado de ν-*SVR*, o qual, automaticamente, minimiza ϵ.

2.4.3 Métricas de Erro

Foram adotadas três métricas para avaliar a acurácia das previsões deste ensaio, onde y_i denota o valor da saída real, $\hat{y}_i$ é o valor da saída prevista e n é o número de observações na massa de dados sobre a qual a previsão é feita:

1. Root Mean Squared Error (RMSE), que é definido como segue:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{2.6}$$

2. Mean Absolute Percentage Error (MAPE), definido de acordo com a seguinte fórmula:

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{y_i} \tag{2.7}$$

3. PRED(25), dado por:

$$PRED(25) = \frac{\text{Número de observações com erro relativo} \leq 25\%}{\text{Número de observações}} \tag{2.8}$$

   Onde o erro relativo é expresso como:

$$RE = \frac{|y - \hat{y}|}{y} \tag{2.9}$$

As seguintes observações são relevantes, como apontado por [Islam et al., 2012]:

- um valor inferior de $RMSE$ indica um esquema de previsão mais efetivo;
- um valor pequeno de $MAPE$ implica um modelo de previsão mais bem ajustado, apontando, portanto, acurácia superior;
- $PRED(25)$ indica o percentual das observações cuja acurácia de previsão jaz entre 25% do valor real. Valores próximos a 1 indicam melhor ajuste do modelo de previsão;
- $RMSE$ é mais sensível do que outras métricas às ocorrências pontuais de grandes erros: a adoção do quadrado dos valores produz pesos desproporcionais para erros muito grandes. Se um erro pontual de grande valor não representa problema em um estado de decisão, MAPE pode, então, se apresentar como um critério mais relevante. Em muitos casos, essas estatísticas variam em uníssono, ou seja, o modelo que se sobressai em uma das métricas também se destaca nas demais. Na realidade, se um modelo é o melhor de acordo com uma métrica e outro modelo é o melhor consoante outra métrica, eles são provavelmente bastante similares em termos de seus erros médios. Em tais casos, um modelo provavelmente deve atribuir maior peso para algum dos outros critérios usados para comparar modelos;
- faz sentido afirmar "O modelo é bom (ruim) porque o erro RMSE é inferior (superior) a x", a não ser que esteja se referindo a um grau específico de acurácia, relevante para uma determinada aplicação;
- se o valor do RMSE é um modelo de 30% inferior ao valor exibido por outro modelo, então, a diferença é provavelmente muito significante. Se é 10% inferior, tal diferença é de algum modo significante. Se um valor de RMSE for apenas 2% inferior, provavelmente, não será significante. Essas distinções são especialmente importantes quando se estiver estabelecendo compromisso entre a complexidade de variados modelos e suas respectivas métricas de erro.

Essas observações constituem a razão pela qual foram escolhidas três métricas distintas de erro. A escolha proverá fundamentos mais sólidos para comparação dos resultados.

2.5 CONCLUSÃO

Este capítulo conteve uma introdução à computação em nuvem e as principais características relacionadas ao gerenciamento de dados em nuvem. Também foram apontados os requisitos da gestão de dados em nuvem. A compreensão destas características é extremamente importante e possibilita fazer escolhas adequadas no desenvolvimento de soluções para a computação em nuvem. Além disso, acredita-se que a combinação entre características de abordagens distintas pode conduzir ao surgimento de soluções mais eficazes no gerenciamento de dados. Também foram discutidas as principais abordagens encontradas na literatura para a migração de banco de dados relacional em nuvem. Por fim, foram mostrados métodos para análise preditiva, destacando os de séries temporais e de aprendizagem de máquina. O próximo capítulo relata os trabalhos relacionados a este ensaio.

CAPÍTULO 3

TRABALHOS RELACIONADOS

Este capítulo relata alguns trabalhos de caráter multi-inquilino, que tratam do desempenho e das garantias de QoS para sistemas de banco de dados em ambientes virtualizados. Inicialmente, são elencados os requisitos para a QoS de banco de dados no contexto da computação em nuvem. Em seguida, descreve-se cada um desses trabalhos, considerando os requisitos levantados. Por fim, um estudo comparativo é descrito, destacando as limitações dos trabalhos.

3.1 INTRODUÇÃO

Para compreender, de maneira mais aprofundada, as características necessárias para manter a QoS de banco de dados em nuvem, foram identificados, com base em trabalhos correlatos, requisitos referentes às aplicações e aos usuários, conforme mostra a Tabela 3.1. Os requisitos consideram as características da computação em nuvem e do gerenciamento de dados na perspectiva do provedor, melhorando a utilização dos recursos, ao mesmo tempo em que garante o modelo de qualidade imposto pelas aplicações.

Requisito	**Descrição**
Gerenciamento de Recursos	Reconfigurar os recursos de infraestrutura
Predição	Evitar violações de SLA
Multi-inquilino	Nível de compartilhamento dos recursos
QoS	Atender o contrato de serviço
Migração	Realiza a transferência do inquilino
Consistência	Garantir consistência forte
Alocação	Utilizar eficientemente os recursos existentes

Tabela 3.1 Requisitos para a Qualidade de Serviço de Banco de Dados em Nuvem

O provedor deve fornecer meios, preferencialmente automatizados, para o gerenciamento de recursos, de acordo com o comportamento da carga de trabalho. Além disso, o provedor também deve garantir a qualidade de serviço, definida por meio de um SLA

com o usuário, mesmo com alterações na carga de trabalho. Para tanto, deve-se fazer uso do gerenciamento automático, já que os sistemas em nuvem utilizam e compartilham uma grande quantidade de recursos. Neste caso, os provedores de IaaS utilizam técnicas de virtualização para auxiliar o gerenciamento dos recursos e melhorar a flexibilidade do sistema.

Para melhorar a utilização dos recursos, deve-se implementar o conceito de multi-inquilinato em algum nível de compartilhamento, tais como SGBD, banco de dados ou tabela. Assim, o provedor reduz os custos, pois utiliza os recursos de forma eficiente. Por fim, as aplicações possuem requisitos de consistência forte, pois muitas destas aplicações estão sendo migradas para a nuvem e foram desenvolvidas considerando esse tipo de consistência.

3.2 TRABALHOS RELACIONADOS

Existem muitas soluções referentes a QoS em banco de dados para ambientes distribuídos. A seguir, são apresentadas as soluções que se relacionam diretamente a esta proposta de tese, ou seja, gerenciamento de dados para apoiar diferentes aplicações, na perspectiva do provedor, que utilizam bancos de dados.

Plataforma proposta por [Elmore et al., 2011a]

Neste trabalho são propostas uma técnica elástica de balanceamento de carga e uma forma eficiente para migração de banco de dados. Com essas técnicas, o experimento tem como objetivo melhorar a utilização dos recursos em nuvem, diminuindo seu custo operacional. Como principais contribuições, destacam-se os seguintes aspectos: uma forma de *Live Migration*; a aplicabilidade de um modelo multi-inquilino para banco de dados; e um protótipo de implementação para o gerenciamento de banco de dados multi-inquilino em nuvem, baseado nas técnicas de migração e balanceamento de carga.

São discutidos diversos modelos multi-inquilino para banco de dados, ressaltando vantagens e desvantagens nos níveis de compartilhamento destes modelos. A Figura 3.1 exibe a arquitetura proposta no ensaio, sendo possível observar que foi utilizado o modelo multi-inquilino baseado em SGBD compartilhado. A escolha é justificada pelo fato de o modelo possuir melhor isolamento entre os inquilinos, facilitando a migração dos dados. Além disso, o modelo permite, segundo o levantamento feito durante a pesquisa, maior

privacidade dos dados, facilidade para aplicação de métodos de replicação e técnicas de recuperação após falhas.

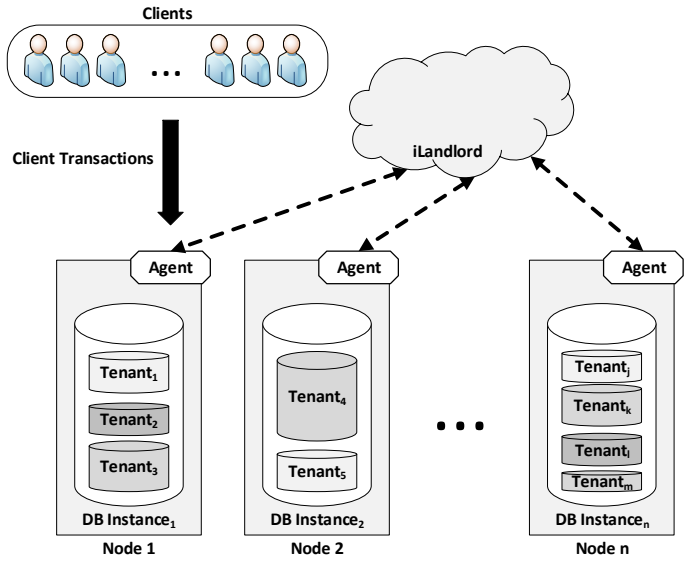

Figura 3.1 Arquitetura de Sistema Proposta por [Elmore et al., 2011a]

A arquitetura ora reproduzida é uma extensão da que foi proposta em [Yang et al., 2009]. Nela existe um coordenador, denominado *iLandlord*, que controla a localização dos inquilinos, gerenciamento da elasticidade, monitoramento e a conexão do usuário com o inquilino. Técnicas de aprendizagem de máquina são utilizadas para melhorar a alocação dos inquilinos no ambiente. Em base ao monitoramento dos recursos de máquina/inquilino e às descrições de SLA, o *iLandlord* decide para onde o inquilino deve ser migrado ou se a elasticidade há que ser aplicada no ambiente.

Foram mostrados, porém, somente, problemas alcançados quando um SGBD possui vários bancos de dados. Além disso, o trabalho não expõe nenhum resultado experimental do estudo realizado. A proposta do trabalho é mais arquitetural e serve de referencial teórico sobre os modelos multi-inquilinos e as estratégias de migração, destacando os diversos tipos de implementação, vantagens e desvantagens para o contexto de banco de dados. Ademais, não foi apresentado algum modelo de consistência para a plataforma proposta.

Zephyr [Elmore et al., 2011b]

O trabalho [Elmore et al., 2011b] propõe um sistema, denominado *Zephyr*, que implementa uma estratégia eficiente para migrar bancos de dados. Do ponto de vista de modelo multi-inquilino, foi utilizado o de SGBD compartilhado. O foco do trabalho é tratar o problema da migração em bancos de dados relacionais em nuvem. O *Zephyr* possui características que minimizam a interrupção do serviço de banco de dados durante o processo de migração, servindo-se da técnica de *Live Migration*. Ademais, são garantidas as propriedades ACID e a confiabilidade diante de falhas. A Figura 3.2 traz uma visão geral da arquitetura do *Zephyr*.

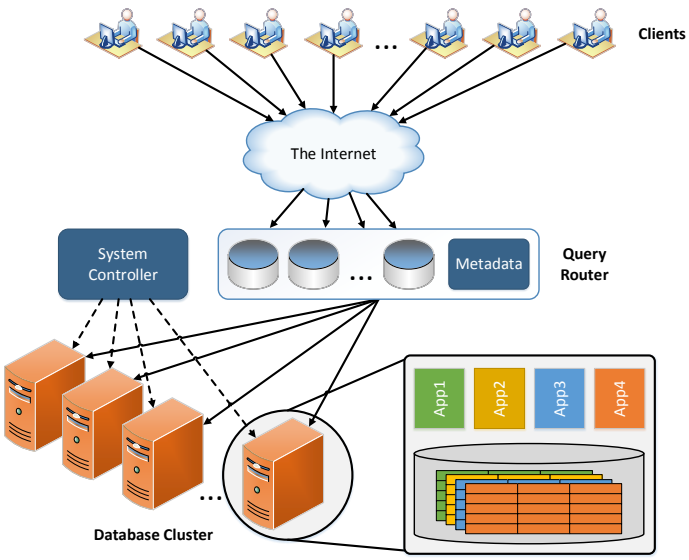

Figura 3.2 Arquitetura do Sistema Zephyr [Elmore et al., 2011b]

A estratégia de migração abordada permite que a cópia-fonte e a cópia destinatária executem transações simultaneamente, diminuindo o processo de propagação de transações após a restauração da cópia de destino, ou melhor, a cópia final. Para isso, as páginas do banco de dados são particionadas em dois SGBDs, onde o SGBD-fonte possui todas as páginas de dados afetadas por transações. Ao iniciar a migração, o SGBD destinatário adquire as páginas, sob demanda, que foram modificadas no SGBD-fonte.

Durante a propagação das páginas de dados, e para garantir o critério de "serializabilidade", o *Zephyr* usa o protocolo *two phase commit* (2PC) para sincronização, ordenamento das mensagens e garantia de entrega. A migração termina quando todas as páginas de dados, afetadas pelas transações, são transferidas da fonte para o destino. A migração é focada na transferência das páginas de dados, para agregar confiabilidade e isolamento dos dados. O *Zephyr* utiliza uma estratégia para controle de concorrência

fortemente baseada em bloqueios. Com isso, garante-se que não haja ocorrência de anomalias de acesso corrente nas páginas de dados.

O *Zephyr* foi implementado como extensão de um banco de dados OLTP chamado H2 [Database, 2014]. Todos os experimentos realizados foram conduzidos com origem nesta extensão. O *Zephyr*, contudo, possui uma solução de caráter intrusiva, pois necessita de acesso aos componentes do SGBD, *buffer pool* e páginas de dados. Isso dificulta a aplicação deste sistema em diversos outros SGBDs.

ProRea [Schiller et al., 2013]

O trabalho encerra uma abordagem, denominada *ProRea*, que implementa um sistema de migração baseado em *Live Migration* projetada para SGBDs multi-inquilinos. O objetivo principal do trabalho é adaptar a estratégia de *Live Migration* para alcançar o mínimo de interrupção do serviço, alta eficiência e menor sobrecarga de propagação de atualização após a transferência do inquilino. Para isso, a estratégia de migração combina as formas reativa e proativa. No trabalho, utilizaram-se aplicações que executam cargas de trabalho OLTP nos inquilinos. A Figura 3.3 ilusta uma visão geral da arquitetura do sistema *ProRea*.

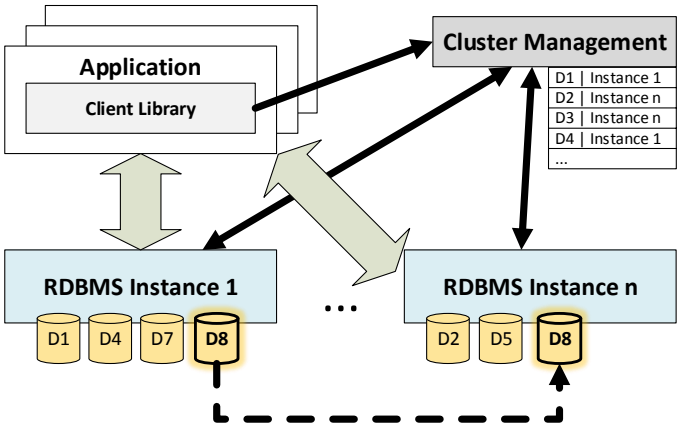

Figura 3.3 Arquitetura do Sistema ProRea [Schiller et al., 2013]

A estratégia de migração do *ProRea* é similiar à estratégia empregada no *Zephyr* [Elmore et al., 2011b], ao permitir que as transações sejam executadas na cópia de origem e na cópia de destino, usando modificações em páginas de dados. O mecanismo para controle de concorrência utilizado, contudo, é diferente do que foi implementado no *Zephyr*.

ProRea foi implementado para operar em um protocolo *multi-version concurrency control* (MVCC) [Schiller et al., 2013]. Neste sentido, o *ProRea* alcança isolamento baseado em *snapshot*, permitindo que o *ProRea* tenha maior grau de concorrência do que o *Zephyr*, pois este último utiliza um protocolo baseado no *strict 2-phase locking* (S2PL).

Para a realização experimental do *ProRea*, foi necessário implementar um protótipo integrado ao PostgreSQL, pois este SGBD utiliza *snapshot isolation*. Os experimentos foram conduzidos para demonstrar uma comparação entre uma abordagem puramente reativa e a implementada pelo *ProRea* que é reativo e proativo; no entanto, a solução oferecida é fortemente integrada com os SGBDs, tipos de cargas de trabalho, controle de concorrência etc. Além disso, como outras desvantagens, é possível mencionar que esse trabalho não garante o critério de "serializabilidade" e não previne todas as anomalias de acesso concorrente.

Estratégia proposta por [Ahmad and Bowman, 2011]

O trabalho exibe o uso de técnicas de aprendizagem de máquina para executar predição quando cargas de trabalho são executadas em ambientes de bancos de dados multi-inquilinos. Neste ensaio, o modelo de compartilhamento adotado foi o de SGBD compartilhado. Para condução dos experimentos, sobre a utilização das técnicas de aprendizagem de máquina, foram testados os métodos *Linear Regression* e *Gaussian processes*. As métricas de predição foram baseadas no emprego de CPU e taxa de transferência de dados em memória secundária.

A técnica de predição serve-se das métricas, apresentadas, diante da combinação dos inquilinos que estão inseridos no mesmo recurso compartilhado, observando cada carga de trabalho de um inquilino. Apesar de não indicar uma arquitetura, a solução deve possuir um processo monitor para coletar informações dos SOs e dos SGBDs. Essas informações são necessárias para treinar os métodos de aprendizagem de máquina. Perante um conjunto de cargas de trabalho de inquilinos que querem compartilhar um recurso, a técnica de predição responde ao estimado valor da métrica que intenta prever.

Para conduzir os experimentos, a solução assume que existe alguma estratégia de monitoramento a fim de obter informações para treinar os modelos de predição para, consequentemente, conseguir informações de predição após tal etapa de treinamento. Durante os experimentos, foram medidos os erros de cada método de predição, servindo-se do *Mean Relative Error* (MRE) [Santos et al., 2013]. Para ambas as métricas especifi-

cadas, o método baseado em *Linear Regression* obteve menor valor MRE, ou seja, teve melhor eficiência comparado ao outro método, *Gaussian processes*.

Este trabalho, entretanto, é um estudo que explora, somente, se servindo do modelo SGBD compartilhado. Os experimentos realizados utilizam múltiplas instâncias do TPC-C/H para simular um ambiente multi-inquilino. O estudo não explora o comportamento da solução expressa em outros *benchmarks*, como o TPC-C, Twitter, Wikipedia, YSCB etc. O objetivo do experimento é fazer com que a carga de trabalho não exceda ou sobrecarregue os recursos do ambiente. Portanto, este ensaio não satisfaz as garantias impostas por um SLA ou, em outras palavras, QoS.

Slacker [Barker et al., 2012]

[Barker et al., 2012] descrevem uma solução de migração de bancos de dados que satisfaz algumas restrições de cargas de trabalho. Emprega o modelo multi-inquilino de SGBD compartilhado e tem como objetivo minimizar o impacto da diminuição de desempenho durante o processo de migração das bases de dados. Para conduzir a especificação e validação do trabalho proposto, foi implementado um *middleware* denominado *Slacker*. A migração é inteiramente baseada na forma *Live migration* para as características do SGBD MySQL.

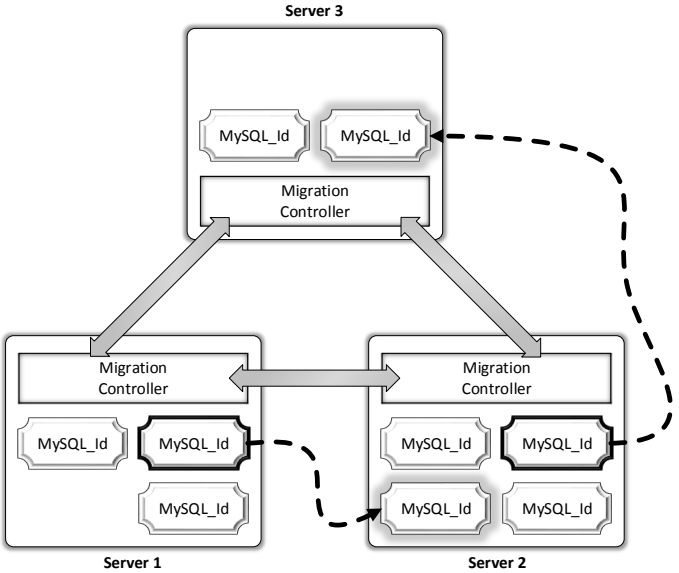

Figura 3.4 Arquitetura do Sistema Slacker [Barker et al., 2012]

A Figura 3.4 ilustra a arquitetura proposta, onde se observa que o *Slacker* interage

com o MySQL utilizando *InnoDB tables*. Para o modelo multi-inquilino, um inquilino é alocado em um SGBD MySQL dedicado. Cada inquilino tem total controle sobre o funcionamento do MySQL, como a criação de bancos de dados, tabelas e usuários. Um inquilino cria um diretório de dados contendo todo os dados do MySQL, incluindo dados de tabelas, *logs* e arquivos de configuração.

A escolha do modelo multi-inquilino, empregado no *Slacker*, é motivada por dois aspectos. O primeiro é o aumento do isolamento entre os inquilinos, pois cada SGBD só pode possuir um inquilino. Isso previne situações em que páginas de dados possam competir no caso de cargas de trabalho de outros inquilinos. O segundo facilita a transferência dos dados, uma vez que cada inquilino é totalmente separado pelo SGBD. Como a implementação foi orientada a MySQL e o ambiente é completamente homogêneo, isso facilita ainda mais a remoção dos dados de um SGBD para outro.

O *migration controller*, presente em cada servidor, monitora todos os inquilinos alocados na VM e gerencia a migração. Os inquilinos são representados por um identificador único e global. A comunicação é realizada entre os *migration controller* e ocorre de forma *peer-to-peer*. Todas as migrações ocorrem sob demanda, transferindo os inquilinos por meio dos componentes *migration controller*. Ademais, o *Slacker* possui um modelo multi-inquilino bastante limitado, um SGBD para um inquilino.

SmartSLA [Xiong et al., 2011]

Neste ensaio, é mostrado o *SmartSLA*, sistema inteligente para o gerenciamento de recursos, que considera a carga de trabalho e o custo da infraestrutura. O trabalho aborda o problema do gerenciamento de recursos virtuais em ambientes de computação em nuvem e utiliza técnicas de aprendizagem de máquina. As técnicas são empregadas para descrever um modelo de desempenho do sistema por meio de uma abordagem orientada a dados. Um modelo preditivo é usado para a alocação de recursos de *hardware*, tais como CPU e memória, assim como para definir o número de réplicas do sistema.

Do ponto de vista do modelo multi-inquilino empregado, cada máquina física pode possuir diversas VMs. Cada VM contém um ou mais SGBDs. Já os SGBDs podem gerenciar vários bancos de dados, caracterizando, portanto, o uso do modelo de SGBD compartilhado. O *SmartSLA* assume que existem dois tipos de clientes: um ouro e um prata, que compartilham recursos de *hardware*. Quando a demanda de carga de trabalho dos clientes muda, o *SmartSLA* adiciona ou remove réplicas do banco de dados, de forma

a atender os requisitos dos clientes. O sistema é monitorado continuamente e os recursos alocados podem mudar periodicamente em cada intervalo de tempo. A Figura 3.5 ilustra a arquitetura do *SmartSLA*.

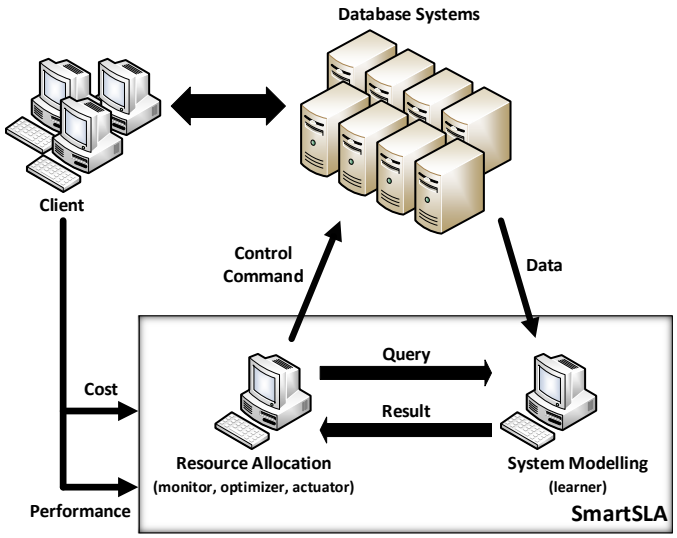

Figura 3.5 Arquitetura do Sistema SmartSLA [Xiong et al., 2011]

O *SmartSLA* consiste de dois componentes principais: módulo de modelagem do sistema e módulo de decisão para alocação de recursos. Os clientes fornecem informações sobre o custo e o desempenho para o *SmartSLA*. O módulo de modelagem coleta informações sobre o estado dos bancos de dados e o módulo de decisão emite comandos para controlar os recursos. O módulo de modelagem utiliza técnicas de aprendizagem de máquina para construir um modelo que descreva o potencial de margem de lucro para cada cliente com diferentes alocações de recursos.

O módulo de modelagem serve-se de um modelo baseado no relacionamento entre os recursos alocados e o custo esperado para cada cliente. O módulo de decisão de alocação ajusta, dinamicamente, os recursos para maximizar os lucros. Outra característica do trabalho é o uso de replicação, implementada com o protocolo de cópia primária de forma assíncrona, enquanto o *SmartSLA* trata apenas da questão do número de réplicas necessárias para garantir o SLA.

Com relação aos aspectos de predição, o trabalho discute o uso de três métodos: *Linear Regression*, *Regression Tree* e *Boosting*. Durante os experimentos, foi mostrado o desempenho de cada um destes métodos. O método que indicou melhor eficiência,

ou seja, menor percentual de erro, foi o *Boosting*. Com relação ao modelo de alocação, do trabalho, é usado o método de predição para dar suporte à tomada de decisão para alocação dos recursos. O problema de alocação é dividido em dois níveis: recursos e número de réplicas. O objetivo é distribuir as réplicas onde menos violarem SLA.

Os modelos mostrados no ensaio, entretanto, não podem ser usados em provedores públicos, como, por exemplo, a Amazon, pois necessita de acesso aos recursos físicos da infraestrutura. Outro ponto a ressaltar é que o trabalho não demonstra os aspectos de elasticidade; somente aspectos de alocação e replicação. Portanto, há dificuldade de aplicação do ensaio em ambientes onde, para contemplar os SLAs, os recursos de infraestrutura devam ser adicionados dinamicamente.

SWAT [Moon et al., 2013]

O trabalho propõe o uso de uma técnica, baseada em balanceamento de carga, para tratar o problema da sobrecarga em ambientes de banco de dados multi-inquilinos. É utilizada também uma estratégia de replicação voltada para o gerenciamento multi-inquilino de banco de dados, visando a melhorar o desempenho quando uma réplica está sobrecarregada. No modelo multi-inquilino, abordado neste ensaio, um SGBD pode hospedar várias réplicas de banco de dados distintos. Ademais, é importante ressaltar que não é possível ter duas ou mais réplicas do mesmo inquilino no mesmo SGBD.

Para conduzir a implementação, uma arquitetura foi proposta e pode ser observada na Figura 3.6. Quando uma carga de trabalho chega ao *middleware* proposto, é direcionada para a réplica menos sobrecarregada. Para auxiliar o critério de decisão de qual réplica deve responder à carga de trabalho, existe um componente que monitora os recursos de todo o ambiente e envia as informações para o módulo de balanceamento de carga. Para melhorar o desempenho na execução da carga de trabalho, o texto proposto utiliza uma estratégia que verifica a possibilidade de que algumas operações possam ser executadas em paralelo em vez de em ordem sequencial.

Cada inquilino possui uma réplica primária e uma ou mais réplicas secundárias. A estratégia de replicação adotada é de forma assíncrona para garantir maior tolerância a falhas e disponibilidade. Todos os tipos de operações (escrita e leitura) podem ser executados em uma réplica primária, de forma assíncrona. Uma réplica secundária, porém, só executa operações de escrita quando são propagadas pela réplica primária. Do ponto de vista de consistência, o trabalho implementa consistência forte sobre as réplicas.

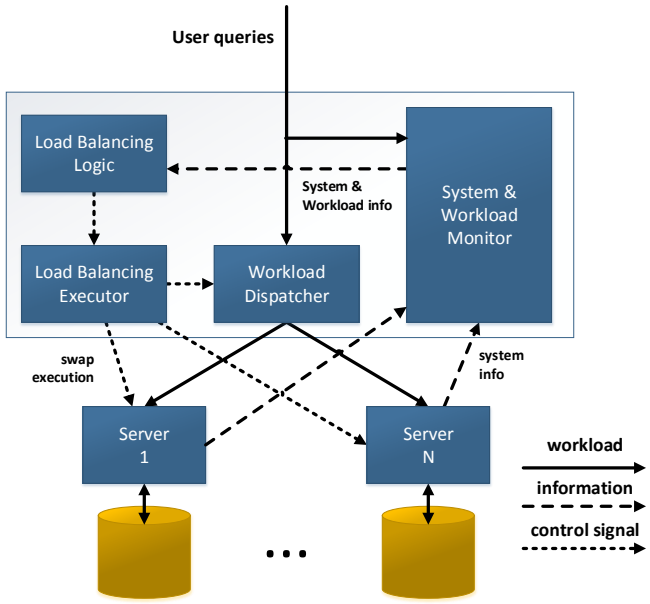

Figura 3.6 Arquitetura do Sistema SWAT [Moon et al., 2013]

O trabalho apresentado, contudo, não elimina violações de SLA quando a réplica de um inquilino está sobrecarregada, pois não implementa mecanismos de provisionamento ou elasticidade. Ambientes em nuvem são caracterizados por possuirem uma variedade de carga de trabalho, empregando a elasticidade para diminuir eventuais sobrecargas, ao mesmo tempo em que gerencia melhor o uso dos recursos. Portanto, este trabalho não privilegia a característica essencial dos ambientes de computação em nuvem. Além disso, ante os experimentos e a conclusão, é enfatizado que o trabalho proposto nem sempre elimina todos os tipos de sobrecarga nos inquilinos, principal motivo da ocorrência de violações de SLA.

Solução proposta por [Hatem A. Mahmoud and El-Abbadi, 2012]

Este trabalho relata um estudo que analisa os impactos bons e ruins entre a largura de banda de memória e disco, quando inquilinos, de característica OLAP, compartilham recursos. Cada SGBD pode possuir vários inquilinos, expressando o modelo multi-inquilino de SGBD compartilhado. A ideia do ensaio é investigar as características em cargas de trabalho onde inquilinos fazem uso intensivo de IO, ou seja, *IO-bound*. Para conduzir a investigação, foi projetado e implementado um conjunto de algoritmos e heurísticas que objetivam balancear o uso dos dois recursos críticos, memória e disco. A Figura 3.7 expõe

o esboço da solução proposta.

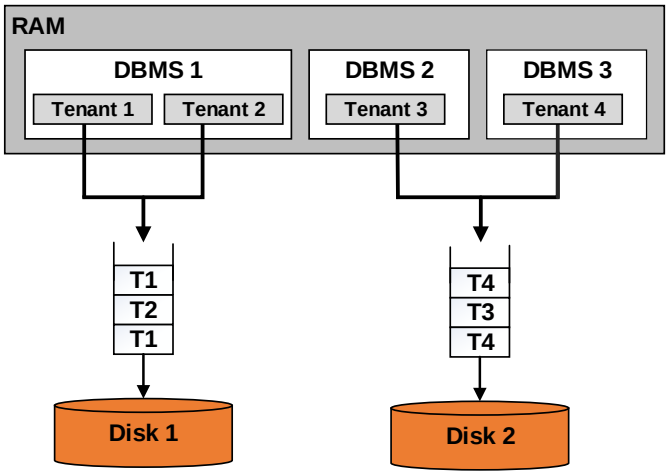

Figura 3.7 Solução Proposta por [Hatem A. Mahmoud and El-Abbadi, 2012]

Do ponto de vista de contribuição, o trabalho propõe uma abordagem para minimizar o número total de servidores necessários para hospedar um conjunto de inquilinos, satisfazendo os SLAs de cada qual. Um algoritmo de aproximação, chamado *Greedy Memory Reduction* (GMR), foi proposto para obter uma aproximação ótima global do tamanho dos recursos que serão necessários para os inquilinos que estão executando em um SGBD. Uma heurística, denominada *Greedy Tenant Consolidation* (GTC), que consolida um conjunto de inquilinos em um SGBD, utiliza, como critério de decisão, os SLAs e o estado do *buffer pool* do SGBD.

A definição do problema trata da alocação de inquilinos de característica OLAP nos recursos do ambiente. Neste sentido, não são mostrados aspectos de migração e provisionamento, mas apenas para consolidação de inquilinos em recursos de VM e SGBD. Os experimentos foram realizados no intuito de alocar os inquilinos diante da demanda das cargas de trabalho e dos recursos de servidores do ambiente. O trabalho é caracterizado como forma de alocação ótima de recursos em ambientes que possuem inquilinos com tais cargas de trabalho.

A solução apresentada, no entanto, é orientada a inquilinos que possuem características de carga de trabalho OLAP que fazem uso intensivo de IO. O foco da proposta

é gerenciar os recursos do ambiente, para que os inquilinos não excedam os limites de recursos à disposição, diminuindo assim a ocorrência de violações de SLA. Limitando os recursos no ambiente, para um conjunto de inquilinos, de certo, modo são soluções voltadas para ambientes que não possuem dinâmica elevada. Portanto, a abordagem não cobre alguns aspectos de computação em nuvem, por exemplo, elasticidade ou provisionamento de recursos.

RepliC [Sousa and Machado, 2012]

Este ensaio indica uma solução, denominada *RepliC*, para replicação elástica, que cobre banco de dados relacional multi-inquilino com QoS. Como características, o *RepliC* implementa elasticidade, disponibilidade e desempenho. Sobre o modelo multi-inquilino, *RepliC* suporta que cada VM contenha um ou mais SGBDs; cada SGBD, por sua vez, pode possuir um ou mais bancos de dados. A técnica de elasticidade adiciona e remove réplicas de acordo com a carga de trabalho dos inquilinos. Técnicas de monitoramento são utilizadas para reaver informações sobre os recursos do ambiente e dar suporte à tomada de decisões para a replicação. Além destas informações, são também coletadas outras sobre os bancos de dados.

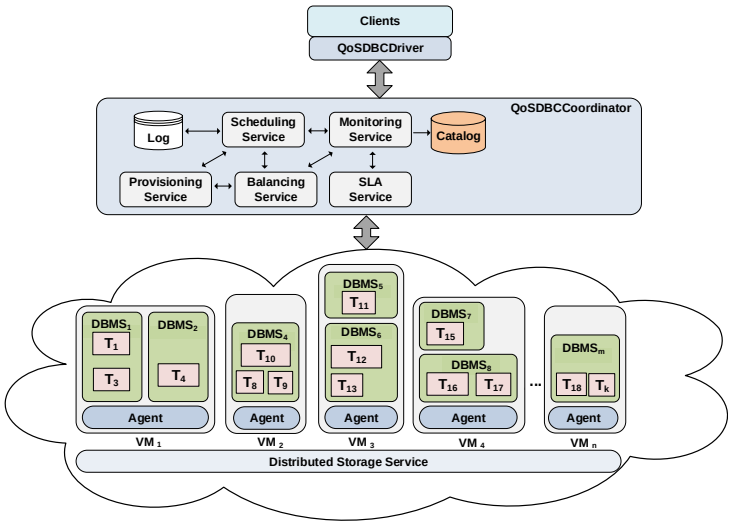

Figura 3.8 Arquitetura Implementada pelo RepliC [Sousa and Machado, 2012]

O *RepliC* foi desenvolvido como extensão da arquitetura, chamada QoSDBC, descrita

em [Sousa et al., 2012]. Uma visão geral da arquitetura que originou o *RepliC* pode ser observada na Figura 3.8. *RepliC* monitora, também, informações sobre os inquilinos, suas cargas de trabalho e os limites dentro dos quais eles podem ser executados sem violação do SLA. A interferência entre inquilinos é ocasionada pela violação do SLA de qualquer um dos inquilinos. A interferência dos inquilinos é modelada por um conjunto de regras. Uma regra de interferência entre dois inquilinos P e Q, denotada por $P \rightarrow Q$, define a carga de trabalho limite dentro da qual a violação do SLA não ocorre. As regras são utilizadas como critério de decisão para adição/remoção de réplicas.

Sobre os aspectos de consistência, o *RepliC* garante consistência forte, utilizando o critério de "corretude" *one-copy serializability*. Para a replicação, é implementado o conceito de grupos virtuais para gerenciar as réplicas. A estratégia divide réplicas de cada banco de dados em dois grupos: um de leitura, que executa, somente, transações de leitura, e outro de atualização, que executa transações de atualização. O grupo de atualização, por sua vez, é dividido em dois tipos de réplicas: primária e secundária. Réplicas de leitura são chamadas de *slaves*. Transações de atualização executam-se de forma síncrona no grupo de atualização; neste caso, toda cópia primária é sempre atualizada imediatamente. As atualizações, em cópias primárias, são enviadas, de forma assíncrona, para o grupo de leitura.

A principal contribuição do *RepliC* é uma abordagem de replicação elástica com QoS. Portanto, ante a ocorrência de violações de SLA, é utilizada a replicação para descentralizar uma eventual sobrecarga, criando cópia dos recursos e utilizando estratégias de sincronização para manter a coerência das réplicas. A replicação implica redundância. Neste sentido, utilizam-se mais recursos do que o necessário. Nenhuma estratégia de alocação da nova réplica é abordada, uma nova réplica sempre fica em uma nova VM que é provisionada, o que pode subutilizar os recursos de VM.

Framework proposto por [Lang et al., 2012]

O trabalho proprõe uma plataforma de serviços de banco de dados, servindo-se de um *framework* de provisionamento e escalonamento que otimiza os custos das operações de banco de dados, ao mesmo tempo em que privilegia SLOs baseados em desempenho. O trabalho é motivado pelos desafios de otimizar *clusters* de SGBD que são guiados por SLOs de desempenho. Dentre os desafios, pode-se destacar: a heterogeneidade de

recursos para suprir as necessidades dos inquilinos e decidir quantos recursos devem ser provisionados para que os inquilinos possam cumprir seus SLOs, ao mesmo tempo em que se utiliza um número mínimo de recursos.

Para conduzir a solução, implementada via *framework*, um fluxo foi definido: (1) analisar os inquilinos e entender seu desempenho e categorizar os inquilinos. Assim, dados os SLOs, determinar o número máximo de inquilinos, de uma determinada categoria, que podem ser escalonados para utilizar certos recursos; (2) calcular os limites que as cargas de trabalho, das categorias de inquilinos, podem atingir,modelos de caracterização de desempenho; (3) dados esses modelos, encontrar uma estratégia ótima para escalonar as cargas de trabalho, levando em consideração os inquilinos e os recursos. Além disso, verificar a necessidade de realizar provisionamento de recursos quando os atuais estão escassos, visando sempre à redução de recursos ociosos no ambiente. Uma versão mais detalhada do fluxo pode ser obtida na Figura 3.9.

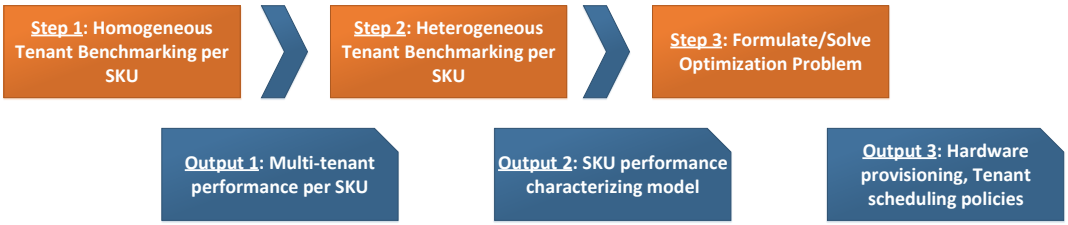

Figura 3.9 Fluxo Definido por [Lang et al., 2012]

No que concerne à avaliação, o trabalho se serve do modelo multi-inquilino de SGBD compartilhado, e de múltiplas instâncias do TPC-C/H, para simular o ambiente multi-inquilino. O trabalho é orientado a recursos, visando a garantir a utilização mínima de tais recursos diante dos SLOs. O fluxo apresentado necessita de uma etapa que analisa os inquilinos. Não existem, no entanto, experimentos que comprovem a eficácia da estratégia com diversos tipos de inquilinos e que possuam estruturas diferentes. Ambientes multi-inquilinos públicos compartilham recursos entre inquilinos oriundos de vários usuários, que podem ter estruturas e cargas de trabalho bastante heterogêneas. Neste sentido, a presença de inquilinos diferentes pode comprometer a etapa de classificação ou categorização dos inquilinos.

3.3 ANÁLISE COMPARATIVA ENTRE OS TRABALHOS RELACIONADOS

Para conduzir esta análise comparativa, foram considerados os requisitos que têm como objetivo manter a QoS para serviços de bancos de dados em nuvem. A plataforma proposta por [Elmore et al., 2011a] especifica uma técnica elástica para balanceamento de carga e migração. São expostos e discutidos vários modelos multi-inquilinos e, dentre eles, a plataforma utiliza o modelo de SGBD compartilhado. O estudo realizado, contudo, não possui nenhuma avaliação experimental que demonstre a eficácia da solução proposta. Neste sentido, dificultam-se a adoção e o conhecimento sobre a viabilidade da solução ante a ausência de experimentação. [Ahmad and Bowman, 2011] oferecem uma solução que usa técnicas de aprendizagem de máquina para executar predição de cargas de trabalho em ambientes multi-inquilinos de SGBD compartilhado. A solução, contudo, não propõe uma arquitetura para implementação e os experimentos realizados utilizam múltiplas instâncias do TPC-C/H. Assim, não demonstram o comportamento da solução preditiva em um ambiente onde os inquilinos são heterogêneos.

Já os trabalhos realizados por [Elmore et al., 2011b] e [Schiller et al., 2013] implementam estratégias de migração baseadas nas técnicas de *Live Migration*. Ambos os trabalhos, no entanto, são soluções intrusivas, que dependem das características dos SGBDs que foram utilizados no processo de validação e experimentação. Ambos os trabalhos são semelhantes, diferenciando apenas na implementação da técnica para controle de concorrência. O controle de concorrência utilizado por [Elmore et al., 2011b] é baseado em protocolos pessimistas que usam bloqueios, o que fornece maior confiabilidade nos critérios de isolamento e consistência, mas possui menor desempenho. Já o controle de concorrência usado por [Schiller et al., 2013] é baseado em *snapshot*. Assim, propõe-se aumentar o desempenho da solução por não utilizar bloqueios, mas não elimina a ocorrência de problemas relacionados ao acesso concorrente dos dados, o que afeta os aspectos de consistência dos inquilinos.

Em [Barker et al., 2012] é demonstrado um *middleware* para migração, baseado em *Live Migration* de banco de dados, com a utilização do modelo multi-inquilino de SGBD compartilhado. O modelo multi-inquilino sugerido é bastante limitado, pois a relação utilizada é um SGBD por inquilino. A migração serve-se desta limitação do modelo multi-inquilino para maior rapidez na transferência do inquilino. Assim como os trabalhos propostos por [Elmore et al., 2011b] e [Schiller et al., 2013], a solução apresentada por [Barker et al., 2012] também é intrusiva e fortemente baseada nas características

internas do MySQL. [Xiong et al., 2011] apresentam uma solução para o gerenciamento de banco de dados em sistemas replicados, servindo-se do modelo multi-inquilino de SGBD compartilhado. O trabalho emprega técnicas de aprendizagem de máquina para alocação dos inquilinos nos recursos do ambiente; no entanto, a solução não pode ser adotada em provedores públicos, como, por exemplo, a Amazon, pois é necessário o acesso aos recursos físicos da infraestrutura em nuvem.

O trabalho elaborado por [Moon et al., 2013] propõe o uso de uma técnica de balanceamento de carga para o problema da sobrecarga em ambientes que utilizam o modelo multi-inquilino de SGBD compartilhado. Assim como em [Xiong et al., 2011], este trabalho utiliza técnicas de replicação, entretanto, a estratégia implementada não elimina violações de SLA quando uma réplica está sobrecarregada, pois não são utilizados mecanismos de provisionamento ou elasticidade. Já a solução proposta por [Hatem A. Mahmoud and El-Abbadi, 2012] investiga características em cargas de trabalho onde inquilinos fazem uso intensivo de IO. Nesse trabalho, foi utilizado um conjunto de heurísticas e algoritmos para balancear o uso de recursos de memória principal e disco. O modelo multi-inquilino usado é o de SGBD compartilhado, no entanto, o trabalho é orientado a inquilinos de características OLAP. Não são abordados aspectos de migração e provisionamento, tratando de alocar inquilinos com base nas cargas de trabalho e dos recursos do ambiente.

[Sousa and Machado, 2012] exprimem uma solução para replicação elástica em ambientes multi-inquilino de SGBD compartilhado. A replicação é utilizada para, diante da ocorrência de SLA, descentralizar uma eventual sobrecarga, criando cópias dos recursos. Essa redundância emprega mais recursos do que o necessário. Além disso, existe sobrecarga para garantir a coerência das réplicas. O trabalho, contudo, não aborda nenhuma estratégia inteligente para alocação de uma nova réplica. Uma nova réplica sempre é colocada em uma nova VM que foi provisionada. Em [Lang et al., 2012], é proposto um *framework* para provisionamento e escalonamento em operações de bancos de dados, contemplando SLOs baseados em desempenho. O trabalho propõe o modelo multi-inquilino de SGBD compartilhado e a avaliação foi guiada por múltiplas instâncias do TPC-C/H. Ademais, o trabalho não expõe resultados em ambientes cujos inquilinos são heterogêneos, uma vez que, no modelo apresentado, existe uma etapa de classificação dos inquilinos. Além disso, o trabalho é orientado a recursos e o objetivo é garantir a utilização mínima de tais recursos diante dos SLOs.

Nenhum dos trabalhos relacionados estudados atende totalmente aos requisitos

definidos nesta investigação, no que se refere à QoS para banco de dados multi-inquilino em nuvem, requisitos estes contemplados pelo PMDB. A Tabela 3.2 traz um resumo deste comparativo.

Trabalhos	Gerenciamento de Recursos	Predição	Multi-inquilino	QoS	Migração	Consistência	Alocação
Elmore et al, 2011a	Elasticidade	Não possui	SGBD compartilhado	Sim	LM	Não menciona	Uso de BC / AM
Zephyr	Elasticidade	Não possui	SGBD compartilhado	Sim	LM intrusivo	Forte	Uso de BC
ProRea	Não mencionado	Não possui	SGBD compartilhado	Sim	LM intrusivo	Eventual	Não menciona
Ahmad and Bowman, 2011	Não possui	Sim, AM	SGBD compartilhado	Não	Não possui	Não menciona	AM na alocação
Slacker	Provisionamento	Não possui	VM comp., um SGBD por inquilino	Sim	LM intrusivo	Não menciona	Não menciona
SmartSLA	Apenas adição de VM	Sim, métodos de regressão	SGBD compartilhado	Sim	Não possui	Forte	Monitoramento de recursos e réplicas
SWAT	Não possui	Não possui	SGBD compartilhado	Sim	Não possui	Forte	BC / execução paralela de tipos de consultas
Hatem A. Mahmoud and El-Abbadi, 2012	Não possui	Não possui	SGBD compartilhado	Não	Não possui	Não menciona	Heurísticas para balancear o uso dos recursos
RepliC	Replic. elástica / Provisionamento	Não possui	SGBD compartilhado	Sim	Não possui	Forte	Sempre provisiona VM
Lang et al., 2012	Provisionamento	Não possui	SGBD compartilhado	Sim	Não possui	Forte	Escalonamento de consultas

Legenda
VM - Máquina Virtual | AM – Aprendizagem de Máquina | BC – Balanceamento de Carga | LM – *Live Migration*

Tabela 3.2 Análise Comparativa entre os Trabalhos Relacionados

3.4 CONCLUSÃO

Neste capítulo, foram discutidas as principais abordagens encontradas na literatura para a QoS em bancos de dados em ambientes distribuídos. Foram destacados os requisitos para a QoS neste ambiente e foi procedida a uma análise comparativa detalhada destas abordagens. Apesar da grande quantidade de trabalhos relacionados, nenhum destes privilegia os diversos aspectos de QoS de dados em nuvem, pontos estes previstos neste trabalho. O próximo capítulo traz a abordagem PMDB e são explanadas suas características, sua especificação, arquitetura e os algoritmos desenvolvidos.

CAPÍTULO 4

ABORDAGEM PARA QUALIDADE DE SERVIÇO EM BANCOS DE DADOS MULTI-INQUILINOS EM NUVEM

O PMDB (*Performace Multi-tenant Databases*) é uma abordagem para manter a QoS em SGBDs multi-inquilinos e reduzir o número de recursos utilizados em nuvens computacionais. Para isso, são adotadas técnicas de migração, baseada em *Live Migration*, para mover um dado inquilino, quando sobrecarregado para outro local, onde ele possa operar com a QoS acertada por meio do SLA. Uma técnica de alocação é usada para decidir qual o melhor local para que um dado inquilino possa operar, mantendo as garantias do SLA. PMDB também se serve de uma técnica de predição, para antecipar a migração quando observado que o inquilino pode violar o SLA. Todas essas técnicas se servem de uma estratégia de monitoramento que coleta informações sobre o ambiente e as cargas de trabalho dos inquilinos.

Para melhorar a utilização dos recursos, o PMDB adota o modelo multi-inquilino de SGBD compartilhado. A técnica de migração, abordada neste trabalho, é a *Live Migration*, de consistência forte, de forma não intrusiva. Com isso, o tempo de inatividade do inquilino é diminuído, durante a migração, ao mesmo tempo em que aumenta a confiabilidade da consistência dos dados. Apesar de o PMDB usar migração para manter QoS, nos casos em que o ambiente esteja totalmente sobrecarregado, a abordagem proposta provisiona recursos para reduzir a sobrecarga do ambiente.

4.1 MODELO MULTI-INQUILINO

Para garantir QoS, adota-se o modelo multi-inquilino de SGBD compartilhado, pois este é mais utilizado que o de tabela e o de banco de dados compartilhado [Elmore et al., 2011a]. Além disso, é o modelo que expressa a melhor relação entre a utilização de recursos, desempenho e segurança [Barker et al., 2012]. Por outro lado, embora o modelo de VM compartilhada denote menor interferência entre os inquilinos, seu nível de compartilha-

mento também é menor. Assim, de acordo com o modelo de SGBD compartilhado, que tem uma base de dados correspondente a um inquilino no sistema, cada máquina virtual pode executar múltiplos SGBDs. Apesar de exigir menos recursos de infraestrutura, esta abordagem aumenta a interferência entre os inquilinos, já que funciona com mais inquilinos no mesmo SGBD.

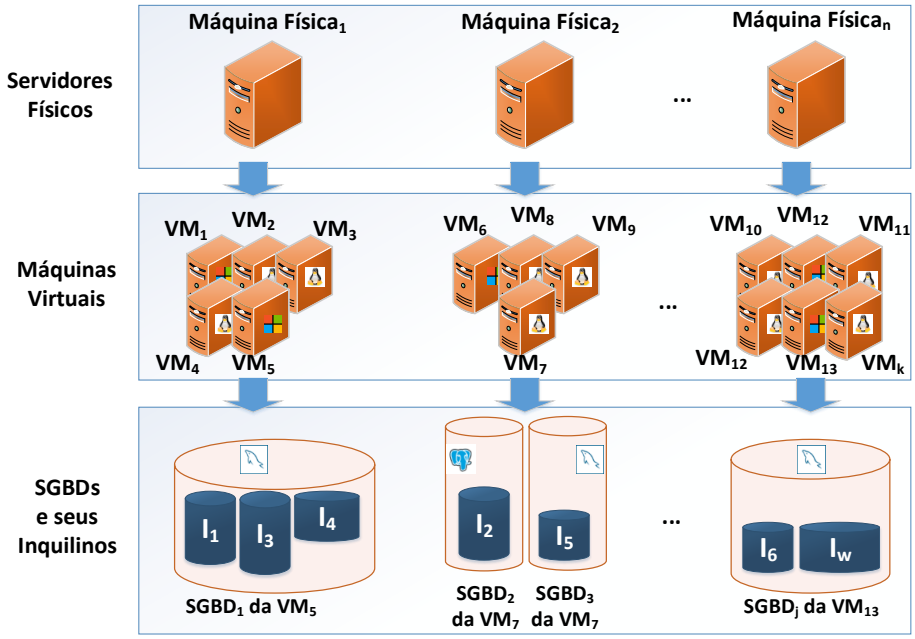

Figura 4.1 Modelo Multi-inquilino Utilizado pelo PMDB

Melhor visualização do modelo multi-inquilino utilizado pelo PMDB pode ser obtida na Figura 4.1. Neste modelo, cada VM possui instância de um SGBD. Cada SGBD, por sua vez, hospeda uma quantidade variável de inquilinos, de acordo com a capacidade dos recursos na VM e da carga de trabalho. Por exemplo, o $SGBD_1$ dentro da VM_5 gerencia os inquilinos I_1, I_3 e I_4. Neste trabalho, é válido ressaltar que cada inquilino é representado por um só banco de dados. Portanto, a abordagem proposta para manter a QoS em bancos de dados multi-inquilinos não implementa estratégias de replicação.

4.1.1 Interferência entre Inquilinos

De acordo com o que já foi apresentado em [Moreira et al., 2012], o modelo multi-inquilino de SGBD compartilhado, utilizado no PMDB, pode indicar interferência entre inquilinos.

Uma interferência ocorre quando o aumento na carga de trabalho, de um inquilino, altera o desempenho dos outros inquilinos que estão sendo gerenciados pelo mesmo SGBD. Em [Moreira et al., 2012] elaborou-se uma metodologia para analisar a interferência em ambientes de SGBDs multi-inquilinos. Dessa metodologia, originou-se um conjunto de experimentos mostrando que a escolha do SGBD ajuda a diminuir a ocorrência de interferências. Ademais, para reduzir as interferências, é necessário identificar o nível de interação dos inquilinos e, consequentemente, melhorar a alocação dos inquilinos de acordo com suas interferências. Portanto, é importante uma análise do perfil do inquilino. Além disso, é crucial isolar inquilinos suscetíveis a este problema.

Alguns trabalhos usam um modelo analítico ou executam experimentos reais para avaliar a alocação de recursos e diminuir as interferências [Ahmad and Bowman, 2011] [Lang et al., 2012]. Estas abordagens, contudo, ocasionam uma sobrecarga significativa com as alterações das cargas de trabalho. Nestas abordagens, faz-se necessário recalibrar e revalidar o modelo ou executar um novo conjunto de experimentos para selecionar uma nova alocação de recursos. Durante o período de adaptação, o sistema pode ser executado com uma configuração ineficiente e não garantir a qualidade [Vasić et al., 2012].

Outros trabalhos, como os de [Sousa and Machado, 2012] e [Vasić et al., 2012], utilizam uma estratégia baseada na coleta de informações sobre o estado do sistema durante a execução, e reutilizam estas informações para garantir uma alocação aproximada para futuras cargas de trabalho semelhantes às cargas anteriores. Apesar desta estratégia ser independente de infraestrutura, as interferências são identificadas quando são ocasionadas violações de SLA. Ante a ocorrência de violações de SLA de um dado inquilino, uma regra de interferência entre dois inquilinos é adicionada no histórico do sistema. Essa regra também denota o limiar da carga de trabalho que ocasionou a regra.

Em virtude destas limitações, PMDB utiliza uma técnica de análise preditiva combinada com uma técnica de alocação para inquilinos, ambos baseados em SLA. Estas técnicas são independentes de componentes de *hardware* e *software*, podendo ser utilizadas em qualquer infraestrutura de nuvem. Tanto a técnica de análise preditiva quanto a técnica de alocação utilizam informações obtidas pelo sistema de monitoramento presente na arquitetura do PMDB. Para ambas as técnicas e de todo o conjunto de informações monitoradas, são destacados o uso de informações sobre as cargas de trabalho e, o estado das VMs e dos SGBDs.

Para exemplificar o tratamento da interferência entre inquilinos, considere-se o

seguinte exemplo. Suponha-se a existência do inquilino A no PMDB. No PMDB, os valores de SLA são baseados nos tempos de resposta médios das consultas que os inquilinos executam. O valor de SLA do inquilino A é $SLA_A = 0.5s$. O tamanho do inquilino é $T_A = 500MB$. O PMDB possui uma Função de Estimativa da Qualidade (FEQ) para presumir o tempo de migração. Essa função será mais bem descrita na seção que relata o modelo de QoS. A função de estimativa da qualidade é, resumidamente, baseada no tamanho do inquilino e nos recursos computacionais da VM. Suponha-se que a estimativa de migração do inquilino A seja $E_A = 1min$.

A técnica de análise preditiva continuamente recupera informações do sistema de monitoramento. A identificação de uma violação de SLA, ou seja, que o inquilino entrou em interferência, é dada pela comparação do SLA do inquilino, tempo estimado de migração e o momento em que a violação ocorrerá. Imagine-se que a técnica de predição identificou que o inquilino A aumentará seu tempo médio de resposta das transações em $0.8s$ nós próximos $2min$. O tempo estimado de migração é inferior ao tempo em que poderá ocorrer a violação de SLA, ou seja, $SLA_A < 0.8s$ e $E_A < 2min$. Neste caso, pode-se migrar o inquilino A para uma VM menos sobrecarregada e, caso não exista, uma VM é provisionada.

4.2 MODELO PARA QUALIDADE DE SERVIÇO

Modelos que especificam a QoS para bancos de dados em nuvem são fundamentais para estabelecer um critério de eficácia da solução proposta. Esses modelos são utilizados como guia durante a avaliação de tais soluções. Na academia, existem diversos modelos para SLA e QoS em serviços em nuvem [Malkowski et al., 2010] [Schnjakin et al., 2010] [Comellas et al., 2010]; no entanto, estes modelos são gerais e não tratam características do gerenciamento de dados em nuvem. Existem modelos para SLA e qualidade específicos para serviços de bancos de dados que são descritos em [Chi et al., 2011] [Yang et al., 2009] [Xiong et al., 2011]. Estes modelos, entretanto, não satisfazem algumas características do gerenciamento de dados em nuvem, tais como métricas específicas para serviços de banco de dados e propõem, apenas, parte de uma solução para qualidade, por exemplo, a definição para um SLA ou abordagens para penalidades. Ademais, estes trabalhos não utilizam técnicas eficazes de monitoramento específicas para SGBDs, fundamentais para tratar a qualidade no âmbito das nuvens computacionais.

Do ponto de vista do modelo de qualidade, PMDB utiliza um subconjunto do

Service-level Agreement for Database (SLADB), um modelo de qualidade proposto em trabalhos anteriores [Sousa et al., 2011] [Moreira et al., 2013]. Este modelo define uma solução para auxiliar a QoS, pois aborda distintas questões, tais como definição do SLA, técnicas de monitoramento e penalidades. Ele combina variadas técnicas de monitoramento, permitindo a melhoria do uso dos serviços de banco de dados em nuvem e contempla características do gerenciamento de dados, tais como tempo de resposta, vazão, disponibilidade e consistência. O SLADB é bastante completo, o PMDB só utiliza um subconjunto de todas essas especificações. Esta tese se interessa pelo aspecto de tempo de resposta. Além disso, não se utilizam aspectos de penalidade neste trabalho.

4.2.1 Especificação

Um SLA contém informações sobre o modelo de receita do provedor de serviço em nuvem, a determinação do valor recebido pelo provedor para cumprir o SLA, bem como penalidades ou multas em caso de falhas. A definição de um SLA é uma tarefa não trivial, pois estes acordos devem refletir o valor econômico, bem como as exigências de serviço ao usuário. Adicionalmente, SLAs têm que descrever as condições comuns de negócios, tais como parâmetros de desempenho e disponibilidade, métricas de avaliação, contabilidade e questões jurídicas, bem como prazos de contrato [Malkowski et al., 2010]. Esta proposta de tese aborda apenas aspectos técnicos relacionados aos parâmetros de desempenho e métricas de avaliação.

As propostas para SLAs indicam muitas diferenças, mas é possível identificar uma estrutura geral comum: informações sobre as partes envolvidas, parâmetros do SLA, métricas utilizadas para calcular os parâmetros do SLA, algoritmo para calcular os parâmetros do SLA, objetivo de nível de serviço (SLO) e ações a serem realizadas em caso de violação do acordo [Schnjakin et al., 2010]. O modelo SLADB define SLA como:

definição 1.1 *Um SLA para Serviço de Banco de Dados em Nuvem é composto por informações das partes envolvidas, métricas do SLA, SLOs, algoritmos para calcular as métricas do SLA e penalidades.*

Informações sobre as partes envolvidas referem-se ao contrato entre o provedor e o inquilino. As métricas do SLA estão relacionadas aos itens a serem monitorados, por exemplo, tempo de resposta e vazão. Já o SLO contém os limites pré definidos para o parâmetro, por exemplo, tempo de reposta menor do que 5 milissegundos. Para cada

parâmetro é definido uma forma de calculá-lo, por exemplo, tempo médio e penalidades referentes às ações em caso de não conformidade dos SLOs; neste caso, multas.

De acordo com [Chi et al., 2011], as métricas de SLA para banco de dados em nuvem devem otimizar o sistema, tratar aspectos relevantes para o gerenciamento de dados e contemplar as características do modelo de computação em nuvem. Apesar de o SLADB utilizar as métricas de tempo de resposta, vazão, disponibilidade e consistência, o escopo do PMDB será restrito apenas à métrica de tempo de resposta. É importante ressaltar, entretanto, que o provedor de serviço pode optar por fornecer apenas algumas destas métricas no seu SLA, visto que existem métricas relacionadas, por exemplo, tempo de resposta e vazão. No modelo deste trabalho, um SLO é associado a métrica, conforme destacado a seguir:

- *tempo de resposta*: o tempo de resposta médio, em segundos, para cada transação, durante um período de tempo t.

4.2.2 Monitoramento das Métricas do SLA

Do ponto de vista do usuário, um serviço de banco de dados opera, de forma satisfatória, se os requisitos de desempenho e disponibilidade que o usuário contrata são cumpridos. O primeiro ponto é traduzir os requisitos de desempenho definidos pelo usuário em um conjunto comum de métricas que pode ser obtido por meio do monitoramento. Exemplos de tais métricas incluem o tempo de resposta e vazão. O tempo de resposta médio é uma das formas mais comuns para verificar a qualidade do serviço.

Em contextos distintos, é importante estabelecer metas mais gerais para o QoS [Schroeder et al., 2006], tais como o percentil, em que $x\%$ dos tempos de resposta é inferior a um valor y. O percentil é solicitado pelos usuários como componente de um SLA, por exemplo, para garantir que pelo menos 90% das transações dos clientes tenham um tempo de resposta abaixo de um limite especificado [Entrialgo et al., 2011]. Para cada métrica do SLA, deve-se utilizar um algoritmo para calcular as métricas do SLA. O PMDB implementa a seguinte estratégia:

- *tempo de resposta*: percentil $x\%$ dos tempos de resposta inferior a um valor y durante um período t.

4.2.3 Função de Estimativa da Qualidade (FEQ)

Para auxiliar no processo de decisão sobre a configuração da nuvem, formaliza-se uma função cujo objetivo é verificar a necessidade de realocar um inquilino. Como entrada, a Função de Estimativa da Qualidade (FEQ), implementada no PMDB, utiliza conhecimentos recuperados de experimentos realizados sobre a transferência de inquilinos entre VMs de mesma capacidade de recursos computacionais. É válido ressaltar que essa transferência foi realizada em ambiente controlado e livre de interferência entre inquilinos. Nos experimentos, varia-se o tamanho do inquilino e verifica-se o tempo levado para transferir o inquilino, restaurar, propagar as atualizações e redirecionar as conexões das aplicações. Com suporte nesses experimentos, foi possível constituir tal função.

Na análise das informações, percebe-se que o comportamento do tamanho do inquilino em relação ao tempo de migração era linear. Neste sentido, utiliza-se a técnica de Regressão Linear [Das and Kempe, 2008] como método para elaborar a função. Seja T_i o tamanho de armazenamento do inquilino i, a equação 4.1 irá determinar o tempo estimado para transferir, completamente, o inquilino i para uma máquina de mesma capacidade computacional. A variável a representa a função de inclinação da reta do gráfico resultante do experimento. A inclinação da linha de regressão linear é dada por meio de pontos de dados nos eixos x (tamanho do inquilino) e y (tempo de migração). A inclinação é a distância vertical dividida pela distância horizontal entre dois pontos quaisquer na linha, que é a taxa de mudança ao longo da linha de regressão. Já a variável b calcula o ponto no qual uma linha irá interceptar o eixo y, usando valores de x e y. O ponto de interseção é baseado em uma linha de regressão de melhor ajuste plotada pelos valores de x e y conhecidos.

$$\mathrm{FEQ}(\mathrm{T}_i) = (a * T_i) + b, \text{ onde } a = \bar{y} - b\bar{x} \text{ e } b = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sum (x-\bar{x})^2} \tag{4.1}$$

O resultado de equação 4.1 para um inquilino T_i indicará o tempo ou frequência de aplicação do serviço de predição para este inquilino. Por exemplo, suponha-se que o resultado da função FEQ para o inquilino T_2 foi de 50 segundos. Neste sentido, é necessário obter a predição da carga deste inquilino, pelo menos, nos próximos 50 segundos mais uma margem de segurança. Essa função é crucial para que o PMDB possa antecipar a visualização de uma possível violação de SLA.

4.3 TÉCNICAS UTILIZADAS NO PMDB

Neste seguimento, serão discutidos os detalhamentos das três técnicas utilizadas para manter a qualidade de serviço em bancos de dados multi-inquilinos nas nuvens computacionais. Tais técnicas foram implementadas, como serviços, na arquitetura usada pelo PMDB.

4.3.1 Técnica de Alocação

A técnica proposta de alocação, utilizada pelo PMDB, tem como objetivo escolher o melhor SGBD para receber um dado inquilino que deverá ser migrado, baseando-se nos tempos de resposta das consultas e no SLA do inquilino. Em alguns casos, não será possível alocar um dado inquilino em outra VM. Estes casos ocorrem quando o ambiente já está sobrecarregado e a mudança do inquilino para outra VM irá comprometer o rendimento do próprio inquilino, e para os que estão na outra VM. Para estes casos, será necessário realizar uma expansão dos recursos do ambiente por meio do provisionamento de VMs. A técnica de alocação foi implementada como um serviço no PMDB, denominado Serviço Alocação, a ser mostrado na seção que descreve a arquitetura do PMDB.

Para desenvolver a técnica de alocação do PMDB, um modelo matemático de otimização foi formalizado. Como variáveis e entradas para o modelo se utilizam dados de inquilinos e suas respectivas cargas de trabalho. O modelo proposto é orientado a SLA e utiliza a métrica do tempo de resposta médio de execução das últimas consultas do inquilino. O objetivo do modelo é encontrar uma VM que possa receber o inquilino que gerou interferência, ou seja, aquele inquilino que pode violar o SLA. Neste sentido, a técnica tenta observar a possibilidade de encaixar o inquilino em um recurso. Se o modelo encontra o melhor SGBD para hospedar o inquilino e o SGBD é o que ele está no momento, supõe-se que o ambiente está sobrecarregado. Neste caso, uma nova VM terá que ser provisionada para hospedar este inquilino.

Suponha-se que os SGBDs sejam enumerados de $\{1 \dots N\}$ e os inquilinos do SGBD_j, $\forall 1 \leq j \leq N$ sejam enumerados em $\{1 \dots N_j\}$, onde N será o número de SGBD existente no ambiente, N_j é a quantidade de inquilinos no SGBD_j. Dado um inquilino l que necessita ser migrado, onde l é inquilino do SGBD_k. Foram definidos MT_{ij} como o tempo médio de resposta das últimas p consultas executadas do inquilino i no SGBD_j, onde p é um parâmetro de entrada, SLA_{ij} será o valor do SLA para o inquilino i no

SGBD$_j$ e MD_j será a diferência média entre MT_{ij} e SLA_{ij}, $\forall 1 \leq i \leq N_j$ do SGBD$_j$. Assim, antes de migrar, MD_j é dado por:

$$MD_j = |\frac{\sum_{i=1}^{N_j}(MT_{ij} - SLA_{ij})}{N_j}|$$

Por esse motivo, se SGBD$_i$ é o alvo do processo de migração, é estimado que MD'_j é dado por:

$$MD'_j = \begin{cases} |\frac{\sum_{i=1}^{N_j}(MT_{ij}-SLA_{ij})+(MT_{lk}-SLA_{lk})}{N_j+1}|, & \text{Se} \quad j \neq k, \\ MD_j, & \text{Caso contrário.} \end{cases}$$

Então, o objetivo é encontrar o x, tal que:

$$x = max_{1 \leq i \leq N}\ MD'_i$$

Se $x \neq k$, então z é migrado para o SGBD$_x$. Caso contrário, será necessário o provisionamento de uma nova VM, e o inquilino z, por sua vez, será migrado para esta nova VM. A Figura 4.2 ilustra o exemplo de um cenário onde há três VMs; cada VM possui um SGBD, que, por sua vez, possuem dois inquilinos.

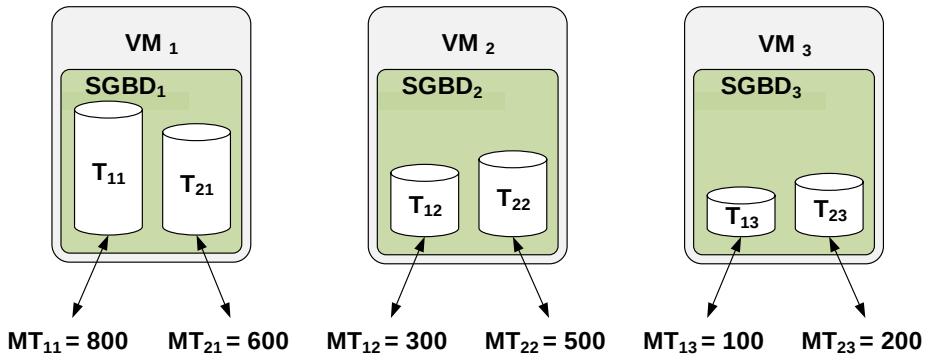

Figura 4.2 Aplicação da Técnica de Alocação [Moreira et al., 2014]

A Tabela 4.1 mostra os valores de cada inquilino no tocante aos seus respectivos SLAs.

Para demonstrar o funcionamento da técnica, foi utilizada a alocação prévia ilustrada na Figura 4.2 e os valores de SLA descritos na Tabela 4.1. Encontra-se o cenário

Inq_{ij}	T_{11}	T_{21}	T_{12}	T_{22}	T_{13}	T_{23}
SLA_{ij}	700	700	600	600	600	600

Tabela 4.1 Valores de SLA para os Inquilinos do Cenário de Alocação

onde o inquilino T_{11} entrou em interferência e será migrado para outra VM no ambiente. Inicialmente, calculou-se a métrica MD para cada SGBD; logo, tem-se que $MD_1 = 0$, $MD_2 = 200$ e $MD_3 = 450$.

Em seguida, tem-se que o MD' para cada SGBD, obteve-se que $MD'_1 = 0$, $MD'_2 = 100$ e $MD'_3 = 266,66$. Desse modo, achou-se $x = 3$, significando que z deverá ser migrado para $SGBD_3$.

4.3.2 Técnica de Predição

Na técnica de predição do PMDB aplica-se a teoria do Scale-Space [Witkin, 1983] para auxiliar as estratégias de migração em recursos dinâmicos, em particular, bancos de dados. Esta teoria permite representar séries temporais de uma forma matemáticamente correta [Sezgin and Davis, 2006], uma vez que fornece uma representação qualitativa do sinal como uma multiescala de medição [Santos et al., 2013], permitindo, assim, interpretações em escalas finas e grossas dos dados.

Por meio da escolha de uma escala adequada, o Scale-Space é uma técnica poderosa para eliminar ou diminuir, consideravelmente, a presença de informações irrelevantes de um sinal. Essas informações irrelevantes podem induzir à tomada de decisão errônea ou tardia por ambos os métodos, proativos e reativos. É importante ressaltar que o Space-Scale provê garantias matemáticas sem necessidade de estruturas adicionais quando o sinal é simplificado. Além disso, novos picos e depressões não são introduzidos no processo e a amostragem original da série temporal é mantida inalterada ao longo do tempo. Esta é uma diferença significativa entre esta abordagem e as demais representações semelhantes multiescala, o que também facilita o acoplamento de outros tratamentos sobre o sinal, como, por exemplo, a subamostragem.

O PMDB combina dois métodos de predição amplamente utilizados *Support Vector Machines for Regression* (SVR) e *Autoregressive Integrated Moving Average* (ARIMA) para auxiliar na construção de uma solução robusta na ocorrência de ruídos e capaz de tratar cenários reativos e proativos [Moreira et al., 2014]. ARIMA e SVR são representa-

tivos, respectivamente, das áreas de aprendizado de máquina e de análise estatística em séries temporais. O PMDB faz um esforço na tentativa de tirar o máximo proveito deles, a fim de maximizar os benefícios que podem ser extraídos em nossa análise preditiva.

Para fazer análises preditivas, o PMDB especifica as informações que são monitoradas, como, por exemplo, a carga de trabalho dos inquilinos. Em particular, o PMDB tem interesse em verificar o desempenho dos inquilinos ao longo do tempo. Neste sentido, a técnica de predição se serve do Serviço de Monitoramento para obter todas essas informações. A técnica preditiva do PMDB é proativa com base em ARIMA e SVR implementados no Serviço de Predição da arquitetura do PMDB.

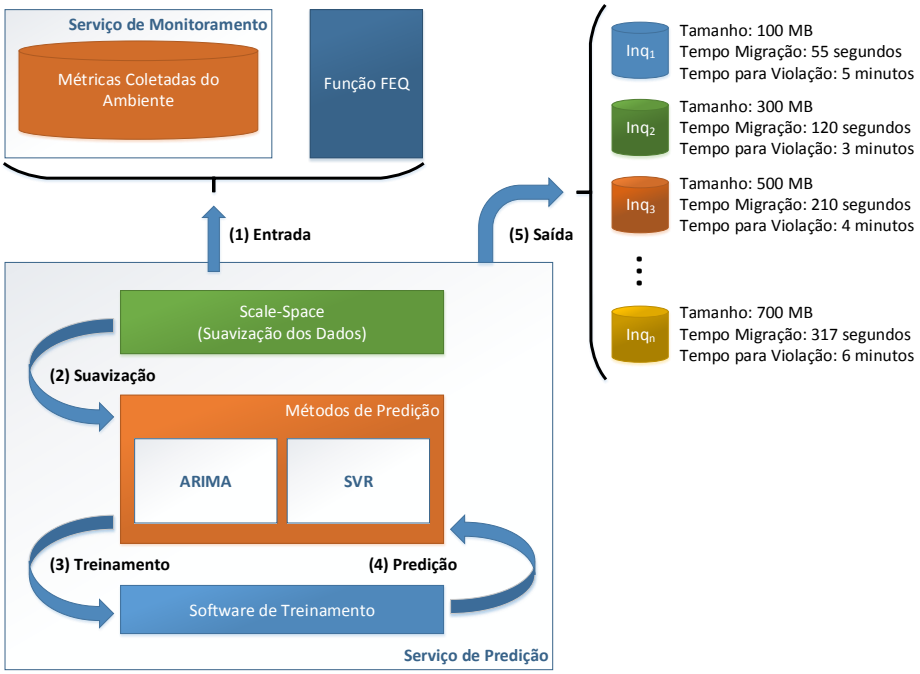

Figura 4.3 Detalhamento do Serviço de Predição Utilizado pelo PMDB

A Figura 4.3 ressalta o funcionamento detalhado da técnica de predição do PMDB. Além disso, destacam-se o fluxo e os cinco passos necessários para concretizar uma análise preditiva de um conjunto de inquilinos. O primeiro passo, (1) entrada, realiza a solicitação de informações do Serviço de Monitoramento e aplicação da função *FEQ*. O Serviço de Predição necessita das informações sobre a carga de trabalho dos inquilinos com o qual se intenta realizar aspectos de predição. O Serviço de Monitoramento do PMDB fornece informações sobre todas as consultas SQL que foram executadas para cada inquilino, além das informações sobre o desempenho destas consultas. Além

das informações sobre a carga de trabalho dos inquilinos, o Serviço de Predição calcula, por meio da função FEQ, a estimativa do tempo de migração para cada inquilino-alvo do processo de predição.

Com dados obtidos sobre a carga de trabalho do inquilino e informações acerca do desempenho das consultas, um conjunto de dados é usado para elaborar uma série que representa o gráfico de desempenho do inquilino em relação ao tempo de resposta médio das consultas. Com isso, aplica-se o passo (2), suavização, objetivando eliminar pontos não significativos. Neste sentido, reduz-se o número de pontos a serem treinados e tratados pelos métodos de predição, ao mesmo tempo em que não se perde a significância dos dados de desempenho do inquilino. Após o tratamento do sinal, são aplicados os passos (3 e 4), passagem dos dados do sinal suavizados para o *software* de treinamento. A etapa de treinamento se dá pela execução dos métodos, por meio da entrada de dados e tendo como saída uma janela de predição da carga de trabalho do inquilino para cada método. É válido ressaltar que, para cada inquilino, haverá duas informações de predições, pois são utilizados dois métodos. O PMDB leva em consideração o método que mostrou o maior tempo médio de resposta.

Como saída, passo (5), a técnica de predição entrega informações sobre cada inquilino, seu tamanho, seu tempo de migração e o instante de tempo de violação. Estas informações servem de suporte à tomada de decisão e podem ser usadas em situações reativas ou proativas, dependendo do ponto de vista em que são invocadas. No caso do PMDB, a técnica de predição é aplicada continuamente, caracterizando-se como abordagem proativa. Caso, porém, haja uma violação de SLA que não foi prevista pelos métodos de predição, a técnica também é ativada diante da ocorrência da violação de SLA, mostrando um aspecto de abordagem reativa.

4.3.3 Técnica de Migração

Existem duas formas básicas de migração: *Stop and copy* e *Live Migration* [Barker et al., 2012]. *Stop-and-Copy* é uma estratégia simples de migração. O banco de dados fica bloqueado enquanto está sendo copiado para outro local. Depois que a cópia é restaurada, as requisições ao banco de dados podem ser respondidas. A técnica de migração proposta é baseada fortemente em *Live Migration*. A *Live Migration* é uma forma imediata de migração, que não faz bloqueio em todas as chamadas ao banco de dados e fornece o mínimo de interrupção do serviço de banco de dados.

Na solução proposta de migração, utilizam-se agentes em cada VM. Esses agentes são usados para executar comandos na VM e nos SGBDs que estão em cada VM. Os comandos são executados nos SGBDs para realizar o *backup* do inquilino, criar esquemas do inquilino, restaurar e transferir o inquilino. A transferência do inquilino é feita por meio do sistema de arquivo distribuído da nuvem. Tolerância a falhas e desempenho são, portanto, alcançadas pelo uso deste meio de persistência e infraestrutura de comunicação.

Quando existe uma necessidade de executar o processo de migração de um inquilino, a solução proposta sinaliza a intenção de executar a operação de *backup* do inquilino. O *backup* do inquilino inicia, somente, quando todas as transação ativas terminam. Quando o *backup* inicia, todas as novas transações mantêm sua execução na cópia antiga e salvas para serem propagadas para a nova cópia do inquilino quando esta for restaurada. Quando termina o *backup*, este é transferido, por meio do sistema de arquivo distribuído, para a nova localização do inquilino, ou seja, para a nova VM. Nessa nova VM, o agente inicia a operação de restauração do inquilino. É importante ressaltar que, durante as operações de *backup*, transferência e restauração, o inquilino não fica inativo e responde às solicitações das aplicações.

Quando o processo de restauração do inquilino se concretiza, se inicia uma inativação do inquilino. Isso é necessário para garantir a consistência do inquilino em sua nova localização, pois todas as transações iniciadas após a operação de *backup* do inquilino são propagadas para o inquilino em sua nova localização. Ademais, o tempo de inativação do inquilino termina quando todas as transações necessárias são propagadas e quando todas as conexões de aplicações são direcionadas para a nova localização do inquilino.

4.3.4 Consistência e Tolerância a Falhas

Assim como o Zephyr [Elmore et al., 2011b], o PMDB emprega as mesmas características de *Live Migration*. Por conseguinte, tanto o Zephyr quando o PMDB utilizam mecanismos persistentes e confiáveis de comunicação e deixando a cargo do SGBD a garantia da consistência e o isolamento dos seus inquilinos. Diferentemente do Zephyr, o PMDB é implementado como uma solução não intrusiva, podendo ser utilizado em qualquer SGBD. O Zephyr necessita de acesso aos componentes do SGBD, *buffer pool* e páginas de dados. Qualquer técnica de migração, para ser confiável e prover consistência forte, deve garantir o critério de "serializabilidade" mesmo na ocorrência de falhas arbitrárias.

O PMDB implementa consistência forte como uma solução não intrusiva.

4.3.4.1 Isolamento Para demonstrar que o PMDB garante o critério de "serializabilidade", procura-se mostrar que a ocorrência de manipulação de dados fantasmas é impossível. O problema de manipulação de dados fantasmas ocorre quando transações escrevem em um dado que é acessado por alguma transação concorrente [Elmore et al., 2011a]. Assuma-se, por contradição, que o problema da manipulação fantasma, durante o processo de migração, é possível. O processo de migração só poderá ser iniciado quando todas as transações ativas forem efetivadas. Seja T_1 e T_2 duas transações ativas, após o início da migração, tais que T_1 possui uma operação e T_2 está escrevendo em pelo menos um dado usado por T_1. T_1 e T_2 não podem executar tais operações em um mesmo SGBD, pois o SGBD irá tratar tais situações, por exemplo, por meio da utilização de algum protocolo que garanta o critério de "serializabilidade". Ao contrário do Zephyr, o PMDB não permite que uma mesma transação execute na cópia de origem e na cópia de destino. Portanto, T_2 não pode manipular um dado usado por T_1, pois se T_2 estiver escrevendo algum dado enquanto T_1 estiver executando; então, isso implica que ambas tiveram acesso ao mesmo dado. Isso resulta em uma contradição.

4.3.4.2 Tolerância a Falhas A abordagem proposta assume que todas as mensagens enviadas utilizam meios confiáveis, que garantem o ordenamento global e entrega das mensagens. Para isso, utilizam-se, para garantir tais características, os sistemas de arquivos distribuídos nas nuvens computacionais. Com isso, o sistema pode tratar a existência de falhas de VMs e de partições de rede [Moreira et al., 2014]. Parte-se da suposição de que, quando uma VM falha, a imagem de disco persistente não é perdida. Tal suposição pode ser reforçada pelos mecanismos de *logging* ou *swapping* dos sistemas de virtualização utilizados pelas nuvens. Na ocorrência de falhas durante a migração, as transações ativas foram efetivadas em um mecanismo persistente confiável da arquitetura, que será descrito na seção de arquitetura. Portanto, mesmo que haja falhas que foram destacadas, o sistema consegue se recuperar para o estado de migração. É válido ressaltar que as transações executam usando *Write Ahead Logging* [Ramakrishnan and Gehrke, 2003] para a recuperação. Após a falha, a VM pode se recuperar a seu estado último confiável, utilizando o mecanismo persistência que contém o armazenamento das transações e as técnicas ARIES [Ramakrishnan and Gehrke, 2003].

4.4 CONCLUSÃO

Este capítulo apresentou PMDB, uma abordagem para manter a QoS do multi-inquilinato para bancos de dados em nuvem, cujo propósito é garantir a qualidade com a utilização eficiente dos recursos. Foram elencados quais os pressupostos considerados pelo PMDB e um modelo de qualidade de serviço foi descrito. O modelo multi-inquilino utilizado pelo PMDB permite compartilhar os recursos de forma eficiente. A estratégia de migração preditiva associada às características da infraestrutura em nuvem garante a qualidade do serviço por meio da adição e remoção de VMs ou realocação dos inquilinos. O próximo capítulo trata de questões arquiteturais e de implementação para demonstrar o funcionamento e o fluxo do PMDB.

CAPÍTULO 5

ARQUITETURA E IMPLEMENTAÇÃO

Este capítulo apresenta o PMDB, a arquitetura que foi especificado para o PMDB e os principais detalhes do seu funcionamento. Para isso, são mostrados e discutidos os principais algoritmos que implementam as técnicas de migração, alocação e predição, além de destacar o funcionamento destas técnicas com a solução para o monitoramento presente na arquitetura.

5.1 INTRODUÇÃO

O PMDB foi implementado como uma extensão do *Quality of Service for Database in the Cloud* (QoSDBC) [Sousa et al., 2012]. O QoSDBC implementa o modelo de qualidade descrito nesta tese e simplifica a utilização dos SGBDs em nuvem. Esta camada é desenvolvida como uma PaaS e pode ser facilmente integrada às diferentes infraestruturas. Para alcançar os objetivos deste trabalho, alguns componentes tiveram que ser modificados; outros foram adicionados.

5.2 ARQUITETURA

A arquitetura estendida é dividida em três partes: *QoSDBCDriver*, *QoSDBCCoordinator* e *Agente*. O *QosDBCDriver* é o componente que fornece acesso ao sistema. Este *driver* substitui o *driver* JDBC específico do banco de dados, mas oferece a mesma *interface* sem a necessidade de modificações no SGBD. O *QoSDBCCoordinator* consiste de um conjunto de serviços que tratam do gerenciamento dos inquilinos. O *Agente* é o componente adicionado em cada VM, sendo responsável por coletar, monitorar e interagir com a VM e o SGBD, bem como verificar o estado dos itens monitorados, por exemplo, se estes estão operacionais ou indisponíveis. Uma visão geral da arquitetura do PMDB pode ser observada na Figura 5.1.

O Serviço de Monitoramento gerencia as informações coletadas pelo *Agente*. Este

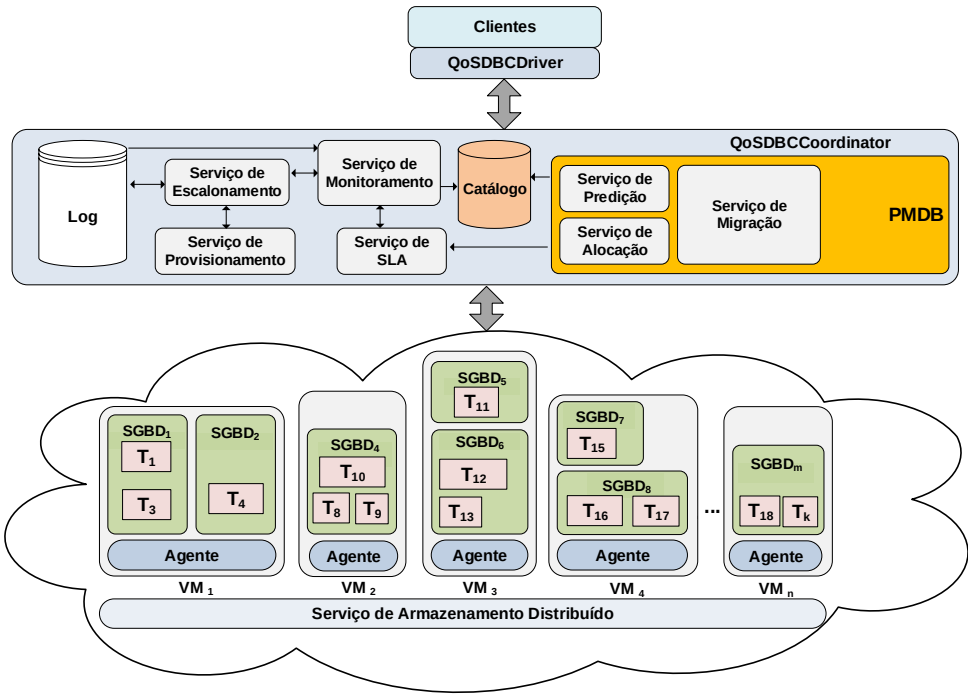

Figura 5.1 Arquitetura do PMDB [Moreira et al., 2014]

serviço monitora informações sobre cada VM e SGBD, estatísticas da carga de trabalho, consulta informações sobre os recursos necessários para os SGBDs e a respeito dos recursos disponíveis na infraestrutura. Essas informações são utilizadas para definir a alocação dos inquilinos dos SGBDs das VM ou provisionar recursos, de forma a garantir a qualidade do serviço. As informações são armazenadas no *Catálogo* e continuamente atualizadas para ajustar os outros serviços. As seguintes informações são monitoradas para cada VM: estado da VM, recursos de CPU, memória e disco. Para cada SGBD, são monitoradas as informações sobre os recursos de CPU, memória, tamanho e número de conexões das bases de dados utilizadas, assim como as métricas definidas no SLA.

A Tabela 5.1 tem como função descrever os SGBDs que estão ativos no ambiente. No ambiente em estudo, quando uma VM inicia, o agente é ativado e, por sua vez, alimenta esta tabela, para que a VM possa ser utilizada. Quando a VM é desligada, o agente remove a linha desta tabela que representa a VM. Na Tabela 5.2, existem dados atualizados, continuamente, pelo Serviço de Monitoramento. Os dados são coletados pelo agentes em todas as VMs. Para conseguir tais informações especificas do SGBD e do inquilino, utilizam-se os dados de usuário e senha de administrador contidos na tabela *db_active*.

Coluna	Funcionalidade
time	*Timestamp* que identifica o ordenamento global das mensagens persistidas nesta tabela
vm_id	Identificador da VM, esse valor deriva o endereço e a porta da VM sinalizadora da mensagem
db_name	Nome do banco de dados (inquilino) que está sendo monitorado
dbms_type	Define o fabricante do SGBD que está hospedando o inquilino, define se é um PostgreSQL ou um MySQL
dbms_user	Nome do usuário administrador do SGBD
dbms_password	Senha do usuário administrador do SGBD
dbms_port	Define a porta VM que recebe conexões para o SGBD

Tabela 5.1 Descrição da Tabela *db_active* Contida no *Catálogo*

Coluna	Funcionalidade
time	*Timestamp* que identifica o ordenamento global das mensagens persistidas nesta tabela
vm_id	Identificador da VM, esse valor deriva o endereço e a porta da VM sinalizadora da mensagem
db_name	Nome do banco de dados (inquilino) que está sendo monitorado
db_size	Tamanho, em *megabytes*, que o inquilino possui neste dado instante de tempo
dbms_connections	Número de conexões que o SGBD, que contém o inquilino, possui
dbms_type	Define o fabricante do SGBD que está hospedando o inquilino, define se é um PostgreSQL ou um MySQL

Tabela 5.2 Descrição da Tabela *db_state* Contida no *Catálogo*

Já a Tabela 5.3, por sua vez, expõe todas as VMs que estão ativas e operantes no ambiente. Assim como na tabela *db_active*, o agente é o responsável por coletar os dados sobre os recursos computacionais da VM. Do mesmo modo, quando a VM é desligada, as informações que identificam a VM são removidas pelo agente. A Tabela 5.4 é, continuamente, atualizada pelo agente contido na VM. Todas as informações são obtidas por meio de comandos do *kernel* do Sistema Operacional instalado na VM. Para não armazenar informações obsoletas das VMs, quando uma VM é desligada, todas as informações pertinentes a VM são removidas.

O Serviço de SLA contém informações sobre o acordo de nível de serviço acertado entre o usuário e o provedor do serviço. Estas informações estão relacionadas às definições do SLA, tais como o tempo de resposta e consistência. Este serviço fornece parâmetros para o Serviço de Monitoramento verificar os valores definidos e ajustar os demais serviços. O Serviço de Provisionamento utiliza informações dos outros serviços para provisionar os

Coluna	Funcionalidade
time	*Timestamp* que identifica o ordenamento global das mensagens persistidas nesta tabela
vm_id	Identificador da VM, esse valor deriva o endereço e a porta da VM sinalizadora da mensagem
mem_total	Total, em *megabytes*, de memória que VM possui
disk_total	Total, em *megabytes*, de memória secundária que VM possui
agent_port	Porta da VM que o agente QoSDBC aguarda por mensagens

Tabela 5.3 Descrição da Tabela *vm_active* Contida no *Catálogo*

Coluna	Funcionalidade
time	*Timestamp* que identifica o ordenamento global das mensagens persistidas nesta tabela
vm_id	Identificador da VM, esse valor deriva o endereço e a porta da VM sinalizadora da mensagem
cpu_free	Percentil do tempo livre de CPU
mem_total	Total, em *megabytes*, de memória que VM possui
mem_free	Total de memória livre, em *megabytes*, que a VM possui
disk_total	Total, em *megabytes*, de memória secundária que VM possui
disk_free	Total de disco livre, em *megabytes*, que a VM possui

Tabela 5.4 Descrição da Tabela *vm_state* Contida no *Catálogo*

recursos de VM. PMDB define quando um inquilino deve ser transferido para uma nova VM, por meio das técnicas de análise preditiva. Este serviço também decide como os recursos devem ser compartilhados entre os vários inquilinos. A infraestrutura em nuvem permite um rápido provisionamento de recursos por meio de técnicas de virtualização, o que melhora o impacto de cargas de trabalho com grandes variações.

O Serviço de Escalonamento, no QoSBDC original, enumera e seleciona os recursos provisionados. Para tanto, o escalonador gerencia as réplicas, garantindo o acesso ao SGBD durante e após o processo de replicação; no entanto, PMDB não trabalha em sistemas replicados. Neste sentido, altera-se esse componente para, somente, gerenciar as transações e direcioná-las para os SGBDs em execução. Cada SGBD é responsável pelo processamento interno e otimização local. Por fim, o Serviço de Escalonamento enumera e seleciona os recursos a serem utilizados para a execução das requisições e o *Log* armazena as transações a serem executadas. A Tabela 5.5 ilustra os campos e suas respectivas funcionalidades que existem na única tabela do banco de dados *Log*.

Coluna	Funcionalidade
time	*Timestamp* que identifica o ordenamento global das mensagens persistidas nesta tabela
vm_id	Identificador da VM, esse valor deriva o endereço e a porta da VM sinalizadora da mensagem
db_name	Nome do banco de dados (inquilino) onde a instrução SQL foi executada
sql	Instrução SQL
sql_type	Tipo da instrução SQL, pode ser SELECT, INSERT, UPDATE ou DELETE
response_time	Tempo de resposta global que instrução SQL executou, tempo coletado pela aplicação
sla_response_time	Valor de SLA imposto ao inquilino
sla_violated	Campo verificador de violação do SLA
connection_id	Identificador da conexão JDBC que elaborou a instrução SQL, no caso, identifica a aplicação
transaction_id	Identificador da transação que originou a instrução SQL
affected_rows	Número de linhas afetadas pela instrução SQL
time_local	Tipo de resposta que a consulta executou localmente no SGBD
in_migration	Campo sinalizador que mostra se a instrução SQL foi executada após o início da migração

Tabela 5.5 Descrição da Tabela *sql_log* Contida no *Log*

As técnicas de migração, predição e alocação encontram-se intimamente ligadas aos demais componentes arquiteturais do QoSDBC. Para cada aplicação, associa-se um SLA descrito pelo modelo de qualidade. Na Figura 5.2, foram destacadas algumas ações que servem para descrever o fluxo e as interações dos componentes do PMDB. Quando o PMDB é iniciado, os agentes começam a monitorar os inquilinos, SGBDs e as VMs que estão no ambiente. Uma vez que os dados do ambiente são monitorados e colocados no *Catálogo* e *Log* (1), o Serviço de Predição começa a ler os dados, treina os métodos de predição e tenta fornecer informações sobre quando ocorrerão violações de SLA para cada inquilino (2).

Quando o *QoSDBCCoordinator*, por meio do Serviço de Predição, identifica uma violação de SLA de um inquilino (3), o Serviço de Migração começa, de forma paralela, o *backup* do inquilino (3.1). Os dados de monitoramento do ambiente são enviados para o Serviço de Alocação. O Serviço de Alocação, por sua vez, verifica se algum SGBD pode receber o inquilino (4). Se isso acontecer, a referência ao SGBD é recuperada em uma dada VM (4.1); caso contrário, uma nova VM é provisionada para receber o inquilino (4.2). Depois disso, o conjunto de transações executadas depois do início do *backup* é executado na cópia de origem do inquilino e armazenado no *Log*. Até a conclusão do

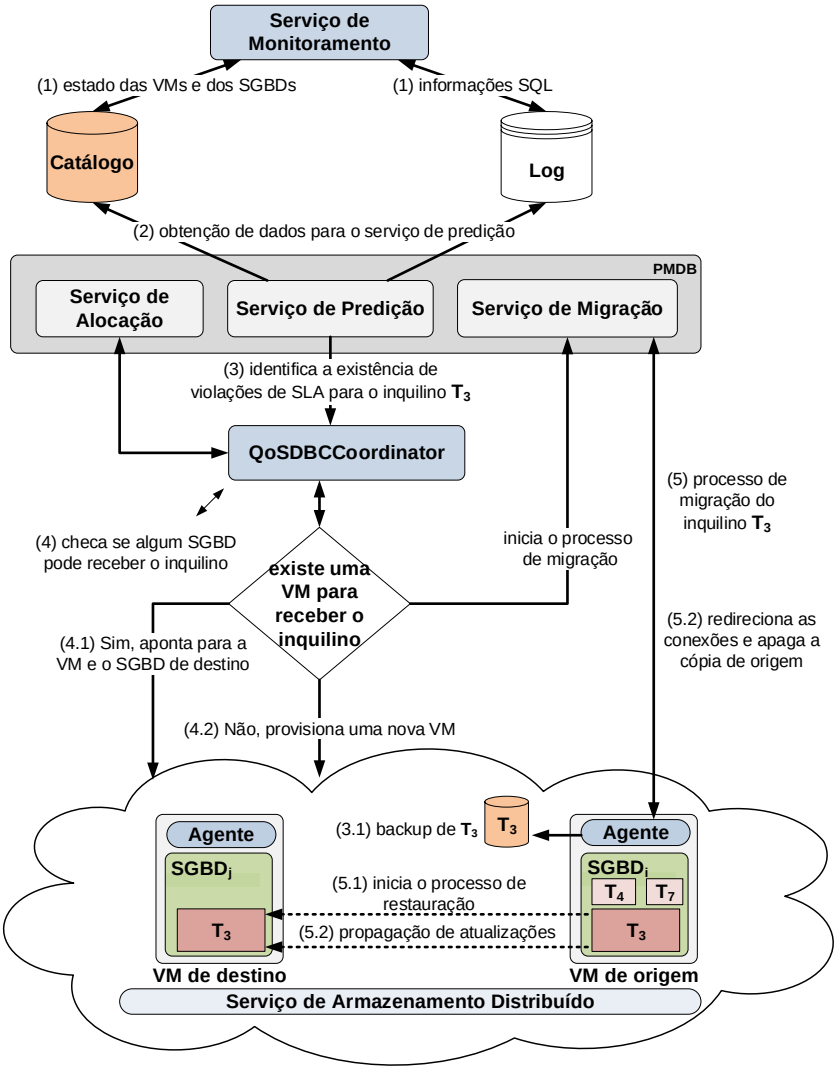

Figura 5.2 Fluxo Geral do PMDB [Moreira et al., 2014]

processo de *backup* (5.1), o inquilino é restaurado na VM de destino.

Para garantir a consistência quando o processo de recuperação for concluído, as transações de atualização, executadas na cópia de origem, são propagadas para a cópia de destino (5.2). Neste momento, há um período de inatividade, pois o inquilino fica indisponível. Quando o inquilino está totalmente migrado, o *QoSDBCCoordinator* sinaliza o redirecionamento das conexões para a nova localização do inquilino, ou seja, a cópia de destino. Por fim, o inquilino de origem é deletado (5.2).

5.2.1 Implementação

Por decisões de projeto, optou-se por desenvolver o PMDB servindo-se da linguagem Java. Características como portabilidade, "reusabilidade", processamento distribuído e *multithreading* são propriedades desta linguagem que favorecem um desenvolvimento confortável, confiável e de altos recursos. A portabilidade está relacionada à independência da plataforma de execução, ou seja, não depende de sistema operacional ou *hardware*. Já a "reusabilidade" está no fato de que, em Java, há diversos componentes, e APIs estão prontas para serem incorporadas à aplicação, aumentando assim a produtividade.

Com relação ao processamento distribuído, Java oferece diversos recursos de comunicação entre componentes, tais como: chamada a funcões remotas (Sockets e RMI) e integração com os protocolos conhecidos de internet (TCP/IP, HTTP, Telnet etc). Assim, torna-se mais fácil o desenvolvimento de aplicações derivadas da arquitetura cliente/servidor. Java possui recursos confiáveis para aplicações *multithreading*, servindo-se de primitivas de sincronização baseadas no uso de monitores. Os mecanismos de sincronização, baseados em monitores da linguagem Java, fornecem um ambiente mais fácil e seguro para o desenvolvimento de aplicações multitarefas.

No que diz respeito ao sistema de comunicação entre componentes arquiteturais na nuvem, optou-se pelo uso da comunicação Java Socket. A implementação Socket de Java é o meio de comunicação mais simples e primitivo desta linguagem, o que proporciona melhor desempenho na troca de mensagens. Pelo fato de que o PMDB foi escrito na linguagem Java, optou-se por utilizar Sockets como meio de comunicação, o que proporciona melhor desempenho na troca de mensagens, ao mesmo tempo em que atende as necessidades de comunicação deste trabalho.

A solução de gestão de infraestrutura da nuvem desenvolvida foi implementada para a Amazon. Foi utilizada a *API* Java fornecida pela *Amazon EC2* para iniciar, terminar ou reconfigurar as VMs. Uma *Amazon Machine Image* (AMI) contendo o SGBD e o agente foi criada, permitindo iniciar uma nova VM rapidamente. Para armazenar e compartilhar os *backups* do banco de dados, foi utilizado o *Amazon Elastic Block Store* (EBS). Para o processo de migração de dados, foi empregada a ferramenta de *backup* de cada SGBD que o PMDB suporta, no caso, MySQL e PostgreSQL. Mais detalhes sobre a implementação dos processos de *backup* e restauração podem sem obtidos em [Moreira et al., 2014] [QoSDBC, 2014].

Os métodos de predição, utilizados no PMDB, foram incorporados da R. R é a união de uma linguagem com um ambiente de desenvolvimento, para cálculos estatísticos e gráficos [Gardener, 2012]. Além disso, R é caracterizado por ser altamente expansível com o uso dos pacotes, que são bibliotecas para funções específicas ou áreas de estudo específicas. Com isso, R é largamente usada entre estatísticos e mineradores de dados para desenvolver *software* de estatística e análise de dados. Experimentos estatísticos e levantamentos de mineradores de dados mostram que a popularidade do R aumentou substancialmente nos últimos anos [Voulgaropoulou et al., 2012] [Contreras-Reyes and Palma, 2013].

A R disponibiliza ampla variedade de técnicas estatísticas e gráficas, incluindo modelação linear e não linear, testes estatísticos clássicos, análise de séries temporais, classificação, agrupamento e outras. Ademais, R é facilmente extensível por meio de funções e extensões, e a comunidade R é reconhecida pelos seus contributos ativos em termos de pacotes. Muitas das funções-padrão do R são escritas no próprio R, o que torna fácil para os usuários seguir as escolhas algorítmicas feitas [de Micheaux et al., 2014]. Usuários da R podem escrever código C ou Java para manipular diretamente objetos R [Tattar, 2013].

5.3 PRINCIPAIS ALGORITMOS

Aqui serão expostos os principais algoritmos que implementam a solução PMDB. Para isso, foram elencados os algoritmos que representam, de forma simplificada, o Serviço de Monitoramento, Serviço de Alocação, Serviço de Migração e o Serviço de Predição. Ademais, algoritmos que compõem o *driver* que as aplicações utilizam e as precípuas funções do Serviço de Monitoramento serão discutidas.

O Algoritmo 1 é uma função executada pelo *Agente* que coleta, principalmente, o estado da VM onde ele se encontra inserido. No modelo multi-inquilino adotado pelo PMDB, uma VM pode possuir nenhum ou vários SGBDs. Neste sentido, a função possui implementação para coletar o número de conexões abertas pelos SGBDs que estão instalados na VM. É válido ressaltar que a função pode recuperar informações de SGBDs PostgreSQL e MySQL. A periodicidade com que essa função lê o estado da VM é especificada, em um arquivo de configuração, no momento em que se instala o *Agente*.

O argumento da periodicidade, representado por *time_to_sleep*, é entrada para

Algoritmo 1 - Função executada pelo *Agente* sob a regência do Serviço de Monitoramento

```
1: function monitor_vm(time_to_sleep)
2:     while true do
3:         cpu_free ← kernel.get_cpu_free()
4:         mem_free ← kernel.get_mem_free()
5:         disk_free ← kernel.get_disk_free()
6:         DBMS[] ← catalog.get_DBMSs(vm_id)
7:         for i ← 1..DBMS.length do
8:             rdbms_connections ← DBMS[i].get_active_connections()
9:             catalog.put_vm_state(vm_id, cpu_free, mem_free, disk_free, rdbms_connections)
10:        end for
11:        sleep(time_to_sleep)
12:    end while
13: end function
```

função *monitor_vm*. O laço especificado na linha (2) reflete que a função executa enquanto o agente estiver ativo. As linhas (3, 4 e 5) representam, respectivamente, a coleta de informações de CPU livre, memória livre e disco livre. Essas informações são recuperadas pelo *kernel* do Sistema Operacional. Já a linha (6) verifica quais instalações de SGBDs existem na VM e monta um vetor de SGBDs. Esse vetor é percorrido no laço que está especificado na linha (7) e, para cada elemento do vetor, obtêm-se o número de conexões ativas no SGBD. Após isso, persistem essas informações no *catálogo*, ação representada na linha (7). O laço executa continuamente; entretanto, existe um atraso para que não sejam coletados dados de forma massiva. Esse atraso é especificado na linha (11), por meio da função *sleep*.

Algoritmo 2 - Função executada pelo *QoSDBCDriver* sob a regência do Serviço de Monitoramento

```
1: function execute_query(connection_id, transaction_id, query)
2:     start_global_time ← System.currentTimeMillis()
3:     SQL_VO ← qosdbc_coordinator.execute(connection_id, transaction_id, query.getSQL())
4:     end_global_time ← System.currentTimeMillis()
5:     global_time ← end_global_time − start_global_time
6:     SLA ← qosdbc.get_sla_service(connection_id)
7:     qosdbc.get_log().put_sql_log(connection_id, transaction_id, SQL_VO, global_time, SLA)
8: end function
```

Já o Algoritmo 2 reflete alguns motivos pelos quais houve necessidades da reescrita de um JDBC *driver*. Na plataforma PMDB, há a necessidade de manter as informações sobre as consultas executadas. Essas informações são utilizadas pelo Serviço de Predição, Serviço de Alocação e Serviço de Migração. No momento em que a aplicação submete uma consulta para o inquilino, o *QoSDBCDriver* intercepta a consulta e realiza algumas

ações. A primeira delas é marcar o tempo global de início a que a consulta foi submetida ainda na aplicação, instrução descrita na linha (2). Após isso, a consulta é enviada ao serviço de escalonamento por meio do *QoSDBCCoordinator*, como mostrado na linha (3).

Um objeto *SQL_VO* é retornado e contém informações sobre o resultado da execução local da consulta, ou seja, no próprio SGBD. Como exemplo, têm-se informações sobre o tempo de resposta local, número de linhas afetadas, consulta e seu tipo etc. O tempo de resposta global é calculado logo após o retorno da consulta para a aplicação, conforme ressaltado na linhas (4 e 5). O SLA do inquilino é obtido passando o identificador da conexão no Serviço de SLA, linha (6). Por fim, todas as informações pertinentes às interações de aplicação e plataforma são persistidas no *Log*, que pode ser visualizado na linha (7).

Algoritmo 3 - Função executada pelo Serviço de Alocação

```
 1: function execute_allocation(VMs[1...N], z, k, l)
 2:     allocate vector MD‘[1...N]
 3:     for j ← 1..N do
 4:         N_j ← qosdbc.monitoring_service().get_tenants_size(DBMS_j)
 5:         if (j ≠ k) then
 6:             sum ← 0
 7:             for i ← 1..N_j do
 8:                 sum ← sum + MT(i, j) − SLA(i, j)
 9:             end for
10:             MD‘[j] ← (sum + (MT(l, k) − SLA(l, k)))/(N_j + 1)
11:         else
12:             for i ← 1..N_j do
13:                 sum ← sum + (MT(i, j) − SLA(i, j))
14:             end for
15:             MD‘[j] ← sum/N_j
16:         end if
17:     end for
18:     x ← MAX(MD‘)
19:     if (x = k) then
20:         return N + 1
21:     end if
22:     return x
23: end function
```

O Algoritmo 3, por sua vez, demonstra o funcionamento do Serviço de Alocação sob a perspectiva de decisão se é necessário provisionar uma VM ou se o inquilino pode ser migrado para algum SGBD de alguma VM. Esse algoritmo implementa a técnica usada na seção 4.4.1. Supõe-se que os inquilinos estão enumerados como descrito nessa seção. Os argumentos, para o algoritmo, são: um vetor de VMs contidas no ambiente, o inquilino z a ser migrado, o índice k do SGBD em que z está e o índice l que identifica o

inquilino z dentro do $SGBD_k$. O retorno é um número inteiro que indica em qual SGBD o inquilino z deve ser migrado; caso contrário, é retornada a quantidade de SGBDs mais um, significando que uma nova VM deva ser provisionada e z deve ser migrado para esse novo recurso.

Vale notar que as funções $MT(i, j)$ e $SLA(i, j)$ correspondem, respectivamente, a MT_{ij} e SLA_{ij} definidos na seção 4.4.1. Esse algoritmo, conforme as linhas (3 e 17), calcula $MD`i$ para todo $1 <= i <= N$ e guarda esses valores no vetor $MD`$. Esse processo é feito iterando sobre o vetor $MD`[1...N]$ e persistindo os valores de MD'. Nota-se que, se a condição da linhas (5) for verdadeira, as linhas (5 a 10) calculam o valor de MD_j e guarda-se o resultado em $MD'[j]$; e, caso a condição seja falsa, computa-se o valor de $MD'j$ nas linhas (10 a 12) e o armazena-se em $MD'[j]$ que é a definição de $MD'j$ como é descrito na seção 4.4.1. Na linha (18), o valor máximo x de MD' é encontrado. Já na linha (19), é verificado se o $SGBD_x$ é o SGBD que contém z; caso isso seja verdadeiro, é retornado $N + 1$ para indicar que uma nova VM precisa ser provisionada, caso contrário, x é retornado.

O Algoritmo 4 ilustra o funcionamento do Serviço de Predição para vislumbrar uma possível violação de SLA dos inquilinos do ambiente. Primeiramente, é criada uma lista que representa todos os inquilinos localizados no ambiente, no dado momento em que houve a execução deste algoritmo. Essa lista pode ser visualizada na linha (2). Na linha (3), por meio do *Catálogo*, são recuperados todas as VMs ativas no ambiente; depois, essas VMs são representadas em um vetor. O laço da linha (4) ilustra o percurso de todas as VMs localizadas no vetor da linha anterior. Para cada VM, são resgatados todos os SGBDs ativos; esses são armazenados em um vetor, como demostra a linha (5).

Todos os SGBDs do vetor são percorridos no laço descrito na linha (6). Para cada SGBD, seus inquilinos são armazenados em vetor, como ilustrado na linha (7). O laço da linha (8) mostra a iteração sobre o vetor de inquilinos obtidos na linha anterior. Para cada inquilino são recuperadas informações sobre sua carga de trabalho, por meio do componente *Log*, como mostra a linha (9). Na linha (10) as informações sobre o inquilino, sua carga de trabalho e a execução da função *FEQ* sobre o inquilino são armazenadas na linha que representa todos os inquilinos do ambiente.

Ao chegar ao final do laço, ponto previsto na linha (13), a lista *tenants_workload* terá todos os inquilinos que foram encontrados em todos os SGBDs ativos de VMs ativas. Na linha (14), essa lista é percorrida e, para cada inquilino, é aplicado o Scale-Space, para

Algoritmo 4 - Função que executa para prever uma possível violação de SLA dos inquilinos

```
1: function execute_forecast_service
2:     List[Forecast] tenants_workload
3:     VMS[] ← catalog.get_all_VMs()
4:     for i ← 1..VMS.length do
5:         DBMSs[] ← catalog.get_DBMSs(VMS[i].id)
6:         for j ← 1..DBMSs.length do
7:             tenants[] ← catalog.get_tenants(DBMSs[j].id)
8:             for k ← 1..tenants.length do
9:                 workload ← log.get_workload(tenants[k])
10:                tenant_workload.add(Forecast(workload, tenants[k], FEQ(tenants[k])))
11:            end for
12:        end for
13:    end for
14:    for i ← 1..tenants_workload.size() do
15:        Forcastf ← tenants_workload.get(i)
16:        f.set_smoothing_signal(kernel.execute_ss(f.get_workload()))
17:    end for
18:    for i ← 1..tenants_workload.size() do
19:        Forcastf ← tenants_workload.get(i)
20:        kernel.execute_training_ARIMA(f.get_smoothing_signal(), f.get_tenant_id())
21:        kernel.execute_training_SVR(f.get_smoothing_signal(), f.get_tenant_id())
22:    end for
23:    for i ← 1..tenants_workload.size() do
24:        Forcastf ← tenants_workload.get(i)
25:        ARIMA_result ← kernel.get_forecast_ARIMA(f.get_tenant_id())
26:        SVR_result ← kernel.get_forecast_SVR(f.get_tenant_id())
27:        f.set_ARIMA_result(ARIMA_result)
28:        f.set_SVR_result(SVR_result)
29:    end for
30:    return tenants_workload
31: end function
```

suavizar e eliminar pontos da carga de trabalho do inquilino, simplificando o processo de predição. A ação de suavização é mostrada nas linhas (15 e 16). Já o laço representado na linha (18) mostra a aplicação do treinamento dos dados suavizados de cada inquilino, como mostram as linhas (20 e 21). Observe-se que são realizados os treinamentos dos dois métodos previstos neste trabalho.

O último laço, representado pela linha (23), recupera cada inquilino e suas informações obtidas nos laços anteriores. Após isso, são executados os dois métodos de predição, como ilustram as linhas (25 e 26). Ao final deste laço, é retornada a lista com informações dos inquilinos envolvidos no processo de predição, linha (30). A lista retornada armazena informações do inquilino, seu tamanho, sua estimativa de tempo de migração, e o instante de tempo em que o SLA poderá ser violado, obtido por cada

método.

Algoritmo 5 - Função executada pelo *QoSDBCCoordinator* para realizar a migração de um inquilino

```
 1: function migrate_tenant(tenant_id, o, d)
 2:     DBMS_d ← catalog.get_DBMS(d)
 3:     DBMS_o ← catalog.get_DBMS(o)
 4:     while exist active transactions in tenant_id do
 5:         DBMS_o.execute(transactions)
 6:         log.put(transactions, SGBD_o)
 7:     end while
 8:     DBMS_o.backup(tenant_id)
 9:     while processing backup of tenant_id do
10:         DBMS_o.execute(transactions)
11:         log.put_updates(transactions, SGBD_o)
12:     end while
13:     distributed_file_system.transfer(DBMS_d, tenant_id)
14:     while transferring the tenant_id do
15:         DBMS_o.execute(transactions)
16:         log.put_updates(transactions, SGBD_o)
17:     end while
18:     DBMS_d.restore(tenant_id)
19:     while restoring and exist active transactions in tenant_id do
20:         DBMS_o.execute(transactions)
21:         log.put_updates(transactions, DBMS_o)
22:     end while
23:     DBMS_o.lock(tenant_id, qosdbc_driver)
24:     for t ← 1..catalog.get_updates(DBMS_o) do
25:         DBMS_d.execute(t)
26:     end for
27:     qosdbc_driver.redirect_connections(DBMS_o, DBMS_d)
28:     qosdbc_driver.restart_tenant(DBMS_d)
29:     DBMS_o.drop_database(tenant_id)
30: end function
```

Por fim, o Algoritmo 5 ilustra o funcionamento do *Live Migration* utilizado pelo PMDB. A função *migrate_tenant* é executada pelo *QoSDBCCordinator* no momento em que ele visualiza que o inquilino entrou ou entrará em conflito. O Serviço de Alocação, por sua vez, determina qual o destino do inquilino. Esse novo destino é representado pelo argumento d da função. O inquilino é o *tenant_id* e sua origem é o. As linhas (2 e 3) recuperam, respectivamente, objetos que representam os SGBDs de destino e de origem. O processo de migração só inicia quando todas as transações ativas do inquilino terminam; isso é representado pelo laço exposto na linha (4). Como a abordagem é *Live Migration*, até que as transações se efetivem, essas são executadas no SGBD de origem, conforme linhas (5 e 6).

Quando já não há transações ativas, o *backup* do inquilino *tenant_id* é realizado;

isso é mostrado na linha (8). Até que o *backup* finalize, o SGBD de origem continua a servir a aplicação, conforme laço das linhas (9, 10, 11 e 12). Após a finalização do processo de *backup*, esse dado é transferido para o novo SGBD por meio do Sistema de Arquivo Distribuído, instrução mostrada na linha (13). Observa-se que a instrução da linha (11) já armazena, no *Log*, as atualizações que necessitam ser movidas para a nova localidade do inquilino, a fim de garantir a consistência. A transferência do inquilino é relativamente rápida, pois o Sistema de Arquivo Distribuído é montado em todas as VMs, tendo a abstração que o *backup* é um arquivo de diretório compartilhado. Mesmo assim, o algoritmo mostra que, enquanto o arquivo de *backup* está sendo transferido, todas as transações são persistidas na origem e armazenadas no *Log*, como mostram as linhas (14, 15, 16 e 17).

Assim que o *backup* do inquilino *tenant_id* está completamente disponível na VM de destino, o SGBD de destino executa o procedimento de restauração do *backup*, processo exibido na linha (18). Até que o processo de restauração se efetive, o SGBD de origem continua a servir à aplicação e todas as transações de atualizações são armazenadas no *Log*, como mostra o laço descrito nas linhas (19, 20, 21 e 22). Ao terminar o laço, começa o período de indisponibilidade da estratégia de *Live Migration*, para se manter a coerência da consistência. A linha (23) demonstra que há um bloqueio sobre a cópia de origem do inquilino *tenant_id*. A partir deste momento, todas as chamadas que a aplicação realiza são guardadas em uma fila. O laço representado nas linhas (24, 25 e 26) realiza a atualização da cópia de destino, executando as transações de atualização executadas na origem e não estão no destino. Após a atualização da cópia de destino, redirecionam-se todas as conexões da aplicação para o SGBD de destino, instrução mostrada na linha (27). A linha (28) sinaliza ao *driver* que já pode descarregar a fila. Por fim, a linha (29) exibe a instrução que remove a cópia de origem do inquilino migrado.

5.4 CONCLUSÃO

Este capítulo discute a arquitetura adotada para a implementação do PMDB, destacando as funcionalidades de cada componente arquitetural. Um fluxo foi ilustrado e discutido para demonstrar o funcionamento geral do PMDB. Também foram destacados os principais algoritmos desenvolvidos, referentes a alocação, predição e migração. O próximo capítulo trata de questões referentes às propostas experimentais para demonstrar a eficácia do PMDB.

CAPÍTULO 6

AVALIAÇÃO EXPERIMENTAL

Neste capítulo, descrevem-se todos os experimentos realizados para validar o trabalho. São apresentadas as diferenças no processo de avaliação de serviços de dados em nuvem e, ao final, são demonstrados o ambiente e os experimentos conduzidos.

6.1 AVALIAÇÃO

A avaliação de SGBDs em nuvem exprime diferenças significativas em relação à avaliação dos SGBDs tradicionais. Os SGBDs tradicionais pressupõem a existência de configurações fixas de recursos, tratam da otimização de desempenho e têm como objetivo minimizar o tempo de resposta para cada requisição [Moreira et al., 2012]. Essa visão não considera os custos operacionais do sistema; entretanto, o modelo de pagamento baseado no uso da computação em nuvem requer que os custos operacionais sejam considerados juntamente com o desempenho. No ambiente em nuvem, o objetivo é minimizar a quantidade de recursos necessários para garantir uma meta de tempo de resposta para cada requisição e, com isso, o custo é um parâmetro importante [Florescu and Kossmann, 2009].

Existem muitos *benchmarks* para SGBDs relacionais executados em infraestruturas tradicionais, como os *benckmarks* TPC [Tözün et al., 2013], que são amplamente utilizados para avaliar implementações de SGBDs diferentes. SGBDs em nuvem apresentam características que não fazem parte dos parâmetros avaliados pelos *benchmarks* TPC. Estes sistemas são dinâmicos e, sendo assim, têm a capacidade de melhorar sua configuração ao longo do tempo. Além disso, os *benchmarks* TPC consideram que o sistema a ser avaliado é baseado em transações com as propriedades ACID, não utilizadas em muitos sistemas em nuvem. Estes pontos indicam que os atuais *benchmarks*-padrão para os SGBDs devem ser repensados [Moreira et al., 2014].

O problema de padronização de *benchmarks* para SGBDs em nuvem é um desafio. Isso decorre principalmente da decorrência da diversidade de SGBDs em nuvem e da forma como estes sistemas são construídos, visto que as implementações variam em termos de

modelo de dados, níveis de consistência, expressividade da linguagem de acesso aos dados, entre outros [Sousa, 2013]. Em razão da elasticidade e do tamanho destes sistemas, que podem possuir centenas ou milhares de máquinas, alguns problemas podem não ocorrer até que um determinado tamanho seja alcançado. Com isso, o desenvolvimento de um *benchmark* que aborde todos os possíveis casos de uso do sistema é muito difícil e inviável [Moreira et al., 2012].

[Florescu and Kossmann, 2009] destacam que sistemas de *benchmarks* para o gerenciamento de dados devem tratar de aspectos de custo, tempo de resposta, vazão, escalabilidade, consistência, flexibilidade, e discutem como estes aspectos podem impactar na arquitetura dos SGBDs. [Binnig et al., 2009] discutem o motivo pelo qual os *benchmarks* tradicionais não são suficientes para analisar os novos serviços em nuvem e apresentam ideias para o desenvolvimento de um novo *benchmark*. Em [Cooper et al., 2010], é mostrado um *framework* com o objetivo de facilitar a avaliação de variados SGBDs chave/valor em nuvem. Este *framework* trata de características de desempenho e escalabilidade. [Huppler, 2010] discute que os *benchmarks* TPC não refletem os aspectos atuais de custo da indústria e propõe sugestões sobre custo, desempenho e disponibilidade que devem ser contemplados pelos novos *benchmarks*. Em [Sethuraman and Taheri, 2011], é proposto o TPC-V, um *benchmark* para avaliar o desempenho de SGBDs em ambientes virtualizados. O trabalho descreve as características deste *benchmark* e o estado de seu desenvolvimento.

O OLTPBenchmark [Difallah et al., 2013] é um *framework* para avaliar o desempenho de distintos SGBDRs, ante configurações de cargas de trabalho OLTP. O *framework* possui diversos *benchmarks* que representam várias aplicações, como TPC-C, Twitter, YCSB, AuctionMark e Wikipedia. O OLTPBenchmark possibilita configurar a taxa de transações, definir o percentual de cada tipo de transação por tempo de experimento, obter informações sobre vazão, média do tempo de resposta e informações sobre utilização dos recursos do sistema gerenciador de banco de dados e do sistema operacional. Este *framework* utiliza o padrão JDBC para a conexão com o SGBDR. Neste sentido, o OLTPBenchmark contempla os principais requisitos para simular aplicações de bancos de dados, desde sua variedade de esquemas, métricas de avaliação e funcionalidades [Moreira et al., 2012]; no entanto, o OLTPBenchmark foi projetado para ambientes centralizados. Para conduzir os experimentos propostos para avaliar o PMDB, teve-se que adaptá-lo para ambientes de multi-inquilinos e distribuídos [Moreira et al., 2014].

6.1.1 Ambiente

A infraestrutura da Amazon foi utilizada neste trabalho. Esta infraestrutura oferece VMs nas quais se podem instalar e executar os SGBDs. Assim, uma imagem de VM com um SGBD pré instalado e pré configurado está disponível para a implantação de forma simples e rápida [Aboulnaga et al., 2009]. As VMs são diferenciadas pelos recursos que elas oferecem, tais como CPU, memória e espaço em disco.

Na condução da avaliação do PMDB, implementa-se uma nova estratégia de gerenciamento com diferentes bancos de dados. Assim, é favorecido um completo ambiente multi-inquilino [Moreira et al., 2014]. Esses bancos de dados são fornecidos pelo *framework* OLTPBenchmark [Difallah et al., 2013]. Em ambiente experimental, usado pelo PMDB, implanta-se um protótipo do PMDB e realizam-se experimentos em um *cluster* de máquinas, utilizando o Amazon EC2 em uma mesma zona de disponibilidade (*us-west-1b*). As VMs possuem sistema operacional Ubuntu 12.04 LTS, o MySQL 5.5 com *InnoDB* e 128 MB de *buffer*. O OLTPBenchmark é utilizado para gerar várias cargas de trabalho para os bancos de dados, ou seja, os inquilinos dos experimentos.

As IaaS fornecem serviços de armazenamento e transferência de dados, por exemplo, o *Amazon EBS*, normalmente com custos adicionais referentes à quantidade de dados armazenados e de requisições de E/S. Estes custos dos serviços de armazenamento são relativamente menores do que os custos associados às VMs, e alguns provedores não cobram por estes serviços. PMDB utiliza os recursos de forma eficiente por meio do compartilhamento de recursos, o que auxilia na redução dos custos operacionais e reflete nos custos monetários. Neste trabalho, contudo, não são considerados os custos monetários, visto que cada provedor utiliza uma estratégia diferente de tarifação (apenas os valores relativos às máquinas virtuais são contabilizados).

Os SGBDs persistem os dados, que consistem de *logs* e *backups*, em um sistema de armazenamento distribuído ou em um *Network-Attached Storage* (NAS) [Das et al., 2011]. Por exemplo, um NAS oferece um armazenamento escalável, altamente disponível e tolerante a falhas para persistirem os dados dos clientes. Esta arquitetura é diferente de sistemas de disco compartilhado, que utiliza um controle entre operações concorrentes [Bernstein and Newcomer, 2009]. Esta dissociação da propriedade de armazenamento auxilia na migração dos dados entre os SGBDs e é utilizada por sistemas como o *Amazon EBS*. Para estes experimentos, utilizou-se uma versão simplicada do modelo de qualidade proposto neste trabalho. Esta versão considerou a métrica de tempo de resposta.

6.1.2 Experimentos

O ambiente de computação em nuvem expressa muitos desafios. Isso não é diferente no que se refere a realizar experimentos em larga escala. Elaborar e estabelecer experimentos que representam aplicações reais torna-se complexo, bem como comparar diferentes abordagens para o gerenciamento de dados em nuvem [Sousa, 2013]. A comparação direta com os trabalhos relacionados é muito complexa de ser realizada, visto que cada trabalho reúne um conjunto de pressupostos próprios. Dessa forma, não serão realizados experimentos diretos comparando o PMDB e outras abordagens.

Com o objetivo de simular um ambiente real, serão definidas aplicações com vários requisitos: SLA, tamanho do banco de dados, esquemas de banco de dados, número de conexões, taxa e outras consultas. Estas aplicações são baseadas nas existentes no *framework* [Difallah et al., 2013]. A avaliação do PMDB busca analisar as diversas características contempladas. Cada experimento explorará um aspecto diferente de um SGBD multi-inquilino em nuvem, em particular, desempenho, utilização de recursos e qualidade do serviço. Estes experimentos serão baseados nos experimentos realizados nos trabalhos propostos por [Elmore et al., 2011b], [Schiller et al., 2013], [Sousa and Machado, 2012] e [Barker et al., 2012].

Os experimentos objetivam uma avaliação detalhada das características do PMDB, verificando o desempenho da técnica de migração, alocação e a eficácia do modelo de qualidade utilizado. Além disso, verifica-se, por meio de experimentos, a viabilidade de combinar a técnica de migração do PMDB como uma solução preditiva, visando reduzir a violação de SLA dos inquilinos. Assim, dividem-se os experimentos em quatro categorias: aspectos de migração, alocação, predição e todos os aspectos em conjunto.

6.1.3 Resultados e Discussões

Técnica de Migração

Aqui são realizados dois experimentos. O primeiro é conduzido no intuito de investigar a velocidade e minimização da indisponibilidade da técnica de migração implementada no PMDB. Neste caso, aumenta-se o tamanho dos inquilinos e observam-se seus respectivos tempos de migração. Na técnica de *Live Migration*, há dois tempos observados e discutidos, o de transferência total do inquilino e o de indisponibilidade. Já o segundo

é realizado com o objetivo de analisar o tempo de violações de SLA na situação, multi-inquilinos, onde há um crescimento da carga de trabalho de um dos inquilinos. Com o aumento da carga de trabalho de um inquilino que compartilha o mesmo recurso com os demais, há uma grande possibilidade de este inquilino diminuir seu desempenho ou dos demais. Neste sentido, mostra-se uma situação em que a migração se faz necessária para realocar o inquilino em uma VM onde ele possa trabalhar mantendo sua qualidade.

O primeiro experimento mostra a velocidade da técnica de migração com a variação do tamanho dos inquilinos. O objetivo é apontar o desempenho e a sobrecarga gerada pela técnica de migração combinada com a estratégia de monitoramento e os outros componentes do sistema PMDB. A Tabela 6.1 exibe os inquilinos usados e seus respectivos tamanhos. Neste experimento, foi utilizada, em cada migração, uma VM do tipo *small* do EC2. Esse tipo de VM possui um processador Xeon 1.7 GHz, 1.7 GB de memória principal e 160 GB de memória secundária.

Inquilino	**Tamanho (MB)**
TPC-C_1	100
TPC-C_2	300
TPC-C_3	500
TPC-C_4	700
TPC-C_5	900

Tabela 6.1 Inquilinos Utilizados e seus Respectivos Tamanhos

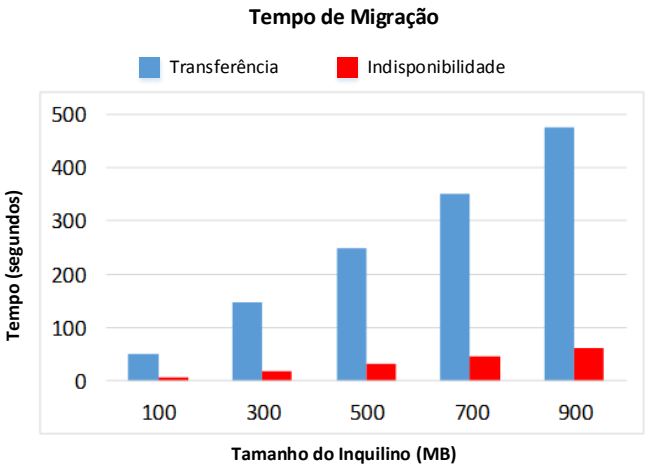

Figura 6.1 Tempo de Migração por Tamanho de Inquilino

A Figura 6.1 mostra os resultados do experimento que mede a velocidade do tempo de migração de inquilinos de variados tamanhos. Como observado, o *Live Migration* implementado no PMDB pode minimizar o tempo de indisponibilidade em face do crescimento dos inquilinos. Neste caso, comparado à estratégia *Stop-and-Copy*, o PMDB reduz o tempo de indisponibilidade por meio do *Live Migration* combinado com os mecanismos de transferência rápida de dados e tolerantes a falhas das infraestruturas em nuvem. Quanto maior for o inquilino, maior será o tempo de transferência e de indisponibilidade. O tempo de transferência engloba as etapas que vão da cópia do inquilino até sua restauração no novo destino. Já o tempo de indisponibilidade é necessário na etapa de atualização da nova cópia e redireciomento das conexões, mantidas pelas aplicações, para o novo destino do inquilino. É válido ressaltar que experimentos como este foram necessários para implementar a função FEQ do PMDB, dado que o tempo de migração cresce, linearmente, ao tamanho do inquilino.

No segundo experimento e com o intuito de realizar uma execução mais próxima de um ambiente real, foram considerados apenas os valores após os primeiros 120 segundos das medidas obtidas. Com isso, evita-se que os atrasos na "inicialização" dos inquilinos impactem nos resultados. Os experimentos foram realizados variando a taxa das transações ao longo de um período. Aqui, as taxas dos inquilinos YCSB e do Wikipédia foram fixadas em 60. Já a taxa do TPC-C foi modificada de acordo com a sequência (50, 100, 150, 200, 250), a cada 300 segundos. Uma VM foi utilizada para hospedar os três inquilinos em um mesmo SGBD e um outra VM usada para executar o OLTPBenchmark e gerar as cargas de trabalho para os inquilinos. Cada VM é do tipo *small* do EC2. A Tabela 6.2 exibe, para cada inquilino, o seu tamanho e o SLA baseado em tempo de resposta.

Inquilino	**Tamanho (MB)**	**SLA (milisegundos)**
TPC-C	400	150
YCSB	800	50
Wikipédia	600	50

Tabela 6.2 Inquilinos e os seus Respectivos Tamanhos e SLAs

A Figura 6.2 mostra os resultados deste primeiro experimento. Em [Moreira et al., 2012], mostra-se que, entre os três inquilinos, o TPC-C é o que causa maior interferência. Portanto, aumentou-se a taxa do TPC-C no intuito de forçar a violação de SLA e demonstrar o comportamento do PMDB. Neste experimento, o inquilino TPC-C, após sua primeira

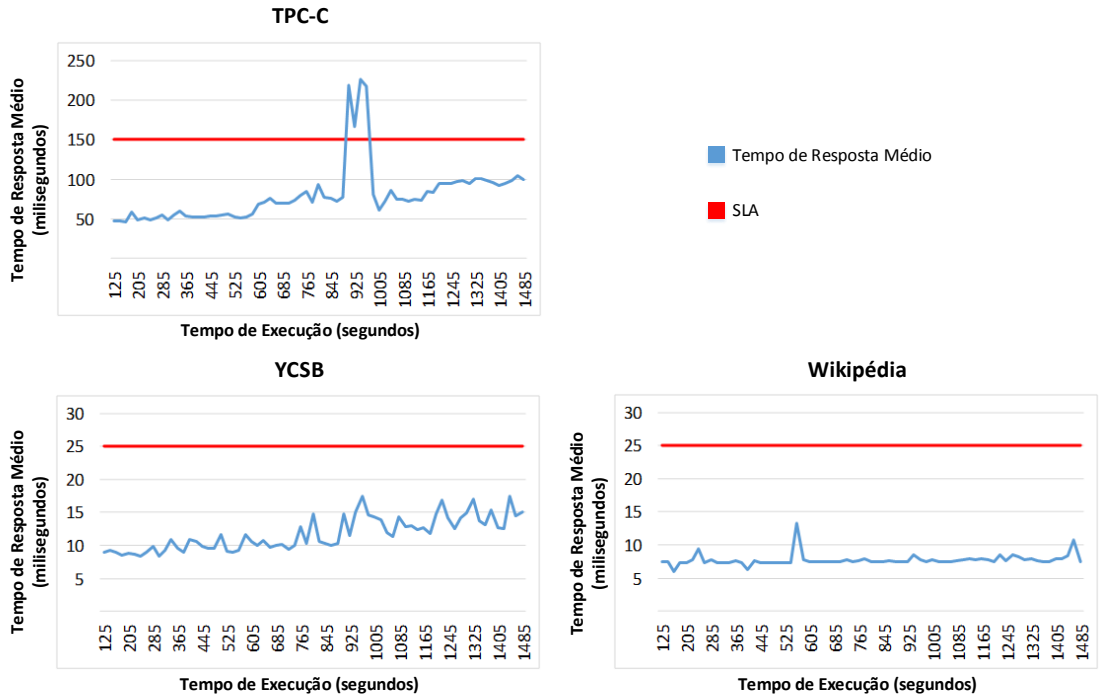

Figura 6.2 Tempo de Resposta Médio dos Inquilinos

violação de SLA, foi migrado para uma nova VM. Foi necessário provisionar uma nova VM, pois não existia nenhuma outra no ambiente. Como mostra o experimento, pode-se perceber que o PMDB reduz o número ou tempo de violação de SLA para o inquilino TPC-C, migrando-o para uma outra VM. Neste caso, o ideal seria uma tomada de decisão antecipada, no processo de migração, para que não ocorram violações de SLA. Ademais, pode-se observar que houve aumento no tempo de resposta médio das transações, pois a sequência da taxa de transações é crescente.

Técnica de Alocação

Neste experimento, mostra-se eficácia da técnica de alocação usada pelo PMDB, comparando-a com uma técnica simples de alocação. O critério utilizado pela técnica simples de alocação é migrar o inquilino para uma VM que possui o maior número de memória principal disponível. Aqui, se intenta mostrar que a técnica de alocação utilizada pelo PMDB realiza menos provisionamento de novas VM. Com isso, demostra-se que o PMDB tende a uma utilização mais eficaz dos recursos. Para forçar o sistema a realocar os inquilinos, foram variadas suas respectivas cargas de trabalho, forçando a diminuição da qualidade para um deles.

Para este experimento, criaram-se quatro VMs. Duas VMs do tipo *small* e duas do tipo *micro*. VMs do tipo *micro* possuem um processador Xeon 1.0 GHz, 0.615 GB de memória secundária e 30 GB de capacidade de disco. Uma VM do tipo *small* foi usada para "inicializar" o OLTPBenchmark, gerando as cargas de trabalho para todos os inquilinos envolvidos no ambiente. As duas VMs do tipo *micro* e uma VM do tipo *small* foram utilizadas para hospedar os inquilinos. Como consequência dos resultados obtidos na técnica de migração, optou-se por utilizar inquilinos TPC-C, pois são os que geram maior interferência [Moreira et al., 2012]. A Figura 6.3 exibe o os inquilinos e suas VMs.

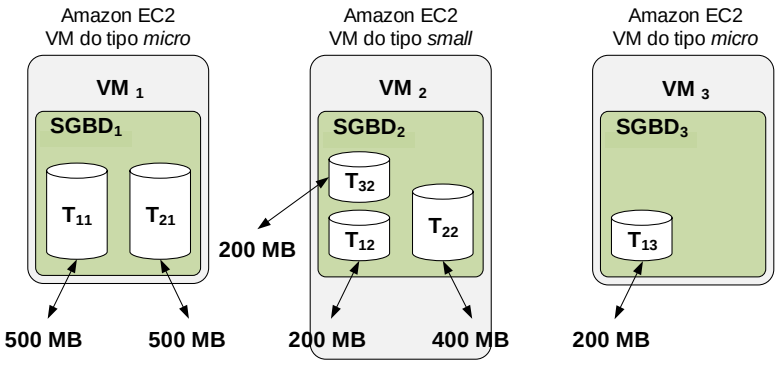

Figura 6.3 Alocação Inicial dos Inquilinos nas VMs

A taxa do inquilino T_{11} variou de acordo com a sequência (1000, 1000, 3000, 3000, 2000, 1000), a cada 300 segundos. Já a taxa do T_{22} foi modificada de acordo com a sequência (1000, 1000, 3000, 3000, 2500, 2500), a cada 300 segundos. Também, a taxa do T_{13} foi alterada de acordo com a sequência (500, 500, 500, 500, 450, 450), a cada 300 segundos. Os demais inquilinos mantiveram suas taxas fixas em 1000. O inquilino T_{11} teve sua SLA atribuída para 1500 milissegundos. A Figura 6.4 (a) ilustra o resultado com a alocação simples. Já a Figura 6.4 (b) exibe o resultado na técnica de alocação implementada no PMDB.

Na Figura 6.4 (a), pode-se observar que o número de violações de SLA foi superior ao ilustrado na Figura 6.4 (b). A estratégia de alocação simples leva em consideração a VM que possui o maior número de memória principal disponível. Por outro lado, a estratégia de alocação adotada no PMDB considera a carga de trabalho de cada SGBD do ambiente e os SLAs dos inquilinos. Neste sentido, o PMDB vai migrar o inquilino que violou o SLA para uma VM onde este possa executar sua carga de trabalho com menor interferência, diminuindo sua violação de SLA.

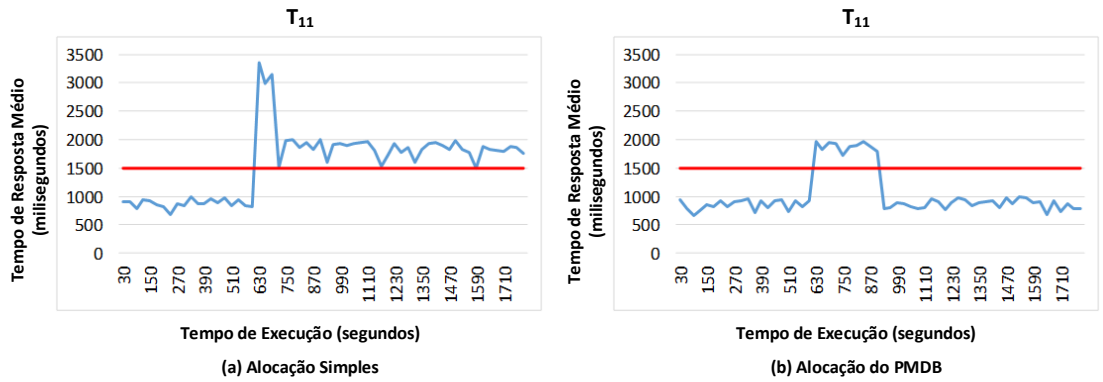

Figura 6.4 Resultados do Experimento de Alocação

Ao atingir, aproximadamente, os 630 segundos de experimentos, ocorre a violação do SLA para o inquilino T_{11}. A alocação simples escolhe a VM_2, que possui o maior número de memória principal livre; no entanto, a VM_2 é a que possui um número maior de inquilinos. Com isso, o $SGBD_2$, que está na VM_2, terá que fazer a gestão de quatro inquilinos, o que leva a uma maior interferência e sobrecarga do $SGBD_2$, elevando o tempo de resposta destes inquilinos. Já a alocação implementada no PMDB escolhe a VM_3 que é uma de capacidade inferior a VM_2; no entanto, a VM_3 possui um número inferior de inquilinos e a diferença média do tempo de resposta do inquilino T_{13} com o seu SLA é bem maior. Com isso, a VM_3 será a que está menos sobrecarregada, levando em consideração a carga do inquilino T_{13} e o seu SLA. Pela Figura 6.4 (b), pode-se perceber que o inquilino T_{11} se comportou de maneira satisfatória em conjunto com o T_{13} na VM_3.

Técnica de Predição

Aqui se realizam dois experimentos. O primeiro intenta observar que o uso do Scale-Space reduz a quantidade de pontos, sem perder a expressividade do sinal resultante. Para isso, pegam-se os dados de monitoramento da carga de trabalho, por intervalo de tempo, e geram-se o sinal original e o sinal suavizado. Já o segundo tem como objetivo mostrar que existe a possibilidade de obter, por meio dos métodos de predição, uma janela de observação das futuras violações de SLA, obviamente, em tempo hábil para a tomada de decisão. Com isso, serve-se como critério para antecipar a migração dos inquilinos, evitando um número maior de violações de SLA e decréscimo da qualidade. Com base nos dois sinais, utilizam-se as métricas de erro, mostradas no capítulo 2, para verificar a eficiência da técnica espaço de escala, para cada método. Além disso, verifica-se, por meio

das métricas de erro, qual dos métodos expressou melhor resultado para as predições.

O experimento de predição tem como objetivo principal exibir a possibilidade de obter uma janela de previsão adequada para o início da reorganização dos inquilinos, ou seja, iniciar os processos de alocação e migração. Primeiro, é importante estudar a escala a ser utilizada para representar a série temporal. De acordo com o experimento ilustrado na Figura 6.1, pode-se observar que uma migração de um inquilino do tamanho 900 MB demanda no máximo 500 segundos. Portanto, parece ser razoável observar os dados no nível de detalhe que corresponde a minutos.

Com base nesta observação, aplica-se, primeiramente, uma transformação da série em segundos para uma série temporal em termos de minutos. O sinal obtido é então suavizado com o espaço de escala com o parâmetro $s = 3$. Isto porque, em experiências realizadas, em que se varia o valor de s, este valor foi o que produziu maior eficiência para efeitos de previsão [Santos et al., 2013]. Com isso, é alcançado o objetivo de se concentrar nas características apresentadas nesta escala específica, o que impede a atenção em informações irrelevantes. Em outras palavras, todos os picos e depressões que duram menos de três minutos são considerados como irrelevantes, evitando assim uma migração desnecessária em razão de picos transitórios. Neste experimento, foi utilizado o inquilino TPC-C em uma VM do tipo *micro*. A taxa do TPC-C variou de acordo com a sequência (10000, 200000, 900000, 200000, 10000, 10000, 200000, 900000, 200000, 10000, 10000, 200000, 900000, 200000), a cada 200 segundos. A Figura 6.5 mostra a aplicação do método na carga de trabalho obtida.

Percebe-se, pela Figura 6.5, que a aplicação do espaço de escala não perde a significância da série temporal original. O espaço de escala conseguiu remover os picos insignificantes, ou seja, os que levam menos de 3 minutos. Esse resultado era esperado, pois isso é uma garantia do espaço de escala, que foi provado pelo *Scaling Theorem* [Santos, 2013] [Santos et al., 2013]. Com isso, melhoram-se as etapas de treinamento dos métodos de predição, pois retira-se um percentual de informações que podem representar valores atípicos e que estes sejam incorporados à série temporal.

Seguindo-se com o experimento, aplicam-se os métodos ARIMA e SVR sobre o sinal já suavizado com o espaço de escala, objetivando avaliar a eficiência dos métodos para uma janela de predição com 10 minutos sobre uma janela de observação com 30 minutos. A Figura 6.6 ilustra os resultados.

O SVR mostra resultados inferiores em comparação ao ARIMA. Isto decorre do

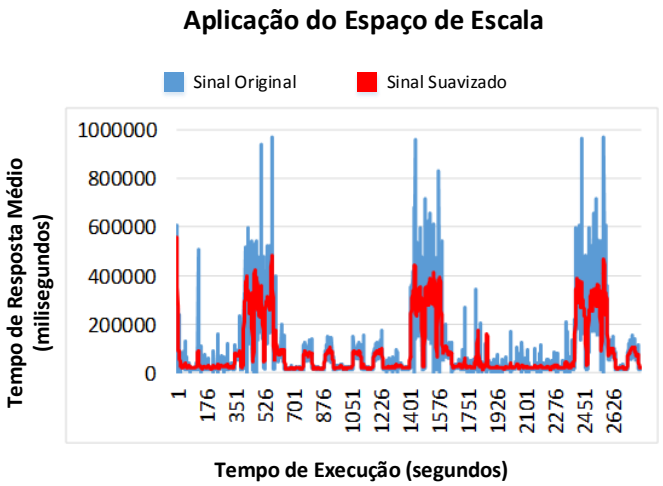

Figura 6.5 Resultados da Aplicação do Espaço de Escala

fato de que o SVR necessita de um esforço na etapa de parametrização. Já o ARIMA é mais simples de parametrizar; basta informar a série temporal e a janela de predição. O ARIMA é ajustado, automaticamente, a cada janela de observação, adaptando-o para cada cenário de predição [Moreira et al., 2014] [Santos et al., 2013]. Em ambos os métodos, utilizou-se a parametrização-padrão. O tamanho das janelas de observação e de previsão foi obtido empiricamente com o objetivo de maximizar o horizonte de previsão com o menor número de observações possíveis. A Tabela 6.3 exibe a medida do erro, utilizando as métricas de erro, para ambos os métodos de predição utilizados no experimento da Figura 6.6.

Método	RMSE	MAPE	PRED(25)
ARIMA	582.760	0.018	1.0
SVR	3120.076	0.105	1.0

Tabela 6.3 Análise de Erros dos Métodos ARIMA e SVR

Conforme a Tabela 6.3, pode-se comprovar que o ARIMA obteve os melhores resultados, pois registrou os menores valores de erro para cada métrica escolhida. Ambos os métodos obtiveram os mesmos valores de erro no $PRED(25)$; isso porque o erro contido nas amostras nao é tão significativo para capturar alterações. Neste caso, é necessária uma amostra de série de maior tamanho; contudo, as métricas $RMSE$ e $MAPE$ confirmam o superioridade do ARIMA neste experimento.

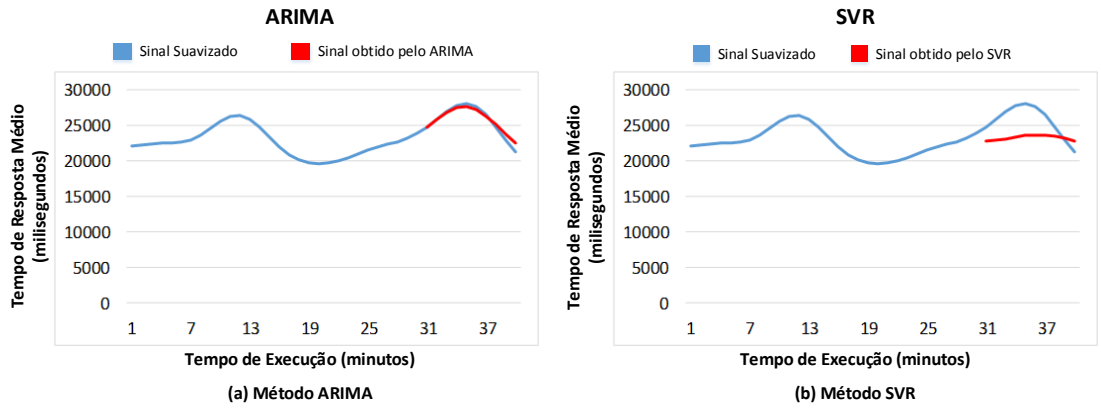

Figura 6.6 Resultados de Predição com o ARIMA e o SVR

Em razão destes resultados, o PMDB foi capaz de obter uma janela de predição com o tamanho de 10 minutos sobre uma janela de observação com 30 minutos. Como esperado, há situações em que o ARIMA mostra bom desempenho, mas também situações em que o método não é capaz de obter resultados precisos de previsão. A Figura 6.6 apresenta um bom resultado, que é útil para o PMDB.

Todas as Técnicas em Conjunto

Este experimento tem como objetivo mostrar o ganho alcançado quando PMDB utiliza as três técnicas em conjunto. Para conduzir a experimentação, compara-se o PMDB com uma técnica simples de provisionamento reativo. Esta técnica faz o provisionamento de uma nova VM quando um inquilino viola SLA, migrando este inquilino para essa nova VM. A técnica simples de provisionamento reativo foi implementada no PMDB, desabilitando a técnica de predição e alocação, forçando a migração para uma nova VM quando o SLA do inquilino for violado.

Aqui criaram-se três VMs do tipo *small*. A quarta VM do tipo *small* foi utilizada para executar o OLTPBenchmark para gerar as cargas de trabalho dos inquilinos. Os inquilinos escolhidos foram o YCSB e o Wikipédia, pois, em outros experimentos, foi constatado que estes são os inquilinos que, ao aumentar ou diminuir a taxa de transações, a carga de trabalho segue o mesmo comportamento [Moreira et al., 2012]. A Tabela 6.4 ilustra a organização do ambiente, dos inquilinos, dos tamanhos dos inquilinos e da alocação inicial proposta para este experimento.

VM	Inquilinos
VM_1	$YCSB_1$ (300 MB), $Wikipédia_1$ (500 MB)
VM_2	$YCSB_2$ (400 MB)
VM_3	$Wikipédia_2$ (100 MB)

Tabela 6.4 Alocação Inicial dos Inquilinos para o Experimento das Técnicas em Conjunto

Para cada inquilino, seguem o SLA aplicado e a sequência das taxas aplicadas, que variam cada 20 minutos. Melhor visualização destas informações pode ser obtida na Tabela 6.5.

Inquilino	SLA (segundos)	Sequência da Taxa
$YCSB_1$	1	(1000, 3000, 5000, 1000)
$YCSB_2$	0,5	(1000, 1000, 1000, 1000)
$Wikipédia_1$	1	(1000, 1000, 1000, 1000)
$Wikipédia_2$	0,5	(1000, 1000, 1000, 3000)

Tabela 6.5 SLA e Taxas Utilizadas no Experimento das Técnicas em Conjunto

A Figura 6.7 mostra o resultado deste experimento, que compara o PMDB com uma solução de provisionamento reativo.

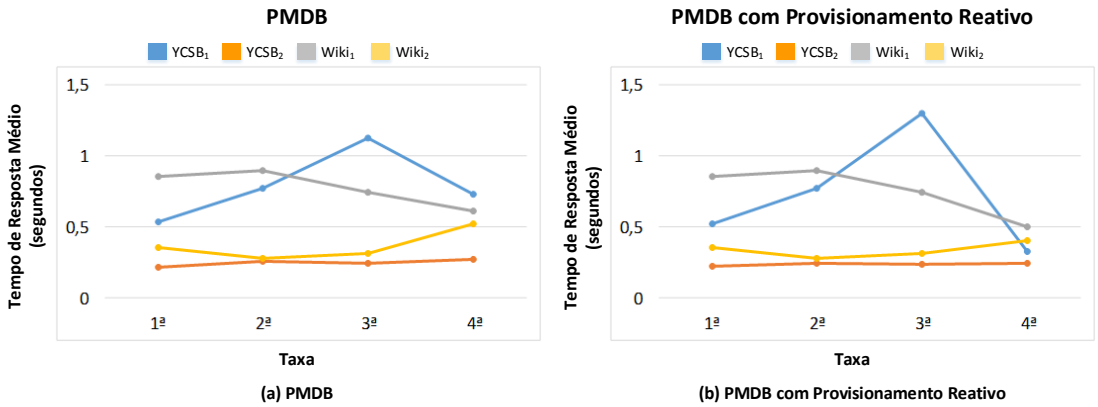

Figura 6.7 Tempo de Resposta Médio com as Técnicas em Conjunto

Observa-se pela Figura 6.7 que há uma primeira violação de SLA, quando o tempo de experimento alcança os 40 minutos. Neste instante, a taxa de transações do inquilino $YCSB_1$ muda de 3000 para 5000. Como se pode perceber, a Figura 6.7 (a) mostra que o

tempo de resposta médio do PMDB foi de 1,1 segundo. Já pela Figura 6.7 (b), o PMDB com provisionamento reativo foi de 1,3 segundo. Com isso, mostra-se que o PMDB com provisionamento reativo tem qualidade inferior ao PMDB, pois violou mais tempo de SLA. Quando o PMDB atinge os 38,5 minutos de experimento, o serviço de predição sinaliza a violação de SLA. Neste instante, é iniciada a migração do inquilino da VM_1 para a VM_3, com decisão do serviço de alocação. O tempo de migração total foi de 217 segundos.

Já o PMDB com provisionamento reativo sinalizou o serviço de migração no instante 40,5 minutos do experimento. O tempo de migração para uma nova VM foi de 254 segundos. Neste sentido foi necessário o provisionamento da VM_3 para hospedar o inquilino $YCSB_1$. Após a migração, pode-se perceber que o tempo de resposta médio do inquilino $YCSB_1$ diminui no PMDB provisionamento reativo, pois este inquilino está isolado em uma VM. Já no PMDB, o inquilino $YCSB_1$ conseguiu manter a sua qualidade, compartilhando o mesmo SGBD que o $Wikipédia_2$. Observa-se também, pela Figura 6.7 (b), que o tempo médio de resposta do inquilino $Wikipédia_1$ diminui, pois, no PMDB com provisionamento reativo, este inquilino está isolado na VM_1.

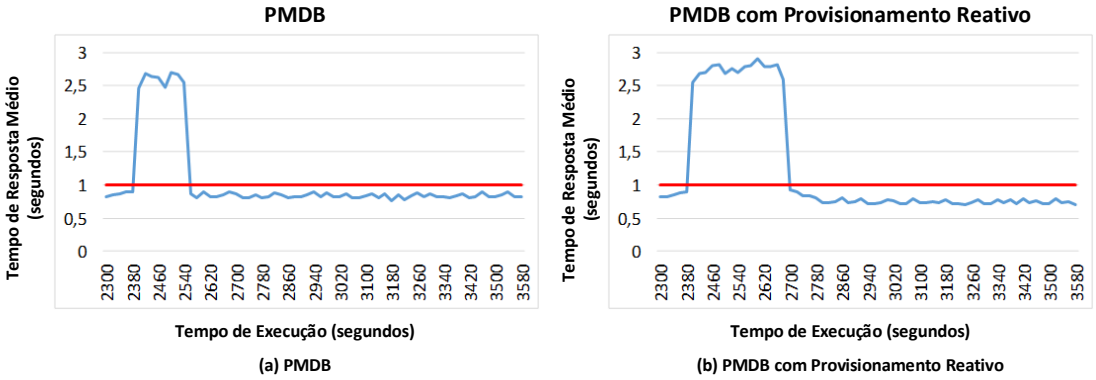

Figura 6.8 Violação de SLA ocorrida entre 38,5 e 60 Minutos do Experimento

Quanto ao aspecto da violação de SLA, a Figura 6.8 mostra o ganho alcançado pelo PMDB por utilizar uma estratégia preditiva. Neste caso, foi possível visualizar uma futura violação de SLA. A violação de SLA ocorreu por volta dos 2400 segundos, ou seja, aos 40 minutos do experimento; no entanto, a Figura 6.8 (a) mostra menor duração de violação de SLA no tempo de experimento, comparando com o PMDB utilizando provisionamento reativo que está ilustrado na Figura 6.8 (b). Isso foi possível pelo fato de que a migração ocorreu, para o PMDB, por volta de 100 segundos antes da violação do

SLA. Com isso, o tempo de violação foi de aproximadamente 117 segundos, levado para migrar, totalmente, o inquilino para outra VM. Já o PMDB com provisionamento reativo migrou o inquilino quando o SLA foi violado. A Tabela 6.6 mostra as novas alocações dos inquilinos após os 40 minutos de experimento.

VM	**Inquilinos com o PMDB**
VM_1	$Wikipédia_1$
VM_2	$YCSB_2$
VM_3	$Wikipédia_2$, $YCSB_1$
VM	**Inquilinos com o PMDB utilizando provisiomanento reativo**
VM_1	$Wikipédia_1$
VM_2	$YCSB_2$
VM_3	$Wikipédia_2$
VM_4	$YCSB_1$

Tabela 6.6 Alocação dos Inquilinos após os 40 Minutos de Experimento

No instante de 60 minutos de experimento, o inquilino $Wikipédia_2$ viola o SLA. Pela Figura 6.7 pode-se notar que o PMDB teve um desempenho inferior ao PMDB com provisionamento reativo, pois o tempo de resposta médio do SLA foi um pouco maior. No caso do PMDB, o serviço de provisionamento conseguiu antecipar o processo de migração do $Wikipédia_2$ para a VM_1. Mesmo assim, o tempo de resposta médio do PMDB foi de 0,521 segundo.

O tempo de resposta médio do PMDB com provisionamento reativo foi de 0,401 segundo, não violando SLA, pois o inquilino se encontra isolado em uma VM. O tempo de migração do $Wikipédia_2$ foi de 58 segundos. É possível perceber que todos os inquilinos do PMDB com provisionamento reativo tiveram seus tempos de resposta médio diminuídos, comparando com o PMDB. O PMDB, no entanto, não precisou realizar nenhum provisionamento, mesmo assim, manteve a qualidade dos inquilinos com pequenas violações de SLA. A organização dos inquilinos após os 60 minutos de experimento pode ser visualizada na Tabela 6.7.

VM	Inquilinos com o PMDB
VM_1	$Wikipédia_1$, $Wikipédia_2$
VM_2	$YCSB_2$
VM_3	$YCSB_1$
VM	**Inquilinos com o PMDB utilizando provisiomanento reativo**
VM_1	$Wikipédia_1$
VM_2	$YCSB_2$
VM_3	$Wikipédia_2$
VM_4	$YCSB_1$

Tabela 6.7 Alocação dos Inquilinos após os 60 Minutos de Experimento

6.2 CONCLUSÃO

Este capítulo exibiu e discutiu a estratégia de avaliação proposta para o PMDB. Foi justificado a escolha dos sistemas/*frameworks* utilizados, do ambiente e dos experimentos a serem realizados, para demonstrar a eficácia desta abordagem para melhorar a QoS de serviços de dados em nuvem. O próximo capítulo expressa as conclusões deste trabalho. Aqui, o objetivo é mostrar que o PMDB tende a violar menos SLA e aproveitar os recursos que estão à disposição do ambiente.

CAPÍTULO 7

CONCLUSÃO

Este capítulo encerra a tese com as considerações finais e as etapas futuras para continuidade do trabalho.

7.1 CONSIDERAÇÕES FINAIS

O trabalho apresentou o PMDB, uma abordagem para manter a QoS em SGBDs em nuvem, utilizando de maneira eficiente para melhorar o uso dos recursos do ambiente por meio das técnicas de migração, alocação e predição. Em particular, o PMDB contempla os requisitos necessários para QoS em nuvem, considerando banco de dados relacional no modelo multi-inquilino de SGBD compartilhado. A arquitetura do PMDB foi descrita e os algoritmos desenvolvidos apresentados. As técnicas para migração e alocação de inquilinos em nuvem foram propostas e métodos para predição de carga de trabalho de bancos de dados foram analisados. Por fim, realizou-se uma avaliação detalhada do PMDB, com a justificação da escolha dos sistemas utilizados, bem como o ambiente e experimentos conduzidos na avaliação.

As principais conclusões da tese encontram-se na sequência.

- A abordagem PMDB que se serve do *Live Migration* preditivo dos serviços de dados em nuvem, diminui o número de violações de SLA por parte dos inquilinos, ao mesmo tempo que tenta aproveitar os recursos existentes. Para isso, o PMDB adota uma arquitetura para banco de dados multi-inquilino, que propõe a alocação de bancos de dados e o provisionamento de recursos para melhorar a qualidade de serviço de bancos de dados com base em informações coletadas sobre a carga de trabalho e o ambiente, ou seja, a estratégia de monitoramento. A abordagem é independente do sistema gerenciador de banco de dados e pode ser utilizada em qualquer infraestrutura de nuvem.

- A técnica para a alocação de serviços de dados em nuvem, baseada em SLA, objetiva o aproveitamento dos recursos existentes no ambiente. O modelo matemático orientado ao desempenho dos inquilinos, ambiente e SLA serve como critério de decisão para atribuir uma nova localidade para um dado inquilino. Este modelo visa a melhorar a utilização dos recursos à disposição e foi implementado como um serviço e utiliza as informações da estratégia de monitoramento. Caso o modelo matemático não encontre um local adequado para o inquilino, ou seja, a atual localização seja a melhor, o serviço de alocação sinaliza a necessidade de um provisionamento de nova VM.

- A técnica para a predição de cargas de trabalho para serviços de dados em nuvem pode auxiliar na tomada de decisão sobre reconfiguração dos inquilinos no ambiente, sob o ponto de vista de sua localização. Com base nas informações de monitoramento do ambiente e das cargas de trabalho, utilizando análise preditiva, a antevisto o comportamento do inquilino em uma futura janela de tempo. Neste sentido e objetivando a não ocorrência de violações de SLA, o provedor pode se servir desta abordagem e tomar decisão sobre o inquilino.

- A estratégia para a qualidade de serviços de dados em nuvem, adotada pelo PMDB, abrange os conceitos de SLA para SGBDs e técnicas de monitoramento para garantir a qualidade do serviço por meio do gerenciamento de bancos de dados. A solução proposta objetiva manter o nível do serviço dentro dos limites definidos, auxiliando as técnicas de alocação e predição.

- A implementação de uma arquitetura para o gerenciamento de banco de dados está baseada em abordagens de migração, alocação e predição. O protótipo desenvolvido fornece suporte à infraestrutura Amazon AWS e pode ser facilmente estendido para outras infraestruturas, uma vez que todas as informações necessárias para sua utilização podem ser extraídas dos SGBDs e das infraestruturas, por meio de agentes de monitoramento.

- A avaliação de serviço de banco de dados multi-inquilino em nuvem, utilizado pelo PMDB, permite formar um ambiente multi-inquilino de SGBD compartilhado para avaliação, e privilegia aspectos da qualidade do serviço e utilização de recursos em nuvem.

- Os resultados dos experimentos realizados confirmaram que PMDB pode melhorar a qualidade de serviço com redução da quantidade de violação de SLA, enquanto

aproveita os recursos existentes no ambiente, mesmo em cenários com diferentes variações de carga de trabalho. Estes resultados comprovam a hipótese deste ensaio de que migração preditiva e provisionamento de recursos, também de forma preditiva, melhoram a qualidade de serviço e a utilização de recursos em ambientes de nuvem.

- Como limitações e diferente de alguns trabalhos correlatos, o PMDB não trata a questão da elasticidade. O PMDB realiza, somente, provisionamento de VMs quando há aumento das cargas de trabalho dos inquilinos. O SLA utilizado pelo PMDB é baseado, somente, em tempo de resposta e consistência forte. Por fim, o PMDB só satisfaz aplicações que utilizam o modelo multi-inquilino de SGBD compartilhado.

7.2 TRABALHOS FUTUROS

Os trabalhos propostos como atividades a serem feitas posteriormente para dar continuidade a este trabalho são os que estão na sequência.

1. *Níveis de Consistência Distintos*

PMDB foi desenvolvido para trabalhar com consistência forte. Por outro lado, existe grande quantidade de aplicações que não necessita de consistência forte. De acordo com [Kraska et al., 2009], consistência e custo estão relacionados e devem ser considerados no desenvolvimento de soluções em nuvem. Consistência forte implica alto custo por transação e, em algumas situações, reduz a disponibilidade. Consistência fraca denota menor custo.

Em [Kraska et al., 2009], é expresso um novo paradigma que permite aos desenvolvedores definir garantias de consistência e o chaveamento automático destas garantias em tempo de execução, com o objetivo de minimizar os custos. Para garantir a disponibilidade e a tolerância a falhas, os serviços de dados em nuvem devem fornecem distintas garantias de consistência.

Em virtude da diversidade de aplicações, pretende-se estender PMDB para implementar vários níveis, tais como consistência fraca ou consistência eventual. De acordo com os requisitos da aplicação em relação à consistência, um nível de consistência para o banco de dados desta aplicação poderia ser utilizado, adequando o custo à consistência

utilizada.

2. *Novos Modelos Multi-Inquilinos*

PMDB utiliza o modelo multi-inquilino de SGBD compartilhado, que permite um bom compartilhamento de recursos. Por outro lado, este modelo exprime interferências entre os inquilinos. O estudo de [Moreira et al., 2012] contém a análise de um conjunto de experimentos para medir a variação do desempenho de SGBDs multi-inquilino em nuvem. A análise relata em detalhes as interferências e suas possíveis causas, tais como a carga de trabalho e o tipo de SGBD utilizado.

Com efeito, as soluções em nuvem devem alocar inquilinos com pouca interferência entre si. Para tanto, é necessária uma análise do perfil do inquilino para identificar o nível de interferência. Também é importante isolar inquilinos suscetíveis à interferência. Neste caso, outros modelos multi-inquilinos podem ser utilizados, por exemplo, SGBD privado para inquilinos suscetíveis à interferência. Além disso, para diminuir tais interferências, é necessário identificar o nível de interação dos inquilinos e, consequentemente, melhorar a alocação dos inquilinos de acordo com suas interferências.

Em trabalhos futuros, pretende-se realizar experimentos com outros modelos multi-inquilinos não trabalhados nesta pesquisa, assim como verificar o nível de interferências em outros SGBDs e os recursos utilizados por cada inquilino.

3. *Novos Experimentos de QoS*

Os experimentos realizados neste trabalho abordaram aspectos específicos de utilização de recursos e QoS do sistema avaliado. Intenta-se-á efetuar experimentos com cargas de trabalho maiores e com variações em um maior intervalo de tempo. Tais experimentos permitirão avaliar o comportamento do PMDB nestes cenários e, se for o caso, aprimorar seus algoritmos. Em relação a experimentos para avaliar a disponibilidade, tenciona-se desenvolver experimentos para validar a dependabilidade total do PMDB, englobando também a confiabilidade. Para tanto, é necessário desenvolver experimentos detalhados com injeção de falhas.

Também é proposta a avaliação do PMDB em diferentes zonas de disponibilidade, com o intuito de identificar a variação no desempenho adicionado em decorrência da latência da rede. Para tanto, é necessário identificar parâmetros e desenvolver cenários

que cubram um ambiente mais geral do que o utilizado nos experimentos aqui exibidos.

PMDB foi avaliado com a infraestrutura da Amazon; entretanto, ele pode ser utilizado por qualquer infraestrutura. Em trabalhos futuros, poder-se-ia avaliá-lo com outras infraestruturas, como OpenNebula, OpenStack e CloudStack, verificando a melhoria proporcionada aos serviços executados em cada uma dessas infraestruturas.

4. *Processo de Tomada de Decisão para Remoção de VMs*

PMDB utiliza uma técnica de alocação baseada nas cargas de trabalho e SLA dos inquilinos para determinar se é necessária a adição de novas VMs. Apesar de esta técnica indicar resultados satisfatórios, ela não contempla características de elasticidade, pois só há mudança na infraestrutura por meio do provisionamento VM. O ideal seria que o PMDB pudesse perceber que alguns inquilinos, de várias VMs, podendo compartilhar a mesma VM sem violar SLA; com isso, removendo algumas VMs e diminuindo o custo de infraestrutura da perspectiva do provedor.

De acordo com uma investigação inicial, percebeu-se que o problema abordado pelo PMDB pode ser modelado como um problema de decisão. PMDB monitora o estado do ambiente e executa ações baseadas na qualidade do serviço. Assim sendo, um direcionamento para trabalhos futuros é utilizar o *Markov Decision Process* (MDP) para auxiliar no processo de remoção de VMs e realocar os inquilinos do ambiente.

5. *Modelo de Custo para SGBDs em Nuvem*

Os provedores de serviços em nuvem devem tratar aspectos de custo baseados na utilização dos SGBDs. As IaaS fornecem serviços de armazenamento e transferência de dados, normalmente com custos adicionais referentes a quantidade de dados armazenados e de requisições de E/S. No contexto deste trabalho, os custos dos serviços de armazenamento foram relativamente menores do que os custos associados às VM. Além disso, cada provedor utiliza uma estratégia diferente de tarifação. Dessa forma, o trabalho considerou apenas os custos referentes às VMs.

Outros trabalhos propuseram modelos de custo abordando diversos aspectos do gerenciamento de dados, como armazenamento, processamento e E/S aos dados [Viana et al., 2011] [Floratou et al., 2012] [Upadhyaya et al., 2012] [Nguyen et al., 2012]. Assim, outro direcionamento consiste em utilizar ou estender estes modelos, de forma a permitir ao PMDB

contabilizar todos os custos associados ao SGBD e, consequentemente, fornecer uma solução com contabilização total dos custos para os usuários. Com isso, mostrando, em termos monetários, a viabilidade de migrar um inquilino ou provisionar uma nova VM.

REFERÊNCIAS BIBLIOGRÁFICAS

[Abadi, 2009] Abadi, D. J. (2009). Data management in the cloud: Limitations and opportunities. *IEEE Data Eng. Bull.*, 32(1):3–12.

[Aboulnaga et al., 2009] Aboulnaga, A., Salem, K., Soror, A. A., Minhas, U. F., Kokosielis, P., and Kamath, S. (2009). Deploying database appliances in the cloud. *IEEE Data Eng. Bull.*, 32(1):13–20.

[Abouzeid et al., 2009] Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A., and Rasin, A. (2009). Hadoopdb: An architectural hybrid of mapreduce and dbms technologies for analytical workloads. *Proc. VLDB Endow.*, 2(1):922–933.

[Agrawal et al., 2010] Agrawal, D., Das, S., and Abbadi, A. E. (2010). Big data and cloud computing: New wine or just new bottles?. *PVLDB*, 3(2):1647–1648.

[Ahmad and Bowman, 2011] Ahmad, M. and Bowman, I. T. (2011). Predicting system performance for multi-tenant database workloads. In *Proceedings of the Fourth International Workshop on Testing Database Systems*, DBTest '11, pages 6:1–6:6, New York, NY, USA. ACM.

[Alpaydin, 2010] Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2nd edition.

[Anderson, 1976] Anderson, O. (1976). *Time series analysis and forecasting: the Box-Jenkins approach*. Butterworth.

[Aulbach et al., 2008] Aulbach, S., Grust, T., Jacobs, D., Kemper, A., and Rittinger, J. (2008). Multi-tenant databases for software as a service: Schema-mapping techniques. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1195–1206, New York, NY, USA. ACM.

[Ayhan et al., 2012] Ayhan, S., Pesce, J., Comitz, P., Gerberick, G., and Bliesner, S. (2012). Predictive analytics with surveillance big data. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '12, pages 81–90, New York, NY, USA. ACM.

[Azure, 2014] Azure (2014). *Microsoft Azure*. `http://www.microsoft.com/azure/`.

[Barker et al., 2012] Barker, S., Chi, Y., Moon, H. J., Hacigümüş, H., and Shenoy, P. (2012). "cut me some slack": latency-aware live migration for databases. In *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pages 432–443, New York, NY, USA. ACM.

[Bernstein and Newcomer, 2009] Bernstein, P. and Newcomer, E. (2009). *Principles of transaction processing: for the systems professional*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, second edition.

[Binnig et al., 2009] Binnig, C., Kossmann, D., Kraska, T., and Loesing, S. (2009). How is the weather tomorrow?: Towards a benchmark for the cloud. In *Proceedings of the Second International Workshop on Testing Database Systems*, DBTest '09, pages 9:1–9:6, New York, NY, USA. ACM.

[Box and Jenkins, 1976] Box, G. and Jenkins, G. (1976). *Time series analysis: forecasting and control*. Holden-Day series in time series analysis and digital processing. Holden-Day.

[Brockwell and Davis, 2009] Brockwell, P. and Davis, R. (2009). *Time Series: Theory and Methods*. Springer Series in Statistics. Springer.

[Buyya et al., 2009] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616.

[Chaudhuri, 2012] Chaudhuri, S. (2012). What next?: A half-dozen data management research goals for big data and the cloud. In *Proceedings of the 31st Symposium on Principles of Database Systems*, PODS '12, pages 1–4, New York, NY, USA. ACM.

[Chi et al., 2011] Chi, Y., Moon, H. J., Hacigümüş, H., and Tatemura, J. (2011). Sla-tree: A framework for efficiently supporting sla-based decisions in cloud computing. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 129–140, New York, NY, USA. ACM.

[Ciurana, 2009] Ciurana, E. (2009). *Developing with Google App Engine*. FirstPress (En ligne). Apress.

[Comellas et al., 2010] Comellas, J. O. F., Presa, I. G., and Fernández, J. G. (2010). Sla-driven elastic cloud hosting provider. In *Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, PDP '10, pages 111–118, Washington, DC, USA. IEEE Computer Society.

[Contreras-Reyes and Palma, 2013] Contreras-Reyes, J. E. and Palma, W. (2013). Statistical analysis of autoregressive fractionally integrated moving average models in r. *Comput. Stat.*, 28(5):2309–2331.

[Cooper et al., 2010] Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 143–154, New York, NY, USA. ACM.

[Curino et al., 2011] Curino, C., Jones, E. P. C., Popa, R., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. (2011). Relational cloud: a database service for the cloud. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, Online Proceedings*.

[Das and Kempe, 2008] Das, A. and Kempe, D. (2008). Algorithms for subset selection in linear regression. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 45–54, New York, NY, USA. ACM.

[Das et al., 2010] Das, S., Nishimura, S., Agrawal, D., and Abbadi, A. E. (2010). Live database migration for elasticity in a multitenant database for cloud platforms. Technical Report 2010-09, UCSB CS.

[Das et al., 2011] Das, S., Nishimura, S., Agrawal, D., and El Abbadi, A. (2011). Albatross: Lightweight elasticity in shared storage databases for the cloud using live data migration. *Proc. VLDB Endow.*, 4(8):494–505.

[Database, 2014] Database, H. (2014). *H2 Database Engine*. `http://www.h2database.com`.

[de Micheaux et al., 2014] de Micheaux, P. L., Drouilhet, R., and Liquet, B. (2014). *The R Software: Fundamentals of Programming and Statistical Analysis*. Springer Publishing Company, Incorporated.

[Debahuti Mishra and Mishra, 2010] Debahuti Mishra, Asit Kumar Das, M. and Mishra, S. (2010). Predictive data mining: Promising future and applications. *International Journal of Computer Communication and Technology (IJCCT)*, 2(1):20–28.

[Difallah et al., 2013] Difallah, D. E., Pavlo, A., Curino, C., and Cudré-Mauroux, P. (2013). Oltp-bench: An extensible testbed for benchmarking relational databases. *PVLDB*, 7(4):277–288.

[Dorffner, 1996] Dorffner, G. (1996). Neural networks for time series processing. *Neural Network World*, 6:447–468.

[Elmore et al., 2011a] Elmore, A., Das, S., Agrawal, D., and Abbadi, A. E. (2011a). Towards an elastic and autonomic multitenant database. In *NetDB 2011 - 6th International Workshop on Networking Meets Databases Co-located with SIGMOD 2011*.

[Elmore et al., 2011b] Elmore, A. J., Das, S., Agrawal, D., and El Abbadi, A. (2011b). Zephyr: Live migration in shared nothing databases for elastic cloud platforms. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 301–312, New York, NY, USA. ACM.

[Entrialgo et al., 2011] Entrialgo, J., García, D. F., García, J., García, M., Valledor, P., and Obaidat, M. S. (2011). Dynamic adaptation of response-time models for qos management in autonomic systems. *J. Syst. Softw.*, 84(5):810–820.

[Floratou et al., 2012] Floratou, A., Patel, J. M., Lang, W., and Halverson, A. (2012). When free is not really free: what does it cost to run a database workload in the cloud? In *Proceedings of the Third TPC Technology conference on Topics in Performance Evaluation, Measurement and Characterization*, TPCTC'11, pages 163–179, Berlin, Heidelberg. Springer-Verlag.

[Florescu and Kossmann, 2009] Florescu, D. and Kossmann, D. (2009). Rethinking cost and performance of database systems. *SIGMOD Rec.*, 38(1):43–48.

[Gardener, 2012] Gardener, M. (2012). *Beginning R: The Statistical Programming Language*. Wrox Press Ltd., Birmingham, UK, UK, 1st edition.

[Hatem A. Mahmoud and El-Abbadi, 2012] Hatem A. Mahmoud, Hyun Jin Moon, Y. C. H. H. D. A. and El-Abbadi, A. (2012). Towards multitenancy for io-bound olap workloads. Technical Report UCSB-2012-02, CS Department, University of California, Santa Barbara.

[Herrera et al., 1999] Herrera, J., Cellier, F., Gennari, G., de Robòtica i Informàtica Industrial, I., and de Catalunya. Institut d'Organització i Control de Sistemes Industrials, U. P. (1999). *Time series prediction using inductive reasoning techniques.*

[Hui et al., 2009] Hui, M., Jiang, D., Li, G., and Zhou, Y. (2009). Supporting database applications as a service. In Ioannidis, Y. E., Lee, D. L., and Ng, R. T., editors, *ICDE*, pages 832–843. IEEE.

[Huppler, 2010] Huppler, K. (2010). Price and the tpc. In Nambiar, R. O. and Poess, M., editors, *TPCTC*, volume 6417 of *Lecture Notes in Computer Science*, pages 73–84. Springer.

[Islam et al., 2012] Islam, S., Keung, J., Lee, K., and Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Future Gener. Comput. Syst.*, 28(1):155–162.

[Jacobs et al., 2007] Jacobs, D., Aulbach, S., and München, T. U. (2007). Ruminations on multi-tenant databases. In *BTW Proceedings, volume 103 of LNI*, pages 514–521. GI.

[Kok, 2007] Kok, S. (2007). Liquid State Machine Optimization.

[Kraska et al., 2009] Kraska, T., Hentschel, M., Alonso, G., and Kossmann, D. (2009). Consistency rationing in the cloud: Pay only when it matters. *PVLDB*, 2(1):253–264.

[Lang et al., 2012] Lang, W., Shankar, S., Patel, J. M., and Kalhan, A. (2012). Towards multitenant performance slos. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 702–713, Washington, DC, USA. IEEE Computer Society.

[Lendasse et al., 2004] Lendasse, A., Wertz, V., Simon, G., and Verleysen, M. (2004). Fast bootstrap applied to LS-SVM for long term prediction of time series. In *In Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, volume 1, pages 705–710. IEEE.

[Liu et al., 2007] Liu, S., Liang, Y., and Brooks, M. (2007). Eucalyptus: A web service-enabled e-infrastructure. In *Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '07, pages 1–11, Riverton, NJ, USA. IBM Corp.

[Malkowski et al., 2010] Malkowski, S., Hedwig, M., Jayasinghe, D., Pu, C., and Neumann, D. (2010). Cloudxplor: A tool for configuration planning in clouds based on empirical data. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 391–398, New York, NY, USA. ACM.

[Martinez, 2012] Martinez, C. G. (2012). Study of Resource Management for Multitenant Database Systems in Cloud Computing. Master's thesis, Faculty of the Graduate School of the University of Colorado, Colorado, United States.

[Moon et al., 2013] Moon, H. J., Hacümüş, H., Chi, Y., and Hsiung, W.-P. (2013). Swat: a lightweight load balancing method for multitenant databases. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 65–76, New York, NY, USA. ACM.

[Moreira et al., 2014] Moreira, L. O., Farias, V. A. E., Sousa, F. R. C., Santos, G. A. C., Maia, J. G. R., and Machado, J. C. (2014). Towards improvements on the quality of service for multi-tenant rdbms in the cloud. In *Proceedings of the 6th International Workshop on Cloud Data Management*, CloudDB '14, pages 162–169, Chicago, IL, USA. IEEE.

[Moreira et al., 2012] Moreira, L. O., Sousa, F. R. C., and Machado, J. C. (2012). Analisando o desempenho de banco de dados multi-inquilino em nuvem. In *27th Brazilian Symposium on Databases*, SBBD '12, pages 161–168.

[Moreira et al., 2013] Moreira, L. O., Sousa, F. R. C., Maia, J. G. R., Farias, V. A. E., Santos, G. A. C., and Machado, J. C. (2013). A live migration approach for multi-tenant rdbms in the cloud. In *28th Brazilian Symposium on Databases*, SBBD '13, pages 73–78.

[Mukherjee et al., 1997] Mukherjee, S., Osuna, E., and Girosi, F. (1997). Nonlinear prediction of chaotic time series using support vector machines. In Principe, J., Giles, L., Morgan, N., and Wilson, E., editors, *IEEE Workshop on Neural Networks for Signal Processing VII*. IEEE Press.

[Natrella, 2010] Natrella, M. (2010). *NIST/SEMATECH e-Handbook of Statistical Methods*. NIST/SEMATECH.

[Nguyen et al., 2012] Nguyen, T.-V.-A., Bimonte, S., d'Orazio, L., and Darmont, J. (2012). Cost models for view materialization in the cloud. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12, pages 47–54, New York, NY, USA. ACM.

[Plummer, 2000] Plummer, E. (2000). *Time Series Forecasting with Feed-forward Neural Networks: Guidelines and Limitations*. University of Wyoming.

[Prevost et al., 2011] Prevost, J. J., Nagothu, K., Kelley, B., and Jamshidi, M. (2011). Prediction of cloud data center networks loads using stochastic and neural models. In *System of Systems Engineering (SoSE), 2011 6th International Conference on*, pages 276–281.

[QoSDBC, 2014] QoSDBC (2014). *QoSDBC*. `http://code.google.com/p/qosdbc/`.

[Ramakrishnan and Gehrke, 2003] Ramakrishnan, R. and Gehrke, J. (2003). *Database Management Systems*. McGraw-Hill higher education. McGraw-Hill Education.

[Robinson, 2008] Robinson, D. (2008). *Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQS Web Services Enables You to Reach Business Goals Faster*. Emereo Pty Limited.

[Sakr and Liu, 2012] Sakr, S. and Liu, A. (2012). Sla-based and consumer-centric dynamic provisioning for cloud databases. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, CLOUD '12, pages 360–367, Washington, DC, USA. IEEE Computer Society.

[Salesforce, 2014] Salesforce (2014). *Salesforce*. `http://www.salesforce.com/`.

[Santos, 2013] Santos, G. A. C. (2013). S-SWAP: Scale-Space Based Workload Analysis and Prediction. Master's thesis, Departamento de Computação, Universidade Federal do Ceará (UFC), Fortaleza, Ceará, Brasil.

[Santos et al., 2013] Santos, G. A. C., Maia, J. G. R., Moreira, L. O., Sousa, F. R. C., and Machado, J. C. (2013). Scale-space filtering for workload analysis and forecast. In *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, CLOUD '13, pages 677–684, Washington, DC, USA. IEEE Computer Society.

[Schad et al., 2010] Schad, J., Dittrich, J., and Quiané-Ruiz, J.-A. (2010). Runtime measurements in the cloud: Observing, analyzing, and reducing variance. *Proc. VLDB Endow.*, 3(1-2):460–471.

[Schiller et al., 2013] Schiller, O., Cipriani, N., and Mitschang, B. (2013). Prorea: live database migration for multi-tenant rdbms with snapshot isolation. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 53–64, New York, NY, USA. ACM.

[Schnjakin et al., 2010] Schnjakin, M., Alnemr, R., and Meinel, C. (2010). Contract-based cloud architecture. In *Proceedings of the Second International Workshop on Cloud Data Management*, CloudDB '10, pages 33–40, New York, NY, USA. ACM.

[Schölkopf et al., 2000] Schölkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. (2000). New support vector algorithms. *Neural Comput.*, 12(5):1207–1245.

[Schroeder et al., 2006] Schroeder, B., Harchol-Balter, M., Iyengar, A., and Nahum, E. (2006). Achieving class-based qos for transactional workloads. In *Proceedings of the 22Nd International Conference on Data Engineering*, ICDE '06, pages 153–, Washington, DC, USA. IEEE Computer Society.

[Sethuraman and Taheri, 2011] Sethuraman, P. and Taheri, H. R. (2011). Tpc-v: A benchmark for evaluating the performance of database applications in virtual environments. In *Proceedings of the Second TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems*, TPCTC'10, pages 121–135, Berlin, Heidelberg. Springer-Verlag.

[Sezgin and Davis, 2006] Sezgin, T. M. and Davis, R. (2006). Scale-space based feature point detection for digital ink. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA. ACM.

[Soror et al., 2008] Soror, A. A., Minhas, U. F., Aboulnaga, A., Salem, K., Kokosielis, P., and Kamath, S. (2008). Automatic virtual machine configuration for database workloads. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 953–966, New York, NY, USA. ACM.

[Sousa et al., 2009] Sousa, F. R., Moreira, L. O., and Machado, J. C. (2009). Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. In *III Escola Regional de Computação Ceará, Maranhão e Piauí ERCEMAPI*, pages 150–175. EDUFPI.

[Sousa, 2013] Sousa, F. R. C. (2013). *RepliC: Replicação Elástica de Banco de Dados Multi-Inquilino em Nuvem com Qualidade de Serviço*. Doutorado, Departamento de Computação, Universidade Federal do Ceará (UFC), Fortaleza, Ceará, Brasil.

[Sousa and Machado, 2012] Sousa, F. R. C. and Machado, J. C. (2012). Towards elastic multi-tenant database replication with quality of service. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, UCC '12, pages 168–175, Washington, DC, USA. IEEE Computer Society.

[Sousa et al., 2010] Sousa, F. R. C., Moreira, L. O., Macêdo, J. A. F., and Machado, J. C. (2010). Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. In *Simpósio Brasileiro de Banco de Dados, SBBD 2010, 1. ed.*, pages 101–130, Belo Horizonte. SBC.

[Sousa et al., 2011] Sousa, F. R. C., Moreira, L. O., and Machado, J. C. (2011). Sladb: Acordo de nível de serviço para banco de dados em nuvem. In *26th Brazilian Symposium on Databases*, SBBD '11, pages 155–162.

[Sousa et al., 2012] Sousa, F. R. C., Moreira, L. O., Santos, G. A. C., and Machado, J. C. (2012). Quality of service for database in the cloud. In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science*, CLOSER '12, pages 595–601.

[Tattar, 2013] Tattar, P. N. (2013). *R Statistical Application Development by Example Beginner's Guide*. Packt Publishing.

[Tözün et al., 2013] Tözün, P., Pandis, I., Kaynak, C., Jevdjic, D., and Ailamaki, A. (2013). From a to e: Analyzing tpc's oltp benchmarks: The obsolete, the ubiquitous, the unexplored. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 17–28, New York, NY, USA. ACM.

[Upadhyaya et al., 2012] Upadhyaya, P., Balazinska, M., and Suciu, D. (2012). How to price shared optimizations in the cloud. *Proc. VLDB Endow.*, 5(6):562–573.

[Vaquero et al., 2008] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55.

[Vasić et al., 2012] Vasić, N., Novaković, D., Miučin, S., Kostić, D., and Bianchini, R. (2012). Dejavu: Accelerating resource allocation in virtualized environments. *SIGARCH Comput. Archit. News*, 40(1):423–436.

[Viana et al., 2011] Viana, V., de Oliveira, D., and Mattoso, M. (2011). Towards a cost model for scheduling scientific workflows activities in cloud environments. *Services, IEEE Congress on*, 0:216–219.

[Voulgaropoulou et al., 2012] Voulgaropoulou, S., Spanos, G., and Angelis, L. (2012). Analyzing measurements of the r statistical open source software. In *Proceedings of the 2012 35th Annual IEEE Software Engineering Workshop*, SEW '12, pages 1–10, Washington, DC, USA. IEEE Computer Society.

[Wang et al., 2012] Wang, L., Fu, D., and Li, Q. (2012). Samples selection based on svr for prediction of steel mechanical property. *2013 Third International Conference on Intelligent System Design and Engineering Applications*, 0:909–912.

[Wang and Xu, 2004] Wang, W. and Xu, Z. (2004). A heuristic training for support vector regression. *Neurocomputing*, 61:259–275.

[Weissman and Bobrowski, 2009] Weissman, C. D. and Bobrowski, S. (2009). The design of the force.com multitenant internet application development platform. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 889–896, New York, NY, USA. ACM.

[Witkin, 1983] Witkin, A. P. (1983). Scale-space filtering. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'83, pages 1019–1022, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Xiong et al., 2011] Xiong, P., Chi, Y., Zhu, S., Moon, H. J., Pu, C., and Hacigumus, H. (2011). Intelligent management of virtualized resources for database systems in cloud environment. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 87–98, Washington, DC, USA. IEEE Computer Society.

[Yang et al., 2009] Yang, F., Shanmugasundaram, J., and Yerneni, R. (2009). A scalable data platform for a large number of small applications. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, Online Proceedings*, pages 1–10.

[Zhang et al., 1998] Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1):35–62.