Uma Abordagem Baseada em Níveis de Estresse para Alocação Elástica de Recursos em Sistema de Bancos de Dados

Diego Henrique Pagani[1], **Luis Carlos Erpen De Bona**[1], **Guilherme Galante**[2]

[1]Departamento de Informática
Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – 81.531-980 – Curitiba-PR

[2]Centro de Ciências Exatas e Tecnológicas
Universidade Estadual do Oeste do Paraná (UNIOESTE)
Caixa Postal 711 – 85.819-110 – Cascavel-PR

{dhpagani,bona}@inf.ufpr.br, guilherme.galante@unioeste.br

Abstract. *This work proposes an alternative approach for exploring the elasticity in databases. In the proposed approach, the system's workload is estimated by the DBMS response time and classified in stress levels using a state machine. Based on the stress level, elasticity actions could be used to maintain the desired performance and to keep an appropriate amount of resources. The elastic allocation based on stress levels is validated by a set of experiments in which the proposed approach is successfully used in the dynamic allocation of virtual processors to a DBMS, maintaining the expected performance.*

Resumo. *Este trabalho tem como objetivo propor uma abordagem alternativa para a exploração da elasticidade em banco de dados. Na abordagem proposta, a carga no sistema é estimada pelo tempo de resposta do banco de dados e classificada em níveis de estresse utilizando uma máquina de estados. Baseado no nível que o sistema se encontra, pode-se alocar ou liberar recursos, mantendo o desempenho desejado com a quantidade mais apropriada de recursos. A abordagem é validada por um conjunto de experimentos, nos quais é utilizada com sucesso na alocação dinâmica de processadores virtuais para um SGBD, mantendo o desempenho dentro do esperado.*

1. Introdução

Nos últimos anos, a computação em nuvem tem atraído a atenção da indústria e do mundo acadêmico, tornando-se cada vez mais comum encontrar na literatura casos de adoção de nuvens por parte das empresas e instituições de pesquisa. Um dos principais motivos é a possibilidade de aquisição de recursos de uma forma dinâmica e elástica. De fato, a elasticidade é um grande diferencial e é atualmente vista como indispensável para o modelo de computação em nuvem.

O termo elasticidade pode ser definido como a capacidade de um sistema de adicionar ou remover dinamicamente recursos computacionais utilizados por uma determinada aplicação ou usuário de acordo com a demanda [Herbst et al. 2013]. A capacidade dos recursos pode ser expandida ou reduzida utilizando elasticidade horizontal, isto é, adição ou remoção de nós (máquinas virtuais) do sistema, ou empregando elasticidade vertical,

que consiste na reconfiguração do hardware de máquinas virtuais pela adição ou remoção de CPUs, memória, e armazenamento. O conceito de elasticidade também pode ser estendido para aplicações. Uma aplicação elástica é aquela capaz de se adaptar às alterações na quantidade de recursos ou de alterar os recursos de acordo com a sua necessidade. Para que se possa tirar proveito da elasticidade, é necessário que tanto a arquitetura quanto a aplicação sejam capazes de suportá-la de alguma forma.

Em trabalho prévio [Galante and Bona 2012], diversos mecanismos de elasticidade foram descritos e classificados. Grande parte dos mecanismos analisados tem como objetivo escalar aplicações cliente-servidor com o intuito de evitar as questões relacionadas com o provisionamento excessivo ou insuficiente de recursos. No escopo de bancos de dados, que é o foco deste trabalho, as soluções do estado-da-arte concentram-se em técnicas de particionamento e migração, e replicação com balanceamento de carga para implementar elasticidade [Minhas et al. 2012].

Considerando o exposto, este trabalho tem como objetivo propor uma abordagem alternativa para a exploração da elasticidade em banco de dados. Pretende-se explorar a elasticidade vertical em banco de dados, utilizando-se dos níveis de estresse do banco de dados para decidir sobre as alocações de novos recursos. Para isso, a carga no sistema é estimada pelo tempo de resposta do SGBD e classificada em níveis de estresse diferentes utilizando uma máquina de estados. Baseado no nível que o SGBD se encontra, pode-se alocar ou liberar recursos, mantendo o desempenho desejado com a quantidade mais apropriada de recursos.

A alocação elástica baseada em níveis de estresse é validada por um conjunto de experimentos, nos quais a abordagem proposta é utilizada com sucesso na alocação dinâmica de processadores virtuais (VCPUs) para um SGBD e mantendo o desempenho dentro do esperado. Compara-se ainda a abordagem baseada em níveis de estresse com outras que utilizam as métricas de tempo de resposta e uso de CPU no provisionamento elástico de recursos [Lorido-Botran et al. 2014].

O restante do trabalho é organizado como segue. Na Seção 2 apresentam-se os trabalhos relacionados. Na Seção 3 aborda-se em detalhes a abordagem proposta para provisionamento de recursos elásticos para banco de dados. Na Seção 4 Na realiza-se a avaliação experimental. Por fim, a Seção 5 conclui este trabalho.

2. Elasticidade em Banco de Dados

É cada vez mais comum a migração de aplicações dos mais diversos tipos para plataformas de nuvem e bancos de dados não são exceção a esta tendência. Muitos serviços desta natureza tem se baseado em recursos de computação e de armazenamento fornecidos por provedores de nuvem. De fato, existem inúmeras vantagens para o modelo baseado em nuvem, como o modelo *pay-as-you-go*, onde se paga apenas pelos recursos realmente utilizados, e alocação de recursos flexíveis, onde os recursos de computação e armazenamento podem ser aumentados ou reduzidos de forma dinâmica com base na evolução das necessidades de carga de trabalho de uma aplicação. A chave para a realização desses benefícios é a *elasticidade*, que permite o ajuste dos recursos atribuídos a uma aplicação em resposta ao crescimento constante da carga de trabalho, variações sazonais, ou picos de carga repentinos [Galante and Bona 2012, Herbst et al. 2013].

Com o intuito de utilizar essa característica importante da computação em nuvem, diversas soluções de elasticidade para banco de dados vem sendo desenvolvidas e apresentadas na literatura. Em geral, a elasticidade neste escopo é implementada por meio de técnicas de particionamento, migração, e replicação com balanceamento de carga [Elmore et al. 2011a].

O particionamento é utilizado para escalar um banco de dados para múltiplos nós quando a carga excede a capacidade de um único servidor, dividindo os dados em partições entre os servidores. Cada partição geralmente é alocada em um servidor, como implementado em diversos SGBDs, como VoltDB [Minhas et al. 2012], NuoDB[1] e ElasTras [Das et al. 2009].

A migração consiste em mudar a aplicação para uma máquina com maior capacidade e pode de ser realizada utilizando técnicas de *stop-and-copy* [Elmore et al. 2011b] ou *live migration* [Sakr et al. 2011]. As técnicas de particionamento e migração podem ainda ser utilizadas em conjunto. Dessa forma, quando um novo servidor é criado, migram-se as partições de dados para a máquina recém criada. Os trabalhos de Elmore et al. (2011) e Das et al (2011), empregam técnicas de migração. Curino et al. (2011) e Serafini et al.(2014) apresentam uma solução que combina particionamento e migração.

Em geral, técnicas de replicação são empregadas em conjunto com balanceamento de carga. Um mecanismo clona a máquina virtual que hospeda o servidor de banco de dados existente que é então iniciado em um novo servidor físico, resultando em uma nova réplica que começa a responder às requisições. Na prática, a nova instancia pode não responder instantaneamente a estas requisições, pois é necessário que os dados sejam sincronizados para que não ocorram inconsistências dos dados, devendo-se considerar ainda a latência necessária de cópia entre as máquinas físicas e a configuração do balanceador de carga para que esta nova instancia faça parte do serviço. CloudBD AutoAdmin [Sakr et al. 2011] é um exemplo de solução que combina técnicas de particionamento e replicação de banco de dados.

Embora a migração possa ver vista como uma forma de implementar elasticidade vertical [Galante and Bona 2012], não foi encontrado em nossa revisão bibliográfica soluções que explicitamente utilizem a reconfiguração de recursos na exploração deste tipo elasticidade em banco de dados. Neste contexto, a principal contribuição deste trabalho é apresentar uma abordagem para a exploração da elasticidade vertical em banco de dados, baseado em níveis de estresse para a alocação dinâmica de recursos. Para isso, a carga no sistema é estimada pelo tempo de resposta do SGBD e classificada em níveis de estresse diferentes utilizando uma máquina de estados. Baseado no nível que o SGBD se encontra, pode-se alocar ou liberar recursos, mantendo o desempenho desejado com a quantidade mais apropriada de recursos. Mas detalhes da proposta são apresentados na Seção 3.

3. Provisionamento Elástico de Recursos Baseado em Níveis de Estresse

Um dos principais problemas existentes ao se utilizar uma nuvem computacional elástica é definir um momento adequado para reajustar os recursos computacionais de um sistema gerenciador de banco de dados (SGBD) e estimar a quantidade necessária neste instante de tempo. Desta forma, apresenta-se neste trabalho uma abordagem baseada na

[1] http://www.nuodb.com/

classificação em níveis de estresse de um SGDB para estimar o momento adequado de reconfiguração dos recursos computacionais. Para estimar a quantidade necessária de recuros é preciso que sejam estudados diversos parâmetros específicas de cada SGBD, e avaliar o impacto destas configurações em diferentes quantidades de recursos.

Em um ambiente de nuvem computacional elástica, um SGDB recebe uma determinada carga de trabalho a partir de seus usuários. Ao longo da sua execução, a carga pode sofrer variações, o que implica em uma mudança no consumo dos recursos, que se refletem em degradação de desempenho em momentos de alto consumo e em recursos subutilizados quando há baixa demanda. Desta forma, a possibilidade de variar os recursos baseando-se na carga do SGBD torna-se uma abordagem interessante, pois em situações ideais, garantirá que os recursos alocados sejam coerentes com a demanda existente. Para isso, é necessário estimar a carga dos sistema e então encontrar o momento adequado para a reconfiguração dos recursos.

Na abordagem proposta, a carga no sistema é estimada pelo tempo de resposta do SGBD e classificada em níveis de estresse diferentes utilizando a máquina de estados ENE - *Estados de Níveis de Estresse*. ENE é uma máquina de estados voltada para classificação de um SGBD, considerando a média e variância do tempo de resposta e o uso dos recursos alocados como métricas. Esta máquina é baseada na DSM (*Database State Machine*) [Meira et al. 2014], que categoriza um SGBD em cinco estados possíveis, baseados no desempenho que ele apresenta ao longo de uma carga crescente. Com a definição do nível de estresse e informações sobre os recursos, ações de elasticidade (alocação ou liberação de recursos) podem ser realizadas.

A Figura 1 mostra a representação da ENE, que é composta por seis estados, que representam as situações de desempenho possíveis que um SGBD pode apresentar. Cada estado foi projetado para analisar os possíveis problemas que a alta concorrência pode causar no SGDB, como lentidão (tempo de resposta acima do esperado), instabilidade (alguns clientes são respondidos rapidamente enquanto outros apresentam lentidão) e perda de comunicação.

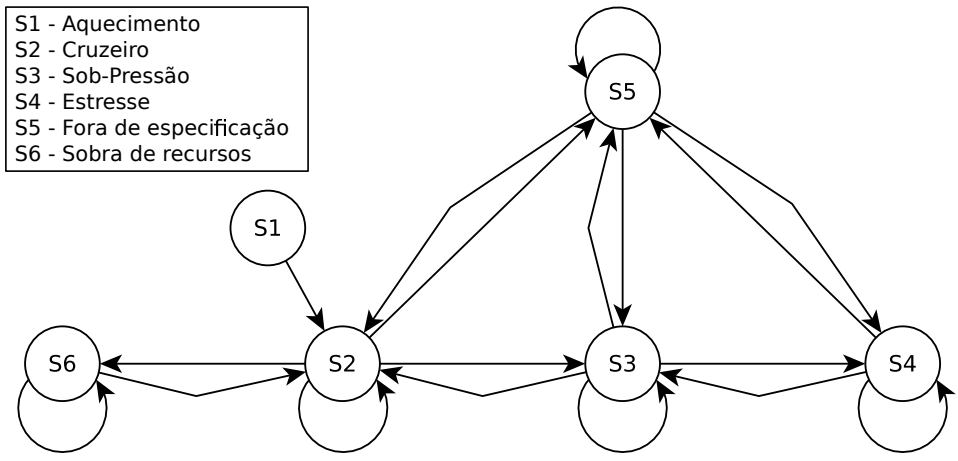


Figura 1. Máquina de estado ENE, baseada na DSM [Meira et al. 2014], que classifica um SGDB em seis possíveis níveis de estresse.

O estado $S1$ é chamado de **Aquecimento**, indica a inicialização do SGBD, que constitui do preenchimento de *caches* internos, que causa um atraso para as primeiras solicitações. A sua transição a outro estado ocorre quando o desempenho torna-se estável. Consideram-se neste estado os recursos mínimos[2] alocados e após a estabilização do tempo de resposta é que o sistema deixa este estado.

Após a saída de Aquecimento, o estado seguinte é chamado de **Cruzeiro**, indicado por $S2$, que representa um SGDB com baixo estresse, estável e respondendo com a velocidade controlada, com os recursos alocados serem adequados à carga. Deseja-se manter o SGBD neste estado o maior tempo possível.

Os estados $S3$ e $S4$ representam o desempenho aceitável, mas fora do ideal, indicando que os recursos alocados são insuficientes e recursos adicionais são necessários para que se retorne ao estado de Cruzeiro. O estado $S3$, chamado **Sob-Pressão**, indica um sistema com tempo de resposta acima do esperado (lentidão), mas a estabilidade (variância do tempo de resposta) ainda é garantida, representando um estresse baixo. Por sua vez, $S4$ indica o **Estresse** alto do SGDB, que representa para o usuário do SGBD lentidão e instabilidade.

Quando não é possível determinar a velocidade e estabilidade ou o tempo de resposta ultrapassa os limiares definidos para o estado $S4$, considera-se que o SGBD está no Estado $S5$ – **Fora da especificação**. Isso pode ocorrer devido a interferências, falha de comunicação ou picos de carga no sistema, e sugere uma medida emergencial, visto que não há informações recentes sobre o nível de estresse do SGBD. As medidas podem ser tomadas automaticamente (por exemplo, aumentar massivamente os recursos) ou por interferência externa, para que se obtenha alguma informação recente sobre seu estado. Diferentemente de $S5$, o estado $S6$ representa um SGBD rápido e estável, com estresse baixo e com recursos subutilizados, sendo recomendável a remoção dos recursos excedentes para que o estado seja alterado para $S2$.

As transições entre os estados representam alterações no comportamento do SGBD, seja por oscilações na carga, falha de rede ou alguma outra aplicação interferindo nos recursos, em ambientes compartilhados. As transições entre estes estados são definidas com base em duas variáveis: (i) Média do Tempo de Resposta (Velocidade) e (ii) Variância do Tempo de Resposta (Estabilidade). Para cada transição desta máquina, existe um conjunto de regras de disparo, que são baseados em limiares especificados no Acordo de Nível de Serviço.

3.1. Provisionador de Recursos Elásticos - PRE

Para permitir que a abordagem proposta seja utilizada, implementou-se um sistema chamado de *Provisionador de Recursos Elásticos* (PRE). O sistema é composto por dois módulos: (i) Inferidor de Estados e (ii) Controlador Elástico.

O *Inferidor de Estados* é responsável por realizar amostragens do tempo de resposta, armazenar estes valores e *inferir* o estado do SGBD, utilizando os conceitos definidos na máquina de estados ENE. A amostragem é feita por uma consulta que deve ser executada periodicamente no SGBD e armazenada em uma janela de N posições. O valor

[2]Todo *software* necessita de requisitos mínimos de *hardware* para operar e que naturalmente suporta uma quantidade de carga.

de N não deve ser grande o suficiente para aumentar a latência de detecção de estado, mas nem pequena para que leituras com pequenas oscilações naturais impactem na eficiência do provisionador.

Com esta janela preenchida e sempre alimentada com novas informações, é feita a classificação do nível de estresse do SGBD e informado ao *Controlador Elástico* o estado atual. Com esta informação e avaliando os recursos atuais e disponíveis, este módulo define se ações elásticas devem ser tomadas e as envia ao controlador da nuvem. As ações elásticas só são tomadas após ocorrer cinco classificações fora do estado de Cruzeiro, para que oscilações naturais não interfiram no resultado. Com a combinação destes módulos, o PRE é capaz de analisar a situação de desempenho que o SGBD se encontra e determina os recursos de modo adequado para a carga de trabalho recebida.

4. Avaliação Experimental

Considerando que a exploração da elasticidade horizontal no contexto banco de dados já é consolidada (ver Seção 2), neste trabalho avalia-se o uso da elasticidade vertical no provisionamento automático de vCPUs, utilizando a abordagem baseada em níveis de estresse.

Como forma de avaliar o modelo proposto, são apresentados dois experimentos. O primeiro visa validar o uso da elasticidade vertical em banco de dados, mostrando que a alocação ou remoção de vCPUs pode influenciar significativamente no tempo de resposta do sistema. O segundo experimento utiliza a abordagem baseada em níveis de estresse para alocação elástica de recursos em sistemas de bancos de dados e a compara com outras duas soluções encontradas na literatura.

4.1. Experimento 1: Uso da Elasticidade Vertical

Neste experimento objetiva-se validar o uso da elasticidade vertical em banco de dados, verificando-se o tempo de resposta do banco de dados em função do número de vCPUs alocadas para a máquina virtual que hospeda o SGBD.

Para gerar a carga de trabalho, executou-se um conjunto de consultas do *benchmark* TPC-H em uma instância de banco de dados PostgreSQL 9.5 incrementando o número de clientes conectados de zero até 800 clientes. Os clientes são simulados por um programa com múltiplas *threads*, sendo que cada *thread* representa um cliente. Ao todo, foram executadas 5 testes, utilizando 1, 2, 4, 8 e 16 vCPUs.

Como ambiente de execução empregou-se um servidor com três processadores AMD Opteron 6136 (totalizando 24 núcleos), 96 GB de memória, dois discos rígidos de 1 TB, no qual as máquinas virtuais XEN[3] são executadas. Em todas as máquinas, físicas e virtuais, utiliza-se a distribuição Ubuntu-Server 14.04.4, com *Kernel* 3.13.0-77. Os resultados são apresentados na Figura 2.

Como pode ser observado, o número de vCPUs alocadas impactou diretamente no tempo de resposta do SGBD e adicionar ou remover vCPUs se mostra uma abordagem interessante a ser analisada no contexto de banco de dados. Assim, dado uma determinada carga de trabalho (número de clientes) e uma expectativa de tempo de resposta, pode-se variar o número de processadores alocados em tempo de execução para alcançar o

[3]www.xenproject.org/

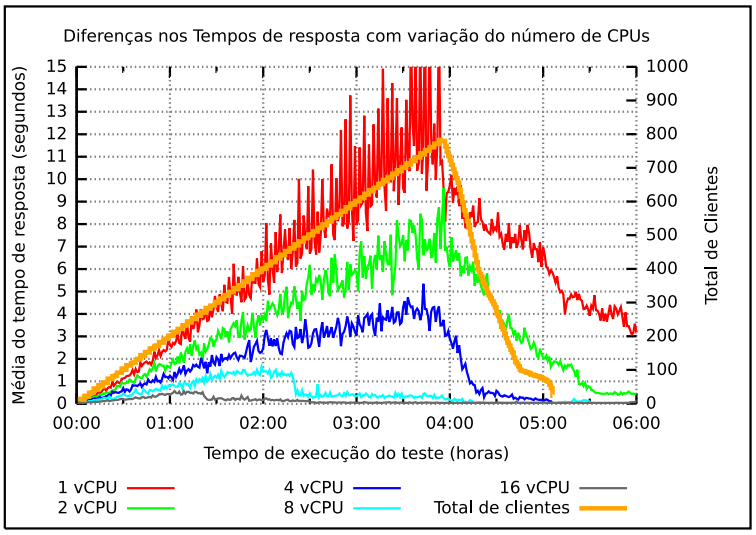


Figura 2. Teste com carga crescente utilizando diferentes quantidades de vCPUs.

resultado desejado. Considera-se ainda que a elasticidade permite que a infraestrutura seja alocada de modo apropriado, evitando tanto a falta de recursos para a aplicação quanto a existência de recursos ociosos.

4.2. Experimento 2: Alocação de Recursos Baseada em Estresse

Esta seção tem como objetivo apresentar os resultados da utilização da abordagem baseada em níveis de estresse para alocação elástica de recursos em sistemas de bancos de dados (PRE). Além disso, compara a abordagem proposta outras duas soluções encontradas na literatura [Lorido-Botran et al. 2014]: (i) baseada em tempo de resposta (TR) e (ii) baseada no uso de CPU (UCPU).

Os métodos PRE e TR utilizam como base o tempo de resposta, entretanto a interpretação desta métrica é diferente entre os métodos. O PRE utiliza o tempo de resposta médio e a variância, como forma de classificar o nível de estresse do SGBD e dependendo deste nível realiza ações de elasticidade. O método TR utiliza apenas o tempo médio de resposta da aplicação, apenas controlando-o entre limiares estabelecidos. Nestes métodos, é comum utilizar uma janela de tempo de tamanho N de modo a evitar as flutuações excessivas, que podem prejudicar o resultado final. A abordagem UCPU possui um funcionamento mais simples. Verifica-se a utilização do recurso e caso os valores estiverem fora de uma faixa pré-determinada (acima ou abaixo), solicita-se a reconfiguração dos recursos.

Para este experimento, deseja-se que o SGBD responda a uma consulta entre 2 e 3 segundos para os métodos PRE e TR, utilizando a menor quantidade de vCPU possível e que o sistema mantenha-se estável. Para o método UCPU é deseja-se que o uso de CPU esteja entre 30% e 80%.

Para o PRE é necessário uma configuração dos níveis de estresse. A Tabela 1 mostra os valores definidos para cada nível de estresse e a quantidade de vCPUs que devem ser alterados em cada um deles. Considerando que deseja-se manter o banco de dados no estado de Cruzeiro, atribui-se o intervalo de tempo de resposta desejado a este estado.

Tabela 1. Valores de tempo de resposta e variância para o PRE (em segundos).

	Sobra	**Cruzeiro**	**Sob-Pressão**	**Estresse**	**Inesperado**
Tempo	< 2	$[2, 3]$	$]3, 8]$	$]8, 15]$	> 15
Variância	< 2	< 2	$[2, 5]$	$[5, 10]$	> 10
Operação - CPU	-1	0	+1	+1	+2

A escolha destes valores deve ser definida por um especialista da carga do sistema, que tenha os objetivos do provisionamento bem especificados para que não ocorram operações desnecessárias e o sistema apresente o resultado esperado.

Para o método TR foi definido que se a média do tempo de resposta coletado dentro da janela de tamanho N ficar acima do intervalo determinado, adiciona-se uma vCPU à maquina virtual e abaixo do intervalo, decrementa-se uma. Para o método de UCPU, foram definidos que o uso de CPU deve manter-se entre o intervalo determinado. Caso sejam coletados valores acima do intervalo, adiciona-se uma vCPU e, abaixo do intervalo, reduz-se o número em uma unidade. Após a reconfiguração dos recursos, o sistema de controle da elasticidade fica desabilitado por 10 segundos até a próxima ação para que a propagação da alteração dos recursos seja detectada sistema.

4.2.1. Ambiente Computacional

Para a realização dos testes utilizou-se duas máquinas. A primeira é uma estação de trabalho com processador Intel i5 com 4 núcleos físicos e 4 GB de memória e é utilizada para gerar a carga de trabalho por meio da submissão de consultas ao SGBD. A segunda máquina é um servidor com três processadores AMD Opteron 6136 (totalizando 24 núcleos), 96 GB de memória, dois discos rígidos de 1 TB, no qual as máquinas virtuais são executadas e o PRE está instalado. Todas as máquinas, físicas e virtuais, utilizam a distribuição Ubuntu-Server 14.04.4, com *Kernel* 3.13.0-77 e o servidor está configurado com o hipervisor XEN versão 4.4.2[4].

Desenvolveu-se também um protótipo do PRE em Java compatível com qualquer SGBD relacional que possibilite acesso via JDBC. Neste conjunto de testes, o SGDB utilizado é o PostgreSQL 9.5[5]. Para gerar as cargas de trabalho do SGBD, desenvolveu-se um programa que simula a conexão de múltiplos clientes, sendo que o número de conexões segue uma função especificada, e cada cliente é responsável por conectar, enviar a consulta, aguardar um retorno e encerrar a conexão.

Conforme ilustrado na Figura 3, o ambiente foi organizado da seguinte forma: O servidor executa uma MV dedicada para o SGBD com a quantidade de vCPUs que pode variar entre 1 e 16 vCPUs, 40 GB de memória e 250 GB de espaço em disco. Pelo fato do SGBD usado ser orientado a disco, um dos discos rígidos do servidor foi dedicado para a MV, para que não ocorram interferências. O PRE e os demais métodos são executados diretamente no servidor físico para não causar ruído ao SGDB.

[4]www.xenproject.org/

[5]www.postgresql.org/

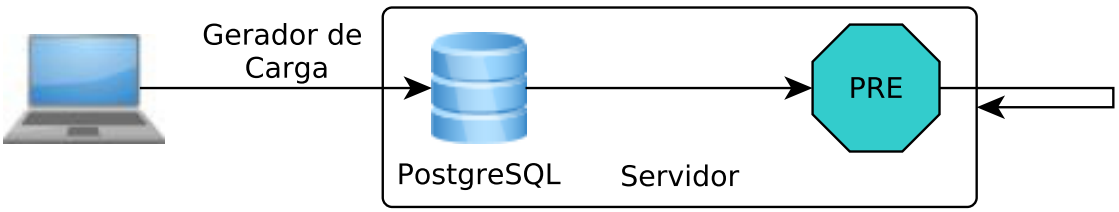


Figura 3. Configuração do ambiente de execução dos experimentos.

Considerando que o PostgreSQL tem seu desempenho dependente de parâmetros internos e as configurações padrão não estarem otimizadas para a quantidade de memória disponível, foi necessário alterar alguns parâmetros para se aproximar de um ambiente real e melhorar seu desempenho. Na Tabela 2 são listados os parâmetros alterados com seus valores e quais as mudanças geradas no desempenho no SGBD. Os valores foram ajustados conforme indicações no manual do PostgreSQL[6].

Tabela 2. Parâmetros alterados no PostgreSQL para melhorias no desempenho.

Parâmetro	**Valor**	**Impacto**
$checkpoint\_completion\_target$	0.9	Porcentagem máxima do intervalo de tempo entre *checkpoints* para escrita em disco.
$wal\_buffers$	8 MB	Quantidade de memória compartilhada não persistida no disco.
$effective\_cache\_size$	28 GB	Espaço disponível para *cache* em disco para uma consulta.
$work\_mem$	96 MB	Espaço em memória disponível para operações internas de ordenação, antes da escrita em arquivos temporários.
$shared\_buffers$	12 GB	Espaço em memória dedicado a *buffers*.

4.2.2. Base de dados e Geração de Carga

Para gerar a carga de trabalho no sistema, utilizou-se os dados e consultas do *benchmark* TPC-H. Este foi criado pela TPC[7] e foi desenvolvido para sistemas de suporte a decisão, composta por consultas de alta complexidade, que examinam grandes volumes de dados e respondem a perguntas críticas do mundo corporativo. Não é o objetivo desde trabalho avaliar o desempenho direto do SGBD utilizando um *benchmark* e suas métricas, mas utilizar as consultas como forma de consumir os recursos computacionais.

Os dados populados e as consultas utilizadas foram geradas utilizando *software* disponibilizado pela TPC, de modo que sejam geradas altas demandas de recursos durante a execução de cada consulta. Foram selecionadas apenas consultas que não alteram os valores da base, pois isso em alto volume pode intensificar o uso de disco, prejudicando o desempenho e a avaliação do provisionamento elástico de vCPUs.

[6]https://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server
[7]http://www.tpc.org/

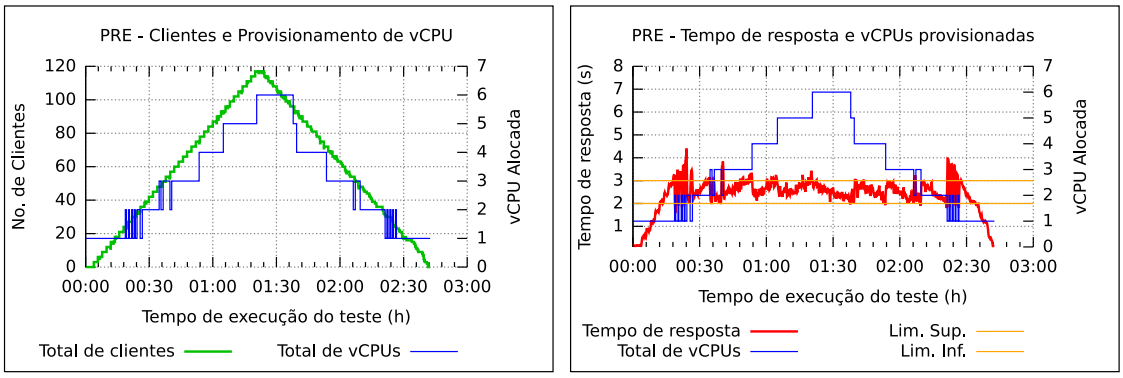
A geração de carga é feita utilizando múltiplos fluxos de execução, de mesma origem do Experimento 1, entretanto oriundos de uma única estação de trabalho, como indicado na Figura 3. Cada fluxo de execução é responsável por: (i) abrir uma conexão com o SGBD; (ii) escolher aleatoriamente uma consulta do TPC-H; (iii) enviar a consulta ao SGBD e aguardar o resultado; (iv) encerrar a conexão. Desta forma, cada fluxo se torna um cliente enviando consultas ao SGBD e acaba gerando carga.

A quantidade de clientes é variável ao longo do tempo, separada em dois momentos: crescimento e declínio. Em cada momento, o número de clientes executando é separado por pequenas etapas, em que a atualização do total necessário é feita a cada 2 minutos. Durante a fase de crescimento, a cada dois minutos a quantidade de clientes é aumentada, e durante a fase de declínio, a quantidade de clientes é reduzida a cada dois minutos. O número máximo de clientes atribuídos neste experimento é 120.

4.2.3. Resultados

Esta seção apresenta os resultados obtidos com o PRE e compara-os com aqueles obtidos com os métodos TR e UCPU.

A Figura 4 mostra os resultados obtidos com o provisionamento utilizando o método PRE. Na Figura 4(a) observa-se a alocação de vCPUs conforme a variação no número de clientes, que alcança o máximo de 6 unidades alocadas a partir de 100 clientes conectados. Nota-se, nos intervalos [0:00–0:30] e [2:30–3:00] a ocorrência de alocações e desalocações intermitente de vCPUs causadas pela oscilação do tempo de resposta, fazendo com que nesses intervalos uma única vCPU fosse insuficiente e a alocação de duas fosse excessiva. No restante da execução, o provisionamento de recursos é coerente e realizado de acordo com o número de clientes.

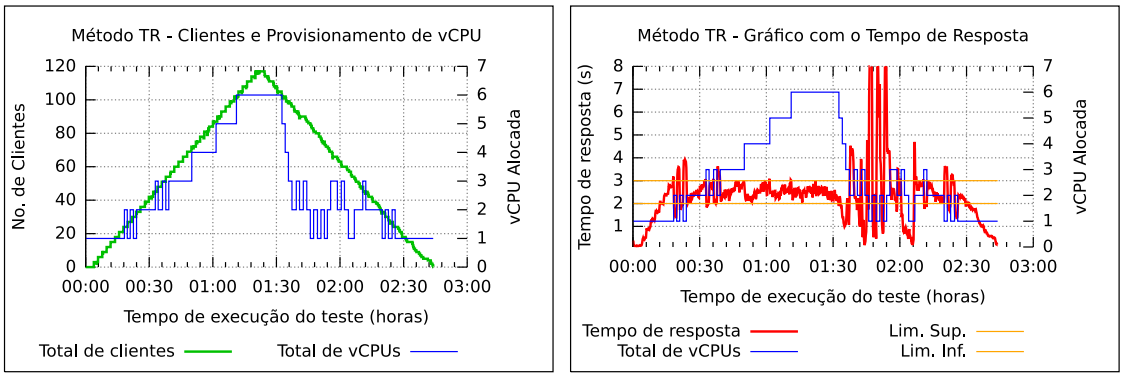


(a) Número de clientes (b) Tempo de resposta

Figura 4. PRE: (a) provisionamento de vCPUs e (b) tempo de resposta.

Na Figura 4(b) apresenta-se o tempo de resposta obtido durante toda a execução do teste. As linhas laranjas, indicam o limiar superior e inferior do estado de Cruzeiro, indicando que o SGBD está no nível de estresse ideal, com recursos apropriados para a demanda. Observa-se nos resultados que na maior parte da execução o tempo de resposta se encontra dentro do esperado, exceto nos intervalos onde há a alocação e desalocação intermitente, conforme comentado anteriormente.

A Figura 5 mostra os resultados obtidos com o método TR. No início e final da execução, pode-se observar um comportamento semelhante ao apresentado pelo método PRE, uma vez que utiliza-se os mesmos limiares. No entanto, quando o número de clientes começa a cair, a partir do instante 1:30 do teste, o número de vCPUs foi reduzido pela metade, diferenciando-se do comportamento do método PRE e impactando diretamente no tempo de resposta, conforme apresentado na Figura 5(b).



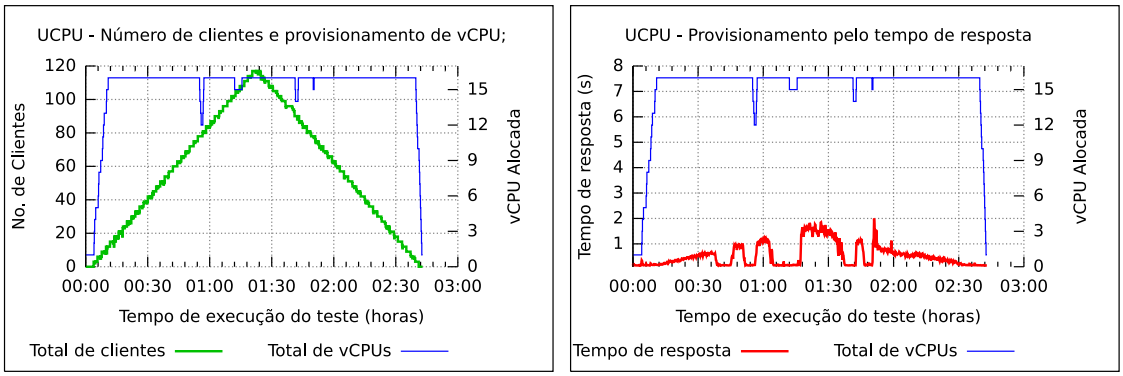
(a) Número de clientes (b) Tempo de resposta

Figura 5. TR: (a) provisionamento de vCPUs e (b) tempo de resposta.

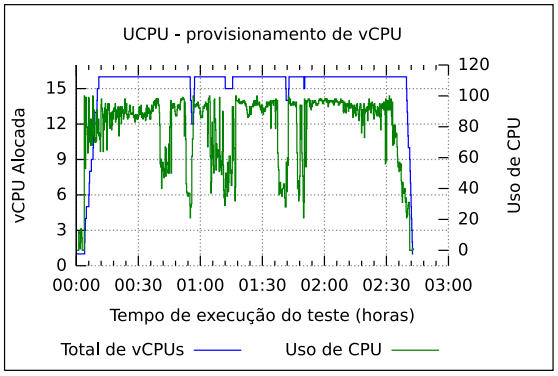
Com a redução do número de vCPUs aumenta-se o tempo de resposta, que fica acima do limite superior estabelecido, prejudicando o desempenho do SGBD. Nota-se também que não há uma tentativa eficiente de recuperação de desempenho por este método.

A Figura 6 apresenta os resultados da abordagem UCPU. Diferentemente dos resultados apresentados anteriormente, no método UCPU as 16 vCPUs são alocadas logo no início da execução do teste (Figura 6(a)), uma vez que o uso de CPU permanece acima dos limiares estabelecidos em todos os testes efetuados, como mostra a Figura 6(c). Como consequência, esta abordagem apresentou os melhores resultados de tempo de resposta (Figura 6(b)), no entanto, mostrou-se ineficiente no objetivo de alocar apenas os recursos necessários, para que o tempo de resposta não seja apenas abaixo do limite superior, mas próximo a ele.

Observa-se ainda na Figura 6(c) que em alguns instantes ocorreu a queda na taxa de uso de CPU e consequentemente uma redução do número de processadores, entretanto, na sequência voltou-se ao patamar anterior. As pequenas oscilações observadas, são devido aos processos estarem aguardando operações de entrada/saída, reduzindo a utilização de vCPU.



(a) Número de clientes (b) Tempo de resposta



(c) Uso de CPU

Figura 6. UCPU: (a) provisionamento de vCPUs, (b) tempo de resposta e (c) uso de CPU.

De modo a sintetizar os resultados obtidos, a Tabela 3 apresenta os tempos de resposta obtidos por cada um dos três métodos comparados. Como dito anteriormente, o objetivo deste experimento é manter o tempo de resposta entre 2 e 3 segundos, consumindo o menor número de vCPUs. Assim, a abordagem baseada nos níveis de estresse mostrou ser uma opção balanceada e apresentou resultados satisfatórios, alcançando tempos de resposta melhores que o método TR e com consumo de recursos inferior ao método UCPU.

Tabela 3. Tempos de resposta: PRE $\times$ TR $\times$ UCPU.

	Tempo de Resposta (s)			
	Média	**Mínimo**	**Máximo**	**Variância**
PRE	2,386	0,140	4,404	0,509
TR	2,279	0,141	9,959	1,508
UCPU	0,548	0,144	1,995	0,188

5. Considerações Finais

Este trabalho teve como objetivo apresentar uma nova abordagem para a exploração da elasticidade em banco de dados, sob a motivação de que que os mecanismos presentes no estado-da-arte apresentam ainda algumas limitações.

Nesta proposta, a carga no sistema é estimada pelo tempo de resposta do SGBD e classificada em níveis de estresse diferentes utilizando a máquina de estados. Com a definição do nível de estresse e informações sobre os recursos, ações de elasticidade são podem ser realizadas para manter o desempenho do banco de dados dentro de uma determinada especificação.

De fato, os resultados apresentados comprovam que o controle de elasticidade baseado nos níveis de estresse é uma alternativa interessante quando comparado a outros métodos que empregam métricas mais simples, como tempo de resposta e uso de CPU.

A abordagem proposta neste trabalho pode ser empregada em Banco de Dados como Serviço (DBaaS). Do ponto de vista do provedor do serviço, a elasticidade garante um melhor uso dos recursos de computação, fornecendo economia de escala e permitindo que mais usuários possam ser atendidos simultaneamente, uma vez que os recursos liberados por um usuário podem instantaneamente ser alocados por outro. Da perspectiva do usuário, a elasticidade garante que o desempenho do banco de dados esteja sempre conforme o especificado no acordo de nível de serviço, independentemente da carga de trabalho recebida.

Referências

Curino, C., Jones, E., Popa, R. A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. (2011). Relational Cloud: A Database Service for the Cloud. In *5th Biennial Conference on Innovative Data Systems Research*, Asilomar, CA.

Das, S., Agrawal, D., and El Abbadi, A. (2009). Elastras: An elastic transactional data store in the cloud. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, Berkeley, CA, USA. USENIX Association.

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.

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, 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.

Galante, G. and Bona, L. C. E. (2012). A survey on cloud computing elasticity. In *Proceedings of the International Workshop on Clouds and eScience Applications Management*, CloudAM'12, pages 263–270. IEEE.

Herbst, N. R., Kounev, S., and Reussner, R. (2013). Elasticity in cloud computing: What it is, and what it is not. In *Proceedings of the 10th International Conference on Autonomic Computing*, ICAC'13, pages 23–27. USENIX.

Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. *J. Grid Comput.*, 12(4):559–592.

Meira, J. A., Almeida, E. C., and Traon, Y. L. (2014). A State Machine for Database Non-functional Testing. In *IDEAS 2014 : 18th International Database Engineering & Applications Symposium*, Porto, Portugal.

Minhas, U. F., Liu, R., Aboulnaga, A., Salem, K., Ng, J., and Robertson, S. (2012). Elastic scale-out for partition-based database systems. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering Workshops*, ICDEW '12, pages 281–288, Washington, DC, USA. IEEE Computer Society.

Sakr, S., Zhao, L., Wada, H., and Liu, A. (2011). CloudDB AutoAdmin: Towards a truly elastic cloud-based data store. In Ian Foster, Louise Moser, Jia Zhang, editor, *Proceedings of the IEEE 9th International Conference on Web Services (ICWS '11)*, pages 732–733, Washington DC, USA. IEEE Computer Society.

Serafini, M., Mansour, E., Aboulnaga, A., Salem, K., Rafiq, T., and Minhas, U. F. (2014). Accordion: Elastic scalability for database systems supporting distributed transactions. *Proc. VLDB Endow.*, 7(12):1035–1046.