Detecção online de estados de um sistema gerenciador de Banco de Dados

Cristian Weiland , Eduardo Maia Machado

[1]Departamento de Informática - Universidade Federal do Paraná (UFPR)

cw14@inf.ufpr.br, emm14@inf.ufpr.br

Resumo. *Este trabalho tem como objetivo apresentar as principais características do MoDaST (Model-based Database Stress Testing), com ênfase em uma máquina de estados que descreve os possíveis cenários de desempenho do SGBD (Warm-up, Steady, Under-Pressure, Stress e Trashing). Além disso, explica a motivação e embasamentos para utilizar esta máquina de estados em uma ferramenta de monitoramento online de desempenho de SGBD. Descreve também a arquitetura e funcionamento da ferramenta proposta, explicando como ela lida com as questões de precisão dos resultados e desempenho dos cálculos e coleta de dados.*

1. Introdução

Disponibilidade [Ji-Woong Chang 2005] e recuperabilidade [Raghu Ramakrishnan 2007] são dois requisitos não funcionais relevantes quando trata-se do estudo de Sistemas Gerenciadores de Bancos de Dados (SGBD). A falta de algum destes implica na diminuição da confiabilidade do SGBD, o que torna vitais as técnicas de monitoramento de carga de um banco de dados. Essas técnicas permitem ao analista tomar certas decisões, como por exemplo atrasar ou bloquear temporariamente requisições, ou até mesmo forçar o encerramento de algumas com custo computacional muito elevado, para garantir a confiabilidade de seu SGBD. Para isso, ser capaz de determinar o estado de *stress* a partir da análise de carga do SGBD é crucial.

2. Fundamentação

O *MoDaST* (*Model-based Approach to Database Stress Testing*, ou "Uma abordagem baseada em modelos para testes de *stress* em SGBDs") [Meira et al. 2005] é uma proposta para analisar o desempenho e os limites de SGBDs, especialmente os que atingem grande número de conexões e/ou altas cargas de requisições de transações. A ferramenta proposta no *MoDaST* é capaz de simular altas cargas e efetuar a análise com os dados obtidos, enquadrando cada momento em um dentre 5 estados pré-definidos, que formam uma máquina de estados (Figura 1). Ambos estados e máquina de estados foram propostos no *MoDaST*. São os estados: *Warm-up (s1)*, *Steady (s2)*, *Under-Pressure (s3)*, *Stress (s4)* e *Trashing (s5)*.

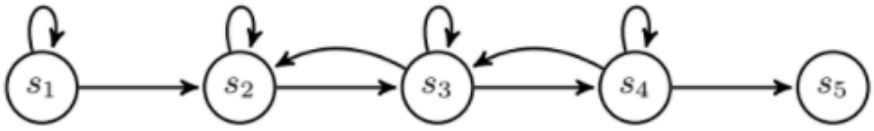


Figura 1: Máquina de Estados do MoDaST [Meira et al. 2005]

Variável	Valor (PostgreSQL)	Objetivo
t_w	0.1	Delimitar estados *warm-up* e *steady*
t_{st}	0.9	Delimitar estados *steady* e *under pressure*
t_{sh}	0.1 * tp_{sup}	Delimitar estados *under pressure* e *stress*
t_h	1	Delimitar estados *stress* e *trashing*

Tabela 1: Variáveis de Limites

	Estado Alvo				
Estados	**s1**	**s2**	**s3**	**s4**	**s5**
s1	$\neg(\Delta > t_w)$	$\Delta < t_w$	-	-	-
s2	-	$\delta > t_s$	$\neg(\delta > t_s)$	-	-
s3	-	$\delta > t_s$	$\neg(\delta > t_s)$	$\Delta > t_{st}$	-
s4	-	-	$\neg(\Delta > t_{st})$	$\Delta > t_{st}$	$\varphi < t_{th}$
s5	-	-	-	-	$\varphi = 0$

Tabela 2: Tabela de transição de estados [Meira et al. 2005]

O *MoDaST* propõe uma forma de determinar o estado do SGBD através do número de requisições enviadas e respondidas, que se utiliza de algumas constantes de limite pré-definidas (*threshold values*) que variam de acordo com o SGBD e a arquitetura utilizados. As constantes de limite são quatro: t_w, t_{st}, t_{sh} e t_h, e seus valores e objetivos podem ser observados através da Tabela 1.

Além das constantes, são definidas três variáveis de desempenho, cujo valor é obtido usando o número de requisições enviadas e respondidas por segundo. As definições de como são calculados os valores destas variáveis podem ser encontradas no *MoDaST* [Meira et al. 2005]. As variáveis e o que representam são:

- *Performance Variation* (Variação de Desempenho): denotada por Δ, simboliza a dispersão do número de requisições tratadas por segundo.
- *Transaction Efficiency* (Eficiência de Transações): denotado por δ, é a proporção entre requisições respondidas por segundo e o número de requisições enviadas por segundo, considerando dados cujo período se enquadra em uma janela de tamanho pré-definido.
- *Performance Trend* (Tendência de Desempenho): denotado por φ, representa o instante em que se estima que o número de requisições respondidas seja 0.

A Tabela 2 [Meira et al. 2005] indica para qual estado o SGBD está se encaminhando, baseando-se no estado atual (*s1*, *s2*, *s3*, *s4* ou *s5*), nas constantes de limite (t_w, t_{st}, t_{sh} e t_h) e nas variáveis de desempenho (Δ, δ e φ).

Os 5 estados definidos são:

1. ***Warm-Up***: Inicialização de serviços internos e memórias cache. O desempenho não é estável, o que inviabiliza uma análise concreta.
2. ***Steady***: Com a inicialização finalizada, o desempenho fica estável, e neste estado a maioria das transações requisitadas são tratadas à tempo. Não possui nenhuma transição para voltar ao *Warm-up*.
3. ***Under Pressure***: O SGBD está no limite do desempenho. Consegue tratar um número de transações à tempo similar ao *Steady*, mas algumas transações não o são. Pode voltar ao estado anterior sem interferências externas, mas elas podem ser necessárias (por exemplo limitando o número de transações requisitadas).
4. ***Stress***: O SGBD ultrapassou o limite de desempenho. Isto significa que o número de requisições tratadas por segundo começa a diminuir. O SGBD está altamente vulnerável e pode facilmente falhar se nenhuma ajuda externa for oferecida. Neste estado outras atitudes também devem ser consideradas pelo analista, tais como usar réplicas do SGBD, adicionar mais máquinas ao ambiente de processamento ou até mesmo encerrar transações muito longas.
5. ***Trashing***: O SGBD está usando um alto número de recursos computacionais para um número mínimo de transações e começa a ter contenção de recursos. Nenhuma transação nova é atendida. É impossível voltar a outros estados, mesmo com intervenções externas, exceto através da reinicialização completa do SGBD.

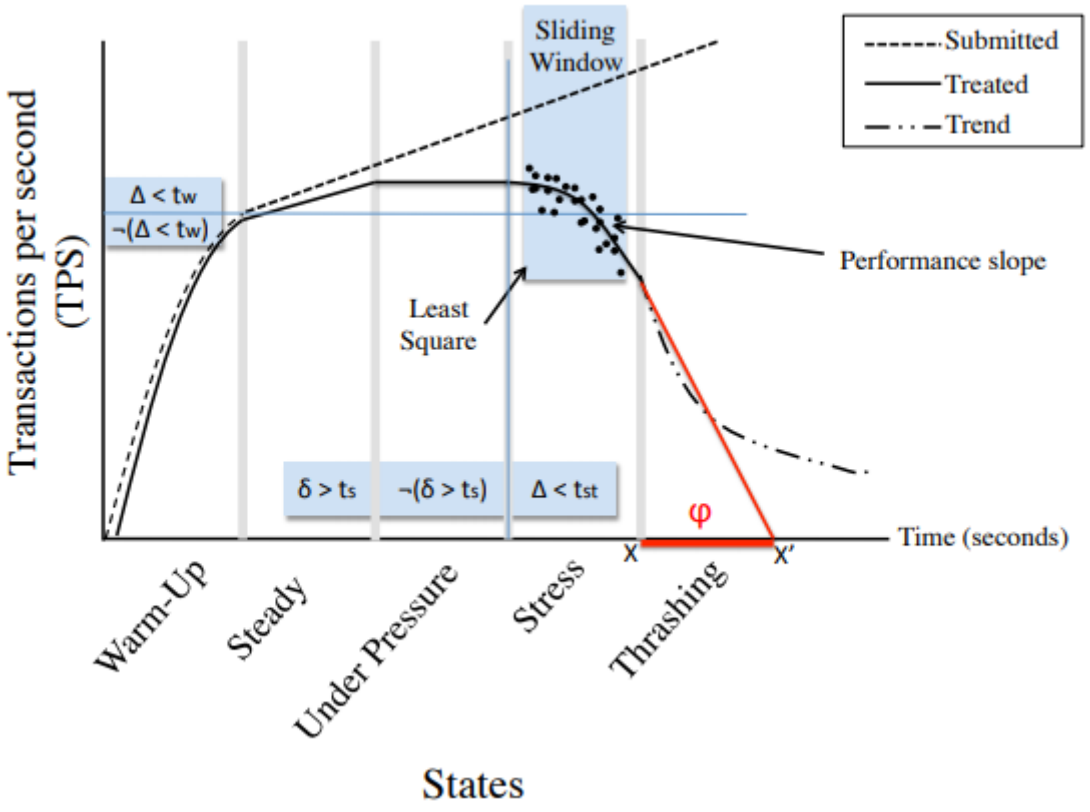


Figura 2: Relação de Transações por Segundo x Estados do SGBD [Meira et al. 2005]

A Figura 2 mostra a relação esperada entre estados e o número de requisições enviadas (*submitted*) e respondidas (*trend*). Além disso, ela indica as condições para que o SGBD seja enquadrado em cada estado, através das variáveis de desempenho. A figura também mostra onde se enquadra o cálculo da variável Tendência de Desempenho, feito através do método de quadrados mínimos (*Least Square*) e representado pela linha vermelha, usando x como instante atual e x' como o instante em que o número de transações por segundo tende a 0.

O *MoDaST* só efetua análises sobre o SGBD após o encerramento de sua execução, isto é, utiliza um modelo *post mortem*, que estenderemos à uma análise *online*.

3. Motivação

O *MoDaST* cumpre sua proposta: simular altas cargas, analisar a execução do SGBD durante estas cargas e identificar os estados nos quais o SGBD se encontra durante a execução. Isto lhe permite identificar *bugs*, compreender o comportamento do SGBD e seus limites de desempenho. Além disto, esta análise auxilia na tomada de decisões do analista de banco de dados, especialmente no que se refere à escolha de *hardware* e configurações do SGBD, que podem ser feitas mais precisamente utilizando-se de informações que o *MoDaST* gera a respeito das prováveis cargas a que o SGBD será submetido. Porém, por ser um modelo de execução *post mortem*, o *MoDaST* não é adequado para monitorar a máquina que está executando o SGBD, já que isto foge à sua proposta.

A análise do *MoDaST*, baseada nestes estados do SGBD, garante um significado relevante aos dados coletados, o que não ocorre com muitas outras ferramentas, que analisam apenas dadoss dos recursos disponíveis no *hardware* de processamento diretamente, como memória, *CPU*, *IO* de disco, etc.

Por estes motivos optamos por utilizar esta definição de estados para uma análise *online*, na qual não é necessário esperar a execução do SGBD ser encerrada para calcular seu estado. Desta forma, é possível trazer mais significado para a análise *online* de dados e mais urgência para o modelo de análise *post mortem* do *MoDaST*.

4. Proposta de Solução

O *OSD* (*Online State Detector*) foi desenvolvido como uma ferramenta de monitoramento, cujo objetivo primordial é fazer a análise de estados *online*, ou seja, conforme os dados são coletados eles são analisados. Isto permite determinar o estado do SGBD no instante atual, o que garantirá ao analista a chance de tomar atitudes imediatas que auxiliem na preservação da disponibilidade do SGBD. Sua arquitetura está ilustrada na Figura 3.

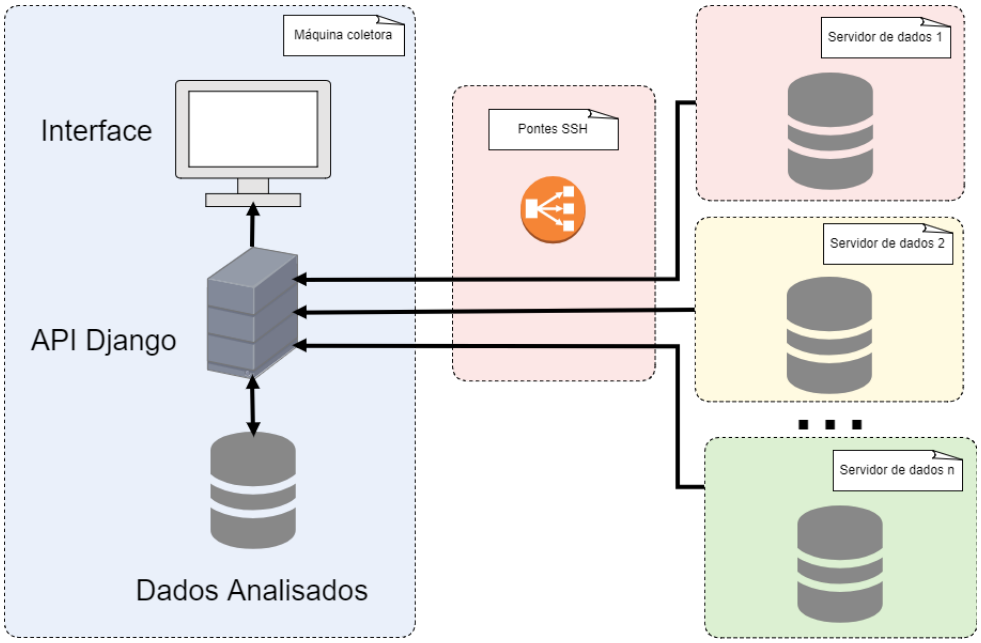


Figura 3: Modelo do *OSD*. Nele, múltiplos SGBDs são conectados, através de pontes *SSH*, a uma máquina coletora, que possui um banco de dados próprio para armazenar dados coletados. A máquina coletora também analisa os dados e comunica-se com uma interface *web*.

A ferramenta proposta não tem nenhuma relação direta com as máquinas que geram requisições para o servidor (neste caso, as máquinas clientes). Isto significa que, diferentemente do *MoDaST*, não é possível coletar dados referentes ao número de requisições realizadas e respondidas pelo cliente. Portanto, isto será feito pela máquina do servidor, no seguinte fluxo:

- A máquina servidora armazena, através dos *logs* do servidor *proxy HTTP*, a duração e status de cada requisição recebida;
- A máquina coletora acessa a máquina servidora (através de uma ponte *SSH*) e coleta, através do mesmo *log*, a duração e o status de cada requisição;

Usando estes dados, a máquina coletora consegue inferir quantas requisições foram realizadas e tratadas a cada segundo e, através disto, executar os cálculos do *MoDaST*. Abordaremos mais a respeito do impacto que isto causa nos cálculos dos estados do SGBD nas subseções 4.3 e 5.5.

Como o modelo é *online*, o início da coleta e da análise dos dados não coincidem (em sua maior parte) com a inicialização do SGBD. Isto significa que o desempenho estará estável na maioria das inicializações, com serviços inicializados e *caches* já preenchidas. Por isto, desconsideramos o estado *Warm-Up* como estado inicial, e iniciamos as análises a partir do estado *Steady*.

4.1. Qual a quantidade de dados necessária?

Esta proposta por si só gera uma pergunta crucial: quanto tempo do histórico de processamento de transações deve ser analisado para obter um resultado suficientemente preciso sobre o estado atual do SGBD? Isto deve levar em consideração também a quantidade de

dados processados, pois ela deve ser pequena o bastante para permitir que a análise seja feita em tempo hábil de notificar os analistas de eventuais mudanças de estados, antes que a recuperação não seja mais possível.

Isto não era tão preocupante para o *MoDaST*, pois seu escopo não incluía resultados da análise calculados imediatamente após a coleta. Desta forma, era possível analisar qualquer tamanho de histórico, inclusive ele completamente, sem maiores preocupações.

Um histórico maior pode garantir um resultado menos volátil e, desta forma, adicionar mais garantia à análise quando ela indica problemas de desempenho no banco. Por outro lado, isto exigirá mais tempo de processamento dos dados, o que poderia gerar atrasos na emissão de resultados, que por sua vez pode ser fatal para o SGBD [Ji Zhu 2002]. Além disso, a análise levaria mais tempo para começar a indicar que o SGBD se encontra em estado precário, pelo fato de considerar muitos dados anteriores que mostravam uma execução saudável.

Por outro lado, selecionar um período de dados muito curto pode fazer com que os resultados sejam imprecisos e gerem outros problemas. Um desses problemas seria dar uma falsa sensação de segurança, fazendo o analista acreditar que o SGBD está com uma carga razoável em momentos críticos. Outro problema seria indicar precariedade quando ainda não se possui certeza, o que poderia levar o analista a tomar decisões drásticas em momentos que não há necessidade (como por exemplo forçar o encerramento de grandes transações em execução enquanto o SGBD está saudável).

Levando isto em consideração, surge a proposta de efetuar a análise através de uma janela deslizante, isto é, uma espécie de tabela na qual cada posição corresponde a um instante, como mostrado na Figura 4. Esta tabela tem tamanho fixo n, e vai avançando junto com o tempo. Os dados utilizados para a detecção do estado são os que estão dentro da janela. Neste trabalho efetuamos análises empíricas baseadas em resultados gerados por testes automatizados para determinar tamanhos adequados de janelas, que preservem a precisão dos resultados bem como um tempo viável de processamento.

Figura 4: Exemplo de funcionamento de uma janela deslizante ao longo do tempo

4.2. Como fazer o monitoramento sem interferir no desempenho do SGBD?

Um dos problemas que surge na análise de estados *online* é como não interferir no desempenho do SGBD durante o monitoramento. Em outras palavras, como processar os dados sem afetar significativamente o tempo de resposta do SGBD, já que a análise necessita de armazenamento e processamento de dados?

Nossa proposta é utilizar uma máquina exclusiva para a inferência de estados. Isto ainda adulterará o desempenho do SGBD, mas em uma escala muito menor, que pode vir a ser desprezível (Descrito na seção 5.4). Para isto, esta máquina exclusiva, a que chamaremos de Máquina de Coleta e Análise, irá coletar, através de uma ponte *SSH*, os dados necessários para efetuar a análise proposta no *MoDaST*

Na Figura 3 é mostrado um esquema conceitual de como funciona a proposta.

Um exemplo de uso seria:

- O administrador do sistema adiciona vários servidores de dados no *OSD*, que está sendo executado na máquina de análise.
- A máquina de coleta e análise cria uma ponte *SSH* entre ela e cada servidor de dados e permite ao administrador ligar ou desligar a coleta de dados em cada servidor.
- Todos os dados são guardados em uma base de dados na máquina de análise.
- A partir do momento em que a coleta começa, a máquina de análise já começa a realizar seus cálculos para mostrar o estado de cada servidor de dados através de uma interface *web*.

4.3. Coletar dados no servidor gera impacto na acurácia dos resultados?

Esta pergunta se torna pertinente pelo fato de o *MoDaST* coletar dados de número de requisições enviadas e recebidas nas máquinas cliente, enquanto o *OSD* coleta diretamente no servidor. Portanto, é natural haver uma divergência entre os valores, devido

principalmente ao tempo de latência do tráfego do pacote *HTTP* entre as máquinas clientes e servidor.

4.4. A utilização de dados da máquina podem ajudar?

Efetuaremos também uma análise de alguns dados da máquina que podem auxiliar na determinação do estado do SGBD, tais como:

- Uso de *CPU*;
- Ocupação da memória principal;
- Uso de disco rígido;
- I/O;
- Número de processos;
- Utilização de *Swap*;
- Número de interrupções por chamadas de sistema;

A partir da coleta destes dados nós determinamos se existe uma correlação direta entre estes e o estado do SGBD através de um cálculo de correlação simples, entre cada tipo de dado na máquina e os estados através da fórmula mostrada na Figura 5:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\left[\sum_{i=1}^{n}(x_i - \overline{x})^2\right]\left[\sum_{i=1}^{n}(y_i - \overline{y})^2\right]}}$$

Figura 5: Fórmula de cálculo de correlação

Nesta fórmula, n equivale a um número finito de momentos, x a um vetor de dados da máquina no decorrer de n momentos, $\overline{x}$ à média dos valores de x, x_i o i-ésimo elemento de x, y a um vetor de estados no decorrer de n momentos, $\overline{y}$ a média dos valores de y, y_i o i-ésimo elemento de y e r a correlação entre os vetores x e y.

Por exemplo, se estivéssemos calculando a correlação de quantidade de memória usada como *buffer* com os estados, faríamos da seguinte forma:

- Obteríamos uma sequência de valores (cada valor sendo a quantidade de memória como *buffer* naquele instante), representará o vetor x;
- Coletaríamos outra sequência de valores, com cada valor sendo um número que indique um estado de desempenho do SGBD, representará o vetor y;
- Obtemos $\overline{x}$ e $\overline{y}$ a partir da média das sequências de dados;
- Aplicamos a fórmula e obtemos a correlação;

Caso o resultado da correlação entre um determinado dado da máquina e os estados seja muito próximo de zero, significa que provavelmente esta variável não tem correlação alta com a troca de estados do SGBD.

5. Experimentos

Pelo fato de o escopo do trabalho ser de cunho muito prático decidimos realizar testes empíricos para responder às questões levantadas na seção 4 e validar a viabilidade da ferramenta.

5.1. Recursos

Para a realização dos experimentos, foram usadas 3 máquinas diferentes, dentre elas uma máquina para simulação de um SGBD, uma máquina cliente e uma máquina de coleta e análise.

Além disto, implementamos a ferramenta proposta no *MoDaST*, além do *OSD*, ambos usando *Python v3.5*. Para gerar números aleatórios utilizamos a função *random* da biblioteca *Random*, integrada ao *Python*.

5.1.1. Máquina de Simulação de SGBD

Esta máquina é uma instância do *Google Cloud Platform*, que foi utilizada para simular um servidor real, cujas configurações são:

- Processador Intel(R) Xeon(R) CPU @ 2.30GHz
- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 46080K
- CPUs: 1
- Memória RAM: 1GB

Decidimos utilizar uma máquina com recursos computacionais escassos como servidor para tornar mais fácil a realização dos testes de *stress* utilizando apenas uma máquina cliente.

O SGBD escolhido para a realização dos testes foi o PostgreSQL, nesta base de dados foi inserida uma tabela com 10 milhões de entradas cujos atributos são 2: valor (*float*) e nome (*string*). Esta tabela foi ordenada pelo valor para tornar as buscas em *range* mais eficientes.

5.1.2. Máquina Cliente

A máquina cliente possui as seguintes configurações:

- Processador Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz
- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 3072K
- CPU(s): 4
- RAM: 8GiB System Memory (2x 4GiB SODIMM Synchronous 1600 MHz)

Esta máquina foi utilizada para submeter um grande número de requisições à máquina de simulação do SGBD.

5.1.3. Máquina de Coleta e Análise

A máquina de coleta e análise possui as seguintes configurações:

- Processador Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz
- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 3072K
- CPUs: 4
- Memória RAM: 4GiB DDR3 Synchronous 1333 MHz (0.8 ns)

Esta máquina foi utilizada para fazer testes usando a ferramenta proposta, o *OSD*. A implementação da ferramenta proposta pelo *MoDaST* faz parte do *OSD*.

5.2. Análise do tempo de processamento de acordo com o tamanho da janela

Para a análise de tempo de processamento, utilizamos testes com dados coletados por 32.768 segundos na máquina de coleta e análise. Calculamos o tempo necessário para processar cada iteração que era potência de 2 e, a partir de uma média entre 5 execuções diferentes, traçamos uma reta (Figura 6) para visualizar quanto tempo era necessário para o cálculo de cada iteração.

Para garantir que os dados sejam tratados em tempo hábil, dado que os dados são coletados a cada segundo, é necessário que cada iteração seja calculada em menos de um segundo.

A partir da reta, estima-se que é possível usar dados de mais de 13 dias sem ultrapassar um segundo de processamento por iteração. Portanto, se conseguirmos acurácia suficiente para efetuar análises com uma janela de tamanho menor que 1123200 (número de segundos em 13 dias), o desempenho não será um problema.

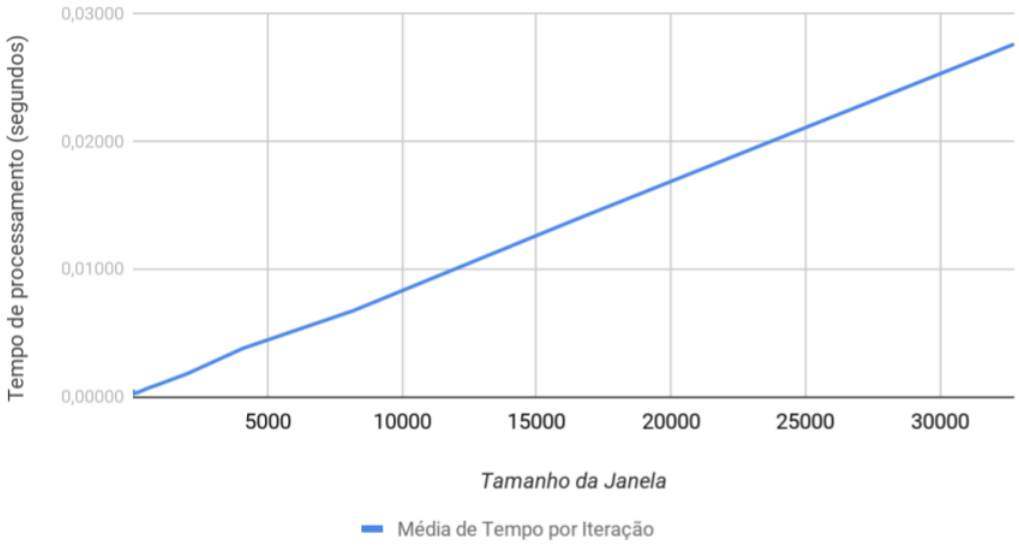


Figura 6: Tamanho da Janela x Tempo de Processamento

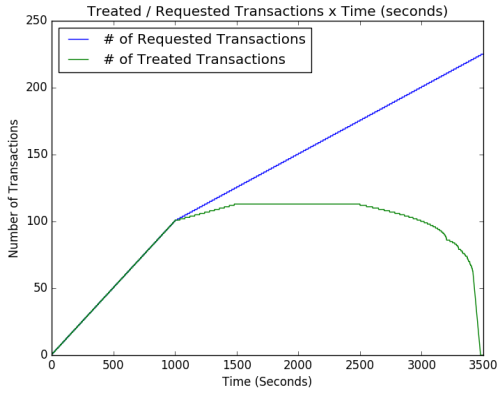
5.3. Análise da acurácia de detecção de estados de acordo com o tamanho da janela

Para efetuar a análise do tamanho da janela, simulamos dados que sabemos como enquadrar em cada estado, a partir de sua definição. Eles se assemelham à Figura 2.

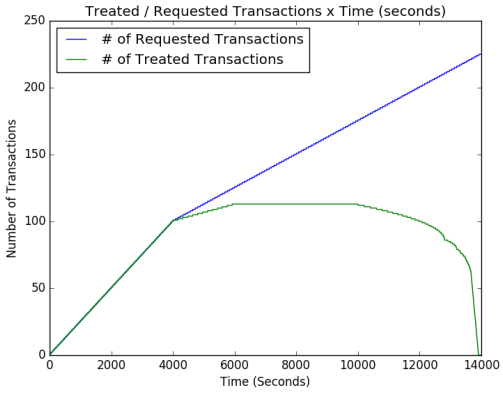
Simulamos dados de diversos tamanhos: 3500, 7000, 14000, 28000 e 56000. O tamanho dos dados simboliza quanto tempo de análise teria sido realizado, em segundos, isto é, o menor conjunto de dados dura pouco menos que uma hora, e o maior dura mais do que 15 horas.

Além destes gráficos, fizemos uma simulação com dados com muita variação, para verificar se é possível identificar um resultado volátil dependendo do tamanho da janela.

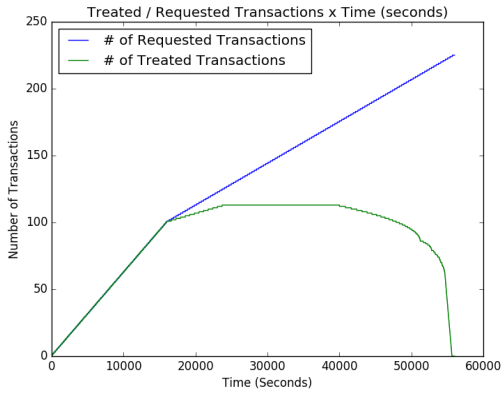
A Figura 7 mostra os conjuntos de dados, através do número de requisições feitas e requisições respondidas.



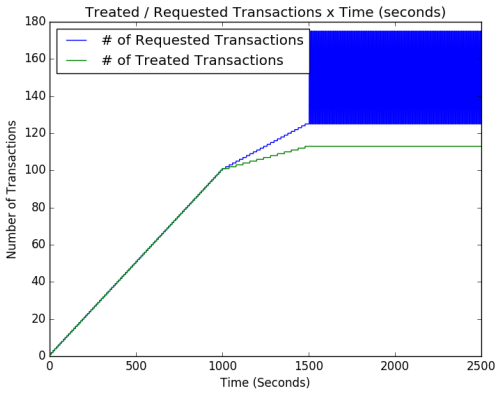
(a) Simulação de 3500 segundos



(b) Simulação de 14000 segundos



(c) Simulação de 56000 segundos



(d) Simulação Volátil

Figura 7: Gráficos do número de requisições tratadas e respondidas simulados

Utilizando estes dados, executamos uma implementação da ferramenta proposta no *MoDaST* na máquina de análise e coleta, e geramos gráficos para analisar os resultados obtidos.

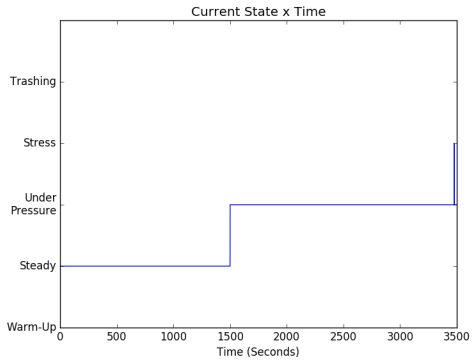
Os resultados consistem de gráficos utilizando diferentes tamanhos de janelas (selecionamos todas as potências de 2 que são menores que o número total de dados), para

cada um dos tamanhos previamente mostrados. Desta forma, para o conjunto de dados com 3500 instantes, os tamanhos de janela utilizados foram 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 e 2048. Por semelhante forma, para o conjunto de dados de 56000 instantes, foram utilizadas janelas de tamanho 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384 e 32768.

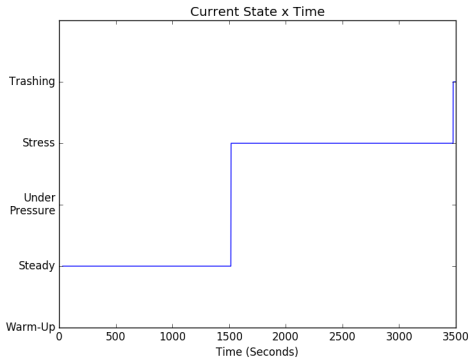
Para a simulação volátil, executamos testes apenas com tamanhos de janela 2, 4, 8, 16 e 32.

Para cada conjunto de dados, mostraremos gráficos relacionando o estado do SGBD obtido em relação ao instante.

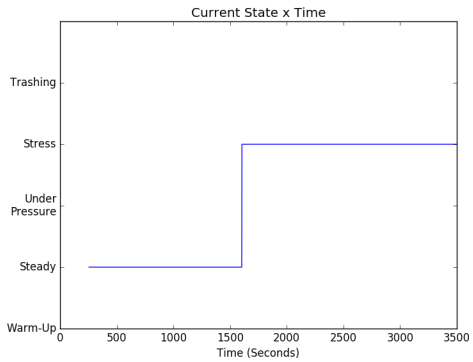
Como mostrar todos os gráficos poderia esconder informações relevantes por uma quantidade muito grande de dados, selecionamos, para cada conjunto de dados, os gráficos que consideramos mais relevantes, dentre os tamanhos 3500, 14000 e 56000 e a simulação volátil. Eles podem ser visualizados nas Figuras 8, 9, 10 e 11.



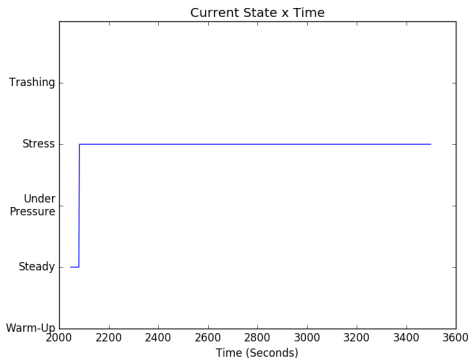
(a) Janela deslizante de tamanho 2



(b) Janela deslizante de tamanho 32

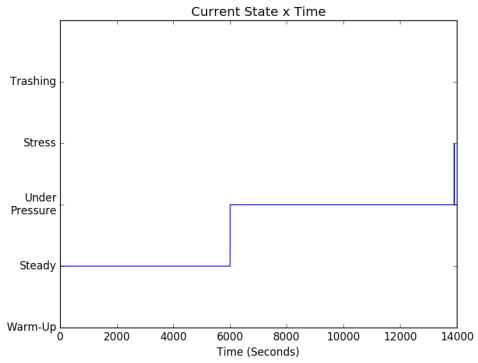


(c) Janela deslizante de tamanho 256

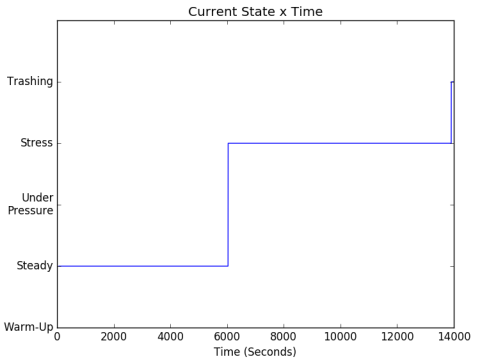


(d) Janela deslizante de tamanho 2048

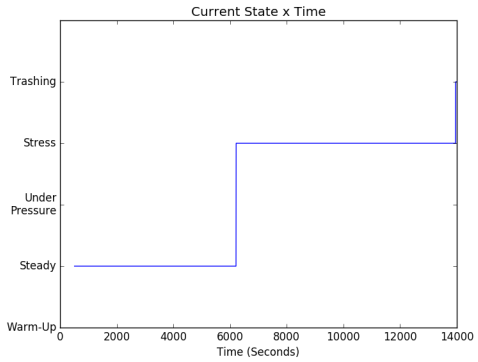
Figura 8: Gráficos com estados resultantes de diferentes tamanhos de janelas para simulação de 3500 instantes



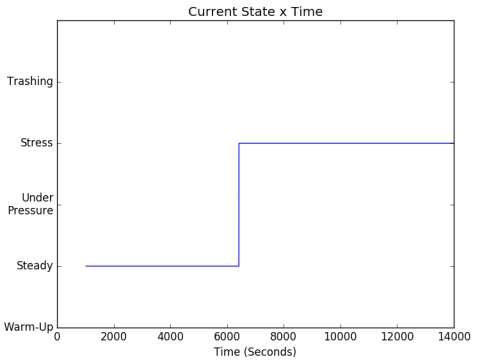
(a) Janela deslizante de tamanho 2



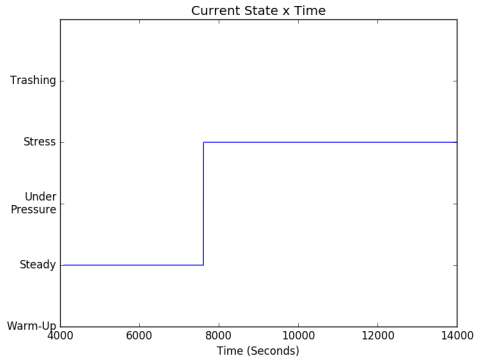
(b) Janela deslizante de tamanho 64



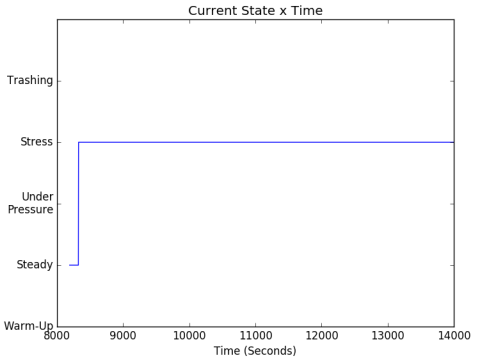
(c) Janela deslizante de tamanho 512



(d) Janela deslizante de tamanho 1024

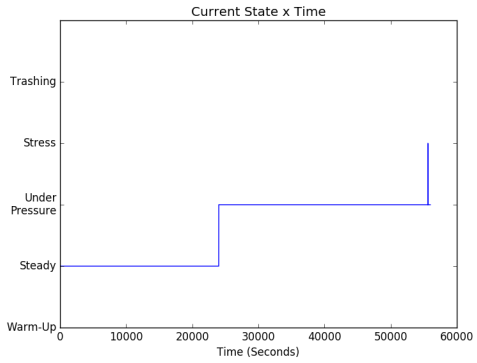


(e) Janela deslizante de tamanho 4096

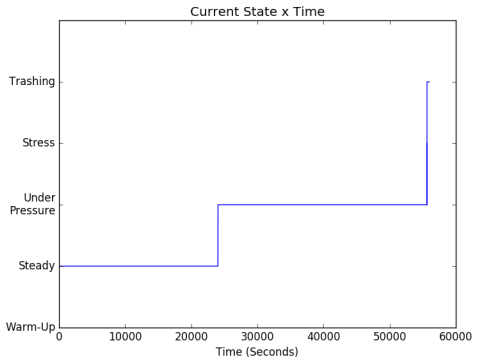


(f) Janela deslizante de tamanho 8192

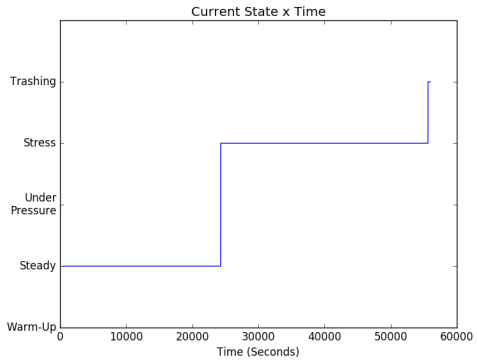
Figura 9: Gráficos com estados resultantes de diferentes tamanhos de janelas para simulação de 14000 instantes



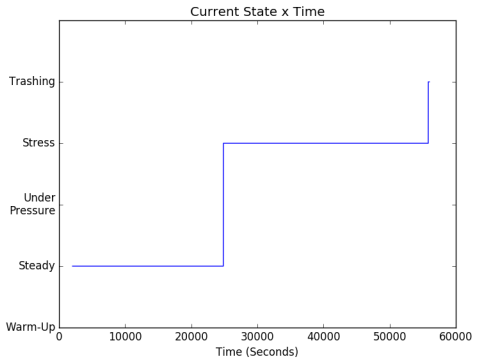
(a) Janela deslizante de tamanho 2



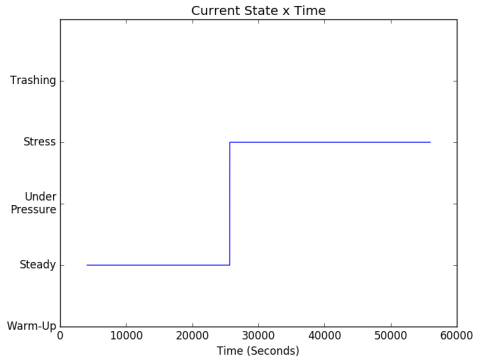
(b) Janela deslizante de tamanho 64



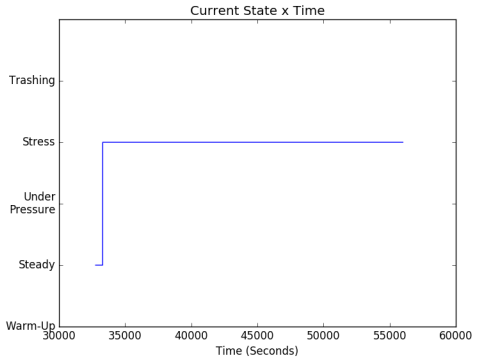
(c) Janela deslizante de tamanho 512



(d) Janela deslizante de tamanho 2048

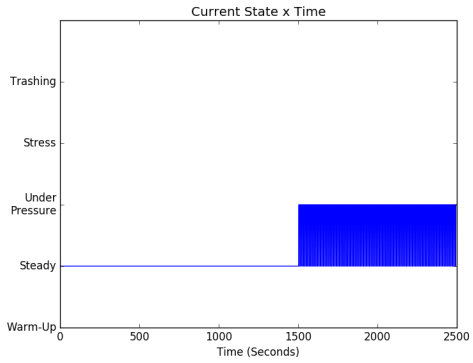


(e) Janela deslizante de tamanho 4096

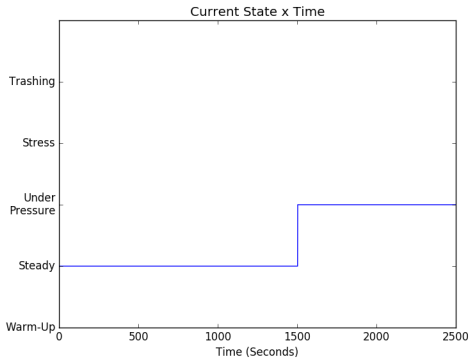


(f) Janela deslizante de tamanho 32768

Figura 10: Gráficos com estados resultantes de diferentes tamanhos de janelas para simulação de 56000 instantes



(a) Janela deslizante de tamanho 2



(b) Janela deslizante de tamanho 8

Figura 11: Gráficos com estados resultantes de diferentes tamanhos de janelas para simulação volátil

Para decidir um tamanho de janela adequado, começaremos eliminando alguns menos viáveis. Por exemplo, alguns tamanhos de janela muito grandes nem conseguiram detectar o estado *Trashing* (Figuras 8c, 8d, 9e, 9f, 10e e 10f). Isto acontece pois estes tamanhos de janela levam uma imensa quantia de dados em consideração, muitos dos quais indicam que o SGBD está saudável, o que explica a ausência da transição para o estado *Trashing*.

É possível visualizar que existe uma proporção entre o tamanho da janela e o tempo que se leva para detectar a mudança de estados, isto é, quanto maior o tamanho da janela, maior pode ser o atraso para detectar transições. Por exemplo, nas Figuras (9c, 9d e 9e) é possível visualizar que o estado *Stress* é detectado próximo ao instante 6200 quando a janela possui tamanho 512, e a mesma transição só é detectada aproximadamente nos instantes 6800, 7700 e 8700 para janelas de tamanho 1024, 4096 e 8192, respectivamente.

Por outro lado, tamanhos muito pequenos são demasiadamente voláteis, e podem dar muita vazão a alarmes falsos. Esta volatilidade pode ser identificada claramente visualizando as Figuras 11a e 11b. Para um tamanho de janela menor (2), os estados identificados se alteram constantemente, enquanto, com janela de tamanho 8, eles se mostram mais constantes por considerar dados anteriores que indicam que o SGBD se mantém constante apesar da variância. Portanto, quanto maior o número de dados analisados, mais garantia temos de que, se foi detectado um estado específico, o SGBD realmente se encontra nele.

Portanto, não existe um tamanho de janela ideal; este deve variar de acordo com o objetivo da análise. Se alarmes falsos não são um problema, deve-se usar um tamanho de janela pequeno. Se tomar conhecimento rápido de que o SGBD entrou em situação precária não for necessário, uma janela maior seria mais adequada.

Para fins de teste, usaremos uma janela de tamanho 60 (equivalente a 1 minuto de dados), para ter garantia de precisão e também velocidade de detecção suficientemente rápida. Este tamanho, por ser muito menor do que 13 dias, nos garantirá que não haverá problemas de velocidade de cálculo do estado do SGBD.

5.4. A coleta de dados influencia no desempenho do SGBD?

O *MoDaST*, por ser um sistema de testes e enviar todas as requisições, obtém dados a respeito do envio e recebimento de requisições / respostas do lado do cliente. No *OSD*, isto não é possível. Como é necessário coletar os dados do lado do servidor, isso pode interferir no resultado, pois não considera o tempo de latência.

Para definir se a coleta de dados influencia no desempenho do SGBD foi feito o seguinte experimento:

- Foi feito um teste de carga que envia 50 mil requisições em uma frequência de cerca de 50 requisições por segundo e espera suas respostas.
- Cada vez que uma requisição era respondida, guardava-se o tempo de duração da requisição (momento em que foi enviada até o momento da chegada da resposta).
- Em seguida, calculamos a média e o desvio padrão dos tempos das requisições.

Este processo foi realizado com a coleta de dados ativa e inativa.

Chamaremos a média de tempo de resposta com coleta inativa de α, e a média de tempo de resposta com coleta ativa de β. Também chamaremos de γ o desvio padrão do tempo de duração de requisições com coleta inativa.

A partir destes dados, geramos um gráfico que pode ser visto na Figura 12, no qual estão traçadas duas linhas referentes à $\alpha \pm \gamma$, representadas em verde e vermelho. Nele também indicamos uma linha referente à β.

É possível verificar que β se encontra entre o desvio padrão de α, o que indica que a influência da coleta no desempenho é muito pequena, e pode ser considerada desprezível.

Neste gráfico também é mostrado o tempo de resposta das requisições (área azul clara) enquanto a coleta estava ativa.

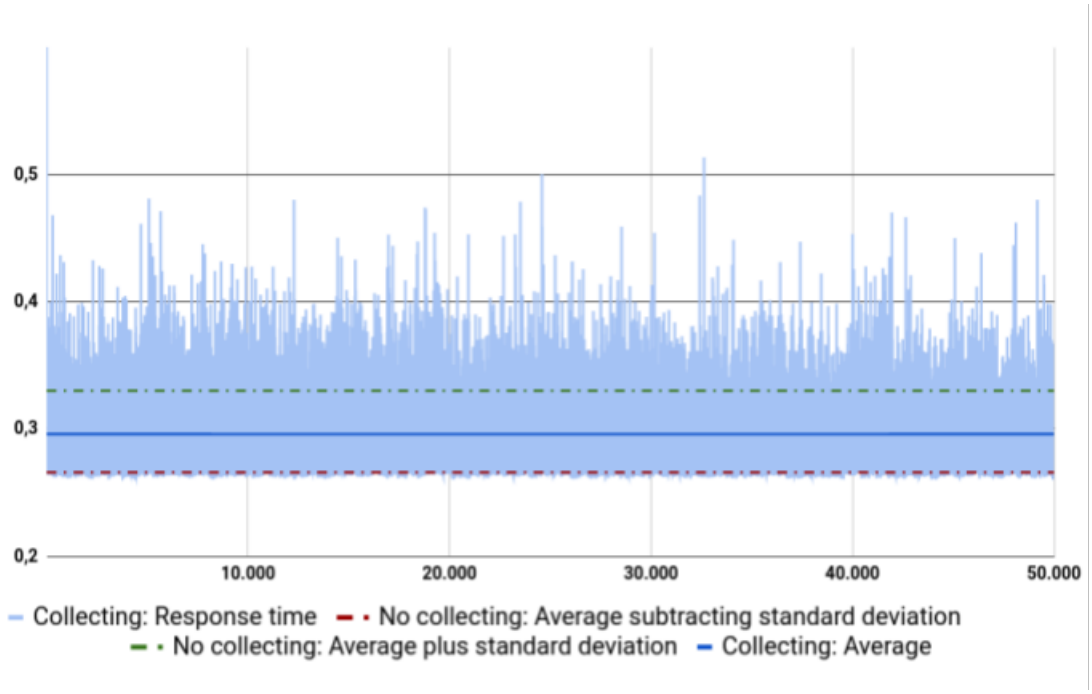


Figura 12: Gráfico de tempo de respostas das requisições

5.5. Há diferença entre os resultados dependendo do local da coleta?

Como o *OSD* precisa coletar dados referente ao número de requisições recebidas e respondidas do lado do servidor, é possível que haja uma divergência nestes dados. Se realmente houver divergência, é provável que isto interfira no resultado.

Em nossa análise, começamos executando diversos testes e coletando o número de requisições recebidas e respondidas de ambos os lados (cliente e servidor). A partir destes conjuntos de dados, escolhemos 3 testes que se mostraram mais promissores para a análise.

Para cada teste é definido uma constante de alcance σ. Os dados são selecionados através de um valor em um intervalo, e este intervalo é obtido através de um valor inicial ϕ acrescido de σ.

Os testes foram executados em 3 padrões diferentes [Ma et al. 2018], cada um obtendo ϕ de uma das seguintes formas:

- Sequencial (Figuras 13 e 14): Os dados de uma tabela do banco de dados são percorridos em ordem sequencial, isto é, na ordem em que foram *indexados*. Para isto, ϕ é inicializado em 0 e incrementado em um valor a cada consulta. Este incremento é aleatório, variando de 0.01 à 0.1.
- Aleatório (Figuras 15 e 16): Os dados de uma tabela do banco de dados são percorridos em ordem aleatória. Para isto, ϕ é definido através do seguinte cálculo: obtemos um número aleatório (que se encontra entre 0 e 1), multiplicamos ele por 100000 e o arredondamos para duas casas decimais.
- Enviesado (Figuras 17 e 18): Os dados percorridos são aleatórios, mas estão entre um intervalo fixo. Neste teste usamos valores entre 10000 e 10100. Desta forma, ϕ é definido a partir da soma entre 10000 e um número aleatório entre 0 e 100.

As Figuras 13, 14, 15, 16, 17, e 18 e apresentam em seu eixo X o tempo, no formato *Unix Timestamp*, isto é, número de segundos passados desde 00:00:00 de 1 de Janeiro de 1970.

As Figuras 14, 16 e 18 apresentam, no eixo Y, o estado de *stress* em que se encontra o SGBD, sendo 0 o estado *Warm-Up*, 1 o estado *Steady*, 2 o estado *Under Pressure*, 3 o estado *Stress* e 4 o estado *Trashing*.

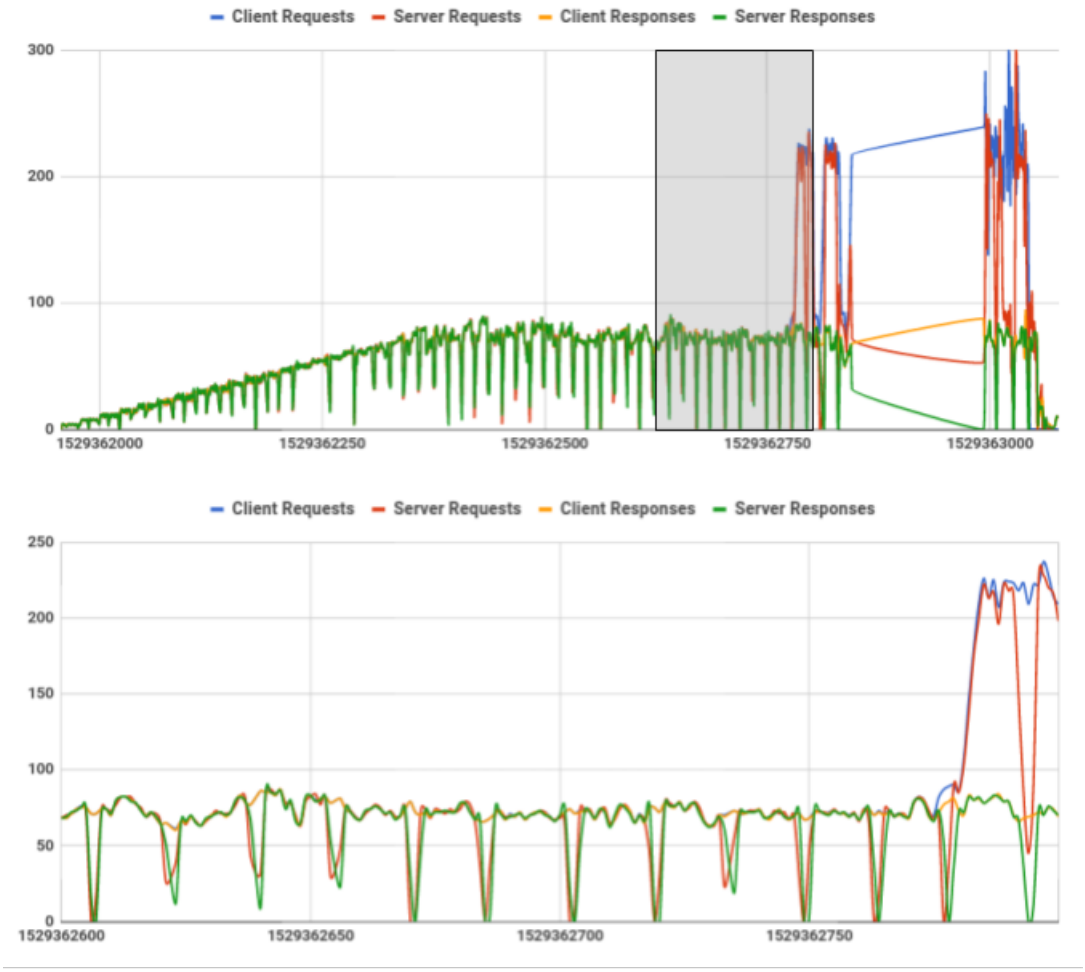


Figura 13: Gráfico de número de requisições realizadas e respondidas durante o teste sequencial e a ampliação do mesmo

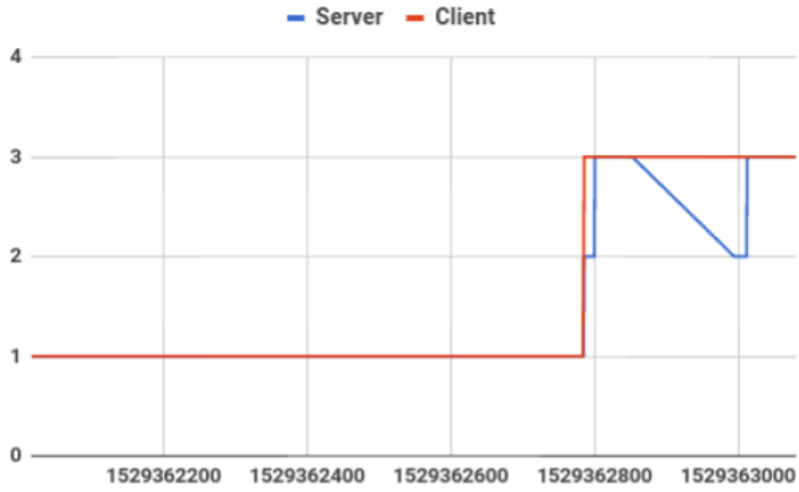


Figura 14: Gráfico de estados durante o teste sequencial

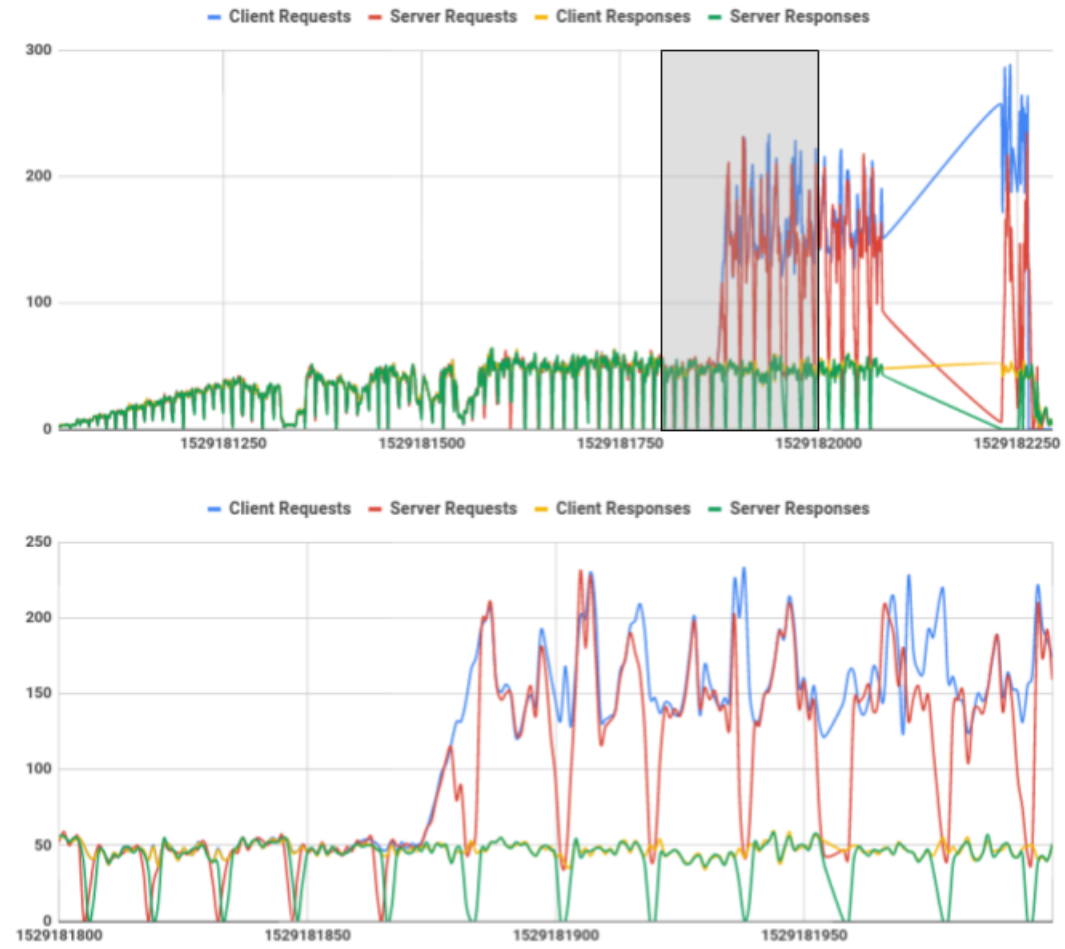


Figura 15: Gráfico de número de requisições realizadas e respondidas durante o teste aleatório e a ampliação do mesmo

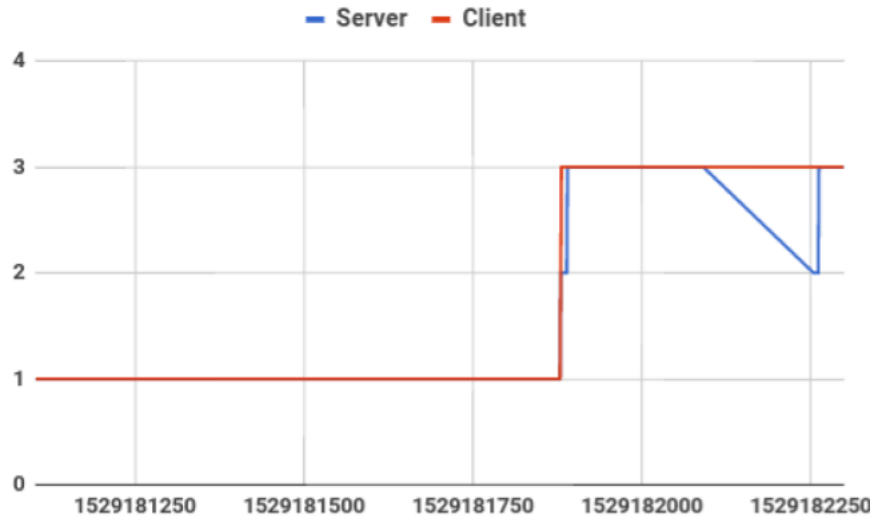


Figura 16: Gráfico de estados durante o teste aleatório

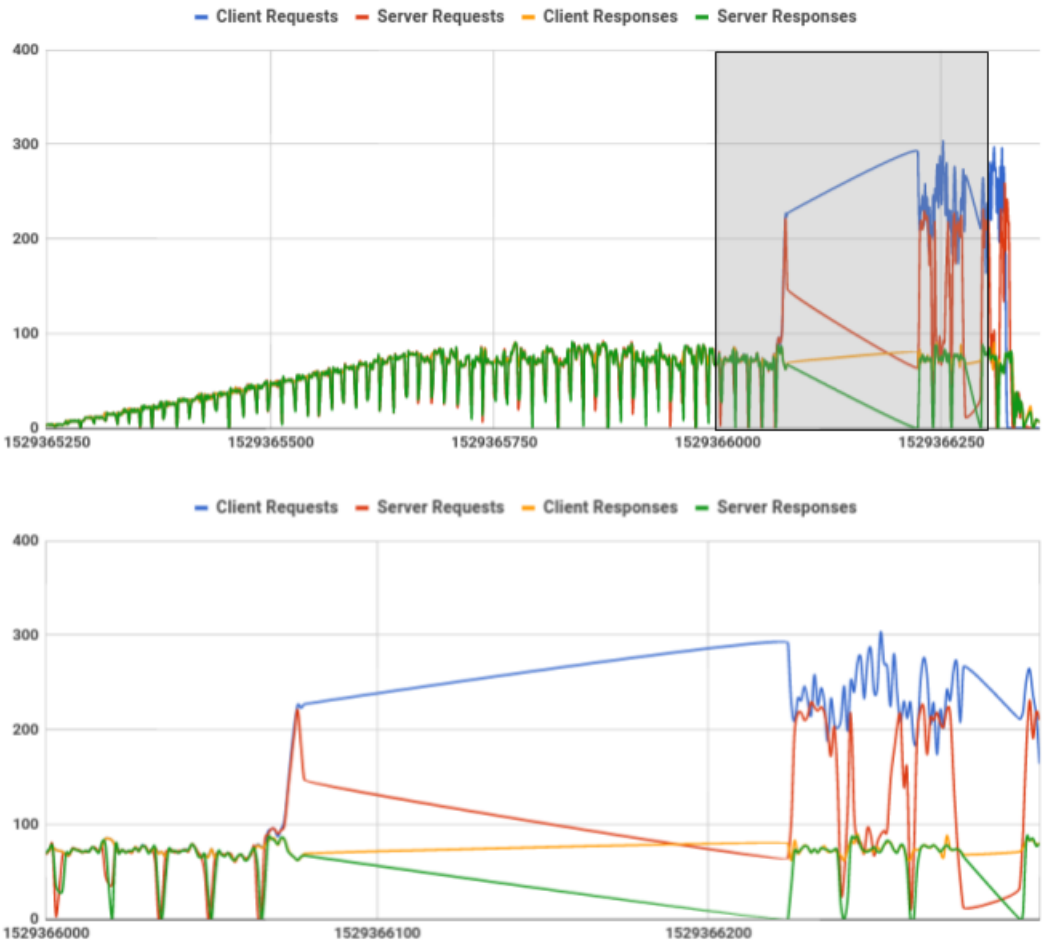


Figura 17: Gráfico de número de requisições realizadas e respondidas durante o teste enviesado e a ampliação do mesmo

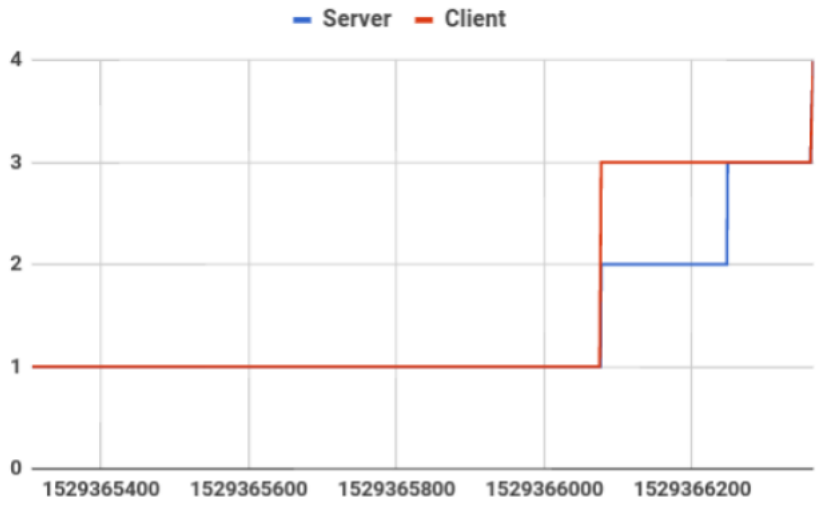


Figura 18: Gráfico de estados durante o teste enviesado

Observando os gráficos dos testes realizados constatamos que, em todos, os dados coletados do lado do servidor conseguem ser bem condizentes com os dados coletados do lado do cliente, tanto no número de requisições quanto no número de respostas e, consequentemente, nos estados do SGBD detectados.

Constatamos também uma falha na coleta no lado do servidor. Ela ocorre por alguns segundos quando o SGBD atinge o nível de *stress* ou *trashing*. Conforme o número de conexões começa a aumentar drasticamente, os recursos da máquina ficam muito ocupados, impedindo a máquina de coleta de estabelecer um canal *SSH* por tempo suficiente para ultrapassar o *timeout* desta conexão, o que gera este buraco de dados.

Apesar de tudo, esta falha não causa grande influência no cálculo e na análise de estados, como podemos ver nas Figuras 14, 16 e 18.

5.6. Dados da Máquina

Os dados de máquina coletados foram obtidos através do comando *vmstat*. As variáveis coletadas pelo comando são:

- *Buff*;
- *CS*;
- *BI*;
- *SO*;
- *BO*;
- *IN*;
- *ST*;
- *Free*;
- *SY*;
- *WA*;
- *R*;
- *Cache*;
- *B*;
- *ID*;
- *US*;
- *SI*;
- *Swaped*;

Como as variáveis são muitas, só iremos discursar a respeito de algumas delas (as que mostrarem maior correlação).

Para obter dados a respeito da correlação, comparamos cada variável com um valor que representa o estado atual do SGBD. Desta forma, utilizando a Correlação de Pearson, obtivemos coeficientes entre -1 e 1, indicando a correlação dos estados com a variável. Se o coeficiente for 1, isto indica que a variável tem correlação perfeita com os estados. Se for -1, indica uma relação perfeitamente inversa. Se for 0, significa que não existe nenhuma correlação entre eles.

Executamos os mesmos 3 padrões de testes e obtivemos o coeficiente de correlação para cada variável. Destes resultados, separamos as que consideramos mais significativas, isto é, aquelas que possuem pelo menos dois coeficientes de correlações com módulo

Tabela 3: Correlação das variáveis mais significativas

Teste/Variável	*CS*	*IN*	*Cache*	*Free*	*Buff*
Aleatório	0,74608870	0,86418905	0,15764650	-0,16217749	-0,78678902
Sequencial	0,66413847	0,57631000	0,78302363	-0,76243107	0,64492162
Enviesado	0,69085399	0,52398874	0,77424882	-0,71892176	0,63485192

maior que 0.5. Os resultados das variáveis mais significativas podem ser visualizados na Tabela 3.

Através da tabela, é possível notar que a maioria das variáveis possui uma correlação baixa o suficiente para que não seja possível usá-la para inferir o estado. Mesmo estas 5 variáveis, cujos valores são melhores, gerariam muita incerteza da acurácia. Mesmo assim, imaginamos motivos pelos quais elas poderiam ter correlação razoável com o estado de *stress*:

- CS e IN: CS é sigla de *Context Switches*, isto é, o número de trocas de contexto efetuadas pelo sistema operacional por segundo. Trocas de contexto são ativadas por 3 motivos principais: interrupções de *hardware*, troca de processo ativo pelo escalonador de processos ou entrada no modo *kernel*. Por isto é possível que as muitas trocas de contexto correspondam a um estado de *stress* elevado através de muitas interrupções das *threads* que executam as transações, por muita concorrência no mesmo núcleo de processamento ou muitas interrupções de aguardo/recebimento de dados. IN indica o número de interrupções do sistema por segundo, o que condiz com o valor de trocas de contexto.
- *Cache* e *Free*: *Cache* simboliza a quantidade de memória utilizada como *cache* e *free* simboliza a quantidade de memória ociosa. É curioso mas não surpreendente estes valores parecerem se opor. O motivo provavelmente é que, conforme a memória ociosa é ocupada, ela é usada como *cache*. A razão de estes valores possuírem uma correlação alta nos testes sequencial e enviesado provavelmente se deve ao fato de acontecerem muitos *cache hits*. Por não haver muitos *cache misses*, a *cache* demora a ficar ocupada, e vai crescendo de forma paralela ao estado de *stress*. Ao mesmo tempo isto explicaria porque o teste aleatório possui relação tão baixa com a *cache*: por serem acessos aleatórios, muitos *misses* acontecem, e muitos dados são trazidos como *cache*, se opondo ao estado de *stress*, que se mantém constante no começo do teste. De forma análoga, isto explica a correlação com a quantidade de memória desocupada.
- *Buff*: Representa a quantidade de memória utilizada como *buffer*. Dado que são armazenados como *buffer* metadados de arquivos (tais como permissões, localização, etc), eles servem de certa forma como uma *cache*, indicando dados a respeito dos arquivos que armazenam tabelas do SGBD, arquivos com códigos-fonte, etc.

6. Discussão

Neste trabalho, propusemos uma ferramenta cujo objetivo era analisar o estado de *stress* de um sistema gerenciador de banco de dados *online*. Ele possui capacidade de monitorar os dados tão instantaneamente quanto necessário, dependendo do tamanho da janela de dados analisada, e pode processar um período extenso, de mais de 13 dias, sem comprometer seu desempenho. Além disso, diferentemente do que pensávamos ao início do

trabalho, nem sempre a maior janela de dados traz o melhor resultado. De fato, ao mesmo tempo que janelas maiores garantem menos volatilidade, elas também geram atraso na detecção de mudança de um estado, pois consideram mais dados de quando o SGBD estava saudável. Verificamos que é possível processar os dados sem impactar significativamente o desempenho do SGBD analisado, e que o *OSD* consegue realizar isto, devido à sua política de buscar dados no servidor mas processá-los em uma máquina auxiliar. Além de não impactarmos no desempenho conseguimos, com uma pequena exceção, coletar dados com uma boa acurácia, boa o suficiente para calcularmos os estados do SGBD baseados no MoDaST. Fizemos apenas uma mudança na máquina de estados proposta, a remoção do estado *Warm-up*, pelo fato de que, no contexto do monitoramento, não há como reinicializar a base de dados para começar a coleta. Analisamos também alguns dados da máquina para ver se existe alta correlação entre algum dos dados analisados e o estado de *stress*, verificando que algumas das variáveis possuem uma correlação que permitiria tentar inferir os estados, mas que não traria grande confiabilidade a eles.

Referências

Ji-Woong Chang, Kyu-Young Whang, Y.-K. L. J.-H. Y. Y.-C. O. (2005). A formal approach to lock escalation. KAIST, Elsevier.

Ji Zhu, James Mauro, I. P. (2002). R-cubed (r3): Rate, robustness, and recovery - an availability benchmark framework.

Ma, L., Aken, D. V., Hefny, A., Mezerhane, G., Pavlo, A., and Gordon, G. J. (2018). Query-based workload forecasting for self-driving database management systems. In *Proceedings of the 2018 ACM International Conference on Management of Data*, SIGMOD '18.

Meira, J. A., de Almeida, E. C., Kim, D., Filho, E. R. L., and Traon, Y. L. (2005). "overloaded!" — a model-based approach to database stress testing. UFPR, University of Luxembourg, Springer, Cham.

Raghu Ramakrishnan, Johannes Gehrke, J. D. S. S. L. Z. (2007). *Database Management Systems Solutions Manual*. Ed. McGraw-Hill, 3rd edition.