A PERFORMANCE DE SOFTWARE CONSIDERANDO O TAMANHO DA BASE, QUANTIDADE DE CLIENTES E DE TAREFAS SIMULTÂNEAS

Morães, Martin (Sistemas de Informação/UNIBRASIL)
Pombeiro, Orlei José (Sistemas de Informação/UNIBRASIL)
Morães, Danilo (Direito/UNIBRASIL)

No desenvolvimento de um sistema, durante a análise e a implementação, tem-se a preocupação quanto a performance do mesmo, no que tange as relações entre a persistência de dados, quantidade de usuários e a quantidade de tarefas que podem ser executadas concorrencialmente (thread). Estes mesmos fatores crescem ao longo da existência do sistema. Os dados são armazenados e a base de dados só cresce, novos colaboradores interagem com o sistema e novas funcionalidades são implementadas, principalmente em função dos mobiles e da web que propiciam várias thread para cada usuário que se conecta ao sistema. Este trabalho focou em mapear as grandezas das relações destes três fatores na performance do sistema. Ter um conjunto de referências com grandezas objetivas propicia, aos profissionais envolvidos no desenvolvimento e manutenção, parâmetros que contribuirão no dimensionamento das tecnologias e soluções a serem adotadas. Para identificar as grandezas, objetivo deste trabalho, foi desenvolvido um software para gerar requisições em bancos de dados (DB), onde o mesmo registra o tempo de cada operação. Neste software implementou-se o recurso de multithread com o objetivo de identificarmos a variação da performance com diferentes quantidades de threads. O estudo foi realizado na base de dados gerado pelo software. Os diferentes senários são analisados com foco na evolução da performance. O estudo proposto foca a performance em relação as três variáveis apresentadas, independente do Sistema de Gerenciamento de Banco de Dados (SGDB). São diversos os estudos que apresentam comparação do desempenho entre os diferentes SGDB, estes estudos são geralmente denominados de benchmarking. Este trabalho não incorpora as preocupações que são relevantes nos estudos de benchmark em softwares. Os cenários analisados são com bases de

dados relacionais em tabelas puras, sem índices, chaves e etc... Identificou-se com que o tamanho da base de dados tem um pequeno impacto nas operações de inserção de dados, já nas operações de consulta o impacto é mais significante. Com requisições por meio das threads, o aumento de requisições no DB não impacta na perda de performance do sistema bem como não gera ganhos significativos. O trabalho mostrou resultados muito úteis aos desenvolvedores, principalmente aos que não tem experiência nos cenários aqui trabalhados.

Palavras-chave: Performance; Banco de Dados; Multithread; Benchmark.

1 Gênese do problema estudado;

Os sistemas interligam-se disponibilizando e consumindo dados com a necessidade de grande desempenho. A expressão “just in time” utilizado na gestão, principalmente no campo da indústria transmite a ideia de produzir e entregar os produtos a tempo (just in time) (LIMA). No contexto deste artigo o produto está relacionado aos dados na forma de textos, imagens, vídeos e sons.

Desenvolver sistemas que atenda em “just in time” significa tratar, transportar e disponibilizar um volume crescente de dados no tempo em que o usuário não tenha que esperar.

Ao utilizarmos um sistema, fazemos requisições por meio da interface do aplicativo e a resposta da requisição nos é entregue/disponibilizada, geralmente em uma interface amigável. Exemplificando: Em um navegador web o usuário digita o endereço desejado ou clica em um link, após esta operação a página requisitada é apresentada com os textos, imagens, outros links e toda formatação de apresentação.

Entre a requisição do usuário e o começo da entrega há um período de latência, ou seja, período de tempo compreendido entre o início de um estímulo e o começo de uma resposta, relacionada ao mesmo estímulo. (DICIONARIO OnLine). No período de latência de um aplicativo acontecem operações visando entregar o solicitado no menor tempo, este enfoque é tratado como desempenho ou performance. Diminuir está latência é aumentar a performance ou o desempenho.

São vários os recursos e as técnicas visando uma melhor performance dos sistemas. Estudos são realizados para diminuir a latência de disco, de memória de SGDB. Tratando de performance de SGDB Anderson Supriano (2006) realizou um trabalho considerando os avanços e necessidades.

> Embora haja uma diminuição maior da latência de disco em relação à da memória principal, ela continua sendo importante em consultas OLTP (Online Transaction Processing), que possuem um padrão de acesso de um bloco por vez.
>
> Estudos mostram que em vários SGBDs a CPU fica parada na maioria do tempo. Grande parte desse tempo parado é causado pela latência de memória. Outro fator é que as instruções de máquina usadas nos algoritmos típicos de SGBD são altamente interdependentes, causando paradas por dependência. (Anderson Supriano, 2006, p. 2)

Outros trabalhos tratando de diminuir todas as latências de hardware e software foram e continuam sendo, realizados. A cada nova tecnologia e recurso colocado no processo, as variáveis se alteram e novos estudos são necessários visando o melhor aproveitamento dos recursos disponíveis.

Este estudo volta-se para observar a latência entre a requisição do usuário e a resposta do banco de dados considerando a quantidade de threads em operação.

As técnicas utilizadas no desenvolvimento dos softwares impactam na performance dos sistemas. A formação dos desenvolvedores é voltada a produtividade, ou seja, implementar (codificar) os casos de uso no menor tempo com um mínimo de qualidade do código. As questões de performance, na maioria das vezes, são percebidas quando o sistema entra em operação.

Monitoramento em tempo real, internet das coisas[1], redes sociais, pesquisas em diversas plataformas e bases de dados, são exemplos de grandes volumes de dados sendo enviados e recebidos, ou seja, sendo gerados e consumidos por diversos usuários de forma simultânea.

Os sistemas podem ser desenvolvidos para executarem todas as tarefas de forma sequencial ou de forma paralela utilizando os recursos de threads. De forma sequencial, enquanto uma tarefa não for concluída outra tarefa não inicia, mesmo que os recursos necessários estejam disponíveis.

---

[1] Internet das Coisas – é a interconexão dos aparelhos, domésticos, de escritórios e etc... com os seus usuários.

Esta situação fica melhor entendida quando visualizamos o seguinte exemplo:

Considere um navegador web implementado para processar as tarefas sequencialmente. O usuário solicita uma determinada página. O servidor envia os códigos da página. O navegador analisa o código da página e identifica que a página tem uma imagem, então o navegador solicita a imagem ao servidor e aguarda até a imagem chegar. Quando a primeira imagem chegar, o navegador continua a analisar o código e identifica que a página tem mais uma imagem, então o navegador solicita a segunda imagem e novamente fica aguardando a chegada da segunda imagem e assim procede até processar toda a página.

Neste processamento sequencial, na latência existente, tem recursos de computador, rede e servidor sobrando e não está sendo utilizado porque o navegador utilizado (software) não trata as tarefas em paralelo, ou seja, simultaneamente.

2 Justificativas de estudo

O desenvolvimento de sistemas (softwares) sempre foi uma tarefa complexa e desafiadora. Com o crescimento da massa de dados, com tipos de dados variados e vários sistemas interconectados, a arte de desenvolver sistemas mantem, mais do que nunca a preocupação com a performance.

> O aspecto de desempenho de sistemas de informação tem se tornado cada vez mais crítico nas organizações, em função do crescente volume de dados manipulado pelos sistemas e da complexidade das requisições feitas à base de dados. Em especial, a complexidade no acesso aos dados armazenados tem grande influência em aplicações OLAP *(On-Line Analytical Processing),* que acessam grandes conjuntos de dados através de consultas de alto custo, de natureza não previsível *(ad-hoc)* (MATTOSO; ZIMBRÃO; LIMA; BAIÃO, 2005, p. 1)

Os sistemas operacionais que são multiprogramados, ou multitarefa aproveitam os recursos de memória, processador, disco e etc... permitindo que vários processos disputem os recursos do sistema. (Rosa Junior, Leomar Soares Da, 2007, p. 19).

Em algumas situações, dividir uma tarefa em outras tarefas, possibilita um melhor desempenho do software e também é possível que o inverso aconteça,

gerando uma perda de performance.Ou seja, assim como um software só executa uma tarefa por vez pode ter uma latência significativa, um software que divide uma tarefa em muitas tarefas pode também gerar latência e sobre carga de todo o sistema.

Conhecer alguns impactos do desenvolvimento de software multitarefa, propicia aos desenvolvedores um conjunto de referências que facilitam as escolhas de técnicas durante a codificação dos sistemas

O desenvolvimento fundamentado em referências e resultados observados, facilitam a aplicação de soluções e favorecem o surgimento de novas soluções e possibilidades.

3 Os objetivos de análise;

Este estudo volta-se para observar a latência entre a requisição do usuário até a resposta do banco de dados, considerando a quantidade de threads em operação.

Identificar a quantidade ideal de threads dentro de um senário definido. Optou se por observar as operações de inserção em um banco de dados, variando a quantidade de dados na operação e a quantidade de threads.

O objeto deste trabalho asemelhasse aos objetivos dos benchmark de bando de dados que são:

> No *benchmark de bancos de dados* um conjunto de transações, ou a carga de trabalho como é comumente denominada, é executada sobre um banco de dados conhecido. Tanto a carga de trabalho quanto o banco de dados podem ser *reais ou sintéticas (artificiais).* (Rafles Frausino Pereira, 1990, p. 6)

Lo Moreira et al. (2012, p. 163) diz que cada benchmark elenca um conjunto de pressupostos próprios, o que torna complexo a utilização.

PIRES et al. (2006, p. 1) trada dos benchmark que definem um conjunto de instruções padronizadas para que os resultados possam ser comparados com outros benchmark. Ele menciona:

> Benchmarks são padrões utilizados em comparações de sistemas computacionais. A aplicação de testes seguindo tais padrões gera medidas quantitativas de desempenho capazes de serem comparadas com as de outros sistemas.
>
> Em ambientes computacionais, benchmarks são utilizados para mensurar o desempenho de bancos de dados, sistemas

> operacionais, entre outros. Dentre os benchmarks para bancos de dados destacamos o AS3AP.
>
> Em ambientes computacionais, um *benchmark* é tipicamente um *software* que realiza um conjunto restrito e pré-definido de operações (uma carga de trabalho) e retorna um resultado em algum formato (uma métrica), que descreve o comportamento do sistema. As métricas dos *benchmarks* computacionais podem medir rapidez (em qual velocidade a carga de trabalho foi completada) ou vazão (quantas cargas de trabalho por unidade de tempo foram medidas). (PIRES; NASCIMENTO; SALGADO, 2006, p. 1)

PIRES et al. (2006, p. 1) cita algumas características de alguns benchmark clássicos:

> O *benchmark* OSDB *(Open Source Database Benchmark)* foi criado com o objetivo inicial de avaliar a taxa de I/O e o poder de processamento da plataforma GNU Linux/Alpha. Sua implementação é baseada no *benchmark* AS3AP, diferindo em alguns aspectos, tais como: quantidade de métricas retornadas e número de módulos. Enquanto a análise dos resultados gerados pelo *benchmark* AS3AP baseia-se em uma única métrica (tamanho máximo do banco de dados suficiente para completar o teste em menos de 12 horas), o OSDB possibilita a comparação através de várias outras métricas, tais como: tempo de resposta das consultas e número de linhas retornadas por segundo.
>
> O *benchmark* TPC-C é a principal referência no mundo quando se trata de desempenho de sistemas computacionais. (PIRES; NASCIMENTO; SALGADO, 2006, p. 2)

Rafles (1990, p. 7) trata da aplicação dos benchmark para apoio a escolha de um SGDB.

> Em termos globais, a realização de um *benchmark* para auxiliar na escolha de SGBDs exige um custo relativamente elevado. Isso se deve não somente ao tempo de máquina utilizado, como também ao custo do pessoal alocado ao processo. Desta forma, para que a aplicação da técnica de *benchmark* possa ser bem sucedida, deve haver um planejamento antecipado, detalhando cada uma das etapas: concepção, coleta de dados, construção do banco de dados de teste, conjunto de transações e finalmente o roteiro de execução dos testes, de modo que a execução do *benchmark* seja realizada dentro do tempo planejado, minimizando eventuais interrupções dos sistemas em produção e evitando, assim, desperdícios de recursos. (Rafles Frausino Pereira, 1990, p. 7)

Estas mesmas ponderações de Rafles (1990, p. 7), se aplicam ao contexto deste trabalho em que buscamos entender o comportamento da latência entre a requisição do usuário e a resposta do banco de dados.

Nos benchmark clássicos não foi encontrado uma proposta que aderisse ao que se deseja observar. Primeiramente busca-se encontrar o fenômeno, sendo que os benchemark tem foco principal na melhor performance.

Com este senário optou-se por construir um software considerando as melhores práticas deste contexto. Dentro das melhores práticas manteve-se o foco de não distanciar de um ambiente de produção, ou seja, como comumente os softwares são implementados e colocados em operação. Buscou-se também manter no software o mais adaptável sem a necessidade de alterar o código.

O software foi desenvolvido em Java. O ambiente utilizado para gerar as amostras foi com o JRE 1.8.0_66, o SGDB MySQL 5.6, com o mysql-connector-java-5.1.37, no SO Windows 10, em uma máquina com processador Intel i7, com 8G de ram. O aplicativo e o servidor SGDB, rodaram na mesma máquina, na geração das amostras.

A geração das amostras consistiu na inserção de dados em uma tabela com um campo chave primária, auto incremente int(11) dois campos varchar(300).

Foi realizada vinte e três (23) tomadas de amostras, com variações entre elas. As variações foram quanto a quantidade de caracteres enviados na inserção de dados nos campos varchar (40 e 200 caracteres), a quantidade de threads em execução (1, 3, 10, 20 e 30 threades) e em bases acumulativas ou únicas.

As bases acumulativas (“a”) consistem em novas tabelas, na mesma base, para um novo ciclo de amostragem em que a quantidade de threads é diferente. As bases únicas (“u”) consistem em uma única tabela na base de dados, para cada ciclo de amostragem. Esta distinção entre bases acumulativas e únicas foi adotada para observar se existe interferência na latência.

As amostras foram nominadas de pa040, pa040a, pu040, pa200 e pu200 mais a relação com a quantidade de threads. O “p” foi adotado como o prefixo para todas as amostras. O “a” seguinte, identifica se a base é “acumulativa” ou pode ser “u” identificando uma base “única”. O 040 ou o 200 indicam a quantidade de caracteres utilizada na inserção. O último “a” (uma única ocorrência) indica que é uma segunda amostra com os mesmos parâmetros de outra. Cada amostra consiste na geração de dados com 1, 3, 10, 20 e 30 threades.

Na Tabela 1 - Distribuição das amostras, apresenta a quantidade de registros em cada situação da amostra, ou seja, vamos tomar como exemplo a amostra “pa040” que foi gerada em uma base acumulativa, primeiramente com uma

thread, depois com três threads, depois com dez threads, depois com vinte threads e por fim com trinta threads. Os dados gerados na mesma base mas para cada quantidade de threads os dados foram inseridos em uma tabela diferente.

A quantidade de registros gerados em cada amostra não são grandezas comparáveis entre si, considerando que o tempo entre as amostras não são iguais.

Tabela 1 - Distribuição das amostras

Bases de dados	Thread				
	01	**03**	**10**	**20**	**30**
pa040	33848	17786	38290	47639	128710
pa040a	33219	42503	46258	78879	42308
pa200	17201	17498	54773	53625	82528
pu040		38898			
pu040			63865		
pu040				58938	
pu040					49923
pu200		21723			
pu200			57379		
pu200				65131	
pu200					53739

Foram geradas três amostras com quarenta (40) caracteres, sendo duas acumulativas denominadas de pa040 e pa040a e uma única denominada de pu040, e foi gerado duas amostras com inserção de duzentos (200) caracteres sendo uma acumulativa e uma única, denominadas de pa200 e pu200.

As considerações e cuidados das técnicas de benchmark, como visto, não se aplicam a este estudo pois o objetivo é observar a latência na execução de instrução com o banco de dados nos senários apresentados, dados estes que servirão de referências em estudos futuros e no desenvolvimento de aplicativos.

Definimos “tempo de execução” o tempo compreendido entre ter um *preparedstatement* e a confirmação da operação pelo banco de dados. Veja a sequência abaixo que exemplifica as tomadas de tempo. Consideramos neste estudo como “tempo de execução” a diferença da tomada de tempo entre o paço 6 e o paço 2.

Tabela 2 - Sequência de operações.

Passos	Síntese das operações
1	Pegar uma conexão com o banco;
2	**Tomada de tempo**
3	Preparar a instrução sql (*preparedstatement*);
4	Enviar a instrução sql ao banco;
5	Receber confirmação de execução da instrução sql;
6	**Tomada de tempo**
7	Encerrar conexão com o banco.

Para cada operação de inserção, foi mantido um log com o tempo de execução. Este tempo de execução registrado em nanosegundos[2] é a base para os resultados aqui observados. Para fins de análise, as ocorrências foram agrupadas, tomando por base a média dos tempos de execução no mesmo segundo[3], tratamos esta média simplesmente como "tempo".

Foi adotado o termo "tempo" para evitar que os números apresentados sejam vistos dentro de uma unidade de tempo, como são tratados nos "benchmarking" tradicionais. Os números aqui apresentados devem ser observados simplesmente como em um período de tempo igual para todas as análises.

4 Resultado e principais doutrinas;

4.1 Threads

Inicialmente destaca-se o comportamento das threads na geração das amostras. Como já era de se esperar as threads não tem uma quantidade de execuções iguais.

A relação entre a thread que mais teve operações e a que teve menos operações, estão apresentadas na Tabela 3 - Relação entre as threads da mesma amostra. Por não haver possibilidade de variação nas operações com uma (1) thread a mesma não foi apresentada na referida tabela.

[2] A referência a nanosegundos visa indicar o grau de precisão dos logs, não tendo valor para outros comparativos.
[3] A referência a segundo visa indicar a primeira transformação nos tempos observados, não tendo valor para outros comparativos.

As operações com três (3) threads, em todas as amostras não apresentou variação, ou seja, as três (3) threads tiveram, basicamente, a mesma quantidade de operações. A variação que teve, quando teve, foi de mais um (1) ou menos uma (1) operação. Desconsideramos esta variação pois pode ocorrer, nas gerações das amostras, o encerramento do processo sem que todas as threads tenham completado o ciclo.

As amostras pa040a e pu040, mostram uma variação aproximada e a amostra pa040 mostra um distanciamento em relação as outras duas. Observa-se também um distanciamento da variação entre as amostras pa200 e pu200. Pode-se observar que o comportamento das operações das threads são variáveis, ou seja, nem todas as threads tem as mesmas oportunidades de processamento, fato este largamente tratado nas documentações específicas.

Para este estudo, este comportamento das threads não é significativo. Considerando que o foco está em observar a latência na execução de instrução com o banco de dados nos senários apresentados.

Tabela 3 - Relação entre as threads da mesma amostra.

Thread	pa040	pa040a	pu040	pa200	pu200
3	0%	0%	0%	0%	0%
10	3%	3%	2%	5%	3%
20	110%	83%	80%	147%	92%
30	195%	240%	267%	132%	229%

4.2 Mediana

A Figura 1 e a Tabela 4 - Comparativo da mediana das amostras. Vinte e cinco amostras, sendo que as amostras de pu040 e pu200 para uma thread, são repetições das amostras pa040a e pa200 respectivamente, pois apresentam o mesmo senário.

Observa-se nas medianas das três (3) amostras com uma thread, que a variação do tempo de execução é de 38,8 a 47,1. A quantidade maior de caracteres (byte) de dados (cinco vezes mais dados) não apresentou uma variação significativa. A amostra pa040 (quarenta caracteres) se aproxima da amostra pa200 (duzentos caracteres). A segunda amostra de quarenta caracteres, a pa040a, apresenta uma variação mais significativa em relação a amostra de duzentos caracteres.

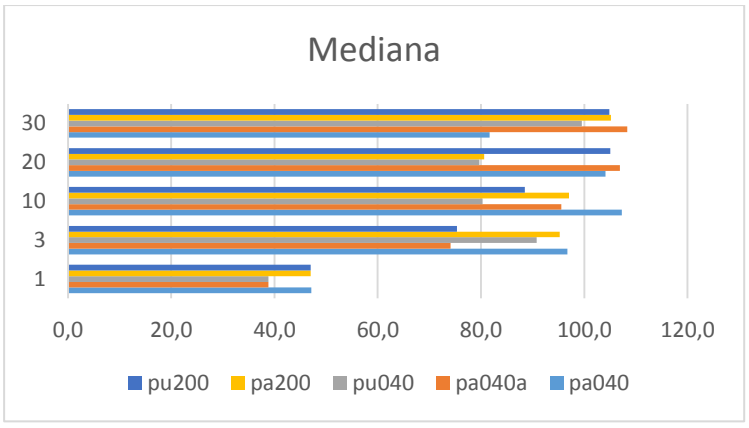
Com três ou mais threads, observa-se um aumento na mediana do tempo de execução. Indica uma tendência no aumento da mediana do tempo de execução conforme aumenta a quantidade das threads em execução.

Não é conclusivo que a quantidade de caracteres interfira nos cálculos da mediada do tempo de execução.

Tabela 4 - Comparativo da mediana das amostras.

Mediana	pa040	pa040a	pu040	pa200	pu200
1	47,1	38,8	38,8	47	47
3	96,8	74,1	91	95	75
10	107,3	95,6	80	97	88
20	104,1	106,9	80	81	105
30	81,7	108,3	100	105	105

Figura 1 - Comparativo das medianas das amostras.



4.3 Média

A Figura 2 e a Tabela 5 - Comparativo da média das amostras. vinte e cinco amostras, sendo que as amostras de pu040 e pu200 para uma thread, são repetições das amostras pa040a e pa200 respectivamente, pois apresentam o mesmo senário.

Para uma thread, a média e a mediana do tempo de execução não se distanciam de forma significativa. Observa-se que neste senário o tempo de execução das operações não sofre variações significativas. Quanto a quantidade de caracteres nas operações, observa-se um aumento na média do tempo de execução, conforme aumenta a quantidade de caracteres.

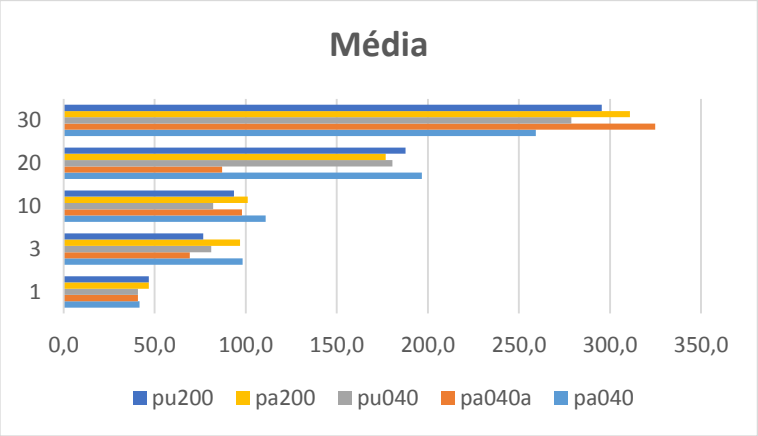
O aumento na média do tempo de execução, significa maior tempo para executar as operações. Observa-se na Figura 2 - Comparativo das médias das amostras. um aumento no tempo médio das execuções das operações. O aumento se dá pela quantidade de threads, ou seja, a média é calcula com a somatória de todos os tempos de execução de todas as threads, dividido pela quantidade de ocorrências.

É natural que encontremos mais tempo de execução quando se tem mais threads em execução. Esta média nos auxiliam a entender os próximos valores, onde observaremos o ganho no tempo médio das execuções com o aumento de threads e o tempo médio por threads.

Tabela 5 - Comparativo da média das amostras.

Média	pa040	pa040a	pu040	pa200	pu200
1	41,7	40,9	40,9	46,7	46,7
3	98,3	69,2	81,0	96,8	76,8
10	110,9	97,9	82,2	101,1	93,5
20	196,7	87,1	180,5	176,8	187,8
30	259,3	324,8	278,8	311,0	295,4

Figura 2 - Comparativo das médias das amostras.



4.4 Ganho no tempo médio

Os valores apresentados como ganho no tempo médio foram obtidos ao comparar os tempos médios de execução com 3, 10, 20 e 30 threads com o valor

resultante do tempo médio de execução de uma (1) thread, vezes a quantidade de threads respectivamente.

Para exemplificar considere a Tabela 6 - Apresentação da identificação do ganho. na linha A temos os tempos médios de execução com uma thread. Para identificar o percentual de ganho ao utilizar três (3) threads, multiplicou-se os valores da linha A por três (3), conforme apresentado na linha B, o que identificamos como **tempo médio estimado** de execução para três.

Efetivamente o tempo médio de execução com três (3) threads está apresentado na linha C que é transcrição da segunda linha de dados da Tabela 5 - Comparativo da média das amostras.

Tabela 6 - Apresentação da identificação do ganho.

Linha	pa040	pa040a	pu040	pa200	pu200
A	41,7	40,9	40,9	46,7	46,7
B	125,1	122,7	122,7	140,1	140,1
C	98,3	69,2	81,0	96,8	76,8
D	26,8	53,5	41,7	43,3	63,3
E	21,3%	43,5%	33,9%	30,9%	45,3%

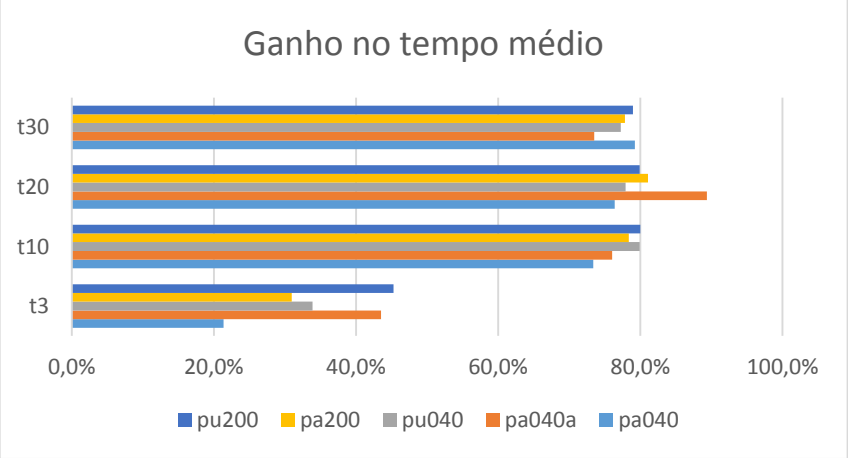
Subtraindo o tempo médio estimado (linha B) do tempo médio efetivo (linha C), obtemos o tempo médio ganho ao utilizar threads, ao invés de ter três execuções do aplicativo. A linha E apresenta o percentual do ganho do tempo médio de execução em relação a três threads.

A Figura 3 e a Tabela 7 - Comparação do ganho por tempo médio. apresenta os percentuais dos ganhos para o uso de threads. Observa-se que com três threads tem um ganho que varia de 33,9% a 45,3%. Observa-se também que a um ganho de três threads para dez threads. Os ganhos obtidos com a utilização de 10, 20 ou 30 threads são semelhantes entre si, no senário apresentado.

Tabela 7 - Comparação do ganho por tempo médio.

Ganho no tempo médio	pa040a	pu040	pa200	pu200
t3	43,5%	33,9%	30,9%	45,3%
t10	76,0%	79,9%	78,4%	80,0%
t20	89,3%	77,9%	81,1%	79,9%
t30	73,5%	77,2%	77,8%	78,9%

Figura 3 - Comparação do ganho por tempo médio.



4.5 Média por threads

A Figura 4 e a Tabela 8 - Comparativo das médias por threads das amostras. Vinte e cinco amostras, sendo que as amostras de pu040 e pu200 para uma thread, são repetições das amostras pa040a e pa200 respectivamente, pois apresentam o mesmo senário.

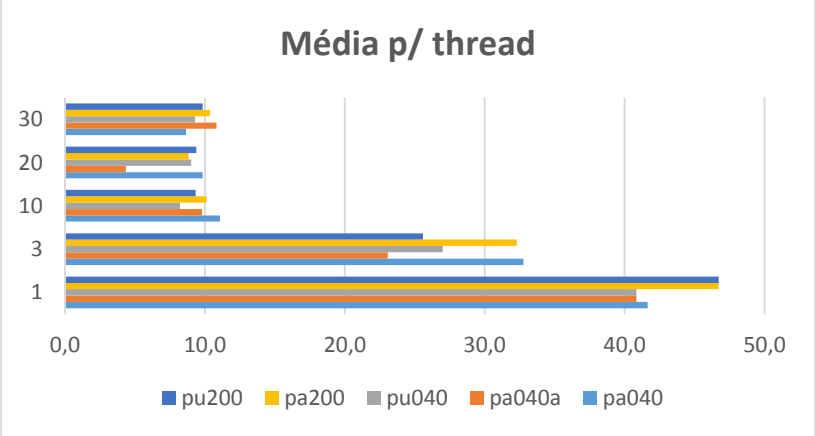
A média por threads é o tempo médio para cada thread realizar as operações. Este valor foi obtido pela divisão do tempo médio das operações pela respectiva quantidade de threads envolvida na amostra.

Observa-se que o tempo médio de execução considerando a quantidade de threads, diminui com o aumento de uma para três threads e de três para dez threads. Entre dez, vinte e trinta threads não tem uma variação significativa.

Tabela 8 - Comparativo das médias por threads das amostras.

Média p/ thread	pa040	pa040a	pu040	pa200	pu200
1	41,7	40,9	40,9	46,7	46,7
3	32,8	23,1	27,0	32,3	25,6
10	11,1	9,8	8,2	10,1	9,3
20	9,8	4,4	9,0	8,8	9,4
30	8,6	10,8	9,3	10,4	9,8

Figura 4 - Comparativo das médias por threads das amostras.



4.6 Desvio Padrão

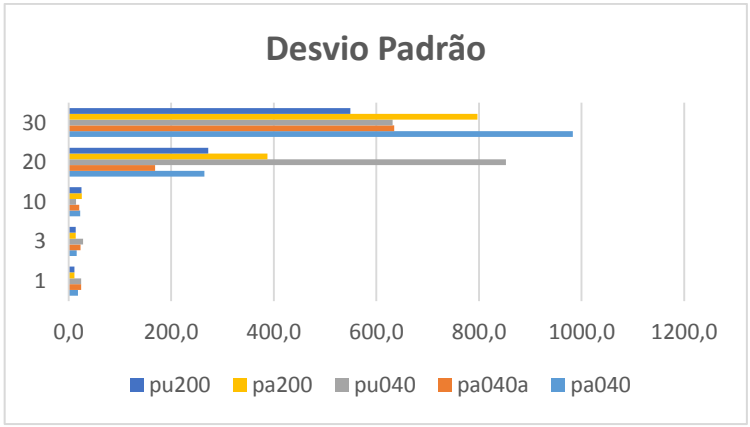
A Figura 5 e a Tabela 9 - Comparativo do desvio padrão das amostras. vinte e cinco amostras, sendo que as amostras de pu040 e pu200 para uma thread, são repetições das amostras pa040a e pa200 respectivamente, pois apresentam o mesmo senário.

Observa-se com no desvio padrão que com a utilização de vinte e trinta threads o tempo médio de execução para algumas operações é muito elevado.

Tabela 9 - Comparativo do desvio padrão das amostras.

Desvio Padrão	pa040	pa040a	pu040	pa200	pu200
1	18,2	24,4	24,4	11,7	11,7
3	15,7	23,2	28,3	14,1	13,9
10	22,9	20,7	14,3	25,9	25,1
20	264,6	168,9	852,8	388,0	272,5
30	982,8	634,8	631,6	797,2	549,1

Figura 5 - Comparativo dos desvio padrão das amostras.



5 Conclusão ou considerações finais;

Considerando o senário de softwares para simuladores, jogos e monitoramento de alta demanda a utilização de threads na persistência de dados tem um ponto de aproveitamento do ambiente em que o software está rodando.

Não são em todos os senários que o aumento de threads dá ganho de performance ao sistema. O aumento de caracteres de 40 para 200 não apresentou impacto nos tempos médios das operações.

Observou-se uma variação significativa nas amostras com o mesmo senário. É apropriado um refinamento neste estudo em busca de amostras mais acuradas.

Os resultados obtidos dão referências importantes aos desenvolvedores de software com senários semelhantes aos aqui testados, com threads e persistência de dados.

6 Referências

Anderson Supriano. Avaliação de um sistema de gerencia de banco de dados em memoria principal para uso em aplicações WEB. Evaluation of a main-memory database for use on web applications. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000401030>.

DICIONARIO OnLine. Disponível em: <http://www.dicio.com.br/>. Acesso em: 13 nov. 2015.

GOETZ. Java theory and practice. Dynamic compilation and performance measurement. Disponível em: <http://www.ibm.com/developerworks/library/j-jtp12214/>. Acesso em: 16 nov. 2015.

LIMA. O que é Just in Time? Disponível em: <http://www.administradores.com.br/artigos/carreira/o-que-e-just-in-time/21936/>. Acesso em: 13 nov. 2015.

Lo Moreira; SOUSA, F. R.C.; MACHADO, J. C. Analisando o desempenho de banco de dados multi-inquilino em nuvem. Analisando o desempenho de banco de dados multi-inquilino em nuvem. **… de Bancos de Dados (…**, 2012. Disponível em: <http://data.ime.usp.br/sbbd2012/artigos/pdfs/sbbd_shp_21.pdf>.

MATTOSO, M.; ZIMBRÃO, G.; LIMA, A. A.B.; BAIÃO, F. ParGRES: Middleware para Processamento Paralelo de Consultas OLAP em Clusters de Banco de Dados. ParGRES: Middleware para Processamento Paralelo de Consultas OLAP em Clusters de Banco de Dados. **Proceedings of the …**, 2005. Disponível em: <http://xa.yimg.com/kq/groups/42114393/1642589564/name/artigo_5middlwarePargresSBBDDemos2005v11-_OLAP.pdf>.

PIRES, Carlos; NASCIMENTO, Rilson O.; SALGADO, Ana. Comparativo de desempenho entre bancos de dados de código aberto. **Escola Regional de Banco de Dados, Anais da ERBD06, Porto Alegre**, 2006. Disponível em: <http://www.itautec.com.br/ifileexplorer/arquivo/empresa/documentos/comparativo_%20bancos%20de%20dados_%20sw%20livre.pdf>.

Rafles Frausino Pereira. **Analise de desempenho de banco de dados utilizando Benchmarks especializados**: Universidade Estadual de Campinas; Instituto de Matemática, Estatística e Ciência da Computação, 1990.

Rosa Junior, Leomar Soares Da. **Implementação de multitarefa sobre arquitetura Java embarcada FemtoJava**: Multitask implementation into femtojava embedded architecture, 2007. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/11301/000603425.pdf?sequence=1>. Acesso em: 13 nov. 2015.

TPC. Transaction Processing Performance Council. Disponível em: <http://www.tpc.org/information/about/abouttpc.asp>. Acesso em: 12 nov. 2015.