Desempenho de Sistemas com Dados Georeplicados com Consistência em Momento Indeterminado e na Linha do Tempo

Mauricio De Diana[1], **Marco Aurélio Gerosa**[1]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)

{mdediana,gerosa}@ime.usp.br

Abstract. *Large-scale web systems replicate data among data centers to achieve high levels of performance and availability. A consistency model defines the tradeoffs between these requirements and replica consistency. This work compared experimentally the performace of a storage system using eventual consistency and timeline consistency. The results show that, depending on the workload and the network conditions, system performances for each consistency model are similar. This comparative is helpful for development cost estimates and capacity planning of large-scale web systems.*

Resumo. *Sistemas web de larga escala replicam dados entre centros de dados para atingir altos níveis de desempenho e disponibilidade. Um modelo de consistência define o balanço entre esses requisitos e a consistência dos dados entre réplicas. Este trabalho comparou experimentalmente o desempenho de um sistema de armazenamento georeplicado usando consistência em momento indeterminado e consistência na linha do tempo. Os resultados mostram que, dependendo da carga de trabalho e das condições de rede, os desempenhos do sistema para cada um dos modelos de consistência são semelhantes. Essa comparação é útil em estimativas de custo de desenvolvimento e de planejamento de capacidade de sistemas web de larga escala.*

1. Introdução

Para atender centenas de milhares de usuários ininterruptamente em sistemas web de larga escala, dados são replicados em milhares de servidores distribuídos em múltiplos centros de processamento de dados em diferentes localizações geográficas. O principal problema do uso de replicação em redes de longa distância (WANs) é que manter as réplicas sempre consistentes entre si tipicamente implica em sacrificar outros requisitos também importantes, como desempenho ou disponibilidade. Um motivo para divergência entre réplicas é que a replicação entre centros de processamento de dados pode apresentar centenas de milissegundos de latência, período durante o qual as réplicas podem divergir. Essa latência é resultado não só das distâncias físicas entre os nós, mas também de limitações na largura de banda disponível e congestionamento de rede. Outro motivo de divergência entre réplicas é a falha temporária de um nó, que deixa de receber atualizações por um período de tempo. Um terceiro motivo é o particionamento de rede, causado por exemplo por uma falha em um equipamento ou enlace de rede que impeça a comunicação entre réplicas.

Nesse cenário, desenvolvedores e administradores de sistemas web de larga escala buscam um balanço entre disponibilidade, desempenho e consistência dos dados. Uma decisão comum em vários desses sistemas é o relaxamento da consistência em troca de alta disponibilidade e baixa latência. Entretanto, modelos de consistência mais relaxados permitem que conflitos entre réplicas aconteçam, o que torna o código da aplicação mais complexo devido a necessidade de implementação de mecanismos de resolução de conflitos e ações de compensação.

Este trabalho é um comparativo de desempenho de um sistema de armazenamento georeplicado usando dois modelos de consistência diferentes. Um deles, a consistência em momento indeterminado (*eventual consistency*[1]) é um modelo mais relaxado e se tornou especialmente popular após a publicação sobre o Dynamo da Amazon [DeCandia et al. 2007]. Um modelo de consistência para sistemas georeplicados menos popular, que busca um meio termo entre consistência forte e consistência em momento indeterminado é a consistência na linha do tempo, usada no PNUTS do Yahoo! [Cooper et al. 2008]. Ela é uma opção interessante por simplificar a programação, desde que seu desempenho seja próximo ao da consistência em momento indeterminado e a aplicação tolere níveis mais baixos de disponibilidade.

2. Consistência em Momento Indeterminado e na Linha do Tempo

A consistência em momento indeterminado garante que as réplicas vão sempre convergir em algum momento no futuro desde que novas atualizações cessem. Enquanto atualizações estiverem acontecendo, réplicas inconsistentes são possíveis, e clientes podem acessar dados desatualizados ou divergentes – por isso o sistema precisa implementar algoritmos de detecção e resolução de conflitos. Uma forma de diminuir as chances de conflitos é o uso de quóruns, cujo contraponto é a diminuição da disponibilidade do sistema quando um determinado quórum não é atingido [Vogels 2009]. Costuma-se usar os parâmetros N, R e W para definir os quóruns. N é o fator de replicação e representa a quantidade de réplicas existentes de um objeto. R/W é a quantidade de réplicas que precisam concordar com o mesmo valor para que uma leitura/escrita seja bem sucedida. Quando $N < R + W$, não existe possibilidade de clientes lerem dados inconsistentes, dado que existe intersecção entre os subconjuntos de réplicas para leitura e escrita.

Algumas aplicações web tornam-se mais simples com um modelo de consistência mais rígido. Por exemplo, uma aplicação de leilão não pode permitir conflitos no histórico de lances de um produto. Num sistema que usa consistência em momento indeterminado, no caso de uma falha que divida a rede em duas partições, usuários em cada partição têm uma visão própria do histórico de lances, equivalente a dois leilões simultâneos sobre o mesmo item.

A consistência na linha do tempo abre mão de disponibilidade em algumas situações em troca de consistência [Cooper et al. 2008]. Para cada objeto armazenado, ela permite atualizações em apenas uma de suas réplicas (réplica mestre). Devido à replicação assíncrona, réplicas podem ter valores desatualizados devido à latência de rede ou falhas, mas a qualquer instante sabe-se qual é a réplica com o valor mais recente. Os clientes

[1]O termo *eventual consistency* não foi traduzido como *consistência eventual* pois *eventual* é um falso cognato: em inglês indica que algo certamente acontecerá no futuro, enquanto em português indica que algo pode ou não vir a acontecer.

escolhem em cada acesso se aceitam como resposta apenas o valor mais recente ou se aceitam valores desatualizados. Além disso, como a réplica mestre define uma ordem de aplicação de atualizações nas outras réplicas, divergências não acontecem e mecanismos de detecção e resolução de conflitos não são necessários. A principal desvantagem da consistência na linha do tempo é que a existência de uma réplica mestre implica que escritas e leituras consistentes (leituras do valor mais recente) ficam indisponíveis em caso de uma falha que impeça o acesso a essa réplica.

O maior fator de impacto no desempenho da consistência na linha do tempo é o fato de que operações consistentes que não são feitas no centro de processamento de dados em que está a réplica mestre incorrem no custo de comunicação pela WAN. Mas os autores do PNUTS indicam que algumas aplicações no Yahoo! apresentam localidade de até 85% e relação escrita/leitura de 0,06 [Kadambi et al. 2011, Cooper et al. 2008]. Dado isso, eles implementaram uma heurística em que a réplica mestre migra para o centro de processamento de dados que recebeu as últimas 3 escritas. Dessa forma, em uma aplicação na qual a quantidade de leituras é muito maior do que a quantidade de escritas, o custo de comunicação de rede é baixo, em especial se as leituras não precisarem necessariamente do valor mais recente.

3. Planejamento dos Experimentos

Para entender o desempenho resultante de cada modelo de consistência, um estudo experimental foi realizado. Seu planejamento e execução usaram [Jain 1991] como principal referência metodológica.

Três técnicas são comumente usadas na análise de desempenho de sistemas: simulação, modelagem analítica e medição [Jain 1991]. Como inicialmente 33 parâmetros foram considerados (ver seções seguintes), seria difícil evitar a perda de precisão decorrente de simplificações necessárias para a criação de simuladores ou modelos com tantos parâmetros. Portanto, medição foi a técnica escolhida.

Um sistema precisava ser escolhido como objeto do estudo. Como nenhum sistema com os dois modelos de consistência foi encontrado, optou-se por implementar a consistência na linha do tempo no Riak, sistema de software livre que já traz consistência em momento indeterminado [Riak 2013]. Além do novo modelo de consistência, também foi implementado um algoritmo de particionamento que garante que existe ao menos uma réplica de cada objeto em cada centro de processamento de dados[2].

Como *benchmark*, foi usado o Basho Bench, específico para Riak [BashoBench 2013]. Ele foi adaptado para executar de forma distribuída, com uma instância para cada centro de processamento de dados.

Os experimentos emularam uma WAN pelo uso do netem, controlado pelo traffic control (tc). Ele provê funcionalidade para emulação de características de rede como latência e perda de pacotes. As configurações de rede foram alteradas nos experimentos de acordo com recomendações sobre otimizações de sistemas Linux para quando esses se comunicam por WANs [ESnet 2012], como usar o dobro do Produto Banda-Atraso (BDP) como tamanho dos buffers de transmissão e recepção, por exemplo.

[2]Implementações disponíveis em `https://github.com/mdediana/riak_kv` e `https://github.com/mdediana/riak_core`

Os experimentos foram executados no Grid'5000, uma plataforma para criação, execução e monitoramento de experimentos de sistemas paralelos e distribuídos [Grid5000 2013].

O trabalho usou experimentos fatoriais, que consistem da combinação de fatores em cada experimento que compõe o estudo [Jain 1991]. Quanto maior a quantidade de fatores e níveis em um estudo, mais recursos são necessários para sua execução. Por outro lado, alguns poucos fatores costumam explicar a maior parte dos efeitos na resposta. Por isso, uma seleção dos fatores mais influentes foi realizada pelo uso de experimentos 2^k. Esse tipo de experimento é realizado com apenas 2 níveis para cada fator, com um total de 2^k experimentos, onde k é a quantidade de fatores.

4. Parâmetros Fixados

Após definido o tipo de estudo experimental, foram levantados 33 parâmetros, dos quais 16 foram fixados por limitação de recursos ou por não serem foco do estudo. Vale notar que os parâmetros de LAN referem-se aos disponíveis no aglomerado, enquanto do de WAN foram emulados. Os parâmetros fixados e seus respectivos valores foram:

- **Aglomerado:** Os experimentos usaram o aglomerado *sol* no Grid'5000. Os nós desse aglomerado possuem CPU AMD Opteron 2218 2.6 GHz, 4 GB de memória e placa de rede de 1 Gb/s.
- **Mecanismo de armazenamento:** Ao usar memória como mecanismo de armazenamento evitou-se ter que considerar os efeitos de disco, cache de disco e a interação entre cache de disco e quantidade de memória, portanto os únicos efeitos de E/S observados foram devidos à rede.
- **Capacidade dos centros de processamento de dados:** Os centros de processamento de dados tinham a mesma capacidade – mesma quantidade de nós por centro e nós com a mesma configuração de hardware.
- **Algoritmo de particionamento das chaves:** O algoritmo padrão do Riak foi usado (espalhamento consistente).
- **Fator de replicação (N):** 3 é o valor que resulta em um balanço razoável entre desempenho, disponibilidade e durabilidade em aplicações reais [DeCandia et al. 2007].
- **Limiar de migração (para consistência na linha do tempo):** 3 é o valor padrão do PNUTS [Cooper et al. 2008].
- **Interface de acesso:** HTTP, por simplicidade de uso.
- **Nível de log:** WARN, já que experimentos exploratórios mostraram perda de desempenho quando o nível de log estava em INFO.
- **Configuração de hardware dos dispositivos de rede intermediários:** O único dispositivo de rede era um comutador FastIron Super X. Testes mostraram que não existiam gargalos no comutador mesmo nos experimentos com maior consumo de banda.
- **Topologia da rede:** Estrela, a única topologia de rede disponível no Grid'5000.
- **Largura de banda da LAN:** 1 Gb/s era a largura de banda das placas de rede dos nós do aglomerado.
- **Latência da LAN:** 167 μs, latência do aglomerado, medida com ping com 60 amostras espaçadas em 5 s.
- ***Jitter* da LAN:** 90 μs, *jitter* do aglomerado medido como a latência.

- **Largura de banda da WAN:** 100 Mb/s, baseado em estudo informal citando que essa largura de banda é comumente observada entre zonas de disponibilidade do AWS [Pujol 2012].
- **Quantidade de enlaces de WAN:** 1, o que resulta em dois centros de processamento de dados usados nos experimentos.
- **Taxa de chegada de requisições:** 15 operações/s para cada thread de cada instância do *benchmark*.

Com isso, ainda restavam 17 candidatos a fatores. Desses, 3 constituíam o modo.

5. Fator Modo

Três fatores receberam um tratamento diferente ao longo do experimento: modelo de consistência, configuração de replicação (para consistência em momento indeterminado) e versão requisitada nas leituras (para consistência na linha do tempo). Isso foi feito pois as combinações entre esses fatores definem configurações do sistema de armazenamento que resultam em proporções de requisições locais e remotas diferentes. Assim, esses fatores foram tratados como um único fator chamado modo. Os modos adotados foram:

- *ind1*: Consistência em momento indeterminado com $W = 1$ e $R = 1$
- *ind2*: Consistência em momento indeterminado com $W = 2$ e $R = 1$
- *lt_qqer*: Consistência na linha do tempo com leituras de qualquer versão
- *lt_rec*: Consistência na linha do tempo com leituras da versão mais recente

Considerando que o fator de replicação foi fixado em 3 e havia ao menos uma réplica em cada centro de processamento de dados, duas situações eram possíveis com relação à localização das réplicas de um objeto: uma local e duas remotas ou vice-versa. Dado isso, o modo *ind1* resultava em leituras e escritas locais. O modo *ind2* resultava em leituras locais e metade das escritas local e a outra metade remota. O modo *lt_qqer* resultava em leituras locais e a quantidade de escritas dependente da localidade dos acessos. Finalmente, o modo *lt_rec* resultava tanto em leituras quanto escritas dependentes da localidade.

Esses modos implicam em trocas além de desempenho e consistência. A principal é durabilidade, que para *ind2* é mais alta do que para os outros casos, em que a confirmação de escrita de uma única réplica é suficiente.

Ainda restavam 14 candidatos a fatores, quantidade grande para o estudo final. Para reduzir essa quantidade, uma triagem de fatores com experimentos 2^k foi realizada.

6. Triagem dos Fatores

Uma abordagem para a seleção dos fatores seria agrupar todos os candidatos a fatores em um único projeto de experimentos 2^k. O problema é que mesmo com apenas dois níveis por fator, a quantidade final de experimentos seria proibitiva.

A opção adotada então foi dividí-los em grupos menores de fatores relacionados e realizar estudos para cada grupo. Com isso, perdeu-se a comparação entre fatores de grupos diferentes e suas interações. Mas como a maioria dos fatores se mostrou pouco influente em seus grupos, como se vê nas subseções a seguir, essa abordagem não apresentou ameaça à validade.

A maioria dos fatores era suscetível a interações com fatores de rede. A latência da WAN em particular havia se mostrado muito influente em estudos exploratórios, fato confirmado posteriormente pelo estudo para fatores de rede. Dado isso, a abordagem adotada foi usar latência como representante da WAN quando necessário.

Existiram casos onde as respostas de todos os experimentos de um estudo eram semelhantes, independentemente dos níveis. Para tratar esses casos, também foram calculados os coeficientes de variação (CVs) das respostas para estimar qual a influência daquele conjunto de fatores e interações como um todo. Assim, um CV baixo indicava que nenhum dos fatores em questão eram influentes.

Para a consistência na linha do tempo, apenas a inserção de objetos na etapa de carga não era suficiente para que o sistema operasse no seu estado estacionário durante os experimentos. Isso porque, ao final da carga, cada objeto no banco de dados tinha recebido apenas um acesso de cada centro de processamento de dados, nenhuma réplica mestre teria migrado por efeito da localidade até esse momento. Por isso, foi necessária uma etapa de aquecimento do sistema após a carga dos dados antes de cada experimento de acordo com a localidade do experimento.

Foram realizados 4 estudos intermediários, descritos nas subseções seguintes. Sempre que modo e localidade precisaram ser fixados, eles o foram respectivamente em *lt_rec* e 50%, valores que resultam em uma quantidade balanceada de leituras e escritas locais e remotas. A latência, quando necessária, foi fixada em 100 ms. Nos resultados, os percentis 10 e 90 representam respectivamente requisições locais e remotas.

6.1. Fatores de tamanho do sistema e *benchmark*

As quantidades de nós de sistema e de instâncias do *benchmark* não só influenciavam as respostas, como também influenciavam questões operacionais ligadas à reserva de nós – pelas regras do Grid'5000, quanto maior a quantidade de nós reservada, menor é o tempo de uso permitido. Portanto, um estudo foi feito para definir a influência desses fatores. Os níveis usados foram os seguintes, sendo o valor entre parênteses o identificador do fator, usado nas tabelas com o resultado:

- Quantidade de nós do sistema (N): 8 e 16
- Quantidade de instâncias do *benchmark* (B): 2 e 4
- Quantidade de threads em cada instância do *benchmark* (T): 32 e 64

O resultado do estudo está na Tabela 1. O tamanho do sistema teve a maior influência nos resultados e as quantidades de instâncias do *benchmark* e de threads não foram desprezíveis, ainda mais ao se considerar as interações entre elas. Apesar disso, esses fatores foram desconsiderados devido ao excesso de fatores.

Operação	Percentil	N	B	T	NB	NT	BT	NBT
leitura	10	30	18	22	10	8	7	4
leitura	90	65	13	15	3	4	0	0
escrita	10	96	2	1	0	0	0	0
escrita	90	65	15	13	3	3	0	0

Tabela 1. Estudo fatores de tamanho do sistema.

Dados os resultados, os valores fixados foram:

- Quantidade de nós do sistema: 16
- Quantidade de instâncias do *benchmark*: 4
- Quantidade de threads em cada instância do *benchmark*: 32

Esses valores foram selecionados pois resultaram em uma configuração "leve", evitando gargalos de rede e não sobrecarregando o sistema. Apesar de desejável, uma quantidade maior de nós implicaria em falta de homogeneidade do hardware e dificuldades operacionais devido às regras do Grid'5000.

6.2. Fatores de banco de dados

Um estudo foi feito para dimensionar o tamanho do banco de dados, que afetava o uso de memória e de banda nos nós. Os níveis usados foram os seguintes:

- Quantidade de objetos armazenados (Q): 64.000 e 256.000
- Tamanho dos objetos armazenados (T): 100 e 10.000 bytes

O resultado do estudo está na Tabela 2. A quantidade de objetos não afetou o desempenho do sistema. O tamanho dos objetos não afetou o desempenho das requisições remotas, mas com relação às requisições locais apareceu com 100% de influência. Apesar disso, o CV das requisições locais indicava que sua influência não era tão grande – 19% para leituras e 16% para escritas.

Operação	Percentil	Q	T	L	QT	QL	TL	QTL
leitura	10	0	100	0	0	0	0	0
leitura	90	0	0	100	0	0	0	0
escrita	10	0	100	0	0	0	0	0
escrita	90	0	0	100	0	0	0	0

Tabela 2. Estudo para fatores de banco de dados, L é a latência.

Dados os resultados, os valores fixados foram:

- Quantidade de objetos armazenados: 128.000
- Tamanho dos objetos armazenados (bytes): 500

O tempo de aquecimento dependia da quantidade de objetos armazenados, portanto quanto menor essa quantidade, mais rápida era a execução dos experimentos. Por outro lado, um número muito pequeno resultaria em excesso de conflitos. No caso do tamanho dos objetos armazenados, o valor foi escolhido baseado em estudo dos sistemas de caching no Facebook, que relata que 90% dos objetos são menores do que 500 bytes [Atikoglu et al. 2012].

6.3. Fatores de rede

Dado o objetivo do trabalho, o estudo para fatores de rede era um dos mais importantes da etapa de seleção de fatores. Os níveis usados foram os seguintes:

- Latência da WAN (L): 100 e 300 ms
- *Jitter* da WAN (V): 1 e 60%

- Taxa de perda de pacotes na WAN (P): 0,01 e 0,3%
- Taxa de duplicação de pacotes na WAN (D): 0,05 e 5%
- Taxa de reordenação de pacotes na WAN (O): 0,05 e 5%
- Variante do TCP (T): CUBIC e H-TCP

Os níveis da latência foram baseados em um estudo que relata as latências entre as regiões dos Amazon Web Services [Sovran et al. 2011], no qual a menor latência foi 82 ms entre os centros de processamento de dados em cada costa dos EUA e a maior foi 277 ms entre Irlanda e Singapura.

O projeto PingER serviu de base para os outros fatores [PingER 2013]. Para janeiro de 2013, ele mostra uma média de latência de 238,062 ms com desvio padrão de 142,996 ms, o que resulta em *jitter* de 60%. Os 11 meses anteriores apresentavam valores semelhantes. A mediana da taxa de perda de pacotes do último ano foi 0,178%. A média da taxa de duplicação de pacotes em janeiro de 2013 foi 0,006%. Os valores usados no experimento para duplicação e reordenação foram maiores do que os observados pelo PingER, mas mesmo assim não influenciaram a resposta.

Tanto H-TCP quanto CUBIC foram projetados com foco em redes com largura de banda e latências grandes (BDP alto) e foram escolhidos pois são citados nas referências sobre otimizações da pilha TCP para WANs [ESnet 2012].

No emulador de rede, a latência define o mínimo e o *jitter* o máximo a que ela pode chegar. Por exemplo, ao fazer a configuração de 100 ms de latência e 60% de variação, o emulador gera valores entre 100 ms e 160 ms. Os valores gerados obedeciam a distribuição normal dentro da faixa de latência especificada.

O resultado do estudo está na Tabela 3, em que colunas das interações entre fatores com todas as células menores que 1% foram suprimidas por uma questão de espaço. As respostas das requisições locais apresentaram CVs de 1%, portanto as respectivas linhas também foram suprimidas – o que indica que a WAN não afeta requisições locais.

Operação	Percentil	L	V	P	D	O	T	LV
leitura	90	72	21	1	0	0	0	6
escrita	90	69	23	1	0	0	0	6

Tabela 3. Estudo para fatores de rede.

A latência, o *jitter* e a interação de primeira ordem entre eles responderam pela quase totalidade dos resultados. Assim, os níveis escolhidos para esses fatores nos experimentos foram:

- Latência da WAN (ms): 0, 100, 200 e 300
- *Jitter* da WAN (%): 0 e 60

Níveis nulos de latência e *jitter* equivalem a ter todo o sistema operando em uma LAN. Os resultados obtidos para esses casos foram usados como auxílio na interpretação dos resultados, mas não foram considerados no estudo final.

Os valores fixados dos fatores desconsiderados foram:

- Taxa de perda de pacotes na WAN (%): 0

- Taxa de duplicação de pacotes na WAN (%): 0
- Taxa de reordenação de pacotes na WAN (%): 0
- Variante do TCP: CUBIC

6.4. Fatores de carga de trabalho

Este era um dos estudos mais importantes juntamente ao estudo para fatores de rede. Os níveis usados foram os seguintes:

- Relação leitura/escrita (R): 2:1 e 10:1
- Popularidade dos objetos (P): uniforme (a taxa de chegada de requisições média para cada objeto é a mesma) e concentrada (a taxa de chegada segue uma distribuição Pareto)
- Localidade (X): 50% (sem localidade) e 90% (90% dos acessos para determinado objeto vindo de um centro de processamento de dados e 10% do outro)

Como os modos possuem comportamentos diferentes para requisições locais e remotas, os experimentos foram executados para cada modo.

A análise da relação leitura/escrita e localidade não usou percentis, mas sim a média do tempo de resposta de todas as requisições (leituras e escritas). Isso porque o primeiro fator diz respeito à composição entre leituras e escritas e o segundo altera a composição entre requisições locais e remotas, portanto esses fatores não fazem sentido nos percentis separados por tipo de requisição. Por exemplo, com localidade de 50% percebe-se que o percentil 70 representa requisições remotas, enquanto com localidade de 90% o mesmo percentil representa requisições locais. Se a análise fosse feita por percentis, essa informação se perderia e localidade nunca teria influência.

O resultado do estudo está na Tabela 4, em que colunas com todas as células menores do que 5% foram suprimidas por uma questão de espaço. Os resultados para requisições locais apresentaram CVs em torno de 2% para todos os modos, o que indica que nenhum dos fatores influencia requisições locais.

Modo	R	X	P	L	RX	RL	XL	PL	XPL
ind1	19	12	2	31	0	2	6	6	8
ind2	50	0	0	39	0	11	0	0	0
lt_qqer	25	30	0	19	9	6	8	0	0
lt_rec	0	53	0	34	0	0	13	0	0

Tabela 4. Estudo para fatores de carga de trabalho, L é a latência.

Como esperado, localidade e latência influenciaram as respostas em geral. O impacto de popularidade dos objetos é praticamente nulo. Apesar de alguns modos aparentemente terem sido impactados pela relação leitura/escrita, esse impacto foi consequência da relação entre requisições locais e remotas. Para *ind1*, tanto leituras quanto escritas são locais e a relação leitura/escrita e suas interações com outros fatores impacta pouco esse modo. Para *lt_rec*, leituras e escritas são locais ou remotas dependendo da localidade e a relação leitura/escrita não impacta esse modo. Para *ind2*, todas as leituras são locais e metade das escritas é remota, portanto quando a relação leitura/escrita muda, a relação entre requisições locais e remotas muda proporcionalmente – como esperado, esse modo

é impactado pela relação leitura/escrita. A mesma observação vale para *lt_qqer*, que tem todas as leituras locais e escritas dependendo da localidade, e também sofre impacto da relação leitura/escrita. A relação leitura/escrita provavelmente sofreria impacto caso o mecanismo de armazenamento fosse disco em vez de memória, dado que escritas seriam afetadas pelo tempo de escrita no disco, enquanto leituras poderiam ser mais rápidas pois parte delas seriam servidas a partir do cache de disco.

Assim, apenas localidade foi selecionada como fator:

- Localidade (%): 50 e 90

Os valores fixados dos fatores desconsiderados foram:

- Relação leitura/escrita: 2:1
- Popularidade dos objetos: uniforme

7. Estudo Final

O estudo final consistiu de um total de 64 experimentos, com os fatores selecionados nos estudos 2^k, apresentados na Tabela 5. Os experimentos usaram uma quantidade de amostras tal que o nível de confiança fosse 99% e a exatidão fosse 1% [Jain 1991]. O único caso com número de amostras menor que o necessário foi *lt_rec* – nesse caso, o mesmo nível de confiança foi adotado e a exatidão foi 2%. Duas replicações do estudo foram feitas para estimar a variabilidade dos experimentos. A média dos CVs dos experimentos foi 1% para leituras e 0,8% para escritas.

Fator	Níveis	Total de níveis
Modo	*ind1*, *ind2*, *lt_qqer* e *lt_rec*	4
Latência da WAN (ms)	0, 100, 200 e 300	4
Jitter da WAN (%)	0 e 60	2
Localidade	50% e 90%	2

Tabela 5. Fatores e níveis do estudo final.

Os resultados para latências de 100 ms, 200 ms e 300 ms apresentam o mesmo comportamento, portanto a opção foi fazer a análise para um deles apenas. O *boxplot* para latência de rede de 200 ms está na Figura 1. Os casos em que a caixa não aparece indicam que todas as requisições delimitadas pelos bigodes do *boxplot* eram locais. A mesma análise foi feita com carga máxima em vez de 15 operações/s por thread. Como a Figura 2 mostra, todos os modos apresentaram um aumento nos tempos de resposta, mas mantiveram o comportamento.

No caso de leituras e localidade de 50%, apenas *lt_rec* apresenta requisições remotas. Uma porção pequena dessas leituras tem tempo de resposta menor que a latência da rede, o que se explica pelo *jitter* de até 60%.

Para leituras e localidade de 90%, *lt_rec* é beneficiado, mas continua apresentando requisições remotas (valores atípicos no gráfico). No caso de escritas e localidade de 50%, *lt_rec* apresenta um desempenho um pouco melhor que *ind2* e *lt_qqer*. Isso ocorre pois o sistema está menos carregado para esse modo do que para os outros, já que as leituras de *lt_rec* são mais lentas. Esse fato é comprovado pela vazão dos modos *lt_rec*, *ev2* e *lt_qqer*, respectivamente, 594, 1072, 941 operações/s.

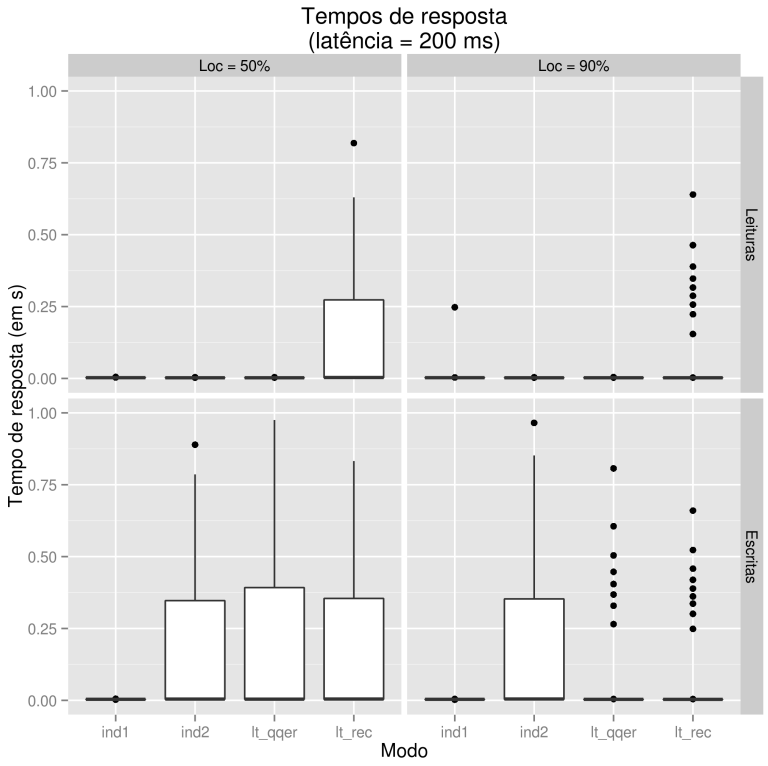

Figura 1. *Boxplot* dos tempos de resposta para 15 operações/s por thread.

Para escritas e localidade de 90%, *lt_rec* e *lt_qqer* apresentam melhor desempenho devido à localidade, mas ainda apresentam requisições remotas (valores atípicos no gráfico). Já *ind2* continua apresentando o mesmo perfil que para localidade de 50%.

Os modos apresentaram o comportamento esperado. A hipótese de que a consistência na linha do tempo é competitiva em termos de desempenho com a consistência em momento indeterminado se confirma para o caso em que a localidade é alta e principalmente quando leituras de "qualquer versão" são usadas.

8. Ameaças à Validade

Parâmetros foram fixados e fatores tidos como influentes foram desconsiderados. Com isso, estudos que usem outros valores para os parâmetros ou considerem outros fatores podem apresentar resultados diferentes. Isso vale particularmente para a quantidade de nós do sistema, que apareceu com influência relativamente alta. Além disso, outras faixas de níveis podem levar a outros resultados [Jain 1991].

Os experimentos consideram que todos os nós sempre operam sem falhas. Experimentos com o sistema operando em algum modo de falha (desde a falha de um nó até de um centro de processamento de dados inteiro) devem apresentar resultados diferentes dos obtidos nesse estudo, mas não foram realizados devido a limitações de recursos da pesquisa.

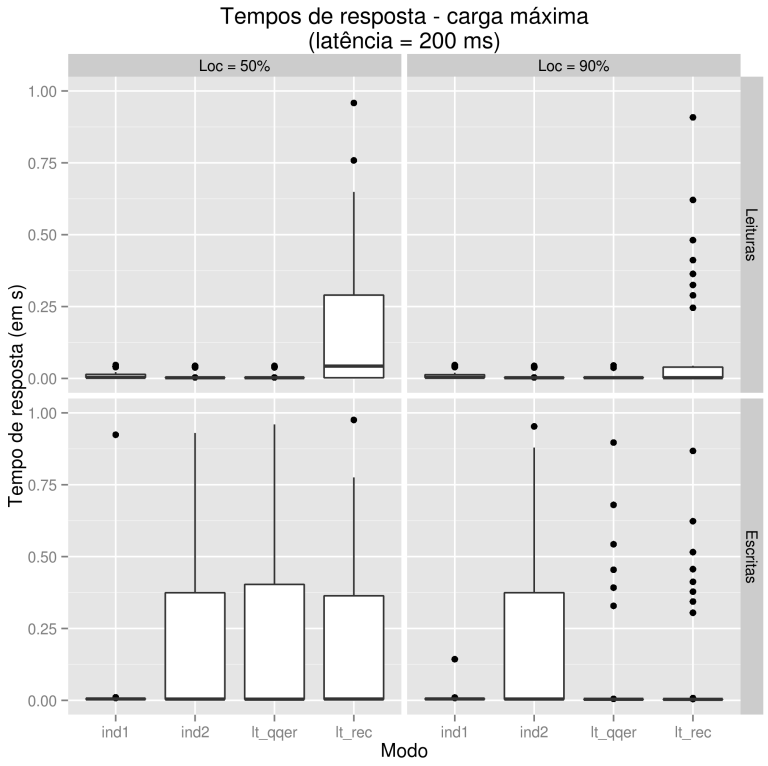

Figura 2. *Boxplot* dos tempos de resposta para carga máxima por thread.

9. Trabalhos Relacionados

Uma abordagem comum na literatura de sistemas distribuídos é a proposta de um novo conceito e a implementação de um sistema que use esse conceito seguida de uma análise de seu desempenho. Tanto o artigo sobre o Dynamo [DeCandia et al. 2007] quanto o sobre o PNUTS [Cooper et al. 2008] apresentam análises de desempenho, sendo que apenas o segundo faz uma análise baseada na carga de trabalho. Há uma segunda publicação com experimentos com o PNUTS que analisa o consumo de banda de diferentes políticas de replicação em uma WAN [Kadambi et al. 2011].

Muitos trabalhos apresentam análises de desempenho de sistemas de armazenamento que usam replicação sobre WANs. Na maior parte dos casos, o objetivo desses sistemas é provar outros conceitos além da eficiência do modelo de consistência escolhido por eles. O COPS usa consistência causal+, que é semelhante à consistência causal com algumas garantia a mais, e implementa transações [Lloyd et al. 2011]. O Scatter propõe uma arquitetura ao mesmo tempo escalável e com consistência forte [Glendenning et al. 2011]. O Windows Azure provê um sistema de armazenamento na nuvem com consistência forte [Calder et al. 2011]. O Megastore usa Paxos para implementar consistência forte [Baker et al. 2011]. Nenhum desses trabalhos apresenta comparações com outros sistemas ou com outros modelos de consistência. Como eles não usam uma aplicação para execução de testes ou ambiente em comum, é difícil fazer comparações a partir deles.

As diferentes configurações de consistência no Cassandra e sua disponibilidade e desempenho foram analisadas em [Beyer et al. 2011], que conclui que configurações

que oferecem consistência mais rígida apresentam pior desempenho. Desempenho e disponibilidade de replicação mestre-escravo e em cadeia são comparados em [van Renesse e Schneider 2004], cada uma delas usando consistência forte e consistência em momento indeterminado. Nenhum desses estudos leva em consideração operação sobre WAN nem diferentes cargas de trabalho.

Um estudo com proposta mais próxima da deste trabalho é a comparação usando diferentes cargas de trabalho feita entre Cassandra, HBase, PNUTS e MySQL particionado horizontalmente [Cooper et al. 2010]. Os resultados servem como uma comparação entre esses sistemas, mas dizem menos sobre seus modelos de consistência, dado que os sistemas apresentam arquiteturas e configurações diferentes. Além disso, os testes são feitos em uma LAN, não em uma WAN.

10. Conclusões

Este trabalho comparou o desempenho de um mesmo sistema de armazenamento usando dois modelos de consistência diferentes operando em uma WAN. Além disso, apresentou resultados sobre a influência de diferentes fatores e suas interações sobre o desempenho do mesmo sistema. A consistência na linha do tempo se mostrou competitiva com a consistência em momento indeterminado quando a localidade de escritas é alta e quando as leituras são de "qualquer versão".

As principais vantagens da consistência na linha do tempo sobre em momento indeterminado são a garantia de que as réplicas não são atualizadas com valores divergentes e a existência de uma réplica mestre que corresponde à versão mais recente. Um cenário interessante para seu uso é o caso em que a aplicação tolera inconsistências na maioria das leituras, mas em algumas poucas situações precisa da versão consistente. Já sua principal desvantagem é a indisponibilidade da réplica mestre impedir escritas e leituras consistentes. Outra desvantagem é que mesmo sendo competitiva, ela apresenta variabilidade relativamente alta nos tempos de resposta. Para aplicações em que os requisitos de tempos de resposta são dados pelo percentil 99,9, como é o caso da Amazon [DeCandia et al. 2007], a consistência na linha do tempo não é adequada.

11. Agradecimentos

Os experimentos apresentados neste trabalho foram realizados utilizando a plataforma para experimentos Grid'5000, desenvolvida pela ação de desenvolvimento ALADDIN INRIA com o apoio do CNRS, RENATER e várias universidades, bem como outros órgãos de financiamento.

12. Referências

Atikoglu, B., et al. (2012). Workload analysis of a large-scale key-value store. *SIGMETRICS Perform. Eval. Rev.*, 40(1):53–64.

Baker, J., et al. (2011). Megastore: Providing scalable, highly available storage for interactive services. Em *Proc. Conference on Innovative Data system Research (CIDR)*, pags. 223–234.

BashoBench (2013). Basho Bench. http://docs.basho.com/riak/latest/cookbooks/Benchmarking/. [Último Acesso em 01/02/2013.].

Beyer, F., et al. (2011). Testing the Suitability of Cassandra for Cloud Computing Environments Consistency, Availability and Partition Tolerance. Em *Proc. 2nd International Conference on Cloud Computing, GRIDs, and Virtualization*, pags. 86–91.

Calder, B., et al. (2011). Windows azure storage: a highly available cloud storage service with strong consistency. Em *Proc. 23rd ACM Symposium on Operating Systems Principles*, SOSP '11, pags. 143–157.

Cooper, B. F., et al. (2008). Pnuts: Yahoo!'s hosted data serving platform. *Proc. VLDB Endowment (PVLDB)*, 1(2):1277–1288.

Cooper, B. F., et al. (2010). Benchmarking cloud serving systems with ycsb. Em *Proc. 1st ACM symposium on Cloud computing*, SoCC '10, pags. 143–154.

DeCandia, G., et al. (2007). Dynamo: Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220.

ESnet (2012). Host Tuning. http://fasterdata.es.net/host-tuning/background/. [Último Acesso em 01/02/2013.].

Glendenning, L., et al. (2011). Scalable consistency in scatter. Em *Proc. 23rd ACM Symposium on Operating Systems Principles*, SOSP '11, pags. 15–28.

Grid5000 (2013). Grid'5000. http://www.grid5000.fr/. [Último Acesso em 01/02/2013.].

Jain, R. (1991). *The Art Of Computer Systems Performance Analysis*. Wiley.

Kadambi, S., et al. (2011). Where in the world is my data? *Proc. VLDB Endowment (PVLDB)*, 4(11):1040–1050.

Lloyd, W., et al. (2011). Don't settle for eventual: scalable causal consistency for wide-area storage with cops. Em *Proc. 23rd ACM Symposium on Operating Systems Principles*, SOSP '11, pags. 401–416.

PingER (2013). Ping End-to-end Reporting. http://www-iepm.slac.stanford.edu/pinger. [Último Acesso em 01/02/2013.].

Pujol, J. M. (2012). Having fun with Redis Replication between Amazon and Rackspace. http://3scale.github.com/2012/07/25/fun-with-redis-replication/. [Último Acesso em 01/02/2013.].

Riak (2013). Riak. http://wiki.basho.com/Riak.html. [Último Acesso em 01/02/2013.].

Sovran, Y., et al. (2011). Transactional storage for geo-replicated systems. Em *Proc. 23rd ACM Symposium on Operating Systems Principles*, SOSP '11, pags. 385–400.

van Renesse, R. e Schneider, F. B. (2004). Chain replication for supporting high throughput and availability. Em *Proc. 6th Symposium on Operating Systems Design & Implementation*, OSDI'04, pags. 7–7.

Vogels, W. (2009). Eventually consistent. *Communications of ACM*, 52(1):40–44.