Avaliação de um SGBD Replicado Usando Simulação de Redes*

L. Soares A. Sousa J. Pereira R. Oliveira A. Correia
L. Rocha F. Moura

Grupo de Sistemas Distribuídos, Dep. de Informática, Universidade do Minho
escada@lsd.di.uminho.pt

Resumo

A replicação de sistemas de gestão de bases de dados (SGBD) é um mecanismo fundamental para a fiabilidade de sistemas de informação. Em sistemas geograficamente distribuídos é ainda útil na recuperação de desastres e disponibilidade ubíqua de dados.

Uma técnica de replicação recentemente proposta é a *Database State Machine* (DBSM), que promete aliar fiabilidade a elevado desempenho tirando partido de sistemas de comunicação em grupo. A avaliação do desempenho desta técnica tem no entanto sido efectuada com redes de comunicação demasiado simples ou irrealistas e com uma carga não representativa.

Este artigo propõe uma avaliação rigorosa de uma concretização desta técnica de replicação, aliando um modelo de simulação realista de redes de comunicação com uma geração de carga efectuada de acordo com os padrões elaborados pelo *Transaction Processing Council* (TPC). Os resultados obtidos confirmam o interesse desta técnica em redes locais, mas mostram que o seu desempenho é condicionado pelas características da rede e da carga.

1 Introdução

A técnica de replicação de bases de dados com o nome Database State Machine (DBSM) [17] surgiu recentemente como uma alternativa atraente para conciliar fiabilidade e desempenho. Esta técnica é composta por três fases: *(i)* a execução local das transacções numa das réplicas de forma optimista; *(ii)* a difusão atómica

*Trabalho financiado pela FCT no âmbito do projecto ESCADA - POSI-CHS-33792-2000

das actualizações para todas as réplicas e *(iii)* um procedimento determinístico de certificação.

A execução das transacções é optimista pois cada réplica executa localmente a transacção sem qualquer sincronização com as outras réplicas, evitando assim a sobrecarga associada ao controlo de concorrência distribuído [10]. Seguidamente, o estado associado à execução da transacção é difundido com garantias de atomicidade e ordem total na entrega, utilizando primitivas de comunicação em grupo [6]. Por fim, o procedimento de certificação garante que as transacções que violem os pressupostos de serialização [4] sejam abortadas. A ordenação total das mensagens aliada ao determinismo do procedimento de certificação assegura um estado global coerente entre as várias réplicas do sistema.

O desempenho desta abordagem tem sido estudado com simulação [16] e com uma concretização no âmbito do PostgreSQL [14]. Os resultados obtidos com simulação são no entanto limitados pela simplicidade dos modelos utilizados para a rede e para os protocolos de comunicação. Por outro lado, a avaliação da concretização no âmbito do PostgreSQL [19] torna difícil comparar diferentes alternativas e fazer experiências em WAN ou com um número elevado de réplicas. Em ambas as situações a carga utilizada é pouco realista e como tal não representativa de situações reais.

Foi anteriormente [24] proposta uma forma de ultrapassar estas limitações, combinando um modelo realista para a rede [9] e para a carga a que a base de dados é submetida [25] com a concretização de DBSM num sistema de simulação centralizada [3]. O mesmo trabalho [24] mostra como o módulo de base de dados centralizada pode ser parametrizado para simular com precisão um servidor real.

Neste artigo, aumenta-se a complexidade do modelo parametrizando-o com uma WAN, de forma a estudar a viabilidade de utilizar a DBSM em grande escala. Em particular, numa primeira abordagem interessa sobretudo investigar o impacto da latência adicional, assumindo que exista largura de banda suficiente. As razões para proceder assim são várias. Em primeiro lugar, estima-se que a largura de banda necessária seja da mesma ordem de grandeza que em qualquer outra estratégia de replicação síncrona e directamente proporcional ao tamanho dos tuplos actualizados. De facto, ao contrário do que acontece com a latência, a limitação de largura de banda pode também, na prática, ser mais facilmente resolvida com um investimento adicional. Em contraste, existem motivos para considerar que o aumento da latência poderá ter impacto considerável: se a certificação distribuída demorar mais, permite que outras transacções concorrentes sejam executadas, or-

denadas, difundidas e certificadas. Sendo assim, as hipóteses de uma transacção ser rejeitada no procedimento de certificação aumentam significativamente e podem degradar o desempenho do sistema. É preciso no entanto quantificar este impacto em sistemas reais.

Em concreto, os resultados obtidos são comparados com os resultados com DBSM em LAN e sistemas multi-processador centralizados. Mostra-se assim, que o aumento da latência por si só tem um impacto negativo no comportamento da DBSM apenas quando carregado. Em segundo lugar, mostra-se que sendo a penalização de latência distinta para diferentes servidores, decorrente da própria rede e do protocolo de difusão atómica usado, poderá existir ainda uma injustiça relativa no tratamento de transacções executadas por diferentes servidores.

O artigo encontra-se estruturado da seguinte forma: a Secção 2 descreve a técnica de replicação de bases de dados utilizada. A Secção 3 apresenta a concretização da técnica de replicação e o sistema de simulação. A Secção 4 apresenta a instanciação do modelo de simulação para este artigo em particular. Os resultados são então apresentados e discutidos na Secção 5. A Secção 6 conclui o artigo.

2 Replicação da Base de Dados com DBSM

O sistema considerado, é composto por um conjunto de processos sequenciais comunicando entre si exclusivamente através da passagem de mensagens. Os processos ou são clientes ou servidores de bases de dados. Cada cliente comunica apenas com um servidor e efectua pedidos de execução de transacções. A cada transacção corresponde um conjunto de tuplos que são lidos e, eventualmente, um conjunto de tuplos que são escritos. A resposta devolvida por um servidor indica se a transacção foi executada com sucesso (*commit*) ou não (*abort*).

A replicação de bases de dados utilizando a técnica *Database State Machine* (DBSM) funciona da seguinte forma: cada transacção é executada de forma optimista por uma das réplicas utilizando mecanismos de controlo de concorrência locais. Depois da execução estar terminada, o conjunto de identificadores dos tuplos lidos (*read set*) e o conjunto de tuplos modificados (*write set*) são difundidos para as restantes réplicas através de uma primitiva de difusão atómica [6]. Quando a mensagem que os contém é entregue, é então desencadeado o procedimento de certificação que garante que a transacção é abortada se existir um conflito com alguma transacção concorrente que tenha já terminado. Um esquema representativo das entidades envolvidas neste processo pode ser encontrado na Figura 1.

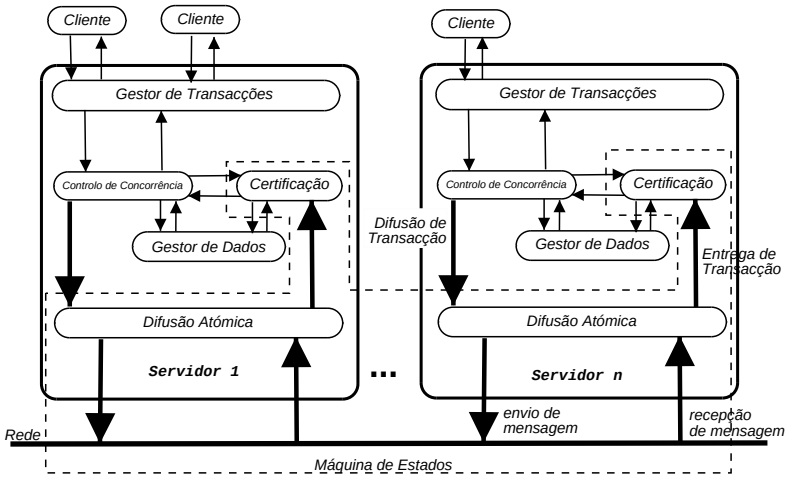

Figura 1: Modelo do Protocolo DBSM

Por exemplo, considerem-se quaisquer duas transacções concorrentes t e t', isto é, nem t precede t' nem t' precede t. Considere-se ainda que a certificação de t' precede a certificação de t. A transacção t é abortada durante a certificação se: *(i)* t' tiver sido certificada positivamente, actualizado a base de dados e terminado com *commit*; e *(ii)* algum dos tuplos lidos por t tiverem sido modificados por t'. Dada a ordenação das mensagens e o determinismo do procedimento de certificação, todas as réplicas chegam ao mesmo resultado [22], assegurando-se desta forma a coerência da base de dados.

Quando uma transacção lê um grande número de tuplos de dados durante a sua execução, por exemplo, todos os tuplos de uma tabela, o conjunto dos identificadores dos tuplos lidos pode atingir tamanhos incomportáveis, inviabilizando a sua transmissão. Este problema pode ser ultrapassado recorrendo a um passo adicional de comunicação [14]. A certificação é efectuada apenas pelo servidor que executou a transacção, que difunde posteriormente o resultado. Em alternativa, pode ser estabelecido um limite superior para a quantidade de tuplos de uma tabela que são enumerados. Quando este limite é ultrapassado, utiliza-se um identificador da própria tabela e não de cada um dos tuplos [24]. Desta forma, evita-se o passo adicional de comunicação, mas introduz-se a possibilidade de abortar transacções devido a falsos conflitos.

3 Modelo de Simulação

Nesta secção é descrito o ambiente de simulação bem como cada um dos componentes utilizados: *(i)* o módulo que simula um sistema de gestão de bases de dados; *(ii)* a concretização da DBSM; e *(iii)* das primitivas de comunicação em grupo. A descrição do simulador de rede é omitida pois foi desenvolvido separadamente [9].

3.1 Simulação Centralizada

Um sistema de simulação centralizado [3] permite combinar a concretização de módulos de *software* com componentes simulados, nomeadamente de outros módulos de *software*, do *hardware* e do ambiente. O sistema de simulação centralizada foi desenvolvido nas linguagens Java e C, sobre o sistema de simulação por eventos SSF [8].

A simulação centralizada utiliza um relógio virtual por cada processador simulado. A execução de eventos em módulos de simulação avança explicitamente o relógio ao agendar novos eventos. A execução de código real é cronometrada e o tempo resultante usado para actualizar também o relógio simulado. O sistema de simulação permite que componentes simulados possam agendar eventos de código real (por exemplo, a chegada de uma mensagem a um nó da simulação da rede desencadeia código da aplicação) e que o código real possa agendar eventos na simulação (por exemplo, que a aplicação possa enviar mensagens através de uma rede simulada).

Este método de simulação reproduz fielmente, com apenas um processador, a execução de um sistema real com diversos processadores idênticos [3], para os quais pode ser calibrado e validado recorrendo a medidas simples de desempenho. Torna-se deste modo possível estudar em detalhe o funcionamento de módulos de *software* sem ter que recorrer a modelos simplificados ou à utilização de recursos dispendiosos. Além disso, torna-se possível a observação global de propriedades em sistemas distribuídos e a injecção de faltas.

3.2 Base de Dados

Na simulação da base de dados, uma transacção é modelada como uma sequência de operações elementares. Uma operação consiste *(i)* na leitura de um tuplo de dados; *(ii)* processamento; ou *(iii)* na escrita de um tuplo de dados. Os recursos geridos pelo servidor de base de dados e pelos quais as transacções competem

são a largura de banda de acesso aos discos para leitura e escrita de tuplos, os mecanismos de exclusão mútua associados aos tuplos e o tempo de processamento total disponível. A modelação a este nível de abstracção de sistemas de gestão de bases de dados foi proposta anteriormente e aplicada com sucesso para diversos fins, nomeadamente para estudo de mecanismos de controlo de concorrência [2, 1].

O controlo de concorrência presente é efectuado com um modelo de *locking* simplificado em que uma transacção obtém todos os *locks* de uma só vez quando inicia a execução. Se não consegue obter algum deles, a transacção é inserida numa fila de espera. O modelo pode ainda ser configurado para simular um sistema de multi-versão, em que a obtenção de *locks* de leitura não se traduz nunca em bloqueio.

Cada cliente está associado a um servidor, ao qual vai submetendo transacções. As transacções são do tipo interactivo, isto é, entre cada duas operações existe interacção com o utilizador, modelada na simulação como um período de espera antes de submeter a próxima operação. Entre cada duas transacções submetidas por um mesmo cliente, existe de igual forma um período de espera entre o fim da transacção anterior e a submissão de uma nova transacção.

A sequência de transacções a serem submetidas por um cliente é gerada previamente de acordo com a carga e a base de dados desejada, em particular, quanto aos tuplos a ler e escrever por cada operação, tempo de espera entre operações, bem como ao tempo de processamento consumido. Os servidores de bases de dados atendem os pedidos de clientes. Quando um cliente submete uma transacção esta é executada, respeitando os tempos de espera, de processamento e consumindo os recursos correspondentes às operações especificadas.

Este modelo pode ser calibrado para reproduzir em detalhe o funcionamento de um sistema real, tal como descrito na Secção 4. A validação do sistema passa pela comparação de medidas de desempenho semelhantes efectuadas em ambos [24].

3.3 Protocolo de Terminação

O protocolo de terminação é executado sempre que uma transacção termina a execução e antes de executar o *commit*. Numa primeira fase, resume-se à utilização do protocolo de difusão atómica para o grupo de servidores. Esta operação implica a conversão da informação referente à transacção para um formato apropriado à sua transmissão. Em particular, a conversão dos identificadores dos tuplos de leitura e de escrita, cada qual um inteiro de 64 bits. Os valores dos tuplos escritos também são obtidos por forma a enviar na mensagem de certificação. O tamanho em

bytes ocupado por cada tuplo escrito é calculado com base no esquema da base de dados utilizada. Pretende-se desta forma gerar um tráfego na rede que represente fielmente a situação real. Para evitar o envio de grande quantidade de informação quando o número de tuplos lidos é grande, envia-se apenas o identificador da tabela correspondente.

Após a entrega da mensagem a cada uma das réplicas, é executado o procedimento de certificação que garante, quando terminado com sucesso, que não existem transacções concorrentes e já terminadas, que tenham executado operações que originem conflitos com a transacção avaliada. Informação adicional, contida na mensagem, permite também eliminar a informação relativa a transacções precedentes a qualquer transacção em execução actualmente. Desta forma, diminui-se o processamento necessário para determinar quais as transacções concorrentes e o espaço necessário para o seu armazenamento. Após a conclusão da certificação, a transacção é enviada para o gestor de dados para ser concluída com o envio da resposta ao cliente.

3.4 Protocolo de Difusão Atómica

O protocolo de difusão atómica utilizado [23] é concretizado em duas camadas: um protocolo de sincronismo virtual e um protocolo de ordem total [6]. O protocolo de sincronismo virtual funciona em duas fases. Em primeiro lugar, é feita uma difusão optimista das mensagens utilizando uma quantidade de largura de banda controlada. Esta fase pode utilizar difusão IP ou envio ponto-a-ponto, conforme as capacidades da rede disponível. Numa segunda fase, a reparação de perdas é assegurada através de mecanismos de confirmação negativa [18] e retransmissão com controlo por janela [7]. A mudança de configuração do grupo usa um protocolo baseado em consenso [21] e o armazenamento necessário para assegurar sincronismo virtual é gerido com um protocolo de detecção de estabilidade [11]. A ordem total é assegurada através da utilização de um processo coordenador [5, 13], que atribui um número de sequência a cada mensagem. Os restantes processos guardam e entregam as mensagens de acordo com a ordem dos números de sequência de cada uma. A utilização de sincronismo virtual assegura a correcção do protocolo ao proporcionar a eleição do processo coordenador.

4 Instanciação do Modelo

Esta secção descreve a instanciação do modelo de simulação descrito com o objectivo de comparar o desempenho da DBSM em WAN com um sistema semelhante em LAN. Esta instanciação consiste na escolha do tráfego, dos parâmetros do servidor e da rede.

A carga usada é gerada de acordo com o especificado pela norma TPC-C [25]. Com um padrão de acesso OLTP (*On-Line Transaction Processing*), esta norma de avaliação de desempenho simula o processamento de pedidos de produtos num contexto de revenda, onde existem diversos armazéns geograficamente dispersos. O número de armazéns, bem como o tamanho inicial das tabelas, está relacionado com o número de clientes que são configurados. Para os resultados apresentados escolheram-se configurações com 20 e 100 clientes, correspondendo respectivamente a uma carga moderada e a uma carga substancial do sistema em termos de capacidade de processamento.

As transacções são de cinco tipos distintos e executadas segundo uma distribuição probabilística. As transacções assim como a probabilidade de ocorrência durante uma execução e a sua descrição são as seguintes:

New Order (44%) Acrescenta uma nova encomenda ao sistema.

Payment (44%) Actualiza o crédito do cliente e as estatísticas dos armazéns, distrito e warehouse.

Order Status (4%) Devolve a última encomenda de um cliente.

Delivery (4%) Regista uma entrega de produtos.

Stock Level (4%) Determina os produtos vendidos recentemente cujo o nível de stock se encontra abaixo de um limite mínimo determinado.

4.1 Carga

Além da identificação dos tuplos a serem lidos e escritos, que podem ser directamente gerados a partir da especificação, é ainda necessário associar a cada transacção um tempo de processamento. No entanto, este depende do SGBD, do tamanho das tabelas e do *hardware* utilizado. Para obter valores realistas, executaram-se estas transacções numa base de dados real (PostgreSQL [19]) numa máquina de referência, Pentium III a 1GHz, e mediu-se o tempo de processamento consumido por

	Tempo de Processamento (ms)		Número de Tuplos	
	20 clientes	100 clientes	Lidos	Escritos
delivery	332.24	434.31	11161	187
neworder	25.25	35.87	21	21
orderstatus-01	21.51	26.80	9	0
orderstatus-02	3.54	5.10	6	0
payment-01	23.65	31.92	4	3
payment-02	5.56	10.03	3	2
stocklevel	20.24	12.52	19	0

Tabela 1: Tempos de processamento e número de tuplos lidos/escritos por transacção.

cada uma delas para cada configuração da base de dados. Tornou-se assim, possível determinar qual a distribuição a usar para gerar os tempos de processamento a consumir na simulação. Para simplificar este procedimento, as transacções *Payment* e *Order Status* são sub-divididas em dois tipo distintos, pois originalmente resultam em distribuições bimodais. As médias das distribuições obtidas, assim como o número médio de tuplos lidos e escritos são apresentados na Tabela 1.

4.2 Servidores

A configuração de um servidor de base de dados consiste na escolha da política local de controlo de concorrência a utilizar, da taxa de acerto na *cache*, do acesso a tuplos de dados, do custo de cada acesso a disco (tanto para leitura como para escrita), e ainda do limite superior para a enumeração de tuplos durante a certificação.

A política de controlo de concorrência escolhida é o *locking*. Uma vez que na simulação todos os tuplos são reservados em simultâneo, evitam-se situações de *deadlock* e como tal não são nunca abortadas transacções num servidor centralizado. Sendo a política mais estrita, evidencia os efeitos de quaisquer conflitos que surjam com a DBSM.

Para uma configuração realista dos restantes parâmetros, recorreu-se de novo ao sistema de referência. Este está configurado com 1 Gbyte de RAM. O armazenamento de dados é feito em 4 discos SCSI de 36 Gbytes com uma configuração RAID-5, ligados através de FibreChannel. O sistema de ficheiros utilizado para conter os dados e executáveis é o *ext3* no sistema operativo *Linux* (versão 2.4.21-pre3).

Com este sistema e com a ordem da carga utilizada para simulação, observa-se

que a taxa de acerto na *cache* é muito elevada, pelo que se considera como 100% no modelo de simulação. A capacidade do disco fica desta forma inteiramente dedicada para escrita. Esta capacidade é caracterizada utilizando o teste IOZone [12], executando escritas aleatórias e síncronas de páginas de 4Kbytes a partir de um número variável de processos. O valor final obtido foi de 9.486MB/s de largura de banda e 6.5 ms de latência.

Finalmente, o número máximo de identificadores de tuplos de uma mesma tabela a serem enumerados foi de 150 tuplos. Este valor é suficiente para a grande maioria das transacções observadas e exclui aquelas que originariam valores incomportáveis.

4.3 Rede e Protocolos de Difusão

Para os testes a efectuar definiram-se duas redes distintas: a primeira é uma rede local LAN com os servidores inter-ligados por uma rede que apresenta uma taxa de transmissão a 100 Mbps. Neste caso, o protocolo de difusão é configurado para utilizar difusão IP na sua etapa inicial.

A WAN utilizada é uma tentativa de aproximação à infra-estrutura actual de rede da RCTS2 [20]. Em detalhe, 1 servidor é inserido na rede do Campus da Universidade do Minho; 1 servidor num Campus similar ao da Universidade do Minho, ligado a Lisboa; 2 servidores internacionais. O número total de servidores é pois o mesmo considerado na rede local, e a largura de banda dos links entre Braga-Porto e Porto-Lisboa é de 34 Mbps. No caso dos links internacionais, estes encontram-se conectados ao nó em Lisboa através de um link com 1 Gbps de largura de banda. Na Figura 2 pode-se encontrar o esquema que representa cada uma das subredes em que estão inseridos os servidores. Neste caso, o protocolo de difusão é configurado para utilizar apenas comunicação ponto-a-ponto.

Em ambas as situações, não é introduzido tráfego adicional na rede. Desta forma existe apenas em circulação o tráfego associado à gestão da rede, o tráfego de controlo do protocolo de difusão atómica e os dados propriamente ditos. Verificou-se ainda que de facto não surgiam situações de congestão durante a realização das experiências.

5 Resultados

Esta secção apresenta e comenta os resultados obtidos para o sistema como um todo, para avaliar o seu desempenho global, bem como para cada um dos servi-

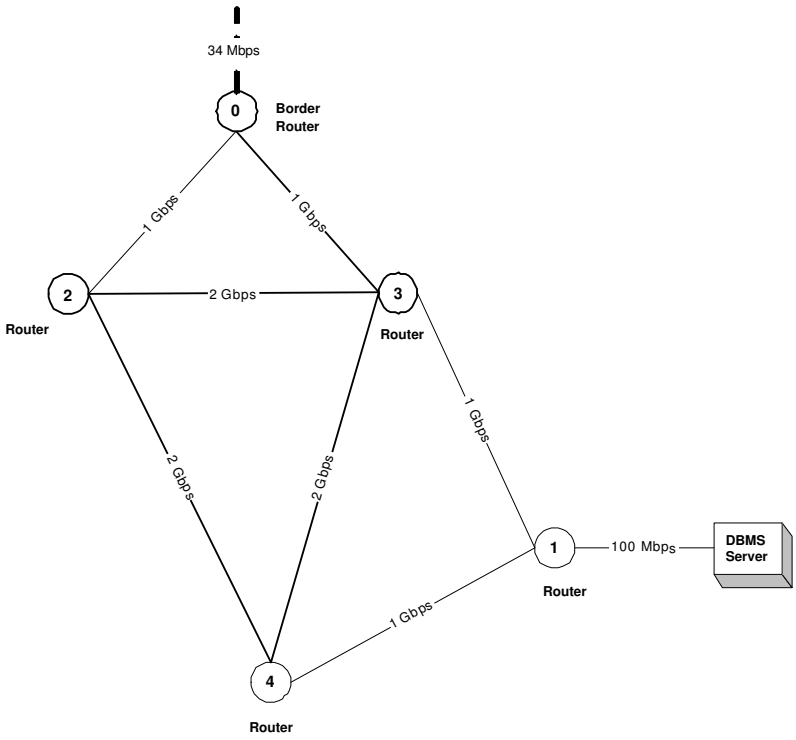

Figura 2: Esquema das redes internas modeladas

dores considerado individualmente, para avaliar a justiça relativa desta técnica de replicação.

5.1 Resultados Globais

A comparação do desempenho da DBSM em LAN e em WAN é feita utilizando dois tipos distintos de carga, correspondendo a 20 e 100 clientes, respectivamente, com 5 e 25 clientes em cada um dos servidores. Cada um dos servidores é configurado com apenas um processador simulado. As medidas de desempenho são efectuadas para cada um dos tipos de transacção, bem como para o total. Para efeitos de comparação, são também obtidos resultados com servidores centralizados de 1 e 4 processadores. Idealmente, o desempenho dos sistemas distribuídos será semelhante àquele do sistema centralizado com 4 processadores, que dispõe de capacidade de cálculo semelhante.

A Tabela 2 apresenta o número médio de re-submissões necessárias para executar com sucesso 100 transacções. Os resultados obtidos com os servidores centralizados são naturalmente sempre nulos, uma vez que é usado controlo de concor-

	20 Clientes				100 Clientes			
	1 CPU	4 CPU	LAN	WAN	1 CPU	4 CPU	LAN	WAN
delivery	0.00	0.00	2.61	30.52	0.00	0.00	31.68	266.39
neworder	0.00	0.00	0.54	4.70	0.00	0.00	0.33	3.06
payment-01	0.00	0.00	3.55	7.55	0.00	0.00	1.80	5.03
payment-02	0.00	0.00	2.29	4.21	0.00	0.00	1.54	4.38
READ-WRITE	0.00	0.00	1.81	6.66	0.00	0.00	2.72	18.90
READS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TOTAL	0.00	0.00	1.65	6.05	0.00	0.00	2.49	17.27

Tabela 2: Numero médio de re-submissões necessárias por 100 transacções.

	20 Clientes				100 Clientes			
	1 CPU	4 CPU	LAN	WAN	1 CPU	4 CPU	LAN	WAN
delivery	1204.38	1182.21	1172.83	1615.10	1281.40	1267.47	1490.29	7447.87
neworder	142.29	109.23	105.07	216.88	424.00	235.61	305.53	742.59
payment-01	139.65	93.32	89.87	229.65	436.47	176.45	287.34	766.68
payment-02	85.36	75.37	71.32	174.31	476.58	188.61	290.91	766.60
READ-WRITE	174.66	142.24	137.26	269.91	469.55	249.62	343.27	1021.97
orderstatus-01	42.47	41.19	47.52	41.81	151.74	46.80	49.76	61.96
orderstatus-02	41.82	23.44	24.16	23.22	120.28	25.10	50.31	49.28
stocklevel	44.91	50.22	40.86	44.68	110.34	42.82	47.25	74.28
READ-ONLY	43.34	40.54	38.78	38.45	123.27	40.93	48.42	68.46
TOTAL	162.29	132.58	128.07	248.66	447.59	235.69	323.28	946.78

Tabela 3: Latência média (ms).

	20 Clientes				100 Clientes			
	1 CPU	4 CPU	LAN	WAN	1 CPU	4 CPU	LAN	WAN
delivery	0.10	0.10	0.10	0.10	0.46	0.47	0.50	0.47
neworder	1.08	1.08	1.08	1.08	6.13	6.19	6.11	5.68
payment-01	0.75	0.74	0.75	0.72	3.40	3.36	3.42	3.42
payment-02	0.55	0.55	0.54	0.54	2.43	2.43	2.43	2.23
READ-WRITE	2.47	2.48	2.48	2.44	12.37	12.42	12.42	11.79
orderstatus-01	0.09	0.09	0.09	0.08	0.24	0.24	0.24	0.21
orderstatus-02	0.06	0.06	0.06	0.06	0.13	0.15	0.16	0.13
stocklevel	0.11	0.11	0.11	0.11	0.48	0.51	0.52	0.67
READ-ONLY	0.25	0.26	0.26	0.25	0.83	0.90	0.92	1.01
TOTAL	2.71	2.74	2.73	2.69	13.21	13.31	13.32	12.80

Tabela 4: Transacções por segundo.

rência local baseado em *locking*. Os resultados relativos a transacções apenas de leitura não são mostrados, uma vez que não necessitando certificação, são também nulos.

Exceptuando a transacção *Delivery*, o número de transacções abortadas é razoavelmente baixo, mesmo com a carga mais intensa obtida com 100 clientes simultâneos. Pelo contrário, os números associados à transacção *Delivery* apenas são satisfatórios com a menor carga e em rede local. Esta situação é justificada pelo comportamento desta transacção, que efectua a leitura de um grande número de tuplos de uma tabela numerosa e tem um processamento significativamente mais demorado que as restantes (ver Tabela 1). É portanto, altamente susceptível a conflitos criados por transacções concorrentes, especialmente quando a sua execução é atrasada devido à concorrência de outras transacções no mesmo servidor (no caso dos 100 clientes) ou de servidores distintos (no caso da WAN). Este facto é duplamente prejudicial, uma vez que sendo a transacção que consome mais recursos é também a que mais vezes tem que ser re-executada. No entanto, é também uma transacção relativamente rara, pelo que o seu impacto no desempenho final não é catastrófico.

A Tabela 3 apresenta a latência média de processamento de transacções tal como vista pelos clientes. No caso de ser necessária a re-submissão de uma transacção abortada pela certificação, o intervalo de tempo considerado é aquele desde a primeira submissão até a resposta final com sucesso.

Nos casos em que o número de re-submissões é baixo e em rede local, observa-se que existe uma vantagem na utilização do sistema distribuído em relação ao sistema centralizado com apenas um processador, aproximando-se até do comportamento ideal representado pelo sistema centralizado com 4 processadores. Esta vantagem é tanto maior quanto a carga existente no sistema. Em WAN, existe um custo adicional em latência introduzido pelo procedimento de certificação distribuído que é evidente. Note-se que esta penalização em termos de latência não se traduz em aumento da utilização de recursos dos servidores.

Finalmente, nos casos em que o número de re-submissões é elevado, independentemente da latência da rede, existe uma penalização significativa em termos de latência. O acréscimo significa ainda uma utilização adicional de recursos dos servidores.

Esta comparação entre a rede WAN por um lado e a rede LAN e servidores centralizados por outro, não é inteiramente justa. De facto, a utilização de uma das soluções locais para suportar um sistema geograficamente distribuído incorreria em

	Servidor 1	Servidor 2	Servidor 3	Servidor 4
LAN	5.63	6.56	6.74	6.74
WAN	7.61	13.65	14.94	14.94

Tabela 5: Re-submissões por servidor.

	Servidor 1	Servidor 2	Servidor 3	Servidor 4
LAN	115.51	109.73	134.27	134.27
WAN	123.05	181.40	255.28	255.28

Tabela 6: Latência médias por servidor.

latência adicional entre os próprios clientes e os servidores.

A Tabela 4 apresenta o número médio de transacções processadas por segundo conseguido por cada um dos sistemas. Sob esta perspectiva e independentemente da carga, não é significativo o impacto da utilização de uma WAN no desempenho global do sistema.

Em resumo, a DBSM é globalmente atraente mesmo em WAN. No entanto, isto acontece apenas porque as transacções cujo desempenho se degrada rapidamente representam apenas 4% do total e os recursos dos sistemas distribuídos são suficientes para acomodar a carga adicional resultante de transacções re-submetidas. Caso contrário, será interessante considerar soluções alternativas para o tratamento de transacções com estas características.

5.2 Resultados por Servidor

As Tabelas 5 e 6 apresentam respectivamente o número re-submissões e a latência em separado para cada um dos servidores envolvidos. Torna-se então evidente que nem todos são igualmente afectados pela degradação de desempenho observada em WAN. Esta assimetria é explicada pelo protocolo de ordem total utilizado pois observa-se que a degradação de desempenho é proporcional à distância, em termos de latência na rede, de cada um dos servidores àquele que é usado como coordenador que nestes resultados é o servidor 1.

Este facto mostra que o protocolo de comunicação escolhido tem um impacto directo no desempenho da aplicação. É pois interessante considerar a utilização de outro protocolo de ordenação total das mensagens no sentido de melhorar a simetria do sistema, por exemplo usando protocolos baseados na história causal [15].

6 Conclusões

Este artigo propõe uma avaliação da técnica de replicação de bases de dados conhecida como *Database State Machine*, combinando simulação da base de dados e da rede, com uma concretização do protocolo de comunicação em grupo e do procedimento de certificação. A utilização de carga gerada de acordo com a norma TPC-C e de um modelo detalhado da rede, aliados à utilização de um protótipo de componentes essenciais, permitem uma avaliação da DBSM com um detalhe sem precedentes.

Neste artigo em particular, é feita uma avaliação do impacto da latência resultante da utilização de uma WAN para interligar servidores replicados geograficamente distribuídos. Embora seja possível manter o débito em transacções por segundo de um servidor centralizado com capacidade idêntica, existe um impacto ao nível da latência de algumas das transacções devido à necessidade de re-execução de transacções abortadas durante a certificação. Como consequência, existe um consumo adicional de recursos dos servidores. Observa-se também que o protocolo de difusão atómica utilizado faz com que este custo não seja distribuído uniformemente por todos os servidores, tornando necessária a re-avaliação da adequação deste protocolo para esta aplicação.

A utilidade do modelo proposto não se esgota no entanto nestes resultados. Actualmente, procede-se à avaliação do impacto da limitação de largura de banda, tanto em absoluto como por existência de outro tráfego e de situações de congestão.

Referências

[1] R. Agrawal, M. Carey, and M. Livny. Models for Studying Concurrency Control Performance: Alternatives and Implications. In *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data*, 1985.

[2] R. Agrawal, M. Carey, and M. Livny. Concurrency Control Performance Modeling: Alternatives and Implications. *ACM Transactions on Database Systems*, 1987.

[3] G. Alvarez and F. Christian. Applying simulation to the design and performance evaluation of fault-tolerant systems. In *Proceedings of The 16th Symposium on Reliable Distributed Systems (SRDS '97)*. IEEE, 1997.

[4] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[5] K. Birman and R. van Renesse. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society Press, 1994.

[6] G. Chockler, I. Keidar, and R. Vitenberg. Group Communication Specifications: a Comprehensive Study. *ACM Computing Surveys*, 2001.

[7] D. Clark. RFC 813: Window and Acknowledgement Strategy in TCP. IETF Request for Comments, 1982.

[8] J. Cowie. *Scalable Simulation Framework API Reference Manual*, 1999.

[9] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards Realistic Million-Node Internet Simulation. In *Proc. of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applicatio ns (PDPTA'99)*, 1999.

[10] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The Dangers of Replication and a Solution. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM Press, 1996.

[11] K. Guo. *Scalable Message Stability Detection Protocols*. PhD thesis, cornell, 1998.

[12] IOzone Filesystem Benchmark. http://www.iozone.org.

[13] M. Kaashoek and A. Tanenbaum. Group Communication in the Amoeba Distributed Operating System.

[14] B. Kemme and G. Alonso. Don't Be Lazy, Be Consistent: Postgres-R, A New Way to Implement Database Replication. In *Proceedings of 26th International Conference on Very Large Data Bases*, 2000.

[15] L. Lamport. Time, Clocks and the Ordering of Events in Distributed Systems. *Communications of the ACM*, 21, 1978.

[16] F. Pedone. *The Database State Machine and Group Communication Issues*. PhD thesis, Département d'Informatique, École Polytechnique Fédérale de Lausanne, 1999.

[17] F. Pedone, R. Guerraoui, and A. Schiper. The Database State Machine Approach. *Distributed and Parallel Databases*, 2003.

[18] S. Pingali, D. Towsley, and J. Kurose. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. In *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 1994.

[19] PostgreSQL. http://www.postgresql.org.

[20] RCTS2. http://www.fccn.pt/rcts2.

[21] A. Schiper and A. Sandoz. Uniform Reliable Multicast in a Virtually Synchronous Environment. In *International Conference on Distributed Computing Systems (ICDCS)*, 1993.

[22] F. Schneider. Implementing Fault-Tolerant Services Using the State Machine Approach: a Tutorial. *ACM Computing Surveys (CSUR)*, 1990.

[23] A. Sousa, J. Pereira, F. Moura, and R. Oliveira. Optimistic total order in wide area networks. In *Proc. 21st IEEE Symposium on Reliable Distributed Systems*, pages 190–199. IEEE CS, 2002.

[24] A. Sousa, J. Pereira, L. Soares, A. Correia Jr., L. Rocha, R. Oliveira, and F. Moura. Evaluating the performance of the database state machine. Technical report, Universidade do Minho - Departamento de Informática, 2003.

[25] Transaction Processing Performance Council (TPC). TPC Benchmark^{TM} C standard specification revision 5.0, 2001.