SPECULA: Protocolo de Replicação Preditivo para Memória Transaccional por Software Distribuída*

João Fernandes, Nuno Carvalho, Paolo Romano, and Luís Rodrigues

{joao.fernandes, nonius, romanop}@gsd.inesc-id.pt, ler@ist.utl.pt
Instituto Superior Técnico, Universidade Técnica de Lisboa / INESC-ID

Resumo Este artigo descreve e avalia um protocolo de replicação preditivo para sistemas de memória transaccional em software distribuídos, baseado num esquema de certificação. O objectivo do algoritmo é mitigar os efeitos da latência na rede através da execução optimista de transacções. As transacções são executadas num único nó, de forma não coordenada, sendo utilizado o resultado da validação local como predição do resultado da validação final. O resultado das transacções (i.e., as modificações ao estado transaccional) é tornado visível de forma optimista e novas transacções são iniciadas com base no estado preditivo. Este processo repete-se de forma encadeada, podendo resultar numa sequência de transacções preditivas. Caso a predição se confirme, as transacções que dela dependiam são tornadas definitivas; caso contrário, a sequência preditiva é cancelada e o estado do sistema aquando da predição é reposto.

Abstract This paper describes and evaluates a speculative replication protocol for distributed software transactional memory systems, based on a certification scheme. This protocol tackles the effects of network latency, through the optimistic execution of transactions. Transactions are executed on a single node, in an uncoordinated fashion, and the result of the local validation is used as a prediction of the result of the final validation. The result of the speculatively executed transactions (i.e., the modifications to the transactional state) is optimistically made visible to their following transactions. This speculative process repeats itself, creating a chain of speculatively executed transactions. If the final validation of a speculatively executed transaction allows it to commit, its result is made definitive; otherwise, a cascading abort takes place and the system restarts its execution in the state that precedes the speculative process.

Keywords: Memória Transaccional por Software, Replicação, Predição, Execução Optimista

* Este trabalho foi parcialmente suportado pela FCT através do projecto "ARISTOS" (PTDC/EIA-EIA/102496/2008), pelo financiamento multianual do INESC-ID com fundos do programa PIDDAC e pela União Europeia, através do programa "Cloud-TM" (257784).

1 Introdução

A Memória Transaccional por Software (ff. STM, do Inglês *Software Transactional Memory*) consiste num paradigma de controlo de concorrência para acesso a estruturas de dados partilhadas por vários fios de execução, que recorre ao conceito de transacção desenvolvido para os Sistemas de Gestão de Bases de Dados (ff. SGBD). De acordo com este paradigma, o programador necessita apenas de delimitar as sequências de instruções que se devem executar de forma atómica, sem ser obrigado a recorrer a mecanismos de controlo de concorrência de baixo nível, como trincos ou semáforos.

Inicialmente, os sistemas de suporte à STM apenas consideravam arquitecturas multi-núcleo com memória partilhada. Mais recentemente, tem-se assistido a um investimento na procura de soluções capazes de suportar o paradigma da STM em sistemas distribuídos, como forma de simultaneamente aumentar a capacidade destes e os dotar de tolerância a faltas. A grande maioria destas soluções recorre a protocolos de difusão com ordem total, por forma a sequenciar de forma global as transacções que se executam no sistema. Um dos factores mais limitativos no desempenho das STMs distribuídas é a latência do protocolo de ordem total, que introduz atrasos significativos, quando comparados com os tempos relativamente curtos das transacções em memória. Este artigo apresenta uma técnica para mitigar este problema recorrendo à execução optimista de sequências de transacções com base em predição da ordem total.

O sistema proposto, designado por SPECULA, valida a transacção num único nó, de forma não coordenada, e utiliza o resultado desta validação como predição do resultado da validação final da transacção. Os resultados das transacções preditivas são tornados visíveis de forma optimista e novas transacções são iniciadas com base no estado preditivo. Este processo repete-se de forma encadeada, podendo resultar numa sequência de transacções preditivas. Caso a predição se confirme, as transacções que dela dependiam são tornadas definitivas; no caso contrário, a sequência preditiva é cancelada e é reposto o estado do sistema no momento da predição. Desta forma, é possível sobrepor o tempo necessário para a coordenação entre os nós do sistema distribuído com a computação (optimista) de transacções, com base na predição do resultado dessa coordenação. Este artigo descreve o sistema SPECULA, as técnicas utilizadas no seu desenvolvimento, e a avaliação do protótipo resultante.

O resto do artigo encontra-se organizado da seguinte forma. Na Secção 2 descreve-se o modelo do sistema, e na Secção 3 apresenta-se a nossa proposta de solução em maior pormenor. Na Secção 4 apresentam-se os resultados da análise experimental, e na Secção 5 regista-se um resumo dos desenvolvimentos na área, discutindo as vantagens e limitações de soluções previamente existentes. Por fim, na Secção 6 tecem-se alguns comentários sumários à solução apresentada e discutem-se direcções para trabalho futuro.

2 Modelo do Sistema

Consideramos o sistema como sendo composto por um conjunto de processos $\prod = \{p_1, ..., p_n\}$ que comunicam através de passagem de mensagens. Assumimos que a maioria dos processos não falha e que a restante minoria pode falhar por paragem. Assumimos ainda que o sistema é assíncrono, mas que existe um detector de falhas imperfeito com o poder necessário para permitir concretizar um Serviço de Comunicação em Grupo (ff. GCS, do Inglês *Group Comunication Service*)[1] oferecendo a semântica de *sincronia na vista*[2].

O serviço de GCS é caracterizado pelas seguintes garantias sobre as vistas entregues:

- **Integridade:** se um processo p entrega a vista v_i, então p pertence à vista v_i.
- **Sincronia na vista estrita:** as mensagens são entregues na mesma vista em que foram enviadas.
- **Vistas lineares:** as sequências de vistas entregues são totalmente ordenadas e para quaisquer duas vistas consecutivas v_i, v_{i+1} existe sempre um processo correcto em v_i que pertence a ambas as vistas.
- **Não trivialidade:** quando um processo falha ou é particionado da vista primária, este será eventualmente excluído da vista do componente primário.
- **Precisão:** um processo correcto p é eventualmente incluído na vista do componente primário.

O GCS disponibiliza um serviço de Difusão Atómica (ff. AB, do Inglês *Atomic Broadcast*)[3], o qual disponibiliza as seguintes duas primitivas de comunicação:

- **AB-envia(m):** usada para solicitar a difusão de uma mensagem m.
- **AB-entrega(m):** usada para entregar a mensagem m à aplicação.

O serviço AB oferece a garantia das seguintes propriedades:

- **Validade:** qualquer processo correcto p na vista v_i que difunde a mensagem m, entrega m.
- **Integridade:** qualquer mensagem m é entregue apenas uma vez por um processo p, e só se tiver sido difundida previamente.
- **Acordo uniforme:** se um processo p entrega a mensagem m na vista v_i, então qualquer processo correcto em v_i entrega m em v_i.
- **Ordem FIFO**[1]**:** se um processo p difunde a mensagem m antes da mensagem m', então qualquer processo correcto entrega m antes de m'.
- **Ordem total:** se dois processos p e q entregam as mensagens m e m', então fazem-no pela mesma ordem.

Relativamente às transacções, assumimos que estas são *dinâmicas* e *determinísticas*, i.e., o conjunto de leitura de uma transacção não é previamente conhecido, e a sua execução sobre um determinado estado transaccional produz sempre o mesmo resultado.

[1] do Inglês *First In, First Out*

3 SPECULA

O SPECULA pode ser classificado na categoria dos sistemas de STM distribuídos e replicados com base num esquema de certificação[4]. Neste tipo de sistemas, uma transacção executa-se localmente numa única réplica. Quando é requisitada a confirmação da transacção, é feita uma validação local. Caso a validação tenha sucesso, o conjunto de escrita da transacção (e opcionalmente o conjunto de leitura) são enviados usando o serviço de difusão atómica, que irá ordenar essa transacção em relação a outras transacções que se executem concorrentemente noutros nós do sistema. Finalmente, quando a transacção é entregue pelo serviço de AB, é realizada uma validação final da mesma. Note-se que entre a validação local e a validação final, outras transacções concorrentes podem ser confirmadas, o que pode levar à invalidação da transacção caso seja detectado algum conflito.

Diferentes variantes da aproximação acima descrita têm sido propostas. No algoritmo base, designado por *certificação sem votação*[5], o conjunto de escrita e leitura de cada transacção é disseminado através do serviço de AB, o que permite que todos os nós certifiquem a transacção de forma independente com resultados deterministas. Uma variante deste algoritmo codifica o conjunto de leitura num filtro de Bloom para reduzir o custo de comunicação [6]. Outra variante, designada por *certificação com votação*[5], envia apenas o conjunto de escrita, o que implica que apenas o nó onde a transacção se executou pode realizar a sua validação, comunicando posteriormente o resultado desta aos restantes nós. Finalmente, alguns algoritmos utilizam protocolos de difusão atómica com suporte a entregas optimistas, para iniciar a certificação o mais cedo possível, com base numa predição da ordem total de entrega das mensagens [7].

Em qualquer uma das soluções propostas previamente, o fio de execução permanece bloqueado até à confirmação (ou cancelamento) definitiva de uma transacção. Neste artigo propomos um sistema de suporte à execução de transacções em STMs que evita este bloqueio. Após a validação local de uma transacção, o seu resultado é tornado visível e o fio de execução desbloqueado, o que permite que a execução prossiga de forma preditiva, sob a hipótese optimista de que a validação final levará à confirmação definitiva da transacção. Este processamento preditivo é repetido se o fio de execução concluir uma nova transacção antes que a anterior seja confirmada (ou cancelada) de forma definitiva. Deste modo, é possível criar uma sequência de transacções preditivas. No caso em que a predição se confirma, os resultados são marcados como definitivos. Quando a predição se revela errada, é necessário cancelar todas as execuções preditivas e repor o estado antes da predição.

Para materializar as optimizações atrás descritas, é necessário realizar várias extensões ao ambiente de suporte à execução de STMs. Em primeiro lugar, é necessário que a lógica de certificação de transacções seja expandida para considerar transacções preditivas, isto é, transacções baseadas em resultados que foram tornados visíveis mas que ainda não estão confirmados de forma definitiva. Para além disso, é necessário alterar o ambiente de execução das transacções de forma a ser possível restaurar uma salvaguarda da aplicação num ponto an-

terior a uma predição que se revelou errada, mesmo após a execução de várias transacções executadas em modo preditivo. Finalmente, é preciso também bloquear um fio de execução antes deste externalizar um resultado baseado numa execução preditiva, até que esta seja definitivo.

Nas secções seguintes, são descritas as alterações realizadas ao ambiente de execução das transacções, as tecnologias utilizadas para realizar estas alterações, e o algoritmo de certificação com suporte para execuções preditivas.

3.1 Arquitectura de uma Réplica

A arquitectura do software instalado em cada réplica é ilustrada na Figura 1, na qual em negrito e itálico se destacam os componentes que são novos ou que foram por nós modificados. As principais diferenças em relação a outras arquitecturas para suporte a STM distribuída são as seguintes:

- Alterações à STM de base para suportar a noção de transacções preditivas. Tipicamente, um gestor de STM só considera os resultados gerados por transacções em curso e transacções confirmadas. No SPECULA é também necessário considerar os resultados gerados por transacções preditivas, os quais são visíveis para outras transacções preditivas, mas que podem vir a ser cancelados.
- É utilizada uma Máquina Virtual Java (ff. JVM, do Inglês *Java Virtual Machine*) com suporte a continuações[8], o que permite capturar o estado de um fio de execução no momento em que é feita uma predição, por forma a ser possível fazer a reposição desse estado caso a predição se venha a verificar errada.
- É concretizado um protocolo de gestão da replicação que tem em conta a existência de transacções preditivas, e é capaz de despoletar a confirmação ou cancelamento das mesmas em função dos resultados da validação final.

É possível recorrer a diferentes tecnologias para concretizar as funcionalidades acima descritas. No desenvolvimento do nosso protótipo optámos por usar os seguintes componentes: a JVM escolhida foi a OpenJDK[2], modificada para oferecer suporte a continuações [9]. A STM utilizada é a JVSTM[10]; esta STM, sendo multi-versão, é mais fácil de adaptar de forma a manter também versões preditivas. O GCS usado para suportar a coordenação entre réplicas é o Appia[11].

Para além disto, foi desenvolvido um *Class Loader* responsável por modificar o *bytecode* das classes da aplicação (utilizando a ferramenta ASM[12]) e carregá-las na JVM. A modificação do *bytecode* consiste em substituir os métodos com a assinatura `public void run()` em todas as classes não-abstractas que concretizam a interface `java.lang.Runnable`. O método injectado permite a outro componente, o Gestor de Predições (ff. GP), controlar o estado da aplicação (contador de programa e pilha de cada fio de execução, e amontoado), o que é representado na Figura 1 pelo conector que liga o GP à Aplicação.

[2] `http://openjdk.java.net`

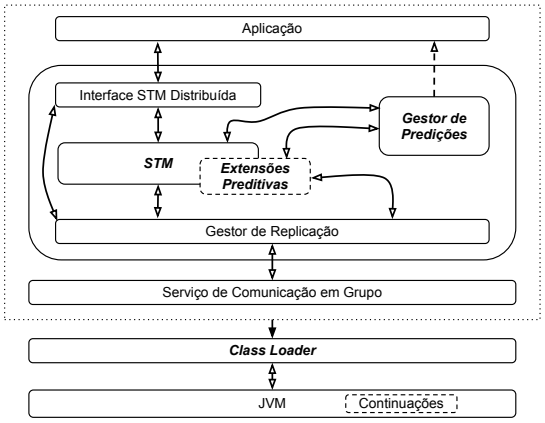


Figura 1: A arquitectura de uma réplica SPECULA

3.2 Ciclo de Vida de uma Transacção

Em qualquer momento, uma transacção t encontra-se, no nó em que é executada, num dos seguintes estados:

EM_EXECUÇÃO t encontra-se em execução, não tendo ainda sido requisitada a sua confirmação ou o seu cancelamento.

PENDENTE Foi requisitada a confirmação de t e o seu resultado foi aplicado de forma preditiva, tendo sido tornado visível para transacções futuras que se executarão em modo preditivo.

CONFIRMADA t foi confirmada de forma final, depois de ter passado com sucesso todas as validações necessárias.

CANCELADA t foi cancelada. O cancelamento pode dever-se i) à detecção de um conflito local, ii) à detecção de um conflito global ou, iii) ao pedido explícito de cancelamento por parte da aplicação.

3.3 Protocolo

O sistema SPECULA utiliza um esquema de replicação baseada em certificação com votação. Pretende-se desta forma reduzir o tamanho das mensagens trocadas na rede, uma vez que não é necessário difundir o conjunto de leitura. Note-se que a certificação com votação exige um passo adicional de comunicação, aumentando a latência do processo de coordenação. No entanto, esta latência é mascarada pelo processamento preditivo de transacções enquanto o processo de coordenação decorre. Sublinha-se que o processamento preditivo de transacções é realizado localmente pelo fio de execução que solicita a confirmação de uma transacção. Os resultados de transacções remotas continuam apenas a ser aplicados em cada nó quando essas transacções são confirmadas de forma definitiva.

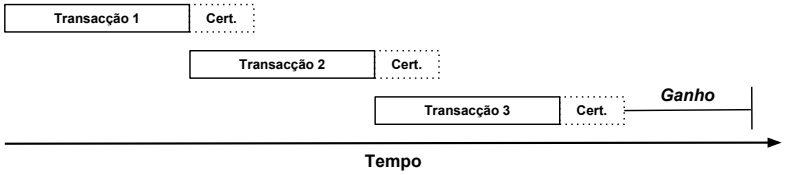


Figura 2: Execução de um fio no tempo, no sistema SPECULA

O protocolo aqui descrito oferece fortes garantias sobre a consistência dos dados replicados (*one-copy serializability*[13]).

A Figura 2 ilustra a execução de um fio no sistema SPECULA. A característica das transacções em memória que potencia o sucesso do SPECULA é o facto de estas serem tipicamente muito curtas [14], o que amplifica severamente o seu custo de replicação quando comparando com transacções em bases de dados.

Arranque de um Fio de Execução Quando é iniciado um fio de execução f, é executado o método `public void run()` que foi injectado na fase de carregamento da classe. Este método é responsável por criar e associar um contexto preditivo a f, através da chamada ao método `GP.novoContexto()`. O contexto fica acessível a f através do método `GP.actual()`.

Em qualquer momento, o contexto de um fio de execução é composto por uma fila com tamanho limite ajustado dinamicamente (com valores máximo e mínimo configuráveis), que contém todas as transacções que nele foram executadas e que ainda se encontram no estado *PENDENTE*, e por uma referência para a primeira transacção que foi cancelada (caso exista alguma) desde a última sincronização do fio. O adaptação dinâmica do tamanho da fila é baseada no pressuposto que, caso as predições se estejam a revelar correctas, se deve tentar aumentar o nível de optimismo, e vice-versa. Desta forma reduz-se ainda a hipótese de sobrecarga do GCS.

Execução de uma Transacção Quando uma transacção t é iniciada, começa no estado *EM_EXECUÇÃO*. t é imediatamente cancelada caso tente ler de estado preditivo criado por uma transacção no estado *CANCELADA*.

Pedido de Confirmação de uma Transacção Primeiro, qualquer transacção t é sujeita a uma validação local, que consiste em verificar se o seu conjunto de leitura reflecte o mais recente estado da aplicação (preditivo ou não). Caso tal não aconteça, t é imediatamente cancelada, transitando para o estado *CANCELADA*, e a aplicação é notificada. Caso a validação local não detecte qualquer conflito, é associada a t uma continuação que representa o ponto de execução em que se encontra o fio. t é então sujeita ao processo de validação final, sendo para este fim difundida uma mensagem através da primitiva de comunicação em grupo *AB-envia*, contendo o seu resultado (i.e., as suas alterações ao estado transaccional)

e identificador, o qual é garantidamente único. A confirmação preditiva de t é realizada imediatamente após a difusão da mensagem, momento a partir do qual o seu resultado se encontra visível para transacções futuras. Por fim, t transita para o estado *PENDENTE* e é associada ao contexto preditivo do fio onde foi executada, através do método `GP.adicionaTx(Transacção)`. O fio de execução é bloqueado caso se exceda o limite do tamanho da fila do contexto exceda o seu limite, por forma a controlar o processo especulativo, sendo desbloqueado assim que este volte abaixo do limite.

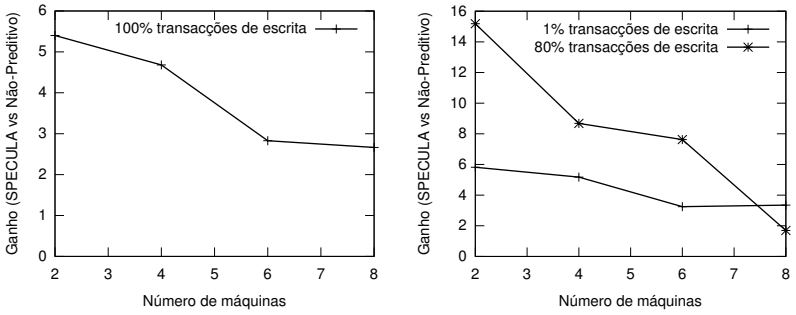
Validação Final de uma Transacção A validação final de uma transacção é feita segundo a sua ordem de entrega final. No sistema SPECULA, uma transacção t passa na validação final se o seu conjunto de leitura reflecte a mais recente versão do estado não-preditivo da aplicação, e se todas as transacções que foram previamente executadas no mesmo fio de execução se encontrarem no estado *CONFIRMADA*. Caso t cumpra estes dois requisitos, o seu resultado é dado como final, transitando para o estado *CONFIRMADA*. O GP é notificado da confirmação através do método `GP.txConfirmada(Transacção)`, e procede à remoção de t do contexto preditivo a que esta se encontrava associada. Caso a validação de t falhe, o GP é notificado através da invocação do método `GP.txCancelada(Transacção)`, e o estado da aplicação é revertido. Tanto o resultado de t, como o de qualquer outra transacção executada posteriormente no mesmo fio de execução, são imediatamente descartados. A execução do fio é retomada na continuação associada a t, a partir de onde é comunicado o cancelamento de t à aplicação.

Qualquer que seja o resultado da validação final de t, o GR é também notificado, e procede à difusão uma mensagem com o identificador e o destino da transacção (i.e., ser confirmada ou cancelada). Esta mensagem pode ser difundida sem garantias sobre a sua ordem de entrega.

Bloqueio de um Fio de Execução Quando uma transacção necessita de externalizar um resultado, tal como se descreve em mais pormenor na próxima secção, torna-se necessário parar a execução preditiva, o que se traduz em bloquear o fio de execução até que o resultado final da sequência de transacções preditivas seja conhecido. Para este efeito existe o método `GP.sincronizar()`, responsável por esperar que as transacções previamente executadas nesse fio de execução deixem o estado *PENDENTE*.

3.4 Lidar com Código Não-Transaccional

As aplicações são tipicamente compostas por partes de código transaccional e por partes de código não-transaccional. O SPECULA pretende que a execução de código não-transaccional possa ser sobreposta com a fase de certificação das transacções, o que implica que o código não-transaccional em questão seja reversível. No entanto, existem operações como as de entrada ou saída de dados que não são reversíveis.



(a) Contenção Nula (b) Contenção Elevada

Figura 3: Resultados Experimentais

Qualquer operação que potencie a passagem de informação para o exterior do sistema não pode ser executada sobre num estado preditivo, já que não é reversível. Estas operações podem ser detectadas através da análise do *bytecode* Java, devendo-se forçar o fio de execução a esperar pelo resultado definitivo das transacções preditivas antes de executar a operação que externaliza informação. Outra alternativa consistiria em modificar as classes na fase de carregamento da JVM, por forma a suportarem a reversão de estado. Este processo apresenta semelhanças com o da *transactificação* automática de código, um problema que não é trivial e que apresenta várias limitações [15].

Na actual concretização do SPECULA, a única sincronização automática é feita imediatamente antes do término de um fio de execução. O programador é por isso responsável por identificar as operações que interagem com o exterior e por forçar explicitamente a sincronização do fio de execução em questão antes da execução das mesmas, através da chamada do método `GP.sincronizar()`.

4 Avaliação

Nesta secção são apresentados os resultados experimentais obtidos através da execução de uma bancada de testes no sistema SPECULA. Executou-se a mesma bancada de testes num sistema replicado através de um esquema baseado em certificação com votação puro. A análise apresentada baseia-se na comparação dos resultados obtidos em ambos os sistemas. O ambiente de testes é composto por um aglomerado de 8 máquinas. Cada uma das máquina possui dois processadores Intel Xeon E5506, perfazendo um total de 8 núcleos físicos a 2,13GHz, e 8GB de RAM, sobre o qual se executa o sistema operativo GNU/Linux, versão 2.6.32 – 64 bits. As máquinas encontram-se interligadas através de um comutador Gigabit Ethernet. Todos os testes foram realizados num ambiente estável, sem qualquer tipo de falhas por parte dos nós envolvidos. Assumimos também que não ocorre qualquer perda de mensagens na rede.

A bancada de testes executada foi utilizada pela primeira vez em [16], sob o nome de *Bank Benchmark*. O trabalho computacional realizado consiste em simular o ambiente de um sistema bancário, efectuando-se a transferência de quantias entre contas, de forma atómica. É um exercício simples, mas que permite controlar de forma fina a taxa de conflitos, já que esta depende das contas envolvidas em cada uma das transacções.

Todos os testes foram executados num ambiente sem concorrência local, em que o limite máximo de transacções preditivas executadas em cada nó é 512 e o mínimo 1, sendo este aumentado de forma incremental a cada confirmação e reduzido para metade a cada cancelamento. Na Figura 3a não existe contenção, pois cada réplica escreve sobre um conjunto de contas bancárias virtuais distinto. Neste cenário, o ganho observado varia entre 5,4x e 2,6x, decrescendo à medida que o número de réplicas aumenta. Com o aumento do número de réplicas, aumenta a latência da rede e o débito total do sistema, o que leva a que o limite máximo definido de 512 transacções no estado *PENDENTE* seja atingido com maior frequência, o que limita o ganho alcançado pelo SPECULA.

Na Figura 3b são criadas $NMaquinas \times 2$ contas bancárias virtuais que são acedidas de forma aleatória por qualquer um dos processos, existindo por isso uma contenção elevada. Consideramos os cenários com 80% e 1% de transacções de escrita. Observa-se que no cenário com 1% de transacções de escrita, o ganho verificado é bastante similar ao observado na ausência de contenção. Tal não é surpreendente, pois quando o número de transacções de escrita é muito pequeno, o número de conflitos entre transacções também o é. No cenário com 80% de escritas observa-se que o ganho obtido é extremamente elevado, variando entre os 8x e 16x para 2 a 6 réplicas, o que ultrapassa os valores registados no cenário sem contenção. Verificou-se que no sistema SPECULA, o GCS tem tendência a entregar grandes sequências de transacções geradas pelo mesmo processo. Contrariamente, com o protocolo não-preditivo a entrega de transacções geradas em diferentes processos é bastante alternada. Como resultado, a taxa de cancelamento exibida pelo SPECULA acaba por ser inferior à do protocolo não-preditivo, o que conduz a ganhos elevados. Por outro lado, no cenário com 8 réplicas, observa-se um elevado decréscimo do ganho obtido (que ainda assim é de 1,6x). Neste caso, a carga adicional exercida sobre o GCS tende a degradar o desempenho do serviço de difusão atómica. Em particular, o GCS reage ao aumento da carga que lhe é imposta com uma diminuição na frequência com que entrega mensagens difundidas por outros nós que não o sequenciador. Este facto traduz-se numa redução significativa da capacidade de outros nós que não o sequenciador verem confirmadas as transacções que executaram, o que conduz a um elevado desnível no número de transacções confirmadas em cada nó e a uma redução do débito total do sistema.

5 Trabalho Relacionado

Existem vários protocolos de replicação para memória transaccional, tais como o BFC[6], o AGGRO[17] e o SCert[7]. O BFC (*Bloom Filter Certifica-*

tion) é um protocolo baseado em certificação cuja característica principal é a codificação do conjunto de leitura num filtro de Bloom, o que diminui o tamanho das mensagens à custa de um aumento residual e controlado de (falsos) conflitos. Isto permite reduzir o tempo necessário para confirmar cada transacção mas não evita que as transacções seguintes fiquem em espera até que o processo de confirmação esteja concluído.

O AGGRO (*AGGRessively Optimistic*) utiliza uma aproximação com base em replicação activa. A sua principal ideia é propagar de forma preditiva os resultados de transacções de acordo com a ordem de entrega optimista de mensagens por parte do serviço de difusão atómica. Avaliações experimentais mostram que, numa LAN, a ordem de entrega optimista das mensagens tipicamente coincide com a ordem total, propriedade que se designa por ordem total espontânea[18]. Quando a ordem de seriação optimista não é equivalente à ordem final, as transacções que leram de estado preditivo inconsistente são canceladas. Este processo permite sobrepor as fases de execução e certificação das transacções. A principal limitação do AGGRO prende-se com a necessidade de executar todas as transacções em todos os nós, o que limita a escalabilidade dos sistemas onde é aplicado.

O SCert (*Speculative Certification*) combina características do AGGRO com um esquema de replicação baseado em certificação com votação. A principal ideia do SCERT é propagar de forma preditiva o resultado de transacções que ainda não se encontram confirmadas, de acordo com a ordem de entrega optimista de mensagens por parte do serviço de difusão atómica. Esta propagação preditiva pretende reduzir o número de transacções que lêem valores desactualizados, o que se traduz numa redução da taxa de cancelamento. Embora este processo permita sobrepor as fases de execução e certificação, tal como no BFC, e ao contrário da nossa solução, o fio onde é executada uma transacção permanece bloqueado durante toda a fase de certificação desta.

6 Conclusões e Trabalho Futuro

Neste artigo introduzimos o SPECULA, um sistema que utiliza um esquema de replicação baseada em certificação com votação. Este sistema foi desenhado para optimizar o desempenho de sistemas de STM distribuídos, ao permitir o progresso do fio de execução com base no pressuposto de que a coordenação entre réplicas não irá alterar o resultado da validação local das transacções.

Como trabalho futuro, pretendemos i) sincronizar as tarefas automaticamente antes da execução de operações que são inerentemente não-transaccionais; ii) automatizar a reversão de alterações ao estado não-transaccional; iii) combinar o SPECULA com outros esquemas preditivos, nomeadamente ao nível da rede, e avaliar o desempenho da solução obtida.

Referências

1. Birman, K., Schiper, A., Stephenson, P., Birman, K., Schiper, A., Stephenson, P.: Lightweight causal and atomic group multicast. ACM Transactions on Computer

Systems **9** (1991) 272–314
2. Birman, K.P., Joseph, T.A.: Reliable communication in the presence of failures. ACM Transactions on Computer Systems **5** (1987) 47–76
3. Cachin, C., Guerraoui, R., Rodrigues, L.: Introduction to Reliable and Secure Distributed Programming (2. ed.). Springer (2011)
4. Pedone, F., Guerraoui, R., Schiper, A.: The database state machine approach. Distrib. Parallel Databases **14** (July 2003) 71–98
5. Wiesmann, M., Schiper, A.: Comparison of Database Replication Techniques Based on Total Order Broadcast. IEEE Transactions on Knowledge and Data Engineering **17** (2005) 551–566
6. Couceiro, M., Romano, P., Carvalho, N., Rodrigues, L.: D2STM: Dependable distributed software transactional memory. In: Proceedings of the 2009 15th IEEE Pacific Rim International Symposium on Dependable Computing. PRDC '09, Washington, DC, USA, IEEE Computer Society (2009) 307–313
7. Carvalho, N., Romano, P., Rodrigues, L.: SCert: Speculative certification in replicated software transactional memories. In: Proceedings of the 4th Annual International Systems and Storage Conference (SYSTOR 2011), Haifa, Israel (2011) (to appear)
8. Haynes, C.T., Friedman, D.P., Wand, M.: Continuations and coroutines. In: Proceedings of the 1984 ACM Symposium on LISP and functional programming. LFP '84, New York, NY, USA, ACM (1984) 293–298
9. `http://wikis.sun.com/display/mlvm/StackContinuations`
10. Cachopo, J., Rito-Silva, A.: Combining software transactional memory with a domain modeling language to simplify web application development. In: Proceedings of the 6th international conference on Web engineering. ICWE '06, New York, NY, USA, ACM (2006) 297–304
11. Miranda, H., Pinto, A., Rodrigues, L.: Appia, a flexible protocol kernel supporting multiple coordinated channels. In: Proceedings of the 21st International Conference on Distributed Computing Systems, Phoenix, Arizona, IEEE (2001) 707–710
12. Bruneton, E., Lenglet, R., Coupaye, T.: ASM: A code manipulation tool to implement adaptable systems. In: In Adaptable and extensible component systems. (2002)
13. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency Control and Recovery in Database Systems. Addison-Wesley (1987)
14. Palmieri, R., Quaglia, F., Romano, P., Carvalho, N.: Evaluating database-oriented replication schemes in Software Transactional Memory systems. In: Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on. (April 2010) 1–8
15. Anjo, I.F.S.D., Cachopo, J.: JaSPEx: Speculative Parallel Execution of Java Applications. In: 1º INFORUM, Faculdade de Ciências da Universidade de Lisboa (September 2009)
16. Herlihy, M., Luchangco, V., Moir, M.: A flexible framework for implementing software transactional memory. SIGPLAN Not. **41** (October 2006) 253–262
17. Palmieri, R., Quaglia, F., Romano, P.: AGGRO: Boosting STM Replication via Aggressively Optimistic Transaction Processing. Network Computing and Applications, IEEE International Symposium on **0** (2010) 20–27
18. Pedone, F., Schiper, A.: Optimistic atomic broadcast: a pragmatic viewpoint. Theoretical Computer Science **291** (2003) 79–101