Auto-Configuração de Máquina de Estados Tolerante a Falhas Bizantinas

Alírio Santos de Sá[1], **Allan Edgard Silva Freitas**[1,2], **Raimundo José de Araújo Macêdo**[1]

[1]Laboratório de Sistemas Distribuídos (LaSiD) / Departamento de Ciência da Computação
Universidade Federal da Bahia (UFBA) / Campus de Ondina - Salvador - BA - Brasil

[2]Instituto Federal da Bahia (IFBA) - Campus de Salvador - BA - Brasil

aliriosa@ufba.br, allan@ifba.edu.br, macedo@ufba.br

Resumo. *Protocolos tolerantes a falhas bizantinas (BFTs) têm sido aplicados em ambientes distribuídos hostis, sujeitos a falhas arbitrárias por causas naturais ou maliciosas (e.g. intrusões). O PBFT foi o primeiro BFT a obter êxito na prática, combinando diversos mecanismos tais como batching de requisições e rejuvenescimento de réplicas, para obter um desempenho aceitável sob condições as definidas durante projeto. Um novo desafio surge quando os mecanismos destes BFTs precisam ser configurados em ambientes dinâmicos, cuja carga das aplicações, a QoS dos canais de comunicação ou a topologia da rede podem mudar dinamicamente. Este artigo propõe uma abordagem para o ajuste automático de parâmetros do mechanismo de batching, baseada em um laço de controle realimentado, que permite aos BFTs manter seu bom desempenho mesmo em ambientes dinâmicos.*

1. Introdução

Replicação de máquina de estados[Schneider 1990, Lamport 1978] tem sido amplamente utilizada para aumentar a confiabiliade de servidores em sistemas distribuídos. O protocolo de gestão das replicas é implementado de modo que seja possível tolerar a falha de até f destes servidores, elevando o nível de serviço obtido. Em ambientes benignos e com hardware e software com especificação correta (ou com alta probabilidade disto), em geral falhas ocorrem por *crash* (parada silenciosa) do servidor devido a causas naturais (e.g. falha elétrica). Contudo, em ambientes de computação distribuída abertos (como a Internet), os nós servidores podem estar imersos em um ambiente hostil, em que os componentes do sistema podem falhar por causas naturais ou maliciosas (e.g. ataques de intrusão), ou mesmo podem existir falhas latentes do ambiente computacional, as quais induzem falhas por comportamento arbitrário – no qual, além da possibilidade de *crash*, os resultados podem ser omitidos ou mesmo alterados. Falhas arbitrárias são muito mais difíceis de serem tratadas que falhas mais benignas, como crash ou omissões.

O trabalho de Castro e Liskov [Castro and Liskov 1998] sobre um protocolo de replicação tolerante a falhas bizantinas, dito PBFT (*Practical Byzantine Fault Tolerance*), foi o primeiro a tornar viável a utilização de tais protocolos em ambientes reais devido à técnicas de otimização utilizadas, tais como rejuvenescimento de réplicas, uso de *batching* para otimizar o processamento de conjunto de requisições enviado em uma dada janela de tempo, mudanças de visões periódicas para rotacionar o papel do coordenador, uso de *checkpoint* periódico para coleta de lixo etc. Na abordagem proposta por Castro

e Liskov, o ajuste desses parâmetros operacionais é definido antes da execução do protocolo (i.e. *offline*). Embora tal ajuste possa produzir o efeito desejado nos ambientes computacionais em que se pode modelar *a priori* as variações das cargas dos servidores e atrasos nas redes, tal configuração *offline* pode não ser satisfatória quando são considerados ambientes distribuídos dinâmicos – nos quais a carga de trabalho dos servidores, a qualidade de serviço dos canais de comunicação ou a topologia da rede podem mudar em tempo de execução.

O presente artigo propõe uma solução inédita para a adaptação dos parâmetros associados ao mecanismo de *batching* (i.e. timeout e tamanho de *batch*), cujos ajustes são fundamentais para o bom desempenho do serviço replicado, em termos do tempo de resposta e da taxa de atendimentos de requisições. Para validação da eficácia dessa nova abordagem, propomos uma versão adaptativa do PBFT (dita aPBFT), a qual foi inteiramente implementada e comparada em vários cenários com a proposta original do PBFT. Os resultados obtidos por simulação demonstraram a maior eficiência de nossa abordagem, dada a sua habilidade de se auto-ajustar a diferentes condições de execução.

Até onde sabemos, a proposta apresentada neste artigo é a primeira solução para replicação bizantina capaz de ajustar os parâmetros operacionais de forma dinâmica e representa uma contribuição na direção de protocolos de replicação bizantina auto-gerenciáveis.

O resto deste artigo se organiza da seguinte forma: a Seção 2 descreve o funcionamento do protocolo original e trabalhos relacionados, a Seção 3 apresenta a nossa proposta de auto-configuração; finalmente, a Seção 4 tece as considerações finais.

2. PBFT e Trabalhos Relacionados

De modo a maximizar o desempenho, o PBFT utiliza chaves públicas para a troca periódica de chaves simétricas de sessão, sendo estas últimas utilizadas na comunicação entre as réplicas. Devido ao custo da autenticação, o primário pode agrupar um conjunto de requisições recebidas em uma única mensagem com um lote de requisições (com um único número seqüencial) a serem processadas (*batching*). Nesse mecanismo de *batching*, cada requisição recebida é armazenada em *buffer* e seu respectivo resumo (*digest*) é gerado. Quando existem requisições suficientes para preencher um lote, uma mensagem de PRE-PREPARE é enviada contendo, entre outras coisas, o conjunto de *digests* calculados para cada requisição do lote – o que permite reduzir o tamanho da mensagem. Para prevenir que o primário fique bloqueado esperando indefinidamente que o lote de requisições seja preenchido, quando a primeira requisição do lote é recebida, um temporizador é iniciado, e caso o lote não seja preenchido dentro do *timeout* de *batching*, então a mensagem de PRE-PREPARE é enviada com as requisições que foram loteadas até então.

Outros protocolos foram publicados com o objetivo de otimizar aspectos do PBFT. O Zyzzyva [Kotla et al. 2007] modifica o sub-protocolo de acordo por meio da execução especulativa. Recentemente, o PBFT foi especializado em [Wester et al. 2009] para permitir execução especulativa para operações que alteram o estado das réplicas. Entretanto, diferente do Zyzzyva, esta versão do PBFT permite que o cliente execute de forma especulativa, mas não externaliza o estado da máquina replicada antes que o acordo seja executado. Em [Guerraoui et al. 2010] é apresentada um protocolo genérico de máquina de estados replicada que permite compor diferentes estratégias de atuação para o sub-

protocolo de acordo.

3. aPBFT

A estratégia de ajuste dinâmico de parâmetros, introduzida neste artigo, é concebida abstraindo os detalhes internos de implementação e considerando um modelo abstrato no qual os diferentes mecanismos, sub-protocolos e procedimentos do protocolo de replicação bizantina são vistos como caixas pretas organizadas de forma a compor um *pipeline* abstrato. Os estágios desse *pipeline* abstrato são selecionados considerando as etapas de interesse da adaptação de parâmetros. Diferentes *pipelines* abstratos podem ser compostos para um mesmo protocolo de replicação, dado os diferentes fluxos de execução e seus respectivos mecanismos, sub-protocolos e procedimentos associados.

A partir do modelo abstrato introduzido neste artigo, baseada no *pipeline* abstrato, propõe-se o aPBFT (Adaptive PBFT), uma extensão adaptativa do PBFT, a qual baseia a sua adaptação no ajuste dinâmico do tamanho da janela de *batch* (BS, *Batch Size*) e do *timeout* de *batch* (BT, *Batch Timeout*) durante um fluxo normal de execução de uma requisição. A escolha dos parâmetros de adaptação se deve ao fato do mecanismo de *batching* representar um ponto importante para conciliar a carga das aplicações com os custos computacionais associados ao fluxo de execução do protocolo de replicação durante o atendimento das requisições.

Para realizar a adaptação, o aPBFT realiza estimativas a respeito da carga imposta pelos clientes e sobre o desempenho do protocolo básico (PBFT). Para isto, a estratégia proposta implementa um laço de controle em que usa informações obtidas a partir dos eventos associados ao envio, recebimento e processamento de mensagens do PBFT e determina os novos parâmetros do *batching*, ver Figura 1.

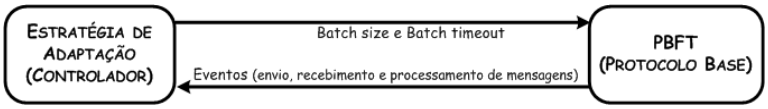

Figura 1. Laço de controle implementado pelo aPBFT.

A estimativa do timeout de *batching* se baseia no intervalo médio de tempo para ordenação e execução das requisições (MTE, *Mean Time To Ordering and Execution*).

A estratégia de adaptação considera que o *timeout* de *batching* deve ser longo o suficiente para permitir o acúmulo de requisições para a composição do lote, mas, por outro lado, o mesmo deve ser curto o suficiente para permitir que pelo menos um lote de requisições esteja no estágio de *ordering* – de modo a manter tal estágio sempre ocupado, na medida em que existem requisições a serem ordenadas e processadas. Entretanto, se o intervalo médio entre chegadas das requisições dos clientes (dito, MTA – *Mean Time Between Arrivals*) é menor que o MTE, então o *timeout* de *batching* pode ser zero, uma vez que a carga imposta pelos clientes é muito baixa e os estágios de *ordering* e *processing* do *pipeline* estão operando em um ritmo menor que suas capacidades de execução.

A idéia central para o ajuste de BS é a mesma usada na adaptação do *timeout* de *batching*, isto é: BS deve ser definido de tal forma que permita ao menos um lote de requisições esteja nos estágios *ordering* ou *processing*. Assim, se os estágios *ordering* e *processing* levam em média MTE unidades de tempo para realizar em conjunto suas

atividades, então o lote deve conter o número de requisições que foram possíveis de ser acumuladas neste intervalo de tempo.

O aPBFT consiste em três atividades básicas: (i) sensoriamento dos eventos do PBFT; (ii) ajuste do *timeout* de *batching*; e (iii) ajuste do tamanho do *batch*.

O aPBFT foi inteiramente implementado e seu desempenho avaliação através de simulação usando o Simulator HDDSS[Freitas and Macêdo 2009]. Devido a restrições de espaço omitimos neste texto os detalhes matemáticos e computacionais de nossa solução, assim como os dados de desempenho - embora possamos afirmar que a análise de desempenho realizada de fato demonstrou a eficácia de nossa proposta em adaptar dinamicamente os parâmetros do *batching*, otimizando o desempenho quando possível.

4. Conclusões

Este artigo propôs uma abordagem de auto-configuração (aPBFT) para o ajuste da janela de *batching* a partir do protocolo PBFT clássico. Uma vez que a essência do protocolo PBFT é mantida, a sua corretude permanece, enquanto focamos em incrementar seu desempenho. Tal abordagem pode ser aplicada para demais parâmetros do PBFT, e para os demais protocolos de tolerância a falhas bizantinas, baseados no mesmo conjunto de sub-protocolos, mesmo que adotem, por exemplo, uma estratégia distinta para o sub-protocolo de acordo.

Referências

Castro, M. and Liskov, B. (1998). Practical Byzantine fault tolerance. *Operating Systems Review*, 33:173–186.

Freitas, A. E. S. and Macêdo, R. J. A. (2009). A simulation tool for dynamic and hybrid distributed systems (in portuguese: *Um Ambiente para Testes e Simulações de Protocolos Confiáveis em Sistemas Distribuídos Híbridos e Dinâmicos*). In *Proceedings of the XXVII Brazilian Symposium on Computer Networks and Distributed Systems (SBRC' 2009)*, pages 826–839.

Guerraoui, R., Knežević, N., Quéma, V., and Vukolić, M. (2010). The next 700 BFT protocols. In *Proceedings of the 5th European conference on Computer systems*, pages 363–376. ACM.

Kotla, R., Alvisi, L., Dahlin, M., Clement, A., and Wong, E. (2007). Zyzzyva: speculative byzantine fault tolerance. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 45–58. ACM.

Lamport, L. (1978). Time, Clocks and The Ordering of Events in a Distributed System. In *Communications Of The ACM*, pages 558–565.

Schneider, F. B. (1990). Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Computing Surveys*, 22(4):299–319.

Wester, B., Cowling, J., Nightingale, E. B., Chen, P. M., Flinn, J., and Liskov, B. (2009). Tolerating latency in replicated state machines through client speculation. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'2009, pages 245–260, Berkeley, CA, USA. USENIX Association.