Explorando Redução de Log para Recuperação Rápida

Luiz Gustavo C. Xavier[1], Cristina Meinhardt[1], Odorico M. Mendizabal[1]

[1]Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

xavier.luizgustavo@gmail.com,

{odorico.mendizabal, cristina.meinhardt}@ufsc.br

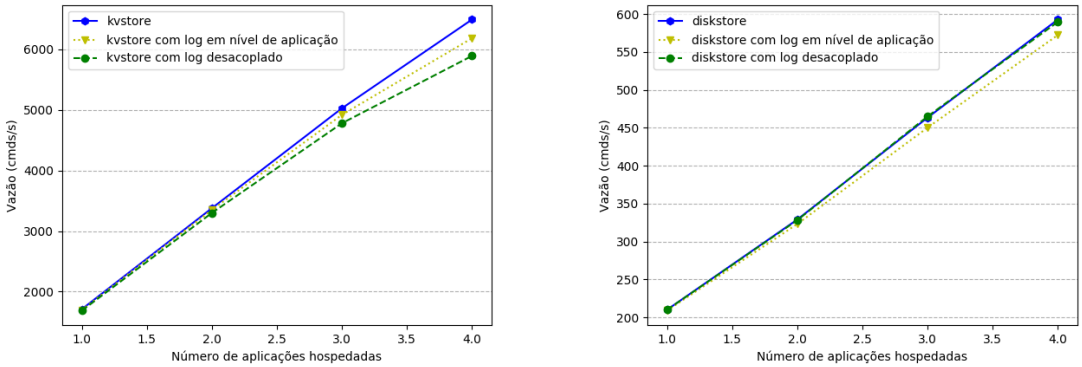
Resumo. *Protocolos de recuperação eficientes são importantes para aumentar a disponibilidade de sistemas replicados tolerantes a falhas. Neste trabalho é apresentado o projeto de um protocolo de recuperação que explora a eliminação consciente de comandos contidos em arquivos de log, utilizando-se de processos logger independentes e fracamente acoplados à aplicação.*

1. Introdução

Sistemas fortemente consistentes e tolerantes a falhas são usualmente implementados através da técnica de Replicação Máquina de Estados (SMR) [Lamport 1978]. Em SMR, réplicas do sistema se comportam como máquinas de estado, partindo de um mesmo estado inicial e evoluindo através das mesmas transições pelo processamento de comandos determinísticos encaminhados por clientes. Para manter níveis de disponibilidade e robustez da aplicação, o sistema deve ser capaz de resgatar réplicas defeituosas a seu estado prévio de computação, anterior à incidência de falhas. Protocolos de recuperação de estado desempenham este papel, realizando o processamento de *logs* de comandos executados ou utilizando-se de estados duráveis capturados durante a execução normal da aplicação. Alguns trabalhos apresentam alternativas procurando minimizar a sobrecarga na execução normal da aplicação da aplicação [Bessani et al. 2013], ou permitindo que uma réplica em recuperação possa atender parcialmente novas requisições [Mendizabal et al. 2017]. Este trabalho explora a característica de que, durante a recuperação, o tempo de processamento de comandos é diretamente proporcional ao número de comandos contidos no *log*. Sendo assim, uma possível abordagem para minimizar o tempo de recuperação é a eliminação de comandos desnecessários, de modo que o processamento do arquivo reduzido propicie à replica a obtenção do mesmo estado consciente que obteria com o processamento do *log* completo.

Para que não seja comprometida a correção do modelo de replicação adotado, deve-se assegurar que o processamento do *log* compactado alcance o mesmo estado consistente obtido com a execução do arquivo completo. Para tal, esta eliminação de operações deve ser implementada utilizando-se de mecanismos capazes de detectar possíveis dependências ou efeitos de sobrescrita entre comandos. Um exemplo são comandos de escrita subsequentes efetuados sobre uma mesma variável, que podem ser eliminados da recuperação se, e somente se, os valores atribuídos em estados intermediários não sejam necessários para computação de outros comandos.

O mapeamento de dependências entre comandos deverá ser realizado com um auxílio de um grafo acíclico dirigido, o que deve representar um alto custo computacional possivelmente comprometendo o desempenho da réplica durante o processamento de



(a) Aplicação *kvstore* (b) Aplicação *diskstore*

Figura 1. Análise de escalabilidade do serviço de *log* desacoplado.

comandos, ou mesmo estendendo a janela de vulnerabilidade do sistema ao incrementar o tempo de recuperação. Nesse sentido, há a hipótese de que tal processamento possa ser minimizado no caso de ser realizado por um processo de *log* independente e fracamente acoplado à aplicação.

Um processo *logger* desacoplado age como uma réplica adicional no sistema replicado, compartilhando do mesmo protocolo de acordo dos demais participantes. Como encarregam-se apenas da tarefa de registro de comandos para recuperação, estes processos representam um menor custo computacional, uma vez que não processam comandos e não encaminham respostas a clientes da aplicação. Além do mais, é possível que tais participantes exerçam papéis passivos no protocolo de acordo, de modo que não participem do quórum de consenso e somente recebam comandos já decididos por outras réplicas. A Figura 1 exibe a vazão observada por diferentes aplicações simultâneas, com e sem o uso do serviço de *log* desacoplado, quando submetidas a cargas de trabalho com uso intenso de CPU (Fig. 1a) e com uso intenso de escritas em armazenamento secundário (Fig. 1b).

2. Próximos Passos

Observa-se que o processo de truncamento evidencie uma pequena sobrecarga na execução normal da aplicação, as custas de possibilitar uma mais rápida recuperação de estado. Por isso, como trabalho futuro, é importante analisar a execução e análise comparativa deste processo de compressão em diferentes níveis de periodicidade: *Imediata*, realizada logo após o registro de um novo comando; *Intervalar*, respeitando uma periodicidade similar a um procedimento de *checkpoint*; e *Adiantada*, realizada somente com o início do processo de recuperação.

Referências

Bessani, A., Santos, M., Felix, J., Neves, N., and Correia, M. (2013). On the efficiency of durable state machine replication. In *USENIX ATC 2013*.

Lamport, L. (1978). Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565.

Mendizabal, O. M., Dotti, F. L., and Pedone, F. (2017). High performance recovery for parallel state machine replication. In *IEEE ICDCS 2017*.