Uma Arquitetura Distribuída e Hierárquica para Validação Diferida de Transações em Sistemas de Armazenamento Chave-Valor*

João Amaro, Luís Rodrigues, Manuel Bravo, Miguel Matos, Paolo Romano

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

Resumo Este artigo descreve um conjunto de extensões para sistemas de armazenamento chave-valor que permitem suportar transações sem comprometer a capacidade de escala do sistema. Estas extensões combinam estratégias de confirmação hierárquica com técnicas de agregação de mensagens em lotes para aumentar o débito do sistema sem comprometer a latência da maioria das transações.

1 Introdução

Os sistemas de armazenamento chave-valor são hoje um componente central nos sistemas distribuídos de grande escala uma vez que a principal vantagem destes sistemas é a sua capacidade de escalar e de suportar elevados débitos no acesso ao armazenamento, mesmo quando a quantidade de dados e o número de clientes é muito elevado. Por exemplo, sistemas de armazenamento como o Cassandra [7] armazenam terabytes de dados e suportam centenas de milhares de acessos por segundo. Infelizmente, os modelos de coerência fraca e a falta de mecanismos de controlo de concorrência tornam o desenvolvimento de algumas aplicações bastante difícil e sujeito a erros [2]. Desta forma, tem-se verificado um interesse crescente em desenvolver mecanismos que permitam reforçar as garantias oferecidas pelos sistemas chave-valor sem comprometer de forma excessiva o desempenho dos mesmos.

Neste artigo estamos interessados em desenvolver mecanismos que permitam suportar a execução de transações em sistemas de armazenamento chave-valor. Em particular, concentramo-nos em estudar mecanismos que permitam aumentar a capacidade de escala do processo de validação das transações. Para este efeito, combinamos estratégias de confirmação hierárquica com técnicas de agregação de mensagens em lotes para aumentar o débito do sistema sem comprometer a latência da maioria das transações. Em particular, os nossos mecanismos combinam ideais da arquitetura de validação hierárquica proposta em [5] com o sistema de ordenação diferida proposto em [6]. A arquitetura proposta permite que transações que acedam a dados de apenas um fragmento possam ser validadas de forma concorrente e com elevado desempenho, e que transações

* Este trabalho foi parcialmente suportado pela FCT através do projeto com referência UID/ CEC/ 50021/ 2013.

que acedem a múltiplos fragmentos não tenham um impacto substancial na validação das primeiras. Resultados experimentais mostram que esta arquitetura apresenta um bom compromisso entre a melhoria de desempenho de transações com elevada localidade e o impacto no desempenho de transações com menos localidade.

2 Trabalho Relacionado

Uma transação é uma sequência de operações de leitura e escrita cuja execução respeita um conjunto de propriedades, conhecidas pela designação de propriedades ACID (atomicidade, coerência, isolamento e durabilidade). Neste caso, consideramos que o sistema pretende oferecer *serializabilidade* [9], ou seja, que uma execução concorrente de várias transações produz resultados equivalentes a uma execução em série dessas transações.

Uma das técnicas para executar o controlo de concorrência consistem em usar trincos no acesso aos dados e um protocolo de confirmação em duas fases para confirmar a transação. Infelizmente, o facto das transações poderem aceder aos dados por diferentes ordens pode gerar situações de interbloqueio, que são de difícil deteção e resolução num sistema distribuído. A alternativa mais frequente, consiste em executar as transações de forma otimista e depois executar um processo de validação no momento em que esta tenta confirmar. Para assegurar a correção do resultado da validação, todas as transações devem ser validadas respeitando uma ordem total. Os principais algoritmos de ordenação total usados em sistemas transacionais são os sequenciadores fixos, sequenciadores distribuídos com coordenação *em-linha*, e sequenciadores distribuídos com estabilização diferida. Exemplos de sistemas que recorrem a sequenciadores fixos são o vCorfu [11] e o Megastore [3]. Um exemplo relevante de um sistema que usa sequenciadores distribuídos com coordenação *em-linha* é o Spanner [4]. Exemplos de sistemas que usam sequenciadores distribuídos com estabilização diferida são o Chariots [8], o Calvin [10] e o FLACS[5], que recorrem a uma hierarquia de validadores. Este último sistema é particularmente interessante pois explora a localidade das transações para executar a sua validação o mais cedo possível: uma transação pode ser validada mal seja ordenada em relação a transações que acedam aos mesmos dados.

3 Validação Hierárquica por Lotes

Neste trabalho exploramos a possibilidade de adaptar o sistema de confirmação hierárquica FLACS proposto em [5] para concretizar suporte para transações em sistemas de armazenamento chave valor com requisitos de suportar um elevado débito. O sistema FLACS foi originalmente proposto para sistema geo-replicados e não foi concretizado de forma a otimizar o débito do sistema. Uma das técnicas mais eficazes para aumentar o débito do sistema consiste em fazer o processamento e a troca de informação por lotes, o que permite combinar múltiplas mensagens lógicas num único pacote de rede, o que permite

reduzir o impacto dos custos fixos associados à troca de mensagens (tais como a transmissão e processamento de cabeçalhos, gestão de interrupções, trocas de contexto, etc). Alguns sistemas de armazenamento relevantes que usam estes princípios são o Chariots [8] e o Eunomia [6].

A ideia chave da nossa arquitetura consiste portanto em aplicar o algoritmo do FLACS, num único centro de dados, recorrendo a técnicas de agregação de mensagens para aumentar o débito do sistema. Assumimos um sistema de armazenamento chave-valor em que os dados estão fragmentados por vários *nós de armazenamento*. Este sistema é complementado por um conjunto de *nós de validação*. Os nós de validação estão organizados numa hierarquia em árvore, em que existe um único nó raiz com dois ou mais filhos. Existe um nó folha co-localizado com cada nó de armazenamento. As transações que são validadas e confirmadas são serializáveis. Para aumentar o débito do sistema, a nossa arquitetura introduz as seguintes modificações ao FLACS: mantemos múltiplas versões por objeto para evitar os conflitos em transações de leitura, usamos controlo concorrência baseado em estampilhas temporais evitando o uso de trincos, e fazemos uma gestão de metadados que permite que os nós validadores possam certificar as transações sem terem de aceder aos dados.

De modo a podermos avaliar a arquitetura proposta de forma realista decidimos concretizá-la sob um sistema de armazenamento chave-valor usado na industria, o Riak KV [1]. Desenvolvemos um protótipo em que o Riak KV foi aumentado com uma hierarquia de validadores que concretiza os mecanismos acima referidos.

4 Avaliação Preliminar

Realizamos uma avaliação preliminar da nossa arquitetura, de forma a comparar o débito suportado em relação ao débito de um sistema baseado na utilização de um sequenciador. Na Figura 1 está representado um gráfico que mostra a variação do débito para quatro cargas de trabalho diferentes. As cargas de trabalho distinguem-se pelo grau de localidade de acesso das transações, que se traduz na percentagem de transações que pode ser validade nas folhas e na percentagem de transações que necessita de ser validada na raiz. Como se pode observar, mesmo com 50% das transações a necessitarem de ser validadas pelo nó raiz, a nossa arquitectura ainda consegue suportar um débito que é praticamente o dobro do que é fornecido pela variante baseada num único sequenciador.

5 Conclusões e Trabalho Futuro

Neste artigo propomos uma nova arquitetura para validação e confirmação de transações de forma distribuída. Resultados experimentais obtidos mostram que a arquitetura proposta apresenta ganhos de débito face às soluções existentes baseadas em sequenciadores. Como trabalho futuro planeamos comparar o desempenho da arquitetura proposta com o desempenho de outros e protocolos de validação e confirmação de transações e planeamos realizar experiências com

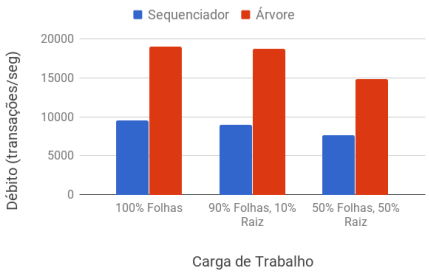

Figura 1: Débito obtido para a nossa concretização de um sequenciador e da arquitetura proposta com diferentes cargas de trabalho.

árvores de gestores de transações maiores de modo determinar a sua capacidade de escala.

Referências

1. `http://basho.com/products/riak-kv/`
2. Bailis, P., Fekete, A., Franklin, M.J., Ghodsi, A., Hellerstein, J.M., Stoica, I.: Feral concurrency control: An empirical investigation of modern application integrity. SIGMOD (2015)
3. Baker, J., Bond, C., Corbett, J.C., Furman, J., Khorlin, A., Larson, J., Leon, J.M., Li, Y., Lloyd, A., Yushprakh, V.: Megastore: Providing scalable, highly available storage for interactive services. In: CIDR. vol. 11, pp. 223–234 (2011)
4. Corbett, J.C.e.a.: Spanner: Google's Globally Distributed Database. ACM Trans. Comput. Syst. **31**(3), 1–22 (2013)
5. Grov, J., Ölveczky, P.: Scalable and fully consistent transactions in the cloud through hierarchical validation. In: Hameurlain, A., Rahayu, W., Taniar, D. (eds.) Data Management in Cloud, Grid and P2P Systems. pp. 26–38. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
6. Gunawardhana, C., Bravo, M., Rodrigues, L.: Unobtrusive deferred update stabilization for efficient geo-replication. arXiv preprint arXiv:1702.01786 (2017)
7. Lakshman, A., Malik, P.: Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review **44**(2), 35–40 (2010)
8. Nawab, F., Arora, V., Agrawal, D., El Abbadi, A.: Chariots: A scalable shared log for data management in multi-datacenter cloud environments. In: EDBT. pp. 13–24 (2015)
9. Papadimitriou, C.H.: The serializability of concurrent database updates. J. ACM **26**(4), 631–653 (Oct 1979)
10. Thomson, A., Diamond, T., Weng, S.C., Ren, K., Shao, P., Abadi, D.J.: Calvin: Fast distributed transactions for partitioned database systems. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. SIGMOD '12, Scottsdale (AZ), USA (2012)
11. Wei, M., Tai, A., Rossbach, C.J., Abraham, I., Munshed, M., Dhawan, M., Stabile, J., Wieder, U., Fritchie, S., Swanson, S., et al.: vcorfu: A cloud-scale object store on a shared log. In: NSDI. pp. 35–49 (2017)