Estimativa da Divergência de Réplicas em Sistemas Geo-replicados

Andy Gonçalves, Valter Balegas, Sérgio Duarte, Rodrigo Rodrigues, and Nuno Preguiça *

CITI — Departamento de Informática,
Universidade Nova de Lisboa, Portugal

Resumo Muitos serviços na Internet são suportados através de infraestruturas de *cloud computing*, em que os dados são replicados em máquinas de vários centros de dados distribuídos geograficamente. Para tal, é frequente o recurso a mecanismos de replicação otimista, já que esta permite melhorar a latência das operações, mas cria o problema de poder haver réplicas divergentes do mesmo objeto.
Uma via para controlar a divergência das réplicas passa pelo uso de métricas que medem o quão desatualizados estão os dados de uma réplica. Neste artigo, propõe-se a utilização de uma abordagem probabilística ao cálculo dessas métricas de divergência. A nossa solução consiste em estimar no cliente a probabilidade que a execução de uma operação tem de violar as restrições da aplicação e, com base nisso, permitir às aplicações decidir se devem executar a operação local ou remotamente. Esta aproximação consegue reduzir significativamente a comunicação necessária entre réplicas, quando comparada com a utilização de soluções deterministas.

Keywords: Cloud Computing; Geo-Replicação; Divergência limitada; Replicação otimista; Operação desconectada; Métricas

1 Introdução

As infraestruturas de *cloud* são atualmente utilizadas para disponibilizar todo o tipo de serviços e aplicações, desde redes sociais e jogos, até ao comércio online ou ao fornecimento de conteúdos multimédia. Estas infraestruturas têm benefícios face às tradicionais arquiteturas cliente/servidor, porque facilitam a escalabilidade dos serviços decorrente da possibilidade de expansão da infraestrutura consoante a necessidade. As infraestruturas de *cloud* tendem ainda a recorrer à geo-replicação de dados para melhorar a disponibilidade e latência dos serviços, colocando réplicas dos dados próximo dos clientes que os acedem.

O teorema CAP mostra que é impossível em simultâneo obter elevada disponibilidade e consistência dos dados e ainda tolerar o particionamento na rede [4]. Para lidar com esta impossibilidade, os sistemas desenvolvidos em ambientes de

* Este projeto é parcialmente financiado por PEst-OE/EEI/UI0527/2011 e PTDC/EIA-EIA/108963/2008

cloud optam frequentemente por relaxar a consistência dos dados, permitindo fornecer elevada disponibilidade apesar de eventuais partições na rede [3,6,2]. Esta aproximação pode ainda minimizar a latência por permitir o acesso à réplica mais próxima do cliente.

Quando se usa esta aproximação, as aplicações podem aceder a versões desatualizadas dos dados, o que pode levar à violação das restrições de integridade das aplicações. Um exemplo disto é uma loja *online*, onde o stock de produtos tem de se manter positivo. Com acessos a uma versão desatualizada do stock, um cliente pode fazer uma operação que torna negativo o valor real do stock, apesar de o valor a que o cliente tem acesso se manter positivo.

Para lidar com este problema, permitindo a execução local das operações nestas situações, foi proposta a utilização de soluções de particionamento das permissões de execução (*escrow*)[8] e controlo da divergência dos dados [11]. Estas soluções não são apropriadas para ambientes de *cloud* com um número elevado de clientes. No primeiro caso, haveria dificuldade em atribuir a todos os clientes as permissões necessárias antecipadamente, o que dificulta a sua utilização. No segundo caso, para limitar a divergência de forma determinista seria necessário efetuar comunicações entre os clientes e servidores sempre que um limite definido em qualquer cliente pudesse ser violado, o que inviabiliza a sua utilização.

Neste trabalho propõe-se a utilização de técnicas probabilísticas para estimar a divergência das réplicas, em particular, das réplicas dos clientes. Para tal, desenvolveram-se algoritmos que estimam o valor da divergência das réplicas em termos do seu valor numérico e do número de operações executadas remotamente. A informação é coligida nos centros de dados e propagada para os clientes quando estes obtêm uma cópia dos dados, permitindo aos clientes estimar a divergência entre a sua cópia local e a cópia do servidor sem necessidade de comunicações adicionais com o servidor.

Com base nesta informação, os clientes podem estimar, com um certo grau de certeza, se uma operação por eles efetuada garante uma restrição. Isto permite que o cliente adapte o nível de coordenação com o centro de dados consoante o pretendido pelo cliente. Os resultados preliminares obtidos mostram que o nosso sistema é capaz de manter restrições de integridade e adaptar o nível de coordenação oportunamente.

Este artigo está organizado da seguinte forma: a secção 2 descreve trabalho relacionado em termos de mecanismos de controlo de divergência; a secção 3 apresenta a abordagem probabilística ao controlo de divergência usada no nosso trabalho; a secção 4 descreve a arquitetura do sistema e a secção 5 resume as conclusões tiradas com este trabalho.

2 Trabalho Relacionado

Várias soluções foram propostas para medir e controlar a divergência de dados replicados. Alguns sistemas [11,9] definem métricas para determinar a divergência de uma réplica face a um estado abstrato, a que chamamos *cópia única*, com

todas as atualizações de todas as réplicas aplicadas por uma ordem de serialização. Estes sistemas forçam a sincronização para que a divergência de uma réplica face à cópia única não exceda os limites que a aplicação pode tolerar.

O TACT [11] define diferentes métricas de divergência (valor númerico, diferença na ordenação das operações e frescura temporal) e propõe algoritmos para manter a divergência das réplicas limitada. Estes algoritmos exigem o contacto entre as réplicas sempre que uma operação pode levar à violação da divergência permitida. Em particular, para a métrica numérica, a solução proposta exige que, ao executar uma operação, uma réplica contacte com todas as réplicas cujos limites possam ser violados, o que pode levar a comunicações permanentes. No Mobihoc [9], uma das réplicas atua como servidor central, recebendo todas as atualizações e disseminando-as para as réplicas conforme os limites definidos por elas. Adicionalmente, o servidor central minimiza a comunicação com as réplicas com base na informação sobre o nível de interesse das réplicas nos diferentes objetos.

Mais recentemente, foram utilizadas técnicas probabilísticas para determinar limites de frescura para leituras feitas em sistemas de quorums parciais, minimizando a comunicação [1]. O trabalho apresentado neste artigo explora igualmente uma aproximação probabilística para estimar a divergência dos dados.

3 Sistema para estimativa da divergência

Nesta secção descreve-se a solução desenvolvida para estimar a divergência dos dados em ambientes de computação *cloud*.

3.1 Modelo do Sistema

Neste trabalho consideramos um ambiente de *cloud* com dois níveis de replicação. No primeiro nível, existe um pequeno conjunto de centros de dados nos quais executam servidores, sendo que a base de dados é replicada totalmente em cada centro de dados. Num segundo nível, os clientes podem fazer *cache* dos dados presentes nos servidores. Os clientes replicam apenas pequenos subconjuntos dos dados de forma a permitir a execução local das operações e a sua submissão assíncrona para os servidores.

Assume-se que os objetos armazenados no sistema são CRDTs [10], garantindo a convergência final das réplicas independentemente da ordem pela qual as atualizações são aplicadas às réplicas.

A aproximação desenvolvida permite estimar a divergência das réplicas dos clientes e estimar a probabilidade de uma operação levar à violação das restrições da aplicação. A ideia base da nossa abordagem consiste em utilizar estatísticas obtidas a partir das atualizações vistas anteriormente, em conjunto com métodos probabilísticos para prever a evolução do estado do objeto ao longo do tempo. Estes mecanismos são usados para estimar a divergência de uma réplica em relação à cópia única, com um grau de certeza que pode ser controlado pela aplicação, com implicações diretas sobre o grau de comunicação entre a réplica do cliente e a réplica do centro de dados.

3.2 As Métricas

As métricas utilizadas na nossa abordagem baseiam-se nas mesmas métricas propostas noutros sistemas [11,9], com as necessárias adaptações ao modelo de consistência usado:

- **valor**: indica o quão diferente está o valor numérico do objeto em relação à cópia única;
- **operações**: indica quantas atualizações feitas por outras réplicas estão por aplicar na réplica local;

A figura 1 ilustra o significado de cada métrica. X é um objeto numérico que se encontra replicado. A atualização mais recente, feita pela réplica B no instante 15, decrementa X em 3 unidades. Essa operação não foi propagada ainda para a réplica A.A métrica de valor indica portanto uma divergência de três unidades, e a métrica das operações indica que há uma operação que ainda não foi propagada para a réplica A.

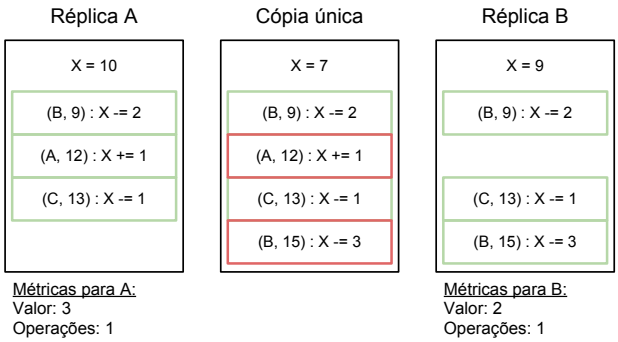

Figura 1. Divergência de réplicas, em relação à cópia única

3.3 Estimar o Valor das Métricas

Nesta secção descreve-se como se usam estatísticas sobre atualizações já vistas, para estimar o valor das duas métricas. A solução usada baseia-se na utilização duma estimativa simples que pressupõe que, quando existe um elevado número de clientes, a agregação das operações de todos leva a um ritmo de atualizações regular.

Assim, para estimar o quão divergente está o objeto replicado face ao estado da cópia única, necessita-se de uma estatística: o ritmo de crescimento do objeto, λ_{obj}, calculado a partir das atualizações observadas. De seguida mostramos como calcular o ritmo para ambas as métricas.

Métrica de Valor Para a métrica de **valor** o ritmo representa a variação do valor do objeto por unidade de tempo. Assim:

$$\lambda_{objVal} = \frac{\Delta_{Val}}{\Delta_{Time}} = \frac{V_f - V_i}{T_f - Ti} \tag{1}$$

onde T_i e T_f são as estampilhas temporais que demarcam o início e fim do período de tempo Δ_{Time}, e V_x é o valor numérico do objeto em T_x.

Métrica de Operações Para a métrica de **operações** procura-se um ritmo que represente o número de atualizações por unidade de tempo, ou seja:

$$\lambda_{objOrd} = \frac{n}{\Delta_{Time}} \tag{2}$$

onde Δ_{Time} é o período de tempo que passou para o qual se quer saber o ritmo, e n o número de atualizações que aconteceram nesse período de tempo.

O Ritmo na Estimativa A diferença entre o estado de uma réplica e o estado de cópia única é a divergência. O nosso objetivo é produzir uma estimativa da divergência, $\eth$, que se aproxime desse valor.

Como λ_{obj} é um crescimento em função do tempo, estimar o estado de cópia única consiste em calcular a divergência que occoreu na réplica para o intervalo de tempo que decorreu desde a última sincronização com a réplica do centro de dados[1]. Assim, sendo Δ_{Time} o tempo passado desde o último estado visto, $\eth = \lambda_{obj} \times \Delta_{Time}$. Portanto o estado atual é estimado adicionando ao último estado visto, a divergência estimada, $\eth$.

O Grau de Certeza Estimada a divergência, há que calcular o grau de certeza da estimativa, que é probabilidade de o estado ter variado em $\eth$. Para isso recorre-se à distribuição de Poisson [5]. Sendo o número de eventos que ocorrem num espaço de tempo representado por uma variável aleatória X, que segue uma distribuição de Poisson com uma média de λ eventos num dado período de tempo, a probabilidade de acontecerem k eventos no mesmo período é dada pela expressão:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k \in \mathbb{N}_+,\ \lambda > 0 \tag{3}$$

Como λ_{obj} é um ritmo por unidade de tempo, para se obter um grau que tem em conta o tempo, Δ_{Time}, passado desde o último estado visto, o parâmetro λ equivale a $\lambda = \lambda_{obj} \times \Delta_{Time} = \eth$. Sendo o parâmetro k a divergência estimada, $k = \eth$, pelo que para estas métricas, $\lambda = k = \eth$.

A distribuição de Poisson, originalmente, só pode ser usada com $\lambda > 0$. Assim, para ritmos negativos, é utilizado o módulo da divergência, porque a incerteza de

[1] O centro de dados pode não ter visto todas as operações feitas até um determinado momento, mas a estimativa é feita com base nas operações que este dispõe.

um valor de divergência negativa é a mesma do simétrico desse valor, portanto passamos a ter os parâmetros $\lambda = k = |\eth|$ na expressão (3).

3.4 Limitar a Divergência

Tal como em outros sistemas, para limitar a divergência nas réplicas locais, são definidos limites para as métricas em cada réplica. Como a estimativa tem um grau de incerteza associado, para limitar a divergência tem de ser definido também o grau de certeza com que se quer que $\eth$ esteja abaixo do limite, lim. Quanto maior este grau de certeza, menor é a probabilidade de o limite ter sido transposto, mas mais frequentemente ocorrerá coordenação com o centro de dados para limites baixos. Por oposição, um limite mais alto relaxa a necessidade de um grau de certeza maior.

O grau de certeza, ζ, de que $\eth$ está abaixo do limite lim será a soma dos graus de certeza de todos os valores entre 0 (o mínimo de divergência permitido, ou seja, nenhuma), e lim:

$$\zeta = P(X \leq k) = \sum_{k=0}^{lim} P(X = k), \quad k \in \mathbb{N}_+ \tag{4}$$

onde P pode ser calculado com a expressão (3).

3.5 Restrições de Integridade

Para garantir que uma restrição é mantida, é necessário definir o mínimo de divergência necessária, $\eth_{res}$, que é necessário verificar-se para que a restrição seja violada. Posto isto, garantir uma restrição de integridade com grau de certeza ζ, torna-se um problema de verificar se o limite de divergência $\eth_{res}$ é mantido, como visto na secção anterior.

Suponha-se que no exemplo da loja *online* há a restrição que o *stock* nunca é negativo, e quer ser verificado com um grau de certeza alto, $\zeta = 0.95$. Um cliente viu há 3 segundos que o *stock* era de 10, com um ritmo de crescimento de -1 por segundo (portanto $\lambda = k = |\eth| = |-1 \times 3| = 3$). Neste caso, $\eth_{res} = 10$, já que para a restrição ser violada, o *stock* teria de variar em 10, para ser 0. Assim, o grau de certeza, ζ_{res} que a divergência foi de 10 ou menos, é obtido usando a expressão (4) com $lim = 10$ e $\lambda = k = 3$, cujo resultado é $\zeta_{res} = 0.9997$. Uma vez que $\zeta_{res} \geq \zeta$, a restrição é garantida com o grau de certeza desejado.

4 Arquitetura

O sistema proposto é fornecido sob a forma de um *middleware* para permitir a sua fácil integração noutros sistemas já existentes. O sistema é composto por dois componentes, um gerador, nos centros de dados, e um estimador, nos clientes. Estes componentes situam-se entre as camadas da aplicação e do armazenamento.

No centro de dados, o componente gerador é responsável por guardar informação das atualizações recebidas, para gerar estatísticas sobre os objetos.Estas estatísticas são usadas para gerar o ritmo de crescimento.

No lado do cliente, o componente estimador é responsável por guardar informação sobre os limites das métricas, as restrições que se querem preservar, e o último estado dos objetos lidos no centro de dados , assim como o ritmo de divergência calculado pelo componente gerador. Quando o cliente invoca uma leitura sobre um objeto, esta é pré-processada pelo estimador para detetar se o objeto está desatualizado, segundo o critério de divergência em vigor para esse objeto.

A atualização dos ritmos é feita periodicamente entre o gerador e o estimador, podendo o gerador comunicar oportunamente quando ocorre uma variação súbita do ritmo de divergência.

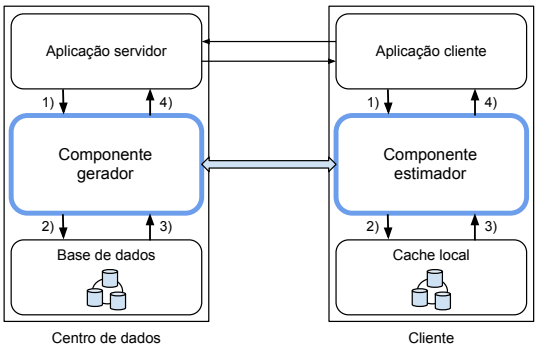

Figura 2. Arquitetura do sistema

A figura 2 mostra esta arquitetura, e o fluxo de execução de uma operação. A aplicação faz um pedido ao componente, que implementa a interface da base de dados subjacente (1). O componente comunica com a base de dados (2) e obtém a resposta (3), e depois de executar as suas funções de geração/estimativa/comunicação, responde à aplicação (4).

5 Conclusão

Este artigo apresenta uma solução que permite estimar a evolução dos dados e a divergência das réplicas em sistemas de *cloud*, usando duas métricas de divergência: o valor numérico dos dados e o número de operações não observadas. Com base nesta solução, permite-se às aplicações estimar o valor dos dados num dado momento e limitar a divergência das réplicas locais. O número de mensagens para calcular as estimativas é relativamente pequeno, quando comparado com outras soluções existentes na literatura, necessitando apenas de coordenar o

cliente e o centro de dados para trocarem meta-dados que indicam a estimativa de evolução do estado do objeto replicado.

Para além de calcular a divergência, a abordagem proposta permite preservar restrições aplicacionais simples (inequações do tipo $x \geq k$). Deste modo, a aplicação pode gerir o risco de executar uma operação localmente ou contactar o servidor para confirmar o resultado, reduzindo a latência no primeiro caso.

No futuro, pretende-se estudar de forma mais completa o impacto da solução na diminuição das comunicações num sistema de *cloud* assim como integrar na solução aproximações mais sofisticadas de previsão [7].

Referências

1. Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. Probabilistically bounded staleness for practical partial quorums. *Proc. VLDB Endow.*, 5(8):776–787, April 2012.
2. Valter Balegas and Nuno Preguiça. Swiftcloud: replicação sem coordenação. *INForum*, 2012.
3. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, October 2007.
4. Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, June 2002.
5. Frank A Haight. *Handbook of the Poisson distribution*. Wiley New York, 1967.
6. Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. Don't settle for eventual: scalable causal consistency for wide-area storage with cops. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 401–416, New York, NY, USA, 2011. ACM.
7. Spyros Makridakis, Steven C Wheelwright, and Rob J Hyndman. *Forecasting methods and applications*. John Wiley & Sons, 2008.
8. Nuno Preguiça, J. Legatheaux Martins, Miguel Cunha, and Henrique Domingos. Reservations for conflict avoidance in a mobile database system. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, MobiSys '03, pages 43–56, New York, NY, USA, 2003. ACM.
9. Nuno Santos, Luís Veiga, and Paulo Ferreira. Vector-field consistency for ad-hoc gaming. In *Proceedings of the 8th ACM/IFIP/USENIX international conference on Middleware*, MIDDLEWARE2007, pages 80–100, Berlin, Heidelberg, 2007. Springer-Verlag.
10. Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In *Proceedings of the 13th international conference on Stabilization, safety, and security of distributed systems*, SSS'11, pages 386–400, Berlin, Heidelberg, 2011. Springer-Verlag.
11. Haifeng Yu and Amin Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Trans. Comput. Syst.*, 20(3):239–282, August 2002.