Proposta de um serviço de replicação desacoplado da aplicação

Rafaela Tillmann[1], **Odorico M. Mendizabal**[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brazil

rafaela.tillmann@grad.ufsc.br, odorico.mendizabal@ufsc.br

Resumo. *Na sociedade contemporânea a indisponibilidade de um serviço pode acarretar em vários prejuízos. Para assegurar a disponibilidade das aplicações, é possível replicá-las em múltiplos servidores dado dois enfoques diferentes, replicação passiva e Replicação de Máquina de Estados. Entretanto, devido aos desafios associados à implementação dessas duas estratégias, este trabalho visa propor o desenvolvimento de um serviço de replicação desacoplado.*

1. Introdução

Informações são o cerne da sociedade, conexão digital é a representação humana da contemporaneidade. Alguns segundos de inatividade em sistemas de armazenamento ou processamento de dados pode representar uma perda significativa para as organizações, como foi como para a Amazon em 2021, quando a indisponibilidade de apenas 59 minutos acarretou em uma perda de 34 milhões de dólares em vendas [Independent 2023]. Portanto, para assegurar a disponibilidade dos serviços, é importante implementar alguma estratégia de replicação dos serviços, como a replicação passiva (*primary-backup*) ou a Replicação de Máquina de Estados (SMR, do inglês *State Machine Replication*).

A replicação passiva mantém apenas um servidor para lidar com as requisições dos clientes e, após o processamento da requisição, este servidor (primário) replica o estado atualizado para os demais servidores (*backups*). Esse enfoque, apesar de assegurar a confiabilidade, pode elevar a latência das requisições e ocasionar indisponibilidade momentânea do serviço no caso de substituição do primário.

Na Replicação de Máquina de Estados, é previsto que todas as réplicas iniciem no mesmo estado, executem as mesmas requisições de modo determinístico e em uma mesma ordem, avançando por estados idênticos. Diferentemente da replicação passiva, todas as réplicas irão executar as requisições dos clientes e, antes de efetuar a operação, há necessidade de estabelecer um acordo sobre a ordenação das requisições entre as réplicas. Esse acordo é assegurado por protocolos de consenso ou protocolos de difusão atômica, os quais são estruturados em nível de aplicação, fato que eleva complexidade de implementação, visto que o projetista deve possuir conhecimento dessas técnicas.

Este trabalho, portanto, propõe o desenvolvimento de um serviço de replicação desacoplado das aplicações, que assegure a ordenação das requisições e atue como intermediário entre o cliente e as réplicas. Espera-se alcançar alta modularidade, escalabilidade e otimizações no desenvolvimento, visto que a lógica da aplicação será desassociada das réplicas.

2. Replicação

Estratégias de replicação são comumente utilizadas para prover tolerância a falhas, garantindo a execução do serviço mesmo na ocorrência de falhas em um número limitado de

réplicas. A seguir são ilustradas duas técnicas amplamente utilizadas.

Replicação passiva (ou primary-backup): é uma estratégia que foca em um único servidor responsável pelo processamento das requisições e por envio das respostas para os clientes. Após o servidor processar a solicitação do cliente, ele é responsável por transmitir o estado final para as demais réplicas, que irão atuar como *backups* caso o servidor principal seja acometido com alguma falha e necessite ser substituído [Budhiraja et al. 1993]. Essa estratégia, apesar de assegurar replicação do sistema, eleva a latência das requisições, visto que após o processamento da requisição o servidor primário deve replicar o estado para um quórum de servidores *backups* antes de efetivamente responder ao cliente. Apesar da possibilidade de responder o cliente logo após o processamento da requisição, essa ação não assegura consistência forte, já que há possibilidade do servidor ser substituído por uma réplica em um estado anterior, no caso de falhas.

Replicação de Máquina de Estados (ou State Machine Replication): é uma estratégia que prevê que todas as réplicas irão executar as requisições de clientes. Schneider aponta que se forem executadas operações determinísticas, ordenadamente, em diferentes réplicas, elas irão evoluir para estados idênticos. Portanto, o enfoque dessa estratégia é iniciar todas as réplicas em um mesmo estado e assegurar que todas as requisições serão executadas, ordenadamente, por todas as réplicas [Schneider 1990].

Para prover essa ordenação, são comumente aplicados protocolos de consenso, como o Paxos e o Raft, porém, há necessidade de implementação desses protocolos em nível de aplicação, que eleva os custos de escalabilidade da aplicação e impossibilita aplicar a solução de replicação para outros serviços, além da própria complexidade associada à implementação dos protocolos.

3. Arquitetura

O projeto visa a implementação de um serviço de replicação totalmente desacoplado, que atue como mediador entre as réplicas e os clientes, assegurando a ordenação das solicitações dos clientes. Nessa perspectiva, a arquitetura proposta visa o desenvolvimento dos aspectos listados.

3.1. Desacoplamento

Estratégias de desacoplamento de funcionalidades de serviços tolerantes a falhas não são inéditas. Como, o *DistributedLog* que é um serviço de *logs* replicados desenvolvido para auxiliar na replicação do Twitter Manhattan. Nele, o *DistributedLog* é responsável por assegurar a ordenação das requisições que serão posteriormente lidas pelas réplicas. Entretanto, ainda há necessidade de uma réplica atuar como líder, intervindo nas requisições dos clientes e na gerência das demais réplicas [Guo et al. 2017]. Xavier *et al.* e Scharf *et al.* também propõem o desenvolvimento de serviços de *logs* desacoplados, porém, as réplicas são responsáveis por estabelecer o consenso e apenas após essa ação que a sequência de *logs* é registrada [Xavier et al. 2020, Scharf et al. 2023].

Esses serviços ilustram a possibilidade de desacoplar algumas funcionalidades das réplicas, entretanto, ainda há a necessidade da atuação das réplicas no fluxo da replicação. Nessa perspectiva, o projeto prevê o desacoplamento integral, possibilitando que as réplicas atuem apenas no processamento das operações solicitadas pelos clientes, além de proporcionar para os projetistas otimizações na implementação da replicação.

3.2. Generalização

Devido à complexidade envolvida na implementação de replicação, há o desenvolvimento de inúmeras aplicações com garantia de replicação que têm possibilidade de serem incorporadas a outros serviços, como o Amazon Aurora [Verbitski et al. 2017], um SGBD com replicação automática compatível com MySQL. Entretanto, como o Amazon Aurora é construído apoiado em um *fork* do código do MySQL, ele é complexo quanto à manutenção, além de possuir uma forte dependência do MySQL.

Para contornar esse vínculo direto com o MySQL, Coelho *et al.* propõe o Loom, que usa o servidor original para facilitar implementação da replicação, além de proporcionar generalização quanto ao SGBD aplicado. Ele cria uma interface entre o banco de dados e a aplicação, possibilitando que toda estratégia de replicação seja desassociada do servidor [Coelho et al. 2023]. Entretanto, apesar de possibilitar generalização quanto ao contexto dos SGBDs, o Loom, tal como Amazon Aurora e outras soluções, está focado em um único contexto e não proporciona um serviço totalmente genérico para replicação. Nesta perspectiva, a arquitetura proposta será implementada com objetivo de possibilitar que qualquer aplicação possa empregar o serviço ao estruturar uma replicação.

3.3. Modularidade

A arquitetura prevê que o consenso seja um módulo acoplável, no qual é possível optar por um serviço de *logs* replicados ou por uma implementação com protocolos de consenso para assegurar a ordenação das requisições. Este módulo visa, além da ordenação, reter as requisições por um tempo configurável até o envio para as réplicas. Além da modularidade prevista na implementação do consenso, a própria proposta é solução modular para a implementação da replicação nas aplicações.

3.4. Escalabilidade

Como o serviço propõe o desacoplamento de toda estratégia de replicação das réplicas, é possível escalar aplicação com facilidade, visto que para adição de servidores não é necessário que a aplicação seja configurada para adentrar no ecossistema. Além disso, pretende-se investigar a possibilidade de aplicar estratégias de *sharding* e particionamento, de modo que requisições de clientes possam ser destinadas para conjuntos de réplicas independentes. O particionamento, entretanto, pode restringir o grau de transparência, pois requer conhecimento específico sobre a semântica do serviço a ser replicado.

4. Implementação

O serviço de replicação está em estágio de revisão e discussões da literatura, entretanto, há intenção do desenvolvimento de um protótipo de um mediador responsável por interceptar as requisições de um cliente e servidor genéricos e que envie estas para uma instância do Apache BookKeeper, serviço que provê *logs* distribuídos com replicação e foi, inicialmente, eleito para o desenvolvido do protótipo.

Após esse desenvolvimento, pretende-se aplicar ao protótipo outras estratégias para assegurar a ordenação (como uma implementação do Paxos), com objetivo de comparar o desempenho entre elas e certificar a modularidade do serviço. Por fim, o serviço proposto irá prosseguir para a etapa de avaliação, para guiar as modificações necessárias até o desenvolvimento do projeto definitivo.

5. Conclusão

Devido à constante evolução e transmissão contínua de informações na sociedade contemporânea, a garantia da disponibilidade de serviços é crucial. Logo, a replicação em múltiplos servidores é uma estratégia para mitigar os prejuízos decorrentes da indisponibilidade, proporcionando confiabilidade e redundância. Entretanto, há desafios associados à replicação passiva e à Replicação de Máquina de Estados. Portanto, a proposta foca no desenvolvimento de um serviço de replicação desacoplado que certifique a ordenação das requisições, atue na interceptação das requisições e como gerenciador das respostas para os clientes. Com isso, o projeto visa superar algumas limitações de outras soluções ao proporcionar desacoplamento, generalização, flexibilidade modular e escalabilidade.

Referências

Budhiraja, N., Marzullo, K., Schneider, F. B., and Toueg, S. (1993). The primary-backup approach. In *Distributed Systems (2nd Ed.)*. ACM Press/Addison-Wesley Publishing Co.

Coelho, F., Alonso, A., Ferreira, L., Pereira, J., and Oliveira, R. (2023). Loom: A closed-box disaggregated database system. In *Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing*, pages 30–39.

Guo, S., Dhamankar, R., and Stewart, L. (2017). Distributedlog: A high performance replicated log service. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1183–1194.

Independent, T. (2023). Amazon down: Internet outage hits sales. https://www.independent.co.uk/news/business/amazon-down-internet-outage-sales-b1861737.html.

Scharf, J. a. L., Xavier, L. G. C., and Mendizabal, O. M. (2023). Joining parallel and partitioned state machine replication models for enhanced shared logging performance. In *Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing*, page 90–99.

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

Verbitski, A., Gupta, A., Saha, D., Brahmadesam, M., Gupta, K., Mittal, R., Krishnamurthy, S., Maurice, S., Kharatishvili, T., and Bao, X. (2017). Amazon aurora: Design considerations for high throughput cloud-native relational databases. In *SIGMOD 2017*.

Xavier, L. G., Dotti, F., Meinhardt, C., and Mendizabal, O. (2020). Scalable and decoupled logging for state machine replication. In *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 267–280.