Cache Aplicacional Consistente e Eficiente ⋆

Francisco Mendes, Nuno Preguiça e João Leitão

NOVA LINCS & DI, FCT, Universidade NOVA de Lisboa

Palavras-chave: Cache aplicacional · Consistência · Concorrência.

Introdução. O *caching* aplicacional [4] é essencial para o bom funcionamento das aplicações e serviços Web modernos que dependem de bases de dados. Esta camada de cache, que armazena dados e resultados de interrogações complexas à base de dados, pode melhorar de forma significativa a performance ao produzir respostas mais rápidas e ao reduzir a carga na base de dados.

Um dos principais desafios é garantir que os dados na cache permanecem consistentes com os dados na base de dados, quando são emitidos pedidos concorrentes à aplicação. A integração na aplicação de mecanismos para garantir essa consistência é complexa, sendo muitas vezes ignorada pelas aplicações ou levando a descartar dados da cache desnecessariamente.

Neste artigo propomos o ClearCache, um sistema que integra uma cache em aplicações que utilizem MongoDB [1] como base de dados. O sistema pretende resolver os problemas previamente mencionados ao efetuar, de forma transparente, uma gestão correta, consistente e eficiente dos dados armazenados na cache.

ClearCache. O ClearCache é uma camada de *middleware* que faz a gestão automática e consistente dos dados acedidos pela aplicação, armazenando-os numa cache aplicacional para melhor desempenho. O ClearCache interceta os pedidos endereçados ao MongoDB e determina se podem aceder apenas à cache ou também à base de dados. O sistema é implementado como uma biblioteca cliente, mais especificamente, como um *wrapper* da interface do cliente do MongoDB (usando a API do mesmo) e por um conjunto de funções definidas no servidor aplicacional Redis.

A escolha da base de dados MongoDB foi motivada pelo seu desempenho, flexibilidade dos dados e escalabilidade das bases de dados NoSQL. A inexistência de outros sistemas semelhantes ao ClearCache que trabalhem com bases de dados NoSQL [6, 5] foi também uma forte motivação. O sistema de cache Redis [2] foi o eleito para integrar com o ClearCache, principalmente porque oferece diversos tipos de dados, e outras funcionalidades, como scripts em Lua que usufruímos para implementar parte da lógica do ClearCache.

Para resolver o problema da consistência, o ClearCache implementa algoritmos nas operações de leitura e escrita para manipular os dados da cache.

A primeira técnica consiste em armazenar em cada documento um objeto do tipo Timestamp, num campo `_ts`. Este campo `_ts` representa o instante em que

⋆ Este trabalho é apoiado pelo NOVA LINCS (UIDB/04516/2020) com o apoio financeiro da FCT.IP

o documento foi criado ou atualizado pela última vez, e é criado pelo servidor para cada operação. Por serem únicos, é possível compará-los e determinar qual a última versão que deve ser armazenada em cache, mesmo na presença de pedidos concorrentes onde pode haver atrasos entre respostas.

A segunda técnica consiste em gerir as propriedades de consistência e isolamento do MongoDB, através de *read* e *write concerns*, semelhantes a *quorums* em conjuntos com réplicas para confirmar as operações de leitura e escrita, respetivamente. Estas ditam o nível da consistência e disponibilidade oferecida pela base de dados. Tendo isto em conta, a ideia é que o sistema ClearCache os aproveite para fazer uma gestão da cache consistente, consoante os *concerns* dos pedidos dos clientes. Aquando a inserção do documento no Redis, é adicionado um campo `_w` com um valor de *majority* ou *not_majority* dependendo do *write concern* utilizado para escrever o documento e determinar que clientes podem ler diretamente da cache.

Avaliação. Na nossa avaliação usámos o *Yahoo! Cloud Serving Benchmark* (YCSB) [3] e executámos as experiências num ambiente com um total de 3 máquinas: uma para a base de dados MongoDB, uma para a cache Redis e uma para o cliente do YCSB. Usámos as cargas de trabalho do YCSB, *update heavy*, *read mostly*, *read only* e *read latest*. Medimos o *throughput* e a latência para cada uma e constatámos que apenas na primeira o *throughtput* da aplicação com o ClearCache é ligeiramente menor do que não usando o nosso sistema. Isto é expetável, uma vez que mesmo sendo as leituras com o sistema mais rápidas, como as escritas são mais lentas, pois requerem lógica adicional, e a proporção destas operações é 50/50, o *throughput* da carga de trabalho é afetado negativamente. A Figura 1 apresenta a distribuição cumulativa de operações de escritas e leituras efetuadas com e sem o ClearCache por latência na carga de trabalho *update heavy*.

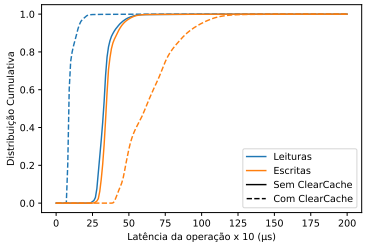

Fig. 1. Distribuição cumulativa das operações de leitura e escrita por latência.

Em todas as outras cargas de trabalho, o desempenho com o ClearCache é melhor, inclusive na *read only* onde o *throughput* é 4× superior.

Conclusão e Trabalho Futuro. Os resultados apresentados mostram que o ClearCache é bastante eficiente para cargas de trabalho maioritariamente constituídas por operações de leitura, com desempenhos superiores a aplicações que acedem diretamente à base de dados. Para além da eficiência, comprova-se ainda a simplicidade do seu uso.

De momento, o sistema ClearCache faz cache de objetos pelo seu `_id`. Todavia, pretendemos estender esta implementação, permitindo o uso da cache para outros campos únicos.

References

1. MongoBD, https://www.mongodb.com/
2. Redis, https://redis.io/
3. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with ycsb. In: Proceedings of the 1st ACM Symposium on Cloud Computing. p. 143–154. SoCC '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1807128.1807152, https://doi.org/10.1145/1807128.1807152
4. Mertz, J., Nunes, I.: Understanding application-level caching in web applications: A comprehensive introduction and survey of state-of-the-art approaches. ACM Comput. Surv. **50**(6) (nov 2017). https://doi.org/10.1145/3145813, https://doi.org/10.1145/3145813
5. Perez-Sorrosal, F., Patiño Martinez, M., Jimenez-Peris, R., Kemme, B.: Elastic si-cache: Consistent and scalable caching in multi-tier architectures. The VLDB Journal **20**(6), 841–865 (dec 2011). https://doi.org/10.1007/s00778-011-0228-8, https://doi.org/10.1007/s00778-011-0228-8
6. Ports, D.R.K., Clements, A.T., Zhang, I., Madden, S., Liskov, B.: Transactional consistency and automatic management in an application data cache. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. p. 279–292. OSDI'10, USENIX Association, USA (2010)