Unisanta Science and Technology, **2024, April**
Published Online 2024 Vol. 13 nº1 http://periodicos.unisanta.br/index.php/sat

UNISANTA
Science and Technology
ISSN 2317-1316

ESTUDO DE CASO DO USO DE CACHE DE OBJETOS EM UM SISTEMA WEB DE ORDEM DE SERVIÇOS

Helio Augusto de Lima Rangel, Nina Maria Bueno, Claudio Ferreira de Carvalho, Cláudio Souza Nunes, Leandro Teixeira Andrade, Luis Fernando Bueno Mauá, Paulo Roberto Schroeder de Souza

UNISANTA– Universidade Santa Cecília – Faculdade de Ciências Exatas, Engenharia e Arquitetura – Bacharelado em Sistemas de Informação e Tecnólogo em Análise e Desenvolvimento de Sistemas
Rua Oswaldo Cruz, 277 - Santos-SP, Brasil - CEP: 11045-907

E-mail*:* hlrangel@unisanta.br
Received February, 2024

Resumo: Este artigo descreve o estudo de caso de um sistema de médio porte que implementa a técnica de uso de cache de objetos, discutindo como foi implementado, vantagens e desvantagens. Problemas encontrados e como foram resolvidos ou mitigados.

Palavras-chave: Dados em memória, cache de objetos, sistemas distribuídos;

CASE STUDY OF USING OBJECT CACHE IN A SERVICE ORDER WEB SYSTEM

Abstract: This article describes a case study of a medium-sized system that implements the object cache technique, discussing how it was implemented, advantages and disadvantages. Problems encountered and how they were resolved or minimized.

Keywords: Data in memory, object cache, distributed systems.

1 Introdução

Na grande maioria dos sistemas comerciais trabalhamos com clientes, produtos, usuários, fornecedores, faturas, pedidos, ordens de serviço e outras identidades comuns no dia a dia dos negócios. Estes objetos são usados com frequência e precisam ser instanciados cada vez que o sistema faz uma referência a eles. A ideia do cache de objetos é manter esses objetos residentes em memória durante o período em que são referenciados com maior frequência, de forma que reduzam o número de consultas ao banco de dados. Neste artigo vamos mostrar o estudo de caso de um Sistema Web. Discutiremos as vantagens, desvantagens e dificuldades encontradas durante todo o tempo em que o sistema está operando. Esperamos com isso poder ajudar os arquitetos de software, fornecendo subsídios que poderão dar suporte na escolha da utilização ou não do cache de objetos e em quais situações seu uso pode ser vantajoso.

1. Desenvolvimento

1.1. O Sistema de Ordem de Serviços (SOS)

O SOS é um Sistema Distribuído (SD) WEB, desenvolvido em C# na plataforma ASP.NET com SQL Server e não utiliza nenhum Framework de persistência de mercado. O sistema é responsável pela geração e acompanhamento de Ordens de Serviço (OS) de uma empresa

multinacional da área de "*Quality Assurance*". O SOS controla toda a documentação envolvida no processo, todos os serviços da OS, geração e controle de faturamento, controle de amostras: análise química; geração e impressão de etiquetas; rastreamento das amostras nos destinos: coleta; laboratório; descarte ou arquivamento. O SOS é integrado ao SAP (*System Applications and Products in Data Processing*) e precisa estar sempre sincronizado com ele. Esta sincronização é feita, em ambos os sentidos, por intermédio de arquivos XML que são disponibilizados em pastas de arquivos e que os sistemas checam periodicamente para agir caso exista alguma atualização pendente. O SOS está em operação desde 2015, com mais de 100.000 OS's geradas até o momento.

1.2. O WebServer

Os sistemas ASP.NET precisam rodar em servidores que viabilizem o acesso a endereços na *internet* via protocolo HTTP. O servidor mais utilizado é o IIS (*Internet Information Services*) que é um WebServer, o ISS não se limita a aplicações ASP.NET, também pode hospedar outras aplicações como páginas HTML e páginas em PHP por exemplo. (BRITO, 2019)

1.2.1. Onde roda a aplicação

A aplicação é executada no servidor de aplicações, que é uma parte do ambiente de hospedagem do WebServer. Ele é responsável por executar o código da aplicação, interagindo com o banco de dados e outros serviços conforme o caso. (BRITO, 2019)

1.2.2. Diferença entre um servidor Web e um servidor de aplicações

Um servidor Web é um *software* que trabalha com dados estáticos, como imagens, arquivos texto etc., já um servidor de aplicações (dinâmico) executa uma aplicação, adicionando lógica de negócios as respostas do servidor Web. (AWS, 2024)

1.2.3. Mecanismos de armazenamento de dados (Escopos e Contextos)

Os servidores de aplicação WEB armazenam dados dos seus aplicativos em locais e formas distintas (contextos), cada uma delas procurando atender determinadas aplicações:

- Requisição (Request): A cada requisição http um novo contexto de *request* é criado pelo servidor e destruído no final do processamento; O tempo de vida deste escopo é muito limitado se prestando apenas para os propósitos imediatos de armazenamento do contexto da página; (FRANCO, 2016)

- Sessão (Session): Armazena variáveis e objetos para um usuário particular e persiste enquanto a sessão do usuário existir. A sessão não é um local apropriado para usar como cache pois os dados armazenados nela serão perdidos no momento que a sessão for fechada ou expirar. Um objeto instanciado e colocado apenas na sessão não será acessível pelas demais sessões ativas. Devemos cuidar para que nenhuma alteração nos atributos do objeto da sessão seja salva no banco de dados, uma vez que isso causará uma diferença nos atributos do objeto em memória e o os atributos do objeto no banco de dados, resultando em um grave problema de integridade; (MENDONÇA, 2017)

- Aplicação (Application): Armazena variáveis e objetos que são compartilhados com todos os usuários da aplicação ao mesmo tempo. É aqui neste contexto que preparamos, abastecemos e consultamos o cache de objetos. Enquanto a aplicação estiver no ar, os dados serão preservados e acessíveis a todos os usuários que estiverem acessando a aplicação. Caso um usuário altere qualquer atributo de um objeto em cache, todos os outros usuários logados na aplicação, terão o objeto atualizado no momento que as páginas forem atualizadas uma vez que o objeto é o mesmo compartilhado por todas as sessões. (MENDONÇA, 2017)

1.2.4. Como funciona o cache de objetos no contexto Application no SOS

Para cada tabela identidade do banco de dados, é criada uma classe onde todos seus atributos são replicados. Caso a tabela possua outras tabelas associadas, a classe instancia listas com todos os objetos associados para cada relação que exista, não importando a profundidade nem a quantidade de registros referenciados.

1.2.5. Abastecendo o cache

Como um atributo da classe "espelho" criamos uma lista estática (*escopo Application*) que receberá os objetos à medida que forem sendo instanciados. A lista de objetos do cache é abastecida no processo a seguir: Quando o sistema precisa acessar um dado objeto, ele verifica se este objeto já está em cache, se estiver, retorna o objeto encontrado, caso contrário, instancia o objeto procurado, adiciona na lista, classifica pelo número identificador (ID) e finalmente retorna o objeto recém instanciado.

1.2.6. Fazendo busca no cache

Todas as tabelas do sistema possuem chaves auto numeradas, as chaves ID, utilizadas para indexar os objetos. A busca na lista de objetos é feita por método que utiliza o algoritmo de *binary search*, método disponível do C#, utilizando o número ID como chave e, toda vez que é inserido um novo objeto, a lista cache é classificada pelo ID. Esta classificação é feita utilizando o método baseado no algoritmo *quick sort*, também disponível do C#;

1.2.7. Manipulando os objetos em cache

O programador acessa os objetos em cache de forma prática e ágil pois todos os atributos e subatributos estão lá disponíveis para serem utilizados. Toda a complexidade de acesso ao banco de dados fica encapsulada, as queries necessárias estão todas escritas nas classes do ADO (*ActiveX Data Objects*) responsáveis por instanciar e popular os objetos. Esta agilidade também é conseguida ao se utilizar *frameworks* de persistência, mas para conseguir isso o programador precisa mapear as tabelas e as relações envolvidas.

1.2.8. Controle de "Life Time" do cache

No SOS não existe critério de "expiração" de objetos na lista do cache, todavia, todos dias à meia noite o servidor é reiniciado e, portanto, o tempo máximo de vida do cache é de 24 horas. O administrador do sistema também pode a qualquer instante, reiniciar todas as listas das tabelas clicando no logo do sistema. Existem outras situações em que o cache é limpo: quando o servidor é parado e reiniciado, quando ocorre uma exceção não tratada na aplicação ou erro no banco de dados em uma tentativa de sincronização. Nos primeiros minutos de operação depois de uma reinicialização, nota-se uma certa lentidão, porém, em poucos minutos, à medida que o sistema vai sendo utilizado, o cache é abastecido e o sistema volta a seu estado normal.

1.3. Comparando as técnicas

Apenas como ilustração, vamos comparar o tempo de carga de um objeto diretamente do banco de dados em relação a busca do mesmo objeto via cache. O objeto escolhido para o teste é instanciado a partir da classe de OS, que é o objeto mais carregado do sistema. Associado a cada OS temos outras classes como Cliente, Responsável, lista de Serviços, lista de Amostras, lista de Contatos, lista de Documentos, Nota Fiscal etc. Cada uma dessas classes ainda pode ter outras classes associadas como é o caso de Cliente, por exemplo, que faz referência a Pessoa Jurídica, Contato, Endereço etc. Uma OS ainda pode fazer uma autorreferência se ela for uma OS desdobrada de uma outra OS.

1.3.1. O teste

O teste carregou 5000 OS's para memória, via consulta direta ao banco de dados, mediu o tempo gasto em milissegundos. Em seguida acessou estes 5000 registros, já em cache, e mediu o tempo médio que cada OS foi carregada. Optamos por usar 5000 registros para forçar algum custo computacional na busca no cache de OS's.

1.3.2. Os resultados

- Tempo para carga de 1 objeto OS via Banco de Dados: 710ms
- Tempo para carga de 1 objeto OS via Cache: 2ms

1.3.3. Análise dos resultados

A relação entre os tempos encontrada foi de 355 vezes, um número muito significativo, mas é preciso considerar que esta não é a relação real de tempo entre sistemas que fazem acesso direto ao banco e os que utilizam cache. Como vimos, no SOS o objeto, no primeiro acesso, é carregado por completo, carregando inclusive todos os objetos que se relacionam com ele. Nos sistemas convencionais, carregamos apenas os atributos que serão realmente utilizados o que representaria um tempo bem menor do que carregar o objeto completo. De qualquer forma, o tempo de 2ms para obter um objeto via cache parece ser imbatível mesmo para um acesso parcial ao banco de dados.

1.4. Casos em que não se recomenda o uso de cache

No item 2.3.2 vimos não apenas que o acesso ao objeto via cache com 2ms é um tempo bastante reduzido, mas que também o tempo para carga via banco de dados no primeiro acesso foi de 710ms, o que é uma eternidade em termos de tempo de acesso a banco.

1.4.1. Relatórios

Normalmente relatórios são filtrados por período, filial, clientes etc., as linhas do relatório geralmente se referem a objetos que já não estão sendo acessados no dia a dia, portanto não deveriam mais estar no cache. Os relatórios também mostram apenas algumas informações específicas do interesse do relatório. Neste caso o uso do cache não faz sentido. Imagine que um relatório de 5000 linhas de OS's levaria só para montar a lista de OS's quase uma

hora se considerasse que nenhum dos 5000 registros estaria em cache. O mais recomendado seria utilizar geradores de relatórios ou ferramentas de BI (Business Intelligence) para preparar os relatórios; (FERREIRA, 2022)

1.4.2. Uso de tabelas com atualização compartilhada com outros sistemas

Em um sistema que utiliza cache de objetos, existe o compromisso de manter os dados no banco de dados sempre sincronizados com os dados em cache. Fazemos isso cuidando para que toda alteração feita no objeto em cache seja imediatamente atualizada no banco de dados. Desta forma, caso o sistema saia do ar, ou venha a ser reiniciado por qualquer motivo, o objeto recuperado via banco, será idêntico a versão do objeto que o usuário trabalhava antes da queda do sistema.

No caso de um outro sistema acessar o mesmo banco de dados e realizar qualquer atualização em um registro que também esteja carregado em cache, o sistema por si só não tem como "saber" que o objeto em cache está diferente do objeto do banco de dados. Nesta situação, utilizamos algum mecanismo capaz de sinalizar que os dados sofreram alguma atualização e, portanto, será preciso recarregá-los. Uma das formas de fazer com que o banco sinalize alterações nas tabelas é utilizar Triggers, todavia o seu uso aumenta o acoplamento e reduz a simplicidade, o que parece diminuir as vantagens do uso de cache de objetos. (FERREIRA, 2022)

Triggers são procedimentos do SGBD (Sistema Gerenciador de Banco de Dados) executados automaticamente quando eventos como inserções, atualizações ou exclusões acontecem no banco de dados.

1.5. O Framework Object-Relational Mapping (ORM) - Entity Framework (EF)

Object-Relational Mapping é uma técnica para ajustar dados entre sistemas usando linguagens de programação orientadas a objetos. É um mecanismo que permite trabalhar com bancos de dados relacionais de uma maneira orientada a objetos, mapeando tabelas de banco de dados para classes e vice-versa. (KELLY, 2022)

O Entity Framework é um *open source* desenvolvido para o ambiente .NET, que permite trabalhar com dados usando objetos específicos do domínio, sem acesso direto as tabelas do banco de dados. Suporta o mapeamento de classes .NET para bancos de dados relacionais, fornecendo todas as funcionalidades manipulação de dados. O EF possui um cache de sessão que ajuda na redução no número de acessos sem, contudo, apresentar risco de integridade e é uma alternativa para os casos em que tabelas são compartilhadas. (BALTIERE, 2020)

Além do Entity Framework, existem outras bibliotecas ORM que podem ser usadas com ASP.NET, como o Dapper, que é conhecido por sua performance e simplicidade. Cada ORM tem suas próprias características e vantagens, e a escolha entre eles dependerá das necessidades específicas do projeto e das preferências do desenvolvedor. (KANJILAL, 2020)

1.5.1. ORM x Cache de Objetos

Não possuem a mesma aplicação, mas ambos têm em comum facilitar o uso do banco de dados, encapsulando boa parte da complexidade do manuseio e da conversão de objetos em tabelas e tabelas em objetos.

- O objetivo de um ORM é reduzir a complexidade de trabalhar com o banco de dados relacional em sistemas Orientado a Objetos;
- Cache de Objetos procura otimizar o desempenho ao armazenar dados em memória para acesso mais rápido. (RODRIGUES, 2023)

2. Conclusões:

Quando são observados os cuidados no uso de cache, ele se mostra como um grande facilitador na vida do desenvolvedor, simplificando o desenvolvimento e manutenção sem trazer nenhuma dificuldade adicional. O ganho de tempo é evidente na maior parte do tempo uma vez que os objetos mais acessados pelos usuários do sistema estarão em cache e não terão que ser recarregados todas as vezes. Para que o melhor do Cache de Objetos seja obtido, a arquitetura do sistema precisa ter sido projetada com esta finalidade.

2.1. Pontos positivos

Na experiência de alguns anos de operação de um sistema construído para operar com Cache de Objetos estes foram os pontos positivos que merecem destaque:

- O cache de objetos é um recurso simples e poderoso na redução do número de acessos a banco de dados. Além disso, agiliza o trabalho no desenvolvimento e manutenção de software. Quanto mais estável é um sistema, mais o uso de cache de objetos traz bons resultados uma vez que raramente o sistema é reiniciado por erro ou problema de segurança;

- Uma preocupação comum de que o uso de dados em cache pode vir a provocar exagerado consumo de memória, na verdade não se confirma. No caso do SOS, por exemplo, para evitar um acúmulo exagerado de dados no cache, ele é reiniciado a cada 24 horas (sempre a zero hora). Durante um dia normal de trabalho, mesmo no horário de pico de utilização, o consumo adicional de memória não representa nem 5% da disponível do servidor;

- Uma grande vantagem do uso de cache é a simplicidade. Pelo ponto de vista do desenvolvedor, é como se ele estivesse utilizado um ORM, lendo ou escrevendo nas propriedades do objeto, com foco apenas em obter, atualizar ou inserir novos objetos quando for o caso. O uso correto do cache de objetos pode facilitar o entendimento da lógica de negócio e, com isso, melhorar a agilidade e produtividade tanto na manutenção das antigas funcionalidades quanto na criação de novas.

2.2. Pontos Negativos

O trabalho com o cache de objetos apresenta alguns aspectos que podem limitar seu âmbito de aplicação. Estes foram os mais significativos:

- Colocar uma grande quantidade de registros em cache, como o que acontece na geração de relatórios, consome muito tempo e por isso não é uma boa utilização de cache de objetos. O acesso direto ao banco de dados, o uso de geradores de relatório ou ferramentas de BI, nesses casos, são as soluções mais apropriadas.

- Tabelas compartilhadas com outros sistemas também não são boas candidatas ao cache de objetos e alternativas que contornem esta limitação muitas vezes acabam consumindo mais recurso computacional do que fazer a carga direta dos registros via banco de dados.

- O péssimo hábito/recurso dos administradores do sistema realizarem atualizações diretas no banco de dados, também são um problema para o uso de cache de objetos. Depois de se proceder uma atualização direta na base de dados, o cache do sistema precisa ser reiniciado para que as modificações sejam novamente carregadas no cache do sistema;

- Os acessos ao banco de dados, principalmente acessos que atualizem atributos das tabelas, precisam ser bastante estáveis, sem erros que forcem o banco de dados a suspender as transações. Sempre que isso acontece, o sincronismo do cache passa a não ser mais confiável e o sistema precisa reiniciar as listas dos objetos envolvidos. Se este problema for frequente, o cache de objetos acaba por prejudicar ao invés de ajudar no desempenho.

Referências:

AWS. **Qual é a diferença entre um servidor Web e um servidor de aplicações?** Disponível em: https://aws.amazon.com/pt/compare/the-difference-between-web-server-and-application-server/. Acesso em: 29 mar. 2023

BALTIERE, André. **Criando um CRUD com Entity Framework**, 2020 Disponível em: https://balta.io/blog/ef-crud, acessado em 28 mar 2023

BRITO, Bruno. **Como Funciona o ISS**, 2019 Disponível em: https://www.brunobrito.net.br/anatomia-de-uma-aplicacao-asp-net-iis-parte-i. Acesso em: 29 set. 2023

FERREIRA, Breno C. **Estratégias de cache para Web APIs,** 2022. Disponível em: https://brenocferreira.medium.com/estrat%C3%A9gias-de-cache-para-web-apis-7a172fc26940. Acesso em 29 set 2023

FRANCO, Cristiano Roberto. **Programação Web II**. Disponível em: https://www.uniasselvi.com.br/ extranet/layout/request/trilha/materiais/livro/livro.php?codigo=21972., 2016. Acesso em 20 set 2023

KANJILAL, Joydip **Criando um CRUD com Entity Framework**. 2020 Disponível em: https://www.infoworld.com/article/ 3705055/how-to-map-object-relationships-using-dapper-in-aspnet-core.html. 2023. Acesso em 23 set 2023

KELLY, Gleiciane. **ORM (Object Relational Mapper)**, 2022. Disponível em https://blog.betrybe.com/tecnologia/orm/. Acesso em 02 abr. *2023*

MENDONÇA, Ewerton. **Linguagem e Programação para Web. Secretaria Executiva de Educação Profissional de Pernambuco**, 2017. Disponível em: https://sisa-

cad.educacao.pe.gov.br/bibliotecavirtual/bibliotecavirtual/texto/ Caderno_INFO_Linguagem_de_Programacao_para_Web_2017.2.pdf. Acesso em 28 set. 2023

RODRIGUES, Kauê. **ORM (Object Relational Mapper): Saiba o Que É**, 2023. Disponível em: https://blog.cubos.academy/orm-object-relational-mapper/. Acesso em 27 mar. 2023

