

Web Proxy Cache Server

José Ribeiro Nº61005 and Luís Vieira Nº61064

University of Minho, Department of Informatics, 4710-057 Braga, Portugal
e-mail: {a61005,a61064}@alunos.uminho.pt

Abstract. Relatório de anexo ao Trabalho de Programação

1 Introdução

Um Web Proxy Cache Server é um programa servidor que guarda em cache objectos que são pedidos pelos clientes, para que nos pedidos seguintes, o conteúdo do objecto seja directamente retornado sem que este seja transferido novamente pelo servidor.

Neste trabalho, o objectivo passa por implementar servidor que faça isto mesmo, tratando pedidos GET e controlando os acessos à Cache bem como as primitivas de armazenamento dos objectos com recurso aos Headers do HTTP/1.1 que contém informações sobre o objecto, tal como será visto...

O servidor foi implementado com recurso à linguagem de programação JAVA e o projecto que é entregue em anexo pertence ao IDE NetBeans, contendo todas as classes e devidamente comentadas, tal como o javadoc correspondente.

2 Protocolo HTTP

HTTP (Hypertext Transfer Protocol) é o protocolo da Camada de Aplicação utilizado para o tratamento de pedidos de objetos entre clientes e servidores na Web.

A versão actual do HTTP/1.1 implementa mecanismos que facilitam o controlo por cache, os quais iremos utilizar na criação do nosso Web Proxy Cache Server.

2.1 HTTP Request

Este protocolo utiliza o modelo cliente-servidor, onde um cliente (por exemplo um browser) cria uma ligação com um servidor, e envia um pedido que contém informações sobre o mesmo, tais como o uri e a versão do HTTP através de vários métodos, como o OPTIONS, GET, POST, HEAD, etc. Neste trabalho utilizamos apenas o método GET para tratamento de pedidos.

O servidor responde com uma linha de estado, onde inclui informações sobre a sua versão, erros ocorridos, entre outros, um header com informações sobre a resposta e ainda o objecto que foi pedido no corpo da resposta.

2.2 Método GET

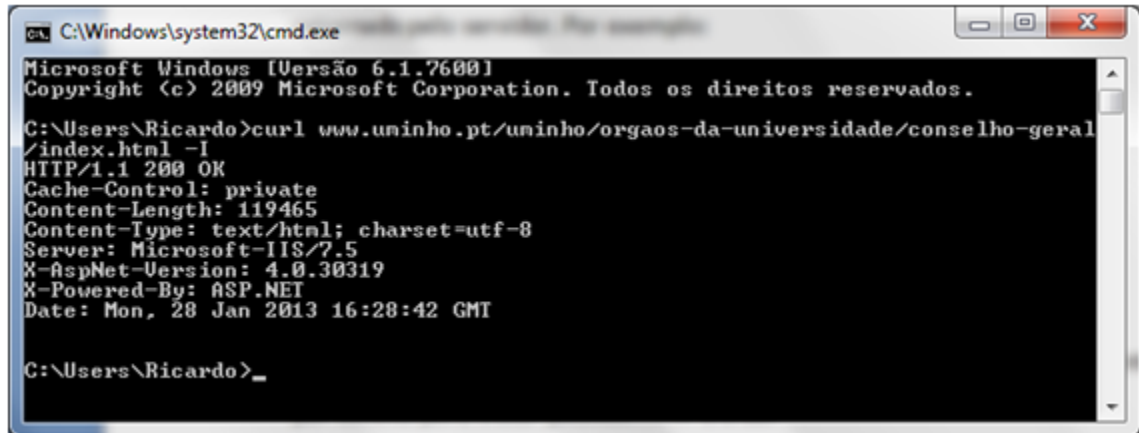
O GET funciona com um pedido de um cliente que contém a uri do objecto que será retornado pelo servidor. Por exemplo:

```
GET /uminho/orgaos-da-universidade/conselho-geral/index.html HTTP/1.1
Host: www.uminho.pt:80
|LINHA EM BRANCO|
```

A porta utilizada pelo HTTP aqui pode ser discriminada pelo Host, no entanto a porta por defeito para tratar pedidos HTTP é a 80.

2.3 HTTP Response

O servidor após receber um pedido irá enviar ao cliente uma resposta, como por exemplo para o pedido acima demonstrado, obtemos esta resposta (apenas os headers, obtidos com recurso ao curl com o parâmetro -I):



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versão 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Ricardo>curl www.uninho.pt/uninho/orgaos-da-universidade/conselho-geral
/index.html -I
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 119465
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Mon, 28 Jan 2013 16:28:42 GMT

C:\Users\Ricardo>_
```

Onde podemos observar a linha de estado, com a versão do http que é a 1.1, e a informação 200 OK que significa que o pedido foi bem-sucedido e a informação foi retornada. Contem ainda informações sobre a length, onde é colocada quantos bytes o corpo tem, o type do documento, html neste caso, a codificação utilizada, utf8, o controlo de cache, private que significa que não deverá ser guardado em cache exceto por user-agents, a data do pedido e informações sobre o servidor.

A informação sobre os cabeçalhos dos http Requests e http Responses podem ser consultados no website da w3, consórcio de standard da web, aqui serão explorados os cabeçalhos que determinam o funcionamento da cache.

3 Cache em HTTP

Os headers do HTTP/1.1 incluem um conjunto de elementos que podem ser utilizados para o controlo de cache, que tal como visto são inseridos nos pedidos e respostas do http.

Os cabeçalhos que permitem este controlo são:

- **Cache-Control:** Pode ter a opção no-cache que diz ao gestor da cache para não utilizar o objecto armazenado mas sim uma nova cópia do objecto. Tem ainda private que mantém o mesmo princípio mas permite user-agents controlar o pedido;
- **Max-Age:** Contém a idade máxima em segundos que um objecto pode ser guardado em cache;
- **Date:** Data em que o pedido é efectuado para ser comparado nas operações de expires e last-modified;
- **ETag:** Hash criada com a informação sobre o objecto, isto é, se a etag se mantiver, o objecto em cache é o actual e é esse que deve ser retornado;
- **Expires:** Contém a data até à qual o objecto deve ser guardado em cache, isto é, se o pedido tiver uma Date superior à Expires, o objecto deve ser pedido e actualizado na cache;

- **Last-Modified:** Contém a data de ultima modificação do ficheiro, pelo que se o objecto for modificado após a date guardada em cache, este deve ser actualizado;
- **Pragma:** no-cache: Que segue o mesmo principio do Cache-Control:no-cache.
- **Content-Length:** Apesar de não ser uma primitiva direta de controlo de cache, pode ser utilizada para verificar que caso a length do objecto guardado seja diferente do objecto remoto e caso os outros falhem a length sendo alterada ocorreu uma alteração no objecto. De notar que o objecto pode ser alterado e a length manter-se a mesma, logo este método não é fiável.

4 Web Proxy Cache Server

O nosso Servidor foi desenvolvido com a linguagem JAVA e o projeto é entregue juntamente com este relatório, devidamente comentado pelo que este relatório mostra a construção superficial do mesmo.

4.1 Classes do Projecto

- **Main:** Classe que inicia o servidor. É aqui que tem a criação do ServerSocket utilizado na porta 6069 e a criação das Threads para atribuir a cada cliente que pede um objecto;
- **Objecto:** Esta classe representa um objecto que é pedido pelo cliente, contendo informações sobre o mesmo entre as quais a data do pedido, a data de expires, a data de last-modified, a etag, a length, a idade máxima do objecto em cache e ainda o body (corpo) do objecto. Tem ainda métodos que permitem verificar a sua validação na cache, como verificar se está expirado, entre outros ;
- **Cache:** Esta classe representa a cache do sistema. É formada por um Map que guarda os objectos tendo como chave a sua uri e contém ainda métodos de verificar a existência de um objecto em cache, adicionar, remover entre outros;
- **ClienteHandler:** Instância de Thread que é atribuída a um cliente de forma a ser executada concorrentemente elevando a performance do servidor. É aqui que todo o processo é executado, desde a recepção do HTTP Request do cliente, toda a verificação da existência do objecto em cache, tratamento dos headers, pedido do objecto ao servidor e consequente HTTP Response de volta para o cliente. Contém ainda um ficheiro de logs que guarda os erros ocorridos no processo bem como um ficheiro de histórico onde são guardadas as uris que são acedidas, bem como a data do pedido;

O projecto do IDE Netbeans foi enviado junto a este relatório tal como a documentação javadoc do projecto contendo a informação sobre o projecto.

5 Conclusão

Um servidor proxy de cache pode ser bastante vantajoso no acesso a objectos da Web. Notou-se com o desenvolvimento deste projecto, que as ferramentas existentes na versão actual do HTTP/1.1 são muito uteis na implementação e controlo do sistema de cache, bem como os headers do request e da response que contêm informações sobre o objecto e ainda de fácil manipulação por parte do cliente.

De notar que este projecto, tal como o próprio conceito de Web Proxy Cache Server foi implementado com o intuito de servir vários clientes, podendo ser utilizado através do endereço e porta da máquina onde este está a correr, podendo-se formar uma cache partilhada entre vários utilizadores.

6 Bibliografia

Este projecto, relatório e código foi desenvolvido com recurso a pesquisa na internet, no entanto não foram apresentadas referências uma vez que essa bibliografia foi apenas utilizadas como pesquisa sobre o tema. Apresentamos de seguida os endereços então utilizados:

<http://www.jmarshall.com/easy/http>

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html>

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>

<http://raturajv.wordpress.com/2005/12/27/conditional-get-request>

<http://www.rnp.br/newsgen/0003/cache.html>

<http://www.symkat.com/understanding-http-caching>

<http://betterexplained.com/articles/how-to-optimize-your-site-with-http-caching>