UNIVERSIDADE D
COIMBRA

Googol Search Engine - Relatório

Sistemas Distribuídos - Meta 1

Marcelo Gomes Nº2021222994
Pedro Brites Nº2021226319

Índice

1. Arquitetura

Na **figura 1**, apresentamos um esquema que proporciona uma visão geral da arquitetura da aplicação.

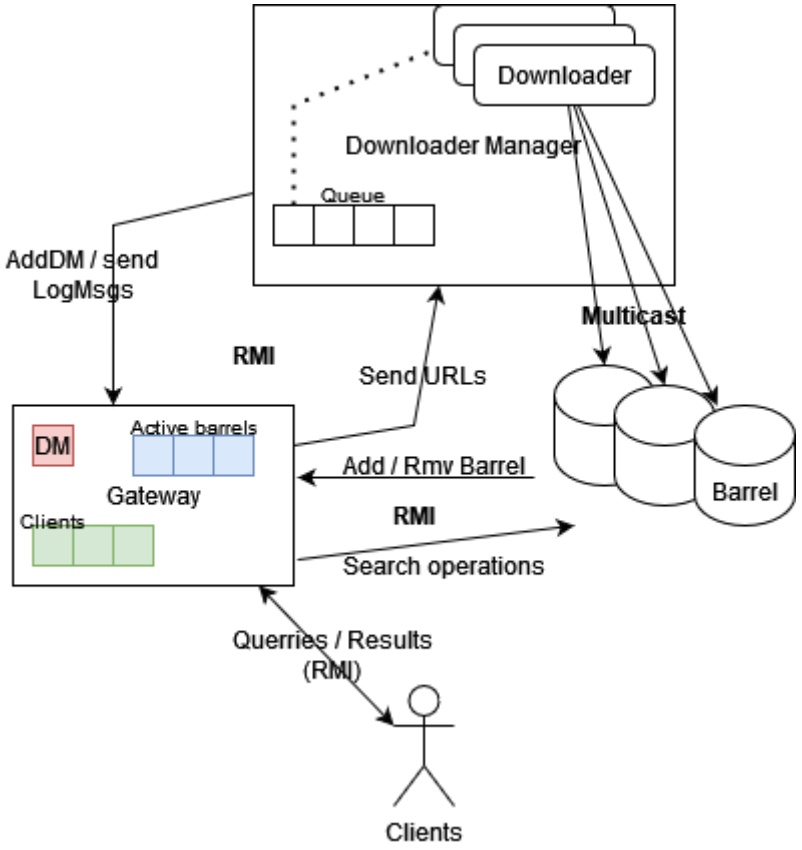

Fig. 1: Arquitetura do sistema.

- **Cliente-Gateway:** A ligação dos Clientes à Gateway é feita através de RMI. As interfaces dos Clientes ativos são guardados numa lista na Gateway.

- **Cliente:** O cliente envia pedidos à Gateway, que por sua vez retorna resultados.

- **Downloader-Gateway:** Temos um Downloader Manager que comunica com a Gateway via RMI. Devido a esta escolha só podemos ter 1 Downloaders Manager ativo na nossa aplicação.

- **Estrutura dos Downloaders e da queue:** Na nossa aplicação optamos por guardar a queue nos downloaders. Temos um Downloaders manager que cria as threads que vão processar os urls e enviá-los para os barrels via multicast.

A queue de urls é acessada por todas as threads e pelo Downloaders Manager. Quando o cliente faz um pedido para indexar um url, o Downloaders Manager adiciona-o à queue. As threads downloaders vão acessar esta queue e processar os respectivos urls, adicionando os urls resultantes recursivamente à queue.

- **Downloaders-Barrels:** As threads downloaders enviam os resultados aos barrels via multicast.

- **Barrels-Gateway:** As instâncias barrel comunicam com a gateway via RMI e são guardadas as interfaces dos barrels ativos numa lista.

- **Barrels:** Quando o barrel inicia é criado um certo número de instâncias de barrel, cada um com o seu ficheiro, que são respetivamente executados por threads. O ficheiro top10 é partilhado pelos barrels sendo atualizado cada vez que uma pesquisa é feita.

- **Gateway**: A gateway é um intermediário entre todos os componentes desta aplicação. Ela gere os clientes e barrels ativos e o downloaders manager. Encaminha pedidos dos clientes para os barrels e recebe logs vindos dos downloaders.

2. Componente Multicast

O downloader envia uma mensagem, através de ligação multicast UDP para todos os barrels usando o seguinte protocolo:

```
String message = "URL: " + url + "\nTitle: " + title + "\nCitation: " + citation + "\nKeywords: " + keywords + "\nLinks: " + urlsList;
```

Envia todas as informações necessárias separadas com um “\n” para poder separar no barrel.

```
String message = new String(packet.getData(), offset:0, packet.getLength());
String[] parts = message.split(regex:"\n");
String url = parts[0].replace(target:"URL: ", replacement:"");
String title = parts[1].replace(target:"Title: ", replacement:"");
String citation = parts[2].replace(target:"Citation: ", replacement:"");
String keywordsString = parts[3].replace(target:"Keywords: ", replacement:"").replace(target:"[", replacement:"").replace(targe…"]", "");
String linksString = parts[4].replace(target:"Links: ", replacement:"").replace(target:"[", replacement:"").replace(targe…"]", "");
```

No barrel, recebe toda a mensagem e divide para um array de strings, separando por “\n”. Após isso retira a parte que indica o que é, transformando numa string vazia e guarda a informação noutra string. No caso das **keywords** e do **UrlsList**, é preciso também separar os “[“ “]”, pois vem como array.

```
String[] keywords = keywordsString.split(regex:", ");
String[] links = linksString.split(regex:", ");
```

De seguida tiramos também as vírgulas e guardamos todas as informações separadas em arrays de strings.

3. RMI

Seguem os métodos remotos utilizados e a sua respectiva explicação.

3.1 IBarrel

- send(): Usado para receber o shutdown signal da gateway e limpar recursos antes de terminar.

- search(): Este método recebe a string vinda do cliente e divide-a por palavra, remove a pontuação e remove as “stopwords”. Após isto, adiciona-as ao ficheiro do top10searches. Para conseguir ver quais os links têm todas as palavras da pesquisa, vai a cada palavra e adiciona os links dessa palavra a uma lista auxiliar. Se for a primeira iteração, guarda na lista final de links. Nas próximas, compara os links da lista auxiliar com os da final e remove os links desta, se não estiverem na auxiliar. Caso não haja uma palavra da pesquisa no hashmap *invertedIndex*, retorna vazio. Caso contrário, ordena a lista por ordem decrescente, verificando a quantidade de links que apontam para cada, recorrendo ao hashmap *linkedPages* e depois vai construir a string com título, citação (presentes ao hashmap *title_citation*) e link para retornar.

- findSubLinks(): Este método recorre ao hashmap *linkedPage*, procurando o url recebido nas chaves e retornando o seu valor (que são os urls que apontam para o url da chave).

- getTop10Searches(): Este método lê, caso exista, o ficheiro “top10.dat” e guarda as informações num hashmap, que por sua vez é ordenado por ordem decrescente dos valores de cada chave. Por fim, retorna uma string com a todas as chaves e os seus respectivos valores.

- getId(): usado para obter o id do barrel.

3.2 IClient

- printOnClient(): Este método é usado para a gateway dar feedback ao cliente, por exemplo, se o cliente enviar um url inválido, a gateway responde com uma mensagem de erro que vai ser mostrada na UI.

3.3 IDownloader

- download(): Este método serve para a gateway enviar o url enviado pelo cliente para o Downloaders Manager e este adicioná-lo à queue de urls.
- send(): Usado para receber o shutdown signal da gateway e limpar recursos antes de terminar.

3.4 IGatewayBrl

- AddBrl(): Adiciona a interface do barrel à lista de barrels ativos e gera um id para o barrel. Este id pode ser novo ou reutilizado caso algum barrel tenha estragado no passado.
- RmvBrl(): Remove a interface do barrel da lista de barrels ativos e faz com que o seu id fique disponível para ser reutilizado.
- BrlMessage(): Usado para enviar mensagens de erros que ocorram nos barrels para o log da gateway.

3.5 IGatewayCli

- send(): Usado para enviar o url a indexar para a gateway. Caso o url seja inválido, o Downloaders Manager não esteja ativo ou não haja nenhum barrel ativo notifica o cliente via rmi-callback.
- subscribe(): Adiciona a interface do cliente à lista de clientes ativos.
- unsubscribe(): Remove a interface do cliente da lista de clientes ativos.
- search(): Escolhe um barrel aleatório para enviar a string e efetuar a operação de search. Retorna o resultado do search ou que não existem barrels ativos.
- findSubLinks(): Escolhe um barrel aleatório para enviar a string e efetuar a operação de encontrar os links que apontam para o url. Retorna os sub links, url inválido ou que não existem barrels ativos.
- getTop10Searches(): Escolhe um barrel aleatório para efetuar a operação de encontrar o top 10. Retorna o top 10 ou que não existem barrels ativos.
- getActiveBarrels(): Percorre a lista de barrels ativos obtendo o id de cada um e adiciona-o a uma string. Retorna a string ou que não há barrels ativos.

3.6 IGatewayDl

- AddDM(): Adiciona o Downloaders Manager à Gateway. Retorna false se já existir um Downloaders Manager ou true se for adicionado com sucesso.
- DlMessage(): Envia mensagens dos downloaders ,seja de erro ou informação, para o log da Gateway.
- RmvDM(): Remove o Downloaders Manager da Gateway. (Cria novo Downloaders Manager comentado, mais detalhes na tabela de testes).

4. Distribuição de tarefas

Nós optámos por fazer a seguinte divisão:

- Pedro Brites - Responsável pelos Barrels e pela componente multicast tanto nos Downloaders como nos Barrels
- Marcelo Gomes - Responsável pela Gateway, Client, Downloaders e ligações RMI.

5. Testes

Durante o desenvolvimento da aplicação, conduzimos alguns testes manuais para identificar possíveis erros. Apresentamos os respectivos resultados na **Tabela 1**.

ID	Descrição	PASS / FAIL
1	Cliente consegue indexar um novo URL	**PASS**
2	Cliente envia URL inválido não causa erros	**PASS**
3	Cliente faz um “search” e consegue ver título, citação e links que contenham uma palavra específica	**PASS**
4	Cliente faz um “search” e consegue ver os resultados 10 a 10	**PASS**
5	Cliente faz um “search” e consegue passar de página para ver novos resultados e voltar para a página anterior	**PASS**
6	Cliente faz um “search” sem resultados não apresenta menu de “search results” mas sim uma mensagem de alerta	**PASS**
7	Cliente faz um “search” sem ter introduzido novos links, mas aparecem resultados de links guardados de outras execuções.	**PASS**
8	Admin pede top10, e mostra sempre a mesma informação correta, apesar de acessar barrels diferentes para apresentar a informação	**PASS**
9	Admin pede top10 quando ainda não foram feitos “searches” não causa erros.	**PASS**
10	Admin pede top10 e aparecem os resultados ordenados por mais pesquisas	**PASS**
11	Admin pede top10 e consegue dar “refresh” para atualizar os dados das maiores pesquisas	**PASS**
12	Admin pede para ver os barrels ativos e aparecem corretamente os ids dos barrels ativos	**PASS**
13	A gateway suporta vários clientes ao mesmo tempo	**PASS**
14	Quando a gateway desliga, os outros programas também desligam com sucesso	**FAIL***
15	Quando o downloader manager vai abaixo e a gateway continua a correr a queue de URLs é guardada para se o DM recuperar	**PASS**
16	Ao ligar vários barrels de terminais diferentes os ids não se repetem	**PASS**

17	Ids de barrels que crasharam são recuperados por novos barrels	**PASS**
18	A gateway adiciona e remove os barrels da lista com sucesso	**PASS**
19	A gateway guarda e remove o DM com sucesso	**PASS**
20	Os downloaders esperam que a queue não esteja vazia, de forma a não causar uma espera ativa	**PASS**
21	UI funciona como pretendido não havendo bugs	**PASS**
22	O cliente envia os pedidos e recebe os resultados com sucesso	**PASS**
23	DM é recuperado pela gateway ao ir abaixo	**
24	A gateway apaga o ficheiro da queue se ele existir ao iniciar	**PASS**
25	Barrels terminam e são removidos da gateway ao receberem SIGINT	**PASS**
26	DM termina e é removido da gateway ao receber SIGINT	**PASS**
27	Se os barrels forem abaixo são recuperados pela gateway	**FAIL**
28	Barrels recuperam pacotes perdidos	**FAIL**
29	Sistema failover	**FAIL**

Tabela 1: testes realizados

* Apenas o DM e um client são terminados. Os barrels e os restantes clients não são terminados.

** Esta feature está comentada no nosso código visto que funciona mas como o shutdown da gateway não funciona como esperado, ao ligar a gateway novamente aparece um erro em que a port usada para o rmi ainda está em uso.