departamento de informática
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Concorrência e Paralelismo – 2013-14

Projecto 2: Introdução ao *Storm Project*

Luis Silva (n 34535º), Ricardo Gaspar (nº 42038)
Departamento de Informática – FCT – UNL
lmt.silva@campus.fct.unl.pt, rf.gaspar@campus.fct.unl.pt

Resumo — Resolver um problema com uma quantidade de dados elevada e cuja produtividade é um factor importante. A aplicação a desenvolver tem de ser capaz de realizar o processamento *online* de dados em tempo real. Além disso, esta deve ser escalável. Para se conseguir satisfazer estes requisitos foi utilizado *Storm Project*.

Palavras Chave — Paralelismo; *speedup*; *scaleup*; *streaming*.

Introdução

O trabalho consiste em desenvolver um programa utilizando o *Storm Project* que seja capaz de processar vários *tweets* geo-localizados extraindo informação neles contida por forma a realizar estudos. Tendo este trabalho um propósito académico, os *tweets* encontram-se armazenados em ficheiros de texto. Cada *dataset* contém meio milhão de *tweets*. Inserir os *tweets* para processamento implica ler um destes ficheiros.

A informação a extrair dos *tweets* está contida num conjunto de palavras que se quer estudar. Sobre estas pretende-se realizar dois estudos. O primeiro pretende saber quais as três palavras mais frequentes num dado ano e, ainda, perceber a sua evolução ao longo do mesmo. O segundo estudo incide sobre a localização de todas palavras do conjunto, querendo explorar quais os continentes onde estas mais ocorrem. No final, os resultados destes estudos são devolvidos em ficheiros para posterior análise.

O *Storm* é um sistema de processamento distribuído em tempo real. O sistema funciona por *streams* de dados e tem dois tipos de componentes, *spouts* e *bolts*, que actuam sobre estes. Um *spout* emite tuplos de dados. Já um *bolt* é capaz de processar e emitir tuplos que provenham de outros componentes, tanto *spouts* como *bolts*. Além disto, pode ser definido um número de *threads* que cada componente tem. Com estes é possível criar uma topologia que se adeqúe ao problema a resolver. A sua escolha influencia a capacidade de *scaleup* e de *speedup* do programa.

Abordagem

Dados os estudos a realizar, a topologia escolhida passou por definir um *spout* para a leitura do ficheiro de texto e cinco *bolts* para processamento de dados (ver fig. 1). O *spout* , *DataExtractorSpout*, lê *tweets* do ficheiro e emite-os para o resto da topologia. Os *bolts* estão organizados por níveis abaixo do *spout* por forma a realizarem processamento distinto. O primeiro *bolt*, designado de *Split*, recebe *tweets* emitidos pelo *DataExtractorSpout,* e divide cada um em vários atributos. No fim, o *bolt* emite tuplos com os seguintes atributos relativos ao *tweet* recebido: *word* – uma das palavras interessantes; *date* – a data e hora; *lat* – latitude da localização; *lon* – longitude da localização. Desta forma, o *Split* realiza o pré-processamento para os estudos a realizar por outros *bolts*.

Existem dois ramos de *bolts* abaixo do *Split*, um para cada estudo. O ramo da esquerda (ver fig. 1) realiza o estudo do *ranking* anual das palavras mais utilizadas. Este tem um primeiro *bolt*, *Date*, que filtra os tuplos emitidos pelo *Split*. A informação extraída incide apenas na palavra e na separação da data em ano e mês. No fim, emite estes atributos ao *bolt Counter* que realiza todo o processamento inerente ao estudo. Na prática, o *Counter* não emite mais nada, apenas devolve o resultado do estudo quando não houver mais tuplos. No entanto, este devolve as três palavras mais frequentes organizadas por ano com o respectivo número de ocorrências. Além disso, devolve a contagem das mesmas ao longo dos doze meses.

O segundo ramo realiza o estudo da localização das palavras que ocorreram ao longo do ficheiro lido. Para tal este ramo tem dois níveis. Num primeiro nível encontra-se o *bolt GeoCoder* que, ao receber tuplos emitidos pelo *Split*, verifica a localização da palavras baseada nas suas coordenadas geográficas. Por conseguinte, este *bolt* emite apenas o continente relativo ao *tweet* onde a palavra interessante ocorre. O *RegionCounter* encontra-se no nível seguinte, abaixo do *GeoCoder*. A sua função é contar o número de vezes que cada um dos continentes ocorre. Tal como acontece com o *bolt Counter*, o *RegionCounter* , na prática , não emite tuplos apenas escreve para um ficheiro o estudo realizado.

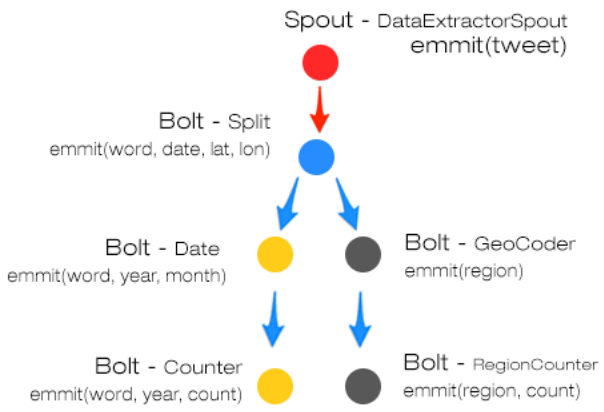


Figura 1 – Topologia utilizada.

Solução

A solução consistiu na implementação da topologia mencionada anteriormente. Esta implementação encontra-se na função *main()* descrita no ficheiro *WordCountTweet.java*. Neste está também definido, como um *hashMap,* o conjunto de palavras interessantes a serem filtradas nos *tweets*.

No que toca à implementação do *DataExtractorSpout*, foi utilizada a classe *DataExtractor* para efectuar toda a leitura dos *datasets*, bem como o *parsing* necessário para a criação de objectos do tipo *Tweet.* Este tipo está definido na classe *Tweet.java* e caracteriza-se pelos atributos: *tweet* – mensagem escrita por um utilizador; *latitude* – latitude da localização; *longitude* – longitude da localização; *userID* - identificador do utilizador; *date* – a data e hora da mensagem; *tweetKey* – identificador do *tweet.* Na classe *DataExtractor* foi optimizado o modo como se realiza a leitura dos ficheiros, assim como o *parsing*, por forma a acelerar a leitura dos *datasets* visto terem grandes dimensões e, portanto, um tempo de leitura maior. Ao invés de usar o *Scanner* do *java.util* usou-se o *BufferedReader* para efectuar a leitura à linha. Já para o *parsing* utilizou-se a função *split* da classe *String*.

No final do ramo esquerdo da topologia, está o *bolt Counter*, que se encontra implementado como *CountWordsByYear()*. Este é responsável por contar as palavras que lhe chegam e agrupá-las por ano, ao mesmo tempo que mantém uma contagem das mesmas ao longo dos meses de um dado ano. Para isto são necessárias duas estruturas. Um *hashMap* designado *wordYearlySum* que guarda por ano a ocorrência de palavras interessantes. A sua chave é o *Ano* e o seu valor é um outro *hashMap* com as palavras como chave e respectivas contagens como valor. Para o histórico mensal ao longo de um dado ano, está definido um outro *hashMap* denominado *wordYearlyCount,* cuja chave é a concatenação da palavra com o ano e o valor é um *array* com 12 posições para o número de ocorrências de cada mês. No final, quando não existem mais tuplos válidos, é necessário ordenar por valor todos mapas de *wordYearlySum*. Isto é, ordenar pelo maior número de ocorrências todos os *hashMap* de palavras respeitantes a cada ano. Assim, consegue-se criar uma tabela classificativa para todas as palavras. Esta ordenação é conseguida usando uma estrutura adicional ordenada que utiliza um comparador definido para usar os valores como comparação, ao invés das chaves.

Por fim, guarda-se num ficheiro *CSV* a classificação das três palavras mais frequentes por ano, bem como o seu histórico mensal.

O último *bolt* do ramo direito da topologia, denominado *RegionCounter*, é implementado por *CountWordsByRegion()*. A sua função é contar os continentes que recebe e agrupá-los. Para tal usou-se um *hashMap* em que a chave é o continente e o valor a respectiva contagem. O *RegionCounter* usa um objecto do tipo *GeoCoderContinents* para determinar o continente a partir das coordenas geográficas. Este tipo de objecto encontra-se implementado pela classe *GeoCoderContinents.java*. No final, quando não existem mais tuplos para contar, o *bolt* escreve para um ficheiro o conteúdo do mapa, bem como a contagem total.

O término da topologia foi considerado como o fim do *dataset*, assim que este acaba o *spout* apenas emite tuplos com os atributos a *null*. Desta forma permite-se que todos os *bolts* sejam informados que o *dataset* chegou ao fim. No entanto, para que se encerre a topologia é necessário indicar ao programa principal, *main()*, pois é este que tem o objecto referente à topologia. A estratégia utilizada para o conseguir foi declarar duas variáveis globais *volatile* do tipo booleano, que são inicializadas com o valor *false*. Assim que os *bolts* de último nível, *Counter* e *RegionCounter*, recebem a indicação para terminar, escrevem os seus resultados para os ficheiros e afectam imediatamente as variáveis com o valor *true*. Cada um destes *bolts* afecta apenas uma das variáveis. Assim o *main()* verifica, a cada segundo, o estado de ambas as variáveis e quando estas se encontram a *true*, encerra a topologia. A razão pela qual se usou variáveis *volatile*, foi o facto de estas não serem guardadas em *cache* obrigando a que sua leitura seja feita sempre a partir da memória principal.

Ainda no *main()*, na criação de cada componente da topologia é possível atribuir um nível de paralelismo. Este representa o número de *threads* que o componente tem para a sua execução. Neste sentido, foram definidos os seguintes valores: 1 para o *DataExtractorSpout*, visto só ler um ficheiro de cada vez; 1 para o *Split*; 4 para o *GeoCoder* e *Date* e 2 para o *Counter* e *RegionCounter*. A razão pela qual há *bolts* com o valor 4 prende-se com o facto de terem de processar mais tuplos. Há ainda a salientar o facto de se utilizar um modo de agrupamento denominado *fieldsGrouping* na criação dos componentes. Desta forma, os tuplos com a mesma *label* são atribuídos à mesma *thread* dentro de um componente.

Avaliação

A avaliação incidiu sobre dois estudos: o primeiro avalia, por ano, as três palavras que mais ocorrem e apresenta a variação da sua quantidade ao longo dos meses; o segundo tenta perceber os continentes onde as palavras mais interessantes aparecem com mais frequência.

Ambos os estudos foram realizados sobre um conjunto de três *datasets*, mais concretamente, os *datasets* 3, 5 e 7. Estes

foram aglomerados num ficheiro único de modo a criar um *dataset* maior.

Com estes estudos pretende-se explorar o uso da topologia do *Storm* por forma a conseguir uma aplicação que seja escalável e tenha um bom desempenho.

1. *Avaliação do Top 3 das palavras mais usadas por ano*

Neste estudo foram avaliados todos os *tweets* do *dataset* mencionado de modo a gerar os gráficos 1 e 2. O primeiro mostra as três palavras mais usadas nos *tweets* do anos presentes no *dataset*.

Gráfico 1 – Top 3 das palavras mais usadas por ano

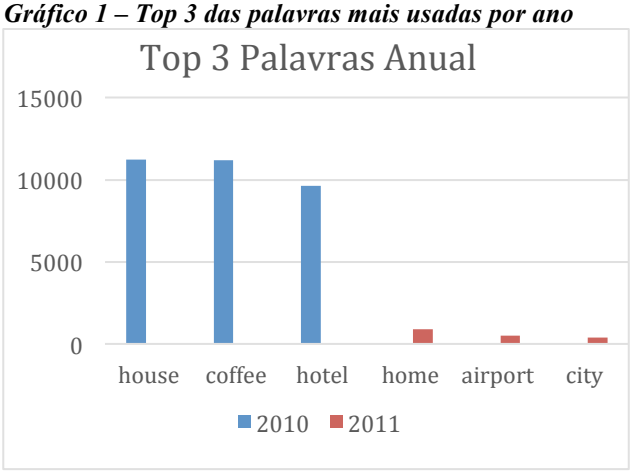
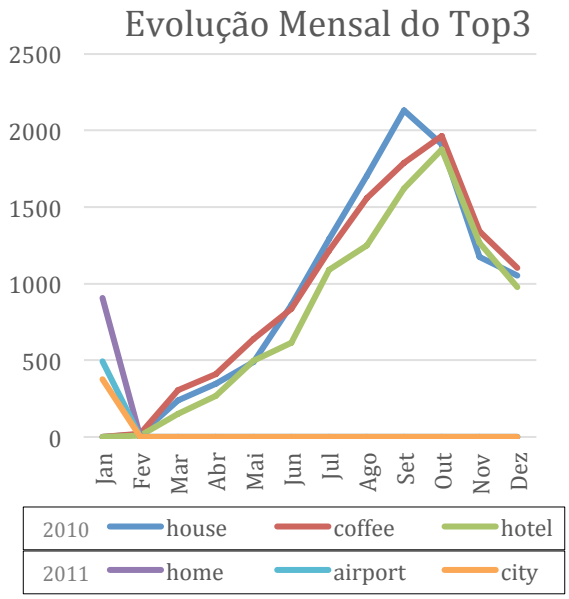


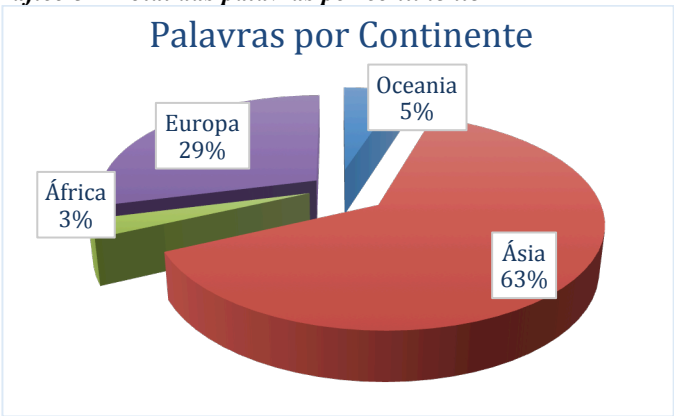
Gráfico 2 – Contagem das palavras por mês



2. *Avaliação das palavras por continente*

Este estudo contabilizou, por continente, todas as palavras interessantes presentes no *dataset* lido. O gráfico 3 mostra o total das palavras por continente.

Gráfico 3 – Total das palavras por continente



Conclusões

Os resultados das avaliações realizadas servem apenas para demonstrar as capacidades do *Storm*. Além disso, a concepção dos estudos a realizar surgiram como ponto de partida para concepção da topologia. Assim, é possível para cada problema adequar a topologia tendo em conta os requisitos a satisfazer. Neste caso, pretendeu-se estabelecer uma topologia que pudesse ser escalável (*scaleup*) e ao mesmo tempo garantir um desempenho aceitável (*speedup*).

O *Storm* não é uma implementação de *MapReduce*, no entanto foi uma óptima ferramenta para se por em prática os conhecimentos acerca do processamento em paralelo de grandes volumes de dados.

Referências

Storm Project –

https://github.com/nathanmarz/storm/wiki/Concepts

http://www.michael-noll.com/blog/2012/10/16/understanding-the-parallelism-of-a-storm-topology/

DataStream – http://en.wikipedia.org/wiki/Data_stream

Tweet – http://en.wikipedia.org/wiki/Tweet_(Twitter)

SpeedUp – http://en.wikipedia.org/wiki/Speedup

ScaleUp – http://en.wikipedia.org/wiki/Scalability

MapReduce – http://en.wikipedia.org/wiki/MapReduce

Paralelismo – http://en.wikipedia.org/wiki/Parallel_computing