MC970/MO644 – Programação Paralela

Laboratório 10 – Spark

Professor: Guido Araújo
Monitor: Hervé Yviquel, Maicol Gomez Zegarra

1 Analisador de Texto

Neste laboratório, iremos implementar um analisador/comparador de textos usando Apache Spark.

2 Enunciado

Neste exercício o objetivo é desenvolver uma analisador de texto usando Spark baseado na contagem de palavras que foi apresentada na aula. Para facilitar o exercício, o esqueleto do projeto é fornecido. O programa tem como argumento o caminho completo para dois arquivos de texto para analisar. O programa é divido em 2 partes:

- A primeira parte do programa deve contar as palavras e imprimir na saída padrão as 5 palavras de mais de 3 letras que tem as maiores ocorrências em ordem decrescente para cada texto. O programa deve considerar letras maiúsculas e minúsculas como sendo a mesma letra, além disso deve ignorar caracteres de pontuação para contar as ocorrências (usando `replaceAll("[,.!?:;]","")`). Aconselhamos de começar para dividir o texto a partir dos espaços, depois *limpar* a pontuação, e finalmente contar as palavras. Nota-se que o regex fornecido para limpar as palavras da pontuação não considera todos os casos mas produze as saídas esperadas por Parsusy.
- Na segunda parte, o programa deve comparar as analises dos 2 textos, e imprimir em ordem alfabética todas as palavras que aparecem mais de 100 vezes em cada um dos dois textos.
- A tercera tarefa é criar um cluster Spark no Microsoft Azure[1] e testar a execução na nuvem. Depois da execução, devem analisar a paralelização usando a interface gráfica de profiling do Spark.

[1]Tutorial: https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-jupyter-spark-sql

Cuidado de usar as funcões paralelas de Spark e não a biblioteca padrão de Scala.

Caso tenha alguma dúvida, use o Google Groups - para este trabalho está liberado discutir a solução direta do problema.

3 Testes e Resultado

Para compilar o seu programa, pode usar qualquer computador com Spark e sbt instalado (incluindo o servidor mo644 e as máquinas dos labs do IC), e digitar o comando seguinte na pasta própria do programa (contendo o arquivo *build.sbt*):

```
$ sbt package
```

Para executá-lo, basta digitar no mesmo computador:

```
$ spark-submit --class Analisador \
        target/scala-2.11/analisador_2.11-0.1.jar \
        texto1.txt texto2.txt
```

Há varias maneiras de executá-lo na nuvem mas aconselhamos conectar com SSH no cluster, copiar os arquivos, e usar spark-submit para rodar.

Os testes serão executados em 3 textos abertos e 1 fechado. Todas as combinações de textos serão testadas (então faz 3 testes abertos e 3 fechados). O programa deve imprimir os resultados na saída padrão respeitando o formato. As saídas esperadas dos testes abertos são fornecidas. Os arquivos de entrada contem somente texto sem formatação.

4 Submissões

A submissão deve ser **um arquivo** (em zip) contendo o **código Scala paralelizado** e um **relatório curto** (em pdf). O relatório deve comparar o tempo de execução serial no host e de execução paralela no cluster. Alem disso, o relatório deve conter uma captura de ecrã da interface gráfica de profiling do Spark mostrando a execução nos núcleos do cluster em paralela.