R on Spark: Utilizando Spark para processamento de grandes volumes em R

Gustavo Epifânio, Jefferson Colares

[1]CEFET - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
Rio de Janeiro - RJ - Brasil

gustavo.epifanio@eic.cefet-rj.br, jefferson.colares@eic.cefet-rj.br

Resumo. *R é a linguagem preferencial para desenvolvimento de aplicações e análises por muitos analistas e cientistas de dados. Entretanto, algumas limitações impedem sua utilização quando é necessário manipular um volume grande de dados.*
O Spark é um motor analítico desenvolvido especialmente para análise de grades volumes de dados e pode ser utilizado para viabilizar aplicações de alto desempenho em diversas linguagens, inclusive o R.
Nesse trabalho, mostramos como utilizar o R em conjunto com o Spark de forma a superar as limitações originais desta linguagem, comparamos as duas principais bibliotecas utilizadas para integrar as duas plataformas e demonstramos como o desempenho do R é melhorado com a utilização do Spark.

1. Introdução

Em seu site oficial, R é descrita como ”uma linguagem e um ambiente para computação estatística e gráfica. É um projeto de código aberto desenvolvido originalmente pela empresa norte-americana AT&T (na época, Bell Laboratories). Ela fornece uma ampla gama de funções estatísticas, como modelagem linear e não-linear, análise de séries temporais, classificação, clusterização e etc. e técincas de visualização gráfica de dados.” [The R-Project Contributors 2018].

R é a linguagem preferencial de milhares de analistas e cientistas de dados, especialmente aqueles que atuam com análise estatística ou possuem formação na área estatística.

O Spark é descrito pela Wikipedia como um framework de computação distribuída em clusters de uso geral. Originalmente desenvolvido na Universidade da California, seu código foi posteriormente doado à Apache Software Foundation, que é sua mantenedora desde então. O Spark provê uma interface para programação em clusters com paralelismo implícito e tolerância a falhas.” [Wikipedia contributors 2018].

O Spark facilita muito a vida dos desenvolvedores de diversas linguagens ao abstrair todas as complexidades da implementação de paralelismo e tolerência a falha em clusters de servidores.

Devido a algumas limitações, que serão detalhadas mais adiante, R não consegue manipular volumes muito grandes de dados. Uma das soluções para contorná-las é utilizar o Spark em conjunto com o R. Deste modo mantém-se o melhor dos dois mundos: os usuários podem continuar a ampla gama de funções estatísticas do R às quais já estão acostumados, ao mesmo tempo que usufruem das capacidades de manipulação de grandes volumes de dados fornecidos pelo Spark.

1.1. Limitações do R

- Processamento em memória
  O inteiro conjunto de dados a ser manipulado precisa caber na memória da máquina.
- Movimentação e duplicação de dados
  As etapas da manipulação dos dados em um script R envolvem a criação de dataframes intermediários. A utilização de memória aumenta conforme a complexidade do script.
- Sem paralelismo
  Ainda que diversos processadores estejam à disposição na máquina, R não é capaz de utiliza-los, pois não implementa paralelismo.

É importante observar que, ao longo dos últimos anos, diversas bibliotecas foram desenvolvidas com o objetivo de superar essas limitações. A utilização do Spark é apenas uma dentre muitas soluções possíveis. Entretanto, é indicada por sua simplicidade e transparência para o usuário, que continua utilizando as mesmas funções R habituais.

Para se beneficiar das capacidades oferecidas pelo Spark é necessário adicionar à instalação normal do R algumas bibliotecas. Pesquisamos as duas mais utilizadas: **SparkR** e **Sparklyr**. Uma terceira opção, que não abordamos nesse trabalho, é utilizar o Databricks, uma plataforma Spark em nuvem que suporta desenvolvimento em diferentes linguagens de programação, incluindo R, através de uma interface baseada em notebooks, como a do Jupyter Notebook.

1.2. SparkR vs. Sparklyr

SparkR (pronuncia-se ”spark-ar”) é uma biblioteca desenvolvida e mantida pela Apache Software Foundation, a mesma organização que mantém o Spark. Pode-se dizer que esta é uma iniciativa para fornecer ao Spark as funções já conhecidas pelos usuários R.

Sparklyr (proncuncia-se ”sparkli-ar”) é uma biblioteca desenvolvida pela RStudio, empresa responsável pela IDE que leva o mesmo nome e é uma das principais do ecossistema ao redor do R. Pode-se dizer que esta é uma iniciativa do R para fornecer a seus usuários as funcionalidades do Spark da forma mais transparente possível.

As duas opções são equivalentes, cada uma apresentando vantagens e desvantagens em relação a outra. Podem ser utilizadas simultaneamente, mas não em conjunto.

O data scientist e pesquisador Ed Berry, em seu blog, escreveu um excelente artigo comparando as duas bibliotecas cujos tópicos utilizamos como referência para nossas comparações [Berry 2018]. No geral, as diferenças que ele aponta entre as duas são consequência do seguinte: Sparklyr é totalmente compatível com a biblioteca Dplyr, que possui fortes capacidades de manipulação de dados (ambas são desenvolvidas pelo mesmo grupo). Deste modo, para os usuários do Sparklyr/Dplyr, a manipulação de dados com ou sem Spark é transparente. Sparkr busca reproduzir as funções mais populares entre os usuários de R, mantendo inclusive os mesmos nomes, o que provoca certos problemas para os usuários já habituados a elas, mas é mais eficiente e tem melhor desempenho no tratamento de dados, com destaque para a manipulação de datas.

2. Desenvolvimento

O presente trabalho foi desenvolvido utilizando um computador laptop comum, com 12GB de memória RAM, disco SSD de 128GB e um processador Intel Core i7 7500U de 2 núcleos e 2.70GHz.

O Spark foi instalado em modo standalone, sem utilização de YARN ou Mesos para escalonamento de processos em cluster.

Após a instalação do Spark, foi realizada a instalação padrão do R e do RStudio e por fim, as duas bibliotecas.

2.1. Instalação das Bibliotecas

Ambas as bibliotecas podem ser instaladas diretamente do console do RStudio.

A instalação do Sparklyr é mais simples e pode ser feita diretamente do repositório CRAM utilizando o comando abaixo:

```
install.packages("sparklyr")
```

Para instalar o SparkR, é necessario fazê-lo a partir de seu repositório no Github, mas isso é simples, utilizando o comando abaixo:

```
if (!require('devtools')) install.packages('devtools')
devtools::install_github('apache/spark@v2.3.2', subdir='R/pkg')
```

2.2. Configuração do Spark

Normalmente, para utilizar do Spark com uma única máquina local, não é necessário instalá-lo previamente, pois ambas as bibliotecas se encarregarão de fazê-lo sempre que não puderem encontrá-lo no diretório apontado pela varíável de ambiente SPARK_HOME ou em algum subdiretório do /home do usuário.

No ambiente de testes utilizado para esse trabalho, o Spark foi previamente instalado, de modo a conseguirmos simular um cluster com dois workers, cada um utilizando dois dos quatro cores disponíveis no computador.

Para simular esse cluster, criamos o shell script abaixo, que deve ser executado antes de usar o RStudio:

```
#Spark conf:
export SPARK_HOME=/usr/local/spark
export SPARK_WORKER_INSTANCES=2
export SPARK_WORKER_CORES=2

#start standalone server
$SPARK_HOME/sbin/start-master.sh

#start a worker
$SPARK_HOME/sbin/start-slave.sh spark://jeff-thinkpad:7077
```

2.3. Experimentações

O objetivo dos testes foi demonstrar os ganhos de desempenho da utilização do R com Spark em relação ao R ”puro”, sem spark. Adicionalmente, foi feita a simulação de um cluster para verificar se há ganho de desempenho quando se utiliza esta técnica, ainda que no mesmo equipamento físico. Os testes foram realizados em 5 cenários:

- Sparklyr com 1 worker
- Sparklyr com 2 workers
- SparkR com 1 worker
- SparkR com 2 workers
- R ”puro”

2.4. Dataset

Para as experimentações, foi utilizado o dataset wra-wu, contendo dados de partidas e chegadas de aeronaves em todos os aerodromos brasileiros no periodo de 2009 a 2017, combinado com os dados de condições climáticas a cada voo.

O dataset vra-wu possui pouco mais de 8 milhoes e 600 mil observações, com 34 variáveis cada. Em formato .CSV, o dataset ocupa aproximadamente 2GB em disco e pode ser obtido no endereço http://eic.cefet-rj.br/ eogasawara/brazilian-flight-datasets/.

2.5. Scripts de teste

Embora não seja possível executar um único script devido a diferenças entre as bibliotecas, as mesmas operações são executadas em todos os cenários, conforme a figura 1:

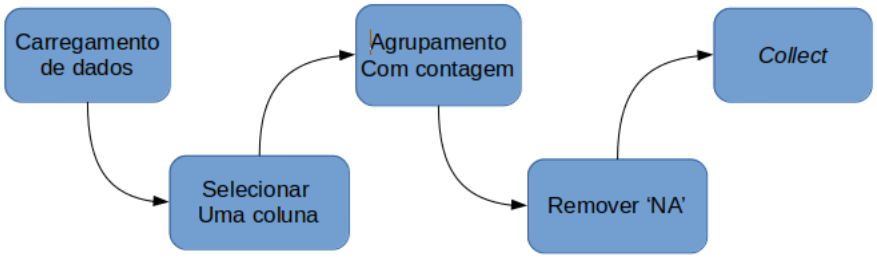

Figure 1. Sequência de operações dos testes

As funções tic() e toc(), da biblioteca tictoc foram utilizadas entre as operações para medir o tempo decorrido na execução de cada uma delas.

O script utilizado para os testes com a biblioteca SparkR foi:

```
# carga de arquivo csv
tic("carregar_dados")
teste <- read.df("/home/jeff/datasets/vra_wu/vra_wu.csv",
   "csv", header = "true")
toc()
# load_data
#preprocessar
tic("selecionar_coluna")
```

```
airlines<-select(teste, teste$airline)
toc()
#remover
tic("remover_na")
airlines<-dropna(airlines)
toc()
#contar
tic("contar")
results <- summarize(groupBy(airlines,airlines$airline),
   count(airlines$airline))
#resultst <- summarize(groupBy(dias,dias), count(dias))
toc()
#collect
tic("coletar_resultados")
collect(results)
toc()
```

O Script utilizado para testes com a biblioteca Sparklyr e com o R puro (sem Spark) foi:

```
# carga de arquivo csv
tic("carregar_dados")
teste <- spark_read_csv(sc, name = "wru", path =
   "/home/jeff/datasets/vra_wu/vra_wu.csv", header = TRUE,
   delimiter = ",")
toc()
#preprocessar
tic("selecionar_coluna")
airlines<-teste %>% select(airline)
toc()
#remover
tic("remover_na")
airlines<-na.omit(airlines)
toc()
#contar
tic("contar")
#results <- summarize(groupBy(airlines,airlines$airline),
   count(airlines$airline))
results <- airlines %>% group_by(airline) %>% summarise(count())
toc()
#collect
tic("coletar_resultados")
collect(results)
toc()
```

2.6. Execução dos testes

Cada script foi executado em cada cenário três vezes, coletando os tempos decorridos nas operações:

- carregar_dados
- selecionar_coluna
- remover_na
- contar
- coletar_resultados

Entre cada uma das execuções reinicializamos a máquina de testes de modo a termos o mesmo contexto de uso de memória e processador em todos eles.

Os dados apresentados na seção Resultados correspondem às médias das três execuções.

Também coletamos a quantidade de memória RAM utilizada em cada cenário.

3. Resultados

3.1. Carga de dados

O primeiro item do script de testes, tempo de carga de dados, foi analisado separadamente. Observamos que:

- O SparR apresenta um desempenho superior que os demais. Isso é coerente com o fato de que essa biblioteca manipula diretamente o RDD que contém os dados.
- Não há diferença considerável entre utilizar um ou dois workers na mesma máquina. Mesmo utilizando apenas um worker, as tarefas são paralelizadas e todos os recursos da máquina são utilizados. Sendo assim, aumentar o número de workers no mesmo nó, como tentamos simular, não oferece ganhos consideráveis de performance, especialmente na carga de dados.

A figura 2 mostra a comparação entre os valores médios obtidos nos testes em cada cenário:

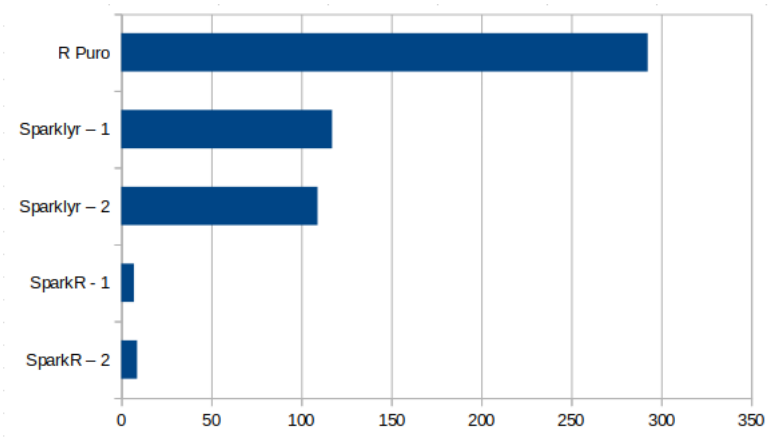

Figure 2. Tempo de carga dos dados

3.2. Tempo de processamento das operações

O processamento do script de testes em todos os cenários forneceu os seguintes valores médios de tempo de pricessamento para cada operação:

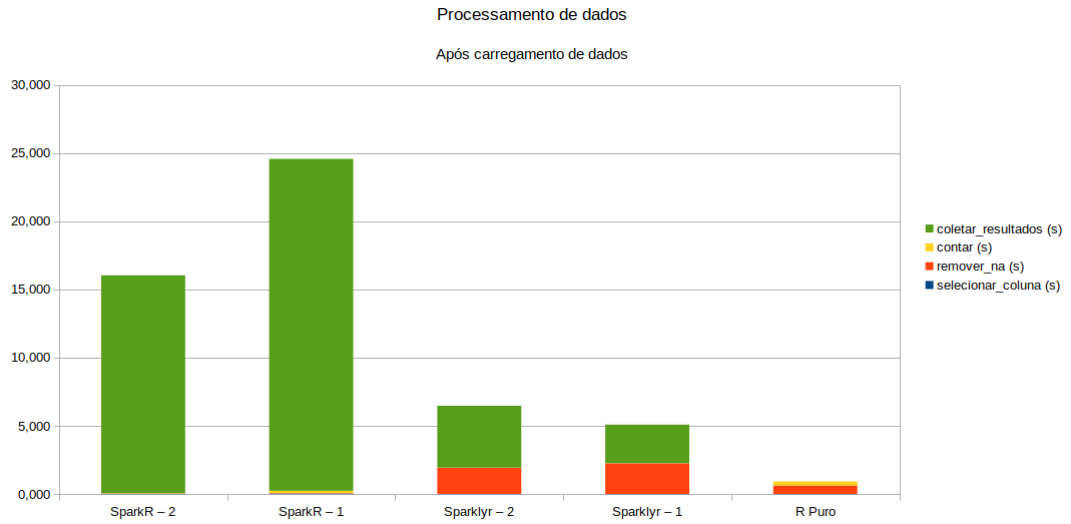

Figure 3. Tempo de processamento das operações

Os resultados obtidos mostram que:

- A utilização de dois workers, ainda que no mesmo nó, melhorou o desempenho das duas bibliotecas na operação de coleta de resultados.
- o Sparklyr é mais eficiente na operação collect (coletar_resultados).
- o R puro, por não paralelizar operações, não executa collects e por esse motivo aparenta ser o mais rápido.

3.3. Uso de memória

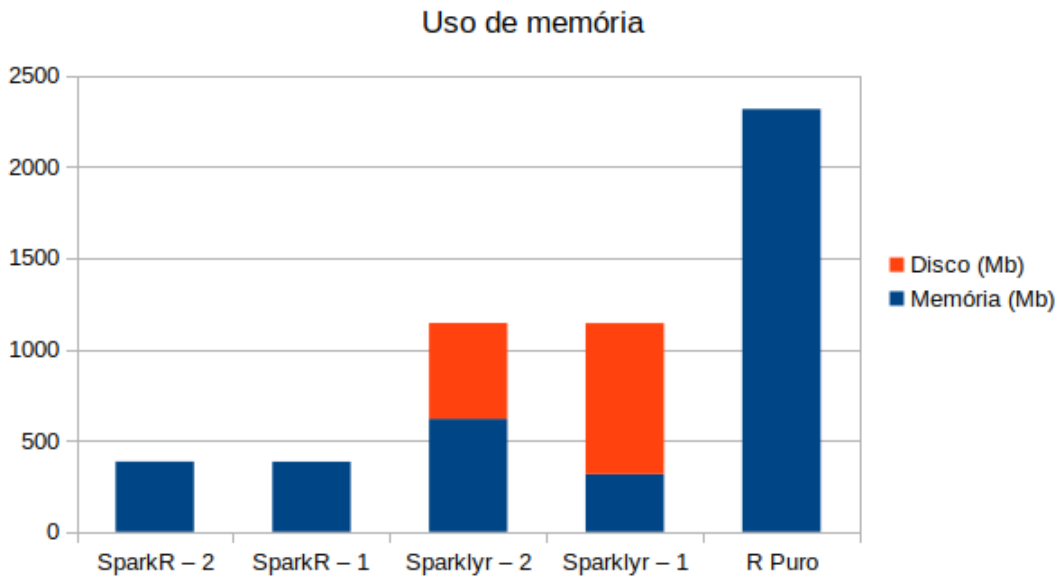

Figure 4. Uso de memória de cada cenário

Coletamos os dados de utilização de memória RAM e de disco em cada cenário de teste. Os resultados (figura 4) mostram que:

- O SparkR é mais eficiente no consumo de memória por se beneficiar dos algoritmos de compressão do Spark.
- O Sparklyr otimiza o consumo de memória fazendo uso do disco. Isso indica que a utilização de discos SSD, mais rápidos, pode otimizar o desempenho dessa biblioteca.
- O R puro trabalha com todo o dataset em memória e é o menos eficiente na utilização desse recurso.

3.4. Conclusões

Ao final dos testes e análises conclui-se que:

- Ambas as bibliotecas apresentam excelente desempenho e cumprem o prometido que é possibilitar a manipulação de grandes volumes de dados em R.
- Nas atividades que envolvem manipulação direta do RDD, o SparkR apresenta vantagem sobre o Sparklyr.
- O Sparklyr é mais fácil de usar, especialmente pelos usuários já habituados com R.
- o R puro, sem Spark, ainda apresenta excelente performance, e é boa opção desde que o volume de dados manipulados não seja grande.
- mesmo em um notebook comum, a utilização do Spark, seja com SparkR ou com Sparklyr, possibilita ao usuário manipular volumes de dados que não poderiam ser manipulados com o R puro.

References

Berry, E. (2018). SparkR vs Sparklyr for Interacting with Spark from R. [Online; acessado em 30/11/2018].

The R-Project Contributors (2018). What is R? [Online; acessado em 30/11/2018].

Wikipedia contributors (2018). Apache spark — Wikipedia, the free encyclopedia. [Online; acessado em 30/11/2018].