



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO



Título do trabalho

Felipe Cordeiro Alves Dias

Natal-RN

Mês (por extenso) e ano

Felipe Cordeiro Alves Dias

Título do trabalho

Monografia de Graduação apresentada ao Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do grau de bacharel em Ciência da Computação.

Orientador(a)

Nome e titulação do(a) professor(a) orientador(a)

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – UFRN
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA – DIMAP

Natal-RN

Mês (por extenso) e ano

Monografia de Graduação sob o título *Título da monografia* apresentada por Nome do aluno e aceita pelo Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Titulação e nome do(a) orientador(a)

Orientador(a)

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Co-orientador(a), se houver

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Departamento

Universidade

Natal-RN, data de aprovação (por extenso).

Agradeço a Deus por todas as oportunidades, pelo apoio da minha amada esposa, Laísa
Dias Brito Alves; família e amigos. Todos especialmente importantes para mim.

Agradecimentos

Agradeço a toda a minha família pelo incentivo e apoio recebidos; aos professores do curso de Engenharia de Software com os quais tive aprendizados importantes para a minha vida acadêmica, profissional e pessoal. Também, ao professor Nélio Alessandro Cacho, pela orientação, apoio e confiança. No demais, a todos que direta ou indiretamente fizeram parte da minha formação.

Queremos ter certezas e não dúvidas, resultados e não experiências, mas nem mesmo percebemos que as certezas só podem surgir através das dúvidas e os resultados somente através das experiências.

Carl Gustav Jung

Título do trabalho

Autor: Felipe Cordeiro Alves Dias

Orientador(a): Titulação e nome do(a) orientador(a)

RESUMO

O resumo deve apresentar de forma concisa os pontos relevantes de um texto, fornecendo uma visão rápida e clara do conteúdo e das conclusões do trabalho. O texto, redigido na forma impessoal do verbo, é constituído de uma seqüência de frases concisas e objetivas e não de uma simples enumeração de tópicos, não ultrapassando 500 palavras, seguido, logo abaixo, das palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores. Por fim, deve-se evitar, na redação do resumo, o uso de parágrafos (em geral resumos são escritos em parágrafo único), bem como de fórmulas, equações, diagramas e símbolos, optando-se, quando necessário, pela transcrição na forma extensa, além de não incluir citações bibliográficas.

Palavras-chave: Palavra-chave 1, Palavra-chave 2, Palavra-chave 3.

Título do trabalho (em língua estrangeira)

Author: Nome do aluno

Advisor: Titulação e nome do(a) orientador(a)

ABSTRACT

O resumo em língua estrangeira (em inglês *Abstract*, em espanhol *Resumen*, em francês *Résumé*) é uma versão do resumo escrito na língua vernácula para idioma de divulgação internacional. Ele deve apresentar as mesmas características do anterior (incluindo as mesmas palavras, isto é, seu conteúdo não deve diferir do resumo anterior), bem como ser seguido das palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores, na língua estrangeira. Embora a especificação abaixo considere o inglês como língua estrangeira (o mais comum), não fica impedido a adoção de outras linguas (a exemplo de espanhol ou francês) para redação do resumo em língua estrangeira.

Keywords: Keyword 1, Keyword 2, Keyword 3.

Lista de figuras

1	Ilustração da arquitetura em pilha do Apache Spark	p. 16
2	Componentes do Spark para execução distribuída	p. 18

Lista de tabelas

1	Tabela sem sentido	p. 22
---	------------------------------	-------

Lista de abreviaturas e siglas

RDD – Resilient Distributed Dataset

DStream – Discretized Stream

FIFO – First In, First Out

DAG – Directed Acyclic Graph

UFRN – Universidade Federal do Rio Grande do Norte

DIMAp – Departamento de Informática e Matemática Aplicada

YARN – Yet Another Resource Negotiator

Lista de símbolos

λ (algum símbolo)

Sumário

1	Introdução	p. 14
1.1	Organização do trabalho	p. 14
2	Fundamentação Teórica	p. 15
2.1	Cidades Inteligentes	p. 16
2.2	Plataformas de processamento em tempo real	p. 16
2.2.1	Apache Spark	p. 16
2.2.1.1	Módulos do Apache Spark	p. 16
2.2.2	Composição Interna do Apache Spark	p. 17
2.2.2.1	Modelo de execução das aplicações Spark	p. 17
2.2.2.2	Job	p. 18
2.2.2.3	Stage	p. 19
2.2.2.4	Task	p. 20
2.3	Seção 4	p. 20
3	Capítulo 3	p. 21
3.1	Seção 1	p. 22
3.2	Seção 2	p. 22
3.2.1	Subseção 2.1	p. 22
3.2.2	Subseção 2.2	p. 23
3.3	Seção 3	p. 23
4	Capítulo 4	p. 24

4.1	Seção 1	p. 24
4.2	Seção 2	p. 24
5	Capítulo 5	p. 25
5.1	Seção 1	p. 25
5.2	Seção 2	p. 25
5.2.1	Subseção 5.1	p. 25
5.2.2	Subseção 5.2	p. 25
5.3	Seção 3	p. 25
6	Considerações finais	p. 26
	Referências	p. 27
	Apêndice A – Primeiro apêndice	p. 28
	Anexo A – Primeiro anexo	p. 29

1 Introdução

A introdução é a parte inicial do texto e que possibilita uma visão geral de todo o trabalho, devendo constar a delimitação do assunto tratado, objetivos da pesquisa, motivação para o desenvolvimento da mesma e outros elementos necessários para situar o tema do trabalho.

1.1 Organização do trabalho

Nesta seção deve ser apresentado como está organizado o trabalho, sendo descrito, portanto, do que trata cada capítulo.

2 Fundamentação Teórica

Este é o primeiro capítulo da parte central do trabalho, isto é, o desenvolvimento, a parte mais extensa de todo o trabalho. Geralmente o desenvolvimento é dividido em capítulos, cada um com subseções e subseções, cujo tamanho e número de divisões variam em função da natureza do conteúdo do trabalho.

Em geral, a parte de desenvolvimento é subdividida em quatro subpartes:

- *contextualização ou definição do problema* – consiste em descrever a situação ou o contexto geral referente ao assunto em questão, devem constar informações atualizadas visando a proporcionar maior consistência ao trabalho;
- *referencial ou embasamento teórico* – texto no qual se deve apresentar os aspectos teóricos, isto é, os conceitos utilizados e a definição dos mesmos; nesta parte faz-se a revisão de literatura sobre o assunto, resumindo-se os resultados de estudos feitos por outros autores, cujas obras citadas e consultadas devem constar nas referências;
- *metodologia do trabalho ou procedimentos metodológicos* – deve constar o instrumental, os métodos e as técnicas aplicados para a elaboração do trabalho;
- *resultados* – devem ser apresentados, de forma objetiva, precisa e clara, tanto os resultados positivos quanto os negativos que foram obtidos com o desenvolvimento do trabalho, sendo feita uma discussão que consiste na avaliação circunstanciada, na qual se estabelecem relações, deduções e generalizações.

É recomendável que o número total de páginas referente à parte de desenvolvimento não ultrapasse 60 (sessenta) páginas.

2.1 Cidades Inteligentes

2.2 Plataformas de processamento em tempo real

2.2.1 Apache Spark

Apache Spark (Apache Software Foundation, 2016c) é uma plataforma de computação em cluster (unidade lógica composta por um conjunto de computadores conectados entre si através da Rede, permitindo compartilhamento de recursos (Techopedia, 2016)), com suporte a consultas a banco de dados, processamento de stream e aprendizado de máquina. Tendo Java, Python e Scala como linguagens de programação suportadas.

A sua arquitetura, conforme a Fig. 1, é composta por uma pilha integrando os seguintes componentes, a serem explicados posteriormente: Spark SQL, Spark Streaming, MLib, GraphX, Spark Core e Administradores de Cluster (Yarn (Apache Software Foundation, 2016a) e Apache Mesos (Apache Software Foundation, 2016b)). Tal estrutura visa ser de fácil manutenção, teste, deploy, e permitir aumento de performance do núcleo impactando seus demais componentes.

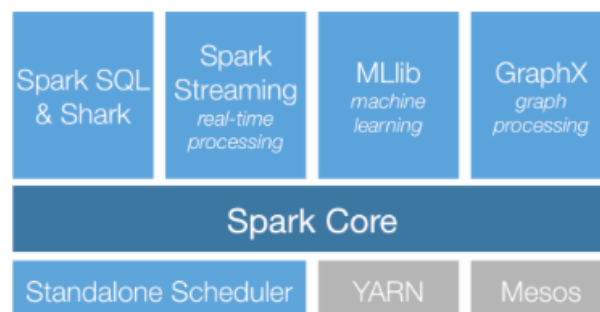


Figura 1: Ilustração da arquitetura em pilha do Apache Spark

Fonte: (KARAU et al., 2015)

2.2.1.1 Módulos do Apache Spark

O Spark Core é o principal módulo do Apache Spark, responsável principalmente pelo gerenciamento de memória, tasks (conceito explicado em 2.2.2.4) e tolerância a falha. Ainda nele, a abstração conhecida como RDD (Resilient Distributed Dataset) é definida, cujo papel é o de representar uma coleção de dados distribuídos e manipulados em paralelo. Os RDDs em Java são representados pela classe `JavaRDD`, criados através da classe `JavaSparkContext` (contexto da aplicação), que é instanciada recebendo como parâmetro um objeto `SparkConf`, contendo informações relacionadas ao nome da aplicação e o

endereço em que ela será executada, podendo ser local, ou, em cluster.

Outro módulo é o Spark Streaming, o qual realiza o processamento de dados em stream. Os streams são representados por uma sequência de RDDs, conhecida como DStream. O contexto de um streaming (JavaStreamingContext) é criado usando as configurações da aplicação e o intervalo no qual cada DStream será definido. Após isso, tais streams são atribuídos a um objeto da classe JavaReceiverInputDStream.

Além dos módulos detalhados nos parágrafos anteriores, é relevante mencionar o (i) Spark SQL, responsável por trabalhar com dados estruturados; o (ii) MLlib, relacionado ao aprendizado de máquina, contendo algoritmos tais como o de classificação, regressão, "clusterização", filtragem colaborativa, avaliação de modelo e importação de dados; e o (iii) GraphX, que é composto por uma biblioteca para manipulação de grafos e computações paralelas. Por fim, o Spark também possui um administrador de cluster padrão, conhecido como Standalone Scheduler (Apache Software Foundation, 2016c), tendo também suporte ao YARN e Apache Mesos.

2.2.2 Composição Interna do Apache Spark

Nesta seção é explicada a composição interna do Apache Spark, expondo os componentes que permitem a execução distribuída de uma aplicação spark, tais como o Driver Program, Spark Context, Worker Node e Executor. Também são explicados os conceitos sobre as operações de Transformação e Ação, e os relativos a Job, Stage, Task e Partição RDD.

2.2.2.1 Modelo de execução das aplicações Spark

O modelo de execução das aplicações Spark é definido pela relação composta por um driver, SparkContext, executors e tasks, conforme ilustrado na Fig. 2. Nessa interação, as aplicações Spark funcionam num Worker Node (qualquer nó capaz de executar código de aplicação localmente ou num cluster) como uma espécie de Driver, o qual executa operações paralelas e define dados distribuídos sobre um cluster. Além disso, o driver é responsável por encapsular a função principal da aplicação e prover acesso ao Spark, utilizando o objeto da classe SparkContext. Tal objeto é usado também para representar uma conexão com um cluster e construir RDDs (KARAU et al., 2015).

Após a construção de um RDD, é possível executar operações sobre ele. Essas operações são realizadas por executors, processos administrados por uma aplicação Spark

(cada aplicação tem os seus próprios executors); nos quais há espaços reservados para execução de tasks (conceito explicado na seção 2.2.2.4). Há dois tipos de operações: (i) Ações (actions) e (ii) Transformações (transformations) (Apache Software Foundation, 2016c). O primeiro tipo retorna um resultado ao driver após computações sobre um conjunto de dados de acordo com uma função. Sendo o segundo, responsável por gerar novos conjuntos de dados (RDDs) através, também, de funções especificadas (KARAU et al., 2015).

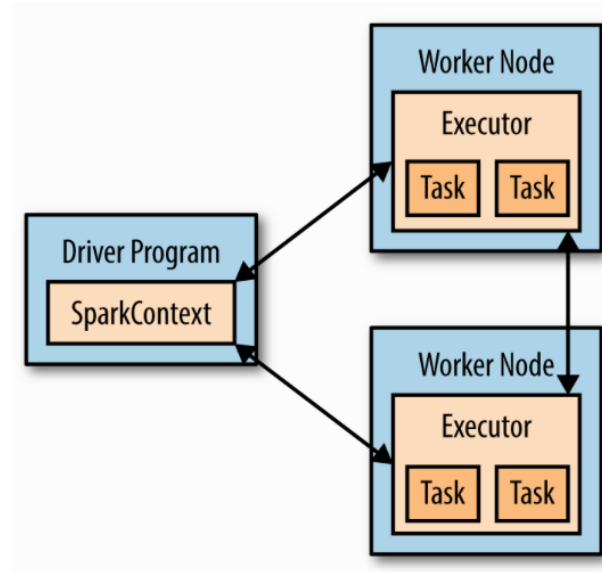


Figura 2: Componentes do Spark para execução distribuída
Fonte: (KARAU et al., 2015)

2.2.2.2 Job

A abstração do conceito de job está relacionada com o de action . Mais especificamente, um job é o conjunto de estágios (stages) resultantes de uma determinada action. Por exemplo, quando o método count() é invocado (para realizar a contagem de elementos dentro de um RDD), cria-se um job composto por um ou mais estágios. Os jobs, por padrão, são escalonados para serem executados em ordem FIFO (First In, First Out) . Além disso, o scheduler do Spark é responsável por criar um plano de execução física, o qual é utilizado para calcular as RDDs necessárias para executar a ação invocada (KARAU et al., 2015), (XU; JU; REN, 2016).

Em tal plano de execução física, defini-se basicamente um grafo acíclico dirigido DAG (Directed Acyclic Graph) , relacionando as dependências existentes entre os RDDs, numa espécie de linhagem de execução (lineage). Após isso, a partir da última RDD do último estágio, cada dependência é calculada, de trás para frente, para que posteriormente os

RDDs dependentes possam ser calculados, até que todas as actions necessárias tenham terminado (KARAU et al., 2015), (XU; JU; REN, 2016).

2.2.2.3 Stage

A divisão de um job resulta no conceito de stage, ou seja, jobs são compostos por um ou mais stages, os quais têm tasks a serem executadas. Nem sempre a relação entre stages e RDDs será de 1:1. No mais simples dos casos, para cada RDD há um stage, sendo possível nos mais complexos haver mais de um stage gerado por um único RDD (KARAU et al., 2015), (XU; JU; REN, 2016).

Por exemplo, havendo três RDDs no grafo RDD, não necessariamente haverá três stages. Um dos motivos para isso acontecer é quando ocorre pipelining, situação na qual é possível reduzir o número de stages, calculando os valores de alguns RDDs utilizando unicamente informações de seus antecessores, sem movimentação de dados de outros RDDs (KARAU et al., 2015), (XU; JU; REN, 2016).

Além da possibilidade do número de stages ser reduzido, o contrário acontece quando fronteiras entre stages são definidas. Tal situação ocorre porque algumas transformações, como a `reduceByKey` (função de agregação por chaves), podem resultar em reparticionamento do conjunto de dados para a computação de alguma saída, exigindo dados de outros RDDs para a formação de um único RDD. Assim, em cada fronteira entre os stages, tasks do estágio antecessor são responsáveis por escrever dados para o disco e buscar tasks no estágio sucessor, através da rede, resultando em um processo custoso, conhecido como `shuffling` (KARAU et al., 2015), (XU; JU; REN, 2016).

O número de partições entre o estágio predecessor e sucessor podem ser diferentes, sendo possível customizá-lo, embora isso não seja recomendável. Tal aconselhamento é devido ao fato de que se o número for modificado para uma pequena quantidade de partições, tais podem sofrer com tasks overloaded (sobrecarregadas), do contrário, havendo partições em excesso, resultará em um alto número de `shuffling` entre elas (KARAU et al., 2015), (XU; JU; REN, 2016).

Em resumo, `shuffling` sempre acontecerá quando for necessário obter dados de outras partições para computar um resultado, sendo esse um procedimento custoso devido ao número de operações de entrada e saída envolvendo: leitura/escrita em disco e transferência de dados através da rede. No entanto, se necessário, o Spark provê uma forma eficiente para reparticionamento, conhecida como `coalesce()`, a qual reduz o número de partições

sem causar movimentação de dados (KARAU et al., 2015).

2.2.2.4 Task

A abstração do conceito task está relacionada principalmente com stage, pois, uma vez que os estágios são definidos, tasks são criadas e enviadas para um scheduler interno. Além disso, tasks estão muito próximas da definição de partition, pelo fato de ser necessária a existência de uma task para cada partition, para executar as computações necessárias sobre ela (KARAU et al., 2015).

Portanto, o número de tasks em um stage é definido pelo número de partições existentes no RDD antecessor. E, por outro lado, o número de partições em um determinado RDD é igual ao número de partições existentes no RDD do qual ele depende. No entanto, há exceções em caso de (i) coalescing, processo o qual permite criar um RDD com menos partições que seu antecessor; (ii) união, sendo o número de partições resultado da soma do número de partições predecessoras; (iii) produto cartesiano, produto dos predecessores (KARAU et al., 2015).

Cada task internamente é dividida em algumas etapas, sendo elas: (i) carregamento dos dados de entrada, sejam eles provenientes de unidades de armazenamento (caso o RDD seja um RDD de entrada de dados), de um RDD existente (armazenado em memória cache), ou, de saídas de outro RDD (processo de shuffling), (ii) processamento da operação necessária para computação do RDD representado por ela, (iii) escrever a saída para o processo de shuffling, para um armazenamento externo ou para o driver (caso seja o RDD final de uma ação). Em resumo, uma task é responsável por coletar os dados de entrada, processá-los e, por fim, gerar uma saída (KARAU et al., 2015).

3 Capítulo 3

Algumas regras devem ser observadas na redação da monografia:

1. ser claro, preciso, direto, objetivo e conciso, utilizando frases curtas e evitando ordens inversas desnecessárias;
2. construir períodos com no máximo duas ou três linhas, bem como parágrafos com cinco linhas cheias, em média, e no máximo oito (ou seja, não construir parágrafos e períodos muito longos, pois isso cansa o(s) leitor(es) e pode fazer com que ele(s) percam a linha de raciocínio desenvolvida);
3. a simplicidade deve ser condição essencial do texto; a simplicidade do texto não implica necessariamente repetição de formas e frases desgastadas, uso exagerado de voz passiva (como *será iniciado*, *será realizado*), pobreza vocabular etc. Com palavras conhecidas de todos, é possível escrever de maneira original e criativa e produzir frases elegantes, variadas, fluentes e bem alinhavadas;
4. adotar como norma a ordem direta, por ser aquela que conduz mais facilmente o leitor à essência do texto, dispensando detalhes irrelevantes e indo diretamente ao que interessa, sem rodeios (verborragias);
5. não começar períodos ou parágrafos seguidos com a mesma palavra, nem usar repetidamente a mesma estrutura de frase;
6. desprezar as longas descrições e relatar o fato no menor número possível de palavras;
7. recorrer aos termos técnicos somente quando absolutamente indispensáveis e nesse caso colocar o seu significado entre parênteses (ou seja, não se deve admitir que todos os que lerão o trabalho já dispõem de algum conhecimento desenvolvido no mesmo);
8. dispensar palavras e formas empoladas ou rebuscadas, que tentem transmitir ao leitor mera idéia de erudição;

9. não perder de vista o universo vocabular do leitor, adotando a seguinte regra prática:
nunca escrever o que não se diria;
10. termos coloquiais ou de gíria devem ser usados com *extrema* parcimônia (ou mesmo nem serem utilizados) e apenas em casos muito especiais, para não darem ao leitor a idéia de vulgaridade e descaracterizar o trabalho;
11. ser rigoroso na escolha das palavras do texto, desconfiando dos sinônimos perfeitos ou de termos que sirvam para todas as ocasiões; em geral, há uma palavra para definir uma situação;
12. encadear o assunto de maneira suave e harmoniosa, evitando a criação de um texto onde os parágrafos se sucedem uns aos outros como compartimentos estanques, sem nenhuma fluência entre si;
13. ter um extremo cuidado durante a redação do texto, principalmente com relação às regras gramaticais e ortográficas da língua; geralmente todo o texto é escrito na forma impessoal do verbo, não se utilizando, portanto, de termos em primeira pessoa, seja do plural ou do singular.

3.1 Seção 1

Teste de uma tabela:

Tabela 1: Tabela sem sentido

Titulo Coluna 1	Título Coluna 2
X	Y
X	W

3.2 Seção 2

Seção 2

3.2.1 Subseção 2.1

Referência à tabela definida no início: 3.1

3.2.2 Subseção 2.2

Subsection 2.2

3.3 Seção 3

Seção 3

4 Capítulo 4

4.1 Seção 1

Teste para símbolo

λ

4.2 Seção 2

Teste para abreviatura

UFRN

DIMAp

5 Capítulo 5

5.1 Seção 1

Seção 1

5.2 Seção 2

5.2.1 Subseção 5.1

Subseção 5.1

5.2.2 Subseção 5.2

Subsection 5.2

5.3 Seção 3

Seção 3

6 Considerações finais

As considerações finais formam a parte final (fechamento) do texto, sendo dito de forma resumida (1) o que foi desenvolvido no presente trabalho e quais os resultados do mesmo, (2) o que se pôde concluir após o desenvolvimento bem como as principais contribuições do trabalho, e (3) perspectivas para o desenvolvimento de trabalhos futuros. O texto referente às considerações finais do autor deve salientar a extensão e os resultados da contribuição do trabalho e os argumentos utilizados estar baseados em dados comprovados e fundamentados nos resultados e na discussão do texto, contendo deduções lógicas correspondentes aos objetivos do trabalho, propostos inicialmente.

Referências

Apache Software Foundation. *Apache Hadoop YARN*. 2016. Disponível em: <<http://hadoop.apache.org>>. Acesso em Abril 22, 2016.

Apache Software Foundation. *Apache Mesos*. 2016. Disponível em: <<http://mesos.apache.org/>>. Acesso em Abril 23, 2016.

Apache Software Foundation. *Apache Spark: Yet Another Resource Negotiator*. 2016. Disponível em: <<http://spark.apache.org>>. Acesso em Abril 22, 2016.

KARAU, H. et al. *Learning Spark Lightning-Fast Big Data Analysis*. 1st. ed. [S.l.]: O'Reilly Media, 2015.

Techopedia. *Computer Cluster Definition*. 2016. Disponível em: <<https://www.techopedia.com/definition/6581/computer-cluster>>. Acesso em Abril 23, 2016.

XU, L.; JU, H.; REN, H. *Gitbook: SparkInternals*. 2016. Disponível em: <<https://github.com/JerryLead/SparkInternals>>. Acesso em Abril 30, 2016.

APÊNDICE A – Primeiro apêndice

Os apêndices são textos ou documentos elaborados pelo autor, a fim de complementar sua argumentação, sem prejuízo da unidade nuclear do trabalho.

ANEXO A – Primeiro anexo

Os anexos são textos ou documentos não elaborado pelo autor, que servem de fundamentação, comprovação e ilustração.