**ESTUDO DA PLATAFORMA DE COMPUTAÇÃO EM *CLUSTER* APACHE SPARK VOLTADA AO *MACHINE LEARNING***

ARMINDO, Thiado de Almeida. DEZANI, Henrique

e-mail:

thiago.armindo@fatec.sp.gov.br; henrique.dezani@fatec.sp.gov.br

**Resumo:** a proposta deste estudo visa analisar a ferramenta Apache Spark aplicada a computação em *cluster*, voltada ao *Machine Learning*, identificando os recursos oferecidos, e conceitos por trás da tecnologia, bem como o funcionamento da mesma. E tendo isso em vista, utilizar essa ferramenta em um estudo prático, para a criação de um modelo de algoritmo de *Machine Learning*.

**Palavras-chave:** Spark. *Machine Learning*. Computação em *cluster*.

***Abstract:*** *The purpose of this study is to analyze the Apache Spark tool applied to cluster computing, aimed at Machine Learning, identifying the resources offered, and concepts behind the technology, as well as its functioning. And with this in mind, use this tool in a practical study, for the creation of a Machine Learning algorithm model.*

***Keywords:*** *Apache Spark, Machine Learning. Cluster computing.*

1. **Introdução**

A computação em *cluster* surgiu como uma alternativa viável e comercial para o paralelismo, teve seu surgimento na década de 1960 com a IBM, como é definido por Gomes (2015), *Cluster* é um termo em inglês que significa “aglomerar” ou “aglomeração” e pode ser aplicado em vários contextos. No caso da computação, o termo define uma arquitetura de sistema capaz combinar vários computadores para trabalharem em conjunto ou pode denominar o grupo em si de computadores combinados.

Cada estação é denominada “nodo” e, combinadas, formam o *cluster*. Em alguns casos, é possível ver referências como “supercomputadores” ou “computação em cluster” para o mesmo cenário, representando o *hardware* usado ou o *software* especialmente desenvolvido para conseguir combinar esses equipamentos.

O Apache Spark se enquadra em um desses *softwares* para realização da clusterização, ele é um *framework* de código aberto, desenvolvido pela universidade de Barkeley em 2009, seu objetivo principal é processar grandes conjuntos de dados de forma paralela e distribuída, sendo algumas de suas funções o gerenciamento de *BigData*, aplicações de *Machine Learning*, *Streaming* de Dados, SQL e processamento de Grafos.

Desse modo, se pretende analisar e estudar essa poderosa ferramenta, combinando seus recursos de *Streaming* de Dados, para processamento clusterizado de algoritmos de *Machine Learning*.

1. **Justificativa**

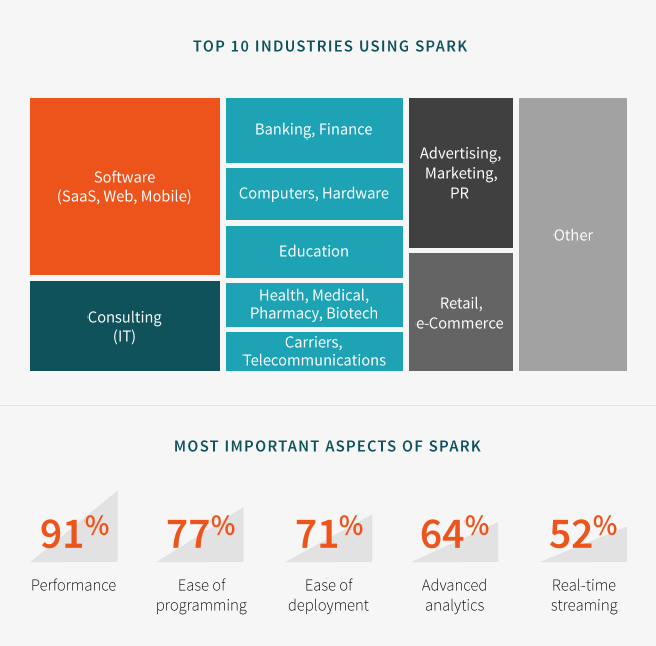
A popularização do uso de dispositivos eletrônicos no dia-a-dia, como *smartphones*, *notebooks* e computadores, gerou uma “pegada” dos usuários nos sistemas digitais, devido aos dados gerados diariamente por eles, quando se está utilizando algum aparelho. “De acordo com o Instituto Gartner, até 2020 é possível que haja um total de 40 trilhões de *gigabytes* de dados no mundo. Isso significa 2,2 milhões de *terabytes* de novos dados gerados todos os dias.” (NAVITA, 2019).

Com toda essa quantidade de dados, as empresas enxergaram a possibilidade de estudo do perfil de seus clientes, o setor de saúde observou a possibilidades de traçar padrões para identificar doenças, e no setor ambiental é possível identificar os sinais da natureza, e prever possíveis desastres naturais, segundo o CEO da Mastercard em entrevista para a revista Época Negócios (2019), os dados são o novo petróleo. As técnicas de Inteligência Artificial e *Machine* *learning* emergiram como uma forma de aplicar e desenvolver soluções em cima dos dados obtidos, conseguindo grande destaque e relevância para o crescimento das empresas, instituições e pesquisas nas mais diversas áreas.

De acordo com Schoch (2018), os primeiros registros de uso do *machine* *learning* são datados de aproximadamente 1950, um desses registros foi feito pelo pesquisador da IBM Arthur Lee Samuels, que desenvolveu um programa com a capacidade de auto aprendizado para jogar damas.

Todavia, ao aplicar esta técnica, em um grande conjunto de dados, é necessário um grande poder computacional, quanto mais dados analisados, mais preciso são os modelos criados, e mais recursos são necessários para realizar o processamento, sendo assim, o Apache Spark desenvolveu o recurso de clusterização do processamento de modelos de *machine* *learning*, distribuindo o peso dessa tarefa em vários núcleos de processamento, que podem ser processadores físicos ou virtuais, isso viabilizou o uso da tecnologia em diversas empresas, “Potências da Internet como Netflix, Yahoo e eBay o implementaram em escala maciça, processando coletivamente múltiplos *petabytes* de dados em clusters de mais de 8.000 nós. Ele rapidamente se tornou a maior comunidade *open* *source* em *BigData*, com mais de 1000 colaboradores de mais de 250 organizações.”(GARCIA, 2020)

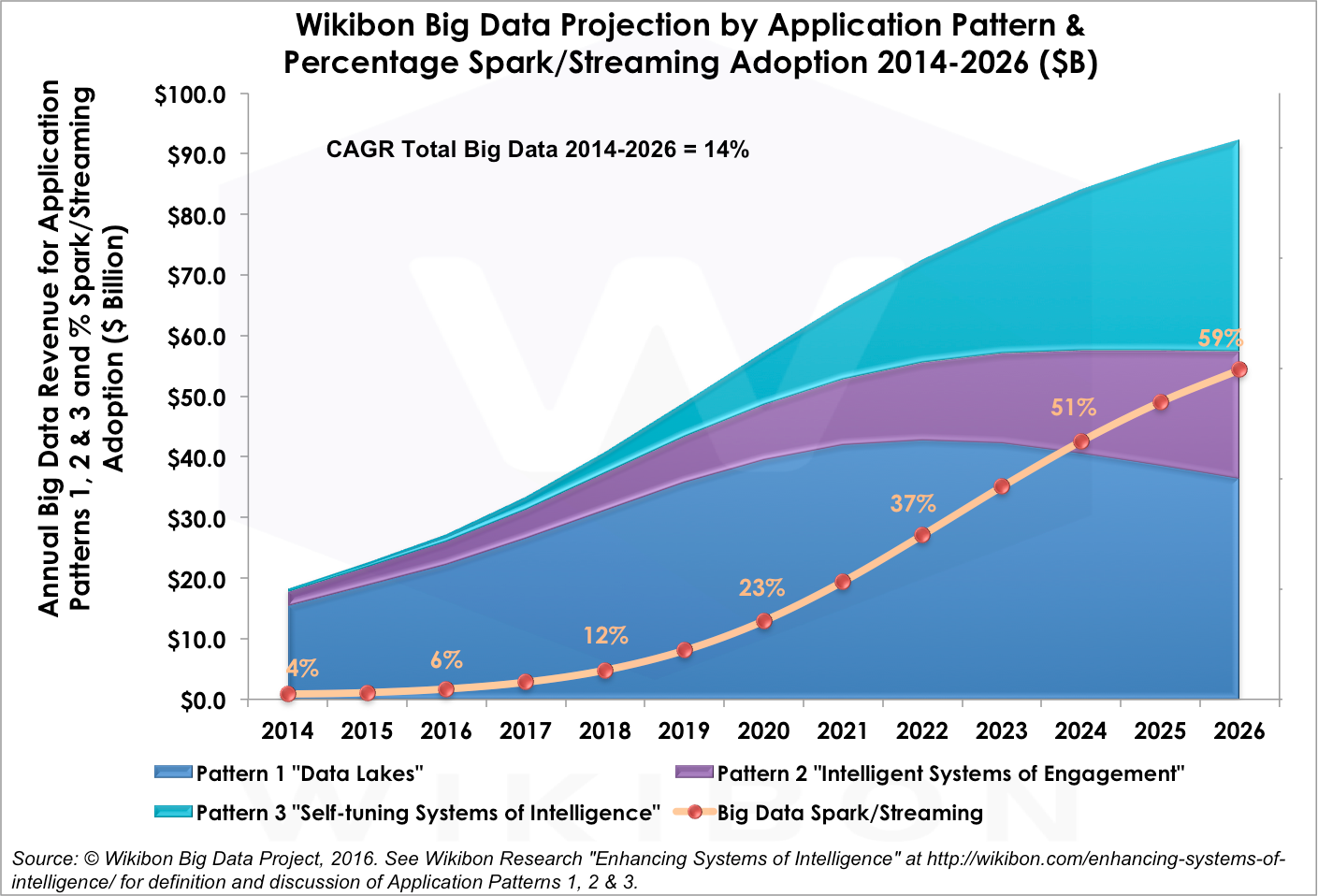
**Figura 1 Top 10 industrias utilizando Spark e seus aspectos mais importantes.**



**Fonte: (Gutierrez, 2015)**

Na figura 1 é possível observar os principais setores que estão utilizando o Spark, apesar da maioria ser do setor de *software*, ele está presente em diferentes setores, como educação, e *marketing* e propaganda, a facilidade com que ele permite que grandes quantidades de dados sejam analisadas, e gerenciadas, torna a tecnologia acessível para empresas cujo o foco principal não é a tecnologia, porém, gostariam de processar e utilizar seus dados. Seus aspectos mais importantes destacam em primeiro lugar a performance, devido a sua arquitetura desenvolvida em multiestágios em memória primitiva, fornece uma performance até 100 vezes mais rápida em certas aplicações (Gutierrez, 2015), em seguida a facilidade de programação, que permite uma a adesão em empresas de diversos setores.

**Figura 2: Projeção de Big Data por padrão de aplicativo e porcentagem de adoção do Spark/Streaming de 2014-2026 em Bilhão de dólares**



**Fonte: (Wheatley, 2016)**

No gráfico apresentado na figura 2, pode-se observar a projeção de crescimento da adoção do Spark, e a equivalência em bilhões de dólares da receita de *Big Data* gerada por ele e seus concorrentes em padrões de aplicação, nota-se que seu crescimento é exponencial, chegando a 59% em 2026 e uma receita de aproximadamente 50 bilhões de dólares.

Portanto, seu estudo se torna relevante para entender seu funcionamento, e as possibilidades que a ferramenta fornece, sendo também um conhecimento importante no âmbito profissional, tendo em vista sua força e presença nas empresas de diversos setores.

1. **Objetivos**

**3.1 Objetivo Geral**

* Estudar e analisar os recursos e arquitetura da ferramenta Apache Spark para computação em *cluster,* aplicado ao processamento de *Machine* *Learning*, com utilização de streaming de dados.

**3.2 Objetivos específicos**

* Estudar os conceitos teóricos de computação em *cluster*;
* Estudar e analisar a ferramenta Apache Spark de maneira geral;
* Estudar a arquitetura de processamento do Apache Spark;
* Estudar o recurso de streaming de dados do Spark Streaming;
* Implantar um cluster do Apache Spark;
* Desenvolver um modelo de *machine learning,* para processamento de linguagem natural, utilizando o Spark Streaming para obtenção dos dados;
* Processar o modelo de *machine learning* através do cluster Spark;
* Identificar qualidades e problemas enfrentados durante o desenvolvimento.

1. **Fundamentação Teórica**

**4.1 Computação em *cluster***

Um *cluster* de computadores é definido quando existem duas ou mais máquinas trabalhando em conjunto para o processamento de requisições, a palavra vem do inglês e significa agrupamento, cada computador ou processador presente no *cluster* é chamado “nodo”.

Apesar de seu surgimento na década de 1960 com a IBM, a tecnologia só começou a ganhar espaço e importância a partir dos anos 80, de acordo com Pitanga (2008), o crescimento do uso da tecnologia foi motivado por três fatores: a construção de processadores de alto desempenho, o surgimento de redes de comunicação de baixa latência e a padronização de ferramentas para computação paralela e distribuída.

Os *clusters* podem ser divididos em duas categorias básicas: Alta Disponibilidade e Alto Desempenho de Computação. Sendo que o primeiro é projetado para prover disponibilidade a uma aplicação pelo maior tempo e da forma mais segura possível, já o segundo, tem a finalidade de propiciar um alto poder computacional, com sua programação conciliando a capacidade de processamento de cada computador, criando um super computador.

O cluster de Alto Desempenho, é o tipo necessário para o processamento do algoritmo de *Machine* *Learning* estudado nesse trabalho, com ele é possível executar aplicações e processamentos que requerem alto desempenho, utilizando equipamentos comuns, e distribuindo o processamento entre os nodos.

**4.2 Aprendizado de máquina (*Machine Learning*)**

Aprendizado de máquina ou *machine learning*, é a ciência da programação de computadores para que eles possam aprender com os dados. De acordo com Géron (2019), nos anos 1959, Arthur Samuel definiu o *machine learning* como “O campo de estudo que dá aos computadores a habilidade de aprender sem ser explicitamente programado”, posteriormente em 1997, Tom Mitchell definiu que é a técnica em que “Um programa de computador aprende pela experiência E em relação a algum tipo de tarefa T e alguma medida de desempenho P se o seu desempenho em T, conforme medido por P, melhora com a experiência E”.

Portanto, ela consiste em elaborar algoritmos capazes de “ensinar” uma determinada informação para uma máquina, dispositivo ou sistema, e definir padrões que possam ser seguidos para se alcançar esse objetivo, porém, diferentemente dos algoritmos tradicionais, em que esses padrões são fixos e definidos no momento da codificação pelos desenvolvedores, nos de *machine learning* são definidos parâmetros e inseridos dados para que a máquina treine e analise os mesmos, encontre os padrões, e defina a importância de cada um deles para prever ou identificar a informação desejada.

**4.3 Apache Spark**

De acordo com a documentação do Apache Spark (2021), ele um mecanismo de análise unificado para processamento de dados em grande escala.

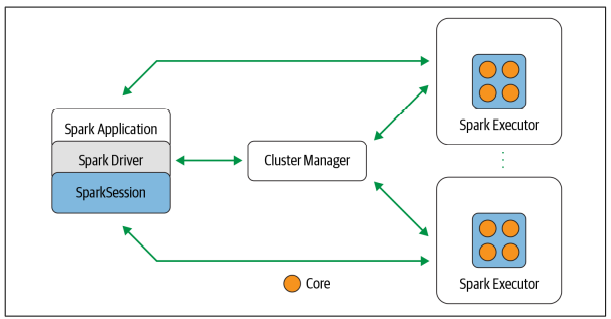
Fornece APIs de alto nível em Java, Scala, Python e R e um mecanismo otimizado que oferece suporte a gráficos de execução geral. Ele também oferece suporte a um rico conjunto de ferramentas de nível superior, incluindo Spark SQL para SQL e processamento de dados estruturados, MLlib para aprendizado de máquina, GraphX para processamento de gráfico e *Streams* estruturados para computação incremental e processamento de fluxo.

Segundo Damji (2020), o Spark provê armazenamento em memória para intermediar o processamento, fazendo com que ele seja muito mais rápido que o seu antecessor Hadoop MapReduce. O Spark teve sua filosofia de design centrado em quatro características chave: velocidade, facilidade de uso, modularidade e extensibilidade.

**4.3.1 Arquitetura do Spark**

O funcionamento do Spark é baseado no processamento distribuído dos dados, através dos componentes funcionando cooperativamente em um cluster, como é definido por Damji (2020), uma aplicação Spark consiste em um programa Driver que é responsável por orquestrar as operações paralelas no cluster Spark, o Driver acessa os componentes do cluster, sendo eles o Cluster Manager e o Spark Executors, através de uma SparkSession, e distribui as operações que estão sendo requisitadas pela aplicação.

**Figura 3: Componentes do Apache Spark**



**Fonte: (Learning Spark, 2020)**

De acordo com a figura 3, podemos visualizar a arquitetura de um cluster Spark, em que a aplicação Spark inicia um Spark Driver através de uma SparkSession, o Driver se comunica com o Cluster Manager que aciona os Spark Executors, estes por sua vez, realizam o processamento solicitado, e retornam as saídas para aplicação.

O Spark possuí suporte para uma variedade de modos de implantação, segundo Damji (2020), ele pode ser executado em diferentes configurações e ambientes, contanto que o cluster manager possa acessar os recursos do sistema e os spark executors, os modos de implantação disponíveis atualmente são: Local, Standalone, YARN cliente, YARN cluster e Kubernetes.

**4.3.2 Spark Driver**

De acordo com Damji (2020), o Driver é responsável por instanciar o SparkSession, ele exerce múltiplas funções, como a comunicação com o cluster manager, requerimento de recursos (CPU, memória, etc.) do cluster manager para os executors e transforma as operações de processamento em processamento DAG, a agenda, e distribui suas execuções em forma de tarefas para os executors, a partir do momento que os recursos são alocados, ele se comunica diretamente com os executors.

**4.3.3 SparkSession**

Como é definido por Damji (2020), o SparkSession é um canal unificado de para todas as operações e processamento de dados do Spark, esse recurso surgiu no Spark 2.0, que tornou o trabalho com o Spark mais simples e fácil.

**4.3.4 Cluster Manager**

O cluster manager, conforme Damji (2020), é responsável por gerenciar e alocar os recursos para o cluster de cada aplicação Spark., atualmente o Spark oferece suporte a quatro diferentes clusters managers: o interno standalone, Apache Hadoop YARN, Apache Mesos, e Kubernetes.

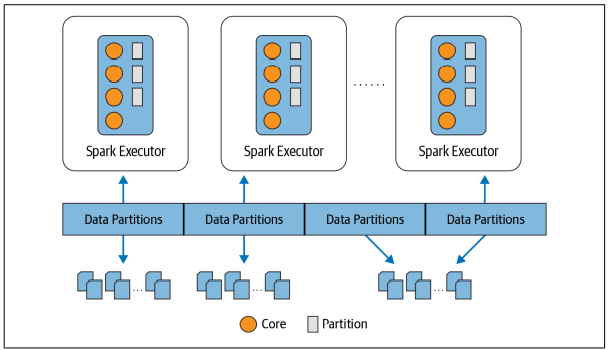
**4.3.5 Spark Executors**

Segundo Damji (2020), o Spark executor é executado em cada nó de trabalho de um cluster, ele se comunica com o driver, sendo responsável por executar as tarefas que são atribuídas aos nós de trabalho, na maioria das implantações, um único executor é utilizado em cada nó.

**4.3.6 Partições e dados distribuídos**

Os dados físicos reais são distribuídos pelo armazenamento como partições em esquemas HDFS ou armazenamento em nuvem, de acordo com Damji (2020), enquanto os dados são distribuídos como partições ao longo do cluster físico, o Spark trata cada partição como uma abstração lógica de alto nível, como por exemplo um DataFrame na memória, sempre que possível, cada executor do Spark é designado para realizar uma tarefa que exige que leia a partição mais próxima a ele na rede de trabalho, observando a localidade dos dados.

**Figura 4: Atribuição dos executors de acordo com a proximidade da partição**



**Fonte: (Learning Spark, 2020)**

Na figura 4 é exemplificado como os executors interagem com as partições de dados, de acordo com Damji (2020), o particionamento é o que permite um paralelismo eficiente, distribuindo os dados em partições, os executors somente processam os dados que estão próximos deles, minimizando a largura de banda.

**4.4 Spark Streaming**

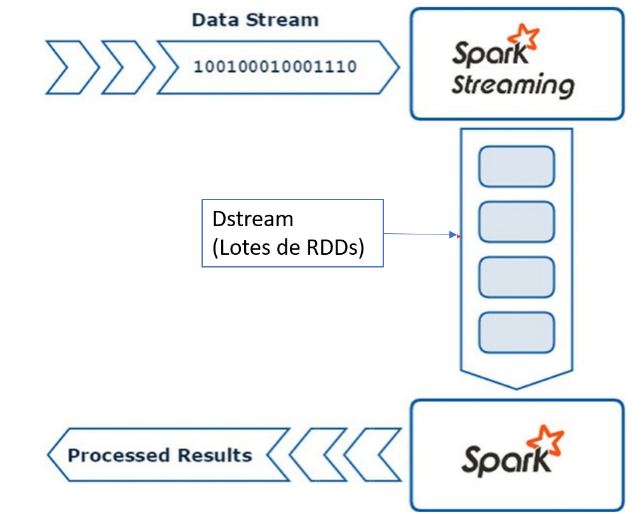
Com o advento dos sistemas de Big Data, a quantidade de dados emitidas por sensores, sites, e dispositivos IoT aumentaram exponencialmente, com isso, a necessidade de se utilizar e analisar estes dados em tempo real se tornou um recurso chave, segundo Aven (2017), por volta de 2008, com o surgimento das redes sociais, como Facebook e Twitter, a empresa BackType, especializada em social media e inteligência de mercado, iniciou um novo projeto para processamento distribuído de streaming de dados, chamado Storm, que era integrado ao ecossistema do Hadoop, rapidamente ele se tornou líder no seguimento open source de processamento de dados em tempo real.

Com isso, os fundadores do Spark, enxergaram a necessidade de desenvolver uma solução para processamento de stream dentro do projeto Spark, porém, eles queriam um sistema tolerante a falhas e com garantia de integridade dos dados, fatores que eram limitantes no Storm, a abordagem do Spark foi de entregar um sistema tolerante a falhas e que garantia que cada evento seria processado exatamente uma vez, mesmo se um nó falhasse, além disso, o sistema deveria ser integrado ao seu framework baseado em batch RDD, e assim surgiu o Spark Streaming. O design do Spark Streaming foi desenvolvido visando quatro objetivos: Baixa latência (escala de segundos), processamento de um evento somente uma vez, escalabilidade linear e integração com o Spark core API.

**4.4.1 Arquitetura do Spark Streaming**

De acordo com Aven (2017), a arquitetura do Spark Streaming introduziu o conceito de discretized streams (DStreams), que são lotes de dados armazenados em múltiplos RDDs, cada lote representa uma janela de tempo, normalmente em segundos, em que os dados foram coletados, os RDDs resultantes podem ser processados utilizando as APIs do Apache Spark.

**Figura 5: Visão geral em alto nível da arquitetura do Spark Streaming**



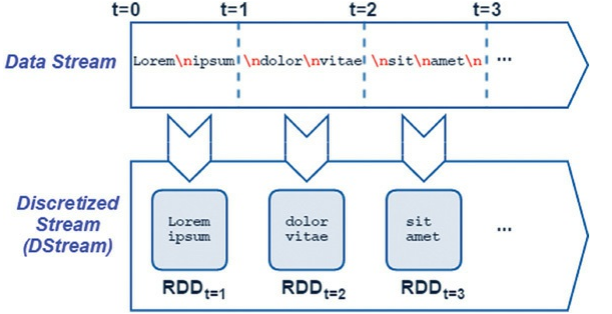
**Fonte: (Sams Teach Yourself Apache Spark™ in 24 Hours, 2017)**

Na figura 5 é possível ter uma visão geral do funcionamento do Spark Streaming, em que o módulo de Streaming recebe a stream de dados, em seguida armazena isso em lotes de RDDs divididos de acordo com a janela de tempo que foi definida, em seguida passa pelo processamento do Spark, e retorna os resultados processados.

**4.4.2 Discretized Streams (DStreams)**

Segundo Aven (2017), DStreams são os objetos básicos de programação do Spark Streaming API, eles representam uma sequencia continua de RDDs que são criados através do stream de dados, sendo cada RDD representante de uma janela de tempo. DStreams podem ser criadas a partir de diferentes fontes de stream de dados, como sockets TCP, sistemas de mensageria, e APIs de streaming (como a do Twitter). DStreams suportam dois tipos de operações: operações de transformação e operações de saída.

**Figura 6: Estrutura da DStream**



**Fonte: (Sams Teach Yourself Apache Spark™ in 24 Hours, 2017)**

A partir da figura 6, é possível verificar a estrutura da DStream, os dados são recebidos através da fonte de stream de dados, de acordo com o intervalo escolhido (representado pela letra t), em seguida a DStream tranforma esses dados em um lote RDD que armazena os dados equivalente a janela de tempo, para cada janela de tempo, haverá um RDD armazenando seus dados recebidos.

**4.5 Natural Language Toolkit (NLTK)**

O Natural Language Toolkit (NLTK), de acordo com a sua documentação (2021), pode ser definido como uma plataforma para construção de programas Python para trabalhar com dados de linguagem humana. Ele fornece mais de 50 corpora e recursos lexicais, em conjunto com um pacote de bibliotecas de processamento de texto para classificação, tokenização, lematização, marcação, análise e raciocínio semântico, e processamento de linguagem natural.

1. **Trabalhos Similares**

Em sua dissertação de mestrado, “Uma análise comparativa de ambientes para Big Data: Apache Spark e HPAT”, Rafael Aquino de Carvalho, realizou uma comparação de desempenho do Apache Spark e seu concorrente High Performance Analytic Toolkit (HPAT), realizando os testes a partir de dois algoritmos, um para soma dos elementos de um vetor unidimensional e o outro sendo o K-means, sendo este utilizado para identificar diferentes grupos de dados, muito utilizado na área de machine learning. Seu objetivo foi descobrir se o HPAT é uma alternativa, em desempenho, ao Apache Spark, levando em consideração dois ambientes distintos de trabalho. Segundo Carvalho (2018), o HPAT obteve um desempenho melhor do que o Apache Spark em um ambiente sem falhas, porém, o Spark teve um desempenho melhor em um ambiente com falhas.

Este comportamento encontrado pelo Carvalho, reforça ainda mais os conceitos do Apache Spark apresentados neste trabalho, pois, no desenvolvimento do Spark, sempre foi uma grande preocupação a tolerância a falhas, o que pode tornar o seu processamento um pouco mais lento em certos cenários, porém, sempre será muito confiável.

O *paper* “Big data analytics on Apache Spark”, dos autores Salman Salloum, Ruslan Dautov, Xiaojun Chen, Patrick Xiaogang Peng e Joshua Zhexue Huang, realiza uma análise completa do Apache Spark, focando nos componentes chaves e seus recursos, detalhando o design de todas as APIs do core Apache Spark, além de destacar algumas pesquisas direcionadas ao uso do Spark para análise de Big Data.

1. **Metodologia**

**6.1 Tipo de pesquisa**

A pesquisa será um estudo teórico sobre a arquitetura e funcionalidades da ferramenta Apache Spark, com foco no recurso de streaming de dados, integrado ao machine learning.

**6.2 População e amostra de dados**

Os dados utilizados contêm informações sobre mensagens enviadas pelo aplicativo Twitter.

**6.3 Coleta de dados**

Os dados foram extraídos do Twitter através de sua API.

**6.4 Tratamento e análise de dados**

Após os dados serem extraídos, eles passarão pelo algoritmo de análise de sentimento, que irá determinar se aquele twitch tinha um teor positivo ou negativo sobre o termo pesquisado.

**6.5 Métodos utilizados**

Inicialmente, será analisado a arquitetura do cluster Spark, identificando seus componentes, e como eles foram desenvolvidos para otimizar o processamento distribuído dos dados, após a compreensão de seu funcionamento, será implantado e configurado um cluster local para processamento do modelo de machine learning que será criado.

Com o cluster implantado, será possível desenvolver a pipeline de tratamento de dados utilizando o Spark Streaming para obtenção e transformação dos dados, esses dados serão obtidos através de requisições a API do Twitter.

Tendo como fonte esses dados obtidos e tratados, será desenvolvido um modelo de machine learning utilizando algoritmo de processamento de linguagem natural, para análise de sentimento das mensagens enviadas pelo Twitter.

Após esse desenvolvimento, o modelo será submetido ao cluster Spark para realizar o processamento de dados em tempo real, de forma distribuída.

**6.6 Instrumento de medida (métricas)**

A métrica utilizada será a verificação da capacidade do Spark coletar, processar e retornar os dados em tempo quase real, verificando seu tempo de atraso com relação a cada lote de stream de dados.

**6.7 Ferramentas e tecnologias utilizadas**

Será utilizado a linguagem de programação Python para codificação, um notebook com sistema operacional Linux para executar o cluster Spark, framework Apache Spark para processamento em cluster dos dados, API Apache Streaming para obtenção dos dados em tempo real, e realizar os tratamentos necessários, API do Twitter como fonte de dados, biblioteca NLTK para criação dos modelos de machine learning, e a IDE Jupyter Notebook como ambiente de desenvolvimento.

**6.8 Recursos materiais**

Para realização da pesquisa, será necessário um notebook ou máquina virtual com sistema operacional Linux, e acesso à internet.

1. **Desenvolvimento**

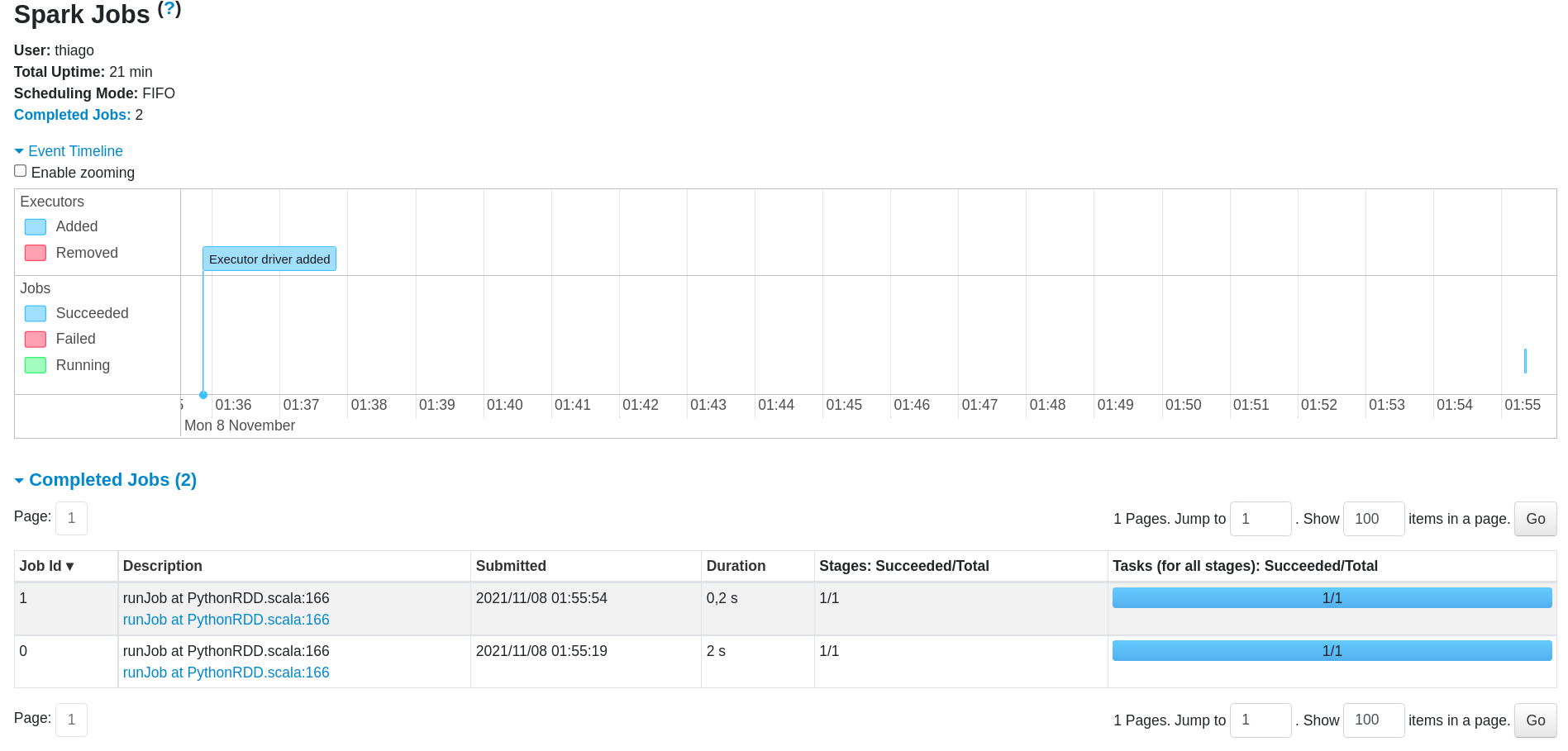
Para realizar o desenvolvimento da aplicação Spark foi utilizado o arquivo no formato notebook, através da IDE Jupyter Notebook, o primeiro passo foi realizar a importação dos módulos e pacotes necessários para aplicação, em seguida, foi criado um objeto StreamingContext, que recebe como argumento uma SparkSession e um intervalo de tempo para a coleta dos lotes de dados (batch), o batch foi definido como 5 segundos.

Com o StreamingContext inicializado já é possível utilizar o processamento do Spark, dessa forma, foi realizada a leitura de um conjunto de dados que continha twitters em inglês e suas classificações entre positivas e negativas, este conjunto de dados foi utilizado para treinar o algoritmo de machine learning.

Após este procedimento, é necessário instanciar o objeto SentimentAnalyzer que cria um modelo utilizando este algoritmo para análise dos twitters, em seguida, é necessário baixar o pacote de stopwords em inglês da biblioteca NLTK, e criar uma lista com estas palavras.

Em seguida, foram utilizados 10 mil twitters como base de treino para o modelo. A partir do momento em que a SparkSession foi iniciada, é possível acessar uma interface gráfica para o Spark através da porta localhost:4040, esta interface possui um menu para exibição dos Jobs, Stages, Storage, Environment, Executors, SQL e Streaming.

**Figura 7: Interface gráfica de exibição dos Jobs no Spark**



**Fonte: (Do Autor, 2021)**

A figura 7 apresenta as informações dos Jobs que foram estão sendo executados, ou já foram concluídos pelos executores, esta aba exibe uma linha do tempo com os eventos registrados pelo cluster, até este momento da aplicação, havia sido registrado o evento de adição de um executor, e a execução de dois jobs, que já estavam completos. Além disso, também é possível ver a hora de envio do job, sua duração, os stages, e as tarefas envolvidas em cada stage.

O próximo passo foi a autenticação na API do Twitter, através de credenciais obtidas ao utilizar uma conta de desenvolvedor no site do Twitter, após realizar a autenticação, é possível definir um termo de busca que será requisitado via URL para API, em seguida, é necessário iniciar um objeto queueStream para começar o stream de dados, para este teste, foi definido uma quantidade de 10 tweets coletados por cada batch.

Com os tweets coletados, é possível aplicar uma função que passa cada tweet pelo modelo criado para analisar o sentimento, e então armazena o resultado em uma lista.

No teste realizado, se utilizou o termo “Trump” para busca, os resultados de saída

da aplicação é exibida da seguinte maneira:

**Figura 8: Resultados de saída da aplicação**

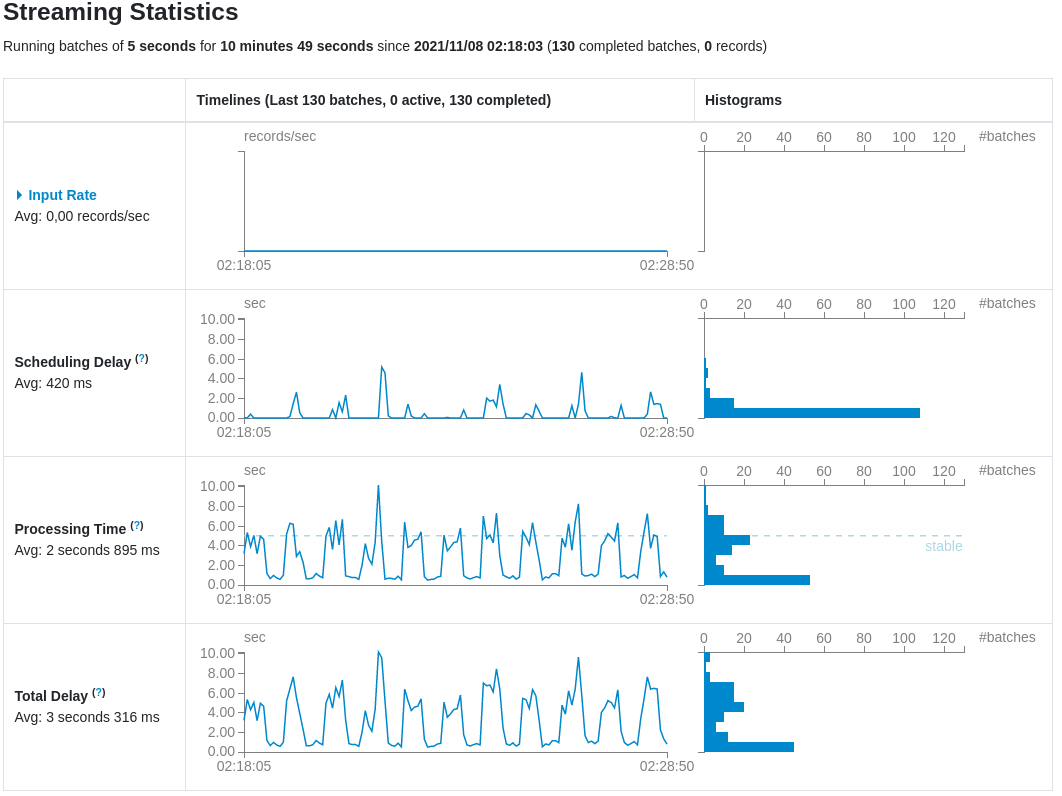


**Fonte: (Do Autor, 2021)**

Na figura 8 pode-se observar a saída da aplicação, que é composta por uma lista com os tweets analisados, e um número na frente de cada um, este número será 0 quando o modelo identificar um sentimento negativo no tweet, ou 1 quando identifica sentimento positivo.

No menu de Streaming, é possível verificar informações estatísticas sobre o desempenho da execução da aplicação.

**Figura 9: Interface gráfica da estatística de desempenho do streaming no Spark**



**Fonte: (Do Autor, 2021)**

Na figura 9 a primeira estatistica apresentada é a taxa de entrada (Input Rate), em seguida é o atraso de agendamento (Scheduling Delay), está métrica representa o tempo que o streaming leva para enviar o job de um batch, após é exibido o tempo de processamento (Processing Time), que é o tempo necessário para realizar todos os processamentos em cada batch, e por último, o atraso total (Total Delay), este é o tempo total para completar o processamento de cada batch.

1. **Resultados e Discussões**
2. **Conclusões**

**Agradecimentos**

**Referências**

APACHE SPARK. **Spark Overview**. [S. l.], 2021. Disponível em: https://spark.apache.org/docs/3.1.2/. Acesso em: 18 jun. 2021.

AVEN, Jeffrey. **Sams Teach Yourself Apache Spark™ in 24 Hours**. [S. l.]: Sams, 2017.

CARVALHO, Rafael Aquino de. **Uma análise comparativa de ambientes para Big Data: Apache Spark e HPAT**. Orientador: Prof. Dr. Alfredo Goldman. 2018. Dissertação (Mestrado em Ciências) - Instituto de Matemática e Estatística da Universidade de São Paulo, [S. l.], 2018. Disponível em: https://teses.usp.br/teses/disponiveis/45/45134/tde-15062018-110116/publico/dissertacao\_rafael\_aquino.pdf. Acesso em: 7 nov. 2021.

DAMJI, Jules S.; WENING, Brooke; DAS, Tathagata; LEE, Denny. **Learning Spark.** 2. ed. [S. l.]: O’Reilly Media, 2020.

DOCUMENTATION: Natural Language Toolkit. [S. l.], 2021. Disponível em: https://www.nltk.org/. Acesso em: 7 nov. 2021.

GARCIA, Marco. **Spark: Saiba mais sobre esse poderoso framework**. [S. l.], 8 ago. 2020. Disponível em: https://www.cetax.com.br/blog/conheca-mais-sobre-o-framework-apache-spark/. Acesso em: 7 jun. 2021

GÉRON, Aurélien. **Mãos a obra: Aprendizado de máquina com Scikit-Learn & TensorFlow**: Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes. 1. ed. [S. l.]: Alta Books, 2019.

GOMES, Pedro César Tebaldi. **O QUE É E COMO FUNCIONA UM CLUSTER?.** [S. l.], 3 mar. 2015. Disponível em: https://www.opservices.com.br/o-que-e-um-cluster/. Acesso em: 8 jun. 2021.

GUTIERREZ, Daniel. **Why is Apache Spark So Hot?.** [S. l.], 17 nov. 2015. Disponível em: https://insidebigdata.com/2015/11/17/why-is-apache-spark-so-hot/. Acesso em: 8 jun. 2021.

JULIO, RENNAN. “Dados são o novo petróleo”, diz CEO da Mastercard – exceto por um pequeno detalhe: Para Ajay Banga, internet das coisas é a mais impactante tecnologia da transformação digital. **Época Negócios**, [s. l.], 5 jul. 2019. Disponível em: https://epocanegocios.globo.com/Empresa/noticia/2019/07/dados-sao-o-novo-petroleo-diz-ceo-da-mastercard.html. Acesso em: 8 nov. 2020

NAVITA. **Big Data: O que é? Conheça seu conceito e definição.** [S. l.], 2 ago. 2019. Disponível em: https://navita.com.br/blog/big-data-saiba-mais-sobre-o-conceito-e-definicao/#:~:text=De%20acordo%20com%20o%20Instituto,dados%20gerados%20todos%20os%20dias. Acesso em: 8 jun. 2021.

PITANGA, Marcos. **Construindo supercomputadores com linux**. 3. ed. Rio de janeiro: Brasport, 2008.

SALLOUM, Salman; DAUTOV, Ruslan; CHEN, Xiaojun; PENG, Patrick Xiaogang; HUANG, Joshua Zhexue. **Big data analytics on Apache Spark**. International Journal of Data Science and Analytics , [s. l.], 2016. Disponível em: https://link.springer.com/content/pdf/10.1007/s41060-016-0027-9.pdf. Acesso em: 7 nov. 2021.

SCHOCH, Andréa.**O que é Machine Learning?** [S. l.], 2 jan. 2018. Disponível em: https://www.appai.org.br/desenrola-machine-learning-aprendizado-de-maquina/. Acesso em: 7 jun. 2020

WHEATLEY, Mike. **Apache Spark will dominate the Big Data landscape by 2022, Wikibon says**. [S. l.], 30 mar. 2016. Disponível em: https://siliconangle.com/2016/03/30/apache-spark-will-dominate-the-big-data-landscape-by-2022-wikibon-says/. Acesso em: 9 jun. 2021.