Aplicação de Ferramentas de *Big Data*: Impacto do COVID-19 em eventos públicos.

Luís Pinto[1*]

[1*]Departamento de Informática, Universidade do Minho, Braga, Portugal.

Corresponding author(s). E-mail(s): pg47428@alunos.uminho.pt;

Abstract

Hoje em dia tudo o que nos rodeia cria dados. Lidar com estes volumes massivos de informação pode ser complicado, daí surgir a necessidade de definir uma arquitetura adequada, capaz de trabalhar com dados em bruto até chegar à exploração de um caso de uso. O presente artigo foca-se no estudo do impacto do COVID-19 em eventos públicos, procurando encontrar métricas que expliquem e avaliem a relação entre ambos os temas através do uso de ferramentas de *big data*. Com este documento pretende-se sensibilizar o leitor acerca das diferentes ferramentas de *big data* existentes e de como podemos aplicá-las em casos práticos. Assim, iremos começar com uma breve introdução apresentando em maior detalhe o *use case*. De seguida, será apresentada qual a metodologia adotada, bem como dado a conhecer o estado da arte em *big data*. Posteriormente, na secção de resultados, existirá um resumo e explicação dos métodos e ferramentas incluídas na arquitetura proposta. Finalmente, chegamos a conclusão do artigo onde serão apresentadas as considerações finais do mesmo.

Keywords: *Big Data*, *Batch Pipeline*, COVID-19, *Public Events*, *Business Intelligence*, *Dashboards*, *ETL*, *Data Analytics*, *Databases*, *Big Data Tools*

1 Introdução

Nestes últimos anos, a COVID-19 influenciou vários aspetos da sociedade, nomeadamente os eventos públicos. As pessoas não se sentiam à vontade a sair para eventos sem máscara ou a estarem em espaços com grandes ajuntamentos, correndo o risco de ficarem infetadas (principalmente quando a vacinação não era uma opção). Por outro lado, muitas vezes viam-se ainda obrigadas a ficarem confinadas em casa ou a verem os seus eventos públicos serem cancelados devido às novas medidas em vigor.

Torna-se evidente que existe uma relação entre a realização de eventos públicos pré, durante e pós-pandemia. Daqui, deriva a principal motivação para a criação do artigo. Pretende-se encontrar **métricas concretas** e explicativas para, através delas, se conseguir **avaliar o impacto que o aparecimento da COVID-19 teve nos eventos públicos**.

2 Metodologias

A metodologia baseou-se numa abordagem ***bottom-up***, ou seja, numa primeira fase ocorreu a pesquisa de *datasets* e de ferramentas, surgindo posteriormente a etapa de definição do(s) caso(s) de uso.

O processo de pesquisa engloba duas categorias, quer a investigação de *datasets*, quer a identificação de ferramentas de *big data* apropriadas para a exploração dos *datasets* e do *use case*. Deste modo, a escolha das **ferramentas** baseou-se em artigos, pesquisas de ferramentas de *big data* (tendo em conta as melhores e mais utilizadas atualmente), bem como ponderando as escolhas das empresas. Relativamente aos ***datasets***, a escolha teve em consideração fatores como o volume de dados ser significativo e existirem alguns aspetos em comum com o *dataset* fornecido pelos docentes, permitindo deste modo agilizar o processo de junção dos *datasets*.

3 Estado da Arte

Nesta secção, culmina a pesquisa das principais tecnologias associadas à área de *Big Data*. Cada uma das ferramentas possui, inerentemente, as suas características (pontos fortes e fracos, vantagens e desvantagens,...) tornando-se necessário efetuar uma análise das tecnologias que possivelmente serviram de suporte para o *pipeline* ou a arquitetura a ser implementada.

De modo a facilitar a apresentação das ferramentas, estas serão introduzidas consoante a sua principal funcionalidade, isto é, podendo pertencer a uma categoria de união de *datasets*, armazenamento, processamento ou visualização de dados. Além disso, será dado um maior ênfase aquelas que, à partida, serão escolhidas e utilizadas na arquitetura.

3.1 União de *Datasets*

O principal objetivo destas ferramentas será o de permitir juntar os diversos *datasets* encontrados, de forma a serem posteriormente armazenados numa base de dados ou num sistema de ficheiros. Assim sendo, as opções apresentadas para resolver esta questão são o uso de **Apache Spark** ou **Python**, mais concretamente, através de métodos e/ou bibliotecas dedicados a cumprir com este objetivo.

Relativamente ao **Apache Spark**, a estratégia passará pelo uso de uma *API* em python - o **PySpark**, que suporta recursos úteis do Spark, tais como o Spark SQL e o DataFrame. [1] Quanto aos **aspetos positivos**, revela-se bastante escalável e eficiente com o processamento de grandes volumes de dados. Quanto aos **aspetos negativos**, é feito para volumes enormes de dados tornando-se mais lento em *datasets* pequenos. [2]

Por sua vez, em **python**, recorrer-se-ia ao uso da biblioteca **Pandas**, mais especificamente a métodos de álgebra relacional como o *join*. [3] No **espetro positivo**, destaca-se pela sua simplicidade, flexibilidade, facilidade de utilização, pela rapidez de execução das operações (uma vez que os *datasets* cabem em memória) e encontra-se ainda bem integrada num ecossistema completo de bibliotecas numéricas, estatísticas e de *machine learning* (SciKit Learn, Tensorflow, etc...). No **espetro negativo**, não consegue fazer uso de múltiplos *CPUs* e exige que o *dataset* caiba na *RAM* da máquina local. [4, 5]

3.2 Armazenamento

O componente de armazenamento pode ser usado para armazenar os dados a serem enviados no *pipeline* ou para guardar o *output* dos dados do **pipeline**. Assim sendo, as tecnologias candidatas são : o sistema de ficheiros **HDFS** e a base de dados **Apache Casandra**. [6]

A base de dados distribuída **Apache Casandra destaca-se positivamente** por ser a única BD NoSQL que garante uma disponibilidade *always-on*, correr sem *SPOFs* e por apresentar uma incrível performance na velocidade de leitura e escrita (devido a sua arquitetura *P2P*). Contudo, apresentam-se

limitações para ficheiros de grandes dimensões sendo mais direcionada para muitos dados pequenos ou, por exemplo, dados em tempo real. [7, 8]

Relativamente ao sistema de ficheiros distribuídos da Hadoop, o **HDFS** apresenta como **vantagem** a sua alta compatibilidade com *batch processing*, apresentando uma arquitetura hierarquizada *master/slave*. Assim sendo, embora garanta robustez e tolerância a falhas, surge **limitado** pelo fator de replicação de ficheiros, definido por omissão em 3, apresentando um *SPOF* no *master node* (denomeado por *Namenode*), dependendo os *data nodes* do mesmo. [8, 9]

3.3 Processamento

As ferramentas de processamento, tal como o nome indica, servem para tratar os dados, permitindo trabalhar os mesmos de forma a aproximar-mo-nos do objetivo, o *use case* definido. De seguida, enunciam-se o **Apache Spark** e o **MapReduce** como candidatas a adotar na arquitetura. [6]

O **Apache Spark** é **caracterizado positivamente** por ser mais rápido e poderoso no processo de processamento. Em contrapartida, um **aspeto negativo** deriva do facto de necessitar de memória para correr, exigindo que os dados caibam em memória.

Já o **MapReduce**, torna a execução do ambiente menos custosa dado que as escritas e leituras de dados ocorrem em disco. Adicionalmente, possui ainda a **vantagem** de suportar dados maiores. O seu **ponto fraco** é o tempo de processamento, sendo ideal para situações onde a velocidade de processamento não seja um fator crítico. [10, 11]

3.4 Visualização

Quanto às ferramentas de apresentação e visualização, destacam-se : o **Power BI** da Microsoft e o **Tableau** da empresa americana Tableau Software.

A respeito do **Power BI**, são enunciadas como **vantagens** : a facilidade de uso para utilizadores iniciantes, a interface intuitiva, ser barato, rápido e com boa performance para volumes limitados de dados. E, como **desvantagens** : a utilização de fórmulas rígidas em DAX, interface sobrecarregada e a configuração dos aspetos visuais.

Enquanto que o **Tableau** apresenta como **pontos fortes** : suportar dados em tempo real, possuir bom apoio ao utilizador e ser mais rápido em grandes volumes de dados. E, como **pontos fracos** : ter uma maior curva de aprendizagem e não possuir controlo de versões. [12, 13]

Posto isto, tendo em conta todos os fatores, as ferramentas tecnológicas selecionadas foram : o **python** (pandas), o **Hadoop HDFS**, o **Apache Spark** (PySpark) e o **Power BI**. De seguida, no próximo tópico, será detalhado o *pipeline* resultante, bem como esclarecido todo o processo pelo qual os dados irão passar.

4 Resultados

Munidos de um melhor entendimento acerca das tecnologias utilizadas em *Big Data*, entramos agora na secção dos resultados. Esta é responsável por apresentar a arquitetura a desenvolver, explicar quais os *datasets* escolhidos, quais os seus atributos e ainda como podemos chegar a partir destes a um *dataset* unificado e capaz de explorar o caso de uso.

4.1 Datasets

De seguida, prossegue-se à apresentação dos *datasets* encontrados. O **primeiro *dataset*** apresenta informações sobre **casos e mortes diárias devido ao COVID-19**. [14] Este foi fornecido pela equipa docente, sendo constituído por 8 atributos :

Atributo	**Descrição**
Date_reported	Data da observação.
Country_code	Código do país no formato ISO Alfa-2.
Country	País, território, área.
WHO_region	Divisão do mundo em seis regiões segundo a OMS (Organização Mundial da Saúde).
New_cases	Novos casos confirmados.
Cumulativate_cases	Casos acumulativos confirmados.
New_deaths	Novas mortes confirmadas.
Cumulative_deaths	Mortes acumulativas confirmadas.

Table 1: Atributos *Dataset* 1

O **segundo *dataset***, tal como os próximos a serem apresentados, resultou de uma pesquisa *online*, sendo proveniente de uma conta no *github* associada ao domínio *ourworldindata.org*. O seu contexto prende-se à área da **vacinação ao COVID-19**, [15] apresentando os seguintes 16 atributos :

Atributo	**Descrição**
country	País.
iso_code	Código do país no formato ISO Alfa-3.
date	Data da observação.
total_vaccinations	Total de doses administradas.
people_vaccinated	Total de novas pessoas vacinadas.
people_fully_vaccinated	Total de pessoas totalmente vacinadas.
total_boosters	Total de doses de reforço.
daily_vaccinations_raw	Mudança diária de doses administradas.
daily_vaccinations	Doses diárias administradas.
total_vaccinations_per_hundred	Vacinação total por 100 pessoas.
people_vaccinated_per_hundred	Pessoas vacinadas por 100 pessoas.
people_fully_vaccinated _per_hundred	Pessoas totalmente vacinadas por 100 pessoas.
total_boosters_per_hundred	Total de doses de reforço por 100 pessoas.

daily_vaccinations_per_million	Vacinações diárias por milhão.
daily_people_vaccinated	Número diário de pessoas que recebem a 1ª dose.
daily_people_vaccinated _per_hundred	Número diário de pessoas que recebem a 1ª dose por 100 pessoas.

Table 2: Atributos *Datasets* 2

Torna-se agora clara a preocupação com a **unificação dos *datasets***, podemos observar que o *dataset* anterior também apresenta campos relacionados com : **o país, o código do país e a data do registo**. Proporcionando, deste modo, uma métrica para a fácil integração dos dados dos diferentes *datasets*. Tal como veremos adiante, os seguintes exemplos também tem em consideração este fator.

Relativamente aos **últimos 4 *datasets***, todos provém do *site ourworldindata.org* e apresentam uma estrutura bastante semelhante entre si, isto é, incluem os atributos : *entity* (país), *code* (código do país no formato alfa-3) e *day* (data da observação). Assim, estes *datasets* apenas diferem no seu último atributo, estando este intrinsecamente relacionado com o âmbito do próprio *dataset* :

- Dataset 3 - **medidas de confinamento**; [16]
- Dataset 4 - **restrições em ajuntamentos**; [17]
- Dataset 5 - **cancelamento de eventos públicos**; [18]
- Dataset 6 - **políticas de obrigatoriedade do uso de máscara**. [19]

Sendo os seus atributos, respetivamente :

Atributo	**Descrição**
stay_home_requirements	0 - Sem restrições; 1 - Recomendado não sair de casa; 2 - Obrigatório não sair mas com excepções; 3 - Obrigatório não sair mas com excepções mínimas; Vazio - sem dados.
restrictions_gatherings	0 - Sem restrições; 1 - Restrição superior a 1,000 pessoas; 2 - Apenas entre 100-1,000 pessoas; 3 - Apenas entre 10-100 pessoas; 4 - Menos de 10 pessoas; Vazio - sem dados.
cancel_public_events	0 - Nenhuma medida ativa; 1 - É recomendado cancelar; 2 - É obrigatório cancelar; Vazio - sem dados.
facial_coverings	0 - Sem medidas ; 1 - Recomendado o uso de máscara; 2 - Obrigatório em alguns espaços partilhados sem distanciamento ; 3 - Obrigatório em todos espaços públicos sem distanciamento; 4 - Obrigatório sempre que fora de casa;

Table 3: Atributos dos *Datasets* 3, 4, 5 e 6.

Por fim, mencionar apenas que todas as hiperligações consultadas encontrar-se-ão citadas e anotadas nas referências.

4.2 Arquitetura

A solução proposta para a arquitetura será descrita de acordo com 4 etapas : União de *Datasets* **(1)**; Armazenamento **(2)**; Processamento **(3)** e Visualização **(4)**.

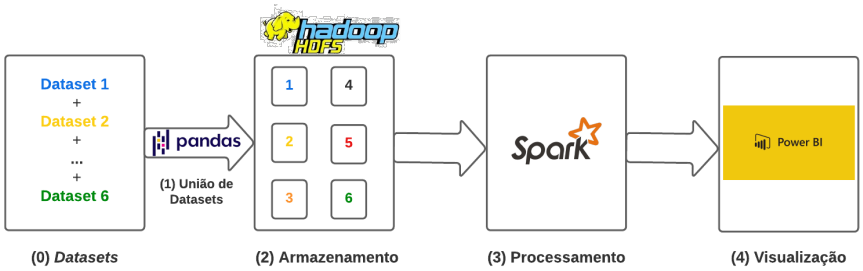

Fig. 1: *Big Data Pipeline*

(1) Uma vez que a estratégia adotada consiste na implementação de um *batch pipeline*, começamos, antes de mais, por proceder à união dos 6 *datasets* apresentados anteriormente. A ferramenta escolhida para esta tarefa foi o python, mais concretamente através da biblioteca **pandas**. A escolha desta biblioteca baseou-se na sua facilidade de utilização e, sobretudo, pela sua rapidez comparativamente à alternativa PySpark.

Posto isto, o processo de **união** dos *datasets* será executado tendo em conta os seus atributos comuns, ou seja, o **país e da data da observação**. Isto levanta algumas questões. Em primeiro lugar, utiliza-se o código do país invés do nome do país por este se apresentar em formato universal. Visto que a maioria dos *datasets* usa o formato alpha-3, será ainda necessário converter os códigos dos países do *dataset 1* de alpha-2 para alpha-3. Em segundo lugar, será necessário ter em consideração que o intervalo temporal dos registos difere de *dataset* para *dataset*. Uma vez que o *dataset 1* apresenta o menor intervalo temporal, este será usado para definir o período temporal da investigação (de 03/01/2020 até 23/02/2022). A terceira questão relaciona-se com questões de formatação, sendo necessário garantir que os *datasets* apresentam o mesmo nome para colunas usadas na unificação dos *datasets*.

Deste modo, serão usados os métodos da biblioteca pandas : ***merge*** - para realizar a união dos atributos de cada dataset; ***read_json*** e ***read_csv*** - para interpretar o conteúdo dos *datasets* conforme se encontrem em formato JSON (*dataset 2*) ou CSV (restantes *datasets*), respetivamente. Dizer ainda que optou-se por não se fazer a injeção de dados sintéticos nesta fase de modo a

manter a integridade dos dados originais. Numa secção posterior será abordado novamente este assunto.

(2) De seguida, surge a etapa de armazenamento. O componente de armazenamento pode ser usado, quer para armazenar os dados a serem enviados no *pipeline*, quer para guardar o *output* dos dados do mesmo. A abordagem utilizada foi a primeira opção devido a ser necessário uniformizar e juntar o conteúdo de diferentes fontes. A opção utilizada foi o sistema de ficheiros distribuídos da Hadoop, o **HDFS**. Esta opção foi a clara vencedora, uma vez que apresenta alta compatibilidade com a estratégia de *batch processing pipeline*, enquanto que a ferramenta Apache Cassandra não se adequa tanto ao contexto do trabalho por não serem usados dados em tempo real.

De modo a incluir o *dataset* unificado, obtido na fase anterior, no HDFS é necessário utilizar alguns comandos simples, semelhantes aos comandos utilizados em linux. Visto que o sistema é *stateless* é necessário indicar sempre os nomes das diretorias as quais queremos aceder. De seguida, segue um exemplo ilustrativo do fluxo de comandos a executar de modo a armazenar o *dataset* unificado no HDFS. Em primeiro lugar, deve-se executar o comando "hadoop fs -mkdir /mydata" de modo a criar uma diretoria (denominada por mydata a titulo de exmeplo) para armazenar o *dataset*. De seguida, bastará usar o comando "hadoop fs -put dataset.csv /mydata/" para adicionar o ficheiro (dataset.csv) à diretoria criada anteriormente.

(3) A terceira etapa da arquitetura denomina-se por processamento, em maior detalhe esta etapa pode ser caracterizada como uma etapa ETL (*extract, transform and load*). O primeiro passo consiste em extrair os dados presentes na componente de armazenamento. Por conseguinte, estes dados prosseguem para uma fase de transformação, onde serão tratados e refinados. Mais concretamente, será ainda aqui que ocorrerá o processo de injeção de dados sintéticos, substituindo os antigos valores nulos ou incoerentes por valores mais adequados, bem como a remoção de colunas e/ou linhas redundantes. Por fim, o último passo consiste apenas em carregar os dados agora processados para a plataforma de visualização escolhida.

A ferramenta selecionada para esta tarefa foi o Apache Spark, mais concretamente o **PySpark**. A escolha desta ferramenta baseou-se no facto de ser mais rápida e poderosa comparativamente ao MapReduce, uma vez que os dados cabem em memória. Recorrendo ao método ***withColumn*** do PySpark é possivel realizar a etapa de transformação dos dados anteriormente mencionada.

(4) Finalmente, na última etapa, após termos os dados processados somos capazes de incorporá-los em ferramentas de visualização e de apresentação. Conseguindo, assim, exprimir as métricas encontradas para avaliar o impacto do aparecimento da covid-19 nos eventos públicos. A titulo de exemplo,

podemos analisar o modo como o aumento do número de casos impactou o cancelamento de eventos públicos.

O **Power BI** vai ser a ferramenta implementada, a razão desta escolha relaciona-se com o facto de o Tableau ser bom para dados em tempo real e como não é esse o objetivo do trabalho, não faria sentido adotar esta solução. Além disso, o Power BI apresenta uma interface intuitiva e é rápido para o volume de dados do projeto. Com o auxílio desta ferramenta, torna-se então possível analisar os dados através de *dashboards*, gráficos, tabelas, entre outros.

5 Conclusão

A elaboração do presente artigo permitiu explorar diferentes ferramentas de *big data* com o intuito de criar uma solução viável para um *data pipeline*. A metodologia seguiu uma abordagem *bottom-up*, onde através dos diversos *datasets* apresentados inferiu-se o caso de uso deste projeto - "Calcular métricas que investiguem o impacto que o aparecimento do COVID-19 teve nos eventos públicos". Com base na pesquisa realizada, é possível afirmar que a proposta de arquitetura apresentada cumpre com todos os requisitos do problema. Conseguindo assim, pegar em dados em bruto de diversos formatos, transformá-los, tratá-los e por fim carregá-los para uma plataforma de visualização de modo a apresentar os resultados da pesquisa efetuada.

References

[1] Spark, A.: pyspark.sql.DataFrame.join. Link - https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.join.html (2022)

[2] Science, T.D.: Spark vs Pandas, part 2 — Spark. Link - https://towardsdatascience.com/spark-vs-pandas-part-2-spark-c57f8ea3a781 (2020)

[3] Pandas: User Guide - Merge, join, concatenate and compare. Link - https://pandas.pydata.org/docs/user_guide/merging.html (2022)

[4] Science, T.D.: Spark vs Pandas, part 1 — Pandas. Link - https://towardsdatascience.com/spark-vs-pandas-part-1-pandas-10d768b979f5 (2020)

[5] Science, T.D.: Spark vs Pandas, part 4— Recommendations. Link - https://towardsdatascience.com/spark-vs-pandas-part-4-recommendations-35fc554573d5 (2020)

[6] DataOpsZone: WHAT IS A DATA PIPELINE IN HADOOP? WHERE AND HOW TO START. Link - https://www.dataopszone.com/what-is-a-data-pipeline-in-hadoop/ (2020)

[7] DataSax: What is Apache Cassandra? Link - https://www.datastax.com/cassandra (2022)

[8] Bekker, A.: Apache Cassandra vs. Hadoop Distributed File System: When Each is Better. Link - https://www.scnsoft.com/blog/cassandra-vs-hadoop (2018)

[9] Pedamkar, P.: Hadoop vs Cassandra. Link - https://www.educba.com/hadoop-vs-cassandra/ (2022)

[10] Bekker, A.: Spark vs. Hadoop MapReduce: Which big data framework to choose. Link - https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce (2017)

[11] Education, I.C.: Hadoop vs. Spark: What's the Difference? Link - https://www.ibm.com/cloud/blog/hadoop-vs-spark (2021)

[12] Biswal, A.: Power BI Vs Tableau: Difference and Comparison. Link - https://www.simplilearn.com/tutorials/power-bi-tutorial/power-bi-vs-tableau (2022)

[13] Pedamkar, P.: Power BI vs Tableau. Link - https://www.educba.com/power-bi-vs-tableau/ (2022)

[14] Organization, W.H.: WHO COVID-19 global data. Link - https://covid19.who.int/WHO-COVID-19-global-data.csv (2022)

[15] in Data, O.W.: Country-by-country data on global COVID-19 vaccinations. Link - https://github.com/owid/covid-19-data/tree/master/public/data/vaccinations (2022)

[16] in Data, O.W.: Stay-at-home requirements during the COVID-19 pandemic. Link - https://ourworldindata.org/grapher/stay-at-home-covid (2022)

[17] in Data, O.W.: Restrictions on public gatherings in the COVID-19 pandemic. Link - https://ourworldindata.org/grapher/public-gathering-rules-covid (2022)

[18] in Data, O.W.: Cancellation of public events during COVID-19 pandemic. Link - https://ourworldindata.org/grapher/public-events-covid (2022)

[19] in Data, O.W.: Face covering policies during the COVID-19 pandemic. Link - https://ourworldindata.org/grapher/face-covering-policies-covid (2022)