Armazenamento/Processamento de Dados

Danilo Balman Garcia[1]
Ra: 2482088
Rafael Machado Wanner[1]
Ra: 2021013
[1]Departamento Acadêmico de Computação (DACOM)
Universidade Tecnológica Federal do Paraná (UTFPR)

Abstract

This paper presents a data storage/processing exercise that simulates the analysis of large volumes of NYC taxi data to optimize data infrastructure and guide decisions about processing tools. The process involves automated data download, schema analysis, data cleansing, and data consolidation. The second step is the Performance Benchmark, based on the hypothesis that "DuckDB with Parquet is faster than Pandas with CSV."The answer to this question involves a series of tests, based on a selection of queries implemented in SQL (DuckDB) and Pandas, verifying queries that generated failures and measuring execution time. Only successfully executed queries were recorded, and graphs were generated for visualization.

Resumo

Este trabalho apresenta um exercício de Armazenamento/Processamento de dados que simula a análise de grandes volumes de dados de táxis de NYC para otimizar infraestrutura de dados e orientar decisões sobre ferramentas de processamento. O processo envolve o download automatizado de dados, a análise de schema, a limpeza de dados e a consolidação dos mesmos. A segunda etapa é o Benchmark de Performance, a partir da hipótese de ´´DuckDB com Parquet é mais rápido que Pandas com CSV", cuja resposta envolve uma série de testes, a partir da escolha de algumas consultas que foram implementaddas em SQL (DuckDB) e em Pandas, com a verificação de consdultas que geraram falhas e a medição do tempo de execução. Foram registradas apenas consultas executadas com sucesso e gerados gráficos para visualização.

1 Introdução

A crescente dependência de dados em larga escala exige infraestruturas de armazenamento e processamento robustas e eficientes. Este relatório documenta um exercício prático de Data Engineering focado na simulação da análise de grandes volumes de dados de corridas de táxi da cidade de Nova Iorque (NYC). O objetivo principal foi otimizar a infraestrutura de dados e fornecer insights técnicos para a seleção de ferramentas de processamento. A primeira fase do projeto cobriu o ciclo essencial de engenharia de dados, incluindo o download automatizado dos dados brutos, a análise de schema, a limpeza e a consolidação dos conjuntos. Na sequência, realizamos um Benchmark de Performance rigoroso para testar a hipótese de que "DuckDB utilizando o formato Parquet é superior em velocidade a Pandas com CSV"para cargas de trabalho analíticas. Implementamos consultas de negócio-chave tanto em SQL (DuckDB) quanto em Pandas, registrando apenas os tempos de execução das consultas bem-sucedidas e gerando gráficos de visualização para comparar o desempenho e embasar futuras decisões arquiteturais.

Corresponding author(s): Danilo Garcia, danilogarcia@alunos.utfpr.edu.br, Rafael Wanner, rafaelwanner@alunos.utfpr.edu.br, Universidade Tecnológica Federal do Paraná (UTFPR)

2 Setup e Consolidação dos Dados

Durante o processo de tratamento dos dados, foram observadas mudanças no schema, tais como: nome de colunas (Airport_fee -> airport_fee), Tipagem (Ex: tpep_dropoff_datetime, que era datetime64[ns] e mudou para datetime64[us]).

Para lidar com dados ausentes ou inconsistentes, as principais estratégias utilizadas foram identificar a natureza do dado, para assim detectar e varrer inconsistências, como idade ou valores negativos, viagens sem passageiros, ou distância negativa ou zerada, além de averiguar se alguma coluna apresentou alta porcentagem de valores nulos, pois ela não seria adequada para a pesquisa.

A reprodutibilidade do processo foi garantida por meio de boas práticas de versionamento e controle de ambiente. O uso do Git permitiu rastrear alterações no código e nos dados, enquanto ambientes virtuais asseguraram a instalação das bibliotecas com versões específicas utilizadas no projeto. Adicionalmente, o uso de Docker possibilitou encapsular todo o ambiente de execução, garantindo consistência entre diferentes máquinas. Por fim, uma documentação detalhada foi mantida para registrar todas as etapas do processo e facilitar sua replicação.

2.1 Recursos Utilizados

- Python 3.12.4
- Editor de texto: Visual Studio code 1.85.2
- Sistema operacional: Windows 10
- Edição de Documentos: Overleaf.com

3 Medidas de Armazenamento

Os arquivos CSV foram concatenados em um único dataset e exportados como CSV, Parquet Snappy, Parquet ZSTD. à medida em que os dados foram tratados, o número de entradasfoi reduzindo aos seguites valores:

- Antes de remover duração <= 0: 38310226
- Depois de remover duração <= 0: 38294657
- Antes de remover passageiros <= 0: 38294657
- Depois de remover passageiros <= 0: 36405739
- Antes de eliminar valores negativos em campos monetarios: 36405739
- Depois de eliminar valores negativos em campos monetarios: 36031004
- Antes de Validar datas dentro do período esperado: 36031004
- Depois de Validar datas dentro do período esperado: 36030907

A tabela abaixo mostra o tamanho dos arquivos após a exportação.

Tabela 1: Tamanhos dos arquivos

Tipo	Tamanho
CSV	3584.70 MB
Parquet (Snappy)	727.19 MB
Parquet (ZSTD)	593.88 MB

A tabela a seguir mostra as taxas de compressão relativas.

Tabela 2: Taxas de compressão relativas

Tipo	Proporção	Compressão
Parquet (Snappy)	20.29%	79.71%
Parquet (ZSTD)	16.57%	83.43%

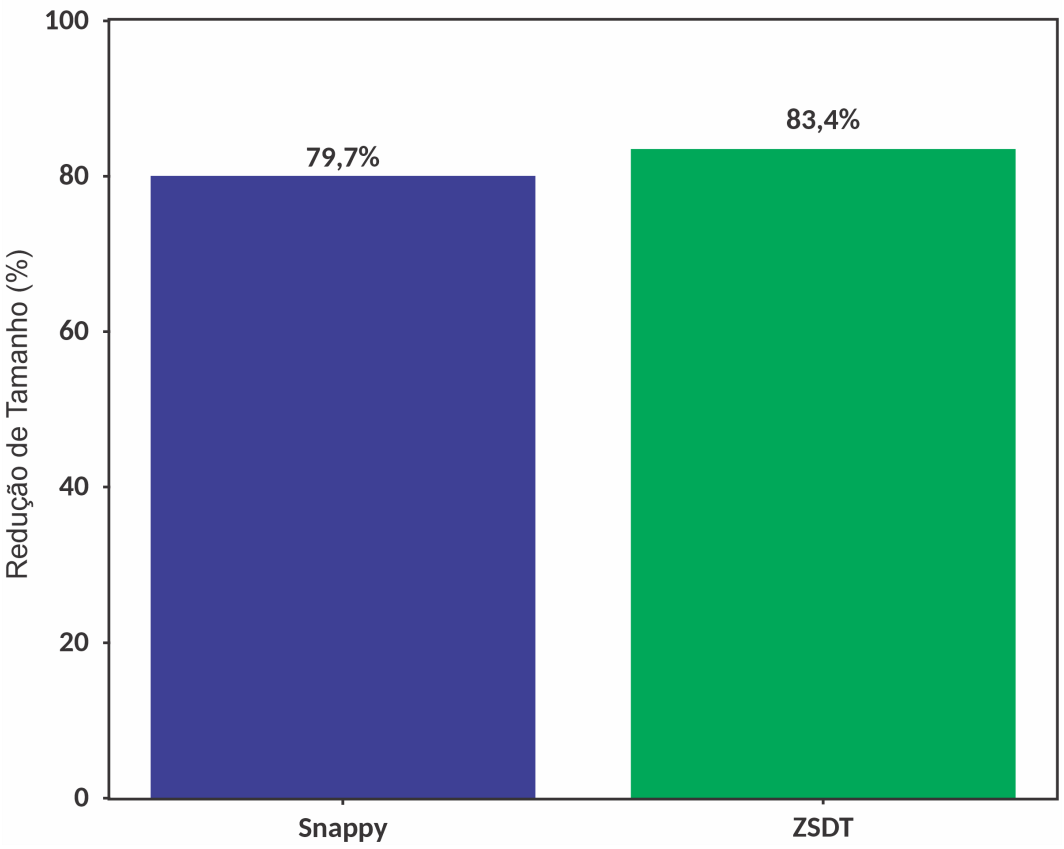
A tabela abaixo mostra o tempo de leitura dos arquivos (em segundos).

Tabela 3: Tempo de leitura (segundos)

Tipo	Tamanho
CSV (amostra 5.000.000)	7.636
Parquet (Snappy)	2.616
Parquet (ZSTD)	2.436

A figura abaixo demonstra graficamente a diferença percentual na taxa de compressão do arquivo CSV em Parquet (Snappy) e Parquet (ZSDT.)

Figura 1: Diferença Percentual de Compressão em Relação ao CSV



Elaboração Própria

4 Benchmark de Performance

O exercício a seguir parte da hipótese de que "DuckDB com Parquet é mais rápido que Pandas com CSV". Para testá-la, foram escolhidos os seguintes parâmetros, dentre os sugeridos na disciplina:

- **Receita por zona:** TOP 10 zonas de pickup por receita total média
- **Padrões temporais:** Agregação por mês/semana (COUNT viagens, SUM receita, AVG distância)
- **Horário de pico:** Ranking de horários mais movimentados usando window functions (RANK/-ROW_NUMBER)
- **Análise de gorjeta:** Taxa de gorjeta média por borough usando CASE WHEN para categorizar
- **Viagens longas vs curtas:** Comparativo usando CASE para categorizar distâncias

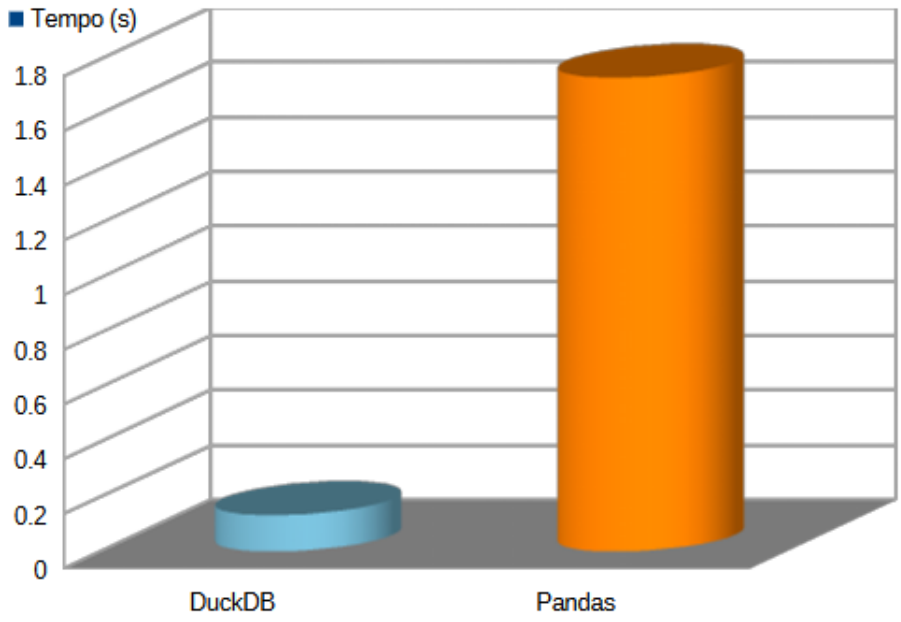
Para cada um dos parâmetros de consulta escolhida, foram realizados os seguintes procedimentos:

- Implementação em SQL (DuckDB) e validação dos resultados inspecionando os dados
- Implementação equivalente em Pandas e verificação para constatar se os resultados são consistentes
- Anotação de quais consultas falharam no Pandas (memória, tempo, erro)
- Medição do tempo de execução rodando cada consulta 5 vezes em cada ferramenta
- Registro apenas das consultas que executaram com sucesso para comparação de performance

4.1 Receita por zona

Na consulta das TOP 10 zonas de pickup por receita total média medida, o DuckDB mostrou-se significativamente mais rápido, conforme gráfico abaixo.

Figura 2: Receita por zona: DuckDB vs Pandas

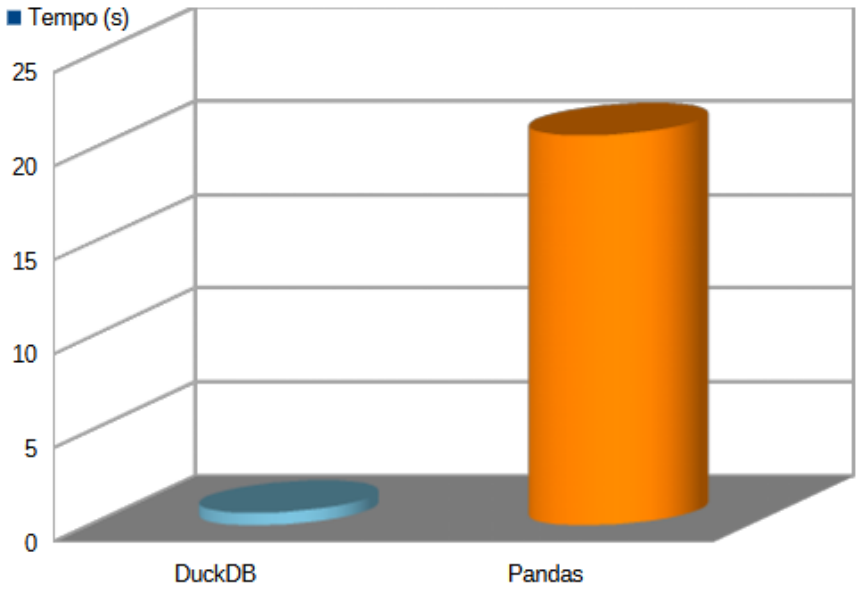


Elaboração Própria

4.2 Padrões temporais

Na consulta de Agregação por mês/semana (COUNT viagens, SUM receita, AVG distância), o DuckDB mostrou-se significativamente mais rápido, conforme gráfico abaixo.

Figura 3: Padrões temporais: DuckDB vs Pandas

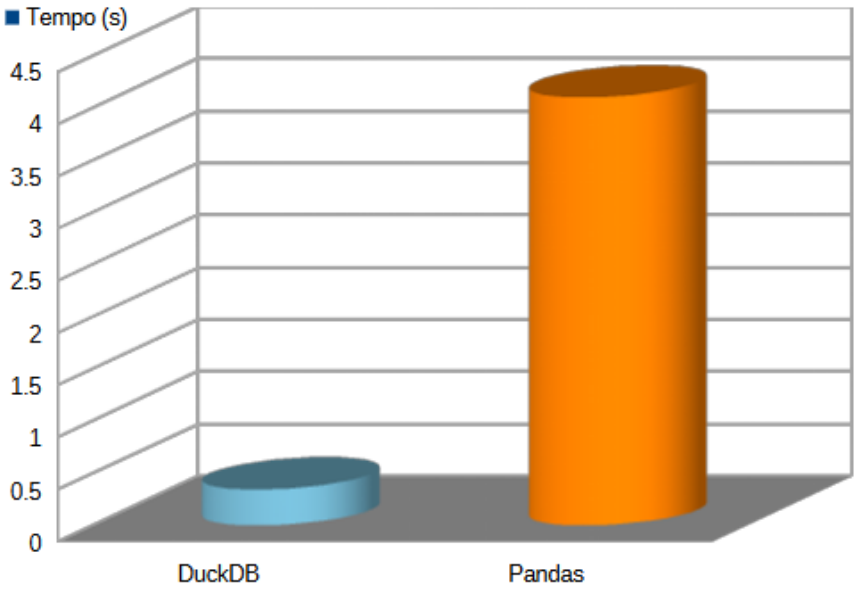


Elaboração Própria

4.3 Horário de pico

Na consulta de Ranking de horários mais movimentados usando window functions (RANK/-ROW_NUMBER), o DuckDB mostrou-se significativamente mais rápido, conforme gráfico abaixo.

Figura 4: Horário de pico: DuckDB vs Pandas

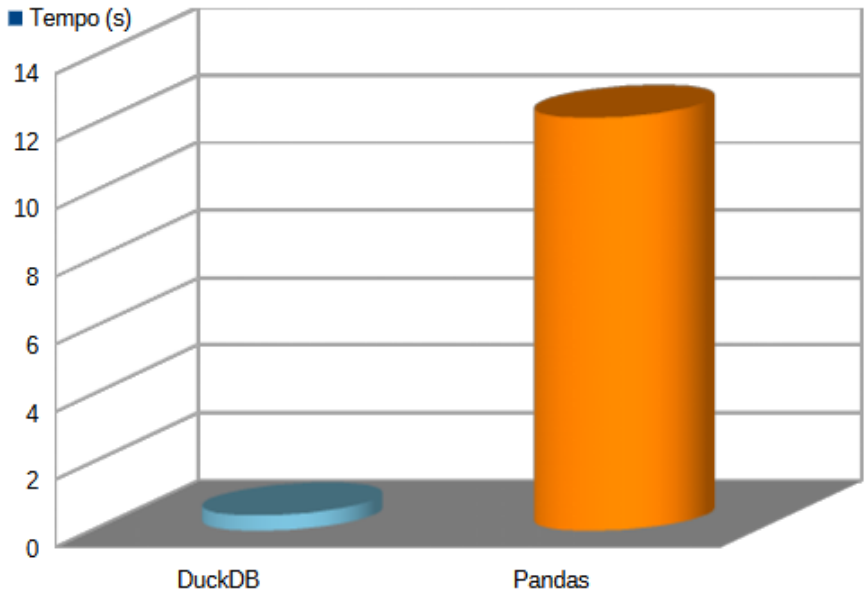


Elaboração Própria

4.4 Análise de gorjeta

Na consulta de Taxa de gorjeta média por borough usando CASE WHEN para categorizar, o DuckDB mostrou-se significativamente mais rápido, conforme gráfico abaixo.

Figura 5: Análise de gorjeta: DuckDB vs Pandas



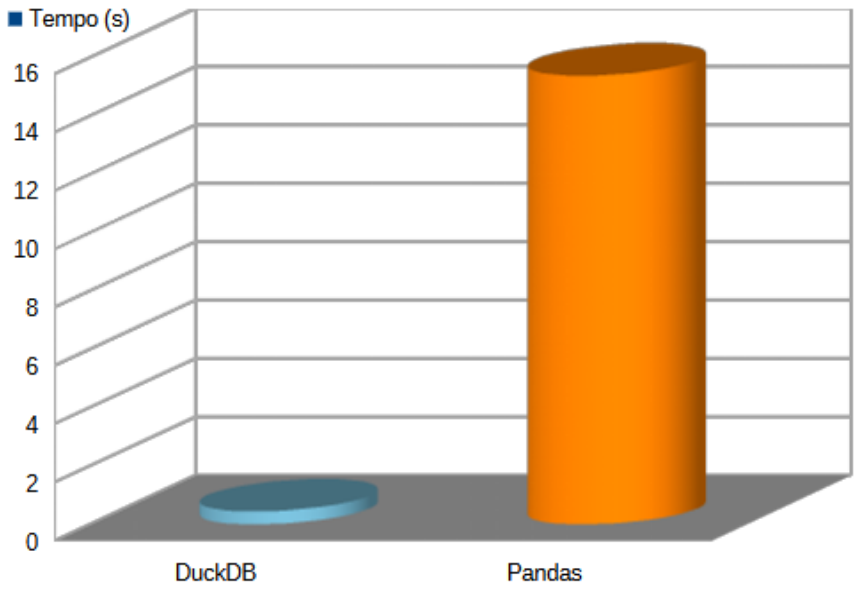
Elaboração Própria

4.5 Viagens longas vs curtas

Comparativo usando CASE para categorizar distâncias Na consulta comparativa usando CASE

para categorizar distâncias, o DuckDB mostrou-se significativamente mais rápido, conforme gráfico abaixo.

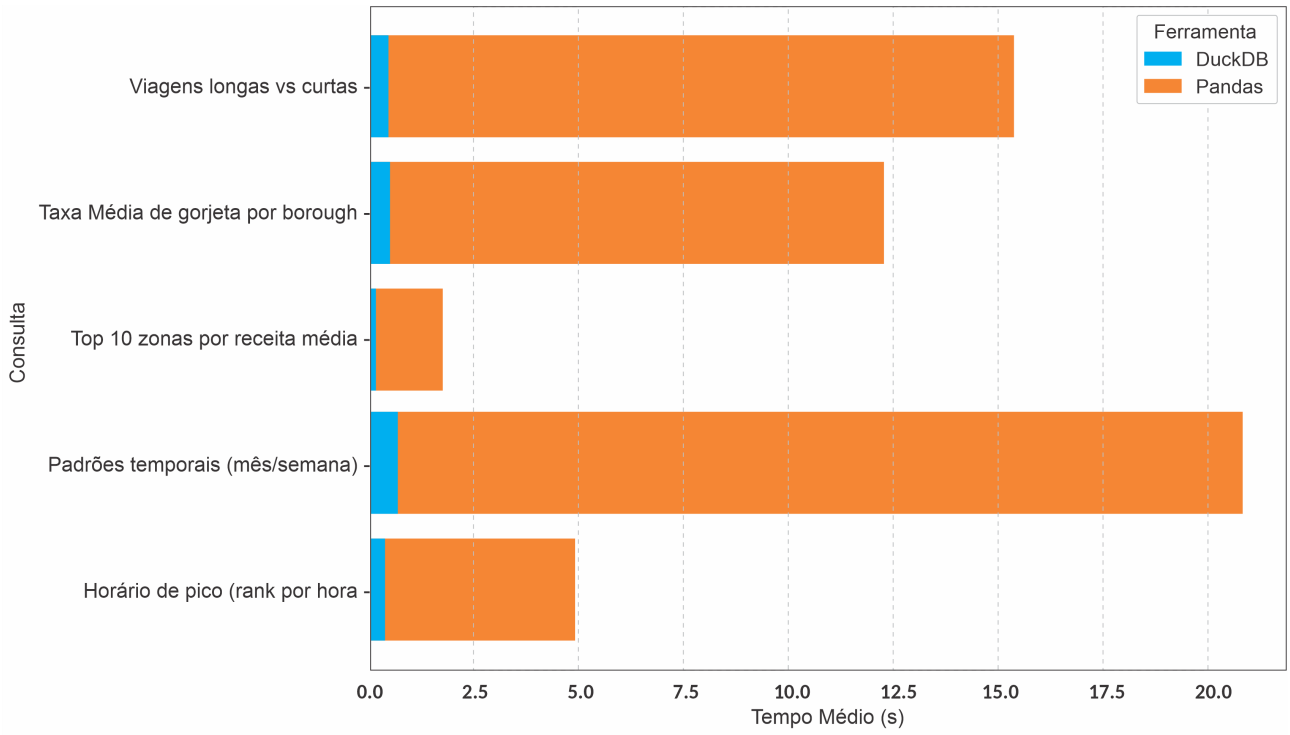
Figura 6: Viagens longas vs curtas: DuckDB vs Pandas



Elaboração Própria

O gráfico a seguir sintetisa as cinco consultas, consolidando a informação de que o DuckDB mostrou-se significativamente mais rápido que o Pandas.

Figura 7: Comparação de Performance: DuckDB vs Pandas



Elaboração Própria

Ficou evidenciado que a melhor combinação para a execução de pesquisas/testes foi entre a ferramenta DuckDB com o formato Parquet (ZSDT), dado que este comprime o arquivo CSV para um tamanho bem reduzido (83,4%) e o DuckDB responde, em média, em menos de 4% do tempo que o Pandas utiliza, confirmando a hipótese original.

O gráfico de barras mostrou-se o mais condizente para evidenciar as performances dos aplicativos. As cores laranja e azul foram utilizadas, inicialmente, por serem definiodas no Visual Studio e, posteriormente, julgadas adequadas para a comparação, pelo nível de contraste, além de que os resultados melhores, que caraterizam um aplicativo mais robusto, érepresentado pelo laranja, uma cor mais quente do que o azul celeste contraposto. Para o gráfico que apresenta a diferença percentual de compressão, as duas cores vivas simbolizam dois programas com ótima performance, daí o uso e aprovação de duas tonalidades fortes.