SECP3133-02

HIGH PERFORMANCE DATA PROCESSING

Optimizing High-Performance Data Processing for Large-Scale Web Crawlers

Prepared By:

VINESH A/L VIJAYA KUMAR A22EC0290

JOSEPH LAU YEO KAI A22EC0055

TIEW CHUAN SHEN A22EC0113

NUR FARAH ADIBAH BINTI IDRIS A22EC0245

Lecturer Name:

DR. ARYATI BINTI BAKRI

Date of Submission:

16/5/2025

Table of Contents

1. Introduction

1.1 Background of the project

The demand for data analytics in real time and low latency is growing nowadays to meet the needs of the market, which high performance data processing is focused on across various industries. High performance data processing emphasizes the use of enhanced computational methods such as high performance computing(HPC) to perform data processing processes such as data collection, cleaning and analysis in a short time. The project is focused on answering:

1. Does HPC infrastructure and methods enhance the performance of the data processing phase?
2. How is the performance of different Python libraries and frameworks (Pandas, Dask) in implementing HPC for data processing tasks?
3. Which combination set of HPC techniques and tools will provide the best and most efficient solution for data cleaning and analysis?

Through this research, the project is aimed to provide comparative analysis on the impact of different libraries to web scraping, data cleaning and analysis.

1.2 Objectives

1. To develop a web crawler that is able to extract at least 100,000 records from a News Straits Times (NST) website.
2. To store extracted data in CSV format for further processing.
3. To clean and preprocess the raw dataset.
4. To evaluate performance before and after optimization using several performance metrics.

1.3 Target website and data to be extracted

In this project, the targeted website is New Strait Times(NST), with the link www.nst.com.my. New Strait Times or NST is one of Malaysia's most known news publishers in English. Various domains are offered by New Strait Times such as national news, business, politics, sports and lifestyle. The platform is providing a huge dataset of articles, enabling the website to be a good source for data analytics. NST is selected for its huge data volume and consistent news structure and format which allows for the smooth extraction process for the project. The articles provide metadata and content sections which are suitable for web crawling.

The main focus will be about extracting the informations from individual news articles from different section, which key data attributes targeted are shown as below:

No	Data Field	Data Type	Description
1	Section	String	News topic(crime, politics, nation, health).
2	Publication date	Date	The date(including time) the article is published with format mm:dd:yyyy @ hh:mm
3	Headline	String	Title of the article.
4	Summary	String	Brief summary of the news.

Table 1: Data attributes planned to be scrapped from website

Through these attributes, a valuable dataset will be collected for analyzing and researching insights. The crawling process will be designed with respects to ethical scraping practices and rules, to avoid adding great workloads to the web server through appropriate delays between requests.

2. System Design & Architecture

2.1 Architecture

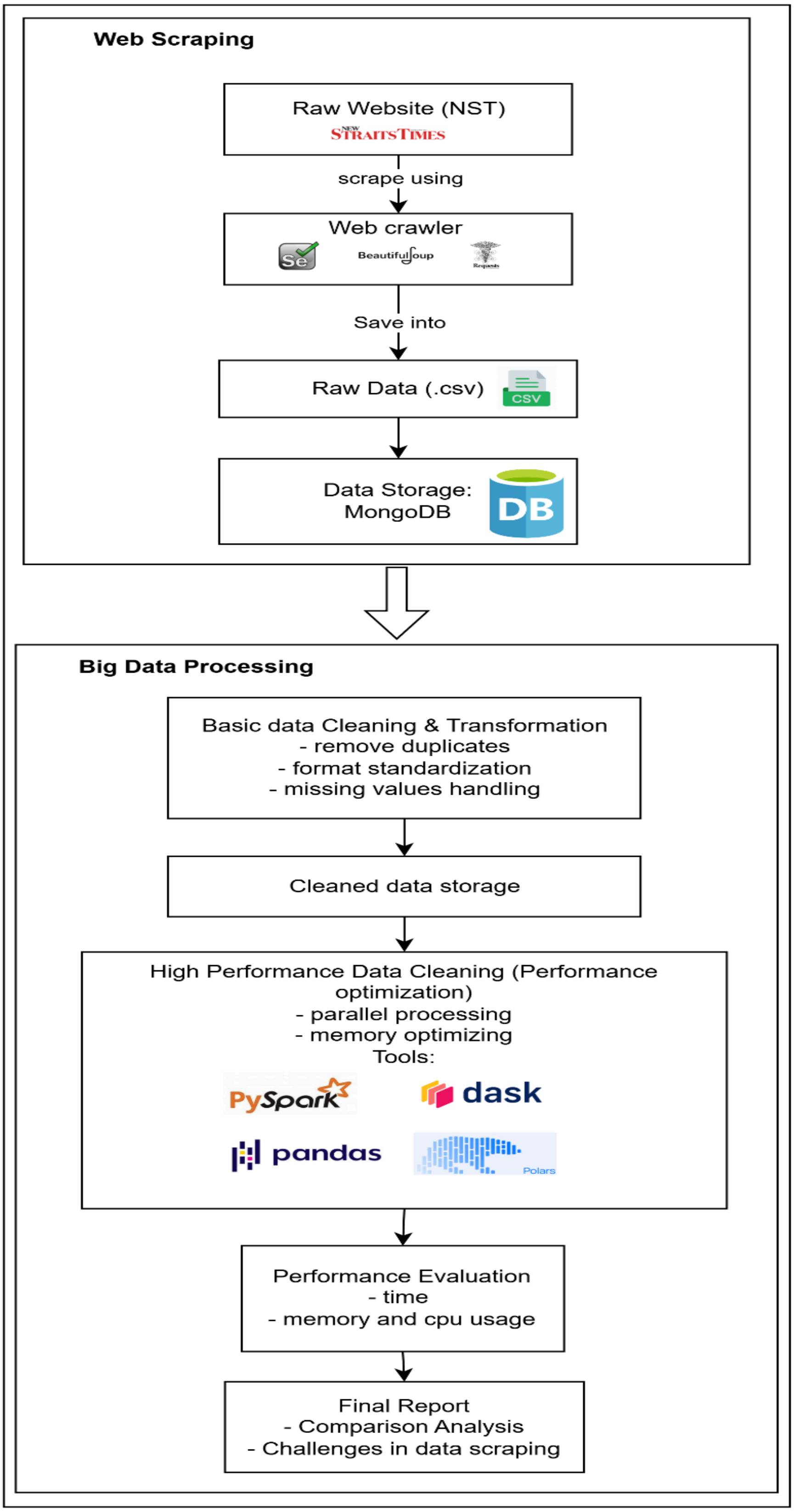


Figure 2.1: System Architecture for web scraping and performance evaluation

2.2 Tools and Framework used

Category	Software/Application/Library/Website
Reporting	Google Docs
System Architecture and Design	Draw.io
Web Scraping Library	Scrapy, BeautifulSoup, Selenium
Data Storage	CSV load to MongoDB
IDE	Google Colab, Vs Code
Coding Language	Python
Data Cleaning+Performance Evaluation	Python
Libraries used in process optimization	PySpark, Polars, Vectorized Pandas, Dask

2.3 Roles of team members

Member	**Role**	**Responsibility**	**Library in Charge**
Joseph Lau Yeo Kai	Web Scraping Process Designer	Design and develop whole web scraping code, at the same time ensures robots.txt and ethical data scraping is followed.	Polars
Nur Farah Adibah Binti Idris	Data Processing Specialist	Design and develop data cleaning process based on the data scraped from NST website. Ensure the cleaned data is consistent for each library used.	Dask
Vinesh A/L Vijaya Kumar	Process Efficiency Tester	Ensure resource usage and optimizes processing speed for web scraping and data cleaning.	Pandas+Vectorized Pandas
Tiew Chuan Shen	Data Visualization & Reporting	Design evaluation metrics and code, for producing visual reports in graphs. Combines all documentation and report for submission.	Pyspark

3. Data Collection

In this phase, there are about 127729 rows of data being scraped from the New Straits Times(NST) website, with each row of data containing attributes- section, date, headline and summary. We develop web crawlers with several libraries such as Scrapy, BeautifulSoup and Selenium, which are used for browsing through pages and handling pagination efficiently. After trial and error, we find out that the best solution for web scraping is the combination of BeautifulSoup and Selenium, which can navigate and browse through pages,

To follow the ethical standards and avoid overloading the server of the NST website, we follow the robots.txt of the NST website, and rate limitation is included in our web scraping process design through adding sleep time(delays) upon each page scraped. Besides, we ensured that only one person will be allowed to perform web scraping at the same time.

3.1 Crawling Method

1. Pagination Handling:
   Implemented systematic page navigation through the news sections
   Used URL parameter modification (?page=X) to access subsequent pages
   Implemented error handling for pagination failures
   Collected data from multiple news categories (Crime-courts and Nation sections)

```
def main():
    news_cat = ['crime-courts']
    all_articles = []

        driver = setup_driver()
    try:
        for cat in news_cat:
            news_path = f'/news/{cat}'
            full_path = urljoin(BASE_URL, news_path)

            for page in range(700,900):
                page_url = f"{full_path}?page={page}"
                articles_data = scrape_nst_articles(page_url, driver)
                all_articles.extend(articles_data)
```

2. Rate Limiting:

```
        time.sleep(5)  # Rate limiting
```

   5 seconds delay between page requests is implemented.

3.2 Number of Records Collected:

The data collection process successfully gathered a dataset with total records collected around 127,729 articles. Data fields per record include Section, Date, Headline, and Summary. Data is collected during multiple sessions across different time periods.

Section	Date	Headline	Summary
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	KL police confirm receiving Istana Negara's report over fake letter to King	KUALA LUMPUR: City police confirmed receiving a report from Istana Negara's representative over a letter allegedly written by Foreign Minister Datuk Seri Hishammuddin Hussein to ap
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Datuk and brother charged with making false claims for cooking oil subsidies	KUALA LUMPUR: A man with a 'Datuk' title and his brother were slapped with 52 counts of filing forged documents for subsidy payments totalling RM1.5 million, since two years ago.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Deputy, asst director charged with using false receipts for claims	KOTA KINABALU: A deputy and assistant director from the Malaysian Communication and Multimedia Commission (MCMC) were separately charged in the Sessions Court here with se
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	13 charged with MCO violation by participating in funeral procession [NSTTV]	
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Sergeant remanded over RM31,000 graft [NSTTV]	GEORGE TOWN: A police sergeant has been remanded for six days in connection with a RM31,000 graft case.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Man arrested for possessing protected birds	JOHOR BARU: The Wildlife and National Parks Department (Perhilitan) detained a man for possessing protected wild birds at a house in Ulu Tiram, here.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	11 held in drug party at Bukit Mertajam homestay	BUKIT MERTAJAM: 11 people, including five women, were detained during a drug-fuelled party at a homestay in Kompleks BM City here past midnight today.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	GOF foils attempts to smuggle subsidised items into Thailand	TUMPAT: The General Operations Force (GOF) foiled two attempts to smuggle Malaysian subsidised cooking oil, flour and sugar into Thailand with the seizure of the items worth abou
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	CITF lodges 2 police reports over sales of Covid vaccines	KUALA LUMPUR: The Covid-19 Immunisation Task Force (CITF) has lodged two police reports, one on June 14 and another on June 22 following public complaints on the selling activiti
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Three nabbed for selling Covid-19 vaccines	KUALA LUMPUR: Police arrested three people in Putrajaya and Brickfields here yesterday for allegedly being involved in the sale of Covid-19 vaccines.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Cops hunting man who sexually assaulted two children	PETALING JAYA: Police have launched a hunt for a man who is believed to have sexually assaulted two children at a block of flats in Section 19 here last Monday.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Jobless man jailed for trying to steal bank aircon compressor	Ersie Anjumin
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Two Pakistani men fined RM1,500 each for flouting MCO SOP	SELAYANG: Two Pakistani men who were caught not wearing face masks in front of a factory in Gombak were both fined RM1,500 each by the magistrate's court here today.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Police bust illicit cigarette trafficking syndicate in Tasek Gelugor	KEPALA BATAS: Bukit Aman Internal Security and Public Order Department (KDNKA) foiled an illicit cigarette trafficking syndicate with the seizure of 18,490,000 sticks of cigarettes wort
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Bung Moktar, Zizie Izette's corruption trial to resume July 26	KUALA LUMPUR: The corruption trial of Datuk Seri Bung Moktar Radin and his celebrity wife Datin Seri Zizie Izette A Samad will resume on July 26 due to the ongoing Movement Control (
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	28 cows worth about RM282,000 seized at Kelantan border	PASIR MAS: A 21-year-old lorry driver was arrested in possession of 28 heads of smuggled cattle worth about RM282,000 here this morning.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Game over for 10 caught gambling in bushes behind Lipis temple	LIPIS: A makeshift gambling den among bushes at a rubber plantation behind a temple at Kampung Baru Penjom near Kuala Lipis here did not stay hidden for long.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Man caught red-handed attempting to break CDM machine	BALIK PULAU: A man was caught red-handed trying to break a cash deposit machine (CDM) at an AmBank branch along Lintang Bukit Penara here last night.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	754 people arrested for violating SOP	KUALA LUMPUR: A total of 754 people were arrested for violating the Movement Control Order (MCO) standard operating procedures (SOP) yesterday.
CRIME & COURTS	Jun 25, 2021 @ 3:59pm	Man nabbed after driving 10km in stolen lorry	GEORGE TOWN: A man was detained shortly after he drove off in a lorry which he stole from the parking lot of a hypermarket in Bayan Baru here, this afternoon.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Man nabbed after driving 10km in stolen lorry	GEORGE TOWN: A man was detained shortly after he drove off in a lorry which he stole from the parking lot of a hypermarket in Bayan Baru here, this afternoon.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	152 bookies nabbed over betting in Euro Cup matches	
CRIME & COURTS	Jun 23, 2021 @ 11:08am	One nabbed, another escapes in 1km police chase in Kepala Batas	KEPALA BATAS: A man managed to escape while his friend was nabbed by police during a 1km-chase in Penaga here yesterday.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Doctor fined RM5,000 for publishing fake vaccine news	KANGAR: A doctor was fined RM5,000 by the Sessions Court here today, in default five months' jail, for publishing a video containing fake news about Sinovac vaccine in April.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Man arrested for threatening woman who supported student	KUALA LUMPUR: Police have arrested a man believed to have threatened a woman who showed support to Ain Husniza Saiful Nizam on Twitter last month.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Duo fined for harbouring illegal immigrants	KOTA KINABALU: Two 46-year-old locals were fined for allowing illegal immigrants to stay at their premises.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Man charged with trying to stab policeman	KOTA BELUD: An unemployed man who tried to stab a police corporal with a knife three years ago, was charged at the magistrate's court here today.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Five-day remand for father who allegedly fed baby alcoholic drink	BALIK PULAU: Police have obtained a five-day remand order against a young father who allegedly fed his infant son liquor straight from the can.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	5, including 3 women, detained in Pasir Mas over syabu, heroin posession	PASIR MAS: Three women and two men were arrested for possession of heroin and syabu during a raid at Kampung Banggol Petai, here, yesterday.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Cleaner in hit-and-run case faces murder, obstructing police, drug charges [NSTT	GEORGE TOWN: A cleaner, who was detained last Tuesday after police were forced to shoot at his rented Perodua Myvi following a fatal hit-and-run accident, has been charged with fo
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Man arrested with about RM28,000 worth of ketum leaves	PASIR MAS: Police detained a man and seized 80 plastic bags of ketum leaves worth about RM28,000 in an operation here today.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Police nab man who fed baby alcoholic drink	BALIK PULAU: Police have detained a 19-year-old man in connection with a 30-second video of him allegedly feeding a baby liquor straight from the can.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Pregnant woman to know fate in Nov after pleading guilty to 6 counts of bribery	KOTA KINABALU: A pregnant woman pleaded guilty to six counts of abusing her position to obtain bribes at the Special Corruption Court, here.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	July 16 for decision on IGP's bid to strike out Indira Gandhi's suit	KUALA LUMPUR: The High Court has fixed July 16 for the decision on the Inspector-General of Police's (IGP) bid to strike out a lawsuit by M. Indira Gandhi.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	NGO wants police to act against man who fed alcoholic drink to baby	GEORGE TOWN: Malaysian Anti-Cheap Liquor Movement (MACLM) president David Marshel lodged a police report at the Prai police station this afternoon in connection with a 30-secc
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Sakai Gang leader, 23 members arrested in four states [NSTTV]	KUALA LUMPUR: Police arrested the infamous Sakai Gang leader together with 23 other gang members after raids in Melaka, Negri Sembilan, Terengganu and Perak on Sunday.
CRIME & COURTS	Jun 23, 2021 @ 11:08am	Sabah MACC nabs man who bribed authorities in relation to gambling activities	TAWAU: The Sabah Malaysian Anti-Corruption Commission (MACC) has nabbed a man who had been bribing enforcement personnel in the district for years to conduct illegal gamblir
CRIME & COURTS	Jun 23, 2021 @ 11:08am	12th man held over funeral procession; police looking for two others	BUTTERWORTH: Police have detained yet another man in connection with a viral video footage of a group of men who were seen carrying a casket along Jalan Siram while others danc

Figure 3.2 Result of data collection process stored in csv

3.3 Ethical Considerations

The implementation has considered ethical web crawling practices to ensure responsible data collection:

1. Implemented delays between requests to prevent server overload
2. Used headless browser configuration to minimize resource usage
3. Implemented error handling to prevent excessive retry attempts
4. Collected only publicly available information
5. Followed NST's robots.txt guidelines

4. Data Processing

The data processing implementation is focused on data cleaning, transformation and storage using different libraries- Pandas, PySpark, Dask and Polars, for performance comparison and evaluation. Raw data is loaded from MongoDB for data cleaning process.

4.1 The cleaning and transformation methods

Data cleaning and transformation methods are applied based on data inconsistencies as below:

1. Null value
2. Duplicate value
3. Inconsistent capital/small number under the column-section
4. Wrong Date format

Code	Explanation

# Drop nulls and duplicates 	
df = df.dropna()
df = df.drop_duplicates(keep='first')
	Delete the rows which are duplicated or have null value.

# Standardize 'Section' column if 'Section' in df.columns: df["Section"] = df["Section"].str.title()	Change all data in section column into Title format.

# Clean 'Date' column 	
def clean_date(date):
if isinstance(date, str):
if "@" in date:
date =
date.split("@")[0].strip()
return re.sub(r'\s+', ' ', date)
return date
 	
if 'Date' in df.columns:
df["Date"] =
df["Date"].apply(clean_date)
df["Date"] =
pd.to_datetime(df["Date"],
errors='coerce')
	Change the format of date into formal format.

4.2 Data structure (Database)

The raw data and cleaned dataset are all stored in MongoDB. Data is called and stored using the code as below.

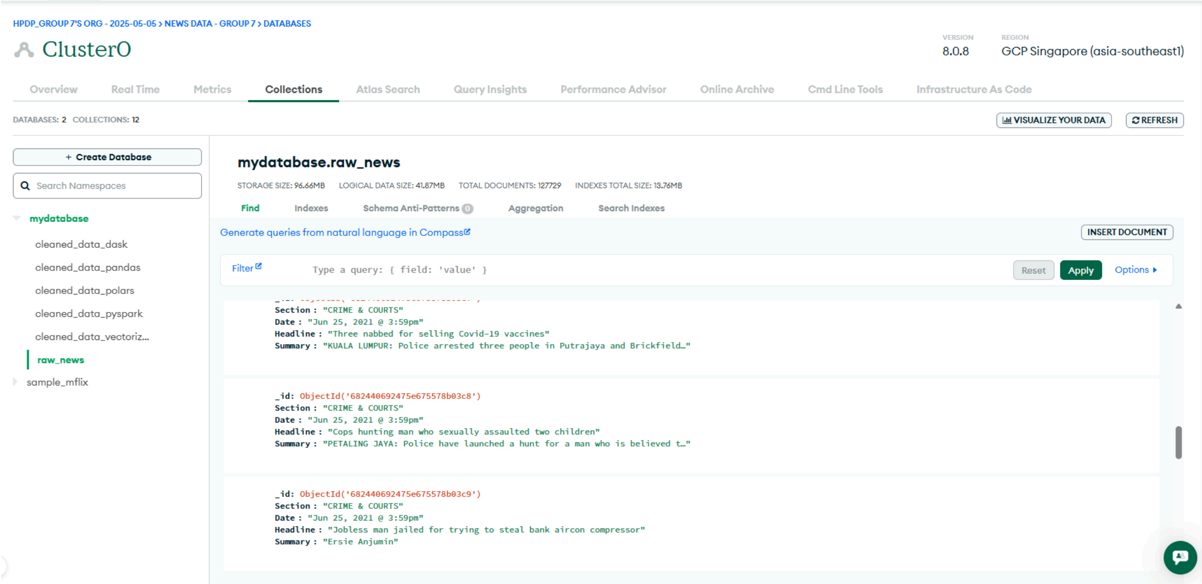


Figure 4.2 MongoDB as storage for raw and cleaned data

Below is the code snippet for connection to MongoDB

```
def connect_mongodb():
    """Connect to MongoDB and return the client."""
    uri =
"mongodb+srv://josephyeo:fPya67QIXrl4ZsV5@cluster0.ihjgjas.mongodb.net/?retryWrite
s=true&w=majority&appName=Cluster0"
    client = MongoClient(uri, server_api=ServerApi('1'))
    try:
        client.admin.command('ping')
        print("Successfully connected to MongoDB!")
        return client
    except Exception as e:
        print(f"Error connecting to MongoDB: {e}")
        return None

# Connect to MongoDB and load data
client = connect_mongodb()
db = client["mydatabase"]
```

5. Optimization Techniques

5.1 Methods used

In this project, the initial implementation for data processing uses the basic pandas library. To optimise the performance of the data processing, we use Polars, Dask, PySpark and Vectorised Pandas. The table below explains these optimization techniques.

Library / Technique	**Explanation on optimisation**
Polars	Built on Rust instead of Python and uses columnar data storage for faster data retrieval and more efficient processing,
Dask	Extends Pandas functionality by processing in parallel and lazily. In the optimisation code, map_partitions are used to divide datasets into partitions for parallel operations.
Pyspark	Uses Apache Spark for distributed data processing.
Vectorised Pandas	Eliminate .apply() in basic pandas code and change it into vectorised functions. Vectorised pandas use Pandas functions that operate on entire columns instead of using .apply() and loops.

5.2 Code overview of techniques applied

The code overview demonstrates different optimization ways of cleaning the ‘Date’ Column, the uncleaned ‘Date’ Column format is String “Jun 25, 2021 @ 3:59pm”, processed into datatype Date 2021-06-25.

Technique	**Code overview for ‘Date’ column processing**

Polars	if 'Date' in df.columns:
	df = df.with_columns(
	df['Date']
	.str.split('@')
	.list.first()
	.str.strip_chars()
	.alias('Date')
	)
	
	# Convert to datetime
	df=df.with_columns(pl.col('Date').str.strptime(pl.Datetime, format='%b %d, %Y', strict=False).alias('Date'))

Dask	df_cleaned = df_cleaned.map_partitions(lambda df: df.assign(Date=df['Date'].map(lambda x: x.split('@')[0].strip() if isinstance(x, str) else x))) df_cleaned = df_cleaned.map_partitions(lambda df: df.assign(Date=pd.to_datetime(df['Date'], errors='coerce')))

Pyspark	# Remove time part
	df_cleaned = df_cleaned.withColumn("Date", regexp_replace(col("Date"), "@.*$", ""))
	# Normalize spaces
	df_cleaned = df_cleaned.withColumn("Date", regexp_replace(col("Date"), "\s+", " "))
	# Trim spaces
	df_cleaned = df_cleaned.withColumn("Date", trim(col("Date")))
	# Parse to date
	df_cleaned = df_cleaned.withColumn("Date", to_date(col("Date"), "MMM d, yyyy"))

Vectorised Pandas	if 'Date' in df.columns: df['Date'] = df['Date'].str.split('@').str[0].str.strip() df['Date'] = df['Date'].str.replace(r'\s+', ' ', regex=True) df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

6. Performance Evaluation

6.1 Before vs after optimization

Initially, we would process the data with plain Pandas, which was decent for small data but would take forever to execute with large-scale datasets, particularly when filtering, aggregating, and joining. To address such shortcomings and boost performance, we introduced and experimented with some high-performant alternatives: Vectorized Pandas, Dask, Polars, and PySpark.

•**Vectorized Pandas** enhanced standard Pandas operations by eliminating slow.apply() and loop constructs and instead using native, column-based operations. This yielded dramatic speed gains with minimal alteration of code.

•**Dask** extended Pandas with parallelism and lazy evaluation, allowing for efficient handling of datasets that do not fit into memory, by partitioning the data and processing in parallel.

•**Polars**, which was implemented with Rust having a multi-threaded, columnar backend, offered the optimum performance for in-memory data processing operations. It greatly outperformed others both in speed and memory usage.

•**PySpark**, while heavier due to its distributed architecture, provided consistent performance for extremely large data sets. While experiencing some initialization overhead, it performed reasonably well in a local distributed environment.

After using all of these optimizations, execution times decreased dramatically on all tasks. Polars handled best overall in terms of speed and memory usage. Dask and Vectorized Pandas handled well too, especially on smaller-to-medium-sized datasets. PySpark was slower in certain scenarios but handled best on distributed loads.

6.2 Comparison of Code Execution Time, Peak Memory Usage, CPU usage and Throughput

Operation	Aspects	Comparisons				
		Dask	Polars	Pyspark	Pandas	Vectorized Pandas
Dataset Loading and Display	Code Execution Time (s)	0.5574	0.13728	0.1715	1.45037	0.81039
	Peak Memory Usage (MB)	9.566	0.3828	0.0	0.0	0.0
	Throughput (rows/s)	217847.737	884573.6415	707961.62	83728.7675	149849.72

Table 2: Comparison between Data Processing and Cleaning Techniques

Conclusion:

Polars > Vectorized Pandas > Dask > Pandas > PySpark

•**Polars:** Offers the overall best performance in every category. With its Rust foundation, multi-threaded operation, and columnar in-memory storage, it was able to handle massive volumes of data with extremely low memory utilization and extremely high rates of computation.

•**Vectorized Pandas:** Displayed extensive improvement over traditional Pandas by embracing efficient, column-based operations. It provided sturdy throughput with zero memory usage, which made it extremely effective for medium-sized datasets without parallel or distributed systems.

•**Dask:** Performed better than standard Pandas by allowing parallelism and chunked data processing. Though slower than Vectorized Pandas for this application, Dask excelled on the scaling aspect and was memory-friendly.

•**Pandas:** Although easy to manipulate, Pandas showed large memory usage and sluggish run times with increasing data size. It was the least scalable but set a good baseline.

•**PySpark:** Registered worst performance during this test in local-mode mode due to too much initialization overhead and resource consumption. But it remains a strong candidate for scaled distributed processing, particularly in multi-node or cluster modes.

6.3 Charts and graphs

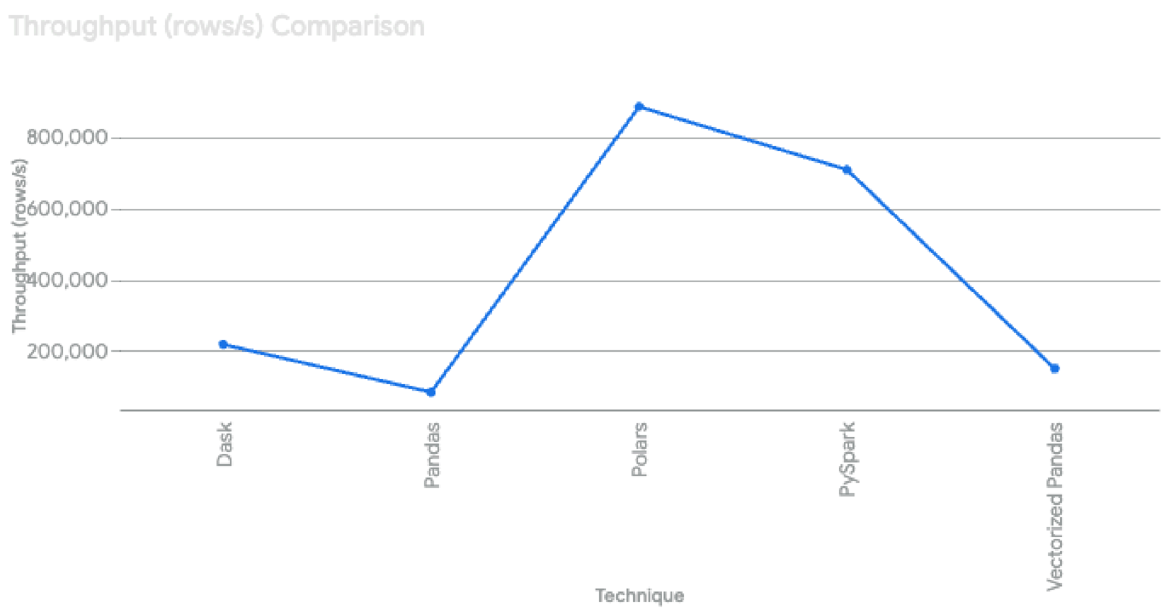


Figure 2.2: shows Throughput(rows/s) of each optimization

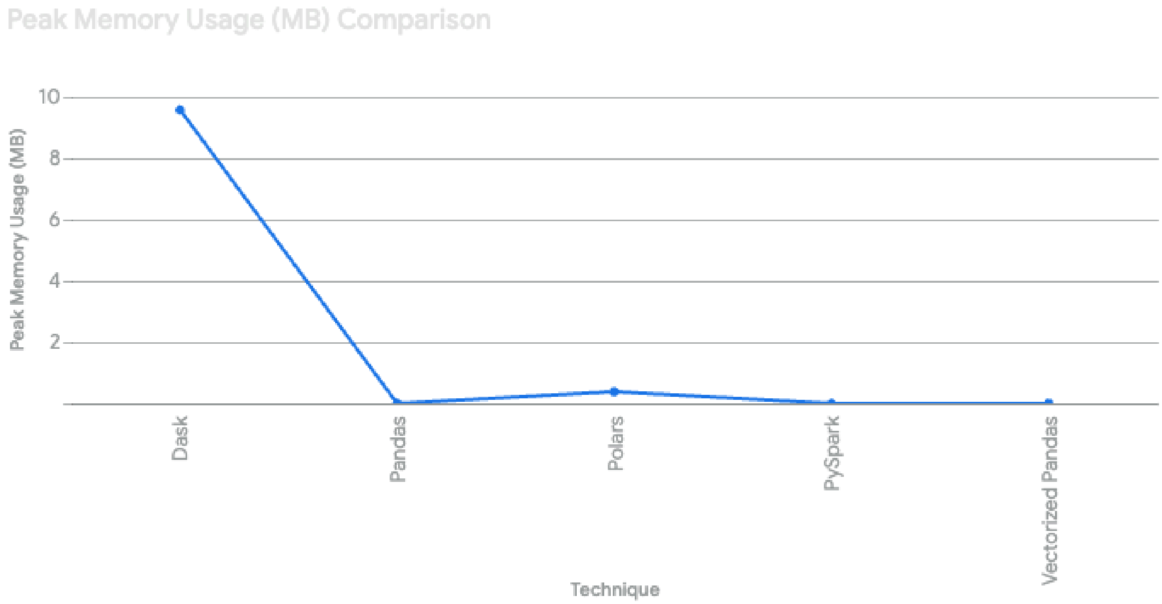


Figure 2.3: shows Peak Memory Usage(MB) of each optimization

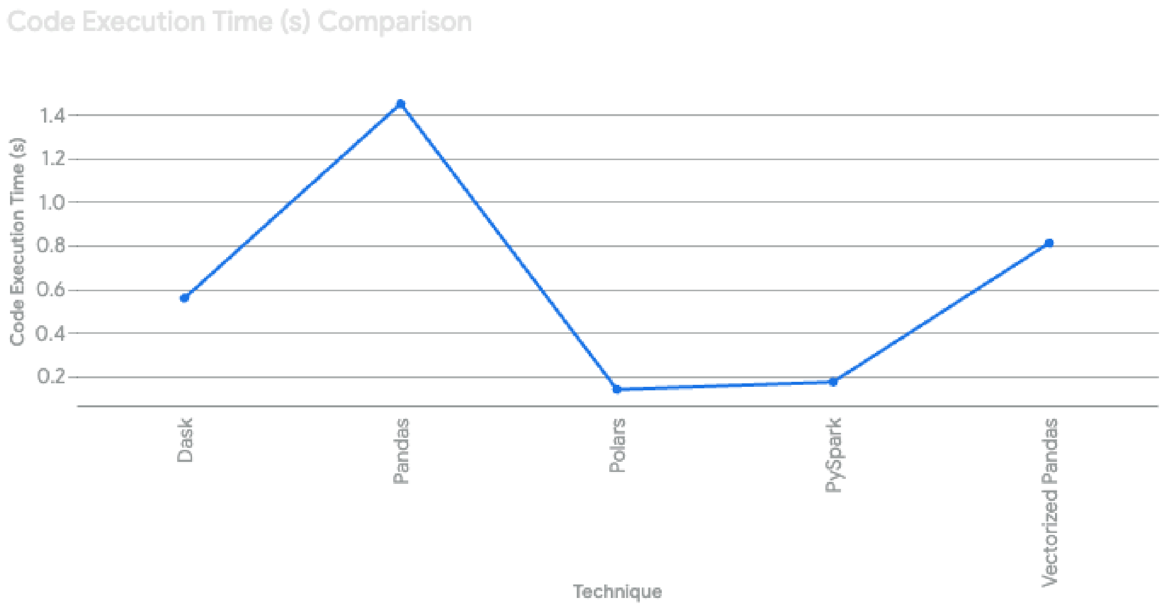


Figure 2.4: shows Code Execution Time(s) of each optimization

7. Challenges & Limitations

The development of the web crawler faced several unforeseen challenges that required the team to pivot from the original plan. Initially, each team member was assigned a different web scraping library—such as Scrapy, BeautifulSoup, and Selenium—with the intention of comparing their effectiveness and dividing the workload evenly. However, during testing, this approach proved unreliable. Scrapy and BeautifulSoup encountered difficulties in handling dynamic page content, pagination, and JavaScript-rendered sections of the New Straits Times (NST) website. As a result, the team collectively decided to shift to using Selenium, which provided more robust interaction with the browser interface and ensured higher success rates in extracting the required data. While Selenium was effective, it introduced new limitations: it was slower due to its reliance on a full browser rendering, and it was more resource-intensive. Additionally, to comply with ethical web scraping practices and avoid violating the site's terms of service, the crawler was intentionally limited to a single user at a time with enforced delays between each request. These measures, while necessary to protect the integrity of the NST website, significantly slowed down the data collection process, making it less scalable for very large datasets or real-time applications.

In terms of data processing, the team encountered several technical challenges that impacted the overall efficiency and performance of the project. The scraped data contained inconsistent formats, particularly in the 'Date' column, which required multiple stages of string manipulation and type conversion to standardize. Initial attempts using Pandas were straightforward but became inefficient as the dataset grew beyond 100,000 records. To optimize performance, the team implemented alternative libraries including Polars, Dask, PySpark, and Vectorized Pandas. While these libraries offered significant improvements in speed and memory efficiency, they also introduced new complexities. For instance, Dask, although capable of parallel processing, showed unexpectedly high memory consumption during some operations, possibly due to internal task graph overheads. PySpark, while powerful for distributed computing, suffered from long initialization times and was less suitable for smaller-scale or interactive tasks in a local development setup. Polars emerged as the fastest and most memory-efficient option, but its API was less familiar to the team and required additional learning time. Another limitation was that the performance benchmarks were only conducted as single-run evaluations, without multiple repetitions to account for variability in system load or execution time. This limits the statistical reliability of the performance metrics presented.

Furthermore, the entire pipeline—from data extraction to cleaning and optimization—was specifically designed around the structure and content of the NST website. As such, the solution may not generalize well to websites with different structures, inconsistent formatting, or anti-scraping protections like CAPTCHA or dynamic loading. Future implementations would need to consider more adaptable scraping strategies and modular data processing pipelines. Lastly, while the current system works effectively in a controlled, academic environment, it lacks a user-friendly interface or automation script that could assist non-technical users in selecting the best processing method based on data size, hardware limitations, or performance requirements.

8. Conclusion & Future Work

8.1 Summary of findings

In conclusion, Polars library is the most suitable data processing tool for New Straits Times dataset. From the time comparison graph, Polars only used 0.14 seconds to process all data cleaning operations, which is faster than the other libraries. If compared to pandas, Polars can achieve more than 30x performance gains. This can be seen through the throughput 884573.64 rows per second compared to the 83728.77 rows per second for pandas. Moreover, memory usage for Polars is only 0.38 MB. Therefore, Polars's overall performance is the most efficient and is considered high performance with the help of multiprocessing and multithreading methods.

8.2 What could be improved

The data transformation for Polars can be performed in the future to test the processing speed. The conversion of data type is a challenging task because the initial dataset type is a string data type. As a result, the string data type is not suitable for future analysis and is hard to transform into actionable insight or data-driven insight. Besides that, the performance metrics are based on a single run. Future work could include multiple runs to account for variability and provide more reliable averages. More complex transformations or larger datasets can be implemented to better stress-test the methods. Not only that, future work can include an investigation into why Dask's memory usage was higher and if it can be optimized further. Extend the analysis to include data from other sources, such as SQL databases, to evaluate performance in different contexts. A user-friendly interface or script can be developed to allow users to select the best processing method based on their dataset size and hardware constraints.

9. References

1. https://blog.jetbrains.com/pycharm/2024/07/polars-vs-pandas/
2. https://pythonspeed.com/articles/pandas-vectorization/

10. Appendices

10.1 Sample code snippets

```
!pip install polars>=0.20.0
!pip install "pymongo[srv]>=4.6.0"
!pip install matplotlib>=3.8.0
!pip install seaborn>=0.13.0
!pip install psutil>=5.9.0
!pip install numpy>=1.24.0


import time
```

```python
import psutil
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import  regex as re
from bson import ObjectId
from datetime import datetime
from multiprocessing import Pool, cpu_count
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi
# polars
import polars as pl
from concurrent.futures import ProcessPoolExecutor, ThreadPoolExecutor

#pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import when, col, regexp_replace, to_date, 
trim, initcap

#dask
import dask.dataframe as dd

#pandas
import pandas as pd
```

Figure 1: Code for Installations and Imports

```python
def connect_mongodb():
    """Connect to MongoDB and return the client."""
    uri = 
"mongodb+srv://josephyeo:fPya67QIXrl4ZsV5@cluster0.ihjgjas.mongodb.net/
?retryWrites=true&w=majority&appName=Cluster0"
    client = MongoClient(uri, server_api=ServerApi('1'))
    try:
        client.admin.command('ping')
        print("Successfully connected to MongoDB!")
        return client
    except Exception as e:
        print(f"Error connecting to MongoDB: {e}")
        return None
```

```
# Connect to MongoDB and load data
client = connect_mongodb()
db = client["mydatabase"]
```

Figure 2: Code for MongoDB Connection

```
file_path =
'https://raw.githubusercontent.com/Jingyong14/HPDP02/refs/heads/main/24
25/project/p1/Group%207/data/raw_data.csv'

# Read the CSV
df_news = pd.read_csv(file_path)
df_news.head()

row_count = len(df_news)
print("Total number of row:", row_count)

# Create or switch to your database
db = client["mydatabase"]
news_collection = db["raw_news"]

# Delete existing data in the collection
news_collection.delete_many({})

# Insert the all rows into MongoDB
news_collection.insert_many(df_news.to_dict("records"))

print("All rows of news data inserted into MongoDB successfully.")
```

Figure 3: Code for Loading Data

```
def track_performance(method_name, start_time, start_memory, df):
    """Track performance metrics for data cleaning operations."""
    end_time = time.time()
    end_memory = psutil.Process().memory_info().rss / (1024 * 1024)  #
MB

    time_taken = end_time - start_time
    throughput = len(df) / time_taken if time_taken > 0 else 0
    memory_used = end_memory - start_memory  # in MB

    return {
        "Method": method_name,
        "Time (s)": time_taken,
        "Throughput (rows/s)": throughput,
```

```
        "Memory Used (MB)": memory_used
    }

def track_performance_pyspark(method_name, start_time, start_memory,
df=None):
    end_time = time.time()
    end_memory = psutil.Process().memory_info().rss / (1024 * 1024)

    time_taken = end_time - start_time

    if df is not None:
        if hasattr(df, "count"):  # PySpark DataFrame
            row_count = df.count()
        else:  # pandas, dask, polars
            row_count = len(df)
    else:
        row_count = 0

    throughput = row_count / time_taken if time_taken > 0 else 0
    memory_used = end_memory - start_memory  # in MB

    return {
        "Method": method_name,
        "Time (s)": round(time_taken, 4),
        "Throughput (rows/s)": round(throughput, 2),
        "Memory Used (MB)": round(memory_used, 2)
    }
```

Figure 4: Performance Tracking Function Code

```
# Load documents into a DataFrame
db = client["mydatabase"]
collection = db['raw_news']
data = list(collection.find())
df_panda = pd.DataFrame(data)

def clean_data(df):
    start_time = time.time()
    start_memory = psutil.Process().memory_info().rss / (1024 * 1024)

    df = df.drop(columns=['_id'])

    # Drop nulls and duplicates
    df = df.dropna()
```

```
    df = df.drop_duplicates(keep='first')

    # Standardize 'Section' column
    if 'Section' in df.columns:
        df["Section"] = df["Section"].str.title()

    # Clean 'Date' column
    def clean_date(date):
        if isinstance(date, str):
            if "@" in date:
                date = date.split("@")[0].strip()
            return re.sub(r'\s+', ' ', date)
        return date

    if 'Date' in df.columns:
        df["Date"] = df["Date"].apply(clean_date)
        df["Date"] = pd.to_datetime(df["Date"], errors='coerce')  #
Invalid dates become NaT


    # Track performance using shared function
    performance_report = track_performance("Pandas ", start_time,
start_memory, df)

    return df, performance_report


cleaned_df, pandas_result = clean_data(df_panda)

print("Final row: ", len(cleaned_df))
# Save cleaned data to MongoDB
cleaned_data = cleaned_df.to_dict("records")
db["cleaned_data_pandas"].delete_many({})
db["cleaned_data_pandas"].insert_many(cleaned_data)

print("Cleaned data inserted into 'cleaned_data_pandas' collection in
MongoDB")
print("")
print("Pandas cleaning completed!")
print(pandas_result)
print("")

# Save cleaned data to CSV
```

```
cleaned_df.to_csv("cleaned_data_unoptimized_pandas.csv", index=False)
print("Cleaned data saved to 'cleaned_data_unoptimized_pandas.csv'")
```

Figure 5: Pandas Data Processing Code

```
def load_data(client):
    """Load data from MongoDB into a Polars DataFrame."""
    db = client["mydatabase"]
    collection = db['raw_news']
    data = list(collection.find())
    df = pl.DataFrame(data)
    return df

df_polars = load_data(client)

def clean_data_polars_default(df):
    """Clean data using Polars' default processing."""
    start_time = time.time()
    start_memory = psutil.Process().memory_info().rss / (1024 * 1024)

    df = df.drop('_id')

    # Enable string cache for better performance
    with pl.StringCache():

        # Define list of fake nulls
        fake_nulls = ["", "NaN", "null"]

        # Apply replacement for all columns
        df = df.with_columns([
            pl.when(pl.col(col).is_in(fake_nulls))
              .then(None)
              .otherwise(pl.col(col))
              .alias(col)
            for col in df.columns
        ])
        # Drop duplicates and nulls
        df = df.drop_nulls()
        df = df.unique()
        # Standardize Section
        if 'Section' in df.columns:
            df = df.with_columns(
                df['Section'].str.to_titlecase().alias('Section')
            )
```

```
        # Clean Date
        if 'Date' in df.columns:
            df = df.with_columns(
                df['Date']
                .str.split('@')
                .list.first()
                .str.strip_chars()
                .alias('Date')
            )

            # Convert to datetime
            df = df.with_columns(
               pl.col('Date').str.strptime(pl.Datetime, format='%b %d,
%Y', strict=False).alias('Date')
            )
            df = df.filter(pl.col("Date").is_not_null())

    return df, track_performance("Polars", start_time, start_memory,
df)

# Run default Polars processing
df_polar_cleaned, polar_result = clean_data_polars_default(df_polars)

print("Final row: ", len(df_polar_cleaned.to_pandas()))
# Save cleaned data to MongoDB
polar_cleaned = df_polar_cleaned.to_dicts()
db["cleaned_data_polars"].delete_many({})
db["cleaned_data_polars"].insert_many(polar_cleaned)
print("Cleaned data inserted into 'cleaned_data_polars' collection in
MongoDB")
print("")

df_polar_cleaned =
df_polar_cleaned.to_pandas().to_csv("cleaned_data_optimized_polar.csv",
index=False)
print("Cleaned data saved to 'cleaned_data_optimized_polar.csv'")
print("")

print("Polars cleaning completed!")
print(polar_result)
```

Figure 6: Polars Data Processing Code

```
df_dask = dd.from_pandas(df_panda, npartitions=4)
```

```
# Function to clean data using Dask (default scheduler)
def clean_data_dask_default(df_dask):
    start_time = time.time()
    start_memory = psutil.Process().memory_info().rss / (1024 * 1024)
# MB

    df_cleaned = df_dask.drop(columns=['_id'])

    df_cleaned = df_cleaned.map_partitions(lambda df: df.dropna())

    # Perform drop_duplicates() in Pandas (to check across all rows),
then switch back to Dask
    df_cleaned_pandas = df_cleaned.compute()
    df_cleaned_pandas =
df_cleaned_pandas.drop_duplicates(subset=["Section", "Date",
"Headline", "Summary"])

    # Convert back to Dask DataFrame after dropping duplicates
    df_cleaned = dd.from_pandas(df_cleaned_pandas, npartitions=4)

    df_cleaned = df_cleaned.map_partitions(lambda df:
df.assign(Section=df['Section'].str.title()))
    df_cleaned = df_cleaned.map_partitions(lambda df:
df.assign(Date=df['Date'].map(lambda x: x.split('@')[0].strip() if
isinstance(x, str) else x)))
    df_cleaned = df_cleaned.map_partitions(lambda df:
df.assign(Date=pd.to_datetime(df['Date'], errors='coerce')))

    # Compute to Pandas
    cleaned_df = df_cleaned.compute()

    # Track performance
    performance_report = track_performance("Dask ", start_time,
start_memory, cleaned_df)

    return cleaned_df, performance_report

df_dask_cleaned, dask_result = clean_data_dask_default(df_dask)

print("Final row: ", len(df_dask_cleaned))
# Save cleaned data to MongoDB
dask_cleaned = df_dask_cleaned.to_dict("records")
db["cleaned_data_dask"].delete_many({})
```

```
db["cleaned_data_dask"].insert_many(dask_cleaned)


df_dask_cleaned.to_csv("cleaned_data_optimized_dask.csv", index=False)
print("Cleaned data saved to 'cleaned_data_optimized_dask.csv'")
print("")

print("Cleaned data inserted into 'cleaned_data_dask' collection in
MongoDB")
print("")
print("Dask cleaning completed!")
print(dask_result)
```

Figure 7: Dask Data Processing Code

```
def clean_data_pyspark(df_spark):
    start_time = time.time()
    start_memory = psutil.Process().memory_info().rss / (1024 * 1024)

    # Drop NA and Duplicates
    df_cleaned= df_spark.drop('_id')

    # List of "fake" nulls to replace
    fake_nulls = ["", "NaN", "null"]

    # Replace in all columns
    for c in df_cleaned.columns:
        df_cleaned = df_cleaned.withColumn(c,
            when(col(c).isin(fake_nulls), None).otherwise(col(c))
        )

    df_cleaned = df_cleaned.dropna()
    df_cleaned = df_cleaned.dropDuplicates()

    # Clean and convert the Date column
    df_cleaned = df_cleaned.withColumn("Date",
regexp_replace(col("Date"), "@.*$", ""))  # Remove time part
    df_cleaned = df_cleaned.withColumn("Date",
regexp_replace(col("Date"), "\s+", " "))  # Normalize spaces
    df_cleaned = df_cleaned.withColumn("Date", trim(col("Date")))  #
Trim spaces
    df_cleaned = df_cleaned.withColumn("Date", to_date(col("Date"),
"MMM d, yyyy"))  # Parse to date
    df_cleaned = df_cleaned.filter(col("Date").isNotNull())
```

```
    # Format Section to Title Case
    df_cleaned = df_cleaned.withColumn("Section",
initcap(col("Section")))

    return df_cleaned, track_performance_pyspark("PySpark", start_time,
start_memory, df_cleaned)

# Initialize Spark
spark = SparkSession.builder.appName("CleanNewsData").getOrCreate()

# Convert MongoDB ObjectId to string
df_panda["_id"] = df_panda["_id"].astype(str)

# Convert to PySpark DataFrame
df_spark = spark.createDataFrame(df_panda)

# Clean and track performance
df_spark_cleaned, pyspark_result = clean_data_pyspark(df_spark)

# Save cleaned data to MongoDB
# Convert PySpark DataFrame to Pandas DataFrame
df_spark_cleaned_pandas = df_spark_cleaned.toPandas()

# Fix dates in pandas df
df_spark_cleaned_pandas =
df_spark_cleaned_pandas[pd.to_datetime(df_spark_cleaned_pandas["Date"],
errors="coerce").notna()]
df_spark_cleaned_pandas["Date"] =
pd.to_datetime(df_spark_cleaned_pandas["Date"])
print("Final row: ", len(df_spark_cleaned_pandas))

# Convert to dictionary for MongoDB insertion
spark_cleaned = df_spark_cleaned_pandas.to_dict("records")
db["cleaned_data_pyspark"].delete_many({})
db["cleaned_data_pyspark"].insert_many(spark_cleaned)
print("Cleaned data inserted into 'cleaned_data_pyspark' collection in
MongoDB\n")

# Save to CSV
df_spark_cleaned_pandas.to_csv("cleaned_data_optimized_pyspark.csv",
index=False)
```

```
print("Cleaned data saved to 'cleaned_data_optimized_pyspark.csv'\n")

print("Pyspark cleaning completed!")
print(pyspark_result)
```

Figure 8: PySpark Data Processing Code

```
# Optimized vectorized cleaning function
def clean_data_pandas_vectorized(df):
    start_time = time.time()
    start_memory = psutil.Process().memory_info().rss / (1024 * 1024)

    # Drop nulls and duplicates in one go (vectorized)
    df = df.drop(columns=['_id'])
    df = df.dropna().drop_duplicates()

    # Standardize 'Section' column (vectorized str methods)
    if 'Section' in df.columns:
        df['Section'] = df['Section'].str.title()

    # Clean 'Date' column (vectorized string methods + datetime)
    if 'Date' in df.columns:
        # Clean strings using vectorized apply (no need for a loop)
        df['Date'] = df['Date'].str.split('@').str[0].str.strip()
        df['Date'] = df['Date'].str.replace(r'\s+', ' ', regex=True)

        # Convert to datetime in one call (vectorized)
        df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

    # Track performance (reuse track_performance function)
    performance_report = track_performance("Vectorized Pandas",
start_time, start_memory, df)

    return df, performance_report

# Test the function
df_vectorized_cleaned, vectorized_result =
clean_data_pandas_vectorized(df_panda)

# Save cleaned data to MongoDB
vectorized_cleaned = df_vectorized_cleaned.to_dict("records")
db["cleaned_data_vectorized_pandas"].delete_many({})
db["cleaned_data_vectorized_pandas"].insert_many(vectorized_cleaned)
```

```
print("Cleaned data inserted into 'cleaned_data_vectorized_pandas'
collection in MongoDB")
print("")

df_vectorized_cleaned.to_csv("cleaned_data_optimized_vectorized.csv",
index=False)
print("Cleaned data saved to 'cleaned_data_optimized_vectorized.csv'")
print("")
print("Vectorized pandas cleaning completed!")
print(vectorized_result)
```

Figure 9: Vectorized Pandas Data Processing Code

```
def plot_results(results):
    """Plot performance comparison results in 3 columns"""
    # Convert results to pandas DataFrame for plotting
    results_df = pl.DataFrame(results).to_pandas()

    # Set plot style
    sns.set(style="whitegrid")

    # Define metrics and color palette
    metrics = ["Time (s)", "Throughput (rows/s)", "Memory Used (MB)"]
    palette = sns.color_palette("pastel",
n_colors=len(results_df['Method'].unique()))

    # Create a single row of 3 subplots
    fig, axes = plt.subplots(1, 3, figsize=(12,4))

    for i, metric in enumerate(metrics):
        ax = axes[i]
        sns.barplot(
            x="Method", y=metric, hue="Method", legend=False,
            data=results_df, palette=palette, ax=ax
        )

        # Add values on top of the bars
        for p in ax.patches:
            ax.annotate(f'{p.get_height():.2f}',
                        (p.get_x() + p.get_width() / 2.,
p.get_height()),
                        ha='center', va='center', fontsize=10,
color='black',
                        xytext=(0, 5), textcoords='offset points')
```

```
        ax.set_title(f"{metric} Comparison", fontsize=14)
        ax.set_xlabel("")
        ax.set_ylabel(metric)

    plt.tight_layout()
    plt.show()


# Collect all results
results = [polar_result, pandas_result]

# Plot results
plot_results(results)
```

Figure 10: Plot Result Code

```
def plot_results_overall(results):
    """Plot performance comparison results in 3 columns
(side-by-side)."""
    import matplotlib.pyplot as plt
    import seaborn as sns
    import polars as pl

    # Convert results to pandas DataFrame
    results_df = pl.DataFrame(results).to_pandas()

    # Set seaborn style
    sns.set(style="whitegrid")

    # Define metrics and color palette (5 distinct pastel colors)
    metrics = ["Time (s)", "Throughput (rows/s)", "Memory Used (MB)"]
    unique_methods = results_df['Method'].unique()
    palette = dict(zip(unique_methods, sns.color_palette("pastel",
n_colors=len(unique_methods))))

    # Create subplots
    fig, axes = plt.subplots(1, 3, figsize=(16, 5))

    for i, metric in enumerate(metrics):
        ax = axes[i]
        sns.barplot(
            x="Method", y=metric, hue="Method",
            data=results_df, palette=palette, legend=False, ax=ax
```

```
        )

        # Annotate bar values
        for p in ax.patches:
            height = p.get_height()
            ax.annotate(f'{height:.2f}',
                        (p.get_x() + p.get_width() / 2., height),
                        ha='center', va='bottom', fontsize=9,
color='black',
                        xytext=(0, 5), textcoords='offset points')

        ax.set_title(f"{metric} Comparison", fontsize=13)
        ax.set_xlabel("")
        ax.set_ylabel(metric)
        ax.tick_params(axis='x', labelrotation=45)

    plt.tight_layout()
    plt.show()


# Collect all results
overall = [pandas_result, vectorized_result, polar_result, dask_result,
pyspark_result ]

# Plot results
plot_results_overall(overall)
```

Figure 11: All Comparison Code Among Libraries

```
# Close the MongoDB connection
client.close()
```

Figure 12: Clean Up Code

10.2 Screenshots of output

```
Requirement already satisfied: pymongo>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from pymongo[srv]>=4.6.0) (4.12.1)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in /usr/local/lib/python3.11/dist-packages (from pymongo>=4.6.0->pymongo[srv]>=4.6.0) (2.7.0)
WARNING: pymongo 4.12.1 does not provide the extra 'srv'
```

Output 1: Installations complete

```
Successfully connected to MongoDB!
```

Output 2: Connected to MongoDB

```
Total number of row: 127729
All rows of news data inserted into MongoDB successfully.
```

Output 3: Data insertion into MongoDB

```
Final row:  121437
Cleaned data inserted into 'cleaned_data_pandas' collection in MongoDB

Pandas cleaning completed!
{'Method': 'Pandas ', 'Time (s)': 1.450361728668213, 'Throughput (rows/s)': 83728.76752029915, 'Memory Used (MB)': 0.0}

Cleaned data saved to 'cleaned_data_unoptimized_pandas.csv'
```

Output 4: Pandas Data Processing

```
Final row:  121437
Cleaned data inserted into 'cleaned_data_polars' collection in MongoDB

Cleaned data saved to 'cleaned_data_optimized_polar.csv'

Polars cleaning completed!
{'Method': 'Polars', 'Time (s)': 0.1372830867767334, 'Throughput (rows/s)': 884573.6415986606, 'Memory Used (MB)': 0.3828125}
```

Output 5: Polars Data Processing

```
<ipython-input-143-fedb2e56790b>:19: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and a
  df_cleaned = df_cleaned.map_partitions(lambda df: df.assign(Date=pd.to_datetime(df['Date'], errors='coerce')))
Final row:  121437
Cleaned data saved to 'cleaned_data_optimized_dask.csv'

Cleaned data inserted into 'cleaned_data_dask' collection in MongoDB

Dask cleaning completed!
{'Method': 'Dask ', 'Time (s)': 0.5574398040771484, 'Throughput (rows/s)': 217847.7373015031, 'Memory Used (MB)': 9.56640625}
```

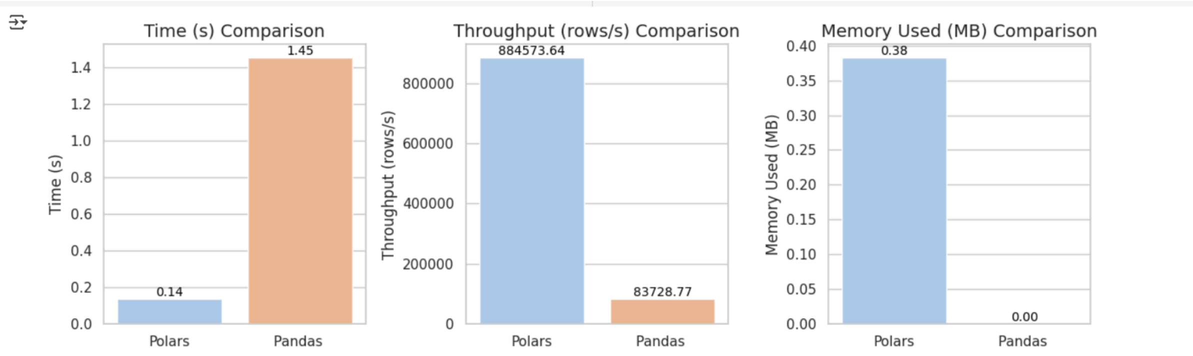
Output 6: Dask Data Processing

```
Final row:  121437
Cleaned data inserted into 'cleaned_data_pyspark' collection in MongoDB

Cleaned data saved to 'cleaned_data_optimized_pyspark.csv'

Pyspark cleaning completed!
{'Method': 'PySpark', 'Time (s)': 0.1715, 'Throughput (rows/s)': 707961.62, 'Memory Used (MB)': 0.0}
```

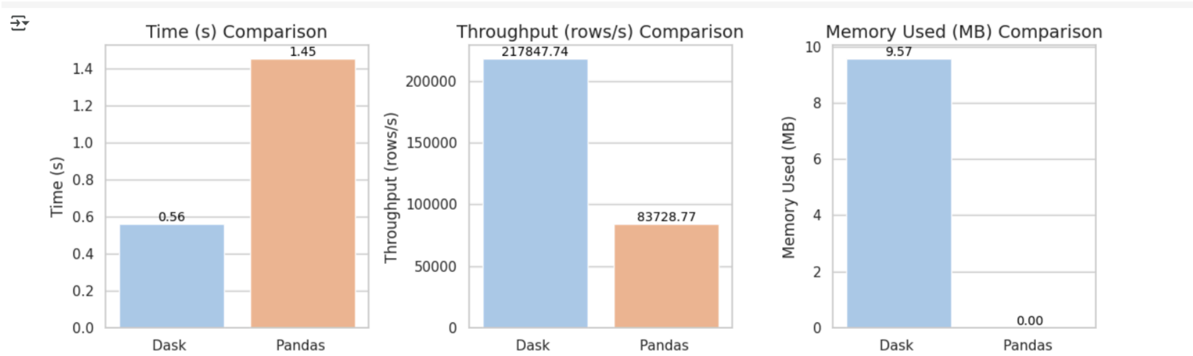
Output 7: PySpark Data Processing

```
Cleaned data inserted into 'cleaned_data_vectorized_pandas' collection in MongoDB

Cleaned data saved to 'cleaned_data_optimized_vectorized.csv'

Vectorized pandas cleaning completed!
{'Method': 'Vectorized Pandas', 'Time (s)': 0.810391902923584, 'Throughput (rows/s)': 149849.72031667858, 'Memory Used (MB)': 0.0}
```

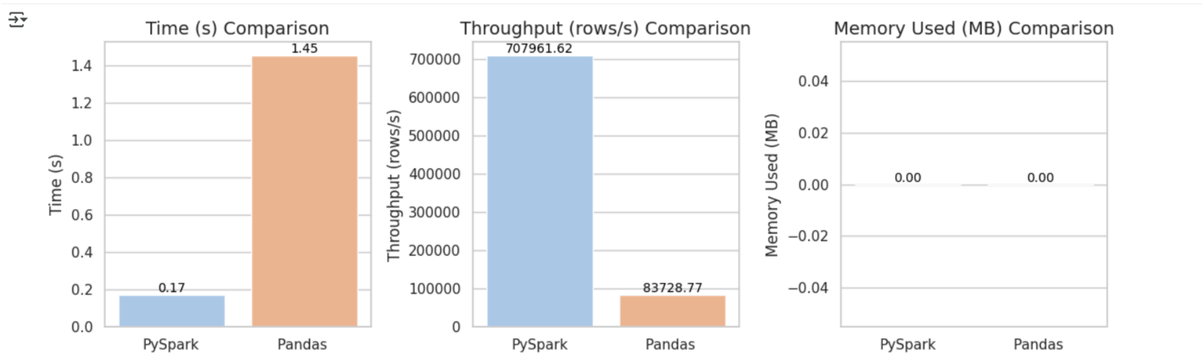
Output 8: Vectorized Pandas Data Processing



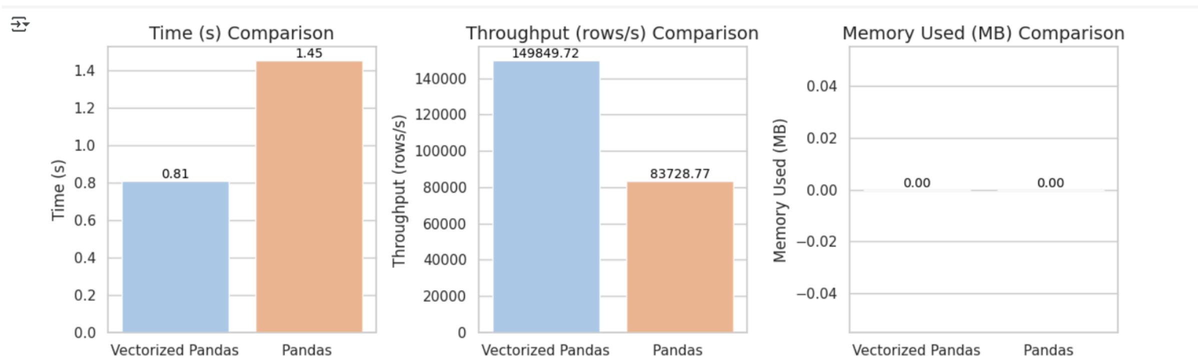
Output 9: Graph of Polars vs Pandas



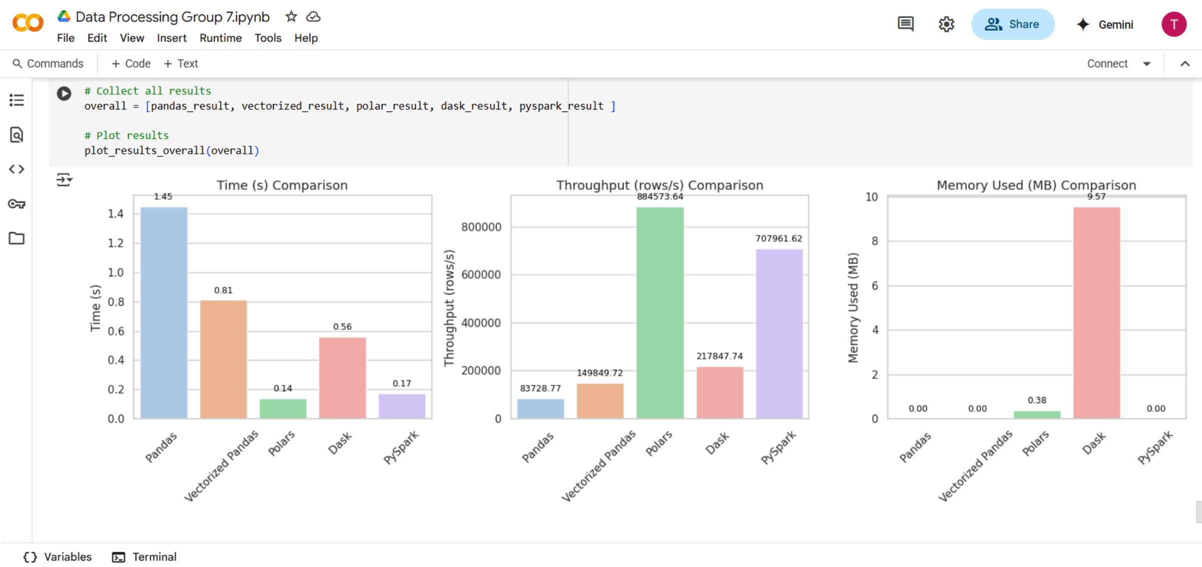
Output 10: Graph of Dask vs Pandas



Output 11: Graph of PySpark and Pandas



Output 12: Graph of Vectorized Pandas vs Pandas



Output 13: All Graphs Compared with Pandas

10.3 Links to full code repo or dataset

Raw Dataset link:
https://github.com/Jingyong14/HPDP02/blob/main/2425/project/p1/Group%207/data/raw_data.csv
Data Processing File Link:
https://colab.research.google.com/drive/1khMiYXUq926IHhzo20M8q18WZEOx_QCy?usp=sharing#scrollTo=w1RDQKQkfOYt