Healthcare Resource Allocation platform

Borsi Sonia[1], Costamagna Filippo[2]

[1]sonia.borsi@studenti.unitn.it

[2]filippo.costamagna@studenti.unitn.it

***Abstract*— The effective distribution of healthcare resources is essential for enhancing patient outcomes and guaranteeing the long-term viability of health services. This paper outlines the development and implementation of an innovative healthcare resource allocation app designed to assist Australian policymakers by providing data-driven insights into the healthcare system. The dashboard incorporates a range of big data technologies, including Apache Spark, RabbitMQ, PostgreSQL, and Streamlit, to process and present extensive datasets from the Australian Institute of Health and Welfare (AIHW) in conjunction with data from the Australian Bureau of Statistics (ABS). By analysing a wide range of metrics, the dashboard enables informed decision-making and strategic resource allocation. The system's architecture is designed for scalability and reliability, with automated daily data refreshes to ensure that the most up-to-date information is available.**

***Keywords*— Healthcare Resource Allocation, Healthcare Efficiency, Big Data, ETL, Data Visualization, Resource Optimization, Scalability**

I. Introduction

In the complex healthcare landscape, the efficient allocation of resources is critical to improving patient outcomes and maintaining the long-term viability of health services. This paper presents an innovative application designed to address these challenges by using detailed and actionable data from healthcare facilities across Australia.

The primary goal of the application is to provide a comprehensive, data-driven view of the Australian healthcare ecosystem, revealing trends and variations in resource utilisation across hospitals and regions. By integrating and analysing a wide range of metrics the application provides a nuanced perspective on operational efficiency and identifies areas for improvement.

Fig. 1 Australian Healthcare

Designed for government decision-makers, this app enables informed policy-making by providing a clear, real-time snapshot of healthcare performance. The insights generated by the application are intended to guide strategic interventions and resource allocation, ultimately improving the effectiveness of healthcare services. This paper outlines the development journey of the application, detailing its system architecture, the technologies used and the underlying model. It also describes the implementation phases and concludes with a discussion of the results achieved through this innovative approach.

II. System Model

A. *System architecture*

The system architecture designed to efficiently allocate healthcare resources in Australia is a multi-stage pipeline, designed to ensure the seamless flow of data from initial collection to final visualisation.

The first stage of the pipeline (Fig. 2) is the **data retrieval** phase, where healthcare data is extracted from the Australian Institute of Health and Welfare (AIHW) API, which is open and free to use. Available data includes:

- Hospital data – Data on elective surgery, emergency department care, admissions, length of stay, available services, and more.
- Multi-level data – Data may be available at multiple levels, from the hospital level to aggregate figures at local hospital network,

state or territory, and national levels. Availability can vary across data collections.

- Geographic data about hospitals – The API offers location data and some mappings for Australian hospitals.

Once the data has been retrieved, the system enters the **message queue**, where RabbitMQ acts as a crucial intermediary. RabbitMQ acts as a robust message broker, facilitating the asynchronous transfer of data between different stages of the pipeline. This stage ensures the reliable delivery of data from the retrieval phase to the processing phase. By publishing data to RabbitMQ queues, the system decouples the different stages, allowing them to operate independently while maintaining an orderly flow of information. Data remains in the queue until it is ready to be consumed by the next stage, maintaining the efficiency and reliability of the system.

During the **data processing and storage** stage, raw data is refined and prepared for storage using Apache Spark. Spark plays a critical role in this stage by consuming data from RabbitMQ and distributing the processing workload across a cluster of two worker nodes. This distributed processing enables the efficient handling of large volumes of data in parallel, significantly accelerating the data refinement process. Once the data is processed, it is stored in a PostgreSQL database, which serves as the persistent storage solution. The database is then exposed to other system components via a standard JDBC interface, ensuring seamless access for the final stage of the pipeline.

The architecture's core data models, stored in PostgreSQL, are structured to support efficient querying and data integrity.

TABLE I : DATABASE STRUCTURE

TABLES		
Table name	**Columns**	**Primary key**
datasets	reporting start date, reported measure code, dataset it, measure code, dataset name, stored	dataset id
hospitals	code, name, type, latitude, longitude, sector, open/closed, state, lhn, phn	code
measurements	measure code, measure name	measure code
reported measurements	reported measure code, reported measure name	reported measure code
info	dataset id, reporting unit code, value, caveats, id	id

These tables are linked together in the PostgreSQL database, allowing efficient storage and retrieval of data, which is then used for comprehensive analysis and visualisation in the Streamlit application.

The final stage of the architecture is the **data visualisation** phase, where the processed data is transformed into meaningful insights. This stage uses Streamlit, a powerful web application framework that enables the creation of interactive data applications. Streamlit enables users to interact with the data through a web interface, providing real-time insights and facilitating informed decision making. By querying the PostgreSQL database, Streamlit retrieves the required data and presents it in various graphical formats such as charts, tables and dashboards.

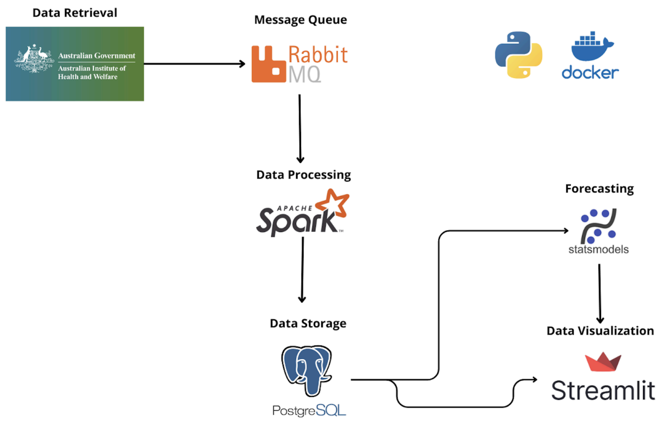


Fig. 2 Application Workflow

B. *Technologies*

The development of the Australian Efficient Resource Allocation App has been supported by a carefully selected suite of big data technologies, each chosen for its ability to address specific aspects of the system's architecture. These technologies work seamlessly together to create a robust, scalable application capable of efficiently processing large volumes of health data. These technologies were carefully selected based on the critical factors of volume, velocity and variety of health data being processed. Specifically, we needed to manage large volumes of data, comprising over 5,000 records, primarily in structured formats such as CSV files. In addition, as the data is updated annually, batch processing was an essential consideration to ensure that the system could handle these updates efficiently.

Apache Spark was chosen as the primary data processing technology because of its powerful distributed computing capabilities. Spark's ability to distribute tasks across a cluster makes it particularly effective in handling the substantial data volumes ingested from the RabbitMQ queue, ensuring high performance and fault tolerance. Given the structured nature of the data, Spark's robust support for structured data processing through its DataFrame API increases the efficiency and scalability of the system. In addition, Spark's compatibility with batch processing perfectly matches the annual update schedule of the datasets, ensuring that the application can efficiently manage the influx of updated data each year without compromising performance.

RabbitMQ was chosen as the message broker to facilitate the asynchronous transfer of data between the data retrieval and processing stages. RabbitMQ's message queuing reliability ensures the smooth transfer of data between different components of the pipeline, even in scenarios where processing tasks are delayed or the system experiences high loads. Its ability to decouple stages of the architecture allows for independent scaling and maintenance of each component, increasing the overall resilience of the system. In addition, RabbitMQ's support for different messaging patterns and its ability to handle high throughput scenarios make it particularly well suited to the volume of data involved in the application. By efficiently managing the flow of over 5,000 datasets, RabbitMQ ensures that data is consistently and reliably delivered to Apache Spark for processing, even under varying loads. This design choice also allows the system to accommodate the variety of data sources and formats, while maintaining the flexibility to handle both real-time and batch processing, increasing the scalability and robustness of the system.

PostgreSQL was chosen as the database management system because of its robustness, scalability and strong support for complex queries. Given the structured nature of our healthcare data, a relational SQL database such as PostgreSQL is the optimal choice for effectively managing and querying this data. PostgreSQL's ability to handle large datasets and perform efficient queries makes it an excellent solution for storing processed healthcare data. In addition, its support for ACID (Atomicity, Consistency, Isolation, Durability) properties ensures data integrity - an essential feature in healthcare applications where data accuracy and reliability are paramount. The structured nature of the data further reinforces the need for a relational database, making SQL the most appropriate choice for our requirements.

Streamlit was selected as the framework for the data visualisation component because of its simplicity and powerful capabilities for creating interactive data applications. Streamlit enables the rapid development of user-friendly interfaces that allow government decision-makers to interact with and explore data in real time. Its seamless integration with Python and other data processing tools makes it particularly well suited to visualising data stored in PostgreSQL.

Python serves as the primary programming language for the application, chosen for its versatility, extensive libraries and strong community support. Python's ability to integrate with various tools and technologies used in the application - such as Spark, RabbitMQ, PostgreSQL and Streamlit - ensures a consistent development environment and simplifies the implementation of complex data processing tasks. Along with python, we utilized a series of libraries that are not specifically developed for big data but serve the specific purpose in which we employed them. For example, we used Pandas to open excel files as it offers for options than Apache Spark and Statsmodels to do time series forecasting, as the data is updated annually and it would take a very long period of time to make the amount of data difficult to handle.

Docker was employed for containerisation purposes, thereby providing a consistent environment for the deployment of the application at various stages of its development and production. Docker's containerisation capabilities ensure that the application remains portable, scalable and easy to manage, regardless of the underlying infrastructure.

Each of these technologies has been selected based on its ability to meet the specific needs of the application, contributing to a cohesive, efficient system that facilitates informed decision making in the allocation of healthcare resources. The application ensures efficient handling of large amounts of data, fast data processing and the ability to manage different types of data.

III. Implementation

The code for our project is fully available at https://github.com/SoniaBorsi/Healthcare-Resource-Allocation. The repository consists of four folders, three of which are relevant to the execution process. The data folder contains the external data necessary to gain more meaningful insights from the data retrieved by the API, as well as the images to be used in the dashboard. The docker folder contains the three dockerfiles required to set up the services that require a more complex configuration. Finally, the src folder contains all the Python code. The processing subfolder contains another subfolder called utilities, which contains all the functions needed for the ETL process. It also contains setup.py, which sets up the database schema and starts the Spark session used by all the other processing scripts. Additionally, it contains ETL.py, which acts as the main script to retrieve, process and load the data into the database. Additionally, the repository contains a Docker-Compose file, which is used to orchestrate and set up the various containers. A bash script is also included, which spins up the containers and schedules a daily run of the ETL process to collect any new data that may be available on the API.

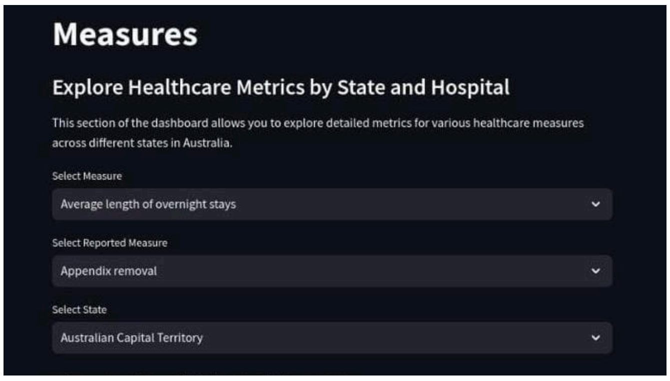
The process is as follows: firstly, the list of all hospitals is retrieved, including information such as geolocation, type and opening status. The corresponding table in the database is then consulted to find any new records to insert. Similarly, the list of available records is fetched and any new records are added to the corresponding table, with the "stored" column set to false. Subsequently, we identify the IDs of unstored records by querying the database and divide the list of such IDs into 20-record chunks to load the corresponding measurements into the 'info' table. Each record is retrieved in CSV format and the entire batch is concatenated and sent as a message by RabbitMQ. This is then processed by Apache Spark and inserted into the database.

To obtain more detailed information about the process structure, please refer to the scripts and relevant comments. To initiate the project, simply clone the repository and execute the run.sh script. It is advisable to ensure that the script is executable on your device, as otherwise you may require the necessary permissions. You can monitor the progress of the process via the logs in the terminal. To stop the script, simply use the keyboard interrupt (typically Ctrl + C) or locate the PID and execute the kill command. Once all the containers have been launched, please navigate to the dashboard at http://localhost:8080 . Please be aware that the data fetching process may take some time. It is therefore recommended that the dashboard is run once the data storing process is complete, in line with the current project status. This will be improved in a future update.

IV. Results

The final result of this project is a Healthcare Resource Allocation Dashboard (Fig. 3) that serves as a tool for decision-makers, providing critical insights into various aspects of the Australian healthcare system. The dashboard is organized into 3 main sections: Measures, Hospitals and Budget.

The Measure section of the dashboard is the primary analytical tool and is designed to provide a comprehensive overview of healthcare metrics across different states and hospitals in Australia. Users can examine a comprehensive range of measures, reported measures and state of interest, viewing each combination in a time series format to gain insight into how these metrics have evolved over time, observe trends and use the foreasting section to plan future resource allocation. In addition the user can explore a wide range of plots about the selected measure: the distribution of values, heatmaps of the variation across different hospitals, the ranking of hospitals, and a correlation analysis with other measures.



Measures

Explore Healthcare Metrics by State and Hospital

This section of the dashboard allows you to explore detailed metrics for various healthcare measures across different states in Australia.

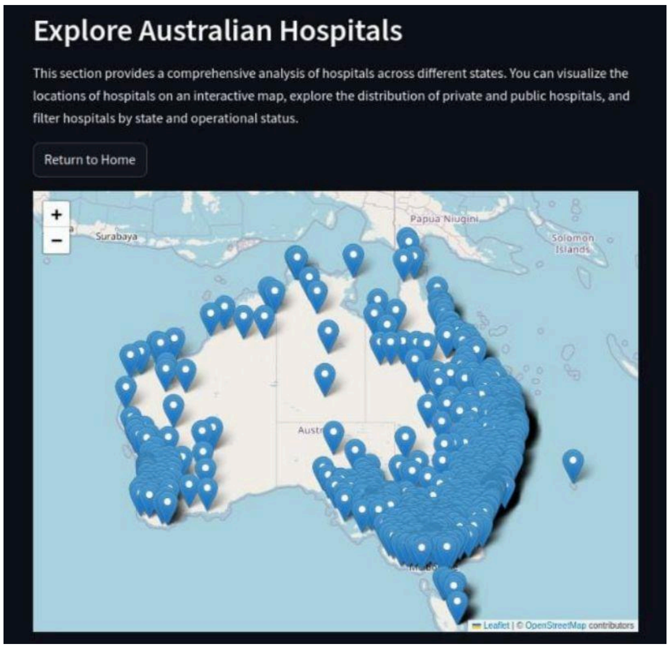
Select Measure: Average length of overnight stays

Select Reported Measure: Appendix removal

Select State: Australian Capital Territory

Fig. 3 Measures section of the dashboard

The Hospitals section provides an analysis of the geographic aspect of healthcare facilities across Australia. It is designed to provide a comprehensive overview of hospital locations, types and sectors, offering valuable insight into the Australian healthcare landscape. The interactive map allows users to visualise the distribution of hospitals and assess the geographic spread of healthcare services, enabling them to identify potential areas of under- or over-servicing.



Explore Australian Hospitals

This section provides a comprehensive analysis of hospitals across different states. You can visualize the locations of hospitals on an interactive map, explore the distribution of private and public hospitals, and filter hospitals by state and operational status.

Fig. 3 Hospitals section of the dashboard

The Budget section of the dashboard is designed for the analysis of healthcare spending at both state and national levels. This section offers users valuable insights into the allocation of financial resources across the healthcare system and the evolution of spending trends over time. Users can investigate historical data on healthcare spending in both current and constant prices, enabling them to evaluate the influence of inflation and economic shifts on healthcare funding. Furthermore, the Budget section incorporates analytical tools for examining the correlation between healthcare spending and GDP, providing a macroeconomic viewpoint on resource allocation.

The application is able to successfully process and manage large datasets retrieved from the Australian Institute of Health and Welfare API. Apache Spark's distributed computing capabilities were instrumental in processing these datasets. One of the key advantages of this app is its capacity to remain current with the latest data, facilitated by an automated daily data refresh process. The Spark job is run every 24 hours to ensure that the most recent data from the AIHW is retrieved, processed and loaded into the database. The integration of RabbitMQ as a message broker guarantees the reliability of data transfer from the AIHW API and the database. PostgreSQL, with its relational structure, is able to support efficient storage and retrieval of the processed health data, and its ACID compliance ensures that such data remains accurate and consistent.

V. Conclusions

The Healthcare Resource Allocation Dashboard developed in this project has demonstrated significant potential as a tool for enhancing decision-making in the Australian healthcare system. By integrating various big data technologies, such as Apache Spark, RabbitMQ, PostgreSQL, and Streamlit, the application effectively processes, analyzes, and visualizes complex healthcare data, offering crucial insights for informed policy-making.

Despite its robust capabilities, the system has some limitations and there are several areas where the application could be improved. One potential improvement is to expand the Spark cluster beyond its current configuration by modifying the Docker Compose setup, allocating more cores or memory to each Spark worker, or simply adding more of them. This would allow the system to scale and handle even larger datasets or more complex analytics, making it adaptable to future increases in data volume or processing requirements. In addition, incorporating machine learning models alongside the time series forecasting we have implemented could provide better predictive analytics capabilities, enabling the system to more accurately predict future trends in healthcare resource allocation and patient outcomes.

Another potential enhancement would be to integrate additional data sources, such as real-time patient data or socioeconomic indicators, in order to gain a more comprehensive understanding of the factors influencing healthcare outcomes. This would further enhance the dashboard's insights, making it an even more powerful tool for decision-makers. One significant constraint was the limitation of the Australian Bureau of Statistics API, which allows only 10 calls per day. Consequently, we had to utilise the required data without utilising the API, instead downloading it manually.

Overall, the Healthcare Resource Allocation app could represent a significant step forward in using data-driven approaches to optimize healthcare services in Australia. By continuously updating with the latest data and offering scalable infrastructure, the application ensures that healthcare resources are allocated efficiently and equitably, ultimately contributing to better patient outcomes and a more sustainable healthcare system. Future developments should focus on expanding the system's capabilities to address its current limitations and to meet the evolving needs of the healthcare sector.

References

[1] Docker Documentation, https://www.docker.com/
[2] PostgreSQL Documentation, https://www.postgresql.org/
[3] Apache Spark Documentation, https://spark.apache.org/
[4] Streamlit Documentation, https://spark.apache.org/
[5] AIHW (Australian Institute of Health and Wealth), https://www.aihw.gov.au/
[6] MyHospitals API Documentation, https://www.aihw.gov.au/reports-data/myhospitals/content/api
[7] Australian Bureau of Statistics, https://www.abs.gov.au