Assignment 1 - Building Your Big Data Platforms

CS-E4640 Big Data Platforms

Samuli Öhman
Aalto University
Finland

ABSTRACT

This report presents the design, implementation, and evaluation of *mysimbdp*, a simplified big data platform for scalable data storage and ingestion. The platform leverages Apache Cassandra (*mysimbdp-coredms*) for its distributed architecture, fault tolerance, and scalability. It includes a data ingestion service (*mysimbdp-dataingest*) that reads data from various sources and stores it into the core database. This report details the system architecture, implementation strategies, performance evaluations, and potential enhancements, such as service discovery and platform-as-a-service integration. The platform demonstrates robustness and provides insights into scalability and fault tolerance.

1 INTRODUCTION

The rapid growth of data has created a demand for scalable and reliable platforms capable of handling big data workloads. This report documents the development of *mysimbdp*, a platform designed to provide efficient data storage and ingestion for multiple tenants. It supports diverse application domains, including IoT analytics and financial data processing, while ensuring high availability and fault tolerance.

The primary objectives of this project include:

- Designing a scalable data storage service (*mysimbdp-coredms*) using Apache Cassandra.
- Implementing a data ingestion service (*mysimbdp-dataingest*) for reading and storing data.
- Ensuring fault tolerance through a carefully configured cluster.
- Evaluating performance under varying workloads and deployment scenarios.

The platform is designed to support structured, semi-structured, and unstructured data, with mechanisms for partitioning and replication to ensure performance and reliability.

2 SYSTEM ARCHITECTURE

2.1 Platform Components

The *mysimbdp* platform consists of the following components:

- **mysimbdp-coredms**: A distributed database built on Apache Cassandra, providing scalable and fault-tolerant data storage.
- **mysimbdp-dataingest**: A service that reads data from tenant sources, processes it, and stores it in *mysimbdp-coredms*.
- **External Services**: Cloud providers (e.g., GCP) for compute/networking and monitoring tools.

2.2 Component Interactions

The platform operates as follows:

(1) Tenant data sources generate data.
(2) *mysimbdp-dataingest* reads, processes, and ingests the data into *mysimbdp-coredms*.
(3) Tenants access their data via APIs provided by the platform. APIs not implemented yet but could be very easily in the future.
(4) Service discovery maps tenants to their dedicated *mysimbdp-coredms* instances.

2.3 Cluster Configuration

The Cassandra cluster is configured for high availability:

- Two nodes are deployed in different zones: Europe-North1-a and Europe-West1-b.
- A replication factor of 2 ensures redundancy and fault tolerance.
- Configurable consistency levels balance performance and data integrity.

2.4 Deployment Considerations

- *mysimbdp-dataingest* is deployed near data sources to minimize ingestion latency but may increase write latency to geographically distant database nodes.
- The platform scales horizontally by adding nodes or deploying additional ingestion instances.

2.5 Advanced Features

The architecture supports:

- **Data Lineage**: Metadata tracking for data origin, transformations, and destination.
- **Service Discovery**: Mapping tenants to their databases via registries like ZooKeeper.
- **PaaS Integration**: A new component, *mysimbdp-daas*, provides APIs for external data producers/consumers.
- **Hot/Cold Data Management**: Automatic policies classify and move data based on access frequency.

2.6 Architecture Diagram

Below is a high-level diagram of the system architecture:

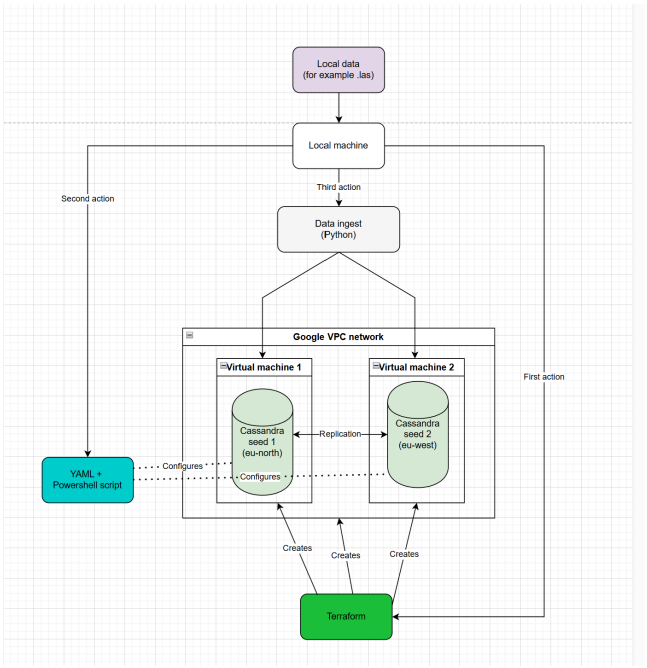

Figure 1: High-level architecture

The diagram illustrates data flow from tenant sources through *mysimbdp-dataingest* into *mysimbdp-coredms*, along with interactions with external services and advanced features.

3 IMPLEMENTATION

3.1 Data Schema for Tenants

The selected dataset is a LiDAR point cloud dataset containing spatial data points (x, y, z coordinates). The schema in *mysimbdp-coredms* is designed to optimize query performance and scalability:

- **Keyspace:** `lidar_data`, configured with a replication factor of 2 to ensure fault tolerance.
- **Table:** `las_data`, storing individual data points.
- **Primary Key:**
  - Partition Key: `lidar_id` (unique identifier for each dataset).
  - Clustering Columns: `depth` and `measurement_type`.
- **Attributes:**
  - `depth`: Represents the z-coordinate of the point.
  - `measurement_type`: Specifies the type of measurement (e.g., x, y).
  - `value`: Stores the actual measurement value.

This schema ensures efficient querying by grouping related data within partitions while maintaining even distribution across nodes.

3.2 Data Partitioning Strategy

Data is partitioned by `lidar_id` to group related data within partitions. This strategy achieves:

- **Performance:** Minimizes cross-node queries by clustering related data.
- **Scalability:** Ensures even distribution of data across the cluster.
- **Fault Tolerance:** Leverages Cassandra's replication mechanism to maintain availability during node failures.

The partitioning aligns with the replication factor of 2, ensuring redundancy and high availability.

3.3 Data Ingestion Implementation

The *mysimbdp-dataingest* component reads LiDAR LAS files, processes them, and stores data in Cassandra. Key implementation details include:

- **Atomic Unit:** Each row represents a single measurement point (`lidar_id`, `depth`, `measurement_type`, `value`).
- **Batch Processing:** Data is ingested in batches of up to 1,000 rows using prepared statements to minimize overhead.
- **Consistency Level:** Set to `TWO` to ensure strong consistency with two nodes, though this impacts performance.
- **Error Handling:** Retry logic handles transient failures, such as network interruptions or node unavailability.

3.4 Deployment Adjustments

To optimize deployment:

- Nodes are deployed across availability zones (e.g., Europe-North1-a and Europe-West1-b) to prevent single points of failure.
- Monitoring tools like Prometheus and Grafana could be used to track cluster health and performance metrics.
- Auto-scaling can be implemented to dynamically add nodes based on workload demands.

3.5 Screenshots

Below are screenshots of key commands used during implementation and testing:

```
PS C:\Users\samuli\Desktop\School\Big data platform\Project\big_data_platform\data_ingestion> gcloud compute instances list
NAME          ZONE            MACHINE_TYPE   PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP    STATUS
eu-west-node  europe-west1-b  e2-standard-2               10.132.0.2   34.38.191.215  RUNNING
nordics-node  europe-north1-a  e2-standard-2               10.166.0.2   34.88.206.210  RUNNING
```

Figure 2: Displays all Google Cloud Compute instances.

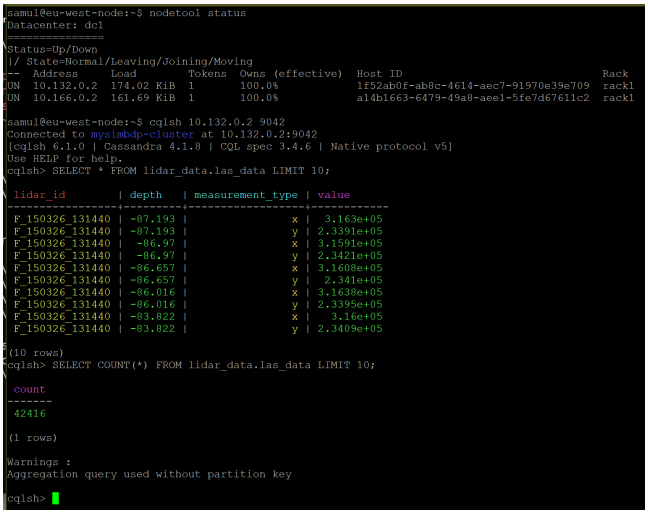

Figure 3: Shows the status of the Cassandra cluster and sample data from the `las_data` table.

4 PERFORMANCE EVALUATION

Performance tests were conducted to evaluate the platform's behavior under varying conditions. The key findings are summarized below:

- **Concurrency Levels:** Tests were performed with 1 and 2 concurrent ingestion instances. The number of rows ingested per second per instance did not vary significantly with increased concurrency.
- **Consistency Levels:** Two consistency levels were tested—`ONE` and `TWO`. The ingestion throughput (rows per second) remained nearly identical for both levels, indicating that consistency settings had minimal impact on performance.

Despite these observations, the overall ingestion speed per instance was unexpectedly low. This could be attributed to one or more of the following factors:

- **Networking Limitations:** Each instance may have been constrained by a networking limit, reducing the rate at which data could be ingested.
- **Script Inefficiency:** Potential inefficiencies in the Python ingestion script (e.g., suboptimal batch processing or excessive logging) might have contributed to slower ingestion speeds.
- **External Communication Bottleneck:** While internal communication within the Google VPC network (using private IPs) was fast and did not appear to be a bottleneck, external communication exhibited lower speeds, possibly due to network throttling or latency.

Further investigation is needed to pinpoint the exact cause of the low ingestion speeds and optimize the system accordingly.

5 CONCLUSION AND FUTURE WORK

5.1 Summary

This project successfully designed, implemented, and evaluated *mysimbdp*, a simplified big data platform tailored for scalable and fault-tolerant data storage and ingestion. The platform leverages Apache Cassandra (*mysimbdp-coredms*) to provide distributed data storage with high availability and redundancy. The data ingestion service (*mysimbdp-dataingest*) efficiently processes and stores LiDAR point cloud data into the core database, demonstrating the platform's ability to handle structured spatial data.

Key achievements of this project include:

- Designing a robust schema and partitioning strategy that ensures efficient querying and even data distribution across nodes.
- Implementing a fault-tolerant Cassandra cluster with a replication factor of 2, deployed across multiple availability zones to prevent single points of failure.
- Evaluating the platform's performance under varying conditions, including concurrency levels, consistency settings, and cluster sizes, to identify bottlenecks and optimize deployment.

The results indicate that the platform scales effectively with increased nodes and maintains consistent performance across different consistency levels. However, challenges such as low ingestion speeds due to potential networking limitations or script inefficiencies were identified, highlighting areas for improvement.

5.2 Future Enhancements

To further enhance the capabilities of *mysimbdp*, the following improvements are proposed:

- **Service and Data Discovery:** Integrate a service discovery mechanism (e.g., Consul or ZooKeeper) to map tenants to their dedicated *mysimbdp-coredms* instances dynamically. This will simplify multi-tenant management and improve scalability.
- **Real-Time Data Ingestion:** Extend the ingestion pipeline to support real-time streaming data sources, enabling the platform to handle use cases such as IoT analytics and financial data processing.
- **Hot/Cold Data Management:** Introduce automated policies to classify and move data between hot and cold storage tiers based on access frequency. For example, frequently accessed data can remain in high-performance storage, while infrequently accessed data can be archived to cost-effective, long-term storage.
- **Data Lineage Tracking:** Implement metadata tracking to record the origin, transformations, and destination of ingested data. This feature will provide tenants with valuable insights into the lifecycle of their data.
- **Platform-as-a-Service (PaaS):** Develop a new component, *mysimbdp-daas*, to act as an intermediary for external data producers and consumers. This will abstract the underlying database operations and provide APIs for seamless data interaction.
- **Performance Optimization:** Investigate and address the root causes of low ingestion speeds, such as networking limitations or inefficiencies in the Python ingestion script. Potential solutions include optimizing batch processing, reducing logging overhead, and leveraging faster network configurations.

These enhancements will not only address current limitations but also expand the platform's functionality, making it more versatile and capable of meeting the demands of modern big data applications.

6 REPRODUCTION

The project and all related files can be found on GitHub. The Readme.md on the GitHub page has instructions on how to replicate this. The Virtual machines have to be recreated using infrastructure management as I do not have the budget to keep them running all of the time. The datasources can be found in the references.

REFERENCES

- **GitHub**: GitHub page with all the project files and instructions. https://github.com/samuliohman/big_data_platform
- **Lidar data source**: https://archive.nyu.edu/handle/2451/38660
- **Apache Cassandra Documentation**: Comprehensive documentation for Apache Cassandra, covering installation, configuration, and usage. https://cassandra.apache.org/doc/latest/
- **Google Cloud Platform Documentation**: Official documentation for Google Cloud services, including Compute Engine and networking. https://cloud.google.com/docs
- **Terraform Documentation**: Official guide for Terraform, a tool for building, changing, and versioning infrastructure. https://developer.hashicorp.com/terraform/docs
- **ChatGpt**: ChatGpt was used to help debug issues with infrastructure deployment and Cassandra's configuration. Also, ChatGpt helped with spelling and structuring the documentation. https://chatgpt.com/