Real-Time Earthquake Monitoring and Prediction System

Mikołaj Malec 298828

1. Executive Summary

High-Level Description

The application provides a global, scalable view of earthquake activity by aggregating and clustering seismic events into meaningful regional insights. Instead of overwhelming users with millions of individual records, it visualises regional patterns, trends, and highlights only the most significant earthquakes. Real-time and recent events are presented to support situational awareness without compromising analytical clarity. The platform enables analysts, seismologists, and emergency operations officers to understand seismic activity quickly, consistently, and at the appropriate level of detail. ML-based clustering transforms high-volume, noisy seismic data into stable analytical units that enable spatial comparison, temporal trend analysis, and contextual interpretation of new events.

Progress Since PM1:

- *Ingestion:*
  Developed Apache NiFi flows for data injection, transformation and normalisation.
- *Storage:*
  Configured HDFS for raw data storage and fault tolerance.
- *Data Warehouse:*
  Established Hive tables for structured querying of historical datasets.
- *Batch Processing:*
  Configured PySpark environment (PyCharm) for data cleaning and batch layer transformations.

2. Updates to PM1 Report

Change in ML task:

This system groups global earthquake events into spatial and temporal clusters to reveal meaningful regional patterns. This clustering supports trend analysis over time, helping users understand how seismic activity evolves. It also provides context for recent events, showing whether new earthquakes are part of ongoing regional activity or isolated occurrence. Possible models for this task include hierarchical clustering or density-based methods such as K-Means, GMM, BisectingKMeans.

There is no change in the target audience:

- Analyst: Interprets data to explain trends and implications.
- Seismologist: Studies seismic behaviour to advance scientific understanding.
- Emergency Operations Officer: Uses seismic signals to guide rapid response decisions.

There is a change in use cases:

Use Case 1: Understand Where Earthquake Activity Is Concentrated
The user needs to quickly understand which regions of the world are currently most seismically active. The user views the clustered global or regional map and reads regional summaries showing aggregated activity levels, intensity indicators and seismic activities trend directions.

Use Case 2: Understand How Seismic Activity Is Changing Over Time. The user needs to know whether earthquake activity in a specific region is increasing, decreasing, or stable. The user

reads regional analytics below the map, including time-series trends, activity metrics, and comparisons to previous periods.

Use Case 3: Understand What Is Happening Right Now
The user needs immediate awareness of newly occurring or recent earthquakes that may require attention. The user reviews the Current Events tab, reading a concise list of recent earthquakes with key details such as time, location, and magnitude.

Justification
Predicting earthquakes with ML is highly uncertain and unlikely to produce reliable results for users. By dropping the prediction task, we can instead focus on hierarchical clustering, which enhances the analytical part of the app and improves aggregation of seismic events into meaningful spatial and temporal patterns. This approach provides clear, actionable insights into where activity is concentrated, how it changes over time, and how new events relate to existing clusters, delivering scalable and interpretable seismic analysis.

Other
There are no changes in the data sources, systems architecture or technical assumptions.

3. Conceptual Data Model

The data model follows a multi-layered Medallion Architecture (Bronze, Silver, Gold) to ensure traceability, data quality, and analytical performance.

Layer 1: Raw Data (Bronze Layer)
This layer stores the raw, immutable data exactly as ingested from the source APIs. It serves as a historical archive and enables reprocessing if transformation logic changes. The tables are partitioned by date.

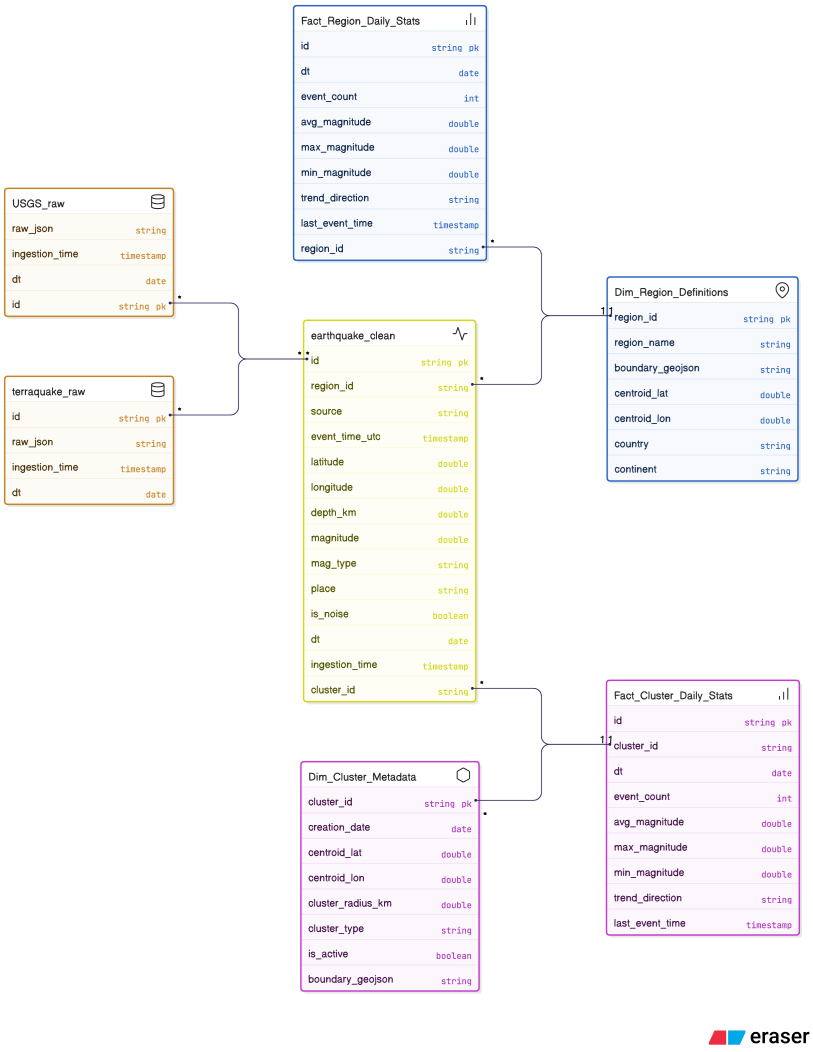
- **USGS_raw** - raw JSON responses from the USGS API.
- **terraquake_raw** - raw JSON responses from the Terraquake API.

Layer 2: Clean & Unified Data (Silver Layer)
This layer merges data from both sources, handles deduplication, and standardises schemas. It is the "Single Source of Truth" for individual seismic events. The table is partitioned by date.

- **earthquake_clean** - a unified, deduplicated table containing individual earthquake events.

Layer 3: Aggregated Data (Gold Layer)
The Gold Layer is the final destination for our data, formatted specifically for fast reporting and dashboards. It provides two different ways to analyze the earthquake data: by Fixed Geographic Regions and by Dynamic ML Clusters.

- **Regional Analysis View (Static)**
  - This view organises data by standard, pre-defined geographic areas
  - **Fact_Region_Daily_Stats** - aggregated counts and magnitudes for that specific region and date
  - **Dim_Region_Definitions** - contains the details that don't change, such as the Region ID, its Name, and its boundary coordinates
  - **Relationship**: The Fact table joins here via region_id to get the region's name.
- **Cluster Analysis View (Dynamic/ML)**
  - This view organises data based on "hotspots" detected by Machine Learning. Unlike fixed regions, these clusters can appear, move, or change over time.
  - **Fact_Cluster_Daily_Stats** - aggregated counts and magnitudes for that specific cluster generated by the Machine Learning and date
  - **Dim_Cluster_Metadata** - describes the cluster's characteristics, including its ID, centroid point (centroid_lat, centroid_lon), and when it was first detected (creation_date)
  - **Relationship**: The Fact table joins here via cluster_id to get the cluster's location details.



4. Data Set Descriptions

Source / API Name	Description	Base Data Volume and Velocity	Key Attributes	Time period available	Format / Structure	Data Quality
TerraQuake API (api.terraquakeapi.com)	Receives near real-time updates for seismic events from monitoring networks around the globe (e.g., INGV).	~50 events per day(Near real-time)	event_id, timestamp, latitude, longitude, depth_km, magnitude, region, alert_level	1985 – Present	JSON (REST API)	Medium - HighRelies on authoritative sources; requires unification with USGS.

Source / API Name	Description	Base Data Volume and Velocity	Key Attributes	Time period available	Format / Structure	Data Quality
USGS Earthquake Hazards API (earthquake.usgs.gov)	Provides global coverage and monitoring of seismic activity; considered the primary scientific authority.	~300 events per day(Near real-time)	id, time, latitude, longitude, depth, mag, place, type, status	1568 – Present	GeoJSON (REST API)	High - Government-standard scientific accuracy and consistency.
Region Definitions File (Local / HDFS)	Static reference dataset defining the geographical boundaries for fixed seismic regions (e.g., "Circum-Pacific").	~16 records(Static / Rare updates)	region_id, region_name, lat_min, lat_max, lon_min, lon_max	N/A (Static)	CSV (Comma Separated)	High - Manually curated and verified definitions.
Earthquake Clean (Internal Silver Layer)	The "Single Source of Truth" repository. Merged and deduplicated events from both USGS and TerraQuake.	~350 events per day(Daily append)	event_id, time_utc, magnitude, mag_type, coordinates, source_system	2025 – Present (Project Inception)	Apache Parquet (Partitioned by Date)	Very High - Deduplicated, standardised schema and type-cast.
Aggregated Statistics for Region / Cluster (Internal Gold Layer)	Pre-computed daily summaries for both Fixed Regions and Dynamic Clusters. Used for dashboards.	~20-50 rows per day (Daily batch)	dt, region_id, cluster_id, total_events, max_magnitude, sum_magnitude, max_depth, sum_depth, min_depth	Aggregated Daily	Apache Parquet (Optimized for OLAP)	High - Validated aggregations ready for visualization.
Cluster Metadata (Internal ML Output)	Dynamic definitions of seismic clusters generated by the clustering machine learning model.	Variable (depends on activity) (Updated per batch)	cluster_id, centroid_lat, centroid_lon, density, creation_date	Dynamic History	Apache Parquet	Variable - Dependent on ML model convergence and noise handling.

5. Data Acquisition and Transformation

Acquisition Strategy

- Data Sources: The system ingests earthquake data from two primary public providers: USGS (U.S. Geological Survey) and Terraquake. Both sources are queried for recent activity (last 5 minutes and last 10 events respectively) to ensure data freshness.
- Apache NiFi: serves as the primary orchestration engine for data ingestion. It handles the scheduled HTTP requests to the public APIs.
- Ingestion Pattern: Data is pulled via an incremental batch strategy every 5 minutes.

Data Transformation Steps

- Processing Engine: Transformation is performed using PySpark to handle large-scale data cleaning and unification.
- Unification Logic:
- Schema Mapping: Diverse JSON structures from USGS and Terraquake are mapped to a Unified Schema, standardising fields such as magnitude, time (UTC), and spatial coordinates (longitude, latitude, depth).
- Deduplication: A windowing function (Window.partitionBy("id")) is applied to identify and keep only the latest record for any given event ID, preventing duplicate entries from overlapping API pulls.
- Data Enrichment: Raw timestamp strings are cast to BigInt/Long types for temporal consistency, and partition keys (dt) are generated for storage optimisation.

Storage Strategy

- Storage Technology: Data is stored locally on the HDFS (Hadoop Distributed File System) and managed via Apache Hive tables.
- Data Formats: The final transformed data is stored in the Apache Parquet format. Parquet is utilised for its superior compression and columnar storage performance.
- The storage is partitioned by date (dt), allowing for efficient querying.
- The system uses coalesce(1) during the write phase to prevent "small file problems" on HDFS, ensuring each daily partition remains compact and performant.
- Write Mode: Spark uses dynamic partition overwrite mode to update specific partitions without affecting the entire dataset.

6. System Architecture

The proposed system follows a Lambda Architecture, integrating streaming and batch data processing components into a cohesive data pipeline.

Input Layer (Preprocessing)

Technologies: Apache NiFi Streams, Apache Kafka
Function: Collects real-time seismic data from APIs, performs initial validation and enrichment, and forwards it to processing layers via Kafka topics.
Output: Cleaned and standardised data ready for further processing.

Speed Layer

Technologies: Apache Spark Structured Streaming, Apache Cassandra
Function: Performs real-time computations on streaming data (e.g., detecting recent seismic activity, calculating moving averages, and updating dashboards).
Output: Low-latency updates stored in Cassandra for immediate dashboard visualisation.

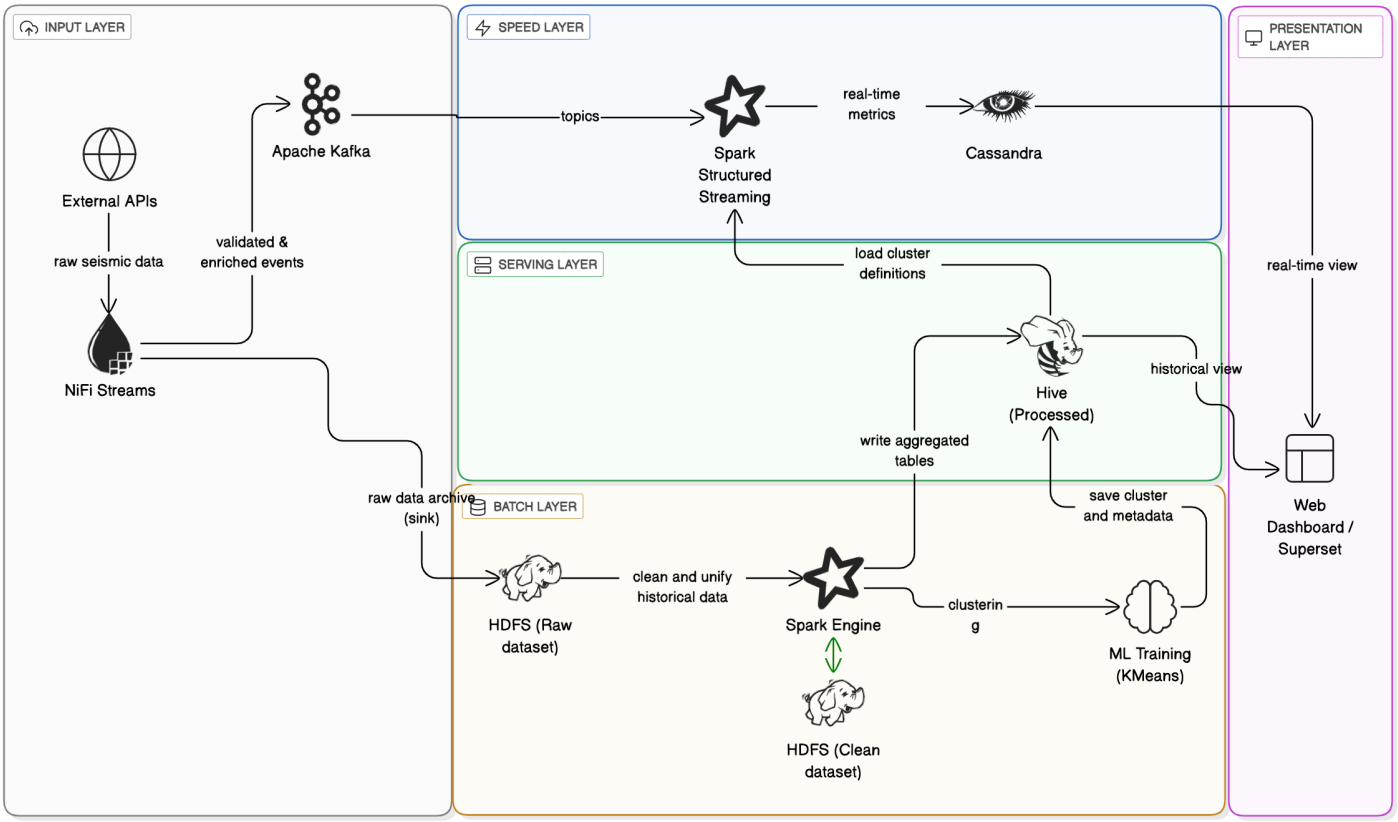
Batch Layer

Technologies: Apache Spark, PySpark, HDFS, Hive
Function: Handles large-scale transformations, aggregation of historical data, and machine learning model training for predictive analytics.

Output: Processed datasets and prediction models stored in Hive for long-term access.

Serving Layer
Technologies: Hive / Cassandra (depending on aggregation scale)
Function: Integrates data from both Speed and Batch Layers to provide a unified, queryable interface for the presentation layer.
Output: Aggregated datasets accessible via APIs or SQL queries.

Presentation Layer
Technology: Apache Superset
Function: Displays real-time and historical earthquake data, clustering insights, and visual analytics through an interactive user interface.
Output: Actionable insights for analysts, seismologists, and emergency officers.



eraser

7. Analytical Approach

Machine Learning Task: Unsupervised Seismic Clustering
To identify persistent seismic zones within the unified data, the system employs an unsupervised machine learning approach applied to the Silver Layer, selecting the optimal model from three PySpark-native candidates: Standard K-Means, Gaussian Mixture Models (GMM), or Bisecting K-Means. The final chosen algorithm will be responsible for partitioning historical events into distinct geographical clusters, enabling the system to generate metadata for active faults and track their intensity evolution over time.

Implementation Strategy:
Daily Batch Retraining
To ensure the system captures rapidly evolving seismic sequences (e.g., aftershocks or new swarms). The clustering algorithm retrains daily on the updated dataset to recalculate density reachability. The output of this task is a refreshed Dim_Cluster_Metadata table. This serves as the updated "map" of global risk zones for the next day of operations.
Stream Layer
As new events arrive via Kafka, the system performs a Spatial Lookup against the latest cluster definitions generated by the previous batch job.
Logic:
- IF the new event falls within distance of an existing cluster, it is immediately tagged with that cluster_id, updating the "Current Cluster Activity" metrics
- IF the event is an outlier, it is tagged as "Noise" until the next daily batch run potentially identifies it as the start of a new cluster

Result Presentation
1. Real-Time Event Feed
   This widget serves as the immediate alert system for the dashboard. It displays a live, auto-updating list of the last 10 detected seismic events.
2. Seismic Activity Map (Layered)
   The central visual component is an interactive, layered map. It combines historical context with current operational data by overlaying today's earthquake events (represented as scatter points) on top of the established ML-detected seismic clusters (represented as polygons).
3. Regional Metric Trends
   To support long-term analysis, this line chart visualises the evolution of seismic activity within fixed geographical regions. It plots key daily metrics for all regions.
4. Cluster Drill-Down
   When a user selects a specific cluster, this widget displays key daily metrics for this cluster