



Fault Tolerant Integration of Apache Beam With Relational Database

Aug 4th 2021

Intro: Speakers

Piaw Na

Senior Staff Software Engineer

- Infrastructure
- Niantic Lightship



Savitha Jayasankar

Software Engineer

- Infrastructure
- Distributed data processing

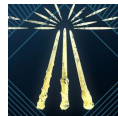


Intro: Niantic Inc.

Games + Platform

Games

Deliver best-in-class AR experiences



Platform

Deliver best-in-class AR developer platform

The screenshot shows the Niantic ARDK developer platform website. On the left is a navigation menu with the Niantic logo at the top, followed by 'Overview', 'Profile', 'Download' (highlighted in orange), 'Features & roadmap', 'FAQs', and 'Documentation'. The main content area is titled 'Download' and contains the following text: 'Niantic's Augmented Reality Developer Kit 'ARDK' is made to work with iOS, Android, and Unity! [Learn about ARDK key features.](#)' Below this is a red button labeled 'Download ARDK (Alpha)' with 'Version 0.6.0' underneath. A note states: 'Important: please ensure your development environment meets the [minimum system requirements](#). An API license key is required for using certain ARDK features.' A section titled 'Before getting started, we recommend checking out the following:' lists two items: '• [Instructions for adding ARDK to your Unity project](#)' and '• [Download ARDK-Getting Started](#)'. At the bottom, it says: 'ARDK's Getting Started Unity Project provides a basic example for getting acquainted with ARDK code and core components. Set up [technical requirements.](#)'

Outline

- Motivation
- Implementation Attempts
- Successful Implementation
- Results
- Lessons Learned

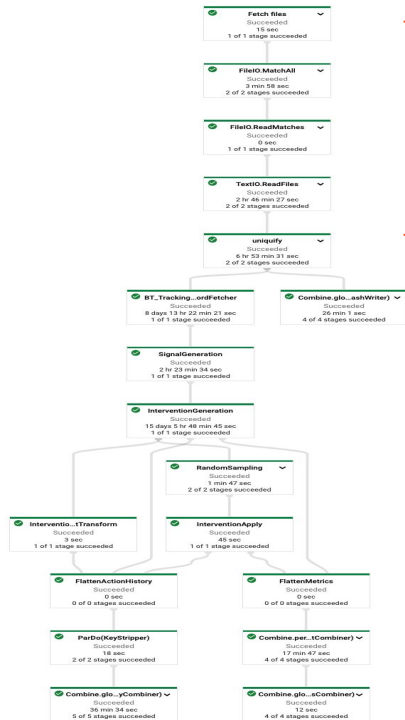
Motivation

- Provide metrics as the CoreInfra Dataflow Pipeline detects malicious explorers and visualize through Grafana.

Simple Right?

- CoreInfra runs as a dataflow periodically
 - Dataflows read from GCP BigTable and write to GCP Spanner.
- Dataflow time \neq Event time \neq Client Time
- Prometheus does not allow backdating of metrics
- Keeping Production cost low.

Core Infra DataFlow Pipeline



Input/Extraction Layer

- Flat files
- Hbase Tables

Transformation Layer

- Data manipulations based on conditions

Output / Loading Layer

- RDMS
- GCP Spanner

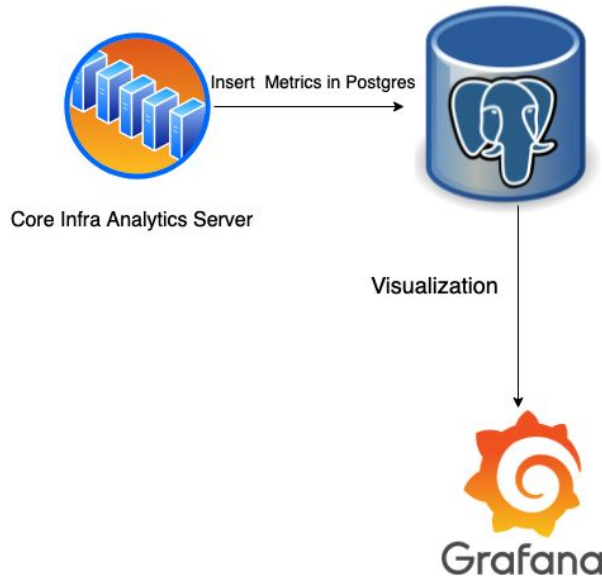
Statistics about Pipeline

- Millions of active Niantic Explorers per hour
- Billions of trackable Niantic Explorer components
- 10 GCP Spanner nodes for Niantic Explorer component history
- 1 Postgres node for Core Infra metrics and action history
- 1 dataflow per hour per game
 - 200 vCPUs (based on game volume)
 - 16 threads per vCPU
 - 30-45 minutes for dataflow completion.

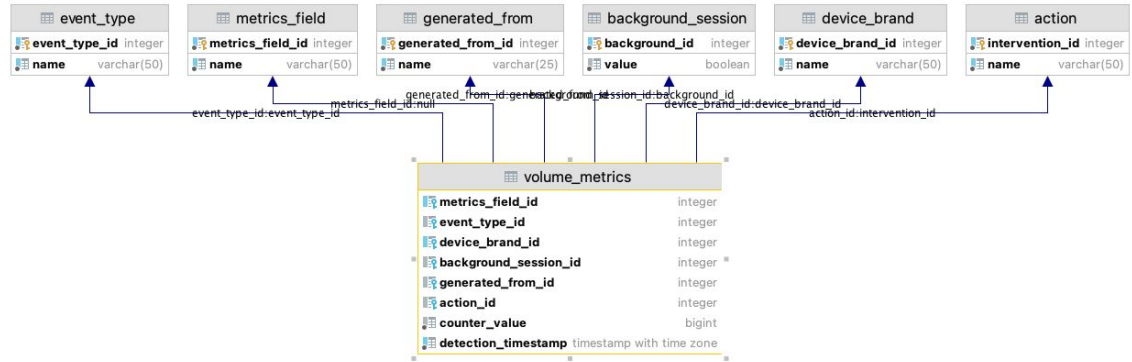
Attempted Implementation

- **Dataflow with Prometheus Implementation : Pull Metrics System**
- **Dataflow with Prometheus with PushGateway : Push Metrics System**
 - Remote Storage in BigQuery
 - Remote Storage in GCP Spanner

CoreInfra Metrics Architecture



- Replaced Prometheus with Postgres DB for metrics Storage
- Customised metrics
- Datetime can be customised to client time or processing time based on the metrics requirement



Dataflow and RDBMs

Naive implementations didn't work:

- Cloud SQL Postgres: Rejects Connection Requests after 150 connections
- JDBCIO : Beam runner writes multiple times for fault tolerance ; risk of duplicates
- Postgres: More connections → high CPU utilization
- Even scaling up (max CPU + max memory) Postgres instance didn't alleviate the above problems

Insight : Computation through dataflow

- Use *Combiners* to combine per-metric information
- Use *.withfanout/.withHotKeyFanout* to initiate the combine function without waiting for all input to come in*
- Each worker then batch inserts to Postgres by using Prepared Insert Statements
- If the DB connection request failed, the connection request would be tried again.

*fanout input value need to be evaluated on a trial basis

Dataflow Implementation using Java

```
metricsMergedCollectionWithFlatten  
    .apply(ParDo.of(new MetricsDBWriter()))
```

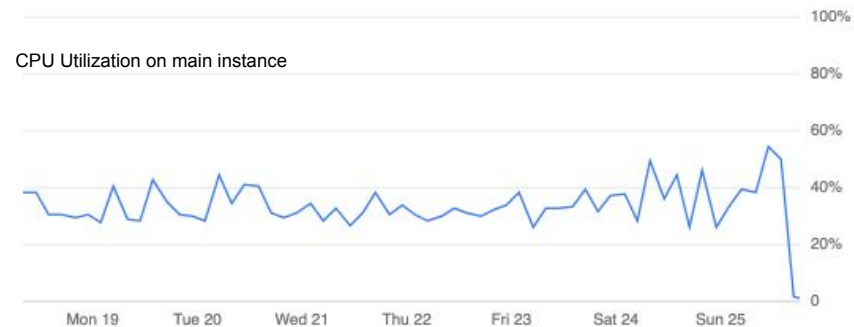
```
metricsMergedCollectionWithFlatten  
    .apply("EventCombiner",  
        Combine.<String, Event, Event>perKey(new  
EventCombiner()).withHotKeyFanout(2))  
    .apply("MetricsCombiner",  
        Combine.globally(new  
MetricsCombiner(flow.getConnection(), gameServer))  
    .withFanout(2));
```

Results

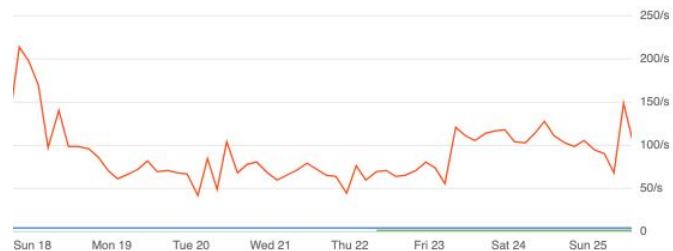
- Scaled down Postgres Instance
- Write performance no longer bottleneck
 - batch inserts are faster than GCP BigQuery streaming inserts
- Saved Compute Cost.
- Improved latency over BigQuery.
- Overall Dataflow elapsed time was improved,
 - replaced other BigQuery usages with Postgres tables
- Zero code changes for porting to other cloud providers.
- Fixed cost of analytics related queries.

Typical dataflow numbers

History_records per 30 mins	6,556,245
Batch-insert-succeeded per 30 mins	11,331
inserts/second	1,806



Transactions/sec on main instance

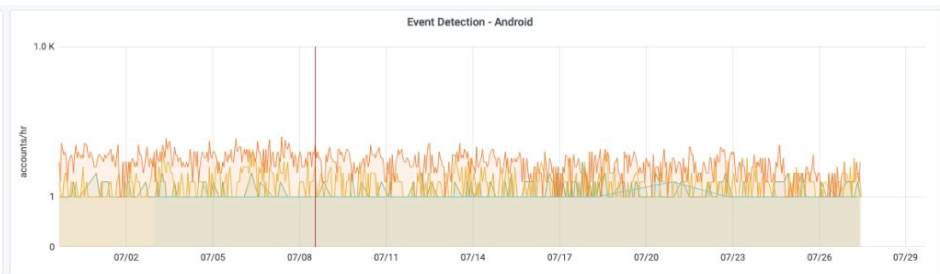
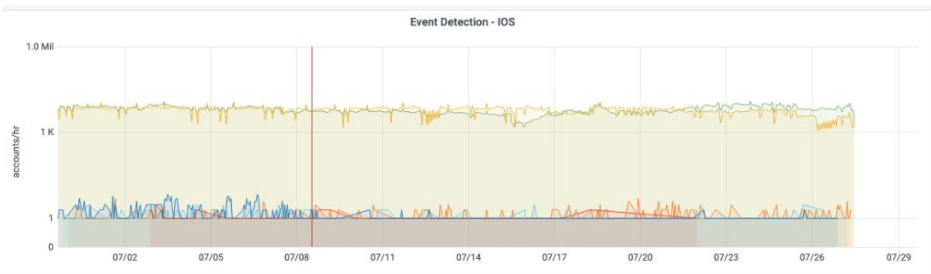


Refinements

- **Ensure proper use of RDBMS techniques like implementing data normalization**
 - Started with jsonb column into RDBMS columns
 - Index on new columns for improved query performance
- **Harden against GCP CloudSQL outages**
- **Setup Monitoring against Postgres outages**
- **Scale down over-provisioned Postgres instance**

Lessons learned

- Postgres can perform 100k appends/second on SSD.
- Don't be afraid of Postgres/RDBM
 - Dataflow can write to Postgres at scale with proper organization of dataflow stages
 - Unlike BigQuery/Spanner/Bigtable, it's the same for all cloud platforms
 - Postgres is cheap!
- Grafana can be decoupled from Prometheus
 - Grafana can alert without prometheus in the picture



Thank you. Questions?

Niantic is Hiring: <https://careers.nianticlabs.com/openings/>

Reach out to us on

Piaw Na: pna@nianticlabs.com

Savitha Jayasankar: sjayasankar@nianticlabs.com