Best Practices to Develop an Enterprise Datahub to Collect and Analyze 1TB/day Data from a Lot of Services with Apache Kafka and Google Cloud Platform in Production

Kenji Hayashida  
Recruit Lifestyle Co., Ltd.  
Data Engineering Group

Toru Sasaki  
NTT DATA Corporation  
Platform Engineering Department  
System Engineering Headquarters
1 Agenda

- Introduction
- The platform design developed in this project
- Technical Topics (Platform)
- Technical Topics (Interface)
- Case Studies and Takeaways
2 Who Am I?

Kenji Hayashida

• software engineer at Recruit Group
• working on advertising technology and data pipeline
• like programming competitions
• writer of an introductory data science book
3 Who Am I?

Toru Sasaki
• Software and Infrastructure Engineer in NTT DATA, Japan.
• Specializes in open-source distributed computing software.
  – e.g. Apache Hadoop, Apache Spark, Apache Kafka

Company: NTT DATA Corporation
• One of the largest IT solution providers in Japan.

Team: OSS Professional Services
• Provides technical support related to open-source software
• One of the most experienced Hadoop teams in Japan
4 Relationship between Recruit Lifestyle and NTT DATA in this project

- Recruit Lifestyle is the project owner.

- NTT DATA supports designing the platform utilizing distributed messaging system for this project.
INTRODUCTION OF OUR PROJECT
About Recruit Group

Recruit Group runs many kinds of services:

- **Gourmet**
- **HR**
- **Housing**
- **Wedding**
- **Car**
- **Online Shopping**
- **Beauty**
- **Education**
- **Travel**
About Recruit Group

We are advancing into overseas markets

- in human resource business by 2020
- in human resource and promotional media business by 2030
Importance of Data Analysis

We have various life activity data **from the cradle to the grave**

Using these data and making insightful analysis is crucial to our business

- improve **UI/UX** of our services
- make **recommendations** for our customers
- **predict the sales** for our clients
- make **ad campaigns** for our clients
- etc
Existing problems

Diversity of organizations/services created a chaotic situation

- Complex data pipelines
  - many ETL batch jobs
  - an upstream application has an effect on downstream applications

- It is not clear for data scientists
  - what kind of data are available?
  - where is it stored?
  - how to access it?

- Data scientists have to spend a lot of time to
  - parse/format/validate the data
Our Project’s Mission

Make data collaboration easy and foster a data-driven culture.

**vision**

**Engineers can easily**
- produce data
- change schemas

**Data scientists can easily**
- search data
- start using data

**goals**

**Pipeline is**
- easy to install
- robust to schema changes

**Data have**
- unified serialization format
- schema
- context information
Difficulties to build data pipeline

There are many challenges due to our service diversities

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data diversity</strong></td>
<td><strong>Pipeline should be</strong></td>
</tr>
<tr>
<td>- high volume (1TB/day)</td>
<td>- scalable</td>
</tr>
<tr>
<td>- ever-changing schema</td>
<td>- robust to schema changes</td>
</tr>
<tr>
<td><strong>System diversity</strong></td>
<td><strong>Pipeline should be</strong></td>
</tr>
<tr>
<td>- different infrastructures</td>
<td>- extensible</td>
</tr>
<tr>
<td>- different languages</td>
<td>- language-agnostic</td>
</tr>
<tr>
<td><strong>Analysis diversity</strong></td>
<td><strong>Pipeline should be</strong></td>
</tr>
<tr>
<td>- batch analysis</td>
<td>- connectable to many tools</td>
</tr>
<tr>
<td>- streaming analysis</td>
<td></td>
</tr>
</tbody>
</table>
OUR APPROACH
Our Approach: “Datahub Architecture”

- “Datahub architecture”
  - Collect **ALL** the data in one place (this is called Datahub)
  - Get **ALL** the necessary data from the Datahub

- We developed a data pipeline based on this architecture
An Overview of Our Data Pipeline

- We developed a data pipeline as below
15 Mapping challenges to our Data Pipeline

- Our data pipeline satisfies our challenges and goals

- Easy to install
- Robust to schema change
- Have schema
- Scalable
- Robust to schema change
- Extensible
- Language-agnostic
- Connectable to many tools
- Unified serialization format
- Schemas
An Overview of Our Data Pipeline

- We considered this in 2 categories, Interface and Platform.
PLATFORM OVERVIEW AND TOPICS
An Overview of Our Platform Design

- Datahub Platform was developed on Google Cloud Platform (GCP).
- App logs will be sent from various environments (AWS, On-Premises, GCP).

![Diagram of platform design]

Recruit service platforms

Google Cloud Platform (GCP)

Recruit service platforms

AWS (Multi regions)

On-Premises

GCP

Apps

Local Kafka cluster

VPN or VPC-Peering

TCP-LB

Kafka

REST Proxy

Kafka cluster

Message format

Schema Registry

Google Compute Engine (VMs)

Google BigQuery

Google Dataflow

Google Pub/Sub

Internet

NGINX (Auth)
Why We Chose Kafka and Kafka Ecosystem as a Datahub Component? (1/2)

- Apache Kafka
  - Handles a lot of messages (around 1TB/day)
    - Kafka is high-throughput messaging system
  - Avoids cloud lock-in
    - Easy to migrate to other environments from GCP
    - Other environments: AWS, On-Premises etc.

- Kafka REST Proxy
  - Programming language-agnostic
    - Most programming languages support the easier HTTP(s) protocol compared to Kafka specific protocol

Component of Kafka ecosystem to provide Restful API interface for Kafka (included in Confluent Platform)

open-source distributed messaging system
• Schema Registry
  – Enables Schema Evolution considering schema compatibility
    • Schemas might be changed by enhanced apps
    • Based on the function provided by Apache Avro

• Apache Avro
  – Handles messages which has schema structure
    • Most messages handled by our platform have schema structure
  – Supported by Schema Registry natively
Best Practices Related to Datahub Platform

We considered some technical points during designing our platform.

This session explains the following 2 issues and best practices.

1. A Network architecture to be connected from a lot of independent services

2. Enable to migrate Kafka Cluster without stopping
22

Topic 1: A Network Architecture to be connected from a lot of independent services

• Motivation
  – Enable to collect app logs from **ALL** apps in Recruit Group

• Problems
  – Applications that send logs are running on various platforms
    • e.g. AWS (multi region), GCP, On-premises (private cloud)
  – Platforms with duplicated network addresses can’t connect to the Datahub platform directly

• Best Practice
  – Enable to connect from all platforms regardless of network addresses by designing suitable network architecture
Network Architecture to Be Connected from Various Platforms (Overview)

Platforms which application running on

- Peering to the same bastion VPC if network addresses are non-overlapping
- Have non-overlapped network address
- VPC Peering or VPN
- Peering to different bastion VPCs if network addresses are the overlapping
- Have overlapped network address
- VPC Peering or VPN

Bastion VPC 1
- subnet
- NIC send
- VPC Peering or VPN
- NOT Peering

Bastion VPC 2
- subnet
- NIC send
- VPC Peering or VPN
- NOT Peering

Datahub VPC
- subnet
- NAT instance
- NIC send
- NIC forward
- TCP-LB
- Kafka (REST Proxy / Brokers)

Create a new bastion VPC if platform has overlapping network address

Peering to the same bastion VPC if network addresses are non-overlapping

Peering to different bastion VPCs if network addresses are the overlapping

Platform has overlapping network address

Application running on

Application running on

Application running on

Application running on
Platforms which have already been running need not to change anything to connect this Datahub platform.

- Most apps which send logs are already running in production, making it difficult to change those apps to connect to the Datahub.

This network design can scale to handle increase in number of users and platforms.

- Users and platforms which send messages to our Datahub will increase significantly in the future because Recruit Group’s business has been growing.
25 Topic 2: Enable to Replace Kafka Cluster without Stopping

• Motivation
  – Our Kafka cluster is running on VMs (using GCE) now
  – Plan to migrate to containers (K8s/GKE) in the future

• Problem
  – Can’t stop collecting logs during the replacement of the cluster

• Best Practice
  – Designed the platform to migration by only changing the TCP-LB configuration

Google Compute Engine
Google Container Engine
The Strategy to Replace Kafka Cluster

- Change TCP-LB configuration to switch to new Kafka cluster.
27 Why We Consider this Strategy as the Best?

• Log collection won’t be affected when migrating Kafka cluster
  – Suspending log collection affects apps which are sending logs
  – These applications must not be affected by migration since they are running in production

• This strategy can be extended to other environments, too
  – It is easy to migrate this Datahub platform to other environments such as AWS, on-premises, etc.
INTERFACE TOPICS
Logging Library

• Motivation
  – Make it easy for engineers to produce logs

• Problems
  – Logging should not have a performance impact on our services
  – Optimal configuration depends on the service
  – We should attach schema and context information to data

• Best Practice
  – User-friendly, highly-configurable logging libraries that deal with technical difficulties under the hood
Logging Library (Architecture)

Recruit service

- conf
- avsc
- app

logging library

- Config Loader
- PK Generator
- Normalizer
- Schema Manager
- Serializer
- Http Logger
- File Logger
- Schema Manager

Kafka (Rest Proxy)

FS Storage (Avro)

- low latency
- high throughput
We designed a very simple user interface.

```python
data_logger = DataLogger.get_logger("pv")
message = data_logger.create_message()
message.put("user_id", "Bob")
message.put("page_id", "P0001")
data_logger.write(message)
```

The library deals the following tasks under the hood.

- data serialization, primary key generation
- schema validation
- buffering, error handling, retry
User can change settings with configuration file.

<table>
<thead>
<tr>
<th>Configuration Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_http_log</td>
<td>use http logging or not</td>
</tr>
<tr>
<td>queue_size</td>
<td>maximum size of buffering queue</td>
</tr>
<tr>
<td>connection_timeout_millisec</td>
<td>connection timeout</td>
</tr>
<tr>
<td>max_messages_num_per_request</td>
<td>maximum number of records sent to rest proxy in a batch</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>enable_file_log</td>
<td>use file logging or not</td>
</tr>
<tr>
<td>local_log_root</td>
<td>root directory of file logging</td>
</tr>
<tr>
<td>local_log_rotate_interval_millisec</td>
<td>rotation interval of file</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
Logging Library (Primary Key)

Embed log context information into primary key

<table>
<thead>
<tr>
<th>Timestamp 42bit</th>
<th>Product Id 12bit</th>
<th>Application Id 6bit</th>
<th>Location Id 18bit</th>
<th>Pseudo Rand 12bit</th>
</tr>
</thead>
</table>

We use an original encoding to preserve the magnitude relation

\[ a > b \implies f(a) > f(b) \]

\[ a, b: \text{90-bit binary} \]
\[ f(a), f(b): \text{encoded 15 characters} \]

In addition to log de-duplication, primary key can be used for

- chronological ordering
- log origin tracking
• Motivation
  – Provide out-of-the-box analysis tools for data scientists

• Problems
  – We should handle high-volume data
  – Calling outside service for schema detection is slow
  – Data transformation might fail

• Best Practice
  – Dataflow jobs with cache mechanism and dead-letter queue
<table>
<thead>
<tr>
<th>Tool</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google BigQuery</td>
<td>- scalability</td>
</tr>
<tr>
<td></td>
<td>- serverless</td>
</tr>
<tr>
<td></td>
<td>- many users in the company</td>
</tr>
<tr>
<td>Google PubSub</td>
<td>- multi-language support</td>
</tr>
<tr>
<td>Google Dataflow</td>
<td>- compatible with BigQuery and PubSub</td>
</tr>
<tr>
<td></td>
<td>- auto scaling</td>
</tr>
<tr>
<td></td>
<td>- unified model for batch and streaming</td>
</tr>
</tbody>
</table>
ETL (architecture)

Confluent

Kafka broker

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>magic byte</td>
</tr>
<tr>
<td>1-4</td>
<td>schema id</td>
</tr>
<tr>
<td>5-</td>
<td>avro data</td>
</tr>
</tbody>
</table>

Schema Registry

Dataflow

Topic list

Topic Finder

create

da topic

another topic

yet another topic

Record Encoder

Kafka IO

refer

Schema Util

PubSub IO

Google PubSub
ETL (calling outside service)

We need **fast** communication with outside service

**SchemaRegistryClient** is provided by Confluent

creating **long-lived object** is a little tricky on Dataflow

- an object should be **serializable**
- when not serializable, use **lazy evaluation** with transient keyword
ETL (dead letter queue)

We should be prepared for exceptional situations

- invalid data
- network error
CASE STUDIES AND TAKEAWAYS
INTRODUCING TO PRODUCTION: HOT PEPPER GOURMET

Restaurant reservation service
- internet reservation: 71,720,000 people (total in 2017)
- available (*) restaurants: 47,494 (as of August 2018)

Producer
- On-premises
- Java

Consumer
- BigQuery

Log
- search keyword, search results, etc.
- about 400 million records so far

* You can make a reservation via internet directly from our service
INTRODUCING TO PRODUCTION: AIRMATE

Restaurant management service (accounting/shift/customer)

**Producer**
- Google App Engine
- Python

**Consumer**
- BigQuery

**Log**
- feature usage history, request latency, etc.
Google App Engine(*) has some limitations

**Limitations**

- HTTP requests should be processed within 60 seconds
- Usage of background threads are not allowed

**Problems**

- Our logging library heavily depends on background threads.

* Here we talk about standard environment with auto-scaling type.
INTRODUCING TO PRODUCTION: AIRMATE

Use App Engine task queue

- Task queues let application perform work outside of user request
- deferred library comes in handy for our case

```python
@app.route('/form')
def form():
    deferred.defer(log_wrapper.log_page_view, page="form")
    return render_template("form.html")

def log_page_view(page):
    data_logger = DataLogger.get_logger("pv")
    message = data_logger.create_message()
    message.put("page", page)
data_logger.write(message)
```
TAKEAWAYS

✓ Data science is crucial to many business fields

✓ You need an easy-to-use data pipeline to foster a data-driven culture

✓ There are many challenges to build a data pipeline in big organizations

✓ We designed and developed Data-hub architecture with Apache Kafka on Google Cloud Platform

✓ We resolved some technical challenges and shared them as “best-practices” and “know-hows”
Thank you for listening!

Any Questions?

We are hiring!

- Recruiting site
  - https://engineer.recruit-lifestyle.co.jp/recruiting/
- Tech Blog
  - https://engineer.recruit-lifestyle.co.jp/techblog/

許可を求めるな、謝罪せよ。従え、心の声に。

It's easier to ask forgiveness than it is to get permission. Development follows your heart.