10 Things to Consider When Using Apache Kafka:
Utilization Points of Apache Kafka Obtained From IoT Use Case

May 16, 2017
NTT DATA Corporation
Naoto Umemori, Yuji Hagiwara
1. Project outlines

2. Tips and pitfalls from IoT use case:
   • Tunes Performance
   • Deals with unusual Operations
   • Availability pitfalls

3. Summary
Project Outlines
Who are we?

• Naoto Umemori : Platform Engineer
• Yuji Hagiwara : Platform Engineer

OSS professional headquarter in NTT Data Corp.

Our main target

• IoT (Connected Vehicle)
• Cloud technology (OpenStack, Docker,...)
• Automation of platforms
### The assumed volume for connected vehicle

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of Connections</td>
<td>&gt; 1 million</td>
</tr>
<tr>
<td>Simultaneous connections</td>
<td></td>
</tr>
<tr>
<td>Amount of Transactions</td>
<td>&gt; 100k TPS</td>
</tr>
<tr>
<td>Total Data rate</td>
<td>&gt; 100Gbps</td>
</tr>
</tbody>
</table>
Apache Kafka is a distributed streaming platform as having three key capabilities:

- Publish/Subscribe is similar to a message queue
- Store streams of records in a fault tolerant way
- Process streams of records

We have used Kafka as a Messaging System in our IoT platform.
Overview of Our IoT Platform

Key Architecture: Separation of Stream and Batch processing unit

- Devices
  - Sensors
  - Mobile phones
  - Servers
  - NW devices
  - Auto mobile

- Connection & Collection

- IoT Platform
  - Accumulation & Conversion
    - Stream Proc. unit
      - Data stores for Stream
    - Batch Proc. unit
      - Data stores for Batch

- Utilize

- Analysis
  - Multiple Data stores for Analysis

- Monitoring & Visualization

- Distribution

- Applications
  - Biz Systems
    - Inventory info.
  - Map info.
  - Traffic info.
  - User info.
  - ...
Architecture of Our IoT Platform

Collection

Gateway (Kafka Producer)

Message Broker

IoT Platform

Accumulation & Conversion

Stream process unit

Stream processing

Apache Storm

Stream Data stores

Batch process unit

Data Buffering

Spark

Spark Streaming

Archive Data stores

Batch Proc.

Temporary Data stores

Analysis

Real-time Analysis API

Analysis ETL

Analysis API

Analysis Data stores

Distribution

Device info.

Monitoring & Visualization

elasticsearch

etl

Grafana

NTT DATA
Tips and pitfalls from IoT use case
Tips and pitfalls from IoT use case

**Tunes Performance**
- Disk I/O of Kafka Broker
- Concurrency of Kafka Producer
- The number of Partitions
- Async/Sync Bridge

**Deals with unusual Operations**
- Offset Monitoring
- Purging Kafka Topics
- Slow Pub/Sub Log

**Availability pitfalls**
- Undesired RAID Group
- Unstable Kafka Topics
- A huge number of Partitions makes Cluster unhealthy
Tips and pitfalls from IoT use case:
Tunes Performance
1. Disk I/O of Kafka Broker

Issues:
- Data writing speed slow downed and the throughput degradation occurred
- The amount of data is too large, exceeding the upper limit of Buffer Cache
- When cache flush occurred, data could not be written and IOPS decreased

Solutions/Actions:
- We restricted the flow rate to the throughput which becomes IOPS or less considering Cache flush

Result:
- Kafka cluster is working stably

Practice:
- We need to take account impact of buffer cache flush in Kafka Cluster sizing since Kafka mainly requires Disk I/O resource
2. Concurrency of Kafka Producer (kafka-client 0.8.2.2)

Issues:
- Throughput saturation
  - Kafka Broker seems not to be CPU, MEM, NW and Disk I/O saturation
  - Kafka Producer seems not to be CPU and NW I/O saturation
- Unknown where the bottle neck is

Solutions/Actions:
- Watch Java thread dump (jstack) to find out what happening is

Result:
- Found the bottle neck - The usage of a CPU core reached to 100%
- Sender Java thread of Kafka Producer is busy

Practice:
- Multiple process may be good idea
2. Concurrency of Kafka Producer (kafka-client 0.8.2.2)

Issues:

- **Sender is a single Java thread**

```
Data Source(s) -> Worker (User’s AP) -> send() -> append() -> Batch -> Batch -> Batch -> drain() -> Request -> Kafka Broker(s)
```

Kafka Producer (This is Gateway in our case)
3. The number of Partitions

Issues:
- There may be not right answer to choice of the number of partitions
- If the number of partitions is:
  - too few: The performance of Consumer can’t be scale
  - too many: The latency from producing to consuming increase

Solutions/Actions:
- We have decided it by multi-times of the number of physical disks heuristically

Result:
- Two times to four times of the number of physical disks looks good

Practice:
- Depending on the physical performance of the disk and the number of messages

4. Async/Sync Bridge

Issues:
• When connecting Kafka and something like “Things” of IoT to collect massive data from “Things”, each communication mode may not match:
  • Kafka: Asynchronous mode
  • “Things”: Synchronous mode

Solutions/Actions:
• The approach so that Kafka Producer should be more high performance is below:
  • Few Kafka Producer accept the connections from many “Things”
  • “Things” supports asynchronous communication

Result:
• We could get the producer to be more performance until the limit of CPU

Practice:
• Need to take account the communication mode
4. Async/Sync Bridge: An example of a sequence diagram when the number of devices is 1

Throughput of Kafka Producer:
\[ T1 = \frac{1}{t1} \text{ [TPS]} \]
4. Async/Sync Bridge: An example of a sequence diagram when the number of devices is 3

Throughput of Kafka Producer:
\[ T2 = \frac{3}{t2} \text{ [TPS]} \]
\( (T2 > T1, \text{ normally}) \)
Tips and pitfalls from IoT use case:
Deals with unusual Operations
5. Offset Monitoring

Issues:
• We want to monitor the difference between Produce Offset and Fetch Offset.
  • For preventing a performance problem caused by Caching out.

Solutions/Actions:
• Visualize Offsets
  • Get Produce Offset
    # bin/kafka-topics.sh --describe --topic <topic>
  • Get Fetch Offset (for storm-kafka 1.0.1)
    # zookeeper-cli get /<zkRoot>/<id>/<topic>/<partition>

Result:
• We could get a kafka cluster sizing for stabilizing.

Practice:
• Understand your workload by monitoring performance metrics.
5. Example of Offset Monitoring by Grafana

We are using the visualizer as Grafana.

The difference between Produce Offset and Fetch Offset
6. Purging Kafka Topics

**Issues:**
- Posterior to 0.8.2, Kafka support Topic (logical) deletion.
  - but we cannot create a topic with a same name: inconvenience for regression test.
- If we delete kafka segment files, the topic *wouldn’t be* deleted.

**Solutions/Actions:**
- We operate this procedure in order:
  - `# bin/kafka-topic.sh --delete -topic <topic>`  (Logical Deletion)
  - `# bin/kafka-server-stop.sh`  (Stop the server)
  - `# rm <directory of kafka log>`  (Delete segment files)
  - `# sysctl -w vm.drop_caches=3`  (Drop caches)
  - `# bin/kafka-server-start.sh`  (Start the server)

**Result:**
- The topic was deleted successfully.

**Practice:**
- Define operation procedure strictly. Observe the order of instructions.
7. Slow Pub/Sub Log

Issues:
• We want to identify the performance bottleneck for analyzing a performance problem related to Kafka.
• We can use Resource Monitoring / Broker metrics.
  • Good point: Overview of Kafka load (it is busy or not.)
  • Bad point: No detail for performance of each requests.

Solutions/Actions:
• We measure processing time on Producing/Consuming by our own application implementation. (it is similar to “Slowlog” like RDB)

Result:
• We can identify the slow process and improve it.

Practice:
• Necessary of Slowlog Feature.
• You can implement your own measurement.
Tips and pitfalls from IoT use case:

Availability pitfalls
8. Undesired RAID Group

Issues:
• We want to use Kafka without RAID. *
• In generally, Kafka has deployed to servers with multi HDD connected to the RAID controller.
• Some cheap RAID controller don’t support 1 Logical Volume / Physical Volume.

Solutions/Actions:
• Using RAID-0, Unfortunately.

Result:
• We cannot configure the Kafka cluster without RAID.

Practice:
• Don’t buy cheap RAID controller. Check and compare specifications.

* The choice (Using RAID or not) has tradeoffs (https://kafka.apache.org/documentation/#diskandfs) but In IoT, No reason to use Kafka with RAID if Each devices send the same amount of data.
9. Unstable Kafka Topics

Issues:
• A part of topics wouldn’t subscribed by the Storm application.
  • when we create topics after launching the Storm application using KafkaSpout (storm-kafka 1.0.1) to subscribe topics with wildcards.

Solutions/Actions:
• Create topics before launching the application.
• Check these topics are created: # bin/kafka-topics.sh --describe

Result:
• The Storm Application subscribed all of topics successfully.

Practice:
• Creating topics are very heavy operation.
• Confirm that the operation succeeded after execution.
10. A huge number of Partitions makes Cluster unhealthy

Issues:
• Kafka Brokers sometimes drop from the cluster during runtime.
  • When we created 1,000 topics (96 partitions/topic), and the Storm application with KafkaSpout (storm-kafka 1.0.1) consume these topics.
  • Kafka Brokers weren’t crashed. Only Zookeepers put logs such as below:

```java
WARN [NIOServerCxn.Factory:0.0.0.0/0.0.0.0:2181:NIOServerCnxn@357] - caught end of stream exception
```

Solutions/Actions:
• Decreasing Topics per Kafka cluster.
  • Huge partitions and/or Huge consumers caused an Overload of Zookeeper.
  • KafkaSpout will be setting Fetch Offset of each partitions to Zookeeper every fetching.

Result:
• The Kafka cluster is stabilizing to healthy.

Practice:
• Check implementations surrounding Kafka.
Summary
We introduced the 10 things that we learn from our IoT use cases.

- These things are important when we optimize the performance and we operate.
- Please be careful as these things cannot be learned from the documents only.
1. Any product name, service name, software name and other marks are trade mark or registered mark of corresponding companies.

2. This presentation is in a purpose of providing the knowledge gained from our activities on IoT field.

3. A presenter and NTT DATA Corporation provide information in as-is basis and have no responsiveness for results that you got according to information in this presentation material.
Any questions?

Naoto Umemori <umemorin@nttdata.co.jp>
Yuji Hagiwara <hagiwarayuj@nttdata.co.jp>