How we built a 1T/day stream processing cloud platform in a year
Who is this guy?

Peter Bakas I @peter_bakas

@ Netflix : Cloud Platform Engineering - Real Time Data Infrastructure
@ Ooyala : Analytics, Discovery, Platform Engineering & Infrastructure
@ Yahoo : Display Advertising, Behavioral Targeting, Payments
@ PayPal : Site Engineering and Architecture
@ Play : Advisor to Startups (Data, Security, Containers)
What should I expect?

- Architectural design and principles for Keystone
- Technologies that Keystone is leveraging
- Best practices
Let’s get down to business
Netflix is a data driven company

Culture of Analytics

Content
Product
Marketing
Finance
Business Development
Talent
Infrastructure
‘It may well be the most important document ever to come out of the Valley’

Sheryl Sandberg, COO @ Facebook
What does Freedom & Responsibility have to do with it?
By the numbers

700 billion unique events ingested / day

1 trillion unique events / day at peak of last holiday season

1+ trillion events processed every day

11 million events ingested / sec @ peak

24 GB / sec @ peak

1.3 Petabyte / day
How did we get here?
Chukwa
Chukwa/Suro + Real-Time branch
Keystone

Event Producer ➔ Control Plane ➔ Samza Router ➔ Consumer Kafka ➔ Stream Consumers

HTTP PROXY ➔ Fronting Kafka ➔ Consumer Kafka

EMR ➔ S3
Netflix Kafka Producer

- Best effort delivery
- Prefer msg drop than disrupting producer app
  - 99.99% delivery SLA
- Wraps Apache Kafka Producer
- Integration with Netflix ecosystem: Eureka, Atlas, etc.
Producing Events to Keystone

- Using Netflix Platform logging API
  - LogManager.logEvent(Annotatable): majority of the cases
  - KeyValueSeriazlier with ILog#log(String)
- REST endpoint that proxies Platform logging
  - ksproxy
  - Prana sidecar
Injected Event Metadata

- GUID
- Timestamp
- Host
- App
Keystone Extensible Wire Protocol

- Invisible to source & sinks
- Backwards and forwards compatibility
- Supports JSON. AVRO on the horizon
- Efficient - 10 bytes overhead per message
  - Message size - hundreds of bytes to 10MB
Fronting Kafka Clusters
Keystone

Event Producer -> Fronting Kafka -> Samza Router

Control Plane

EMR

S3

Consumer Kafka

Stream Consumers

Stream Consumers
Fronting Kafka Clusters

- Normal-priority (majority)
- High-priority (streaming activities etc.)
Fronting Kafka Clusters

- 3 ASGs per cluster, 1 ASG per zone
- 3000+ AWS instances across 3 regions for regular & failover traffic
## Deployment Configuration

<table>
<thead>
<tr>
<th></th>
<th>Fronting Kafka Clusters</th>
<th>Consumer Kafka Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of clusters</strong></td>
<td>24</td>
<td>12</td>
</tr>
<tr>
<td><strong>Total number of instances</strong></td>
<td>3,000+</td>
<td>900+</td>
</tr>
<tr>
<td><strong>Instance type</strong></td>
<td>d2.xl</td>
<td>i2.2xl</td>
</tr>
<tr>
<td><strong>Replication factor</strong></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>Retention period</strong></td>
<td>8 to 24 hours</td>
<td>2 to 4 hours</td>
</tr>
</tbody>
</table>
Partition Assignments

• All replica assignments zone aware
  • Improved availability
  • Reduce cost of maintenance
Current Challenges

- By using the d2-xl there is trade off between cost and performance
- Performance deteriorates with increase of partitions
- Replication lag during peak traffic
Routing Service
Keystone

Event Producer

Fronting Kafka

Samza Router

Control Plane

Consumer Kafka

Kafka

Kafka Control Plane

Keystone

S3

EMR

Stream Consumers
Routing Infrastructure

- Docker
- MySQL
- Samza 0.9.1
- Checkpointing Cluster
Router
Job Manager
(Control Plane)

Reconcile every min.

Zookeeper
(Instance Id assignment)

ASG

EC2 Instances

Job

Checkpointing Cluster

Router
Job Manager
(Control Plane)

Reconcile every min.

Zookeeper
(Instance Id assignment)

ASG

EC2 Instances

Job

Checkpointing Cluster
Routing Infrastructure

- Total of 13,000 containers on 1,300 AWS C3-4XL instances
  - S3 sink: ~7000 Containers
  - Consumer Kafka sink: ~ 4500 Containers
  - ElasticSearch sink: ~1500 Containers
Routing Job Details

- One Job per sink and Kafka source topic
  - Separate Job each for S3, ElasticSearch & Kafka sink
  - Provides better isolation & better QOS
- Batch processed message requests to sinks
- Offset checkpointed after batch request succeeds
Processing Semantics
Data Loss & Duplicates
Backpressure

Producer ⇐ Kafka Cluster ⇐ Samza job router ⇐ Sink

Keystone - at least once
Data Loss - Producer

- buffer full
- network error
- partition leader change
- partition migration
Data Loss - Kafka

• Lose all Kafka replicas of data
  • Safe guards:
    • AZ isolation / Alerts / Broker replacement automation
    • Alerts and monitoring
  • Unclean partition leader election
    • ack = 1 could cause loss
Data Loss - Router

- Lose checkpointed offset & the router was down for retention period duration
- Messages not processed past retention period (8h / 24h)
- Unclean leader election cause offset to go back
  - Safe guard:
    - Alerts for lag > 0.1% of traffic for 10 minutes
Data Duplicates - Router

- Messages reprocessed - retry after batch S3 upload failure
- Loss of checkpointed offset (message processed marker)
- Event GUID helps dedup
End to End Metrics

Producer to Router to Sink Average Latencies

- **Batch processing** [S3 sink]: ~1 sec
- **Stream processing** [Consumer Kafka sink]: ~800 ms
- **Log analysis** [ElasticSearch]: ~13 seconds (with back pressure)
Keystone Monitoring Service

- Broker monitoring *(Are you there?)*
- Heart-beating & continuous message latency monitoring *(Are you healthy?)*
- Consumer partition count and offset monitoring *(Are you delivering?)*
But wait, there's more
A True Story
Keystone went live 10.27.2015
2 days later...
Multiple ZooKeeper servers became unhealthy.

ZooKeeper quorum lost.

Producers dropped messages.

ZooKeeper quorum recovered.

Producers recovered?
Message drop resumed for two largest Kafka clusters with 300+ brokers. Controllers bounced and changed. Began rolling restart.
Lessons Learned

• There are times things can go wrong ... and no turning back
• Reduce complexity
• Minimize blast radius
• Find a way to start over fresh
Deployment Strategy

- Prefer multiple small clusters
  - Largest cluster has less than 200 brokers
- Limit the total number of partitions for a cluster to 10,000
- Strive for even distribution of replicas
- Have dedicated ZooKeeper cluster for each Kafka cluster
Failover

- Cold standby Kafka cluster with 3 instances and different instance type
- Different ZooKeeper cluster with no state
- When failover occurs
  - Scale up failover cluster
  - Create topics/new routing jobs for failover cluster
  - Switch producer traffic!
Post Failover

• Fix it!
• Rebuild it!
• Offline maintenance
• Fail back when ready
Failover

Time is the essence - failover as fast as 5 minutes

Fully Automated
Best Practices
This is an alert generated by Winston - Automated Troubleshooting and Remediation Platform.
Winston execution ID: 560caf56a42c9c0c998778a4

PROBLEM : Following kafka broker instances were reported to be offline.

kafkabroker i-74d2e982 - Instance is in 'running' state.

No disk failure detected. Kafka broker restarted Successfully.
Kafka Kong

In addition, we do Kafka Kong once a week.
What’s Next?
Keystone Messaging Service
Keystone Messaging Service

- Keystone - Unified event publishing, collection, routing for batch and stream processing
  - 85% of the Kafka data volume
- Ad-hoc messaging
  - 15% of the Kafka data volume
- Characteristics
  - Non-transactional
  - Message delivery failure does not affect user experience
Keystone Messaging Service

• Standard service layer for both producers and consumers
• Managed, multi-tenant service built on top of Kafka for its performance, scalability, simplicity and lower cost
Keystone Stream Processing Service
Keystone Stream Processing Service

- Stream processing consumers are **weary** of
  - Complexity of self-managed infrastructure
  - Choice of multiple runtimes across different platforms - none of which can solve all their use cases.
- Stream processing consumers **need**
  - Simple unified model / API / UI / System
Keystone Stream Processing Service

- Managed, multi-tenant platform with a self-service model
- Works across stream processing engines in use at Netflix
Keystone Management Service
Keystone Management Service
Creating a new Data Stream

- **Data Stream Name**: <data stream>
- **Owner Email**: <team@netflix.com>
- **Description**: <description>

**eu-west-1**
- **Max Bytes Per Second**: <blank or 0 for no traffic to eu-west-1>

**us-east-1**
- **Max Bytes Per Second**: <blank or 0 for no traffic to us-east-1>

**us-west-2**
- **Max Bytes Per Second**: <blank or 0 for no traffic to us-west-2>

- **QoS Preference**: Normal
- **Create**
- **Cancel**
Keystone Management Service
Keystone Management Service
Keystone Management Service
Keystone Management Service
Keystone Management Service
Keystone Management Service
Keystone Management Service
Keystone Management Service