Kafka: a Distributed Messaging System for Log Processing

Jay Kreps, Neha Narkhede, Jun Rao
LinkedIn
AGENDA

• Kafka usage at LinkedIn
• Kafka design
• Kafka roadmap
ABOUT LINKEDIN

• Professional social network platform
• top 50th largest site in the world (traffic)
• 100M+ members
LOGGING OVERVIEW

- Many types of events
  - user activity events: impression, search, ads, etc
  - operational events: call stack, service metrics, etc
- High volume: billions of events per day
- Both online and offline use case
  - reporting, batch analysis
  - security, news feeds, performance dashboard, ...
KAFKA DESIGN PRINCIPLES

• Simple API
• Efficient
• Distributed
PRODUCER API

```java
void send(String topic, ByteBufferMessageSet messages)

producer = new KafkaProducer(...);
message = new Message("test message str".getBytes());
set = new ByteBufferMessageSet(message);
producer.send("test", set);
```
CONSUMER API

```java
streams[] = Consumer.createMessageStreams("test", 1)

for(message: streams[0]) {
    bytes = message.payload()
    // do something with bytes
}
```
EFFICIENCY #1: SIMPLE STORAGE

• Each topic has an evergrowing log
• A log == a list of files
• A message is addressed by a log offset
EFFICIENCY #2: CAREFUL TRANSFER

- Batch send and fetch
- No message caching in Kafka layer
- Rely on file system page cache
  - mostly, sequential access patterns
- Zero-copy transfer: file -> socket
EFFICIENCY #3: STATELESS BROKER

- Each consumer maintains its own state
- Message deletion driven by retention policy, not by tracking consumption
  - acceptable in practice
  - rewindable consumer
AUTO CONSUMER LOAD BALANCING

- brokers and consumers register in zookeeper
- consumers listen to broker and consumer changes
- each change triggers consumer rebalancing
CONSUMER PERFORMANCE
ROADMAP

• New Kafka features
  • compression
  • replication
  • stream processing (online M/R)

• http://sna-projects.com/kafka/