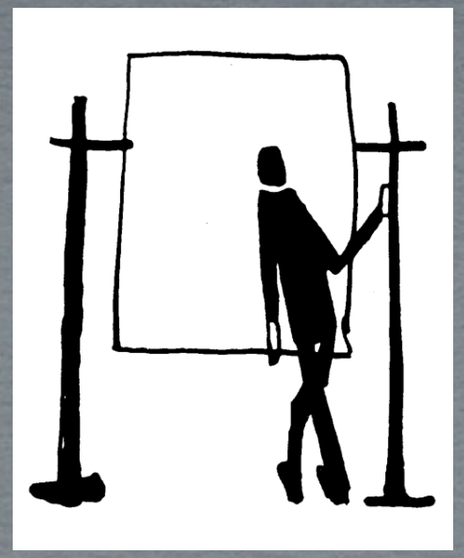
Kafka

A pub-sub messaging system for web-scale stream processing

Neha Narkhede - Distributed Systems at LinkedIn



Kafka

A pub-sub messaging system for web-scale stream processing

Neha Narkhede - Distributed Systems at LinkedIn

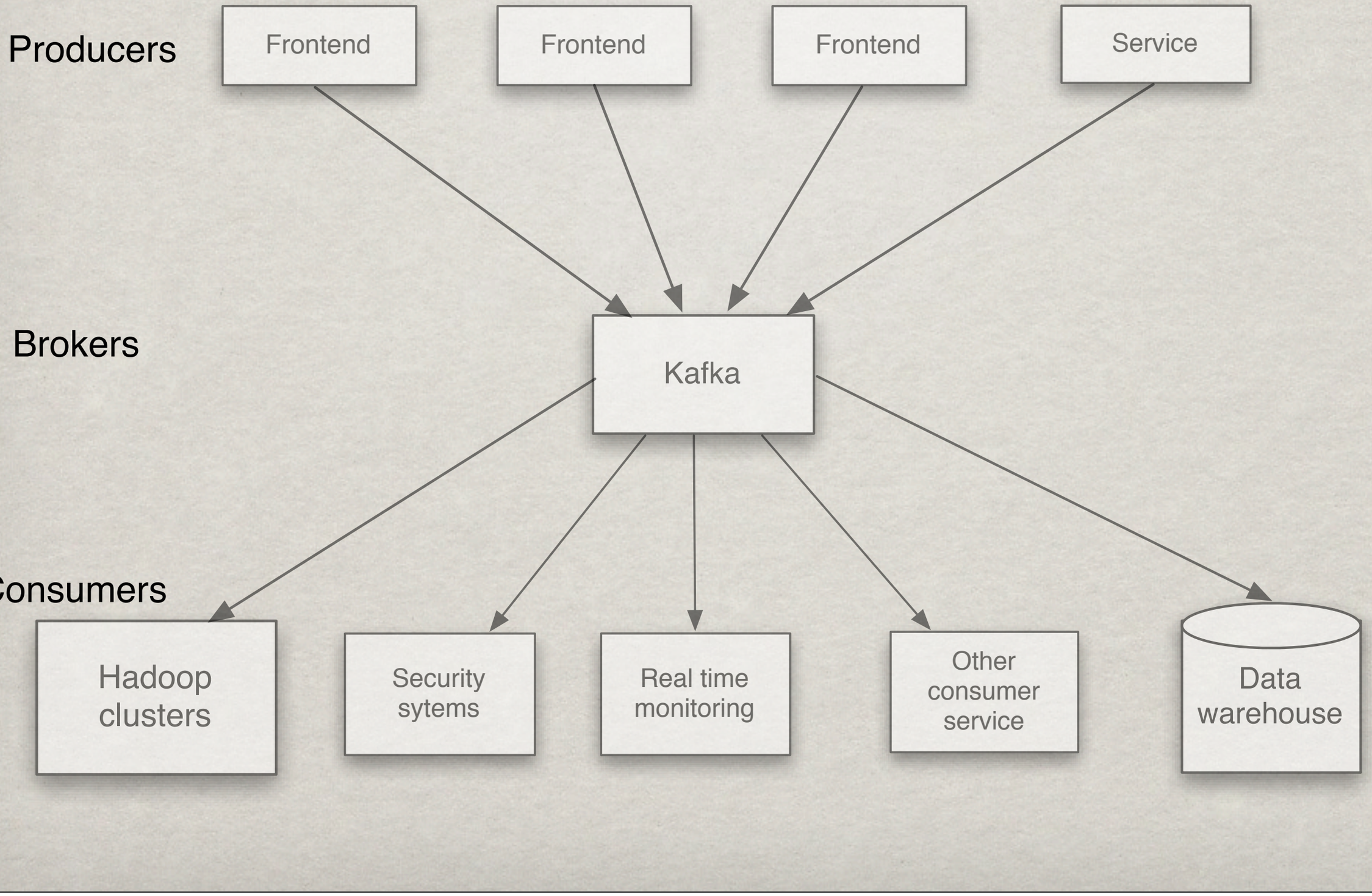
One sentence:

Kafka is a distributed, high throughput, persistent pub-sub messaging system

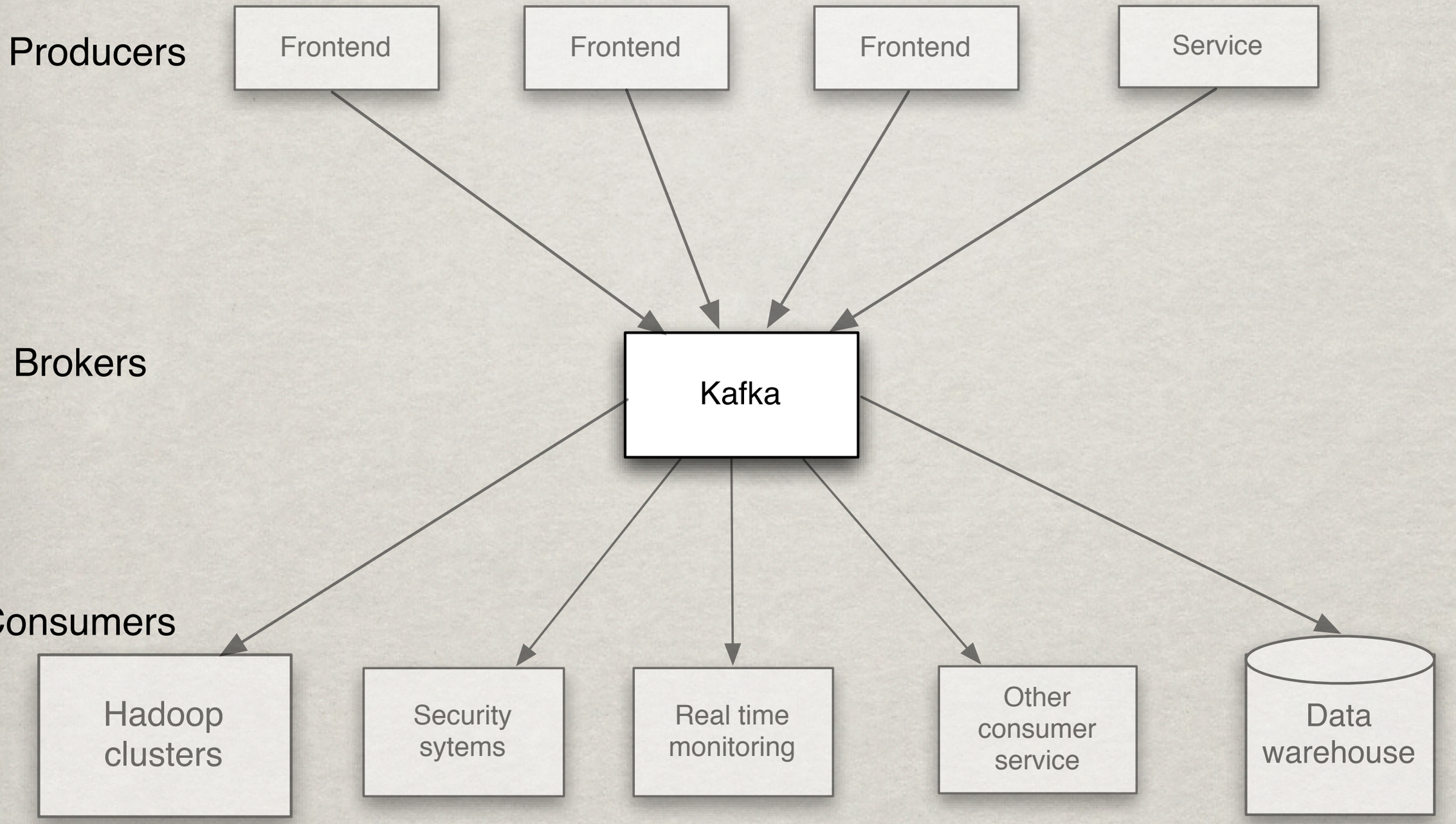
REQUIREMENTS

- Support consumers operating at different rates (real time & offline)
- Allow very parallel consumption (e.g. Hadoop)
- Pick throughput and performance over fancy features

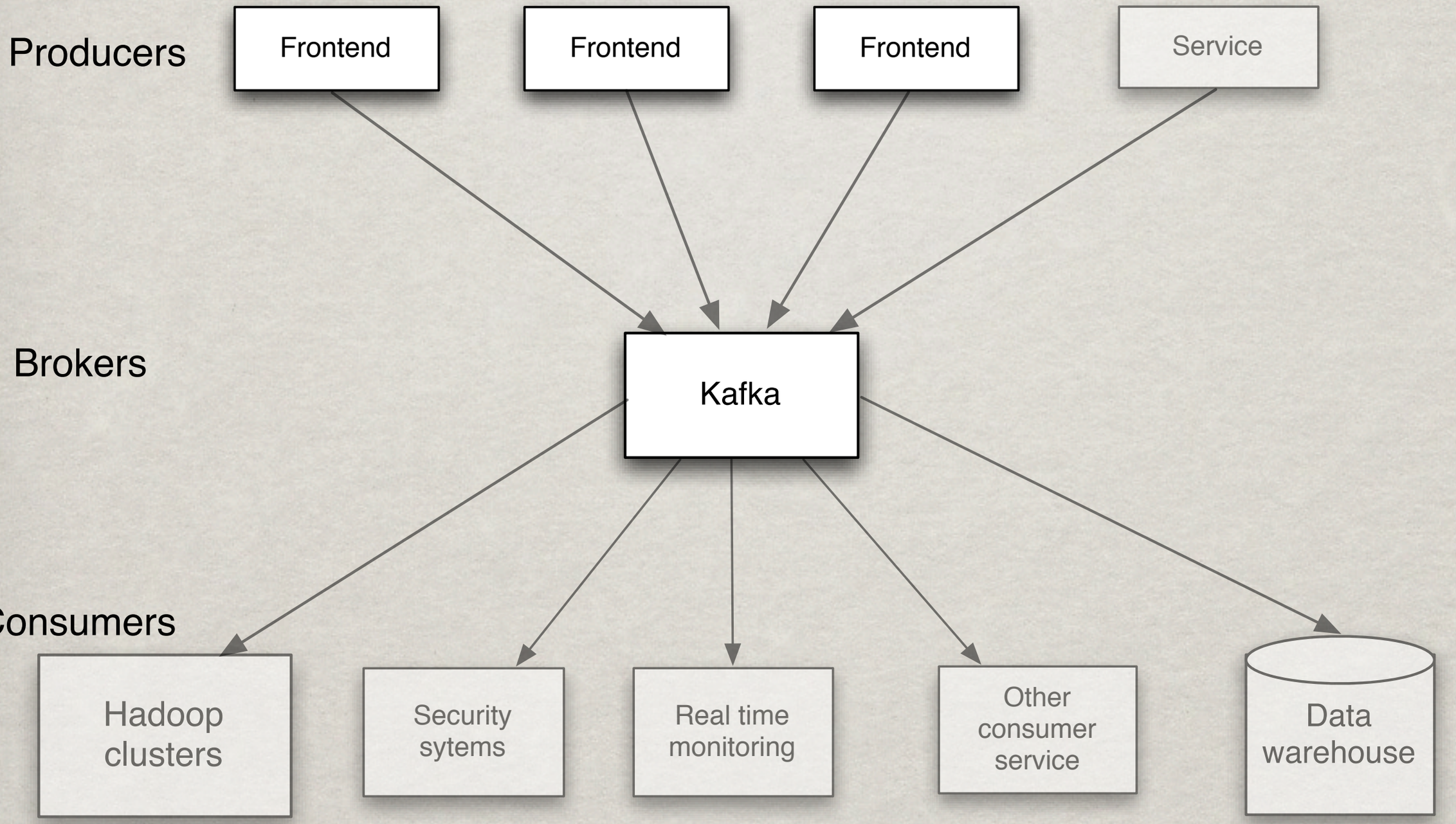
KAFKA AT LINKEDIN



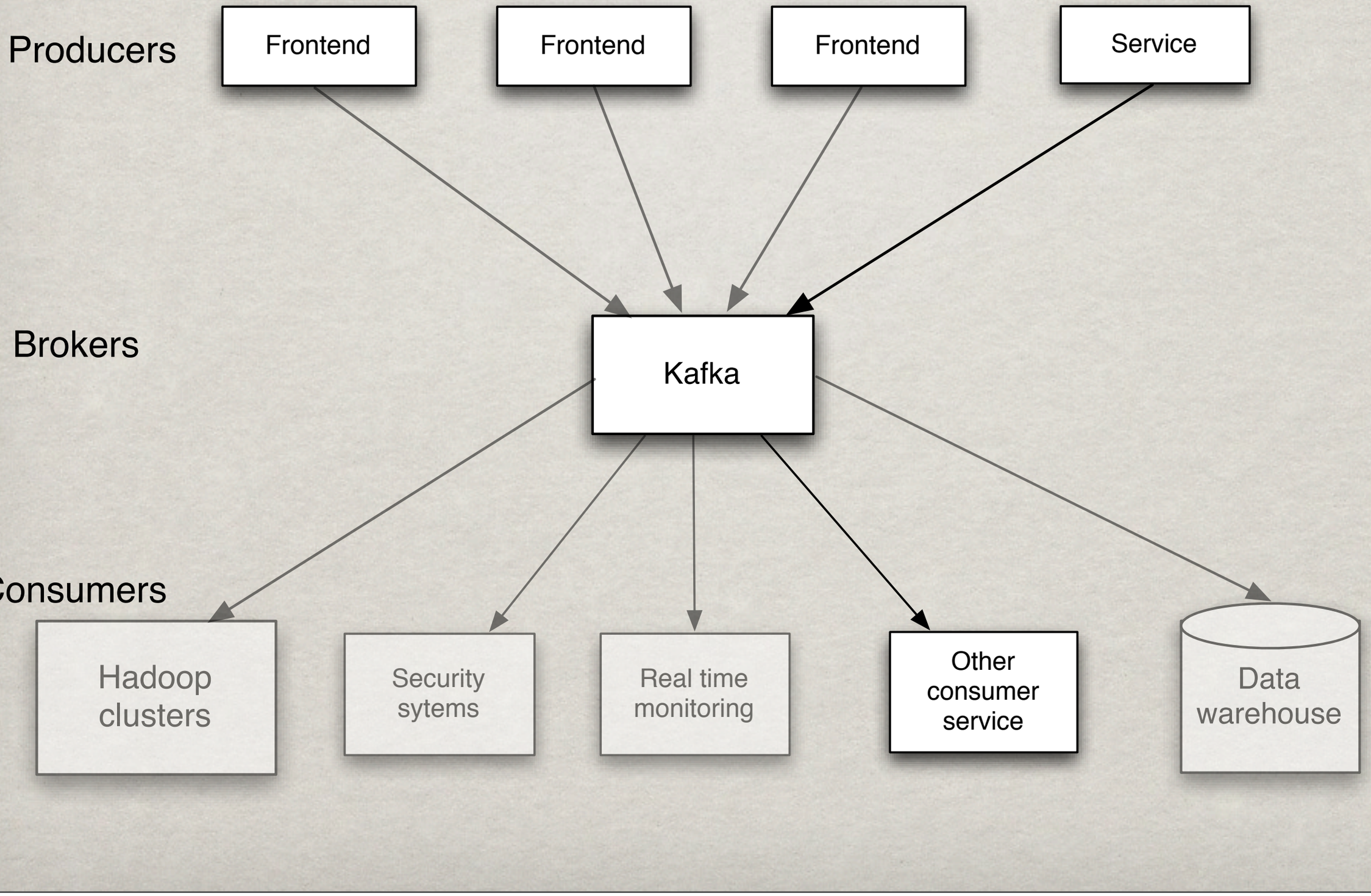
KAFKA AT LINKEDIN



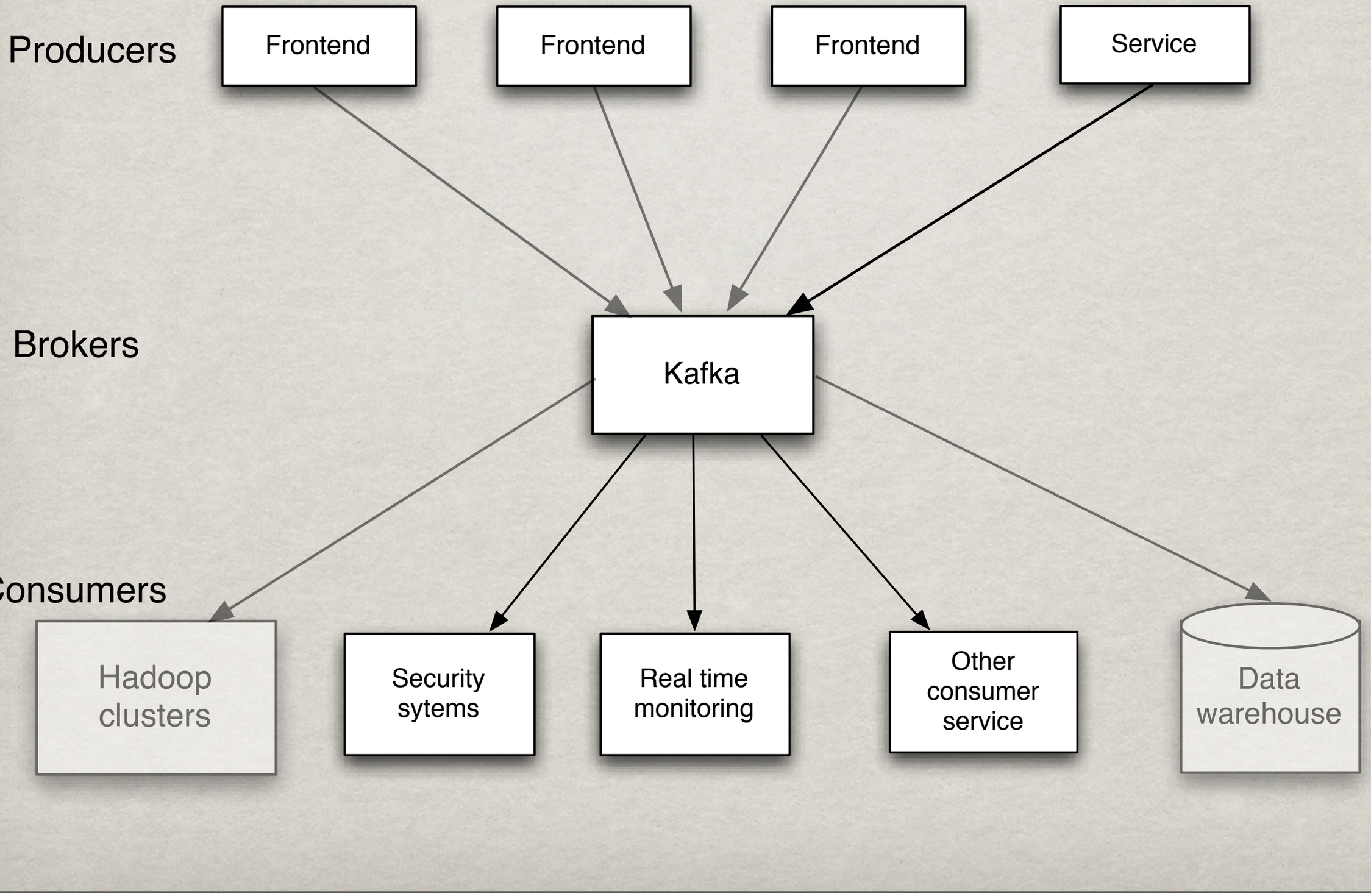
KAFKA AT LINKEDIN



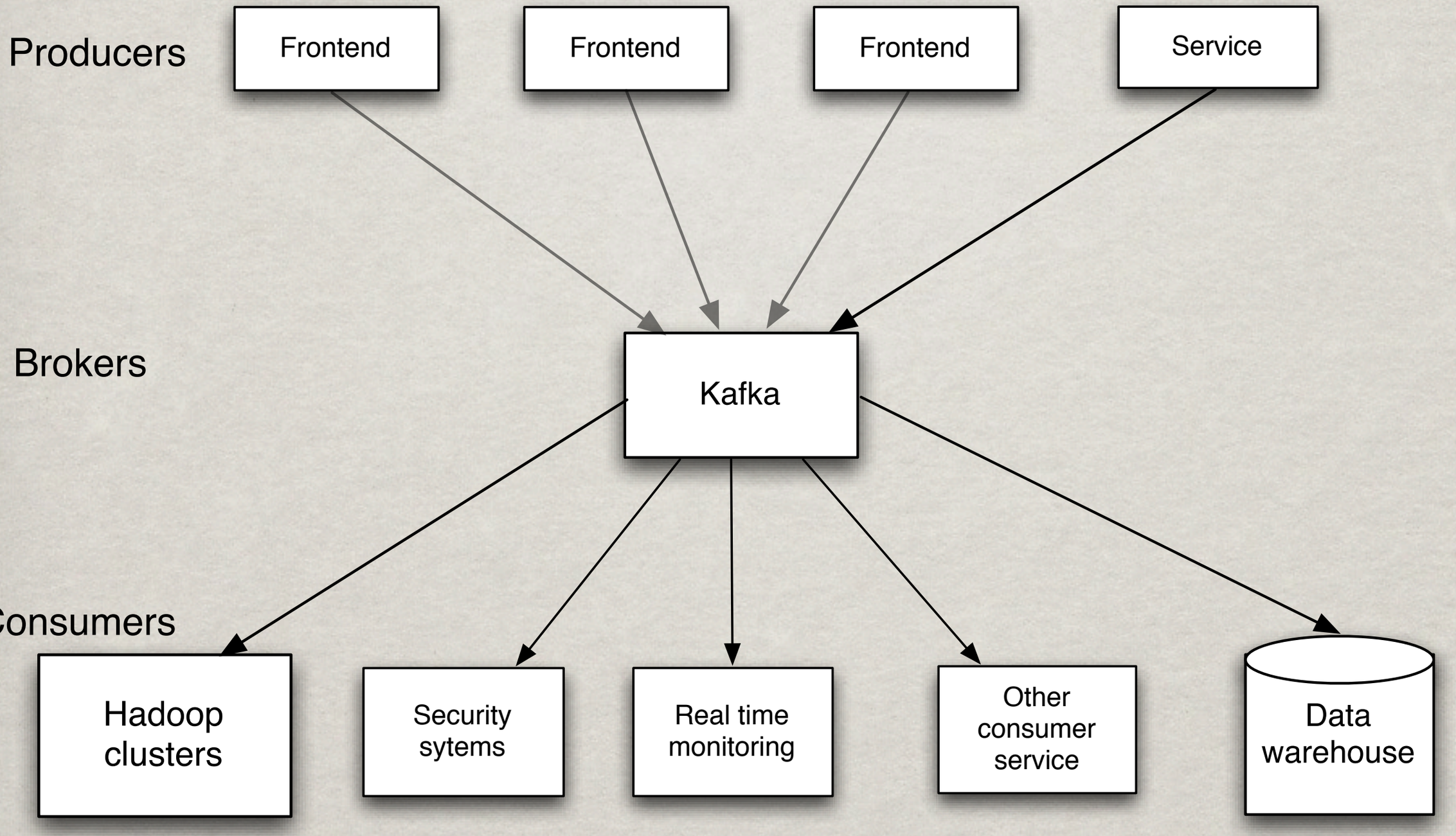
KAFKA AT LINKEDIN



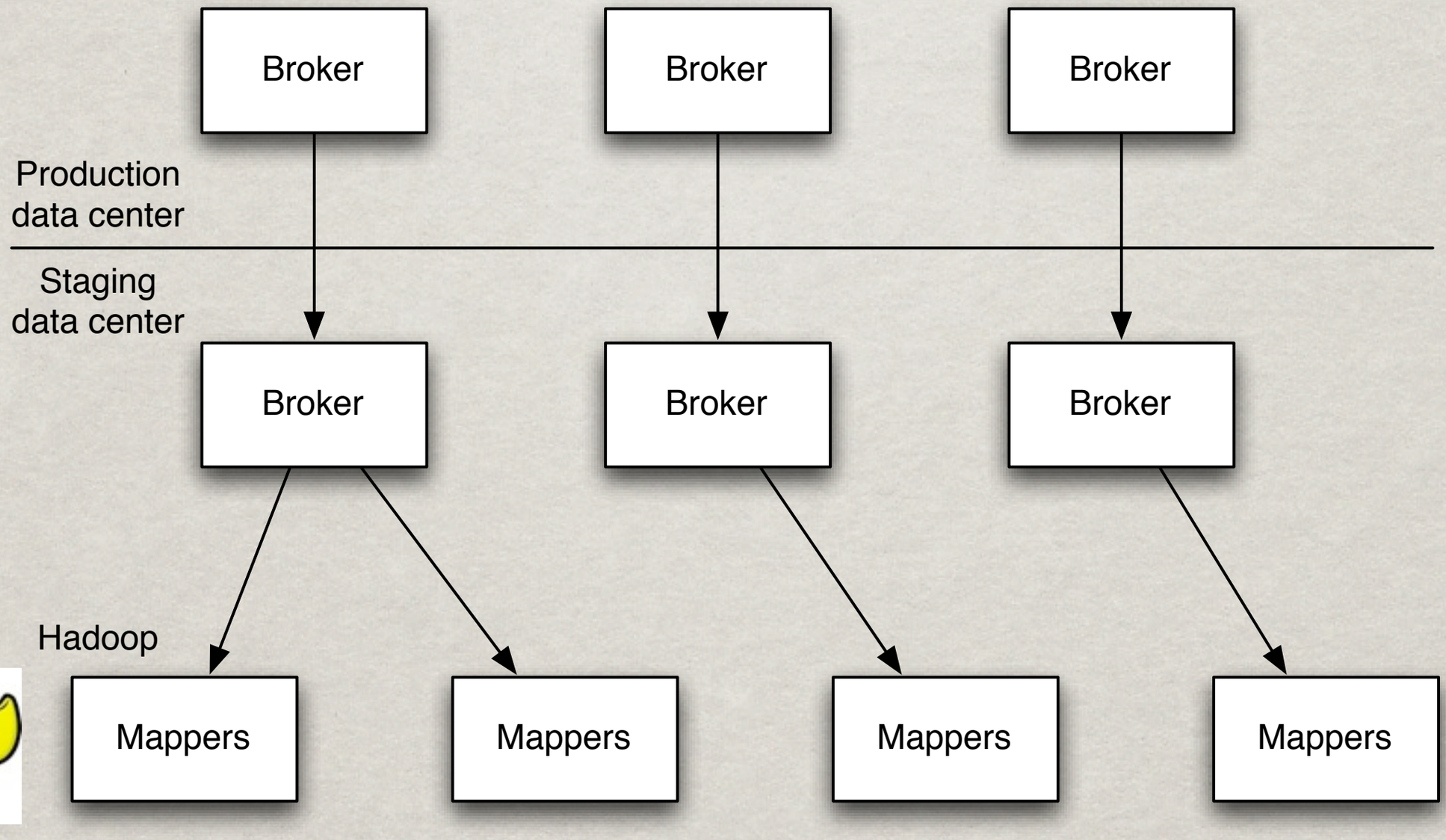
KAFKA AT LINKEDIN

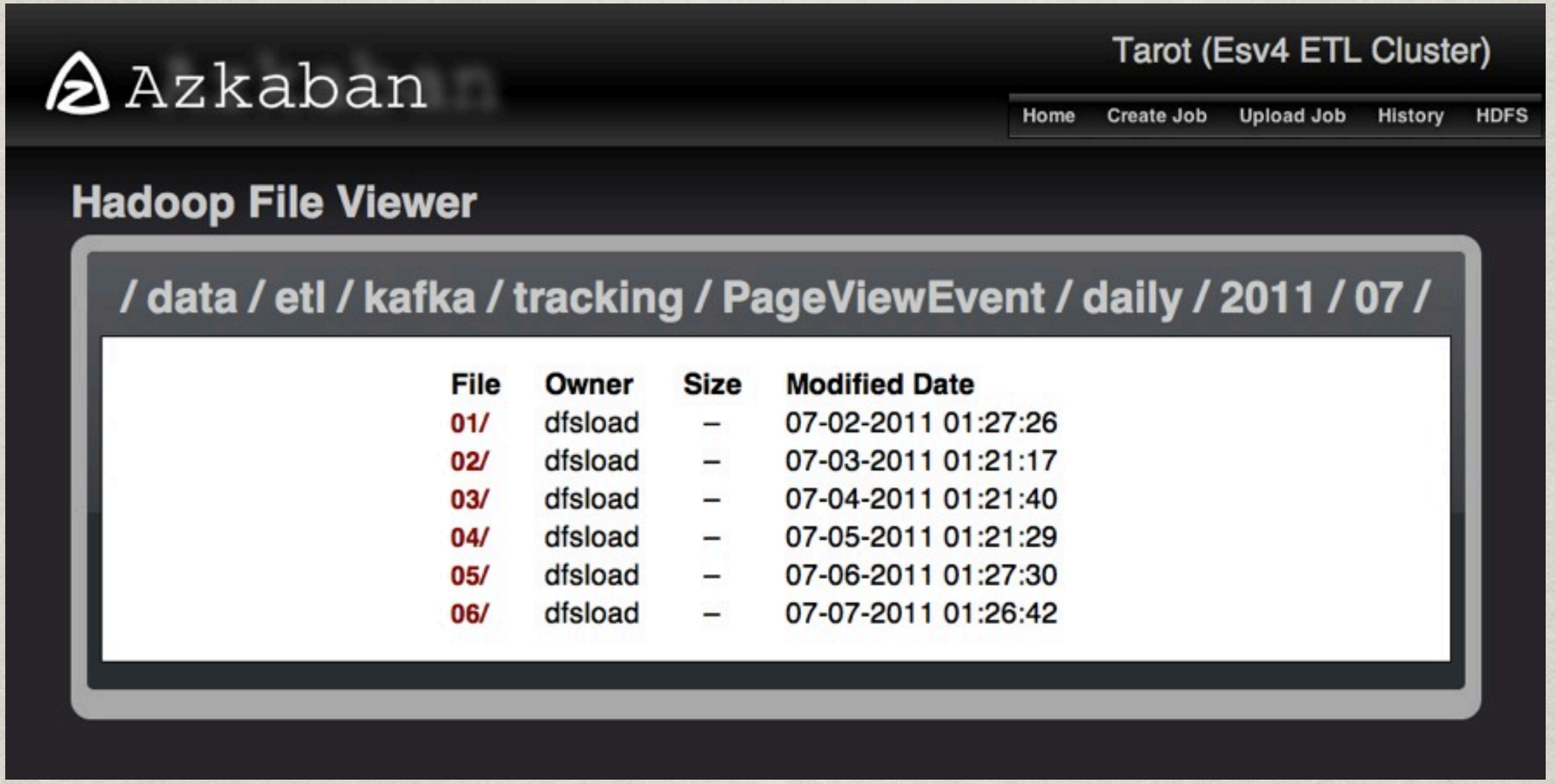


KAFKA AT LINKEDIN



HADOOP INTEGRATION




Azkaban
Tarot (Esv4 ETL Cluster)
Home
Create Job
Upload Job
History
HDFS
Hadoop File Viewer
/ data / etl / kafka / tracking / PageViewEvent / daily / 2011 / 07 /
File
Owner
Size
Modified Date
01/ dfsload – 07-02-2011 01:27:26
02/ dfsload – 07-03-2011 01:21:17
03/ dfsload – 07-04-2011 01:21:40
04/ dfsload – 07-05-2011 01:21:29
05/ dfsload – 07-06-2011 01:27:30
06/ dfsload – 07-07-2011 01:26:42

Production Kafka

Broker

Broker

Broker

Production data center

Staging data center

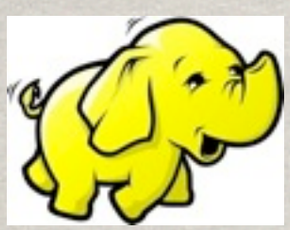
Staging Kafka

Broker

Broker

Broker

Hadoop



Hadoop

Hadoop

Hadoop

VOLUME

- billions of events / day
- couple of terabytes / day

GUARANTEES

- No corruption of data
  - over the network
  - on the disk
- At least once delivery

Claim: These are the strongest possible guarantees given the performance constraints

PRODUCER API

```
package scalathon.kafka.producer

import java.util.Properties
import kafka.message.Message
import kafka.producer.{ProducerConfig, ProducerData, Prod
ucer}

object ProducerExample {
  def main(args: Array[String]) {
    // set the right config options
    val props = new Properties
    props.put("zk.connect", "127.0.0.1:2181");
    val config = new ProducerConfig(props);
    val producer = new Producer[Message, Message](config);
    // Kafka is not in the serialization business
    val topic = "Kafka-Novels"
    val data = "The Metamorphosis".getBytes
    val producerData =
      new ProducerData[Message, Message](topic,
                                         new Message(data))
    // send data to Kafka
    producer.send(producerData)
    producer.close
    println("Sent message -> test-message")
  }
}
```

Consumer Api

```scala
package scalathon.kafka.consumer

import java.util.Properties
import kafka.consumer.{Consumer, ConsumerConnector, ConsumerConfig}
import java.util.concurrent.Executors
import org.I0Itec.zkclient._
import kafka.utils.StringSerializer

object ConsumerExampleSingleThread {
  def main(args: Array[String]) {
    // set the right config options
    val props = new Properties
    props.put("zk.connect", "127.0.0.1:2181")
    props.put("groupid", "kafka-fans")

    val topic = "Kafka-Novels"
    // pull all the Novels in ONE stream
    val partitions = 1

    val consumerConfig = new ConsumerConfig(props)
    val consumerConnector: ConsumerConnector =
      Consumer.create(consumerConfig)
    // create the message streams for topic Novels
    val topicMessageStreams =
      consumerConnector.createMessageStreams(Predef.Map(topic -> partitions))

    val novelsTopicStreams = topicMessageStreams.get(topic).get
    // we just have one novel topic stream
    val novelsTopicStream = novelsTopicStreams.head
    for(novel <- novelsTopicStream)
      println("Woohoo ! Time to read " + kafka.utils.Utils.toString(novel.payload,
  }
}
```

Project ideas

- http://sna-projects.com/kafka/projects.php
- Producer ACK
- Consumer pluggable decoder
- RESTful Kafka server

ROADMAP

- Apache incubator inclusion
- Successfully deployed in production at LinkedIn
- https://github.com/kafka-dev/kafka
- New Kafka features
  - End-to-end block compression
  - Replication
  - stream processing (online M/R)

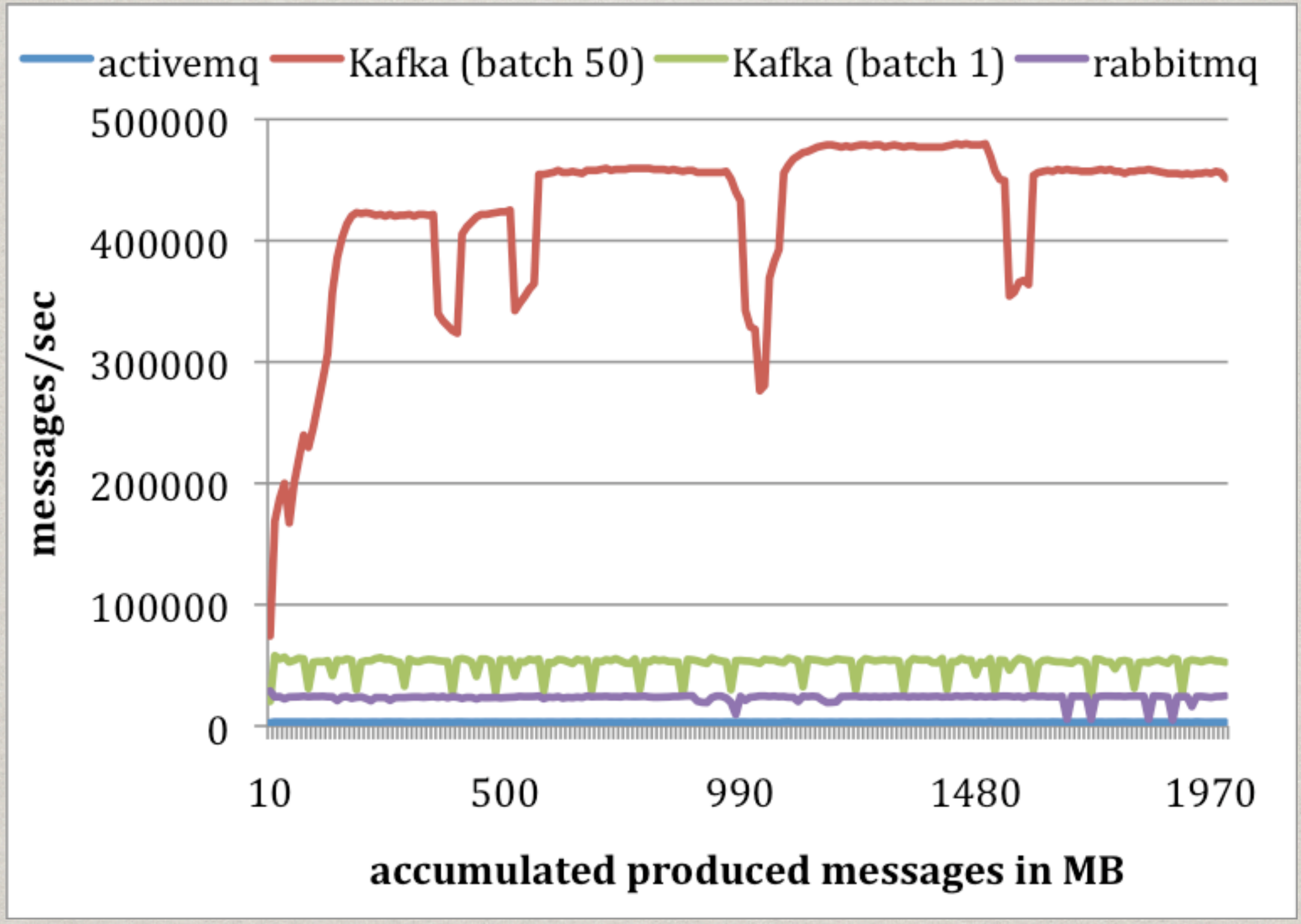
QUESTIONS ?

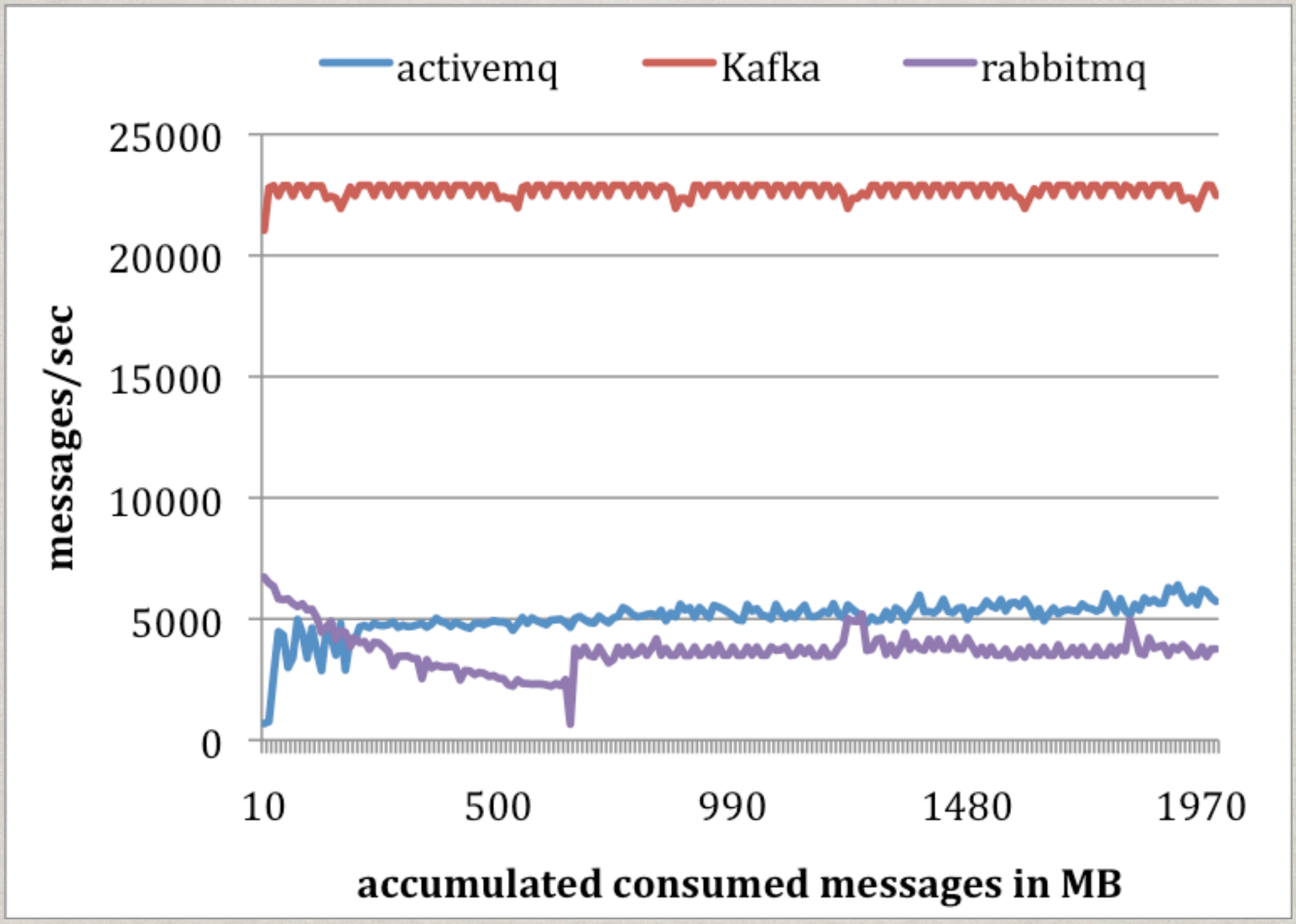
http://sna-projects.com/kafka
https://github.com/kafka-dev/kafka
kafka-dev@incubator.apache.org
neha.narkhede@gmail.com

PERFORMANCE

- 2 Linux boxes
  - 8 2.0 GHz cores
  - 6 7200 rpm SATA drive RAID 10
  - 16GB memory
  - 1Gb network link
- 200 byte messages
- 10 million messages
- 1 consumer thread

- Kafka
  - flush 10K messages
  - batch size = 1 and 50
- Active MQ
  - syncOnWrite=false
  - KahaDB
- RabbitMQ
  - mandatory=true, immediate=false
  - persistent=true


activemq
Kafka (batch 50)
Kafka (batch 1)
rabbitmq
500000
400000
300000
200000
100000
0
messages/sec
10
500
990
1480
1970
accumulated produced messages in MB


activemq
Kafka
rabbitmq
25000
20000
15000
10000
5000
0
messages/sec
10
500
990
1480
1970
accumulated consumed messages in MB