Machine Learning in Production / AI Engineering - Recitation 2

Outline

1. Background
2. Synchronous vs Asynchronous communication
3. Synchronous Communication
4. Asynchronous Communication - Stream Processing
5. Parts of a Kafka system
6. SSH Tunneling
7. Kafkacat utility and kafka-python
8. Parts of a Kafka system
9. Activity - Design decisions
10. Useful Resources

Background

- Enormous volume of data - Big Data
- Challenges with Big Data:
  - Data collection and management
  - Data analysis
  - Data communication

Synchronous vs Asynchronous Communication

Synchronous Communication

Sender and receiver both need to be active at the same time

Scenario: Point-to-point, On-demand messages

Example: Rest API, RPC

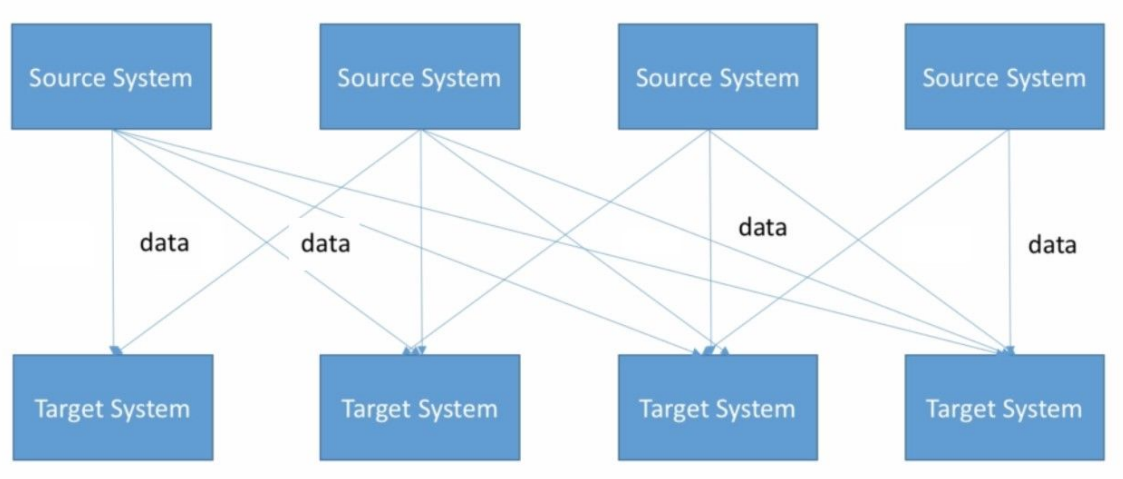
Asynchronous Communication

Sender and receiver need not be active at the same time

Scenario: Stream processing, Publisher/Subscriber model

Example: Message Queues, Kafka

Synchronous Communication



Point-to-point communication

Not scalable, every link needs to be established explicitly as source and targets increase

Asynchronous Communication - Stream Processing

- Data processing paradigm that processes data in motion
- Computation performed on data as it is produced and received
- Data enters into a queue processed in a first-in-first-out order
- Data in the queue can be parallelized to scale throughput
- **Event-based architecture or publisher-subscriber architecture**

Kafka

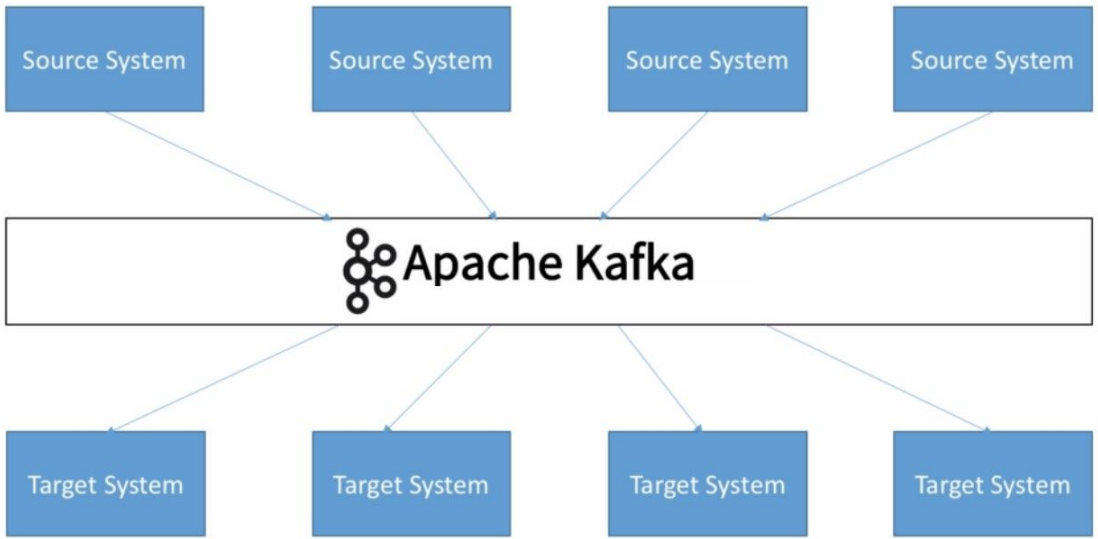
- Open source, distributed streaming platform
- Allows for development of real-time event-driven applications
- Applications that continuously produce and consume data
- Data produced can be consumed by high volumes of users simultaneously
- Maintains order of occurrence of data

Kafka

- ★ Easy to scale
- ★ Resilient to failures, as messages are replicated
- ★ Allows for a Publisher/Subscriber model
- ★ Highly configurable as required
- ★ Kafka is becoming a de facto standard for streaming data processing

Thanks to its versatility!

Kafka

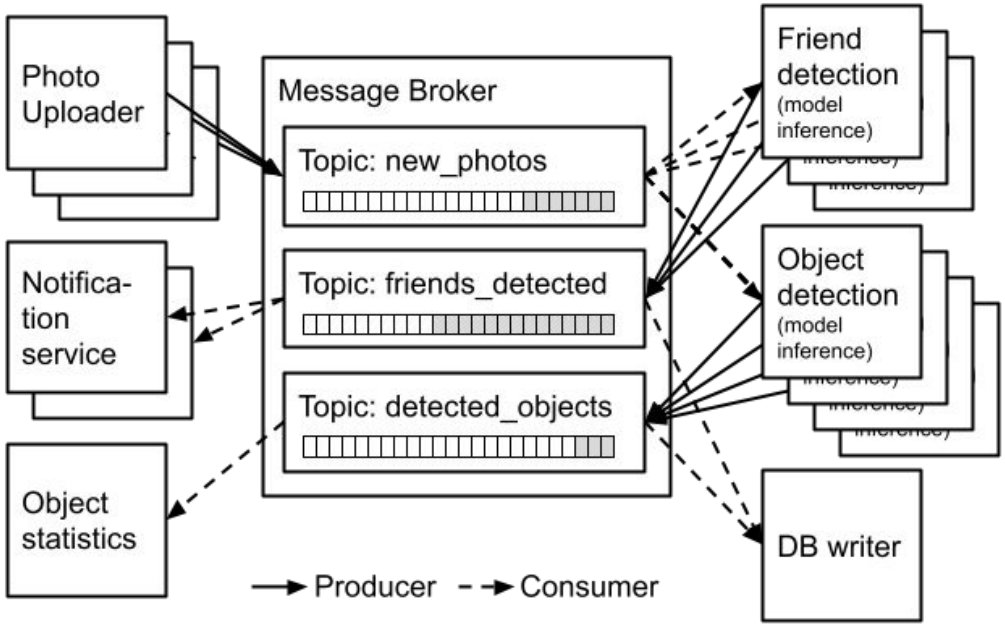


Source and target systems now talk to Kafka.

Very scalable, Kafka handles all incoming and outgoing communication.

Systems do not need to know about each other to function.

Parts of a Kafka System



Parts of a Kafka System

- Producer: Applications or processes that produces messages
- Consumer: Applications or processes that consume messages
- Broker: A server that manages storage of messages
  - Multiple brokers come together to form a cluster
- Zookeeper: Service to manage brokers and monitor clusters

SSH Tunneling

- Why SSH tunneling?
  - VMs on cloud are always vulnerable. Ports should be kept closed.
  - SSH establishes secure tunnel to connect a service running on a port on a VM to your local system
  - Generic way to treat remotely hosted services as if they’re running locally

- How to set up a tunnel?

```
ssh -L <local_port>:localhost:<remote_port> <user>@<remote_server> -NTf

## Understanding the options used left as an exercise
```

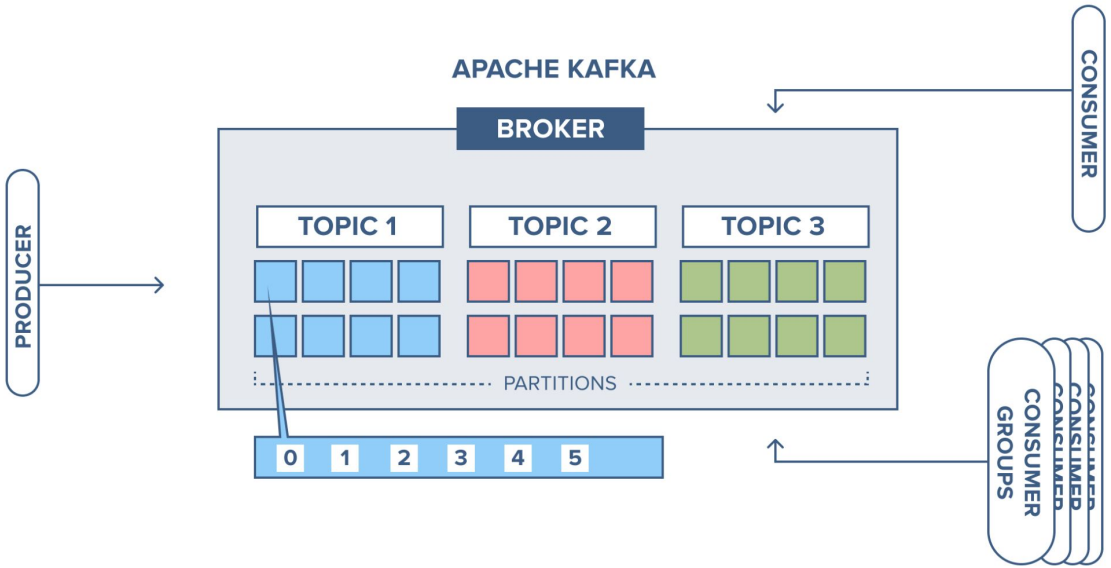
Kafkacat utility & kafka-python

- Kafkacat: CLI tool for interacting with Kafka broker
  - Produce and consume events
  - Manage topics
- Kafka-python: Python library for Kafka
- In the demo, we would be:
  - Opening an SSH tunnel to Kafka broker
  - Examining usage of Kafkacat and kafka-python

Demo

Kafkacat utility & kafka-python

Parts of a Kafka System



Source: [What is Apache Kafka?]

Parts of a Kafka System

- Topics: How Kafka messages are stored and organized
  - Producers publish to topics, and consumers subscribe to topics
  - Single producer can publish to multiple topics; single consumer can subscribe to multiple topics

- Partitions: An ordered split of a topic
  - Done for increasing throughput via parallelism
  - Message ordering is only preserved within a partition
  - Messages within a partition are immutable

Parts of a Kafka System

- Offsets: An identifier associated with a message within a partition
  - Used by consumer to read messages in any order
  - Typically, it would be linearly advanced

- Events: A message in Kafka is also referred to as an event
  - Events are nothing but byte arrays
  - Kafka inherently supports messages of any format required

Activity - Design decisions

- You have a system that accepts an audio clip as an input and transcribes it into text
  - How about video?

- You work for Uber, and you want to keep track of your drivers' location

- You are designing Amazon's recommendation system, and you want to gather users' preferences from their online activities

Useful Resources

- https://kafka.apache.org/
- https://www.cloudkarafka.com/blog/2016-11-30-part1-kafka-for-beginners-what-is-apache-kafka.html
- https://www.tibco.com/reference-center/what-is-apache-kafka
- https://www.youtube.com/watch?v=PzPXRmVHMxI
- https://www.youtube.com/watch?v=JalUUBKdcA0