Kafka Fundamentals Brain Teasers: M2: Use Cases and Motivations

Overview

The first part of learning Kafka fundamentals is understanding what Kafka is good at and exploring use cases. Key concepts are:

- Real time vs. batch processing
- Use cases

Here's the quick quiz on Module 2 (https://forms.gle/ww7MZ74BmLsYrc1R6) from the Online Talk Series.

Note: For this form of these brain teasers, I've opted to number the problems so that they correspond with module numbers from the Fundamentals course. Thus, we start with Problem #2, corresponding to Module 2; there is no #1.

Problem #2A: Applying Kafka to a Digital Jukebox Use Case

Suppose you are building a network of digital jukeboxes that operate in restaurants, bars, event spaces, etc. Users can maintain accounts and use credit cards and a phone app to purchase credits to play songs. Using the same app, users can play songs, as long as they are located within a reasonable distance of a jukebox they are targeting. So, thinking of some of the things Kafka is cut out for, what are at least 3 reasons Kafka is well-suited for this application? (Hint: Review the Module 2: Motivation & Customer Use Cases lecture and/or the handbook.)

Solution #2A: Applying Kafka to a Digital Jukebox Use Case

Kafka is designed to process events in *real time*. Imagine you played a song at a digital jukebox somewhere. You want to hear it right away. Now, you might have to wait your turn. Say you play a song somewhere you're at for dinner and then have some daily batch process come in and process all the song play requests at 3:00 a.m. Then it queues them up to play the next day. This does you no good. We want a real-time reaction.

In the world we live in, people are becoming increasingly more hungry to have this happen "now" or "ASAP," and Kafka is great at supporting that. Behind the scenes, when you pick your song, the Jukebox user application is using a Kafka producer application to write a message (with your song request) to its cluster. Separately, to put it simply, a consumer is subscribed to all of the unplayed song play requests and picking out the ones on this jukebox and playing them.

The system might look like this:

Kafka is also well suited for this use case for security reasons. It is possible to enable Kafka various security features, and in this application, most users would agree that's an absolute must. Here's why:

- Users are using credit cards to pay for credits to play songs. It goes without saying that credit card information must be kept secure. With Kafka, it's easy to configure the underlying system to encrypt credit card information so hackers can't sniff it as it is going over the "wire."
- Location data: We probably don't want someone at another location playing music at that location's jukebox. Certainly, if you're the user and you play a song, you want to hear it where you are sitting, not a thousand miles away. (Well, in most cases!) But to do this, the application needs location data from the users. A user's location data is sensitive personal information. This is something else that you would like to keep secured from the outside world if you're running the digital jukeboxes and want to gain customer trust. Kafka's security features can make it easy. (Furthermore, if you're writing information about song play requests to an external system, e.g., a database, there's a tool called Kafka Connect[https://kafka.apache.org/documentation/] that can help make that super easy, and it features Single Message Transforms[https://www.confluent.io/blog/kafka-connect-single-message-transformation-tutorial-with-examples/] that make it easy to mask personal information.)

Another important feature of Kafka is fault tolerance. In case of a failure, with the right settings, your Kafka cluster can keep running and no data will be lost. As the jukebox implementor, you want to keep track of people's accounts and plays accurately. As a user, you want to be able to access your account and access all the songs you can play any time. You also want the jukebox not to miss your plays. Let's hone in on the latter. The back end for deciding what to play next would use a consumer to fulfill outstanding play requests; you as a user want *your* play requests always to be available so that you hear your song. The good news is that, in administering Kafka or developing for Kafka, it's easy to turn on the key feature that makes data highly available.

Other reasons why Kafka fits here can be categorized in two ways: Events are saved and Kafka supports multiple consumption. Here's what that means: In some messaging systems, once a message is processed, it's removed from the system. It can only be processed at most once. In Kafka, once a message has been read by a consumer, it does not get automatically deleted. So an event might be processed for some reason where a real-time response is needed, but we might also want to process it again for some different reason with less immediate needs.

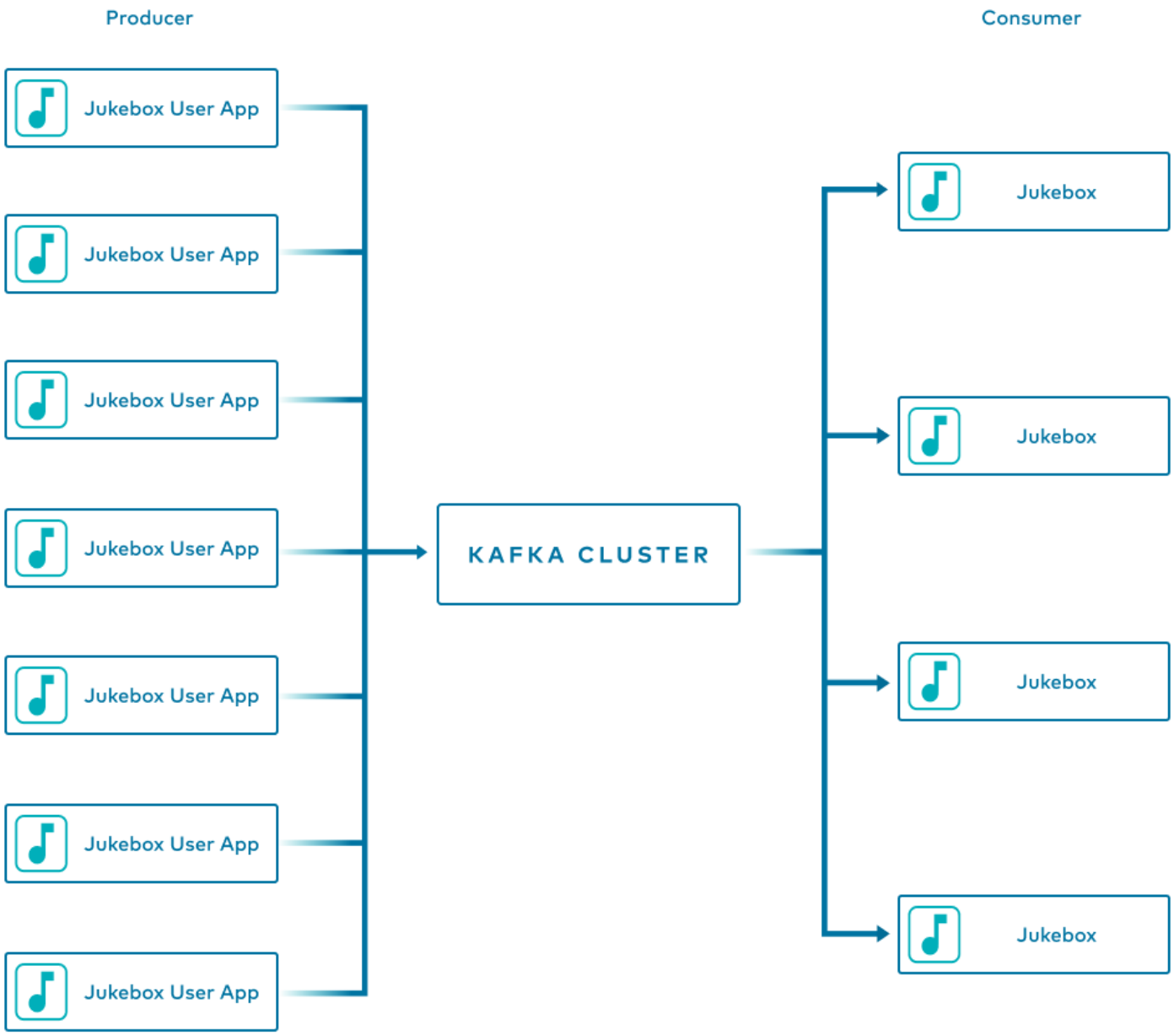
This fits into the jukebox example because:

- As the implementer of the jukebox application, the immediate goal is to do just what one of the jukeboxes 50 years ago did: Play the song and be done. But with Kafka, you can do more. For example, you can consume play requests again to:

- Create a “top 10 most played songs here” list, daily, weekly—you name it
- Display a screen of songs people played on other jukeboxes most recently
- Provide a user with a “play again” screen with options to pick from play history
- Provide a user with a “other songs from your artists” screen, built by looking at play requests already fulfilled

- You can also perform other actions not immediately tied to the jukebox, which we’ll address soon.

Finally, the “single platform” concept of Kafka is a nice feature here. Without a doubt, such a system is going to need network connectivity to access songs to play, unless it has a very limited selection. So we’re already going to be connected to the internet. While we could implement all of the software local to the device, having all of the data in a central platform—the Kafka cluster—not only are users able to use the same app at different jukebox locations, but song play data is also shared across multiple locations and a lot can be done with that. Here’s a tweaked visual to wrap this up:



Problem #2B: How Kafka Can Make the Digital Jukeboxes Go Even Further

Suppose you are partnered with a company that sells tickets to concerts and music-related merchandise. Could you add additional services to your existing Kafka cluster? What kinds of things could you do to advance both businesses? What Kafka features would be important?

Solution #2B: How Kafka Can Make the Digital Jukeboxes Go Even Further

The key motivation for this question gets back to that issue of multiple consumption—"what makes Kafka *Kafka*." Aside from playing the song, with the appropriate legal permissions from the user, you have data that could be used again outside of the core jukebox application.

Because you're partnered with a concert ticketing company, you can tap into their database and produce messages or events into Kafka (using Kafka Connect) about concerts, their locations, and their dates. You can use the data stored within Kafka to identify events that might interest your users. Perhaps you aggregate each user's play data to identify users who've played a certain artist five or more times in the last month. You can then join this with the concert data to find a nearby upcoming show that the user might be interested in. Streaming systems like Kafka Streams (https://docs.confluent.io/platform/current/streams/index.html) and ksqlDB (https://ksqldb.io/) are your best tools for this job. Further, you can produce messages including user email and upcoming shows nearby. You can then have a consumer consume such messages and send concert promotional emails. All of this is possible using Kafka and the tools available in the Kafka ecosystem.

Similarly, you could ingest information from a new music releases database (maybe from your music merchandise partner or in general). Then you could join this information with information from aggregating based on your users' favorite artists and let them know when new music is about to be released, in a similar fashion to concerts. The same goes for working with the music merchandise company partner to promote sales of new album releases, clothes, posters, etc.

As for Kafka features that would be important to make this application possible, multiple consumption is based on the "event saved" feature, and certainly, any of these applications involve personal information, so security is vital.

Next, let's shadow Module 3 of the Apache Kafka Fundamentals course.