Enterprise Message Queues

Overview

Silos/Stovepipes

- Silos (sometimes called stovepipes) occur when a production system exists independently with little or no outside connectivity
- Analogy to grain silos
- Analogy to wood- or coal-burning stovepipe

Goals

Goals for getting data from our production silo

- Minimal impact on performance of the production system
- Fresh, not stale, data
  - As close to real time as possible
- Generic, reusable interface
  - Not writing a custom interface for each system that needs data
- Any system that needs data can get it
- Scales up

Enterprise Message Queue

- Allow production systems to stream data in the form of messages in real time
- Minimal impact on the production system
  - Only must write the message once
- Enterprise message queues
  - Buffer messages
  - Can retain them for a specified period
  - Can distribute them to numerous other systems

Topics

- Analogous to a file system hierarchical structure
- Topics of topics of topics etc.
- Messages are written to one or more topics
- Read from a topic
- Several models for how messages are read from topics

Models

- Publisher-subscriber model
- Producer-consumer model
- Streaming model

JSON

- Messages can be any data binary or text.
- Originally, messages were XML.
- In the modern era, JSON is typical.

Object Store

- If we have large objects that we do not want to put in a message—like images, audio, video, etc.
- Write the large object to a file in object store
- Put a link to the object in the message

The End

Publisher-Subscriber Model

Concepts

Production System

- Production system has important events
- Numerous analytical systems want to know about these events as soon as they happen
- Production system is very important—we want minimal performance impact

Nightly Backups

- Every night, we take a backup using a storage layer mirror
  - No downtime
- Backup can be restored to a reporting database and data extracted into analytical systems
- Issue
  - Up to 24 hours stale
  - We want fresh data as soon as it happens

Enterprise Message Queues

- Production system writes messages to a topic every time an important event happens
- Minimal impact to production system
- Other systems can get the message soon after it is written to the topic

Publisher-Subscriber Model

- Publish once to a topic (or topics)
- Enterprise message queue system buffers, stores, and distributes the message for us
- Subscribers subscribe to a topic and read the messages

Subscribers

- Subscribers subscribe to the topic
- Unlimited number of subscribers to a topic
- Each subscriber must keep up with last message ID they have read from the topic

Message Deletion

- When a subscriber reads a message from a topic, the message is not deleted.
- The message is preserved so other subscribers can read it.
- Topics have a specified retention time.
  - 10 days, 30 days, 60 days, 90 days, etc.
- Messages are deleted when the specified retention time passes.

Side Effect

- One good side effect of the message retention in topics is that we can often pull data that is weeks old from topics if we cannot get it elsewhere
- Great for emergency, unplanned, ad hoc analytics

The End

Publisher-Subscriber Model

Business Cases

POS Sales Messages

- Create several topics
  - A topic for all POS (point of sale) sales
  - A topic for region POS sales
  - A topic for store POS sales
  - A topic for store department POS sales
  - A topic for each POS terminal’s sales

POS Sales Messages (cont.)

- After each sales transaction is completed, we write a message to the list of topics
  - Sales amount, tax amount, line items, product ID, quantity, etc.
- Systems can subscribe and read any of the topics to get real-time sales data at several levels
- Delete messages after they are X days old

Airline Boarding Pass Scanner

- Create several topics
  - A topic for all boarding pass scans
  - A topic for country boarding pass scans
  - A topic for region boarding pass scans
  - A topic for airport boarding pass scans
  - A topic for flight boarding pass scans
  - A topic for O/D (origin, destination) boarding pass scans

Airline Boarding Pass Scanner (cont.)

- After each boarding pass is scanned, the scanner computer will write a message to the list of topics
  - Customer, flight, date, origin, destination, etc.
- Systems can subscribe and read any of the topics to get real-time boarding pass scan data at several levels
- Delete messages after they are X days old

Images/Video

- For security reasons, we want to include an image (or a video) of the customer from the POS sale or boarding pass scan
- Probably too big to put in a message
- Place an image or video ID in the message
- We will save the image (or video) locally
- During off-peak times, we copy the image (or video) to the object store which can be retrieved using the ID in the message

The End

Producer-Consumer Model

Concepts

Read and Delete

- In the publisher-subscriber model, subscribers do not delete messages after they read them.
- Messages have a retention period specified and are deleted when the retention period expires.
- Consider the case where we want to read a message and delete it.

Queues

- Data structure that is a list of messages
- Messages inserted on one end
- Messages read and deleted on the other end
- FIFO (first in, first out)

Queuing Systems

- Queuing systems use queues to processes events in the order they come in
- One or more systems (or humans) write messages to the queue
- One or more systems (or humans) read and delete messages from the queue
- Ensures that no message is read more than once
- Queuing systems pre-date computers

Producer-Consumer Model

- Supports queueing systems
- Producers
  - One or more processes that write messages to a topic
- Consumers
  - One or more processes that read and delete messages from the topic
- Ensures that no message is read more than once

Scales Up Naturally

- Queuing systems naturally scale up
- Add more producers
- Add more consumers

The End

Producer-Consumer Model

Business Cases

Refunds Review, Part I

- Big box store allows refunds
- Multiple customer service terminals process refunds
- All refunds need a quick review by external AI system to detect fraud
- A random sample of refunds needs to go to third party independent auditors for auditing purposes

Refunds Review, Part II

- Create topic to request AI system review
  - Producer: terminal (multiple)
  - Consumer: AI system (multiple processes)
- Create topic to receive response
  - Specific to each terminal
  - Producer: AI system
  - Consumer: terminal

Refunds Review, Part III

- Create topic for auditing sample
  - Producer: terminal (multiple)
  - Consumer: third-party auditing system (multiple processes)

Home Loan Processing

- Initial pre-qualification
  - Customer intake (topic)
  - Initial credit check (topic)
  - Short form application review (topic)
  - AI module generates pre-qualification (topic)
  - Human review of pre-qualification for borderline cases (topic)

Home Loan Processing (cont.)

- Full application
  - Assist customer with application, gathering documents (topic)
  - Full application review, make sure ready for underwriting (topic)
- Underwriting
  - AI module recommends approval/disapproval (topic)
  - Human final approval based on AI recommendation (topic)

Airline Frequent Flyers

- Customers make reservations
- We are not sure they will actually take a flight until they get on an aircraft
- Boarding pass scan
  - 99.9% probability they are taking the flight
  - Grant frequent flyer credit ASAP (as soon as possible) for customer service purposes
  - Make sure we only grant credit once

Airline Frequent Flyers (cont.)

- Topic
  - Producer: scanner
  - Consumer: frequent flyer processes
- Compare and contrast this to our boarding pass scanner publisher-subscriber business case

Urgent

- Some requests may be urgent
- May want to create two topics
  - One for normal requests
  - One for urgent requests
- Or several topics with different priorities
- Some producer-consumer systems allow messages to have priorities

The End

Streaming Model

Concepts

Streaming Data

- Typically comes at a lightning-fast pace
- Do not read a message more than once
- Do not lose any messages
- Multiple processes to read the streaming data

Streaming Model

- Turns out enterprise message queues are a reasonable fit for the streaming model
- Early days
  - Use the publisher-subscriber model
    - Good for a single subscriber
    - Would not scale out as we would read messages more than once
  - Use the producer-consumer model
    - Allows multiple consumers
    - Would scale out
    - Not an exact fit

Streaming Model (cont.)

- Software designed specifically for streaming now available

Big Data Architecture

- Lambda architecture
  - Speed layer: immediate analytics for streaming data
  - Batch layer: store steaming data for later use
  - Serving layer: present stored data and/or analytical results
- Kappa architecture
  - Speed layer only

Object Store

- Object store is a good place to dump streaming data
- More on this when we get to data lakes

The End

Streaming Model

Business Cases

Social Media

- Messages
- Postings
- Etc.
- Huge volumes at high velocity
- Special purpose streaming software and scale up needed

Railroad Track Sensors

- Railroad has thousands of miles of track
- Sensors
  - Throughout the track
  - Transmit data in real time using cell phone network
  - Remote areas
  - Thousands of sensors
  - Stream data

Railroad Track Sensors (cont.)

- AI system
  - Read sensor data
  - Determine any anomalies in track
  - Immediately notify of any dangers
- Batch system
  - Store sensor data for further deep analytics
  - If we have an incident, go back and see sensor data before accident
  - Make the AI system better

Factory Automation, Part I

Modern factory

- Smart machines—stream data
- Smart robots—stream data
- Sensors—stream data
- Images streamed
- Video streamed
- Etc.

Factory Automation, Part II

AI system

- Reads streaming data
- Determines if there are any anomalies in the factory
- Immediately notified of any danger
- Flow analytics—recommendations to throttle or speed up processes

Factory Automation, Part III

Batch system

- Store streaming data for further deep analytics
- If we have an incident, go back and see data before accident
- Make the AI system better
- Flow analytics at a deeper level

Air Traffic Control, Part I

- Streaming data from all aircraft
  - Location using GPS
  - Speed
  - Direction
  - Rate of climb/descent
  - Turbulence from onboard sensors
- Current weather data
- Weather forecasts

Air Traffic Control, Part II

- AI system
  - Real-time assurance that no aircraft are on collision course
  - Real-time mapping of turbulence
  - Suggest routes for aircraft around weather/turbulence
  - Update arrival times for aircraft

Air Traffic Control, Part III

- Batch system
  - Deeper analytics
    - Turbulence
    - Weather
  - Help AI system perform better
  - See how well human air traffic control routers are doing

The End

Data Lakes

Concepts

Old Days

- Data not seen as a commodity
- Used to purge production systems of data
  - Two years old
- Purged data was lost forever
- Data storage was expensive

Data as a Commodity

- In the data science era, data is seen as a commodity
- We want to keep data just in case it ever proves useful
- Getting cheaper and cheaper to store data

Analogy of a City Water Supply

- Water is a precious commodity that is necessary for life.
- Lakes store dirty, unprocessed water.
- Lake storage is cheap.
- Dirty water is taken from lakes and processed into clean water.
- Processing water and storing clean water is expensive.

Data Lakes 1.0, Part I

- Original data lakes followed the city water supply analogy
- Data is a precious commodity—need to store it in case we ever need it
- Never purge data
- Data lake
  - Store unprocessed, raw data in the data lake
  - Storage is cheap

Data Lakes 1.0, Part II

- If we ever need the data:
  - We pull it from the data lake
  - Process it
  - Store it in a proper database
  - More expensive

Data Lakes 1.0, Part III

- E**L**T shift from traditional E**T**L
  - Extract
  - Load into data lake
  - Hold in data lake until needed
  - When needed, translate and load into proper database

Software Tools Get Better

- Processing data from the data lake into a proper database is a time-consuming and expensive process
- Opportunity for software tool vendors
- Build tools that can work with unprocessed data in data lakes
- Skip processing
- Skip loading into a proper database

Data Lakes 2.0

- Dump raw data into data lake
  - Typically, object store in modern era
- Perform analytics in place on the raw data using modern software tools
- Minimal translation of data
- Keeps working better and better
- Keeps getting cheaper and cheaper
- Serverless SQL—we will cover in later section

Security and Privacy Issues

- Raw data may contain sensitive data
- If we do not process it, how will we know?
- Possible someone may see sensitive data they should not see
- Possible privacy law violations
  - HIPAA, FERPA, etc.

The End

Data Lakes

Business Cases

Publisher-Subscriber Strategies

- Publisher-subscriber topics get purged when the data retention time has expired.
- At a minimum, we should dump the data into a data lake prior to it getting purged.
  - Ensures that we do not lose that data for future analytics

Publisher-Subscriber Strategies (cont.)

- We may also want to dump data into a data lake in batches every X minutes.
- Users can read from topic.
- Users can read data from data lake (will be X minutes stale).
- This reduces load on enterprise message queue.

Producer-Consumer Strategies

- Producer-consumer messages get purged when they are read
- Consider dumping the data into a data lake after reading it
  - Have the producer also write a message to a publisher-subscriber topic in addition to the producer-consumer topic
  - Have the consumer write a message to a publisher-subscriber topic
- Same situation we just discussed for publisher-subscriber and data lakes

Streaming Strategy

- In the Lambda architecture batch layer, we store the streaming data for later analytics.
- Write the streaming data to a data lake.

Production Systems

- Some may not be using enterprise message queues.
  - Or have gaps
- Some may not dump data into the data warehouse.
  - Or have gaps
  - More about data warehousing later in this course
- Production systems typically back end to a database.
- Use the nightly database backup to dump data into a data lake.

Common Datasets, Part I

- Demographics
- Weather
- Economic data
- Housing data by address (home values)
- IRS data

Common Datasets, Part II

- Public domain datasets
- Purchased commercial datasets
- Government datasets (data.gov)

Common Datasets, Part III

- Useful throughout the company by various departments and systems
- Place common datasets into a data lake so entire company has access to them
- Serverless SQL (next section) can greatly enhance this concept

The End

Serverless SQL

Concepts

Revisit Software Tools for Data Lakes

Software tools for data lakes are getting better and better.

Serverless SQL

- SQL executed against raw files in a data lake stored in object store
- Allows us to use raw data files immediately
- Saves development time on E**T**L/E**L**T processes and database design processes
- Will not be as fast as a real database, but if performance is acceptable, a good solution
- Generally considered the best advancement in data lakes

Table Is a Directory of Files

- We are using files to store our database
- We do not want to update a file every time we get new data
- Solution
  - Create a directory for each table
  - Data in every file in the directory is considered data in the table
  - Allows us to batch data every X minutes by just creating a new file in the directory

CSV Files

- CSV file equates to a table structure
- Obviously will work best for serverless SQL
- Just add a CSV file to the table directory and it becomes part of the table

JSON Files

- Flat JSON will work well
- Nested JSON not so well—does not immediately translate to a table
- Solutions
  - Some tools can read nested JSON with varying degrees of success
  - Extract the JSON into CSV files mimicking tables

Schema on Read

- Traditional relational databases impose schema on write.
  - If we insert a row out of format, it will reject the row
- Serverless SQL uses schema on read.
  - Schema is not imposed until we actually read the data.
  - Data out of format will not generate errors until we try to read it.
- One advantage—multiple schemas can be imposed on the same data.

Schema Repositories

In serverless SQL, schema on read is stored separately from the data in a schema repository.

Enterprise Data Catalogs

- Taking the schema repository concept to the next level
- Create an enterprise data catalog
  - Each dataset listed
  - Schemas for each dataset
- Software tools can read the enterprise data catalog, find the schemas, find the data, and run serverless SQL against the dataset

Columnar File Formats

- Columnar file formats
  - Binary data structures carved into a file
  - Horizontal partitioning
  - Vertical partitioning
    - Aka wide column
    - Aka columnar
    - Column headers
    - Column compression

Columnar File Formats (cont.)

- Converting CSV files to a columnar file format can increase performance dramatically
- More work—must convert them to a columnar file format
- Lose the human readability of CSV
- Often worth it
- Can store in both CSV and columnar

The End

Serverless SQL

Business Cases

Data Lake

- All our data lake business cases can use serverless SQL
- Create the data lake in object store
- For each table equivalent:
  - Create a directory in object store
  - Dump file for each table into the directory
  - Create a schema (or schemas)
  - Add entry to the enterprise data catalogue

Data Lake (cont.)

- If anyone in our company needs access to the data, we can simply give them access using the enterprise data catalogue

Leveraging Existing Resources

Corporate reports

- Most report writing software can connect to serverless SQL using adapters.
- Leverage existing report writing personnel and software with our data lake.

Leveraging Existing Resources (cont.)

BI (business intelligence)

- Most BI tools can also connect to serverless SQL using adapters.
- Leverage existing BI personnel and software with our data lake.

Scaling Up

- Typically we only scale up to the next stage if performance is unacceptable at current stage
- Stages
  - Start with serverless SQL in CSV or JSON format
  - Consider breaking JSON into CSV
  - Consider converting to a columnar file format
  - Move from serverless SQL to relational database
  - Move from relational database to data warehouse platform
- Tools are available to help with each stage

Scaling Down

As hardware and software improve, it may be possible to scale down.

- Suppose today our dataset is too big to work with serverless SQL, so we move it to a relational database.
- Several years from now, if the dataset has not grown that much and if serverless SQL has gotten a lot better, it may be feasible to move back to serverless SQL.

The End