Big data ecosystem in linkedin

Science, Computer Science

Workflows are complex and built using a variety of available tools for the Hadoop ecosystem: Hive, Pig, and MapReduce are the three primary processing interfaces. Avro is used as the serialization format. The data stored on HDFS is processed by chained MapReduce jobs that form a workflow, a directed acyclic graph of dependencies. If the data for the day is incomplete, the system automatically fallsback. To manage these workflows, LinkedIn uses Azkaban, an open source workflow scheduler. Its characteristics are

- Azkaban supports collection of jobs which form a workflow. Each job runs as an individual process and can be chained together to create dependency graphs. Configuration and dependencies for jobs are maintained as files of simple key-value pairs.
- Azkaban allows scheduling of graphs while maintaining logs and statistics of previous executions. The system provides monitoring and restart capabilities. If a job fails, alerts are triggered and when problem is solved, failed subgraph needs to be restarted.
- Azkaban also provides resource locking, though this feature is often not used as most of our data is append only. The construction of a production workflow is as follows
- A researcher starts experimenting data through a script, trying to convert it into a form they need. If machine learning, data cleaning and feature engineering require the large time. Then this becomes an individual Azkaban job followed by the output of these jobs being joined.

- These features can be trained into a model and we have packaged jobs to do this.
- The researcher can add or modifies these features, they configure Azkaban to only execute the changed jobs and their dependencies; As workflows are iterated upon and mature, they naturally become more complex and Azkaban becomes more useful. LinkedIn maintains three Azkaban instances,
- Azkaban on the ETL grid manages the Hadoop load and is completely hidden from users of the Hadoop ecosystem.
- For the development and production environments, a researcher deploys their workflows on the Azkaban instance to test the output of their algorithms. Once tested, every workflow goes through a production review where basic integration tests are performed, and any necessary tuning is done.
- Post-review, the workflow is deployed onto the production Azkaban instance. The datasets and the tool suites are kept in sync across environments to allow easy reproduction of results. EGRESS The result of workflow are pushed into other systems, either back for online serving or as a derived data-set for further consumption. This job consists of a simple 1-line command for data deployment. There are three main mechanisms available depending on the needs of the application:
- Key-value: The derived data is accessed as an associative array or as a collection.
- Streams: The data is published as a change-log of data tuples.

- OLAP: The data is transformed into multi-dimensional cubes for online analytical queries.

KEY-VALUE: It is the most frequently used vehicle of transport from Hadoop at LinkedIn and is made possible using Voldemort. Tuples are grouped into logical stores which correspond to database tables. Each key is replicated to multiple nodes depending on the replication factor of its corresponding node is further split into partitions, with every key in a store mapped using consistent hashing to multiple partitions across nodes. Voldemort is a distributed key-value store akin to Amazon’s Dynamo with a simple get(key) and put (key, value) interface. It was built for read-write patters with its storage engine using MYSQL/Berkeley DB. It provides a pluggable nature which introduces storage engine tailored to HDFS. It is made up of chunk sets of index and data files. The index file is a compact structure containing a hash of the key followed by the offset to the corresponding value in the data file, with the entire file sorted by the hashed keys. The identity mappers in MapReduce job only emit the hash of the key “ replication factor” number of times, followed by a custom partitioner that routes the key to the reducer responsible for the chunk set.

The reducers finally receive the data sorted by the hash of the key and perform an append to the corresponding chunk set. This job supports producing multiple chunk sets per partition, thereby allowing one to tweak the number of reducers to increase parallelism. On the Voldemort side, configurable number of directories are maintained on a per-store basis with just one serving live requests while others acting as backups. After generating new chunk sets on Hadoop, Voldemort nodes pull their corresponding chunk sets in parallel into new directories. By adopting a pull methodology instead of push, Voldemort can throttle the data being fetched. A check is also performed with pre-generated checksums to verify integrity of pulled data.

After the pull operation has succeeded, the chunk set files of the current live directory are closed and the indexes in the new chunk sets are memory mapped, relying on the operating system's page cache. This " swap" operation runs in parallel across machines and takes a sub-millisecond amount of time. The last step, after the swap, is an asynchronous clean-up of older versions to maintain the number of backups. Maintaining multiple backup versions of the data per store aids in quick rollback to a good state in case of either data or underlying algorithm problems. This complete chunk generation, pull, and swap operation is abstracted into a single line Pig StoreFunc that is added by the user to the last job of their workflow. Key-value access is the most common form of egress from the Hadoop system at LinkedIn. We have been successfully running these clusters for the past 3 years, with over 200 stores in production. We have seen consistent 99th percentile latency on most stores to be below sub 10ms.

Streams

The second mechanism for derived data generated in Hadoop is as a stream back into Kafka. This is useful for applications that need a change log of the underlying data. This ability to publish data to Kafka is implemented as a Hadoop OutputFormat. Here, each MapReduce slot acts as a Kafka producer that emits messages, throttling as necessary to avoid overwhelming the

Kafka brokers. The driver verifies the schema to ensure backwards compatibility. An example script to push a stream of session-based page-views is shown below sessions = FOREACH pageviews GENERATE Sessionize(*); STORE sessions INTO ’kafka: //kafka-url’ USING Streams(’topic= SessionizedPageViewEvent’); OLAP: The final mechanism offered is multidimensional data (OLAP) for online queries. Queries allow the end user to look at the data by slicing and dicing across various dimensions OLAP solutions solve this problem by coupling the two required subsystems: cube generation and dynamic query serving At LinkedIn, we have developed a system called Avatara that solves this problem by moving the cube generation to a high throughput offline system and the query serving to a low latency system.

The resulting data cubes are to be served in the request/response loop of a website. The use cases at LinkedIn have a natural shard key thereby allowing the final large cube to be partitioned into multiple “ small cubes”. To generate cubes, Avatarauses Hadoop as offline engine to run user-defined joins and pre-aggregation steps on the data. To ease the development, we provide a simple configuration file driven API that automatically generates corresponding Azkaban job pipelines. The underlying format for these small cubes is multidimensional arrays where a tuple is combination of dimension and measure pairs. The output cubes are then bulk loaded into Voldemort as a read-only store for fast online serving. Due to its architecture, Avatara is not tied to Voldemort and can support other online serving systems. Because Avatara provides the ability to materialize at both the offline and online engine, the developer has option to choose faster response time or flexible query combinations.

Applications

Features of the LinkedIn depend on the data pipeline either explicitly or implicitly. Explicitly mean determine where the data is the product and implicitly refers to how the derived data is infused into the applications. LinkedIn application can be categorized into 3 types based on the application and they are

- Key–Value

  Key-value accessing uses Voldemort which is the most common Egress mechanism used from Hadoop. Some of techniques used are

- People You May Know

  This is used to find other members a user may know on the social network. This is a link prediction problem where the node and the edge features of one user in social graph is used to predict whether an invitation can occur between two unconnected nodes through a user who is known to both the user. Hadoop provides extraction tasks like determining the common connections, company and school overlap, geographical distance, similar ages, and many others. This system contains100 Azkaban tasks and several MapReduce jobs to build these recommendations every day.

- Collabarative Filltering

This technique shows relationships between pairs of items like “ people who did this also did that”. This type of computation was done only for member-to-member co-occurrence, but quickly grew to meet the needs of other entity types, including cross-typerecommendations. LinkedIn’s frontend framework as it contains LinkedIn’s base member activity tracking

- Skill Endrosment

  Endorsements are a light-weight mechanism where a member can connect to another member in their network for a skill, which then shows up on the endorsed member’s profile. This feature provides recommendations on who to endorse by suggesting lists of users. A workflow first determines whether the same skills exist across the member base. This is a deep information extraction problem, requiring deduplication, synonym detection.

- Jobs You May Be Intrested In

  This provides the user with the jobs which might be of his interest depending on the information which is filled while creating the user account for LinkedIn.

- Related Searches

  LinkedIn’s related search system builds on several signals and filters that capture several dimensions of relatedness across member search activity. Related search recommendations are the query in “ Hadoop” that provides relevant results to their queries.

- Streams

  Streams require push-based model of Kafka as the delivery mechanism. Both online and offline systems negotiate a predefined Kafka topic, with the offline system producing and the online system consuming the output.

- News Feeds Update

  News feed are generated by an online system. Updates are generated on a per-member basis based on interactions with other members in the ecosystem. As data processing is easy it provides quick prototyping and testing of updates.

- Email

  Email can be generated either online or offline. These mails can be activity driven which require online generation/delivery and the other mails are digest mails which are offline generated. The architecture of the email system is like the network update stream. The email system packages this data with the template into an email message that is eventually sent to the user.

- Relationship-Strength

  LinkedIn’s Social graph are scored by offline jobs which helps in determining the strong and weak relationships. Scores generated are publishes to a stream and read by the consumers to populate their indexes.

- OLAP Linked

In profiles provides various dimensions and views of company, school, group, geography, etc. Some of its application are

- Who’s Viewed My Profile? Linked

  In provides user with a facility to view the members who have viewed his/her profile. This is generated depending on the small cube computation based on the key of member id. The data is generated by aggregating the member profiles at different level and performing a join operation on the data set. The data cubes generated is loaded on to Voldemort read-only store followed by the real-time grouping/ordering to view the data based on the query.

- Who’s Viewed This Job?

  It provides a list of members who viewed a job by title, time or company. The job ID is the primary key with a similar workflow as “ Who’s Viewed My Profile? ” to generate a cube for 3 dimensions on job view.

Conclusion

Case study is about the end-to-end Hadoop-based analytics stack at LinkedIn. This case study helped in understanding the data flows into the offline system, the mechanism of workflows being executed, and the mechanisms available for sending data back to the online system.