Curso de extensão em Data Science

GERÊNCIA DE INFRAESTRUTURA PARA BIG DATA

Prof. Tiago Ferreto – tiago.ferreto@pucrs.br

ESCOLA POLITÉCNICA

HDFS – DATA INGESTION

Data Ingestion

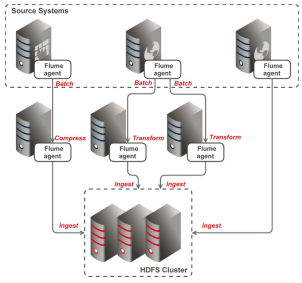
- How to capture data produced from source systems in real time?
  - Examples: web logs, databases, sensors
- HDFS standard interface is not practical enough for more complex scenarios
- Tools
  - Flume
  - Sqoop
  - HDFS RESTful interfaces
    - WebHDFS
    - HttpFS

Flume

- Hadoop ecosystem project
  - https://flume.apache.org/
- Developed originally by Cloudera
- Goal: capture, transform, and ingest data into HDFS using one or more agents
- Initial use case: capturing log files, or web logs from a web server, and routing them to HDFS as they are generated

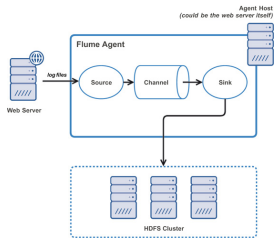
Flume Architecture

- Implemented using one or more agents
- Agents connect a data source to HDFS or another agent
- Agents can be chained together or be used in parallel (horizontal scalability or fault tolerance)
- Agents can perform in-flight data operations (e.g., compression, encryption, batching of events, etc)



Flume Agent Architecture

- Flume agents contain a sink, a source, and a channel



Source

- Indicates where the data is to be received from
- Examples
  - HTTP - Used to consume data from RESTful services using POST and GET methods
  - Syslog - Log protocol to capture system events
  - JMS - Java Message Service
  - Kafka - Popular open source messaging platform
  - Avro - Open source, cross platform data serialization framework for Hadoop
  - Twitter - Flume source that connects to Twitter's Streaming API to continuously download tweets

```
agent1.sources = source1
agent1.sources.source1.type = exec
agent1.sources.source1.command = tail -F /tmp/events
```


Sink

- Specifies where to send data (usually HDFS, but can also be another agent or another filesystem, such as S3)
- Examples
  - HDFS - Most common sink used to ingest data into HDFS
  - Hive - Ingests data into HDFS while updating the Hive partitions and the Hive metastore
  - Hbase - NoSQL data store built on HDFS
  - Kafka - Popular open source messaging platform
  - Solr - Open source search platform

```
agent1.sinks = sink1
agent1.sinks.sink1.type = hdfs
agent1.sinks.sink1.hdfs.path = /flume/events
agent1.sinks.sink1.hdfs.filePrefix = events-
agent1.sinks.sink1.hdfs.round = true
agent1.sinks.sink1.hdfs.roundValue = 10
agent1.sinks.sink1.hdfs.roundUnit = minute
```


Channel

- Queue between agent's source and sink
- Flume implements a transactional architecture for reliability (supports rollback and retry operations)
- Possible configurations: **in-memory** or **durable**
- Durable channels use persistent storage (disk) to maintain state (transactional integrity)
- Examples of durable channels
  - File Channel
  - JDBC Channel
  - Kafka Channel

```
agent1.channels = channel1
agent1.channels.channel1.type = memory
agent1.channels.channel1.capacity = 100
agent1.channels.channel1.transactionCapacity = 100
agent1.channels.channel1.byteCapacityBufferPercentage = 20
```

FLUME - HANDS-ON

Sqoop

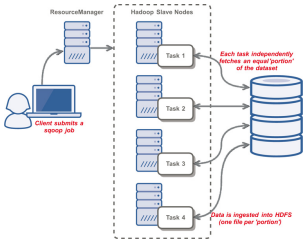
- Top-level Apache project
  - http://sqoop.apache.org/
- Sqoop → sql-to-Hadoop
- Originally developed by Cloudera
- Goal: source data from a relational database and ingest this data into files (typically delimited files) in HDFS
- Can also be used to send data from Hadoop to a relational database
- Integrates with Hive (SQL abstraction to MapReduce)

Sqoop common operations

- Listing databases and tables on a database system
- Importing a single table from a database system, including
  - Specifying which columns to import
  - Specifying which rows to import using a WHERE clause
- Importing data from one or more tables using a SELECT statement
- Incremental imports from a table on a database system (importing only what has changed since a known previous state)
- Exporting of data from HDFS to a table on a remote database system

Sqoop operation

- Implemented using MapReduce (Map-only)
- Steps (import operation)
  1. Connect to the database system using JDBC or a customer connector
  2. Examine the table to be imported
  3. Create a Java class to represent the structure (schema) for the specified table. This class can then be reused for future import operations.
  4. Use YARN to execute a Map-only MapReduce job with a specified number of tasks (mappers) to connect to the database system and import data from the specified table in parallel. The default number of parallel tasks is 4.



Sqoop commands

```
Available commands:
  codegen            Generate code to interact with database records
  create-hive-table  Import a table definition into Hive
  eval               Evaluate a SQL statement and display the results
  export             Export an HDFS directory to a database table
  help               List available commands
  import             Import a table from a database to HDFS
  import-all-tables  Import tables from a database to HDFS
  import-mainframe   Import datasets from a mainframe server to HDFS
  job                Work with saved jobs
  list-databases     List available databases on a server
  list-tables        List available tables in a database
  merge              Merge results of incremental imports
  metastore          Run a standalone Sqoop metastore
  version            Display version information
```


Sqoop example

- Importing all tables from a MySQL database named “mydb”

```
$ sqoop import-all-tables \
--username javen \
--password ***** \
--connect jdbc:mysql://mydbserver.local/mydb
```


Sqoop2 (Sqoop-as-a-Service)

- Server-based implementation of Sqoop
  - Server hosts the application and connectors, and a lightweight client, web UI, or API connects to the server to submit a Sqoop request
  - Server executes the request on the client’s behalf
- Advantages
  - Centralized management of security, as credentials are centrally stored and managed
  - Capability to restrict or control the number of connections to a database system
  - Capability to implement network isolation between client systems and subnets and database systems
  - Eliminates the need to manage software on client systems (e.g., custom connectors, binaries, etc.)

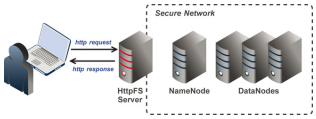
WebHDFS

- WebHDFS provides RESTful access to HDFS using HTTP or HTTPS
- Supports read and write operations
- Command line utilities such as wget or curl can be used to access HDFS
- Example
  - `$ curl -i -L http://namenode:50070/webhdfs/v1/data/file.txt?op=OPEN`
- Limitations
  - WebHDFS does not support High Availability (HA) HDFS implementations
  - Clients must be able to access every DataNode in the cluster

WEBHDFS - HANDS-ON

HttpFS

- HttpFS provides RESTful access via a service
- Solution is more scalable, supporting HA HDFS implementations and not requiring direct client accessibility to DataNodes in the cluster
- HttpFS server acts as a proxy accepting REST requests from clients and submitting them to HDFS on the clients' behalf



Gerência de Infraestrutura para Big Data – Prof. Tiago Ferreto – PUCRS

Data Ingestion – Considerations

- When a file is being written to HDFS, the file initially appears to clients as a zero byte file in the incoming directory with a ._COPYING_ suffix
- Data is committed to the file in block size increments (typically 128MB)
  - File size increases in 128MB increments
- The file is visible in the target HDFS directory and can be read (and therefore used as input for processing) while the file is not yet complete
  - Problems can happen processing incomplete files
- Recommendation
  - Use a special directory (e.g., ./incoming) to ingest data
  - Once the write operation is complete, move the file to an accessible directory (e.g., ./ingested)
    - Move operation is (almost) instantaneous – only requires a metadata operation