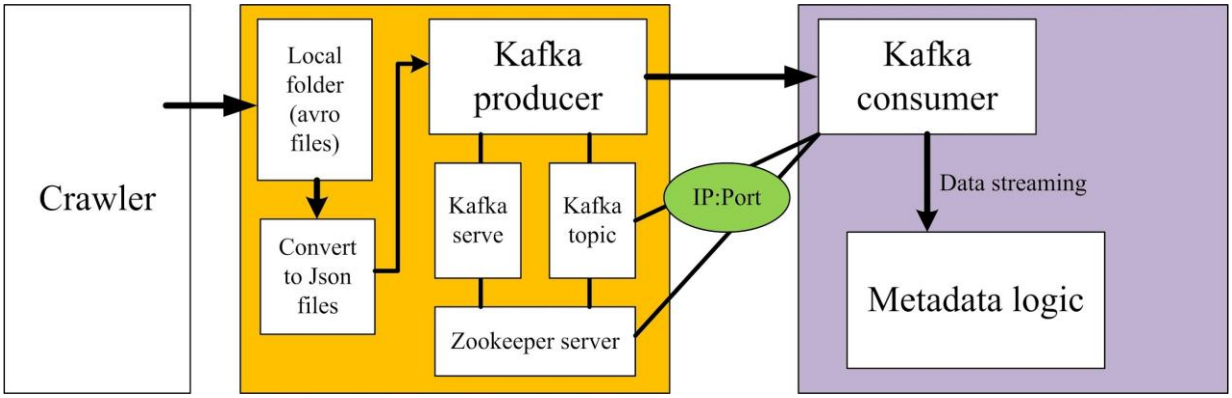
CS286A Metadata Data Mover Architecture

Jialiang Gu, Haoyu Chen

Overview

Data Mover acts like a intermediate station for Crawler and Metadata. It receive the files from the Crawler, stores it if necessary, and send them to the proper destinations (i.e. metadata). The basic task for us is to ensure the safe, fast, reliable data movement.
In order to achieve these objectives, we use Kafka to implement the framework. There are roughly two parts of our system: Kafka producer and Kafka consumer. The Kafka producer sends the local file to the buffer. When Kafka consumer is online, it will fetch the data in buffer. Kafka consumer receives file corresponding to one json object at a time from Kafka producer, and commit them to Metadata logic.

Architecture



Data Flow

The Data Mover runs like the process below:
When Crawler is ready to save some file to Metadata, it will send the files to Data Mover machine by SCP. The destination, we will keep these files is: dataMover/kafka/incoming. These files are in the tar.gz format. After Data Mover finished sending the file, it will call our bash file (i.e. dataMover/kafka/bin/dataMoverJob.sh). This bash file will unzip the received files, and convert them into Json format from avro format. After getting one Json file, we will send it by kafka and delete it in the local directory.
Kafka consumer runs on the repo side. As soon as Kafka consumer received one Json file, it calls the repo's API to commit this Json object into the repo's database.

Structures

In this part, we explain the details of our code. In the dataMover/kafka/bin/dataMoverJob.sh, there is a for loop, which find each avro file in the folder, convert it to json (i.e. "java -jar

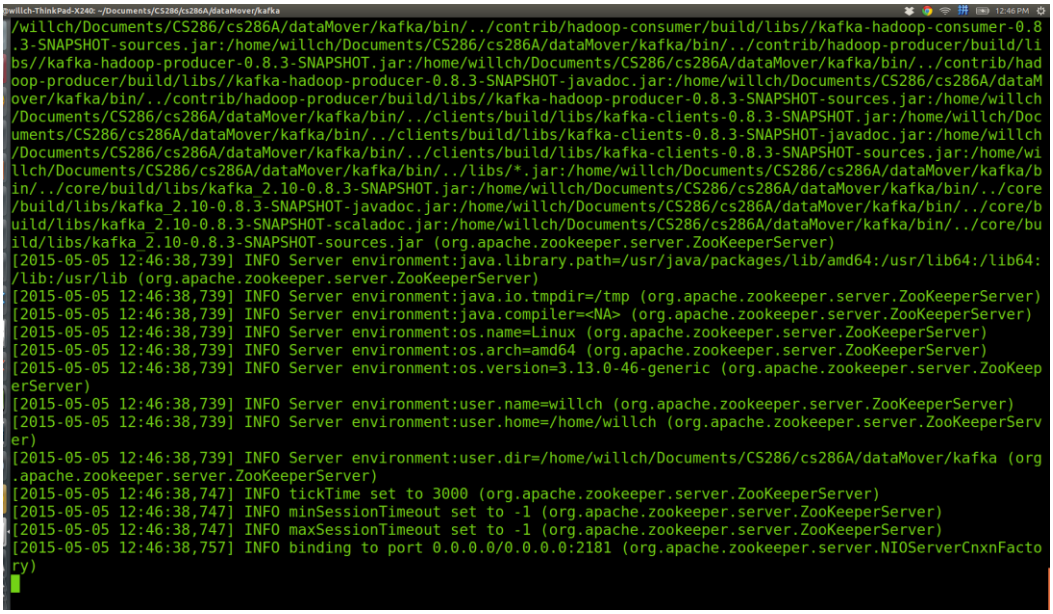
../bin/avro-tools-1.7.7.jar tojson $f > $currFileName.json"), and send it by Kafka producer (i.e. "../bin/kafka-file-producer.sh --broker-list localhost:9092 --topic test --input-file $currFileName.json"). Which will call the source file "/dataMover/kafka/core/src/main/scala/kafka/tools/FileProducer.scala", this class will receive a filename from command line and transfer its content to network, waiting for the consumer to fetch.

For the kafka consumer, the core part is in the "/dataMover/kafka/core/src/main/scala/kafka/tools/FileConsumer.scala". After we run the kafka consumer, we will new a object at line 173 (i.e. "var repo : MetadataRepo = new MetadataRepo("54.69.1.154")"). Then for each line we read (i.e. a json file), we parse it to a map. And call the repo's API repo.commit to store the message (i.e. "repo.commit(namespace.asInstanceOf[String], filename.asInstanceOf[String], str.asInstanceOf[String], timestamp.asInstanceOf[String].toLong)").

In order to run the repo's API, we need to put their files in "dataMover/kafka/core/src/main/scala/kafka/tools" (right now we just put their MetadataRepo.java into the folder). Since, in the MetadataRepo.java, they includes the online package from MogoDB, we need to revise the gradle files to include this package in our kafka system too. Thus, we add one line in build.gradle to include the MogoDB online package (i.e. line 211: "compile 'org.mongodb:mongo-java-driver:3.0.0'"). If repo group changes their files, they need to update the MetadataRepo.java in our folder. If repo group imports more packages, you need to add the import path in the build.gradle as well.

Example
(In this part, one simple example is shown:)

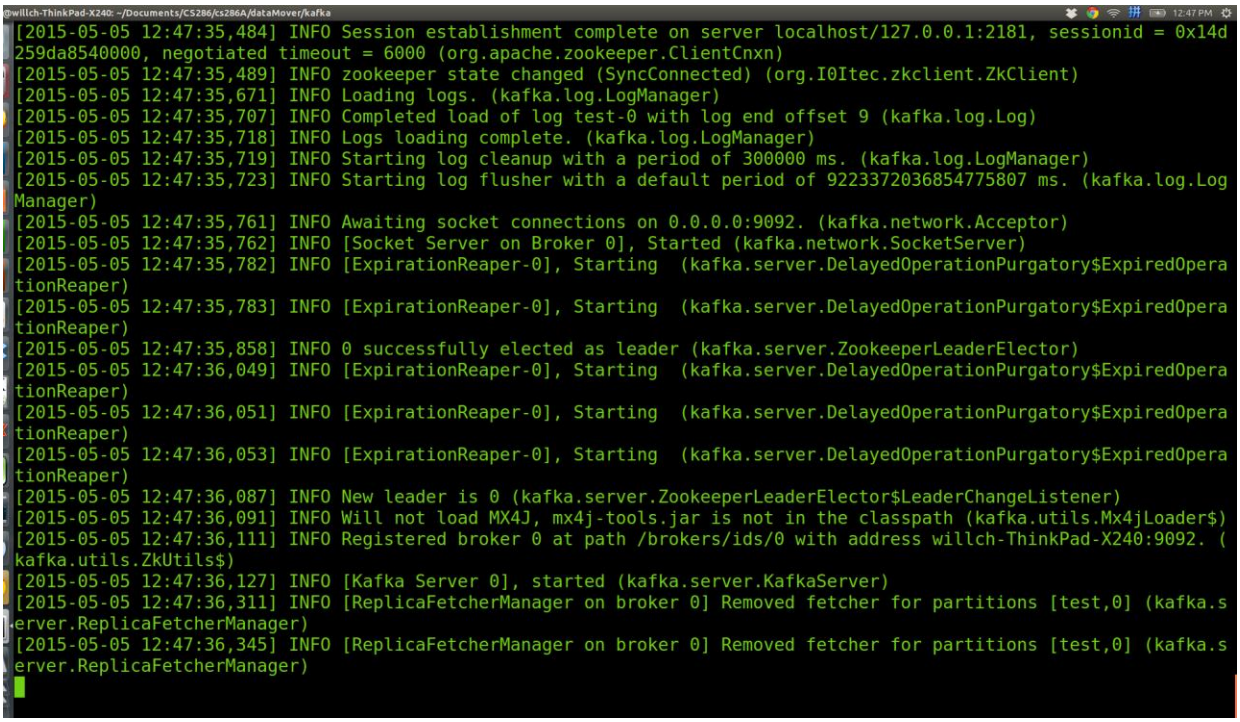
SETUP:
For the producer side:
First, open a terminal, and change the directory to the kafka. Run:
`bin/zookeeper-server-start.sh config/zookeeper.properties`
you will see:



Then open another terminal, direct to the kafka repo, and run:

`bin/kafka-server-start.sh config/server.properties`

you will see:



Then open the third terminal, direct to the kafka repo, and run:

`bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test`

you will see:

```
willch@willch-ThinkPad-X240:~/Documents/CS286/cs286A/dataMover/kafka$ bin/kafka-
topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partition
s 1 --topic test
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/
kafka/core/build/dependant-libs-2.10.4/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/S
taticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/
kafka/core/build/dependant-libs-2.10.4/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/S
taticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Created topic "test".
```

For the consumer side:

Open a terminal, direct to the kafka, and run:

`bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test`

you will see:

```
willch@willch-ThinkPad-X240: ~/Documents/CS286/cs286A/dataMover/kafka
willch@willch-ThinkPad-X240:~/Documents/CS286/cs286A/dataMover/kafka$ bin/kafka-file-consumer.sh --zookeeper localh
ost:2181 --topic test --output-file $LOG_FILE
Option ['output-file'] requires an argument
willch@willch-ThinkPad-X240:~/Documents/CS286/cs286A/dataMover/kafka$ bin/kafka-console-consumer.sh --zookeeper loc
alhost:2181 --topic test
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/kafka/core/build/dependant-libs-2.1
0.4/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/kafka/core/build/dependant-libs-2.1
0.4/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
```

PASS MESSAGE

For the producer side:

at the dataMover repo run:

`./inotifywait.sh`

you will see:

```
willch@willch-ThinkPad-X240:~/Documents/CS286/cs286A/dataMover$ ./inotifywait.sh
Setting up watches.
Watches established.
```

After a new compressed file is scp to the income folder, it will send it automatically:

```
willch@willch-ThinkPad-X240:~/Documents/CS286/cs286A/dataMover$ ./inotifywait.sh
Setting up watches.
Watches established.
kafka/incoming/20150502sdsnd.tar.gz
curr dir is kafka/incoming/20150502sdsnd
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/kafka/core/build/dependant-libs-2.1
0.4/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/kafka/core/build/dependant-libs-2.1
0.4/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/kafka/core/build/dependant-libs-2.1
0.4/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/kafka/core/build/dependant-libs-2.1
0.4/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/kafka/core/build/dependant-libs-2.1
0.4/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/willch/Documents/CS286/cs286A/dataMover/kafka/core/build/dependant-libs-2.1
0.4/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
```

And at the consumer side, the message will be received and printed on the screen:

```
n@willch-ThinkPad-X240: ~/Documents/CS286/cs286A/dataMover/kafka                                    12:51 PM
{"filename":"/home/ian/cs286A/crawler/gobblin/test_source/dir1/test3.csv","timestamp":"1430539320025","namespace":"
default","lastAccessTime":"2015-04-29T07:09:22Z","lastModifiedTime":"2015-04-25T07:21:59Z","creationTime":"2015-04-
25T07:21:59Z","size":"71","isSymbolicLink":"false","isRegularFile":"true","isOther":"false","isDirectory":"false"}
{"filename":"/home/ian/cs286A/crawler/gobblin/test_source/test2.txt","timestamp":"1430539320026","namespace":"defau
lt","lastAccessTime":"2015-04-29T07:09:22Z","lastModifiedTime":"2015-04-21T08:05:06Z","creationTime":"2015-04-21T08
:05:06Z","size":"29","isSymbolicLink":"false","isRegularFile":"true","isOther":"false","isDirectory":"false"}
Some(Map(lastModifiedTime -> 2015-04-21T08:05:06Z, isOther -> false, isSymbolicLink -> false, timestamp -> 14305393
20026, size -> 29, isRegularFile -> true, isDirectory -> false, filename -> /home/ian/cs286A/crawler/gobblin/test_s
ource/test2.txt, lastAccessTime -> 2015-04-29T07:09:22Z, namespace -> default, creationTime -> 2015-04-21T08:05:06Z
))
Map(lastModifiedTime -> 2015-04-21T08:05:06Z, isOther -> false, isSymbolicLink -> false, timestamp -> 1430539320026
, size -> 29, isRegularFile -> true, isDirectory -> false, filename -> /home/ian/cs286A/crawler/gobblin/test_source
/test2.txt, lastAccessTime -> 2015-04-29T07:09:22Z, namespace -> default, creationTime -> 2015-04-21T08:05:06Z)
data
{"filename":"/home/ian/cs286A/crawler/gobblin/test_source/test2.txt","timestamp":"1430539320026","namespace":"defau
lt","lastAccessTime":"2015-04-29T07:09:22Z","lastModifiedTime":"2015-04-21T08:05:06Z","creationTime":"2015-04-21T08
:05:06Z","size":"29","isSymbolicLink":"false","isRegularFile":"true","isOther":"false","isDirectory":"false"}
{"filename":"/home/ian/cs286A/crawler/gobblin/test_source/test1.txt","timestamp":"1430539320027","namespace":"defau
lt","lastAccessTime":"2015-04-29T07:09:22Z","lastModifiedTime":"2015-04-21T08:05:06Z","creationTime":"2015-04-21T08
:05:06Z","size":"28","isSymbolicLink":"false","isRegularFile":"true","isOther":"false","isDirectory":"false"}
Some(Map(lastModifiedTime -> 2015-04-21T08:05:06Z, isOther -> false, isSymbolicLink -> false, timestamp -> 14305393
20027, size -> 28, isRegularFile -> true, isDirectory -> false, filename -> /home/ian/cs286A/crawler/gobblin/test_s
ource/test1.txt, lastAccessTime -> 2015-04-29T07:09:22Z, namespace -> default, creationTime -> 2015-04-21T08:05:06Z
))
Map(lastModifiedTime -> 2015-04-21T08:05:06Z, isOther -> false, isSymbolicLink -> false, timestamp -> 1430539320027
, size -> 28, isRegularFile -> true, isDirectory -> false, filename -> /home/ian/cs286A/crawler/gobblin/test_source
/test1.txt, lastAccessTime -> 2015-04-29T07:09:22Z, namespace -> default, creationTime -> 2015-04-21T08:05:06Z)
data
{"filename":"/home/ian/cs286A/crawler/gobblin/test_source/test1.txt","timestamp":"1430539320027","namespace":"defau
lt","lastAccessTime":"2015-04-29T07:09:22Z","lastModifiedTime":"2015-04-21T08:05:06Z","creationTime":"2015-04-21T08
:05:06Z","size":"28","isSymbolicLink":"false","isRegularFile":"true","isOther":"false","isDirectory":"false"}
```

NOTICE:

In this example, we only print out the received message. If you want to call repo's API and put message into the database, you need to run:

```
bin/kafka-file-consumer.sh --zookeeper localhost:2181 --
topic test --output-file $LOG_FILE
```