Integrating Hadoop and Elasticsearch - Part 1 - Loading into and Querying Elasticsearch from Apache Hive

Posted by Prasanth Kothuri on Wednesday, 30 March 2016

Introduction

As more and more organisations are deploying Hadoop and Elasticsearch in tandem to satisfy batch analytics, real-time analytics and monitoring requirements, the need for tigher integration between Hadoop and Elasticsearch has never been more important. In this series of blogposts we look at how these two distributed systems can be tightly integrated and how each of them can exploit the feaures of the other system to achieve ever demanding analytics and monitoring needs.

Hadoop

Hadoop is a broad ecosystem of tools that support bulk ingestion of data, efficient data formats and distributed processing of large data sets across clusters of commodity hardware. It allows for many ways (batch, interactive, SQL) to interact with the data that is stored in HDFS.

ElasticSearch

Elasticsearch together with Logstash for log tailing and Kibana for visualisation is gaining a lot of momentum as it is fairly easy to setup & get started and its near real-time search and aggregation capabilities covers lots of use cases in the area of web analytics and monitoring (log & metric analysis).

Integrating Apache Hive with Elasticsearch

We first look at the required setup and configuration to integrate Apache Hive with Elasticsearch and then go through scenarios of querying data between Hadoop and Elastic

Setup and Configuration

Following steps guide you on the required setup and configuration both for pilot and production purposes

Pilot

If you are getting started with testing ES-Hadoop connector then the following setup is suitable

- Download and Copy Elastic-Hadoop connector to /tmp on HDFS

```
wget -P /tmp http://download.elastic.co/hadoop/elasticsearch-hadoop-2.2.0.zip
unzip /tmp/elasticsearch-hadoop-2.2.0.zip -d /tmp
hdfs dfs -copyFromLocal /tmp/elasticsearch-hadoop-2.2.0/dist/elasticsearch-ha
```

- Connecting to beehive

```
beeline -u "jdbc:hive2://hiveserver2:10000/default;principal=hive/hiveserver2
```

- Add JAR manually (you need to do to for every beehive session)

```
add JAR hdfs://hiveserver2.cern.ch:8020/tmp/elasticsearch-hadoop-2.2.0.jar;
list JAR;
```

Please note that hiveserver2 in many cases is Hadoop namenode, you can also find this information from /etc/hive/conf/hive-site.xml

Production

Once you have carried out satisfactory testing and would like to use then you should deploy as below

- You will need to copy the jar file into $HIVE_HOME/lib (e.g /usr/lib/hive/lib) on all nodes in the hadoop cluster and restart the HiveServer2

OR

- Register the JAR through hive.aux.jars.path parameter of hive-site.xml and restart the hiveserver2

```
<property>
  <name>hive.aux.jars.path</name>
  <value>/path/to/elasticsearch-hadoop.jar</value>
</property>
```

Now that the required setup is completed, we look at the following scenarios

1. Querying data from Elasticsearch index
2. Writing data to Elasticsearch index
3. Offloading / archiving Elasticsearch index to Apache hive table

Querying data from Elasticsearch index

If you are able to query ElasticSearch from Hive this is in essence giving you an SQL interface to ElasticSearch data with out having to learn lucene query language.
This you achieve by creating an external table in Hive on top of ElasticSearch index using the elastic-hadoop driver we configured above

```
drop table fmem;

CREATE EXTERNAL TABLE fmem (
    dt      timestamp,
    server    STRING,
    mem   bigint)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES('es.nodes' = 'ela1', 'es.resource' = 'fmem/log', 'es.query' = '?q=*'
```

TBLPROPERTIES clause is used to specify the configuration parameters, I discuss below the important (essential) configuration parameters

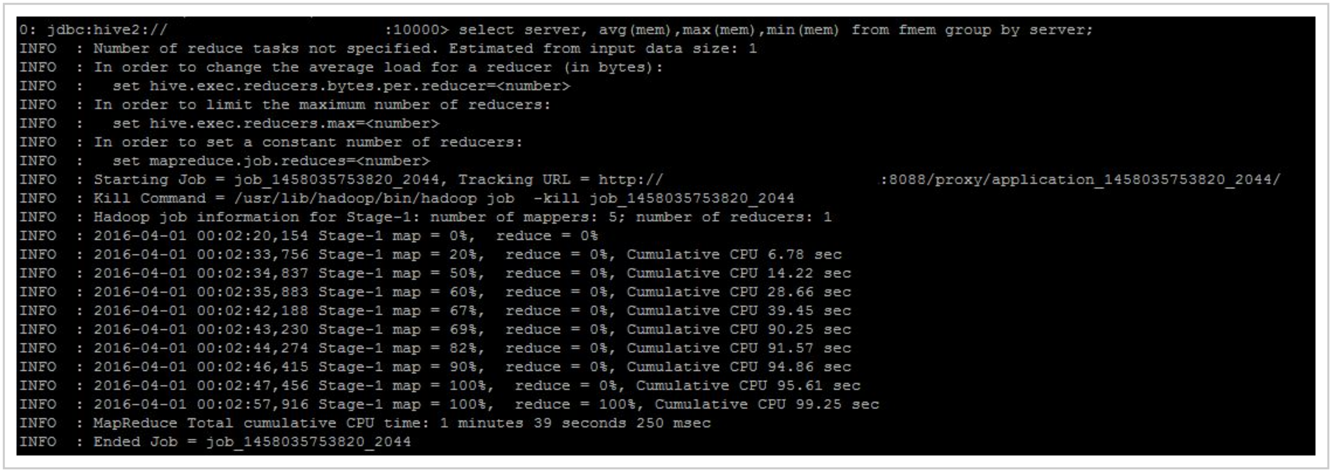
```
es.nodes - list of es nodes, specifying one is enough as the other nodes in the cl
es.port - only need to mention if you are using non default port
es.resource - es index
es.query - query that is applied to es.resource, this allows you to externalize a
es.net.http.auth.user - es username
es.net.http.auth.pass - es password
```

And finally you can also change the fields names using es.mapping.names property (e.g 'es.mapping.names' = 'date:dt , memory:mem')

Once the ElasticSearch index is externalized in the form of Hive table, you can apply the full force of Hive query language to perform batch analytics

```
-- simple query to check if we are able to see data from ES index
select * from fmem limit 10;


-- we are collecting flume agent memory utilisation from our server farm over seve
select server, avg(mem),max(mem),min(mem) from fmem group by server;
```



```
0: jdbc:hive2://[redacted]:10000> select server, avg(mem),max(mem),min(mem) from fmem group by server;
INFO  : Number of reduce tasks not specified. Estimated from input data size: 1
INFO  : In order to change the average load for a reducer (in bytes):
INFO  :   set hive.exec.reducers.bytes.per.reducer=<number>
INFO  : In order to limit the maximum number of reducers:
INFO  :   set hive.exec.reducers.max=<number>
INFO  : In order to set a constant number of reducers:
INFO  :   set mapreduce.job.reduces=<number>
INFO  : Starting Job = job_1458035753820_2044, Tracking URL = http://[redacted]:8088/proxy/application_1458035753820_2044/
INFO  : Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1458035753820_2044
INFO  : Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 1
INFO  : 2016-04-01 00:02:20,154 Stage-1 map = 0%,  reduce = 0%
INFO  : 2016-04-01 00:02:33,756 Stage-1 map = 20%,  reduce = 0%, Cumulative CPU 6.78 sec
INFO  : 2016-04-01 00:02:34,837 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 14.22 sec
INFO  : 2016-04-01 00:02:35,883 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 28.66 sec
INFO  : 2016-04-01 00:02:42,188 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 39.45 sec
INFO  : 2016-04-01 00:02:43,230 Stage-1 map = 69%,  reduce = 0%, Cumulative CPU 90.25 sec
INFO  : 2016-04-01 00:02:44,274 Stage-1 map = 82%,  reduce = 0%, Cumulative CPU 91.57 sec
INFO  : 2016-04-01 00:02:46,415 Stage-1 map = 90%,  reduce = 0%, Cumulative CPU 94.86 sec
INFO  : 2016-04-01 00:02:47,456 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 95.61 sec
INFO  : 2016-04-01 00:02:57,916 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 99.25 sec
INFO  : MapReduce Total cumulative CPU time: 1 minutes 39 seconds 250 msec
INFO  : Ended Job = job_1458035753820_2044
```

As you can see in the above screen shot Hive start map-reduce to execute the query.

Inserting data to Elasticsearch index

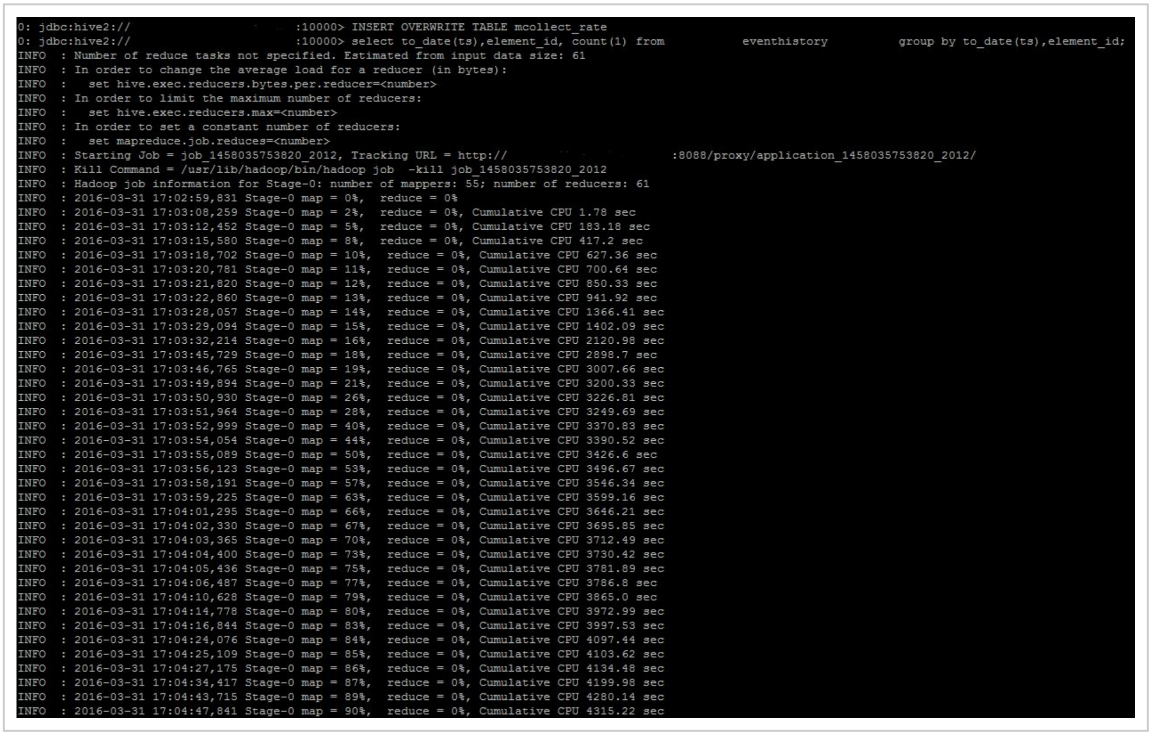
There are use cases where you perform batch analytics using Hive and index these metrics to ElasticSearch for visualisation purpose. You can insert into Elastic index from Hive as below

- Create an external hive table on the Elasticsearch index to which you would like to insert data, the index in elastic will be auto created on first insert

```
CREATE EXTERNAL TABLE mcollect_rate (
    day      STRING,
    metricid    BIGINT,
    count   BIGINT)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES('es.nodes' = 'ela1', 'es.resource' = 'mcrate/log');
```

- Insert data to Elasticsearch from another table called event_history, here I am indexing the temporal frequency of metric collection from IoT sensors in LHC. You can see in the below screenshot that Hive starts map-reduce to execute the query and then inserts the results into elastic index

```
INSERT OVERWRITE TABLE mcollect_rate
select to_date(ts),element_id, count(1) from event_history group by to_date(t
```



```
0: jdbc:hive2://[redacted]:10000> INSERT OVERWRITE TABLE mcollect_rate
0: jdbc:hive2://[redacted]:10000> select to_date(ts),element_id, count(1) from          eventhistory          group by to_date(ts),element_id;
INFO  : Number of reduce tasks not specified. Estimated from input data size: 61
INFO  : In order to change the average load for a reducer (in bytes):
INFO  :   set hive.exec.reducers.bytes.per.reducer=<number>
INFO  : In order to limit the maximum number of reducers:
INFO  :   set hive.exec.reducers.max=<number>
INFO  : In order to set a constant number of reducers:
INFO  :   set mapreduce.job.reduces=<number>
INFO  : Starting Job = job_1458035753820_2012, Tracking URL = http://[redacted]:8088/proxy/application_1458035753820_2012/
INFO  : Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1458035753820_2012
INFO  : Hadoop job information for Stage-0: number of mappers: 55; number of reducers: 61
INFO  : 2016-03-31 17:02:59,831 Stage-0 map = 0%,  reduce = 0%
INFO  : 2016-03-31 17:03:08,259 Stage-0 map = 2%,  reduce = 0%, Cumulative CPU 1.78 sec
INFO  : 2016-03-31 17:03:12,452 Stage-0 map = 5%,  reduce = 0%, Cumulative CPU 183.18 sec
INFO  : 2016-03-31 17:03:15,580 Stage-0 map = 8%,  reduce = 0%, Cumulative CPU 417.2 sec
INFO  : 2016-03-31 17:03:18,702 Stage-0 map = 10%,  reduce = 0%, Cumulative CPU 627.36 sec
INFO  : 2016-03-31 17:03:20,781 Stage-0 map = 11%,  reduce = 0%, Cumulative CPU 700.64 sec
INFO  : 2016-03-31 17:03:21,820 Stage-0 map = 12%,  reduce = 0%, Cumulative CPU 850.33 sec
INFO  : 2016-03-31 17:03:22,860 Stage-0 map = 13%,  reduce = 0%, Cumulative CPU 941.92 sec
INFO  : 2016-03-31 17:03:28,057 Stage-0 map = 14%,  reduce = 0%, Cumulative CPU 1366.41 sec
INFO  : 2016-03-31 17:03:29,094 Stage-0 map = 15%,  reduce = 0%, Cumulative CPU 1402.09 sec
INFO  : 2016-03-31 17:03:32,214 Stage-0 map = 16%,  reduce = 0%, Cumulative CPU 2120.98 sec
INFO  : 2016-03-31 17:03:45,729 Stage-0 map = 18%,  reduce = 0%, Cumulative CPU 2898.7 sec
INFO  : 2016-03-31 17:03:46,765 Stage-0 map = 19%,  reduce = 0%, Cumulative CPU 3007.66 sec
INFO  : 2016-03-31 17:03:49,894 Stage-0 map = 21%,  reduce = 0%, Cumulative CPU 3200.33 sec
INFO  : 2016-03-31 17:03:50,930 Stage-0 map = 26%,  reduce = 0%, Cumulative CPU 3226.81 sec
INFO  : 2016-03-31 17:03:51,964 Stage-0 map = 28%,  reduce = 0%, Cumulative CPU 3249.69 sec
INFO  : 2016-03-31 17:03:52,999 Stage-0 map = 40%,  reduce = 0%, Cumulative CPU 3370.83 sec
INFO  : 2016-03-31 17:03:54,054 Stage-0 map = 44%,  reduce = 0%, Cumulative CPU 3390.52 sec
INFO  : 2016-03-31 17:03:55,089 Stage-0 map = 50%,  reduce = 0%, Cumulative CPU 3426.6 sec
INFO  : 2016-03-31 17:03:56,123 Stage-0 map = 53%,  reduce = 0%, Cumulative CPU 3496.67 sec
INFO  : 2016-03-31 17:03:58,191 Stage-0 map = 57%,  reduce = 0%, Cumulative CPU 3546.34 sec
INFO  : 2016-03-31 17:03:59,225 Stage-0 map = 63%,  reduce = 0%, Cumulative CPU 3599.16 sec
INFO  : 2016-03-31 17:04:01,295 Stage-0 map = 66%,  reduce = 0%, Cumulative CPU 3646.21 sec
INFO  : 2016-03-31 17:04:02,330 Stage-0 map = 67%,  reduce = 0%, Cumulative CPU 3695.85 sec
INFO  : 2016-03-31 17:04:03,365 Stage-0 map = 70%,  reduce = 0%, Cumulative CPU 3712.49 sec
INFO  : 2016-03-31 17:04:04,400 Stage-0 map = 73%,  reduce = 0%, Cumulative CPU 3730.42 sec
INFO  : 2016-03-31 17:04:05,436 Stage-0 map = 75%,  reduce = 0%, Cumulative CPU 3781.89 sec
INFO  : 2016-03-31 17:04:06,487 Stage-0 map = 77%,  reduce = 0%, Cumulative CPU 3786.8 sec
INFO  : 2016-03-31 17:04:10,628 Stage-0 map = 79%,  reduce = 0%, Cumulative CPU 3865.0 sec
INFO  : 2016-03-31 17:04:14,778 Stage-0 map = 80%,  reduce = 0%, Cumulative CPU 3972.99 sec
INFO  : 2016-03-31 17:04:16,844 Stage-0 map = 83%,  reduce = 0%, Cumulative CPU 3997.53 sec
INFO  : 2016-03-31 17:04:24,076 Stage-0 map = 84%,  reduce = 0%, Cumulative CPU 4097.44 sec
INFO  : 2016-03-31 17:04:25,109 Stage-0 map = 85%,  reduce = 0%, Cumulative CPU 4103.62 sec
INFO  : 2016-03-31 17:04:27,175 Stage-0 map = 86%,  reduce = 0%, Cumulative CPU 4134.48 sec
INFO  : 2016-03-31 17:04:34,417 Stage-0 map = 87%,  reduce = 0%, Cumulative CPU 4199.98 sec
INFO  : 2016-03-31 17:04:43,715 Stage-0 map = 89%,  reduce = 0%, Cumulative CPU 4280.14 sec
INFO  : 2016-03-31 17:04:47,841 Stage-0 map = 90%,  reduce = 0%, Cumulative CPU 4315.22 sec
```

- check that the records are available with a quick count

```
select count(1) from mcollect_rate;
```

- Using the elastic commands you can also check the index state

```
curl 'myelastic1.mycompany.com:9200/_cat/indices?v'
```

Using the configuration properties available you can also specify the metadata fields of ES index, like @timestamp or id (es.mapping.id)
Another option available is if you already have data in json format this can be loaded directly to ES index

```
CREATE EXTERNAL TABLE fmem_json (data STRING)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES('es.nodes' = 'ela1', 'es.resource' = 'mcrate/log', 'es.input.json` =
```

And finally writing to dynamic ES index can also be quite useful, this is determined by the value of the Hive table column at runtime

```
CREATE EXTERNAL TABLE fmem (
    dt      timestamp,
    server    STRING,
    mem   bigint)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES('es.nodes' = 'ela1', 'es.resource' = 'fmem/{server}', 'es.query' = '
```

In the above example you will insert each server's data into its own index.

Offloading / Archiving Elasticsearch index to Apache Hive table

For longer term retention and batch analytics you can offload 'older data' to Hive and drop it from ES index. This can be achieved as below

- Create external Hive table on the index you would like to offload Hive

```
CREATE EXTERNAL TABLE fmem-2016-03-10 (
    dt      timestamp,
    server    STRING,
    mem   bigint)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES('es.nodes' = 'ela1', 'es.resource' = 'fmem-2016-03-10', 'es.que
```

- Insert as select into the Hive table from external table

```
INSERT TABLE fmem
select * from fmem-2016-03-10;
```

- Drop the Elasticsearch index

```
curl -XDELETE 'http://myelastic1:9200/fmem-2016-03-10/'
```

Conclusion

This blog post goes into the details of required setup and configuration for integrating Apache Hive with ElasticSearch and it explains at how to query Elastic from Hive, Indexing Hive data into Elastic and offloading indexes from Elastic to Hive for long term retention, There can be several use cases where such integration can be beneficial. In part 2 of this blog series we look at how Apache Spark can interact with Elasticsearch.

Tags: Hadoop Elastic Elasticsearch hive

Add new comment

Comments

Tom-Kun (not verified)
07 Jul 2016
permalink

Great article Prasanth

Great article Prasanth Kothuri, to understand how to plug and configure a Hive table on an Elasticsearch Index and do some queries. Is it possible to configure something similar as you did but with a batch process or an automatic process.
Great article.
Thomas.
But the offloading part interested me.

reply

Bosco (not verified)
02 Nov 2016
permalink

Excellent post

Amazing post. Not even a single piece of code error out. To the point and outstanding content. Really appreciate.

reply

Proquotient (not verified)
28 Feb 2017
permalink

Very in depth guide.

Thanks for sharing this highly in depth guide on integrating Hadoop and Elastic search , this is a very informative and easy to understand guide which makes it easy to learn.

reply

Add new comment

Your name *

E-mail *

The content of this field is kept private and will not be shown publicly.

Homepage

Subject

Comment *

- No HTML tags allowed.
- Web page addresses and e-mail addresses turn into links automatically.
- Lines and paragraphs break automatically.

More information about text formats

CAPTCHA

This question is to prevent automated spam submissions. Please note that you are not authenticated and therefore your comment will be held in a moderation queue and will not be published until it is approved.

Integrating Hadoop and Elasticsearch – Part 2 – Writing to and Querying Elasticsearch from Apache Spark

Integrating Hadoop and Elasticsearch - Part 1 - Loading into and Querying Elasticsearch from Apache Hive

Using SQL Developer to access Apache Hive with kerberos authentication

Save

Also by Prasanth Kothuri

Large Scale data reduction of AWAKE experiment data with Apache Spark and Notebooks
Benchmarking Apache Kafka on OpenStack VM's
Offline analysis of HDFS metadata
Real-time visualisation of Hadoop resources