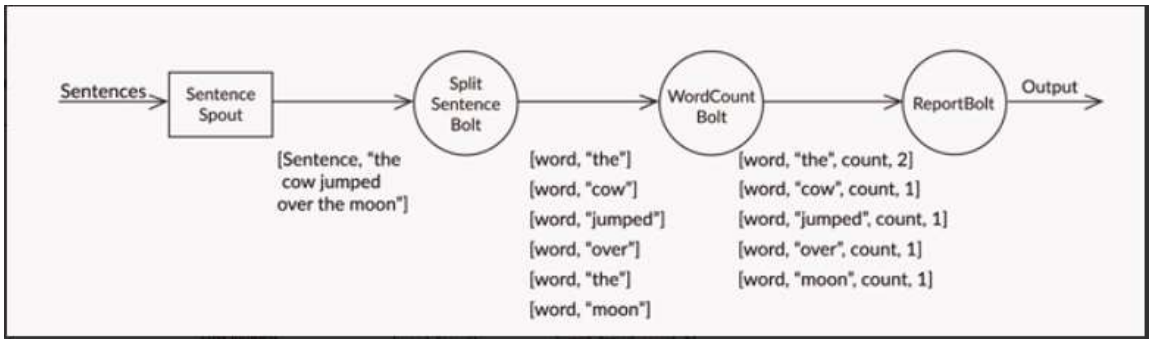
****

Section 1

****

Problem :

This Topology keeps a track of the count of each word in the input stream as shown in the graphics below.



In this WordCountTopology, we used Fields Grouping to send the stream of sentences from "SplitSentence" Bolt to the "WordCount" Bolt.

But this leads to an issue of load imbalance in the topology because there might be situations where a particular word has a high frequency in the data stream.

In such cases, one particular instance of the "WordCount" bolt would have to process a much higher number of words than the other instances of the

"WordCount" bolt. This can lead to a severe load imbalance problem.

Solution to this problem:

References :

http://www.corejavaguru.com/bigdata/storm/stream-groupings

https://www.quora.com/How-is-grouping-done-in-Apache-Storm#

As learnt from the session on Stream groupings and on research from above web pages, it could be referred:

****Shuffle grouping distributes tuples in a uniform, random way across the tasks. An equal number of tuples will be processed by each task. This grouping is ideal when you want to distribute your processing load uniformly across the tasks and where there is no requirement of any data-driven partitioning**.

Shuffle grouping: Tuples are randomly distributed across the bolt's tasks in a way such that each bolt is guaranteed to get an equal number of tuples.

Based on this , I arrived at a solution which uses Shuffle Grouping to send the stream of sentences from "SplitSentence" Bolt to the "WordCount" Bolt.

Hence to implement this , I have used below code lines in WordCountTopology class :

```
builder.setBolt(COUNT_BOLT_ID, countBolt,4).shuffleGrouping(SPLIT_BOLT_ID);
```

While this solution ensures that the load is uniformly distributed , there is still a glitch to counting total number of words.

Current scenario:

Now for e.g. a word like - "fleas" will not go to a single instance of wordcount bolt but might go to 3 out of 4 instances of wordcount bolt.

This would mean in each instance of Wordcount bolt the count of fleas will happen and the result would be emitted out like :

WordCountBolt_Instance1 -> (Fleas , 20)

WordCountBolt_Instance2 -> (Fleas , 10)

WordCountBolt_Instance3 -> (Fleas , 50)

Nowhere in our code in any of the classes , we are aggregating the results which would give us final count of fleas like - (Fleas , 20+10+50) = (Fleas,80)

This is a problem.

Solution:

Instead of counting the frequency of each word in WordCountBolt instances , I am modifying WordCountBolt to just emit the tuples like - (Word,1) . This means every word is emitted as tuple with count -> 1.

```
public void execute(Tuple tuple) {
    String word = tuple.getStringByField("word");
    Long count = 1L;
    this.collector.emit(new Values(word, count));
    this.collector.ack(tuple);
  }
```

I am finally going to receive all these tuples in Report Bolt's single instance through global streaming. This ensures that all the tuples with (word,1) are reaching Report Bolt. Hence, the word count logic is being moved to the Report bolt.

Here is how I am going to count the frequency of each word in Report Bolt:

```
public void execute(Tuple tuple)
        {
                String word = tuple.getStringByField("word");
                Long count = tuple.getLongByField("count");
                Long newCount = 0L;
                if(this.ReportCounts.containsKey(word)) {
                        newCount = this.ReportCounts.get(word) + count;
                }
                else {
                newCount = count;
                }
                this.ReportCounts.put(word, newCount);
```

Each tuple of the form: (word,count) being received in Report Bolt instance is being added as key,value (K,V) pair in a HashMap.

The code snippet above for Execute method does the following:

1. It gets the word (for e.g. - Fleas) and checks if it is already present in the HashMap.

2. If yes , it takes the value (count) for this key(word) from the Hashmap and adds the current count which is (1L) coming in the tuple to it. This increments the word count.

3. If not , it adds the new count (1L) for the incoming word in the Hashmap.

The counts of each word are hence incremented each time execute method is called for the incoming tuple. The counts of the words in the input stream are then stored in the MySQL database.

For ensuring the **at least-once processing** of input messages, following has been done:

(i) At least once processing at Spout Level:

SpoutOutputCollector contains implementations of emit() method which can attach a message ID to the outgoing Tuple. This message ID is unique for all the tuples emitted from the spout. I have used this in my **SentenceSpout** as:

```
public void nextTuple()
   {
     UUID msgID = UUID.randomUUID();
      this.collector.emit(new Values(sentences[index]),msgID);
      index++;
```

```
        if (index >= sentences.length)
        {
            index = 0;
        }
    }
```

** Storm's event logging functionality will only work if the messageId is serializable via Kryo or the Serializable interface. The emitted values must be immutable. Hence, I am making use of UUID here because **Java UUID class represents an immutable universally unique identifier and represents 128-bit value**.

(ii) At Least once processing at Bolt Level:

OutputCollector provides you with the feature of anchoring the outgoing Tuples. In each Bolt of the solution – **SplitSentenceBolt** , **WordCountBolt** and **ReportBolt** , to ensure the Reliability or at least once processing, following steps are made:

1. Read the incoming tuple with the message ID.
2. Anchor the outgoing tuple.
3. Acknowledge the tuple has been processed.

```
public void execute(Tuple tuple) {
        try {
                String sentence = tuple.getStringByField("sentence");
                String[] words = sentence.split(" ");
                for(String word : words){
                        this.collector.emit(tuple , new Values(word));

                }
                this.collector.ack(tuple);
        }catch(Exception e) {

                this.collector.fail(tuple);
        }
    }
```

Here, the emit method takes the input tuple as a parameter and hence the message ID from the incoming tuple is copied to the outgoing tuple. This is called anchoring. This links the newly originating tuple to the root message from which it originated.

For acknowledging the tuple has been processed successfully, ack method is invoked. In case, a tuple is failed to process , collector.fail(tuple) is called by Bolt to notify the Spout and the message is replayed.

****

Section 2

****

Import the JAR file of your solution to AWS CDH install instance using **WinSCP**. Put it in the home directory.

Running the JAR in Local Mode:

Step1: Move inside the Apache Storm folder using the following command

cd ~/apache-storm-1.2.1

Step2: For running the WordCount_MySQL Topology in local mode, I used the following command:

bin/storm jar ~/WordCount_MySQL-0.0.1-SNAPSHOT.jar WordCountTopology

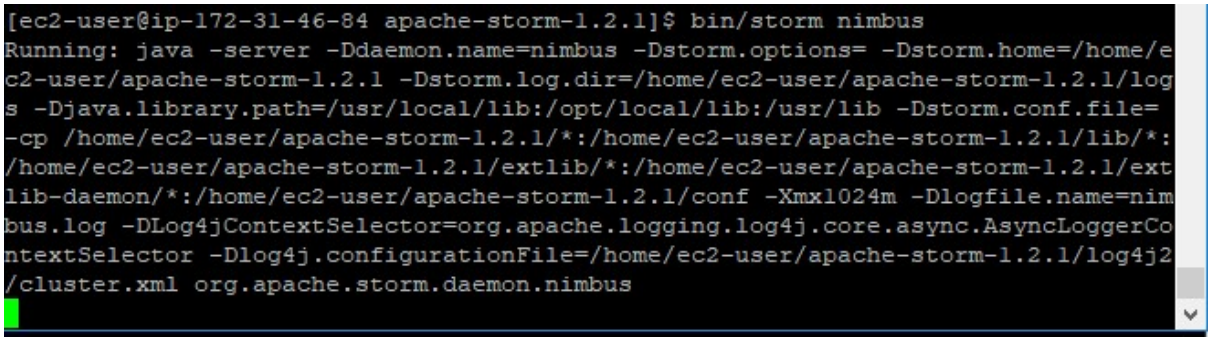
```
op interrupted!
67721 [SLOT_1024] INFO  o.a.s.d.executor - Shut down executor SplitSentenceBolt:
[3 3]
67721 [SLOT_1024] INFO  o.a.s.d.executor - Shutting down executor ReportBolt:[1
1]
67721 [Thread-26-ReportBolt-executor[1 1]] INFO  o.a.s.util - Async loop interru
pted!
67721 [Thread-25-disruptor-executor[1 1]-send-queue] INFO  o.a.s.util - Async lo
op interrupted!
--- FINAL COUNTS ---
a : 41599
ate : 41599
beverages : 41599
cold : 41599
cow : 41599
dog : 83198
dont : 83198
fleas : 83197
has : 41599
have : 41599
homework : 41599
i : 124795
like : 83197
man : 41599
my : 83198
the : 41599
think : 41599
--------------
67722 [SLOT_1024] INFO  o.a.s.d.executor - Shut down executor ReportBolt:[1 1]
67722 [SLOT_1024] INFO  o.a.s.d.executor - Shutting down executor WordCountBolt:
[6 6]
```

Running the JAR in Production Mode:

Step1: Open the Putty terminal and go to the folder Storm folder. Start Nimbus on the node using the following command:

cd ~/apache-storm-1.2.1

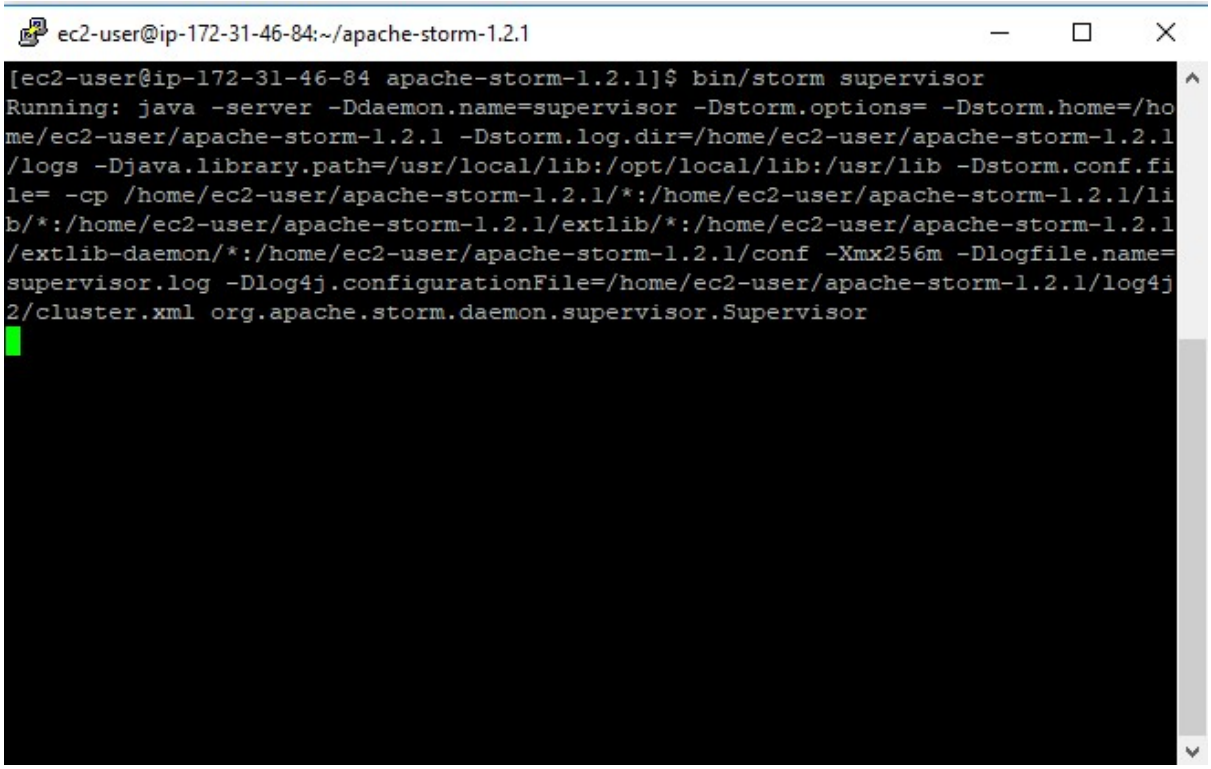
bin/storm nimbus



Step2: Open a new Putty terminal. Again move to the storm folder and start Supervisor on the node using the following command:

cd ~/apache-storm-1.2.1

bin/storm supervisor

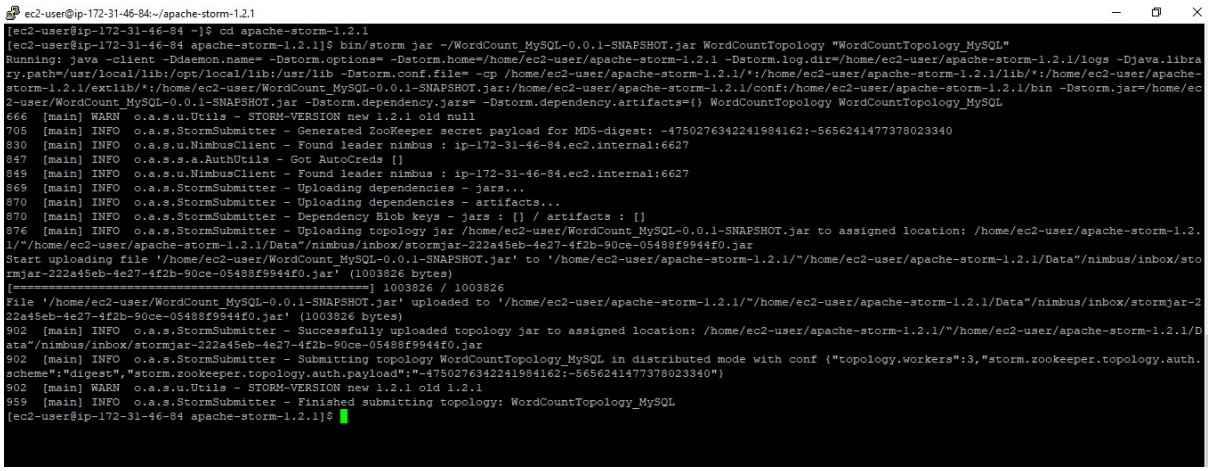


Step3: Open another Putty terminal and again move to the Storm folder.

cd ~/apache-storm-1.2.1

I used the following command to run the Topology in Production mode:

bin/storm jar ~/WordCount_MySQL-0.0.1-SNAPSHOT.jar WordCountTopology "WordCountTopology_MySQL"



Commands for creating database and table in MySQL :

Open an instance of Putty terminal , log into the AWS CDH install instance and go to MySQL as ec2-user :

mysql -u root -p

password: 123

MySQL database name: **upgrad**

Table name to store count of words: **wordcounts**

Steps:

1. Create database : **create database upgrad;**
2. Use database : **use upgrad;**
3. Create table : **create table wordcounts (word varchar(1024) , count bigint);**
4. Check table schema : **describe wordcounts;**
5. Check results: **select * from wordcounts;**

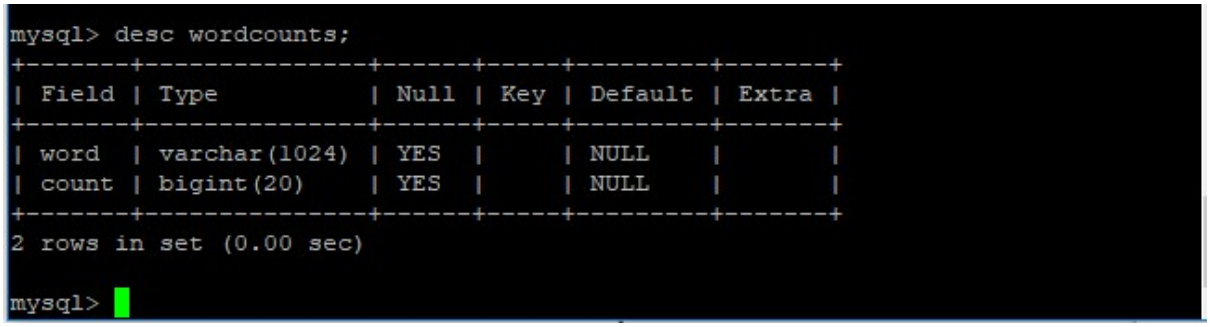
****

Section 3

****

A snapshot of the table(wordcounts) in upgrad database which stores the count of words :

mysql> desc wordcounts;



mysql> select * from wordcounts;

```
ec2-user@ip-172-31-46-84:~
Database changed
mysql> select * from wordcounts;
+-----------+-------+
| word      | count |
+-----------+-------+
| a         | 11567 |
| ate       | 11514 |
| beverages | 11554 |
| cold      | 11514 |
| cow       | 11590 |
| dog       | 23048 |
| dont      | 22964 |
| fleas     | 22845 |
| has       | 11507 |
| have      | 11540 |
| homework  | 11516 |
| i         | 34362 |
| like      | 23022 |
| man       | 11503 |
| my        | 23091 |
| the       | 11467 |
| think     | 11396 |
+-----------+-------+
17 rows in set (0.00 sec)
```

Enabling the Storm UI

Step 1: Ensure Zookeeper, Nimbus and Supervisors are running on the CDH instance.

Step 2: Open a new terminal and navigate to the Storm folder.
cd ~/apache-storm-1.2.1

Once inside the Storm folder, write the following command:
bin/storm ui

```
[ec2-user@ip-172-31-46-84 ~]$ cd apache-storm-1.2.1
[ec2-user@ip-172-31-46-84 apache-storm-1.2.1]$ bin/storm ui
Running: java -server -Ddaemon.name=ui -Dstorm.options= -Dstorm.home=/home/ec2-u
ser/apache-storm-1.2.1 -Dstorm.log.dir=/home/ec2-user/apache-storm-1.2.1/logs -D
java.library.path=/usr/local/lib:/opt/local/lib:/usr/lib -Dstorm.conf.file= -cp
/home/ec2-user/apache-storm-1.2.1/*:/home/ec2-user/apache-storm-1.2.1/lib/*:/hom
e/ec2-user/apache-storm-1.2.1/extlib/*:/home/ec2-user/apache-storm-1.2.1/extlib-
daemon/*:/home/ec2-user/apache-storm-1.2.1:/home/ec2-user/apache-storm-1.2.1/con
f -Xmx768m -Dlogfile.name=ui.log -DLog4jContextSelector=org.apache.logging.log4j
.core.async.AsyncLoggerContextSelector -Dlog4j.configurationFile=/home/ec2-user/
apache-storm-1.2.1/log4j2/cluster.xml org.apache.storm.ui.core
```

Step 3: Launch a browser and launch Storm UI using this command in the address bar:

<public IPv4 IP of your instance>:8080/

(1) Topology Summary table

Topology Summary

Search:

Name	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Assigned Mem (MB)	Scheduler Info
WordCountTopology_MySQL	ec2-user	ACTIVE	5m 57s	3	12	12	1	2496	

Showing 1 to 1 of 1 entries

Going inside Topology Summary :

Storm UI

Search WordCountTopology_MySQL-2-1534686456: Search Search Archived Logs:

Topology summary

Name	Id	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Assigned Mem (MB)	Scheduler Info
WordCountTopology_MySQL	WordCountTopology_MySQL-2-1534686456	ec2-user	ACTIVE	6m 41s	3	12	12	1	2496	

Topology actions

Activate | Deactivate | Rebalance | Kill | Debug | Stop Debug | Change Log Level

Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked	Failed
10m 0s	49692040	49692040	12329.165	66400	4515740
3h 0m 0s	49692040	49692040	12329.165	66400	4515740
1d 0h 0m 0s	49692040	49692040	12329.165	66400	4515740
All time	49692040	49692040	12329.165	66400	4515740

(2) Spout Table and Bolt Table

Spouts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Error Port	Last error	Error Time
SentenceSpout	1	1	5047100	5047100	12329.165	66400	4515740				

Showing 1 to 1 of 1 entries

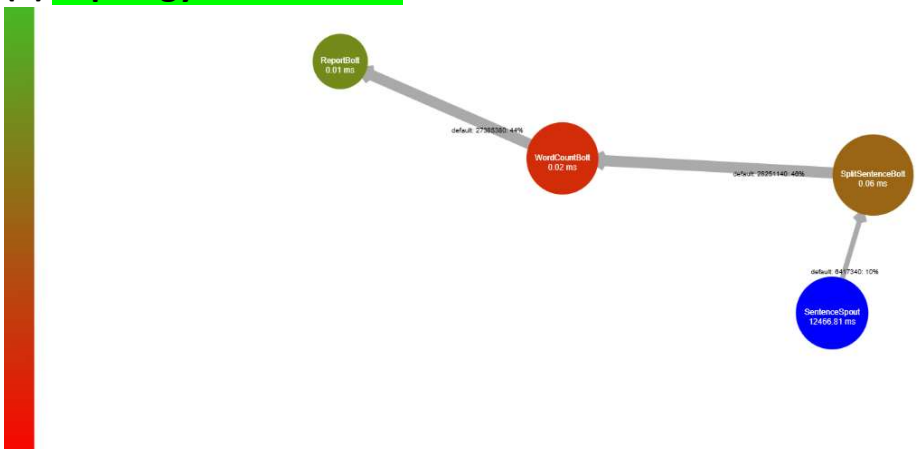
Bolts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	Error Host	Error Port	Last error	Error Time
ReportBolt	1	1	0	0	0.295	0.337	14000	0.439	14000	0				
SplitSentenceBolt	3	3	22554820	22554820	0.875	0.075	4698540	0.067	4698520	0				
WordCountBolt	4	4	22090120	22090120	0.579	0.017	22088240	0.027	22088240	0				

Showing 1 to 3 of 3 entries

(3) Topology Visualisation



****

Killing the Topology after getting expected results

****

Step 1: On a Putty terminal, go to Storm folder.

cd ~/apache-storm-1.2.1

Once inside the Storm folder, write the following command:
bin/storm kill WordCountTopology_MySQL

```
[ec2-user@ip-172-31-46-84 apache-storm-1.2.1]$ bin/storm kill WordCountTopology_MySQL
Running: java -client -Ddaemon.name= -Dstorm.options= -Dstorm.home=/home/ec2-user/apache-storm-1.2.1 -Dstorm.log.dir=/home/ec2-user/apache-storm-1.2.1/logs -Djava.libra
ry.path=/usr/local/lib:/opt/local/lib:/usr/lib -Dstorm.conf.file= -cp /home/ec2-user/apache-storm-1.2.1/*:/home/ec2-user/apache-storm-1.2.1/lib/*:/home/ec2-user/apache-
storm-1.2.1/extlib/*:/home/ec2-user/apache-storm-1.2.1/extlib-daemon/*:/home/ec2-user/apache-storm-1.2.1/conf:/home/ec2-user/apache-storm-1.2.1/bin org.apache.storm.com
mand.kill_topology WordCountTopology_MySQL
5925 [main] INFO  o.a.s.u.NimbusClient - Found leader nimbus : ip-172-31-46-84.ec2.internal:6627
5976 [main] INFO  o.a.s.c.kill-topology - Killed topology: WordCountTopology_MySQL
[ec2-user@ip-172-31-46-84 apache-storm-1.2.1]$
```

Step 2: Checking the Storm UI after killing the topology:

Topology Summary

Search:

Name	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Assigned Mem (MB)	Scheduler Info
No data available in table									

Showing 0 to 0 of 0 entries

Supervisor Summary

Search:

Host	Id	Uptime	Slots	Used slots	Avail slots	Used Mem (MB)	Version
ip-172-31-46-84.ec2.internal (log)	da662aa1-2738-426d-8f98-2879a6be6d57	31m 36s	4	0	4	0	1.2.1

Showing 1 to 1 of 1 entries