Datastax Software Engineer in Test Screen: Jmx Logging

Introduction

The goal of this assignment is to automate a typical QA/Performance Engineering task. You will design a program to record JMX values from a running Cassandra Instance while checking that certain thresholds are not exceeded. Once this is completed you will graph the recorded metrics and provide some analysis of the operation.

We know that this is a difficult task to accomplish when you aren't familiar with the tools provided or Cassandra and would be glad to review any attempts at completing this set of tasks.

Task

First

1. Download and install Cassandra on a Linux Based System
2. Setup any additional libraries you may want to use (see MX4j below)
3. Start Cassandra

Write an automated script in java, scala or python (unless a different language has been specifically required) which performs the following actions:

1. Ensures Cassandra is Running
2. Begins Recording the following JMX Metrics
    - org.apache.cassandra.metrics.ColumnFamily.keyspace.columnfamily.LiveSSTableCount
    - o.a.c.m.ColumnFamily.keyspace.columnfamily.AllMemTablesDataSize
    - o.a.c.m.ClientRequest.Write.Latency.95thPercentile)
3. Runs the external tool Cassandra Stress (Extra Challenge: Write your own stress method using a driver)
4. Once the stress session has completed stop recording JMX Metrics
5. Perform some assertion checks that your recorded metrics make sense (are they always 0? do they go negative?)
6. (Extra Challenge) Record the metrics back into a Cassandra Table

You do *NOT* need to do the following in your script

- Automatically launch EC2 instances or set up your Linux Environment
- Automatically Setup Up Cassandra (do this manually before running the test)

Once your script is running well please

1. Graph the results (Extra Challenge: Have the automated script create these graphs at the end of the run)
2. Write a brief description of what is happening in each graph (what do does it mean when the value goes up or down rapidly? etc...)

Deliverables

Please Return

1. Source in the language of your choice for running the the automated task and recording the resultant metrics
2. Graphs showing the chosen metrics over the duration of the experiment. These graphs should be labeled such that it is obvious at a glance what is being graphed.
3. Brief description of what is occurring in each graph, and what it means for the system. Please use this as an opportunity to show off your understanding of Cassandra's internals.

Tools

Below is a non-exhaustive list of tools you could use to accomplish this project. The majority of QA tasks will require integrating multiple tools from a variety of sources. For this project feel free to use any tools you like, below is just a list of helpful suggestions.

Cassandra

The product, a Java based distributed database. Since this application runs on the Java Virtual Machine it exposes a variety of internal state metrics through the Java Metrics Extension (JMX). These values let an external user poll the internal state of the server. We commonly watch these values to make sure that everything is behaving appropriately and various bundled tools (nodetool, opscenter) rely on connecting to JMX to function.

Cassandra Stress

A java based tool for quickly inserting/reading a large amount of data to a Cassandra Database. Running this without any command line options will insert 100k keys into a database accessible on localhost.

JConsole

A tool for browsing JVM state including JMX mbeans.

JMX Recording Tools:

Choose any one (or more) of these tools to query jmx state if you aren't using a language that can directly communicate with the JVM (Java, Jython, Jruby ...)

JMXTerm

A commandline interface for querying jmx metrics. This is useful if you aren't using Java and would like to get the current state of a particular jmx mbean. To use it just download the jar

```
#Example grab the org.apache.cassandra.metrics.ColumnFamily.Keyspace1.Standard1 mbean and return the value of MemTableDataSize
echo get -s -b org.apache.cassandra.metrics:keyspace=Keyspace1,name=MemtableDataSize,scope=Standard1,type=ColumnFamily Value | java -j
```

Jmxterm home site

JMXTrans

A recording tool for JMX mbeans which will record metrics to a file. Useful if you would like to require a secondary system to do the actual data aggregation and parsing. Jmxtrans uses a query structure to query JMX metrics on a regular basis. Download from JmxTrans

```
#Sample Query File for downloading all of the metrics from Keyspace1 Standard1
[ {
    "outputWriters" : [ {
        "@class" : "com.googlecode.jmxtrans.model.output.KeyOutWriter",
         "settings" : {
         "outputFile" : "/tmp/perf-regression-jmx-metric-out.txt",
         "maxLogFileSize" : "10MB",
         "maxLogBackupFiles" : 200,
         "typeNames" : ["name"]
         }
      } ],
     "obj" : "org.apache.cassandra.metrics:type=ColumnFamily,keyspace=Keyspace1,scope=Standard1,name=*",
     "attr" : ["Value"],
     "resultAlias" : "Keyspace1.Standard1"
} ]

# Running jmxtrans
export SECONDS_BETWEEN_RUNS=%s;'
export JAVA_HOME=%s;'
./jmxtrans.sh start query_file
```

Installation and setup for Jmxtrans

Mx4j

A http interface to a running JVM process. This is yet another access point for recording JMX data. To install simply place the jar in the cassandra lib directory. The server should start by default when starting a cassandra cluster. Acessing it is a simple REST service so it requires no additional software for the client.

```
#Access org.apache.cassandra.db mbean Storage Service, Attribute LiveNodes - The current live connection points this node knows about
http://localhost:8081/getattribute?objectname=org.apache.cassandra.db:type=StorageService&attribute=LiveNodes&template=identity&format
#Other Examples
http://localhost:8081/getattribute?objectname=org.apache.cassandra.db:type=StorageService&attribute=OperationMode&template=identity
http://localhost:8081/getattribute?objectname=org.apache.cassandra.db:type=StorageService&attribute=ReleaseVersion&template=identity
```

Mx4j Project Site

Graphing tools

1. R A great statistics programming language with built in tools for graphing. There are hooks into almost every major language for R so it is great for automation.
2. MATLAB A commerical mathematics programming languge which also is easily accessed programatically.
3. Excel
4. Google Drive : Spreadsheets