Use a YAML file to run cassandra-stress

Benchmark Any Schema – Part 1

The YAML file is split into a few sections:

1. DDL - for defining your schema
2. Column Distributions - for defining the shape and size of each column globally and within each partition
3. Insert Distributions - for defining how the data is written during the stress test
4. DML - for defining how the data is queried during the stress test

```
mkdir -p /home/hadoop/Admatic/Cassandra
cd /home/hadoop/Admatic/Cassandra

vim blogpost.yaml
```

```
### DML ###

# Keyspace Name
keyspace: stresscql

# The CQL for creating a keyspace (optional if it already exists)
keyspace_definition: |
  CREATE KEYSPACE stresscql WITH replication = {'class': 'SimpleStrategy'
, 'replication_factor': 1};
# Table name
table: blogposts

# The CQL for creating a table you wish to stress (optional if it already
 exists)
table_definition: |
  CREATE TABLE blogposts (
        domain text,
        published_date timeuuid,
        url text,
        author text,
        title text,
        body text,
        PRIMARY KEY(domain, published_date)
    ) WITH CLUSTERING ORDER BY (published_date DESC)
      AND compaction = { 'class':'LeveledCompactionStrategy' }
      AND comment='A table to hold blog posts'
### Column Distribution Specifications ###

columnspec:
  - name: domain
    size: gaussian(5..100)       #domain names are relatively short
    population: uniform(1..10M)  #10M possible domains to pick from

  - name: published_date
    cluster: fixed(1000)         #under each domain we will have max 1000
 posts

  - name: url
    size: uniform(30..300)

  - name: title                  #titles shouldn't go beyond 200 chars
    size: gaussian(10..200)

  - name: author
    size: uniform(5..20)         #author names should be short

  - name: body
    size: gaussian(100..5000)    #the body of the blog post can be long

### Batch Ratio Distribution Specifications ###

insert:
  partitions: fixed(1)            # Our partition key is the domain so on
ly insert one per batch

  select:    fixed(1)/1000        # We have 1000 posts per domain so 1/10
00 will allow 1 post per batch

  batchtype: UNLOGGED             # Unlogged batches


#
# A list of queries you wish to run against the schema
#
queries:
   singlepost:
      cql: select * from blogposts where domain = ? LIMIT 1
      fields: samerow
   timeline:
      cql: select url, title, published_date from blogposts where domain
= ? LIMIT 10
      fields: samerow
```

Inserts

```
cassandra-stress user profile=blogpost.yaml ops\(insert=1\) > inserts.txt

tail -f inserts.txt
```

Queries

```
cassandra-stress user profile=blogpost.yaml ops\(singlepost=1\) > query1.
txt

tail -f query1.txt
```

```
cassandra-stress user profile=blogpost.yaml ops\(timeline=1\) > query2.tx
t

tail -f query2.txt
```

Mixed

This syntax sends three queries for every one insert.

```
cassandra-stress user profile=blogpost.yaml ops\(singlepost=2,timeline=1,
insert=1\) > mixed.txt

tail -f mixed.txt
```

Use the -graph option

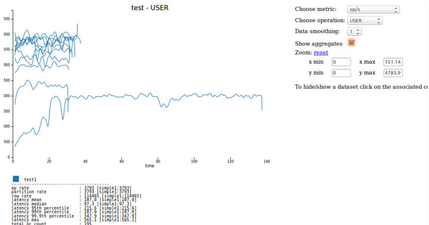
The -graph option provides visual feedback for cassandra-stress tests. An interactive graph can be displayed with a web browser.

```
cassandra-stress user profile=blogpost.yaml ops\(insert=1\) -graph file=i
nserts.html title=test revision=test1

cassandra-stress user profile=blogpost.yaml ops\(singlepost=1\) -graph fi
le=query1.html title=test revision=test1

cassandra-stress user profile=blogpost.yaml ops\(timeline=1\) -graph file
=query2.html title=test revision=test1

cassandra-stress user profile=blogpost.yaml ops\(singlepost=2,timeline=1,
insert=1\) -graph file=mixed.html title=test revision=test1
```



Cassandra Stress Graph