Adobe Analytics - POC Test Harness

MongoDB Consulting Notes

Eric Reid <eric.reid@mongodb.com> Senior Consulting Engineer, MongoDB Inc.

v1.2	16 June 2021	Added new operations; improved documentation
v1.1	*27 May 2021*	*Added ability to create a percentage mix of operations; refined initial operations*
v1.0	*26 May 2021*	*Initial Draft*

Table of Contents

Background

Adobe Analytics is a SaaS product for instrumenting, tracking, and reporting web interactions. Common metrics which are captured and reported include IP address, OS, User Agent, Page Name, etc. Custom metrics can be configured for things such as Product Name/SKU, Article Id, Campaign Id, Search terms, etc.

The primary focus for this consult was the Proof of Concept for running **Name Lookup Service (NLS)** on Atlas. NLS is a service used to store key/value pairs comprising a variable being tracked for analytics and the 32 or 64-bit hash of the value as a key.

User reports require matching on a string and returning a list of hashes which are provided to a separate reporting system.

After standing up a POC Atlas cluster, MongoDB recommends use of a simple JMeter[1] -based test harness for driving load-testing of Atlas.

As of this writing, this test harness has been run against M50, M60 and M80 Atlas Clusters, and has been run on AWS EC2 instances up to m5.16xlarge (64 vCPUs, 256GB memory). As it is fairly easy to overload a single Test Harness VM with this test harness, we strongly recommend:

- Use a large enough instance for your test harness, based on max # simultaneous test threads, or
- Consider use of multiple JMeter hosts (*note: unable to get this working in Adobe EC2 instances*)
- Monitor instance resources during load testing to determine if resource exhaustion on the test harness system is occuring

Requirements

- Simple test harness which can drive Atlas clusters for the POC
- Intended to run on separate(s) VM on AWS or Azure

Assumptions/Prerequisites

- The Adobe team has a basically familiarity with running JMeter and analyzing results
- Load Harness runs on a VM (or VMs) that are not running any other applications
- Atlas Project has whitelisted the Test Harness IP address(es)
- Atlas VMs stats need to be gathered separately, especially if running tests off-hours
- Insure the test harness system(s):
    1) are whitelisted in Atlas
    2) Have a Java 1.8 or later JVM installed
- Insure the Atlas instance to be tested is populated with the latest agreed-upon collections (ex: use of contiguous Integers for _id)

Creating the Test Harness

- Download and install the following to your test system(s):
    - JMeter latest[2] - install wherever you choose

[1] https://jmeter.apache.org/
[2] https://jmeter.apache.org/download_jmeter.cgi#binaries

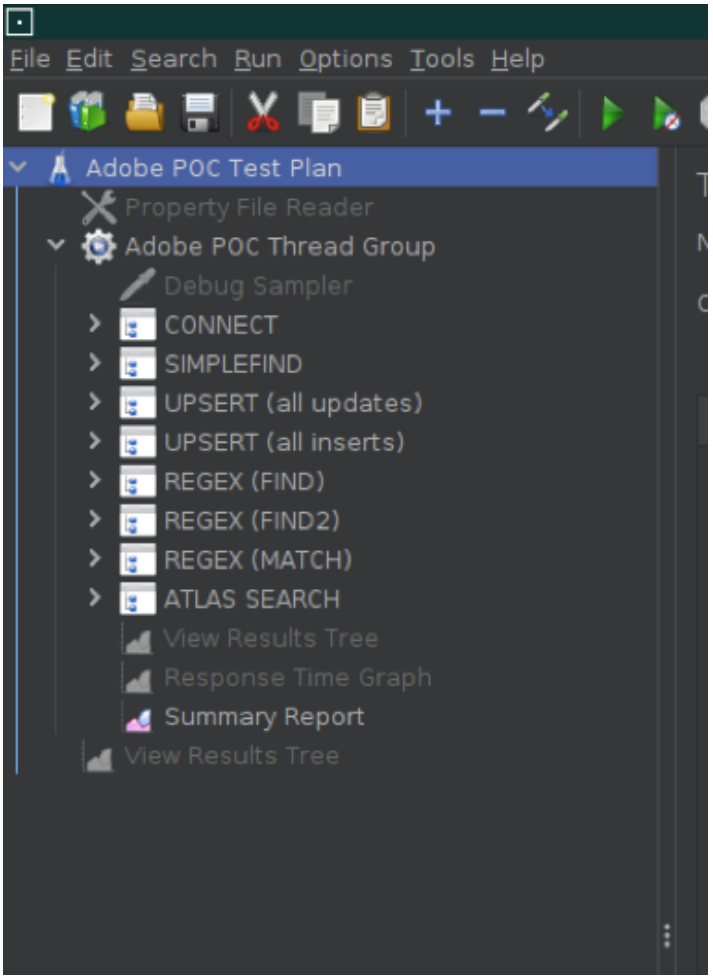
- Applicable **MongoDB Java 'sync' jarfile**[3] and **BSON jarfile**[4] (I used the latest v4.2.3 of both) - place these in $JMETER_HOME/lib
- The JMeter plugin "Property File Reader"[5]

- Make sure **`$JMETER_HOME`**, **`$PATH and $JAVA_HOME`** are correctly set
- Take the following files supplied by MongoDB
  - AdobeNLS.jmeter.properties (configurable test plan Properties)
    - Wherever this is to live (I placed mine in ~), make sure that you modify the test plan (usually via GUI) to point to its proper location
  - AdobeNLS_test.jmx (the test plan)
    - This file can live anywhere

Checking the Test Harness

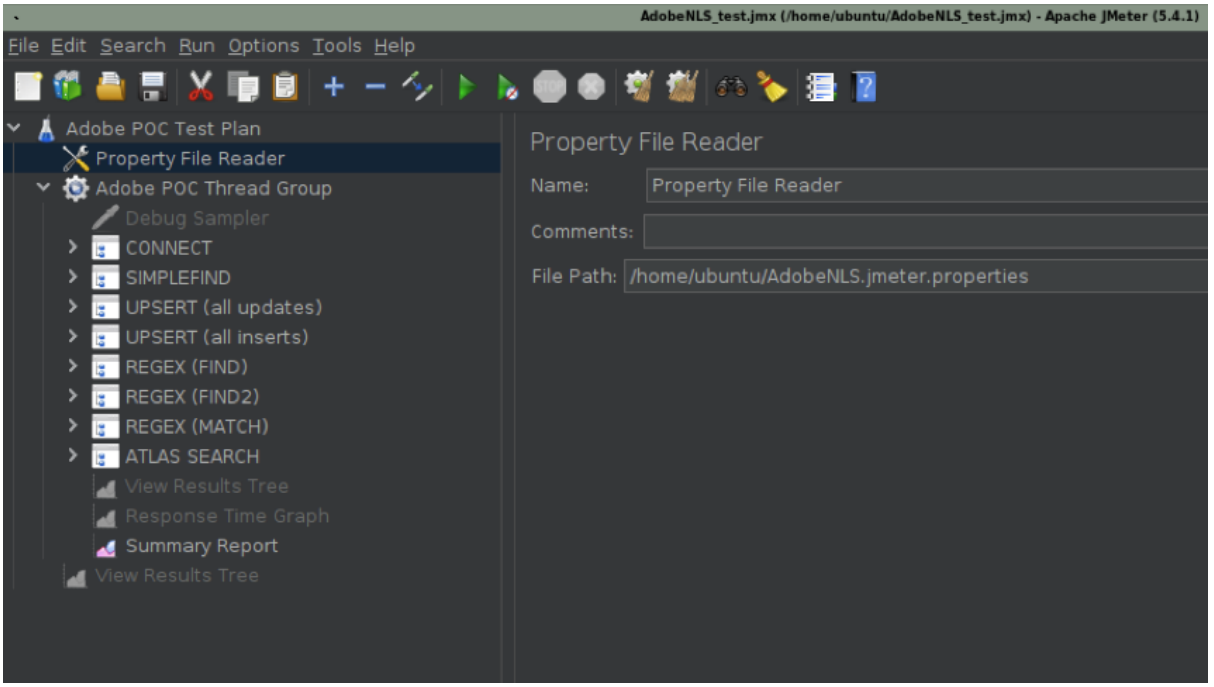
Start JMeter in GUI mode ($JMETER_HOME/bin/jmeter.sh), and check:

1) The Test Plan loads correctly. JMeter's left hand-side pane should look like this:



[3] https://github.com/mongodb/mongo-java-driver
[4] https://repo1.maven.org/maven2/org/mongodb/bson/4.2.3/bson-4.2.3.jar
[5] https://www.vinsguru.com/download/87/

2) Make sure you set the Property File Reader "File Path" to the properties file mentioned above:



3) Insure you can connect to your Atlas instance from this VM

Test Harness Data

Various-sized 'lookup' collections (1 million, 20 million, 50 million documents) of the following form:

```
{ "_id" : NumberLong("100000000000"), "n" : "rsQQlet+jqHlLH0tpPMGqw==", "nu8" : false,
"m" : [ 10088, 10026, 10023, 10007, 10014, 12138, 12128, 12033 ] }
{ "_id" : NumberLong("100000000001"), "n" : "rsQQlet+jqHlLH0tpPMGqw==", "nu8" : false,
"m" : [ 10046, 10025, 10040, 10037, 10074, 12152 ] }
{ "_id" : NumberLong("100000000002"), "n" : "rsQQlet+jqHlLH0tpPMGqw==", "nu8" : false,
"m" : [ 10098, 12165 ] }
{ "_id" : NumberLong("100000000003"), "n" : "rsQQlet+jqHlLH0tpPMGqw==", "nu8" : false,
"m" : [ 10092, 10063, 10072, 12020, 12086 ] }
{ "_id" : NumberLong("100000000004"), "n" : "rsQQlet+jqHlLH0tpPMGqw==", "nu8" : false,
"m" : [ 10100, 12047, 12090 ] }
{ "_id" : NumberLong("100000000005"), "n" : "rsQQlet+jqHlLH0tpPMGqw==", "nu8" : false,
"m" : [ 10066, 10024, 10077, 12102, 12112, 12005 ] }
{ "_id" : NumberLong("100000000006"), "n" : "rsQQlet+jqHlLH0tpPMGqw==", "nu8" : false,
"m" : [ 10027, 10068, 10084, 10039, 12058, 12149 ] }
{ "_id" : NumberLong("100000000007"), "n" : "rsQQlet+jqHlLH0tpPMGqw==", "nu8" : false,
"m" : [ 10069, 10094, 10093, 10019, 10076, 12042, 12031, 12006 ] }
```

Instead of hash values for _id, integers are use in a contiguous range.

Test Harness Logic

A single Test Plan with a single Thread Group is provided, but can be split up anyway the Adobe testers see fit.

The JSR223 Samplers which comprise this Test Plan are:

- CONNECT - Run only once at the beginning, creates connection to the Atlas instance
- SIMPLEFIND - Runs **`find()`** on an existing **`_id`** value
- UPSERT (all updates) - Runs upsert commands similar to what Adobe is using on-prem, against existing document
- UPSERT (all inserts) - Runs upsert commands similar to what Adobe is using on-prem, but with no matches; **Note: after running this test, additional documents will be inserted that do not conform to the 'contiguous _id range' convention - the collection must be restored to original before additional tests can be run against it**
- REGEX (FIND, FIND2, MATCH) - Three versions of a basic find() command similar to what Adobe is using on-prem using RegExs
- ATLAS SEARCH (Placeholder) (Not yet implemented) - Runs a single $search against Atlas Search

Other Samplers can be added for other desired operations.

Runs are currently not timed, but run forever until killed. This can be easily changed.

Running the Test Harness

While you can use the JMeter GUI to modify and experiment with your Test Plan, actual testing should only be done via the non-GUI mode, to minimize impact to the Test Harness system.

Example:

```
% $JMETER_HOME/bin/jmeter \
  -n -t AdobeNLS.jmx -l <testName>.jtl -j <testName>.log \
  -o <testName>.output
```

The **`AdobeNLS.jmeter.properties`** currently has the following supported Properties:

Property	***Current Value***	***Description***

URI	**mongodb+srv://atlas_admin:i0DfblS04OadcCxw@pnw-nls-poc.lk9tu.mongodb.net/?retryWrites=true&w=majority**	URI of Atlas Cluster
DATABASE	**lookupservice**	Database name
COLLECTION	**lookups_50mildocs**	Collection name
THREADS	**300**	Number of simultaneous JMeter threads
RAMPUP	**1**	Ramp-up time for each thread (secs)
PERCENT_SIMPLEFIND	**100**	% of operations
PERCENT_UPSERT_ALL_UPDATES	**0**	
PERCENT_UPSERT_ALL_INSERTS	**0**	
PERCENT_REGEX_FIND	**0**	
PERCENT_REGEX_FIND2	**0**	
PERCENT_REGEX_MATCH	**0**	
PERCENT_ATLASSEARCH	**0**	

Once you have set the values in your **AdobeNLS.jmeter.properties** file, you are ready to run a test:

```
% $JMETER_HOME/bin/jmeter.sh -n -e -t AdobeNLS.jmx -l <testName>.jtl \
  -j <testName>.log -o <testName>.output
```

Note: Any changes to parameters in the parameters file requires a restart of the GUI version of JMeter.