TPC-DS on Hadoop

The TPC-DS benchmarks are the de-facto industry standard benchmark for decision support systems and provide a challenging suite of queries run against a realistic data set (http://www.tpc.org/tpcds/).

This document describes how to generate and load the TPC-DS data onto a Hadoop cluster, then query it using Kognitio software – this can be done either with 8.1 production software running on separate nodes, or with the Kognitio on Hadoop (kodoop) pre-release.

Appendix A will be enhanced to describe an additional test that can be run against the TPC-DS data demonstrating Kognitio's high concurrency capabilities.

Note
This benchmark is designed to show how the database responds under stress, it should not be run on a production system.

Prerequisites
- A Kognitio system (8.1 or pre-release Kognitio on Hadoop) setup to access Hadoop – see existing documentation from community forum at http://www.kognitio.com/forums/viewtopic.php?f=2&t=3 including quick reference sheets for HDFS and Map Reduce connectors).
- Permission to create and access HDFS files.
- SYS level access to the Kognitio database

Limitations
- The TPC-DS data refresh test is not yet implemented.
- Query 67 is not yet implemented.
- The base version of the TPC-DS distribution used to build this test suite is 1.1, the latest version is 2.1.

Setup
The benchmarks must be run from a Linux machine connected as a user with the ability to create HDFS files and you will also need SYS level access to the Kognitio database. The test suite is contained in the tar file kogtpcads.tar which is available from the Kognitio on Hadoop forum at (http://www.kognitio.com/forums/viewforum.php?f=13) and it should be unpacked into an empty directory.

The root directory of the test suite contains the script kogtpcads.bash and you should run the script from that directory.

The subdirectories contain support files and working directories and should not be altered.

Overview
The TPC-DS benchmarking process consists of 5 phases...

1. Creation of the test data set in HDFS.
2. Creation of the Kognitio tables that link to the data on HDFS.
3. Creation of the in-memory table images the queries are run against.
4. Execution of the test queries against the data set.
5. Removal of the in-memory images to free up space.

Each of these phases can be executed individually or all together in sequence using the –all command line parameter.

Example command line:

This runs the complete test suite, generating data, linking tables, creating memory images, running the queries and freeing up memory. The following is a breakdown of the options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-all</td>
<td>Run all phases in sequence</td>
</tr>
<tr>
<td>-fu</td>
<td>Collect full statistics on the memory images (recommended).</td>
</tr>
<tr>
<td>-sf 100</td>
<td>Generate a 100GB data set (the smallest data set to comply with the TPC-DS specifications – you can use 1 for initial testing to save time)</td>
</tr>
<tr>
<td>-hd /user/jeff/tpc-ds</td>
<td>Specifies the HDFS directory to create the data in – a subdirectory of SF&lt;scale factor&gt; will be created to contain the data to allow different scale factors to coexist on one system.</td>
</tr>
<tr>
<td>-s 10.12.11.15</td>
<td>The address of the Kognitio server node</td>
</tr>
<tr>
<td>-u sys</td>
<td>The Kognitio user (requires create user, schema, connector privileges so normally SYS)</td>
</tr>
<tr>
<td>-p syspwd</td>
<td>The password for the user</td>
</tr>
<tr>
<td>-nn 10.12.11.81:8020</td>
<td>The address and port for the HDFS namenode.</td>
</tr>
</tbody>
</table>

For large data sets (see TPC-DS documentation for recommended sizes) you may want to run more parallel data generation streams using the –ls option – be aware that running too many streams at once can cause problems for the host as the TPC supplied data generation program dsdgen is very CPU intensive.

Errors

If there are any errors the processing will abort with an error message. It may refer you to the logfiles which are in ./logs/sf<scale factor> - for example ./logs/sf10 for a run with a scale factor of 10. You may want to occasionally clear down this directory if you are doing a lot of runs.

Once the problem has been rectified, it is generally fine to just re-run the process since it will clear down anything left over from a previous run.

If you have a Kognitio support contract, please contact us via the support portal at http://kognitio.com/support/customer-support/ otherwise support is available on our community forum at http://www.kognitio.com/forums/viewforum.php?f=13

Results

Progress through the tests for each query stream is shown during the run and at the end of the run, minimum, average and maximum times are shown for each query. If a query has been omitted it will show a time of 999999.0.

Data Set

The data set is described in the TPC-DS documentation available in the docs directory. For each scale factor created, a database user is created with the name TPCDS<scale factor> and a password the same as the user name. An external connector is also created with the name HDFS_TPCDS<scale factor>. Dropping the user and connector with the cascade option will remove the data set from the database.

Command Line Parameter Reference

The command line parameters for kogtpcds.bash are described in detail below – many of these are optional and it is recommended that the basic parameters described above are used to run the standard tests unless there is a specific need to change them.

Verbose logging can be enabled with the –v parameter. If this is enabled there will be a number of error messages emitted – this is to be expected and they are generally errors caused by deletion of objects that may have been created in previous runs but were not.

Processes to run

As described above, there are 5 phases which need to be run in order but not necessarily at the same time. For example, you may want to build the data in HDFS and then run the queries against it with different parameters.
### Parameter | Description
--- | ---
-gd | Generate HDFS data
-ct | Create the tables that link to the HDFS data
-ci | Create the memory images of the HDFS tables
- rq | Run the TPC-DS tests (requires -ci to be specified now or previously)
-di | Drop the memory images (requires -ci to be specified now or previously)
-all | All of the above

If not using the –all option, each stage depends on the previous stage and you will get errors if was omitted.

### HDFS Options

The following options pass in information about the HDFS filesystem.

| Parameter | Description |
--- | ---
-hd | HDFS directory to create test data under – e.g. /user/name/tpcds (must exist) (required for data generation and table creation)
-ls <n> | The number of parallel load streams to be used (default=2) (required for data generation)
-nn <namenode:port> | The address and port of the HDFS namenode) (required for data generation and table creation)

### Kognitio Database Options

The following options pass in information about the Kognitio database – all are required for every phase except the HDFS data generation phase.

| Parameter | Description |
--- | ---
-s svr | Specify server
-u user | Specify administrative user (will normally be SYS)
-p password | Password for connection

### TPC-DS Query Options

The following options control the running of the TPC-DS queries and can generally be omitted.

| Parameter | Description |
--- | ---
-sf <n> | The required TPC-DS scale factor (default=1[GB]) Official factors are: 100,300,1000,3000,10000,30000,100000
-qd | Diagnose all queries before running them
-ti filename | Query template list to be used (default=templates.lst)
-iq filename | Store individual query results in <filename>
-fu | Gather full statistics
Appendix A – High Concurrency

The TPS-DC benchmark targets the moderate concurrency high complexity query performance typical of decision support and analytical systems. The Kognitio database also performs extremely well in moderate complexity high concurrency situations. The test described below demonstrates this capability known as location hash value (LHV) optimisation.

As part of query optimisation, the query predicates are examined to determine whether they hash to a single ramstore. If they do, the optimiser only sends the query to that ramstore which then leaves all the other ramstores available for other queries. This feature allows query concurrency to scale with database size typically providing a two orders of magnitude performance increase on a medium sized system.

We are currently building tests to demonstrate this capability which will be available in the near future.