STREAM HORIZON
DataProcessingPlatform

StreamHorizon –
Test Demo deployment Guide

Version 3.1.8

StreamHorizon - Test Demo Deployment Guide

INTRODUCTION

INSTALLATION REQUIREMENTS

Thank you for downloading StreamHorizon test demo!

We will refer to directory where you have installed the demo software as $ENGINE_HOME.

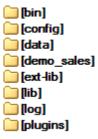There are few requirements that need to be met in order to run StreamHorizon.

1. Mainstream operating system (Linux, Windows, Solaris)
2. JDK 1.7+ (we recommend Oracle HotSpot)
3. Database must support JDBC interface.

NOTE: Even if you intend to run multiple (clustered) StreamHorizon instances there is no need to install binaries more than one per single (physical) server.

NOTE: 64-bit Java deployment is desireable (but not mandatory) for performance reasons

DEMO DIRECTORY STRUCTURE

Demo you have downloaded comes in two flavours, Oracle and MySQL. Demo is located in $ENGINE_HOME/demo_sales/ directory as shown below.

[bin]
[config]
[data]
[demo_sales]
[ext-lib]
[lib]
[log]
[plugins]

Directory name	Purpose
$ENGINE_HOME/bin/	All startup scripts should be placed in this directory. There are two default scripts (start.bat and start.sh) that can be utilised for starting single instance

	(default). If you are customizing new startup scripts always place them in this directory.
$ENGINE_HOME/config/	All configuration files are placed in this directory: • engine-config.xml is single most important configuration file which desciribes database connectivity, directory structues and your database (demo database model).
$ENGINE_HOME/data/	This is engine private directory used for housekeeping. You should not add, delete or modify content of this folder.
$ENGINE_HOME/demo_sales/	Demo feature demonstrating capabilities of StreamHorizon engine (performance and functionality).
$ENGINE_HOME/ext-lib/	Additional dependencies required by your custom plugins.
$ENGINE_HOME/lib/	External dependencies needed by engine are placed here. You should not add, delete or modify anything within this directory.
$ENGINE_HOME/log/	Engine log files can be found here.
$ENGINE_HOME/plugins/	Java plugins. Plain java (*.java) files can be placed in this directory. Engine will compile and load them (when configured to do so IN ENGINE-CONFIG.XML FILE).

CONFIGURING STREAMHORIZON DEMO

ORACLE DEMO

Oracle deployment comes in three flavours: JDBC, external tables and named pipes. JDBC performs slower than external tables generally, however setup script for external tables requires you to create Oracle directories (as instructed in execution script).

TO TEST ORACLE JDBC DEPLOYMENT

1. Move $ENGINE_HOME/demo_sales/oracle/jdbc_deploy/engine-config.xml file into $ENGINE_HOME/config/ folder.
2. Open engine-config.xml file and change parameters commented out with string "SET ME!". The engine-config-example.xml is provided as an arbitrary example of correctly configured file.
3. Execute Oracle script StreamHorizon/demo_sales/oracle/jdbc_deploy/oracle_create_schema.sql. Please read NOTE's at the beginning of the script prior to execution
4. Copy sample file $ENGINE_HOME /demo_sales/sales_20140107_data.csv and file multiplier script file_multiplier.sh (or .bat) to your <property name="directoryPath"> as configured in engine-config.xml. Execute file_multiplier script in order to create 500 files.
5. NOTE: by default data in fact table (sales_fact) will be stored in your default tablespace. This tablespace (if not big enough) may be filled up during processing and if so will cause errors in StreamHorizon logs ($ENGINE_HOME/log/sh-engine*log). To avoid this either edit file multiplier script file_multiplier to create less than 500 copies of the feed files or extend your default tablespace. Alternatively, change create table statement for sales_fact table to point to tablespace of your choice.
6. Please skip to section RUNING STREAMHORIZON DEMO

TO TEST ORACLE IN EXTERNAL TABLE (BULK LOAD) MODE

1. Move $ENGINE_HOME/demo_sales/oracle/ext_table_deploy/engine-config.xml into $ENGINE_HOME/config/ folder.
2. Open engine-config.xml and change parameters commented out with string "SET ME!". The engine-config-example.xml is provided as an arbitrary example of correctly configured file.
3. If you have already executed $ENGINE_HOME/demo_sales/oracle/jdbc_deploy/oracle_create_schema.sql as part of install of JDBC setup please skip next step.
4. Execute Oracle script $ENGINE_HOME/demo_sales/oracle/jdbc_deploy/oracle_create_schema.sql. Please read NOTE's at the beginning of the script prior to execution.
5. Execute Oracle script $ENGINE_HOME/demo_sales/oracle/ext_table_deploy/externalTableCreate.sql. Please read NOTE's at the beginning of the script prior to execution
6. Copy sample file $ENGINE_HOME/demo_sales/sales_20140107_data.csv and file multiplier script file_multiplier.sh (or .bat) to your <property name="directoryPath"> as configured in engine-config.xml. Execute file multiplier script in order to create 500 files.
7. NOTE: by default data in fact table (sales_fact) will be stored in your default tablespace. This tablespace (if not big enough) may be filled up during processing and if so will cause errors in StreamHorizon logs ($ENGINE_HOME/log/sh-engine*log). To avoid this either edit file multiplier script file_multiplier to create less than 500 copies of the feed files or extend your default tablespace. Alternatively, change create table statement for sales_fact table to point to tablespace of your choice.
8. Please skip to section RUNING STREAMHORIZON DEMO

NOTE: On some I/O systems you may get better performance if you remove following setting from engine-config.xml:

<property name="bulk.file.acceptance.timeout.millis">2000</property>

Note that in that case in some (less performant) I/O systems you may get the following error:

java.nio.file.DirectoryIteratorException: java.nio.file.AccessDeniedException

This occurs as DB loader threads start consuming files before ETL threads finishes writing them. If you see this error please set property "bulk.file.acceptance.timeout.millis" to adequate value which will give enough time to ETL threads to persist bulk files before DB threads try to consume them. This setting is subject to performance of your I/O system and need be tuned on every deployment.

ORACLE SUBSECOND QUERY LATENCY AND AGGREGATIONS

StreamHorizon enables you to achieve data throughput of 1+ million records per second on single commodity server. In most deployments 1million+ throughput exceeds the rate at which data is produced. If your data source produces 300K records per second (rather than 1 million records per second) it may be beneficial that you create indexes and aggregations.

Indexing fact table heavily will enable Oracle to return majority of user queries within 0-2 seconds. Creating indexes and aggregations will reduce throughput of StreamHorizon, however, as long as throughput of StreamHorizon is higher than rate of production of your source data this is desirable design approach. By indexing and aggregating fact tables you will achieve low query latency which outperforms most of MOLAP (OLAP) cube deployments.

Testing indexes and aggregation features:

1. Note that Aggregation and Indexing demo is available only in external table operation mode. With few small adjustments you can alter JDBC deployment to create aggregations and perform indexing as well if required.

2. Truncate fact table sales_fact "truncate table sales_fact"

3. Execute $ENGINE_HOME/demo_sales/oracle/ext_table_deploy/oracleCreateAggregations.sql script. Script will create required indexes

4. Update engine-config.xml by changing aggregation flag from value 0 to value 1:

   <command type="sql">

   CALL p_sh_external_table_load(${ffn[2]}, ${bulkProcessingThreadID},'${bulkFileName}', 1)

   </command>

5. Run StreamHorizon demo. Note that you may need to ensure that feeds are copied from archive to source directory before you execute demo.

6. After load has finished execute statistics for database schema. Statistics script is located in $ENGINE_HOME/demo_sales/oracle/ext_table_deploy/oracleStatistics.sql file

7. Run queries against fact and aggregated fact table in order to analyze low latency of sql queries, sample queries are supplied in $ENGINE_HOME/demo_sales/oracle/ext_table_deploy/oracleQueries.sql

TO TEST ORACLE IN PIPE LOAD MODE

1. Move $ENGINE_HOME/demo_sales/oracle/pipe_deploy/engine-config.xml into $ENGINE_HOME/config/ folder.
2. Open engine-config.xml and change parameters commented out with string "SET ME!". The engine-config-example.xml is provided as an arbitrary example of correctly configured file.
3. If you have already executed $ENGINE_HOME/demo_sales/oracle/jdbc_deploy/oracle_create_schema.sql as part of install of JDBC setup please skip next step.
4. Execute Oracle script $ENGINE_HOME/demo_sales/oracle/jdbc_deploy/oracle_create_schema.sql. Please read NOTE's at the beginning of the script prior to execution.
5. Copy sample file $ENGINE_HOME/demo_sales/sales_20140107_data.csv and file multiplier script file_multiplier.sh (or .bat) to your <property name="directoryPath"> as configured in engine-config.xml. Execute file multiplier script in order to create 500 files.
6. NOTE: by default data in fact table (sales_fact) will be stored in your default tablespace. This tablespace (if not big enough) may be filled up during processing and if so will cause errors in StreamHorizon logs ($ENGINE_HOME/log/sh-engine*log). To avoid this either edit file multiplier script

file_multiplier to create less than 500 copies of the feed files or extend your default tablespace. Alternatively, change create table statement for sales_fact table to point to tablespace of your choice.

7. Copy directory sqlldr located in $ENGINE_HOME/demo_sales/oracle/pipe_deploy into your $ENGINE_HOME/bin directory. Ensure that StreamHorizon has required priviledges for sqlldr directory.

8. Change following parameters in $ENGINE_HOME/bin/sqlldr/pipe_invoke.sh file:
   a. Enter your valid SID: export ORACLE_SID=orcl
   b. Enter your Oracle install path: export PATH=${PATH}:/db/oracle/product/11.2.0/bin
   c. Enter your logging credentials and Oracle SID: userid=sh/sh@orcl
   d. Enter path to sqlldr directory: control=$ENGINE_HOME/bin/sqlldr/

9. Note that all t*.ctl files located in $ENGINE_HOME/bin/sqlldr dirctiry have hardcoded values for table subpartition name, sub attribute, booking_date attribute and file_id attribute. They need not be changed. In case that you wish to run more than 10 parallel ETL threads (<etlProcessingThreadCount>) you will need to create more t*.ctl files. Demo example runs with maximum of 10 ETL threads.

10. For simplicity reasons, logs generated by *ctl files will be located in your $ENGINE_HOME/bin directory this should be changed for strategic deployments as you see fit.

11. If StreamHorizon in pipe mode appears 'stuck' after startup that is probably because there is nothing to drain the pipes created by StreamHorizon. This means that pipe_invoke.sh and/or t*.ctl aren't configured properly (data is not inserted into Oracle database). To verify that this is the case you can output to your terminal data content of the pipes. If you see the data (coming out of pipe files) that implies that StreamHorizon is processing data correctly and that Oracle isn't accepting the data. Most likely, this implies the issue with pipe_invoke.sh file. StreamHorizon pipes are located in $ENGINE_HOME/data/instance_0 directory. One 'bulkFile#.pipe' file is created for every ETL thread. Draining pipe files of data can be achieved by issuing command: cat $ENGINE_HOME/data/instance_0/bulkFile0.pipe

12. Please skip to section RUNING STREAMHORIZON DEMO

MSSQL DEMO

MSSQL demo achieves loads of 1 million records per second by utilizing partitioned heap table (table with no references or indexes). Note that fact table (sales_fact) is partitioned and is utilizing non PRIMARY data and log files. This is in order to achieve higher I/O parallelism/throughput. You can achieve similar throughput (1million per second) into MSSQL table if sales_fact wasn't partitioned. Supplied demo is using partitioned table (sales_fact).

INSTALLING JDBC DRIVERS FOR MSSQL

1. Please install MSSQL JDBC driver of your choice
2. If installing Microsoft JDBC drivers:
   o Microsoft driver (at the time of writing) is named sqljdbc_3.0.1301.203_enu.exe and can be found at following location: http://www.microsoft.com/en-gb/download/details.aspx?id=28562
   o Place adequate sqljdbc_auth.dll file into $ENGINE_HOME\lib directory
   o Change PATH system variable to contain path to sqljdbc_auth.dll file
   o Copy sqljdbc4.jar file into $ENGINE_HOME\lib\directory.

3. Ensure that your MSSQL instance runs in mixed authentication rather than Windows authentication mode (alternatively you can alter script provided to work in Windows authentication mode, script: $ENGINE_HOME\demo_sales\mssql\mssql_create_db.sql)

4. Note that JDBC connectivity connection string <jdbcUrl> located in ($ENGINE_HOME\demo_sales\mssql\engine-config_JDBC.xml) contains following tuned parameters:
   o Maximum packet size (PacketSize=-1)
   o Set implicit transaction mode (TransactionMode=implicit)
   o Attempt to use NCALRPC protocol (protocol=ncalrpc). NCALRPC setting should be used only if database and StreamHorizon instance are hosted on a single server. Alternatively it should be removed from <jdbcUrl> element ($ENGINE_HOME\demo_sales\mssql\engine-config_JDBC.xml)

StreamHorizon supports official Microsoft JDBC driver for MSSQL and also JTDS (http://jtds.sourceforge.net/). When using JTDS be aware that connection URL format is different than the one for Microsoft driver.

TO TEST MSSQL JDBC DEPLOYMENT

1. Move $ENGINE_HOME\demo_sales\mssql\engine-config_JDBC.xml into StreamHorizon\config\ folder. After move file should be renamed from engine-config-JDBC.xml into engine-config.xml

2. Open engine-config.xml and change parameters commented out with string "SET ME!". The engine-config-JDBC-example.xml is provided as an arbitrary example of correctly configured file.

3. If required change default server:port setting from engine-config.xml (element <jdbcUrl>) from "localhost:1433" to adequate values

4. Ensure that your MSSQL instance runs in mixed authentication rather than Windows authentication mode (alternatively you can alter script provided to work in Windows authentication mode, script: $ENGINE_HOME\demo_sales\mssql\mssql_create_db.sql)

5. Execute MSSQL script $ENGINE_HOME\demo_sales\mssql\mssql_create_db.sql. Please read NOTE's at the beginning of the script prior to execution.

6. Copy sample file $ENGINE_HOME\demo_sales\sales_20140107_data.csv and file multiplier script file_multiplier.sh (or .bat) to your <property name="directoryPath"> as configured in engine-config.xml. Execute file multiplier script in order to create 500 files.

7. Please skip to section RUNING STREAMHORIZON DEMO

8. NOTE: majority of systems run with JDBC batch size of 2000 records. In order to tune your throughput feel free to default batch size:

<property name="jdbc.bulk.loading.batch.size">2000</property>

TO TEST MSSQL BULK INSERT DEPLOYMENT

1. Move $ENGINE_HOME\demo_sales\mssql\engine-config-BULK-INSERT.xml into StreamHorizon\config\ folder. After move file should be renamed from engine-config-BULK-INSERT.xml into engine-config.xml

2. Open engine-config.xml and change parameters commented out with string "SET ME!". The engine-config-BULK-INSERT-example.xml is provided as an arbitrary example of correctly configured file.

3. If required change default server:port setting from engine-config.xml (element <jdbcUrl>) from "localhost:1433" to adequate values

4. Ensure that your MSSQL instance runs in mixed authentication rather than Windows authentication mode (alternatively you can alter script provided to work in Windows authentication mode, script: $ENGINE_HOME\demo_sales\mssql\mssql_create_db.sql)

5. Execute MSSQL script $ENGINE_HOME\demo_sales\mssql\mssql_create_db.sql. Please read NOTE's at the beginning of the script prior to execution. This step need be omitted if database has already been created.

6. Copy sample file $ENGINE_HOME\demo_sales\sales_20140107_data.csv and file multiplier script file_multiplier.sh (or .bat) to your <property name="directoryPath"> as configured in engine-config.xml. Execute file multiplier script in order to create 500 files.
7. Change following values from $ENGINE_HOME\demo_sales\mssql\bulk_loader.bat file:
8. Change "server\myInstance" into adequate "server[\instance]" value
9. Change "G:\Data\StreamHorizon\bulk\" into path to your bulk directory as configured in <bulkOutputDirectory> element of engine-config.xml. Note that "%1" needs to remain as is in order for parameters to be adequately passed to the bulk insert routine.
10. Change "G:\Data\StreamHorizon\error\" into path to your error directory as configured in < errorDirectory> element of engine-config.xml. Note that "%1" needs to remain as is in order for parameters to be adequately passed to the bulk insert routine.
11. You may want to alter batch size setting ROWS_PER_BATCH=11000 of bulk_loader.bat file. However, in most cases default setting will deliver highest throughput.
12. Please skip to section RUNING STREAMHORIZON DEMO

NOTE: On some I/O systems you may get better performance if you remove following setting from engine-config.xml:

<property name="bulk.file.acceptance.timeout.millis">2000</property>

Note that in that case in some (less performant) I/O systems you may get the following error:

java.nio.file.DirectoryIteratorException: java.nio.file.AccessDeniedException

This occurs as DB loader threads start consuming files before ETL threads finishes writing them. If you see this error please set property "bulk.file.acceptance.timeout.millis" to adequate value which will give enough time to ETL threads to persist bulk files before DB threads try to consume them. This setting is subject to performance of your I/O system and need be tuned on every deployment.

INDEXING WITH MSSQL

MSSQL delivers query latency of 0-2 seconds if fact table is adequately indexed. This eliminates need to deliver MOLAP(OLAP) as a part of Data Warehousing solution. Load throughput into MSSQL heap table is approximately 1 million records per second, if fact table is indexes to deliver low latency SQL queries by end users load performance is reduced to 333K records per second. This still provides throughput of 1.2 million records per hour (on out of the box (systems which aren't I/O tuned)).

*   If your application requires 1million records per second throughput indexes should be disabled before and rebuilt after the data load.
*   If your data sources supply your target database with data at pace less of 28K-50K records per second you may want to leave indexes online during your data load. This eliminates need to disable and rebuild indexes.

Note: Performance of database load throughput doesn't linearly depend on number of indexes created on the fact table. Our findings are that fact table with one and three indexes has negligible different data load throughput.

Note: All given throughput metrics do not assume SSD but rather Tier 2 SAN storage. These figures are for systems which aren't tuned but are rather utilized with their default configuration 'out of the box'. Tuning MSSQL I/O may deliver throughput figures higher than those listed in this document.

Script containing create, drop, rebuild index statements and test sql queries is located in $ENGINE_HOME\demo_sales\mssql\mssql_IndexesAndQueries.sql file.

Conducted tests (on I/O & network 'out of the box' (not optimized) systems utilising SAN Tier 2 (non SSD) storage following throughput can be achieved (42 core server):

Load Method	Indexes	Number of Etl threads	Number of DB loader threads	Throughput per second
Bulk insert	no indexes (or indexes disabled)	48	40	1.1 million
Bulk insert	3 indexes	40	40	333K
JDBC insert	no indexes (or indexes disabled)	48	0	141K
JDBC insert	3 indexes	48	0	133K

MYSQL DEMO

MySQL deployment comes in two flavours, JDBC and reading rows from a text file (LOAD DATA INFILE).

It is very important to understand how to tune MySQL database engine for high throughput and bulk loading. Please refer to official MySQL documentation.

TO TEST MYSQL JDBC DEPLOYMENT

1. Move $ENGINE_HOME/demo_sales/mysql/engine-config.xml file into $ENGINE_HOME/config/ folder.
2. Open engine-config.xml file and change parameters commented out with string "SET ME!"
3. Execute script $ENGINE_HOME/demo_sales/mysql_create_schema.sql.
4. Make sure that element of engine-config.xml is set to <bulkLoadDefinition outputType="jdbc">
5. Make sure that <bulkLoadInsert> element has adequate <command> tag uncomented. Value starting with 'insert into sales_fact….' Should be uncommented if you run in JDBC mode.
6. Copy sample file $ENGINE_HOME/demo_sales/sales_20140107_data.csv and file multiplier script file_multiplier.sh (or .bat) to your <property name="directoryPath"> as configured in engine-config.xml. Execute file multiplier script in order to create 500 files.
7. Please skip to section RUNNING STREAMHORIZON DEMO

TO TEST MYSQL IN BULK LOAD MODE

1. Execute first four steps needed for running MySQL JDBC demo (see above)
2. In $ENGINE_HOME/config/engine-config.xml change <bulkLoadInsert> so that it uses first command (with LOAD DATA INFILE syntax)
3. Make sure that element of engine-config.xml is set to <bulkLoadDefinition outputType="file">
4. Make sure that <bulkLoadInsert> element has adequate <command> tag uncomented. Value starting with 'LOAD DATA INFILE….' Should be uncommented if you run in JDBC mode.
5. Copy sample file $ENGINE_HOME/demo_sales/sales_20140107_data.csv and file multiplier script file_multiplier.sh (or .bat) to your <property name="directoryPath"> as configured in engine-config.xml. Execute file multiplier script in order to create 500 files.
6. Please skip to section RUNNING STREAM HORIZON DEMO

NOTE: On some I/O systems you may get better performance if you remove following setting from engine-config.xml:

<property name="bulk.file.acceptance.timeout.millis">2000</property>

Note that in that case in some (less performant) I/O systems you may get the following error:

java.nio.file.DirectoryIteratorException: java.nio.file.AccessDeniedException

This occurs as DB loader threads start consuming files before ETL threads finishes writing them. If you see this error please set property "bulk.file.acceptance.timeout.millis" to adequate value which will give enough time to ETL threads to persist bulk files before DB threads try to consume them. This setting is subject to performance of your I/O system and need be tuned on every deployment.

RUNNING STREAMHORIZON DEMO

Before we run StreamHorizon we need to set StreamHorizon threadpools. StreamHorizon has two threadpools, ETL and DB threadpools.

- If you run test via JDBC (for Oracle, MSSQL and MYSQL) your ETL threadpool need be greater than zero while DB Threadpol need be set to zero.

- If you run in BULK MODE (BULK for MYSQL or EXTERNAL TABLES for Oracle or BULK INSERT for MSSQL) both, ETL and DB threadpools whould be set as advised in engine-config.xml.

- ETL threadpool (number of threads) is set by assigning number to element <etlProcessingThreadCount> of $ENGINE_HOME/config/engine-config.xml

- DB threadpool (number of threads) is set by assigning number to element <databaseProcessingThreadCount> of $ENGINE_HOME/config/engine-config.xml NOTE: for JDBC deployment this number should be set to zero.

WHAT IS DESIRED THREADPOOL SIZE?

Rough estimate is that (assuming that your server is not under heavy load) you should set threadpool size to number of cores of your server. So, for 24 core server for JDBC following settings should be used:

- <etlProcessingThreadCount>24</etlProcessingThreadCount>

- <databaseProcessingThreadCount>0</databaseProcessingThreadCount>

For BULK LOAD deployment following should be used:

- <etlProcessingThreadCount>24</etlProcessingThreadCount>

- <databaseProcessingThreadCount>24</databaseProcessingThreadCount>

Please modify your threadpool sizes and save changes in engine-config.xml file.

Maximum database load parallelism <databaseProcessingThreadCount> element for Oracle is 50 as fact table has 50 subpartions, for MSSQL it is 40 as fact table has 40 partitions.

For JDBC mode may want to test tuning of JDBC batch size <property name="jdbc.bulk.loading.batch.size"> property. Oracle usually delivers best performance with setting of 10000 while MSSQL usually performs best with setting of 2000.

Other parameters critical for tuning are:

- <property name="read.buffer.size.mb">10.0</property> This parameter should ideally be set to value (in MB) which is bigger or equal to the size of your largest data file size. If I/O system is saturated due to high number of etl threads <etlProcessingThreadCount> it may be beneficial to set "read.buffer.size.mb" to value less than average file size (approximately 1/10 of average file size). Most suitable buffer size is determined by testing of different buffer sizes

- <property name="write.buffer.size.mb">10.0</property> This parameter should be set to be just over size (in MB) of largest bulk file generated by StreamHorizon. If I/O system is saturated due to high number of etl threads <etlProcessingThreadCount> it may be beneficial to set " write.buffer.size.mb" to value less than average file size (approximately 1/10 of average file size). Most suitable buffer size is determined by testing of different buffer sizes

STARTING STREAMHORIZON INSTANCE

Please start StreamHorizon instance by executing $ENGINE_HOME/bin/start.sh (or .bat)

Please read "REALISTIC TESTING" section below

CHECKING THROUGHPUT OF YOUR SERVER FOR ORACLE & MSSQL DATABASES

- Use select * from sh_dashboard to view statistics/throughput of JDBC or file/feed processing (case when output feeds (files) are created without being loaded into the database)
- Use select * from sh_dashboard_bulk to view statistics/throughput of load via external tables or SQL*Loader

Views return statistics of total number of data records loaded, time window and achieved throughput per second of StreamHorizon instance(s). You will notice that throughput of StreamHorizon increases as batch progresses.

Note that views sh_dashboard* will only show throughput metrics for data loaded for latest start of StreamHorizon engine instance. This is because $ENGINE_HOME/config/engine-config.xml has '<OnStartupCommands> SQL call to truncate metrics table (<command type="sql">truncate table sh_metrics</command>). If you wish to see metrics over many runs of StramHorizon instance uncomment or remove this setting.

OTHER METRICS & VIEWS

1. To see detailed performance & activity logs look at sh_metrics table (or view sh_all_metrics). This table contains one record for every file processed if you run in JDBC mode and two records (one created by ETL thread and one created by DB thread) if you run in Oracle external table mode. To troubleshoot please look at $ENGINE_HOME/log/sh-engine*log

2. View sh_all_erros will show all feeds which have had errors during processing
3. View sh_dashboard_bulk shows throughput figures when StreamHorizon runs in external table mode (Oracle) or in Bulk Insert mode (MSSQL)
4. View sh_dashboard_db_loader_mode shows throughput figures when StreamHorizon runs only in db loader mode (when number of etl threads < etlProcessingThreadCount> is set to zero)
5. View sh_dashboard_file_2_file shows performance of creating output feeds (bulk feeds or other type of feeds (files)) in case when data isn't inserted into database.
6. View sh_dashboard_jdbc shows throughput figures when StreamHorizon is running in JDBC mode
7. View sh_dashboard_pipe shows throughput figures when StreamHorizon is running in pipe mode. This mode is available only for Oracle database running on Unix/Linux platforms.
8. View sh_sales_fact selects all data from sales_fact table
9. View sh_allsales_fact_count returns number of records loaded in sales_fact table
10. Views sh_etl_* are advanced views used in monitoring of etl and database load performance

ANALYZING DELAYS IN ETL AND DB LOAD

Use select * from sh_etl_bulk to analyze time delays introduced by ETL or DB threads. This view is intended to work only for bulk load modes of operation for single instance monitoring. View shows only events for a last startup of the StreamHorizon engine.

CUSTOMIZE METRICS & LOGGING

You may customize all above tables & views to your liking by changing calls <afterFeedProcessingCompletion> & <onBulkLoadCompletion> of default $ENGINE_HOME/config/engine-config.xml configuration.

CHECKING THROUGHPUT OF YOUR SERVER FOR MYSQL DATABASE

Use *select * from sh_metrics* to see throughput of StreamHorizon instance when using MySQL database.

REALISTIC TESTING

As first run of StreamHorizon will form all cardinalities of the all dimensions from scratch average throughput will be ~3 times slower than actual.

After first successful run, after dimensions are repopulated kill instance of StreamHorizon. Copy files back from <archiveDirectory> to <property name="directoryPath">. Start instance of StreamHorizon again and observe the throughput. To avoid repeating this every time you run StreamHorizon write a script which moves files from archive to source directory and change engine-config.xml to invoke it on every startup of StreamHorizon engine instance:

```
<onStartupCommands>

      <command type="shell">/yourMoveScriptComesHere.sh (or .bat)</command>

      <command type="sql">truncate table sales_fact</command>

      <command type="sql">truncate table sh_metrics</command>

</onStartupCommands>
```

NOTE: if you run one set of files, than pause let's say an hour (don't kill StreamHorizon instance) and then run another batch of files view sh_dashboard will show very low throughput. This is because view calculates time window of first file processed by instance after startup and last file processed since instance startup. If batch was loaded in 1 minute, if idle time of server server was 98 minutes, and if second batch was running for 1 minute, throughput shown by sh_dashboard will be 2% of actual throughput of the StreamHorizon.

To avoid this simply observe throughput during the batch immediately after all files are loaded and make sure you restart StreamHorizon before next test.

StreamHorizon increases data throughput as batch progresses.

THROUGHPUT GUIDELINES

In our testing environments we have used commodity Windows, Linux and Solaris servers.

StreamHorizon performs better if deployed on the same server as database server. It is resource lightweight and can efficiently coexist/operate with database instance.

We have used 24 core servers and have achieved througput above 800K per second for Oracle External Tables (running with 24 ETL and 24 DB threads) and throughput above 600K for JDBC Oracle setup running with 26 ETL and zero DB threads.

If you deploy StreamHorizon on the same server as database server if your server has 32 cores number of ETL and DB threads should be around 26. These, of course, are rough guidelines; you should perform multiple tests by adjusting ETL and DB thread count in order to determine optimal settings for your hardware configuration.

REALISTIC DATA

Our tests utilize data typical for Time Series Analysis, Credit Risk, Market Risk and PnL datasets.

File sizes are large (but realistic), data is partially 'sorted' because of the way Tick and Risk data is produced. Such nature of data increases throughput of the StreamHorizon (or any other data processing & ETL platform).

If you are happy with StreamHorizon next step would be to configure it to work with your data model and data feeds.

We are happy to help you in setting up, deploying and delivering your StreamHorizon Instance.

TUNING STREAMHORIZON

Note that extensive sample xml configuration which presents more detailed ability of the platform is available in $ENGINE_HOME/config/streamHorizonQuickXMLTutorial_engine_config.xml file. XML file is full of descriptive comments about some useful features of the platform of which many are not included in this demo due to simplicity.