Configuration

Minimum Requirements

See the Cluster Requirements.

Minimum Node Requirements

NOTE: These requirements are recommended for production clusters.

Processors per node	• small = 2 cores • medium = 4 cores • large = 8 cores or more
Memory per node	• small = 16 GB of RAM • medium = 64 GB or RAM • large = 128 GB of RAM
Number of nodes and concurrency	• 2 small nodes = 4 concurrent queries • 2 medium nodes = 64 concurrent queries • 2 large nodes = 256 concurrent queries
HBase memory per node	This setting will vary depending on the type of queries and concurrency level. Use these as starting values: • small: Region Server Java Heap Size = 2 GB • medium: Region Server Java Heap Size = 8 GB • large: Region Server Java Heap Size = 16 GB

HBase Configuration and Fine Tuning

Installer Modifications to HBase

The Trafodion Installation scripts (traf_cloudera_mods, traf_ambari_mods) modify the HBase configuration with the following settings:

HBase Configuration Property	Installer Setting
hbase.master.distributed.log.splitting	false
hbase.coprocessor.region.classes	org.apache.hadoop.hbase.coprocessor.transactional.TrxRegionObserver, org.apache.hadoop.hbase.coprocessor.transactional.TrxRegionEndpoint, org.apache.hadoop.hbase.coprocessor.AggregateImplementation
hbase.hregion.impl	org.apache.hadoop.hbase.regionserver.transactional.TransactionalRegion

hbase.regionserver.region.split.policy	org.apache.hadoop.hbase.regionserver.ConstantSizeRegionSplitPolicy **NOTE:** This setting causes region splitting to occur only when the maximum file size is reached.
hbase.snapshot.enabled	true
hbase.regionserver.region.transactional.hlog	false
hbase.regionserver.region.transactional.tlog	true
hdfs.namenode.java.heapsize **(Cloudera only)**	1073741824 (1GB)
hdfs.secondary.namenode.java.heapsize**(Cloudera only)**	1073741824 (1GB)

Recommendations

It is recommended that you modify these HBase configuration settings as described below.

HBase Configuration Property	Recommended Setting	Notes
hbase.rpc.timeout	10 minutes	This setting depends on the tables' size. Sixty (60) seconds is the default. Increase this value for big tables. Make it the same value as hbase.client.scanner.timeout.period. We have found that increasing the setting to six-hundred (600) seconds will prevent many of the timeout-related errors we encountered, such as OutOfOrderNextException errors.
hbase.client.scanner.timeout.period	10 minutes	Similar to the above setting. Sixty (60) seconds is the default. Depending on the size of a user table, we have experienced timeout failures on count(*) and update statistics commands from this setting. The underlying issue is the length of the execution of the coprocessor within HBase.
hbase.snapshot.master.timeoutMillis and hbase.snapshot.region.timeout	10 minutes	HBase's default setting is 60000 milliseconds. If you experience timeout issues with HBase snapshots when you use the Trafodion Bulk Loader or other statements, you can set the value for these two HBase properties to 10 minutes (600,000 milliseconds).
hbase.hregion.max.filesize	107374182400 bytes	HBase's default setting is 10737418240 (10 GB). We have increased the setting to 107374182400 (100 GB), which reduces the number of HStoreFiles per table and appears to reduce disruptions to active transactions from region splitting.
hbase.hstore.blockingStoreFiles	10	CDH 4.5 has this value default to 7. We have increased this setting to 10, which appears to be the default in earlier versions of HBase.
hbase.regionserver.handler.count	*See notes.*	This setting should match the number of concurrent sessions (mxosrvr). The default is 10.

Disabling HBase Region Rebalancing

The Trafodion HBase transactional region endpoint coprocessor (`org.apache.hadoop.hbase.coprocessor.transactional.TrxRegionEndpoint`) does not currently handle the movement of a region when a Trafodion transaction is active. If a region is rebalanced when a Trafodion transaction is active, it might disrupt the transaction, causing the transaction to be rolled back and error 97 to be returned. To avoid disruptions to transactions, you should disable HBase region balancing while Trafodion is running.

Using the HBase shell, set the HBase property for `balance_switch` to `false` to disable the movement of a region from one server to another. For example:

```
# hbase shell
hbase(main):002:0> balance_switch false
true  <-- Output will be the last setting of the balance_switch value
0 row(s) in 0.0080 seconds
```

When set through the HBase shell, the `balance_switch` property will be persisted for subsequent invocations of HBase. This applies to HBase versions 0.95 and later.

NOTE: To optimize the performance of a Trafodion system, we recommend that you enable HBase balancing during a maintenance window. For recommendations, see Maintenance Window.

Hadoop Distribution Configuration and Fine Tuning

It is recommended that you modify these HDFS heap size settings as described below.

HDFS Property	Recommended Setting	Notes
DataNode Java Heap Size	2 GiB	Use this setting for a large configuration.
NameNode Java Heap Size	2 GiB	Use this setting for a large configuration.
Secondary NameNode Java Heap Size	2 GiB	Use this setting for a large configuration.

Trafodion Configuration and Fine Tuning

Connectivity

In the Trafodion software, the heap size for Database Connectivity Services (DCS) is set by default to 2GB. However, in our testing we found that limiting the DcsMaster to 64MB of heap size resulted in better throughput.

Typically, OLTP workloads require many connections. We found that setting 2 DCS servers per node provided the best performance for a cluster with 16 GB of RAM per node. For an 8-node cluster with 96 GB of RAM per node and with the DcsMaster's heap size set to 64MB, 128 concurrent sessions per node was optimal.

In ZooKeeper, the default maximum number of connections is 60. Depending on the number of ZooKeeper servers you have running and the number of DCS servers (that is, MXOSRVRs) you have started, that number might be too low. Therefore, it is recommended that you set the `maxClientCnxns` para meter in ZooKeeper to 0 to disable the maximum number of connections. For more information about configuring and using DCS, see the *Trafodion Database Connectivity Services Reference Guide*.

For ODBC connections on Linux workstations, we found that using at least version 2.3.*x* of the third-party driver manager, unixODBC, provided the best throughput. For more information about how to install the Trafodion ODBC Driver for Linux, see the *Trafodion Client Installation Guide (pdf, 1.12 MB)*.

Table Maintenance

It is important to maintain current statistics for your Trafodion tables. This allows the compiler to generate optimal Trafodion execution plans in a performant manner. The update statistics SQL command will generate up-to-date statistics for a table. To perform an update statistics command, we recommend a basic query.

```
trafci
  UPDATE STATISTICS FOR TABLE <table name> ON EVERY COLUMN SAMPLE;
  exit;
```

Update statistics should be performed whenever there are significant changes to the content of your Trafodion tables. For example:

```
[trafodion@n1~]$ trafci
Welcome to Trafodion Command Interface
Copyright(C) 2013-2014 Hewlett-Packard Development Company, L.P.
Host Name/IP Address: localhost:37800
User Name: zz
Connected to Trafodion
SQL>UPDATE STATISTICS FOR TABLE TRAFODION.SEABASE.TABLE1 ON EVERY COLUMN SAMPLE;
--- SQL operation complete.
SQL>exit;
```

Data Load

To maximize the performance of loading data into Trafodion tables, we recommend using the UPSERT USING LOAD syntax.

UPSERT USING LOAD loads data without going through a transaction.

Using INSERT to load data will go through a transaction. This could cause errors if inserting a large amount of data.

```
trafci
  create table <table creation syntax>;
  upsert using load into <table name> <upsert syntax>;
  select * from <table name>;
  exit;
```

```
[trafodion@n1 scripts]$ trafci
Welcome to Trafodion Command Interface
Copyright(C) 2013-2014 Hewlett-Packard Development Company, L.P.
Host Name/IP Address: localhost:37800
User Name: zz
Connected to Trafodion
SQL>create table trafodion.seabase.table1(a int not null, b char(10), primary key(a));
--- SQL operation complete.
SQL> upsert using load into trafodion.seabase.table1 values (1,'a'), (2,'b');
 --- SQL operation complete.
SQL>select * from trafodion.seabase.table1;
A           B
----------- ----------
          1 a
          2 b
--- 2 row(s) selected.
SQL>exit;
```

For more information, see Data Loading.

JVM Heap Size

The default maximum heap size is 2 GB. This setting results in a large allocation of resident memory space. We have reduced this value to 512 MB for OLTP workloads to reduce the default memory consumption of mxosrvr and tdd_arkcmp processes.

Table Partitioning and Salting

We choose to load all user tables with a pre-salting equal to the number of region servers on the system. For example: `create table … salt using <n> partitions` where <n> is the number of regions servers on the system.

OLTP Performance Improvements

For improved OLTP performance, HBase and the Trafodion DTM should be configured to match the same level of concurrency as DCS.

- In the HBase configuration: hbase.regionserver.handler.count=128
- In `$SQ_HOME/etc/ms.env`: TM_JAVA_THREAD_POOL_SIZE=128

Maintenance Window

HBase, in its normal operation, performs memstore flushes and compactions per region. Region splits and balancing also occur to help optimize HBase load and performance. However, the frequency and duration of those activities can adversely affect overall Trafodion transaction execution.

The Trafodion HBase transactional region endpoint coprocessor (`org.apache.hadoop.hbase.coprocessor.transactional.TrxRegionEndpoint`) currently does not support the dynamic movement of regions while a Trafodion transaction is active. To prevent region splits from occurring when transactions are active, a Trafodion HBase transactional region observer (`org.apache.hadoop.hbase.coprocessor.transactional.TrxRegionObserver`) delays the region split. To prevent region balancing from occurring when transactions are active, we recommend that you disable region balancing through the HBase shell. See Disabling HBase Region Rebalancing.

HBase will also flush the memstore for a region when HBase services are being terminated. HBase also compacts needed regions when HBase services are initiated.

To optimize the performance of a Trafodion system, we recommend that you add a routine maintenance window to your production schedule. This will allow HBase to self-manage its regions in a window where no Trafodion transactions can execute. This also helps HBase manage the optimal location, size, and format of its regions.

We recommend that you follow these steps in the maintenance window:

1. Allow Trafodion SQL queries to quiesce.
2. Stop Trafodion by executing `sqstop`. See Stopping Trafodion. This stops the Trafodion instance and all DCS services.
3. Stop the HBase services using your Hadoop distribution's administrator GUI.
4. Start the HBase services using your Hadoop distribution's administrator GUI.
5. Enable HBase balancing through the HBase shell and allow time for regions to be balanced.
   ```
   [trafodion@n1 ~]$ hbase shell
   hbase(main):002:0> balance_switch true
   ```
6. Disable HBase balancing through the HBase shell for continued normal execution.
   ```
   [trafodion@n1 ~]$ hbase shell
   hbase(main):002:0> balance_switch false
   ```
7. Start Trafodion by executing `sqstart`. See Starting Trafodion. This starts the Trafodion instance and all DCS services.
8. Perform update statistics for recently used tables. For tips, see Updating Statistics on Trafodion Tables. This ensures that optimal Trafodion SQL plans are generated through the Trafodion SQL compiler.
9. Resume normal SQL work.