clustrix_import

clustrix_import is a parallel import tool to import MySQL dumps into a Xpand Cluster. clustrix_import is the recommended tool to load data into a Xpand cluster as quickly and efficiently as possible, with the following main goals:

- Take full advantage of all cluster resources by doing inserts in parallel
- Ensure proper balancing of tables and indexes across the cluster by setting appropriate slice sizes and index distribution

In this document, we will briefly describe how clustrix_import works, how it is best used, and overview the numerous options available to tune and configure your import.

Sample Usage of clustrix_import

clustrix_import located in the /opt/clustrix/bin directory, is highly configurable, see options below. Once you generated a dump file, simply run clustrix_import :

Local Invocation on Node

```
shell> clustrix_import -u importuser -p sekrit -i /clustrix
/bigdump.sql
```

The simplest use case is to run clustrix_import directly on a node just specifying the dump file. In this case clustrix_import connects through the MySQL socket to obtain the IP for each node.

You should have created a user with the proper permissions for importing. e.g. mysql>grant all on *.* to '*<importuser>*'@'%' identified by '*<importuserpassword>*';

Compressed Dumps

```
shell> zcat bigdb.mysqldump.sql.gz | clustrix_import -i - -H clx -u importuser -p sekrit--scan-for-newlines=0 \
--initial-slices=12 --slice-down-threshold=2
```

If a dump is too large to be uncompressed in place, zcat it into clustrix_import. Note the use of -i to specify standard input. We are also specifying --scan-for-newlines =0 because we know the input is from mysqldump which guarantees each INSERT is on a single line (this optimizes the input scanning thread). Since we cannot scan-ahead to estimate table sizes when reading from standard input we are providing a higher initial slice count so all tables will initially be created with 12 slices (default is 3 times the number of nodes so supposing this were a 4 node cluster, this argument would be unnecessary). The --slice-down-threshold option means that only indexes and base tables less than 2GB will be sliced down (to one slice per node). Note that these particular slice settings are illustrative only and not recommended for any particular usage; default values should provide good slicing, or consult with Xpand Support on your specific import challenges.

Piping Directly From mysqldump

```
shell> mysqldump -h mysqlhost -u root -p mysekrit somedb | tee /tmp/somedb.sql | clustrix_import -i - -H clx -D
somedb -u importuser -p sekrit
```

This example demonstrates pipelining a mysqldump file directly into clustrix_import. Using tee to copy the dump output into a file allows us to restart the import if it fails for some reason without needing to completely redo the dump (we'd need to make sure /tmp is sufficiently large to accommodate the dump). If the dump needed to be restarted we could then use the -S option to start at the table where the failure had occurred:

```
shell> clustrix_import -i /tmp/somedb.sql -D somedb -H clx -u importuser -p sekrit -S
employees
```

Using the Dump Map Options

```
shell> clustrix_import -u importuser -p sekrit -i hugedump.sql --write-dump-map dump.map -H
master_vip
shell> clustrix_import -u importuser -p sekrit -i hugedump.sql --read-dump-map dump.map -H
slave_vip
```

This example shows how to import the same file into two different clusters. For the cluster we ask to have the scan-ahead thread output the table sizing info to dump.map. Note that this file will be finished long before the import completes. Once the dump.map is finished we can start the second import, specifying the dump.map to avoid the additional scan and providing table size info for every table in the dump.

Options

Connection Options

option flag (s)	description	default
--host/-H	hostname or IP to reach the cluster; may also specify comma-delimited list to override automatic detection of IPs through the database (use if cluster is NAT'd)	localhost (mysql socket)
--database/-D	database to import into; must already exist, and assumes that dump does not specify database with USE statement	none
--port/-P	MySQL port to connect to	3306
--user/-u	username with which to connect to the database; must have sufficient privileges to CREATE and INSERT data (or just INSERT if --inserts-only)	logged in linux user
--passwd/-p	password for above user	none
--threads-per-node/-T	number of parallel threads to connect, per node	8
--retry-limit	Number of times to retry a failed statement	30

Input Parsing Options

option flag (s)	description	default
--aggregate/-G	aggregate multiple small insert statements into large ones	don't aggregate
--scan-for-newlines	when enabled, scan lines to detect newlines embedded in strings, which can significantly slow scanning thread for some inputs (particularly that containing many backslashes). mysqldump guarantees that INSERT statements are on a single line, so we can avoid scanning. If input is a file (not standard input), we detect whether input is a mysqldump file and disable if so; otherwise this is enabled.	detect if input is a mysqldump and disable if so

Options Controlling What is Imported

option flag (s)	description	default
--databases/-d	comma separated list of databases to be imported from the dump	import all DBs
--start-byte/-b	seek into dump to specified byte offset (much faster than --start-database/--start-table)	none
--start-database/-s	seek into dump until specified database is found; most useful when restarting a failed import	none
--start-table/-S	seek into dump until specified table is found; most useful when restarting a failed import. Can be specified after --start-database, in which case we first seek to the database, then the table (in case there are tables with the same name in different databases)	none

Insert Options

option flag (s)	description	default

--inserts-only/-I	only run INSERT statements, ignoring DDL, etc.	disabled (run all statements)
--ignore-dup-key/-K	convert INSERT to INSERT IGNORE to ignore any possible duplicate keys; use with caution: mysqldump should not be generating duplicate keys	disabled (dup key will halt import)

Slicing and Distribution Options

option flag(s)	description	default
--no-auto-slice/-A	Disable logic to automatically change slice sizing based on (estimated) size of table	
--slice-down-threshold	Maximum size of table/index (in GB) which will be resliced to reduce slice count after completion (does not apply when not using scan-ahead or --no-auto-slice)	5GB
--initial-slices	When not using scan-ahead, number of slices for initial table creation. Has no effect on slice down.	3x number of nodes
--min-slices	When using scan-ahead, number of slices for initial table creation. When slicing down, reduce slice count no lower than this	number of nodes
--slice-file/-y	Specifies a file consisting of table names and slice counts, which is used to set slicing on initial table creation. Format is one table per line, with format table_name N where N is an integer specifying number of slices. Rule will be applied to any table with that name (no support for specifying database). Slice file may contain a subset of tables in the dump, with unspecified tables obtaining slicing as per above options /defaults. Unless --no-auto-slice is specified, tables specified in slice file may still be resliced upon completion.	

Guardrails

option flag(s)	description	default
--ignore-binlog-checks	Do not check whether binlogs are enabled on the cluster	Halt if binlog is found
--no-log-bin	Set sql_log_bin =false to avoid logging import to binlogs	Import is logged to binlog
--allow-parallel-imports	Override check which prevents running multiple instances of clustrix_import	Prevent multiple instances of clustrix_import

Misc Options

option flag(s)	description	default
--schema-file/-x	As import is performed, write DDL (i.e. CREATE DATABASE, CREATE TABLE) statements to specified file	none
--version/-V	Indicate version of clustrix_import utility and exit	
--verbose/-v	Output additional logs during execution, useful (only) for troubleshooting	
--no-auto-inc-opt	Disable optimization which removes AUTO_INCREMENT property during row insertion; on Xpand, this optimization provides up to 2X performance improvement for inserts of tables with an AUTO_INCREMENT column. Disabling may be preferable in conjunction with --is-mysql	optimization enabled
--write-dump-map	Save to a file the results of the scan-ahead thread which determines the size of each table within the input file	
--read-dump-map	Use the dump file saved with --write-dump-map in place of re-running the scan-ahead thread	

Sample Output from clustrix_import

Initial Output

```
Found Cluster Ips:
 -  10.2.13.44  -
 -  10.2.13.45  -
 -  10.2.13.46  -

2013/08/18-18:51:08 +0000 detected mysqldump input, disabling scan-for-newlines (use --scan-for-newlines=1 to
override)
```

This shows the tool discovering the IPs for each node that it will connect the threads. Since the first line of input indicated mysqldump input we're able to apply the optimization to read full lines at a time without scanning for newlines within strings.

Status Lines

```
2013/08/18-18:51:43 +0000 At 612.1 MB of 47476.2 MB (1.29%), Concurrency: 3, 30s read rate: 19.5 MB/s, Inserts:
16817 rows/sec (inserting)
```

These status lines print out every 5 seconds. The first part of the line just after the time stamp indicates how far the reader thread has read into the file and total size and percentage read, if reading from a file.

Concurrency indicates the number of threads currently executing INSERT statements. If this number is significantly less than the number of threads configured (default 8x number of nodes) this may indicate the reader thread is the bottleneck rather than write speed on the cluster. In this example the read rate of nearly 20MB/s corroborates this suspicion.

The 30 second read rate indicates how quickly the reader thread is reading through the file, measured over the last 30 seconds. Note that the reader thread will pause if INSERT threads are all busy so read rate is also gated by INSERT rate.

The rows/second insert rate is also measured over the last 30 seconds.

The final status, enclosed in parenthesis, will indicate when we are doing something other than inserting. Possible states are:

inserting	reader is parsing input, insert threads running
draining	reader is paused, waiting for inserts on current table to finish, prior to reslice/redistribution step
altering	reader and insert threads paused waiting for ALTER of current table to finish
seeking	when --databases, --start-database, or --start-table are given, indicates reader is scanning input for the first/next applicable database /table
load complete	reader has finished reading input, waiting for INSERT to finish, and possibly any reslice or other rebalancer actions

Optimizing Performance of clustrix_import

Here are some options to explore to optimize performance of clustrix_import:

Slicing

Proper pre-slicing is crucial to both efficient data loading and subsequent high performance parallel query execution. If these are not defined as part of the data import process, the rebalancer will take care of this operation eventually, but it is best done ahead of time. An import operation that does not have proper pre-slicing can end up "racing the rebalancer" when we are writing into slices which are actively being split, multiplying the amount of work asked of the I/O subsystem.

clustrix_import can employ various strategies to ensure optimal slicing. Here are a few strategies that may be employed to ensure adequate slice count for imported tables:

- The initial slice count can be specified so that each table starts with this many slices. This is important because starting with too few slices will lead to racing the rebalancer as described above.
- For representations (tables and indexes) that end up smaller than 5GB (tunable with --slice-down-threshold), the table is sliced back down to one slice per node (or the number of slices specified by --min-slices).
- A slice file consisting of a list of table names and corresponding slice counts can be specified (this should be used in concert with -A to disable auto-slicing, which can otherwise slice down afterwards).

The auto-slicing can be disabled, as described above.

Note that a mysqldump taken from a Xpand system will include SLICES=n within /$* */ comments. clustrix_import will process these comments and set slicing accordingly. This will override --initial-slices and unless -A is specified, scan-ahead and slice-down may still occur.

By default, a slice is 8GB This is the size at which the rebalancer splits slices and is set by the global variable: rebalancer_split_threshold_kb.

INSERT Aggregation

Multi-row inserts are significantly more efficient for Xpand to process. By default, mysqldump will generate multi-row inserts of approximately 1MB each. However, in some cases (e.g. --skip-extended-insert), there is only one row per insert. This might also be the case if the import file is being generated through a means other than mysqldump, such as pgdump as part of a Postgres migration.

In such cases where there is only one row per insert, clustrix_import can aggregate these INSERT statements into a larger multi-row statement before executing on the cluster. While this is not as efficient as having multi-row inserts to begin with (since the clustrix_import utility spends CPU cycles scanning and aggregating), performance is greatly improved vs. running with single-row inserts. This option is enabled with -g as described below.

How clustrix_import Works

clustrix_import is primarily designed to read data generated by the mysqldump tool which generates a series of SQL queries to create and then populate tables. Additionally clustrix_import can read any set of SQL queries and execute them, so it can ingest data generated by other tools (such as pgdump for data coming from Postgres), provided the import data is valid SQL.

Input Parsing

The clustrix_import tool can read from either a specified input file or from standard input. When reading from a file a thread will scan through the entire file in order to determine the size of each table in the dump; this information is used to calculate the estimated size of each table and its indexes in order to set the slice count. This estimation cannot be done when reading from standard input. In addition, when reading from a mysqldump file (as indicated by the first line of the input file) the tool assumes the input will always have one query per line, which can result in much faster input parsing (particularly given rows containing text data escaped with backslashes); this optimization is disabled by default when reading from standard input but --scan-for-newlines =false will force this behavior.

One thread of the tool (separate from the scan-ahead thread mentioned above) will read the input file (or stream) and dispatch the queries it reads appropriately. In general:

- DDL statements are processed immediately
- SET statements are captured and distributed to all insert threads
- INSERT statements are placed into a queue for distribution amongst the insert threads.

Optionally, aggregation can be performed to combine single-row INSERT statements into multi-row INSERT statements which can be processed by Xpand much more efficiently.

Several options allow selecting which databases or tables are to be imported, or a byte offset into the dump can be provided.

Parallel Insert Threads

By default eight insert threads per node are started. Insert threads can reconnect on failure and failed operations are automatically requeued to be retried (if an INSERT or other query fails multiple times, e.g. due to syntax error, clustrix_import will exit). When an insert thread reconnects, it regains all necessary state (i.e. SET statements and USE database).

Note that threads connect evenly to all the nodes available in the cluster regardless of which IP (or VIP) is specified. The host specified on the command line is used for an initial connection to the cluster through which we obtain the IPs of all nodes to which we will connect.

In some cases the set of IPs obtained by connecting to the cluster cannot be reached by the client running clustrix_import, such as when clients are accessing the cluster through a NAT'd IP. In such cases, you can specify a comma delimited list of IPs (with the -H or –host flag the client should use instead of determining IPs from the cluster.

Guardrails

clustrix_import includes a number of checks to avoid common mistakes during data import.

Binlog Check

For large data loads it is often undesirable to log all insert data to the binlog. mysqldump often precedes an INSERT with DROP TABLE, and inadvertent logging to a slave could be problematic. Accordingly, clustrix_import will halt by default if binlogging is enabled on the cluster. This can be overridden with --ignore-binlog-checks, or if --no-log-bin is specified the statements are not logged to the binlog (sql_log_bin global is set to false for all threads).

Note that if -d or -D are used to specify the source or target databases, the binlog check will look at whether the specified databases are covered by a binlog. If these flags are not provided the presence of any binlog will prevent clustrix_import from running (unless the check is overridden as described above).

One clustrix_import Instance per Cluster

clustrix_import is designed to fully utilize all cluster resources so running multiple instances of clustrix_import in parallel is not necessary and may be problematic. To prevent this from happening inadvertently a locking mechanism will detect if another clustrix_import already running and will halt a second attempt. If you are certain that you want to run multiple imports this check can be overridden with --allow-parallel-imports.

Specifying --database/-D vs. USE statement

If the dump does not include a USE statement to specify which database is to be loaded the --database/-D option must be given to specify the target database. If the --database/-D option is given and a USE statement is encountered clustrix_import will halt. If you wish to override the USE statement you must edit the dump (e.g. use grep -v "^USE" to remove the USE query).