Migrating from MySQL to InfiniDB™

Calpont Corporation

3011 Internet Blvd, Suite 100

Frisco, TX 75034

www.calpont.com

info@calpont.com

Migrating from MySQL to InfiniDB
Version 1.0
March, 2010

Table of Contents

Introduction

There is little debate that Open Source has now gone mainstream with many corporations, Web sites, government organizations, educational institutions, and others using Open Source software to power key areas of their business. After Linux, the most used Open Source software is database software, with MySQL being the clear leader in that area. Used initially by Web companies, MySQL has now become one of the most prevalent databases is use today, with some surveys showing it coming in third behind Oracle and Microsoft.

While MySQL is used for various Web-style applications and bundled in with certain packaged software products, it has also been deployed for small data marts and/or warehouses that serve business intelligence (BI) systems or other similar read-intensive and reporting applications. However, developers, DBAs, and system architects have sometimes expressed disappointment over the fact that MySQL does not scale well when large data volumes are involved or complex, analytic-styled queries are routinely used against the server. While MySQL is certainly reliable, easy-to-use, and cost effective, it hasn't yet become the database IT professionals normally turn to for help with standard data warehousing or analytic database systems that power critical BI functions.

Fortunately, there is a way to use MySQL in a way where the problems stated above disappear. With InfiniDB™ from Calpont, a person can enjoy the benefits of MySQL, but have at their disposal a powerful database engine that is specially designed to handle lots of data and delivers very fast performance for data warehouses and read-intensive applications. This paper discusses the technical and business benefits of migrating to InfiniDB and also includes strategies on how best to migrate MySQL databases to the InfiniDB server.

Technical Reasons for Migrating to InfiniDB

This section discusses some of the technical reasons why migrating from MySQL to InfiniDB can be worthwhile. In general, while MySQL is a great database for Web sites and various departmental operations, it has not really been used for large scale data warehousing or analytic systems due to the following acknowledged limitations:

- Except for InnoDB and MySQL Cluster (which are not suitable storage engines for data warehousing), no MySQL storage engine can fully address more than 4 CPU's/cores. Hence, MySQL cannot scale-up on modern hardware well
- The MySQL optimizer currently contains various inefficiencies in subquery and join operations (e.g. no hash joins), as well as in the area of unions and other functions
- MySQL does not offer parallel SQL or load capabilities
- MySQL has no way to do massive parallel processing (MPP) across multiple machines
- Once a database gets past 500GB, the issues noted above tend to become fairly noticeable
- MySQL does not support objects such as materialized views/summary tables, bitmap indexes, etc., that other RDBMS systems use to get around their row-based architectures

What does InfiniDB offer MySQL users to get around these and other limitations?

Open Source Model

InfiniDB is offered in two different editions: the Community Edition and the Enterprise Edition. The Community Edition is a free and open source (FOSS) product that contains no artificial limits on the amount of RAM, disk, CPU, or number of users that can be used on a single machine. The Community Edition can be freely downloaded from www.infinidb.org and is supported via free online forums on the infinidb.org website.

The Enterprise Edition is the commercial version of InfiniDB and it comes with three additional components over the Community Edition:

1. Massive parallel processing (MPP) with multiple nodes operations being supported
2. Management, monitoring, and alerting from a single console on InfiniDB's operations, health, and performance is included
3. Formal production support is supplied, along with various other benefits like hot-fix build support and bug escalation privileges

Column-Oriented Architecture

Most legacy relational databases currently being offered today (including MySQL) were and are primarily designed to handle online transactional processing (OLTP) workloads. A transaction typically maps to one or more rows in a relational database, and the vast majority of RDBMS designs are based on a per row paradigm. For transactional-based systems, this architecture is well-suited to handle the input of incoming data and row-by-row removals of information.

However, for applications that are very read intensive where selective information is being requested, the OLTP database design isn't a model that is best. Whereas transactions are row-based, most database queries are column-based. Inserting and deleting transactional data are well served by a row-based system, but selective queries that are only interested in a few columns of a table are handled much better by a column-oriented architecture. On average, a row-based system does 5-10x the physical I/O that a column-based database does to retrieve the same information. Taking into account that physical I/O is typically the slowest part of a query, and that an analytical query typically touches significantly more rows of data that a typical transactional database operation does, the performance gap between row-oriented architectures and column-oriented architecture oftentimes widens as the database grows.

To get around query performance inefficiencies, row-based RDBMS's utilize indexing, horizontal partitioning, materialized views, summary tables, and parallel processing, all of which can help complex queries perform better, but each comes with their own set of drawbacks. For example, while indexing can certainly help queries complete faster in some cases, they also require more storage, impede insert/update/delete and bulk load operations (because the indexes must be maintained as well as the underlying table), and can actually degrade performance when they become heavily fragmented. Further, in business intelligence/analytic environments, the ad-hoc nature of such scenarios makes it nearly impossible to predict which columns will need indexing, so tables end up either being over-indexed (which causes load and maintenance issues) or not properly indexed and so many queries end up running much slower than desired.

Column-oriented databases like InfiniDB are designed especially for analytics and overcome the limitations that exist in traditional RDBMS systems by storing, managing, and querying data based on columns rather than rows. Because only the necessary columns in a query are accessed rather than entire rows, I/O activities as well as overall query response times can be reduced. The end result is the ability to interrogate and return query results against either moderate amounts of information (tens or hundreds of GB's) or large amounts of data (1-n terabytes) in much less time than general RDBMS systems like MySQL can. Commenting on this fact is Gartner database expert Donald Feinberg who says:

> *"If you're bringing back all the columns, a column-store database isn't going to perform any better than a row-store DBMS, but analytic applications are typically looking at all rows and only a few columns. When you put that type of application on a column-store DBMS, it outperforms anything that doesn't take a column-store approach."*[1]

Scale-Up Abilities

Unlike MySQL engines such as MyISAM and others (not including InnoDB and MySQL Cluster), InfiniDB is multi-threaded and therefore capable of utilizing modern hardware that consists of more the 4 CPU's or cores. This means that InfiniDB can scale up (both query and load) on such hardware and deliver performance that is commensurate with a machine's capabilities.

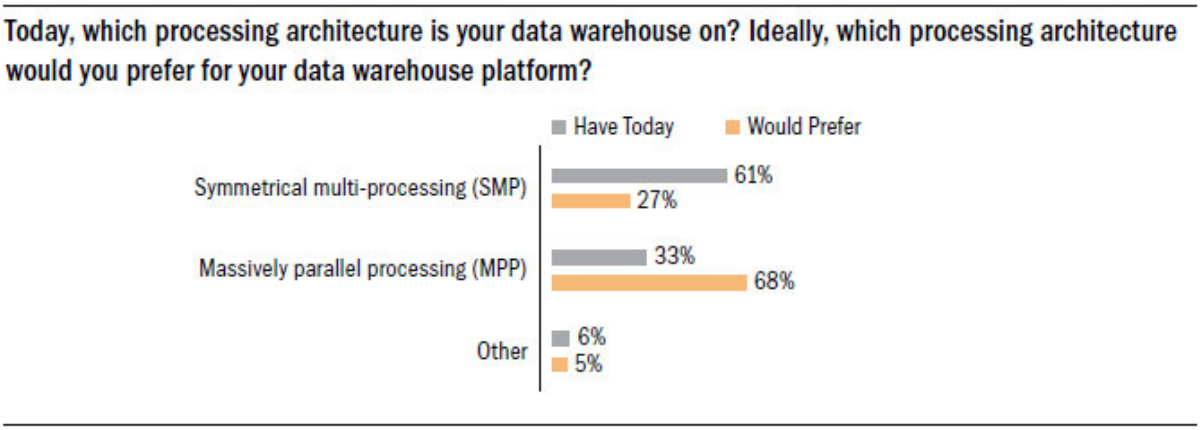
As an example, the below table shows the query response time differences (using the Star Schema Benchmark - SSB) on a machine that is disabled to first only utilize one CPU core and then is enabled so that all 8 cores are available to InfiniDB. Depending on the query, using a scale up or SMP approach with InfiniDB results in an 80-90% response time reduction.

SSB Query (@100 scale)	InfiniDB 1 Core (elapsed time in seconds)	InfiniDB 8 cores (elapsed time in seconds)	Overall Percent Reduction with additional cores
Q2.1	210.21	44.65	79%
Q2.2	151.20	19.70	87%
Q2.3	121.33	15.94	87%
Q3.1	316.79	55.04	83%
Q3.2	164.12	22.14	87%

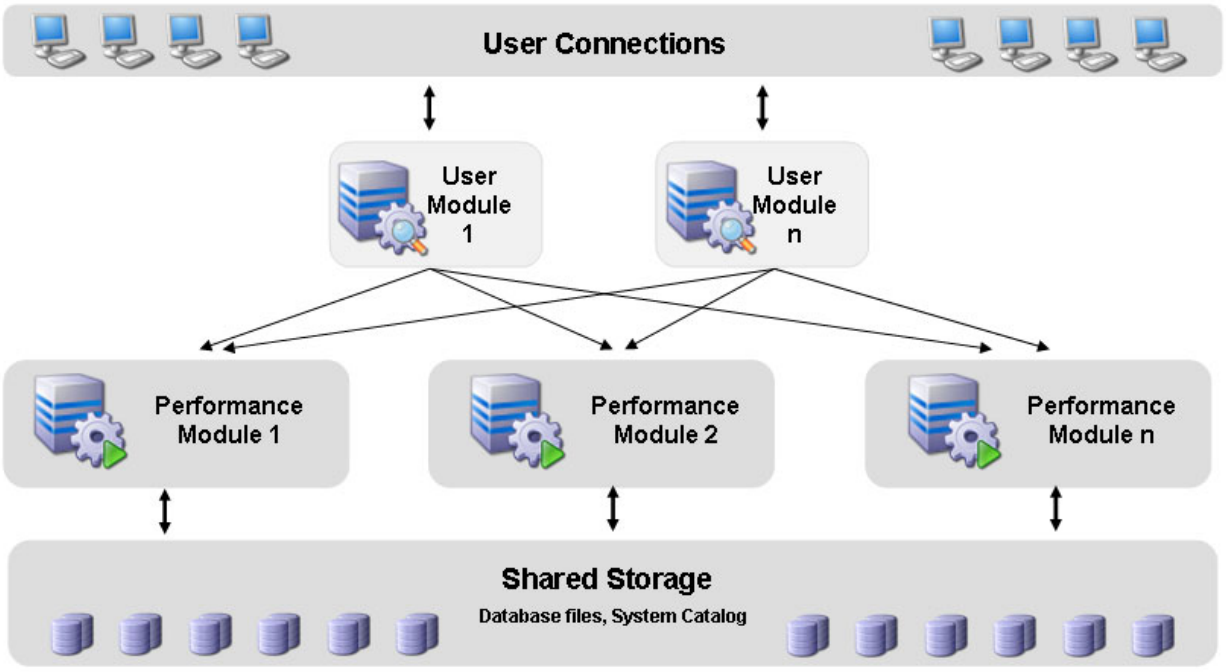
[1]Doug Henschen, "Column-Store Databases and DW Appliances: How to Make the Right Choice", InformationWeek Research and Reports, March 2008.

MPP Ready (Scale-Out)

While InfiniDB's scale-up capabilities are important, today more IT professionals are indicating that scaling out – via massive parallel processing – is the direction in which they are heading. This is underscored by a survey recently performed by TDWI[2]:



InfiniDB is the only MySQL-based solution that offers MPP capabilities for data warehousing and other BI-styled applications. InfiniDB does this through its software-only, modular architecture approach that utilizes allows a system architect to scale out both at the concurrency layer (via InfiniDB's User Modules) and at the I/O layer (with InfiniDB's Performance Modules):



[2]Philip Russom, "Next Generation Data Warehouse Platforms", TDWI, 2009.

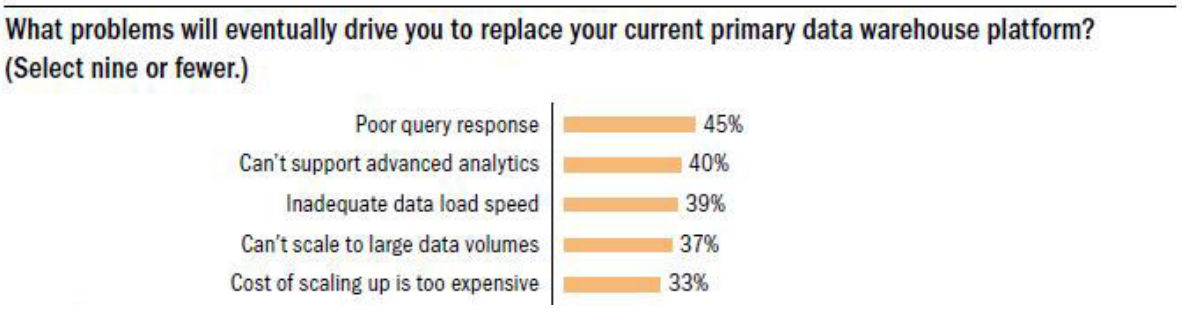
InfiniDB's MPP abilities result in linear performance gains when additional nodes are added to a system. The table below shows the Star Schema Benchmark (SSB) at a larger scale and shows the response time reductions that are achieved when various, MPP, multi-node configurations of InfiniDB are used:

SSB Query @1000	1 Node (elapsed time in seconds)	2 Node (elapsed time in seconds)	4 Node (elapsed time in seconds)	8 Node (elapsed time in seconds)	Overall Percent Reduction from 1 – 8 Nodes
Q2.1	531.34	261.35	129.90	68.21	87%
Q2.2	430.25	214.87	106.37	56.41	87%
Q2.3	386.66	192.03	96.03	51.36	87%
Q3.1	848.79	425.25	316.50	134.21	84%
Q3.2	597.97	297.46	148.49	77.74	87%

Again, response time reductions of close to 90% are experienced when using InfiniDB in MPP fashion.

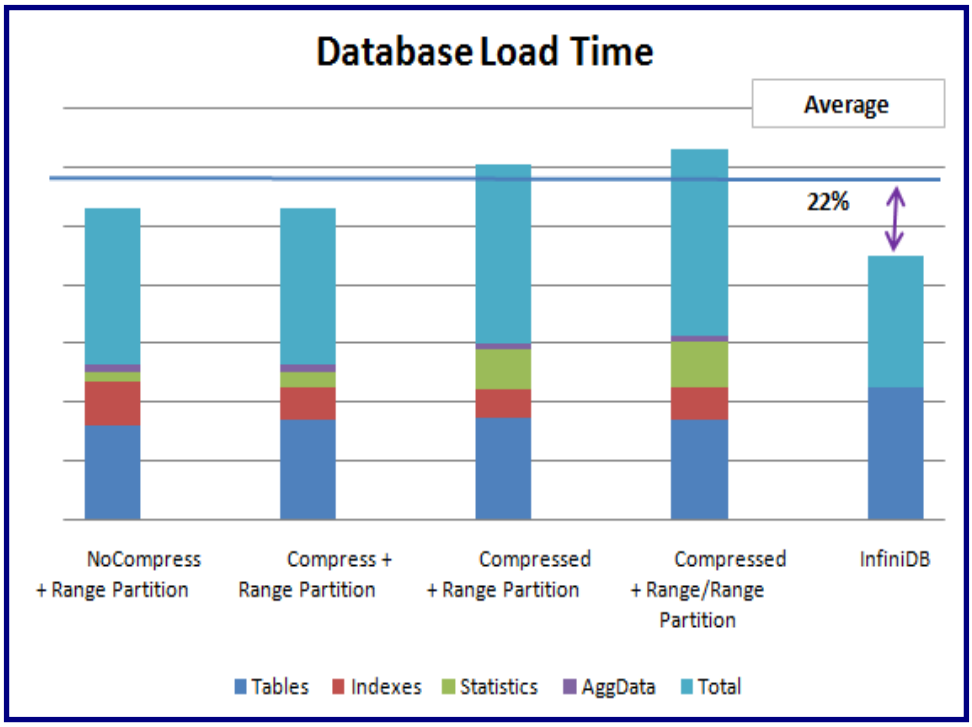
More Proof of Performance

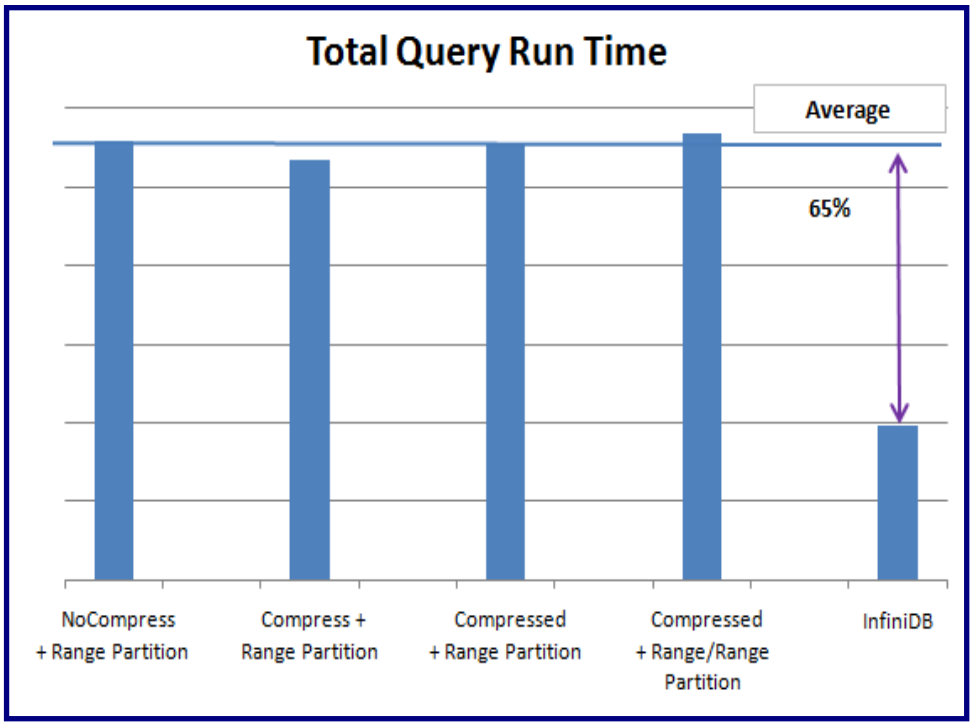
The previously mentioned TDWI study[3] shows that the number one reason IT decision makers move to a new data warehouse database is because their current one runs too slowly:



Because InfiniDB takes a column-oriented approach, and scales both up and out, it can offer much better performance than standard MySQL as well as many other row-based databases and even other column-based RDBMS's. For example, Bert Scalzo, a data warehouse expert, recently benchmarked[4] InfiniDB against one of the leading row-based databases and found that InfiniDB loaded and queried data much faster than its row-based rival:

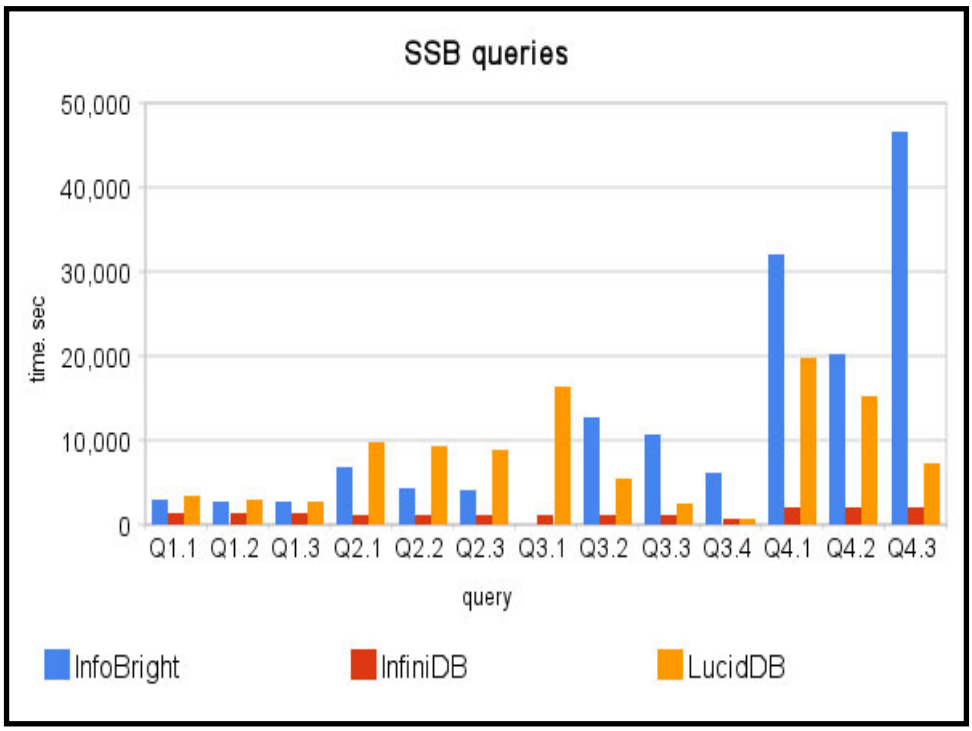
[3]Russom.
[4]Tests run on standalone machine: 16 CPU, 16GB RAM, CentOS 5.4 with 2TB of raw data





In a different benchmark, Percona (a leading MySQL consulting/performance company) tested InfiniDB[5] against a number of other column databases and found that InfiniDB was faster in most queries:

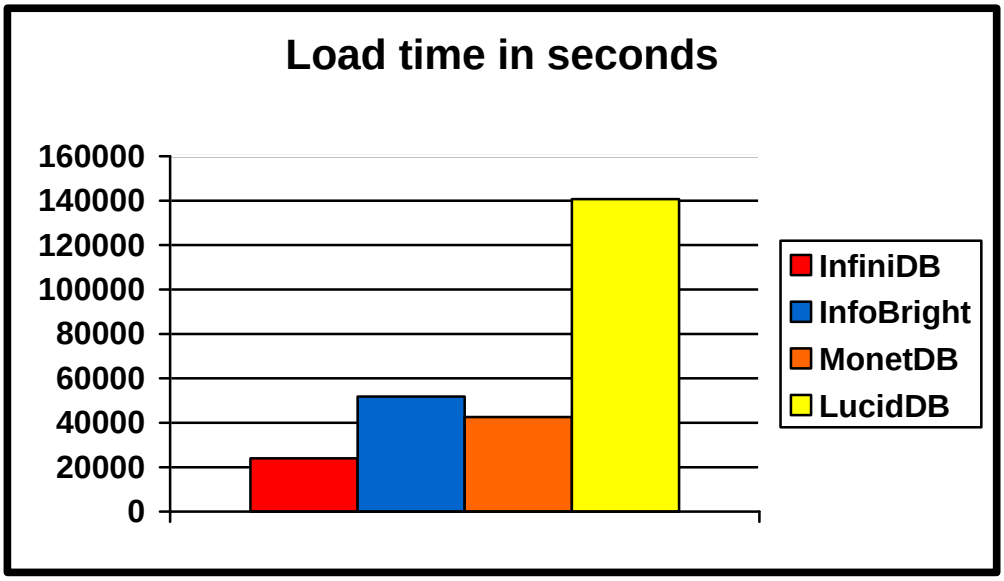
[5] http://www.mysqlperformanceblog.com/2010/01/07/star-schema-bechmark-infobright-infinidb-and-luciddb/



So whether it's against popular row-based databases or other column DB's, evidence shows that InfiniDB has an edge when it comes to delivering fast performance.

Faster Data Loading

As shown above, InfiniDB loaded data much faster than its row-based competitor. Percona's test also found that InfiniDB loaded data more quickly than other open source/column-based databases. When loading a database with 610GB of raw data, Percona recorded the following times:



InfiniDB supplies a high-speed loader that loads data much faster than MySQL with its Load Data Infile utility, which means data is available for BI and query work much sooner with InfiniDB than MySQL.

Solid Feature Set

In addition to the benefits described above, InfiniDB offers a number of features that make it a solid performer for data warehouses, marts, and other read-intensive applications:

- **High concurrency support**: InfiniDB's limits as far as concurrency is concerned are only restricted by the server machine's capabilities; no theoretical concurrency limit exists.
- **DML support:** In addition to supporting the high-speed bulk loading of data, InfiniDB supports full DML (insert, update, delete) operations as well.
- **Transactional support:** ACID-compliant transactional support is provided in the InfiniDB database. Transactions can easily be committed or rolled back, and deadlock detection support is also provided to handle conflict resolution.
- **Crash recovery:** InfiniDB provides full crash recovery capabilities. In the event of a system crash, InfiniDB automatically maintains data integrity and upon system restart, InfiniDB supports roll forward and rollback operations to return the database to a consistent state.
- **MVCC design:** InfiniDB supports multi-version concurrency control (MVCC) or "snapshot read" so query operations are never blocked on a table; a query will always see the data as it existed at the time the query was issued.
- **Logical Data compression**: InfiniDB uses transparent logical data compression to store data that benefits from being compressed. The end result is storage savings and a smaller data footprint. InfiniDB also reduces storage requirements by not relying on indexing or other objects such as materialized views.
- **Alter table support:** Columns may also be added and dropped from a table with the `ALTER TABLE` command, with specialized support for online column additions.

- **Performance diagnostics:** To help tune performance, InfiniDB supplies monitoring and diagnostic utilities that help a user monitor their database and troubleshoot poorly running SQL.
- **MySQL front end:** InfiniDB utilizes MySQL for its basic user interface. This allows anyone familiar with MySQL to become immediately productive with InfiniDB. For those not acquainted with MySQL, the learning curve is minimal as MySQL supports almost all ANSI standard SQL operations. Moreover, there are many freely supplied GUI tools from MySQL as well as other vendors that may be used to develop against and administer an InfiniDB database.

Certain features that are part of InfiniDB Enterprise's MPP design include:

- **Distributed shared-nothing data cache**: In a multiple-node InfiniDB configuration, data is distributed among the various nodes and their data caches. No node shares data with the other, however all are accessed in the InfiniDB MPP architecture when data is read to satisfy queries. In essence then, InfiniDB creates one large logical data cache that is accessed in a distributed fashion in parallel by all participating nodes. This allows InfiniDB to literally cache large databases when enough nodes are present with generous amounts of memory.
- **Automatic failover**: InfiniDB's architecture contains built-in automatic failover for the software modules responsible for performing the actual database I/O operations.
- **Automatic concurrency scale-out**: Nodes can be added to a configuration that allows concurrency to be automatically scaled across 1-n machines.
- **Automated software patch management:** When patches or upgrades need to be applied to more than one InfiniDB server, an automatic process takes the software from the first node where an upgrade is applied and automatically upgrades all other participating nodes.

No Manual Indexing or Partitioning Required

Getting exceptional data load and query performance from InfiniDB does not require seasoned database experts to design and tune the system. InfiniDB takes the database design and administration headaches away from business intelligence professionals through its auto-management architecture framework.

Being column-oriented, InfiniDB requires no indexing, which means database administrators and developers don't have to worry about complex indexing schemes and the storage overhead such structures impose. Being column-oriented, InfiniDB obviates the need for traditional indexing because, in essence, the column is the index. Rather than use traditional indexing, InfiniDB utilizes an automatically maintained structure called the Extent Map that keeps track of where the data is and how it should best be retrieved.

Next, InfiniDB removes the need for manual data partitioning and storage placement. InfiniDB uses a form of logical horizontal range partitioning that does not require special storage placement or schema design. Using both vertical (column-orientation) and logical horizontal range partitioning allows InfiniDB to reduce I/O in both directions (column and row). Both vertical and horizontal partitioning are automatically handled by the InfiniDB database and require no user intervention.

Easy Management and Monitoring

InfiniDB Enterprise has a management, monitoring, and alerting system built into it that a DBA or sysadmin can use to maintain an InfiniDB configuration. Full management and monitoring of both the database and operating system is included.

Further, system alerts can be configured so that an administrator can be notified of issues that may threaten either the performance or availability of the system. These alerts can be delivered in various ways, including SNMP traps that can be sent to trouble ticketing systems like HP OpenView or Tivioli.

Integration with MySQL

InfiniDB bundles MySQL with its installation and uses MySQL in its User Module for basic SQL parsing, security, and a few other things. This means that, on the surface, InfiniDB looks and feels just like MySQL and has its same great ease-of use, but is a lot more powerful underneath the covers when it comes to managing lots of data and delivering fast response times against that data.

All GUI and command line management and development tools, BI suites, application IDE's, and connectors/drivers that work with MySQL can be used with InfiniDB. No other specialized software needs to be purchased to work with InfiniDB.

Also note that other MySQL storage engines can be used in the same database as InfiniDB tables.

Business Reasons for Migrating to InfiniDB

Having examined the technical reasons of why it makes sense to move BI and data warehouse databases to InfiniDB, the next area to investigate is the business aspect of making such a move. What are some considerations that come into play?

Proven Platform

The first concern of many CTO's and CIO's is the 'provability' of a software vendor – in other words, does the software vendor being considered have a historical track record that is compelling?

Because InfiniDB uses MySQL as its front-end, this concern is a non-issue. MySQL has over 12 million installations and powers some of the most demanding applications in use today. When combined with InfiniDB, MySQL's capabilities are only strengthened and enlarged.

Dual Licensing Model

As previously stated, Calpont makes the InfiniDB Community Edition available under the open source (GPL v2) license with no restrictions whatsoever being imposed in terms of the number of CPU's/cores, RAM, size of database, or number of concurrent users.

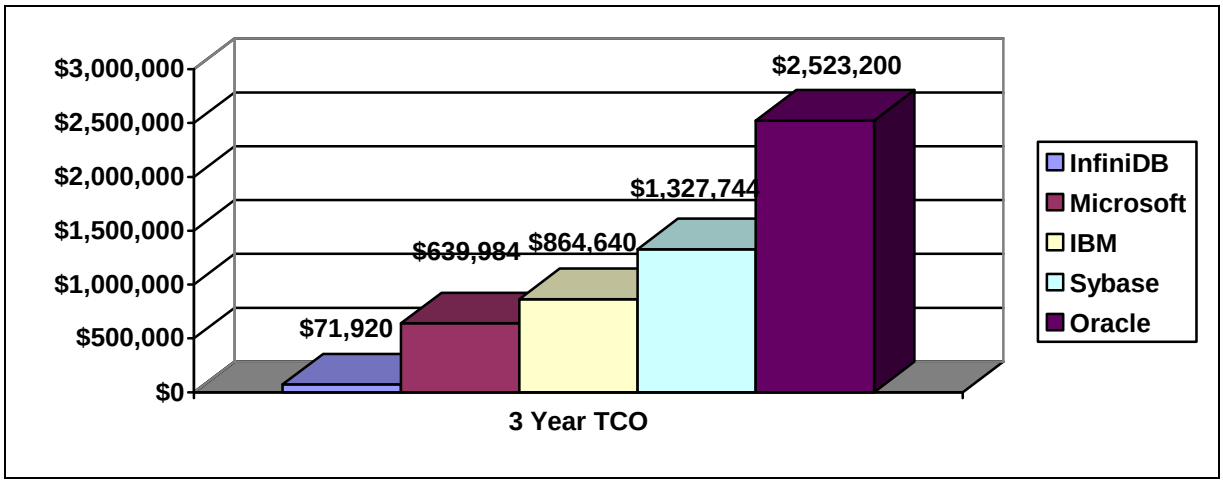
InfiniDB Enterprise, however, is offered under a commercial license. Because Calpont provides a dual-licensing model, there is no need for a commercial vendor to worry about their application falling under open source guidelines. Those wishing to embed or package InfiniDB with their application can use the commercial, Enterprise license that legally functions as any other commercial piece of software does.

Cost-Effective Solution

As opposed to proprietary database vendors, and even the other few open source business intelligence databases on the market, InfiniDB offers a highly attractive licensing and pricing model just like MySQL that is designed to make powerful business intelligence database software available to everyone. Calpont does not price InfiniDB by amount of space used (e.g. Vertica, Infobright) or by number of CPU's/cores used (e.g. Oracle, Microsoft), but instead offers simple per-server pricing model with standard yearly per-server maintenance costs.

As an example, below is a three year total cost of ownership comparison with the major legacy database vendors for four servers that have a total of 4 CPU's and 2 cores per CPU:

	InfiniDB Enterprise	**Microsoft SQL Server Enterprise**	**IBM DB2 Enterprise**	**Sybase ASE**	**Oracle Enterprise**
Product	Enterprise	Enterprise	Enterprise	Enterprise	Enterprise
Model	Per server	Per CPU	Per core	Per core	Per core
License per unit cost	$11,995	$24,999	$38,600	$24,995	$47,500
Support per unit cost	$1,995	$5,000	$7,720	$5,499	$10,450
Totals for 4 servers:					
Total license costs	$47,980	$399,984	$617,600	$799,840	$1,520,000
Total support costs	$23,940	$240,000	$247,040	$527,904	$1,003,200
3-Year TCO	$71,920	$639,984	$864,640	$1,327,744	$2,523,200
InfiniDB Savings		**89%**	**92%**	**95%**	**97%**



Editions of the above vendor's business intelligence databases (e.g. Oracle Exadata) can cost even more. For example, Microsoft's pricing on their parallel data warehouse (project Madison) is $57,498 per CPU (without factoring in the necessary SQL Server database or reporting software), which would increase the total licensing cost for Microsoft in the above scenario to $919,968 and the total support costs to $607,179, for a total three year TCO of $1,527,147. Compared to Microsoft, InfiniDB would offer a TCO savings of 95%. Including the other Microsoft components makes the InfiniDB savings even higher.

Newer business intelligence databases, such as Vertica, Greenplum, and Paraccel price by the amount of space used (priced by the TB, which is the smallest unit of measure) and not by a machine's hardware specifications. Average per TB costs are $70,000-100,000 with these vendors. Such a pricing model can quickly become expensive for businesses with rapidly expanding business intelligence databases.

InfiniDB, however, has a flat per-server pricing model that does not impose additional costs simply because a database grows in size. Should additional data volumes result in queries exceeding internal performance guidelines, additional processors can be added to standalone implementations to increase performance (with no additional software licensing costs being incurred), and/or one or more additional nodes can be added to an MPP configuration to improve query speed (at a flat per-server cost).

Little to No Learning Curve

Because it uses MySQL as its front end, those migrating from MySQL to InfiniDB will be immediately productive. There is little to no learning curve experienced, so there are no retraining costs that will have to be incurred.

For those coming from non-MySQL databases, because ANSI-standard SQL is employed by InfiniDB, nearly all skills from other databases will transfer over intact.

Migration Strategies

This section contains recommendations and suggestions for how to migrate both MySQL schema and data to InfiniDB, with a variety of different approaches being presented.

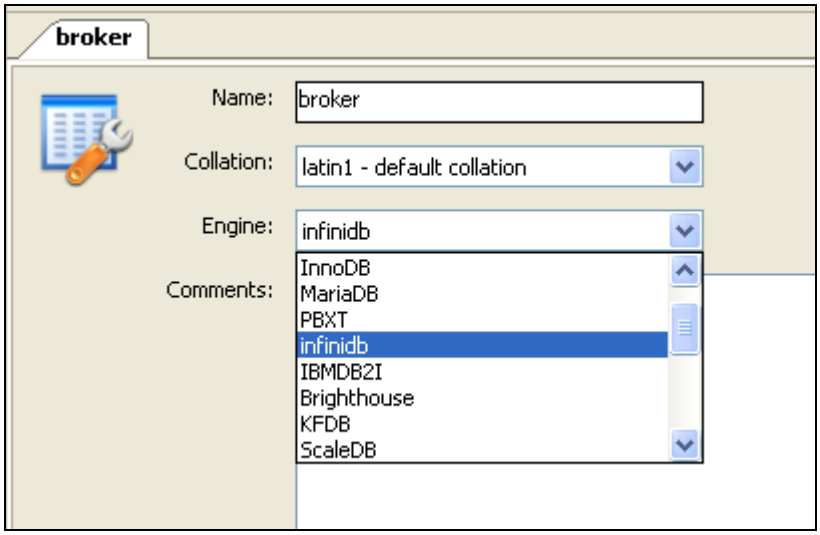
The typical set of migration steps consist of:

1. Documenting MySQL sources
2. Creating InfiniDB objects
3. Transferring MySQL data to InfiniDB

Documenting MySQL Sources

There are a number of different ways to document the MySQL databases and objects a DBA wants to move to InfiniDB. For example, doing simple `SHOW CREATE TABLE` statements in the mysql command line client program can capture a database's DDL code.

Using the MySQL Workbench/GUI tools with an existing MySQL database is another way. With MySQL Workbench, a user can reverse engineer an existing database into a model and then change the engine to be InfiniDB instead of the existing engine type:



Once done, the developer can forward engineer the model and create an InfiniDB database. Note there are a number of MySQL third party tools on the market (both free and pay-for) that can edit and manage MySQL objects as well.

Creating InfiniDB Objects

Once a DBA or developer has obtained the MySQL source objects and their DDL, they can modify the code to create InfiniDB objects. The most obvious change that needs to be performed is to change the `engine=` clause to use InfiniDB.

However, there are other changes that may have to be made as well. Some of them may include:

- The removal of all index `CREATE` statements (because InfiniDB does not use indexes)
- The removal of all primary and foreign key constraints
- The removal of any NULL column constraints
- The removal of any other column constraints
- The alteration of datatypes to include only those supported by InfiniDB (see InfiniDB's online documentation – the SQL Syntax Guide – for a current list of supported datatypes).

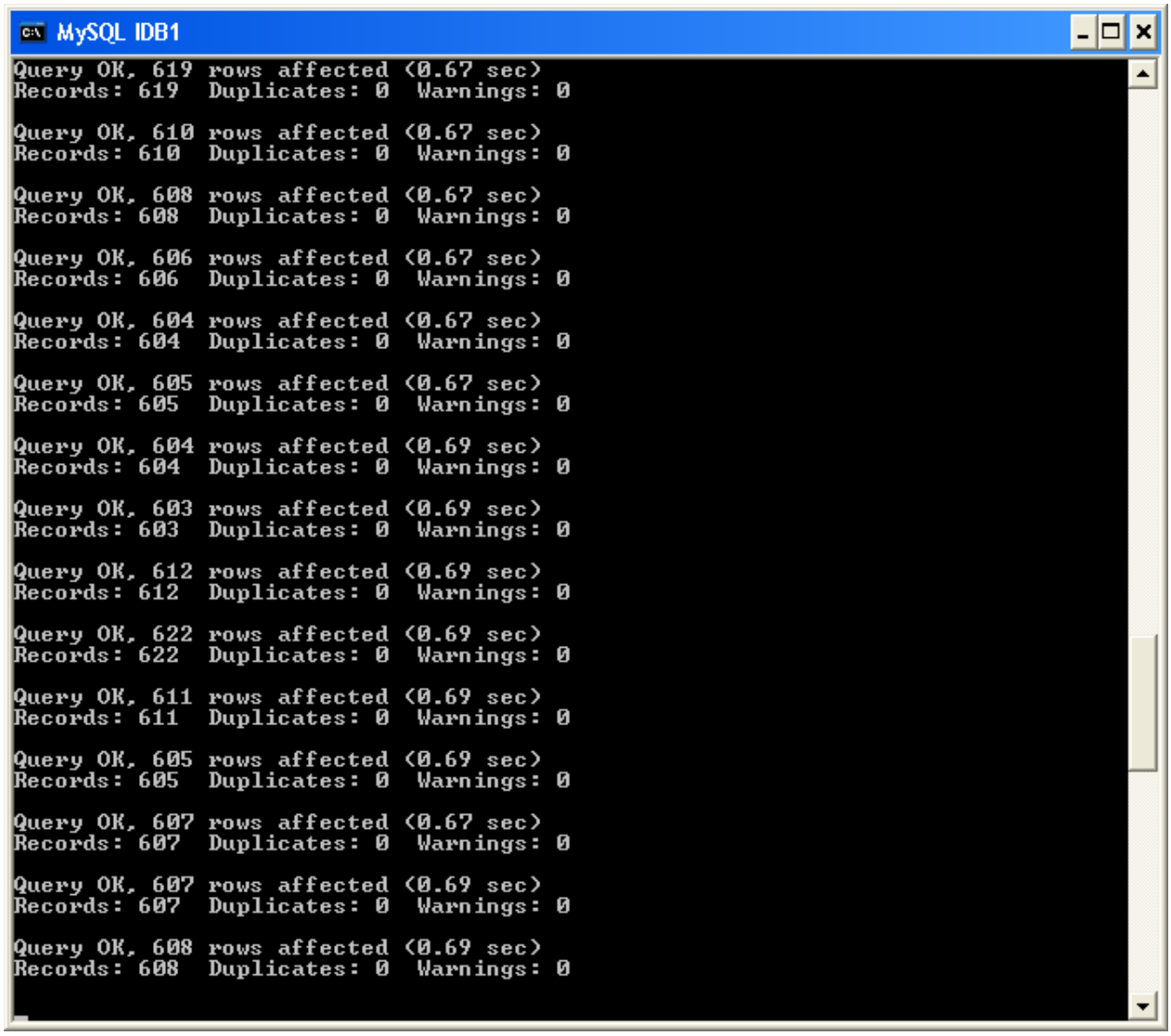
Migrating MySQL Data to InfiniDB

There are a number of different methods that may be used to move data from MySQL to InfiniDB. Some of the more common approaches follow.

Use mysqldump

The mysqldump utility is primarily used to backup and restore MySQL databases, but it can also be used to move data from MySQL to InfiniDB. The basic steps to use mysqldump with InfiniDB are:

- Dump tables from the source MySQL database with the `--no-create-info` option that won't issue create table statements
- Create the target InfiniDB database and tables
- Connect to the InfiniDB server with the idbmysql command line tool and set the correct database using the `use <database name>` command
- Issue the command `SOURCE <mysql dump file location and name >` to import data
- Data should be imported into InfiniDB a manner as shown in the example below:



Use SELECT INTO OUTFILE

The `SELECT INTO OUTFILE` command writes the output from a SQL query to a file on disk. This allows a developer or a DBA to create a flat file that can then be used to load data into InfiniDB with InfiniDB's high-speed loader called cpimport.

As an example, if a developer wanted to load the data from a MySQL table called 'broker' into InfiniDB, they would first log into MySQL via the mysql command line client tool and issue a `SELECT INTO OUTFILE` command such as the following to create a file from the table's data:

```
mysql> select * into outfile 'c:/dev/broker.dat'
    -> fields terminated by '|'
    -> from broker;
Query OK, 23 rows affected (0.00 sec)
```


In the above example, the pipe character ('|') is used as the field delimiter. Once the file has been created, the developer would transfer the file to the InfiniDB server and load the data. This is done by first creating a load job file with InfiniDB's 'colxml' utility that sets the file up for loading into a column-oriented database:

```
[root@srvprodtest1 bin]# ./colxml gim2 -t broker -j100
Running colxml with the following parameters:
2010-02-26 14:06:21 (28641) INFO :
        Schema: gim2
        Tables: broker
        Load Files:
        -b      0
        -c      1048576
        -d      |
        -e      10
        -f      CSV
        -j      100
        -m      50
        -n
        -p      /usr/local/Calpont/data/bulk/job/
        -r      5
        -s
        -u
        -w      10485760
        -x      tbl

File completed for tables:
        gim2.broker

Normal exit.
```

In the above example, a job file numbered 100 has been created. The next step is to invoke cpimport and load the data into the InfiniDB broker table:

```
[root@srvprodtest1 bin]# ./cpimport -j100

Bulkload root directory : /usr/local/Calpont/data/bulk
job description file : Job_100.xml
2010-02-26 14:07:08 (28796) INFO : successfully load job file
/usr/local/Calpont/data/bulk/job/Job_100.xml
2010-02-26 14:07:08 (28796) INFO : PreProcessing check starts
2010-02-26 14:07:08 (28796) INFO : PreProcessing check completed
2010-02-26 14:07:08 (28796) INFO : preProcess completed, total run time : 0 seconds
2010-02-26 14:07:08 (28796) INFO : No of Read Threads Spawned = 1
2010-02-26 14:07:08 (28796) INFO : No of Parse Threads Spawned = 3
2010-02-26 14:07:08 (28796) INFO : For table gim2.broker: 23 rows processed and 23
rows inserted.
2010-02-26 14:07:08 (28796) INFO : Bulk load completed, total run time : 0 seconds
```

Note that the MySQL `LOAD DATA INFILE` command may be used with InfiniDB, however it runs much slower than the high-speed cpimport utility.

Use the MySQL CSV Engine

Another way of moving data from an entire table and importing it into InfiniDB is by use of MySQL's CSV storage engine (available in MySQL 5.1 and above). The Comma-Separated-Value storage engine stores data for a table in a flat file where the columns are separated in the file by commas. A user can take an existing MySQL table on a non-InfiniDB installation, alter the table to use the CSV engine instead of its current engine, copy the table file to the InfiniDB server, and then alter the table back to its original engine.

Once the user has the table file in comma separated format on the InfiniDB server, they can then move it to the InfiniDB import directory and use the `cpimport` utility to load it into InfiniDB. As an example, imagine a developer has the following table in MySQL:

```
mysql> show create table client_dup\G
*************************** 1. row ***************************
       Table: client_dup
Create Table: CREATE TABLE `client_dup` (
  `client_transaction_id` int(11) NOT NULL DEFAULT '0',
  `client_id` int(11) NOT NULL DEFAULT '0',
  `investment_id` int(11) NOT NULL DEFAULT '0',
  `action_type` varchar(10) NOT NULL,
  `price` decimal(12,2) NOT NULL DEFAULT '0.00',
  `number_of_units` int(11) NOT NULL DEFAULT '0',
  `transaction_status` varchar(10) NOT NULL,
  `transaction_sub_timestamp` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `transaction_comp_timestamp` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `description` varchar(200) NOT NULL,
  `broker_id` bigint(10) NOT NULL,
  `broker_commission` decimal(10,2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

They could alter the MySQL engine to be CSV:

```
mysql> alter table client_dup engine=csv;
Query OK, 112050 rows affected (5.09 sec)
Records: 112050  Duplicates: 0  Warnings: 0
mysql>
```

Once the engine transfer is complete, they would then transfer the new CSV file to the InfiniDB server and use InfiniDB's colxml and cpimport utilities to load the data into the pre-created InfiniDB table:

```
[root@srvprodtest1 bin]# ./colxml gim2 -t client_dup -d , -j101
Running colxml with the following parameters:
2010-02-26 14:43:27 (29208) INFO :
        Schema: gim2
        Tables: client_dup
        Load Files:
        -b      0
        -c      1048576
        -d      ,
```



```
        -e      10
        -f      CSV
        -j      101
        -m      50
        -n
        -p      /usr/local/Calpont/data/bulk/job/
        -r      5
        -s
        -u
        -w      10485760
        -x      tbl

File completed for tables:
        gim2.client_dup

Normal exit.
[root@srvprodtest1 bin]# ./cpimport -j101

Bulkload root directory : /usr/local/Calpont/data/bulk
job description file : Job_101.xml
2010-02-26 14:43:51 (29371) INFO : successfully load job file
/usr/local/Calpont/data/bulk/job/Job_101.xml
2010-02-26 14:43:51 (29371) INFO : PreProcessing check starts
2010-02-26 14:43:51 (29371) INFO : PreProcessing check completed
2010-02-26 14:43:51 (29371) INFO : preProcess completed, total run time : 0 seconds
2010-02-26 14:43:51 (29371) INFO : No of Read Threads Spawned = 1
2010-02-26 14:43:51 (29371) INFO : No of Parse Threads Spawned = 3
2010-02-26 14:43:52 (29371) INFO : For table gim2.client_dup: 112050 rows processed
and 112050 rows inserted.
2010-02-26 14:43:52 (29371) INFO : Bulk load completed, total run time : 1 seconds
```

Again, the MySQL `LOAD DATA INFILE` command may be used with InfiniDB, however it runs much slower than the high-speed cpimport utility.

Use Third-Party ETL Tools

Another strategy is to make use of free open source ETL (extract-transform-load) tools to move data from one MySQL server to an InfiniDB server. Tools are available from Pentaho, Jaspersoft, Talend, and others. Proprietary tools such as Informatica will also work with InfiniDB.

Note that standard MySQL connectors such as ODBC and JDBC can be used with the above tools to connect to InfiniDB; these connectors are available on the MySQL web site at: http://www.mysql.com/downloads/connector/.

Conclusion

From both a technical and business perspective, migrating data warehouses, data marts, and other analytic databases from MySQL to InfiniDB makes sense. By following the guidelines and steps in this paper, a developer, DBA, or system architect can feel confident that they will succeed in their migration from MySQL to InfiniDB, and experience much greater performance than they've ever seen on MySQL.

Moving to InfiniDB not only guarantees better performance and scale today, but also provides a future-proofed architecture for tomorrow as well.

About InfiniDB and Calpont

Calpont Corporation is a provider of scalable, high-performant, and simple to use analytic database and data warehouse solutions. Calpont offers both Free and Open Source (FOSS) and Commercial versions of the InfiniDB™ database product that are designed to enable pervasive business intelligence and data warehousing throughout all organizations.

For more information on the InfiniDB Community Edition, including free downloads of the database, visit www.infinidb.org. For more information on the Enterprise Edition of InfiniDB, which includes scale-out/massive parallel processing (MPP) capabilities, visit www.calpont.com.