Frequently Asked Questions

What is ClustrixDB?

ClustrixDB is a shared-nothing clustered scalable database based on commodity hardware and parallel software. Parallelism throughout the system integrates the various nodes of the cluster into one very large (huge) database, from both programming and management perspectives. There are no bottlenecks and no single points of failure. All processors are enlisted in support of query processing. Queries are parallelized and distributed across the cluster to the relevant data. New nodes are automatically recognized and incorporated into the cluster. Workloads and data are automatically balanced across all nodes in the cluster. Cluster-wide SQL relational calculus and ACID properties eliminate multi-node complexity from the development and management of multi-tiered applications. The complexity commonly required to scale existing db models to handle large volumes of data is eliminated. And as your database grows, just add nodes.

What are the main features of ClustrixDB?

- ACID transactions and relational calculus across all nodes of the cluster, as opposed to doing without, or developing this capability in applications.
- JOINs across all nodes of the cluster, as opposed to writing join logic in applications.
- Automatic balancing of data and query workload across all nodes of the cluster, as opposed to balancing data manually, and adapting applications accordingly.
- Easy cluster-oriented management, as opposed to managing individual nodes.
- Online schema changes across all nodes while continuing to process read & write transactions, as opposed to taking some nodes offline, making the schema changes, then bringing them online again.
- Horizontal federation and master-slave replication are not required for scale. The MySQL replication protocol/system is supported for testing and database loading purposes.
- Fault tolerant - able to continue operating through all single, and some multiple, node or disk failures.
- Add nodes just by plugging them in - no programming or administration required.

Does ClustrixDB use any MySQL code?

No MySQL code is used. ClustrixDB is entirely original, based on decades of experience in the development of scalable parallel file systems, very large time-series databases, and some of the world's fastest super-computers.

Is ClustrixDB available as open source?

No, ClustrixDB is available as licensed, downloadable software.

On which platforms is ClustrixDB supported?

ClustrixDB is supported on RHEL or CentOS 7.4.

What makes ClustrixDB scalable?

There are several things that affect scalability and performance:

- Shared-nothing architecture, which eliminates potential bottlenecks. Contrast this with shared-disk / shared-cache architectures that bottleneck, don't scale, and are difficult to manage.
- Parallelization of queries, which are distributed to the node(s) with the relevant data. Results are created as close to the data as possible, then routed back to the requesting node for consolidation and returned to the client.

This is very different from other systems, which routinely move large amounts of data to the node that is processing the query, then eliminate all the data that doesn't fit the query (typically lots of data). By only moving qualified data across the network to the requesting node, ClustrixDB significantly reduces the network traffic bottleneck. In addition, more processors participate in the data selection process, By selecting data on multiple nodes in parallel, the system produces results more quickly than if all data was selected by a single node, which first has to collect all the required data from the other nodes in the system.

- Since each node focuses on a particular partition and sends work items to other nodes rather than requesting raw data from other nodes, each node's cache contains more of that node's data, and less redundant data from other nodes. This means cache hit rates will be much higher, significantly reducing the need for slow disk accesses.

How does a client know with which node of the cluster to connect?

It doesn't matter. Clients can connect to any node in the cluster. The ClustrixDB parallel database software will route the queries to the appropriate nodes - the ones that have the relevant data. Clustrix recommends using an external load balancer.

How does ClustrixDB compare with the master-slave replication approach to scalability?

Replication only scales reads. In a master-slave configuration, all writes are done to the master, then replicated to the various slaves. This causes two problems:

- It takes time to replicate the writes to the slaves. If a slave is read before the write is replicated, then the data that's read will be obsolete.
- Eventually, the system spends all of its time replicating writes, and no cycles are left for reads.

How does ClustrixDB compare to application-level horizontal federation (a.k.a. sharding)?

Essentially, ClustrixDB is doing horizontal federation. The key is making the federation invisible to applications and to administrators. In addition, ClustrixDB provides:

- Full ACID (Atomicity, Consistency, Isolation & Durability) properties across partitions.
- Full relational calculus (i.e. left, inner & outer joins, etc.) across partitions.
- Automatic management of the cluster - little administrator intervention is required, other than specifying the number of data replicas, and the priorities for various system functions, such as data replication.

By making the federation invisible to applications, ClustrixDB eliminates the need for custom programming and administration for partitioning. This increases the customer's ability to query and update transactions across partitions, ultimately leading to greater functionality at lower cost.

What are data replicas?

All data in ClustrixDB is replicated on a per-table or per-index basis. Customers may prefer to maintain more replicas of base representations (data tables), and fewer replicas of indexes, since they are reconstructable.

How does ClustrixDB optimize joins?

The query planner is cluster-aware, and it knows which nodes of the cluster contain which indexed rows. Here's how it works:

- Indexes are pointers to rows. A hashing function is used to store indexes, so indexes created on multiple tables will hash to the same node when the values that are indexed (or hashed) are the same. So the index for certain rows of table A will be stored on the same node as the index for rows of table B which have the same index values.

- If an index is a primary key, then the rows are stored with the index. If the index is not a primary key, then the rows may be on a different node than the index.

- This means that the rows for a table with a primary key are located on the same node as the index for rows in another table with its own index. The second index has pointers to the actual rows.

- The effect of this is to reduce or eliminate cross-node traffic. For example, say an application wants to join table A's primary key Ap with table B's index Bi. The hashing function has already placed Ap rows and Bi indexes on the same node. The join operation will be dispatched to that node, and the only rows that need to be moved between nodes are the rows in B that meet the join criteria, as indicated by Bi. There's little of the cross-node data movement that's required for joins on other systems.

- If the join is on two primary keys (Ap and Bp), then A's rows and B's rows will already have been hashed to the same nodes, completely eliminating the need for movement of raw data between nodes.

Note: if the join is on columns that have no indexes, then table scans are required, but the scans can be done in parallel on multiple nodes, so the operation, while not optimal, is still accelerated.

What steps are required to start a ClustrixDB database?

See ClustrixDB Installation Guide Bare OS Instructions.

What steps are required to add more nodes to an existing ClustrixDB database?

The short answer is: just add nodes. Refer to these instructions for guidance in Expanding Your Cluster's Capacity - Flex Up.

What happens to the system if a component fails?

The system is designed to continue operating through inevitable component failures, as follows:

- If a disk fails, then new replicas will be created in accordance with priorities defined by the administrator; future transactions will run against the other replicas.
- If a node fails, then new replicas of all data on that node will be created in accordance with priorities defined by the administrator; future transactions will run against the other replicas. Commonly accepted best practices for transaction retry are recommended.

What levels of redundancy are provided?

The node is the fundamental redundant unit. Multiple nodes can fail without a system outage. In addition, all data paths and all data are redundant. Administrators can specify the desired level of redundancy (number of data replicas) and can specify priorities for the re-creation of additional replicas when storage or nodes fail.

Is ClustrixDB a new storage engine for MySQL?

No, it's a complete database, built from the ground up for high-performance, clustered OLTP. It is wire-compatible with MySQL, but is implemented without any MySQL code.

Does the product support online backup operations?

Yes. For complete details, please see ClustrixDB Fast Backup and Restore. ClustrixDB also supports MySQL operations such as mysqldump.