



Postgres-XC

A PostgreSQL Clustering Solution

What started with a simple relational database system, is expanding its horizons by developing new technology that satiates the ever-increasing need for more data storage, greater transaction throughput and higher availability. Using a cluster to solve these scalability problems is a present trend. This article talks about Postgres-XC, a clustering solution based on the popular PostgreSQL RDBMS.

A cluster is a collection of commodity components that provide scalability and availability at a low cost to the consumer. A database cluster is a collection of database servers that store and process data using commodity hardware, satisfying the need for more data storage, higher throughput and providing high availability. Postgres-XC is such a database cluster system; it is based on the world's most advanced open source database, PostgreSQL, and follows the same open source model.

The Postgres-XC project began in 2009, through a collaboration between NTT and EnterpriseDB. The goal was to build an open source clustering solution based on PostgreSQL with 100 per cent compatible client APIs. Having PostgreSQL-compatible APIs allows existing PostgreSQL applications to use Postgres-XC with little (or no) change. The licensing terms of this project are the same as that of PostgreSQL.

Postgres-XC architecture

Postgres-XC is a write-scalable, synchronous multi-master, transparent PostgreSQL clustering solution based on shared-

nothing architecture. It is a collection of tightly coupled database components, which can be installed on one or more physical or virtual machines. The components do not share any resources such as disk, cache or memory.

- *Write-scalability* means that Postgres-XC can be configured with as many database servers as needed; Postgres-XC is able to handle more writes than a single PostgreSQL server.
- *Multi-master* implies that clients can connect to multiple database servers, and that each database server provides a single, consistent, cluster-wide view of the database.
- *Synchronous* means that a write from any of the masters is immediately visible to other transactions running on other masters.
- *Transparent* means that applications do not have to worry about how the data is stored in multiple database servers, internally.

Figure 1 gives the architectural overview of Postgres-XC with its three main components:

1. *Global Transaction Manager (GTM)* gathers and

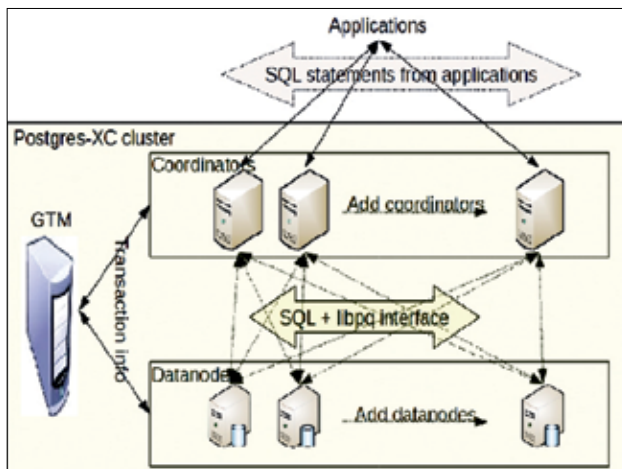


Figure 1: Postgres-XC architecture

manages information about transactional activities in Postgres-XC, issues global transaction identifiers to transactions (to maintain a consistent view of the database on all nodes), and provides ACID properties. It provides support for other global data, such as sequences and time-stamps. It stores no user data, except control information.

2. **Coordinators** (masters) provide a point of contact for the application/client. They are responsible for parsing and executing queries from the clients, and returning the results (if needed). They do not store any user data themselves, but gather the data from datanodes, with the help of SQL queries fired through a PostgreSQL-native interface. The coordinators also process the data if required and even manage the two-phase commit. Although coordinators do not store user data, they use the catalogue data to parse queries, resolve symbols, plan queries, locate data, etc.
3. **Datanodes** store user data and catalogues. The datanodes execute the queries received from the coordinator and return results to the coordinator.

Distribution of data and scalability

Postgres-XC allows two ways of storing the tables on the datanodes:

1. **Distributed tables:** A table is distributed on a given set of datanodes using strategies like hash, round-robin, or modulo partitioning. Every row of a distributed table resides on a single datanode. Multiple rows can be modified or written in parallel to various datanodes; we can also read the rows from various datanodes in parallel. Performance is greatly improved by parallel writes and reads from different datanodes.
2. **Replicated tables:** A table is replicated on a given set of datanodes using statement-level replication, which performs better than log-based replication, since the size of the logs that must be shipped is

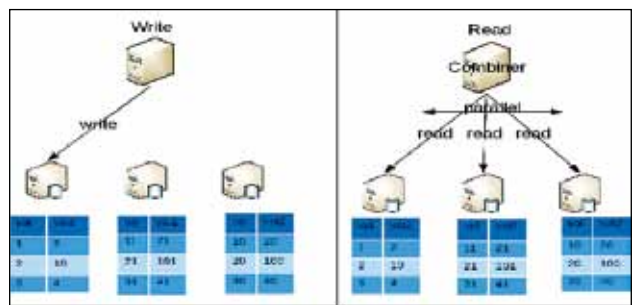


Figure 2: Distributed tables

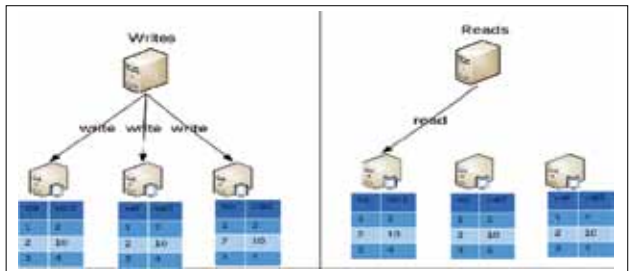


Figure 3: Replicated tables

much greater than the size of the statement. In the case of a replicated table, a row in the table resides on each datanode on which the table is replicated. Any modifications to the row must be duplicated to each replicated copy. Since all the data in the table is available on a single datanode, the coordinator can gather all the data from a single node and in some cases, act as a proxy between the client application and the datanode. This allows multiple queries on the same table to be directed to different datanodes, thus balancing the load and increasing the read throughput.

Figures 2 and 3 depict the read and write concepts for distributed and replicated tables, respectively.

High availability

To achieve high availability, one needs data redundancy, component redundancy and automatic failover. In Postgres-XC, data redundancy can be achieved by using the PostgreSQL native replication with Hot Standby for datanodes. Since each coordinator is a master (capable of writing data) and is capable of reading writes performed by any other coordinator instantaneously, every coordinator is capable of replacing any other, should that coordinator fail. GTM-standby acts as a redundant component for GTM. However, third-party tools are required for automatic fail-over of all the three types of components.

Performance evaluation

Initial transaction throughput measurements carried out using the DBT-1 benchmark have shown significant throughput scalability, as shown in Figure 4. The figure plots the Scaling Factor versus the Number of Servers in Postgres-XC. The Scaling Factor is the ratio of the number

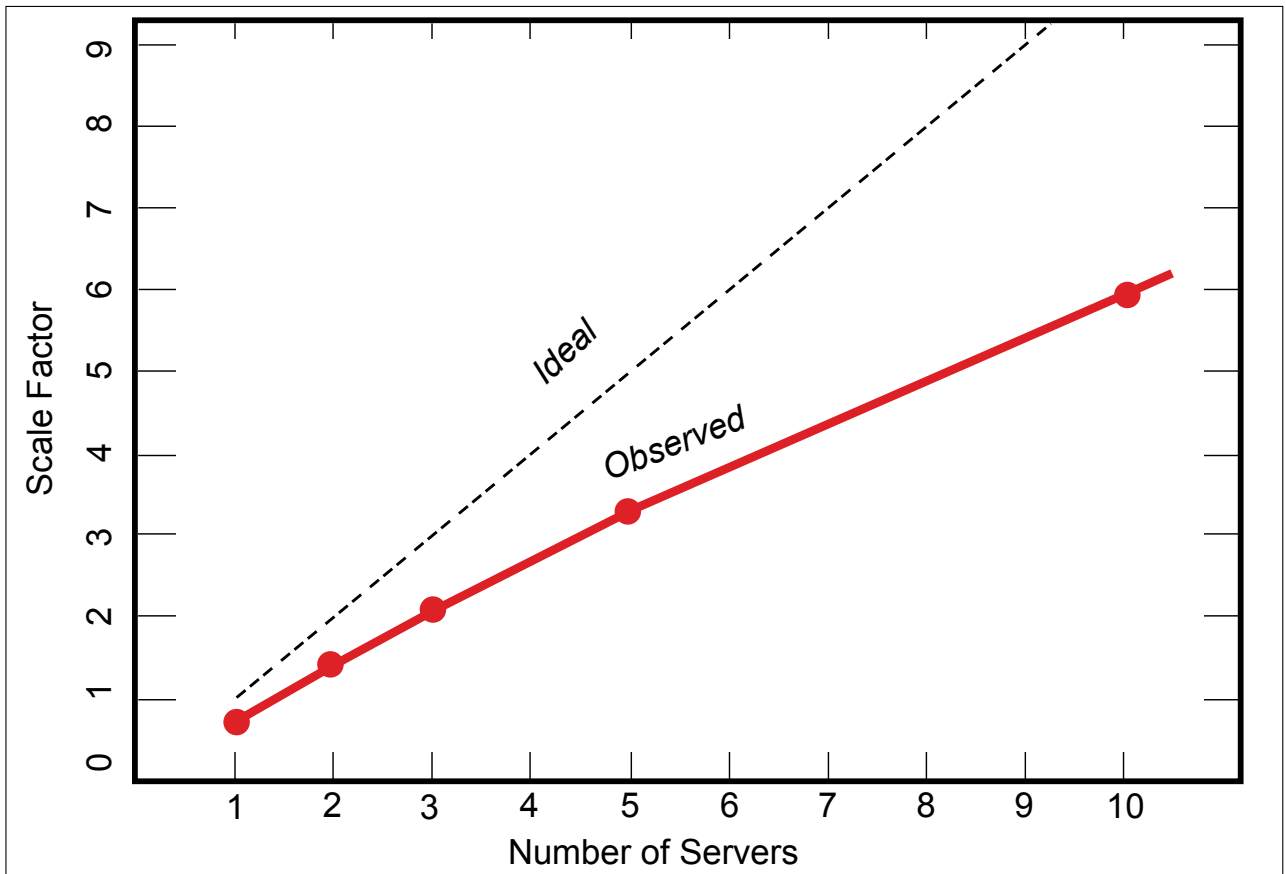



Figure 4: Performance evaluation

of transactions completed per unit time by Postgres-XC, to that completed by PostgreSQL. A Server comprised of a coordinator and a datanode run on single machine. This benchmark demonstrated an improvement in throughput of approximately 6 times, when using 10 servers.

Release management and development processes

The Postgres-XC project is hosted at <http://postgres-xc.sourceforge.net/>. The Postgres-XC team tries to release a minor version of Postgres-XC every three to four months to ensure that the latest Postgres-XC features are available to users. The team also tries to make the latest PostgreSQL features available in Postgres-XC by doing frequent merges with the latest stable release of PostgreSQL. The last release (0.9.6) of Postgres-XC supports most of the SQL syntax and features of PostgreSQL 9.1. The team is currently working on the first major release of Postgres-XC 1.0, due in 2012, with maximum PostgreSQL compliance. Some of the other features like the dynamic addition and removal of components, global deadlock detection, global constraint support, etc, will be targeted for major release after 1.0.

The development team follows the open source development model, where the issues, features or any

other development related items are discussed on the public mailing list postgres-xc-developers@lists.sourceforge.net. The mailing list postgres-xc-general@lists.sourceforge.net is used to discuss other Postgres-XC matters and to solicit help about Postgres-XC. 

Wish to contribute?

The Postgres-XC team needs help with feature development, bug fixing, creating installers and distribution packages, testing, and evaluation of Postgres-XC on real applications. To be part of the Postgres-XC community, please feel free to contact the Postgres-XC team at the appropriate mailing list

By: Ashutosh Bapat Susan Douglas

Ashutosh is a software engineer working at EnterpriseDB. He has more than 6 years of experience in DBMS development, with special expertise in distributed RDBMS. He is one of the core members of the Postgres-XC development team. His other interests include distributed computing, music technologies and sound engineering.

Susan is a technical writer and editor for EnterpriseDB, and co-author of PostgreSQL (Sams Publishing, 2005). In her free time, she enjoys beekeeping, gardening and horseback riding.