the Availability Digest

www.availabilitydigest.com

Master/Slave Replication with Continuent's Tungsten

May 2009

The Tungsten replication engine from Continuent, Inc. (www.continuent.com), provides asynchronous heterogeneous replication between MySQL and Oracle databases. Tungsten is open-source and is also available in an enterprise edition supported by Continuent.

In a previous paper entitled "Scaling MySQL with Continuent's uni/cluster," we discussed Continuent's uni/cluster, a two-node active/active system. The two nodes are LAN-connected, and each node connects to one or more MySQL database servers. The MySQL databases are kept synchronized by synchronous replication. Reads are spread between the MySQL databases for scaling. Should a node fail, all transactions are automatically routed to the surviving node. Uni/cluster runs on commodity hardware under Linux.

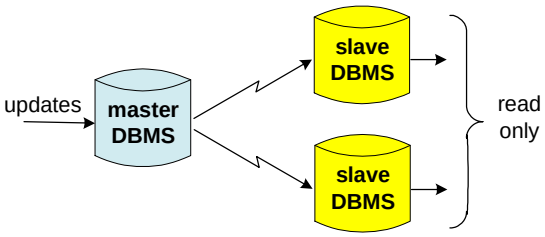
In this article, we review Continuent's Tungsten asynchronous-replication engine, which provides master/slave replication between MySQL and Oracle databases as well as to other JDBC-compliant databases.

Tungsten Replicator

The Tungsten Replicator provides asynchronous master/slave replication between off-the-shelf SQL databases running on commodity hardware.

Tungsten Master/Slave Replication

In a master/slave configuration, one master database replicates its updates to one or more slave databases. The master database is available to all applications, and all updates are made to the master database. The slave systems are available for read-only applications such as queries, reports, and backups.



Master/Slave Replication

Master/slave configurations bring many advantages to an application:

- *Availability* - High availability can be achieved since a slave can be promoted to master status should the master database fail.
- *Scaling* – Reads can be spread across many slave copies of the database.
- *Data Locality* – Database copies can be located at remote clusters of users.

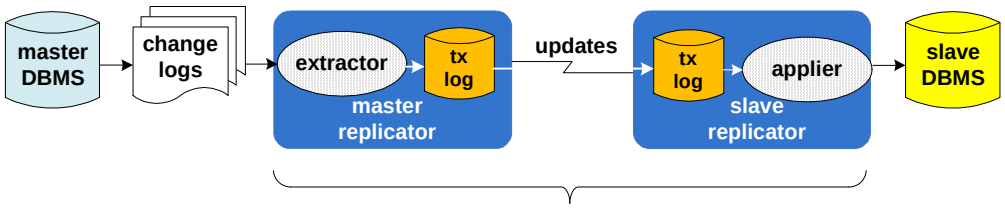
- *Disaster Recovery* – A current copy of the database can be maintained at a remote site to be available in the event of a primary-site failure.
- *Change Data Capture* – Changes to independent databases can be fed to a centralized database, such as a data warehouse, for enterprise-wide data consolidation.
- *Cross-Site Clustering* – Application processing can be distributed to multiple sites for load balancing and disaster recovery by partitioning the database.
- *Zero-Downtime Upgrades* – Upgrades may be rolled through the system without denying access to users.

Tungsten supports asynchronous master/slave replication. As changes are made to the master database, they are replicated to the slave database copies. Replication may either be via SQL statements or via rows. In addition to inserts, updates, and deletes, Tungsten also supports schema changes, stored procedures, views, and triggers.

MySQL, Oracle, and PostgreSQL databases are supported as master databases. They and any other JDBC-compliant databases may be mixed in a master/slave configuration as slave databases.

Architecture

The Tungsten replication engine comprises an Extractor and an Applier. As changes are made to the master database, a record of those changes is entered into a change log of some sort. For Oracle databases, this is the Redo Log maintained by Oracle for transaction integrity. For MySQL, it is the binary log that contains all SQL statements that modify the database.



Tungsten Replication Engine

The Extractor at the master database reads changes made to the master database from the change log and writes them to Tungsten's transaction history log. These changes are then sent over the communication channel to each slave, where they are written to the slaves' transaction history logs. At each slave is an Applier that reads the changes from its transaction history log and applies the changes to the slave database.

The transaction history logs provide persistent storage for replication events for recovery in the event of a failure.

Replication may be by row or by SQL statement.

Both the Extractor and the Applier support pluggable modules to support any target entity. The Tungsten Replicator supports any JDBC-compliant database. Continuent has certified Oracle, MySQL, and PostgreSQL, as master databases. It has also certified these databases along with SQL Server and DB2 as slave databases. In addition, modules may be written to replicate data to applications, to message queues, to flat files, or to any other target.

Included in the Extractor and Applier are exits for user-provided filters that can transform data as it is replicated to targets with schemas different from that of the master database. These exits may be written in Java or in Java Script.

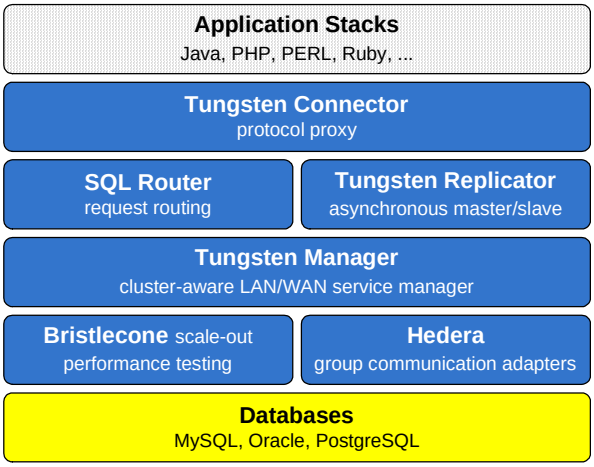
The replication engine can provide incremental consistency checks to validate that the slave databases are, in fact, consistent with the master database without stopping the applications or without stopping data replication.

The Tungsten Replicator runs on commodity hardware. It can be installed on any system that supports Java. Continuent has currently certified Tungsten for the Linux and Windows operating systems.

Tungsten Stack

Tungsten is organized as a stack comprising layers that interact to provide replication and master/slave cluster management. The modules in the stack are largely written in Java. The stack includes:

- Tungsten Connector, a protocol proxy for connecting applications to the Tungsten Replicator.
- Tungsten Replicator, which provides master/slave replication.
- SQL Router, which routes read and write requests to the appropriate database.
- Tungsten Manager, which provides cluster-aware services.
- Hedera, which provides group communications between cluster processes.
- Bristlecone, a performance testing tool.



The Tungsten Stack

Tungsten Connector

The Tungsten Connector is an optional proxy for MySQL and PostgreSQL databases. It allows applications to connect transparently to Tungsten master/slave clusters. It provides load balancing of read requests against multiple slave databases. Should a database fail, it provides transparent failover without session loss to a surviving database if necessary. MySQL and PostgreSQL clients can connect without changing libraries as the Tungsten Connector provides an interface that is identical to that provided by the native database.

Tungsten Replicator

The Tungsten Replicator moves data between live database copies, as described above.

SQL Router

The SQL Router is an optional module that is responsible for the proper routing of requests to the appropriate databases. It will route change requests to the master database and will route query requests to the best database, master or slave, based on a quality of service specification. It handles load balancing and capacity expansion, and it supports failover from a failed master to a promoted slave.

Tungsten Manager

The Tungsten Manager handles the policies for failover following a database fault. It monitors the systems in the cluster by using Hedera group communications to implement cluster membership and to provide distributed messaging between services running on different hosts.

Should Tungsten Manager determine that the master database has failed, it promotes one of the slave databases to be the new master and notifies the other slaves of the role switch. Failover is executed with a single manual command. Failover can be automated by integrating the failover command with a cluster manager.

Hedera

Hedera provides a group communications facility to exchange messages between processes within a cluster. Group communication is needed to support replicated databases, where the replicas are managed by different database controllers.

Bristlecone

Bristlecone provides tools for testing database performance. It can generate mixed loads of inserts, updates, and deletes. Benchmark test cases can be run with systematically varying parameters. It is useful for quantifying read and write scaling, replication latency, and transaction throughput.

Tungsten Enterprise

Tungsten Enterprise is a Continuent-supported version of the open-source Tungsten stack. It brings several additional capabilities to the open-source version:

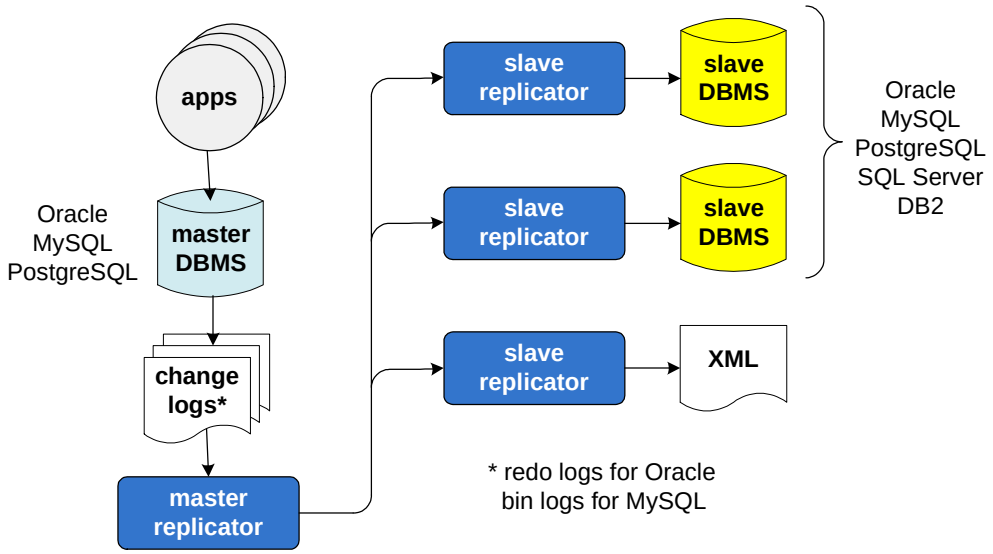
- Oracle is supported, using its Redo Logs as the source for replication.
- A single slave database can be fed by multiple masters in a fan-in configuration, allowing the support of data warehouses.
- Efficient administration tools are provided.
- Performance and robustness are enhanced.
- Installation utilities are provided.
- The stack modules are fully documented.
- Regular releases are provided.
- Continuent provides technical support and maintenance.

Heterogeneous Support

The open-source version of Tungsten is aimed at replication between MySQL and PostgreSQL databases. By using Continuent's supported Tungsten Enterprise edition, Oracle replication is added and allows heterogeneous replication between MySQL, PostgreSQL, and Oracle databases. All Oracle editions are supported, including Express, SE, and EE.

With this capability, the master database may be MySQL, Oracle, or PostgreSQL. If Oracle is used, Tungsten uses the Oracle Redo Log reader to read changes from the Oracle Redo Log. Since the Redo Log reader runs only under Linux, an Oracle master must run on a Linux platform.

In addition to the master databases, all of which may also be slaves, Continuent has certified SQL Server and DB2 as slave databases.



Tungsten Heterogeneous Replication

MySQL and PostgreSQL masters and any slaves, Oracle included, can run on any platform that supports Java. Currently, Linux and Windows platforms have been certified by Continuent.

If replication is heterogeneous, only row replication may be used. In addition, rows must contain a primary index. To account for differences in schema in different databases, user-provided filters written in Java or Java Script may be incorporated into the Tungsten Replicator to filter rows and to modify, delete, add, or modify columns in the slave database. In addition, modules can be written to allow replication to unsupported targets such as applications, message queues, and flat files.

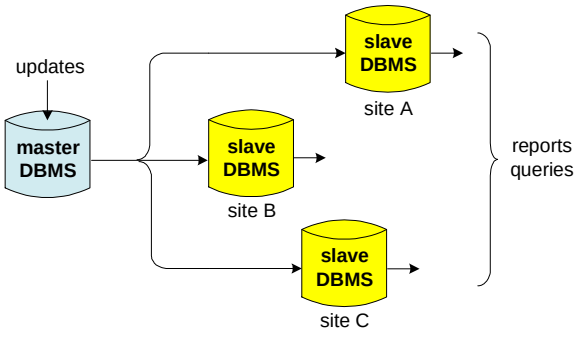
Configurations

The Tungsten Replicator can be configured in many ways to achieve a variety of objectives.

Scale-Out / Disaster Tolerance

Master/slave configurations are useful for scaling applications that are read-intensive. One master may feed many slaves. Read requests are distributed among the slaves to balance the load across the system.

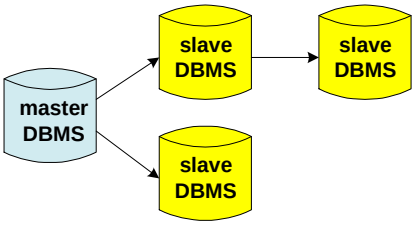
Capacity expansion is straightforward. New slaves can be easily added to increase the capacity of the system. Likewise, slaves can be removed if capacity requirements should shrink.



Scale-Out

The slave databases can be geographically distributed to provide local database access to remote communities of users. Geographical distribution also provides disaster tolerance. In the event that the master site should be put out of commission, a slave at a remote site can be promoted to master, allowing the system to continue to provide user services.

Chaining / Data Locality



Chaining

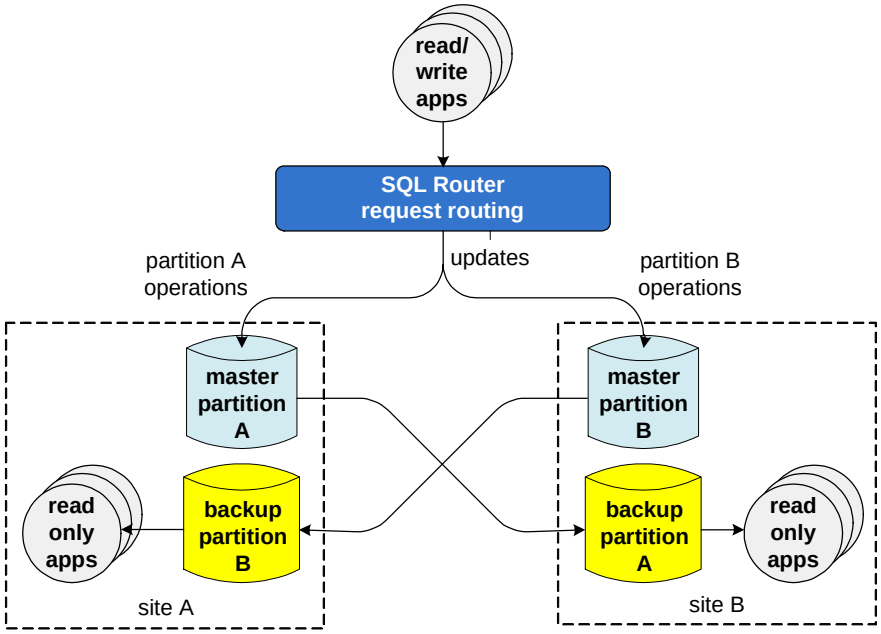
Slave databases can be chained. With chaining, a slave database can act as a master database to one or more subservient slave databases. Changes made to it are in turn replicated to other slaves.

With chaining, communication costs between geographically-distributed slaves can be minimized. This encourages the use of slave systems to provide data locality to remote communities of users. Though user updates must be forwarded over the network to the master database, reads can be executed by a slave at or near a user community, thus significantly improving read performance by eliminating communication latency.

Master/Master

Tungsten supports partitioned active/active systems by allowing multiple sites to actively participate in the same application. At each site is a master database and one or more slave databases. In this configuration, the database must be partitioned so that each master 'owns" a particular partition. Only the partition owner can update that partition. Tungsten does not support unpartitioned master/master configurations because it does not handle data collisions that occur when two sites update the same data object at the same time.

Each site has one or more copies of its partition on slave systems at other sites. Updates must be routed to the site containing the master database for the partition to be updated. After applying the updates, that master replicates the changes to its slave backups at other sites.



Partitioned Active/Active

To support this configuration, SQL Router is being enhanced to provide update transaction routing. SQL Router will analyze each SQL statement, will determine whether it is an update operation, and if so, will route it to the appropriate master owning the partition to be updated.

If the operation is a read operation, SQL Router determines to which slave database to send it for execution. Therefore, all databases in the application network are effectively used. Both reads and updates are distributed across the application network for maximum performance.

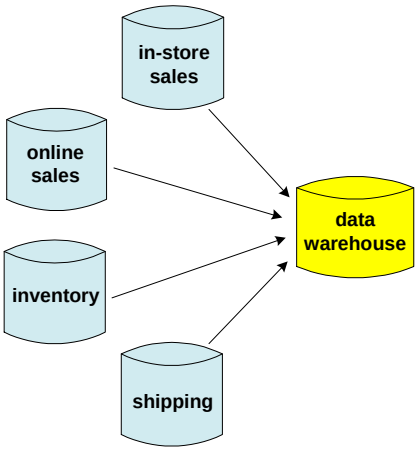
SQL Router currently handles the proper serialization of transactions and configures around network failures.

As an active/active system, this configuration also provides high availability. Should a master node fail, the master database at another site is assigned the partition originally owned by the failed master. SQL Router will now route all updates for the failed partition to the partition's new master, thus guaranteeing continuous operation.

Fan-In / Data Warehousing

The Enterprise edition of Tungsten supports fan-in, in which multiple master databases can replicate to one or more slave databases. With Tungsten's transformation and filtering capabilities, the unique schema of each of the independent masters can be modified on-the-fly to match the schema of the slave databases.

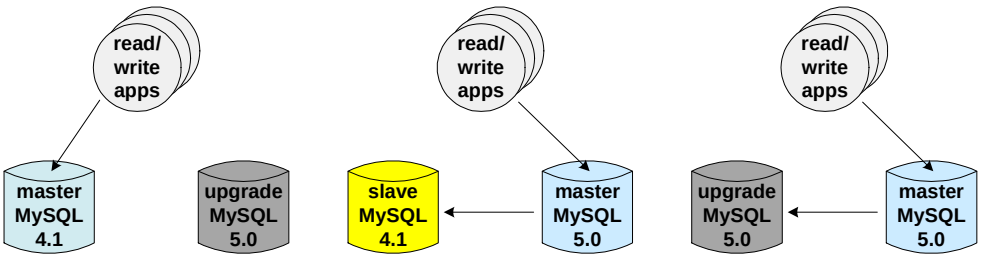
This fan-in capability is especially important to create data warehouses and data marts. Since Tungsten replication occurs in near-real time, this means that the data warehouses can be used not only for classical strategic data mining but for tactical real-time business intelligence as well. Typical real-time business intelligence applications include fraud detection, instant promotions to online or in-store customers, and rapid product reordering.



Real-Time Data Warehouse

Zero-Downtime Upgrades

The fact that there are two or more current copies of the database in the application network allows upgrades to be made without denying service to the system's users. This applies to hardware, operating system, application, and database upgrades.



upgrade slave **test upgrade** **upgrade old master**

Zero-Downtime Upgrades and Migrations

Zero-downtime upgrades are accomplished by rolling upgrades through the system, one node at a time. The process is started by taking a slave node offline. This node is then upgraded and is promoted to master, forcing the original master to now be a slave node. Replication is initiated from the new master to the new slave so that a failback capability is created. Since Tungsten supports data transformations during the replication process, there is no problem in replicating from a newly upgraded system to the old system running on the node that is now the slave.

The upgrade can be thoroughly tested in this new configuration. If there is a problem, the roles can be reversed and the upgrade problem corrected. If the upgrade works properly, the current slave node is taken offline, upgraded, and returned to service as the slave. The roles can now be reversed once again if it is desired to return to the original master/slave configuration.

Fault Recovery

Tungsten implements fault recovery via virtual IP addressing. All applications send updates to a virtual IP address that is owned by the current master node. The master also sends heartbeat messages to the slave nodes.

Should a slave node fail to receive a heartbeat from its master node, it informs the Tungsten Manager of this situation. Using the Hedera group communications facility, the Tungsten Manager will determine if the problem is a real failure of the master node or, instead, a network problem.

If it is determined that the master has, in fact, failed, the Tungsten Manager will determine which slave is the best candidate to be the new master. It might do this by choosing the slave that is the most up-to-date or by choosing a slave based on an established priority. Tungsten Manager will direct that slave to assume the virtual IP address and will promote it to master. The applications are unaware of this change and simply keep sending update requests to the virtual IP address and thus to the new master.

This failover scenario is not trivial. The failure of the master must first be detected. The best slave to be promoted to master must be determined, and all of the other slaves must be notified of this promotion. The other slaves must then reconnect to the new master. To solve this problem, Continuent provides a fully-integrated failover command as part of Tungsten Enterprise.

By geographically separating the master and slave nodes, this failover capability provides full disaster tolerance for the application network.

Continuent

Continuent is a privately-funded company with a strong Nordic heritage. Its corporate office is in San Jose, California. It has a sales office in Espoo, Finland.

Continuent products are used by such companies as Capgemini, Telstra, Sonoma, Thomson, La-Z-Boy, Alcatel-Lucent, CNET Networks, and NOAA.