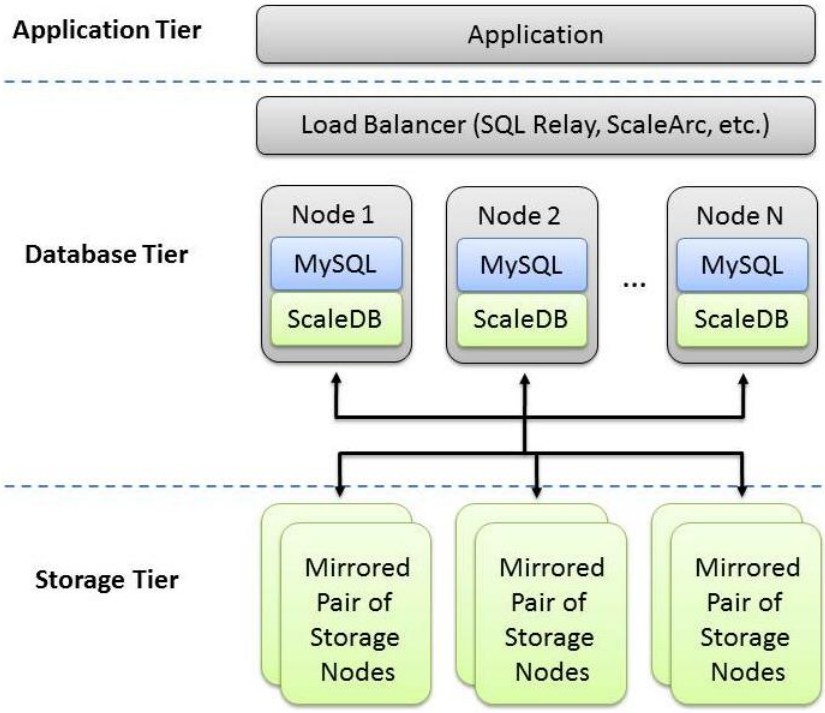
ScaleDB: Multi-Tenant (Cloud & SaaS) Solution

Supporting multi-tenant situations introduces a number of challenges around scaling, ensuring quality of service (QoS), fulfilling service level agreements (SLAs), demands for data isolation due to security concerns, maintenance issues and much more. Traditional SQL databases are not designed to address such issues, but NoSQL databases lack functionality, tools and ACID compliance that are also very important. ScaleDB extends MySQL, the world's most popular SQL database, and the one that is dominant in the cloud, enabling it to address the challenges of multi-tenancy.



How it Works

ScaleDB extends MySQL, turning it into a shared-data cluster. ScaleDB provides the green boxes above. It runs standard MySQL, which interfaces with ScaleDB through the standard Storage Engine API. Both the MySQL instances and the mirrored storage nodes run on standard virtual machines or commodity servers. The data is distributed across the mirrored storage nodes and each database instance operates across the entire collection of data in the storage tier. This architecture virtualizes the database, enabling nodes to be allocated on-the-fly from pooled compute resources. Running a MySQL application on a ScaleDB cluster involves only a configuration change; it does ***not*** require any change to the application of the MySQL code.

Supporting Multi-Tenant Environments

The following describes how ScaleDB uniquely addresses multi-tenant challenges:

1. Dynamic Scaling: Multi-tenant applications can experience extreme scaling demands and the database must be able to respond dynamically. If a compute bottleneck emerges, you can dynamically and automatically add database nodes. If an I/O bottleneck emerges, you can add storage nodes on-the-fly.

2. High Availability: A ScaleDB cluster has no single point of failure. The storage nodes are mirrored, so that failure of a storage node doesn't result in downtime or loss of data. The database nodes each have full access to the shared-data. If a database node fails, the database requests are routed to remaining nodes, again avoiding downtime or data loss.

3. Quality of Service (QoS): Cloud/SaaS offerings include a Service Level Agreement that defines a certain QoS level. Unlike static SQL databases, ScaleDB enables processes to be moved across a shared pool of compute resources to ensure that you are not overtaxing servers, which typically results in a degradation of performance.

4. Maintenance: With ScaleDB, servers can be taken out of service or put back into service on the fly, enabling maintenance without downtime.

5. No Slave Lag: Unlike replicated MySQL, ScaleDB does not require slaves to provide either fail-over or read scaling; those are inherent in ScaleDB's shared-data cluster. This eliminates troublesome and annoying slave lag issues, where the slave is temporarily out of sync with the master.

6. Write Scaling: Unlike shared nothing DBMS, which funnels all writes to dedicated masters, ScaleDB enables writes on any database node. This eliminates the write bottleneck and provides cluster-wide load-balancing.

Summary

As more and more applications move to the cloud and to SaaS architectures, they impose unique demands on the underlying database. At the same time, developers want the richness of features and the maturity that is found in MySQL. ScaleDB extends MySQL, supporting the unique demands of cloud and SaaS-based applications.

ScaleDB
3723 Haven Avenue
Menlo Park, CA 94025
☎ (650) 587-8787
(650) 587-1571
info@scaledb.com
www.scaledb.com