How to Eliminate MySQL Performance Issues

Learn how to resolve performance challenges associated with a MySQL DBMS, with the NuoDB NewSQL solution. NuoDB enables greater scalability while enhancing the efficiency of replication and better managing IT costs.

"The NuoDB solution has matured rapidly due to generous feedback from hundreds of dedicated users and learning derived from many proof-of-concept projects."

Barry Morris, CEO and Co-Founder of NuoDB

Executive Summary

Managing database applications in the datacenter or in the cloud is a balancing act. Optimizing your applications to keep costs under control; overcoming network latency and performance issues; maintaining transactional consistency across geographic regions; guaranteeing 99.99% or higher application availability; applying scale-out workarounds imposed on IT by 20th century database technology and a long list of other challenges all impact your ability to provide the best value to the business.

In this whitepaper, we examine how to resolve the challenges associated with a MySQL DBMS: scale out and performance issues; achieving rolling upgrades; optimizing hardware utilization; ensuring transactional consistency and obtaining zero downtime.

Conventional Approach to Solving MySQL Performance Pains

As with many traditional, 20th century database management systems, optimizing MySQL is complex. You must examine the hardware to obtain the right balance of CPU, memory and performance under heavy loads as MySQL is strongly limited by hard disk performance. This is especially true for write latency.

So sometimes, administrators tackle issues with cache hit ratios, memory consumption formulas or optimization through the user interface. Are you getting false positives from your monitoring tool? Better find out. Workload profiling? Indexing? Of course. The possibilities, as the saying goes, are endless.

MySQL replication is a very common approach for overcoming performance, scaling and high availability problems. In many cases, replication provides some relief but…then again the replication can grind to a halt due to errors. Restoring a MySQL replication is not simple. You can try setting locks on the master to get a consistent MySQL replication, however, during that time your website is unavailable. In a 24 X 7 X 365 world, that downtime is an issue. A big one.

Sharding MySQL is another option. Inherent in this approach are inevitable issues like network latency and your SQL will no longer be declarative. Also with functional sharding no single table can be larger than one instance.

There are any number of tricks-of-the-trade we haven't touched upon that will work if you do the work and lots of it. In fact, making MySQL perform better has been an art and a science since it was first released way back in the last century – May of 1995 to be exact.

Is it time to consider a new solution for a new age? An age where data has grown astronomically larger and larger; where the cloud is becoming commonplace; where the cost of RAM is negligible. For your next application or you next replication….consider NuoDB.

The New Approach

An ideal DBMS should scale elastically, allowing new machines to be introduced to a running database and become effective immediately.

NuoDB offers a proven NewSQL solution for eliminating the common issues associated with MySQL, all **without sacrificing the safety and familiarity** of SQL and ACID. It is an operational DBMS to handle transactions, interactions and observations – in the datacenter, in the cloud, across clouds, anywhere.

NuoDB's distributed three-tier architecture is the foundation for scale-out performance. It decouples management; transactions and storage, meaning each tier can scale a single, logical database elastically. It scales linearly to improve transactions per second performance and handle both concurrency and data volume. Out and in; by simply adding or removing processes.

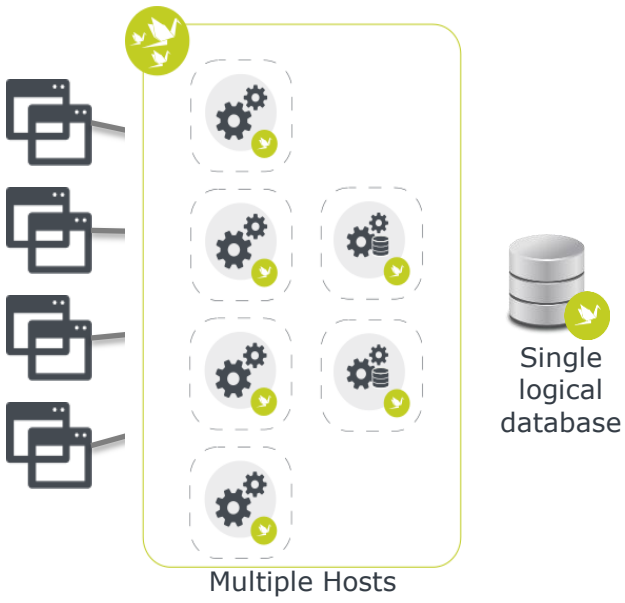

Figure 1: NuoDB's tiered architecture enables scale-out without sharding, provides built-in on-demand replication and offers easy migration from MySQL.

NuoDB provides support for most of the popular development languages and frameworks such as Java, .NET, Node.JS, Perl, PHP and Python. It's the ideal database for managing high-performance cloud, datacenter, mobile and SaaS applications without worrying about sharding and replication. Its innovative architecture is unique in that it allows you to build SQL / ACID compliant, geo-graphically distributed database solutions with guaranteed 24/7 availability, rolling hardware upgrades, built-in data redundancy and disaster recovery.

NuoDB performs all the coordinated actions you would expect from a MySQL database (like creating, deleting or updating a record) without relying on master-slave configurations or locking. It uses asynchronous, peer-to-peer communications to provide client applications with scale-out performance, low latency and transactional consistency. NuoDB is equally able to scale a single operational database or many separate databases, such as a series of virtualized databases hosted in Amazon EC2 or other cloud infrastructure.

To make things even easier, NuoDB behaves just like any other MySQL database as far as applications are concerned. It is ANSI SQL92 compliant with SQL99 extensions and works with many popular Object Related Mapping (ORM) tools such as Hibernate, PDO and Active Record. Although not plug-compatible with MySQL, NuoDB supports many of the common MySQL functions, which makes code migration of MySQL apps to NuoDB straightforward in most cases.

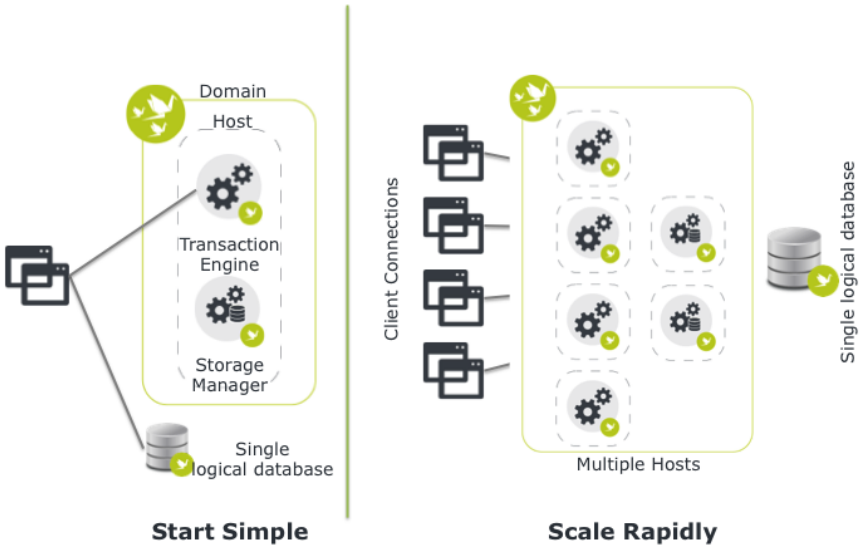

Figure 2: NuoDB's distributed database continues to present itself as a single logical database as it scales across virtual and physical resources.

Three core components make NuoDB powerful and simple to deploy:

- **The NuoDB Transaction Engine** is comprised of one or more processes running on compute hosts. These in-memory processes execute the SQL layer by operating on database Atoms—self-describing objects—and listen for changes and communicate changes with peers.

- **The NuoDB Storage Manager** is a specialized transaction engine that knows how to store and retrieve data on disk. Adding a new storage process automatically creates a copy of the entire database, which increases both performance and resiliency. Storage managers can leverage virtually any type of data storage that can manage a key value store, for example a local file system, Amazon S3, Hadoop Distributed File System (HDFS) or a SAN.

- **A NuoDB Broker** is a process running on at least one host that provides applications with SQL/JDBC access to the database.

NuoDB enables you to configure and manage one or more databases within an administrative domain, referred to as the NuoDB Domain. Each NuoDB database within a domain can be as simple as a single Transaction Engine, a single Storage Manager and a Broker. Each database can be configured to run a number of Transaction Engines and Storage Managers across hosts in a datacenter, in the cloud or a combination of the two.

Unlike any other DBMS, NuoDB can provide a single, logical database view for databases that scale across datacenters and geographically distributed regions. Obviously, this helps with high-availability applications but it's also critical for any application where users are physically separated and you want to get their applications and data closer to them. NuoDB supports geographic distribution because data tends to be localized relative to the clients that access it. Between simple load-balancing, on-demand caching and asynchronous replication between regions, the single NuoDB database keeps most coordination messages local to a given region.

NuoDB's atom structure is a powerful component of the architecture that supports scalability. However, without some way of handling conflicts and enforcing consistency, NuoDB couldn't support ACID semantics. For this, Multi-Version Concurrency Control (MVCC) is used. Unlike global lock-managers or distributed transaction coordinators, MVCC works by treating data as versioned, and treating the database overall as an append-only set of updates. Because MVCC gives NuoDB an efficient snapshot isolation mode, it's very good at supporting **operational** and **analytic** workloads against the same data. Long-running, read-only queries run without conflict against data that is frequently updated. This helps simplify deployments that would otherwise be running multiple databases while trying to synchronize the data set. [For more information, see the InfoQ article, Exploring the Architecture of the NuoDB Database]

Solving MySQL Performance Issues

From an historical view, ease of installation for small deployments contributed to the popularity of MySQL for powering web, social media and Software as a Service sites. As the World Wide Web grew exponentially, traffic to successful websites grew beyond what a single server could handle. So the need for scale out grew and typically required some form of MySQL replication. Not always a straightforward experience, MySQL replication has created a proliferation of books, third-party tools and even entire products aiming to tame it and mitigate the associated pains of scaling.

DBAs and DevOps personnel have had to become intimately familiar with its many quirks, and maintaining large MySQL replication deployments has quickly become a full-time job in many companies. The new-generation technologies implemented in NuoDB provide much easier scale-out from one to 100 servers and beyond, and remove many of the limitations inherent to the approach used in MySQL. The following are some of the issues organizations run into with MySQL replication and the ways NuoDB's architecture bypasses them **with considerably less development effort and operator involvement:**

Masters and Slaves

Replication in the standard MySQL product is based on the concept of master and slaves, where all updates flow from the master to the slave in a pre-configured hierarchy of database machines. In NuoDB, each process is an equal peer to each other, so there are no masters or slaves. This fact alone greatly simplifies management of many processes, since there are no master machines that require special treatment and no slave machines that must be configured to replicate from a particular master.

Setting up Replication

NuoDB supports easy provisioning of new servers, while adding a new slave to an existing MySQL setup requires several steps. Since it is unlikely that all the binary logs that are required to create a slave from scratch have been preserved, usually an initial, consistent snapshot of the data needs to be transferred to the slave first using some other means before actual replication can begin. NuoDB provides **"one-click"** creation for both Storage Managers and Transaction Engines. Storage Managers will automatically synchronize with the rest of the database, even if they start up empty. The Transaction Engines will self-organize and fetch whatever data is required to process queries without having to be pointed to it. To MySQL, multi-master—or being able to perform updates on multiple machines at once—does not come out of the box; it requires the use of customized setups or third-party products or tools. By that definition, NuoDB is multi-master from the start, as soon as a second host is added to the database.

Running Applications in a Replicated Environment

Unlike MySQL, NuoDB is always consistent. A dynamic NuoDB cluster is more than a collection of separate databases that are replicated from one another. It is a unified whole that is always consistent and presents a single interface to the application, even if the cluster is spread across multiple datacenters. The application can select the appropriate transaction isolation level for each transaction, and all NuoDB servers will work as one to provide the expected transactional semantics.

With NuoDB, there is no difference between reads and writes. In the replication that comes with standard MySQL, update queries can only be sent to the master server. The separation of writes from reads of a workload does not come naturally, so dedicated drivers, proxies or frameworks are required. In NuoDB, reads and writes are treated identically and any host process can handle both requests, so any transaction can be executed from any connection. The application needs to open and manage just a single database connection. Unlike in NuoDB, updates to the master and reads from the slave in MySQL can not participate in the same transaction and therefore can not be fully consistent with each other.

There are various instances where MySQL replication can "break", meaning that the flow of replicated data from the master to its slaves stops. One example would be if the data on the slave is changed outside of replication, such as by operator error. NuoDB does not require that DBAs partake in the inner workings of the replication—there is no need to configure replication. The NuoDB database presents a unified view of a single collection of data, so there is no opportunity for a single change made on a single slave to cause replication to stop. There is no need to explicitly set slaves to be read-only to shield them from harm.

Availability and Handling a Failure

A failure in a MySQL master causes the cluster to become unavailable for writing until a slave is promoted to take its place and the entire replication topology is reconfigured. This process does not happen automatically and requires a separate monitoring mechanism to detect failures of the master. NuoDB includes built-in monitoring of all processes, which allows non-responsive processes to be removed automatically from the database. In addition, every Transaction Engine is ready to receive the clients of the failed process without an external failover and promotion procedure. Every Storage Manager maintains a complete copy of the database, so that it will continue to record transactions to disk without interruption. Since no process receives special treatment, the failure of any individual Storage Manager or Transaction Engine does not impact the availability of the database as a whole. New client connections are automatically directed to one of the remaining processes.

In MySQL, a node that was previously a master cannot rejoin the cluster back in its previous role without going through a re-synchronization process and a change in the replication topology. In NuoDB, a process that has gone offline for some reason can rejoin the database without any special administrative procedures. Bringing it up to speed with the rest of the processes happens automatically and it is ready to begin serving requests. The reduced downtime per process in turn increases the overall availability of the system. [For more information, see the Techblog, NuoDB Scalability Removes MySQL Replication Pain Points]

Moving to On-Demand Data Replication

With MySQL deployments, typically all of the data is replicated from an active master database to one or more passive slave databases. This leads to synchronization challenges in a distributed environment. NuoDB takes a different view of what should be replicated and why it should be replicated in its distributed database architecture. NuoDB enables both on-demand replication at the Transaction Engine level for scaling and complete asynchronous replication at the Storage Manager level. The combination of these two approaches simplifies and streamlines replication.

Explicit, full database replication schemes used in MySQL deployments usually rely on a separate caching tier with the associated application logic and often a sharded database, where each shard may be replicated to one or more locations. There are usually two reasons for explicitly replicating a MySQL database, either for high availability or scalability. For high availability, a single active database implements changes such as inserts, updates and deletes and pushes them out to a single, passive version of that database. The passive database supports durability and provides the enterprise with insurance because if the active host goes down applications can failover to the passive database. This one-to-one replication relationship provides the enterprise with a high-availability solution, but it offers limited growth potential.

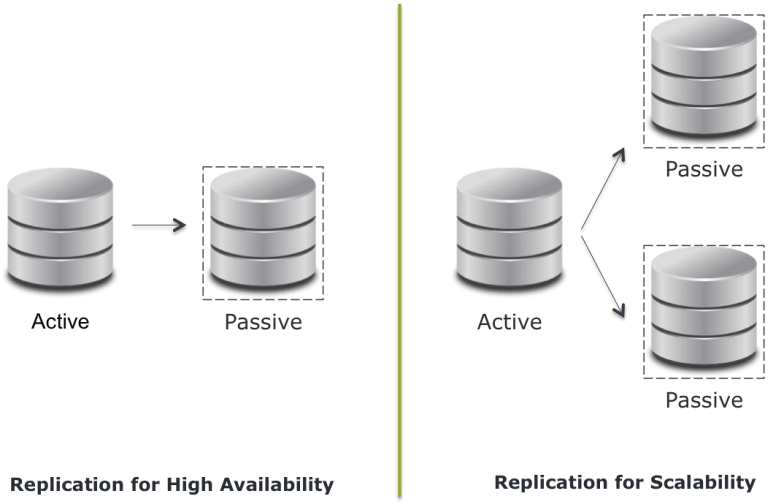

Figure 3: Replication is typically done for high availability, by pushing from a single active to a single passive database. Or, for better scalability by pushing from a single active to multiple passive databases.

The enterprise can more effectively scale out MySQL implementations by pushing information from a single active database to multiple passive databases. Any number of replicated passive databases can read data, but all changes still have to be made at the central active master database. This approach enables greater scalability, but there will always be a replication delay between updates to the active database and when the new information is pushed out to the replicated databases. Programmers need to understand how quickly the data is replicated, and operations personnel need to maintain multiple replication points, connections and failover schemes.

The enterprise can avoid the limitations of explicit, full database replication by migrating MySQL to NuoDB, which provides a single, global view of the database with a caching layer that scales. You don't have to configure replication, and you gain a consistent view of enterprise information that scales out, even when running across multiple datacenters. Atoms are accessed on-demand, avoiding the overhead, replication delays and administrative demands of explicit, full database replication.

For example, you could have one NuoDB Storage Manager and two Transaction Engines distributed across three machines. If a Transaction Engine needs an object from your durable store it will pull it into memory cache. You don't have to worry about failures—if the machine hosting the Transaction Engine fails, the object is still being maintained in your durable store. If the second Transaction Engine needs the object, it will be replicated from the cache of the first Transaction Engine because it is faster to get it there than from the durable layer. Again, if one of the machines hosting either of these Transaction Engines fails, the system still maintains its consistency and correctness. NuoDB manages the transactional consistency across multiple transaction engines enabling active / active replication.

From a database programmer's perspective, the advantage of this approach is the ability to always maintain a consistent view of the object—there are no concerns about inconsistent views of objects due to replication update times. NuoDB's default isolation model eliminates the inconsistency and delays of explicit, full database replication and provides greater scalability than MySQL deployments. In this example, you could easily add a third Transaction Engine; it doesn't have to wait until it hears any explicit replication messages before it interacts with the data from the database. This architectural difference supports geo-distributed data management and allows NuoDB to replicate effectively across multiple regions.

Since NuoDB offers an on-demand cache and does not try to explicitly replicate every object in every location, IT does not have to worry about active versus passive replication schemes. The enterprise can rely on consistent replication and a consistent

view into enterprise resources, and avoid the negative side effects of explicit, full database replication while gaining the ability to scale out gracefully whether you have a single datacenter or datacenters in multiple regions worldwide. [For information, view the Data Replication video demo by NuoDB CTO Seth Proctor]

Migrating a MySQL App to NuoDB

Whether you are looking to avoid MySQL performance pains by migrating an existing MySQL application to NuoDB or seek to run a MySQL application in parallel on NuoDB to support analytics or to serve as a trial before migrating, the NuoDB NewSQL database's peer-to-peer architecture manages to remove many of the sources of difficulty when migrating or replicating data.

You can augment your existing MySQL infrastructure or gradually migrate from MySQL to NuoDB. You maintain control over the migration pace, and scale-out, consistency and reliability are achieved with reduced administration effort and without having to make changes to the application itself. NuoDB provides a migration tool that enables easy evolution from MySQL.

Just download NuoDB, install it and then run the NuoDB migration tool. The actual migration is just three simple steps:

1. Migrate the database schema from MySQL to NuoDB using a single command. At this point you will have migrated the schema and table information from MySQL to NuoDB.

2. Load the data from the MySQL database using a second command. Now the table data from the MySQL database will be stored using comma separated value (CSV) format in the /tmp directory.

3. Load that data into the NuoDB database using a third command. You then change the server to use the NuoDB data source and restart the server to complete the migration. [For more information and a detailed example, see the Techblog, Migrating a MySQL Application to NuoDB]

If you'd like to run NuoDB side-by-side with MySQL, view the webcast, Transitioning to NuoDB with the Tungsten Replicator. This step-by-step webinar uses the Tungsten Replicator to show how easy it is to replicate from MySQL to NuoDB. It explains how to replicate your MySQL application to NuoDB and how to augment your MySQL deployment with a NuoDB Operational Data Store.

Summary

The enterprise can quickly and easily migrate existing MySQL deployments to NuoDB to avoid the pain points of MySQL. Organizations can avoid the cost and distraction of sharding and enable greater scalability while enhancing the efficiency of replication and better managing IT costs. The simplicity of the NuoDB architecture means that no special replications are needed for managing a separate reporting and analytics infrastructure, and it allows the enterprise to eliminate sharding and all of the replicas that were previously needed to support high availability demands.

NuoDB also provides:

- **Scale-out performance—**An ideal DBMS should scale elastically, allowing new machines to be introduced to a running database and become effective immediately. NuoDB's distributed three-tier architecture is the foundation for this exact kind of scale-out performance. It decouples management; transactions and storage, meaning each tier can scale a single, logical database elastically. It scales out to improve transactions per second performance and handle both concurrency and data volume. NuoDB scales out and in, simply by adding or removing processes.

- **Zero downtime—**NuoDB lets you build a global data management platform with guaranteed high availability, rolling upgrades, built-in data redundancy and disaster recovery. Store data in whatever storage system is most appropriate—on a directly attached file system, a local key/value store or on a cloud-based storage service. It provides an administrative interface to bring new resources online in addition to a scriptable interface. Both are able to dynamically provision additional resources when demand exceeds capacity or when hardware fails. The very nature of the NuoDB distributed architecture means there is no single point of failure and therefore zero downtime.

- **Geo-distributed data management—**Today's applications are deployed at global scale. Often this results in a poor user experience where latency and responsiveness reduce human productivity. NuoDB lets you distribute a single database across geographies. This approach provides a significantly improved experience for your customers. NuoDB's geo-distribution enables you to build an active/active, highly responsive database for high-availability and low latency. This allows you to bring your database closer to your customers for a better and more productive user experience. Geo-distribution also eliminates the need for complex replication, backup and recovery schemes.

NuoDB is an elastically scalable distributed database that provides developers and administrators with single intuitive interface for centrally monitoring deployments. Download NuoDB for free today and see how easy is it to migrate your MySQL environment to the industry's only high-performance, distributed cloud database.

About NuoDB

Everyday businesses face serious challenges coping with application performance, maintaining business continuity and gaining operational intelligence in real-time. NuoDB leads the industry with a proven NewSQL solution to solve all these challenges. It provides a unique combination of scale-out performance, zero downtime and geo-distributed data management. It's an operational DBMS to handle transactions, interactions and observations anywhere.

Launched in 2010 by industry-renowned database architect Jim Starkey and accomplished software CEO Barry Morris, the company is based in Cambridge, MA. Used by thousands of developers worldwide, NuoDB's customers include automotive after-market giant AutoZone, NorthPoint Solutions and other innovative companies. The company is the recipient of numerous, prestigious industry awards including: Gartner Cool Vendor and The Red Herring Top 100. For more information, visit http://www.nuodb.com.

215 First Street
Cambridge, MA 02142
+1 (617) 500-0001
www.nuodb.com

Published September 10, 2013.