Guide to Scaling Web Databases with MySQL Cluster

Accelerating Innovation on the Web and in the Cloud

A MySQL® White Paper

October 2016

Table of Contents

Introduction

It should come as no surprise that data volumes are growing at a much faster rate than previously anticipated, with estimates predicting a doubling every 14 months. Increasing internet penetration rates, social networking, high speed wireless broadband connecting smart mobile devices, and new Machine to Machine (M2M) interactions are just some technologies fueling this growth.

The databases needed to support such a massive growth in both user load and data have to meet new challenges, including:

- **Scaling write operations**, as well as reads, across commodity hardware;
- **Real-time user experience** for querying and presenting data;
- **24 x 7 availability** for continuous service uptime;
- **Agility and ease-of-use,** enabling developers to quickly launch and scale new, innovative services.

It is important to recognize that not all data is created equal. It is not catastrophic to lose individual elements of log data or status updates – the value of this data is more in its aggregated analysis. But some data is critical, such as OLTP operations, ecommerce transactions, financial trades, customer profile updates, utilization and billing operations, ad placements, AAA data, service entitlements, real-time analytic results, etc.

Services generating and consuming this type of data need the underlying database to meet the challenges above, while still:

- Preserving transactional integrity with ACID compliance
- Enabling deeper insight by running complex queries against the data, increasingly generating results in real-time
- Leveraging the proven benefits of industry standards and skillsets to reduce cost, risk and complexity.

It is these types of workloads that MySQL Cluster is designed for. Examples include:

- High volume OLTP
- Real time analytics
- Ecommerce, inventory management, shopping carts, payment processing, fulfillment tracking, etc.;
- Online Gaming
- Financial trading with fraud detection
- Mobile and micro-payments
- Session management & caching
- Feed streaming, analysis and recommendations
- Content management and delivery
- Communications services
- Subscriber / user profile management and entitlements.

This Guide explores the technology that enables MySQL Cluster to deliver web-scalability with 99.999% of availability (i.e. less than 5 ½ minutes of downtime per year), and provides the resources to get you started in building your next successful web, mobile or cloud service.

MySQL Cluster Architecture

The realities of today's successful web and mobile services means developers and architects need to consider multiple dimensions to scalability:

- The need to automatically scale writes, as well as reads, both within and across geographically dispersed data centers
- The need to scale operational agility to keep pace with demand. This means being able to add and remove capacity and performance to the database, and to evolve the schema – all without downtime
- The need to scale queries by having flexibility in the APIs used to access the database – including SQL and NoSQL interfaces

The architecture of MySQL Cluster is designed to accommodate these requirements with the following capabilities – all of which are discussed in more detail during the Guide:

- Auto-sharding for write-scalability
- In-memory, real-time responsiveness
- Active/Active geographic replication
- Online scaling and schema upgrades
- SQL and NoSQL interfaces
- 99.999% availability
- GUI-based configuration and provisioning for DevOps ease-of-use.

MySQL Cluster is a highly scalable, real-time, ACID-compliant transactional database, combining 99.999% availability with the low TCO of open source. Designed around a distributed architecture with no single point of failure, MySQL Cluster scales horizontally on commodity hardware with auto-sharding to serve read and write intensive workloads, accessed via SQL and NoSQL interfaces.

MySQL Cluster's multi-master real-time design delivers predictable, millisecond response times with the ability to service millions of operations per second. Support for in-memory and disk-based data, automatic data partitioning (sharding) with load balancing and the ability to add nodes to a running cluster with zero downtime allows linear database scalability to handle the most unpredictable web-based workloads.

Alcatel-Lucent, Big Fish, PayPal, Shopatron, Telenor, Zillow and many more deploy MySQL Cluster in highly demanding web, mobile and cloud applications for the types of high-scale, mission-critical services discussed in the Introduction section.[1]

MySQL Cluster is also a key component of the MySQL Web Reference Architectures – a collection of repeatable best practices for building highly scalable and available web services, developed with the world's leading web properties. The Web Reference Architectures are discussed in more detail later in this Guide.

From an architectural perspective, MySQL Cluster comprises three types of node which collectively provide service to the application:

- Data nodes manage the storage and access to data. Tables are automatically sharded across the data nodes which also transparently handle load balancing, replication, failover and self-healing.
- Application nodes provide connectivity from the application logic to the data nodes. Multiple APIs are presented to the application. MySQL provides a standard SQL interface, including connectivity to all of the leading web development languages and frameworks. There are also a whole range of NoSQL inerfaces including memcached, JavaScript, REST/HTTP, C++ (NDB-API), Java and JPA.
- Management nodes are used to configure the cluster and provide arbitration in the event of a network partition.

The roles played by each of these nodes in scaling web and cloud databases are discussed in the following sections of this Guide.

[1] http://mysql.com/customers/cluster/

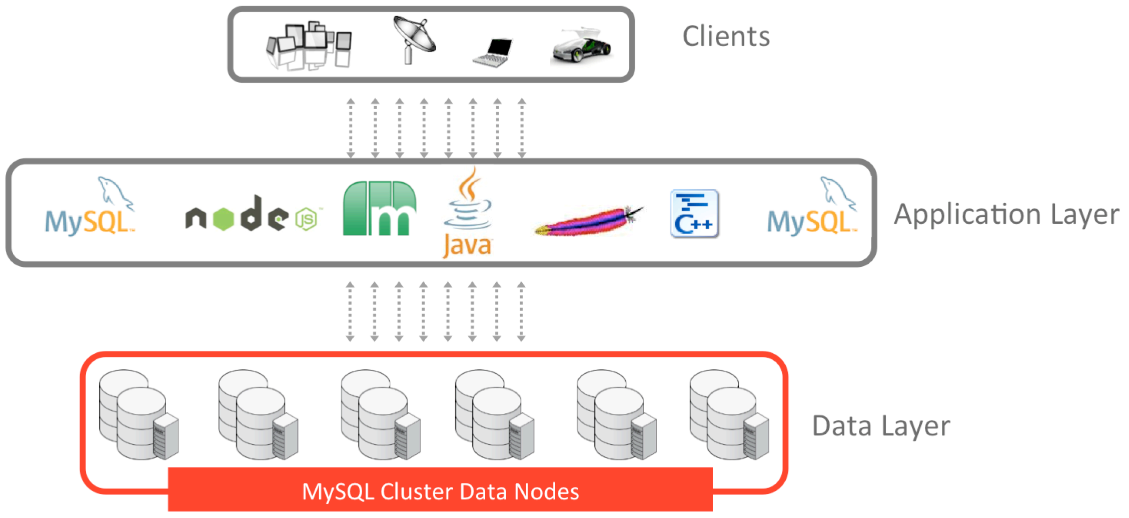


Figure 1: The MySQL Cluster architecture provides high scalability with no single point of failure

Scaling Write-Intensive Web Services

MySQL Cluster is designed to support both read and write-intensive workloads with the ability to dynamically scale from several to several hundred nodes – allowing services to start small and scale quickly as demand takes-off. As demonstrated later in this section, MySQL Cluster excels at both aggregate and per-node performance, eliminating the requirement to provision and manage racks of hardware that end up being poorly utilized.

Auto-Sharding

MySQL Cluster is implemented as a fully distributed multi-master database ensuring updates made by any client are instantly available to all of the other clients accessing the cluster. All data nodes can accept write operations and automatically resolve conflicting updates.

Tables are automatically sharded (partitioned) across a pool of low cost commodity data nodes, enabling the database to scale horizontally, accessed both from SQL and directly via NoSQL APIs.

By automatically sharding tables at the database layer, MySQL Cluster eliminates the need to shard at the application layer, greatly simplifying application development and maintenance. Sharding is entirely transparent to the application which is able to connect to any node in the cluster and have queries automatically access the correct shards needed to satisfy a query or commit a transaction.

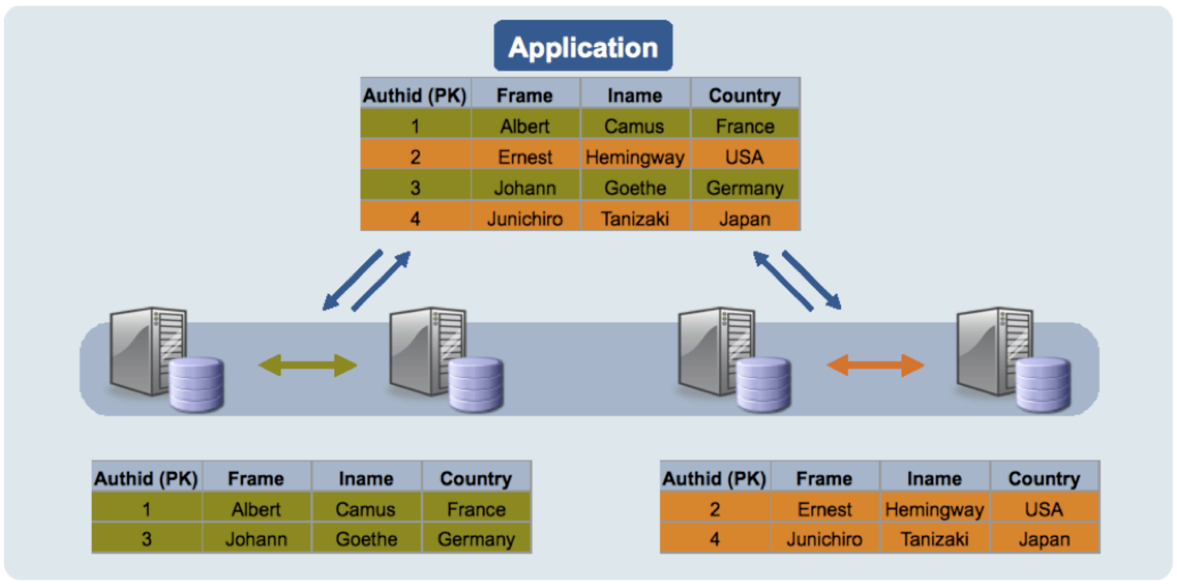
By default, sharding is based on hashing of the primary key, which generally leads to a more even distribution of data and queries across the cluster than alternative approaches such as range partitioning. Developers can also add “distribution awareness” to applications by partitioning based on a sub-key that is common to all rows being accessed by high running transactions. This ensures that such rows are localized on the same shard, thereby reducing network hops.

It is important to note that unlike other distributed databases, users do not lose the ability to perform JOIN operations or sacrifice ACID-guarantees when performing queries and transactions across shards. Adaptive

Query Localization (AQL) pushes JOIN operations down to the data nodes where they are executed locally and in parallel, significantly reducing network hops and delivering 70x higher throughput and lower latency[2].

The end-result is that users can perform complex queries, enabling real-time analytics across live data sets, side-by-side with high throughput OLTP operations. Examples include recommendations engines and clickstream analysis in web applications, pre-pay billing promotions in mobile telecoms networks or fraud detection in payment systems.

From MySQL Cluster 7.3, Foreign Keys are supported, enforcing referential integrity between tables that can be located on different shards, and in even in different data centers.



Authid (PK)	Frame	lname	Country
1	Albert	Camus	France
2	Ernest	Hemingway	USA
3	Johann	Goethe	Germany
4	Junichiro	Tanizaki	Japan

Authid (PK)	Frame	lname	Country
1	Albert	Camus	France
3	Johann	Goethe	Germany

Authid (PK)	Frame	lname	Country
2	Ernest	Hemingway	USA
4	Junichiro	Tanizaki	Japan

Figure 2: Auto-Sharding in MySQL Cluster

The figure above demonstrates how MySQL Cluster shards tables across data nodes of the cluster.

From the following figure, you will also see that MySQL Cluster automatically creates "node groups" from the number of replicas and data nodes specified by the user. Updates are synchronously replicated between members of the node group to protect against data loss and enable sub-second failover in the event of a node failure. This is discussed in more detail in the "Scaling with Continuous Availability" section of the Guide.

The following figure shows how MySQL Cluster creates primary and secondary fragments of each shard. The user has configured the cluster to use four physical data nodes with two replicas, and so MySQL Cluster automatically creates two node-groups.

To ensure optimum performance, developers should always define a primary key on their tables; otherwise MySQL Cluster will create a hidden key. Even if the application doesn't have a primary key, users should create an integer column for their table and auto-increment it. You can learn more about performance and tuning optimization of MySQL Cluster from the following whitepaper: http://www.mysql.com/why-mysql/white-papers/guide-to-optimizing-performance-of-the-mysql-cluster/

[2] http://www.mysqlhighavailability.com/mysql-cluster/70x-faster-joins-with-aql-in-mysql-cluster-7-2/

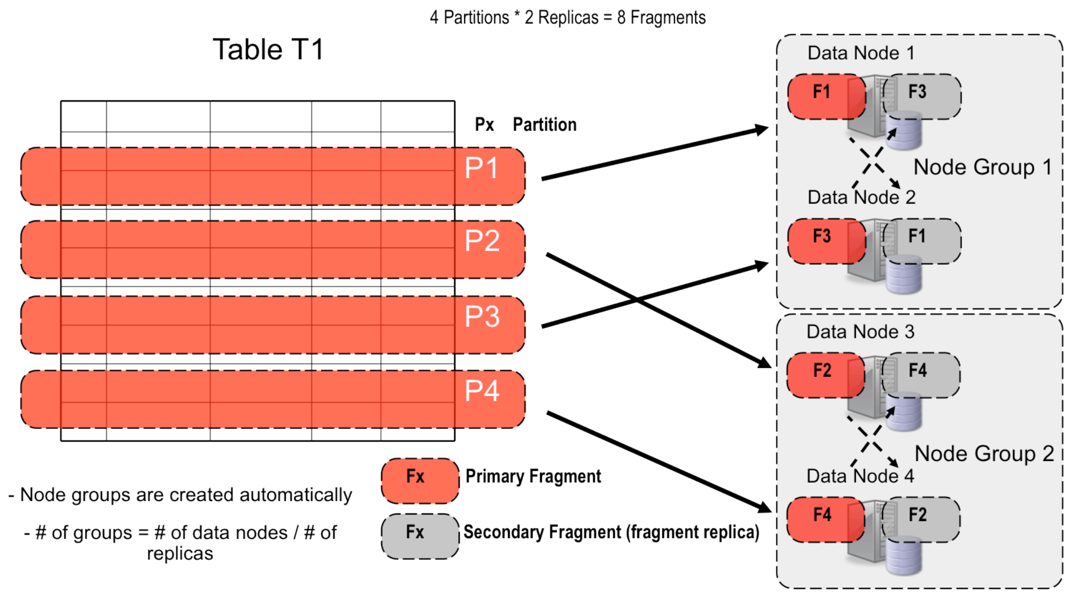

4 Partitions * 2 Replicas = 8 Fragments
Table T1
Px Partition
P1
P2
P3
P4
- Node groups are created automatically
- # of groups = # of data nodes / # of replicas
Fx Primary Fragment
Fx Secondary Fragment (fragment replica)
Data Node 1
F1
F3
Node Group 1
Data Node 2
F3
F1
Data Node 3
F2
F4
Node Group 2
Data Node 4
F4
F2

Figure 3: Automatic Creation of Node Groups & Replicas

Scaling Across Data Centers

Web and mobile services have global reach and so developers will want to ensure their databases can scale-out across regions. MySQL Cluster offers Geographic Replication, which distributes clusters to remote data centers, serving to reduce the affects of geographic latency by pushing data closer to the user, as well as providing for disaster recovery.

MySQL Cluster provides Geographic Replication, allowing the same data to be accessed in clusters located in data centers separated by arbitrary distances. This reduces the effects of geographic latency by pushing data closer to the user, as well as providing a capability for geographic redundancy and disaster recovery.

Geographic replication has always been designed around an Active / Active technology, so if applications are attempting to update the same row on different clusters at the same time, the conflict can be detected and resolved. This ensures each site can actively serve read and write requests while maintaining data consistency across the clusters. It also eliminated the overhead of having to provision and run passive hardware at remote sites.

If replicating between a single pair of clusters then Active-Active (update anywhere) replication has become significantly simpler and more complete in in recent releases, culminating in the final MySQL Cluster 7.4 solution with these added advantages:

- Developers need to make no changes to the application logic or tables
- Conflict-triggered rollbacks can be made to whole transactions rather than just individual operations
- Transactions that are dependent on rolled-back transactions can also be rolled back
- Conflicts involving reads, writes and deletes can all be detected and resolved
- Information on detected conflict is recorded for users information or action

Geographic Replication can also be used as a mechanism to replicate data in real-time to other MySQL storage engines. A typical use-case is to replicate tables from MySQL Cluster to the InnoDB storage engine in order serve as an active archive or to generate complex reports from real-time data, with full performance isolation from the live data store.

MySQL Cluster also offers Multi-Site Clustering, providing additional options for cross data center scalability and fault tolerance. Data nodes can be split across data centers with synchronous replication between them. Failovers between sites are handled automatically by MySQL Cluster. This deployment model works best for data centers that are within the same geographic region and connected by high quality WAN links.

Whether you are replicating within the cluster, across data centers or between storage engines, all replication activities are performed concurrently – so users can combine scaling and High Availability (HA) strategies.

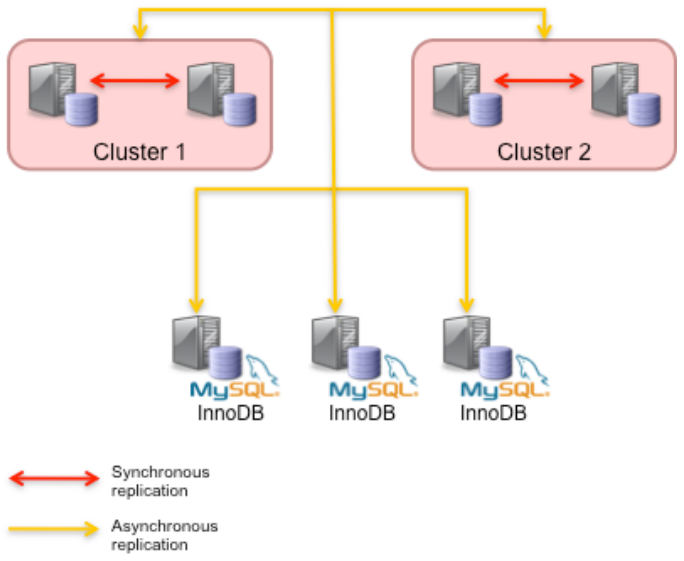


Figure 4: Scaling Across Data Centers with Geographic Replication

Delivering a Real-Time User Experience

MySQL Cluster was originally designed for real-time applications and so kept all data in memory, with continuous redo logging and check-pointing to disk for durability. Support for storing tables on disk was added in 2007, which increased the size of data sets that could be managed by MySQL Cluster far beyond the combined memory of all data nodes.

The original real-time design characteristics remain a central part of MySQL Cluster including:

- Data structures optimized for in-memory access, rather than disk-based blocks
- Persistence of updates to logs and check-points on disk as a background, asynchronous process, therefore eliminating the I/O overhead that can throttle other databases
- Binding of threads to CPUs to avoid CPU sleep-cycles
- Storing all indexed columns in memory for low latency access.

As a result, MySQL Cluster is able to deliver consistenly low levels of latency as users interact with the web service in real time, for example delivering status updates as soon as they occur, without duplicates or reappearances.

Users also have the flexibility of tuning the level of durability by de-configuring check-pointing and logging. Tables stored in this way would still survive a node failure (as they are synchronously replicated to other nodes in the node group), but not a failure of the entire cluster – for example a power failure – unless they were geographically replicated to a remote site. This mode of operation is commonly used in session management applications.

Response times are highly application and environment-dependent, but in OLTP tests[3] using the TPC-C like DBT2 test suite, MySQL Cluster delivered an average latency of just 3 millisecond across SQL queries – and would have been faster if using one of the NoSQL access methods which will be discussed in a following section of the Guide.

Benchmarking Performance on Commodity Hardware

All of the technologies discussed above are designed to provide read and write scalability for your most demanding transactional web applications – but what do they mean in terms of delivered performance?

The MySQL Cluster development team ran a series of benchmarks[4] with Intel that characterized performance across 30 commodity dual socket (2.6GHz), 8-core Intel servers, each equipped with 64GB of RAM, running Linux and connected via Infiniband.

As seen in the figure below, MySQL Cluster delivered over 1 Billion write operations per minute across the cluster, with 650k UPDATEs per second, from each node. Synchronous replication between node groups was configured, enabling both high performance and high availability – without compromise.

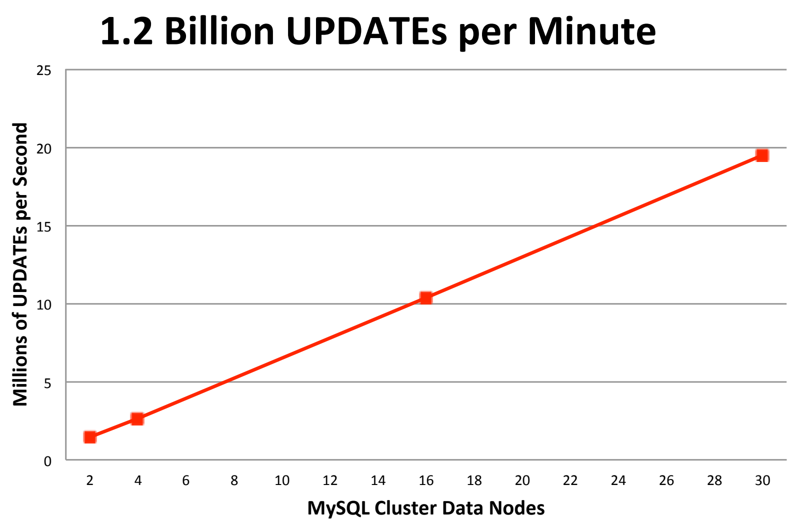


Figure 5: MySQL Cluster performance scaling-out on commodity nodes.

Download the whitepaper to learn more about these benchmarks:
http://www.mysql.com/why-mysql/white-papers/mysql-cluster-benchmarks-1-billion-writes-per-minute/

The DBT2 Benchmark[5] has been used to assess how well SQL performance scales as more data nodes are added.

As can be in Figure 6 the scaling of SQL reads is almost linear and with 16 data nodes, a throughput of 2.5 Million SQL operations per second is achieved. This equates to around 5 Million Transactions Per Minutes or 2.2 Million NewOnly TPM.

[3] http://www.mysqlhighavailability.com/mysql-cluster/mysql-cluster-database-7-performance-benchmark/
[4] http://mysql.com/why-mysql/benchmarks/mysql-cluster/
[5] http://dev.mysql.com/downloads/benchmarks.html

This benchmark was performed with each data node running on a dedicated 56 thread Intel E5-2697 v3 (Haswell) machine.

Benchmarking Scaling NoSQL Performance
The flexAsynch[4] benchmark has been used to measure how NoSQL performance scales as more data nodes are added to the cluster. These tests were performed on the same hardware as the DBT2 benchmark above but scaled out to 32 data nodes (out of a maximum supported 48).

The results are shown in Figure 7 and again it can be observed that the scaling is virtually linear. At 32 data nodes, the throughput hits 200 Million NoSQL Queries Per Second.

These results demonstrate how users can build a highly performing, highly scalable MySQL Cluster from low-cost commodity hardware to power their most mission-critical web services.

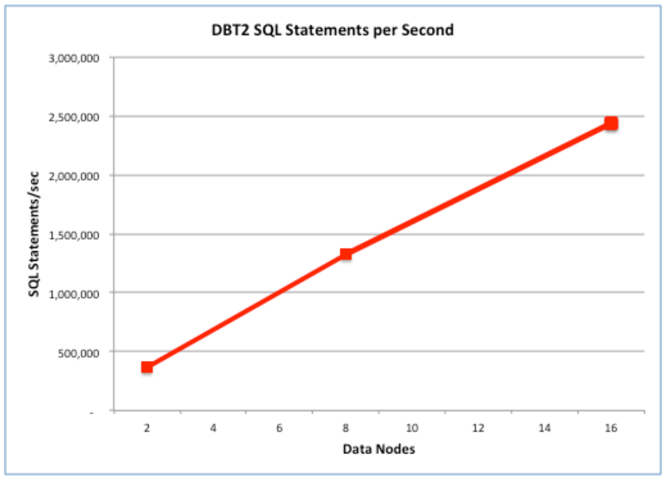


Figure 6 Scaling SQL Reads to 2.5 Million Reads/Second

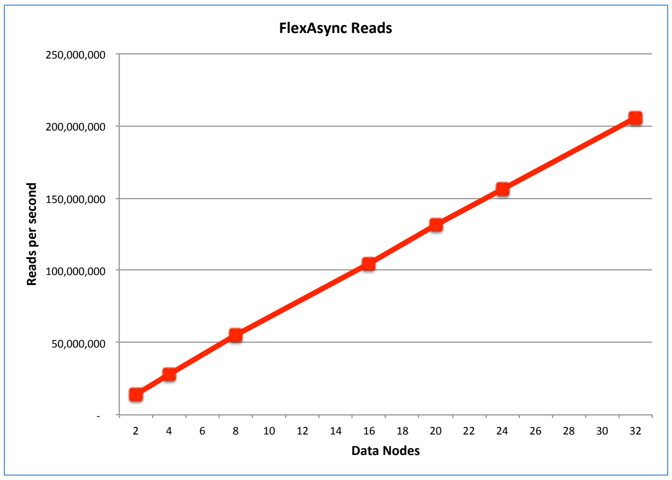


Figure 7 200 Million NoSQL Reads/Second

Scaling Operational Agility

Scaling performance is just one dimension – albeit a very important one – of scaling a database for web and mobile applications. As the new service grows in popularity it is important to be able to evolve the underlying infrastructure seamlessly, without incurring downtime and without having to add lots of additional DBA or developer resource.

Users may need to increase the capacity and performance of the database; enhance their application (and therefore their database schema) to deliver new capabilities and upgrade their underlying platforms.

MySQL Cluster can perform all of these operations and more on-line – without interrupting service to the application or clients.

On-Line, On-Demand Scaling

MySQL Cluster allows users to scale both database performance and capacity by adding Application and Data Nodes on-line, enabling users to start with small clusters and then scale them on-demand, without downtime, as a service grows. Scaling could be the result of more users, new application functionality or more applications needing to share the database.

In the following example, the cluster on the left is configured with two application and data nodes and a single management server. As the service grows, the users are able to scale the database and add management redundancy – all of which can be performed as an online operation. An added advantage of scaling the Application Nodes is that they provide elasticity, so can be scaled back down if load on the database decreases, making MySQL Cluster ideally suited to deployments in the cloud.

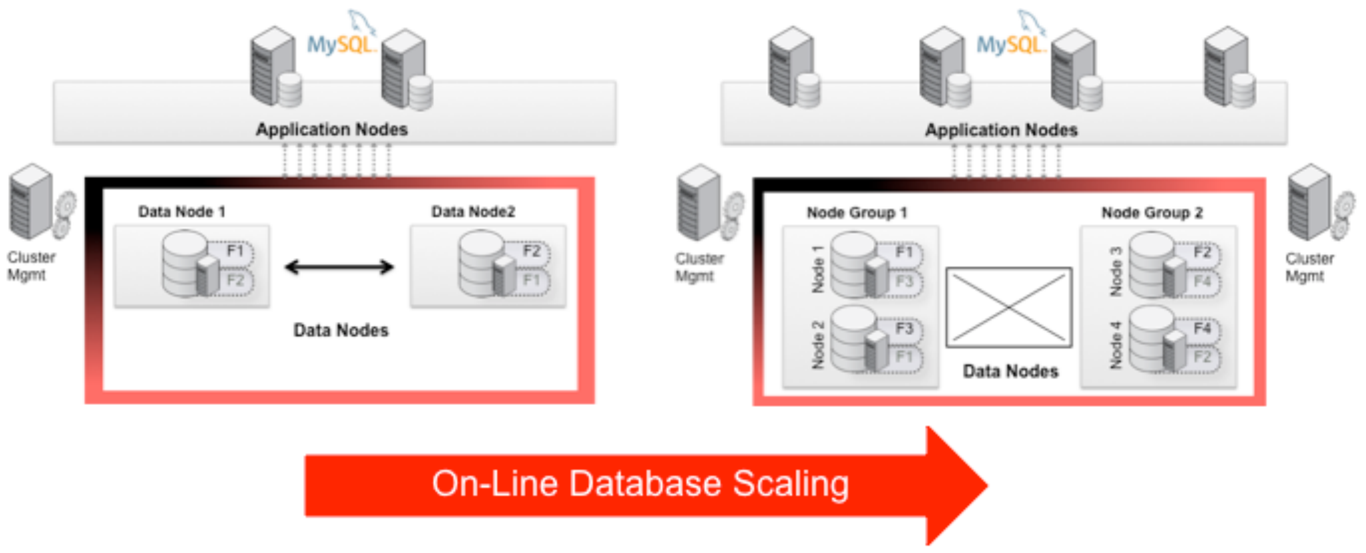


Figure 8: Doubling Database Capacity & Performance On-Demand and On-Line

When new data nodes and node groups are added, the existing nodes in the cluster initiate a rolling re-start to reconfigure for the new resource. This rolling restart ensures that the cluster remains operational during the addition of new nodes. Tables are then repartitioned and redundant rows are deleted with the OPTIMIZE TABLE command. All of these operations are transactional, ensuring that a node failure during the add-node process will not corrupt the database.

The operations can be performed manually from the command line or automated with MySQL Cluster Manager[6], part of the commercial MySQL Cluster CGE edition discussed later.

On-Line Cluster Maintenance

Rolling restarts can also be used to upgrade nodes within the cluster:

- Upgrade or patch the underlying hardware and operating system
- Upgrade or patch MySQL Cluster, with full online upgrades between releases.

MySQL Cluster supports on-line, non-blocking backups to ensure service interruptions are again avoided. Users are able to exercise fine-grained control when restoring a MySQL Cluster from a backup. Users can restore only specified tables or databases, or exclude specific tables or databases from being restored, using ndb_restore options --include-tables, --include-databases, --exclude-tables, and --exclude-databases.

On-Line Schema Evolution

As services evolve, developers often want to add new functionality, which in turn may demand updating the database schema.

This operation can be very disruptive for many databases, with ALTER TABLE commands taking the database offline for the duration of the operation. When users have large tables with many millions of rows, downtime can stretch into hours or even days.

MySQL Cluster supports on-line schema changes, enabling users to add new columns and tables and add and remove indexes or Foreign Keys – all while continuing to serve read and write requests, and without affecting response times.

[6] http://mysql.com/products/cluster/mcm/

Unlike other approaches to on-line schema updates, MySQL Cluster does not need to create temporary tables, therefore avoiding the user having to provision double the usual memory or disk space in order to complete the operation.

Scaling Database Access: SQL & NoSQL

As MySQL Cluster stores tables in data nodes, rather than in the MySQL Server, there are multiple interfaces available to access the database.

Developers have a choice between:

- SQL for complex queries and access to a rich ecosystem of applications and expertise
- Simple Key/Value and Key/Object interfaces bypassing the SQL layer for blazing fast reads & writes
- Real-time interfaces for microsecond latency.

With this choice of interfaces, developers are free to work in their own preferred environments, enhancing productivity and agility, enabling them to deliver new services to market faster.

SQL or NoSQL: Selecting the Right Interface

The following chart shows all of the access methods available to the database. The native API for MySQL Cluster is the C++ based NDB API. All other interfaces access the data through the NDB API.

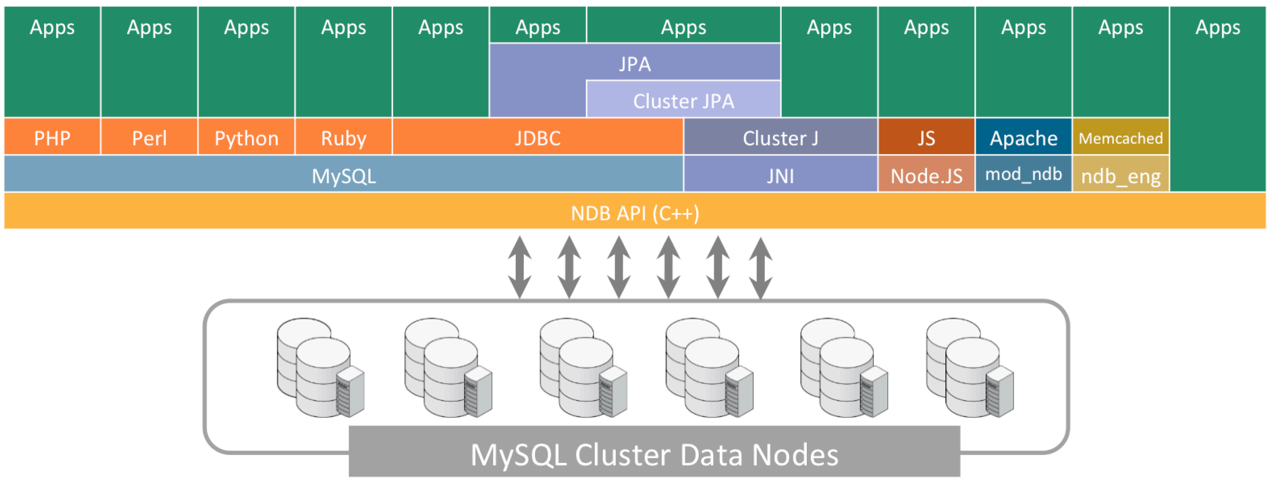


Figure 9: Developer Flexibility – MySQL Cluster SQL & NoSQL APIs

At the extreme right hand side of the chart, an application has embedded the NDB API library enabling it to make native C++ calls to the database, and therefore delivering the lowest possible latency.

On the extreme left hand side of the chart, MySQL presents a standard SQL interface to the data nodes, and provides connectivity to all of the standard MySQL connectors including:

- Common web development languages and frameworks, i.e. PHP, Perl, Python, Ruby, Ruby on Rails, Spring, Django, etc;
- JDBC (for additional connectivity into ORMs including EclipseLink, Hibernate, etc)
- .NET
- ODBC

Whichever API is chosen for an application, it is important to emphasize that all of these SQL and NoSQL access methods can be used simultaneously, across the same data set, to provide the ultimate in developer flexibility. Therefore, MySQL Cluster maybe supporting any combination of the following services, in real-time:

- Relational queries using the SQL API
- Key/Value or Key/Object-based web services using the JavaScript, REST/HTTP and memcached APIs
- Enterprise applications with the ClusterJ and JPA APIs
- Server-side JavaScript with Node.
- Real-time mobile services (i.e. presence and location based) using the NDB API.

Schema-less Data Store with memcached API

Like memcached, MySQL Cluster provides a distributed hash table with in-memory performance for caching. MySQL Cluster extends memcached functionality by adding support for write-intensive workloads, a full relational model with ACID compliance (including persistence), rich query support, auto-sharding and 99.999% availability, with extensive management and monitoring capabilities.

Using the memcached API, users can simplify their architecture by compressing the caching and database layers into a single data tier, managed by MySQL Cluster. All writes are committed directly to MySQL Cluster, eliminating cache invalidation and the overhead of data consistency checking to ensure complete synchronization between the database and cache. Duplication of data between the cache and a persistent database can be eliminated, enabling simpler re-use of data across multiple applications, and reducing memory footprint.

By default, every Key / Value is written to the same table with each Key / Value pair stored in a single row – thus allowing schema-less data storage. Alternatively, the developer can define a key-prefix so that each value is linked to a pre-defined column in a specific table.

Of course if the application needs to access the same data through SQL then developers can map key prefixes to existing table columns, enabling Memcached access to schema-structured data already stored in MySQL Cluster.

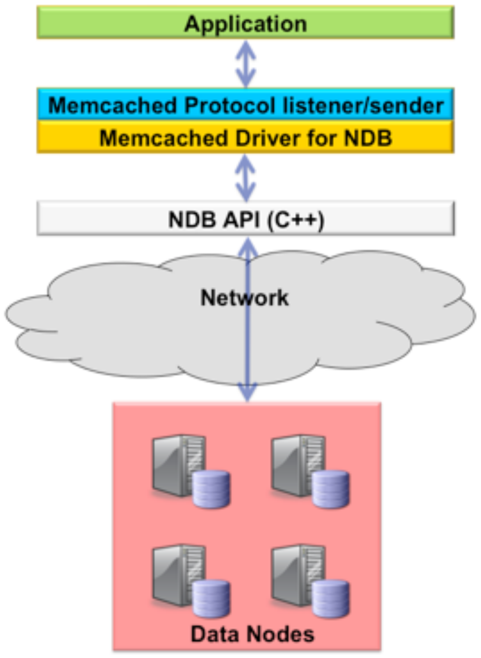


Figure 10: Memcached API Implementation with MySQL Cluster

Scaling with Continuous Availability

No matter how well a web service is architected for scale, if it is not up and running, it will never be successful.

Data and transactional states are usually the hardest parts of a web service to make highly available. Implementing a database that is itself highly available makes it simpler for the application to become highly available as well. This approach permits delegating the complexity of data management and transactional states to the database layer. The clear advantage of this design is that the database will always be the most competent, efficient and reliable mechanism in handling these duties when compared to other components of the system.

The architecture of MySQL Cluster is designed to deliver 99.999% availability, which includes both regularly scheduled maintenance operations, as well as systems failures (i.e. “planned” and “unplanned” downtime). The Guide has already discussed many of the capabilities within MySQL Cluster to deliver high availability, and in this section, we explore them in more detail.

Resilience to Failures

The distributed, shared-nothing architecture of MySQL Cluster has been carefully designed to ensure resilience to failures, with self-healing recovery:

- MySQL Cluster detects any failures instantly and control is automatically failed over to other active nodes in the cluster, without interrupting service to the clients
- In the event of a failure, the MySQL Cluster nodes are able to self-heal by automatically restarting and resynchronizing with the rest of the cluster, all of which is completely transparent to the application
- The data within a data node is synchronously replicated to all nodes within the Node Group. If a data node fails, then there is always at least one other data node storing the same information
- In the event of a data node failure, the MySQL Server or application node can use any other data node in the node group to execute transactions. The application simply retries the transaction and the remaining data nodes will successfully satisfy the request.

Designing the cluster in this way makes the system reliable and highly available since single points of failure have been eliminated. Any node can be lost without it affecting the system as a whole. As illustrated in the example below, an application can continue executing even though multiple Data Nodes are down, provided that there is one surviving node in each node group. Techniques used to increase the reliability and availability of the database include:

- Data is synchronously replicated between all data nodes in the node group. This leads to very low fail-over times in case of node failures as there is no need to recreate and replay log files for the application to fail over
- Nodes can be distributed across multiple hosts or data centers, allowing MySQL Cluster to operate even during hardware or region failures
- With its shared-nothing architecture, each data node has its own disk and memory resource, so a failure in shared storage does not cause an outage of the cluster.

As demonstrated in the figure below, MySQL Cluster continues to deliver service, even in the event of catastrophic failures.

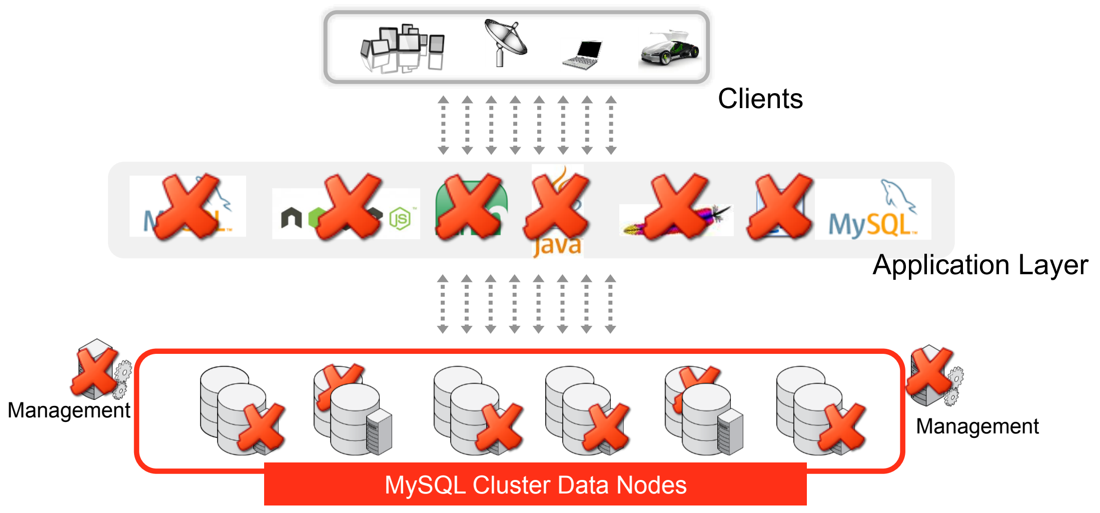


Figure 11: With no single point of failure, MySQL Cluster delivers extreme resilience to failure

In addition to the site-level high-availability achieved through the redundant architecture of MySQL Cluster, geographic redundancy can be achieved using replication between two or more Clusters, or splitting data nodes of a single cluster across sites. More information on options geographic redundancy can be found in the "Scaling Across Data Centers" section of this Guide.

Eliminating Planned Downtime

Building on the shared nothing architecture discussed above, users are able to perform maintenance tasks to MySQL Cluster without having to bring the database down. These have already been discussed in the "Scaling Operational Agility" section of the guide, and are summarized below. All of these are on-line operations, meaning that query and insert, update and delete transactions continue to be processed throughout:

- Scale the cluster by adding new data, application and management nodes
- Update the schema with new columns, tables and indexes
- Re-sharding of tables across data nodes to allow better data distribution
- Upgrade or patch the underlying hardware and operating system
- Upgrade or patch MySQL Cluster, with full online upgrades between releases.

Through the capabilities described above, MySQL Cluster is able to eliminate both planned maintenance and unplanned downtime in order to deliver 99.999% availability demanded by web-based applications.

MySQL Web Reference Architectures

MySQL Cluster is a key component of the MySQL Web Reference Architectures – a collection of repeatable best practices for building highly scalable and available web and mobile services, developed with the world's leading web properties.

The Reference Architectures profile four components common to most web properties and define the optimum deployment architectures for each:

- User authentication and session management
- Content management
- Ecommerce
- Analytics and big data integration

The reference architectures are categorized by "small", "medium", "large" and "extra large", based on sizing and availability requirements appropriate to each environment.

				Social Network
	Small	**Medium**	**Large**	**Extra Large**
Queries/Second	**<500**	**<5,000**	**10,000+**	**25,000+**
Transactions/Second	**<100**	**<1,000**	**10,000+**	**25,000+**
Concurrent Read Users	**<100**	**<5,000**	**10,000+**	**25,000+**
Concurrent Write Users	**<10**	**<100**	**1,000+**	**2,500+**
Database Size				
Sessions	**<2 GB**	**<10 GB**	**20+ GB**	**40+ GB**
eCommerce	**<2 GB**	**<50 GB**	**50+ GB**	**200+ GB**
Analytics (Multi-Structured Data)	**<10 GB**	**<1TB**	**10+ TB**	**100+ TB**
Content Management (Meta-Data)	**<10 GB**	**<500 GB**	**1+ TB**	**2+ TB**

Figure 12: MySQL Web Reference Architecture Sizing

The Large Web Reference Architecture is shown below:

- MySQL Cluster powers the Session Management and eCommerce components
- MySQL 5.7 with InnoDB and a caching layer are deployed for content management
- Data is replicated to a Hadoop cluster for aggregation and processing with unstructured data before then being loaded into an Analytics application powered by a MySQL database.

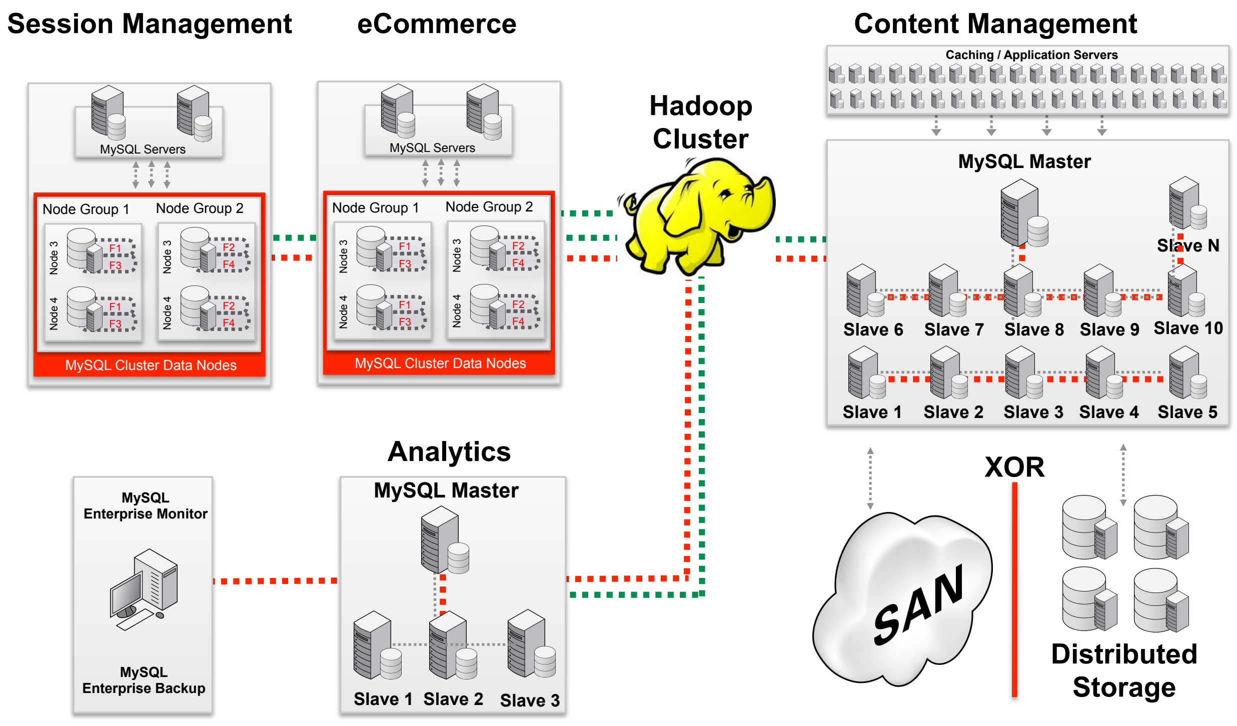


Figure 13: MySQL Cluster powering Session Management and eCommerce

With 2 x data nodes, MySQL Cluster is supporting 6,000 sessions (page hits) a second, with each page hit generating 8 – 12 database operations.

By using the scale-out capabilities of MySQL Cluster, it is possible to combine the session management and ecommerce databases onto one larger cluster.

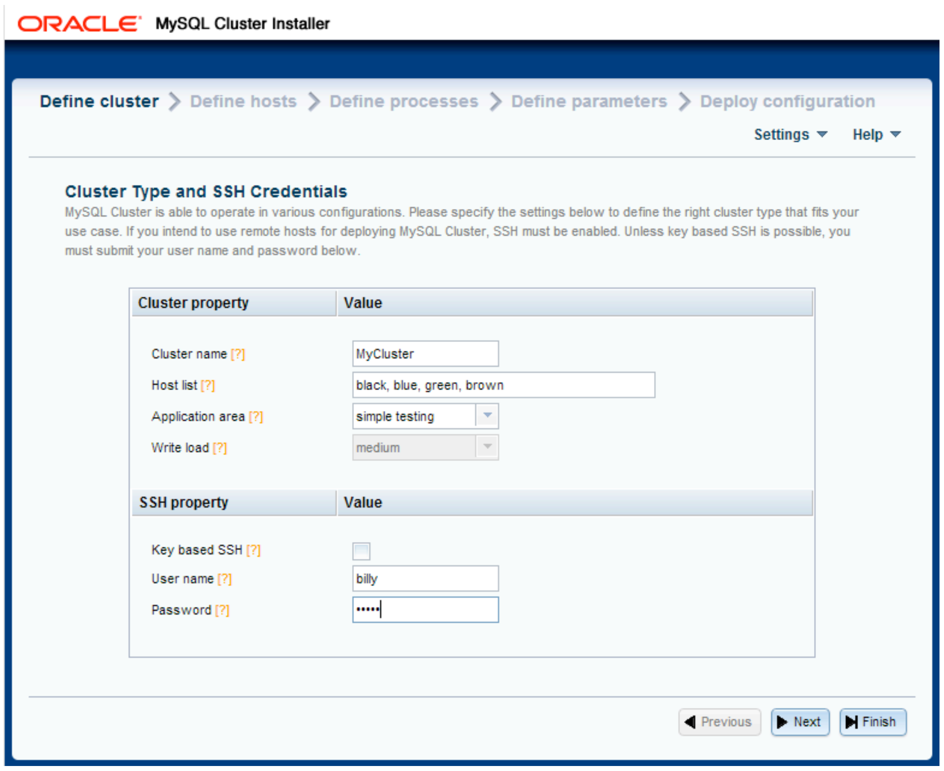
Master / Slave replication is used to scale-out the Content Management and Analytics components of the web property. The content management architecture would be able to scale to 100k+ concurrent users with around 20 slaves – dependent on traffic patterns of the workload.

Provisioning MySQL Cluster

MySQL Cluster's GUI-based Auto-Installer makes it simple for DevOps teams to quickly configure and provision highly optimized clusters - whether on-premise or in the cloud, stepping users through each stage of cluster creation:

- Workload Optimized: On launching the browser-based installer, users specify the throughput, latency and write-load characteristics of their application
- Auto-Discovery: The Installer automatically discovers the underlying hardware resources from each server that will make up the cluster.

With these parameters, the installer creates optimized configuration files and starts the cluster.



Cluster Type and SSH Credentials

MySQL Cluster is able to operate in various configurations. Please specify the settings below to define the right cluster type that fits your use case. If you intend to use remote hosts for deploying MySQL Cluster, SSH must be enabled. Unless key based SSH is possible, you must submit your user name and password below.

Cluster property	Value
Cluster name [?]	MyCluster
Host list [?]	black, blue, green, brown
Application area [?]	simple testing
Write load [?]	medium

SSH property	Value
Key based SSH [?]	☐
User name [?]	billy
Password [?]	•••••

Figure 14: Automated tuning and configuration of MySQL Cluster

Developed by the same engineering team responsible for the development of the MySQL Cluster database, the installer provides standardized configurations that make it simple, quick and easy to build stable and high performance clustered environments.

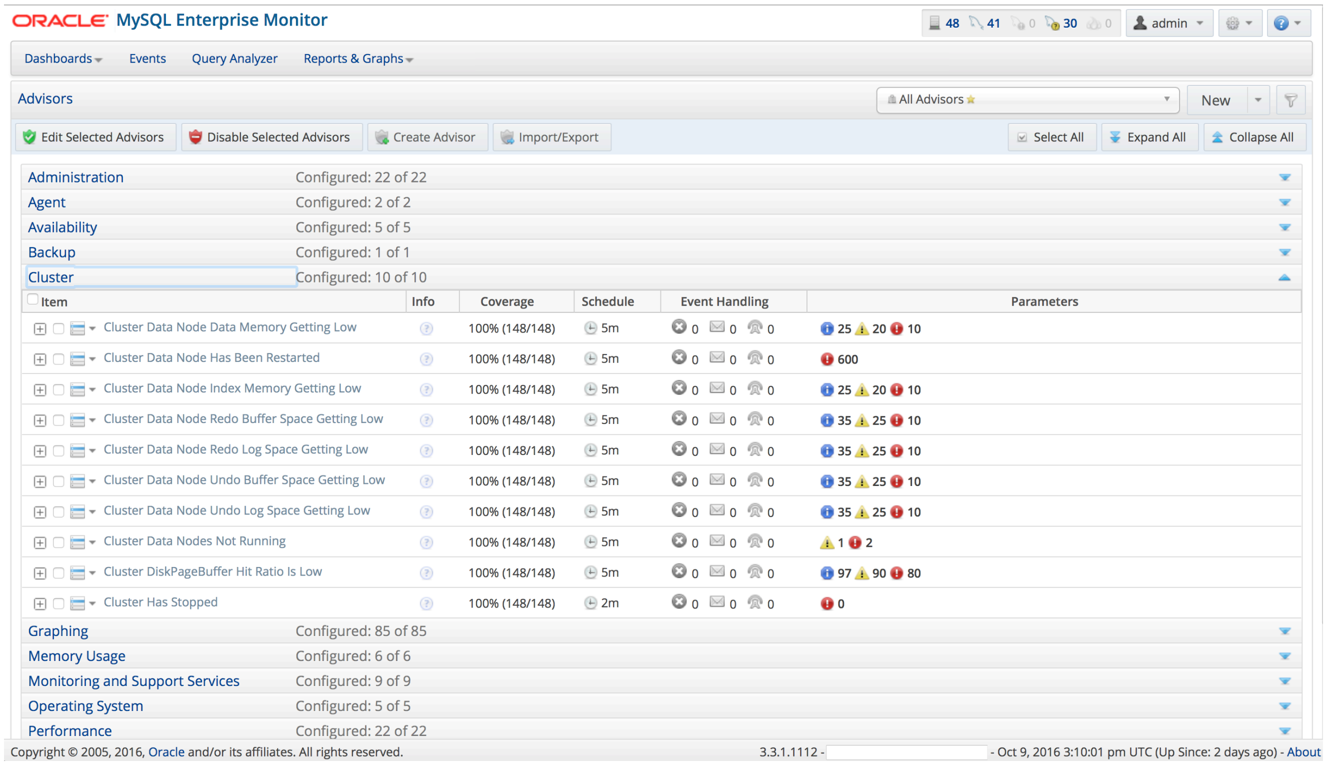
MySQL Cluster Carrier Grade Edition

MySQL Cluster Community Edition is licensed under the GPL and freely downloadable for development and deployment.

MySQL Cluster is also available in the commercial CGE edition, which includes 24x7 Oracle Premier Support and access to an extensive array of security, auditing and management tools.

MySQL Cluster Manager simplifies the provisioning, scaling and reconfiguration of MySQL Cluster by automating common management tasks. DevOps teams are more productive, able to focus on strategic initiatives and respond faster to changing user requirements. At the same time, risks of database downtime, which previously resulted from manual configuration errors, are significantly reduced.

MySQL Enterprise Monitor provides at-a-glance views of the health of your cluster. It continuously monitors the MySQL servers and data nodes, alerting you to potential problems before they impact clients, using a series of Expert Advisors to recommend best practices developed by the engineers who build the MySQL database.



Administration	Configured: 22 of 22
Agent	Configured: 2 of 2
Availability	Configured: 5 of 5
Backup	Configured: 1 of 1
Cluster	Configured: 10 of 10

Item	Info	Coverage	Schedule	Event Handling	Parameters
Cluster Data Node Data Memory Getting Low		100% (148/148)	5m	0 0 0	25 20 10
Cluster Data Node Has Been Restarted		100% (148/148)	5m	0 0 0	600
Cluster Data Node Index Memory Getting Low		100% (148/148)	5m	0 0 0	25 20 10
Cluster Data Node Redo Buffer Space Getting Low		100% (148/148)	5m	0 0 0	35 25 10
Cluster Data Node Redo Log Space Getting Low		100% (148/148)	5m	0 0 0	35 25 10
Cluster Data Node Undo Buffer Space Getting Low		100% (148/148)	5m	0 0 0	35 25 10
Cluster Data Node Undo Log Space Getting Low		100% (148/148)	5m	0 0 0	35 25 10
Cluster Data Nodes Not Running		100% (148/148)	5m	0 0 0	1 2
Cluster DiskPageBuffer Hit Ratio Is Low		100% (148/148)	5m	0 0 0	97 90 80
Cluster Has Stopped		100% (148/148)	2m	0 0 0	0

Graphing	Configured: 85 of 85
Memory Usage	Configured: 6 of 6
Monitoring and Support Services	Configured: 9 of 9
Operating System	Configured: 5 of 5
Performance	Configured: 22 of 22

Copyright © 2005, 2016, Oracle and/or its affiliates. All rights reserved. 3.3.1.1112 - - Oct 9, 2016 3:10:01 pm UTC (Up Since: 2 days ago) - About

Figure 15: MySQL Cluster expert advisors recommend best practices

MySQL Query Analyzer helps developers and DBAs improve complex query performance by accurately pinpointing SQL code that can be optimized. Queries are presented in an aggregated view across all MySQL servers so developers can filter for specific query problems and identify the code that consumes the most resources.

MySQL Enterprise Security provides ready to use external authentication modules to easily integrate MySQL Cluster with existing security infrastructures including LDAP, Linux PAM and Windows Active Directory, ensuring secure access to your most sensitive data.

MySQL Enterprise Audit enables administrators to quickly add policy-based auditing compliance to applications by logging user-level activities; implementing activity-based policies and managing audit log files.

MySQL Thread Pooling - by default MySQL provides a complex thread-handling model that provides excellent throughput and performance for online and web-based applications. User connections are mapped to execution threads on a one-to-one basis with each connection/thread assignment remaining intact until the connection is terminated by the client. Under this model MySQL provides scalable concurrency of both user connections and query executions.

While this model serves and scales most web deployment use cases very well it does have the potential to limit scalability as connection and query loads increase at an accelerating rate. This use case is becoming more common as application clients now extend to mobile and other web-enabled devices. For the most highly-trafficked applications when concurrent connections grow from hundreds to thousands and associated query executions grow proportionally, scalability challenges and limitations with the default model are potentially exposed:

- Current model does not prioritize connection queries for execution, regardless of the number that have been submitted or that are in a “wait” status. No prioritization of queries means that all attempt to execute in parallel with no regard for server resource limitations.
- More concurrency of query executions requires significantly more server memory. In an extreme case if the amount of memory needed by all active connections exceeds server memory, the MySQL server may revert to memory/disk swapping, which will greatly impact user response times.
- More query executions also leads to more cache flushing, which leads to more cache misses and disk I/O requests. More disk I/O leads to longer query execution and user response times.

To meet these challenges around the most demanding application user and workloads MySQL Enterprise Edition and MySQL Cluster CGE provide the MySQL Thread Pool. The Thread Pool is a user configurable option that provides an efficient, alternate thread-handling model designed to sustain performance and scalability as concurrent user loads continue to grow. In these use cases the Thread Pool addresses the limitations to scalability by:

- Managing/controlling query execution until the MySQL server has the resources to execute it.
- Splitting threads into managed Thread Groups. Inbound connections are assigned to a group via a round-robin algorithm and the number of concurrent connections/threads per group is limited based on queue prioritization and nature of queries awaiting execution. Transactional queries are given a higher priority in the queue than non-transactional, but queue prioritization can be overridden at the user level as needed.
- Avoiding deadlocks when queries are stalled or executing for long period of time.

Oracle Premier Support delivers 24x7, global support for MySQL Cluster. The MySQL Support team is composed of seasoned MySQL developers, who are database experts and understand the issues and challenges you face. With Oracle Premier Support, you can more rapidly innovate in the development of new services, lower cost and complexity and optimize the value of your database-driven solutions.
Oracle Premier Support for MySQL includes the following features:

- 24 x 7 global production support in 29 languages
- Direct access to MySQL support engineers, backed by the MySQL developers
- Unlimited support incidents
- Knowledge Base
- Maintenance releases, hot fixes, patches and updates
- MySQL consultative support

Conclusion

To be truly successful, a web service needs to scale in multiple dimensions:

- Performance (throughput and latency
- Operational agility
- Data access interfaces
- Availability

With auto-sharding, active / active geographic replication, online operations, SQL and NoSQL interfaces and 99.999% availability, MySQL Cluster is already serving some of the most demanding web and mobile services on the planet.

This Guide has been designed to provide an overview of these capabilities, and the resources below will enable you to learn more in building out your next successful web service with MySQL Cluster.

Additional Resources

Download MySQL Cluster:
http://dev.mysql.com/downloads/cluster/

On-Line Demonstration - MySQL Cluster in Action:
http://www.youtube.com/watch?v=DnWItDTZL2c

MySQL Cluster Evaluation Guide:
http://www.mysql.com/why-mysql/white-papers/mysql-cluster-evaluation-guide/

MySQL Cluster Manager:
http://mysql.com/products/cluster/mcm/

MySQL Cluster Customer Success:
http://mysql.com/customers/cluster/

Contact MySQL Sales:
https://mysql.com/about/contact/sales.html