



MySQL Reference Architectures for Massively Scalable Web Infrastructure

Best Practices for Innovating on the Web & in the Cloud

A MySQL[®] Strategy White Paper

March 2013



Table of Contents

Executive Summary	3
Small Web Reference Architecture	4
Sizing & Topology	4
InnoDB	5
Scaling the “Small” Web Reference Architecture	5
Protecting Users’ Data on the Web	5
Data Backups	6
Medium Web Reference Architecture	7
Sizing & Topology	7
MySQL Replication: Scaling and High Availability	9
Managing MySQL Replication	11
OS-Level Clustering	11
Scaling User Connections	12
Large Web Reference Architecture	14
Hadoop Cluster for Big Data	15
MySQL Cluster for Session Management and Ecommerce	16
Social Networking (Extra Large) Reference Architecture	17
Sizing & Topology	17
Sharding	18
NoSQL APIs	19
Schema Flexibility	19
The Perfect Server for MySQL	20
MySQL Cluster	20
Operational Best Practices	22
Management & Monitoring of your MySQL Server Estate	23
Conclusion	24
Additional Resources	25



Executive Summary

Organizations around the world are constantly seeking new ways to leverage the power of the web, mobile and cloud to drive their business. They recognize their ability to compete depends on optimizing their use of technology and leveraging best practices; whether their aim is to help their customers communicate, socialize and engage, create and share information, entertain, or shop for products and services.

Success or failure of new web or mobile initiatives is highly dependent on selecting the right technologies from the start, eliminating extensive trial and error that may result in missed market opportunities.

Key design goals for the architecture supporting new web and mobile services include:

- **Fast response times and continuous availability** for exceptional user-experience
- **Simplicity in design** to achieve fast time to market
- **Seamless scalability** to meet rapidly growing demands
- **Highly adaptable and agile** to support rapidly evolving service requirements
- **Cloud and DevOps friendly** to automate deployment and management
- **Open and extensible** to support integration with big data and analytical systems
- **Rock-solid security** to protect customer data
- **Low total cost of ownership** for fast ROI

MySQL is deployed in 9 of the top 10 most trafficked sites on the web¹ including Facebook, Twitter, Google, YouTube and Wikipedia. Many of the fastest growing properties rely on MySQL to support their exceptional scaling demands, including Tumblr, Pinterest and box.com. MySQL is also the most deployed database in public and private Clouds.

This gives MySQL unique insight into how to design database-driven web architectures – whether deployed on-premise or in the cloud - that deliver the highest levels of scalability, agility and availability with the lowest levels of cost, risk and complexity.

In this whitepaper, we present four Reference Architectures based on best practices developed from working with the most successful web properties on the planet.

We have selected the four components common to most web and mobile properties, and defined 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.

The whitepaper concludes with recommendations on the server and storage configurations needed to support the reference architectures.

¹ <http://www.alexa.com/topsites>

² <http://www.mysql.com/products/enterprise/audit.html>



Small Web Reference Architecture

Sizing & Topology

The “Small” Web reference architecture is defined by the sizing below:

	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 1: Sizing for Small Web Reference Architecture

The figure below shows the recommended topology to support the “Small” workload.

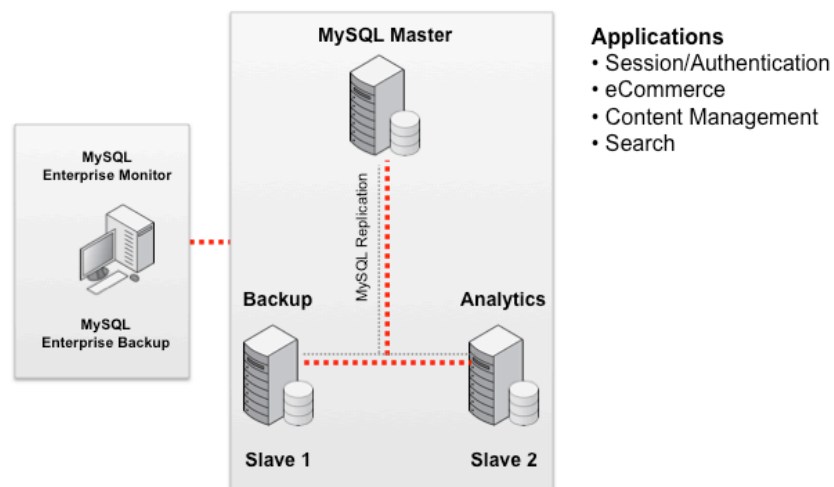


Figure 2: Topology for Small Web Reference Architecture

In this topology, a single MySQL master server is deployed to support all applications, including session management, ecommerce, content management and search.

To ensure the MySQL master is able to dedicate resources to serving the web applications, MySQL’s native replication is used to scale the database across two slaves; one handling backup, the other handling analytics. MySQL replication is discussed in more detail later in the Guide.

(Note: MySQL Enterprise Backup can perform online "Hot", non-blocking backups of MySQL databases. Full backups can be performed on all InnoDB data, while MySQL is online, without interrupting queries or updates, thereby eliminating the need to provision a dedicated slave).



InnoDB

MySQL's default InnoDB storage engine is suitable for supporting all of the applications that are part of the Small Web Reference Architecture.

InnoDB is fully ACID-compliant, offering fast, reliable recovery from crashes with zero committed data loss. InnoDB supports highly concurrent applications with row level locking and MVCC (Multi-Version Concurrency Control) and scaling on commodity systems with 48 and more threads. InnoDB includes support for Foreign Keys, Full-Text Search, Online DDL (schema changes), Transportable databases and a NoSQL interface via the Memcached API for high velocity data ingestion and low latency read operations.

New backup and recovery of the InnoDB bufferpool on a restart in MySQL 5.6 enables quicker spinning up of new instances when scaling applications and recovery of failed instances within web and cloud environments. The Performance_Schema provides detailed instrumentation of the database, enabling developers to optimize queries and configurations, in addition to detailed statistics capturing resource consumption by users and applications.

Scaling the “Small” Web Reference Architecture

Common session management techniques like browser and web server-based solutions will satisfy requirements if traffic is low and the application does not need to manage large or complex session data. However as application traffic increases, so do the sessions that will need to be supported on the server. Coupled with the size and complexity of the data, scalability and performance issues may also quickly arise.

Using MySQL to store session variables versus traditional session management solutions can result in better overall performance. Therefore, if the service grows, it is recommended that the Session Management application be managed by MySQL and migrated to its own dedicated MySQL database.

As the load from the web service grows, allocating memory and tuning for each application on shared hardware resources becomes increasingly complex. It is therefore recommended that the Small Web Reference Architecture should only be selected when load is light and the user expects limited growth to their service or application.

For the reasons above, if high growth is anticipated, it is recommended users start with the “Medium” configuration, presented below, which provides greater capacity, more flexibility for business change and improved availability.

Protecting Users’ Data on the Web

Data security on the web is critical and increasingly mandated – no matter the size of the user-base for your application.

PCI compliance guidelines ensure credit card data is secure within e-commerce apps. From a corporate standpoint, Sarbanes-Oxley, HIPAA, etc. guard medical, financial, public sector and other personal data with required logging, archiving and "upon request" access to audit trails that reveal the eyes and hands that have viewed and acted upon the most sensitive of data. In all use cases, requirements for capturing application level user activity are most commonly implemented on the back-end database.

To meet these requirements, MySQL provides an open pluggable audit interface that enables all MySQL users to write their own auditing solutions based on application-specific requirements. To help users quickly and seamlessly add auditing compliance to new and existing services MySQL Enterprise Edition includes MySQL Enterprise Audit² an easy to use policy-based auditing solution that enables users to:

² <http://www.mysql.com/products/enterprise/audit.html>



- Dynamically enable/disable the audit stream;
- Implement policies that capture login or query based activities;
- Automatically rotate audit log files based on size;
- Integrate XML-based audit log stream with MySQL, Oracle and other third party solutions.

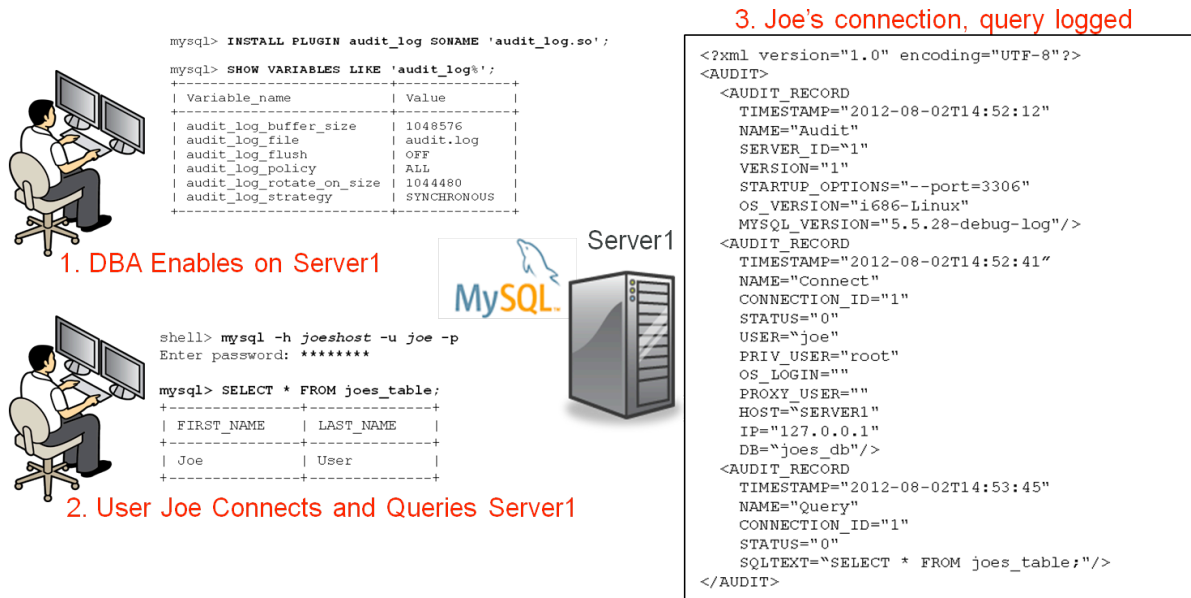


Figure 3: MySQL Enterprise Audit Set Up and Use Case

In addition to auditing, MySQL also supports an open, pluggable authentication interface that enables users to develop plug-ins to authenticate MySQL client connections against external resource such as LDAP, Linux PAM, Windows Active Directory, etc. This enables MySQL to easily integrate with existing security standards and infrastructure that have been established to protect data from your web and mobile applications. It also simplifies later scaling of the architecture as new servers automatically leverage authentication data in the centralized directory.

As with auditing, MySQL Enterprise Edition provides ready to use external authentication modules³ for applications that authenticate users via Pluggable Authentication Modules ("PAM") or native Windows OS services. The modules are developed and fully supported by Oracle.

You can learn more about MySQL Enterprise Edition in the Operational Best Practices section of this Guide.

Data Backups

Implementing proper database backup and disaster recovery plans to protect against accidental loss of data, database corruption, hardware/operating system crashes or any natural disasters is one of the most important responsibilities of the Database Administrator (DBAs).

Unfortunately, if you don't have an adequate database backup and recovery strategy and implementation in place then you are left with nothing to fall back on. As a DBA you need to make sure that the databases are backed up regularly and timely recovery procedures are in place.

³ <http://www.mysql.com/products/enterprise/security.html>



MySQL is distributed with the mysqldump client backup tool. Alternatively MySQL Enterprise Backup provides DBAs with a higher performance, online “hot” backup solution with data compression technology.

MySQL Enterprise Backup performs online "Hot", non-blocking backups of MySQL databases. Full backups can be performed on all InnoDB data while MySQL is online, without interrupting queries or updates. In addition, incremental backups are supported so that only data that has changed from a previous backup are captured. Also partial backups are supported when only certain tables or tablespaces need to be backed up.

MySQL Enterprise Backup restores data from a full backup with full backward compatibility. Consistent Point-in-Time Recovery (PITR) enables restoration to a specific point in time. Using MySQL backups and the Binary Log (binlog), you can also perform fine-grained roll forward recovery to a specific transaction. A partial restore allows recovery of targeted tables or tablespaces. In addition, you can restore backups to a separate location, or create clones for fast replication setup or administration.

MySQL Enterprise Backup supports creating compressed backup files, typically reducing backup size from 70% to over 90% when compared to the size of actual database files, reducing storage and other costs.

MySQL Enterprise Backup is provided as part of MySQL Enterprise Edition. You can learn more here: <http://www.mysql.com/products/enterprise/backup.html>

Medium Web Reference Architecture

Sizing & Topology

The “Medium” web reference architecture is defined by the sizing below:

				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 4: Sizing for Medium Web Reference Architecture

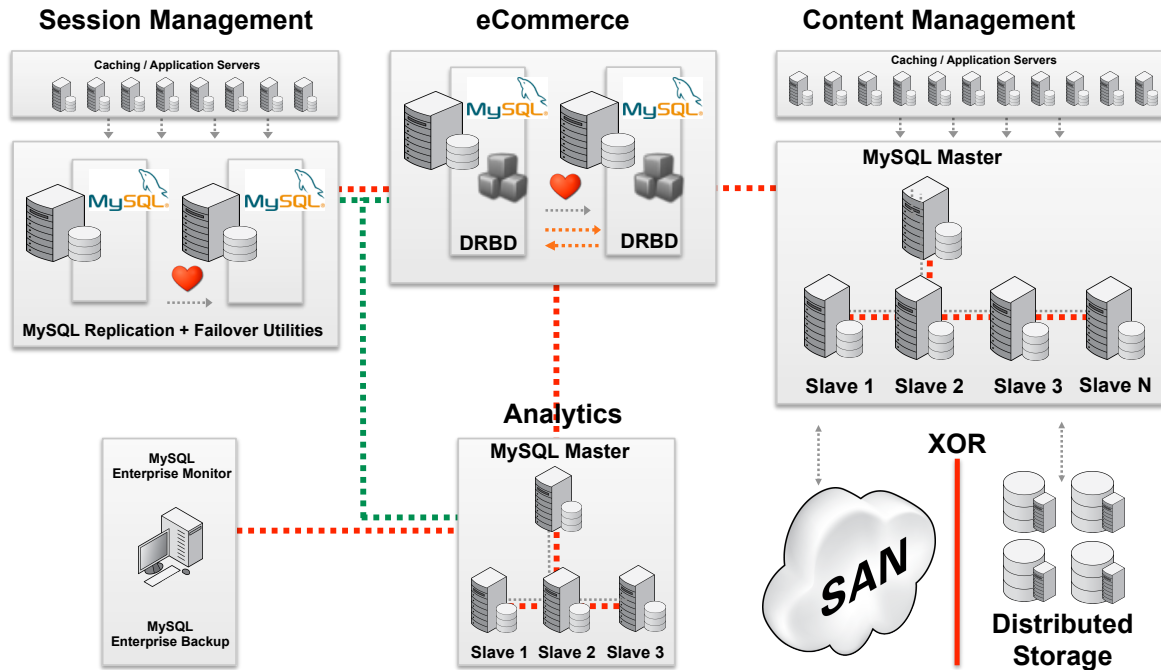


Figure 5: Topology for Medium Web Reference Architecture

Unlike the Small Web Reference Architecture described earlier, the Medium Architecture distributes the core functions of Session Management, Ecommerce, Content Management and Analytics across their own dedicated MySQL databases and hardware infrastructure, enabling each application to be deployed, managed and scaled independently.

If a user expects their web service to scale over time to support the sorts of loads defined above, it is recommended they start out by initially deploying this Medium topology. This approach also provides much simpler evolution and manageability of the architecture as the workloads evolve and grow beyond the initial design and business expectations.

To size the application server to MySQL server ratio, a good rule-of-thumb is that each MySQL server will support 10 application servers, with additional slaves provisioned as application servers scale out in a read-intensive environment. For PHP applications, more application servers are generally required, and for Java, it is fewer.

Session Management & Ecommerce

Both workloads use the default InnoDB storage engine to provide transactional support and crash recovery. Use of the NoSQL Memcached API for InnoDB (discussed later) provides an effective way of scaling the performance of the session management workload.

To deliver high availability, options include MySQL replication with Global Transaction Identifiers and auto-failover, or OS-based solutions like DRBD supporting synchronous replication (both are discussed later in this section of the Guide). MySQL Enterprise Backup is also used.

In typical web properties, session state is usually maintained for 45 – 60 minutes in a dedicated database partition, with rolling partitions used to quickly delete aged session data by dropping the affected table.

For data mining and business intelligence applications, Session and Ecommerce data is captured in an analytics database for off-line report generation. Options also exist for loading MySQL data into Apache



Hadoop or other big data platforms where it can be analyzed together with unstructured data sources. This is discussed in more detail in the “Large Web Reference Architecture” section of the Guide.

As web properties seek to enhance their users’ experience through personalization based on historic browsing behaviors and social media sentiments, session data is becoming more critical. As larger volumes of session data are managed and persisted in real-time the workload can become increasingly write-intensive, while also demanding very high levels of availability to ensure a seamless customer experience. In these scenarios, it makes sense to replace the InnoDB storage engine with MySQL Cluster, which is discussed later in the Guide.

Content Management

Scalability of the content management application is critical as this is a core part of the web service. MySQL Replication is used to deliver read scalability with each MySQL master typically supporting 20 slaves, though higher numbers of slaves are found in some larger environments. In a regular content management workload, each slave should be able to support up to 5,000 concurrent users.

These metrics are guides only, and are highly dependent on a number of factors, including:

- Read and Write volumes
- Distribution and load balancing of web traffic
- Caching mechanisms used

Distributed file systems are often used to store the physical assets of the content management system, with the metadata and indexing for each asset is stored within InnoDB tables managed by MySQL, protected by either MySQL replication or OS (Operating System) clustering solutions.

The content assets themselves – images, videos, documents, etc. – are not stored in the database, but rather within the file system, CDN (Content Delivery Network), or Cloud Storage (such as AWS S3 or OpenStack Object Storage). For physical storage, the content assets can be stored either on a SAN (Storage Area Network) or distributed across local storage devices attached to each server. As a SAN can be a Single Point of Failure (SPOF), it is recommended that only a mid-range or high-end product be selected as it usually includes the reliability mechanisms needed to deliver High Availability.

Caching

In the Medium Reference Architecture, a caching layer is deployed with MySQL to support scaling the session and content management services. Distributed memory caching layers, such as Memcached or Redis, have been widely used with MySQL in some of the largest web properties to enable high levels of scalability, allowing for faster page loads and the more efficient use of existing database resources.

Once the data is populated in the cache, future requests for the same data make use of the cached copy rather than fetching the data from the source database, reducing load on the database layer. Caches can hold not only data and result sets, but also, objects, such as pre-compiled HTML.

In-memory caches can be installed on dedicated servers or co-located with web, application or database servers. Memcached can be incrementally scaled-out in an on-demand fashion. Because caches can scale to support dozens or even hundreds of nodes with minimal overhead, any existing spare memory resource represents an opportunity to further scale the application. Caches present a non-blocking event-based service with no special networking or interconnect requirements.

Alternatively, MySQL also offers native NoSQL access via the Memcached API in both MySQL Server and MySQL Cluster, each of which can compress the caching and database tiers into a single data source. See the NoSQL API section later in this guide for more information.

MySQL Replication: Scaling and High Availability

MySQL Replication enables users to cost-effectively deliver application performance, scalability and high availability. Many of the world's most trafficked web properties rely on MySQL Replication to rapidly scale



beyond the capacity constraints of a single system, enabling them to serve hundreds of millions of users and handle exponential growth.

MySQL Replication is easy to provision, allowing for simple Master/Slave topologies through to complex chained clusters achieving massive scalability.

MySQL Replication is implemented by configuring one instance as a master, with one or more additional instances configured as slaves. The master will log the changes to the database, which are then sent and applied to the slave(s).

By default, MySQL replication is asynchronous. The master does not wait for the slave to receive the update, and so is able to continue processing further write operations without being blocked as it waits for acknowledgement from the slave. This gives high performance, but if the master crashes before the event is replicated, that data will be lost.

Alternatively, MySQL replication can be configured to be semi-synchronous. In this mode, a commit is returned to the master (and then the client) only when a slave has received the update. As a result, the risk of data loss from failure of the master failure is reduced.

By mirroring data between instances, MySQL replication is the most common approach selected by MySQL users for delivering High Availability. Global Transaction Identifiers (GTIDs) released in MySQL 5.6 enable replicated events to be easily tracked through a replication topology of masters and slaves. Using the MySQL replication utilities⁴ failures are detected, with automatic failover or user-controlled switchover⁵ and reliable slave promotion, allowing users to maintain service in the event of outages or planned maintenance.

MySQL 5.6 also supports transactional replication with crash-safe slaves and masters, allowing automatic rollback and recovery in the event of a failure. Replication Event Checksums ensure the integrity of data being replicated to a slave by detecting data corruption and returning an error, preventing the slave itself from becoming corrupt.

Using MySQL Replication, organizations are able to scale out their applications on commodity hardware by distributing their data across MySQL server farms, within and across data centers and regions. This gives users the flexibility to elastically add or remove instances to dynamically adjust capacity based on demand.

MySQL 5.6 includes support for multi-threaded slaves and Binary Log Group Commit, delivering 5x higher replication performance on both the master and the slave.

To further improve scalability, long running jobs such as analytics workloads can be offloaded to slaves, allowing masters to be dedicated to serving real-time transactional workloads.

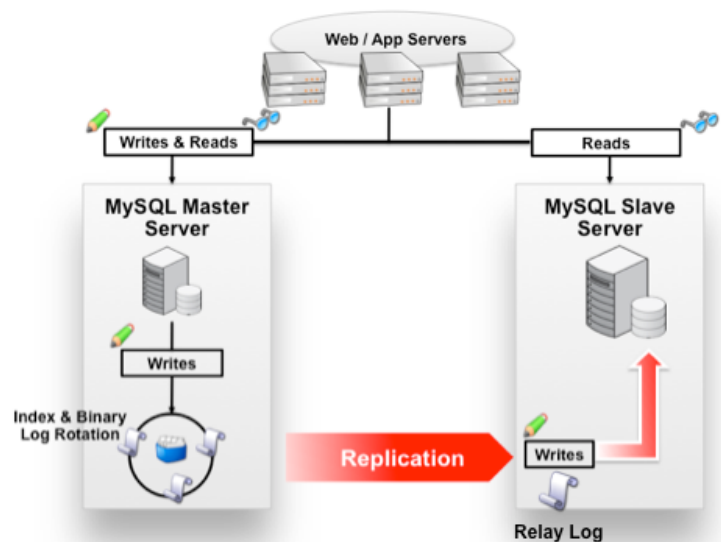


Figure 6: MySQL Replication Supports HA and Read Scalability “Out of the Box”

⁴ <http://dev.mysql.com/doc/workbench/en/mysqlfailover.html>

⁵ <http://dev.mysql.com/doc/workbench/en/mysqlrpladmin.html>



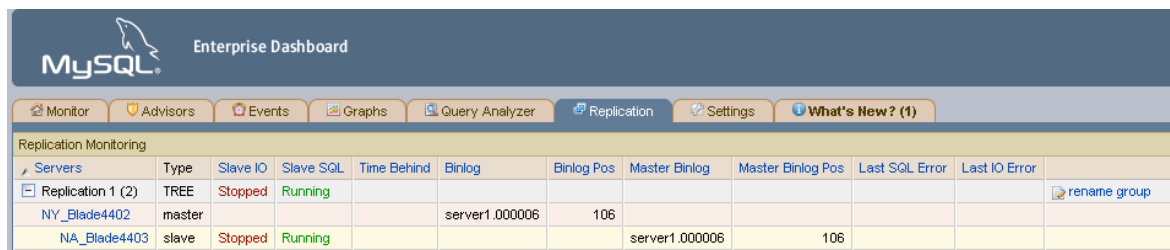
Learn more about MySQL replication and all of the latest features in MySQL 5.6 by downloading the Introduction to MySQL Replication:
<http://www.mysql.com/why-mysql/white-papers/mysql-replication-introduction/>

Managing MySQL Replication

In order to achieve high availability and scalability it is crucial to monitor systems and receive automatic notifications of issues or potential problems before they impact performance or availability of the service. Therefore, comprehensive management and monitoring tools should be regarded as mandatory for any serious web, cloud or mobile service.

Many MySQL customers use the MySQL Enterprise Monitor (discussed in more depth later in the guide) with its GUI dashboard to manage their replication topologies. MySQL Enterprise Monitor makes it easier to scale-out, providing auto detection, grouping, documenting and monitoring of all master / slave hierarchical relationships. Changes and additions to existing replication topologies are also auto-detected and displayed, providing DevOps teams with instant visibility into newly provisioned instances.

As the Replication Advisor identifies a problem and sends out an alert, the DBA can use the alert content along with the Replication Monitor to drill into the status of the affected master and/or slave. Using the Replication Monitor and the expert advice from the Replication Advisor they can review the current master/slave status and drill down into metrics such as Slave I/O, Slave SQL thread, seconds behind master, last error, etc. that are relevant to diagnosing and correcting any problems.



Replication Monitoring										
Servers	Type	Slave IO	Slave SQL	Time Behind	Binlog	Binlog Pos	Master Binlog	Master Binlog Pos	Last SQL Error	Last IO Error
Replication 1 (2)										
NY_Blade4402	master	Stopped	Running		server1.000006	106				
NA_Blade4403	slave	Stopped	Running				server1.000006	106		

Figure 7: MySQL Enterprise Replication Monitor

OS-Level Clustering

MySQL replication is widely deployed for HA in many classes of web applications, including the content management, session control and analytics services documented in these Reference Architectures.

When handling use-cases such as eCommerce, some users prefer the stricter HA and data durability constraints afforded by Operating System (OS)-level clustering, which deploy systems in more tightly coupled failover clusters using heartbeating mechanisms and cluster resource managers.

A failover cluster is a group of independent nodes that are physically connected by a local-area or wide-area network and programmatically connected by cluster software. The group of nodes is managed as a single system and shares a common namespace. The group usually includes multiple network connections and data storage connected to the nodes via storage area networks (SANs) or local distributed storage. The failover cluster operates by moving resources between nodes to provide service if system components (i.e. Server, OS, storage, network, database process, etc.) fail.

Heartbeating mechanisms implement protocols that send messages at regular intervals between two or more nodes. If a message is not received from a node within a given time interval, it is assumed the node has failed and the cluster resource manager initiates a failover action. Typically the IP address of the server will be virtualized (Virtual IP or VIP) which also fails over to the standby server, enabling



applications to continue running without having to connect to a different IP address. The virtual IP address is associated with a specific service, and can move between specific servers.

Multiple OS-Clustering solutions are certified and supported for MySQL, comprising both shared (i.e. SAN-based) and distributed (i.e. local) storage.

Shared storage solutions include:

- Oracle VM Template for MySQL Enterprise Edition: Packaged as a single downloadable image, the Oracle VM Template provides a pre-installed and pre-configured virtualized MySQL software image running on Oracle Linux and Oracle VM.
 - Users can rapidly provision multiple, virtualized MySQL instances to an Oracle VM Server pool, which integrates HA capabilities to provide resource monitoring, failover, recovery and load balancing of MySQL.
- Oracle Solaris Cluster and Windows Failover Clustering with agents enabling failover and recovery of MySQL.

Shared storage ensures the consistency and durability of data across different MySQL instances. Losing a master does not risk losing committed data.

DRBD (Distributed Replicated Block Device) provides an alternative approach using storage which is local to each instance. Using synchronous storage replication, all updates are propagated to a standby node before commits are returned to the application. Like shared storage, this enables “lossless failover”, ensuring no committed transactions are lost in the event of an outage to the master.

DRBD is complemented by Corosync and Pacemaker to provide clustering and resource management. All components are open source, enabling users to build high availability clusters from low cost, commodity infrastructure. In addition, the entire stack is certified and supported by Oracle.

To learn which technology is right for you, download the MySQL Guide to HA:

<http://www.mysql.com/why-mysql/white-papers/mysql-guide-to-high-availability-solutions/>

Scaling User Connections

By default the MySQL Database provides a thread-handling model that delivers excellent throughput and performance for 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 client terminates the connection. Under this model the MySQL Database provides scalable concurrency of both user connections and query executions.

While this model serves and scales most web deployments very well it does have the potential to limit scalability as connection and query loads rapidly increase. 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:

- The default model does not prioritize connection queries for execution, regardless of the number that has been submitted or that are in a “wait” status. No prioritization of queries means that all will attempt to execute in parallel with no awareness of 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 significantly increase user response times.
- If the number of parallel active threads is much higher than the number of available CPU cores, this leads to more process context switches and therefore to more CPU cache misses. This reduces the efficiency of the CPU cache, and as a consequence reduces the execution speed of threads and further increases contention.



MySQL Thread Pool

To meet these challenges, MySQL Enterprise Edition provides the MySQL Thread Pool⁶. 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 MySQL 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 the queries awaiting execution. Transactional queries are given a higher priority in the queue than non-transactional queries, 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.

The result is sustained performance and scalability as concurrent user connections and workloads grow, as shown in the benchmark below:

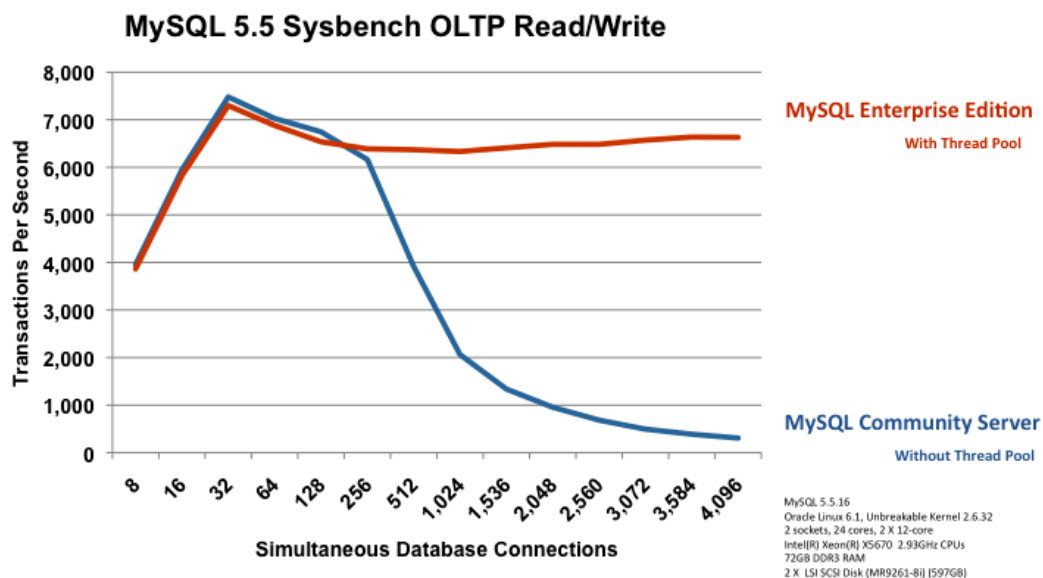


Figure 8: MySQL Enterprise Edition provides 20x better scalability for OLTP Read/Write activity with MySQL Thread Pool

⁶ <http://www.mysql.com/products/enterprise/scalability.html>.

Large Web Reference Architecture

				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 9: Sizing for Large Web Reference Architecture

The figures below show both a conceptual and detailed view of the recommended topology to support the “Large” workload.

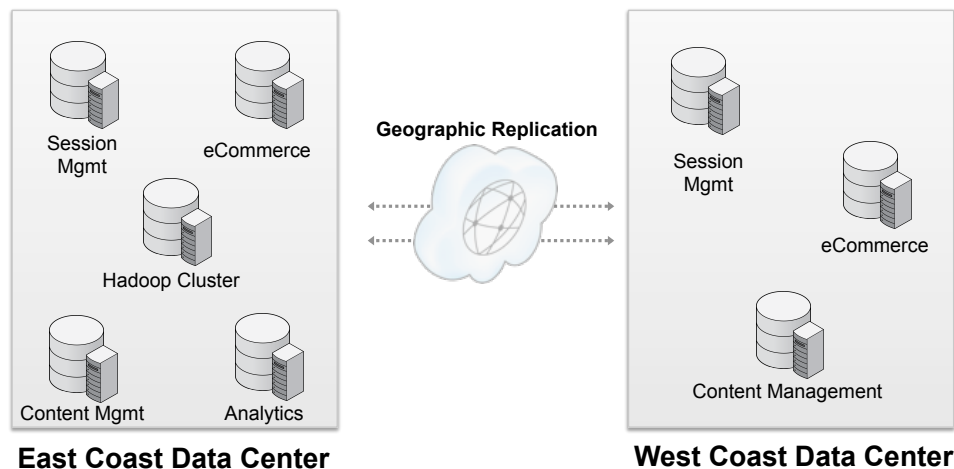


Figure 10: Conceptual View - Large Web Reference Architecture

In the conceptual view, we can see that MySQL databases are mirrored to a remote data center using MySQL replication, providing disaster recovery capabilities. Distributing the databases across regions also enables data to be physically co-located closer to users in order to reduce the effects of geographic network latency.

Typically the session management and ecommerce databases would be the prime candidates for replication. IP management could determine the user’s location and route them to the data center that was physically closest to them.

The Large Reference Architecture builds upon the best practices defined earlier in the Medium Reference Architecture. 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 – again dependent on traffic patterns of the workload.

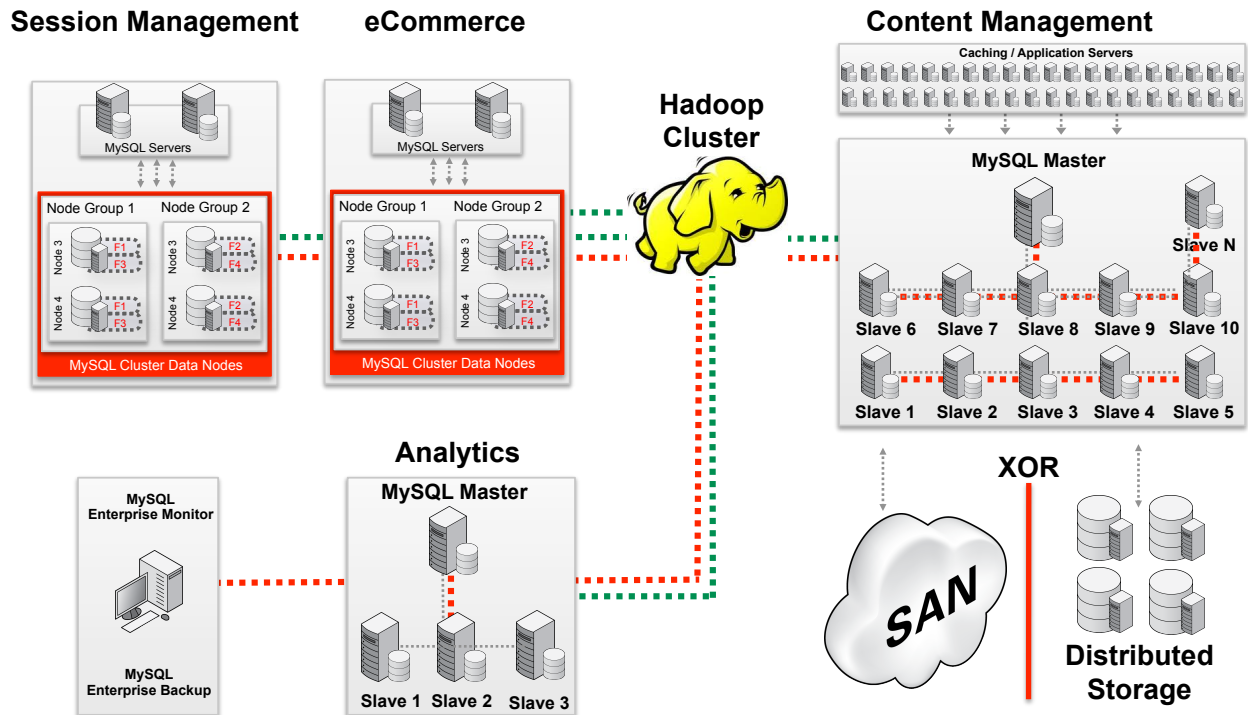


Figure 11: Detailed View - Large Web Reference Architecture

Hadoop Cluster for Big Data

In addition to the existing workloads, an Apache Hadoop cluster is added to support higher volume analytics and the integration of both structured data from MySQL and unstructured content collected from other sources such as clickstreams, social media, call records, etc.

As an example of the benefits of this integration, on-line retailers can use multi-structured data from their web properties to better understand site visitors' activities, such as paths through the site, pages viewed and comments posted. This data can be combined with user profiles and eCommerce history to gain a better understanding of customers, enabling the delivery of highly targeted offers and personalized experiences.

As with the Medium-sized Reference Architecture, session management and ecommerce data are used for analysis, but in this instance, they are first replicated to the Hadoop cluster, which then builds complete data sets and data warehouse dimensions for loading into the Analytics database.

Tables from MySQL can be imported in batches to Hadoop using Apache Sqoop, or with real-time streaming via the MySQL Binlog API. The results of Map-Reduce jobs can then be loaded into the MySQL analytics database, again using Apache Sqoop.

You can learn more about integration with Hadoop from the MySQL Guide to Big Data:

<http://www.mysql.com/why-mysql/white-papers/mysql-and-hadoop-guide-to-big-data-integration/>



MySQL Cluster for Session Management and Ecommerce

To handle the higher performance and availability demands of the “Large” web architecture, the session management and ecommerce databases are deployed on MySQL Cluster. With just two data nodes, it is possible to support 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 also possible to consolidate the session management and ecommerce databases into one larger cluster.

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, multi-master architecture with no single point of failure, MySQL Cluster scales horizontally on commodity hardware with auto-sharding to serve both read and write intensive workloads, accessed via SQL and NoSQL interfaces.

MySQL Cluster's 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.

Storing indexed columns in memory delivers very low latency when using MySQL Cluster. With this design, there are physical limits to database size, with most clusters not exceeding 3TB-4TB. There are many users who configure MySQL Cluster to manage the “hot” (most frequently accessed) data and then use its in-built replication to mirror aged data to InnoDB.

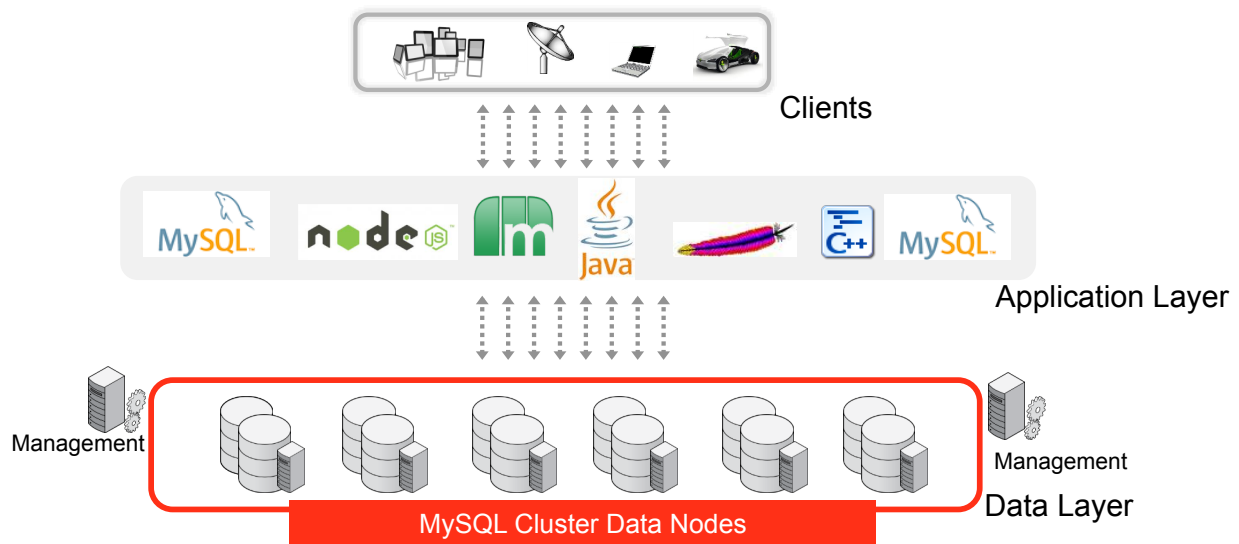


Figure 12: The MySQL Cluster architecture distributes your database across nodes, supporting high scalability and eliminating any single point of failure

MySQL Cluster presents a single namespace to the application, allowing clients to connect to any node. Under the covers, there are 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 interfaces including Memcached, JavaScript⁷, 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.

MySQL Cluster CGE includes 24x7 Oracle Premier Support and MySQL Cluster Manager to simplify the creation and management of the cluster by automating common management tasks. More information is included later in the guide.

To learn more about the use-cases and architecture, refer to the Guide to Scaling Web Databases with MySQL Cluster:

<http://www.mysql.com/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/>

Social Networking (Extra Large) Reference Architecture

Sizing & Topology

				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 13: Sizing for Social Networking Reference Architecture:

The Social Networking reference architecture re-uses many concepts defined in the “Medium” and “Large” reference architectures, including dedicated infrastructure for each of the workloads that make up the service.

MySQL Cluster is used for authentication of users and to provide the shard catalog - used by applications to direct queries.

User data is stored in database shards with high availability for each shard delivered by MySQL replication. Unlike the Web reference architectures, the Social Networking architecture manages session state on the shards themselves.

⁷ Part of the MySQL Cluster 7.3 Development Release

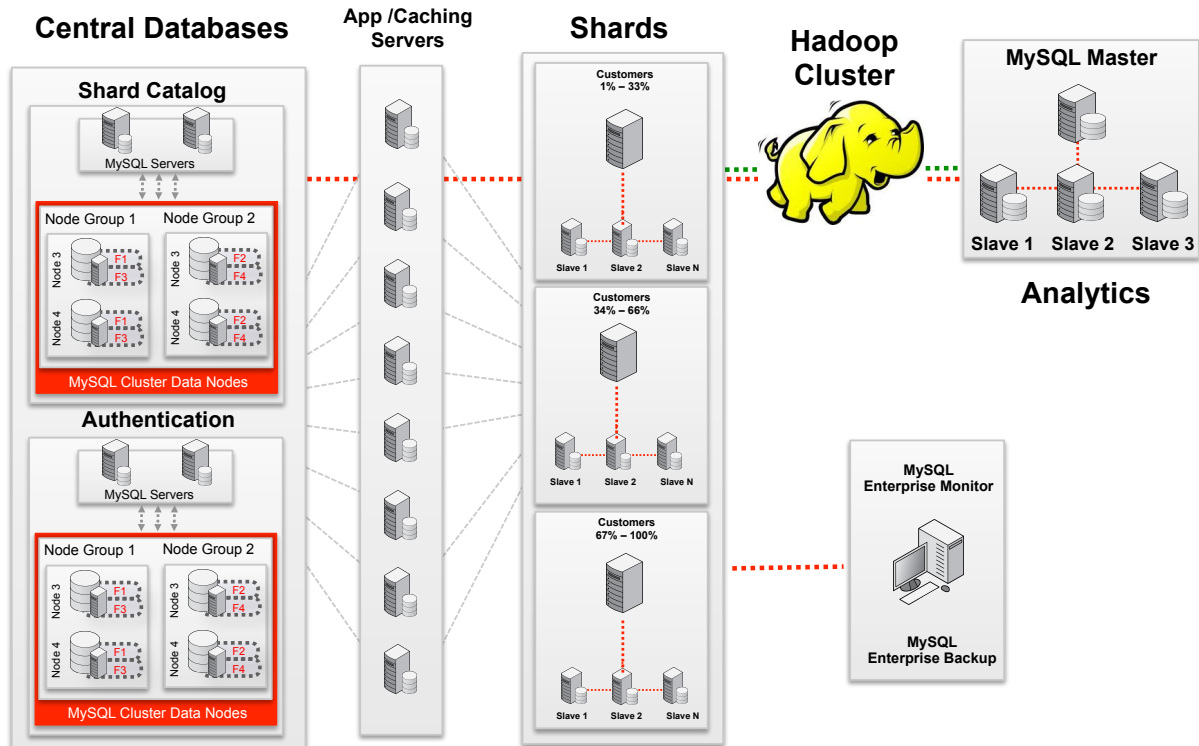


Figure 14: Topology for Social Networking Reference Architecture

Sharding

Sharding is commonly used by large web properties to increase the scalability of writes across their database. Sharding – also called application partitioning - involves the application dividing the database into smaller data sets and distributing them across multiple servers. This approach enables very cost effective scaling as low-cost commodity servers can be easily deployed when capacity requirements grow. In addition, query processing affects only a smaller sub-set of the data, thereby boosting performance.

In high traffic Web and mobile environments such as social networking much of the content is generated by users themselves, thereby demanding high levels of write scalability, handling rapidly changing data. It must be remembered that reads will still predominate in these environments as typically each record will be read before an update is applied.

It is also important to emphasize that millions of users can still be serviced without requiring any database sharding at all, especially with the latest MySQL developments in scaling on multi-core commodity hardware and replication.

When using sharding, applications need to become “shard-aware” so that writes can be directed to the correct shard, and JOIN operations are minimized. If best practices are applied, then sharding becomes an effective way to scale relational databases supporting web-based write-intensive workloads.

Hashing against a single column (key) often proves the most effective means to shard data. In the Social Networking Reference Architecture we have sharded the user database by User ID, so Shard One handles the first third of users, the second shard handles the second third of users, and the remainder are stored in the third shard. An alternative approach used in some social networks is “functional sharding”. In this scheme, each shard manages a different service or data type, with each shard storing the entire user base.



The approaches described above initially provide the most logical sharding mechanism and make it simple to scale-out as the service grows. However, it is likely that some shards will contain more active users than other shards; meaning re-balancing of shards may be required in the future. Both the application and the database need to be sufficiently flexible to accommodate change in the architecture. Implementing a sharding catalogue is a common way to achieve this. The central catalogue keeps track of data distribution and the application bases its sharding on this catalogue. This way it is easy to rebalance even individual data objects.

As sharding is implemented at the application layer, it is important to utilize best practices in application and database design. The Architecture and Design Consulting Engagement offered by the MySQL Consulting group in Oracle has enabled many customers to reduce the complexity of deployments where sharding is used for scalability.

By deploying MySQL with sharding and replication, leading social networking sites such as Facebook, Tumblr, Twitter and Pinterest have been able to exponentially scale their MySQL databases.

If applications involve high volumes of write operations coupled with the need for continuous operation, it is worthwhile also considering MySQL Cluster.

NoSQL APIs

With web and mobile data volume and velocity exploding, it is vital to be able to query and ingest data at high speed. For this reason, MySQL has implemented NoSQL interfaces directly to the InnoDB and MySQL Cluster storage engines. These bypass the SQL layer completely, without SQL parsing and optimization. As a result, Key-Value data can be written directly to MySQL tables up to 9x faster, while maintaining ACID guarantees.

In addition, users can continue to run complex queries with SQL across the same data set, providing real-time analytics to the business or anonymizing sensitive data before loading to big data systems, while still maintaining all of the advantages of their existing relational database infrastructure.

A native Memcached API is available in the MySQL database and MySQL Cluster. By using the ubiquitous Memcached API for writing and reading data, developers can preserve their investments in Memcached infrastructure by re-using existing Memcached clients, while also eliminating the need for application changes.

As discussed earlier, MySQL Cluster also offers additional NoSQL APIs beyond Memcached, including Node.js⁸, Java, JPA, HTTP/REST and C++.

To learn more about NoSQL APIs and MySQL, download the Guide:

<http://www.mysql.com/why-mysql/white-papers/guide-to-mysql-and-nosql-delivering-the-best-of-both-worlds/>

Schema Flexibility

Speed, when combined with flexibility, is essential in rapidly evolving web and mobile services.

Support for on-line DDL (Data Definition Language) operations in MySQL 5.6 and MySQL Cluster enables DevOps (Developer / Operations) team to dynamically update their database schema to accommodate rapidly changing requirements, such as the need to capture additional data generated by new services. Schema evolution can be made without database downtime.

⁸ Part of the MySQL Cluster 7.3 Development Release



The Perfect Server for MySQL

MySQL is supported across all leading platforms, comprising multiple CPU architectures and operating systems. The following section identifies the most common platforms that have proven to deliver optimum performance, scalability and availability for web-based architectures.

When sizing the server, it is important to profile the size and characteristics of the databases you will be running now, and estimate future sizing requirements as the workload scales. It is also important to remember that replication slaves should be as powerful as those deployed as replication masters. The recommended server specification is as follows:

- Up to 48 x86-64 bit CPU threads (MySQL 5.6 and above). Note that some benchmarks show linear scalability on 60+ threads!
- Recommended RAM at least equal to or larger than the “hottest” (most regularly accessed) data set.
- Linux, Oracle Solaris or Microsoft Windows operating systems.
- Minimum of 4 x SSDs or HDDs. 8 – 16 drives will increase performance for I/O intensive applications.
- Hardware RAID with battery-backed cache.
- RAID 10 recommended. RAID 5 is suitable if the workload is read-intensive.
- 2 x Network Interface Cards and 2 x Power Supply Units for redundancy.

MySQL Cluster

Due to the distributed nature of MySQL Cluster, the recommended server configurations are slightly different than a MySQL server running InnoDB storage engine.

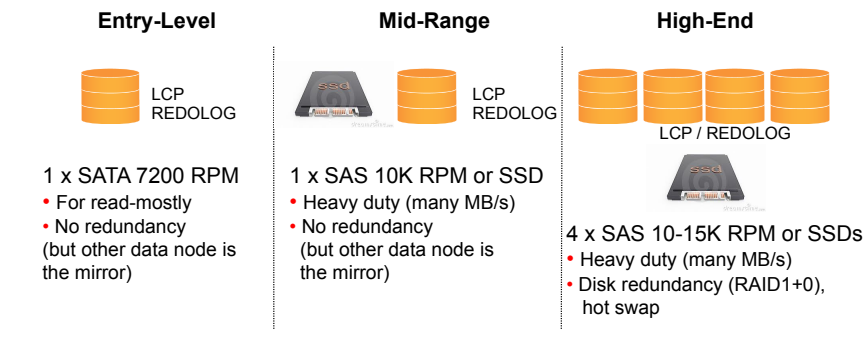
Application Nodes (MySQL Servers) Recommendation

- 4 - 24 x86-64 bit CPU threads.
- Minimum 4GB of RAM. Memory is not as critical at this layer, and requirements will be influenced by connections and buffers.
- 2 x Network Interface Cards and 2 x Power Supply Units for redundancy.

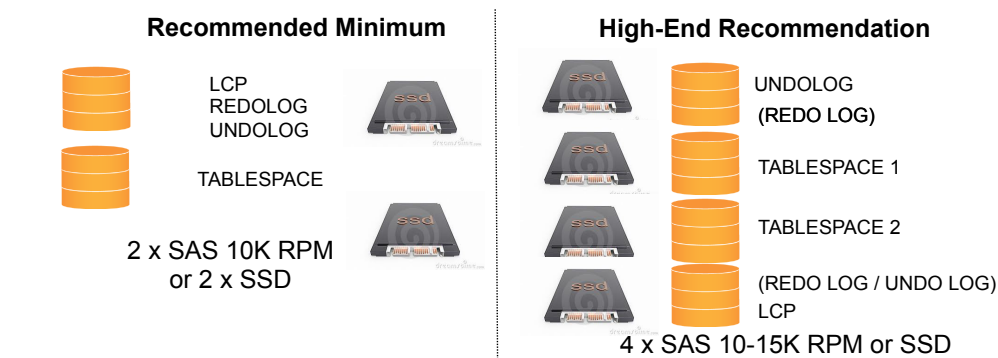
Data Nodes Recommendation

- Up to 64 x86-64 bit CPU threads. Use as high a frequency as possible as this will enable faster processing of messages between nodes.
- $\text{RAM per Server} = \text{Database Size} * \# \text{ Replicas} * 1.25 / \# \text{ data nodes}$. (Data redundancy + indexes drive overall memory requirement)
 - Example: 50GB database * 2 replicas * 1.25 / 2 data nodes = 64GB of RAM per data node.
 - This assumes the database is stored entirely in-memory. MySQL Cluster supports the storage of non-indexed columns as disk-based tables, reducing per-node RAM requirements.
- Linux, Oracle Solaris or Microsoft Windows operating systems.
- 2 x Network Interface Cards and 2 x Power Supply Units for redundancy.
- Disk-drive recommendations as follows.

Figure 15: Recommended Disk Drive Configurations for Checkpointing & REDO Logs:



- REDO, LCP, BACKUP – written sequentially in small chunks (256KB)
- If possible, use Odirect = 1



- Use High-End Recommendation for heavy read / write workloads
 - (1000's of 10KB records per sec) of data (i.e. Content Delivery platforms)
- Having TABLESPACE on separate disk is good for read performance
- Enable WRITE_CACHE on devices

Figure 16: Recommended Disk Drive Configurations for Disk-Based Tables

The greatest benefit from using SSDs comes with disk-based tables, which can fully exploit the advantages of fast random file access.

It is recommended to deploy the data and application nodes on a dedicated network (i.e. IP addresses starting at 10.0.1.0), using 1 Gigabit Ethernet as a minimum transport. The MySQL servers would then also be attached to the public network.

To provide resilience to network failures, it is recommended to configure each server with bonded and redundant Network Interface Cards (NICs) connected to redundant switches. If multiple data nodes are deployed onto a single host, it is recommended to bond four or more NICs together to handle the increased network bandwidth requirements. It is best practice that data nodes within the same node group are deployed on separate physical hosts in order to eliminate any single point of failure.



Operational Best Practices

For business critical applications that are part of web and mobile services, the support, advanced features and tooling that are included in MySQL Enterprise Edition and MySQL Cluster CGE will help you achieve the highest levels of performance, scalability and availability.

MySQL Enterprise Edition and CGE include:

- **Oracle Premier Support for MySQL:** Staffed with MySQL Experts, Oracle's dedicated MySQL technical support team can help you solve your most complex database issues rapidly, and get the most value from your MySQL deployments. Oracle MySQL Support Engineers are available 24/7/365 either online or by phone and have direct access to the core MySQL developers. Oracle Premier Support is available in 29 languages and includes an unlimited number of support incidents.
- **MySQL Enterprise Monitor:** Continuously monitoring your databases and alerting you to potential problems before they impact your system. See the following section: "Managing and Monitoring Your Server Estate".
- **MySQL Query Analyzer:** Included in the MySQL Enterprise Monitor, the 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. Queries can be analyzed during active development and continuously monitored and tuned when in production.
- **MySQL Enterprise Scalability:** Enables you to meet the sustained performance and scalability requirements of ever increasing user, query and data loads typical with web and mobile services. MySQL Thread Pool provides an efficient, thread-handling model designed to reduce overhead in managing client connections, and statement execution threads, delivering up to 20x higher performance.
- **MySQL Enterprise Security:** Provides ready to use external authentication modules to easily integrate MySQL with existing security infrastructures including PAM and Windows Active Directory, ensuring secure access to your most sensitive data.
- **MySQL Enterprise Audit:** Enables you to quickly and seamlessly add policy-based auditing compliance to web applications. You can dynamically enable user level activity logging, implement activity-based policies, manage audit log files and integrate MySQL auditing with Oracle and third-party solutions.
- **MySQL Enterprise Backup:** Reduces the risk of data loss and minimizes the impact of backup to the user experience by enabling online "Hot" backups of your databases.
- **MySQL Enterprise High Availability:** Provides you with certified and supported solutions, including MySQL Replication, DRBD, Oracle VM Templates for MySQL, Oracle Solaris Clustering and Windows Failover Clustering.
- **MySQL Cluster Manager:** Simplifies the creation and administration of the MySQL Cluster CGE database by automating common management tasks, including on-line scaling, backups, upgrades and reconfiguration.
- **MySQL Workbench:** Provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, and much more.



- **MySQL Enterprise Oracle Certifications:** MySQL Enterprise Edition is certified with Oracle Linux, Oracle VM, Oracle Fusion middleware, Oracle Secure Backup, Oracle GoldenGate, and additional Oracle products.

You can learn more about each of the features above from the MySQL Enterprise Edition Product Guide: http://www.mysql.com/why-mysql/white-papers/mysql_wp_enterprise_ready.php

Additional MySQL Services from Oracle to help you develop, deploy and manage highly scalable & available MySQL applications for web and mobile services include:

Oracle University

Oracle University offers an extensive range of MySQL training from introductory courses (i.e. MySQL Essentials, MySQL DBA, etc.) through to advanced certifications such as MySQL Performance Tuning and MySQL Cluster Administration. It is also possible to define custom training plans for delivery on-site.

You can learn more about MySQL training from the Oracle University here: <http://www.mysql.com/training/>

MySQL Consulting

To ensure best practices are leveraged from the initial design phase of a project through to implementation and sustaining, users can engage Professional Services consultants. Delivered remote or onsite, these engagements help in optimizing the architecture for scalability, high availability and performance. You can learn more at <http://www.mysql.com/consulting/>

Management & Monitoring of your MySQL Server Estate

MySQL Enterprise Monitor, included with MySQL Enterprise Edition, is a distributed web application that continually monitors your MySQL servers and proactively sends SNMP/SMTP alerts on potential problems and tuning opportunities, before they become costly outages. It also provides MySQL Advisors and 160+ Advisor Rules that deliver MySQL best practices advice relating to administration, security, performance, replication setup and more so you know where to spend your time in optimizing MySQL applications.

Enterprise Dashboard for Monitoring all MySQL Servers

Using the Enterprise Dashboard, you can monitor MySQL and OS specific metrics for single or groups of servers, and can stay on top of all their replication topologies. The Enterprise Dashboard is designed so you can easily understand the complete security, availability, and performance landscape of all MySQL servers in one place, all from a thin, browser-based console.



Figure 17: MySQL Enterprise Dashboard



Availability and Performance Diagnosis

The Enterprise Dashboard includes a color-coded Heat Chart that provides an at-a-glance view into the availability and performance of all of the MySQL servers across the enterprise. From the Heat Chart you can instantly tell:

- The Up/Down status of all MySQL servers;
- Key OS metrics that may be affecting MySQL;
- Which MySQL servers need attention;

MySQL Advisors

The MySQL Enterprise Monitor differs from traditional third-party database monitoring tools in that it supplies a complete set of MySQL Advisors that are designed to automatically analyze a MySQL server's configuration, security, and performance levels; identify problems and tuning opportunities; and provide you with specific corrective actions. The MySQL Enterprise Monitor ships with the following set of best practice Advisors:

- **MySQL Enterprise Backup** – Monitors and advises on MySQL Enterprise Backup operations, status and MySQL servers needing backup.
- **Upgrade** - Monitors and advises you on using the most secure and up to date version of MySQL you should deploy.
- **Administration** – Monitors and advises on problems relating to general database administration, recoverability and performance configuration settings
- **Security** – Monitors for security vulnerabilities in the MySQL database and advises on how to protect against potential security breaches
- **Replication** – Monitors and advises on problems relating to Replication setup, security and Master/Slave latency
- **Memory Usage** - Monitors dynamic memory related server metrics (cache usage, hit ratios, etc.) and advises on configuration changes to improve performance
- **Performance** – Monitors dynamic performance related server metrics and advises on configuration and variable settings to improve performance
- **Schema** – Monitors for unplanned changes to a database schema and advises on possible schema issues.
- **MySQL Cluster** – Monitors MySQL Cluster data nodes memory, undo/redo buffer space, undo/redo log space, and node up/down status, advising on how to best optimize them.
- **Custom** - Create your own Best Practice Advisors and Rules to fit your specific use of MySQL.

MySQL Enterprise Monitor employs 160+ MySQL Advisor Rules that monitor over 600 MySQL and OS specific variables and metrics that track and report on the overall health, security, availability and performance of each MySQL server.

Conclusion

Designing, managing and scaling web-based infrastructure is challenging. Technology innovations are happening at a much greater pace than ever before, and user requirements change even more rapidly.

The demand for new services, coupled with the explosion in data that needs to be managed creates significant demands for any organization looking to use the web and cloud platforms to drive their business.

By leveraging the best practices identified in this Guide – developed by working with some of the largest properties on the planet - we hope to have provided a starting point to enable you to build the next web and mobile phenomenon.



Additional Resources

MySQL 5.6 Developer and DBA Guide:

<http://www.mysql.com/why-mysql/white-papers/whats-new-mysql-5-6/>

MySQL Replication: Introduction

<http://www.mysql.com/why-mysql/white-papers/mysql-replication-introduction/>

MySQL Cluster: Guide to Scaling Web Databases:

<http://www.mysql.com/why-mysql/white-papers/guide-to-scaling-web-databases-with-mysql-cluster/>

MySQL & NoSQL – The Best of Both Worlds:

<http://www.mysql.com/why-mysql/white-papers/guide-to-mysql-and-nosql-delivering-the-best-of-both-worlds/>

MySQL and Hadoop – Big Data Integration:

<http://www.mysql.com/why-mysql/white-papers/mysql-and-hadoop-guide-to-big-data-integration/>

MySQL: An Ideal Choice for the Cloud:

<http://www.mysql.com/why-mysql/white-papers/mysql-an-ideal-choice-for-the-cloud/>