WHITE PAPER

Adding Speed and Scale to MySQL

In-Memory Computing Options for MySQL Server Deployments

MySQL® is one of the most popular open source databases. Even after the acquisition by Oracle of MySQL as part of Sun, innovation has continued and the options for MySQL customers have grown. Three major distributions and many more options for improving speed and scale give companies almost too many choices. But new business needs and performance demands have pushed many applications and their underlying RDBMSs beyond their architectural limits. In many cases the issues cannot be remedied by just fixing MySQL.

Part of the challenge is scalability. The adoption of digital business, IoT, and other new technologies over the last decade has increased query and transaction loads 10-1000 times, and the amount of data collected by 50 times.

Another major challenge is speed. Customer-facing web and mobile apps, and their underlying APIs, all require sub-second roundtrip latencies. This low latency requirement is impossible to support given all the network hops, software layers, data queries, and joins that occur between the new customer apps and underlying applications and databases. In addition, the amount of data needed for different analytics and other computing types—including machine and deep learning—has become too big to move quickly enough over the network.

Another challenge lies in the development of the applications themselves. New business initiatives (such as changes to improve the end-to-end customer experience) require the delivery of new capabilities in days or weeks, rather than months or years. But most existing applications are inflexible to change. It takes months just to deliver minor changes. Most applications also do not support many newer technologies such as streaming analytics or machine and deep learning, which are required to help streamline, automate, and improve real-time processes.

This white paper explains your options for adding speed, scale, and agility to end-to-end IT infrastructure. It examines different MySQL and third party options including MySQL InnoDB Cluster, MySQL NDB Cluster, Galera Cluster, MariaDB, Percona Server for MySQL with XtraDB Cluster, Clustrix, Vitess, as well as cloud options such as Amazon RDS for MySQL and Amazon Aurora, caching options such as memcached and Redis, and in-memory computing (IMC) technology.

It also explains the third party options available that help evolve your architecture for increased speed and scale, and create flexible IT infrastructures that support digital business and other new technology initiatives.

THE CHALLENGE WITH SPEED, SCALE, AGILITY, AND AUTOMATION

The latest variations of MySQL and their extensions cannot deliver the performance and scalability required by changes in the demands placed on deployments, including:

- The adoption of new web, mobile, and other self-service channels.
- The addition of personalization and other automation.
- The use of new types of data.

These new demands require speed and scale not just at the database, but at other system layers as well.

Over the last decade, these innovations have led to query and transaction volumes growing 10 to 1000 times. They have resulted in 50 times more data about customers, products, and interactions. They have also shrunk customers expected end-to-end response times from days or hours to seconds or less. The required roundtrip latency from a mobile device calling an API that accesses applications over the network—which in turn access MySQL variations over the network—must always be less than one second.

The existing architecture cannot deliver with existing applications and databases. For one, vertical scalability is not a long-term option: the 10-1000 times growth rates (which show no signs of slowing down) are faster than Moore's Law. Even if more is spent on hardware at the outset, eventually no single hardware server can keep up with the growth. Latency is also a major issue. The combination of multiple network hops and queries involving large data sets make achieving the required speed and scalability practically impossible with the current architecture. As one architect put it, "You can't violate the laws of physics." The system needs lower end-to-end network latency, not just database latency.

Even if databases alone could address the speed and scale challenges, the applications themselves introduce a third major challenge. Customers have come to expect rapid change. The new innovators that are disrupting just about every business model deliver new capabilities in weeks or days. Today's applications are not that flexible: it can take months or years to develop and deploy changes.

Then there is the final challenge: automation. Streamlining a process, creating a one-click shopping experience, or proactively fixing issues as they occur all require real-time intelligence and automation. The architecture of existing applications and databases separates online transactions processing (OLTP) from online analytical processing (OLAP), data warehousing, and business intelligence. OLTP systems

cannot take the additional analytics load and were not designed for that task. OLTP applications usually only solve one problem and only use a portion of a business' data. To create a single view of the business for analytics, companies must Extract, Transform, and Load (ETL) data into a warehouse, cleanse the data, enrich it, and make it consistent across different data types—all to make reporting and ad hoc analytics work. When these ETL pipelines and data warehouses were created, neither the freshness of the data nor the speed of reporting was as important as the accuracy of the result.

Now the analytics and the decisions must happen in real-time (often during a transaction) to deliver the required sub-second experience. That is impossible to do with existing applications. The only way to run analytics quickly enough is to perform the analytics on the transaction data in the same location where the transaction takes place. Most applications, however, do not include analytics or technologies such as machine or deep learning that help with decision automation.

THE EVOLUTION TOWARDS IN-MEMORY COMPUTING

Many innovators have been able to address all the speed, scale, agility and automation challenges with what Gartner coined a Hybrid Transactional/Analytical Processing (HTAP) architecture. One of the most common foundations for HTAP is in-memory computing. One of the most common in-memory computing platforms are Apache Ignite and GridGain (the commercially supported version of Apache Ignite).

With Apache Ignite, companies could adopt in-memory computing incrementally, one project at a time. Most companies started by implementing Ignite as an in-memory data grid (IMDG) in-between existing databases and applications rather than buying additional database licenses or adding new database hardware. This method offloaded reads from the databases, giving the databases substantially more room for growth on their existing hardware and lowering latency.

IMDG deployments also gave companies a way to evolve using existing applications. They could be more agile by unlocking the data inside their applications for use by other projects. For each project, as more data was added into their common in-memory computing infrastructure, the combined data could be used to:

- Perform real-time analytics.
- Add speed and scale for new APIs.
- Process new types of data with streaming analytics.
- Help with automation using machine and deep learning.

These deployments also improved business agility. New workloads are easy to accommodate by adding more nodes as needed, and more copies of the data, without impacting existing applications. Ignite is also cloud native, which makes using it in a DevOps environment straightforward.

OPTIONS FOR ADDING SPEED AND SCALE TO MYSQL

Adopting HTAP and in-memory computing is the right long-term approach for adding speed, scale, agility, and digitally transforming businesses into real-time enterprises. But there are also other short- and long-term options for helping with speed and scale.

MySQL has been around for over 20 years. It was founded in 1994, and first released in 1995. In 2003 MySQL AB acquired Alzato Tech, an Ericsson-based software company that built cluster technology to make MySQL telco-grade technology. MySQL AB proceeded to integrate the NDB Cluster technology into MySQL. Over the next few years, third party options started to emerge for MySQL.

- NDB Cluster: MySQL buys Alzato Tech in 2003.
- InnoDB: Oracle buys InnoDB, a storage engine for MySQL in 2005.
- Percona: released Percona Server for MySQL in 2006.
- Galera Cluster released for MySQL in 2007.
- MariaDB: A MySQL fork started the same day Oracle closed the Sun (and hence MySQL) acquisition in 2010.
- Clustrix: a drop-in replacement for MySQL founded in 2006, acquired by MariaDB in 2018.

Oracle's acquisition of Sun (Sun acquired MySQL AB in 2008) birthed a new fork of MySQL: MariaDB. This further fragmented the MySQL market.

Today, companies can choose between Oracle, Percona, and MariaDB for MySQL deployments, and have a host of options for adding speed and scale. Many options address the different challenges with scale, but not nearly enough adequately address the challenges with speed. There are also other open source and third-party options that address these challenges outside of the database. This whitepaper provides a comparison of all the options for improving scale, as well as guidelines for how to improve both speed and scale together by moving towards in-memory computing overtime.

First, we will compare the main MySQL options available for adding speed and scale, along with recommendations for when to choose each option. We did not evaluate for OLAP/analytical scale, in part because there are no easy ways to just add columnar support. MariaDB ColumnStore and InfiniDB are two options. But benchmarks for these features tend to show that while they provide MySQL compatibility, it comes with slower performance. MemSQL is wire-compatible with MySQL, which means if data is migrated into MemSQL using the same table structure, then the MySQL clients can connect to MemSQL. The challenge with MemSQL is that while it has great OLAP performance, it is not as good for OLTP workloads.

The main option to achieve true OLAP scale is to build a dual stack. For example, in the cloud use Amazon RDS for Aurora or MySQL, and then pipeline data into Redshift. But this is all outside the scope of this paper. If you are interested, stay tuned for an in-memory computing best practices whitepaper on real-time analytics.

LOWERING LATENCY: MEMORY TABLES, MEMCACHED, AND REDIS

As with other open source databases (like PostgreSQL®), there are only a few options for lowering the latency of a MySQL deployment. One option for any databases is simply to use a RAM disk, or something like Intel Optane DC Persistent Memory along with an operating system that supports Optane at the file system level for storage. We assume you know this trick.

The first in-memory option is MySQL Memory Tables. The MySQL memory storage engine can create tables in memory. It is only recommended that this method is used as temporary tables in smaller deployments, since memory tables are not backed up to disk, can't be clustered, and don't support foreign keys. Even more importantly, they don't support transactions or guarantee MVCC. Even write scalability is limited as they can only be locked at the table level. Note also that the standard InnoDB storage engine does not support in-memory tables.

The second option is using memcached, which is a well-documented technology to use with MySQL. But it requires work. Applications must be changed so that they populate and check memcached for data, and also manage refreshing memcached. It is important to note the limitations of memcached:

- While it offloads reads from MySQL and lowers read latency, it does not easily scale data sets horizontally. You can either have memcached locally with an application, which means you are limited by the amount of available RAM on the machine, or you can install several memcached servers remotely and fetch data from the cache over the network. This may not seem like a big issue, but most corporate networks can only move one gigabyte per second across all servers at most. Companies are increasingly hitting that limit.
- Memcached is still hard to scale horizontally. For large data sets it is probably best to look for solutions elsewhere.
- There is no real security within memcached. Evaluate security requirements carefully before proceeding.
- Fault tolerance can be challenging, though it is manageable.

Another option is Redis. It is very similar to memcached, both in capabilities and challenges. It is key-value only, and a cache-aside cache that requires changes to the application to use it. Redis does scale better, but in practice it is hard to manage data partitioning (or sharding). When scaling with Redis, you usually need to create a dedicated cluster on different machines, which means you have the same network limitations as memcached. Security is also an issue with open-source Redis. Only Redis Enterprise contains reasonable security options.

In short, there is no perfect option within the MySQL community for lowering latency—especially at scale.

OPTIONS FOR SCALING MYSQL VERTICALLY AND HORIZONTALLY

To improve scale, scaling vertically might seem like a good option in the short term. MySQL does scale vertically (like most other databases), but it is not as cost effective or (in several cases) a viable long-term option. Data is growing too fast for hardware in general, and the capacity of a single machine or network to cope. Hardware costs also explode once they exceed the capacity of current commodity hardware.

Horizontal scalability is the only viable long-term option. Traditional horizontal scaling solutions such as replication can help with read scalability, but they do not help with scaling the size of the database or with write scalability. The best answer for both is to create a distributed database by horizontally replicating and partitioning the data, the SQL, and other computing.

Read scalability through replication works well. Some solutions provide multi-master write support as well. But the consistent message from any research online or with experts is the same: when it comes to scaling to supporting large data sets, partitioning is generally considered as a last resort. Core MySQL provides support for sharding data manually by key ranges. This might be enough for some deployments. But managing the partitions for adding write scalability and failover is still largely a manual job. It is also hard to rebalance data as nodes are added or dropped, or the data distribution changes. Sharding gets more complicated with more tables. In addition, different applications that use the data in different ways often require different data partitioning strategies.

In general, the best option is either evaluating options for MySQL clustering or options that allow you to add horizontal scale beyond the database layer. Within the MySQL world, the options are clustered around three vendors: Oracle, Percona, and MariaDB.

It used to be that drop-in support for MySQL worked well. But this has gotten harder to maintain in recent years with recent underlying changes. First, while MariaDB and Percona do their best to merge in MySQL changes, it still takes time. Second, these companies each make their own improvements. MariaDB in particular has been adding new features almost from the beginning and is less compatible. Percona adds their own implementation of MySQL Enterprise-equivalent features and rolls in other changes as well, though they still are drop-in compatible with MySQL today. Depending on your needs, you may also decide to migrate to another vendor to get the best combination of MySQL with the right clustering option.

The following is a comparison of the main open source and commercial options that can be considered for MySQL:

- MySQL InnoDB Cluster
- MySQL NDB Cluster
- Galera Cluster
- MariaDB (with Galera Cluster)
- Percona Server for MySQL with XtraDB Cluster
- Clustrix
- Vitess
- Cloud versions of MySQL (e.g., Amazon RDS for MySQL, Amazon Aurora)

There are several routers that sit in front of MySQL. In general, there are de facto choices used with the various clustering options. These are covered below with each Cluster option:

- MySQL Router: designed for use with InnoDB Cluster. Designed to support read-only, read/write and MySQL Fabric configurations.
- ProxySQL: Used with Percona XtraDB Cluster. Support for MySQL Replication and Group Replication. Support for Galera Cluster through scripting.
- MaxScale: Used with MariaDB and Galera Cluster.
- HAProxy: Widely used reverse proxy for applications but does not support MySQL as well for basic routing such as splitting reads and writes.
- Keepalived: Another reverse proxy, which, like HAProxy, is not specific to SQL.
- Nginx (Plus): Great HTTP load balancer as an alternative to appliances, but like HAProxy, not meant for SQL-specific routing.

The three main SQL-specific routers are covered: MySQL Router, ProxySQL and MaxScale.

There are also various cloud options to consider if you are considering migrating to the cloud. Amazon offers RDS for MySQL, for example, which makes it easier to manage features like replication and failover. Azure and Google have similar offerings. Amazon Aurora is compatible with MySQL and has horizontal scalability for online transactional processing (OLTP) applications. This comparison will summarize those options as well.

Multi-Master Read/Write Replication: MySQL InnoDB Cluster

Since buying InnoDB in 2005, and then MySQL in 2010, Oracle has made some investments. One was to build the InnoDB Cluster. Beyond open source MySQL, which is free, Oracle sells:

- MySQL Standard Edition ($2,000 per server): support and maintenance for MySQL.
- MySQL Enterprise Edition ($5,000): adds MySQL Router, Partitions, monitoring and management, backup and recovery, security, thread pooling, Group Replication, and InnoDB Cluster.
- MySQL Cluster CGE ($10,000): Adds the NDB storage engine, NDB Cluster, MySQL Cluster Manager, and Geo-Replication.

There are really three different replication and clustering technologies, which are compared below:

- NDB Cluster, which was acquired by MySQL in 2003.
- Galera Cluster, which has three variants offered by Codership, the inventors of Galera, Percona and MariaDB.
- MySQL InnoDB Cluster. By the end of 2016, Oracle had introduced Group Replication, which they rewrote from the ground up.

Using InnoDB Cluster requires purchasing Enterprise Edition at the very least. The standard configuration is MySQL Router, with InnoDB Group Replication and the InnoDB storage engine as part of MySQL. InnoDB Group Replication provides single (master-slave) replication by default, but also supports multi-master replication. For each change, the entire database is replicated synchronously or asynchronously using MySQL Group Replication to each node. Each node can accept both reads and writes. Any write conflicts detected and handled based on timestamps, which can be thought of as a form of optimistic transactions. Whichever transaction fails after the conflict resolution may need a compensating transaction. Group Replication does have some split brain prevention as well. Management is not as good as some of the other options. For example, there is no automatic provisioning of nodes for group replication. Node failures need to be managed explicitly. In addition, flow control (or throttling) is done separately by each node. This means each node holds statistics on others, versus in a centrally controlled cluster where statistics and flow control are managed in a single place for all nodes (which is typically a better solution). There is also no WAN support for multiple data centers. For WAN-based deployments, evaluate the Cluster CGE edition.

Note that MySQL InnoDB Cluster is a shared-something architecture, like Oracle Database RAC, in that the cluster coordinates locks across the nodes, as well as detects and resolves any contentions. The challenge is, as with Oracle RAC, that the amount of resources needed for this coordination increases exponentially with the number of nodes. This growth in resource consumption puts a clear limit on horizontal read and write scalability.

Regarding partitioning, InnoDB Group Replication is limited. With InnoDB, there are several limitations with partitions. For example, foreign keys, stored procedures, UDFs, or plugins are not all supported when using partitions, There are also limitations in what functions are supported in each of these features. You can still partition and rebalance manually, as well as restart and rejoin nodes as issues occur. But it is mostly manual.

Without partitioning, the InnoDB storage limit is 64TB, which is not that large given current data growth rates. If you need to grow your database and use partitioning, you should evaluate some of the other options.

Multi-Master Read/Write Scalability and Partitioning: NDB Cluster

Oracle offers two options for MySQL. One uses InnoDB as the de facto storage engine. The other uses the NDB storage engine and NDB Cluster. NDB was originally part of Alzato, an Ericsson-based venture startup focused on delivering telco-grade MySQL that MySQL acquired in 2003 for its clustering technology. Oracle has continued to sell it for specific scale-out use cases.

NDB is designed to deliver real-time performance at scale of up to 128TB for each NDB engine. Unlike InnoDB, NDB does support in-memory tables with some data optionally on disk, along with data durability for both. It provides horizontal scale using replication along with automatic partitioning that is transparent to applications. Changes are propagated using synchronous replication within a cluster, and MySQL Replication for asynchronous replication between clusters. It is designed to provide five nines (99.999%) availability. Node recovery and failover is automatic and typically recovers within one second.

But there are some limitations, even if you are not concerned about the two times cost increase compared to Enterprise Edition. The biggest limitation is that it does not provide multi-version concurrency control (MVCC) for concurrent transactional access. NDB only supports read committed, not repeatable read or serializable. If you need or expect higher levels of transactional support—which is often the case—you will need to look elsewhere.

Horizontal Read and Write Scalability: Galera Cluster

In 2007, Codership released Galera Cluster, a synchronous multi-master database cluster that supports synchronous replication for MySQL. Today it is used both for MySQL and MariaDB, and uses InnoDB as the storage engine. In multi-master replication, every node in the cluster has the entire data set and can handle reads and writes.

Galera Cluster has several strengths over MySQL Group Replication, including stronger split brain prevention, coordinated flow control, and stronger WAN support. It also has a few challenges. For one, while its synchronous replication is designed to be synchronous without using two phase commit (2PC) and is a very elegant solution, transactions can fail

if the replication fails. Like InnoDB Cluster, it also provides multi-master read/write support with a shared-something architecture that limits its horizontal scalability. The only way to get linear horizontal scalability is partitioning. Galera Cluster has similar limitations to InnoDB Cluster, such as issues with foreign keys. However, you can manually partition and have nodes recover and rejoin the cluster automatically. If this works, it is one reason to choose Galera (or Percona XtraDB Cluster) over MySQL Group Replication.

If you need greater scalability, you need to consider some of the options below that don't use InnoDB Cluster or Galera Cluster.

MariaDB

MariaDB started as a fork of MySQL the day Oracle's acquisition of Sun closed in 2010. With part of the original MySQL development team, MariaDB focused on adding new features. Today, while a lot of the core technology continues to be the same as MySQL and use both InnoDB and Galera Cluster, MariaDB has been diverging from MySQL in some areas. This means there are workloads where MariaDB will perform better than MySQL or Percona. But you may encounter some incompatibilities as a result.

MariaDB offers an integration architecture of:

- MariaDB for OLTP.
- MariaDB ColumnStore for OLAP.
- MariaDB MaxScale, both for routing and for Change Data Capture (CDC) from the MariaDB OLTP to MariaDB OLAP instances.

Then add Galera Cluster on top of this.

Just like MySQL and Percona Server for MySQL, if you are looking to MariaDB to scale MySQL, you are choosing the whole stack. This is not a major issue when compared to MySQL and InnoDB Cluster. For example, MariaDB recently decided to only support InnoDB and drop XtraDB support. While XtraDB has historically been a step ahead as a storage engine in comparison to InnoDB, this difference has become mostly about performance and scalability. MariaDB also includes its own version of Galera Cluster for its read/write scalability. In general, having the vendor perform the integration of the different components is a good thing.

One key difference is better OLAP performance with ColumnStore. MariaDB is the only one of the three distributions that offers integrated columnar support, a feature that has already appeared in several commercial databases including Oracle Database. ColumnStore is a great option to consider OLAP-related query needs, and could be enough. However, in some benchmarks it has not performed as well as dedicated columnar-centric disk-based or in-memory columnar databases. Make sure to perform benchmarks around your use cases before you make a decision.

MariaDB has some of the same limitations as the MySQL stack. But if some of the additional features of MariaDB help, including columnar support, and manually partitioning to achieve scale is not a problem, then MariaDB may be a great choice. Galera Cluster can at least manage partitioned nodes, even if it cannot partition or rebalance.

If a use case requires automated partitioning that is transparent to the application the options below might be a better fit.

Distributed Read Scalability: Percona Server for MySQL and XtraDB Cluster

You can think of Percona Server for MySQL as a free version of MySQL Enterprise Edition. It is a drop-in replacement for MySQL Community or Enterprise that does not require application changes or schema redesign. Percona focuses on adding speed and scale to MySQL through patches and extra features. While several MySQL enterprise features are only available in Oracle's Enterprise version, they are offered as part of the free and open source Percona Server for MySQL. Examples include I/O optimization, thread pools, authentication and logging plug-in support, improvements in table locking for backups, and visibility for monitoring and troubleshooting. Percona also merges improvements from MariaDB or other vendors. There are a lot of improvements that are easy to find on the Percona web site in the Percona Feature Comparison.

Percona XtraDB Cluster is a full solution with XtraDB and ProxySQL. Generally available since 2012, it is considered a stable solution. XtraDB is basically an enhanced version of InnoDB. XtraDB Cluster is based on Galera, starting when Galera became open source.

In short, just as with MariaDB, people choose Percona for its unique features, especially its performance and scalability features, over MySQL and MariaDB. But it still has the same eventual scalability limitations as MariaDB. It can manage partitioned nodes well, but does not do the partitioning or rebalancing. XtraDB Cluster also only really supports optimistic transactions. It does not support XA transactions across the cluster.

Distributed MySQL: Vitess

Vitess was originally started in 2010 within YouTube to solve its MySQL scalability issues, especially around sharding. It was as one of the early technologies to adopt Kubernetes.

Since then, it been adopted at Slack, Square, HubSpot, and a host of other software/SaaS vendors. PlanetScale, founded in 2018, uses Vitess as the foundation for its commercially supported version on-premise and in the cloud. In 2018, Vitess also became a hosted project at the Cloud Native Computing Foundation (CNCF).

Vitess has many strengths. It is well suited for Kubernetes-based architectures, including microservices. It supports sharded materialized views, which is the ability to shard using multiple sharding keys. Vitess also provides transparent sharding, where the application does not need to know that sharding is happening. It is based on a single master for each shard that then replicates out to other nodes. Unlike multi-master coordination on transactions, this helps ensure horizontal linear scale.

But in the grand scheme of MySQL, Vitess is relatively young technology. As a new technology, watch out for any issues such as:

- Vitess supports distributed transactions all the way up to a two-phase-commit (2PC) that will roll back. But it does NOT support isolation, so intermediate results can show up in queries, or "dirty reads." Also, if you have written SQL with "side effects" caused by stored procedures, foreign keys, or triggers, you can end up with inconsistent data.
- Early adopters like Slack say Vitess needed work to fit their needs, which were different from YouTube's needs. Make sure to fully test out any solution before implementing it.
- Make sure to investigate the list of unsupported queries, which is available in the FAQ section of the Vitess User Guides.
- Vitess does not provide out-of-the-box master management for identifying or changing a master. Orchestrator is the recommended add-on option.
- Vitess does not provide out-of-the-box monitoring, and does not appear to be supported by common tools for MySQL monitoring such as SeveralNines. It can be a challenge to determine how to monitor Vitess and MySQL together.

Distributed MySQL: Clustrix

Clustrix is a true, shared-nothing distributed version of MySQL. It does not share any code with MySQL, but is a drop-in replacement for MySQL. To be fair to Clustrix, it is hard to say anything bad about it, other than it is only commercially available and not open source. It is truly distributed with automatic sharding. It enforces MVCC with read committed and repeatable read. While it also uses serializable internally to do data movements across the cluster, serializable is not available to developers. In addition, Clustrix uses two-phase commit pessimistic locking, not optimistic.

Clustrix was acquired by MariaDB in 2018, which means it is in good hands as a MySQL database. This is a great option for horizontal, linear scalability for MySQL. The other may be NDB. You can also evaluate other distributed disk-based databases like Google Spanner, or in-memory databases like Oracle TimesTen, GridGain/Apache Ignite, and CockroachDB.

Distributed MySQL: MySQL as a Service

Most companies do not evaluate MySQL as a service to see whether they can get increased performance and scalability. The main reason to compare MySQL as a service to on-premise versions is to see whether it is possible to move applications to the cloud and avoid maintaining related database infrastructure as an internal resource.

If you have already made the decision to migrate your applications that rely on MySQL to the cloud, there are several great options available to you. Often companies choose a strategic cloud vendor first, and then evaluate their database options. The good news is AWS, Azure, and Google Cloud Platform all have reasonably good implementations of MySQL that have many of the scalability features such as read replication for scale, and simplify management. AWS provides two options; Amazon RDS for MySQL and Aurora (which is MySQL and PostgreSQL compatible). Aurora generally gives you better performance, though there are some exceptions such as high write workloads with secondary indexing. But Aurora also costs you a little more (as much as 20% more, by some estimates).

But none of these cloud services improve performance or scalability. These managed services for the most part implement best practices with the latest hardware. MySQL still faces the 64TB limit per database instance. While they all use replication for read scale, none of the services use sharding to increase write scale.

ADDING SPEED AND SCALE WITH IN-MEMORY COMPUTING

The other main option for adding speed and scale is to use in-memory computing. The end goal of in-memory computing is to move data into memory for speed, and to use a combination of a shared nothing architecture and MPP for linear, horizontal scale for all data-intensive workloads.

HTAP needs both existing data in relational databases and new data, such as streaming data from Web interactions or devices, or social data that helps it understand customer preferences and relationships.

The most common first step is the use of in-memory computing as an IMDG to existing applications, for two reasons. First, an IMDG adds in-memory speed and horizontal scalability that is more cost-effective in the longer term than scaling up with expensive hardware.

Do the math. Add up all the expected read and write scalability needs for the next 3-5 years. Then figure out your long term options. Most companies discover the following: they can either spend the money now on expensive hardware like Exadata, and then must implement an IMDG in the future or add the IMDG now and slowly grow it to the same size in the future assuming no other uses.

Second, an IMDG unlocks existing data for new uses, such as for real-time analytics, HTAP, streaming analytics or machine and deep learning. To support all these projects requires other capabilities, namely:

- IMDB support for storing and managing new types of data alongside existing data.
- Streaming analytics support, including integration with other streaming technologies like Apache Kafka and Spark.
- Machine and deep learning support.

Again, do the math. Identify the projects that can be achieved with existing data accessible in memory and add up the ROI over those 3-5 years. That is money lost without an IMDG as part of a broader in-memory computing platform. The ROI on those additional projects should be added to help decide between different in-memory computing technologies and other options.

HOW AN IMDG ADDS SPEED AND SCALE, AND UNLOCKS DATA

An IMDG adds speed and scale by sitting in-between applications and databases, in the path of all reads and writes. It stores all data in-memory and keeps the data up to date by supporting a read-through/write-through cache pattern. It receives all writes, writes to memory, and then passes it on to the database as a transaction. If the database transaction succeeds, the IMDG commits to memory as well. Since this keeps all data in the IMDG in sync with the database, the IMDG can handle all reads directly. This lowers latency for reads because the data is accessed directly from RAM, not a disk-based database. An IMDG also adds scale by offloading all read workloads from the database. Most IMDGs can scale horizontally on commoditized hardware to handle increased read loads without putting additional loads on the database. This is much less expensive than buying specialized database hardware.

The easiest way to slide an IMDG in-between an application and MySQL is for the IMDG to support SQL. If it does not support SQL, then you must write new code that replaces the SQL with a key-value API, and more code for the IMDG to access MySQL. Be sure to look for MySQL support.

Most IMDGs also provide some form of massively parallel processing (MPP) where they divide up data into smaller sets across nodes and collocate code with the data (like Hadoop). MPP allows horizontal scalability of both the data and computing, like the way MapReduce or Spark work. If the data is partitioned so that the computing has all the data it needs on each node, then the data does not need to be fetched over the network. This approach helps eliminate one of the most common performance bottlenecks in big data analytics and general big data computing, the network.

A MySQL database does not support MPP. If moving data over the network is part of the performance issue, scaling MySQL will not solve the problem. Also, part of the reason for adding an IMDG is to unlock data that is in MySQL, to be able to use the data in new projects, including HTAP, without overloading MySQL or requiring a hardware upgrade. MySQL, and a database in general, does not support real-time ana-

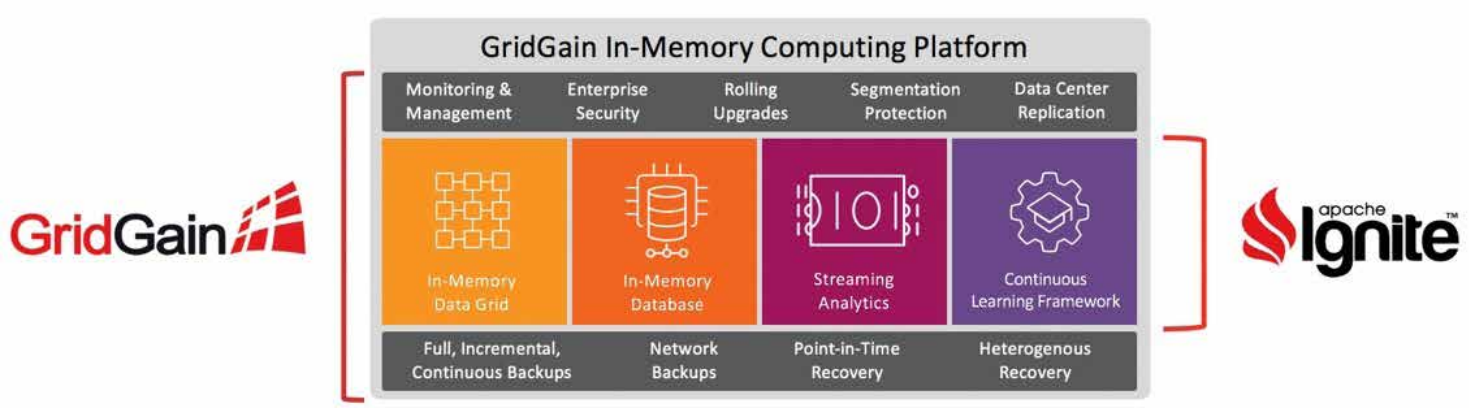


Figure 1. Apache Ignite and the GridGain In-Memory Computing Platform

lytics or high performance computing at scale. It doesn't support Spark or other streaming analytics technologies. It does not support general-purpose machine or deep learning. All of these rely on MPP.

APACHE IGNITE AND THE GRIDGAIN IN-MEMORY COMPUTING PLATFORM

GridGain is the leading in-memory computing platform for real-time business. It is the only enterprise-grade, commercially supported version of the Apache® Ignite™ (Ignite) open source project. GridGain includes enterprise-grade security, deployment, management, and monitoring capabilities which are not in Ignite, plus global support and services for business-critical systems. GridGain Systems contributed the code that became Ignite to the Apache Software Foundation and continues to be the project's lead contributor.

GridGain and Ignite are used by tens of thousands of companies worldwide to add in-memory speed and unlimited horizontal scalability to existing applications, and then add HTAP to support new initiatives to improve the customer experience and business outcomes. With GridGain, companies have:

- Improved speed and scalability by sliding GridGain in-between existing applications and databases as an IMDG with no rip-and-replace of the applications or databases.
- Improved transactional throughput and data ingestion by leveraging GridGain as a distributed IMDB.
- Improved the customer experience or business outcomes by adding HTAP that leverages real-time analytics, streaming analytics and continuous learning.

GridGain customers have been able to create a new shared in-memory data foundation. This single system of record for transactions and analytics enables real-time visibility and action for their business. With each project, they have unlocked more information for use by other applications on a platform with real-time performance at peak loads and always-on availability. As a result, they have been able to develop new projects faster, be more flexible to change, and more responsive in ways that have improved their experiences and business outcomes.

ADDING SPEED AND SCALABILITY TO EXISTING APPLICATIONS WITH AN IMDG

One of the core GridGain capabilities and most common use cases is as an IMDG. GridGain can increase the performance

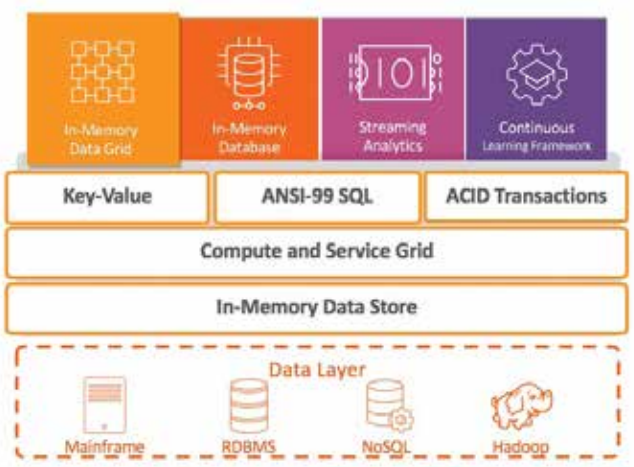


Figure 2. GridGain as an In-Memory Data Grid (IMDG)

and scalability of existing applications and databases by sliding in-between the application and data layer with no rip-and-replace of the database or application and without major architectural changes.

This is because GridGain supports ANSI-99 SQL and ACID transactions. GridGain can sit on top of leading RDBMSs including IBM DB2®, Microsoft SQL Server®, MySQL, Oracle® and PostgreSQL as well as NoSQL databases such as Apache Cassandra™ and MongoDB®. GridGain generates the application domain model based on the schema definition of the underlying database, loads the data, and then acts as the new data platform for the application. GridGain handles all reads and coordinates transactions with the underlying database in a way that ensures data consistency in the database and GridGain. By utilizing RAM in place of a disk-based database, GridGain lowers latency by orders of magnitude compared to traditional disk-based databases.

STORING DATA FOR HIGH VOLUME, LOW LATENCY TRANSACTIONS AND DATA INGESTION WITH AN IMDB

A GridGain cluster can also be used as a distributed, transactional IMDB to support high volume, low latency transactions and data ingestion, or for low cost storage.

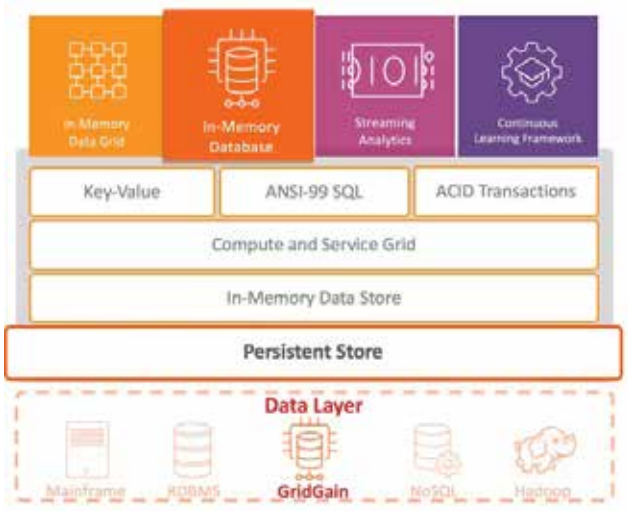


Figure 3. GridGain as an IMDB

The GridGain IMDB combines distributed, horizontally scalable ANSI-99 SQL and ACID transactions with the GridGain Persistent Store. It supports all SQL, DDL and DML commands including SELECT, UPDATE, INSERT, MERGE and DELETE queries and CREATE and DROP table. GridGain parallelizes commands whenever possible, such as distributed SQL joins. It allows for cross-cache joins across the entire cluster, which includes joins between data persisted in third party databases and the GridGain Persistent Store. It also allows companies to put 0-100% of data in RAM for the best combination of performance and cost.

The in-memory distributed SQL capabilities allow developers, administrators and analysts to interact with the GridGain platform using standard SQL commands through JDBC or ODBC or natively developed APIs across other languages as well.

ADDING REAL-TIME ANALYTICS AND HTAP WITH MASSIVELY PARALLEL PROCESSING (MPP)

Once GridGain is put in place, all the data stored in existing databases or in GridGain is now available in memory for any other use. Additional workloads are easily supported by GridGain with unlimited linear horizontal scalability for real-time analytics and HTAP.

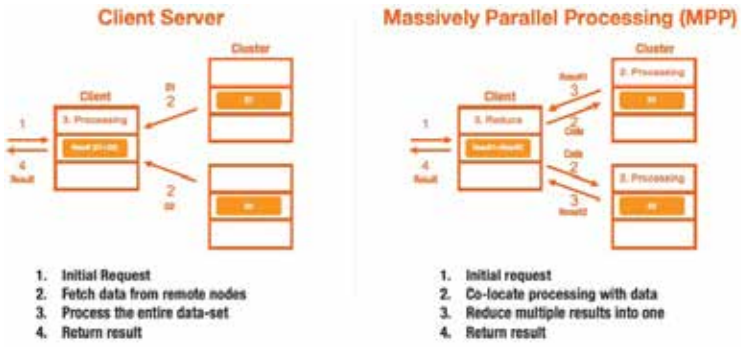


Figure 4. GridGain Compute Grid – Client Server vs Collocated Processing (MPP)

GridGain accomplishes this by implementing a general purpose in-memory compute grid for massively parallel processing (MPP). GridGain optimizes overall performance by distributing data across a cluster of nodes and acting as a compute grid that sends the processing to the data. This collocates data and processing across the cluster. Collocation enables parallel, in-memory processing of CPU-intensive or other resource-intensive tasks without having to fetch data over the network.

The GridGain Compute Grid is a general purpose framework that developers can use to add their own computations for any combination of transactions, analytics, stream processing, or machine learning. Companies have used GridGain's MPP capabilities for traditional High-Performance Computing (HPC) applications as well as a host of real-time HTAP applications.

GridGain has implemented all its built-in computing on the GridGain Compute Grid, including GridGain distributed SQL as well as the GridGain Continuous Learning Framework

for machine and deep learning. Developers can write their own real-time analytics or processing in multiple languages, including Java, .NET and C++, and then deploy their code using the Compute Grid.

Collocation is driven by user-defined data affinity, such as declaring foreign keys in SQL DDL (data definition language) when defining schema. Collocation helps ensure all data needed for processing data on each node is stored locally either as the data master or copy. This helps eliminate the network as a bottleneck by removing the need to move large data sets over the network to applications or analytics.

ADDING DEEPER INSIGHTS AND AUTOMATION WITH STREAMING ANALYTICS AND CONTINUOUS LEARNING

The capabilities GridGain supports are not just limited to real-time analytics that support transactions. GridGain is also used by the largest companies in the world to improve the customer experiences or business outcomes using streaming analytics and machine and deep learning. These companies have been able to incrementally adopt these technologies using GridGain to ingest, process, store and publish streaming data for large-scale, mission critical business applications.

Figure 5. GridGain for Stream Ingestion, Processing and Analytics

GridGain is used by several of the largest banks in the world for trade processing, settlement and compliance. Telecommunications companies use it to deliver call services over telephone networks and the Internet. Retail and e-commerce vendors rely on it to deliver an improved real-time experience. And leading cloud infrastructure and SaaS vendors use it as the in-memory computing foundation of their offerings. Companies have been able to ingest and process streams with millions of events per second on a moderately-sized cluster.

GridGain is integrated and used with major streaming technologies including Apache Camel™, Kafka™, Spark™ and Storm™, Java Message Service (JMS) and MQTT to ingest, process and publish streaming data. Once loaded into the cluster, companies can leverage GridGain's built-in MPP-style libraries for concurrent data processing, including concurrent SQL queries and continuous learning. Clients can then subscribe to continuous queries which execute and identify important events as streams are processed.

GridGain also provides the broadest in-memory computing integration with Apache Spark. The integration includes native support for Spark DataFrames, a GridGain RDD API for reading in and writing data to GridGain as mutable Spark RDDs, optimized SQL, and an in-memory implementation of HDFS with the GridGain File System (GGFS). The integration allows Spark to:

- Access all the in-memory data in GridGain, not just data streams.
- Share data and state across all Spark jobs.
- Take advantage of all GridGain's in-memory processing including continuous learning to train models in near real-time to improve outcomes for in-process HTAP applications.

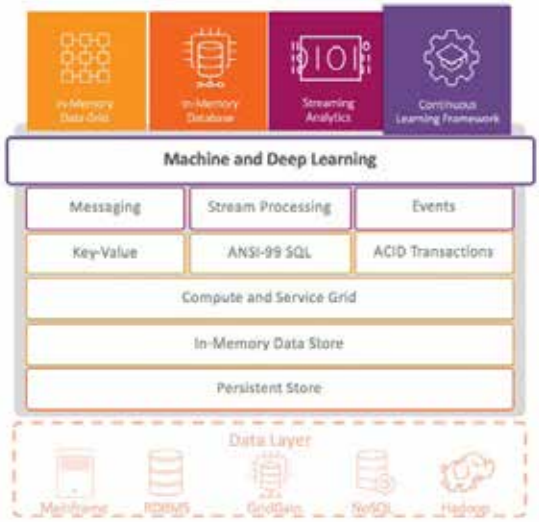


Figure 6. GridGain for Machine and Deep Learning

GridGain also provides the GridGain Continuous Learning Framework. It enables companies to automate decisions by adding machine and deep learning with real-time performance on petabytes of data. GridGain accomplishes this by

running machine and deep learning in RAM and in place on each machine without having to move data over the network.

GridGain provides several standard machine learning algorithms optimized for MPP-style processing including linear and multi-linear regression, k-means clustering, decision trees, k-NN classification and regression. It also includes a multilayer perceptron and TensorFlow integration for deep learning. Developers can develop and deploy their own algorithms across any cluster as well as using the compute grid. The result is continuous learning that can be incrementally retrained at any time against the latest data to improve every decision and outcome.

SUMMARY

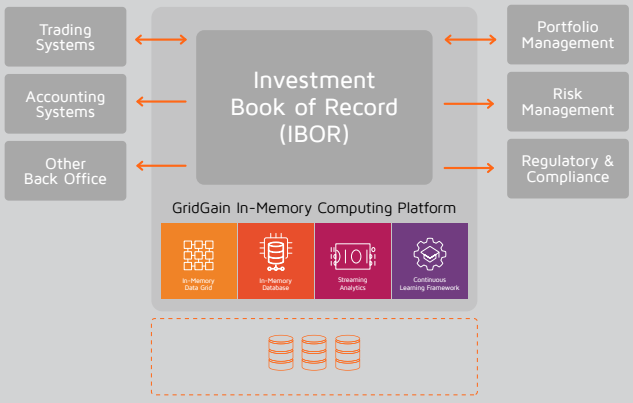
Applications and their underlying RDBMSs have been pushed beyond their architectural limits by new business needs and new software layers. Companies must add speed, scale, agility, and new capabilities to support digital transformation and other business critical initiatives. There are many options for adding speed and scale to MySQL–and each has its place. But when the speed and scale needs extend beyond what can be addressed at the database layer, the best long term approach is in-memory computing. Not only does it add speed and scale, but it unlocks data, enabling companies to be much more agile.

Adding Speed, Scalability and Analytics at Wellington Management

Wellington Management is one of the top 20 global asset management firms in the world, with more than $1 trillion in client assets under management.

Wellington had three major challenges:

1. Its current systems were no longer scalable due to an exploding growth of financial data. It needed horizontal scalability to handle the long-term growth.
2. The 2008 financial crisis resulted in a wave of new financial regulations that resulted in more complexity and risk in existing systems.
3. Many more new and complex asset classes have been introduced in the last few years based on customer demand, and there is a big need to release new asset classes faster.



Wellington's solution was to deploy its investment book of record (IBOR) on the GridGain in-memory computing platform. The Wellington IBOR serves as the single source of truth for investor positions, exposure, valuations, and performance. All trading transactions and account and back office activity flow through the IBOR in real time.

- **Horizontal Speed and Scalability**: Wellington's IBOR has unlimited horizontal scalability. It uses GridGain's SQL support to add speed and scalability by sliding in-between Oracle, its system of record, and the applications. The result is at least 10 times faster performance by adding in-memory computing on top of its Oracle database deployment.
- **Use of HTAP**: The IBOR is an HTAP system that is used by portfolio management teams for real-time position, market value, exposure, and performance analytics; by risk management teams for risk analytics and overall risk management; and by compliance teams to ensure, in real-time, that all regulatory requirements are met.

Why Wellington chose GridGain

- In-memory computing
- Horizontally scalable
- Supports distributed SQL
- ACID compliant (consistent data)
- Collocates data and computing
- Combines operational and analytical workflows (HTAP)

Contact GridGain Systems

To learn more about how GridGain can help your business, please email our sales team at sales@gridgain.com, call us at +1 (650) 241-2281 (US) or +44 (0)208 610 0666 (Europe), or complete our contact form at www.gridgain.com/contact and we will contact you.

About GridGain Systems

GridGain Systems is revolutionizing real-time data access and processing with the GridGain in-memory computing platform built on Apache® Ignite™. GridGain and Apache Ignite are used by tens of thousands of global enterprises in financial services, fintech, software, e-commerce, retail, online business services, healthcare, telecom and other major sectors, with a client list that includes ING, Raymond James, American Express, Societe Generale, Finastra, IHS Markit, ServiceNow, Marketo, RingCentral, American Airlines, Agilent, and UnitedHealthcare. GridGain delivers unprecedented speed and massive scalability to both legacy and greenfield applications. Deployed on a distributed cluster of commodity servers, GridGain software can reside between the application and data layers (RDBMS, NoSQL and Apache® Hadoop®), requiring no rip-and-replace of the existing databases, or it can be deployed as an in-memory transactional SQL database. GridGain is the most comprehensive in-memory computing platform for high-volume ACID transactions, real-time analytics, web-scale applications, continuous learning and hybrid transactional/analytical processing (HTAP). For more information on GridGain products and services, visit www.gridgain.com.