Minimizing downtime and maximizing elasticity with Galera Cluster for MySQL

www.codership.com/whitepapers

Executive Summary

Online services today face 3 requirements that are driving the choice of a database clustering solution: 1) The need for continuous operations and greater level of high availability than was traditionally needed, 2) ability to scale-out when service popularity grows - jumps of 20 times overnight increase or more are common, 3) ability to scale-in minimizes infrastructure costs when running on cloud technology.

Increasingly, the cost of downtime can be measured directly as lost revenue per second. Studies have estimated that businesses lose $5000 per minute, or $500,000 per outage due to service downtime. Inability to scale-out when needed will similarly incur lost business transactions - often at the very moment when there was a unique opportunity to grow the business!

Galera Cluster for MySQL is a multi-master, active-active clustering solution based on high performance synchronous replication. All database nodes hold the same data at all times to minimize data loss and failover downtime. Automated provisioning of new nodes makes scale-out and scale-in easy.

Table of Contents

The cost of downtime

For online businesses, be it e-commerce, trading and gaming or simply a per-click advertising revenue stream, revenue bearing transactions happen every second. Therefore downtime, whether planned or unplanned, now has a cost that can be directly measured in lost revenue - every second.

There used to exist a concept known as planned downtime. An internet or intranet service could have scheduled maintenance breaks, for example 3 AM at night. However, today's internet services increasingly target a global audience and therefore need to be online 24/7 - there simply is no good time to schedule even a planned break. The technology stack needs to support maintenance while having continuous operations.

As technology has improved to meet these demands, the bar on High Availability has also been raised just in general. Popular sites like Github[1] or Netflix may not strictly speaking lose revenue on a per-second basis, but their outages in 2012 were widely reported in online media and discussed in social media. While a single disaster could happen to anyone, the associated publicity means that recurring outages are simply not acceptable to online services today.

Hence, the "number of nines" targeted for service uptime have steadily risen and today it is common for service to target 99.95 % to 99.999 % uptime goals. It should also be noted that for each component of a service, such as the database, the individual uptime has to be even better than what is targeted for the whole service.

Uptime percentile	Max downtime per year
90 %	36 days
99 %	3.65 days
99.5 %	1.83 days
99.9 %	8.76 hours
99.99%	52.56 minutes
99.999 %	5.26 minutes
99.9999 %	31.5 seconds

In fact, Netflix has a policy to offer their customers a credit for the time which the service was unavailable. For the 1 hour 40 minute lasting incident in 2012, the video-on-demand service therefore offered a credit of 3% to all of its customers. If all customers would redeem that credit, the total cost to Netflix would be over $4 million.[2] This way, the cost of an outage becomes quite significant after all.

A study by Ponemon Institute reports that **businesses lost on average $5,000 per minute of downtime, or $505,000 per incident.** Average incident duration was 90

1 GitHub DBAs were kind enough to post an unusually detailed report of their recent MySQL outage: https://github.com/blog/1261-github-availability-this-week

2 http://www.huffingtonpost.com/2011/03/24/netflix-outage-credit_n_840156.html

minutes. For some critical industries such as telecom and ecommerce, the cost of downtime was more than double the average.[3]

One online gaming site - who now is a Galera support customer - generates $600,000 / day in revenue. Their recent 90 minute outage, which was caused by a failed data center network re-configuration, therefore will have cost them approximately $37,000 in lost revenue.

Cloud and elasticity

The other challenge for global online services is the unreasonable requirements for scalability. It is not uncommon to see number of hits per second, ie number of users, number of visitors, number of database transactions, jump by 20 times or more just because the service got some publicity in a widely read online media, or simply due to spreading in social media or a mobile app store.

Services need technology that can cope with such jumps in load. If the load exceeds what the service can handle, users will again see the service as unavailable just the same as if an outage due to a hardware or software crash. This again means lost business, and that at the very moment when there was a unique opportunity to grow it! At the same time it is not economical to overprovision all of your services 20 times higher than current needs.

The cloud was born out of this need, and it is now possible to launch additional servers within minutes when increased service load demands it. This then has shifted focus onto the software stack: It used to be a negligible time to spend a day installing and configuring your database cluster, but now we need to be able to scale-out new database nodes in an automated way that matches what cloud technology has enabled the infrastructure layer to do.

Finally, it is not enought to be able to scale-out. To optimize infrastructure costs, it also becomes necessary to scale-in (ie scale-down) when load is lower. This is known as *elasticity*.

While a global service will have users at all hours, a majority of them might still come from America and/or Europe and a difference of 2 times or more can be seen between "day" and "night" wrt this majority. Similar fluctuation is often also evident for business days versus weekends - for instance for www.codership.com the number of hits on Mondays is 3 times higher than it is for Sundays.

The most extreme example is commonly seen in e-commerce and retail, where there might be more business transactions in December month alone than the rest of the year combined. In such a case implementing an elastic architecture could mean up to 90% savings in infrastructure cost!

While scale-out can be done manually if automation was not in place, it will be a painful and stressful experience and it is not something one will ever want to do on a daily, weekly or monthly basis. Instead, servers are left to the "if it works, don't touch it" management principle.

Scale-in therefore is only possible if the software stack supports simple, automated scale-out operations to begin with. After that, simple and online scale-in operations

3 http://www.eweek.com/c/a/IT-Infrastructure/Unplanned-IT-Downtime-Can-Cost-5K-Per-Minute-Report-549007/

also must be supported, which is not always the case with all database products either (see below).

Why High Availability and Cloud are more difficult for databases

In a modern software stack the web servers and application servers are usually *stateless*. They receive a HTTP request, they process it, and forget it.

This is a very good design for scale-out, because deploying new stateless nodes is usually quite simple: Boot a new server instance (in the cloud, if you're in a hurry), install the needed binary packages, apply a configuration file or two, start services and maybe re-configure the load balancer to be aware of them. This can be done in minutes even when done manually.

Databases on the other hand are not just the software packages, they are the servers where the data needs to go. For a large database just copying a snapshot of the data to the new servers may take hours - and this is on a good network. When implementing a multi-datacenter, multi-continent setup using a Fedex service is sometimes a viable option for deploying new nodes!

Yet it is not enough to just copy the data. The database keeps the state of the service, of each finished and ongoing customer transaction. In fact, the other components in the software stack are able to become stateless exactly because they have pushed this problem down to the database layer. Therefore, when adding new nodes to a database cluster, it is not just that the data needs to be made available on the new node, the process must precisely record the point where the snapshot was taken, and the replication technology used must be able to join the new node into the cluster at that very position.

As was already emphasized in the previous section, such deployment of new database nodes must be able to happen in an online, "hot", manner, without causing any service break.

And finally, it would be preferable if the database clustering technology employed does not cause too much performance overhead. After all, one motivation to implement a scale-out cluster is to save costs, but this purpose is then defeated if adding a node causes a 50% performance degradation, as can often happen.

The cloud has completely commoditized the hardware infrastructure, and many services use free-of-cost open source solutions for their software stack. The complexities of the database layer enumerated above explain why - even when businesses increasingly choose an open source database solution such as MySQL®[4] - many still prefer to invest part of their IT budget in a good database clustering solution, as well as the necessary support expertise to accompany it.

How databases have solved this problem so far

Relational databases have existed for over 30 years, and the oldest database clustering solutions still in use today have existed for more than half of that time. As explained above, database clustering is a hard problem and these solutions have

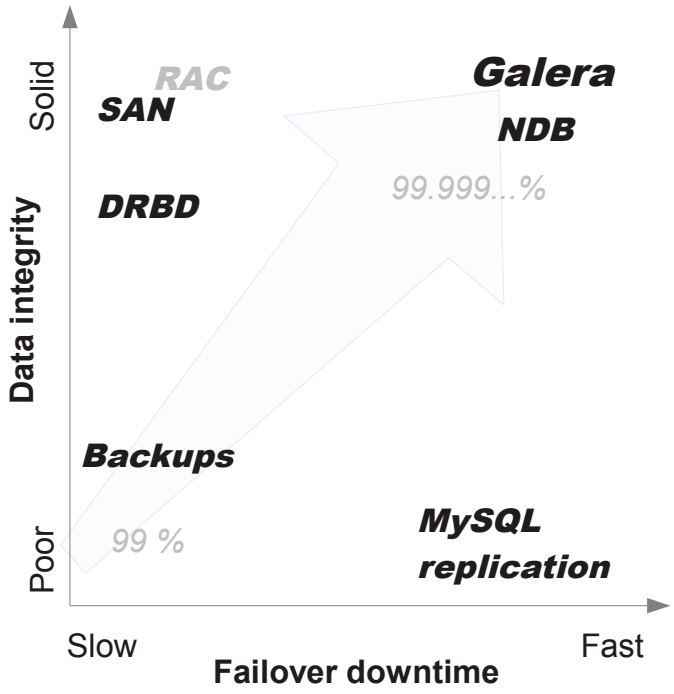
4 MySQL® is a product and registered trademark of Oracle America Inc.

served us well. However, with the increased demands for availability and scale-out, these tried and true solutions are increasingly at their limits.

Well known database clustering solutions typically have at least two of the following downsides: 1) risk of lost data due to asynchronous replication, 2) Up to 50% performance overhead, 3) downtime at planned and unplanned failovers may be up to several minutes, 4) poor compatibility compared to running with a non-clustered database, 5) prohibitively difficult to use, 6) not suitable for Widea Area Network replication between datacenters/continents, 7) prohibitively expensive to use.

Most of the available database clustering solutions either offer no scale-out benefit at all (active-passive) or can only scale read-only transactions (master-slave). In addition provisioning new servers into the cluster may require hours of manual work, and tying up well-paid DBA labor therefore defeats the whole point of cost savings by elastic scale-out and scale-down in the cloud.

The following picture illustrates well the trade-offs a database architect needs to consider when choosing a database clustering solution:



As can be seen from the picture above, the system architect is often presented a choice of solutions that have failover time on the order of minutes (DRBD and SAN, but even Oracle RAC falls into this category, even if it is an Active-Active solution when not doing failover) or solutions where failover is instant but experience has shown that data integrity is at risk. (MySQL built-in replication is by far the most popular solution today).

This is not the only trade-off though. Typically a HA solution also comes with a performance penalty. For example using DRBD will typically reduce maximum throughput by 50%. MySQL replication is single threaded, which can often result in slave lag that again constrains the maximum transaction rate that can be replicated.[5]

5 See for example this sysbench benchmark: http://openlife.cc/blogs/2011/may/drbd-and-semi-sync-shootout-large-server

Finally, none of these traditional solutions provide automated addition of new nodes (scale-out), rather the system architect is left to script those operations himself, or find some 3rd party solutions to do scale-out. Cold standby systems like DRBD provide no scale-out opportunity at all.

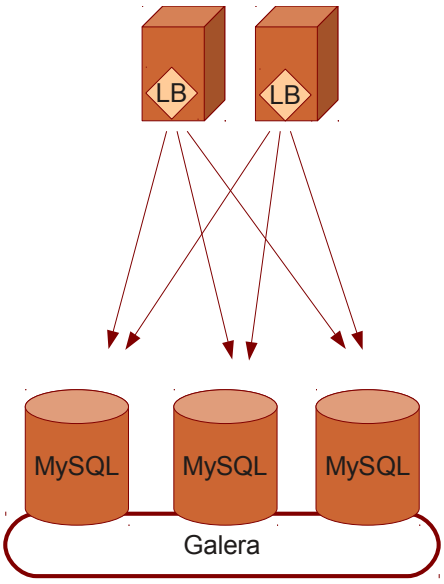
MySQL NDB Cluster is the only solution out there today that offers similar benefits than Galera Cluster for MySQL, but only a small amount of of those currently using the MySQL InnoDB engine can migrate to using NDB engine, as they are not 100% compatible and in fact exhibit very different performance characteristics. NDB engine since some releases back actually provides automated online addition of new nodes for scale-out, however the ability to remove nodes (scale-in) was never implemented and is not possible without service break.

Understanding the different trade-offs inherent in all of the above clustering solutions provide an important context for appreciating the benefits offered by Galera Cluster for MySQL:

Galera Cluster for MySQL

Coderhip's Galera Cluster for MySQL is a database clustering solution that is using synchronous replication to build a single cluster entity, a so called multi-master cluster, in which all servers have identical data each time.

The application can read and write to any server, which provides scale-out opportunity for both read and write transactions. Adding nodes to the cluster is completely automated, unlike most of the alternatives. Removing nodes (scale-in) is simply a matter of shutting down unwanted nodes.



Since the application developer no longer needs to implement cumbersome read-write splitting logic, the scale-out potential can be immediately realized without need to change the application logic, which saves time when developing applications and when migrating existing ones to Galera Cluster.

There is no delay in replicating the data and therefore Galera offers better security against data loss and inconsistent databases. If there is failure in one of cluster nodes - or the whole datacenter - the application/service does not notice anything but will continue serving the users using the other nodes - which may also be located in other data centers.

The replication implementation can use multiple slave threads, without limitations on the application or database schema that is used. This ensures good performance and benchmarks have shown that Galera does not cause any performance overhead, on the contrary it only increases performance when used as a scale-out solution.

Synchronous replication has proven quite vulnerable to network latency, and in practice none of MySQL 5.5 semi-sync replication, MySQL NDB Cluster or DRBD are usable in *Wide Area Network* setup, at least not across multiple continents. The implementation of synchronous replication in Galera Cluster for MySQL has however proven quite resilient to network conditions, and deploying a Galera Cluster across 2 or 3 continents is a popular choice. This is not only a good disaster recovery option, but a common strategy to minimize service response times towards the end user.

Galera Cluster for MySQL is a fully open source solution. It is integrated with the GPL version of MySQL and available from http://www.codership.com/downloads/download-mysqlgalera. This provides an easy-to-use experience as using Galera is not much different from using a single-server MySQL/InnoDB database. Many of our users also use Galera via products of our partners who have integrated Galera to provide a clustered version of their MySQL based forks: http://www.codership.com/partners/technology . Commercial 24/7 support and expert consulting is available from Codership and certified partners.

Conclusions

Galera Cluster for MySQL has proven a perfect match for the needs of modern online services:

- Popular database clustering solutions typically offer the system architect a choice of either risking his data integrity or risking down time due to long failover times. Galera synchronous replication guarantees that all nodes have an identical copy of the database, and being active-active there is zero failover time when a single node crashes.
- As a multi-master solution Galera is an ideal scale-out solution that can transparently load balance both read and write transactions. Adding nodes is a completely automated process.
- To maximize cost savings from using cloud infrastructure, the database cluster must support both scale-out and scale-in, this is known as *elasticity*. In Galera Cluster adding new nodes to the cluster is fully automated. Scale-in is possible simply by shutting down unwanted nodes.
- By using cloud services it is now easy to deploy a service in multiple continents. Galera replication works well also for Wide Area Network clustering, even across multiple continents.
- Galera Cluster is fully open source and integrated with the GPL version of MySQL.