severalnines

Introducing Severalnines ClusterControl™

The Virtual DBA Assistant for clustered databases

A Severalnines Business White Paper
March 2011

severalnines

Table of Contents

Introduction

The demand for highly available online services is ever increasing. At the same time, IT budgets are declining, or are flat at best. Relational databases have had a tough time coping with the new database requirements. Usually being single instance, and relying on scaling up capacity with high-end machines, it is of no surprise to see emerging and innovative database technologies in the NoSQL space. Many of the problems that NoSQL databases attempt to solve arise from a lack of money. Clustering of commodity servers is a very cost-effective way to deal with scale and availability of data.

Clustered environments, however, involve more effort and resources to administer than standalone systems. Clustered databases are often complex products, difficult to configure, deploy, manage, monitor and scale. Nodes on different machines, with each node possibly having a different role, myriads of configuration parameters, imply the need of experts to keep the system running.

Recognizing this, Severalnines has built ClusterControl™ to help organizations with clustered database platforms by automating development and deployment, as well as enabling management, monitoring and scaling via a graphical interface. Severalnines is today recognized globally by thousands of organizations, with its products being used in different applications in a number of industries:

- Online gaming and game state management
- Subscriber/User databases
- Telecom
- Finance applications (e.g ticker data)
- Session management
- Ad/click tracking
- E-commerce

These applications have in common:

- Minimal service interruption (99.999% availability or more) – if service is lost, then a lot of users are affected or a lot of money is lost
- Transactions volumes of tens or hundreds of thousands per second
- Online scaling to handle increased load and projected growth targets

ClusterControl™ currently supports clustered MySQL topologies based on MySQL Cluster and MySQL Replication resident either in the cloud or on premise. This whitepaper presents Severalnines ClusterControl™, a Virtual DBA that turns these MySQL topologies into a zero maintenance database infrastructure by simplifying Development, Deployment, Management, Monitoring and Scaling.

1.1 Total Database Lifecycle

Investing in a clustered database system is an important decision that will have a major effect on a business. By the time a developer or DBA is up and running with a prototype, the greatest temptation is to jump forward to full implementation. This is a risky path, for it leaves out important decisions that might affect the application at a later point. Making adjustments in the evaluation stage or development stage is far easier and more cost effective than later, after full implementation, or even worse after the database has been taken into production, and it is not able to scale with the growing demand. Staffing and downtime account for about 50% of database total cost of ownership (TCO)[1], and therefore, management and monitoring are very important factors.

As a result one has to take into consideration the total database lifecycle.

The database lifecycle would typically consist of different phases:

- Development
- Deployment
- Management
- Monitoring
- Scaling

The rest of this paper describes the different phases listed above, and also discusses how Severalnines ClusterControl™ for MySQL addresses each phase.

[1] IDC, Maximizing the Business Value of Enterprise Database Applications

2 Development Phase

2.1 Building a test cluster

During the development phase of a project, a database administrator will typically want to test the characteristics of the database to ensure it satisfies the requirements of the project (e.g. query response times, fail-over times, scale out by adding more nodes, geo redundancy, etc.). This will be done on dedicated servers, in order to have as accurate results as possible.

2.1.1 Challenges in building a test cluster

Getting a database cluster up and running, with the right configuration parameters, is not a trivial task, especially when it is the first time the administrator is exposed to the product. Having experience with other database products may certainly help to understand the concepts, but every database has its own specificities and constraints and one should not underestimate the learning curve at the beginning. It is very common for IT professionals to pick tools that they already are familiar with, and not necessary the best tools for the actual problem to be solved. This is a dangerous strategy, with respect to the current generation shift in building scalable and highly available platforms on COTS hardware.

2.1.2 Severalnines ClusterControl™ Configurator

To address these issues, Severalnines has created the ClusterControl™ Configurator[2], which automates and simplifies the creation of a cluster. It is available online as a wizard, asking a set of questions including the number of nodes and the hostnames of the servers, amount of data to be stored, expected type of workload, etc. It even has support for deployment in the cloud, e.g. Amazon EC2. Once it has gathered the appropriate information from the user, the Cluster Configurator generates the required scripts to download the database binaries, deploy them on the specified servers and automatically configure a database cluster.

[2] Available online at http://www.severalnines.com/config/

2.2 Database application testing

At the same time as a database administrator will be testing the characteristics of the database, an application developer or database developer will be writing the application that access the database. It is important to ensure the application is properly tuned. Performance degradation of the database is often due to inefficient code in the application. So however hard the database administrator tries to fine-tune the database parameters, performance may not improve unless the logic in the application is made more efficient.

2.2.1 Common mistakes when testing cluster database applications

The most common scenario observed among developers is to have a single instance of the database deployed on the development machine, with the application accessing all data in one process (e.g. MyISAM or InnoDB tables in MySQL). This is understandable, as a single database instance is much easier to install than a database cluster. However, it leads to bad query and schema design. For e.g., the SQL optimizer of the MySQL database server works and behaves differently for different MySQL Storage Engines. An algorithm that assumes locality of data in the MySQL Server might not be suitable when data is distributed in different processes, running on separate machines connected in a LAN environment.

Testing the application on a shared test cluster might not be feasible during the development project, as it would typically be subjected to a myriad of scalability and high availability tests. Also, different developers might need different data sets and table definitions, as they might have different code versions or are in different phases of development.

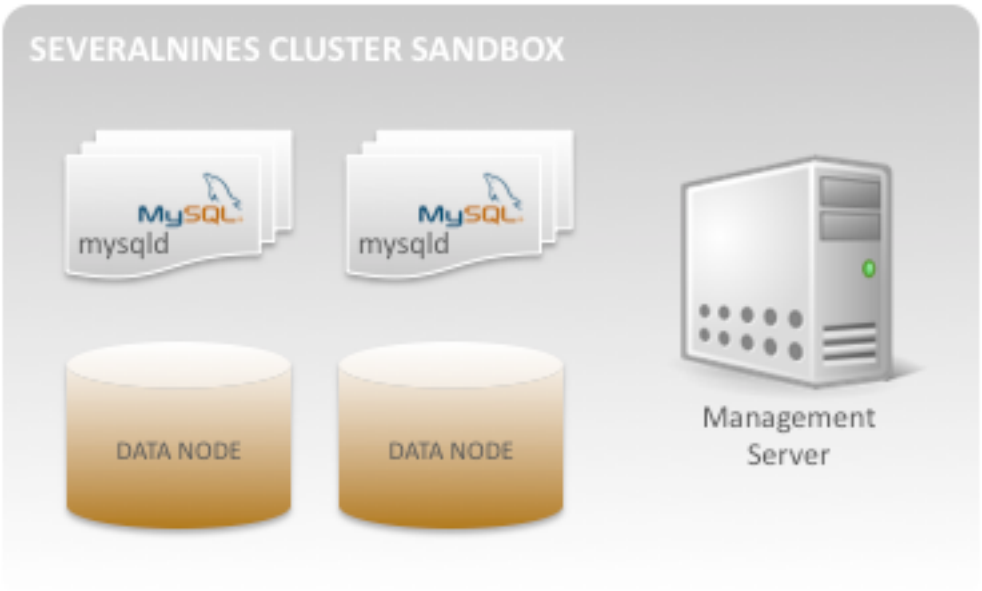
Severalnines supports the installation/configuration of a minimal MySQL Cluster consisting of at least two Data Nodes, so as to test queries on distributed data. Node failures can also be simulated, so as to verify retry code that handles different types of transaction failures.

2.2.2 Introducing the Cluster Sandbox for MySQL Cluster

The Cluster Sandbox[3] is a single package that installs in one directory within a few minutes. It is available on different Linux flavours, MacOSX and Solaris. It automatically downloads MySQL Cluster binaries and sets up a minimal MySQL Cluster environment on the development machine. The cluster consists of two SQL nodes, two Data Nodes and one Management Node. The environment is self-contained in one directory, and no root privileges are needed for the installation.

One can also setup master and slave Sandboxes, and test geo-replicated clusters (master-master, master-slave, etc.) on localhost.

The Sandbox can be customized to work with different MySQL Cluster versions, different configurations depending on how powerful the development machine is, and the number of CPU cores to be dedicated to the Sandbox.



[3] Available online at http://www.severalnines.com/sandbox/

3 Deployment

When the project is ready to go live, probably with a smaller subset of users, it is crucial to have a well-tested configuration. One should ensure the system is deployed so as bring the risk for downtime to a minimum. The system will be assembled into a release, and tested in a test and pre-production environment before being transferred into the production environment. Once the configuration files have been written, a four-node cluster needs at least about 40 commands to get deployed, which are different dependent upon the production environment.

However, a manual process increases the potential risk for human errors. To address these issues, Severalnines has created ClusterControl™ Deployer which automates the whole process of deploying to the pre-production and production environments. Automated procedures are accurate, consistent, and repeatable. However, even while having own homegrown scripts in the solution, there can be times when one cannot exactly remember what was changed and what needs to go out with the project deployment to the production database.

3.1 Common problems in Deployment

With all the steps involved in deploying a cluster, it is not unlikely to have something go wrong. Some examples are:

- User Permissions
- Lack of disk space
- Wrongly configured database resources (e.g., for MySQL Cluster, the Data Nodes start to swap or the user gets an OOM in the Linux Kernel)
- Not enough resources for multi-threaded environments (risk for contention)
- Older pre-installed versions of the cluster software
- Wrong PATH and software versions on the different nodes

Debugging can easily take hours or even days, if the user has not already given up. It is very common to find companies who have tried and failed to set up a cluster, and have instead settled for a sub-optimal solution.

3.2 Severalnines ClusterControl™ Deployer

The ClusterControl™ Deployer automates and simplifies the deployment of the database software on the specified nodes. It downloads the specified version of the database software and automatically deploys it on all the defined hosts in the cluster. It makes use of the information provided by the ClusterControl™ Configurator to set the right configuration parameters on the different types of nodes in the system. It has a verification phase that verifies the setup and that the cluster has been properly deployed.

As an example, below is a non-exhaustive list of tasks that are automated in the ClusterControl™ Deployer for MySQL Cluster:

- copy and install binaries on all servers
- update PATHs
- create datadirs
- create the mysql user for the MySQL Servers
- install system tables on MySQL Servers
- set permissions
- tune the OS
- start all cluster processes in the correct order
- verify the whole cluster has properly started
- in case of errors, inspect the logs on the different machines to find the root cause
- create database users
- align GRANTs, STORED PROCEDURES, TRIGGERS, VIEWS on all MySQL Servers

The deployment procedure follows best practices acquired over a decade of experience in deploying database clusters. The ClusterControl™ Deployer helps reduce project risk and avoid common pitfalls, thus saving time and money.

4 Management

Once the database has been deployed, a production DBA needs to be able to manage the overall database environment. A production DBA often faces unique and conflicting pressures, both from an operations perspective and an application service perspective. The DBA has to effectively manage resources, while maintaining the service experience of the users.

4.1 Common problems in Database Management

Managing clustered and distributed databases is more challenging than single-instance databases. It is generally recognized that cluster architectures, by their inherent complexity, require more management and administration. As processing is distributed between nodes, with different dependencies existing between nodes, there is usually an order in which tasks need to be performed. Timings of the tasks might depend on the state of the cluster or of an individual node.

Some examples of such challenges in MySQL Cluster are:

- Starting or restarting a cluster requires starting each node in turn, in the correct sequence
- Some types of Data Node failures require manual restart, e.g., the angel daemon dies
- A simple restart does not help if a Data Node's file-system is corrupted
- Configuration changes require rolling restarts to nodes
- Some parameters are dependent on other parameters, and therefore cannot be changed in isolation. E.g., the REDO log is dependent on Data Memory
- Some decisions require extensive inspection into different logs on the different nodes, e.g., how to restart failed Data Nodes in the correct order

4.2 Severalnines ClusterControl™ Manager

Severalnines ClusterControl™ Manager provides the ability to control the entire cluster as a single entity, while supporting granular control down to the individual nodes in the cluster. It helps administrators manage the overall database environment by providing facilities for availability management, backup and recovery, security management and change management.

Administrators are able to start, stop and restart a cluster with a single command.

The management tool automates online management operations, including:

- Reconfiguration of a running cluster
- Availability management, e.g., detecting all types of node failures, and automatically restarting each node in the right sequence
- Ability to re-initialize a database instance, e.g. if its file-system is corrupted
- Upgrade advisor
- Defragmentation utility
- Auto-scaling with addition of database nodes
- Sanity checking of configuration parameter changes
- Centralized backup/scheduling and restore management

Operational DBAs increase productivity by being able to manage problems with agility and accuracy. By resolving the root cause of problems faster, their skills can be applied to prevent problems, optimize resources and improve user response times.

5 Monitoring

Service interruptions directly impact revenue streams and customer satisfaction, and a solid monitoring infrastructure is a fundamental part of a mission-critical operation. The data tier is a highly centralized asset that needs to be monitored around the clock, as any database downtime has direct impact on service availability.

A single, consolidated view of database health is fundamental to addressing the challenges associated with managing and monitoring a database cluster. The coverage must extend across all nodes in the cluster. This ensures that the view presented to the DBA is accurate, and facilitates the user workflows that drive problem resolution and prevention, as well as performance optimization. Since each node in the cluster is also dependent on the operating system, and hardware resources (CPU and storage subsystem), coverage of these is necessary to accurately isolate the source of problems.

5.1 Requirements for Monitoring

When monitoring a clustered database, important requirements include:

- A view into the performance of every single node in the cluster, that could potentially disrupt database operations
- An interface that provides a centralized and unified view of the database cluster
- Visibility into the transaction workload of the database
- Visibility into CPU and storage subsystem and network resources
- Sufficient depth of information to support detailed analysis and optimization activities

The complexity of a database cluster requires far more sophisticated analysis of performance than is provided by the raw metrics of the database engine. DBAs need to fully understand performance in order to make good decisions and avoid creating new problems, which is always a possibility with inadequate data. Intelligent analysis of performance history requires real-time data as well as data that have aged by weeks or months. With sufficient historical data, the DBA can determine important trends, identify chronic conditions and prevent emerging issues.

5.2 Severalnines ClusterControl™ Monitor

Severalnines ClusterControl™ Monitor provides the ability to monitor one or multiple database clusters from a simple Web user interface, and supports granular information down to the individual nodes in the cluster. Statistics collection is lean and minimal, so as not to impact the performance of the database. Subagents running on the different nodes collect data around CPU utilization, network interconnect, available disk space and IO throughput.

The ClusterControl™ Monitor includes the following features:

- Node status information: it gathers all information that is possible to get from the different types of nodes in the cluster (e.g. for MySQL Cluster, node types include Data Nodes, MySQL Servers and Management Servers)
- Node information includes data memory, index memory, tablespace usage, REDO logs, REDO buffers, transaction counters, cluster events, status variables, configuration variables, etc.
- Historical information in graph format for any database status variables or counters in the database, as well as Operating System counters (CPU, Disk Activity, Disk Storage, RAM, etc.)
- Database activity, and query activity (e.g. MySQL process lists to detect long-running transactions or queries waiting for locks). This is available either in a global consolidated list or by database instance.
- CPU utilization, available disk space, IO throughput
- Backup statistics
- Centralized view of all error logs
- Concurrency control, with information on long running transactions
- Alarms if certain resource thresholds are reached, causing nodes failures
- Alarm the user of constantly failing/restarting nodes (‘ping pong’ behaviour)
- Adaptive Baseline Alerting to report on emerging problems as deviations from normal behaviour. Leverages historical performance to construct a baseline range of normal values for each collected metric.

Severalnines ClusterControl™ Monitor provides a consolidated view of database health, and can substantially improve resolution time and DBA productivity.

6 Conclusions

With business requirements growing faster than IT budgets, DBAs are being asked to do more, without compromising service levels. Therefore, enhancing the productivity of the DBA team is essential in maximizing cost effectiveness. Tools that automate tasks are often the first step in improving DBA productivity. For DBAs managing critical databases, productivity is measured by the number of databases they can manage while maintaining service levels.

Database developers and administrators have better things to do than piece together builds, follow checklists full of release commands, search the internet for sample configurations, copy files around on servers, and monitor running database processes. Fortunately, with the global competition for building scalable and always-on services heating up, automation can help DBAs and developers get more productive. Automated procedures are accurate, consistent, and repeatable, and automation reduces the need for documentation. Automation makes the job of an administrator easier, and critical database procedures can be performed as often as they should.

Severalnines ClusterControl™ builds upon the open source tools that have helped thousands of developers evaluate MySQL Cluster by automating configuration and deployment, and enabling management and monitoring via a graphical user interface delivered through a browser . It is many years of best practices and field expertise put in one package:

- *simplifies development*
- *generates* custom configurations
- *simplifies deployment*
- *automatically recovers* failed nodes
- *execute backup schedules*.
- *monitors* all internal database counters
- provide *statistics* on transaction load and resource usage
- *sends alarm notifications* in case of resource shortage or node failures
- *auto-scales* by adding nodes online

Severalnines ClusterControl™ integrates and consolidates the key tasks required to Develop, Deploy, Manage, Monitor and Scale a MySQL Cluster database. In addition to this, Severalnines provide expert consultancy services as well as training to get your system up and running on time and on budget. Our goal is to substantially improve productivity while containing risks.

7 Appendix – Supported MySQL Topologies

ClusterControl™ supports two different ways of clustering MySQL. It supports MySQL Cluster, which is an advanced shared-nothing clustered database with built-in partitioning and replication. It also supports topologies based on the standard MySQL Server (INNODB and MyISAM tables), which replicate using the MySQL Replication protocol.

The following sections describe both approaches.

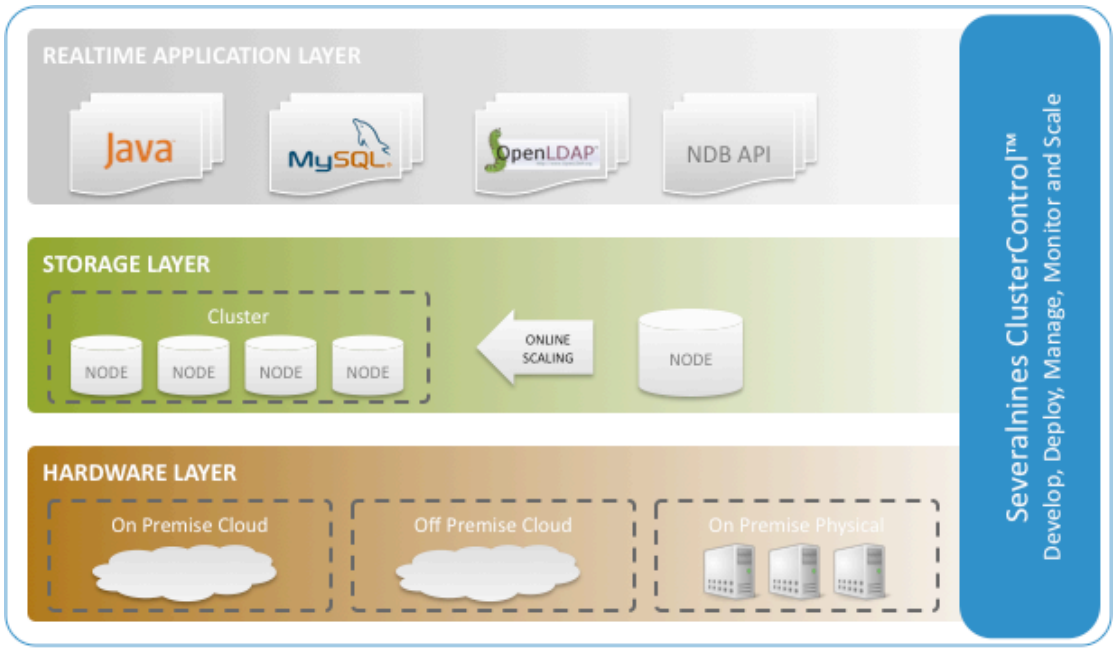
7.1 MySQL Cluster-based topology

MySQL Cluster attempt to solve the clustering problem, and provide horizontal scaling and access to distributed data via SQL or non-SQL interfaces (key row C++ API, object-oriented Java interface, LDAP, Web Services API, etc.). Data is stored in the NDBCLUSTER storage engine.

It has inbuilt partitioning, replication (both within a cluster and between clusters) and support for in-memory tables. In contrast to many NoSQL systems, it has no single point of failure (e.g., Hadoop) and supports strongly consistent data (e.g. Apache Cassandra, MongoDB). MySQL Cluster enables scalability without resorting to eventual consistency models, that leaves difficult conflict-resolution decisions to the programmer. It can be scaled by simply adding nodes online. It supports extremely high throughput low latency transactions for data that is stored in memory. It can also be scaled to store terabytes of data using disk based storage.

The storage layer consists of a number of data nodes that synchronously replicate data on two or more partitions using a Linear Two Phase Commit protocol. This means data nodes can fail and applications can still service as long as there is a copy of the data available.

The access layer can use either (or a mix of) SQL (from MySQL Servers), LDAP, or some of the direct APIs, such as Cluster/J (Java) or the NDBAPI (C++). It is all a matter of preference. So really, the data store can have a number of connectors, and SQL is just one of them.



The access layer can scale by adding, while online, more MySQL Servers, LDAP servers, Cluster/J nodes or NDB API nodes.

MySQL Cluster is not a great fit in case of:

- Joins that merges large data sets from multiple tables
- Foreign keys, currently not supported

7.2 MySQL Replication-based topology

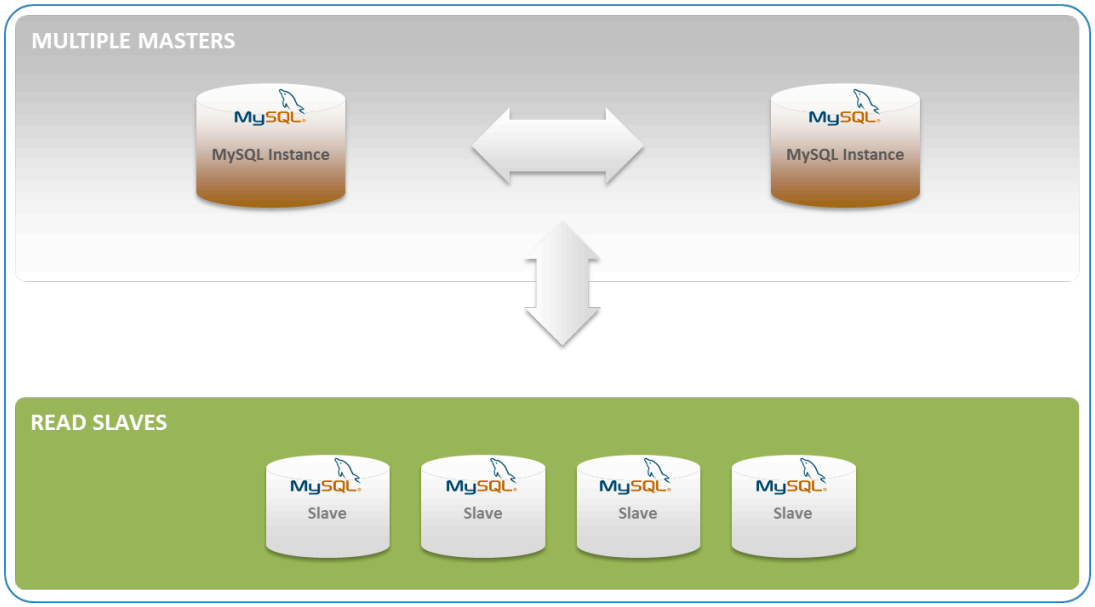
Replication enables data from one MySQL Server (the master) to be replicated to one or more MySQL database Servers (the slaves). It is a simple and robust mechanism.

The master keeps a log of all database updates that it has performed. Then, the slave(s) connect to the master, read each log entry, and perform the indicated update. The master server keeps track of housekeeping issues such as log rotation and access control. Each slave server has to maintain an idea of its current position within the server's transaction log. As new transactions occur on the server, they get logged on the master server and downloaded by each slave. Once the transaction has been committed by each slave, the slaves update their position in the server's transaction log and wait for the next transaction. This is all done asynchronously, which means that the master server does not have to wait for the slaves to catch up. It also means that if a slave is unable to connect to the master for a period of time, it can simply download all the pending transactions when connectivity is re-established.

The replication mechanism enables quite a few different server topologies to fit different needs, e.g. chains or rings of replicated database servers. However, things can get very complicated very quickly when a database instance fails and e.g. the ring or chain is broken. There is no support for fail-over. When the failed instance comes up again, there is no automatic mechanism for it to resynchronize and rejoin the ring.

Severalnines Cluster Deployer has support for the following topologies that rely on MySQL Replication:

- n stand-alone MySQL Servers
- n shards (each shard is a master/slave pair)
- multi-master + read slaves (one master/master pair with n slaves)



ClusterControl™ for MySQL has support for failure detection, and reconfigures the cluster to maintain availability (e.g., promoting a slave to a master, or redirecting slaves to a new master if the original master went down). ClusterControl™ will try to restart the failed node, and if not possible (e.g., because of a corrupted file-system), it will reinitialize the node. Once the node is up again, ClusterControl™ takes care of the resynchronization to re-introduce it into the cluster.

This multi-master + read slaves topology is not a great fit in case of:

- heavy write loads
- synchronous replication is required[4]

[4] ClusterControl™ supports the beta semi-synchronous Replication feature

About Severalnines

Severalnines provides software for fast, scalable and highly available cloud database platforms. This enables organizations to more efficiently build mission-critical clustered database environments, whether deployed on premise or in a cloud infrastructure, hence lowering capital expenditures as well as increasing productivity and innovation. Severalnines is recognized by thousands of organizations globally, with its products being used in different applications in a number of industries spanning from online gaming to telecom and finance.

Severalnines's flagship product, ClusterControl™, enable customers to Develop, Deploy, Manage, Monitor and Scale their clustered database platforms, free from the complexity and learning curves associated with database clusters.

Contacting Severalnines

Phone	+46 73 073 60 99
Email	sales@severalnines.com
Mail	Severalnines AB
	Skeppargatan 25 D
	114 52 Stockholm
	Sweden