目前数据库高可用方面有两个主流的架构：MMM架构和MHA架构。

# 高可用

## 宕机原因

一般来说，“运行环境”是排名第一的宕机类别，大约35%的事件属于这一类。运行环境可以看作是支持数据库服务器运行的系统和资源集合，包括操作系统、硬盘以及网络等。性能问题紧随其后，占大约35%，然后是复制，占20%，最后剩下的10%包含各种类型的数据丢失和损坏，以及其他问题。

以下问题需要注意：

1. 在运行环境的问题中，最普遍的问题时磁盘空间耗尽；
2. 在性能问题中，最普遍的宕机原因却是是运行很糟糕的SQL，但也不一定都是这个原因，比如也有很多问题时由于服务器bug或错误的行为导致的；
3. 糟糕的Schema和索引设计是第二大影响性能的问题；
4. 复制问题通常是由于主备数据不一致导致；
5. 数据丢失问题通常由于DROP TABEL的误操作导致，并总是伴随着缺少可用备份的问题。

## 实现高可用性

## 避免单点失效

## 故障转移和故障恢复

# MMM架构

MMM（Master-Master replication manager for MySQL）是一套支持双主故障切换和双主日常管理的脚本程序。MMM采用Perl语言开发，主要用来监控和管理MySQL Master-Master（双主）复制，虽然叫做双主复制，但是业务上同一时刻只允许对一个主进行写操作，另一台备选主上提供部分读服务，以加速在主主切换时备选主的预热。可以说MMM这套脚本程序一方面实现了故障切换的功能，另一方面其内部附加的工具脚本也可以实现多个slave的read负载均衡。

由于MMM无法完全地保证数据一致性，所以MMM适用于对数据一致性要求不是很高，但是又想最大程度的保证业务可用性的场景。对于那些对数据的一致性一致性要求很高的业务，非常不建议采用MMM这种高可用架构。

假设存在Master1（db1），Master2（db2），Slave（db3）,db1、db2、db3之间为一主两从的复制关系，一旦发生db2、db3延时于db1时，这个时刻db1 MySQL宕机，db3将会等待数据追上db1后，再重新指向新的主db2，进行change master to db2操作，在db1宕机的过程中，一旦db2落后于db1，这时发生切换，db2变成可写状态，数据的一致性将无法保证（即slave节点db3追上原来主节点db1，然后指向新的主节点db2，此时数据不一致）。

# MHA架构

MHA（Master High Availability）是一套优秀的作为MySQL高可用环境下故障切换和主从提升的高可用软件。在MySQL故障切换过程中，MHA能做到在0~30秒之内自动完成数据库的故障切换操作，并且在进行故障切换过程中，MHA能在最大程度上保证数据的一致性，以达到真正意义上的高可用。

它由两部分组成：MHA Manager（管理节点）和MHA Node（数据节点）。MHA Manager可以单独部署在一台独立的机器上管理多个master-slave集群，也可以部署在一台slave上。MHA Node运行在每台MySQL服务器上，MHA Manager会定时探测集群中的master节点，当master出现故障时，它可以自动将**最新数据**的slave提升为新的master（这样就避免数据不一致），然后将所有其他的slave重新指向新的master。整个故障转移过程对应用程序是完全透明的。

在MHA自动故障切换过程中，MHA试图从宕机的主服务器上保存二进制日志，最大程度地保证数据不丢失，但这并不总是可行的。例如，主服务器硬件故障或无法通过ssh访问，MHA无法保存二进制日志，只进行故障转移而丢失了最新数据。使用MySQL5.5的**半同步复制**，可以大大降低数据丢失的风险。如果只有一个slave已经收到了最新的二进制日志，MHA可以将最新的二进制日志应用于其他所有的slave服务器上，因此它们彼此保持一致性。

目前MHA主要支持一主多从的架构，要搭建MHA，要求一个复制集群中必须最少有三台数据库服务器，一主二从，即一台充当master，一台充当备用master，另一台充当slave。MHA的工作原理：

1. 从宕机崩溃的master保存二进制日志事件（binlog events）；
2. 识别含有最新更新的slave；
3. 应用差异的中继日志（relay log）到其他slave；
4. 应用从master保存的二进制日志事件（binlog events）；
5. 提升一个slave为新master；
6. 使其他的slave连接新的master进行复制。