文章编号:1000-5641(2014)05-0173-07

OceanBase 高可用方案

杨传辉

(阿里巴巴集团,北京 100020)

摘要: 传统关系数据库基于共享存储或者主备同步的方式实现高可用.这些方案要么依赖硬件的高可用,但成本高昂;要么无法同时满足强一致性和高可用性. OceanBase 将云计算和数据库两种技术有机地融合起来,实现了基于 Paxos 协议的高可用方案.该方案构建在普通服务器上,成本低廉,且同时满足强一致性和高可用性.

关键词: 共享存储; 主备同步; 高可用; 强一致; Paxos

中图分类号: TP31 文献标识码: A DOI:10.3969/j.issn.1000-5641.2014.05.015

High availability solution of OceanBase

YANG Chuan-hui

(*Alibaba Inc.*, *Beijing* 100020, *China*)

Abstract: Shared storage or master-slave replication is used in traditional RDBMS to achieve high availability. The first solution relies on high available hardware, and so are of high cost, while the second solution cannot meet the requirements of strong consistency and high availability concurrently. OceanBase combines cloud computing and database technology. Its high availability solution is based on Paxos protocol. This solution is built on top of commodity machine. It meets requirements of both strong consistency and high availability with low cost.

Key words: shared storage; master-slave replication; high availability; strong consistency; Paxos

0 引 言

数据库管理系统广泛应用于银行、信用卡交易、商品销售、航空与铁路运输、电信、电力系统、教育、医疗与健康、电子商务等各个领域.数据库管理系统已经成为订票、金融、电子商务等应用所不可缺失的核心信息系统[1].以“天猫双十一”购物节为例,开始抢购的 1 分钟内有 1 千万买家登录天猫系统.系统 1 个小时之内的成交金额达到 67 亿元,支付宝支付2 500万笔[2].以此推算,如果支撑交易的数据库系统出现 5 分钟不可用,将会损失成交金额约5.6亿元,并损害成百上千万用户的体验.因此,持续可用是数据库管理系统的核心目标.

收稿日期:2014-06

作者简介:杨传辉,男,阿里巴巴集团技术专家,研究方向为分布式数据库. E-mail: rizhao.ych@alipay.com.

传统数据库的高可用依赖于底层的高可用硬件,如高端服务器和高端存储等.虽然这些方案部分满足了高可用需求,然而它们成本高昂,且不可扩展,无法满足互联网业务应用的需要.互联网业务要求底层的数据库系统构建在普通的服务器上,并假设故障不可避免.这些故障可能是硬件故障,如硬盘故障;也可能是天灾,如地震、火灾等.

OceanBase[3]高可用方案构建在不可靠的主流 PC 服务器上,通过软件实现自动容错,使得系统内部故障对外部使用者不可见.这种方案成本低廉,且同时满足强一致性和高可用性.OceanBase 的高可用方案使得它能够被广泛应用在淘宝、天猫、支付宝等多个阿里巴巴集团核心业务的生产环境中.

1 传统数据库高可用方案

1.1 高可用集群

Oracle 的集群方案[4]如图 1 所示.多个数据库实例(Node1-Node3)通过高端的存储区域网(SAN,Storage Area Network)访问共享存储.数据库的数据文件和操作日志都需要持久化到共享存储.当某个数据库实例出现故障时,集群中的其它数据库实例能通过共享存储来恢复服务.可以看到,数据库的整体可用性取决于共享存储的可用性.

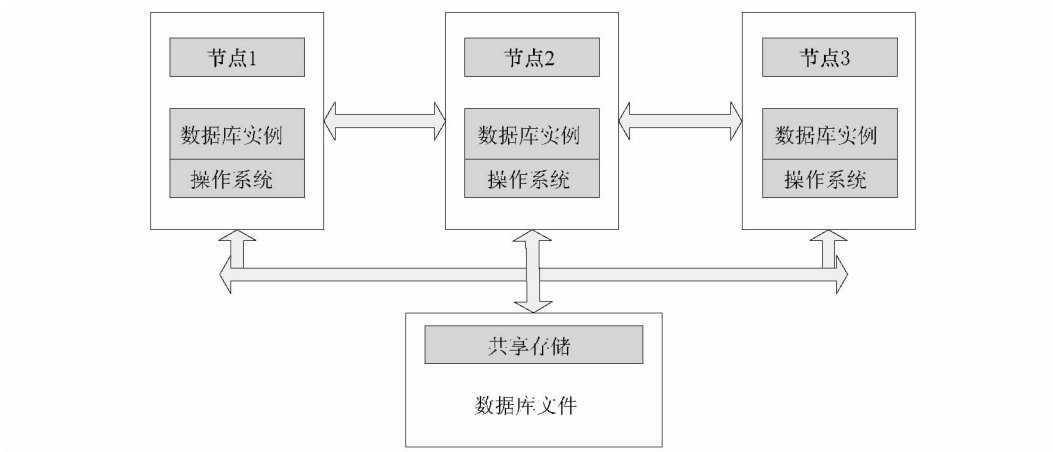

图 1 Oracle 集群方案

Fig. 1 Oracle real application clusters

为了保证性能,数据库实例和共享存储之间一般采用专用的 InfiniBand 网络,其带宽可达 40 Gbit/s,延迟为 1 μs.与之对比,普通的局域网之间的网络延迟一般为 250 μs 左右.因此,数据库实例和共享存储必须部署在同一个数据中心内,无法容忍单个数据中心整体故障.另外,InfiniBand 网络和共享存储价格高昂,且容量不够时很难扩展.

1.2 主备同步

另外一种常见的数据库高可用方案为主备同步[5].

如图 2 所示,数据库主机(Primary DataBase)将重做日志(Redo Log)同步到备机(Standby DataBase).主机提供服务,执行读写事务,备机可以执行不要求读到最新数据的读事务.当主机出现故障时,备机可以替换为主机继续提供服务.

主备同步往往分为 3 种模式:

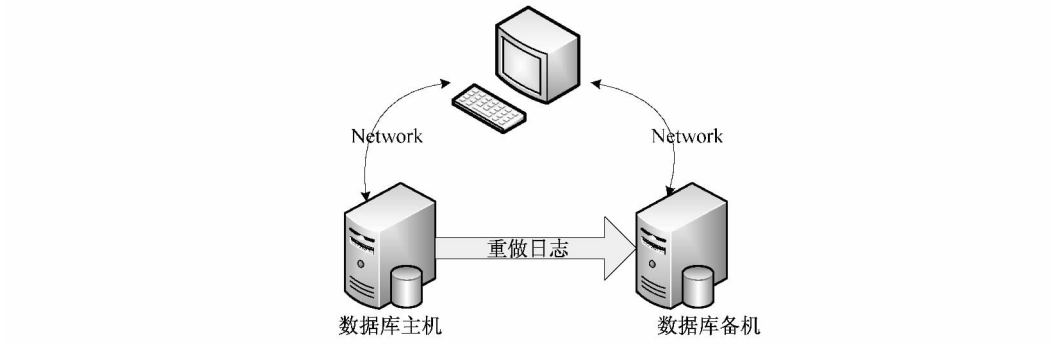

图 2　Oracle 主备同步

Fig. 2　Master-slave replication of Oracle

• 最大保护模式：任何一个事务必须首先同步到备机，接着在主机执行成功，才能应答客户端成功；

• 最大性能模式：事务只需要在主机执行成功，就可以应答客户端成功；

• 最大可用模式：当备机以及主备之间网络正常时，采用最大保护模式，即任何一个事务都需要首先同步到备机；当备机或者主备之间网络出现故障时，退化到最大性能模式，即只需要在主机执行成功.

假设采用最大保护模式，当备机或者主备之间网络出现故障时，无法提供服务，系统的可用性受到影响. 假设采用最大性能模式，当主机出现故障时，备机的数据往往落后于主机，如果将服务切到备机，可能丢失最后提交的一部分事务，出现数据不一致. 最大可用模式也有类似的问题，当主机出现时，备机无法知道是否和主机完全一致，如果要求强一致性，也无法将服务切到备机.

1.3　最高可用性

传统数据库中，可以将高可用集群和主备同步 2 种方案组合起来，实现最高可用性(MAA，Maximum Availability Architecture)体系结构[6]. 图 3 中有 2 个数据中心，IDC-1 为主，IDC-2 为备. IDC-1 和 IDC-2 内部通过共享存储实现高可用集群，IDC-1 和 IDC-2 之间通过主备同步的方式同步操作日志. 可以看到，只要 IDC-1 内部的共享存储不出现问题，系统都能正常服务. 当 IDC-1 内部的共享存储出现问题或者 IDC-1 整体故障，IDC-2 和 IDC-1 之间虽然可能不一致，不过相差不会太多，如果 IDC-1 永远无法恢复，可以忍受损失一部分数据的代价把 IDC-2 强制切为主后继续提供服务.

总而言之，即使通过高昂的成本实现了最高可用性，当整个数据中心或者共享存储出现故障时，都无法将服务无损地切换到备份的数据中心. 也就是说，只要保证强一致性，就无法做到数据中心容灾；只要实现数据中心容灾，就无法做到强一致性.

2　OceanBase 高可用方案

OceanBase 高可用方案构建在普通的 PC 服务器之上，假设硬件故障是常态，实现时使用了主备强同步策略和分布式选举协议. 当主机出现问题时，它保证至少有一个备机和主机完全同步，并通过分布式选举协议选出唯一的总控节点执行主备切换，既保证了强一致性，又实现了高可用性.

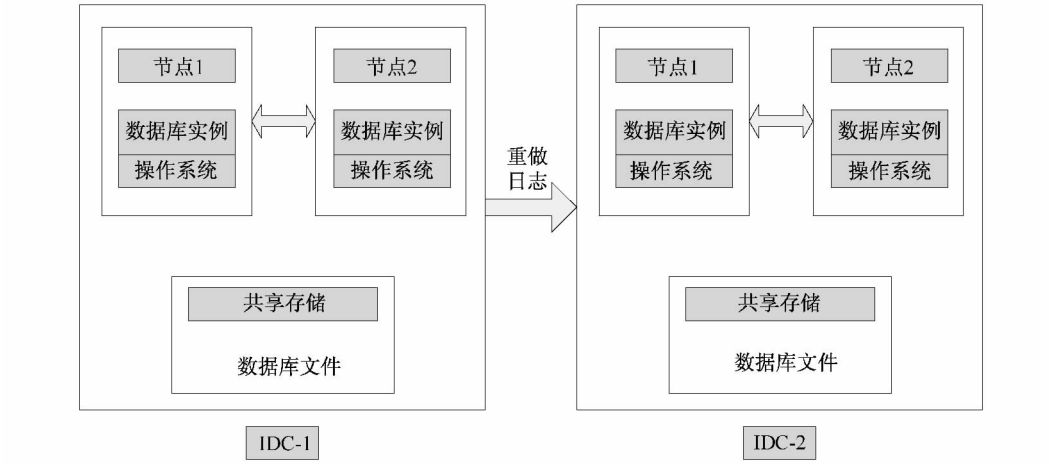

图 3 最高可用性体系结构

Fig. 3 The architecture of maximum availability

2.1 3 机房架构

如图 4,OceanBase 部署为 3 个集群,每个集群包含如下 4 种角色:

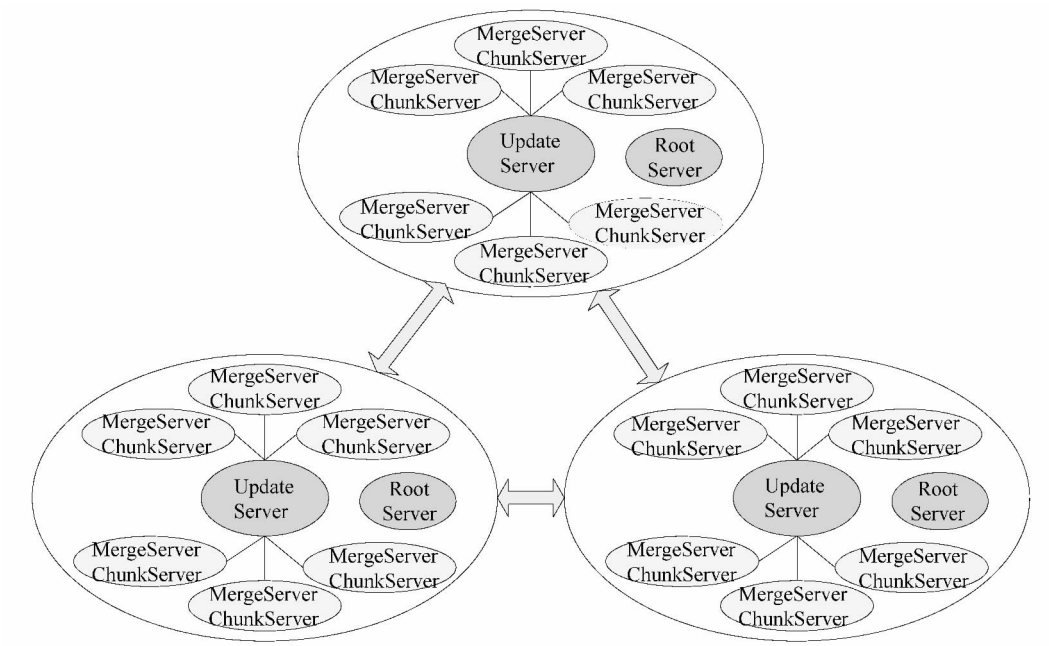

图 4 OceanBase 3 集群架构

Fig. 1 OceanBase architecture (triple clusters)

(1) MergeServer:OceanBase 的应用接口,支持 JDBC/ODBC 协议,负责进行 SQL 解析并生成执行计划.

(2) ChunkServer:保存存储于 sstable 的基线数据,并提供读访问. 为了防止由于 ChunkServer 故障导致服务不可用甚至数据丢失,每个 sstable 保存了多个副本并分布在不同的 ChunkServer 上.

（3）RootServer：OceanBase 的总控中心，负责 ChunkServer/MergeServer 的上线、下线管理以及 sstable 的负载均衡.

（4）UpdateServer：执行写事务、保存修改增量（到内存）、写重做日志（redo log）（到磁盘）并提供读服务.

其中，RootServer 以及 UpdateServer 和本文阐述的高可用性方案有关. UpdateServer 分为主 UpdateServer 和备 UpdateServer 2 种角色. 同一时刻，只有一个 UpdateServer 为主 UpdateServer. 每个写事务都需要在主 UpdateServer 执行成功并且同步到至少一台备 UpdateServer，才可以应答客户端. 当主 UpdateServer 出现故障时，至少有一台备 UpdateServer 和主 UpdateServer 完全同步，可以切换为主 UpdateServer 继续提供服务.

UpdateServer 主备切换操作由 RootServer 来执行，同一时刻只有一台 RootServer 为主机. 当主 RootServer 出现故障时，需要通过基于 Paxos 的分布式选举协议保证有且只有一台备 RootServer 被选为主 RootServer 继续提供服务，从而保证强一致性和高可用性.

2.2　主备强同步

UpdateServer 每次执行一个写事务时，都会生成一条操作日志，同步到备机并写入到本地磁盘，接着在内存中应用事务的修改操作. OceanBase 的强同步策略要求至少同步成功一台备机，才可以应答客户端.

如图 5，假设 DB1 为主机，DB2 和 DB3 为备机，DB1 包含 8 个事务，其中，DB2 已经收到前 6 个事务，DB3 只收到前 4 个事务. 由于前 6 个事务已经同步到 DB2，因此认为事务“已提交”，可以应答客户端；而后面 2 个事务没有同步到任何备机，处于“未决”状态. 当 DB1 出现故障时，OceanBase 会选择日志号最大的备机切换为主机. 假设 DB2 切换为主机，那么，前 6 个已经提交的事务将恢复，后面 2 个“未决”事务将被回滚；假设 DB1 再次参与到故障恢复过程并且重新选为主机，那么，后面 2 个“未决”事务将会被最终提交.

为了提高主备同步的性能，UpdateServer 实现了一些优化措施，包括：

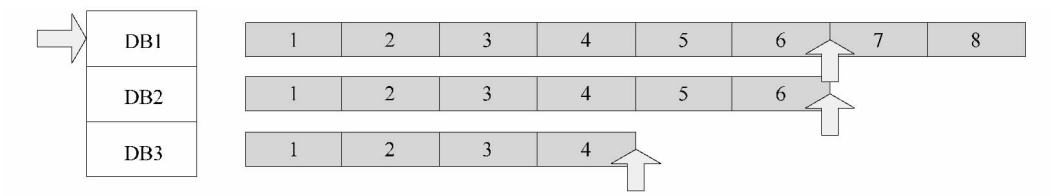

图 5　OceanBase 主备日志同步

Fig. 5　Master-slave replication in OceanBase

（1）成组提交（group commit）：每次执行写事务都需要将对应的操作日志写磁盘并同步到备机. 成组提交技术将正在并发执行的事务的操作日志打包到一个缓冲区中，一次性刷盘并同步备机，从而提升数据库系统的整体吞吐量.

（2）异步化：UpdateServer 执行写事务时，除了写磁盘，还需要同步到多个备机. 实现时将写磁盘以及同步备机操作完全异步化，从而降低事务延时.

（3）降低锁粒度：UpdateServer 提交事务时需要将并发执行的事务的操作日志拷贝到日志缓冲区并且保证顺序，当海量事务并发执行时，日志缓冲区的锁冲突非常严重. UpdateServer 实现时尽可能降低拷贝日志的锁粒度，在保证顺序的前提下，允许将多个并发事

务的操作日志同时拷贝到日志缓冲区.

2.3 分布式选举

OceanBase 包含多个 RootServer,同一个时刻只允许有一个 RootServer 作为主 RootServer 提供服务. 当主 RootServer 出现故障或者整个集群重新启动时,OceanBase 通过分布式选举协议选出唯一的主 RootServer.

分布式选举协议往往基于 Lamport 提出的 Paxos 协议[7]. Paxos 协议假设:

1. 成员不说假话(非拜占庭式).
2. 单个成员说话不自相矛盾.
3. 任何决议需要"多数派"同意.

假设有 $2N+1$ 个成员,那么,$N+1$ 个成员构成"多数派". 如果某个决议得到了"多数派"的同意,且任何一个成员说话不自相矛盾,那么,显然不会存在另外一个"多数派"同意其他的决议. 因此,可以证明协议的正确性. 然而,协议的可终止性无法得到证明,可能出现永远无法达成一致的情况. 另外,Paxos 协议,尤其是多轮 Paxos 协议(Multi Paxos)非常复杂,工程上更是难以实现. 目前已经有一些开源的工程实现对 Paxos 协议做了很大的简化,例如 Raft[8],而 OceanBase 则做得更加彻底.

OceanBase 将 Paxos 协议改造成如下的"同步模式".

如图 6,所有成员在同一时刻(T1)"同时"发起选举,执行步骤如下:

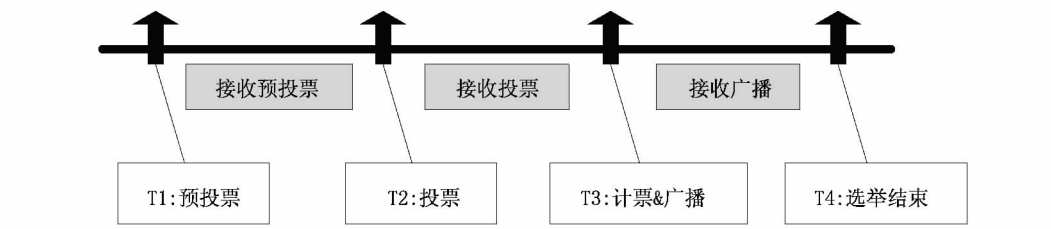

图 6 OceanBase 选举协议

Fig. 6 The protocol of OceanBase election

1. T1 时刻:预投票,即每个成员广播自己的"投票权重";
2. T2(>T1)时刻:投票,即每个成员向收到的"投票权重"最大者投票;
3. T3(>T2)时刻:计票并广播,即统计投票,得票过半者当选新主并向所有成员广播;
4. T4(>T3)时刻:选举结束,成员接收到新主当选的广播.

与 Paxos 协议类似,新主当选要求获得"多数派"同意,因此,最多选出一个新主,协议的正确性得到证明. 另外,由于所有成员"同时"发起选举,只要成员之间时钟偏差不会太大,且"多数派"工作正常,都能够选出新主. OceanBase 集群采用 ntp 协议来做时钟同步,大部分情况下成员之间的时钟偏差都在几毫秒之内,最坏情况下也不会超过 100 ms,从而保证选举协议是可终止的.

新主当选后,允许在租约有效期内提供服务. 当租约快到期时,主机可以发起"有主选举"来续约,使自己再次被选为主. 如果主机出现故障,那么,OceanBase 会在它的租约过期后发起新的分布式选举过程,其他成员可以被选为新主继续提供服务.

下面分析协议的选举时间. 假设成员之间的时钟偏差最大为 T_{diff},网络来回最大延迟为

T_{st}. 那么，某个成员在 T1 时刻发送预投票后，其他成员最晚在 $T1+2*T_{diff}+T_{st}$ 时刻收到预投票. 也就是说，T2 等于 $T1+2*T_{diff}+T_{st}$，每个成员收到所有其他在线成员的预投票请求后开始投票，保证不会错过任何一个成员的预投票请求. 同理，T3 等于 $T2+2*T_{diff}+T_{st}$，每个成员收到所有其他在线成员的投票请求后开始计票，保证不会错过任何一个成员的投票请求. T4 等于 $T3+2*T_{diff}+T_{st}$，每个成员收到所有其他在线成员的广播请求后选举过程结束. 综上所述，T4 等于 $T1+6*T_{diff}+3*T_{st}$，一次选举过程耗时 T_{elect} 等于 $6*T_{diff}+3*T_{st}$. 之前已经提到，ntp 协议保证时钟偏差最大不超过 100 ms，即 T_{diff} 等于 100 ms；而网络延迟 T_{st} 往往不超过 200 ms，可以算出 T_{elect} 等于 $6*T_{diff+}3*T_{st}=1\ 200$ms. 选举协议额外保留 200 ms，得到扩展的选举时间 T_{elect2} 等于 1 200 ms + 200 ms = 1 400 ms，即 1.4 s.

OceanBase 租约有效期为 4 倍选举时间，即 5.6 s；选举周期为 5 倍选举时间，即 7 s. 当主机出现故障，OceanBase 会等到它的租约过期后，在选举周期的整数倍时间点发起新的分布式选举过程. 因此，最坏情况下，从主机故障到选出新主的时间为 5.6 s(租约有效期) + 7 s(选举周期) + 1.2 s(选举时间)，即 13.8 s.

2.4　优缺点

Google Spanner 和 Raft 系统中都实现了基于 Paxos 协议的高可用方案. 其中，Spanner[9] 中采用原生的 Paxos 协议，Raft 对协议进行了简化. OceanBase 进一步将 Paxos 协议由“异步模式”改进为“同步模式”，工程实现上更加简单，且能够保证最长约 2 个选举周期就能选出新主. 无论是 Spanner 还是 Raft，都无法做到这一点. 然而，OceanBase 选举协议依赖时钟同步，要求多台服务器之间的时钟偏差不能过大，底层的 ntp 服务需要严格保证这一点.

3　结　　论

本文介绍了传统数据库的高可用方案，分析了其中的不足，并给出了 OceanBase 高可用方案以及其中的几个性能优化措施. OceanBase 高可用方案同时满足强一致性和高可用性，且成本低廉，是未来数据库高可用的趋势.

[参　考　文　献]

[1] RAMAKRISHNAN R, GEHRKE J. Database management systems[M]. 3rd ed. [s. l]: McGraw-Hill, 2003.

[2] 双十一战记[EB/OL]. http://ucwap.ifeng.com/tech/hulianwang/news?aid=73824780.

[3] OceanBase 开源[EB/OL]. http://alibaba.github.io/oceanbase/.

[4] GOPALAKRISHNAN K. Oracle database 11g oracle real application clusters handbook[M]. 2nd ed. [s. l]: Oracle Press, 2011.

[5] Oracle data guard[EB/OL]. http://docs.oracle.com/cd/B19306_01/server.102/b14239/concepts.htm.

[6] Oracle maximum availability architecture[EB/OL]. [2014-08-31]. http://www.oracle.com/technetwork/database/availability/maa-consolidation-2186395.pdf?ssSourceSiteId=ocomuk.

[7] LAMPORT L. The part-time parliament[J]. ACM TOCS, 1998, 16(2): 133-169.

[8] ONGARO D, OUSTERHOUT J. In search of an understandable consensus algorithm[C]//Proceedings of the 2014 USENIX Annual Technical Conference. 2014: 305-319.

[9] CORBETT J C, DEAN J, EPSTEIN M, et al. Spanner: Google's globally-distributed database[C]. OSDI'12, 2012: 251-264.

(责任编辑　林　磊)