Chubby分布式锁

2013年11月15日 00:58:44 继续微笑lsj 阅读数：3384

Google云计算技术具体包括：Google文件系统GFS、分布式计算编程模型MapReduce、分布式锁服务Chubby和分布式结构化数据存储系统Bigtable等。与之对应的有HDFS，MapReduce，Zookeeper，Hbase。

Chubby是Google设计的提供粗粒度锁服务的一个文件系统，它基于松耦合分布式系统，解决了分布的一致性问题。这种锁只是一种建议性的锁（Advisory Lock）而不是强制性的锁（Mandatory Lock），如此选择的目的是使系统具有更大的灵活性。 Chubby是怎样实现这样的“锁”功能的？就是通过文件。**Chubby中的“锁”就是文件**，创建文件其实就是进行“加锁”操作，创建文件成功的那个server其实就是抢占到了“锁”。用户通过打开、关闭和读取文件，获取共享锁或者独占锁； 并且通过通信机制，向用户发送更新信息。

GFS使用Chubby来选取一个GFS主服务器，Bigtable使用Chubby指定一个主服务器并发现、控制与其相关的子表服务器。

Chubby系统

Chubby被划分成两个部分：客户端（client）和服务器端（Chubby cell），客户端和服务器端之间通过远程过程调用（RPC）来连接。在客户这一端每个客户应用程序都有一个Chubby程序库（Chubby Library），客户端的所有应用都是通过调用这个库中的相关函数来完成的。服务器一端称为Chubby Cell，一般是由五个称为副本（Replicas）的服务器组成的，这五个副本在配置上完全一致，并且在系统刚开始时处于对等地位。当 Chubby工作的时候，首先它需要从这些replicas中选举出一个master。Chubby是通过采consensus protocol（很可能就是Paxos算法）来解决这个问题的。

每个master都具有一定的期限，称为master lease。在这个期限中，副本们不会再选举一个其它的master。为 了安全性和容错的考虑，所有的replicas（包括master）都维护的同一个DB的拷贝。但是，只有master能够接受client提交的操作对 DB进行读和写，而其它的replicas只是和master进行通信来update它们各自的DB。所以，一旦一个master被选举出来后，所有的 client端都只和master进行通信。如果是读操作，那么master一台机器就搞定了，如果是写操作，master会通知其它的 replicas进行update。这样的话，一旦master意外停机，那么其它的replicas也能够很快的选举出另外一个master。

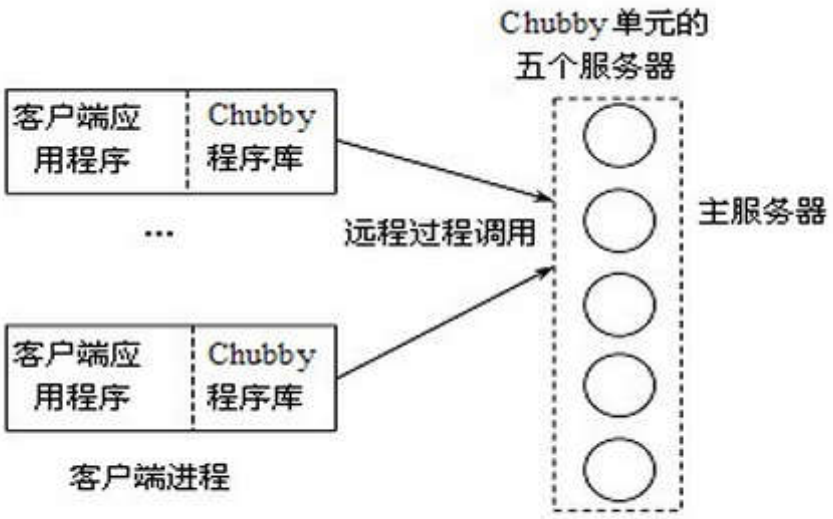


图 3 Chubby 的基本架构

Chubby文件系统

Chubby的底层实现其实就是一个分布式的文件系统。Chubby的文件系统由于它的特殊用途做了很多 的简化。例如它不支持文件的转移，不记录文件最后访问时间等等。整个文件系统只包含有文件和目录，统一称为“Node”。文件系统采用Berkeley DB来保存Node的信息，主要是一种map的关系。Key就是Node的名字，Value就是Node的内容。

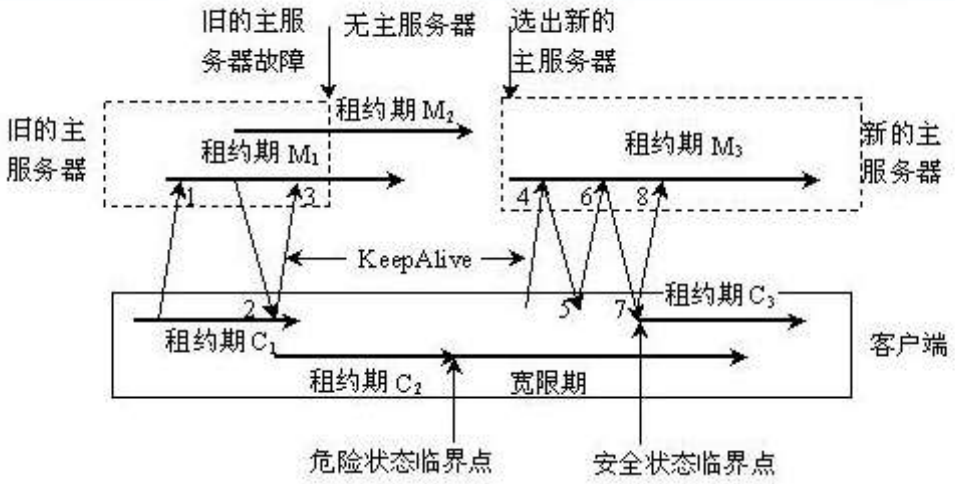
Chubby cell和client之间用了event形式的通知机制。client在创建了文件之后会得到一个handle，并且还向服务器可以订阅一系列的event，例 如文件内容修改的event。这样的话，一旦client相关的文件内容被修改了，那么cell会通过机制发送一个event来告诉client该文件被 修改了。

Chubby客户端与服务器交互

为 了降低client和cell之间通信的压力和频率，client在本地会保存一个和自己相关的Chubby文件的cache。例如如果client通过 Chubby library在cell上创建了一个文件，那么在client本地，也会有一个相同的文件在cache中创建，这个cache中的文件的内容和cell 上文件的内容是一样的。这样的话，client如果想访问这个文件，就可以直接访问本地的cache而不通过网络去访问cell。

cache有两个状态，有效和无效。当 有一个client要改变某个File的时候，整个修改会被master block，然后master会发送无效标志给其他cache存储了这个数据的client（它维护了这么一个表），当其它client端收到这个无效标志 后，就会将cache中的状态置为无效，然后返回一个acknowledge；当master确定收到了所有的acknowledge之后，才完成整个 modification。

对于KeepAlive协议（就是说服务器有可能会宕机挂掉，此时必须保证服务器和客户端定期保持联系），则是为了保证client和master随时都保持着联系。client和master每隔一段时间就会KeepAlive 一次，这样的话，如果master意外停机，client可以很快的知道这个消息，然后迅速的转移到新的master上。



参考

http://blog.csdn.net/historyasamirror/article/details/3870168

http://blog.csdn.net/xiaoyao3857/article/details/8148899

http://www.csdn.net/article/2010-05-31/267199