# 定义

什么是云硬盘呢？腾讯云的云硬盘叫 CBS，CBS 就是 Cloud Block Storage，基于云的分布式块存储服务。

云的概念听起来很玄乎，其实我理解的云的核心技术就是分布式，云是分布式的一个商业包装，通过云这个接口把它推出去，给大家提供就像水、电一样的基础服务，这是我给云的定义。 云里边最核心的是分布式技术，云硬盘也是基于分布式去实现的一个硬盘，跟电脑里面的硬盘功能相同，但是又比个人电脑里的硬盘有更丰富的能力。

比如基于分布式实现，性能上就使得横向拓展的能力非常好。个人电脑硬盘只能提供几百兆的吞吐能力，CBS 基于云的硬盘可以提供更大的存储性能，我们现在可以做到5GB，这是本地传统物理硬盘很难做到的。

 在可靠性方面，CBS 是基于多副本的存储策略，也就是说原则上会将用户数据存多份，一般是三份，这样的策略让数据的可靠性更强。

另外是可用性方面。传统的物理硬盘坏了，上面的数据就丢了，也就影响到了用户的服务。但是云硬盘不一样，它是基于分布式实现的，当这一块硬盘故障的时候可以快速进行隔离，因为是多副本存储数据的，另外的副本就可以及时的提供服务，不影响用户的可用性。

当前云硬盘服务主要的对象是大家在腾讯云上买的云主机，是给云主机提供持久化的块存储服务的。

以上这些就是我给 CBS 云硬盘下的定义，首先它是基于分布式技术实现的，就像个人电脑硬盘一样，硬盘可以提供的所有能力它都可以提供，并且在高性能、可靠性和可用性进行了增强，提供持久化的块存储服务。

# 演进



CBS 的演进经过了三代架构 —— CBS apllo、CBS atlas 和 HiSTOR。

## TBS

块存储最早是从 2011 年开始做的，当时不叫 CBS，而是叫 TBS。最早的一代架构是基于以前腾讯云架构平台部已有的存储平台打造了块存储的服务。但这个已有的存储平台一开始并不是为块这个场景专门设计的，所以存在很多问题。

## CBS Apllo

 2013 年、2014 年是云发展比较快的阶段，当 2014 年规模到了几个 PB 的时候，发现当时的架构不能承载用户的需求，所以在 2014 年时用 CBS apllo 及时替代了更老的一代架构，适应了云快速发展的需求，快速的把块存储的规模从几 PB 增加到了上百 PB，个人认为这是非常优秀的一代架构。 但是到 2016 年，虽然以前的架构很优秀，但是成本方面会略高一些、性能上也不尽如意，所以在2016年希望能够在成本和性能方面有所改进。

## CBS Atlas

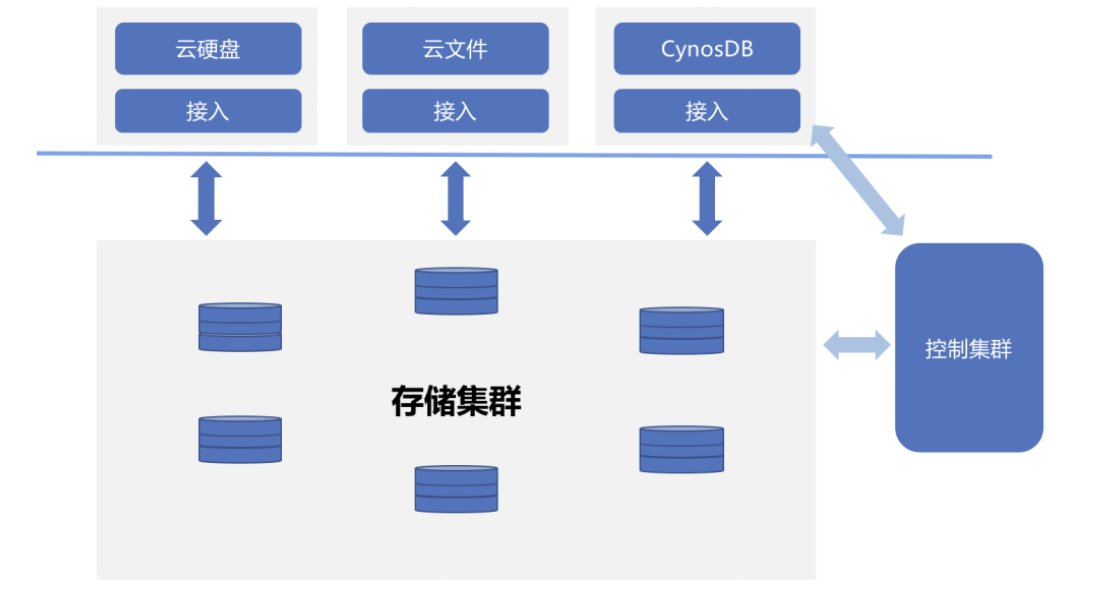
在 2016 年我们设计了新一代 CBS 架构 Atlas，2017 年 Atlas 正式上线接用户业务。 目前在腾讯云上跑的主流的架构是 CBS atlas，承载着数 EB 的数据规模。现在很多用户把数据放在云上，腾讯云上也已经衍生了几十亿、上百亿、甚至上千亿美金的独角兽公司出来，他们就是依赖于云去支撑他们的服务。 在现在这个阶段，就要全场景的看用户的需求，最主要的一块就是对性能的要求慢慢成为了矛盾的焦点。原先预计云盘很快便会取代本地盘，但在去年的时候发现取代的节奏在变缓，分析之后发现用户仍用本地盘是因为云盘某些场景没有服务好，典型的是关于 DB 的需求。 云盘是基于分布式的，中间通过网络互联，天生比本地盘的延时略高，但是到底高多少呢？如果高的程度能够在提供的优势可以覆盖住的情况下，用户就会愿意用 CBS 盘，但如果影响大于优势，那么用户就不接受了。

## HiSTOR

这时候这个矛盾就凸显出来了，要解决用户在性能上的需求，这时候“极速存储”就应运而生，它在性能上有了很大的提升。到今年上半年的时候平台侧就达到了我们认为能够满足用户需求的能力，今年 8 月产品层面推出相关产品。

以上就是三代架构以及前面那代架构演进的历程。我列了三个时间点，其中2014年是非常重要的时间点，云快速发展推出 applo 架构，2017 年推出 atlas 架构，到去年为了解决用户全面上云对性能的要求，我们投入精力搞了 HiSTOR 的架构。

# 架构



CBS 存储系统架构到底是怎么样的呢？最新的架构可以分成三块：

客户端

存储集群

控制集群

先介绍一下存储集群，存储系统很重要的一块就是存储集群，这是后端的存储给用户提供存储服务；另外给用户提供服务肯定要有个接入点，也就是前面的客户端；分布式系统要有对集群做控制的模块，也就是控制集群，当然控制模块本身也是多机的分布式系统；CBS存储系统可以分为这三块。

这里要讲的是块存储，其实现在 CBS 存储系统已经被打造成了统一的存储平台，现在腾讯云上购买的云文件也是这一套存储系统提供的服务，另外云原生的数据库，后端的存储系统也是它。 CBS 的存储系统 HiSTOR 的特点首先是极简的架构。多年前我也对外分享过一个“大道至简”的设计理念，用极简的架构满足用户的需求是最好的。

整个集群分为三部分，客户端直接访问存储，中间没有传统的存储系统的接入模块，所以是一种极简的架构。

另一个特点是统一存储。我们提供的不仅是块存储服务，而是多种存储类型服务共享的一个平台。CBS 存储系统为什么一直向前演进呢？因为要解决问题。

Atlas 取代 Applo 核心的目的是什么？首先是成本，因为架构层面足够的简单，只有两层，没有中间的接入集群、接入设备和其他的管理设备是没有的，所以在成本上是大大的降低了。

这套架构比之前的架构在成本上有优势，物理成本上节约了 50% 左右，但产品怎么能够让用户去买单呢？要让用户有买单的意愿就要做到成本足够低，成本问题是几年前解决用户上云的需求的主要矛盾。

慢慢的在用户上云了以后，包括核心的业务开始往云上切，腾讯云上已经孕育出了很多十亿、上百亿、上千亿美金的独角兽公司，这个时候如何让用户放心把业务放到云上呢？一定要让其用得放心，质量一定要过关，解决成本问题后我们投入大量的精力在质量上。

质量要过关，可靠性、可用性和数据安全方面一定要过关。

## 客户端

## 存储集群

## 控制集群

# 挑战

现在很多公司已经把它的全部业务都放到云上，这时候就需要把他服务好，让他用得爽。云盘相对本地盘主要的劣势就是延时相对本地盘比较高，所以这里主要分析怎么在这一点上有所突破。



性能方面有三个维度，除了一再强调的延时维度，还有吞吐维度和 IOPS 的维度。

## 吞吐维度

吞吐维度指的是带宽。现在很多大数据场景、AI 训练的场景对带宽的需求非常大，SSD 云盘提供的 260 兆的带宽，跟用户需求是有落差的，所以除了在延时上需要做到极致以外，带宽方面也要满足用户的需求。

## IOPS

以前都是本地盘间的比较，大家觉得 IOPS 不是个矛盾点，本地物理硬盘随机访问的IOPS 也就是在 200、300 的级别。而云盘产生之后，最老的一代产品能够提供的是几千的IOPS。

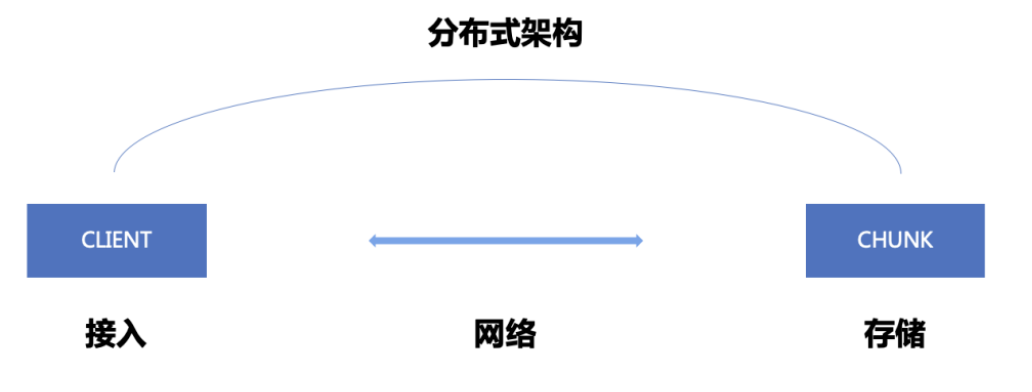
当时是可以满足需求，但慢慢的上云的场景越来越多后，云主机的母机上单个服务对 IOPS 的需求不高，但是加在一起需求就变得很大。

## 延时维度

延时维度最典型的是 DB 的业务。可能多年前放在云硬盘上的数据价值并不高，但是 DB 数据价值密度非常高，如何把 DB 应用场景服务好也是需要花很大的精力思考的问题。

# 优化

CBS延时优化：

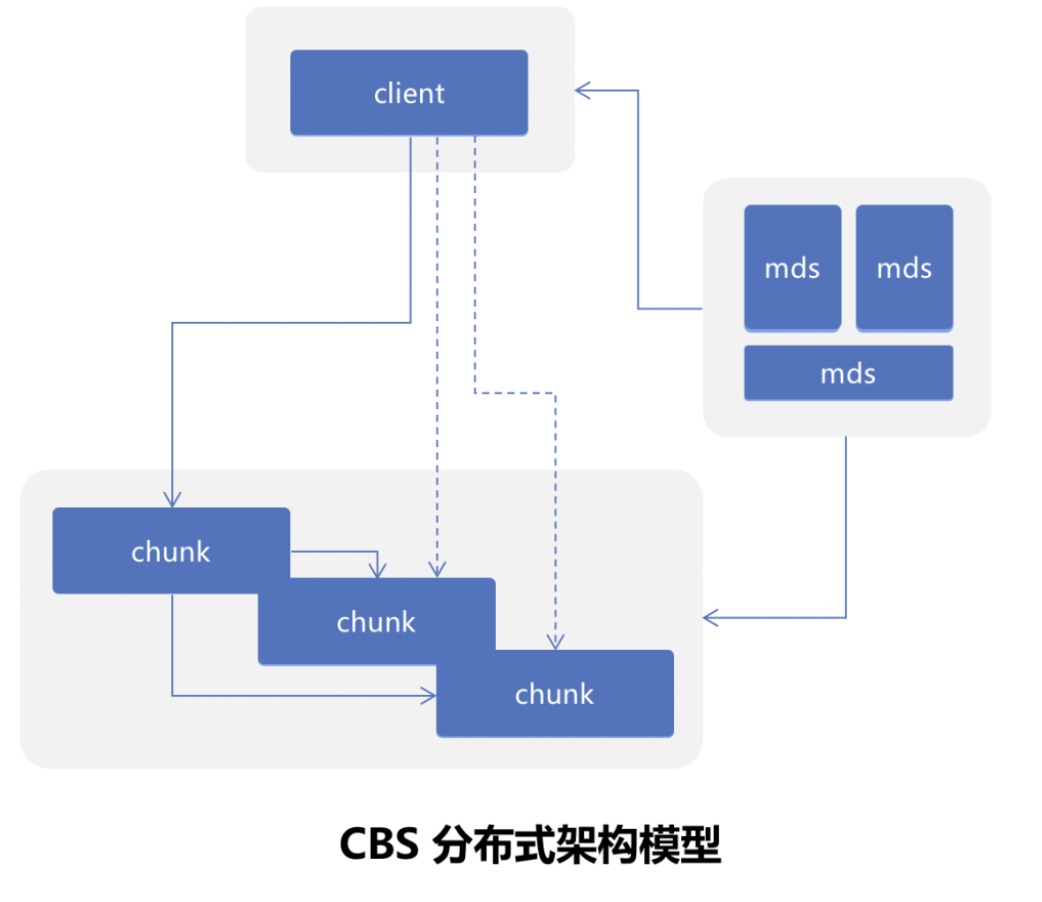


那么 CBS 延时如何优化呢？很多人提出过很多具体手段，但是总结起来就是两个方面，一是怎么把并发做上去，二是怎么把延时做下来。

把并发做上去，这是分布式存储天然有的优势，只是技术实现上的难度大小而已，但跟延时比起来并不是这么困难。

低延时无非是两种手段，首先用更快的硬件去解决，另外是怎么把软件 IO 栈做到尽量的扁平化，这里主要是基于软件的维度考虑延时优化，具体是从 CBS 的四个维度入手：分布式层面优化、接入端、存储侧和中间交互的网络的优化方面。

## 分布式架构



CBS 存储系统是极简的架构，客户端访问存储中间就是网络，涉及优化的也就是三个层面加一个架构，我们按这几点来进行分解。

首先从架构层面，CBS altas 直接从客户端访问存储，是两层分布式架构，没有传统的接入点也就没有了接入瓶颈。另外是客户端直接访问存储中间访问路径可以做到最短，这就是当前架构的优势。

说起来很简单，但做成两层架构实现起来技术难度很大。首先是路由管理上如何做？CBS 的存储系统用的路由方式是一致性哈希，这就涉及到哈希路由如何推送到客户端。

本身路由的变更是由控制集群维护的，路由更新后如果直接推送客户端的话看起来很简单，但面对云上的集群规模就会发现问题。腾讯目前是上百万台的规模，如果把一个信息推送到上百万个节点根本是不可行的事情。

所以在实现上我们想个了方式叫“惰性路由同步”，集群管理节点变更后不是直接推送客户端而是推送到存储节点，再由节点中转到客户端。

为什么这样可行呢？接入节点上百万台，但是存储节点比客户端节点是数量少得多，推送到存储节点对中心控制节点的负载是可以容忍的，所以先推送到存储节点，之后对中控系统来说工作就完成了，推送也就完成了。

惰性路由同步的惰性体现在哪里？起名字的时候我们借鉴了内核内存分配的方式，内核去申请内存的时候只是一个虚拟的，并没有把真实物理内存分配给你。

分布式架构里路由同步的策略也是类似的，先推送之后并不急于推送到客户端节点，如果有需求访问存储的时候，再同步把路由推送到客户端节点。这时候拿到最新的路由就可以访问到需要的数据了，使用的时候再去推送的策略便是“惰性路由同步”。

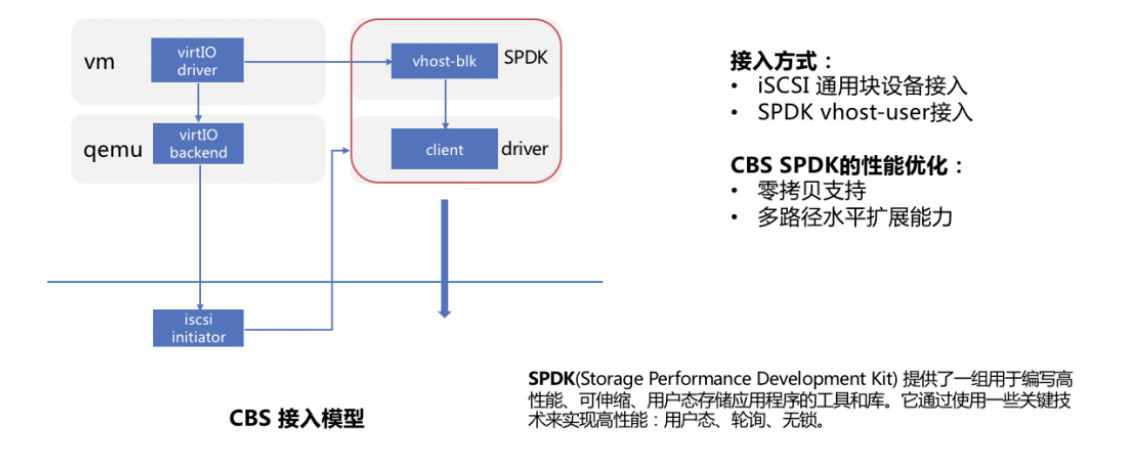
说明：这里没有采用分布式数据库的，客户端-元数据-存储这种链路，而是直接访问存储获取最新的路由，这个也是为了尽可能减少网络开销。

架构层面，原先的架构虽然说是两层，但因为是多副本存储，数据到了存储节点后还有一次复制的过程，需要再复制成多份数据存到节点上去，这时候虽然架构上是极简的，但写的时候有两次的路由，就相当于数据走了两跳。

为了做到延时上的极简，我们是根据 IO 大小做了不同的策略，如果是小 IO 的话，走的是以前的模式，直接客户端出三份，这样对小 IO 便只需要一跳。

但为什么大 IO 不这么做呢？客户端出流量的话，出三份其实是占用计算的网络流量的，所以这样是需要跟前端计算的同事协商的。但小的 IO 这么做，不需要增加前端的计算带宽，这就是为什么大小 IO 需要做不同策略的原因。

## CBS接入端



上图所示的客户端架构是一个主流的架构，接入方式是基于 iSCSI 的通用块设备接入，虚拟机通过访问存储的时候是 QEMU 访问一个主机上的块设备，往下走 SCSI 设备。

SCSI 的底层就是 iSCSI initiator，通过网络访问 CBS 的客户端，然后再访问后端存储。路径非常的长，首先要复制数据、切换访问内核，之后通过 iSCSI ，再通过网络访问 CBS 的客户端，客户端再通过网络把数据发出来，中间经过了多少次的上下文切换和数据拷贝，所以延时上非常不友好。

有没有其他的接入方式呢？第一次想这个问题的时候是在 2016 年，我们在设计架构时，当时想直接用共享内存的方式把数据旁路出来，就不用经过这么多次的来来回回的切换了。

后来老板们说你为什么这么做呢，英特尔搞了一个更好的东西，就是可以直接在虚拟机上把数据旁路出来，也就是 SPDK。IO 设备是虚拟机的话，SPDK 可以做 IO 的后端，直接把数据同步出来，再把客户端做到 SPDK 里面，这个数据路径是要短非常多。

虚拟机里面直接用数据共享的方式跟 SPDK 交互数据，SPDK 在后端访问存储，再经过网络处理，比起刚才说的来来回回好几次要简单得多了。

但是不是这样就做到极致了呢？最初做的方式是 SPDK 跟客户端的访问没有在一个线程里面，而是两个线程，之间通过共享内存。

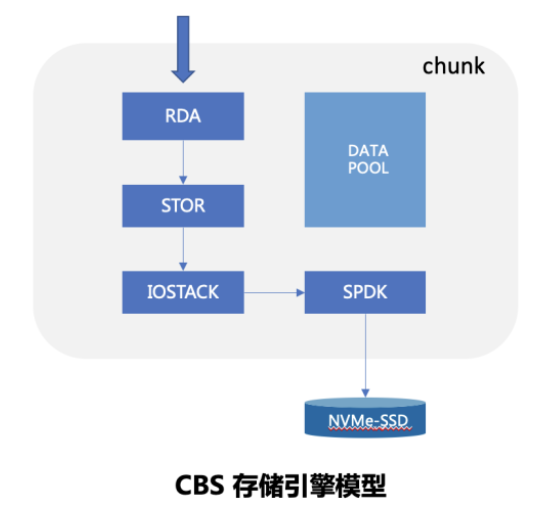
但这样对延时并不友好，因为经过了线程的一次切换，哪怕只是几个微秒的区别，但在一个超高性能场景中几微秒也是比较多的。所以我们又做了优化，把客户端跟 SPDK 放在同一个线程里做，这样就没有线程上下文的切换了。

总结下来，是用 SPDK 的接入方式取代了 iSCSI 的接入方式，用零拷贝的轮询方式取代了原来两线程数据共享的方式。最后是有一个水平扩展的能力，这跟延时无关。

刚才提到老板们站的高度更高，在 2016 年便提出了用 SPDK，那 SPDK 到底是什么呢？

这是一个英特尔用来做高性能存储的开发套件，提供一些 lib 库之类的工具，通过它做高性能存储的应用开发。具体技术实现首先是在用户态实现的，可以减少对内核的切换，用户态、内核间的上下文切换对 CPU 消耗也比较高，另外就是轮询，说白了就是 CPU 的死转。SPDK 主要就是这些方面的技术。

## CBS存储引擎



2016 年在做架构的时候，首先开发了一个高性能的开发框架叫 CEDA。这是基于封装的事件驱动的框架，核心理念是基于微服务模型，是事件驱动的、事件触发的高性能的开发框架，存储引擎是基于 CEDA 框架实现的。

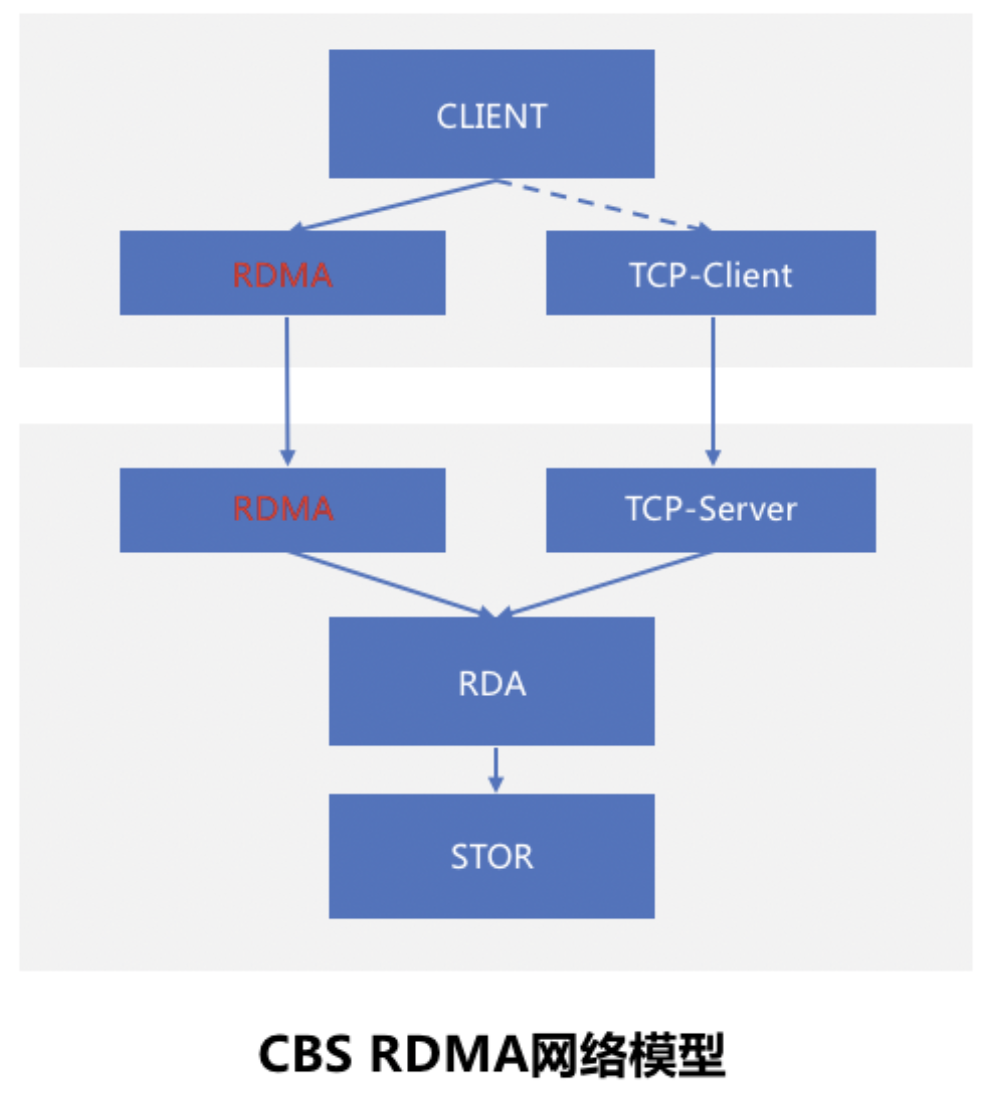
上述图片里面有个 DATA POOl，也就是数据池，IO 过来之后从数据池中分配数据存储的内存，整个生命周期数据不做拷贝，所以可以做到数据零拷贝。

核心路径上跟其他的 SPDK 理念一样，不是用事件驱动的方式，而是用轮训方式，虽然对 CPU 消耗高了，但没有线程的唤醒和休眠，可以做到相对比较好的水平。

存储引擎访问硬盘，现在用的也是 SPDK 方式，可以尽量的减少访问硬盘时在用户端内核进行切换的时间消耗。

但这个过程仍然有一次切换，说白了还是没有达到极致，用户态的任务调度可以让一个 IO 的处理从进存储引擎到最后落地访问存储硬件整个在一个上下文里面做，没有切换的开销，这样就可以把延时做到极致。

## CBS (极速型云盘) 网络



架构、客户端、存储引擎、中间是网络，我们之所以把网络放到最后，并不是说它不重要。软件做到这样是相当不错的，但它的延时和网络比起来还是小巫见大巫，一次跳转的话大几十个 us。

那么如何在网络上做优化呢？腾讯有两款产品，CBS 极速型云盘和增强型云盘，极速云盘是网络层面用的 RDMA，远程直接内存访问，说得简单一些就是一个硬件卸载的理念，把网络的协议的处理尽量放在硬件里面做，这样可以旁路掉操作系统内核的工作。

在硬件和用户态的应用之间直接共享数据，也就是全栈都没有了内存的拷贝，也不需要内核和用户态做协议的解析处理，这些硬件就帮你做了。通过这种操作这样可以把延时降到极至，用的是旁路内核减少切换的理念。

那我们做得跟别人做的有什么不同呢？总结起来，我们做的比较好的几点，虽然是  RDMA 网络，但我们并不单纯是 RDMA，而是 RDMA 和 TCP 并存，在故障的时候可以快速自动切换。

另一方面拥塞控制，腾讯云的量级是百万级别的服务器规模，放眼全球 RDMA 的使用超过一万台机器规模的都很少，RDMA 集群应用规模一般并不大，RDMA 的拥塞控制是比较大的隐患，或者说是现在存在的主要问题之一。

CBS 的 RDMA 拥塞控制层面如何做的呢？除了硬件提供的拥塞控制能力，应用层也做了控制，比如存储和客户端的交互用的 RDMA\_read 方式。

RDMA 有几种交互方式，一种是 send/recv 方式，一种是 read/write 的交互方式，简单而言，前一种模式交互流程是短的、少的，但是切入应用之后会有一次数据的拷贝，因为要有交互协议处理的 buff 和应用之间的 buff 拷贝，对大 IO 来说并不友好。

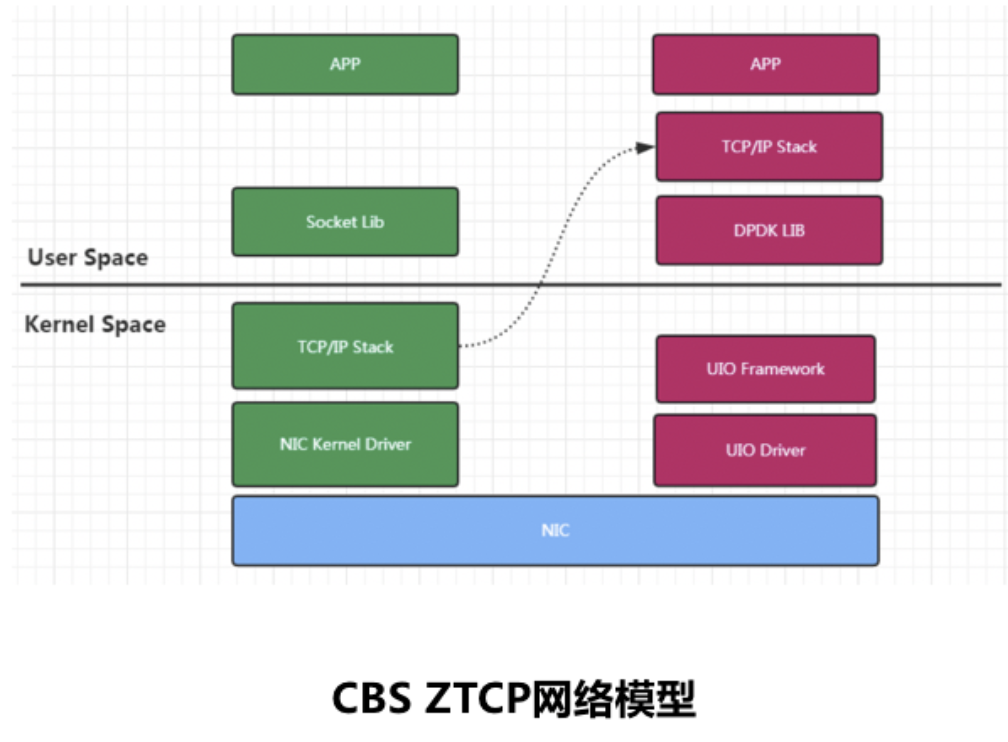
另一种方式 RDMA\_read，最主要的好处是没有了刚才说的拷贝过程，所以对大 IO 比较友好，但是交互流程比前面说的方式要多几跳，这样对小 IO 并不友好。

所以从交互协议层面来说，用了这两种相结合的方式。当需要带宽的时候用的是 read 方式，当需要短的延时时候用的是 send/recv 方式。

用 read 方式做拥塞控制，存储侧来看，比如需要去写个数据到存储的时候，先跟存储握手一次，告诉他我要写了，这时候存储向客户端发一个 read 指令，控制权完全在存储。

因为整个集群模型是非常多的客户端访问少量的存储，存储本身是容易发生拥塞的点，存储端主动的发起数据的读写请求的话，可以根据自己的负载去做拥塞控制，这是基于应用的拥塞控制的可行性，以上是 CBS 在 RDMA 网络方面做得相对比较有特色的几个方面。

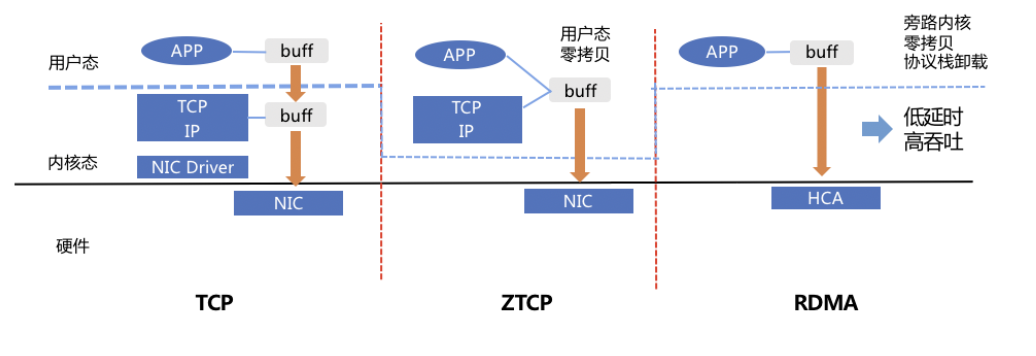
## CBS（增强型云盘）网络



极速云盘用的是 RDMA 网络，增强型云盘用的是 TCP 网络，TCP 网络是延时消耗的大头，内核协议处理占了整个网络延时的百分之八九十，如果还是用 TCP 网络的话就看如何解决掉内核占消耗比较高的问题就可以了。

内核 TCP 方式是内核消耗高，因为上下文切换很频繁，解决这个矛盾放到用户态来看，就是协议处理放到用户态，没有上下文切换。腾讯做的 ZTCP 用户态协议栈同时也可以零拷贝，用户协议也做了用户态零拷贝以求性能的极至，这是增强型云盘的网络协议的优化。

## CBS网络模型对比



CBS 网络模型的交互方式有三种：

传统的内核 TCP 协议栈交互；

极速云盘用的 RDMA 网络交互；

增强型云盘用的用户态 TCP 协议栈交互。

我们对三者进行一个对比。内核态 TCP 把 TCP 解析放到内核，主要问题是频繁的上下文的切换、太多的数据拷贝，这是它的问题。

用户态协议栈是减少数据拷贝和上下文的切换，腾讯做 ZTCP 的特点是同时支持零拷贝，RDMA 是不需要软件做而是硬件来帮助做，协议栈卸载到硬件里面以求做到旁路掉内核、数据零拷贝，达到这个技术的要求，同而将延时降到最低，这是三种网络交互形式的对比。