

系统运维的工作怎么做

一、系统运维概述

提到系统运维大家可能较多想到的是和 OS（Operating System）类相关的工作，但在这里我们延伸并扩展了它的范围。我们把运维里整个基础资源和基础服务的建设，都涵盖进来，比如数据中心建设、网络规划、CDN 服务。

如果能把运维比喻成一栋大厦，每个楼层入驻的是不同业务，系统运维就是大厦的根基和供水供电系统。满足海量业务的资源需求和确保服务的稳定运行是系统运维的核心价值。

数据中心和网络建设

移动互联网爆发，快速地占据入口是制胜根本，而一个平台的建成和完善至少需要 2~3 年的时间。

回顾一些成功的互联网公司的业务发展多数是典型的从无到有，直接到爆发，从业务曲线看，几乎是无平滑的过程。因此系统建设面临了巨大的挑战，这其中最凸显的是数据中心和网络建设。

初创公司前期是不会自建数据中心的，一般租用 ISP 数据中心，然而中国的 ISP 是大欺小的垄断状态，根本没有平等对接；政策把控得严，互通只能在骨干做。

推进技术前进的永远是需求，在骨干城市出现 SP 通过单一 IP 代播静态实现的 BGP 链路，在一定程度上曲折地改善了中国的互联互通质量。静态 BGP 资源满足了业务自身没有用户调度能力的场景。

创业初期，业务功能和未来增长规划不明确，很难做到资源的预估，在数据中心选择上，往往发展不到一年就出现瓶颈，预留又会有过多的成本支出。

有一段时间，我们数据中心的发展一直处于“蜕壳式”成长，通过不断建设新的、更大的数据中心来满足业务。但业务搬迁耗时耗力，服务中断影响用户体验；网络中核心设备不易选型，核心设备如果不投入高端型号，后续扩容升级难度大，替换下来的设备无法再利用。在那段时间里，系统运维的工作是非常痛苦的。

为了解决数据中心和网络建设“蜕壳式”的不足，规划和建设进入了第二阶段。我们把数据中心和网络建设划分成核心数据中心、节点数据中心、传输网三部分。

核心数据中心全部采用“双中心”+异地备份规划，用来承载最主要的业务逻辑单元，关键的数据和内容都放在这里。加强了 SmartDNS 的建设，通过 View 支持、HTTP 支持、EDNS 扩展多种能力，提升浏览器访问质量，同时在客户端层面，通过 HTTPDNS 来提升稳定性和访问质量。系统运维和应用运维、研发一起把业务做到具有多站点部署的能力，实现业务“双活中心”的架构。

一个新数据中心从选择到可以上线业务需要 4~6 个月的时间。所以选择的数据中心一定要有能支持规划内和应对突发事件能力的机柜容量，数据中心供应商最好是有二期、三期的建设计划。

规划需要充分调研各业务线的业务增长，我们每年的 Q3 会把业务线的技术负责人召集到一起，收集至少未来一年的增长计划，突发增长部分的支撑，需要根据历史突发概率的评估来确认，通过商务手段做一些预留。核心网络的规划遵循了一个原则：核心网络设备要适当地超前投入。

双中心间数据要实现同步，逻辑上业务自己解决，物理上要依赖传输网（传输网：数据中心互联网络，也称为 DCI（Data Center Interconnection））来打通。

我们规划建设了两个链路汇聚点，汇聚的物理节点位置都是自有的或长期租赁并有物业管理权的，能自由进出光纤。通过光纤把汇聚点和数据中心进行互联，每两个点之间至少链路双路由互备。传输网建设完成后同时也解绑了链路和机柜的关系，更为建设自己的 BGP、带宽临时调度做好了铺垫。

另外，还把我国内的其他几个节点数据中心连接起来。海外业务发展得也很快，在中国香港地区建设了两个汇聚点，国内的核心和所有海外的数据中心专线都和香港地区的连接起来。

成本虽然会比较高，但是完美地解决了全球的联通性问题。之前没有专线的时候，尝试过走广域网、加密和非加密的 VPN，服务可用性基本没法保证，各种链路差、被拦截、被丢弃、解包。长途部分因为距离长，光纤断的几率自然会高很多，所以长途专线尽量选择有保护的链路。

节点数据中心的应用场景是 CDN Cache 类，业务结构就是分布式的，有冗余的能力。地域选择主要在东部沿海地区，华中、东北、西北、西南少量覆盖节点。这和中国的网民用户分布有直接关系。

节点数据中心的选型过程一般是先分析用户分布，然后预选几个点，通过自开发的用户访问质量系统（UAQ），真实用户抽样测试选出两个点做互备。因为节点数据中心会受到自身容量、骨干拥塞的影响，所以质量是动态的，建设完成后会通过用户在用户访问的接入主机做 TCP 连接和传输速度的分析，通过数据来周期性地调度优化，也会新建优质节点和撤离质量变差的节点，进而形成了一个持续优化的闭环流程。

数据中心内部的选择也尤为重要，硬件条件主要考量建筑、电器、机械 3 个部分，都要做到 T3 标准以上。其他部分考量 Internet 接入、网络攻击防御能力、扩容能力和空间预留、外接

专线能力、现场服务支撑能力几个方面。一般也是预选一批，然后采用分规格和权重打分的模式综合评比。

海外业务的支撑主要选用了相对成熟的 AWS 云服务。应用较多的是 S3 存储、EC2、CDN 加速几个产品。

资产管理的演进

主机和网络的交付是资产管理中主要的内容；系统管理平台的建设要跟上不同量级下的需求。

运维的最早期，采购需求是没有计划的，多是业务增长了就零散地购买。购买完成后设备上架，逐台装机，然后将权限交付给应用运维，资产靠 Excel 记录。随着采购量的提升，逐渐变成了每周有采购、每天有采购，到货批次混乱、无序，原始的手段已经逐渐不能满足发展的需求。

为了改变这种状态，和业务部门做了深度沟通，把零散采购归纳为预算梳理，按月度采购。再后来通过设备 Buffer 池的建设，改为按季度采购，但仍不能避免有一些紧急采购。

虚拟化主机推广范围扩大，对业务自身的隔离、成本的优化都有显著成果。主要在预发布、测试、接入层、中间层的场景大量使用。

在物理机方面，根据业务类型差异对机型进行了定制，这里讲的只是基于成熟的品牌型号进行电源、CPU、内存、存储的定制。还需要结合接入交换机的端口数来布置。我们通过使用高密度机型，最高做出过一个机架 45 个计算节点、13A 电、1 台 48 口交换机的标准，但也存在搬迁节点困难的弊端，并非所有场景都适合。

提供资产管理平台，对物理资产信息进行管理记录，对上层平台提供数据信息。按季度采购的主机需要在短时间内集中化交付，所以开发了高并发的自动装机系统。逐渐把系统运维的工作服务化交付，提供给应用运维主机重启、系统重装、Console 查看等基础功能。和应用运维管理平台定义交互接口，实现资产管理系统和运维管理系统的对接。比如初始化完成的主机自动 Push 到运维管理平台的服务树节点中，应用运维可以进行后续环节的操作。

当设备数量上规模后，硬件、软件故障会成为一种常态，故障的响应处理需要分级别，并制定服务响应等级（SLA）。目前我们的系统管理平台将所有的故障报修服务化起来，对其他平台开放 API，实现报障自动化。每周、每月会评估故障总量、平均故障处理时长，从而挖掘出不合理的环节并优化。

网络设备的配置实现了标准化版本管理，每台设备的配置可以在系统中查看、复制等。

重新设计了设备申请和交付流程，实现不同资源申请对用户同一接口输入/输出。把使用到的公有云、主机租赁、自购主机多种资源的申请，都封装到统一入口，把现有设备使用率监

控系统加入到审批环节，并把交付进度可视化，真正服务化运作起来。

临时性、有突发增长的业务优先使用公有云或私有云。随着公有云和开源私有云系统 OpenStack 的快速发展，公有云计算、存储服务都已成熟，自建私有云交付能力也大大增强，很多场景完全可以使用云化的资源。

强 IO 需求的 DB、Hadoop 服务，仍然跑在自建数据中心。物理主机+混合云的多构成资源模式，将是接下来要重点探索和实践的方向。

基础服务

在服务方面，提供稳定的 DNS 域名解析、LVS 负载均衡、SNAT 集群出口、CDN 加速以及各种基础保障服务（YUM、NTP、SYSLOG）等，并开发对应的管理系统，如 LVS 配置的 Portal 和对自动部署提供 API 接口、DNS 调度管理系统等。

同时，系统运维需要配合业务进行服务优化、硬件优化、内核优化来满足不同的业务场景（CDN 大小文件、域名解析、LVS）。像 DNS、LVS 都是上层业务强依赖的服务，服务短时间的中断都会造成大面积故障，服务可用性都需要定义到 99.99% 以上。要达到高可用性，在架构方面需要设计高可用的集群方案。通过对网络设备以及主机的带外、带内监控及时发现异常。

除了基础设施的建设，安全在系统运维工作中也同等重要。比如网络的公网区域是所有用户访问的第一入口，需要在边界做黑白名单机制，如对 22 等端口做第一防线的防护。

站点知名度提升后，各种攻击就会成为常态，需要在网络方面做 DDoS 防护，特殊的业务入口有流量清洗服务，LVS 做半连接攻击等防护。内部特殊安全区的业务需要各种 ACL，对 ACL 的管理是非常痛苦的事情，我们把这部分功能都流程化审批和管理，将来希望能够做到自动化线上配置。

我们定义了一个第三方 CDN 和自建 CDN 共存的生态环境，选择市场上 2~3 家成熟的 CDN 供应商，细致分析自身业务的特性，比如分析总内容量、大小、曲线规律等，让业务能使用到最适合它的资源。同时做到一个源站对应多个前端 CDN，当一家的服务出现异常时可以快速实现切换。

在业务 CDN 使用量 95% 值达到 30Gbps 的时候，选择了自建 CDN。出发点主要有两个：一是追求质量可控；二是整体成本优化。

在本章中，我们把系统运维涵盖的内容整体串讲了一遍，对系统运维有了一个整体的认识。

二、服务器硬件测试选型

面对琳琅满目的服务器硬件品牌和五花八门的硬件型号规格，如何选择高性价比的硬件配置，是系统运维的一项重要工作。系统工程师需要根据产品线的不同需求，测试服务器的各项性能以及功耗，同时结合成本确定出性价比最高的服务器配置。因此，硬件测试便成为了服务器硬件选型的必要依据。

此外，处理器、内存、磁盘、SSD、磁盘阵列、网卡等配件的不同型号或规格，搭配起来存在无数种配置方案。面对产品线提出的各种配置需求，是否存在一种方法既能够满足业务需求，同时又能让系统工程师轻松、高效地预算和管理服务器？在服务器规模达到千台的时候，我们开始了服务器配置套餐化的工作。

随后为了能够持续地优化成本，我们又开始了多品牌服务器、配件选型、硬件性能和功耗优化的工作。随着服务器数量开始呈现出指数级增长，硬件故障处理的工作量也变得十分庞大。为了能够实现自动化检测和管理硬件故障，我们又开始了服务器硬件领域新的探索和实践。

服务器硬件测试

引入硬件性能评测，提供选型依据

如何将不同硬件之间的差别更加具体化描述呢？比如，如何描述两款不同型号处理器之间的性能差异呢？可视化的性能数据是最直观的。这里就要引入硬件性能评测机制，通过运行权威的性能评测软件将不同硬件之间的差异数据化，为选型提供更强的理性依据。比如 E5-2630v2 处理器性能评测得分是 10000 分，而 E5-2640v2 处理器得分是 12000 分，那么就能得到后者性能是前者 1.2 倍的数据性结论。

常用的硬件评测软件有 GeekBench、Stream、fio、netperf 等。

（1）GeekBench 主要针对处理器的计算性能进行测试评分，能够分别评估整数计算和浮点数计算能力。

（2）Stream 单独针对内存提供较详尽的性能测试，比如针对单线程和多线程的内存操作有不同的测试项目。

（3）fio 是一个灵活性很大的 IO 评测工具，可以用来评估 SAS、SATA 磁盘、SATA SSD 以及 PCIe SSD 等各种 IO 硬件设备的性能。

（4）netperf 是一个常用的网络性能评测工具，可以用来模拟各种网络流量场景，可以评估网卡在不同场景下的包处理和数据吞吐能力。

如图 8-1 所示是通过 GeekBench 和 Stream 工具对两款处理器（E5-2620v2 和 E5-2630v2）做计算和内存性能评估收集到的原始数据。

处理器品牌 处理器型号		Intel E5-2620 v2	Intel E5-2630 v2
GeekBench	Integer Score	18850	22306
	Floating Point Score	27796	32824
	Memory Score	4028	4676
	Stream Score	5595	6489
	Geekbench Score	17691	20879
Stream (MB /s)	Single_Copy	5584.4	6005.0
	Single_Scale	5797.0	6273.5
	Single_Add	7675.9	8321.8
	Single_Triad	7734.3	8347.8
	Mutil_Copy	38841.8	53611.0
	Mutil_Scale	42238.7	67001.7
	Mutil_Add	46145.1	69020.7
	Mutil_Triad	47654.1	69243.9

图8-1 通过GeekBench、Stream工具获取到的测试数据

通过测试数据可以发现 E5-2620v2 的计算能力整体要比 E5-2630v2 低一些，而且两者之间的性能差异也可以数据化。

- （1）整数计算性能（Integer Score 项目）后者比前者高 18%，意味着数据处理能力强 18%。
- （2）单线程混合内存操作（Single_Triad 项目）后者比前者高 8%，意味着单线程下内存处理效率高 8%。
- （3）多线程混合内存操作（Mutil_Triad 项目）则高 45%，意味着多线程下内存处理效率高 45%，后者更适合多任务环境使用。

如图 8-2 所示是通过 fio 工具对不同阵列模式下的 SAS 磁盘做性能测试后收集到的原始数据。

600G 10K SAS		8p RAID 10		6p RAID 50		5p RAID 5	
块大小	队列深度	IOPs	MBPs	IOPs	MBPs	IOPs	MBPs
随机读 (4KB 8KB 16KB 32KB)							
4K	32	2623	10	1990	8	1692	6
8K	32	2585	21	1960	16	1669	13
16K	32	2522	41	1894	31	1621	26
32K	32	2420	79	1842	60	1558	51
随机写 (4KB 8KB 16KB 32KB)							
4K	32	1952	7	1053	4	544	2
8K	32	1927	15	1038	8	548	4
16K	32	1874	30	1009	16	537	8
32K	32	1787	58	964	31	513	16
4KB混合随机读写 (30%写)							
4K	32	2223	9	1496	6	1015	4
顺序读 (128KB 256KB 512KB)							
128K	32	7532	987	5031	659	4179	547
256K	32	3784	991	2340	613	2129	558
512K	32	1411	739	1045	547	1009	529
顺序写 (128KB 256KB 512KB)							
128K	32	36734	4814	34196	4482	16682	2186
256K	32	18951	4967	17196	4507	5280	1384
512K	32	11770	6170	9812	5144	10856	5691

图8-2 通过iio工具获取到的不同阵列模式下SAS磁盘的原始性能数据

通过数据可以更理性对比不同 IO 设备在随机读/写、顺序读/写方面的差异，为选型提供理性依据。

同类硬件只考虑一种规格

通过硬件性能评测能够准确地评估出处理器、磁盘、SSD 等各种硬件的极限性能，便于我们将服务器按照计算、I/O、存储为单元进行拆分并单独评估性价比。在套餐精简的基础上，为了能够形成更强的规模效应，应该从统一配件规格上着手优化。因此，硬件性能便成为了评估性价比的重要指标。

比如，通过评测工具得到候选处理器的性能评分，同时结合处理器的价格来评估性价比。假设处理器 A 的单价为 4000 元，性能得分是 13000 点；处理器 B 的价格仅为 2000 元，性能得分是 10000 点。虽然处理器 A 的性能是 B 的 1.3 倍，但是 A 的价格比 B 高一倍。通过性价比计算公式：性价比（点/元）= 计算性能（点）/ 单颗价格（元），得到处理器 A 的性价比是 3.25 点/元，处理器 B 是 5 点/元，那么处理器 B 的性价比是处理器 A 的 1.53 倍。

请参考图 8-3。

评估项目	处理器A	处理器B
计算性能 (点)	13000	10000
单个价格 (元)	4000	2000
性价比 (点 / 元)	3.25	5

图8-3 处理器A和处理器B性价比评估

在满足业务性能的前提下，优先选择处理器 B，那么就不必考虑在套餐中同时存在两种型号规格的处理器的了。此外，同种规格的处理器的采购量越大，成本的规模效应越明显，也更容易通过部件进行议价。其他配件也同样适用这个原则。

追求单机性价比

在前面的例子中，虽然处理器的性价比是很重要的一项指标，但偶尔也会有意外的情况发生，比如某些注重单机计算能力的业务应用。这时单纯考量单个处理器的性价比就不适用了，应该转向考量整机的计算性价比。

比如，处理器 C 售价是 6000 元，是处理器 B 的 3 倍，但是 C 的计算评分是 25000 点，是 B 的 2.5 倍。C 的整机成本是 25000 元，而 B 的整机成本为 20000 元。虽然处理器 C 在单个处理器的性价比上不如 B，但如果从单机成本对比来看，C 整机成本为 B 整机成本的 1.25 倍，因此处理器 C 的单机性价比是处理器 B 的 2 倍。请参考图 8-4。

评估项目	处理器C	处理器B
计算性能 (点)	25000	10000
单个价格 (元)	6000	2000
性价比 (点 / 元)	4.16667	5
整机成本 (元)	25000	20000
整机性价比 (点 / 元)	1	0.5

图8-4 处理器C和处理器B性价比评估

因此，选择处理器 C 来提高业务的单机计算能力，节省成本。对于其他硬件也同样适用该原则。

服务器套餐化

在大部分情况下产品线并不知道自己的真实需求：用什么处理器、多大的内存、多少块磁盘。通常都是凭以往的使用经验拍脑袋决定的：今天多加几条内存，明天多加几块磁盘。这样给系统工作带来了极高的服务器配置管理成本。

例如，某公司每个月都会采购新的服务器，某个月产品线提出了共 70 个配置需求，服务器供应商总共有 4 家，那么系统工程师等同于要面对和管理共 280 个服务器配置。从需求收集到落实采购，这期间和产品线、供应商的沟通、确认将耗费大量的精力，时间几乎都要耗在预算采购这一件事情上，而且还容易出错。同时，如此众多的服务器配置也无法形成相同配置的规模效应，无形之中增加了公司的运营成本。因此，服务器配置套餐化十分有必要。

什么是套餐化

现在有两家菜馆：A 餐馆有 100 道菜，B 餐馆有 10 个套餐。那么，用户去哪家餐馆点餐最快？显然是 B 餐馆，而 B 餐馆就是现在大家熟知的快餐店。在大部分情况下用户并不清楚想吃什么，怎么搭配菜品更合理等。而套餐的存在能够快速将餐品按一定原则分类搭配，给客户提供了预选的方案，同时还精简了菜品数量。

服务器配置套餐化也是同样的道理，按照业务方需求提供预先设置好的配置，能够让业务更容易做出选择，而且套餐数量精简后更容易形成规模效应，增加采购议价能力。

对服务器套餐进行分类

建立服务器配置套餐以后，为了方便管理就需要将套餐进行分类。根据业务需求，我们将服务器配置抽象分类成以下几类。

- （1）计算型：Web 前端、缓存类业务等；这类配置一般处理器主频规格和内存配置较高，对数据存储的要求较低。
- （2）计算 IO 均衡型：业务中间件、数据库等；这类配置通常处理器性价比较高，内存和磁盘配置适中。

（3）重 IO 型：数据库等；这类配置在均衡型的基础上往往会将磁盘更换成 SSD，以满足高 IO 负载的需求。

（4）存储型：分布式存储等；这类配置注重单 GB 成本，通常会使用廉价高容量的 SATA 磁盘，对处理器和内存的要求并不高。

分类后，同一类套餐可能存在多种配置，接下来就是配置近似套餐的抽象合并过程。

精简套餐数量，实现规模效应

比如，另外一家公司总共拥有 4 个套餐配置，也面对 4 家服务器供应商，每次采购只需要管理和维护共 16 个配置。这样一来，无论是采购还是资源管理都变得简单、高效起来。同时，套餐数量越少也就意味着规模效应越明显，采购议价能力越强。这就是为什么连锁快餐店总是能把套餐价格做得很低，同时还能保证高效的供应，并且还能拥有较高的利润的原因，一切都归功于规模效应。

套餐配置分类以后总是能够出现配置接近的套餐，尽可能地合并这些配置，将相似的配置抽象成一种套餐，不断地精简套餐数量。比如，X 和 Y 套餐仅仅在内存配置上存在差异：X 内存 128GB，Y 内存 64GB，而 X 需求量是 Y 的 10 倍，那么就应该考虑将 Y 套餐合并成 X 套餐，因为 X 套餐的采购价格很可能低于 Y 套餐。

成本的持续优化

服务器套餐化带来了成本的大量节省，但追求成本的持续优化只能另辟蹊径。我们在如下几个方面尝试了一些探索。

多品牌服务器的引入

通过多品牌差异化的竞争增加议价能力，降低成本。当然，引入多品牌服务器后很快出现了各种新问题，比如某些品牌服务器不支持共享口，BIOS 配置标准各家有差异，管理口对运维支持情况各异等一系列运维可用性问题。很快，运维可用性测试便成为了多品牌服务器引入后的一项重要测试指标。

通过可用性测试提前发现各种服务器机型的基础运维问题，以便后续要求厂商按照用户的运维习惯进行定制化。入围的品牌我们会按季度来总结故障率，对故障率不达标又没有完成可规避方案的品牌后续减少采购量，甚至不采购。

配件选型减少服务器规模

前面提到了同类硬件尽可能要考虑一种规格，而且单机性价比才是最重要的，因此配件的测试选型也就显得格外重要。

同一类配件如 SSD，也存在 PCIe 和 SATA 两种规格。假设某业务对存储设备的 IO 需求较高，而使用 SATA SSD 的方案单机最多能够提供的 IOPs 能力为 20000，现在有一款 PCIe SSD 能够提供的单机 IOPs 能力为 60000。

虽然同容量 PCIe SSD 的成本是 SATA SSD 的 2 倍，但从整机价格来看，PCIe SSD 却只有 SATA SSD 的 1.2 倍，那么使用 PCIe SSD 可以有效减少 2/3 的服务器采购量，而单机成本仅提高了 20%。因此，业务方全面使用 PCIe SSD 来缩小服务器投入规模，在降低成本的同时还减少了运维的开销。

硬件性能调优

以往提升单机性能的方法就是升级硬件，如何在不增加成本的前提下提升硬件的性能呢？通常的做法是软件层面的优化，比如修改 BIOS 参数、修改内核、升级固件、修改操作系统参数等做法均能达到预期收益。

比如修改 BIOS 参数，打开处理器的 Turbo（睿频）功能能够瞬间提升处理器的运行频率，提升处理器的性能（如图 8-5 所示是 E5-2630v2 处理器在开启 Turbo 前后的性能测试数据，可以看到打开 Turbo 后整数计算性能得到了 15%的提升）。

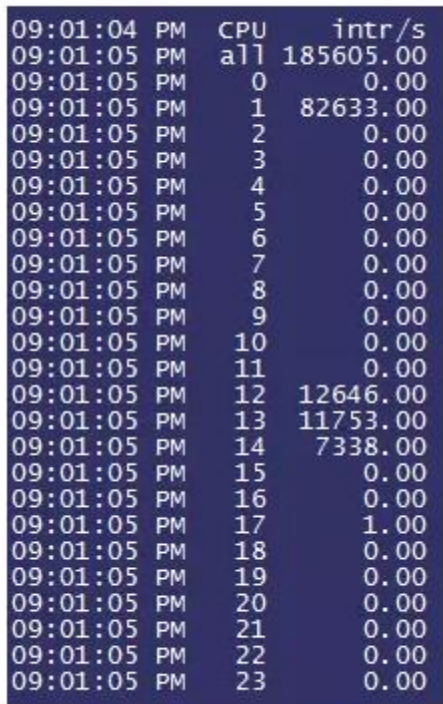
处理器品牌		Intel	Intel
处理器型号		E5-2630 v2	E5-2630 v2
BIOS设置	Turbo	off	on
GeekBench	Integer Score	22306	25661
	Floating Point Score	32824	37524
	Memory Score	4676	4953
	Stream Score	6489	7039
	Geekbench Score	20879	23809

图8-5 E5-2630v2处理器在打开Turbo设置前后的性能测试数据对比

再比如操作系统参数调优，举的一个例子是在 SATA SSD 加直连卡的硬件环境下使用中断多核绑定来优化 IO 性能。业务使用 8 块 SATA SSD 加一块直连卡的硬件方案，分别将数据保存在 8 块 SSD 中进行读/写。

在优化前业务的 QPS 仅能够达到 27K，SSD 的使用率为 60%~70%，8 块 SSD 的 IOPs 总和达

到了 80K。通过 mpstat 命令可以发现直连卡产生的 IRQ 中断全部压在了处理器的第一个核心上（如图 8-6 所示），产生了 82K 个中断，这和前面提到的 IOPs 数量吻合。



09:01:04	PM	CPU	intr/s
09:01:05	PM	all	185605.00
09:01:05	PM	0	0.00
09:01:05	PM	1	82633.00
09:01:05	PM	2	0.00
09:01:05	PM	3	0.00
09:01:05	PM	4	0.00
09:01:05	PM	5	0.00
09:01:05	PM	6	0.00
09:01:05	PM	7	0.00
09:01:05	PM	8	0.00
09:01:05	PM	9	0.00
09:01:05	PM	10	0.00
09:01:05	PM	11	0.00
09:01:05	PM	12	12646.00
09:01:05	PM	13	11753.00
09:01:05	PM	14	7338.00
09:01:05	PM	15	0.00
09:01:05	PM	16	0.00
09:01:05	PM	17	1.00
09:01:05	PM	18	0.00
09:01:05	PM	19	0.00
09:01:05	PM	20	0.00
09:01:05	PM	21	0.00
09:01:05	PM	22	0.00
09:01:05	PM	23	0.00

图8-6 优化前mpstat情况

这已经达到了单核的处理极限，而 SSD 的使用率却还有盈余。因此，考虑对直连卡做多核的中断绑定来优化 I/O 性能。通过 /proc/interrupt 可以发现直连卡中断号为 107~118，因此执行以下命令对中断做多核绑定（如图 8-7 所示），将不同的 IRQ 分别绑定到不同的处理器核心上。



```
echo 2 > /proc/irq/107/smp_affinity
echo 4 > /proc/irq/108/smp_affinity
echo 8 > /proc/irq/109/smp_affinity
echo 10 > /proc/irq/110/smp_affinity
echo 20 > /proc/irq/111/smp_affinity
echo 40 > /proc/irq/112/smp_affinity
echo 80 > /proc/irq/113/smp_affinity
echo 100 > /proc/irq/114/smp_affinity
echo 200 > /proc/irq/115/smp_affinity
echo 400 > /proc/irq/116/smp_affinity
echo 800 > /proc/irq/117/smp_affinity
echo 1000 > /proc/irq/118/smp_affinity
```

图8-7 执行多核中断绑定

优化后业务 QPS 达到了 39K，SSD 的使用率也提升至 80%~90%，性能提升达到 40%之多。硬件性能调优是最理想的成本优化方案，它在不产生任何直接成本的前提下就可得到单机性能提升的收益。

机柜高密度部署的探索

业务对服务器的需求规模从几百台一下子增长到了几千台，服务器规模的突增给运维带来了很大的冲击。很快，公司原先的机架资源便面临枯竭。由于原先公司体量不大，在和各数据中心谈合作时很难及时获取到机柜资源，而业务却等着要上线服务。这个时候为了能够缓解机架资源的紧张程度，我们便开始探索机柜高密度部署，很快高密度服务器便进入了我们的视线中。

降低服务器运行功率

通过 BMC（BMC 即服务器带内管理控制卡，它可以获取服务器状态和故障信息，以及控制服务器的停机、开机等电源操作）获取服务器实时运行功率，是另外一种实际且有效地掌握服务器用电情况的方法。结合大数据还能够分析出同配置服务器的功耗运行规律，为单机柜实现更高密度的部署提供真实的数据。

此外，还可以结合各种服务器功耗优化方案来实现单机柜更高密度的服务器部署。服务器功耗优化的方式有很多种，例如：

- （1）使用高标号电源（比如白金、黄金电源）提高电源转换效率，减少电源谐波干扰；
- （2）使用低功率电源提高电源负载率，减少转换损耗；
- （3）使用低功耗硬件设备，比如低电压内存和低功耗处理器等；
- （4）优化服务器风扇散热策略，在保证散热的同时降低转速，优化功耗。

硬件状态扫描和故障预警

服务器达到规模后，故障维护也相应变得困难起来，主要体现在：

- （1）硬件故障无法提前预见。
- （2）硬件故障分类判断不精准。
- （3）硬件故障无法被及时感知。

人工干预的故障排查不切实际，这些棘手问题都给系统运维工作带来困扰。比如，某个服务器阵列类型是 RAID 5，一块盘出现故障后，业务仍旧能继续正常运行。但是由于该故障无法被业务应用或运维人员及时感知，因此没有及时更换磁盘。随着时间的推移，继而出现了第二块磁盘的故障，最后导致整个阵列丢失了数据。在实际线上生产环境中，这类问题屡见不鲜。

大部分故障都是显性的

在积累了大量服务器硬件故障维护经验后，我们发现大部分故障都是可预见的。可以通过系统、BMC（带内管理卡）日志或者硬件本身记录的日志，以及各种硬件状态检测工具进行故障定位。比如阵列卡本身带有对磁盘设备的管理功能，可以通过专门的工具获取到当前阵列卡、磁盘的状态信息（如图 8-8、图 8-9 所示），并且阵列卡本身也记录了各类日志，可以方便地通过日志判断故障（如图 8-10 所示）。

```
Virtual Drive: 1 (Target Id: 1)
Name                : vd02
RAID Level          : Primary-1, Secondary-0, RAID Level Qualifier-0
Size                : 792.0 GB
Mirror Data         : 792.0 GB
State               : Optimal
Strip Size          : 128 KB
Number Of Drives per span: 2
Span Depth          : 4
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Default Access Policy: Read/Write
Current Access Policy: Read/Write
Disk Cache Policy   : Disk's Default
Encryption Type     : None
Default Power Savings Policy: Controller Defined
Current Power Savings Policy: None
Can spin up in 1 minute: No
LD has drives that support T10 power conditions: No
LD's IO profile supports MAX power savings with cached writes: No
Bad Blocks Exist: No
Is VD Cached: No
```

图8-8 通过MegaCli工具获取到的阵列卡虚拟盘的状态信息


```

Enclosure Device ID: 32
Slot Number: 7
Drive's position: DiskGroup: 0, Span: 3, Arm: 1
Enclosure position: 0
Device Id: 7
WWN: 50015178F3667C6D
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
PD Type: SATA
Raw Size: 223.570 GB [0x1bf244b0 Sectors]
Non Coerced Size: 223.070 GB [0x1be244b0 Sectors]
Coerced Size: 223.0 GB [0x1be00000 Sectors]
Firmware state: Online, Spun Up
Device Firmware Level: DC02
Shield Counter: 0
Successful diagnostics completion on : N/A
SAS Address(0): 0x4433221103000000
Connected Port Number: 2(path0)
Inquiry Data: BTDA327501B32403GN INTEL SSDSC2BW240A4
FDE Enable: Disable
Secured: Unsecured
Locked: Unlocked
Needs EKM Attention: No
Foreign State: None
Device Speed: 6.0Gb/s
Link Speed: 6.0Gb/s
Media Type: Solid State Device
Drive: Not Certified
Drive Temperature :19C (66.20 F)
PI Eligibility: No
Drive is formatted for PI information: No
PI: No PI
Drive's write cache : Disabled
Drive's NCQ setting : Disabled
Port-0 :
Port status: Active
Port's Linkspeed: 6.0Gb/s
Drive has flagged a S.M.A.R.T alert : No

```

图8-9 通过MegaCli工具获取到的阵列卡下磁盘或SSD的状态信息

```

04/24/15 11:08:34: initFreeDDBs=xorPool c0e38000-c0e4a000 genPool c0e00000
-c0e18000 runTime 1 numXorDdb 80 numCipherDdb =400
04/24/15 11:08:34: initBladeSGL c0e18000-c0e38000 runTime 1 numSglFrame 40
0
04/24/15 11:08:34: ld sync: all LDs sync'd
04/24/15 11:08:34: ld sync: all LDs sync'd
04/24/15 11:08:34: ld sync: all LDs sync'd
04/24/15 11:09:34: EVT#284017-04/24/15 11:09:34: 113=Unexpected sense: Enc
1 PD 20 Path 5a0b10c00c6daa00, CDB: 1c 01 b1 04 00 00, Sense: 5/24/00
04/24/15 11:09:34: Raw Sense for PD 20: 70 00 05 00 00 00 00 0a 00 00 00 0
0 24 00 00 00 00 00
04/24/15 11:09:35: EVT#284018-04/24/15 11:09:35: 113=Unexpected sense: Enc
1 PD 20 Path 5a0b10c00c6daa00, CDB: 1c 01 b2 04 00 00, Sense: 5/24/00
04/24/15 11:09:35: Raw Sense for PD 20: 70 00 05 00 00 00 00 0a 00 00 00 0
0 24 00 00 00 00 00
04/24/15 11:09:36: EVT#284019-04/24/15 11:09:36: 113=Unexpected sense: Enc
1 PD 20 Path 5a0b10c00c6daa00, CDB: 1c 01 b3 04 00 00, Sense: 5/24/00
04/24/15 11:09:36: Raw Sense for PD 20: 70 00 05 00 00 00 00 0a 00 00 00 0
0 24 00 00 00 00 00
04/25/15 3:00:00: EVT#284020-04/25/15 3:00:00: 292=Patrol Read can't be
started, as PDs are either not ONLINE, or are in a VD with an active proce
ss, or are in an excluded VD
04/25/15 3:00:00: Next PR scheduled to start at 05/02/15 3:00:00

```

图8-10 通过MegaCli工具获取到的阵列卡终端日志

磁盘的SMART信息也能给诊断磁盘故障带来帮助（如图8-11所示）。此外，处理器、内存等故障也可以通过BMC的SEL日志获取到（如图8-12所示）。很快，我们就可以将故障按照部件进行分类，不同部件的故障定位使用不同的方法应对，工具和日志相结合，综合分析。这便给自动化硬件故障监控提供了技术基础。

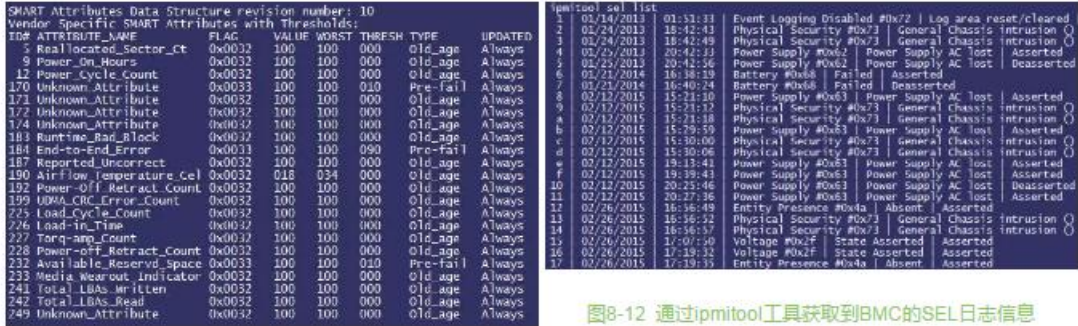


图8-11 通过smartctl工具获取到的磁盘SMART信息

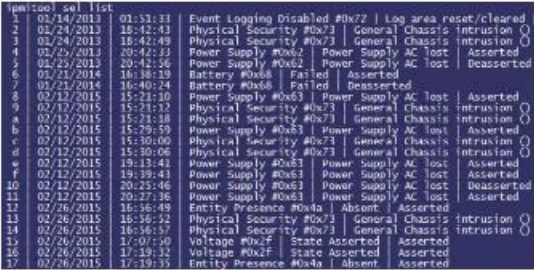


图8-12 通过pmtool工具获取到BMC的SEL日志信息

统一硬件健康检查工具

为了能够更加高效地管理监控服务器故障，统一硬件健康监控检查工具的设计很快便被提出来。前面提到大部分故障都是显性且可以分类的，因此如果能够编写一个对服务器不同硬件分别做状态检查的软件工具，就可以及时了解服务器当前是否存在异常或者故障。

前面提到过不同服务器硬件的状态信息可以用不同的方法查看，而统一硬件健康检查工具可以通过调用相关的硬件检查工具，按照一定的方法逻辑检查硬件状态并分析日志，对服务器硬件整体健康状态做一一检查。其结果就是，工具在任何型号服务器上运行后便能输出当前服务器的健康状态信息：没有问题就不输出任何结果，而存在问题便输出对应的故障分类和具体的故障信息（如图 8-13 所示）。

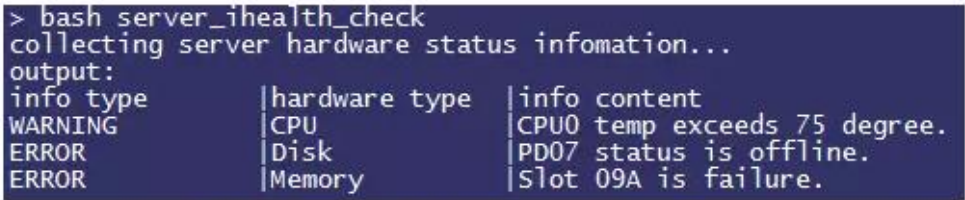


图8-13 通过统一硬件健康检查工具输出硬件状态检查信息

常用的检查硬件状态的工具和方法如下。

- (1) 磁盘类：MegaCli、hpacucli、smartctl 等。
- (2) 处理器、内存类：mcelog 以及通过 ipmitool 工具查看 SEL 日志。
- (3) 电源、风扇类：通过 ipmitool 工具查看 SDR（传感器）状态。
- (4) 其他类：通过 ipmitool 工具查看 SEL 日志。

再进一步，我们可以将工具与现有的监控系统结合，通过监控代理程序定期运行工具便能准确及时地获取到服务器的硬件故障信息。甚至还可以对故障的严重程度进行分级，以便运维人员在获取到报警信息后快速判断故障的严重程度。

磁盘健康检查工具

在所有服务器故障中磁盘故障占了相当大的比例（一般使用 SAS 磁盘的服务器 50% 以上均为磁盘故障），因此很有必要对磁盘做自动化健康检查。磁盘系统作为服务器的 IO 设备是一个复杂且庞大的体系，它并不像内存那样只有相对单一的结构和规格，光是磁盘的接口类型就有 SAS、SATA 两种，而且还有阵列卡、直连卡等 IO 中间设备作为磁盘系统的一部分。通常使用阵列卡的服务器较多，下面来讲讲在阵列卡环境下如何实现磁盘的自动化健康检查。

磁盘健康检查工具的实现原理

阵列卡作为磁盘系统的一部分，能够将多块物理盘通过不同的阵列算法组合成一个大的虚拟盘；同时阵列卡又承担起管理虚拟盘和物理盘的责任，对于虚拟盘和物理盘出现的错误进行检查和管理。而我们开发的自动化健康检查工具，正是基于能够和阵列卡信息交互的工具来了解虚拟盘和物理盘的健康状况的（如图 8-14 所示）。

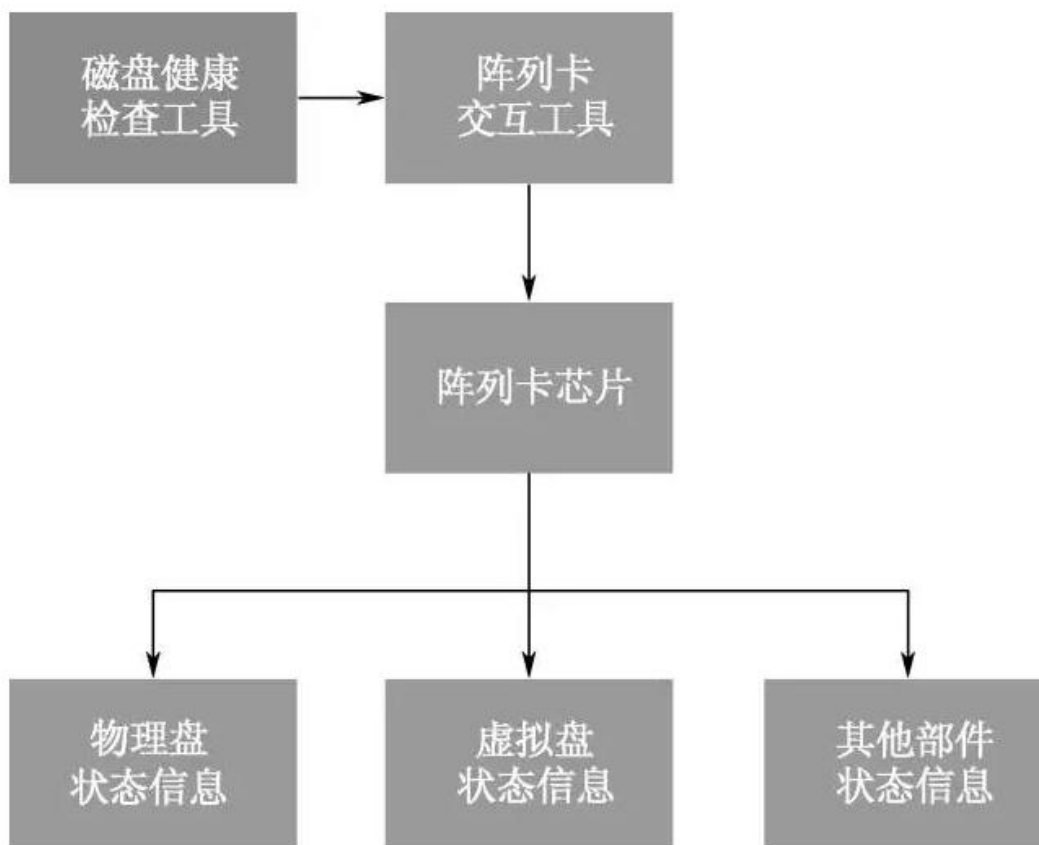


图8-14 磁盘健康检查工具的实现原理图

市面上的服务器大部分使用的阵列卡是基于 LSI 公司设计生产的芯片，因此和阵列卡的交互都可以通过 MegaCli 工具实现。

通过 MegaCli 工具可以获取到虚拟盘和物理磁盘的拓扑结构以及一些基础信息，比如某个阵列卡下有一个虚拟盘 VD0 由物理盘 PD0、PD1、PD2 组成，阵列类型是 RAID 5，缓存设置为 No Read Ahead、Write Back，并且当前状态为 Online 等；物理盘 PD0、PD1、PD2 分别对应的槽位 ID 为 slot0、slot1、slot2，并且磁盘状态均为 Online；以上信息说明虚拟盘 VD0 和物理盘 PD0、PD1、PD2 均处于 Online 状态，没有任何故障出现。

进一步，我们可以单独了解 PD0 的更详细信息。

比如它的某项错误计数器 Media Error Count 数量已经大于 100，表明已经有超过 100 个的物理坏道出现，但是磁盘仍旧处于 Online 状态，这说明磁盘虽然还能正常工作，但是已经处于非健康状态，这块盘很可能在近期出现故障，因此通过 MegaCli 工具获取到这些信息提前预见了故障。通过上面的这些判断逻辑，我们将健康检查分成两类：虚拟盘的健康检查和物理盘的健康检查。

定义不同级别的故障信息

为了能够区分磁盘故障的紧急程度，我们根据故障紧急程度的不同定义了不同级别的故障信息通知 INFO、WARN、ERROR 和 FATAL。

- ◎ **INFO: information**，即信息级别，通知用户磁盘处于非异常也非完全健康的状态，紧急程度最低；比如虚拟盘正处于修复状态中。
- ◎ **WARN: warning**，即警告级别，通知用户磁盘系统存在一些问题，但是并非完全被定义为故障；比如出现 **Unconfigured Good** 状态的磁盘，可能是虚拟盘出现了掉盘的问题。
- ◎ **ERROR: error**，即故障级别，通知用户磁盘系统出现了故障但并不致命，不会出现数据丢失的情况，级别比 **WARN** 高，需要用户尽快介入处理故障以避免故障级别升级；比如一组 **RAID 5** 的虚拟盘出现了一块磁盘故障的情况，虽然虚拟盘仍旧可以提供 **IO** 服务，但是无法再忍受更多的磁盘故障，因此需要尽快更换故障盘。
- ◎ **FATAL: fatal error**，即致命故障级别，通知用户磁盘系统出现了致命的故障，已经无法提供 **IO** 服务，并且有极大的概率会出现数据丢失的问题；比如一组使用 **RAID 0** 的虚拟盘有一块磁盘出现故障，致使虚拟盘已经无法继续提供 **IO** 服务，并且数据全部丢失。

虚拟盘健康检查的设计逻辑

根据虚拟盘的不同状态来判断其健康状况，虚拟盘分别有 **Optimal**、**Offline**、**Recovery** 和 **Degraded** 四种状态。

- ◎ **Optimal** 为最佳状态，意味着虚拟盘目前处于健康状态，没有任何故障和异常。
- ◎ **Offline** 为离线状态，意味着虚拟盘已经出现了严重的故障，比如 **RAID 0** 出现一块磁盘故障，或者 **RAID 5** 出现两块磁盘故障，表示虚拟盘已经不可用并且数据有丢失。
- ◎ **Recovery** 为虚拟盘修复中状态，意味着虚拟盘之前出现了磁盘故障并降级，更换故障盘后阵列卡对新盘进行有效数据填充的过程。
- ◎ **Degraded** 为虚拟盘降级状态，意味着带有阵列保护的虚拟盘出现了磁盘故障，但不至于对虚拟盘的数据完整性造成破坏，比如 **RAID 5** 出现一块磁盘故障就会导致虚拟盘降级。

通过上面的这些状态我们就可以设计一套判断虚拟盘是否健康的逻辑，具体的设计逻辑图如图 8-15 所示。如果检查一个虚拟盘状态为 **Optimal**，则不输出任何信息，直接检查下一个虚拟盘。检查状态为 **Offline**，则按照最高级别 **FATAL** 输出错误通知；状态为 **Degraded**，则按照比 **FATAL** 低一级别的 **ERROR** 输出错误通知；状态为 **Recovery**，则表示处于非异常也非健康状

态，因此以 INFO 级别输出错误通知。

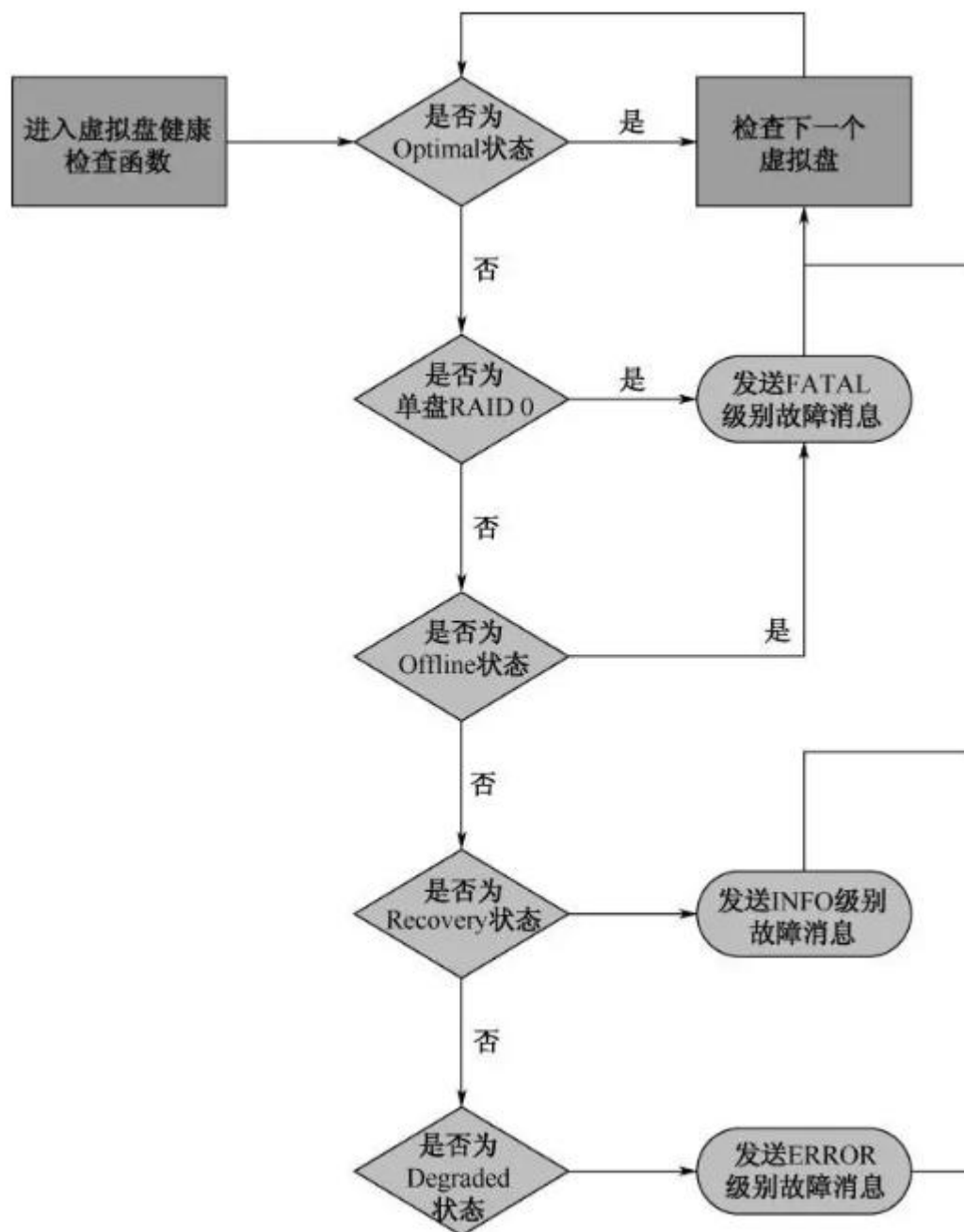


图8-15 虚拟盘健康检查的设计逻辑图

物理盘健康检查的设计逻辑

物理盘也有类似的一些状态来判断是否健康,如 Unconfigured Good、Unconfigured Bad、Online、Offline、Rebuild 等。

◎ **Unconfigured Good** 为未被配置为阵列的健康盘，可能是新加的磁盘或者是某个阵列出现了掉盘都会被阵列卡设置为这种状态。

◎ **Unconfigured Bad** 为被阵列卡判断为异常状态的磁盘，可能之前属于某个阵列，但是由于出现错误较多而被阵列卡从阵列配置中踢出，也有可能是新加入的磁盘被阵列卡检查健康状态后判断为故障盘。

◎ **Online** 为在线状态，但并不意味着磁盘本身不存在问题，它仅代表该磁盘尚能提供 IO 服务。某些出现坏道而引发降速的磁盘，由于尚能提供 IO 服务，仍会被阵列卡判断为在线状态，因此还需要增加其他的因素去判断一块磁盘是否真正健康。

◎ **Offline** 为离线状态，拔盘或者因为接口通信异常而掉线的磁盘都会被阵列卡设置为该状态，更严重的是磁盘故障到已经无法通信的情况也会被设置为离线状态。

◎ **Rebuild** 为修复中状态，当一块新盘被用以替换故障盘而参与整个虚拟盘数据修复时，阵列卡会将该盘设置为修复状态。

◎ 此外，上面讲到即使一块物理盘处于 **Online** 状态，也无法判断为健康状态，因此物理盘比虚拟盘还多了一种可以辅助判断是否健康的因素，那就是错误计数器：**MEC**（Media Error Count）、**OEC**（Other Error Count）、**PFC**（Predictive Failure Count）。

◎ **MEC**（Media Error Count）为物理坏道计数器，通常指的是磁盘出现无法修复的物理坏道数量，该计数器积累到一定程度后即可判断为磁盘故障。

◎ **OEC**（Other Error Count）为其他错误计数器，指的是物理坏道以外的错误数量。由于磁盘是一种比较精密且复杂的机械和电子部件，因此任何部件的异常磁盘都有一种检查机制能够感知并写入磁盘的 **SMART** 信息中，因此该计数器可作为判断磁盘是否故障的辅助手段。

◎ **PFC**（Predictive Failure Count）为故障预警计数器，它通常集合了一套判断磁盘是否即将故障的逻辑，阵列卡根据磁盘提供的各种错误代码和信息判断磁盘即将故障，该计数器积累到一定程度后即可判断为磁盘故障。

通过上面的这些状态和错误计数器，我们就可以设计一套判断物理盘是否健康的逻辑，具体的设计逻辑图如图 8-16 所示。如果检查一块磁盘为非 **Online** 状态，就会开始一系列检查：**Unconfigured Good** 状态会以 **WARN** 级别输出错误通知；**Unconfigured Bad** 或 **Offline** 状态均会以 **ERROR** 级别输出错误通知；而 **Rebuild** 状态则会以 **INFO** 级别输出信息通知。

最后，无论该物理盘状态是否为 **Online**，都会对计数器值进行检查判断，如果三个计数器有任何一个超过临界值，都会以 **WARN** 级别输出错误通知，以及时通知用户磁盘存在异常。

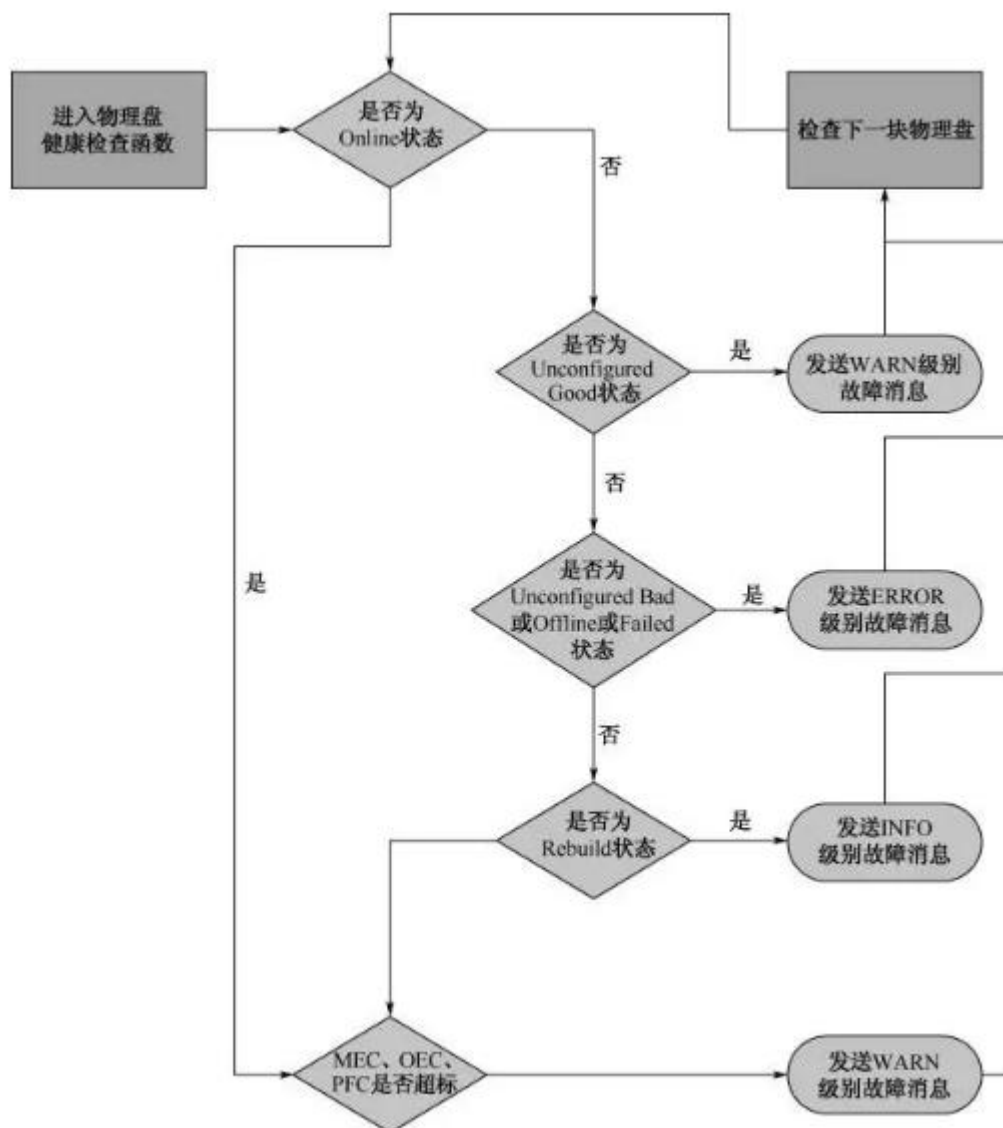


图8-16 物理盘健康检查的设计逻辑图

故障信息的输出格式

关于健康检查的故障通知的输出设计逻辑是：检查到任何异常便输出通知，没有异常不输出任何信息。因此，用户可以专注于出现故障信息的服务器，而不必被一些过多的状态信息所干扰。此外，对于信息的输出格式我们也有精心设计，一条故障信息由故障定级、故障来源和故障原因三部分组成。具体的输出格式演示如图 8-17 所示。

故障定级：前面提到过错误信息的定级分类，错误输出信息里也应该有这部分内容，以使用

户在获取到信息后再次对错误信息按照紧急程度进行分类，增强用户体验。

故障来源：错误信息中应该包含具体哪个虚拟盘、哪块物理盘的消息，甚至可以告知用户物理盘的槽位号，帮助用户精准定位故障来源。

故障原因：有了错误信息的具体来源还不够，还需要告知用户具体的错误原因，比如某块物理盘处于 **Offline** 状态，某个虚拟盘由于掉盘而出现 **Degraded** 状态，这些具体的错误原因都要在错误信息中输出，帮助用户更好地判断如何处理故障。

```
> bash disk_health_check.sh
WARN:VD0,WriteCache is Disabled
WARN:PD2 slot=2,Media Error Count is more than 10
WARN:PD9 slot=9,Media Error Count is more than 10
WARN:PD13 slot=13,State is Unconfigured Good
```

图8-17 故障信息输出格式演示

三、服务器资源使用率

服务器数量达到一定规模后，老板们开始担心了：“每季度这么大金额的服务器采购支出，我们的服务器资源使用率如何？”

指标计算

了解资源使用率的前提是记录所拥有的服务器资源以及归属，在服务器采购到位后，就开始跟踪服务器的各项资源指标。但每次老板们问到这个问题时，作为运维人员还是很难直接回答。我们记录了每台机器历史的各项资源使用数据，但如何体现整体的资源使用情况，通过简单的指标表示出来还是比较麻烦的。

首先，我们不能把所有服务器罗列出来，逐台给老板汇报。

其次，每台服务器每天的资源占用情况是随时间，随业务特性动态变化的，需要将 CPU、内存、磁盘、IO 等每个动态变化的资源指标分别换算成一个值。

例如：一台 CPU 密集型的服务器，每个时间点访问量的不同，CPU 使用率也是不同的，如图 4-1 所示。

最后，每台服务器每天的平均值，由于高峰期或低峰期的差值很大，均值没有任何参考意义。

取每台服务器每天的最大峰值，有时候可能由于某一次数据传输，或者跑一个临时 MD5 运算导致 CPU 使用率突增，也不能合理代表这台服务器的 CPU 使用率。

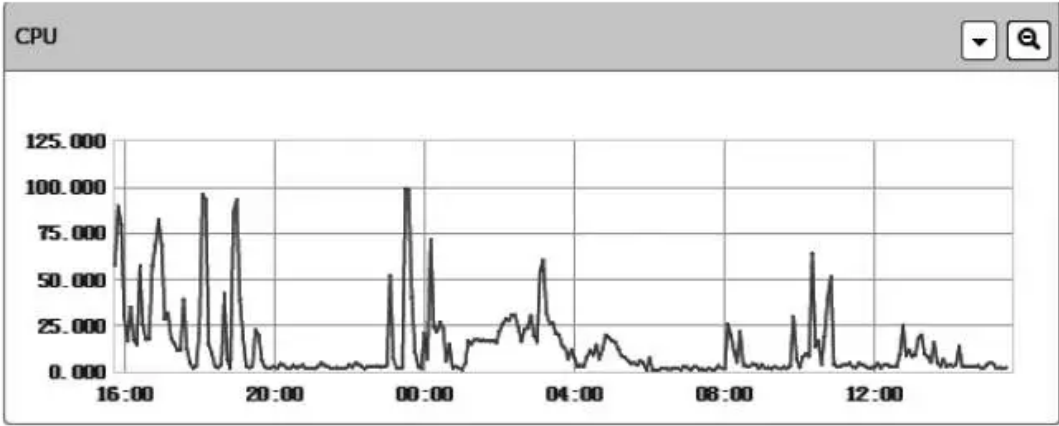


图4-1 CPU使用率

了解到的业界一般的计算方法是：

- ◎ 定义每天的业务高峰期为 10:00AM—10:00PM，求这段时间平均值表示该台服务器当天的使用率；
- ◎ 每天取峰值的 3 个点，求这 3 个峰值点的平均值表示该台服务器当天的使用率。

对于定义业务高峰期，然后求平均值，我们认为不够精准，很多业务不一定白天时间达到它的峰值。如果每个业务都个性化定义高峰时间，工作量比较大，管理起来也很麻烦。每天取峰值的 3 个点求平均值，很多时候人工操作或者临时性的计算很容易把这个峰值提得太高。

经过多次讨论后，确定了下面的计算方法：

- （1）每台服务器的 CPU、内存、磁盘、IO、网卡数据每 2 秒钟采集一次；
- （2）每天分别取这些数据的 TOP n%求平均值。

如图 4-2 所示为某台服务器某天的监控数据，我们计算出 CPU、内存、磁盘和 IO 的数值如表 4-1 所示。

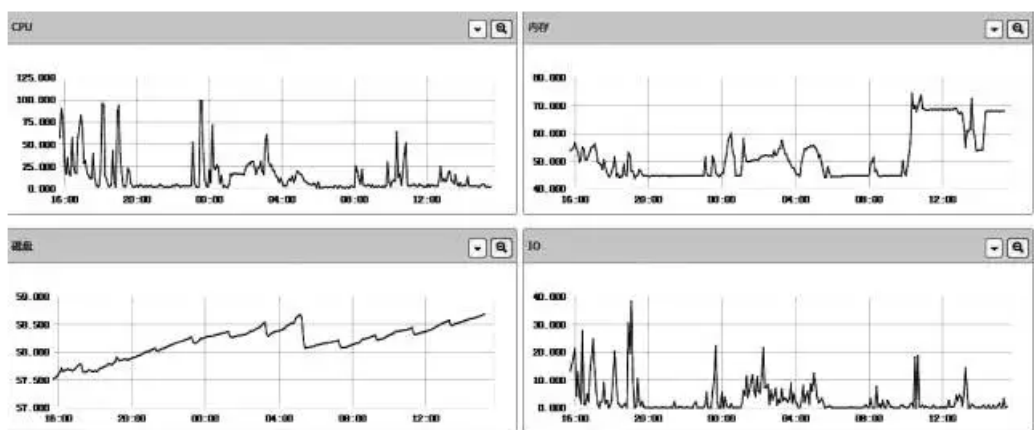


图4-2 监控数据

表4-1 资源数值

资源项	数值
CPU	98.12%
内存	73.29%
磁盘	58.69%
IO	29.05%

当然，怎么计算这个资源数值都没有问题，只要是统一的计算模型，能够表示所有机器的资源占用指标即可。我们这么计算是希望能够更接近服务器的平均峰值，消除一些临时性尖峰导致数值较大的问题。

指标展现

完成了资源使用率计算，但如何整体展现所有服务器的资源使用率也是一个问题。由于业务特性的不同，有消耗 CPU 的，有消耗内存的，也有类似离线存储消耗磁盘空间的，不可能把所有服务器的某项值加起来，然后求平均值，这样说明不了任何问题，而且值会低得可怜。在一个案例中，公司所有服务器每天的 CPU 值求平均值，CPU 使用率只有 23%左右。所有人看到这样的数据都会问一句话：“为什么我们的服务器资源使用率这么低？”

对于如何体现公司整体的服务器资源使用率这个问题，我们也没有什么好的方法，不过可以

从另外两个维度去体现。

(1) 将整个公司这个维度拆解到最小的相同功能集群的粒度，统计和展现这些小集群的资源使用率情况。这个时候就需要结合之前提到过的“服务树”，我们通过服务树将公司的产品划分到产品线->子系统->集群这样的粒度，每个小集群的功能是类似的，这群服务器的资源使用模型也是类似的。研发或运维在提交采购预算时，以最小集群为单位。领导在审批预算时，可以很方便地查看该集群的资源使用情况，决定是否购买。

(2) 每月统计出资源使用率不达标的服务器，发给相关的研发主管和运维人员，督促他们尽快利用或者归还资源。比如某个集群每月不达标的服务器数量超过该集群的 5%，则认为这个产品线资源使用不达标。

这样侧面解决了如何整体展现公司资源使用率的问题，给出最小集群资源使用率不达标的情况，展现的数据量比较少，而且能够很好地跟进和解决。还可以给出资源使用率不达标服务器和所有服务器的占比曲线，用来观察整个公司服务器资源使用率的变化情况。

这样既满足了工程师需要了解具体信息，执行改进的需求，也满足了老板需要了解整体信息，把握全局的需求。

怎么计算、怎么展现才算合理，每个公司都有自己的做法。当然，在你认为公司整体资源使用率都还不错的前提下，所有服务器的平均值也许可以作为一个基准，用来宏观监控整体资源使用率有没有变好或变坏也是挺不错的。

不达标原因分析

完成上述工作后，我们开展了一次资源使用率未达标服务器原因分析，按照未达标的原因进行了归类。如图 4-3 所示。

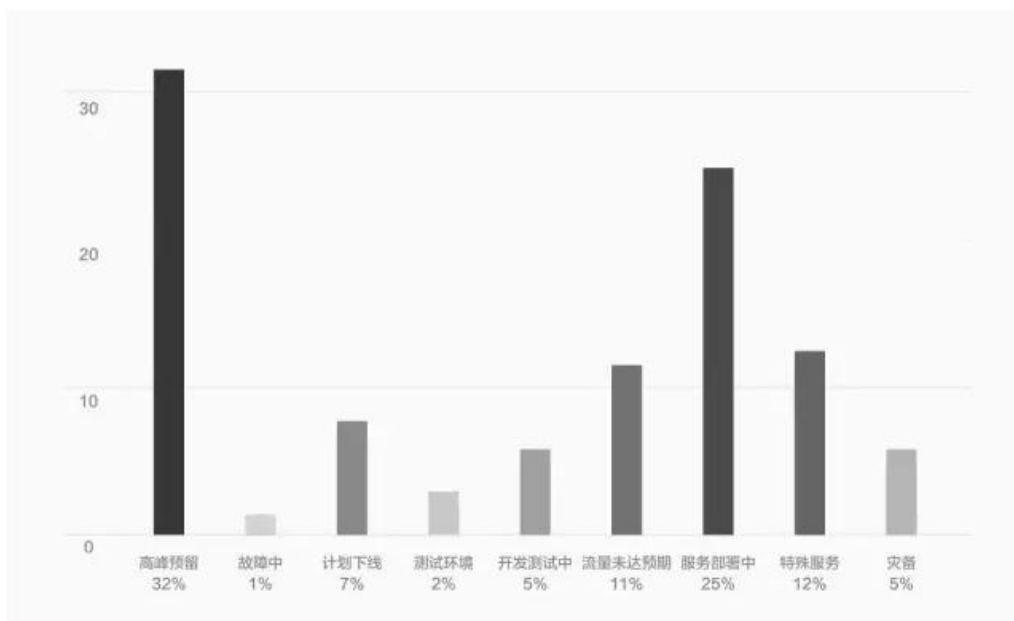


图4-3 资源使用率未达标原因归类

每类原因说明如下：

- ◎ 灾备：该类型服务器属于对重要服务的备份服务器，平时没有或有很少的流量。
- ◎ 特殊服务：该类型服务器属于特殊用途需要独占或者多副本存在，如 ZooKeeper，或类似博客对外服务又需要安全隔离的业务等。
- ◎ 服务部署中：服务扩容、搬迁时，暂时未引入业务流量。
- ◎ 流量未达预期：业务已经在线上运行，但由于流量预估不准确，服务上线后流量和计算量未达到预期。
- ◎ 开发测试中：新服务小流量运行，处于开发联调阶段。
- ◎ 测试环境：研发或测试的线下集群，用于压力测试、功能测试等。
- ◎ 计划下线：服务优化、架构调整、业务功能裁剪等，空闲服务器处于待下线归还状态。
- ◎ 故障中：服务器故障停止业务，处于维修状态中。
- ◎ 高峰预留：为了应对比如业务促销活动、特殊节假日等流量高峰，提前储备的服务器资

源。

下面分析导致前几类原因出现的问题。

1. 业务线预算不准确

◎ 预估的业务流量较大，实际业务流量未达到预期，导致已经上线的服务器资源浪费。

◎ 项目计划不准确，原计划 3 月份进行新服务上线或者扩容，结果项目延迟到 6 月份，导致 3 个月时间的服务器资源闲置。

2. 服务器采购效率低，缺少快速伸缩的能力

◎ 随着公司对于服务器需求的数量越来越大，很早以前那种随用随买的模式已经不适合了，转而采用服务器集采的模式，从预算申请，到领导审批、采购招标、供货商送货、服务器上架、系统装机交付等多个环节，在正常情况下会耗时 2 个月。那么业务部门往往会提前并较多地购买服务器，造成一定时间段的服务器资源浪费。

◎ 为了应对某次突增几倍的业务促销活动流量，需要提前准备服务器资源。当活动结束后，服务器资源的归往往不及时，归还后其他业务部门消化这部分资源的时间也比较长。

◎ 服务器采购时资产是划分到各个部门的。服务器资源归还后，还存在部门间资产更新计算的问题。

3. 缺少资源监控平台

缺少资源监控和审查机制，业务部门浪费的机器也不会及时归还。

资源优化

看到这几类原因后，第一时间能够想到的解决方案就是公有云。类似 AWS 的服务，能够快

速地交付或归还虚拟机资源，按时间、按流量收费。虽然大部分业务是可以部署在公有云上的，但还有很多类似 HBase、离线计算等对磁盘 IO、内网带宽有较高需求的业务，不太适合在虚拟机上部署，成熟公司这部分业务的服务器数量一般占到 30%~60%。

先不讨论公有云是否真正省成本，仅就对目前国内云服务商的安全评估，我们还不把高敏感、高安全要求的用户数据、现金支付类业务放置在云端。另外，多机房服务冗余、机房间专线互通等各种因素也需要考虑进去。

我们的目标是通过缩短服务器采购周期，资源能够具备快速伸缩的能力，结合预算审批、资源监控等手段，将不达标服务器控制在合理范围内。综合考虑后，可选的解决思路如下：

- (1) 支持物理机租赁模式，提高服务器交付效率；
- (2) 支持私有云和公有云的使用，降低物理机的需求；
- (3) 统一预算平台，限制不达标业务的申请；
- (4) 资源监控，不达标服务器资源能够及时归还。

作为一些互联网公司，不希望投入过多的资源在固定资产上。运维人员也不愿意投入过多的精力在服务器的采购、上架、维保以及后续的报废处理上。当前已经有很多公有云支持虚拟机和物理机的租赁业务，传统的 IDC 厂商也陆续提供物理机租赁业务，由他们负责服务器的采购、上架等一系列工作，按照我们要求的时间、地点（IDC、机柜位置等）进行交付。公有云或服务器外包公司提供足够的备机池资源，能够应对我们对于服务器快速伸缩的需求，按照实际租赁时间收费。

成本方面，按照服务器 3 年淘汰，每次物理机采购实际是预付了 3 年的使用费用。采用租赁的方式按实际使用时间付费，不需要提前支付未来的费用。对于项目计划不合理、预算不准确、高峰期预留等问题，我们可以通过快速的服务器交付、归还的方式缓解这些问题，将成本支出降到最低。当前，足够的备机池、快速的交付时间是有成本支出的。

预算平台和资源监控的结合，限制和发现那些资源使用率不达标的服务器，督促业务线及时归还资源，减少成本支付。

提升服务器资源使用率还有很多方法，比如服务模块混布、资源闲时利用，这些属于运维可控的范围。另外，研发对于代码和功能的优化，效果往往会更加明显。

四、网络成长之路

互联网公司的生存根本是服务安全、高效、稳定，这三个条件也是互联网运维从业者的工作基础，而这些基础工作绝大部分都依赖于底层网络服务的健壮性。

互联网行业的网络服务与传统行业的网络服务还是有所区别的。

传统行业由于业务单一。增加趋势平稳，所以对于网络只有安全性和健壮性的需求；但互联网行业的特点是服务多元化、业务无规律增加、热点内容等，所以互联网行业的网络服务除了具备安全性、健壮性特点外，还需要支持较高的扩容能力和灵活的按需变化能力。

初期

公司创业初期，一方面没有专职的网络运维人员，另一方面对未来网络容量规划缺少可以参考的历史数据（服务器数量、带宽数据等），所以第一阶段的网络建设是在对未来规划毫无数据支撑的情况下进行的。并且这里有个更重要的因素就是在公司产品还没有盈利的情况下，基础建设往往都会处于滞后状态，这应该也是绝大部分创业公司都会面临的问题。

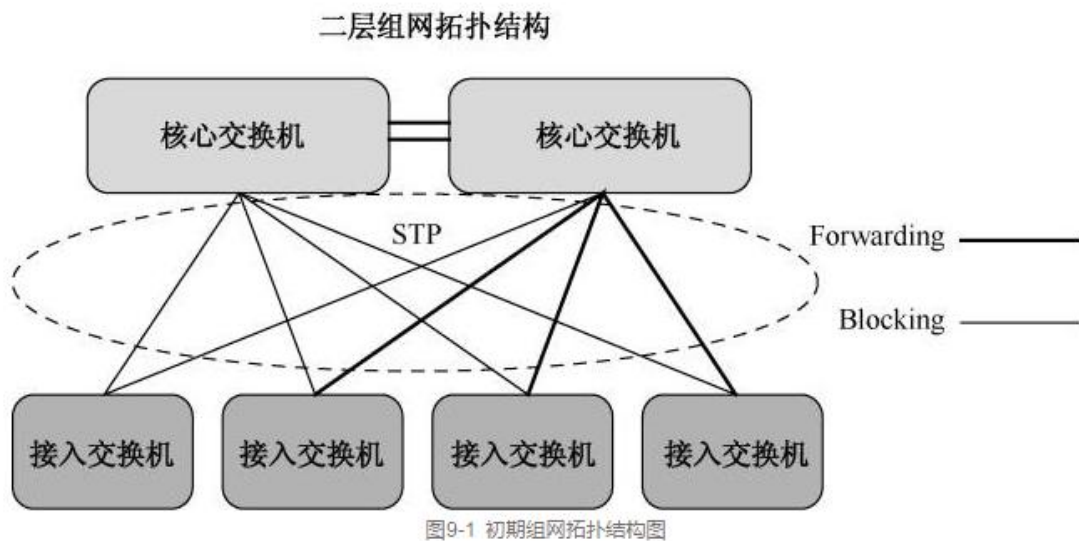
在一穷二白的时候对网络方案的选择更看重以下三个因素。

- ◎ 组网方案的成熟性。
- ◎ 可靠性（主要依赖产品）。
- ◎ 运维便利性。

组网方案和产品选择

初期网络产品自身的稳定性决定了整体网络的稳定性，选择一家业界靠谱的网络设备厂商如 H3C、Cisco 等，是保障初期网络稳定性的必要条件。

初期组网拓扑结构图如图 9-1 所示。



第一阶段的网络结构采用了传统分层的 Layer 2 组网网络，物理上区分了互联网接入层、核心层、接入层设备。自上而下，我们先看互联网接入层。

互联网接入层

互联网接入层设备可以选择路由器，也可以选择三层交换机，这里就不展开介绍这两种设备的功能区别了。在某案例中，第一阶段的互联网接入层设备采用的是三层交换机，主要基于两个因素考虑。

（1）端口密度：在流量未形成规模时，同时对互联网接入层设备没有过多功能要求时（初期只运行简单的静态路由），三层交换机的端口密度指标作为优先考虑的条件。

（2）成本问题：同样的成本，端口使用量是路由器的几倍，三层交换机的性价比更高。通常在数据中心接入互联网线路时，线路提供商可以提供如下几种不同的接入方式（不同的互联网供应商可能提供的方式有所区别）。

- ◎ VRRP 主备线路
- ◎ ECMP 双主线路

在初期网络建设中，互联网接入层设备采用的是双上联接口 VRRP 主备方式，在互联网流量较小的情况下基本可以满足对业务带宽和冗余的需求。

关于互联网接入线路的选择，由于不同数据中心的规模和承建方、人力投入、技术支持、运营能力都不一样，导致数据中心提供的线路也有所区别。

国内比较常见的现象是，一级运营商直接提供数据中心资源，那么数据中心内只能有一级运营商的线路资源。例如，中国联通的数据中心只能提供中国联通的互联网线路。在由二级运营商承建或由不同的一级运营商联合建设的数据中心（此类较少）是可以提供多线或多种单

线线路资源的。

线路资源可以分为如下两类。

- ◎ 单运营商线路：中国联通、中国电信、中国移动等国内一级运营商线路。
- ◎ 多运营商 BGP 线路：双线、四线、六线、八线国内二级运营商提供的线路。

单运营商线路由于是一级运营商直接提供的，所以单线资源在质量、扩容、故障、成本这 4 个方面要略优于多线资源。但由于中国各运营商自治和网间结算流量费昂贵，导致业务层面需要使用额外技术手段（如 DNS 调度、全局负载均衡、CDN 等）来提升不同区域和不同运营商用户的访问质量。

多运营商 BGP 线路是由二级运营商和各一级运营商建立的 BGP 邻居关系，将自有 IP 地址通过 BGP 在已建立邻居关系的一级运营商线路内广播。从应用角度来看，可以轻松地完成 IP 多线路的运行模式，不受骨干网各运营商之间的访问限制，提升了应用访问的速度和品质。

目前国内家用宽带接入运营商以中国电信、中国联通为主，例如，主营客户端游戏的公司通常选择双线（中国电信、中国联通）的线路为主。但随着 2008 年中国移动和中国铁通的合并和移动互联网的兴起，移动线路和教育网在各数据中心中所占的比重也越来越重要了。各运营商流量占比如图 9-2 所示。

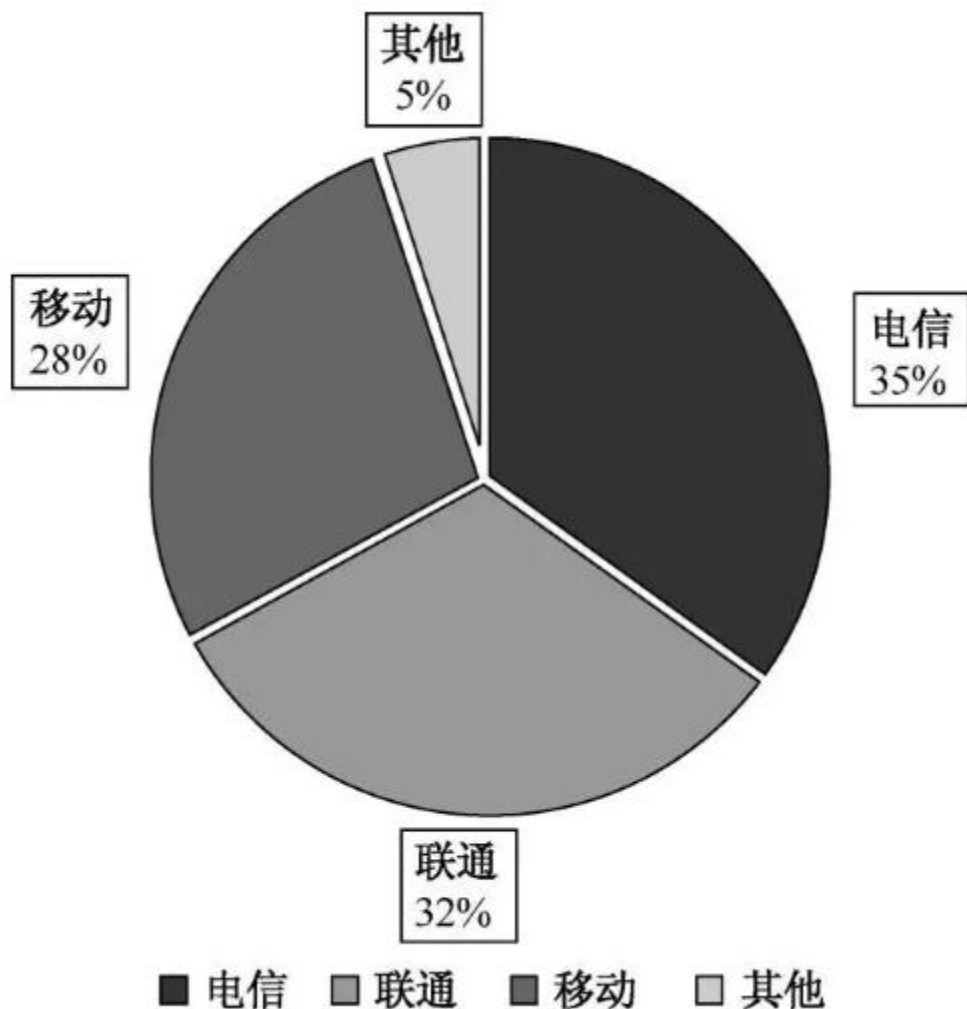


图9-2 各运营商流量占比

在用户至上的时代，显然在初期使用多线 BGP 线路这种复合型线路资源解决用户的问题是立竿见影的。但随着业务规模的扩张、调度系统的完善，以及对成本方面的考虑，多线 BGP 线路的弊端也逐步显现出来。

首先就是成本的压力，随着用户量的增加，带宽数量也呈现突飞猛进的增长态势，以四线 BGP 线路和单运营商线路为例，它们的成本差距在 5 倍左右，在强调“节能环保”的今天，这种成本的差距显然是不可接受的。

其次是故障率和冗余切换能力，由于多线 BGP 线路主要是二级运营商承建的，从网络角度来看，多一跳就意味着多一个故障的隐患。

再者是冗余切换能力，多线 BGP 线路在冗余切换上分为两类，业内称为真 BGP 多线和伪 BGP 多线，两者的区别在于真 BGP 多线实现了各大一级运营商线路路由的互相备份，当多线 BGP 线路中其中一个运营商线路出现故障时，故障运营商线路上的路由可以通过其他线路进行绕行访问；然而，伪 BGP 多线则只能实现多线路的用户访问覆盖，无法做到各线路路由的互相备份，当伪 BGP 多线其中一个运营商线路出现故障时，网络工程师能做的只是等待运营

商进行故障修复。

还有就是多线 BGP 线路定向扩容能力。多线 BGP 线路的优势在于它具备优秀的线路整合能力，将原本复杂的互联网环境简单化。这种复合型线路资源势必会产生多资源配比的问题，例如，我们购买了 1Gbps 的四线 BGP 线路，这是不是代表这 4 个运营商线路我们单独都能跑到 1Gbps 呢？

答案肯定不是这样的。通常 BGP 带宽资源池中的中国电信、中国联通的带宽会比较富裕，但一些相对占比较少运营商的带宽往往会很少。比如某真实案例中，业务移动端流量爆发，导致移动运营商流量突增，从带宽监控图上看带宽 Buffer 空间仍然有很多富余，但论坛总是能收到用户投诉超时、连接不上服务器等问题。

最后梳理用户投诉发现均为移动用户，协同二级运营商排查后才确定为是由于二级运营商中的移动线路带宽瓶颈导致的。对于这个问题，由于多线 BGP 线路每家提供商的资源都有所不同，业界也没有一个固定配比原则，所以这部分依然是盲区，不容易规避。

核心接入层和服务器接入层

初期在对业务发展毫无概念的情况下，选择一套简单的通用的组网方案和使用相对靠谱的网络设备是保障网络可用性的两个关键条件。

初期核心接入层和服务器接入层采用 Layer 2 传统二层互联的方式。这种 Layer 2 传统组网方式，接入交换机后端服务器的默认网关都在核心交换机上运行，核心交换机采用 VRRP/HSRP（热备份冗余路由协议）提供一个虚拟地址为服务器默认网关。

核心接入层和服务器接入层运行 STP（生成树协议）提供服务器接入层交换机双上联核心交换机冗余性。STP 协议需要生成一个无环并且冗余的网络拓扑，基于 STP 协议的特性，需要通过算法在核心和接入层之间选择一个接口将其置为 Blocking 状态，在默认情况下使用 Forwarding 状态的接口进行数据转发，当 Forwarding 状态的接口出现故障时，则再次进行 STP 计算，然后将原 Blocking 状态的接口置为（默认 35 秒完成）Forwarding 状态后恢复数据传输。

Layer 2 传统组网方式提供了便捷的服务器接入能力，服务器只需要接入交换机接口，并将接口归属到指定的 VLAN 即可完成服务器上线。但如果在此阶段没有良好的 IP 地址规划，以及在业务爆发期无规律地上线服务器，再加上某些特殊的应用需要在二层环境下才可以工作，如数据集群（Keepalived）、LVS-DR 模式等，这样的需求和现状会导致需要频繁的对接口 VLAN 信息进行变更，工作量巨大。

为了避免此类操作，初期网络采取了比较粗暴的方式进行处理，在核心交换机 SVI（Switch Virtual Interface）以辅助地址的形式运行多 IP，这样可以解决因服务器上线和接口变更所带

来的网络设备的变更问题（不过因此也为后期埋下了祸根）。

```
interface vlan 100
```

```
ip policy route-map LVS-DR（坑 1）
```

```
ip address 192.168.1.254 255.255.255.0（坑 2）
```

```
ip address 10.0.1.254 255.255.255.0 secondary（坑 3）
```

```
ip address 10.0.2.254 255.255.255.0 secondary
```

大体上来看，这时的网络结构满足了业务需求，并且通过“技术手段”减少了烦琐的服务器上线和其他业务需要的网络设备变更问题，弥补了初期人力不足的短板。但实际上该阶段的网络规划并没有看起来那么美好和健壮。

比如在该案例中，使用了 192.168.1.0/24 地址给忘了运维带来了困扰。现在做 IP 地址规划大家一定不会选择 B 类 192.168.0.0/16 私有地址作为内网的主要地址。原因不止是 B 类地址比 A 类地址可使用的地址数少这么简单。这里面所带来的问题是，如果数据中心存在 192.168.1.0/24 这个网段的服务器，我们如何通过远程 VPN 的方式管理这台服务器呢？因为这个地址段默认被 D-Link、TP-Link 这些家用无线路由器厂商当作初始化的 IP 地址段。当我们使用 PPTP 或 L2TP 远程拨号 VPN 远程访问目标服务器时，由于本地直连路由比静态路由的优先级更高，所以访问远程目标服务器的数据不会通过 VPN 发往数据中心目标服务器，数据包会在本地局域网进行二层查询。

上面这个小问题只是让远程办公的同学们有些不爽而已，还没有真正地对生产网络产生影响。下面两个问题对初期网络造成了不可小觑的影响，一个是单 SVI 多 IP 工作模式引起的问题，二是 LVS-DR 工作模式引起的问题。

网络设备对不同的数据处理分别是基于控制层面和转发层面进行的。这部分内容在网络设备厂商官网有大量的介绍，这里就不详细讲解了。简单来说，进入数据控制层面的报文主要靠 CPU 能力进行处理，转发平面主要靠硬件能力进行处理。

控制层面：控制层面就是各种协议工作的层面。

它的作用是控制和管理各种网络协议的运行。控制层面主要处理两类报文，一是抵达设备自身的，如 Telnet、SSH、SNMP；二是负责网络选路和拓扑变化所使用的协议，如生成树协议、动态路由协议、组播协议，这些都是由控制层面直接进行处理的。控制层面的处理能力依赖设备自身的 CPU 性能，所以在网络优化中也需要对控制层面进行特殊的防护，如 CoPP（Control Plane Policing，控制层面策略）。总结：控制层面通过软件（CPU）处理，效率低。

转发平面：转发平面的工作主要是对穿越网络设备的报文进行处理，Layer 2 数据包、Layer 3 数据包、访问策略、QoS 这些都属于转发平面的任务范畴。转发平面主要依赖设备接口芯片的处理能力。总结：转发平面通过硬件（芯片）处理，效率高。

在初期，某案例中，应用时常出现超时和访问失败等问题，排查后确定是由于核心交换机 CPU 负载高引起的（见图 9-3），那么到底是什么原因引起的 CPU 负载高呢？

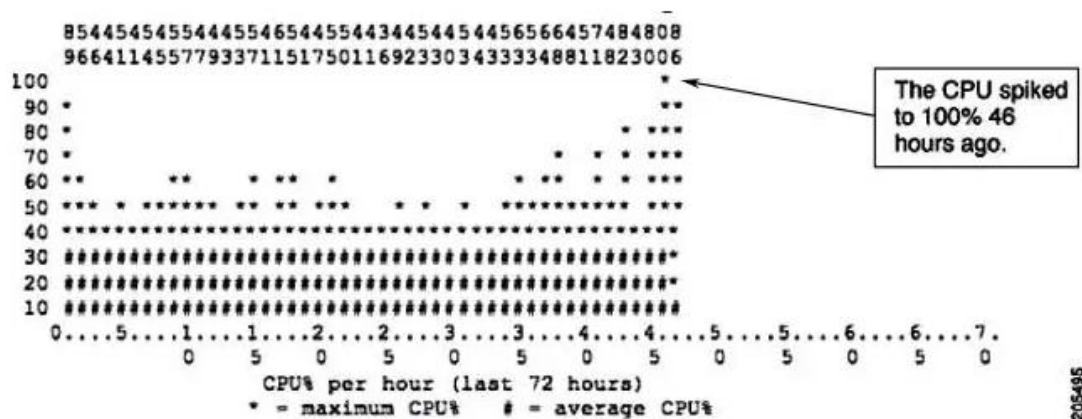


图9-3 核心交换机CPU负载高

原因一：

交换机在进行三层转发时，如果路由下一跳在一个二层网段或者在相同的 VLAN 下，则会产生 ICMP Redirects 报文，这种报文是直接进入交换机的控制层面处理的，这类流量由交换机 CPU 直接处理。单 SVI 多 IP 工作模式会过度产生直接由交换机 CPU 处理的报文，这是导致核心交换机 CPU 负载高的一个主因。

High CPU Due to Excessive ICMP Redirects

You can get ICMP dropped redirects when one VLAN (or any Layer 3 port) receives a packet where the source IP is on one subnet, the destination IP is on another subnet, and the next hop is on the same VLAN or layer 3 segment.

原因二：

由于 LVS-DR 工作模式的特性，Real Server（后端目标服务器，后续简称 RS）在进行响应时，需要使用服务器自身 Lo 绑定的 VIP 地址进行回包。LVS 后端服务器存在两种部署模式，第一种为后端 RS 配置公网 IP；第二种为后端 RS 配置内网 IP。第一种模式下可以依赖后端 RS 自身的默认网关完成数据包的回包工作；第二种模式下则需要网络层通过 PBR（Policy-Based Route，策略路由）对流量进行特殊的处理。

由于第一种部署模式会将后端服务器直接暴露在互联网上，存在安全风险，所以当初出于安全的考虑，后端服务器采用了第二种部署模式。这就导致需要在网络层使用 PBR 对这部分特殊的流量进行处理。某些低端核心交换机在开启 PBR 这个特性时存在一些限制，会导致导致核心交换机 CPU 负载过高。

High CPU Due to Policy Based Routing

Policy Based Routing (PBR) implementation in Cisco Catalyst 3750 switches has some limitations. If these restrictions are not followed, it can cause high CPU utilization.

- | You can enable PBR on a routed port or an SVI.
- | The switch does not support route-map deny statements for PBR.
- | Multicast traffic is not policy-routed. PBR applies only to unicast traffic.
- | Do not match ACLs that permit packets destined for a local address. PBR forwards these packets, which can cause ping or Telnet failure or route protocol flapping.

I Do not match ACLs with deny ACEs. Packets that match a deny ACE are sent to the CPU, which can cause high CPU utilization.

I In order to use PBR, you must first enable the routing template with the sdm prefer routing global configuration command. PBR is not supported with the VLAN or default template.

但随着大数据和分布式计算的应用，业务层面给网络带来的挑战还远远不止这些。分布式计算类型的服务会存在大量的数据交互和复制，其特点是：高带宽、大突发。它们在一定时间里被我们亲切地称之为“内网杀手”。以 Hadoop 为例，从图 9-4 可以看出，用户在推送数据块时数据节点之间的复制操作需要频繁地消耗接入层交换机的上行带宽，在初期上行收敛比还是 12:1 时（8 口千兆上行，48 口千兆下行），这部分流量必然会给相同机架的其他服务造成影响。

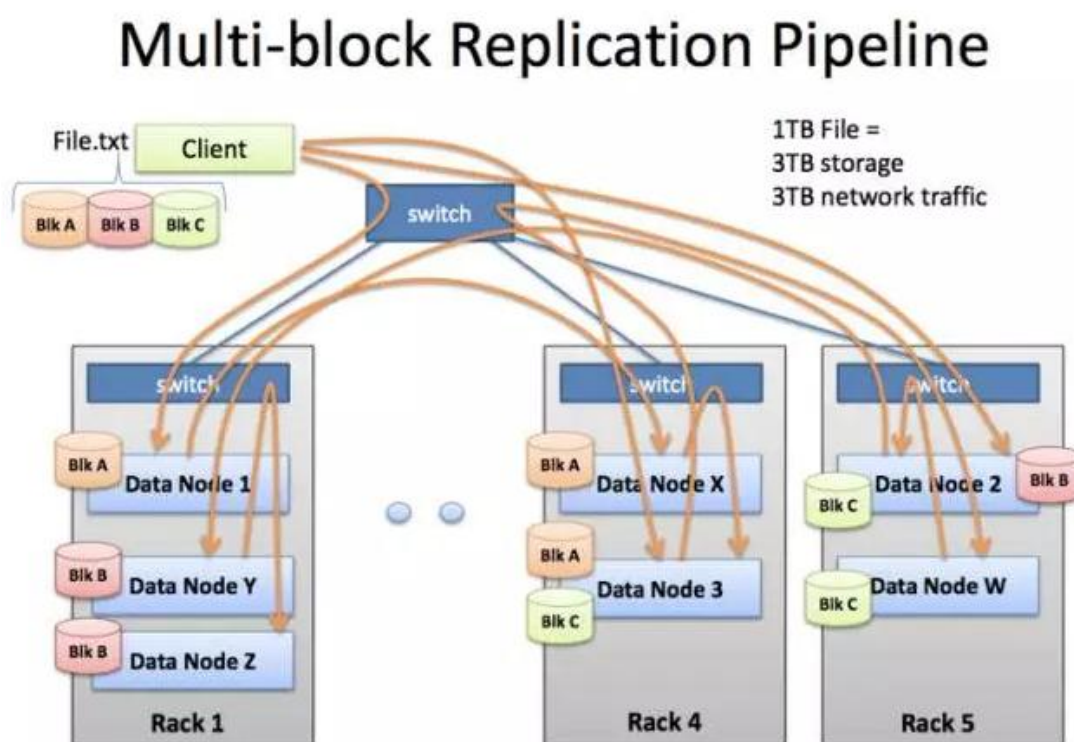


图9-4 分布式计算大量消耗内网上行带宽

针对此类业务的大数据交互的特性，在该案例中，做了以下两个方面的调整。

- ◎ 优化数据节点，减少跨机柜的交互。
- ◎ 扩容接入层交换机，达到核心上行带宽。

刚才我们谈到收敛比时，应该有读者看出初期阶段核心和接入层之间存在的问题。8 口千兆上行收敛比应该是 6:1，为什么这里写的是 12:1 呢？就如之前所介绍的一样，核心和接入层之间运行的生成树协议（STP），天生就会浪费 50% 的带宽资源。STP 会带来两个问题，一是资源浪费，二是冗余切换能力，如果当上行接口聚合组其中一个接口出现异常关闭时，那么接入层交换机的上行收敛比就又会减少 1/4，并且这时 STP 并不会重新进行拓扑计算。

我们在处理 Hadoop 大流量的问题时，意识到 STP 这种古老的协议已经不适用于当前数据中心网络了。
提升收敛比的首要任务就是将 STP 备份线路充分利用起来。

业界替代 STP 的组网方式有很多，如 Layer 3 组网、网络虚拟化、大二层。但这种在原有网络基础上改造的方式，不同于新建，更多的是需要考虑原有设备利用、割接窗口时长等客观因素。

所以在考虑上述因素后，改造方案选择使用网络虚拟化的方式替代 STP 组网方式(见图 9-5)。这样可以在不更换接入层交换机的情况下，将原有收敛比提升一倍，并且很好地规避了前面提到的 STP 存在带宽浪费、冗余切换的问题。

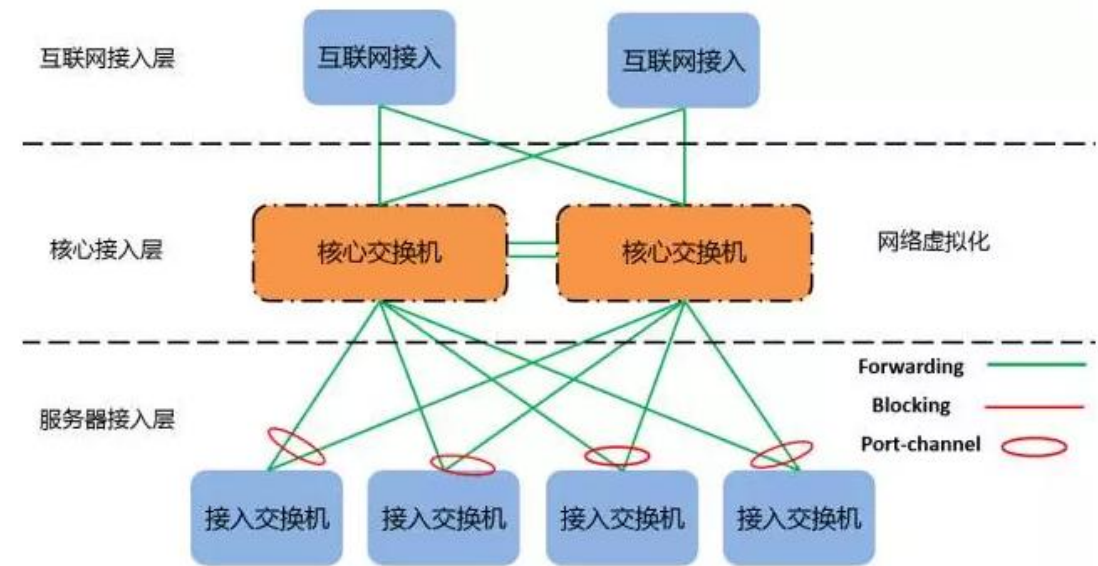


图9-5 网络虚拟化

网络虚拟化技术有两个技术类别分支，分别是整机虚拟化和接口虚拟化。

整机虚拟化

以 Cisco VSS(Virtual Switching System) 和 H3C IRF(Intelligent Resilient Framework)为代表，这类整机控制层面虚拟化已经是比较成熟的技术，已有很多商用案例。控制层面虚拟化提供了两种工作模式，分别是一虚多和多虚一，分别应对不同的场景。

多虚一的主要作用是简化网络结构，提升网络横向扩展能力。通常是两台设备进行多虚一，但目前也有部分厂商实现了 N:1 的整机虚拟化，如 H3C IRFv2 (Intelligent Resilient Framework 2，第二代智能弹性架构) 技术目前最多可 4 台核心设备 N:1 虚拟化，H3C 等较为主流的交换机产品线均支持 IRFv2 技术 S1250、S9500E、S7500E、S5800 等。一虚多技术多数用在虚

拟化多租户环境下，整体提升硬件资源的利用率。此类技术多使用在金融和虚拟化公有云环境中。

接口虚拟化

以 Cisco VPC(Virtual Port Channel)和 Arista MLAG（Multi-Chassis Link Aggregation）为代表，接口虚拟化和整机虚拟化在网络结构上没有太大的变化，都进行了跨设备的接口聚合操作。但由于接口虚拟化只关注跨设备的接口聚合，所以在核心层面依然需要使用 HSRP/VRRP 技术提供冗余支持。

在实际部署中对接口虚拟化各家厂商限制的条条框框较多，相比 IRFv2 整机虚拟化技术部署起来比较复杂，反观整机虚拟化提供最多 4 台核心设备的虚拟化能力，接口密度为乘的关系，去除了复杂的 VRRP/HSRP 配置和大量的地址配置，逻辑上只管理和监控一台设备，提供了良好的管理便利性（见图 9-6）。

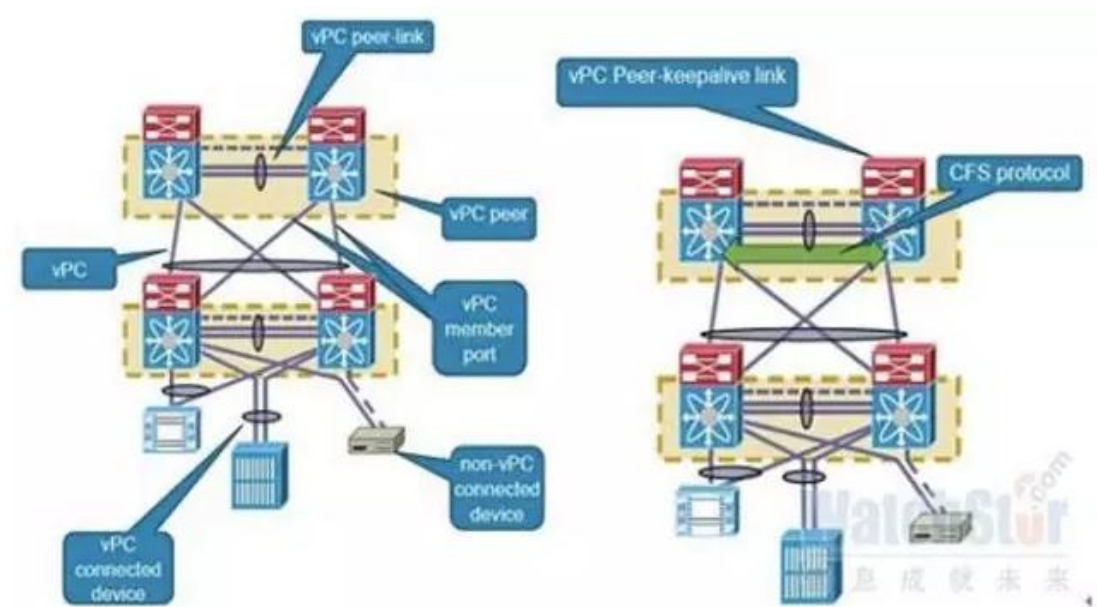


图9-6 整机虚拟化和接口虚拟化部署图

基于以上的分析，在应对初期 Hadoop 高带宽需求时，我们的解决方法是通过替换核心交换机设备，提升一倍的带宽收敛比，短暂性满足带宽需求，并且尽可能地将 Hadoop 业务放置在单独的机柜，以防止影响到其他业务。

初期网络运维总结

在没有明确需求做数据支撑的前提下，组网方案只停留在了简单的设备堆砌阶段。应用存在不确定增长或大规模爆发情况时，对基础网络造成的冲击非常大。网络运维工程师这时主要

承担了救火队的角色。

但这一阶段也有不少收获：第一，逐渐摸清了公司业务流量模型和特点，为后续的规划积累了数据；第二，通过不断的网络平台扩容、改建，总结出以下两条真理。

◎ 改造的难度远大于新建。

◎ 对于增长不确定的网络规划，核心交换机的规格和投入适当超前。

中期

在该案例中，总设备数量的增长，推动了建设新数据中心的需求，让网络工程师有机会在“白纸”上重新构建基础网络，规避初期遇到的问题，重新梳理需求。网络建设和数据中心规模有着密切的关系，如原计划建设一个约 500 台服务器规模的数据中心，网络的设计容量为支持 1000 台设备，但如果业务增长超过数据中心规划，这样就导致又要去规划新的数据中心。

这种情况所带来的问题有以下几点。

◎ 大量小规模数据中心管理复杂。

◎ 数据中心网络设备投入成本大。

◎ 小规模数据中心间互联是难题。

当发现这个现象后，为了避免后期出现上述几个问题，采取了“脱壳前进”的数据中心扩容方式。当存在更大规模的数据中心时，将规模小的进行撤离融合到大规模的数据中心去，整体的网络结构、建设成本、运维成本均得到了提升。但这里痛点不在网络层面，而是在业务层面。在这个阶段大量业务还不支持双数据中心冗余切换，需要有损地进行业务迁移。所以需要推动业务架构支持双数据中心部署，为后期双数据中心冗余打好基础。

中期阶段的网络规划在业务分区、边界安全、线路层等方面都进行了相应的优化。同时业务部门也对网络安全提出了更明确的需求，如业务线间安全隔离、互联网访问安全隔离等。下面让我们来看看中期阶段到底为什么这样规划？还遇到了什么挑战？

组网方案

为将来建设支撑万台以上设备的大型网络积累经验，在中期规划了两套不同的组网方案，来对比各自的优缺点。但基础的业务分区、安全需求、LVS 区域的实现是一致的。中期组网结构如图 9-7 所示。

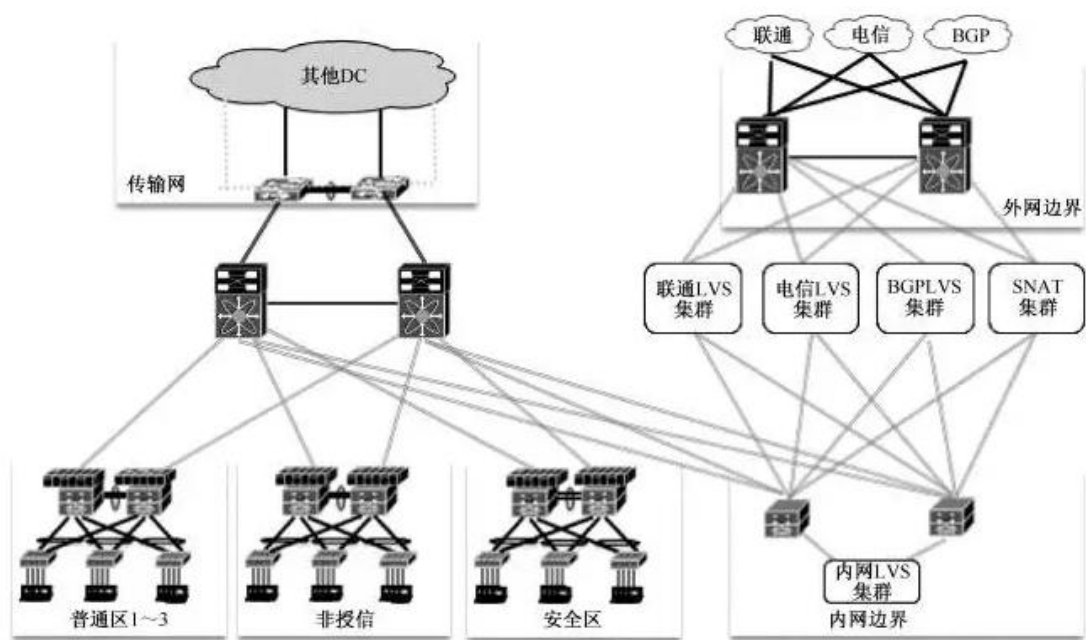


图9-7 中期组网结构图

我们在网络中明确了区域的概念，对公司的不同业务划分了区域。各区域使用不同的物理层设备进行物理和逻辑的双重隔离，做了 IP 地址段的分区规划，不同 IP 地址对应不同业务。这样做的好处是逻辑结构清晰，并且便于后续运维及权限划分。可以分为以下三大类：

- ◎ 互联网区域
- ◎ 内网区域
- ◎ DCI（Data Center Interconnection）区域

互联网区域

互联网区域示意图如图 9-8 所示。主要功能是接入多条运营商线路，各线路入网对应不同的负载均衡集群（LVS），出网对应 SNAT 集群。入网、出网实现统一化、服务化。

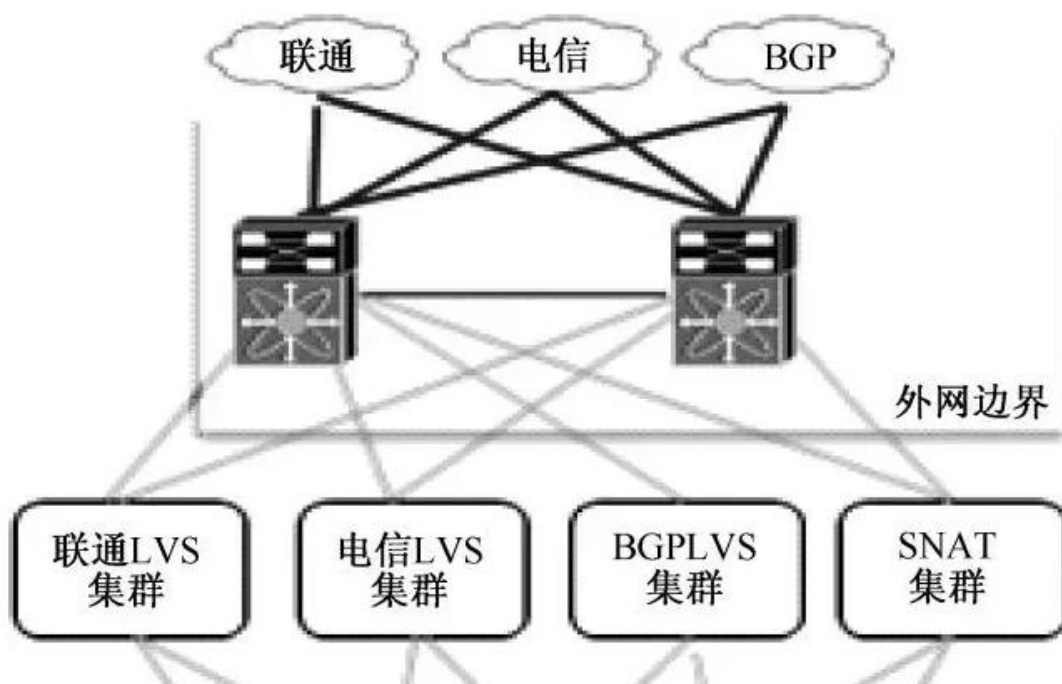


图9-8 互联网区域示意图

如前文所述，该阶段，业务对网络的可用性、安全性都提出了更高的要求，所以运营商线路接入都是采用双上联 ECMP 方式进行的，并且在每条运营商线路的入向接口提供互联网接口白名单的规则。

LVS/SNAT 集群和网络一起规划了 OSPF 的方式联动部署。LVS 也升级到了 FULLNAT 模式，相比 DR 模式对网络的要求降低很多。从网络角度来看，FULLNAT 的数据流是一个正常的从哪里来从哪里回的数据包，在网络层可以不用对这些报文做特殊的处理（策略路由）。这也极大地简化了网络的复杂程度，LVS 的部署和详细内容可以参考 LVS 章节。

在某案例中，此阶段，互联网线路上更多的是关注质量、故障、成本三个因素，所以我们设计多线路的 LVS 集群方式，并建议应用尽量运行在互联网单线资源上。这里分享一些线路选择和带宽预留的原则。

1. 互联网线路使用指引

- (1) 可调度业务的电信、联通、移动用户使用单线资源覆盖。
- (2) 可调度业务非三大运营商的用户使用多线 BGP 资源覆盖。
- (3) SNAT 默认只提供 BGP 出网线路。

2. 带宽预留原则

(1)物理接口: 物理接口使用总和的 50%(单 10Gbps 接口带宽使用范围为 5Gbps, 双 10Gbps 负载接口带宽使用范围为 10Gbps)。

(2) 逻辑带宽: 带宽 Buffer 数量为最近一周带宽峰值的 50%。

内网区域

在某案例中, 初期阶段在收敛比上吃了亏, 在中期阶段专门规避了这个问题, 收敛比初期设计为 1:2, 但随着采购量的增加和设备成本的下降, 逐步地扩容到 1:1。根据监控数据看非分布式计算业务机柜在 2:1 的收敛比下也可以良好地工作, 这和业务自身调优也有比较大的关系。但分布式计算业务如 Hadoop 机柜则建议 1:1 收敛。

在实际部署中内网区域使用了两套当时比较主流的组网方案, 这主要集中在核心接入层和服务器接入层之间。其中一种是 Layer 3 互联组网方案; 另一种则是以 Cisco Nexus 系列为代表的 Pod Design 互联组网方案。这两套组网方案各有优劣势。

1. Layer 3 互联组网

Layer 3 互联组网, 顾名思义, 是核心层和接入层运行 OSPF 或者 BGP 路由协议进行互联。业内一直有一种说法是, Layer 3 互联组网是一种临时数据中心组网的过渡方案。这种说法其实没有错。我们可以设想一下, 在核心设备不支持网络虚拟化或现在主推的大二层解决方案的年代之前(2009 年 Cisco Catalyst 6500 平台开始支持网络虚拟化技术 VSS), 除了这种 Layer 3 互联组网方案, 核心设备在当时是没有技术支持横向扩展的。

换个角度来说, 就是一个数据中心可承载的服务器数量其实局限于核心设备的接口密度。Layer 3 的解决方案在满足扁平化的基础上将数据中心可承载的服务器数量提升了 8~16 倍(主要参考 ECMP 的数量决定)。当然, Layer 3 互联组网方式也完全规避了 STP 产生资源浪费和冗余切换时间长的问题。

(1) Layer 3 互联组网的挑战

Layer 3 互联组网方式也对运维管理、成本提升、二层应用的部署带来了一定程度的挑战。

◎ 运维管理

首先, 运维管理的挑战是 IP 地址管理、基础服务应用、接入交换机上限三个方面。可想而知, IP 地址管理将变得很复杂。以 4 个核心为例, 每台接入交换机上都会配置 4 个 P2P (Point-To-Point) 互联 IP 地址和一个服务器网段。如果服务器使用共享 ILO (Integrated Lights-Out) 管理方式, 那么还将有服务器管理卡网段。在正常情况下每个接入交换机上将产生 5~6 个小网段。假设我们有 200 台接入交换机, 在没有良好的 CMDB 系统协助管理记

录的话，运维这种类型的网络难度可想而知。

其次，是基础服务应用。例如服务器上架和装机，因为每台接入交换机自治的原因，所以需要服务器上架人员和装机人员对此类网络有充分的了解。但在实际环境中往往是负责上述工作的人员和构建网络人员分属两个不同的组或者部门，所属不同的专业维度，要求他们对网络结构、IP 地址与网络工程师有一样的认识显然不太合适。最终就变成一些服务器上架和变更需要主机人员和网络人员配合，费时、费力，效率极低。

最后，是接入交换机上线。就如刚才所提到的 IP 地址管理的复杂性，在初期没有 Netconf 和 Poap（Netconf 和 Poap 实现网络设备加电后自动下发配置的功能）技术支持的情况下，一台接入交换机的初始化配置涉及大量的人工修改，配置的准确性和上线效率都潜在隐患。

◎ 接入交换机成本的提升

在传输二层环境下，我们进行接入交换机选型主要关注接口密度、接口 Buffer、背板带宽、转发能力、MAC 地址容量这几个性能指标即可。但对于 Layer 3 互联组网，我们在关注上述指标之外，还需要关注设备是否支持三层功能、路由表的容量等信息。这样势必会带来接入交换机的成本提升。这部分设备在数据中心基数较大时，产生的成本量级还是比较大的。

◎ 二层应用部署

典型的二层应用就是 VRRP（热备份冗余路由协议），如需要提供虚地址方式进行集群冗余的应用，均依赖此协议（如 Linux Keepalived）。这样导致依赖此协议的集群服务需要部署在相同的交换机上或者相同的机柜上。显然这样的部署方案在冗余上并不是那么完美。针对这种业务，在 Layer 3 互联组网方案中，通常的处理方式是将前后排背靠背两个机柜做成多虚一的虚拟机交换机，来满足不同的接入交换机和不同的机柜达到冗余的效果。这里还涉及一个现实的问题是，通常研发或应用运维提交服务器使用申请时是不会标注应用类型的。这样导致负责主机的人员在交付后还要针对使用这种类型的服务进行单独的机柜调整操作，涉及的工作量可想而知。

（2）Layer 3 互联组网的优势

不可否认，在特定的阶段 Layer 3 互联组网方案提供了良好的扩展性、冗余性、安全性，这也是此类组网方案的优势所在。扩展性主要体现在可以针对核心层面进行横向扩容；冗余性体现在支持多条冗余上行线路，并且有着良好的切换能力；安全性体现在众所周知的安全性最差的数据链路层上，减少了数据链路层广播风暴的可能性，降低了受广播风暴的影响，但是对特定应用存在一定的部署难题。Layer 3 互联组网方式在核心设备与接入层设备品牌选择上提供了差异性支持。接触网络较早的读者一定清楚不同厂商的设备工作在 STP 环境下会产生莫名其妙的问题。这种组网方式可以规避此类问题，让不同品牌的核心设备和接入层设备工作在网络中。

2. Pod Design 互联组网

Pod Design 互联组网方式，是 Cisco Nexus 交换平台上推出的组网方案，在物理结构上类似于传统的核心、汇聚、接入三层。实际部署的区别主要在汇聚和接入两层。传统的汇聚和接入

采用 Layer 2 或 Layer 3 进行互联。Pod Design 以 Pod 概念替代传统区域的概念，在结构上差异并不大，但功能上略有差别，传统区域定义除了划分区域外，区域汇聚可以提供安全、负载均衡、流量控制等功能，这主要依赖汇聚交换机的设备产品形态。但当前数据中心的网络需求已经从多功能向高带宽、低延迟方向转变，这些较为复杂的功能已经不适合当前数据中心的网络需求了。Pod Design 互联组网方案在这两层采用了俗称远端板卡接入方式（Fex），类似于在汇聚层和接入层进行纵向虚拟化，Pod 内所有接口的操作、监控、变更在 Pod 两台核心设备上上进行，简化网络结构、减少网络中可管理设备的台数。

（1）Pod Design 互联组网的挑战

Pod Design 的设计思想是将网络按模块进行归类，这种思想和我们设计网络时提到的 IP 关联业务线不谋而合，甚至是天生就实现的。但此类组网方式也同样面临着 Layer 3 组网所遇到的三个挑战。

◎ 运维管理

运维管理所带来的挑战是布线系统、Pod 容量、管理风险三个方面。传统的数据中心核心到接入层布线有两种方式，即 EoR（该方式指服务器机柜中所有的服务器接口，都通过跳线连接到机柜上的配线架，再由配线架上的铜缆延伸到网络机柜（位于一组机柜尾部）中的接入交换机）和 ToR（该方式将接入交换机放置在每个服务器机柜顶部，机柜内服务器直接通过短跳线连接到顶部的交换机，再经由光纤从交换机的上行链路接口连接到核心交换机）这两种方式的优劣势网上分析文章很多，这里就不做过多的分析了。

目前主流的是 ToR 布线方式。但 Pod 互联组网模式的布线方式还和 ToR 方式略有不同，主要是在汇聚交换机提供到接入交换机的接口方式上有所区别。传统的 ToR 方式汇聚和接入设备分别提供 4 个 10Gbps SFP+线缆接口互联接口，但 Pod Design 出于机架空间和端口密度的考虑，多数是提供如图 9-9 所示的 40Gbps MPO 线缆类型的接口。此时我们需要通过 MPO-to-41C 针对汇聚和接入设备线缆进行跳接。

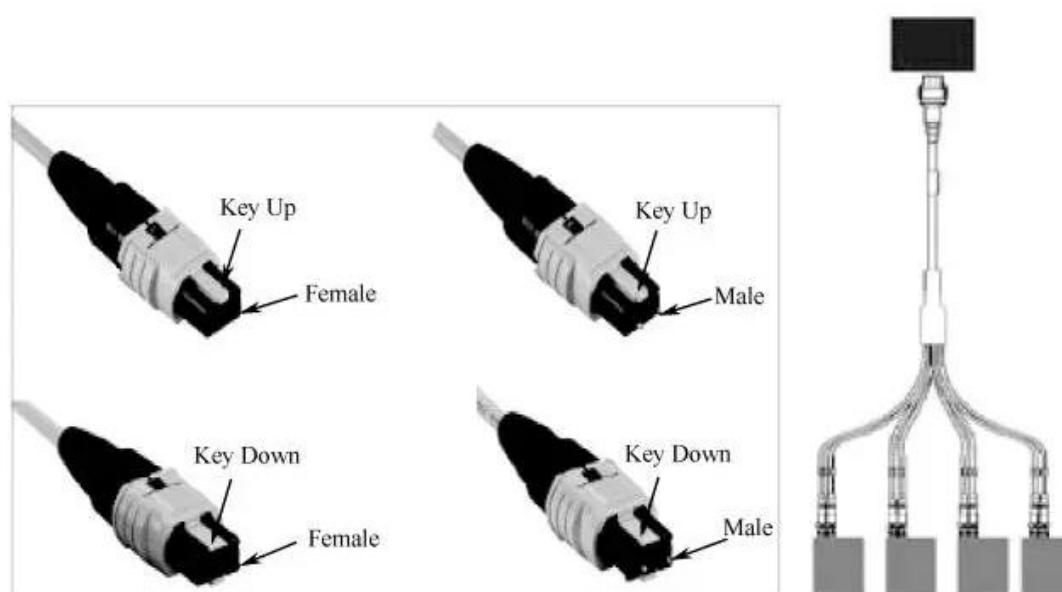


图9-9 40Gbps MPO线缆类型的接口

这带来了一个问题，就是线缆长度的问题。在设计此类型组网方式时，为了避免单一 40Gbps 接口故障影响过多的接入层上行带宽的容量，在实际环境中 40Gbps 接口是分别接入 4 台不同的接入层设备的，这样当汇聚单设备单 40Gbps 接口故障时，只是 4 台接入交换机上行带宽从 40Gbps 减少到 30Gbps 而已，这个带宽减少的比例还是可以接受的。

但根据机柜位置的不同，线缆长度也不同，在实际采购过程中发现 MPO-to-4lc 线缆都不提供定做 LC 侧超过 10m 以上规格的线缆。在部署和线缆备件的处理上都比较麻烦，所以采取的是汇聚交换机机架顶部配线架的方式，如图 9-10 所示。



两个配线架以背靠背 48 端口 LC 方式对接，在这样的布线方式下，汇聚交换机使用短距 MPO-TO-4LC 线缆和配线架 1 对接，接入层设备根据机柜实际位置选择不同长度的线缆与配线架 2 互联完成对接。

这样的布线方式较为简单地解决了线缆长度和备件所带来的问题，同时也增加了管理成本，并且配线架背靠背对接存在后期维修故障的难度。这样的配线架模式在后期如果其中一对接口出现问题，那么基本上是不可维修的，因为在配线架上直接进行光纤熔接操作非常容易误碰到其他线缆引起不必要的损失。针对这个问题，布线厂商在后期提供了预端接布线方案，大家可以自行百度查看技术细节，主要是以模块化思路处理了这部分对接，出现异常可以通过更换预端接模块进行简单处理。

Pod 内服务器容量设计依赖于厂商限定的条件，以 Cisco Nexus 为例，在主流千兆服务器接入环境下，根据汇聚交换机工作模式的不同，单 Pod 容量可分为 1152 台和 2304 台服务器两种规格。受限于此条件，我们面临 Pod 内部署什么样的业务，以及业务超过 Pod 容量后 Pod 间带宽如何计算配比等问题。

此类组网方式减少了网络可管理的设备台数，全网设备管理集中在几台核心设备上。这样带来的管理是一些较为轻量级的操作，如接入层接口配置变更等都需要登录核心汇聚设备进行。但是为后期的权限划分增加了难度。

◎ 成本问题

Pod Design 在成本问题上可以分为两块内容，分别是成本提升和成本优化。成本提升主要是

在这类组网方案中多了一层汇聚交换机的成本投入。成本优化则体现在业务流量模型清晰化上，可以减少 Pod 间收敛比、设备板卡及模块的投入，但这个需要视实际业务情况而定。

◎ 二层应用部署

二层应用部署其实面临着跟 Layer 3 互联组网一样的问题，只是在 Layer 3 二层环境中以接入交换机为单位，Pod Design 以汇聚交换机为单位。

（2）Pod Design 互联组网的优势

Pod Design 采用类似多虚一的虚拟化技术，剥离传统接入交换机的控制层面。这样的优势是以 Pod 为单位只管理两台汇聚交换机，劣势就是接入交换机只能工作在 Pod Design 工作模式下，因为它自身不具备控制层面，无法单独使用，如 Cisco Nexus 2000 系列。

安全的挑战

安全是任何一家互联网公司都不能忽视的问题，但安全需求在执行落地时都会给业务发展和基础架构带来一定的挑战，在此阶段安全对网络提出了以下需求。

1. 互联网端口白名单

随着 LVS 平台对运维工程师的管理权限下放，以及一些 LVS 无法支持的业务直接暴露在公网上，需要在互联网入口做防护，其规则制定和访问控制是保障业务安全的第一要素。

网络和安全人员一起制定了默认的白名单端口范围，以及特殊端口的申请及审批流程。在某案例中，这个功能在讨论初期存在两种方案，其中一种是在各 LVS 集群生效访问控制；另一种是在网络设备如防火墙、交换机生效访问控制。在网络设备层面生效的优势是可以在最初的入口完成访问控制，这也是访问规则就近生效的原则。另外，对于部分直接暴露在互联网上的业务，在网络设备层面统一入口生效也有助于对这类业务的保护。

互联网端口白名单可以选择在出口互联网接入层交换机层面生效，也可以通过上层增加硬件防火墙生效。从安全角度出发，使用硬件防火墙是最好的选择，因为硬件防火墙自身是可以实现如 Syn Flood、Ack Flood 等攻击的防御的，并且基于会话的硬件防火墙可以非常轻松地实现业务单向访问的需求。反观交换机传输 ACL 隔离，仅可以实现单纯的访问控制，在单向访问和防御 Syn Flood、Ack Flood 攻击方面则显得无可奈何。

但最终我们还是选择在交换机层面实施互联网端口白名单，其中一个主要原因是设备性能，因为交换机不记录会话，所以会话的多少是不会直接影响交换机的转发性能的。反观硬件防火墙设备，防火墙设备在选型时考虑的两个重要指标就是并发数和每秒新建连接数。当存在高并发业务时，防火墙设备对此类业务的安全过滤显得有些力不从心。另一个原因是成本，在出口并发数为 200 万，支持 200 万并发的场景下，防火墙产品和交换机的价格可是天壤之

别。

互联网端口白名单有两种实现模式，其中一种是黑名单即明确拒绝特定端口；另一种是白名单即明确允许特定端口。第二种模式明显限制得更为严格，并且具备完善的审批和记录机制，如果遇到安全问题，可以快速地进行事件追溯（也可以通过全流量分析完成）。但这也在一定程度上带来了人力成本的上升。可以根据实际情况酌情选择。

2. 内网隔离

内网隔离在金融和军政网络中存在比较多，坦白地讲，金融和军政内网隔离相对较容易实现，因为此类网络对服务器数量和未来流量的预期相对有据可依，所以对这类网络可以选择防火墙隔离或网闸等手段来完成内网隔离。

但互联网公司业务存在很多不确定性和高突发流量的情况，在安全和性能之间很难找到一个平衡点，所以在互联网公司内网隔离就变得难以实现。

需要实施内网隔离一般有以下场景。

（1）第三方评审

如 PCI-DSS，全称为 Payment Card Industry（PCI）Data Security Standard，第三方支付行业(支付卡行业 PCI-DSS)数据安全标准。

（2）非授信区域

部署第三方开源应用区域，如 Discuz、Phpwind 等开源程序。

（3）业务线间

不同业务线均存在不同类型的敏感数据，并相互不信任其对方的安全性，要求不同业务线间进行内网隔离。

针对不同的场景，应当采用不同的隔离方案。首先，安全合规应用属于第三方评审要求，此类内网隔离建议采用硬件防火墙设备，在这种安全至上的业务面前，性能反而不是第一考虑因素了；其次，针对后两种场景的这类安全需求属于公司内部需求，且对网络性能要求很高，这类需求建议采取交换机 ACL 方式进行安全隔离。

但采取交换机 ACL 方式实现内网隔离困难和问题依然很多，这主要体现在功能、容量、管理与变更三个方面。

① 功能

比如 ACL 实现单向访问困难。在互联网端口白名单中问题不大，因为互联网涉及的 IP 较少，类型也比较单一。但应用在内网隔离中这部分缺陷就显现出来，内网应用协议繁多，如 FTP、DNS 这两类应用是不能通过传统 ACL 实现单向访问的。所以一个折衷的做法是将此类无法实现单向访问的应用和必要的服务（如 YUM/NTP 等）进行梳理全部放行，同时将此类服务定义为基础设施白名单，该名单中的服务默认在所有生效的 ACL 区域放行，进入此名单的服务需要通过完善的安全审核机制和流程，保障其安全性。

② 容量

这方面问题主要是因为后期业务之间调用繁多,超过交换机自身设备所能承载 ACL 条目的规格。交换机 ACL 规格和防火墙 ACL 规格不在一个数量级上,防火墙 ACL 规格在几万甚至几十万级别,但交换机 ACL 规格往往只在千级别。

③ 管理与变更

设想一下,上千条访问规则分别应用在不同的设备上,管理与变更这么庞大的访问规则不是件容易的事情。

中期网络运维总结

在这个阶段我们完成了数据中心网络标准化的雏形规划和建设,通过与业务磨合,清晰地了解了业务对网络的需求。随着业务从不规律到逐步稳定上升,业务部门可以提供一定的可参考数据,这对后期建设标准化数据中心网络提供了良好的数据积累。业务部分对网络可提供的服务也有了一定的了解,如数据中心容量、线路资源、安全特性等。在这个阶段数据中心网络模型满足当前业务的需求。

总结

前两个阶段,是某真实案例中,从初期到当前现状的核心网络的进化史和血泪史,期间所使用的组网技术和方案不是业界先进的技术和解决方案,都是一些传统的技术和组网解决方案,所以在介绍中并没有过多的技术描述,涉及的技术大家可以自行搜索,有大量的技术文档可供参考。希望通过我们的介绍,让大家对处理不同业务需求和初期网络建设少走一些弯路。

近两年随着云计算的发力,公司在初期阶段和对网络自主性要求不高时,云主机的解决方案是一个非常不错的选择。但随着 Docker 和 OpenStack 这些虚拟化服务的大量应用,后期需要考虑的是如何在网络架构中更好地支持虚拟化业务。传统的网络架构提供的扩展性和稳定性已经远远不能满足云计算中心对网络的需求,除了扩展性和稳定性这些基本需求外,对网络灵活性和网络快速适配能力都有较高的要求。

业界应对主流云计算网络架构的技术有很多,如 SPB、TRILL、FabricPath、VXLAN、OTV、SDN 等,希望这些技术在我们正式应用后继续分享给大家。当然,这个阶段的网络还存在一些问题,如监控的精准度、网络管理自动化等。这些问题和业务无关,是网络自身的运维管理问题,我们希望后续通过平台化以及对上述新技术的研究解决这类问题。

五、传输网建设

在前面我们提到数据中心采用“双中心”+异地备份的模式，这种模式得以应用的前提首先是三点之间互联专线的稳定性和时效性。对于数据中心间带宽的需求，逐渐会大于数据中心外网带宽的需求，数据中心间传输网的容量和稳健显得尤为重要。传输网除了承担多数据中心间数据库同步、日志传输、容灾切换等需求外，还要保证海外数据中心和国内高质量的通信要求，如运维管理、代码发布、部分业务信息同步等。

传输网的建设也是一个逐步发展的过程，首先是从核心网互联需求开始的，然后经历了由核心到周边，由国内到海外的阶段。为实现核心网的互联，首先规划了两个 POP 点（汇聚中心），各数据中心都通过双路由与 POP 点联通，形成星型架构，通过建设波分系统来扩大容量，构建出高可用、高容量的传输体系。

CDN 业务需求和用户本地化交互业务需求，推动把外地和海外数据中心也纳入到传输体系中。在中国大陆以外，规划中国香港地区为整个海外到国内的中继，同样规划了两个 POP 点，这个覆盖全球的传输网络骨架就搭建起来了。

传输网雏形

数据中心初期建设时，由于业务规模较小，机柜、带宽等基础资源无法预估，随着业务的快速发展，造成单数据中心基础资源趋于饱和，业务无法进行扩容。另外考虑到单数据中心业务部署可靠性问题，多数据中心之间的数据热备是推动传输网建设的主要因素。开始是简单的专线互联，随着专线越来越多，形成了传输网初期雏形。

传输网建设初期专线带宽量比较小（小于 2Gbps），出于维护便利性、低成本因素考虑，专线资源主要依托当前租用机柜服务提供商提供整体解决方案。首先，要求至少提供两条异步路由传输通道保障可靠性；其次，要求提供二层 VLAN Tag 全透传模式来保障未来业务扩展性（后续增加互联需求时，自行添加 SVI 互联接口进行路由策略添加即可）。另外，与业务方沟通专线带宽需求，掌握业务系统部署架构，确定专线使用原则，在此阶段尽量减少跨数据中心业务调度，避免业务过度依赖专线。

总结：此阶段传输网处于被动建设，无长远规划，专线资源强依赖第三方服务商。

传输网初期建设

在进行第三个数据中心规划时出现了很多问题亟待解决，简要分析主要有以下几个方面。

（1）数据中心间专线容量扩展性。随着数据中心服务器数量的线性增长（以 Hadoop 业务增长为代表），跨数据中心日志传输量、DB 同步量对跨数据中心之间的专线容量需求上升到 10Gbps 级别，迫切需要对线路资源进行扩容。但由于初期多数据中心建设时出于安全风险、商务竞争因素考虑，数据中心基础设施机柜资源是由不同供应商提供的，专线出现质量问题时难以划分责任主体，从而需要协调两家乃至多家供应商共同进行故障处理；开通专线和扩容时，需要涉及的供应商应同时具备资源才能进行。在某真实案例中，就曾遭遇过由于其中一家供应商专线资源容量不足，而导致无法进行扩容的问题。

（2）特殊业务对双运营商专线需求。电商仓储、金融支付类系统需要使用专线与外部第三方公司进行对接，出于可靠性考虑，通常要求采用双运营商线路进行对接，而单运营商数据中心只允许本运营商线路接入，造成在基础设施上难以满足此类业务专线接入需求。

（3）机柜、带宽资源捆绑限制。在初期数据中心选型过程中，业务需要覆盖多运营商用户，互联网线路覆盖能力是考核供应商非常重要的指标之一，供应商本数据中心提供多运营商线路是首要条件，在这种情况下就造成带宽与机柜强捆绑，只能选择与第三方服务商进行资源合作才能满足多线路接入（一般情况下，单运营商数据中心只提供单线路资源，严格限制其他运营商带宽资源引入数据中心）。而这些第三方供应商提供的所谓多线路资源也是通过各种手段拼凑在一起的，总体来说，风险还是比较高的，在使用过程中可能会经常发生非本数据中心运营商线路故障问题。

为了解决以上存在的各种问题，自建传输网提升日程，至此进入传输网初期规划阶段。通过自建传输网，不仅可以从根本上解决上述问题，而且还会带来诸多额外的技术、成本优化价值。下面就介绍传输网基础架构规划。

传输网物理架构

在传输网设计规划时，在地域上将传输网划分为城域网节点、广域网节点、海外节点，然后根据每种节点类型进行功能划分，这样就能为基于需求使用特定接入方式提供最优的技术和成本方案。

传输节点物理位置的选择直接影响到传输网的可靠性、维护便捷性、未来扩展性。

首先，至少选择两个物理数据中心来保障可靠性，数据中心间距离 $\geq 60\text{km}$ ，减少光缆接入时同路径路由、区域电力以及灾害等不可抗力因素对传输网的整体影响。

其次，传输节点最好选择自有产权办公楼数据中心，一方面，可以根据自己的需求，对数据

中心的物理安全及电力等配套基础设施提出高标准建设要求，如市电故障时，UPS 需要能提供至少 6~12 小时后备电力，进出数据中心需要严格审核等，进一步保障物理安全；另一方面，可以提高进出数据中心维护的便利性、可控性。

最后，传输节点物理线路接入能力，对于传输网建设也起到举足轻重的作用，要求入楼管井及数据中心弱电管路等资源可自行控制，对入楼管井、熔接包、光缆资源拥有产权等措施，可以提高后续光缆接入的可扩展性；在入楼光纤配线架首次施工前，充分评估未来 2~3 年业务增长量，减少重复施工；入楼管井双入口，要求供应商线路通过不同管井入楼；多数据中心互联对接时，至少有两家供应商，通过两条物理上不同路由的链路，分别接入到两个传输节点，保障可靠性。传输网物理架构示意图如图 10-1 所示。

在某案例中，通过租用第三方供应商专线的方式联通，遇到了可扩展性及容量问题；另外，由于运营商政策限制，多数据中心间无法进行直接联通。因此必须建设自有传输数据中心，来担当核心数据中心间的汇聚节点。

我们规划了传输网城域网节点（如图 10-1 中的城域网节点①），将所有运营商数据中心，通过不同路由的两条裸光纤，分别接入至城域网节点 A、B。通过使用裸纤线路互联，至少可提供主备各 10Gbps 容量，未来可通过采购密集波分复用设备（DWDM）进行扩容；为解决特殊业务对双运营商小带宽（容量为 2~100Mbps）专线接入需求，两个城域网节点分别引入了电信、联通、移动专线。

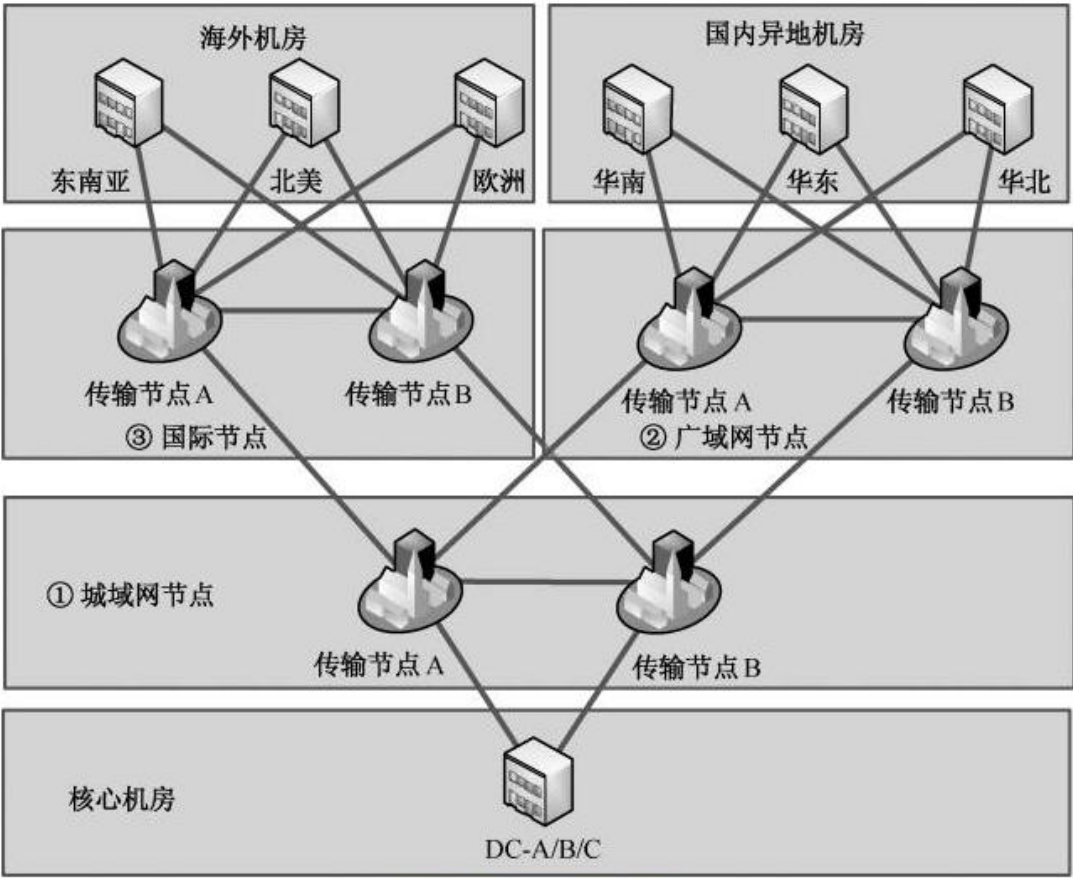


图10-1 传输网物理架构示意图

随着多数据中心业务系统架构逐步建设完成，关键业务系统的异地灾备开始提上日程。经过调研业务需求，灾备数据中心与核心数据中心间需要提供大容量（1~10Gbps）专线带宽，并提供服务质量保障。出于安全风险考虑，一般灾备数据中心距离核心数据中心近千公里，于是我们规划了广域网节点（如图 10-1 中的广域网节点②），用于国内异地数据中心接入。经过调研长途大带宽专线，国内三大运营商成本非常高，相比之下，缆信、中信、铁通线路的资源，可以提供相对较高的性价比，目前长途专线主要租用这三家资源。

随着海外业务的开展，初期业务系统的监控报警、部署、维护等都需要与国内核心数据中心联通，某些业务安全级别要求较高也需要经专线传输。在对接过程中尝试过多种方案，比如搭建 IPSec VPN 隧道打通，经过验证这种方案的稳定性基本无法保障（主要原因是到海外互联网带宽资源紧张，电信运营商不提供服务质量保障，经常发生不规律、不明原因的 TCP Reset 造成业务中断），于是我们在中国香港地区建设了国际专线汇聚接入点，中国大陆与国际数据中心以及国际数据中心之间的数据交互通过香港交换，不需要再返回至大陆汇聚节点。

经过综合考虑国际线路成本及业务部署稳定性问题，以及与业务线商讨，确认国际线路使用原则如下：

- （1）国际专线不作为业务系统使用，仅作为运维使用。
- （2）对于特殊的高安全级别业务，需求走专线的，需要进行特殊报批申请。
- （3）非敏感类业务的需求，使用中国香港地区的国际互联网带宽（香港地区的国际互联网访问海外质量较好，规避了 GFW 影响），尽量降低业务对专线的强依赖。
- （4）由于国际专线带宽资源有限，在充分复用基础上，部署 QoS 策略保障关键业务服务质量。

在传输网上规划了中国香港地区国际节点（如图 10-1 中的国际节点③，主要基于其国际线路资源优势，以及与中国大陆联通性考虑），通过香港地区国际节点将海外资源、国内资源进行联通，即可实现成本优化（国内直接联通海外专线商务成本高，高成本主要是由于国际运营商与中国大陆运营商之间结算费用较高）。

传输网逻辑架构

在多数据中心架构设计上，每个数据中心都是内外网物理架构的，传输网的逻辑架构设计也主要基于以上因素的考虑，将跨数据中心间的内外网流量分离，并采用分区域模块化设计，在每个数据中心设计独立的传输网区域，多数据中心间的内外网流量优先通过传输网交互。传输网逻辑架构示意图如图 10-2 所示。

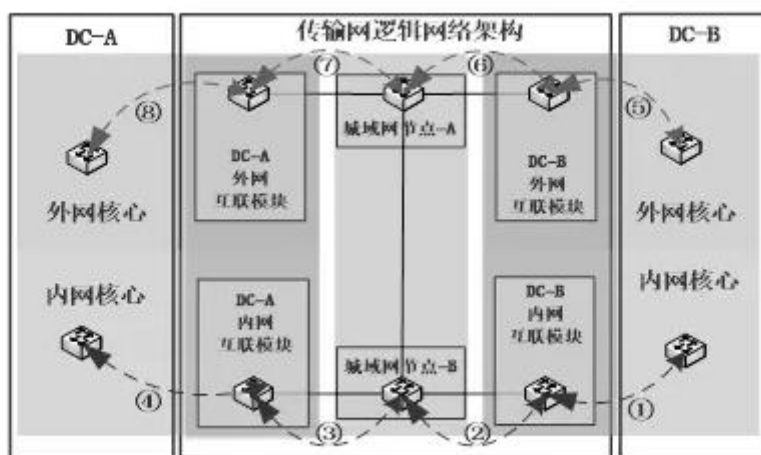


图10-2 传输网逻辑架构示意图

在传统的网络结构下，虽然是自有业务间的访问需求，但不同运营商的外网间访问均需要跨 ISP，带来访问延迟较大、丢包等问题，并且会有额外的带宽成本支出。通过传输网的外网互联模块，打通了多数据中心间外网，不但可以解决延时、丢包等技术问题，而且还可以将多数据中心间自有业务的互访流量控制在传输网内部，提供了专线级别的访问质量，同时减少了出公网带宽成本。

流量模型如图 10-2 所示，⑤→⑧代表跨数据中心间外网流量交换。

多数据中心内网之间的交互主要依靠传输网区域的内网互联模块，通过城域网节点裸纤对接至少可提供 10Gbps 容量，后续业务增长，只需要部署一套波分设备扩容即可。

流量模型如图 10-2 所示，①→④代表跨数据中心间内网流量交换。

传输网线路容量

传输网容量主要受传输节点的物理空间、电力、波分设备档次影响，一旦建设完成，再进行割接改造成本较高，对业务影响范围比较广，因此需要考虑 1~3 年业务跨数据中心间传输容量需求。在传输网容量需求的不同阶段，使用不同的技术方案，以匹配业务需求。

根据传输网建设经验来看，多数据中心间的传输容量小于 1Gbps，主要基于运维简易性考虑，通常租用两家第三方服务商双线路传输通道接入，即可实现高可用，当传输容量需要扩容至 10Gbps 时，则可以考虑租用裸纤方案，并配套使用长距单模万兆模块对接，同时为未来扩容到大于 10Gbps 容量做好光纤等基础资源储备。未来传输容量发展到大于 10Gbps 时，基于裸纤搭建波分网络扩容即可。

波分的选择：首先，根据传输网容量大小，确定采用密波分或粗波分设备；其次，传输网整体架构采用汇聚型结构，在网络数据层面进行冗余保护，减少波分物理环[1w1] 保护建设成本；最后，在波分设备部署架构上优先采用链型方式组网。这样做的好处是，一方面，物理层结构简单清晰，扩容方便；另一方面，利于减少波分设备厂商绑定（波分设备成对配套使

用，不能使用不同厂商的设备对接)，在设备选型上可以使多家波分设备供应商参与竞标，降低商务成本。关于传输设备厂商的选择，考虑到品牌知名度、可靠性、售后服务等因素，建议优先考虑华为、中兴等本地化厂商。

传输网服务质量

虽然专线容量在不断扩充，但仍然会有数据中心间专线、国际专线带宽拥塞丢包的情况发生，造成业务服务质量严重下降，甚至不可用。因此，基于各业务在专线使用中对带宽要求各不相同的考虑，需要对突发流量进行控制，重点保障高优先级业务的服务质量，尽量避免由于专线资源满载造成的业务降级或不可用的问题发生。解决此类问题的关键在于，如何精准识别不同业务类型需求，根据业务需求提供不同的服务质量/带宽保障。

结合上面提出的解决方案，对专线的服务质量给出了 QoS 保障流程。

◎ 定义金、银、铜三类服务级别，通过 DSCP 区别不同的业务服务级别，金、银、铜按照总带宽容量 6:3:1 比例（具体比例按实际情况进行划分）分配，在此之上再将不同业务划分入不同的服务级别。

◎ 业务服务器出入网卡流量依据服务级别执行 DSCP 打标签。

◎ 传输网区域模块根据 DSCP 值提供对应的服务质量保障策略。

传输网调度策略

上面介绍的主要是传输网的架构和业务的互访需求，接下来我们看几个具体的应用场景和实现方式。

场景一：多数据中心内部不同运营商间流量交互调度

（见图 10-3）。

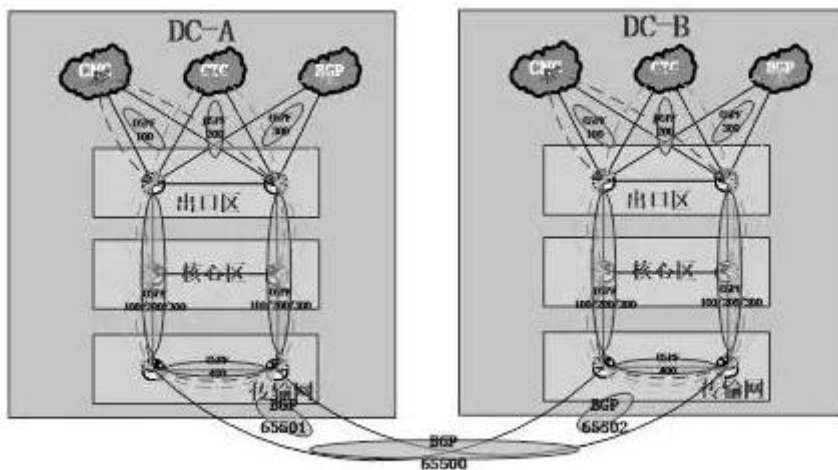


图10-3 传输网流量调度路由示意图

多数据中心间外网流量交互访问，通过传输网区域的外网互联模块，根据调度策略进行灵活控制。在正常情况下，多运营商需要多条默认路由保障出入向流量保持一致，使用一张 Public 路由表会导致默认路由冲突，这是无法满足需求的。

为了解决默认路由问题，需要在同一套物理设备上使用 VRF 技术进行路由表逻辑隔离。使用 VRF 多实例（CNC/CTC/BGP）、多进程 OSPF（进程号 100/200/300）逻辑分离多个运营商。

为了实现单数据中心内的多个 ISP 之间互联互通，传输网区域使用 VRFALL_ISP 标识并使用 OSPF（进程号 400）将三类 ISP VRF 路由打通，并过滤掉默认路由（避免由于默认路由引起的多个运营商之间流量串扰）。

为了实现多数据中心间外网之间互联互通，多数据中心传输网外网之间采用 BGP 协议对接，在每个数据中心传输网边缘将外网路由重分布，进而达到多数据中心间进行路由传递的目的。

场景二：跨数据中心带宽借用。

依靠传输网资源，可以将 CDN 数据中心带宽，临时调用至其他数据中心使用，通过复用 CDN 大带宽资源，满足业务突发活动的带宽需求。

技术实现原理比较简单，主要是使用 VRF 技术，将核心数据中心的运营商带宽资源，在传输网外网互联模块上进行延续扩展。在城域网节点，将引入到核心数据中心的运营商，设置相同的 VRF，在使用外部调度资源的数据中心，通过动态路由协议，进行 IP 地址路由通告，即可实现资源的引入（见图 10-4）。

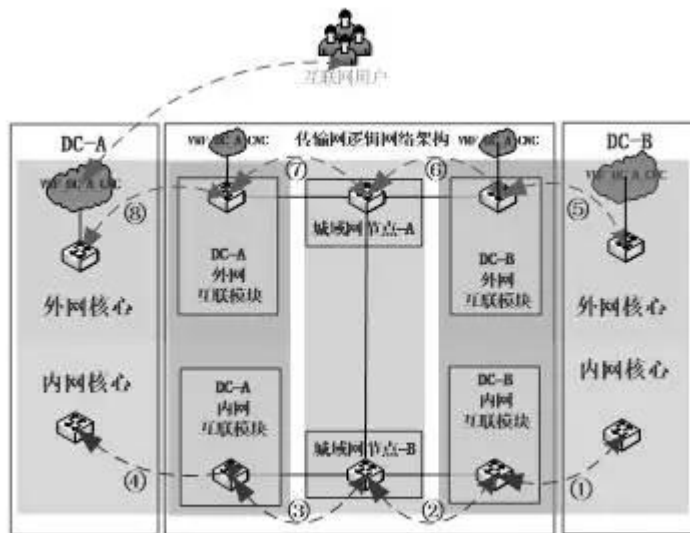


图10-4 传输网带宽资源引入示意图

总结：经过传输网的初期建设，传输网的整体架构已经搭建完成。在未来 1~2 年内，基于现有架构进行波分、网络设备扩容即可实现传输容量的升级扩容。

六、域名和接入

域名是所有互联网公司服务的入口，也是重要的公司品牌。域名的使用和设计是否合理，不仅影响用户能否快速访问到业务，而且也直接影响内部服务的部署架构、管理复杂度、自动化运维实现等方面。众所周知，域名的解析结果是一个或多个 IP 地址，利用域名解析的结果，就可以实现：

- ◎ 基本的负载均衡；
- ◎ 引导流量接入到期望的目的地址。

若只使用域名解析来接入服务，则存在以下问题：

- ◎ 受 Cache 等因素影响，利用域名解析多 IP 地址进行负载均衡效果有折扣；
- ◎ 域名指向的 IP 地址必须要直接接入互联网（公网）。

Cache 还不足以造成严重的影响，但要求所有业务前端必须接入公网，在小规模情况下问题还不突显，但当业务发展到一定规模时，所引起的副作用就足以让运维人员疲于应付。主要问题有：

- ◎ 服务器接入公网对网络规划的要求；
- ◎ 服务器接入公网后的安全性考虑；
- ◎ 服务器接入公网后的防攻击考虑。

因而，基于 VS/NAT 的四层负载均衡器和基于 URL 分流之类的七层负载均衡器应运而生，其核心思想就是只利用少量的、直接暴露在公网的负载均衡层接入用户请求，再通过算法和策

略，平均地转发到内网的业务前端进行处理。同样地，也将内网前端的处理结果，原路返回发送给对应用户。负载均衡层在用户和业务之间起到代理和转发的作用。通过图 12-1 可以看出有/无负载均衡层在结构上的区别。

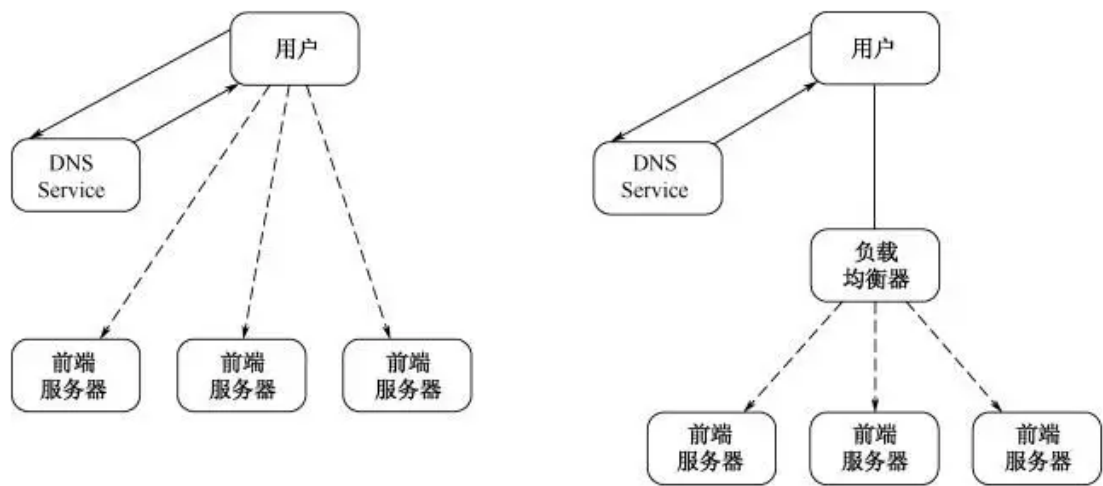


图12-1 有/无负载均衡层在结构上的区别

负载均衡技术的引入有效地控制了对外暴露服务器的规模，使得安全策略和防攻击都能够被集中和统一地解决。并且与前端服务器规模不成正比关系，无论业务规模如何发展，对外暴露服务器的规模都是有限的。同时，负载均衡技术使前端服务的扩展更加平滑、透明和可控（不受用户 DNS Cache 影响）。

下面分别介绍我们在域名和负载均衡实践应用中的运维设计和自动化管理方面的经验与考量。

负载均衡及自动化

负载均衡器在大规模业务集中发展为必不可少的设施，在业务乃至整个基础环境中都起着至关重要的作用。包括：

1. 简化网络结构和网络维护成本

以最原始的互联网服务为例，对互联网用户提供服务就必须将服务器接入到公网中，如图 12-2 所示。在小规模业务中，为满足这样的需求进行网络规划和实施并非难事，也不存在太大的成本，甚至内网和外网都可以共享同一套物理设备，所以也是比较常见的形态。

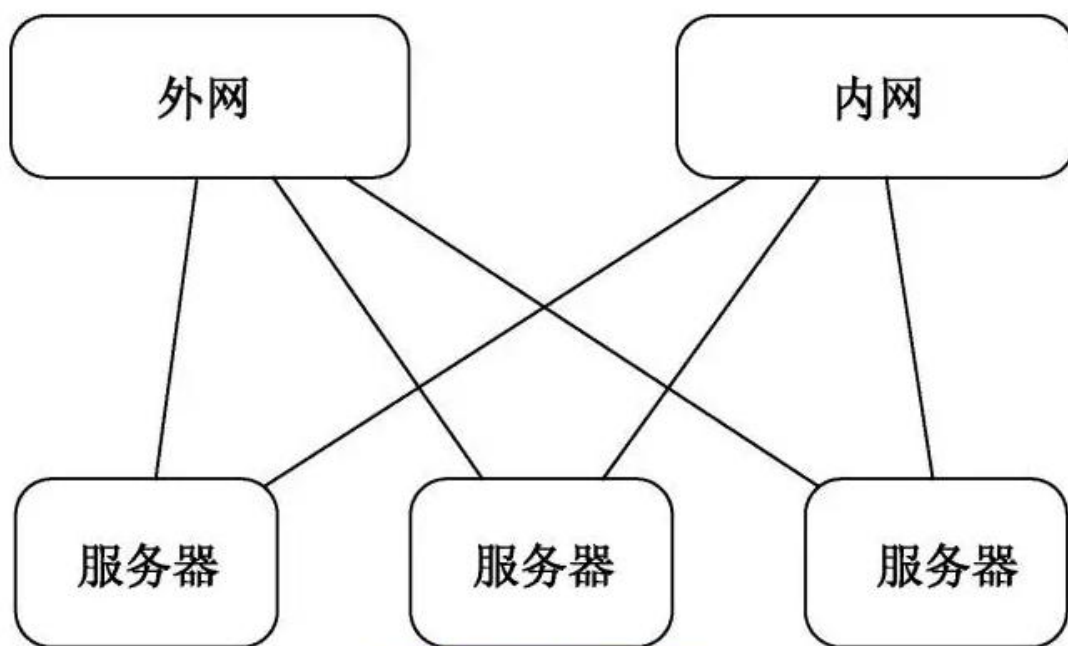


图12-2 增加负载均衡器前的网络结构

但随着业务的发展、规模的扩大，会逐渐地发现：

◎ 除了前端服务器（业务的入口），大部分服务器并没有访问外网或被外网访问的需求，因此配置外网实际上是一种浪费，甚至还额外引入了安全风险；

◎ 如果内、外网共享物理设备，则有相互干扰的风险。例如，外网被攻击导致设备工作异常，内网也会同时受到影响。但如果拆分内、外网，则又会出现两套网络，浪费物理资源。增加负载均衡器后，网络结构变得更清晰和简洁，如图 12-3 所示。

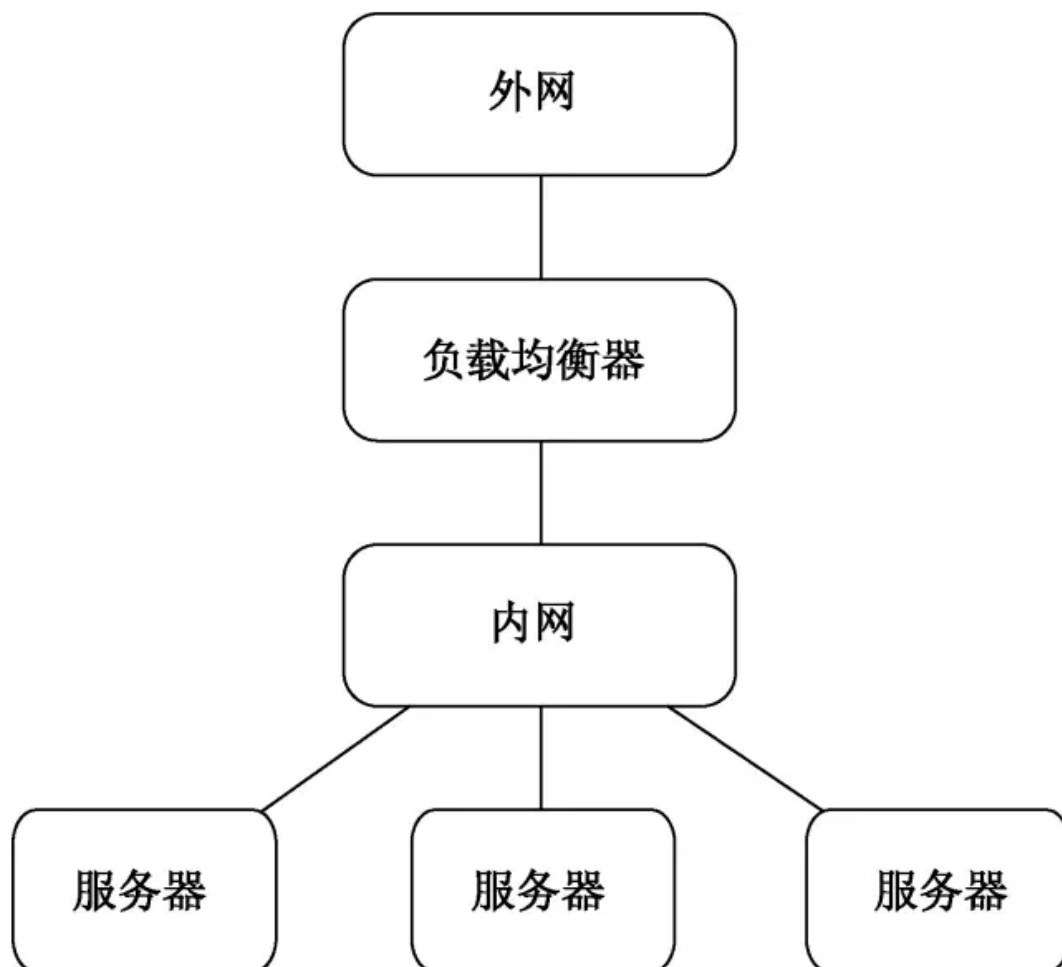


图12-3 增加负载均衡器后的网络结构

在增加了负载均衡器的结构中内、外网是分离的，外网流量通过负载均衡器转发进入内网，即使外网发生故障异常也不会不影响内部服务的运转。负载均衡器的规模与流量和转发性能成正比，与内网服务的规模无关。而通常业务功能的扩展和集群规模的扩大都主要发生在内网。

2. 减轻在业务升级维护过程中对用户的直接影响

从图 12-2 中可以看到，用户请求直接与服务器交互，服务器正常的维护、升级、故障等因素都会导致短暂的服务中断，虽然用户端可以设置重试策略，在请求失败后重试到其余服务器，但是对于新用户请求来说仍然会有一部分会访问到异常服务器上，即服务中断期间，有 n/m 的请求会出现失败和重试（ n 为服务中断服务器数量， m 为提供该服务的服务器总量）。

而从图 12-3 中可以看到，用户请求与负载均衡器进行交互，再由它进行转发，因此如何进行转发是可控和可配置的。当内网中某服务器异常或维护时，在负载均衡层屏蔽该服务器即可使用外网流量，不再转发到该服务器，从而消除了用户请求存在 n/m 失败重试的可能性。

3. 更集中的安全策略控制和防攻击

增加负载均衡器后，与互联网（公网）互联的服务器由分散的多个入口收敛为集中的一个入口，从而使安全防护策略也跟随集中和统一，更方便地进行维护和变更，消除和避免安全防护策略的缺失、遗漏而造成的隐患和损失。在防攻击方面，图 13-2 所示的结构是以寡敌众，容易被逐个击破；而图 13-3 所示的结构通过负载均衡器将内网的服务集群虚拟为外部可见的一台服务器，是以众敌众，强者胜之。

4. 支撑业务前端更灵活和实时的容量扩展

如前面所述，负载均衡器将内网的服务集群进行了虚拟，在外部用户看来就是一台服务器，内、外网之间的流量由其决定如何转发，因此内网服务器的扩展和调整是可控制的，变更的生效是更为实时的，集群的规模几乎是不受限制的（例如，使用域名->IP 地址会受到 DNS 相同域名的最大条目数限制集群规模），并且变更的过程对于用户透明、无感知。

负载均衡的解决方案有多种，如基于四层或七层，基于硬件或软件；在大规模业务场景中历练过的不乏有 F5、LVS、Nginx、HAProxy 等。作为一个重要的基础设施，负载均衡方案的性能、可扩展性、均衡策略、对业务是否透明等指标直接关系到业务的稳定性、处理能力和接入维护成本。

下面介绍的是以经典代表 LVS+Keepalived 为基础的负载均衡器接入设计和自动化管理方案。

选择 LVS+Keepalived 的考虑

当前 LVS 在业内已得到广泛的应用和认可，属于四层负载均衡实现，最初具有 NAT、DR、TUNNEL 三种工作模式，在随后业内的工程实践中又增加了 FULLNAT 工作模式，并由 taobao 开源，使之更适用于大规模业务场景。四层负载均衡如同底层交换设备一样具有天生的业务透明性，接驳到业务流中时对应用层无感知，无须进行额外的设置和调整。

而 FULLNAT 则是专为大规模工程应用而设计的，消除了原有 DR/NAT 模式在应用中的诸多限制，比如要求前端服务器在同一广播域或同一网段等。FULLNAT 模式使网络结构的设计更为清晰，同时跨网段的转发使服务集群的扩展更加灵活、便捷。LVS 可以运行在普通服务器上，有不俗的性能表现，并且安装和配置不复杂，名副其实的低成本和高收益。针对 LVS 的部署过程已有丰富的文档和手册详细介绍，在此不再赘述。

Keepalived 则是基于底层负载均衡框架（如 LVS）进一步扩展和增强了负载均衡的高可用性，使之具备构建集群的能力，提供了更适合于维护的配置格式、健康检查能力等。概括来说，LVS 实现了内部的业务集群抽象和虚拟，而 Keepalived 则完成了 LVS 集群的构建，以及 LVS 映射关系管理。

负载均衡器接入结构

负载均衡器接入结构如图 12-4 所示。

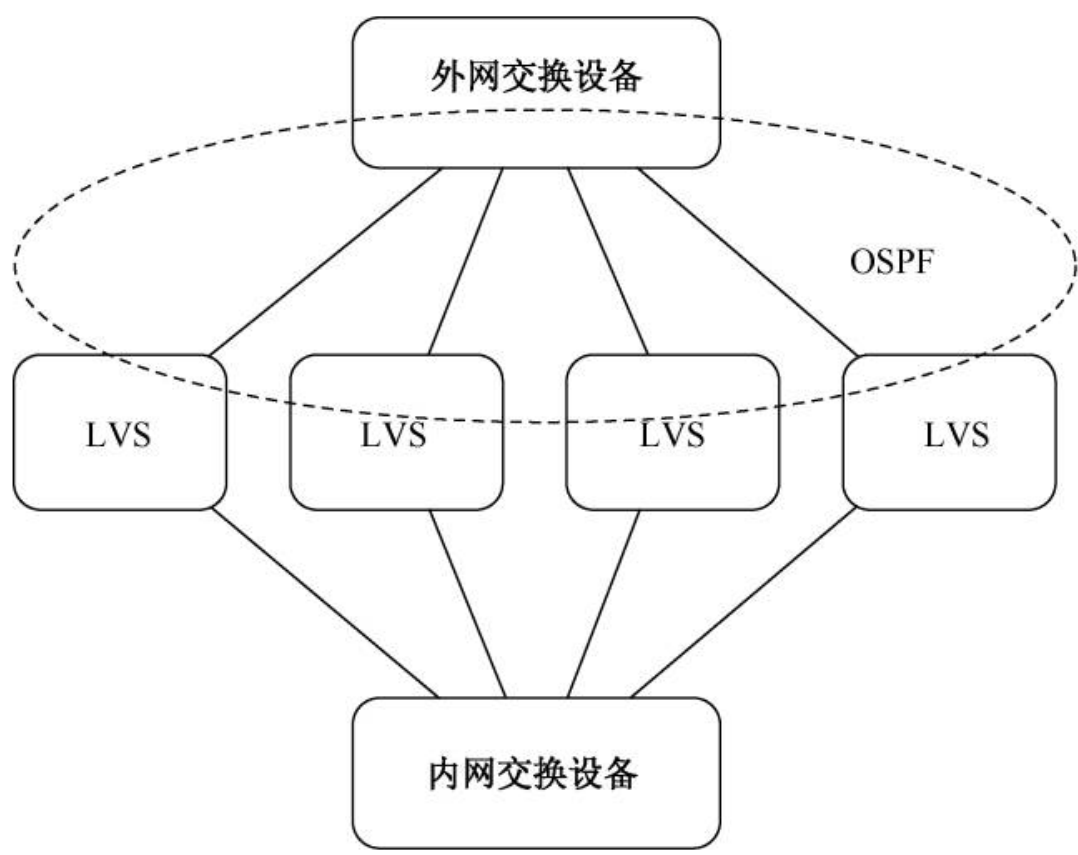


图12-4 负载均衡器接入结构

在实际应用中基于 LVS+Keepalived 的负载均衡器采用了如图 13-4 所描述的结构，LVS 的外网侧与交换设备通过 OSPF 进行接驳，LVS 的内网侧与内网交换设备直连。这种结构的特点和优势如下。

1. 容量可扩展的 LVS 集群

在前面已经提到，Keepalived 使 LVS 具备了构建集群的能力，但是 Keepalived 使用 VRRP 进行容错备份，而 VRRP 协议造成了两方面的局限：

- ◎ 在一个集群中，同一时刻只有 Master 能够工作，若干 Slave 只能作为冷备。
- ◎ VRRP 只能工作在二层网络，因此所有 LVS 节点必须部署在同一广播域。

通过 VRRP 协议构建的 LVS 集群，起到了冗余和备份的作用，但在集群处理性能和容量上仍然是单机性能为上限，并不能通过扩展节点来扩充容量。另外，同属于一个广播域的要求也必须要在网络架构上进行设计和支持，在某种程度上也会增加网络维护的成本和复杂度，其中不能水平扩展在大规模服务的应用场景中是明显的软肋。

所以，必须要使 LVS 节点具备热备的能力以及水平扩展的能力。通过在 LVS 与交换设备间建立 OSPF 邻居关系便是可行的方案之一，OSPF 邻居关系建立后，交换设备可以动态发现和感知 LVS 节点变化，并从 LVS 节点学习路由。若有多个 LVS 节点同时宣告了相同路由，交换设备则会通过 ECMP（Equal Cost Multipath Routing，等价路由）机制进行分流，而这恰好就是我们所需要的热备机制。在热备的基础上，增加或减少 LVS 节点数便能够重新分流，达到容量伸缩的目的。

2. 独立的 LVS 节点

不再依赖 VRRP 作为构建集群的基础后，LVS 节点之间便消除了相互通信和选举的需求，不会再发生类似 VRRP 模式下出现多个 Master 或无 Master 的情况。各个 LVS 节点更具独立性，单节点故障或进行维护不影响其他节点的正常运行。

在 OSPF+LVS 构建的集群化负载均衡结构中，虽然解决了大规模应用中的诸多问题，但也并非十全十美。

首先，单节点故障会造成交换设备路由的变化，造成 ECMP 机制重新计算。在交换设备中 ECMP 通常只是简单 hash 计算，没有一致性 hash 的特性，因此数据流将会完全打乱（简单 hash 计算保证了转发的性能和均衡性，而一致性 hash 除了增加复杂度外，也不能保证绝对均衡，所以当前交换机基本上都不支持一致性 hash 的特性）。若没有对 LVS 节点的 Session 进行同步，打乱后的数据流无法再进行转发，从而造成连接中断。例如，ECMP 重新计算前业务流是经过 LVS NODE1 分流和转发的；而 ECMP 重新计算后，业务流则可能会被分配到 LVS NODE2。由于 LVS NODE2 上并没有之前会话的任何信息，因而无法正常继续之前的转发和交互。

其次，FULLNAT 模块 in/out 流量都会经由 LVS 节点穿行，对 LVS 节点转发性能造成额外的开销。

最后，也是由于各节点的独立性，在配置有差异的情况下，各节点仍然能够正常运行，使得配置上错误或不同等小隐患不容易被及时发现。

LVS 的集群化增强了业务架构体系的稳定性和高可用性，但规模化的集群管理是衍生而来的运维烦恼。如何保证集群的同步一致和可靠运行，便是随之而来的课题。

自动化管理系统

Keepalived 在 LVS 的基础上，提供了更为友好的配置形式，便于人工维护。但在大规模应用中，由于配置的条目多，造成人工维护的变更成本高、误操作概率高等衍生问题，从而使得规模越大配置越难维护。而同时，LVS 又是内、外网流量接驳的咽喉，一旦出现误操作，就会损失全部用户流量。因此，LVS 配置的有效性、变更的准确性至关重要。抛弃人工管理实现自动化，既是顺势而为，又是必然的趋势。

系统结构如图 12-5 所示。

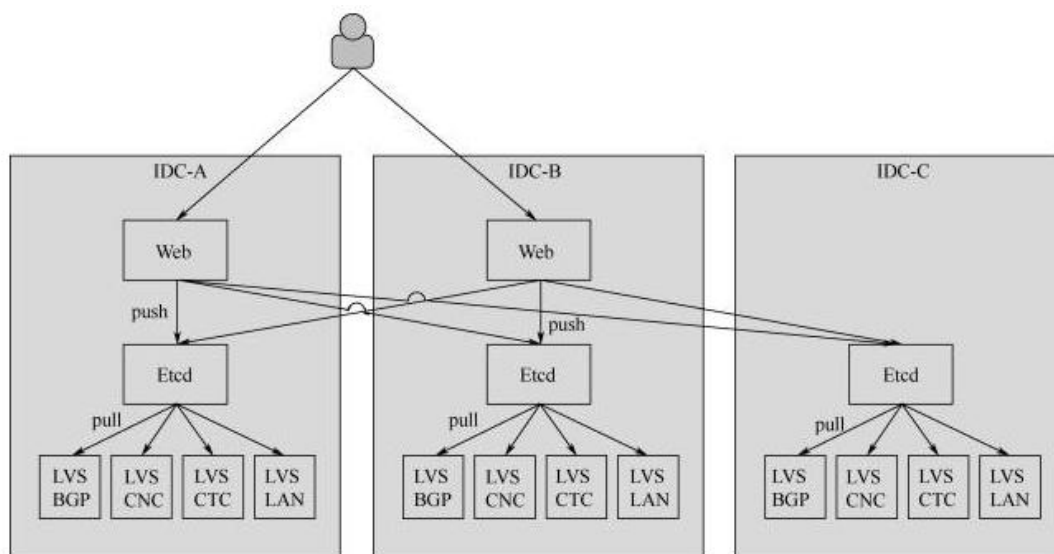


图12-5 系统结构

首先，系统的核心是 Etcd，一款用于配置管理和服务发现的开源的分布式 KV 存储。Etcd 数据是以树状结构存储的，而 Keepalived 配置同样也是树状的层次关系，DC->CLUSTER->VS->RS->RS HealthCheck，因此 Keepalived 经过层次化转换后，可以很容易地保存到 Etcd 中，并由 Etcd 自身存储结构来始终保证这样的层次性。同样在读取方面，指定一个节点（如 CLUSTER），拉取以该节点为 root 的子树，经过简单的合并，就可以轻松地将整个节点配置转换为 Keepalived 的格式。这也是我们选取 Etcd 作为 Keepalived 配置存储的考量。

其次，利用 Etcd 已有的 HTTP API 接口，封装和完善数据的权限及有效性检查，从而通过 UI 或工具就能自动化地进行 Etcd 数据操作，保证配置变更的准确性和可靠性，减少人为误操作。

再次，Etcd 作为整个系统的核心，同样也需要保证高可用性和高可靠性，而其自身的分布式设计便能满足高可用性需求。

最后，在 Keepalived 端，设计采用拉取的策略从 Etcd 查询得到集群配置并转换为 Keepalived 配置文件生效。pull 策略无疑牺牲了配置生效的实时性，但在稳定性和实时性之间，整个负载均衡器的稳定性是具有更高优先级的。

为什么 pull 策略会更稳定？主要体现在如下几个方面。

（1）pull 动作的触发并非为用户行为，因而是可控的、周期性的，即使有大量的集中的配置变更，也不会造成 KPD 的频繁加载。

（2）pull 形式更便捷地保证了 LVS 集群配置的一致性，例如某节点在故障或维护后，重新加入集群时，配置可能就已经发生过变化或更新。pull 方式在加入集群时强制拉取一次，便可以完成线上配置的同步。而若换作 push 方式，事情就会变得复杂，在节点故障期间，要记录状态不能往故障机器发布，节点恢复后，再重置状态触发配置的推送和发布。如果状态

处理得不正确，就会导致集群配置的不一致，造成隐患。

（3）避免功能模块间的耦合，从图 12-6 中可以看出，UI、Etcd 和 LVS 是相对独立的业务模块，Etcd 通过标准接口接受更新、查询请求，各个模块的升级不影响和依赖其他。

在实现 LVS 的自动化管理之后，才能进一步整合其他系统，实现体系的自动化。比如通过与部署系统的接驳，实现应用部署完成后自动流量引入；通过与域名管理系统的接驳，实现域名申请后自动进行 LVS 虚拟 IP 地址申请等。

总结

在互联网行业，高可用性和高性能业务架构是永远不能懈怠的话题，通过负载均衡技术进行流量的接入，不仅提高了业务的可靠性和冗余性，还使资源利用率和请求响应时间得到提升。如何利用好负载均衡这件利器，是持续的实践和探索。

七、CDN

CDN 是 Content Delivery Network 的缩写，直译为“内容分发网络”。工作思路是把用户需要访问的内容缓存在边缘节点，就近访问，进而绕开经互联网骨干传输数据的不稳定性和速度瓶颈，起到提升最终用户体验的效果。通过在网络各处放置节点服务器所构成的基于现有互联网基础的一层智能虚拟网络，CDN 系统能够实时地根据网络流量和各节点的连接、负载状况，以及到用户的距离和响应时间等综合信息，将用户的请求重新导向离用户最近的服务节点上。其目的是使用户就近获取所需的内容，解决 Internet 拥挤的状况，提高用户访问网站的响应速度。

CDN 分为静态加速和动态加速两部分，分别介绍如下。

静态加速

通过缓存静态资源到 CDN 节点，使用户可以就近访问 CDN 节点来获取资源，避免用户直接从源站获取数据，提升用户访问速度和质量。

如图 11-1 所示，相比用户直接访问源站，访问 CDN 节点可以带来以下收益：CDN 节点更靠近用户，可以提供更小的网络延迟及更快的访问速度；CDN 节点数量远大于源站，可以支撑更大的请求数量；CDN 节点多地域部署，可以提供更好的容灾能力。

静态加速可以分为小文件类业务、下载类业务。两者之间的区别在于文件大小以及访问方式，以及由此带来的业务不同侧重点。

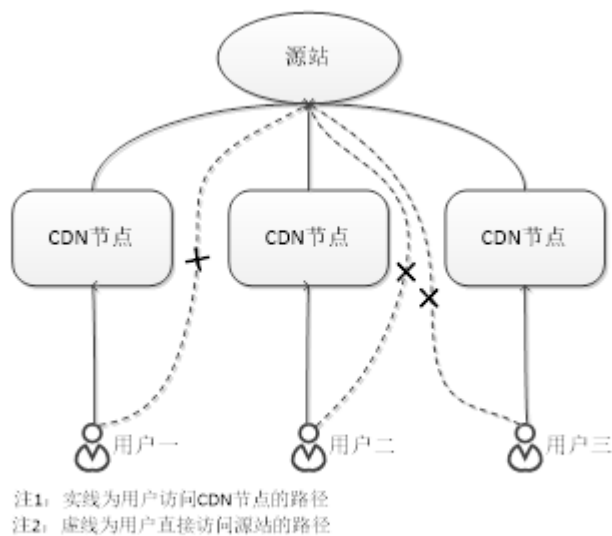


图11-1 用户访问CDN节点和源站的路径

小文件类业务：主要是通过浏览器访问的网页、图片、JS、CSS 等内容尺寸比较小（无严格限制，一般为数 KB 至几百 KB）的文件。特点是单次请求文件较小，请求总数较大，主要消耗 CPU 和带宽资源。一般都是直接在浏览器中展示，对访问成功率要求较高，要求 RTT 延时要尽量低，即节点要部署得离用户近。

下载类业务：相对于静态资源类业务，下载类业务的单文件尺寸要大很多，需要更多的磁盘空间和带宽资源，对 CPU 的消耗不如小文件类业务大。并且下载类业务的下载时间会比较长，相对于静态资源类业务更侧重于吞吐量，而不是 RTT 延时。

动态加速

静态加速业务是通过将文件缓存至 CDN 节点来提高用户的访问速度的，而动态加速是通过尽量减少用户到源站之间请求的网络耗时来提高用户体验的。可用的技术有 TCP 单边加速、多路 TCP 合并回源、回源压缩。

TCP 单边加速：主要原理就是在用户通过公网直接访问源站比较慢时，通过各种方式让用户请求绕过拥堵的网络环节。例如，在用户访问质量良好的区域建立 CDN 节点，用户访问先经过 CDN 节点，通过 CDN 节点中转至源站；或者在某些关键 CDN 节点使用专线和源站联通，以确保 CDN 节点到源站的回源质量（见图 11-2）。

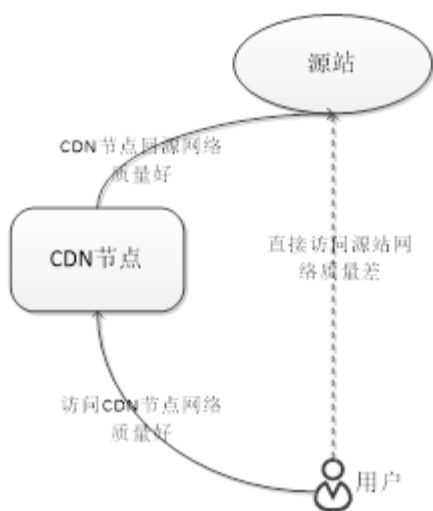


图11-2 TCP单边加速示意图

多路 TCP 合并回源：在 CDN 节点将多个用户的数据合并在一个 TCP 连接上回源至源站，以减少回源时新建 TCP 连接带来的额外时间消耗（见图 11-3）。

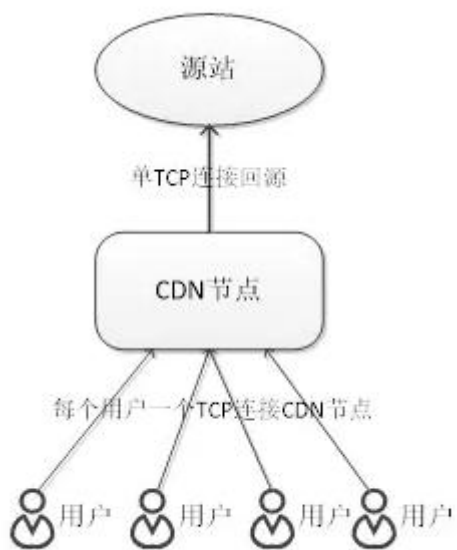


图11-3 多路TCP合并回源示意图

回源压缩：将回源数据流进行压缩，减少在网络上传输的回源数据量。

自建 CDN

初期公司更多的是使用第三方公司提供的 CDN 服务，随着业务量和业务需求的增加，第三方 CDN 在质量和运维效率上逐渐不能满足业务需求。为了提升 CDN 的质量，提高运维效率，

自建 CDN 是一个不错的选择。在某案例中，使用自建 CDN 后，不仅业务质量有了明显提升，成本也有了大幅度降低，并且从侧面帮助公司在使用第三方 CDN 服务时提高了议价能力。

该案例中，自建 CDN 支持静态加速和动态加速业务，带宽容量达数百 Gbps，日常支撑了全公司 90%以上的 CDN 流量，多次支撑了业务的大型发布推广活动（活动带宽相比日常带宽增长数倍）。自建 CDN 在设计之初就针对公司业务特性，以及从节点选择、软硬件配置到参数优化都进行了有针对性的调整。

在该案例中，针对公司业务的特点，我们首先开始了大文件静态加速的工作，随后逐步进行小文件静态加速和动态加速的工作。同时，CDN 管理和调度系统也随着 CDN 的发展不断完善。CDN 系统整体框图如图 11-4 所示。



图11-4 CDN系统整体框图

- ◎ **CDN 业务系统：**对用户直接提供 CDN 加速的缓存服务器和代理服务器。
- ◎ **私有调度系统：**公司的 HTTP 调度服务器和 302 调度服务器。
- ◎ **GSLB 系统：**支持智能解析的 DNS 服务器。
- ◎ **流量收集系统：**采集节点交换机、服务器和各业务的流量。
- ◎ **质量收集系统：**对质量评测数据的采集、存储。
- ◎ **日志统计系统：**统计 CDN 访问日志、调度系统日志。
- ◎ **质量分析系统：**通过质量评测数据和日志统计数据，汇总统计出质量排序结果。
- ◎ **CMDB 系统：**管理 CDN 节点的资产信息及状态跟踪。
- ◎ **调度决策系统：**根据质量排序结果、流量数据以及 CMDB 系统的服务器状态，计算出一个最优化的调度方案，并生成相关调度系统的配置文件。
- ◎ **监控系统：**对节点各类设备的基础监控及业务可用性监控。
- ◎ **告警系统：**CDN 告警系统会做两件事情，一是在告警发生时通知运维人员；二是会同时向调度决策系统发送信令，可以实现某些特定故障的自动化修改调度处理。
- ◎ **自动化接口：**提供给 CDN 业务使用者的一些刷新、预加载等接口。

- ◎ Portal 展示系统：展示 CDN 业务运行的各项数据。
- ◎ 配置变更系统：CDN 运维人员进行各种日常运维操作的系统。

CDN 硬件及节点选型

CDN 的目的是在尽量低的成本下为用户提供尽可能高的服务质量，所以在硬件选型时成本是一个很重要的考虑因素。

针对静态资源类业务，在硬件选型上主要评估在节点出口带宽满载时，所使用的硬件总成本。例如，相同配置的 1 台万兆服务器和 10 台千兆服务器都可以跑满 10Gbps 带宽，这时肯定会选择万兆服务器。当然，在硬件选型时还要考虑实际业务对 CPU、内存、磁盘的使用情况。

CDN 服务器选型策略

静态资源类业务使用的机型规格为 E5-2620 *2 + 128GB RAM + 2TB SAS *4 + 480GB SSD *6。关键指标：万兆网卡+大内存+SSD 存储。使用万兆网卡可以有效避免单机网络吞吐达到瓶颈；大内存可以将更多的数据缓存在内存中，降低磁盘 IO 的使用；SSD 可以提供更好的随机 IO 读/写性能。总体目标就是避免出现木桶效应。

动态加速类业务使用的机型规格为 E5-2620 *2 + 64GB RAM + 600GB SAS *2。因为此类业务对带宽使用量较少，主要是大量 TCP 连接数对 CPU 的消耗，所以在硬件选型上主要侧重于 CPU 和内存，对于磁盘和网卡没有要求。

在做硬件选型时，发现相对于 Broadcom 网卡，Intel 网卡无论是在 CPU 使用率还是小包性能上都要优秀很多，所以无论是千兆还是万兆服务器，都推荐使用 Intel 网卡。

CDN 节点建设策略

伴随着业务发展，会需要陆续新建 CDN 节点，以保证 CDN 的整体带宽使用率控制在一个合理的水平。显而易见，在带宽缺口最大的区域新建 CDN 节点是最优的节点建设策略。通过日志统计系统可以得出各个省份运营商的带宽使用量及使用比例（将日志中的客户端 IP 地址通过 IP 地址库映射为具体的省份运营商，再将每条请求响应的大小以 5 分钟为粒度做聚合，除以 300 秒后即得到各个省份运营商的带宽使用量），再和当前已有节点的分布做对比，即可得到各省份运营商带宽需求及当前带宽供应、当前带宽缺口一览表（见表 11-1，非真实数据，仅做示例）。

省份运营商	带宽需求	当前带宽供应	当前带宽缺口	备注
广东电信	38Gbps	20Gbps	18Gbps	紧缺
辽宁联通	18Gbps	20Gbps	-2Gbps	均衡
江苏电信	30Gbps	40Gbps	-10Gbps	富裕

表11-1 各省份运营商带宽需求及当前带宽供应、当前带宽缺口一览表

确定了在哪些区域新建 CDN 节点后，后续的工作就是实际的选点了。在选点过程中，主要考核候选节点的成本和质量，节点的成本可以用单位带宽成本来衡量，这个不是本文重点，暂且按下不表。

这里主要介绍如何测量评估候选节点质量。为了使评测结果更接近用户真实体验及更具有横向可比性，我们使用 JS 测速来评测一个候选节点的质量。简单来讲，就是让部分真实用户去下载被评估的候选节点的一个固定大小的文件，然后记录每次用户下载的成功率及耗时，统计全部用户的平均下载成功率及耗时作为衡量节点质量的依据。从实践上看，此种方法效果良好。

在节点建设方面，我们采用的是模块化方案，一个节点有多少交换机、多少服务器，交换机以及服务器的型号配置如何，网线怎么连、电源线怎么插，全部都是固定统一的。基本可以说，各个节点的区别就只有 IP 地址是不一样的。并且交换机和服务器都是预先配置好后才发往外地节点上架的。采用这样的模块化方案，不仅可以显著降低使用过程中的运维成本，而且在节点建设中的便利性也是很显著的——无须我们的工程师去现场施工，数据中心现场代维工程师只要参照我们提供的一份标准化上架文档，就可以快速地完成上架施工。并且配置和流程都是高度标准化的，可以最大程度地避免（以及快速发现）上架误操作或漏操作。

CDN 缓存系统

常见缓存服务器的简要对比

- ◎ Squid：老牌的缓存服务器，但是性能较差，无法满足自建 CDN 对于高性能的需求。
 - ◎ Nginx：性能好，插件丰富，支持广泛，但缓存功能偏弱，在某案例中，动态加速服务就是在 Nginx 的基础上修改而来的。
 - ◎ Varnish：配置方便，性能好，但不支持持久化缓存是其一大缺陷。
 - ◎ Apache Traffic Server (ATS)：性能好，对于缓存的支持很完善，但是插件不够丰富，很多业务需求需要自行开发插件来完成。
- 经过综合对比，自建 CDN 的静态加速服务使用了 ATS。

ATS 功能介绍

- ◎ 具有完整的正向代理和反向代理功能，并且支持集群工作模式。
- ◎ 使用 **parent** 配置时可以自动屏蔽故障源站。
- ◎ 具有内存缓存和磁盘缓存自动淘汰功能，支持直接写裸磁盘。
- ◎ 拥有丰富的日志格式配置。
- ◎ 支持插件开发，可以定制业务专有需求。

ATS 配置要点说明

1. 回源配置

回源配置示意图如图 11-5 所示。

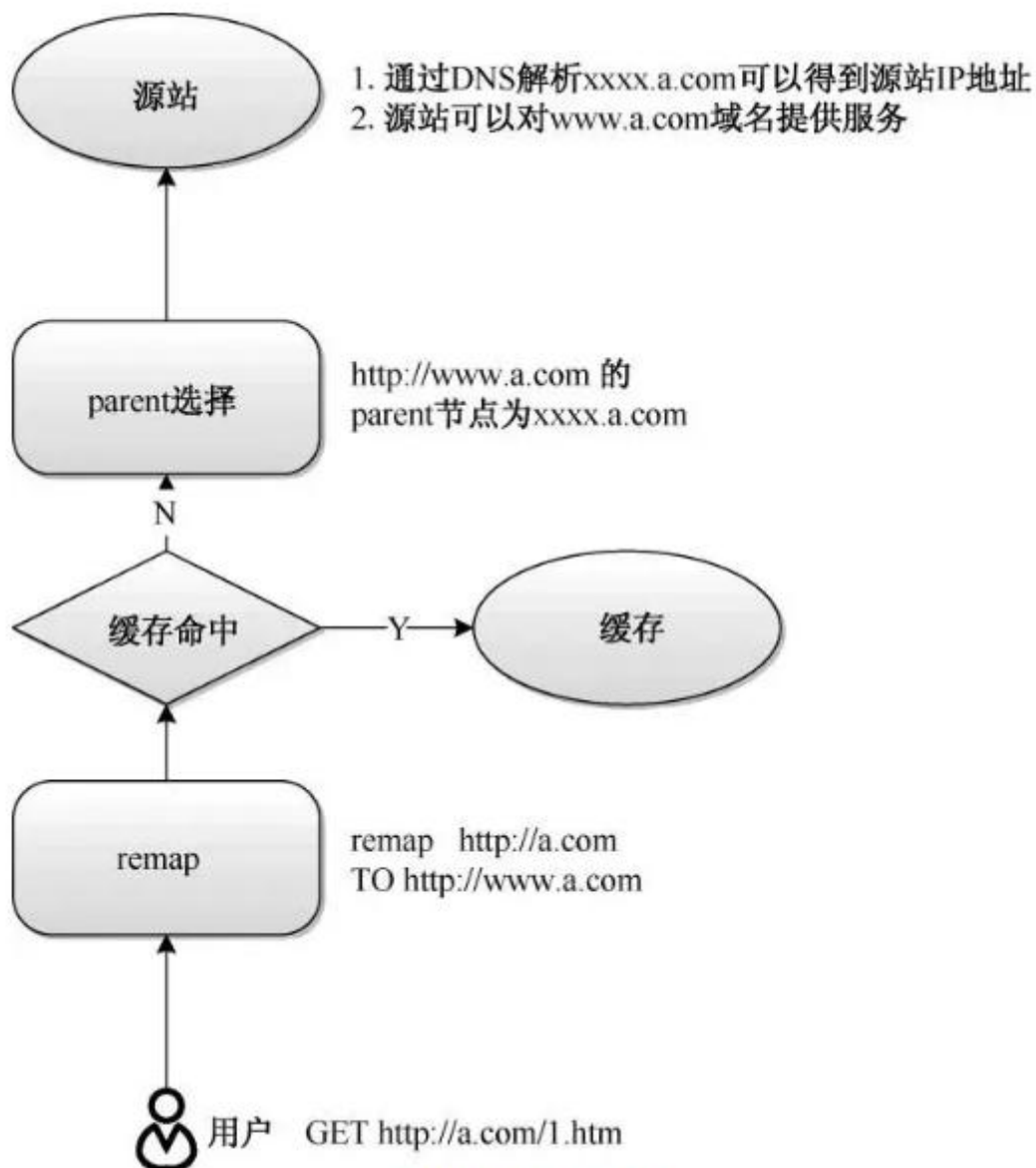


图11-5 回源配置示意图

使用 parent 配置有以下好处：

实现多个源站负载均衡；故障源站自动屏蔽；源站 DNS 解析和回源 Host 解耦合。

remap 和 parent 配置的区别如下：

remap 是将用户访问的域名映射成另一个域名（可以是其本身），映射后的域名将用作回源请求和用来生成缓存的 key。即，如果两个不同的域名同时 remap 到同一个域名，则两个域名下的相同的 URI 指向同一份缓存。而 parent 则负责将回源请求发送到特定的 parent 域名（或 IP 地址），而不是去 DNS 查询 remap 后的域名。

record.config 文件配置如下：

```
CONFIG proxy.config.http.no_dns_just_forward_to_parent INT 1 #启用 parent.config 配置
CONFIG proxy.config.url_remap.remap_required INT 1 #启用 remap.config 配置
CONFIG proxy.config.url_remap.pristine_host_hdr INT 0 #这个配置项的意思是在 remap 后是否保留 HTTP 请求头中的 Host 项，务必配置成 0（不保留）
```

remap.config 文件配置如下：

```
remap      http://a.com          http://www.a.com
remap      http://www.a.com      http://www.a.com
regex_map  http://www(.*)a.com   http://www.a.com
```

上面的三个 remap 配置都是很典型的配置，按字面意思理解即可。

2. 缓存配置

（1）内存缓存配置（record.config 文件）

CONFIG proxy.config.cache.ram_cache.size INT 1G #配置内存缓存大小

CONFIG proxy.config.cache.ram_cache_cutoff INT 50M #配置内存缓存的最大单文件尺寸

CONFIG proxy.config.http.record_tcp_mem_hit INT 1 #记录内存缓存命中

CONFIG proxy.config.cache.ram_cache.algorithm INT 1 #内存淘汰方式，默认为 1（LRU）即可

（2）磁盘缓存配置（storage.config 文件）

ATS 支持两种磁盘缓存方式，一种是以文件的形式缓存在文件系统上，配置项是：/var/cache 500G（路径+存储大小）；另一种是直接写裸盘，配置项是：/dev/sdb（磁盘设备的路径），这时不需要指定存储大小。

需要注意的是，ATS 使用裸盘时需要更改磁盘设备的权限。在 CentOS 环境中可以在 /etc/udev/rules.d/下新建一个 99-trafficserver.rules 文件，并添加如下内容：

```
KERNEL=="sd[c-z]*",MODE="0660",OWNER="root",GROUP="root"
```

（3）interim storage 缓存配置（record.config 文件）

为了解决 SATA 磁盘随机读/写差的问题，ATS 支持内存——SSD（interim storage）——SATA 磁盘三级缓存。将最常访问的内容放到 SSD 中，降低对 SATA 磁盘的随机读数量。相关配置项如下：

LOCAL proxy.config.cache.interim.storage STRING /dev/sdb

注意：interim storage 机制默认是不开启的，请在编译时使用“--enable-interim-cache”参数开启。

更详细的资料请参见官方文档：<http://trafficserver.readthedocs.org/en/latest/>。

CDN 调度系统

DNS 调度

DNS 调度示意图如图 11-6 所示。

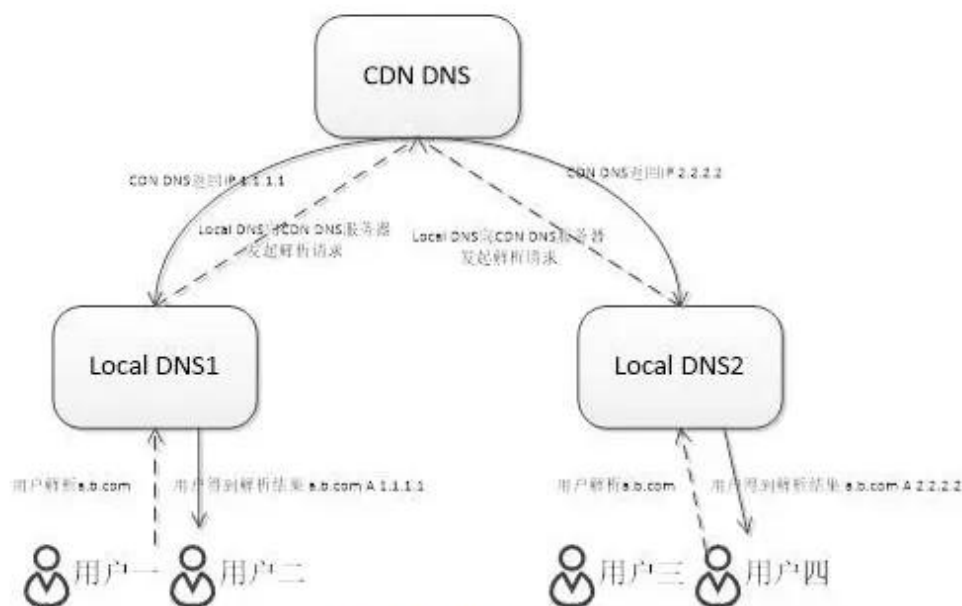


图11-6 DNS调度示意图

DNS 调度是通过对不同的 Local DNS 服务器返回不同的解析结果来实现智能解析和就近调度的。比如，给黑龙江联通的 Local DNS 返回黑龙江联通 CDN 节点的 IP 地址，给广东电信 Local DNS 返回广东电信 CDN 节点的 IP 地址。

CDN 的 DNS 服务主要实现两个需求：智能解析以及 edns-client-subnet 支持。也有很多开源的 DNS 软件支持 IP 解析调度和 edns-client-subnet，如 PowerDNS。在某案例中，CDN 主要使用 DNSPod 服务，小部分使用自建 DNS 服务（后续会逐步迁移到自建 DNS 服务）。自己研发 DNS 服务，主要考虑到一些特殊的业务场景和监控需求。

1. 一个自建 DNS 服务实现简述——SmartDNS

SmartDNS（智能 DNS），根据配置，能够支持针对不同的 DNS 请求返回不同的解析结果。SmartDNS 获取 DNS 请求的源 IP 地址或者客户端 IP 地址（支持 EDNS 协议的请求可以获得客户端 IP 地址），根据本地的静态 IP 地址库获取请求 IP 地址的特性，包括所在的国家、省份、城市、ISP 等，然后根据调度配置返回解析结果。支持 A、SOA、NS 记录的查询，支持 DNS forward 功能。

一个开源的 Python 版本 SmartDNS，供大家参考：<https://github.com/xiaomisa/smartdns>。接下来介绍它的部分细节实现。

SmartDNS 响应 DNS 请求的处理流程如图 11-7 所示。

IPPool 类的初始化和该类中的 FindIP 函数进行解析处理是 SmartDNS 中最关键的两个要素，这两个要素在下面详细介绍。IPPool 类的初始化示意图如图 11-8 所示。

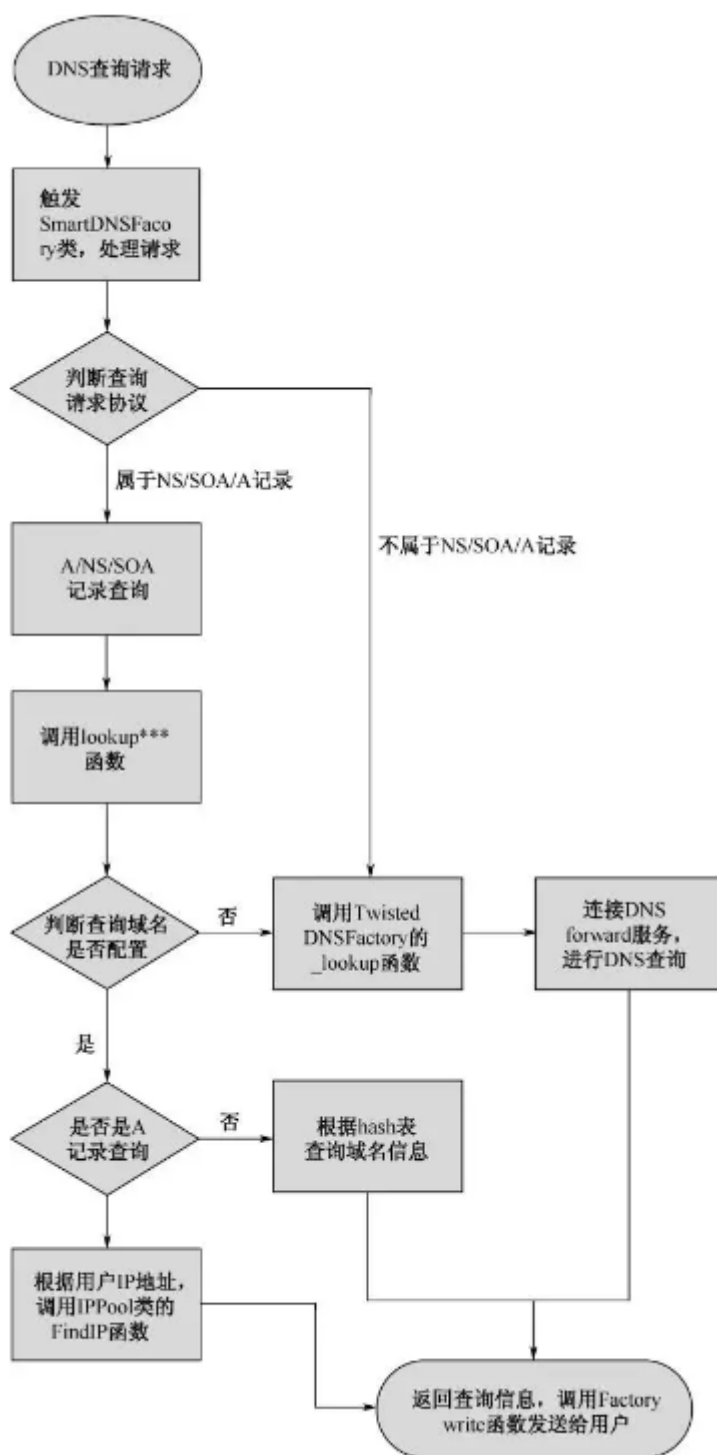


图11-7 SmartDNS响应DNS请求的处理流程

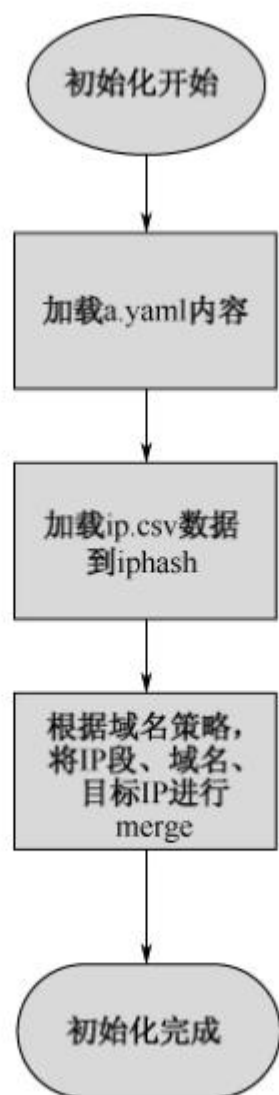


图11-8 IPPool类的初始化示意图

ip.csv 为 IP 地址库文件，格式如下：

200000001, 200000010, 中国, 陕西, 西安, 电信

其中各个字段的含义分别为 IP 段起始地址，IP 段截止地址，IP 段所属国家，IP 段所属省份，IP 段所属城市，IP 段所属 ISP。

a.yaml 配置文件格式如下：

test.test.com:

ttl: 3600

default: 5.5.5.5 2.2.2.2

中国,广东,,联通: 1.1.1.1 3.3.3.1

中国,广东,,电信: 1.1.1.2 3.3.3.2

配置中地域信息的 key 包括 4 个字段，分别带有不同的权重——国家：8；省份：4；城市：2；运营商：1。

在初始化阶段，会生成一个名为 iphash 的字典，结构如图 11-9 所示。

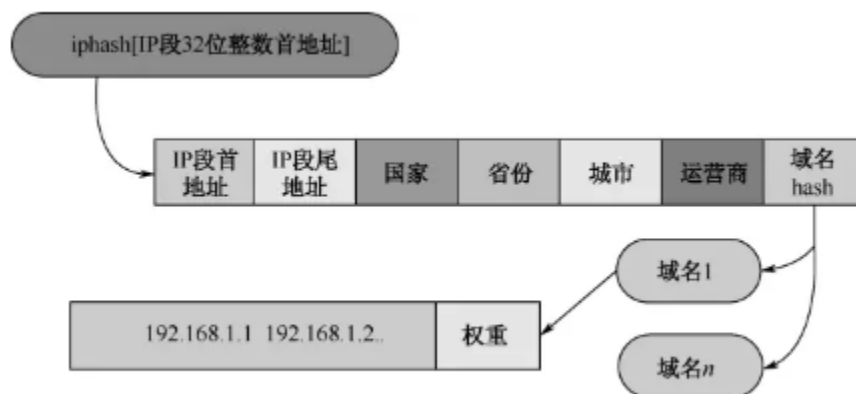


图11-9 名为iphash的字典结构

其中，iphash 的 key 为 ip.csv 每一条记录的起始 IP 地址，value 为一个 list，list 长度为 6，list 的前 5 个字段分别为以该 key 为起始 IP 记录的 IP 段截止、IP 段所属国家、IP 段所属省份、IP 段所属城市、IP 段所属 ISP，第 6 个字段是一个 hash，key 为 a.yaml 里面配置的域名，value 为长度为 2 的 list。iphash[IP 段起始地址][6][域名 1][0] 为域名 1 在该 IP 段的最优解析，iphash[IP 段起始地址][6][域名 1][1] 为该最优解析的总权值，该总权值暂时只做参考。

在 iphash 初始化过程中最关键的是 iphash[IP 段起始地址][6][域名 1] 的最优解析的计算，最简单、直接的方式是直接遍历域名 1 的所有调度配置，挑选出满足条件且总权值最高的解析，即为最优解析。这种方式记录整个 iphash 的时间复杂度为 $O(xyz)$ ，x 为 ip.csv 记录数，y 为域名总数量，z 为各个域名的调度配置数。为了优化启动速度，优化了寻找最优解析的方法：事先将每个域名调度配置生成一棵树，这棵树是用字典模拟出来的，这样需要最优解析的时候就不需要遍历所有的调度配置了，而是最多检索 15 次即可找到最优，即时间复杂度为 $O(15xy)$ 。具体实现请参考 IPPool 的 LoadRecord 和 JoinIP 两个方法。

有了初始化后的 iphash 数据结构之后，每次请求处理的时候，只需要定位请求 IP 地址处于哪个 IP 段，找到 IP 段起始 IP 地址，然后从 iphash 中取出最优解析即可。具体流程如图 11-10 所示。

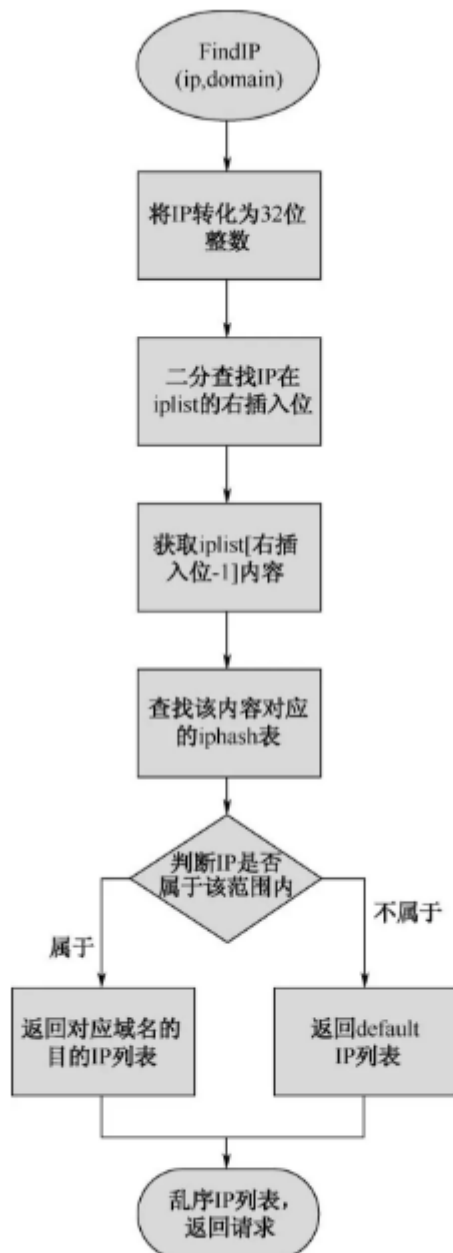


图11-10 计算最优解析的具体流程图

2. edns-client-subnet 支持

CDN 使用 DNS 获取查询 IP 地址，根据 IP 地址对用户进行地域调度。但这里获取的 IP 地址是 DNS 地址，而不是用户真实的 IP 地址。在大多数情况下，我们假设用户通常会使用离自己网络最近的 Local DNS，CDN 调度基本还是准确的。但也有很多 Local DNS 设置错误的情况，或者用户使用 Google Public DNS（8.8.8.8/8.8.4.4）或 openDNS。

比如国内用户设置了 Local DNS 为 8.8.8.8，我们得到的 DNS Query IP 地址是 74.125.16.208，判断 IP 地址属于美国加利福尼亚州山景市谷歌公司，这个时候，我们的 DNS 会返回离美国加州最近的 CDN 节点 IP 地址给用户，于是国内用户错误地调度到美国 CDN 节点上。

为了解决上述问题，Google 提交了一份 DNS 扩展协议，允许 Local DNS 传递用户的 IP 地址给 authoritative DNS Server。CDN 的 DNS 支持该协议后，就可以获取用户真实的 IP 地址，进行准确的调度。

edns-client-subnet 流程示意图如图 11-11 所示。

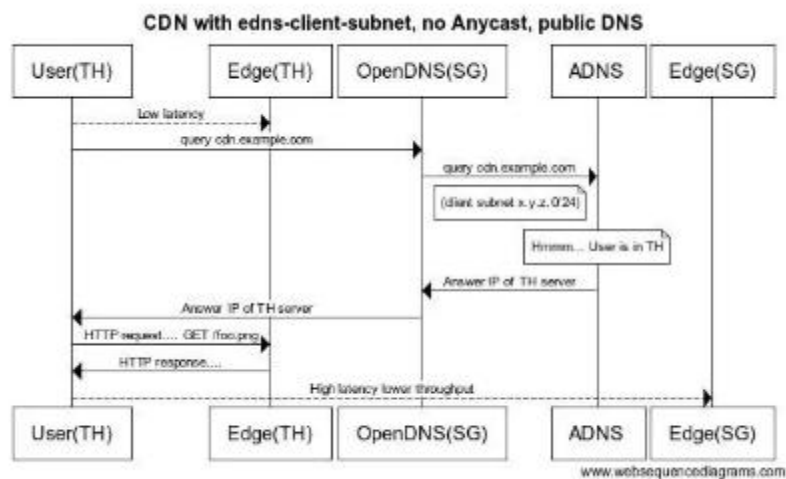


图11-11 edns-client-subnet流程示意图

DNS Query 会包含 header 和 RR 两部分，这里只介绍我们关注的地方，在网上可以搜到很多关于 DNS 协议的介绍，在这里就不做过多描述了。header 会描述本次请求中 Questions、Answer RRs、Authority RRs 和 Additional RRs 的数量，RR 部分会详细描述每个资源的内容，所有的 RR 格式是相同的，如下所示：

edns-client-subnet 是对 EDNS 协议的扩展，附加在一个 DNS 请求的 Additional RRs 区域，这里重点描述 edns-client-subnet 的结构。EDNS 协议 Extension mechanisms for DNS (EDNS0)，请参考 <http://tools.ietf.org/html/draft-ietf-dnsind-edns0-01>。

EDNS0 每个字段的结构和描述如下：

Field Name	Field Type	Description
------------	------------	-------------

NAME	domain name	empty (root domain)
TYPE	u_int16_t	OPT
CLASS	u_int16_t	sender's UDP payload size
TTL	u_int32_t	extended RCODE and flags
RDLEN	u_int16_t	describes RDATA
RDATA	octet stream	{attribute,value} pairs

OPT 的值为 41，详细的协议值如下：

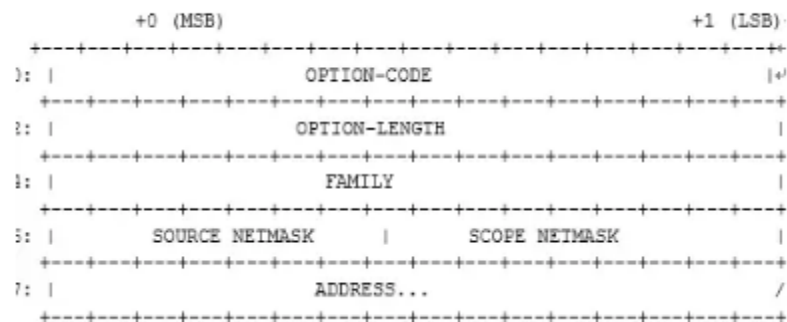
(A, NS, MD, MF, CNAME, SOA, MB, MG, MR, NULL, WKS, PTR, HINFO, MINFO, MX, TXT, RP, AFSDB)
= range(1, 19)
AAAA = 28
SRV = 33
NAPTR = 35
A6 = 38

DNAME = 39

SPF = 99

OPT = 41

RDLENGTH 描述 RDATA 的长度，edns-client-subnet 的详细格式存在 RDATA 中，如下所示：



OPTION-CODE: 两个字节。

OPTION-LENGTH: 两个字节，描述它之后的内容长度 (byte)。

FAMILY: 两个字节，1 表示 IPv4，2 表示 IPv6。

ADDRESS: 实际存放 IP 地址的地方，IPv4 长度为 4，Google 发送过来的长度一般为 3，隐藏了 IP 地址最后一位。

理解了以上协议，就可以动手实现了。判断 DNS Query 是否包含 Additional RRs，如果包含，则按照 EDNS 协议描述获取地址。用获取到的地址进行来源 IP 地址判断调度，封装结果返回。

BIND 9.10 版本后自带的 dig 工具可以支持生成带 EDNS 扩展的请求来验证 EDNS 功能是否生效。命令示例：

```
./dig test.com @127.0.0.1 +subnet=1.1.1.1
```

使用 WireShark 抓取的支持 edns-clinet-subnet 的 DNS 请求截图如下。

发包：

```
▼ Domain Name System (query)
  [Response In: 4]
  Transaction ID: 0xa35a
  ▶ Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  ▼ Queries
    ▼ a.xm.mi0.cc: type A, class IN
      Name: a.xm.mi0.cc
      Type: A (Host address)
      Class: IN (0x0001)
  ▼ Additional records
    ▼ <Root>: type OPT
      Name: <Root>
      Type: OPT (EDNS0 option)
      UDP payload size: 4096
      Higher bits in extended RCODE: 0x0
      EDNS0 version: 0
      Z: 0x0
      Data length: 12
      Data
```

回包:

```
▼ Domain Name System (response)
  [Request In: 3]
  [Time: 0.012421000 seconds]
  Transaction ID: 0xa35a
  ▶ Flags: 0x8580 (Standard query response, No error)
  Questions: 1
  Answer RRs: 1
  Authority RRs: 0
  Additional RRs: 1
  ▼ Queries
    ▼ a.xm.mi0.cc: type A, class IN
      Name: a.xm.mi0.cc
      Type: A (Host address)
      Class: IN (0x0001)
  ▼ Answers
    ▼ a.xm.mi0.cc: type A, class IN, addr 58.68.235.55
      Name: a.xm.mi0.cc
      Type: A (Host address)
      Class: IN (0x0001)
      Time to live: 2 minutes
      Data length: 4
      Addr: 58.68.235.55 (58.68.235.55)
  ▼ Additional records
    ▼ <Root>: type OPT
      Name: <Root>
      Type: OPT (EDNS0 option)
      UDP payload size: 4096
      Higher bits in extended RCODE: 0x0
      EDNS0 version: 0
      Z: 0x0
      Data length: 12
      Data
```

私有调度

1. HTTP 调度

用户通过访问一个 HTTP 接口，HTTP 接口会判断用户的来源 IP 地址及其他诸如业务类型等信息，然后返回给用户一个 JSON 字符串，JSON 字符串中包含相关的调度结果。客户端通过解析 JSON 字符串可以得到所需要访问的服务器 IP 地址列表。

HTTP 调度相比 DNS 调度有以下优势：

可以识别用户来源 IP 地址，调度结果更精确；可以在 URL 中包含更多的业务信息，功能更强大；可以支持一次返回多个查询的结果。

HTTP 调度的不便之处就是要依赖客户端支持。

2. 302 调度

当 Web Server 收到一个用户请求时，通过识别用户来源 IP 地址和其 URL，返回一个 302 响应将用户的请求重定向至一个新的 URL，用户使用新的 URL 再去下载真实的资源。

302 调度可以做到针对单次请求动态调度，比 DNS 调度粒度更细，变更生效也更迅速，并且可以在 URL 上添加防盗链等信息。

302 调度的缺点是增加了一次 HTTP 访问的时间，只适用于下载及流媒体等下载耗时较长的业务，静态小文件不适用于 302 调度。

302 调度既可以独立使用，也可以作为 DNS 调度的补充。自建 CDN 当前就是在混合部署 DNS 调度和 302 调度，将访问国外 CDN 节点的中国大陆用户再重定向至国内 CDN 节点。

全局调度逻辑

为了实现更好的业务容灾，当前公司业务都是在多 CDN 供应商同时部署的。在需要时可以通过 DNS View 功能实现区域用户的导向；同一区域用户可以通过第三方和自建互备，提高了业务的容灾能力（见图 11-12）。

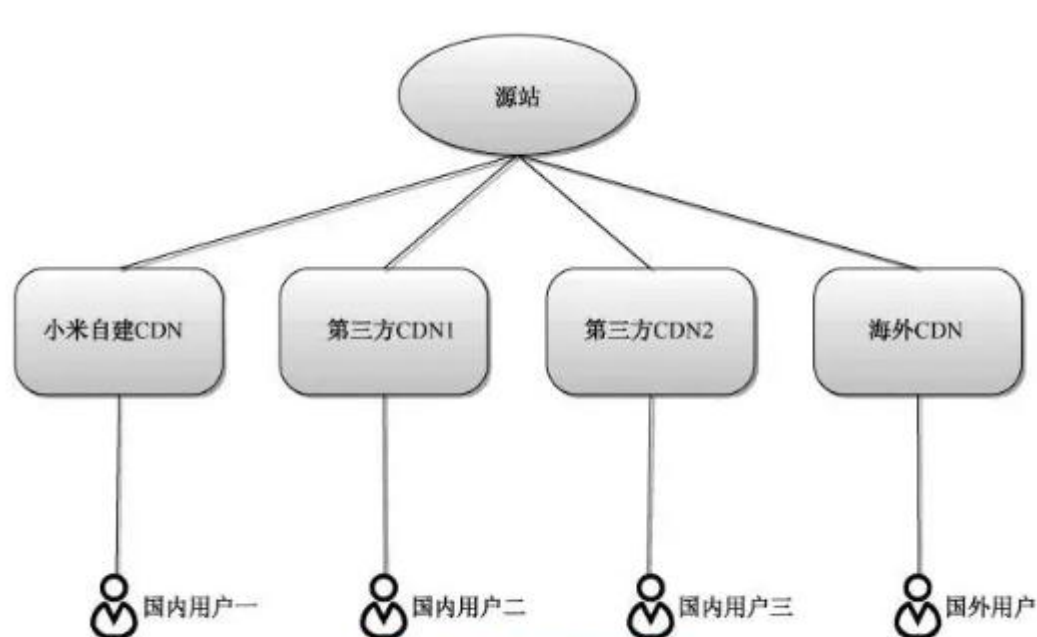


图11-12 全局调度逻辑示意图

IP 地址库维护

不管使用什么样的调度方式，一个准确的 IP 地址库都是调度系统中重中之重的部分，并且 IP 地址库在诸如日志分析等方面也有着重要的用途。所以维护一份可靠的、高准确度的 IP

地址库的工作具有重要意义。

我们的 IP 地址库最开始是直接使用网上的纯真 IP 地址库，这个地址库现在来看是一团糟，地理位置标记很不规范。后来陆续使用业内的收费 IP 地址库及友商的 IP 地址接口对此地址库进行了矫正规范，最终得出了一个格式规范、可信度比较高的地址库。

首先，我们将 IP 地址库的格式及地理位置命名进行了规范，纯真 IP 地址库大量存在某某学校、某某网吧的地理位置。规范后 IP 地址库格式为：起始 IP 地址，结束 IP 地址，国家，省份，城市，运营商。规范化格式带来的好处就是后期使用及合并裁剪都变得轻而易举。

其次，我们通过收集和购买基础库，采用投票方法进行整合，对 IP 地址库进行了矫正，得到自己的基础库。矫正的流程是，我们将整个 IP 地址库拆成以 C 段为单位，分别和其他的 IP 地址库进行比对，按照多数服从少数的原则对每个条目的信息做聚合，最后得出一个集各家之长的 IP 地址库。

在得到集各家之长的 IP 地址库后，我们和公司业务团队合作，拿到了几千万条 IP 地址和 GPS 地理坐标的关系数据（这些数据是通过天气 App 获取的，可信度很高），通过这些数据校验和进一步修正我们的 IP 地址库。经过上述一系列校验、矫正后，针对个别仍不能确认的 IP 地址，我们通过多节点 `traceroute`，拿到各地到此 IP 地址的路由路径，以此来反推 IP 地址的归属。

海外 CDN 建设

自建还是第三方

当我们考虑海外自建 CDN 时，有很多现实的问题摆在眼前：服务器使用物理机还是虚拟机？服务器购买还是租赁？数据中心托管怎么解决？现场代维如何做？困难重重。反向比较的话，海外自建 CDN 相比海外现成的第三方 CDN 服务有什么优势？访问质量更好，还是成本更低？仔细评估下来，出于快速部署和扩展的考虑，主要采用了成熟的产品。海外是按照流量计费的，所以在数量小的规模下，自建优势不大。后续随着量的增长，我们会在合适的阶段评估自建。

具体到海外自建 CDN 节点，针对某些用户量大、我们自身掌控力也强的地区（例如中国香港地区），我们采用部署物理服务器的方式将节点直接建设到本地；针对其他用户量大的地区，则使用第三方云服务虚拟机进行节点部署。

海外源站部署

在 CDN 海外部署的过程中，我们遇到最多的问题就是海外回源：海外 CDN 节点回源到国内质量差，还会由于各种原因导致回源连接被墙。针对上述问题，我们在海外 CDN 回源上也做了很多优化。

针对一些文件总量较小的业务（官网静态页面等），我们将海外源站直接部署到了海外虚拟机上，这样就完全规避了回源的问题。

针对一些文件总量较大或者发布时效性要求比较高的业务，我们在海外源站新增了一个 Proxy 层，Proxy 层会优先将回源请求代理到海外源站，针对暂时没有发步到海外源站的资源会代理到国内源站。

针对一些不便进行海外源站部署的业务，我们在中国香港地区做了一组中间层缓存服务器，通过香港地区的缓存服务器再统一回源至国内。

总而言之，尽可能地降低海外 CDN 回源到国内源站的次数，提升回源速度，降低被墙的风险。这是海外源站部署的主体思路。

质量评测系统

质量评测的目的是评估 CDN 服务质量的好坏，而监控系统则是为了及时发现服务中的问题并告警、处理。质量评测关注大层面的服务质量，监控系统关注细节的服务质量，这两个系统的主要目的还是有区别的。当然，质量评测的数据发生异常波动时，也会发送告警、处理。

客户端测试

从客户端角度去评测一个服务的质量情况，大体上可以分成以下几种方式。

（1）类似于博睿所提供的服务，通过部署在全国各地的测试节点去定期访问我们的被测试业务，统计相应的成功率和访问耗时。此种方式由于可以拿到详细的访问数据，所以在发现问题时更容易定位问题的原因。但是劣势也很明显，需要在全国各地部署节点或者使用第三方的付费服务，并且测试样本偏少，不能有效地代表真实用户，所以此种方式在我们的质量评测中已经很少使用了。

（2）对于有客户端的业务，我们联合业务开发，对一些关键的网络交互环节进行打点，统计访问成功率及访问耗时，这样得出来的结果是业务真实的质量数据，可信度及数据的丰富程度都很高。缺点是需要有客户端支持，不是所有的业务都有此条件。

（3）对于 HTTP 类的业务，通过在 Web 页面中嵌入一个 JS 脚本，使用 JS 脚本去下载业务的一个资源，统计访问的成功率及访问耗时。相比客户端打点，此种方式具有更广泛的适用性，只要被测试的业务是 HTTP 类型即可（CDN 绝大部分业务都符合此条件）。当前我们大量使用的是此种测试方式。

服务器抓包分析

我们开发了一个工具，可以抓取用户访问的 TCP 流，通过服务器端全流量分析，可以对访问请求分析 TCP 流各个过程耗时情况（例如三次握手完成后，服务器返回了某个特定信息，累积下载了 1MB 的文件等）。通过大数据分析，可以得到各用户单元到不同节点的整体图表，对调度表格进行循环优化。下面就简单介绍一下这个抓包工具的实现。Tcpxm 已经开源：<https://github.com/xiaomi-sa/tcpxm>。

Tcpxm 是使用 Python 开发的，调用 Pylibcap 进行抓包的工具。它共有 3 个线程：一个负责抓包并分析内容；一个负责写日志；一个用来清除过期数据。Tcpxm 可以很方便地抓取和分析 TCP 请求，打印成所需要的日志形式。

最初我们用它来抓取和分析某个 App 的登录时间，稍加改造就可以抓取网站访问等时间，计算用户建立 TCP 连接的时间、第一次发包的时间等。如下是我们之前抓取 App 登录时间的日志格式和每个字段说明。

```
2012-09-13 21:25:25 tcpxm.py [line:229] [INFO] 221.179.36.189:3103->xxx.xxx.xxx:2424
[usr:54298295] [login(t6-t0+rtt):2760] [t1:0] [rtt:217] [t3:137] [t4:0] [t5:118] [t6:2069] [t7:193]
```

在此用户的登录时间 = t_6 （发送<success>的时间） - t_0 （收到 SYN 的时间） + rtt（估算出的收到 SYN 包和发送 ACK 包的路径时间）。

具体每个 t 代表的时间含义如图 11-13 所示。日志中的 $t_3 = T_3 - T_2$, $t_4 = T_4 - T_3$, $rtt(t_2) = T_2 - T_1$, ……

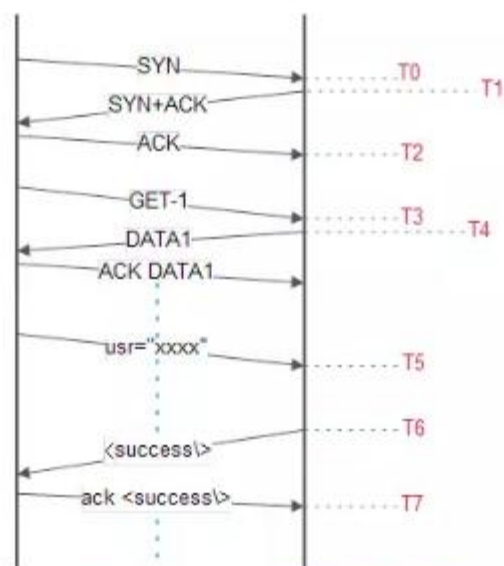


图11-13 某个App登录过程时序图

CDN 故障处理预案

在 CDN 系统运行过程中，各种大大小小的故障或者突发情况是不可避免的。因此，针对各种已知故障进行归纳总结，整理成故障处理预案，并逐步沉淀成自动化的处理过程。通过这样的不断迭代，可以不断提高服务能力，降低故障发生的概率；在故障发生时可以快速有效地介入处理，减少故障影响的范围及时间。

在 CDN 服务中，某业务突发增长数倍甚至十几倍的情况还是时有发生。针对这种情况，我们建立了详细的应对预案。

故障场景：业务突发（未提前通知）导致 CDN 整体带宽利用率超过 80%，且有进一步上涨的趋势。

关键动作：联系业务同事，确认业务突发原因（推广活动还是受攻击，抑或是其他）；通盘评估，确定对此次突发是进行资源保障还是资源打压，并和业务同事达成一致；发出通告，通知可能受此影响的第三方业务。

CDN 故障处理预案：

（1）我们的所有 CDN 业务都是多厂商部署的，并且平时对热门的资源在各厂商都有做预加载，当某一家资源不能满足业务需求时，可以快速灵活地将部分流量调度到其他厂商。

（2）我们在 CDN 交换机上预配置了高、中、低等不同的 QoS 策略，通过切换业务使用的 CDN 域名，让突发业务匹配高优先级的 QoS 策略，可以重点保证此业务的质量。如果需要打压突发业务的流量，则让突发业务匹配低优先级的 QoS 策略。

（3）在极端情况下，会损失突发业务的部分可用性，隔离突发业务。修改调度，将突发业务的流量调度到预配置的有限个节点，将其流量和正常业务从物理上隔离（类似数据中心将受攻击的 IP 流量牵引到黑洞），只提供有限的服务能力。

ZYXW、参考

《运维之下》

第六章、系统运维概述

http://mp.weixin.qq.com/s?__biz=MzIxMzlyNDkyMw==&mid=2654108056&idx=1&sn=7de0d33032cf15381663b8ee6b168db9&scene=21#wechat_redirect

第八章、服务器硬件测试选型

http://mp.weixin.qq.com/s?__biz=MzIxMzlyNDkyMw==&mid=2654108142&idx=1&sn=5cbb2db7ed27c8188d7cee9c7acea42d&chksm=8c7cc55abb0b4c4cf3e20ad41c84015c146163cc93e359a50b81ee3618a842a2ea7dbf184db9&scene=21#wechat_redirect

第四章、服务器资源使用率

http://mp.weixin.qq.com/s?__biz=MzIxMzlyNDkyMw==&mid=2654107999&idx=1&sn=3d136086

9fb5747f3d859eee538459b3&scene=21#wechat_redirect

第九章、网络成长之路

http://mp.weixin.qq.com/s?__biz=MzlxMzlyNDkyMw==&mid=2654108187&idx=1&sn=20149a4e47f58e6c9537c463eca40890&chksm=8c7cc6afbb0b4fb99b65a53939ccad693745e0cde3a7b5d30d3f4b93a7c269e881cd2a031912&scene=21#wechat_redirect

第十章、传输网建设

http://mp.weixin.qq.com/s?__biz=MzlxMzlyNDkyMw==&mid=2654108195&idx=1&sn=e59373bfa181f9d8cd62f3c18f3ef5dd&chksm=8c7cc697bb0b4f8178d1f2944e5ff428865c3966f0b01d5564078a6e2c5c7c7765120961a2fa&scene=21#wechat_redirect

第十二章、域名和接入

http://mp.weixin.qq.com/s?__biz=MzlxMzlyNDkyMw==&mid=2654108295&idx=1&sn=310444b038b2519c8c0ba902dc486e20&chksm=8c7cc633bb0b4f25939ab115bf13dddb78ae531a0f413d997d2c6d551ad8a200cc74d597e731&scene=21#wechat_redirect

第十一章、CDN

http://mp.weixin.qq.com/s?__biz=MzlxMzlyNDkyMw==&mid=2654108274&idx=1&sn=210c1f1415cf7a2b757ff121cf8f8734&chksm=8c7cc6c6bb0b4fd072aad99b387bef2416eb8b6aae6f5670cd1539b0aa656f9a2419c3761bfe&scene=21#wechat_redirect