ASDN

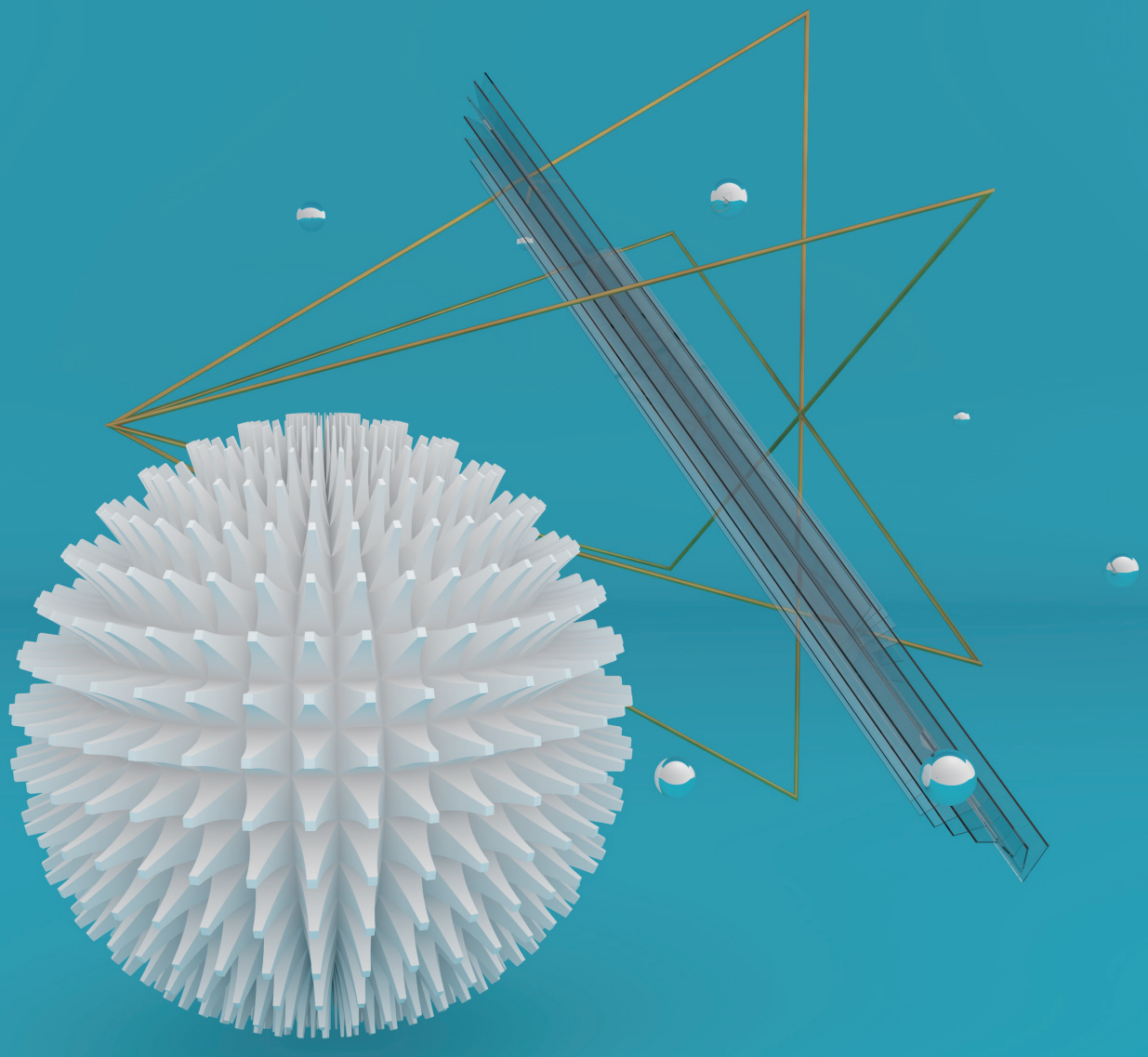
AlibabaCloud Software Developer Network

#2

阿里云ASDN月刊

总第2期

2017.01



阿里云

阿里云 ASDN 月刊

AlibabaCloud Software Developer Network

主编　净业

顾问　广慧　路平　风驰　仙客　云篆

编委　耘睿　念想　徒骇

设计　曳影　野一　十渡

主办部门　阿里云产品应用技术部

合作方　阿里云飞天体验技术部　阿里云研究中心

云栖社区　云市场

鸣谢　铭晗　不老　乐果　狄公　颖杰

芸珊　孙慧丽　黎山　黯灭　念想

鸿猷　隽勇　李俊　师长　予与

振宇　简志　觉宇　法喜　羲遥

李泉　晓哥　炳烛　徒骇　酒道

丁原　隐林　储文姬　一荷　叔度

厚泼　翔贺　水月　玄惭　席嘉

（ 排名不分先后 ）

胡晓明

阿里云总裁

阿里用电商、金融、物流本身的体温，在给中国和阿里云的客户提供技术的样板。

只有把自己的命押上，我们才能进一步驱动技术的变化，这个事情我们100%在做，但是没有几家企业是敢这么做的。

Game Operation Solutions
游戏数据运营解决方案

陆宝
阿里云专家

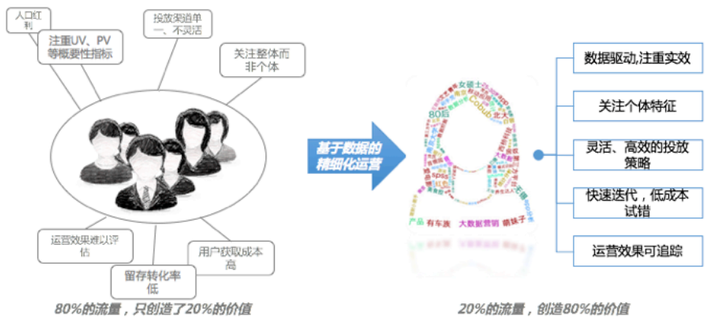
翔贺
阿里云技术专家

背景

1.行业综述

随着游戏行业市场竞争局面的扩大，玩家对于品质的要求越来越高，游戏项目的生命周期越来越短，直接影响项目的投入产出比，通过数据运营则可以有效的延长项目的生命周期，对各个阶段的业务走向进行精准把控。而随着流量成本的日益上升，如何构建经济、高效的精细化数据运营体系，以更好的支撑业务发展，也变得愈发重要起来。

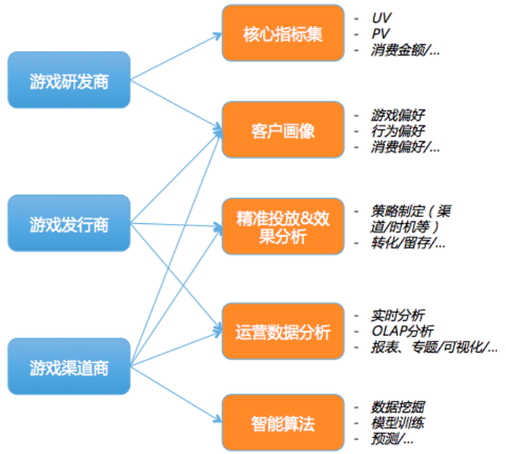
在这样的背景下，越来越多的游戏企业加入到数据运营行列，也促进了大数据产业生态链的发展，第三方数据公司TalkingData、Dataeye、友盟、热云数据等就是在这一个时代中快速成长起来的。但同时受限于业务理解，通用平台往往无法满足游戏企业的定制化需求，所以随着业务的发展，最终还是要选择自建数据分析平台——因为技术门槛、资源投入等原因，目前大部分游戏企业只实现了数据统计，少部分企业实现了数据挖掘，在深度学习层面目前趋近空白。



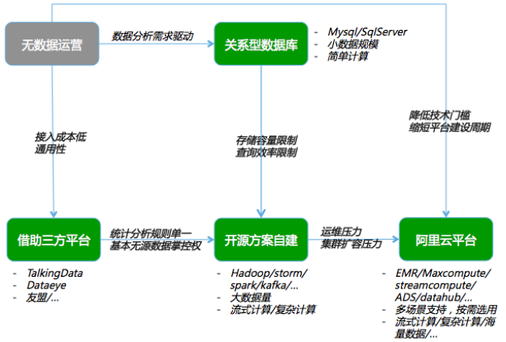
2.业务需求及痛点分析

按照游戏领域的行业细分，不同类型的公司对数据化运营的业务需求各有侧重，构建数据化运营平台的技术手段也表现为不同的方式。

按照行业属性，可以将生态中的公司分为游戏研发商、游戏发行商、游戏渠道商三类，根据业务特点他们对于数据运营的需求也各有侧重，从表现形式讲，基础指标集、客户画像、精准投放&效果分析、智能算法等等，不一而足。



而从实现数据运营的技术手段来分析，也分别表现出不同的特征，各阶段使用的技术栈、驱动因素及演进方向，可以简单通过下图来表述：



而在这样的业务背景下，传统来料加工、被动响应的数据处理架构，显然无法匹配数据化运营的分析需求，主要存在的问题：

1、数据来源单一，缺少精准用户画像，运营策划、实施不能”投其所好”，用户转化率低；

2、平台沉淀、积累了大量的数据，但是通过数据驱动业务创新、辅助决策方面没有经验，导致数据业务价值的转化率低；

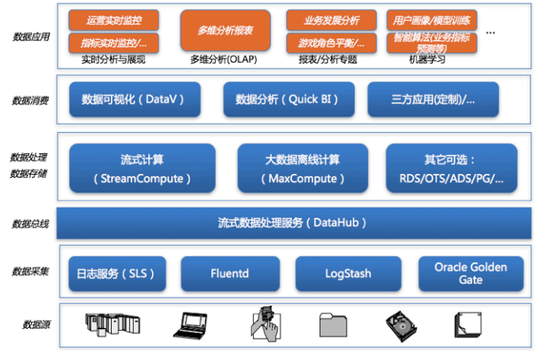
3、开发定制化，项目实施周期长，响应需求速度慢，无法有效支撑业务的灵活变化；

4、数据的应用场景单调，大多只是止于简单的看指标、报表，对于机器学习等复杂场景缺少技术储备；

阿里云整体解决方案

1.功能架构

基于阿里云大数据平台，构建一站式数据运营支撑平台。



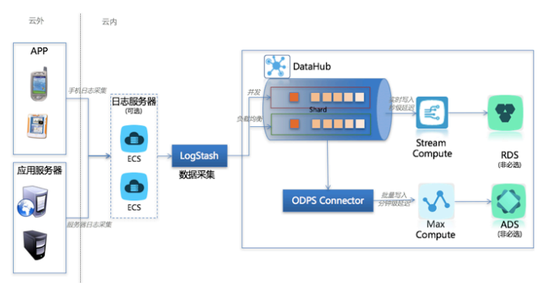
重点简介：

1）数据采集+数据总线，丰富业务数据源：基于开源框架封装的多种数据采集工具，按需选用以支持不同类型的异构数据采集，丰富业务数据来源，同时配合阿里云提供的流式数据处理服务（DataHub），轻松构建基于流式数据的高可用，低延迟，高可扩展，高吞吐的分析和应用；

2）数据处理/数据存储，支持不同场景的数据化运营需求：数据处理/数据存储作为总线数据的消费端，提供面向不同应用场景（实时分析、OLAP、离线计算、智能算法等）的数据计算/存储引擎，支持不同层次、视角的数据化运营需求；

3）数据消费，发挥数据在业务创新、辅助决策方面的价值：基于底层的数据计算能力，在应用侧通过阿里云提供的可视化大屏（DataV）、数据分析配置工具（Quick BI）轻松构建不同场景的数据分析应用，充分发挥数据的业务价值。

2.数据运营平台基础数据框架



重点简介：

1、 Datahub：

a) 实时、高吞吐的并发数据处理能力；

b) 数据自动冗余多份，高可用性保障；

c) 数据流吞吐能力动态伸缩，按需弹性等能力；

2、 Stream Compute：

a) 支持类SQL语法，深度整合多类云数据存储

b) 性能优越，关键指标超越storm6~8倍

c) 优化执行引擎，计算任务资源消耗低

主要支持实时数据处理、分析等应用场景；

3、 Max Compute：

a) PB级超大规模数据的计算及存储

b) 支持丰富的计算模型

c) 数据自动冗余多份，高可用性保障

主要支持多维分析(T+1，OLAP)、报表/分析专题、机器学习等应用场景；

3.产品技术架构

离线分析

为了更加全面的了解业务情况、用户行为偏好，需要对积累的游戏数据做多维度的深度探查，以挖掘数据中蕴含的业务价值。这一类场景的普遍特点是：

1、数据量大

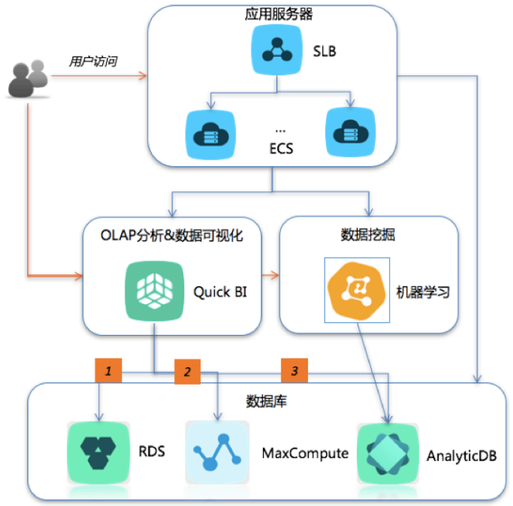
2、计算复杂度高

3、分析视角&可视化的要求灵活多变

4、允许一定的数据时延

常见的产品形态有OLAP报表（多维分析、用户行为分析等）、分析专题（游戏角色平衡等）、数据挖掘（用户画像、业务预测等）等。

实现此类场景的基本产品技术架构如右图所示：



架构重点简介：

1、数据库：按照不同的应用场景、数据规模，可以选用合适的计算/存储引擎；

2、OLAP分析&数据可视化：Quick BI提供拖拽式、所见即多得的OLAP分析&报表的配置及可视化能力，无需代码开发，可快速实现数据化运营的分析场景；

3、数据挖掘：机器学习提供可视化方式的算法配置平台，快速实现机器学习、模型训练、智能算法（预测、偏好分析等）等高复杂度的数据应用场景；

同时，对于应用侧提供灵活的可集成能力，可以完全使用可视化配置，也可组合使用定制开发的方式实现。

实时分析

实时分析更多应用在监控关键的业务指标，以及时获知游戏运营的动态，指导业务部门及时调整业务策略，快速响应业务的变化，这一类场景的普遍特点是：

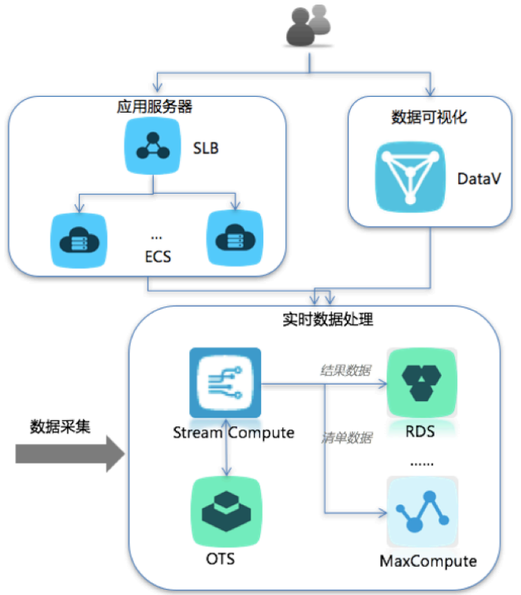
1、单位时间内数据量中小规模

2、计算复杂度一般

3、并发&数据实时性要求高等

常见的产品形态：运营实时监控、PV、UV、消费金额、在线人数、区域分布等业务核心指标实时监控等。

实现此类场景的基本产品技术架构如图：



架构重点简介：

1、实时数据处理：通过阿里云流式数据总线（Data-Hub）+流计算平台（Stream Compute）的组合，构建高可用、高并发、可弹性扩容的实时数据处理、计算模块；

2、实时数据分析：通过DataV平台提供的多种可视化控件，可以快速实现大屏类的实时分析场景，同时如果有定制要求，也可只使用底层数据源的计算能力，通过自建web端应用的方式提供实时数据分析能力。

方案优点总结

针对前文总结的几个业务痛点，提供对应的能力支撑，分别描述：

1、业务数据来源单一

a) 支持多种数据类型；

b) 高可用、多并发流式数据总线，保障数据处理效率；

c) 根据不同的应用场景，按需选用合适的存储/计算引擎；

2、应用场景单调，数据 à 业务价值转化率低

a) 游戏行业模型抽象/核心指标体系构建；

b) 常见业务分析场景总结、提炼、固化；

c) 通过工具支持模型训练、机器学习（客户画像、精准投放、用户行为预测）等复杂场景，无需代码开发；

3、开发运维的效率低

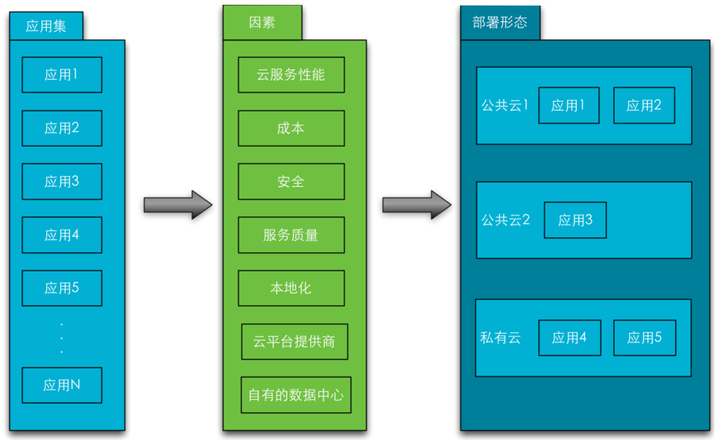
a) IAAS层应用、计算资源基于云平台提供的弹性能力，按需伸缩，无需人工介入；

b) PAAS层提供指标、报表、分析页面等分析场景的配置能力， 无需代码开发；

企业应用如何解决Multi-Cloud
基础设施管理及应用部署问题

黎山
阿里云高级专家

“Multi-Cloud”即企业将多个应用部署在多个云平台上，比如阿里云、AWS、Azure、OpenStack或其他云平台。通过下面的模形我们可以看到企业选择Multi-Cloud的因素及应用部署策略，包括7个主要因素：云计算平台数据中心的访问速度/云服务的性能/可用性、云计算平台的成本、云计算平台的安全性、客户服务质量、本地化、云计算平台的提供商、以及企业自有的数据中心地点。



RightScale的调查数据显示目前有越来越多的企业选择Multi-Cloud模式，用户规模已经从2015年58%占比，增长到2016年的71%。

虽然越来越多的企业选择Multi-Cloud模式，但其也有很多挑战，例如各个云服务平台采用不同的技术、不同的用户操作界面、提供不同的云服务、不同的专业术语。即便是相同的服务（如云服务器），由于是来自多家云计算平台，其API的定义不同，SDK也不同，各自提供的小工具也不能适用于其他云平台。同时对于企业的运维人员要求也较高，运维人员要知道怎样构建各个云计算平台的基础设施，以及部署、运维、监控等，以及各个应用对于云计算平台的最佳选择。

本文章将讲述针对于Multi-Cloud模式的基础设施管理及应用部署的解决方案。

Terraform（https://www.terraform.io）是来自HashiCorp家族的开源工具，通过“WRITE，PLAN，AND CREATE INFRASTRUCTURE AS CODE”管理多个云计算平台的基础设施，支持阿里云、AWS、Azure、GoogleCloud、DigitalOcean等。他通过一致的模板形态定义基础设施的创建/更新/销毁的全生命周期。下面列举了阿里云和AWS的模板，可以看到模板形态一致，定义resource，填写不同的参数，比如数量、镜像、实例类型，对于Multi-Cloud的多平台基础设施管理将大幅度降低学习成本：

```
resource “alicloud_instance” “app” {
  count = 5
  image_id = “ubuntu1404_64_40G_cloudinit.raw”
  instance_type = “ecs.n1.small”
}

resource “aws_instance” “app” {
  count = 5
  ami = “ami-408c7f28”
  instance_type = “t1.micro”
}
```

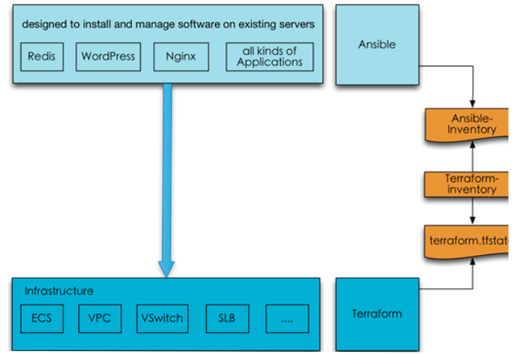
阿里云针对于 Terraform Provider的官方仓库地址：
https://github.com/alibaba/terraform-provider
其中Example包括云服务器ECS、负载均衡SLB、安全组SecurityGroup、专有网络VPC、路由器VSwitch、Nat网关NatGateway的模板，利用这些模板可以做资源的创建/更新/销毁的管理，如创建多台ECS后，再统一修改Tag；或挂载、卸载SLB的后端服务器等等。

Terraform的命令比较简单，最重要的只需要记住3条命令
terraform plan （预览要创建或更新的资源）
terraform apply （创建或更新资源）
terraform destroy （销毁资源）

运行 terraform apply命令后会在本地生成 terraform.tfstate文件，此文件做为更新、销毁资源的依据。

同时根据terraform.tfstate文件，由另一个开源工具Terraform-inventory（https://github.com/adammck/terraform-inventory）可直接生成Ansible的Inventory文件，Ansible利用这个Inventory文件将其用playbook编写的应用部署过程部署到相应的云服务中，实现基础资源管理到应用部署的闭环。

某企业的应用采用Terraform+Ansible的架构将其应用从OpenStack迁移至阿里云，只用了1.5天的时间，而使用传统的手工模式需要一周多的时间。



下面以创建云服务器ECS及磁盘，更新其数量，然后部署Nginx为例详细讲解操作过程。

1. 下载Terraform：
https://www.terraform.io/intro/getting-started/install.html

2. 设置环境变量时，需要指向terraform所在的父目录，如：terraform的路径是”~/work/terraform_0.8.2”，则指定环境变量时设定为export PATH=$PATH:~/work/terraform_0.8.2

3. 下载terraform-alicloud-provider，各平台的下载地址如下：
Mac OS X 64-bit：
http://tf-mac.oss-cn-shanghai.aliyuncs.com/terraform-provider-alicloud.zip
Linux 64-bit：
http://tf-linux.oss-cn-shanghai.aliyuncs.com/terraform-provider-alicloud.zip
Windows 64-bit：
http://tf-windows.oss-cn-shanghai.aliyuncs.com/terraform-provider-alicloud.exe
并将其放置与第2步中terraform相同的目录下，如”~/work/terraform_0.8.2”

4. 安装terraform-inventory，运行命令“brew install terraform-inventory”，详细的说明请参见https://github.com/adammck/terraform-inventory

5. 编写云服务器ECS的模板，如下代码
然后依次运行命令
terraform plan //按照提示输入您的AK信息
terraform apply //按照提示输入您的AK信息

6. 此时登录ECS控制台可以看到创建的云服务器。

```
    resource “alicloud_instance” “webserver” {
        count = 1

        # cn-beijing
        availability_zone = “cn-beijing-b”
        security_groups = [“****”] //需要替换为真实的安全组ID
        allocate_public_ip = true
        instance_charge_type = “PostPaid”
        instance_type = “ecs.n1.small”
        internet_charge_type = “PayByTraffic”
        internet_max_bandwidth_out = 5

        system_disk_category = “cloud_efficiency”
        image_id = “ubuntu_140405_64_40G_cloudinit_20161115.vhd”
        instance_name = “tf_snat”

        tags {
                role = “webserver”
        }
    }
```

7. 修改第5步中的模板”count”参数，将其修改为2，再次运行：

terraform plan //可以看到变更的资源，即新增加一台ECS以及磁盘

terraform apply

8. 此时登录ECS控制台可以看到又创建一台云服务器

9. 编写Ansible的playbook安装nginx，playbook如下：

```
- hosts: instance
  tasks:
  - name: Install Nginx
    apt: pkg=nginx state=installed update_cache=true
  notify:
  - Start Nginx

  handlers:
  - name: Start Nginx
    service: name=nginx state=started
```

10. 执行命令：

ansible-playbook -i `which terraform-inventory` play-book.yml -u root -k -c paramiko -vvv

11. nginx即安装在以上2台ECS上，可以ssh登录ECS云服务器查看或修改nginx的配置。

12. 如果想释放两台ECS及磁盘，执行 terraform destroy 即可

至此创建ECS云服务器、修改数量、安装Nginx完成。通过Terraform+Ansible的方式可以把基础资源的管理和应用部署分开，Terraform的模板依照一致的模板形态实现对Multi-Cloud的基础设施管理，Ansible的playbook可以被重复执行将应用部署在不同的云平台上。

阿里云会持续提供更多的与流行开源工具的集成为企业的云上部署及运维提供便利。

由于篇幅有限，本文只讲述了基础设施管理及搭建Nginx应用的实践，关于更多的应用场景下Terraform的最佳实践我们会持续发布。大家有问题也可以在github（https://github.com/alibaba/terraform-provider）的Issue中提问，欢迎大家关注。

API Solutions

API经济时代的思考

森屿
阿里云产品专家

API经济，是指企业通过API建立合作关系而产生的经济活动。这已经不再是一个简单的概念，很多企业已经运用在商业活动之中，通过API粘合更多合作伙伴，扩充企业服务场景，促进企业的转型和升级，甚至重构整个行业的商业价值链。

SOA面向服务的架构出现，很多企业开始将跨平台、无状态的API是用来作为系统间创建联系的通道，以消除前期系统建设的信息孤岛。从而降低协同成本，提升利润空间。API开始受到空前的重视，成为各行业中驱动数字化、信息化变革的主要力量。

Web2.0出现，先进的企业将自身业务系统的服务能力通过API有限开放，开发者、合作伙伴通过重组不同API服务，并整合到自己的应用、网页，衍生出新的服务。从而串联企业的上下游，同合作伙伴形成经济共同体、相互弥补服务场景来发展新的业务以满足市场的需要，加速产品迭代。从而提升市场影响力、带来直接或间接的经济效益。

传统行业也意识到互联网+的重要性，用API开放服务整合线上、线下资源。比如银行允许第三方点在钱包实现快捷支付、12306供第三方的票务查询、交通局的违章查询等都是通过API来提供的。

传统架构　　API经济

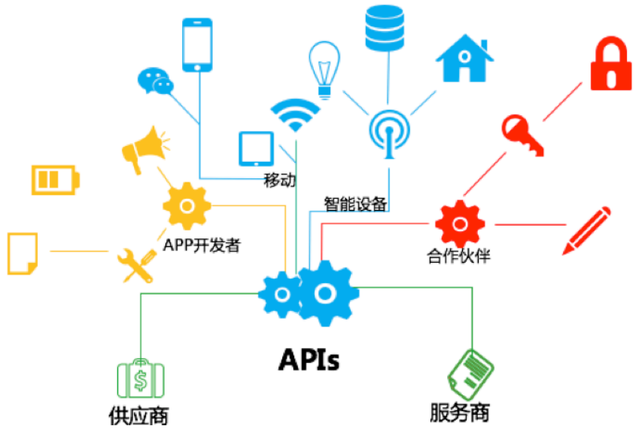
随着云计算、移动、万物互联的到来，大数据、机器学习的兴起，互联网和实体经济结合，引发商业模式的重大变革。更是加速了API的发展。

企业不断的强化自身核心竞争力，希望能够服务更多的用户，将的服务通过API衔接合作伙伴、APP开发者、智能设备生产厂商，实现数据、服务的有限开放，从而服务更多的业务场景，快速形成一个庞大产业链。使企业在不改变现有生产模式的情况下满足用户碎片化且日益膨胀的需求。

1.API带来的机遇

API开放带来了前所未有的商机，在国际上已经非常流行，已经还有很多公司通过API获得了巨大的成功。然而在国内，除大的互联网公司之外，其他企业API化水平还比较低，这是一个现状也是机遇，因为API化的程度将会成为一个企业的衡量标准。

另外，很多大公司通过API将能力和数据的开放。如果您是一个传统行业的企业，您希望向互联网+转型；或者您是一个创业公司，你需要更多的创业素材，那么恭喜您。这可以提供了足够的素材，并降低了技术门槛。



2. API衍生新的商业模式

创新是企业发展的基石，但在实现企业价值时，不是只要有高新的技术就可以让企业快速发展。与此同时，还要有好的商业模式。

商业模式，是利益相关者的交易模型。所有企业在发展过程中需要不断思考：企业的利益相关者是谁？利益相关者之间有什么价值、资源可以进行交换，如何构建共赢模式？需要通过何种手段来实现？

API可以实现企业间资源的快速交换，企业的不同阶段，可以开放不同类型的API，使用不同的运作方式，根据实际情况选择合适的商业模式。API是企业的增值产品，按着不同的业务目标，可以将API商业模式分为以下4种：

1）将API作为一种商品

企业在发展过程中，会积累一定的业务能力（功能）或者沉淀一些有价值的数据。在不涉及企业机密的前提下，将能力或者数据API化，有偿提供给其他企业使用，增加企业的营收。工具或者服务类的API多使用此种方式，如：天气服务、图像识别、人脸识别、短信服务等。

此商业模式的API，需要能够有良好的API运营经验，需要有合理的收费机制。当然也可以借助第三方平台来实现，阿里云的API市场是个不错的选择。

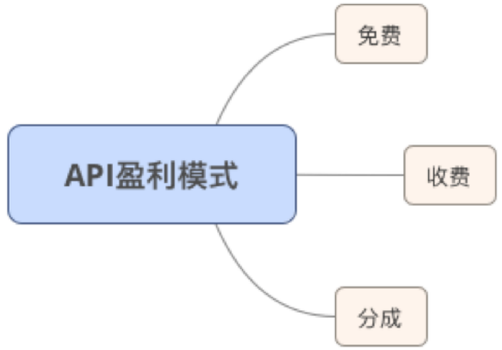
2）通过API延伸产品服务

企业在发展过程中，期望自己的产品能够延展到各个领域，然而受限于成本，无法快速大规模式推广。这时可以将功能API化，让开发者或者合作伙伴在API基础上实现创新，从而更具广度和深度的服务客户，提升产品竞争力。

在这种商业模式下，API一般是免费调用的。企业需要用有价值的API来辅助用户创新，从中获得间接的收入。一般电商、资源类型API使用这种商业模式，如：淘宝、-Salesforce.com、Workday等。

3）通过API来发展合作伙伴，打通企业上、下游，推广产品

开放此类型的API，目的是希望通过API和合作伙伴建立更佳紧密的联系，发展大规模的销售渠道获取更多的收入。比如：一个电商企业，会期望通过API让合作伙伴售卖他的产品，就像在售卖合作伙伴自己的产品一样。企业出售了商品，而合作伙伴也因为产品丰富而获得了更大的影响力，达到双赢的目的。



4）通过API给合作伙伴提供增值服务

通过API，让开发者享受增值服务，来强化产品使之快速发展。如：YouTube、Facebook、Foursquare，都期望能低成本具有丰富、优质的视频资源。而如何鼓励用户上传是一个很大的难题，最终他们找到一个解决方案：让用户在上传视频的同时，可以用API在视频播放时植入广告。从而大大的提升了用户上传视频的积极性，且为了保证广告能够有好的效果，视频质量也大大提升。从而用户得到了相应的广告收入，视频网站也收获了大量的优质视频。

3. API盈利模式

盈利模式没有好坏之分，在企业的不同阶段也需要适实的调整不同的盈利模式。需要企业在开放API就要想清楚企业目前所需要的是什么，开放API的目的是什么，应该如何运营，根据设定的业务目标来确认API的盈利模式。

1）免费模式

API免费调用。免费是指API免费调用，而并非没有盈利模式。提供免费API的企业主要意义是在于，可以扩大企业影响力或者吸引更多的开发者在API基础上进行创新，从而间接的服务更多用户，并从中活的相应的利益。使用此模式的企业，一般有以下几种情况：

a）企业为多元化发展产品、扩大影响力、深度服务最终客户，提供API供开发者/合作伙伴免费使用。

b）为了扩充产品的使用场景，期望合作伙伴或者开发者，在API基础上提供创新，以满足更多用户的需求，从而服务更多的用户群体。

c）为了吸引更多的开发，将API设置多个收费维度，基础功能API免费，高级功能收费，并引导免费用户向收费用户转化。

免费的API容易吸引更多的用户，但也会因为免费会引来一些恶意用户来爬库，造成严重的数据风险。所以开放免费类的API一定要对API开放的范围与调用频率做相应控制，预防核心数据的泄漏。

2）按调用次数或频率收费

一般试用于公开API，是企业在发展过程中形成的有商业价值的功能或者数据。开放出来供其他用户有偿使用，从而增加企业的应收。

大部分按调用次数和调用频率来进行收费。比如：付费100元，可调用1000次，允许调用频率为1次/分钟；付费1000元，可调用1万次，允许的调用频率为1次/秒。

此类的API一定要有商业价值，否则开放出来也无人购买。最好能够结合一定的业务场景，比如：通过你的这一套服务就能实现完整的支付，或者视频编码、解码等，这样对用户会有更大的心引力。

另外，稳定是系统发展的根本，用户在选择服务的同是也会关注您服务的稳定性。您的API一旦打算商用，一定要能够提供稳定的服务，否则您有可能会面临赔偿等问题。

3）利润分成模式

API有限的开放给合作伙伴，并鼓励合作伙伴使用API售卖产品，并将产品利润按一定比例分配给合作伙伴。适用于资源类型企业，如：发展分销、代理模式。提供此类API，一定要能形成一个业务闭环，来保证能够实现一个完整业务。

4.使用第三方API

API经济中企业不仅仅是API提供者，还是API的消费者。企业还可以选用成熟、稳定的第三方API，来完善系统功能。这样不仅可以减少自身系统的代码、加快开发进度，且让开发人员有更多的时间处理自身领域的问题，这要比重复构建别人已经成熟的功能有价值。如：支付、天气、图像识别等等，其他公司已经提供了标准的服务，并不需要我们再耗费人力物力重新开发一遍，并且我们耗时耗力研发出来的也不一定有他人开发的稳定。

5.API的挑战

然而API化的战役是具有挑战性的，并不是所有的企业在API化的路上都能够顺风顺水，反而引来新的麻烦，阻碍了企业的发展。

1）首先是API设计，需要设计者合理的抽象概念、考虑用户场景，设计套一功能套完备、简单易用、可扩展的API，要仔细思考API化的过程，有序发展API经济。 否则用户将因为使用困难而放弃使用。

2）API管理成本上升，在开放API后需要实时的知道API的运行情况、健康度以及用户的调用情况，以便企业后续制定运营和运维策略。其次，要让用户能够了解API的调用方法，我们需要编写API使用文档，甚至SDK，最大的难题是不能随着API的迭代而实时更新。需要有一套完整的管理方法。

3）API安全成本上升，攻击、请求劫持、竞争对手爬库等，让企业措手不及。

4）很多情况下企业无法预知API的流量高峰，而突来的流量会严重影响系统的稳定性。所以，一个好的流量控制，来保护业务系统、实现业务分级和用户分级也是必要的。

阿里云API网关提供了一套此类问题的解决方案，各位可以借鉴。

6.总结

总之，API经济提出了一种新的商业模式，可以辅助企业以低成本快速响应市场需求，建立企业生态，促使跨产业链的企业能力整合而创新出新的经济形式。国内API化程度并不成熟，这也是给我们留下的机会。今后，API开放程度将会成为衡量企业竞争力的核心指标。

HTTPS Solutions

全球HTTPS时代已来，你跟上了吗?

经伦
阿里云安全专家

互联网发展20多年，大家都习惯了在浏览器地址里输入HTTP格式的网址。但前两年，HTTPS逐渐取代HTTP，成为传输协议界的“新宠”。早在2014年，由网际网路安全研究组织Internet Security Research Group(ISRG)负责营运的 “Let’s Encrypt”项目就成立了，意在推动全球网站的全面HTTPS化；今年6月，苹果也要求所有IOS Apps在2016年底全部使用HTTPS；11月，Google还宣布，将在明年1月开始，对任何没有妥善加密的网站，竖起“不安全”的小红旗。

去年，淘宝、天猫也启动了规模巨大的数据“迁徙”，目标就是将百万计的页面从HTTP切换到HTTPS，实现互联网加密、可信访问。

更安全、更可信，是HTTP后面这个“S”最大的意义。HTTPS在HTTP的基础上加入了SSL/TLS协议，依靠SSL证书来验证服务器的身份，并为客户端和服务器端之间建立“SSL加密通道”，确保用户数据在传输过程中处于加密状态，同时防止服务器被钓鱼网站假冒。

HTTP为什么过时了?

很多网民可能并不明白，为什么自己的访问行为和隐私数据会被人知道，为什么域名没输错，结果却跑到了一个钓鱼网站上?互联网世界暗流涌动，数据泄露、数据篡改、流量劫持、钓鱼攻击等安全事件频发。

而未来的互联网网络链路日趋复杂，加重了安全事件发生。可能在星巴克被隔壁桌坐着的黑客嗅探走了口令，或者被黑了家庭路由器任由电子邮件被窃听，又或者被互联网服务提供商秘密注入了广告。这一切都是由互联网开始之初面向自由互联开放

的HTTP传输协议导致的。

1. HTTP数据在网络中裸奔

HTTP明文协议的缺陷，是导致数据泄露、数据篡改、流量劫持、钓鱼攻击等安全问题的重要原因。HTTP协议无法加密数据，所有通信数据都在网络中明文“裸奔”。通过网络的嗅探设备及一些技术手段，就可还原HTTP报文内容。

- 网页篡改及劫持无处不在

篡改网页推送广告可以谋取商业利益，而窃取用户信息可用于精准推广甚至电信欺诈，以流量劫持、数据贩卖为生的灰色产业链成熟完善。即使是技术强悍的知名互联网企业，在每天数十亿次的数据请求中，都不可避免地会有小部分流量遭到劫持或篡改，更不要提其它的小微网站了。

- 智能手机普及，WIFI接入常态化

WIFI热点的普及和移动网络的加入，放大了数据被劫持、篡改的风险。开篇所说的星巴克事件、家庭路由器事件就是一个很有意思的例子。

- 自由的网络无法验证网站身份

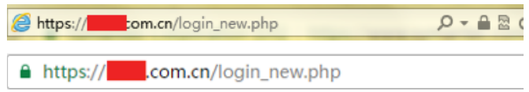
HTTP协议无法验证通信方身份，任何人都可以伪造虚假服务器欺骗用户，实现“钓鱼欺诈”，用户根本无法察觉。

HTTPS，强在哪里?

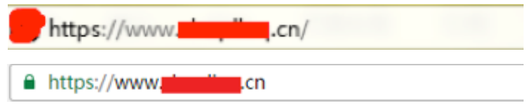
我们可以通过HTTPS化极大的降低上述安全风险。

从上图看，加密从客户端出来就已经是密文数据了，那么你的用户在任何网络链路上接入，即使被监听，黑客截获的数据都是密文数据，无法在现有条件下还原出原始数据信息。

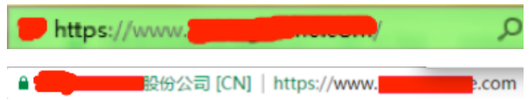
各类证书部署后浏览器呈现效果：



免费SSL数字证书（IE上，Chrome下）



OV SSL数字证书（IE上，Chrome下）

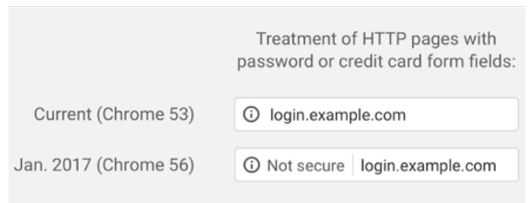


EV SSL数字证书（IE上，Chrome下）

全世界都对HTTPS抛出了橄榄枝

- 浏览器们对HTTP页面亮出红牌

谷歌、火狐等主流浏览器将对HTTP页面提出警告。火狐浏览器将对“使用非HTTPS提交密码”的页面进行警告，给出一个红色的阻止图标；Google Chrome浏览器则计划将所有HTTP网站用“Not secure”显注标识。



图片来源：Googleblog

对于一般用户来讲，如果是这样标识的网站，可能会直接放弃访问。

- 苹果iOS强制开启ATS标准

苹果宣布2017年1月1日起，所有提交到App Store 的App必须强制开启ATS安全标准(App Transport Security)，所有连接必须使用HTTPS加密。包括Android也提出了对HTTPS的要求。

- HTTP/2协议只支持HTTPS

Chrome、火狐、Safari、Opera、IE和Edge都要求使用HTTPS加密连接，才能使用HTTP/2协议。

- HTTPS提升搜索排名

谷歌早在2014年就宣布，将把HTTPS作为影响搜索排名的重要因素，并优先索引HTTPS网页。百度也公告表明，开放收录HTTPS站点，同一个域名的http版和https版为一个站点，优先收录https版。

- 英美强制要求所有政府网站启用HTTPS

美国政府要求所有政府网站都必须在2016年12月31日之前完成全站HTTPS化，截至2016年7月15日，已经有50%政府网站实现全站HTTPS。英国政府要求所有政府网站于2016年10月1日起强制启用全站HTTPS，还计划将service.gov.uk提交至浏览器厂商的HSTS预加载列表，只有通过HTTPS才能访问政府服务网站。

- 超级权限应用禁止使用HTTP连接

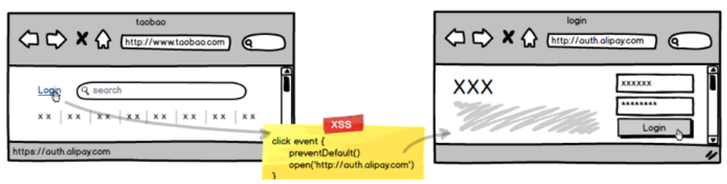
采用不安全连接访问浏览器特定功能，将被谷歌Chrome浏览器禁止访问，例如地理位置应用、应用程序缓存、获取用户媒体等。从谷歌Chrome 50版本开始，地理定位API没有使用HTTPS的web应用，将无法正常使用。

只有部分网页可不够，全站HTTPS才是最佳方案

很多网站所有者认为，只有登录页面和交易页面才需要HTTPS保护，而事实上，全站HTTPS化才是确保所有用户数据安全可靠加密传输的最佳方案。局部部署HTTPS，在HTTP跳转或重定向到HTTPS的过程中，仍然存在受到劫持的风险[1]。

情况一：从HTTP页面跳转访问HTTPS页面

事实上，在 PC 端上网很少有直接进入 HTTPS 网站的。例如：支付宝网站大多是从淘宝跳转过来，如果淘宝使用不安全的 HTTP 协议，通过在淘宝网的页面里注入XSS，屏蔽跳转到 HTTPS 的页面访问，那么用户也就永远无法进入安全站点了。



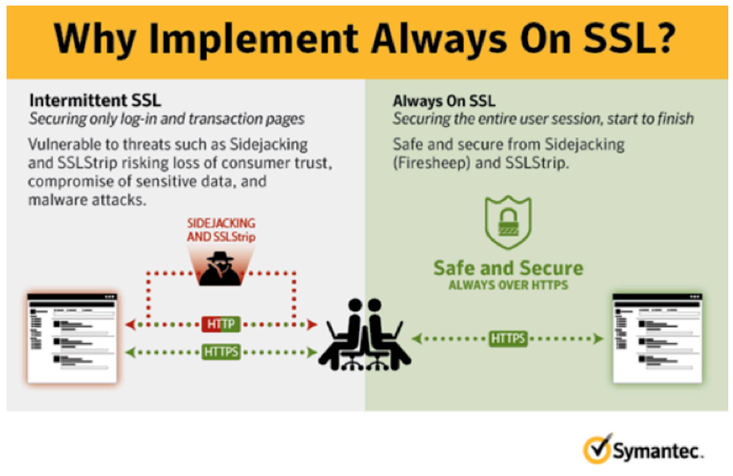
图片来源：EtherDream《安全科普：流量劫持能有多大危害？》

情况二：HTTP页面重定向到HTTPS页面

有一些用户通过输入网址访问网站，他们输入了 www.alipaly.com 就敲回车进入了。然而，浏览器并不知道这是一个 HTTPS 的站点，于是使用默认的 HTTP 去访问。不过这个 HTTP 版的支付宝的确也存在，其唯一功能就是重定向到自己 HTTPS 站点上。劫持流量的中间人一旦发现有重定向到 HTTPS 站点的，于是拦下重定向的命令，自己去获取重定向后的站点内容，然后再回复给用户。于是，用户始终都是在 HTTP 站点上访问，自然就可以无限劫持了。

图片来源：EtherDream《安全科普：流量劫持能有多大危害？》

而全站HTTPS化可以确保用户在访问网站时全程HTTPS加密，不给中间人跳转劫持的机会。国外各大知名网站（PayPal,Twitter,Facebook,Gmail,Hotmail等）都通过Always on SSL（全站https）技术措施来保证用户机密信息和交易安全，防止会话劫持和中间人攻击。[2]



图片来源：EtherDream《安全科普：流量劫持能有多大危害？》

那么问题来了，为什么HTTPS百般好，全世界却还有过一半的网站，还在使用HTTP呢?

首先，很多人还是会觉得HTTPS实施有门槛，这个门槛在于需要权威CA颁发的SSL数字证书。从证书的选择、购买到部署，传统的模式下都会比较耗时耗力。目前，主流CSP都集成了多家证书颁发机构的SSL证书，部署过程也相对更容易一些。因“麻烦”和“门槛”而不HTTPS化的现象，预测也将有所缓解。

第二是性能。HTTPS普遍认为性能消耗要大于HTTP。但事实并非如此，用户可以通过性能优化、把证书部署在SLB或CDN，来解决此问题。举个实际的例子，“双十一”期间，全站HTTPS的淘宝、天猫依然保证了网站和移动端的访问、浏览、交易等操作的顺畅、平滑。通过测试发现，经过优化后的许多页面性能与HTTP持平甚至还有小幅提升，因此HTTPS经过优化之后其实并不慢。

最后是安全意识。相比国内，国外互联网行业的安全意识和技术应用相对成熟，HTTPS部署趋势是由社会、企业、政府共同去推动的。不过，随着国内等保、网络安全、P2P监管措施的普及，HTTPS也有望造福更多网民。

[1]参考来源：EtherDream文章《安全科普：流量劫持能有多大危害？》

[2]参考来源：Symantec文章《Protect the Entire Online User Experience: with Always On SSL》

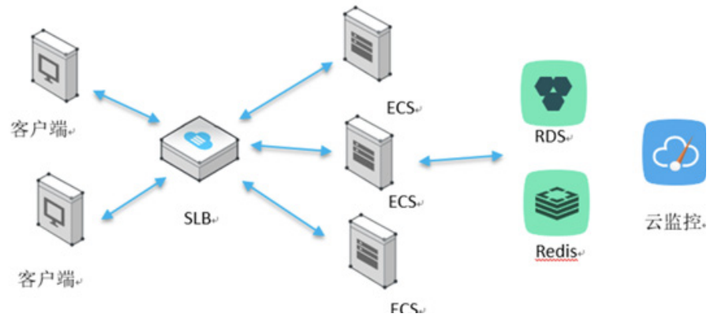
惠每医疗
快速搭建BI数据分析平台

王实
惠每科技 CTO

我所在公司（惠每医疗）的一个主要产品是面向中小诊所的运营SaaS软件，就是诊所可以通过登录网站的方式完成诊所的日常工作，如完成病历，开药以及日常的盘点等。同时产品的推广渠道比较多元化，包括地面销售团队，合作伙伴推广以及常规的搜索引擎推广。作为一个资源相对有限的创业公司，重点关注的就是不同渠道的用户转化以及产品使用情况。

因为自己之前一直在数据算法团队，对于数据收集、处理与应用的流程比较熟悉，加上公司本身规模不大，业务也相对简单，所以在和神策数据沟通后还是觉得自己可以尝试基于阿里云平台快速搭建一个公司自有的数据分析平台。

以阿里云产品体系为例，一个典型的网站架构如下：



除了云服务之外，我们还购买了负载均衡SLB，云数据库RDS，云数据库Redis，云监控等服务，主要是考虑是初期没有专业的运维，而这些产品很好的解决了运维需求。

言归正传，主要还是介绍一下数据分析平台的搭建，下面是一个简单的数据流程架构图。

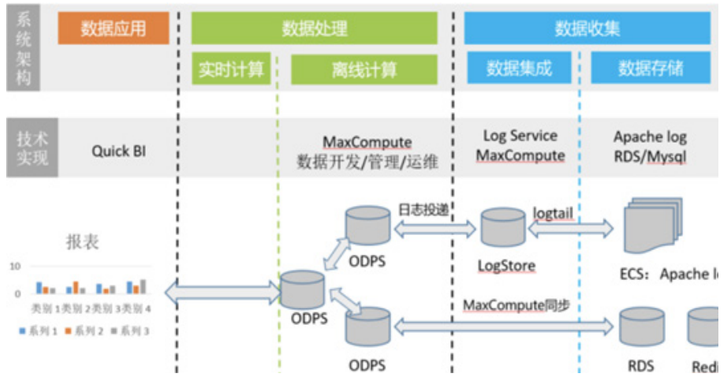
我们以常规的七日留存率为例，来说明数据分析流程构建。

七日留存率的一个简单计算公式如下：

七日留存率 = 七天前注册的当日活跃用户数 / 七天前注册的用户数

根据不同公司业务的运营策略，有时也直接将登录系统的用户等价为活跃用户，这个例子中允许定义活跃用户的关键行为（以诊所软件为例，活跃用户需要开处方、售药等）。

数据收集



通常来说数据包括前端日志和后端结构化数据两部分。

以计算常规的七日留存率为例，活跃用户数（必须满足指定的使用轨迹）来自于前端服务器（Apache或者 Nginx）日志的解析。

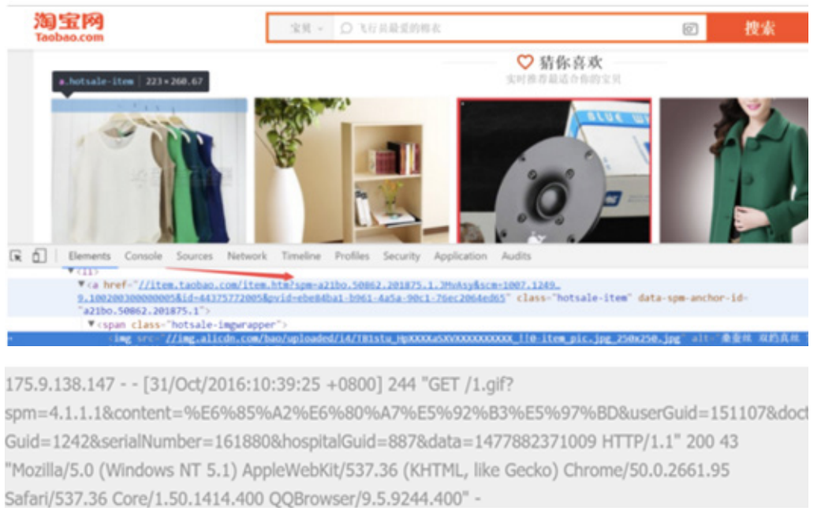
步骤一：采用埋点的技术来记录用户的访问行为：

1. URL埋点

最简单的方法就是投放到不同推广渠道的注册页在url上使用特殊的参数来标记渠道，如channel=xxx，这样解析前端访问日志时就可以通过refer字段来区分渠道。

2. 点击埋点

点击埋点是被广泛使用的技术，简单的做法是在前端页面在发送需要记录的行为时额外发送一个后端请求，如请求一个非常小的图片文件1.gif，文件本身没有任何内容，主要是用于在前端日志中产生一个记录，同样的，在这个请求会带上特殊的参数，如spm=xx.xx.xx.xx，而且参数本身可以采用类似于ip的段位来构建埋点体系，如第一段表示业务，第二段表示产品，第三段表示功能，第四段表示位置等等。如图的淘宝网埋点示例：



最终前端产生的日志（以apache为例）如图所示：

通过URL参数和点击埋点，我们可以就可以监控用户在网站上的使用轨迹，以七日留存率指标为例，我们可以定义活跃用户的行为必须包括哪些关键路径，即spm埋点必须符合哪些规则。

步骤二：自动解析并同步集群机器日志到日志数据库

1. 在所有ECS上安装阿里云logtail工具，Logtail会自动根据设定的时间间隔提交数据。

2. 创建日志数据库

开通日志服务后，在日志服务控制台创建Project（支持多个LogStore），Project下面创建LogStore（支持多个解析配置），创建配置（即解析规则），如后图所示。

这样就完成日志数据的结构化存储，而计算七日留存所需的用户注册数据则已经以结构化的形式存储在RDS Mysql中，下一步需要做的是定时提取LogStore和Mysql中的数据进行运算。

数据处理

云端的数据处理就好像Evernote和有道云笔记一样，可以非常方便的实现在线多人协作。阿里大数据计算平台好像也是今年9月份左右开始对外公测，之前的名

称叫做ODPS，后来改名叫做大数据计算服务（MaxCompute），虽然对于我来说都一样拗口难懂（后面我就用ODPS来统一代指这个服务）……

现在数据处理这块，阿里云其实是单独开个了产品线，命名为阿里云数加平台，对应的服务入口并没有集成到缺省的阿里云控制台目录菜单，在控制台形形色色的产品命名中并不容易定位，所以，真正用起来只能收藏夹管理入口了……

吐槽结束，数据处理包括数据同步、数据运算和运维两个核心部分， 数据同步将LogStore和RDS Mysql数据同步到ODPS， 数据运算和运维则基于ODPS实现多人协作开发数据处理任务并进行上线管理。

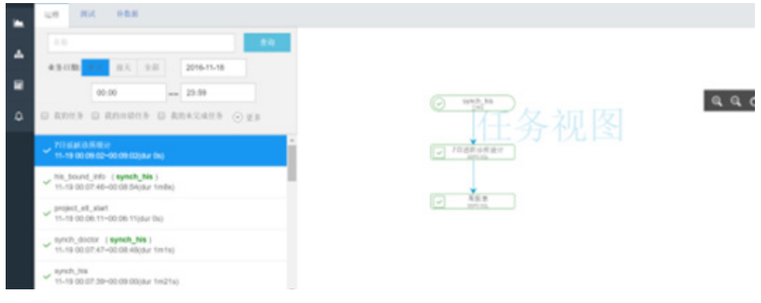
进入阿里云数加控制台后，进入数据开发目录，然后创建Project，创建后通过右边的Project数据开发链接进入了一个在线的IDE（有点像云笔记），就可以开始干活了。

步骤一：同步LogStore和RDS Mysql数据到ODPS数据库表

1. LogStore的数据同步

基本做法就是在IDE的数据管理模块中创建和LogStore相同结构的ODPS表，然后回到日志服务控制台相应的LogStore配置管理中配置一个投递任务，这样就可以自动同步啦。（非常有用的是，在ODPS表中可配置一个时间分区，按天同步日志，这样方便的实现后面的按天调度和计算任务，也可以提升查询效率）。

2. RDS Mysql的数据同步



这个也需要在ODPS中创建一个对应的表（字段可以比原始表少，即可以只同步部分数据），然后再IDE中配置一个同步任务就好了：

步骤二：基于ODPS数据库表的定时任务开发

步骤一基本上完成了数据在ODPS平台上的准备，如每日的用户行为数据和注册数据，下面所需的就是开发定时计算任务了（如计算每天的七日留存指标），下图可以看出，可以使用SQL或Shell脚本开发简单的任务，也可以开发复杂的MapReduce任务，甚至是机器学习任务，也可以用拖拽的方式配置任务的执行顺序。

我们使用SQL任务就可以计算出每日的活跃用户、注册用户以及留存。下图是开发SQL任务的界面，右边可以配置任务的执行周期和依赖，同时也支持多人编辑同一个任务。

步骤三：数据计算任务的运维

在完成开发和测试后，可以通过IDE将任务发布到线上，如图9所示，比较方便的是，运维工具支持补数据，譬如在搭建这个数据流程之前，我们的日志和数据已经积累了数月，我可以补运行任务，从而得到之

前数月的统计指标。

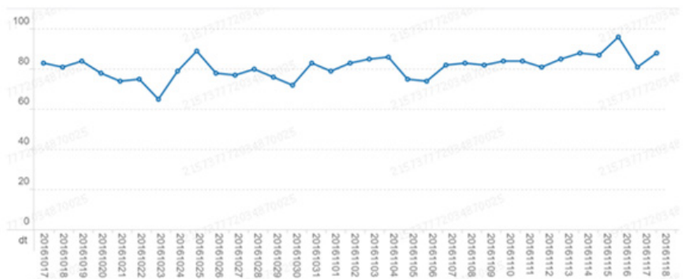
个人觉得大数据计算平台是阿里云数加较为独特的产品，可以实现稳定的数据计算和管理。

数据处理

理论上经过自行设计的数据计算和运维后，企业可根据自己的业务搭建出高度个性

化的应用。这里仍然以初创企业广泛使用的BI报表为例，看如何实现七日留存的数据报表应用。下图是阿里云数加控制台中的BI报表制作示意图。

我们主要使用MaxCompute也就是ODPS计算任务产生的七日留存率数据表作为报表数据源，使用图形化的操作工具即可实现七日流程率的展示，这种乐高积木式的操作



比较简单，就不多啰嗦了，下面是一个完成后的报表图表，官网说可以支持以接口方式将制作的报表嵌入到第三方软件。

基本上，通过基于日志服务的数据采集、基于ODPS的数据计算和运维、基于Quick BI的报表制作，小规模的初创工具可以在2天左右快速的搭建完一个适合业务的、可以扩展的数据分析平台，当然，这一切的前提是你的BOSS能知道数据分析的价值。

企业简介

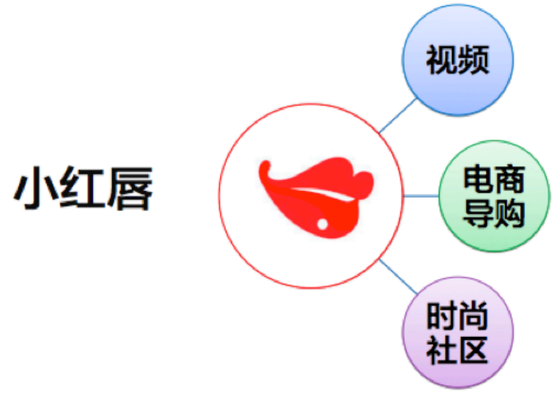
北京惠每科技有限公司为惠每医疗管理咨询（北京）有限公司全资子公司，主要产品包括全科医生的全程临床辅助诊疗工具，包括智能问诊和分诊系统，可以灵活对接HIS和EMR系统，嵌入医疗健康APP和护士工作站，客户包括连锁诊所机构，地方卫计委，一体机厂商和互联网医疗公司。

美妆视频小红唇
如何打开大数据之门

王洋
小红唇 CTO

阿里云数加平台为社区电商小红唇打开了大数据之门，从数据同步→数据加工（数仓搭建+工作流定义+定时调度配置）→BI报表展现，小红唇只花了1天就完成了全链路的自动化报表展现，解决了数据运营难题。那么它究竟是怎么从传统技术发展瓶颈中突破的呢？

小红唇是美妆类的短视频社区电商，各种快速上线的新功能和线上线下的营销活动的效果数据，对于产品计划和公司策略有着决定性的指导意义。但随着业务的快速发展，小红唇面临着如下瓶颈：如何从纷繁的日志、业务数据中提取出有价值的信息，并通过产品数据来指导每一步的运营决策，是小红唇快速发展中亟待解决的问题。



“在使用数加之前，我们采用报表自开发的模式，随着业务的不断拓展，BI的需求越来越多强烈。”小红唇技术负责人王洋说，报表开发代码量越来越大、也越来越复杂，维护十分吃力，弊端也越来越明显。

这是因为小红唇的应用服务器主要是由PHP和Node两种语言开发，由于报表需求分散，缺少很好的规划，最后PHP和Node都各自实现了图表绘制、Excel导出等等底层的功能组件，但结果却是重复建设非常耗费人力、可复用性差等。

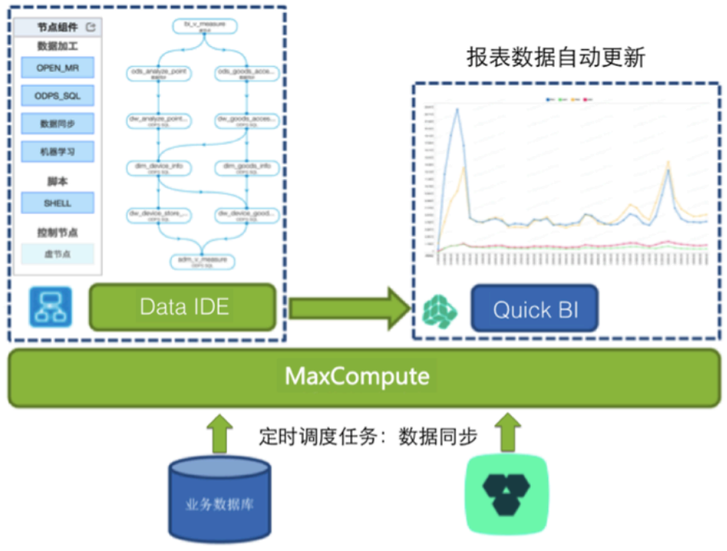
其次，数据库本身的选型不够清晰，MySQL、Postgres、Mongo、Redis都有。一张报表的数据散落在多种数据存储上，报表对应的程序也是异常复杂，需要从

多个数据源获取。同时，由于没有清晰的数据仓库设计，各种数据表混杂在一起，导致报表背后的查询语句异常复杂，动辄就有几百行的SQL。

那这家美妆类短视频社区电商是如何解决这些瓶颈的呢？小红唇技术负责人王洋说，数加平台大数据开发及Quick BI为他们打开了大数据的大门。

“本着不侵入现有系统的原则，我们使用阿里开源的离线数据同步工具DataX把大量分散的日志数据同步到MaxCompute（原ODPS）中。通过简单的前缀(ods_、dw_、dim_ …)，完成对于数仓表和数据挖掘各阶段的表的区分，加上其自身海量数据存储和强大的基于SQL的离线处理能力，使得大数据的数据挖掘变得简单。”

王洋指出，他们所有的数据开发都在Data IDE（ODPS在数加上的Web控制台）中完成。这是一站式的开发维护环境，尤其是自定义的可视化工作流，使得ETL计算节点、报表生成过程一目了然；定时调度任务，让整个过程全自动化，使后期的报表维护变得十分便捷。“这是我们在产出BI报表过程中最大的亮点。”他说。



他还表示，本身的Quick BI上手也非常容易，数据表准备好后，几分钟就产出了报表，不仅能够访问到项目中的所有离线表、数据集中的各种产品表，还能使用自定义SQL再加工形成数据集，通过字段构建关联模型。丰富的图表控件使业务有多种展现方式，也方便决策者能够通过仪表盘和丰富的UI元素从多种视角审视业务状况作出决策。

架构搭建也非常快，从数据同步→数据加工（数仓搭建+工作流定义+定时调度配置）→报表展现，小红唇只花了一天就完成。没错，只花了1天，就完成了整个数据仓库的搭建和全链路自动化的报表展现。有了数仓的基础，后续再新增报表，只花1小时甚至几分钟，就可以产出报表。

“这使得BI报表的开发变得十分高效。”小红唇技术负责人王洋说，“而在平常，我们每新增一张报表都需要花费至少1人周的时间，更别提前期已经花费了至少半年时间来做的底层报表组件积累。”

深入地去回顾，小红唇团队发现：跟以前的报表系统比，他们的数据决策能力也有了很大的提升。“一个原因是由于阿里云Quick BI仪表盘方便的将各种报表汇聚到一起，很容易看出一次商业活动或者一次推广对整个产品各个方面的影响，提高了整个团队整体的分析和决策能力；另外个原因是有了基于时间维度划分，我们也能更方便的结合短期和中长期的数据变化趋势，更灵活更敏锐的采取相应的应对策略。”

从数据采集，到开发，到最终应用于自身业务的完整闭环。极大的提高了小红唇大数据应用的开发到上线的生产效率，弥补了这家初创公司的技术短板，让他们能够更加专注于业务上的创造。“随着更深入的使用阿里云BI产品线，相信还会给我们带来更多的惊喜。”王洋期待到。

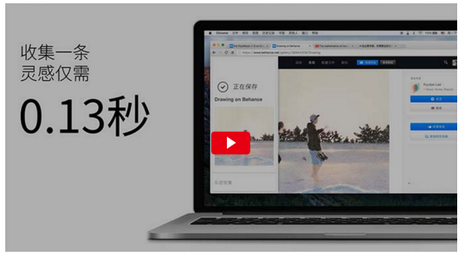
方片收集
碎片化知识收集工具

随着信息技术的飞速发展，我们已经无法避免的进入了碎片化信息时代。信息碎片化是指通过网络接触到的快餐式的、条目式的、海量信息。从早上起床第一次刷朋友圈，工作时用搜索引擎查找资料浏览公众号文章，到晚上入睡前看最后一条微博更新，我们每天都会接触到大量的碎片化文字、图片、视频、链接等，如何从纷乱的碎片化信息中挑选和收集对自己长期有用的信息是很多人都会遇到的问题。

方片收集（原OK记）以做全星球最快的碎片化知识收集工具为愿景，致力于解决由碎片化阅读来带来的各种问题。2015年，方片收集（原OK记）作为阿里巴巴创新中心长沙站的明星孵化团队，摘得阿里云创客+“诸神之战”创客大赛全国总冠军。今年八月，方片收集登榜付费工具榜第一名，子产品方片记事一上架便获得App Store(苹果商店)官方推荐并登上了精品推荐首页头条，以及八月最佳APP。

直击痛点，打造简洁、优雅、高效收集工具

以方片收集PC端主要功能为例，下载方片收集插件（目前已支持火狐、Chrome、360、猎豹、QQ、百度、UC六大浏览器）将所需图片、文字、网址、视频向左拖拽，收集完成，最快收集速度达到了0.13秒。方片收集还提供全库



搜索功能，包括标题、文本、图片等都可以快速检索。

方片收集手机端产品功能则更加高端。很多人都会遇到这样一个情景：在微信上看文章，遇到干货文章只能整个收藏，想摘取其中某些好词好句过程异常繁琐，更不要说边看文章边整理再将笔记同步到电脑里了。方片收集移动端解决了这个问题。

在通知中心添加“方片已收集”（安卓不用这个步骤），选择要摘录的文字，点击复制，拉下通知中心，显示“收集完成”后，再回到文章阅读界面，继续阅读。看完文章回到APP，点击右下方类似抽屉的图标，拉出历史剪贴板，笔记自然生成，链接也会一并保存。

而方片记事，更像是一个笔记软件。打开方片记事APP，整个界面设计极简而考究。点击中间笔形按钮，即可开始码字记录。长按笔形按钮，即可开始说话录音，结束后自动生成一条包含语音和文字识别的笔记。点击笔形按钮同时向左滑即可

扫描二维码，向右滑即可拍照。笔记可以同步到云端在浏览器上阅读，还可以将整理的笔记通过微信微博一键分享。

相比于市面上现有的收集类工具，方片收集有着与众不同的优势：

1）公开化，鼓励知识共享，有社交基因。方片默认收集资料是公开的，这种公开的基因决定了方片在未来有先天性的社交运营优势。我们经常会遗忘自己收藏的文章、图片，而这些收藏对其他有相同兴趣的人来说却是珍贵的。基于这些精品信息的沉淀，在相同兴趣用户之间做社交格外有优势。

2）以收集、摘录为目标，定位小众。在这样一个碎片化信息时代，沉下心来看一本书很难，写一篇文章更难，更何况并不是每个人都会写作，这也是博客会陨落而微博火热的原因，但140字有时也很难，对大部分人来说，更多的动作是mark，转。方片鼓励的是筛选、摘录和收藏，用印象笔记的可能更多的是作家，是文字的生产者，方片的用户可能更多是编辑、设计师、插画师等。

3）易用性强，用户体验更优化。跟很多创业者不同，因为是设计师出生，方片在设计上和功能上特别追求简洁、效率、优雅。

腾云驾雾，借阿里云、创客+之力来创业

对于创业，方片收集的创始人田飞这样说：“并不是为了创业而创业，是因为想创造。大概每个学设计的人都有创造的愿望，创造一款工具、一种方式或者一个社区，是设计师的本能。以往我们做工业设计，只能改变一小部分人，而如果能做出一款好的互联网产品，就可以让更多人的生活和工作因此而改变，这就像一个魔法”。

田飞09年硕士毕业于湖南大学设计艺术学院，曾长期为诺基亚，吉利汽车等企业提供新媒体设计服务，之后因为家庭缘故回到湖南，在湖南工业大学教书，是学校交互设计学科的带头人。创办方片收集之前，他曾创办过独立品牌“Sighvos”和书籍设计事务所“造书房”，但做产品，特别是好的互联网产品的梦想一直未变。2014年下半年，田飞决定去做真正的互联网产品。从自身需求出发，作为一个重度的碎片化阅读消费者，希望用“最小努力成本”来选择和保留有价值的资料信息，并且可以轻易整合、内化以及分享传播；带着设计师与生俱来的独特而挑剔的眼光，这款工具还必须简洁、优雅、高效。2015年8月，方片收集获得北京飞图创投300万元天使融资。11月，在阿里巴巴云栖大会上，获得阿里集团以及在场所有的投资机构一致好评，夺得阿里云创客+大赛全国总冠军。

对于这次创业田飞感触很深：“虽然之前经历过两次创业，但真的再启动一个创业项目时，依然会有各种坑。相比之下，与之前给客户做设计真的大不相同，要考虑的问题也非常不一样：比如产品方向、钱，比如人力与技术，还有团队的士气与质疑，各种缺口和救火，很黑暗的”。创业最艰难的时候，团队中相继有成员因无法承受压力而离开，有人借口回家洗个澡，然后就再也没回来。田飞对这段创业过程的回顾，其实反应了大部分创业者在创业过程中都会遇到的困境：钱、人、技术、方向。为了解决各种创业困境，田飞将目光锁定在阿里云。

方片收集的所有数据和技术都依托于阿里云平台。将所有数据放在阿里云上，一方面降低了数据存储和研发成本，这对小型创业公司来说是至关重要的。另一方面，大大提升了产品服务效率和数据安全，这是创业者和用户首要关注的问题。说到数据安全，田飞经常骄傲的跟用户说，“我们是阿里巴巴创新中心旗下产品，所有的数据都是备份在阿里云平台的，高容灾备份，无限接近绝对安全”。

使用网页端方片收集，最快收集速度达到了0.13秒，最疯狂的一个用户，晚上11 点多注册，当晚就收集了1000多条资料，这验证了产品的易用性和体验效率，对后台的处理速度却是极大的挑战。依托阿里云平台强大的计算速度，用户感觉操作起来特别简单流畅。同时，方片收集支持与印象笔记同步，所有的信息都存储在云上，实现电脑、手机、各移动端即时同步。抛弃过时的U盘、移动硬盘，信息永远存储在云端，再也不怕丢失。

除了完成收集的动作，用户经常还需要对这些信息进行搜索整理，而这些数据又非常碎片化，搜索起来也很难。“我们不用自己写搜索程序，我们用阿里云的搜索服务，只要把相应接口调入，大大节约了开发成本。另外，方片也在考虑升级图片直接转化文字功能，这需要用到OCR（Optical Character Recognition）技术，但小型创业公司的研发成本根本无法支持这种技术，这更需要云计算提供支持。”除此之外，作为一个信息收集类工具，每时每刻会产生海量图片、视频、网页类存储信息，“你永远不知道用户在什么地方什么时候存储了什么图片视频，这些信息会不会违反法律和道德标准”。阿里云的图片鉴黄服务解决了这个问题。“小微企业很难自己做这种服务，借助阿里云我们来腾云驾雾，大大节约了成本提升了效率。”

云计算，推动创新创业的魔法棒

无论我们是否已经准备好迎接信息碎片化时代，这已经成为一个大趋势，无法回避。关于碎片化阅读目前主要的两种观点：一种认为碎片化信息分散和非结构化使得读者获取信息的程度比较浅，不利于知识积累；另一种观点则认为生活节奏非常快的今天，我们的时间本身就很碎片化，阅读的碎片化才是符合未来发展的。无论从哪个观点出发，方片收集的创办都很有价值，且有必要。“如果说碎片化阅读是有害的，那方片的整理功能可以有助于用户把碎片化的知识结构化，系统化。如果说碎片化阅读是正确的，那我们的探索就是顺势而为。”

在碎片化信息的今天，我们需要简洁、高效、优雅的信息聚合工具，来帮助我们组织信息、管理信息。可以说，是信息技术的飞速发展导致了信息的碎片化，却又是信息技术发展来推进了信息的整合。云计算的产生和发展，使各种信息技术的创造和创新变成可能，更让千千万万像田飞那样怀有创造梦想的创业者圆梦。从创业者到用户，我们每时每刻都感受着云计算带来的生活方式和工作方式的改变，这本身就是一个魔法。

项立
阿里研究院博士后

田飞
方片收集创始人

筱鲜
阿里云高级运营专员

墨迹天气
大数据挖掘洞察个性化需求

北京墨迹风云科技股份有限公司于2010年成立，是一家以“做卓越的天气服务公司”为目标的新兴移动互联网公司，主要开发和运营的“墨迹天气”是一款免费的天气信息查询软件。“墨迹天气”APP目前在全球约有超过5亿人在使用，支持196个国家70多万个城市及地区的天气查询，分钟级、公里级天气预报，实时预报雨雪。提供15天天气预报，5天空气质量预报，实时空气质量及空气质量等级预报，其短时预报功能，可实现未来2小时内，每10分钟一次，预测逐分钟逐公里的天气情况。特殊天气提前发送预警信息，帮助用户更好做出生活决策。在墨迹天气上，每天有超过 5 亿次的天气查询需求和将近20亿次的广告请求，这个数字甚至要大于Twitter 每天发帖量。墨迹天气已经集成了多语言版本，可根据手机系统语言自动适配，用户覆盖包括中国大陆、港澳台，日韩及东南亚、欧美等全球各地用户。

挑战

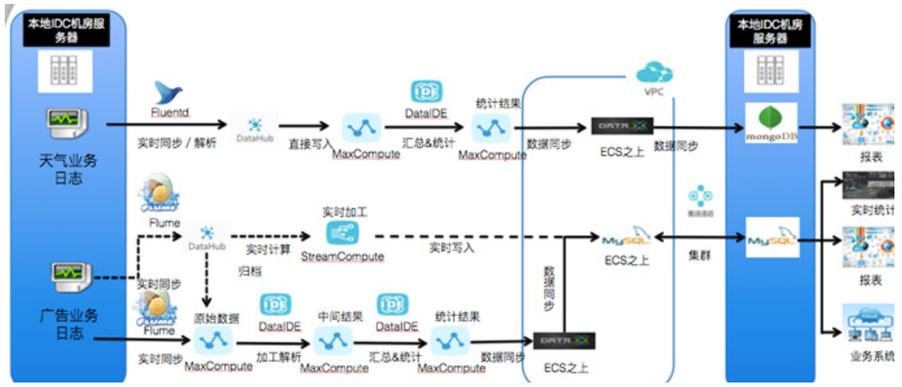
墨迹运营团队每天最关心的是用户正在如何使用墨迹，在他们操作中透露了哪些个性化需求。这些数据全部存储在墨迹的API日志中，对这些数据分析，就变成了运营团队每天的最重要的工作。墨迹天气的API每天产生的日志量大约在2TB左右，主要的日志分析场景是天气查询业务和广告业务。

“用户每天产生的日志量大约在2TB。我们需要将这些海量的数据导入云端，然后分天、分小时的展开数据分析作业，分析结果再导入数据库和报表系统，最终展示在运营人员面前。”墨迹天气运维部经理章汉龙介绍，整个过程中数据量庞大，且计算复杂，这对云平台的大数据能力、生态完整性和开放性提出了很高的要求。

之前墨迹使用国外某云计算服务公司的云服务器存储这些数据，利用Hadoop的MapReducer和Hive对数据进行处理分析，但是存在以下问题：

1.成本：包括存储、计算及大数据处理服务成本对比阿里云成本很高。

2.网络带宽：移动端业务量大，需要大量的网络带宽资源支持，但数据上传也需要占用网络带宽，彼此之间相互干扰造成数据传输不稳定。



解决方案

针对上述情况，墨迹将日志分析业务逐步迁移到阿里云大数据平台-数加平台之上。

新的日志分析架构如图所示：

方案涉及的阿里云数加平台组件有：

阿里云大数据计算服务MaxCompute

大数据计算服务(MaxCompute，原名ODPS)是一种快速、完全托管的TB/PB级数据仓库解决方案。MaxCompute向用户提供了完善的数据导入方案以及多种经典的分布式计算模型，能够更快速的解决用户海量数据计算问题，有效降低企业成本，并保障数据安全。

阿里云大数据开发套件（DataIDE）

大数据开发套件（Data IDE），提供可视化开发界面、离线任务调度运维、快速数据集成、多人协同工作等功能，为您提供一个高效、安全的离线数据开发环境。并且拥有强大的Open API为数据应用开发者提供良好的再创作生态

阿里云流计算（StreamCompute）

阿里云流计算(Aliyun StreamCompute)是运行在阿里云平台上的流式大数据分析平台，提供给用户在云上进行流式数据实时化分析工具。

阿里云流式数据服务（DataHub）

DataHub服务是阿里云提供的流式数据(Streaming Data)服务，它提供流式数据的发布 (Publish)和订阅 (Subscribe)的功能，让您可以轻松构建基于流式数据的分析和应用。

另外，由于每天产生的数据量较大，上传数据会占用带宽，为了不影响业务系统的网络资源，客户开通了阿里云高速通道，用于数据上传。通过此种手段解决了网络带宽的问题。

通过阿里云数加日志分析解决方案，墨迹的业务得到以下提升：

1.充分利用移动端积累下来的海量日志数据。

2.对用户使用情况和广告业务进行大数据分析。

3.利用阿里云数加大数据技术，基于对日志数据的分析，支持运营团队和广告团队优化现有业务。

客户评价

1.迁移到MaxCompute后，流程上做了优化，省掉了编写MR程序的工作，日志数据全部通过SQL进行分析，工作效率提升了5倍以上。

2.存储方面，MaxCompute的表按列压缩存储，更节省存储空间，整体存储和计算的费用比之前省了70%，性能和稳定性也有很大提升。

3.可以借助MaxCompute上的机器学习算法，对数据进行深度挖掘，为用户提供个性化的服务。

4.阿里云MaxCompute提供更为易用、全面的大数据分析功能。MaxCompute可根据业务情况做到计算资源自动弹性伸缩，天然集成存储功能。通过简单的几项配置操作后，即可完成数据上传，同时实现了多种开源软件的对接。

总结

海量的用户数据、精准的气象信息、全方位的生活服务，都是墨迹天气能够稳居天气类手机应用第一的原因。对于未来，无论是从气象数据源、数据处理能力还是结合行业触点的价值等方面都需深度发掘气象价值，墨迹天气将继续前行。

张锴
墨迹天气大数据技术经理

北京墨迹风云科技股份有限公司成立于2010年，以“极致、责任、创新”为理念，以“做卓越的天气服务公司”为目标的新兴移动互联网公司。主要开发和运营的“墨迹天气”是一款免费的天气信息查询软件。墨迹天气目前拥有超过5亿用户，天气日查询次数过亿。

空格APP
云上多场景技术架构实践经验

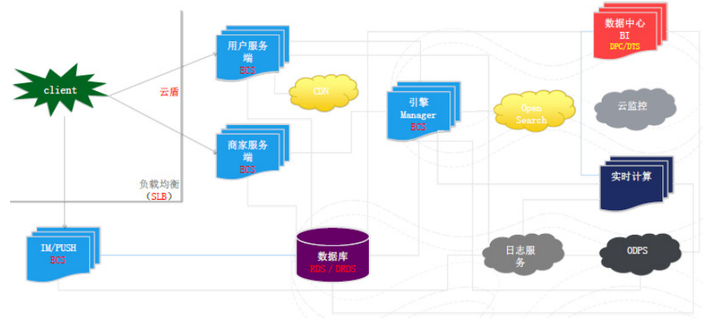
刘博
空格APP技术合伙人

上线仅仅60天就获得1亿A轮融资，同时依靠阿里云只用了两个礼拜就实现了APP上线，平台服务人次超过10万，还得到了中央电视台《焦点访谈》的关注，空格APP是如何做到的？本文从空格APP初始创业阶段谈起，技术合伙人刘博分享了空格利用阿里云服务在搜索、推荐和数据挖掘业务场景下的探索实践。

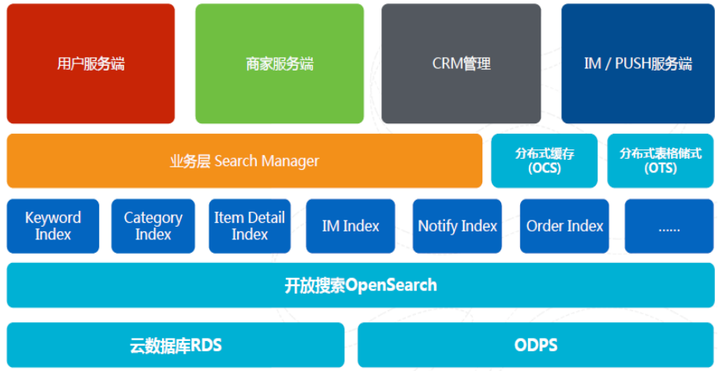
阿里云在空格

在空格初始创业阶段，人员十分缺乏，但依靠着阿里云，空格两周便实现APP上线。空格服务端整体架构包括在线和离线两大部分。在线服务端的前端包括用户服务端集群、商家服务端集群和IM PUSH集群；在线服务端的后端由搜索/推荐引擎集群组成；架构底层的存储采用传统的MySQL数据库。离线服务端由日志搜集系统、离线计算平台、实时流计算平台、监控系统以及数据BI中心组成。

空格整体架构上广泛采用了阿里云产品。在网络层采用了阿里云云盾和负载均衡，利用云盾有效拦截了DDoS等网络攻击，采用负载均衡进行流量智能分配。服务器集群由ECS服务器搭建而成。数据库方面最开始使用的是单机版的RDS，随着数据量的增长，需要进行扩容，通过采用DRDS进行分库分表，很简单地解决了数据库扩展问题。同时采用CDN来存储图片和静态的网页，起到网络加速的效果。在搜索方面采用OpenSearch，快速地搭建搜索引擎，避免了流量激增的情况下运维成本大幅度增加情况，仅需在索引配置上对相应参数进行调高或者调低便可实现扩容。服务端离线部分采用阿里云日志服务，实现在线日志实时收集并同步到离线计算平台，打通了离线到在线再回流到在线的过程。离线计算平台主要采用ODPS，可满足大规模的计算需求。其监控系统采用阿里云云监控产品，对服务器、数据库的关键指标实时监控



报警。数据BI中心底层框架搭建采用的是DPC/DTS数据同步服务。



搜索场景

如图是空格的搜索业务架构。最上层是对搜索有需求的服务端，包括用户服务端、商家服务端、CRM管理系统以及IM/PUSH服务端在内；应用层（业务层）是该业务架构的关键，业务存储方式分为分布式缓存和分布式表格存储；应用场景层包括关键词搜索场景、类目搜索场景、IM消息搜索场景等二十多个搜索场景；底层采用阿里云的开放搜索OpenSearch为支撑，同时OpenSearch无缝衔接云数据库RDS与ODPS，做到数据互通。

阿里云技术方案与普通工程技术方案相比，业务层仍保持一致，但在SearchNode节点上有很大的不同，如果是自建节点，不仅需要考虑到分布式架构和业务间的隔离，还需要考虑离线的大文件与搜索引擎的衔接，同时还需要企业自行开发全量Dump和Build流程，以及建立起索引全量的实时调度，这一切将直接导致运营成本和技术复杂度的增加。更为关键一点，服务索引的在线实时需要更新做到秒级以内，自建搜索引擎实现难度系数很大。但如果采用云技术方案，一切变得很简单。阿里云的OpenSearch将服务器的扩容、全量更新、实时更新、切换调度全部屏蔽掉了，使用者只需简单的配置即可建立新的索引，底层采用RDS和ODPS可实现内部之间数据互通，做到无缝链接。

搜索方面采用阿里云OpenSearch服务，是因为其优势很多。在线方面：具有简单的API接入方式，通过HTTP+Json的服务接口与在线服务对接；内部支持复杂语法和排序规则；同时还有丰富的辅助功能，如查询分析（同义词，停用词，模糊匹配）、下拉提示等。离线方面优势体现在：通过简单配置即可创建索引，无需写代码；字段增删，修改操纵简单；并且与RDS、ODPS无缝衔接，自动管理DUMP数据；可灵活配置索引构建任务。实时更新方面：RDS、ODPS数据源支持实时索引更新，无需构建实时更新系统。运维方面：无需自建分布式集群；无需管理数据备份及冗余；无需考虑扩容。

推荐场景

目前空格推荐业务场景具有以下要求：

1.千人千面，根据用户信息和历史行为进行个性化推荐；

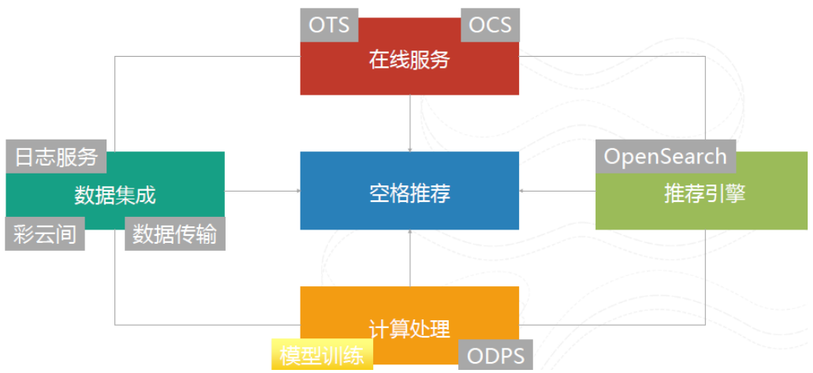
2.大数据，可进行离线和实时计算，并且数据量远超传统的数据库可处理范畴；

3.分布式，可实现分布式服务、分布式计算；

4.弹性扩展，能快速扩容服务能力，应对业务发展；

5.迭代迅速，业务迭代速度极快，两周左右更新一个版本。

针对上述需求，空格结合阿里云服务构建了如上图所示的整体推荐服务架构。数据集成方面，采用阿里云日志服务、采云间和数据传输，将数据传输到离线的ODPS；结合ODPS对数据进行大规模预处理和加工，之后通过自建模型训练进行机器学习，将训练的结果推送到OpenSearch，实现在线检索服务。



数据平台

上面这张幻灯片是空格的数据平台的整体架构，该架构底层通过采用阿里云的ODPS、RDS、日志服务、云监控、采云间一系列服务组合做支撑，搭建了底层的离线计算框架和实时计算框架；中间层的产生的数据主要是系统数据、应用数据、业务数据；应用层针对不同需求产生与之相对应的是运营数据、渠道数据、用户数据、监控数据；这些数据通过API输入给服务层的数据门户、算法Ranking/QR、Open API系统。

上图是空格的门户系统实例，底层数据准备，采用ODPS+采云间计算流程，再到RDS数据同步，实现数据开发和同步；前端展示，基于Bootstrap + Echarts单元框架，开发自适应手机门户页面，PC和手机都可以自如访问；同时对接到钉钉[微应用]，便于使用访问，进一步提高了观察数据的效率。

为什么选择阿里云

总结来看，阿里云服务于自行构建系统相对比，具有相当大的优势。首先数据集成方面：自建系统需自行搭建各类数据服务，打通各类数据管道；而阿里云服务中数据库、分布式存储、在线缓存应有尽有、各服务间数据互通，稳定性可靠性有保障。计算处理方面：自建系统需要自行搭建计算平台，资源缺乏弹性；而阿里云服务仅需开通ODPS，资源弹性、无需担心平台运维，计算框架丰富；引擎方面：自建系统搭建成本高，索引构建复杂，新业务迭代迟缓；阿里云服务开通OpenSearch，自行配置即可完成，开发成本极低。运维方面：自建系统运维复杂，搜索节点历来是故障重灾区；阿里云服务运维简单，无需专职运维团队，安全及稳定性高。成本方面：自建系统搭建耗时、运维成本高；阿里云服务开通服务简单，几乎无运维成本。效率方面：自建系统拖累团队精力、业务迭代缓慢；使用阿里云服务后团队可专注于业务迭代。

阿里云服务极大降低企业系统自建时间和人力成本，使得系统集成高效简单，并且为企业提供健全的配套基础服务，其完整的基础服务保证了系统及数据间互通，高稳定性的保障解放了运维人员的压力，同时阿里云提供了高效的技术支持，短时间内解决企业遇到的难题，促进企业的快速发展。

	自行构建	阿里云服务
数据集成	需自行搭建各类数据服务，打通各类数据管道。	数据库、分布式存储、在线缓存应有尽有、各服务间数据互通，稳定性可靠性有保障。
计算处理	需自行搭建计算平台，资源缺乏弹性。	开通ODPS，资源弹性、无需操心平台运维，计算框架丰富。
引擎	系统搭建成本高，索引构建复杂，新业务迭代迟缓。	开通OpenSearch，自行配置即可完成，开发成本极低。
运维	运维复杂，搜索节点历来是故障重灾区	运维简单，无需专职运维团队，安全及稳定性高
成本	搭建耗时、运维成本高。	开通服务简单，几乎无运维成本。
效率	拖累团队精力、业务迭代缓慢。	团队可专注于业务迭代。

高并发IM系统架构优化实践

少强
阿里云专家

在构建社交IM和朋友圈应用时，一个基本的需求是将用户发送的消息和朋友圈更新及时准确的更新给该用户的好友。为了做到这一点，通常需要为用户发送的每一条消息或者朋友圈更新设置一个序号或者ID，并且保证递增，通过这一机制来确保所有的消息能够按照完整并且以正确的顺序被接收端处理。当消息总量或者消息发送的并发数很大的时候，我们通常选择NoSQL存储产品来存储消息，但常见的NoSQL产品都没有提供自增列的功能，因此通常要借助外部组件来实现消息序号和ID的递增，使得整体的架构更加复杂，也影响了整条链路的延时。

功能介绍

表格存储（Table Store）新推出的主键列递增 功能可以有效地处理上述场景的需求。具体做法为在创建表时，声明主键中的某一列为自增列，在写入一行新数据的时候，应用无需为自增列填入真实值，只需填入一个占位符，表格存储系统在接收到这一行数据后会自动为自增列生成一个值，并且保证在相同的分区键范围内，后生成的值比先生成的值大.

主键列自增功能具有以下几个特性：

1）表格存储独有的系统架构和主键自增列实现方式，可以保证生成的自增列的值唯一，且 严格递增 。

2）目前支持多个主键，第一个主键为分区键，为了数据的均匀分布，不允许设置分区健为自增列。

3）因为分区健不允许设置为自增列，所以主键列自增是 分区键级别的自增 。

4）除了分区键外，其余主键中的任意一个都可以被设置为递增列。

5）对于每张表，目前 只允许设置一个主键列为自增列 。

6）属性列不允许设置为自增列。

7）自增列自动生成的值为 64位的有符号长整型 。

8）自增列功能是 表级别 的，同一个实例下面可以有自增列的表，也可以有非自增列的表。

9）仅支持在创建表的时候设置自增列，对于已存在的表不支持升级为自增列。

介绍了表格存储的主键列自增功能后，下面通过具体的场景介绍下如何使用。

场景

我们继续文章开头的例子，通过构建一个IM聊天工具，演示主键列自增功能的作用和使用方法。

功能

我们要做的IM聊天软件需要支持下列功能：

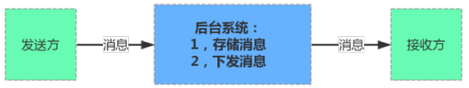
1）支持用户一对一聊天

2）支持用户群组内聊天

3）支持同一个用户的多终端消息同步

现有架构

第一步，确定消息模型



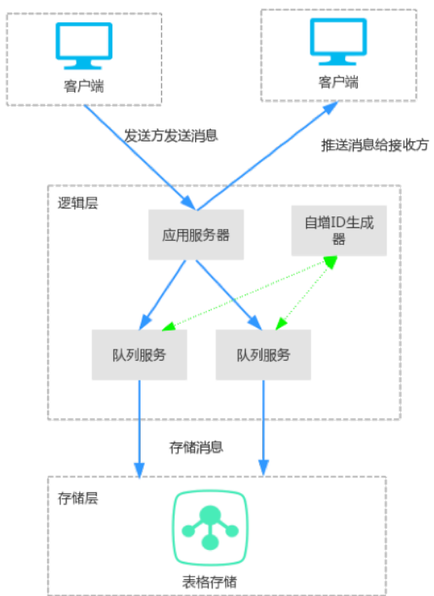
上图展示这一消息模型

发送方发送了一条消息后，消息会被客户端推送给后台系统

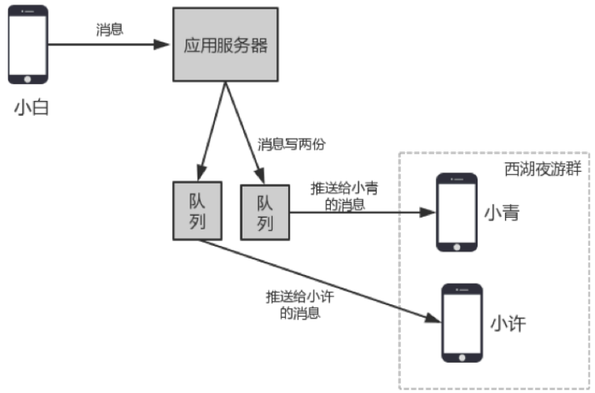
后台系统会先存储消息

存储成功后，会推送消息给接收方的客户端

第二步，确定后台架构



- 后台架构主要分为两部分：逻辑层和存储层。
- 逻辑层包括应用服务器，队列服务和自增ID生成器，是整个后台架构的核心，处理消息的接收、推送、通知，群消息写复制等核心业务逻辑。
- 存储层主要是用来持久化消息数据和其他一些需要持久化的数据。
- 对于一对一聊天，发送方发送消息给应用服务器后，应用服务器将消息存到接收方为主键的表中，同时通知应用服务器中的消息推送服务有新消息了，消息推送服务会将上次推送给接收方的最后一条消息的消息ID作为起始主键，从存储系统中读取之后的所有消息，然后将消息推送给接收方。
- 对于群组内的聊天，逻辑会更加复杂，需要通过异步队列来完成消息的扩散写，也就是说发到群组内的一条消息会给群组内的每个人都存一份。



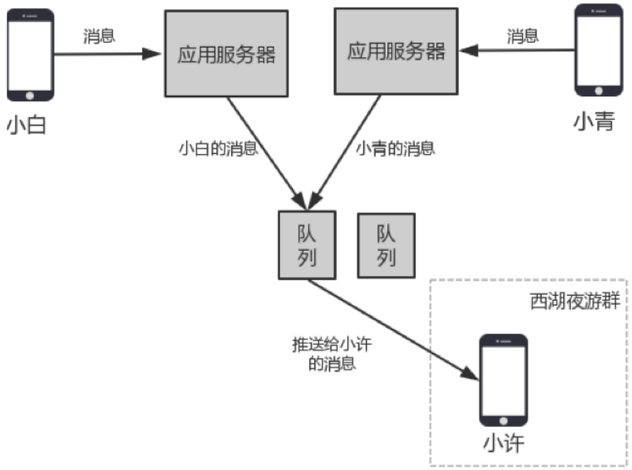
- 上图展示了省略掉存储层后的群消息发送过程。
- 使用扩散写而非扩散读，主要是由于以下两点原因：

1）群组内成员一般都不多，存储成本并不高，而且有压缩，成本更低。

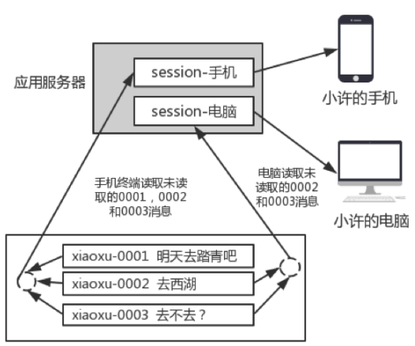
2）消息扩散写到每个人的存储中（收件箱）后，为每个接收方推送消息时，只需要检查自己的收件箱即可，这时候，群聊和单聊的处理逻辑一样，实现简单。

- 发送方发送了一条消息后，这条消息被客户端推送给应用服务器，应用服务器根据接收者的ID，将消息分发给其中一个队列，同一个接收者的消息位于同一个队列中，在队列中，顺序的处理每条消息，先从自增ID生成器中获取一个新的消息ID，然后将这条消息写入表格存储系统。写成功后再写入下一条消息。
- 同一个接收方的消息会尽量在一个队列中，一个队列中可能会有多个接收方的消息。
- 群组内聊天时可能会出现同一个时刻两个用户同时发送了消

息，这两个消息可能会进入不同的应用服务器，但是应用服务器会将同一个接收方的消息发给同一个队列服务，这时候，对于同一个接收方，这两条消息就会处于同一个队列中，如下图：



- 每个队列中的数据串行处理，每次写入表格存储的时候，分配一个新的ID，比之前的ID要大，为了保证消息可以严格递增，避免前一个消息写失败导致无法严格递增的情况出现，需要在写入数据到存储系统的时候，持有一个用户级别的锁，在没有写成功之前，同用户的其他消息不能继续写，以免当前消息写失败后导致乱序，当写成功后，释放这个锁，下一个消息继续。
- 上一步中，如果队列宕机，这些消息需要重新处理，这时候，原有消息就会进入一个新的队列，这时候新的队列需要一个新的消息ID，但要比之前已有的消息ID更大，而这个新队列并不知道之前的最大ID是啥，所以，这里每个队列没法自主创建自增ID，而需要一个全局的自增ID生成器。
- 为了支持多终端，在应用服务器中会为每个终端持有一个session，每个session持有一个当前最新消息的ID，当被通知有新消息时，会去存储系统读取当前消息之后的所有消息，这样就保证了多终端同时在线时，每个终端都可以同步消息，且相互不影响，见下图。



- 在多终端中，如果有部分终端由在线变成了离线，那么应用服务器会将这个终端的session保存到存储系统的另一张表中，当一段时间后，这个终端再次上线时，可以从存储系统中恢复出之前的session，继续为此终端推送之前未读取的消息。

第三步，确定存储系统

存储系统，我们选择了阿里云的 表格存储，主要是因为下列原因：

- 写操作不仅支持 单行写 ，也支持 多行批量写 ，满足大并发写数据需求。
- 支持按 范围读 ，消息多时可翻页。
- 支持 数据生命周期管理 ，对过期数据进行自动清理，节省存储费用
- 表格存储是阿里云已经商业化的云服务， 稳定可靠 。
- 表格存储 价格便宜，对于数据量大的用户还可以以更优惠的价格购买套餐。
- 读写性能优秀，对于聊天消息，延迟基本在毫秒，甚至微妙级别。

第四步，确定表结构

确定的表格存储的表结构如下：

- 表格存储的表结构分为两部分，主键列部分和属性列部分，主键列部分最多支持4个主键，第一个主键为分区健。
- 使用前，需要确定主键列部分的结构，使用过程中不能修改；属性列部分是Schema Free的，用户可以自由定制，每一行数据的属性列部分可以不一样，所以，只需要设计主键列部分的结构。
- 第一个主键是分片键，目的是让数据和请求可以均衡分布，避免热点，由于最终读取消息时是要按照接收方读取，所以这里可以使用接收方ID作为分片键，为了更加均衡，可以使用接收方ID的md5值的部分区域，比如前4个字符。这样就可以将数据均衡分布了。
- 第一个主键只用了部分接收方ID，为了能定位到接收方的消息，需要保存完整的接收方ID，所以，可以将接收方ID作为第二个主键。

主键顺序	主键名称	主键值	说明
1	partition_key	hash(receive_id)前4位	分区键，保证数据均匀分布，可以使用 md5 作为 hash 函数
2	receive_id	receive_id	接收方的用户 ID
3	message_id	message_id	消息 ID

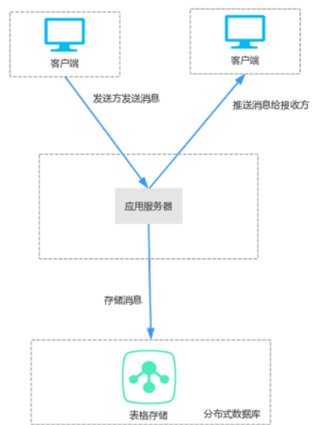
- 第三个主键就可以是消息ID了，由于需要查询最新的消息，这个值需要是单调自增的。
- 属性列可以存消息内容和元数据等。

到此，我们已经设计出了一个完整的聊天系统，虽然这个系统已经可以运行，且能处理大并发，性能也不差，但是还是存在一些挑战。

挑战

- 多个用户在一个队列中，这个队列串行执行，为了保证消息严格递增，这里执行过程中要持有锁，这里就会有一个风险点：如果发送给某个用户的消息量很大，这个用户所在的队列中消息会变多，就有可能堵塞其他用户的消息，导致同队列的其他用户消息出现延迟。
- 当出现重大事件或者特定节假日，聊天信息量大的时候，队列部分需要扩容，否则可能扛不住大压力，导致整体系统延迟增大或者崩溃。

针对上述两个问题，问题2可以通过增加机器的方式解决，但是问题1没法通过增加机器解决，增加机器只能缓解问题，却没法彻底解决。那有没有办法可以彻底解决掉上述两个问题？



新架构

上面两个问题的复杂度主要是由于需要消息严格递增引起的，如果使用了表格存储的主键列自增功能，那么上层的应用层就会简单的多。

使用了表格存储**主键列 自增功能**后的新架构如下：

- 最明显的区别是少了队列服务和自增ID生成器两个组件，架构更加简单。
- 应用服务器接收到消息后，直接将消息写入表格存储，对于主键自增列message_id，在写数据时不需要填确定的值，只需要填充一个特定的占位符即可，这个值会在表格存储系统内部自动生成。
- 新架构中自增操作是在表格存储系统内部处理的，就算多个应用服务器同时给表格存储中的同一个接收方写数据，表格存储内部也能保证这些消息是串行处理，每个消息都有一个独立的消息ID，且严格递增。那么之前的队列服务就不在需要了。这样也就彻底解决了上面的问题1
- 表格存储系统是一个云服务，用户并不需要考虑系统的容量，而且表格存储支持按量付费，这样也就 彻底解决了上面的问题2
- 之前只能有一个队列处理同一个用户的消息，现在可以多个队列并行处理了，就算某些用户的消息量突然变大，也不会立即堵塞其他用户，而是将压力均匀分布给了所有队列。
- 使用主键自增列功能后，应用服务器可以直接写数据到表格存储，不再需要经过队列和获取消息ID， 性能表现会更加优秀。

实现

有了上面的架构图后，现在可以开始实现了，这里选用JAVA SDK，目前4.2.0版本已经支持主键列自增功能。

第一步，建表

按照之前的设计，表结构如下：

第三列PK是message_id，这一列是主键自增列，建表时指定message_id列的属性为AUTO_INCREMENT，且类型为INTEGER。

```
private static void createTable(SyncClient client) {
    TableMeta tableMeta = new TableMeta(“message_table”);

    // 第一列为分区建
    tableMeta.addPrimaryKeyColumn(new PrimaryKeySchema(“partition_key”, PrimaryKeyType.STRING));

    // 第二列为接收方ID
    tableMeta.addPrimaryKeyColumn(new PrimaryKeySchema(“receive_id”, PrimaryKeyType.STRING));

    // 第三列为消息ID，自动自增列，类型为INTEGER，属性为PKO_AUTO_INCREMENT
    tableMeta.addPrimaryKeyColumn(new PrimaryKeySchema(“message_id”, PrimaryKeyType.INTEGER, PrimaryKeyOption.AUTO_INCREMENT));

    int timeToLive = -1; // 永不过期，也可以设置数据有效期，过期了会自动删除
    int maxVersions = 1; // 只保存一个版本，目前支持多版本

    TableOptions tableOptions = new TableOptions(timeToLive, maxVersions);

    CreateTableRequest request = new CreateTableRequest(tableMeta, tableOptions);

    client.createTable(request);
    }
```

通过上述方式就创建了一个第三列PK为自动自增的表。

第二步，写数据

写数据目前支持PutRow和BatchWriteRow两种方式，这两种接口都支持主键列自增功能，写数据时，第三列message_id是主键自增列，这一列不需要填值，只需要填入占位符即可。

```
private static void putRow(SyncClient client, String receive_id) {
    // 构造主键
    PrimaryKeyBuilder primaryKeyBuilder = PrimaryKeyBuilder.createPrimaryKeyBuilder();

    // 第一列的值为 hash(receive_id)前4位
    primaryKeyBuilder.addPrimaryKeyColumn(“partition_key”, PrimaryKeyValue.fromString(hash(receive_id).substring(4)));

    // 第二列的值为接收方ID
    primaryKeyBuilder.addPrimaryKeyColumn(“receive_id”, PrimaryKeyValue.fromString(receive_id));

    // 第三列是消息ID，主键递增列，这个值是TableStore产生的，用户在这里不需要填入真实值，只需要一个占位符：AUTO_INCREMENT即可。
    primaryKeyBuilder.addPrimaryKeyColumn(“message_id”, PrimaryKeyValue.AUTO_INCREMENT);
    PrimaryKey primaryKey = primaryKeyBuilder.build();

    RowPutChange rowPutChange = new RowPutChange(“message_table”, primaryKey);

    //  这里设置返回类型为RT_PK，意思是在返回结果中包含PK列的值。如果不设置ReturnType，默认不返回。
    rowPutChange.setReturnType(ReturnType.RT_PK);

    //加入属性列，消息内容
    rowPutChange.addColumn(new Column(“content”, ColumnValue.fromString(content)));

    //写数据到TableStore
    PutRowResponse response = client.putRow(new PutRowRequest(rowPutChange));

    // 打印出返回的PK列
    Row returnRow = response.getRow();
    if (returnRow != null) {
    System.out.println(“PrimaryKey:” + returnRow.getPrimaryKey().toString());
    }

    // 打印出消耗的CU
    CapacityUnit cu = response.getConsumedCapacity().getCapacityUnit();
    System.out.println(“Read CapacityUnit:” + cu.getReadCapacityUnit());
    System.out.println(“Write CapacityUnit:” + cu.getWriteCapacityUnit());
    }
```



```
private static void getRange(SyncClient client, String receive_id, String lastMessageId) {
    RangeRowQueryCriteria rangeRowQueryCriteria = new RangeRowQueryCriteria(“message_table”);

    // 设置起始主键
    PrimaryKeyBuilder primaryKeyBuilder = PrimaryKeyBuilder.createPrimaryKeyBuilder();

    // 第一列的值为 hash(receive_id)前4位
    primaryKeyBuilder.addPrimaryKeyColumn(“partition_key”, PrimaryKeyValue.fromString(hash(receive_id).substring(4)));

    // 第二列的值为接收方ID
    primaryKeyBuilder.addPrimaryKeyColumn(“receive_id”, PrimaryKeyValue.fromString(receive_id));

    // 第三列的值为消息ID，起始于上一条消息
    primaryKeyBuilder.addPrimaryKeyColumn(“message_id”, PrimaryKeyValue.fromLong(lastMessageId + 1));
    rangeRowQueryCriteria.setInclusiveStartPrimaryKey(primaryKeyBuilder.build());

    // 设置结束主键
    primaryKeyBuilder = PrimaryKeyBuilder.createPrimaryKeyBuilder();

    // 第一列的值为 hash(receive_id)前4位
    primaryKeyBuilder.addPrimaryKeyColumn(“partition_key”, PrimaryKeyValue.fromString(hash(receive_id).substring(4)));

    // 第二列的值为接收方ID
    primaryKeyBuilder.addPrimaryKeyColumn(“receive_id”, PrimaryKeyValue.fromString(receive_id));

    // 第三列的值为消息ID
    primaryKeyBuilder.addPrimaryKeyColumn(“message_id”, PrimaryKeyValue.INF_MAX);
    rangeRowQueryCriteria.setExclusiveEndPrimaryKey(primaryKeyBuilder.build());

    rangeRowQueryCriteria.setMaxVersions(1);

    System.out.println(“GetRange的结果为:”);
    while (true) {
    GetRangeResponse getRangeResponse = client.getRange(new GetRangeRequest(rangeRowQueryCriteria));
    for (Row row : getRangeResponse.getRows()) {
    System.out.println(row);
    }

    // 若nextStartPrimaryKey不为null，则继续读取.
    if (getRangeResponse.getNextStartPrimaryKey() != null) {
    rangeRowQueryCriteria.setInclusiveStartPrimaryKey(getRangeResponse.getNextStartPrimaryKey());
    } else {
    break;
    }
    }
    }
```

第三步，读数据

读消息的时候，需要通过GetRange接口读取最近的消息，message_id这一列PK的起始位置是上一条消息的message_id+1，结束位置是INF_MAX，这样每次都可以读出最新的消息，然后发送给客户端。

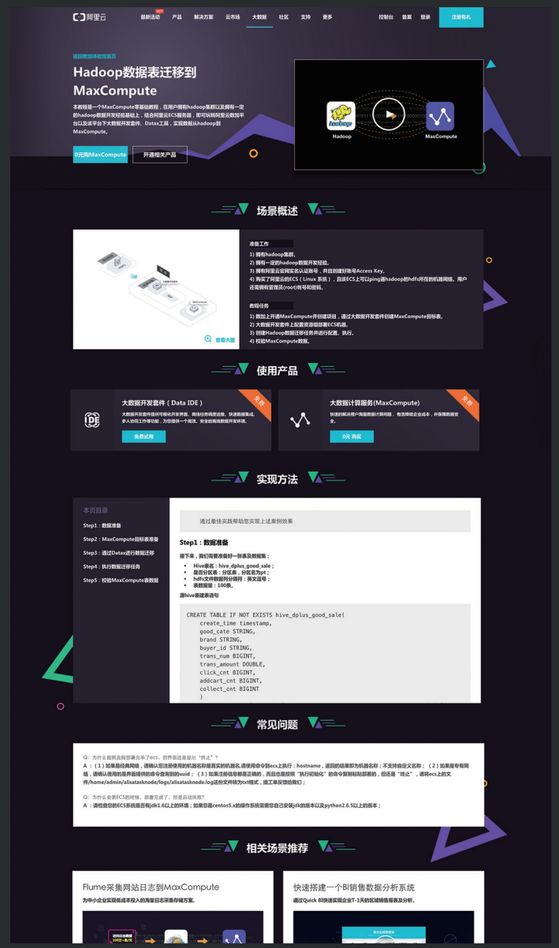
上面演示了表格存储及其主键列自增功能在聊天系统中的应用，在其他场景中也有很大的价值，期待大家一起去探索。

表格存储OTS

DemoCenter

数加大数据体验馆

数加体验馆，是阿里云数加推出的大数据DemoCenter，场景案例涉及气象、金融、新闻媒体、人工智能等大数据重点发展领域，可以帮助企业改善大数据业务实施的敏捷性，如测试、部署周期长，投入成本高、业务投入市场慢等问题。



海量日志数据快速上云

通过海量日志数据上云教程，帮助中小企业实现低成本海量数据存储。

快速搭建一个BI销售数据分析系统

通过海量数据实时在线分析教程，帮助企业轻松自如地完成数据分析、业务数据探查。

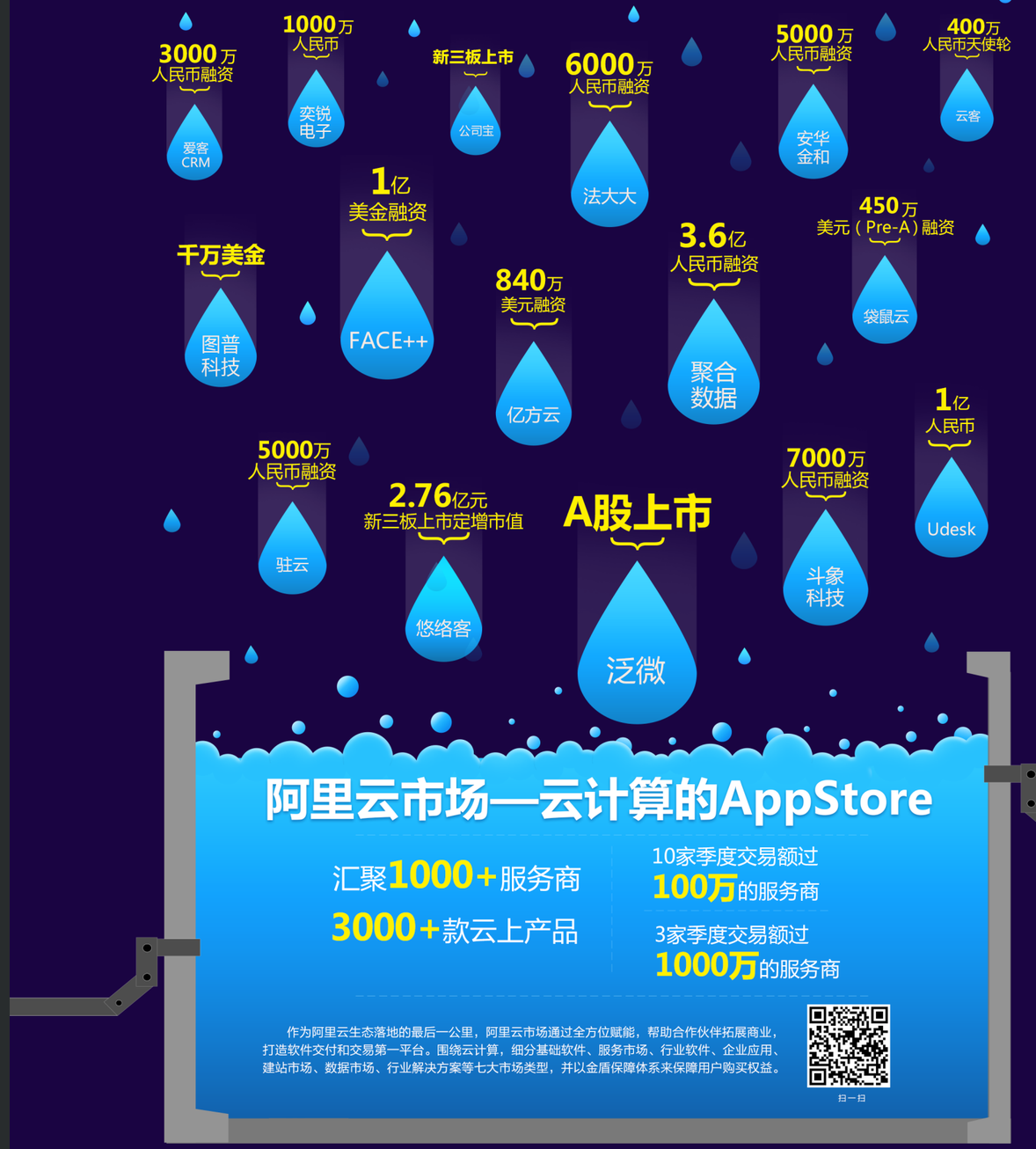
轻松搞定日志实时分析及监控大屏

使用阿里云数加流计算及数据可视化DataV快速制作网站运营数据实时监控大屏。

数加体验馆 更多案例欢迎您

雾霾治理哪家强 净化器市场监测
机器学习帮助您挖掘金融欺诈用户
Hadoop数据表迁移到MaxCompute
机器学习实现海量新闻自动分类

2016阿里云生态
最受资本追捧企业榜



域名解析

优化你的移动互联网络

泠茗
阿里云高级技术专家

域名(Domain Name)，是由一串用点分隔的名字组成的互联网上某台计算机或某组计算机的标识，它的目的是为了方便人们更简单便捷地访问互联网上的服务。在实际的系统实现中，域名通过DNS（Domain Name System）系统转化为服务器的IP地址，以方便机器通过IP进行寻址和通信。上述行为，我们称之为域名解析。

作为一次网络通信最前置的环节，域名解析的重要性不言而喻。在传统的基于浏览器的网站访问场景下，域名解析环节由浏览器内核实现，网站开发者无需关心域名解析的细节。But there are always two sides to every coin，一旦域名解析环节发生异常，开发者面对这样的黑盒架构就会显得束手无策，一个很典型的例子即域名劫持问题，关于这一点我们在后文会有更详细的介绍。

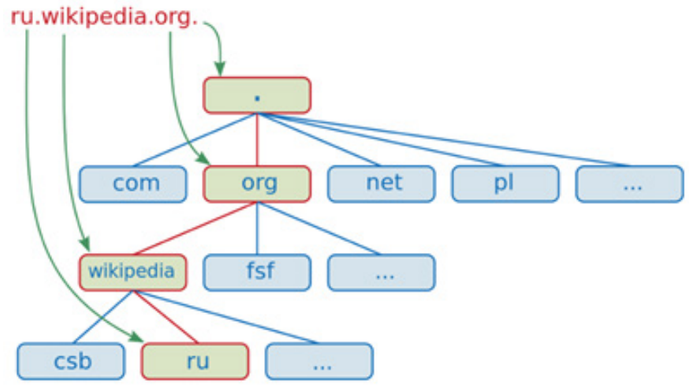
进入移动互联网时代，大量的应用基于C/S架构构建。相较于传统的面向浏览器的Web App，C/S架构的应用赋予了我们非常大的软件定制空间，开发者甚至可以渗透到整个应用的底层网络实现当中，域名解析环节的优化因此变为了可能。本篇文章我们就一起来看一看传统域名解析存在的问题，对应的根源，以及可能的优化方案。

关于域名解析，你应该知道的基本概念

在了解传统域名解析的流程之前，有几个专有名词我们需要了解一下：

1.根域、顶级域、二级域

DNS系统一般采用树状结构进行组织，以ru.wikipedia.org为例，org为顶级域名，wikipedia为二级域名，ru为三级域名，域名树状组织结构如下图所示。



域名

2.权威DNS

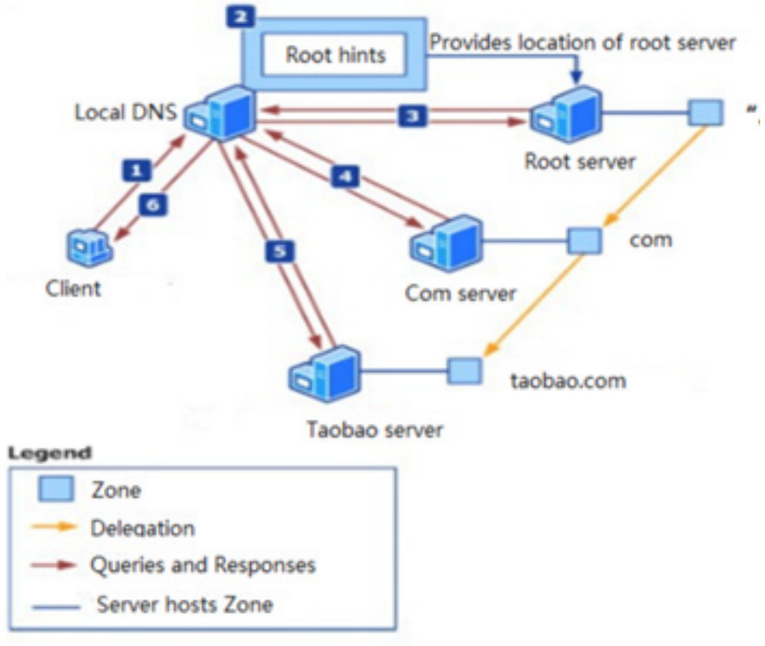
权威DNS即最终决定域名解析结果的服务器，开发者可以在权威DNS上配置、变更、删除具体域名的对应解析结果信息。阿里云云解析即权威DNS服务提供商。

3.递归DNS

递归DNS又称为Local DNS，它没有域名解析结果的决定权，但代理了用户向权威DNS获取域名解析结果的过程。递归DNS上有缓存模块，当目标域名存在缓存解析结果并且TTL未过期时（每个域名都有TTL时间，即有效生存时间，若域名解析结果缓存的时间超过TTL，需要重新向权威DNS获取解析结果），递归DNS会返回缓存结果，否则，递归DNS会一级一级地查询各个层级域名的权威DNS直至获取最终完整域名的解析结果。关于域名解析的具体流程下文会举例说明。

4.公共DNS

公共DNS是递归DNS的一种特例，它是一种全网开放的递归DNS服务，而传统的递归DNS信息一般由运营商分发给用户。一个比较典型的公共DNS即Google的8.8.8.8，我们可以通过在操作系统配置文件中配置公共DNS来代替Local DNS完成域名解析流程。



在实际的使用过程中，我们通常不需要手工指定自己的Local DNS地址。运营商会通过DHCP协议在系统网络初始化阶段将Local DNS地址分配给我们的计算机。当我们需要使用公共DNS服务时，我们就必须手工指定这些服务的地址。以Linux为例，我们可以通过在’/etc/resolv.conf’中添加Local DNS地址项来改变本机Local DNS的地址。

了解了上述域名解析相关的常见术语，我们再来仔细看一看一次域名解析流程具体是如何发生的。

如左图所示，以访问www.taobao.com为例，一次完整的域名解析流程包括：

1）终端向Local DNS发起域名解析请求；

2）Local DNS在获取到域名解析请求后首先从Root hints获取根域名服务器的地址（Root hints包含了互联网DNS根服务器的地址信息）；

3）获取了根域名服务器地址后Local DNS向根域名服务器发起DNS解析请求，根域名服务器返回com顶级域名服务器地址；

4）随后Local DNS向com域名服务器发起解析请求，并得到taobao.com二级域名服务器的地址；

5）Local DNS向taobao.com二级域名服务器发起解析请求，并最终获得了www.taobao.com的IP地址信息；

6）Local DNS将递归查询获得的IP地址信息缓存并返回给客户端；

Local DNS服务器包含缓存模块，在实际域名解析过程中Local DNS服务器会首先查询缓存，缓存命中且解析结果TTL未过期的情况下直接返回，否则才启动递归查询的流程。

传统域名解析面临的问题

了解了域名解析的基本概念和整体流程，我们再一起来探究一下传统域名解析存在的一系列问题。

1.域名劫持

域名劫持一直是困扰许多开发者

的问题之一，其表现即域名A应该返回的DNS解析结果IP1被恶意替换为了IP2，导致A的访问失败或访问了一个不安全的站点。下面我们一起看看几种常见的域名劫持的场景。

一种可能的域名劫持方式即黑客侵入了宽带路由器并对终端用户的Local DNS进行篡改，指向黑客自己伪造的Local DNS，进而通过控制Local DNS的逻辑返回错误的IP信息进行域名劫持。另一方面，由于DNS解析主要是基于UDP协议，除了上述攻击行为外，攻击者还可以监听终端用户的域名解析请求，并在Local DNS返回正确结果之前将伪造的DNS解析响应传递给终端用户，进而控制终端用户的域名访问行为。

上述攻击行为的影响面相对比较有限，另一种我们最常碰到的域名劫持现象是缓存污染。我们知道在接收到域名解析请求时，Local DNS首先会查找缓存，如果缓存命中就会直接返回缓存结果，不再进行递归DNS查询。这时候如果Local DNS针对部分域名的缓存进行更改，比如将缓存结果指向第三方的广告页，就会导致用户的访问请求被引导到这些广告页地址上。

对比第一种攻击，这类缓存污染往往能带来更明显的群体伤害，比如某个省份某个运营商的用户群可能因为该地区Local DNS的缓存污染而导致访问服务异常。这类缓存污染行为往往是间歇性、局部性发生的，没有明显的规律，导致开发者很难对其进行量化、评估、预防。

有的同学可能会问，“我使用了HTTPS，是否就可以避免域名劫持的问题”，答案是否定的。域名解析环节发生在网络加密请求交互之前，试想一下，如果客户端还没有服务端的确切地址信息，我们又如何知道应该和谁进行加密的握手协商与通信呢？

2.调度不精准

除了域名劫持问题，基于传统Local DNS的域名解析还会带来域名调度精准性的问题。对于类似CDN域名访问这类需要按地

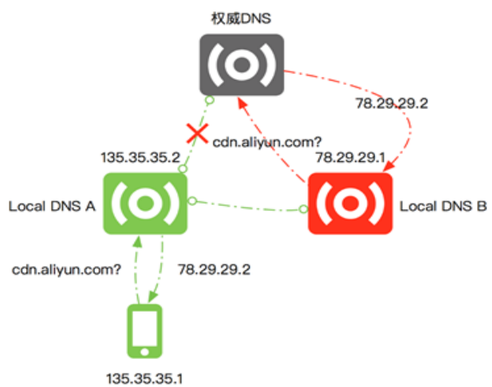
域、运营商进行智能解析调度的场景，精准调度的诉求是十分强烈的。

关于调度不精准的原因，我们主要可以从两个方面来探究一下。第一个常见的问题即解析转发。

部分Local DNS供应商为了降低运营成本，会将请求到自己节点的域名解析请求转发给其他供应商的Local DNS节点，如下图所示。假如用户请求解析一个CDN域名cdn.aliyun.com，用户分配到的Local DNS A为了节省成本，把该次请求转发给了另一运营商的Local DNS B，权威DNS在进行域名解析时会根据Local DNS的IP信息进行智能调度，即权威DNS会根据Local DNS B的IP78.29.29.1进行调度，分配与78.29.29.1相同运营商并且地理位置最近的CDN节点78.29.29.2，然而这个CDN节点对于终端135.35.35.1而言并不是最优的CDN节点，他们分属不同的运营商，并且地理位置上可能相隔很远。这类解析转发行为会严重影响域名解析的精准性并对用户业务访问延迟带来影响。

除了解析转发对调度精准性带来的影响外，Local DNS的布署情况同样影响着域名智能解析的精准性。

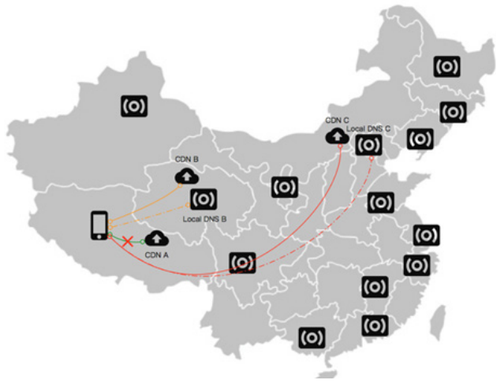
如图所示，部分运营商Local DNS的布点受成本因素制约分布并不均匀，比如在东部地区部署比较密集，但在西部地区部署比较稀疏。这时候当一位西藏的用户准备访问CDN节点时，我们预期他应该会被调度到西藏的CDN节点A上以实现就近接入和访问加速。但由于Local DNS的资源有限，西部地区的终端用户被统一调度到青海的Local DNS B上，这时候权威DNS根据Local DNS B的IP进行CDN域名的智能解析，并将青海的CDN节点B返回给西藏用户，导致用户的网络访问延迟上升。另一种我们实际发现的情况是Local DNS的分配甚至并非遵循就近原则，比如有实际案例显示西藏的用户甚至被分配了北京的Local DNS节点C，



导致西藏的用户在进行CDN资源访问时被调度到了北京的CDN节点C上，类似的由于调度精度的缺失带来的访问体验的影响是非常严重的。

3.解析生效滞后

部分业务场景下开发者对域名解析结果变更的生效时间非常敏感（这部分变更操作是开发者在权威DNS上完成的）

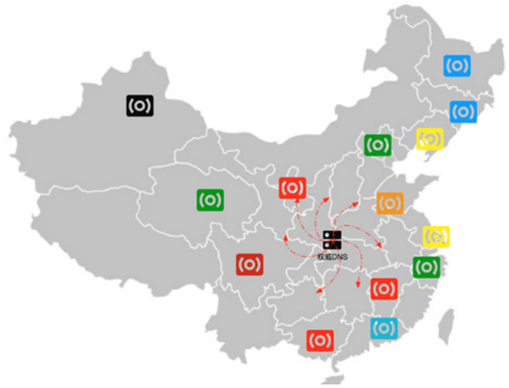


，比如当业务服务器受到攻击时，我们需要最快速地将业务IP切换到另一组集群上，这样的诉求在传统域名解析体系下是无法完成的。

Local DNS的部署是由各个地区的各个运营商独立部署的，因此各个Local DNS的服务质量参差不齐。在对域名解析缓存的处理上，各个独立节点的实现策略也有区别，比如部分节点为了节省开支忽略了域名解析结果的TTL时间限制，导致用户在权威DNS变更的解析结果全网生效的周期非常漫长（我们已知的最长生效时间甚至高达48小时）。这类延迟生效可能直接导致用户业务访问的异常。

4.延迟大

DNS首次查询或缓存过期后的查询，需要递归遍历多个DNS服务器以获取最终的解析结果，这增加了网络请求的前置延时时间。特别是在移动互联网场景下，移动网络质量参差不齐，弱网环境的RTT时间可能高达数百毫秒，对于一次普通的业务请求而言，上述延时是非常沉重的负担。另一方面，弱网环境下的解析超时、解析失败等现象屡见不鲜，如何合理优化DNS解析对于整体网络访问质量的提升至关重要。



HTTPDNS

通过上文的介绍，聪明的读者应该可以发现，传统域名解析面临的诸多问题与挑战本质根源在于Local DNS的服务质量不可控，如果有一个更安全、稳定、高效的递归DNS服务帮助我们代理了域名解析的过程，上述问题看起来就可以彻底地得到解决。

HTTPDNS在这样的背景下应运而生。我们一起来看看HTTPDNS的基本概念以及它是如何解决传统DNS解析面临的问题的。

1.防域名劫持

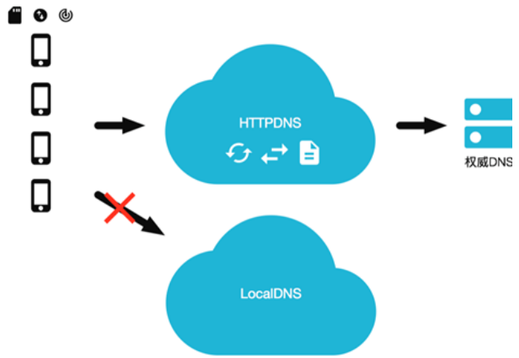
HTTPDNS使用HTTP协议进行域名解析，代替现有基于UDP的DNS协议，域名解析请求直接发送到HTTPDNS服务端，从而绕过运营商的Local DNS，如下图所示。

HTTPDNS代替了传统的LocalDNS完成递归解析的功能，基于HTTP协议的设计可以适用于几乎所有的网络环境，同时保留了鉴权、HTTPS等更高安全性的扩展能力，避免恶意攻击劫持行为。另一方面，商业化的HTTPDNS服务缓存管理有严格的SLA保障，避免了类似Local DNS的缓存污染的问题。

2.精准调度

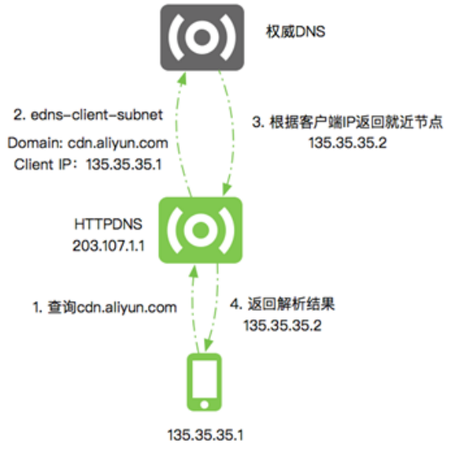
传统域名解析的调度精准性问题，本质根源在于Local DNS的部署和分配机制上。由于碎片化的管理方式，这些环节的服务质量同样很难得到保障。HTTPDNS在递归解析实现上优化了与权威DNS的交互，通过edns-client-subnet协议将终端用户的IP信息直接交付给权威DNS，这样权威DNS就可以忽略Local DNS IP信息，根据终端用户的IP信息进行精准调度，避免Local DNS的坐标干扰（当然上述精准调度方案的前提是权威DNS需要

支持edns-client-subnet，可喜的是当前主流的权威DNS服务都已支持该协议）。精准调度的流程示例如图。



3.实时生效

在域名解析生效周期方面，HTTPDNS也有着传统域名解析体系所无法具备的能力。前文中我们提到由于各个地区的Local DNS是独立维护的，服务质量参差不齐，缓存实现不一，因此导致的解析变更全网生效滞后的问题，在商业化的HTTPDNS服务上就不会存在（HTTPDNS严格遵循DNS　TTL限制进行缓存更新）。另一方面，即便我们假设Local DNS严格遵循域名TTL时间进行缓存管理（这里我们假设开发者配置的域名TTL时间为5min），当开发者业务受到攻击并需要快速进行切换时，Local DNS也会遵循域名TTL，在持续5min的时间段内返回旧IP信息，这5min的业务影响对于中大型企业而言是一个不小的损失（对于电商类的大型企业，5min的访问异常可能意味着几百万的交易额下跌）。以阿里云HTTPDNS服务为例，HTTPDNS在快速生效方面有专有的方案，配合阿里云的权威DNS服务云解析，用户在权威DNS变更的解析结果将快速同步给HTTPDNS，覆盖原有的缓存记录，帮助用户实现秒级的域名解析切换。



在DNS解析延迟方面，由于HTTPDNS基于HTTP协议，而HTTP基于TCP协议，对比传统的UDP传输多了一些冗余的握手环节，因此从原理上而言网络请求方面的开销并没有降低。但在实际使用过程中，我们可以通过端上的策略来实现一个零延迟DNS解析的方案。接下来我们一起来看看HTTPDNS服务在移动端的最佳实践方案。实时生效的流程如图所示。

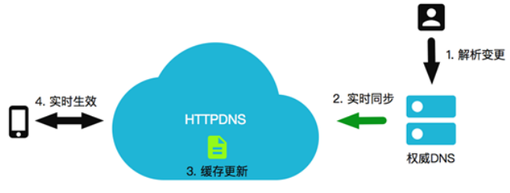
域名解析最佳实践

通过HTTPDNS服务，我们可以实现包括防止域名劫持、精准调度、实时解析生效等功能，但在DNS解析开销的优化上，我们需要客户端一起配合。

1.预解析

绝大多数的APP在应用初始化阶段都有一个启动期，我们可以在这个启动期做一些preflight工作，即在初始化阶段我们可以针对业务的热点域名在后台发起异步的HTTPDNS解析请求。这部分预解析结果在后续的业务请求中可以直接使用，进而消除首次业务请求的DNS解析开销，提升APP首页的加载速度。

在客户端实际使用HTTPDNS的过程中，有一个大家需要关注的点。标准的Web服务器（以Nginx为例）一般会将HTTP请求头中的Host头的值作为请求的域名信息进行处理（取决于服务端的配置，但一般情况都如此）。比如当我们通过标准的网络库访问www.aliyun.com/index.html这个地址时，发出的网络请求一般是这样的：




```
> GET /index.html HTTP/1.1
> Host: www.aliyun.com
> User-Agent: curl/7.43.0
> Accept: */*
```

使用HTTPDNS后，我们需要将HTTP请求URL中的Host域（注意这里的Host域指的是URL中的Host字段，而非HTTP请求头中的Host头）替换为HTTPDNS解析获得的IP，这时由于标准的网络库会将URL中的Host域赋值给HTTP请求头中的Host头，发出的网络请求如下：

```
> GET /index.html HTTP/1.1
> Host: 140.205.63.8
> User-Agent: curl/7.43.0
> Accept: */*
```

上述Host信息将导致服务端的解析异常（服务端配置的是域名信息，而非IP信息，试想一下如果我们的服务端服务了两个域名www.a.com和www.b.com，这时候它接收到一个140.205.63.8/index.html请求，它如何判断应该返回a的首页还是b的首页信息呢？）。为了解决这个问题，我们需要主动设置HTTP请求Host头的值，以Android的官方网络库HttpURLConnection为例：

```
String originalUrl = “http://www.aliyun.com/index.html”;
URL url = new URL(originalURL);
String originalHost = url.getHost();
// 同步获取IP
String ip = httpdns.getIpByHost(originalHost);
HttpURLConnection conn;
if (ip != null) {
// 通过HTTPDNS获取IP成功，进行URL替换和Host头设置
url = new URL(originalUrl.replaceFirst(originalHost, ip));
conn = (HttpURLConnection) url.openConnection();
// 设置请求Host头
conn.setRequestProperty(“Host”, originHost);
} else {
conn = (HttpURLConnection) url.openConnection();
}
```

主动设置Host头后，发出的网络请求就与未替换URL的网络请求一模一样了。

2.智能缓存

通过预解析获取的IP有一定的TTL有效时间，我们需要合理地缓存下来进行管理。操作系统本身的DNS缓存粒度比较粗，在客户端我们可以应用更细粒度的缓存管理来提升解析效率。比如在不同的网络运营商环境下，对CDN域名的解析结果会发生变化，当我们使用电信WIFI时，DNS解析会返回就近的电信CDN节点IP，当我们使用联通3G时，DNS解析会返回就近的联通CDN节点IP，针对不同运营商的解析结果缓存可以确保我们在网络切换时能够快速地进行网络请求，减免DNS解析带来的额外开销。甚至更激进的，我们可以做本地的持久化缓存，当下一次APP启动时直接读取缓存用于网络访问，以提升首屏加载的速度。

3.懒加载

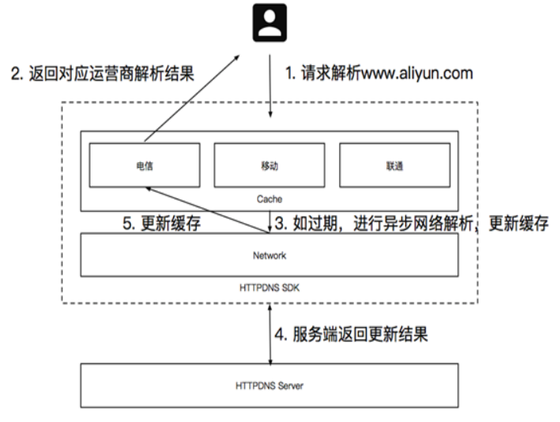
懒加载策略的实施可以让我们真正实现DNS的零延迟解析。所谓懒加载策略，核心的实现思路如下：

1）业务层的域名解析请求只和缓存进行交互，不实际发生网络解析请求。如果缓存中存在记录，不论过期与否，直接返回业务层缓存中的记录；

2）如果缓存中的记录已过期，后台发起异步网络请求进行HTTPDNS解析；

有的同学可能会有疑惑，返回一个过期的IP岂不是违背了TTL设计的初衷？的确，上述行为并不符合标准的规范，但是当我们重新审视一下自己的业务特点，上述的变通策略就显得非常有意义了。绝大多数的业务场景下我们的后端IP是固定的若干个节点，因此连续的解析结果在环境不变的情况下有很大概率是保持一致的，这在一定程度上保证了懒加载的可行性。另一方面，即便我们由于返回过期IP导致了访问异常的行为，后台很快会进行新IP的异步解析和缓存更新，业务本身可以进行重试和快速的复原，因此上述行为带来的影响也是非常小的。再进一步，TTL过期的IP的服务在绝大多数场景下还是持续的，可预期的，因此懒加载可能带来的业务风险是完全可控的。通过0.1%场景下的业务瞬时访问风险来换取99.9%场景下的用户体验提升，这笔买卖还是非常划算的（当然懒加载的使用有赖于合适的业务场景，如果你的业务场景下IP变化频繁，并且TTL过期的IP访问不可用，是不建议应用懒加载策略的）。

下图描绘了预解析+懒加载的实现框架：



综上可以看到，当我们需要实现零延迟解析的效果时，在客户端还是有比较多的工作需要做的。商业化的HTTPDNS服务提供了终端SDK方便开发者进行终端上的集成和使用，推荐大家可以尝试一下。

阿里巴巴ALIWARE

十年微服务架构演进历程中的挑战与实践

银时
阿里巴巴产品专家

1、说说微服务

微服务（Microservices）是一种重要的架构风格，目前越来越多的大型系统，开始将系统组件拆分成多个微服务，其已被公认为是云计算时代互联网应用的首选构建方式。按照微服务架构构建起来的系统，其内部各个微服务通常都是单独独立部署的，各自完成自己的功能职责，互相之间耦合度非常低，通过标准的协议进行互相调用。

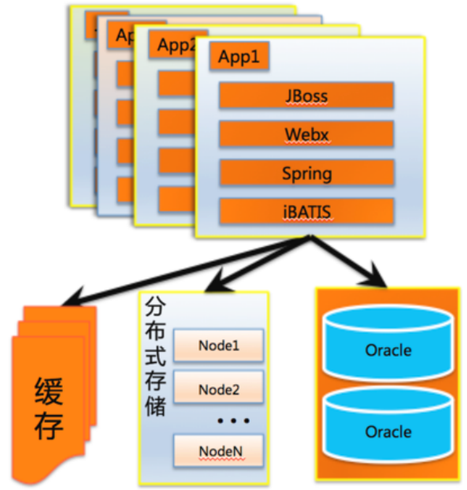
在如今的阿里巴巴平台上，生态百花齐放，业务创新不断涌现，而这都得益于阿里巴巴IT系统底层高可扩展的微服务架构。而谁能想到，早在10年以前，偌大的淘宝网，其整个站点都是运行在单一的部署包内，往往开发人员对其中某个模块的改动，都会对整个站点的稳定性提出挑战。此时的淘宝网，系统岌岌可危，业务发展也因为系统构架的缺陷而遇到了瓶颈。

自从2007年以来，在这近10年时间里，阿里巴巴技术团队一直在微服务的道路上摸索前进着，其间伴随着互联网和移动互联网的盛行，海量的用户一次又一次的洗礼了各个机构的IT系统，而在阿里，这种冲击无疑更加频繁和剧烈——这些年下来，阿里中间件技术完成了从1.0到3.0时代的蜕变，对于海量微服务的治理能力处于业界领先，并形成了自主技术产品和品牌——Aliware。同时，这些中间件技术也进行了产品升级，以商业软件和服务的形式普惠众多IT同仁。

互联网中间件

2、服务化缘起

在2007年的时候，阿里技术团队规模大概是500人左右，当时的主要业务站点是淘宝网，都使用一个单一的WAR包进行部署，现在我们把当初这样的系统架构称为烟囱式架构；在架构上，采用传统的JAVA EE应用开发架构，使用JBoss作为应用服务器，底层使用Oracle数据库——而与此同时，淘宝网的业务每年翻倍增长。

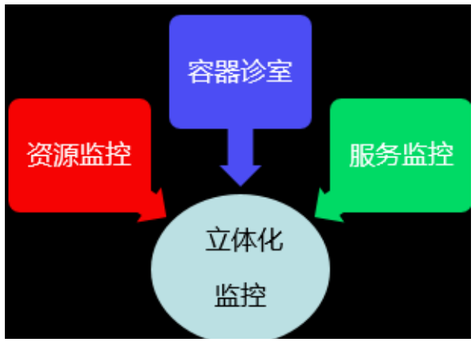


在那个阶段，我们面临着非常多的挑战，总体归纳下来有三点：

第一，系统的研发成本非常高。整个技术部门上百人一起维护一个核心工程，会碰到很多问题，其中大量的源代码冲突和高昂的协同成本是两个最主要的问题；此外，每次项目发布前，各个技术团队之前配合非常困难，往往一些团队已经早早的做好了发布准备，而另一些团队还在紧张的开发中，从而导致项目发布周期非常长；更为严重的是，整个站点上所有的逻辑都是耦合的，相互之间错误难以隔离，通常对淘宝网整个工程里的某个模块或某个系统功能进行一些变更时，整个系统都会面临非常大的技术风险。

第二，数据库能力达到上限。淘宝早期使用的是Oracle数据库，随着业务的不断增长，应用数也不断增多，单机Oracle数据库的连接数越来越不够用；同时，单机数据库的吞吐达到瓶颈，CPU长期运行于90%以上;此外，数据库本身的容量也日趋饱和——总得来讲，Oracle数据库长期高负荷运转，接近崩溃边缘。

第三，形成大量数据孤岛。业务快速发展，烟囱式架构应用越来越多，一方面，系统的重复建设非常严重；另一方面，每一个烟囱式架构应用都对应各自的数据库，相互之间的数据处于隔离状态，数据不一致问题突出，无法复用，也无法进行大数据分析建模。



3、微服务架构的形成

正是面临这些问题，从2007年开始，阿里巴巴开始了对分布式微服务架构的探索。整个微服务架构落实到技术层面，就是把原本烟囱式系统上的模块，按照共享服务中心的理念进行拆分，分布式的部署到各个机器上，实现服务的共享。为了实现这样的架构，最基本的就是需要这样一个框架：该框架能够低侵入地构建起不同机器、机房和模块之间的服务化调用，并且能够高效的组织服务发布、注册以及发现等过程。目前阿里巴巴生产环境使用的是第三代RPC框架：EDAS-HSF，公司内部90%以上应用（包括所有的在线核心交易链路系统）当中使用，前后历经了8次双11大促高并发与大流量的考验，同时支持高性能的分布式事务。此外，阿里也早在2011年的时候，就已经将第一代RPC框架EDAS-Dubbo进行了开源，目前已经成为国内最活跃开源软件之一，开源分支达4000多个，蝉联2016年度国内开源软件Top10。

如果说RPC框架用来解决分布式微服务系统的同步调用问题，那么消息中间件则为异步解耦而生。消息中间件是一种由消息传送机制或消息队列模式组成的最典型的中间件技术。通过消息中间件，应用程序或组件之间可以进行可靠的异步通讯来降低系统之间的耦合度，从而提高整个系统的可扩展性和可用性。MQ是阿里巴巴自主研发的消息队列产品，是支撑双11最为核心的系统之一，在淘宝、天猫和支付宝的核心交易场景中都有大量使用。如今，阿里已将MQ的开源版本RocketMQ捐赠给 Apache 基金会，成为 Apache 孵化项目

对应用的监控是我们非常关注的事情，常规的应用监控往往只会针对系统级别的监控，常见的包括：CPU使用率、-

Load（负载）、内存使用率、磁盘使用率、磁盘读写IO和网卡上下行流量等。系统级别的监控是非常关键，也是必不可少的——但是这样的监控毕竟过于宏观，对于问题的定位不是特别精准。因此，在我们的监控方案里面，除了对系统级别的指标进行监控外，还会尝试从不同层面收集信息，实现对应用立体化的监控，包括系统指标、应用容器和服务三个层面的监控，具体包括以下三大方面：

- 系统指标

系统指标比较常见，这里不再赘述。

- 应用容器

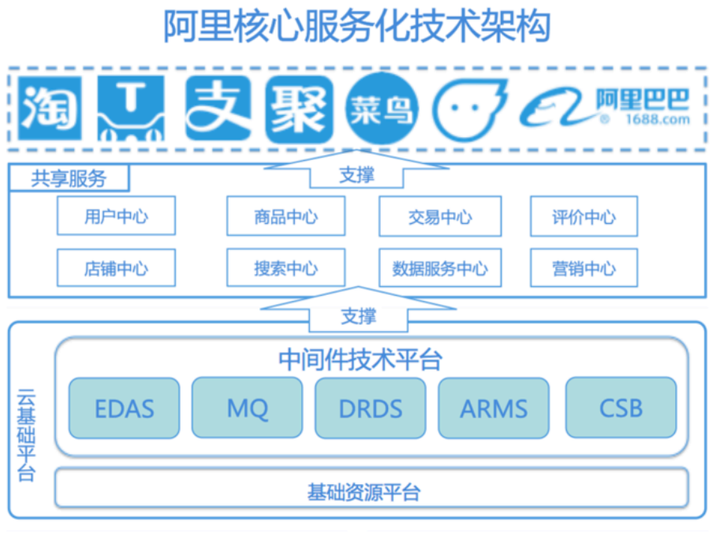
阿里巴巴的技术体系绝大部分围绕Java技术体系，作为包含Java运行环境的Java容器是监控的重点，因此我们对Apache Tomcat进行了改造，其中一个便是增加了应用容器层面的多个监控，包括：堆内存与非堆内存使用情况、类加载情况、异常线程（提前将线程情况全部显示出来）和HTTP请求处理情况等方面的监控。

- 服务监控

在集中式的系统架构里面，每个页面会贯穿非常多的模块，每个模块都耦合在一个系统中，最终监控出的是系统的整体表象，比如：网站打不开了、系统访问变慢了以及接口调用出错了等等——但终究无法迅速定位到页面打开慢是因为哪个模块的哪个功能逻辑上的问题。在阿里内部，我们会对每一个服务接口以及方法的实时调用情况进行监控，包括对服务调用的实时QPS、响应时间、出错情况进行统计，同时快速感知系统流量变化。

4、微服务架构的阶段性成果

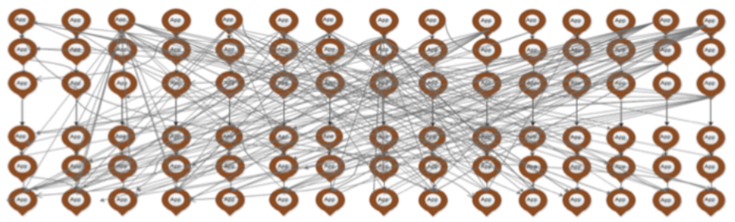
淘宝网围绕EDAS微服务技术体系进行了一系列的服务化改造，在这个改造过程中，我们首先将数据复用度最高的用户模块进行拆分，剥离出第一个共享服务中心：用户中心——所有和用户相关的基础操作，都由该用户中心统一提供。接下来又陆续有千岛湖项目和五彩石项目，这些项目的背后都是阿里一系列的服务化中心拆分的过程，后来经过6～7年的服务化演进，目前服务中心数已达50多个。



上图是阿里巴巴技术平台核心服务化架构。如今回过头去看这段服务化改造历程，总得来说，通过自主创新走出了技术困境，同时沉淀一大批成熟互联网中间件技术；而随着共享服务中心理念的提出与落地，打破了原有应用“烟囱式”的建设方式，不仅提升了系统稳定性，而且具备了支撑业务快速创新的能力。

5、海量微服务的挑战与实践

随着业务的不断发展，服务化拆分会变得越来越精细，应用数量也会变得越来越多，应用之间的服务依赖则会变得日益复杂。



图示便是随着服务化进行，大规模分布式系统之间服务依赖关系的变化。当服务拆分初期，我们可以依赖于架构师个人能力，来进行整体的系统架构、服务依赖梳理。然而，随着服务化的不断深入，完全人工的梳理变得不现实。因此，在阿里内部，我们开发了一系列数据化运营功能，来自动化的分析并定位海量服务场景下的服务治理。

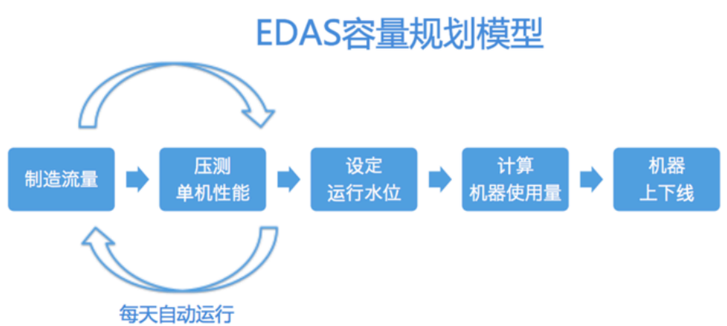
上图阿里内部日常使用的EDAS鹰眼监控系统。图中右半部分，从上至下表示的是用户从页面打开到页面完整响应所经历的所有分布式各层系统的调用。我们依靠这一整套的链路跟踪系统，能够在系统出现故障的时候，非常清晰的定位到故障根源在哪。

EDAS鹰眼监控系统不仅对问题定位起到了非常大的帮助，在对于分布式系统的数据化运营上，也起到了非常大的作用。当我们把类似的请求调用链路全部汇总起来进行分析后，就可以进行实时的数据分析，通过对海量调用链进行统计，得到链路各个依赖的稳定性指标。以峰值QPS为例，其代表的是在某一段时间内，某一个业务服务在服务化的调用过程中达到的最大的QPS，如图中标记可以看出，虽然其上层系统暴露于最前端，但不一定是压力最大的，相反，该系统反而是整个分布式系统中最忙碌的模块——这就是数据可视化带给我们的价值所在。

在过去的6～7年时间，我们沉淀了一整套的容量规划模型。 第一步：制造流量

首先我们希望整个压测过程都是用的真实的环境和流量，因此在第一步将线上真实流量进行引流，压测集群中部分机器的单机性能。

第二步：压测单机性能。



在进行单机性能压测时候，我们会设置到单机系统指标（通常包括CPU、Load和内存），当系统指标到达指定阈值，便不再加压。举个例子来说，设定CPU的阈值为70%，那就表明当某台机器的CPU使用率达到70%的时候，就不会继续对该机器继续引流——此时单机的服务指标QPS便可以作为该机器的最佳单机性能。值得一提的时，这样的单机性能指标才是真正可以被参考的，而不是我们很多工程师惯用的将单机压到极限瓶颈时候的单机指标。通常，生产环境集群单机CPU使用率70%是一个比较合理的水位。

有了一个可靠单机性能有，之后的计算过程并没有特别精妙的设计，这里不再赘述。

最后，为了在大促时保证系统更稳定的运行，采用了限流和降级的手段。根据不同服务的优先级，不同服务的重要程度来执行限流和降级的措施。限流降级是阿里最有特色的功能之一。每年，我们都会面对双11大流量和高并发的考验，需要在成本和体验中选择一个最好的平衡点。

6、总结

正是由于有了双11这样真实的业务场景验证以及阿里技术人的不断创新，成就了阿里巴巴Aliware世界领先的高可用架构基础设施和世界一流的中间件产品体系，而阿里巴巴也成为了业界为数不多的将内部最核心系统直接对外开放的公司。如今，Aliware系列产品已经为上万客户提供高可靠、高性能和高可扩展的分布式架构基础设施。诚邀您一起拥抱互联网转型。

轻松定制
跨终端视频点播服务

前段时间的朋友圈被《奔跑吧，兄弟》、《欢乐颂》、《琅琊榜》等各种刷屏。现在使用点播技术的视频网站越来越火。本文将介绍如何通过阿里云定制跨终端的视频点播服务。

首先，普及下视频点播的起源：视频点播（Video on Demand ,简称VOD）是20世纪90年代在国外发展起来的，根据观众的要求播放节目的视频点播系统，把用户所点击或选择的视频内容，传输给所请求的用户。

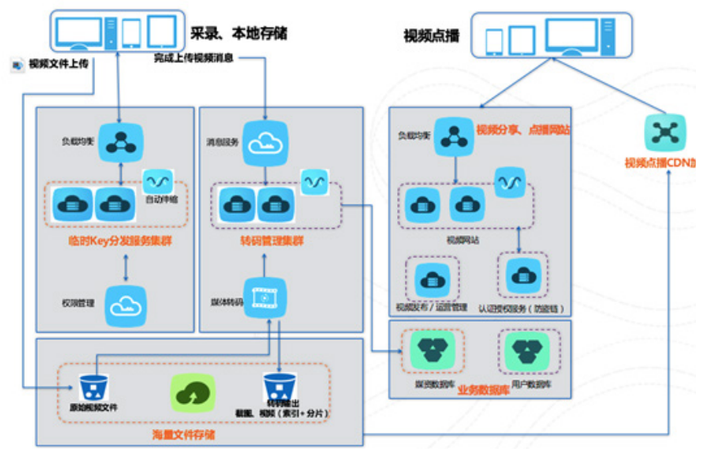
那么，视频点播系统是如何实现的呢？当用户发出点播请求时，流媒体服务系统就会根据点播信息，将存放在片源库中的节目信息检索出来，以视频和音频流文件，通过告诉传输网络传送到用户终端。

目前，视频点播使用的主流协议有哪些呢？协议之间有什么区别？

1.RTSP／RTP／RTCP协议簇，最早的视频传输协议。RTSP协议用于视频点播的会话控制；RTP协议用于具体的视频数据流的传输；RTCP协议用于在视频流数据之外，丢包或者码率之类的控制。

2.HTTP协议，主要是在互联网普及之后，主要用于PC端或者网页端，视频点播业务，最常见的解决方案，资源一般采用flv格式，也可以使用mp4格式。

3.HLS、HDS、MSS、DASH，苹果推出HLS（HTTP Live Streaming），随着苹果设备的普及得以广泛应用的一种。HTTP采用m3u8作为索引文件，视频为MPEG2-TS格式的片段文件；相应的，adobe公司推出类似的HDS（HTTP dynamic streaming），这种方式本质和HLS的策略是类似的，也就是通过索引文件＋视频片段的方式，但是采用的索引格式和视频片段格式都不一样。HDS采用视频格式是flv或f4v；类似的，微软也推出MSS（Microsoft Smooth Streaming），采用的视频格式是分段mp4格式。MPEG标准则推出DASH（Dynamic Adaptive Streaming over HTTP），采用的视频格式为3GPP。



4.HTML5，HTML制定厂商推出HTML5，本质上和HTTP视频协议没有任何区别。但是，播放器端不再依赖于特定的插件，如flash或者其他播放软件。而是，采用html中嵌入video标签，同时指明视频的url的方式。比较通用的视频H.264格式，音频ACC格式，封装格式MP4。

5.RTMP，是adobe公司推出的视频协议。需要专用的服务器，如FMS等。

一般情况下，视频点播的主要业务场景：视频网站和家庭监控录像点播。在如上的两个业务场景中，用户经常会遇到如下的技术问题：

1.资源消耗大且增长迅速，不同于Web服务，视频点播业务十分消耗存储资源，一个成规模的视频点播网站会有百TB甚至PB级别存储资源，普通IDC或小云服务提供商有限的基础设施很容易成为云点播业务爆发增长的瓶颈，且彼时扩容难度大，迁移成本高。

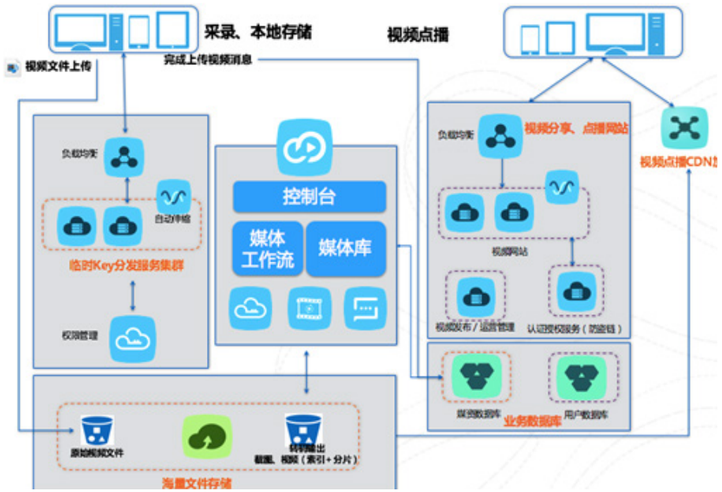
2.对网络宽带、网络质量敏感。高清、流畅是视频点播最重要的用户体验，高清视频码率高，需要有充足、优质的网络宽带来保障首播延迟在容忍范围内，保障视频可以在大并发访问场景下流畅观看。并且，还要兼顾视频分发带来的高带宽的成本问题。

因此，建议将视频点播迁移到阿里云上，阿里云提供一个稳定弹性的多媒体架构，对视频文件进行一系列的处理，满足不同场景的需求。

左图是视频点播的架构图，对于不是很了解多媒体行业的用户来说较为复杂，实现也有较大的难度和开发量。但是通过最新的阿里云产品“视频点播”，无需编

写代码就可快速搭建视频点播服务。

视频点播服务（VOD）是集音视频上传、自动化转码处理、媒体资源管理、分发加速于一体的一站式音视频点播解决方案。借助灵活、可伸缩的存储、处理及内容分发服务，帮助企业和开发者快速搭建安全、弹性、高可定制的点播平台和应用。



阿里云视频点播服务，支持快速搭建云端音视频点播解决方案。用户可以在短时间内，0代码完成常见云端音视频处理流程配置，文件上传完毕即可自动触发执行。所有服务按实例付费，服务能力自动伸缩，告别复杂的架构设计和编程开发，维护成本几近于零。提供多终端上传，自定义媒体工作流、高可定制转码方案等；窄带高清和H.265技术，提供高画质、低码率的自适应转码输出；跨运营商、跨地域的全网覆盖；以及安全和高可靠的云存储服务保障海量音频文件永久可靠存储。

在使用视频点播产品后，基于阿里云产品搭建的视频点播架构，可以简化右图：

阿里云提供的视频点播服务主要提供如下的服务：

1.媒资，支持媒资库、媒体工作流；

2.存储，集成对象存储服务（OSS），提供海量、安全和高可靠的云存储服务；用户可以通过视频点播控制台或使用面向Web、iOS、Android的上传SDK工具进行文件上传工作，支持分片上传、断点续传、批量上传。也可以直接使用OSS客户端工具。

3.转码，集成媒体转码服务（MTS），满足定制转码需求；

4.分发，集成内容分发网络（CDN），跨运营商、跨地域全网覆盖的网络加速服务，支持千万级并发播放及灵活可定制的防盗链能力；

5.消息，集成消息服务（Message Service，原MQS），保障媒体工作流执行消息传递，使用户可以将视频点播服务灵活于自由系统和服务集成；

用户第一次使用阿里云视频点播服务就会感觉到：“太人性化了！”。阿里云视频点播提供各种配置的工作流模板：比如M3U8切片工作流，预智能工作流，多码率多格式工作流，等等。这样，就可以满足用户对视频点播的大部分场景的需求。不仅如此，用户还可以个性化定制专属于自己的工作流模板。

阿里云官方实验平台：云中沙箱（http://lab.aliyun-du.net），为用户提供一个真实的阿里云环境，用户可以亲自动手实践阿里云视频点播服务。

视频点播

鲍天舒
上海驻云信息科技有限公司

升功
阿里云高级技术专家

ROS 以更优雅的方式实现弹性架构

为什么弹性架构是重要的

谈到IT系统架构，我们经常会用建筑架构来做类比，事实上，Architecture这个词也正是来自于传统的建筑行业。系统架构图就像建筑设计图一样，用来指导软件构建。同时，两者之间也存在一个巨大的不同，建筑物追求的是屹立千百年不变，而IT系统追求的是灵活性，以便随时应对真实世界的业务变化。从这个角度来看，IT系统更像一个有机的生命体，需要不停的改变自己去适应外界变化，才能让自身更有生命力。系统压力是一个重要的变化维度，是否能够随着系统压力的增减而动态的调整系统的某一部分体现了架构是否具有足够的弹性。

水平扩展能力，是互联网架构重要的弹性表现。当系统面对增加的用户和访问量时，是否能够快速进行水平扩展，往往决定了互联网应用的生死。另一方面，当访问量回落后，是否能够释放掉资源以节约成本会变成核心竞争力之一。根据扩展的粒度不同，这种扩展可能是机器或组件粒度的，比如增加或减少云服务器的数量，也可能是整体架构粒度的，比如在线游戏开新服，就需要复制整个系统。

实现弹性架构有很多方法，重要的问题是，如何更优雅的让系统具备弹性？让IT系统应对变化时能够快速、可靠和自动化？

实现弹性架构的基本原则

数据化运维

让系统具备弹性的目的是为了应对业务变化，而感知变化是首要的事情。这就需要系统能够具备数据化的运维能力，从收集系统运行数据，到最终根据数据做出是否需要触发弹性变化的判断。

按需获取资源

系统的弹性变化最终会落到对资源的调整上，可能是流量增加时增加云服务器，流量减少时减少云服务器。这就需要系统能够随时根据需要获取和释放计算资源。

可复制和可维护

系统的弹性变化是随时都有可能发生的，所以弹出来的部分一定是被内置到系统内部的，是可以无限次被执行的。同时，又能够很好的被开发人员维护，以代码的形式成为系统的一部分，可以被修改、测试、追踪版本等。

自动化

自动化是减少系统错误的重要手段，所以以上原则都需要能够自动化执行。

在阿里云上实现弹性架构的实践

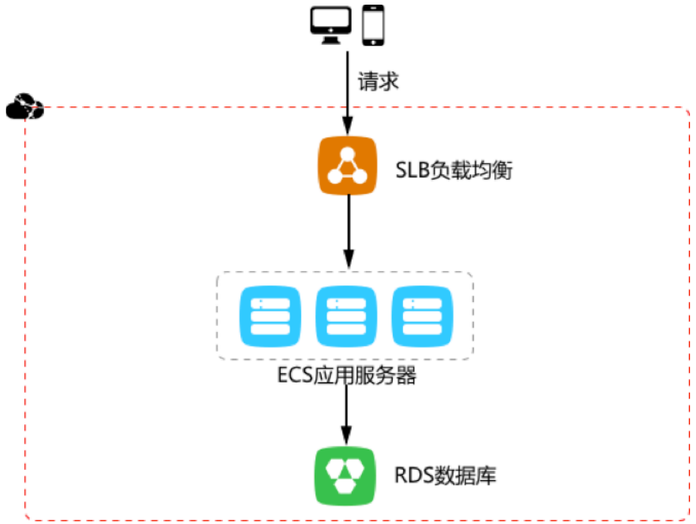
基于阿里云提供的运维管理服务，我们可以很容易的实现弹性架构。其中最重要的两个服务是云监控和资源编排。

云监控(CloudMonitor)可用于收集阿里云资源的监控指标，探测服务可用性，并针对指标设置报警。通过OpenAPI，可以实时获取系统的运行状况，用来作为弹性变化的触发依据。

资源编排服务(ROS)，提供了通过模板管理云资源的能力,用户可以在JSON格式的模板文件中，描述系统的资源和配置，ROS会根据用户的模板，创建和释放资源，并对

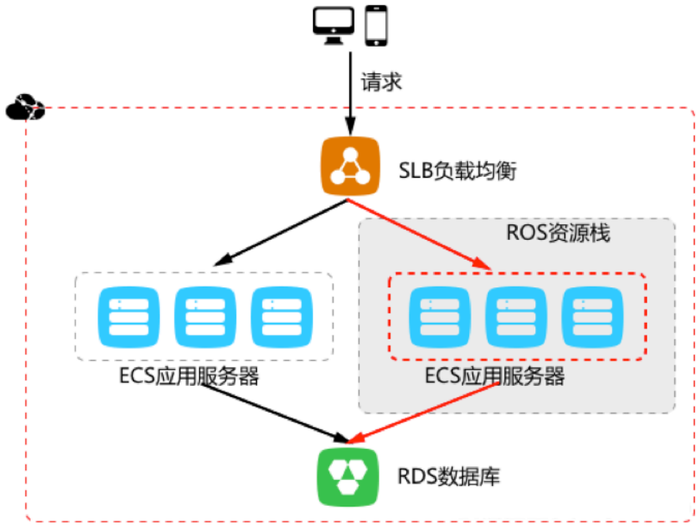
资源、软件做指定的配置。模板是一种描述性的文本文件，可以像代码一样被编辑、测试和版本控制。通过模板，我们可以把弹性变化的部分进行完整的定义，并且可以被无限次的重复执行。同时，ROS是一种无需人工值守的完全自动化服务。通过ROS提供的OpenAPI和CLI工具，结合云监控的数据，可以把弹性变化整合成自动的机制，让系统拥有完全的弹性能力。

下面用一个例子来介绍如何基于阿里云提供的服务来构建弹性架构。架构如右图：



这是一个非常典型的基于阿里云的Web系统。系统通过ECS提供Web服务，通过RDS来做数据持久化存储，通过SLB来做访问负载均衡。我们要解决的问题是，如何才能让系统具有弹性，以应对可能出现的访问量暴增。一个直观可行的思路就是，直接增加ECS服务器的数量。但是这涉及到一些复杂而琐碎的事情。比如创建ECS实例、初始化系统、安装软件、把ECS实例挂在到SLB，当流量回落以后把增加的ECS从SLB摘除并释放实例。

我们的目标是更优雅的实现弹性架构，也就是要让架构符合之前所说的原则。通过云监控和资源编排，可以很容易做到这一点。云监控可以在控制台直观的配置。我们重点来看看资源编排(ROS)的角色。ROS通过模板创建一批资源，这些资源被成为资源栈（Stack）。所以，当通过ROS来为系统弹出资源以后，系统会变成右图所示：



依据监控数据，我们通过ROS创建了一个资源栈，其中包含3台按量付费的ECS实例，根据我们在模板中的定义，ROS会在这些实力中安装我们的业务软件，这些软件会访问系统已存在的RDS存取数据。当ECS实例启动后，ROS把这些机器加入到SLB，新的访问就会被分发到这些新增加的ECS上。

当访问量回落后，我们通过ROS销毁这个资源栈，ROS会首先把ECS从SLB中摘除，这样新的访问就回到了最开始的ECS上，ROS接着把ECS删除。整个过程不会对用户访问产生影响，系统又回到了最开始的状态。

上面所有的事情都是自动进行的，我们把弹出的内容用ROS的模板进行了定义，其中描述了所有关于ECS创建、软件安装、SLB挂载的细节。因为模板是可以被像代码一样管理的，所以我们弹出的内容是经过了严格测试的，并且能够跟踪回溯版本。

ROS的模板是申明式的JSON文件，所以我们只需要在里面描述清楚我们希望系统达到的状态，而不需要关注如何实现。这个模板文件里的主要部分(需要根据实际情况修改)：

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Resources": {
    "newEcsInstances": {
      "Type": "ALIYUN::ECS::InstanceGroup",
      "ImageId": "m-25xovkjg5",
      "InstanceType": "ecs.s2.large",
      "MinAmount": 1,
      "MaxAmount": 3,
      "SecurityGroup": "sg-25f9isr5m",
      "VpcId": "vpc-bp168go6c51f16786k9dl",
      "VSwitch": "vsw-bp1r47wdty5ecx4e82j9qaa"
    },
    "backendServerAttachment": {
      "Type": "ALIYUN::SLB::BackendServerAttachment",
      "Properties": {
        "LoadBalancerId": "15187200816-cn-beijing-btc-a01",
        "BackendServerList": {
          "Fn::GetAtt": [
            "newEcsInstances",
            "InstanceIds"
          ]
        }
      }
    }
  }
}
```

上面的例子稍显简单，其实，阿里云的资源编排服务基本实现了对所有阿里云服务的支持，更有像实例克隆、资源栈更新这样的高级特性，我们可以在一个模板中描述包含大量资源的复杂架构。

总结

我们在上面介绍了弹性架构的重要性和基本原则，并结合阿里云的云监控和资源编排服务，介绍了如何在阿里云上实现弹性架构。随着我们的系统与云计算结合的越来越密切，弹性架构已经成为使用云计算的标准能力，同时也是系统的核心竞争力之一。构建更加优雅的弹性架构，不仅需要我们具备理论，也要能够熟练的整合云计算资源和服务。

资源编排ROS

OSS 对象存储

如何使用RTMP功能直播鉴黄

闫卫斌
阿里云专家

概述

当前直播发展如火如荼，越来越多的直播平台开始涌现；RTMP则是直播中使用最为广泛的协议之一；另外，监控产品也开始广泛的使用RTMP协议。为了方便直播/监控用户使用OSS来存储音视频数据，OSS近期推出了RTMP收流功能。用户可以直接用RTMP协议将音视频数据上传到OSS。

OSS LiveChannel介绍

LiveChannel是OSS为了管理RTMP推流新增的概念；LiveChannel即直播频道，用来保存推流配置，获取推流状态等；直播频道从属于bucket，一个用户可以在bucket中创建无限多的直播频道。

用户创建一个LiveChannel即可以获取一个对应的RTMP推流地址，随后用户可以将音视频数据通过RTMP协议推送到OSS，转储为HLS协议的ts、m3u8文件。转储的文件可以用来做点播；在延迟要求不高的场景下也可以直接使用HLS协议做直播；另外，OSS还支持收流的同时实时截图，鉴黄来做协助用户做内容审查。

通过LiveChannel，用户可以在直播的同时将内容存储到OSS，不需要任何的本地临时存储；在享受OSS高可靠、低成本、无限扩展的云存储的同时，还可以利用OSS强大并且还在不断丰富中的数据处理能力（截图、转码、鉴黄等等）来进行各种后期处理；还可以非常便捷的使用OSS的Lifecycle等功能来做数据的生命周期管理。

使用场景说明

1.RTMP直播转录为HLS

步骤说明

1）APP开始直播，推送RTMP流到CDN；

2）APP或者CDN发送“开始直播”的消息到MNS Topic，信息中包含本次直播的CDN拉流地址；

3）MNS推送消息到用户搭建在ECS上的RTMP Proxy；

4）RTMP Proxy使用ffmpeg从CDN拉流；

5）RTMP Proxy在OSS创建LiveChannel，并向OSS推流；

6）转推结束后，RTMP Proxy发送一条“推流结束”的消息到另一个MSN Topic；

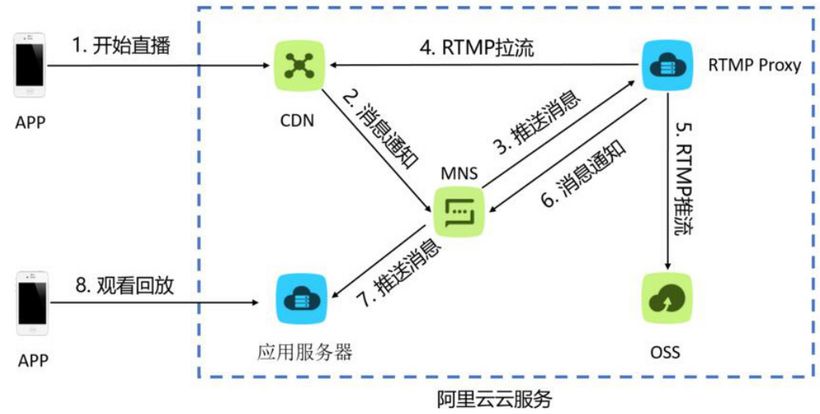
7）MNS向客户的应用服务器推送“转推结束”的消息，供后续处理（例如保存点播地址到数据库）；

8）APP从应用服务器得到点播地址，访问OSS获取点播视频。

备注

1）用户可以自己使用OSS/MNS SDK、ffmpeg、librtmp等来实现自己的RTMP Proxy，后续OSS会提供示例程序；

2）整个步骤中涉及到的组件都可以使用阿里云提供的服务搭建。



2.实时鉴黄

步骤说明

1）直播时使用RTMP Proxy转推一路数据到OSS，并设置LiveChannel开启截图；

2）OSS按用户指定的间隔截图，并保存到OSS；

3）OSS调用第三方鉴黄服务对图片进行打分；

4）OSS将打分的结果推送到MNS Topic；

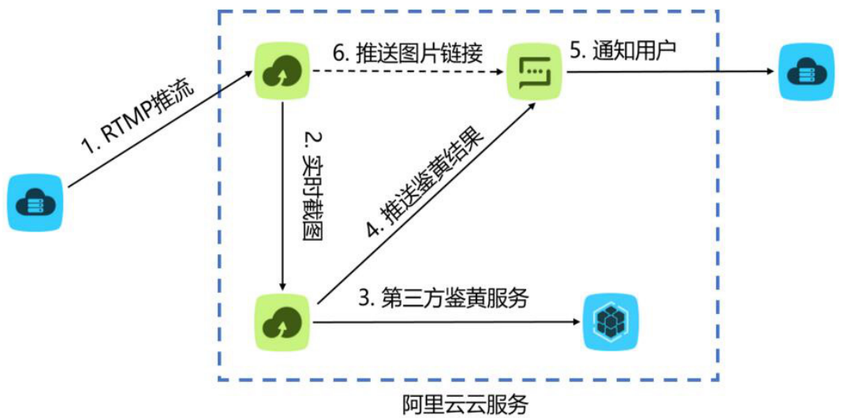
5）MNS回调用户的应用服务器通知鉴黄结果；

6）另一种处理方式：用户也可以选择让OSS直接推送截图的图片链接，应用服务器将图片地址聚合后采用其他方式鉴别；

备注

1）第三方鉴黄服务会部署在阿里云提供的容器服务中，调用的延迟等有充分的保证；

2）截图的同时OSS仍然会将RTMP流转储为HLS文件；



权限控制

OSS提供了丰富的鉴权/授权机制让用户可以精细的控制数据的访问权限；对于HLS直播/点播场景，我们同样提供了“动态签名m3u8”的机制，使用户可以使用私有bucket提供HLS播放服务。

用户只需要使用URL签名方式访问m3u8，并且增加参数“x-oss-process=hls/type”，OSS会对返回的播放列表中的所有ts地址按照与m3u8完全的相同的方式进行签名（相同的accessId、accessKey、expireTime）。

假设某个m3u8文件的内容如下：

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:54
#EXT-X-TARGETDURATION:6
#EXTINF:6.006,
1470971233380.ts
#EXTINF:6.006,
1470971233398.ts
#EXTINF:1.944,
1470971233415.ts
```

动态签名后返回的内容如下：

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:54
#EXT-X-TARGETDURATION:6
#EXTINF:6.006,
1470971233380.ts?Expires=1470973160&OSSAccessKeyId=YJjHKOKWDWINLKX-v&Signature=6aJidj9VCRqnv%2Bwszh9MeROeHPM%3D
#EXTINF:6.006,
1470971233398.ts?Expires=1470973160&OSSAccessKeyId=YJjHKOKWDWINLKX-v&Signature=ek3l5uK3R8FlnZLLWCmzsJXo7wk%3D
#EXTINF:1.944,
1470971233415.ts?Expires=1470973160&OSSAccessKeyId=YJjHKOKWDWINLKX-v&Signature=JMjdUcCGu63bgtiIHeEi0USyY18%3D
```

备注

1. “动态签名”不会改变存储在OSS中的m3u8文件的内容；
2. 支持子账号、STS，使用STS访问时，Token必须通过URL参数提供；
3. x-oss-process参数需要参与签名。

总体来讲，目前OSS已经利用RTMP协议实现了直播视频的转录，同时还能够进行视频内容的鉴黄，为视频直播企业提供了快速搭建业务系统的便利方案，助力视频直播企业快速增长。

对象存储OSS

专有网络VPC
网络中省钱的秘密

辰宿
阿里云产品经理

目前，基于VPC构建应用和网络已经成为一个主流，VPC网络不但在功能上和安全上有着非常卓越的表现，其实在成本上也有很大优势。本文整理自云栖大会 厦门峰会上阿里云网络专家产品专家辰宿的演讲，本文将带大家领略VPC省钱的秘密。

本文和大家分享云计算平台上网络方面一些可优化成本的秘密，以及在云计算平台上真正使用这些产品时的省钱小技巧。

假设大家是创业公司的CTO，第一天需要把企业的网络组建起来，需要做什么事情呢？第一件事情就是要去买网络设备，包括交换机、路由器、网关和网闸等等。也许很多同学对于在物理世界中组建网络已经非常熟悉了，但因为有一些同学不太熟悉，所以简单分享一下，这里的路由器和家庭使用的路由器完全不是一个概念，它是数据中心级的路由器，这种路由器最便宜的也需要上千元，当网络规模大的时候这项费用达到五位数或者六位数也很常见。所以在这里需要投入很多资金，当购买完设备以后，需要租用机房的机架，通过网线将它们链接起来和服务器一起组成整个应用的部署环境。当部署完网络环境以后会发现其中一部分是面向互联网提供服务的，所以自然需要购买公网带宽。其实在物理世界里真正需要的不只是一根线，有可能很多根。这是因为在中国的互联网里，联通的用户访问电信的机房会很慢，同样电信用户访问联通机房也会很慢，所以当需要对不同运营商的用户提供服务的时候需要多根不同服务商的宽带。

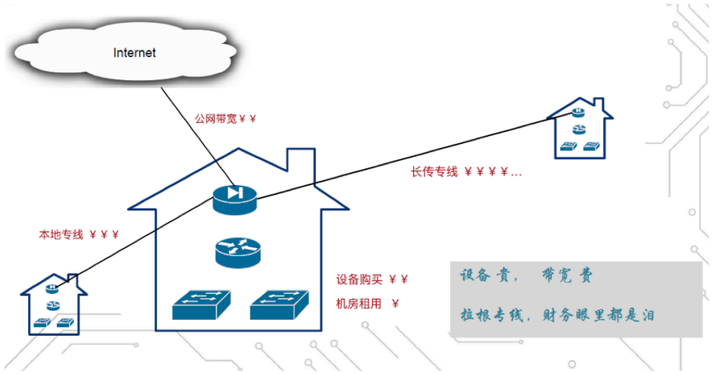
随着业务发展可能会遇到很多问题，比如所租用的机房的机架位置不够了，但是业务的飞速增长，必须要租用新的机架，只好另找机房。也有可能是随着业务价值越来越大，需要考虑高可用和多中心部署这些问题，于是又租用了其他机房的机架。这个时候往往需要专线连接来满足高性能的内部调用或者高速的数据传输与同步，所以这部分也需要投入大量资金。而在实际情况下第三方的IDC专线费用会相当高。

随着业务规模进一步扩大，就出现了所谓的“两地三中心”的情况。当业务价值进一步提高，可能需要考虑远距离多地域的多中心部署，这时候需要专用的线路去保证整个企业应用的网络互通性。而所谓的长距离传输专线的价值就不是一般昂贵了。

其实整体看来可以将传统物理世界里的组网成本分成这样几块：数据中心成本，公网带宽成本也就是南北向组网成本，还有就是IDC也就是自己的部署环境里的组网称为东西向的组网成本。

而在这样的组网成本面前可以总结几句话就是：设备贵，带宽费，拉根专线，财务眼里都是泪。

而到了云计算平台上要做刚才那些事情有什么不同呢？首先需要创建VPC。很多云计算用户还不知道VPC是什么，简单而言，VPC就是在云计算平台上使用免费的路由器和交换机组建的可自定义网段的私网环境。在VPC里面可以添加用户的服务器和数据库。在云计算平台上IDC和虚拟设备等成本基本为零，仅仅针对某些需要带有特性或增值的虚拟网络设备的高级用户会收取一些费用，除此之外都是免费的。如果大家还没有用过VPC可以直接去阿里云官网上创建试试，阿里云VPC不仅一分钱都不



传统IDC组网结构与成本

要，还可以自定义很多功能，这使得使用VPC与用户在线下自己组建网络基本没有差别。

当然在这种途径下用户也需要购买公网带宽，但是云计算厂商提供的公网带宽和用户在线下直接找供应商购买的公网带宽有很大差别。所以具体哪个省钱，其中又有哪些技巧后面将详细分享。

接下来的一部分内容对应同城专线来搞定多中心部署。阿里云上推出的一款产品叫做高速通道，用户在控制台上或者通过API调用仅花费一两分钟时间做几个简单操作所得到的效果就相当于在物理世界里花费一两个月通过运营商在两个机房之间搭建一根专线的效果。所以这是很方便的，而其中的费用也可以通过对比来看一下优劣。

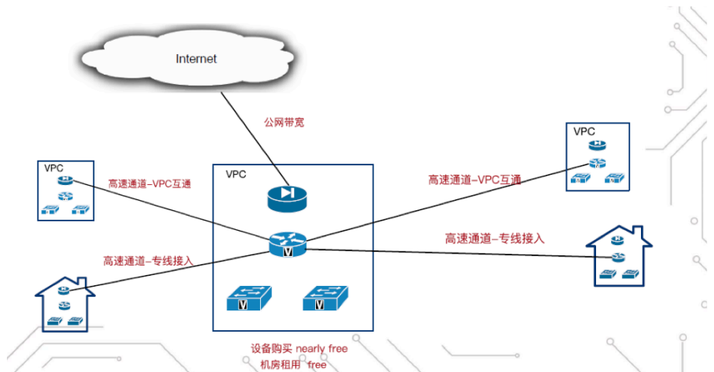
也许企业在拥抱云计算之前并不想把之前机房里面的服务器当做废铁扔掉，而是想拥有混合云的架构，把线下机房的服务器和云端的服务器结合在一起。但是除了阿里云以外几乎所有的云计算平台都没有提供远距离的VPC互通，因为这中间有很多技术门槛。大家可以直观地感受到阿里云在虚拟网络技术方面在业界的水平，阿里云支持远距离的VPC互通，自然也可以通过远距离接入来支持混合云架构。

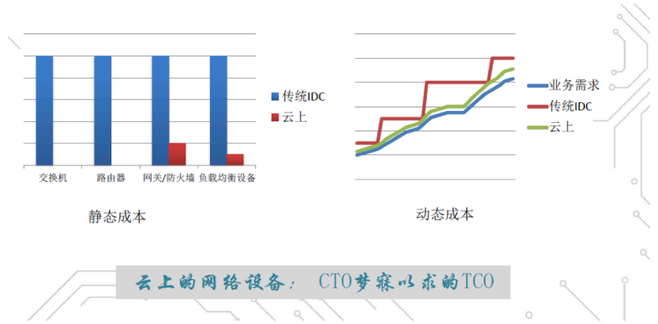
之后的专线接入其实还是需要找运营商把自己的IDC和阿里云的数据中心接入点连接起来。这件事情依然是很烦恼和痛苦的，所以阿里云在专线接入方面推出了基于合作伙伴的方式：如果用户租用的机房是阿里云合作运营商的，就可以复用一根已经和阿里云连接好的线路，这样时间大大缩短了，成本也将大大降低。

之前讲了很多是为了让大家了解在物理世界里面将整个企业的网络架构组建起来需要哪些步骤，对应地在云计算平台上实现这些步骤有哪些成本优势。

接下来具体看看VPC的成本优势，先说数据中心的成本，主要是网络设备的成本。这张图表达意思很明确，红色部分是在云上的成本，这项成本基本上没有，因为在云计算平台上用到的路由器和交换机是免费的，只有用到一些带有超大规模能力网关或者增值服务的SLB实例的时候才会需要付费，而这个价格与传统世界中购买相应能力的设备的投入相比而言是很少的。

云上组网结构与成本





网络设备成本

提到云计算这项技术对于成本的优化不得不提动态成本。假设图中蓝线是实际业务需求，有时增长快有时增长慢，也可能某段时间比较平稳；红线是在物理世界用传统的方式组建网络来满足业务需求的成本曲线。这条曲线有几个非常明显的阶跃，假想一下数据中心里面有价值8千元的核心交换机来支持核心业务，但是随着业务增长这个交换机无法满足业务需求，而市面上下一个规格的交换机可能就是2万或3万的，即使有9千或者1万的一般也不会选，因为我们不希望投入1万元使用两个月又需要升级，所以此时就会选择将路由器升级为2万的，几个月后又将升级为5万的。

而这几次升级中间有非常长的时间内硬件能力是浪费的。而在云计算平台上用户所花的钱都是与用户实际需要的资源成正比的，换句话说，云计算让用户的IT架构从面向预算变成了面向实际需求，使用户几乎是按使用付费。这就是为什么提到云计算对成本优化不得不提的动态成本优化。

对于公网带宽部分，我们先了解一下传统IDC世界和在云计算平台上购买公网带宽的区别，之后分享一些具有实战意义的省钱的技巧。

首先我们没有直接比较在运营商那里购买带宽和在云平台上购买带宽的价格，这是因为这两种方式的计费技术是不同的。在传统IDC世界中经常需要接一根联通的线再接一根电信的线，而在阿里云却不需要。云计算平台提供的公网带宽一般叫做BGP带宽，这是一个网络动态路由协议的名字，大家可以忽略这些技术细节，只需要记住一个事情就是BGP带宽有一个非常明显的特征就是无论来自什么运营商的终端用户来访问架在BGP带宽上的服务都会很快。所以这两种带宽在技术栈是有区别的，无法直接比较其绝对价值，但是动态成本还是可以比较的。在传统的IDC里面想找电信接一个1G的带宽线路，就需要保底1G签一年合同，当扩容的时候也需要提前告知运营商扩容周期。如果部署计划变更了，准备弃用这个机房搬到另外一个机房去，那么在原本机房购买的带宽的投入也就白费了。

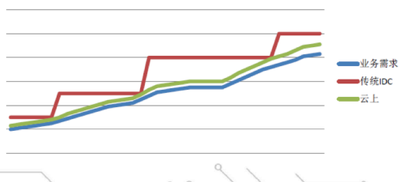
但是在云计算平台上可以做到今天买了30M，明天可以升到100M，而后天降到5M，这都没有问题，这就是所谓的按需付费。

说到按需付费，我们来讲一下计费方式选择，这个是大家在实际使用云计算特别是Iaas层的产品包括虚拟服务器ECS，负载均衡SLB以及弹性公网IP等等这些资源的时候在公网付费方式部分都需要去选择的，我们需要选择是按带宽还是按流量的计费方式。

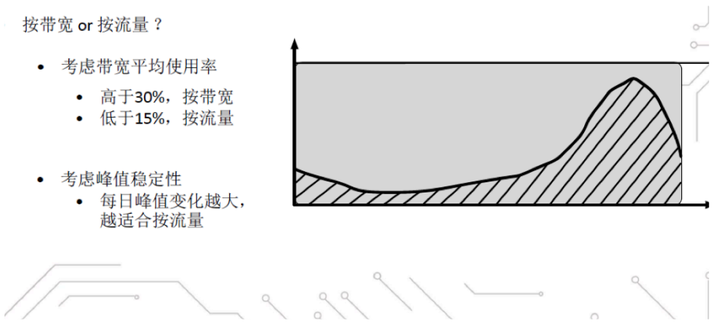
在云计算产品计费方式的选择上，包括阿里云和其他友商的用户按流量计费的居多。如果大家在上云之前对友商进行了调研的话会发现，特别是在美国，基本上看不到按照带宽付费的方式，其实只有国人特别喜欢使用按照带宽付费的这种方式，大家可以体会一下到底为什么。

接下来分享一下具体该怎么选。第一个需要参考的因素就是带宽的平均使用率，其实这些选择方式在阿里云产品文档里有专门的详细介绍，在这里还是大家详细说明一下。这张图是很明显的互联网业务的曲线，上午和中午这段时间流量比较低，但是到了午后流量就会上升，晚上的黄金时段流量也比较高，过了凌晨就会下降。首先假设按照带宽去购买计算一下需要多少带宽，假设图中横线的高度就是需要购买的带宽，这里一般而言要比实际的流量的最高峰高一点，因为互联网业务有一个显著的特征就是永远无法控制用户的行为，很可能某个时间段就会出现很多用户集中使用。

- 云计算省钱在什么地方
  - 单线带宽 VS BGP
  - 动态成本
    - 合同签署期限 VS 按需购买



公网宽带成本



公网带宽成本 — 计费方式选择

为了有弹性余量所以需要多买一些带宽，灰色的面积就是带宽乘以一天24小时的时长计算一下，当带宽利用率是百分之百时需要多少流量，阴影面积是业务实际使用的流量，使用阴影面积除以整个灰色的面积就按照带宽方式购买时的带宽综合利用率。简单的结论就是如果计算出的结果是带宽综合利用率高于30%的话，建议使用按带宽的付费方式；如果是低于15%的话建议选择按流量付费。

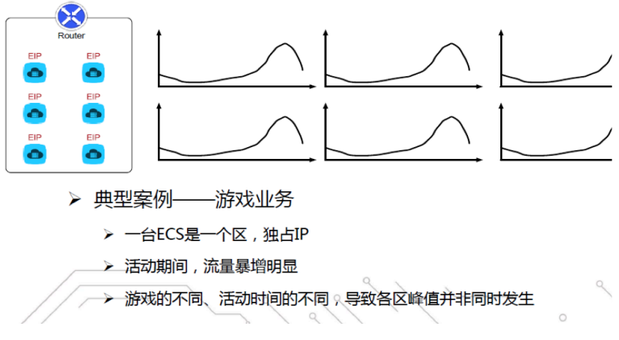
而对于中间这个阶段而言，怎样选择最划算呢？其实这是因人而异的，因为不同的业务还有其他不同的特征。比如说一个特征叫做峰值稳定性，就是假如说今天或者本周每天的峰值是100M，但是下个月有促销活动，一下就会上升到300M，而再下个月只需要150M就够了。当业务有这种特征的时候建议也选择按流量付费。大家回头想一想为什么今天SLB上的业务按流量计费的多，而ECS上按带宽计费的多。

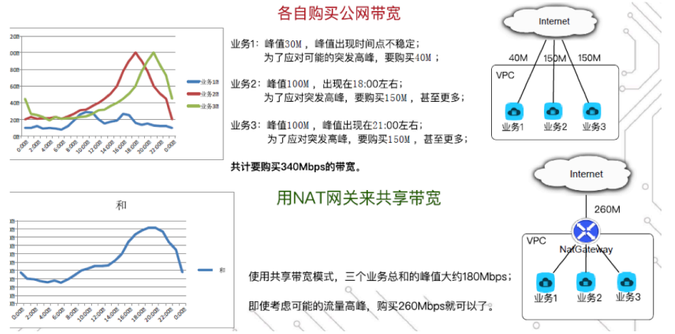
接下来和大家分享另外一个在带宽这方面和成本相关的技巧问题，也就是共享带宽。之前有很多用户向阿里云提出这样功能的需求，阿里云现在已经支持了。共享带宽是什么意思呢？假如说现在有一套业务的六台机器，每台机器都是这样的一个流量的曲线，最简单的方式就是为每一台机器购买一个IP还有带宽，因为刚才提到的带宽需要比流量峰值更加高一点，但是每台机器都多买很多的带宽会极大浪费。

这样的例子中最典型的就是游戏业务，因为在游戏里一台ECS就是游戏里面的一个区，它们都会独占一个IP，独占一份带宽。因为游戏业务本身的流量特征导致大多数情况下按带宽计费更加划算，所以也就需要为每一台服务器都多购置带宽，因而产生了大量的浪费。

共享带宽的好处是什么呢？为了让大家可以直观地通过数据感受一下我们假设了一个场景，假设有三个业务，而三个颜色的曲线代表着各个业务的流量。如果按照右边的图，为每个服务器都购买带宽的话，大概需要购买300多兆的带宽；假如说可以共享带宽，把左边三个曲线每一个点的带宽都加起来就会发现，它们的和产生的流量曲线在高峰也超不过200兆。如果支持这样的模式，也就是网关上面总共购买200多兆的带宽给不同的业务复用，这样就可以节省不少的资金投入。

公网宽带成本—共享宽带





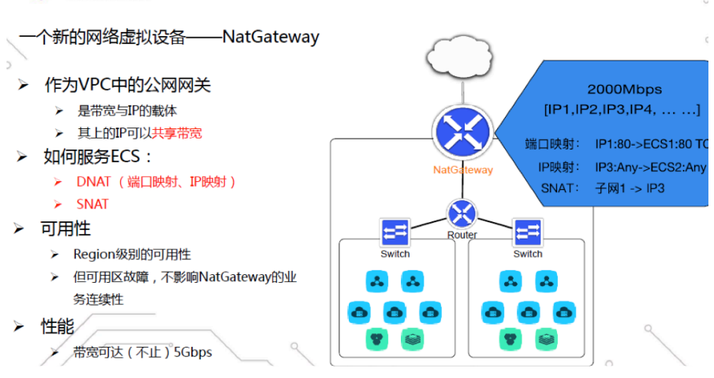
公网宽带成本—共享宽带

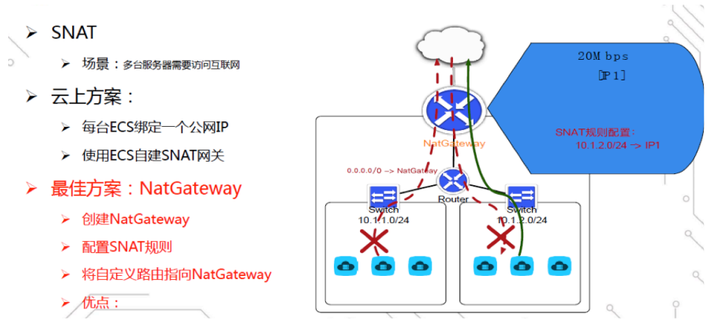
今天阿里云已经支持了这样的模式，不久之前新推出的一款产品叫做NatGateway，一句话描述它就是VPC里面的公网网关，非常容易理解。用户可以在公网网关上买IP和带宽，买到的带宽是为公网网关上所有的IP共享复用的，用户可以使用这些IP做DNAT和SNAT给后端的服务器使用，当然可以实现一台服务器独占一个IP或者很多个服务器独享一个或者一组IP，可以实现端口级别的映射也可以实现IP级别的映射，也可以实现SNAT这样一种典型的抓取互联网内容的架构方式。

只要是企业级用户都会关心可用性和性能这样两个问题。NatGateway的可用性是区域级别的，换句话说在同一个区域内挂掉一两个是不会造成大的影响的。至于性能方面，它的带宽不只5G，如果有哪个用户想要购买10G带宽问能不能支持，阿里云也会非常欢迎。

从技术架构上讲，阿里云Nat网关是彻底的分布式实现，其物理的吞吐量是没有上线的，而是只取决于部署的规模。

公网宽带成本—共享宽带





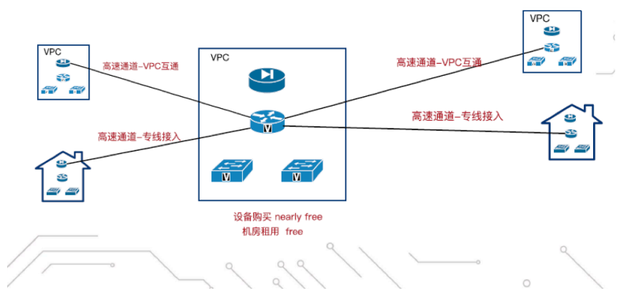
公网带宽成本 — one more thing

提到这里就再加一个东西就是SNAT，今天有非常多的用户向阿里云提出这样的需求。什么意思呢？就是说今天在服务器端，经常需要更新软件或者抓取互联网内容，所以经常需要服务器去主动地访问互联网，简单的架构方式是每个服务上都挂载一个公网IP，这个方式看上去实现比较简单，但是实际上管理的复杂度和安全风险都会很难把控。SNAT就是让没有公网IP的服务器使用网关的IP去访问互联网，其实今天大家每天都在使用这个功能，当我们手机连入WIFI的时候，手机是没有公网IP的而只有一个私网IP，而为什么能访问互联网就是因为实际上手机向外部发送数据包的时候在WIFI网关那里做了一次SNAT，只不过大家感知不到这个技术的存在。而当做自己的企业架构的时候就会出现这样的需求，阿里云的SNAT近期也将会发布，也会支持在Nat网关上，在VPC产品页面上将会有申请入口，希望大家能够关注。

讲完了南北向公网带宽成本的分析和使用技巧，我们进入最后一部分内容，就是东西向的组网成本，还要将这张图翻出来，注意这张图里面四个红色的地方实际上就是今天在云计算平台上的进行东西向组网的四个最基本的场景，分别是本地的VPC互通，异地的VPC互通，本地的专线接入也就是混合云的模式和异地的专线接入。

做一个简单的对比，把传统IDC当做比较的标地，今天云计算技术的其他厂商也当做比较的标地，还有就是阿里云以及通过阿里云的合作伙伴进行接入，通过这四种方式在上述四种场景之下支持的能力和成本情况的比较，大家就能形成一个非常直观的印象。

东西向组网成本



东西向组网成本

图中最左边的传统IDC只有两根柱子，换句话说传统IDC上没有所谓的专线接入这个概念，而只有所谓的同城专线和跨域专线。上面的一万或三万这样的价格是大致拟合出来的用户一个月需要为线路花多少钱买单。当然对于不同城市，不同距离不同运营商的价格也会有所不同，这里只是给出一个大概的量级。

其他的大多数云厂商基本上都会支持同地域的VPC互通，但是除了阿里云之外基本上没有云计算厂商支持跨地域的VPC互通，因为这中间有非常高的技术门槛。而且它们的本地和异地的IDC接入非常贵，因为这个钱不只是云计算平台收取的，大部分还是运营商收取的。

阿里云对于以上所有的这些场景都是支持的，只要用户需要就可以买到。至于成本可以横向比较一下，阿里云基本上在每一个场景下都具备价格优势，所以今天技术优势不仅仅是阿里云的主打点，用户在价格和成本上依然可以在阿里云上找到实惠。

而最后一个是通过阿里云的合作商去完成两种不同的专线接入场景，就像刚才说的如果用户租用的机房是阿里云的合作伙伴的机房，就能以非常低的时间成本和资金成本去完成这样的专线接入从而实现融合云架构。

回顾一下，我们对比了传统IDC的花费也就是虚拟设备和网络设备的花费，南北向就是公网带宽的花费，东西向就是跨机房间进行私网互通和生产网络互通的花费。通过分享大家应该对于企业级网络架构有了初步的概念，并且对于每一步的资金投入有了大致概念，最后希望大家的企业发展越来越快，产生这样的需求。因为有这样需求就意味着企业在一步一步长大，业务价值也在一步步增长。最后祝大家产品使用愉快，业务发展顺利。

专有网络VPC

创业十八般武器

朋友圈每天都有得了“创业病”的小伙伴，他们不是在创业的路上，就是在提辞职报告的路上，前赴后继带着“梦想还是要有的，万一实现了呢？”的人生信条雄赳赳、气昂昂地走在创业的康（bu）庄（gui）大道上。

于是，当你决定创业，加入全国每天新成立的数千家创业企业中，成为“创始人”的一员。在此之前，不妨先问自己几个小问题：你知道创办公司的各种手续吗？你知道怎样同BAT、FLAG出身的创业者竞争吗？你知道如何快速获取新的流量、如何把现有的流量做到极致吗？

初创企业，如何轻装上阵，如何快速试错，如何稳妥立足，如何筹钱、组队、捕获第一批核心用户等是创业者们最为关心的问题。如果能玩转以下的创业十八般武器，则可能让企业的发展事半功倍。

创业十八般武器之一：公司注册

创业的第一步自然是注册公司，但你了解注册公司的流程吗？

公司注册流程依次为：核名（准备5个以上名称到工商局确定公司名字）→开验资户→验资（完成公司注册资金验资手续）→签字（客户前往工商所核实签字）→申请营业执照→申请组织机构代码证→申请税务登记证→办理基本帐户和纳税账户→办理税种登记→办理税种核定→办理印花税业务→办理纳税人认定→办理办税员认定→办理发票认购手续。

是不是已经感觉头昏眼花了？另外根据工商局最新规定，注册后会邮寄信件到公司地址，如果退信，工商局会将公司列入异常名单！

觉得用个人住宅注册公司就可以？你要知道，个人住房注册公司需要缴纳房产税：房产登记价*70%*1.2%。北上广一个500 万房子要交 4.2万元一年，在这里被坑的创业者大把大把的！

福利来了，如果你想跳过这些麻烦事儿，请看创业十八般武器之一：公司注册

准备好你和合伙人的身份证，就可以开始注册公司，坐等一证五章到手，创业快速开始。

创业十八般武器之二：人员招募

有了公司还要什么？招人啊！如何找到最顶尖最合适的人才始终是创业者最头疼的问题。

尤其是近年人力成本快速上升，五险一金费率达到税前工资总额的66.33%，以北京税前工资1万元计算，员工在扣除他个人部分的五险一金和个税后，实际的到手工资7454元，而企业实际支出14410元，因为公司为该员工缴纳企业部分的五险一金需要4410元，还不包括招聘成本、培训成本、离职成本等。

我们都知道场地（入住孵化器/在小出租房里挤挤更健康）、设施（自带电脑、卫生纸）可以节省开支，但人员方面其实也可以做好节流、节约成本。人才共享，即云工作方式，放弃使用全职员工，将需求临时众包给经验丰富，更加专业的自由工作者远程办公。

并且，众包平台会免费提供项目金托管、项目监控管理、即时沟通工具、项目经理全程跟进和争议处理等服务，项目进度、质量和资金均有足够的保证。还等什么，现在就用创业十八般武器之二：众包平台，尝试低人力成本、产品快速上线的创业之路吧。

创业十八般武器之三：建设网站

在有了公司、人员、产品后，你还缺一个优秀的官网展示自己、售卖商品。怎么才能绕过建站商家的套路？以下五点你不速度get，怎么一展你的宏图大志？

1、安全的空间、域名

电商网站对于安全性的要求远高于其他平台，所以在空间、域名的选择上一定要慎之又慎，一定要找国内知名的服务提供商，让稳定性、安全性得到最优质的保障。

2、判断建站公司的实力

国内建站公司多达上百家，质量良莠不齐。一家靠谱的开发公司很大程度上决定了网站的后续发展空间，技术团队是否齐全、有没有知名公司案例、售后规模实力都是需要关注的重点。

3、避免层层外包

有些财大气粗的公司，从最初的网站建设，到页面设计，再到运营推广……每一个环节都外包给第三方来干，但往往付出惨痛的代价，应该尽量避免层层外包。优秀的员工加上良好的控制，一定能够让电商网站在短时间内获得转化。

4、需求明确合理

明确合理的需求是电商建站的第一步，你越是清楚地知道自己想要什么，程序员也就能更加明确什么样的站点是你需要的。

5、酌情获取源码

对于小型企业来说，在初创阶段可以不必急于购买源码，

初创企业，如何轻装上阵，如何快速试错，如何稳妥立足，如何筹钱、组队、捕获第一批核心用户等是创业者们最为关心的问题。如果能玩转以下的创业十八般武器，则可能让企业的发展事半功倍。

等企业成长稳定了，再去考虑系统的源码。对于中大型企业来说，购买系统的时候，必须去购买源码，做到对整套系统的详

尽了解，避免高昂的二次开发费用和拖沓的开发进程。

以上五点请谨记在心，定不会让你在建站时吃亏！别以为我能帮你的只有这么多，我这还有创业十八般武器之三：一站式电商建站解决方案：

一套可以满足运营商、供货商、采购商、用户分销等，在PC与移动设备上都能够兼容运行的完整电商生态解决方案，让更多企业迅速创建自己的电商王国。

只需要企业展示也不要紧，云市场9元建站活动，一顿午饭的价格就能拥有自己的企业网站，还等什么？

9元建设网站：https://market.aliyun.com/tpldns

创业十八般武器之四：ICP证件办理

万事俱备，就差上线？慢着，根据《互联网信息服务管理办法》和《关于互联网信息服务办理许可与备案的通知》，经营性网站必须办理ICP证，否则就属于非法经营。

根据管理办法，有违法所得的，没收违法所得，处违法所得3倍以上5倍以下的罚款；没有违法所得或者违法所得不足5万元的，处10万元以上100万元以下的罚款；情节严重的，责令关闭网站。未在其网站主页上标明其经营许可证编号或者备案编号的，由省、自治区、直辖市电信管理机构责令改正，处5000元以上5万元以下的罚款。

办理经营性ICP经营许可证则需要满足以下条件：

（1）经营性ICP经营许可证须在省、自治区、直辖市范围内经营的，注册资本最低限额为壹100元人民币；在全国或者跨省、自治区、直辖市范围经营的，注册资本最低限额为1000元人民币。

（2）经营性ICP经营许可证须有必要的场地、设施及技术方案。

（3）办理经营性ICP经营许可证的公司属于全内资企业（注册资金不能有外资）

还需要提供以下材料：

公司营业执照副本、公司法人身份证、公司股东身份证、股东公司提供营业执照副本、公司章程、公司股权结构图、验资报告和审计报告原件、房屋租赁合同及出租方房产证明原件、公司近期为员工所上社保证明、已设立分公司营业执照或控股子公司营业执照复印件及相应章程、公司域名注册证书、公司行业主管部门前置审批文件

如果你想快速办理ICP，请戳：创业十八般武器之四：ICP证件办理

创业十八般武器之五：数据库优化

于是你注册了公司，开发了产品，上线了网站，享受着从0到1的痛并快乐着的过程。于是你从从无人知晓，到成为行业最大的公司之一。

然而，快速的发展，让系统服务器尤其是数据库经历考验：核心数据库压力大、核心MySQL数据库数据量过大、系统架构设计不合理、数据库请求过于集中，当然，你可以用高配置掩盖技术架构问题，但解决问题，才会满足现在和未来的公司发展需求。

那么，如何解决问题呢？

短期解决方案：先满足当前性能需求。

1、对重点数据表进行优化

2、数据库读写分离

3、慢SQL优化和索引调整

4、调用次数高SQL采用缓存

长期解决方案：满足未来2-3年的业务发展目标

1、缓存设计、页面渲染、前后端交互

2、业务模块垂直拆分

3、数据库水平拆分

4、建立数据库开发规范

5、严格的数据库上线流程

7、设计数据库割接迁移方案

问题来了，要知道一个专业的DBA成本不菲，如果自己的技术人员不足怎么办？创业十八般武器之五：数据库优化

数据库优化服务（Oracle性能优化/MySQL性能优化/SQLServer性能优化）

袋鼠云数据库专家核心成员来自原淘宝DBA团队，云

上数据库出现性能瓶颈影响业务，综合考虑系统资源，数据库架构设计以及系统压力等因素，分析性能瓶颈原因，提供有针对性的优化和扩容方案建议

原淘宝DBA团队为你分析性能瓶颈原因，提供有针对性的优化和扩容方案建议，淘宝都扛得住，你也能扛得住！

至此，创业的第一步你已经顺利走过，懂得利用工具的你已经站在了其他创业者身前，但从0到100的过程中仍然需要你披荆斩棘。下期创业十八般武器，我们继续为你带来更多加快效率、减少成本的指南，黑客增长、创意营销、协同办公……还有什么你想了解的？邮件至mkt.marketing@list.alibaba-inc.com，或许下期的武器中就有你！

夏炙
阿里云运营专家

负载均衡（SLB）

健康检查的使用误区和最佳实践

隽勇
阿里云技术专家

分析用户提交的负载均衡（简称 SLB）工单，我们发现很多SLB异常问题与不合理的健康检查策略相关。例如：

- 健康检查间隔设置过长，SLB无法及时准确发现后端ECS出现服务不可用，造成实际业务中断。
- 使用HTTP模式健康检查，未合理配置后端日志记录模块，导致后端日志内容记录大量健康检查HEAD日志，造成磁盘负载压力。

原理要点

SLB服务通过健康检查机制检测后端云服务器池中ECS的健康状态，自动隔离异常状态的ECS，从而解决了单台ECS的单点问题，同时提高了应用的整体服务能力。关于健康检查的原理，详情请参考官网知识点，

- 负载均衡健康检查原理浅析

文章重点阐述了如下几个方面：

- 健康检查处理过程
- 健康检查策略配置说明
- 健康检查机制说明
- 健康检查状态切换时间窗

总结要点如下：

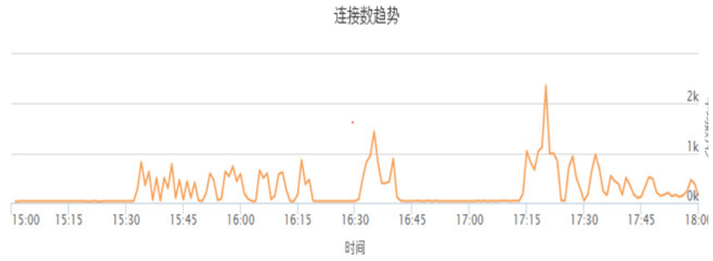
1、4层TCP协议的健康检查通过TCP 3次握手检查后端 ECS 端口是否处于监听状态。与常规TCP连接使用TCP FIN方式销毁不同，健康检查的TCP连接在3次握手建立成功后，发起方以TCP Reset方式主动将连接断开。

2、7层HTTP协议的健康检查通过HTTP HEAD 请求检查指定 URL 路径的返回值，如果返回值与自定义的健康检查成功返回值相匹配，则判定检查成功。

3、SLB使用 4层 TCP 协议监听时，健康检查方式可选 TCP 协议 或 HTTP 协议。

4、4层 UDP 协议的健康检查是通过ICMP Port Unreachable消息来判断，可能存在服务真实状态与健康检查不一致的问题，我们后续会持续产品改进解决该问题。

5、健康检查机制的引入，有效提高了承载于SLB的业务服务的可用性。但是，为了避免频繁的健康检查失败引发的切换，对系统可用性带来的冲击，健康检查只有在【连续】多次检查成功或失败后，才会进行状态切换（判定为健康检查成功或失败）。



使用误区

如下是通过工单梳理总结的健康检查常见的理解和使用误区。

误区1 ：怀疑SLB健康检查形成DDoS攻击，导致服务器性能下降。

例如，有用户误认为 SLB 使用上百台机器进行健康检查，大量高频次的 TCP 请求会形成DDoS攻击，造成后端ECS性能低下。但实际上，无论是4层TCP/UDP或7层HTTP协议的健康检查，都根本无法形成DDoS攻击。

原理上，SLB集群会利用多台机器 (假定M，个位数级别) 对后端 ECS 的每个服务监听端口 (假定N)，按照配置的检查间隔(假定 O 秒，一般建议最少 2 秒) 进行健康检查。以4层 TCP 协议为例，每秒的TCP连接建立数为：M*N/O。

因此，健康检查带来的每秒TCP并发数，主要取决于创建的监听端口数量。除非监听端口达到巨量，否则健康检查对后端 ECS 的 TCP 并发连接请求根本无法达到 SYN Flood 的攻击级别，实际对后端 ECS 的网络压力极低。

误区2 ：用户为了避免 SLB 健康检查的影响，将健康检查间隔设置很长。

该配置会导致当后端ECS出现异常时，SLB需要较长时间才能侦测到后端ECS不可用。尤其当后端ECS间断不可用时，由于SLB需要【连续】检测失败时才会移除异常ECS，这会导致SLB集群可能根本无法发现后端ECS不可用。

举一个实际案例：

背景信息：

用户使用后端有2台ECS的1个4层 TCP协议的公网SLB，对外提供服务，TCP协议的健康检查间隔设置为50秒。某天用户做了应用代码改进后，发布到1台测试ECS，而后将该ECS加入SLB。但是由于应用问题以及不合理的健康检查的设置，出现了连续2次的业务不可用。上图失败连接数趋势”代表SLB转发到后端ECS但无法建立连接的TCP 3次握手失败的连接数。在SLB正常工作时其值应该为0。

从图中可以看到，出现了2次SLB转发失败，业务间断不可用的情况。

阶段1：15:29 – 16:44：该问题由于应用代码的原因导致。

阶段2：17:12起：该问题由于ECS内核参数的配置错误，单台ECS无法承担业务量导致。

如下是复盘的详细过程：

【15:26】 用户创建ECS机器(假定名称i-test)，部署最新的应用代码，加入SLB集群中。

【15:29】由于新应用代码的问题，SLB将请求转发至该ECS后，如下图，使其CPU飙升至100%。

前图”失败连接数趋势”显示15:30分SLB的TCP转发失败连接数开始飙升，原因就是新加入的ECS i-test机器在CPU 100%的情况下出现业务不可用。15:34分开始，后端的ECS健康检查日志中出现健康检查失败的信息。但由于健康检查的间隔为50秒，而后端ECS i-test也不是完全不响应，偶尔的TCP

三次握手健康检查还可以成功，因此未出现【连续】健康检查失败，所以该SLB的监听端口未报出异常状态。

用户接到终端客户反馈业务出现断续不可用的情况后开始紧急自查。但由于SLB状态未报出异常，用户将排查集中在应用侧。

【16:44】用户将ECS i-test从SLB中下线，业务恢复正常，从前图”失败连接数趋势”中，可以看到失败的数量降到0。

【16:56】用户将装有老版本应用的ECS i-test 加入SLB集群，后端3台ECS正常提供服务。

【17:12】业务恢复后，用户认为之前问题是由于业务代码的问题导致。此时用户通过将ECS权重设置为0的方式将2台ECS下线，由新上线的那台ECS i-test承担所有业务。由于业务量陡增，ECS i-test机器未合理配置内核TCP参数，IP Conntrack表耗尽，前图”失败连接数趋势”显示大量新建TCP连接失败。但由于健康检查间隔高，SLB无法及时判断后端ECS实际状态，SLB日志显示该ECS的状态在很长一段时间内保持正常，当前端用户反馈后，才发现实际业务影响，导致业务中断时间较长。后续通过修改ECS 内核TCP配置参数解决。

从该示例可以看出，合理的配置健康检查间隔对于及时发现业务不可用非常重要。

误区3 ： 使用HTTP模式健康检查，未合理配置后端，导致后端日志内容记录大量健康检查HEAD日志，造成磁盘负载压力。

我们确实遇到过一例健康检查配置导致后端ECS的性能低下的案例。用户购买低配(1核1G) ECS，使用7层HTTP健康检查，健康检查间隔低，同时配置多个监听，后端ECS的Web服务记录大量的HEAD请求日志，导致IO占用高，引起性能低下。

解决方案请参考知识点”健康检查导致大量日志的处理”。

最佳实践

结合SLB健康检查的技术要点和实际使用误区，我们提供如下最佳实践。

1、根据业务需要合理选择4层TCP 或 7层HTTP的健康检查模式。

如果使用HTTP 健康检查模式，请合理配置后端日志配置，避免健康检查HEAD请求过多对IO的影响。

HTTP健康检查请不要对根目录进行检查，可以配置为对某个HTML文件检查以确认Web服务正常。

2、合理配置健康检查间隔

根据实际业务需求配置健康检查间隔，避免因为健康检查间隔过长导致无法及时发现后端ECS出现问题。具体请参考：

健康检查的相关配置，是否有相对合理的推荐值?

3、合理配置SLB架构，监听、虚拟服务器组合转发策略

为满足用户的复杂场景需求，目前SLB可以配置虚拟服务器组和转发策略。详情请参考文档：

- 如何实现域名 / URL 转发功能

对于虚拟服务器组、转发策略，其与监听对健康检查并发的影响相同，如左侧下表：

建议用户根据实际业务需求，合理配置监听、虚拟服务器组和转发策略。

a、对于4层TCP协议，由于虚拟服务器组可以是由后端ECS不同的端口承载业务，因此在配置4层TCP的健康检查时不要设置检查端口，而”默认使用后端服务器的端口进行健康检查”， 否则可能导致后端ECS健康检查失败。

b、对于7层HTTP协议，使用虚拟服务器组合转发策略时，确保健康检查配置策略中的URL在后端每台机器上都可以访问成功。

c、避免过多的配置监听、虚拟服务器组和转发策略，以免对后端ECS形成一定的健康检查压力。

维度	健康检查配置	健康检查目标服务器
后端服务器	使用配置监听时的健康检查配置	所有后端 ECS
虚拟服务器组	使用配置监听时的健康检查配置	相应虚拟服务器组包含的服务器
转发策略	使用配置监听时的健康检查配置	相应虚拟服务器组包含的服务器

不同文档下健康检查的对比

ECS Windows服务器

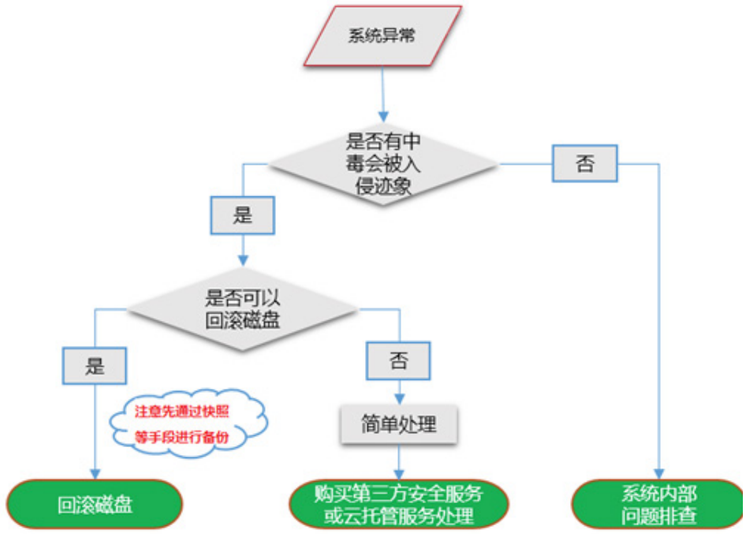
中毒或被入侵的快速诊断

五贤
阿里云售后专家

当业务迁移上云后，会面临更高的安全挑战。特别是对于 Windows 服务器，由于攻击门槛低，可能会遭遇病毒或被入侵等安全风险。系统中毒或被入侵后，通常会导致系统报错、负载异常、无法远程、业务不稳定等诸多不可控问题。本文结合阿里云大量相关案例的处理经验，介绍如何快速的诊断和处理该类问题。

一、处理思路

判断系统是否有中毒或被入侵的迹象，是此类问题处理的关键。完整的处理思路如下图所示：



二、快速诊断方法

可以结合如下三种方式进行快速诊断分析，用以判断系统是否有中毒或被入侵迹象。本文不会涉及更多深入的安全排查和诊断方法的讨论。

1. 可疑文件诊断分析（4W1H）

可以通过如下 4W1H 法来对可疑文件进行快速诊断分析。

注意：操作前，请先设置 显示所有文件（包括受保护的系统文件） 和 显示文件

后缀，相应配置方法本文不再详述。

What — 文件类型是什么？

病毒或木马文件一般都为可执行文件。所以，排查时，请重点关注后缀为如下类型的文件：

- exe： Windows可执行文件
- bat： 批处理文件
- com：DOS可执行文件
- vbs： Windows脚本

说明：这只是一种简单的判断方法，因为文件后缀可以随意修改，无法真实反应出文件类型。

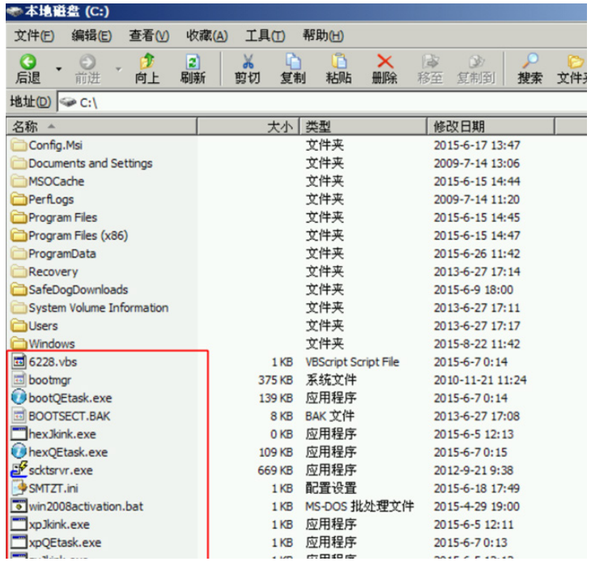
Where — 文件在哪？

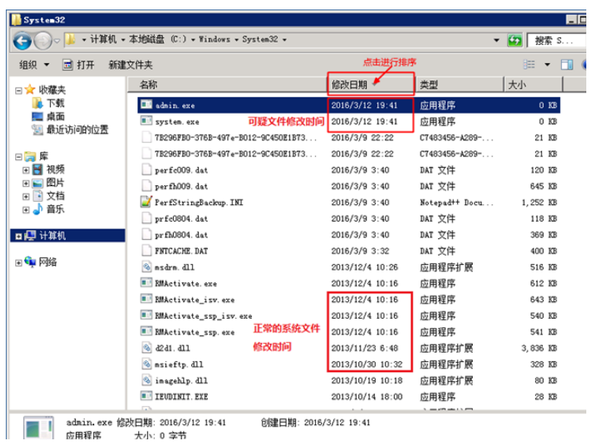
病毒或入侵者通常会在系统目录生成病毒或木马文件，并设置隐藏属性。诊断时，可依次到如下目录排查可疑文件（再次提醒，注意显示隐藏文件）：

- C:\ 根目录
- C:\Windows 目录
- C:\Windows\System32 目录
- C:\Windows\Temp 目录
- C:\Users\<用户名>\AppData\Local\Temp 目录（用户缓存目录）

When — 文件是什么时候创建的？

病毒或木马是在系统创建之后修改或生成的。所以，可以通过对比文件的【修改时间】来甄别可疑文件。如下图所示，进入前述系统目录后，点击【修改时间】列，按修改时间进行倒序排序，然后对比正常系统文件的修改时间来甄别可疑文件。





间来甄别可疑文件。

Who — 文件名是什么？

常见病毒或木马文件的文件名可能具有如下特征。可以据此甄别可疑文件：

- 随机字母和/或数字组合：病毒或木马通常会自动复制，然后在系统目录使用随机字母和/或数字生成副本。比如：2g-pjNw.exe，kuUYjp.exe，zJ5NQD.exe 等。
- 易与系统文件混淆：病毒或木马为了伪装，可能会使用与系统近似的文件名来混淆用户。比如：exp1orer.exe（正常文件名为 explorer.exe），svch0st.exe （正常文件名为 svchost.exe）等。

How — 如何综合判断？

最终，如本页顶部图示，综合前述几个维度，可以使用一台正常服务器做对比，在相关系统目录下，根据文件类型、文件属性（是否隐藏）、文件修改日期、文件名称等因素，共同分析判定可疑文件。

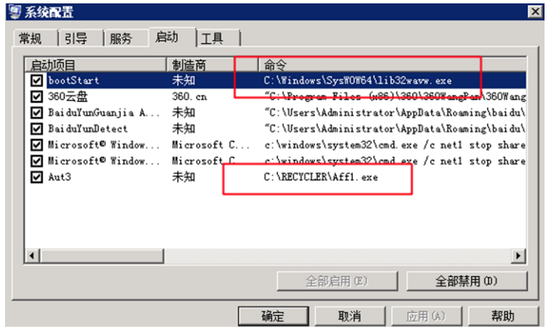
2. 可疑启动项诊断分析

病毒或木马通常会将自身配置为开机自动启动。可以通过如下方法来排查可疑启动项。

- 使用 msconfig 实用程序分析

通过 Windows 系统自带的 msconfig 实用程序，可以查看系统自启动项。

用法：开始菜单 > 运行 > msconfig ，如下图所示，切换到【启动】选项卡，然后重点关注 制造商 和 命令 这两列。如果相应启动项的文件路径或文件名具有前文所述可疑文件特征，则可以尝试取消勾选相应启动项后重启系统做对比分析。

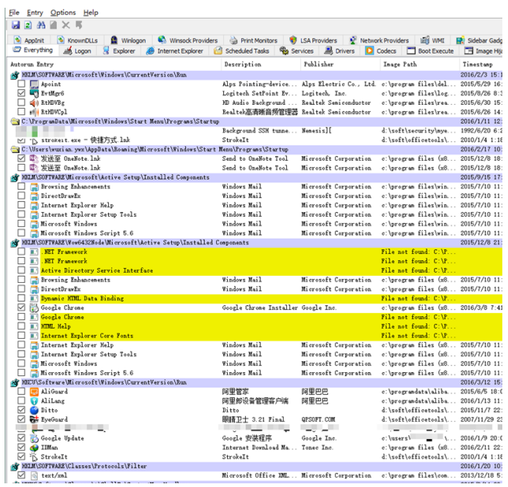


- 使用 Sysinternals 工具包中的 autoruns 工具查看启动项

微软官方免费提供的 Sysinternals 工具包 中的 autoruns 工具，能查看当前用户启动项、系统启动项、计划任务及浏览器挂载插件等更完整启动信息。如果问题服务器还能上传文件，则可以上传 autoruns 做更详细的排查分析。分析原则与使用 msconfig 一致。判定为异常启动项后，删除或取消勾选相应条目即可。

用法： 运行软件后，如下图所示，切换到 Logon 选项卡查看启动项。

- 通过注册表查看启动项



如果服务器内 msconfig无法正常运行，也无法上传 autoruns 工具，则可以尝试直接通过注册表分析可疑启动项。

注意： 请谨慎操作注册表，异常操作可能会引发系统无法启动甚至崩溃等严重问题。建议操作前先创建系统盘快照进行备份。

用法： 开始菜单 > 运行 > regedit，打开注册表编辑器，然后定位到Run注册键，64位系统Run注册键，Userinit注册键（默认值为userinit.exe），Explorer\Run注册键，RunServicesOnce注册键，RunServices注册键，RunOnce注册键，Load注册键等路径，查看是否存在可疑启动信息。分析原则与使用 msconfig 一致。判定为异常启动项后，删除相应条目即可。

3. 可疑进程及服务诊断分析

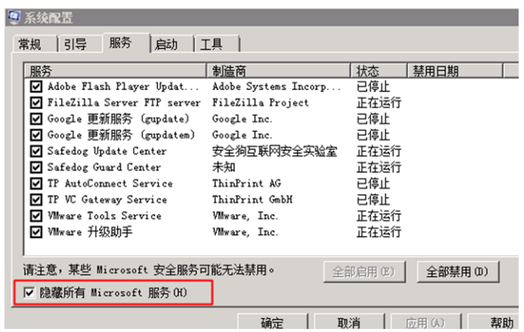
病毒或木马通常还会将自身注册为系统服务，以实现常驻内存和开机自启动。可以通过如下方法来排查可疑进程或服务。

- 通过 msconfig 剔除系统服务

如果服务器内 msconfig 实用程序还能正常运行，则可以通过其剔除系统自带服务，以便更快速的排查分析异常服务。

用法：如下图所示，运行 msconfig 后，切换到【服务】选项卡，勾选 隐藏所有Microsoft服务 复选框即可。

- 疑名称



和可疑文件的分析类似，也可以使用如下名称规则来甄别可疑进程或系统服务：

- 以随机字母和/或数字命名
- 易与系统服务或进程混淆的命名

三、处理方法

根据系统可控程度，如果出现异常时，还可以通过远程桌面或管理终端正常进入服务器，则可以参阅前文相关说明进行自助诊断分析。否则，则需要提交工单，反馈阿里云工程师，从后端通过 WinPE 工具盘引导服务器后做相关排查分析。

诊断分析后，如果确认系统存在中毒或被入侵迹象。则可以先通过如下方式进行简单处理，尝试恢复系统的远程访问、桌面显示等基础功能，以便更好的进行数据备份。

1、 创建备份目录，将可疑文件移动到备份目录。

2、 取消可疑启动项。

3、 重启服务器重新验证。

另外，由于无法明确被恶意破坏或篡改了哪些系统服务或文件。所以，为了保障服务器的长期稳定运行，如果经上述处理后系统仍然存在异常，则建议您先创建一个最新快照对系统盘进行备份，再通过回滚快照或初始化磁盘等方式进行系统恢复。

注意：回滚快照或初始化磁盘操作会导致数据丢失。

如果上述操作会对业务造成重大影响，需要做进一步的病毒查杀，则可以通过云市场等渠道购买第三方安全服务，做更深入的安全分析和处理。

艾瑞咨询

2016年中国云服务市场规模达520亿元

艾瑞咨询最新数据显示，中国云市场未来前景乐观，2016年中国云服务市场规模达520亿元，未来几年将保持30%的年复合增长率，在2019年将达到1182亿元的规模；云服务得到企业用户认可，有84.6%的企业用户表示未来将迁移至公共云；从公共云迁移的时间上看，当前的热点是视频云，未来政务、教育、医疗和金融是热点；另外混合云将成为主流，2016年71%的企业用户使用混合云，比去年增加了13%。

阿里云占中国公共云市场50%份额

12月13日，美国知名投行摩根士丹利发布报告，对中国云计算市场做出评估称，2016年中国公共云市场份额约20亿美元，其中阿里云占据约50%市场份额。凭借对公共云市场的绝对领导地位，大摩认为，阿里巴巴正在搅动传统企业级IT市场， 在中国市场上急速成长为IT巨头。

分析认为2016年阿里云在中国公共云市场上占据绝对主导地位，市场份额是AWS、Azure、腾讯云、百度云、华为云等市场追随者的总和。数据显示，阿里云在中国公共云市场占据50%份额，另外两家国际云计算服务商微软Azure和亚马逊AWS市场份额总和约10%到15%，而其他本土服务商并未在市场份额方面形成有效的竞争力。

同时分析认为，云计算行业需要大规模的早期投入。在中国市场上，阿里云成立于2009年，而腾讯、百度等在近几年才开始云计算业务，错失行业窗口期。华为基于此前在硬件市场的基础，目前主要提供私有云服务。

从2016年开始，中国市场对公共云的应用将会加速，并且拥有巨大的成长空间。此外，中国整体经济的放缓，也将刺激云计算应用加速，因为公共云能够为政府或者企业节省更多成本。大摩预计，阿里云2017财年的营收规模约10亿美元，到2019财年提高到30亿美元。

阿里云总裁胡晓明

荣登2016年度商业人物榜

在《经济观察报》于2016年最后一天发布的“2016年度商业人物榜单”中，阿里云总裁胡晓明与万达集团董事长王健林、顺丰速递CEO王卫、比亚迪董事局主席王传福等32位企业家一同登榜。

作为唯一入选的云计算企业领袖，胡晓明在接受记者专访时表示：“用煽情一点的话来说，阿里用电商、金融、物流本身的体温，在给中国和阿里云的客户提供技术的样板。”胡晓明说，“只有把自己的命押上，我们才能进一步驱动技术的变化，这个事情我们100%在做，但是没有几家企业是敢这么做的。” 作为国内云计算领域的龙头，阿里云已经迎来全球化布局时代，准备在与亚马逊、微软等国际强者的竞争中，树立中国品牌。

阿里云通过API经济激活第三产业新能量

《纽约时报》称，中国科技产业在一定程度上已领先美国，西方巨头如AWS等正转而从中国公司寻求新创意。

今年11月的Re:Invent全球合作伙伴峰会上，Amazon API Gateway与AWS Marketplace进行了集成。这类似阿里云云市场在今年6月发布的数据与API服务。第三方API公司可以通过云市场发布和货币化他们的API产品，从个人开发者、互联网企业甚至金融、交通等各行业的购买者中获取收益。

这迅速吸引了大量关注。据悉，5个月时间内云市场收到了上百家相关企业的入驻申请，目前在售的商品超过300款，每月成交额增长超2倍。

阿里云运营事业部总经理杨名透露，双11数据与API服务五分钟内突破百万成交额，单个商家当天在云市场的成交额超180万——对单价几分钱的API商品来说，这足以证明阿里云正成为API新生态的领跑者。

但阿里云的版图不仅限于此。蚂蚁金服、墨迹天气、-Face++、聚合数据……接连引入金融、电商、交通、民生、人工智能等领域的第三方API公司，似乎阿里云云市场不只满足于记录企业的每一次互联网心跳，也要为全域企业搭上互联网+的脉搏。

在摩根大通金融期货专家Rajeev Shah看来，拥有一个强大的API策略不仅仅是一个很好的“软件实践”（software practice），它也是一个强大的商业实践（business practice）。

FACEBOOK&ALIYUN

数据中心PUE对比

【新闻摘要】Facebook数据中心电源使用效率已经达到了1.06—1.08的水平，甚至比Google的数据中心能效还要高。在瑞典吕勒奥的数据中心园区内，Facebook实现了其Open Compute Project（OCP，开放计算项目）勾勒的服务器和数据中心设计，在这里服务器、供电和UPS都是定制的设计。公司称预计到2018年采用50%的清洁和可再生能源。

【评论】5年前，我国数据中心的平均PUE是2.5以上。阿里云2016年张北机房落成时，平均PUE可以达到1.25，最低可到1.13。

《中国桌面云标准化白皮书》正式发布

2016年12月28日，由中华人民共和国工业和信息化部信息化和软件服务业司指导、国家标准化管理委员会工业标准二部指导的电子技术标准化研究院主办的第六届中国云计算标准和应用大会在北京万寿宾馆召开。

这是2016年度云计算标准成果的盛会，本次大会解读了2016年度中国云计算标准化工作报告、重磅发布了1项国家标准化成果、1项联盟标准、3份白皮书、揭晓第四批云测评名单、中国优秀开源案例评选结果、首批联盟标准应用示范单位名单，以及第五届中国OpenStack黑客松成果。也是开源技术及社区的技术分享盛会，来自国际开源社区，技术组织，以及国内优秀企业在云计算标准与应用落地方面的深入实践和经验。

阿里云

订阅ASDN杂志

网站：www.aliyun.com　　联系人：pengfei.pf@alibaba-inc.com　　地址：北京朝阳区望京街道阿里中心望京A座