Sina App Engine架构

——云计算时代的分布式Web服务解决方案

■ 文 / 丛磊

Sina App Engine（简称SAE）是新浪研发中心于2009年上半年开始内部开发，并在2009年11月3日正式推出第一个Alpha版本的国内首个公有云计算平台（http://sae.sina.com.cn），是新浪云计算（简称浪云）战略的核心组成部分。

SAE作为国内的公有云计算，借鉴吸纳了Google、Amazon等国外公司的公有云计算的成功技术经验，并很快推出具有自身特色的云计算平台。SAE选择PHP作为首选的支持语言，Web开发者可以在Linux/Mac/Windows上通过SDK或者Web版在线SDK进行开发、部署、调试，团队开发时还可以进行成员协作，不同的角色将对代码、项目拥有不同的权限。SAE还提供了一系列分布式计算、存储服务供开发者使用，包括分布式文件存储、分布式数据库集群、分布式缓存、分布式定时服务等，这些服务将大大降低开发者的开发成本。同时又由于SAE整体架构的高可靠性和新浪的品牌保证，大大降低了开发者的运营风险。另外，作为典型的云计算，SAE采用“所付即所用，所付仅所用”的计费理念，通过日志和统计中心精确的计算每个应用的资源消耗（包括CPU、内存、磁盘等）。

总之，SAE就是分布式Web服务的开发、运行平台。

SAE的目标和发展

云计算在国外已经有4~5年的历史。2006年，Amazon就推出了以EC2为代表的公有云计算，并且实现了大规模盈利；2008年，Google推出了以Google App Engine为代表的公有云计算。国内的云计算却一直是炒得很厉害，各大互联网公司都在宣传，但真正有技术实力做出来而又对外公开使用的少之又少。

从2004年开始，新浪就开始了私有云方向的研究和实践，以此为基础的动态应用平台目前已经支撑新浪内部的绝大部分业务。从2008年起，新浪又启动了“浪云”的公有云计算计划，相继开发了分布式队列服务、P2P文件系统、分布式计算框架等一系列基础服务。实际SAE就是“浪云”战略的产物。

SAE从架构设计和代码编写开始，就明确了自身的两个目标：第一，做公有云计算平台，公有云不同于私有云，更强调安全性和可靠性，这也对整体的架构设计和技术实现提出了更苛刻的要求；第二，为分布式Web

表1　SAE和传统的虚拟主机托管VPS的主要区别

类项	SAE	VPS
核心用户	Web开发者	无核心用户
使用方式	服务使用	设备租用
目标	力争覆盖Web服务所有需求，提供多种服务给开发者使用	仅基本需求
SLA（服务承诺）	高可靠性及严格的服务承诺	依服务商变化，无严格协议
计费方式	所付即所用，所付仅所用	预付费，无精确计费

作者简介：
丛磊，资深架构师，SAE技术主管，擅长应用算法、编译器实现、分布式系统设计、人工智能博弈论等，2006年毕业后加入新浪，2008年开始带领技术团队从事云计算方向的研究与开发。

服务提供一整套的解决方案，SAE争取提供开发者开发Web应用过程中所用到的所有服务。

经过技术团队一年的开发，SAE目前已经提供了十多种服务，整体上分为计算型和存储型，计算型又包括同步计算和异步计算，而存储型则分为持久化存储和非持久化存储，如表2所示。

表2　SAE提供的服务

服务名称	类型	说明
HTTP+PHP	同步计算	带SAE沙盒的Apache和Zend为用户提供Web计算服务
Stor	持久化存储	提供分布式文件存储
Memcache	非持久化存储	提供分布式缓存服务
RDC	持久化存储	分布式数据库集群，提供MySQL服务
Taskqueue	异步计算	异步离线轻量级任务队列，HTTP方式调用
DeferredJob	异步计算	异步离线重量级任务队列，系统方式调用
Cron	异步计算	分布式定时服务
FetchURL	同步计算	分布式抓取服务
Tmpfs	非持久化存储	提供临时文件存储
Appconfig		提供应用配置功能，取代Apache htaccess
Mail	异步计算	邮件发送服务
Image	同步计算	图像处理服务
XHProf	同步计算	Facebook提供的强大的PHP调优工具
其他工具		
SDK		Windows GUI SDK、Linux/Mac command line SDK
Online SDK		在线代码编辑器

SAE在2009年11月3日发布了Alpha1版本，2010年2月1日发布了Alpha2版本，2010年9月1日发布了Beta版本，经过将近一年的不断完善和改进，尽管SAE一直没有开放注册（实际云计算的模式也不以注册用户的规模为评价标准），但已经拥有了一批有价值的App和粉丝开发者。截止10月1日，SAE拥有开发者4000多名，App总数3000个，活跃App将近1000个，每天独立代码部署行为超过1000次。

整体架构

SAE从架构上采用分层设计，从上往下分别为反向代理层、路由逻辑层、Web计算服务池。而从Web计算服务层延伸出SAE附属的分布式计算型服务和分布式存储型服务，具体又分成同步计算型服务、异步计算型服务、持久化存储服务、非持久化存储服务。各种服务统一向日志和统计中心汇报如图1所示。

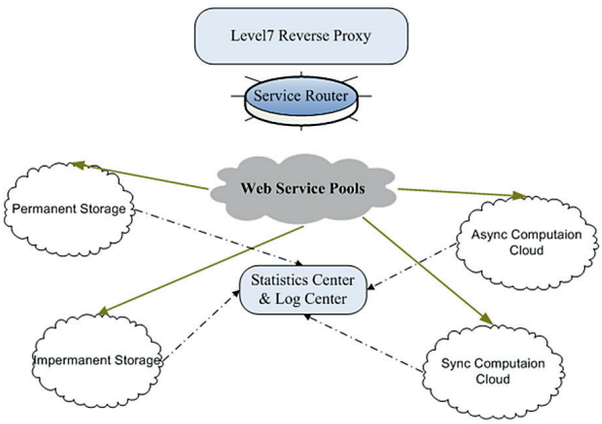

图1　SAE整体架构图

7层反向代理层：HTTP反向代理，在最外层，负责响应用户的HTTP请求、分析请求并转发到后端的Web服务池上，提供负载均衡、健康检查等功能。

服务路由层：逻辑层，负责根据请求的唯一标识，快速地映射（O(1)时间复杂度）到相应的Web服务池及相应的硬件路径。如果发现映射关系不存在或者错误，则给出相应的错误提示。该层对用户隐藏了很多具体地址信息，使开发者无需关心服务的内部实际分配情况。

Web服务池：由一些不同特性的Web服务池组成。每个Web服务池实际是由一组Apache Server组成的，这些池按照不同的SLA提供不同级别的服务。每个Web服务进程实际处理用户的HTTP请求，进程运行在HTTP服务沙盒内，同时还同样内嵌运行在SAE沙盒内的PHP解析引擎。用户的代码最终通过接口调用各种服务。

日志和统计中心：负责对用户所使用的所有服务的配额进行统计和资源计费，这里的配额有两种，一种是分钟配额，用来保证整个平台的稳定；一种是天配额，用户可以给自己设定每天资源消耗的最高上限。日志中心负责将用户所有服务的日志

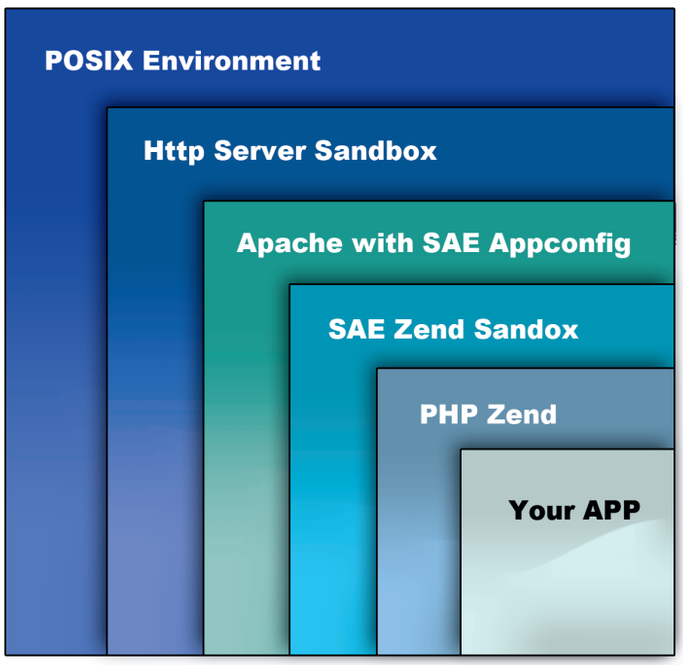

图2　SAE沙盒结构图

汇总并备份，并提供检索查询服务。

各种分布式服务：SAE提供几乎可以覆盖Web应用开发所有方面的多种服务，用户可以通过StdLib（可以理解为SAE PHP版的STL）很方便地调用它们。

真正的用户代码是跑在SAE提供的Web运行环境下的，为了提供公有云计算特有的安全性，SAE设计多层沙盒来保证用户应用之间的隔离性，如图2所示。

最内层的就是用户代码，大部分PHP代码不需要做任何修改就可以跑在SAE平台上，小部分代码需要做一些修改以适应SAE的平台特性。这主要有两部分：第一，SAE因为安全性禁用了本地I/O，所以fwrite等函数需要修改为使用Tmpfs读写本地临时文件或者直接通过Stor读写我们的分布式文件存储；第二，用户在SAE上不能通过Curl访问非新浪域的资源，用户有抓取公网资源等需求，需要修改为调用FetchURL服务。我们提供了PHP Wrapper以方便用户的修改。

PHP Zend为标准的PHP官方解释器，我们采用的版本为5.3.0。

SAE Zend Sandbox为一个逻辑概念，为用户的代码运行提供良好的隔离性。这里有两个层面，第一是通过标准的php.ini，我们设定了一些特殊配置和禁用函数；第二，为了达到一些php.ini无法实现的沙盒功能，我们对Zend解释器核做了一些改进，以便通过用户标识将资源进行隔离。另外我们还把一些SAE的特定服务也在Zend层做了融合。

Apache为标准的Apache Web Server，版本为2.2。不过我们禁用了htaccess，并提供了自己实现的替换方案AppConfig。用户可以通过类自然语言的方式编写AppConfig，如 - compress: if(out_header[“Content-Length”] >= 500) compress 表示按条件启动页面压缩。目前AppConfig提供的功能有：目录默认页面、自定义错误页面、压缩、页面重定向、页面过期、设置响应头的content-type、设置页面访问权限。我们选择自行实现AppConfig还有一个考虑，就是因为传统Apache的htaccess因为要按目录递归方式合并配置文件，效率不能满足SAE的需求。

HTTP Server沙盒为Apache的安全可靠运行提供了多种保护功能，比如防止某个用户恶意占用连接数从而导致整个Web服务不正常。

最外层的是标准POSIX环境，目前我们的服务跑在Linux 2.6上。

上面就是对SAE整体架构的概述，接着将详细讨论我们在架构设计上的一些具体考量点。

扩展性

扩展性是分布式系统的两个主要目的之一，SAE作为公有云计算，同样把服务的扩展性作为架构设计的重要指标，要求在用户增长、压力提升的情况下，可以实现自动的服务扩展，同样当压力降低时，可以将服务收缩，以节约资源，整个过程无需人工参与。SAE人工只需做好容量规划和管理。目前国外的公有云计算架构的扩展性主要有静态和动态两个思路。

静态扩展：用户和资源有强绑定关系。最典型的例子为Amazon的EC2和Ruby云计算平台Heroku，用户申请的资源和用户有严格的一对一关

系，换句话说，A用户申请的虚拟机在A退还资源前，B用户不能使用，哪怕A用户的虚拟机处于闲置状态。

动态扩展：用户和资源没有强绑定关系。最典型的例子为Google App Engine，用户申请的资源和用户没有严格的一对一关系，换句话说，处理A用户请求的进程在处理完之后，可以马上处理B用户的请求。

两种扩展性各有利弊，静态扩展的长处是为平台提供了良好的隔离性，资源可以固定映射在某个用户下，但缺点是资源利用率不高；动态扩展的长处是资源利用率高，这样整个云计算平台的成本会很低，但缺点是对隔离性有更高的要求，因为资源可以在很短的时间被多个用户使用。相比较，在安全性上，动态扩展要比静态扩展的技术门槛更高。

在SAE平台上，我们采用以动态扩展为主、静态扩展为辅的兼而有之的设计。在Web计算池层是典型的动态扩展。而在SAE的某些服务中，又是以静态扩展的方式展现，如RDC（Relational DB Cluster）分布式数据库集群，当用户申请了MySQL服务，我们就会在RDC后端根据SLA创建一主多从的DB给用户，在用户显式删除该DB前，该DB都不会被别人使用。当然，通过RDC，任何一个用户也无需知道后端DB的实际地址，只需访问RDC统一的Host和Port即可。

高可靠性（High Availability，简称HA）

HA是分布式系统的另一个主要目的，SAE同样以提供服务的高可靠性为架构设计的重要指标。HA的实现途径主要有两个：一个是硬件保证，另一个是架构的冗余设计。

在SAE平台上，所有服务器都是新浪标准采购的硬件设备，运行在国内最好机房内，网络资源方面则享用门户网站所使用的带宽环境。另外，所有的硬件设备都有专门的运维部门负责，故障的响应速度和新浪内部服务一样。

在架构设计上，SAE通过对所有服务都进行冗余设计来提供服务的高可靠性。这里的服务可以分成计算型和数据型两种类别讨论。

针对计算型服务，冗余设计就是程序在多节点运行。我们要求SAE所有的内部代码程序要做到Stateless（无状态依赖），即无依赖部署无依赖启动，随时终止进程随时重启进程，这样一旦出现机器故障或者程序自身Bug时，所有进程能够随着硬件环境的重新恢复而第一时间重启。而多点执行的程序可以保证，当某些程序出现故障时，整个系统仍然能够正常提供服务。

计算型程序多点部署，会带来一致性问题，最主要的困扰就是选举问题，如何在多个节点中选出一个主节点来执行。比如SAE上的分布式定时服务Cron，采用多点部署方式，多个计算节点相互隔离，通过时钟同步服务同时触发用户设定的定时任务，但要求只能有一个节点负责执行。为了解决这个问题，SAE设计出了一套分布式锁算法来提供选举服务。该算法可以在牺牲某些特定条件下的一致性来提供比Paxos算法更高的可靠性（3台机器在最高任意2台机器发生故障的情况下整个选举过程仍然正常，而Paxos算法最多容忍1台）。目前，该算法正在申请专利，并广泛应用在SAE内部。

针对数据型服务，SAE主要是通过复制来保证服务的高可靠性。SAE上的数据存储服务普遍采用被动复制和主动复制两种方式。如SAE上MySQL之间的主从Binlog同步就是典型的被动复制，用户只写写库，数据从写库同步到多个读库中。Taskqueue、DeferredJob等服务也采用被动复制的方式，用户的任务描述会写到主内存级队列中，主队列利用后台线程将写操作同步到从队列上，一旦主队列发生故障，从队列会快速的切换为主队列。另外SAE上也有部分服务采用主动复制（双写复制）的方式来保证HA，比如Cron，当用户通过App的工程配置文件appconfig.yaml设定定时任务时，任务信息会以双写的方式写到多个持久化DB中，以供后续的到时触发。

另外，SAE在整体架构设计时，充分考虑服务之间的“优雅降级”，尽量降低服务之间的耦合度，我们要求任何一个服务都不要假设其他服务是可靠的。目前在SAE平台上的所有服务均不存在单点设计，服务的平均HA在99.9%，即年平均服务不可用时间在8~9个小时之间。

未来

未来SAE有两个重要工作，一个是为开发者提供公平完善的评估奖励机制，并开放多种支付接口，目的就是早日形成和开发者之间双赢的良性循环，来吸引更多的开发者在SAE开发应用；另一个就是在保证现有服务稳定的前提下，为用户提供更多更便捷的服务，涉及CDN、网络代理、Key-Value数据库等。以后我会继续给大家介绍SAE内部具体服务的架构和技术实现细节。