云计算架构：最佳实践

2010 年 1 月

上次更新 – 2011 年 1 月

Jinesh Varia

jvaria@amazon.com

简介

近几年来，软件架构师发现并落实了一些概念和最佳实践，用来构建高度可扩展的应用程序。在当今的“万亿时代”，鉴于日益增长的数据集、不可预测的流量形式，以及对更快响应时间的需求，这就使得这些概念更具适用性。本文将强化并重申一些传统观念并对它们在云计算背景下的发展展开讨论。本文还将讨论一些前所未有的概念，例如弹性，因为云计算的动态本质使得这些概念显现。

本文针对的是*云架构师*，他们正摩拳擦掌打算把一个企业级的应用程序从一个固定的实体环境移到虚拟化的云计算环境中。本文的重点是强调创建新的*云应用程序*或将现有的应用程序移植到云端的概念、原则和最佳实践。

背景

作为一个云架构师，首要是能够了解云计算的效益。在本节中，您将了解云计算带来的一些业务与技术优势和一些目前已可利用的 AWS 服务。

云计算的业务效益

在云中新建应用程序有着一些明显的商业回报。这里列出了其中的一部分：

前期基础设施投资几乎为零：如果您必须组建一个大型系统，那么在不动产，物理安全，硬件（机架，服务器，路由器，备用电源），硬件管理（电源管理，冷却），和操作人员上的投资将会耗去您大量的资产。昂贵的前期成本使得在启动该项目之前，通常都要经过管理层的多次审核批准。现在，使用实用型的云计算，则不再需要固定的花费或启动成本了。

即时基础设施：在过去，如果您的应用程序流行开来，但您的系统或基础设施不能及时扩展，那么您将成为自己成功的受害者。相反的，如果您投入了大量的资金，但却没获得成功，您为自己的失败所累。如果您能通过部署在云中具有即时自我资源调配的应用程序，则不必再担心预采购大型系统的能力。因为它提高了灵活性、降低风险与降低了运营成本，您只需配合您的成长而扩展以及按您的使用量付费。

*更有效的资源利用：*系统管理员经常会忧虑硬件的采购（当它们耗尽内存时）和提高基础设施利用率（当它们有过剩和空闲容量时）。使用云计算，可以更实际与更有效率的通过应用程序请求和按需撤除资源的使用来管理资源。

基于使用的成本花费：通过效用计算的定价方式，您只会收到所使用过的基础设施的账单。您无需为那些相关却未使用的基础设施付款。这为节约成本提供了一个新维度。当您部署一个优化补丁更新您的云应用程序后，您将很快能看到成本的降低（有时甚至会早在您下月的账单中）。例如，如果一个缓存层能够减少 70% 的数据请求，成本的缩减将会立刻开始累积并且您很快就会在下一个账单中看到它。此外，如果您在云上搭建平台，那么您将能够将同样灵活，多种基于使用的成本结构传递给您自己的顾客。

缩短上市时间：并行化是一个很好的方法，用来加快处理的速度。如果在一台机器上并行处理一个计算密集型或数据密集型工作需要 500 个小时，那么使用云架构 [6]，它就能够衍生并且启动 500 个实例，在 1 小时内处理

相同的工作。弹性的基础设施为应用程序提供了利用并行化的能力，从而使其能以一个节省成本方式缩短产品上市时间。

云计算的技术优势

云计算的一些技术优势包括：

自动化 – "编写脚本基础设施" : 您可以通过利用可编程的（API – 驱动）基础设施来创建可重复构建和部署的系统。

自动扩展 : 您可以在无任何人工干预的情况下，通过自动扩展来满足你预期之外的需求。自动扩展使自动化和驱动更有效的运行。

主动扩展 : 根据您的流量模式主动扩展您的应用程序系统，从而使您在缩放期间仍能保持低成本，又能满足您的预期需求。

更有效地开发生命周期 : 生产系统应该更容易复制做为开发和测试环境。暂存环境可以很容易的升级到生产环境。

改善的可测性 : 永远不会用完的测试硬件。开发过程中的每一个阶段都能进行注入和自动化测试。您可以使用只适用于测试阶段的预配置环境来生成一个"即时测试实验室"。

灾难恢复和业务存续 : 云为维护灾难恢复服务器组和数据存储提供了低成本的可能。有了云，您就可以充分利用地理分布并用数分钟时间在其他位置复制该环境。

流量"溢出"到云 : 使用有效的负载均衡策略，点击几下，您就可以创建一个完整的防溢应用程序，从而使超出的流量路由到云。

了解 Amazon Web Services Cloud

Amazon Web Services (AWS) 云计算提供了一个高度可靠、 可扩展的基础设施以部署 web 规模的解决方案，以最低的支持和管理成本，以及高于您对基础设施、应急设施或数据中心所预期的灵活性来运营。

现在，AWS 可以提供多种基础设施服务。下面的图解将为您介绍 AWS 名词，帮助您了解您的应用程序如何与不同的 Amazon Web Services 互动和不同的服务之间是如何互动的。

Amazon Elastic Compute Cloud (Amazon EC2)[1]是一种 Web 服务，可在云中提供大小可调的计算容量。您可以将操作系统，应用软件及相关的配置设置一起捆绑进*亚马逊系统映像*(AMI) 中。然后，您可以使用 这些 AMI 来开通与停用多个虚拟化*实例*，使用简单的 Web 服务调用依照您容量要求的变化来快速扩展容量。您可以按小时购买*按需实例*或者低价一次性支付购买*预留实例*，但运行该实例要比按需实例或*竞价型实例*运行的利用率要低，其中竞价型实例中您可以预先为未使用的容量付款并进一步降低您的成本。您可以在一个或多个地理*区域*中启动实例。每个地区都有多个*可用区域*。可用区域的设计是依不同的位置隔离其他可用区域故障的影响，并且可向同一地区中的其他可用区域提供低价与低延迟的网络连接。

[1] Amazon EC2 的更多信息请登录 http://aws.amazon.com/ec2

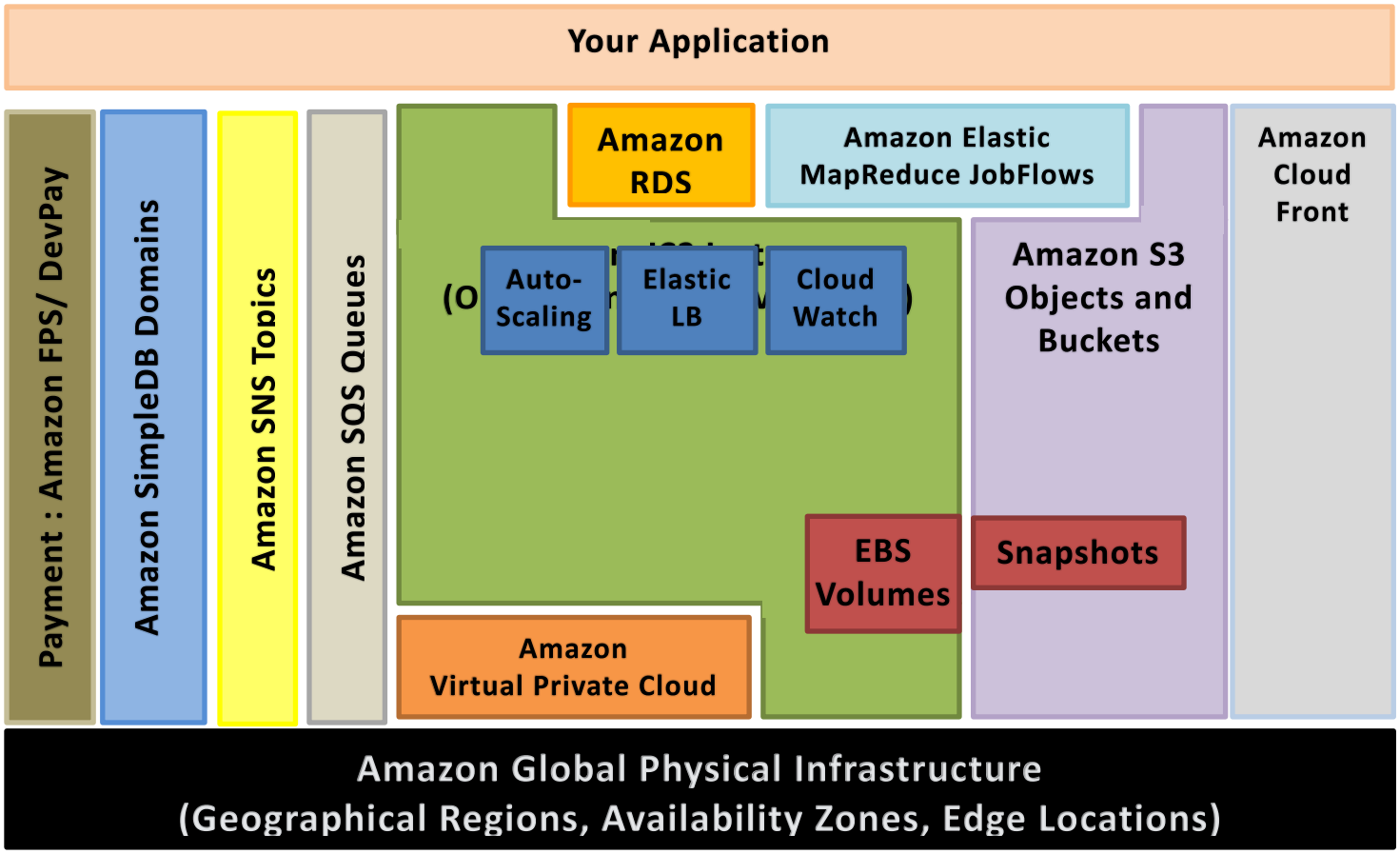

图 1：Amazon Web Services

弹性 IP 地址可让您分配一个静态 IP 地址并以编程方式将它分配到一个实例。您可以使用 Amazon CloudWatch[2]在一个 Amazon EC2 实例上使能监控，从而增加资源利用率，经营业绩和整体需求模式（包括指标，如 CPU 使用率，磁盘读 写 ，网络流量）的可见性。您可以使用 Auto-scaling 功能创建 *Auto-scaling 组* 来根据 Amazon CloudWatch 采集的指标，在一定条件下[3]自动扩展容量大小。您还可以使用 Elastic Load Balancing 服务创建一个*弹性负载平衡器*来分配传入流量[4]。AmazonElastic Block Storage(EBS)[5]*卷*为 Amazon EC2 实例提供有网络连接的持久性存储。可在 Amazon Simple Storage Service (Amazon S3) 上创建并存储时间点一致的 EBS *卷快照*。

Amazon S3 具有高可持久性并且是分布式的数据存储。使用一个简单的 Web 服务接口，您就可以随时，从使用标准 HTTP 协议的 Web 上将存储段（容器）中的大量数据作为*对象*进行*存储和取回*。使用 Amazon CloudFront 服务创建一个分发可以将对象的副本分发并缓存到遍布世界的 ？ 个*边缘位置*，Amazon

[2] Amazon CloudWatch 的更多信息请登录 http://aws.amazon.com/cloudwatch/
[3] Auto-scaling 功能的更多信息请登录 http://aws.amazon.com/auto-scaling
[4] Elastic Load Balancing 功能的更多信息请登录 http://aws.amazon.com/elasticloadbalancing
[5] Elastic Block Store 的更多信息请登录 http://aws.amazon.com/ebs
[6] Amazon S3 的更多信息请登录 http://aws.amazon.com/s3

CloudFront[7]服务是一个用于内容分发（静态或流传输内容）的 Web 服务。Amazon SimpleDB[8]是一个提供实时数据库查找和结构化数据简单查询核心功能的 Web 服务，它无需复杂的操作。您可以将数据集组织到*域*中，并对特定域中存储的所有数据运行查询。域是*项目*的集合，通过*属性-值对*进行描述。Amazon Relational Database Service[9](Amazon RDS) 为云中关系数据库的设置，操作和扩展提供了一种简易的方法。您可以启动一个 *DB 实例*和访问一个全功能的 MySQL 数据库并且不用考虑如备份，补丁管理之类的常见的数据库管理任务。

Amazon Simple Queue Service (Amazon SQS)[10] 是一个可靠且高度可扩展的分布式托管队列，当消息在计算机之间和在应用程序组件之间传输时它可用于存储*消息*。

Amazon Simple Notifications Service (Amazon SNS) [11] 通过创建*主题*和使用发布-订阅协议为应用程序或人员提供了一种简易的通知方法。

Amazon Elastic MapReduce[12] 提供了一个在 Web 范围的 Amazon Elastic Compute Cloud (Amazon EC2) 和 Amazon Simple Storage Service (Amazon S3) 基础设施上运行的托管 Hadoop 框架，并且使您能创建自定义的*任务流程*。任务流程是一个 MapReduce *步骤序列*。

Amazon Virtual Private Cloud (Amazon VPC)[13] 允许您将您的公司网络扩展进一个包含于 AWS 的私有云。Amazon VPC 使用 IPSec 隧道模式就使得您能够在您的数据中心网关和 AWS 中的网关之间创建一个安全连接。

Amazon Route53 是一个允许您在您想要管理的每一个域内创建一个*托管区域*来管理您的 DNS 记录的，高度可扩展的 DNS 服务。

AWS Identity and Access Management (IAM)[14] 可让您使用独特的安全凭证在 AWS 账户下创建多个用户，并且分别管理这些用户的权限。IAM 是以原生的方式集成到 AWS 服务中的。无需改变服务 API 既可支持 IAM，现有的构建在 AWS 服务 API 基础之上的应用和工具都可以使用 IAM 继续正常工作。

AWS 还充分利用 Amazon 的支付平台为您提供了多种支付和账单服务[15]。

所有的 AWS 基础设施服务都提供了无需长期承诺或合同的效用型定价。例如，您可以按小时支付 Amazon EC2 实例使用，并且对于 Amazon S3 您就可以按 GB 来支付存储和数据传输。这些随用随付的服务定价更多信息，请您登录 AWS 网站查询。

请注意，使用 AWS 云仍能使您保持您已经习惯的灵活性和控制度：

---

[7] Amazon CloudFront 的更多信息请登录 http://aws.amazon.com/cloudfront
[8] Amazon SimpleDB 的更多信息请登录 http://aws.amazon.com/simpledb
[9] Amazon RDS 的更多信息请登录 http://aws.amazon.com/rds
[10] Amazon SQS 的更多信息请登录 http://aws.amazon.com/sqs
[11] Amazon SNS 的更多信息请登录 http://aws.amazon.com/sns
[12] Amazon ElasticMapReduce 的更多信息请登录 http://aws.amazon.com/elasticmapreduce
[13] Amazon Virtual Private Cloud 的更多信息请登录 http://aws.amazon.com/vpc
[14] Amazon Identity and Access Management 的更多信息请登录 http://aws.amazon.com/iam
[15] Amazon Flexible Payments Service 的更多信息请登录 http://aws.amazon.com/fps， Amazon DevPay 的更多信息请登录 http://aws.amazon.com/devpay

- 您可以自由使用您选择的编程模型、语言或操作系统（Windows、OpenSolaris 或任何版本的 Linux）。
- 您可以自由挑选最令您满意的 AWS 产品，可以单独使用任一种或多种服务组合。
- 由于 AWS 提供了可调整大小（存储、带宽和计算）的资源，所以您可以自由使用，使用无论是多是少，您只需支付您消耗的部分即可。
- 您可以自由使用以前用过的系统管理工具并可以将您的数据中心扩展到云中。

云的概念

云强化了构建高度可伸缩网络架构[13]的一些旧概念，并引入了一些完全改变构建和部署应用程序方法的新概念。因此，当您从概念发展到应用时，您可能会发现“一切都变了，但又没什么不同。”云改变了一些过程、模式、实践，原理并加强了一些您已经了解的、传统的面向服务架构原则，它们变的比以前更为重要。本节中，我们将为您介绍一些新的云概念，并重申 SOA 概念。

传统的应用程序是以既定的思维方式来构建，在它们开发的当时是合乎经济和架构意义的。而云带来了一些您需要了解的新理念，如下面的讨论：

构建可扩展的架构

为了充分利用可扩展的基础设施，构建一个可扩展的架构是至关重要的。

云的目的就是提供无限可扩展性的理念。然而，如果您的架构没有扩展性，您就不能充分利用基础设施里所有的可扩展性。二者必须同时具备。您必须先辨认出您架构中的单块组件和瓶颈，认识到在您的架构中不能充分利用按需配置能力的部分，并且要*重建*您的应用程序，从而能够充分利用可扩展的基础设施和云的优势。

一个真正可扩展应用程序的特点：

- 增加资源可以得到按比例增高的性能
- 可扩展的服务能够解决异构性问题
- 可扩展的服务是能够高效的运行
- 可扩展的服务是有弹性的
- 可扩展的服务是能够随着它的扩展（单位数量增加，使得单位成本降低）变得更为经济高效

这些都应成为您的应用程序中一个固有部分，如果您在设计架构时能牢记上面的特点，那么您的架构与基础设施能够一起工作，达到您想要的可扩展性。

了解弹性

下列图表说明一个云架构师可以采取不同的方法来扩展他们的应用程序，从而使其满足需求。

*向上扩展方法：*不考虑可扩展应用程序架构，巨额投资和使用更大更强力的计算机（垂直扩展）来适应需求。这种方法通常只会有效至某个程度，但很可能是所费不赀（参见图表中“巨额资本开支”）或者需求会在新的“大型机”部署之前超出增长能力（参见图表中“您丢失了您的客户”）。

*传统横向扩展方法：*创建一个横向扩展架构并且以小块为单位地向基础设施投资。大多数的企业和大型 Web 应用程序都沿袭这种模式，他们按需求分配他们的应用程序组件，联合他们的数据集和采用一种面向服务的设计。这方法会比向上扩展法更高效。然而，这种方法仍需要定期预测需求，再以小块的形式部署基础设施来满足需求。这往往会导致容量过剩（“烧钱”）和长期的人工监控（“过度人力循环”）。另外，如果应用程序受到病毒的侵染，那么它通常就会停止工作（通常被称为“点杠效应”[16]）。

注意：这两种方法都有初始启动成本并且这两种方法在本质上都是被动响应式。

[16] http://en.wikipedia.org/wiki/Slashdot_effect

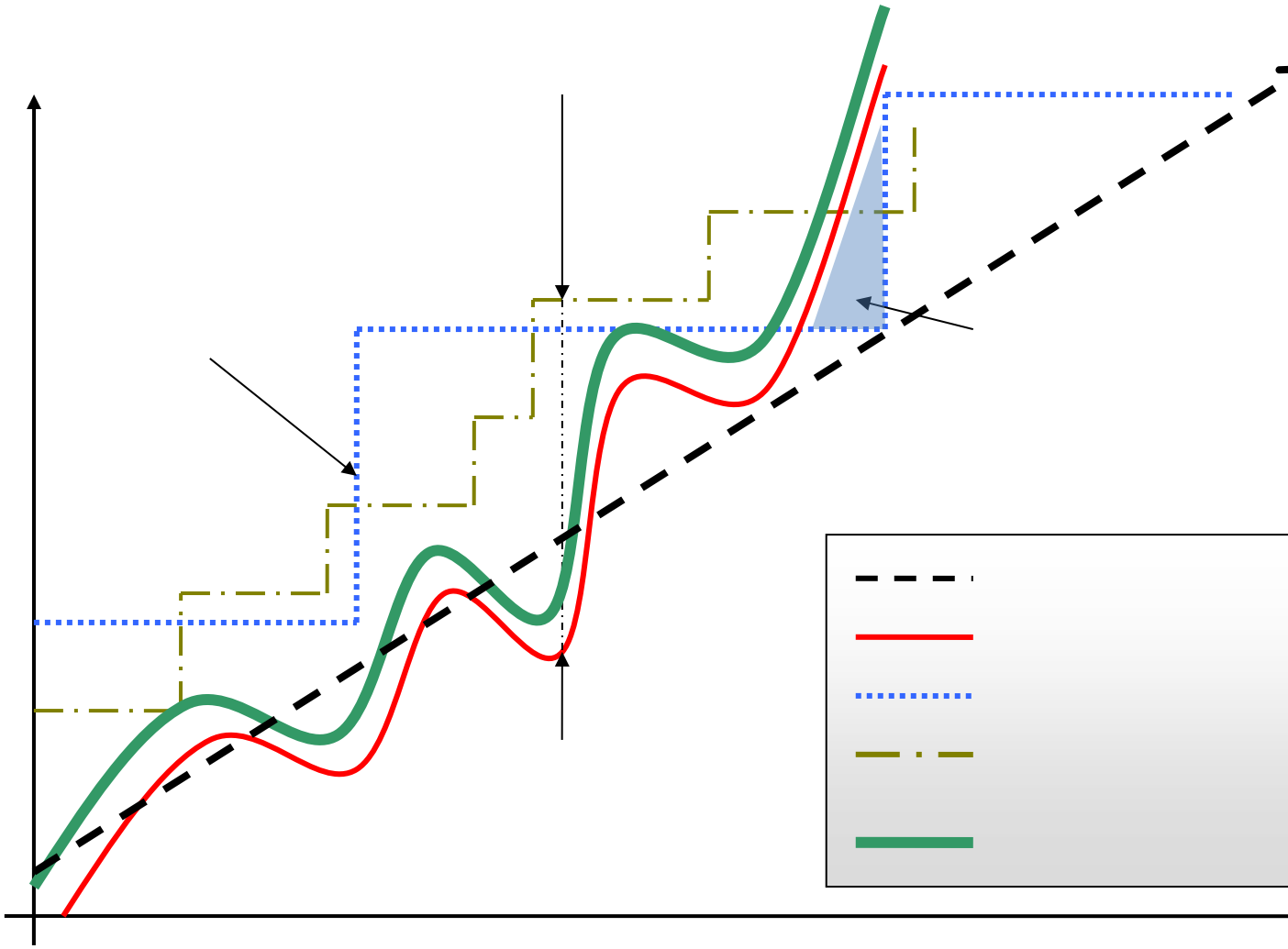

图 2：自动弹性

传统基础设施通常都需要预测您的应用程序在接下来几年里将会用到的计算资源数量。如果您低估了，那么您的应用程序可能没有足够能力处理预期之外的流量，可能会引起客户的不满意。如果您多估了，您又浪费金钱在多余的资源上。

按需分配和弹性本质是*云的处理方式*（自动弹性），使基础设施能与实际需求紧密结合（随它扩大和收缩），从而提高整体利用率并降低成本。

*弹性是云的基本的属性之一。*弹性是以最容易的方式提供上下扩展计算资源的动力。重点是要能了解弹性终将带来大多数云的好处。作为一个云架构师，您需要从本质上理解这一概念并将它带入到您的应用程序架构中，从而最大化云计算的收益。

传统上来讲，应用程序是构建在固定的，刚性的和预配置的基础设施上。公司从来不需要每天配置和安装服务器。结果导致大多数软件架构都不会处理快速部署或硬件的减少。由于获取新资源的配置时间和前期投资太高，所以软件架构师从不会投入时间和资源在优化硬件的利用率上。如果运行应用程序的硬件未被充分利用，仍然是可以被接受的。一个架构内“弹性”的概念之所以会被忽略，因为都认为在几分钟之内获得新资源是不可能的事。

随着云的出现，这一心态应随之而改变。云计算简化了获得必要资源的过程；将不再需要提前订购和保存未使用的硬件。相反的，云架构师可以充分利用云中规模庞大及响应快速的优势，只需在他们实际要使用之前的几分钟发出请求或将采购流程自动化即可。当没有需求时也同样的适用，可以释放不再需要或者未被利用的资源。

如果您不能接受这种变化并且不能将弹性应用到您的应用程序架构中，那么您可能没有充分利用云的优势。作为一个云架构师，您应该创造性的思考，思考如何在您的应用程序应用到弹性。例如，日夜运行的基础设施，在每夜凌晨 2:00 构建与执行回归和单元测试两个小时（通常被称为“QA /生成框”），则一天中其他的时间都在闲置。现在，使用弹性基础设施，就可以在那些“工作”的盒子上运行每晚构建，而且只需支付夜间 2 小时的费用。同样，一个内部故障标签 Web 应用程序在白天通常都是以峰容量（5 台服务器 24x7x365 ）运行来满足需求，现在可以将它根据流量模式按需配置（5 台从上午 9 点到下午 5 点运行，2 台从下午 5 点到上午 9 点运行）。

设计智能型的弹性云架构，使基础设施只在您需要的时候运行，这本身是一种艺术。弹性应是架构设计需求之一或一个系统属性。您应该会问的问题：我的应用程序架构中有哪些组件或层可以设计成弹性？怎样才能将该组件设计为*弹性*？实现弹性将会对我的整体系统架构产生什么影响？

在下一节中，你将会看到在你的应用程序执行弹性的具体技巧。有效的运用云的好处是架构师重要的心态。

不怕制约因素

当您决定将您的应用程序转移到云中并试图将您的系统规格映射到云中那些可用的规格上时，您会发现云中可能没有如同您实地拥有一般的资源规格。例如，“云无法提供某数量的 RAM 给一个服务器”或者“相比于在中，我的数据库需要比单一实例所能提供的更多的 IOPS”。

您要明白，云提供的是*抽象资源*，并且只有在您将它们与按需配置模式相结合时，它才能表现出它的强大之处。在使用云资源时，您不必害怕而受到这种资源的约束，因为重要的是您能够明白，在云环境中，虽然您不能复制一模一样的硬件规格，但您可以找到更多相关的资源来弥补这种需求。

例如，如果云无法提供给您的服务器相同或者更多的 RAM，那么您可以使用一个分布式缓存，像 *memcached*[17] 或者将您的数据分割到多个服务器上。如果您的数据库需要更多的 IOPS，又 无法在云中直接映射，您可以根据您的数据类型和使用情况进行多种选择。如果是一个大量读取的应用程序，那么您可以将读取负载在一组同步从机间进行分配。或者，您可以使用一个*分片*[10]演算法将数据路由到它需要的地方或者您也可以使用各种不同的数据库集群解决方案。

回头来看，当您将按需配置功能与其灵活性结合起来时，你会发现明显的制约因素实际上可以拆散成不同的方式，而能改进其可扩展性和系统整体的性能。

虚拟管理员

云的出现使系统管理员的角色变为了“虚拟系统管理员”。这意味着系统管理员所执行的日常工作现在变得更为有趣，当他们对应用程序了解更多自然会明白什么才是对整个企业最有利的。系统管理员将不再需要配置服务器和安装软件及连接网络设备，因为这些繁重的工作都将被点击鼠标和命令行调用所代替。因为基础设施是可编程的，所以云鼓励自动化。系统管理员需要提升累积的技术和学习如何使用脚本管理抽象云资源。

同样，数据库管理员的角色也转变成一个“虚拟数据库管理员”，他/她通过一个基于 Web 的控制台管理资源，执行新增的可编程功能脚本来防止硬件容量耗尽和自动的进行日常流程。今天，虚拟数据库管理员必须学习新的部署方法（虚拟机镜像），接受新模型（查询并行化，地理冗余和异步复制 [11]），重新思考数据架构方法（分片 [9]，水平分区 [13]，整合 [14]），并充分利用不同的存储选项可在云中用于不同类型的数据集。

在传统的企业公司中，应用程序开发人员可能不会与网络管理员接触工作，而网络管理员可能也不清楚何谓应用程序。结果导致网络层和应用程序架构层中几种可能的优化被忽略了。有了云，这两个角色将在一定程度上合而为一。在架构未来的应用程序时，公司应鼓励两个角色间的知识互换和使他们认知到他们是一体的。

[17] http://www.danga.com/memcached/

云最佳实践

在本节中，您会了解能帮助您在云中构建应用程序的最佳实践。

设计时考虑到故障，最后就不会有故障

经验法则：在设计云中的架构时要是一个悲观主义者，且假定有可能会发生故障。换言之，始终要设计、实施和部署故障自动恢复措施。

尤其是假定您的硬件*将*发生故障。假定*会*发生断电情况。假定某些灾难*会*使您的应用程序停止运转。假定某一天每秒的请求超过了预期数量，您*会*感到挫折。假定与此同时，您的应用程序软件也发生了故障。作为一个悲观主义者，您最终会在设计期间考虑恢复策略，这会帮助您设计更好的整体系统。

如果你意识到事情是随着时间的推移而失败的，并把这种想法集成到您的架构中，为了处理一个可扩展的基础设施，您可在灾难来袭之前建立解决该故障的机制，这样一来，你最终会创建一个优化于云的容错架构。

您需要问的问题就是：如果系统中的一个节点发生故障，那么会发生什么情况？您是如何意识到该故障的？如何更换该节点？我要策划什么样的情景？我的单点故障有哪些？如果负载平衡器位于应用服务器组的前面，那么如果该负载平衡器出现故障的话会出现什么情况？ 如果在您的架构中有主机和从机，当主节点发生故障的话会出现什么情况？ 该故障转移是如何发生的，以及如何实例化一个新的从机并将之与主机同步？

就像为硬件故障设计一样，您还须为软件故障设计。您要问的问题是：如果从属服务改变了其接口，那么我的应用程序会发生什么情况？ 如果下游服务器超时或返回意外错误的话会发生什么情况？ 如果缓存键的增长超出了一个实例的内存限制的话会发生什么情况？

建立新的机制来处理故障。例如，一旦发生故障，以下策略可以帮助您：

1. 为您的数据准备一个连贯的备份和恢复策略并将之自动化
2. 建立在重新启动时恢复的进程线程
3. 通过重载队列中的消息来允许系统状态重新同步
4. 通过保持预配置和预优化的虚机镜像来支持在（2）和（3）上的启动/引导
5. 避免内存内会话或有状态的用户上下文，将之移动到数据存储。

良好的云计算架构应该是不受重新引导和重新启动影响的。在 GrepTheWeb 中（已在云架构文件 [6] 中进行了讨论），通过把 Amazon SQS 和 Amazon SimpleDB 组合使用，对本节列出的几种故障类型来说，整个控制器架构是非常有弹性。例如，如果控制器线程上的实例运行死亡，则可以把它提出来并将之恢复以前的状态，就像仿佛什么都没有发生过一样。这是通过创建一个预配置亚马逊系统映像 (AMI) 来完成的，此时 Amazon SQS 队列会把所有消息从队列中取出，并从一个重启的 Amazon SimpleDB 域中读取其状态。

设计时假定底层硬件会发生故障，当其真发生故障时，您就可为未来做好准备了。

这种设计原则将帮助您设计业务友好型应用程序，同时在 Hamilton 的[11] 文件中也强调了此原则。如果您可以扩展这一原则来积极主动地开展措施并动态地均衡负载，您就能够处理由于云的多租户性质而存在的网络和磁盘性能之间的差异。

实施这个最佳实践的 AWS 具体策略是：

1. 故障转移平稳地使用弹性 IP 地址：弹性 IP 是一个静态的 IP，是动态地可重映射的。您可以快速重新映射并故障转移到另一服务器组，这样，您的流量就会被路由到新的服务器。您要把旧版升级到新版本或在硬件故障的情况下，它能很好地运作
2. 采用多个可用区域：在概念上可用区域与逻辑数据中心是类似的。通过把架构部署到多个可用区域，您可以确保高可用性。采用 Amazon RDS Multi-AZ [21] 部署多个可用区域的功能，以便自动复制数据库的更新。
3. 维持一个亚马逊系统映像 (AMI)，您就可以在不同的可用区域很容易地恢复并复制的环境；维护可用区域之间多个数据库从机并设置热复制。
4. 采用 Amazon CloudWatch（或各种实时开源监视工具），以获得更多的可见性，且在硬件故障或性能下降的情况下采取适当的措施。通过设置一个自动扩展组来维持一个固定数量的服务器集群规模，以便让它用新的实例取代不健康的 Amazon EC2 实例。
5. 采用 Amazon EBS 并设置时钟守护作业，从而使增量快照自动上传到 Amazon S3 且数据是持久独立于您的实例的。
6. 采用 Amazon RDS 并设置备份的保留期限，以便它可以执行自动备份。

解耦组件

云强化了 SOA 的设计原则，系统的耦合组件越松散，它越能更大更好地进行扩展。

关键是要建立彼此间依赖关系不紧密的组件，以便于假如由于某种原因一个组件死掉了（发生故障）、睡着（没有响应）或处于忙碌状态（响应速度慢）时，能够构建系统的其他组件，以便继续工作，就像未发生任何故障一样。从本质上讲，松散耦合会把多种层和应用程序的组件隔离开来，可以使每个组件与其他组件异步交互，并把它们当作一个“黑盒子”。例如，如果是在 Web 应用程序架构，您可以把应用程序服务器从 Web 服务器和从数据库中隔离开来。应用程序服务器并不知道您的 Web 服务器，反之亦然，这就为这些层之间进行了解耦，并且无论是从代码级别上看还是从功能角度看，都不存在依赖关系。在批处理架构的情况下，您可以创建彼此独立的*异步*组件。

您需要问的问题是：哪些商业组件或功能可以从当前单一的应用程序中隔离开来，且能分别独立运行？然后我怎样做才可以添加更多的组件实例，且不会不打破我目前的系统并同时能为更多的用户提供服务呢？ 多少花费多大的精力来封装组件，以便它可以与其他组件异步互动？

在云背景下，解耦组件、构建*异步*系统和水平扩展就变得非常重要。它不仅可以让您通过添加相同组件的更多实例进行横向扩展，也可让您设计创新性混合模型，在该模型中，一些组件会在户内继续运行，而其他组件则可以利用云规模并使用云额外的计算能力和带宽。这样，通过实施智能负载均衡策略，您可以用最小的工作量把超出的流量“溢出”到云。

可以通过使用*消息队列*来建立一个松耦合系统。如果一个队列/缓冲区是用来把任意两个组件的连接到一起的，那么它就支持并发性、高可用性和负载高峰。因此，即使部分组件暂时不可用，整体系统仍将继续运行。如果一个组件死亡或暂时无法使用，系统会在组件恢复时缓冲该消息并对它们进行处理。

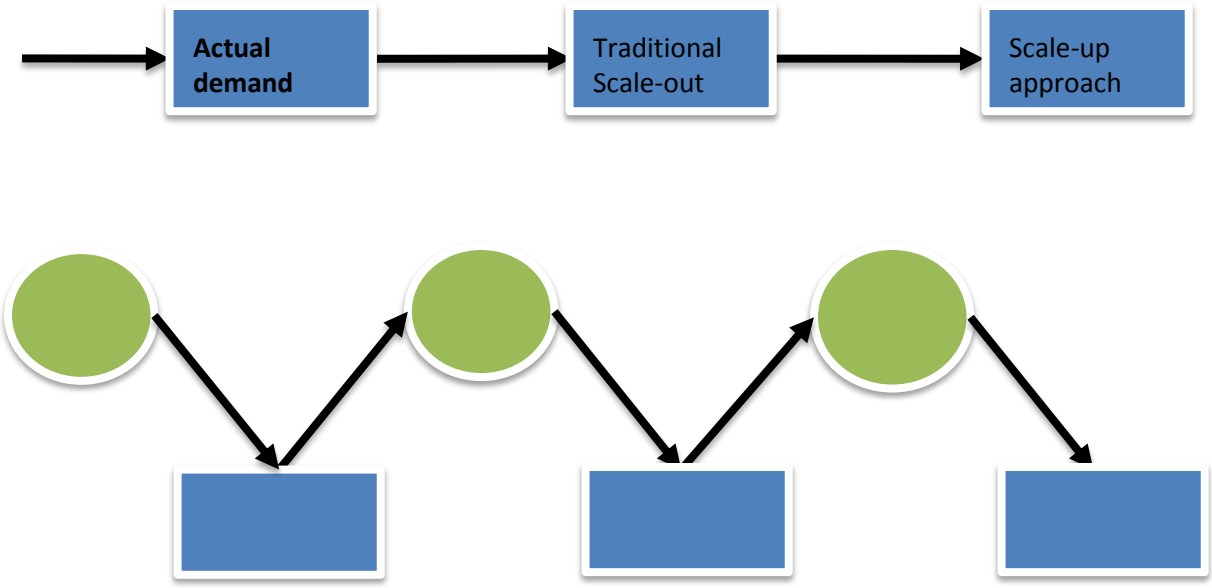

图 3：使用队列的解耦组件

您会看到集中体现在云架构文件 [6] 中的 GrepTheWeb 架构队列的大量使用。在 GrepTheWeb 中，如果大量请求突然到达服务器（Internet 引起的过载情况）或处理正则表达式需要花费比中间值更多的时间（组件响应率迟缓），在 Amazon SQS 队列会用一种耐久的方式来缓冲这些请求，从而使这些延迟不会影响到其他组件。

实施这个最佳实践的 AWS 具体策略是：

1. 使用 Amazon SQS 隔离组件 [18]
2. 把 Amazon SQS 用作组件 [18] 之间的缓冲
3. 设计每个组件，使之暴露一个服务接口，并负责其自身在所有适当的维度上的可扩展性，并与其他组件异步互动
4. 把一个组件的逻辑结构捆绑到亚马逊系统映像中，以便可以更频繁地对之进行部署
5. 尽量使您的应用程序实现无状态。存储组件外部的会话状态（在 Amazon SimpleDB 中，如果适当的话）

应用弹性

云为您的应用程序中带来了一个新的弹性概念。弹性可以在三个方面实施：

1. 主动周期型扩展：以固定的时间间隔（每日、每周、每月、每季）进行周期性的扩展
2. 主动基于事件扩展：因为排定的商业活动（新产品推出，营销活动）而预期流量将大幅飙升前进行的扩展。
3. 按需的 Auto-scaling。通过使用一种监控服务，您的系统就能发送信息来触发适当的行为，从而使它可以根据指标（例如，服务器或者网络 I/O 的使用率）进行扩展。

为了实现“弹性”，首先，必须要自动化部署过程，并简化配置和构建过程。这将保证该系统能够在无需任何人工干预的情况下进行缩放。

确保您的资源与需求紧密相连来提高整体利用率，而不是那些可能未被充分利用的服务器，将会立刻导致您的成本收益。

自动化您的基础设施
使用云环境最重要的好处之一就是能够使用云的 API 来自动化您的部署过程。建议您在迁移早期就花些时间创建一个自动化的部署过程，而不是等到迁移结束时。创建一个自动化和可重复的部署过程，将有助于减少错误的发生，并促进高效和可扩展的更新过程。

自动化部署过程：

- 使用脚本，创建一个小而常用的“配方”库（用于安装和配置）
- 使用内置于 AMI 的代理来管理配置和部署过程
- 启动配置您的实例

启动配置您的实例
让您的实例在启动时问您一个问题“我是谁与我的角色是什么？” 每一个实例都应充当在环境中的一个角色（例如，在一个 Web 应用程序中有“DB 服务器”，“应用程序服务器”，“从服务器”）。当引导发生后，这些步骤实例化时，指示 AMI 启动过程中，这些角色可能会被作为一个参数进行传递。在引导过程中，实例应根据其角色抓住必要的资源（代码，脚本，配置）并将其自身“附加”到一个集群来执行其功能。引导实例的好处：

1. 花费最少的精力，点击几下即能重建（开发，分级，生产）环境
2. 对您的抽象的云资源有着更多控制
3. 减少人工引起的部署错误
4. 创建一个自我修复和自我发现的环境，使其对硬件故障更具弹性

自动化基础设施的 AWS 特定策略
1. 使用 Amazon EC2 中的 Amazon Auto-scaling 功能来为不同的集群定义 Auto-scaling 组。
2. 使用 Amazon CloudWatch 监控您的系统指标（CPU，内存，磁盘 I/O，网络 I/O）并采取适当的行动（使用 Auto-scaling 服务动态的启动新的 AMI）或发送通知。
3. 动态的存储和获取机器配置信息：在一个实例的引导时间过程中，利用 Amazon SimpleDB 来获取配置信息（例如，数据库连接字符串）。SimpleDB 也可以用来存储一个实例的信息，例

如它的 IP 地址，机器名称和作用。
4. 设计一个构建过程，从而使它能够将最近的构建转储到 Amazon S3 中的一个存储段中；在系统启动过程中，下载最新版本的应用程序 。
5. 投资构建资源管理工具（自动化脚本，预配置图像），或使用智能开源的配置管理工具，如 Chef[18]、Puppet[19]、CFEngine[20]或 Genome[21]。
6. 把 Just Enough Operating System (JeOS[22]) 和相关的软件捆绑到一个 亚马逊系统映像 中，从而使它更容易管理和维护。启动时传递配置文件或参数和启动后检索用户数据[23]和实例元数据。
7. 通过从 Amazon EBS 卷中启动[24]和将多个 Amazon EBS 卷附加到一个实例上来减少捆绑和启动时间。创建通用卷快照，并在适当的账户之间共享[25] 快照。
8. 应用程序组件不应承担它所运行的硬件的健康或位置责任。例如，将一个新节点的 IP 地址动态的连接到集群。自动故障转移，并在出现故障的情况下，启动一个新的克隆。

考虑并行

云使并行化变得轻而易举。作为一个云架构师，当您在云中设计架构时，无论是要从云中请求数据，还是将数据存储到云，或者在云中处理数据（或者执行任务），您都需要从本质上理解并行化的概念。最好是在应用并行化的同时也将它自动化，因为云能使您很容易的创建出一个可重复的过程。

当涉及到访问（检索和存储）数据时，云就会被设计为处理大量并行操作的类型。为了获得最高性能和吞吐量，您应该充分利用*请求并行化*。使用多个并发线程进行的多线程处理请求比顺序处理请求存储或获取数据的速度要快。因此，在可能的情况下，云应用程序的过程应该通过无共享理念，做成线程安全的并能充分利用多线程的设计。

当涉及到在云中处理或执行请求时，充分利用并行化就显得更为重要了。在 Web 应用程序中，一个常见的最佳实践就是使用负载平衡器将传入请求分配到多个异步 Web 服务器之间。在批处理应用程序中，您可以用主节点产生出多个能够并行处理任务的从工作节点（就像在分布式处理框架中一样，如 Hadoop[26]）

[18] Chef 的更多信息请登录 http://wiki.opscode.com/display/chef/Home
[19] Puppet 的更多信息请登录 http://reductivelabs.com/trac/puppet/
[20] CFEngine 的更多信息请登录 http://www.cfengine.org/
[21] Genome 的更多信息请登录 http://genome.et.redhat.com/
[22] http://en.wikipedia.org/wiki/Just_enough_operating_system
[23] 实例元数据和用户数据信息请登录 http://docs.amazonwebservices.com/AWSEC2/latest/DeveloperGuide/index.html?AESDG-chapter-instancedata.html
[24] Boot From Amazon EBS 功能的更多信息请登录 http://developer.amazonwebservices.com/connect/entry.jspa?externalID=3121
[25] 登录 http://aws.amazon.com/ebs/ 查询快照共享方法
[26] http://hadoop.apache.org/

当您将弹性和并行化结合在一起时，云将魅力四射。您的云应用程序可以带来一个计算实例的集群，它能通过 API 调用在数分钟内完成配置，以并行的方式执行任务，存储结果和终止所有实例。在 [6] 中讨论的 GrepTheWeb 应用程序就是一个这样的例子。

对于并行化 AWS 的特定策略：

1. 多线程处理您的 Amazon S3 请求在最佳实践白皮书 [2] 中做了详细的描述
2. 多线程处理您的 Amazon SimpleDB GET 和 BATCHPUT 请求 [3] [4] [5]
3. 使用 Amazon Elastic MapReduce 服务为您的日常批处理（索引，日志分析等）创建一个工作流程。这将并行计算该任务并节省时间。
4. 使用 Elastic Load Balancing 服务并将您的负载*动态*地均衡到多个 Web 应用程序服务器之间。

保持动态数据与计算接近和静态数据与最终用户接近

一般来说，使您的数据与您的计算或处理单元尽可能的靠近以减少延迟，是一种最佳实践。在云中，这种最佳实践将更具相关性和重要，因为在云中，您会经常需要处理网络延迟问题。另外，在云中您是以数据传输的 GB 量来支付出入云的带宽，如果距离过远，那么成本也将显著增加。

如果有大量在云外部的数据需要处理， 若能先将数据“运送”和传输到云，然后再执行计算，或许是更便宜并且更快的方法。例如，对于一个数据仓库应用程序来讲，最好是先将数据集移动到云，然后再对数据集进行并行查询。对于从相关数据库中存储和检索数据的 Web 应用程序来说，最好是能一次性将数据库和应用程序服务器全部移入云中。

如果数据是在云中产生的，那么使用这些数据的应用程序也应该部署在云中，从而使它们能够充分利用云中的自由数据传输和低延迟。例如，在一个生成日志和点击流数据的电子商务 Web 应用程序中，最好就是能在云中运行日志分析和报告引擎。

反之，如果数据是静态的并且不会经常变化（例如，图像，视频，音频，PDF，JS，CSS 文件），那么最好的做法就是利用内容分发服务，使静态数据缓存在边缘位置更接近最终用户（请求者），从而降低访问延迟。由于存在缓存，所以内容分发服务能为访问普通对象提供更快捷的方式。

实施这个最佳实践的 AWS 具体策略是：

1. 使用 Import/Export 服务把数据盘发送到 Amazon [27]。使用 sneakernet[28]会比用 Internet 上传更便宜、更快速地移动大量数据。
2. 采用相同的可用区域启动计算机群集
3. 为你存储在 S3 上的存储容器创建分发，使用 Amazon CloudFront 世界各地的 14？ 个节点缓存该存储容器的内容

[27] Amazon Import Export Services 的更多信息请登录 http://aws.amazon.com/importexport
[28] http://en.wikipedia.org/wiki/Sneakernet

安全最佳实践

在一个多租户环境中，云架构师常常就安全问题表示担忧。安全性问题应在云应用程序架构的每一层都能落实。实际安全问题通常是由您的服务提供商处理（安全白皮书 [7]），这是使用云的一个额外优点。网络和应用层的安全是由您责任的，您应该执行最佳实践，以便使之尽可能地适用于您的业务。在本节中，您将了解一些特定的工具、功能和如何确保在 AWS 环境中您的云计算应用程序安全的指导方针。建议充分利用这些工具和功能，以便落实基本安全，然后再酌情使用标准的或他们认为合适的方法来执行附加的安全最佳实践。

保护在传输中的数据

如果您需要在一个浏览器和 Web 服务器之间交换敏感或机密信息，您就需要配置 SSL 到服务器实例上。您将需要一个由外部证书颁发机构颁发的像 VeriSign[29] 或 Entrust[30]一样的证书。证书中的公钥会从服务器到浏览器进行验证，是创建共享会话密钥的基础，用来双向加密数据。

通过一些命令行调用来创建一个虚拟私有云（使用 Amazon VPC）。这可以让您在 AWS 云内使用自己逻辑上隔离的资源，然后通过使用行业标准加密的 IPSecVPN 连接，将这些资源直接连接到您自己的数据中心。

您还可以把 Amazon EC2 实例上的设置为 [15] 一个 OpenVPN 服务器，并在所有用户的 PC 机上安装 OpenVPN 客户端。

保护处于静态的数据

如果您担心在云中存储敏感和机密的数据，您应该在上传到云之前对数据（个别文件）进行加密。例如，应在数据被存储为 Amazon S3 数据元之前，使用任何开源[31]或商用基于 PGP 的商业[32]工具来对数据进行加密，下载完成后再进行解密。在创建 HIPPA 符标的应用程序 [2]时，需要存储受保护的健康信息（PHI），通常会当作一个很好的实践。

在 Amazon EC2 上，文件的加密取决于操作系统。Amazon EC2 运行 Windows 的实例可以使用 Windows 内置的加密文件系统 (EFS)功能 [16]。此功能可以自动处理文件和文件夹的加密和解密，让客户对进程 [19] 一目了然。然而，不管其名称为何，EFS 不能加密整个文件系统，而是对单个文件进行加密。如果您需要一个完整的加密卷，则可以考虑使用开源 TrueCrypt[33]产品，该产品会对 NTFS 格式的 EBS 卷集成得很好。在 Amazon EC2 运行 Linux 的实例 能够使用加密文件系统或各种不同方法 (EncFS[34]、Loop-AES[35]、dm-crypt[36]、TrueCrypt[37])来安装 EBS 卷。同样的，Amazon EC2 运行 OpenSolaris 的实例也可以利用 ZFS[38]加密支持[20]。无论您选择哪一种方法，加密 Amazon EC2 上的文件和卷都有助于保护文件和日志数据记，所以，只有在服务器上的用户及进程可以清晰地看见文本中的数据，而在服务器之外的任何事物或任何人都只能看见已加密的数据。

---

[29] http://www.verisign.com/ssl/
[30] http://www.entrust.net/ssl-products.htm
[31] http://www.gnupg.org
[32] http://www.pgp.com/
[33] http://www.truecrypt.org/
[34] http://www.arg0.net/encfs
[35] http://loop-aes.sourceforge.net/loop-AES.README
[36] http://www.saout.de/misc/dm-crypt/
[37] http://www.truecrypt.org/
[38] http://www.opensolaris.org/os/community/zfs/

无论您选择哪个操作系统或哪种技术，处于静止状态的数据加密都会带来一个挑战：管理用于数据加密的密钥。如果您丢了密钥，那么您将永远丢失去您的数据，如果泄露了密钥，数据就会有风险。所以，无论您所选择的任何产品必须熟习密钥管理功能，同时建立一个丢失密钥风险最小化的过程。

除了保护您的数据不被窃听外，还需要考虑如何保护其免受灾难。采取对 Amazon EBS 卷的定期快照，以确保其是高度耐久性和高可用性的。快照在性质上是逐量增加的，存储在 Amazon S3 中（各别的地理位置），只须点击几下或是命令行调用即可恢复您的数据。

保护您的 AWS 证书

AWS 提供两种类型的安全证书：AWS 访问密钥和 X.509 证书。您的 AWS 访问密钥分为两个部分：您的*访问密钥 ID* 和您的*私有访问密钥*。当使用 REST 或 Query API 时，您必须使用您的私有访问密钥来计算包括在您的身份验证请求中的签名。为了防止在运行中被篡改，所有的请求都应通过 HTTPS 发送。

如果您的 亚马逊系统映像 (AMI) 所运行的进程需要与其他 AWS Web 服务进行沟通时（例如，检查 Amazon SQS 队列或从 Amazon S3 中读取数据元），一个常见的设计错误就是把 AWS 证书嵌入到 AMI 里。应当在启动期间把这些证书作为参数进行传输，而不是将证书嵌入，并且在传送 [17] 这些证书之前对之加密。

如果您的私有访问密钥被泄露了，您应该轮换[39]一个新的访问密钥 ID，以获取一个新的密钥。一个很好的实践方法就是，把一个密钥交替机制集成到您的应用程序架构中，这样一来，您就可以定期的使用它或偶尔为之（在满腹牢骚的员工离开了公司时），以确保被泄露的密钥无法永远持续下去。

或者，您也可以使用 X.509 证书对某些 AWS 服务进行身份验证。证书文件包含 base64 编码的 DER 证书上的公钥。另外一个文件包含相对应的 base64 编码的 PKCS＃8 私钥。

AWS 支持多因子身份验证[40]，当您在 aws.amazon.com 和 AWS 管理控制台上处里您的帐户信息时，用作一项额外的保障[41]。

管理多个用户及其具有 IAM 的权限

AWS Identity and Access Management (IAM) 可让您在 AWS 账户下创建多个用户，并且分别管理这些用户的权限。[42]一个用户就是一个具有独特的安全证书的身份验证（在您的 AWS 账户中），可通过使用该证书来访问 AWS 服务。IAM 省去了共享密码或访问密钥的要求，可以轻松地启用或禁用某个用户的访问权限。

IAM 让您可以实现安全最佳实践，如最低权限等，通过向您 AWS 帐户内的每个用户授予唯一的证书，且仅授予对 AWS 服务和用户为运行其工作所需资源的访问权限。IAM 默认状态下启用安全保护；新的用户只有被明确授予了权限之后，才能访问 AWS 资源。

---

[39] http://aws.amazon.com/about-aws/whats-new/2009/08/31/seamlessly-rotate-your-access-credentials/
[40] 有关多因素身份验证的更多信息， 请登录 http://aws.amazon.com/mfa/
[41] AWS 管理控制台网址：http://aws.amazon.com/console/
[42] 更多信息请登录 http://aws.amazon.com.com/iam

IAM 是生来就集成到多数 AWS 服务上的。没有服务 API 有所改变来支持 IAM，且当使用 IAM 时，在 AWS 服务 API 上构建的应用程序和工具将持续工作。只需使用一个新用户生成的访问密钥就可以开始应用程序。

当集成您的 AWS 服务并利用 IAM 用户证书访问 AWS 服务和资源时，您应尽量最小化 AWS 账户证书的使用。

保护您的应用程序

每一个 Amazon EC2 实例都是受一个或多个*安全组*[43]，已命名规则集的保护，指明哪个才能进入（即接收）网络流量的传输到您的实例中。您可以指定 TCP 和 UDP 端口、ICMP 类型及其代码和源地址。安全组为您正在运行的实例，提供基本类似于防火墙一样的保护。例如，一个 Web 应用程序的实例可以有以下安全组设置：

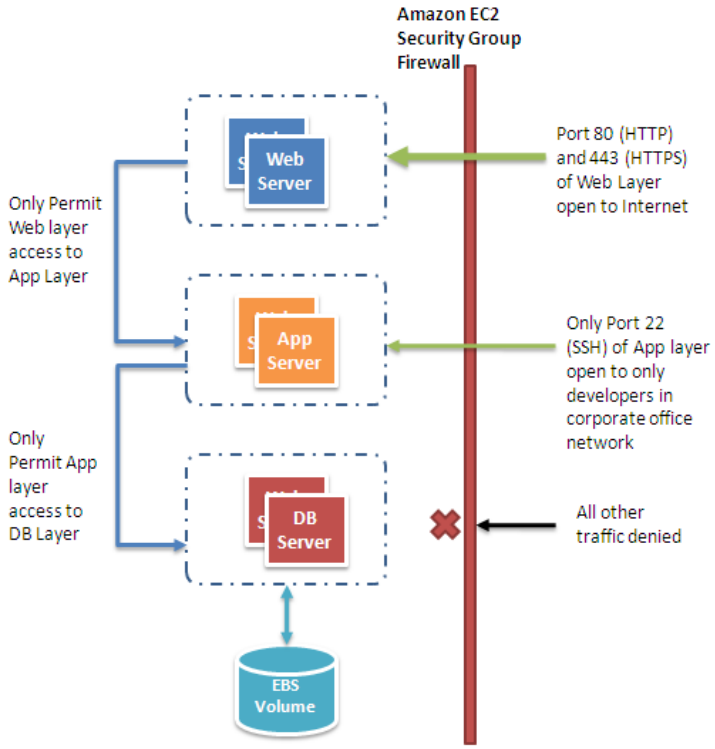

图 4：使用 Amazon EC2 安全组来保护您的 Web 应用程序

限制传入流量的另一种方法是在您的实例上配置基于软件的防火墙。Windows 实例可以使用内置的防火墙[44]。Linux 实例可以使用 *netfilter*[45] 和 *iptables*。

[43] 安全组的更多信息可登录 http://docs.amazonwebservices.com/AWSEC2/2009-07-15/UserGuide/index.html?using-network-security.html

[44] http://technet.microsoft.com/en-us/library/cc779199(WS.10).aspx，2003 年 3 月

[45] http://www.netfilter.org/

经过一段时间，软件中的错误会被发现并需要补丁来修复。您应该确保下列基本指导方针，以便最大限度地提高您的应用程序的安全性：

- 定期从供应商的 Web 站点下载补丁并更新您的 AMI
- 重新部署新的 AMI 中的实例并测试您的应用程序，以防止补丁破坏任何程序。确保已把最新的 AMI 部署在*所有*实例中
- 授予测试脚本，这样您就可以定期运行安全检查并自动化流程。
- 确保第三方软件被配置到最安全的设置中
- 除非绝对必要，请不要以根目录或管理员登录来运行您的进程

在云时代之前的的所有标准安全实践，例如采用良好的编码惯例和隔离敏感数据的，仍然是适用的并应该要实施。

回顾一下，云把您的实体安全的复杂性抽象化了，让您可以通过工具和功能实现控制，这样您就可以保护您的应用程序了。

未来的研究方向

在不久的将来，应用程序将不再受制于物理硬件。就像将微波炉插上电源一样，不需要任何的电力知识，同样的，我们应该能够将一个应用程序插入云中，有如微波炉一样得到所需的电源。作为一个架构师，您将要管理的是抽象的计算，存储和网络资源，而不是物理服务器。应用程序应该要能继续工作，即使底层物理硬件出现故障或者已被移除或被替换。应用程序随着需求模式不断的变化，能够立即自动的部署资源来调整自身，从而实现在任何时候都能达到最大利用率。可扩展性，安全性，高可用性，容错性，可测试性和弹性都是应用程序架构的可配置的属性，它们成为在构建平台时自动与内部组成的一部分。

然而，我们至今仍未实现。现在，您可以采用在本文中强调的最佳实践，在云中构建一些具有这些特性的应用程序。云计算架构中的最佳实践将继续发展，作为研究人员，我们不仅应注重加强云，而且还要注重构建工具、技术和处理，能让开发人员和架构师更容易将应用程序置入云中。

结论

本文为云架构师设计高效的云应用程序提供了规范性指导。

通过专注于概念和最佳实践 – 像故障设计，解耦应用程序组件，理解和实现弹性，将其与并行化相结合，和在应用程序架构的每一个方面中的集成安全 – 云架构师可以明白在构建高度可扩展云应用程序中的必要设计因素。

AWS 云提供了高度可靠的按需付费基础设施服务。本文中的 AWS 特定策略将帮助您使用这些服务设计云应用程序。作为一名研究人员，我们建议您多熟悉这些商业服务，能取人之长，向上提升和进一步创造云计算。

感谢

作者谨向 Jeff Barr，Steve Riley，Paul Horvath，Prashant Sridharan 和 Scot Marvin 致以诚挚的谢意，感谢他们对本文初稿提出的宝贵意见。特别感谢 Matt Tavis 提出的宝贵见解。没有他的帮助，本文将无法问世。

参考文献与深入阅读

1. **Amazon S3 组合，使用 Amazon S3 的最佳实践，**
   http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1904 2008 年 11 月 26 日
2. **Amazon S3 组合，Amazon S3 误差最佳实践，**
   http://docs.amazonwebservices.com/AmazonS3/latest/index.html?ErrorBestPractices.html，2006 年 3 月 1 日
3. **Amazon SimpleDB 组合， 查询 201：Amazon SimpleDB 的查询提示和技巧，**
   http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1232&categoryID=176 2008 年 2 月 7 日
4. **Amazon SimpleDB 组合，使用 Amazon SimpleDB 对性能和可靠性的构建，**
   http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1394&categoryID=176，2008 年 4 月 11 日
5. **Amazon SimpleDB 组合，查询 101：构建 Amazon SimpleDB 查询，**
   http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1231&categoryID=176，2008 年 2 月 7 日
6. **J. Varia，云架构，**
   http://jineshvaria.s3.amazonaws.com/public/cloudarchitectures-varia.pdf，2007 年 7 月 1 日
7. **Amazon 安全组合，安全过程概览，**
   http://awsmedia.s3.amazonaws.com/pdf/AWS_Security_Whitepaper.pdf，
   2009 年 6 月 1 日
8. **Amazon Web Services 组合，使用 AWS 创建 HIPPA 兼容医药数据应用程序**
   http://awsmedia.s3.amazonaws.com/AWS_HIPAA_Whitepaper_Final.pdf，2009 年 4 月 1 日
9. D. Obasanjo，构建可扩展的数据库：各种数据库分片方案的优点和缺点，
   http://www.25hoursaday.com/weblog/2009/01/16/BuildingScalableDatabasesProsAndConsOfVariousDatabaseShardingSchemes.aspx，2009 年 1 月 16 日
10. D. Pritchett，分片课程，http://www.addsimplicity.com/adding_simplicity_an_engi/2008/08/shard-lessons.html，
    2008 年 8 月 24 日
11. J. Hamilton，设计和部署上的互联网规模的服务，2007 年，*21st 大型安装系统管理会议 (LISA '07)*，
    http://mvdirona.com/jrh/talksAndPapers/JamesRH_Lisa.pdf
12. J. Dean and S. Ghemawat，MapReduce：大型集群上的简单数据处理，2004 年 12 月 1 日，在 Proc. 第 6 OSDI 的，http://labs.google.com/papers/mapreduce-osdi04.pdf

13. T. Schlossnagle，*可扩展的网络架构*，Sams 出版，2006 年 7 月 31 日，
14. M. Lurie，整合 – 数据库互操作性，http://www.ibm.com/developerworks/data/library/techarticle/0304lurie/0304lurie.html，2003 年 4 月 23 日
15. E. Hammond，摆脱约束/EC2 上带有开放 VPN 的非信任网络 http://alestic.com/2009/05/openvpn-ec2，2009 年 5 月 2 日
16. R. Bragg，The Encrypting File System（加密文件系统），http://technet.microsoft.com/en-us/library/cc700811.aspx, 2009
17. S. Swidler，如何在一个 EC2 实例上安全地保持您的 AWS 证书，http://clouddevelopertips.blogspot.com/2009/08/how-to-keep-your-aws-credentials-on-ec2.html，2009 年 8 月 31 日
18. **Amazon SQS 组合，使用 Amazon SQS 构建可扩展的、可靠的 Amazon EC2 应用程序，**http://sqs-public-images.s3.amazonaws.com/Building_Scalabale_EC2_applications_with_SQS2.pdf，2008 年
19. Microsoft 支持团队，加密文件系统 (Windows) 的最佳实践 http://support.microsoft.com/kb/223316，2009 年
20. Solaris 安全团队，ZFS 加密项目 (OpenSolaris)，http://www.opensolaris.org/os/project/zfs-crypto/，2009 年 5 月 1 日
21. **Amazon RDS 团队，Amazon RDS 多-AZ 部署，**http://docs.amazonwebservices.com/AmazonRDS/latest/DeveloperGuide/Concepts.DBInstance.html#Concepts.MultiAZ。2010-05-15
22. **Amazon SimpleDB 团队，Amazon SimpleDB 的一致性增强**http://developer.amazonwebservices.com/connect/entry.jspa?externalID=3572 2010 年 2 月 24 日