doi:10.3969/j.issn.1000-436x.2012.z1.025

云应用引擎的资源监控和计费机制研究

任怡[1]，张菁[1]，陈红[1]，吴庆波[1]，孔金珠[1]，戴华东[1]，管刚[2]
（1. 国防科学技术大学 计算机学院, 湖南 长沙 410073；2. 腾讯研究院，北京 100080）

摘 要：在归纳分析已有云计费机制的基础上，选取开源的云应用引擎 AppScale 作为研究对象，分析了其组成与架构，探讨了其资源监控功能的实现机理和不足。修改并扩展了 AppScale 的资源监控代码，设计实现了云应用引擎的资源监控和计费机制 CloudMB，该机制支持进程级资源监控和面向多租户的计费功能。经测试，CloudMB 占用系统 CPU 时间不超过 3%，对平台性能影响较小。

关键词：云应用引擎；平台即服务；资源监控；多租户；计费
中图分类号：TP319 **文献标识码**：B **文章编号**：1000-436X(2012)Z1-0192-09

Research on resource monitoring and billing mechanisms of application engine in cloud computing environment

REN Yi[1], ZHANG Jing[1], CHEN Hong[1], WU Qing-bo[1], KONG Jin-zhu[1], DAI Hua-dong[1], GUAN Gang[2]
(1.College of Computer Science, National University of Defense Technology, Changsha 410073, China;
2. Tencent Research Institute, Beijing 100080, China)

Abstract: A comprehensive study on the billing methods of existing cloud computing platforms was made. Then, AppScale, an open sourced scalable and typical PaaS software, as the foundation platform of our target resource monitoring and billing mechanisms was chose. Appscale's structure, its components and how they worked together was analyzed. CloudMB, which enhanced AppScale by supporting process level resource monitoring and multi-user oriented billing function with flexible billing policies was designed and implemented. Finally, experiment verified that CloudMB is effective and its occupation of CPU time is lower than 3%.

Key words:application engine; PaaS; resource monitor; multi-tenant; billing

1 引言

按照服务交付模式的不同，云计算分为软件即服务（SaaS, software as a service）、平台即服务（PaaS, platform as a service）和基础架构即服务（IaaS, infrastructure as a service) 3 个层次。其中，PaaS 平台作为云应用引擎，为分布式或互联网应用开发者提供一个透明、安全、高效的应用开发和运行环境，提供应用所需的各种核心功能，并支持将开发的应用程序部署到该平台上。目前，商用 PaaS 平台的典型代表包括 GAE(google app engine)、Salesforce.com 的 Force.com、微软的 Azure 等，开源的 PaaS 平台主要是 AppScale。

PaaS 平台资源规模庞大、管理集中、支持大量

收稿日期：2012-08-06
基金项目：国家高技术研究发展计划（“863”计划）基金资助项目（2011AA01A203）；国家科技支撑计划基金资助项目（2011BAH14B02）；国家自然科学基金资助项目（60603063）
Foundation Items: The National High Technology Research and Development Program of China (863 Program) (2011AA01A203); The National Science and Technology Support Program Foundation of China (2011BAH14B02); The National Natural Science Foundation of China (60603063)

用户并发请求，为保证其稳定高效运行，资源监控功能必不可少。资源监控可以实现对平台内各种资源使用和负载情况的有效监测，并为资源动态部署和负载均衡提供依据；通过对系统资源实时监控，提取平台资源使用信息，便于更好地完成系统资源的分配和管理。与此同时，作为商用 PaaS 平台获取和掌握用户使用资源情况的最直接途径，资源监控功能也为客观、公正、合理地进行资源计费起到了不可替代的作用，它是 PaaS 平台向用户收取服务费用的最根本数据来源。

AppScale 的开发和运行接口与 GAE 兼容，是具有代表性的开源云应用引擎平台[1~3]。以 AppScale 为研究平台，分析了其架构及特点，探讨了其资源监控功能的实现机理和不足。在研究已有云计算平台计费策略的基础上，基于 AppScale 资源监控机制，采用 Ruby on Rails Web 架构，设计和开发了具有进程级资源监控和面向多租户的资源计费功能的 CloudMB。

2 AppScale 云应用引擎的组成与架构分析

AppScale 平台与 GAE 的开发和部署接口兼容，实现并扩展了 GAE 的 SDK 及其开放 API。AppScale 可自动、透明地运行于 Amazon EC2 和 Eucalyptus 等云基础架构之上，其目标一方面是在应用部署到 GAE 平台前，向用户提供一个部署、测试、调试、估量、监控 GAE 应用的 PaaS 平台，使其可在自身集群之上运行和验证 GAE 应用，另一方面是在服务、运行时环境与底层云基础设施的互操作方面对 PaaS 平台进行扩展。

AppScale 的开源、兼容 GAE 应用、便于搭建私有云、适于实验等特点，使其适用于针对云计算 PaaS 平台各项功能实现的研究工作。另外，该平台提供了普适而完善的资源监控系统，对研究云计算平台的资源监控十分具有代表性。

2.1 AppScale 平台组成

AppScale 平台由工具包、AppServer（AS）、AppLoadBalancer（ALB）、AppController（AC）、AppDB（application distributed database support）等组成。

工具包提供命令行和 Web 界面工具支持，系统管理员可以通过上述工具配置、启动、停止 AppScale 实例、上传/移除部署的应用、监控资源/应用的状态。该工具支持在基于 Xen 的虚拟集群、EC2 或 Eucalyptus 上部署 AppScale 平台。

AS 是应用程序的执行引擎，它通过 HTTPS 协议与 AppDB 交互来存储和访问数据。每个 AS 一次只可执行一个应用程序。为了托管多个应用程序，可以添加多个 AS。

AC 负责启动、初始化、停止 AppScale 实例，支持 AppScale 各组件之间进行交互（系统中的所有通信均通过 SSL 加密）。

ALB 负责初始化用户与 AS 中运行的应用之间的连接，即在用户成功登录后，将请求路由给指定的、唯一的 AS 以便为该应用程序实际处理请求，ALB 不参与用户应用请求的处理。因此，通过记录用户被指定的 AS 所执行过的应用信息，便可以对用户应用情况进行统计。

AppDB 分为 DBM（database support master）和 DBS（data base support slave）2 种。前者实现 AS 与多种数据存储之间的透明访问，后者代表具体的数据存储服务。

在上述组件的基础上，AppScale 还支持 Memchache、Email、数据存储、内嵌认证、Hadoop 等服务。通过在 AS 与数据存储实体之间引入与 GAE 兼容的协议缓冲服务器（PBS, protocol buffer server），实现了 AS 到多种分布式数据存储机制之间的接口映射。目前，数据存储服务支持 Cassandra、Voldemort、MySQL、MongoDB、MemCacheDB、HBase、HyperTable 7 种数据存储机制。

2.2 AppScale 平台架构

每个 AppScale 实例被部署在一个到多个虚拟操作系统实例（即客户 VM）之上。其中，客户 VM 是运行在 Xen VMM、KVM 或 EC2/Eucalyptus 之上的虚拟 Linux 系统，每个客户 VM 也称一个节点。每个 AppScale 实例中包括一个 ALB、一个或多个 AS、一个 DBM、一个或多个 DBS。其中，ALB 所在的节点被称作是头节点（head node），在一个 AppScale 平台部署中，只能有一个头节点实例。AC 被部署在每个节点上，头节点上的 AC 还负责监控和管理资源的使用、从其他节点周期地收集应用程序信息、节点的使用、根据系统需要和开发人员的要求增加和收缩 AppScale 部署的节点数量、监视系统中是否有故障节点、在需要的时候重启故障组件和重新生成节点。DBM 通过 PB（protocol buffer）Datastore API 访问 PB Server，后者会访问 DBS，不同的 DBS 代表不同的数据存储机制。

AppScale 的组成和部署结构如图 1 所示，系统

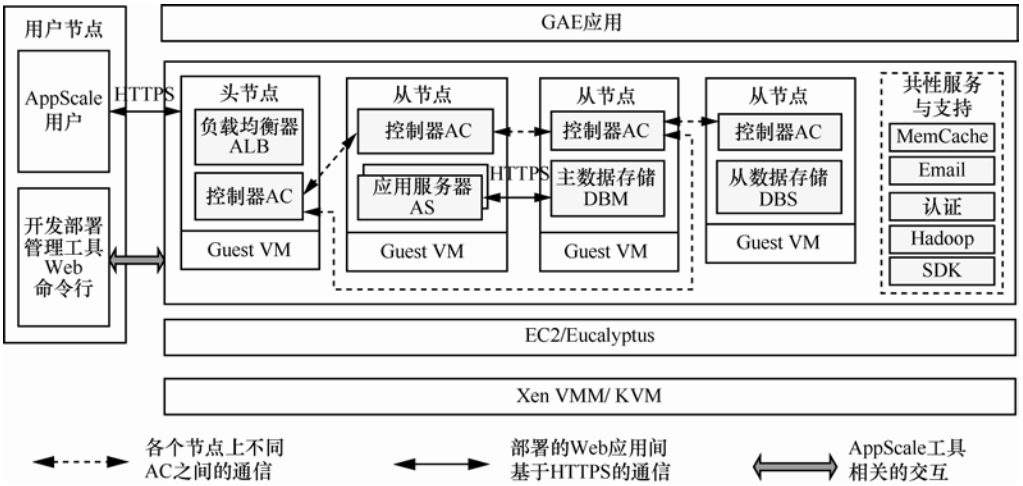


图 1　AppScale 的组成和部署结构

各组成部分之间的通信分为 3 类：1）各个节点上不同 AC 之间的通信；2）AppScale 工具相关的交互；3）用户与所部署 Web 应用之间的基于 HTTPS 的通信。

2.3　AppScale 平台的资源监控

AppScale 平台节点部署采用主从式结构，主节点负责应用的负载均衡，从节点负责计算和存储服务。同样，其监控架构也是采用集中式的，由主节点定时向从节点获取监控数据，并保存在主节点上，而从节点只是定时将监控数据传送给主节点。AppScale 通过整合资源监控工具 Collectd 及环状数据库工具 RRDtool（round robin database tool）实现资源监控、状态存储及资源使用情况的可视化。

Collectd 以后台守护进程的形式运行于各个节点上，主动以周期性的方式从工作节点获取 CPU、内存、磁盘、网络等监控数据。Collectd 具有良好的性能和可移植性，支持网络、RRDtool、Process、Exec 等多种插件，功能灵活多样，但不能将监控到的数据可视化成统计图形。为监控和收集各个节点的资源，需在对应节点加载 Collectd 的相应资源插件；为了将监控数据传送到主节点并保存，则需加载其网络插件，该插件利用多播技术可以保证每个节点之间通信；为利用 RRDTool 在主节点上存储并用图形显示监控的数据，需在主节点上加载 RRDtool 插件，将各节点上收集的数据完整地存储在其固定大小的.rrd 数据库文件中，并支持以图表方式显示。AppScale 资源监控实现原理如图 2 所示。

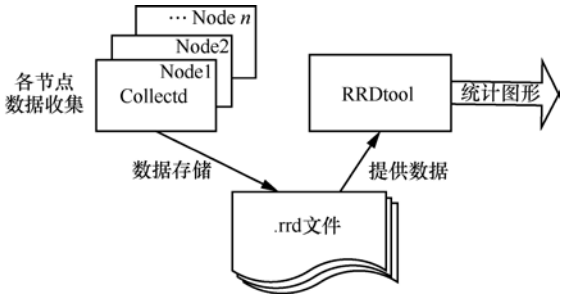


图 2　AppScale 资源监控的实现原理

3　主流云平台计费方法分析

在云计算环境下，众多企业等用户摒弃固有的购买服务器、存储设备的模式，转向选择更富有灵活性的服务模式。在该模式下，云计算提供商亟需解决的重要问题之一就是该商业模式下的计量、定价和收费等核心问题。

传统网络环境中，服务器、存储等设备以私有财产的方式存在，用户只需一次性投入，之后需付给第三方的服务费用主要包括网络使用费等。传统网络计费模式存在基于时长的计费、基于流量的计费、基于服务质量的计费和基于内容的计费等多种模式[4]。其中，基于时长的模式由于具有简单、计费成本低的特点而被广为采用。而考虑到了网络资源的稀缺性，基于流量的计费考虑了资源的实际使用量，基于服务质量的计费考虑应用对网络的不同要求，基于内容计费则考虑了各种业务所具有的不同商业价值。

在云计算环境下，计费不再只是对网络资源进行计费，而是涉及更多的软硬件资源，更多的资源

类型和计费因子给云计算平台的计费带来了挑战——从所需存储空间、到所使用的时间周期、再到每个月的流量分配、以及 SLA 隐性因素等，还需要考虑到不同的服务类型和特色。

3.1　IaaS 平台的计费方法

IaaS 云平台的重要推动者 Amazon 公司向用户提供了弹性计算云 EC2、简单存储服务 S3、弹性 MapReduce、CloudFront、SimpleDB、关系数据库服务、简单队列服务 SQS、云资源监控服务 CloudWatch、虚拟私有云等多个层面的云产品。其中，在 IaaS 层面，弹性计算云 EC2 和简单存储服务 S3 成为其营收的两大来源，这 2 种服务的计费模式很大程度上代表了 IaaS 平台的主流模式。

Amazon 将全球划分为若干个地理区域，对于不同区域，其数据中心提供的服务收费有所区别。Amazon 提供费用计算器供用户评估需求与预算，支持针对不同产品的计费策略，提供计费和支付服务 Amazon FPS 和 DevPay。由于 EC2 和 S3 本身产品形式不同，其计费模式也不同。

1) Amazon EC2 的计费[5]

Amazon EC2 提供给用户的是计算资源，其交付物表现为定制的虚拟机实例。EC2 向用户提供按需型（on-demand）、预约型（reserve）和限价型（spot）三个类型的计费模式。

按需型：无须支付预付费用，按时长计费，且可通过 Auto Scaling 功能自动增删所租用的虚拟资源，适用于“即买即用”的短期用户。

预约型：按时长计费，与按需型不同之处在于需签订长期合同（如 1 年），并预支一定费用，此后的实际使用中仍按时长收费，但平均单价低于按需型收费，服务等级高于按需型。

限价型：根据供求情况周期性地发布即时价格，而用户也给出可接受的最高价格。当用户最高限价低于即时价格时，系统自动终止服务，否则为用户提供所需服务。该计费模式使得用户以较低价格充分利用系统的闲散资源，适合于需要大量计算能力但对计算响应要求不高的用户。

Amazon EC2 提供标准、小型、高内存、高 CPU、集群、集群 GPU 共 6 种类型的定制虚拟机实例，每种实例按照不同规格还可细分为若干种，每一种按时长计费的定价有所区分。目前，定制虚拟机所支持的操作系统有 Redhat 企业级 Linux、Windows Server、Oracle 企业级 Linux、SUSE 企业级 Linux 和 Amazon Linux AMI，同时安装了 DBMS、Web 服务器、资源管理、应用服务器、媒体播放等软件。

2) Amazon S3 的计费[6]

Amazon S3 是一种安全、可靠、可伸缩的在线存储服务，用户可通过授权访问其标准 Web 接口来随时存储及获取存放在 Amazon 数据中心中的数据。使用 S3 的费用包括存储费用、请求费用和数据迁移费用。存储费用由选定的数据中心、服务质量等级和数据量确定；请求费用由选定的数据中心、各种存储请求的种类和数量确定；数据迁移费用则是由数据量决定（受数据源和目的地影响）。

目前 S3 提供 2 种服务质量等级。

标准存储（standard storage）：提供 99.99%的可用性保障，一年的数据丢失率不高于 0.000 000 001%，可以从 2 个存储设备同时失效中恢复，该级别用以存储关键数据。

去冗余存储（RRS, reduced redundancy storage）：提供 99.99%的可用性保障，一年的数据丢失率不高于 0.01%，可以从一个存储设备的失效中恢复，价格相对便宜，该级别用于存储重要程度相对较低的数据，如图片缓存等。

目前，针对 S3，Amazon 将数据量从 1TB 到数千 TB 以上划分为若干计费区间，不同区间价格不同；数据量越大，存储费用单价越低。

对访问请求计费的原因很大程度上是对基于 S3 的分布式存储应用开发者的一种制约，促使其采用访问数较少的设计，达到优化 S3 访问量的目的，价格相对低廉。将请求分为 2 种定价，第一种包括 PUT、COPY、POST、LIST 请求，第二种包括 GET 和其他请求。

数据传入、传出流量主要是指跨数据中心的数据迁移费用，同一数据中心内部的数据迁移不收费。数据迁移采取单向收费原则，通常数据传入流量单价按 0 计算，数据传出流量单价随着数据量的增大而降低。

3.2　PaaS 平台的计费方法

与 IaaS 层相比，PaaS 平台的计费不仅仅局限于虚拟机操作系统或存储等层面，还涉及应用层的进程、调用等细粒度的资源监控和计费。目前，在 PaaS 领域，GAE 占据着主导地位，其计费模式也具有典型的代表性[7]。

GAE 早期采用较为简单的计费标准，分别对传出/传入带宽（以千兆计）、CPU 时间（以小时计）、

存储数据（以千兆字节×月计）、接收电子邮件的接收人等计费，低于限额以下免费使用，超出部分单价固定且收费的资源单位粒度较大。支持配额管理，允许用户设置最高预算阀值，从而资源使用会控制在预算以内。

2011 年 9 月，Google 宣布将结束预览期，正式对外收费，其计费模式和资费标准也发生了调整。新的收费模式下，收费对象调整为输出带宽（以千兆计）、前端/后端/打折实例（以 instance-hours 计，其中后端实例分为 4 个级别，不同级别价格不同）、数据存储（分级别以千兆字节×月计）、channel（以打开的 channel 数计）、邮件收件人（以 Email 数计）、XMPP（以 stanza 计）等。

其中最主要的变化表现为从按 CPU 使用时间付费（CPU cycles）转向按进程实例时间（instance-hours）付费。在新的计费模式下，即使某个应用大部分时间处于空闲状态，如 I/O 等待中，用户仍需要为运行的所有应用实例付费，不同于早期的计费标准，多个进程并发执行会被分别计费。另外，新模式的免费资源用量和 API 调用数有所下降。总之，新计费模式的资源粒度更加细化，费用有所提升。

为了防止费用增加带来的用户流失，GAE 在功能和工具方面有所改进，支持用户付费管理、查看付费历史、优化资源使用、用量报告以及相应的分析工具。

与 AWS（amazon Web services）相比，在 CPU 使用方面，GAE 按进程实例用时计费，而 AWS 则以虚拟机实例用时计费，前者粒度更细，即使进程在空闲或等待过程中仍需收费。因此，在多进程并发情况下，为减低平台使用成本，用户可选择 AWS 平台或者优化自身程序以提高 CPU 利用率。

3.3 SaaS 平台的计费方法

SaaS 平台通过 Internet 向用户提供应用级软件服务，用户可根据其需求订购所需服务。SaaS 模式下，用户的使用方式发生了变化，从传统的拥有软件向租赁软件发展。SaaS 支持多租户的方式来降低提供商软件部署费用，通过规模经济和专业化来降低供应商软件服务的成本及用户费用。

由于应用软件的丰富多样性，SaaS 平台的计费需考虑不同服务中的各个计算点的分析和控制，另外，还需考虑多租户前提下各个用户的使用生命周期以及系统如何进行定价和组装[8]。从通用的角度看，SaaS 平台的计费一般遵循按需订阅、按量计费的原则，提供商与用户在一定协议期内，在每月订阅费用上达成一致。根据使用人数或数据量等，可支持对计费和定价进行优化。

这种按月计费的模式取代了以往初期软件投资成本高的传统软件使用模式，有助于以较低的花销快速部署运行所需软件，并获得更好的可伸缩性，降低维护成本。

3.4 分析与小结

可以看出，云平台计费具有以下特点。

1) 计费因子多。包括 CPU、内存、存储、数据传输、虚拟机实例等多种基础资源的使用计量。

2) 灵活的计费策略。可根据服务等级、资源子类型细化计费因子，并可根据所处地区、时间段、资源消耗量等的不同采用多种计费方式，支持通过非线性、分级等灵活定价策略平衡稀缺资源和用户使用量关系。

3) 支持多租户。多租户是公有云使用的基本特征之一，支持多租户资源使用计量的独立性和隔离性。

4) 提供计费管理辅助工具。提供需求与预算评估、计费分析等软件供用户使用，界面简单、易操作。

4 CloudMB 的设计与实现

对外租赁使用是云平台区别于网格等分布式计算环境的一个重要特征，该使用模式要求云平台能支持多租户使用和计费。作为开源云应用引擎平台，目前 AppScale 尚未实现资源计费功能，其资源监控也是以单个节点为基础，不支持面向多租户的资源监控。改进了 AppScale 1.5 的资源监控机制，基于 AppScale 1.5 源代码实现了资源监控和计费软件 CloudMB，该软件针对云计费的多租户需求，采用进程级资源监控的方式，从而支持多租户资源使用的细粒度计量。

4.1 设计思路

4.1.1 资源监控粒度

云平台通常由众多计算集群组成，集群软件安装在虚拟机或物理机上，虚拟机则部署在物理机之上。云平台的资源监控通常包括 2 种粒度：一是对集群、物理机以及其上部署的虚拟机的数量、运行状态等的监控，主要用于向用户提供全系统总体可用资源信息；二是对虚拟机/物理机内部资源使用等情况的监控，该类监控信息可作为平台自动部署、系统性能分析的依据。

目前，AppScale 提供的主要是以虚拟机/物理机为单位的单节点资源监控。在每个节点内部，提供 CPU、内存、存储等资源甚至更细粒度的使用及特有属性信息，但并不针对每个租户加以区分。每个租户部署在 AppScale 平台上的应用是以实例的形式存在的，而一个实例对应一个进程。为了面向多租户实现计费，需要跟踪进程的资源状态和使用信息，即实现进程级的资源监控，再通过合理的计费策略进行计量，最终获得用户的使用费用。

4.1.2　计费策略及其有效性分析

CloudMB 实现中用以计算租户 U 的应用 *app* 在第 k 个节点上的资源使用费用的基本公式如下

$$C_U^{app_k}=\sum_{i=1}^{m}\left(\sum_{j=1}^{k}\left(C_{ij}W_{ij}\right)\right)P_i$$

其中，系统中各种资源共有 m 类，每个资源又可以细分成更小粒度的多种资源，例如 CPU 用时可根据进程实例进一步细分为前端、后端等多种，网络流量可分为流入和流出等，C_{ij} 是第 i 类资源的第 j 种细分资源的消耗量，W_{ij} 是第 i 类资源的第 j 种细分资源的计费权重，P_i 是第 i 类资源的单价。CloudMB 调用 Collectd 统计资源使用量，使用 RRDtool 存储上述信息，并支持通过时间、已消耗资源量等系统状态信息动态自动配置 W_{ij}。

MB 支持可定制的计费策略。通过对参数进行配置，系统管理员可根据自身计费策略设定各类资源的权值、各类资源单价、查询时间范围等，并支持根据时间段、资源消耗量等值的变化来动态确定计费参数，从而能够满足面向不同应用类型的多种用户的需求，提供灵活的计费策略支持。

4.1.3　主要功能分析与设计

云应用引擎平台向应用开发者提供开发和运行接口，允许后者开发、调试、部署、运行面向最终用户的分布式应用程序；平台本身的资源管理、监控恢复、性能调优、结构扩展等则由平台管理员来负责。因此，云应用引擎平台主要面向 2 类用户：应用开发者和平台管理员。从这 2 类角色出发，平台计费需满足以下功能。

平台管理员：登录计费功能软件，管理应用开发者的账号；查看周期时间内平台计费汇总以及消费清单；支持根据某种计费因子对应用的资源消费进行排序；增删计费因子；修改计费权重或编写计费权重设置脚本，从而支持根据系统参数动态自动改变计费权重，实现灵活的计费策略；查询每个节点的资源使用情况和系统性能参数。

应用开发者：登录计费功能软件；查看所部署应用的清单；查看周期时间内应用的消费清单及汇总；按消费量对应用进行排序；回溯历史消费清单；查看每个应用的运行概况，包括请求数、响应延迟等。

4.2　结构和组成

基于 AppScale 开发的资源监控和计费软件 CloudMB 主要由资源监控模块、数据组织与存储模块、资源计费模块和 Web 界面 4 部分组成，如图 3 所示。其中，资源监控模块负责单节点进程级资源监控及全系统资源使用情况汇总。资源计费模块根据资源监控模块产生的原始数据，整合生成用户应用的资源消耗数据，根据计费策略、单价和使用量计算费用。数据组织与存储模块保存了以.rrd 文件形式存储的原始监控数据以及以关系数据库表形式存储的用户、应用、计费数据等信息及其关联信息。Web 界面由 Ruby on Rails 实现，包括登录管理、节点监控信息、用户应用监控信息、计费信息等的显示。

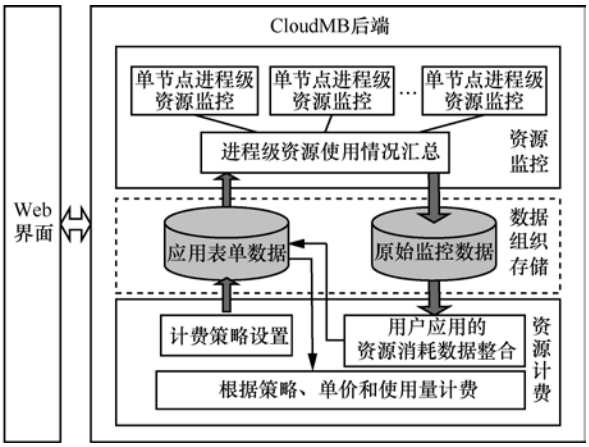


图 3　CloudMB 的结构和组成

4.2.1　资源监控模块

目前，AppScale 不支持面向用户应用的进程级的资源监控，为此需对其源码进行改进。注意到每个租户通常部署有一到多个应用，AppScale 在同一节点上为同一应用创建 3 个进程（以冗余换取可靠性）。因此，当系统中应用名唯一时，要获取某用户的某个应用的资源使用情况，可通过计算单个节点中此用户对应的该应用所有进程的资源使用情况，并将各节点使用情况进行累积即可。

Collectd 的 Exec 插件支持运行用户自定义的脚本，通过自定义脚本可收集所需信息并写入 RRDtool 数据文件中。利用 Linux 自带的进程监控工具编写了 app_monitor.sh，该脚本主要流程如下：

1) 查看主机名称；
2) 设定 Collectd 监控时间间隔；
3) 通过传入方式获取被监控的应用名；
4) 获取应用对应的进程信息；
5) 初始化 CPU、内存、磁盘等资源参数；
6) 获取各个进程的资源使用量；
7) 计算此节点该应用使用的各种资源量；
8) 将计算出的数据按照监控时间间隔写入数据库文件中。

接着，在 Exec 插件配置文件指定运行 app_monitor.sh，并在 collectd.rb 文件的 write_app_config 函数中加入应用和自定义脚本信息即可实现对单节点的进程级资源监控。在此基础上，通过 Collectd 及其网络插件以及 RRDtool，实现对整个系统的进程级资源监控。

4.2.2 资源计费模块

资源监控模块为资源计费提供了原始的计费数据，利用这些数据进行整合、处理和计算可以获得用户的各个应用的资源使用情况，然后根据计费策略和单价即可计算出各种资源的使用费用。

由于 Collectd 周期性监控数据，RRDtool 中存储的是平台实时运行数据的周期性采用，需对其进行处理以得到时段性数据。为此，对不同的资源采用不同的计算方法。例如，进程时间通过该进程运行时间、CPU 频率、各个采样点的 CPU 使用率等计算而来；内存使用量的采样点形成了与时间轴的不连续函数，通过求解近似曲线函数与横轴的面积，然后再除以时长，即得到内存较为准确的使用量；网络流量采用累加的方式。目前，磁盘使用量不进行细分，按照实际分配给租户的量计算。

CloudMB 中拥有权限的平台管理员可设置计费权重。系统将根据某些参数（如时间段）的变化而动态调节计费权重，从而支持灵活的计费策略。目前支持通过修改配置脚本来设置计费权重等。

整合出的统计数据、计费权重等被写入应用表单数据库，该数据库中还存储有用户、用户与应用、应用与进程之间的关联信息以及各种资源的单价，通过读取这些信息，并进行计算，CloudMB 资源计费模块完成资源的计费。

资源的使用量以天为单位存储在系统中，根据请求，可对某租户的某个或几个应用的总资源使用量进行统计，系统支持历史账单的查询。

4.2.3 数据组织与存储模块

数据是计费的基础，CloudMB 中有 2 类数据：一类存储的是从各个节点收集的各类资源的原始监控数据以及这些数据的初步汇总信息；另一类平台管理员和应用开发者的账号信息、部署的应用信息、计费策略设置信息、资源单价、整合的计费数据、历史账单等应用相关信息。

第一类数据的数据量更新频率高、在不断变化中，累积数据量大，常规数据库的结构对于这类数据不适合；第二类数据相对稳定、需要持久性存储，且涉及用户体验，要求较高的存取效率。2 类数据具有不同特点，其存储方式也有所差异。

通过分析 RRDtool 数据库的特点，发现该数据库适合动态监控数据的存储，可在一定程度上整合监控数据而不丢失监控的真实性，且其数据库文件具有循环记录的特点，其大小固定，能有效控制存储空间的大小，有利于存储可扩展性管理。因此，对原始监控数据及其初步整合的数据存储在 RRDtool 中，而应用相关信息则选取 MySqL 存储。

为方便原始数据的存储和查询，按照数据的层次关系设置数据库表，主要数据库表包括节点表、资源类别表、细化资源表和细化资源参数配置表。其中，每张表都包含有对应层次的实体名称、ID、计量、创建时间和修改时间，以适应其动态更新的特点。应用相关的数据对应的数据库表则主要包括管理员账号表、应用开发者账号表、应用−进程关联表、计费权重表、资源单价表、历史账单表等。

因此，数据组织与存储模块包括原始监控和应用相关 2 部分数据，该模块与资源监控模块和资源计费模块交互，为 Web 界面显示提供数据依据。

4.2.4 Web 界面显示

CloudMB 前端采用 Ruby on Rails 实现 Web 网页显示功能，包括登录管理、节点监控信息的显示、节点计费显示、用户应用的计费显示、历史账单显示等。为简化实现，采用与 AppScale 资源监控相同的局部模板。当有页面请求到来时，首先到应用相关信息数据库中查询，若查到相关数据，则随即在页面中显示，若无所需信息，则需从原始监控信息数据库中调取相关信息进行整合和计算，然后保存在应用相关数据库中，并在 Web 页面中显示。

4.3　系统代价分析

资源监控和计费机制作为辅助功能，理想的状况是占用尽可能少的平台资源。目前，CloudMB 对系统潜在的影响主要在于以下 3 个方面：1）Collectd 周期性采集数据并存入数据库，被监控资源细分种类多，采样周期短，会带来一定的 I/O 和 CPU 损耗；2）Collectd 将各节点的监控数据汇总到主节点，由于原始监控数据更新周期短、种类多，就使得主节点网络带宽受到影响；3）从原始监控数据到汇总数据以及最终计费数据过程中的整合、计算代价。

实验的硬件环境为 Intel(R) Core(TM)2 DUO CPU T5 870 CPU、2G 内存、250G Sata 硬盘。软件环境为 Ubuntu 10 操作系统、Ruby1.8.7 和 Rubygem2.4.2。Ubuntu 中共部署 3 个虚拟机，平分系统内存。平台中部署了 4 个节点，一个主节点（节点 1），2 个从节点（节点 2 和 3）和一个数据存储节点（节点 4）。测试时，分别针对系统轻负载和重负载 2 种情况，以 4 个不同组成节点的 CPU 占用率作为观察对象，对比在未运行 CloudMB 与运行了 CloudMB 之后的 CPU 占用率，如图 4 所示。2 种情况下，分别测试 50 次，取 CPU 占用率的平均值。

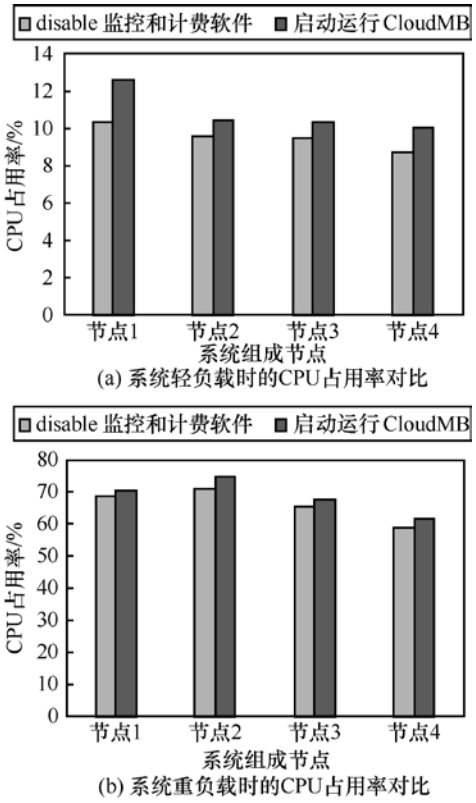


图 4　使用 CloudMB 前后系统 CPU 占用率的对比

无论在平台轻负载或是重负载下，未部署运行 CloudMB 时平台消耗的 CPU 资源比部署运行 CloudMB 时消耗的 CPU 资源要少，但总的差异不超过系统 CPU 总量的 3%。

5　结束语

本文在研究已有云计费机制以及 AppScale 源码的基础上，改进了 AppScale 1.5 的资源监控机制，基于 AppScale 1.5 实现了资源监控和计费软件 CloudMB，该软件针对云计费的多租户需求，采用进程级资源监控的方式，对多租户细粒度资源使用计费进行了有益的探索。下一步将研究如何确定较优的监控数据采样周期，考虑服务质量以及其他细化因素，建立更为合理的计费模型和算法。

参考文献：

[1] CHOHAN N, BUNCH C, PANG S, *et al*. AppScale: scalable and open appengine application development and deployment[A]. Proceedings of First International Conference on Cloud Computing[C]. Berlin, Germany, 2009. 57-70.

[2] CHOHAN N, BUNCH C, PANG S, *et al*. Appscale design and implementation[EB/OL]. www. cs.ucsb.edu/~ckrintz/papers/appscale 2009-02TR. pdf, 2009.

[3] BUNCH C, CHOHAN N, KRINTZ C. Appscale: open-source platform-as-a-service[EB/OL]. http://www.cs.ucsb.edu/research/tech_reports/ reports/ 2011-01. pdf, 2011.

[4] 唐国芳. 基于 QoS 服务级别的内容计费及其在 3G 中的应用[D]. 南京：南京邮电大学,2008.
TANG G F. QoS based Content Billing and Its Applications in 3G Services[D]. http://Nanjing: Nanjing University of Posts and Telecommunications, 2008.

[5] Amazon EC2 pricing[EB/OL]. http://aws. amazon.com/ec2/ pricing/, 2011.

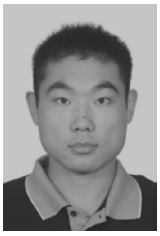
[6] Amazon simple storage service (amazon S3) pricing[EB/OL]. http://aws. amazon.com/s3/#pricing,2011.

[7] Google app engine-pricing and features[EB/OL]. http://www.google.com/enterprise/cloud/appengine/pricing. html,2012.

[8] Meter, price & bill for cloud services[EB/OL]. http://www.zuora.com/why-zuora-subscription-management/technology.html,2011.

作者简介：

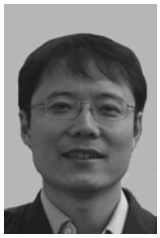
任怡（1977-），女，陕西西安人，博士，国防科学技术大学副研究员，主要研究方向为系统软件、分布式计算与中间件、云计算技术等。



张菁（1987-），男，山东济南人，国防科学技术大学博士生，主要研究方向为系统软件、数据存储和云计算技术。

陈红（1987-），男，湖南益阳人，国防科学技术大学硕士生，主要研究方向为云环境中的资源管理和计费机制。

吴庆波（1969-），男，浙江宁波人，博士，国防科学技术大学研究员，主要研究方向为系统软件、高性能计算和嵌入式系统。



孔金珠（1974-），男，山西闻喜人，硕士，国防科学技术大学副研究员，主要研究方向为系统软件、虚拟化和高性能计算。

戴华东（1975-），男，湖北随州人，博士，国防科学技术大学研究员，主要研究方向为系统软件和高性能计算。

管刚（1982-），男，湖北蕲春人，硕士，腾讯研究院副总监，主要研究方向为面向公众服务的网络操作系统技术。

（上接第 191 页）

[13] ROSA-VELARDO F, MARROQUIN-ALONSO O, DE FRUTOS-ESCRIG D. Mobile synchronizing petri nets: a choreographic approach for coordination in ubiquitous systems[J]. Electronic Notes in Theoretical Computer Science, 2006, 150(1): 103-126.

[14] 张永晖. 基于用户行为的下一代移动互联网络若干关键问题的研究[D].长沙: 中南大学, 2010.
ZHANG Y H. Key Technologies Research of Next Generation Mobile Internet Based on User Behavior[D]. Changsha: Central- South U., 2010.

作者简介：

张永晖（1973-），男，湖南长沙人，博士，福建工程学院讲师，主要研究方向为移动互联网接入和容迟网络。

林漳希（1953-），男，福建漳州人，德克萨斯理工大学博士生导师，主要研究方向为 QoS 路由和网络安全。

刘建华（1967-），男，江西南昌人，博士，福建工程学院副教授，主要研究方向为路由和高性能算法。

梁泉（1972-），男，湖南洞口人，博士，福建工程学院副教授，主要研究方向为移动网络和机会网络。