UbiCloud:一种面向普适终端的云计算系统

陈援非[1] 崔 丽[2] 朱珍民[1] 曾 益[1] 吴元昆[1]
(中国科学院计算技术研究所 北京 100190)[1] (中国青年政治学院 北京 100089)[2]

摘 要 针对低性能终端的复杂计算需求,设计了面向普适终端的云计算系统UbiCloud。通过对云中计算资源的描述、发现、访问、评价等问题进行研究,在前端普适终端与后端云之间构建虚拟计算环境,实现了一个异构终端的云计算服务发布及访问平台。在对服务质量的评价上结合AHP算法给出了一个有效的服务评价模型,在建立判断矩阵的过程中提出了两种一致性矩阵构建方法。最后对系统进行了性能测试,结果表明系统在应用响应时间等指标方面可以达到设计目的。

关键词 普适计算,终端,云计算,SoA,UPNP
中图法分类号 TP316.4 **文献标识码** A

UbiCloud:A Cloud Computing System for Ubiquitous Terminals

CHEN Yuan-fei[1] CUI Li[2] ZHU Zhen-min[1] ZENG Yi[1] WU Yuan-kun[1]
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[1]
(China Youth University for Political Sciences, Beijing 100089, China)[2]

Abstract A cloud computing system, UbiCloud, was developed to enable the ubiquitous terminals access to powerful and reliable computing resources anywhere and anytime through building a virtual computing environment between the front-end ubiquitous terminals and the back-end servers (cloud). The experiment results show that the system performance is good enough to support most applications deployment on resource-poor terminals.

Keywords Pervasive computing, Terminal, Cloud computing, SoA, UPNP

1 引言

普适计算的特点是将集计算、通信、传感功能于一身的各种信息设备通过网络有效组织起来,并按照用户的个性需求进行定制,以嵌入式产品的方式呈现在人们的工作和生活中,为人们提供一种随时、随地、随环境自适应的信息服务。其最终目标是将由通信和计算机构成的信息空间与人们生活和工作的物理空间融为一体。

在普适计算环境中,终端的形态千差万别,但大致上可分为资源贫乏的终端(或称弱终端)和资源丰富的终端(或称强终端)两类。典型的弱终端,如移动电话、PDA、瘦客户机等,在处理器速度、内存大小、磁盘容量等方面相对普通的桌面计算机和服务器要弱得多。未来,从用户使用方便的角度来看,弱终端在体积上不能过大、过重,电池使用时间也不能过短。虽然嵌入式技术在不断发展,但弱终端将永远是资源相对受限的硬件,因此用户在弱终端上运行一些复杂的应用时,将会觉得力不从心。

同时,也应注意到存放在企业中的大型服务器的数据及软件的使用率很低,造成大量的资源浪费。因此,云计算技术应运而生。云计算是基于互联网的计算,可以将各种计算资源、软件和数据提供给前端设备,包括弱终端和强终端。终端通过使用云端发布的资源来完成自己的工作,如编辑文件、存取数据等。云计算技术使得服务器的计算资源得到充分的使用,降低了前端设备(尤其是弱终端)的成本和能耗。

我们针对低性能终端的复杂计算需求,设计了面向普适终端的云计算系统UbiCloud(Ubiquitous Cloud Computing),目的是通过对云中计算资源的描述、发现、访问、评价等问题进行研究,在前端普适终端与后端服务器(云)之间构建虚拟计算环境,使普适终端可以随时随地获取强大可靠的计算资源。

本文第2节介绍了UbiCloud系统架构;第3节详细叙述了UbiCloud系统的实现细节;第4节对系统的性能进行了测试,并对数据进行了分析;第5节介绍了其他研究者的一些相关工作;最后进行了总结。

2 系统架构

UbiCloud是针对弱终端对复杂计算资源的需求而设计的一个云计算系统,可以实现在弱终端上对服务器端软件的无缝调用和资源融合。为了构建UbiCloud,我们在一个服务器集群中建立了多个虚拟机,组成虚拟机池;然后每个虚拟机

到稿日期:2010-12-31 返修日期:2011-04-15 本文受新一代宽带无线移动通信网重大专项(2009ZX03001-019-02),中国青年政治学院2010—2011年度科研基金项目(1890408)资助。

陈援非(1976—),男,博士,助理研究员,主要研究方向为普适计算、虚拟化,E-mail:cyf@ict.ac.cn;崔 丽(1977—),女,硕士,讲师,主要研究方向为普适计算、泛在教育。

上通过 UPNP 协议发布了多个应用，构成服务池；普适终端在需要调用计算资源时，通过 UPNP 协议动态发现网络中适配的服务，然后通过用户端虚拟化协议（例如 SeamlessRDP）进行访问（见图 1）。

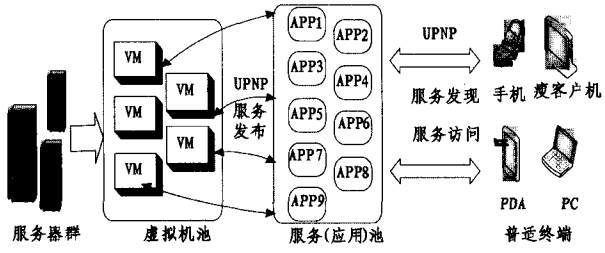

图 1　UbiCloud 系统架构

Ubicloud 核心系统由 3 部分组成，分别为：服务器（组）、客户端和注册中心（见图 2），其中：

服务器（组）（UbiCloud Server，UCS）：负责维护应用软件池，并通过 UPNP 协议发布应用软件的描述文档（XML）。UCS 上还有一个 QoS 探测模块，用来探测服务器的当前状况，以提供最优服务推荐；

客户端（UbiCloud Client，UCC）：通常为弱终端，通过 UPNP 协议发现服务器发布的应用软件，然后利用用户端虚拟化协议获取应用窗口句柄，并与客户端 GUI 和文件系统融合；

注册中心（Registration Center，RC）：负责收集服务器 QoS 指标，然后进行服务质量评价。

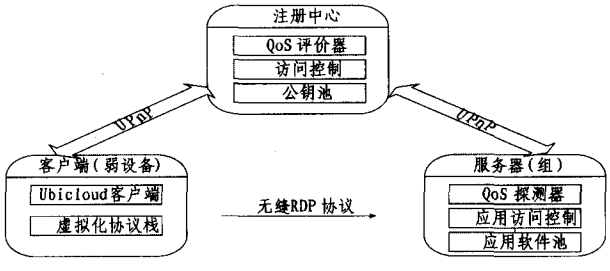

图 2　UbiCloud 系统的 3 类设备

3　系统实现

在普通的本地计算模式下，从“双击一个文件”到“调用应用程序将其打开”这一过程中，操作系统做了许多工作。虽然不同的系统，其实现的方式存在一些差别，但大部分系统都包含如下操作步骤：

（1）探测所双击的文件属于哪种类型：对于 Linux 系统，一般采用 MIME 类型来判断文件；对于 Windows 系统，一般采用文件后缀来判断文件；

（2）根据探测得出的文件类型查找关联表，获得支持该文件类型的应用程序地址；

（3）调用应用程序并将文件地址作为其输入参数，将文件打开。

UbiCloud 将这个过程从单一计算机模式扩展到分布式计算模式。UCS 负责将应用描述和发布为 Web 服务。RC 收集并创建应用程序以描述 XML 和 MIME 类型支持表。当用户打开一个 UCC 的文件时，UCC 上的 UbiCloud 客户端将 MIME 类型发送到 RC，并得到了应用服务的 URL。然后 UCC 通过虚拟化协议调用应用程序的 URL，打开用户的文件，并最后将应用程序界面显示给用户。

UbiCloud 的分布式计算模式的优势是显而易见的：弱终端计算能力一般相对较弱，如果可以方便、无缝地调用远程应用程序，就没有必要将应用软件安装在弱终端上。此外，运行用户端虚拟化协议的开销比一般运行复杂应用程序的开销要低得多，因此可以大大降低终端的硬件成本。

3.1　服务描述

应用软件是一种计算资源，我们需要知道该资源的一些基本信息，如：服务类型、调用参数、支持的 MIME 类型等信息；同时我们还需要了解该计算资源的访问方式，即如何调用该应用程序，例如服务的协议、IP 地址和端口等。

在 UbiCloud 中，服务的描述主要是注册中心响应用户命令服务和 QoS 状态通告服务。在系统设计中，为使用户调用更加简单，对于注册中心端的服务描述，我们将用户命令设计成统一的名字，调用时依据参数的不同来实现不同的命令调用。图 3 是注册中心服务描述文档的一个例子。

```
〈? xml version="1.0"?〉
〈scpd xmlns="urn:schemas-upnp-org:service-1-0"〉
〈actionList〉
  〈action〉
    〈name〉MIMETYPE〈/name〉
    〈argumentList〉
      〈argument〉
        〈name〉MIMETypeIn〈/name〉
        〈retval/〉
〈relatedStateVariable〉MIMETypeIn〈/relatedStateVariable〉
        〈direction〉in〈/direction〉
        〈name〉MIMETypeOut〈/name〉
        〈retval/〉
〈relatedStateVariable〉MIMETypeOut〈/relatedStateVariable〉
        〈direction〉out〈/direction〉
      〈/argument〉
    〈/argumentList〉
  〈/action〉
〈/actionList〉
  〈serviceStateTable〉
  〈stateVariable sendEvents="yes"〉
    〈name〉MIMETYPE 〈/name〉
    〈dataType〉string〈/dataType〉
    〈defaultValue〉〈/defaultValue〉
  〈/stateVariable〉
〈/serviceStateTable〉
〈/scpd〉
```

图 3　服务描述文档模板

3.2　服务发现

在 UbiCloud 中，服务的发现需要分两个部分来实现：（1）注册中心发现服务器；（2）用户发现注册中心。注册中心负责服务的认证评价以及对用户合法性的认证工作。

（1）注册中心发现服务

注册中心发现服务的过程也是计算服务向注册中心发布的过程，使用 UPnP Notify 消息及 UPnP M-Search 消息来通告设备的存在。当注册中心发现设备之后，设备将其服务全部加密后宣告给注册中心。具体的服务发现过程如下：

第一步　服务器向注册中心广播 Notify 消息或是注册中心广播 M-Search 消息；

第二步　服务器收到 M-Search 消息或是注册中心收到

Notify 消息后，服务器向注册中心发送设备描述文档；

第三步　注册中心解析设备描述文档，获取 QoS 服务，如果该服务不存在，则认为是不合法的设备，忽略该设备并回到第一步等待其他设备的加入；

第四步　读取 QoS 服务的服务描述文档得到该设备的 QoS 信息，之后按序获取其他计算服务的描述并忽略它们的服务描述文档下载工作；

第五步　服务发现结束，将该设备加入到控制列表中，并返回到第一步等待其他设备的加入。

(2)用户发现注册中心

在 UbiCloud 中，注册中心对用户来说相当于一小片云：在用户需要服务时，往注册中心发送一个请求即可，其他工作都由注册中心完成。为了让这种服务请求能够无缝进行，在设计时也采用 UPnP 互联协议，整个过程不需要用户的参与。

3.3　服务访问

UPnP 服务访问方式采用 SOAP 协议，通过将 SOAP 消息嵌入到 HTTP 消息体中传给服务响应端。在 UbiCloud 中，服务的访问过程在 SOAP 的基础加入了 QoS 的定时通告及用户端虚拟化服务访问技术。图 4 说明了该服务框架的大体服务流程，其具体访问过程通过以下步骤来完成：

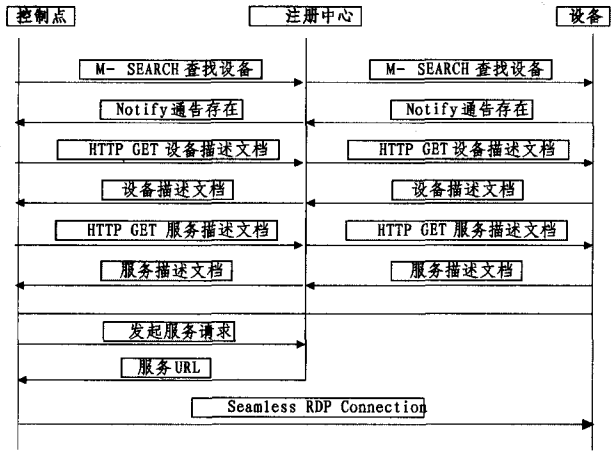

图 4　UbiCloud 服务流程

第一步　各服务器端定时向注册中心宣告它当时的 QoS 状态消息；

第二步　注册中心接收到服务器的 QoS 信息后，更新对应服务器的 QoS 信息表；

第三步　用户向注册中心发送服务请求命令；

第四步　注册中心判断用户命令是否合法，合法则继续，否则返回，等待用户命令；

第五步　注册中心检测完命令合法性后，搜索当前服务器列表，看是否有服务器可以提供该服务，有则判断该用户是否具有使用权，有使用权则将其加入到临时 QoS 评价列表中；

第六步　注册中心对得到的 QoS 列表进行评价，对提供该服务的服务器进行 QoS 评分；

第七步　注册中心将 QoS 列表中的所有服务器或是 TOP10 加上 QoS 分值后提供给用户，注册中心工作完成，等待下一用户命令；

第八步　用户接到注册中心返回的服务器列表后解析呈现给用户，并将最好的服务器设为默认服务器推荐给用户使用；

第九步　用户使用推荐服务器或是自己选取一个服务器进行服务，具体的服务方式依据具体的服务内容而定：采用直接解析结果或是用户端虚拟化技术等；

第十步　完成一次服务过程，返回第一步等待用户命令。

在 UbiCloud 中，服务的访问过程与普通的 UPnP 访问方式有些不同，采用了 QoS 通告和末端虚拟化的服务访问方式。

为了向用户提供与本地应用无异的虚拟应用使用体验，必须将远程应用界面和本地系统良好融合，提供与本地应用相同的使用机制，例如窗口栏控制、任务栏图标等。我们采用了 Rdesktop＋Seamlessrdp 方式来实现无缝的应用程序界面传送。其中 Rdesktop 是 Linux 下的一个开源 RDP 实现；Seamlessrdp 为 Windows Server 上的无缝连接代理软件。通过修改 Rdesktop 源码可无缝连接服务器应用程序，实现异构平台的应用呈现及融合。

3.4　QoS

与普通的设备资源不同，计算资源有其特有的属性。主要的计算服务属性包括：计算的功能、计算量、服务器性能、当前负载、使用成本、对客户端能力的要求、物理空间位置等，这些特性都是对该服务进行评价的重要因素。

在 UbiCloud 中，服务质量的评价因子主要为服务器主频、CPU 空闲率、总内存大小、可用内存大小、网络速率等。对于 QoS 的目标则是自动选取一个评价最高的服务器推荐给用户。如果用户接收系统推荐的服务器，则可认为评价结果符合用户需求。

我们通过研究基于不同特征的 UbiCloud 服务评价体系和评价方法，结合层次分析算法建立了一个能良好评价服务的 QoS 模型：通过对各评价因子建立层次模型，将 QoS 评价转换为层次决策过程，最后对所评价的对象进行量化比较得出质量序列。

3.4.1　QoS 层次结构模型

对于 UbiCloud 系统，层次结构划分成如下形式：

目标层：选择性能最好的服务器。

准则层：CPU 主频、CPU 空闲率、总内存大小、可用内存大小、网速度。

措施层：服务器 1、2、3…。

依据划分出来的层次结构，建立了数学模型(见图 5)。其中，目标层与准则层的连线表示最终的目标需要考虑哪些因素。在该模型中，选择一个高 QoS 的服务器作为目标，它需要考虑主频、CPU 空闲率、总内存大小、可用内存大小、网速等因素。准则层与措施层的连线表示该可选择目标中包含哪些因素，如果某个供选择目标没有包含某一因素，则该因素在构造判断矩阵时需将该目标剔除。

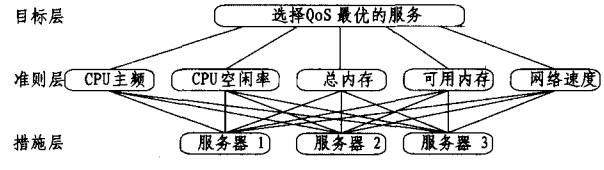

图 5　层次结构模型

3.4.2 层次判断矩阵

层次结构反映了因素之间的关系，准则层中的各准则在目标衡量中所占的权重并不一定相同，在决策者的心目中，它们各占有一定的权重，所以在建立完层次模型后我们需构造判断矩阵。

在 UbiCloud 系统中，各准则所占的权重并不是固定不变的，例如：如果当前网络中服务器的主频都相当的高，完全可以满足用户需求，那么主频在判断矩阵中所占的权重就相对小，而其他准则权重增加。所以，对于不同的环境需要构造不同的总判断矩阵；在同一个环境中，总判断矩阵可基本维持不变。

例如，对于 UbiCloud 系统中的 Office 应用，我们可构造如表 1 所列的准则层判断矩阵。

表 1 影响因子判断矩阵

	主频	CPU 空闲率	内存	可用内存	网速
主频	1	3	3	3	1/2
CPU 空闲率	1/3	1	3	3	1/3
内存	1/3	1/3	1	2	1/4
可用内存	1/3	1/3	1/2	1	1/6
网速	2	3	4	6	1

在表 1 中，我们认为网络速度对云端 Office 应用的响应速度最为重要，所以设置的权重为最大，其他因素的权重按具体要求进行设定。设置的原则为 Saaty[19] 等建议的 1～9 标度法。

3.4.3 构造各准则判断矩阵

在构造出总判断矩阵后，需要动态生成各准则的判断矩阵。在 UbiCloud 中，设计实现了两种构造方法（为简化表述，我们以主频为例，并假设已经得到 3 台服务器 A、B 和 C 的主频信息）：

(1)参考值法

由于总判断矩阵构造采用的是 1～9 标度，因此需要把主频信息也归约化到 0～10 之间的数，但是这种规约化一般并不是简单的线性关系。例如，远大于（远小于）推荐值的主频并没有比用户设置的最大（最小）主频表现出更显著的优势（劣势），因此大于最大值（小于最小值）的主频归约后不需要快速上升（下降）。根据这一特点，我们构造了一个归约化的打分函数，使其大于某一个数及小于某一数时打出的分数变化很小，而处于中间的一段打出的分数变化大，分布曲线见图 6。

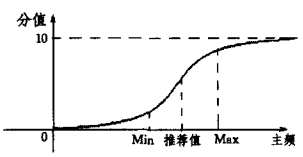

图 6 归约函数曲线图

我们选取了如下分段函数进行模拟：

$$\begin{cases} y=\dfrac{20}{\pi}(\arctan(a_1(f-b_1))+\dfrac{\pi}{2}) & (x\leqslant \text{ReqFreq}) \\ y=\dfrac{20}{\pi}(\arctan(a_2(f-b_2))+\dfrac{\pi}{2}) & (x>\text{ReqFreq}) \end{cases}$$

式中，ReqFreq 为用户推荐值，由 ReqFreq 和 Min 确定分段函数的前半段；再由 ReqFreq 和 Max 确定函数的后半段。

从上面的算法实现过程中可以看出，该算法最大的优点是能准确地计算出相互之间的权重关系，同时不会受突变数据影响，该算法中并不认为远大于理想值的数据是最好的选择，这是符合实际情况的。

(2) 相对值法

参考值法需要先计算分段函数各系数，同时还需要用户的参与。相对值法则能动态地调整相互之间的权重值，在满足用户需求的前提下构造出一致性的判断矩阵。

在一个网络中，所有服务器的主频平均值代表了这个网络的平均处理能力。因此在精度要求不高的场合，我们可以将所得到的服务器主频平均值作为用户的推荐值。为了将主频值归约到 0～10，同时考虑到判断矩阵是两两的相对比较值而非绝对关系，为此可以通过求两者之间的差值来得到其大小关系，同时设置一个步长 *step*，采用如下归约函数计算两服务器主频之间的相互大小：

$$y=(A_f-B_f)/step$$

式中，*step*＝主频平均值/9。

说明：步长 *step* 表示的是针对归约函数的结果，两服务器主频相差多少个 *step* 就说明 A 比 B 重要多少。

该算法的优点是自动化完成判断矩阵的构造，脱离了用户的干预。算法实现简单，不需要用户的参与，其缺点是不能防止突变数据，例如，网络中所有服务器的主频都是 1G，但如果新加入一台 10G 主频的服务器，此时计算出来的步长值 *step* 将不符合实际的网络环境，致使最后的判断矩阵出现偏差。

3.4.4 权向量计算

层次分析法的目的是通过求解各判断矩阵的权向量，得出总的决策向量，权向量为判断矩阵中最大特征值对应的特征向量归一化后得到的单位向量，求解过程如下：

(1)求解特征值和特征向量。

(2)取特征值最大的特征向量进行归一化。

从上面的结果中我们选取特征值最大的一组数据，也即最后一组，采用如下公式进行归一化操作：

$$P(i)=\frac{ABS(Eigenvector(i))}{\sum_{k=0}^{n}ABS(Eigenvector(k))}$$

3.4.5 一致性检验

一致性检验的目的是检验构造的判断矩阵中是否存在诸如 $A>B,B>C,C>A$ 的矛盾关系。对判断矩阵的一致性检验的步骤如下：

(1)计算一致性指标 CI

$$CI=\frac{\lambda_{max}-n}{n-1}$$

(2)查找相应的平均随机一致性指标 RI。对 $n=1,\cdots,9$，Saaty 给出了 RI 的值，如表 2 所列。

表 2 Saaty RI 值

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45

RI 的值是这样得到的：用随机方法构造 500 个样本矩阵：随机地从 1～9 及其倒数中抽取数字构造正互反矩阵，求得最大特征根的平均值 λ'_{max}，并定义

$$RI=\frac{\lambda'_{max}-n}{n-1}$$

(3)计算一致性比例 CR

$$CR=\frac{CI}{RI}$$

当 CR 小于 0.10 时，认为判断矩阵的一致性是可以接受的，否则应对判断矩阵做适当修正。

3.4.6 计算决策向量

决策向量是整个 QoS 求解的最终目的，表示各服务器的总评分情况。参照该决策向量我们即可选择出总评最高的服务器，从而推荐给用户使用。决策向量为准则层各因素权向量相对总权向量的算术平均数。下面以一个例子来说明决策向量的求解。

假设网络中有 A,B,C 3 台服务器，准则层各因素的权向量为：

主频(a1,b1,c1)；

CPU 空闲率(a2,b2,c2)；

内存(a3,b3,c3)；

可用内存大小(a4,b4,c4)；

网速(a5,b5,c5)；

总权向量(k1,k2,k3,k4,k5)。

计算决策向量为：

$(\sum a_i * k_i, \sum b_i * k_i, \sum c_i * k_i)$

最后选择决策向量中值最大的一项推荐给用户，完成 QoS 评价。

4 实验及结果分析

4.1 实验平台

我们建立了一个 UbiCloud 测试环境，对系统的功能、稳定性及性能进行了测试。实验环境由 27 台终端、6 台服务器及 1 台注册中心组成(见表 3)。每台服务器上发布多个不同的应用程序，如 Office、浏览器、OA、图像处理软件、程序开发软件等，并根据软件的不同，建立不同的影响因子判断矩阵以进行 QoS 计算。

表 3 实验平台

	CPU	主频	操作系统	数量
便携终端	龙芯 SoC	200 MHz	Embedded Linux	10 台
瘦客户机	龙芯 2F	600 MHz	Debian 2.6.27.1	15 台
台式机	Intel X86	2 GHz	Windows XP SP2	2 台
服务器	Intel X86	3 GHz	Windows Server 2003	6 台
注册中心	Intel X86	1.7G*2	Windows XP SP2	1 台

4.2 开发工具

UbiCloud 系统采用 C 和 C++语言开发开发，并利用了一个开源的 UPnP 工具包(libupnp)来支持基于 IP 的异构系统通信。

4.3 系统响应性能分析

定义 1(应用响应时间，Application Response Time，ART)应用响应时间是指客户端发出请求到客户端呈现出远程应用界面所需的时间。

$$ART=T_{interface}-T_{request}$$

我们定义了应用相应时间(Application Response Time，ART)作为系统性能评测的指标(见定义 1)。当通过双击打开系统中某一文件时，如果应用界面响应时间过长，势必会降低用户对该系统的印象。对于 UbiCloud，应用的响应时间是性能评价的主要指标。我们分别针对不同的服务和平台测试了系统的 ART。

首先测试 UbiCloud 在不同类型终端上的应用响应情况。测试方法为在 3 种终端平台上使用 UbiCloud 客户端分别测试打开一个 Word 文档、TXT 文档及 IE 浏览器的时间(见图 7)。可以看出，应用响应时间在不同终端上差异不大，这是因为不管是在什么平台上，该系统的响应时间大部分都取决于网络的流畅情况和服务器的负载大小，所以在网络相同的情况下，平台的差异不会造成很大的变化。

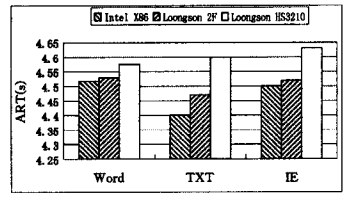

图 7 在不同类型终端上的 ART

然后，测试不同类型的终端在 Ubicloud 计算模式下和本地计算模式下的应用响应差异。测试方法为在 3 种终端平台上分别使用 UbiCloud Word 和本地 Openoffice 打开一个 Word 文档，统计其 ART(见图 8)。可以看出在网络通畅的情况下，使用 UbiCloud Word 可以明显地降低系统的响应时间。

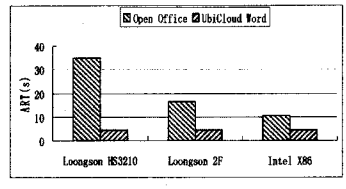

图 8 Ubicloud 模式下和本地模式下的 ART

结合上述两图可以发现，在低性能的弱客户端上，UbiCloud 方式将更显优势，实现了服务器(云)和普适终端的性能权衡，降低了终端自身对计算能力的要求。

4.4 QoS 评价器性能分析

在 UbiCloud 中，QoS 通过服务器端主动广播 QoS 状态信息给注册中心来实时获取。当注册中心接收到一个用户服务请求时，先扫描全局设备列表，将 MIME 类型匹配成功的服务加入到 QoS 评价列表中，然后通过前文介绍的 QoS 评价算法计算得出各个服务的 QoS 分值，最后将评价的结果结合服务信息返回给用户。

系统会自动标记出 QoS 分值最高的服务器推荐给用户使用。如果用户选择系统推荐的服务器，则认为评价结果正确；反之，如用户另外选择别的服务器，则认为评价结果错误。最后对用户的选择结果进行统计。

根据上述实验方法，对系统中 2000 次服务访问的评价质量进行统计，结果见图 9。

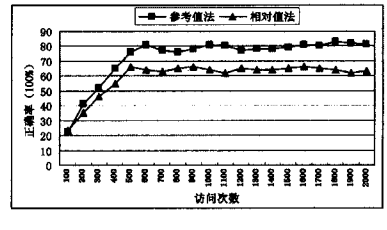

图 9 QoS 评价实验结果

实验结果显示，系统运行的最初阶段，用户自主选择服务器的情况较为普遍，但随着使用次数增多，用户逐渐接收系统推荐的服务器，由此可见系统的 QoS 评价模型和评价算法是可用的。在两种准则判断矩阵的构造方法比较中，参考值法比相对值法准确率高，这也符合我们事先的预测。

性能方面，QoS 的时间复杂度随着影响因子的增加呈指数级上升，这是由于其计算过程中需要对判断矩阵进行特征值求解及多次比较。从图 10 中可以看出，QoS 计算的时间复杂度与服务器数量近似呈指数级关系，如果同时有大量服务器进入评价列表将会对用户体验造成严重的影响。为此，在评价服务器之前，需要对所评价的服务器做一个初步的筛选，如将网速低于某个限值的服务器排除等。

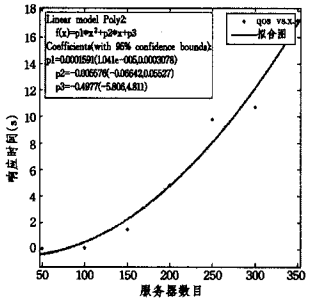

图 10　QoS 计算时间与服务器数量关系

5　相关工作

Giurgiu[2] 提出了一种可以在移动客户端和服务器间自动分发应用组件的中间件。Satyanarayanan[3] 提出了一种新的系统架构，移动用户可以利用虚拟机技术来访问附近的 cloudlet 服务。Byunggon[4] 提出了一种新架构，实现了智能手机操作系统影像在远程云中的克隆与卸载。Huifeng[9] 介绍了一种高性能的远程计算平台，它使瘦客户端可以远程访问服务器的应用程序。朱珍民[16] 实现了一个异构应用资源共享的原型系统。该原型系统选择了虚拟桌面作为用户视图的呈现方式，通过虚拟桌面对应用接口进行无缝集成。李忠[17] 介绍了基于应用推送的虚拟化技术，并详细叙述了远程应用与本地桌面的无缝融合方法。

据我们所知，UbiCloud 是第一个构建在 UPNP 协议之上并利用 SoA 技术进行计算资源发布的异构云计算系统。该系统可以为普适终端提供无缝的云资源调用接口，从而降低终端自身对资源的需求。

结束语　在普适计算环境，服务发现和访问技术在设备互操作、云计算等领域具有非常好的应用前景。我们基于这些技术，设计了 UbiCloud，目的是通过对计算服务资源的描述、发现、访问、评价等问题进行研究，在前端普适终端与后端服务器（云）之间构建虚拟计算环境，使普适终端可以随时随地获取强大可靠的计算资源。

此外，本文通过研究基于不同特征的普适计算服务评价体系和评价方法，结合层次分析算法建立了一个能良好评价服务的 QoS 模型，并基于该模型提出了两种构造一致性 QoS 判断矩阵的方法：相对值法和参考值法。实验结果表明，这两种方法能够较为准确地评价出计算服务的质量。

参 考 文 献

[1] Hao F, et al. Enhancing dynamic cloud-based services using network virtualization[J]. SIGCOMM Comput. Commun. Rev., 2010, 40(1): 67-74

[2] Giurgiu I, et al. Calling the Cloud: Enabling Mobile Phones as Interfaces to Cloud Applications[C]// Middleware 2009. 2009: 83-102

[3] Satyanarayanan M, et al. The Case for VM-Based Cloudlets in Mobile Computing[J]. IEEE Pervasive Computing, 2009, 8: 14-23

[4] Byunggon C, Petros M. Augmented SmARTphone Applications Through Clone Cloud Execution[C]// HotOS 2009. 2009

[5] Rudolph L. A Virtualization Infrastructure that Supports Pervasive Computing[J]. IEEE Pervasive Computing, 2009, 8(4): 8-13

[6] Hayes J. Thin client's fat challenge [IT Desktop Computing] [J]. Engineering & Technology, 2010, 4(21): 52-53

[7] Beaty K, Kochut A, Shaikh H. Desktop to cloud transformation planning[C]// Parallel & Distributed Processing, 2009. IPDPS 2009, IEEE International Symposium on. 2009

[8] Al-Turkistany M, Helal A S, Schmalz M. Adaptive wireless thin-client model for mobile computing[J]. Wirel. Commun. Mob. Comput., 2009, 9(1): 47-59

[9] Huifeng S, Yan L, Feng W, et al. A high-performanance remote computing platform[C]// Pervasive Computing and Communications, 2009, PerCom 2009. IEEE International Conference on, 2009. 2009: 1-6

[10] Doyle P, et al. Ubiquitous desktops with multi-factor authentication[C]// Digital Information Management, 2008. ICDIM 2008, Third International Conference on. 2008

[11] Eick S G, et al. Thin Client Visualization[C]// Visual Analytics Science and Technology, 2007. VAST 2007, IEEE Symposium on. 2007

[12] Lubonski M, Gay V, Simmonds A. A Conceptual Architecture for Adaptation in Remote Desktop Systems Driven by the User Perception of Multimedia[C]// Communications, 2005 Asia-Pacific Conference on. 2005

[13] Starner T. Thick clients for personal wireless devices[J]. Computer, 2002, 35(1): 133-135

[14] 张建勋，古志民，郑超. 云计算研究进展综述[J]. 计算机应用研究，2010，27(2)：429-433

[15] 陈援非，朱珍民，叶剑. 一种基于多量级虚拟机的可扩展普适计算架构[C]// 第四届和谐人机环境联合学术会议. 武汉，2008

[16] 朱珍民，蒋发群，苏晓丽，等. 面向普适计算的应用共享模型[J]. 软件学报，2007，18：54-62

[17] 李忠. 基于应用推送的桌面虚拟化架构关键技术[D]. 北京：中国科学院计算技术研究所，2008

[18] 李伯虎，柴旭东，侯宝存，等. 一种基于云计算理念的网络化建模与仿真平台——"云仿真平台"[J]. 系统仿真学报，2009(17)：5292-5299

[19] Saaty T L. A scaling method for priorities in hierarchical structure[J]. Journal of Mathematical Psychology, 1977, 15(3): 234-281

[20] UPnP™ Technology-The Simple, Seamless Home Network [OL]. http://www.upnp.org/resources/whitepapers/UPnP