

# 适用于富客户端的云计算模型

孙放, 陈云芳, 林杭锋

(南京邮电大学 计算机学院, 江苏 南京 210003)

**摘要:**在云计算技术兴起的今天,还并没有一套能够和富客户端很好结合的云计算模型。对于此种情况,文中的目的在于提出了一种高效的适用于富客户端的云计算模型,可以将用户的数据处理需求融入云计算环境中。在此模型的基础上,通过实现一套能够实现大图像信息的高效分布处理的在线图像处理系统的方法,对本模型进行实验验证。由此对系统可行性、实用性和性能等进行了相关测试,根据测试结果研究了所提出云模型的优劣之处并提出了可能的改进方案。

**关键词:**云计算;富客户端;分布式

**中图分类号:**TP393

**文献标识码:**A

**文章编号:**1673-629X(2010)08-0096-04

## Cloud Computing Model Applicable to Rich Client Applications

SUN Fang, CHEN Yun-fang, LIN Hang-feng

(Computer College, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

**Abstract:** Today, the cloud-computing technology rapidly rises, but it is still a blank area of the combines between the cloud computing and rich client application. Aiming at this, an efficient cloud computing model that can be applied to rich-client application is proposed. The model can be integrated the user's data processing needs into cloud computing environment. Then complete a experimental verification by achieving a set of distributed-line image processing system on this basis and realizing an efficient distributed processing of huge image information. Finally, some experiments on it has been done and bring up the improvements of the model by the result of the experiments.

**Key words:** cloud-computing; rich-client; distributed

## 0 引言

云计算是并行计算(Parallel Computing)、分布式计算(Distributed Computing)和网格计算(Grid Computing)的发展,或者说是这些计算机科学概念的商业实现。目前业界对云计算尚无一个统一的、严格的定义。一般认为,广义上的云计算便是用户友好的分布式计算。它有几种实现形式,如软件即服务(SaaS)、平台即服务(PaaS)等<sup>[1]</sup>。

云计算的主要特性有超大规模、虚拟化、高可靠性和按需服务等,核心支撑技术主要为网格计算、虚拟化技术、容错技术和灾难恢复技术和相关并行处理的模型和算法,如 MapReduce 等<sup>[2]</sup>。

实现云计算的方式一般是建立一个云架构,把普通的本地应用放到该架构上运行。目前现有的云架构有以下三种<sup>[3,4]</sup>:①提供云平台,供开发者开发应用程

序部署与其上来实现云计算的,属于平台即服务模式,如 Google 的 App Engine。②提供基础硬件平台,通过虚拟化技术在其上运行操作系统等的分布式架构,属于基础设施即服务,如亚马逊的 AWS EC2 弹性计算云。③直接提供在线应用或者 RIA 程序让用户使用,并具有分布式处理功能的,属于软件即服务。这一种程序一般是由开发者使用成熟的分布式框架(如 Hadoop)或自行开发分布式架构来设计的<sup>[5]</sup>。无论上述哪一种实现形式,都并没有和目前另一种新的互联网应用模式——富互联网应用程序(RIA: Rich Internet Application)很好地相结合。

RIA 是一种基于 Web 的类 C/S 架构,综合了 C/S 架构和 B/S 架构的优点。RIA 的核心是富客户端,提供了丰富的用户界面和与后台应用服务器的异步通信方式,解决了 B/S 架构的弊端,提高了 Web 应用的用户体验。可以认为 RIA 使程序直接在浏览器中运行<sup>[6]</sup>。而云计算,恰是将计算提交至服务器端进行,并可在云端存储数据。如果两者相结合,那么可以最大限度地解除程序对客户端硬件设备的依赖,实现瘦客户端。

收稿日期:2009-12-14;修回日期:2010-03-13

基金项目:国家博士后项目(20090451241)

作者简介:孙放(1989-),男,研究方向为云计算、富客户端应用等;陈云芳,博士,讲师,研究方向为入侵检测、人工免疫、云计算等。

文中提出了一种适用于富客户端的云计算模型,并以此开发了一款基于 Adobe Flex RIA 技术的分布式图像处理系统,测试该模型的性能并提出了相关改进方案。

## 1 云计算模型

在文中所提出的云计算模型中,存在三种主要的角色:客户端、主服务器、节点服务器。在基于 RIA 模式的 Web 程序中,由于在浏览器或其插件中执行的程序代码功能强大,故可视为客户端。客户端主要负责和用户进行交互、对待处理数据进行初步的处理和编码、和主服务器端进行通讯等。

在本模型中,主服务器又可分为两个模块,指令服务器和数据服务器,两者可以在同一台物理服务器上,也可在不同的物理服务器上。指令服务器主要负责管理客户端、节点服务器的连接和注销;接收节点发送的任务处理请求;记录节点服务器的负载状态,以此进行任务的分派;负责监视各节点的处理数据的状况,判断任务是否全部完成,若完成则通知客户端收取处理后的数据。

数据服务器则负责要处理的数据的存放及映射与分割的工作。数据服务器接收从客户端发来的待处理的数据,之后进行数据的分割与映射。一般来讲,可以由指令服务器在衡量各节点负担后,向负担相对较少的节点发送处理请求,再由节点根据本身的编号(一般由节点连接到主服务器时由主服务器分配)请求数据服务器进行数据的分割与映射。当节点处理数据完成后,再向数据服务器传回处理后的数据,数据服务器将其拼合后传输给客户端。

云计算节点将负责数据的实际的处理。一般来讲,节点服务器程序可采用和客户端相同的编程技术,以方便数据处理代码的重用和数据传输时的兼容性。节点服务器必须亦能和主服务器的两个部分进行通讯,但不必和客户端直接通讯。

本体系结构的示意图如图 1 所示。

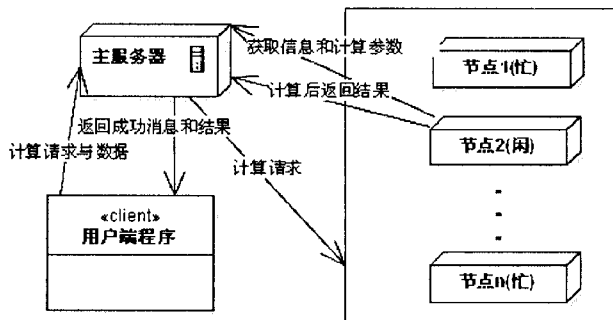


图 1 云模型的体系结构

综上所述,本套架构中,客户端和节点都只与主服务器通信,两者之间不会直接通信。这么做的主要原因有三点:一是富客户端程序由于是直接承载于浏览器中,故会受到安全沙箱等的限制,若要访问额外的域的话会非常困难。二是富客户端程序的一个重要特性是跨平台性,这就意味着富客户端程序极可能会在手机一类的移动平台上运行。最后,尽量避免过多的服务器暴露给客户端,减少安全隐患。因此,在这种环境下通过主服务器与节点通信自然比直接和各节点通讯要好。

## 2 关键实现技术

### 2.1 基于消息传递的协同

上文中,指令服务器部分主要管理的是客户端和各节点的协同。这就意味着指令服务器程序必须具有在广域网上的消息传递能力,并且其采用的通信协议具有能保持连接的特性。在传统的分布式编程中,MPI(Message Passing Interface,一种并行编程的框架)框架便是负责处理此类工作的<sup>[7]</sup>。但是在 RIA 程序设计中,必须另行选择。一种方案是直接基于 socket 编写一套简单的消息传递的协议,并对参与其中的各部分程序编写处理消息的函数。在开发 RIA 的常用技术都对 socket 提供了良好的支持。二是采用一些为 RIA 程序开发的应用程序交互或远程调用的框架和技术,例如,Adobe 为 Flex RIA 开发提供的一种用于编写用户间交互程序的服务器软件 Flash Media Server,采用的 RTMP(Real Time Messaging Protocol,实时消息传送协议)协议满足上述全部要求,并且较易使用,是开发类似系统较好的选择。

一般来讲,在指令服务器向客户端、节点服务器发送消息时,宜采用广播的方式,向所有在线的客户端和节点发送消息,在消息的第一个参数或消息头上加入目标的编号,由客户端或节点程序判断是否是针对自己的消息。这样可以在复杂的网络环境中最有效率地传递消息。

在消息中一般要附上若干参数,如请求任务的客户端编号、要处理数据的方式等。一般来讲,客户端编号这样必须的参数可作为固定的一部分加入到消息中,而其他所有参数以类似于数组的形式传递,如果是基于 socket 自行定义消息传递的协议,则还需以固定格式在消息中存入参数数组的长度,以便程序处理数据。而上文中的 RTMP 协议本身就对此有良好的支持,则不需要。

### 2.2 任务数据的分块

一般来讲对任务数据的分块首先要能够将数据序

列化,如将图像转化为一维数组的类似的结构,并在序列化后的数据的特定位置加入能使数据还原的特征值,如图像的长宽等,以上处理通常在客户端进行。最后要将处理完成后的数据变为字节流,发送至主服务器数据服务器部分。

数据在数据服务器端的存储一般是在内存中的。当数据需要分割时,一般由数据服务器读取序列化后的数据中的特征值<sup>[8,9]</sup>,进行换算之后,把相应部分的数据和特征值分配给相应节点。

上文提到,数据服务器不宜在指令服务器向节点分配任务后,就直接开始分割数据,而是应由节点在收到开始处理数据的指令后主动请求数据服务器进行数据分割。这主要是为了避免指令服务器的处理指令没有发送到相应节点或相应节点故障而没有响应造成的处理错误的发生。亦可采用由节点直接向数据服务器发送下载数据的请求时,附带发送分割数据的参数,数据服务器这时直接将节点请求的部分发送给节点。

### 2.3 任务的分配

记录各节点的当前负担是实现任务分配的重要保障。一般来讲,在指令服务器中设置一个数组,分别存储对应节点的当前负担。节点的当前负担主要靠节点当前处理的数据量来衡量,因为对相等单位的数据做相同的处理的运算量在理论上是相同的。当指令服务器分发给节点任务后,即可将该节点处理的数据量记入负担值中,若收到节点返回处理完成的消息,则在负担值中减去该量,以实现节点负担的动态记录<sup>[10]</sup>。

用以上的方式记录节点负担后,只需在指令服务器程序上对各节点的负担值排序即可找出负担最小的节点,一般来讲,比负担值最小的节点的负担值大一定单位(记为  $det$ )的节点可视为第二个较闲的节点,再由第二个较闲的节点负担值加上  $det$ ,来查找第三个较闲的节点,以此类推,直至无法找到比最后一个节点负担值 +  $det$  小的节点为止。找到的都认为是较闲的节点,以此来确定一个任务由哪些节点处理。

综合以上三点,本系统的主要工作流程如图 2 所示。

### 2.4 节点的检测和故障恢复控制

在实际的应用中,可能会存在节点服务器突然故障的情况。这时为了保证整套系统的正常工作,就必须暂停向故障节点分配任务。在本体系中,节点状态的检测主要由主服务器指令服务器部分负责。若是基于 socket 自行编写指令服务器,则一般要采用主服务

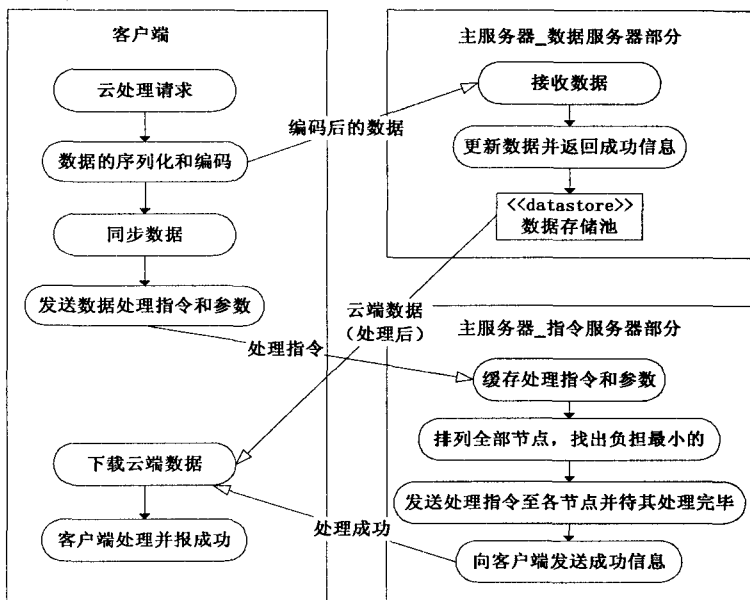


图 2 请求处理流程

器轮询节点的方式来判断节点是否故障。如果是使用了类似 FCS 一类的系统做指令服务器,则它们一般都自带节点状态检测功能,不需另行处理。总之,主服务器一旦检测到了节点的异常,就应立即将节点踢出节点列表,并且待节点服务器正常后若发现与主服务器断开,应该有能力自行重新连接主服务器。

### 2.5 主服务器和节点服务器状态的监控

除了在节点程序内部对程序的运行进行检测外,对各个服务器本身状态的自动监控也是必要的。这种监控是可以单独开发和部署的,无需整合至云系统内部。一般来讲,在 Linux 操作系统下,可以通过读取控制台命令返回的信息来监控系统状态<sup>[11]</sup>;而在 Windows 系统下,一般要安装一些获取系统状态的小程序,如 PHPSysInfo 等。当然也可安装专门的集群管理软件。

## 3 性能测试及分析

### 3.1 测试环境

笔者采用所提出的云计算模型,实现了一套使用 Adobe Flex Builder 3 及 Java Servlet 技术的分布式在线图像处理系统,测试硬件环境如下:

主服务器端配置: Intel Xeon 2.8GHz, 512MB RAM, Win Server 2003, JDK 1.6.0\_18, Tomcat 5.5, FMS 3.5;

节点服务器配置: Intel Pentium4 2.93 GHz, 1024MB RAM, WinXP, Flash Player 10.0.082, 浏览器 IE 7.0;

客户端配置: Intel Pentium M 1.86GHz, 2.0GB

RAM, WinXP, Flash Player10.0.082, 浏览器 IE 8.0;  
网络连接情况:同交换机、同网段 100Mbps 网络。  
该分布式在线图像处理系统具有多种图像处理功能,选取了如下功能作为具体测试操作:黑白渐变、模糊滤镜、浮雕滤镜、锐化滤镜、查找边缘滤镜,其实现如表 1 所示<sup>[12]</sup>。

表 1 图像处理操作和实现方法

操作	实现方法
渐变	遍历所有像素,将位于坐标(x,y)的像素的颜色设为其渐变的起始颜色和终止颜色之差的 d 倍,d 为 (x,y)与渐变起始点的距离与渐变长度的商
模糊	采用对所有像素进行卷积运算的方式,卷积矩阵的所有元素均为 1,结果除以 9 以防过饱和
浮雕	采用对所有像素进行卷积运算的方式,卷积矩阵的各个元素为[-2,-1,0,-1,1,1,0,1,2]
查找边缘	采用对所有像素进行卷积运算的方式,卷积矩阵的各个元素在水平边缘和垂直边缘的 mat 值分别为 [1,2,1,0,0 0 -1,-2,-1]和[1,0,-1,2,0,-2,1,0,-1],结果除以 9 以防过饱和

另外,由于实现的测试系统是基于 Adobe Flex Builder 3 实现的,其代码是在 AVM2(Action Script Virtual Machine)中执行的,所以其本身执行代码的性能也必须予以考虑。由 Adobe 官方公布的数据可得:

在执行 count = 0; for i = 0 to 1234 { for j = 0 to 1234 {count += i \* j;}} 类似代码 50 次的平均计算时间为 87ms。

3.2 测试结果及分析

从测试结果可以看出本系统在图像处理数据量不大时,其使用云计算模式的效率并不如使用本地模式的效率高,而只有图像处理的数据量增大到一定程度时,其云模式的优势才会显现出来。这主要是由于待处理数据在网络传输中的延迟。并且本次测试是在一个较理想的局域网环境中进行的,若在环境较差的广域网下进行,云模式的优势会更不明显。并且可以看出,在待处理的数据量较小时,增加节点数对提升处理速度的作用并不明显,甚至可能起反作用,这是由于主服务器和增加的节点间的额外通信所占用的时间造成的。测试结果数据比较如表 2 所示。

另外,经测试,在处理客户端请求时,对主服务器的负担是相当小的,但对节点服务器的性能却有一定要求。尤其是在有较密集的请求待处理的情况下,可能会对节点造成一定的负担。

在测试中出现了某一节点服务器突发错误的情况,该节点服务器能够及时弹出错误窗口,并且断开和主服务器的连接,并没有影响到随后的任务的处理。

表 2 测试结果数据比较

操作时间 (ms)	处理模式	图像分辨率(px)			
		100 *	500 *	1000 *	2000 *
		100	500	1000	2000
黑白渐变	本地模式	16	29	328	704
	云模式(5 节点)	125	172	343	547
	云模式(15 节点)	131	194	171	392
模糊	本地模式	2	14	157	330
	云模式(5 节点)	74	188	235	394
	云模式(15 节点)	106	211	187	265
浮雕	本地模式	16	84	407	904
	云模式(5 节点)	93	156	266	392
	云模式(15 节点)	101	172	203	272
查找边缘	本地模式	15	106	359	574
	云模式(5 节点)	78	187	256	325
	云模式(15 节点)	89	125	125	293

4 结束语

文中提出了一种适用于富客户端的云计算模型,并且用其实现了一套分布式在线图像处理系统并进行了性能测试,总体来讲,本模型是具有相当的优势和实用性的,不过也有需要改进的地方。

(1)对网络情况的依赖性较大。测试数据明显反应出由于网络传输的延迟,使小计算量的云模式的处理时间明显高于在客户端本地的处理时间。因此在图像数据的传输中应采用高效的压缩算法,以减少对网络的依赖;

(2)在本模型分配任务时,判断节点忙闲程度的算法较为死板,在大规模的应用中可能会对系统性能的发 挥造成影响,故应该研究和改进出更好的动态分配任务的算法;

(3)在本系统的实现中,发现对主服务器的监控较易做到,但是由于节点部署成本和力度较低,故对节点服务器的监控能力较差,所以还需再次开发一套针对节点服务器的监控系统;

(4)在本系统的工作流程中,数据是以非加密的形式传输的。在一些对保密性要求较高的应用中,应采用相关协议的加密版本,如 RTMPS(RTMP 协议的加密版本)及 HTTPS 协议等。

参考文献:

[1] 曾 诚,李 兵,何克清. 云计算的栈模型研究[J]. 微电子学与计算机, 2009,26(8):22-25.  
[2] 陈 全,郑倩妮. 云计算及其关键技术[J]. 计算机应用, 2009,29(9):2562-2567.  
[3] 王 鹏. 走进云计算[M]. 北京:人民邮电出版社,2008.  
[4] 江晓庆,杨 磊,何斌斌. 未来新型计算模式—云计算[J].

$$\begin{cases} C_r = \text{rand}() \% k_r / k_r \\ C_g = \text{rand}() \% k_g / k_g \\ C_b = \text{rand}() \% k_b / k_b \end{cases} \quad (10)$$

式  $k_r, k_g, k_b$  中控制烟花的颜色。

### 3.5 烟花粒子的消亡

烟花粒子在爆炸后经过一定的时间间隔后,就会从系统中消亡。文中粒子消亡的条件是粒子的生命周期以 fade 速率递减到零时,粒子完全透明,消失不见。

### 3.6 烟花绘制

图3给出了实验中仿真出的五彩缤纷的烟花,具有很好的真实感。

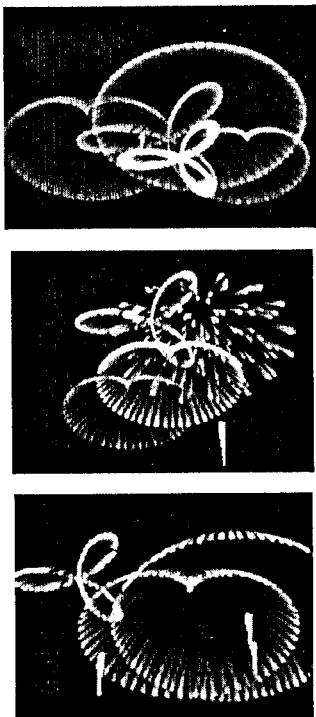


图3 仿真的五彩烟花

## 4 结束语

文中在分析粒子系统基本理论的基础上,采用粒

子系统对烟花进行了仿真。同时结合 OpenGL 图形库,仿真出了效果逼真的五彩缤纷的烟花,较传统的粒子系统仿真出的烟花视觉效果更好,而且速度快,简单。

### 参考文献:

- [1] Reeves W T. Particle systems - A technique for modeling a class of fuzzy objects [J]. ACM Transaction on Graphics, 1983, 2(2): 91 - 108.
- [2] 丁强,陈青林,左福强. 基于 LOD 的火焰粒子生成技术 [J]. 计算机应用, 2005, 25(12): 257 - 259.
- [3] 庞新,王相海. 基于 OpenGL 的礼花粒子系统模拟研究 [J]. 计算机科学, 2008, 35(5): 216 - 219.
- [4] Reeves W T, Blau R. Approximate and probabilistic algorithms for shading and rendering structured particle system [J]. Computer Graphics, 1985, 19(3): 313 - 322.
- [5] Loke T, Tan D, Seah H. Rendering Fireworks Displays [J]. IEEE Computer Graphics and Applications, 1992, 12(3): 33 - 43.
- [6] 丁纪云,陈利平,李思昆. 基于 OpenGL 的烟花动态模拟方法的研究与实现 [J]. 计算机工程, 2002, 28(4): 240 - 241.
- [7] 陈利平,王国才. 基于粒子系统的蜡烛焰火实时模拟 [J]. 计算机技术与发展, 2006, 16(5): 186 - 188.
- [8] 甘露,王文永,孙博. 基于粒子系统的烟花建模方法研究 [J]. 长春师范学院学报: 自然科学版, 2008, 27(1): 63 - 66.
- [9] 罗玉玲. 粒子系统与纹理映射相结合模拟礼花的研究 [J]. 电脑知识与技术, 2004, 20: 70 - 72.
- [10] 李清畅,杨高波,王小静. 基于粒子系统的焰火建模及其算法仿真 [J]. 系统仿真学报, 2009, 21(8): 2179 - 2184.
- [11] 尹星云,胡长俊. 基于 OpenGL 的烟花粒子系统设计 [J]. 电脑开发与应用, 2006, 19(7): 28 - 30.
- [12] 王红霞,王文永,钟绍春,等. 基于粒子系统的烟花仿真算法的改进 [C] // 第十四届全国图象图形学学术会议论文集. 北京: 清华大学出版社, 2008: 644 - 648.

(上接第99页)

计算机与数字工程, 2009, 37(10): 46 - 50.

- [5] 陈康,郑纬民. 云计算: 系统实例与研究现状 [J]. 软件学报, 2009, 20(5): 1337 - 1348.
- [6] 石永革,许建林,石峰. 富客户端技术应用研究与实现 [J]. 计算机工程与设计, 2008, 29(3): 639 - 641.
- [7] 郭本俊,王鹏,陈高云,等. 基于 MPI 的云计算模型 [J]. 计算机工程, 2008, 35(24): 84 - 86.
- [8] Dean J. MapReduce: Simplified Data Processing on Large Clusters [C] // Proc. of the 6th IEEE Symposium on Operating System Design and Implementation. San Francisco, CA,

USA: [s. n.], 2004.

- [9] Lamell R. Google's mapreduce programming model - revisited [M]. Redmon, USA: Data Programmability Team Microsoft Corp, 2007.
- [10] 秦金磊,朱有产,李玉凯. 基于网格计算的关键技术研究 [J]. 计算机技术与发展, 2006, 16(11): 103 - 105.
- [11] 万至臻,陈刚,寿黎但. 基于 MapReduce 的并行计算平台的设计与实现 [D]. 杭州: 浙江大学, 2008.
- [12] 李樱,王永滨,王珂,等. 广域网环境下分布式动漫渲染研究 [J]. 微电子学与计算机, 2009, 26(8): 25 - 27.