# 实验报告

——M201773256 郑展

# 1 介绍

这篇论文在开篇提出了一个重要的问题，就是如今大量的云计算的收费模式是租户根据自己需求购买相应的带宽服务，付相应的价钱。但是VM实例没有定量的网络性能，因此具有相同的CPU和内存或类似定价的虚拟机，带宽性能的差异可能高达16倍。对于同一云中的虚拟机，较便宜的VM的平均带宽可能超过昂贵的带宽。对于单个虚拟机，网络性能在各个时间的变化也不能预测。这种拥有不确定性的定价模式，不能保证租户得到服务的性价比。同时，由于现代大量的云服务器带宽资源利用率并不高，很多虚拟机会有闲置状态，但为该虚拟机提供的带宽服务不会变，导致服务器很多带宽没有达到充分利用，即使有些在传输的虚拟机想要更多的带宽来为它服务，也并没有一种定价机制来提供保障。

为了以性能保证对带宽进行定价，解决方案中不仅要实时的了解虚拟机所需带宽，还要考虑将服务器的剩余带宽合理利用，鉴于目前的云规模数据中心网络利用率比较低，作者想到可以采用一种过度承诺带宽保证，并为租户提供基于用户使用的定价。因此，本文的作者想要通过制定一种新的定价模式SoftBW来解决这个问题。该模式的主要特点是在每个VM上进行实时的速率监控，根据用户的保证带宽，履行带宽以及定价模型进行定价的计算，可以保证用户的性价比。同时，空闲的虚拟机资源并不会保留，服务器会把空闲的资源给一些需要快速完成的虚拟机，同时对该虚拟机在原有的带宽保证的基础上，收上基于带宽的少量的服务费加成。这样一来，用户的服务也得到了保障，并且一些急需完成传输的用户也可以利用闲置带宽，付上少量的附加服务费，达到更快的传输需求，用户体验增加。而服务器也不会出现大量带宽闲置的问题，带宽的利用率会提高，通过闲置带宽收取少量服务费，盈利也会提高。

# 2 SoftBW介绍

本论文作者所提出的SoftBW定价模型就是一种基于用户使用的定价模型。该模型拥有三种保证机制，严格保证，动态保证和公平性保证。其中严格保证和动态保证是用户一开始由自己选择的带宽保证方式，而公平性保证则是由服务器通过合理利用闲置带宽，以达到充分分配服务器带宽的目的。

* 严格保证为虚拟机提供实时最小带宽保证，用户在使用期间，服务器会为其实时保证这个最小带宽，如果虚拟机空闲，服务器才会将他的闲置带宽暂时分配出去。在虚拟机再次需要传输时为其提供不低于严格保证的带宽。
* 动态保证确保在一定的时间段内，VM总可以交付相应大小的流量，动态保证和严格保证最大的区别就在于，严格保证是实时的保证有不低于保证带宽的速率，而动态保证只需要在指定的时间内完成相应的传输任务即可，它的带宽可以一开始很高后来很低，或者一开始很低某一刻很高。反正它的要求只是保证这一段时间内的平均带宽。服务器可以根据使用情况进行自动分配他的实时带宽。

1. 若在指定时间段内，提前完成了传输任务，服务器会回收所有带宽，并等到指定时间过后的下一次传输任务，再考虑这个虚拟机的带宽分配。
2. 若在指定时间段快要结束的时候，传输任务没有完成，服务器会从目前闲置的带宽中分配大量带宽给该虚拟机，以保证改时间段的平均带宽。

* 公平保证提供了公平的VM级公平性，用于共享所有虚拟机之间具有的严格和动态保证的虚拟机所留下来的剩余带宽。对分配完的带宽，收取基于附加带宽的少量附加收费。

SoftBW模型会在租户的虚拟机上安装一个速率检测软件，用于实时的检测虚拟机的使用带宽，比如1秒，2秒或者其他。方便基于用户使用情况进行定价，并且可以让服务器知道是否存在带宽资源闲置，以达到充分利用带宽资源的目的。当用户选择使用严格保证时，用户所要承担的服务费用需要考虑到用户的实时速率，用户所签订的严格保证带宽与总物理带宽的比值以及单位带宽的费用等 ; 若用户选择的是动态保证，用户所要承担的服务费用需要考虑到用户的实时速率，用户所规定的传输时间段，传输任务，服务器总带宽以及单位带宽费用等。但由于动态保证更有利于提高服务器对总体资源的利用率，因为严格保证可能存在其实实时速率很低但是还是需要给严格保证用户严格保证的带宽的情况，导致带宽闲置，降低服务器带宽的利用率。而动态保证只要完成传输任务，闲置的时候，带宽可以任由服务器分配，于是服务器提供商应该鼓励动态保证的策略，因此需要在最终的付费结果上乘以一个小于1的数来降低动态保证用户的收费；当服务器将一些闲置的带宽分配给一些虚拟机使用时，这时候需要收取一定的基于带宽的服务费，这个附加服务费所要考虑的主要是用户实时的使用带宽，服务器的总带宽以及单位宽带费用等。同样的，公平分配原则也有利于提高服务器带宽的使用率，因此也是服务器提供商所支持的，所以也会在最终结果后，给定一个小于1的值来降低费用。

# 3 实验设计

本篇论文的主要核心在于这个定价策略SoftBW,所以这篇论文的实验报告我选择实现该论文中的定价模式，并且模拟定价模型的执行过程。执行过程中，对于SoftBW中的实时速率检测软件（通过每秒的测试数据返回用户的实时速率），本次实验是采用迭代的方式来模拟每秒的实时速率。设计两种类型的类，一种是StrictVM，即是对应于严格保证的用户，主要的属性有与该用户签订的严格保证带宽和该用户所需的服务价格，另一种是DynamicVM，即是对应于动态保证的用户。主要的属性有数据大小，数据传输的时间段和该用户所需的服务价格。在测得用户实时带宽的时候，用户的实时带宽会有浮动，而浮动的方式是采用随机函数的方式来表示浮动。

# 4 实验实现

实验的实现选择采用java语言环境。

StrictVM.java(严格保证的用户)

|  |
| --- |
| **public** **class** StrictVM {  **private** **double** cmtBandwidth;//给虚拟机的承诺带宽 commitment Bandwidth  **private** **double** price;    **public** StrictVM(){  **this**.cmtBandwidth = 0;  **this**.price = 0;  }    **public** **double** getCmtBandwidth() {  **return** cmtBandwidth;  }  **public** **void** setCmtBandwidth(**double** cmtBandwidth) {  **this**.cmtBandwidth = cmtBandwidth;  }  **public** **double** getPrice() {  **return** price;  }  **public** **void** setPrice(**double** price) {  **this**.price = price;  }    } |

DynamicVM.java(动态保证的用户)

|  |
| --- |
| **public** **class** DynamicVM {  **private** **double** size;//要发送的数据大小  **private** **double** time;//完成的时间  **private** **double** price;    **public** DynamicVM(){  **this**.size = 0;  **this**.time = 0;  }    **public** **double** getSize() {  **return** size;  }  **public** **void** setSize(**double** size) {  **this**.size = size;  }  **public** **double** getTime() {  **return** time;  }  **public** **void** setTime(**double** time) {  **this**.time = time;  }  **public** **double** getPrice() {  **return** price;  }  **public** **void** setPrice(**double** price) {  **this**.price = price;  }  } |

Mainform.java(运行模拟)

其中包括添加服务器中的虚拟机并为其设置相关带宽保证参数，有两种不同类型的虚拟机，在添加完虚拟机后可以通过运行SoftBW来模拟整个分配和定价的过程。最后输出运行情况和每个虚拟机在这个时间段的定价。

|  |
| --- |
| **import** java.math.BigDecimal;  **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.Scanner;  **public** **class** Mainform {  **public** **static** **final** **double** *p0*=0.5; //基础付费  **public** **static** **final** **double** *decrease* = 0.5; //若不使用严格的带宽要求则可以降低基础费用  **public** **static** **final** **double** *C*=1000; //服务器的总物理带宽    **public** **static** **void** main(String[] args) {  List VMs = **new** ArrayList();  Scanner in = **new** Scanner(System.*in*);  **int** select=0;    **double** speed=0; //记录现在的传输速率    **do**{  System.*out*.println("\*\*\*\*\*\*\*\*\*\*选择对应操作\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("1.添加虚拟机");  System.*out*.println("2.运行");  System.*out*.println("3.退出");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  select = in.nextInt();    **switch**(select){  **case** 1:*addVM*(VMs);**break**;  **case** 2:*SoftBW*(VMs, *C*);**break**;  }    }**while**(select != 3);    }    **public** **static** **void** addVM (List VMs){  **int** select=0;  Scanner in = **new** Scanner(System.*in*);  **do**{  System.*out*.println("\*\*\*\*\*\*\*\*\*\*选择带宽保证类型\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("1.严格带宽保证");  System.*out*.println("2.动态带宽保证");  System.*out*.println("3.退出");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  select = in.nextInt();    **switch**(select){  **case** 1:*addStrict*(VMs);**break**;  **case** 2:*addDynamic*(VMs);**break**;  }    }**while**(select != 3);    }    **public** **static** **void** addStrict(List VMs){  System.*out*.println("请输入承诺带宽");  Scanner in = **new** Scanner(System.*in*);  **double** Bandwidth = in.nextDouble();    StrictVM strictVm = **new** StrictVM();  strictVm.setCmtBandwidth(Bandwidth);    VMs.add(strictVm);  }    **public** **static** **void** addDynamic(List VMs){  System.*out*.println("请输入报文大小");  Scanner in = **new** Scanner(System.*in*);  **double** size = in.nextDouble();  System.*out*.println("请输入执行时间");  **double** time = in.nextDouble();    DynamicVM dynamicVm = **new** DynamicVM();  dynamicVm.setSize(size);  dynamicVm.setTime(time);    VMs.add(dynamicVm);    }    **public** **static** **void** SoftBW(List VMs , **double** C){  //该方法即为SoftBW的实现,C服务器总带宽  **int** time = 10; //此处将每次循环作为一个单位时间,更新保证速率  **double** cmtBandwidth=0;  **double** grtBandwidth=0;  **double** change;  **double** price=0;  **double** size;  **double** dynamicT;  **double** rt;  String type="";    BigDecimal b ;  **double** f=0;    Object obj = **new** Object();  **for**(**int** i=1 ;i<=time;i++){    System.*out*.println("第"+i+"秒：序号 类型 速率");  **for**(**int** j=0;j<VMs.size();j++){  obj = VMs.get(j);    **if**(obj **instanceof** StrictVM){  type = "严格带宽保证型";  StrictVM strictVm = (StrictVM) obj;  cmtBandwidth = strictVm.getCmtBandwidth();  change = Math.*random*()+0.5;  grtBandwidth = cmtBandwidth\*change;    **if**(cmtBandwidth < grtBandwidth){  price = strictVm.getPrice();  price = price+ *getPrice*(C,cmtBandwidth,cmtBandwidth);  price = price + *getPrice*(C, grtBandwidth-cmtBandwidth);  strictVm.setPrice(price);  }  }    **if**(obj **instanceof** DynamicVM){  type = "动态带宽保证型";  DynamicVM dynamicVm = (DynamicVM) obj;    size = dynamicVm.getSize();  dynamicT = dynamicVm.getTime();  change = Math.*random*()+0.5;  grtBandwidth = size/dynamicT \*change;  price = dynamicVm.getPrice();  price = price + *getPrice*(C, size, grtBandwidth, dynamicT);  dynamicVm.setPrice(price);  }  b = **new** BigDecimal(grtBandwidth);  f = b.setScale(2, BigDecimal.*ROUND\_HALF\_UP*).doubleValue();  System.*out*.println(" "+(j+1)+" "+type+" "+f);  }  }    System.*out*.println(" 序号 \t 类型 \t 总价");  **for**(**int** i=0 ;i<VMs.size();i++){  obj = VMs.get(i);  **if**(obj **instanceof** StrictVM){  type = "严格带宽保证型";  StrictVM strictVm = (StrictVM) obj;  price = strictVm.getPrice();  }  **if**(obj **instanceof** DynamicVM){  type = "动态带宽保证型";  DynamicVM dynamicVm = (DynamicVM) obj;  price = dynamicVm.getPrice();  }  b = **new** BigDecimal(price);  f = b.setScale(2, BigDecimal.*ROUND\_HALF\_UP*).doubleValue();  System.*out*.println(" "+ (i+1)+" "+type+" "+ f);  }    }    **public** **static** **double** getPrice(**double** C, **double** B,**double** rt){  **double** price = 0;  price = rt\*(1 + B/C)\**p0*;  **return** price;  }    **public** **static** **double** getPrice(**double** C, **double** size ,**double** rt , **double** time){  **double** price = 0;  price = rt\*(1 + size/time/C)\**decrease*\**p0*;  **return** price;  }    **public** **static** **double** getPrice(**double** C , **double** rt){  **double** price = 0;  price = rt\*rt/C\**decrease*\**p0*;  **return** price;  }  } |

# 5 实验结果

在本次测试中，添加了2台严格保证型和2台动态保证型的虚拟机，并输出10秒的运算情况以及最终定价结果。

|  |
| --- |
| \*\*\*\*\*\*\*\*\*\*选择对应操作\*\*\*\*\*\*\*\*\*\*\*  1.添加虚拟机  2.运行  3.退出  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  1  \*\*\*\*\*\*\*\*\*\*选择带宽保证类型\*\*\*\*\*\*\*\*\*\*\*  1.严格带宽保证  2.动态带宽保证  3.退出  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  1  请输入承诺带宽  20  \*\*\*\*\*\*\*\*\*\*选择带宽保证类型\*\*\*\*\*\*\*\*\*\*\*  1.严格带宽保证  2.动态带宽保证  3.退出  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  2  请输入报文大小  100  请输入执行时间  5  \*\*\*\*\*\*\*\*\*\*选择带宽保证类型\*\*\*\*\*\*\*\*\*\*\*  1.严格带宽保证  2.动态带宽保证  3.退出  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  1  请输入承诺带宽  30  \*\*\*\*\*\*\*\*\*\*选择带宽保证类型\*\*\*\*\*\*\*\*\*\*\*  1.严格带宽保证  2.动态带宽保证  3.退出  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  2  请输入报文大小  100  请输入执行时间  2  \*\*\*\*\*\*\*\*\*\*选择带宽保证类型\*\*\*\*\*\*\*\*\*\*\*  1.严格带宽保证  2.动态带宽保证  3.退出  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  3  \*\*\*\*\*\*\*\*\*\*选择对应操作\*\*\*\*\*\*\*\*\*\*\*  1.添加虚拟机  2.运行  3.退出  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  2  第1秒：序号 类型 速率  1 严格带宽保证型 29.93  2 动态带宽保证型 22.26  3 严格带宽保证型 44.04  4 动态带宽保证型 34.42  第2秒：序号 类型 速率  1 严格带宽保证型 15.32  2 动态带宽保证型 21.63  3 严格带宽保证型 29.58  4 动态带宽保证型 34.46  第3秒：序号 类型 速率  1 严格带宽保证型 28.9  2 动态带宽保证型 20.71  3 严格带宽保证型 32.22  4 动态带宽保证型 26.74  第4秒：序号 类型 速率  1 严格带宽保证型 26.51  2 动态带宽保证型 19.07  3 严格带宽保证型 36.73  4 动态带宽保证型 39.3  第5秒：序号 类型 速率  1 严格带宽保证型 10.02  2 动态带宽保证型 10.4  3 严格带宽保证型 41.0  4 动态带宽保证型 63.51  第6秒：序号 类型 速率  1 严格带宽保证型 25.49  2 动态带宽保证型 27.9  3 严格带宽保证型 18.73  4 动态带宽保证型 30.3  第7秒：序号 类型 速率  1 严格带宽保证型 16.18  2 动态带宽保证型 10.96  3 严格带宽保证型 32.58  4 动态带宽保证型 69.44  第8秒：序号 类型 速率  1 严格带宽保证型 23.21  2 动态带宽保证型 26.61  3 严格带宽保证型 31.05  4 动态带宽保证型 54.0  第9秒：序号 类型 速率  1 严格带宽保证型 19.4  2 动态带宽保证型 16.03  3 严格带宽保证型 41.92  4 动态带宽保证型 64.4  第10秒：序号 类型 速率  1 严格带宽保证型 27.0  2 动态带宽保证型 10.32  3 严格带宽保证型 19.57  4 动态带宽保证型 59.02  序号 类型 总价  1 严格带宽保证型 61.28  2 动态带宽保证型 47.4  3 严格带宽保证型 108.28  4 动态带宽保证型 124.84 |

通过运行结果可以看出，带宽要求高的严格要求用户所需要承担的费用要更高。序号1的严格保证型用户的承诺带宽和序号2的动态保证型用户的平均带宽一样，但是序号2的用户所需要支付的价格比序号1的低，这也反映了服务器提供商更加支持动态保证的方式，可以让他们分配闲置带宽，提高服务器总带宽的利用率。