# 云虚拟机资源分配的效用最大化模型

师雪霖 徐 恪

(清华大学计算机科学与技术系 清华信息科学与技术国家实验室(筹) 北京 100084)

**摘 要** 随着云计算和虚拟化技术的发展,为云资源管理提供了一种更高层次的调度选择:一个作业不再只能分配到一台物理机上,而是可将一台或多台物理机的计算资源虚拟化成一台虚拟机来运行该作业.根据作业需要,高效分配定量的物理资源放置虚拟机,是决定云系统性能的关键因素,即云资源调度问题实质就是一个虚拟机和物理机之间的映射问题.文中借鉴网络效用最大化模型,提出了一种云资源调度模型——云效用最大化(Cloud Utility Maximization,CUM)模型,与传统调度模型相比,目标函数不再是最小化最大完工时间,而是以达到效用最大为调度目标,可以充分提高用户的满意程度.通过求解 CUM 优化问题得到最优的虚拟机和物理机映射关系.设计了针对该模型的分解优化算法——简化次梯度算法求解拉格朗日对偶问题,证明了该算法可以获得原始模型问题的最优解.仿真实验表明算法可行且具有良好的收敛特性,并给出了 CUM 模型在真实云环境下的应用场景.

**关键词** 云计算;资源调度;虚拟机放置;拉格朗日松弛;次梯度算法

**中图法分类号** TP312 **DOI 号** 10.3724/SP.J.1016.2013.00252

# Utility Maximization Model of Virtual Machine Scheduling in Cloud Environment

SHI Xue-Lin XU Ke

(*Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

**Abstract** With development of cloud computing, especially wide application of virtual machines, it is possible that one or more physical machines can be virtualized one virtual machine to support a job. Virtual machine placement is a key factor for cloud performance, which is practically a mapping problem between virtual machine and physical machine. This paper gives a cloud scheduling model, Cloud Utility Maximization (CUM), which applied approach of Network Utility Maximization into computing resource scheduling. Comparing with traditional scheduling problem, CUM aims to maximize the utility of cloud instead of early finishing time. The optimal virtual machine placement policy can be achieved by solving CUM. The decomposition and optimization algorithm for the CUM, subgradient algorithm for solving Lagrangian relaxation dual problem, is proposed. Convergence of the algorithm is verified by the simulation experiments. At last the paper gives an application scenario of CUM in cloud environment.

**Keywords** cloud computing; scheduling; virtual machine placement; Lagrange relaxation; subgradient algorithm

---

收稿日期:2012-07-01;最终修改稿收到日期:2012-09-01.本课题得到国家科技支撑计划(2011BAK08B05-02)、国家"九七三"重点基础研究发展规划项目基金(2009CB320501,2012CB315803)、国家自然科学基金(61170292,60970104)和国家科技重大专项基金(2012ZX03005001001)资助.**师雪霖**,女,1977 年生,博士,助理研究员,主要研究方向为网络优化、云计算. E-mail: shixuelin@sina.com.
**徐 恪**,男,1974 年生,博士,教授,博士生导师,主要研究领域为下一代互联网、交换和路由结构、P2P 网络、Overlay 网络、物联网.

## 1 引　言

云计算(Cloud Computing)是继并行计算、分布式计算、网格计算后的新型计算模式[1]，云可视为集群和网格的组合. 云环境中包含着大量分散、异构资源，包括处理器、内存、存储、可视化设备、软件等等. 云计算要解决的问题是如何有效安全地管理和共享接入云的各种资源，并提供相应的服务，它强调的是全面的资源共享，全面的应用服务. 云计算提出了全互联网范围内共享资源的最高目标，最大限度地充分利用计算/存储资源，是整合全社会高性能计算资源的有效方法.

在云环境中，不仅资源的地理位置分布广泛，甚至属于不同的自治系统，而且资源往往具有异构性、动态性，如何有效分配计算资源是决定整个云性能、效率的关键问题. 因此，随着云计算技术的日益普及，有效的云资源调度模型和算法将成为高效利用这些资源的关键. 由于云计算模式从提出伊始即考虑到其商业实现，所以从经济效用度量的调度模型更具意义.

云计算中的资源调度算法多延续网格资源调度，这些传承于 20 世纪六、七十年代即开始的多处理器、并行机上的作业调度研究，其数学模型是整数规划(integer programming)问题，以尽可能短的任务完成时间为调度目标，采用启发式算法求解次优解. 随着网格计算和云计算的普及，资源调度越来越多考虑经济效益，在模型中增加了经济度量参数，如机器价格、最迟完工时间等. 但是无论是否包含经济效益参数，这些调度模型的解都是 0-1 变量矩阵，即若作业 $i$ 分配到机器 $j$ 上，则 $x_{ij}=1$；否则为 0.

由于云计算普遍采用虚拟化技术，给云用户分配的计算资源，并不是真正独占一台或多台的物理机(Physical Machine，PM). 以亚马逊 EC2 平台为例，用户通过付费可以购买“一台”计算服务器若干小时的使用权，但实际上在用户使用过程中，并不是始终独占使用一台真正的物理机，而有可能是一台或者多台物理机提供的虚拟机(Virtual Machine，VM)服务. 虚拟化技术是推动云计算发展的重要动力，一台物理机可以实例化多个虚拟机，而多台物理机的剩余计算资源也可以虚拟化成一台虚拟机. 随着虚拟化技术的不断发展，可以预计虚拟化本身消耗的计算资源会降低，而且从便于管理和安全性角度考虑，虚拟机方式也会被云服务提供商广泛采用.

因此，云计算资源调度可以不使用整数规划模型，而是变成如何将物理机的计算资源按比例分配的优化问题，通过虚拟机技术将这些资源提供给云用户使用. 本文提出了一种云资源调度效用最大化模型，描述云计算资源优化问题，从经济学角度分析了该模型的理论意义. 此外设计了求解该问题的次梯度优化算法. 最后给了模拟实验数据，分析了算法性能.

## 2 相关研究工作

最基本的多处理器、并行机上的作业调度研究，其数学模型实质是整数规划问题，以尽可能短的任务完成时间为调度目标，目标函数为最大完工时间最小，数学模型为

$$\begin{cases}\min C_{\max}\\ \text{s. t.}\ \sum\limits_{1\le j\le m} x_{ij}t_{ij}\le C_{\max}\\ \qquad x_{ij}\in\{0,1\},\ \sum\limits_{1\le j\le m} x_{ij}=1\end{cases}\tag{1}$$

其中变量定义如下：

$M=\{1,2,\cdots,m\}$：表示机器的集合，共有 $m$ 台机器；

$J=\{1,2,\cdots,n\}$：表示所有作业的集合，共有 $n$ 个作业；

$\boldsymbol{T}$：加工时间矩阵($n\times m$ 矩阵)，矩阵元素 $t_{ij}$ 表示作业 $i$ 在机器 $j$ 上的加工时间；

目标函数最大完工时间定义如下：

$$C_{\max}=\max\Big(\sum_{1\le j\le m} x_{ij}t_{ij}\Big)\tag{2}$$

该优化问题的目标解为 $n\times m$ 矩阵 $\boldsymbol{X}$，矩阵元素 $x_{ij}$ 表示作业 $i$ 是否在机器 $j$ 上加工，$x_{ij}\in\{0,1\}$，如果加工为 1，否则为 0.

根据目标函数定义，可以将模型(1)整理为如下简化形式：

$$\begin{cases}\min \sum\limits_i\sum\limits_j x_{ij}t_{ij}\\ \text{s. t.}\ x_{ij}\in\{0,1\}\\ \qquad \sum\limits_{1\le j\le m} x_{ij}=1\end{cases}\tag{3}$$

这类整数规划模型，多采用启发式算法来寻找次优解：如经典的 Min-min、Max-min、sufferage、Xsufferage 等作业调度算法[2] 等. 网格调度中也多采用此类模型. 随着现代优化计算方法的快速发展，禁忌搜索(Tabu Search)、模拟退火(Simulated

Annealing)、遗传算法(Genetic Algorithm)/进化算法(Evolutionary Algorithm)、蚁群优化算法(Ant Colony Optimization Algorithm)和人工神经网络(Artificial Neural Networks)算法等也被用于网格调度,被很多后续研究加以改进,如文献[3-5].

随后也出现了一些考虑经济效益的模型,增加了经济参数,如机器使用价格、作业预算、作业最迟完工时间(deadline)等参数[6-7],这类模型通常形式如下:

$$\begin{cases} \min \sum_i \sum_j x_{ij} t_{ij} P_j \\ \text{s.t.} \sum_j x_{ij} t_{ij} P_j \leq B_i \\ \sum_j x_{ij} t_{ij} \leq D_i \\ x_{ij} \in \{0,1\}, \sum_{1 \leq j \leq m} x_{ij} = 1 \end{cases} \tag{4}$$

其中新增的经济参数定义如下:

$P_j$:机器 $j$ 的价格;

$B_i$:作业 $i$ 的预算;

$D_i$:作业 $i$ 的最迟完工时间.

这类模型在基本平行机调度模型(3)的基础上增加了更多约束条件,也对求解算法提出了更高的要求,解决方法仍然采用遗传算法及衍生算法居多.

上述模型研究的都是离线排序(off-line scheduling)问题,即预先知道作业的全部信息,其中作业在某机器上的加工时间 $t_{ij}$ 需要估计得到.方法如下:估算作业所包含的机器指令条数[8-9],根据不同计算节点的 MIPS(每秒可处理的百万条指令数,Million Instructions per second)性能,计算出该作业在该节点上所需要的处理时间.在所有作业进入排序系统前,可以先经过一个预处理环节,估算加工时间.

但是这类调度模型都存在如下问题:首先,目标解限定为 0-1 变量,其实质是线性整数规划问题,从理论上说线性整数规划问题可以转化为一个线性规划问题,但是从计算角度看实现这种转化是相当困难的.其次,针对此类模型多采用启发式算法搜索局部最优解,针对不同的输入条件,算法的收敛速度可能差别较大,很难保证调度效率.

随着云计算的不断普及,尤其是虚拟化技术的发展,计算资源的调度可以不再局限于作业是否分配到某台物理机的二元选择限制,而是从效益最大化的角度出发,进行整体计算资源的配置,分配给某个作业的计算资源可以是多台物理机提供的剩余资源.虽然多台物理机并行处理会产生额外开销,但是却使得计算资源的管理更为有效.

这样资源调度问题就相当于虚拟机放置问题,如何分配虚拟机和物理节点的映射成为影响云性能的关键,这也是云资源调度的研究热点[10].

因此,本文借鉴网络效用最大化(Network Utility Maximization,NUM)模型,提出了一种云资源调度效用最大化模型,简称为云效用最大化(Cloud Utility Maximization,CUM)模型.

近年来,NUM 模型成为计算机网络体系架构和协议的重要研究方向.文献[11]最先提出了 NUM 模型,文献[12]给出了针对 NUM 模型常用的优化分解方法和分布式算法.基本的 NUM 模型可表示成如下形式[13]:

$$\begin{cases} \max \sum_s U_s(x_s) \\ \text{s.t.} \quad \boldsymbol{Rx} \leq \boldsymbol{c} \end{cases} \tag{5}$$

其中,$\boldsymbol{R}$ 为路由矩阵(routing matrix);向量 $\boldsymbol{c}$ 为各链路最大的传输带宽;$U_s$ 为第 $s$ 个服务的效用函数,通常效用函数是一个光滑、递增的凹函数,并且只依赖于该服务所获得的网络带宽 $x_s$;问题的目标解向量 $\boldsymbol{x}$,每个分量 $x_s$ 表示为第 $s$ 个服务所分配的网络带宽.

NUM 优化问题往往是凸规划,可以获得全局最优解.通过反向工程分析,已经证实了按照一些效用函数进行网络带宽分配,可以保证不同的公平策略要求.

考虑到云计算经济化、市场化的发展趋势,云环境中的计算资源可以参照网络带宽分配策略.有学者认为计算资源将成为继水、电、气、电信之后的第五大公共基础设施(utility)[1],越来越多的用户会直接租用计算资源而不再自己购买硬件.因此,未来的云可能会象目前的 Internet 网络一样,同时会有很多并发请求.如何像网络一样,将有限的资源分配给多个等待的作业,保证总效用最大才是调度的关键.而虚拟化技术也使得将所有计算资源统一度量分配成为可能,可以不考虑物理硬件的异构和限制,而是采用统一的描述框架量化所有的物理资源,在效益最大化的目标下进行整体分配,这将是进行云资源调度的一个有效方法.

基于效用最大化来调度云资源也开始被逐渐采用,在文献[14]中提出了云环境中内容提供商如何分发服务的方法,但不属于作业和资源调度层面的问题.文献[15]提出了一种云资源管理方法,模型的

目标是使云效用最大，不过最终构造的仍然是 NP-Hard 问题. 本文将 NUM 模型的效用函数、分解算法加以改进，用在云计算资源调度中.

## 3　云效用最大化模型

在 NUM 模型中，网络资源一般采用带宽作单位，不同类型的网络传输介质，采用带宽这个单位可以统一衡量. 网络用户发起的一个网络服务可以使用多条网络路径，一条网络路径上也可以承载多个服务. 参照网络资源模型，本节首先提出了云资源描述框架，使异构的云计算资源能用单一量化标准计量，便于统一调度. 其次介绍了 CUM 模型及其效用函数的定义.

### 3.1　云资源描述框架

与传统的并行机调度不同，在云环境下，采用虚拟机技术，可以利用多台物理机(PM)的剩余计算资源实例化成一台虚拟机(VM)，云资源调度问题就可以简化成：如何分配物理机资源以达到效用最大化. 因此可以用图 1 来表示云资源物理机和虚拟机的框架结构.

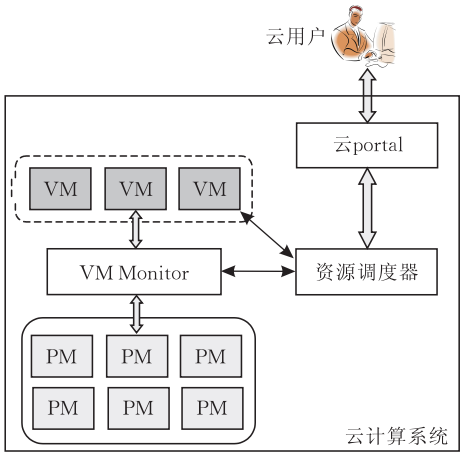



图 1　云资源框架

理想的云计算系统中地理位置分散的、异构的物理机都可以统一调度. 用户通过云 portal 提交作业，资源调度器根据调度策略，决定如何为作业分配物理资源，通过虚拟机监控器(VM Monitor)把相应的物理资源映射为虚拟机.

如同计算机网络环境下，不同拓扑路径都可以用统一的带宽来衡量. 在云计算中，一般衡量计算资源主要考查 CPU、内存、I/O 等，但由于调度性能的限制，常见的调度算法只考虑一个 CPU 维度，少量工作能扩展到 CPU、内存两个维度[11].

在本文提出的云资源描述框架下，可以不必关心物理机的差异，而是为所有计算资源制定统一的衡量方式. 这也符合目前一些云服务提供商的经营模式，例如按 CPU 小时计费等. 在本文的云资源描述中，做如下假设：首先只考虑作业对 CPU 的占用，假设内存和 I/O 足够；其次所有物理机(PM)的计算能力采用同一标准度量，即 MIPS. 在实际应用环境下，还可以根据需要设定别的衡量方式，如可用 CPU 机时，或是 CPU 和内存的混合限制.

因此，云资源可以用集合$\{C_1,\cdots,C_j,\cdots,C_m\}$表示，集合中的每个元素 $C_j$ 表示物理机 $j$ 的最大计算能力，单位为 MIPS.

### 3.2　CUM 模型

在上述云资源描述框架下，可以将云资源调度问题用如下 CUM 模型表示(设有 $n$ 个作业，$m$ 台物理机)：

$$p:\begin{cases}\max \sum_i U_i(y_i)\\ \text{s.t.} \sum_j x_{ij}=y_i\\ \sum_i x_{ij}\le C_j\\ \frac{I_i}{y_i}\le D_i\\ x_{ij}\ge 0\end{cases}\tag{6}$$

其中，$C_j$ 表示某物理机 $j$ 的最大计算能力，单位为 MIPS；$I_i$ 表示作业 $i$ 的计算量，作业所包含的机器指令条数，单位 MI(百万条指令数)，可估算得到；$D_i$ 表示作业 $i$ 的最迟完工时间，从提交作业到作业完成所能允许的最长时间，单位 s(秒).

为了便于表示，引入中间变量 $y_i$，表示分配给作业 $i$ 的所有计算资源总和，即 $y_i=\sum_j x_{ij}$，单位为 MIPS. $\sum_i x_{ij}$ 表示某物理机 $j$ 被占用的计算资源总合，必须小于等于该物理机 $j$ 的最大计算能力 $C_j$.

约束条件$\frac{I_i}{y_i}\le D_i$表示作业所获得的计算资源总和必须能使得作业在最迟完工时间前完成. 为了便于处理，将该约束条件变形为

$$y_i\ge\frac{I_i}{D_i}.$$

目标函数为所有作业(即所有云用户)的效用函数 $U_i(y_i)$ 最大，选取不同的效用函数可以实现不同的公平控制策略，在下一小节中将介绍效用函数选择.

显然,CUM 模型要求所有输入参数必须满足如下条件:

$$\sum_{j=1}^{m} C_j \geq \sum_{i=1}^{n} \frac{I_i}{D_i},$$

即当所有作业要求的计算资源超出物理机总计算能力的情况不在模型考虑范围内.

在模型(6)中,目标解为矩阵 $\boldsymbol{X}$,矩阵中每个元素 $x_{ij}$ 的含义为每个物理机分给每个作业的计算资源,不局限为 0-1 变量,与传统调度模型(3)、(4)相比,不再是整数规划问题. 传统的调度模型,一个作业只能分给一台机器,要求目标解必须是 0-1 矩阵. 而本文提出的 CUM 模型目标是找出物理机分配多少比例的计算资源给某个作业,一个作业可以通过虚拟机技术同时使用多个物理机的资源,而且并不独占,物理机剩余的计算资源可以分给其它作业.

模型(6)没有考虑虚拟化技术、并行通信所造成的额外 CPU 消耗. 通常作业分配到越多的物理机上执行额外开销就越大,但这个约束条件难以确定描述. 此外虚拟机技术虽然带来额外开销,但是可以给云用户提供不同的操作系统环境,安全性也更有保证,随着虚拟机技术的普遍应用,这些额外开销可以视为固定成本,在云资源调度中可以不单独计算.

此外模型(6)中没有考虑作业到达时间,也不考虑排队,而是针对现有作业,如何将所有的 PM 资源划分成不同 VM 分给不同的作业. 在传统并行机、网格调度中都是要考虑作业到达时间,而且都是有排队情形,作业排队优先级不同. 本文 CUM 模型中没体现时间因素,因为在常见的网络 NUM 模型中,也只考虑当前发起的网络请求如何共享带宽,也没考虑时间问题. 网络一般都是满载的,现有的应用能合理分配带宽避免拥塞即可. 随着云计算的普及,云计算资源的分配也会向网络分配带宽一样,针对现有作业合理分配计算资源. 而且考虑到与网络的不同,本文模型还是考虑了部分时间因素:作业的最迟完工时间,要保证分配给作业足够计算资源以便在最迟完工时间前完成.

### 3.3 效用函数

效用函数描述了用户对于所获得的某种服务在一定衡量单位下的满意度[16]. 以 NUM 模型为例,效用函数是所获得的网络带宽的单调递增函数,一般采用如下效用函数:

$$U(x)=\begin{cases} w\log x, & \alpha=1 \\ w(1-\alpha)^{-1}x^{1-\alpha}, & \alpha>1 \end{cases} \quad (7)$$

其中,$x$ 表示用户获得的服务;$w$ 为用户获得该服务而愿意提供的支付(willingness-to-pay). 通过计算机网络反向工程分析结果证实,当 $\alpha=1$ 时,效用函数 $U(x)=w\log x$ 也称为对数函数,对数效用函数可以实现竞争网络资源的各用户之间的比例公平性(proportional fair);当 $\alpha>1$ 时,效用函数 $U(x)=w(1-\alpha)^{-1}x^{1-\alpha}$ 也称为负指数函数,可提供 $\alpha$-公平性($\alpha$-fair).

对数效用函数被广泛用于有线网络和无线网络的网络资源分配算法中,传统的 TCP 协议实际达成的带宽分配恰恰符合的是 $w=1$ 的对数效用函数. 对数效用函数在 $[0,+\infty)$ 区间内的曲线如图 2 所示.

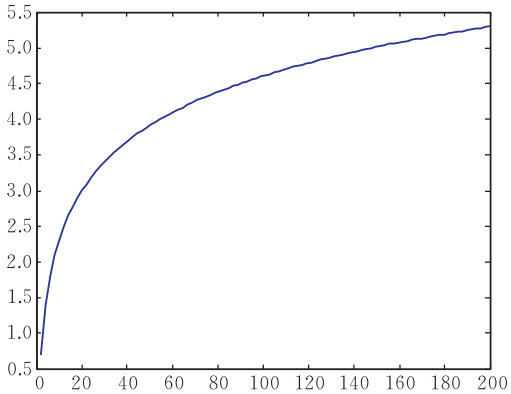



图 2 对数效用函数

如图 2 所示,对数效用函数为单调递增的凹函数(concave function),且增长速度趋于平缓,即在区间内任意点 $x_0$ 处的一阶导数大于 0,二阶导数小于 0,这也是经济学中选取效用函数的基本要求. 为了便于描述,给出如下条件.

**引理 1.** 效用函数选取条件:如果函数 $U(x)$ 在自变量约束区间内任意点 $x_0$ 处的一阶导数 $U'(x_0)>0$,二阶导数 $U''(x_0)<0$,则可表示在某种情况下用户对获得某种服务的满意度.

本文目前的工作只选定对数效用函数进行实验模拟. CUM 模型本身不局限于特定的效用函数,只要引理 1 的函数均可使用. 选择不同的效用函数,可以实现不同目标的资源调度策略. 在后续的工作中将对不同效用函数产生的结果进行比较.

### 3.4 模型最优性条件

已知效用函数 $U_i(y_i)=w_i\log(y_i)$ 为递增的凹函数,则式(5)描述的 CUM 模型的目标函数也是凹函数. 现考察其约束域情况,式(5)中显式约束条件有两个,$\sum_i x_{ij}\leq C_j$,$y_i\leq\frac{I_i}{D_i}$,将变量 $y_i$ 替换成 $\sum_j x_{ij}$,

化为如下形式：

$$\sum_i x_{ij} \leq C_j \tag{c1}$$

$$\sum_j x_{ij} \geq \frac{I_i}{D_i} \tag{c2}$$

其中，约束条件(c2)表示为作业 $i$ 分配的资源要足够保证作业能在最迟完工时间前完工，实际只要不晚于最迟完工时间即可. 因此，在不影响最优解的前提下，可以将约束条件(c2)写成：

$$\sum_j x_{ij} = \frac{I_i}{D_i} \tag{c2}$$

即 CUM 模型的显式约束条件可表示为如下线性不等式和等式集合：

$$p = \{\boldsymbol{x} \mid \boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x} \leq C_j,\ j=1,\cdots,m,$$

$$\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{x} = \frac{I_i}{D_i},\ i=1,\cdots,n\}.$$

该集合定义了一个多面体，是一个典型的凸集. 图 3 以 5 台物理机、5 个作业($m=5, n=5$)为例，给出了 CUM 模型解空间.

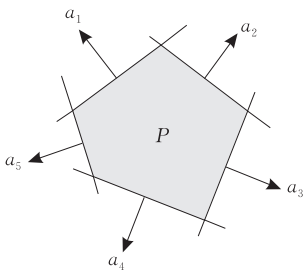



图 3　CUM 模型解空间

如上所述，目标函数为单调递增凹函数，约束域是凸集，则 CUM 优化问题是一个凸规划问题(convex optimization)，存在最优解.

# 4　优化算法

本节将通过拉格朗日分解方法简化模型，设计次梯度求解算法.

拉格朗日松弛算法是 20 世纪 70 年代被提出来的[17-18]，该算法主要包括以下几个要点：松弛约束的选择、拉格朗日算子的调整和可行解的构造等. 次梯度算法是处理拉格朗日对偶函数的有效方法[19-20].

## 4.1　拉格朗日松弛

用拉格朗日因子松弛原问题的优化条件，得到拉格朗日函数：

$$L(\boldsymbol{x},y;\lambda,\mu) = \sum_{i=1}^{n} U_i(y_i) + \sum_{j=1}^{m} \lambda_j \left(C_j - \sum_{i=1}^{n} x_{ij} - \delta_j\right) + \sum_{i=1}^{n} \mu_i \left(\frac{I_i}{D_i} - y_i - \gamma_i\right) \tag{8}$$

其中，$\lambda_j \geq 0, \mu_i \geq 0$ 为拉格朗日因子；$\delta_j \geq 0, \gamma_i \geq 0$ 为松弛变量.

从实际经济含义考虑，拉格朗日因子 $\lambda_j$ 可以看作物理机为提供单位计算资源而收取的价格，$\mu_i$ 可以看作提前完成作业的奖励价格(同时也是无法在最迟完工时间前完成作业的惩罚价格).

松弛变量 $\delta_j$ 表示物理机 $j$ 剩余的计算能力，松弛变量 $\gamma_i$ 表示作业 $i$ 提前完工的时间(即最迟完工时间和实际完工时间的差值). 因为从效益上来讲，每个作业应尽可能获得更多的计算资源，因此可令松弛变量 $\delta_j$ 为 0. 为了简化优化问题，只需要尽量保证作业在最迟完工时间时执行完毕，不必考虑提前完成，因此可令松弛变量 $\gamma_i$ 为 0. 将式(7)整理后如下：

$$L(\boldsymbol{x};\lambda,\mu) = \sum_{i=1}^{n} U_i \left(\sum_{j=1}^{m} x_{ij}\right) + \sum_{j=1}^{m} \lambda_j \left(C_j - \sum_{i=1}^{n} x_{ij}\right) + \sum_{i=1}^{n} \mu_i \left(\frac{I_i}{D_i} - \sum_{j=1}^{m} x_{ij}\right) \tag{9}$$

为了简化问题，再不影响 CUM 模型含义的情况下，将对数效用函数简化，代入式(9)，得到

$$L(\boldsymbol{x};\lambda,\mu) = \sum_{i=1}^{n} w_i \left(\sum_{j=1}^{m} \log x_{ij}\right) + \sum_{j=1}^{m} \lambda_j \left(C_j - \sum_{i=1}^{n} x_{ij}\right) + \sum_{i=1}^{n} \mu_i \left(\frac{I_i}{D_i} - \sum_{j=1}^{m} x_{ij}\right) \tag{10}$$

式(10)为最终得到的拉格朗日函数，记作 $L$. 由此可得原问题的对偶问题，记作 $LD$. 针对对偶问题 $LD$，设计了优化算法求解目标解.

## 4.2　简化次梯度算法

次梯度优化算法(Subgradient Optimization Algorithm)通过求解对偶问题而逐步逼近原问题. 次梯度算法是求解不可微优化问题的最常用方法[21]. 在每次迭代中更新拉格朗日因子 $\boldsymbol{u}$：

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + s^{(k)}\boldsymbol{g}^{(k)} \tag{11}$$

其中，$s^{(k)}$ 为迭代步长；$\boldsymbol{g}^{(k)}$ 为第 $k$ 次迭代中 $L^{(k)}$ 的次梯度，即迭代的下降方向.

针对 CUM 模型，本文设计了一种简化的次梯度算法，称为 CUM 简化次梯度算法. 主要思想是简化每一步对下降方向的求解，以拉格朗日因子的导数作为次梯度 $\boldsymbol{g}$.

在式(10)中，拉格朗日因子 $\boldsymbol{u} = [\boldsymbol{\lambda}, \boldsymbol{\mu}]^{\mathrm{T}}$，其中每个分量 $\lambda_j$ 和 $\mu_i$ 均可微，因此令

$$\boldsymbol{g}=\begin{bmatrix}\dfrac{\partial L}{\partial \lambda}\\ \dfrac{\partial L}{\partial \mu}\end{bmatrix}=\begin{bmatrix}\left[C_1-\sum_{i=1}^{n}x_{i1}\ \cdots\ C_m-\sum_{i=1}^{n}x_{im}\right]^{\mathrm{T}}\\ \left[D_1-\dfrac{I_1}{\sum_{j=1}^{m}x_{1j}}\ \cdots\ D_n-\dfrac{I_n}{\sum_{j=1}^{m}x_{nj}}\right]^{\mathrm{T}}\end{bmatrix} \tag{12}$$

即次梯度 $\boldsymbol{g}$ 为 $m+n$ 维矢量.

完整的 CUM 简化次梯度算法描述如下.

**算法 1.** CUM 简化次梯度算法.

输入：$m$ 台物理机计算资源集合 $C=\{C_1,\cdots C_j,\cdots,C_m\}$，

$n$ 个作业愿意提供的支付集合 $W=\{w_1,\cdots,w_i,\cdots,w_n\}$，

$n$ 个作业的计算量集合 $I=\{I_1,\cdots,I_i,\cdots,I_n\}$，

$n$ 个作业的最迟完工时间集合 $D=\{D_1,\cdots,D_i,\cdots,D_n\}$

输出：目标解矩阵 $\boldsymbol{X}(n\times m$ 矩阵)

算法步骤：

1. 设定初始点——$\boldsymbol{X}^{(1)}$ 为全零矩阵，即任意元素 $x_{ij}=0$，$\boldsymbol{\lambda}^{(1)}=[1,1,\cdots]^{\mathrm{T}}$，$\boldsymbol{\mu}^{(1)}=[w_1,w_2,\cdots,w_n]^{\mathrm{T}}$，允许误差 $\varepsilon>0$，置 $k=1$.

2. 计算次梯度 $\boldsymbol{g}^{(k)}$.

3. 将 $\boldsymbol{\lambda}^{(k)}$ 和 $\boldsymbol{\mu}^{(k)}$ 代入拉格朗日函数 $L$，令 $\dfrac{\partial L}{\partial x_{ij}}=0$，解方程组，得到第 $k$ 次迭代目标解 $\boldsymbol{X}^{(k)}$.

4. 若 $\|\boldsymbol{g}^{(k)}\|\leq\varepsilon$，则停止迭代；否则求步长 $s^{(k)}$，令

$$s^{(k)}=\frac{\bar{L}-L(\boldsymbol{x}^{(k)},\boldsymbol{\lambda}^{(k)},\boldsymbol{\mu}^{(k)})}{\|\boldsymbol{g}^{(k)}\|^2},$$

其中，$\bar{L}$ 为拉格朗日对偶问题的估计最优目标值，采用估计值；若计算出的步长 $s^{(k)}\leq 0$，则继续采用上一次步长，即 $s^{(k)}=s^{(k-1)}$.

5. 利用式(11)更新拉格朗日算子，求 $\boldsymbol{\lambda}^{(k+1)}$ 和 $\boldsymbol{\mu}^{(k+1)}$，$k=k+1$，转步 2.

可以根据需要设置允许误差 $\varepsilon$，获得不同精度的最优解. 算法中使用的估计最优目标值 $\bar{L}$ 可以通过启发式搜索算法估算，或者直接使用 Matlab、LINGO 等软件计算得到. 文献[21]从理论上分析了估计值 $\bar{L}$ 的偏差对计算结果产生的影响. 在 CUM 简化梯度算法中，通过步 4 的调整，减少了这种偏差对结果的影响.

### 4.3 算法收敛性

假设 $\boldsymbol{u}^*$ 为拉格朗日对偶问题的最优解，$L^*$ 为此时拉格朗日函数值. 在 CUM 简化次梯度算法中，第 $k+1$ 次迭代得到的拉格朗日因子 $\boldsymbol{u}^{(k+1)}$ 必定比第 $k$ 次迭代的 $\boldsymbol{u}^{(k)}$ 更逼近 $\boldsymbol{u}^*$，即

$$\|\boldsymbol{u}^*-\boldsymbol{u}^{(k+1)}\|<\|\boldsymbol{u}^*-\boldsymbol{u}^{(k)}\|,\ \forall k \tag{13}$$

证明如下：

$$\begin{aligned}&\|\boldsymbol{u}^*-\boldsymbol{u}^{(k+1)}\|^2=\|\boldsymbol{u}^*-\boldsymbol{u}^{(k)}-s^{(k)}\boldsymbol{g}^{(k)}\|^2\\&=\|\boldsymbol{u}^*-\boldsymbol{u}^{(k)}\|^2-2s^{(k)}\boldsymbol{g}^{(k)}(\boldsymbol{u}^*-\boldsymbol{u}^{(k)})+(s^{(k)})^2\|\boldsymbol{g}^{(k)}\|^2\\&=\|\boldsymbol{u}^*-\boldsymbol{u}^{(k)}\|^2-s^{(k)}(2\boldsymbol{g}^{(k)}(\boldsymbol{u}^*-\boldsymbol{u}^{(k)})-s^{(k)}\|\boldsymbol{g}^{(k)}\|^2).\end{aligned}$$

根据求步长公式，可得

$$\|\boldsymbol{u}^*-\boldsymbol{u}^{(k+1)}\|^2=\|\boldsymbol{u}^*-\boldsymbol{u}^{(k)}\|^2-s^{(k)}(2\boldsymbol{g}^{(k)}(\boldsymbol{u}^*-\boldsymbol{u}^{(k)})-(L^*-L^{(k)})).$$

根据拉格朗日对偶函数的性质，可知，

$$\boldsymbol{g}^{(k)}(\boldsymbol{u}^*-\boldsymbol{u}^{(k)})\geq L^*-L^{(k)},$$

所以，

$$2\boldsymbol{g}^{(k)}(\boldsymbol{u}^*-\boldsymbol{u}^{(k)})-(L^*-L^{(k)})>0,$$

则

$$-s^{(k)}(-2\boldsymbol{g}^{(k)}(\boldsymbol{u}^*-\boldsymbol{u}^{(k)})+L^*-L^{(k)})<0,$$

可得

$$\|\boldsymbol{u}^*-\boldsymbol{u}^{(k+1)}\|^2<\|\boldsymbol{u}^*-\boldsymbol{u}^{(k)}\|^2.$$

式(13)得证，因此 CUM 简化次梯度算法是收敛的.

## 5 实验和分析

为了验证 CUM 简化次梯度算法的可行性和收敛性能，本节给出仿真计算结果，并分析了算法复杂性. 最后给出 CUM 模型在真实云环境下的应用场景.

### 5.1 仿真结果

随机生成物理机和作业相关参数，作为算法输入，以模拟不同规模的云资源调度情况. 生成的随机数满足如下条件：

$$\sum_{j=1}^{m}C_j\geq\sum_{i=1}^{n}\frac{I_i}{D_i} \tag{14}$$

即所有物理机计算能力总和(单位：MIPS)≥所有作业的计算量(单位：MI)与最迟完工时间(单位：s)商之和.

实验运行硬件环境：一台 PC 机，CPU 主频 2.4 GHz，内存 4 GB.

首先，为了验证算法的可行性，设计了一组规模较小的输入参数(3 个作业，3 台物理机)，这样规模较小的问题，Matlab 的 fmincon 函数可算出较好结果，以便和本文算法比较，该类模拟问题记作 $P(3,3)$. 指定输入参数如下：有 3 台物理机，计算资源分别为 $C=\{1,2,3\}$；有 3 个作业：$W=\{1,2,3\}$，$I=\{3,2,1\}$，$D=\{3,2,1\}$. 此时构造了问题 $P(3,3)$ 的一个实例，记作 $P_1(3,3)$.

表 1 给出了针对 $P_1(3,3)$，本文算法和 Matlab 的计算结果.

**表 1　本文算法所得解与 fmincon 比较**

| | $P_1(3,3)$的计算结果 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $x_{11}$ | $x_{12}$ | $x_{13}$ | $y_1$ | $x_{21}$ | $x_{22}$ | $x_{23}$ | $y_2$ | $x_{31}$ | $x_{32}$ | $x_{33}$ | $y_3$ | $\sum_i U_i(y_i)$ |
| 本文算法 | 0.2333 | 0.3333 | 0.4333 | 0.9999 | 0.5667 | 0.6667 | 0.7667 | 2.0001 | 0.2000 | 1.0000 | 1.8000 | 3.0000 | 4.6820 |
| fmincon | 0.0901 | 0.2887 | 0.6221 | 1.0009 | 0.2883 | 0.6890 | 1.0224 | 1.9997 | 0.6216 | 1.0223 | 1.3556 | 2.9995 | 4.6821 |

从表 1 中可以看出，对于问题 $P_1(3,3)$，使用本文算法得到的最优解矩阵 $\boldsymbol{X}$ 和 Matlab 的 fmincon 函数结果虽然不尽相同，但是目标函数值基本相同，证明了本文算法的正确性.

为了验证算法的收敛性，将问题规模扩大，构造的问题记作 $P(n,m)$，其中 $n$ 表示作业数，$m$ 表示物理机数. 分别构造 3 类规模的问题：$P(5,10)$、$P(10,5)$、$P(50,50)$.

3 类问题的输入参数：物理机计算能力 $C_j$、作业愿意提供的支付 $w_i$、作业计算量 $I_i$、最迟完工时间 $D_i$，均采用符合均匀分布的随机数，如果生成的随机数不满足式(13)的条件，则重新生成.

对上述 3 类模拟问题，主要分析算法的迭代次数. 每类问题随机生成 4 组输入参数，即构造出问题的 4 个实例，比较迭代次数.

表 2 给出了问题 $P(5,10)$的 4 个实例的计算结果.

**表 2　$P(5,10)$时的迭代情况**

| 迭代次数 | 4 个子问题的目标函数值 | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 10 | 17.515 | 14.390 | 13.575 | 9.468 |
| 20 | 18.153 | 15.398 | 13.972 | 10.158 |
| 30 | 18.274 | 15.918 | — | 10.472 |
| 40 | — | — | — | — |

$P(5,10)$规模下构造的 4 个子问题，其迭代收敛情况如图 4 所示.

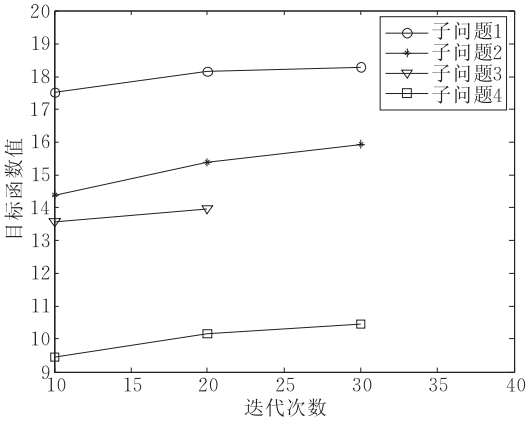



图 4　$P(5,10)$迭代收敛曲线图

表 3 给出了问题 $P(10,5)$的 4 个实例的计算结果.

**表 3　$P(10,5)$时的迭代情况**

| 迭代次数 | 4 个子问题的目标函数值 | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 10 | 63.959 | 69.390 | 77.972 | 63.068 |
| 20 | 64.153 | 70.918 | 78.758 | 64.269 |
| 30 | 65.477 | 71.858 | 79.038 | 65.828 |
| 40 | 66.827 | 72.133 | 80.254 | 66.593 |
| 50 | 67.013 | 73.254 | 81.442 | 67.397 |
| 75 | 67.886 | — | — | — |
| 100 | — | — | — | — |

图 5 为对应的收敛曲线图.

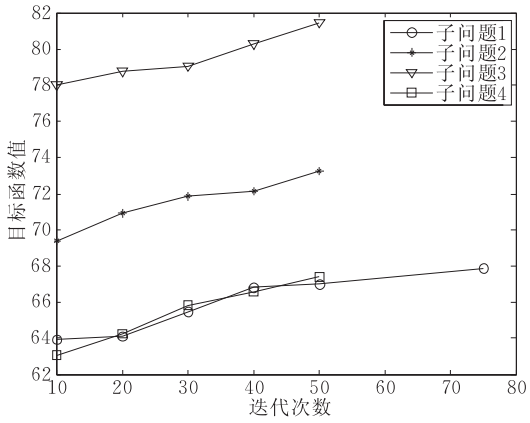



图 5　$P(10,5)$迭代收敛曲线图

表 4 给出了问题 $P(50,50)$的 4 个实例的计算结果.

**表 4　$P(50,50)$时的迭代情况**

| 迭代次数 | 4 个子问题的目标函数值 | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 10 | 520.876 | 408.879 | 653.869 | 718.983 |
| 20 | 548.021 | 439.203 | 681.021 | 742.387 |
| 30 | 579.954 | 471.936 | 714.763 | 770.865 |
| 40 | 593.286 | 506.147 | 748.011 | 791.012 |
| 50 | 602.011 | 521.978 | 769.613 | 813.432 |
| 75 | 616.947 | 529.743 | 778.864 | 819.165 |
| 100 | 620.842 | 533.879 | 781.276 | 822.034 |
| 125 | 623.431 | 535.327 | 784.012 | 824.986 |
| 150 | 625.017 | 536.414 | 785.873 | 825.484 |
| 175 | 626.525 | — | 786.352 | — |
| 200 | — | — | — | — |

图 6 为对应的收敛曲线图.

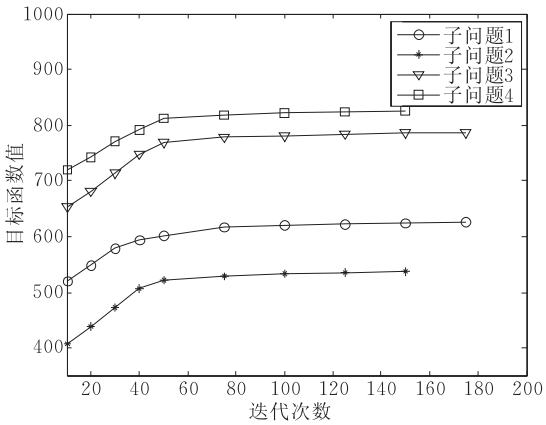



图 6 $P(50,50)$迭代收敛曲线图

通过上述 3 组数据可以看出,不同规模、不同输入的问题,在有限次迭代后均得到了最优解.同一规模的问题虽然迭代次数略有差别,但是总体相差不大.

此外为了考察更大规模问题下算法的收敛性,分别构造了小规模、中规模、大规模 3 类输入参数,比较求解时的迭代次数,实验结果如表 5 所示.

**表 5 不同规模问题下的迭代次数**

| 问题规模 | 作业数 | 物理机数 | 迭代次数 |
|---|---|---|---|
| 小规模 | 50 | 50 | 163 |
| 中规模 | 200 | 100 | 487 |
| 大规模 | 1000 | 200 | 935 |

由表 5 数据可见,当问题规模增加时,迭代次数仍然在可接受范围内.图 7 给出了问题规模和迭代次数的关系曲面图.

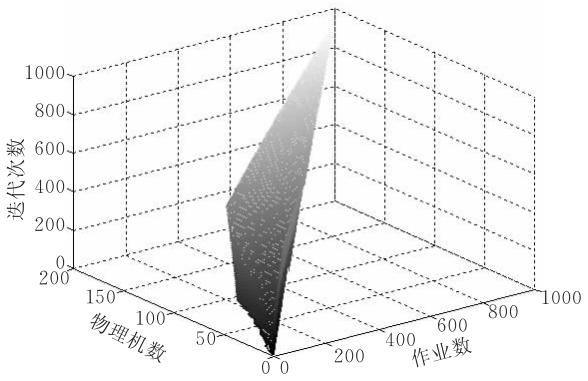



图 7 问题规模和迭代次数曲面图

通过理论证明和实验数据可得出,CUM 简化次梯度算法属于线性时间算法,对于规模为 $m$ 个作业、$n$ 个物理机的输入问题 $P(m,n)$,通过有限 $k$ 次迭代可获得最优解,算法时间复杂度为 $O(mn)$.

### 5.2 应用场景

CUM 模型可应用于规模较大的云环境中,图 8 给出了其应用模式.

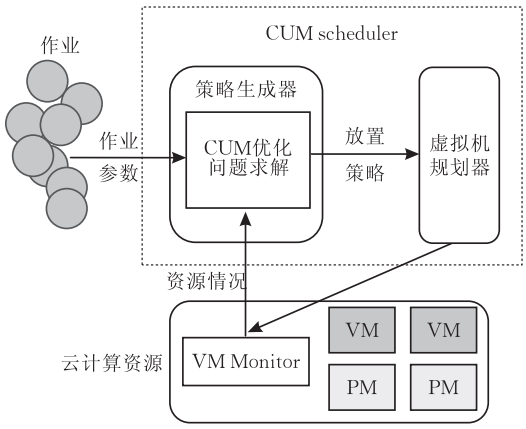



图 8 CUM 在云中的应用场景

图 8 中,CUM 全局调度管理器(CUM scheduler)实时接收虚拟机监控器(VM Monitor)发来的计算资源使用情况,根据作业、剩余资源确定效用最大化的虚拟机放置策略,创建虚拟机,将作业分配到虚拟机上执行.

与传统的集群、网格计算环境中采用的作业调度管理器(scheduler)相比,CUM scheduler 首先确定虚拟机放置策略,即多少物理计算资源实例化成一个虚拟机,然后将作业分配到虚拟机上执行.

在真实云环境中,可在 CUM scheduler 前端增加作业预处理模块,通过用户竞价、用户选择偏好来划分作业层次,由高至低排列,设置高低不同的愿意提供的支付值 $w_i$.经过分级后的作业提交给 CUM scheduler,进行调度.

CUM 模型以效用最大化作为优化目标,这种优化方式在计算机网络带宽分配中取得了很好的效果.计算机网络承载的业务往往具有如下特点:某一时刻往往很多业务同时到达请求服务,业务对服务质量的要求差距大,高峰期网络资源基本满负荷.因此当云计算不断发展,用户对计算资源的需求如同对网络需求一样普遍时,CUM 模型的调度优势更为明显.

## 6 结 论

本文提出的云资源 CUM 调度模型,和以往 0-1 整数规划的调度方法不同,利用计算机网络中 NUM 模型思路,实现了对物理机计算资源的更高层次分配:以效用最大为调度目标,一台或多台物理机资源如同网络链路的带宽一样,通过虚拟化技术分配给一个或多个作业.此外给出了该模型的优化算法,通过简化的次梯度算法求解该模型的拉格朗日对偶问题.通过模拟实验表明,算法具有可行性和

较好的收敛性.

下一步将探索不同的效用函数对虚拟机资源调度效益的影响.此外,将云计算资源从单纯的 CPU 计算能力度量方式扩展成多种资源(包括内存、I/O、磁盘存储等)共同约束,即将云资源的描述从一维发展到多维.

## 参 考 文 献


[1] Buyya Rajkumar, Yeo Chee Shin, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 2009, 25(6): 599-616

[2] Ibarra O, Kim C. Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM, 1977, 77(2): 280-289

[3] Duan Rubing, Prodan Radu, Fahringer Thomas. Performance and cost optimization for multiple large-scale grid workflow applications//Proceedings of the 2007 ACM/IEEE Conference on Supercomputing. Reno, Nevada, USA, 2007: 110-121

[4] Nascimento Aline P, Boeres Cristina, Rebello Vinod E F. Dynamic self-scheduling for parallel applications with task dependencies//Proceedings of the 6th International Workshop on Middleware for Grid Computing (MGC'08). Belgium, 2008: 1-6

[5] Atakan D, Fusun O. Genetic algorithm based scheduling of meta-tasks with stochastic execution times in heterogeneous computing systems. Cluster Computing, 2003, 7(2): 177-190

[6] Buyya R, Murshed M, Abramson D, Venugopal S. Scheduling parameter sweep applications on global grids: A deadline and budget constrained cost time optimization algorithm. Software-Practice and Experiences, 2005, 35(5): 491-512

[7] Kumar Subodha, Dutta Kaushik et al. Maximizing business value by optimal assignment of jobs to resources in grid computing. European Journal of Operational Research, 2009, 194(3): 856-872

[8] Yang J, Khokhar A, Sheikh S, Ghafoor A. Estimating execution time for parallel tasks in heterogeneous processing (HP) environment//Proceedings of the Heterogeneous Computing Workshop. Cancun, 1994: 23-28

[9] Beltrame G, Brandolese C, Fornaciari W, Salice F, Sciuto D, Trianni V. Dynamic modeling of inter-instruction effects for execution time estimation//Proceedings of the 14th International Symposium on System Synthesis. Canada, 2001: 136-141

[10] Li Qiang, Hao Qin-Fen et al. Adaptive management and multi-objective optimization for virtual machine placement in cloud computing. Chinese Journal of Computers, 2011, 34(12): 2253-2264(in Chinese)
(李强, 郝沁汾等. 云计算中虚拟机放置的自适应管理与多目标优化. 计算机学报, 2011, 34(12): 2253-2264)

[11] Kelly F P, Maulloo A K, Tan D K H. Rate control for communication networks: Shadow prices, proportional fairness and stability. Journal of the Operational Research Society, 1998, 49(3): 237-252

[12] Chiang M, Low S H, Calderbank A R, Doyle J C. Layering as optimization decomposition: A mathematical theory of network architectures. Proceedings of the IEEE, 2007, 95(1): 255-312

[13] Chiang M. Balancing transport and physical layer in wireless multihop networks: Jointly optimal congestion control and power control. IEEE Journal Selected Areas, 2005, 23(1): 104-116

[14] Pathan Mukaddim, Broberg James, Buyya Rajkumar. Maximizing utility for content delivery clouds//Vossen Gottfried, Long Darrell D E, Yu Jeffrey Xu eds. Web Information Systems Engineering(WISE 2009). Lecture Notes in Computer Science 5802. Berlin: Springer, 2009: 13-28

[15] Wuhib F, Stadler R, Spreitzer M. Gossip-based resource management for cloud environments//Proceedings of the International Conference on Network and Service Management (CNSM). Niagara Falls, Canada, 2010: 1-8

[16] Li Shi-Yong, Yang Dong, Qin Ya-Juan, Zhang Hong-Ke. Network cross-layer mapping based in utility maximization. Journal of Software, 2011, 22(8): 1855-1871(in Chinese)
(李世勇, 杨冬, 秦雅娟, 张宏科. 基于效用最大化的网络跨层映射. 软件学报, 2011, 22(8): 1855-1871)

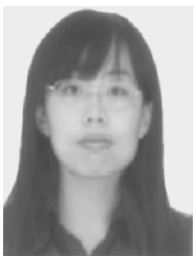
[17] Fisher M L. The lagrangean relaxation method for solving integer programming problems. Management Science, 1981, 27(1): 1-18

[18] Shapiro J F. Generalized lagrange multipliers in integer programming. Operations Research, 1971, 19(1): 68-76

[19] Held M, Wolfe P, Crowder H P. Validation of subgradient optimization. Mathematical Programming, 1974, 6(1): 62-88

[20] Geoffrion A M. Lagrangean relaxation for integer programming. Mathematical Programming Study, 1974, 2(2): 82-114

[21] Shor Naum Z. Nondifferentiable Optimization and Polynomial Problems. Boston: Boston Kluwer, 1998


**SHI Xue-Lin**, born in 1977, Ph.D., assistant researcher. Her main research interests include cloud computing, network optimization and scheduling.

**XU Ke**, born in 1974, Ph.D., professor. His main research interests include next generation Internet (NGI), switching routing structure, P2P network, Overlay network and Internet of Things.

**Background**

With development of Information and Communication Technology, computing will one day be the 5th utility (after water, electricity, gas, telephony). Computing resources are always distributed dispersedly, which connected with networks. How to provide transparent computing services for users over such heterogeneous environment is a key problem. To deal with it a number of computing paradigm has been proposed: cluster computing, Grid Computing, and more recently cloud computing.

Clouds often deal with large amount of resources including processors, memories, storages, visualization equipment, software, and so on. To assure performance efficiency and economic effectiveness of such huge infrastructure, resource scheduling is a fundamental challenge and is critical. More and more market-based resource management strategies have been brought out. Comparing with market-based cloud scheduling, traditional scheduling policies are mostly heuristic algorithms for scheduling $n$ independent tasks on $m$ processors in early finishing time. In wide area, heterogeneous and non autonomous cloud environments, economic business objective also became crucial for the success of the scheduling. Cloud scheduling is no more simple processors scheduling, but need taking price, budget, penalty cost into accounts.

This paper gave a cloud scheduling model, which applied approach of Network Utility Maximization into computing resource scheduling and aimed to maximize the utility of cloud. The proposed solution could effectively allocate resources and provide optimal virtual machine placement policy in a huge cloud.

This work is partly supported by the China National Science and Technology Ministry (No. 2011BAK08B05-02), the National Basic Research Program (973 Program) of China (Nos. 2009CB320501, 2012CB315803), the National Natural Science Foundation of China (Nos. 61170292, 60970104) and the National Science and Technology Major Project (No. 2012ZX03005001001).