中国科学: 信息科学

SCIENTIA SINICA Informationis

论文

一种异构计算系统中带任务包的能量感知利润最大化在线算法

张庆辉[1], 李伟东[2], 张学杰[1*]

1. 云南大学信息学院, 昆明650091
2. 云南大学数学与统计学院, 昆明650091
* 通信作者. E-mail: xjzhang@ynu.edu.cn

中国国家自然科学基金（批准号：12071417,61762091,62062065），云南大学第十三届研究生科研创新项目（批准号：2021Z079）以及云南省教育厅科研基金(批准号：2022J0002)

摘要 高性能计算中心的任务调度一直是一个备受关注的研究课题。在本文中，我们考虑一个拥有异构机器的高性能计算中心。用户提交的并不是单个任务而是一个包含多个同类型任务的任务包。我们以高性能中心管理者最大化其单位时间收益为目标，建立了考虑任务包的能量感知利润最大化问题模型，并为之设计了一种高效的在线算法。该算法的运行时间与用户数和机器类型数相关。在最后的实验中，我们通过与另外两种常用算法进行对比，本文提出的在线算法能够在多项式时间内，得到目标值拟最优的调度方案。

关键词 作业调度, 高性能计算, 负载均衡, 任务包

1 引言

随着数据中心能耗的快速增长，高性能计算系统中关注于能效的任务调度越来越重要。最近学界提出了一种名为任务包（Bag-of-tasks）的静态调度模型 [1]。与以往的经典调度模型相比，该台机器上的固定执行时间（Estimated time to compute）取决于任务类型与机器类型。根据这个新的模型，最终我们将要确定每种类型机器上执行的不同类型任务的数量。由于高性能计算中心的机器成百上千，并且需要执行的任务书目也无法估计。如果我们要单独地为每一个任务决策到哪一台机器进行执行，那将要耗费难以接受的时间来进行决策。所幸机器和任务的类型数量是很有限的。这使得设计一种更高效的算法来得到拟最优调度成为可能 [5,6]。

经典的能量感知调度模型旨在最小化任务包所消耗的能量或最大完工时间(Makespan)。然而，对于高性能计算中心运营商而言，将每个单位时间的运行利润最大化会更有价值，其中利润等于用户为一个任务包支付的价格减去执行该任务包消耗的电力成本。通过综合考虑能源成本和最大完工时间，最求单位时间利润最大化的目标。我们称之为考虑任务包的能量感知利润最大化问题（Energy-Aware Profit Maximizing, 简写为EAPM）。Tarplee等人 [4]以此为目标提出了一种新的高性能计算系统调度模型，该模型具有两个重要特征：(a) 它们通常由不同类型的机器组成；(b) 需要执行的任务数总量很多，但任务类型有限。通过使用一种新颖的基于线性规划(Linear programming, 简写为LP)的舍入算

法，Tarplee等人 [4]在每台机器将要执行很多相同类型任务的条件下，设计了一个能够得到接近最优调度的高效算法。

在该文中，作者用一种机器的完工时间下界来代替最大完工时间，其中最大完工时间定义为所有机器的最大完工时间。因此，他们提出的数学模型是不准确的。在Tarplee等人所提出的基于LP的舍入步骤过程中，能耗成本可能会增加，这可以通过使用基于匹配的舍入技术来改善。在实际应用中，算法的执行时间也会随着任务数量的增加而增加。此外，该篇文章也并未分析他们所提出的基于LP的舍入算发的近似比。

在本文中，我们为这个基于新模型的问题提出了一个多项式时间算法。本文的主要贡献如下:

(1).为EAPM问题建立了一个准确的数学模型；

(2).提出了一个在最坏情况下也能多项式时间运行的算法，该算法能给出一个准确的可行解；

(3).通过对比实验，说明了本文提出的算法的优越性。

本文剩余内容结构如下。第二节中我们总结了一些与本文相关的研究。第三节，我们为EAPM问题构造了与之对应的线性规划模型。在第四节中，我们详细介绍了我们提出的基于任务类型的在线算法。在第五节中，我们进行了对比实验。最后，我们对本文的工作做出了总结，并展望了未来的工作。

2 相关工作

最近十几年有大量关于异构计算系统中任务分配的调度模型的研究。Braun等人 [8]比较了十一种静态启发式算法，他们将一类互相独立的任务映射到异构分布式计算系统，来最小化最大完工时间。Diaz，Pecero 和Bouvry [9]评估了三个高度异构分布式计算系统的调度启发式算法，目标是最小化最大完工时间和流程时间。假设用户提交一组独立的任务(任务包)，Friese 等人 [1,2]提出了一种改进的多目标遗传算法生成多个不同的调度方案，以此平衡能源消耗和最大完工时间(或系统的经济效益)之间的得失。Friese 等人 [3]创建了一个工具，该工具能帮助系统管理员对系统性能和系统能量分配进行权衡。Oxley 等 [7]提出并分析了几种启发式方法，以此来解决联合考虑了能源约束和截止时间约束的能源感知资源分配问题。对于确定每种类型的任务要分配多少给各个不同类型的机器的调度模型，Tarplee 等 [5]提出了一种基于线性规划的资源分配算法，该算法能高效地给出最小化最大完工时间的调度方案。Tarplee 等 [6]之后设计了一种基于线性规划的舍入方法来生成一组高质量的调度方案，这些调度方案能够很好地在最大完工时间和能耗之间权衡。

在云计算环境中，云资源管理是云供应商的一个重要内容。Ebrahimirad，Goudarzi 和Rajabi [10]为虚拟数据中心中的优先级受限并行虚拟机(VM)提出了两种调度算法，以此来最小化能源消耗。Tchana 等 [11]提出了一个将软件整合到虚拟机上的解决方案，以减少私有云的功耗和公共云中收费的虚拟机数量。Hsu 等 [12]提出了一种能量感知任务整合技术，以最大限度地减少云中心的能量消耗。Khemka 等 [13]为能源受限的环境设计了四种能量感知的资源分配启发式方法，目的是使系统获得的总效用最大化。Khemka 等 [14]设计了几种启发式方法，以最大限度地提高完成任务所能获得的总效用，其中任务动态到达，调度程序将任务映射到机器进行执行。Mashayekhy 等 [15]提出了一个提高MapReduce 应用程序能源效率的框架，它能在给定最后期限的限制条件下最大限度地减少能源消耗。更相关的结果可以在最近的能源效率资源调度综述中找到 [16~19]。

3 在线调度模型

在一个异构计算机系统中包含了m种不同的机器类型和n种不同类型的用户。用户i提交的任务包中的任务相互独立，数量为a_i [8]，执行这类任务能产生的收益为p_i。以下两个变量是异构计算机系统调度问题中经常用到的概念：$\mathbf{ETC} = (ETC_{ij})$是一个$n \times m$维矩阵，其中$ETC_{ij}$是用户$i$的任务在机器$j$上执行所需的固定执行时间（Estimated time to compute，简写为ETC）；$\mathbf{APC} = (APC_{ij})$同样是一个$n \times m$维矩阵，其中$APC_{ij}$是用户$i$的任务在机器$j$上执行所需的平均功率消耗（Average power consumption，简写为APC） [5] 。我们用x_{ij}来表示用户i的任务分配给机器j执行的任务数。对于一个可行解$\mathbf{x} = (x_{ij})$，机器j的负载可以定义为：

$$L_j = \sum_{i=1}^{n} ETC_{ij} x_{ij}. \tag{1}$$

那么所有机器的最大完工时间$MS(\mathbf{x})$，定义为：

$$MS(\mathbf{x}) = \max_j L_j. \tag{2}$$

相应的，n个用户的能量消耗为：

$$E(\mathbf{x}) = \sum_{j=1}^{m} \sum_{i=1}^{n} x_{ij} APC_{ij} ETC_{ij}. \tag{3}$$

用c表示每个单位能耗的成本。那么EAPM问题可以形式化的用以下非线性整数规划表示：

$$Maximize_{\mathbf{x}} \quad \frac{\sum_{i=1}^{n} p_i - cE(\mathbf{x})}{MS(\mathbf{x})} \tag{4}$$

$$\text{subject to:} \quad \sum_{j=1}^{m} x_{ij} = a_i, \forall i = 1, 2, ..., n \tag{5}$$

$$\sum_{i=1}^{n} x_{ij} ETC_{ij} \leqslant MS(\mathbf{x}), \forall j = 1, 2, \ldots, m \tag{6}$$

$$x_{ij} \in \mathbb{Z}_{\geqslant 0}, \forall i, j. \tag{7}$$

目标函数(4)要最大化单位时间的收益，其中$\mathbf{x}$是决策向量。约束(5)确保了每一个用户的每一个任务都被分配给某个机器。由于最大化单位时间收益的目标等价于最小化最大完工时间，约束(6)确保了$MS(\mathbf{x})$是所有机器的最大完工时间。

然而在实际场景中，当某个用户到达时就需要在机器不知道未到达用户的信息的情况下分配该用户的所有任务。因此研究考虑任务包的EAPM问题的在线算法是很有必要的。该在线算法考虑在用户i的任务会在用户$i+1$到达之前就被分配，此处$i = 1, 2, ..., n-1$。不失一般性，我们假设用户i的所有任务会在用户$i+1$到达之前就被分配给某个机器。更重要的一点是，当用户i提交的的任务数a_i非常大时，我们不能逐个分配这些任务。因此为考虑任务包的EAPM问题设计一个高效的在线算法是很有必要的。

4 在线算法

在本节中，我们为考虑任务包的EAPM问题提出了一个高效的在线算法。对于每一个用户i，我么用L_j^i和E^i表示分配完前i个用户的任务之后机器j的负载和系统的总能耗。初始情况下$L_j^0 = 0, j = 1, ..., m$，$E^0 = 0$。根据定义，对于任意$i = 1, 2, ..., n$，我们能得到一下关系：

$$L_j^i = \sum_{k=1}^{i} x_{kj} ETC_{kj}, \text{ and } E^i = \sum_{k=1}^{i} \sum_{j=1}^{m} x_{kj} APC_{kj} ETC_{kj}. \tag{8}$$

当用户i到达时，我们将确定x_{ij}的值并让$\sum_{j=1}^{m} x_{ij} = a_i$，此时的目标值为：

$$\frac{\sum_{k=1}^{i} p_k - cE^{i-1} - c\sum_{j=1}^{m} x_{ij} APC_{ij} ETC_{ij}}{MS^i} \tag{9}$$

其中

$$MS^i = \max_j L_j^i, \text{ and } L_j^i = L_j^{i-1} + x_{ij} ETC_{ij}, \forall i, j. \tag{10}$$

该问题可以形式化的表示为以下整数规划（integer program, IP）：

$$\begin{cases} \text{Maximize } \dfrac{\sum_{k=1}^{i} p_k - cE^{i-1} - c\sum_{j=1}^{m} x_{ij} APC_{ij} ETC_{ij}}{MS^i} \\ \displaystyle\sum_{j=1}^{m} x_{ij} = a_i \\ L_j^{i-1} + x_{ij} ETC_{ij} \leqslant MS^i \\ x_{ij} \in Z^+ \cup \{0\}, j = 1, \ldots, m. \end{cases} \tag{11}$$

其中第二个约束等价于：

$$x_{ij} \leqslant \lfloor \frac{MS^i - L_j^{i-1}}{ETC_{ij}} \rfloor. \tag{12}$$

为了便于操作，我们将服务于用户i的任务的机器按照$APC_{ij} ETC_{ij}$降序排序。不失一般性，假设

$$APC_{i1} ETC_{i1} \geqslant APC_{i2} ETC_{i2} \geqslant \cdots \geqslant APC_{im} ETC_{im}. \tag{13}$$

我们的算法是基于以下引理实现的。

引理1 存在一个最优解，该最优解符合以下情形：

$$x_{i1} = \cdots = x_{i(\tau-1)} = 0, \text{and } x_{ij} = \lfloor \frac{MS^i - L_j^{i-1}}{ETC_{ij}} \rfloor, j = \tau, \ldots, m,$$

其中$\tau \in \{1, ..., m\}$。

证明 假设我们已知整数规划(11)最优解中MS^i的值。那么规划(11)的目标函数便等价于最小化$\sum_{j=1}^{m} x_{ij} APC_{ij} ETC_{ij}$的值，其中$L_j^{i-1}$ 和ETC_{ij}是两个常量。很明显，在规划(11)约束限制条件下，为了让$\sum_{j=1}^{m} x_{ij} APC_{ij} ETC_{ij}$的值尽可能小，若$APC_{ij} ETC_{ij}$值小那么就让其对应的$x_{ij}$大，若$APC_{ij} ETC_{ij}$值大那么就让其对应的$x_{ij}$小。

对于规划(11)的一个最优解$(x_{i1}, x_{i2}, \ldots, x_{im})$，我们假设$\tau_1$是使得$x_{i\tau_1} > 0$成立的最小机器下标。如果存在一台机器$\tau_2$（$\geqslant \tau_1$），有$x_{i\tau_2} < \lfloor \frac{MS^i - L_{\tau_2}^{i-1}}{ETC_{i\tau_2}} \rfloor$，那么我们令

$$x'_{ij} = \begin{cases} x_{ij} - 1, \text{ if } j = \tau_1 \\ x_{ij} + 1, \text{ if } j = \tau_2 \\ x_{ij}, \text{ if } j \neq \tau_1, \tau_2. \end{cases} \tag{14}$$

很容易证明$(x'_{i1}, x'_{i2}, \ldots, x'_{im})$是规划(11)的一个可行解。由于(13)重排序使得$APC_{i\tau_1}ETC_{i\tau_1} \geqslant APC_{i\tau_2}ETC_{i\tau_2}$，该可行解的目标值会比$(x_{i1}, x_{i2}, \ldots, x_{im})$的目标值更小。对所有机器$j(\geqslant \tau$，$\tau$是使得$x_{i\tau} > 0$的最小机器下标)重复上面的过程，直到$x_{ij} = \lfloor \frac{MS^i - L_j^{i-1}}{ETC_{ij}} \rfloor$。这预示着，我们最终能找到一个最优解满足

$$x_{i1} = \cdots = x_{i(\tau-1)} = 0, \text{and } x_{ij} = \lfloor \frac{MS^i - L_j^{i-1}}{ETC_{ij}} \rfloor, \text{ for } j = \tau + 1, \ldots, m.$$

因此，该引理成立。

那么对于每个$\tau = 1, ..., m$，我们只需考虑部分的决策变量$x_{ij}(j = \tau, ..., m)$。此时我们想要得到用户i到达时的所有x_{ij}值，以及MS^i的值。我们能通过以下方程组得到我们想要的结果：

$$\begin{aligned} &\sum_{j=\tau}^{m} x_{ij} = a_i; \\ &x_{ij} = \frac{MS^i - L_j^{i-1}}{ETC_{ij}}, j = \tau, \ldots, M \quad . \end{aligned} \tag{15}$$

注意到该线性方程组总共有$m - \tau + 2$个等式和$m - \tau + 2$个未知数，未知数为MS^i和$x_{ij}(j = \tau, ..., m)$。因此，该线性方程组能在多项式时间内得到解。对于每一个$\tau = 1, ..., m$，我们都能得到一组x_{ij}的值。比较这m组可行解的目标值，我们便能找到最优的那个。之后，我们从$j = m$到1来分配任务，将$\lceil x_{ij} \rceil$个用户i的任务分配给机器j，直到所有任务都被分配。

以下是我们算法的伪代码:

算法 1 Online

输入： $m, n, \mathbf{ETC}, \mathbf{APC}$,User arrival sequence;

for $i = 1$ to n **do**
  Relabel the indices of tasks such that $APC_{i1}ETC_{i1} \geqslant \ldots \geqslant APC_{iM}ETC_{iM}$;
  for $\tau = 1$ to m **do**
    Solve (15) to find a solution x_{ij}^{τ};
    Comparing these M solution to find the best solution x_{ij} such that $\frac{p - cE^{i-1} - c\sum_{j=1}^{M} x_{ij}APC_{ij}ETC_{ij}}{MS^i}$ is maximized.
  end for
  for $j = 1$ to m **do**
    Assign $\lceil x_{ij} \rceil$ tasks of type i to machines of type j, until all tasks are assigned;
    Update the L_j of type j.
  end for
end for

输出： x

表 1　目标值随着γ从1.05增长到1.5的变化

Table 1　Objective values, with γ varied from 1.05 to 1.5.

$\gamma =$	1.05	1.1	1.15	1.2	1.25	1.3	1.35	1.4	1.45	1.5
Online	22.23	45.26	70.47	99.5	130.69	162.65	190.94	224.4	255.72	297.67
Greedy	22.63	43.44	67.89	90.52	113.15	135.78	158.41	181.04	203.68	226.31
Average	-382.80	-347.77	-312.73	-277.7	-242.67	-207.63	-172.6	-137.56	-102.53	-67.5

5　实验评估

本节介绍具体的实验方法及实验结果。

5.1　实验环境

为了进行对比，除了本文提出的在线方法（Online），我们还使用了贪心算法(Greedy)和平均分配算法(Average)来解决EAPM问题。其中Greedy算法将会在一个用户到达时，将该用户的整个任务包分配给能使值$ETC\dot{A}PC$最小的那台机器，而Average将会把该用户的整包任务平均分配给每种机器。我们使用C++编程语言实现上述算法，并在以下硬件配置环境中进行了实验：CPU型号为Itel i7-10700，8核16线程2.9Ghz，16GB内存以及1TB硬盘容量。实验的部分数据源于 [20]中的9种机器类型以及5中任务类型，文章 [4]在其基础上进行了扩充，本文实验最终包括9中机器类型和30种任务类型。

不失一般性，在我们所有的实验中都都令$c = 1$。若不考虑最大完工时间，我们将E^i_{min}定义为用户i的能耗下界，$E^i_{min} = \min_j ETC_{ij}APC_{ij}$。此外，我们让$p_i = \gamma E^i_{min}$，那么$\gamma = p_i/E^i_{min}$便是一个能够影响用户$i$出价的参数。在实际工业活动中，高性能中心的管理者为了盈利，一般不会选择较小的γ（$\leqslant 1$）。根据文章[25]中的结论，当γ足够大时($\gamma > 1.5$)，寻求单位时间效益最大化时分配结果会更趋向于最小化最大完工时间。因此，在本节的实验中，我们仅考虑$\gamma \in (1, 1.5]$。

5.2　不同γ值对系统的影响

本文第一个实验是在上述9种机器类型和30种任务类型的基准上进行的。我们考虑30个用户按任意顺序提交了类型各不相同的任务包，其中对于用户i其任务包中的任务数$a_i \in [200, 1000]$。随着γ的变化，三种方法得到的目标值变化如表1所示。为了减小随机性带来的影响，我们对于每种γ取值都重复了100次实验并将结果取平均值。我们可以看到本文提出的Online方法在所有情况下都优于另外两种方法。当γ较小时，Online方法与Greedy方法得到的目标值差距并不大，但当γ逐渐增大后Online方法便体现了其优越性。这是由于Greedy方法在决策的过程中很容易陷入局部最优的困局。我们可以明显看到Average方法与另外两种方法得到的目标值有着较大差距。它虽然能使得系统的最大完工时间最小，但是由于并未考虑成本的原因，会使得最终的目标值变得很差。

5.3　不同用户数n对系统的影响

在本实验中，我们研究了用户数变化对系统最终目标值的影响，其中n逐渐从30增长到2000。为了更全面地评估n的增长带来的影响，我们分别在$\gamma = 1.2, 1.3 \text{and} 1.5$三种条件下对比了上述三种方法。同样的，用户$i$提交的任务包中的任务数$a_i \in [200, 1000]$。

从图1中我们能够明显观察到，当用户数较小时Online方法得到的目标值更大，当用户数不断增加，Online方法与Greedy方法得到的目标值几乎变得相等。这是由于当用户数n变得足够大时，总的任

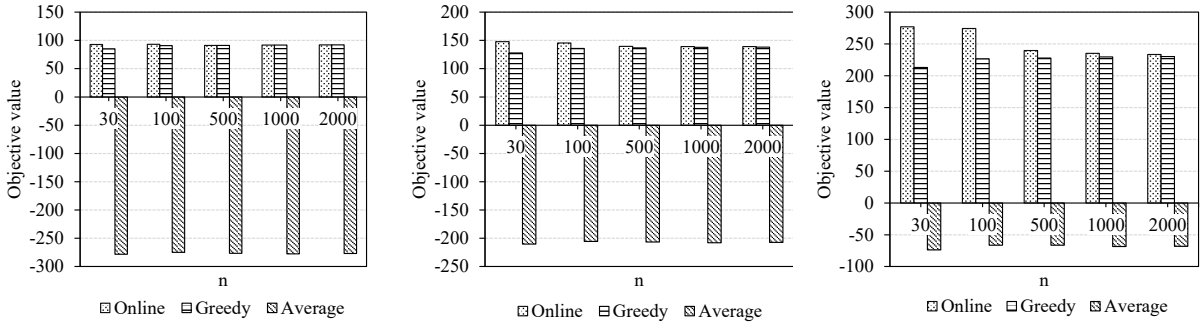

图 1　三种方法的目标值随着n增长的变化

Figure 1　Objective values of two methods with varying n.

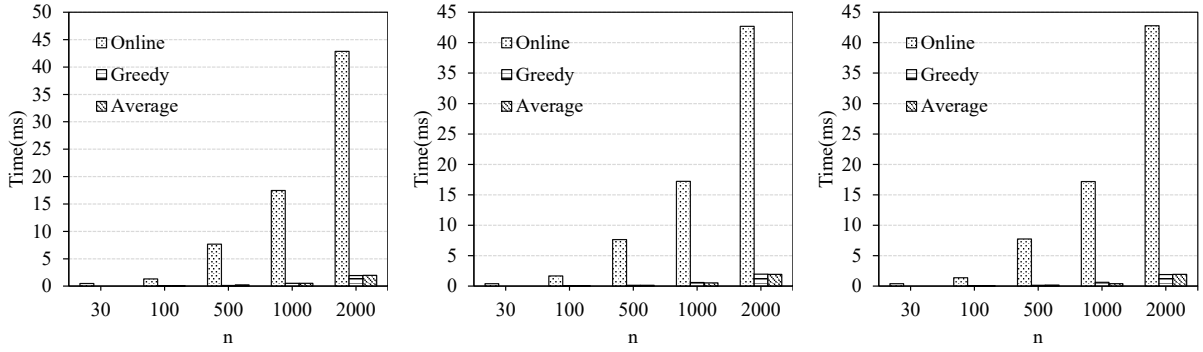

图 2　三种方法的执行时间随着n增长的变化

Figure 2　Execution time of two methods with varying n.

务数理所应当的也会增加，这会使得两种方法给出的调度方案产生的最大完工时间将会逐渐靠近最优调度所产生的最大完工时间。

很明显，解所有用户对应的方程组15的时间复杂度与机器类型数m与用户数n有关。同样的，另外两种方法的时间复杂度也与二者有关。为了评估三种在线算法的时间性能，我们在同样的实验条件下进行了比较。

在图2中我们可以看出三种方法的执行时间都在随着用户数的增加而增加。虽然Online方法的执行时间最长，但它得到的目标值也是最大的。并且Online方法执行时间的增长是线性的，即使在$n = 2000$时其执行时间也是毫秒级的。

5.4　用户到达顺序对系统的影响

在实际场景中，用户到达后所提交的任务数可能很大，也可能很小，这是无法预知的。为了评估用户到达顺序对系统的影响，我们进行了本实验。我们设置了用户数$n = 30$，他们的类型是随机的。如果用户i提交的任务包中的任务数$a_i > 500$，我们定义该用户提交的任务包为大任务包；若用户i提交的任务包中的任务数$a_i < 200$，我们定义该用户提交的任务包为小任务包。我们对比了四种到达顺序对应的目标值，“BaS”为前一半的用户提交大任务包，后一半用户提交小任务包；“SaB”为前一半的用户提交小任务包，后一半用户提交大任务包；“Rand”为提交大任务包与小任务包的用户到达是随机的；“Equal”为所有用户提交的任务包任务数相同，为400。

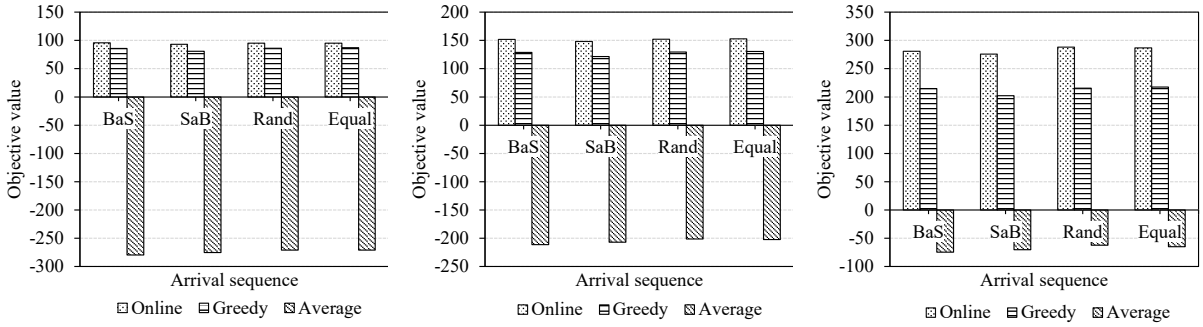

图 3　三种方法的执行时间随着n增长的变化

Figure 3　Execution time of two methods with varying n.

从图3我们可以看到，在所有情况下Online方法都能得到最好的目标值结果。此外，我们观察到用户到达的顺序并未对目标值造成明显的影响，只有当SaB情形时Online方法与Greedy得到的目标值会稍微降低。这是由于，先将小任务包分配之后，为了不引起最大完工时间的快速增长，大包任务到达时有较小可能被分配给成本更大的机器进行执行。同时，我们也能很明显观察到，当$\gamma = 1.5$时，Online方法得到的目标值更具优势。

6　总结与未来工作

在本文中，我们提出了一种名为Online方法，其执行时间依赖于用户数以及机器种类数。我们通过新颖的方法构造线性方程组得到最终的分配结果，并在最终通过实验说明了此方法能在很多情况下得到令人满意的结果。

值得一提的是，本文并未给出Online方法的近似比，这是将来值得进一步分析的研究内容。此外，基于机器学习的在线算法也是当下十分流行的方法。

参考文献

1　R. Friese, T. Brinks, C. Oliver, H. J. Siegel, and A. A. Maciejewski, “Analyzing the trade-offs between minimizing makespan and minimizing energy consumption in a heterogeneous resource allocation problem, ” in *The Second International Conference on Advanced Communications and Computation*, pp. 81-89, 2012.

2　R. Friese, B. Khemka, A.A. Maciejewski, H.J. Siegel, G.A. Koenig, S. Powers, M. Hilton, J. Rambharos, G. Okonski, and S. W. Poole, “An analysis framework for investigating the trade-offs between system performance and energy consumption in a heterogeneous computing environment”, in *IEEE 27th International Parallel and Distributed Processing Symposium Workshops*, pp. 19-30, 2013.

3　R. Friese, T. Brinks, C. Oliver, H.J. Siegel, A.A. Maciejewski, and S. Pasricha, “A machine-by-machine analysis of a bi-objective resource allocation problem”, in *International Conference on Parallel and Distributed Processing Technologies and Applications*, pp. 3-9, 2013.

4　K.M. Tarplee, A.A. Maciejewski, and H.J. Siegel, “Energy-aware profit maximizing scheduling algorithm for heterogeneous computing systems, ” in *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 595-603, 2014.

5　K. M. Tarplee, R. Friese, A. A. Maciejewski, and H. J. Siegel, “Scalable linear programming based resource allocation for makespan minimization in heterogeneous computing systems, ” *Journal of Parallel and Distributed Computing*, vol. 84, pp. 76-86, 2015.

6 K. M. Tarplee, R. Friese, A. A. Maciejewski, H. J. Siegel, and E. Chong,“ Energy and makespan tradeoffs in heterogeneous computing systems using efficient linear programming techniques, ” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1633-1646, 2016.

7 M.A. Oxley, S. Pasricha, A.A. Maciejewski, H.J. Siegel, J. Apodaca, D. Young, L. Briceno, J. Smith, S. Bahirat, B. Khemka, A. Ramirez, and Y. Zou, “ Makespan and energy robust stochastic static resource allocation of bags-of-tasks to a heterogeneous computing system, ” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2791-2805, 2015.

8 T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, “ A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, ” *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810-837, 2001.

9 C. O. Diaz, J. E. Pecero, and P. Bouvry, “ Scalable, low complexity, and fast greedy scheduling heuristics for highly heterogeneous distributed computing systems,” *The Journal of Supercomputing*, vol. 67, no. 3, pp. 837-853, 2014.

10 V. Ebrahimirad, M. Goudarzi, and A. Rajabi, “ Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers, ” *Journal of Grid Computing*, vol. 13, no. 2, pp. 233-253, 2015.

11 A. Tchana, N. D. Palma, I. Safieddine, and D. Hagimont, “Software consolidation as an efficient energy and cost saving solution,” *Future Generation Computer Systems*, vol. 58, pp. 1-12, 2016.

12 C. H. Hsu, K. D. Slagter, S. C. Chen, and Y. C. Chung, “Optimizing energy consumption with task consolidation in clouds, ” *Information Sciences*, vol. 258, no. 10, pp. 452-462, 2014.

13 B. Khemka, R. Friese, S. Pasricha, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, R. Rambharos, and S. Poole, “Utility maximizing dynamic resource management in an over subscribed energy-constrained heterogeneous computing system,” *Sustainable Computing: Informatics and Systems*, vol. 5, pp. 14-30, 2015.

14 B. Khemka, R. Friese, L. D. Briceno, H. J. Siegel, A. A. Maciejewski, G. A. Koenig, C. Groer, G. Okonski, M. M. Hilton, R. Rambharos, and S. Poole, “ Utility functions and resource management in an oversubscribed heterogeneous computing environment, ” *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 2394-2407, 2015.

15 L. Mashayekhy, M.M. Nejad, D. Grosu, Q. Zhang, and W. Shi, “ Energy-aware scheduling of mapreduce jobs for big data applications, ” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2720-2733, 2015.

16 M. Dayarathna, Y. Wen, and R. Fa, “ Data center energy consumption modeling: a survey, ” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732-794 2016.

17 A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P.h Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, S. U. Khan, and A. Zomaya, “A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems,” *Computing*, vol. 98, pp. 751-774, 2016.

18 T. Kaur, and I. Chana, “Energy efficiency techniques in cloud computing: a survey and taxonomy, ” *ACM Computing Surveys*, vol. 48, no. 2, Article No. 22, 2015.

19 A.-C. Orgerie, M. D. de Assuncao, and L. Lefevre, “A survey on techniques for improving the energy efficiency of large-scale distributed systems, ” *ACM Computing Surveys*, vol. 46, no. 4, Article No. 47, 2014.

20 (2013, May) Intel core i7 3770k power consumption, thermal. [Online]. Available: http://openbenchmarking.org/result/ 1204229-SU-CPUMONITO81