异构计算系统中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 云南 昆明 650091；2.云南大学 数学与统计学院 云南 昆明 650091)

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。本文以高性能中心管理者单位时间收益最大化为目标，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。每到达一个用户，该在线算法能够通过系统当前状态构造多个线性方程组。能让利润最大的线性方程组的解，就是对当前用户提交的任务包的分配策略。该算法的运行时间复杂度为$O(nm^4)$。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

中图分类号：TP302 **文献标志码：**A **文章编号：**1671-6841（2022）00-0000-00

DOI：10.13705/j.issn.1671-6841. 2022250

An online algorithm for the energy-aware profit maximizing problem in heterogeneous computing system

ZHANG Qinghui[1]， LI Weidong[2]， ZHANG Xuejie[1]

(1.School of Information Science and Engineering, Yunan University, Kunming 650091, China; 2.School of Mathematics and Statistics, Yunnan University, Kunming 650091, China)

Abstract: Task scheduling in high-performance computing center is always a hot research topic. In this article, we consider a high-performance computing (HPC) center with heterogeneous machines. Instead of a single task, a user submits a bag-of-tasks that contains multiple tasks of the same type. With the goal of maximizing unit time profit for HPC managers, we model an energy-aware profit maximization problem with bag-of-tasks and design an efficient online algorithm for it. For each user, the online algorithm can construct multiple sets of system of linear equations based on the system's current state. The most profitable system of linear equations is the allocation strategy for the task packages submitted by the current user. The running time of the algorithm is $O(nm^4)$. In the experiments, we compared the proposed algorithm with the other two common algorithms and found that the proposed online algorithm can obtain a quasi-optimal scheduling scheme in polynomial time.

Key words: Task scheduling; HPC; Load balancing; Bag-of-tasks.

收稿日期：2022-**-**

基金项目：国家自然科学基金资助项目(12071417, 61762091, 62062065)，云南大学第十三届研究生科研创新项目(2021Z079)。

作者简介：张庆辉(1998—)，男，硕士研究生，主要从事边缘计算中资源分配问题研究，E-mail:zhangqinghui@mail.ynu.edu.cn。

通信作者：张学杰（1965—），男，教授，博士，主要从事高性能计算以及云计算等领域的研究，E-mail:xjzhang@ynu.edu.cn。

0 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。通过这一概念，我们就能把高性能计算中心成千上万的任务分类，而任务的类型数量往往十分有限。对于机器而言，它们的类型数量同样很有限。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。而考虑任务包的调度模型的问题规模会被很好地限制，这也使得设计一种高效的算法来得到拟最优调度成为可能[2,3]。

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带来更多的经济收益，其中利润等于用户为一个任务包支付的费用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，实现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，该模型具有机器类型数与任务数都十分有限的特征[3]。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，设计了一个能够得到接近最优调度的高效算法。

在该文中，Tarplee等人用一个最大完工时间下界代替了原本的最大完工时间。最大完工时间下界是将任务平均分配给所有机器时的完工时间。这个下界与真实的最大完工时间是有一定差距的。因此，他们提出的数学模型是不准确的。而且在基于LP的舍入步骤过程中，能耗成本可能会增加。这即便可以通过使用基于匹配的舍入技术来改善，但执行时间会随着问题规模的扩大而急剧上升，这导致该算法在大型数据中心无法有很好的性能表现。

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，该模型能精确计算每到达一个用户并分配其任务包后系统的最大完工时间；(2) 提出了一个时间复杂度为$O(nm^4)$的算法，该算法通过在每个用户到达时构造多个线性方程组。这些线性方程组中最好的结果，便是当前针对该用户任务包的调度结果。(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。

1 相关工作

最近几十年有大量关于异构计算系统中任务调度模型的研究。Braun 等人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[4]。Bingfei Dai 等在包含两台平行机的系统中，设计了一种半在线算法，该算法能很好地限制机器的最大完工时间[5]。针对考虑任务包的调度模型，Tarplee 等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[2]。胡逸骉针对面向异构计算集群的任务调度和能量消耗问题，提出一种面向异构计算系统的能量感知任务调度算法[6]。

Friese 等人针对任务包这种情形提出了一种改进的多目标遗传算法来生成多个不同的调度方案，该算法能很好地平衡能源消耗和最大完工时间之间的得失[1,7]。他们还创建了一个工具，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[8]。张晓璐等设计了一个整数线性双目标优化模型，并提出了两阶段的启发式分配算法以找到高质量的可行解决方案[9]。除此之外，追求能量感知利润最大化的目标也能很好地平衡最大

完工时间和能耗。李伟东等针对考虑任务包的能量感知利润最大化问题，设计了一个最坏情况近似比接近 2 的近似算法[10]。随后李伟东等又提出了一个针对该问题的多项式时间近似算法，该算法同样能在某些情况下有接近 2 的近似比效果[11]。

在云计算环境中，云资源管理同样是云供应商的一个重要内容。Khemka 等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[12]。姜春茂等人提出面向实时云任务的细粒度任务合并调度算法，他们提出的算法在满足用户 SLA 的前提下,能够有效降低云能耗[13]。张骥先等通过拍卖机制对云计算虚拟资源进行分配和定价，以提升资源提供商的社会福利[14]。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[15–18]。

2 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 [4]，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）[3]。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 问题可以形式化地用以下非线性整数规划表示：

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

$$\text{s.t. } \sum_{j=1}^{m} x_{ij}=a_i, \forall i=1,2,...,n \tag{5}$$

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

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

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，$i=1,2,...,n-1$。更重要的一点是，当用户 i 提交的任务数非常大时，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。

3 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 $i=1,2,...,n$，我们能得到以下关系：

$$L_j^i=\sum_{k=1}^{i} x_{kj}ETC_{kj}, \tag{8}$$

$$E^i=\sum_{k=1}^{i}\sum_{j=1}^{m} x_{kj}APC_{kj}ETC_{kj}. \tag{9}$$

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：

$$\frac{\sum_{k=1}^{i} p_k - cE^{i-1} - c\sum_{j=1}^{m} x_{ij} APC_{ij} ETC_{ij}}{MS^i} \quad (10)$$

其中：

$$MS^i = \max_j L_j^i, \quad (11)$$

$$L_j^i = L_j^{i-1} + x_{ij} ETC_{ij}, \forall i, j. \quad (12)$$

令 $C_i = c\sum_{j=1}^{m} x_{ij} APC_{ij} ETC_{ij}$，那么该问题可以形式化地表示为以下整数规划：

$$\text{Max} \quad (\sum_{k=1}^{i} p_k - cE^{i-1} - C_i) \Big/ MS^i \quad (13)$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} = a_i \quad (14)$$

$$L_j^{i-1} + x_{ij} ETC_{ij} \le MS^i \quad (15)$$

$$x_{ij} \in Z^+ \cup \{0\}, j = 1, \ldots, m. \quad (16)$$

其中约束(15)等价于：

$$x_{ij} \le \left\lfloor \frac{MS^i - L_j^{i-1}}{ETC_{ij}} \right\rfloor. \quad (17)$$

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

$$APC_{i1}ETC_{i1} \ge APC_{i2}ETC_{i2} \ge \cdots \ge APC_{im}ETC_{im}. \quad (18)$$

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：

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

其中 $\tau \in \{1, ..., m\}$。

证明 假设我们已知整数规划(13)最优解中 MS^i 的值。那么规划(13)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，在规约(13)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 (13) 的一个最优解 $(x_{i1}, x_{i2}, \ldots, x_{im})$，我们假设 τ_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 τ_2（$\ge \tau_1$），有 $x_{i\tau_2} < \left\lfloor \left(MS^i - L_{\tau_2}^{i-1}\right) \Big/ ETC_{i\tau_2} \right\rfloor$，那么我们令

$$x'_{ij} = \begin{cases} x_{ij} - 1, \text{ if } j = \tau_1, \\ x_{ij} + 1, \text{ if } j = \tau_2, \\ x_{ij}, \text{ if } j \ne \tau_1, \tau_2. \end{cases} \quad (19)$$

很容易证明 $(x'_{i1}, x'_{i2}, \ldots, x'_{im})$ 是规划(13)的一个可行解。由于 (18) 重排序使得 $APC_{i\tau_1}ETC_{i\tau_1} \ge APC_{i\tau_2}ETC_{i\tau_2}$，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 （$\ge \tau$，τ 是使得 $x_{i\tau} > 0$ 的最小机器下标）重复上面的过程，直到 $x_{ij} = \left\lfloor \left(MS^i - L_j^{i-1}\right) \Big/ ETC_{ij} \right\rfloor$。这预示着，我们最终能找到一个最优解满足

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

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：

$$\begin{cases} \sum_{j=\tau}^{m} x_{ij} = a_i, \\ x_{ij} = \dfrac{MS^i - L_j^{i-1}}{ETC_{ij}}, j = \tau, \ldots, M. \end{cases} \quad (20)$$

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，我们能在多项式时间内对该线性方程组进行求解。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，便能得到当前的最优调度。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。

输入：m，n，**ETC**，**APC** 以及用户到达序列

输出：**x**

1）for i 从 1 到 n

2） 重索引下标使得 $APC_{i1}ETC_{i1} \geq ... \geq APC_{iM}ETC_{iM}$

3） for τ 从 1 到 m

4） 解方程组(20)得到一个解 x_{ij}^{τ}

5） 比较这 m 个解，找到使得目标值(13)最大的 x_{ij}

6） for j 从 1 到 m

7） 将 $\lceil x_{ij} \rceil$ 个用户 i 的任务分配给机器 j，知道所有任务都被分配

8） 更新机器 j 对应的 L_j。

定理 1 上述算法的时间复杂度为 $O(nm^4)$。

证明 该算法中主要耗费时间最主要的步骤是第 4 行解线性方程组。我们通过矩阵运算来对该方程组进行求解，花费的时间为 $(m-\tau+2)^3$。对于每个用户到达，我们需要求解 m 个线性方程组，共有 n 个用户到达。因此，此算法的时间复杂度为 $O(nm^4)$。

4 实验评估

4.1 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 $ETC \cdot APC$ 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 OpenBenchmark 提供的数据集，共有 5 种任务在 9 台机器上执行的真实数据，其中便包含了各个任务在不同机器上执行时的 ETC 与 APC[19]。Tarplee 在其基础上进行了扩充，最终包含 9 种机器类型与 30 种任务类型[20]。本文实验使用的是该文中扩充后的数据。

不失一般性，在我们所有的实验中都令 $c=1$。若不考虑最大完工时间，我们将 E_{min}^{i} 定义为用户 i 的能耗下界，$E_{min}^{i}=\min_j ETC_{ij}APC_{ij}$。此外，我们让 $p_i=\gamma E_{min}^{i}$，那么 $\gamma=p_i/E_{min}^{i}$ 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 (≤ 1)。根据 Tarplee 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[20]。因此，在本节的实验中，我们仅考虑 $\gamma \in (1,1.5]$。

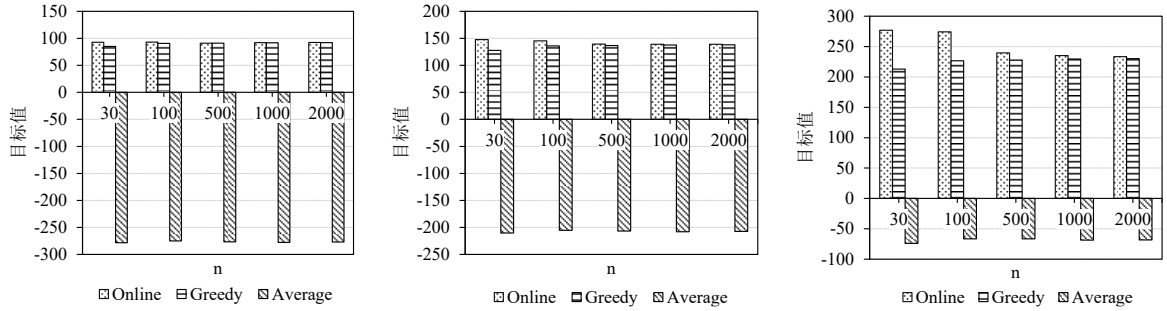
4.2 不同 γ 值对系统的影响

本实验是在上述 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。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。

γ=	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

表 1 目标值随着 γ 从 1.05 增长到 1.5 的变化

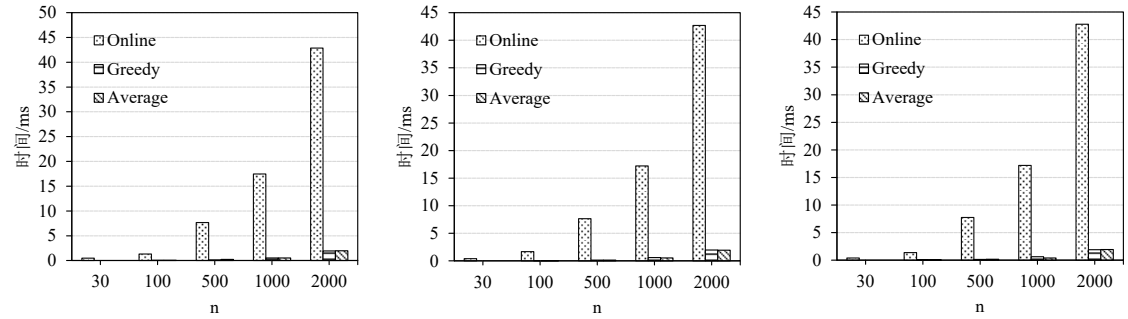
Table.1 Objective values, with γ varied from 1.05 to 1.5



(a). γ=1.2　　(b). γ=1.3　　(c). γ=1.5

图 2　三种方法的目标值随着 n 增长的变化

Figure.2　Objective values of two methods with varying n



(a). γ=1.2　　(b). γ=1.3　　(c). γ=1.5

图 3　三种方法的执行时间随着 n 增长的变化

Figure.3　Execution time of three methods with varying n.

4.3　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 $\gamma = 1.2,1.3$ 和 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 $a_i \in [200,1000]$。

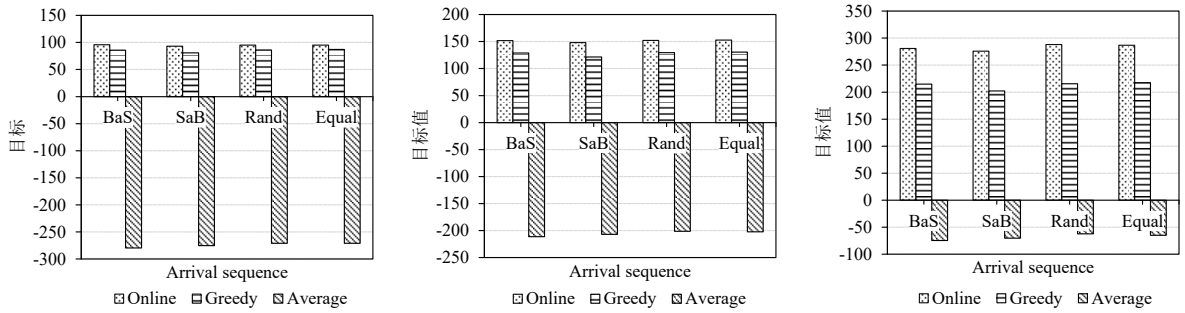
从图 2 中我们能够明显观察到，当用户数较小时 Online 方法得到的目标值更大，当用户数不断增加，Online 方法与 Greedy 方法得到的目标值几乎变得相等。这是由于当用户数 n 变得足够大时，总的任务数理所应当的也会增加，这会使得两种方法给出的调度方案产生的最大完工时间逐渐靠近最优调度所产生的最大完工时间。

在图 3 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。

4.4　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。我们令用户数 $n = 30$，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 $a_i > 500$，那么该任务包为大任务包；若用户 i 提交的任务包中的任务数 $a_i < 200$，那么该任务包为小任务包。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随机到达；“Equal”为所有用户提交的任务包任务数相同，为 400。

从图 4 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，



(a). γ=1.2 (b). γ=1.3 (c). γ=1.5

图 4 不同用户到达顺序对目标值的影响

Figure.4 The impact of different user arrival sequences on the objective value.

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。

4 总结

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。利用基于机器学习的方法也是未来值得探究的课题。

参考文献：

[1] FRIESE R, BRINKS T, OLIVER C等. Analyzing the Trade-offs Between Minimizing Makespan and Minimizing Energy Consumption in a Heterogeneous Resource Allocation Problem[C]//The Second International Conference on Advanced Communications and Computation. 2012: 81-89.

[2] TARPLEE K M, FRIESE R, MACIEJEWSKI A A等. Scalable linear programming based resource allocation for makespan minimization in heterogeneous computing systems[J/OL]. Journal of Parallel and Distributed Computing, 2015, 84: 76-86.

[3] TARPLEE K M, FRIESE R, MACIEJEWSKI A A等. Energy and Makespan Tradeoffs in Heterogeneous Computing Systems using Efficient Linear Programming Techniques[J/OL]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(6): 1633-1646.

[4] BRAUN T D, SIEGEL H J, BECK N等. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems[J/OL]. Journal of Parallel and Distributed Computing, 2001, 61(6): 810-837.

[5] DAI B, LI J, LI W. Semi-online Hierarchical Scheduling for Bag-of-tasks on Two Machines[C/OL]//Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence - CSAI '18. New York, New York, USA: ACM Press, 2018: 609-614.

[6] 胡逸騠. 面向高性能计算的能耗感知任务调度算法及应用[D]. 湖南：湖南大学, 2021.

[7] FRIESE R, KHEMKA B, MACIEJEWSKI A A等. An Analysis Framework for Investigating the Trade-Offs between System Performance and Energy Consumption in a Heterogeneous Computing Environment[C/OL]//2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum. IEEE, 2013: 19-30.

[8] FRIESE R, BRINKS T, OLIVER C等. A Machine-by-Machine Analysis of a

Bi-Objective Resource Allocation Problem[C]. 2013: 3-9.
[9] ZHANG X, LIU X, LI W等. Trade-off Between Energy Consumption and Makespan in the Mapreduce Resource Allocation Problem[C/OL]//The 5th International Conference on Artificial Intelligence and Security. 2019: 239-250.
[10]LI W, LIU X, ZHANG X等. A Task-Type-Based Algorithm for the Energy-Aware Profit Maximizing Scheduling Problem in Heterogeneous Computing Systems[C/OL]//2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, 2015: 1107-1110.
[11]LI W, LIU X, CAI X等. Approximation algorithm for the energy-aware profit maximizing problem in heterogeneous computing systems[J/OL]. Journal of Parallel and Distributed Computing, 2019, 124: 70-77.
[12]KHEMKA B, FRIESE R, PASRICHA S等. Utility maximizing dynamic resource management in an oversubscribed energy-constrained heterogeneous computing system[J/OL]. Sustainable Computing: Informatics and Systems, 2015, 5: 14-30.
[13]姜春茂, 王凯旋. 基于三支队列的实时云任务节能调度算法[J]. 郑州大学学报：理学版, 2019(2): 6.
[14]ZHANG J, XIE N, ZHANG X等. An online auction mechanism for cloud computing resource allocation and pricing based on user evaluation and cost[J/OL]. Future Generation Computer Systems, 2018, 89: 286-299.
[15]DAYARATHNA M, WEN Y, FAN R. Data Center Energy Consumption Modeling: A Survey[J/OL]. IEEE Communications Surveys & Tutorials, 2016, 18(1): 732-794.
[16]HAMEED A, KHOSHKBARFOROUSHHA A, RANJAN R等. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems[J/OL]. Computing, 2016, 98: 751-774.
[17]KAUR T, CHANA I. Energy Efficiency Techniques in Cloud Computing[J/OL]. ACM Computing Surveys, 2015, 48(2): 1-46.
[18]ORGERIE A C, ASSUNCAO M D de, LEFEVRE L. A survey on techniques for improving the energy efficiency of large-scale distributed systems[J/OL]. ACM Computing Surveys, 2014, 46(4): 1-31.
[19]OPENBENCHMARKING. Intel core i7 3770k power consumption, thermal.[EB/OL]. (2013).
[20]TARPLEE K M, MACIEJEWSKI A A, SIEGEL H J. Energy-Aware Profit Maximizing Scheduling Algorithm for Heterogeneous Computing Systems[C/OL]//2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, 2014: 595-603.