

一种异构计算系统中考虑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

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。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

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。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 [5, 6]。根据该模型, 异构计算系统中的任务调度需要确定某个类型的任务该如何分配到各个类型的机器上运行。

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, 简称为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, 简称为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 [4]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。最大完工时间定义为所有机器中最大的完工时间, 而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。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。而且, 该舍入算法的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。此外, 该篇文章也并未分析他们所提出的基于LP的舍入算法的近似比。

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, 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).提出了一个在最坏情况下也能多项式时间运行的算法, 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。该算法的时间复杂度为 $O(nm^4)$ 。

(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

最近几十年有大量关于异构计算系统中任务调度模型的研究。研究者们往往关注于如何最小化最大完工时间、最小化系统能耗或者同时考虑两者对系统的影响。对于最小化最大完工时间的工作, 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。为了最小化最大完工时间和流程时间, 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。Bingfei Dai等 [21]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。为对于考虑任务包的调度模型, 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。

Chunling Cheng等 [27]提出了一种基于休假排队论的任务调度算法来对计算系统的能耗进行优化。胡逸骥 [17]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。该算法根据应用程序的能耗约束, 为各个计算任务分配合适的计算节点, 并使用动态电压频率调整技术调整各计算节点的工作频率, 从而有效降低系统的整体能耗。

若用户提交的是上文提及的任务包, Friese 等人 [1,2]针对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。Friese 等人 [3]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。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。张晓璐等 [20]设计了一个整数线性双目标优化模型, 并提出了

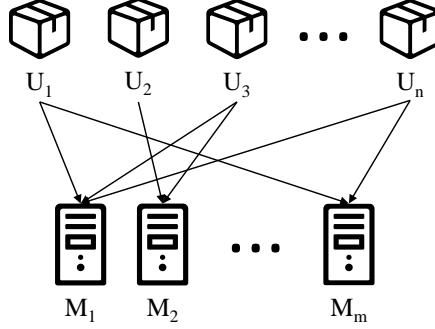


图 1 异构计算系统任务调度示例

Figure 1 Task scheduling for heterogeneous computing systems

两阶段的启发式分配算法以找到高质量的可行解决方案。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。Oxley 等 [7]提出并分析了几种启发式方法,以此来联合考虑了能源约束和截止时间约束的能量感知资源分配问题。李伟东等 [18]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。随后李伟东等 [19]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。

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(Visual machine, 简称为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。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。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。张骥先等 [24]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。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, 22, 23, 25]。

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 。图1是该异构计算系统对用户提交的任务包进行调度的一个示例。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, 简称为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, 简称为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:

$$L_j = \sum_{i=1}^n ETC_{ij} x_{ij}. \quad (1)$$

那么所有机器的最大完工时间 $MS(\mathbf{x})$, 定义为:

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

相应的, n 个用户的能量消耗为:

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

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:

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

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

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

$$x_{ij} \in \mathbb{Z}_{\geq 0}, \forall i, j. \quad (7)$$

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 $i = 1, 2, \dots, n-1$ 。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 , 我们用 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 $L_j^0 = 0, j = 1, \dots, m, E^0 = 0$ 。根据定义, 对于任意 $i = 1, 2, \dots, n$, 我们能得到以下关系:

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

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:

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

其中

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

该问题可以形式化地表示为以下整数规划 (Integer program, IP):

$$\begin{cases} \text{Maximize} & \frac{\sum_{k=1}^i p_k - cE^{i-1} - c \sum_{j=1}^m x_{ij} APC_{ij} ETC_{ij}}{MS^i} \\ \sum_{j=1}^m x_{ij} & = a_i \\ L_j^{i-1} + x_{ij} ETC_{ij} & \leq MS^i \\ x_{ij} \in Z^+ \cup \{0\}, j & = 1, \dots, m. \end{cases} \quad (11)$$

其中第二个约束等价于:

$$x_{ij} \leq \lfloor \frac{MS^i - L_j^{i-1}}{ETC_{ij}} \rfloor. \quad (12)$$

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

$$APC_{i1}ETC_{i1} \geq APC_{i2}ETC_{i2} \geq \dots \geq APC_{im}ETC_{im}. \quad (13)$$

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:

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

其中 $\tau \in \{1, \dots, m\}$ 。

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 $(x_{i1}, x_{i2}, \dots, x_{im})$,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 $\tau_2 (\geq \tau_1)$,有 $x_{i\tau_2} < \lfloor \frac{MS^i - L_{\tau_2}^{i-1}}{ETC_{i\tau_2}} \rfloor$,那么我们令

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

很容易证明 $(x'_{i1}, x'_{i2}, \dots, x'_{im})$ 是规划(11)的一个可行解。由于(13)重排序使得 $APC_{i\tau_1}ETC_{i\tau_1} \geq APC_{i\tau_2}ETC_{i\tau_2}$,该可行解的目标值会比 $(x_{i1}, x_{i2}, \dots, x_{im})$ 的目标值更小。对所有机器 $j(\geq \tau, \tau$ 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

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

因此,该引理成立。

那么对于每个 $\tau = 1, \dots, m$, 我们只需考虑部分的决策变量 $x_{ij}(j = \tau, \dots, m)$ 。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:

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

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 $x_{ij}(j = \tau, \dots, m)$ 。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 $\tau = 1, \dots, m$, 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:

算法 1 Online

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

```

1: for  $i = 1$  to  $n$  do
2:   Relabel the indices of tasks such that  $APC_{i1}ETC_{i1} \geq \dots \geq APC_{iM}ETC_{iM}$ ;
3:   for  $\tau = 1$  to  $m$  do
4:     Solve (15) to find a solution  $x_{ij}^\tau$ ;
5:     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.
6:   end for
7:   for  $j = 1$  to  $m$  do
8:     Assign  $\lceil x_{ij} \rceil$  tasks of type  $i$  to machines of type  $j$ , until all tasks are assigned;
9:     Update the  $L_j$  of type  $j$ .
10:  end for
11: end for

```

输出: \mathbf{x}

定理1 算法1的时间复杂度为 $O(nm^4)$ 。

证明 算法1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 $O(nm^4)$ 。

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 $ETC \cdot APC$ 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 Intel 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 [26] 中的5种任务在9台

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

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。文章 [4]在其基础上进行了扩充,最终包含9种机器类型与30种文本类型,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 $\gamma = 1.2, 1.3$ 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。

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。

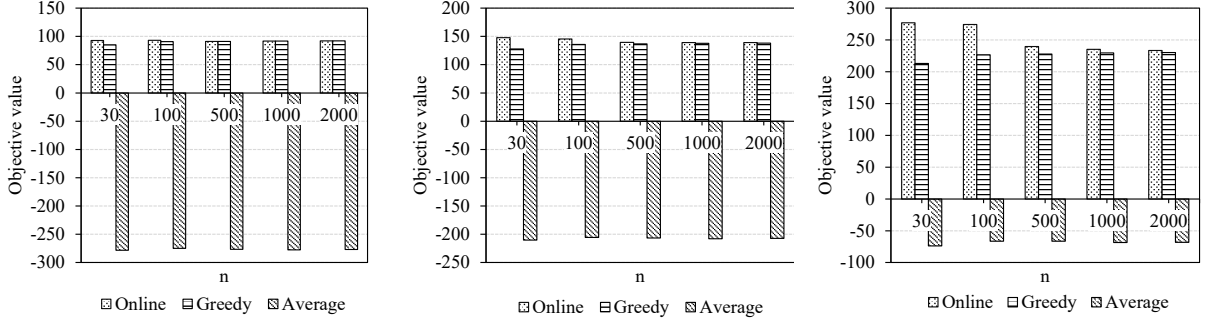


图 2 三种方法的目标值随着 n 增长的变化

Figure 2 Objective values of two methods with varying n .

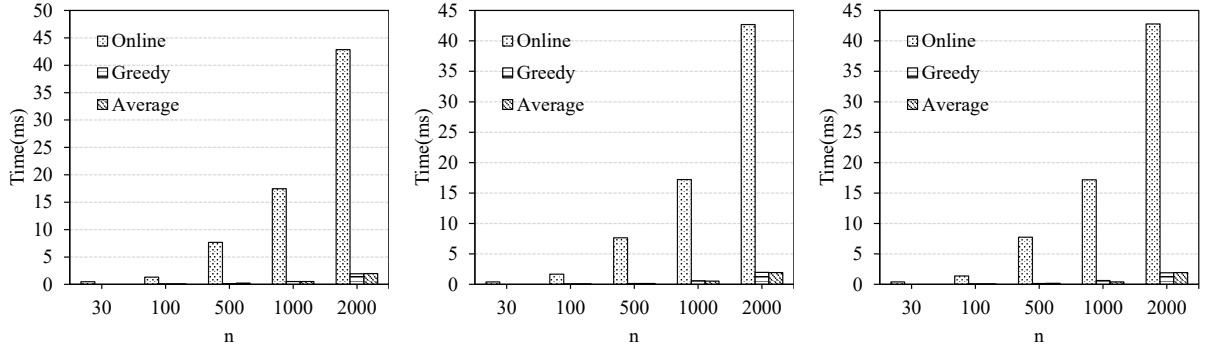


图 3 三种方法的执行时间随着 n 增长的变化

Figure 3 Execution time of three methods with varying n .

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。

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，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

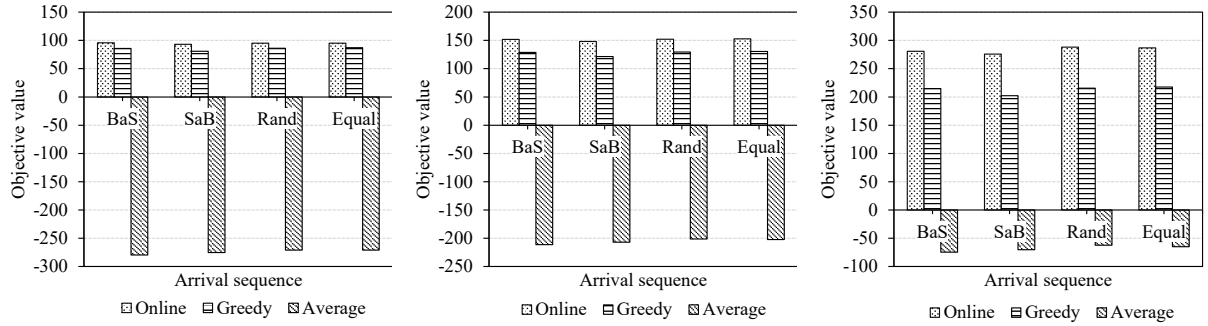


图 4 不同用户到达顺序对目标值的影响

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

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. Frieze, 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. Frieze, 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. Frieze, 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. Frieze, 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. Frieze, 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 胡逸骥.面向高性能计算的能耗感知任务调度算法及应用.2019.湖南大学,PhD dissertation.
 - 18 W. Li, X. Liu, X. Zhang and X. Cai, “A Task-Type-Based Algorithm for the Energy-Aware Profit Maximizing Scheduling Problem in Heterogeneous Computing Systems,” *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 1107-1110, 2015.
 - 19 Weidong Li, Xi Liu, Xiaobo Cai and Xuejie Zhang, “Approximation algorithm for the energy-aware profit maximizing problem in heterogeneous computing systems,” *Journal of Parallel and Distributed Computing*, vol. 124, pp. 70-77, 2019.
 - 20 Zhang, X., Liu, X., Li, W., Zhang, X. (2019). “Trade-off Between Energy Consumption and Makespan in the Mapreduce Resource Allocation Problem.” *2019 The 5th International Conference on Artificial Intelligence and Security*, Lecture Notes in Computer Science, vol. 11633, 2019.
 - 21 Bingfei Dai, Jianping Li, and Weidong Li. 2018. “Semi-online Hierarchical Scheduling for Bag-of-tasks on Two Machines.” *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence (CSAI '18)*, pp. 609 - 614, 2018.
 - 22 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.
 - 23 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.
 - 24 J. Zhang, N. Xie, X. Zhang, W. Li, “An online auction mechanism for cloud computing resource allocation and pricing based on user evaluation and cost,” *Future Generation Computer Systems*, vol. 89, pp. 286-299, 2018.
 - 25 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.
 - 26 (2013, May) Intel core i7 3770k power consumption, thermal. [Online]. Available: <http://openbenchmarking.org/result/1204229-SU-CPUMONITO81>
 - 27 Chunling Cheng, Jun Li and Ying Wang “An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing,” *Tsinghua Science and Technology*, vol. 20, pp. 28-39, 2015.