# 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

1. 背景

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资源分配越来越重要。最近学界提出了一种名为任务包（bag-of-tasks）的静态调度模型。与以往的经典调度模型相比，任务在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。这个新的模型会确定每个不同类型机器上执行的不同类型任务的数量，而不是任意的分配这些任务。虽然高性能计算中心会有成千上万台机器，并且需要执行的任务数也无法估计。但是二者的类型是很有限的。这让我们为该模型设计一个能得到拟最优调度的算法百年的可能。

经典的能量感知调度模型旨在最小化bag-of-tasks所消耗的能量或最大完工时间(makespan)。然而，高性能计算中心的应该尝试将每个单位时间的运行利润最大化，其中利润等于用户为一个任务包支付的价格减去执行期间消耗的电力成本。通过将能源成本和完工时间纳入单位时间利润最大化的目标，最近有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） 它们执行许多任务，但任务类型的数量有限。

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。本发明的主要贡献如下：

1. 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基于任务类型的算法，该算法能给出一个准确的可行解；
3. 分析了该基于任务类型的算法的近似比。

二、在线调度模型

在一个异构计算机系统中包含了中不同的机器和中不同的用户。用户提交的任务包中的任务数为，执行这类任务能产生的收益为。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：是一个维矩阵，其中是用户的任务在机器上的固定执行时间（estimated time to compute）；同样是一个维矩阵，其中是用户的任务在机器上的平均功率消耗（average power consumption）。我们让来表示用户的任务分配给机器的数量。对于一个可行解，机器的负载被定义为：



那么所有机器的最大完工时间(makespan)，定义为：



相应的，个用户的能量消耗为：



用表示每个单位能耗的成本。那么考虑任务包的能量感知利润最大化问题（Energy-Aware Profit Maximizing, EAPM）可以形式化的用以下非线性整数规划表示。

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

目标函数是要最大化单位时间的收益，其中是决策变量。第一个约束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，第二个约束确保了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它的所有任务。因此研究考虑任务包的EAPM问题的在线算法是很有必要的。这个在线算法考虑在用户的任务会在用户到达之前就被分配，这里。不失一般性，我们假设用户的所有任务会在用户到达之前就被分配摸个机器，这里。更重要的一点是，当用户要被分配的任务数非常大时，我们不能逐个分配这个任务。因此为考虑任务包的EAPM问题设计一个有效地算法是很有必要的。

三、在线算法

在本节中，我们为考虑任务包的EAPM问题提出了一个有效的在线算法。对于每一个用户，我么用和表示分配完前个用户的任务之后机器的负载和总能耗。初始情况下，,，。根据定义，对于任意，我们能得到一下关系：



当用户到达时，我们将确定的值并让，此时的目标值为：



其中



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

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

其中不等式1.14等价于：



为了便于操作，我们将服务于用户的任务的机器按照降序排序。不失一般性，假设



我们容易证明在最优情况下，存在某个，使得



那么对于每个，我们只需考虑部分的决策变量。要得到用户到达时的所有值，同时的值也是未知的。但我们能通过以下方程组得到我们想要的结果：



注意到该线性方程组总共有个等式和个未知数，未知数包含和。因此，该线性方程组能在多项式时间内得到解。对于每一个，我们都能得到一组的解。比较这组可行解的目标值，我们能找到最优的那个。之后，我们从到1来分配任务，将个用户的任务分配给机器，直到所有任务都被分配。

以下是我们的算法的伪代码：

|  |
| --- |
|  |
| 1: |
| 2: |
| 3: |
| 4: |
| 5: |
| 6: |
| 7: |
| 8: |
| 9: |
| 10: |
| 11: |
| 12: |

四、应用实例

某数据中心现共有一下4种CPU类型的机器：

1. AMD A8-3870K
2. AMD FX-8150
3. Intel Core i3 2120
4. Intel Core i5 2400S

需要执行的任务类型如下：

1. C-Ray
2. 7-Zip Compression

这2种任务在4种机器上的ETC如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 59.72 | 27.01 | 72.81 | 42.15 |
| 105.96 | 55.72 | 109.78 | 88.14 |

这2种任务在4种机器上的APC如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 139.6 | 223.1 | 97.5 | 121.3 |
| 137.2 | 215 | 98.7 | 119.8 |

任务类型为1的任务数为40，任务类型为2的任务数为80。我们令，当任务类型为1的任务到达时，我们首先计算的值，并按的值从大到小排列。结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| AMD A8-3870K | Intel Core i3 2120 | AMD FX-8150 | Intel Core i5 2400S |
| 8336.912 | 7098.975 | 6025.931 | 5112.795 |

此时机器的序号也被按照上面从左到右的顺序重新索引。

我们令，此时没有，我们通过解以下线性方程组能得到时的解：



解为,此时目标值为-125.363。依次的计算时的解及目标值，我们能得到当时收益最大为-2.676，此时。

最后我们将类型1的所有任务都放到Intel Core i5 2400S机器上执行。当类型2的任务到达时，重复以上步骤。

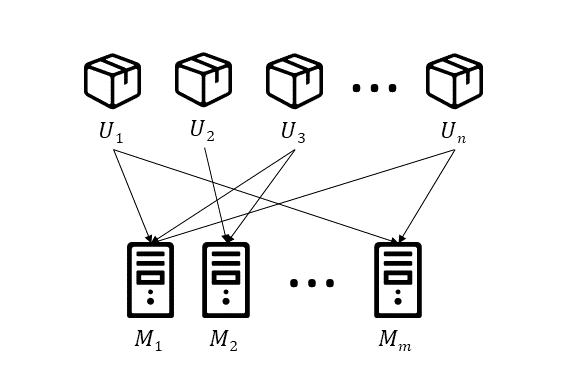
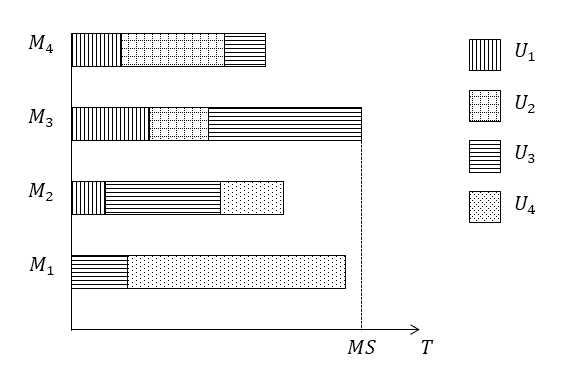


图1. 种用户提交任务给拥有种机器的高性能计算中心

图2. 包含4种用户与4种机器的分配实例