# 论文分享

论文：A Case for Task Sampling based Learning for Cluster Job Scheduling

# 摘要

准确估计作业运行时属性的能力允许调度器有效地调度作业。最先进的在线集群作业调度器使用基于历史的学习，它使用过去的作业执行信息来估计新到达作业的运行时属性。然而，随着集群技术（硬件和软件）的快速发展以及用户输入的变化，作业运行时属性可能会随着时间的推移而变化，从而导致不准确的预测。

在本文中，通过主动地对每个作业的一小部分任务进行采样和调度，探索了实时学习作业运行时属性的潜力和局限性。这种基于任务采样的方法利用了同一作业的任务的运行时属性之间的相似性，并且天生不受作业行为变化的影响。在比较基于任务抽样的学习（空间学习）和基于历史的学习（时间学习）时，的研究关注两个关键问题：

（1）空间学习比时间学习更准确吗？

（2） 如果是这样的话，将剩余任务的调度延迟到完成抽样任务之前，是否可以通过提高精度得到更多补偿，从而提高工作绩效？

对3条不同倾斜和工作分布的生产轨迹进行了分析和实验分析，结果表明，在空间中学习可以更加准确。在Azure上对使用3个生产集群作业跟踪的通用作业调度器中两种学习方法进行了模拟和试验台评估，结果表明，尽管存在在线开销，但与现有技术的基于历史的预测器相比，空间学习将平均作业完成时间（JCT）减少了1.28倍、1.56倍和1.32倍。

# 背景

在大数据计算集群中，当作业上线时，它们的运行时特征是不知道的。由于这种信息的缺乏，集群调度器很难确定正确的作业执行顺序，从而难以进一步优化。解决集群调度问题的一个有效方法是了解暂挂作业的运行时特征，这允许调度程序利用已知最优的离线调度算法，例如，最短作业优先(SJF)，以最小化平均完成时间。

所有以前的学习算法都是通过观察相同作业的历史执行来学习作业运行时特征的。基于历史的学习计划的有效性主要取决于两个条件：（1）工作是重复的；（2） 随着时间的推移，相同或类似工作的表现将保持一致。在实践中，这两个条件往往不成立。

在本文中，探索了一种学习在线分布式作业运行时特性的替代方法，以促进集群作业调度。该方法的动机是对运行的分布式作业有以下观察发现:

(1)作业通常有一个空间维度，即它通常由许多任务组成;

(2)作业的任务(在同一阶段)通常执行相同的代码，处理大小相似的不同数据块[9,16]。

这些观察结果表明，如果调度器首先调度一个作业的几个采样任务(称为试点任务)，并运行到完成，那么它可以使用观察到的这些任务的运行时属性来准确估计整个作业的运行时属性。这种**基于任务抽样的方法**有效地学习了工作在空间维度上的属性。将新的学习方案表示为SLearn，表示**“空间学习”**。

直观地说，通过使用先导任务的执行来预测其他任务的属性，SLearn避免了基于历史的学习技术的主要缺点，即依赖于重复的任务和任务属性在一段时间内保持不变。但是面临两个挑战：

(1)其估计精度会受到任务运行时属性的变化，即任务倾斜的影响;

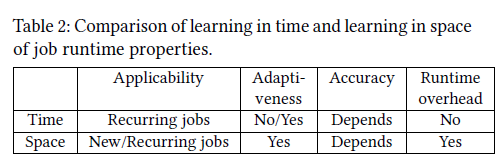
(2)延迟安排作业的剩余任务，直到完成采样的任务，可能会影响作业的完成时间。

本文对基于历史的学习(时间学习)和基于样本的学习(空间学习)进行了全面的比较研究，系统地回答了以下问题:

(1)空间学习是否比时间学习更准确?

(2)如果是这样，是否可以延迟调度任务的剩余任务，直到采样任务完成，以提高精度和提高工作性能，如完成时间?

# SLearn – Learning in Space



* 适用性：SLearn可以应用于任何新的job
* 准确度：基于历史的方法的准确性受作业运行时属性随时间保持的稳定性的影响，而基于采样的方法的准确性受任务运行时属性的变化（即任务倾斜的程度）的影响。
* 适应性：根据历史数据得出结果，不能持续性的调整。
* 运行时开销：基于历史的方法有一个固有的优点，即运行时开销非常低，甚至为零。它对历史数据进行精细分析，生成预测模型。相比之下，基于抽样的预测器没有固定的成本，但需要先运行几个试点任务，直到完成，然后再安排剩余的任务。这可能会延迟非采样任务的执行

## 精度分析

假设两个job，j1和j2，每个job有n个task。

假设j1中task的大小为1，j2的task大小未知。

x表示j2中task的平均大小。则j1的所有task大小综合为n，j2的为n\*x。

如果x<1，则先调度j2，如果x>1则先调度j1

假设x服从方差为σ0，均值为μ的正太分布0。

如果x为确定值，job的各个task的随机大小服从方差为σ1，均值为x的正太分布。

……

基于历史的方案或基于采样的方案对于未知作业是否具有更好的预测精度取决于作业方式方差σ0和任务方式方差σ1之间的关系。

如果σ0大，而σ1小，则基于采样的方案效果更好。

如果σ0小，而σ1大，则基于历史的方案效果更好。

大多数情况下，都是处于第一种的情况..所以…

# Scheduler and Predictor Design

## Generic Scheduler GS

**作业间调度**：

对总估计运行时间，即(平均任务运行时间 x task数)进行排序，根据排序结果进行调度

**无饥饿调度**：

使用多级优先级队列结构来避免工作饥饿。

GS收到作业的运行时估计值后，将其分配给优先级队列。在队列中，我们使用FIFO来调度作业。在整个队列中，我们使用加权资源共享，其中优先级队列根据其优先级接收资源共享。

**基本调度操作**：

GS跟踪每个优先级队列使用的资源。它将下一个可用资源分配给一个队列，以便在队列之间保持资源的加权共享，以避免饥饿。分配给队列的资源总是分配给队列最前面的作业。

## 集成SLearn

为了将SLearn与GS无缝集成，我们需要使用一个优先级队列来调度采样任务。我们将其表示为采样队列。

**快速采样：**

一个困难是如何确定采样队列与其他优先级队列的优先级。一方面，采样任务应具有较高的优先级，以便作业运行时估计能够快速完成。另一方面，已经估算了运行时间的作业不应因学习新作业而进一步延迟。为了平衡这两个因素，我们使用GS中的第二高优先级作为采样队列。

**处理ThinJob：**

在SLearn中，当一个新作业到达时，SLearn只安排其试点任务，并延迟其他任务，直到试点任务完成并估计作业运行时间。这样的设计选择可能会无意中导致thin job的JCT更高。

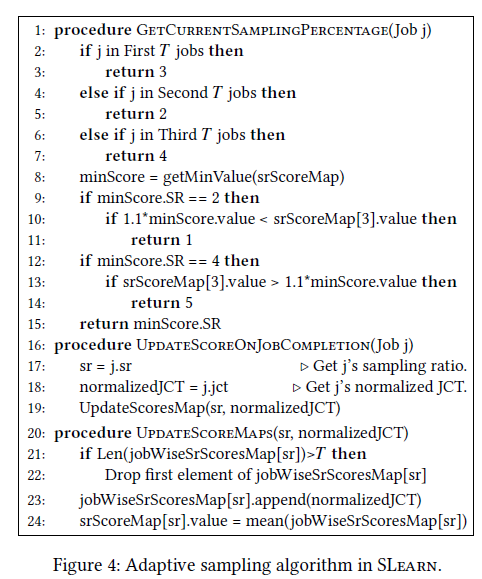
如果作业的任务数量小于了thin job定义的阈值thinLimit，直接调到最优先执行。

**基础操作：**

新作业到达后，群集管理器会将作业信息异步传递给GS，GS会将信息转发给SLearn。如果作业中的任务数处于thinLimit之下，SLearn会将其分配给优先级最高的队列；否则，作业将被分配到采样队列，其任务的一个子集（试点任务）将被安排在该队列中运行。一旦通过采样估计出作业的运行时间，它就会被放置在与其运行时间估计相对应的优先级队列中，其他任务将被安排在该队列中。

**试点任务调度：**

当新工作到来时，SLearn首先需要确定试点任务的数量。采样更多的任务可以提供更高的估计精度，但也会在早期消耗更多资源，这可能会延迟其他作业，如果该作业被证明是一个较长的作业，并且应该计划在SJF下稍后运行。此外，我们还发现，最佳采样率似乎在不同的记录中有所不同。为了平衡权衡，使用自适应算法动态确定采样率，如图所示。



该算法的基本思想是建议一个采样率，该采样率会关联最低作业完成时间。为了实现这一点，对于定义的可能采样率范围内的每个值（在1%到5%之间），它会维护一个运行分数（srScoreMap），这是使用相应采样率的最近完成的T个作业的平均JCT。在实践中，我们发现T值为100相当有效。在系统启动期间，它尝试对前3T作业（第2-7行）的采样率分别返回2%、3%和4%。如果从3%下降到2%或从3%上升到4%会降低标准化的JCT，它会进一步尝试1%和5%的采样率。之后，对于每个新作业，它使用具有最低运行分数的抽样比率。最后，每次作业完成后，都会更新分数图（第16-24行）。

# 实验和结论

* 通过定量、跟踪和实验分析，证明SLearn可以比基于历史的方案预测作业运行时属性的准确性高得多。对于2Sigma，谷歌2011和谷歌2019聚类跟踪，SLearn的预测误差中值分别为18.98%，13.68%和51.84%，而基于历史的最先进的3Sigma的预测误差中值分别为36.57%，21.39%和71.56%。
* 实验表明，通过采样作业任务来学习作业运行时属性，尽管会延迟调度作业的剩余任务，但可以通过提高精度得到补偿，从而降低平均JCT。实验结果表明，与基于历史的预测器相比，对于提取的2Sigma、谷歌2011和谷歌2019轨迹，SLearn分别减少了平均JCT 1.28x、1.56x和1.32x。
* 将展示如何将基于采样的学习扩展到DAG作业的调度。使用由谷歌2019跟踪生成的DAG跟踪，展示了基于采样和基于历史记录的混合方案比纯基于历史记录的方案降低了平均JCT 1.25倍。

# 思考

不足：ThinLimit不能动态调整，难以判定。文中没什么图示内容，理解起来比较生涩。