生产级高性能计算系统中实现无中断调度迁移的运维策略

Glen MacLachlan* Joseph Creech* Rubeel Muhammad Iqbal maclach@gwu.edu jcreech@gwu.edu rubeel@gwu.edu 乔治·华盛顿大学 乔治·华盛顿大学 乔治·华盛顿大学 美国华盛顿特区 美国华盛顿特区 美国华盛顿特区

Clark Gaylord cgaylord@gwu.edu 乔治·华盛顿大学 美国华盛顿特区

Jake Messick jake_messick@gwu.edu 乔治·华盛顿大学 美国华盛顿特区

arXiv:2603.27863v1 [cs.DC] 29 Mar 2026

摘要

将异构高性能计算（HPC）系统迁移到资源感知调度模式，会带来技术和行为方面的挑战，尤其是在用户工作流已成熟的生产环境中。本文介绍了一项案例研究：在生产级学术高性能计算集群的生命周期中期，将调度方式从节点独占迁移为可消耗资源调度，同时不影响活动工作负载。我们阐述了一种运维策略，结合基于 atime 的兼容层、可观测性驱动的反馈和有针对性的用户参与，引导用户采用显式资源声明。在整个迁移过程中，这种方法保障了活跃研究工作流，避免了直接切换给用户社区带来的干扰。部署后，CPU 工作负载的队列等待时间中位数从 277 分钟降至不足 3 分钟，GPU 工作负载则从 81 分钟降至 3.4 分钟。采用基于 TRES 的提交方式的用户表现出很强的长期留存率。这些结果表明，成功的调度迁移不仅取决于系统配置，还取决于可观测性、用户参与和运营设计之间的协调。

CCS 概念

• **分布式计算方法**； • **调度**； • **网络监控**；

关键词

高性能计算、Slurm、TRES、资源调度、集群运维、可观测性

ACM 参考文献格式：

Glen MacLachlan、Joseph Creech、Rubeel Muhammad Iqbal、Clark Gaylord 和 JakeMessick。2026年。《生产级高性能计算系统中实现平稳调度转换的运维策略》。载于。 ACM，美国纽约州纽约市，4页。 https://doi.org/XXXXXXX.XXXXXXX

*两位作者对本研究的贡献相同。

Conference'17，美国华盛顿特区2026年。
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

1 引言

异构高性能计算（HPC）系统中的高效资源分配，依赖于能够明确向调度器和用户公开计算资源（包括 CPU、内存和 GPU）的调度策略；这一挑战已在并行作业调度领域得到广泛研究 [1]。

本文介绍了一项生产环境案例研究，探讨如何使用 Slurm 可跟踪资源（TRES） [5, 8]，将异构高性能计算集群从节点独占调度迁移至可消耗资源调度。本文重点介绍调度器配置，以及如何制定运营策略，在在线生产环境中部署可能造成中断的变更，同时维持服务不中断。

乔治·华盛顿大学的旗舰高性能计算系统 Pegasus [2], 是一个生产集群，为规模庞大的跨学科研究群体提供支持。迁移时，Pegasus 包含约 205 个计算节点（42 个 GPU 节点）、8,600 个 CPU 核心、总计 280 TB 的系统内存、2 PB 存储，以及 800 名活跃研究人员。

随着时间推移，节点独占调度模式造成了若干结构性低效问题，包括节点装箱不理想、队列等待时间增加，以及资源分配模式不匹配。这暴露出技术和行为方面的挑战：作业提交模式造成了可衡量的低效，而引导研究人员转向显式资源声明，则需要审慎的运营设计。

我们提出了一种生产部署策略，将分阶段过渡与可观测性驱动的反馈相结合，使系统能够在不干扰活动工作负载的情况下完成过渡。让调度器配置与面向研究人员的运营实践保持一致，可大幅缩短队列等待时间、提高资源利用率，并促进用户持续采用显式资源请求。为解决技术层面的问题，集群迁移至 Slurm 的可追踪资源模型，通过 cgroup 强制执行，并借助 Slurm 通用资源（GRES）框架 [6, 8]实现 GPU 记账。尽管技术实现本身已有详尽文档记录，运营方面的挑战主要在于用户采用。本文其余部分将重点介绍过渡策略、可观测性机制，以及迁移期间积累的运营经验。

2 旧模型摩擦与过渡策略

在 Pegasus 原有的节点独占调度机制下，区分资源类别的唯一方式是分配分区。随着时间推移，这导致专用分区不断增多：高内存节点、GPU 类型、调试队列、可视化节点以及高吞吐量配置。每增加一个分区，都能逐步揭示资源能力，但同时也让用户觉得提交作业的环境更加复杂。在可消耗资源调度模式下，这些区别通过每作业资源请求来表达，而不是通过分区标识。由此产生的复杂性进一步强化了几种行为：

- 基于推断的资源可用性而非资源声明进行分区选择
- 分配了 GPU，却没有相应地利用 GPU
- 由于默认假设节点具有排他性，导致每个节点的核心利用率偏低

迁移前遥测数据揭示了持续存在的模式：CPU 打包效率低、频繁分配 GPU 资源却未加以利用，以及由节点独占调度导致的队列拥塞。这些低效现象与节点独占调度已知的局限性一致，包括资源碎片化和利用率不佳。在大规模 GPU 集群环境中也观察到了类似的低效现象：请求的资源与实际资源使用情况不匹配，导致资源利用不足 [3]。这些观察结果表明，有必要制定兼顾调度策略和用户提交行为的过渡策略。

Slurm 的可跟踪资源（TRES）模型通过 select/cons_tres 插件实现，可在整个集群范围内启用或禁用 [5]。因此，调度模型的变更必须统一实施，这就需要借助运维策略，支持用户在在线生产环境中完成过渡。因此，我们没有采用直接切换，而是选择了一种能够让研究群体逐步适应、同时不中断活动工作负载的方案。

为减少用户在过渡过程中的阻力，我们设置了为期 90 天的过渡期，在此期间允许使用旧式提交模式，但会对其进行监控。这种做法既保障了服务连续性，也让用户能够逐步适应显式资源声明。轻量级包装脚本会在可能的情况下，将旧式作业提交转换为能够感知资源的等效提交，使现有工作流得以继续运行，同时引导用户显式请求 CPU、内存和 GPU 资源。这些包装脚本仅在 90 天过渡期内提供，为用户逐步转向原生的基于 TRES 的提交模式提供了明确路径。

过渡期间，允许使用旧式提交模式，但会进行监控，并向用户提供有关历史低效问题的针对性反馈。为促进用户采用，团队编写了文档，强调资源声明而非分区标识；同时通过答疑时间和全体交流会，从吞吐量和公平性的角度说明此次变更。

3 可观测性作为稳定机制

系统可观测性在评估调度迁移的影响方面发挥了关键作用。监控结合了 Slurm [7] 记账数据与 Zabbix 基础设施遥测数据 [9]。Slurm 记账记录提供了作业级遥测数据，包括申请的

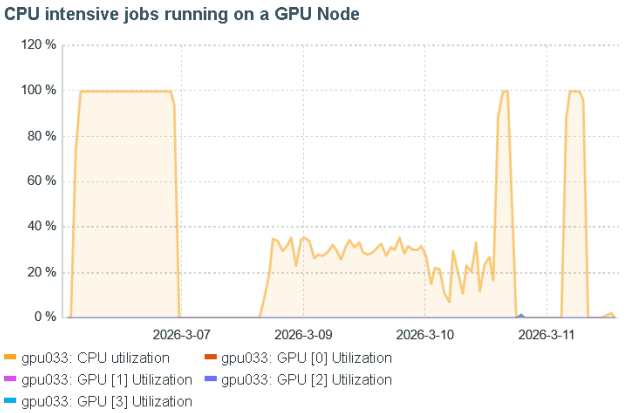


图 1：GPU 节点上运行的 CPU 工作负载，未能充分利用可用的 GPU 设备。

资源、已分配资源、作业时长和队列等待时间。Zabbix 代理收集了节点级指标，包括内存使用量、GPU 利用率和 CPU 核心利用率。

跟踪的运行指标包括：

- 按分区统计的队列等待时间百分位数
- CPU 打包效率
- GPU 分配匹配度
- 内存利用率与请求内存的对比

这些指标使我们能够向用户提供针对性反馈，并帮助他们迅速调整行为。例如，在全体交流会和一对一咨询期间，我们使用 Zabbix 实时可视化 GPU 利用不足情况，以展示请求的资源与实际资源使用情况之间的不匹配。这种可见性帮助研究人员了解自身的资源使用模式，并引导他们更有效地提交作业。

图 1 展示了在五天内，单个支持 GPU 的节点上运行的三个不同旧式作业。尽管占用了支持 GPU 的节点，这些作业的 GPU 利用率几乎为零或完全没有，表明 GPU 资源虽已分配，却未被使用。这种模式反映了作业提交策略优先考虑快速启动，而非恰当的资源匹配，导致稀缺的 GPU 容量未得到有效利用。相比之下，在基于 TRES 的调度下，这类不匹配会被明确揭示并受到限制，因为资源请求必须与实际使用情况相符。

4 测量效果

我们分两个阶段评估结果：迁移前的基线阶段和 90 天过渡期。在这两个阶段，我们跟踪了队列等待时间的变化，以及用户对基于 TRES 的提交方式的采用和持续使用情况。这些指标共同反映了系统级性能，以及用户对调度过渡的行为响应。

4.1 队列等待时间

表1显示了转型期间的队列等待时间分布。由于各阶段的工作负载不同，各类别的作业数量也有所变化。对于 CPU 工作负载，TRES 降低了等待时间中位数

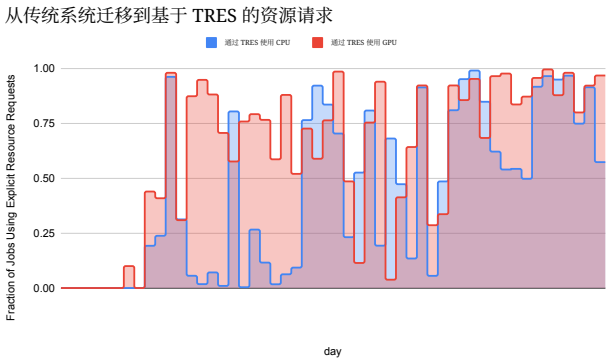


图2：转型期间使用 CPU 和 GPU 资源显式声明的作业每日占比。数据表明，基于 TRES 的提交模式逐步得到采用，但这一趋势并非单调变化，反映了工作负载构成和用户行为随时间变化的情况。

表1：TRES 调度部署前后的队列等待时间对比。所有等待时间均以分钟为单位。

分区	Jobs	P50	P75	P90
旧版 CPU（前）	393 188	277.18	800.69	3026.26
CPU（TRES）	**31 563**	**2.97**	**106.12**	**797.65**
旧版 CPU（后）	49 428	294.80	513.19	642.53
旧版 GPU（前）	61 077	81.05	1031.55	5567.88
GPU（TRES）	**14 796**	**3.40**	**137.83**	**461.60**
旧版 GPU（后）	6484	344.05	590.57	748.14

等待时间从 277 分钟降至不足 3 分钟，第 90 百分位延迟则从超过 3,000 分钟降至不足 800 分钟。GPU 工作负载也有类似改善：等待时间中位数从 81 分钟降至 3.4 分钟，第 90 百分位数则从超过 5,500 分钟降至不足 500 分钟。过渡后提交的旧版 GPU 作业仍面临显著更长的等待时间中位数（344 分钟），这表明，随着用户群转向基于 TRES 的调度，继续依赖传统作业提交方式与显著更长的等待时间相关。

4.2 采用后的行为与持续使用

图2展示了过渡期间 TRES 采用的变化情况，既突显了显式资源声明的总体增长，也体现了短期波动。图3展示了持续使用 TRES 的卡普兰–迈耶 [4] 估计值。回退前作业数（JUR）定义为初次采用 TRES 后提交的作业数，直到首次回退到传统系统为止；若未发生回退，则计算至观测窗口结束。曲线在早期急剧下降，大多数回退发生在最初几个作业之后；随后则缓慢下降，且置信区间较宽，反映出高频用户这一较小风险集中的持续使用情况较为稳定。这种模式与早期提交起到筛选

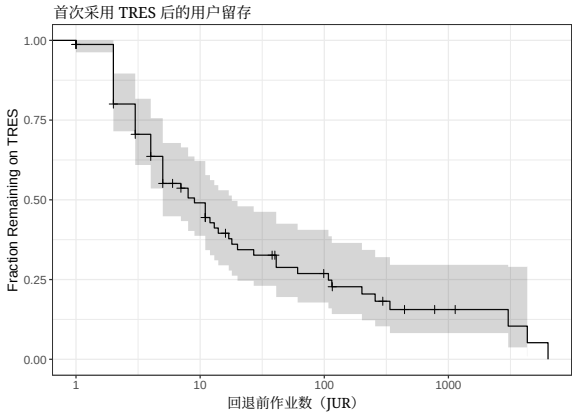


图 3：卡普兰–迈耶估计：持续使用 TRES 的概率与恢复到传统系统前的作业数（JUR）之间的关系（横轴采用对数刻度）。曲线先是迅速下降，随后趋于平稳，表明在最初几项作业后仍坚持使用的用户具有稳定的持续使用率。

阶段一致：观察到调度结果有所改善（包括队列等待时间中位数缩短，例如从 277 分钟降至 3 分钟）的用户，往往会继续使用 TRES。

部署约一周后，一位计算生物学家主动提供了反馈："我只是想对新调度系统的实施表示感谢。从节点利用率来看，它似乎取得了巨大成功。"

5 结论

此次部署表明，在生产级高性能计算环境中引入可消耗资源调度，本质上是一项运营和行为挑战，而不仅仅是技术挑战；单靠技术变革只能暴露低效之处，却无法解决这些问题。

可观测性推动了变革，使用户能够发现申请资源与实际使用资源之间的差异；而为期 90 天的兼容期则为研究人员调整工作流提供了所需时间，同时尽可能减少了干扰。

这些结果表明，要成功实现调度迁移，最好协调运用遥测、用户参与和分阶段推出策略。

致谢

作者感谢 Fong Banh、Kai Leung Wong 和 Kevin Weiss 在向用户提供支持方面所作的贡献，帮助用户过渡到资源感知型调度，包括指导用户采用更新后的提交流程，以及升级处理运营问题。

参考文献

[1] Dror G. Feitelson 和 Larry Rudolph。1997年。《并行作业调度：问题与方法》。《计算机科学讲义》1291（1997），1–18。[2] 乔治·华盛顿大学信息技术部门。2026年。Pegasus 高性能计算集群。https://it.gwu.edu/hpc-pegasus 访问于2026年3月。[3] Myeongjae Jeon、Shivaram Venkataraman、Amar Phanishayee、钱俊杰、Wen- cong Xiao 和 Fan Yang。2019年。《大规模多租户 GPU 集群分析》

深度神经网络训练工作负载。载于《*2019 年 USENIX 年度技术会议论文集（USENIX ATC*）》。USENIX 协会，美国华盛顿州伦顿，947–958。[4] Edward L. Kaplan 和 Paul Meier。1958年。《基于不完整观测的非参数估计》。《美国统计学会期刊》53卷，第282期（1958年），457–481。[5] SchedMD LLC。2026年。《Slurm 中的可消耗资源》。https://slurm.schedmd. com/cons_tres.html 访问于2026年3月8日。[6] SchedMD LLC。2026年。《通用资源（GRES）调度》。https://slurm.schedmd. com/gres.html 访问于2026年3月8日。

[7] SchedMD LLC。2026年。Slurm 工作负载管理器文档。https://slurm.schedmd.com 访问于2026年3月。[8] Andy B. Yoo、Morris A. Jette 和 Mark Grondona。2003年。SLURM：简单 Linux 资源管理工具。收录于作业调度策略与并行处理（*JSSPP*）（《计算机科学讲义》，第 *2862* 卷）。施普林格，美国华盛顿州西雅图，44–60。[9] Zabbix LLC。2026年。Zabbix 监控解决方案。https://www.zabbix.com 访问于2026年3月。