1.1 国内外研究现状

1.1.1 编排系统与负载感知

综：

如今大数据应用被广泛采用，主要业务场景包含但不限于日志统计，离线数据分析等，这些应用通常被称为批处理应用(离线应用)。这类应用的特点为: (1)延迟不敏感。(2)资源占用高。

批处理应用领域著名的框架有 Spark 错误!未找到引用源.，Flink 错误!未找到引用源.；批处理应用的分发与资源管理由一些调度器处理，主要的知名开源调度编排系统有 Mesos 错误!未找到引用源.，Yarn 错误!未找到引用源.。

与批处理应用相对应的是在线应用，这类应用的特性为: (1)延迟敏感。(2)资源需求周期性强。

早期，在线应用都是以二进制形式，单机部署的。资源浪费严重，且动态伸缩灵活性很差。Openstack 错误!未找到引用源.出现充分利用虚拟化的特性，将物理机器池化成虚拟机的粒度，提高了动态伸缩的灵活性，不过此时应用的部署还是单机粒度。为了推动进一步的资源节省，Docker 错误!未找到引用源.与 Kubernetes 错误!未找到引用源.应运而生。Docker 基于轻量虚拟化的机制，实现应用更细粒度的部署，标准化的形式实现应用的交付与生命周期管理。在 Docker 的基础上，Google 根据运营了十多年的编排系统 Borg 进行开源，取名为 Kubernetes；Kubernetes 提供了一个易于拓展且标准化的容器编排系统。

述：

然而，如今开源的编排系统，都没有真正地实现负载感知。开放的调度决策中，仅考虑业务方为应用定义的任务资源需求与节点资源状态进行调度决策。但是这存在两个缺陷: (1)用户定义的资源需求是静态的，应用实际的资源需求却是动态的。(2)用户定义的资源需求是不准确的，业务方通常不会在意应用究竟需求多少资源，业务方通常会直接选择最大可选资源，如图 错误!文档中没有指定样式的文字。.1 某集群一段时间内 CPU 资源使用量与请求量对比图。所以采用目前开源的编排系统实现混部平台将引起的问题包括: (1)集群中节点的负载不均衡，极易出现高负载节点，集群存在应用服务质量受影响的风险。(2)集群资源浪费。

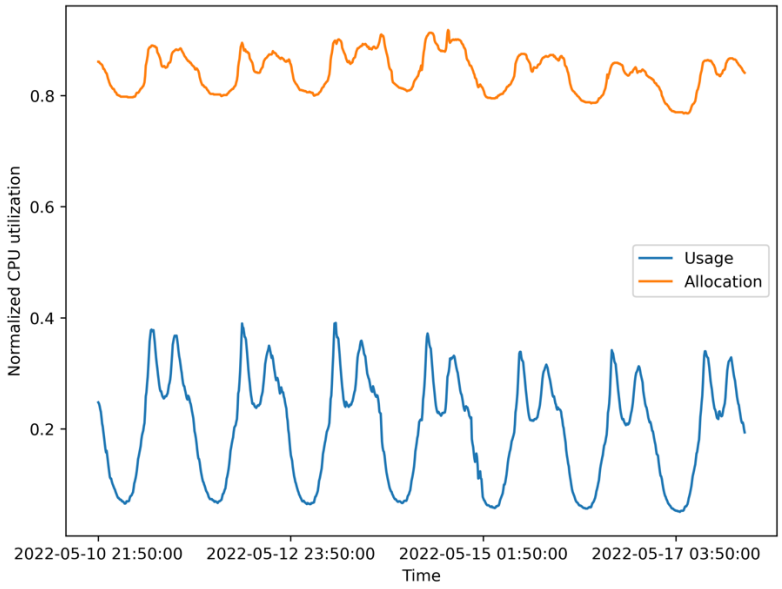

图 **错误!文档中没有指定样式的文字。**.1 某集群一段时间内 CPU 资源使用量与请求量对比图

综:

在研究领域，为了感知应用的实际资源需求，即构建应用资源画像。autopilot **错误!未找到引用源。**采用基于半衰期的移动窗口算法优化机器资源的超卖，他们的方法比起人工设置资源需求，将应用的实际资源使用量与为该应用分配的资源量的差距从 46%缩短到了 23%；peak oracle **错误!未找到引用源。**， Pawel Janus **错误!未找到引用源。**，3Sgima **错误!未找到引用源。**都采用概率分布的方法构建应用资源画像，优化超卖策略，或者是结合 Bin-Packing 算法**错误!未找到引用源。**进一步优化调度策略。随着机器学习，神经网络等技术在学术界研究的深入，一些研究也将这些技术采用到应用资源画像的构建，如 Janardhanan et al.**错误!未找到引用源。**使用 LSTM 预测应用 CPU 使用量的时间序列，Eli Cortez **错误!未找到引用源。**使用 XGBoost 预测应用的资源使用量与生命周期。不过，对于生产落地，企业更愿意采用统计或者其他传统的可解释性强的方法。

除了未实现真正的负载感知外，目前开源的编排系统在调度时考虑的资源维度较少，仅考虑了 CPU Cores，内存容量，硬盘大小等。根据本文的调研与实验验证，在资源受到竞争时对应用性能会产生明显影响的资源包括但不限于：(1)CPU Core。(2)内存子系统资源**错误!未找到引用源。**.**错误!未找到引用源。**，包括 LLC **错误!未找到引用源。**，内存控制器。来源于 Google 的研究表明，在他们的验证的在线服务

中，单独受到 LLC 竞争引起的性能损失最高达 11%，而同时收到 LLC 与内存控制器两类资源竞争引起的性能损失最高达 22%[错误!未找到引用源。]。

关于内存子系统的压力的感知与其对应用性能造成的影响控制方面，如前文所述，开源的编排调度框架完全没有提供这方面的能力。而研究领域，多数工作聚焦在单机层面进行管控，保障机器上高优左右的服务质量。Heracles [错误!未找到引用源。]采用分层资源管理逻辑，根据主机当前内存带宽负载水平，使用 CAT 动态调整离线应用的 LLC Capacity 与 CPU Cores 资源；LIBRA [错误!未找到引用源。]，EMBA [错误!未找到引用源。]，Parties [错误!未找到引用源。]通过应用压测或是历史数据分析的方式构建出应用关于内存子系统资源的回归模型，实时监听应用的性能情况，通过 RDT [错误!未找到引用源。]技术动态调整离线应用可用的内存子系统资源。这些研究工作较为前沿，但是难点在于无法生产落地，多数需要通过压测对应用构建性能关于内存子系统资源的回归模型，这在大规模集群是不可能做到的。同时，RDT 技术对集群机器的 CPU 型号与操作系统有要求，而数据中心机器的更新换代需要较长的时间周期。

然而，在集群调度层面，感知内存子系统资源的相关工作很少，Spread-n-Share [错误!未找到引用源。]在应用前，将应用单独部署到测试机上，构建应用性能与内存子系统资源的线性回归模型，另外，允许用户定义应用的 QoS 目标，同时，通过 RDT 监控机器的内存子系统资源负载，根据节点负载，优先选择内存子系统资源负载较低的节点。

述：

总结下来，资源感知方面主要存在以下问题：

1. 目前开源的编排系统中，无应用资源画像模块，无法感知应用的实际资源需求。
2. 编排系统缺失对内存子系统资源的感知，而这部分资源对应用性能影响不可忽略。
3. 学术界构建应用性能预测模型都需要进行应用压测，这些方法无法满足本文的方法要能够在生产落地的需求。
4. RDT 技术的应用对机器的 CPU 与操作系统有要求，不具备通用性。

1.1.2 集群跨节点资源分配均衡性

综：

在集群调度领域中，主要目的都是为了提升资源利用率与提高应用运行时的 QoS，实现目标的一类方法是提升集群跨节点资源分配的均衡性。对于上文提及的开源算法与用户定义的资源需求不准确两方面因素引起的集群中节点负载不

均衡的问题，有一些研究工作尝试进行了优化，LIN M 错误!未找到引用源。提出了基于蚁群优化算法的策略来平衡集群跨节点资源。LV L 错误!未找到引用源。将容器间的通信负载作为优化目标，在调度层面优化节点的通信负载压力均衡程度。谢雍生错误!未找到引用源。基于 DQN 设计了面向负载均衡的调度优化算法，进一步提升集群的资源利用率，并借助集群资源负载均衡性的提高，更好地保证了服务可靠性。Xiongchao Tang 错误!未找到引用源。将批处理作业的内存带宽、LLC Capacity、CPU Cores 三类资源的需求纳入调度考虑，结合作业短时间压测与加权平均调度策略，进一步提升了集群跨节点的三类资源均衡化提升，最后实现了集群中批处理作业的吞吐量提升。

述：

这些工作都为本文提供了提升集群中应用的可靠性与集群资源利用率的研究思路，即将研究目的转变为提升集群跨节点的负载均衡性。

1.1.3 调度仿真

综：

工业界在调度方面会有不同需求，如提升资源利用率，提升集群负载整体的均衡性，减少高负载节点等。

调度策略的具体实施是一件极其严肃的事情，改动较大的调度策略具体实施到线上通常会需要很长时间的验证，避免出现高风险事件。而调度仿真是件比较困难的事情，困难主要在于：

性能干扰的模拟与复现错误!未找到引用源。：应用混部到同一台机器后，应用之间会产生资源竞争，从而产生性能干扰。计算机体系结构模拟器可通过指令集精细化复原作业的执行过程，但是成本极其高，复原大规模集群的调度几乎不可能。

述：

集群调度体系庞大：无论是中大型公司，还是小型企业，其集群通常涉及的业务数量都不会少；即使进行一次小规模的线下调度仿真可能会涉及的准备工作有启动测试机器，复制线上数据，部署集群，部署业务应用，配置依赖环境等。如此的准备成本，没有公司会愿意投入做一次调度仿真，更多的时候是开发人员仔细检查过后，担着风险将调度策略上线。

综：

目前本文已知的调度仿真工作如:(1)Google 的 peak oracle 错误!未找到引用源。，3sigma 错误!未找到引用源。使用 python 基于应用资源需求与节点资源容量定义，对其调度算法进行模拟。(2)CloudSim 错误!未找到引用源。依据作业的资源需求，无法模拟出作业资源竞争引起的性能干扰。

述：

若采用 Google 的方法进行调度仿真，仅能对调度策略进行验证。无法对整个调度系统进行验证，公司通常需要对整个调度系统体系做一定验证，因为这还包含了对系统调度器，系统所依赖中间件等的验证，并不仅仅是调度策略。所以本文将会实现一个对调度系统端到端的仿真工具，对本文实现的调度策略进行结果验证。