混部集群资源利用分析

葛浙奉[1] 王济伟[1] 蒋从锋[1] 张纪林[1] 俞 俊[1]
林江彬[2] 闫龙川[3] 任祖杰[4] 万 健[5]

[1](杭州电子科技大学计算机学院 杭州 310018)
[2](阿里云计算有限公司 杭州 311121)
[3](国网电力信息通信有限公司 北京 100053)
[4](之江实验室 杭州 311121)
[5](浙江科技学院信息与电子工程学院 杭州 310023)

摘 要 现代互联网数据中心的规模随应用服务需求的增长而越来越大,但数据中心资源利用率低已逐步成为云计算进一步发展的制约因素.为了提高数据中心的资源利用率,云服务提供商将在线服务和离线批处理作业混合部署到同一个生产集群中.但混合部署增加了数据中心管理系统复杂性,对数据中心系统调度和工作负载分配提出了新的挑战.本文从资源使用角度出发,统计分析了阿里巴巴最新发布的包含 4034 台机器长达 8 天的混部集群日志数据集,刻画了机器对离线批处理任务与在线服务容器资源分配策略,以及离线批处理作业与在线服务之间的相互干扰.并根据不同的负载特征,以多种方式对机器进行分类,研究机器分工对集群效率提升的意义.通过分析阿里巴巴集群日志数据集,我们发现:(1)集群中 43.271%的机器存在容器对 CPU 核心"超订"现象,而内存不存在"超订"现象;(2)集群中存在"备用节点",确保集群出现故障时,任务能及时被转移到"备用节点"继续执行;(3)延迟敏感的在线任务的 CPU 利用率较低,但对内存资源的需求比较高,而离线批处理作业的 CPU 利用率较高,在线任务和离线任务的资源占用互补;(4)混合部署显著提高了 CPU 利用率,而内存可能是限制集群性能的主要因素;(5)集群中容器分布存在不平衡性;(6)离线任务的混合部署导致容器内存利用率有所下降,且当在线服务资源需求激增时,调度器缺少一定的容错性和健壮性;(7)离线任务如何部署与容器当前性能紧密相关,尤其是容器 CPI 指标,与离线任务部署呈现显著相关性.本文对集群负载特性、资源使用及离线和在线任务相互干扰进行研究,有助于其他研究人员对集群系统调度和负载分配分析优化,以提高现代数据中心的资源利用率.

关键词 混部集群;资源利用;负载特性;在线服务;批处理作业;调度;服务质量;数据中心
中图法分类号 TP393 **DOI 号** 10.11897/SP.J.1016.2020.01103

Analysis of Resource Utilization of Co-Located Clusters

GE Zhe-Feng[1] WANG Ji-Wei[1] JIANG Cong-Feng[1] ZHANG Ji-Lin[1] YU Jun[1]
LIN Jiang-Bin[2] YAN Long-Chuan[3] REN Zu-Jie[4] WAN Jian[5]

[1](*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou* 310018)
[2](*AliCloud Computing Co. Ltd., Hangzhou* 311121)
[3](*State Grid Electrical Information Communication Co. Ltd., Beijing* 100053)
[4](*Zhejiang Lab, Hangzhou* 311121)
[5](*School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou* 310023)

Abstract The scale of Modern Internet Data Centers (IDCs) is increasing with the growth of application service demand, but the low resources utilization has gradually become a constraint

收稿日期:2019-09-27;在线出版日期:2020-02-23.本课题得到国家重点研发计划项目(2017YFB1010000)、国家自然科学基金面上项目(61972118)、浙江省重点研发计划项目(2017C01SA160069)资助.**葛浙奉**,硕士研究生,主要研究方向为云计算和数据分析.E-mail:gezhefeng@hdu.edu.cn.**王济伟**,硕士研究生,主要研究方向为云计算和数据分析.**蒋从锋**(通信作者),博士,教授,中国计算机学会(CCF)会员,主要研究领域为云计算、系统优化和性能评估.**张纪林**,博士,教授,中国计算机学会(CCF)会员,主要研究领域为高性能计算和分布式系统.**俞 俊**,博士,教授,主要研究领域为机器学习、多媒体分析与图像处理.**林江彬**,硕士,工程师,主要研究方向为分布式存储系统、系统基准测试和可靠性优化.**闫龙川**,硕士,高级工程师,主要研究方向为数据中心能效优化和系统虚拟化.**任祖杰**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为分布式系统和云计算.**万 健**,博士,教授,主要研究领域为云计算、大数据.

factor for the further development of cloud computing, which leads to higher operating costs and power consumption. To improve resources utilization in the data center, cloud service providers co-locate online services and offline batch jobs in the same production cluster. However, co-located deployment increases the complexity of data center management system and challenges the ability of data center system scheduling and workload distribution. Mutual interference, including performance degradation, resource allocation, between online services and offline batch jobs becomes the key concern in workload co-located clusters. To improve IDCs resource utilization with QoS (Quality of Service) guaranteed for online services, it is necessary to coordinate the two different type of workload scheduling and resource allocation more efficiently. Therefore, analysis of workload characteristics, resources utilization, container performance, etc. is crucial to design a better collaboration mechanism between online service and offline batch jobs. This paper, from the perspective of resource utilization, statistically analyzes the newly co-located cluster trace released by Alibaba, which contains various information of 4034 machines for 8 days. And we describe the resources allocation strategy of offline batch jobs and online services and the interaction between them. To better understand the workload characteristics in co-located cluster, machines are classified according to different workload, and resources usage of each type of machine is analyzed in detail. Finally, we explored the impact of the co-locating of offline batch jobs on the performance of online services, as well as some existing problems, by counting three performance indicators of container CPI, MPKI, and memory bandwidth. Through the analysis of Alibaba cluster trace, we found the following: (1) The CPU cores of 43.271% machines in the cluster are "over reserved" by containers, while the memory is not "over reserved"; (2) There are "standby nodes" in the cluster to ensure that tasks can be transferred to "standby nodes" to continue execution in time in case of cluster failure; (3) Latency-sensitive online services have low CPU utilization and high memory demands, while offline batch jobs have high CPU utilization. Online services and offline jobs occupy complementary resources; (4) Co-located deployments significantly improve CPU utilization, while memory might be the primary factor limiting cluster performance; (5) Imbalance phenomenon of container distribution is still existing in the Alibaba co-located cluster, which may cause degradation of fault tolerance and workload balancing; (6) The co-located deployment of offline jobs resulted in a decrease in memory utilization of the container and the scheduler lacks some robustness when the demand for online service resources surged; (7) How to deploy offline jobs is closely related to the current performance of the container, especially the container CPI indicator that can reflect the system delay, significantly related to the deployment of offline jobs. We studied cluster workload characteristics, resource utilization and mutual interference, which are helpful for researchers to analyze and optimize cluster scheduling and workloads distribution, so as to improve the resource utilization of IDCs.

Keywords co-located cluster; resource utilization; workload characterization; online services; batch jobs; scheduling; quality of service; data center

1 引 言

近年来,云计算应用为人类生产、生活的方方面面提供着便利.随着社会对于云计算服务的依赖程度加大,作为云计算服务基础设施的服务器集群面临着前所未有的负载压力.但是,简单地通过购进更多的服务器以扩展集群规模从而减小负载压力的方式会引起更高的运营成本与电力消耗.因此,在云计算产业竞争激烈的今天,如何降低资源成本成为了各云服务提供商面临的首要问题.

集群的负载可以简单分为两类,一类是延时敏

感的在线服务，一旦服务崩溃或者延时过大，将导致用户体验不佳，需要留有较多的资源余量以应对突发的请求高峰；另一类是离线批处理作业，虽然资源消耗较大，但是对延时相对不敏感，可以容忍一定范围内执行时间的调整.

当前一种行之有效的策略是利用在线服务与离线批处理作业的差异性，将二者混合部署（后文简称混部）在同一个生产集群中，通过协调二者调度策略，提高硬件资源利用率. 然而，由于需要协调批处理作业与在线服务的资源分配，混部将增加调度过程的复杂度. 如何高效协调两种负载，在保证 QoS（Quality of Service，服务质量）的基础上提高资源利用率成为了云服务提供商所面临的难题.

目前，主流的大规模云服务提供商都开始了混部机制的尝试，并获得了一定的成效[1-3]. 其中阿里巴巴在其混部集群中成功应对了电商活动中激增的在线负载高峰，证明了其混部机制在实际生产环境中的灵活性与鲁棒性. 为了进一步完善混部机制，提升调度效率，阿里巴巴于 2018 年底公开了业界一份真实混部集群的跟踪数据集（Cluster-Trace-v2018，后文简称为 CT18），期望吸纳业界与学术界的优化建议. CT18 记录了 8 天时间内 4034 台机器的运行情况，具有较高的研究价值.

本文聚焦混部集群的资源使用情况，通过提取 CT18 中资源相关数据项，分析了离线任务与在线任务对资源的使用情况，以及两者的相互影响. 比较不同年份数据集间资源“超订”现象（任务预订资源总量大于实际物理资源），发现不同的集群调度策略发生了改变. 根据实例调度次数的分布，研究了机器分工现象，以热力图形式表现了机器的资源利用情况，并探明了“备用节点”存在的意义. 同时，根据负载类型将机器分类，比较混部与非混部情况下的运行情况，指出混部集群相较于传统非混部集群资源使用效率大幅提升. 另外，通过探讨在线任务和离线任务混部后之间的相互影响，发现离线任务的混部导致容器内存利用率有所下降，且当在线服务资源需求激增时，调度器缺少一定的容错性和健壮性，离线任务仍旧不断被部署运行，同时也增加了离线任务失败数量，导致问题不断恶化. 另外，容器部署存在不平衡性. 本文研究还发现离线任务的部署与容器当前性能紧密相关，尤其是容器 CPI 指标，与任务部署呈现显著相关性.

本文第 2 节介绍阿里巴巴混部 CMS（Cluster Management System，集群管理系统）架构与 CT18 数据集中资源利用相关的项目；第 3 节给出计划资源的特征与“超订”现象的分析；第 4 节分析不同类别机器的硬件资源利用特征；第 5 节探讨在线任务和离线任务之间的相互影响；第 6 节是相关工作；第 7 节总结本文的工作并提出未来的研究展望.

2　背　景

阿里巴巴于 2015 年开始混部集群测试，并逐年提高混部集群的比例以提高资源利用率，其背后的混部 CMS 也不断完善以适应更为严苛的使用场景. 本节简要介绍阿里巴巴混部 CMS 的组成架构，并重点说明 CT18 数据集内容，为后续进一步分析奠定基础.

2.1　阿里巴巴混部 CMS

阿里巴巴混部 CMS 由 Sigma 和 Fuxi 调度器协同调度在线和离线任务：

（1）Sigma 调度器由 SigmaMaster、SigmaAgent 和 PouchContainer 三个组件组成.

SigmaMaster：为集群中物理机的容器部署进行资源调度分配和算法优化.

SigmaAgent：每台物理机上运行一个 SigmaAgent 进程，向 SigmaMaster 报告物理机状态信息，以及保证运行在物理机上的容器的性能.

PouchContainer：运行不同类型的在线服务.

（2）Fuxi 调度器遵循主从架构，由 FuxiMaster、FuxiAgent 和 ApplicationMaster 组成.

FuxiMaster：收集集群中各物理机资源使用信息，快速公平地调度用户作业至相应节点，并回收空余资源.

FuxiAgent：每台物理机上将运行一个 FuxiAgent 进程，主要用于向 FuxiMaster 报告物理机状态信息和监控、保护、隔离应用程序执行.

ApplicationMaster：表示不同的计算范式（如 MapReduce、Spark 或者 Storm 等）.

在线服务属于长生命周期、延时敏感的任务，而离线任务主要是计算密集型、生命周期短且时延不敏感的任务，在线任务调度要求低延时性，离线任务调度要求高并发率和吞吐率. 针对这两种任务不同的需求，在混部架构上将两种调度并行处理，即一台物理机上既有 Sigma 调度器又有 Fuxi 调度器，并有各自的资源池. Sigma 调度通过启动 Pouch 容器运行在线服务，Fuxi 调度器也启动离线计算任务，争

抢此台物理机上的空余资源，以期望提高物理机资源利用率. Level 0 作为控制器协调 Sigma 和 Fuxi 调度器的资源分配问题. 阿里巴巴混部 CMS 的架构图如图 1 所示.

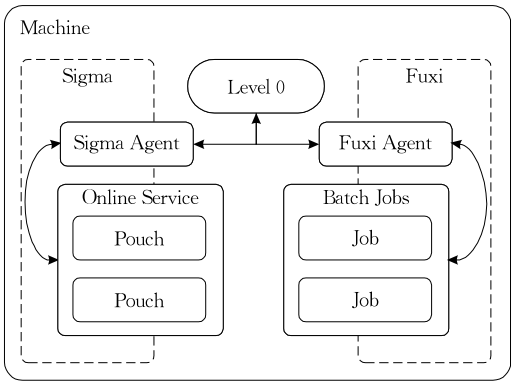


图 1 阿里巴巴混部 CMS 架构

为保证在线服务的 QoS，一种常见的方法是为在线服务预留充足的资源，Sigma 调度器分配给容器足够的资源. 而在实际运行环境中，在线服务大部分时间无法充分利用所分配资源，此时 Fuxi 调度器利用容器中空闲资源部署离线批处理作业. 而当容器资源使用率升高时，调度器能暂停甚至杀死正在运行中的离线任务，收回资源给容器使用，以满足在线服务 QoS.

在线服务和离线任务的混合部署，必然导致资源争抢，在一定程度上会影响在线服务的响应时间，所以就如何混部及混部比例问题有必要进一步探讨.

2.2 跟踪数据集概览

阿里巴巴于 2017 年和 2018 年相继发布了混部集群跟踪数据集 cluster-trace-v2017 和 cluster-trace-v2018. 不同于阿里巴巴 2017 年公布的混部集群数据集，CT18 中的机器在配置上基本是同构的，每台机器拥有相同的 CPU 数量(96 核)和内存容量(因阿里巴巴方面保密需要，内存容量参照内存最大值进行了归一化处理). 通过将阿里巴巴在 2017 年与 2018 年发布的混部数据集进行对比(见表 1)，发现主要有两点不同：(1) 每台物理机器的 CPU 核心数发生了变化；(2) 内存容量由原来的大小不一变为完全一致.

表 1 2017 年与 2018 年公开的阿里巴巴混部集群跟踪中机器配置的对比

数据集	机器数量	CPU核心数(每台)	内存容量(每台)
cluster-trace-v2017	1313	64	57.5%～100%
cluster-trace-v2018	4034	96	100%

我们可以发现：(1) cluster-trace-v2017 和 cluster-trace-v2018 数据采集自不同批次的物理机器；(2) 物理机器的硬件配置发生变化，集群中机器配置也由异构转化为同构. 阿里巴巴混部集群任务调度和资源分配方案可能发生变化，所以有进一步研究的必要.

由于商业机密，云计算服务提供商很少将实际生产环境中的跟踪数据集公布以供业界和学术界研究. 谷歌于 2012 年公布了机器数量超过 12500 台长达 29 天的异构跟踪数据集(clusterdata-2011-2)，其采用目前业界最为成熟的大规模混部调度工具 Borg 作为集群管理系统；在 2017 年，微软也公布了部署在微软 Azure 云平台上 200 多万台虚拟机长达一个月的工作负载跟踪数据集. 相比谷歌 Borg 集群管理系统采用名为 cell 的中间层架构管理集群，阿里巴巴混部集群采用双调度器协同的方式管理集群，在集群调度模式上有一定区别. 且谷歌公布的数据集已经相对久远，可能无法反映当前集群管理系统的功能和优点；而微软公布的数据集只包含虚拟机的负载信息，对集群混合部署参考意义不大.

本文的关注点是集群的资源使用情况，故将 CT18 中资源相关的数据项单独列出，并在表 2 中展示. CT18 的描述实体为机器、容器、批处理作业，其中资源主要指 CPU、内存、网络和磁盘 IO 等硬件资源.

表 2 CT18 中资源相关数据项

表格	数据项名称	数据项含义
machine_meta.csv	*cpu_num*	机器的 CPU 核心数
	mem_size	机器内存容量
machine_usage.csv	*cpu_util_percent*	机器的 CPU 使用率
	mem_util_percent	机器的内存使用率
	mem_gps	内存带宽
	net_in	接收数据包数量
	net_out	发送数据包数量
	disk_io_percent	磁盘读写 IO
container_meta.csv	*cpu_request*	容器计划 CPU 数量
	cpu_limit	容器可用 CPU 上限
	mem_size	容器计划内存容量
container_usage.csv	*与 machine_usage 数据项类似，不再赘述	
batch_task.csv	*plan_cpu*	计划 CPU 核心数
	plan_mem	计划内存容量
batch_instance.csv	*cpu_avg*	实例平均 CPU 使用量
	cpu_max	实例最大 CPU 使用量
	mem_avg	实例平均内存使用量
	mem_max	实例最大内存使用量

在表 2 中深色背景的部分为描述配置信息的数据项，这些数据项是预先设置完毕的静态属性，在同

一个实体中，这些数据项的数值随着时间的推进基本没有发生变化. 而浅色背景部分则为描述运行时资源占用情况的数据项，在大部分情况下，不同采样点之间的数值存在较大差异.

此外，在混部 CMS 中，容器与批处理作业都遵循自身的层次架构. 每个容器都从属于一个部署组(deploy group)，同一个部署组下的容器都为同一个应用服务提供支持. 而对于批处理作业，则存在“job-task-instance”形式的三层架构，一个作业(job)至少包含一个任务(task)，而一个任务下，至少包含一个实例(instance)，如图 2 所示.

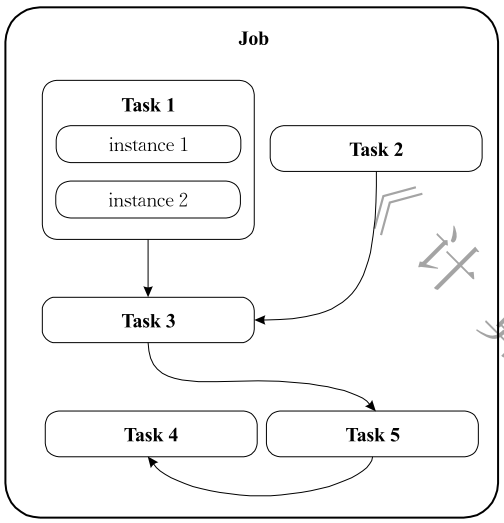


图 2　批处理作业的“job-task-instance”层次架构

相较于 cluster-trace-v2017，CT18 在机器数量与时长上拥有更大的规模. 作为一项新特性，DAG(Directed Acyclic Graph，有向无环图)结构被引入用以表达任务之间的依赖结构[4]，在图 2 中以黑色箭头表示. 任务必须等待其依赖的任务完成后，才能进行处理. 利用该特性，研究人员可以研究真实调度环境中任务间依赖对于资源分配策略的影响.

在后面的章节中，我们将研究在线服务与批处理作业的资源分配机制对于机器实际资源消耗量的影响.

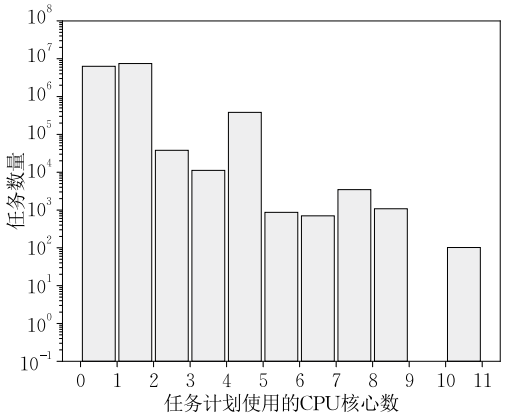
3　资源预订

本节对 CT18 中离线任务和在线容器对资源预订进行了统计分析，并阐述了资源“超订”的现象.

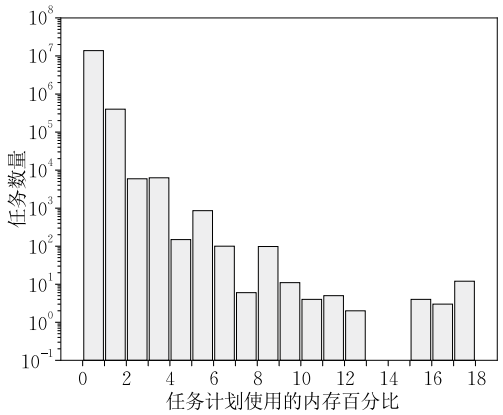
3.1　离线任务资源预订

batch_task 表中字段 *plan_cpu*、*plan_mem* 表示该任务下每个实例的 CPU/内存资源的计划需求量. 据此，我们对离线任务的 CPU/内存资源预订情况进行统计，其结果见图 3，横坐标为单个任务计划 CPU 核心数或内存使用情况，纵坐标为任务的数量. 为了表达上的直观性，我们以长度为 1 的区间聚合任务，作图时每个条形图取左闭右开区间. 例如，在图 3(a)中，计划预订 0.05 个核的任务记录被聚合在[0,1)区间上的柱形中；计划预订 5 个核的任务记录被聚合在[5,6)区间的柱形中. 图 3(b)中内存预订情况的作图方式与(a)保持一致.

从图 3(a)中，我们可以看出任务计划 CPU 核数大部分集中在较低的区间. (b)中的内存预订情况与(a)中的 CPU 核数相比，分布集中在更低的区间. 二者都在其较高计划资源的位置存在一个空缺区间，(a)是在[9,10)上，(b)是在[13,15)上. 总体来说，任务下每个实例的计划资源预订，大部分是轻量的，且 CPU 资源的计划预订量相较于内存更大.



(a) CPU资源预订



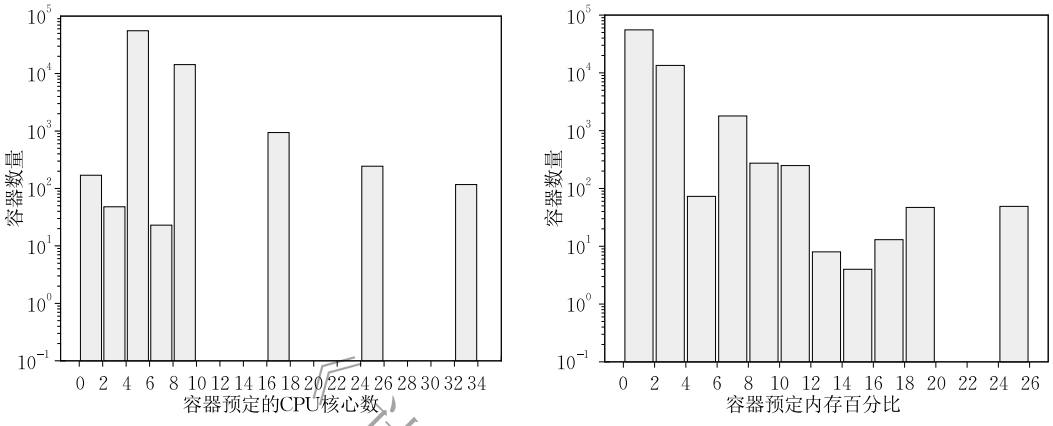
(b) 内存资源预订

图 3　任务计划资源统计(每个条形图通过左闭右开刻度内的数值通过求和聚合)

CT18 中同一个任务下的实例具有相同计划资源，所以在资源调度时可以省去很多操作上的复杂性. 经过统计，约 90% 的任务中含有的实例数量是小于等于 10 的. 在该数量级的实例上实现更细粒度的计划资源分配，可能并不会大幅增加调度的复杂性，但能使整体资源分配更加精细合理.

3.2 在线任务资源预订

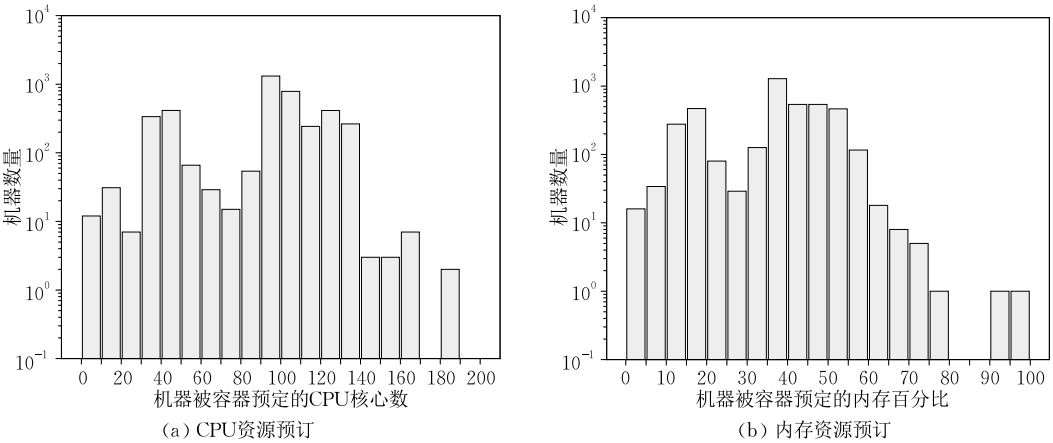
接下来，我们根据表 container_meta 对容器的资源预订情况作了统计分析，以相同的聚合方式作图 4，横坐标为容器预订资源情况，纵坐标为容器数量. 与离线任务资源预订类似，在线任务资源预订的分布集中在低区间；不同的是，容器对 CPU 核心数预订的类别更少，只有 7 种预订 CPU 核心数的情况.



(a) CPU资源预订 (b) 内存资源预订

图 4 单个容器资源预订情况

由于容器在机器上是持续运行的，我们将容器的资源预订情况聚合到对应机器上，进一步研究机器上资源被容器预订的情况，见图 5，横坐标为单台机器上资源被预订情况，纵坐标为机器数量. 我们已知每台机器的 CPU 核心数是 96，观察图 5(a)可发现一部分机器上被预订的 CPU 核心数超过了机器的物理核心数，我们称之为“超订(Over Reserved)”. 通过具体的数值计算，CPU 核心被“超订”的机器数量达到机器总数的 43.271%(1733/4005). 我们再比较图 5(b)中机器内存被预订的情况，发现每一个柱体都在 100 刻度内，即内存没有发生“超订”现象. 另外，CT18 中的“超订”现象与 Liu 等人[5] 在 cluster-trace-v2017 中所描述的恰好相反. 在 Liu 的文章中，机器上的内存资源出现了很明显的“超订”现象，而被预订的 CPU 核心数却没有超过机器上的物理核心数. 导致这一现象的原因可能是：CPU 计算能力发展速度要显著高于内存容量的增长速度，内存逐步成为集群性能瓶颈，对内存资源“超订”将带来更大的不稳定性，而 CPU 资源仍有余量，对 CPU 资源“超订”能提高资源利用率.



(a) CPU资源预订 (b) 内存资源预订

图 5 单台机器资源被容器预订情况

虽然环境与调度机制发生了一定的变化，但是“超订”的现象依然存在.“超订”作为一种预先的资源承诺，超过了物理资源的总量，能够让一台机器上部署的容器数量更多，提高资源的利用效率.但是“超订”会增加调度上的难度，当一台机器上的容器在某一时刻总的实际资源需求量超过了该机器物理资源总量，将造成在线服务的 QoS 下降.我们将在后续章节中，对运行过程中的实际资源使用量进行分析.

4　资源使用

本节将对集群运行时资源使用情况进行统计分析.在离线批处理作业运行时的特征探究中，我们根据每台机器上实例调度次数对混部集群中的机器进行分类，并根据 8 天内每类机器的 CPU 和内存资源使用率绘制热力图，观察离线任务负载特征.本节随后根据机器的离/在线任务运行与否将机器分为只运行在线任务、只运行离线任务、同时运行在线离线任务和不运行任何任务 4 类，并对每类机器的 CPU、内存、磁盘 IO 和网络 IO 资源进行统计分析.

4.1　离线批处理作业在运行时的特征探究

离线批处理作业的调度操作，在本质上是通过实例调度实现的.在图 6 中，通过将每台机器上接受实例调度的次数按升序排序，我们获得了 CT18 中所有机器接受实例调度次数的分布.

值得注意的是，将图 6 进行局部放大，我们发现分布曲线在一定范围内具有阶梯状的抬升.结合具体数据，根据实例调度次数对机器进行初步分类，我们得出了导致该现象的原因，见表 3.

表 3　根据实例调度次数对机器进行分类

机器	数量	调度次数下界	调度次数上界
类别 A	167	0	0
类别 B	16	6	26400
类别 C	76	57778	87323
类别 D	3775	140461	493865

通过观察表 3 可以发现，不同类别的机器接受调度次数存在较大间隔(即前一个类别的机器接受调度次数的上界与后一个类别接受调度次数的下界之间具有较大差值)，在图 6 中体现为阶梯式的曲线.

为了揭示同构机器上的实例调度次数出现的不平衡性，我们将上述分类方式下机器的 CPU 和内存资源使用率作热力图以进行探究，见图 7(使用率的高低参考每个图右边的渐变色条，白色区域代表数据空缺)，横坐标为时间戳，单位为 s，时间区间在[0，691200]，表示第 1 天至第 8 天.

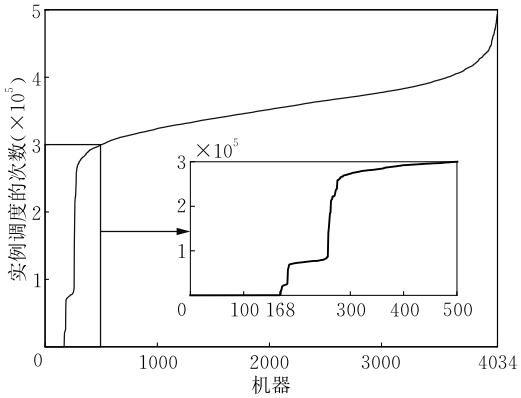
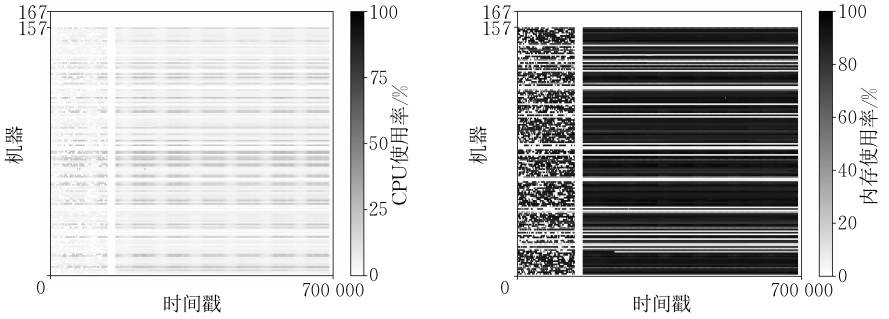


图 6　实例在全部物理机器上的调度次数分布图

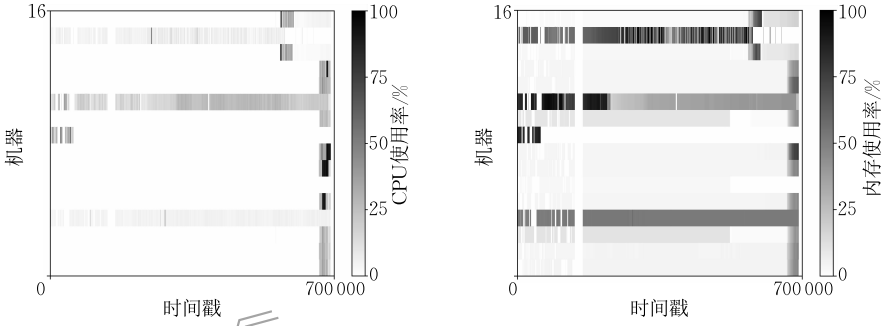
观察图 7，可以发现每幅图在第 2 天中都有一个明显的中断，并且中断之前的数据空缺较多.我们认为，集群管理系统的日志记录策略发生了改变，导致第 1 天内对实例调度记录缺失较多.同时，我们从每类机器中抽出较为典型的一台机器，绘制其 CPU 与内存的变化曲线图，见图 8.结合图 7 和图 8，我们可以看出每个分类下的 CPU 与内存资源使用情况显现出明显的差异：

类别 A 机器的 CPU 使用率偏低，但是呈现以天为周期的波动，内存使用率则普遍偏高.我们推测 A 类机器运行的负载主要为在线任务，在线任务在大部分时间中对 CPU 资源的消耗较少，但是对内存资源消耗较大.并且，在线任务对数据中心造成的压力是具有一定周期性的，而这个压力的变化主要体现在 CPU 的使用率上.这一类机器完全避免了离线批处理作业的部署，可以保持 CPU 有充足的余量，保证在线服务在峰值时依旧保持服务质量.

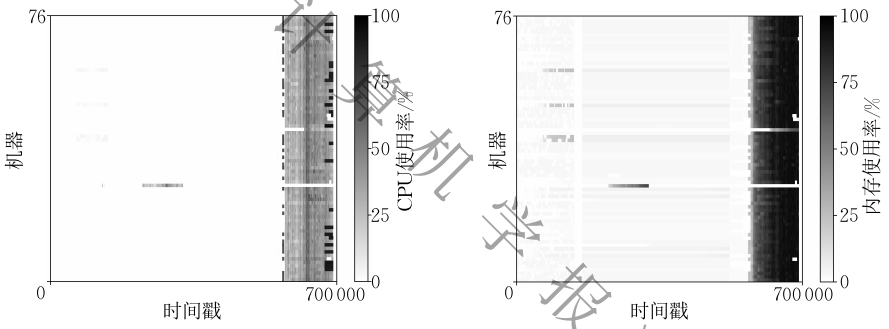
类别 B 和类别 C 机器的 CPU 和内存使用率较为相似.在类别 B 机器中，CPU 与内存使用率普遍偏低，但在最后半天有所升高.在类别 C 的机器中，前 6 天机器几乎空转，第 7 天出现大量记录空缺，从第 8 天开始使用率逐步提升至满载.该现象在图 8 中表现为类型 B 和类型 C 机器在前 7 天曲线贴近于 0，在最后一天曲线明显抬升.我们推断，集群上存在部分机器作为“备用节点”，在集群运行正常情况下处于空闲状态，保持了较低的实例调度次数.确



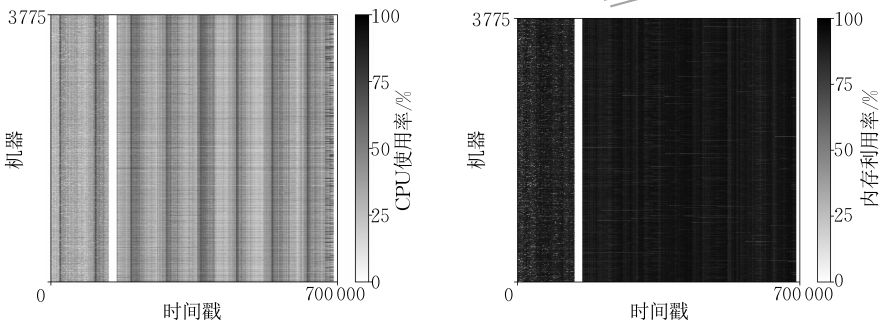
(a) 类别A



(b) 类别B



(c) 类别C



(d) 类别D

图 7 根据实例调度次数分类的各类机器的 CPU 与内存使用情况

保集群中出现故障时,不能继续工作的节点上的任务能够转移到“备用节点”上继续工作,避免服务质量的下降,提高集群的可靠性.

类别 D 包含了数据集中 93.58%的机器. 该类机器的资源使用特征代表了混部集群常规机器的资源利用率:CPU 使用率呈现出明显的以天为周期的变化,内存使用率则持续保持接近满载状态. 类别 D 机器的 CPU 与内存使用率较类别 A、B、C 机器更高,且机器数量占混部集群较高比例,可以作为混部集群调度策略优化的理想参考样本.

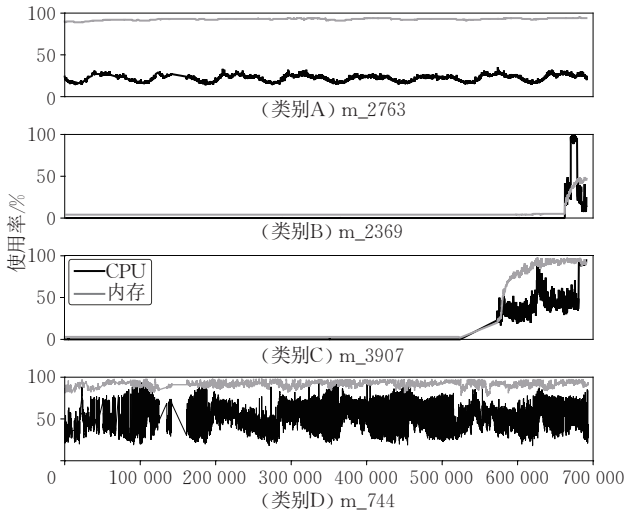


图 8　根据实例调度次数分类的各类机器抽出样例的 CPU 与内存使用情况

4.2　机器资源使用统计

上一小节我们对离线批处理作业在机器上的调度分布进行了探究，发现其中有 167 台机器没有运行离线批处理作业，于是进一步统计了机器上运行在线任务和离线任务的情况.

在表 4 中，我们对 4034 台机器根据运行任务类型分为只运行在线任务、只运行离线任务、同时运行在线与离线任务和不运行任何任务这四类，并统计了各类的数量和占比. 其中集群中同时运行在线和离线任务的机器数量最多，占总数的 95.612%.

表 4　机器运行在线任务、离线任务情况统计

机器类型	数量	占比/%	描述
类型 1(M_Container)	148	3.669	只运行在线任务
类型 2(M_Batch)	10	0.248	只运行离线任务
类型 3(M_CB)	3857	95.612	运行在线和离线任务
类型 4(M_Idle)	19	0.471	不运行任何任务

在数据集的表 machine_usage 中记录了各台机器不同时刻下的资源利用率，我们统计了表 4 中 4 类机器的资源使用情况. 由于部分机器在较长时间段内没有资源使用情况记录，为了反映 8 天时间内机器的资源使用情况，我们将每 5 分钟区间内的资源利用率取均值聚合作图 9(剔除时间戳为 0 的无效记录). 此外，我们还统计了 4 种类型机器的资源利用率情况，结果如表 5 所示.

表 5　根据是否运行在线/离线任务分类的 4 种类型机器的利用率统计

	利用率范围/%					平均利用率/%				
	CPU	内存	磁盘 IO	net_in	net_out	CPU	内存	磁盘 IO	net_in	net_out
类型 1(M_Container)	4.66～18.17	72.83～97.0	1.28～5.49	18.25～31.55	15.34～29.63	7.51	81.48	1.97	21.55	18.38
类型 2(M_Batch)	1.91～77.6	19.77～95.65	0.43～33.1	1.99～39.12	0.01～26.56	26.68	77.36	4.92	29.10	19.81
类型 3(M_CB)	16.97～81.42	78.07～95.09	3.62～27.15	35.45～48.54	28.03～38.57	39.28	88.56	7.98	41.92	33.19
类型 4(M_Idle)	0.02～1.41	3.27～9.16	0.01～2.42	0.01～0.08	0.01～0.1	0.89	6.57	0.16	0.05	0.06

图 9 中横坐标为时间戳(时间区间(0,691 200]，单位为 s)，纵坐标为各类型机器的平均资源使用率，子图(a)、(b)、(c)、(d)分别表示 CPU、内存、磁盘 IO、网络 IO 的利用率，每张子图中分别作了 4 种类型机器的资源使用曲线图. 由于 machine_usage 表中缺失时间戳在[141 160，160 370]区间内的记录，所以将该时间段上各类机器的资源利用率暂设为 0.

4.2.1　CPU 利用率

观察图 9(a)所示 CPU 利用率情况，我们发现：(1)除了 M_Idle 机器(不运行任何任务的机器)的 CPU 利用率没有明显规律性外，另外三种类型机器的 CPU 利用率呈现明显的以天为周期的波动；(2)M_Batch(只运行离线任务的机器)和 M_CB(同时运行在线和离线任务的机器)的 CPU 利用率呈正相关，其相关系数为 0.877，而 M_Container(运行在线任务的机器)和 M_CB 机器的 CPU 利用率呈弱负相关，其相关系数为−0.095.

在线服务的高峰期在白天段，而离线任务为了避免争抢有限的资源，主要在凌晨部署，图中体现为只运行在线任务机器的 CPU 利用率与只运行离线任务机器的 CPU 利用率波峰是错开的. 所以在不影响在线任务性能的情况下，将离线任务与在线任务混部在同一台机器上，可以提高机器的总体 CPU 利用率.

根据表 5 可以发现，只运行离线任务的机器其 CPU 利用率的变化范围比只运行在线任务的机器大得多，且只运行离线任务的机器的平均 CPU 利用率较只运行在线任务的机器更高. 所以我们推断，离线任务对 CPU 的需求更大，CPU 利用率受离线任务影响变化较大，而且 M_Batch 机器与 M_CB 机器的 CPU 利用率呈高相关性，在阿里巴巴混部集群的 CPU 利用率变化中，离线任务的影响相较在线任务更具主导性.

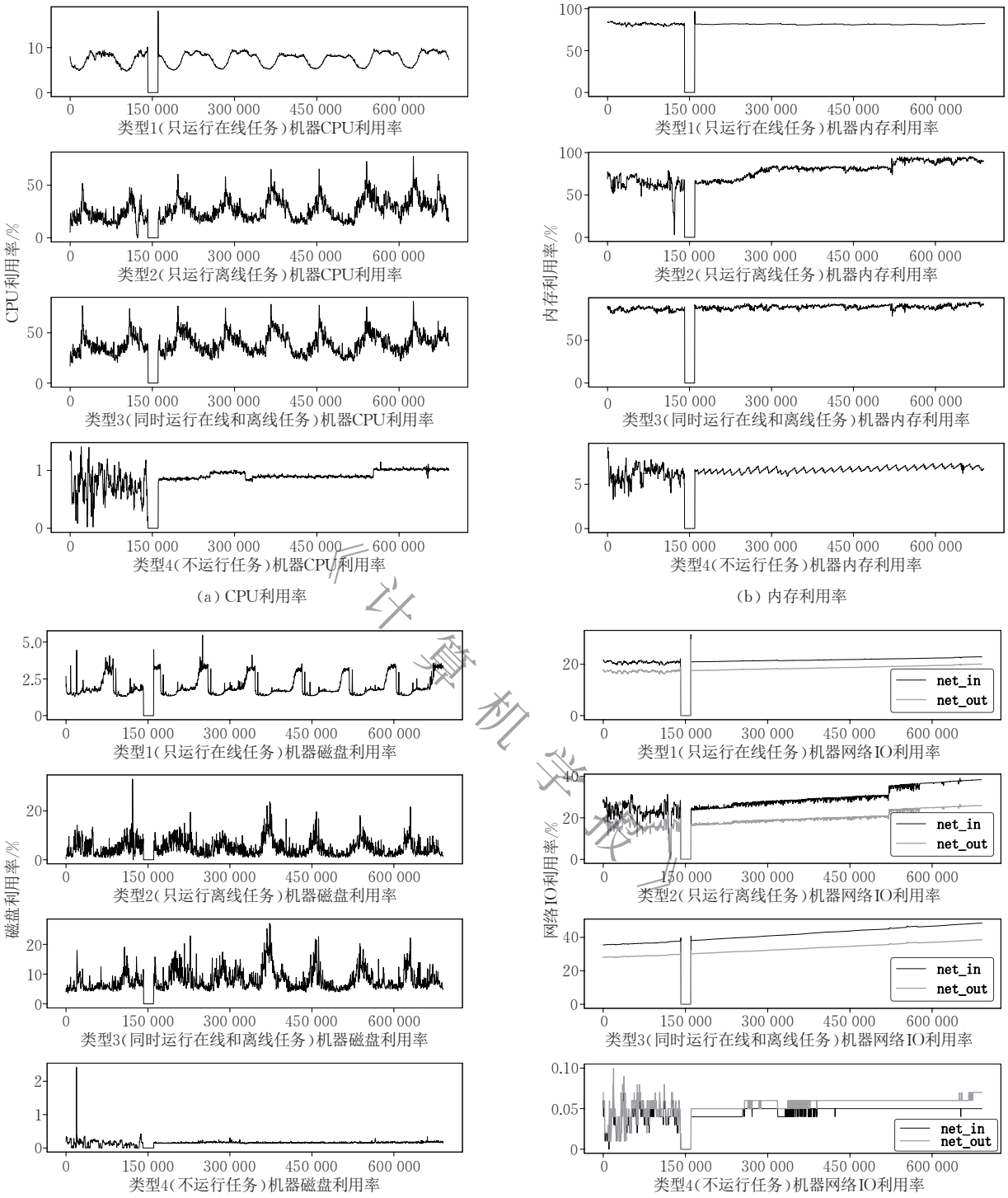


图 9 根据是否运行在线/离线任务分类的 4 种类型机器 8 天内的资源利用率随时间变化情况

4.2.2 内存利用率

观察图 9(b)所示内存利用率情况,我们发现:与 CPU 利用率相反,4 种类型机器的内存利用率都没有呈现一定的周期性;M_Container 机器的内存利用率变化相对平稳,而 M_Batch 机器的内存利用率可能由于批处理任务的不同而变化较大.

可以发现只运行在线任务的机器的内存利用率比较高,如果去掉最高 97%的这个点(时间戳在[141160,160370]区间内数据缺失,记录恢复后,时间戳在[160380,160530]只有 m_2178 一台机器有记录,且此台机器的内存利用率较高,导致波形有突起现象,所以去除该点能更好反映只运行在线任务

机器的内存使用的总体情况)，其内存利用率的范围为 72.83%～85.27%，比只运行离线任务的机器变化(19.77%～95.65%)要小得多，且 M_Container 机器的平均内存利用率为 81.48%，比 M_Batch 机器的平均内存利用率(77.36%)高.说明在线任务对内存的需求比离线任务高，且内存利用率相对稳定，所以内存可能是限制在线任务性能的主要原因.

据我们所知，在阿里云中在线服务大部分为 Java 程序，而运行这些 Java 程序需要启动多个独立的 Java 虚拟机(JVM).JVM 占用了一些潜在可被使用而未被释放的空闲内存空间，而且多个 JVM 的垃圾回收器的相互干扰进一步恶化了内存占用[6].目前阿里云中内存管理机制无法有效回收内存资源，这可能是阿里云混部集群内存资源使用率高的主要原因.所以需要设计更高效精细化的内存管理机制，从有限的内存资源中挖掘潜在可用的资源.

4.2.3　磁盘 IO 利用率

观察图 9(c)磁盘 IO 利用率情况，可以发现磁盘 IO 的使用情况呈一定的周期性，且运行在线任务的机器与运行离线任务的机器的周期性呈负相关性，即在线任务磁盘 IO 利用率高的时候离线任务的磁盘 IO 利用率低.

根据表 5 可以发现，只运行在线任务的机器的磁盘 IO 利用率相对较低，且变化跨度较小，相对趋于稳定.各类型机器对磁盘 IO 的利用率都不高，我们可以断定磁盘 IO 利用率不是影响性能的主要因素.

4.2.4　网络 IO 利用率

观察图 9(d)网络 IO 利用率情况，黑线表示接收网络数据包(net_in)，灰色表示发送网络数据包(net_out)，可以发现类型 1、2、3 机器的网络 IO 利用率都相对比较平稳，且 net_in 的利用率高于 net_out.值得注意的是，随着时间的增长，网络 IO 利用率逐渐增大，其中 M_Container 机器的网络 IO 利用率增长相对较少，而 M_Batch 机器的网络 IO 利用率增长幅度较大.

根据表 5，只运行在线任务的机器和只运行离线任务的机器的网络 IO 平均利用率相差不大，但离线任务的网络 IO 利用的范围相对较大，且可以明显观察到只运行离线任务的机器时间戳在 150000 s 之前的网络 IO 利用率变化波动比较大，且在 123000s(1.42 天)左右只运行离线任务的机器的网络 IO 利用率突然降低，后又上升回归正常，可能由于这段时间内的离线批处理作业比较少，导致网络 IO 波动较大.

根据上述对各类型机器的 CPU、内存、磁盘和网络资源使用情况的分析，可以得出结论：延迟敏感的在线任务的 CPU 利用率较低，而对内存资源的需求比较高.为了提高集群中总体的资源利用率，在不影响在线任务性能的前提下，可以在运行在线任务的机器上部署离线任务，从而实现在线任务和离线任务的资源占用互补，达到提高资源利用率的效果.此外，在线任务对内存使用比较紧张，所以想要提高在线任务的性能，可能需要提供更多的内存资源或优化程序设计.相较 CPU 和内存，机器的磁盘 IO 和网络 IO 利用率未到达瓶颈，暂时可以不用考虑优化.如果 M_Container 机器的平均资源利用率作为非混部集群(即只提供在线服务)的资源利用率，M_CB 机器的平均资源利用率作为混部集群的资源利用率，则可得到表 6.

表 6　传统非混部集群与混部集群资源利用率比较

	CPU/%	内存/%	磁盘 IO/%	net_in/%	net_out/%
非混部集群	7.51	81.48	1.97	21.55	18.38
混部集群	39.28	88.56	7.98	41.92	33.19
资源利用率提高百分比	423.04	8.69	305.08	94.52	80.58

从表 6 可以发现，通过将在线任务和离线任务混合部署的方式，数据中心资源使用率得到明显提升，尤其是 CPU 和磁盘 IO 的资源使用率分别提高了 423.04%和 305.08%，网络 IO 也提高了相当高的资源利用率，而内存利用率由于内存资源有限，只提高了 8.69%，这也佐证了内存资源是限制集群性能的一个重要因素.

5　混部影响

本节将探讨混部策略实施后，在线任务和离线任务之间的相互影响，尤其是离线批处理作业的加入对在线任务 QoS 的影响.

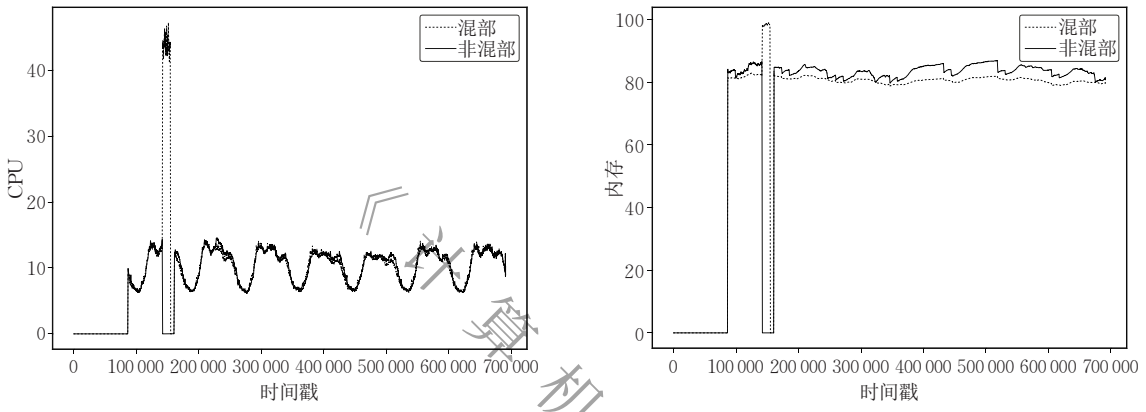
5.1　混部后对容器资源使用影响

在上一节中已经根据运行任务类型将机器分为只运行在线任务、只运行离线任务、同时运行在线与离线任务和不运行任何任务四类，则机器上运行的容器可分为两类，即混部机器上的容器(container_mixed)和非混部机器上的容器(container_unmixed).阿里巴巴发布的数据集中表 container_usage 记录了两类容器运行时资源使用和性能变化.表 7 为容器的平均使用统计信息，需要注意的是表 container_usage 中 *cpi*、*mem_gps*、*mpki* 等字段记录存在大量缺失.

表 7 容器使用统计

	数量	CPU 利用率/%	内存利用率/%	CPI	Mem_GPS	MPKI	net_in/%	net_out/%	磁盘 IO/%
Container_mixed	65945	10.953	80.307	1.616	0.119	1.075	0.215	0.215	9.096
Container_unmixed	1831	10.061	80.911	1.573	0.707	1.057	0.243	0.262	2.380

同样,我们统计了容器在 8 天时间内的资源使用情况,图 10 为容器 CPU/内存资源使用情况,由于这 8 天中的第一天(时间区间[0,86 400])缺失容器运行记录,所以有效时间戳范围在[86 400,691 200].图中虚线表示混部机器中的容器,实线表示非混部机器中的容器.可以发现混部机器中容器的 CPU 利用率和非混部机器中的容器保持一致,而混部机器中容器的内存利用率低于非混部机器中的容器.由于在线服务对计算需求不大,Sigma 调度器分配给容器的 CPU 资源有很多空闲(容器约只使用 10%CPU 资源),所以离线任务的混部对容器的 CPU 利用影响不大.但在线服务对内存消耗极大,离线任务的混部一定程度上抢占了容器的内存资源,导致容器内存利用率有所下降.

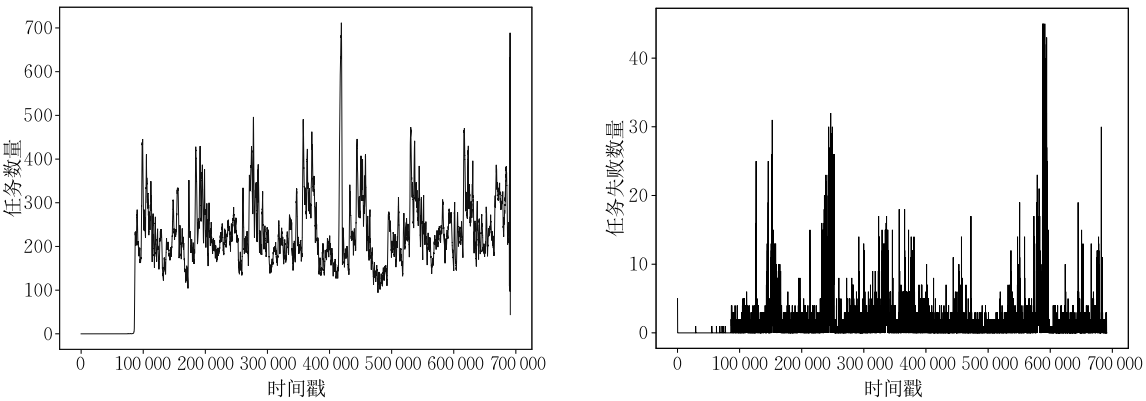


(a) CPU资源使用 (b) 内存资源使用

图 10 容器资源使用情况

值得注意的是,时间戳在[141 610,154 630]附近混部机器中容器 CPU 和内存资源利用率突增,CPU 利用率超过 40%,内存几近满载,而非混部机器中容器的 CPU 和内存利用率记录为 0.表明由于内部或者外部原因导致集群中混部机器出现故障(也可能由于机器新加入生产集群但未可用),可推测离线任务的混部使得集群调度系统变得更加复杂,导致集群内机器利用率和任务分布出现不平衡.这段时间内集群可能出现问题,导致日志服务也无法工作,这解释了图 7、图 9 中记录存在空缺的原因.

为了探究这段时间内集群可能的故障与离线任务混部之间的关系,我们在图 11 作离线任务随时间部署/失败变化曲线.从图 11(a)可以发现,任务在集群中部署呈明显日周期变化,离线任务部署数量于每日早上 4 时达到峰值,白天为了满足在线服务



(a) 任务部署 (b) 任务失败分布

图 11 离线任务随时间部署/失败变化曲线

需求，部署离线任务较少. 理想情况下，当混部集群中容器(在线服务)对 CPU、内存等资源需求激增时，Fuxi 调度器应该根据一定的策略减少离线计算任务的部署，甚至在必要时强制停止部分正在进行的计算任务以释放其占用的资源，从而优先保证在线服务的质量. 而在时间段[141 610，154 630]内，任务的部署数量并没有由于系统出现问题(混部机器下容器资源高负荷使用)而做出相应调整. 且从图 11(b)可看出在此段时间内任务失败数量有明显增加，说明容器的资源高负荷使用引起了任务失败数量的增加. 可见当混部集群出现问题时，当前混部集群中的调度器缺少一定的容错性和健壮性，无法在系统出现问题时，及时做出优化调整. 上述现象表明当前混部集群的任务协调机制仍有一定的改善空间. 针对这个问题，需要更好地协调 Sigma 和 Fuxi 调度器工作，Sigma 调度器实时监控所管理容器的运行状态，当出现异常时，需要通过 Level 0 及时通知 Fuxi 调度器做出相应调整.

5.2　容器部署不平衡性

阿里巴巴以电商平台为主营业务，其在线服务包括商品浏览、订单查询、在线支付等，这些服务对系统响应要求较高. 不同的在线服务对应不同的应用分组(Application Group)，在数据集 cluster-trace-v2018 中用字段 *app_du* 加以区分. 而一个在线服务被部署在多个容器中，即一个应用分组中含有多个容器以支持在线服务，且这些容器分布于不同的物理机中. 在 CT18 中，总共有 9790 个应用分组，其中包含容器最多的应用分组有 629 个容器，最少的只有 1 个容器. 另外，一个应用分组最多被部署在 444 台物理机器上. 虽然将相同应用分组中的容器分布于不同的物理机上，可能会带来数据传输、调度等额外的开销，但这能有效避免由于物理机宕机等系统故障而导致整个在线服务不可用，同时在一定程度上保证了负载均衡，从而避免了资源竞争，提高了系统的容错性.

反亲和性(Anti-Affinity)可以用于描述同一个应用分组下容器的部署分散程度，一个高反亲和性的服务，会将同一个应用分组下的容器分散在不同的物理机上，以避免发生故障后服务完全崩溃. 同时高反亲和性保证了服务的并行性，由于同一服务下的容器会经常性集中请求相同的资源，容器对于物理节点的独享程度越高，则磁盘、网络 IO 带宽的拥塞几率越低. 为了量化反亲和性，我们定义了一个公式用于计算反亲和性：

$$anti\_affinity = \frac{n_{\text{app}}}{n_{\text{container}}},$$

其中，n_{app}是一台物理机上部署的应用总数，$n_{\text{container}}$是同一台物理机上容器总数.

为了检验阿里巴巴混部集群在线任务的反亲和性情况，将有容器分布的 4005 台机器的反亲和性分布数据集中绘制了 CDF 图(图 12)，可以发现，集群同一应用分组下容器部署反亲和性仍有很大的提升空间. 在不考虑调度复杂性的理想情况下，每台机器的反亲和性应为 1，但是在实际情况中，90%的机器反亲和性在 0.57 以下，99%机器的在 0.9 以下. 说明大部分相同应用分组下的容器并没有完全分散于不同的物理机上，这在一定程度上降低了在线服务容错能力和负载均衡性.

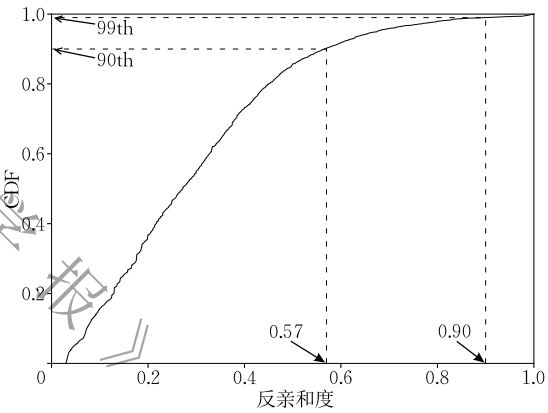
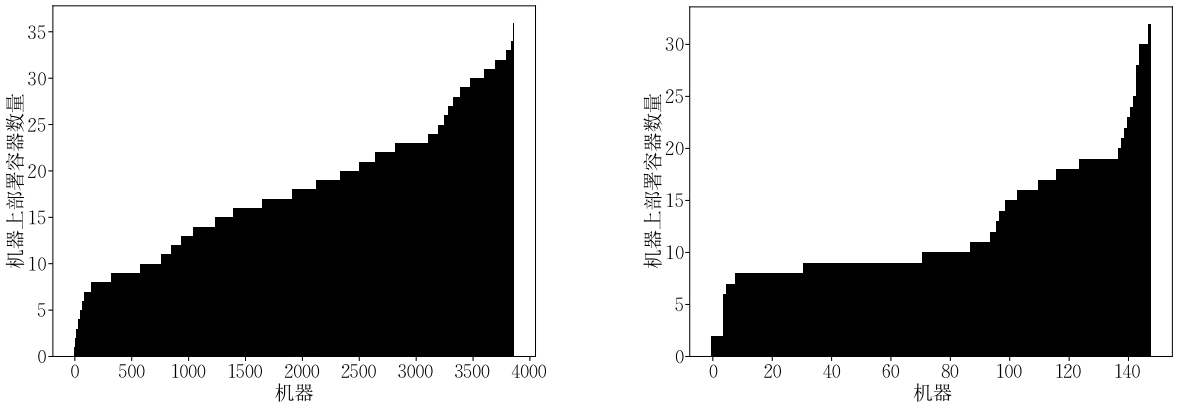


图 12　集群机器反亲和性 CDF 图

同时，我们也统计了混部机器和非混部机器中容器的分布情况，集群中总共部署了 71 476 个容器，其中混部机器中含有 69 645 个，非混部机器中有 1831 个.

图 13(a)和(b)分别为混部机器和非混部机器中的容器分布情况(已按机器上部署容器数量排序)，混部机器中单台机器部署容器最多为 36 个，最少为 1 个，平均数为 18.057 个，方差为 7.171，而非混部机器中单台机器部署容器最多为 32 个，最少为 2 个，平均数为 12.371 个，方差为 5.731. 可以发现集群中容器分布存在不平衡性，且混部机器下的容器分布更加不平衡. 这可能导致集群资源使用的不平衡.



(a) 混部机器下容器分布 (b) 非混部机器下容器分布

图 13 容器分布图

为此,我们统计了混部机器的 CPU 资源使用情况,图 14 为混部机器的 8 天内平均 CPU 资源使用情况直方图. 横坐标为机器 8 天平均 CPU 利用率,纵坐标为平均 CPU 利用率区间内机器数量. 可以发现大部分机器的 CPU 资源使用率在 40% 左右,且呈正态分布,其拟合优度为 0.969. 但是仍有一部份机器 CPU 利用率极低. 这也反映了当前集群负载分布不均衡.

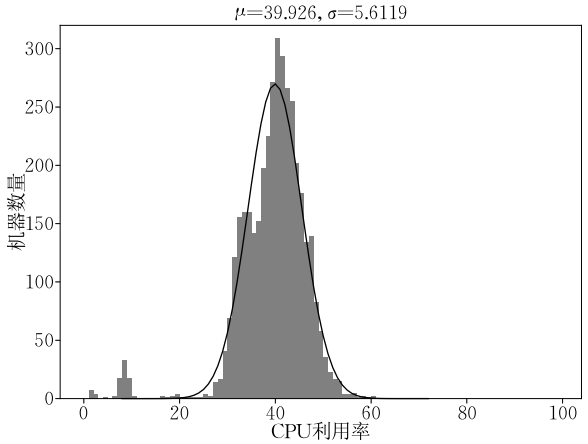


图 14 机器 CPU 利用率分布

5.3 混部后对容器性能影响

为了探究混部后离线任务对容器性能的影响,我们统计分析了容器中 CPI、Mem_GPS 和 MPKI 等性能指标的变化,表 8 为容器性能指标的描述. 由于表 container_usage 中这三个字段的记录存在缺失,或有异常数据(如 MPKI 的值异常大,如 MPKI 的值为 1736102),所以我们在统计作图时去除了这些记录.

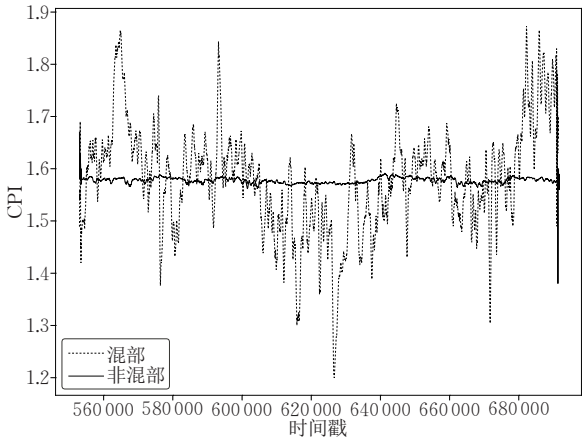
表 8 容器性能指标描述

	描述
CPI	每指令时钟周期(Cycles Per Instructions,CPI)
Mem_GPS	内存带宽,[0,100]归一化处理
MPKI	每千条指令缓存未命中数(Cache Miss Per Kilo Instructions,MPKI)

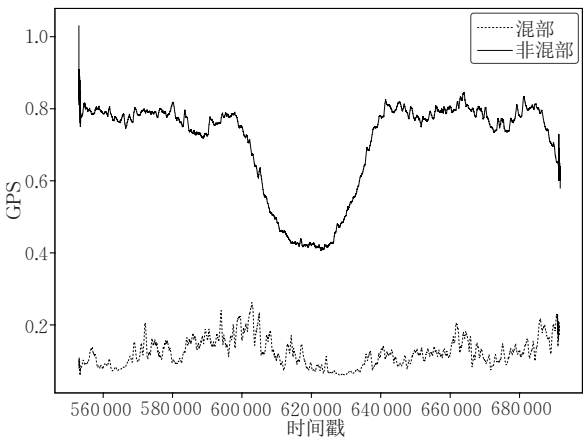
首先我们比较了混部机器和非混部机器下容器的 CPI、Mem_GPS 和 MPKI 的性能变化,图 15 为两者的比较图. 图 15 中虚线表示的是混部机器中容器的性能变化,即该部分容器与离线任务共享物理机器,实线表示的是非混部机器中容器的性能变化,即该部分容器单独使用物理机器. 横坐标表示为时间戳. 由于 CPI、Mem_GPS、MPKI 字段存在缺失,所以横坐标范围为[553 010,691 200],纵坐标为每个时间戳上所有容器各指标的平均值. 另外,图中的曲线都做了平滑处理,以更好的反映各指标随时间变化.

CPI 作为系统一个基本的性能指标,用以表示系统应用程序或功能的时延,CPI 值越高,意味着系统时延越高. 可以从图 15(a)发现非混部机器中的容器 CPI 非常稳定,而混部后的容器的 CPI 波动变化较大,即混部后容器的系统时延变化较大,这可能导致用户在使用在线服务时响应时间也变得不稳定. 同样的趋势也在图 15(c)中表现,非混部机器中的容器 MPKI 变化较稳定且 MPKI 值相对较小,而混部机器中的容器 MPKI 值波动较大,意味着混部机器中容器缓存未命中率变高,这将增加系统时延,以及消耗更多的系统资源.

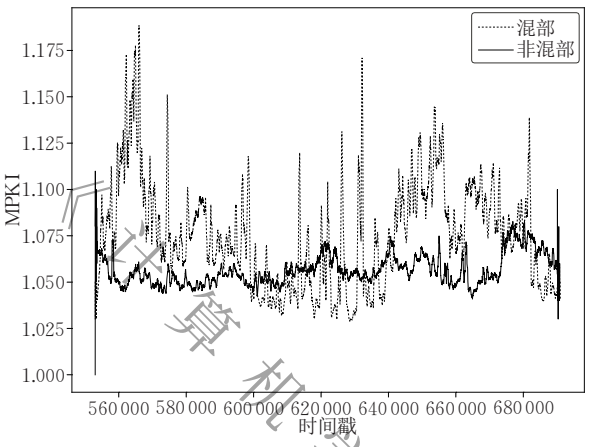
而在图 15(b)中,混部机器中容器的 Mem_GPS 明显低于非混部机器中的容器. 在时间戳 620 000 附近,非混部机器中容器的内存带宽有塌陷现象,结合图 11(b)任务失败分布,在时间戳 600 000 之前任务失败数量急剧增加,推断离线任务的混合部署会影响到集群中非混部机器中容器的性能,反映出混部集群存在一定波动性和互干扰性. 上一小节中



(a) CPI



(b) Mem_GPS



(c) MPKI

图 15　混部和非混部下容器性能比较

图 10(b)容器内存利用率也表现出混部后容器内存利用率下降，表明离线任务抢占了内存资源. 而在上一节中推断内存可能是影响集群性能的主要因素，混部后容器的内存带宽急剧下降，可能导致在线服务的性能也受到极大影响. 由此表明，混部后离线任务对于在线服务的服务质量影响较大，造成容器的运行状况不稳定，这是目前阿里云混部集群所需要解决的问题之一. 针对这个问题，混部集群中的调度器可以尝试根据所需内存大小和对系统时延容忍程度高低将容器分为不同等级，优先将运行在线服务内存需求较小和系统时延容忍性较高的容器与离线任务混部，以保证在线服务的 QoS.

混部机器中容器的 CPI、Mem_GPS 和 MPKI 性能指标的变化是由于离线任务的混合部署导致的，为了进一步探究在线任务和离线任务之间如何相互影响，我们分析了混部机器中离线任务部署与容器 CPI、Mem_GPS 和 MPKI 之间的关系，图 16 为对应关系曲线图.

图 16(a)～(c)左边纵坐标为各指标平均值，右边纵坐标为离线任务部署数量，图中深灰色实线为混部机器中容器各指标随时间变化曲线，浅灰色虚线为离线任务部署数量随时间变化曲线，黑色实线虚线分别代表对应拟合曲线. 图 16(d)为任务部署数量与容器 CPI、GPS、MPKI 拟合曲线，并计算了任务部署数量与容器 CPI、GPS、MPKI 的相关度，其中任务部署数量与 CPI 的相关度最显著为 −0.499，而任务部署与 GPS、MPKI 的相关度分别为 −0.200 和 0.160. 推断集群调度器是根据容器的当前性能决定离线任务的部署，即容器执行效率较高时，调度器会增加对该容器所在物理机的离线任务部署数量，而离线任务的部署又反过来影响了容器的执行性能. 以容器的 CPI 指标为例，当 CPI 呈现下降趋势时，意味着容器所遭受的资源抢占程度减轻，服务响应时延减小，此时混部调度器倾向于将离线任务混部到容器所运行的物理机上，以提高资源利用率；而随着部署离线任务数量的不断增加，容器系统时延开始上升(CPI 增大)，此时调度器开始限制离线任务部署量，以保证容器承载的在线服务

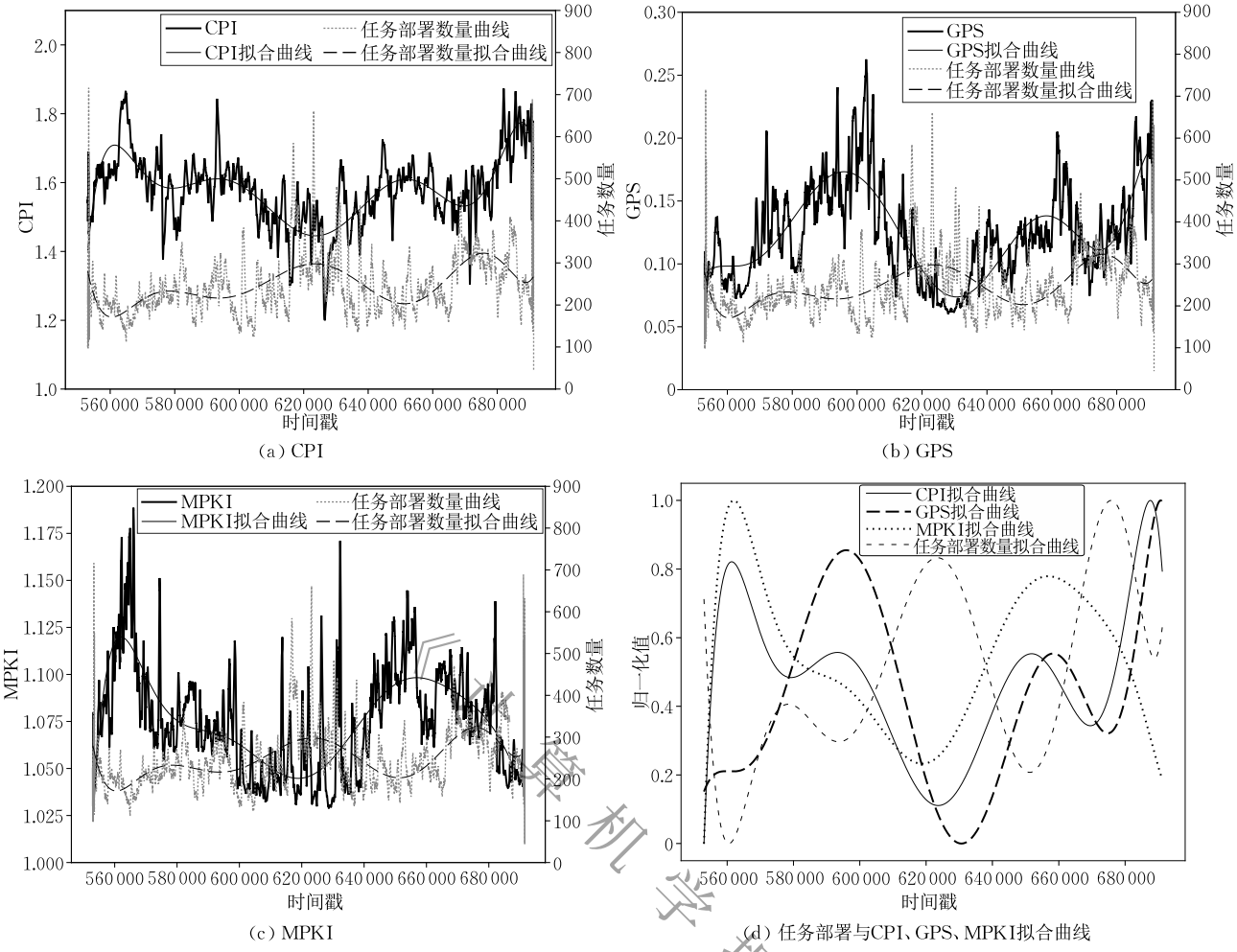


图 16 任务部署与容器性能关系比较图

QoS. 离线任务部署数量与容器 CPI 的高相关性，表明离线任务的部署很大程度上受容器 CPI 影响，可以将容器 CPI 作为协调 Sigma 调度器和 Fuxi 调度器在线服务和离线任务的重要指标.

虽然依据容器性能状态来动态部署离线任务，能动态调整离线任务和在线任务混部比例，但这样会由于离线任务不断增加或减少导致容器性能振荡，在线服务 QoS 也因此不能被保障，无法保证服务的稳定性，而且这也一定程度上增加了调度器的系统开销. 选择一个合适的在线任务和离线任务的混部比例是当前阿里云混部技术面临的一个主要问题. 而通过设定阈值可能无法很好解决当前容器性能振荡问题. 目前，自适应约束调度成为了学术界研究的热门领域[7-9]. 如何根据捕获到的系统动态，灵活地改变调度策略，成为了提高集群整体性能地突破口. 针对阿里云混部系统，自适应调度面临着更大的挑战，多调度器的协调为策略的设计增添了复杂性. 机器学习技术是解决设计复杂性的一项重要方法，在先验知识不足的情况下，可以通过大规模样本的训练挖掘特征间的关联，从而建立更为高效的调度策略.

6 相关工作

数据中心为用户提供着高效、廉价的云计算服务，但是云数据中心仍然存在资源利用率较低、能耗较大等问题. 因此，国内外众多研究人员对此展开了相关研究工作. Chen 等人[10]和 Chong 等人[11]从能耗的角度分析工作负载特征，认为负载特征对于提高数据中心资源利用率、降低能耗至关重要，并提出算法以提高资源利用率. Mazumdar 等人[12]认为更好地理解数据中心资源的使用，可以更有效地利用系统资源，降低能源成本，所以他们利用单变量方法、多元解释方法等多种统计方法对数据中心中不同时间尺度上的 CPU、内存、网络等资源进行分析，以提高资源使用率. Jiang 等人[13-14]和 Qiu 等人[15]

从能耗和能源效率角度出发，通过监视资源使用情况，对不同工作负载类型进行描述，及提出相应的任务调度策略，以降低数据中心能耗和提高资源使用率. Xu 等人[16]提出了干扰感知的虚拟机迁移策略，以减小服务器性能影响和能耗开销. Yi 等人[17]提出可根据用户需求指定作业执行效率，作为云服务定价新的因素. 云服务提供商在保证用户所指定作业在截止完成时间之前执行完毕，可灵活地调用提供给用户作业的资源，提高作业的弹性，以提高集群资源使用率. 考虑到在云服务中，租户提交的大多为体量小且运行时间短的任务，对资源需求较少，云服务提供商无法提供合适的资源配置，而造成资源浪费. Yi 等人[18]提出了通过组购买机制，将多个租户提交的多个小任务分组，云服务提供商以组方式提供云服务资源，同一小组中的任务可灵活地共享组中资源，以提高资源利用率.

然而，出于安全保密考虑，云服务提供商很少将集群中的日志数据公布出来. Google 于 2012 年公开了其异构集群的跟踪数据集[19]，数据集包含了 12500 台服务器 29 天的日志数据[20]. 许多研究人员根据此数据集，对大型集群的负载特征、调度机制、容错机制等相关方向进行了深入探究，取得了一定的研究成果[21-31]. 阿里巴巴于 2017 年首次公布了包含 1313 台机器的混部集群 24 小时的日志数据，许多学者对此展开了研究. Lu 等人[32]对该数据集进行分析，发现阿里巴巴混部集群在空间分布、时间分布、负载资源请求和利用等多个方面的不平衡性，并认为正是由于这种分布上的不平衡而极大影响集群效率. Jiang 等人[33]探究该数据集中多个混部特征，对失败任务和实例进行分析，并根据 CPU、内存、磁盘资源使用情况将在线服务分成 25 个类别. Chen 等人[34]对该数据集特征提取生成特征向量，并根据特征向量对负载进行分类，发现离线批处理作业的平均 CPU 利用率呈双峰分布，超过 50% 的机器只运行 1 类批处理作业，且作业持续时间较短，资源使用较少；对于在线实例，资源利用率非常低，且最多只有 6 类在线任务在用一台机器上运行. Liu 等人[5]提出塑性(plasticity)和弹性(elasticity)两个特征，深入研究离线实例利用容器预订但未使用的资源以提高集群资源利用率，并设置边界以保证在线服务性能. Deng 等人[35]、Cheng 等人[36-37]、Sun 等人[38]也对阿里巴巴的 cluster-trace-v2017 数据集进行了研究.

7 总　结

在本文中我们首先对阿里巴巴混部数据集 cluster-trace-v2018 进行概述，并根据多个不同负载特征对离线任务的作业模式进行分类，发现不同模式的作业在机器上的分布没有出现显著的相关性，分布之间较为均匀. 接着我们对资源预订情况进行探究，发现容器对资源预订存在“超订”现象. 并根据不同的分类标准对集群中的机器进行分类，对不同分类机器的资源进行统计，发现集群中存在“备用节点”，以应对集群中机器出现故障时，将故障机器上的任务调度至“备用节点”；发现延迟敏感的在线任务的 CPU 利用率较低，而对内存资源的需求比较高，离线任务的 CPU 利用率较高，在不影响在线任务性能的前提下，可以将在线任务和离线任务混部到同一台机器上，以达到提高资源利用率的效果. 最后我们分析了在线任务与离线任务之间的相互干扰，重点探究了离线任务对容器性能的影响，发现了一些当前系统存在的问题，提出了对应的优化建议.

本文主要研究了机器、容器和离线批处理作业的资源利用率、性能、运行状态等记录信息，希望通过三者的使用信息，从中探索有意义的发现，并比较彼此之间的相关性来探究离线任务和在线任务在资源使用和性能的相互干扰，发现了其中的混部调度问题，并提出改进建议.

致　谢　感谢阿里云公司调度团队徐国耀、丁海洋、吕齐、李治等工程师给本论文提供的富有启发的讨论和交流！

参　考　文　献

[1] Verma A, Pedrosa L, Korupolu M, et al. Large-scale cluster management at Google with Borg//Proceedings of the 10th European Conference on Computer Systems. Bordeaux, France, 2015: 1-17

[2] Hazelwood K, Bird S, Brooks D, et al. Applied machine learning at Facebook: A datacenter infrastructure perspective //Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). Vienna, Austria, 2018: 620-629

[3] Iorgulescu C, Azimi R, Kwon Y, et al. Perfiso: Performance

isolation for commercial latency-sensitive services//Proceedings of the 2018 USENIX Annual Technical Conference. Boston, USA, 2018: 519-532

[4] Tian H, Zheng Y, Wang W. Characterizing and synthesizing task dependencies of data-parallel jobs in alibaba cloud// Proceedings of the ACM Symposium on Cloud Computing. Santa Cruz, USA, 2019: 139-151

[5] Liu Q, Yu Z. The elasticity and plasticity in semi-containerized co-locating cloud workload: A view from Alibaba trace// Proceedings of the ACM Symposium on Cloud Computing. Carlsbad, USA, 2018: 347-360

[6] Guo J, Chang Z, Wang S, et al. Who limits the resource efficiency of my datacenter: An analysis of Alibaba datacenter traces//Proceedings of the International Symposium on Quality of Service. Phoenix, USA, 2019: 1-10

[7] Xu F, Liu F, Jin H. Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud. IEEE Transactions on Computers, 2015, 65(8): 2470-2483

[8] Xu F, Liu F, Jin H, et al. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. Proceedings of the IEEE, 2013, 102(1): 11-31

[9] Li F, Hu B. DeepJS: Job scheduling based on deep reinforcement learning in cloud data center//Proceedings of the 2019 4th International Conference on Big Data and Computing. Guangzhou, China, 2019: 48-53

[10] Chen T, Wang X, Giannakis G B. Cooling-aware energy and workload management in data centers via stochastic optimization. IEEE Journal of Selected Topics in Signal Processing, 2015, 10(2): 402-415

[11] Chong F T, Heck M J R, Ranganathan P, et al. Data center energy efficiency: Improving energy efficiency in data centers beyond technology scaling. IEEE Design & Test, 2013, 31(1): 93-104

[12] Mazumdar S, Kumar A S. Statistical analysis of a data centre resource usage patterns: A case study//Proceedings of the International Conference on Computing and Communication Systems. Shillong, India, 2018: 767-779

[13] Jiang C, Wang Y, Ou D, et al. Energy efficiency comparison of hypervisors. Sustainable Computing: Informatics and Systems, 2019, 22: 311-321

[14] Jiang C, Fan T, Qiu Y, et al. Interdomain I/O optimization in virtualized sensor networks. Sensors, 2018, 18(12): 4395

[15] Qiu Y, Jiang C, Wang Y, et al. Energy aware virtual machine scheduling in data centers. Energies, 2019, 12(4): 646

[16] Xu F, Liu F, Liu L, et al. iAware: Making live migration of virtual machines interference-aware in the cloud. IEEE Transactions on Computers, 2013, 63(12): 3012-3025

[17] Yi X, Liu F, Li Z, et al. Flexible instance: Meeting deadlines of delay tolerant jobs in the cloud with dynamic pricing// Proceedings of the 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS). Nara, Japan, 2016: 415-424

[18] Yi X, Liu F, Niu D, et al. Cocoa: Dynamic container-based group buying strategies for cloud computing. ACM Transactions on Modeling and Performance Evaluation of Computing Systems, 2017, 2(2): 1-31

[19] Reiss C, Wilkes J, Hellerstein J L. Google cluster-usage traces: Format+schema. Google Inc., White Paper, 2011: 1-14

[20] Jassas M, Mahmoud Q H. Failure analysis and characterization of scheduling jobs in Google cluster trace//Proceedings of the IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society. Washington, USA, 2018: 3102-3107

[21] Reiss C, Tumanov A, Ganger G R, et al. Heterogeneity and dynamicity of clouds at scale: Google trace analysis// Proceedings of the 3rd ACM Symposium on Cloud Computing. San Jose, USA, 2012: 1-13

[22] Reiss C, Tumanov A, Ganger G R, et al. Towards understanding heterogeneous clouds at scale: Google trace analysis. Intel Science and Technology Center for Cloud Computing, Tech. Rep, 2012, 84

[23] Di S, Cappello F. GloudSim: Google trace based cloud simulator with virtual machines. Software: Practice and Experience, 2015, 45(11): 1571-1590

[24] Alam M, Shakil K A, Sethi S. Analysis and clustering of workload in Google cluster trace based on resource usage// Proceedings of the 2016 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC) and 15th International Symposium on Distributed Computing and Applications for Business Engineering (DCABES). Paris, France, 2016: 740-747

[25] Rasheduzzaman M, Islam M A, Islam T, et al. Task shape classification and workload characterization of Google cluster trace//Proceedings of the 2014 IEEE International Advance Computing Conference (IACC). Gurgaon, India, 2014: 893-898

[26] Amvrosiadis G, Park J W, Ganger G R, et al. On the diversity of cluster workloads and its impact on research results// Proceedings of the 2018 USENIX Annual Technical Conference. Boston, USA, 2018: 533-546

[27] Alnooh A H A, Abdullah D B. Investigation and analysis of Google cluster usage traces: Facts and real-time issues// Proceedings of the 2018 International Conference on Engineering Technology and Their Applications (IICETA). Al-Najaf, Iraq, 2018: 60-65

[28] Islam T, Manivannan D. Predicting application failure in cloud: A machine learning approach//Proceedings of the 2017 IEEE International Conference on Cognitive Computing (ICCC). Honolulu, USA, 2017: 24-31

[29] Han R, Zong Z, Zhang F, et al. CloudMix: Generating diverse and reducible workloads for cloud systems//Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD). Honolulu, USA, 2017: 496-503

[30] Xu C, Wang K, Guo M. Intelligent resource management in blockchain-based cloud datacenters. IEEE Cloud Computing, 2017, 4(6): 50-59

[31] Dabbagh M, Hamdaoui B, Guizani M, et al. An energy-efficient VM prediction and migration framework for overcommitted clouds. IEEE Transactions on Cloud Computing, 2016, 6(4): 955-966

[32] Lu C, Ye K, Xu G, et al. Imbalance in the cloud: An analysis on alibaba cluster trace//Proceedings of the 2017 IEEE International Conference on Big Data (Big Data). Boston, USA, 2017: 2884-2892

[33] Jiang C, Han G, Lin J, et al. Characteristics of co-allocated online services and batch jobs in Internet data centers: A case study from alibaba cloud. IEEE Access, 2019, 7: 22495-22508

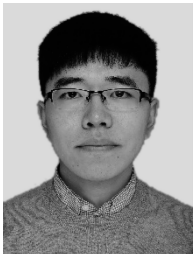
[34] Chen W, Ye K, Wang Y, et al. How does the workload look like in production cloud? Analysis and clustering of workloads on Alibaba cluster trace//Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). Singapore, 2018: 102-109

[35] Deng L, Ren Y L, Xu F, et al. Resource utilization analysis of Alibaba cloud//Proceedings of the International Conference on Intelligent Computing. Wuhan, China, 2018: 183-194

[36] Cheng Y, Anwar A, Duan X. Analyzing Alibaba's co-located datacenter workloads//Proceedings of the 2018 IEEE International Conference on Big Data (Big Data). Seattle, USA, 2018: 292-297

[37] Cheng Y, Chai Z, Anwar A. Characterizing co-located datacenter workloads: An Alibaba case study//Proceedings of the 9th Asia-Pacific Workshop on Systems. Jeju Island, South Korea, 2018: 1-3

[38] Sun Q, Wu C, Li Z, et al. Colocation demand response: Joint online mechanisms for individual utility and social welfare maximization. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3978-3992



GE Zhe-Feng, M. S. His research interests include cloud computing and data analysis.

WANG Ji-Wei, M. S. His research interests include cloud computing and data analysis.

JIANG Cong-Feng, Ph. D. His research interests include cloud computing, system optimization and performance evaluation.

ZHANG Ji-Lin, Ph. D. His research interests include high-performance computing and distributed system.

YU Jun, Ph. D. His research interests include machine learning, multimedia analysis and image processing.

LIN Jiang-Bin, M. S. His research interests include distributed storage systems, system benchmarking and reliability optimization.

YAN Long-Chuan, M. S. His research interests include data center energy efficiency optimization and system virtualization.

REN Zu-Jie, Ph. D. His research interests include distributed system and cloud computing.

WAN Jian, Ph. D. His research interests include cloud computing and big data analytics.

Background

Modern Internet Data Centers (IDCs) provide the convenience of various computing service in our daily life. And the scale of IDCs is increasing with the growth of application service demand, but high energy consumption and electric cost have become a tough problem to cloud service providers because of low resources utilization, which hampers the further development of cloud computing. To handle such inefficiencies and improve the resources utilization in IDCs, a worked strategy has been applied that cloud service providers co-locate the online services and offline batch jobs in the same production cluster. However, co-located deployment increases the complexity of online/offline tasks scheduling system, which may cause an imbalance in the distribution of workload and decrease the robustness of the whole cluster. Many works have studied the co-located cluster through the published Google cluster trace and Alibaba cluster trace etc., and found the problems and given the corresponding suggestions.

The Alibaba Cluster Trace Program helps the researchers,

students and people who are interested in the field to get better understanding of the characteristics of IDCs and the workloads. From Alibaba perspective, the main purpose of publishing the trace dataset is to explore the interaction between online services and offline batch jobs in the co-located cluster and get the feedback about how to adjust resource allocation between online service and batch jobs to improve throughput of batch jobs while maintain acceptable QoS, which helps design a better collaboration mechanism between online service scheduler (Sigma) and batch jobs scheduler (Fuxi).

This paper, from the perspective of resource utilization, statistically analyzes the newly co-located cluster trace released by Alibaba, which contains 4034 machines for 8 days. We gathered the statistics of cluster resources utilization and found the resources utilization has been improved after co-locating, especially the CPU utilization. And we describe the resources allocation strategy of offline batch jobs and online services and the interference between offline batch tasks and online services. In the meantime, some current problems about deployment and schedule were revealed, and we analyzed the problems and present recommendations. Our findings and insights raised in this paper can help the industrial and academia communities better understand the workload characteristics and scheduling strategy in co-located IDCs.

This paper is supported by the National Key Research and Development Program of China (No. 2017YFB1010000), the National Natural Science Foundation of China (No. 61972118), and the Key Research and Development Program of Zhejiang Province (No. 2017C01SA160069).