Website：ycyk.brit.com.cn

基于多级队列择优的信号处理资源调度策略研究

马潇潇[1]，孙甲琦[2]，朱宏涛[1]，杨罗兰[1]
（1 北京遥测技术研究所 北京 100076；
2 中国航天电子技术研究院 北京 100094）

摘要：新型航天地面测控装备采用虚拟资源池架构和容器云技术，按需进行测控资源调度和测控业务能力生成。资源池系统包含种类繁多的异构资源，针对其中信号处理资源的调度需求，本文设计了一种基于多级队列择优的信号处理资源调度策略，将资源聚合管理、统一分配，根据任务需求分级择优进行分配调度，保证系统资源调度的高可靠性，兼顾资源的高效利用和负载均衡。

关键词：资源管理；分配策略；调度流程
中图分类号：V556.1; TP33 **文献标志码：**A **文章编号：**2095-1000(2024)04-0097-11
DOI：10.12347/j.ycyk.20240419001
引用格式：马潇潇，孙甲琦，朱宏涛，等. 基于多级队列择优的信号处理资源调度策略研究[J]. 遥测遥控，2024，45(4)：97–107.

Research on Signal Processing Resource Scheduling Strategy Based on Multilevel Queue Optimization

MA Xiaoxiao[1]，SUN Jiaqi[2]，ZHU Hongtao[1]，YANG Luolan[1]
（1. Beijing Research Institute of Telemetry, Beijing 100076, China;
2. China Academy of Aerospace Electronics Technology, Beijing 100094, China）

Abstract: The new aerospace ground TT&C equipment adopts the virtual resource pool architecture and container cloud technology to carry out on-demand scheduling of measurement and control resources and generation of measurement and control business capabilities. The resource pool system contains a wide variety of heterogeneous resources, for which the signal processing resources are task-oriented scheduling requirements, this paper designs a signal processing resource scheduling strategy based on multilevel queue optimization for the resource pool, which aggregates the resources for management and unified allocation, and allocates and schedules them according to the task requirements in a hierarchical and optimal way to ensure the high availability of the resource pool system and the high utilization rate of resources.

Keywords: Resource management; Distribution strategy; Scheduling process
Citation: MA Xiaoxiao, SUN Jiaqi, ZHU Hongtao, et al. Research on Signal Processing Resource Scheduling Strategy Based on Multilevel Queue Optimization[J]. Journal of Telemetry, Tracking and Command, 2024, 45(4): 97–107.

0 引言

随着我国航天器走向深空，航天测控网建设进入了全面发展新阶段[1]。发射密集的各类卫星和测控任务资源需求的异构性对航天测控地面站网的容量、速率以及资源调度能力等都提出了更高要求，激励着航天测控资源调度方法不断改进。不少学者采用遗传算法对测控资源调度问题进行了研究。国防科技大学的陈峰等人[2]将被调度弧段按时间划分为两部分，设计了两阶段递进的遗传算法，对第一部分进化求解后，将最优解与第二部分整合并进一步优化，明显减少了运算时间。航天工程大学的薛乃阳等人[3]引入微元思想，提出了基于改进遗传算法的求解策略，提高了测控需求满足率和测控收益。遗传算法虽然可以有效改善测控资源调度的效率问题，但无法避免计算复杂度高的问题，只适用于小规模卫星需求的资源调度。

收稿日期：2024-04-19；修回日期：2024-05-16

针对遗传算法的不足，结合启发式策略的算法逐渐被引入地面站资源规划问题中，国内外不少学者基于不同的启发式因子提出了多种启发式求解策略。西安卫星测控中心的刘建平等人[4]聚焦我国航天测控网调度对任务优先级的需求，融合最大可用窗口价值规则和最早可用窗口集规则，提出了一种基于混合启发式的解构造算法，通过动态构造启发式规则提高求解质量。LI Z Y 等人[5]采用先进先出调度算法对测控业务进行仿真，对任务满意度和资源利用率等指标进行了量化分析。XHAFA F 等人[6]在分析测控资源调度问题复杂性的基础上提出了元启发式的解决方案。典型用于卫星地面站资源规划问题的元启发算法还有模拟退火法、进化计算法等。启发式算法应用于大规模调度问题，能够降低问题求解的计算复杂度，但其仅以单个优化目标作为评价准则，难以应对新型航天地面测控任务调度的复杂性和资源需求的异构性。

现阶段，我国航天地面测控站网面临着大量卫星同时过境进行数传服务导致资源紧张的难题。传统解决资源冲突的方法是增加设备数量，地面装备研制建设紧跟卫星型号的发展，各套设备单独研制、独立运行，占地面积大、建设成本高，导致站内资源统一调度灵活性差，综合运用效率低。

针对这类问题，新型航天地面测控装备采用资源池架构和容器云技术[7]，按需进行测控资源调度和测控业务能力生成。这种资源池系统包含种类繁多的异构资源，信号处理资源是管理和调度的重点。其中，综合运管平台采用商用国产服务器集群，部署服务化应用软件来完成整个测控数传设备的综合管控[8]，接收测控中心发送的工作计划，根据任务要求对全系统资源的属性和能力统一规划，结合资源能力和状态进行分配和调度，最大化发挥系统的使用效能。

资源池系统综合运管平台总体架构如图 1 所示，分为基础设施层、虚拟资源层、系统服务层和综合运管层。基础设施层的各项测控设备资源通过分类和虚拟化形成虚拟资源层[9]；虚拟资源层基于开源容器编排引擎(Kubernetes)对各类池化资源进行虚拟化管理；系统服务层是集中管理、高效调度的基础，主要包括设备监控服务、资源调度服务、信号处理服务等，其中资源调度服务是本文的研究重点；综合运管层是实现灵活运管、能力生成的关键和核心，主要包括系统综合监控、任务运行管理和健康管理等[10]。本文重点阐述了资源调度服务中信号处理资源的调度策略，面向不同场景任务的应用需求设计了资源管理、分配和自动调度的一系列流程和方法，将集群资源聚合管理、统一调度，保证资源池系统的高可用性和资源的高利用率。

1 资源多级队列概述

为了高效地管理资源，实现可视化的资源状态监控与合理的分配调度，本文建立了设备级资源队列和单元级资源队列，设计了多级队列择优的资源分配策略，面向不同任务采取相适应的调度策略，优先考虑高可靠、高性能需求，兼顾资源的高效利用和负载均衡。

设备级资源队列是指将历史任务成功执行时使用的任务资源组合队列保存在独立的数据库表中进行标记管理的资源队列，以备相同任务到来时可以直接采用原资源组合队列调度，这种队列设计在工程应用中具有较大优势：

① 使用执行过任务的资源队列成功率更高，不易出现偶发故障[11]；

② 使用同样的资源组合能够保持执行环境不变，便于进行场景复原和问题分析[12]；

③ 在进行任务演练后保持状态不变，保证任务的顺利执行。

单元级资源队列则是以不同节点的 GPU、CPU、内存、磁盘以及网络 IP 等为单元进行细粒度资源分配而形成的队列。这种队列设计用以在没有匹配到设备级资源队列时，根据任务资源需求在分类资源池中按设计策略分类分别分配。

设备级资源队列的优势是可靠性强、效率高，而单元级队列的优势是通过两级匹配策略维持节点紧凑和负载均衡，达到节约节点资源的目的，用最优配置的资源发挥最大的任务效能。

此外，通过设计历史任务队列、资源状态表等数据库表，可以全方位、全流程地对系统资源进行可视化监控和管理，增强系统运行的稳健性[13]。

2 资源调度服务设计

资源调度[14]服务是资源池综合运管平台系统服务层的核心功能之一，负责根据不同任务的资源

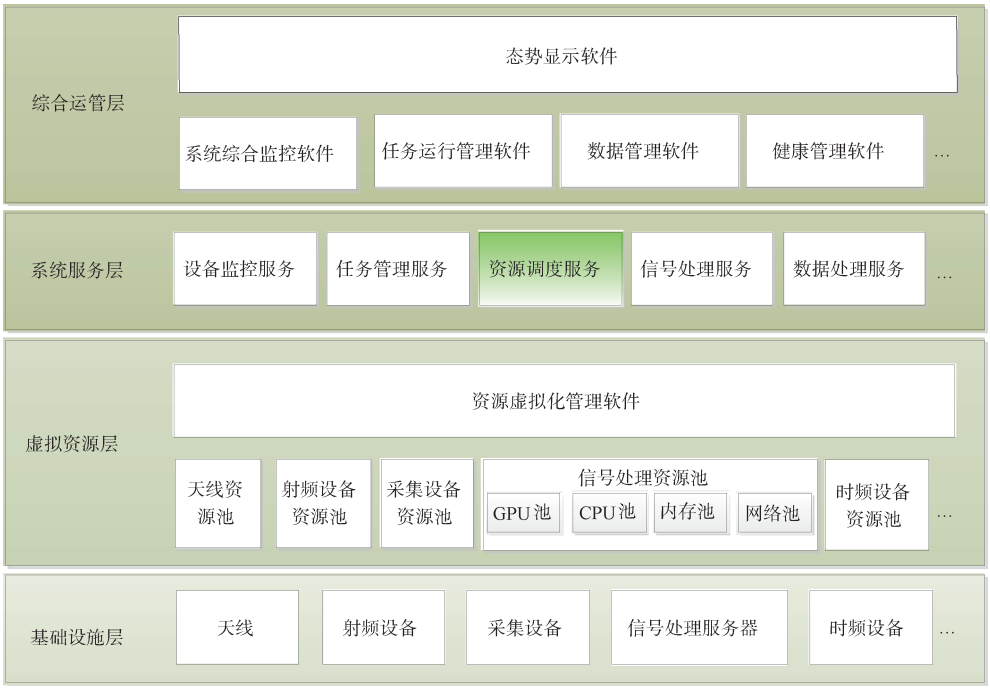

图 1　资源池综合运管平台总体架构

Fig. 1　Overall architecture of the integrated resource pool operations management platform

需求采取相应策略分配各类资源，并响应任务流程进行资源调度。同时能够处理资源请求冲突[15]，提供资源监控信息，提高资源利用率。针对这种资源调度功能需求，本章进行了资源调度服务的架构设计与功能设计。

2.1　总体架构设计

基于多级队列择优的资源调度在应用场景上应充分考虑集群节点和云平台的部署应用、系统资源面向任务的动态管理分配，以及基于业务历史数据的资源调度等，在功能上应重点具备以下三方面：

① 资源分配：能够根据计划任务需求，结合系统资源的能力、状态和占用情况，进行资源的规划和分配；

② 资源调度：根据资源分配结果，自动生成系统资源使用计划，组织调度相关设备，配合完成系统能力的动态重构；

③ 资源监控：能够实现对各类资源属性和能力的统一建模，在任务执行和资源分配调度的整个过程进行资源属性和状态的监控，具备资源的动态注册和注销功能，支持系统资源的扩展升级。

结合地面测控资源池系统特点和上述功能需求，本文对基于多级队列择优的资源池资源调度服务进行了总体架构设计，总体架构图如图 2 所示。

2.2　资源监控单元功能设计

资源监控[16]单元是资源调度服务的基础，设计资源监控线程负责对系统各类资源池资源进行监控和维护，基于数据库表对资源状态和队列进行统一管理，为任务执行和资源调度做好准备工作。

2.2.1　单元级资源状态管理

本文设计如表 1 所示的资源状态管理表，以节点清单为基础，对资源池系统中各类型资源池中的单元资源可用性状态进行统一管理。以 Master-1、Master-2 节点为示例，采用数据库存储各节点的资源状态，标识资源的类型、编号、数量及可用情况，将其作为资源的可行性判决条件。

GPU 资源主要用于处理高速信号数据，以编号 GPU 卡为分配单位，表中两个节点均有 8 块 GPU 卡可供分配，其可用情况包括：空闲、占用、维护。CPU 资源以 CPU 卡为分配单位，主要用于处理常规数据，为减少分配复杂度，不进行编号管理。内存和磁盘以容量为分配单位，当剩余容量不满足一次资源需求时不再进行分配。

每个节点单元资源可用性状态以事件驱动方式由资源监控单元进行维护，当该单元资源被分配前、被调度占用后、被维护前或被释放后即锁定其状态并进行更新。

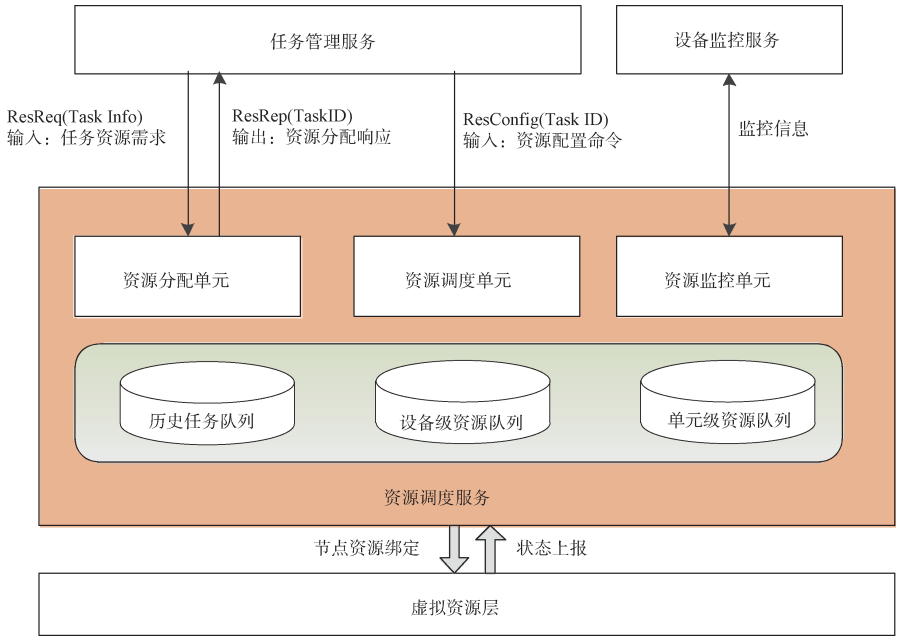

图 2 资源调度服务总体架构

Fig. 2 Resourece scheduling service overall architecture

表 1 资源状态管理表设计示例

Table 1 Resource status management table design example

节点名称	资源类型	资源编号/数量	资源可用情况
Master-1	GPU	GPU-1	空闲
		GPU-2	占用
		GPU-3	占用
		GPU-4	空闲
		GPU-5	空闲
		GPU-6	空闲
		GPU-7	空闲
		GPU-8	维护
	CPU	64(个)	空闲 64(个)
	内存	257 321(MB)	空闲 255 357(MB)
	磁盘	208(MB)	空闲 145(MB)
Master-2	GPU	GPU-1	占用
		GPU-2	维护
		GPU-3	空闲
		GPU-4	空闲
		GPU-5	空闲
		GPU-6	空闲
		GPU-7	空闲
		GPU-8	空闲
	CPU	64(个)	空闲 59(个)
	内存	257 321(MB)	空闲 256 437(MB)
	磁盘	208(MB)	空闲 100(MB)

2.2.2 设备级资源队列管理

设备级资源队列管理表对历史任务最新一次成功执行时分配使用的资源组合进行标记管理。如表 2 所示，以任务代号和任务类型为任务索引，每一行记录该任务最新一次成功执行时所使用的节点、GPU 卡编号、CPU 数量以及占用的内存和磁盘容量。

表 2 记录了 3 个测控任务分配使用过的资源队列，若当前新到来的任务为其中 3 个任务之一，则对该任务对应的资源进行可用性判断，如果均可用则直接将该资源组合分配给该任务；如果其中某一项资源不满足需求，则需要进行单元级资源匹配，输出新的资源分配结果并更新设备级资源队列管理表中该任务的相应资源信息。

2.2.3 历史任务队列管理

为了保存全部历史任务资源分配情况，进行场景复原及问题分析等，本文设计了历史任务队列，存储所有历史任务执行时调用的资源组合情况。

表 3 中记录了每次任务所调度的节点名称、使用的系统资源以及任务完成状态。通过历史任务队列，用户可以随时分析执行情况、查询系统环境，同时可以统计资源的使用情况，便于用户对资源分配、调度方案进行调整和优化。从上表的示例可以看出，当 0ZYCKJ 测控任务再次出现时，历史资源组合中的 GPU-1 已被正在执行的 2ZYCKJ 数传任务调度占用，只能通过分配策略选择 GPU-

表 2　设备级资源队列管理表设计示例

Table 2　Device-level resource queue management table design example

任务代号	任务类型	节点名称	GPU 使用	CPU 使用	内存使用/MB	磁盘使用/GB
0ZYCKJ	测控	Master-1	GPU-1	CPU：3	4 096	164
1ZYCKJ	数传	Master-2	GPU-2	CPU：4	4 096	208
2ZYCKJ	数传	Master-1	GPU-1	CPU：3	4 096	164

表 3　历史任务队列数管理表设计示例

Table 3　Historical task queue management table design example

任务代号	任务类型	节点名称	GPU 使用	CPU 使用	内存使用/MB	磁盘使用/GB	任务状态
0ZYCKJ	测控	Master-1	GPU-1	CPU：3	4 096	164	已完成
1ZYCKJ	数传	Master-2	GPU-2	CPU：4	4 096	208	已完成
2ZYCKJ	数传	Master-1	GPU-1	CPU：3	4 096	164	正执行
0ZYCKJ	测控	Master-1	GPU-3	CPU：3	4 096	164	正执行

3 执行任务。历史任务队列将 0ZYCKJ 测控任务的不同执行情况全部存储，用户可通过不同资源分配方案的任务执行情况进行复盘分析。

2.3　资源分配单元功能设计

由于设备级和单元级队列的资源调度策略应用场景和所具有的优势不同，因此需要设计资源分配线程，针对不同任务进行相适应策略的分配调度，提高任务的调度效率，增强系统的资源利用能力。本节主要进行资源多级队列择优分配的流程设计与策略设计。

2.3.1　任务资源分配流程

任务资源分配流程图如图 3 所示，其中四个分支判决是本流程设计的重点。

① 新任务是否为历史任务

查询设备级资源队列管理表，如果新到来的计划任务是成功执行过的历史任务，则分析相对应的历史资源组合的可用性，否则直接进行单元资源分配。

② 历史资源组合是否可用

对于成功执行过的历史任务，查询资源可用性状态管理表，如果该任务对应的资源组合全部可用，则直接进行分配，否则需对各类型资源的不可用性情况进行分析。

③ CPU、内存、磁盘资源是否均满足可用性

对于 CPU、内存、磁盘等非编号资源，若资源不足，说明该节点不可用，需要重新进行单元级资源分配。

④ 同一节点是否存在可用 CPU

若非编号资源满足需求，则判断同一节点是否存在可用 GPU，如果没有可用 GPU，为避免跨节点分配，需要重新进行单元级资源分配。

根据资源分配和调用结果需同步更新历史任务队列、设备级资源队列等数据库表，完成资源的多级队列择优分配。

2.3.2　单元资源分配策略

对于集群系统，单元资源分配设计资源预选

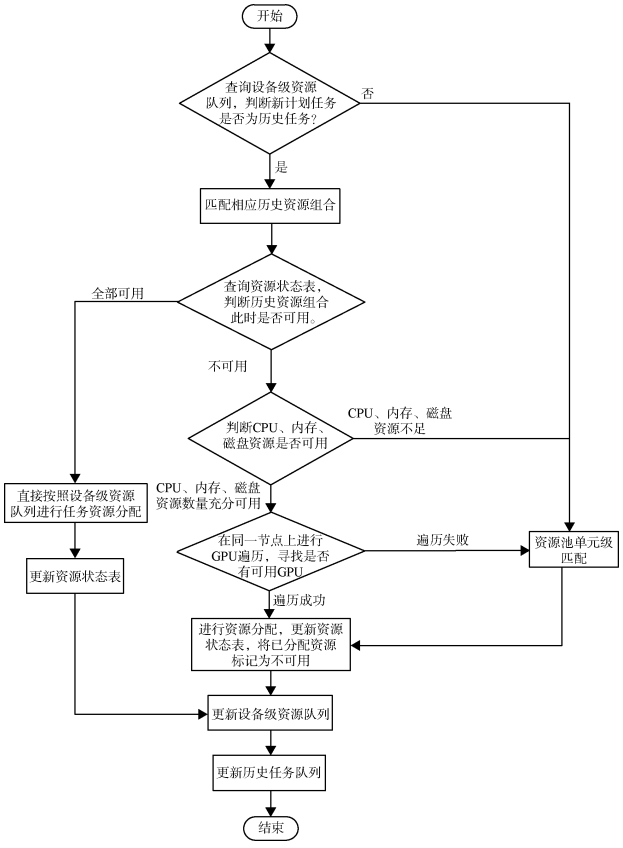

图 3　任务资源分配流程图

Fig. 3　Flowchart for the allocation of mission resources

和资源优选两级分配方式[17]。首先在预选阶段过滤不可用节点，制定适配资源池系统和任务类型的预选策略，遍历所有节点，筛选出符合要求的计算节点列表[18]。如果没有计算节点符合预选策略规则，则该任务就会被挂起，直到有计算节点能够满足要求。然后，在预选基础上，采用优选策略计算出每个候选节点的积分，积分最高者被择优分配[19]。

单元资源分配流程图如图4所示。本文重点根据卫星任务特点以及测控设备常规需求，对预选策略进行优化，对优选策略设计可选资源约束条件和可用性评分公式，选出最高分节点与Pod绑定，完成单元资源分配。

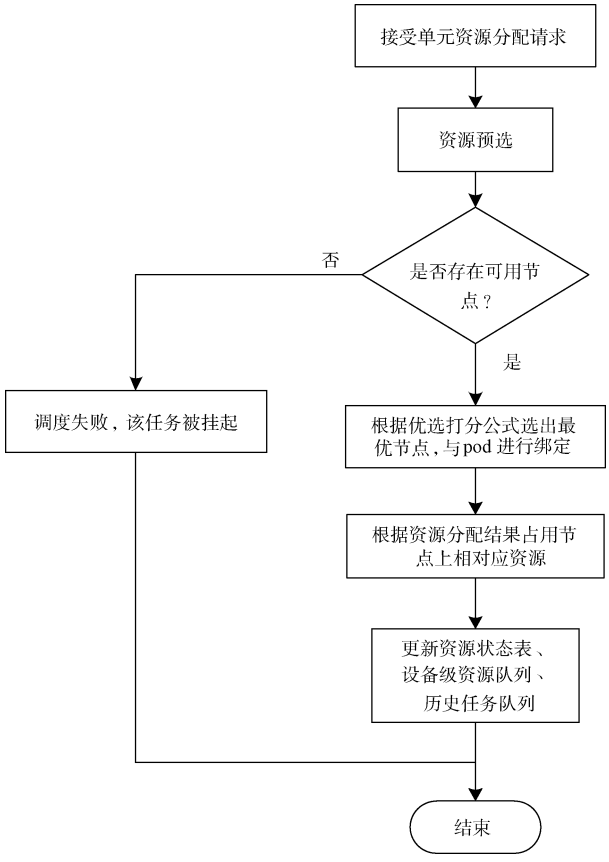

图4 单元资源分配流程

Fig. 4 Module resource allocation process

2.3.2.1 预选策略设计

预选阶段初步筛选出候选节点列表，筛选规则可以根据需求自主设计。本文对Kubernetes的预选策略进行研究分析，并根据地面测控系统的应用场景，进行优化设计。

Kubernetes默认的预选策略[20]包括：

① PodFitsHostPorts：确定节点上关于待调度Pod所请求的端口是否空闲。

② PodFitsHost：确定待调度Pod是否指定了特定的节点。

③ PodFitsResources：确定节点是否还存在剩余资源供待调度Pod使用。

④ MatchNodeSelector：确定节点的标签是否与Pod所要求的一致。

Kubernetes默认调度器具有一定通用性，但无法满足资源池系统资源需求多样性及地面测控任务复杂性，本文在默认调度方法的基础上对预选阶段增加以下优化策略：

⑤ 节点亲和性策略选择：允许用户自定义一组规则，指定Pod应该被调度到满足这些规则的节点上。

在任务到来时，若存在两个应用交互频繁的情况，用户可选择使用亲和性将他们部署在同一节点中，以减少网络通信开销；当应用采用多副本部署时，可以使用反亲和性确保副本分布在不同节点。

⑥ 负载均衡策略选择：默认策略只能保证单节点的服务情况，未考虑整个集群的负载均衡，本文将根据负载情况把节点排成有序队列，用户可根据节点使用情况适当选取一定比例节点作为选择对象。

为了预防因资源过载而影响节点中Pod的正常运行，本文为节点上的资源设置最大负载率，定义L_N为节点的负载均衡度，计算公式如下：

$$L_N = v_1(M_{CPU} - L_{CPU}) + v_2(M_{GPU} - L_{GPU}) + v_3(M_{Dick} - L_{Disk}) + v_4(M_{Mem} - L_{Mem}) \quad (1)$$

其中，L_{CPU}、L_{GPU}、L_{Disk}和L_{Mem}分别代表节点当前CPU、GPU、磁盘和内存的负载率；M_{CPU}、M_{GPU}、M_{Dick}和M_{Mem}表示资源的临界负载，取值范围为(0，1)，用户可根据项目经验和任务调度情况自定义；v_1、v_2、v_3和v_4为自定义均衡度权重，权重大小根据资源对节点均衡的影响程度由用户设置，其中$v_1+v_2+v_3+v_4=1$。

资源综合负载均衡程度越高L_N越大，将所有节点按L_N从大到小的顺序存入节点队列中，以备优选阶段选用。

2.3.2.2 优选策略设计

最常用的Kubernetes默认优选策略为LeastRe-

questedPriority[21](最低要求优选法)。根据节点上CPU和内存的资源消耗程度打分，资源消耗越少，该节点分数越高，越能部署更多应用，目的是将应用尽可能分布到所有节点上去，是最为传统的平铺策略。该策略在选择最优节点的过程中忽略了GPU对资源调度的影响，打分方式单一，资源空闲率高，不适合本文场景。

资源池系统任务资源需求具有差异性，同时需综合考虑后续任务对资源的潜在需求，因此在优选环节设置三个资源约束条件可选项，对满足约束条件的预选队列节点进行可用性评分。

① 负载均衡约束：如需考虑负载均衡，则对预选后的节点队列设置最低L_{N}值。

② 最大GPU数约束：当系统所有节点的GPU数均不能满足一个高资源需求任务运行时向用户告警。该可选约束的主要目的是为后续可预见的任务预留至少一个节点可用的GPU资源，最大GPU需求数可以设置。

③ 最小GPU数约束：考虑到本文涉及的任务基本为长管卫星任务，当单节点剩余GPU数小于任务最小GPU数需求时，为避免分配后容易形成碎片化资源，该节点不参与此次分配。

本文对满足资源约束条件的节点队列提出了节点资源可用性评分优选策略，突出可用资源与需求匹配程度的计算权重。

定义$score_{\mathrm{cpu}}$为节点上CPU的使用情况得分，具体计算如式(2)所示，其中T_{cpu}为节点上CPU的资源总量；U_{cpu}为节点上已用CPU资源量。

$$score_{\mathrm{cpu}}=\frac{T_{\mathrm{cpu}}-U_{\mathrm{cpu}}}{T_{\mathrm{cpu}}} \tag{2}$$

同理，定义$score_{\mathrm{mem}}$为节点上内存使用情况得分；$score_{\mathrm{disk}}$为节点上磁盘使用情况得分；$score_{\mathrm{gpu}}$为节点上GPU使用情况得分。具体计算分别对应式(3)、式(4)、式(5)，其中T为节点上某个资源的总量，U为节点上某个资源的已使用量。

$$score_{\mathrm{mem}}=\frac{T_{\mathrm{mem}}-U_{\mathrm{mem}}}{T_{\mathrm{mem}}} \tag{3}$$

$$score_{\mathrm{disk}}=\frac{T_{\mathrm{disk}}-U_{\mathrm{disk}}}{T_{disk}} \tag{4}$$

$$score_{\mathrm{gpu}}=\frac{T_{\mathrm{gpu}}-U_{\mathrm{gpu}}}{T_{\mathrm{gpu}}} \tag{5}$$

采用相同的资源类型(CPU、内存、磁盘、GPU)分别表示Pod与节点上的资源情况，定义Pod向量、节点向量如公式(6)、公式(7)所示，分别代表Pod各类资源请求情况、节点上各类资源剩余情况。

$$\overrightarrow{Pod}=(req_{\mathrm{cpu}},req_{\mathrm{mem}},req_{\mathrm{disk}},req_{\mathrm{gpu}}) \tag{6}$$

$$\overrightarrow{Node}=(rest_{\mathrm{cpu}},rest_{\mathrm{mem}},rest_{\mathrm{disk}},rest_{\mathrm{gpu}}) \tag{7}$$

使用夹角余弦度量两个向量之间的相似度，并以$score_{\mathrm{match}}$表示Pod应用请求资源与节点剩余资源的相似程度，即Pod应用与节点的匹配程度。$score_{\mathrm{match}}$值越大代表向量相似度越高，应用与节点的匹配度越高；反之匹配度越低。$score_{\mathrm{match}}$具体计算如公式(8)所示：

$$score_{\mathrm{match}}=\frac{\overrightarrow{Pod}\bullet\overrightarrow{Node}}{\left\|\overrightarrow{Pod}\right\|\left\|\overrightarrow{Node}\right\|} \tag{8}$$

综合以上分析，本文设计的节点资源可用性评分公式如(9)所示：

$$score=w1\times score_{\mathrm{cpu}}+w2\times score_{\mathrm{mem}}+w3\times score_{\mathrm{disk}}+w4\times score_{\mathrm{gpu}}+w5\times score_{\mathrm{match}} \tag{9}$$

其中，w代表各类资源对应的权重值。本文将CPU、GPU、内存、磁盘这三类资源的权重设置为m，将GPU以及匹配程度的权重设置为4 m[22]。

2.4　资源调度单元功能设计

资源调度指的是为Pod分配好其所需的资源后，将Pod调度到集群中某个合适的节点上去的过程[23]。将Pod调度到所选节点上后，还需要根据资源分配结果，将已分配好的资源进行物理占用绑定。节点的各项资源调度途径如图5所示。

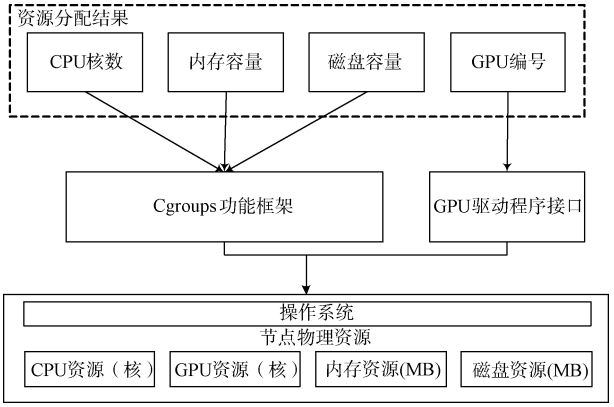

图5　节点资源调度途径示意图

Fig. 5　Schematic diagram of the node resource scheduling pathway

对于GPU资源来说，为保证设备级资源队列的有效性，必须保证GPU编号与物理GPU卡的确

定关系。建立GPU资源逻辑序号与GPU卡所在PCIE卡槽的映射关系表，根据所需的GPU编号，按照映射关系通过调用驱动接口实现对GPU核的绑定。

对于CPU、内存和磁盘资源来说，Cgroups功能框架可以对其进行物理占用[24]。Cgroups是Linux内核提供的一种可以限制进程使用资源的机制，为每种可以控制的资源定义了子系统。Cgroups功能框架通过进程调度模块根据CPU子系统的配置指定CPU核，通过内存子系统和磁盘子系统的配置限制内存和磁盘资源。节点资源的调度方式如表4所示，CPU与GPU按核调度，内存与磁盘资源根据容量需求动态分配。

表4 节点内计算资源调度方式

Table 4 Node resource scheduling approach

资源类型	调度途径	调度策略
CPU	Cgroups功能框架	根据核数需求，按空闲CPU核顺序分配、调度
GPU	GPU驱动程序接口	根据所需的GPU编号，按照映射关系表调度
内存	Cgroups功能框架	根据容量需求动态分配
磁盘	Cgroups功能框架	根据容量需求动态分配

3 仿真实例与结果分析

本节重点针对信号处理资源，模拟仿真应用实例，分析多级队列择优策略的调度能力，并通过实验进一步测试GPU资源利用率和任务调度时间，验证本文调度算法的性能。

3.1 资源调度能力分析

资源池系统中测控任务处于动态调度中，各类任务资源需求不同，单纯以数字指标衡量系统的总体资源利用率具有偶然性和局限性，本文设计以下的仿真测试用例，模拟任务执行情景，评估本文提出的多级队列择优的资源调度策略。

表5记录了测试用例中每个任务的代号、类型以及GPU、CPU需求数量，在地面测控任务中，对调度起主导限制作用的是GPU资源，因此本测试用例中暂不考虑内存和磁盘资源。定义模拟测试环境有三个节点，每个节点具有8个GPU核、64个CPU核可供调用。

表5 模拟任务测试用例

Table 5 Simulated task test cases

任务代号	任务类型	GPU需求/个	CPU需求/个
0ZYCKJ	测控	2	4
1ZYCKJ	数传	2	4
2ZYCKJ	数传	2	4
3ZYCKJ	测控	4	8
4ZYCKJ	测控	8	10

根据上文的单元资源分配策略，调度结果如图6所示，测试中不考虑特殊工程需求，三个节点属性完全一致，皆通过预选阶段成为备选节点；对于0ZYCKJ测控任务，三个节点优选分数相同，任务按序调度至Master-1节点；对于1ZYCKJ数传任务，通过上文设计的优选打分公式计算可得Master-1节点的最终得分 $score_1 \approx 0.6$；Master-2节点的最终得分 $score_2 \approx 0.559$。可以看出 $score_1 > score_2$，1ZYCKJ数传任务调度至Master-1节点。同理可计算出2ZYCKJ数传任务应调度至Master-1节点。此时Master-1节点仅剩两个GPU核，不满足资源需求，无法通过预选，3ZYCKJ测控任务和4ZYCKJ测控任务分别调度至Master-2、Master-3节点。

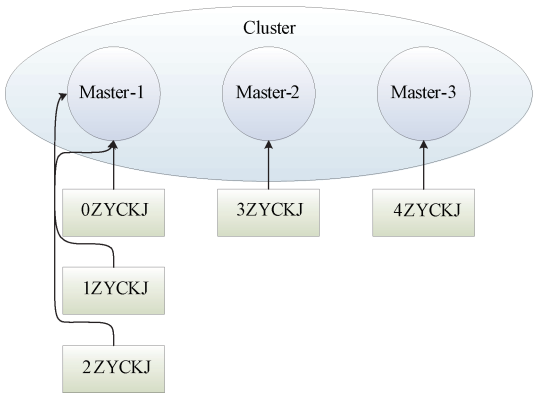

图6 本文策略下资源分配结果

Fig. 6 The results of resource allocation under the strategy of this paper

若使用传统平铺分配策略，分配结果如图7所示。平铺策略的计算原则是尽量将Pod调度到计算资源占用比较小的节点上，因此0ZYCKJ测控任务、1ZYCKJ数传任务和2ZYCKJ数传任务按顺序分别平铺调度至三个节点上，3ZYCKJ测控任务调度至Master-1节点，此时三个节点的剩余GPU资源分别为2、6、6，均不满足4ZYCKJ测控任务需求，无可用节点，已有资源无法执行4ZYCKJ测控任务。

可以看出，同样的任务资源需求，使用传统

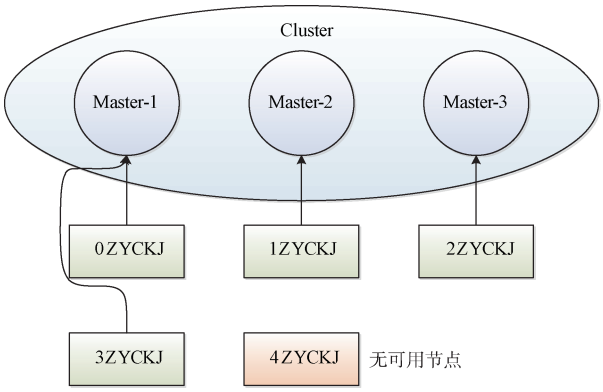

图 7　传统平铺策略下资源分配结果

Fig. 7　The results of resource allocation under the strategy of traditional tiling strategy

平铺策略需要具备四个节点方能调度完成，而使用本文设计的单元资源分配策略，只需三个节点便能完成调度。系统资源的使用率与任务调度情况和任务所需资源有关，通过测试可以看出，对于同样的任务执行情况，本文策略的资源使用率更高、调度能力更强。

3.2　节点 GPU 利用率测试

在仿真实例分析的基础上，为进一步验证算法性能，本文使用两台 Linux 服务器搭建 Kubernetes 集群作为实验平台，分别测试默认调度算法和本文设计的多级队列择优调度算法作用下的 GPU 利用率情况和资源调度时间，进行对比分析，其中 Kubernetes 版本为 1.18.6，Docker 版本为 19.03.9。

本实验将一些常见测控任务组成任务集，在综合运管平台中设置好任务的开始和结束时间，让任务集根据算法进行自动化调度。在每个节点上每隔 180 s 统计 GPU 的资源状态，计算 GPU 资源的利用率，实验结果如表 6 所示。

实验数据显示，相较于默认调度器，使用自定义调度策略能够让 GPU 利用率更高，原因是自定义调度策略能够有效减少资源碎片的产生，充分利用 GPU 资源，在相同条件下能够满足更多的任务需求。根据实验结果，使用 K8s 默认调度策略时 GPU 平均利用率为 31.86%，使用自定义优化策略时 GPU 平均利用率为 44.0%，相较默认策略将节点的 GPU 利用率提升了 12.14%。

3.3　资源调度时间测试

本实验通过综合运管平台的日志显示功能[25]捕捉了 8 个任务分别在默认调度策略和自定义调度策略下的资源调度时间，实验结果如图 8 所示。

表 6　各节点 GPU 利用率统计情况

Table 6　GPU utilization statistics by node

时间/s	默认调度策略 GPU 利用率/%	自定义调度策略 GPU 利用率/%
0	0	0
180	12.3	45.1
360	24.7	46.8
540	24.5	45.6
720	21.6	39.5
900	46.1	58.6
1 080	42.3	38.7
1 260	39.2	42.3
1 440	37.6	38.1
1 620	36.2	38.1
1 800	34.1	47.2

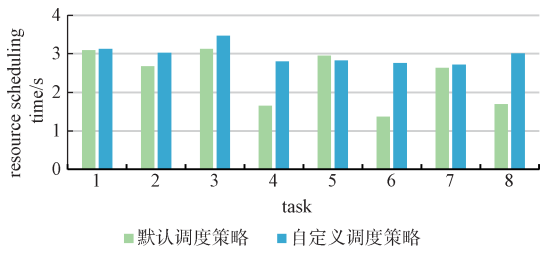

图 8　资源调度时间实验结果

Fig. 8　Experimental results on resource scheduling time

从图 8 可以看出：大部分情况下两种策略的调度时间十分接近，均为秒级，只有 4 号、6 号及 8 号任务，在自定义调度策略下的调度时间远小于默认调度器。产生这种现象的原因是这三个任务均为历史任务，在资源调度时成功通过设备级资源队列匹配到了任务执行所需资源，不需要重新进行单元级资源分配。实验结果表明，对于系统中已经出现过的历史任务，使用多级队列择优的调度策略可以有效地提高资源调度的执行速度，提高任务的执行效率。

4　结束语

本文设计的基于多级队列择优的资源池信号处理资源调度策略，充分结合资源池架构和多级调度方法，聚合了资源的监控、分配和调度功能，适用于资源池系统信号处理资源的集中管理和统一分配。该方法解决了当前资源池系统面临的资源空闲率高、任务执行效率低等难题。针对资源调度能力和执行速度的测试结果表明，该方法基

本满足信号处理资源的管理调度需求。未来的资源池调度策略可以在智能化方面做进一步研究，探索任务资源调度失败自动重启、漏洞自查和问题分析处置等处置策略。

参考文献

[1] 雷厉, 朱勤专. 飞行器测控通信技术发展趋势与建议[J]. 飞行器测控学报, 2014, 33(6): 463-468.
LEI Li, ZHU Qinzhuan. Analysis of the trend of development of spacecraft TT&C and communication technologies and suggestions[J]. Journal of Spacecraft TT&C Technology, 2014, 33(6): 463-468.

[2] 陈峰, 武小悦. 天地测控资源调度的两阶段递进遗传算法[J]. 国防科技大学学报, 2010, 32(2): 17-22.
CHEN Feng, WU Xiaoyue. Two-stage successive genetic algorithm for space and ground TT&C scheduling[J]. Journal of National University of Defense Technology, 2010, 32(2): 17-22.

[3] 薛乃阳, 丁丹, 王红敏, 等. 引入微元法思想的混合测控资源联合调度方法[J]. 系统仿真学报, 2022, 34(4): 826-835.
XUE Naiyang, DING Dan, WANG Hongmin, et al. Idea of infinitesimal method-introduced hybrid TT&C resources joint scheduling[J]. Journal of System Simulation, 2022, 34(4): 826-835.

[4] 刘建平, 李晶, 张天骄. 航天测控网调度的混合构造启发式算法[J]. 系统工程与电子技术, 2015, 37(7): 1569-1574.
LIU Jianping, LI Jing, ZHANG Tianjiao. Hybrid constructive heuristics of space measurement and control network scheduling problem[J]. Systems Engineering and Electronics, 2015, 37(7): 1569-1574.

[5] LI Z Y, MING W, LIU J P, et al. Analysis of ground station network resources for giant constellation TT&C service[C]//Proc. of the IEEE 3rd International Conference on Electronic Information and Communication Technology. 2020: 6-11.

[6] XHAFA F, IP A W H. Optimisation problems and resolution methods in satellite scheduling and space-craft operation: A survey[J]. Enterprise Information Systems, 2019, 15(1): 1-24.

[7] 吴逸文, 张洋, 王涛, 等. 从Docker容器看容器技术的发展: 一种系统文献综述的视角[J]. 软件学报, 2023, 34(12): 5527-5551.
WU Yiwen, ZHANG Yang, WANG Tao, et al. Development exploration of container technology through Docker containers: A systematic literature review perspective[J]. Journal of Software, 2023, 34(12): 5527-5551.

[8] 刘瑞奇, 李博扬, 高玉金, 等. 新型分布式计算系统中的异构任务调度框架[J]. 软件学报, 2022, 33(3): 1005-1017.
LIU Ruiqi, LI Boyang, GAO Yujin, et al. Heterogeneous task scheduling framework in emerging distributed computing systems[J]. Journal of Software, 2022, 33(3): 1005-1017.

[9] 王友祥, 李轶群, 张澜, 等. 基站虚拟化技术研究[J]. 邮电设计技术, 2016(11): 47-50.
WANG Youxiang, LI Yiqun, ZHANG Lan, et al. Research on base station virtualization technology[J]. Designing Techniques of Posts and Telecommunications, 2016(11): 47-50.

[10] 彭永强. 分布式综合任务调度平台的设计与应用[D]. 北京: 北京交通大学, 2019.

[11] 左利云, 曹志波. 云计算中调度问题研究综述[J]. 计算机应用研究, 2012, 29(11): 4023-4027.
ZUO Liyun, CAO Zhibo. Review of scheduling research in cloud computing[J]. Application Research of Computers, 2012, 29(11): 4023-4027.

[12] 彭丽苹, 吕晓丹, 蒋朝惠, 等. 基于Docker的云资源弹性调度策略[J]. 计算机应用, 2018, 38(2): 557-562.
PENG Liping, LYU Xiaodan, JIANG Chaohui, et al. Elastic scheduling strategy for cloud resource based on Docker[J]. Journal of Computer Applications, 2018, 38(2): 557-562.

[13] KANG D, CHOI G, KIM S, et al. Workload-aware resource management for energy efficient heterogeneous Docker containers[C]// 2016 IEEE Region 10 Conference (TENCON), Singapore. 2016: 2428-2431.

[14] 杜帅. 面向Kubernetes的容器资源调度研究[D]. 徐州: 中国矿业大学, 2023.

[15] 李天宇. 基于强化学习的云计算资源调度策略研究[J]. 上海电力学院学报, 2019, 35(4): 399-403.
LI Tianyu. Research on cloud computing resource scheduling strategy based on reinforcement learning[J]. Journal of Shanghai University of Electric Power, 2019, 35(4): 399-403.

[16] 李轲, 窦亮, 杨静. 面向Kubernetes的多集群资源监控方案[J]. 计算机系统应用, 2022, 31(7): 77-84.
LI Ke, DOU Liang, YANG Jing. Multi-cluster resource monitoring scheme for Kubernetes[J]. Computer Systems & Applications, 2022, 31(7): 77-84.

[17] 耿棒棒, 王勇. 基于改进秃鹰搜索算法的Kubernetes资源调度应用[J]. 计算机系统应用, 2023, 32(4): 187-196.
GENG Bangbang, WANG Yong. Improved bald eagle search algorithm for Kubernetes resource scheduling application[J]. Computer Systems & Applications, 2023, 32(4): 187-196.

[18] 胡程鹏, 薛涛. 基于遗传算法的Kubernetes资源调度算法[J]. 计算机系统应用, 2021, 30(9): 152-160.
HU Chengpeng, XUE Tao. Kubernetes resource scheduling algorithm based on genetic algorithm[J]. Computer Systems & Applications, 2021, 30(9): 152-160.

[19] 常旭征, 焦文彬. Kubernetes资源调度算法的改进与实现[J]. 计算机系统应用, 2020, 29(7): 256-259.
CHANG Xuzheng, JIAO Wenbin. Improvement and implementation of Kubernetes resource scheduling algorithm[J]. Computer Systems & Applications, 2020, 29(7): 256-259.

[20] 郭玥鑫. 基于战场分析系统的K8s云平台设计与实现[D]. 西安: 西安工业大学, 2022.

[21] VOLKOVA V N, CHEMENKAYA L V, DESYATIRIKOVA E N, et al. Load balancing in cloud computing [C]// 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering(EIConRus), Moscow and St. Petersburg, Russia. 2018: 387-390.

[22] 邓雪, 李家铭, 曾浩健, 等. 层次分析法权重计算方法分析及其应用研究[J]. 数学的实践与认识, 2012, 42(7): 93-100.
DENG Xue, LI Jiaming, ZENG Haojian, et al. Research on computation methods of AHP wight vector and its applications[J]. Mathematics in Practice and Theory, 2012, 42(7): 93-100.

[23] 赵鹏, 刘畅, 贾智宇, 等. 高可用环境下的服务网格部署实践[J]. 信息通信技术, 2021, 15(3): 42-47.
ZHAO Peng, LIU Chang, JIA Zhiyu, et al. Service mesh deployment practice in a high-availability environment [J]. Information and Communications Technologies, 2021, 15(3): 42-47.

[24] 大叶子不小. Linux资源管理之cgroups简介[EB/OL]. (2022-10-17) [2024-4-14]. https://blog. csdn. net/qq_32907195/article/details/127371332.

[25] 马潇潇, 孙甲琦, 朱宏涛, 等. 基于容器云的地面测控资源池日志管理系统的研究[J]. 遥测遥控, 2023, 44(6): 57-63.
MA Xiaoxiao, SUN Jiaqi, ZHU Hongtao, et al. Research of ground TT&C resource pool log management system based on container cloud[J]. Journal of Telemetry, Tracking and Command, 2023, 44(6): 57-63.

[作者简介]

马潇潇　1999年生，硕士研究生。
孙甲琦　1976年生，博士，研究员。
朱宏涛　1977年生，硕士，研究员。
杨罗兰　1992年生，硕士，工程师。

(本文编辑：傅　杰)
(英文编辑：赵尹默)