# 高性能云原生大数据平台设计与实现

林 健,黄 林,黄进军,谢冬鸣,洪志刚

(东云睿连(武汉)计算技术有限公司,湖北 武汉 430200)

**摘 要:** 传统大数据平台软件历经十余年发展,已日趋成熟。近年来,以容器化为主要特征的云原生架构成为基础设施建设的首选方案。尤其在高性能计算技术融入云原生环境的趋势下,新一代大数据平台设计也面临着不同的挑战。这些挑战涉及云原生环境下的作业调度、高性能网络的容器化适配、存算分离架构下的存储管理等。针对这些问题,提出一组高性能云原生大数据平台关键技术,包括多模式负载容器化调度技术、容器化RDMA数据交换技术、云原生存算分离架构等,并在此基础上研发了OMBD大数据平台。OMBD能够有效适配高性能云原生环境的特点,以体系化的技术方案使得多模式大数据作业在带有高性能网卡的容器化集群中实现有效调度和高效执行。实验数据和真实应用效果证明,OMBD是一款具有实用性和高效性的生产级大数据平台。

**关键词:** 大数据;云原生;高性能计算;容器化;平台软件

**DOI:** 10. 11907/rjdk. 231049 **开放科学(资源服务)标识码(OSID):**

**中图分类号:** TP274 **文献标识码:** A **文章编号:** 1672-7800(2024)003-0099-08

# Design and Implementation of a High-Performance Cloud-Native Big Data Platform

LIN Jian, HUANG Lin, HUANG Jinjun, XIE Dongming, HONG Zhigang

(*Oriental Mind* (*Wuhan*) *Computing Technology Co.*, *Ltd.*, *Wuhan* 430200, *China*)

**Abstract:** After more than ten years, the traditional big data platform software has become increasingly mature.In recent years, the cloud-native architecture featuring containerization has become the preferred solution for infrastructure construction. Especially under the trend of integrating high-performance computing technology into the cloud-native environment, the design of a new generation of big data platforms is facing different challenges. These challenges involve job scheduling in a cloud-native environment, containerized adaptation of high-performance networks, and storage management under a storage-compute separation architecture. In response to these problems, this paper proposes a set of key technologies for a high-performance cloud-native big data platform, including the multi-mode workload containerized scheduling technology, the containerized RDMA data exchange technology, and the cloud-native storage-compute separation architecture. And on this basis, developed the OMBD big data platform. OMBD can effectively adapt to the characteristics of high-performance cloud native environment, and realize effective scheduling and efficient execution of multi-mode big data jobs in containerized clusters with high-performance network cards with systematic technical solutions. Experimental data and real-world application results prove that OMBD is a practical and efficient production-grade big data platform.

**Key Words:** big data; cloud-native; high-performance computing; containerization; platform software

## 0 引言

随着互联网、物联网、人工智能等行业的高速发展,大数据技术在社会生产生活的诸多领域都得到广泛应用,并创造出可观的经济价值。与此同时,云计算技术也成为大数据系统与算法发展的重要助推器,它帮助大数据业务实现大规模集群下的按需服务和弹性伸缩,使得大数据技术

**收稿日期:** 2023-01-29

**基金项目:** 武汉东湖新技术开发区第十三批"3551光谷人才计划"支持项目(M165)

**软著编号:** 2021SR0509412;2021SR1279980

**作者简介:** 林健(1985-),男,博士,CCF高级会员,东云睿连(武汉)计算技术有限公司总经理、高级工程师,研究方向为计算机系统结构;黄林(1990-),男,硕士,CCF专业会员,东云睿连(武汉)计算技术有限公司软件工程师,研究方向为电子与通信工程;黄进军(1985-),男,东云睿连(武汉)计算技术有限公司软件工程师,研究方向为电子信息工程;谢冬鸣(1982-),男,东云睿连(武汉)计算技术有限公司软件架构师,研究方向为通信工程;洪志刚(1984-),男,东云睿连(武汉)计算技术有限公司软件架构师,研究方向为通信工程。本文通讯作者:林健。

成为一种可灵活购买和部署的在线服务。

云计算技术在近年来的新发展趋势是“云原生”(Cloud-Native)[1],即软件全栈架构从规划之初就专门针对云上运行而设计。在技术实现方面,云原生以容器化、服务网格、开发运维一体化(DevOps)等为主要特征。对云服务提供商而言,云原生技术具有高度灵活性、高资源利用率、低运维成本等显著优势。因此,新一代云服务出现了向云原生架构迁移的趋势。

大数据平台的演化也不例外。业界经历了起初私有化部署的大数据平台,以及后来的大数据云服务,目前正在迎接云原生大数据时代。然而,由于大数据与云原生2个技术栈各自的固有特性差异与历史包袱问题,当前云原生大数据平台发展尚不成熟。尤其是在高性能计算技术不断融入云环境的背景下,如何构建具有高性能特征的云原生大数据平台也成为新的挑战。

## 1 大数据平台发展概览

早期大数据系统软件主要是支持单一计算模式的大数据框架,例如以批式MapReduce计算为核心模式的Hadoop、以类SQL即席查询为核心模式的Hive等。这类框架通常具有各自独立的作业调度、分布式存储等外围服务,并且主要面向局域网私有化环境。在独立框架基础上,以CDH(Cloudera’s Distribution Including Apache Hadoop)、Hortonworks数据平台(Hortonworks Data Platform,HDP)为代表的大数据框架集成发行版形成了初代大数据平台软件。它们并非简单打包各个框架,而是将框架的共性能力作为公共服务部署,将可协同工作框架的业务流程贯通,并提供具有全局视图的用户界面,进而实现了大数据能力的平台化。

随着云计算技术的成熟,以亚马逊(Elastic MapReduce,EMR)、华为云(MapReduce Service,MRS)为代表的大数据云服务平台开始涌现。它们并非简单地将大数据框架部署在基础设施即服务(Infrastructure as a Service,IaaS)集群之上,而是将框架封装为按需购买、在线使用的平台即服务(Platform as a Service,PaaS),并且将存储、网络、鉴权、日志等共性能力对接到其他专业化的云服务组件上,以实现底层基础设施的高可靠和上层接口的一致性。大数据云服务平台除具有公有云版本外,也存在私有云版本,例如华为云FusionInsight MRS。私有云版本为传统局域网大数据平台增添了自动部署、弹性伸缩等云服务特有能力。

云原生技术趋势下出现的大数据平台大致可分为3类:①以Snowflake[2]为代表的全新设计的大数据服务,它们没有过多的技术包袱,因此可以充分发挥云原生架构的优势,但仍需要应用软件加以优化适配才能取得最佳性能;②以亚马逊Lambda为代表的“函数即服务”(Function as a Service,FaaS)[3],它们将数据处理生命周期拆解为细粒度的函数,通过合理的调度机制优化计算性能与资源利用率,然而编程范式的改变会对大数据算法开发带来挑战;③本文关注的基于容器技术构建的“容器即服务”(Container as a Service,CaaS),早期典型代表是Kubernetes生态中的批处理调度插件[4]和大数据框架Operator[5],期望将Hadoop等经典大数据框架的系统服务和作业进程以容器化方式执行,为此需要解决若干技术适配问题。

## 2 相关研究

在学术界,Zhu等[6]分析了Spark框架在容器化前后的性能特征,但其Spark on Kubernetes方案简单使用开源软件原生能力,不满足大数据平台的特性要求。黄凯旋[7]研究了大数据框架与容器服务2个层面的调度器兼容设计及其性能优化问题,并提出一套支持批处理作业容器化调度的原型平台。张丽敏[8]关注大数据系统服务而非作业进程的容器化调度问题,提出基于Kubernetes资源监控信息的优化调度算法。

在工业界,XenonStack[9]提出在Kubernetes集群上运行大数据系统服务的初步思路,但其在作业调度方面仍使用传统的YARN方式,因而不能称为一套完善的云原生平台。星环[10]提出以容器集群作为资源调度层的“大数据3.0”理念,并研发了Venus统一调度器,重点解决混合负载和多租支持等问题。NetApp[11]推出的Astra服务则以多云应用托管方式,管理Kubernetes集群上的大数据框架,相比之下控制粒度较粗,不涉及容器级调度。

这些工作兼容大数据框架原有接口的同时,部分地获得了云原生架构在资源分区灵活性、运维便利性等方面的优势。然而,总结现有工作可以发现,它们并未完全解决大数据技术在容器化时面临的各项问题与挑战。

## 3 问题与挑战

在采用CaaS方式构建大数据平台时,技术问题可从运行时调度、网络接入和存储管理3个维度展开。其均与大数据作业所需的外围支撑能力相关,是云原生架构对运行环境带来的主要变化所在。相比之下,大数据作业内部算法(计算模型维度)则可以被容器化机制透明地封装,友好地迁移到云原生环境。

### 3.1 云原生环境下的运行时调度问题

大数据框架的运行时实体包括系统服务和作业进程。系统服务一般以常驻后台服务方式运行,作业进程则通过调度器加载,动态执行。二者均具有跨节点分布式执行的特性,作业进程还具有与计算模型相匹配的分布式运行模式,例如批式作业对应的有状态群调度模式、即席查询作业前端组件对应的多副本无状态模式等。在非云环境下,

系统服务的部署和配置由人工或脚本完成,作业进程的调度由大数据框架自带的调度服务完成,框架对二者的基本假设是处于静态的局域网环境。在经典云服务模式下,虚拟机或裸金属服务器取代了本地节点、虚拟私有网络取代了局域网,环境因而具备了弹性,但计算和网络资源对上层业务提供的接口没有质变,因此框架运行时的调度模式无须显著改变。

云原生环境下,容器成为一等公民。如果简单地将大数据框架中原先对应到节点的概念生硬地映射到容器,那么无疑是一种低效的蹩脚设计。生硬映射的问题包括容器网络动态网络地址与系统服务静态配置要求的矛盾、容器内部存储易失性与系统服务及作业进程中间数据持久性要求的矛盾、容器内单一主进程的最佳实践与大数据框架单节点内多进程协同的矛盾等。同时,生硬映射也无法将云原生架构资源的分区灵活性、运维便利性等优势充分发挥。因此,云原生环境下的运行时调度有必要重新设计。

云原生环境下的运行时调度设计需要解决的主要问题包括:①如何管理传统上不属于调度器负责的系统服务,使得大数据框架的所有运行时实体以合理的方式容器化,从而发挥云原生架构的优势;②如何满足多模式大数据作业的调度需求,将调度能力作为框架间兼容的公共服务部署,从而可以统一调度批式、流式、即席查询等作业,以提升资源利用率和灵活性。

### 3.2　高性能网络的容器化适配问题

在商用集群中引入 InfiniBand 或基于融合以太网的 RDMA(RDMA over Converged Ethernet,RoCE)等支持远程直接数据存取(Remote Direct Memory Access,RDMA)协议的高性能网络,是近年来大数据云服务发展的新趋势[12]。这一技术通过提升带宽利用率并降低通信延迟,能够提高大规模分布式作业的执行效率。高性能网络、容器化封装和大数据框架三者的两两组合已有若干研究成果和业界产品。例如,Kubernetes 生态的设备插件能够在容器集群中实现 RDMA 网络的容器化接入,大数据生态的 YARN on Docker[13]解决方案能够将某些大数据作业以容器化方式调度,而 Mellanox SparkRDMA[14]等设计则为经典大数据框架增添了对高性能网络的支持。

然而,3 项技术协同工作时,则会遇到新的问题,例如:①大数据框架一般需要在启动时绑定具体的 RDMA 设备,而容器集群的动态调度特性使得实时的硬编码绑定不再可行;②RDMA 通信需要使用锁页内存,而这一特性在容器化环境下依赖特权容器,这与容器集群下的多租安全设计存在冲突。这些问题制约了高性能云原生大数据平台的设计与实现。

独立混洗(Shuffle)服务在大数据领域的应用愈发广泛。相比传统上集成在大数据框架内部的混洗机制,独立混洗服务对计算进程的本地存储要求更低,更加适合于存算分离架构。同时,独立混洗服务对组件职责的清晰划分也有利于对大数据平台实施系统性优化。在云原生架构中引入高性能网络后,自然需要考虑独立混洗服务与 RDMA 协议的适配问题。

### 3.3　存算分离架构下的存储管理问题

存算分离架构是大数据平台设计的一项新趋势[15]。在这一架构下,存储集群独立于计算集群建设,能够实现存储空间在大数据框架间的弹性分配、按需伸缩,在优化空间利用率的同时有利于计算与存储软件栈的独立演进。这一架构能够取得性能优势的前提包括高效的存算集群间通信网络、低开销的存储访问接口、与应用适配的本地缓存机制等。

在云原生环境中,常见的存算分离设计方案是将计算组件以容器化方式调度运行,同时部署一套公共存储集群(例如 Ceph)。公共存储集群提供包括可移植操作系统接口(Portable Operating System Interface of UNIX,POSIX)文件系统、对象存储(Object-based Storage)、Hadoop 分布式文件系统(Hadoop Distributed File System,HDFS)在内的多类兼容接口,以逻辑多租方式分配给不同的计算组件使用。这一方案在大规模公有云服务中较为适用,然而在小规模专有集群中,建设并运维一套公共存储集群并为之配套存算集群间高性能通信网络的难度大、成本高。因此,有必要探索一种新的云原生存算分离架构,使得存储组件也能够受益于云原生容器化架构的优势。

在存算分离架构下,为了克服计算组件远程访问存储服务所带来的性能开销,特别是多作业、多框架并存时的网络性能干扰,计算集群侧的缓存机制往往不可或缺。常见的缓存服务包括 Alluxio、JuiceFS 等。缓存服务在容器化环境中如何实现灵活部署、如何与计算组件高效通信、如何选取合理的本地存储方案,均是在云原生技术背景下实施存算分离架构时需要回答的问题。

## 4　高性能云原生大数据平台关键技术

在高性能计算技术融入云原生环境的背景下,为了充分发挥两个生态各自的优势,有必要通过一系列关键技术解决已知问题,构建新的大数据平台架构。

### 4.1　多模式负载容器化调度技术

针对云原生环境下的运行时调度问题,本文提出多模式负载容器化调度技术。该技术主要涉及后端的框架调度器封装机制与前端的统一作业抽象机制。

#### 4.1.1　框架调度器封装机制

框架调度器封装机制[16]提出的目的是通过统一的机制管理多类大数据框架中的调度类系统服务,使之与作业进程一并成为容器集群服务管理的对象,从而实现大数据平台与大数据框架的解耦,提升二者独立演进的能力。大数据框架的系统服务中,作业调度类能力可细分为 2 部

分:①作业间调度和作业内调度。框架调度器封装机制引入后,原有的作业间调度组件不再使用,而是改由容器集群服务的调度组件接管;②作业内调度组件则封装在作业镜像内,以容器方式运行,每个作业对应一套生命周期与之相同的专属调度组件实例。图 1 以 Spark 框架为例,给出了本机制涉及的两类调度的关系示意图。

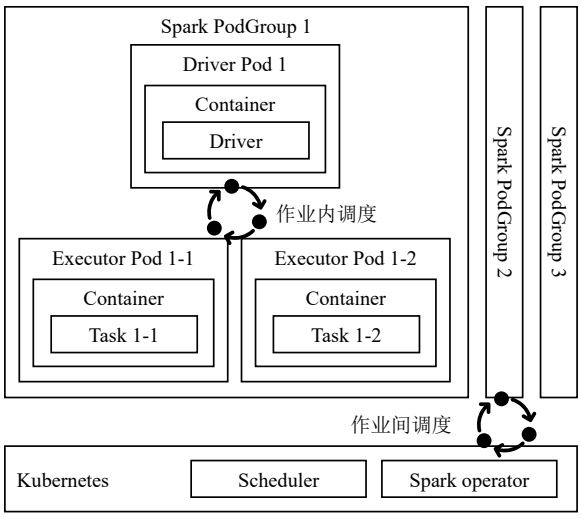


**Fig. 1 Relationship between the two types of scheduling involved by the framework scheduler encapsulation mechanism**

**图 1 框架调度器封装机制涉及的两类调度之间的关系**

作业间调度组件改在容器集群层面实现,这使得不同框架的作业可以在同一容器集群上共存,接受统一的资源分配和优先级管理,解决不同框架自有的调度器资源视图不统一、调度策略不兼容等问题。在具体实现时,采取 2 种方式:①对于运行模式简单的框架,基于 Kubernetes 生态中成熟的批处理调度组件 Volcano[4],编写上层封装脚本将作业的具体调度需求映射到容器集群的底层机制上;②对于运行模式复杂的框架,则基于 Kubernetes 的 Operator 机制,研发框架专用的 Operator 以管理作业调度的各项逻辑。不同于现有多种只顾及单一框架甚至单一作业实例的开源 Operator,本文提出的 Operator 需要考虑并解决多种框架及多作业实例共存时的兼容性问题,例如在使用主机网络时的端口分配问题。

作业内调度组件包含大数据框架原生调度组件(例如 Spark 的 Driver)及外围封装组件,它们均在容器内部运行。Kubernetes 集群上的每个作业经由 Operator 启动后,成为一个由自定义资源(Custom Resource Definition,CRD)定义的独立实体,其本质为一组接受统一管理的容器集合。容器集合的隔离作用一方面允许将原生调度组件精简到最小集合,另一方面支持同一框架不同版本的作业在同一容器集群上共存。原生调度组件可见的资源范围仅限于这组容器集合,因此能够受控地使用分配给本作业的资源容器以运行每个计算进程。外围封装组件则负责将动态分配资源的容器集合模拟为传统集群环境,其主要工作包括配置节点列表文件、执行栅障同步等。

#### 4.1.2 统一作业抽象机制

统一作业抽象机制用于规范多种不同大数据框架上的多模式作业的元信息描述方式和生命周期管理接口,从而简化上层组件及开发者调用大数据平台能力的操作。通过提炼业务语义、隐藏技术细节,为大数据业务调度管理的高效性和稳定性提供保障。这一抽象对开发者提供一种基于声明式领域专用语言(YAML 或 JSON)的作业描述机制,其中描述了作业的基本信息、框架和调度模式、任务规模、资源需求等。此外,抽象机制还涉及对作业启动、停止、恢复等语义的统一管理接口,便于上层组件集成。

统一作业抽象机制中的框架和调度模式信息是本机制实现高度抽象的关键。这里将常用大数据框架的作业总结为若干种业务模式(例如批式、流式、即席查询等),再将业务模式细分为多种实现模式(例如无状态多副本服务、有状态多任务作业等)的有序组合。每种框架对应的具体组合方式由统一作业抽象机制的后台实现决定,并支持多框架级联组合,实现细节对调用者透明。每种实现模式在容器集群层面的调度策略最终构建出了业务模式在应用层所需的调度语义。对于平台尚不支持的新框架,调用者也可以使用这些实现模式自由组合,满足其容器化调度需求。

以 Hive 框架为例,开发者可以在作业描述语言中将调度模式描述为“即席查询”,并且选择性地给出前端(HiveServer2)和后端(Hadoop/Spark)集群的规模和资源说明,本机制即可以自动创建容器化的 Hive 及其级联的各项服务,并执行应用层输入的 Hibernate 查询语言(Hibernate Query Language,HQL)查询。Hive 的客户端组件、前端服务和后端集群各自的容器化封装方式、调度策略、状态维护策略则无须开发者显式编写。

### 4.2 容器化 RDMA 数据交换技术

针对高性能网络的容器化适配问题,本文提出容器化 RDMA 数据交换技术。该技术主要涉及 RDMA 容器化接入机制和 RDMA 数据混洗机制。

#### 4.2.1 RDMA 容器化接入机制

在以 Kubernetes 为代表的云原生容器化环境中引入 RDMA 高性能网卡并对上层提供多租复用能力的路径有多种,其中主要的决策问题包括:①直接挂载并复用网卡设备(Host Channel Adapter,HCA)还是基于单根输入/输出虚拟化(Single Root I/O Virtualization,SR-IOV)对网卡做虚拟化;②在虚拟机还是容器层面对网卡做复用。现有的 Mellanox 插件[17]提供了在物理机或虚拟机上实现容器间复用 HCA 的方案,缺点在于容器网络间没有安全隔离和性能隔离。Intel 插件[18]则提供了将物理机上的 HCA 经由 SR-IOV 虚拟化之后挂载在容器中的方案,其具有隔离性,但不适用于基本输入输出系统(Basic Input Output System,BIOS)且不支持 SR-IOV 的虚拟机环境,因而对云基础设施不够友好。此外,RDMA 容器化方案还需要考虑与高性能

TCP/IP 网络(包括 InfiniBand 的 IPoIB 网络和 RoCE 的原生 IP 网络)的兼容性。

本文提出一种新的 SR-IOV 容器网络接口(Container Network Interface,CNI)插件[19],该插件调度宿主环境中的 IaaS 服务,在物理机层面对高性能网卡实施虚拟化,将虚拟功能(Virtual Function,VF)按需分配给虚拟机;然后,通过 etcd 共享存储记录虚拟机环境中的 VF 可用性,依据作业资源需求在容器层面对 VF 进行动态挂载,以满足作业接入高性能网络的要求,由此形成的设备映射关系如图 2 所示。同时,该插件将 VF 上的高性能 TCP/IP 网络直通到容器内,并针对每个作业的子网路由配置,实现双栈网络可用。这一机制兼顾了隔离性和对云原生环境的兼容性。

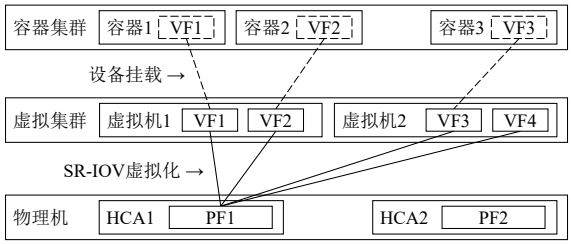


**Fig. 2　Example of the device mapping relationship for RDMA containerized access**

**图 2　RDMA 容器化接入的设备映射关系示例**

在此基础上,针对不同大数据框架的作业对环境的具体需求,RDMA 容器化接入机制还提供一组作业适配软件,用于屏蔽环境的特殊性。该软件的具体工作包括以下 3 点:①以最小特权方式配置容器内操作系统的锁页内存,以确保 RDMA 通信可用;②识别当前容器中动态分配的 VF 设备名、端口、GID 等信息,以环境变量方式传递给作业进程,以确保作业可发现高性能网卡;③识别当前容器中多个 TCP/IP 网络平面的 IP 地址,将其中高性能网卡的 IP 地址传递给作业进程,以确保作业选用最佳的通信链路。

#### 4.2.2　RDMA 数据混洗机制

将 MapReduce 计算模型中的混洗过程从计算进程中抽取出来,以独立服务方式部署,是近年来大数据框架为提升数据交换效率、适配存算分离架构而逐步采纳的新架构。Uber[20]、阿里云[21]等厂商推出了面向 Spark 等主流框架的数据混洗服务。现有的数据混洗服务一般只支持 TCP/IP 网络,在具有高性能网卡的环境中无法发挥最佳性能。对接高性能网卡不单只需简单替换通信协议,还必须考虑通信协议对编程范式的影响,否则性能优势将被适配层消耗。在混洗服务场景,主要需要解决传统数据交换的双边通信要求与 RDMA 单边通信特点不匹配的问题。

本文提出一种面向 MapReduce 计算的 RDMA 数据混洗机制[22]。该机制包括独立于大数据框架的混洗服务以及集成到大数据框架的客户端。客户端在 Map 进程中通过 RDMA Put 操作将数据推送至混洗服务,在 Reduce 进程中通过 RDMA Get 操作从混洗服务拉取数据,从而实现 Map 与 Reduce 的解耦。这 2 类操作均为异步执行的单边通信,其涉及的数据分片 ID、通信缓冲区地址和长度等元信息仍通过 TCP/IP 网络以同步双边通信方式在作业进程与混洗服务间交换,从而解决通信的自举问题。

混洗服务和客户端均具有 RDMA 通信内存管理器组件,负责在预置的大块连续锁页内存空间中分配每次通信所需的缓冲区。在混洗服务中,锁页内存空间兼具缓存功能,在缓冲区内并不来自于 Map 进程的数据被持久化之后立即复用,而且按照先入先出的顺序延迟回收。如果在延迟回收之前收到 Reduce 进程的数据拉取请求,则可以直接使用内存数据,省去加载持久化数据的开销。

### 4.3　云原生存算分离架构

针对存算分离架构下的存储管理问题,本文提出一种云原生存算分离架构。该架构涉及云原生 HDFS Operator 和云原生缓存服务。

#### 4.3.1　云原生 HDFS Operator

以云原生容器化方式管理大数据存储组件的难点体现在控制、网络和硬盘 3 个方面。控制方面,需要以统一的入口管理存储服务所有组件的生命周期,使得分布式执行的组件之间具备行为一致性。网络方面,需要考虑同一容器集群中多个存储集群的网络端口互不冲突,从而使得存储集群成为一种可动态创建的云服务。硬盘方面,需要选用支持动态挂载且性能较佳的存储方案,以避免虚拟化和远程通信带来的开销,使得容器化的存储组件达到生产级性能要求。

本文以主流的存储服务——HDFS 为例,给出以容器化方式管理存储组件的具体设计[23]。在控制方面,Kubernetes Operator 机制提供对自定义应用实例的全生命周期管理能力。基于此,提出一种面向云服务场景、支持多 HDFS 集群按需服务的云原生 HDFS Operator,以区别于传统上只管理私有环境、单一 HDFS 实例的 Operator。如图 3 所示,该 Operator 包含一个状态管理器,能够通过状态映射算法,基于底层容器状态推断 HDFS 服务的整体状态。该 Operator 通过 CRD 抽象实现 HDFS 的规格配置与启停控制,并通过 etcd 共享存储实时记录每个 HDFS 集群及其级联资源的元信息,进而维护服务的可用性。在网络方面,出于性能考虑,应选用宿主环境网络而非容器虚拟网络,但这在多 HDFS 集群共存时就会存在端口冲突问题。为此,研发一套基于冷却队列算法的端口分配器,它能够自动为多个 HDFS 集群分配不同的端口集合,并在出现意外冲突时尽可能减少重试开销。为了支持存算分离架构,该机制还能够将端口绑定到 NodePort 或 LoadBalancer,以便外部访问。此外,该机制还支持基于虚拟局域网(Virtual Local Area Network,VLAN)或网络策略的安全隔离。在硬盘方面,综合考虑性能与成本,最佳选项是使用存储集群宿主环境的本地硬盘,这就需要 Operator 具备按需匹配、实时分区和动态挂载这 3 种能力。本文基于 TopoLVM 插

件[24]，设计了支持存储感知调度和延迟绑定技术的动态卷管理器，以提升硬盘空间准备的效率。综合控制、网络与硬盘3方面的技术，云原生HDFS Operator成功地将HDFS适配到按需多租的存算分离架构。

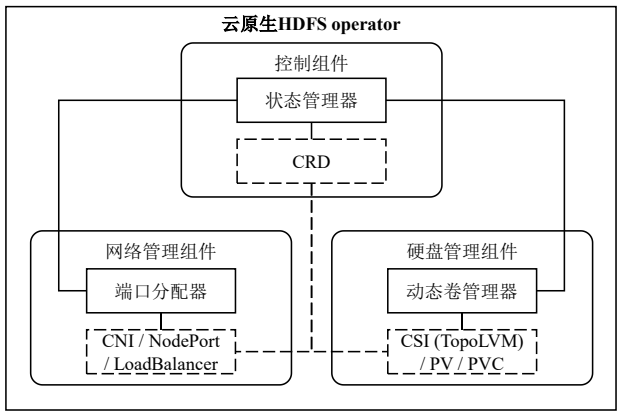


**Fig. 3 Architecture of the cloud-native HDFS Operator**

**图3 云原生HDFS Operator架构**

4.3.2 *云原生缓存服务*

存算分离架构带来的主要副作用是存储访问的局部性缺失，尤其体现在存算集群独立设置时的额外网络传输开销。为缓解这一问题，业界提出了多种缓存层解决方案，其中的代表是Alluxio。缓存服务部署于计算集群，依据特定的策略自动保存频繁使用的数据，从而减少作业对存储集群的访问。在云原生环境下，缓存服务设计也需要适配容器化管理和按需多租等特性。

云原生缓存服务[25]的核心是容器化的Alluxio服务，与原生HDFS服务相同点在于该服务的设计与实现所面临的问题相似，可采用相似的Operator解决方案。不同点在于，该服务位于计算集群，与大数据框架的计算组件及作业负载共存。为提升数据访问效率和安全性，设计了一组亲和性与反亲和性调度策略，使得大数据作业同其配套的缓存服务尽可能运行于相同的宿主节点，且不同租户间的缓存服务尽可能运行于不同的宿主节点。同时，通过配置Kubernetes的网络策略，屏蔽跨租户的缓存访问。

云原生缓存服务针对高性能云原生环境，还具有一套基于三网络平面的优化设计。该设计允许在存算分离架构中分设数据面、控制面与存储面网络，并将大数据作业的缓存访问、数据混洗等通信路由到宿主环境中专用的数据面网络上，以避免容器虚拟化网络开销，同时防止增加容器控制通信的延迟。

## 5 OMBD大数据平台

基于上述关键技术，本文提出了东云睿连大数据（Oriental Mind Big Data，OMBD）平台。OMBD定位于生产级的大数据软件基础设施，是OMStack云原生服务栈[26]的重要组成部分。OMBD以“云原生”和“高性能”作为区别于传统大数据平台的核心特性，解决了相关研究中普遍存在的多种框架和多作业实例共存能力的不足，以及RDMA网络特性的缺失。OMBD能够为多模式作业负载提供统一的管理机制，实现大规模计算和存储调度，提高硬件资源利用率，并提升大数据应用业务部署和执行效率。

### 5.1 系统架构

OMBD的逻辑架构采用层次式设计，如图4所示，自上而下可划分为用户界面层、作业服务层、负载调度层和集群管理层。其中，用户界面层包含图形用户界面和REST API，便于最终用户操作和上层业务集成。作业服务层以容器化方式管理多模式的作业负载与支撑服务。作业负载包括批式、流式与即席查询等多种模式的分布式计算进程，支撑服务则包括混洗服务和缓存服务等。负载调度层是容器集群的前端，提供对计算、网络和存储等多类负载的编排和管理能力。OMBD的各项关键技术主要位于这一层次，它们在Kubernetes原生能力基础上实现了大数据业务专有的丰富语义。集群管理层则是容器集群的后端，负责接入和维护云原生基础设施，包括计算和存储集群、通信网络等。这一层次在Kubernetes原生能力基础上重点加强了对高性能网络的支持。

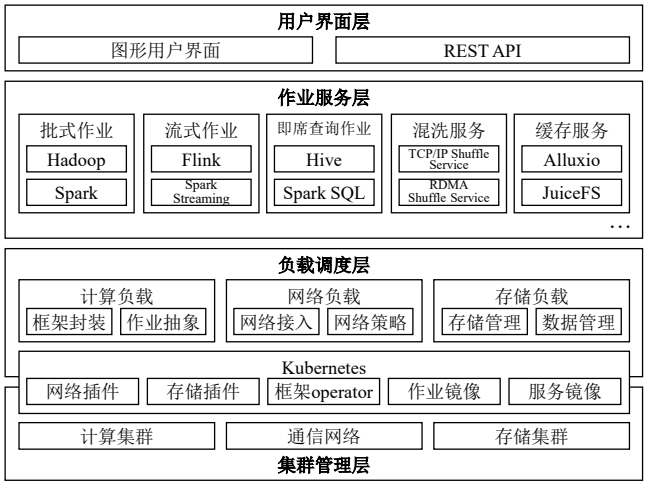


**Fig. 4 Logic architecture of OMBD**

**图4 OMBD逻辑架构**

OMBD的物理实现包括一组分布式运行的中间件、一组Kubernetes插件以及一系列作业和服务镜像。主要中间件Logic Manager和Resouce Manager分别实现负载调度层与集群管理层的功能，主要插件包括高性能网络接入插件、动态存储管理插件和多种框架的Operator等，作业和服务镜像则是大数据框架及其外围封装组件的载体。

### 5.2 功能特性

OMBD能够为最终用户和上层组件提供大数据业务的全生命周期服务，重点功能简述如下：

（1）大数据服务按需部署。以容器化方式动态部署和维护存储服务、混洗服务、缓存服务等大数据支撑服务，以及部分需要以常驻服务方式运行的大数据框架，实现对多类组件的微服务化管理。

（2）多模式作业调度执行。支持批式、流式、即席查询

等多模式大数据作业负载,提供分布式作业大规模并发调度能力;实现系统服务和作业进程的容器化封装,从而支持多版本框架的共存。

(3)作业监控与集群运维。对上层业务负载的全生命周期实施监控,实现作业可恢复性,并记录作业日志和资源用量;对底层基础设施实现动态接入和退出,确保环境稳定性并降低运维成本。

(4)数据存储与资产管理。提供全局对象存储、NFS存储和用户专属的HDFS容器化集群等多种大数据存储服务,并支持维护存储服务之上的数据元信息,帮助用户有效管理数据资产。

(5)多租户角色权限管理。支持"租户——用户"二级权限体系和角色定义,权限涵盖功能与资产维度,既适用于公有云服务,也能够在私有化部署时适配企业现有信息系统的管理模式。

### 5.3　性能测试

为了验证OMBD的性能水平,对其进行全面测试和横向对比,本文主要介绍部分代表性测试结果。测试使用的宿主环境为OpenStack虚拟集群,每个节点配置2块Xeon Gold 6248 CPU、256GB内存、8块2.4TB硬盘与10G RoCE网络。每台虚拟机配置16个vCPU、32GB内存与2TB数据硬盘。软件环境包括CentOS 7.9、Kubernetes 1.19和Hadoop 3.3.0,测试工具为HiBench 8.0。

图5给出了Hive即席查询框架在本地部署和OMBD容器化部署两种情况下的TPC-DS典型查询执行时间对比,主要展示了3条代表性语句在处理不同规模数据时的执行时间。可以看出,OMBD在不同测试中的性能均优于同等配置的本地部署环境,这主要得益于OMBD对调度器职责的分解和对存储访问的优化等设计。

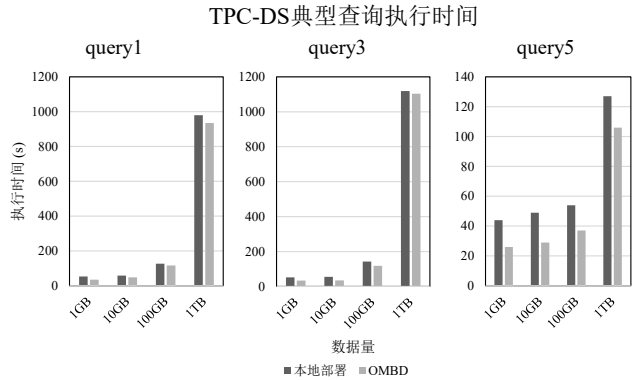


**Fig. 5　Comparison of typical queries' execution time of TPC-DS**

**图5　TPC-DS典型查询执行时间比较**

图6给出了TeraSort在不同存储配置下的性能比较情况。可以看出,OMBD的云原生HDFS Operator在引入容器化服务和基于LVM的动态硬盘分区机制后,存在微小的性能损失。但在实际应用中,由于计算与通信任务的交叠,性能开销可以忽略。相比远程访问全局存储或使用容器文件系统,本方案的性能优势仍然明显。

OMBD的更多性能测试数据参见文献[23]、文献[26]。

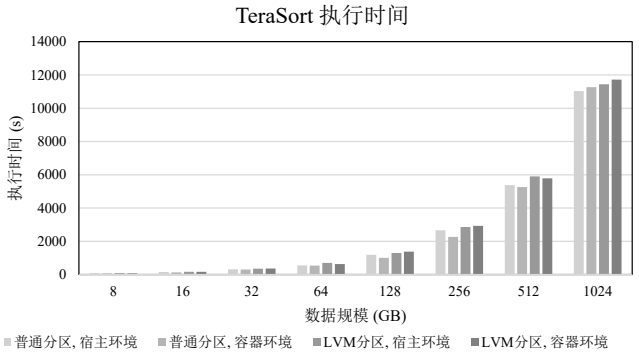


**Fig. 6　Performance comparison of TeraSort in different storage configurations**

**图6　TeraSort在不同存储配置下的性能比较**

### 5.4　应用案例

OMBD已在多个真实业务场景中得到应用。在某车企的数据分析系统中,OMBD用于支撑基于传感器数据的故障智能预测。本平台对大数据框架的按需创建能力很好地适应了有限资源上负载压力波动变化的特点,以较低的建设成本实现了尽可能高的资源利用率。在某政务数据检索平台项目中,OMBD服务于大规模数据的清洗、入库、离线分析和在线检索全流程,其功能完备性、规模可伸缩性和性能优越性得到了充分验证。

## 6　结语

在云原生环境下,传统大数据平台在运行时调度、网络接入和存储管理等方面的缺点逐渐显现。尤其是在引入高性能计算技术后,能够适配技术环境变化的新一代大数据平台呼之欲出。本文在该方向上展开研究,提出了框架调度器封装机制、RDMA容器化接入机制和云原生HDFS Operator等一系列关键技术,并将这些技术集成在一套完整的软件栈——OMBD大数据平台之内。OMBD以CaaS理念和容器集群架构实现了对大数据框架的系统服务和作业进程的有效管理,能够支撑批式、流式、即席查询等多模式负载的高效执行。实验数据和应用案例证明了OMBD设计和实现的有效性,OMBD为高性能云原生大数据平台研究探索了一条可行道路。

**参考文献:**

[1] GONIWADA S R. Cloud native architecture and design: a handbook for modern day architecture and design with enterprise-grade examples[M]. New York: Apress, 2022.

[2] DAGEVILLE B, CRUANES T, ZUKOWSKI M, et al. The snowflake elastic data warehouse[C]//2016 International Conference on Management of Data (SIGMOD), 2016: 215-226.

[3] BALDINI I, CASTRO P, CHANG K, et al. Serverless computing: current trends and open problems[C]//Research Advances in Cloud Computing, Singapore: Springer, 2017: 1-20.

[4] CNCF. Volcano [EB/OL]. https://volcano. sh/.

[5] DOBIES J, WOOD J. Kubernetes Operators: automating the container orchestration platform [M]. Sebastopol: O'Reilly Media, 2020.

[6] ZHU C P, HAN B, ZHAO Y L. A comparative study of spark on the bare metal and kubernetes [C]//6th International Conference on Big Data and Information Analytics (BigDIA), 2020: 117-124.

[7] HUANG K X. Research on containerized big data cloud platform and its storage and scheduling optimization [D]. Nanjing: Nanjing University, 2018.
黄凯旋. 容器化大数据云平台及其存储与调度技术研究[D]. 南京:南京大学,2018.

[8] ZHANG L M. Research and implementation of container scheduling for big data services[D]. Beijing: North China University of Technology, 2020.
张丽敏. 面向大数据服务的容器调度研究与实现[D]. 北京:北方工业大学,2020.

[9] XENONSTACK. How to enable big data applications on kubernetes [EB/OL]. https://www. xenonstack. com/insights/big-data-on-kubernetes.

[10] LIU W G, SUN Y H. Big data 3. 0—the key technologies of big data in post-Hadoop era [J]. Frontiers of Data and Domputing, 2019, 1 (1): 94-104.
刘汪根,孙元浩. 大数据 3. 0——后 Hadoop 时代大数据的核心技术[J]. 数据与计算发展前沿,2019,1(1):94-104.

[11] NETAPP. Modern data analytics: different solutions for different analytics strategies [R]. San Jose: NetApp, Inc. ,58015-sb-4154,2021.

[12] WEI X D, CHEN R, CHEN H B. Optimizing distributed systems with remote direct memory access [J]. Big Data Research, 2018,4(4):3-14.
魏星达,陈榕,陈海波. 基于 RDMA 高速网络的高性能分布式系统[J]. 大数据,2018,4(4):3-14.

[13] APACHE. Docker container executor[EB/OL]. https://hadoop. apache. org/docs/r2. 8. 5/hadoop-yarn/hadoop-yarn-site/DockerContainerExecutor. html.

[14] MELLANOX. SparkRDMA shufflemanager plugin [EB/OL]. https://github. com/Mellanox/SparkRDMA, .

[15] ANANTHANARAYANAN G. Disk-locality in datacenter computing considered irrelevant[C]//13th Workshop on Hot Topics in Operating Systems (HotOS), 2011: 1-5.

[16] XIE D M, HUANG J J, LIAO Z N, et al. Multi-mode big data job scheduling system and method based on container cluster [P]. China, 202210491445. 9, 2022-10-25.
谢冬鸣,黄进军,廖子南,等. 基于容器集群的多模式大数据作业调度系统及方法[P]. 中国,202210491445. 9,2022-10-25.

[17] MELLANOX. K8S RDMA shared dev plugin[EB/OL]. https://github. com/Mellanox/k8s-rdma-shared-dev-plugin.

[18] INTEL. SR-IOV network device plugin for kubernetes[EB/OL]. https://github. com/intel/sriov-network-device-plugin.

[19] HONG Z G, HUANG L, LIN J, et al. Deep learning job operation method and system based on RDMA (Remote Direct Memory Access) equipment[P]. China, 202111478534. 1, 2022-04-19.
洪志刚,黄林,林健,等. 基于RDMA设备的深度学习作业运行方法及系统[P]. 中国,202111478534. 1,2022-04-19.

[20] BANSAL M, YANG B. Zeus: uber's highly scalable and distributed shufe as a service [EB/OL]. https://databricks. com/session_na20/zeus-ubers-highly-scalable-and-distributed-shuffle-as-a-service.

[21] ALIYUN. Open-source big data platform E-MapReduce: RSS[EB/OL]. https://help. aliyun. com/document_detail/446491. html.
阿里云. 开源大数据平台 E-MapReduce: RSS [EB/OL]. https://help. aliyun. com/document_detail/446491. html.

[22] LIN J, HONG Z G. Big data transmission system, method, device and equipment based on RDMA (Remote Direct Memory Access) and storage medium[P]. China, 202210047977. 3, 2022-04-26.
林健,洪志刚. 基于RDMA的大数据传输系统、方法、装置、设备及存储介质[P]. 中国,202210047977. 3,2022-04-26.

[23] LIN J, HUANG L, ZHOU T, et al. An on-demand cloud-native containerized storage design and its practice of HDFS-on-Kubernetes[C]//9th International Conference on Computing and Data Engineering (ICCDE), 2023: 1-9.

[24] CYBOZU. TopoLVM[EB/OL]. https://github. com/topolvm/topolvm.

[25] HUANG L, YU B, XIE D M, et al. Alluxio-based big data job operation system and method[P]. China, 202111092499. X, 2023-05-26.
黄林,余波,谢冬鸣,等. 一种基于Alluxio的大数据作业运行方法和系统[P]. 中国,202111092499. X,2023-05-26.

[26] LIN J, XIE D M, HUANG J J, et al. A multi-dimensional extensible cloud-native service stack for enterprises[J]. Journal of Cloud Computing, 2022, 11(83): 1-18.

(责任编辑:孙 娟 周星宇)