计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191100111

容器技术在科学计算中的应用研究

徐蕴琪[1,2]　黄　荷[1,3]　金　钟[1,3]
1 中国科学院计算机网络信息中心　北京 100190
2 中国科学院大学　北京 100049
3 中国科学院计算科学应用研究中心　北京 100190
(jocelyn_1234@163.com)

摘　要　作为一种新兴的虚拟化技术,容器能够以低廉的资源开销为应用程序和服务提供隔离的运行环境,近年来在持续集成和持续部署、自动化测试、微服务等多种业务场景中获得了广泛应用。在科学计算领域,容器技术的应用也正获得越来越多的关注。借助自身的打包能力及日益壮大的生态系统,容器技术有望为科学计算领域的生产力提升提供助力。文中对容器技术在科学计算中的应用现状进行了调研分析,并根据现有的应用实例讨论了在科学计算中使用容器及相关技术的多种方式。对不同应用模式的分析研究表明,通过提升应用程序的可移植性、改善研究的可重复性、提供非传统应用部署方案、简化云资源调度管理等多种方式,容器及相关技术可以为科学计算领域带来多方面的效率提升。

关键词:容器;科学计算;作业调度系统;容器编排框架;云计算
中图法分类号　TP391

Application Research on Container Technology in Scientific Computing

XU Yun-qi[1,2], HUANG He[1,3] and JIN Zhong[1,3]
1 Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
2 University of Chinese Academy of Sciences, Beijing 100049, China
3 Center of Scientific Computing Applications & Research, Chinese Academy of Sciences, Beijing 100190, China

Abstract　Container is a new virtualization technology that has emerged in recent years. Due to its ability to provide isolated environment for running applications and services with minimal resource overhead, it quickly explodes in popularity among enterprises and has seen wide applications in a number of business scenarios such as continuous integration, continuous deployment, automated testing and micro-services. Although not as fully utilized as in industry, the packaging ability of containers also holds promise for improving productivity and code portability in the domain of scientific computing. In this paper, we discuss how the container and related technologies can be used in scientific computing by surveying existing application examples. The different application patterns represented by these examples suggest that the scientific computing community may benefit from the container technology and the ecosystem evolving around it in many different ways.

Keywords　Container, Scientific computing, Job scheduling system, Container orchestration framework, Cloud computing

1　引言

作为一种轻型的虚拟化技术,容器通过将业务应用、中间件、启动命令等封装为一体,保证了本地环境与部署环境的高度一致,在持续集成与持续部署、微服务等业务场景中有着广泛应用。科学计算则是使用先进的计算能力来理解和解决科学研究和工程技术中的问题,其传统模式需配置和安装专业的计算应用软件,并通过作业调度系统提交计算。

随着云计算和容器技术的发展,容器带来的简化部署、多环境支持和易于迁移等优势逐渐被科学计算领域所关注。在实践中,容器技术和科学计算的结合逐渐形成了若干种应用模式。由于开发目的和使用场景的不同,这些模式在技术路线和实现方式上存在一定的差异。本文对容器技术在科学计算中的应用现状进行了调研分析,试图对其中极具代表性的模式进行归纳对比,并通过具体案例进行阐述,进而对容器技术在科学计算中的应用方式进行总结和展望,以期为容器技

到稿日期:2019-11-14　返修日期:2020-03-24　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家重点研发计划项目(2016YFB0201700);广东省生物医药计算重点实验室(2016B030301007);中国科学院科研信息化应用工程项目(XXH13506-202)
This work was supported by the National Key Research and Development Project(2016YFB0201700), Guangdong Provincial Key Laboratory of Biocomputing(2016B030301007) and Information Program of the Chinese Academy of Sciences(XXH13506-202).
通信作者:黄荷(huanghe@sccas.cn)

术在科学计算领域的进一步研究和应用提供参考。

2 容器技术与科学计算

2.1 容器技术

容器技术为程序提供了一种便利的打包机制，这种机制将应用及其依赖项打包和隔离为单个对象，具有快速、高效、易迁移等优点。与虚拟机不同，容器环境无需安装操作系统，可以直接运行于宿主机操作系统上(见图 1)，通过共享底层资源来节省开销，容器对系统资源的额外需求远低于虚拟机，从而能够快速、可靠、一致地部署应用[1]。2013 年初，基于 Go 语言实现的开源容器引擎 Docker 诞生。Docker 对容器 Namespace，Cgroups 等底层技术进行了封装抽象，提供了高效、敏捷和轻量级的容器方案，很快得到了微软、红帽、IBM 等厂商的支持，成为了目前主流的容器技术[2]。

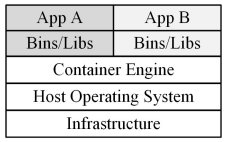

图 1 容器技术示意图

Fig. 1 Diagram of container technology

容器的本质是一组受到资源限制、彼此间相互隔离的进程[3]，其实现依赖于 Linux 内核的 Namespace，Cgroups 和 Chroot 等技术。容器进程在启动时通过指定 Namespace 参数，使该进程属于一个独立的空间。在此空间中，进程既看不到其在宿主机的真正进程，也看不到宿主机的其他进程，实现了容器进程间的隔离。Cgroups 将一系列进程集中在一起，为其设置能够使用的资源上限，容器启动时即通过设置指定的 Cgroups 参数来限制其对资源的访问[4]；Chroot 能将进程的根目录切换为指定的目录，容器进程即使用 Chroot 将其根目录指定为一个隔离的子目录，使得在容器中无法访问到宿主机操作系统的文件目录，从而实现容器文件系统的隔离。

随着容器技术的研发和应用的推广，容器生态圈得到了迅速扩充。当面对单主机多容器场景时，简单的容器工具如 Docker Compose 便可以满足部署和管理的需求。当需要高效地抽象和管理大规模容器集群时，便涉及到多容器工作负载的生命周期管理等问题，此时则需利用生态圈内的容器编排框架，将容器按照规则运行，并且完全自动化地处理好容器之间的关系，以提供应用部署、维护、扩展等功能[5]。目前生态圈内主流的容器编排框架有 Kubernetes[6]，Mesos[7] 和 Docker Swarm[8] 等。近年来，容器及周边生态圈的发展引起了业内的广泛关注，并被应用于持续集成与持续部署、微服务架构等多个场景[9]。在持续集成与持续部署场景中，利用容器平台便可配置实现应用的编译、构建、测试、打包、发布等自动化流程，使整体应用交付变得更为快速，而且保证了线上和线下环境完全一致，最大化地降低了运维成本；在微服务场景中，传统的业务系统被拆分解耦成细粒度的服务模块，每个服务模块运行在独立的容器中，服务模块之间互相协调配合，独立部署升级，具有高效、轻量、易迁移等特性。继虚拟化技术后，容器技术对云计算的发展产生了深远的影响。

2.2 科学计算

作为科学研究和技术创新的重要方法，科学计算已经在分子模拟、蛋白质与基因组分析、药物设计、气象分析等领域发挥了重要的作用[10]。相比其他类型的计算，科学计算往往具备如下特点。

(1)计算密集型：科学计算的工作负载通常是科学工程计算、数值模拟等，需要进行大量计算，通常需要高计算能力、高网络性能、快速存储等条件。

(2)一次性任务：与长时运行服务不同，科学计算的工作负载一般是一次性任务，最终得到一个完成态结果。

(3)计算时间长、要求高：通常科学计算任务的持续时间长，且对算法的执行效率和计算精度要求较高。当科学计算任务的规模较大时，对内存和计算性能等都有较高的要求。

科学计算的上述特点决定了其运行的一般模式。首先，科学计算任务主要运行于高性能工作站或具备一定规模的计算集群；其次，科学计算任务在硬件资源上的调度与执行通常由作业调度系统完成。调度系统针对多用户场景，允许用户向已预先配置硬件资源及优先级等运行条件的作业队列提交特定资源需求(节点、内存、GPU 等)的作业任务，并依据一定的调度算法进行作业资源分配，保证用户占用资源的公平性[11]。常用的作业调度系统有 LSF[12]，Slurm[13] 和 PBS[14] 等。但在该模式下，科学计算面临着一些问题和挑战。首先是计算软件与其所依赖的集群系统软件间的兼容性问题。由于计算集群一般为多用户环境并同时安装有较多的计算软件，在此情况下调整系统软件版本带来的风险相对较高。而且在传统的工作负载外，数据密集型应用和人工智能方面的应用大量出现，这些新应用通常需要大量的软件依赖和复杂的环境配置，这给系统的部署和维护带来了一定的挑战。其次，研究工作的可靠性依赖于数据结果的可重复性，这就需要整个系统堆栈的完全可再现性机制。若仅仅依靠源代码的一致，并不能保证重现得到特定计算结果的环境，进而难以保障科学计算的可靠性。另外，研究中有时会涉及应用服务的开发，这就需要由基础设施提供开放的平台供长时服务进程或应用运行，但现有环境(如超算集群)对任务运行时间有着严格的限制，导致满足该需求存在一定的困难。

3 容器技术在科学计算中的应用

容器技术通过内核轻量级的虚拟化技术对软件和其依赖环境进行标准化打包，逐渐成为可以解决以上问题的新兴技术[15]。相比传统的工作负载，将科学计算任务容器化有一些显而易见的优势[16]。

(1)灵活：容器打包和隔离的能力可以为应用及其依赖环境构建一个独立的运行环境，从而实现不同应用之间的兼容，使得进行不同工作负载的用户均可完成计算任务。

(2)一致：用户的工作环境与科学计算系统或多个科学计算系统之间的运行环境可能完全不同，维持多个环境间的可移植性往往会存在一定的问题，而借助容器技术便可以避免这些问题的发生。

(3)可重复：容器作为安装和运行应用程序的一种方式，如果镜像描述的依赖关系足够精确，其他人就可以重新生成

完全一致的环境以重现计算结果。

通过对近年来相关文献、会议及网络资料的充分调研发现,容器技术在科学计算领域的应用逐步形成了若干种模式,并带来了不仅限于上述列举的优势。由于开发目的和使用场景不同,其具体应用方式有着一定差异,下文将通过实例对不同的应用模式进行介绍。

3.1 通过作业调度系统运行容器化科学计算作业

目前科学计算领域最主流的容器技术应用模式是将计算任务本身通过容器打包后整合进现有的作业调度系统并运行。该模式利用的是容器诸多能力中最基础的对应用程序及运行环境的打包能力,作业任务仍然通过原有作业调度系统提交运行,仅在具体任务的执行中由运行某计算程序转变为调用该计算程序对应的容器(见图 2)。对传统超算中心而言,该模式可无缝兼容非容器化的传统作业类型,从而将新技术的引入对现有业务的影响降至最低,并使传统作业调度方案在多用户批量作业场景下经过长期发展演化所积累的优势得以保留。

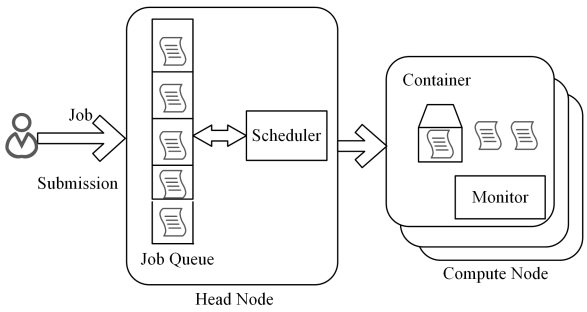

图 2 通过作业调度系统运行容器化作业示意图

Fig. 2 Diagram of running containerized workloads under existing job scheduler

尽管模式相对简单,但若将主流容器技术 Docker 直接应用于该模式时仍存在一些问题。一方面,多用户的资源共享场景带来了对权限限制和安全的严格要求,而该方面的要求难以为 Docker 所满足[17]。另一方面,传统高性能类型的科学计算应用通常需要借助以 MPI 为代表的并行编程技术,利用多处理器及多节点对计算任务进行并发式处理,而 Docker 对 MPI 任务并无原生支持[18]。针对这些问题,一系列面向科学计算的容器技术得以开发,这些 Docker 的替代方案普遍增强了权限控制,增加了与作业调度系统及 MPI 兼容的能力,使用户可以轻松地实现容器与超算环境的集成,其中 Shifter,Singularity 和 Charliecloud 是突出的代表技术[19]。

(1)Shifter[20]

Shifter 起源于美国国家能源研究科学计算中心(The National Energy Research Scientific Computing Center)和高性能计算厂商克雷(Cray)公司的项目,用于支持打包为 Docker 容器的计算应用在作业调度系统中的部署。其研究重点是使用容器作为打包应用程序运行环境的方式,将 Docker 容器带入科学计算系统中。由于 Docker 与现有作业调度系统存在兼容问题,而且 Docker 以 root 身份运行的守护程序和容器内用户权限提升等问题也给系统引入了不安全的因素,因此 Shifter 复用了与 Docker 相同的 API,同时将一些关键问题涉及的特性移除或进行二次开发,从而达到目的。例如,在隔离性方面,Shifter 只使用了挂载点命名空间,实现了基本的文件系统隔离。Shifter 还限制了容器内的权限,运行容器的用户即容器内操作的用户,不存在权限提升问题[21]。Shifter 也支持 MPI 应用程序,且以 MPICH 为主[22]。

使用 Shifter 的工作流程如图 3 所示。首先用户需要为应用创建镜像并将其提交到远程仓库;然后在已安装 Shifter 的系统中将镜像从远程仓库拉取到系统中,Shifter 的镜像网关(Image Gateway)会将镜像自动转换为 Shifter 格式;最后用户编写作业调度系统的脚本或者直接用调度系统的命令将镜像以容器形式运行[23]。目前,Shifter 已经吸引了来自高能物理、核物理和基础能源科学等研究领域的用户,未来还将引入带有访问限制的私人 Shifter 镜像等功能。

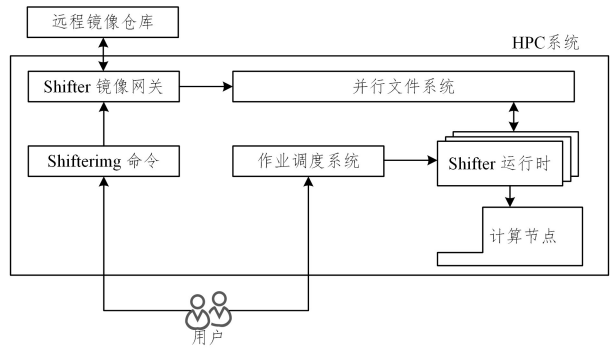

图 3 Shifter 的使用流程示意图

Fig. 3 Diagram of shifter architecture and user workflows

(2)Singularity[24]

Singularity 始于 2015 年美国劳伦斯伯克利国家实验室(Lawrence Berkeley National Laboratory)的开源项目,定位与 Shifter 相似,其目的是解决容器在科学计算领域中特有的挑战。Singularity 的一个突出特点是它的镜像格式是包含所有依赖项的单个镜像文档,而并非包含多个只读层且每一层均为 tar 格式文档的 Docker 镜像,因此 Singularity 的镜像更便于存档和分发,也为应用跨计算环境和软件版本的转移保障了可重复性。在安全性方面,Singularity 容器启动时用户上下文始终保持不变,权限在容器内部和外部是相同的,并且没有单独的守护进程,避免了相应的问题[25]。Singularity 工作负载可以通过脚本像常规 Linux 命令一样启动,因此可以直接与作业调度系统集成,无需为科学计算环境做额外的适配。另外,Singularity 在运行 MPI 应用程序方面也具有优势,带有与主机相同 MPI 版本的 Singularity 容器可直接通过 mpirun 命令启动[26]。其使用流程与 Shifter 类似,此处不再赘述。

除此之外,CharlieCloud 是美国洛斯阿拉莫斯国家实验室(Los Alamos National Laboratory)为超算集群开发的开源容器技术。该技术提供了一个可重现和无特权的容器工作流程,仅利用 800 行代码便可帮助科学计算用户在超算环境中运行工作负载,避免了多数安全风险,实现了用户自定义软件堆栈与主机操作系统的有效隔离[27]。

使用容器作为计算作业新的承载形式,在得到打包隔离的灵活性与可再现性的同时,用户普遍担心其可能会带来性

能方面的牺牲。戴尔 EMC HPC&AI 创新实验室的研究团队进行的测试表明，与裸机相比，在 Singularity 上的深度学习应用没有性能损失，且在其基准测试中容器运行与裸机运行的相对性能差异小于 2%[28]。另外，在阿里云服务器上进行的 HPL(High-Performance Linpack)性能测试表明[29]，Shifter 和 Singularity 容器的性能略优于 Docker 容器，且 Singularity 与宿主机的实测 HPL 性能相当。而有研究人员专门针对 Docker 和 Singularity 容器所做的测试也证实了其各方面的性能均达到近乎裸机的水平[30]。截至 2019 年初，Singularity 已经被安装超过 40 000 次，其中包括顶级 HPC 中心，如美国得克萨斯高级计算中心、圣地亚哥超级计算机中心和橡树岭国家实验室等[31]。

3.2 通过容器编排框架提供辅助性科学计算服务

随着技术研究的发展，科学计算的需求不再是单纯的一次性计算任务，围绕科学计算提供如数据库、数据门户、Web 应用和工作流系统等持久化服务的需求越来越多，这意味着环境需要提供运行持久服务或守护进程的能力。在科学计算传统模式中，针对特定的部署服务需求，用户往往需要与资源管理人员单独沟通以获取额外资源、开放额外权限并获得技术支持。对资源管理人员而言，该过程的工作量较大，难以扩展且缺乏统一的管理方案。而容器及容器编排框架的原生应用场景正是持久化服务，此类技术为满足该场景下的需求提供了一条可行的技术路线。资源管理人员可通过容器编排框架搭建与计算服务连通的容器服务平台，供用户通过运行容器的方式自助运行辅助科学计算的持久性服务。橡树岭领导计算设施(The Oak Ridge Leadership Computing Facility, OLCF)提供了该应用模式的一个典型案例[32]。

橡树岭领导计算设施使用容器编排框架 Kubernetes 构建了一个运行用户自定义服务的平台环境。Kubernetes 是一个管理和部署容器化应用的开源解决方案，为搭建容器集群和进行容器编排提供了标准的部署接口和模块化的 API[33]，涵盖了自动化容器部署、动态扩缩容、服务滚动升级、容器组间负载均衡等功能[34]。面对 Kubernetes 企业级支持的需求，红帽公司开发的 OpenShift 平台为企业提供了生产环境下使用 Kubernetes 的技术支持，强化了 Kubernetes 的功能，集成了包括网络、监控、安全等更适用于企业生产场景的额外组件[35]。OLCF 正是利用了 OpenShift，在超算集群外搭建了一个容器应用集群(见图 4)，为用户自定义服务提供了统一的运行环境。在容器中，应用能够与超算集群的作业调度系统交互，提交计算任务以及获取队列信息；同时，容器可以访问超算集群的共享文件系统，为计算数据的处理提供了便利[36]。

该应用模式从超算用户尤其是超算应用服务开发用户的角度出发，解除了一般在超算集群环境中难以运行对外服务或守护程序、难以在计算节点之外共享文件系统的限制。模式以容器的形式部署运行持久化对外服务，由容器编排框架帮助处理管理服务的细节，具备运行监控、负载均衡、故障恢复等特性，降低了由用户部署和运维管理应用服务的复杂性[37]，有力支持了应用快速迭代的需求，为支持辅助性科学计算服务提供了一套统一和高效的平台方案。

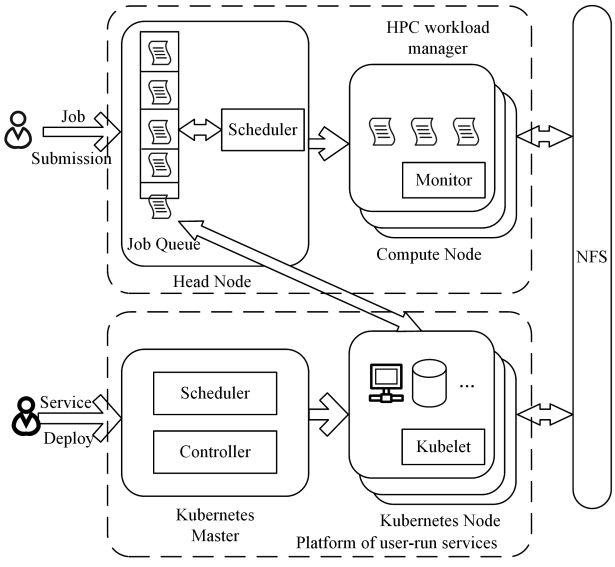

图 4 OLCF 应用模式示意图

Fig. 4 Diagram of application pattern used in OLCF

3.3 通过容器编排框架构建云上的科学计算平台

各科研单位和企业在使用自建或超算中心的高性能计算集群时，往往会遇到难以在短时间内获取足够计算资源的问题，而且集群的硬件设备与新技术整合的周期较长，无法满足各类型科研任务的计算需求[38]。而随着云时代的兴起，公有云的硬件更新速度相对传统计算集群更快，细粒度的资源划分和按需计费的模式使其在多方面都具有一定的优势。根据全球知名高性能计算咨询与服务公司 Intersect360 Research 的调查显示[39]，2016—2017 年，公有云客户在高性能云上的支出增长了 44%，远高于高性能市场的整体增长率 1.6%，有 64%的超算中心在公有云上运行了部分工作负载。云化计算资源逐渐成为了超算资源的重要补充。

当前，科学计算使用云资源的方式一般是租用云服务器(即虚拟机)，该形式与传统计算集群的使用方式比较相似，此处不再赘述。随着虚拟化和容器技术的发展，计算的抽象化和服务的细粒度化成为了云服务的突出趋势。云服务提供商在提供基础架构技术之外，也提供了如容器、自动化运维、中间件等服务，保障了应用的稳定性和扩展性，使开发者可专注于业务逻辑的研发，进一步降低了开发成本。在此背景下，研究人员可利用云服务提供商的计算资源，使用容器编排框架替代传统作业调度系统进行任务调度，从而高效地完成计算任务。

作为最广泛使用的编排框架，Kubernetes 已经逐渐成为云原生应用部署管理方式的事实标准。针对科学计算任务的特点，虽然 Kubernetes 的设计之初是以服务为核心，但在 1.2 版本之后，其开始支持 Job 类型即批量处理的一次性任务，其中包括固定结束次数的 Job、并行 Job 等多种形式[40]。而且 Kubernetes 拥有插件式的调度器，支持用户根据自身需求定制调度策略，这大大增强了其面对不同任务调度需求的扩展性。如图 5 所示，基于云平台提供的可灵活扩展的海量资源，借助 Kubernetes 完善的集群管理能力并搭配定制化的科学计算调度插件，可以作为科学计算下资源调度的新方案，在云上构建容器化的科学计算任务运行平台。

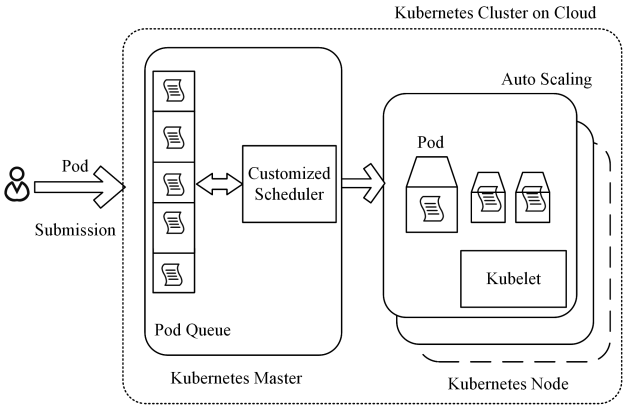

图 5 利用容器编排框架(如 Kubernetes)实现云上科学计算示意图

Fig. 5 Diagram of scientific computing on cloud using container orchestration framework, e. g. Kubernetes

国内某科技公司的“药物固相筛选与设计平台”[41]即是此应用模式的一个例子。该平台利用 Docker 容器封装了与业务相关的算法,利用 AWS 和腾讯云的云服务构建了计算集群,使用 Kubernetes 配合定制化的科学计算调度策略及工具库管理容器化工作负载,另外通过编排竞价实例和其他不同类型的实例降低了计算成本。基于 Kubernetes 的科学计算平台在云上通过容器化完成了高效的装箱,实现了对大规模容器实例的管理[42],并且有效降低了计算任务大规模使用高性能硬件带来的成本压力,为该公司在计算费用上的成本节省了 50%~60%[43]。

总体而言,云端使用容器化相关策略进行科学计算的实践,能够使科研企业和机构方便、快速地管理和控制计算集群,加快研发速度,为计算带来可用性和可扩展性的潜力[44],进而取得海量资源、灵活扩展和节省成本的多方面效益。

3.4 通过无服务器技术运行容器化科学计算应用

在 KubeCon North America 2018 会议上,英国布里斯托大学的研究软件工程组(Research Software Engineering Group)介绍了他们利用容器技术进行科学计算的研究动机与应用模式[45]。在过去的数十年中,尽管生命科学领域内的研究理论有了很大的突破,但研究方式几乎没有变化:研究人员用终端登录到高性能计算集群的节点,利用描述文件写好作业描述,提交作业到计算队列中,最后把所有的结果文件下载到本地进行分析。在技术快速迭代的当下,研究人员认为仍使用一成不变的终端命令行方式进行计算非常不便,因此有了新方式的构建。

整个系统的框架如图 6 所示。该研究组选择了 Jupyter Notebook 作为前端浏览器服务。它是一个将文本、代码和可视化组合到一个文档中的 Web 应用。Jupyter Hub 在 Jupyter Notebook 基础上实现了多用户的管理,可以生成、管理用户 Jupyter Notebook 服务的多个实例[46]。在部署 Jupyter Hub 方式的选择上,考虑到 Kubernetes 在应用服务领域有着强大的优势,研究人员在一个轻量级的 Kubernetes 集群上部署了 Jupyter Hub 服务,为科学计算提供了一个有良好交互性与易用性的前端环境。

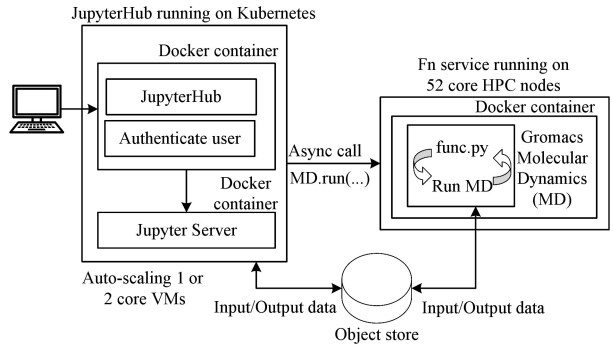

图 6 系统架构示意图

Fig. 6 Diagram of system architecture

在 Jupyter Notebook 中,所有的计算和数据都在后端,只有当需要展示在浏览器中时才会传输到前端。由于运行前端服务的 Kubernetes 集群较小,无法处理大规模模拟计算,因此计算并不是在其中运行。研究人员使用了无服务器平台 Fn Project,从 Kubernetes 中的 Jupyter Notebook 上所触发的计算会通过 Fn 向一个 52 核的公有云裸金属节点调度计算负载。Fn 是甲骨文公司推出的由事件驱动的无服务器平台,无服务器意味着开发者不需要关心与服务器相关的事情,应用逻辑采用函数即服务(Function as a Service,FaaS)架构通过功能组合实现[47]。在该模式下,进行分子模拟计算的过程就是一个无服务器函数,运行函数的代码被包装在容器中。不同的分子动力学的 Fn 函数可以调用不同规模的硬件资源,因此计算允许按需分配高内存、多 CPU 或 GPU 节点响应函数调用。另外,他们还配置了共享对象存储(Shared Object Store),当函数被调用时,只需要将输入数据上传至对象存储中,数据就能够被复制进容器以运行计算流程,然后通过 watchdog 进程将输出数据以流的形式写回对象存储,从而保证了输入和输出数据的持久性。

在该案例中,易用性良好的前端和云端按需供给资源的后端结合无服务器计算的模式,为部署和管理应用提供了一种更高级别的抽象,使开发者仅需专注于为应用编写函数代码及触发器,帮助运维者从基础设施配置和管理工作中解放出来[48],减少在非业务核心任务上花费的时间成本,也令使用者运行科学计算任务变得简单和可交互。该模式有效降低了科学计算的使用门槛,为服务型科学计算应用提供了良好的范例。

结束语 随着软件环境和应用程序的复杂性不断提高,容器在科学计算领域中的使用越来越普遍。纵观国内外应用现状的种种模式可以发现,除了利用打包机制提升应用程序的可移植性与改善研究的可重复性之外,容器还结合容器编排等技术提供了科学计算作业调度系统的替代调度方案,同时提供了长时服务高效的部署管理方式,并通过无缝对接云端弹性基础设施为计算与服务带来了更高的扩展性。总体而言,不同应用模式的侧重点可归纳为两方面:1)利用容器的打包和隔离能力,解决计算中环境兼容与工作流程迁移等问题;2)利用容器编排框架的调度管理能力,提供传统作业调度系统的替代方案,自动化基础设施的配置与管理。本文的调研分析表明,容器及相关技术可以为科学计算带来工作流程执

行与迁移、应用部署与管理、资源控制与扩展等多方面的效率提升。

随着云时代的兴起，云端成为科学计算极具吸引力的执行环境。未来，针对如何将工作负载高效迁移上云[49]以及利用容器构建多云架构避免单一供应商锁定[50]等将成为进一步发展的趋势。另外，容器及容器编排框架技术仍存在一些问题，例如如何保障多租户间资源共享与隔离的安全性[51]、如何优化资源分配以提高集群整体利用率[52]等。随着领域内的研究与实践，这些问题有望得到进一步的突破并使容器技术更广泛地应用在科学计算中。容器已经彻底改变了许多行业开发和部署服务的方式，未来有望在科学计算领域以多样化的方式得到更多的应用。

参 考 文 献

[1] Cloud Developer. The Evolution of Container Technology[EB/OL]. (2018-04-17) [2019-08-02]. https://www.cnblogs.com/bakari/p/8868850.html.

[2] YANG B H, DAI W J, CAO Y L. Docker Primer[M]. Beijing: China Machine Press, 2015: 3-6.

[3] MENAGE P B. Adding Generic Process Containers to the Linux Kernel[C]// Proceedings of the Linux Symposium. 2007, 2: 45-57.

[4] WU Z X. Advances on Virtualization Technology of Cloud Computing[J]. Journal of Computer Applications, 2017, 37(4): 915-923.

[5] WU S, WANG K, JIN H. Research Situation and Prospects of Operating System Virtualization[J]. Journal of Computer Research and Development, 2019, 56(1): 58-68.

[6] BERNSTEIN D. Containers and Cloud: From LXC to Docker to Kubernetes[J]. IEEE Cloud Computing, 2014, 1(3): 81-84.

[7] HINDMAN B, KONWINSKI A, ZAHARIA M, et al. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center [C]// NSDI. 2011, 11(2011): 22.

[8] Docker Inc. Docker Swarm Overview [EB/OL]. (2018-02-24) [2019-08-06]. https://docs.docker.com/swarm/overview/.

[9] LOU C. An Analysis of the Application Scenari-os of Docker Container[EB/OL]. (2018-09-11) [2019-08-06]. https://sq.163yun.com/blog/article/197408114326077440.

[10] ZHU S P. A Brief Report on Scientific Computing[J]. Physics, 2009(8): 545-551.

[11] ZHANG X L, ZHONG Y P. A Performance Evaluation and Compare for Cluster-based Resource Management Systems[J]. Application Research of Computers, 2003(9): 56-59.

[12] ZHOU S, ZHENG X, WANG J, et al. Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems[J]. Software: Practice and Experience, 1993, 23(12): 1305-1336.

[13] YOO A B, JETTE M A, GRONDONA M. Slurm: Simple Linux Utility for Resource Management[C]// Workshop on Job Scheduling Strategies for Parallel Processing. Springer, Berlin, Heidelberg, 2003: 44-60.

[14] NITZBERG B, SCHOPF J M, JONES J P. PBS Pro: Grid Computing and Scheduling Attributes[M]// Grid Resource Management. Springer, Boston, MA, 2004: 183-190.

[15] LEHTO O P. Containers, Meet HPC [EB/OL]. (2015-12-30) [2019-08-12]. https://medium.com/@ople/containers-meet-hpc-2aab7aa2d54a.

[16] TELFER S. The State of HPC Containers[EB/OL]. (2018-08-16) [2019-08-12]. https://www.stackhpc.com/the-state-of-hpc-containers.html.

[17] WANG J, HU W, ZHANG Y H, et al. Trusted Container Based on Docker[J]. Journal of Wuhan University (Natural Science Edition), 2017, 63(2): 102-108.

[18] MEDRANO-JAIMES F, LOZANO-RIZK J E, CASTAÑEDA-AVILA S, et al. Use of Containers for High-Performance Computing[C]// International Conference on Supercomputing in Mexico. Springer, Cham, 2018: 24-32.

[19] YOUNGE A J, PEDRETTI K, GRANT R E, et al. A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers And Clouds[C]// 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, 2017: 74-81.

[20] GERHARDT L, BHIMJI W, FASEL M, et al. Shifter: Containers for HPC[C]// J. Phys. Conf. Ser. 2017, 898: 082021.

[21] JACOBSEN D M, CANON R S. Contain This, Unleashing Docker for HPC[C]// Proceedings of the Cray User Group, 2015: 33-49.

[22] NERSC. How to Use Shifter[EB/OL]. (2019-03-22) [2019-08-18]. https://docs.nersc.gov/programming/shifter/how-to-use.

[23] PAN X. Shifter: Containers for HPC [EB/OL]. (2017-08-23) [2019-08-18]. https://www.ibm.com/developerworks/cn/linux/l-panxun-shifter-compute-container/index.html.

[24] KURTZER G M, SOCHAT V, BAUER M W. Singularity: Scientific Containers for Mobility of Compute[J]. PLoS One, 2017, 12(5): e0177459.

[25] IDG Communications. Utilizing Containers for HPC and Deep Learning Workloads [EB/OL]. (2018-05-17) [2019-08-19]. https://www.cio.com/article/3269351/utilizing-containers-for-hpc-and-deep-learning-workloads.html.

[26] Singularity. Singularity Docs-Admin Guide[EB/OL]. (2018-06-21) [2019-08-19]. https://www.sylabs.io/guides/2.5/admin-guide/installation_environments.html.

[27] PRIEDHORSKY R, RANDLES T. Charliecloud: Unprivileged Containers for User-defined Software Stacks in HPC[C]// Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2017: 36.

[28] DANDAPANTHULA N. Singularity Containers for HPC & Deep Learning[EB/OL]. (2018-03-19) [2019-08-20]. https://www.nextplatform.com/2018/03/19/singularity-containers-for-hpc-deep-learning.

[29] BAO X. Alibaba Cloud Supercomputing: Singularity for High-Performance Container Solutions [EB/OL]. (2018-11-16) [2019-12-20]. https://yq.aliyun.com/articles/669961.

[30] SAHA P, BELTRE A, UMINSKI P, et al. Evaluation of Docker Containers for Scientific Workloads in the Cloud[C]// Proceedings of the Practice and Experience on Advanced Research Com-

puting. ACM,2018:11.
[31] BAUER M. The Singularity runtime [EB/OL]. (2019-06-20) [2019-12-20]. http://qnib.org/data/hpcw19/1_RUN_3_singularity.pdf.
[32] KINCL J,ADAMSON R. Allowing Users to Run Services at the OLCF with Kubernetes[EB/OL]. (2018-05-16) [2019-08-21]. https://www.fbcinc.com/e/nlit/presentations/Kincl_Adamson-Allowing_Users_to_Run_Services_at_th.pdf.
[33] Kubernetes. Kubernetes Documentation[EB/OL]. (2019-02-22) [2019-08-21]. https://kubernetes.io/docs/home.
[34] GAO Q. Random Talk on Container,Container Cluster Management Platform and Kubernetes[EB/OL]. (2018-11-12) [2019-08-21]. https://www.kubernetes.org.cn/4786.html.
[35] CHEN G. OpenShift:Enterprise Container Platform Based on Kubernetes[M]. Beijing:China Machine Press,2017:3-9.
[36] HINES J. OLCF Testing New Platform for Scientific Workflows [EB/OL]. (2017-06-06) [2019-08-21]. https://www.olcf.ornl.gov/2017/06/05/olcf-testing-new-platform-for-scientific-workflows/.
[37] SHAH J,DUBARIA D. Building Modern Clouds:Using Docker,Kubernetes & Google Cloud Platform[C]// 2019 IEEE 9th Annual Computing and Communication Workshop and Conference(CCWC). IEEE,2019:0184-0189.
[38] Reeko. Solutions of Scientific Computing in the Cloud[EB/OL]. (2017-12-22) [2019-08-27]. http://www.reeko.net.cn/page108?article_id=245.
[39] FELDMAN M. Cloud Computing in HPC Surges [EB/OL]. (2018-06-29) [2019-08-27]. https://www.top500.org/news/cloud-computing-in-hpc-surges/.
[40] Kubernetes. Jobs-Run to Completion [EB/OL]. (2019-12-02) [2019-12-20]. https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/.
[41] Xtalpi Inc. Products and Services [EB/OL]. (2018-10-25) [2019-08-29]. http://www.jingtaikeji.com/products/.
[42] LIN S K. Building Container-based Scientific Computing Platform on Cloud[EB/OL]. (2018-07-16) [2019-08-30]. https://cloud.tencent.com/developer/article/1157276.
[43] Amazon Web Services. XtalPi Case Study[EB/OL]. (2019-05-18) [2019-08-29]. https://aws.amazon.com/solutions/case-studies/xtalpi/?nc1=h_ls.
[44] THURGOOD B,LENNON R G. Cloud Computing With Kubernetes Cluster Elastic Scaling[C]// Procee-dings of the 3rd International Conference on Future Networks and Distributed Systems. ACM,2019:5.
[45] WOODS C. Running Serverless HPC Workloads on Top of Kubernetes and Jupyter Notebooks[EB/OL]. (2018-12-13) [2019-09-13]. https://schd.ws/hosted_files/kccna18/77/woods_serverless_hpc.pdf.
[46] HE Z P,ZHANG X D,LIU Y. Microservice Architecture for Jupyter-Based Interactive Analysis Plat-form[J]. Computer Systems and Applications,2019,28(8):63-70.
[47] KAVIANI N,KALININ D,MAXIMILIEN M. Towards Serverless as Commodity:A Case of Knative[C]// Proceedings of the 5th International Workshop on Serverless Computing. ACM,2019:13-18.
[48] GROβMANN M,IOANNIDIS C,LE D T. Applicability of Serverless Computing in Fog Computing Environments for IoT Scenarios[C]// Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion. ACM,2019:29-34.
[49] KEHRER S,BLOCHINGER W. A Survey on Cloud Migration Strategies for High Performance Computing[C]// Proceedings of the 13th Advanced Summer School on Service-Oriented Computing. IBM Research Division,2019:57-69.
[50] HONG J,DREIBHOLZ T,SCHENKEL J A,et al. An Overview of Multi-cloud Computing[C]// Workshops of the International Conference on Advanced Information Networking and Applications. Springer,Cham,2019:1055-1068.
[51] NOSSIK M,DU L,LINCOURT JR R A,et al. Security Layer for Containers in Multi-tenant Environments:U. S. Patent 10,326,744[P]. 2019-6-18.
[52] RATTIHALLI G,GOVINDARAJU M,LU H,et al. Exploring Potential for Non-Disruptive Vertical Auto Scaling and Resource Estimation in Kubernetes[C]// 2019 IEEE 12th International Conference on Cloud Computing(CLOUD). IEEE,2019:33-40.

XU Yun-qi, born in 1994, postgraduate. Her main research interests include scientific computing and application service development.

HUANG He, born in 1982, Ph.D, associate professor. Her main research interests include scientific computing and data service.