分 类 号： TP311　　密级： 公开
论文编号：2015021567

贵　州　大　学

2018 届硕士研究生学位论文

一种面向云环境的 Ceph 集群能耗管理策略研究

学科专业：计算机科学与技术
研究方向： 计算机应用技术
导　　师： 吕晓丹 副教授
研 究 生： 彭丽苹

中国 ▪ 贵州 ▪ 贵阳
2018 年 6 月

目 录

一种面向云环境的 Ceph 集群能耗管理策略研究

摘要

近年来，云计算与大数据技术得到了飞速发展。从智慧医疗到智慧城市、从精准扶贫到精密工业、从政府办公到个性化的推荐系统，都展示着云计算与大数据给人们生活带来的变化。但与此同时，数据中心巨大的能耗问题引起了人们的广泛关注，如何降低数据中心能耗成了工业界和学术界研究的热点问题。

本文首先介绍了云平台能耗管理的研究背景、意义、国内外现状及相关基础知识；其次，针对 CRUSH 算法在节能方面的不足，建立了一个 Ceph 集群能耗优化模型，并在此基础上提出了一种 Ceph 集群节能数据副本放置策略。该策略将集群分为运行区和待机区，在考虑用户 SLA 和集群性能的条件下，将顺序存储和随机存储相结合，从节约能耗的角度对 Ceph 集群进行了改进；再次，为进一步节约集群能耗，基于改进的 Ceph 集群建立了一个 Docker 云资源调度模型，并基于该模型提出了一种节能的云资源调度策略。该调度策略将 Ceph 集群和 Docker 容器数据卷的特点相结合，包括应用部署算法和应用迁移算法，其中应用部署算法对 Docker Swarm 编排器进行了改进，而应用迁移算法则解决了应用迁移过程中容器数据卷不能跨主机迁移的问题；最后，用实验验证了两种策略的有效性和可行性。实验结果表明，该 Ceph 集群数据副本放置策略，在满足用户 SLA（Service-Level Agreement）需求和保证集群性能的条件下，使得数据中心耗电量降低了 14.3%；而提出的云资源节能调度策略与原始的 Docker Swarm 调度策略相比，对集群资源进行了更细粒度的划分，提高了集群资源利用率，进一步降低了数据中心能耗。

关键词：Ceph集群；Swarm编排器；数据放置策略；弹性调度；能耗管理

Ceph cluster based energy consumption management strategy study in cloud platform

Abstract

Cloud computing and big data technology have developed rapidly in recent years. From smart healthcare to smart cities, from precision poverty alleviation to precision industry, from government offices to personalized recommendation systems, all of these are showing the changes that cloud computing and big data brought to people's lives. However, at the same time, the huge energy consumption of data centers has caused widespread concern. How to reduce the energy consumption of data center has become a hot topic in the industry and academia.

Firstly, the research background, significance, research status and related basic knowledge of cloud platform energy consumption management are introduced; secondly，aiming at the shortcomings of CRUSH algorithm in energy saving，an energy consumption optimization model based on Ceph clusters is established and a data-place policy for energy saving based on this model is proposed. Ceph cluster can be divided into Operating zone and Standby zone under this strategy, and Sequential storage and Random storage are combined while considering the Service-Level Agreement(SLA) of users and the performance of Ceph cluster, which can improve the Ceph cluster from the perspective of Energey-saving-ability; Besides, in order to further save the energy consumption of Ceph cluster, a energy saving model --Docker cloud resource scheduling based on the improved Ceph cluster is established and a energy-saving cloud resource scheduling strategy, including deployment algorithm and application migration algorithm, based on this model is proposed, which combines the characteristics of Ceph cluster and data volumes of Docker container. The deployment algorithm and migration algorithm are used to improve the Docker Swarm orchestrator and solve the problem that container volumes cannot be migrated across hosts during application migration, respectively; finally, a series of experiments are desighed to verify the effectiveness and feasibility of the two

proposed strategies. The experimental results show that the data-place policy can reduce the energy consumption of the data center by 14.3% under the condition of satisfying the users' SLA (Service-Level Agreement) requirement and ensuring cluster performance. Furthermore, compared with the original Docker Swarm scheduling strategy, the proposed cloud resources scheduling strategy performs a more fine-grained partitioning of cluster resources, which improves the utilization of cluster resources and further reduces the energy consumption of data center.

Keywords: Ceph cluster; Docker swarm; Data-place policy; Elastic scheduling; Energy consumption management

第一章 绪论

1.1 研究背景及意义

近年来，各种各样的云平台层出不穷，基于云平台的应用也随即增多。云计算与大数据是社会和经济发展的一个里程碑，它给人们的生活带来了翻天覆地的变化，而在云计算与大数据技术飞速发展的同时，数据中心巨大的能耗问题得到了人们的广泛关注。在实际生产中，能耗成本占数据中心总体拥有成本（Total Cost of Ownership,TCO）的很大一部分。据统计，2011 年，美国数据中心的耗电量占其总耗电量的 2%[1]，同年，我国数据中心的耗电量占全国用电量的 5%[2]。2015 年，全球数据中心耗电量占世界总耗电量的 1.1%~1.5%，而我国年耗电量超过全社会用电量的 1.5%[3]。而据估计，到 2020 年，数据中心消耗的电能占全球耗电量的比例将上升到 8%[1]。随着云计算数据中心规模的继续扩大，其能耗需求也将成倍增长，而世界电力的最主要来源是火力发电，耗电量过大会加速化石燃料的枯竭、增加 CO2 排放，间接造成环境污染，这使得社会不得不关注数据中心的能耗问题。2014 年 8 月，美国自然资源委员会（Natural Resources Defense Council，NRDC）发布了对数据中心能耗的评估结果，评估结果表明数据中心巨大的耗电量对自然环境造成严重的威胁。2015 年 3 月 18 日，我国工业和信息化部、国家机关事务管理局、国家能源局联合发布了《关于印发国家绿色数据中心试点工作方案的通知》的文件[4]。由上述内容可知，如何降低数据中心能耗已经迫在眉睫，研究数据中心能耗问题具有重大意义。

1.2 国内外研究现状

现今数据中心能耗管理问题已成为工业界和学术界关注的热点问题，大量学者对其进行了深入研究，下面从硬件层和软件层两个方面对数据中心能耗问题的国内外研究现状进行介绍。

（1）硬件层：硬件层相关技术主要用于构建绿色数据中心。面对日益枯竭的化石燃料与数据中心巨大能耗之间的矛盾，绿色数据中心的理念深入人心。绿色数据中心是指数据机房中的 IT 系统、制冷、照明和电气等能取得最大化的能源效率和最小化的环境影响[5]，因而电能、水能、风能、智能电网等新技术被大量的用于构建云计算数据中心。邓维等人在文献[6]中针对新能源不稳定的特点，

从数据中心评估标准、能源按需配给和能量预测等方面对数据中心能耗问题进行了研究，并在最后指出了数据中心新能源应用的不足和发展趋势。李翔在文献[7]中将数据中心的能耗分为计算能耗和制冷能耗两类，它指出制冷能耗占数据中心总能耗的 50%并分析了其原因，然后对现有的能耗评价方法进行分类，最后针对现有热量管理策略不全面、不成熟等不足，提出了一种面向多节点的能耗管理框架。叶冉在文献[8]中针对数据中心能耗问题，从电源和负载量方面对数据中心能源存储设备进行了探讨，并从新能源集成和负载削峰两个方面说明了研究智能储能在构建绿色数据中心的重要性。除此之外采用 DVFS 技术[9-10]来调整 CPU 的电压和频率，从而降低单台服务器的能耗，也是现今数据中心比较常用的方法，但该方法会影响 CPU 的性能，所以在集群负载较高的情况下并不适合使用。以上介绍的方法，虽然能在一定程度上降低数据中心能耗，但对硬件设备要求较高，并需要提前对数据中心做好整体规划，而对于已存在的数据中心，由于已有数据中心结构、场地、供电系统等的限制，要进行转型非常困难。

（2）软件层：软件层主要是从集群负载重新分配、云资源弹性调度、虚拟机迁移等方面出发，将云计算应用整合在较少的服务器中运行，并通过关闭数据中心部分服务器或将部分服务器置于休眠状态来降低数据中心能耗。Kaushik 在文献[11]中首次将集群分为热区和冷区两个部分，其中热区使用性能较高的服务器存储热数据，冷区使用性能较差的服务器存储冷数据，并针对冷区提出了一种基于能耗感知的应用数据放置方法，该方法通过将冷区负载较低的节点调整到休眠状态来节约数据中心能耗，但他并没有提出针对热区的能耗管理方法。郝亮在文献[12]中针对不同规模的随机任务建立了四种模型，并基于这四种模型提出了基于时间预测、任务忍耐值、资源相关性、执行成本的四种算法，对云计算平台的能耗问题进行了优化，但提出的方法过于繁琐，且对随机任务规模的大小没有给出定量的分析。黄庆佳在文献[13]中提出了一种能耗成本感知的云数据中心资源调度机制，从大规模并行化任务处理能耗、虚拟机动态整合能耗和跨地域多数据中心能耗成本三个方面解决数据中心能耗最小化问题。此外，改进的蚁群算法[14-17]、神经网络算法[18-20]、遗传算法[21-24]等也被广泛的用来解决该类问题。针对特定的云平台，Kang D K 等人在文献[25]中针对 Docker 云平台数据中心能耗问题，提出了一种负载感知的异构容器集群能效管理方法，该方法利用 K-medoid

算法对云平台应用进行分类，在此基础上建立了一个异构集群的能效模型，最后用不同类型的应用进行实验测试，证明该调度方法能在保证服务性能的情况下，节约数据中心能耗，降低云平台运营成本。Meng Y 等人在文献[26]中针对容器应用在不同阶段所需计算资源不同所造成的资源浪费和集群动态扩展带来的性能损失问题，提出了一种基于时序分析的资源预测模型，该模型根据历史负载数据对容器下一时隙所需的计算资源进行预测，然后根据预测值，利用容器技术实现应用的快速弹性收缩，达到了合理部署云平台资源的目的。Guan X 等人在文献[27]中提出了一种基于 Docker 容器的面向应用的数据中心资源分配方法，该方法针对不同应用的资源使用情况，综合考虑物理机资源和容器资源，建立一个资源优化模型，并考虑到 Docker 容器资源分层复用的特点，实现了对数据中心资源的合理调度，进而达到降低集群能耗的目的。何松林在文献[28]中利用 Docker 虚拟化技术，构建了一个可弹性扩展的弹性集群，并针对集群的资源使用率问题，提出了基于容器和集群节点负载历史数据的调度策略、弹性收缩容器时的宿主机调度策略和改善用户响应延迟的负载预测调度策略。以上介绍的方法，虽然能在一定程度上减低数据中心能耗，但他们要么只对数据中心的能耗进行评估，并没有考虑到云平台资源利用率问题；要么只考虑应用部署时的资源调度问题，要么只考虑应用迁移过程中的云资源整合问题，都没有同时考虑应用数据存储、应用部署和应用迁移给数据中心能耗带来的影响。

本文针对 Ceph 集群能耗问题，在同时考虑了应用数据存储、应用部署和应用迁移的情况下，基于 Ceph 集群和 Docker Swarm 集群，提出了一种面向云环境的 Ceph 集群能耗管理策略。首先，对 Ceph 集群进行分析，并建立了一个集群能耗优化模型。在此基础之上，针对 CRUSH 算法在节能方面的不足，提出了一种新的 Ceph 集群节能数据副本放置策略，以此对 Ceph 集群进行了改进；其次，基于改进的 Ceph 集群和 Docker 容器虚拟化技术，建立了一个 Ceph 集群节能资源调度模型，并基于该模型提出了一种节能的云资源调度策略，该策略从应用数据存储、应用部署和应用迁移三个角度出发，对云平台资源进行了更细粒度的划分，提高了云资源利用率，并实现了应用容器的在线迁移。最后，对提出的策略进行实验和对比分析，实验结果表明，本文提出的 Ceph 集群能耗管理策略，能在满足用户 SLA 需求的情况下，提高云平台资源利用率和降低 Ceph 集群能耗。

同时，本文提出的节能资源调度策略与常见的 Docker 调度策略相比，不仅对集群资源进行了更细粒度的划分，实现了容器应用的跨主机迁移，提高了集群资源利用率，而且在应用迁移方面具有一定的优势，达到了进一步降低 Ceph 集群能耗的目的。

1.3 主要研究内容

本文首先对数据中心能耗管理策略的研究背景意义和国内外研究现状进行了归纳总结，接着对 Openstack 云平台、Ceph 分布式存储系统、Docker 容器虚拟化技术的组件、工作原理和整个云平台的架构分别进行了介绍。然后，针对 CRUSH 算法在节能方面的不足，对其进行了改进，提出了一种 Ceph 集群数据副本放置策略，使得集群在满足用户 SLA 需求的前提下，降低数据中心能耗；为进一步节约集群能耗，将改进的 Ceph 集群与 Docker 容器虚拟化技术相结合，提出了一种节能的云资源调度策略，通过屏蔽 Docker 数据卷不能跨主机访问的缺陷，实现了应用容器的跨主机迁移，并提高了云平台的资源利用率。本文主要从应用数据存储、应用容器部署及应用容器迁移三个方面研究了 Ceph 集群能耗管理问题，现将主要内容总结如下：

（1）阅读大量相关的国内外文献，对现有的云平台能耗管理策略进行比较和总结，指出其中的优点和不足之处，并巧妙的将开源云计算管理项目 Openstack、分布式存储系统 Ceph 和容器虚拟化技术 Docker 相结合构建了一个云平台，并在此基础上研究云平台的能耗管理问题。

（2）对分布式存储系统 Ceph 的存储原理进行深入研究，分析其优缺点。然后针对 Ceph 集群的数据副本放置算法——CRUSH 算法，在数据中心节能方面存在的不足，以集群节点的 CPU 平均使用率和用户 SLA 中的应用完成限制时间为参数，建立一个 Ceph 集群能耗优化模型。

（3）基于该能耗优化模型提出一种 Ceph 集群节能数据副本放置策略，该策略针对 Ceph 集群 CRUSH 算法在节能方面的不足，将 Ceph 集群分为运行区和待机区两个部分，在运行区中引入了数据恢复域，且采用了顺序存储和随机存储相结合的存储方式，并通过将集群中的部分空闲节点置于待机状态来节约集群能耗。最后用实验证明，本文提出的数据副本放置策略，能在保证 Ceph 集群性能和用户 SLA 需求的前提下，使 Ceph 集群的能耗降低了 14.3%。

（4）将改进了数据副本放置策略的 Ceph 集群与 Docker 容器虚拟化技术相结合构建了新的云计算基础平台，指出了 Docker 容器数据卷（volume）不能跨主机访问，使得应用容器不能跨主机迁移的问题，然后对其进行分析并建立了一个节能云资源调度模型。该模型将集群节点的 CPU、内存和网络的平均使用率做为主要参数，并分别引入了对应的资源调节参数，针对不同类型的应用将集群节点分成三种状态，对集群资源进行了细粒度的划分。

（5）基于该节能云资源调度模型，提出一种云资源弹性调度策略。该策略包括了应用容器部署算法和应用容器迁移算法。其中应用部署算法与提出的 Ceph 集群节能数据副本放置策略相结合，解决应用在部署时的资源分配问题；应用迁移算法，解决了集群节点负载过高时的资源整合问题。最后用实验证明，该策略对集群资源进行了细粒度划分，实现了应用容器的跨主机迁移，提高了云平台资源利用率，进一步了降低了 Ceph 集群能耗。

1.4 论文组织结构

本文一共分 5 个章节来阐述所研究的能容，论文结构如下：

第一章 绪论：首先阐述了云平台能耗管理问题的研究背景与意义，并对其国内外研究现状进行了概述。接着介绍了本文的主要研究内容。最后介绍了本文的组织结构。

第二章 相关理论知识：对 Openstack 云平台管理系统、Ceph 分布式存储系统、Docker 容器技术的架构和工作原理进行了简单介绍，并给出了利用该三种技术构建的云平台架构图。

第三章 改进的 Ceph 集群节能数据副本放置策略：针对 Ceph 集群原有的数据副本放置策略——CRUSH 算法在节约集群能耗方面的不足，对整个云平台进行深入分析，建立一个 Ceph 集群能耗优化模型，在该模型的基础上提出一种新的 Ceph 集群数据副本放置策略，对 Ceph 集群进行了改进。最后用实验证明了该数据副本放置策略的有效性，实验结果表明不仅提出的数据副本放置策略能在保证用户 SLA 请求的情况下降低 Ceph 集群能耗，而且提出的能耗优化模型能在一定程度上预测数据中心能耗的变化趋势，为数据中心的能耗管理奠定了良好的基础。

第四章 基于改进 Ceph 集群的节能云资源调度策略：为了能进一步降低数

据中心能耗，本章在第三章的基础上，针对 Docker 数据卷不能跨主机迁移的不足，将改进后的 Ceph 集群与 Docker 数据卷特性相结合。首先建立一个 Docker 节能的云资源调度模型。该模型以集群服务器的内存、CPU 和网络使用率为参数，并分别引入了对应的资源调整参数；然后基于该模型提出一种节能的云资源调度策略。该策略包括应用容器部署算法和应用容器迁移算法。最后用实验证明该调度策略对云平台资源进行了细粒度的划分，实现了 Docker 容器跨主机迁移，提高了集群资源利用率，达到了降低数据中心能耗的目的。

第五章 总结与展望：对本文所做的工作进行了总结，指出论文中存在的不足和下一步要研究的方向。

第二章 相关理论知识

信息技术的飞速发展伴随着大量数据的产生，为了满足人们对美好生活的追求，云计算与大数据技术应运而生。从当初的蜿蜒探索，到现在的日趋成熟，云计算与大数据技术给人们的生活带来了积极的影响，但与此同时，也带来了一些问题。云平台作为云计算与大数据技术的基础，其能耗过高问题得到了国内外工业界和学术界的广泛关注，很多学者对此作出了大量的研究[29-36]。

本章首先介绍了 Openstack 云平台管理系统、Ceph 分布式存储系统和 Docker 容器技术等理论知识，然后在此基础之上讲述了这三种热门技术构成的大型 Openstack 云平台的基本架构，最后进一步明确指出了本文的研究内容。

2.1 云计算与 Openstack

1、云计算概述

2006年8月谷歌首席执行官埃里克在“Google 101计划”中首次正式提出了云计算的概念，在过去的十几年时间里，云计算发展迅猛，给人们的生活带来了非常大的变化。自云计算提出以来，Google、亚马逊和阿里巴巴等国内外大型互联网公司纷纷建立了自己的云计算平台，出现了私有云、公有云和混合云三种形态的云计算基础平台，并衍生了边缘计算，其应用范围也从当初的大学校园扩大到农业、工业和服务业等行业，总之云计算技术正在日益改变着人们的生活。云计算技术发展如此迅速，但云计算的概念却至今仍没有一个非常准确的定义，但大多数云计算专家认为：云计算是一种按使用量付费的模式，这种模式提供可用的、便捷的、按需的网络访问，进入可配置的计算资源共享池（资源包括网络，服务器，存储，应用软件，服务），这些资源能够被快速提供，只需投入很少的管理工作，或与服务供应商进行很少的交互[37]。无论云计算的概念和云计算技术如何发展变化，云计算著名的三层架构模型却始终对学习和研究云平台技术具有重要的指导意义，该三层模型如图 2.1 所示。从下往上分别为：基础设施即服务层（Infrastructure as a Service，IaaS）、平台即服务层（Platform as a Service，PaaS）和软件即服务层（Software as a Service，SaaS ）[38]，而开源云计算管理平台 Openstack 属于云计算基础架构中的 IaaS 层，它几乎提供了 IaaS 层的所有云计算功能，管理着数据中心成千上万台的物理服务器，是上两层架构的基础[39]。

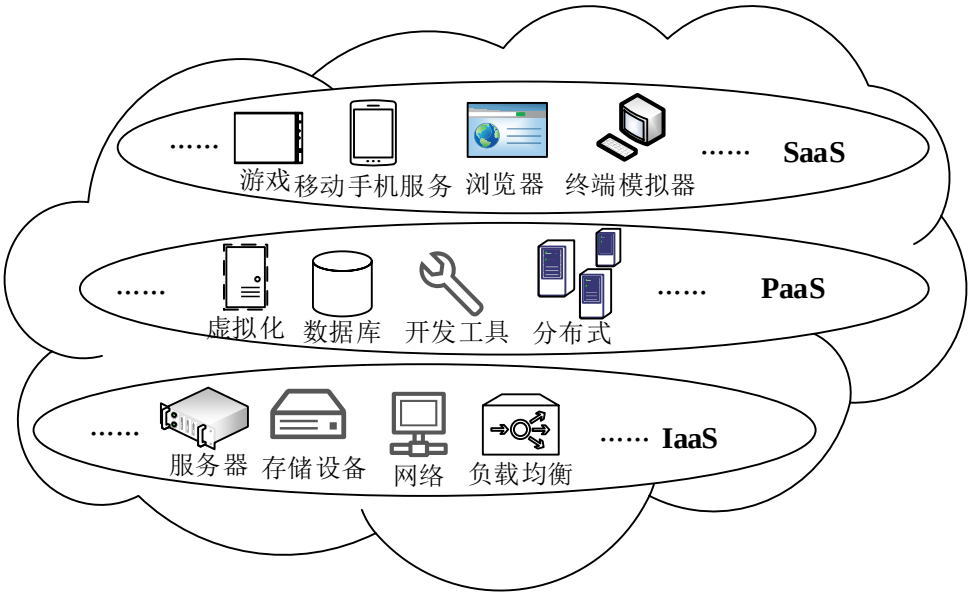


图 2.1　云计算基础架构图

2、Openstack 概述

Openstack 是一个开源的云计算平台项目，由美国国家航空航天局（National Aeronautics and Space Administration，NASA）和 Rackspace 于 2010 年共同发起，是现今比较成熟的开源云平台管理项目[40]。Openstack 底层支持 KVM、VMware、XEN、QEMU、VirtualBox 等虚拟化技术，基于其强大的虚拟化技术实现了对数据中心服务器资源的综合管理和弹性扩展[41]。Openstack 能将廉价的物理机整合成能够统一管理和支配的资源池，这受到了现今广大厂商的青睐，因此被广泛的用于构建大型的云计算基础平台，包括私有云、公有云或混合云。由于开源和各大云计算供应商的广泛支持，Openstack 发展迅速，随着功能的不断完善，其包含的组件也由早期的 Horizon、Keystone、Neutron、Nova、Glance、Cinder、Swift 七大组件扩展到 Heat、Cellometer 等 9 大组件，每个组件都是一个软件模块，具有不同的功能，下面进行简单的介绍[42]：

Horizon 组件：一个 web 展示页面，方便用户使用或管理 Openstack 云平台；

Keystone 组件：为访问 Openstack 云平台的用户提供身份认证功能，只有认证通过后的用户才能访问 Openstack 平台中的服务；

Nova 组件：主要负责云虚拟机实例（Compute 或 Instance) 的生成、监测、终止等管理功能，是 Openstack 的核心组件之一；

Neutron 组件：提供虚拟网络服务，为每个不同的租户建立独立的网路环境，

最早是 Nova 组件中 Nova-network 模块，后分离出来取名 Quantum，后由于商业问题更名为 Neutron，是 Openstack 的核心组件之一；

Glance 组件：为 Openstack 云平台虚拟机实例提供镜像服务；

Cinder 组件：提供 Openstack 中的块存储功能，一般与网络文件系统（Network File System，NFS）、Ceph 分布式存储存储系统等配合使用；

Swift 组件：提供 Openstack 中的对象存储功能，存储一些资源文件，如图片、代码等文件；

Heat 组件：Heat 是一个基于模板来编排复合云应用的服务，目前支持基于 JSON 格式的 CFN 模板和基于 YAML 格式的 HOT 模板。它从基础架构、软件的配置和部署、资源自动伸缩、负载均衡四个方面实现对 Openstack 云平台资源的编排，简化了用户对 Openstack 云平台的管理。

Cellometer 组件：为 OpenStack 系统提供资源监控功能，为上层的计费、结算或者监控应用提供统一的资源使用数据收集功能。

3、Openstack 与云计算的关系

在过去的十几年时间里，随着云计算技术发展迅速，Openstack 作为云计算经典架构中 IaaS 层的一种开源云平台管理项目也逐渐发展成熟，其功能不断完善。由于云计算应用非常广泛，云计算的应用场景和需求更是千变万化，因此国内外许多大型云计算厂商构建的云平台都不尽相同。虽然云计算研发人员构建的云平台形式多样，但他们一定程度上都参考了 Openstack 的功能和架构，所以很大程度上我们可以认为 Openstack 是一种云平台架构，他向人们解释了什么是云平台，云平台应该解决什么样的问题，应该怎么样来解决数据存储、资源管理和网络部署等相关问题，因此 Openstack 作为一个云计算基础平台对研究云计算相关技术具有重大意义。

2.2 Ceph 分布式存储系统

1、Ceph 概述

Ceph[43]由是由 Sage 等人于 2012 年 3 月提出的一种分布式存储系统，是一种典型的软件定义存储（Software Define Storage，SDS）实现。Ceph 提供了文件存储、对象存储、块存储三种存储方式的统一解决方案。与传统的存储系统不同，Ceph 解决了存储系统中的单点故障问题，更由于其易用性、高可扩展性、高可

靠性等优点得到了工业界的广泛青睐[44]，无论是公有云、私有云、还是混合云，Ceph 都能为其提供很好的后端存储方案，因此 Ceph 被大量的用于构建云平台的后端存储系统[45]。

2、Ceph 组件及其基本架构

Ceph 主要由 Client、MON 和 OSD 三个组件组成，Client 提供用户与 Ceph 集群的交互命令行；MON 负责监控整个集群的状态；OSD 负责存储 Ceph 集群中的数据。Ceph 的基本架构[46]如图 2.2 所示。

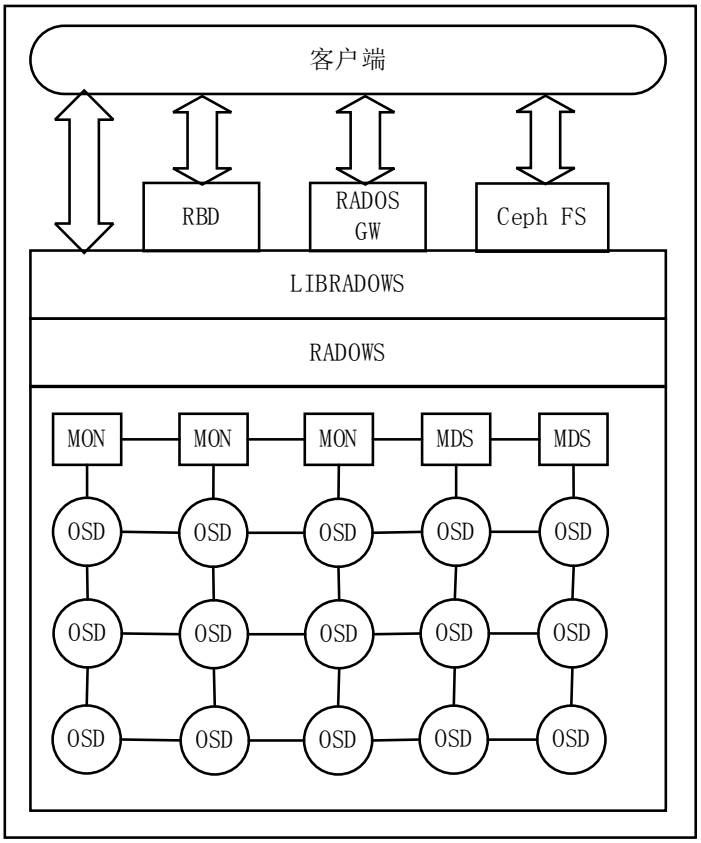


图 2.2　Ceph 基本架构图

Ceph monitors：Ceph 监视器，简称 MON，负责监控和维护 Ceph 集群的健康状态。集群中的每个 OSD 节点都要定期向 MON 节点汇报自己的状态信息，MON 节点维护着 Ceph 集群的 OSD Map、MON Map、PG Map、CRUSH Map 等集群状态映射信息；

Ceph Object Storage Device (OSD)：Ceph 对象存储设备，Ceph 集群中的数据都以对象的形式存储在 OSD 中，OSD 是 Ceph 集群中唯一能存储用户数据的组件。在商业应用中，一般一个 OSD 就对应着一块物理磁盘；

Ceph Metadata Server(MDS)：Ceph 元数据服务器，只用于管理 CephFS 文件系统的层次结构和元数据，因此 Ceph 块存储中并不涉及 MDS 进程；

Reliable Autonomic Distributed Object Store(RADOS)：由于用户数据都是以对象的形式存储在 Ceph 集群中，而 RADOS 则负责将任何格式的用户数据以对象的形式存储在 Ceph 集群中，因此，RADOS 是 Ceph 集群的基础。另外，RADOS 层还能通过复制数据、故障检测和恢复功能来保证用户数据的一致性；

Librados：为 PHP、JAVA、PHP、C、C++、python、ruby 等编程语言提供访问 RADOS 的接口，同时为 RBD、RGW 和 CephFS 提供了访问 RADOS 的原生接口，另外还允许上层应用通过 Librados 直接访问下层的 RADOS，进而访问整个 Ceph 集群；

RADOS Block Devices (RBDs)：是 Ceph 中提供持久化块存储的设备，它精简可靠的将用户数据存储在 Ceph 集群的多个 OSD 中，已经被封装成基于 Librados 原生接口；

RADOS Gateway interface (RGW)：提供对象存储服务，RGW 允许应用通过 Librgw (the Rados Gateway Library) 和 Librados 与 Ceph 对象存储进行交互；

CephFS：CephFS 利用 Ceph 存储集群将用户数据存放到文件系统中，并与 POSIX 兼容。

传统架构中，用户要访问集群，都必须先访问元数据服务器中的元数据列表来查询要访问的目的主机；而在 Ceph 架构中，客户端能通过上层的 RADOS 库直接访问 Ceph 集群的 OSD 和 MON 节点，而 Ceph 集群中的 MON 和 OSD 都是集群感知的，它们能够与集群中的其它 OSD 进程和 MON 进程通信，此外，客户端通过认证后，Ceph 允许客户端根据其对应的 Cluster Map 直接访问集群中的 OSD。因此与传统方式相比，Ceph 不仅解决了传统集群中的单点故障问题，还打破了集群从 PB 级扩展到 EB 级 I/O 瓶颈。而除此之外，RADOS 的多数据副本、数据检测和数据恢复等机制，使得 Ceph 集群具有高可靠性，从而保证了集群的稳定性和用户数据的高可用性。

3、Ceph 客户端认证过程

客户端想要访问 Ceph 集群，必须先通过 MON 认证，其认证过程[47]如图 2.3 所示：

① 客户端 Client 把自己的用户名发送给 MON 并提出创建新用户的请求；

② MON 生成一个用客户公钥加密过的会话秘钥，MON 会存储该会话秘钥并将其发送到对应的客户端。与此同时，MON 会与集群中所有的 MON、OSD 和 MDS 分享该会话秘钥，这样客户端就能与集群中的各个组件进行会话了，但还不能访问集群中的服务；

③ 客户端收到共享秘钥后，用私钥解密，就得到会话秘钥，这样就在该会话中标记了该客户；

④ 客户端向 MON 请求获得一个用该会话秘钥加密过的访问凭证；

⑤ MON 产生用一个用该用户会话秘钥加密过的集群访问凭证，并回传给客户端，客户端收到一个访问集群的凭证；

⑥ 客户端收到一个访问集群的凭证后，就能访问 Ceph 集群服务了。它可以向集群中的所有 MON 和 OSD 提出请求。

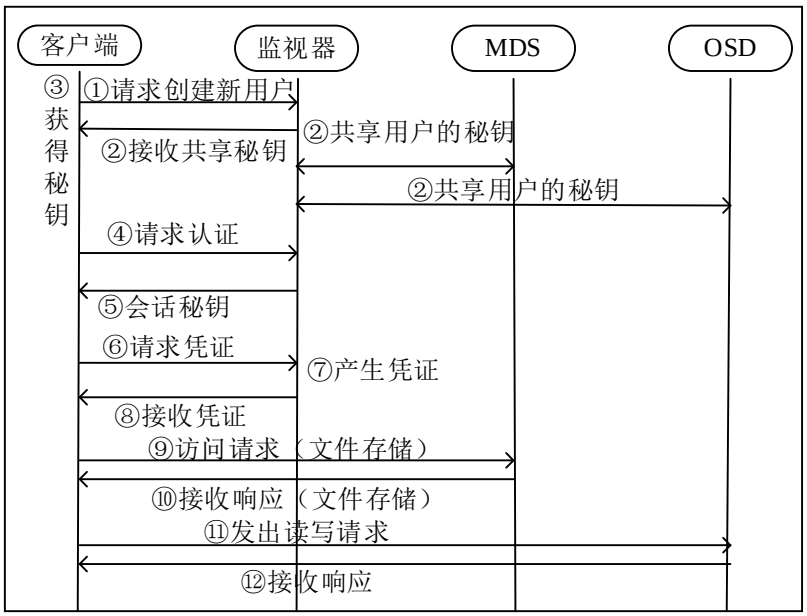


图 2.3 Ceph 客户端认证过程

从上面的认证过程可以看到，客户端要访问 Ceph 集群提供的服务，需要获得会话秘钥和凭证，其中会话秘钥允许客户端与 Ceph 集群建立连接，而凭证则允许客户端访问 Ceph 集群服务，两者缺一不可。

4、Ceph 客户端写数据过程

通过上一小节的介绍，我们知道 Ceph 客户端要访问 Ceph 集群提供的服务之前，必须要先通过 MON 认证。同样，当客户端要向 Ceph 集群写数据时，客

户端必须先获得 MON 认证，认证通过之后才能获得向 OSD 中写数据的权限。同时，MON 会将集群状态图 Cluster Map 发送给 Ceph 客户端，Ceph 客户端在向集群中写数据之前，会经过 DATA STRIPING 过程将用户数据分片，并将分片后的数据封装成 Object 对象，这些 Object 对象会映射到 Pool（Ceph 集群中的一个逻辑分区概念，Pool 可以设置 Object 的访问权限或所有权、PG（Pool Group）数量和所使用的 CRUSH 规则参数）中，CRUSH 算法会根据 Pool 中设定的参数，计算出用户数据应该存放在哪些 OSD 中，最后把 PG 映射到相应的 OSD 中，实现用户数据的存储，整个过程[47]如图 2.4 所示。

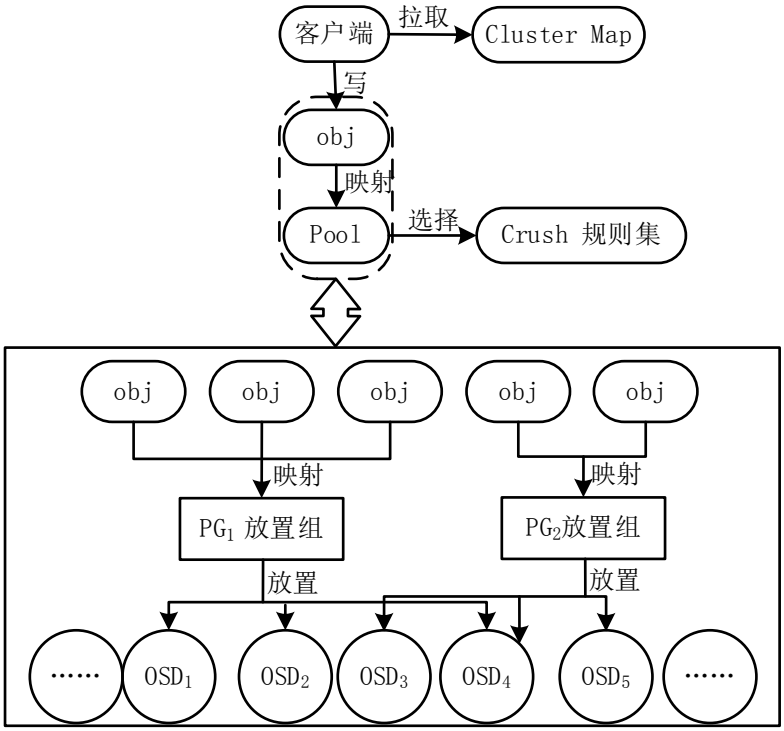


图 2.4　PG 映射过程

值得注意的是，Ceph 集群默认采用 3 个数据副本存储策略，3 个数据副本分别存储在 3 个不同的 OSD 中，这 3 个 OSD 组成一个活动集（Active set），活动集中会有一个主 OSD 和两个次 OSD，只有主 OSD 能接受客户端的写数据请求，而其余两个次 OSD 中的数据由主 OSD 负责写，当相应的数据副本写完之后，次 OSD 向主 OSD 返回写完成信息，最后主 OSD 会向客户端发送写完成信息，此时用户数据才算写入到 Ceph 集群中。在写数据的过程中，Ceph 通过数据的版本号保持同一个活动集中 OSD 数据的一致性。处在同一个活动集中的 OSD 之间会定时通信，若发现主 OSD 出现故障，则其中的一个次 OSD 会被选举成为主

OSD，主 OSD 将该用户数据写入到集群中另外一个 OSD 中，并向 MON 报告新的状态；若发现次 OSD 出现故障，则主 OSD 直接将数据写入集群中的另一个 OSD 中，并向 MON 报告新的存储状态，该过程即为 Ceph 集群自我恢复数据过程。Ceph 活动集中的写数据过程[47]如图 2.5 所示。

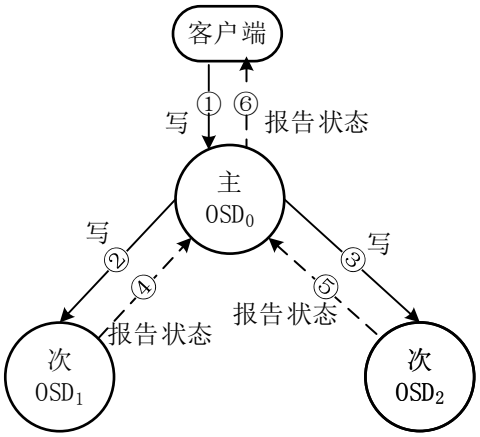


图 2.5　Ceph 活动集中的数据写过程

Ceph 的这种写数据机制，大大减少了客户端的 I/O 压力，其自我修复行为更保证了集群的稳定性、可靠性和用户数据的高可用性、安全性。

2.3 Docker 容器技术

1、Docker 容器概述

Docker[48]容器技术是近年来出现的一种操作系统级虚拟化技术。与传统的虚拟机不同，容器（Container）的运行不需要安装额外的客户机操作系统，而是基于 Docker 引擎（Docker Engine）以进程的形式直接运行在宿主机操作系统中，与其它容器共享一个 Linux 内核并且互不干扰，而除此之外 Docker 还支持 AUFS，Btrfs，VFS，和 Device Mapper 等联合文件系统（UnionFS），此外 Docker 镜像采用分层存储架构，Docker 的这些特性使得它比传统的虚拟机具有更高的资源利用率、更快速的启动时间、更轻松迁移、更易维护和扩展等优点[49]。

2、Docker 组件及基本架构

Docker 采用 C/S 架构，其基本架构如图 2.6 所示。Docker 的核心组件包括客户端、服务器端、镜像、仓库和容器（Container），下面分别对其进行简单介绍：

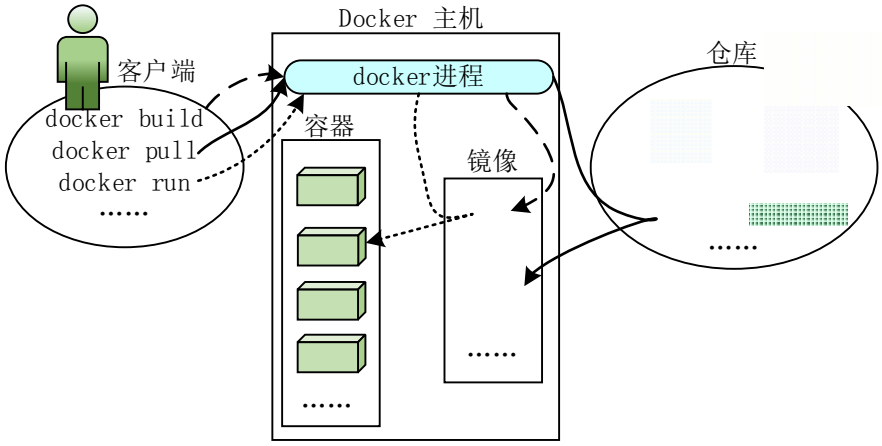


图 2.6　Docker 架构图

客户端（Client）：通过命令行或其它工具与 Docker 进行交互，一个客户端可同时与多个 Docker 进程通信；

服务器端（Docker_HOST）：调用 Docker API 执行客户端提交的命令；

Docker 进程（Docker daemon）：用于监听 Docker API 和 Docker 对象（包括镜像、容器和服务等），一个 Docker 进程可以通过与其它 Docker 进程通信来管理 Docker 服务；

镜像（Images）：用于创建 Container 的模板；

仓库（Registries）：用于存放镜像；

容器（Container）：一个容器就是一个镜像的可运行实例，以进程的形式运运行在宿主机中。

当需要创建一个 Container 时，客户端会将命令通过 Docker 进程提交到服务器端，服务器端先检查本地是否有对应的镜像，若有则直接根据该镜像生成一个 Container，若没有对应的镜像则先从仓库中把对应的镜像一层一层的下载到本地，然后创建相应的 Container，所有 Container 都以进程的形式运行在宿主机操作系统中，各进程之间相互隔离互不干扰。需要指出的是，Docker Container 将数据存储到数据卷 Volume 中，而数据卷不能跨主机访问，这使得运行在 Docker Container 中的应用的在线迁移存在着困难。

3、Docker Swarm 编排器

2014 年 12 月，Docker 公司发布了 Docker Swarm 的第一个版本。Docker Swarm 采用 Go 语言开发，是 Docker 原生的一种容器编排工具，它能将多个

Docker 主机虚拟成一个单一的 Docker 主机，并进行统一管理。由于 Docker Swarm 采用标准的 Docker API，因此任何形式的 Docker 客户端（go, docker_py, docker 等）都能直接与 Swarm 集群通信。Docker Swarm 采用典型的 Master/Slave 结构，其架构如图 2.7 所示。

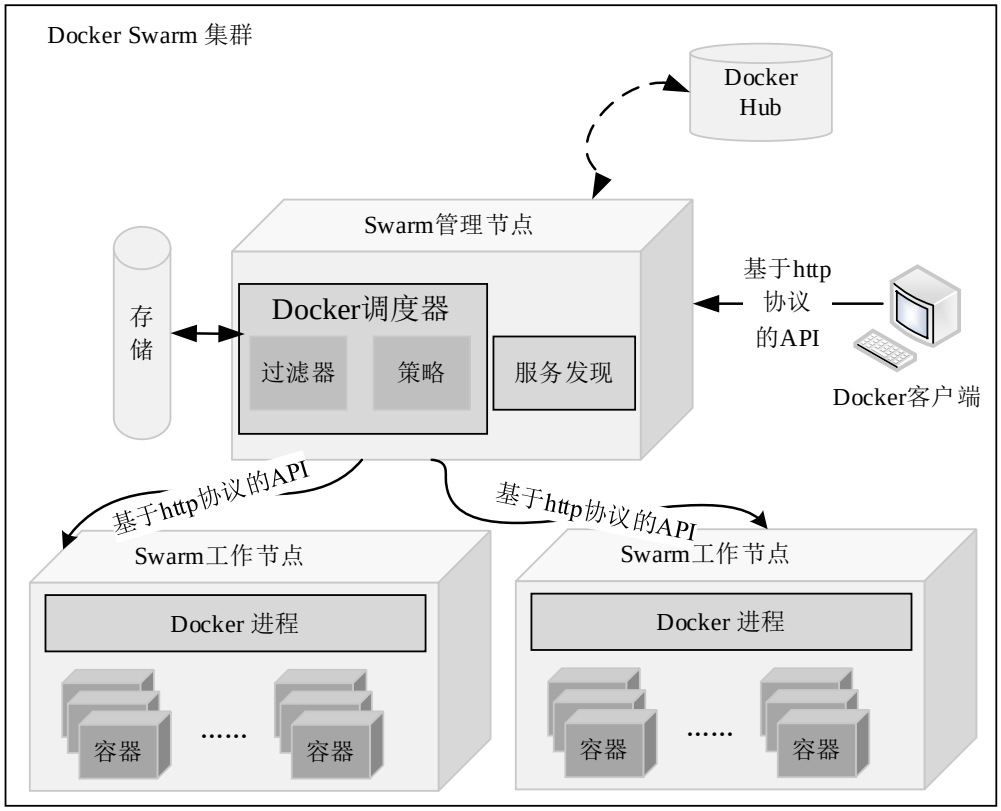


图 2.7　Swarm 基本结构图

Docker Swarm 集群包含两种不同的节点，分别是 Manager 节点和 Worker 节点[50]。Manager 的作用有：1、管理集群状态信息并维护集群数据的一致性；2、对集群服务进行调度，接收客户端服务请求，并将用户请求以 Task 的形式分发给多个 Worker 节点执行；3、提供 Http API 服务。一个 Swarm 集群中可以有多个 Manager 节点，但只能有一个主 Manager 节点。一个 Swarm 集群中的节点，可以既是 Manager 节点又是 Worker 节点。Worker 节点主要完成 Manager 节点下发的 Task。

Task 是 Swarm 调度的最小单位，一个 Task 通常包括一个 Container 及与对应服务相关的命令，因此，Swarm 集群中的 Container 调度工作是由 Manager 节点来完成的。Swarm 中有几种自带的容器过滤器（Filter），用于筛选符合条件的 Worker 节点，此外 Swarm 的调度器还可以按照指定的调度策略(Strategy)自

动分配容器到节点，具体的 Swarm 集群的工作流程[51]如图 2.8 所示。

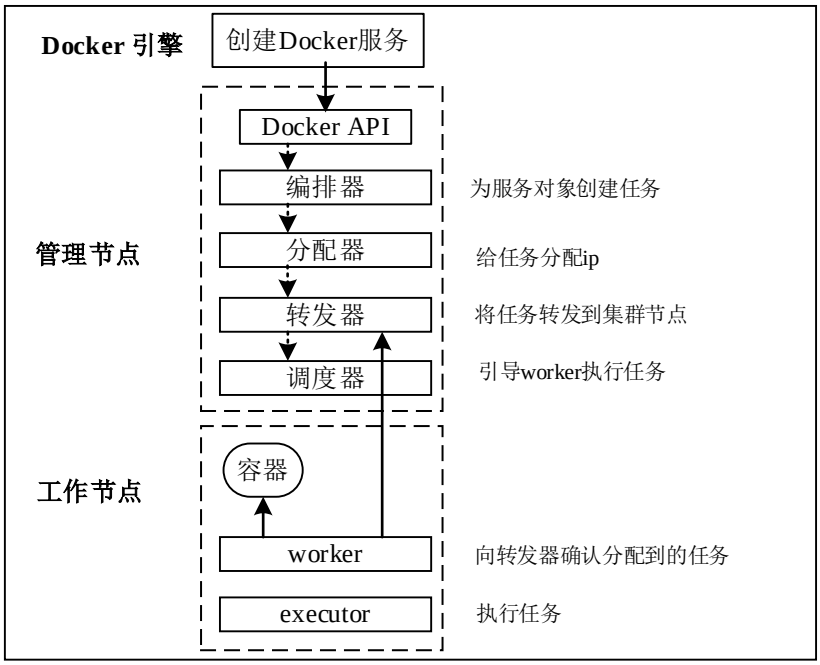


图 2.8　Swarm 集群工作原理图

2.4 Openstack、Ceph 和 Docker 的整合

随着数据的飞速增长，传统的存储方案已经很难满足需求。Ceph 作为一个开源的分布式存储系统，能够为数据存储提供统一的解决方案，并且完全满足 TB 级的数据存储需求，此外其易扩展性、高可用性、自我检测和自我恢复等优点，也令其在众多存储方案中脱颖而出，受到广大云计算厂商的青睐，经常被用于构建大型的云平台后端存储系统。把 Ceph 作为 Openstack 云平台的后端存储系统就是一种非常典型的云平台构建方案，据 2015 年的数据显示，62%的 Openstack 云平台的后端存储系统都是 Ceph 集群，这与 Ceph 和 Openstack 两者的特点是分不开的。Openstack 作为一个开源的云平台基础设施管理软件，主要以虚拟机的形式提供服务，而 Docker 作为一种新兴的虚拟化技术，其高效的资源利用率、秒级启动、易于迁移和部署等优点，为构建 Openstack 云平台提供了新思路。目前 Openstack 官方已经集成了与 Ceph 和 Docker 相关的 API，通过配置文件和调用 API，可以实现三者的无缝结合，图 2.9 是三者结合的典型云平台架构。

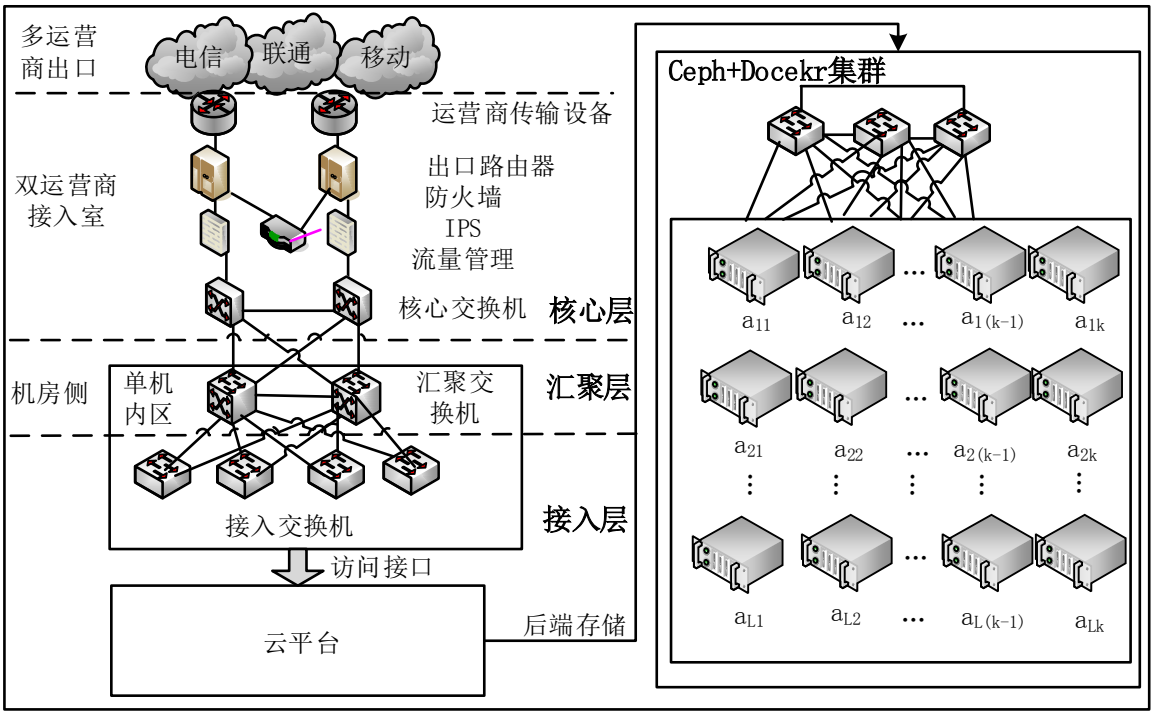


图 2.9　云平台架构图

2.5 本章小结

本章对与本文研究相关的云平台理论知识进行了介绍，其中包括了开源云平台项目 Openstack、分布式存储系统 Ceph 和 Docker 容器虚拟化技术的基本架构及原理。除此之外，还讲述了云计算与三者之间的关系：首先，指出 Openstack 处于经典云计算三层架构中的最底层——基础设施即服务层（IaaS 层），它为云计算架构的上两层——平台即服务层（PaaS 层）和软件即服务层（SaaS 层）服务提供基础；然后，指出 Ceph 不仅为块存储、对象存储和文件存储提供了统一的存储方案，而且具有易用性、高可扩展性、高可靠性、自我检测故障及自我恢复等优点，被广泛的用于构建大型云平台的后端存储系统，而由于 Ceph 与 Openstack 的自身的优点和特点，两者能够实现无缝结合；最后，指出 Docker 容器虚拟化技术与传统的虚拟机技术相比，具有能更高效的利用系统资源、更快速的启动时间、更轻松的迁移、更容易维护和扩展等优点，而 Openstack 通过虚拟机提供服务，因此能够将 Docker、Openstack 与 Ceph 三者结合，以此来构建高效稳定的云平台，并给出了一个将三者相结合的云平台基本架构。本章介绍的内容，将为后面的第三章和第四章提供重要的理论依据和支撑。

第三章 改进的 Ceph 集群节能数据副本放置策略研究

3.1 问题描述

Ceph 分布式存储系统具有易用性、高可扩展性、高可靠性、故障检测及故障恢复等优点，为云平台提供了块存储、对象存储和文件存储的统一存储解决方案，且没有单点故障，这从系统的高效性、稳定性和便于管理的角度来看都是非常重要的，而 Sage 等人在设计时特别注重 Ceph 文件系统的性能，并没有过多的考虑集群能耗问题，这使得 Ceph 集群数据副本放置策略——CRUSH 算法[52]，在节约能耗方面存在着不足。CRUSH 算法是一种伪随机 Hash 值散列算法，该算法用于计算部署应用数据的目的主机，Ceph 集群根据该计算结果将对应的用户数据映射到该目的主机的 OSD 中，即经过 Hash 计算得到数据的存储位置，因此 Ceph 总是尽可能的将用户数据均匀地存放在 Ceph 集群中。而事实上，集群中大多数服务器在大部分时间都处于低负载状态，这就造成了集群资源的严重浪费，同时这也是 Ceph 集群能耗高的一个重要原因，而良好的数据副本放置策略能够降低数据中心能耗，因此本文提出了一种 Ceph 集群节能数据副本放置策略。

服务等级协议（Service Level Agreement，SLA）是云计算服务供应商与客户签订的一项重要协议，其中定义了服务类型、服务质量和客户付款等术语[53]。云服务供应商都有自己的数据中心 SLA 技术规范，一般数据中心 SLA 中定义的服务指标有：数据中心机房环境、能同时服务的客户数目、分配给用户的最小带宽及带宽极限、应用完成时间等，其中用户应用完成时间是 SLA 中定义的最能够体现云计算服务性能的一项硬指标，因此本文在建立能耗模型时，以应用完成限制时间 T_t 作为衡量集群服务性能的一个重要参数。此外，大量研究表明[54-57]，CPU 的使用率与数据中心的能耗呈正相关关系，因此本文将 CPU 的平均使用率作为能耗优化模型中的另一个重要参数。

3.2 能耗优化模型

存储系统是云计算数据中心的重要组成部分，数据中心的绝大多数物理服务器都是存储节点，而大多数云平台的后端存储系统都采用 Ceph 集群，因此本文主要研究基于 Ceph 集群的能耗管理策略。

设 Ceph 集群节点分为运行区（$zone = A$）和待机区（$zone = B$）且分别位于不同的机架上，运行区一共有 a 个机架，机架编号为 $rack_r(0 \le r < a)$，用于正常的业务需求；待机区一共有 b（$b \ge \left\lceil \frac{a}{2} \right\rceil$）个机架，编号为 $rack_v(0 \le v < b)$，当集群负载低时，待机区的服务器处于休眠状态；当集群负载较高时，待机区的节点将从休眠状态唤醒，切换到运行态对外提供服务。每个机架中都有 m 台服务器，从下向上的服务器编号为 $host_h(0 \le h < m)$。每台服务器就是一个 Ceph 集群节点，假定每个服务器上有 n 块磁盘，每一块磁盘对应一个 Ceph 集群的 osd，则运行区一共有 $N = a \cdot m \cdot n$ 个 osd，编号为 $0 \le i < (a \cdot m \cdot n - 1)$；待机区一共有 $b \cdot m \cdot n$ 个 osd，编号为 $a \cdot m \cdot n \le i < (a + b) \cdot m \cdot n$，Ceph 集群的目录树结构如图 3.1 所示。通过设置 $zone_x.rack_r.host_h.osd_i$ 中 x、r、h、i 的值，便可指定集群中的任意一个 osd。

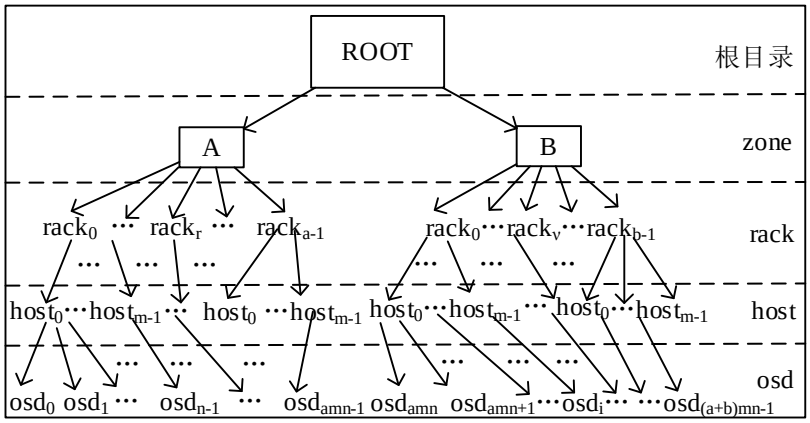


图 3.1　Ceph 集群目录树结构

现将运行区的机架分组，每组 K 个机架，则共有 $L = \left\lceil \frac{a}{K} \right\rceil$ 组，最后一组不足 K 个机架的，用 $(L \cdot K - a)$ 个虚拟机架补足，虚拟机架的计算能力参数为 0；用 $a_{lj}(1 \le l \le K, 1 \le j \le L)$ 表示集群中第 j 行、第 l 列台机架中的集群节点计算能力参数，则可将运行区抽象成如图 3.2 所示的逻辑结构，图中的黑色部分表示虚拟机架，即 $a_{LK} = 0$。

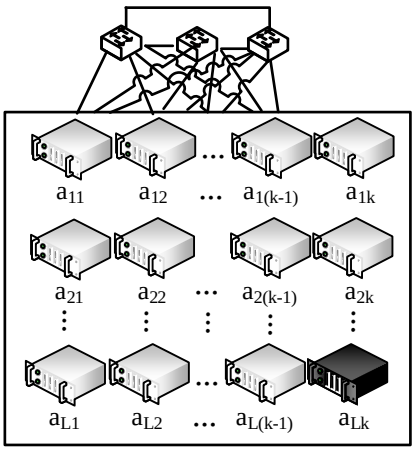


图 3.2　Ceph 集群逻辑抽象图

假定集群中有 ν 个应用程序待执行，每个应用程序的要求的完成时间 T_{SLA} 分别为 $T_t(0 \le t < L)$，则可得到下面的集群能耗计算模型：

$$\boldsymbol{A}=\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1K} \\ a_{21} & a_{22} & \ldots & a_{2K} \\ \ldots & \ldots & \ldots & \ldots \\ a_{L1} & a_{L2} & \ldots & a_{LK} \end{pmatrix};\quad \boldsymbol{X}=\left(x_1,\ x_2,\ \ldots\ ,x_K\right)^T;$$

$$\boldsymbol{B}=\left(b_1,b_2,\cdots b_L\right)^{\mathrm{T}};\ b_t=\begin{cases} M*\dfrac{1}{T_t} & 1\le t\le \nu \\ 0 & \nu < t \le L \end{cases}.$$

能耗优化模型的基本形式如下：

$$P_{\min}=\mathrm{Q}\times\left(\sum_{j=1}^{L}\sum_{i=1}^{K}\left|x_{ij}^{n*}x_{ij}^{(n-1)*}\text{-}1\right|+\varphi_n\right)+\eta \qquad (3\text{-}1)$$

$$st\begin{cases} \boldsymbol{AX}+\boldsymbol{W}=\boldsymbol{B}; & (3\text{-}2) \\ \boldsymbol{A}>0; \\ \boldsymbol{B}>0; \\ -1<\boldsymbol{W}\le 0; \\ 0\le\varphi_n\le K\text{且}.\varphi_n\in Z \\ x_i\in\{0,1\}. \end{cases}$$

上述节能模型中 $\boldsymbol{X}$ 为决策变量、$P_{\min}$ 为目标函数、st 为约束条件。$\boldsymbol{A}$ 为节点的计算能力参数矩阵，$\boldsymbol{B}$ 为用户的 T_t 矩阵，M 为一个常数系数，可以根据 T_t 来进行调整，$\boldsymbol{W}$ 为 T_t 调整参数，$P_{\min}$ 指 Ceph 集群的最小能耗。另外，除了能耗模

型中的一些参数，在模型求解过程中用到的其它参数有：$\boldsymbol{A}_n^*$为部署第n个应用后集群运行区状态矩阵，x_{ij}^{n*}为$\boldsymbol{A}_n^*$的元素，φ_n表示待机区开启的节点数，η为切换集群状态时产生的额外能耗，具体的参数含义和求解过程，将在本文的 3.3 小节中介绍。

关闭节点的数量和集群性能是一个很难平衡的问题，关闭的节点太多，虽然节约了集群能耗，却会使得集群性能急剧下降；关闭的节点太少，则达不到节约能耗和合理利用集群资源的目的。因此将应用完成限制时间T_t作为能耗优化模型的参数是很有必要的，这样能在合理关闭集群节点的同时，保证集群性能。

由于引入了T_t调整参数$\boldsymbol{W}$，则保证了（2）式必定有解$\boldsymbol{X}$，且当$\boldsymbol{W}$取得最小值时，取得最优解$\boldsymbol{X}$，而将向量$\boldsymbol{X}$与集群计算能力参数矩阵$\boldsymbol{A}$的每个行向量相与，便可得到满足用户 SLA 需求时的集群运行区部署矩阵$\boldsymbol{A}^{*'}$，在该矩阵中，元素$x_{ij}^{n*'}(1\le i\le K,1\le j\le L,i\in Z^+,j\in Z^+)$的值若为 1 则表示必须让运行区对应机架中的某一个节点处于运行态，若为 0 则表示无需对相应机架中的集群节点做任何操作。设$\boldsymbol{A}_0^{*'}=\boldsymbol{0}$，则由$\boldsymbol{A}_n^{*'}$和$\boldsymbol{A}_{n-1}^{*'}$可计算出部署第$n$个应用后集群中至少需另外开启的节点数目$\triangle \mathrm{T}_n=\left|x_{ij}^{n*'}x_{ij}^{(n-1)*'}\text{-}1\right|$，则集群在部署第个$n$应用后，集群中至少需要开启的节点数目一共为$\mathrm{T}_{\min}^{'}=\sum_{j=1}^{L}\sum_{i=1}^{K}\left|x_{ij}^{n*'}x_{ij}^{(n-1)*'}\text{-}1\right|$。但如果这样处理的话，仅仅能够在集群不出故障时，保证应用的完成时间，一旦集群出现故障，集群的可靠性将急剧下降，应用自然不能按时完成，并不能保证用户数据的可用性，这显然是不合理的。由此可以看出，在研究如何降低集群的能耗时，除了要保证集群的计算能力以保证应用在规定时间内完成之外，还需要考虑集群的可靠性和用户数据的可用性。因此，在本小节的基础上，本文第 3.3 小节提出了一种即考虑降低集群能耗又考虑集群性能的 Ceph 集群数据副本存储策略，

3.3 Ceph 集群节能数据副本存储策略

Ceph 集群默认采用三个数据副本存储策略，CRUSH 算法将用户数据平均存放在 Ceph 集群的 OSD 中，由于集群中大多数节点都处于低负载状态，显然这种平均存储策略在提高集群资源利用率和节约集群能耗方面都存在着很大的不

足。良好的数据副本存储策略不仅能保证集群性能，同时还能提高集群资源利用率，从而达到节约集群能耗的目的。本文第 3.2 小节中的 T'_{min} 保证了满足用户 SLA 所需的集群计算能力，本节在此基础之上，提出了一种针对 Ceph 集群的节能数据副本存储策略。

本文在 2.2 小节中（Ceph 客户端写数据过程）详细介绍了 Ceph 集群客户端写数据过程，因此这里不再赘述。假定每个 PG 都会映射到 Ceph 集群的 R 个 OSD 中，即集群采取 $R(R \in Z^+)$ 个数据副本存储策略，其中运行区存储 $\left\lceil \frac{R}{2} \right\rceil$ 个数据副本，待机区存放 $R-\left\lceil \frac{R}{2} \right\rceil$ 个数据副本，下面先讨论运行区的数据存储方法。

针对用户数据的任意一个 PG，设主 OSD（由于待机区的节点处于待机状态，所以用户数据必定先写运行区的节点，因此主 OSD 必定处于运行区）所在节点为 $host_c$， $host_c$ 所在的主机架为 $rack_c$，数据恢复域（Data Recovery Field，DRF）为 $\delta(\delta \geq 1 且 \delta \in Z)$，则运行区另一份存储在与 $rack_c$ 相邻的 λ 个机架上，即另一份数据副本的存储位置为 $\delta_c(t)=\{rack_t \mid \max(0,c-\delta) \leq t \leq \min(c+\delta,a-1)\}$，由于引入数据恢复域，因此减少了 Ceph 集群数据自动同步时节点之间传输数据的时间。待机区存放的 $R-\left\lceil \frac{R}{2} \right\rceil$ 个数据副本采用随机存放策略，当集群处于应用高峰期时，待机区节点需从睡眠状态切换到运行状态以满足集群性能需求。

图 3.3 表示集群数据副本数 $R=3$ 时，采用上述数据副本存储策略后集群中客户数据的存储状态。其中 $Region_A$ 为运行区，$Region_B$ 为待机区，数据副本数 $R=3$，数据恢复域 $\delta=2$，主 OSD（即图 3.3 中的 M_1（主））所在服务器（即图 3.3 中的 $A.rack_i.host_2$）设为主节点 $host_c$， $host_c$ 所在机架设为主机架（即图 3.3 中的 $A.rack_i$）$rack_c$ 与 $A.rack_i.host_2$ 用线连接的区域展示了用户数据的存储状态，无填充的 OSD 表示没有存储相应的用户数据。用灰色填充的 OSD 表示存储了数据，第二个数据副本采用随机存储策略，存放在数据恢复域内任意一台节点的 OSD 中（次 OSD），而第三个数据副本同样采用随机存储策略存储在待机区

中任意一个节点的 OSD 中。需要特别指出的是，当应用要求存储的数据副本数 $R>3$时，建议将$\left\lceil \frac{R}{2} \right\rceil$个数据副本存放在运行区，将剩下的$R$-$\left\lceil \frac{R}{2} \right\rceil$个数据副本存放在待机区，以保证集群的性能。若此时集群状态稳定后总是达不到用户的 SLA 需求（如果数据的读写数据速度太慢，即使 CPU 计算性能能跟上，也会降低整个集群的读写性能，从而达不到用户 SLA 需求），则应根据具体情况增大运行区和待机区的数据副本数量的比率，以满足用户 SLA 需求。

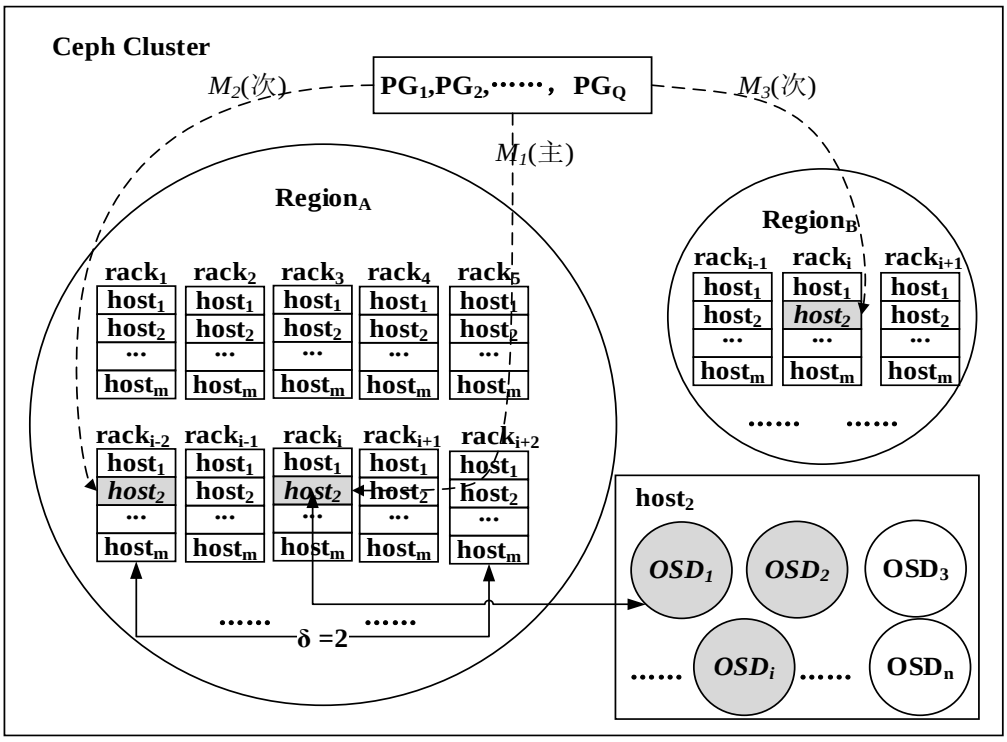


图 3.3　Ceph 集群节能数据副本放置策略

设采用上述数据副本放置策略后，集群的运行区状态图为$\boldsymbol{A}^{*}$。由以上数据副本存储策略结合本文 3.2 小节的能耗优化模型可知，运行区至少需要开启的节点数目为$\mathrm{T}_{\min}=\sum_{j=1}^{L}\sum_{i=1}^{K}\left|x_{ij}^{n*}x_{ij}^{(n-1)*}\text{-}1\right|$，假定待机区需额外开启的节点数目为$\varphi_n$，则集群中至少要开启的节点总数为$\mathrm{T}_{\min}=\sum_{j=1}^{L}\sum_{i=1}^{K}\left|x_{ij}^{n*}x_{ij}^{(n-1)*}\text{-}1\right|+\varphi_n$，每台服务器的平均能耗为$\mathrm{Q}$，开启待机区服务器所需的额外能耗为$\eta$，则集群总能耗可记为：$P_{\min}=\mathrm{Q}\times\left(\sum_{j=1}^{L}\sum_{i=1}^{K}\left|x_{ij}^{n*}x_{ij}^{(n-1)*}\text{-}1\right|+\varphi_n\right)+\eta$，而当集群足够大时，$\eta$可以忽略，故集群

的最小能耗为 $P_{\min} = \mathrm{Q} \times \left(\sum_{j=1}^{L} \sum_{i=1}^{K} \left| x_{ij}^{n*} x_{ij}^{(n-1)*} \text{-} 1 \right| + \varphi_n \right)$。

该数据副本放置策略核心伪代码如下所示：

输入：*每个机架中服务器的计算能力参数* a_{ij}，*应用完成时间* T_t，*数据副本数* R，*数据恢复域* δ。

输出：*集群运行区状态部署图* $\boldsymbol{A}^*$。

1）根据 a_{ij}，T_t *构造矩阵方程* $\boldsymbol{AX} = \boldsymbol{B}$；

2）if $\boldsymbol{AX} = \boldsymbol{B}$ *有解*

3）return $\boldsymbol{X}$；

4）else

5）设定 $\boldsymbol{W}$ *的值，求得方程* $\boldsymbol{AX} + \boldsymbol{W} = \boldsymbol{B}$ *的解；*

6）return $\boldsymbol{X}$；

7）将 $\boldsymbol{A}$ *的行向量与* $\boldsymbol{X}$ *相与，得到集群运行区部署图* $\boldsymbol{A}^{*'}$

8）根据 3.3 小节的数据副本存储策略，调整 $\boldsymbol{A}^{*'}$ *得到集群运行区部署状态图* $\boldsymbol{A}^*$；

9）return $\boldsymbol{A}^*$；

3.4 实验结果与分析

Shekhar Srikantaiah 教授针对数据中心服务器资源利用率和电量消耗问题做了深入研究，他在文献[58]中的研究结果表明：当服务器的磁盘利用率超过 50%，同时 CPU 利用率超过 70%时，其消耗的电量将随着磁盘使用率的提高而急剧上升（是一个三维空间的 U 形函数），这必然会导致磁盘的单位 I/O 能耗值（单位 I/O能耗值 $= \dfrac{\text{磁盘总能耗}}{\text{磁盘总I/O次数}}$）增加，使得集群能效下降。而且当 CPU 使用率过高，服务性能会由于资源竞争问题而下降，这必然会使得单个服务产生的能耗增多，当服务器的磁盘使用率控制在 50%时，单个服务随 CPU 变化所产生的能耗最小，因此为了降低不同服务对集群能耗的影响，本实验基于本文 3.3 小节中的最小开启节点数目 $T_{\min}$ 以及本文提出的数据副本放置策略，只将服务器的

一半磁盘用作 OSD，将节点的 CPU 平均使用率控制在 70%左右，实际上，当达到这个点时，服务器的资源使用率已经很高了。当节点的这两个参数指标超出这个范围时，触发报警且将对应该服务器的计算能力置为 0，那么由本文 3.2 小节的能耗优化模型可知，将不会有新的数据存储到该节点上。另外，由于 Ceph 集群自我平衡的功能会严重影响实验结果，所以在该实验中，并没有使用 Ceph 集群的数据自动平衡功能。

3.4.1 实验环境

为了验证和评估该数据副本存储策略的有效性，在云平台中进行了实验。其中 Ceph 集群中一共用到了 3 台华为交换机(S3352P-SI 48 口)和 20 台戴尔 R710 服务器，每台服务器采用的硬件配置为：8 核 CPU、64G 内存、3×1T 容量磁盘做了 RAID0、3 张千兆网卡；采用 Centos 7（64 位）操作系统，使用的 Ceph 版本为 Jewel（10.2.5）。同时，为了模拟异构集群，在其中一些服务器上运行了不同配置的虚拟机。20 台服务器中，其中待机区 8 台服务器，放在$b=4$个机架中，运行区 12 台服务器，放在$b=6$个不同的机架上，每个机架中存放了$m=2$台服务器。之所以将服务器存放在不同的机架中，是因为可以方便的从 IDC 室的 UPS 设备中读取服务器的耗电量。实验数据大小为 40G。另外用云平台监控系统对每台服务器进了监控，在监控页面可以采集到服务器和集群的 CPU 平均使用率、平均负载和 IOPS 等。整个 Ceph 集群的逻辑部署状态如图 3.4 所示。

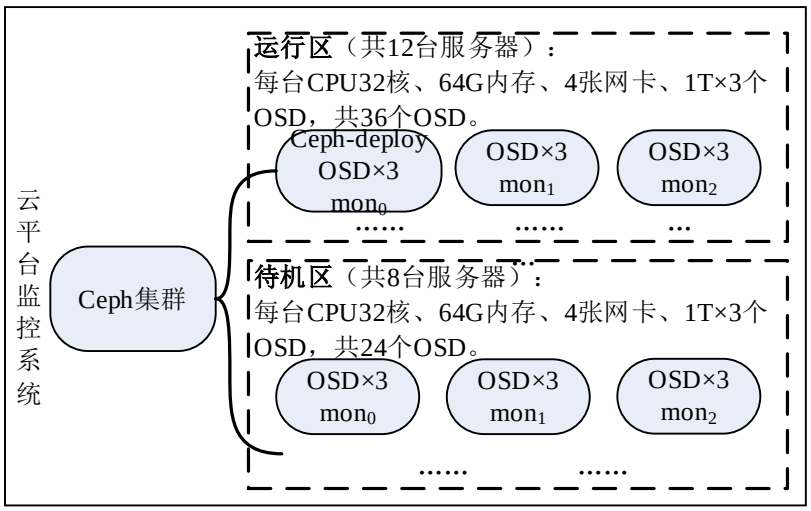


图 3.4　Ceph 集群逻辑部署图

3.4.2 实验过程

OSD 是 Ceph 集群中真正存储用户数据的组件，也是产生能耗最主要的部分。

在本文 3.2 小节提出的能耗模型的基础上，本文 3.3 小节提出了一种新的 Ceph 集群节能数据副本放置策略，下面验证该策略的有效性。在该实验中，数据副本数 $R=3$，数据恢复域为 $\delta=2$，集群隔离域为 host，每 10s 采集一次 CPU 使用率，并求其均值，得到各节点的 CPU 平均使用率，具体实验步骤如下：

Step1：输入用户应用完成限制时间 T_t，得到 $\boldsymbol{B}=1000/T_t$,其中 $M=1000$；

Step2：从监控系统中采集数据，并得到每个集群的 CPU 平均使用率，构造计算能力参数矩阵 $\boldsymbol{A}$；

Step3：根据本文 3.2 小节的能耗模型计算出 $\boldsymbol{W}$ 和 $\boldsymbol{X}$，将 $\boldsymbol{A}$ 的行向量与向量 $\boldsymbol{X}$ 相与，得到集群运行区部署矩阵 $\boldsymbol{A}^{*\prime}$；

Step4：由本文 3.3 小节提出的 Ceph 集群节能数据副本存储策略，得到集群运行区状态图 $\boldsymbol{A}^{*}$；

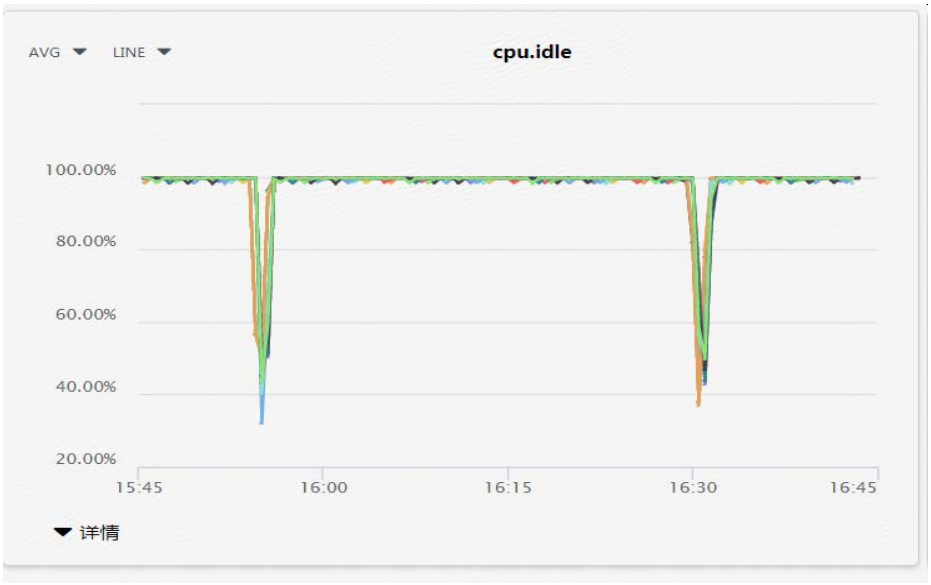
Step5：反编译 CrushMap，并根据 $\boldsymbol{A}^{*}$ 修改其内容，然后对其进行编译并将其应用于 Ceph 集群中；

Step6：根据能耗优化模型中的（3-1）式计算出当前集群总能耗 p；

Step7：用 FIO 压力测试工具，模拟不同的集群负载，对集群的读写性能进行测试。

下面对本实验的具体实验过程进行详细介绍：

考察 Ceph 集群中个节点的磁盘和 CPU 使用情况，系统监控界面分别如图 3.5 所示：



（a） Ceph 集群 CPU 平均使用

（b） Ceph 集群磁盘平均使用率

图 3.5 初始状态时集群磁盘和 CPU 平均使用率

从图 3.5 可以看到，CPU 和磁盘的平均使用率是非常低的，将服务器的平均计算能力参数设为 1，则可得到 $\boldsymbol{A}_0=\begin{bmatrix}1&1&1&1\\1&1&0&0\end{bmatrix}$，其中 $Rack_6$ 和 $Rack_7$ 是虚拟机架。采集数据后，将 Ceph 集群中运行区和待机区的所有服务器都至于休眠状态。假定应用完成限制时间为 $T_t=500s$，则 $\boldsymbol{B}_1=(2,0)^{\mathrm{T}}$，根据能耗优化模型中的（3-2）式和实验步骤的 **Step3** 可得：$\boldsymbol{X}_1=(0,0,1,1)^{\mathrm{T}}$，$\boldsymbol{W}_1=\boldsymbol{0}$，$\boldsymbol{A}_1^{*'}=\begin{bmatrix}0&0&1&1\\0&0&0&0\end{bmatrix}$，则

根据得到的结算结果 $\boldsymbol{A}_1^{*'}$ 可知要分别将机架 $Rack_2$ 、机架 $Rack_3$ 中的一个集群节点从休眠状态切换到运行状态才能满足 $T_t = 500s$ 的用户需求，根据本文 3.3 小节提出的节能数据副本放置策略，即根据实验步骤中的 **Step4** 可选择 $Rack_3$ 为主机架，则 $c = 3$，由 $\delta = 2$ 、$a = 6$ 可得 $1 = \max(0, c-\delta) \le 2 \le \min(c+\delta, a-1) = 5$，因此可选择 $Rack_2$ 为附加机架，故集群运行区的状态图为 $\boldsymbol{A}_1^* = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$。根据状态图 $\boldsymbol{A}_1^*$，可将 $Rack_3 \,.\, host_0$ 和 $Rack_2 \,.\, host_0$ 两个节点和待机区的任意一个节点从睡眠状态切换到运行态，其中 $Rack_3 \,.\, host_0$ 为主 OSD 所在节点，$Rack_2 \,.\, host_0$ 和待机区的另一个节点为次 OSD 所在节点，数据完全写到 Ceph 集群后，将待待机区节点置于休眠状态。接下来由实验步骤中的 **Step5** 知，需修改 Ceph 集群反编译后的 CrushMap 配置文件，然后将修改后的 CrushMap 重新编译后，应用到 Ceph 集群中。这样，该用户的三个数据副本就分别存放到了 ClusterMap 中指定的 3 个 Ceph 集群节点中。修改后的 CrushMap 文件的内容如图 3.6 所示，部分没写出来的 OSD 的权重均为 0。

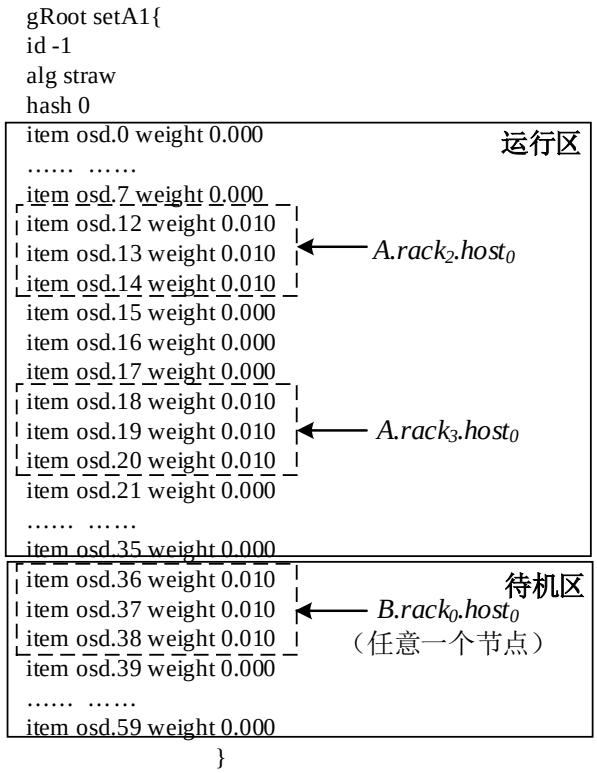


图 3.6　修改后的 Crush Map

图 3.6 中，osd.12~osd.14 对应集群运行区的节点 $A.Rack_2.host_0$，osd.18~osd.19 对应集群运行区节点 $A.Rack_3.host_0$，osd.36~osd.38 对应集群待机区节点 $B.Rack_0.host_0$（该节点是随机选择的）。将该 CrushMap 重新编译后应用于 Ceph 集群，即将本文提出的 Ceph 集群节能数据副本放置策略应用于 Ceph 集群中。现给出用户应用完成限制时间 $T_t=500s$ 时，不同用户数下各个机架中节点的平均 CPU 使用率统计情况，如图 3.7 所示。其它情况下的计算过程与上述过程相似，限于篇幅，这里不再赘述，现将其计算结果列于表 3.1 中。

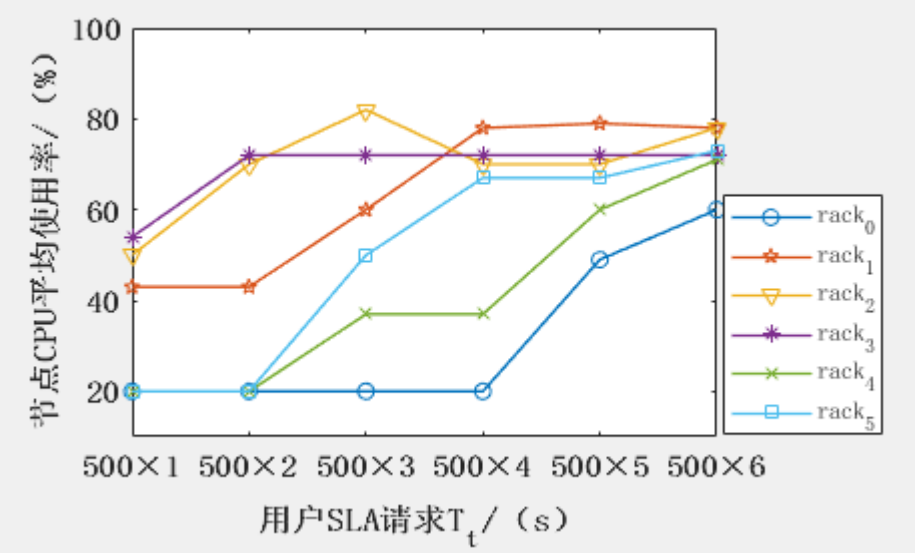


图 3.7　各机架中节点的 CPU 平均使用率

表 3.1 能耗模型中各个参数值

T_{SLA}（s） / 参数值	500	500×2	500×3	500×4	500×5	500×6
$\boldsymbol{A}$	$A_1=\begin{pmatrix}1&1&1&1\\1&1&0&0\end{pmatrix}$	$A_2=\begin{pmatrix}1&1&1&1\\1&1&0&0\end{pmatrix}$	$A_3=\begin{pmatrix}1&1&0&0\\1&1&0&0\end{pmatrix}$	$A_4=\begin{pmatrix}1&1&0&0\\1&1&0&0\end{pmatrix}$	$A_5=\begin{pmatrix}1&0&0&0\\1&0&0&0\end{pmatrix}$	$A_6=\begin{pmatrix}1&0&0&0\\1&0&0&0\end{pmatrix}$
$\boldsymbol{B}$	$\boldsymbol{B}_1=(2,0)^{\mathrm{T}}$	$\boldsymbol{B}_2=(1,0)^{\mathrm{T}}$	$\boldsymbol{B}_3=\left(\frac{2}{3},0\right)^{\mathrm{T}}$	$\boldsymbol{B}_4=\left(\frac{1}{2},0\right)^{\mathrm{T}}$	$\boldsymbol{B}_5=\left(\frac{2}{5},0\right)^{\mathrm{T}}$	$\boldsymbol{B}_6=\left(\frac{1}{3},0\right)^{\mathrm{T}}$
$\boldsymbol{W}$	$\boldsymbol{W}_1=\boldsymbol{0}$	$\boldsymbol{W}_2=\boldsymbol{0}$	$\boldsymbol{W}_3=\left(\frac{1}{3},\frac{2}{3}\right)$	$\boldsymbol{W}_4=\left(\frac{1}{2}\right)$	$\boldsymbol{W}_5=\left(\frac{3}{5},0\right)$	$\boldsymbol{W}_6=\left(\frac{2}{3},0\right)$

续表 3.1

T_{SLA}（s） / 参数值	500	500×2	500×3	500×4	500×5	500×6
$\boldsymbol{X}$	$\boldsymbol{X}_1=(0,0,1,1)^T$	$\boldsymbol{X}_2=(0,0,1,0)^T$	$\boldsymbol{X}_3=(0,1,0,0)^T$	$\boldsymbol{X}_4=(0,1,0,0)^T$	$\boldsymbol{X}_5=(1,0,0,0)^T$	$\boldsymbol{X}_6=(1,0,0,0)^T$
$\boldsymbol{A}^{*'}$	$A_1^{*'}=\begin{pmatrix}0&0&1&1\\0&0&0&0\end{pmatrix}$	$A_2^{*'}=\begin{pmatrix}0&0&1&0\\0&0&0&0\end{pmatrix}$	$A_3^{*'}=\begin{pmatrix}0&1&0&0\\0&1&0&0\end{pmatrix}$	$A_4^{*'}=\begin{pmatrix}0&1&0&0\\0&1&0&0\end{pmatrix}$	$A_5^{*'}=\begin{pmatrix}1&0&0&0\\0&0&0&0\end{pmatrix}$	$A_6^{*'}=\begin{pmatrix}1&0&0&0\\0&0&0&0\end{pmatrix}$
附加机架	$Rack_2$	$Rack_3$	$Rack_4$	$Rack_5$	$Rack_5$	$Rack_5$
$\boldsymbol{A}^{*}$	$A_1^{*}=\begin{pmatrix}0&0&1&1\\0&0&0&0\end{pmatrix}$	$A_2^{*}=\begin{pmatrix}0&0&1&0\\0&0&0&0\end{pmatrix}$	$A_3^{*}=\begin{pmatrix}0&1&0&0\\0&1&0&0\end{pmatrix}$	$A_4^{*}=\begin{pmatrix}0&1&0&0\\0&1&0&0\end{pmatrix}$	$A_5^{*}=\begin{pmatrix}1&0&0&0\\0&0&0&0\end{pmatrix}$	$A_6^{*}=\begin{pmatrix}1&0&0&0\\0&0&0&0\end{pmatrix}$

3.4.3 实验对比及分析

经一个多月的数据统计，一个集群节点忙时的平均功耗大约为 500W，闲时的平均功耗为 200W，可以计算出两种情况下对应 OSD 的平均能耗大致为 167W 和 67W。在优化前和优化后，分别向 Ceph 集群中存储 40G 数据，在不同 $T_t=500s$ 请求个数下，监控集群的运行情况。图 3.8 是优化前和优化后 Ceph 集群中处于运行状态的 OSD 个数对比图。

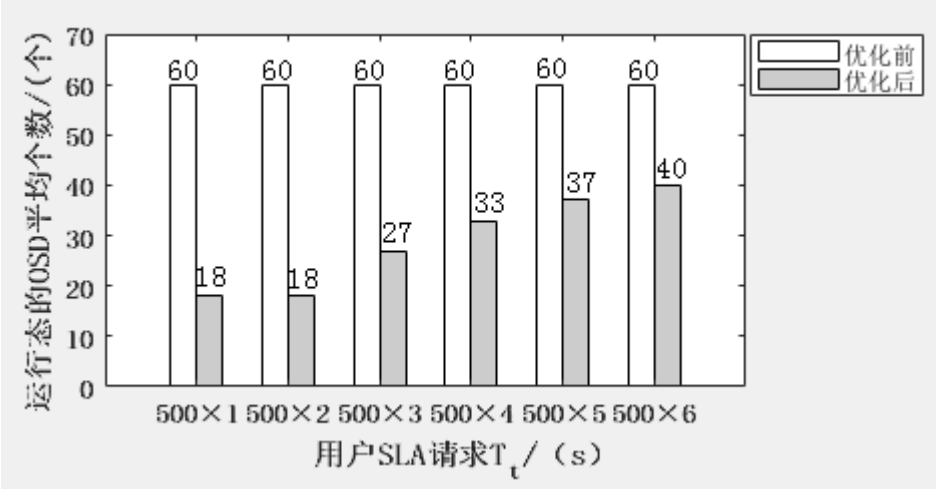


图 3.8　优化前后 Ceph 集群中运行的 OSD 对比图

从上图中可以看到，相比优化前，采用本文提出的节能数据副本存储策略后，Ceph 集群中运行的 OSD 数目减少了。

图 3.9 为 Ceph 集群能耗对比图，将从 UPS 读取到的数据称为实际数据，则图 3.9 中包括优化前的实际能耗数据、优化后的实际数据和集群优化后用公式

（3-1）计算出的能耗值三者之间的对比。从中能看出，当集群稳定时，用（3-1）式计算出的能耗值与优化后集群所产生的实际能耗值虽然在大小上存在着差异，但两者在上升趋势上基本保持一致，且相对误差（相对误差=$\left|\frac{\text{计算能耗}}{\text{实际能耗}}-1\right|\times100\%$）维持在 6%左右，由此可以得出：使用本文 3.2 节提出能耗优化模型计算出的集群能耗值，能一定程度上预测集群能耗值的走向，为更好的管理集群能耗奠定了基础。

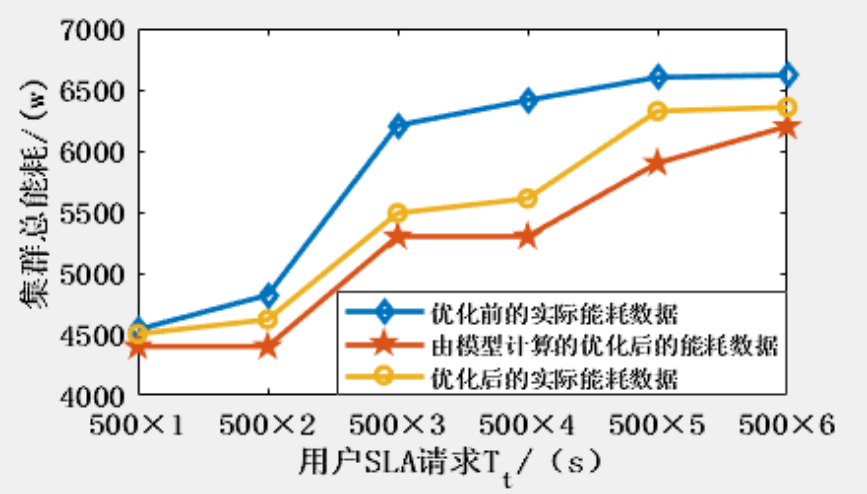


图 3.9　集群能耗对比图

此外，在图 3.9 中，随着 $T_t=500s$ 个数的增长，优化前的实际能耗值与优化后的实际能耗值之间的差值由小变大再变小，这是因为刚开始时，集群单个节点产生的能耗并不高，所以虽然关闭了集群中的部分节点，却并没有大量的降低 Ceph 集群能耗。此外，计算时采用了节点能耗的最大值来计算，且服务器中产生能耗的部件除了磁盘还有 CPU、内存等其他硬件，因此计算出的每个 OSD 平均能耗存在误差。而随着 T_t 数目的增长，集群节点负载逐渐增大，单个节点产生的能耗值也增大，尤其是在 $T_t=500\times3$ 至 $T_t=500\times5$ 时，节约的能耗值特别明显；当从 $T_t=500\times5$ 增大到 $T_t=500\times6$ 时，节约的能耗降低，是因为运行区的节点已经全部处于运行状态，且从图 3.8 中可以看出，此时待机区的部分节点也已经处于运行态，整个集群中关闭的服务器数目减少了，因此在这种情况下，为了保证集群性能，该能耗管理策略并不能大量的降低 Ceph 集群能耗。从上面的分析可知：节约的集群能耗值会随着集群中节点负载以及关闭的节点数目的变化而

有所波动，只有在集群负载适中时，该能耗管理策略能达到比较好的效果，此时降低的集群耗电量可达到 14.3%，而当集群中单个节点产生的能耗值越大时，节约的耗电量会高于该值。

集群响应时间是体现集群性能的一个非常重要的指标，它是满足用户 SLA 需求的前提条件。如果集群的响应时间过长，必定会使得应用完成时间大量增加，这就必然不能保证用户的 SLA 需求。但由于该能耗管理策略中将集群中部分节点置于待机状态，所以 Ceph 集群的能耗必定会有所下降。为了测试集群改进后的性能，用 FIO 压力测试工具对改进前后 Ceph 集群的顺序读响应时间和随机读响应时间进行了测试，测试数据大小为 40G，测试结果如图 3.10 所示。

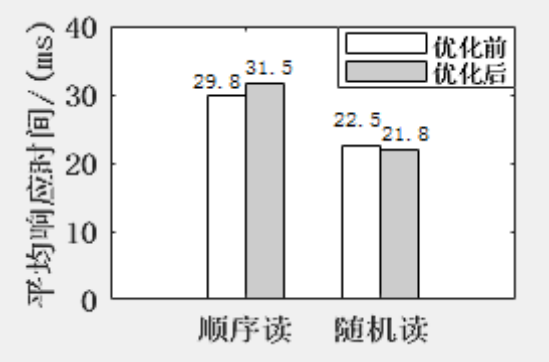


图 3.10　集群平均响应时间

由图 3.10 可知，集群优化后的顺序读取数据的响应时间大约增加了 5.7% ，而随机读数据的响应时间反而减少了 3.1%，这可能与数据中心的网络传输速度有关，但总的来说本文 3.3 小节提出的节能数据副本放置策略对集群的性能影响不大，并能在保证用户 SLA 需求的前提下，有效的降低 Ceph 集群能耗。

3.5 本章小结

本章针对数据中心巨大的能耗问题，对 Ceph 集群数据副本放置策略进行了改进。首先，对云平台的后端存储系统——Ceph 集群进行了深入分析，指出了 Ceph 集群的数据副本放置策略——CRUSH 算法，在节约数据中心能耗方面的不足，并建立了一个 Ceph 集群能耗优化模型；然后，基于该模型提出了一种 Ceph 集群节能数据副本放置策略；最后，在以 Ceph 集群为后端存储系统的云平台中做了实验。实验结果表明，建立的集群能耗优化模型，能够预测集群能耗的变化趋势，为能耗管理提供依据；而基于该模型提出的 Ceph 集群节能数据副本放置策略，在对集群服务质量影响不大的前提下，使得集群能耗降低了 14.3%。

第四章 基于改进 Ceph 集群的节能云资源调度策略研究

4.1 问题描述

Docker 技术是近年来兴起的一种操作系统级虚拟化技术，Docker 容器与传统虚拟机相比具有更高的系统资源利用率、更快速的启动时间、更轻松的迁移过程、更易维护和更易扩展等优点，这无疑给传统的虚拟化技术带来了挑战，但同时也必定为构建大型云平台提供了新思路。然而，Docker 容器虚拟化技术在云计算叱咤风云的同时，也存在着一些问题，其中 Docker 容器数据卷不能跨主机访问，便是应用容器在线迁移过程中需重点考虑的难点问题。

现有的基于 Docker 容器的云资源调度策略，虽然能一定程度上降低数据中心能耗，但他们要么只考虑容器部署时的能耗问题，要么只考虑容器应用迁移过程中的能耗问题，都没有同时考虑应用数据存储、应用容器部署和应用容器迁移对集群能耗造成的影响。

4.2 Docker 云资源节能调度模型

本文从降低 Ceph 集群能耗的角度出发，综合考虑应用数据存储、应用容器部署和应用容器迁移对集群能耗的影响，在保证服务性能的前提下，将第三章改进的 Ceph 集群和 Docker 容器虚拟化技术的有点相结合，病对其不足进行改进，提出了一种基于改进 Ceph 集群的节能云资源调度策略。该策略对 Ceph 集群资源进行了细粒度的划分，并实现了应用容器的在线迁移，基于 Dokcer 技术对 Ceph 集群资源进行了弹性调度，由此进一步降低集群能耗。

4.2.1 改进的 Ceph 集群与 Docker Swarm 集群的整合

本文在第三章对 Ceph 集群的数据副本放置策略进行了优化，降低了 Ceph 集群能耗。为了更合理的利用 Ceph 集群资源，进一步降低 Ceph 集群能耗，现将改进的 Ceph 集群与 Docker 容器虚拟化技术相结合。两者结合后的集群架构如图 4.1 所示。在该架构中，对集群建模的抽象过程与本文 3.1 节中介绍的建模过程基本一致，不同之处在于，将改进的 Ceph 集群与 Docker Swarm 集群相结合后，集群中的每台服务器既是 Ceph 集群中的 Node 节点，又是 Docker Swarm 集群中的 Worker 节点。由于 Ceph 集群节点中的每一个 OSD 都可以看做是一个 Linux 系统文件，因此可以将每一个 Node 节点的 OSD 与相应 Worker 节点中的

容器数据卷 Volume 互相绑定，那么当把应用数据存储在 Volume 中时，实际上是将应用数据存储在了 Ceph 集群的 OSD 中，而利用 Ceph 集群的多数据副本存储策略和同一活动集中数据同步的特点，Docker 应用容器在迁移时，只需要在相应 OSD 所在的节点上重新运行一个相同 Docker 应用容器即可，并不需要迁移该应用对应的数据，这样就实现了应用容器的在线迁移，解决了 Docker 数据卷不能跨主机访问的问题，而这一切在用户看来都是透明的。

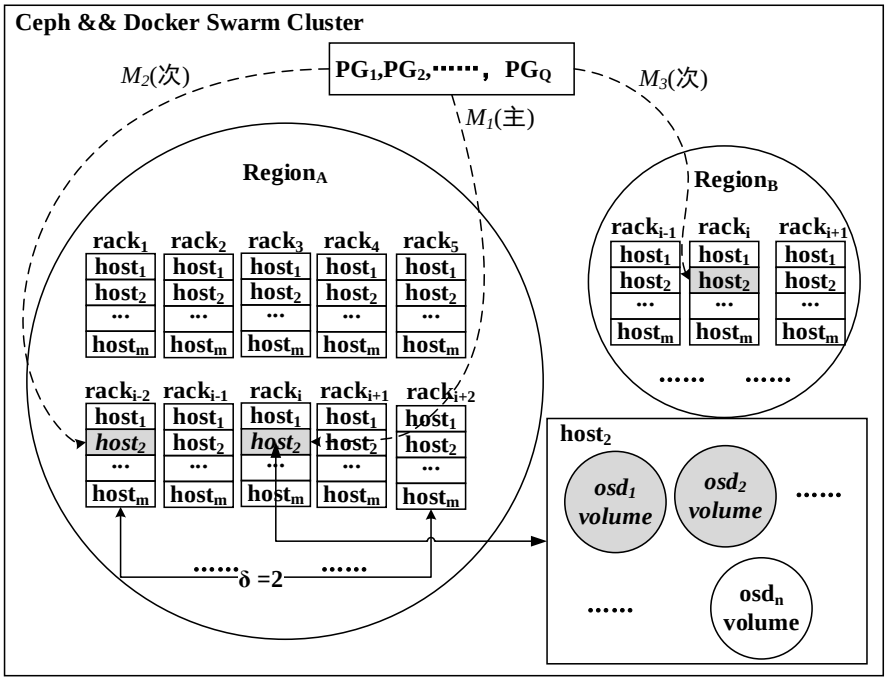


图 4.1　Docker Swarm 集群与 Ceph 集群架构图

4.2.2 基于改进 Ceph 集群的节能调度模型

在本文 4.2.1 小节的基础上提出了一种基于 Docker 技术的 Ceph 集群资源节能调度模型，该模型考虑到集群负载，以 Ceph 集群节点的 CPU、内存、网络的平均使用率作为主要参数，并引入了对应的资源调整参数、针对不同类型的应用对同一服务器的综合负载进行动态调整，同一个集群节点，对于不同类型的应用，可能处于三种不同的状态，这样便对集群资源进行了细粒度划分。该节能调度模型如下：

$$W_i^{*m}=\frac{1}{3}(W_c+W_m+W_n);\qquad (4\text{-}1)$$

$$\lambda+\alpha+\beta=1;\qquad (4\text{-}2)$$

$$W_i=\lambda W_c+\alpha W_m+\beta W_n;\qquad (4\text{-}3)$$

$$color=\begin{cases}green & 0\le W_i\le 20\%;\\ blue & 20\%<W_i\le 80\%;\\ red & 80\%<W_i\le 100\%.\end{cases}\qquad (4\text{-}4)$$

以上节能调度模型中，W_i^{*m}表示部署了第i个应用容器后，服务器$host_m$的综合负载；W_i表示某节点针对某个应用i的综合负载使用情况；W_c、W_m、W_n分别表示集群中任意一个节点的 CPU、内存和网络平均使用率；由于不同类型的应用对资源的需求不一样，同一个集群节点，对于 I/O 密集型应用可能已经处于高负载状态，但对于 CPU 密集型应用或网络密集型应用就可能处于低负载状态，因此引入资源调整参数λ，α，β，针对不同类型的应用i对W_i进行调整；$color\in\{green,blue,red\}$用来标记集群节点针对某个应用$i$的状态。$color=green$表示该节点处于空闲状态；$color=blue$表示该节点处于低负载状态；$color=red$表示该节点处于高负载状态。

4.3 调度策略

4.3.1 应用容器部署算法

良好的资源调度策略，不仅能提高云计算资源利用率，还能降低集群能耗。本节基于本文 4.3 小节提出的节能资源调度模型，在同时考虑应用容器数据存储和集群节点负载的的情况下，提出了一种应用容器部署算法。设数据采集时间间隔为T，共采集D次，在云平台监控系统采集服务器的 CPU、内存和网络的平均使用率，其值分别为W_c、W_m和W_n，下面讲述部署应用i的过程。首先判断该应用属于 CPU 密集型、内存密集型、网络密集型三者中的哪种类型，根据应用类型设定W_c、W_m和W_n的资源调整参数值，即设定λ、α和β的值；然后根据（4-2）式计算出λ、α和β的值，将这些值带入（4-3）式，可得到各个节点针对该应用i的综合负载W_i，根据得到的W_i由（4-4）式可将集群中节点的状态分

为 *green*、*blue* 和 *red* 三种状态。对于处于 $color = red$ 状态的服务器，表示对于该应用这类服务器负载已经很重了，因此不应将该应用部署在该类节点上。由于每个集群节点都是 Docker Swarm 集群中的 Worker 节点，所以对于不部署该应用的服务器，都应该将其置于 Drain 状态，这样 Docker Swarm 编排器就不会将新的应用部署在这类节点上；对于处于 $color = green$ 状态的服务器，表示该类服务器比较空闲，但从节约集群能耗和减少资源浪费的角度来看，应该尽量将这类服务器置于休眠状态，只有当所有处于运行状态节点的资源都充分利用时，才将该类节点作为部署该应用的目标节点，因此这类服务器并不是部署该应用的首选；对于处于 $color = blue$ 状态的服务器，表示该类服务器上运行着一些应用，且能够提供应用 i 所需的资源，因此应尽量先把该应用部署到这类节点上，只有当集群中不存在 $color = blue$ 的情况下，才将处于 $color = green$ 状态的一些服务器从休眠状态切换到运行状态，然后将其作为部署应用 i 的目标节点。将筛选出的要部署该应用的目标节点作为主节点，在部署应用之前，先将该应用的数据按照文章 3.3 小结提出的 Ceph 集群节能数据副本放置策略存放到集群中，然后再将该应用部署到对应主节点的 OSD 中，若是应用性能要求较高，可在主节点中再部署一个该应用，多个应用共享数据，也可将该应用部署在恢复域中的次 OSD 所在的节点上。应用容器部署算法流程图如图 4.2 所示。

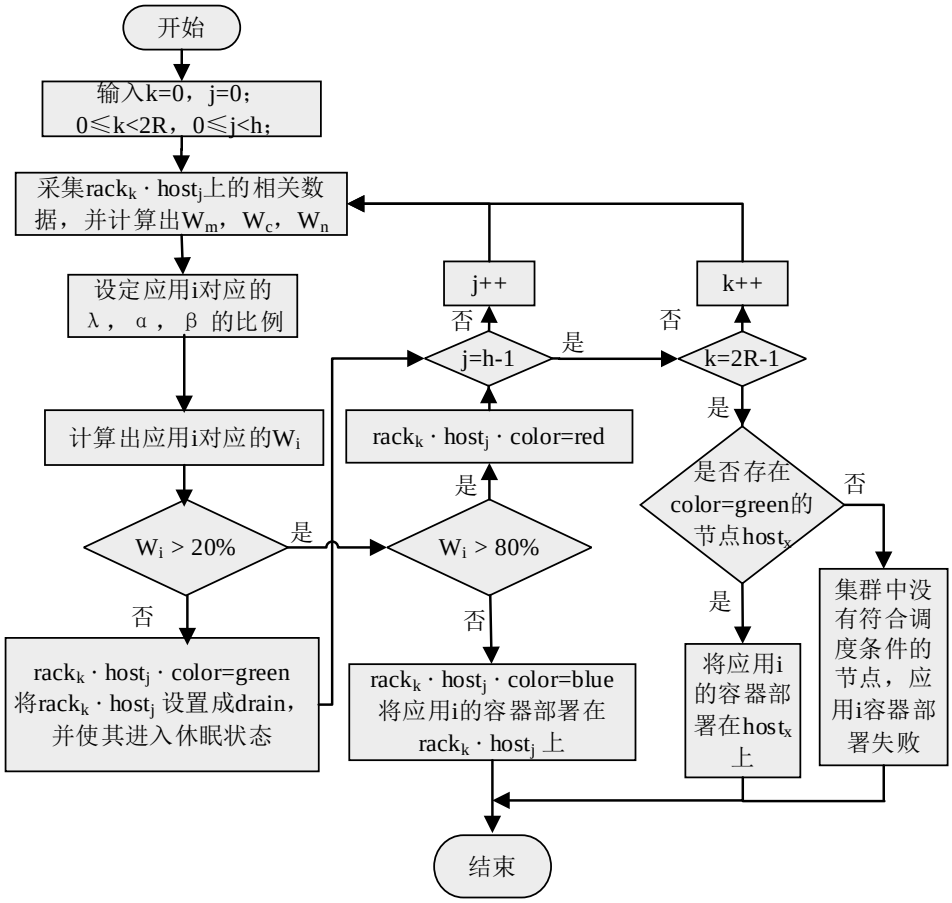


图 4.2　应用 i 容器部署流程图

4.3.2 应用容器迁移算法

为了保证应用的性能，对于处于 $color=red$ 状态的节点负载过重或一些异常节点，需要对在该类节点上运行的应用容器进行在线迁移，而对于有状态的应用，需要将其应用数据进行在线迁移，而 Docker 容器数据卷 Volume 不能跨主机访问的不足，使得应用在线迁移存在着困难。而把本文改进后的 Ceph 集群作为 Docker 容器的存储系统，将 Ceph 集群的 OSD 与 Swarm 集群中容器的数据卷（Volume）绑定，应用容器将数据存储在 Volume 中即是把数据存储在 OSD 中，由于 Ceph 集群中处于同一个活动集的主 OSD 和次 OSD 之间具有自动同步数据的特性，因此并不需要对应用数据进行迁移，只需在次 OSD 所在节点上也运行一个该应用容器，然后再将原节点上的应用容器删除，这样就可以实现应用的在线迁移。由于异常节点的情况比较繁杂，因此下面只讨论节点负载过重的情况。对于处于 $color=red$ 状态的节点，当在一段时间内，其平均资源利用率都大于阈值 $\boldsymbol{\varepsilon}=(\varepsilon_c,\varepsilon_m,\varepsilon_n)$（$\boldsymbol{\varepsilon}$ 为常量向量，常数 ε_c 、ε_m 和 ε_n 分别表示节点 CPU、内

存和网络的平均使用率允许达到的最大值，可由管理员根据云平台情况设定)时，会由于节点资源匮乏，影响应用性能，因此需要对该类节点上运行的部分应用容器进行在线迁移，以保证应用的性能。

假设节点$host_o(0\le o<m,o\in Z)$上运行着C^o个应用容器，现因该节点的综合负载过重，需要将其上运行的一些应用容器迁移至目标节点$host_d(0\le d<m,d\in Z且d\neq o)$。用$Container_r^o(0\le r<C^o,r\in Z)$表示$host_o$节点上编号为$r$的应用容器，$W_r^{*o}=\frac{1}{3}(W_c+W_m+W_n)$表示在$host_o$上部署了第$r$个应用容器$Container_r^o$后节点的平均综合负载，$\Delta W_r^{*o}=(W_r^{*o}-W_{r-1}^{*o})$表示部署$Container_r^o$后节点$host_o$的平均综合负载增长量，则$\overline{\Delta W_r^{*o}}=\frac{1}{C^o}*\sum_{r=1}^{C^o}\Delta P_r^0=\frac{\sum_{r=0}^{C^o-1}(W_r^{*o}-W_{r-1}^{*o})}{C^o}$表示节点$host_o$部署应用容器的平均综合负载增量；向量$\boldsymbol{W}=(W_c,W_m,W_n)$表示服务器当前状态下 CPU、内存和网络的平均负载；用向量$\boldsymbol{W}_T^c=(W_{Tc}^c,W_{Tm}^c,W_{Tn}^c)$表示$Container_T^m$的 CPU、内存和网络的平均使用率；用$host_x^A(0\le x<m且x\in Z)$表示其恢复域中存储了$Container_T^m$应用数据的另一台服务器；$\boldsymbol{W}^{A\cdot x}=(W_c,W_m,W_n)$表示$host_x^A$的资源使用情况；用$host_x^B(0\le x<m且x\in Z)$表示待机区中存储了$Container_T^m$应用数据的服务器；$\boldsymbol{W}^{B\cdot x}=(W_c,W_m,W_n)$表示$host_x^B$的资源使用情况；当迁移的应用容器本身使用的资源过少时，即使将这类应用容器从源节点迁移至目标节点，源节点的负载并不会降低到阈值$\boldsymbol{\varepsilon}$以下，这种情况下，要使得源节点的综合负载降到阈值以下，就必定会增加迁移应用容器的数量；当迁移的应用容器本身使用的资源过多时，迁移一个这类应用容器就能使得源节点的平均综合负载降低到阈值$\boldsymbol{\varepsilon}$以下，但是这类应用容器的迁移时间过长，而由于需要的资源较多，可能不能筛选出符合条件的目的主机；以上两种情况都过于极端，因此并不可取。设集合$\Delta W=\{\Delta W_1^o,\ \Delta W_2^o\cdots\cdots\Delta W_i^0\cdots\cdots\Delta W_k^o\}$, $k=C^o$, 将应用部署的平均增长量$\overline{\Delta W_r^{*o}}$作为参数，则本文要迁移的应用容器需满足条件（4-5）和条件（4-6）：

$$\left|\overline{\Delta W_r^{*o}}-\Delta W_i^o\right|\le\mu\ (0\le i<\mathrm{C}^m, i\in Z) \tag{4-5}$$

其中常数μ表示系统能接受的应用容器迁移代价的最大值，设集合S的初始值为$\varnothing$，若应用容器i不满足条件（4-5），则令$S=S\cup\Delta W_i^o(0\le i<\mathrm{C}^m, i\in Z)$且$\Delta W^*=\Delta W-S$，然后在集合$\Delta W^*$中进一步选择合适的应用容器进行迁移。令

$$\left|\overline{\Delta W_r^{*o}}-\Delta W_T^o\right|=\min\left\{\left|\overline{\Delta W_r^{*o}}-\Delta W_i^o\right|\right\}\ (0\le i<\mathrm{C}^m 且\Delta W_i^o\in\Delta W^*) \tag{4-6}$$

令t[$i++$]$=T$，则数组t[ ]记录了迁移代价由小到大且适合迁移的应用容器编号值；用$Container_T^o$表示节点$host_o$中筛选出的需要进行迁移的应用容器；

本文提出的应用容器迁移算法主要包括两个部分：第一部分是从源主机$host_o$中查找满足条件的待迁移应用容器$Container_T^o$；第二部分是Ceph集群中筛选出符合迁移条件的目的主机$host_d$。该算法具体如下：

输入：S、ΔW、μ、ε、i=0、j=0；

输出：$host_d$或$host_d$不存在；

Step1：由ΔW计算出综合负载的平均增长量$\overline{\Delta W_r^{*o}}$；由条件（4-5）得到$\Delta W^*$；由条件（4-6）得到数组t[$i++$]；

Step2：查看CrushMap，得到$host_x^A$和$host_x^B$，并采集数据分别得到服务器当前状态下CPU、内存和网络的平均负载$\boldsymbol{W}^A=\left(W_c^A,W_m^A,W_n^A\right)$、$\boldsymbol{W}^B=\left(W_c^B,W_m^B,W_n^B\right)$；

Step3：令$p=$t[$j++$]，则$Container_T^o=Container_p^o$，得到应用容器$Container_T^o$的CPU、内存和网络平均使用率$\boldsymbol{W}_T^c=\left(W_{Tc}^c,W_{Tm}^c,W_{Tn}^c\right)$；

Step4：若$W^A+W_p^c\le\varepsilon$，则输出$host_d=host_x^A$，否则执行**Step5**；

Step5：若$W^B+W_p^c\le\varepsilon$，则输出$host_d=host_x^B$，否则执行**Step6**；

Step6：输出$host_d$不存在；

Step7：若$host_d$存在，则将源节点$host_o$上的$Container_T^o$迁移至$host_d$。

Step8：当集群稳定后，若源节点$host_o$当前状态下的平均综合负载$W \le \varepsilon$，则运行在$host_o$上的其它容器不需继续迁移，应用容器迁移结束；若$W > \varepsilon$，则转执行 **Step3**；

选出需要迁移的应用容器和对应的目的主机$host_d$后，应用容器迁移过程为：用 commit 命令将应用容器$Container_T^o$打包成镜像，并上传至本地仓库；使节点$host_d$处于运行状态，并在$host_d$上用$Container_T^o$的镜像部署一个应用容器$Container^d$，同时确保$host_d$中的所有 OSD 都是$Container^d$的数据卷，等到应用运行稳定后杀死$host_o$中的$Container_T^o$容器进程，应用迁移成功。若没有得到$host_d$，则表名集群中没有符合迁移条件的目的主机，应用迁移失败，则应适当的增大阈值ε，或向云平台中增加一定数量的集群节点。

4.4 实验结果与分析

4. 4. 1 实验环境

为了验证该调度策略的可行性和有效性，在改进了数据副本放置策略的 Ceph 集群和 Docker 集群构成的云平台上进行了实验测试。实验中使用 20 台戴尔 R710 服务器。服务器的部署环境如下：运行区包括$a=6$个机架，每台机架中有$m=2$台服务器，共有$a\times m=12$台物理服务器。待机区包括$b=4$个机架，每台机架中有$m=2$台服务器，共有$b\times m=8$台物理服务器。服务器均采用 Centos7（64 位）操作系统、8 核处理器、3 张千兆网卡和 64G 内存，3 块容量为 1T 的磁盘做了 RAID0。所使用的 Ceph 版本为 Jewel（10.2.9），每个服务器中将 3 个磁盘对应的目录分别作为 Node 节点的 OSD，即$N=3$,数据副本数 R=3，使用的 Docker 版本为 17.06.0-ce，每个 Ceph 集群中 Node 节点的 OSD 都将绑定到 Docker 集群中 Worker 节点的数据卷 Volume。使用的 Kubernetes 版本为 1.8.0。为了更真实的模拟云平台环境，我们在某些服务器上运行了占用不同服务器计算资源的虚拟机，并用以下两个应用进行实验：

（1）CPU 密集型应用，后续成为应用①。该程序中用到了加减乘和根号等算数运算，为了便于统计应用完成时间，将程序执行的开始时间和结束时间都输出到终端，并将程序计算结果存入 Ceph 集群中，数据量为 8.5GB

（2）网络密集型应用，后续成为应用②。用 Nginx 部署一个 Java web 应用，该应用实现对磁盘中文档的搜索功能，实验用的数据文档（10000 篇，数据大小为 8.5GB）存放在集群节点的 OSD 中，并将搜索到的文档路径返回到屏幕上。

4.4.2 实验过程

在集群中部署第一个应用，由本文 3.4.2 的实验过程可知，此时集群中大多数节点都处于低负载状态，所以我们可先考虑节点 $A\cdot rack_0\cdot host_0$ 、$A\cdot rack_1\cdot host_0$ 和 $A\cdot rack_2\cdot host_0$ 的负载情况，将集群中的其它节点都置于休眠状态。从云平台监控系统采集实验相关数据，数据采集时间间隔为 10 分钟，一共采集 20 次。对采集的数据进行整理后，计算出内存、CPU 和网络的平均使用率，如表 4.1 所示。

表 4.1 处于运行态节点的资源平均使用率(%)

资源 / 节点	CPU（W_c）	内存（W_m）	网络（W_n）	平均综合负载（W_i）
$A\cdot rack_0\cdot host_0$	42.7	23.6	8.5	24.9
$A\cdot rack_1\cdot host_0$	40.6	30.7	7.8	26.4
$A.rack_2\cdot host_0$	19.8	12.4	7.6	13.3

表 4.1 中，因为在节点 $A\cdot rack_0\cdot host_0$ 和 $A\cdot rack_1\cdot host_0$ 上运行了占用系统资源的虚拟机，故其综合负载较高。由于节点 $A\cdot rack_0\cdot host_0$ 是 Docker Swarm 集群中的一个管理节点，需要定时和集群中的其它节点通信，故其网络平均使用率稍高。表中的平均综合负载由节能调度模型中的（4-1）式计算得出。

先选择 CPU 密集型应用，即应用①来进行实验。设 $\lambda:\alpha:\beta=3:2:1$，由调度优化模型中的（4-2）式可得 $\lambda=\frac{1}{2},\alpha=\frac{1}{3},\beta=\frac{1}{6}$，结合节能调度模型中的（4-3）式可得出当前状态下该三个节点对于应用①的综合负载值分别为 30.6%、31.8% 和 15.3%，对应的状态分别为 $blue$, $blue$, $green$ ，所以按照 4.3.1 小节提出的应用

容器部署算法，应该尽量将应用部署在节点 $A\cdot rack_0\cdot host_0$ 和节点 $A\cdot rack_1\cdot host_0$ 上，让节点 $A\cdot rack_2\cdot host_0$ 进入休眠状态。假定选择将应用部署在 $rack_1\cdot host_0$ 上，则按照本文提出的调度策略，应先将应用①所需的数据存储在节点 $A\cdot rack_1\cdot host_0$ 的 OSD 中。设数据恢复域 $\delta=2$，则按照本文 3.3 小结节提出的存储策略可将该应用数据 M_1、M_2、M_3 分别存放在 $A\cdot rack_0\cdot host_0$、$A\cdot rack_1\cdot host_0$ 和 $B\cdot rack_0\cdot host_0$ 三个节点中，其中 $A\cdot rack_1\cdot host_0$ 为主 OSD 所在节点，其余两个节点为次 OSD 所在节点。由于 Ceph 集群中 Ceph 客户端只负责写主 OSD，次 OSD 由主 OSD 负责写，主次 OSD 之间会通过 peering 过程同步数据，且具有自我修复数据的能力，因此在写主 OSD 时，应先确保三个节点都处于运行状态。修改后的 Crush Map 内容如图 4.3 所示，省略的 OSD 对应的权重均为 0.

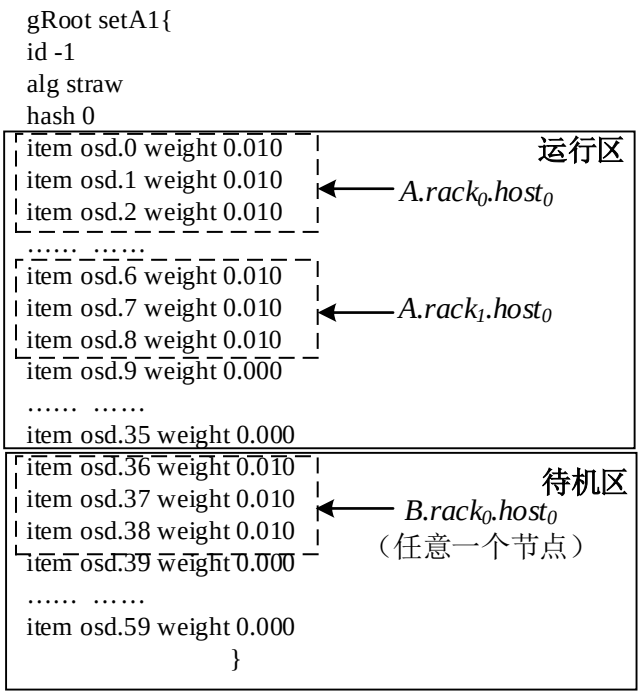


图 4.3　修改后的 Crush Map 内容

将应用数据存储好后，就在节点 $A\cdot rack_1\cdot host_0$ 中部署应用①，如果有需要也可同时在其余两个节点上部署应用①，否则可将待机区节点的状态切换到休眠状态。应用①执行过程中会把中间结果写入到主 OSD 中，然后按照之前的数据采集方法，经计算可得出 $A\cdot rack_1\cdot host_0$ 在部署了应用①后 CPU、内存和网络的平均利用率分别为 $W_c=80.3\%$、$W_m=73.6\%$、$W_n=38.0\%$，令 $host_o=A\cdot rack_1\cdot host_0$，则可计

算出 $W_1^{*o}=\frac{1}{3}\times(80.3\%+73.6\%+38.0\%)=64.0\%$，$\Delta W_1^{*o}=W_1^{*o}-W_0^{*o}=37.6\%$，设矩阵 $\boldsymbol{\varepsilon}=(\varepsilon_1,\varepsilon_2,\varepsilon_3)=(90\%,90\%,90\%)$，此时矩阵 $\boldsymbol{W}=(W_m,W_c,W_n)=(80.3\%,73.6\%,38.0\%)\le\boldsymbol{\varepsilon}$。接下来继续用本文 4.3.2 小结的提出的应用容器部署算法部署应用②。应用②是一个网络密集型应用，设 $\lambda:\alpha:\beta=1:1:2$，对于节点 $host_o=A\cdot rack_1\cdot host_0$，由（4-2）式和（4-3）式可得 $host_o$ 针对应用②的平均综合负载 $W_2=57.0\%\le80\%$，因此对于应用②节点 $host_o=A\cdot rack_1\cdot host_0$ 处于 *blue* 状态，同理可知节点 $A\cdot rack_0\cdot host_0$ 针对应用②也处于 *blue* 状态，因此可将应用②继续部署在 $host_o=A\cdot rack_1\cdot host_0$ 节点上。随后按照部署应用①的方法部署应用②。记录了进行 10 次文件搜索时的内存、CPU 和网络利用率，经计算得出 $A\cdot rack_1\cdot host_0$ 在当前状态下的平均资源利用率分别为 $W_c=90.7\%$、$W_m=80.2\%$、$W_n=82.4\%$，$W_2^{*o}=\frac{1}{3}\times(90.7\%+83.2\%+85.4\%)=86.4\%$，$\Delta W_2^{*o}=W_2^{*o}-W_1^{*o}=86.4\%-64.0\%=22.4\%$，$\overline{\Delta W_2^{*o}}=\frac{1}{2}\times(37.6\%+22.4\%)=30.0\%$。现在对运行在节点 $host_o=A\cdot rack_1\cdot host_0$ 上的应用进行迁移。此时节点 $host_o$ 平均资源利用率向量 $\boldsymbol{W}=(W_m,W_c,W_n)=(90.7\%,83.2\%,85.4\%)\le\boldsymbol{\varepsilon}$，用虚拟机调整内存平均使用率和网络平均使用率，使得 $W_m=90.2\%$、$W_n=90.4\%$，则有 $\boldsymbol{W}=(90.7\%,90.2\%,90.4\%)>\boldsymbol{\varepsilon}$，此时经满足迁移条件。

接下来对运行在节点 $host_o=A\cdot rack_1\cdot host_0$ 上的应用容器进行迁移。由于 $\mu=50\%>\left|\Delta W_1^{*o}-\overline{\Delta W_2^{*o}}\right|=34.0\%>\left|\Delta W_2^{*o}-\overline{\Delta W_2^{*o}}\right|=7.6\%$，故可设 $T=2$，即要对 $host_0$ 中的应用②进行迁移，则要迁移的容器为 $Container_2^o$，采集相关数据，并得到该容器中的平均资源使用率为 $\boldsymbol{W}_2^c=(10.4\%,9.6\%,47.4\%)$。根据本文 4.3.2 小节提出的应用容器迁移算法中的 **Step2** 得到向量 $\boldsymbol{W}^A=(42.8\%,23.6\%,26.9\%)$，且 $\boldsymbol{W}^A+\boldsymbol{W}_2^c\le\varepsilon$ 得出迁移的目的主机为 $host_d=A\cdot rack_0\cdot host_0$，即为集群运行区中第 1 个机架中的第

一台服务器。而该容器迁移完成后 $host_o = A \cdot rack_1 \cdot host_0$ 的负载已经小于阈值 ε，故不需要继续对运行在该节点上的应用容器迁移。

4.4.3 实验对比及分析

查询 10000 篇不同的文本，并记录搜索开始时间和结束时间，应用②在各种情况下的完成时间对比结果如表 4.2 所示。从表 4.2 中可以看出不同情况下的应用完成时间不同。在应用不发生迁移的情况下，在物理机中执行的完成时间最短，而在虚拟机中执行的时间最长，而从 $t_{容器} - t_{物理机} < t_{虚拟机} - t_{容器}$ 可以看出，应用②在容器中运行的性能与在物理机中运行的性能相差不大，这是因为容器直接以进程的形式运行在物理机中，比虚拟机通过客户机间接调用宿主机资源的速度要快。而在发生迁移的情况下，容器的迁移完成时间比不发生迁移的完成时间仅仅多了大概 30s，而虚拟机尽管使用了其快照功能，其应用迁移时的完成时间比应用不发生迁移时的完成时间多出了大约 90s，可见虚拟机迁移的代价比容器的迁移代价要大很多。另外，容器发生迁移时的应用完成时间与虚拟机不发生迁移的应用完成时间相近，表明本文提出的调度策略，在应用发生迁移时存在着明显的优势，且其迁移代价在能接受的范围之内。

表 4.2 应用②完成时间对比

运行环境		完成时间 t/（s）
容器	发生迁移	182
	不发生迁移	151
虚拟机	发生迁移	267
	不发生迁移	177
物理机	不发生迁移	140

为了进一步验证该调度策略的有效性，采用本文 4.3.1 小节提出的应用容器部署算法，在部署不同个数应用①容器的条件下，并将其与改进前的 Ceph 集群进行比较。在部署不同个数应用①的情况下，当集群处于稳定状态时，处于运行状态节点的数目对比结果和集群的负载状态对比结果如图 4.4 所示。

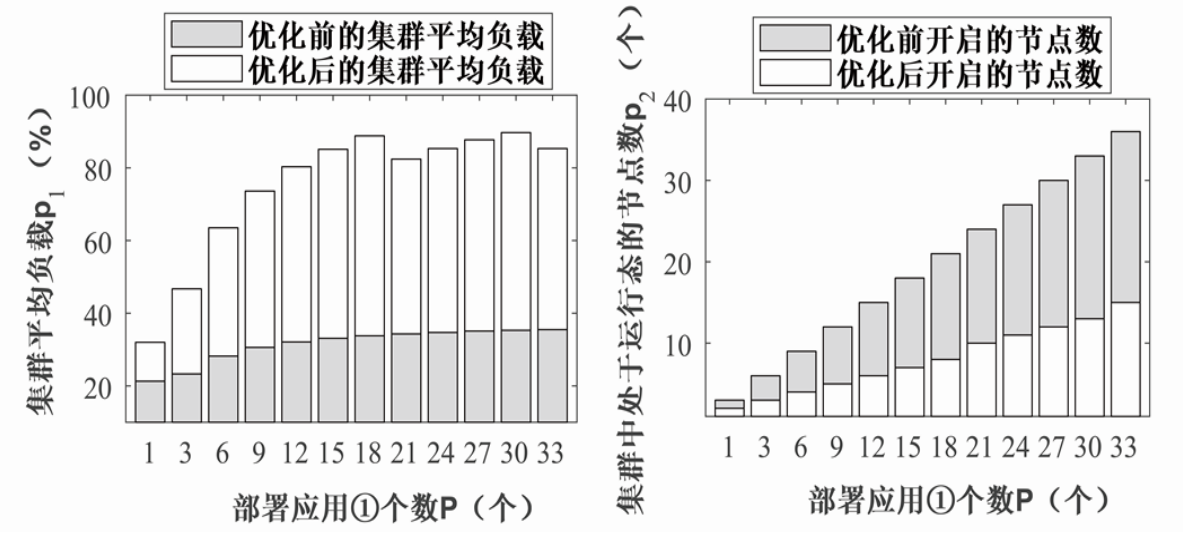


图 4.4　集群中处于运行态的节点数及集群负载

图 4.4 中，左图表示部署不同数目的应用①时，集群优化前后的平均负载情况；右图为与之对应的集群中处于运行状态节点的数目。从整体来看，与优化前的 Ceph 集群相比，优化后的 Ceph 集群中节点的负载较高，且集群中运行的节点数目明显减少了，说明本文提出的调度策略，能提高 Ceph 集群的资源使用率，而从集群中处于运行态的节点数减少的情况来看，该调度策略在一定程度上进一步减低了集群能耗。此外，对于采用了本文的调度策略的 Ceph 集群，当 $p=18$ 时，集群的平均负载最高，这是因为此时开启的节点数目较少，节点负载较高，单台服务器产生的能耗很高，故集群的平均能耗值最大。而随着 $p=18$ 值的增大，集群的平均负载又下降了，是因为集群中有一些处于休眠状态的节点切换到运行状态，而这类节点的负载较低，且在这类节点中，单台服务器产生的能耗较低，因此集群中的平均负载降低了。而当 $p=21$ 和 $p=33$ 时，与 p 取其它值的时候相比，集群中运行的节点数目上升趋势增大，这是因为在 $p=18$ 和 $p=30$ 集群中节点的平均负载已经很高了，需要大量的增加集群资源来保证应用的性能，而从图 4.4（左）也可以看出，此时集群的平均负载确实很高，这说明本文提出的调度策略能更充分的利用集群资源，达到了进一步降低集群能耗的目的。

为了进一步验证本文提出的调度策略的有效性，在不同集群负载的情况下，将改进后的 Docker Swarm 调度器（Swarm-opt）、原生的 Docker Swarm 容器编排器和 Kubernetes 容器编排器从集群负载、应用迁移时的完成时间和应用不迁移

时的完成时间三个方面进行了对比。需要指出的是，由于 Kubernetes 的容器复制控制器（Replication Controller）可以通过复制而产生相同的容器副本，而 Swarm 是从仓库中拉取镜像生成容器，两者的差异对应用的完成时间影响很大，为了避免这种差异对实验结果造成的影响，实验中并没有使用 Kubernetes 的容器复制控制器功能。此外，Kubernetes 中的最小调度单位是 Pod，而 Swarm 是以 Container 为最小调度单位的，因此实验中，每个 Pod 中只包含一个 Container,每个应用都是一个作业（job）。对比结果如表 4.3 所示。

表 4.3 不同容器编排工具对比

容器编排工具	集群负载 W（%）	完成时间 T（s）	
		不迁移	迁移
Swarm	80.3%	239	341
Swarm_opt	92.8%	345	409
Kubernetes	81.1%	247	359

从表 4.3 中可以看到，对于不同的集群负载，在应用不发生迁移时，应用的完成时间从小到大依次为$T(Swarm_opt) > T(Kubernetes) > T(Swarm)$，这是因为优化后的集群负载较高，应用容器的性能有所下降，但在应用发生迁移时的应用完成时间与应用不迁移时的应用完成时间增长量为：$\Delta T(Swarm_opt) < \Delta T(Swarm) < \Delta T(Kubernetes)$，从这里可以看出本文提出的调度算法在应用发生迁移时付出的代价最小，而由于 Kubernetes 中的容器启动较慢，在应用发生迁移时付出的代价最大，这表明本文提出的应用容器迁移算法在应用发生迁移时具有一定的优势。

4.5 本章小结

本章在第三章的基础之上，首先对 Docker Swarm 集群进行了研究，指出了 Docker 数据卷不能跨主机访问的不足，并将 Docker 容器虚拟化技术与改进后的 Ceph 集群相结合，基于 Ceph 集群建立了一个 Docker 云资源调度模型；然后基于该模型提出了一种云资源节能调度策略，该策略包括应用容器部署算法和应用容器迁移算法，从应用数据存储、应用容器部署和应用容器迁移三个方面考虑了如何提高云平台资源利用率以降低集群能耗问题；最后，用实验将该调度策略与 Docker Swarm 容器编排工具进行了对比，包括应用迁移和应用不迁移时的应用完成时间、优化前后 Ceph 集群中处于运行状态的节点数目、集群的平均负载情

况以及不同容器编排工具下应用发生迁移和应用不发生迁移时的完成时间。实验结果表明，本文提出的调度策略实现了云资源的弹性调度，并在应用容器的迁移中的性能体现了一定的优势，达到了合理利用 Ceph 集群资源和进一步减低 Ceph 集群能耗的目的。

第五章 总结与展望

5.1 总结

随着云计算与大数据技术的飞速发展，其应用领域也越来越广泛，从当初的科学计算到现在农业、工业、服务业，这使得人们的生活愈加方便、智能。但人们在习惯享受云计算与大数据带来的好处时，也看到了它带来的问题，其中数据中心能耗高就是一个亟待解决的问题，大量学者对其进行了深入研究，并取得了不错的成就。然而，随着新技术的出现与日益成熟，云计算平台所采用的技术也在发生着变化，这无疑给解决数据中心能耗问题带来了新的启发。

本文针对数据中心能耗高问题，在基于开源云平台管理项目 Openstack、分布式存储系统 Ceph、Docker 容器虚拟化技术、监控系统 Zabbix 以及其它云平台相关技术（负载均衡、高可用、备份技术等）构建的云平台上，对数据中心的能耗管理问题进行了深入研究，现将本文所做的工作总结如下：

（1）了解国内外数据中心能耗问题的研究背景、意义及研究现状。阅读了大量书籍和文献，熟悉国内外现有的与数据中心能耗相关的理论与技术，并对前人的工作进行归类总结，对比应用场景和云平台的差异，找到其中的优点和可改进之处。

（2）建立 Ceph 集群能耗优化模型。基于前人的工作，通过对现有云平台进行分析得知：云平台能耗过高绝大一部分原因是由于 Ceph 集群中节点的资源利用率过低。因此本文以云平台后端存储系统——Ceph 作为切入点，经研究发现 Ceph 集群数据副本放置策略（CRUSH 算法）在节约能耗方面存在着不足，并结合 Zabbix 监控系统和用户 SLA 需求，利用 Ceph 集群节点的计算能力参数，建立了一个能耗优化模型。

（3）提出 Ceph 集群节能数据副本放置策略。基于建立的能耗优化模型，对 Ceph 集群能耗副本放置策略——CRUSH 算法进行了改进。在该数据副本放置策略中将 Ceph 集群分为运行区和待机区，利用 Ceph 集群多数据副本和数据自我恢复等优点，同时考虑 Ceph 集群资源利用率和性能，对 Ceph 集群进行了改进。最后的实验结果表明，该节能数据副本放置策略能在对集群性能影响不大和保证用户 SLA 需求的情况下，使集群能耗降低 14.3%。

（4）建立了一个 Docker 云资源节能调度模型。针对 Docker 容器数据卷不

能跨主机访问给应用容器在线迁移带来的困难，将改进了数据副本放置策略的Ceph 集群与 Docker 容器虚拟化技术相结合，以节点 CPU、内存和网络的平均使用率作为主要参数，建立了一个基于 Ceph 集群的 Docker 云资源节能调度模型。该模型中引入了资源调整参数，使得同一个集群节点，针对不同类型的应用可能处于三种不同的负载状态，对集群资源进行了更细粒度的划分。

（5）提出了一种 Docker 云资源节能调度策略。基于建立的 Docker 云资源节能调度模型，提出了一种 Docker 应用容器调度策略。该调度策略包括应用容器部署算法和应用容器迁移算法，从应用数据存储、应用容器部署和应用容器迁移三个方面考虑了如何降低 Ceph 集群能耗问题。最后的实验结果表明，该调度策略实现了应用容器的在线迁移和资源的弹性调度，提高了 Ceph 集群资源利用率，达到了进一步降低 Ceph 集群能耗的目的。

5.2 展望

随着云计算与大数据技术的日益发展和成熟，云计算用户和产商也会逐渐增多，数据中心的数量和规模也会随之增大，这必将导致数据中心的用电量上升，那么如何合理利用数据中心资源以降低数据中心能耗，将会是未来需要继续研究的重点问题。本文提出的 Ceph 集群节能数据副本放置策略和基于改进 Ceph 集群的 Docker 云资源节能调度策略虽然在一定程度上节约了数据中心能耗，但有些问题还需进一步研究：

（1）不同的应用场景会选择不同的云平台技术构建不同的云平台，如何将现有的或将来出现的云平台新技术融入到不同的应用场景中构成具有特色的云平台，并针对不同的云平台采用不同的能耗管理策略以降低数据中心能耗，是未来要研究的问题；

（2）如何将硬件方法和软件方法相互结合，以便更好的降低数据中心能耗，也是未来值得考虑的一个问题。

以上提出的两个问题，将在以后的工作和学习中进一步思考和研究，望各位专家批评和指正。

致 谢

草长莺飞二月天，拂堤杨柳醉春烟。一片春意黯然里，研究生的三年时光已经悄悄地走了最后。2015 年的那个秋天，带着些许稚气、怀揣着憧憬的我，自信满满地走进了贵州大学这座校园，遇见了我的两位导师和那些跟我一样的同学们，真真正正的开始了我的研究生时光。

三年的时间不算长久但绝算不上短暂，在这段时光里，认识的每一个人、做的每一件事情，都足以影响着我的余生。非常感谢我的两位导师——吕晓丹老师和蒋朝惠老师，他们是那么的和蔼可亲和认真负责，感谢三年时光里两位导师在生活上的细心关照和学习上尊尊教导，是他们无私奉献地把无知的我一步一步引入科研的殿堂，传授给我严肃认真的办事态度。因为有两位导师，仿佛从一开始就注定我的硕士生涯是与众不同的。在这三年里，吕老师给我提供了特别好的学习环境，在这里我要特别感谢贵州翔明科技有限责任公司，在该公司我的专业实践能力有了很大的提升，数据中心的硬件资源也为做科学研究提供了非常好的条件，我不会为了做实验没有服务器、没有路由器而烦恼，只要我有想法，吕老师都特别的支持我，为了做实验，吕老师给了我们专门的服务器，因此我可以搭建集群来做实验，实现自己的想法，这种感觉特别棒。总记得吕老师对我们说过，研究生期间是人生中最无忧无虑的阶段，是最能做出研究成果的时候，他不想我们为了生活而影响了学习。另外，面对我的任性和生活中的烦恼，吕老师总是开导我、帮助我，真的非常感谢有这么一位像父亲一样的导师，感觉自己特别幸运。

此外，我还要感谢我的另外一位导师——蒋朝惠老师。蒋老师是学术上的专家，他总是能在学术上给我们非常好建议，很多事情蒋老师说不行，那就真的不行，以至于蒋老师特别辛苦，很多个周末他都来实验室帮我们改毕业论文相关的东西，经常有人跟蒋老师约时间谈事情，但他总是把我们的学习摆在很重要的位置，专门抽出时间来看我们的毕业论文。另外，蒋老师和吕老师都特别在意我们的人生安全问题，每次学校放假了，两位老师都会召集我们开会，一方面是总结和规划我们的学习，督促我们进步；另一方面会提醒我们注意人生安全。在这里再次感谢蒋老师，他永远值得让人尊重。

接下来我要感谢我的室友和实验室的小伙伴们，是你们陪伴我走过了研究生中的大部分时光，我们总是在一起诉说生活中的烦恼，在一起讨论和解决学习上

的困难，在最开心的时候放声的大笑，就这样我们相互促进，共同成长。我们就要毕业了，要各走一方，以后也许再也不会相见，既然怀念是在所难免的，那就让我们好好珍惜在一起的最后这段时光吧。

最后，我要感谢我的父母和我自己。感谢我的父母一直以来的支持和鼓励。在求学的这条道路上，你们为我付出了太多太多，谢谢你们，你们是我最坚强的后盾和最真实的依靠。同时我也要感谢我自己，没有辜负这么多年的时光，认认真真的走着人生的每一步，找到了满意的工作。越努力越幸运，希望未来我会遇见更好的自己。

参考文献

[1] 吕天文.2013 年数据中心能效现状深度分析[J].电源世界，2013(6)：7.

[2] 中国 IDC 圈未来五年数据中心能耗将翻一番[EB/OL].(2012-03-29)[2017-11-21]. http://tech.idcquan.com/34910.shtml.

[3] 新华网.三部门联合制定国家绿色数据中心试点工作方案[EB/OL].(2015-03-24)[2017-11-21].http://www.xinhuanet.com/info/2015-03/24/c_134091432.htm.

[4] 中国人民共和国工业和信息化部.三部门关于公布国家绿色数据中心试点单位的通知[EB/OL].(2015-12-24)[2018-02-10].http://www.miit.gov.cn/newweb/n1146290/n4388791/c4552917/content.html.

[5] 金科，阮新波.绿色数据中心供电系统[M].北京：科学出版社，2017，30-31.

[6] 邓维，刘方明，金海等.云计算数据中心的新能源应用：研究现状与趋势[J].计算机学报，2013，36(3)：582-598.

[7] 李翔，姜晓红，吴朝晖等.绿色数据中心的热量管理方法研究[J].计算机学报，2015，38(10)：1976-1996.

[8] 叶冉，李超，梁晓峣.面向绿色数据中心的储能系统：体系结构和管理方法[J].计算机研究与发展，2016，53(2)：326-340.

[9] Fan X, Weber W D, Barroso L A. Power provisioning for a warehouse-sized computer [C]. ACM SIGARCH Computer Architecture News. ACM, 2007, 35(2): 13-23.

[10] Qiong Cai,José González,Grigorios Magklis,et al .Thread shuffling:combining DVFS and teread migration to reduce energy consumption for multi-cores systems [C]. Proceedings of IEEE 2010 International Symposium on Low Power Electronics and Design (ISLPED), 2011: 379-384.

[11] Kaushik R T, Bhandarkar M. GreenHDFS: towards an en-ergy-conserving, storage-efficient, hybrid Hadoop compute cluster [C]. International Conference on Power Aware Computing and Systems. USENIX Association, 2010: 1-9.

[12] 黄庆佳.能耗成本感知的云数据中心资源调度机制研究[D].北京：北京邮电大学，2014.

[13] 郝亮.面向能耗优化的云计算资源调度算法研究[D].哈尔滨：哈尔滨工业大学，2015.

[14] 张海红.云环境下基于改进的蚁群算法任务调度策略的研究[D].沈阳：东北大学，2013.

[15] Xiang Y, Liu Z, Zhang G. Cloud data centers energy-saving scheduling algorithm based on CPU frequency scaling [C]. International Conference on Communication Technology, ICCT, 2014: 1027-1037.

[16] Kumar R, Sahoo G, Yadav V, et al. Minimizing the Energy Consumption of Cloud Computing Data Centers Using Queueing Theory [M]. Advances in Computational Intelligence. Springer, Singapore, 2017: 201-210.

[17] 刘茜妮.基于免疫蚁群算法的云任务节能调度策略的研究[D].哈尔滨：哈尔滨工业大学，2017.

[18] Duy T V T, Sato Y, Inoguchi Y. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing [C]. Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on. IEEE, 2010: 1-8.

[19] 陈辉.基于虚拟化环境的数据中心节能管理技术研究[D].北京：北京邮电大学，2012.

[20] Antal M, Burnete A, Pop C, et al. Self-adaptive task scheduler for dynamic allocation in energy efficient data centers [C]. Intelligent Computer Communication and Processing (ICCP), 2017 13th IEEE International Conference on. IEEE, 2017: 535-541.

[21] 侯伟.云计算中基于遗传算法的能效管理研究[D].武汉：武汉理工大学，2013.

[22] 张艳璐.一种基于遗传算法的低能耗云计算数据中心资源调度策略[D].杭州：杭州电子科技大学，2015.

[23] Fu D, Xiong Y, Lu C, et al. A Task Scheduling Method for Energy-Efficient Cloud Video Surveillance System Using A Time-Clustering-Based Genetic Algorithm [C]. Parallel and Distributed Systems (ICPADS), 2016 IEEE 22nd International Conference on. IEEE, 2016: 661-668.

[24] Peng Y, Kang D K, Al-Hazemi F, et al. Energy and QoS aware resource allocation for heterogeneous sustainable cloud datacenters [J]. Optical Switching and Networking, 2017, 23: 225-240.

[25] Kang D K, Choi G B, Kim S H, et al. Workload-aware resource management for energy efficient heterogeneous Docker containers [C]. Region 10 Conference (TENCON), 2016 IEEE. IEEE, 2016: 2428-2431.

[26] Meng Y, Rao R, Zhang X, et al. CRUPA: A container resource utilization prediction algorithm for auto-scaling based on time series analysis [C]. Progress in Informatics and

Computing (PIC), 2016 International Conference on. IEEE, 2016: 468-472.

[27] Guan X, Wan X, Choi B Y, et al. Application Oriented Dynamic Resource Allocation for Data Centers Using Docker Containers [J]. IEEE Communications Letters, 2017, 21(3): 504-507.

[28] 何松林.基于 Docker 的资源预调度策略构建弹性集群的研究[D].浙江：浙江理工大学，2017：36-60.

[29] David Meisner Brian T.Gold Thomas F. Wenisch. PowerNap: eliminating server idle power [J]. ACM SIGARCH Computer Architecture News, 2009, 37(1): 205-216.

[30] Íñigo Goiri, Kien Le, Md. E. Haque, et al. GreenSlot: Scheduling energy consumption in green datacenters [C]. Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2010, 20, SC '11.

[31] Indigo Goiri, Kien Le, Thu D. Nguyen, et al. GreenHadoop: Leveraging Green Energy in Data-Processing Frameworks [C]. Proceedings of 7th AVM european conference on Computer Syetems, 2012: 57-70.

[32] 柯尊旺，于炯，廖彬.适应异构集群的 Mesos 多资源调度 DRF 增强算[J].计算机应用，2016，36（5）：1216-1221.

[33] 林伟伟，吴文泰.面向云计算环境的能耗测量和管理方法[J].软件学报，2016，27(4)：1026-1041.

[34] Xiaofeng H, Pengtao S, Weichao T, et al. Green Hierarchical Management for Distributed Datacenter Containers [J]. Journal of Computer Research and Development, 2016,7:007.

[35] Aksanli B. Datacenter Peak Power Management with Energy Storage Devices [J]. IEEE Internet Computing, 2017.

[36] Son A Y, Huh E N. A Study on Resource Scaling Scheme for Energy Efficiency in Cloud Datacenter [M]. Advances in Computer Science and Ubiquitous Computing. Springer, Singapore, 2018: 1066-1071.

[37] 百度百科.云计算[EB/OL].[2017-12-15]. https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97#reference-[4]-1316082-wrap.

[38] 周品.Hadoop 云计算实战[M].北京：清华大学出版社，2012：12-15.

[39] Mahajan K, Makroo A, Dahiya D．Round Robin with Server Affinity: a VM Load Balancing

Algorithm for Cloud Based Infrastructure [J]. Journal of Information Processing Systems，2013, 9(3): 379-394.

[40] 戢友. OpenStack 开源云王者归来[M].北京：清华大学出版社，2012：12-15.

[41] 英特尔开源技术中心. OpenStack 设计与实现（第 2 版）[M].北京：电子工业出版社，2017：2.

[42] 凯文·杰克逊，科迪·邦奇. OpenStack 云计算实战手册（第 3 版）[M].北京：人民邮电出版社，2018：1-200.

[43] 维基百科. Ceph [EB/OL]. [2017-12-20]. http://tracker.ceph.com/projects/ceph/wiki/.

[44] Weil S A, Brandt S A, Miller E L, et al. Ceph: A scalable, high-performance distributed file system [C]. Proceedings of the 7th symposium on Operating systems design and implementation.USENIX Association, 2006: 307-320.

[45] Singh K. Ceph Cookbook [M]. Birmingham, UK: Packt Publishing Ltd, 2016: 53.

[46] Miell I, Sayers A H. Docker in Practice [M]. Shelter Island, New York: Manning Publications Co. 2016: 124-157.

[47] Ceph Homepage. Architecture [EB/OL]. [2018-01-23]. http://docs.ceph.com/docs/master/architecture/.

[48] Get Docker. What is Docker [EB/OL]. [2018-02-13]. https://www.docker.com/what-docker.

[49] Turnbull J. The Docker Book: Containerization is the 7new virtualization [M]. Brooklyn, New York: James Turnbull, 2014: 4-7.

[50] Docker Documentation. Swarm mode overview [EB/OL]. [2018-02-17] https://docs.docker.com/engine/swarm/.

[51] Docker Documentation. Swarm mode overview [EB/OL]. [2018-02-17]. https://docs.docker.com/engine/swarm/how-swarm-mode-works/services/

[52] Weil S A, Brandt SA, Miller EL, et al. CRUSHR: Controlled, Scalable, Decentralized placement of Replicated Data [C]. Proceedings of 2006 ACM/IEEE Conference on Supercomputing. Tampa, USA: ACM Press, 2006: 367-378.

[53] 百度百科.服务等级协议 [EB/OL]. [2018-02-18]. https://baike.baidu.com/item/SLA/2957862?fr=aladdin.

[54] Feller E, Morin C, Leprince D. State of the art of power saving in clusters and results from the EDF case study [D]. Institut National de Recherche en Informatique et en Au-tomatique

(INRIA), 2010.

[55] Ren C, Wang D, Urgaonkar B, et al. Carbon-aware energy capacity planning for datacenters [C]. Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on. IEEE, 2012: 391-400.

[56] Kontorinis V, Zhang L E, Aksanli B, et al. Managing distrib-uted ups energy for effective power capping in data cen-ters [C]. Computer Architecture (ISCA), 2012 39th Annual International Symposium on. IEEE, 2012: 488-499.

[57] 罗亮，吴文峻，张飞.面向云计算数据中心的能耗建模方法[J].软件学报，2014(7)：1371-1387.

[58] Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for cloud computing [C]. Proceedings of the 2008 conference on Power aware computing and systems. 2008: 1-5.

附 录

A 作者在硕士研究生期间参加的科研项目及成果

（一） 参加的科研项目

参加贵州省基础研究重大项目 1 项

（二） 发表的论文

[1] 在北大中文核心期刊《计算机工程与应用》中发表论文 1 篇。

[2] 在北大中文核心期刊《计算机应用》中发表论文 1 篇。

（三） 申请的软件著作权

申请软件著作权三项

B 表目录

C 图目录

D 运行区与待机区机架数量关系$b \geq \left\lceil \frac{a}{2} \right\rceil$证明。

本文第三章提出的副本策略中（本文第 23 页），数据副本数是$R(R \geq 3)$，为了保证用户数据的可用性和达到节约能耗的目的，将$\left\lceil \frac{R}{2} \right\rceil$个数据副本存储在运行区，剩下的$R - \left\lceil \frac{R}{2} \right\rceil$个数据副本存储在待机区，所以运行区和待机区的服务器数量存在着数量关系，又因为每个机架中服务器的数量是一样（每个机架中都有m台服务器），因此运行区和待机区的机架数量也存在着相同的数量关系，则有：

$$\frac{a}{b} = \frac{\left\lceil \frac{R}{2} \right\rceil}{R - \left\lceil \frac{R}{2} \right\rceil} \leq \frac{\left\lceil \frac{R}{2} \right\rceil}{R - \frac{R}{2}} = \frac{\left\lceil \frac{R}{2} \right\rceil}{\frac{R}{2}} = Z,$$

当R为偶数时，$\left\lceil \frac{R}{2} \right\rceil = \frac{R}{2}$，故$Z = 1 < 2,$

当R为基数时，$\left\lceil \frac{R}{2} \right\rceil = 1 + \frac{R}{2}$，故$Z = \frac{(1 + \frac{R}{2})}{\frac{R}{2}} = 1 + \frac{2}{R} < 2,$

故$\frac{a}{b} \leq Z < 2,$即$b > \frac{a}{2},$

又由于$b \in Z^{+}$，故$b \geq \left\lceil \frac{a}{2} \right\rceil.$

原 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究在做出重要贡献的个人和集体，均已在文中以明确方式标明。本人在导师指导下所完成的学位论文及相关的职务作品，知识产权归属贵州大学。本人完全意识到本声明的法律责任由本人承担。

论文作者签名：_____ 日　期：_____年　月　日

关于学位论文使用授权的声明

本人完全了解贵州大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权贵州大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

本学位论文属于：

保　密（　），在_____年解密后适用授权。

不保密（　）

（请在以上相应方框内打“√”）

论文作者签名：_____导师签名：_____

日 期：_____年　月　日