一种提高虚拟化 Hadoop 系统数据本地性的资源调度方法

孙瑞琦[1,2] 杨杰[2] 高瞻[2] 贺志强[1,3]
[1] (北京航空航天大学计算机学院，北京 100191)
[2] (联想研究院云与智能计算实验室，北京 100085)
[3] (联想集团，北京 100085)
(sunruiqi2012@buaa.edu.cn)

A Resource Scheduling Approach to Improving Data Locality for Virtualized Hadoop Cluster on YARN

Sun Ruiqi[1,2], Yang Jie[2], Gao Zhan[2], and He Zhiqiang[1,3]
[1] (School of Computer Science and Engineering, Beihang University, Beijing 100191)
[2] (Cloud and Intelligent Computing Lab, Lenovo Research & Technology, Beijing 100085)
[3] (Lenovo Group, Beijing 100085)

Abstract In Hadoop system, the data locality is a critical factor impacting on performance of large-scale data analysis applications. However, most of existing resources scheduling approaches to improving data locality study on the traditional Hadoop cluster which is built using physical machines. These approaches are not effective in virtualized Hadoop clusters because of two levels distribution of data: virtual machines and physical servers. In this paper, we deploy virtualized Hadoop cluster in which computing node and storage node are placed in respective virtual machines to improve flexibility. Moreover, we propose a novel resource scheduling approach which aims to improve data locality for virtualized Hadoop cluster through two key methods. One is adjusting the computational capability of virtual machine acted as computing node whose input data stored in a virtual machine acted as storage node running in the same physical server, and the other is migrating the virtual machine acted as computing node to the physical server running virtual machine acted as storage node that holds a data replica needed by that node. Our experiment results show that our approach improves performance of 86% typical MapReduce applications in HiBench benchmark suite at varying degrees. Specially, in the case of running TeraSort application in the benchmark suite processing 10GB input data, the job completion time in our approach is 33% shorter than that of traditional manner.

Key words virtualized Hadoop; data locality; resource scheduling; live migrating

摘要 在 Hadoop 系统中，大规模数据分析应用程序的数据本地性是影响其性能的关键因素。传统的 Hadoop 系统是部署在物理机中的，目前针对传统 Hadoop 系统提高数据本地性的资源调度方法在虚拟化的 Hadoop 系统中效果不佳。这是因为在虚拟化的 Hadoop 中，数据的分布被分为了两个层次：虚拟机和物理机。该文采用将计算节点和存储节点分别部署在不同虚拟机中的方式部署虚拟化 Hadoop 系统，并提出了一种提高数据本地性的资源调度方法。首先，在任务提交阶段，调节作为计算节点的虚拟机的计算能力，使其能够运行较高数据本地性的任务；其次，在任务运行阶段，通过迁移计算节点到任务所需数据存储节点所在的物理机以提高数据本地性。实验表明，本文提出的方法能够使 86%的测试程序的作业完成时间在不同程度上有所减少。特别地，在测试案例 TeraSort 中处理 10GB 的数据，本文的方法比传统方法缩短了 33%的作业完成时间。

收稿日期：2014-11-15

关键词 虚拟化 Hadoop；数据本地性；资源调度；在线迁移

中图法分类号 TP39

近年来，大数据分析技术[1]广泛应用于各大 IT 企业的数据中心，如：Yahoo!、Google、Facebook 和 IBM 等。大数据技术之所以能够在现代企业中得到广泛应用的原因之一就是分布式计算框架 MapReduce[2] 的流行，尤其是其开源实现 Hadoop。Hadoop 提供了一套并行计算的框架，使开发人员只需关注具体业务，不需要关心负载的分布、容错等细节。

随着 Hadoop 的广泛使用，人们搭建 Hadoop 系统的规模越来越大，处理的数据越来越多，并且应用的场景也越来越复杂。Hadoop 系统最初设计时的一些缺点逐渐暴露出来，如：只支持 MapReduce 单一编程模型、集中式任务调度等。为了解决上述问题，新一代 Hadoop 计算平台 YARN[3]应运而生。YARN 使 Hadoop 具有更强的可扩展性、并且除 MapReduce 之外还支持其他多种编程模型，如：Tez、Storm、Spark[4]、Dryad[5]、REEF[6]等。

采用物理服务器搭建的 Hadoop 系统，随着规模的不断扩大，一些缺点表现的越来越明显，如：繁重的物理机管理任务、波动的资源利用率等。随着云计算[7-8]技术的不断发展，越来越多的应用系统部署到了云平台中，即虚拟化的数据中心。虚拟机集群的使用，可以简化集群的管理，通过负载聚合可以提高资源利用率和节约能耗。因此，越来越多的研究者致力于虚拟机集群的研究，包括：高可用与故障恢复[9-10]、安全与隐私保护[11]、大规模数据分析[12-20]等。虚拟化为系统管理带来诸多方便的同时，也为系统的性能带来了一定的损失，尤其是 I/O 性能。运行在 Hadoop 计算平台 YARN 中的应用程序，特别是 MapReduce 程序，具有大量的 I/O 操作（本地读写和跨节点读写）。跨节点的数据传输是制约此类应用程序在虚拟化 Hadoop 系统中性能的一个重要瓶颈。

传统的 Hadoop 系统使用物理服务器搭建，根据计算任务所运行的节点及其要处理的数据所存储的节点的相对关系，数据本地性被分为以下三种：“节点数据本地性”，即计算任务及其要处理的数据都在同一个节点中；“机架数据本地性”，即计算任务所运行的节点和所要处理的数据存储的节点不是相同节点，但位于相同的机架上；“跨机架数据本地性”，即计算任务所运行的节点和所要处理的数据存储的节点不在相同的机架上。Hadoop 系统的应用程序具有大量的数据读写操作，因此提高“节点数据本地性”是提高此类应用程序性能的关键，也是目前 Hadoop 系统性能优化的一个研究热点。

研究者们在围绕“数据本地性”优化 Hadoop 系统应用程序性能的工作中，从不同角度提出了相关方案，如：数据副本的放置[12-13]、任务调度[14-16]和资源管理方法[17-20]。然而，在虚拟化的 Hadoop 系统中，由于增加了虚拟化层，数据存储分布被分为了两个层次：虚拟机和宿主机。虚拟机作为逻辑上和管理上的 Hadoop 系统计算节点和存储节点，宿主机作为实际的计算和存储节点。现有的针对传统 Hadoop 系统，提高数据本地性的方法，在虚拟化环境中效果不佳。

本文研究虚拟化 Hadoop 系统的性能优化问题，采用计算节点和存储节点分离部署的方式搭建虚拟化 Hadoop 系统，并提出了一种资源调度方法通过两种机制提高 Hadoop 系统应用程序的数据本地性。一方面，在计算任务提交阶段，调节计算节点虚拟机的计算能力使其能够运行“节点数据本地性”的计算任务；另一方面，在计算任务运行阶段，通过在线迁移[21]计算节点虚拟机到数据所存储的节点宿主机以实现“节点数据本地性”。本文使用 HiBench[22]基准测试套件中 7 种典型 MapReduce 应用程序对我们提出的方案进行了测试，实验表明，该方案在不同程度上缩短了 86%的应用的作业完成时间。特别地，在 TeraSort 测试案例中，使用 96 个 Map 任务和 48 个 Reduce 任务对 10GB 的数据进行排序，比传统方式缩短了 33%的作业完成时间。

本文的主要贡献如下：提出了一种分离式虚拟化 Hadoop 的部署方式，即计算节点和存储节点分别部署在不同的虚拟机中，便于提高数据本地性的资源管理；提出了一种新颖的资源调度方法，通过两种机制提高虚拟化 Hadoop 系统的“节点数据本地性”；使用多种典型的 MapReduce 应用程序在三种不同场景中进行了测试并对比分析了试验结果。

1 相关工作

目前，很多研究者致力于提高 Hadoop 应用程序性能的研究。然而，现有的大部分研究工作主要针对物理机搭建的 Hadoop 系统。本文旨在研究虚拟化 Hadoop 系统中应用程序性能的优化问题，与本文相关的研究工作可归纳为以下三类：数据副本的放置策略、任务调度以及资源管理。

收稿日期：2014-11-15

1.1 数据副本的放置策略

在 MapReduce[2]的最初实现中就考虑了数据本地性对应用程序性能的影响，采取的策略是优先将计算任务调度到存储数据的节点上运行，如果失败就尽量将计算任务调度到与存储数据的节点在相同机架的其他节点运行。此后，很多研究者关注提高数据本地性的任务调度算法，并提出了不同任务调度策略。文献[12]提出的系统 Scarlett 根据数据块被访问的频度决定数据块副本的存储位置，从而缓解数据副本热点并提高任务执行速度。文献[13]的提出的 ADAPT 是基于数据块可用性的敏感程度决定数据副本放置的策略，目的在于提高数据本地性并降低网络流量。

1.2 任务调度

文献[14]于 2008 年提出的 LATE（Longest Approximate Time to End）算法目的在于提高异构环境中 MapReduce 程序的性能，它通过正确地识别出 Stragglers[2]，即执行进度明显比同类任务慢的任务并进行精确地推测执行。文献[15]于 2010 年提出的延迟调度算法通过牺牲部分公平性以提高数据本地性。这些方法在虚拟化 Hadoop 系统中效果不佳，是因为虚拟化环境中的数块存储分布被分为了两个层次：虚拟机和宿主机，原有的数据块副本定位机制不够准确。

文献[16]针对虚拟化环境提出的 ILA（Interference and Locality-Aware）任务调度算法是通过减少相同宿主机上运行的多虚拟机之间的性能干扰并优化数据本地性以提高 MapReduce 应用的性能。但是，该方法并未考虑虚拟化 Hadoop 系统的传统部署方式带来的性能开销。

1.3 资源管理

文献[17]提出的 TRACON 是一种干扰相关的资源管理算法，在虚拟化环境中提高数据密集型应用的性能。文献[18]提出的 HybridMR 是在分别使用物理机服务器和虚拟机搭建的混合式 Hadoop 系统中，根据任务的紧迫程度分别分配到不同系统中。文献[19]提出的 MROrchestrator 能够动态监测运行环境中的资源需求瓶颈并通过细粒度调整按需调度资源。文献[20]提出的 Mesos 是一种资源管理平台，可用来管理多种不同计算集群（如：Hadoop，MPI 等）的资源，使之能够资源共享。

本文首先从虚拟化 Hadoop 系统的部署方式入手，提出了一种分离部署方式，即将计算节点和存储节点分别部署到不同的虚拟机中。在此基础上，对 Hadoop 系统的应用程序的数据本地性进行优化。

2 虚拟化 Hadoop 系统的数据本地性

在 Hadoop 系统中，MapReduce 并行程序计算框架将处理一大批数据的计算作业划分为多个小的计算任务，每个小的计算任务处理一小块数据（YARN 计算平台默认为 128MB）。根据一定的资源调度策略，每个小的计算任务被调度到特定计算节点进行运算。每一小块数据会存储多个副本，YARN 默认的副本数为 3，即每一小块数据会同时存储在三个不同存储节点中。在 YARN 资源调度策略中优先将一个计算任务调度到存储该计算任务所要处理的数据的任意存储节点中。如果相应的数据块所有副本存储节点都无法运行该计算任务，则将其调度到与该数据副本存储节点相同机架的其他节点运行。

如前文所述，在原生 Hadoop 系统中，根据计算任务所运行的节点及其要处理的数据所存储的节点的相对关系，数据本地性被分为三种，即：“节点数据本地性”、“机架数据本地性”和“跨机架数据本地性”。在虚拟化的 Hadoop 系统中，由于增加了虚拟化层，数据本地性可分为以下四种：

(1) “虚拟机数据本地性”，即计算任务及其要处理的数据都在同一个虚拟机节点中；

(2) “宿主机数据本地性”，即计算任务及其要处理的数据都在同一台宿主机中；

(3) “机架数据本地性”，即计算任务所运行的虚拟机节点和所要处理的数据存储的虚拟机节点不在相同宿主机上，但这些宿主机位于相同的机架中；

(4) “跨机架数据本地性”，即计算任务所运行的虚拟机节点和所要处理的数据存储的虚拟机节点不在相同的宿主机上，且这些宿主机位于不同的机架中。

实验表明[16]，在同等条件下，满足不同数据本地性的计算任务执行速度由快到慢的顺序是：“虚拟机数据本地性”、“宿主机数据本地性”、“机架数据本地性”、“跨机架数据本地性”。其中，满足“虚拟机数据本地性”和“宿主机数据本地性”的作业完成时间接近，而满足“机架数据本地性”和“跨机架数据本地性”的作业完成时间分别是前者的 3 倍和 4 倍。

Hadoop 系统的应用程序具有大量的数据传输操作，并且虚拟化会对 I/O 的性能带来一定的影响，随着要处理的数据量的增加，数据本地性对计算任务的执行效率影响越来越大。因此提高“虚拟机数据本地性”和“宿主机数据本地性”可以有效减少节点间的网络数据传输，从而提高此类应用程序的性能。

3 虚拟化 Hadoop 系统的部署方式

3.1 传统部署方式

原生的 Hadoop 系统是部署在物理机中的。通常计算节点兼做存储节点，这样可以在一定程度上保证“节点数据本地性”。在虚拟化环境中也可以采用这种部署方式，该部署方式的逻辑架构如图 1 所示。该图中，RM、NN、NM 和 DN 分别表示 YARN 系统中的 Resource Manager、Name Node、Node Manager 和 Data Node。

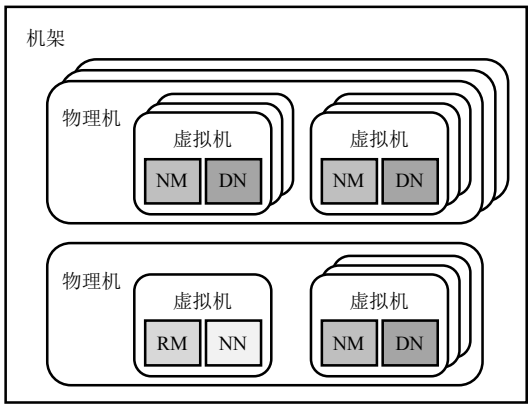

图 1 传统部署方式逻辑架构

在虚拟化环境中采用这种传统部署方式部署的 Hadoop 系统具有以下两个缺点。

第一，系统扩展性差。在虚拟化环境中，可以通过动态增删虚拟机来实现虚拟机集群的弹性伸缩。在虚拟化的 Hadoop 系统中，通常针对计算节点数量进行弹性伸缩以保证任务的高效运行和节省资源成本。然而，在这种部署方式中，作为计算节点的虚拟机兼做存储节点，增加或删除计算节点虚拟机时，相应的存储节点也不得不增加或删除。增加存储节点，会增加资源成本；删除存储节点，会造成数据副本缺失从而引起后台的数据块自动备份。

第二，虚拟机迁移效率低。在虚拟化环境中，为了提高资源利用率、降低能耗、或方便物理机维护等，通常会通过在线迁移虚拟机进行负载整合。在这种部署方式中，作为计算节点的虚拟机兼做存储节点，迁移虚拟机时要进行大量是数据移动。

在采用这种传统部署方式的虚拟化 Hadoop 系统中，前文所述的四种数据本地性都可能存在。YARN 的资源调度算法中已经考虑到了数据本地性问题，但是采用的策略是优先将计算任务调度到存储数据的虚拟机节点上运行，如果相应的存储节点无法运行该计算任务，则尽量将其调度到与存储数据的虚拟机节点在相同机架的其他虚拟机节点运行。事实上，这种策略无法保证“虚拟机数据本地性”，另一方面，在虚拟化的 Hadoop 系统中，由于增加了虚拟化层，数据块副本的分布被分为两个层次：虚拟机和宿主机。因此原生的 Hadoop 系统中数据本地性机制效果会受到一定影响。

3.2 分离部署方式

在虚拟化环境中，另一种 Hadoop 系统的部署方式是分离部署方式，即将计算节点和存储节点分别部署到不同的虚拟机中，该部署方式的逻辑架构如图 2 所示。该图中，RM、NN、NM 和 DN 分别表示 YARN 系统中的 Resource Manager、Name Node、Node Manager 和 Data Node。其中 Resource Manager 虚拟机和 Name Node 虚拟机运行在同一台物理机中，这两台虚拟化也可以运行在不同的物理机中。

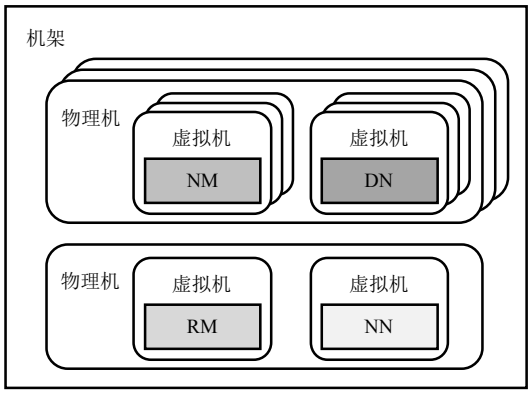

图 2 分离部署方式逻辑架构

分离部署方式可以克服传统部署方式的缺点。首先，扩展性强。在该部署方式下，可以单独增加或删除计算节点，而存储节点不受影响，因此不会造成后台服务对数据块副本的调整。第二，虚拟机迁移灵活。当对计算节点进行迁移时，不会同时迁移存储节点。

在采用这种分离部署方式的虚拟化 Hadoop 系统中，不存在“虚拟机数据本地性”，然而满足“虚拟机数据本地性”和“宿主机数据本地性”的计算任务的执行效率接近，原因是运行在同一台物理机中的虚拟机之间进行网络数据传输时并不消耗网络带宽[23]，这跟本地数据传输效率接近。因此可以通过提高“宿主机数据本地性”来优化该部署方式下的 Hadoop 系统应用程序的性能。同时，在该部署方式中，为提高“宿主机数据本地性”提供了可行的优化空间，例如：调整计算节点虚拟机的计算能力，使其能够运行满足条件的计算任务；在线迁移计算节点虚拟机到相应的宿主机，使其运行的计算任务满足“宿主机数据本地性”等。另外，在该部署方式中，由于计算节点和存储节点分别部署在不同的虚拟机中，因此可以很方便地单独对计算节点进行计算能力的调整或在线迁移，同时不会影响到存储节点的运行和性能。

本文采用分离部署方式搭建虚拟化 Hadoop 系统，在该部署方式中，不同的数据本地性如图 3 所示。如前文所述，在分离部署方式中不存在“虚拟机数据本地性”，图 3 中三种不同类型的箭头分别表示其他三种数据本地性。其中 NM 和 DN 分别表示 YARN 系统中的 Node Manager 和 Data Node，而 Resource Manager 和 Name Node 可以部署在任意物理机中的虚拟机上，故图 3 中没有标出。

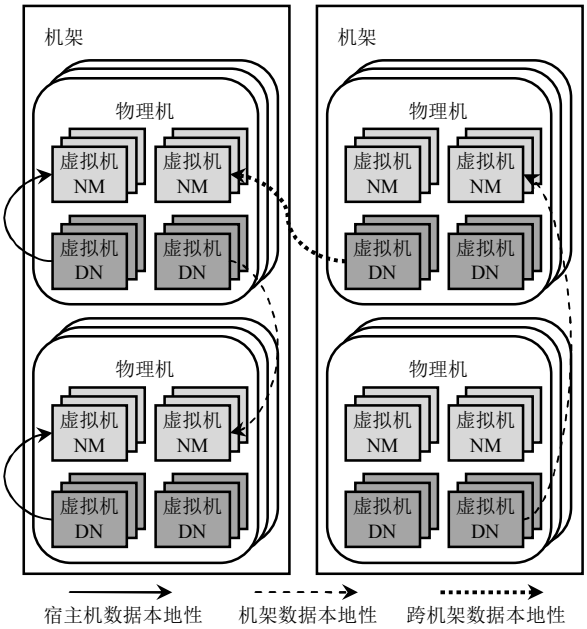

图 3 分离部署方式中的不同数据本地性示意图

为了提高虚拟化 Hadoop 系统的“宿主机数据本地性”，本文提出了一种资源调度方法，在应用程序运行的不同阶段采用不同的调度机制。第一，在任务提交阶段，对不能满足“宿主机数据本地性”的计算任务，需要根据数据块副本的分布，通过调整合适的计算节点虚拟机的计算能力，使其能够运行相应计算任务。由于宿主机的计算能力的限制，在该阶段无法满足“宿主机数据本地性”的计算任务会在第二阶段进行调整。第二，在任务运行阶段，由于系统资源在动态变化，对于没有满足“宿主机数据本地性”的计算任务，可以通过在线迁移计算节点虚拟机到合适的宿主机运行，使其满足“宿主机数据本地性”。

4 资源调度系统

4.1 资源调度系统的设计

本文以虚拟化环境中的 Hadoop 计算平台 YARN 为研究对象，提出了一种资源调度方法，目的是通过提高数据本地性来优化 Hadoop 应用程序的性能。

在 Hadoop 系统中，MapReduce 并行程序计算框架将处理一大批数据的计算作业划分为多个 Map 任务和 Reduce 任务。每个 Map 任务处理一个小数据块（YARN 计算平台默认为 128MB），并生成中间结果。每个 Reduce 任务处理 Map 任务输出的中间结果并计算生成最终结果。

本文提出的资源调度系统的逻辑架构如图 4 所示。该系统包括以下四个主要部分：作业解析器、资源调节器、任务解析器和迁移控制器。图 4 中的编号表示该系统的工作流程。首先，作业被提交给作业解析器（步骤①）。作业解析器负责解析该作业包含的所有任务以及每个任务要处理的数据块副本的分布信息。如果按照 YARN 的资源调度策略可以将一项任务调度到满足“宿主机数据本地性”的计算节点中，则不做调整；否则，由资源调节器（步骤②）根据数据块副本的分布，选择具备增加计算机能力的计算节点虚拟机，通过资源调节器增加该节点的计算能力。在调整虚拟机计算能力的过程中，会遇到所在宿主机计算能力有限的情况，不足以支持虚拟机计算能力的增加，此时，需要迁移控制器（步骤③）将该宿主机上的部分虚拟机在线迁移到其他宿主机。在上述过程中，对于最终无法满足“宿主机数据本地性”的计算任务，我们采取 YARN 默认方式，即尽量将其调度到“机架数据本地性”的计算节点运行。在运行阶段，由任务解析器（步骤④）负责监测运行任务的数据本地性及其执行进度，尤其是没有满足“宿主机数据本地性”的计算任务。对于执行进度明显比其他计算任务缓慢的计算任务，由迁移控制器（步骤⑤）根据数据副本块的分布与宿主机的计算机资源使用情况，选择出能够满足“宿主机数据本地性”的宿主机，并对该计算节点虚拟机进行在线迁移。

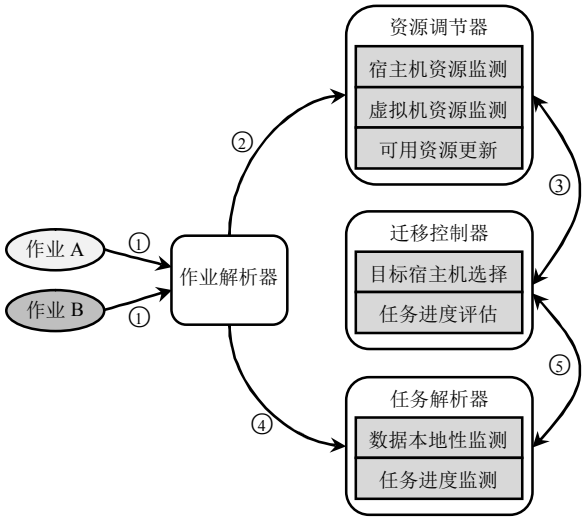

图 4 资源调度系统逻辑架构

4.1.1 作业解析器

当一项作业被提交到系统中后，首先由作业解析

器分析该作业包含的所有任务以及每个任务要处理的数据块副本的分布信息。

在 YARN 中，资源管理节点（Resource Manager，RM）是整个 Hadoop 集群所有资源的最终仲裁模块。当一项作业被提交到 Hadoop 系统中后，RM 决定该作业的所有任务的执行计划，RM 决定哪些任务处理哪些数据，哪个计算任务被调度到哪个节点上运行。关于作业执行计划的所有细节，都可以在 RM 运行的日志文件中找到。本文设计的资源调度系统需要的信息都包含在该日志文件中，如：该作业包含的任务列表，每个任务处理的数据块副本分布信息等。

4.1.2 资源调节器

在任务提交阶段，按照 YARN 默认的资源调度策略，对于无法满足“宿主机数据本地性”的计算任务，资源调节器根据数据块副本分布、宿主机的资源使用情况以及计算节点虚拟机的资源配置情况等信息，调节计算节点虚拟机的计算能力使之能够满足“宿主机数据本地性”。资源调节器包括以下三个模块：宿主机资源监测模块、虚拟机资源监测模块和可用资源更新模块。

宿主机资源监测模块，负责探测指定宿主机的最大可用资源量（如：CPU，内存，硬盘和网络带宽等）并监测当前可用资源量。虚拟机资源监测模块，负责探测指定宿主机中所有虚拟机最大可用资源量和当前可用资源量。可用资源更新模块，负责增加或减少 YARN 的计算节点（Node Manger，NM）虚拟机的计算能力，并将 NM 虚拟机计算能力的更新通知给 RM。

对计算节点虚拟机计算能力的资源调节涉及到数据块副本的存储分布，虚拟机可调整的资源配置量，宿主机可用资源量以及计算任务所需资源量等，算法详细见第 4.2.1 节。

4.1.3 任务解析器

任务解析器负责监测运行任务的数据本地性及其执行进度，尤其是没有满足“宿主机数据本地性”的计算任务。任务解析器包括两个模块：数据本地性监测模块和任务进度监测模块。

数据本地性监测模块的主要功能是监测所有运行任务的数据本地性类型，并输出所有“机架数据本地性”和“跨机架数据本地性”的计算任务列表。

任务进度监测模块的主要功能是监测所有运行任务的执行进度并找出执行进度缓慢的任务。在 Hadoop 系统中，Progress Score（PS）是一项度量运行任务执行进度的度量指标，取值在 0 和 1 之间[15]。对于 Map 任务，PS 表示被 Map 任务处理过的输入数据的比例；对于 Reduce 任务，PS 表示被 Reduce 任务处理过的中间数据的比例。从一个作业中分解出来的所有 Map 任务或者 Reduce 任务要处理的数据和处理算法基本一致，理想条件下所有类似任务的进度基本一致，但实际运行过程中不同计算任务的完成时间存在很大差距。因此 PS 可以用来表示计算任务的执行进度并找出运行缓慢的任务。

另外，为运行缓慢的计算任务选择合适的宿主机时，要用到一项重要度量指标，即任务的剩余完成时间。本文采用一种简单算法[15]来预测任务的剩余完成时间，表达式如下：

$$T_{remain} = T_{elapsed} \frac{1 - ProgressScore}{ProgressScore}, \quad (1)$$

表达式（1）中 $T_{elapsed}$ 表示任务已经运行的时间，$ProgressScore$ 表示该任务的 PS 值，T_{remain} 即该任务剩余完成时间的预测值。

4.1.4 迁移控制器

迁移控制器是本文提出的资源调度系统的核心组成部分。它负责选举合适的宿主机作为迁移计算节点虚拟机的目的地并评估计算任务在目的地宿主机上的运行进度。迁移控制器包括两个模块：目标宿主机选择模块和任务进度评估模块。

目标宿主机选择模块，负责选举合适的宿主机用于迁移计算节点虚拟机的目的地宿主机。算法详细见第 4.2.2 节。

任务进度评估模块，根据任务的剩余完成时间的预测值来评估计算任务在目的地宿主机的执行进度。

4.2 资源调度系统的核心算法

4.2.1 NM 虚拟机计算能力的调整策略

在虚拟化环境中，基础设施的物理资源通常是被分时复用的[24]，即将一定量的实际物理资源按照超过 1 倍的量分配给多台虚拟机，实际上并未真正向虚拟机分配物理资源，只有当虚拟机真正需要相应资源时才分配实际的物理资源。按照虚拟机运行时所需资源的类型，可以合理放置虚拟机到不同宿主机中。在虚拟机运行过程中可以采用 ballooning 技术实现虚拟机计算能力的在线调整，并且虚拟化管理工具 libvirt 提供了相应的命令，如：setmaxmem（修改最大内存分配量）、setmem（修改内存分配量）和 setvcpus（修改虚拟 CPU 数量）。

在 YARN 中，RM 调度资源的最小单位是 Container，每个 Container 封装了一定量的资源，如：CPU、内存、硬盘和网络带宽等。目前的 YARN 版本（Hadoop 2.4.1）只支持内存的在线调整，对其他资源的支持还在开发中。为了全面支持各种资源的在线调整，我们建立了通用的模型。假定一台宿主机中运

行了 m 台 NM 虚拟机，一项新提交的作业需要 n 种资源，我们给出的定义见表 1。

表 1 变量表示

定义	格式	注释
Host_Max	1*n 的行矩阵	宿主机的最大可用资源量
Host_Ava	1*n 的行矩阵	宿主机的当前可用资源量
Task_Req	1*n 的行矩阵	任务需求的资源量
Init_Max	m*n 的矩阵	每一行表示一个 NM 虚拟机被创建时设置的最大可用资源量
Curr_Max	m*n 的矩阵	每一行表示一个 NM 虚拟机当前的最大可用资源量
Marg_Max	m*n 的矩阵	每一行表示一个 NM 虚拟机剩余可供调整的资源量
Curr_Ava	m*n 的矩阵	每一行表示一个 NM 虚拟机当前可用资源量

本文中，我们提出了一种 NM 虚拟机计算能力调整策略的算法（算法 1）。该算法是任务提交阶段的资源调度机制，由于宿主机的资源限制，并不能使所有任务都满足“宿主机数据本地性”。对于该算法失效的计算任务，我们暂且将其部署到“机架数据本地性”或“跨机架数据本地性”的节点运行。进一步，在任务的运行阶段，由任务解析器和迁移控制器负责监控这些计算任务的运行并进行优化。

算法 1. NM 虚拟机计算能力调整策略.

输入：Host_Max, Task_Req, Init_Max, Curr_Max, Curr_Ava;

输出：指定的任务是否可以部署在指定宿主机中，Success;

①初始化 Marg_Max=Init_Max-Curr_Max;
② FOR i =0 to m
③ IF Task_Req≤Curr_Ava 的第 i 行; THEN
④ 部署该任务到第 i 行的 NM 虚拟机中;
⑤ Success=True;
⑥ RETURN Success;
⑦ ENDIF
⑧ ENDFOR
⑨IF Task_Req＞(Curr_Ava+Marg_Max)的所有行之和; THEN
⑩ Success=False;
⑪ RETURN Success;
⑫ ELSE
⑬ FOR j=0 to m
⑭ IF Host_Max≥(Curr_Ava+Marg_Max)的第 j 行; THEN
⑮ 调整第 j 行的 NM 虚拟机，并部署该任务到此 NM;
⑯ Success=True;
⑰ RETURN Success;
⑱ ENDIF
⑲ ENDFOR
⑳ 随机选择 VM 并在线迁移到其他宿主机，增加本宿主机的可用资源量;
㉑ FOR k=0 to m
㉒ IF Task_Req≤Curr_Ava 的第 k 行; THEN
㉓ 部署该任务到第 k 行的 NM 虚拟机中;
㉔ Success=True;
㉕ RETURN Success;
㉖ ENDIF
㉗ ENDFOR
㉘ Success=False;
㉙ RETURN Success;
㉚ ENDIF

4.2.2 目标宿主机的选择策略

在任务运行阶段，由任务解析器监测并识别出执行进度缓慢的任务，然后由迁移控制器选举出适合的目标宿主机对缓慢任务运行的虚拟机进行迁移，使其能够满足“宿主机数据本地性”。本文提出了一种基于数据块副本分布信息和任务剩余完成时间预测的目标宿主机选择算法，见算法 2。

算法 2. 目标宿主机选择策略.

输入：执行缓慢的计算任务，Straggler;

输出：适合运行该计算任务的目标宿主机列表，Lserver;

① 初始化 Lserver=null;
② 获取计算任务 Straggler 当前运行的节点，记为 Stask;
③ 使用表达式（1）计算 Straggler 的剩余完成时间，记为 Tstraggler
④ 从作业解析器获取可作为目标宿主机的候选者列表，记为 Cserver;
⑤ FOR EACH Cs∈Cserver DO
⑥ IF Cs≠Stask THEN
⑦ IF Tremain(Straggler, Cs)＜Tstraggler THEN
⑧ 添加 Cs 到 Lserver 中;
⑨ ENDIF
⑩ ENDIF
⑪ ENDFOR
⑫ IF Lserver=null THEN
⑬ 添加 Stask 到 Lserver 中;
⑭ ENDIF
⑮ RETURN Lserver;

在算法 2 中，我们忽略了在线迁移虚拟机产生的时间开销，这是因为在线迁移虚拟机的时间开销相对于长时间运行的 Hadoop 作业而言是微乎其微的。

5 实验与分析

5.1 实验环境

实验平台选用了两台配置均为 2 颗 64 位的 Intel Xeon 处理器 E5-2640（6 Cores，12 threads，2.50GHz）、64GB 内存、8 块 600GB 硬盘的 ThinkServer RD630 服务器，并通过千兆交换机相连。服务器采用内核版本为 3.9.11 的 SUSE Linux Enterprise Server 11 操作系统和版本为 1.2.0 的 qemu-kvm 虚拟化平台。配置 2 个 vCPU，4GB 内存和 20GB 硬盘的虚拟机，并安装与物理服务器相同版本的 Linux 操作系统。

我们采用分离部署方式搭建了 22 台虚拟机节点的 Hadoop 系统，其中计算节点虚拟机和存储节点虚拟机分别 10 台，另外两台虚拟机分别运行 RM 和 NN。实验用到的 Hadoop 系统版本为 2.4.1，所有参数使用系统默认值。

目前，本文提出的方案是基于 KVM 虚拟化平台实现的，对于其他支持虚拟机在线调整计算能力和在线迁移功能的虚拟化平台，如 Xen 和 VMWare 等，本方案都可以很方便地支持。

5.2 性能分析

在本实验中，我们分别对以下三种测试场景进行了测试。

传统部署方式下默认调度方法(Traditional Deployment with Default scheduling approach，TDD):这是目前大部分虚拟化 Hadoop 使用者们最普遍的使用方式，部署方式的逻辑架构见图 1。在该场景中，所有 Hadoop 虚拟机节点同时扮演计算节点和存储节点的角色；YARN 资源调度方法没有保证“宿主机数据本地性”。

分离部署方式下默认调度方法(Dispersal Deployment with Default scheduling approach， DDD):部署方式的逻辑架构见图 2。在该场景中，Hadoop 系统的计算节点和存储节点分别部署在不同的虚拟机中，并使用 YARN 默认的资源调度方法。

分离部署方式下本地性相关调度方法(Dispersal Deployment with Locality-aware scheduling approach，DDL): 部署方式的逻辑架构见图 2。Hadoop 系统的计算节点和存储节点分别部署在不同的虚拟机中，并使用前文所述提出的资源调度方法。

本文使用了 HiBench[22]基准测试套件。该套件包括了 5 种不同类型的 Hadoop 应用程序：微基准测试、HDFS 基准测试、Web 搜索基准测试、机器学习基准测试和数据分析基准测试。我们选用了其中 7 个典型的 MapReduce 应用程序，在上述三种不同测试场景下分别独立运行 5 次并记录作业完成时间。对每一种 MapReduce 应用程序的测试结果进行如下处理：以本文提出的资源调度方法（DDL）的作业完成时间为基准，对另外两种场景的测试结果做标准化，如图 5 所示。

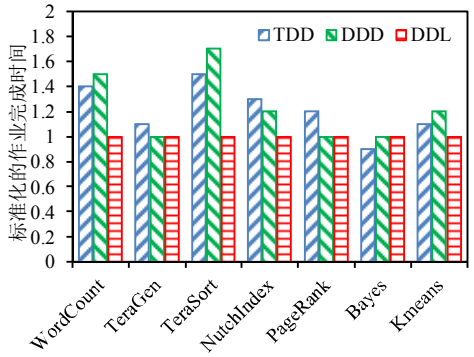

图 5 不同测试场景下标准化的作业完成时间对比

从图 5 中可以看出的最重要的是，本文所提出的资源调度方法（DDL）在不同程度上提高了 86%的 MapReduce 典型应用程序的性能。我们测试了 HiBench 基准测试套件中的 7 种典型的应用程序，我们的方法使其中的 6 种应用程序都在不同程度上比传统的虚拟化 Hadoop 系统中作业完成时间有所减少。

5.2.1 微基准测试

在这一类基准测试的应用程序中，WordCount、TeraGen 和 TeraSort 是用于评估 Hadoop 系统的三个典型的 MapReduce 应用程序。在实验中，词频统计程序 WordCount 处理 10GB 的输入数据，这些数据是由 Hadoop 系统的示例程序 RandomTextWriter 生成的。在本文提出的资源调度方法中（DDL），WordCount 的作业完成时间比传统方式（TDD）缩短了 29%。

数据排序程序 TeraSort 处理 10GB 的输入数据，该数据是由数据生成程序 TeraGen 生成的。TeraGen 程序仅包含 96 个 Map 任务，没有 Reduce 任务。该程序在三种测试场景中的作业完成时间接近。TeraSort 程序包括 96 个 Map 任务和 48 个 Reduce 任务，本文提出的资源调度方法（DDL）比传统方法（TDD）的作业完成时间缩短了 33%。

5.2.2 Web 搜索基准测试

索引程序 NutchIndex 是 Nutch 项目中的索引子系统，是一个流行的 Apache 开源搜索引擎。在实验中，我们设置了 100 万个网页，该程序的作业完成时间在

本文提出的资源调度方法中（DDL）比传统方法（TDD）缩短了 23%。

页面分级程序 PageRank 是使用 MapReduce 框架实现的页面分级算法。在试验中，我们设置了 10 万个网页，并将该程序配置为 96 个 Map 任务和 48 个 Reduce 任务，该程序的作业完成时间在本文提出的资源调度方法中（DDL）比传统方法（TDD）缩短了 17%。

5.2.3 机器学习基准测试

贝叶斯分类算法和 K 均值聚类算法是机器学习领域常用的基本算法。本实验中，K 均值聚类算法在三种测试场景下的作业完成时间接近。贝叶斯分类算法的作业完成时间在本文提出的资源调度方法中（DDL）比传统方法（TDD）延长了 10%。总体看来，本文提出的资源调度方法在机器学习算法中效果不佳。这是因为，本文提出的方法主要针对大量数据传输的 I/O 密集型 MapReduce 应用，而贝叶斯分类算法和 K 均值聚类算法等机器学习算法的网络数据传输不频繁，但对 CPU 计算资源要求高。下一步，我们计划深入研究此类机器学习算法的 MapReduce 实现的性能优化问题。

6 结束语

为了提高 MapReduce 应用程序在虚拟化的 Hadoop 系统中的性能，我们发现提高“宿主机数据本地性”是一种有效的方法。本文中，我们采用分离部署方式部署虚拟化的 Hadoop 系统，即计算节点和存储节点分别部署在不同的虚拟机中。我们提出了一种新颖的资源调度方法，目的是提高虚拟化 Hadoop 系统的应用程序的数据本地性从而优化其性能。该方法包括两种机制：一是在任务提交阶段，通过调整计算节点虚拟机的计算能力使之能够满足“宿主机数据本地性”；二是在任务运行阶段，迁移计算节点虚拟机到能够满足计算任务“宿主机数据本地性”的宿主机中运行。实验表明，在 HiBench 基准测试套件中，本文所提出的方法在不同程度上缩短了 86%的应用的任务完成时间。特别地，在 TeraSort 测试案例中，使用 96 个 Map 任务和 48 个 Reduce 任务对 10GB 的数据进行排序，比传统方式缩短了 33%的任务完成时间。

在下一步工作中，我们计划深入研究如何提高 MapReduce 实现的机器学习算法的性能，并且探索本文所提出的方法在 Spark[4]计算框架中的可行性。

参 考 文 献

[1] Karthik Kambatla, Giorgos Kollias, Vipin Kumar, Ananth Grama. Trends in big data analytics. Journal of Parallel and Distributed Computing, 2014, 74(7): 2561-2573

[2] Dean J, Ghemawat S. MapReduce: Simplifief data processing on large clusters. Communications of the ACM, 2008, 51(1): 107-113

[3] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. Apache Hadoop YARN: yet another resource negotiator//Proceedings of the 4th annual Symposium on Cloud Computing (SOCC'13). Santa Clara, California, USA, 2013: 1-16

[4] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets//Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud'10). Berkeley, CA, USA, 2010: 10-10

[5] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks//Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys'07). New York, USA, 2007: 59-72

[6] Byung-Gon Chun, Tyson Condie, Carlo Curino, Chris Douglas, Sergiy Matusevych, Brandon Myers, Shravan Narayanamurthy, Raghu Ramakrishnan, Sriram Rao, Josh Rosen, Russell Sears, and Markus Weimer. REEF: retainable evaluator execution framework. Proceedings of the VLDB Endowment, 2013, 6(12): 1370-1373

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. Communications of the ACM, 2010, 53(4): 50-58

[8] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 2009, 25(6): 599-616

[9] Sun Rui-Qi, Yang Jie, Gao Zhan, He Zhi-Qiang. LSoVC: a framework for taking live snapshot of virtual cluster in the cloud//Proceedings of the 15th IEEE International Conference on High Performance Computing and Communications (HPCC'13). Zhangjiajie, China, 2013: 1727–1732

[10] Sun Rui-Qi, Yang Jie, He Zhi-Qiang. An approach to minimizing downtime induced by taking live snapshot of virtual cluster//Proceedings of the International Conference on Cloud and Service Computing (CSC'13), Beijing, China, 2013: 63–68

[11] Zhang Xu-Yun, Liu Chang, Nepal Surya, Yang Chi, Dou Wan-Chun,

Chen Jin-Jun. SaC-FRAPP: a scalable and cost-effective framework for privacy preservation over big data on cloud. Concurrency and Computation: Practice and Experience, 2013, 25(18): 2561-2576

[12] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, E. Harris. Scarlett: Coping with skewed content popularity in mapreduce clusters//Proceedings of the 6th European Conference on Computer Systems (EuroSys'11). New York, USA, 2011: 287–300

[13] H. Jin, X. Yang, X.-H. Sun, I. Raicu. ADAPT: Availability-aware mapreduce data placement for non-dedicated distributed computing//Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS'12), Washington, DC, USA, 2012: 516–525

[14] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, I. Stoica, Improving mapreduce performance in heterogeneous environments//Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (OSDI'08), Berkeley, USA, 2008: 29–42

[15] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, I. Stoica. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling//Proceedings of the 5th European Conference on Computer Systems (EuroSys'10), New York, USA, 2010: 265–278

[16] Bu Xiang-Ping, Rao Jia, Xu Cheng-Zhong. Interference and locality-aware task scheduling for MapReduce applications in virtual clusters//Proceedings of the 22nd International Symposium on High-performance Parallel and Distributed Computing (HPDC'13). New York, USA, 2013: 227-238

[17] R. C. Chiang, H. H. Huang. TRACON: Interference-aware scheduling for data-intensive applications in virtualized environments// Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11), New York, USA, 2011: 47:1–47:12

[18] Bikash Sharma, Timothy Wood, and Chita R. Das. HybridMR: A hierarchical MapReduce scheduler for hybrid data centers//Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems (ICDCS'13). Philadelphia, USA, 2013: 102-111

[19] B. Sharma, R. Prabhakar, S. H. Lim, M. T. Kandemir, C. R. Das. Mrorchestrator: A fine-grained resource orchestration framework for mapreduce clusters//Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD'12), Washington, DC, USA, 2012: 1–8

[20] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center//Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI'11), Berkeley, USA, 2011:22–22

[21] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield. Live migration of virtual machines//Proceedings of the 2nd USENIX Conference on Networked Systems Design and Implementation (NSDI'05), Berkeley, USA, 2005: 273–286

[22] S. Huang, J. Huang, J. Dai, T. Xie, B. Huang. The hibench benchmark suite: Characterization of the mapreduce-based data analysis// Proceedings of the 26th IEEE International Conference on Data Engineering Workshops (ICDEW'10), 2010: 41–51

[23] Qi Zhang, Ling Liu, Yi Ren, Kisung Lee, Yuzhe Tang, Xu Zhao, and Yang Zhou. Residency aware inter-VM communication in virtualized cloud: performance measurement and analysis//Proceedings of the 6th IEEE International Conference on Cloud Computing (CLOUD'13). Washington, DC, USA, 2013: 204-211

[24] Sangwook Kim, Hwanju Kim, Joonwon Lee, Jinkyu Jeong. Group-based memory oversubscription for virtualized clouds. Journal of Parallel and Distributed Computing, 2014, 74(4): 2241-2256

孙瑞琦，男，1986 年生，博士研究生，主要研究方向为云计算基础设施、大规模计算资源调度、大数据分析平台管理。

杨杰，男，1978 年生，博士，主要研究领域为软件体系结构、云计算、分布式计算，先后在国内外发表论文 20 多篇。

高瞻，男，1982 年生，博士，主要研究领域为云计算。

贺志强，男，1963 年生，研究员，博士生导师，中国计算机学会(CCF)会员，主要研究领域为计算机系统结构、计算机应用技术。