2016 15th International Symposium on Parallel and Distributed Computing

一种针对大数据系统的新型性能评估与优化模型

Jungang Xu[1], Guolu Wang[1,2], Shengyuan Liu[1,2], Renfeng Liu[1]
1. 中国科学院大学
2. 中国科学院计算技术研究所计算机体系结构国家重点实验室
中国北京
电子邮件：xujg@ucas.ac.cn，{wangguolu14，liushengyuan12}@mails.ucas.ac.cn，lrflt0@gmail.com

摘要 - 近年来，互联网的发展使得全球数据量迅速增长，大数据时代的到来给传统计算带来了巨大的挑战。大数据系统，如Hadoop、Spark，正成为处理大数据的重要平台，但由于大数据应用本身的设计缺陷以及分布式框架配置不合理，大数据系统中应用的性能难以达到计算机理论的峰值速度，因此如何定位大数据系统的性能瓶颈并分析瓶颈成因值得研究。在本文中，提出了一种大数据系统的五层性能评估模型，这是性能分析的可靠基础，同时，还提出了一种大数据系统的性能优化模型，能够协助性能瓶颈定位和瓶颈分析，并进一步优化性能。基于这两个性能模型，实现了一个基于事件的性能工具来分析性能数据。实验结果表明，这两个性能模型对于大数据系统的性能评估和优化是有效的，能够使大数据系统的平均运行时间缩短19%。

MapReduce;并行程序;性能模型;瓶颈性能分析

I. 引言

随着社交网络服务、移动互联网和电子商务的迅速和持续发展，互联网用户已经贡献了巨大的内容，并且未来还将继续贡献越来越多的内容。根据IDC的预测，到2020年，数字宇宙的大小将超过44ZB。数据量的增长速度也令人震惊，全球创造的数据量每年增长超过50%。2012年，IDC预测，在未来十年内，这个数字宇宙大约每两年就会翻一番。企业越来越多地跟踪和分析这些数据以及我们与它们的互动，但到目前为止，世界上只有一小部分数据被用于分析价值。这些统计数据突显了“大数据”变得越来越重要的事实。大数据意味着巨大的深度价值，被称为新的石油，未来将在科技和经济发展中产生深远的影响。

由于大数据处理需求的紧迫性和重要性，近年来大数据技术越来越受到学术界、工业界和政府的关注[4][5][6][7]，许多国家在技术战略层面提出了一系列大数据研究计划，以推动政府、学术界和工业界探索大数据技术的研究与应用。

大数据给我们带来了巨大的技术挑战，同时也为我们提供了许多巨大的技术创新和商业机会。积累的大数据包含大量深度知识和价值，这是少量数据所不具备的。大数据分析和挖掘将为公司带来巨大的商业价值和各种高附加值的服务，并能够提升公司的经济和社会效益。

大数据时代的传统计算模型无法满足性能和效率的要求，Hadoop[8]、Spark[9]、MapReduce[10]等分布式框架或分布式编程模型应运而生。这些分布式计算框架在互联网业务、科学计算和商业应用中取得了良好的效果，极大地提高了这些应用的效率和效果。

然而，在实际计算环境中，大多数大数据应用难以达到计算机理论的峰值速度，原因可以总结如下：分布式系统的硬件资源未得到充分利用，底层软件栈影响大数据系统的上层，大数据应用未进行优化。这些瓶颈通常只有经验丰富的专家用户才能发现，但需要大量的手动操作，使得这个过程相当繁琐和耗时。对于普通程序员和集群管理员来说，很难理解大数据系统的行为并发现性能瓶颈，从而无法对系统进行优化。而且，由于大数据具有容量、种类、速度和价值等特点，这些大数据应用与传统高性能计算中使用的传统并行程序有很大的不同。传统的性能分析方法无法直接应用于大数据系统，因为大数据系统的任何细节都会变得非常重要，在规模更大的分布式环境中，一次小小的优化就可能带来巨大的性能提升。

978-1-5090-4152-7/16 $31.00 ©2016 IEEE
DOI 10.1109/ISPDC.2016.24

随着数据量的增长，大数据应用对处理速度提升的需求逐渐增加；传统软件性能分析中单一软件栈模型的性能分析复杂性已无法满足诸如 Spark、Hadoop 等大数据系统的性能分析需求。此前已开展了众多专注于性能模型的研究[11][12][13][14]。然而，这些模型大多只包含大数据系统层，并且都是静态模型，无法全面体现大数据系统的性能特征。我们需要一个分层性能模型，能够整合硬件、软件栈、用户应用程序、数据流的性能指标，以便用它来评估系统性能、定位系统性能瓶颈并优化系统。

在本文中，我们为大数据系统提出了一个五级性能评估和优化模型，该模型可用于评估系统性能，并方便地为系统管理员或大数据应用程序员定位性能瓶颈。该模型涵盖了大数据系统的五个级别，包括硬件、架构、操作系统、大数据框架和用户应用程序。模型的每个级别包含几个到几十个性能指标，并且在大数据框架的运行阶段，任意两个级别之间的性能指标是相互关联的。基于这个五级性能模型，我们开发了一个以事件为中心的性能分析器，它可以检测大数据框架中的关键操作，并从作业中的每个任务中分析五级性能指标。该性能模型采用 k 均值聚类算法来定位性能瓶颈所在的作业运行阶段或节点，并使用主成分分析方法辅助瓶颈分析。

本文的主要贡献包括：提出了一种相对完整的大数据系统五级性能评估模型，建立了大数据系统的五级性能优化模型以定位性能瓶颈，开发了一个以事件为中心的性能分析器来分析性能数据并辅助性能分析。基于中国科学院计算技术研究所开发的开源大数据基准套件 BigDataBench [7] 中搜索引擎领域常见的两种工作负载 wordcount 和 sort，我们使用这两个模型和分析器来定位我们集群中的性能瓶颈，这可以使大数据系统的平均运行时间缩短 19%。

本文的结构如下。第二部分简要概述了大数据系统面临的挑战和机遇。第三部分和第四部分分别提出了一个五级性能评估模型和一个五级性能优化模型。第五部分介绍了以事件为中心的分析器的实现。第六部分给出了实验方法和实验结果分析。第七部分讨论了相关工作，第八部分总结了结论和未来的工作。

II. 挑战与机遇

对于传统的关系型数据管理应用和高性能计算应用，业界已经开发出一系列复杂的性能分析方法，但在大数据领域，相关工作并不太实用。大数据应用和大数据系统架构导致在其性能分析方面面临许多具有挑战性的问题：

首先，与高性能计算领域相比，大数据系统的软件堆栈层级更为复杂。大多数大数据系统往往提供更高级别的数据处理接口，例如由 Hadoop 提供的 MapReduce 处理接口，Hive、Shark 等系统提供的类似 SQL 的接口 HQL。应用接口将用户与分布式计算处理细节（数据分布策略、资源）隔离开来，使得用户无需处理数据处理的细节。然而，这种方法会导致系统软件堆栈层级变厚，并给性能分析带来诸多困难。如何在大数据系统的性能分析过程中对各级进行相关性分析是一个关键问题。

其次，大数据系统的数据分布具有多种新特征，数据流不清晰。在关系数据库系统中，数据的分布通常会考虑优化因素；元数据可以帮助用户清晰地了解数据的存储。在大数据系统中，数据存储在分布式文件系统中。例如，由 MapReduce 程序处理的输入数据可以存储在一个大文件中，也可以存储在大量的小文件中，或者介于两者之间的其他情况，因此大数据系统的数据分布没有规律可言。由于数据并行处理是通过一个固定的框架实现的，对于上层应用程序来说，数据流是不透明的，这也给大数据系统的性能分析带来了巨大挑战。

第三，大数据应用通常运行在分布式平台上，这可能是大规模集群或云计算平台。一个大数据应用可能由数千个任务组成；节点规模甚至可以达到数千个。因此，为了了解应用的运行状态，将大量节点的多个级别的性能数据聚合在一起进行性能分析是非常必要的。

最后，为了适应开发社区的环境，大多数大数据框架都是用 Java 语言或 Java 系列语言开发的，所有用这些语言编写的程序都是基于 Java 虚拟机（JVM）的，尽管在跨平台方面有一定优势，但整体性能受 JVM 的性能影响，JVM 的性能直接影响上层大数据应用程序的执行。例如，系统的内存管理受 JVM 的垃圾回收机制的影响，这极大地影响了整个系统的性能，使得系统难以充分利用所有物理资源。

因此，大数据系统的性能分析也需要包含对底层软件堆栈的分析。

III. 性能评估模型

大数据系统的独特性导致其性能模型与传统软件系统有很大差异，传统性能模型无法应用于大数据系统。在大数据系统中，各层相互依赖，不同性能层之间存在复杂的层次关系。因此，需要了解每一层的每一个细节，并且在分析和评估大数据系统性能的过程中，有必要对软件框架、操作系统、硬件等性能数据进行剖析。为了对大数据系统进行更全面的性能分析，提出了一种新颖的五层性能评估模型。

五层性能评估模型包含大数据系统的五层内容，如下所示：（1）用户应用层，主要用于描述用户应用的性能，例如算法、数据结构、代码有效性等；（2）大数据框架层，大数据框架如 Hadoop、Spark、Storm 等都有其性能指标，例如配置、主要阶段时间和数据访问；（3）系统层，性能指标包括操作系统的关键配置和性能参数，以及框架依赖项如 JVM；（4）架构层，用户在此反复检查生成的指令及其微架构瓶颈；（5）硬件层，包含硬件的相关性能参数，例如能耗、网络拓扑、硬件配置。因此，大数据系统的整体性能受到所有节点的硬件、架构、操作系统、所有任务的运行状态以及所有服务大数据框架的影响。

大数据系统的性能评估模型如公式 1 所示。

$$P_{total} = \langle \sum_{i=1}^{n}(P_{app})_i , \sum_{j=1}^{m}(P_{fw})_j , \sum_{k=1}^{p}((P_{os})_k, (P_{arch})_k, (P_{hw})_k) \rangle \quad (1)$$

其中，⊠⊠⊠⊠⊠denotes 大数据系统的整体性能，⊠ ⊠⊠⊠⊠denotes 用户应用程序中第 i 个任务的性能，⊠ ⊠⊠⊠denotes 大数据框架中第 j 个服务的性能，⊠ ⊠⊠⊠⊠⊠ ⊠⊠⊠⊠⊠⊠ ⊠⊠⊠⊠ 分别表示大数据系统中第 k 个节点的操作系统、架构和硬件的性能，n 是用户应用程序中任务的总数量，m 是大数据框架中服务的总数量，p 是大数据系统中的节点总数。

A. 用户应用与数据流

在大数据系统中运行的用户应用程序种类繁多，但大多数大数据应用程序都在大数据框架上运行，这类应用程序是按照该框架指定的特定模式编写的，在这类应用程序中有固定数量的阶段，例如 MapReduce 应用程序有 Map 阶段和 Reduce 阶段，Spark 作业被划分为许多弹性分布式数据集（RDD）。在关键作业运行阶段的时间戳以及每个阶段处理的数据大小都会被记录下来。

大数据系统的核心是数据，在大数据框架中，输入/输出往往会影响整个系统，并且通常是主要的瓶颈。分析每个任务和每个数据处理过程中的数据流可以极大地提高性能分析的可靠性和准确性。在性能模型中，记录了每个子操作的数据源；数据来自缓存、磁盘、内存和网络。

B. 大数据框架

如今，学术界和工业界针对不同类型的大数据应用提出了多种计算框架，例如 Hadoop 适用于离线批处理，Spark 和 Tez 适用于有向无环图（DAG）计算；Storm 适用于流处理等等。尽管这些框架的架构和实现方式有所不同，但它们都基于单指令多数据（SIMD）架构，在这种架构中，一个巨大的数据会被分割成许多部分数据，并且可以在集群中以相同的程序执行方式传递到每个节点。不同的架构有一些相似之处；从各种框架中提取出相似的部分，如任务调度、资源分配管理、框架核心配置、数据访问等等。

C. 系统

一般来说，大多数大数据框架都运行在 Linux 操作系统上，因此 Linux 的性能往往会直接影响大数据系统的性能，甚至一些系统设置也会进行调整以满足特定上层大数据框架的要求。在这一层，性能参数包括与 CPU 相关的指标（用户模式下 CPU 时间、内核模式下 CPU 时间、空闲时间、中断次数等）、与内存相关的指标（空闲大小、缓冲区大小、缓存大小、交换缓存大小、活动大小等）以及与 I/O 相关的指标（磁盘读取时间、磁盘写入时间、字节数、网络数据包计数等）。

像 Hadoop、Spark 等大数据框架需要在 JVM 上运行，因此 JVM 的配置也会影响大数据系统的性能；在此，对 JVM 的一个主要方面——垃圾回收性能进行了监控。

D. 建筑学

分析架构层的性能有助于我们了解程序指令的结构以及缓存的使用情况。并且它能够帮助我们进一步分析大数据框架代码和用户应用代码的效率，这有助于应用程序开发人员提高代码执行的效率，例如减少循环代码和分支，提高代码复用率。

在架构层，主要的性能指标包括 IPC（表示指令执行速度）、各级数据缓存命中率、指令缓存命中率、指令的组成等等。

E. 硬件

集群的基础设施和硬件配置通常会影响大数据系统的性能。大数据系统经常使用分布式文件系统，如DFS、HDFS，对数据流的分析需要对集群的物理架构有全面的了解，例如数据本地性需要满足磁盘本地性、节点本地性、机架本地性、数据中心本地性。数据中心通常由异构集群构成，其中硬件配置与其他节点不同，尽管程序是单指令多数据（SIMD）类型，但整个作业往往有一些串行过程。例如，MapReduce 程序的混洗阶段，整个集群中的数据集将被转移到少数几个 Reduce 节点，此时几个选定节点的网络和磁盘性能将是影响整个作业性能的关键因素。此外，数据中心的能耗也在被考虑，如何在性能评估模型中调整系统以使其低能耗也应得到解决。

F. 每一层的主要性能指标类别

在绩效评估模型中，每一层都包含若干特定的绩效衡量类别，如表 1 所示。

表一：各层的主要性能指标类别

层名称	性能指标类别
用户应用程序	代码结构
	数据访问
大数据框架	配置
	主要阶段时间
系统	中央处理器（CPU）、内存、输入/输出
	Java 虚拟机
架构	缓存
	工业个人计算机
硬件	网络拓扑结构
	功耗

IV. 性能优化模型

基于上述五层性能评估，建立了一个性能优化模型，这有助于我们分析跨级性能，定位大数据系统的性能瓶颈。大数据系统的性能优化模型分为性能检测模型和性能分析模型。

A. 性能检测模型

大多数大数据应用属于单指令多数据（SIMD）类型，不同节点中作业的每个子任务都执行相同的代码，不同节点中代码执行的性能应该相似。然而，由于数据本地性、不合理的节点配置、其他程序干扰等因素，某些任务的执行特征与其他任务不同，这些任务往往会降低整个集群的作业效率，对这些低效任务进行分析和优化可以提高整个作业的效率。

1) 瓶颈任务、瓶颈节点检测方法

为了定位这些关键任务，我们定义每个任务阶段的 DPS（每秒处理的数据量）来量化任务性能，如公式 2 所示。

$$DPS = \frac{Data\ volume(KB)}{Time(s)} \qquad (2)$$

任务的数据处理速度能够反映任务调度时当前节点的性能。在本文中，我们在所有子阶段中将任务处理速度分为最高、较高、中等、较低和最低五个等级。由于瓶颈任务的处理速度与正常处理速度之间存在较大差距，瓶颈任务的处理速度在聚类分析中是离群点。我们选择 PAM（围绕中心点的划分）算法，一种 k 中心点聚类算法，来进行聚类分析，因为它具有很强的稳健性。

大数据系统的性能分析模型专注于找出瓶颈任务和瓶颈节点。根据 80/20 原则，80%的性能瓶颈存在于 20%的任务中，处于最低聚类的任务会被标记出来，这些任务是影响整个系统性能的性能瓶颈任务。然后，标记出任务处于较低聚类的前 20%的节点，这些节点是瓶颈节点。性能瓶颈检测算法在算法 1 中有描述。

Algorithm 1: Performance bottleneck detection algorithm 1

Input:

PD: 5-layer performance data sets.

Output:

BN: Sets of bottleneck tasks, bottleneck nodes

Method:

(1) ***For*** every task

(2) Calculate the ***DPS***;

(3) ***End for***

(3) ***For each*** task type (Map or Reduce) ***T do***

(4) Use ***PAM*** clustering algorithm to get the ***DPS*** grade of T;

(5) Mark the tasks with lowest ***DPS*** grade as bottleneck tasks;

(6) Select top 20% of nodes which has lowest ***DPS*** grade or lower ***DPS*** grade task, as bottleneck nodes;

(7)***End for***

2) 子阶段瓶颈检测方法

众所周知，MapReduce 分为 Map 和 Reduce 阶段。Map 包括读取、映射、收集、溢出、合并这五个子阶段。Reduce 包括洗牌、合并、排序、归约、写入这五个子阶段。当我们发现瓶颈节点和瓶颈任务时，需要进一步确定导致瓶颈的子阶段。

大数据系统的软件栈包含多个复杂的层次结构，性能瓶颈往往受到多层软件栈的影响。为了便于检测性能瓶颈，采用主成分分析（PCA）方法进行跨层性能分析。

主成分分析（PCA）是一种数学变换方法，它可以通过线性变换将一组相关变量转换为另一组不相关的变量，并使用更少的变量来解释大部分原始变量，将众多变量之间的高度相关性转化为独立或非相关变量。所选变量的数量通常少于原始变量的数量，几个新变量可以解释原始变量中的大部分信息，这被称为主成分，它能够解释原始数据的综合情况。

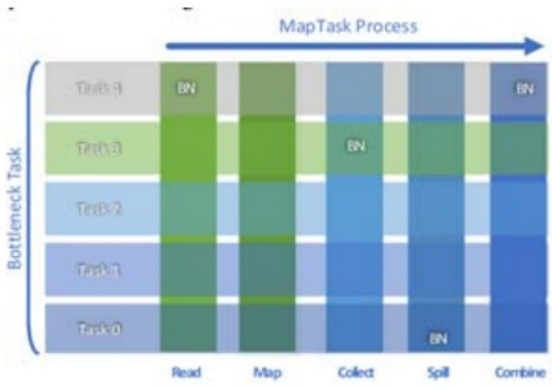

图 1 。子阶段瓶颈检测

收集到的性能数据根据大数据框架的执行阶段被划分为多个部分，如图 1 所示。存在读取、映射、收集、溢出、合并阶段。通过使用主成分分析对性能检测模型检测到的性能瓶颈进行分析，图 1 中标记为“BN”的区域是特定节点中的瓶颈阶段。接下来，在这些瓶颈中构建关于 IPC 的相关系数矩阵，获取系数的绝对值，并选择总体系数值的顶部 80%，这些指标是影响该阶段性能的主要成分。根据主要成分的构成，可以找出导致性能瓶颈的核心原因。性能瓶颈分析算法如算法 2 所述。

Algorithm 2: Performance bottleneck detection algorithm 2
Input:
PD: 5-layer performance data sets, BN_t:bottleneck tasks
Output:
BN_p: Bottleneck sub-phases
Method:
(1) ***For each*** bottleneck task BN_t ***do***
(2) Calculate DPS_p for every phase of BN_t
(3) Calculate DPS_t of BN_t
(4) Use PCA calculate correlation coefficient matrix with DPS_p relative to DPS_t
(5) Sorting the correlation coefficient and mark the sub-phases that corresponding coefficient on top 20% as bottleneck sub-phases.
(6) ***End for***

B. 性能分析模型

在定位到性能瓶颈之后，有必要进一步优化大数据系统的性能，因此我们需要一个性能分析模型来协助瓶颈优化。大数据系统的软件栈包含多个复杂的层次结构，性能瓶颈往往受到多层软件栈的影响。为了便于对性能瓶颈进行优化，还采用了主成分分析（PCA）方法来进行跨层的性能分析。

IPC（每周期指令数）能够反映出系统在一个 CPU 周期内处理了多少条指令，可用于反映系统在特定阶段的整体性能。

性能瓶颈分析算法如算法 3 所述。

Algorithm 3: Performance bottleneck analysis algorithm
Input:
PD: 5-layer performance data sets,
BN_{st}:Bottleneck nodes, bottleneck tasks, bottleneck sub-phase
Output: Bottleneck reason
Method:
(1)Normalize all performance metrics in each layer;
(2)Calculate ***IPC*** for every phase
(3)***For each*** bottleneck task BN ***do***
(4) Use PCA to calculate correlation coefficient for every performance metric relative to IPC
(5) Sorting the correlation coefficient and mark the coefficient of taking sum of 80% as bottleneck reason.
(6) ***End for***

V. 以事件为中心的分析器

A. 分析器概述

我们为大数据系统（Hadoop 生态系统）设计并实现了一个名为事件中心分析器（简称 ECP）的集成性能工具，其特点包括：

1. 在收集性能数据时，用户应用程序无需修改软件栈。可以抓取 MapReduce 作业在每个任务的所有阶段的运行时间、数据流、系统和架构级别的性能数据；

2. 设计了一种统一的数据格式和协议，可用于收集每个节点的多层性能数据；

3. 提供了一个直观且友好的可视化界面，以帮助用户发现性能瓶颈，然后进行性能优化。

每个节点的 ECP 通过动态检测技术在软件堆栈的所有级别收集性能数据；然后将所有节点的所有性能数据聚合、统计处理并保存到数据库中；最后通过网络浏览器展示所有性能数据。ECP 由三个模块组成：动态信息分析模块、数据收集模块和图形显示模块。

B. 动态信息分析模块

为了收集每个节点的用户应用层、大数据框架层、操作系统层和架构层的性能数据，在每个节点上部署了动态信息分析模块，该模块收集每个节点的相关配置文件和日志文件，并控制每个节点的收集行为。性能数据以统一格式和协议传输到分析处理系统，最终可以实现相应的统计结果，整个过程

动态信息收集如图 2 所示。

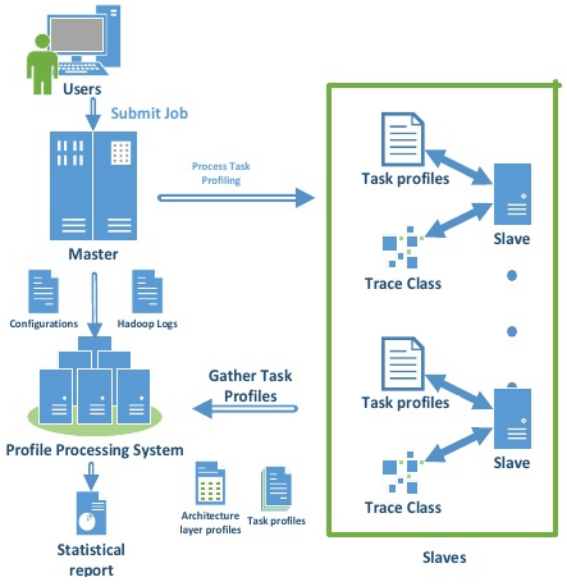

图 2 。动态信息收集流程

C. 数据采集模块

数据采集模块基于五层性能模型，在不同层采用了不同的配置文件方法。
如图 3 所示。

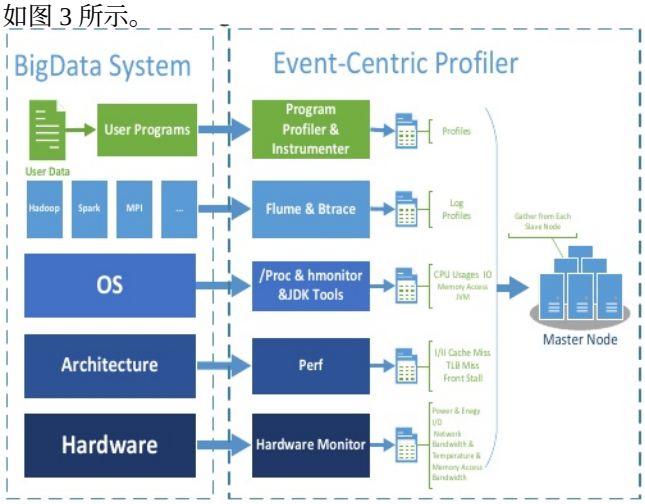

图 3 。多层性能数据采集

1) 用户应用层的性能数据收集

代码区域被视为一段代码，它可以独立执行，除了在代码的入口和出口处没有跳转。代码区域可以是一个函数、循环、子流程或其他基本块。如果一个代码区域被归类为非常严重或严重程度类别之一，则它被视为关键代码区域（CCR）。

为了剖析用户程序的性能数据，一些特定的代码被插入到用户程序中，插入位置可以在每个关键代码区域之前和之后。这些特定代码能够被程序剖析器/插入器识别，然后就可以收集用户程序的性能数据。

2) 大数据框架层的性能数据收集

大数据框架层的性能数据由日志信息和 Hadoop 生态系统的配置组成，这些数据由 BTrace [15] 进行剖析。

Hadoop 作业跟踪器中的日志信息主要包含各种级别的 Hadoop 计数器数据、任务分配信息、任务性能等数据。Hadoop 配置中存储的数据指定了日志目录，作业完成后，日志文件会通过 HDFS API 复制到本地文件系统。

BTrace 的动态检测主要用于获取 MapReduce 实现时间线中关键步骤的所有时间戳以及与任务相关的信息，例如数据大小、起始时间、溢出、混洗、合并和排序阶段的结束时间。这些数据被存储在 MapReduce 分析中，并且在测试作业完成时将通过 HDFS API 复制到本地文件系统。

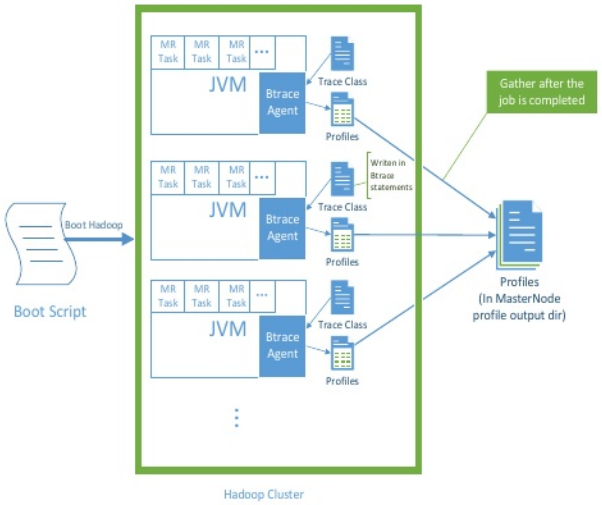

图 4. 在 Hadoop 集群中的跟踪代码执行

ECP 建立在 MapReduce 配置文件收集机制之上，并使用 BTrace 来处理 CCR 检测。如图 4 所示。BTrace 代理受到 Java 代理的启发，然后 BTrace 代理会将跟踪代码插入到代码的执行路径中。

3) 操作系统层的性能数据收集

操作系统层的性能数据主要从 Linux 的 proc 虚拟文件系统中收集。同时，使用 hmonitor 脚本以特定间隔获取并保存指定目录中每个节点的配置，然后在作业完成后使用 scp 将配置文件目录复制到主节点。

操作系统层的性能数据还包括 JVM 的性能数据。对于每个 JVM，都会收集主要函数调用信息和垃圾回收（GC）信息。表二展示了 JVM 与垃圾回收相关的信息。

表二. JVM 的 GC 相关信息

列	描述
S0C	当前幸存者空间容量为 0（千字节）。
S1C	当前幸存者空间 1 的容量（千字节）
S0U	幸存者空间 0 利用率（千字节）
S1U	幸存者空间 1 的利用率（千字节）
欧洲联盟	当前伊甸空间容量（千字节）
欧盟	伊甸空间利用率（千字节）
客观	当前旧空间容量（千字节）
哦	旧的空间利用率（千字节）
个人电脑	当前永久性空间容量（千字节）
聚氨酯（PU）	永久性空间利用率（千字节）
年轻一代	年轻一代 GC 活动的数量
预计到达时间	年轻一代垃圾收集时间。
首次游戏成本	完全垃圾回收事件的数量。
快速增长型公司测试	垃圾全部收集时间。
全球气候变暖	垃圾收集的总时间。

4) 架构层的性能数据收集

架构层中的性能数据由位于性能工具中的一个 Python 脚本收集，该脚本在每个节点运行，并在作业启动时开始收集性能数据，脚本将性能数据文件输出到配置文件中指定的目录，例如操作系统层，在作业结束时，使用 scp 将性能数据文件复制到主节点中指定的目录。

5) 硬件层的性能数据收集

每个节点的硬件信息由用 Python 编写的自动收集脚本收集，网络拓扑结构通过手动方式收集，而功耗数据则通过精确的功率表收集。

VI. 实验

我们使用大数据基准测试来评估所提出的性能评估和优化模型，使用以事件为中心的分析器来分析大数据系统不同层的性能数据。

A. 基准测试和工作负载

1) 大数据基准测试

BigDataBench 是由中国科学院计算技术研究所开发的一个开源大数据基准测试套件[7]。本质上，BigDataBench 是一个针对横向扩展工作负载的基准测试套件，与 SPEC CPU 工作负载（顺序工作负载）和 PARSEC 工作负载（多线程工作负载）不同。目前，BigDataBench 对五个典型且重要的大数据应用领域进行了建模：搜索引擎、社交网络、电子商务、多媒体分析和生物信息学。在指定具有代表性的大数据工作负载时，BigDataBench 专注于每个应用领域中经常存在的 OLTP、Cloud OLTP、OLAP、交互式和离线分析中的计算单元。同时，BigDataBench 考虑了各种具有不同类型和语义的数据模型，这些模型是从真实世界的数据集中提取的，包括非结构化、半结构和结构化数据。BigDataBench 还提供了一个端到端的应用基准测试框架，通过抽象数据操作和工作负载模式来创建灵活的基准测试场景，这些场景可以扩展到其他应用领域，例如用水量应用领域。

2) 工作负载

实验中采用了来自大数据基准测试（BigDataBench）的搜索引擎领域中包括词频统计和排序在内的两种常见工作负载。输入数据量遵循大型电子商务网站淘宝的工作负载特征报告中报道的分布[16]。表3展示了在我们的实验中不同输入数据量的百分比。

表三：工作负载中不同的输入数据量

输入数据量	百分比
<25兆字节	40.57%
25兆字节 - 625兆字节	39.33%
1.2 亿至 5 亿	12.03%
>5 千兆字节	8.07%

B. 测试台

为了评估大数据系统的性能，我们部署了一个由5个节点（一个主节点和四个从节点）组成的Hadoop集群。每个节点都具有相同的硬件配置：英特尔Xeon E5310 CPU、4GB内存和4TB磁盘。这些节点通过千兆交换机连接。每个节点还使用相同的软件栈：CentOS 5.5与内核2.6.34、Hadoop 1.2.1。表4列出了Hadoop集群的详细参数配置。

表四：Hadoop 集群的配置

中央处理器（CPU）类型		英特尔 CPU 内核	
英特尔®至强 E5310		4 核@1.6 千兆赫	
一级指令/数据缓存	二级缓存	内存	磁盘
32千字节	4兆字节	4 千兆字节	4 太字节
操作系统	CentOS 5.5	哈达普	1.2.1

C. 性能瓶颈检测

我们运行了第 6.1.2 节中描述的综合工作负载，并使用以事件为中心的分析器在大数据系统的 5 个不同层级收集性能数据，然后在作业完成后根据 MapReduce 阶段将 5 层性能数据划分为几个部分，采用性能优化模型分别处理性能数据的每个部分。Map 任务有五个阶段，包括将数据读入内存的读取阶段、在用户程序中执行 Map 函数的 Map 阶段、收集 Map 阶段输出的键值对的收集阶段、当循环缓冲区满时运行的溢出阶段、将临时输出文件合并为一个文件的合并阶段。Reduce 任务也有五个阶段，包括将 Map 任务中合并的数据分发到 Reduce 任务的洗牌阶段、合并洗牌数据的合并阶段、收集具有相同键的值的排序阶段、在用户程序中执行 Reduce 函数的 Reduce 阶段、将最终结果写入磁盘的写入阶段。

每个任务中的读取阶段主要负责从 HDFS 读取数据，通过分析器工具收集的总读取时间和读取数据量大小用于计算每个任务中的读取速度。然后使用 k 均值聚类算法实现 5 个聚类，聚类结果如图 5 所示。

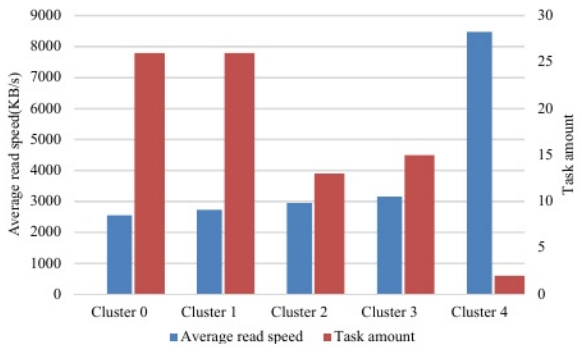

图 5 。Map 任务中读取阶段性能数据的聚类结果

然后选择“集群 0”，即最小的集群，在图 6 中标记此集群中的所有任务，我们可以发现 GD41 节点包含最慢的任务，显然 GD41 节点是读取阶段的瓶颈节点。

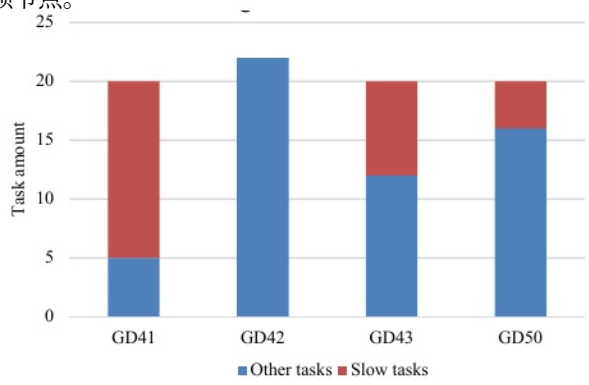

图 6. 不同节点中不同任务的数量

在溢出阶段，我们还使用类似的方法来分析性能数据，计算排序和溢出的数据处理速度，并且使用 k 均值算法将记录聚类为五个簇，聚类结果如图 7 所示。

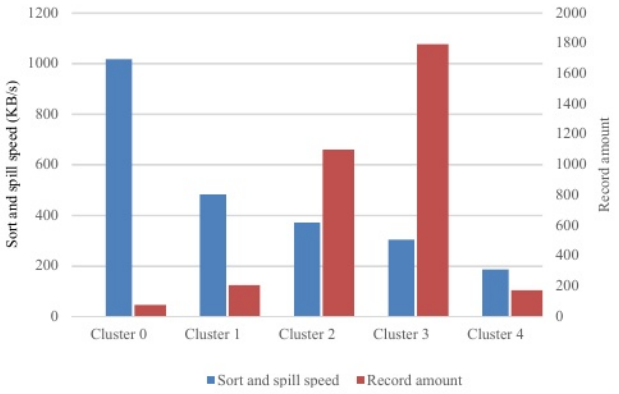

图 7 。溢出阶段性能数据的聚类结果

然后，计算最慢的集群 4 中每个节点的记录数量，结果如图 8 所示，我们可以发现集群 4 中的大多数记录都位于 GD43 节点，显然 GD43 节点是溢出阶段的瓶颈节点。

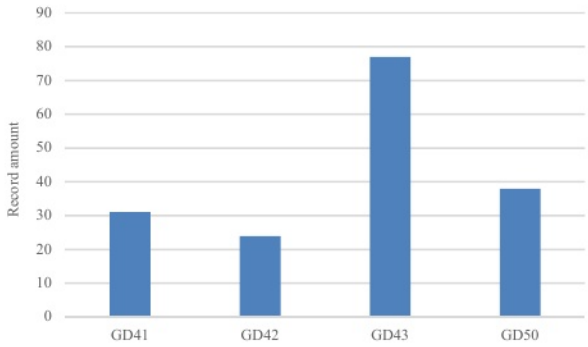

图 8. 不同节点的记录金额

D. 原因分析

采用主成分分析方法辅助瓶颈原因分析。首先，从架构层和操作系统层收集 Map 任务中读取阶段的性能数据，然后去除常量数据，对所有性能数据进行归一化处理，计算每个阶段的 IPC，并使用主成分分析法得出每个指标与 IPC 的相关系数。根据 80/20 原则，我们选择操作系统层中总体系数值的前 80%，如表 3 所示。

表五. OS 层中的相关系数

度量	系数	度量	系数
闲置	0.68735	流程	0.541811
rpackets	0.672522	pgout	0.460739
秒字节	0.669911	写入扇区	0.460739
用户	0.587909	写入合并	0.453033
运行	0.571993	读取字节数	0.387817
中断	0.568738	免费	0.358849
括号	0.557178	写入时间	0.273462

从表 5 中，我们可以看到网络流量相关指标，如 rpackets（接收数据包数）、sbytes（发送字节数）、spackets（发送数据包数）、rbytes（接收字节数），对 Map 任务的读取性能有很大影响，Map 任务的数据读取应尽量确保其局部性，也就是说，大部分数据应在本地磁盘处理，尽可能避免或减少网络传输。我们在 GD41 节点统计数据 I/O，发现 31.3%的数据 I/O 是网络 I/O，这证实了上述发现。溢出阶段负责将 Map 任务的输出数据存储到本地临时文件中，这非常依赖 CPU 计算和磁盘 I/O；应首先通过 CPU 测试消除 CPU 检测，然后检查监控记录以判断是否存在其他运行中的程序，如果没有，最后分析磁盘 I/O。然后我们在 GD43 节点测试硬盘，发现写入速度为 115.35 MB/秒，而其他节点的平均写入速度为 147.74MB/秒，比 GD43 节点快约 22%。因此，我们可以得出结论，磁盘的写入速度对溢出的速度有很大影响。

E. 性能优化

为了优化 Hadoop 系统的性能，我们尝试通过用公平延迟调度器替换原始的先进先出调度器来减少网络 I/O。公平延迟调度器优先调度具有数据本地性的任务，这可以大大降低跨节点 HDFS 读取的成本；此外，在 GD43 节点中，我们将过度使用的磁盘替换为新的磁盘。然后我们使用相同的工作负载对优化后的系统进行测试，数据处理速度如图 9 所示。

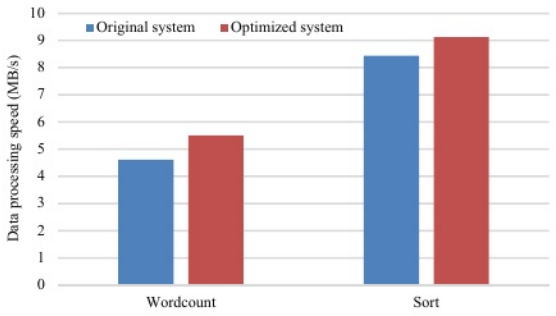

图 9. 原系统和优化系统的数据处理速度

VII. 相关工作

此前，针对大数据系统性能模型的研究已经开展。参考文献[11]提出了一种向量式成本模型，并给出了计算每个成本项目的公式。此外，还提供了基于成本向量估算任务执行时间的公式。参考文献[12]提出了一种简单通用的MapReduce性能模型，以更好地理解每个组件对整体程序性能的影响，该模型可以预测MapReduce系统的性能及其与配置的关系。参考文献[18]确定了一个适用于MPI性能特征化的性能模型，并比较了几个平台上MPI通信的性能。参考文献[19]描述了在准确、可重复测量消息传递性能方面固有的困难。参考文献[14]使用统计模型来预测云计算应用程序的资源需求。这样的预测框架可以指导系统设计和部署决策，如规模、调度和容量。参考文献[13]确定了影响Hadoop性能的五个设计因素，并研究了针对每个因素的替代但已知的方法。结果表明，通过仔细调整这些因素，可以提高 Hadoop 的整体性能。然而，这些工作中的模型均为静态模型，并且仅处理了一到两层大数据系统，无法全面体现大数据系统的性能特征。

性能数据分析器也得到了广泛的研究。参考文献[20]介绍了一种名为Manimal的工具，它可以自动分析MapReduce程序并应用适当的数据感知优化，从而不需要程序员的额外帮助。它还可以检测到一系列数据操作中的优化机会，并加速之前编写的MapReduce程序。Mochi[21]是一种新的可视化、基于日志分析的调试工具，它可以关联Hadoop在空间、时间和容量方面的行为，并提取一个因果的、统一的控制和数据流模型，该模型覆盖了集群中的所有节点。Mochi的分析使用户能够推理和调试性能问题，并生成Hadoop行为的可视化结果。PerfXplain[22]是一个系统，它允许用户询问关于成对MapReduce作业的相对性能（即运行时间）的问题，它提供了一种新的查询语言来表达性能查询，并提供了相应的算法。

从过去的 MapReduce 作业执行日志中生成解释。大多数此类工具收集的性能指标并不全面，而且有几种工具仍然需要修改用户集群的软件栈。

VIII. 结论与未来工作

在本文中，我们为大数据系统提出了一个五层性能评估和优化模型，该模型可用于评估系统性能，并方便地为系统管理员或大数据应用程序员定位性能瓶颈。该模型涵盖了大数据系统的五个层面，包括硬件、架构、操作系统、大数据框架、用户应用程序，并且在大数据框架的运行阶段，任意两个层面之间的性能指标是相互关联的。基于这个五层性能模型，我们开发了一个以事件为中心的性能分析器，它能够在大数据框架中检测关键操作，并从作业中的每个任务中分析五层性能指标。在实验评估中，采用 k 均值聚类算法在作业运行阶段或节点定位性能瓶颈，并采用主成分分析方法辅助瓶颈分析。实验结果表明，性能评估和优化模型对于定位性能瓶颈、分析瓶颈成因以及优化大数据系统的性能是有效的，能够使大数据系统的平均运行时间提高 19%。

未来，我们将继续改进性能评估和优化模型，包括在每一层增加更多的性能指标、引入粗糙集来辅助性能分析等等。此外，我们还将改进性能分析工具，使其能够支持更多的大数据框架。

鸣谢

我们的工作部分得到了中国国家自然科学基金委员会（项目编号：61372171）和计算机体系结构国家重点实验室开放项目（项目编号：CARCH201409）的支持，中国科学院计算技术研究所。

参考文献

[1] J. Gantz, D. Reinsel, “The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the Far East,” IDC Report, 2012.

[2] BAEDetica, “The big data refinery: Distillin intelligence from big data, ” https://www. baesystemsdetica. com/uploads/rsources/Big_ Data_ Refinery_Whitepaper_single_pages_19.06.12.pdf, July 2012.

[3] IDC, “The digital universe in 2020: A universe of opportunities and challenges,” http://www.emc.com/leadership/digital-universe/ iview/executive-summary-a-universe-of.htm, 2012.

[4] J. Zhan, L. Wang, X. Li, W. Shi, C. Weng, W. Zhang, and X. Zhang, “Cost-aware cooperative resource provisioning for heterogeneous workloads in data centers” IEEE Transactions on Computers, vol. 62, pp. 2155-2168, 2013.

[5] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Job scheduling for multi-user MapReduce clusters,” Technical Report UCB/EECS-2009-55, EECS Department, University of California at Berkeley, April 2009.

[6] L. Wang, J. Zhan, W. Shi, and Y. Liang, “In cloud, can scientific communities benefit from the economies of scale?” IEEE Transactions on Parallel and Distributed Systems, vol. 23, pp. 296-303, February 2012

[7] W. Gao, Y. Zhu, Z. Jia, C. Luo, L. Wang, Z. Li, et al., “Bigdatabench: a big data benchmark suite from web search engines,” Proc. the 3rd Workshop on Architectures and Systems for Big Data (ASBD 2013) in conjunction with the 40th International Symposium on Computer Architecture (ISCA 2013), arXiv preprint, May 2013, arXiv:1307.0320.

[8] Apache™, “Hadoop,” http://hadoop.apache.org.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: cluster computing with working sets” Proc. the 2nd USENIX Conference on Hot Topics in Cloud Computing, pp. 1765-1773, June 2010.

[10] J. Dean, and S. Ghemawat, “MapReduce: simplified data processing on large clusters” Communications of the ACM, vol. 51, pp. 107-113, 2008.

[11] X. Lin, Z. Meng, C. Xu, and M.Wang, “A practical performance model for hadoop mapreduce” , Proc. IEEE International Conference on CLUSTER Computing Workshops, pp. 231-239, September 2012.

[12] X. Yang, and J. Sun, “An analytical performance model of mapreduce”, Proc. IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), pp. 306-310, September 2011.

[13] D. Jiang, B. C. Ooi, L. Shi, and S.Wu. “The performance of MapReduce: an in-depth study” Proc. the VLDB Endowment,vol. 3 No. 2, pp. 472-483, 2010.

[14] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, “Statistics-driven workload modeling for the cloud”, Proc. IEEE 26th International Conference on Data Engineering Workshops (ICDEW), pp. 87-92, March 2010.

[15] A. Lal, J. Lim, M. Polishchuk, and B. Liblit. “BTRACE: Path optimization for debugging,” Technical Report 1535, University of Wisconsin-Madison, 2005.

[16] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou. “Workload characterization on a production Hadoop cluster: a case study on Taobao,” Proc. IEEE International Symposium on Workload Characterization (IISWC2012), pp. 3-13, November 2012.

[17] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling,” Proc. the 5th European Conference on Computer Systems (EuroSys 2010), pp. 265−278, April 2010.

[18] K. Al-Tawil, and C. A.Moritz. “Performance modeling and evaluation of MPI,” Journal of Parallel and Distributed Computing, vol. 61, pp. 202-223, 2001.

[19] W. Gropp, and E. Lusk. “Reproducible measurements of MPI performance characteristics,” Proc. Recent Advances in Parallel Virtual Machine and Message Passing Interface, pp. 11-18, September 1999.

[20] E. Jahani,M. J. Cafarella, and C.Ré. “Automatic optimization for MapReduce programs,” Proc. the VLDB Endowment, vol. 4 No. 6, pp. 385-396, 2011.

[21] J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan. Mochi: visual log-analysis based tools for debugging hadoop”, Proc. USENIX Workshop on Hot Topics in Cloud Computing (HotCloud), June 2009.

[22] N. Khoussainova, M. Balazinska, and D. Suciu, “Perfxplain: debugging mapreduce job performance,” Proc. the VLDB Endowment, vol. 5 No. 7, pp. 598-609,2012.