·先进计算· 文章编号:1000-3428(2016)01-0001-06 文献标识码:A 中图分类号:TP301

Hadoop 节点资源参数优化策略

曾婉琳,陈兴蜀,罗永刚

(四川大学计算机学院,成都 610065)

摘 要:针对 Hadoop 平台的节点资源优化问题,提出 MapReduce 参数优化策略。获取新作业执行时的资源占用特征值,计算其与作业特征库中作业的相对距离,选择相对距离最小作业的配置作为新作业的最优配置,如果获取失败,则以迭代方式获取新作业的最优配置并更新作业特征库。实验结果表明,与默认参数配置相比,该策略能够提高作业执行效率,缩短作业运行时间。

关键词:Hadoop 集群;MapReduce 框架;参数优化;资源利用率;执行效率;特征库;相对距离

中文引用格式:曾婉琳,陈兴蜀,罗永刚. Hadoop 节点资源参数优化策略[J]. 计算机工程,2016,42(1):1-6.

英文引用格式:Zeng Wanlin,Chen Xingshu,Luo Yonggang. Node Resource Parameter Optimization Strategy for Hadoop[J]. Computer Engineering,2016,42(1):1-6.

Node Resource Parameter Optimization Strategy for Hadoop

ZENG Wanlin,CHEN Xingshu,LUO Yonggang

(College of Computer Science,Sichuan University,Chengdu 610065,China)

【Abstract】 For solving the problem of node resources optimization in Hadoop platform, this paper proposes a MapReduce parameter optimization strategy. When a new job is submitted, it first gets feature value of resource utilization, and then calculates the relative distance with the jobs in the signature database. At last, it selects the configuration of the job with the minimum relative distance as the optimal configuration. If the configuration is not found, it gets optimal configuration by the way of iteration and then updates the feature database. Experimental results show that the proposed strategy can effectively improve the efficiency of job execution and reduce the execution time compared with the default parameter configuration.

【Key words】 Hadoop cluster; MapReduce framework; parameter optimization; resource utilization rate; execution efficiency; feature database; relative distance

DOI:10.3969/j.issn.1000-3428.2016.01.001

1 概述

近年来,随着互联网的普及和 Web 技术的飞速发展,全球数据呈现爆炸式增长,使得大数据处理成为一个新的研究热点[1]。Hadoop[2]是 Apache 基金会开发的一个基于 MapReduce 编程模型的开源框架,在 Web 搜索、数据挖掘以及科学计算等大规模数据处理方面得到了广泛的应用。

目前,许多公司和企业都使用 Hadoop 进行大规模数据处理,将 Hadoop 与自身产品结合部署,形成各种服务于具体业务应用的大数据处理系统,如网页索引、数据挖掘、机器学习等,或者通过与虚拟技术结合,构建基于 Hadoop 系统的大规模数据处理平台,统一化地以服务的形式提供给用户使用,如百度云平台[3]、Amazon 云平台[4]等。这些平台以租赁的方式提供用户所需的 IT 资源,按需计费。许多中小型企业或者个人团体,为节省成本开支,则通过租赁的方式获取基础设施、软件平台以及软件系统等资源。

在按需计费的平台上,用户可以在云平台上随意申请集群运行作业,当作业完成后可以立即结束使用,并且用户只需要支付所使用资源的费用。因此,如何在最短时间内租用最少的资源完成作业,即如何最大化节省开支并最快速完成作业成为用户关心的问题。因此,针对 Hadoop 集群作业性能优化的问题就成为了一个研究热点。

基金项目:国家自然科学基金资助项目(61272447);国家科技支撑计划基金资助项目(2012BAH18B05)。

作者简介:曾婉琳(1990-),女,硕士研究生,主研方向为云计算;陈兴蜀,教授、博士、博士生导师;罗永刚,博士研究生。

收稿日期:2014-12-11 **修回日期**:2015-02-04 **E-mail**:chenxsh@scu.edu.cn

文献[5]从作业调度优化的角度提出了一种针对异构集群的作业调度算法,达到优化集群性能的目的。文献[6-7]从工作流的角度出发,提出了一种优化的 Hadoop 工作流框架。文献[8]根据对 Hadoop 工作流和数据流的分析,针对不同作业的执行特征提出关于 Hadoop 集群节点数目的优化方案。在这些优化方案中,需要对 Hadoop 中 MapReduce 框架进行了解和分析,但是随着 Hadoop 平台用户的增加,非专业用户比例也在增加,他们缺乏相关的专业知识,更需要一种简单易操作的优化方案。因此,如何通过 Hadoop 参数配置来提升 Hadoop 性能成为了许多研究者关注的焦点。

在 Intel 公司技术白皮书中[9],作者指出可以调整多个层面的不同参数来提高作业的运行效率,从而优化集群性能,但是未给出具体的参数优化方案。已有的参数配置方案[10-11]简单地采用 Hadoop 默认参数来自动部署作业,然而不同的作业对不同类型资源(CPU,I/O 和内存等)的需求程度不同,因此,在一定的硬件环境条件下,对于不同的作业,默认参数未必都能使其达到资源利用最大化且运行时间最短。

文献[12]基于 MapReduce 的工作流特征,通过对 Hadoop job 的部分配置参数调优而优化 job 性能,但是调优算法仅针对于 Nutch 爬行性能的优化,优化方案具有局限性。

合理的配置能够协调 Hadoop 作业运行的各个环节,使得集群的各种资源得到充分利用,从而提高集群的资源利用率,优化集群的整体性能,提高作业的执行效率。在 Hadoop 中有超过 180 个的相关配置参数,如输入数据的副本数、可并行执行最大的 Map/Reduce 任务数等。目前 Hadoop 参数配置方案存在以下问题:(1)多数采用默认配置,不能使所有的作业类型都有较好的运行性能,并且集群的资源利用率较低;(2)根据经验修改参数值,但是基于经验的配置往往是对已经运行过的作业具有较好的性能提高能力,针对陌生作业的优化有限;(3)针对特定的作业类型建立优化模型,这样的优化方案较为精准但却具有局限性,一旦作业类型发生改变,优化方案则失效。

针对上述问题,本文提出一种新的节点参数配置优化策略,主要研究 mapred. tasktracker. map. tasks. maximum 和 mapred. tasktracker. reduce. tasks. maximum 这两个参数(分别用 *MAX_MAPPERS* 和 *MAX_REDUCERS* 表示)对集群的资源利用率以及作业执行效率的影响,并给出相关调优策略。

2 问题描述

对于一个典型的 Hadoop MapReduce 作业,一般包括 3 个阶段:Map,Copy 和 Reduce。在作业提交后,默认情况下输入文件将被分成 64 MB 大小的分片进行处理。在 Map 阶段,主要负责对每个块进行处理,产生键值对并存放到本地文件中;在 Copy 阶段,主要负责将 Map 阶段产生的键值对传递给 Reduce,作为输入;在 Reduce 阶段,会调用自定义的函数来处理这些键值对,并产生输出结果。具体执行过程由 Map Task 和 Reduce Task 完成。

图 1 为 Hadoop MapReduce 工作流程,可以看出,如果能够适当提高 Map 阶段以及 Reduce 阶段的并发率,则可以提高整个作业的执行效率。

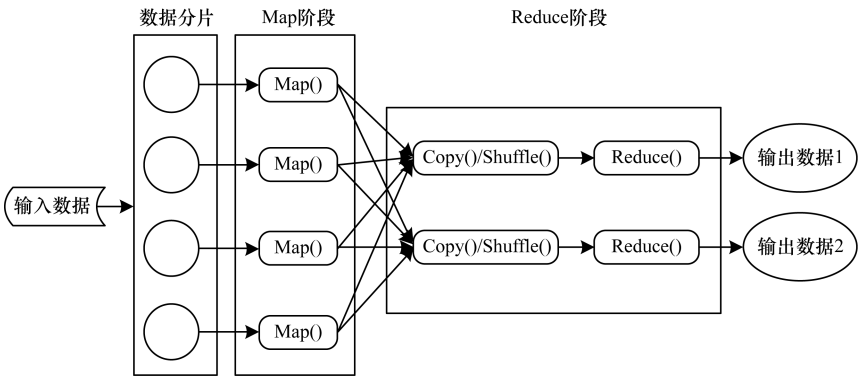

图 1 Hadoop MapReduce 工作流程

MAX_MAPPERS 是指单个节点上能并行执行的最大 Map 任务数,*MAX_REDUCERS* 是指单个节点上能并行执行的最大 Reduce 任务数。在 Hadoop 的配置参数中,*MAX_MAPPERS* 和 *MAX_REDUCERS* 是对 CPU 的利用率影响最大的 2 个参数[13]。在每个节点上,一个 Map/Reduce 任务作为单独的线程运行,这两个参数的值如果设置过大,会导致线程之间争夺资源,从而降低作业执行效率。但如果设置太小,又会导致资源浪费,使得作业执行效率无法达到最优。因此,需要合理地设置这 2 个参数来提高资

源利用率,从而提高作业的执行效率。

为验证这两个参数是否能够影响 Hadoop 作业的执行效率,笔者在一个具有 3 个节点的 Hadoop 集群上进行以下测试:(1)Grep 作业,输入数据大小为 3 GB;(2)WordCount 作业,输入数据大小为 3 GB。

图 2 和图 3 分别是 Grep 作业和 WordCount 作业的执行时间随 *MAX_MAPPERS*(默认值为 2)、*MAX_REDUCERS*(默认值为 2)的变化曲线。

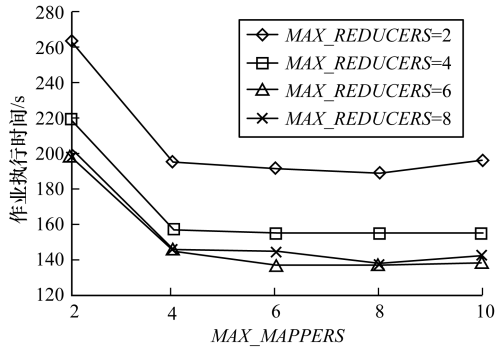

图 2　Grep 作业执行时间受参数的影响

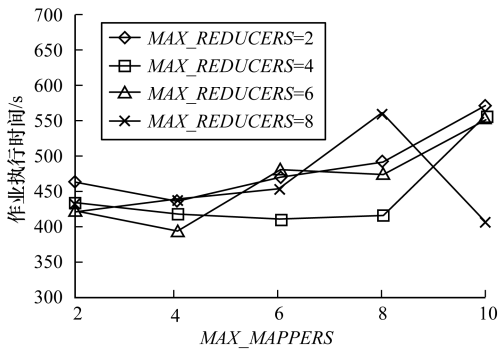

图 3　WordCount 作业执行时间受参数的影响

从图 2 和图 3 可以明显看出,在默认配置参数的情况下,2 个作业的执行时间都不是最短。由于不同的作业对不同资源的需求程度不同,因此不同的作业会有不同的最优参数配置。本文研究的问题描述如下:对于一个 MapReduce 作业 J,输入数据为 $D(size(D)>1\ \text{GB})$,假设资源为 R,作业的执行时间为 T,需要找到一种配置 $Conf(MAX\_MAPPERS, MAX\_REDUCERS)$ 最大化利用资源 R,并且使得作业 J 的执行时间 T 最短。

3　优化策略

图 4 为优化策略流程,首先根据典型作业的资源占用情况以及执行时间建立作业特征库,对新的作业进行预处理,通过 Python 脚本获取资源占用情况以及执行时间,并与特征库进行对比,根据优化策略获取该作业的最优配置。

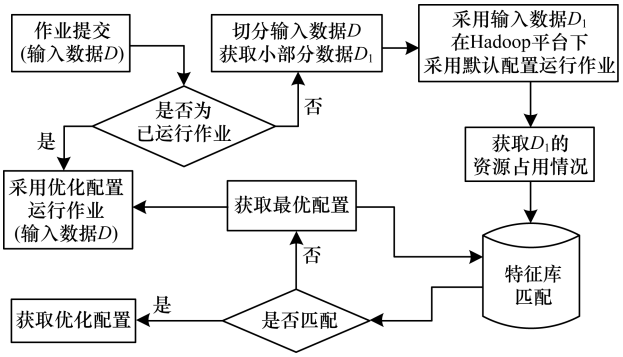

图 4　优化策略流程

Hadoop 作业在初始阶段会将输入数据 $D(size(D)>1\ \text{GB})$ 分为多个 64 MB(默认值)的数据块。对于一个新的作业 J,输入数据为 $D(size(D)>1\ \text{GB})$。在预处理阶段,首先需要拷贝输入数据 D 中的多个数据块,得到数据 $D_1(size(D_1)=1\ \text{GB})$。

2 个作业如果在执行过程中有比较相似的资源占用情况,那么它们会面临相似的资源瓶颈[14]和相似的资源特征比例(CPU、磁盘和网络),也就是说,它们在最优执行效率时的参数配置也会很接近。设置合适的 *MAX_MAPPERS* 和 *MAX_REDUCERS* 取值能够使得节点资源最大化利用,避免资源浪费的情况出现,从而提高集群单个节点上任务执行的并发性,达到优化集群性能的目的。

(1)获取作业资源占用特征值

一个 Hadoop 作业通常分为 3 个阶段:Map,Copy 和 Reduce。正常来说,资源占用特征值应针对每个不同的阶段建立,对比不同作业相同的阶段下的资源特征值,从而决定 *MAX_MAPPERS* 和 *MAX_REDUCERS* 的取值。但作业在实际的运行过程中,3 个不同的阶段并不能完全准确地将其分开,例如,Copy 阶段是将 Map 阶段的输出数据拷贝传递给 Reduce 作为输入,该阶段会在 Map 阶段完成 20%(默认值)后开始执行,并不会等所有的 Map 阶段全部完成后执行。并且在同一时间段内,同一节点上可能同时存在 3 个阶段。因此,本文通过时间片将作业分为不同的阶段,建立基于时间片的资源占用特征值。

在作业执行过程中,通过编写 Python 脚本调用 Psutil[15]接口来获取集群每个节点的资源(CPU,I/O 和内存等)占用情况。假设作业执行时间为 t,将其划分为相等的 m 个时间段,对于集群中的第 i 个节点,在第 j 个时间段内第 k 种资源的平均占用情况用 U_{ij}^{k} 来表示,那么在第 i 个节点上资源占用特征可以表示为:

$$F(i)=\begin{pmatrix} U_{i1}^1 & U_{i2}^1 & \cdots & U_{ij}^1 & \cdots & U_{im}^1 \\ U_{i1}^2 & U_{i2}^2 & \cdots & U_{ij}^2 & \cdots & U_{im}^2 \\ \vdots & \vdots & & \vdots & & \vdots \\ U_{i1}^k & U_{i2}^k & \cdots & U_{ij}^k & \cdots & U_{im}^k \\ \vdots & \vdots & & \vdots & & \vdots \\ U_{i1}^r & U_{i2}^r & \cdots & U_{ij}^r & \cdots & U_{im}^r \end{pmatrix}$$

其中,r 为资源种类数。

(2)匹配作业特征库

为了衡量 2 个作业的资源占用特征 $F_a(i)$ 和 $F_b(i)$之间的相似度,本文定义了作业相对距离 RD,具体计算方式如式(1)所示。

$$RD(F_a(i),F_b(i))=\frac{\sum_{k}^{r}\sum_{j}^{m}|F_a(i)[k][j]-F_b(i)[k][j]|}{m} \tag{1}$$

$RD(F_a(i),F_b(i))$指作业 a 和作业 b 之间的相对距离,即作业 a 和作业 b 在一定的时间间隔向量空间里资源占用特征值之间的距离。

当 $RD(F_a(i),F_b(i))<\delta$ 时,认为作业 a 与作业 b 存在相似特征,2 个作业运行时的最优配置参数比较接近;当 $RD(F_a(i),F_b(i))\geqslant\delta$ 时,认为作业 a 与作业 b 不相似,2 个作业运行时的最优配置参数可能不一致。

对于一个新作业 J_{new},在输入数据为 D_1($size(D_1)=1$ GB)时其资源占用特征值为 F',将 F' 与作业特征库进行匹配,如果存在作业 $J_1,J_2,\cdots,J_t$,它们的资源占用特征与 F'的相对距离小于阈值 δ,则以与 F'的相对距离最小的作业的最优配置作为作业 J_{new}的最优配置;如果不存在,则对该作业 J_{new}建立新的作业特征值,并更新作业特征库。

(3)建立作业特征库

创建作业特征库,用于存储已知作业 J 的作业特征值 $V=<Name,F,Conf>$,作业特征值包括作业名称 $Name$、作业资源占用特征值 F、作业最优配置值 $Conf$。

针对作业 J(输入数据大小为 1 GB),首先在默认配置下,执行作业,获取作业的资源占用特征 F,然后在不同的 $<MAX\_MAPPERS,MAX\_REDUCERS>$ 值下执行作业,获取最优配置 $Conf$ 值,为了减少迭代次数,获取最优配置 $Conf$ 值的具体步骤如下:

1)在默认配置的基础上,以步长 1 来增大 $MAX\_REDUCERS$ 的取值,并在调整配置后用数据 D_1 运行作业,获取作业 J 的执行时间 T。

2)当 $MAX\_REDUCERS$ 的值增加到 i 时,如果 $T_i>T_{i-1}$,则将 $i-1$ 作为 $MAX\_REDUCERS$ 的最优取值 $Conf\_red$。

3)在 $MAX\_REDUCERS$ 确定的基础上,以步长 1 来增大 $MAX\_MAPPERS$ 的取值,并在调整配置后用用数据 D_1 来运行作业,获取作业 J 的执行时间 T'。

4)当 $MAX\_MAPPERS$ 的值增加到 k 时,如果 $T'_k>T'_{k-1}$,则将 $k-1$ 作为 $MAX\_MAPPERS$ 的最优取值 $Conf\_map$。

5)结束。

获得作业 J'的最优配置 $Conf=<Conf_map,Conf_red>$后,将作业 J 的作业特征值 $V=<Name,F,Conf>$添加到作业特征库中。

4 实验与结果分析

为了验证优化策略的有效性,本文进行了对比实验。先采用 Hadoop 默认参数配置分别运行了 TeraSort,WordCount 和 RandomWriter(RW)作业,然后再采用优化策略在相同集群下运行这 3 个作业。

本文的实验平台为 hadoop-0.20.1。集群拥有 1 个主节点、3 个从节点,集群中每个节点的配置均为一个 4 核 CPU、4 GB 内存、1T 硬盘、CentOS 操作系统。

4.1 作业相对距离对比

本组实验主要用来观察不同作业的资源占用特征之间的相对距离,实验结果如图 5 所示。

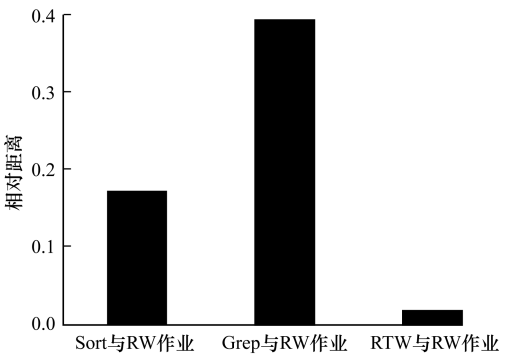

图 5 资源占用特征的相对距离

可以看出,RW 与 RTW 之间的相对距离小于 0.1,而与 Sort 和 Grep 之间的相对距离大于 0.1。RTW 与 RW 都是生成随机数作业,两者采用相同的算法,只是输出类型不一致,并且都对 I/O 资源需求较大。因此,RW 与 RTW 属于类型相似的作业,有相似的资源需求,所以具有相似的最优配置。而 Grep 主要是进行正则表达式的解析与匹配,对 CPU 需求较大;Sort 主要是对输入数据进行排序,对 CPU、内存和 I/O 等资源的需求较为平均。因此,RW 与 Grep、Sort 属于类型不相似的作业,不具有相似的资源需求,故导致 RW 与 Grep 和 Sort 作业运行时资源占用特征相差较大。笔者通过多次实验发

现,相似作业的资源占用特征的相对距离小于 0.1,不相似作业的资源占用特征的相对距离大于 0.1,因此,本文实验 δ 取值为 0.1。

4.2　作业执行效率对比

本组实验主要用来对比在默认参数下作业的运行时间和采用优化策略之后作业的运行时间。实验结果如图 6 和图 7 所示,图 6 描述了在输入数据大小为 5 GB 的情况下作业运行时间对比情况,图 7 描述了输入数据大小为 10 GB 的情况下作业运行时间对比情况。可以看出,在采用优化策略之后,Wordcount,Terasort 和 RW 作业运行的时间都要比采用默认参数配置时的运行时间要短。

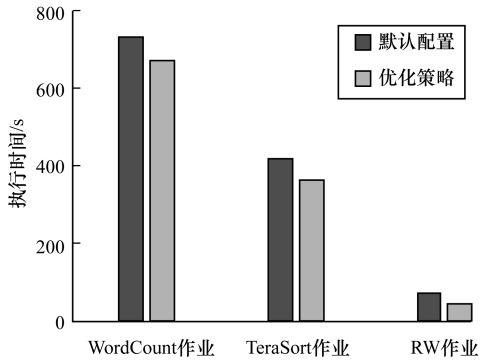

图 6　输入数据大小为 5 GB 时的作业执行时间对比

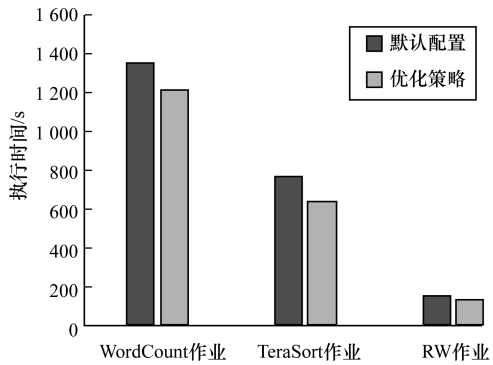

图 7　输入数据大小为 10 GB 时的作业执行时间对比

4.3　作业占用资源情况对比

本组实验主要对比采用默认配置参数和采用优化策略 2 种情况下作业运行时节点资源占用情况。

图 8 为 Terasort 作业在采用默认配置参数运行时的节点 CPU 占用情况,图 9 为采用优化策略时节点 CPU 占用情况。可以看出,在同一个集群下,采用优化策略之后,节点的资源使用率有明显提高。

图 10 显示了 Terasort 作业在采用默认配置参数运行时的节点 I/O 占用情况,图 11 显示了采用优化策略时节点 I/O 占用情况。可以看出,在优化策略下的节点资源使用率明显高于采用默认参数配置下的资源使用率。

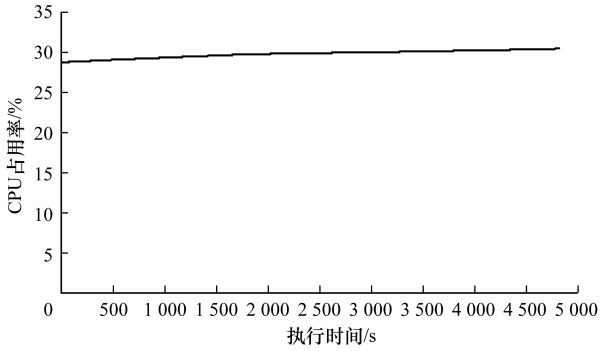

图 8　默认参数下 CPU 占用情况

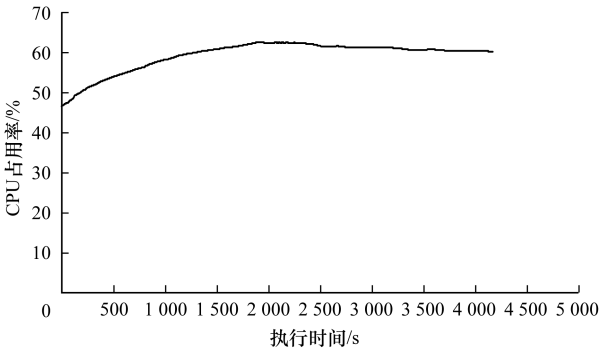

图 9　优化策略下 CPU 占用情况

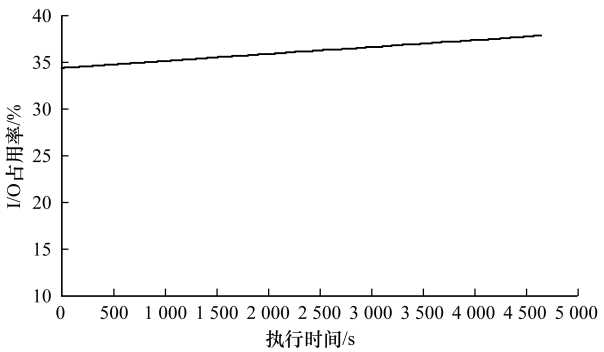

图 10　默认参数下 I/O 占用情况

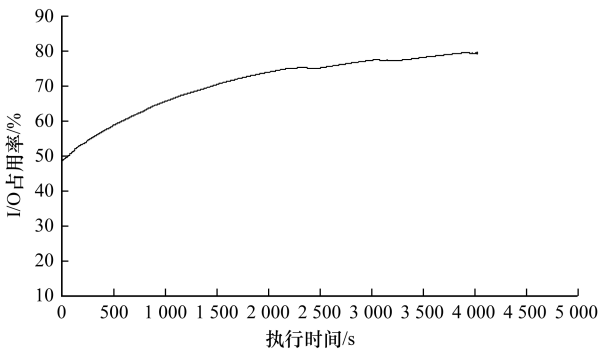

图 11　优化策略下 I/O 占用情况

图 12 为 Terasort 作业在采用默认配置参数运行时的节点内存占用情况,图 13 为采用优化策略时的节点内存占用情况。可以看出,在优化策略下的节点内存资源平均占用率要高于默认参数配置下的占用率。

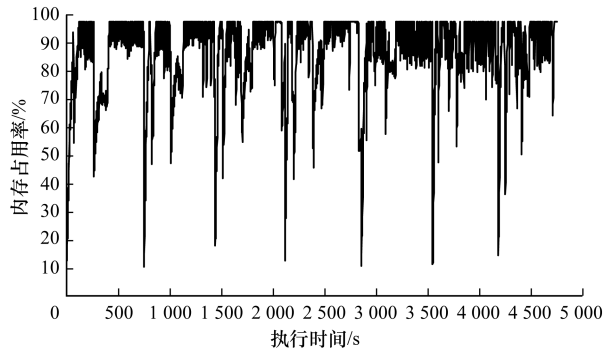

图 12 默认参数下内存占用情况

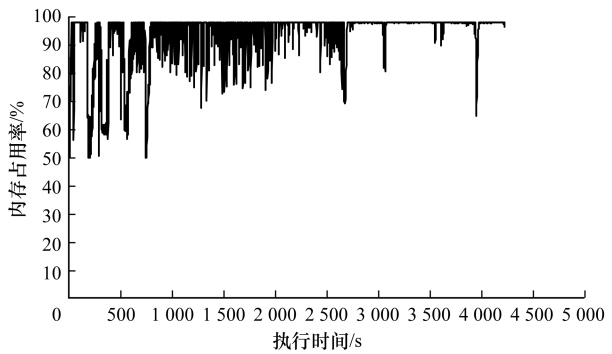

图 13 优化策略下内存占用情况

5 结束语

本文从参数优化角度出发,研究如何设置合适的 *MAX_MAPPERS* 和 *MAX_REDUCERS* 取值增加作业在运行过程中的任务并行效率,以达到优化 Hadoop 集群性能的目的。本文提出的优化策略根据作业资源占用特征在特征库中查找与其相似作业的最优配置,为作业生成更合理的参数取值,从而提高节点的资源利用率,达到优化作业执行效率的目的。实验数据表明,该策略能够显著提高作业执行效率,优化 Hadoop 集群性能。

但是,本文仅通过调整 2 个参数提高任务的并行性进行集群优化,其优化的效果是有限的。因此,如何在增加任务并行效率的基础上,通过修改其他配置参数达到单个任务效率最优化,以及如何通过单个任务最优化的配置参数与任务并行最优化的配置参数达到更大的性能,在下一步工作中将针对这些问题做进一步研究。

参考文献

[1] 黄承真,王 雷,刘小龙,等. Hadoop 任务分配策略的改进[J]. 计算机应用,2013,33(8):2158-2162.

[2] Hadoop[EB/OL]. (2012-12-27). http://hadoop. apache. org/.

[3] 北京百度网讯科技有限公司. 百度开放云[EB/OL]. [2014-09-10]. http://bce. baidu. com/Product/BMR. html.

[4] The Apache Software Foundation. Applications Powered by Hadoop[EB/OL]. (2009-07-01). http://wiki. apache. org/hadoop/PoweredBy.

[5] Rasooli A, Down D G. COSHH: A Classification and Optimization Based Scheduler for Heterogeneous Hadoop Systems [J]. Future Generation Computer Systems, 2014, 36(3): 1-15.

[6] Nykiel T, Potamias M, Mishra C, et al. MRShare: Sharing Across Multiple Queries in MapReduce [J]. Proceedings of the VLDB Endowment, 2010, 3(1/2): 494-505.

[7] Lim H, Herodotou H, Babu S. Stubby: A Transformation-based Optimizer for MapReduce Workflows [J]. Proceedings of the VLDB Endowment, 2012, 5(11): 1196-1207.

[8] Herodotou H, Dong F, Babu S. No One(Cluster) Size Fits All: Automatic Cluster Sizing for Data-intensive Analytics[C]//Proceedings of the 2nd ACM Symposium on Cloud Computing. New York, USA: ACM Press, 2011: 1-18.

[9] Intel Corporation. Intel Distribution for Apache Hadoop Software: Optimization and Tuning Guide [EB/OL]. (2012-10-11). http://hadoop. intel. com/pdfs/IntelDistributionTuningGuide. pdf.

[10] Amazon Elastic MapReduce[EB/OL]. [2014-09-10]. http://aws. amazon. com/elasticmapreduce/.

[11] Clouder [EB/OL]. [2014-09-10]. http://www. cloudera. com/.

[12] 周世龙,陈兴蜀,罗永刚. Hadoop 视角下的 Nutch 爬行性能优化[J]. 计算机应用,2013,33(10):2792-2795.

[13] Tannir K. Optimizing Hadoop for MapReduce[M]. [S.l.]: Packt Publishing Co., Ltd., 2014: 28-29.

[14] Lama P, Zhou Xiaobo. AROMA: Automated Resource Allocation and Configuration of MapReduce Environ-ment in the Cloud [C]//Proceedings of the 9th Inter-national Conference on Autonomic Computing. San Jose, USA: ACM Press, 2012: 63-72.

[15] Python Software Foundation. Psutil [EB/OL]. [2014-09-10]. https://pypi. python. org/pypi/psutil/.

编辑 金胡考