基于MapReduce的主成分分析算法研究

易秀双[1] 刘 勇[1] 李 婕[1] 王兴伟[2]
(东北大学计算机科学与工程学院 沈阳 110819)[1] (东北大学软件学院 沈阳 110819)[2]

摘 要 随着MapReduce并行化框架的流行,各种数据挖掘算法的并行化也成为了当下研究的热点。主成分分析(Principle Components Analysis,PCA)算法的并行化也得到了越来越多的关注。通过对目前PCA算法的并行化研究的成果进行总结,发现这些PCA算法并行程度并不完全,特别是特征值计算过程。整个PCA算法流程分为两个阶段:相关系数矩阵求解阶段和矩阵的奇异值分解(Singular Value Decomposition,SVD)阶段。通过当前最流行的并行框架MapReduce,融合矩阵的QR分解,提出了一种奇异值分解的并行实现方法。利用随机产生的不同维度大小的双浮点矩阵比较并行奇异值分解相对传统串行环境下的算法效率的提升情况,并分析算法效率。之后,将并行奇异值分解融合到PCA算法中,同时提出相关系数矩阵的并行计算过程,将PCA计算的两个部分完全并行化。利用不同维度的矩阵对提出的并行PCA算法与已存在的未完全并行PCA算法、常规的PCA算法的运算速度进行比较,分析完全并行化PCA算法的加速比,最终得出所提算法在处理一定规模的大数据情况下的时间消耗要少许多。

关键词 主成分分析,奇异值分解,MapReduce
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.02.007

Research of Distributed Principle Components Analysis Algorithm Based on MapReduce

YI Xiu-shuang[1] LIU Yong[1] LI Jie[1] WANG Xing-wei[2]
(School of Computer Science and Engineering, Northeastern University, Shengyang 110819, China)[1]
(School of Software, Northeastern University, Shengyang 110819, China)[2]

Abstract With the population of the parallel framework of MapReduce, the parallelization of various types of Data Mining algorithms is becoming a hot area of research. Principle components analysis(PCA) algorithm is getting more and more attention too. Summarizing the recent research result of the parallelization of PCA, we found that these PCA algorithms are not fully parallelized, especially the process of calculating the eigenvalue of the matrix. Whole process of PCA algorithm is divided into two stages, which are solution of the correlation coefficient matrix and the singular value decomposition of the matrix. Through the combination of the most popular MapReduce parallel framework and the QR decomposition of matrix, a new way to parallel the SVD was proposed in this paper. Analyzing the calculation speed of the parallel algorithm through the experiment on the data set which is consisted of random produced double floating point matrix of different dimensions, the result with the traditional serial algorithm was compared to show the efficiency improvement of the mentioned algorithm. Then we integrated the SVD algorithm into the PCA algorithm, and proposed the parallel computing process of the correlation coefficient matrix, and this will parallel the two stages of PCA algorithm. Subsequently, we conducted the comparison between the existing not fully parallelized PCA algorithm and normal PCA algorithm with the proposed algorithm on different dimensions of the matrix. Then analying the speed-up ratio of the proposed algorithm, we can find that our algorithm will consume less time when processing massive data set.

Keywords Principle components analysis, SVD, MapReduce

1 引言

互联网的普及使得数据量呈指数级增长,且数据的价值也越来越大。但随着数据体积越发庞大,结构越发复杂,如何有效地处理数据就成了各个行业面临的首要难题。

随着MapReduce分布式计算框架的出现,并行化分布式

到稿日期:2015-11-13 返修日期:2016-09-03 本文受国家杰出青年科学基金资助项目(61225012,71325002),国家自然科学基金资助项目(61572123),高等学校博士学科点专项科研基金优先发展领域资助课题(20120042130003)资助。
易秀双(1969—),男,博士,教授,CCF高级会员,主要研究方向为计算机网络和数据处理;刘 勇(1992—),男,硕士生,主要研究方向为计算机网络与数据处理,E-mail:546043655@qq.com;李 婕 博士,副教授,主要研究方向为数据处理;王兴伟(1968—),男,教授,博士生导师,主要研究方向为计算机网络。

处理数据比之前更为简单可行，可以更高效地分析处理数据，并且 MapReduce 可扩展性高，可配置在大量廉价的机器上。因此，与分布式计算结合的数据挖掘算法自然成为了数据处理的主要手段。

PCA 是统计分析中一种广为人知的降维的模型，PCA 的降维作用使其可以作为数据提取的主要方法，当前对于体积庞大的数据，利用 PCA 可以很快提取其主要特征，清除冗余噪音[1,2]，同时在提取过程中的数据过滤也使得 PCA 可以作为一种压缩手段对数据进行有效的有损压缩[3]。因此，随着 PCA 模型对数据提取方面的作用越来越大，PCA 算法的并行化也得到了广泛的关注。

PCA 求解过程分为两个部分：相关系数的矩阵分解和矩阵特征值的求解。夏慧明和周永权[4]通过数学手段探究出求解矩阵特征值和特征向量的新方法，采用迭代分析中最为常用的遗传算法对矩阵进行有效的特征求解。而 Chao Jin 等人[5]已经通过 MapReduce 框架实现了遗传算法的并行化改进，即能够完成 PCA 的 MapReduce 并行化。但是，遗传算法有最终结果依赖于相关参数的选取及很容易达到部分最优解而失去全局最优解等缺点，以及 MapReduce 框架本身并不能很好地支持迭代算法的实现，这反而降低了其作为并行框架的优势。Carlos Ordonez 等人[6]首次利用 Mapreduce 实现了 PCA 前半部分相关系数矩阵在 Hadoop 环境下的并行计算，使算法的可扩展性有了很大的提高，大大降低了算法的复杂度。另外，此算法在实现 PCA 的过程中使用了奇异值分解的方法，但 Carlos 认为奇异值分解在串行环境下的实现效率是可以接受的，所以其采用 LAPACK[7]直接对矩阵的奇异值分解进行了计算。但事实上，随着矩阵规模的不断增大，串行的奇异值分解的计算方法必然会成为整个算法的瓶颈。

由于矩阵特征值分解和矩阵奇异值分解的相似性，以及矩阵奇异值分解的优势和并行化，将其应用到 PCA 并行化中也得到了广泛的关注。Martin Becka 等人[8]实现了利用 Jacobi 算法进行 SVD 的并行化计算，该算法主要采用两个新方法：将问题分为几个小的子问题进行解决以及对矩阵进行 QR 迭代的预处理。算法本身很好地实现了 SVD 的并行化，在效率上有了很大的提升，但算法仍然限定在 MPI 环境下，且迭代依旧是算法中一个很难把握的问题。

综合以上算法，以往的 PCA 并行实现都有两个缺陷：

(1)之前在 CUPA 或 MPI 环境下，并行化是针对特定问题实现的，缺乏可扩展性，类似的这种并行化实现只能针对少部分领域的专业人才，很难达到普及，从而使得算法的局限性约束了相关应用的发展。

(2)算法并行程度并不完全，大部分 PCA 只实现了算法一部分的并行化，在面对大数据时仍然会造成瓶颈问题。

基于以上两点，本文提出了一种基于并行框架 MapReduce 的并行 PCA 算法。文章结构分为以下几个部分：第 2 节介绍 SVD，说明 SVD 可以代替特征值求解来完成 PCA 的计算；第 3 节主要讲解并行 PCA 算法，其包含两个部分即 SVD 的并行化和将 SVD 并行融入到 PCA 并行化中，完成对 PCA 的并行化实现；第 4 节是通过实验验证并行 SVD 和 PCA 算法的效率提升，分析并行 PCA 算法的加速比；最后总结全文。

2 奇异值分解和主成分分析

奇异值分解(SVD)[9-10]在线性代数中是一种很常用的矩阵分解方法，它有着明显的物理意义，SVD 分解可以将一个大的矩阵分解为 3 个小矩阵乘积的形式，这 3 个矩阵都有着其在实际数据中的重要意义。与特征值分解只能适用于方阵不同，SVD 是适用于任意矩阵的一种特征提取的方法。其分解形式为：

$$A=U\Sigma V^{T} \tag{1}$$

其中，A 为任意 $m\times n$ 的矩阵；U 为 $m\times m$ 的方阵，称为左奇异矩阵，矩阵中的向量都是正交的；Σ 为对角矩阵，Σ 中记录的就是矩阵 A 的奇异值；V^{T}(V 的转置)则是 $n\times n$ 的正交方阵，称为右奇异矩阵。

奇异值与特征值都可以在一定程度上表示数据矩阵的特征，因此 SVD 与 PCA 算法也就有一定程度的联系。根据 PCA 的理论，方差最大的坐标轴为第一个奇异向量，根据方差大小得到的第二大坐标轴就是第二个奇异向量，以此类推，在奇异值分解的方程两边同时乘以矩阵 V 可得到：

$$A_{m\times n}V_{n\times r}\approx U_{m\times r}\Sigma_{r\times r}V^{T}_{r\times n}V_{n\times r} \tag{2}$$

由于矩阵 V 为正交矩阵，则可得：

$$A_{m\times n}V_{n\times r}\approx U_{m\times r}\Sigma_{r\times r}=P \tag{3}$$

以矩阵 P 与特征值方程的对比可以看出，两种方法都是对矩阵 A 右乘以某一个变换矩阵后得到特征矩阵。通过对矩阵 P 进行特征提取即可得到矩阵 A 的压缩形式。由此可以看出，PCA 可以看作是对 SVD 的一种变形，假设实现 SVD 是可行的，那也就能够完成 PCA 的实现，并且更重要的是，SVD 的实现经过计算可以获得行、列两个方向上的 PCA 计算结果。

本文的并行 PCA 采用 SVD 而非特征求解的解决方法，实际上上文已经对 SVD 与特征求解的关系进行了论述，SVD 计算所得的奇异值可以简单理解为矩阵的另一种特征表示，它与特征值能对矩阵起到相同的作用[11]。同时，SVD 计算所得的左右奇异矩阵实际上就是矩阵特征向量所合成的矩阵，只不过行列上有一定的区别，实际上这也是本文采用 SVD 而非特征求解的原因之一。

3 并行主成分分析算法

PCA 算法的计算过程的两个步骤，首先对矩阵 A 进行标准化变化，对其相关矩阵 R 进行求解；在得到矩阵的相关矩阵 R 后，对相关矩阵进行 SVD 求解得到所需的奇异值矩阵和左右奇异向量。在计算奇异值的同时，计算各个奇异值的累计贡献率，若累计贡献率大于或等于 90%，则认为所选取的奇异值已经达到主成分要求，记录主成分的个数以及对应的奇异向量。

3.1 并行奇异值分解

3.1.1 算法思想

本文尝试利用 QR 分解实现矩阵的 SVD 分解并将算法应用到 MapReduce 环境中。Householder 变换[12-13]是计算矩阵 QR 分解最为常用的方法，本文的奇异值分解算法也用此方法来完成 QR 分解步骤。

但是，MapReduce 框架并不擅长于处理迭代算法。这里体现出了 MapReduce 的两个缺陷[14]：

(1)需要频繁地对分布式文件系统进行读写，程序的 I/O 消耗增加，大量的时间花费在数据的传输上，降低了分布式所带来的效率提升。

(2)在对数据划分的过程中必须保证分配到每个节点上的并行数据是独立且互不影响的，即对于每个节点所计算出的结果最终可以有效地在 Reducer 中进行合并，且结果相对于非分布式环境下的计算不会产生偏差。

考虑到矩阵的计算很大程度上会受到矩阵各个维度数据之间的相互影响，最好的方法是对矩阵本身进行分块，针对每一个矩阵分块进行计算。即将大数据划分为小数据，利用原有的单机情况下的算法对划分好的小数据进行计算，简单地说，对算法的 MapReduce 并行化主要是对数据矩阵进行并行化而放弃对算法步骤本身进行并行化。

本文的 MapReduce 并行化过程就是基于此思想，在对 SVD 以及 QR 分解算法进行并行化时，注重于将矩阵进行有效的分块处理。这就需要考虑到 MapReduce 的缺陷(1)。在对矩阵数据进行有效分块的同时，必须保证数据块在计算过程中不会产生过多的 I/O 消耗。众所周知，MapReduce 更加擅长处理少量的大数据，对矩阵进行有效分块时需考虑整个分布式计算过程中的瓶颈问题。

3.1.2 算法设计

本文的主要目的是对原有线性算法的 MapReduce 进行并行化，因此为了实现 SVD 的并行化，要对源矩阵 A 进行分块操作。将整个 MapReduce 并行化 SVD 算法的过程设计为 3 个步骤来实现(如图 1 所示)，每个步骤都设计若干 map 或 reduce 操作，同时在对矩阵进行分块时主要考虑实际情况(包括矩阵的规模大小、矩阵的复杂程度以及用于程序设计的集群节点数)，可以根据各自的实际情况对算法或 Hadoop 集群中的部分参数进行适当的调整，保证算法在满足需求的同时达到效率的最大化。

Step1　首先对需要处理的源矩阵 A 进行分块，根据集群的节点数或 map tasks 的个数将矩阵分为 3 部分。

$$A=\begin{bmatrix}A_1\\A_2\\A_3\end{bmatrix} \tag{4}$$

在 Step1 中只需对 map tasks 而不需要对 reduce 进行编写。对于每一个分块矩阵，在对应的节点下，计算一次 QR 分解，将得到的矩阵 Q 和 R 保存在文件系统中用于下面步骤的操作，这里将矩阵 Q 和 R 分别存于不同的文件中。整个分解的过程可以概括为：

$$A=\begin{bmatrix}A_1\\A_2\\A_3\end{bmatrix}=Q_1R=\begin{bmatrix}Q_{11}&&\\&Q_{12}&\\&&Q_{13}\end{bmatrix}\begin{bmatrix}R_1\\R_2\\R_3\end{bmatrix}=\begin{bmatrix}Q_{11}R_1\\Q_{12}R_2\\Q_{13}R_3\end{bmatrix} \tag{5}$$

每次 $Q_{1i}R_i$ 的 QR 分解都是在第 i 个 task 中进行的。

Step2　利用 Step1 得到的矩阵 R 进行 QR 分解以及 SVD 分解。Step2 中只需要一个 reduce task 即可。将数据收集为一个矩阵 R 并对其进行 QR 分解，整个过程可表示为：

$$R=\begin{bmatrix}R_1\\R_2\\R_3\end{bmatrix}=Q_2R'=\begin{bmatrix}Q_{21}\\Q_{22}\\Q_{23}\end{bmatrix}R' \tag{6}$$

在得到矩阵 R' 之后直接计算 R' 的 SVD 分解，则可以得到：

$$R'=U\Sigma V^{\mathrm{T}} \tag{7}$$

Step3　在 map task 中进行分块计算即可，map 函数的输入数据为 Step1 中所产生的矩阵 Q_1。而对于矩阵 Q_2，可以将 Q_2 整个作为一个文件，或根据实验机器的内存需要可将矩阵 Q_2 存放在一个类对象中，在进行 Q 的计算时为每个 map 都分配一次。则只需要将 Q_1 与 Q_2 矩阵相乘即可得到最终所需的矩阵 Q。

$$Q=\begin{bmatrix}Q_{11}&&\\&Q_{12}&\\&&Q_{13}\end{bmatrix}\begin{bmatrix}Q_{21}\\Q_{22}\\Q_{23}\end{bmatrix}=\begin{bmatrix}Q_{11}Q_{21}\\Q_{12}Q_{22}\\Q_{13}Q_{23}\end{bmatrix} \tag{8}$$

在得到矩阵 Q 后就可以将其与 Step2 中得到的矩阵 U 相乘，最终得到源矩阵 A 的左奇异矩阵 $U_e=QU$。则最终形式为：

$$A=QR'=(QU)\Sigma V^{\mathrm{T}}=U_e\Sigma V^{\mathrm{T}} \tag{9}$$

综合以上 3 个步骤叙述，可以看到算法很好地利用了 Map-Reduce的分布式特性，算法注重于对矩阵的分块处理，利用了“以小建大”的思想，根据现有稳定算法完成对小数据的操作，同时又不破坏整个矩阵的数据结果。在进行并行 SVD 的过程中，不需要像并行 Householder 算法对 QR 分解进行的操作那样每次处理一个数据列之后必须改变整个矩阵。节点 task 之间的处理数据完全独立，结果不会互相影响。

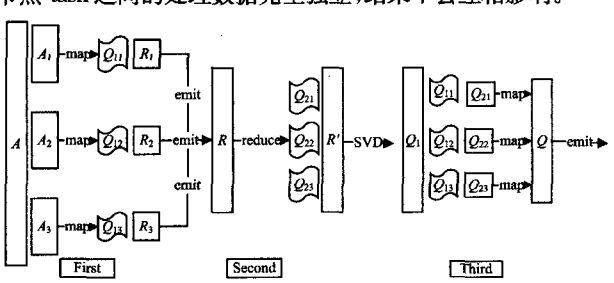

图 1　基于 MapReduce 的并行 SVD 流程图

3.2 并行主成分分析

3.2.1 算法思想

第 2 节第一部分已经完成了对 SVD 并行化的实现，所以所提算法首要的问题是如何在并行 MapReduce 的环境下求得矩阵 A 的相关矩阵 R，这样就可以将整个 PCA 算法并行化。

在整个 PCA 的计算过程中，相关系数矩阵的计算相比

于矩阵的 SVD 所消耗的时间更少，但对于大数据集的情况，如果不对相关系数矩阵的计算过程并行化，仍然会在一定程度上造成算法的瓶颈。如果要并行计算矩阵 A 的相关系数矩阵，则首先必须对矩阵 A 进行分块，而最常见的分块莫过于两种：按数据点分块或按维分块。如果对矩阵 A 按维进行分块，则不能得到理想的并行效果，因为在对矩阵 Q 进行计算时需要矩阵每一维的数据互相进行平方差积的运算，容易造成数据之间的交集叠加。算法必须避免在计算的过程中对数据频繁读取和对大量数据反复分配。理想情况下，算法应该满足根据数据计算节点数量的线性变化，并且算法应保持其稳定性不变。

3.2.2 算法设计

以往的研究中已经给出了最好的计算相关矩阵的线性方法，这里设原矩阵 $A_{m\times n}$ 满足 $A=(a_1,a_2,\cdots,a_n)$，则设

$$L=\sum_{i=1}^{n}a_i \tag{10}$$

$$Q=AA^{\mathrm{T}}=\sum_{i=1}^{n}a_i\cdot a_i^{\mathrm{T}} \tag{11}$$

其中，n 表示数据集矩阵的列向量个数，L 是矩阵 A 的所有向量求和所得的 $m\times 1$ 的向量，Q 是矩阵 A 所有向量平方和所得的 $m\times m$ 的对称矩阵。而对于相关矩阵 R，只通过一步计算矩阵 A 就可以有效地计算出相关系数值，同时利用求得的统计量 n,L,Q(也称其为和矩阵)[15]。而根据以上数学公式与相关系数矩阵的计算关系，可以很容易基于 n,L,Q 推导出相关系数值，即

$$R_{ab}=\frac{nQ_{ab}-L_aL_b}{\sqrt{nQ_{aa}-L_a^2}\sqrt{nQ_{bb}-L_b^2}} \tag{12}$$

综合以上，算法步骤为：

(1)利用 map 函数并行计算每个分块 A_i 所对应的 n_i，L_i，Q_i，$i=1\cdots N$。在处理环境相同的情况下，这一步在所有计算节点的过程中消耗的时间为 $O(m^2n/N)$。

(2)利用 reduce 函数同步所有 N 个节点的数据，将各个节点所计算的 n_i，L_i，Q_i 合并取和为一个整体的 n,L,Q，并将部分结果传输给 Master 主节点，这步需耗费的时间为 $O(m^2)$。

(3)根据步骤 2 的计算结果 n,L 和 Q 计算矩阵 A 的相关系数矩阵 R，最后对矩阵 R 进行适当地分块排列，这步需耗费的时间为 $O(m^2)$。

(4)根据并行 SVD 算法对相关系数矩阵 R 进行奇异值分解，得到左右奇异向量 U 和 V 以及奇异值矩阵 Σ，并根据奇异值的大小计算累计贡献率，最终计算出主成分的个数。

4 实验

4.1 SVD 并行化

利用双精度浮点型矩阵计算并行化 SVD 算法的运行效率，为了体现并行环境下算法的特点，分别采用 100，500，600，1000，1500，2000，2500 7 个维度的方阵验证算法的运行时间，同时记录相同情况下串行的 SVD 算法的运行时间，以比较两种环境下的并行效率，实验结果如图 2 所示。

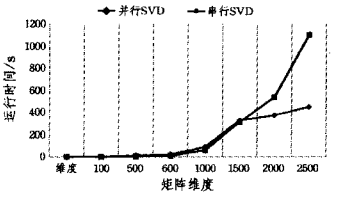

图 2 SVD 运行时间对比

图 2 显示了两种环境下的算法所体现的算法运行速度，在数据矩阵维度较小的情况下，串行环境下 SVD 的速度要明显快于并行 SVD 的计算速度，原因是并行环境下的 SVD 主要考量大数据矩阵，每一步的计算过程结果必须存入文件中进行保存，以供下一步处理，而在处理维数较少的矩阵时，文件的读取就会耗费大量的时间。从图 2 中可以看到，在矩阵维度接近 1500 时，两种算法的处理时间已经接近相同，而随着矩阵维度的进一步提升，并行 SVD 就展现了明显的算法优势，曲线的增幅明显下降，而相对的串行环境下的 SVD 则随着矩阵规模的变大而展现疲态，运行时间几乎呈指数型增长。

4.2 PCA 并行化

4.2.1 数据集

在验证过程中需要截取一定量的用户对几个电影的评分，截取的数据要求一定量的电影评分的用户完成了所选电影的 90%以上的评分，充分保证所选取的数据矩阵能够达到尽量高的稠密度。给出了并行 PCA 算法与未完全的 PCA 算法以及常规 PCA 在计算不同规模矩阵时的对比情况，其中矩阵的行数固定为 100000，逐渐扩大矩阵的列数据量并比较 3 种并行程度不相同的算法在不同矩阵维度下的时间对比情况。

4.2.2 实验结果分析

如图 3 所示，随着矩阵列数的不断增减，并行 PCA 在大数据的情况下展现了明显优势，在数据量较小的情况下，常规 PCA 算法和 SVD 过程中未完全并行的 PCA 算法在处理数据时优于并行 PCA 算法，原因是在处理函数远大于列数的情况下，并行 SVD 过程在数据的输入输出上要耗费大量的时间，对于此类数据，并行 PCA 并没有明显的优势可言。如图 3 所示，在数据量并不够大的情况下，3 种算法的计算时间实际上并没有明显的区别，8000 列以内数据量的运行时间的差距最大也只有不到 200s，可以说这种差距完全是能够接受的。而随着矩阵规模的扩大，并行 SVD 在算法中的优势可以完全得到体现。综上所述，并行 PCA 在处理行列数都比较大的矩阵数据时在算法的运行速度上有着明显的优势；而面对行列数都很小的矩阵时并行 PCA 的计算速度不如其他两种算法，但差距并不明显；而在列数很小但行数很大的情况下，所提并行 PCA 算法在计算效率上不如常规的串行 PCA 算法和未完全并行 PCA 算法。

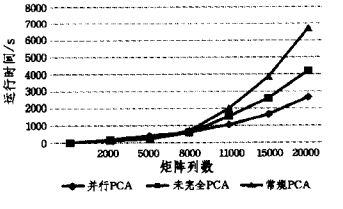

图 3 不同 PCA 算法对比

4.2.3 *加速比分析*

本节分析了完全并行化 PCA 算法的性能。由于加速比是用来衡量并行系统或程序并行化的性能和效果的评价指标，因此要分析完全并行化 PCA 算法和串行 PCA 算法的加速比。

本实验使用了包含 7 个节点的集群，其中一个节点是 Master 节点，负责任务调度和资源分配，其余 6 个节点负责计算。数据集是 MoiveLens 的两个用户评分矩阵，维数分别是 2500 * 4000(矩阵 1)和 10000 * 20000(矩阵 2)。加速比结果如图 4 所示。

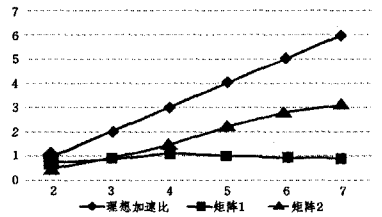

图 4　并行化的加速比

图 4 显示了在不同节点时不同矩阵体现出的不同加速比。从矩阵 1 的加速比(-■-)可以看出，当数据集很小时加速比不会随着节点数量的增加而显著增加，因为即使是小数据集，集群的启动时间也是不变的；反而当节点数量增加时，加速比有下降的趋势，数据的网络传输是重要的影响因素。

矩阵 2 的加速比(-▲-)表现出随着集群节点数量的增加而明显地增大，这是因为集群节点数量较少时，运算速度提升有限，中间结果使磁盘读写耗费更多的时间；而当集群节点数量增加时，运算速度提高了很多，而磁盘读写所需的时间不变。

结束语　所提的并行 PCA 算法具有很明显的并行计算优势，在运算速度上领先其他算法。而在矩阵数据较小的情况下，特别是行列数都比较小或行数远大于列数的情况下，并行 PCA 算法的优势并不明显，但面对小矩阵，并行 PCA 在运行上并没有太大的计算差距。

在利用 MapReduce 对矩阵进行相关系数矩阵的求解时，只对矩阵的部分结果进行了 MapReduce 操作，而最后计算相关系数矩阵值的过程是在单节点环境下进行的，通过实验也发现，随着数据量的增长，这很容易成为算法的瓶颈。因此如何优化相关系数矩阵的并行求解过程，使整个计算过程能够完全并行化是一个待研究的问题。

参 考 文 献

[1] Z L H, G Z K. Face Recognition Method Based on Adaptively Weighted Block-Two Dimensional Principal Component Analysis[C]// Third International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), 2011. IEEE, 2011: 22-25.

[2] FAN C, CHEN X. Research of face recognition based on wavelet transform and principal component analysis[C]// Eighth International Conference on Natural Computation (ICNC), 2012. IEEE, 2012: 575-578.

[3] LIM S T, YAP D F W, MANAP N A. Medical image compression using block-based PCA algorithm[C]// IEEE 2014 International Conference on Computer, Communications, and Control Technology (I4CT), 2014. IEEE, 2014: 171-175.

[4] JIN C, VECCHIOLA C, BUYYA R. Mrpga: an extension of mapreduce for parallelizing genetic algorithms[C]// IEEE Fourth International Conference on eScience, 2008. IEEE, 2008: 214-221.

[5] XIA H M, ZHOU Y Q. A new solution to solve the eigenvalue and eigenvector of mateix[J]. Computer engineering, 2008, 36 (11): 189-191. (in Chinese)
夏慧明，周永权. 求解矩阵特征值和特征向量的新方法[J]. 计算机工程，2008，36(11)：189-191.

[6] OORDONEZ C, MOHANAMN, GARCIA-ALVARADO C. PCA for large data sets with parallel data summarization[J]. Distributed and Parallel Databases, 2014, 32(3): 377-403.

[7] DEMMEL J W. Applied numerical linear algebra [M]. Siam, 1997.

[8] BEČKA M, OKŠA G, VAJTERŠIC M. Parallel Block-Jacobi SVD Methods[M]// High-Performance Scientific Computing. Springer London, 2012: 185-197.

[9] DUMAIS S T. Latent semantic analysis[J]. Annual review of information science and technology, 2004, 38(1): 188-230.

[10] ALTER O, BROWN P O, BOTSTEIN D. Singular value decomposition for genome-wide expression data processing and modeling[C]// Proceedings of the National Academy of Sciences, 2000, 97(18): 10101-10106.

[11] MA L, LI C, SONG S. Robustness analysis of watermarking spectral images with PCA-SVD under various illumination conditions[C]// IEEE International Conference on Information and Automation (ICIA), 2010. IEEE, 2010: 1835-1839.

[12] ZHAO Y P, LI B, LI Y B, et al. Householder transformation based sparse least squares support vector regression[J]. Neurocomputing, 2015, 161: 243-253.

[13] BALLARD G, DEMMEL J, GRIGORI L, et al. Reconstructing Householder vectors from tall-skinny QR[C]// 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 2014. IEEE, 2014: 1159-1170.

[14] XIONG K, HE Y. Power-effiicent resource allocation in MapReduce clusters[C]// 2013 IFIP/IEEE International Symposium on Integrated Network Management, 2013. IEEE, 2013: 603-608.

[15] ZHANG T, WANG C, LI W. Distributed parallel PCA algorithm in the application of large sample data set[EB/OL]. http://www. paper. educn/releasepaper/content/201112-636 (in Chinese)
张涛，王纯，李炜. 分布式并行 PCA 算法在大样本数据集中的应用[EB/OL]. http://www. paper. educn/releasepaper/content/201112-636.