Journal of Computer Applications
计算机应用,2017,37(1):37-41,53
ISSN 1001-9081
CODEN JYIIDU
2017-01-10
http://www.joca.cn

文章编号:1001-9081(2017)01-0037-05
DOI:10.11772/j.issn.1001-9081.2017.01.0037

MapReduce 并行加速数据流多模式相似性搜索

付 晨[1],钟 诚[1*],叶 波[2]
(1.广西大学 计算机与电子信息学院,南宁 530004; 2.广西科技信息网络中心,南宁 530012)
(*通信作者电子邮箱 chzhong@gxu.edu.cn)

摘 要:设计时间序列数据在 Hadoop 分布式文件系统(HDFS)中的有效存储方式,利用分布式缓存工具 Distributed Cache 将各子序列分发到 Hadoop 集群的计算节点上,将动态时间弯曲距离矩阵划分成多个子矩阵,采取并行迭代计算每条反对角线上子矩阵的方法,基于 MapReduce 编程模型,实现高效并行计算时间序列动态弯曲距离,通过改进剪裁冗余计算方法,设计实现一种数据流多模式相似性搜索并行算法。中国雪深长时间序列数据集的实验结果表明,当每条时间序列的长度达到5000以上时,并行计算动态弯曲距离所需时间少于串行计算所需时间,当每条时间序列的长度达到9000以上时,参与计算的集群节点越多,并行计算所需时间越少;当模式长度达到4000、参与计算的集群节点数达5个以上时,从数据流中并行搜索出与模式匹配的相似子序列所需时间约为串行搜索所需时间的20%。

关键词:时间序列;数据流;动态时间弯曲距离;模式搜索;Hadoop
中图分类号:TP338.6;TP301.6 **文献标志码**:A

Accelerating parallel searching similar multiple patterns from data streams by using MapReduce

FU Chen[1], ZHONG Cheng[1*], YE Bo[2]
(1. *School of Computer, Electronics and Information, Guangxi University, Nanning Guangxi* 530004, *China*;
2. *Guangxi Scientific and Technological Information Center, Nanning Guangxi* 530012, *China*)

Abstract: The effective storage mode for time series was designed on Hadoop Distributed File System (HDFS), the sub-series were distributed to the compute nodes on Hadoop cluster by applying Distributed Cache tool, and the matrix of dynamic time warping distances was partitioned into several sub-matrixes. Based on MapReduce programming mode, by parallel computing sub-matrixes in each back-diagonal iteratively, the parallel computation of dynamic time warping distances was implemented, and an efficient parallel algorithm for searching similar patterns from data streams was developed by improving pruning redundant computation. The experimental results on the data set of snow depth long time series in China show that when the length of each time series is equal to or longer than 5000, the required time of parallel computing dynamic time warping distances is less than that of the corresponding sequential computation, and when the length of each time series is equal to or longer than 9000, the more the compute nodes used, the less the required parallel computation time; furthermore, when the length of each pattern is equal to or longer than 4000 and the number of compute nodes is equal to or larger than 5, the required time of parallel searching similar sub-series from data streams is 20% of the corresponding sequential searching time.

Key words: time series; data stream; dynamic time warping distance; pattern searching; Hadoop

0 引言

时间序列数据流相似性搜索在网络点击流分析、金融分析、气象监测、语音识别等诸多领域具有广泛的应用[1]。大数据环境下长时间序列数据流多模式相似性搜索十分耗时。因此,在分布式并行计算 Hadoop 平台上,研究设计实现高效的数据流多模式相似性搜索并行算法具有重要应用价值。

欧氏距离及其扩展的相似性度量方法针对时间序列在时间轴上拉伸、收缩、平移等变形的健壮性不强[2]。动态时间弯曲(Dynamic Time Warping, DTW)距离[3]能够度量不同步不等长的时间序列,使它们通过一定的变形进行比较,适合于时间序列相似性比较。文献[4]利用 MPI(Message Passing Interface)和 Open MP 机制,采取均匀划分数据分配方法,设计机群计算时间序列动态弯曲距离的并行算法,获得了较好的加速。文献[5]针对无线传感器网络环境下不确定异常时间序列检测效率低下的问题,对不确定时间序列进行压缩变换,以减少不确定数据量,利用 MapReduce 架构实现基于期望距离的不确定时间序列动态弯曲距离算法并行化,同时提出了基于显著特征匹配的局部约束算法,对弯曲路径进行局部限制,提高了检测效率。文献[6]提出一种判断累加距离是否超出阈值的推算方法,以减少一些不必要的冗余计算。文献[7]利用值域划分柱图,将时间序列映射到 k 维空间,构

收稿日期:2016-08-25;**修回日期**:2016-09-03。 **基金项目**:广西自然科学基金资助项目(2014GXNSFAA118396)。
作者简介:付晨(1988—),女,江西瑞昌人,硕士,主要研究方向:网络软件工程、大数据高性能计算; 钟诚(1964—),男,广西桂平人,教授,博士,CCF 高级会员,主要研究方向:并行计算、大数据高性能计算、网络软件工程; 叶波(1974—),男,广西南宁人,教授级高级工程师,博士,主要研究方向:网络软件工程、管理信息系统。

造新的距离函数,获得动态时间弯曲距离的新的上、下界,从而缩小原始候选集,使算法仅需在小规模候选集上计算 DTW 距离,降低了计算复杂度。时间序列动态弯曲距离的一个重要应用是被用来搜索数据流中的相似模式。通过保存时间序列累计距离和候选序列的起始点,采取计算时间子序列匹配矩阵的方法,文献[8]的 SPRING 算法解决了针对数据流连续、实时、无限制等特性的时间序列的子序列匹配问题,但该算法存在冗余计算。文献[9]构造一个计分函数来降低动态时间弯曲距离的计算量,然后利用得到的时间弯曲距离来搜索发现数据流中公共的局部相似模式。

本文基于 Hadoop 平台,通过设计时间序列数据在 HDFS 中的有效存储方式和改进剪裁减少冗余计算方法,采取 MapReduce 并行迭代计算每条反对角线上动态时间弯曲距离子矩阵的方法,设计实现高效的数据流多模式相似性搜索并行算法,在获得较好匹配效果的同时,大大缩短计算时间。

1 MapReduce 并行计算动态时间弯曲距离

1.1 递推计算动态时间弯曲距离

采用递推方法计算时间序列 $X=\{x_1,x_2,\cdots,x_m\}$ 和 $Y=\{y_1,y_2,\cdots,y_n\}$ 之间的动态弯曲距离矩阵 $\boldsymbol{D}(X,Y)$[9]:

$$d(i,j)=\begin{cases}0, & \text{若 } i=0 \text{ 且 } j=0\\ \infty, & \text{若 } i=0 \text{ 或者 } j=0\\ (x_i-y_j)^2+c, & \text{其他}\end{cases} \quad (1)$$

其中 $c=\min\{d(i,j-1),d(i-1,j),d(i-1,j-1)\}$,$i=1,2,\cdots,m$,$j=1,2,\cdots,n$。从动态时间弯曲距离矩阵中可以寻找出弯曲路径 $DTW(X,Y)=\{w_1,w_2,\cdots,w_k\}$,从而得到 X 和 Y 之间的匹配关系。

1.2 数据存储与算法设计

1.2.1 算法设计

采用 MapReduce 编程模型可有效处理大规模科学计算问题[10]。为了在 Hadoop 平台上实现并行计算动态时间弯曲距离矩阵,将序列 X 划分为长度分别为$\lceil m/p\rceil$的 p 个子序列 $X_0,X_1,\cdots,X_{p-1}$,并将序列 Y 划分为长度分别为$\lceil n/q\rceil$的 q 个子序列 $Y_0,Y_1,\cdots,Y_{q-1}$。利用 Hadoop 分布式缓存工具 Distributed Cache 将各子序列分发到 Hadoop 集群计算节点上,于是构造 $p\times q$ 个子矩阵 $\boldsymbol{D}(f,g)$,$f=1,2,\cdots,p$,$g=1,2,\cdots,q$,每个子矩阵规模为$\lceil m/p\rceil\times\lceil n/q\rceil$,这些子矩阵 $\boldsymbol{D}(f,g)$ 可以并行计算[11]。

算法 1 描述了 Hadoop 平台上并行计算动态时间弯曲距离(Parallel Computing Distances of Time Winding, PCDTW)算法。

算法 1 PCDTW 算法。

输入:时间序列 X 和 Y。

输出:动态时间弯曲距离矩阵 $\boldsymbol{D}$。

Begin

1) 计算子矩阵 $\boldsymbol{D}(1,1)$,将其最后一行和最后一列存入 HDFS 中;
2) 读取 HDFS 中 $\boldsymbol{D}(1,1)$ 最后一行,存入 $\boldsymbol{D}(2,1)$ 的第 0 行,用于 $\boldsymbol{D}(2,1)$ 的计算;读取 HDFS 中 $\boldsymbol{D}(1,1)$ 最后一列,存入 $\boldsymbol{D}(1,2)$ 的第 0 列,用于 $\boldsymbol{D}(1,2)$ 的计算;将 $\boldsymbol{D}(2,1)$ 的最后一行与 $\boldsymbol{D}(1,2)$ 的最后一列存储到 HDFS 中;
3) 获取子矩阵 $\boldsymbol{D}(2,1)$ 的最后一行,存入 $\boldsymbol{D}(3,1)$ 的第 0 行;获取 $\boldsymbol{D}(1,2)$ 的最后一行及 $\boldsymbol{D}(2,1)$ 的最后一列,分别存入 $\boldsymbol{D}(2,2)$ 的第 0 行和第 0 列;获取 $\boldsymbol{D}(1,2)$ 的最后一列,存入 $\boldsymbol{D}(1,3)$ 的第 0 列;并行计算 $\boldsymbol{D}(3,1)$、$\boldsymbol{D}(2,2)$ 和 $\boldsymbol{D}(1,3)$;
4) 依此类推,获取子矩阵 $\boldsymbol{D}(f-1,g)$ 的最后一行及 $\boldsymbol{D}(f,g-1)$ 的最后一列,存入 $\boldsymbol{D}(f,g)$ 的第 0 行和第 0 列,计算 $\boldsymbol{D}(f,g)$,将 $\boldsymbol{D}(f,g)$ 的最后一行传送给 $\boldsymbol{D}(f+1,g)$ 计算,将 $\boldsymbol{D}(f,g)$ 的最后一列传送给 $\boldsymbol{D}(f,g+1)$ 计算;并行计算每条反对角线上的子矩阵,每次 Map/Reduce 过程完成计算一条反对角线上的子矩阵;
5) 若距离累加超过事先设定的阈值,则算法结束;否则迭代 $k=p+q-1$ 次,并行计算 $\boldsymbol{D}(f,g)$;

End

1.2.2 动态时间弯曲距离矩阵数据存储设计

HDFS 文件中每一行存储动态时间弯曲距离子矩阵中每个元素的信息,子矩阵信息用坐标(子矩阵分块行号,子矩阵分块列号)表示,元素信息包括(行号,列号#元素值)。HDFS 中每一行的存储格式为〈(子矩阵分块行号,子矩阵分块列号)(元素行号,元素列号#元素值)〉,即以〈$(f,g)(i,j\#d[i][j])$〉的形式存储,f 和 g 代表子矩阵在动态时间弯曲距离矩阵分块中的行、列号,i 和 j 为子矩阵元素在子矩阵中的行、列号,$d[i][j]$ 代表元素值。

1.2.3 Map 过程的设计

并行计算 Map 阶段的主要工作:从 HDFS 文件中逐行读取动态时间弯曲距离矩阵数据〈$(f,g)(i,j\#d[i][j])$〉;筛选出本次计算的子矩阵分块坐标(f,g),获取子矩阵的第 0 行元素和第 0 列元素。

如果上一轮迭代计算有相关子矩阵传递最后一行记录数据,那么本次计算的子矩阵用第 0 行接收。如果上一轮迭代计算有相关子矩阵传递最后一列记录数据,那么本次计算的子矩阵用第 0 列接收。

算法 2 并行计算 DTW 距离的 Map 函数。

输入:〈*key*1,*value*1〉为〈行号,HDFS 中每一行记录〉。

输出:中间结果〈*key*2,*value*2〉。

Begin

1) 逐行读取 HDFS 文件,获取子矩阵中每个元素信息〈$(f,g)(i,j\#d[i][j])$〉:
① *key*1 ← 行号;
② *value*1 ← 〈$(f,g)(i,j\#d[i][j])$〉。
2) 筛选出本轮迭代计算的子矩阵分块坐标(f,g),记录其第 0 行和第 0 列信息;
3) 将中间结果〈*key*2,*value*2〉写入本地节点的中间文件:
① *key*2 ← (f,g);*value*2 ← $(0,j\#d[0][j])$;
将中间结果〈*key*2,*value*2〉以“〈(子矩阵分块行号,子矩阵分块列号)(第 0 行,元素列号#元素值)〉”的形式(即〈$(f,g)(0,j\#d[0][j])$〉)写入本地中间文件;
② *key*2 ← (f,g);*value*2 ← $(i,0\#d[i][0])$;
将中间结果〈*key*2,*value*2〉以“〈(子矩阵分块行号,子矩阵分块列号)(元素行号,第 0 列#元素值)〉”的形式(即〈$(f,g)(i,0\#d[i][0])$〉)写入中间文件;

End

算法 2 中步骤 3)的中间结果 *key*2 值为本次计算的子矩阵坐标(f,g),相应的 *value*2 值为本次计算子矩阵第 0 行每个元素信息$(0,j\#d[0][j])$或者第 0 列元素的信息$(i,0\#d[i][0])$。通过算法 2 可筛选本次计算的子矩阵分块坐标(f,g),获取本次计算的子矩阵的第 0 行元素和第 0 列元素,并将其记录到中间文件。

1.2.4　Reduce 函数的设计

并行计算 Reduce 阶段的任务:根据输入的键值对〈*key*2, *list*〈*value*2〉〉,接收相同 *key*2 值对应的 *list*〈*value*2〉得到本次计算子矩阵第0行元素集合和第0列元素集合。提取分布式缓存中本次参与计算子序列 X_f 和 Y_g。记录本轮迭代计算的反对角线上的各个 DTW 距离子矩阵,同时将子矩阵最后一行和最后一列写到 HDFS 结果文件中,以传递给下一轮迭代计算时 Map 函数使用。

算法 3　并行计算 DTW 距离的 Reduce 函数。

输入:*key*2 值为子矩阵坐标(f,g),*list*〈*value*2〉值为计算子矩阵第 0 行元素集合和第 0 列元素集合。

输出:结果〈*key*3,*value*3〉。

Begin

1) for(相同 *key*2 值(f,g)对应的 *list*〈*value*2〉) do

begin

2) 获取 *value*2,给子矩阵第 0 行和第 0 列赋值;

3) 提取分布式缓存中参与计算子矩阵 $\boldsymbol{D}(f,g)$ 的子序列 X_f 和 Y_g;

4) 计算并记录子矩阵中各 DTW 距离 $d[i][j]$;

5) 将〈*key*3,*value*3〉写入 HDFS 的结果文件:

① *key*3 ← 本轮迭代计算的子矩阵坐标(f,g);*value*3 ← $(i, j\#d[i][j])$;将本次计算的子矩阵结果〈*key*3, *value*3〉以"〈(子矩阵分块行号,子矩阵分块列号)(元素行号,元素列号 # 元素值)〉"的形式(即〈$(f,g)(i, j\#d[i][j])$〉)写入 HDFS 的结果文件中;

② $d[c][j]$ ← 本次计算的子矩阵最后一行元素值;*key*3 ← 下轮迭代要计算的子矩阵坐标$(f+1,g)$;*value*3 ←$(0, j\#d[c][j])$;将下轮迭代计算的子矩阵〈*key*3,*value*3〉以"〈(子矩阵分块行号,子矩阵分块列号)(第0行,元素列号#元素值)〉"的形式(即〈$(f+1, g)(0, j\#d[c][j])$〉)写入 HDFS 的结果文件中;

③ $d[i][c]$ ← 本次计算的子矩阵最后一列元素值;*key*3 ←下轮迭代计算的子矩阵坐标$(f,g+1)$;*value*3 ←$(i,0\#d[i][c])$;将下轮迭代计算的子矩阵〈*key*3, *value*3〉以"〈(子矩阵分块行号,子矩阵分块列号)(元素行号,第 0 列 # 元素值)〉"的形式(即〈$(f,g+1)(i, 0\#d[i][c])$〉)写入 HDFS 的结果文件中;

end

End

并行计算 DTW 距离矩阵的主函数迭代地启动 Job 调用算法 2 和 3 的 Map/Reduce 函数。一轮迭代过程结束则计算完成一条反对角线上的子矩阵块,迭代次数加 1,输出路径编号加 1。算法采用 MapReduce 的多目录输出,将要传递的子矩阵最后一行及最后一列与本次计算的子矩阵分开不同目录存放。仅将本次 MapReduce 的输出路径中存放子矩阵最后一行及最后一列的目录,作为下轮迭代计算的输入目录,从而避免了不必要的数据复制和传递。在 Reduce 过程中,若出现 DTW 距离超过阈值,则算法提前结束。所有 Reduce 写入的子矩阵结果集合即为最终结果的 DTW 距离矩阵。

2　数据流多模式相似性并行搜索

模式 Q 与数据流 S 子序列 $S[t_s,t_e]$ 之间的 DTW 距离为 $d(S[t_s,t_e],Q)$,起始点位置记为 $sp(t,i)$[8]:

$$\begin{cases} d(S[t_s,t_e],Q) = d(t_e,m) = \min(d(t,m)) \\ d(t,i) = \| s_t - q_i \| + d_{\text{best}} \\ d_{\text{best}} = \min\{d(t,i-1), d(t-1,i), d(t-1,i-1)\} \end{cases} \tag{2}$$

其中 $d(t,0)=0, d(0,i)=\infty, t=1,2,\cdots,n, i=1,2,\cdots,m$。

$$sp(t,i) = \begin{cases} sp(t,i-1), & d(t,i-1) = d_{\text{best}} \\ sp(t-1,i), & d(t-1,i) = d_{\text{best}} \\ sp(t-1,i-1), & d(t-1,i-1) = d_{\text{best}} \end{cases} \tag{3}$$

从式(2)可看出,若 $\min\{d(t,i-1), d(t-1,i-1), d(t-1,i)\} > \varepsilon$ 且 $d(t,i) \geqslant 0$,则 $d(t,i) > \varepsilon$,ε 为相似性阈值。因此,可以按式(2)裁剪计算,提前排除超出阈值的子序列,以提高搜索效率。

算法 4 描述了 Hadoop 平台上时间序列数据流多模式相似性并行搜索(Multi-Pattern Parallel Searching Of Distributed Streams, MPPSODS)算法。

算法 4　MPPSODS 算法。

输入:数据流 S,模式 $Q_1, Q_2, Q_3, \cdots$。

输出:S 中与每个模式相匹配的前 k 条数据流子序列对应的起止点。

Begin

1) Map 阶段。将数据流分段分配到不同的 Map 任务中,每个任务在相应的 DataNode 上采用改进的 SPRING 算法进行相似性搜索:

① 将数据流分段 $S_1, S_2, S_3, \cdots$ 以及模式 $Q_1, Q_2, Q_3, \cdots$ 发送到各计算节点上,Map 获取计算的数据流分段及其 *id* 号、模式及其 *id* 号;

② 每个节点运行改进的SPRING算法,判断相似性阈值是否符合提前终止的条件;

③ 将模式 *id* 号与每次计算得到的 DTW 距离连接作为中间结果〈*key*,*value*〉中的 *key*,将匹配子序列的起始点、结束点存储在 *value* 中。

2) 中间阶段。将所有中间结果〈*key*,*value*〉经过一个名为 Shuffle 的过程,按 *key* 值排序并分配给对应的 Reducer 处理;

3) Reduce 阶段。输入的〈*key*,*list*〈*value*〉〉已按从小到大的顺序排列好,分解 *key* 中模式 *id* 与 DTW 距离,对于模式 *id* 相同的记录只读取前 k 条记录作为结果输出,同时将前 k 条记录匹配的数据流子序列对应的起止点写入文件;

End

3　实验分析

3.1　实验数据及环境

实验数据来源于"寒区旱区科学数据中心"(http://westdc.westgis.ac.cn)的中国雪深长时间序列数据集(1978—2012)[12-14],将其中的时间序列文件预处理为时间序列的每个数值点占一行,以〈序号,数值〉的形式存储。

Hadoop 平台采用通过交换机连接的 6 台计算机组成分布式并行计算环境,其中每台计算机的内存容量为 2 GB、处理器为 Intel Pentium CPU G2020、主频为 2.90 GHz,将 1 台计算机的角色作为主节点(Master)、名称节点(NameNode)和作业跟踪节点(JobTracker),其余 5 台计算机的角色作为从节点(Slave)、数据节点(Data Node)和任务跟踪节点(TaskTracker),系统网络传输速率为 100 Mb/s。Hadoop 版本为 hadoop-1.0.4。运行的操作系统为 ubuntu-10.04.4 版本。开发环境为 Eclipse,采用 Java 语言编程实现算法。

3.2　实验结果

首先测试并行计算动态时间弯曲距离(PCDTW)算法对运行时间改进的程度。

对于 12 个不同规模的数据集,当 Hadoop 平台上参与计算的节点数目逐步增多时,图 1 给出了 PCDTW 并行算法和

串行算法的运行时间。

从图 1 的结果可以看出，当时间序列长度小于 5 000 时，PCDTW 并行算法的运行时间多于串行算法的运行时间，且集群中参与计算节点越多反而越耗时。这是因为对于较短的时间序列，需要计算的 DTW 距离矩阵也相对较小，而每次计算一条反对角线上的子矩阵，都要启动一次 Job 调用 Map/Reduce 过程，通信交互和输入输出 IO 操作都相对耗时。

当每条时间序列的长度达到 5 000 以上时，在 Hadoop 上运行 PCDTW 算法的耗时低于串行算法。时间序列越长，PCDTW 算法加速效果越明显。当每条时间序列的长度达到 9 000以上时，集群中参与计算的节点越多，PCDTW 算法所需运行时间越少。

值得注意的是，当每条时间序列的长度达到 8 000 以上时，运行串行算法产生内存溢出，无法获得计算结果。

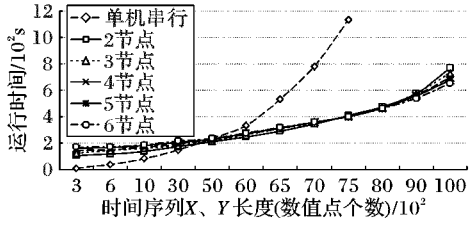

图 1 PCDTW 算法与串行算法的运行时间

Fig. 1 Execution time of PCDTW algorithm and serial algorithm

对于时间序列 X 和 Y 的长度在 10 000 以上的数据集 A、B、C、D 和 E，PCDTW 算法计算 DTW 距离矩阵需要的存储容量如表 1 所示。

表 1 计算 DTW 距离矩阵需要的存储容量

Tab. 1 Required memory capacity of computing DTW distance matrix

数据集编号	时间序列长度	存储数据量/GB
A	10 000	3.6
B	15 000	8.4
C	20 000	13.7
D	25 000	22.4
E	30 000	32.3

从表 1 可知，当时间序列较长时，计算两序列之间的 DTW 距离矩阵需要较大的存储容量。

图 2 给出了 Hadoop 集群中参与计算的节点数目分别为 2,3,4,5 和 6 时，PCDTW 算法计算数据集 A、B、C、D 和 E 的 DTW 距离所需的时间。

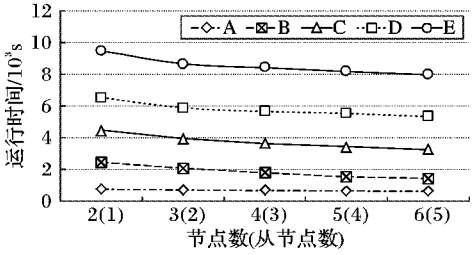

图 2 参与计算节点增多时 PCDTW 算法的运行时间

Fig. 2 Execution time of PCDTW algorithm when increasing compute nodes

从图 2 可以看出，当参与计算的节点数目固定时，时间序列数据集规模越大，PCDTW 算法所需时间也越多；对于同一个数据集，当集群中参与计算的节点增多时，PCDTW 算法所需时间逐渐减少。这表明，Hadoop 平台上并行计算时间序列 DTW 矩阵的 PCDTW 算法能够利用集群中不断增多的节点的计算能力。

加速比反映了算法的并行性对运行时间的改进程度。Hadoop 集群中参与计算的节点数目分别为 2,3,4,5 和 6 时，PCDTW 并行算法获得的加速比如图 3 所示。

图 3 的结果表明，随着时间序列的增长，PCDTW 算法获得的加速比逐渐提高。当每条时间序列长度达到 7 000 以上时，在 Hadoop 集群上运行的 PCDTW 算法加速更加明显。这说明，PCDTW 并行算法适用于计算时间序列长、数据集大的动态时间弯曲距离。

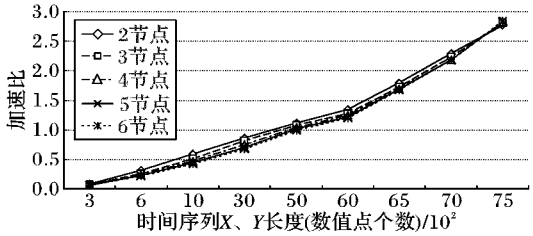

图 3 参与计算节点增多时 PCDTW 算法的加速比

Fig. 3 Speed-up ratio of PCDTW algorithm when increasing compute nodes

为了测试数据流多模式相似性搜索并行算法 MPPSODS 相对于串行算法 STRING 运行时间的改进程度，将包含 25 000 个数据点的数据流划分为 5 段，依次对表 2 中的 H、I、J 和 K 这 4 组模式集中每组的 3 个模式（查询序列）进行并行搜索，算法所需的运行时间如图 4 所示。

表 2 模式（查询序列）集及相似性阈值

Tab. 2 Pattern (search sequence) set and similarity threshold

模式集	模式 1	模式 2	模式 3	模式长度	相似性阈值
H	HQ1	HQ2	HQ3	1 000	40
I	IQ1	IQ2	IQ3	2 000	80
J	JQ1	JQ2	JQ3	3 000	120
K	KQ1	KQ2	KQ3	4 000	160

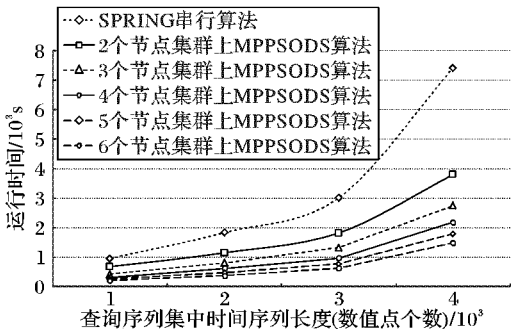

图 4 MPPSODS 算法与串行算法的运行时间

Fig. 4 Execution time of MPPSODS algorithm and serial algorithm

从图 4 中可看出，随着模式（查询序列）中时间序列长度的增大，并行算法 MPPSODS 和串行算法 SPRING 搜索时间也随之增长，但 SPRING 算法所需的运行时间增长较快，而 MPPSODS 算法的运行时间增长较为缓慢，且 Hadoop 集群中参与计算的节点越多，MPPSODS 算法运行时间越少。

对于 H、I、J 和 K 这 4 组模式（查询序列），图 5 给出了 Hadoop 集群参与计算的节点增多时，MPPSODS 算法的运行时间。

图 5 的结果表明，对于同一组模式（查询序列），随着 Hadoop 集群中参与计算的节点增多，MPPSODS 算法运行时间逐渐减少；而且模式越长、参与计算的节点越多，MPPSODS 算法所需时间越少。这说明，MPPSODS 算法有效利用了

Hadoop 集群中逐步增多的节点的计算能力。

图 6 ~ 8 分别给出 MPPSODS 算法对模式(查询序列)集 H 中 3 条序列 HQ1、HQ2、HQ3 进行相似性搜索得到的结果。

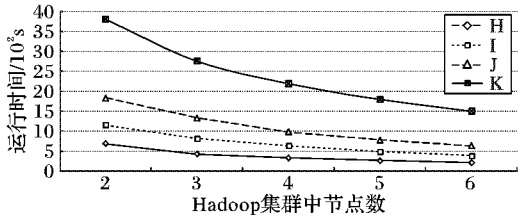

图 5　参与计算节点增多时 MPPSODS 算法的运行时间

Fig. 5　Execution time of MPPSODS algorithm when increasing compute nodes

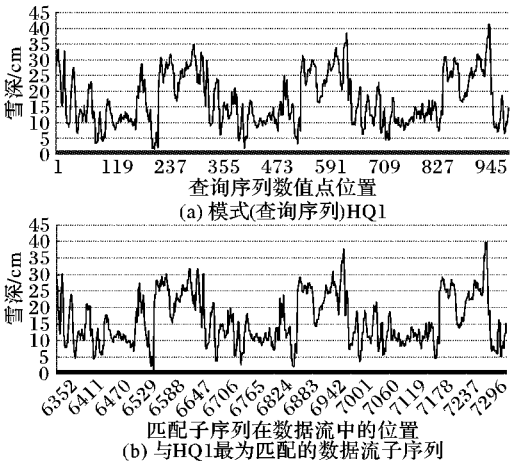

图 6　MPPSODS 算法对 HQ1 搜索的结果

Fig. 6　Results of searching pattern HQ1 using MPPSODS algorithm

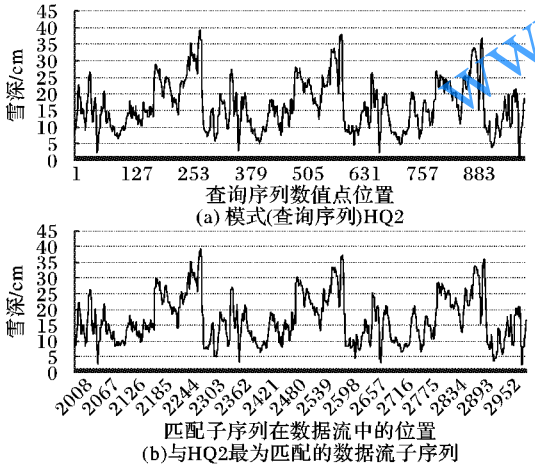

图 7　MPPSODS 算法对 HQ2 搜索的结果

Fig. 7　Results of searching pattern HQ2 using MPPSODS algorithm

从图 6 可以看到,在数据流上搜索出与模式(查询序列)HQ1 最为匹配的数据流子序列的开始位置为 6 362、结束位置为 7 337、长度为 986。

从图 7 可以看到,在数据流上搜索出与模式(查询序列)HQ2 最为匹配的数据流子序列的开始位置为 2 008、结束位置为 3 001、长度为 994。

从图 8 可以看到,在数据流上搜索出与模式(查询序列)HQ3 最为匹配的数据流子序列的开始位置为 11 455、结束位置为 12 425、长度为 971。

图 6 ~ 8 的结果表明,MPPSODS 算法并行搜索数据流得到匹配的子序列与模式(查询序列)在形态上具有很高的相似性。

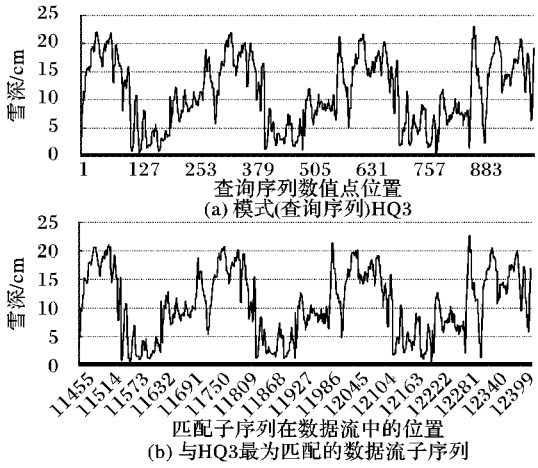

图 8　MPPSODS 算法对 HQ3 搜索的结果

Fig. 8　Results of searching pattern HQ3 using MPPSODS algorithm

4　结语

时间序列数据流具有连续、实时和无限制特性。大数据环境下长时间序列的动态弯曲距离计算和数据流多模式相似性搜索相当耗时,并行求解算法提供了有效的解决方案。基于 Hadoop 平台和 MapReduce 编程模型,本文设计实现的并行算法能够利用集群中不断增加的节点的计算能力,高效地计算动态时间弯曲距离和从数据流中搜索出与模式相似的子序列,所得到的匹配子序列与模式序列在形态上具有很高的相似性。下一步的工作将在 Hadoop 平台上开发一个时间序列数据流多模式相似性并行搜索软件包供人们在线使用。

参考文献(References)

[1] MATSUBARA Y, SAKURAI Y, FALOUTSOS C, et al. Fast mining and forecasting of complex time-stamped events [C]// Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2012: 271 - 279.

[2] WANG L, LEEDHAM G. Near and far infrared imaging for vein pattern biometrics [C]// Proceedings of the 2006 IEEE International Conference on Video and Signal Based Surveillance. Piscataway, NJ: IEEE, 2006: 52 - 52.

[3] 陈乾, 胡谷雨. 一种新的 DTW 最佳弯曲窗口学习方法[J]. 计算机科学, 2012, 39(8): 191 - 195. (CHEN Q, HU G Y. New leaning method for optimal warping window of DTW [J]. Computer Science, 2012, 39(8): 191 - 195.)

[4] 莫倩芸, 钟诚. 机群系统上并行计算时间序列的动态弯曲距离[J]. 微电子学与计算机, 2008, 25(10): 155 - 158. (MO Q Y, ZHONG C. Parallel computing dynamic warping distances for time sequences on the cluster computing systems [J]. Microelectronics & Computer, 2008, 25(10): 155 - 158.)

[5] 张建平, 李斌, 刘学军, 等. 基于 Hadoop 的不确定异常时间序列检测[J]. 传感技术学报, 2014, 27(12): 1659 - 1665. (ZHANG J P, LI B, LIU X J, et al. Uncertain abnormal time series detection based on Hadoop [J]. Chinese Journal of Sensors and Actuators, 2014, 27(12): 1659 - 1665.)

[6] 沙剑. 基于 GPU 的时间序列并行检索算法研究[D]. 大连: 大连理工大学, 2011: 42 - 55. (SHA J. The research on parallel time series retrieval method based on GPU [D]. Dalian: Dalian University of Technology, 2011: 42 - 55.)

(下转第 53 页)

于位图局部索引,设计云环境下起源图查询算法,并验证了算法的可行性。总之,提出的基于双层索引结构的起源图查询方法具有高效性,有应用价值和前景。

参考文献(References)

[1] DAVIDSON S B, LUDASCHER B, MCPHILIPS T, et al. Provenance in scientific workflow systems[J]. IEEE Data Engineering Bulletin, 2007, 30(4): 44 - 50.

[2] CHEBOTKO A, ABRAHAM J, BRAZIER P, et al. Storing, indexing and querying large provenance data sets as RDF graphs in Apache HBase [C]// Proceedings of the 2013 IEEE Ninth World Congress on Services. Piscataway, NJ: IEEE, 2013: 1 - 8.

[3] 朱敏. 基于 HBase 的 RDF 数据存储与查询研究[D]. 南京: 南京大学, 2013: 24 - 48. (ZHU M. Research on storage and query of RDF data based on HBase [D]. Nanjing: Nanjing University, 2013: 24 - 48.)

[4] W3C. RDF 1.1 concepts and abstract syntax [EB/OL]. [2016-06-15]. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225.

[5] W3C. PROV—overview [EB/OL]. [2016-06-15]. https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/.

[6] ATRE M, CHAOJI V, ZAKI M J, et al. Matrix bit loaded: a scalable lightweight join query processor for RDF data [C]// Proceedings of the 19th International Conference on World Wide Web. New York: ACM, 2010: 41 - 50.

[7] 杨彧剑, 林波. 分布式存储系统中一致性哈希算法的研究[J]. 电脑知识与技术, 2011, 7(22): 5295 - 5296. (YANG Y J, LIN B. Research on consistent hashing method in distributed storage system [J]. Computer Knowledge and Technology, 2011, 7(22): 5295 - 5296.)

[8] 赵彦荣, 王伟平, 孟丹, 等. 基于 Hadoop 的高效连接查询处理算法 CHMJ[J]. 软件学报, 2012, 23(8): 2032 - 2041. (ZHAO Y R, WANG W P, MENG D, et al. Efficient join query processing algorithm CHMJ based on Hadoop [J]. Journal of Software, 2012, 23(8): 2032 - 2041.)

[9] 郭峻峰. 数据仓库查询优化方法及索引技术研究[D]. 合肥: 合肥工业大学, 2010: 14 - 43. (GUO J F. Research of query optimization and index of data warehouse [D]. Hefei: Hefei University of Technology, 2010: 14 - 43.)

[10] 孟必平, 王腾蛟, 李红燕, 等. 分片位图索引: 一种适用于云数据管理的辅助索引机制[J]. 计算机学报, 2012, 35(11): 2306 - 2316. (MENG B P, WANG T J, LI H Y, et al. Slice bitmap index: an auxiliary indexing mechanism for cloud data management [J]. Chinese Journal of Computers, 2012, 35(11): 2306 - 2316.)

[11] 郭栋, 王伟, 曾国荪. 基于一致性树分布的数据分布式存储方法[J]. 计算机应用, 2013, 33(12): 3432 - 3436. (GUO D, WANG W, ZENG G S. Distributed data storage method based on consistent tree distribution [J]. Journal of Computer Applications, 2013, 33(12): 3432 - 3436.)

[12] University of Texas Provenance Benchmark (UTPB) [EB/OL]. [2016-06-12]. http://faculty.utpa.edu/chebotkoa/utp.

This work is partially supported by the National High Technology Research and Development Program (863 Program) of China (2013BAB06B04), the Technology Project of China Huaneng Group Headquarters (HNKJ13-H17-04), the Natural Science Foundation of Jiangsu Province (BK20130852), the Special Fund for Public Welfare Industry of the Ministry of Water Resources of China (201501007).

XU Guoyan, born in 1971, Ph. D., associate professor. Her research interests include big data, data provenance.

LUO Zhangxuan, born in 1989, M. S. candidate. His research interest is data provenance management.

SONG Jian, born in 1991, M. S. candidate. His research interest is big data management.

LYU Xin, born in 1983, Ph. D., lecturer. His research interests include cryptography, network information security.

(上接第 41 页)

[7] 欧阳一村. 基于 DTW 距离的两步式时间序列相似搜索[D]. 广州: 中山大学, 2010: 33 - 54. (OUYANG Y C. Two-step similarity search of time series based on DTW distance [D]. Guangzhou: Sun Yat-sen University, 2010: 33 - 54.)

[8] SAKURAI Y, FALOUTSOS C, YAMAMURO M. Stream monitoring under the time warping distance [C]// Proceedings of the 23rd International Conference on Data Engineering. Piscataway, NJ: IEEE, 2007: 1046 - 1055.

[9] TOYODA M, SAKURAI Y, ISHIKAWA Y. Pattern discovery in data streams under the time warping distance [J]. VLDB Journal, 2013, 22(3): 295 - 318.

[10] SRIRAMA S N, JAKOVITS P, VAINIKKO E. Adapting scientific computing problems to clouds using MapReduce [J]. Future Generations Computer System, 2012, 28(1): 184 - 192.

[11] 钟诚, 陈国良. PRAM 和 LARPBS 模型上的近似串匹配并行算法[J]. 软件学报, 2004, 15(2): 159 - 169. (ZHONG C, CHEN G L. Parallel algorithms for approximate string matching on PRAM and LARPBS [J]. Journal of Software, 2004, 15(2): 159 - 169.)

[12] 寒区旱区科学数据中心. 中国雪深长时间序列数据集(1978—2012) [EB/OL]. [2014-09-28]. http://westdc.westgis.ac.cn. (Cold and Arid Regions Science Data Center at Lanzhou. Snow depth long time series data set in China (1978—2012) [EB/OL]. [2014-09-28]. http://westdc.westgis.ac.cn.)

[13] CHE T, LI X, JIN R, et al. Snow depth derived from passive microwave remote-sensing data in China [J]. Annals of Glaciology, 2008, 49(1): 145 - 154.

[14] DAI L, CHE T, WANG J, et al. Snow depth and snow water equivalent estimation from AMSR-E data based on a priori snow characteristics in Xinjiang, China [J]. Remote Sensing of Environment, 2012, 127: 14 - 29.

This work is supported by the Natural Science Foundation of Guangxi (2014GXNSFAA118396).

FU Chen, born in 1988, M. S. Her research interests include network software engineering, high-performance computing for big data.

ZHONG Cheng, born in 1964, Ph. D., professor. His research interests include high-performance computing for big data, network software engineering.

YE Bo, born in 1974, Ph. D., senior engineer. His research interests include network software engineering, management information system.