计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(8):214−218 [doi: 10.15888/j.cnki.csa.006508]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Spark 的分布式数字信号处理算法库设计①

乔　昕, 刘　峰, 于碧辉

(中国科学院大学, 北京 100049)
(中国科学院 沈阳计算技术研究所, 沈阳 110168)
通讯作者: 乔　昕, E-mail: 1027833685@qq.com

摘　要: 传统的基于 DSP 与 FPGA 的数字信号处理技术更加适用于实时信号处理, 且受到数据规模和频率分辨率的限制, 使得其不适于进行大规模数据下的离线式数据处理、分析与挖掘的应用. 目前工业大数据分析平台可以采用 Spark 作为实时信号处理和离线信号处理加速的计算引擎, 但该分析平台缺少适用于分布式并行计算引擎的数字信号处理等数学计算的解决方案. 基于此, 本文提出了基于 Spark 的分布式数字信号处理算法库, 为面向分析的工业大数据应用场景提供支撑. 本文介绍了该算法库的架构设计, 并以 FFT 算法和 DFT 算法为例介绍了传统数字信号处理算法在 Spark 下的分布式实现, 最后对算法库进行了正确性测试和性能分析. 结果表明该算法库能够正确完成数字信号处理的功能, 同时可以满足工业大数据分析平台对于大规模数据集进行数字信号处理的需求.

关键词: Spark; 数字信号处理; 分布式计算; 算法库

引用格式: 乔昕,刘峰,于碧辉.基于 Spark 的分布式数字信号处理算法库设计.计算机系统应用,2018,27(8):214–218. http://www.c-s-a.org.cn/1003-3254/6508.html

Design of Distributed Digital Signal Processing Algorithm Library Based on Spark

QIAO Xin, LIU Feng, YU Bi-Hui

(University of Chinese Academy of Sciences, Beijing 100049, China)
(Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)

Abstract: The traditional DSP and FPGA-based digital signal processing technology is more suitable for real-time signal processing. It is limited by the size and frequency resolution of the data. So it unsuitable for applications of off-line data processing, data analysis and data mining under large-scale data. Currently the industrial big data analytics platforms can use Spark as a computational engine for real-time signal processing and off-line signal processing acceleration. However, there is a lack of mathematical solutions such as digital signal processing for distributed parallel computing engines. Consequently, this paper presents a library of distributed digital signal processing algorithms based on Spark, which provides a support for the analysis of industrial big data application scenarios. This paper describes the architecture of the algorithm library design and takes the FFT algorithm and DFT algorithm as examples to introduce the distributed implementation of the traditional digital signal algorithm in Spark. Finally, this paper persents a test and analysis for this algorithm library. The results show that the algorithm library can correctly accomplish the function of digital signal processing and it can fulfill the industrial big data analysis platform for large-scale data sets for digital signal processing needs.

Key words: Spark; digital signal processing; distributed computing; algorithm library

① 收稿时间: 2018-01-02; 修改时间: 2018-01-23; 采用时间: 2018-02-07; csa 在线出版时间: 2018-07-28

随着新一代信息技术的发展, 制造业与信息技术的融合应用已经成为研究的热点. 基于 DSP 与 FPGA 的数字信号处理技术在实时数字信号处理方面已经较为成熟[1], 可以满足常规应用场景下的数字信号处理需求. 但是该技术存在诸多限制, 首先其对于历史数据的分析处理能力不足, 其次受到存储空间的制约, 所以需要将工业设备采集到的数据通过采样缩小样本空间, 再使用计算单元进行计算. 但是由于工业设备的精度越来越高、功能越来越丰富, 使得其产生的数据呈现了大数据特征, 如数据量大、数据类型多、数据价值密度低等. 这就需要使用一种新的面向海量历史信号数据分析的方案解决以上问题, 作为对离线数字信号处理方案的有效补充.

随着 HDFS 等分布式文件系统的出现, 存储海量数据已经成为可能. 然而数字信号处理算法是迭代式的计算, 在计算过程中会产生大量的中间结果, 并且这些中间结果还需要参与下一轮计算, 所以中间结果的存储就可能变成整个系统的性能瓶颈. Spark 是面向大数据处理的并行计算引擎, 在大数据处理方面有广泛的应用, 并且它是基于内存计算的[2], 在计算过程中数据是存储在内存中的, 不存在由于 I/O 而造成的时间和磁盘空间的消耗. 由于 Spark 比较适合迭代计算, 因此工业大数据分析平台可以采用 Spark 作为实时信号处理和离线信号处理加速的计算引擎, 但是目前该分析平台仍然缺少适用于分布式并行计算引擎的数字信号处理等数学计算的解决方案.

针对这些问题, 本文提出了基于 Spark 并行计算引擎的分布式数字信号处理算法库, 用于解决大规模数据下的离线式数字信号处理问题, 为面向分析的工业大数据应用场景提供支撑. 该算法库不仅适用于大规模数据的并行处理, 在速度和性能方面也有很大的提升.

本文主要分为四个部分, 第一部分介绍了该分布式数字信号处理算法库的功能组织形式和 Spark 层面的架构设计. 第二部分完成了对多种常用数字信号处理算法的分布式实现, 并以 FFT 算法和 DFT 算法为例分别介绍了两类不同算法的分布式实现思想. 第三部分对本文设计的分布式数字信号处理算法库进行了正确性测试和性能分析. 第四部分对本文进行了总结以及对未来工作的展望.

1 分布式数字信号处理算法库的功能组织与架构设计

本文所设计的分布式数字信号处理算法库基于 RDD 技术实现, 部署在 Spark 上, 利用了 Spark 基于内存计算的特性, 更加适用于大规模数据的快速并行计算[3–5].

该算法库的结构主要包含三部分:

第一部分为计算平台接口及数学算法封装, 主要包括 Spark 并行计算 API 的封装以及矩阵、向量计算的数学算法库的封装. 这部分主要完成包括矩阵、向量等数据结构的定义和计算.

第二部分为算法工具包, 包括优化算法、特征提取、算法评估、外部数据读入等. 这部分主要用于数据预处理和应用分布式算法计算过程中的调优与结果评估.

第三部分为分布式算法集合, 包含了常用的数字信号处理算法的分布式实现.

算法库的功能组织图如图 1 所示. 图 1 中的分布式算法集合是算法库结构中的第三部分, 即在 Spark 下以分布式方式实现的常用的数字信号处理算法. 工具包是算法库结构中的第一部分与第二部分, 作为分布式算法集合的辅助工具, 主要用于数据预处理、结果评估和数学计算.

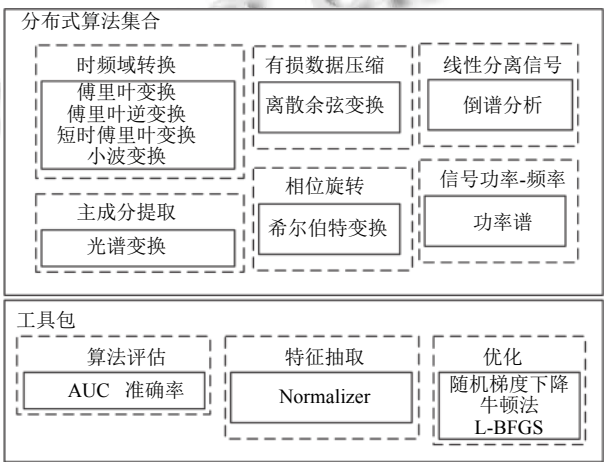

图 1 算法库功能组织图

从能否进行并行化计算优化的角度, 算法库的数字信号处理算法大致可以分为两类, 第一类是以离散傅里叶变换 (DFT) 算法为代表的无法通过基于时间的拆分进行优化的算法, 这一类算法将基于 Spark 并行

计算架构进行计算, 在保证计算正确性的同时, 实现大规模数据支撑与加速计算. 第二类是可以通过基于时间的拆分进行优化的算法, 这类算法在基于 Spark 并行计算框架实现大规模数据支撑与加速计算的基础上, 通过基于时间的拆分实现算法优化, 达到二次加速目的, 快速傅里叶变换 (FFT) 算法就是这类算法的典型代表[6–8].

图 2 所示为 Spark 层面的算法库的架构图. 其中包括两部分, 通过拆分优化加速部分即为对第二类算法的处理部分, 当第二类算法进入该部分时, 首先会对数据进行预处理, 然后根据具体算法的拆分方式进行数学拆分. 而基于 Spark 计算框架优化部分即为对第一类算法的处理部分, 由于第一类算法无法进行公式拆分, 所以这部分直接对其数学公式进行累加计算, 并使用 Spark 计算引擎的内部特性, 如内存计算特性等对算法的计算过程进行优化加速.

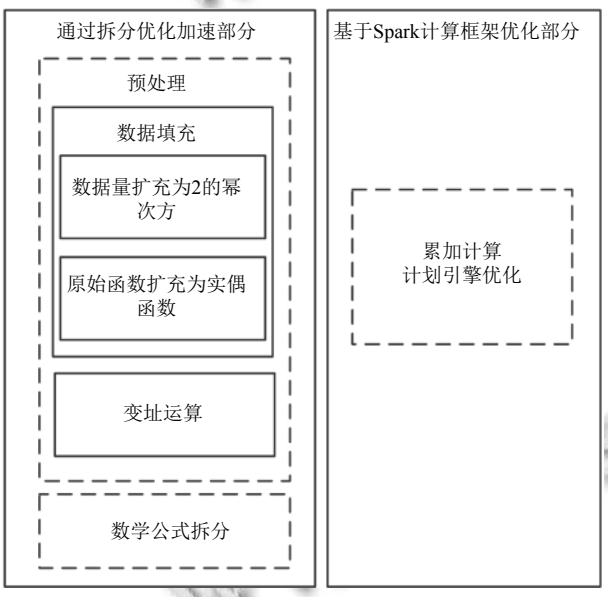

图 2 Spark 层面算法库架构图

对于某一具体算法, 无论是适用于拆分优化还是计算框架优化, 当其通过算法库进行处理后, 便可交由 Spark 计算引擎进行并行化计算, 从而得到分布式计算结果.

算法库在设计和实现上充分考虑了对于数字信号处理算法的扩展性, 所以对于本文中没有涉及的数字信号处理算法可以在完成相应的分布式实现后按照功能加载入相应的模块中.

2 分布式算法实现

由于傅里叶变换 (FFT) 算法可以作为大多数数字信号处理算法的实现基础, 所以本文以 FFT 算法为例, 作为第二类算法的实现代表, 详细介绍其在 Spark 并行计算引擎下基于并行计算优化的数字信号处理算法的实现思想. 分布式 FFT 算法的思想可以作为通用实现思想, 对算法库中的其他同类算法在此基础上做适应性调整, 从而均可完成其他算法的分布式实现.

2.1 基于 Spark 并行计算引擎的分布式 FFT 算法实现

Spark 中的数据是通过 RDD 进行存储和表示的, 使用 map 和 reduce 函数进行 RDD 之间的转换[3–5]. 通过文献[6–8]可知, FFT 算法的关键步骤为蝶形运算和变址运算, 所以本部分主要介绍对蝶形运算和变址运算中的 map 和 reduce 的设计思想. 除此之外, 由于 FFT 算法可以使用按照时间拆分的基 2 方法进行优化计算, 所以在本部分中还详细介绍了数据填充部分的步骤, 作为对原始数据的预处理环节.

2.1.1 数据填充

由于 FFT 算法可以使用按照时间抽取的基 2 方法实现, 其核心思想是将 N 点序列按照奇偶序进行逐级迭代拆分, 直到最后只剩下两项做累加计算时停止, 所以要求原始数据量必须是 2 的幂次方, 否则无法进行拆分. 这就需要先对原始数据集进行数据量大小的判断, 若数据量为 2 的幂次方, 则无需进行填充, 可直接用于计算; 若数据量不是 2 的幂次方, 则需要在数据末尾进行填充至大于原始数据量的最小的 2 的幂次方. 这里使用的填充方式为零填充, 即在数据集末尾补零. 具体过程如下所示:

(1) 创建 RDD 用于存储原始数据.

(2) 将 RDD 传递给 map 函数, map 函数首先会为每条数据赋予一个递增的 key 值, 然后判断该数据块是否需要填充, 若需要则进行填充, 否则不做任何操作. map 过程将会产生新的 RDD.

(3) 将第二步的 RDD 传递给 reduce 函数进行简单格式整合, 生成这一阶段的结果 RDD.

经过这一阶段后产生的结果 RDD 即为符合长度要求的数据集, 存储在内存中, 以便进行下一阶段操作.

2.1.2 变址运算

由于按照时间抽取的基 2-FFT 算法的结果序列的存储下标与输入序列的存储下标不同, 而原始序列的

存储下标又是按序排列的, 与结果序列相同, 所以需要将原始序列变换为输入序列. 本阶段的具体步骤如下:

(1) 将经过数据填充阶段的结果 RDD 作为该阶段的原始序列.

(2) 将原始序列传递给 map 函数, map 函数会将每条记录的 key 值转换为二进制表示, 然后将二进制 key 值进行逆序转换, 再将逆序 key 值排序, 最后把排序好的结果存储在新的 RDD 中.

(3) 将第二步生成的 RDD 传递给 reduce 函数进行简单格式整合, 生成新的 RDD. 该 RDD 中存储的数据即为符合要求的输入序列.

2.1.3 蝶形运算

本阶段是 FFT 算法的核心. 在这一阶段中会涉及复数运算, 由于 RDD 可以被抽象地理解为一个大的数组, 它存储的不是真实数据, 而只是真实数据的分区信息和针对每个分区的读取方法, 所以可以直接使用 RDD 进行复数的操作[3,4]. 具体步骤如下所示:

(1) 将经过变址运算阶段的 RDD 作为输入序列.

(2) 将输入序列传递给 map 函数, map 函数将每条记录的 key 值更换成统一值.

(3) 将第二步的结果传递给 reduce 函数, 在 reduce 函数中按照传统的 FFT 算法中的蝶形运算进行计算, 并将结果进行格式整合, 输出到文件中.

这一过程结束后, 生成的结果文件中存储的数据即为原始信号数据经过 FFT 之后的计算结果, 接下来可以使用其他工具进行画图分析.

2.2 基于 Spark 并行计算引擎的分布式 DFT 算法实现

对于第一类算法, 则可以使用 Spark 集群的并行计算优势进行加速计算. 由于 Spark 集群可以利用多台处理器的存储能力和计算能力, 即将任务分配给多台机器协同工作, 从而使该类算法的累加计算成为可能, 并且还能利用 Spark 计算引擎的特性在性能方面实现优化. 以离散傅里叶变换 (DFT) 算法为例, 具体实现步骤如下:

(1) 创建 RDD 用于存储原始数据.

(2) 将 RDD 传递给 map 函数, map 函数按照该算法的数学公式进行逐级累加, 并产生新的 RDD 用于存储中间结果.

(3) 将第二步的 RDD 传递给 reduce 函数进行简单格式整合, 并将结果输出至文件存储.

3 分布式数字信号处理算法库的正确性测试与性能分析

本部分对本文所设计的算法库进行正确性测试, 并分析在不同数据量的情况下该算法库的运行时间的变化趋势. 本次实验使用由 3 个节点构成的 Spark 集群, 3 个节点的硬件配置相同, 均为 16 G 内存, 4 GHz 主频, 4 核 CPU.

3.1 分布式数字信号处理算法库的正确性测试

本实验用于验证算法库中实现的分布式算法的正确性. 所使用的测试数据集为对六组信号采集器采集的振动信号使用采样频率为 256、采样点数为 256 进行采样得到的采样数据集. 该数据集以文件形式存储, 数据之间以逗号分隔. 将该数据集分别通过本文设计的算法库和 Matlab 中的对应算法进行计算, 将计算结果存储在结果文件中, 并对结果文件中的数据进行对比.

通过比较计算结果可知, 该数据集通过本文设计的算法库进行计算与使用 Matlab 对应算法进行计算得到的结果相同, 即可以说明本算法库中实现的分布式算法均能正确完成数字信号处理的功能.

3.2 分布式数字信号处理算法库的性能分析

本实验用于测试该算法库的性能. 实验使用六组输入信号, 均为信号采集器采集的振动信号, 以文件形式存储, 数据之间以逗号分隔, 具体参数如表 1 所示.

表 1 信号文件参数列表

编号	文件大小 (G)	每组采样点数 (w)	组数
1	1.6	100	1000
2	3.2	100	2000
3	4.8	100	3000
4	6.4	100	4000
5	8.0	100	5000
6	9.3	100	6000

经过本文设计的算法库中各算法计算后, 得到每个文件中数据的计算结果, 并存储在结果文件中.

以上六个文件的数据量与运行时间的关系图如图 3 所示.

由上图可知, 当数据集较小时, 数据量与时间基本成线性关系, 随着数据集的增大, 计算时间呈现非线性增长趋势. 出现此趋势的原因是集群的存储能力和计算能力受节点数量的影响.

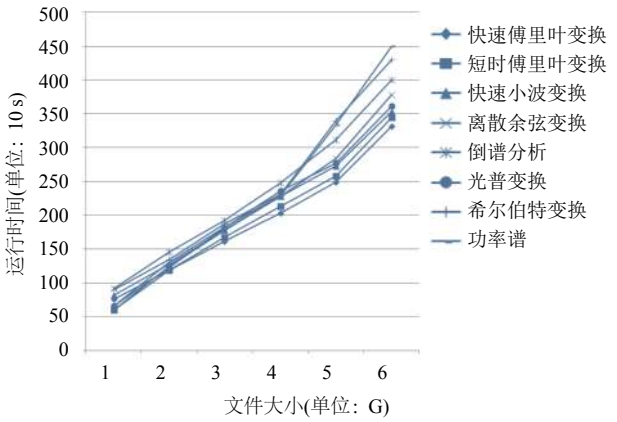

图 3 数据量-运行时间关系图

4 结论与展望

本文基于大规模数据下的数字信号处理及离线分析的需求, 设计了基于 Spark 并行计算引擎的分布式数字信号处理算法库, 详细介绍了该算法库的架构设计, 并以 FFT 算法和 DFT 算法为例介绍了两类数字信号处理算法的并行化计算实现及基于 Spark 计算引擎的优化方法, 最后对该算法库进行了正确性测试和性能分析. 根据测试结果可以看出, 本文设计的分布式数字信号处理算法库能够正确完成数字信号处理的功能, 并且能够处理大规模数据集, 可以满足工业大数据分析平台对于大规模数据集进行数字信号处理的需求.

下一阶段的工作内容主要是完成对本文实现的分布式数字信号处理算法库的性能评估与调优和基于 Spark 计算引擎的其他分布式数字信号处理算法的优化方法研究.

参考文献

1 彭宇, 姜红兰, 杨智明, 等. 基于 DSP 和 FPGA 的通用数字信号处理系统设计. 国外电子测量技术, 2013, 32(1): 17–21. [doi: 10.3969/j.issn.1002-8978.2013.01.008]

2 冯兴杰, 王文超. Hadoop 与 Spark 应用场景研究. 计算机应用研究 (优先发表), 2018, 35(9).

3 李玮. Apache Spark 技术研究与应用前景分析. 电信技术, 2016, (9): 67–68, 71. [doi: 10.3969/j.issn.1000-1247.2016.09.017]

4 英昌甜, 于炯, 卞琛, 等. 基于 RDD 关键度的 Spark 检查点管理策略. 计算机研究与发展, 2017, 54(12): 2858–2872. [doi: 10.7544/issn1000-1239.2017.20160717]

5 Zaharia M, Xin RS, Wendell P, *et al*. Apache spark: A unified engine for big data processing. Communications of the ACM, 2016, 59(11): 56–65. [doi: 10.1145/3013530]

6 刘大庆, 林浩然, 陈树越. 快速傅里叶变换中计算倒序的新思路. 电子与信息学报, 2018, 40(3): 758–762.

7 马学娟. 基于快速傅里叶变换 (FFT) 和小波变换的大型风机机械振动故障的分析. 科技与创新, 2016, (11): 121, 125.

8 Chen L, Hu Z, Lin JM, *et al*. Optimizing the fast Fourier transform on a multi-core architecture. Proceedings of 2017 IEEE International Parallel and Distributed Processing Symposium. Rome, Italy. 2007. 1–8.