计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191000086

基于 Apache Storm 的增量式 FFT 及其应用

赵　鑫[1,2]　马再超[1,2]　刘英博[1,2]　丁雨亭[1,2]　魏慕恒[3]
1 清华大学软件学院信息系统与工程研究所　北京 100084
2 工业大数据系统与应用北京市重点实验室　北京 100084
3 中国船舶工业系统工程研究院　北京 100070

摘　要　针对传统单机版批处理式的快速傅里叶变换(Fast Fourier Transform,FFT)难以满足工业生产现场海量流数据实时处理的需求,提出一种基于 Apache Storm 的增量式 FFT 方法。该方法设计了非递归 FFT 的流式计算逻辑,并实现于 Apache Storm。基于清华数为框架(DataWay Framework,DWF),采用 Bently 转子实验台的不对中故障流数据,构建了转子合成轴心轨迹的可视化监测界面,结果表明该方法能实时更新流数据频谱。

关键词:增量式 FFT;Apache Storm;清华数为框架;转子;合成轴心轨迹
中图法分类号　TP311.1

Incremental FFT Based on Apache Storm and Its Application

ZHAO Xin[1,2],MA Zai-chao[1,2],LIU Ying-bo[1,2],DING Yu-ting[1,2] and WEI Mu-heng[3]
1 Institute of Information System and Engineering,School of Software,Tsinghua University,Beijing 100084,China
2 Beijing Key Laboratory for Industrial Big Data System and Application,Beijing 100084,China
3 CSSC Systems Engineering Research Institute,Beijing 100070,China

Abstract　The conventional Fast Fourier Transform which is difficult to process the industrial big data in real time is a stand-alone algorithm with the batch processing techniques. In this paper,an Incremental FFT based on Apache storm is proposed. A non-recursive computational logic is first designed in Apache Storm. Then,a rotor misalignment experiment is performed on a Bently rotor test bench. With the rotor vibration data,a visual monitoring interface is developed by DataWay Framework. The result shows that the frequency spectrum of the stream data can be updated in real time with the proposed method and its realization.

Keywords　Incremental FFT,Apache storm,DWF,Rotor,Axis orbit

1　引言

大数据是机械故障诊断技术快速实现并走向智能化的新途径。2012 年,Li 等就指出大数据是现有产业升级与新产机械故障诊断技术业诞生的重要推动力量[1]。Yan 等综述了大数据故障诊断技术在解决数据质量、成本效益和方法选择等方面的问题[2]。Lei 认为机械故障诊断领域已经进入了“机械大数据”时代[3]。传统的离线式机械故障诊断一般基于内积变换原理,源于傅里叶变换[4-5],但该类方法一直缺乏工程化应用,难以满足工业现场的实时监测需求。

大数据技术为工业现场的实时计算与数据存储提供了有力保障,若将传统的机械故障诊断方法融合大数据技术并加以实现,则有望加速其工程化应用步伐。2014 年,Qin 等就指出为从工业大数据中挖掘有效信息,需要大数据计算框架的开发与应用方法,在此基础上研究模型或算法的重组、优化与实现[6]。Jiang 等探讨了大数据技术在电力设备状态评估研究中发挥的作用,并指出对物联网数据及其特征数据的集成、存储与管理需求[7]。FFT 是机械故障诊断技术的基础方法,本质是单机版的数据批处理方法,难以借助分布式、流处理等大数据技术提高算法执行的效率,在工业现场的高维高频大数据场景中应用困难。目前有少数基于分布式大数据计算框架的 FFT 应用研究,如 Qiao 等采用 Spark 作为数据批处理计算引擎,设计了 FFT 蝶形运算和变址运算的 map 和 reduce 计算流程,并采用 RDD 进行数据存储[8];Yang 等采用 mapreduce 机制设计了 FFT 的分布式计算机制,通过优化计算资源调度方法来避免冗余的计算节点数据交换,从而加速 FFT 的计算流程[9],并指出当前缺乏基于大数据技术支撑的 FFT 并行计算实现方法;Ji 等采用 Greenplum 中的 segment 设计 FFT 的分布式计算流程,并通过数据重分布来调整节点负载,优化 UDF 算法的执行性能[10]。其他经典方法也有相

本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家重点研发计划(2016YFB0501504);国家自然科学基金(U1509213)智能船舶 1.0 研发专项
This work was supported by the National Key Research and Development Program of China (2016YFB0501504) and National Natural Science Foundation of China (U1509213).
通信作者:赵鑫(zhao-x19@mails.tsinghua.edu.cn)

似的应用研究，如 Zhang 等采用 Storm 实时预处理风电机组状态监测流数据，并结合 Spark 并行实现基于朴素贝叶斯和 BP 神经网络的在线诊断和预警模型[11]。这些应用研究表明，大数据分析需求开始驱动传统的单机算法向分布式集群迁移实现。其他的 FFT 大数据改进方法主要是集成应用研究，如 Hu 等将 FFT 与深度神经网络结合，自适应提取信号时-频故障特征，并将其应用于高速列车的故障诊断[12]；Liu 等将 morlet 小波和深度神经网络结合后应用于往复式压缩机的故障诊断中[13]；Li 等将短时傅里叶变换的时频特征输入卷积神经网络，以识别滚动轴承故障类型[14]。

可以看出，大数据时代的机械故障诊断技术开始进入应用化阶段。通过集成应用研究，来验证故障诊断方法的业务有效性；通过分布式应用研究，来验证故障诊断方法的实施有效性。当前针对单台机组诊断精度提高的方法研究较多，但针对工业现场批量机组诊断功能的系统应用研究偏少。本研究以 FFT 的工程化应用为目标，以 Bently 转子系统为实验验证对象，结合 Apache Storm 和清华数为系统集成框架，对分布式计算和存储资源条件下的增量式 FFT 方法进行了探讨和实现，以期为工业现场大数据的流处理问题提供参考。

2　基本概念

2.1　FFT 的 Cooley-Turkey 算法

Cooley 和 Turkey 提出了 FFT[14]，通过将时间序列数据划分为奇数项和偶数项，可将其离散傅里叶变换（Discrete Fourier Transform，DFT）计算式表达为式(1)，时间复杂度由 $O(n^2)$ 降为 $O(n\log n)$。

$$X(n)=G(n)+W_N^nH(n) \tag{1}$$

其中，$G(n)=\sum_{k=0}^{N/2-1}x(2k)W_{N/2}^{nk}$，

$H(n)=\sum_{k=0}^{N/2-1}x(2k+1)W_{N/2}^{nk}$，$n=0,1,2,\cdots,N-1$

根据 W_N 的定义，$G(n)$ 和 $H(n)$ 都是关于 n 的周期为 $N/2$ 的函数，存在如式(2)和式(3)所示的关系。

$$X(n)=G(n)+W_N^nH(n) \tag{2}$$

$$X(n+N/2)=G(n)-W_N^n\boldsymbol{H}(n) \tag{3}$$

记任意一次调整顺序之后奇数位置的第 k 个信号的 DFT 为 $\boldsymbol{X}_1(k)$，偶数位置的第 k 个信号的 DFT 为 $\boldsymbol{X}_2(k)$，则定义蝶形运算符号如图 1 所示。

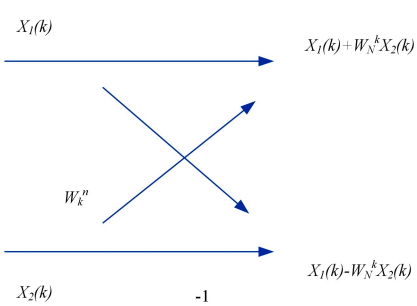

图 1　蝶形运算符号

Fig. 1　Butterfly operator

因此，只需要求出 2 个 $N/2$ 点的离散傅里叶变换 $X_1(k)$ 和 $X_2(k)$，再经过蝶形运算就可以求出全部的 $X(k)$ 值。以信号长度 $n=8$ 的序列为例，其蝶形运算如图 2 所示。

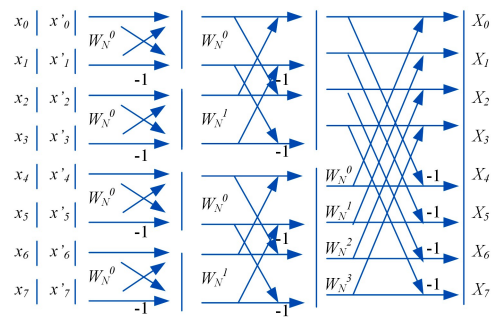

图 2　长度为 8 的 FFT 计算流程

Fig. 2　FFT process of length 8

2.2　Apache Storm

Apache Storm 是一个开源的分布式实时计算框架，可以处理流数据[15]。Apache Storm 拓扑如图 3 所示，Spout 负责分发流数据，Bolt 负责接收数据流并根据预先设定的处理方式计算，Tuple 建立数据传输通道。

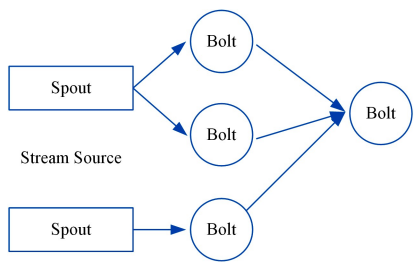

图 3　Apache Storm 拓扑图

Fig. 3　Topology of Apache Storm

2.3　DWF

DWF 是清华数为软件栈的核心组件之一，定位于快速开发应用系统，具有一站式部署、敏捷定制、低代码量开发、多数据源接入等特点。一站式部署，即解决技术栈选型和维护困难等问题，避免调试系统的时间开销；敏捷定制，即通过一系列可配置的控件组合来定制应用系统功能，既降低了开发门槛，又能快速适应需求变更；低代码量开发，即在线编程，无需搭建集成开发环境，避免了应用的重新部署，配合敏捷定制可大幅减少代码量和系统缺陷。多数据源接入，即支持接入结构化、非结构化、时序数据库等十余种异构数据源。

本实验需要用到相关可视化组件对实验结果进行辅助验证，单独开发相应可视化工具较为繁杂，而 DWF 的上述性质，决定了本实验可以利用 DWF 快速定制流数据的可视化展示页面。

3　增量式快速傅里叶变换

3.1 节阐明核心算法框架的选择标准，并介绍基于 Apache Storm 的快速傅里叶变换算法的实现；3.2 节介绍基于 Apache Storm 的增量式快速傅里叶变换的算法实现。

3.1　基于 Apache Storm 的快速傅里叶变换算法

因 FFT 是递归算法，Apache Storm 在直接实现 FFT 时，将在其逻辑拓扑中产生环结构，不符合 Apache Storm 计算框架的逻辑拓扑的要求。

因此，需要将递归形式转换为非递归形式。根据 Cooley-Turkey 算法，将原来的序列下标的后 $\log_2 N$ 位进行翻转（N 为数据长度），就可以得到迭代后该元素在输出结果中对应的下标。另一方面，每一层进行归并其实就是在合并两个规模

相同的子问题，根据蝶形运算的规律，可以给出如下非递归快速傅里叶变换代码。

算法 1 CT-FFT 算法

```
1. procedure CT-FFT(X)
2.   len←X.length
3.   X′←BitReverse(X)
4.   for i=2,len,i≪=1 do
5.     W_0←e^{-2iπ/n}
6.     for j=i,len,j+=i do
7.       for k=j,j+i/2,k++ do
8.         W←W_0^k
9.         u←X′[k]
10.        t←W * X′[k+i/2]
11.        X′[k]←u+2
12.        X′[k+i/2]←u−t
13.      end for
14.    end for
15.    end for
16.    return X′
17. end procedure
```

算法 1 首先对输入的时间序列 X 进行字节翻转，接着从长度为 $n=2$ 的子序列开始计算，直到数组处理完毕。

上述非递归形式的 FFT 可以在 Apache Storm 中实现。在本算法中，除去序列的输入，算法可以大致分为 3 个模块：字节翻转模块、按照 CT-FFT 算法合并的模块、调整合并后的序列顺序并输出。根据以上 3 点，本文构建了 3 个不同的处理模块进行分别计算，程序的拓扑如图 4 所示。

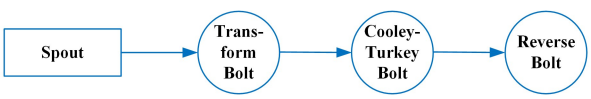

图 4 基于 Apache Storm 的 FFT 拓扑

Fig. 4 FFT topology based on Apasche Storm

3.2 基于 Apache Storm 的增量式快速傅里叶变换算法

在实际的生产环境中，传感器信号一般是高频采集的时间序列数据，数据前后关联且连续不断，在数据传输和计算的过程中很难保证数据不丢失和不产生乱序，难以直接采用 3.1 节的基于 Apache Storm 的 FFT 拓扑进行大规模并行计算。本节将研究需要对前后互相关联且连续不断的数据点进行 FFT 的算法，即增量式的快速傅里叶变换算法。

为了适应连续不断进入系统的数据点要求，系统设计了一个队列结构，当一个新的数据到达时，队首的节点出队，新的数据点加入到队尾，队列长度前后不变。但是，上述设计也产生了线程安全问题。以图 5 为例，假设在 t_0 时刻，原始数据序列 $X=\{x(0),x(1),x(2),\cdots,x(n)\}$，在 t_1 时刻接收到新的数据点 $x(n+1)$，此时新序列应为 $X'=\{x(1),x(2),x(3),\cdots,x(n+1)\}$。两序列分别对应的傅里叶变换结果为 X 和 X'。

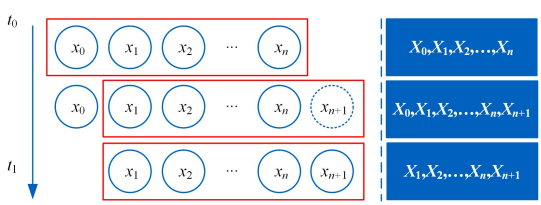

图 5 增量式 FFT 中的线程安全问题

Fig. 5 Thread safety issue of incremental FFT

由于 Apache Storm 的计算程序分布在不同的服务器上，彼此相互独立，或者在同一服务器中，但因为线程安全问题，可能会产生以下两个现象：

(1)$x(0)$未出列就进行了计算，并被认为是出列后的序列变换得到的值；

(2)$x(0)$出列了但 $x(n+1)$未立即入列，导致序列长度不满足变换运算的要求。

为了解决这个问题，需要创建和维护一个队列，使得 X 和 X'不相同的部分在运算完成之后出队，而新增的数据在计算前入队。在此过程中，同时对整个队列上锁，不允许 FFT 的计算过程对该队列进行操作。当新的队列构建完成时，在内存中将拷贝一份该队列的副本，并将这一副本发送给后续的 Bolt 进行 FFT 计算，原始的队列则进行下一次的出队和入队。

因此，我们将对上一节中的程序结构进行调整，使其适应增量式的计算，结构示意图如图 6 所示。

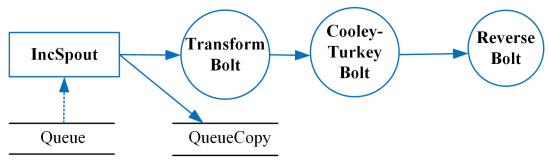

图 6 基于 Apache Storm 的增量 FFT 拓扑

Fig. 6 Incremental FFT topology based on Apache Storm

相对上一节中的内容，该结构主要进行了以下两方面修改：

(1)新增了一个队列 QueueCopy，用来对序列的入队和出队进行操作，通过加锁确保线程安全；

(2)原 Spout 被替换为 IncSpout，该模块除了分发数据，还要对序列进行入队和出队操作，并将处理完成后的队列复制给 QueueCopy，供其他模块使用。

以上为基于 Apache Storm 计算框架实现的增量式 FFT。

4 实验及分析

本节对前文设计的算法进行系统性的测试，以得到其具体的性能指标并了解性能瓶颈，以便于后续根据具体的项目进行调优。

4.1 实验硬件环境的搭建

采用 Bently-RK4 转子实验台模拟转子不对中故障，实验台由转子系统和振动测试系统组成，如图 7 所示。转子系统包括转子、质量盘、电机、轴承座和基座；振动测试系统包括 6 组位移传感器、数据采集模块和上位机。其中的 1—4 号传感器以 45°和 135°方向放置并分为两组，用于采集转子振动信号；5 和 6 号传感器分别用于测量相位和获取转速。

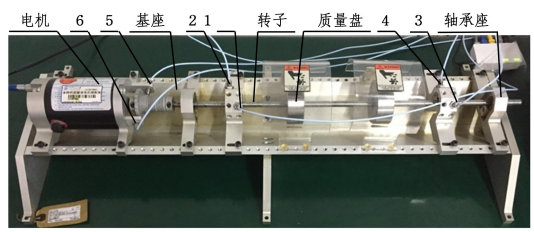

图 7 Bently-RK4 转子实验台

Fig. 7 Bently-RK4 rotor experimental platform

采用 NI CompactDAQ 进行振动数据的采集。

采用的计算集群由 3 台服务器组成，配置为：CPU：Intel (R) Xeon(R) CPU E5-2650 v4 @ 2.20 GHz×2，共计 24 核 48 线程；内存：128 GB；硬盘：2 TB，7200 转，SAS 盘×4。

4.2　实验软件环境的搭建

本次实验中所使用的 Apache Storm 1.2.1 版本，运行的 JDK 为 1.8 版本，操作系统为 Ubuntu 16.04。

实验还需结合其他大数据软件接收和发送数据，从而搭建一整套可以在线实时统计分析数据并生成所需图像的系统。本文搭建的系统框架如图 8 所示。

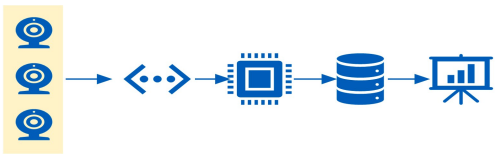

图 8　系统框架示意图

Fig. 8　Architecture of system

系统框架包括如下几个部分。

Kafka：接收传感器数据，并作为传感器和实时计算模块之间的数据管道，对传输速度较快的时序数据可以起到缓冲作用；

Apache Storm 集群：将运行 FFT 算法的模块与其他相关模块连接起来，并提高运算效率；

数据库：对于本文介绍的算法以及应用而言，选取时序数据库作为系统的基础架构是十分合适的；

数据展示：本文选择清华数为系统集成框架(DWF)作为数据展示模块的基础架构。

4.3　实验内容

本节将测试第 3 节的增量式 FFT。

一般来说，设备生产商通常会在转子的水平方向(x 轴方向)和垂直方向(y 轴方向)安装两个传感器，用于实时记录转子的振动信号。通过傅里叶变换将原始的时域振动信号转换成为频域信号，提取某些特定频率的信号逆变换后合成轴心轨迹图像，如果呈现明显的“香蕉形”或者是“8 字形”，则可以认为存在转子不对中。

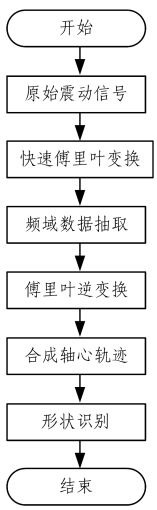

图 9　合成轴心轨迹流程

Fig. 9　Compositing axis orbit

对于一个转子，在某一刻采集的水平方向传感器信号如图 10 所示。

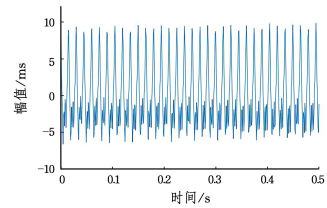

图 10　水平方向振动信号

Fig. 10　Horizontal vibration signal

该信号采样频率为 4 000 Hz，数据长度为 1 024。经过 FFT 变换之后，可以得到图 11 所示图像。

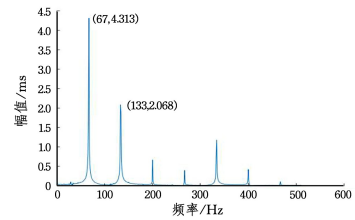

图 11　水平信号 FFT 的变换结果

Fig. 11　Horizontal vibration signal after FFT

图 11 是水平方向的振动信号经过了 FFT 之后得到的结果，垂直方向的信号类似。由于该信号的采样频率为4000 Hz，因此只有频率为 66.67 Hz 及其倍数是有实际意义的。这里选取幅值最大的两个频率：$f=67$ Hz，$f=133$ Hz。

将其他频率分量全部置 0，进行快速傅里叶逆变换(Inverse Fast Fourier Transform，IFFT)，就得到相应频率的合成轴心轨迹，如图 12 所示。

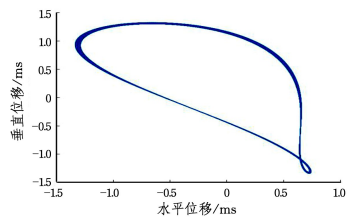

图 12　合成轴心的轨迹图

Fig. 12　Trail of composited axis orbit

通过上述转换得到的合成轴心轨迹呈现出了“8”字形，因此可以认为在该时刻转子存在着不对中的问题。而通过前文所述系统可以实现本文中提及的对给定的时间序列计算其合成轴心轨迹，并且进行可视化的功能，结果如图 13 所示。

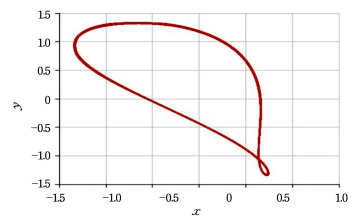

图 13　合成轴心轨迹的可视化

Fig. 13　Visualization of result axis trajectories

可以看出，图 12 和图 13 所呈现结果是相同的。用户可以通过该结果判定设备存在转子不对中的问题，并对设备进行维护。这验证了 4.2 节系统框架的可行性，也验证了基于 Apache Storm 的增量式 FFT 可以实时更新流数据频谱，产生正确的转子合成轴心轨迹。

(下转第 540 页)