基于MapReduce的傅里叶变换的设计与实现

指导教师：李天瑞

- 班级：软件二班
- 学生：李明
- 学号：20092378

Contends

课题研究的背景和意义

- 背景

2004年，Google共享了其在集群上运行的分布式计算 模型MapReduce，由于其框架的开源性、简洁性等优势，Google搜索引擎海量数据下的成功实践，MapReduce一经提出就受到强烈关注。

傅里叶变换是一种积分变换，在许多领（物理学、数论、组合数学、信号处理、概率论、统计学、密码学、声学、光学、海洋学、结构动力学等）都有着广泛的应用，但其往往受到海量数据等问题困扰。

课题研究的背景和意义

- 意义

本课题将以MapReduce编程模型为基准，探讨傅里叶变换的并行化工作，设计与实现基于MapReduce模型的傅里叶变换算法，为海量数据的傅里叶变换提供有效方法。

DFT与FFT的简介

◆DFT

离散傅里叶变换（DFT），是傅里叶变换在时域和频域上都呈离散形式。其计算公式如下：

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk}$$

式中 $W_N=e^{-j\frac{2\pi}{N}}$

$$k=0,1,\ldots,N-1$$

DFT与FFT的简介

◆FFT

FFT算法由库里和杜克于1965年提出，其加快了数字信号分析过程，为后期信号处理，如滤波、频谱分析奠定了基础。

FFT并不是一种新的算法，只是DFT的一种快速计算方法，也不能计算海量数据。

DFT与FFT的简介

◆FFT发展现状

信号序列长度N等于2的整数次幂情况，如基-2和基-4算法等，将长DFT序列按奇偶位置分解出更小点数的DFT短序列

信号序列长度*N*不等于2的整数次幂情况，以威诺格兰德为代表所提出的PFTA算法，利用下标映射和数论知识，去掉旋转因子，减小运算量，但是控制复杂

为了能够处理更大规模的信号数据以及提升算法执行效率，并行FFT算法也得到诸多发展，如多线程FFT算法和网格FFT算法等

基于Hadoop平台的并行FFT的设计

◆Hadoop简介

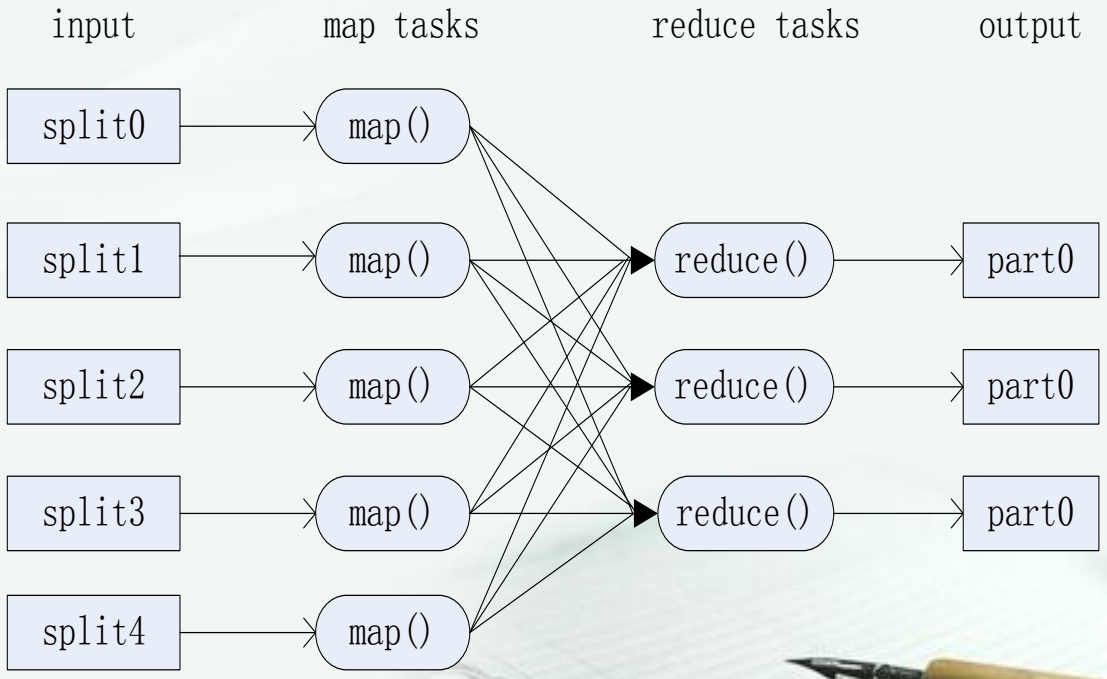
Hadoop是一个开源的基于MapReduce模型的开发平台。Hadoop起源于Apache Nutch，一个开源的网络搜索引擎，它本身也是Lucene项目的一部分。至2006年逐渐成为一套完整而独立的软件，起名为Hadoop。它主要包括两个模块：分布式文件系统HDFS和并行编程模型MapReduce

基于Hadoop平台的并行FFT的设计

◆MapReduce模型

MapReduce过程

1. map和reduce函数的输入输出都是以<key,value>键值对的方式进行的
2. 每个split对应一个map函数,map函数的结果传给reduce函数处理
3. 系统把相同key值得map结果传给同一个reduce函数



基于Hadoop平台的并行FFT的设计

◆基-2FFT

序列长度N满足2^L 的形式，时域上按奇偶位置抽取出短序列DFT，递归分解L次，直到分 解为2点DFT为止

由于基-2时间抽取FFT结构清晰，运算方法明确，故此FFT算法被广为应用

基于Hadoop平台的并行FFT的设计

◆并行FFT整体流程图

基于Hadoop平台的并行FFT的设计

◆Job1-补齐运算

算法设计思想

1.判断输入数据是否为2^L

2.对于不满足2^L的，求出最小的L并用0补齐

3.对每组数据加一个编号，为变址运算做准备

基于Hadoop平台的并行FFT的设计

◆Job2-变址运算

算法设计思想

1.这里巧妙用到了Hadoop自带的排序功能，把<编号，测试数据>设置为<key,value>键值对形式

2.对编号进行变址运算，然后系统就会根据key值从新排序也就实现了变址运算

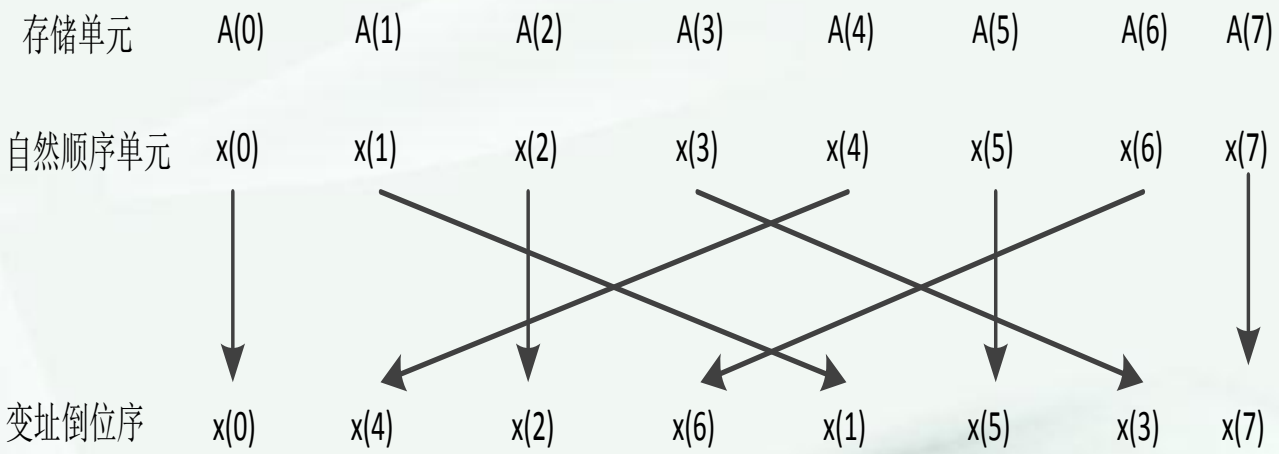
基于Hadoop平台的并行FFT的设计

当N为8时对编号进行倒位序

自然序列n	二进制数	倒位序二进制数	倒位序顺序数n_1
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

基于Hadoop平台的并行FFT的设计

◆对N为8的变址运算效果如下图所示：



基于Hadoop平台的并行FFT的设计

◆Job3-蝶形运算

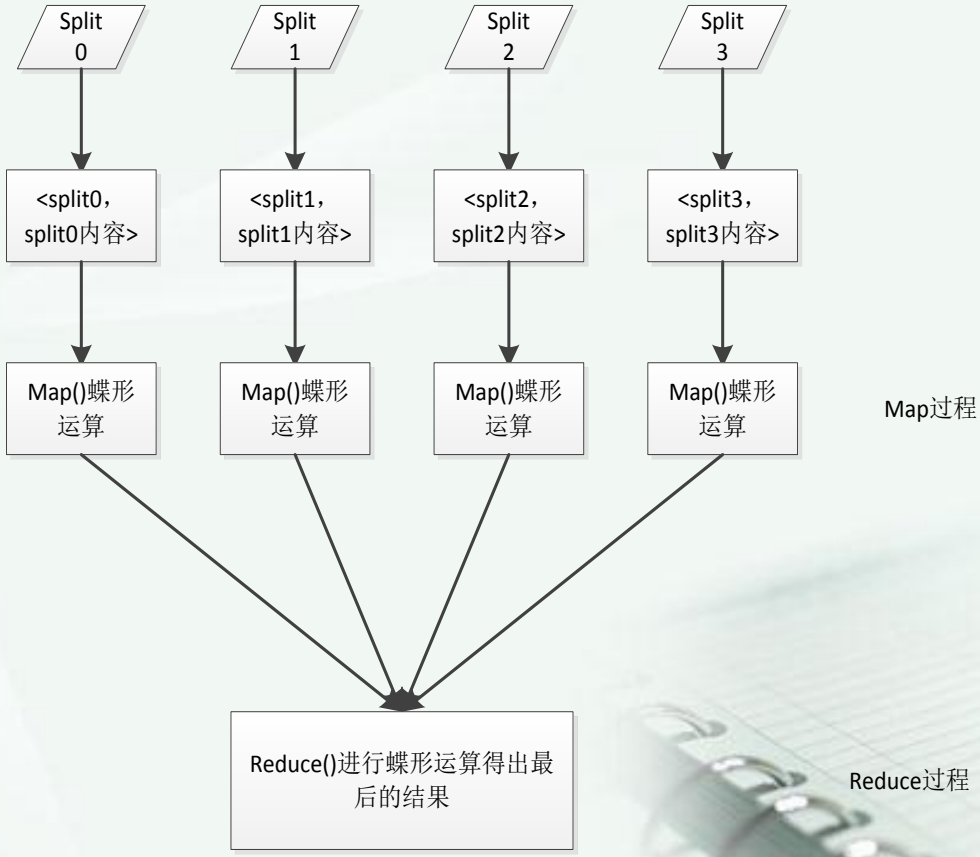
算法设计思想

1.由于FFT运算的是复数，所以本文构建的一个新的Writable类型为TextPair，用来操作两个字符串。

2.对输入数据进行迭代运算

基于Hadoop平台的并行FFT的设计

◆Job3的工作流程如下图所示



并行FFT算法的实验与评测

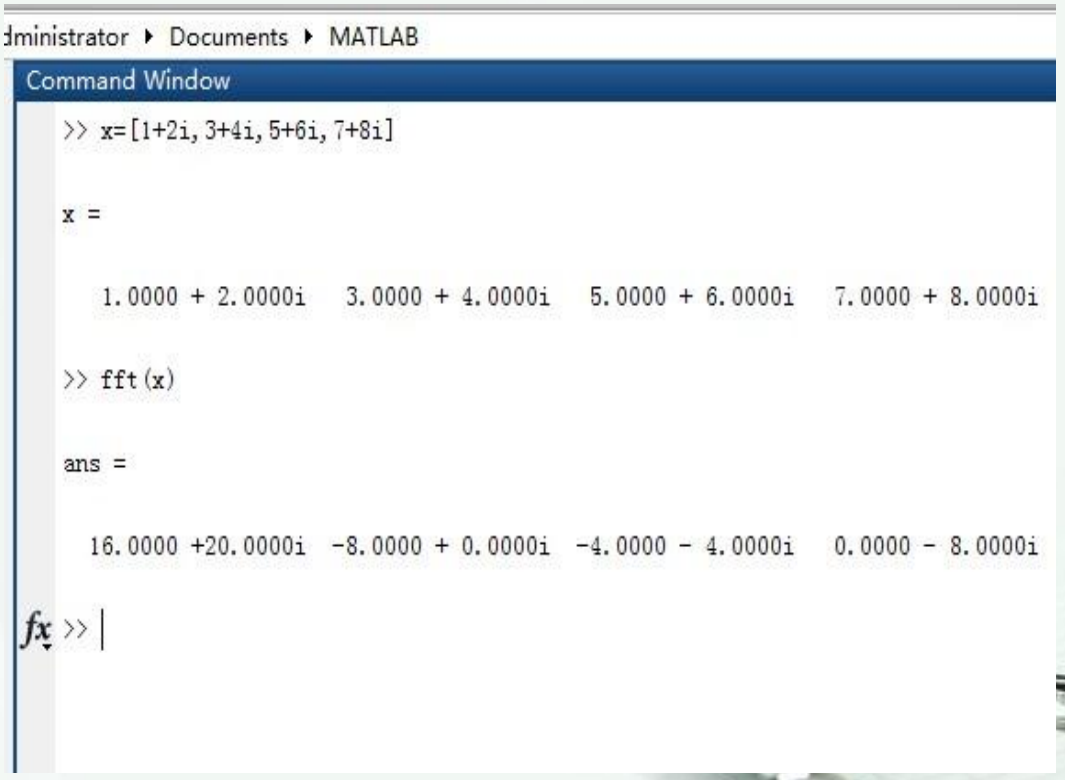
实验集群配置情况

节点个数	4
Hadoop版本	Hadoop 1.1.2
Java版本	JDK 1.7.0
节点CPU	Intel Core 2.93GHz
节点内存	4 GB

并行FFT算法的实验与评测

◆准确性评测

小数据评测：当x(n)={1+2i,3+4i,5+6i,7+8i}时



MATLAB计算结果

并行FFT算法的实验与评测

◆准确性评测

小数据评测：当x(n)={1+2i,3+4i,5+6i,7+8i}时

```
13/05/30 15:52:45 INFO mapred.JobClient:   Job Counters
13/05/30 15:52:45 INFO mapred.JobClient:     Launched reduce tasks=1
13/05/30 15:52:45 INFO mapred.JobClient:     Launched map tasks=1
13/05/30 15:52:45 INFO mapred.JobClient:     Data-local map tasks=1
13/05/30 15:52:45 INFO mapred.JobClient:   FileSystemCounters
13/05/30 15:52:45 INFO mapred.JobClient:     FILE_BYTES_READ=62
13/05/30 15:52:45 INFO mapred.JobClient:     HDFS_BYTES_READ=25
13/05/30 15:52:45 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=156
13/05/30 15:52:45 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=38
13/05/30 15:52:45 INFO mapred.JobClient:   Map-Reduce Framework
13/05/30 15:52:45 INFO mapred.JobClient:     Reduce input groups=1
13/05/30 15:52:45 INFO mapred.JobClient:     Combine output records=0
13/05/30 15:52:45 INFO mapred.JobClient:     Map input records=4
13/05/30 15:52:45 INFO mapred.JobClient:     Reduce shuffle bytes=0
13/05/30 15:52:45 INFO mapred.JobClient:     Reduce output records=4
13/05/30 15:52:45 INFO mapred.JobClient:     Spilled Records=8
13/05/30 15:52:45 INFO mapred.JobClient:     Map output bytes=48
13/05/30 15:52:45 INFO mapred.JobClient:     Combine input records=0
13/05/30 15:52:45 INFO mapred.JobClient:     Map output records=4
13/05/30 15:52:45 INFO mapred.JobClient:     Reduce input records=4
root@ubuntu:~/hadoop-0.20.2# bin/hadoop fs -cat output03/*
16.0    20.0
-8.0    0.0
-4.0    -4.0
0.0     -8.0
```

并行计算结果

并行FFT算法的实验与评测

◆准确性评测

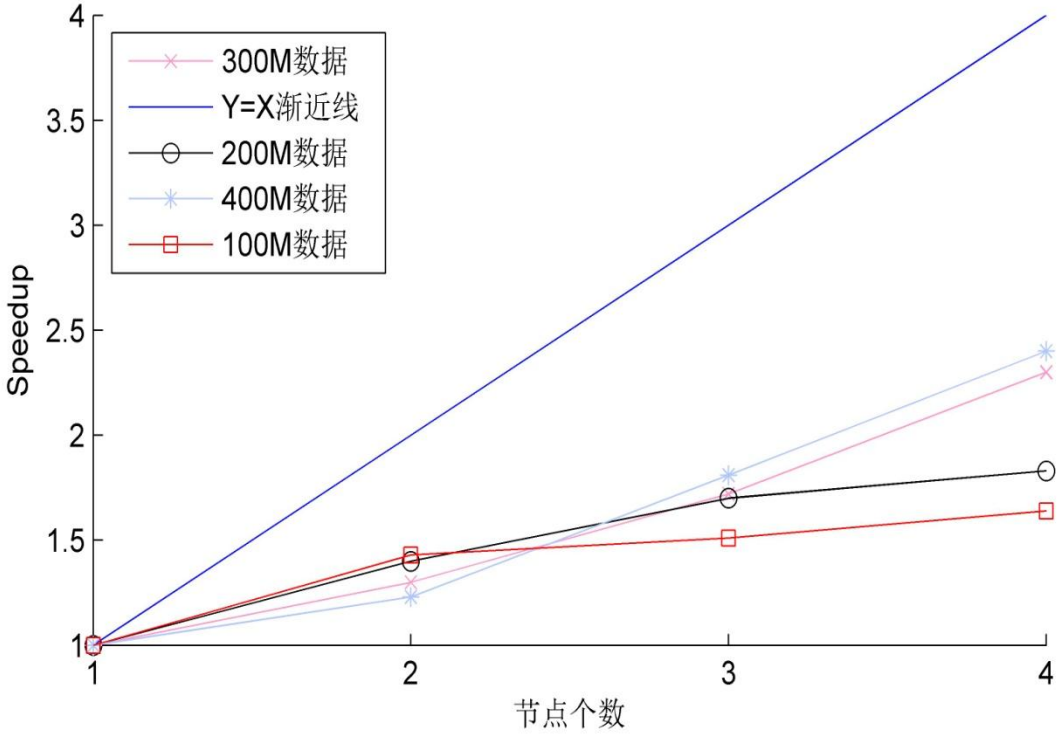
本文还进行了大数据测试，与MATLAB比较了前100项计算结果，结果全部一致。

◆结论

通过与MATlAB的计算结果进行比较，并行FFT的结果与其结果一致，说明算法在准确性方面具有可靠性

并行FFT算法的实验与评测

◆并行FFT效率评测-speedup

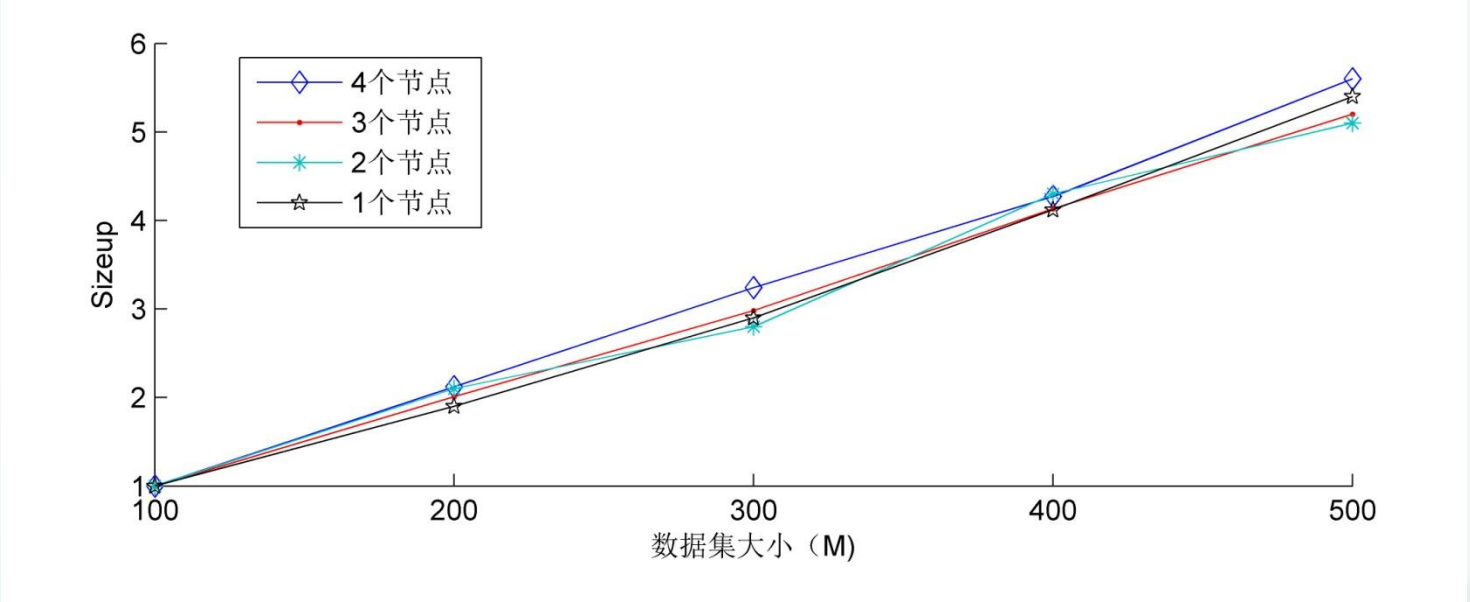


◆结论

从图中可以看出，**speedup**效果不是很理想，单个节点处理大块数据，负载压力过大，导致并行性效果不理想。

并行FFT算法的实验与评测

◆并行FFT效率评测-sizeup



◆结论

sizeup反应的算法自身的复杂度，随着数据集大小的增加sizeup呈线性增长，可见sizeup特性良好。

结论与展望

◆结论

本文实现了基于MapReduce的FFT算法，能处理MATLAB不能进行计算的大数据。但在算法性能评测的时候表现不是很好。

◆展望

并行FFT算法的并行效率需要进一步提高，算法设计过程中，减少迭代次数，增加并行性，想办法避免在reduce函数里进行蝶形运算。

致谢

由衷地感谢李天瑞老师在毕业设计对我的支持与帮助。

感谢实验室的老师及学长学姐在学习和生活方面给予我帮助。

最后感谢在座的评审老师，谢谢你们在百忙之中抽空参加我的答辩评审会

Thank You !