西南科技大学本科毕业设计（论文）开题报告

学　　院	计算机科学与技术学院	专业	软件工程	班级	软件 1804
姓　　名	肖劲涛	学号	5120184509	指导教师	苏波
设计（论文）题目	快速傅里叶变换的并行算法研究及实现				

一、选题背景（目的、意义）

选题背景：

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的形式，是科学与工程领域中一个重要的数学方法[1-3]。给定一个长度为 N 的队列 $x(n)$ ，它的 DFT 也是一个长度为 N 的队列 $X(k)$ ，

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk},\quad k=0,1,...,N\text{-}1 \qquad (式 1)$$

其中，$W_n=e^{-j2\pi/N}$ ，$j=\sqrt{-1}$ 。傅里叶变换认为，任何连续的时域信号，都可以表示为不同的正弦波信号的叠加。傅里叶变换算法直接利用时序采样信号，以累加的方式计算时序采样信号中不同正弦波的频率信号的振幅、频率以及相位。通过这种方法，傅里叶变换将原本难以处理的时域信号变换为易于处理的频域信号，可以利用一些工具对其加工、处理。最后可以通过傅里叶逆变换，将频域信号重新变换为时域信号。傅里叶逆变换从本质上来说，也是一种累加处理算法。

在不同的研究领域，傅里叶变换具有多种描述形式。用现代数学的观点看，傅里叶变换将满足一定条件的函数表示为积分或正弦函数的线性组合，从这点来看，傅里叶变换可以视作一种积分的特殊形式。

DFT 通常使用一种叫快速傅里叶变换（FFT）的技术来实现，即 FFT 是 DFT 的快速算法。同时，FFT 也可用作 DFT 的逆变换。直接计算 DFT 的时间复杂度是 $O(N^2)$，而使用 FFT 的时间复杂度则是 $O(NlogN)$。通常，快速傅里叶变换要求 N 内部因子分解，但不是所有的 FFT 都要求 N 是合数。对于所有的整数 N，都存在复杂度为 $O(NlogN)$的 FFT。

随着近些年来数字计算机和大规模电路的不断发展，中央处理单元（CPU）与图形处理单元（GPU）的核心数量与核心频率得到了显著提高。如今，我们面临的问题的数据规模日益增加，传统的串行算法在面对大量数据时不能充分利用处理器的多个内核。这要求编写软件程序的思想发生变化，以适应硬件的变化，并充分利用硬件的性能。同时，针对大规模的 FFT 计算，GPU 相比于 CPU 在处理能力和存储带宽上有较为明显的优势。CPU 的设计兼顾了不同任务的需要，其晶体管都用于大量的缓存和复杂的逻辑控制，相对来说运算单元所占不多。多核 CPU 的发展使得其晶体管数量得到了增加，但并没有增加 CPU 的利用率。而

GPU 提供了跟多的计算单元和存储控制单元，使得在大规模数据计算以及存储带宽得到提高。CPU 程序通常时单线程编写，如果有需要则可以使用多进程、多线程编程技术，而 GPU 则默认并行计算模式。得益于其结构设计，GPU 使得每次可以同时计算多个数据，而不是像 CPU 一次只能计算一个数据。但与之相对的，则是在复杂计算与指令上，GPU 的计算速度不如 CPU。

选题意义：

傅里叶变换在信号处理、力学、数学、金融等领域都有着广泛的应用[4-7]。傅里叶变换的一个典型用途是，将时域信号分解为为频域内振幅分量和频率分量。直接计算长度为 N 的序列卷积的离散傅里叶变换（DFT）的复杂度为 $O(N^2)$。Cooley-Tukey 快速傅里叶变换(FFT) [8]将计算复杂度降低到了 $O(NlogN)$。

尽管 FFT 表现不凡，但还是有很多问题需要解决。这些方面包括：计算精度的提高、高效低耗的 FFT 处理器设计、计算复杂度在理论与实际的最小边界、FFT 数据迁移量的减少、系数访问效率的改善等。本项目将对比不同的傅里叶变换算法在不同的程序设计方案（串行、并行等）下的性能，并给出指导性报告。

二、国内外研究现状综述

快速傅里叶变换最早由 Guass 提出，后来又被 Runge 和 Konig 发现[9]。但直到 1965 年，快速傅里叶变换被 Cooley-Tukey 重新提出后才得到了充分认识[8]。从此以后，关于快速傅里叶变换的研究快速增加，包含了高阶基-K FFT 算法[10]、分裂基 FFT 算法[11-14]、混合基算法[15, 16]、质数因子算法[17-19]、递归 FFT 算法[20]、Winograd 傅里叶变换算法[21, 22]。

高斯在 18 世纪使用和 Cooley-Tukey 相同的分治法计算三角级数。值得注意的是，因其算法基于任何长度的整数的队列变换，其拥有很高的通用性。在 1903 年，Runge 提出了一个用来计算 2 的幂次方的队列长度算法，该算法也能用来计算 3 的幂次方。1965 年，Cooley 和 Tukey 提出了一种快速傅里叶变换算法[8]，该方法减少了长度为 $N=2^m$ 的 DFT 操作数的阶数，使其从 N^2 降低到了 $Nlog_2N$。同时，该算法适应任何长度的 DFT。

当一个 DFT 被分解为一些不是互素的子 DFT 时，分治会导致辅助的复数乘的出现，最初被叫做旋转因子。Coolkey-Tukey 算法适用于任意长度的 DFT，按通用形式解释。Coolkey 和 Tukey 给出了一个长度为 $N=2^m$ 的例子，即提出了一个现在叫按时域分解的基-2FFT 算法。

1965 年，Coolkey-Tukey 算法可以理解为一个假的一维到多维的镜像。而 Good 算法[23]可以在变换长度为因子互素乘的 DFT 中，实现真正的一维到多维的镜像，并且不产生旋转因子。Good 算法适应的分解长度，其因子时互素的；并不适用长度为 K 的幂次方的情况。Rader 的方法[10]展示了如何将一个长度为 N 的 DFT 镜像为一个长度为 $N\text{-}1$ 的循环卷积，其中 N 为素数。Winograd 算法[22]，首先使用 Good 算法镜像一个 DFT 成多维的 DFT 后，使用卷积方案来得到一个多种乘法的循环。9

Jiang 在文章[24]中提出，以前的 FFT 方法在存储访问问题。除非处理器提供了大量寄存

器，否则重复的访问寄存器去装载旋转因子时不可避免的。同时，Jiang 提出了基-2FFT 宽度优先算法。N-点基-2FFT 算法的两个情况考虑宽度优先算法，即 W^0 和 $W^j(j \neq 0)$ 。对应的 FFT 结构被组织成两个阶段，第一阶段计算所有需要被 $W^j(j \neq 0)$ 乘的蝶。在该阶段，系数 W^j 被装入寄存器，一直使用到当前处理层不需要使用到为止。该方法系数被装载的次数为 $N/2-1$，前面提到的方法则是 $NlogN$ 次。第二个阶段计算所有需要被 W^0 乘的所有蝶。

麻省理工大学 Frigo 和 Johnson 开发的 FFTW 软件库[1, 2]，可以计算任意长度、一维或多维的实数或复数 DFT。FFTW 通过支持多种算法并选择它估计或测量在特定情况下更可取的算法（将转换特定分解为更小的转换）来快速转换数据。它在具有小素数因子大小的数组上效果最佳，2 的幂次方和大素数情况下效果最差，但其复杂度仍然是 $Nlog_2N$。该软件库使用 Cooley-Tukey 算法、Rader 算法或 Bluestein 算法。

英伟达公司提供了一种基于 GPU 的 FFT 算法：CUFFT。CUFFT 是 CUDA 的函数库，其相对于 CPU 在运算速度上有着明显优势，但仍然未能充分发挥 GPU 在并行计算方面的优势。Govindaraju、Lloyd 和 Dotsenko 提出了一种在 GPU 上运行的分层混合基 FFT 算法，该算法通过 Stockham 公式利用 GPU 上的共享内存来减少分层算法中的存储器转置开销。相比于 CUFFT 算法，该算法的性能提高了 2-4 倍。Gu、Li 和 Siegel 提出的一种基于多维 Cooley-Tukey 算法的 GPU 实现方案，通过减少计算内核的数量，并优化最小全局存储器的访问数量来提高算法效率，该算法相比于 CUFFT 算法有着精度上的优势。

三、研究目标与研究内容

研究目标：

本项目内容主要为基于对 CPU 与 GPU 上 FFT 的并行算法的研究。同时介绍几种典型的实现多线程、多进程编程方法。在此基础上分析 FFT 算法的可行性，并给出多线程优化。

具体研究内容包括：

(1) 介绍多种传统串行 FFT 算法，包括 Cooley-Tukey 算法、Rader 算法等。

(2) 介绍多种并行技术下的 FFT 算法，包括 CUFFT 算法、Govindaraju-Lloyd-Dotsenko 算法等。

(3) 比较不同算法之间的性能，评估方法有：蝶分析、算法复杂度分析、旋转因子分析等。

(4) 通过不同算法直接比较，给出并行 FFT 算法的选择、优化建议。

四、拟采用的研究思路（方法、技术路线、可行性论证等）

(1) 查询文献。利用图书馆、档案馆及互联网等途径，广泛查找相关的文献资料，加以分析与研究。

(2) 构建模型。使用 C++编写串行的 FFT 算法；利用 MPI 或 OpenMP，同时使用 C++实现基于 CPU 的多进程的 FFT 算法；利用 C++的 STL 库实现基于 CPU 的多线程 FFT 算法；利用 CUDA 与 C++实现基于 GPU 的并行 FFT 算法。

(3) 算法分析。通过不同的指标参数，分析各 FFT 算法的性能。

五、研究工作进度安排

(1) 2022 年 1 月 20 日—2022 年 2 月 10 日 查阅有关资料，明确课题研究的目的及意义。

(2) 2022 年 2 月 11 日—2022 年 2 月 21 日 编写串行 FFT 算法。

(3) 2022 年 2 月 22 日—2022 年 4 月 1 日 编写并行 FFT 算法并进行需求分析。

(4) 2022 年 4 月 1 日—2022 年 4 月 19 日 分析各 FFT 算法的性能，并给出意见与建议。

(5) 2022 年 3 月 19 日—2022 年 5 月 15 日 根据论文撰写规范，完成初稿。

(6) 2022 年 5 月 16 日—2022 年 6 月 1 日 完善论文，准备答辩。

六、参考文献

[1] FRIGO M, JOHNSON S G. FFTW: An adaptive software architecture for the FFT; proceedings of the Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat No 98CH36181), F, 1998 [C]. IEEE, 1998.

[2] FRIGO M, JOHNSON S G. The design and implementation of FFTW3 [J]. Proceedings of the IEEE, 2005, 93(2): 216-231.

[3] KATOH K, KUMA K-I, TOH H, et al. MAFFT version 5: improvement in accuracy of multiple sequence alignment [J]. Nucleic acids research, 2005, 33(2): 511-518.

[4] 任山. 基于查找表的 FFT CUDA 并行算法研究 [D]. 长沙:湖南大学, 2017.

[5] 徐金棒. 基于多核多线程的 FFT 算法和堆排序算法的并行优化和实现 [D]. 郑州:郑州大学, 2011.

[6] 郑伟华. 快速傅立叶变换-算法及应用 [D]. 长沙:湖南大学, 2015.

[7] 周益民. 图像处理并行算法的研究 [D]. 长沙:电子科技大学, 2006.

[8] COOLEY J W, TUKEY J W. An algorithm for the machine calculation of complex Fourier series [J]. Mathematics of computation, 1965, 19(90): 297-301.

[9] PRESS W H, TEUKOLSKY S A, FLANNERY B P, et al. Numerical recipes in Fortran 77: volume 1, volume 1 of Fortran numerical recipes: the art of scientific computing [M]. Cambridge Eng: Cambridge university press, 1992.

[10] RADER C M. Discrete Fourier transforms when the number of data samples is prime [J]. Proceedings of the IEEE, 1968, 56(6): 1107-1108.

[11] BOUGUEZEL S, AHMAD M O, SWAMY M. Improved radix-4 and radix-8 FFT algorithms; proceedings of the 2004 IEEE International Symposium on Circuits and Systems (IEEE Cat No 04CH37512), F, 2004 [C]. IEEE, 2004.

[12] BOUGUEZEL S, AHMAD M O, SWAMY M. A general class of split-radix FFT algorithms for the computation of the DFT of length-2^m[J]. IEEE Transactions on signal processing, 2007, 55(8): 4127-4138.

[13] BOUGUEZEL S, AHMAD M O, SWAMY M S. A new radix-2/8 FFT algorithm for length-q/spl times/2/sup m/DFTs [J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2004, 51(9): 1723-1732.

[14] DUHAMEL P, HOLLMANN H. Split radix'FFT algorithm [J]. Electronics letters, 1984, 20(1): 14-16.

[15] HSIAO C-F, CHEN Y, LEE C-Y. A generalized mixed-radix algorithm for memory-based FFT processors [J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2010, 57(1): 26-30.

[16] JO B G, SUNWOO M H. New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy [J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2005, 52(5): 911-919.

[17] BLUESTEIN L. A linear filtering approach to the computation of discrete Fourier transform [J]. IEEE Transactions on Audio and Electroacoustics, 1970, 18(4): 451-455.

[18] KOLBA D, PARKS T. A prime factor FFT algorithm using high-speed convolution [J]. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1977, 25(4): 281-294.

[19] BURRUS C, ESCHENBACHER P. An in-place, in-order prime factor FFT algorithm [J]. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1981, 29(4): 806-817.

[20] MARTENS J-B. Recursive cyclotomic factorization--A new algorithm for calculating the discrete Fourier transform [J]. IEEE transactions on acoustics, 1984, 32(4): 750-761.

[21] JIANG Y, ZHOU T, TANG Y, et al. Twiddle-factor-based FFT algorithm with reduced memory access; proceedings of the Proceedings 16th International Parallel and Distributed Processing Symposium, F, 2002 [C]. IEEE, 2002.

<table>
<tr><td>指导教师意见</td><td>该生针对快速傅里叶变换算法研究从选题背景、目的和意义以及国内外研究现状进行了综述。指出了 FFT 算法在不同行业使用的重要程度，并且指出了部分 FFT 算法在特定环境下存在的不足。归纳总结了前人在该领域中的工作成果与得失。开题报告体现出研究内容丰富和目标明确，给出的研究方案，技术路线合理，具有较强的可实施性。开题报告给出的毕业设计各个环节的进度安排时间节点合理，同意开题。

指导教师（签名）______________

2022 年 3 月 11 日</td></tr>
<tr><td>答辩小组意见</td><td>□通过 □不通过

答辩组成员（签名）__________________________

答辩组组长（签名）_______________

年 月 日</td></tr>
<tr><td>学院审核意见</td><td>分管教学院领导签字（公章）___________

年 月 日</td></tr>
</table>