2020 并行应用挑战赛

优化组决赛赛题

题目：

三维非标准快速傅里叶变换

简介：

非标准快速傅里叶变换（Non-Uniform Fast Fourier Transform, NUFFT）是一种处理非标准采样的快速算法，在信号处理方面广泛应用。快速傅里叶变换算法 FFT 广泛地应用于科学计算的各个领域，被评价为二十世纪十大算法之一。但是，FFT 算法对数据采样分布于非均匀网格上的情况时，如雷达信号处理、CT、MRI 等等，传统的 FFT 算法就束手无策了。而 NUFFT 算法就是为了解决这个问题应运而生。

本次赛题选用的背景是 3D-MRI 成像，NUFFT 算法通常使用两个 NUFFT 算子：forward NUFFT（FWD）和 adjacent NUFFT（ADJ）。

计算一个信号 f 的 forward NUFFT 包括以下三步：

（1） point-wise scaling of f by s (s is symmetric scaling function)
（2） execution of a M-point FFT
（3） convolution interpolation onto the set of $\omega's$

计算一个信号的 adjoint NUFFT 可以看做是前向 NUFFT 的逆运算：

（1） convolution interpolation from the set of $\omega's$
（2） execution of an inverse M-point FFT
（3） point-wise scaling of f by s and M

convolution 在其中是最耗时的计算，其算法如下：

```
// Part 1: Common to FWD and ADJ
// N2: width of Cartesian Grid
// wx[p]: x co-ordinate of sample
// Form x interpolation kernel
kx = wx[p]
x1 = ceil(kx − W)
x2 = floor(kx + W)
lx = x2 − x1 + 1
for i = 0 to lx − 1 do
    nx = x1 + i
    kx[i] = mod(nx, N2)
    winX[i] = LUT(abs(nx − kx))
end for
// Form y and z interpolation kernels
...
```

```
// Part 2a: Only in FWD Convolution
// Perform separable convolution
// raw[p]: Sample value
// f[x, y, z]: Cartesian Grid point
raw[p] = 0
for x = 0 to lx − 1 do
    for y = 0 to ly − 1 do
        for z = 0 to lz − 1 do
            raw[p] += f[kx[x], ky[y], kz[z]]
                * winX[x] * winY[y] * winZ[z]
        end for
    end for
end for
```

```
// Part 2b: Only in ADJ Convolution
// Perform separable convolution
// raw[p]: Sample value
// f[x, y, z]: Cartesian Grid point

for x = 0 to lx − 1 do
    for y = 0 to ly − 1 do
        for z = 0 to lz − 1 do
            f[kx[x], ky[y], kz[z]] += raw[p]
                * winX[x] * winY[y] * winZ[z]
        end for
    end for
end for
```

Figure 2. Convolution Code

Convolution 的并行方法可以把所有采样分配到每个并行线程中计算，但是在相邻的点会出现互斥操作，影响并行效率。优化方法可参考文献 [1]，使用数据块划分进行多线程设计，并利用 Intel 的 AVX 指令集进行 SIMD 向量化优化。

[1] Dhiraj D. Kalamkar, Joshua D. Trzasko, et al. High Performance Non-uniform FFT on Modern x86-based Multi-core Systems, IPDPS 2012.

程序说明：

代码为 C++代码，已经实现 OpenMP 功能，源代码在根目录下，输入数据在 data 目录下，编译需要安装 boost 和 fftw。

使用最新的 Intel 编译器

编译：

- 使用 source 命令加载计算节点或编译节点的 Intel 编译器
- 默认基准编译选项为：

  -Wall -O3 -xCore-AVX512 -fopenmp -fp-model precise

- 编译命令：make

编译完成后，会在当前 NUFFT 目录下生成可执行代码：Demo

运行：

./Demo data/Orig $NUM_THREAD $NUM_ITERATION

$NUM_ITERATION 为迭代次数，最大设为 50。

输入数据为四组：

(1) Orig: Orig.coord, Orig.KS, Orig.raw;
(2) Random: Random.coord, Random.KS, Random.raw;
(3) Radial: Radial.coord, Radial.KS, Radial.raw;
(4) Spiral: Spiral.coord, Spiral.KS, Spiral.raw;

Orig 数据规模最小，可以作为调试数据集。

正确性验证：

规则：

1. 可以用其它功能相同的库或自行实现的函数替换 boost 和 fftw。
2. 计时由输入文件读入介绍开始，到开始输出结果文件前结束，禁止改变计时函数位置，一经查实，该题目成绩作废。
3. 最终优化成绩比较，以程序计时输出时间为评测依据。时间越短得分越高。
4. 代码提交后，由评委/组委会对代码进行验证和进行其他算例以验证代码正确性。规则最终解释权在优化组评委