从benchmark开始做Stdpar模型的迁移

方式：将benchmark的workload从其他模型迁移到stdpar模型上——学习其他模型的运算方式并使用PSTL库实现。

1.尝试较为简单的BabelStream、miniBude

都由UoB-HPC开发，模式很相近——实现不同模型的测试代码，在构建时由用户选择模型进行测试。

BabelStream – 使用stream data进行计算（copy\add\mul\triad\dot）并输出运算时间

发现BabelStream已经支持std-data、std-indices、std-ranges模型，三者均为std parallel stl模型，其中std-indices使用自定义类ranged管理迭代器；std-ranges使用了c++20的ranges特性（views）

exe\_policy = std::execution::par\_unseq

std-data:std::fill(exe\_policy, a, a + array\_size, initA);

std-indices:std::transform(exe\_policy, range.begin(), range.end(), b,

[c = this->c, scalar = startScalar](int i) {return scalar \* c[i];});

std-ranges:std::for\_each\_n(exe\_policy,std::views::iota(0).begin(), array\_size,

[&] (int i) {a[i] = initA;b[i] = initB;c[i] = initC;});

std-data模型（构建时可以选择编译器、开启NVHPC(GPU后端)的offload模式、使用onedpl）

（onedpl：intel oneAPI的高性能计算库，提供了与 C++ STL 类似的接口）

使用Adaptivecpp的Stdpar模型需要在构建时指定acpp为c++编译器，并且将--acpp-stdpar –acpp-targets=...作为额外flags传入。经过测试，可以运行并成功offload

miniBude – 对NDM-1蛋白进行的虚拟筛选，反复运行单一代pose的能量评估，迭代次数可以配置

miniBude已经支持std-indices;std-ranges模型，但两个模型的运算主要使用openmp进行加速。因此尝试将其改为完全的stdpar（pstl）模型，不管miniBude的框架代码，将主要测试代码的运算逻辑通过pstl函数实现。（主要将openmp的循环修改为pstl库函数）

2.自动迁移

从openmp之类的并行模型迁移到stdpar模型显然更简单。openmp已经标识出了可以无序并行运行的代码块。

以openmp为例

示例（BabelStream测试代码）

omp -> stdpar(pstl)

#pragma omp parallel for

for (int i = 0; i < array\_size; i++)

{

a[i] = initA; std::fill(exe\_policy, a, a + array\_size, initA);

b[i] = initB; std::fill(exe\_policy, b, b + array\_size, initB);

c[i] = initC; std::fill(exe\_policy, c, c + array\_size, initC);

}

omp -> stdpar(pstl)

const T scalar = startScalar;

#pragma omp parallel for

for (int i = 0; i < array\_size; i++) std::transform(exe\_policy, c,c +array\_size,

{ b,[scalar = startScalar](T ci){

b[i] = scalar \* c[i]; return scalar\*ci; });

}

→

std::for\_each(std::execution::par\_unseq,0,array\_size,[&](int i){a[i]=initA;b[i]=initB;c[i]=initC;});

→

const T scalar=startScalar; std::for\_each(std::execution::par\_unseq,0,array\_size,[&](int i){b[i] = scalar\*c[i];});

问题在于不太清楚该怎么自动地通过循环体内的变量、语句信息选择对应的PSTL库函数实现迁移；如果不改变openmp的语义，直接使用pstl库函数（for\_each）打开循环可以一定程度实现自动迁移，不需要改变循环体内的代码。

关于从串行模型迁移到stdpar模型

有一些编译器和工具可以实现自动并行化，检测代码中可以并行化的部分。例如intel编译器加上-qparallel(Windows\*)

(<https://learn.microsoft.com/zh-cn/cpp/parallel/auto-parallelization-and-auto-vectorization?view=msvc-170>[)](https://learn.microsoft.com/zh-cn/cpp/parallel/auto-parallelization-and-auto-vectorization?view=msvc-170)微软在visual) 微软在visual studio中设置/Qpar编译器选项，实现自动并行化和自动向量化。

或许可以利用这些工具将串行模型转换为并行模型（找到可以无序并行化运行的代码）再迁移到stdpar模型上。