OneAPI 创新奖材料

PAC20220824 山东大学 Robuster

PART Ⅰ SYCL / Data Parallel 编程

主要参考资料:

1. https://sycl.readthedocs.io/en/latest/index.html SYCL Reference Documantation

2. 《Data Parallel C++》 Mastering DPC++ for Programming of Heterogeneous Systems using C++ and SYCL. Apress Open

以下是常用 sycl 命令在决赛题目中的使用

- 设备寻访：对于平台设备的寻访 以及 GPU 的指定

```
//queue init
sycl::device de[7];
int dp=0;
auto platforms = sycl::platform::get_platforms();
for(auto &platform : platforms) {
    auto devices = platform.get_devices();
    for (auto &device : devices) {
        de[dp]=device;
        dp++;
    }
}
queue=sycl::queue(de[2], async_handler);
```

- 申请显存：对于显存的多种申请方式

```
duration<double> elapsed01 =end01-start;
std::cout<<"queue init :"<<elapsed01.count()<<" secs\n";
//USM init
A_usm=new kernel_type*[devide];
C_usm=new kernel_type*[devide];
// B_usm = sycl::malloc_shared<kernel_type>(long(n)*k, queue);

//sycl::aligned_alloc_device()
B_usm = sycl::aligned_alloc_device<kernel_type>(sizeof(kernel_type), long(n)*k, queue);
```

- 数据传输：host端 (CPU) 到 device 端 (GPU) 的数据传输

```
for (size_t i = 0; i < devide; i++){
    queue.memcpy(A_usm[i], &origin_dense.host_data()[i*quotient*k], (long)sizeof(kernel_type)*quotient*k);
}
queue.wait();

auto end03=system_clock::now();
duration<double> elapsed03 =end03-end02;
std::cout<<"\n memcpy init :"<<elapsed03.count()<<" secs\n\n";
```

- 使用 dpcpp 进行 编译

```
1798  . ./run.sh
1799  dpcpp mkl_gemm_usm.cpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I$MKLROOT/include -L$MKLROOT/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_cor
e -lsycl -lOpenCL -lpthread -lm -ldl -O3 -o usm.exe
1800  cd ..
1801  dpcpp mkl_gemm_usm.cpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I$MKLROOT/include -L$MKLROOT/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_cor
e -lsycl -lOpenCL -lpthread -lm -ldl -O3 -o usm.exe
1802  . ./run.sh
1803  dpcpp mkl_gemm_usm.cpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I$MKLROOT/include -L$MKLROOT/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_cor
e -lsycl -lOpenCL -lpthread -lm -ldl -O3 -o usm.exe
1804  cd ..
```

- **协同计算**： CPU & GPU 任务协同计算

```
gpu_thread = std::thread(gemm_on_gpu,m,n,k);
if(cpu_m!=0){
    cpu_thread = std::thread(gemm_on_cpu,cpu_m,n,k,&origin_dense.host_data()[gpu_m*k],dense_mat.host_data(),&tem_result[gpu_m*n]);
}
```

PART II OneAPI 工具包 使用方法

2.1 oneAPI DPC++/C++ Compler

编译指令：

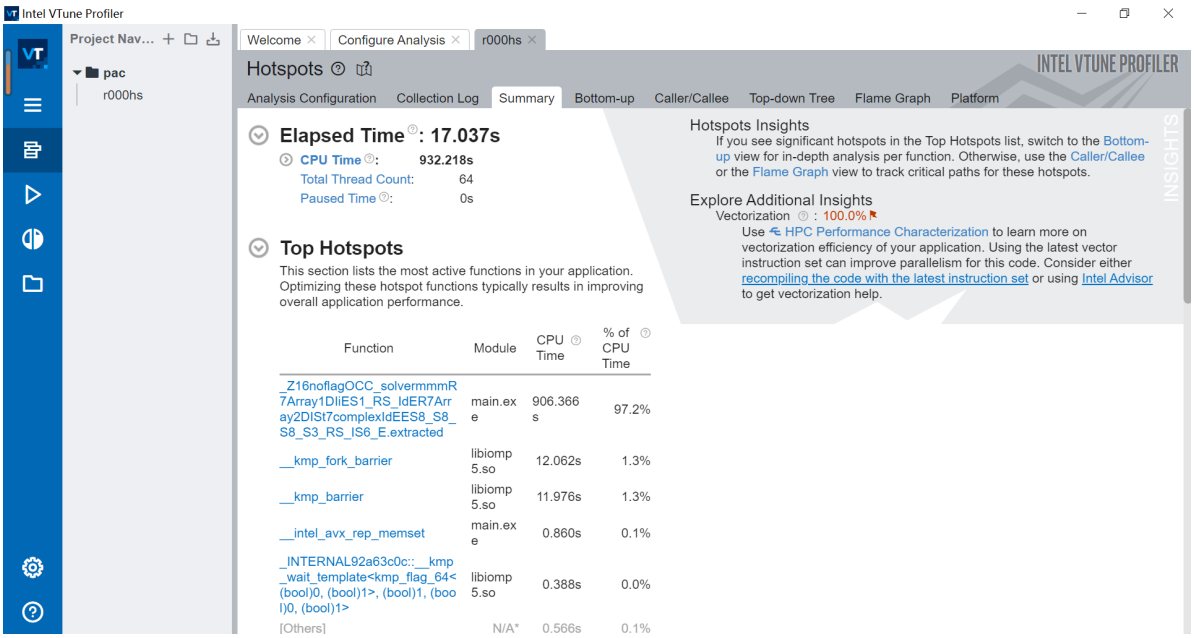
```
1798  . ./run.sh
1799  dpcpp mkl_gemm_usm.cpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I$MKLROOT/include -L$MKLROOT/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_cor
e -lsycl -lOpenCL -lpthread -lm -ldl -O3 -o usm.exe
1800  cd ..
1801  dpcpp mkl_gemm_usm.cpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I$MKLROOT/include -L$MKLROOT/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_cor
e -lsycl -lOpenCL -lpthread -lm -ldl -O3 -o usm.exe
1802  . ./run.sh
1803  dpcpp mkl_gemm_usm.cpp -fsycl-device-code-split=per_kernel -DMKL_ILP64 -I$MKLROOT/include -L$MKLROOT/lib/intel64 -lmkl_sycl -lmkl_intel_ilp64 -lmkl_sequential -lmkl_cor
e -lsycl -lOpenCL -lpthread -lm -ldl -O3 -o usm.exe
1804  cd ..
```

```
编译指令与 GCC/G++ 基本一致
dpcpp [-fsycl-unnamed-lambda] test.cpp [-ltbb|-fopenmp] -o test
dpcpp test.cpp -lmkl-sycl ...
```

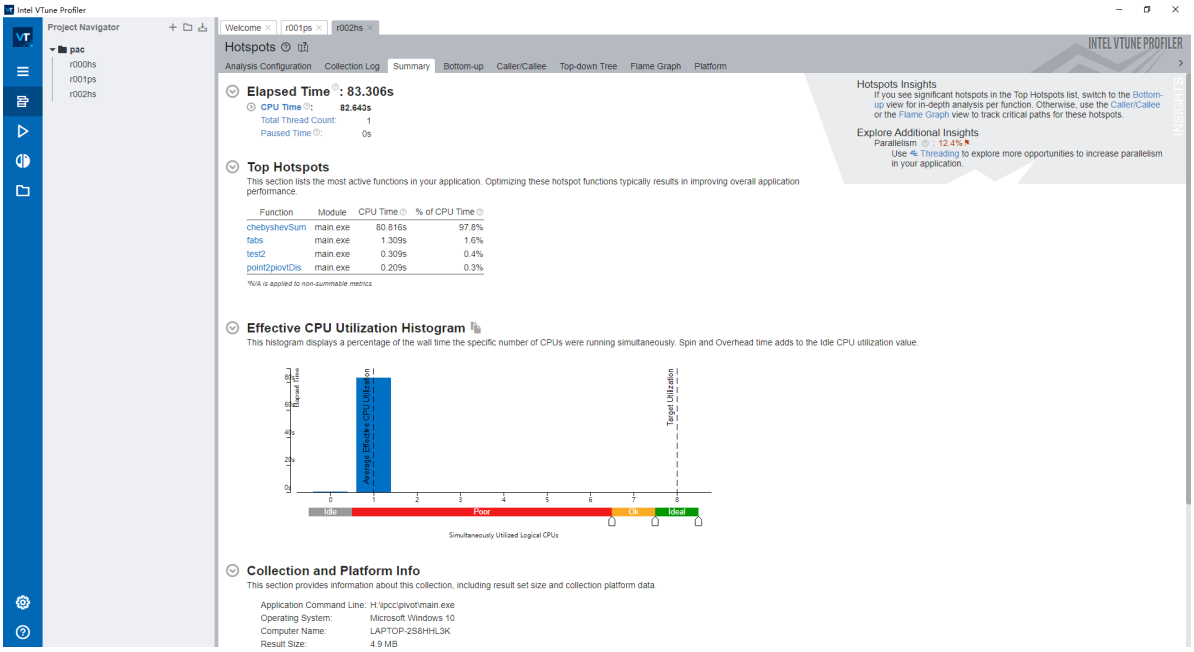
2.2 VTune Profiler

软件使用示意图

左侧是 **程序内部主要函数耗时占比**，右侧有 **指令向量化** 程度

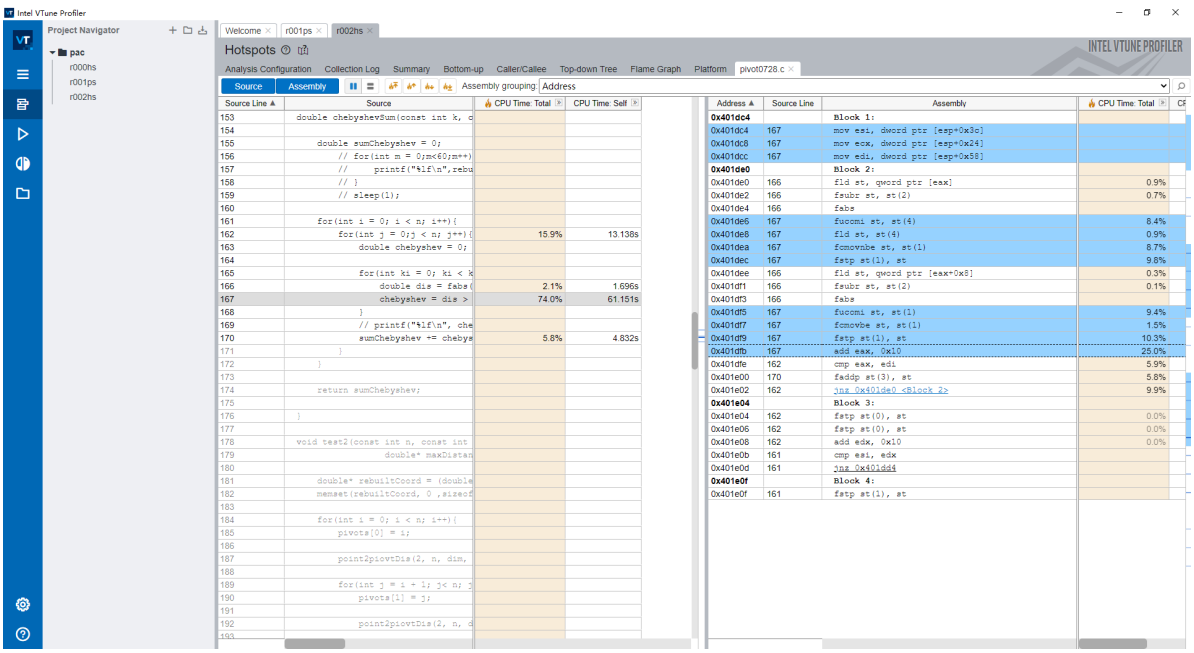


对于代码整体的分析



配合源代码可以查看每一行代码的实际开销

查看 hotspots 每一行代码对应的汇编指令 以及 CPU时间开销占比



2.3 oneAPI Math Kernel Library

在决赛题目进行中，使用 Intel MKL 库中对于矩阵的操作

以下是代码中的基础对于MKL的使用：

- 一次使用 MKL 矩阵操作 的接口调用

```
// kernel_type *B_usm=sycl::malloc_shared<kernel_type>(long(n)*k, queue);
// kernel_type *C_usm=sycl::malloc_shared<kernel_type>(long(m)*n, queue);
// queue.memcpy(A_usm, origin_dense.host_data(), (long)sizeof(kernel_type)*m*k);
queue.memcpy(B_usm,dense_mat.host_data(),(long)sizeof(kernel_type)*n*k);
queue.wait();

float alpha=1.0,beta=0.0;
for (size_t i = 0; i < devide; i++){
    gemm_done = mkl::blas::row_major::gemm(queue, mkl::transpose::nontrans,mkl::transpose::nontrans, quotient,(long) n,(long) k, alpha, A_usm[i]
}
//这里计算余数部分
cblas_sgemm(CBLAS_LAYOUT::CblasRowMajor, CBLAS_TRANSPOSE::CblasNoTrans, CBLAS_TRANSPOSE::CblasNoTrans, m_remainder, n, k, 1.0,&origin_dense.host
queue.wait();
```

- 转置操作：有 outplace 以及 inplace 两种方式，图中使用 inplace 原地转置

```
// result.host_data() [m, n] => tem_result [n, m]
//mkl_somatcopy('R', 'T', m, n, 1, tem_result, n, result.host_data(), m);
mkl_simatcopy('R', 'T', m, n, 1, tem_result, n, m);
```

2.4 oneAPI DPC++ Library

参考资料： https://oneapi-src.github.io/oneDPL/parallel_api/range_based_api.html oneapi 官方库文档

环境需求：Use of the range-based API requires C++17 and the C++ standard libraries that come with GCC 8.1 (or higher) or Clang 7 (or higher)

```
sycl::buffer<kernel_type> buf {n};
auto buf_begin = oneapi::dpl::begin(buf);
auto buf_end   = oneapi::dpl::end(buf);
std::fill(oneapi::dpl::execution::dpcpp_default, buf_begin, buf_end, 0.0);
```