**实验四：PCG Solver实验报告**

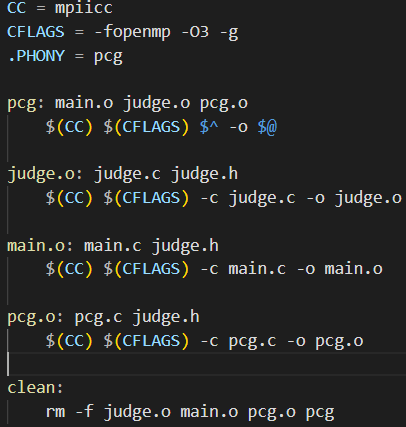
3220105775 王傲哲

1. 优化
2. 编译

source /opt/intel/oneapi/setvars.sh

make

Makefile:



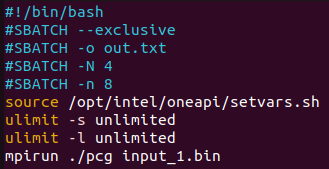
修改CC = mpiicc :采用IntelMPI编译

编译选项添加-g :输出调试信息，便于在vtune-gui中的buttom-up界面，双击函数的源文件以得到源文件中每行代码的热点情况

1. 运行

sbatch job.sh

job.sh:



经测试认为每个节点分配两个进程时性能最佳故设置为-N 4 -n 8

1. 源代码

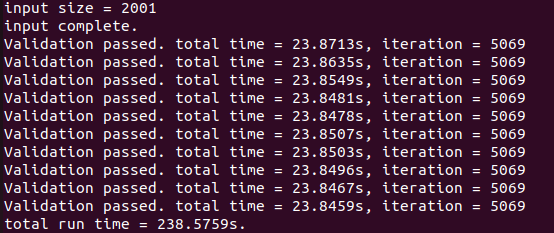
修改<judge.h>:加入#include <mpi.h>

修改<main.c>:完成MPI初始化，在主进程上发起计时代码及input函数

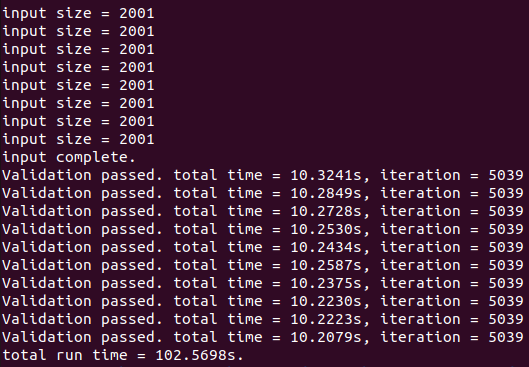
修改<pcg.c>

1. 相对性能

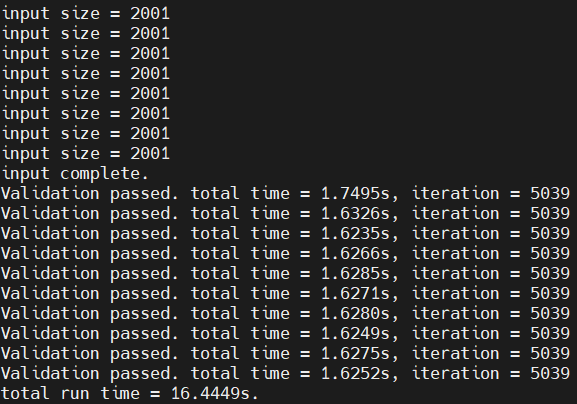
baseline:



第一次优化:

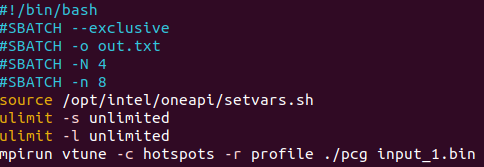


第二次优化：

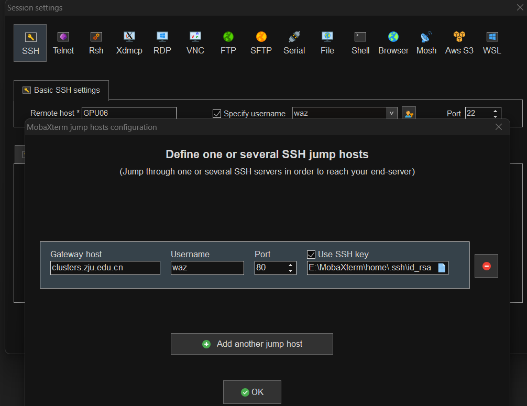
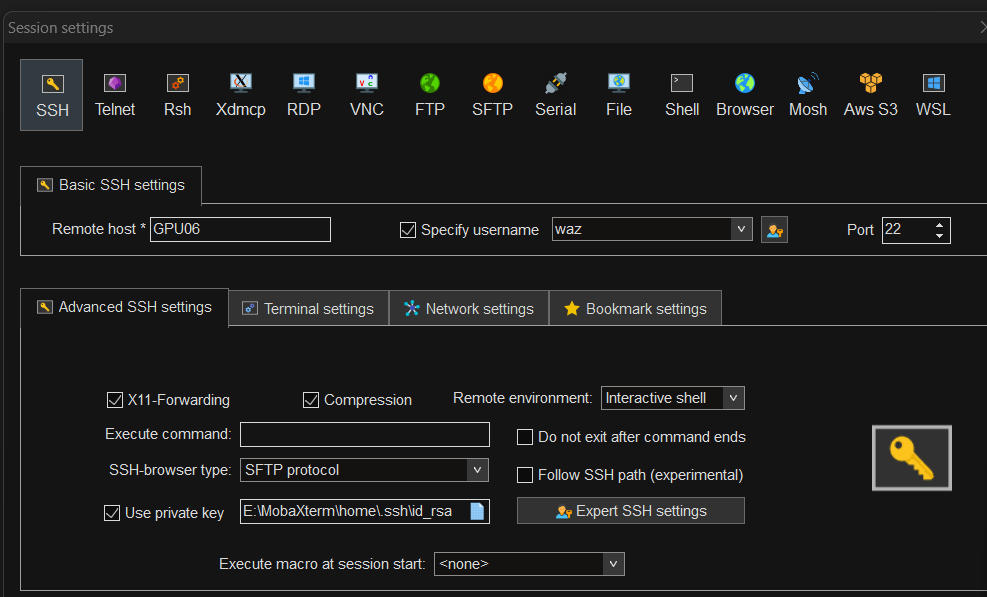


1. Profile

提交如下脚本通过vtune收集性能分析数据

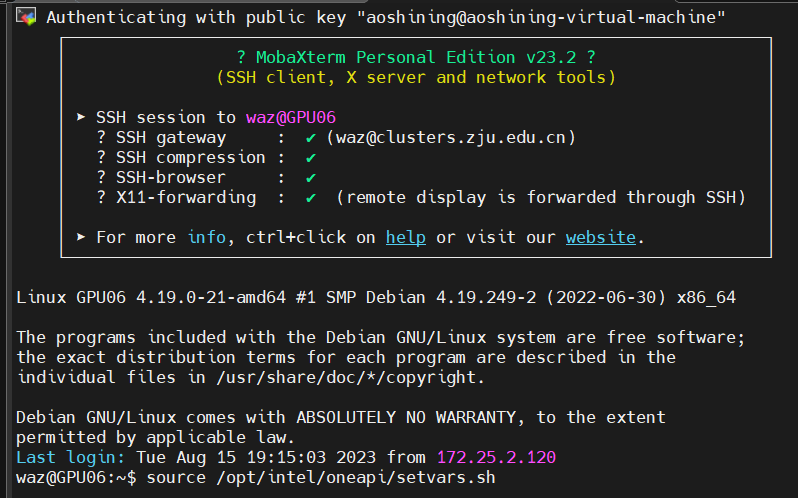


在Mobaxterm上配置从H248跳转到GPU06的ssh，从而实现转发X server



salloc -w GPU06 -p 2080Ti

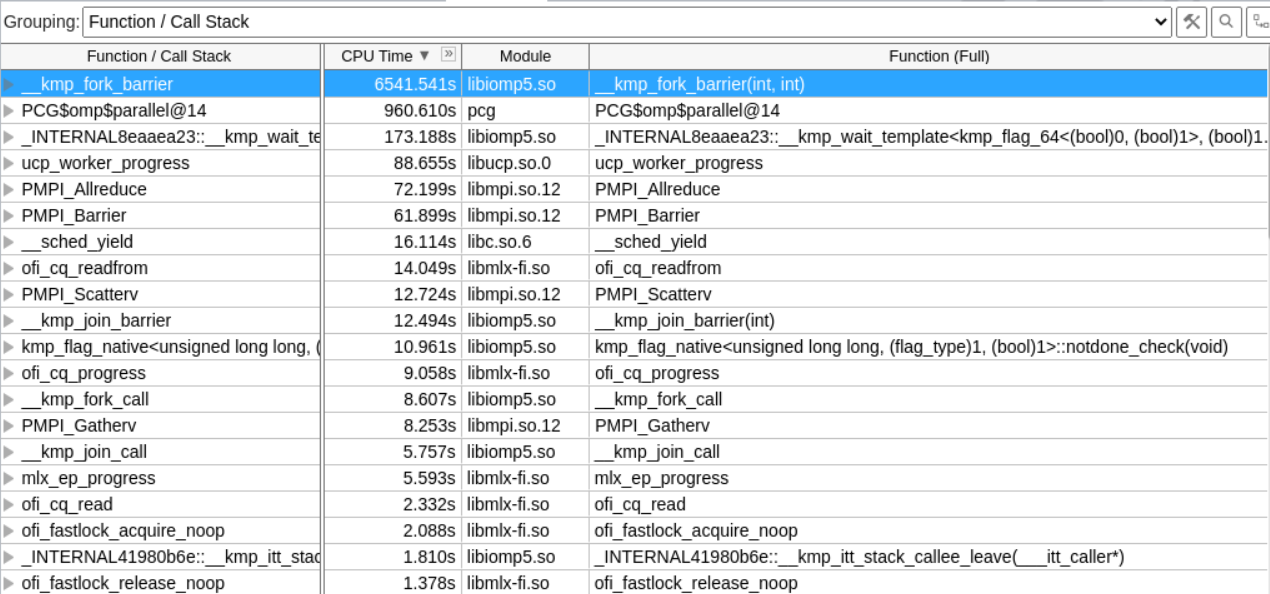
在Mobaxterm上直接访问计算节点



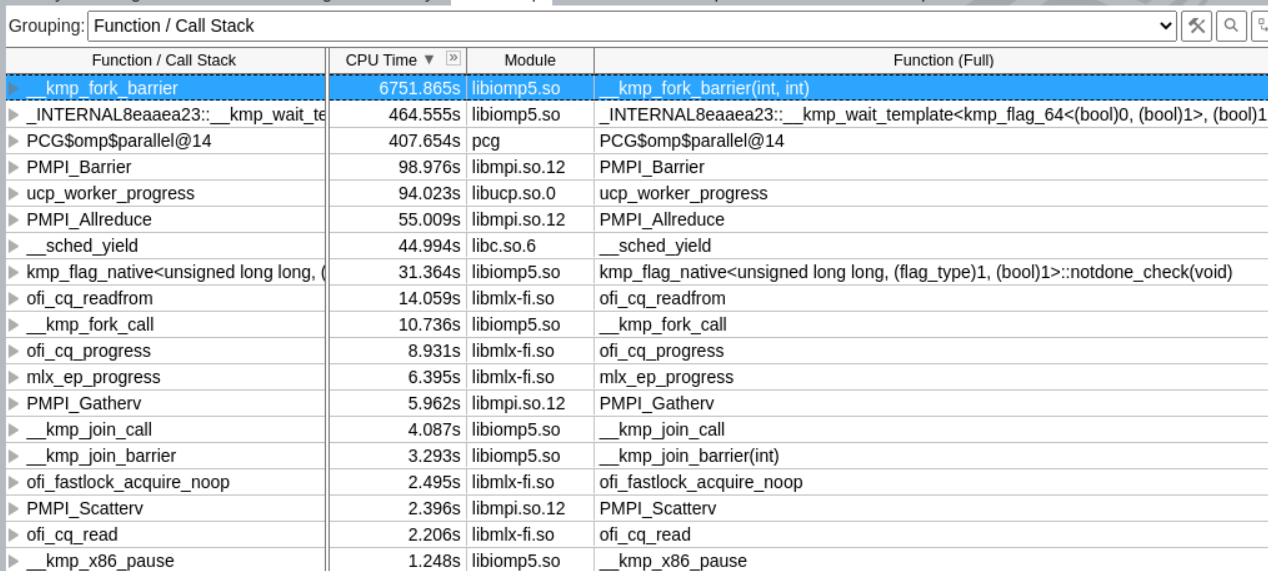
vtune-gui profile.NAS00/GPU01/GPU02/GPU02

1. 最耗时的三个MPI函数:（由大到小）

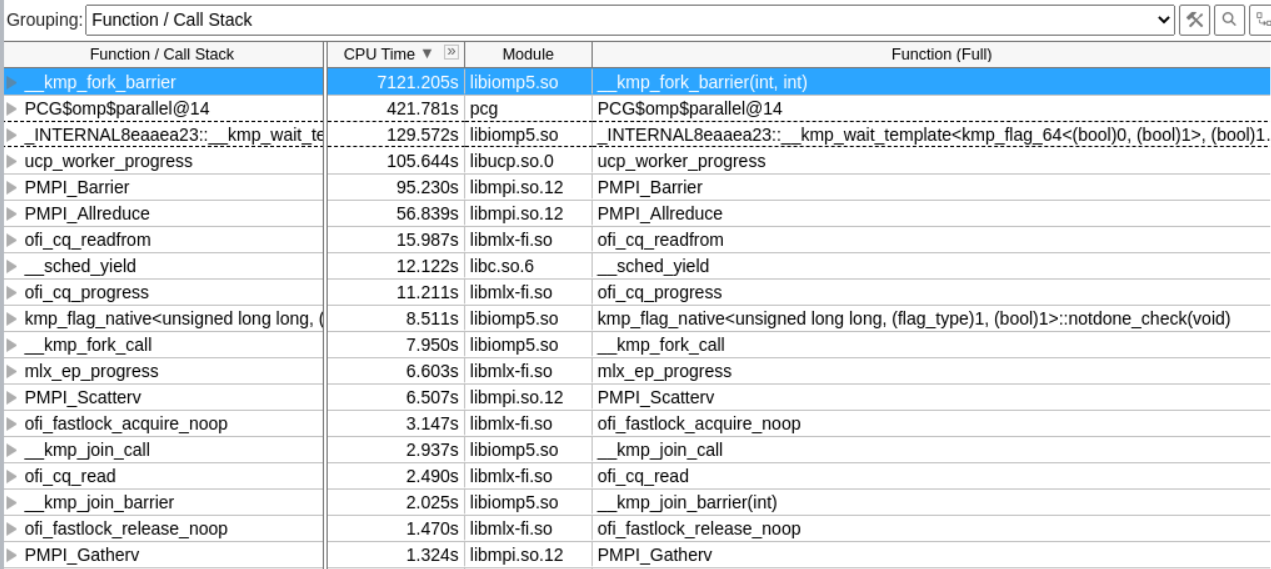
NAS00： MPI\_Allreduce MPI\_Barrier MPI\_Scatterv



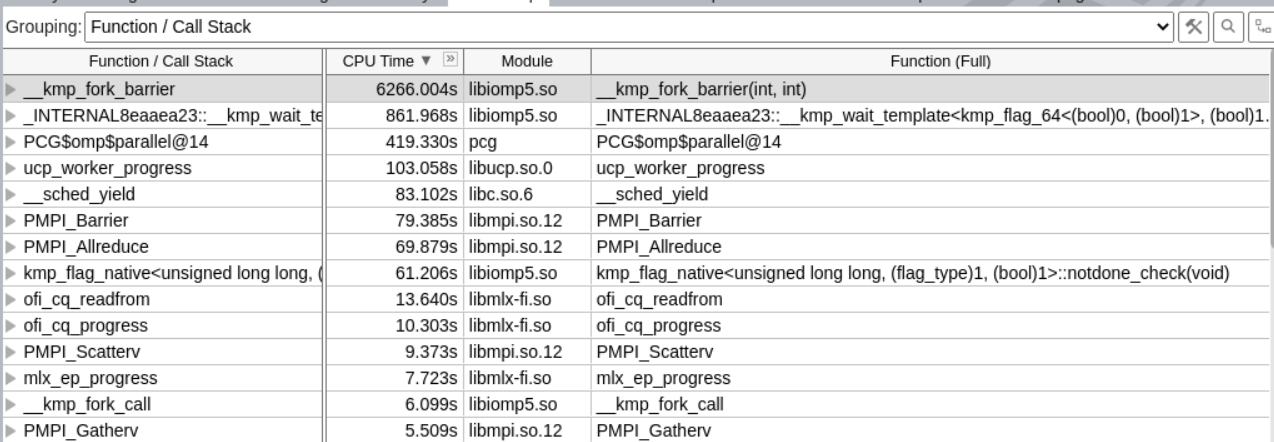
GPU01：MPI\_Barrier MPI\_Allreduce MPI\_Gatherv



GPU02：MPI\_Barrier MPI\_Allreduce MPI\_Scatterv



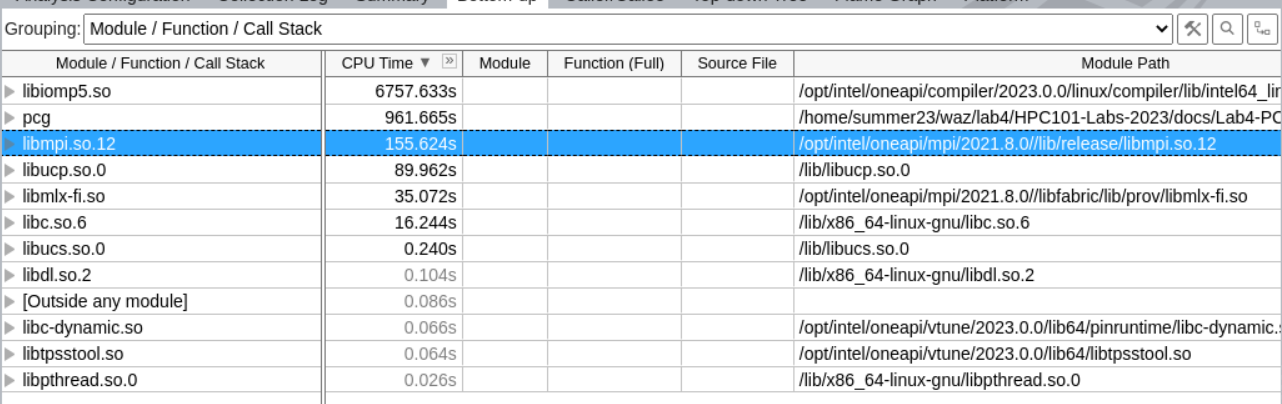
GPU03：MPI\_Barrier MPI\_Allreduce MPI\_Scatterv



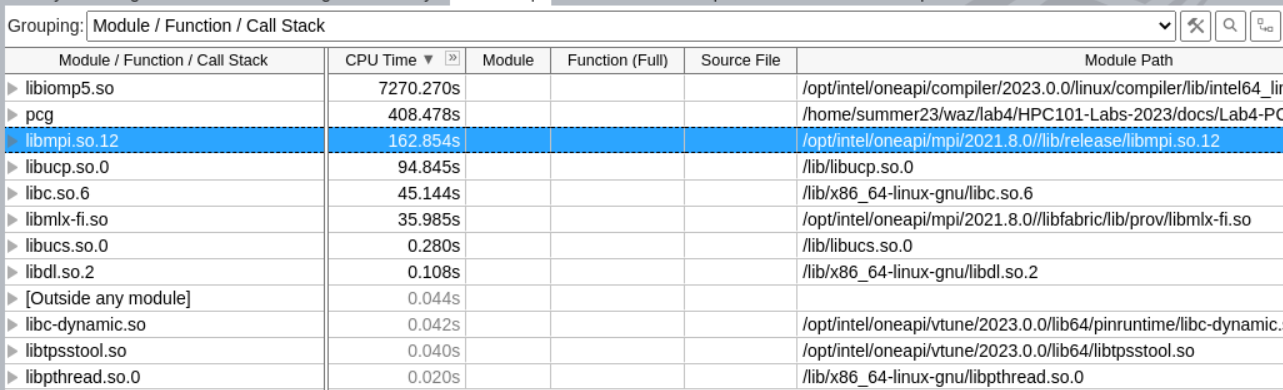
总体来说最耗时的三个MPI函数：MPI\_Barrier MPI\_Allreduce MPI\_Scatterv

1. 程序在MPI上总耗时为643.586s

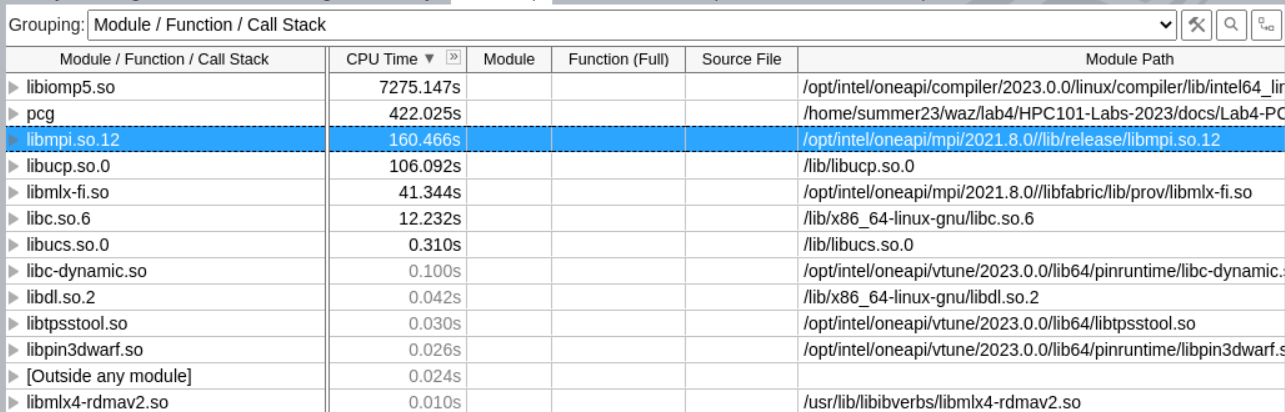
NAS00：



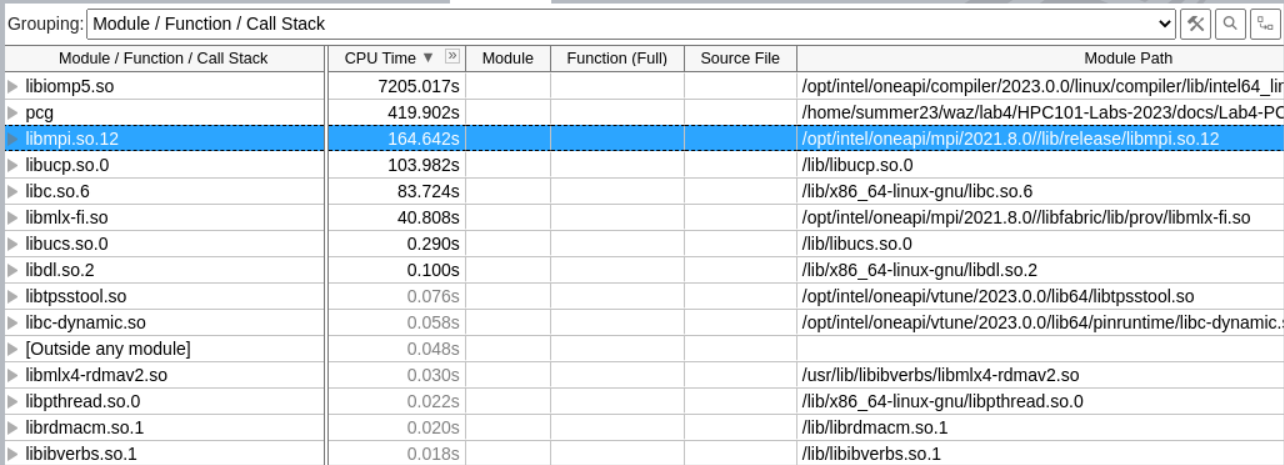
GPU01：



GPU02：



GPU03：



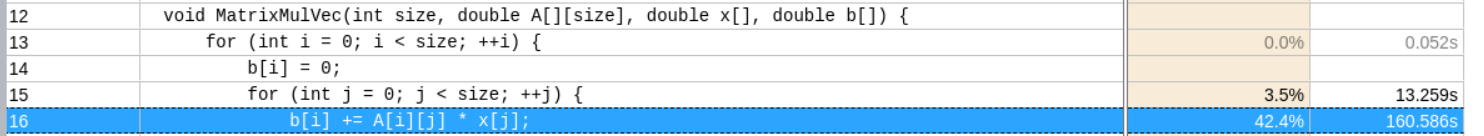
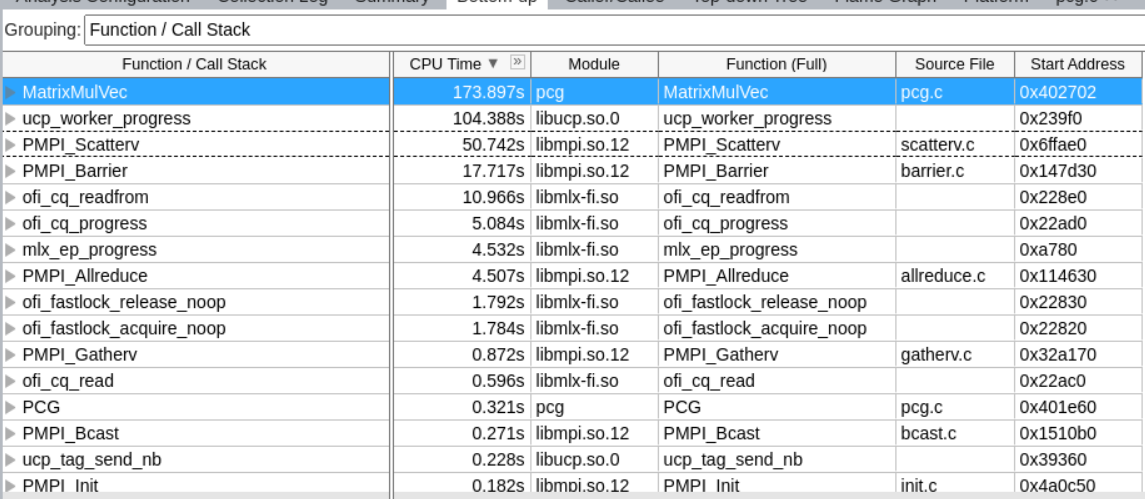
1. 优化过程

第一次优化主要采用MPI改写了pcg函数，大致思路为通过MPI\_Scatterv函数将原有b向量按块划分并按顺序分配到各个进程中，而MPI\_Scatterv的好处就是进程数不必强制满足要被向量长度size整除的条件，从而实现了后续向量初始化以及计算时也是分配到各个进程中并行完成的。但不解的是如果将矩阵A按行类似地分配到各个进程中并行完成矩阵向量乘法的计算会输出答案错误，故决定保留原有MatrixMulVec函数，只是在执行该函数前先将各进程中的p向量汇总，计算出A\_dot\_p后再将其分发到各进程中去。

由第一次profile的结果可知该程序的并行度并不好，主要还是因为在MatrixMulVec函数上耗费了大量时间，且根据线程统计可知线程级并行并未很好地运用，因而也有了第二次优化的方向。

（截图以profile.NAS00为例）





第二次优化采用OpenMP的自动向量化对MatrixMulVec函数进行了优化，虽然b[i] += A[i][j] \* x[j] 耗时929.391s，直观来看比原MatrixMulVec函数此步160.586s要长，合理猜测此时间是所有线程执行此步的总和，优化后统计可知该节点上线程数为50，故每个线程执行此步耗时18.588s，优化效果比较明显。

（截图以profile.NAS00为例）



