# 第一次作业说明

琚锡廷

2021 年 9 月 30 日

## 1 作业背景

矩阵乘法是 HPC 领域最为基础的计算核心之一, 广泛应用于数值计算、机器学习等领域。本次作业的内容是单核矩阵乘法。可以选用的优化包括: 内存对齐 Alignment，数据预取 Prefetch，循环变化 Loop refactorization，分块 Cache blocking，向量化 Vectorization(intrinsic 指令)，循环展开 Loop unrolling。熟悉这些方法的原理，评估其效果，可以让我们更好地掌握程序优化。

## 2 作业描述

### 2.1 作业目标

掌握主要的体系结构优化手段。

### 2.2 作业要求

1. 独立完成代码实现与优化。

2. 提交文件夹命名格式为学号 + 作业编号 + 姓名, 如第一次作业: 2021000000_h1_name，其中包含文件夹 gemm，和报告 report.pdf。

3. 推荐在登陆节点上编译程序，禁止在登录节点上运行程序, 可采用 srun 或 sbatch 来提交任务，希望大家自觉遵守。

4. 注意 DDL，第 5 周周二 24:00 前提交。

5. 不能将 BLAS 数学库的实现作为作业成果。

## 2.3 作业任务

### 2.3.1 单核矩阵乘法

实现矩阵乘法 $C = C + A * B$, 其中, A, B, C 是 $N * N$ 的单精度稠密矩阵

**Input:** $A$: 输入稠密矩阵
$B$: 输入稠密矩阵

**Output:** $C$: $A * B$ 的结果

```
1 receive matrix A B C;
2 for i ← 1 to N do
3     for j ← 1 to N do
4         for k ← 1 to N do
5             C(i,j) ⟵ C(i,j) + A(i,k) * B(k,j);
6         end
7     end
8 end
```

**Algorithm 1:** GEMM

本次作业，需要在集群 (166.111.69.13) 中运行。代码可以从网络学堂上获取，在集群个人帐号的根目录下的/readonly/h1/路径下也有一份原始代码，可以拷贝到个人目录下。解压之后作业中包含 sgemm-naive.c，sgemm-blas.c，sgemm-blocked.c，三个样例，naive 作为最简单的实现，blas 作为本次作业性能的上限，blocked 作为本次作业要改进和优化的基础代码。

编译 benchmark-blocked，可以通过提供的 Makefile 执行：

```
1 make benchmark – test
```

测试 benchmark-blocked 的性能，可以通过执行：

```
1 srun –n 1 ./benchmark – test
```

排他执行可以采用下面的命令来保证节点上只有自己的程序在运行：

```
1 srun –n 1 ––exclusive ––job – name = benchmark –
  test ./benchmark – test
```

也可以使用 sbatch 非阻塞运行，sbatch 需要的脚本示例 test.slurm 在

代码包中已给出。

```
1 sbatch test.slurm
```

## 2.4 作业要求

1. 采用单精度运算，运算结果通过以下的正确性验证 (eps 为机器精度)：

$$square\_gemm(n, A, B, 0) - A * B\| < 3 * n * eps * \|A\| * \|B\|$$

2. 矩阵矩阵乘法的复杂度为 $O(N^3)$，在计算性能指标的时候采用 ($2N^3$ 计算，如果采用了一些非 $O(N^3)$ 而导致通过不了正确性测试，这种情况可以放宽精度的要求，但是需要在作业报告中指出。

3. 开展必要的性能分析，比如某些矩阵规模性能出现明显的降低，可以采用性能分析的工具进行性能分析。

4. 测试 N 取 {127,128,129,255,256,257,383,384,385,511,512,513,639,640,641,767,768,769,895,896,897,1023,1024,1025,1151,1152,1153,1279,1280,1281} 的情况，作为矩阵性能的评价标准，会多次排他测试取平均值作为最后的性能成绩。

## 2.5 作业评分

## 2.6 gemm 100%

1. 评测 gemm 的性能结果，按照提交后的性能排序结果，以及代码质量进行打分 (60%)

2. **详细描述**在实现 gemm 中采取的优化手段，代码对应的部分，以及对应的实验结果，来解释目前取得的性能结果 (30%)。

3. 给出一张完整的实验结果图，描述当前算法的性能，横坐标为矩阵规模，纵坐标为 $Gflop/s$(10%)。

## 2.7 额外的加分 20%

4. 实现其他的 gemm 算法，比如Strassen 算法，实现之后请同时提供不同规模矩阵乘法计算的完成时间与标准算法进行对比，为此需针对性修改 benchmark.c；正确性测试上需要通过 benchmark.c 中提供的检验。

### 2.8 作业提示

1.input sensitivity 问题，算法性能的表现和矩阵规模的大小具有很强的相关性，分析 input sensitivity 并设计更为合理的优化策略将能更有效的提升性能。

2. 可以先利用编译选项来辅助自己完成一部分优化从而减少代码量（能提升 10% 到 20% 的性能），再进行代码级别的优化。

3. 优化中也会存在一些超参数，可以搜索最优参数，来提升性能。

4. 与助教和老师及时交流。

## 3 关于优化的一点建议

建议利用性能工具先进行分析, 简单的加入编译选项可能并不会有明显的性能提升。

向量化是一个非常强有力的工具，可以先从编译选项去考虑。

访存优化是第一次作业的核心。

## 4 参考资料

课程将会提供一些资料帮助大家更加深入的了解高性能计算, BLAS 是一个高性能矩阵运算库，可以参考 GotoBLASGoto & Geijn (2008), 或者 Franchetti 的Chellappa et al. (2007) 中会介绍 gemm(即本次作业) 的一些思考角度，很适合新手阅读，Rothberg 在Lam et al. (1991) 中会详细介绍分块的策略，这里也给大家提醒一下，请关注矩阵应该如何分块才能更加有效的利用 cache，以及 cache 的行为（局部性）。关于向量化推荐大家先在课件 (鲲鹏 920 性能优化介绍) 上看看向量化指令的样例。Slurm 的使用只需掌握本文中的命令即可完成作业，当然如果想了解更多可以参考Slurm 的官方文档。

本次采用的硬件平台是基于 ARM 架构的华为鲲鹏处理器。可以参考ARM 手册，以及一些中文博客。可以参考这些例子来帮助大家快速入门。上述课程文件中也简要介绍了一些性能分析工具，以及 ARM 架构上的优化手段，可以进行参考。

# 参考文献

Chellappa, S., Franchetti, F. & Püschel, M. (2007), How to write fast numerical code: A small introduction, *in* 'International Summer School on Generative and Transformational Techniques in Software Engineering', Springer, pp. 196–259.

Goto, K. & Geijn, R. A. v. d. (2008), 'Anatomy of high-performance matrix multiplication', *ACM Transactions on Mathematical Software (TOMS)* **34**(3), 1–25.

Lam, M. D., Rothberg, E. E. & Wolf, M. E. (1991), 'The cache performance and optimizations of blocked algorithms', *ACM SIGOPS Operating Systems Review* **25**(Special Issue), 63–74.