




# High Performance Linpack

上海交通大学超算队 2023 第一次培训

# 开始之前

回顾一下上周的作业，熟悉一下 slurm




项目链接：<https://github.com/HPC-SJTU/2023-tutorial>

-  **Task1** - 简单的 C 程序编译运行
-  **Task2** - Oh my password
-  **Task3** - 选做，编译运行的真实样例

# 开始之前

回顾一下上周的作业，熟悉一下 slurm

项目链接：<https://github.com/HPC-SJTU/2023-tutorial>

-  **Task1** - 简单的 C 程序编译运行
-  **Task2** - Oh my password
-  **Task3** - 选做，编译运行的真实样例

## Slurm 运行

相关的命令还包括 `srun`、`salloc` 等

```
# 运行 slurm 脚本
sbatch task1.sh
```

```
#!/bin/sh

#SBATCH --job-name=zlzhong-w1t1
#SBATCH --partition=64c512g
#SBATCH -N 1
#SBATCH --exclusive

# 切换目录
cd ~/zlzhong/tutorial/w1t1
# 编译
make
# 运行
./main
```

Task1 参考答案，同时也是 slurm 脚本范本

# Task 1: Hello, slurm

- Makefile 是 C 源程序常见的编译脚本文件
- 用于节省大型 C 项目的编译时间
- 但其实可以干各种各样的事情
- 基础语法比较简单，进阶语法天花乱坠

```
# 声明变量
CC = gcc
OPTS = -Ofast -fopenmp

# 声明一个目标 (target)，名字为 all，依赖于 main
all: main

# 声明一个目标 (target)，名字为 main，没有依赖
main:
    $(CC) $(OPTS) main.c -o main

# 声明 clean 目标每次都执行
.PHONY: clean
# 声明一个目标 (target)，名字为 clean，没有依赖
clean:
    @rm -f main
```

Task1 中的 Makefile 简化版

# Task 2: Oh my password

- 跑一个单进程的 Python 脚本破译密码
- 需要使用 conda 管理 python 环境
- 单进程大约 18 分钟，多进程大约 18 秒

## 怎么把核利用起来？

- 本质上是一个从 `\0` 到  $256^{KBC}$  的游戏
- KBC 即 KEY\_BYTE\_CNT，指密码的位数
- 简单的想法：任务划分

```
from multiprocessing import Pool, cpu_count

def process(nums: List[int]):
    pass

with Pool(cpu_count()) as p:
    p.map(process, tasks)
```

Task2 多进程优化参考

```
# 激活 conda 环境
source ~/scripts/conda.sh
# 创建新的 conda 环境，指定 Python 版本
conda create -n zlzhong-w1t2 python=3.11
# 安装程序依赖
pip install -r requirements.txt
```

Task2 中的 conda 环境配置

```
#!/bin/sh

#SBATCH --job-name=zlzhong-w1t2
#SBATCH --partition=64c512g
#SBATCH -N 1
#SBATCH -exclusive

cd ~/zlzhong/tutorial/w1t2
# 激活 conda 环境
source ~/scripts/conda.sh
conda activate zlzhong
python exploit.py
```

Task2 参考脚本

# High Performance Linpack

What, Why and How

# What is HPL?

High Performance Linpack

- 一个用于评估超级计算机性能浮点数运算性能的基准测试 (benchmark)
- 用于 TOP500 排名
- 理论峰值性能计算  $R_{peak} = \text{\#N Cores} \times \text{AVX Freq} \times \text{DP\_ops\_per\_cycle}$
- 思源一号：
  - Intel® Xeon® Platinum 8358 Processor
  - 每节点有两个 CPU
  - 每个核有两个 AVX-512

$$\begin{array}{ccccccc} 2 & \times & 32 & \times & 2.9 & \times & 32 & = & 5.9 \\ \text{CPU} & & \text{Cores} & & \text{GHz} & & \text{DP ops} & & \text{TFLOPS} \end{array}$$

# Run HPL

参考：<https://docs.hpc.sjtu.edu.cn/app/benchtools/hpl.html>

- 将思源一号上已经编译好的 HPL 跑起来

```
# 切换到你的目录
cd your_name

# 创建一个新文件夹
mkdir ~/HPL && cd ~/HPL

# 使用 module load 加载相关模块
module load oneapi/2021.4.0

# 将 HPL 拷贝到当前目录
cp -r $MKLR00T/benchmarks/mp_linpac ./

# 进入 HPL 所在目录
cd mp_linpac/
```

```
# 直接跑
./runme_intel64_dynamic
```

## # 样例输出

```
gpu10      : Column=019712 Fraction=0.655 Kernel=322257
gpu10      : Column=020480 Fraction=0.675 Kernel=539202
gpu10      : Column=020992 Fraction=0.695 Kernel=691088
gpu10      : Column=024064 Fraction=0.795 Kernel=416654
gpu10      : Column=026880 Fraction=0.895 Kernel=333272
gpu10      : Column=029952 Fraction=0.995 Kernel=185925
```

```
=====
T/V          N    NB    P    Q          Time
-----
WC00C2R2    30000  256    1    2          36.77
```



# Compiling HPL

用 Intel OneAPI SDK 编译能够省去编译 BLAS 的时间

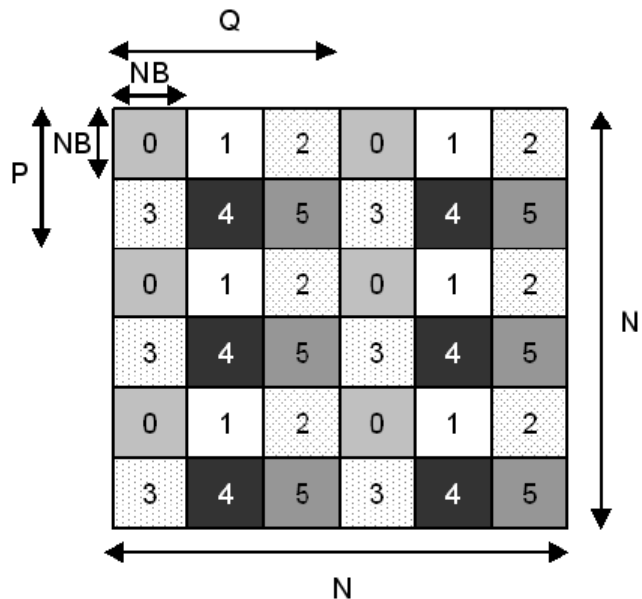
- HPL 2.3 Download Link: <https://www.netlib.org/benchmark/hpl/hpl-2.3.tar.gz>

```
# 切换到自己的目录
cd your_name
# 下载 HPL 2.3
wget https://www.netlib.org/benchmark/hpl/hpl-2.3.tar.gz
# 解压
tar xzvf ./hpl-2.3.tar.gz
# 切换到 hpl 2.3
cd hpl-2.3

# 复制 Makefile
cp setup/Make.Linux_Intel64 ./
# 修改 Make.Linux_Intel64
# - TOPdir 改为当前目录 (可以使用 pwd 查看当前目录的绝对路径)
# 偷懒的话可以用以下命令
# sed -i -e "s#^[[[:space:]]*TOPdir[[[:space:]]]*= [[[:space:]]]*.*#TOPdir = $(pwd)#" Make.Linux_Intel64
make arch=Linux_Intel64
```

# Tuning HPL

- 为了压榨出机器的最高性能，我们需要为机器“量身定做”一份测试用例
- 主要考虑 `Ns`、`P`、`Q`、`NB`、`N` 五个参数
- 可以参考 [https://www.advancedclustering.com/act\\_kb/tune-hpl-dat-file/](https://www.advancedclustering.com/act_kb/tune-hpl-dat-file/)
- <https://ulhpc-tutorials.readthedocs.io/en/latest/parallel/mpi/HPL/>



Thanks