

Lab1: 简单集群搭建

1. 概况

实验环境: `macOS 26.5.1, Debian GNU Linux`

使用到的Ask/Agent: `GitHub Copilot: ChatGPT-5.x & deepseek v4-pro`

通过此次环境配置和小集群的搭建, 我意识到了AI的日益强大, 也意识到自己亟需掌握更多Linux环境的知识和经验。文档的信息量不小, 我需要慢慢消化。

另外, 这个文档只记录了最基础的部分, 将在期末考试后有所更新。

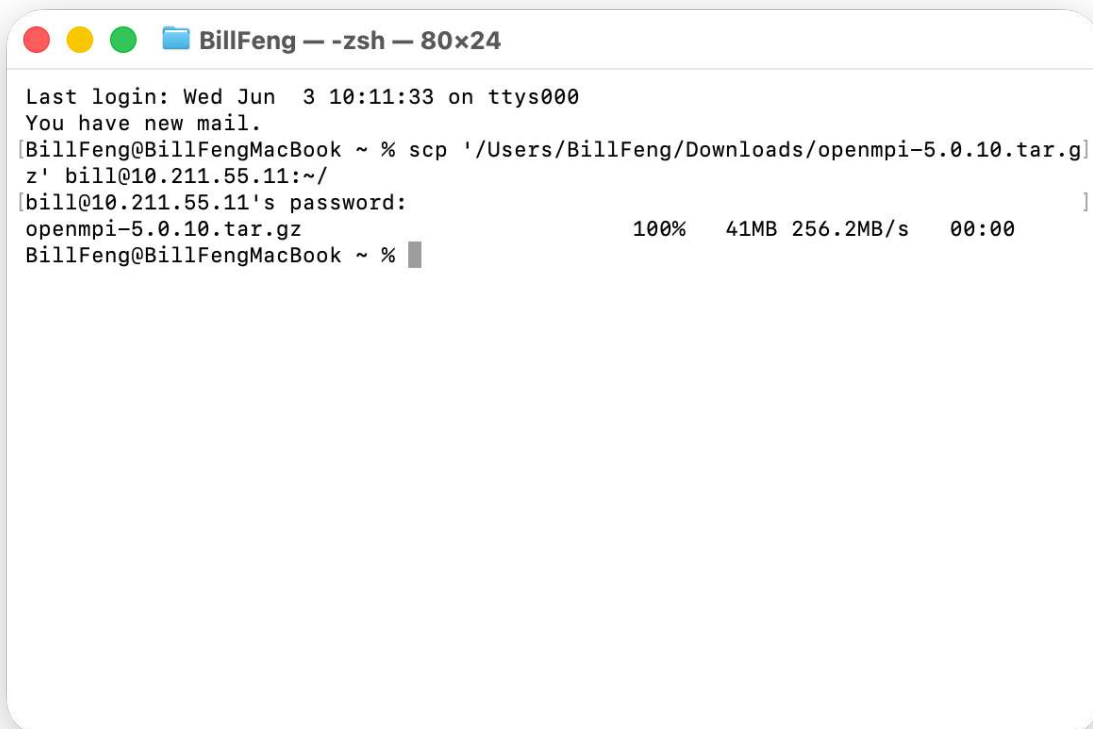
应该有挺多感想和挫折的, 但是现阶段时间紧张, 留到考试之后写。

所以文档中目前主要是记录配置过程。

2. 任务一: 从源码构建 OpenMPI、BLAS 和 HPL

2.1 OpenMPI的构建

- 下载并 `scp` 压缩包



```
BillFeng — -zsh — 80x24

Last login: Wed Jun  3 10:11:33 on ttys000
You have new mail.
[BillFeng@BillFengMacBook ~ % scp '/Users/BillFeng/Downloads/openmpi-5.0.10.tar.gz'
z' bill@10.211.55.11:~/
[bill@10.211.55.11's password:
openmpi-5.0.10.tar.gz                                100%  41MB 256.2MB/s   00:00
BillFeng@BillFengMacBook ~ %
```

Image 1

- 解压并安装

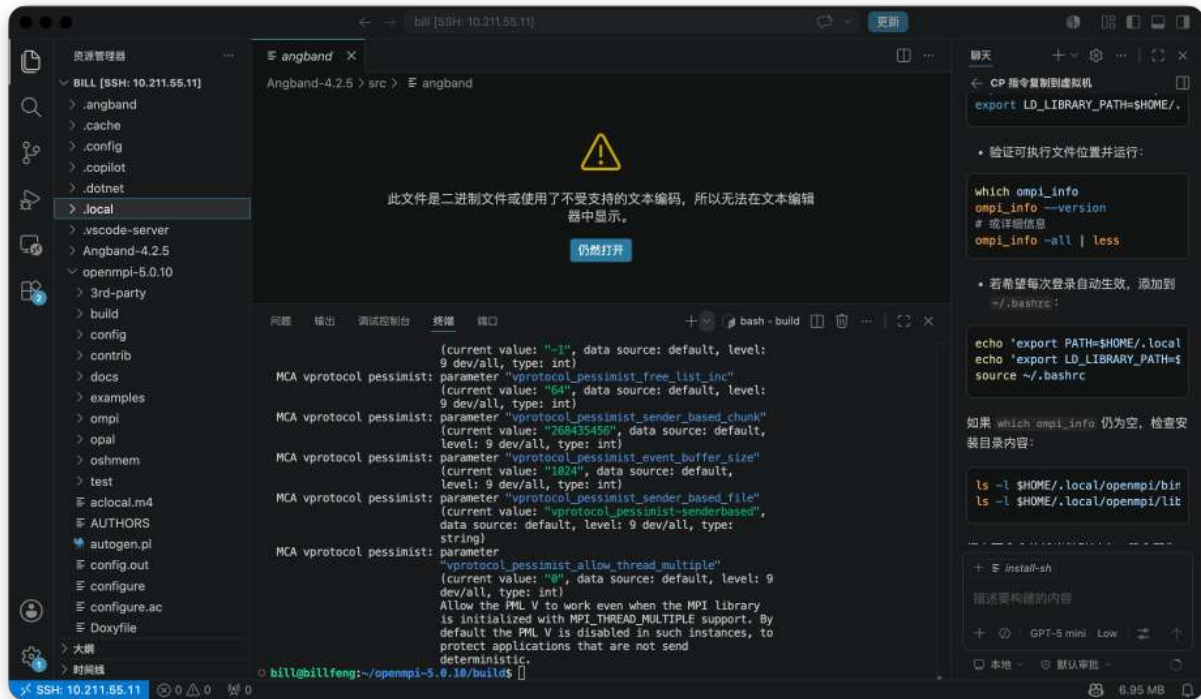


Image 2

2.2 BLAS&CBLAS

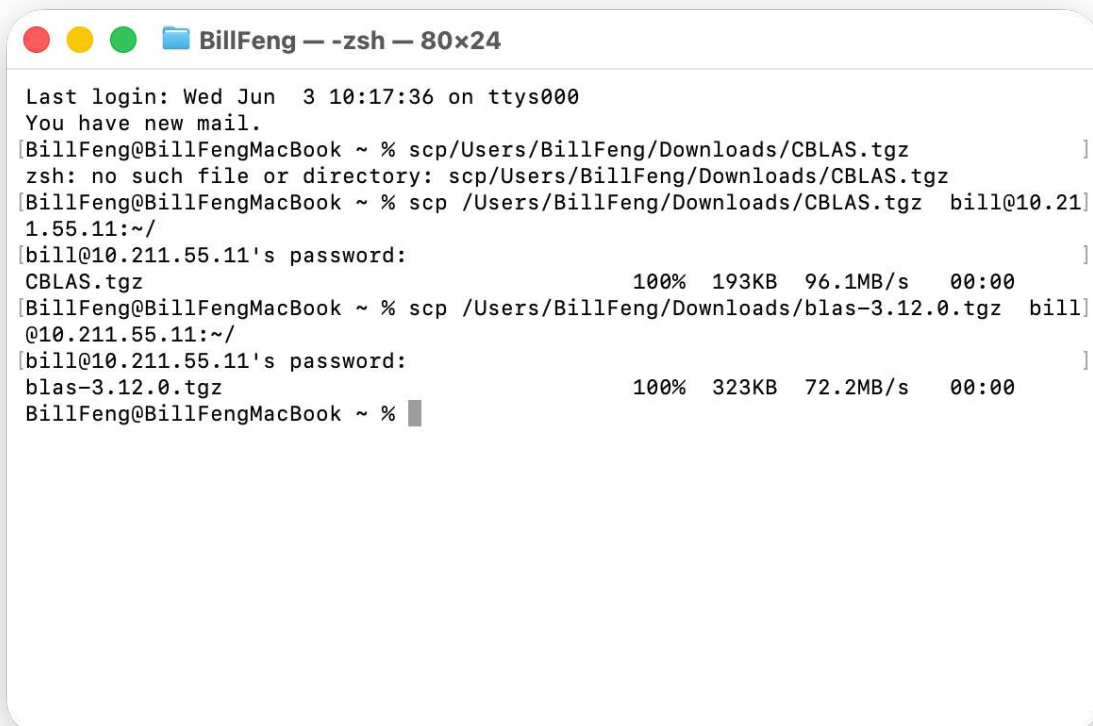


Image 3

- 修改编译器配置:

```

• bill@billfeng:~$ cd BLAS-3.12.0
❌ bill@billfeng:~/BLAS-3.12.0$ make
gfortran -O2 -frecursive -c -o isamax.o isamax.f
make: gfortran: No such file or directory
make: *** [: isamax.o] Error 127

```

Image 4

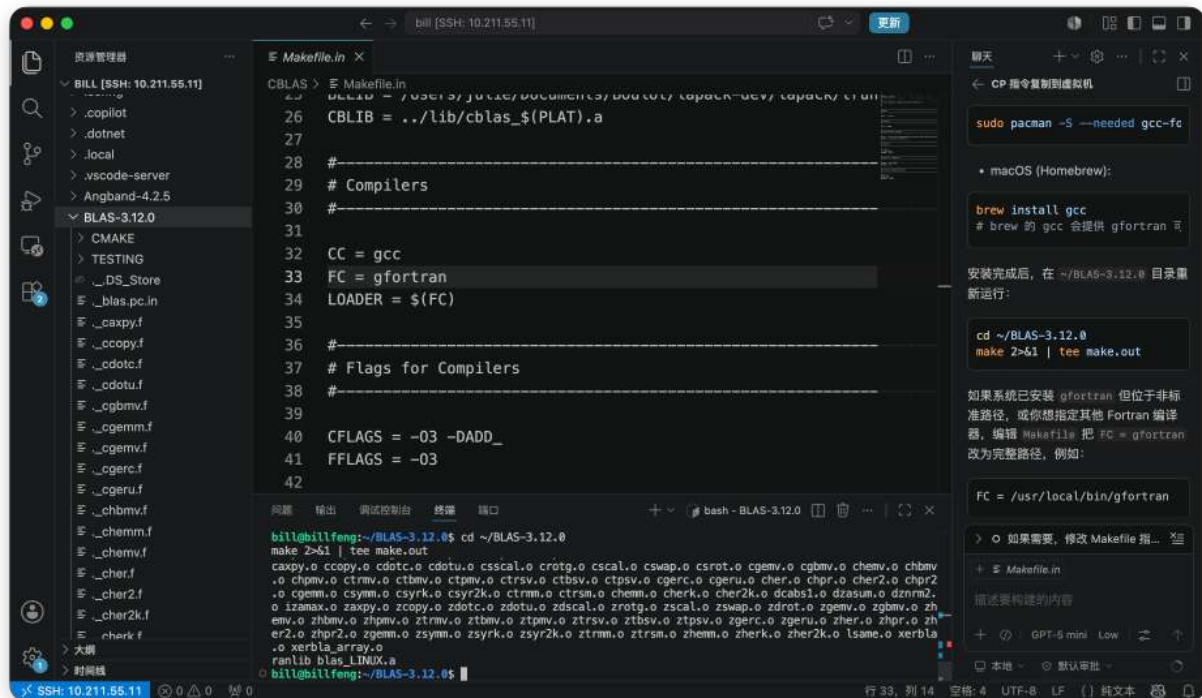


Image 5

- 遇到了文档中的问题，修改 `TOPdir` 路径：

```

bill@billfeng:~$ wget "https://netlib.org/benchmark/hpl/hpl-2.3.tar.gz"
tar xvf hpl-2.3.tar.gz
cd hpl-2.3
cp setup/Make.Linux_PII_FBLAS . ...
make[1]: Entering directory '/home/bill/hpl-2.3'
( cd src/auxil/Linux_PII_FBLAS; make )
make[2]: Entering directory '/home/bill/hpl-2.3/src/auxil/Linux_PII_FBLAS'
Makefile:47: Make.inc: No such file or directory
make[2]: *** No rule to make target 'Make.inc'. Stop.
make[2]: Leaving directory '/home/bill/hpl-2.3/src/auxil/Linux_PII_FBLAS'
make[1]: *** [Make.top:54: build_src] Error 2

```

Image 6

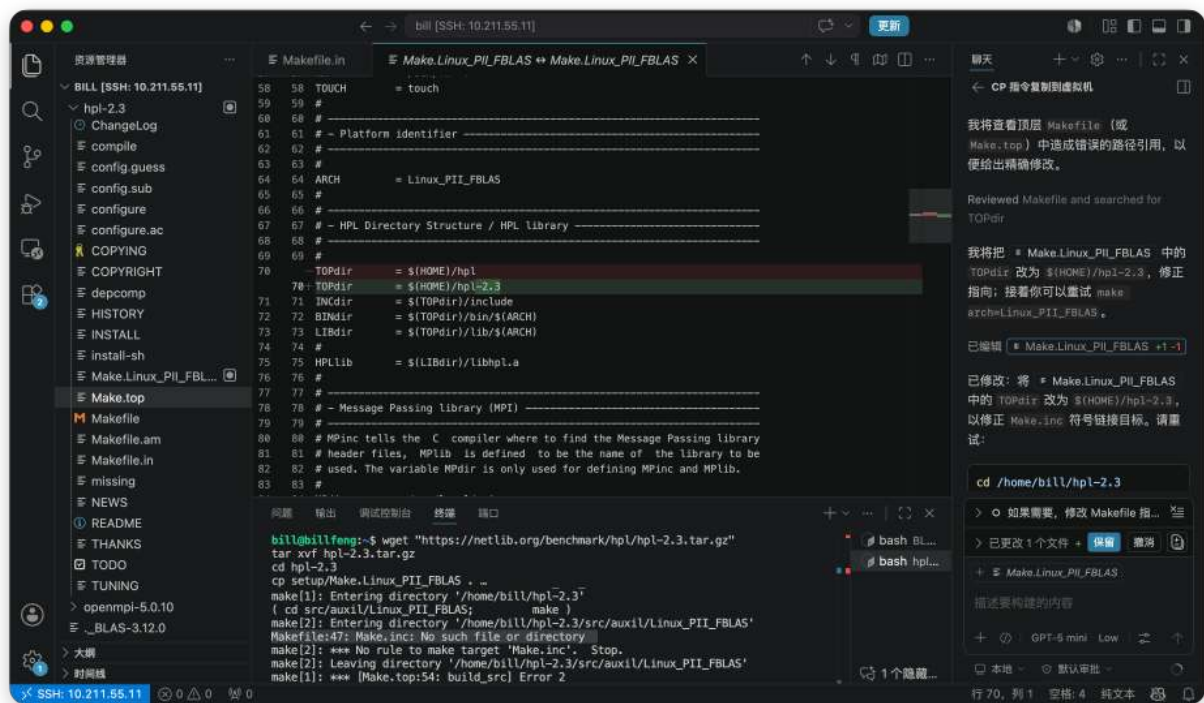


Image 7

2.3 HPL

下面是在 `Make.Linux_PII_FBLAS` 中修改或设定的参数：

- `TOPdir` : `$(HOME)/hpl-2.3`
- `MPdir` : ``/home/bill/.local/openmpi`
- `MPinc` : ``-I$(MPdir)/include`
- `MPLib` : `$(MPdir)/lib/libmpi.so`
- `LAdir` : `$(HOME)/BLAS-3.12.0`
- `LAlib` : `$(LAdir)/blas_LINUX.a $(LAdir)/cblas_LINUX.a`
- `LINKER` : `/usr/bin/gfortran`

成功生成可执行文件在 `bin/Linux_PII_FBLAS` 目录下：

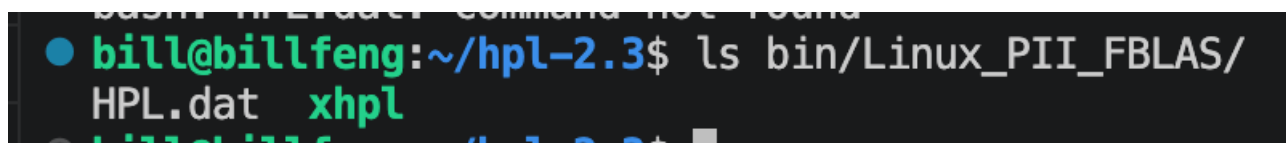


Image 8

3. 任务二：配置 NFS 共享目录

- 这是四个节点映射的IP地址：

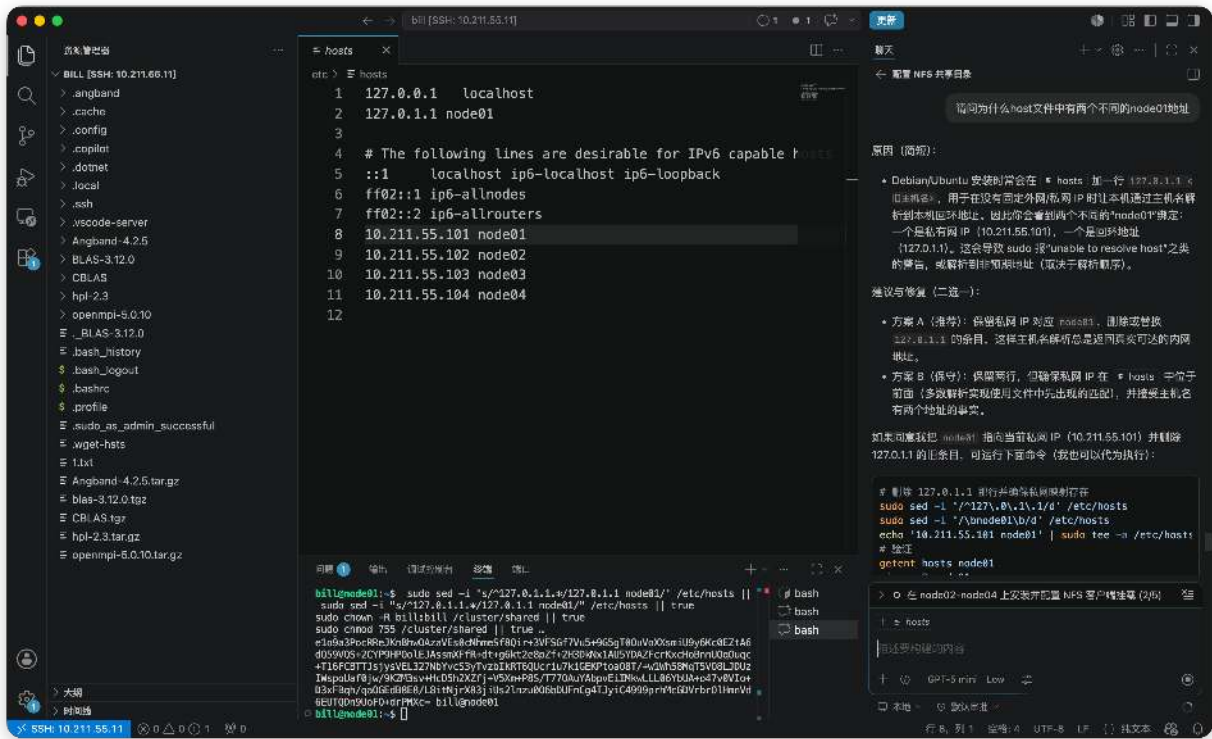


Image 9

- 我的思路是先配置 node02，然后将配置好的节点拷贝两份。下面三张图是我在 node02 中操作的记录。

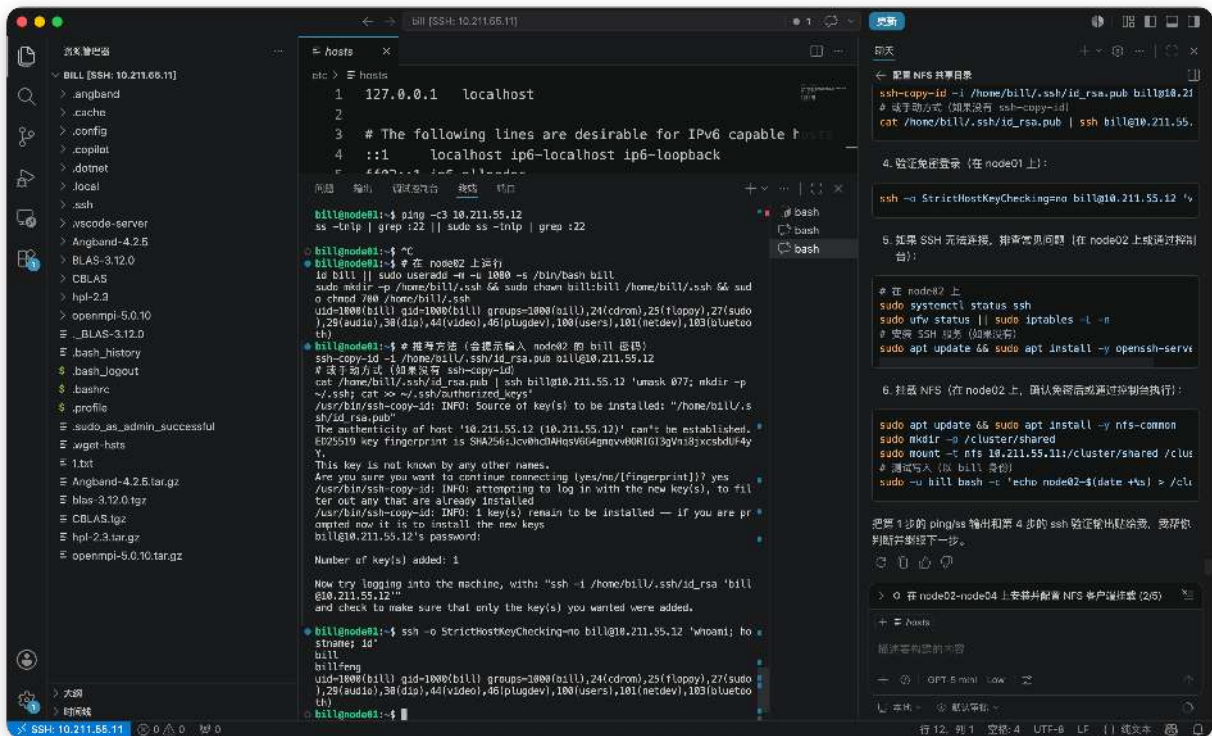


Image 10

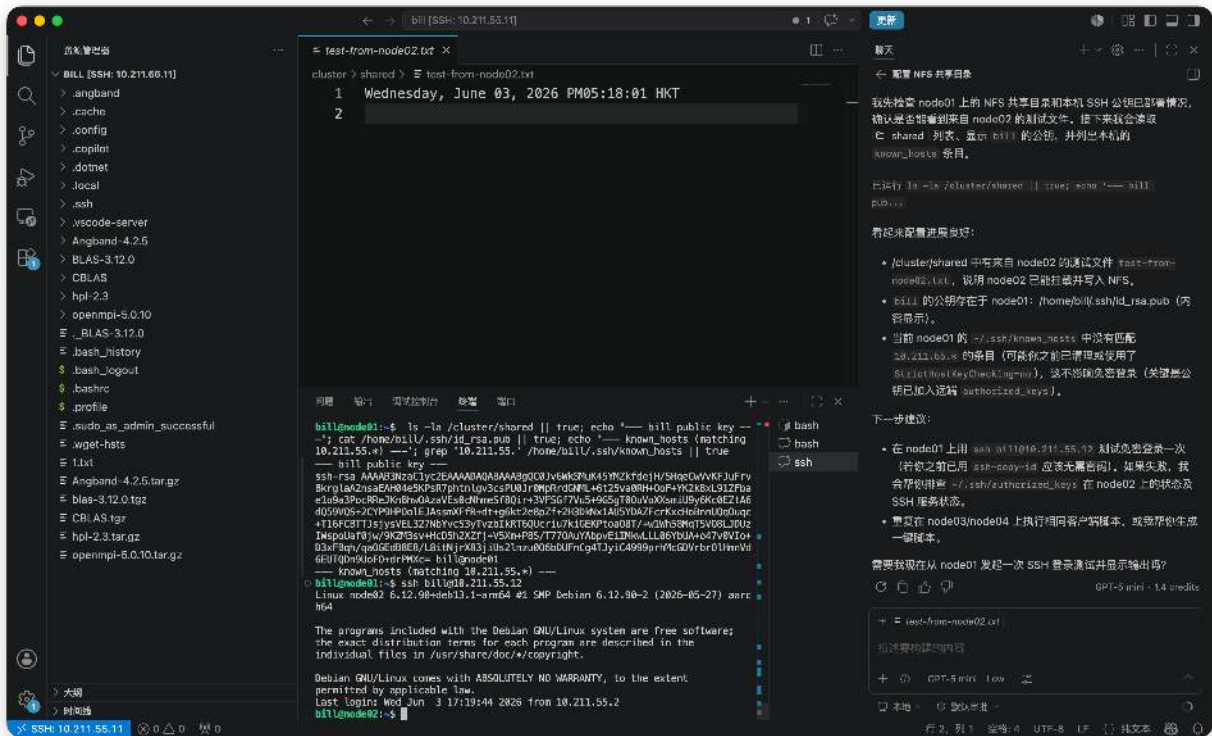


Image 11

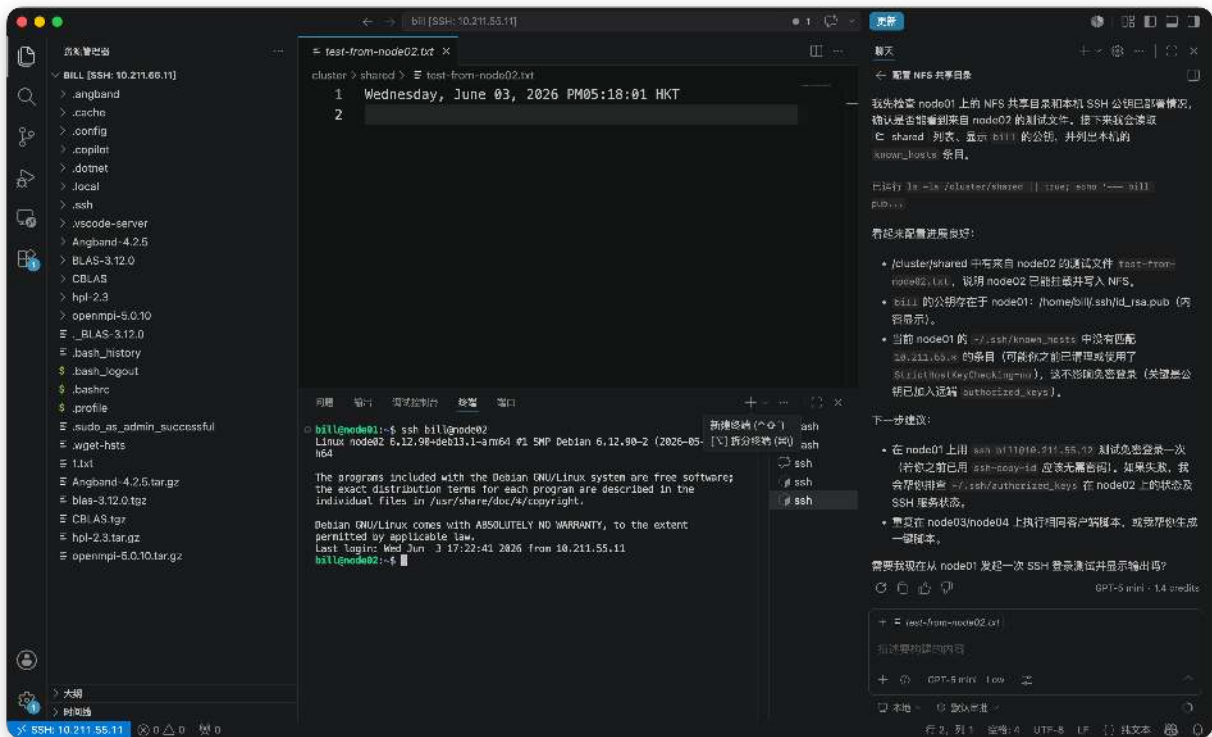


Image 12

- 将密钥生成算法修改为更推荐的 `ed25519`，并在此处清理了多余的密钥。
- 在此处我还保持着对AI的掌控，尽量不用 `Agent` 而是用 `Ask` 模式，亲自输入或者粘贴命令，了解每一步都在做什么。到任务三点时候就有点“破戒”了。



- Image 14

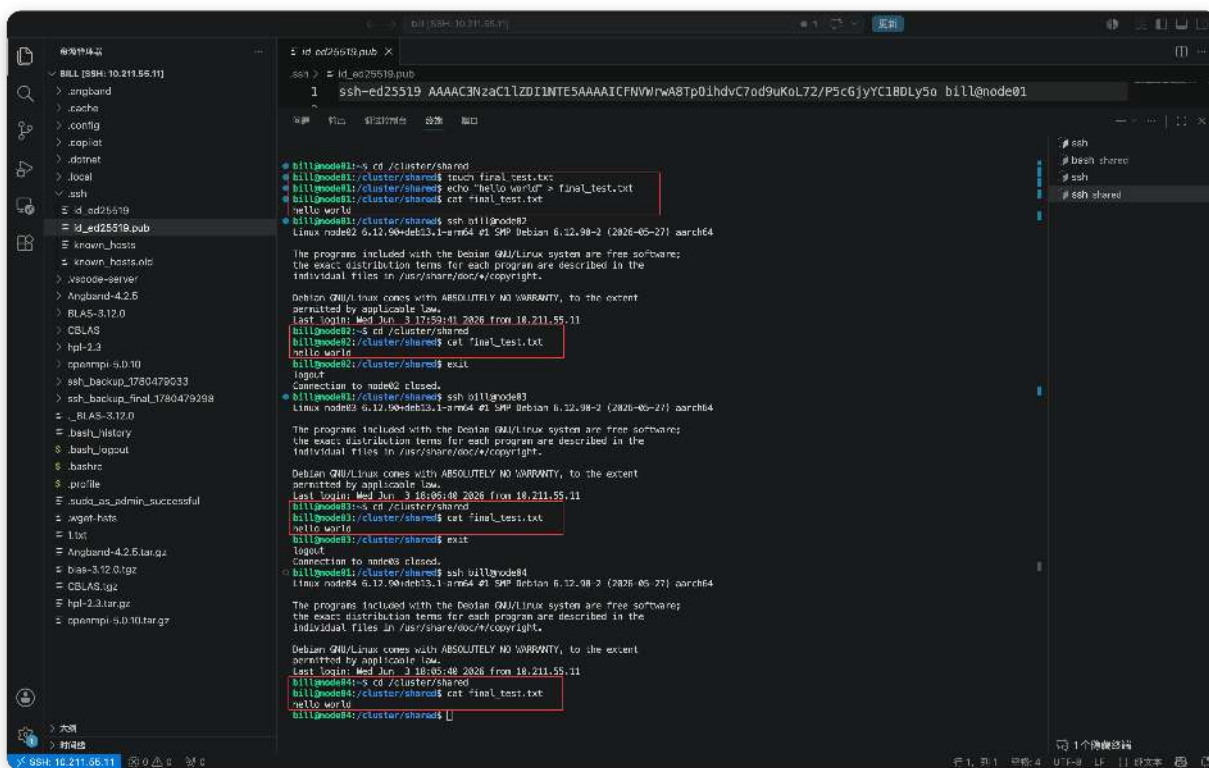


Image 17

- 读写文件时只和 `node01` 通信, `node02` 不通过 `node03` 读文件。

```

1 | node02 ┐
2 | node03 ┤ node01 (NFS 服务端)
3 | node04 ┘
4 | //有点累, 吃饭去了, 明天继续。

```

4. 任务三：配置 Slurm 作业调度系统

- 集群搭建后 `srund` 单节点正常, 但多节点作业报错。
- 我开始意识到自己的能力比较局限, 借用了 `Agent` 来排查问题, 发现是因为 `UID` 不一致导致 Slurm 跨节点通信失败



5. 任务四：使用 Slurm 提交 HPL

- 关于 `HPL.dat` :

把我们刚才关于dat的关键对话的精华总结一下

Reviewed key points about HPL.dat

HPL.dat 核心总结

文件中每行的格式：数量 + 具体的值

例：

1	# of NBs	← 1 个 NB 值
128	NBs	← 值 = 128, 分块大小

三个关键参数：

参数	含义	你的值	规则
N	矩阵是 N×N	10000	越大计算越多，受内存限制 (N²×8 字节)
NB	分块粒度	128	影响缓存效率，通常 32~256
P, Q	进程网格的行列	2, 2	P × Q = 进程数，尽量接近正方形

P×Q 与 Slurm 的关系（最容易搞错）：

Slurm: `--ntasks=4` → 4 个 MPI 进程
HPL.dat: `P×Q=2×2=4` → 必须相等！

不等会怎样：
`P×Q < ntasks` → 浪费资源
`P×Q > ntasks` → HPL 报错

HPL 会遍历所有参数组合：

`1 个 N × 1 个 NB × 1 组 P×Q` = 跑 1 次
`3 个 N × 2 个 NB × 2 组 P×Q` = 跑 12 次

DeepSeek V4 Pro · 0x

Image 20

- 根据文档推荐，去掉 `node01` 的计算节点属性，此时 `node01` 是登陆和控制节点，`node02-04` 是计算节点。

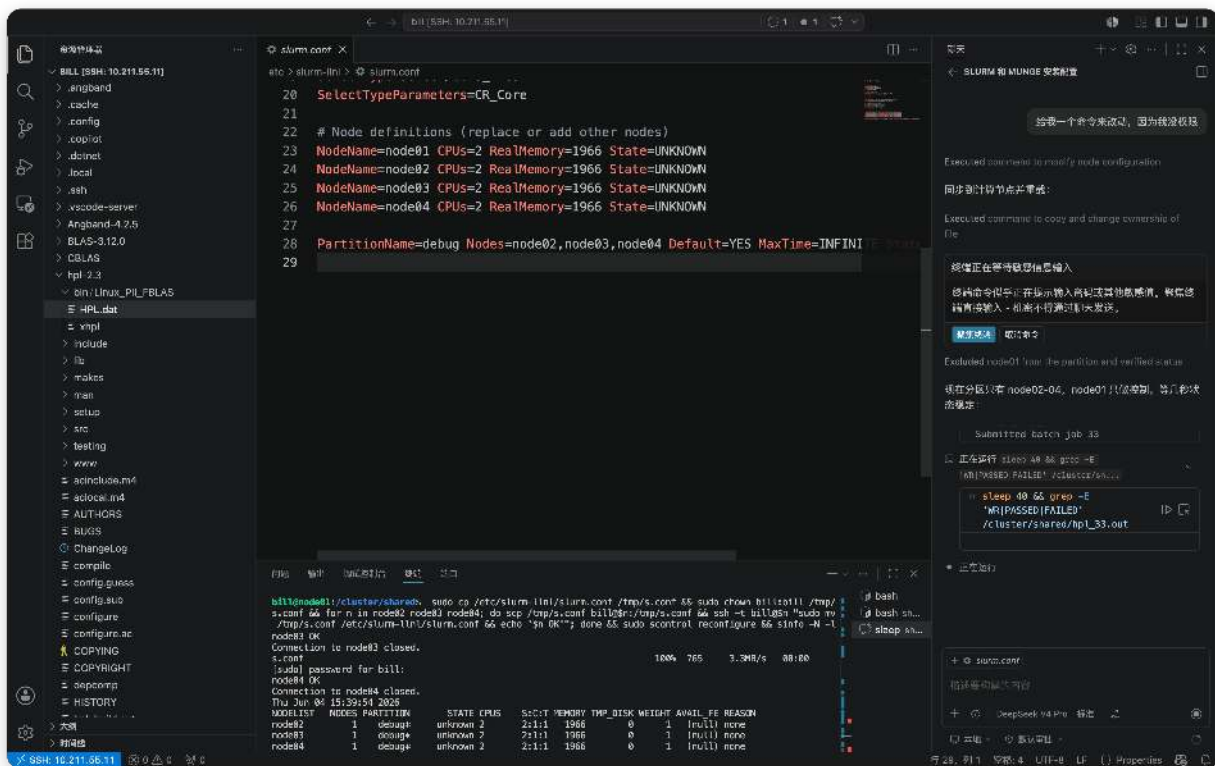


Image 21

- 用于提交HPL的脚本 `run_hpl.sbatch`

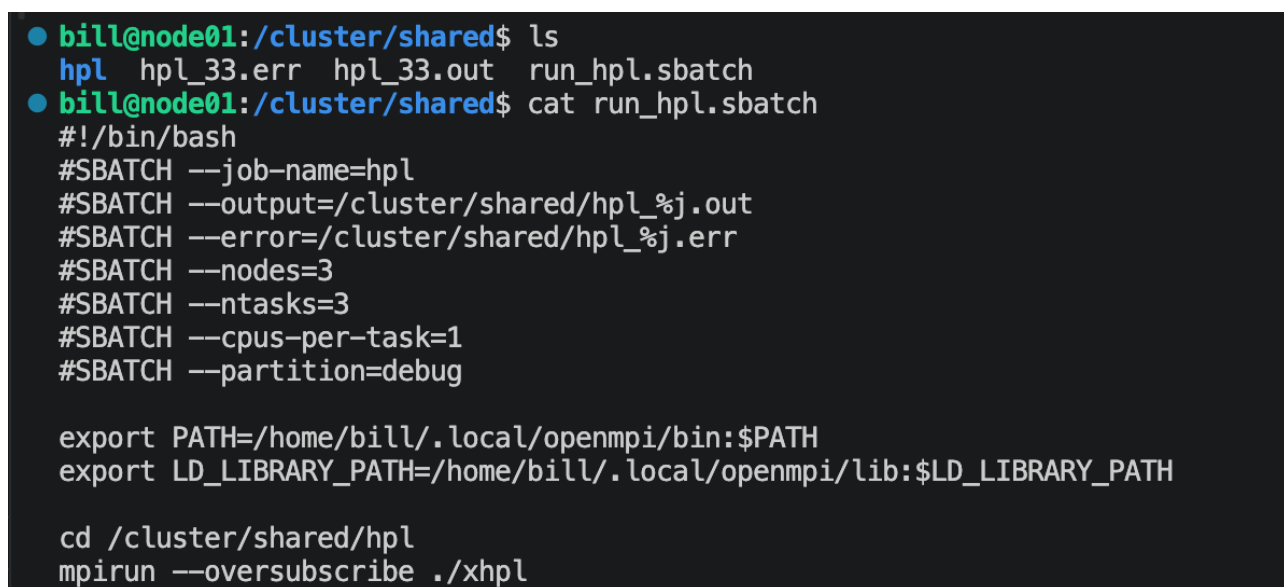


Image 22

- 见图片批注。


```
bill@node01:/cluster/shared$ cat HPL.dat
cat: HPL.dat: No such file or directory
bill@node01:/cluster/shared$ cat hpl/HPL.dat
cat: hpl/HPL.dat: No such file or directory
bill@node01:/cluster/shared$ cat hpl/HPL.dat
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (0=stdout,7=stderr,file)
1            # of problems sizes (N)
10000       N
1            # of NBs
128         NB
0           PMAP process mapping (0=Row-,1=Column-major)
1           # of process grids (P x Q)
1           Ps
3           Qs
16.0        threshold
1           # of panel fact
0           PFACTs (0=left, 1=Crout, 2=Right)
1           # of recursive stopping criterium
2           NBMINs (>= 1)
1           # of panels in recursion
2           NDIVs
```

矩阵规模

分块粒度

node01作为控制节点，node02-04作为计算节点。三个计算节点只能分配1*3的条形进程网络

Image 23

- 随后通过任务管理器查看CPU资源情况，发现资源在 node02 上堆满，而其他节点几乎没有任务。

0.49
0.76为什么主机显示node02吃满了而其他的没有

Analyzed HPL workload distribution across nodes

因为 `mpirun` 没有跟 Slurm 通信，它把所有 3 个进程全丢在 node02 上了，node03/node04 在空转。修一下 `sbatch` 脚本：

已运行 `cat > /cluster/shared/run_hpl.sbatch << 'EOF'`

```
#!/b...
```

Updated script to specify nodes for OpenMPI usage

原因解释：

之前：`mpirun --oversubscribe ./xhpl`

- OpenMPI 不知道 Slurm 分了哪些节点
- 三个进程全挤在 node02, node03/node04 闲着

现在：`mpirun --host $SLURM_JOB_NODELIST ./xhpl`

- Slurm 自动填成 `mpirun --host node02,node03,node04 ./xhpl`
- 三个进程真正分发到三台机器

提交：

```
sbatch run_hpl.sbatch && queue
```

🔄 📄 👍 💬

DeepSeek V4 Pro • 0x

Image 24

- 来自AI的解释: `$SLURM_JOB_NODELIST` 返回压缩格式 `node[02-04]`, OpenMPI不认得。展开它。
- 修改后性能有所提升, 见下图:
 - 上面框内部分是 `node02` 单独计算
 - 下面框内部分是 `node02-04` 协同计算

```

- The relative machine precision (eps) is taken to be 1.110223e-16
- Computational tests pass if scaled residuals are less than 16.0

=====
T/V          N    NB    P    Q          Time          Gflops
=====
WR00L2L2     10000  128    1    3          50.97          1.3083e+01
HPL_pdgesv() start time Thu Jun  4 15:49:16 2026
HPL_pdgesv() end time   Thu Jun  4 15:50:07 2026

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 4.51323610e-03 ..... PASSED
=====

- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
  ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be 1.110223e-16
- Computational tests pass if scaled residuals are less than 16.0

=====
T/V          N    NB    P    Q          Time          Gflops
=====
WR00L2L2     10000  128    1    3          39.83          1.6741e+01
HPL_pdgesv() start time Thu Jun  4 15:53:00 2026

```

Image 25

- 任务管理器的资源分配说明进程调度是有效的!

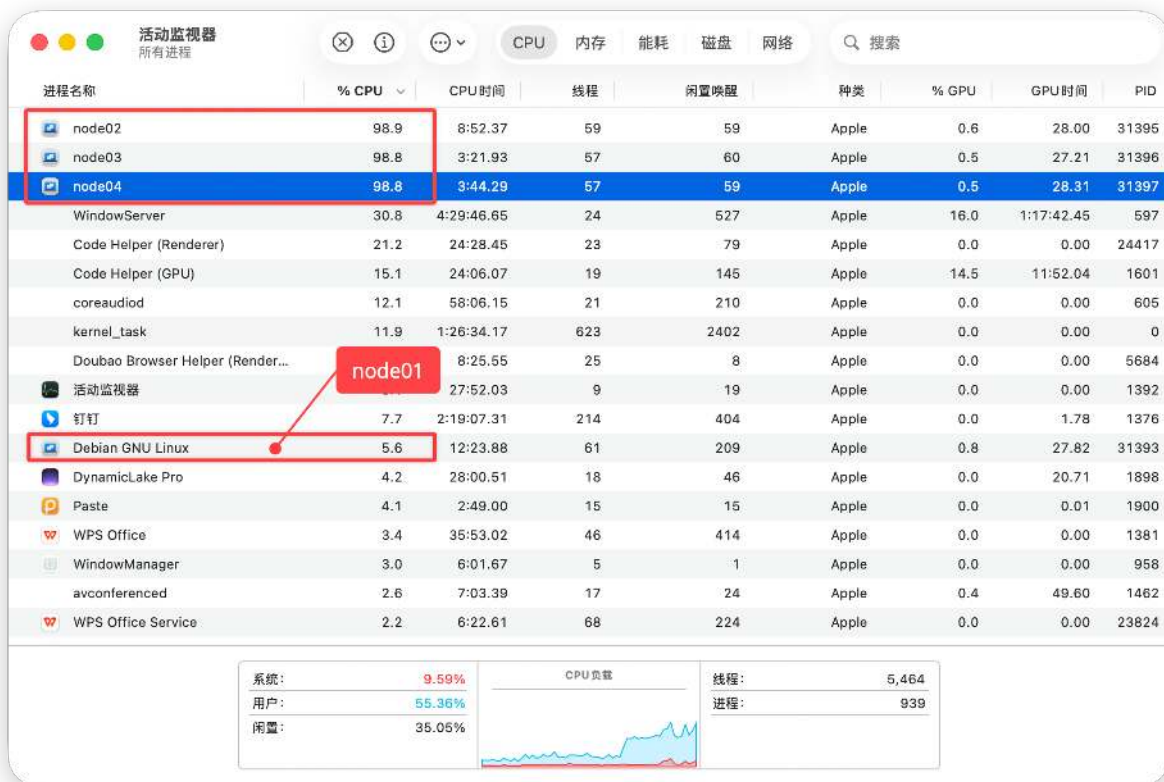


Image 26

对于调优等任务将在以后更新。