集群运行 Numpy 程序的测试实践与注意事项

你正确安装 Numpy了吗?

本测试主要使用的是某 Numpy为底层的计算包，任务的核心在于求解矩阵对角化，使用的是 LAPACK 。计算的大头在于 500次尺寸约为 60000×60000的稀疏矩阵对角化计算。

从不同渠道安装的 Numpy，其所使用的 LAPACK 优化可能并不一样，在不同的硬件上性能表现也大不相同。常见的优化有 MKL、OpenBLAS和 ATLAS，其中通常说来 Intel的 CPU上跑 MKL会更快一些 。

一般来说，不考虑指定渠道，默认情况下

```
conda install numpy
```

从 Anaconda下载安装会根据你的 CPU自动选择，若是 Intel则用 MKL，若是 AMD则用 OpenBLAS 。当然，你可以切换到 `conda-forge` 之类的去指定版本。

如果使用 `pip` 安装，

```
pip install numpy
```

[illegible] OpenBLAS[illegible]

MKL[illegible]

[illegible] MKL[illegible] Numpy[illegible] `MKL_NUM_THREADS`

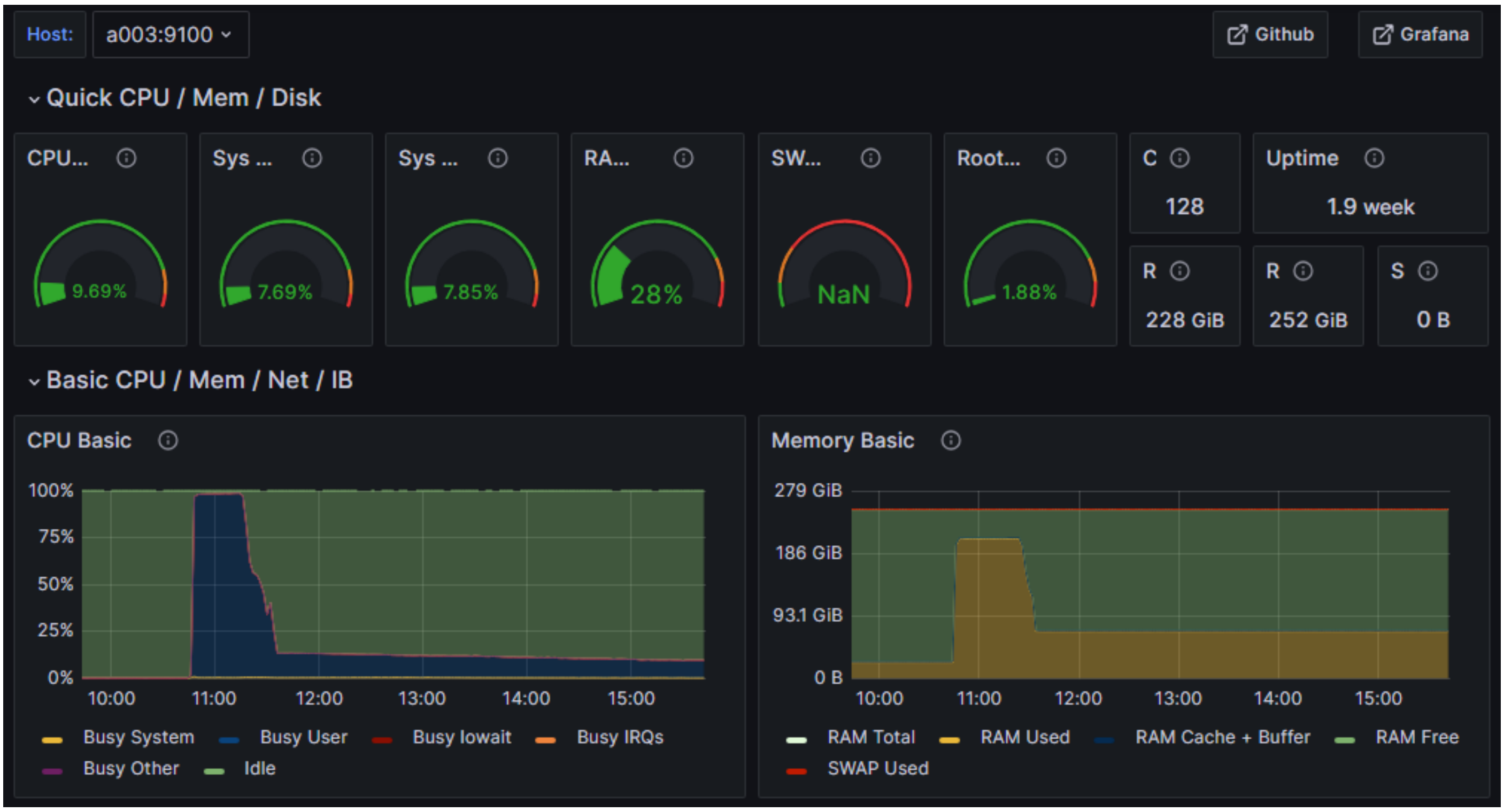
[illegible] Numpy[illegible]

[[illegible] -[illegible] CPU[illegible]]()[illegible]

MKL[illegible] OpenBLAS[illegible] AMD[illegible] `MKL_DEBUG_CPU_TYPE=5` [illegible]

MKL[illegible]

[illegible]



[illegible] eScience[illegible] 40

[illegible] 16[illegible] 40[illegible] “

[illegible] ”

[illegible]

[illegible]

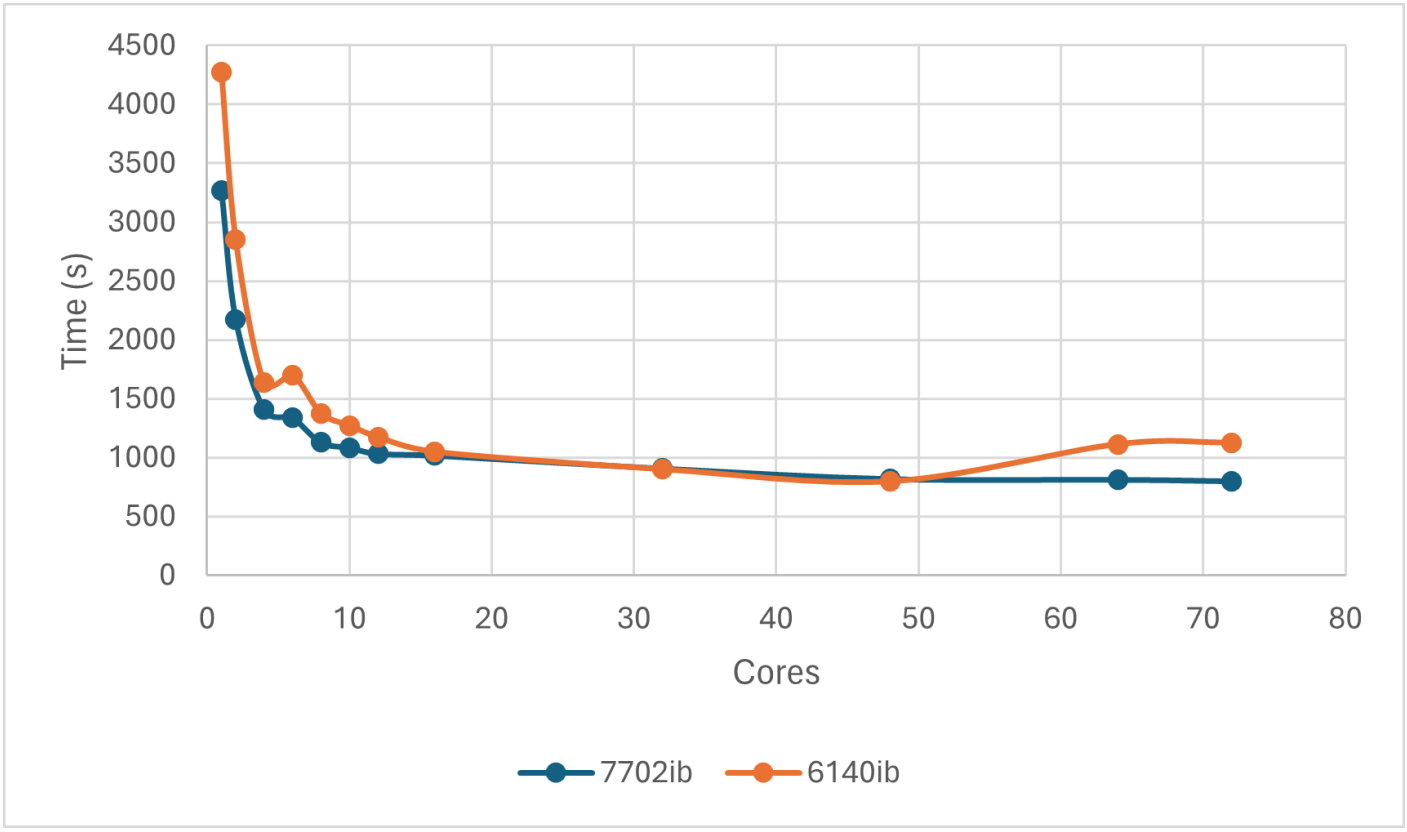
```
(work) [hfd_jitz@login1 test_50nm]$ sh show_results.sh
output_44813828:#BSUB -n 48
output_44813828:    Run time :                                  1726 sec.
output_44813830:#BSUB -n 16
output_44813830:    Run time :                                  835 sec.
output_44813831:#BSUB -n 12
output_44813831:    Run time :                                  5551 sec.
output_44813832:#BSUB -n 10
output_44813832:    Run time :                                  5654 sec.
output_44813833:#BSUB -n 8
output_44813833:    Run time :                                  958 sec.
output_44813834:#BSUB -n 6
output_44813834:    Run time :                                  1252 sec.
output_44813836:#BSUB -n 2
output_44813836:    Run time :                                  2543 sec.
```

最终计算完成的时间统计也剧烈起伏。经过网上的一通搜索，并未发现很多案例。作为猜测，将其它的 LAPACK 优化库限制核数加入，都没有改善。更换 AMD队列，仍然是同样的问题。也就是说，这个情况的出现与是 Intel 还是 AMD的 CPU没有关系。

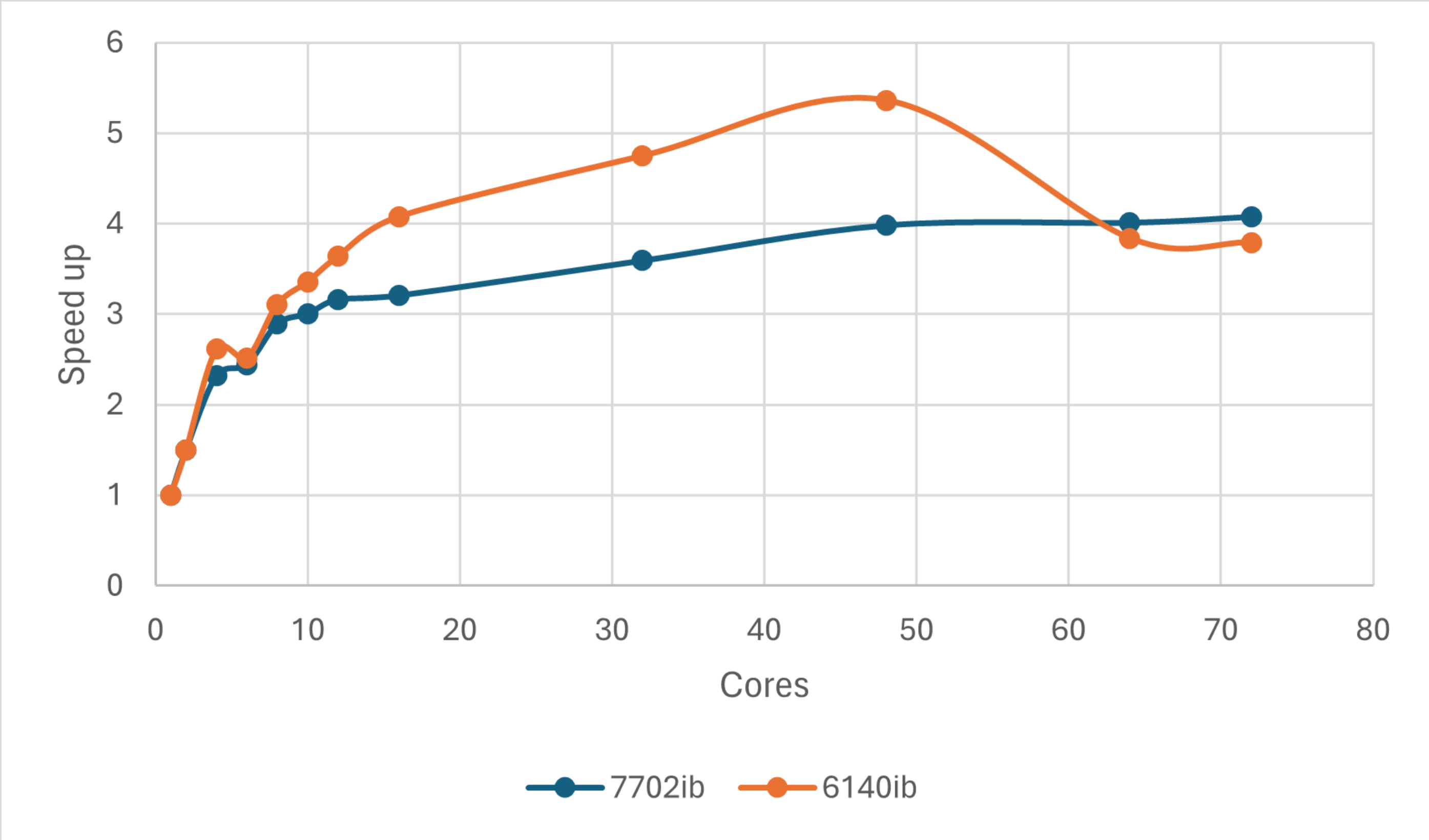
然而最终， 尝试性更换至 **OpenBLAS**版本 **Numpy**，发现问题神奇地解决了。

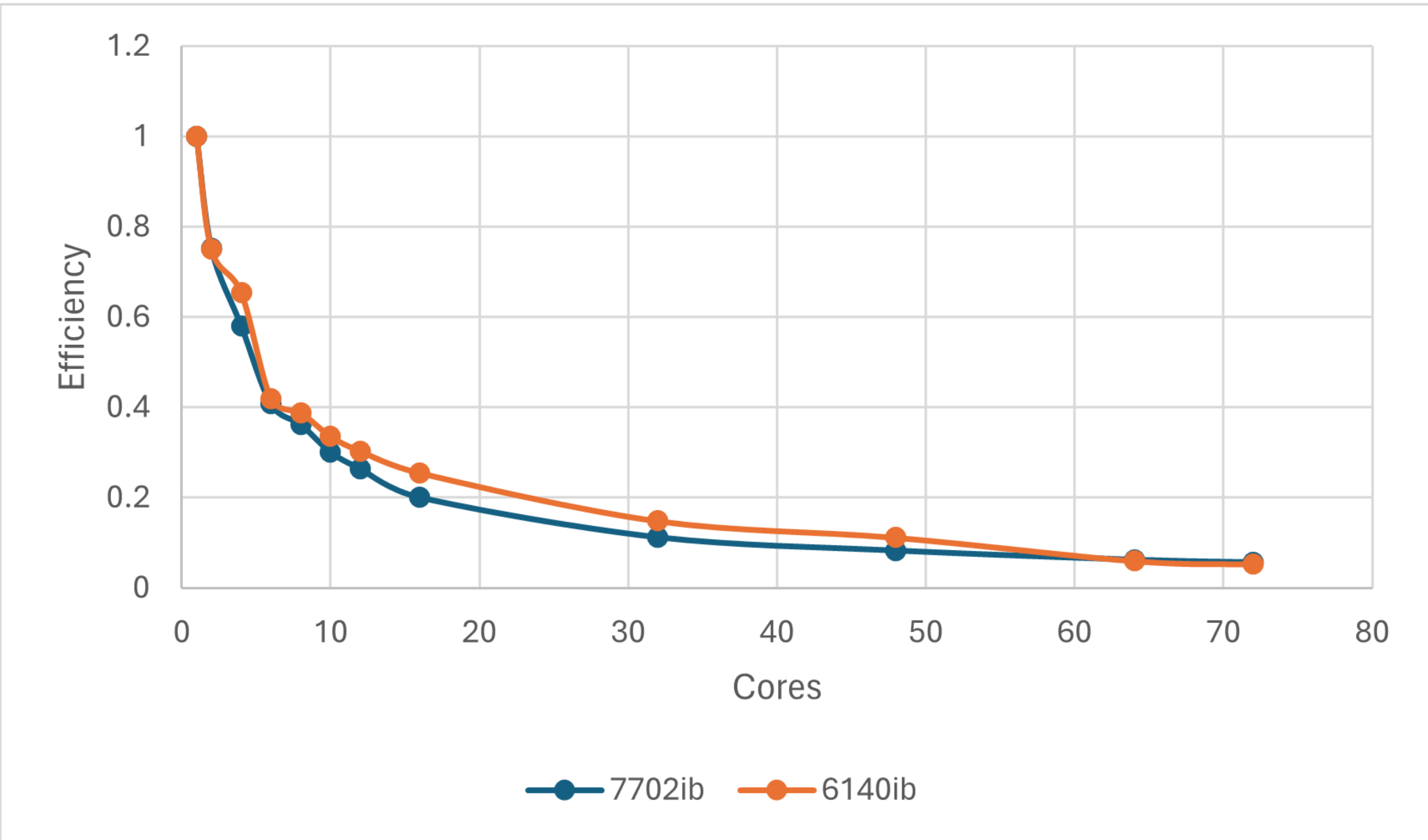
多核情况测试

由于 MKL版本的神秘 BUG，后来的测试通过 OpenBLAS版本进行。多核计算单个任务的总耗时结果如下:

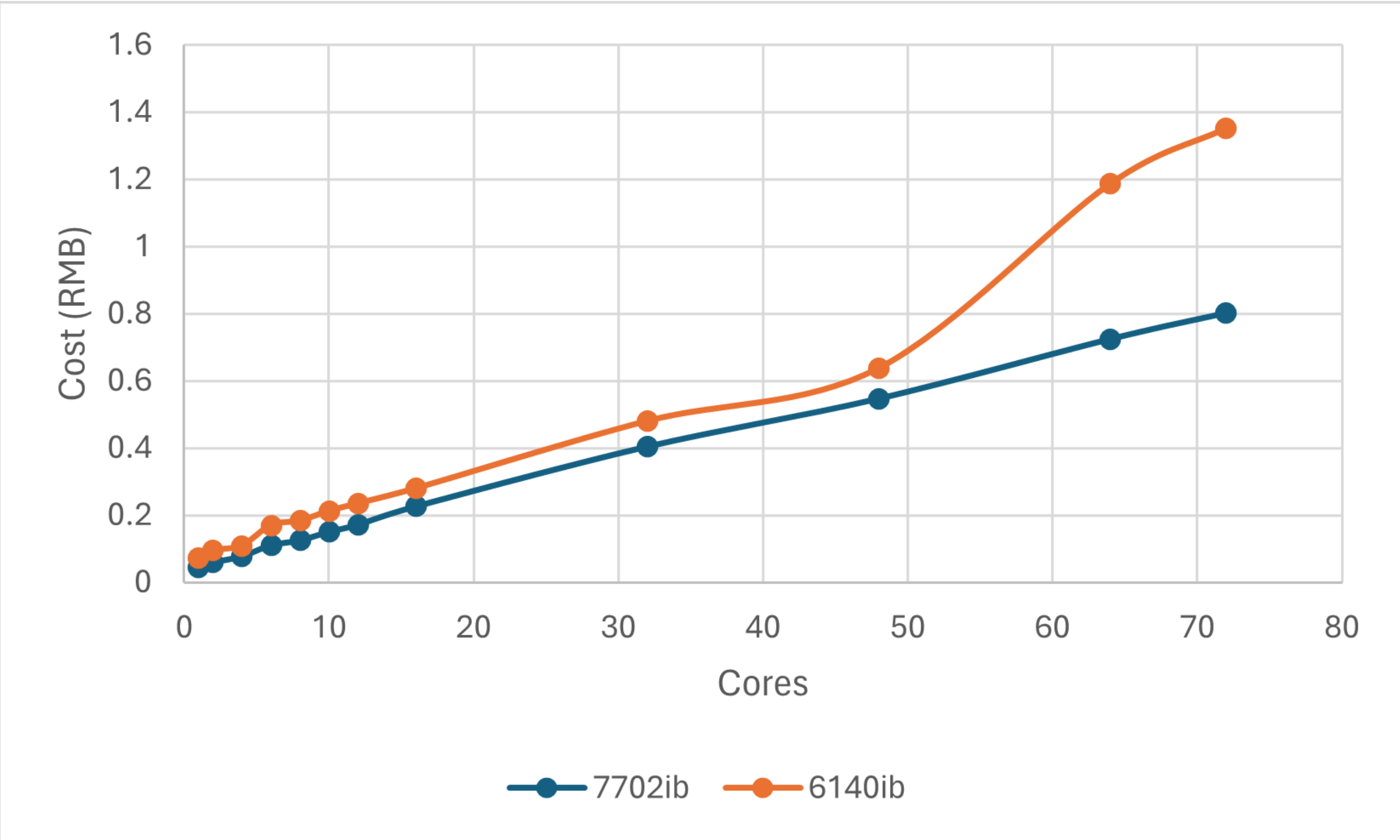


□□□□□ OpenBLAS□ Numpy□□□□ 7702ib□□□ 6140ib□ 48□□□□□□□□□□□□ 48 □□□□□□ 7702ib□□□□□□□□□□□□□□□□□□□□□□□□ 6140ib □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 7702ib□□□□□□□□□□□□□□□□□□ 72 □□□□□□□□□□





从加速比（相对串行计算的时间比值）来看，对于本测试任务，OpenBLAS Numpy的加速比极限约为 4~ 5 倍。



并行效率（相对串行计算的时间比值再除以核数）则随核数增多衰减极快，

核往上效率极低，全靠数量取胜。

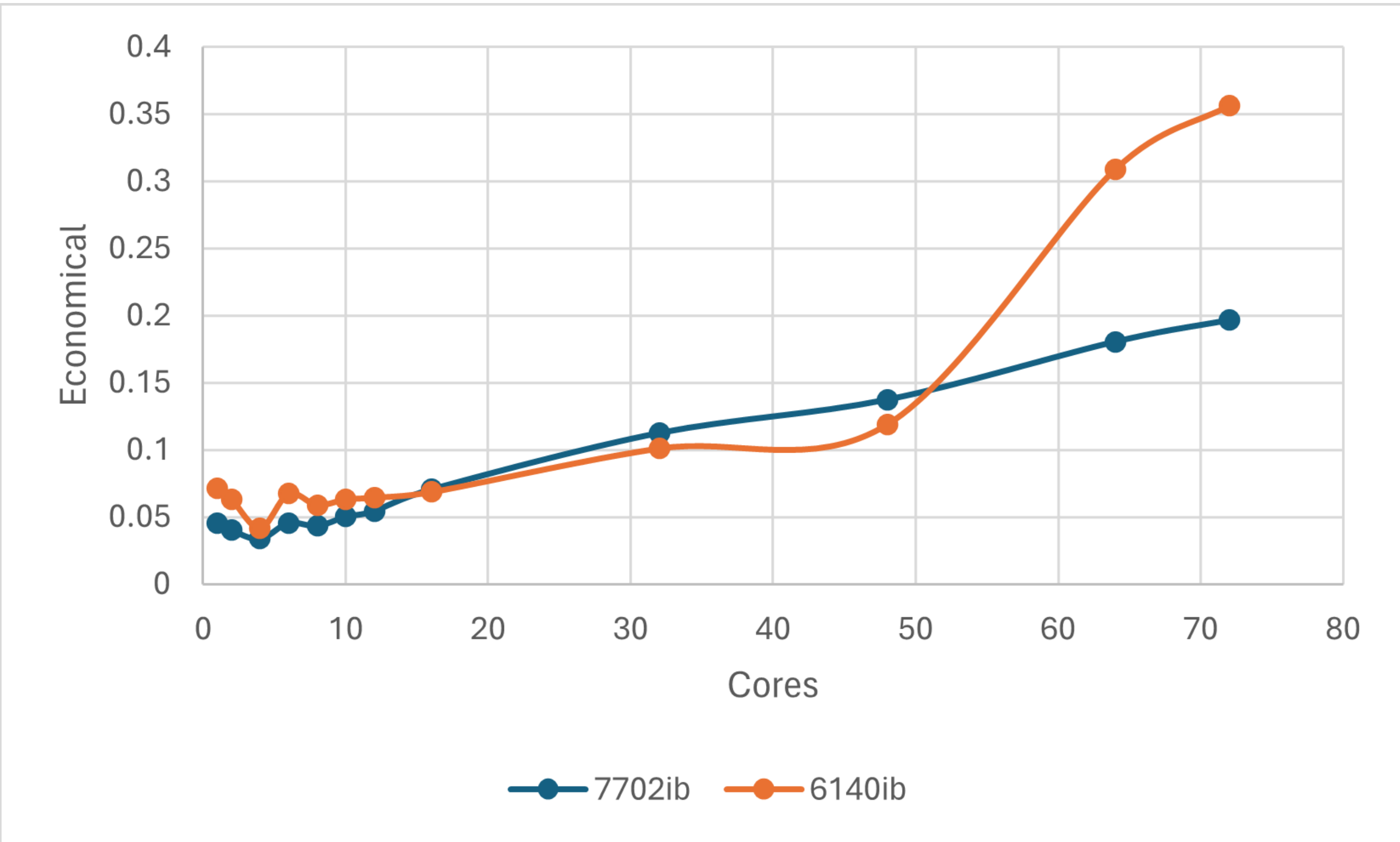
投放任务的策略

对单个任务来说，我们只关心这一次计算的时间，因此时间肯定是越短越好。但是往往时间和核数（准确来说是申请 CPU乘以时间）影响计算费用，因此计算核时的指标额外决定了是否划算。根据

收费办法

我们可以画出实际的花费。显然有些队列上有时候申请过多的核并不一定是划算的，这也是做基准测试的意义所在。

更多时候，我们可能面对的是多个计算量相当的任务，只是计算参数不同。这种情况下，我们可以根据下列公式计算：



$$T_{total} = N_{task} \times \left(\frac{N_{max}}{N_{cores}} \right)^{-1} \times T_p(N_{cores})$$

$$= \frac{N_{task}}{N_{max}} \times N_{cores} \times T_p(N_{cores})$$

其中，N_task 指的是总任务数，N_max指的是考虑申请的总核数，N_cores指的是单个任务申请的核数，T_p是单个任务并行计算的耗时，与单个任务申请的核数有关系。N_cores与 T_p相乘即为单个任务的计算核时。

可以看到，总的核时（T_total乘以 N_max）也和上文的单个任务类似。

单从性价比而言，不同的核数下，我们对单个任务每加速一倍花了多少钱，也可以算出：

对于 Numpy（OpenBLAS）计算本任务而言，性价比最高的莫过于4核。因此综合考虑时间、性价比，4核是个不错的选择。

总结

本次尝试使用集群对 Numpy计算进行加速。在尝试过程中，发现需要注意几个问题：

1. Numpy（并行版本）是隐性并行，所以也需要有特定的环境变量来控制与申请核数匹配，不然会被作业监控杀掉。
2. 基于 MKL版 Numpy或者调用 Numpy的计算包计算时，出现了突然任务变慢的情况，而且会长时间不结束。这与 LAPACK优化库（ MKL或 OpenBLAS或者其他）同当前计算包的兼容性可能有关系，如果发现类似情况，可以尝试排除此类软件依赖的数值计算库是否存在这样的不同优化（甚至编译方式）的区别。
3. Numpy的并行计算存在一个加速效果的上限，对本次测试的任务来说约为 4~5倍。从时间、性价比的角度来说，并非核数越多越划算，例如本任务中，较为经济而且时间可接受的一个合理范围为 4~16核，其中 4核最具性价比；对单个任务，其花费可能尚可，但是日积月累也许是笔不小的数目。

Revision #3
Created 2023-09-11 10:38:43 CST by LadderOperator
Updated 2023-09-14 16:29:19 CST by LadderOperator