VASP 应用运行优化

张文帅

wszhang@ustc.edu.cn

April 8, 2018

Contents

1 动机

在校内乃至全球范围内，VASP 应用占用了很大一部分超算资源，所以对其计算速度与计算有效性的改进非常重要，可以显著提升超算资源的使用效率与用户体验。本文将从应用并行参数方面来展示 VASP 的优化运行测试，希望不仅能给用户的 VASP 使用提供计算指导，还计划能以测试经验为基础，在 VASP 作业运行前，自动化的检测分析用户设定的各种系统与应用的并行参数问题，在不影响计算结果的前提下，给出优化设置建议。本文档简明总结了一些早期的相对丰富的探索，先行提供给用户参考，并期望得到足够反馈，以便进一步改进与完善，得到更加可靠有用的经验基础。

2 问题

首先，引用一段话来表明，在 VASP 计算中，具体问题具体测试的必要性：

> "Geun Ho Gu,
> University of Delaware
> For the NPAR, I recommend doing a test to find out the most efficient number. e.g. run a same calculations multiple times with different NPAR. Also, do the same for LPLANE parameter as well. The manual instructs to use the number of node as NPAR as each parallel calculation can be run at each node minimizing communication overhead between each node. If not optimized, VASP takes extra time to comminucate between nodes, eating up your computation time. However, I have found that this instruction does not always hold up, and, really, this parameter is heavily dependent on the batch server/ node configuration. So, it is wise to do your own test to optimize this parameter (and other parameters as well). "

此外，测试发现，VASP 手册中的一些优化设置建议并非普遍适用。这是否意味着，在每次计算前，总是需要做很多次的测试才能找到最优参数呢？如果我们尽可能完整的测试参数，又将耗费很多测试时间，同时增加了人力与计算机的工作量。这使科研人员陷入两难之地，经常仅靠经验来猜测一个参数，因为速度对科研的结果没有影响，即便晚 50% 时间出结果，在科研上也是可以接受的。但这在资源利用效率上却是不可接受的，其带来了巨大的隐形的浪费。

事实上，已有的测试已经有迹象表明，不同并行进程数的计算具有一组通用的较合适的参数空间 [1]，只是与 VASP 的标准指导手册并不相同，当前的诸多测试报告（包括引文）并没有注意到这一点。这一点的重要之处在于，它可以显著的缩小待测参数空间，使我们有机会通过不多的几次测试便可将作业优化到接近最优的参数点，使得对 VASP 计算进行自动化封装成为可能，本文将努力为读者展示这种可行性，以便自动化的优化 VASP 的运行。

3 计算对象

本次测试的计算对象为六方晶胞结构的 ZrNCl 体系。选择这个体系源于其具有多个特征，如晶胞非立方，结构接近分层，同时轻重元素种类丰富，在只测一个材料的情况下，其具有较多的非平凡代表性，期望后期在用户的反馈下，可以对更多有代表性的体系做测试。本文中，为了全面的测试，我们不仅测试原胞与中等单胞下的多 KPOINTs 计算，也测试中等单胞与超胞下的 Gamma 点计算。

我们取五个代表算例，分别为：

1. 原胞，18 Atoms (Zr6N6Cl6)，272 irreducible k-points
2. 超胞 221，71 Atoms (Zr24N24Cl23)，36 irreducible k-points
3. 超胞 221，71 Atoms (Zr24N24Cl23)，Gamma point
4. 超胞 441，284 Atoms (Zr96N96Cl92)，Gamma point
5. 超胞 661，630 Atoms (Zr96N96Cl92)，Gamma point

以下分别简记为 A18K272，A71K36，A71K1，A284K1，A630K1。前两者因为计算多 K-points，计算程序选用标准的 vasp_std。后三者我们只做 Gamma 点计算，因为在很大的超胞下，达到同样的计算精度，并不需要很多的 kpoints，此时，计算选用 VASP 特别为 Gamma 点计算优化的版本 vasp_gam。

4 并行参数优化

4.1 应用输入参数

计算参数设置:

SYSTEM = ZrNCl
ISTART = 0
ISMEAR = 0
SIGMA = 0.4
ENCUT=400
PREC=Normal
NELM = 5
NELMIN = 5
NELMDL = 0
ISYM = 1
EDIFF = 1E-7
LREAL = Auto
LPLANE = .TRUE.
KPAR = $KPAR # 1 2 4 8 16 default: 1
NCORE = $NCORE # 1 2 4 8 16 default: 1
#NPAR = $NPAR # 4 6 8 16
#NSIM = $NSIM # default:4

本文主要讨论最重要的运行核心数与 KPAR、NPAR/NCORE 并行参数的关系，KPAR 指代有多少 KPOINT 被并行的处理，NCORE 指代一个 BAND 在多少 CPU 核中进行计算，NPAR 指代总计有多少个 BAND 并行处理，NCORE 与 NPAR 只能指定一个有效。一般而言，假设总运行核心数为 32，KPAR=4，NCORE=4, 那么意味着，有并行计算 4 个 KPOINTs，每个 KPOINT 使用 8 个核心，同时每个 KPOINT 内部，4 个 CPU 核共同做一个 BAND，总计有 2 个 BAND 在并行。所以 KPOINT 并行属于外层的并行，BAND 并行属于较内层的并行。当人 VASP 内的并行过程还很多，测试工作量也很大，所以当前 VASP 应用的运行时并行效率优化还会有较大的提升空间。本测试仍有 NSIM、ECUT 等其他较重要的参数使用默认值或常用值，并未测试他们对结果的影响等等，这将是今后需要继续完善之处。

4.2 应用参数优化结论

我们先给出一些经常在官方手册、VASP 程序屏幕输出及网络上出现的不合适的设置建议，他们可能因为时间的久远，考虑硬件较旧，或者测试算例的不同，而导致在 TC4600 上很多不同类型的节点都不合适。本文的测试中，考虑了众多硬件条件，与不同体系大小与 KPOINTs 下的多种代表算例，以此得出的结论，应该具备更普适的优势，同时将来会进一步的通过自动化系统的运行更加确证或改进本文的结果。

常见的不适用的设置建议

- NPAR = 4 ~ approx SQRT(number of cores)
  - 运行时屏幕经常输出：“For optimal performance we recommend to set NCORE = 4 - approx SQRT(number of cores) NCORE specifies how many cores store one orbital (NPAR=cpu/NCORE). This setting can greatly improve the performance of VASP for DFT. ”
  - 此建议在 KPOINT 很多时特别不适用，其他条件下也不总是最优。
- NPAR = number of cores per compute node [2]
  - 测试发现，在本测试系统的 2 路节点中，这种设置不合理，经常不是最优。
- not recommend attempting run with KPAR>compute nodes, even though you may have more k-points than compute nodes. [3]
  - 测试结果表明，单节点较多核心的 E5V4 节点，在 KPOINTs 较多时，最优的运行 KPAR 值远大于节点数。

优化设置结论

- NCORE 比 NPAR 具有更小的最优取值空间，可以更好的适应不同的并行核心数与节点硬件

  - 在文献 [1] 中，测试了 NPAR 对运行时间的影响，对应的 NCORE 值被忽略。但是，当我们计算一下相应的 NCORE 后可知，最优 NPAR 的取值范围的上下边界值相差数十倍，而对应的最优 NCORE 空间的上下边界值基本保持数倍以内，此结果与本文档测试结果基本符合：在不同节点类型 (甚至包括构架很不同的 4 核心 E3V5 节点)，不同总并行核心数，与不同的算例下，NCORE = 8 下的运行时间经常处于最优值，且基本处于 [4,16] 空间范围内。

- VASP 默认并行参数（KPAR=1 & NCORE=1）非常低效，最优的运行参数可大大提高并行扩展性与运行速度
  - 在 E5V4-A18K272、E5V4-A71K36 例子中，默认设置下的并行极限为 24、64 核心，但是经过优化并行参数后，可以轻松扩展到 256、128 核心，并仍保有进一步扩展空间，最大运行速度可以提高 10 倍。特别得，在 630 个原子的超大晶胞算例 E5V4-A630K1 中，经参数优化后，不仅并行核心数从 256 提高到 384，尤其运行时间从 354s 降低到 183s。
- 由 K 点数 NKpoints 与原子数 Natoms 两者，可大致估算最佳运行并行核心数 [3]
  - 当仅作单 KPOINT 计算时，并行核心数可扩展到大约 Natoms/2
  - 当进行多 KPOINTs 计算时，并行可进一步扩展 8-16 倍。
  - 由于初始输入的 KPOINTs 可约，所以准确的 NKpoints 应该以考虑对称后的约化值为准，目前在 KPOINTs 较大时，暂没有考虑具体数值对并行扩展性的精确影响。
- 当单节点核心数可被 NCORE 整除时，能够在部分多节点计算算例中增加效率
  - 单节点核心数可被 NCORE 整除时，可使 BAND 并行通信限制在节点内，理论上总会带来好处，否则。
  - 实践上，在单 KPOINT 多节点算例中，BAND 通信影响较小，原因可解释为多节点的单 KPOINT 计算本身的通讯时间很长，抑制了“可整除”带来的 BAND 并行通讯降低的好处。
  - 在多 KPOINT 多节点算例中，当设置 KPAR 较大时，单 KPOINT 在 1-2 个节点内运行，总体通信较小，此时 BAND 通信占比更大，故而影响更加突出，“可整除” 带来的 BAND 并行通讯降低，会显著降低运行时间。如在图7.2 A18K272 算例中，当 KPAR 从 1 增大到 4 时，NCORE 为 8/7 时的运行时间从相差无几的 130/124，变为 40/28，具有显著的差异。更进一步，在图7.2中，详细展示了对于 A18K272 体系，两节点 56 核心，KPAR 设为 1，NCORE 分别取 8 与 7 时的性能分析图，可知 NCORE 为 7 时相比 NCORE 为 8 时，通信占比从 79% 降到 72%，在纯计算时间保持约 50s 不变的情况下，总的运行时间从 254s 降低到 182s。

5 硬件适应性分析

5.1 测试硬件

本次测试的硬件种类丰富，包括四种 Intel Xeon CPU 节点 (如新旧普通节点：E5V3/E5V4，高主频节点：E3V5，胖节点：Fat144)，以及两种模式下的 Xeon Phi KNL 节点 (本文将默认设置的 All2All Cluster mode & Flat Memory Mode 简称为 AF Mode，将 Quadrant Cluster mode & Cache Memory Mode 简称为 QC Mode)。本文不加入对 GPU 平台测试结果的分析，因为我们的测试结果并没有提供比已有的报告更新的见解，已有的 GPU 测试报告表明，目前 VASP & GPU 无法成为有竞争力的计算平台 [4]。

节点类别	CPU	内存 (DDR4)	硬盘	计算网络
E5V4	2*E5-2680 v4(2.4GHz-3.3GHz, 35MB L3 Cache)，共 28 核	128GB 2400MHz	240GB	100Gbps OPA
E5V3	2*E5-2680 v3(2.5GHz-3.3GHz, 30MB L3 Cache)，共 24 核	64GB 2133MHz	300GB	56Gbps FDR
E3V5	1*E3-1240 v5(3.5GHz-3.9GHz,8MB Cache)，共 4 核	32GB 2400MHz	500GB	100Gbps EDR
Fat144	8*E7-8860 v4(2.2GHz-3.2GHz,45MB L3 Cache)，共 144 核	1TB 2400MHz	480GB	100Gbps OPA
KNL-AF	1*Xeon Phi 7210(64 核，1.3GHz-1.5GHz，16 GB MCDRAM，AF Mode)	96GB 2133MHz	160GB	100Gbps OPA
KNL-QC	1*Xeon Phi 7210(64 核，1.3GHz-1.5GHz，16 GB MCDRAM，QC Mode)	96GB 2133MHz	160GB	100Gbps OPA

Table 1: 测试系统硬件列表

5.2 硬件适应性优化结论

- 在科大 TC4600 集群的配置下，E5V4 节点均比 E5V3 具有明显的速度提升与更好的并行扩展性，速度提高达 10%-30%
  - E5V4 节点相比 E5V3 主频稍低，睿频相同，核心数较多，Cache/核心数之比相同，配合的 OPA 通讯设备较快。测试数据表明，E5V4 节点的通讯优势掩盖了主频劣势，总体性能更好。
  - 在较大体系 A284K1 下，E5V4 节点比 E5V3 的优势在最佳应用并行参数下更明显。原因考虑为：在大体系与最好的并行参数下，通信占比被优化达到极致，进而通讯性能的微小差异将给速度带来更加明显的影响。
- E3V5 节点相比 E5V4 节点，在通信优化较好的并行参数下，具有明显快的计算速度，但在通信优化不佳的并行参数下，速度提高不显著甚至会更低
  - E3V5 相比 E5V4 节点具有显著更高的主频，对串行程序会具有明显的优势。对于 VASP，我们需要考虑通信差异，虽然两种节点都配置了相似的 100Gbps 通信网，但是 E3V5 单节点核心数较少，故而跨节点通信的比例会高很多。所以，我们更加需要选择优化的并行参数，以使 VASP 在 E3V5 上的运行更具优势。例如，A18K272 测试表明，在 128 核心下的最佳并行参数下，E3V5 运行时间为 24s，比 E5V4 的 31s 运行时间，性能上有较大改进；但是在 KPAR x NCORE 设置为 1 x 16 时，在所有运行核心数下，E3V5 的运行时间都要明显长于 E5V4 的运行时间。
- E3V5 节点核心数少，在多原子体系需要更多并行节点，会在较低的核心数下遭遇通讯瓶颈，不适合运行大规模的 VASP 计算
  - 如 A71K36 算例，E3V5 在 96 核心时具有最佳的速度，更多核心反而带来速度的降低，而 E5V4 则在 128 核心区间依然有很好的并行效率。
  - 对 Gamma Only 计算，如 A284K1，在 128 核心时，E3V5 虽然运行时间更短一些，对比 E5V4 分别为 36s/44s，但是两节点类型下，128 核心下的计算时间相比 96 核心时的降低幅度分别为 0.86/0.77，表现出 E5V3 节点下的运行时间降低更慢，显示 E5V4 下的并行扩展效率更好一些。
- Fat144 节点具有 8 路 18 核心 CPU，内存较大，在运行小并行规模（32~48 核心以内）的大中小体系时有明显优势，但是在多并行核心时，反而没有 E5V4 队列的并行扩展性好，建议仅运行特殊需要大内存的 VASP 作业。
  - 在图（7.5）中，可以发现，对于多核心（64 核心以上时），Fat144 下的 VASP 并行扩展性比 E5V4 较差，反应此时多路 CPU 间的内存共享通信出现瓶颈，相关的详细理解应该更仔细的考虑进程在 CPU 核心间的分配与通讯问题，由于篇幅时间关系，暂留作后文研究。
- KNL 节点具有 64 核心，每核心有 4 个矢量加速核，在开启超线程下，我们发现 QC 模式相比默认的 AF 模式具有全面更优的运行效率，同时在每个 KNL 节点运行超过 64 核心的单个 VASP 作业并不会带来更好的效果，在 AF 模式效果更差。
  - 在图（7.6）中，KNL3 下的 A284K1 算例，相比 KNL25 下的 A284K1 算例，在 128 并行进程下，前者的运行时间为后者的 2 倍还多，说明 AF 模式下，不适合在单节点运行超过 64 并行进程的单 VASP 作业。
  - 比较
- QC 模式下的单 KNL 节点，对比 E5V4 单节点，都取优化的运行参数时：在小体系 A18K272 下，运行时间比约为 120 比 90，E5V4 具有单节点计算优势。但是在大体系 A284K1 下，运行时间比约为 130 比 140，稍具有优势。
  - 需要提及，我们另测试了 Intel 提供的一个 56 个原子的中等体系的算例，结果单机 E5V4 与单机 KNL 下的运行时间比约为 3637: 1997，KNL 单机的计算优势很大，这说明 KNL 的计算效率比较依赖于计算体系。
  - 如果结合 KNL 单机的价格较低，则现有数据表明，单机 KNL 运行大中体系的 VASP 作业具有较高性价比。
- 比较 KNL 与 E5V4 节点的多节点并行最佳计算时间，KNL 节点的并行扩展性较低。
  - 对比图（7.7）与图（7.1），在 A18K272 体系，KNL 节点可以有效扩展到 96 核心，计算时间约为 123s，E5V4 节点可以有效扩展到 256 核心，计算时间约为 22s；在 A284K1 体系，KNL 节点可以有效扩展到 48 核心，计算时间约为 133s，在 E5V4 节点可以有效扩展到 96 核心，计算时间约为 52s。
  - 此结果表明，KNL 在最佳的计算速度上大大受限于并行扩展度，在需要快速出结果的计算上具有劣势。

6 应用编译优化

6.1 编译选项与数学库

本部分测试中，可调节的编译参数为：

- Intel MKL：Sequential/OpenMP
- ScaLAPACK：Enable / Disable
- FFT implementation: Intel wrapper / Juergen Furtmueller (JF)
- DCACHE_SIZE：4000 / 0

默认情况下（前面测试中使用的），我们的编译参数为 V8: Intel MKL Sequential & Enable ScaLAPACK & Intel FFT & DCACHE_SIZE = 4000

本文另外测试 3 种其他的编译版本，分别为：

V12：JF FFT

V14：Disable ScaLAPACK

V16：DCACHE_SIZE = 0

此外，在 V16 编译版本中，因为官方的介绍"CACHE_SIZE=0 has a special meaning. It performs the FFT's in x and y direction plane by plane", 所以我们额外调整了结构的基矢方向，将 abc 调整为 cba，以测试改变 z direction 的影响。

6.2 编译优化结论

- 对比 V8 与 V12，Intel FFT 相比 JF FFT 显著提升了 VASP 的运行速度，在小体系与大体系下提升可达 2/5，1/5，FFT 效率的提升也使得 VASP 整体并行扩展性更好
- 对比 V8 与 V14，开启 ScaLapack 可以显著提升多节点下（NP>24/28）的运行时间与并行扩展效率
- 对比 V8 与 V16，讲 VASP 官方文档中建议优化的 DCACHE_SIZE 参数设为 0 后，V16 版本并没有迹象影响运行速度与并行扩展效率，当改变结构的 Z 轴方向后，仍然没有明显的迹象。

7 测试数据汇总

7.1 E5V4 节点运行结果

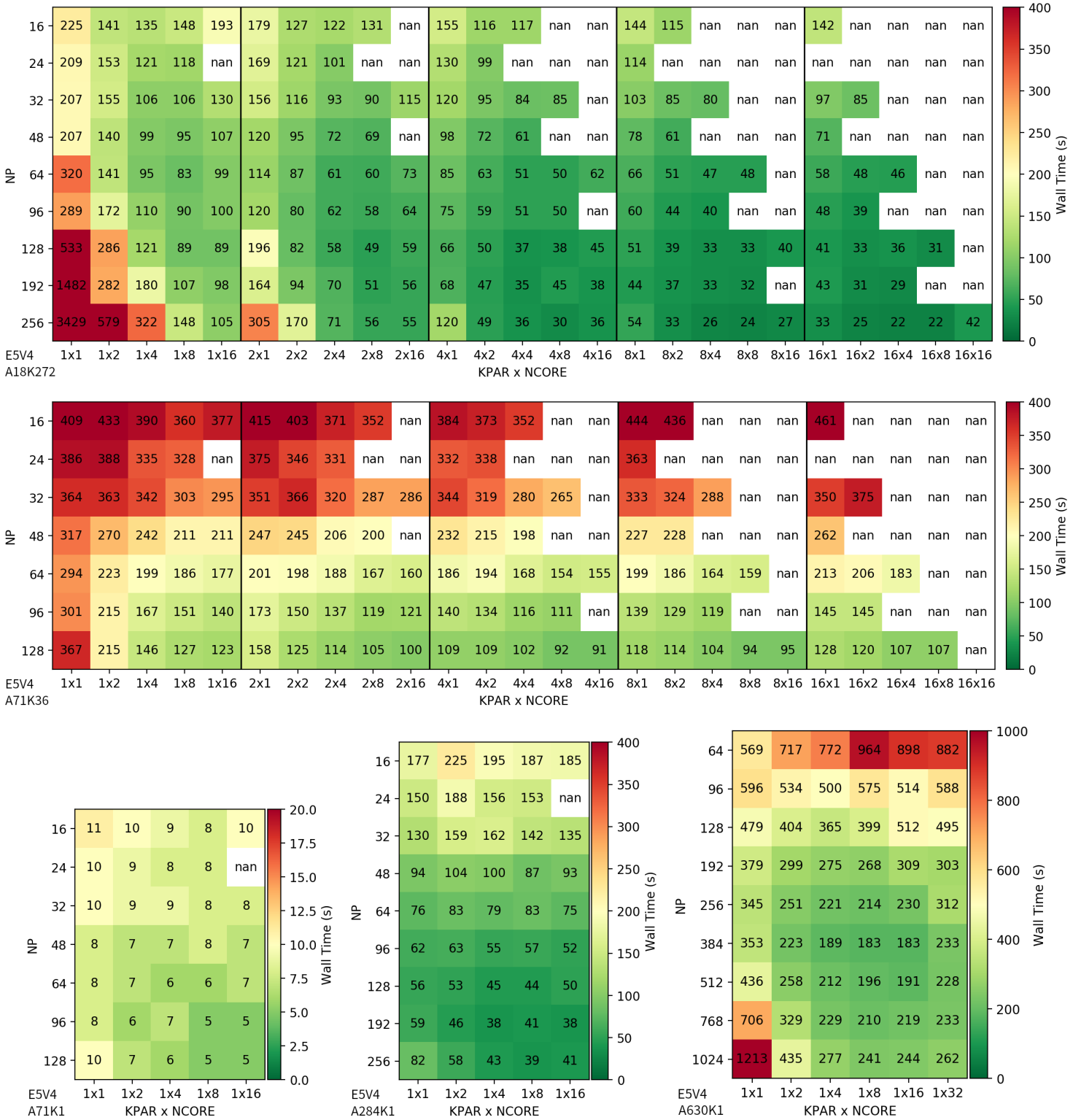

Figure 7.1: E5V4 节点下，5 种结构的测试结果（图片坐下角为运行平台与算例类型）

7.2 测试单节点核心数可被 NCORE 整除时是否更优

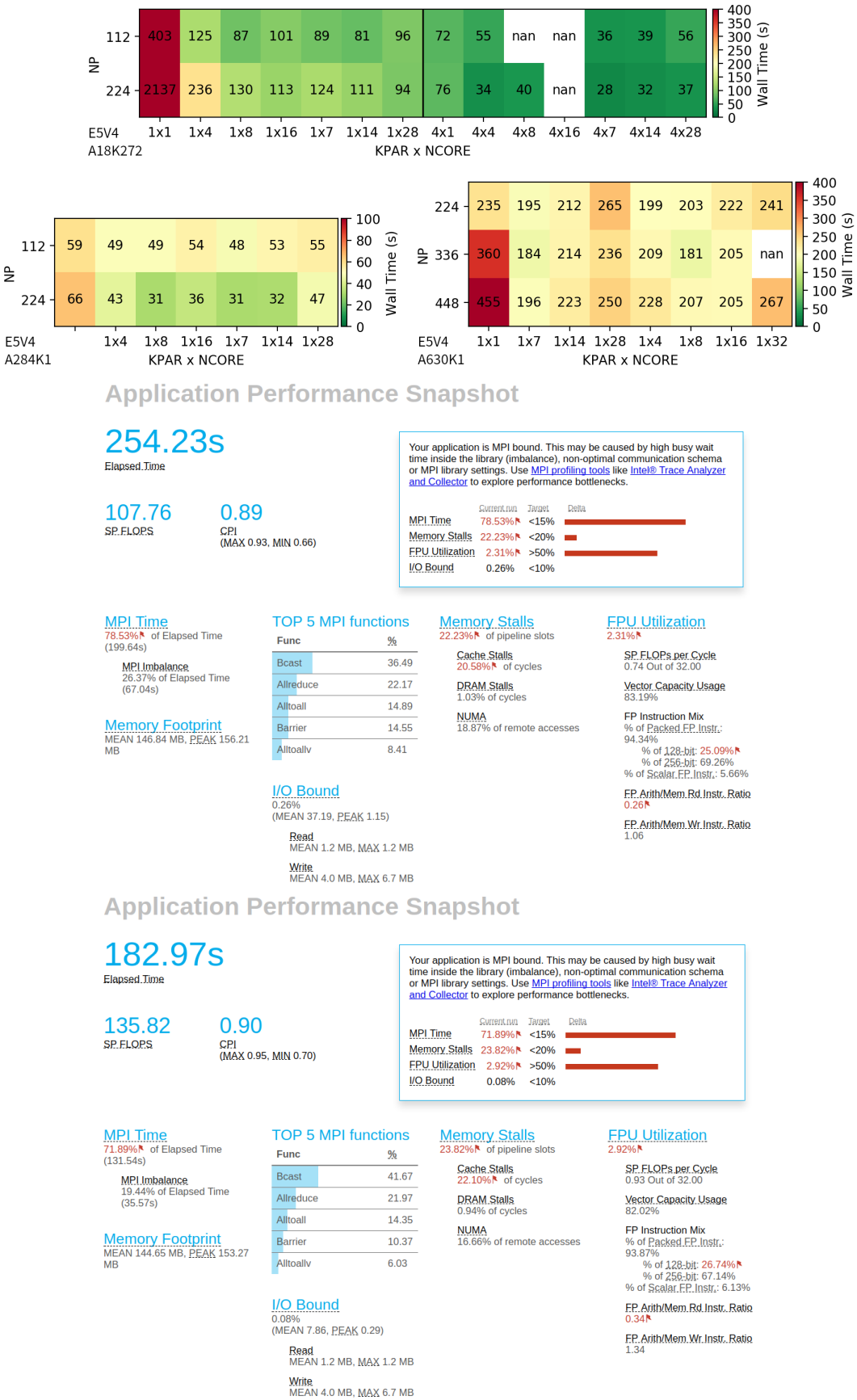

Figure 7.2: E5V4 节点下，测试核心数 28 可被 NCORE 整除是否会更加有益。除前面 3 张常规测试外，后面两张展示了，对于 A18K272 体系，在两节点 56 核心，KPAR 设为 1，NCORE 分别取 8 与 7 时的性能分析图。

7.3 E5V3 节点测试结果

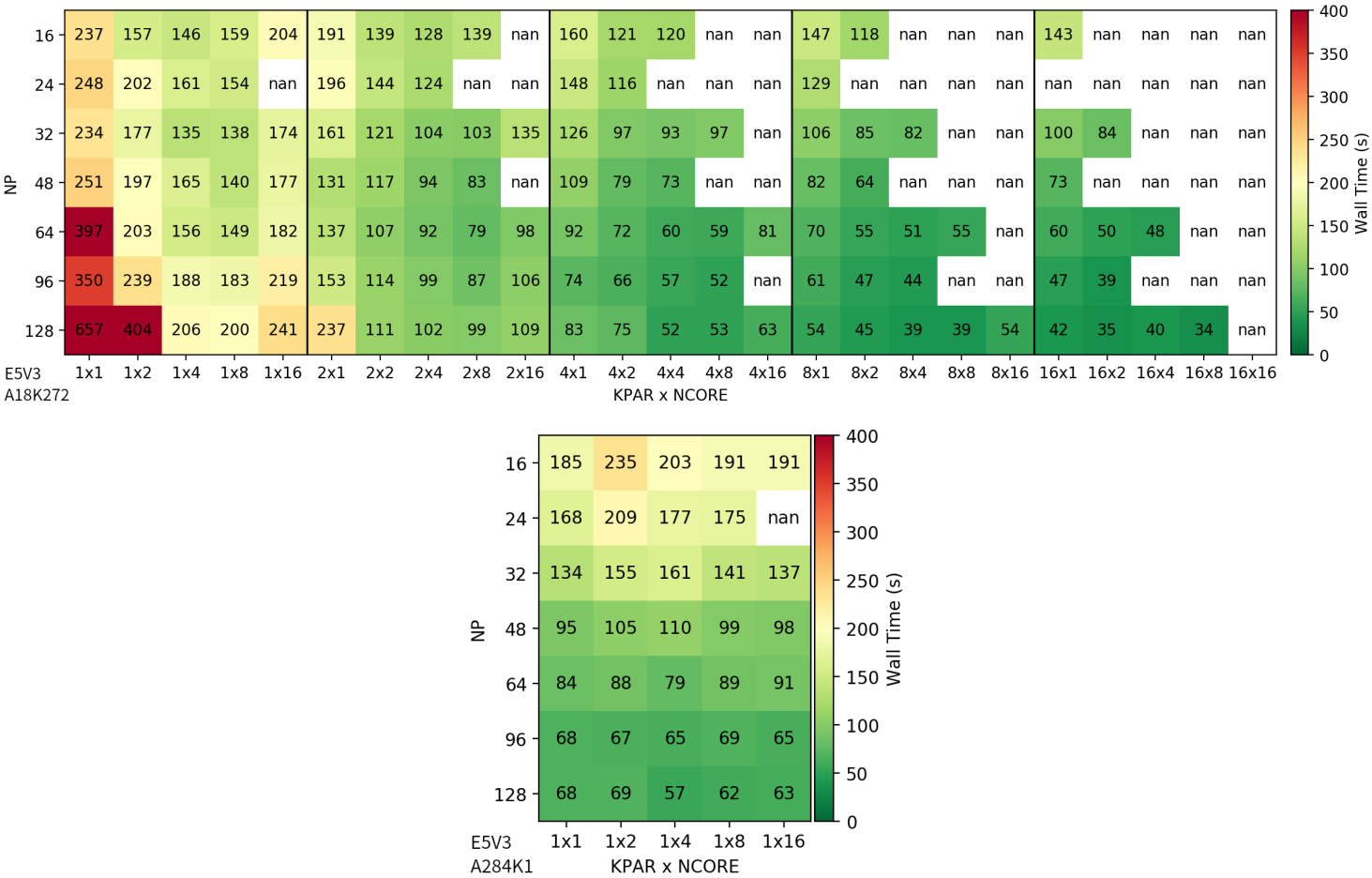

Figure 7.3: E5V3 节点下的测试结果（图片坐下角注明了运行平台与算例类型）

7.4 E3V5 节点运行结果

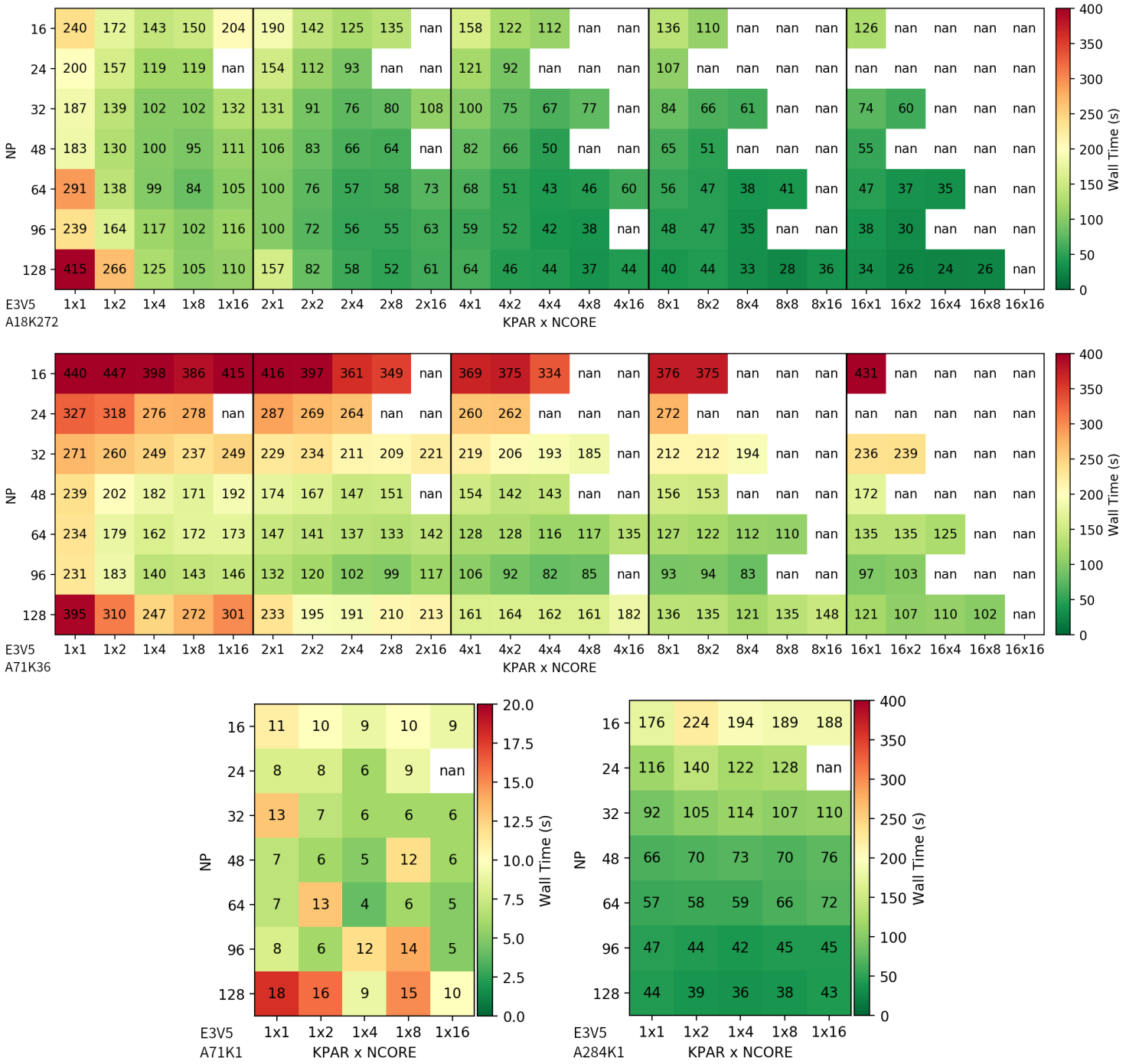

Figure 7.4: E3V5 节点下的测试结果（图片坐下角注明了运行平台与算例类型）

7.5 Fat144 节点运行结果

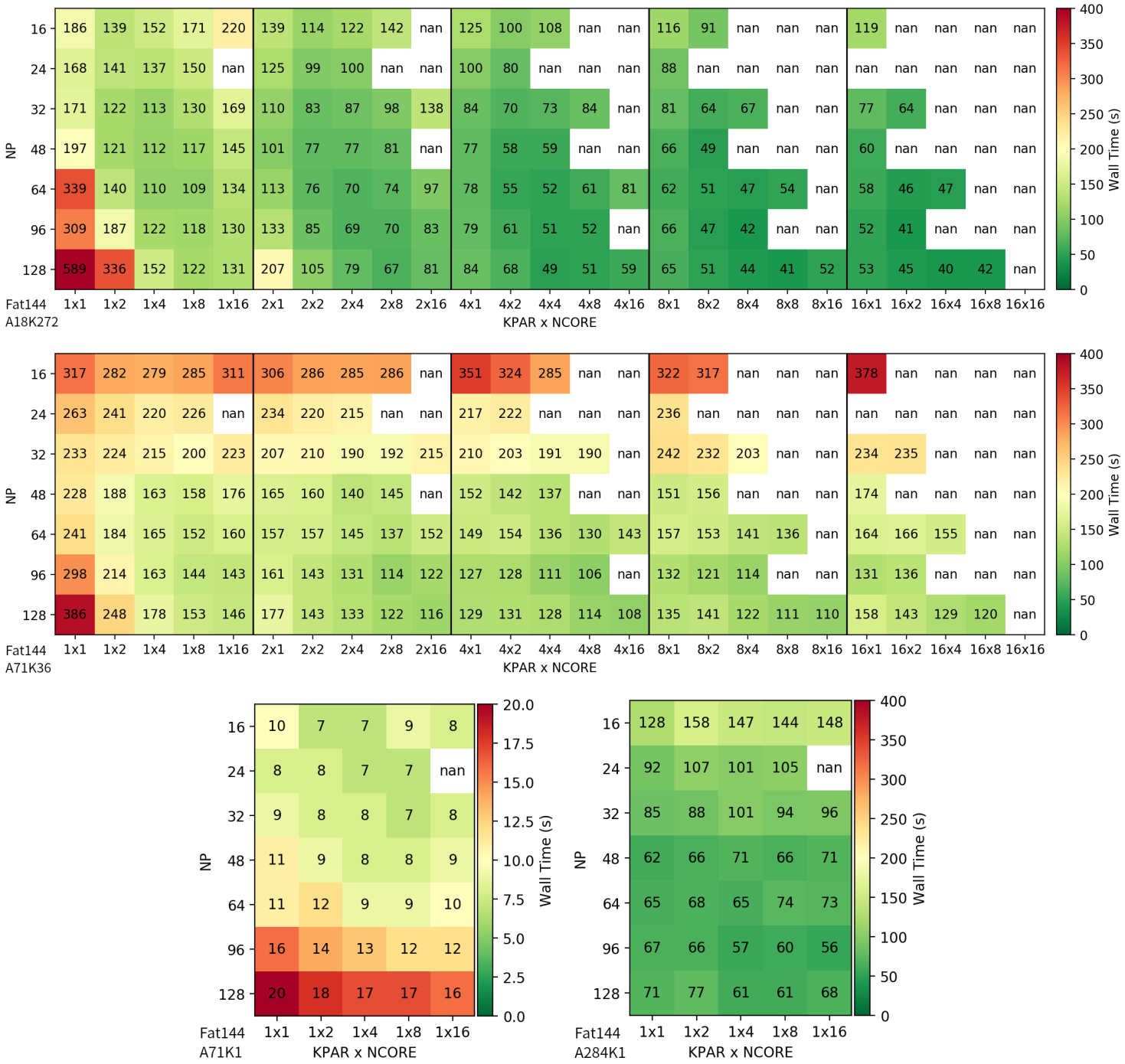

Figure 7.5: Fat144 节点下的测试结果（图片坐下角注明了运行平台与算例类型）

7.6 KNL 节点运行结果 (开启超线程)

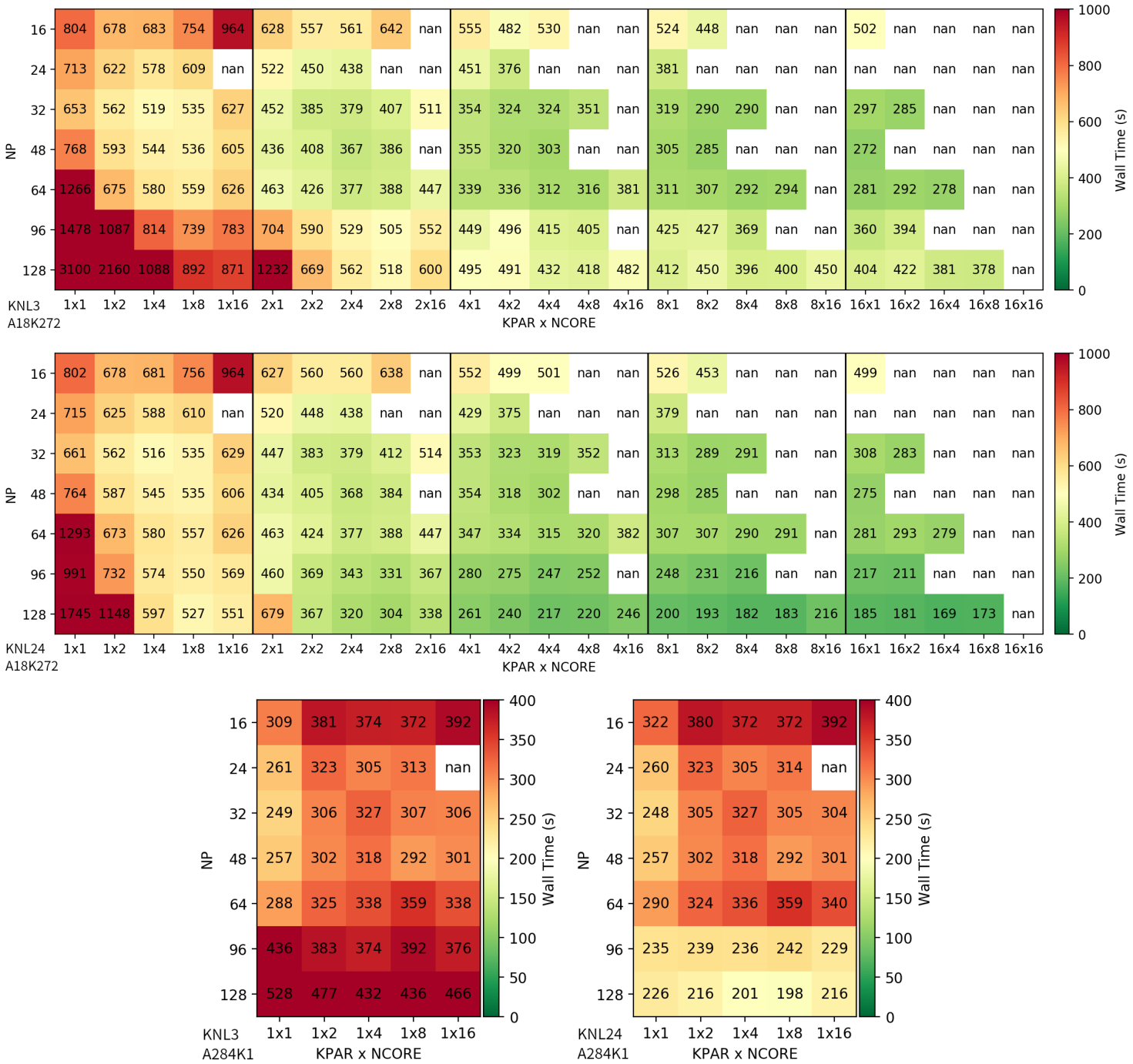

Figure 7.6: KNL 节点 AF Mode 下的测试结果（图片坐下角注明了运行平台与算例类型，KNL3 表示作业分配到 KNL3 单节点上运行，每节点最多 128 进程。KNL24 表示作业分配到 KNL2 与 KNL4 两个节点上，每节点最多 64 进程）

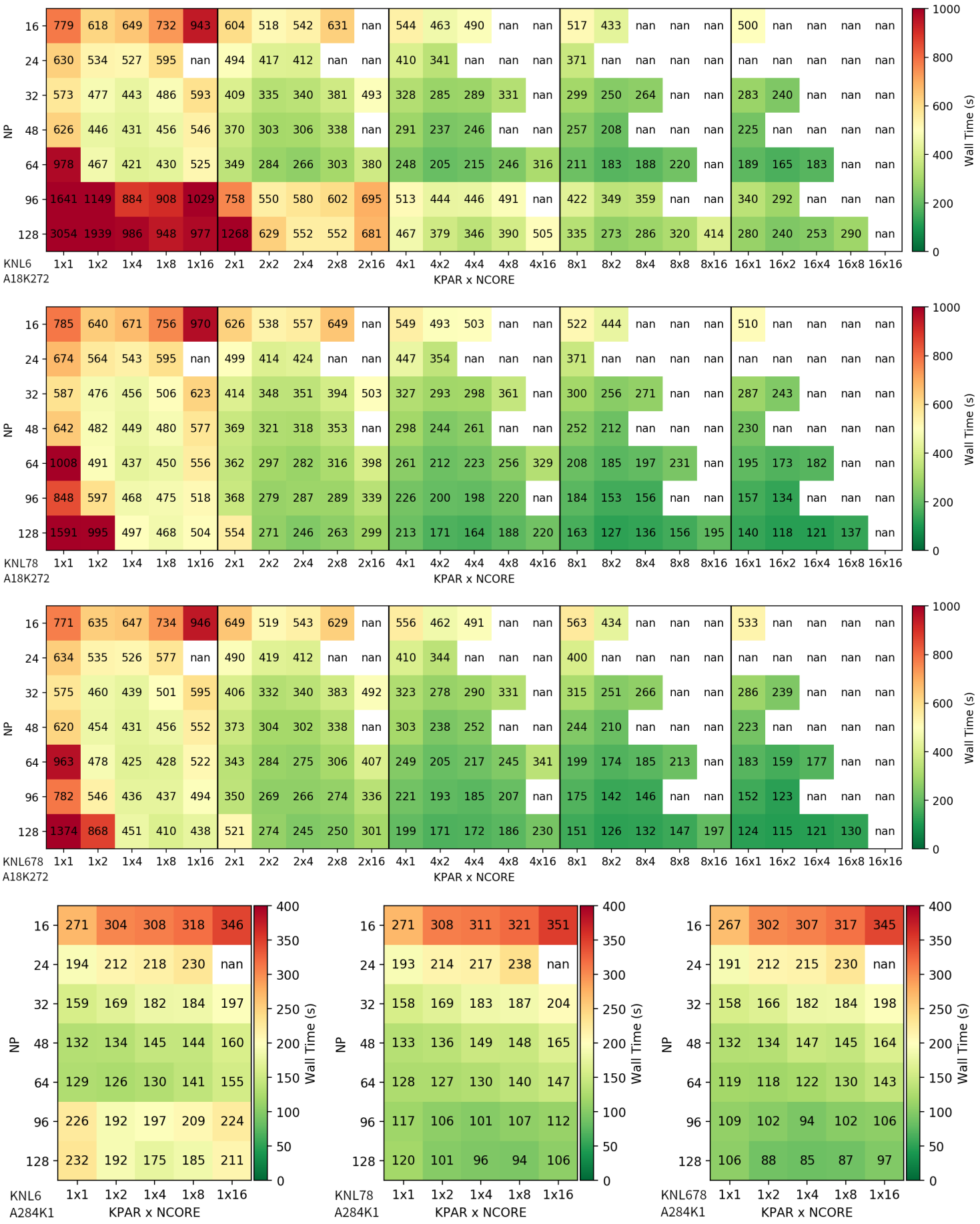

Figure 7.7: KNL 节点 QC Mode 下的测试结果（图片坐下角注明了运行平台与算例类型，KNL6 与 KNL78 含义同上，KNL678 表示作业分配到 KNL6、KNL7、KNL8 三个节点上，每节点最多 60 进程）

7.7 多种编译选项与数学库测试结果

本测试部分，时间仍取前五个电子步的运行时间总和，与之前测试不同之处在于，应用输入参数 NELMDL 取为默认值（即前五步中，以固定的初始电子密度构造新的哈密顿量），因此与五步之后的电子步运行稍有差异，因为本项测试仅比较不同编译参数下的相对速度，不研究最佳运行速度大小，所以影响不大。

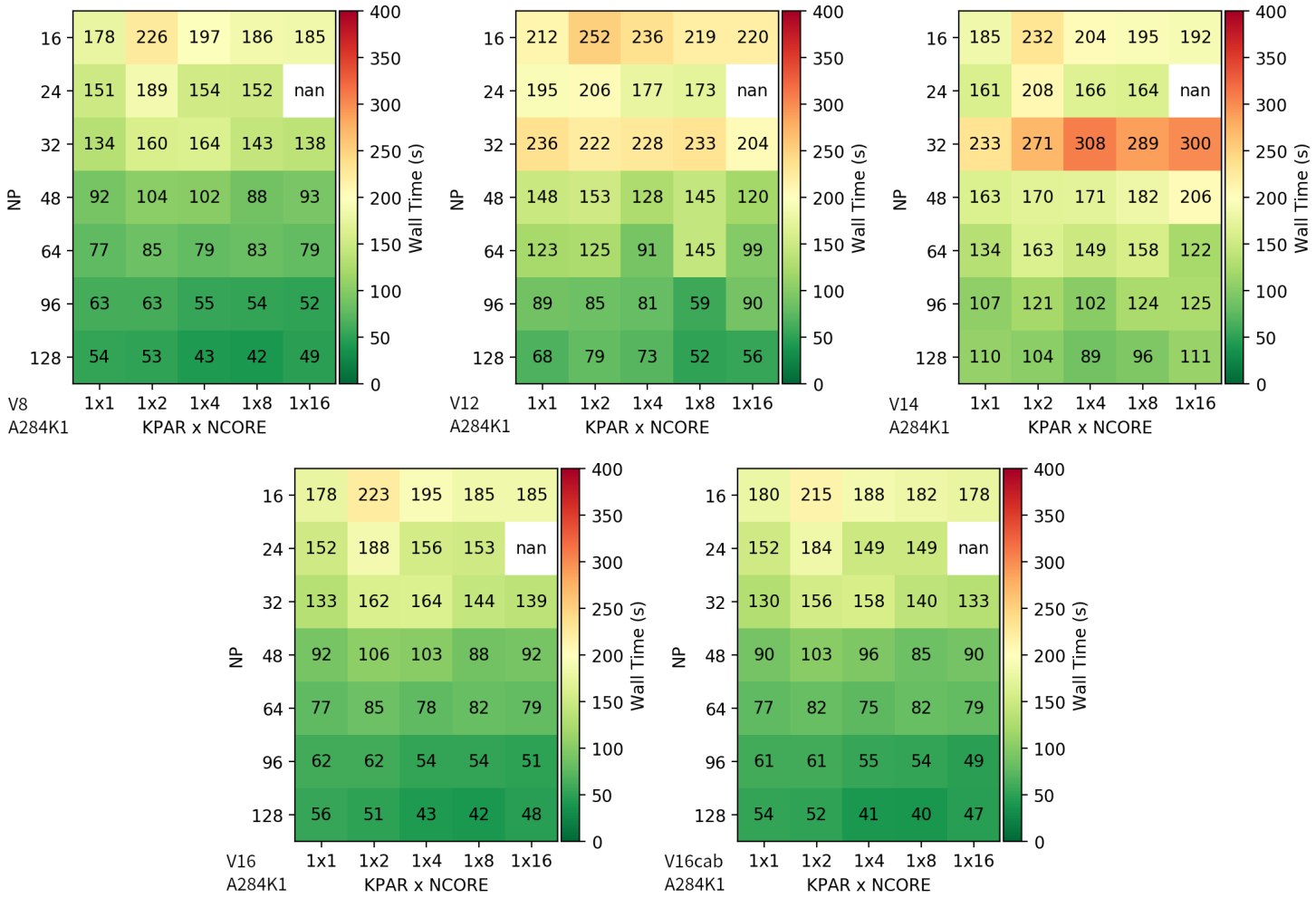

Figure 7.8: 多种编译选项与数学库测试结果

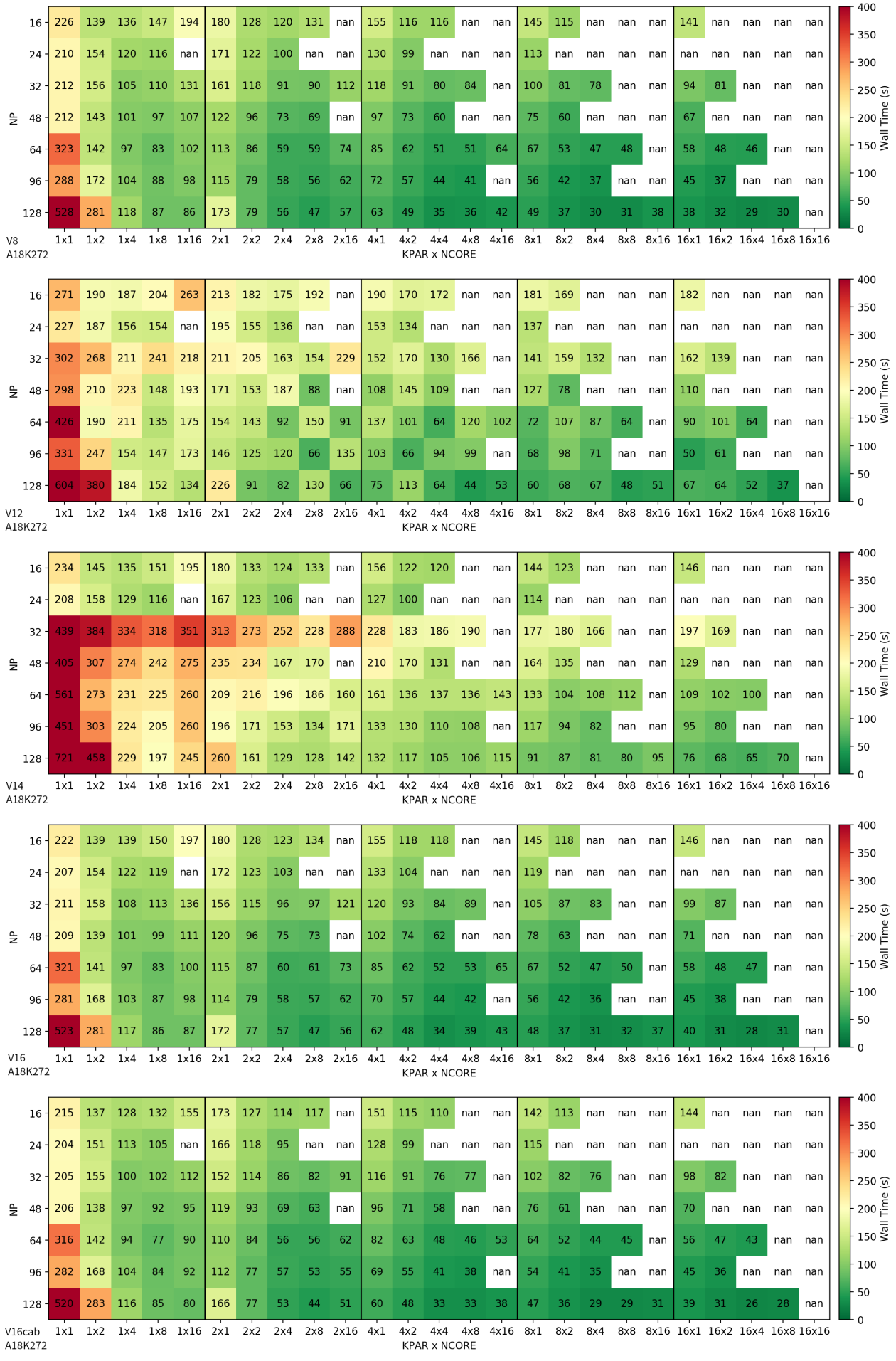

Figure 7.9: 多种编译选项与数学库测试结果

References

[1] http://www.hector.ac.uk/support/documentation/software/vasp/Ncore_and_npar_summary.pdf

[2] https://cms.mpi.univie.ac.at/vasp/vasp/Parallelisation_NPAR_NCORE_LPLANE_KPAR_tag.html

[3] https://www.nsc.liu.se/~pla/blog/2015/01/12/vasp-how-many-cores/

[4] https://www.slideshare.net/jmskelton/vaspgpu-on-balena-usage-and-some-benchmarks