Benchmarks for VASP Parallel CPU Performance on Taiyi Supercomputer of SUSTech

Mingqi Yan, Kebin Sun, and Jia-Wei Mei

1 Introduction

We have run a collection of VASP cases to investigate the parallel efficiency as a function of KPAR and NCORE on Taiyi supercomputer. This note summaries the results to allow users to select the optimal configuration for running their own VASP calculations.

Before starting, some terminology and facts should be addressed.

Hardwares: Each computation node are equipped with 2 Intel© Xeon™ Gold 6148 CPU with 20 cores.

NCORE: NCORE determines the number of computation cores that work on an individual orbital. You can think it as how many cores work together to solve one equation. Therefore, if you set NCORE=20, one CPU is fully used to solve one equation.

KPAR: KPAR determines the number of k-points that are to be treated in parallel which means all the k-points are distributed over KPAR groups of computation cores. For example, if you have 256 different k-points to be calculated, and you set KPAR=4, then each group calculates 64 different k-points.

2 Benchmarks

In this note, a paramagnetic CrN was simulated using a 2 by 2 by 2 supercell with 32 Cr atoms and 32 N atoms. The disorder of magnetic moments was guaranted using the function of constraint magnetic moment. 105 cycles are needed to obtain a convergence of EDIFF=10^{-6} in our own tests. However, only 50 cycles are executed to ensure a fair comparison. VASP 5.4.4 and GGA for exchange-correlation effects were used to perform these calculations.

These benchmark tests can be divided into two rounds. In round 1, we try to find out the best NCORE settings under different KPARs. In round 2, by setting NCORE=20, different KPARs are tested using both $4 \times 4 \times 4$ k-points and $8 \times 8 \times 8$ k-points.

3 Round 1, try to find out the best NCORE settings

In round 1, different KPARs were set from 1 to 6, and NCOREs were set for all the positive factors of 40. When submitting the job, the number of cores is set as 40 × KPAR.

The results are summarized in the Table 1.

Table 1: The elapsed time of different KPAR and NCORE settings

Time(min)	KPAR=1	KPAR=2	KPAR=3	KPAR=4	KPAR=5	KPAR=6
NCORE=1	246.033	127	86.2333	66.0833	55.55	51.5333
NCORE=2	193.917	101.25	70.6667	52.9333	44.6667	36.3667
NCORE=4	136.483	69.1667	63.1167	37.7167	32.65	25.3
NCORE=5	129.317	66.0167	47.5667	34.9	29.5333	23.8833
NCORE=8	121.75	66	45.0833	33.95	28.2333	22.65
NCORE=10	119.4	61.7333	52.1333	32.9	28.3167	21.4167
NCORE=20	115.817	70.3833	42.1	31.8333	26.55	21.4333
NCORE=40	120.433	64.4167	44.65	32.25	28.05	22.7667

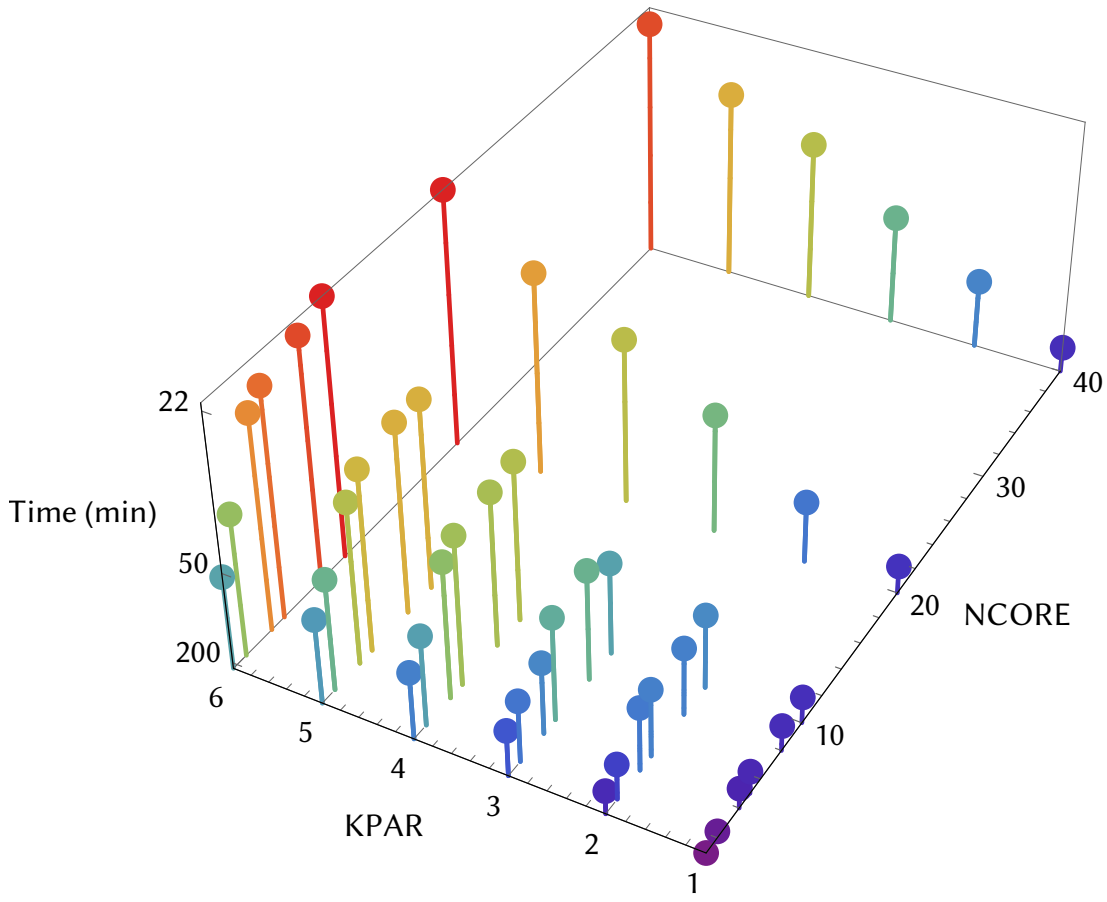

Figure 1: The time elapsed using different `KPAR` and `NCORE` settings

In Table 1, for each `KPAR`, the quickest one is highlighted by given an organ background. We can also plot the above results in a 3d space as shown in Figure 1. In Figue 1, the quicker the case is, the higher it is in the z direction.

From the above results, we can conclude that `NCORE=20` would be the best setting on Taiyi supercomputer since it is just the number of physics cores of each CPU. The hint may lie in that the data transferring within a CPU is very fast. In the next round, we'd like to find out how fast we can achieve if we increase `KPAR` while keeping `NCORE=20`.

RECOMMENDATION: We recommend `NCORE=20` in the general case on `Taiyi` Cluster.

4 Round 2, How fast can we achieve using k-points parallel?

In this round, both $4 \times 4 \times 4$ k mesh and $8 \times 8 \times 8$ k mesh were used to test the limit. The results are shown in Figure 2 and Figure 3.

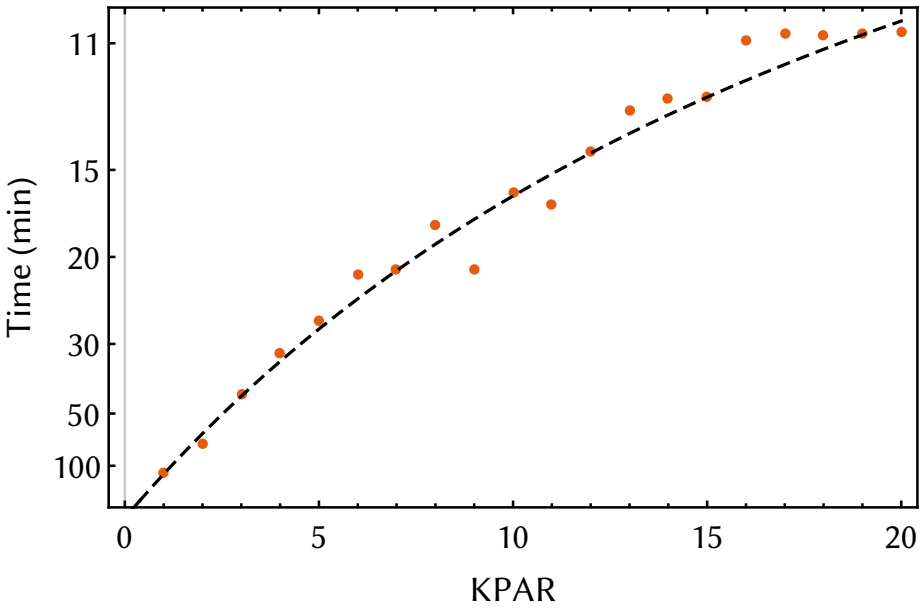

Figure 2: The time elapsed by a $4 \times 4 \times 4$ k mesh using `NCORE=20`

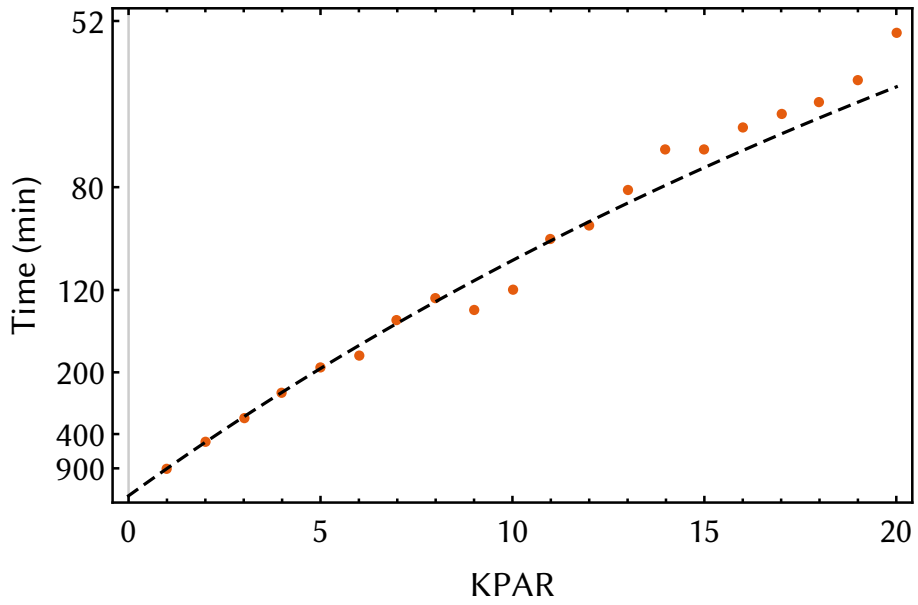

Figure 3: The time elapsed by a $8 \times 8 \times 8$ k mesh using `NCORE=20`

In Figure 2 and Figure 3, the orange dots denote the time consumed by that case. Note that the y axis is under reciprocal scaling to check the data's linearity. The dashed line is a fitting using function $y = \frac{a}{x} + b$, where x represents `KPAR` here and y is the time.

In Figure 2, the time consumed by `KPAR=16 20` was almost the same, but things do not remain the same if we increase k mesh as shown in Figure 3 where a $8 \times 8 \times 8$ k mesh were used. This can be understood by the nature of k-points parallel. k-points parallel will wait for the slowest machines to start a new iteration. In the $4 \times 4 \times 4$ k mesh case, if we choose `KPAR=16`, then 16 groups of computer will work together to calculate these 64 k-points and every group will calculate 4 k-points. However, between `KPAR=17` and `KPAR=20`, there will always some groups needed to calculate 4 k-points which means those groups only need to calculate 3 k-points should wait those slower groups to star a new cycle. We believe that this is the reason why time is almost the same for the last few points in Figure 2.

In figure 3, k-points were increased by 8 times compared to figure 2 where 256 k-points are considered. If we set `KPAR=16`, then these k-points were distributed to 16 groups of computers and each computer will calculate 16 k-points. If we set `KPAR=20`, each computer will calculate around 13 k-points. You can see that the number of k-points calculated by each group of computer is decreased by increasing the `KPAR` parameter.

We have also compared the time ratio between $8 \times 8 \times 8$ k mesh and $4 \times 4 \times 4$ k mesh as shown in Figure 4.

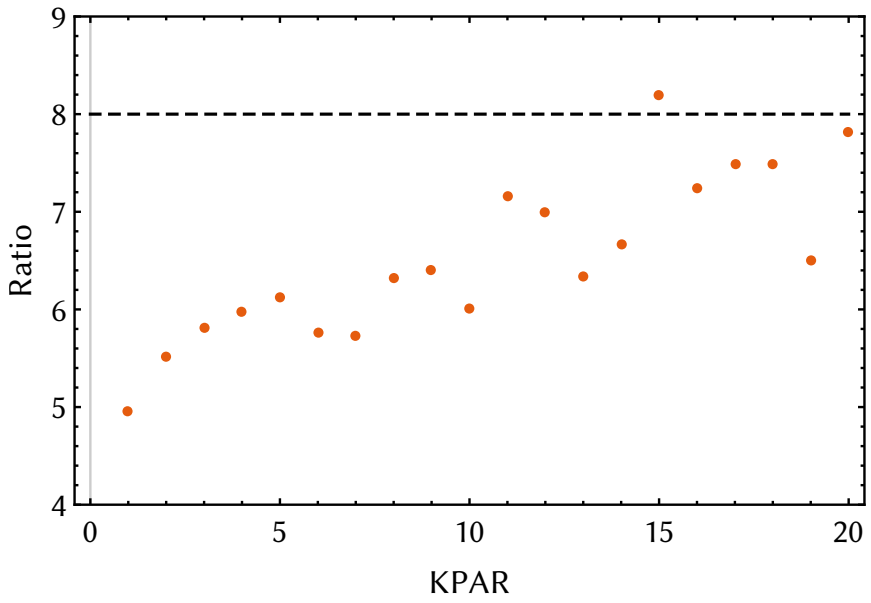

Figure 4: The time elapsed by a $8 \times 8 \times 8$ k mesh using `NCORE=20`

In figure 4, the theoretical value 8 was delineated by a dashed line. We can see that for much of the cases tested, the time consumed was less than the theoretical value. This means that using `KPAR` parameter in your own calculation may get more acceleration than your expectation.

RECOMMENDATION: For small tasks, `KPAR=1` is already OK. For medium tasks, `KPAR <= 15`. For large tasks, the larger `KPAR`, the better.

5 Acknowledgement

We thank Jing Fan for useful discussions.

6 Input files

The list of input files used in this benchmark is as follows.

File 1: The `INCAR` file

```
SYSTEM = paraCrN
ISTART     =      0
ICHARG     =      2
KPAR =
NCORE =
PREC       =    high
ADDGRID    = .TRUE.
NWRITE = 2
LCHARG     = .TRUE.
LWAVE      = .TRUE.

#Electronic Relaxation
ENCUT      = 500
EDIFF      = 1E-08 #    stopping-criterion for ELM
LORBIT     = 11
LREAL      = F
ISYM       = 0
NELMIN     = 10
NELM       = 50
LSORBIT    = .TRUE.
GGA_COMPAT = .FALSE.
```

```
MAXMIX = 60
LREAL= Auto

LWAVE = .FALSE.

#DOS
ISMEAR     = 1
SIGMA      = 0.05

#Magnetism
MAGMOM =   0. 0. 1.44 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44
LCONSTRAINED_M = 2
RWIGS      = 1.1 1.1 0.0
LAMBDA     = 10
M_CONSTR = 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44 0. 0. 1.44

LMAXMIX=4
```

File 2: The `KPOINTS` file

```
Automatic mesh
0       ! Automatic sheme
G       ! M/m: MP,  G/g/A/a: Gamma-centered, A/a: Fully automatic
4 4 4
0 0 0 ! shift ! M with no shift = G + 1/2 1/2 1/2 shift
```

File 3: The `POSCAR` file

```
2_by_2_by_2_supercell_CrN
1.0
         8.3030004501          0.0000000000          0.0000000000
         0.0000000000          8.3030004501          0.0000000000
         0.0000000000          0.0000000000          8.3030004501
   Cr    Cr    N
   16    16    32
Direct
      0.750000000          0.250000000          0.000000000
      0.000000000          0.000000000          0.500000000
      0.750000000          0.000000000          0.250000000
      0.750000000          0.500000000          0.250000000
      0.750000000          0.250000000          0.500000000
      0.750000000          0.750000000          0.000000000
      0.250000000          0.250000000          0.000000000
      0.000000000          0.000000000          0.000000000
      0.250000000          0.000000000          0.250000000
      0.250000000          0.000000000          0.750000000
      0.500000000          0.000000000          0.000000000
      0.500000000          0.000000000          0.500000000
      0.250000000          0.500000000          0.750000000
      0.500000000          0.750000000          0.750000000
      0.750000000          0.500000000          0.750000000
      0.000000000          0.500000000          0.000000000
      0.000000000          0.250000000          0.250000000
      0.000000000          0.250000000          0.750000000
      0.000000000          0.500000000          0.500000000
      0.000000000          0.750000000          0.250000000
      0.000000000          0.750000000          0.750000000
      0.250000000          0.250000000          0.500000000
      0.250000000          0.500000000          0.250000000
      0.250000000          0.750000000          0.000000000
      0.250000000          0.750000000          0.500000000
```

```
0.500000000  0.250000000  0.250000000
0.500000000  0.250000000  0.750000000
0.500000000  0.500000000  0.000000000
0.500000000  0.500000000  0.500000000
0.500000000  0.750000000  0.250000000
0.750000000  0.000000000  0.750000000
0.750000000  0.750000000  0.500000000
0.250000000  0.000000000  0.000000000
0.250000000  0.000000000  0.500000000
0.250000000  0.500000000  0.000000000
0.250000000  0.500000000  0.500000000
0.750000000  0.000000000  0.000000000
0.750000000  0.000000000  0.500000000
0.750000000  0.500000000  0.000000000
0.750000000  0.500000000  0.500000000
0.000000000  0.250000000  0.000000000
0.000000000  0.250000000  0.500000000
0.000000000  0.750000000  0.000000000
0.000000000  0.750000000  0.500000000
0.500000000  0.250000000  0.000000000
0.500000000  0.250000000  0.500000000
0.500000000  0.750000000  0.000000000
0.500000000  0.750000000  0.500000000
0.000000000  0.000000000  0.250000000
0.000000000  0.000000000  0.750000000
0.000000000  0.500000000  0.250000000
0.000000000  0.500000000  0.750000000
0.500000000  0.000000000  0.250000000
0.500000000  0.000000000  0.750000000
0.500000000  0.500000000  0.250000000
0.500000000  0.500000000  0.750000000
0.250000000  0.250000000  0.250000000
0.250000000  0.250000000  0.750000000
0.250000000  0.750000000  0.250000000
0.250000000  0.750000000  0.750000000
0.750000000  0.250000000  0.250000000
0.750000000  0.250000000  0.750000000
0.750000000  0.750000000  0.250000000
0.750000000  0.750000000  0.750000000
```

File 4: The VASP submit scipt

```
#!/bin/bash
#BSUB -q medium
#BSUB -J job name
#BSUB -n 40 * KPAR
#BSUB -o out_%J.out
#BSUB -e error_%J.err
#BSUB -R "span[ptile=40]"

module load intel/2018.4
module load mpi/intel/2018.4
module load vasp/5.4.4-vtst
ulimit -s unlimited
mpirun -np $LSB_DJOB_NUMPROC vasp_ncl
```