

軟體安裝、效能測試及調校

國立臺灣師範學物理學系 陳俊明

chunming@ntnu.edu.tw

應用軟體

- VASP
- Quantum Espresso
- LAMMPS
- MATLAB
- Gaussian

應用軟體安裝-Python3

- 安裝相關套件

```
[root@master ~]# yum groupinstall "Development tools"  
[root@master ~]# yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-  
devel tk-devel
```

- 下載python3.12

```
[root@master ~]# wget https://www.python.org/ftp/python/3.12.5/Python-3.12.5.tgz  
[root@master ~]# tar zxvf Python-3.12.5.tgz  
[root@master ~]# cd Python-3.12.5  
[root@master Python-3.12.5]# ./configure --prefix=/opt/python3.12
```

- 使用python3.12

```
[root@master ~]# export PATH=/opt/python3.12/bin:$PATH  
[root@master ~]# export LD_LIBRARY_PATH=/opt/python3.12/lib:$LD_LIBRARY_PATH
```

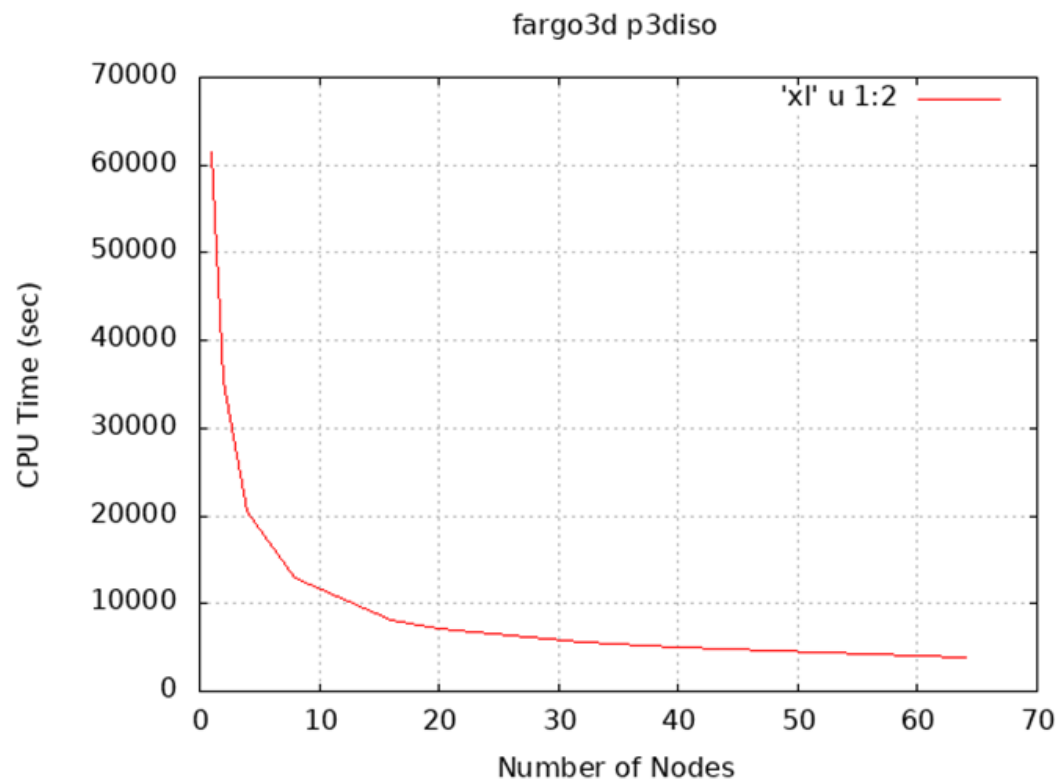
效能測試軟體

- HPL : <https://www.netlib.org/benchmark/hpl/>
- HPCG : <https://www.hpcg-benchmark.org/>
- HPC Challenge : <http://icl.cs.utk.edu/hpcc/>
- STREAM : <https://www.cs.virginia.edu/stream/>
- SPEC : <https://www.spec.org/>
- Phoronix Test Suite : <https://www.phoronix-test-suite.com/>
- MGBench : <https://github.com/tbennun/mgbench>

程式的 Benchmark

使用已知的計算實例，在不同數量的節點跟計算核心進行測試，並將測試的結果繪圖。

- 測試過程中，同時檢查每一計算步驟的結果
- 檢查最後結果的一致性
- 檢查收斂的過程
- 檢查計算的時間
- 根據“節點數”以及“計算核心”數跟計算時間繪圖



HPL 安裝

- 安裝OpenBLAS

```
[root@master ~]# yum install openblas openblas-devel
```

- 前往 <https://www.netlib.org/benchmark/hpl/> 下載，並且解壓縮 hpl-2.3.tar.gz

```
[user1@master ~]# wget https://www.netlib.org/benchmark/hpl/hpl-2.3.tar.gz
[user1@master ~]# tar zxvf hpl-2.3.tar.gz
[user1@master ~]# cd hpl-2.3
[user1@master hpl-2.3]# export PATH=/opt/openmpi/4.1.6_gnu_8.5/bin:$PATH
[user1@master hpl-2.3]# export LD_LIBRARY_PATH= /opt/openmpi/4.1.6_gnu_8.5/lib:$ LD_LIBRARY_PATH
[user1@master hpl-2.3]# ./configure --prefix=/home/user1/hpl CC=mpicc
[user1@master hpl-2.3]# make -j < number of processor > && make install
```

HPL 安裝

- 手動安裝方式，編輯 **Make.XXX** 檔案，進行編譯安裝

```
[user1@master ~]# source /opt/intel/oneapi/setvars.sh intel64
```

```
[user1@master ~]# cd hpl-2.3
```

```
[user1@master hpl-2.3]# cp setup/Make.Linux_Intel64
```

```
[user1@master hpl-2.3]# vi Make.Linux_intel64
```

```
TOPdir    = $(HOME)/hpl-2.3
```

```
CC        = mpiicx
```

```
OMP_DEFS = -qopenmp
```

```
CCFLAGS = $(HPL_DEFS) -O3 -w -ansi-alias -static-intel -z noexecstack -z relro -z now -Wall
```

```
[user1@master hpl-2.3]# make arch= Linux_intel64
```

```
[user1@master hpl-2.3]# cd bin/Linux_intel64
```

HPL 設定與調教

- FLOPS : floating point operations per second.
- R_{peak} : the theoretical peak performance GFLOPS for the machine.
- R_{max} : the performance in GFLOPS for the largest problem run on a machine.

$R_{peak} = (\text{CPU speed in GHz}) \times (\text{number of CPU cores}) \times (\text{CPU instruction per cycle}) \times (\text{number of CPUs per node})$

Intel Gold 6130, 16 cores, 2 CPU			
ISA	Base Frequency	Turbo Frequency	# of flops per cycle
AVX512	2.1 GHz	3.2 GHz	32

$$R_{peak} = 16 \times 3.2 \times 32 \times 2 = 3276.8 \text{ GFLOPS}$$

HPL 設定與調教

Microarchitecture ↕	ISA ↕	FP64 ↕	FP32 ↕	FP16 ↕
Intel CPU				
Intel Netburst Pentium 4 (Prescott, Cedar Mill) Intel Netburst Pentium D (Smithfield, Presler) Intel P6 Core (Yonah)	SSE3 (64-bit)	2	4	?
Intel Core (Merom, Penryn) Intel Nehalem ^[10] (Nehalem, Westmere)	SSSE3 (128-bit) SSE4 (128-bit)	4	8	?
Intel Atom (Bonnell, Saltwell, Silvermont and Goldmont)	SSE3 (128-bit)	2	4	?
Intel Sandy Bridge (Sandy Bridge, Ivy Bridge)	AVX (256-bit)	8	16	0
Intel Haswell ^[10] (Haswell, Devil's Canyon, Broadwell) Intel Skylake (Skylake, Kaby Lake, Coffee Lake, Comet Lake, Whiskey Lake, Amber Lake)	AVX2 & FMA (256-bit)	16	32	0
Intel Xeon Phi (Knights Corner)	IMCI (512-bit)	16	32	0
Intel Skylake-X (Skylake-X, Cascade Lake) Intel Xeon Phi (Knights Landing, Knights Mill) Intel Ice Lake, Tiger Lake and Rocket Lake	AVX-512 & FMA (512-bit)	32	64	0
AMD CPU				
AMD Bobcat	AMD64 (64-bit)	2	4	0
AMD Jaguar AMD Puma	AVX (128-bit)	4	8	0
AMD K10	SSE4/4a (128-bit)	4	8	0
AMD Bulldozer ^[10] (Piledriver, Steamroller, Excavator)	AVX (128-bit) Bulldozer-Steamroller AVX2 (128-bit) Excavator FMA3 (Bulldozer) ^[11] FMA3/4 (Piledriver-Excavator)	4	8	0
AMD Zen (Ryzen 1000 series, Threadripper 1000 series, Epyc Naples) AMD Zen+ ^{[10][12][13][14]} (Ryzen 2000 series, Threadripper 2000 series)	AVX2 & FMA (128-bit, 256-bit decoding) ^[15]	8	16	0
AMD Zen 2 ^[16] (Ryzen 3000 series, Threadripper 3000 series, Epyc Rome)) AMD Zen 3 (Ryzen 5000 series, Epyc Milan)	AVX2 & FMA (256-bit)	16	32	0

圖表來源：<https://en.wikipedia.org/wiki/FLOPS>

HPL 設定與調教

HPLinpack benchmark input file

HPL.dat

Innovative Computing Laboratory, University of Tennessee

HPL.out output file name (if any)

6 device out (6=stdout,7=stderr,file)

1 # of problems sizes (N)

29184 Ns

1 # of NBs

192 NBs

0 PMAP process mapping (0=Row-,1=Column-major)

1 # of process grids (P x Q)

2 Ps

4 Qs

16.0 threshold

1 # of panel fact

2 PFACTs (0=left, 1=Crout, 2=Right)

1 # of recursive stopping criterium

4 NBMINS (>= 1)

1 # of panels in recursion

2 NDIVs

1 # of recursive panel fact.

1 RFACTs (0=left, 1=Crout, 2=Right)

1 # of broadcast

1 BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)

1 # of lookahead depth

- $$N = \sqrt{\frac{\text{total mem size(byte)}}{8}} \times 0.8$$
- NB 值：在網格 (grid) 內的區塊 (block) 大小，經過實際測試來得到最佳，值通常小於384。
建議測試：96, 104, 112, 128, ..., 196, 256, 384
- P×Q表示二维处理器網格(grid)，P表示水平方向處理器個數，Q表示垂直方向處理器個數
- P、Q 兩個相乘等於預計使用核心數，建議 $P \leq Q$ 、 $P = 2^n$

HPL 設定與調教

How do I tune my HPL.dat file?

N 建議值：

$$N = \sqrt{\frac{\text{total mem size(byte)}}{8}} \times 0.8$$

$$N = \sqrt{\frac{64GB \times 1024 \times 1024 \times 1024}{8}} \times 0.8 \cong 82897$$

最佳化 N 值：通常把 N 值與 NB 值標齊對正，就可以找出最價值。以上面 64GB 記憶體為例，假設要測試的 NB 是 224，以 N 值除以 NB 值 (82,897 / 224 ≈ 370) 後乘以 NB 值 (370 * 224 = 82,880)，就可以找出最佳化 N 值

HPL 執行

- 編輯好 HPL.dat 檔案後，直接執行

```
[user1@master ~]$ cd hpl/bin  
[user1@master bin]$ vi HPL.dat  
[user1@master bin]$ mpirun -np <number of processor> ./xhpl
```

```
=====
T/V              N    NB    P    Q              Time              Gflops
-----
WR11C2R4         20160  192    2    2              457.72              1.1935e+01
HPL_pdgesv() start time Fri Aug 11 23:35:35 2023

HPL_pdgesv() end time   Fri Aug 11 23:43:14 2023
```

HPCG 安裝

- 前往<https://www.hpcg-benchmark.org/> 下載，並且解壓縮 hpcg-3.1.tar.gz

```
[user1@master ~]$ wget http://www.hpcg-benchmark.org/downloads/hpcg-3.1.tar.gz
[user1@master ~]$ tar zxvf hpcg-3.1.tar.gz
[user1@master ~]$ mkdir ~/hpcg
[user1@master ~]$ source /opt/intel/oneapi/setvars.sh intel64
[user1@master ~]$ cd ~/hpcg
[user1@master hpcg]$ /home/user1/hpcg-3.1/configure Linux_MPI
[user1@master hpcg]$ vim setup/Linux_MPI
CXX          = mpicxx
CXXFLASG = $(HPCG_DEFS) -O3 -ffast-math -ftree-vectorize
[user1@master hpcg]$ make
```

HPCG 設定及執行

- 設定 **HPCG.DAT** 檔案

###：此為矩陣大小，該數字為 8 的倍數。數字越大使用記憶體越多，建議調整到使用記憶體的60%~80%

###：運算時間，單位為秒，要跑 30 分鐘以上才能測出效能，建議值 3600

- 編輯好 HPCG.DAT 檔案後，直接執行

```
[user1@master bin]$ mpirun -np <number of processor> ./xhpcg
```

系統調校技巧

- BIOS 關閉 Hyper Threading
- BIOS 的 Power 設定為 Maximum Performance
- 如果系統碟是使用 HDD 設定 RAID 0，或是改使用 SSD / NVMe 不設定 RAID
- 避免將記憶體完全吃滿，關閉 swap 或避免使用 swap
- 確保使用者每次執行完程式能清除記憶體
- 確保節點對節點能跑接近網路理論值

程式錯誤狀況排除

- 仔細的看錯誤訊息
- 是否能夠重現程式的錯誤？
- 透過指令能夠幫助縮小範圍：
 - 確認環境變數：`echo`, `ifort`, `icc`, `ldd`, `which`, `ompi_info`
 - 確認記憶體的使用量：`free -m`, `vmstat`
 - 確認 I/O 的狀態：`netstat -an`, `iostat`, `sar -n DEV 1 10`, `df -hT /tmp`
 - 確認計算時的負載：`uptime`, `top`
 - 確認 Kernel 的訊息：`dmesg | tail -n 100 > ~/klog`
- 使用者回報錯誤時，一定要要求提供至少4個訊息 (4W)
 - Who：哪一個帳號？
 - Where：哪一台機器？哪一個節點？工作路徑？程式路徑？
 - When：什麼時候發生？
 - What：從輸出檔看到什麼錯誤訊息或是沒有錯誤訊息？