高性能集群搭建指南（四） - 性能测试及功耗控制

2017-05-15 22:05:49

## 一. 计算性能测试

### 1. 理论峰值的计算

理论峰值仅代表着CPU的极限运算性能，只是一个参考。在实际中，由于CPU频率的调整、内存等其他硬件限制，和并行程序的并行效率等软件层面的限制；测试出来的计算性能会与理论峰值有些差距。

#### 1.1 支持 AVX2 指令集的 Intel64 CPU/MIC 的计算

主要有以下系列CPU：Haswell(v3); Broadwel(v4); Skylake (v5); Knights Corner(KNC)  
（ASC17用到的是 E5-2680v4）

此类CPU每个核心有16个向量化单元，即每秒可进行16次浮点运算。

DP FLOPS = cores/CPU \* CPU hz \* 16

如E5-2680v4为： 2（每个服务器有两块CPU） \* 14(每个CPU有14核心) \* 2.5GHz \* 16 = 1120 GFlops

#### 1.2 支持 AVX512 指令集的 CPU/MIC 的计算

目前支持 AVX512 指令集的只有 Knights Landing (KNL)

此类CPU每个核心有32个向量化单元，每秒可进行32次浮点运算。

DP FLOPS = cores/CPU \* CPU hz \* 16

如PHI-7210为：64（每块KNL有64核心） \* 1.3GHz \* 32 = 2662.4 GFlops

### 2. HPL(Linpack)

#### 2.1 HPL和Linpack介绍

Linpack现在在国际上已经成为最流行的用于测试高性能计算机系统浮点性能的benchmark。通过利用高性能计算机，用高斯消元法求解N元一次稠密线性代数方程组的测试，评价高性能计算机的浮点性能。

Linpack测试包括三类，Linpack100、Linpack1000和HPL。Linpack100求解规模为100阶的稠密线性代数方程组，它只允许采用编译优化选项进行优化，不得更改代码，甚至代码中的注释也不得修改。Linpack1000要求求解规模为1000阶的线性代数方程组，达到指定的精度要求，可以在不改变计算量的前提下做算法和代码上做优化。HPL即High Performance Linpack，也叫高度并行计算基准测试，它对数组大小N没有限制，求解问题的规模可以改变，除基本算法（计算量）不可改变外，可以采用其它任何优化方法。前两种测试运行规模较小，已不是很适合现代计算机的发展，因此现在使用较多的测试标准为HPL，而且阶次N也是linpack测试必须指明的参数。

HPL是针对现代并行计算机提出的测试方式。用户在不修改任意测试程序的基础上，可以调节问题规模大小N(矩阵大小)、使用到的CPU数目、使用各种优化方法等来执行该测试程序，以获取最佳的性能。HPL采用高斯消元法求解线性方程组。当求解问题规模为N时，浮点运算次数为(2/3\*N^3－2\*N^2)。因此，只要给出问题规模N，测得系统计算时间T，峰值=计算量(2/3\*N^3－2\*N^2)/计算时间T，测试结果以浮点运算每秒（Flops）给出

#### 2.2 HPL源码的编译

这里以官方版本的 HP-L2.2 为例。

选择合适的 Makefile，官方已经为不同架构、OS和编译器写好了一些，位于 TOPdir/setup 目录中。

$ cp setup/Make.Linux\_Intel64 ./

修改 Make.Linux\_Intel64，指定 TOPdir 位置

TOPdir = /where/you/install/hpl-2.2

编译

$ make arch=Linux\_Intel64 #也可以在 Makefile 中指定 arch;然后直接 make  
  
#编译正常后，生成了 bin 目录  
$ ls  
bin HISTORY INSTALL Make.Linux\_Intel64 man src TUNING  
BUGS hpl lib makes README testing www  
COPYRIGHT include Makefile Make.top setup TODO  
  
$ ls bin/Linux\_Intel64/  
HPL.dat xhpl

#### 2.3 HPL 在Intel64平台的测试

Intel MKL 库中带有优化过的 benchmark  
一般位于 /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mkl/benchmarks 中。

cd /opt/intel/compilers\_and\_libraries\_2017.4.196/linux/mkl/benchmarks/mp\_linpack  
vim HPL.dat

修改 HPL.dat

* Ns 在大规模集群上，N\*N\*8 一般为内存的80%，当集群规模较小时，这个比例会低一些。（如在ASC17中，选取的 N 为240000[应该不是最佳参数]，内存占比为67%，最终结果是 7.3TFlops ）  
  如果想快速测试，也可以选择一组小一些的参数。
* NBs 在 E5 及 KNL上测试发现是 192 为最佳参数。

$ vim HPL.dat  
  
HPLinpack benchmark input file  
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)  
50000 Ns  
1 # of NBs  
192 NBs  
1 PMAP process mapping (0=Row-,1=Column-major)  
1 # of process grids (P x Q)  
1 2 Ps  
1 2 Qs  
16.0 threshold  
1 # of panel fact  
2 1 0 PFACTs (0=left, 1=Crout, 2=Right)  
1 # of recursive stopping criterium  
2 NBMINs (= 1)  
1 # of panels in recursion  
2 NDIVs  
1 # of recursive panel fact.  
1 0 2 RFACTs (0=left, 1=Crout, 2=Right)  
1 # of broadcast  
0 BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)  
1 # of lookahead depth  
0 DEPTHs (=0)  
0 SWAP (0=bin-exch,1=long,2=mix)  
1 swapping threshold  
1 L1 in (0=transposed,1=no-transposed) form  
1 U in (0=transposed,1=no-transposed) form  
0 Equilibration (0=no,1=yes)  
8 memory alignment in double ( 0)

运行HPL

$ export OMP\_NUM\_THREADS=28 #总核心数  
$ export MKL\_NUM\_THREADS=28  
$ .xhpl\_intel64\_dynamic  
  
  
--------------------------------------------------------------------------------  
Ax-b\_oo(eps(A\_oox\_oo+b\_oo)N)= 0.0045345 ...... PASSED  
mic3 Column=000384 Fraction=0.005 Kernel= 0.05 Mflops=1433902.37  
mic3 Column=000576 Fraction=0.010 Kernel=820465.80 Mflops=1149605.94  
mic3 Column=000768 Fraction=0.015 Kernel=824814.65 Mflops=1047665.59  
mic3 Column=001152 Fraction=0.020 Kernel=817380.27 Mflops=959001.49  
mic3 Column=001344 Fraction=0.025 Kernel=811729.33 Mflops=935323.50  
mic3 Column=001536 Fraction=0.030 Kernel=853574.35 Mflops=924556.05  
mic3 Column=001920 Fraction=0.035 Kernel=814404.21 Mflops=900944.94  
mic3 Column=002112 Fraction=0.040 Kernel=818573.97 Mflops=893093.53  
mic3 Column=002304 Fraction=0.045 Kernel=835688.62 Mflops=888227.39  
mic3 Column=002688 Fraction=0.050 Kernel=815766.35 Mflops=877614.73  
mic3 Column=002880 Fraction=0.055 Kernel=830918.27 Mflops=874518.04  
mic3 Column=003072 Fraction=0.060 Kernel=829651.65 Mflops=871745.04  
mic3 Column=003264 Fraction=0.065 Kernel=830779.03 Mflops=869381.82  
mic3 Column=003648 Fraction=0.070 Kernel=835668.02 Mflops=865950.94  
mic3 Column=003840 Fraction=0.075 Kernel=793611.73 Mflops=862315.32  
mic3 Column=004032 Fraction=0.080 Kernel=851690.33 Mflops=861843.77  
mic3 Column=004416 Fraction=0.085 Kernel=815254.39 Mflops=857935.27  
mic3 Column=004608 Fraction=0.090 Kernel=839449.17 Mflops=857220.41  
mic3 Column=004800 Fraction=0.095 Kernel=821844.63 Mflops=855887.10  
mic3 Column=005184 Fraction=0.100 Kernel=828691.61 Mflops=854017.78  
mic3 Column=005376 Fraction=0.105 Kernel=808901.12 Mflops=852502.22  
mic3 Column=005568 Fraction=0.110 Kernel=824008.89 Mflops=851600.16  
mic3 Column=005760 Fraction=0.115 Kernel=819855.29 Mflops=850629.38  
mic3 Column=006144 Fraction=0.120 Kernel=830269.63 Mflops=849483.98  
mic3 Column=006336 Fraction=0.125 Kernel=813456.77 Mflops=848491.23  
mic3 Column=006528 Fraction=0.130 Kernel=830828.00 Mflops=848031.17  
mic3 Column=006912 Fraction=0.135 Kernel=824199.60 Mflops=846857.05  
mic3 Column=007104 Fraction=0.140 Kernel=842360.80 Mflops=846752.60  
mic3 Column=007296 Fraction=0.145 Kernel=816742.67 Mflops=846056.54  
mic3 Column=007680 Fraction=0.150 Kernel=820905.53 Mflops=844961.16  
mic3 Column=007872 Fraction=0.155 Kernel=825957.16 Mflops=844563.90  
mic3 Column=008064 Fraction=0.160 Kernel=847857.36 Mflops=844629.05  
mic3 Column=008256 Fraction=0.165 Kernel=823622.84 Mflops=844213.59  
mic3 Column=008640 Fraction=0.170 Kernel=826474.17 Mflops=843549.73  
mic3 Column=008832 Fraction=0.175 Kernel=805635.33 Mflops=842845.33  
mic3 Column=009024 Fraction=0.180 Kernel=826354.40 Mflops=842554.62  
mic3 Column=009408 Fraction=0.185 Kernel=823713.08 Mflops=841919.66  
mic3 Column=009600 Fraction=0.190 Kernel=843652.87 Mflops=841947.32  
mic3 Column=009792 Fraction=0.195 Kernel=816310.42 Mflops=841535.15  
mic3 Column=010176 Fraction=0.200 Kernel=827674.46 Mflops=841114.98  
mic3 Column=010368 Fraction=0.205 Kernel=820677.75 Mflops=840811.63  
mic3 Column=010560 Fraction=0.210 Kernel=819496.91 Mflops=840502.40  
mic3 Column=010752 Fraction=0.215 Kernel=837126.18 Mflops=840455.57  
mic3 Column=011136 Fraction=0.220 Kernel=812985.65 Mflops=839703.56  
mic3 Column=011328 Fraction=0.225 Kernel=824705.22 Mflops=839506.78  
mic3 Column=011520 Fraction=0.230 Kernel=821189.71 Mflops=839270.88  
mic3 Column=011904 Fraction=0.235 Kernel=817064.62 Mflops=838718.66  
mic3 Column=012096 Fraction=0.240 Kernel=835432.24 Mflops=838679.76  
mic3 Column=012288 Fraction=0.245 Kernel=802802.08 Mflops=838247.56  
mic3 Column=012672 Fraction=0.250 Kernel=828727.27 Mflops=838033.65  
mic3 Column=012864 Fraction=0.255 Kernel=843865.81 Mflops=838096.33  
mic3 Column=013056 Fraction=0.260 Kernel=787007.22 Mflops=837520.21  
mic3 Column=013440 Fraction=0.265 Kernel=831731.15 Mflops=837401.09  
mic3 Column=013632 Fraction=0.270 Kernel=828051.40 Mflops=837306.93  
mic3 Column=013824 Fraction=0.275 Kernel=849740.35 Mflops=837426.51  
mic3 Column=014016 Fraction=0.280 Kernel=812580.92 Mflops=837181.71  
mic3 Column=014400 Fraction=0.285 Kernel=819944.15 Mflops=836856.68  
mic3 Column=014592 Fraction=0.290 Kernel=799944.32 Mflops=836509.00  
mic3 Column=014784 Fraction=0.295 Kernel=845354.79 Mflops=836586.31  
mic3 Column=015168 Fraction=0.300 Kernel=816536.46 Mflops=836235.60  
mic3 Column=015360 Fraction=0.305 Kernel=803021.79 Mflops=835947.59  
mic3 Column=015552 Fraction=0.310 Kernel=854329.19 Mflops=836094.58  
mic3 Column=015936 Fraction=0.315 Kernel=807391.66 Mflops=835624.74  
mic3 Column=016128 Fraction=0.320 Kernel=812608.68 Mflops=835442.17  
mic3 Column=016320 Fraction=0.325 Kernel=806923.12 Mflops=835218.68  
mic3 Column=016512 Fraction=0.330 Kernel=832295.28 Mflops=835196.89  
mic3 Column=016896 Fraction=0.335 Kernel=815558.38 Mflops=834907.49  
mic3 Column=017088 Fraction=0.340 Kernel=821441.93 Mflops=834811.37  
mic3 Column=017280 Fraction=0.345 Kernel=812260.62 Mflops=834651.61  
mic3 Column=017664 Fraction=0.350 Kernel=823159.13 Mflops=834495.87  
mic3 Column=017856 Fraction=0.355 Kernel=801996.42 Mflops=834275.36  
mic3 Column=018048 Fraction=0.360 Kernel=831701.28 Mflops=834258.82  
mic3 Column=018432 Fraction=0.365 Kernel=827402.18 Mflops=834172.95  
mic3 Column=018624 Fraction=0.370 Kernel=820058.34 Mflops=834085.92  
mic3 Column=018816 Fraction=0.375 Kernel=810101.01 Mflops=833938.94  
mic3 Column=019008 Fraction=0.380 Kernel=836988.59 Mflops=833956.70  
mic3 Column=019392 Fraction=0.385 Kernel=831515.76 Mflops=833928.93  
mic3 Column=019584 Fraction=0.390 Kernel=796233.11 Mflops=833710.41  
mic3 Column=019776 Fraction=0.395 Kernel=817330.62 Mflops=833619.58  
mic3 Column=020160 Fraction=0.400 Kernel=822896.69 Mflops=833504.93  
mic3 Column=020352 Fraction=0.405 Kernel=866136.11 Mflops=833666.70  
mic3 Column=020544 Fraction=0.410 Kernel=767393.58 Mflops=833302.66  
mic3 Column=020928 Fraction=0.415 Kernel=831902.60 Mflops=833288.88  
mic3 Column=021120 Fraction=0.420 Kernel=811938.47 Mflops=833183.88  
mic3 Column=021312 Fraction=0.425 Kernel=838562.29 Mflops=833209.04  
mic3 Column=021504 Fraction=0.430 Kernel=798961.18 Mflops=833043.96  
mic3 Column=021888 Fraction=0.435 Kernel=840818.91 Mflops=833113.13  
mic3 Column=022080 Fraction=0.440 Kernel=783361.28 Mflops=832881.48  
mic3 Column=022272 Fraction=0.445 Kernel=836998.56 Mflops=832899.10  
mic3 Column=022656 Fraction=0.450 Kernel=799428.90 Mflops=832607.87  
mic3 Column=022848 Fraction=0.455 Kernel=824515.17 Mflops=832574.58  
mic3 Column=023040 Fraction=0.460 Kernel=820378.76 Mflops=832525.07  
mic3 Column=023424 Fraction=0.465 Kernel=827059.37 Mflops=832482.32  
mic3 Column=023616 Fraction=0.470 Kernel=835901.92 Mflops=832495.22  
mic3 Column=023808 Fraction=0.475 Kernel=794697.95 Mflops=832348.00  
mic3 Column=024192 Fraction=0.480 Kernel=806653.30 Mflops=832156.52  
mic3 Column=024384 Fraction=0.485 Kernel=847053.68 Mflops=832208.04  
mic3 Column=024576 Fraction=0.490 Kernel=778749.11 Mflops=832010.68  
mic3 Column=024768 Fraction=0.495 Kernel=841170.11 Mflops=832041.42  
mic3 Column=025920 Fraction=0.515 Kernel=821099.19 Mflops=831831.63  
mic3 Column=026880 Fraction=0.535 Kernel=810236.56 Mflops=831515.99  
mic3 Column=027840 Fraction=0.555 Kernel=818209.96 Mflops=831341.08  
mic3 Column=028800 Fraction=0.575 Kernel=813029.56 Mflops=831121.60  
mic3 Column=029760 Fraction=0.595 Kernel=804420.63 Mflops=830829.39  
mic3 Column=030912 Fraction=0.615 Kernel=806813.10 Mflops=830549.43  
mic3 Column=031872 Fraction=0.635 Kernel=807366.11 Mflops=830349.65  
mic3 Column=032832 Fraction=0.655 Kernel=806102.31 Mflops=830162.86  
mic3 Column=033792 Fraction=0.675 Kernel=792432.87 Mflops=829900.31  
mic3 Column=034752 Fraction=0.695 Kernel=798952.39 Mflops=829711.73  
mic3 Column=039936 Fraction=0.795 Kernel=787867.30 Mflops=828814.92  
mic3 Column=044928 Fraction=0.895 Kernel=700457.74 Mflops=827735.23  
mic3 Column=049920 Fraction=0.995 Kernel=363372.03 Mflops=826632.56  
================================================================================  
TV N NB P Q Time Gflops  
--------------------------------------------------------------------------------  
WC00C2R2 50000 192 1 1 101.00 8.25147e+02  
HPL\_pdgesv() start time Tue May 16 224657 2017  
  
HPL\_pdgesv() end time Tue May 16 224838 2017  
  
--------------------------------------------------------------------------------  
Ax-b\_oo(eps(A\_oox\_oo+b\_oo)N)= 0.0038828 ...... PASSED  
================================================================================

如上，简单的结果为 825.147GFlops 计算性能正常。

#### 2.4 HPL在GPU上的测试

* NVIDIA 开发者社区提供了开源的Fermi版本的HPL:  
  <https://developer.nvidia.com/rdp/assets/cuda-accelerated-linpack-linux64>  
  由于显存占用问题，在Pascal架构下跑这个版本的HPL只能达到理论峰值的一半左右。
* 在Pascal架构下，NVIDIA提供了HPL的二进制版本，直接运行即可。不出意外的话，之后的Volta架构也会有NVIDIA官方的HPL版本。 在运行时可能遇到动态链接库没链上的问题，把它重新链上就OK。

## 二、 网络测试

见 高性能集群搭建指南（二） - 系统配置 中 以太网带宽测试 及 IB读写带宽测试 。

br

## 三、 内存测试

通过 Stream 测试内存带宽实际性能，Intel Parallel Stido 带有优化过的 Stream 可执行文件，直接执行即可

$ cd /opt/intel/clck2017.2.019/provider/share/stream/intel64  
$ .stream  
  
-------------------------------------------------------------  
STREAM version $Revision 5.10 $  
-------------------------------------------------------------  
This system uses 8 bytes per array element.  
-------------------------------------------------------------  
Array size = 64000000 (elements), Offset = 0 (elements)  
Memory per array = 488.3 MiB (= 0.5 GiB).  
Total memory required = 1464.8 MiB (= 1.4 GiB).  
Each kernel will be executed 10 times.  
 The best time for each kernel (excluding the first iteration)  
 will be used to compute the reported bandwidth.  
-------------------------------------------------------------  
Number of Threads requested = 24  
Number of Threads counted = 24  
-------------------------------------------------------------  
Your clock granularityprecision appears to be 1 microseconds.  
Each test below will take on the order of 23663 microseconds.  
 (= 23663 clock ticks)  
Increase the size of the arrays if this shows that  
you are not getting at least 20 clock ticks per test.  
-------------------------------------------------------------  
WARNING -- The above is only a rough guideline.  
For best results, please be sure you know the  
precision of your system timer.  
-------------------------------------------------------------  
Function Best Rate MBs Avg time Min time Max time  
Copy 46400.0 0.027587 0.022069 0.037249  
Scale 46989.9 0.026737 0.021792 0.031459  
Add 48192.7 0.039438 0.031872 0.058144  
Triad 59983.3 0.036007 0.025607 0.044139  
-------------------------------------------------------------  
Solution Validates avg error less than 1.000000e-13 on all three arrays  
-------------------------------------------------------------

## 四、CPU功耗控制

### 1. 通过RAPL控制CPU功耗上限。

源码可在 <https://github.com/PUPiL2015/PUPIL/tree/master/src/RAPL> 上下载。

#### 1.1 编译

在重新编译前，需要修改每个socket的核心数。

$ vim RaplSetPowerSeprate.c

int main(int argc, char \*\*argv) {  
 int fd1,fd2;  
 int core1=0;  
 int core2=14; //修改为每个socket的核心数。  
 ...  
}

$ vim RaplPowerLimitDisable.c

fd=open\_msr(14); //找到这一行，改为每个socket的核心数。

$ make #重新编译，会有一些警告信息，可以忽略。

#### 1.2 设置功耗上限

./RaplSetPowerSeprate 120 120 #将两个socket最高功率分别设为120w

#### 1.3 重置初始设置

./RaplPowerLimitDisable

### 2. 通过BIOS设置RAPL

在浪潮的机器上，存在以上文件无法修改功率上限的问题。只能通过BIOS中RAPL进行控制。

### 3. 通过cpu调频控制CPU功耗

linux内核带有控制cpu频率的功率cpupower <https://wiki.archlinux.org/index.php/CPU_frequency_scaling>

$ sudo cpupower frequency-set -g performance # 设置为高性能模式  
$ sudo cpupower frequency-set -u 2400mhz # 将所有cpu核心的时钟频率设置为2400mhz

$ cpupower frequncey-info # 查看cpu频率设置信息  
  
analyzing CPU 0:  
 driver: intel\_pstate  
 CPUs which run at the same hardware frequency: 0  
 CPUs which need to have their frequency coordinated by software: 0  
 maximum transition latency: Cannot determine or is not supported.  
 hardware limits: 1.20 GHz - 3.30 GHz  
 available cpufreq governors: performance powersave  
 current policy: frequency should be within 1.20 GHz and 2.40 GHz.  
 The governor "powersave" may decide which speed to use  
 within this range.  
 current CPU frequency: Unable to call hardware  
 current CPU frequency: Unable to call to kernel  
 boost state support:  
 Supported: yes  
 Active: yes

$ watch "cat /proc/cpuinfo | grep MHz" #查看实时cpu频率,  
  
cpu MHz : 2199.937  
cpu MHz : 2299.968  
cpu MHz : 2299.968  
cpu MHz : 1200.093  
cpu MHz : 1199.906  
cpu MHz : 1200.000  
cpu MHz : 1200.000  
cpu MHz : 1200.000  
cpu MHz : 1599.750  
cpu MHz : 2299.875  
cpu MHz : 2299.593  
cpu MHz : 2300.062  
cpu MHz : 1200.000  
cpu MHz : 1200.000  
cpu MHz : 1979.156  
cpu MHz : 2299.968  
cpu MHz : 1698.937  
cpu MHz : 1200.000  
cpu MHz : 1349.062  
cpu MHz : 1699.968  
cpu MHz : 1399.968  
cpu MHz : 2299.781  
cpu MHz : 1200.093  
cpu MHz : 1200.000  
cpu MHz : 2300.062  
cpu MHz : 2300.156  
cpu MHz : 1199.906  
cpu MHz : 1199.906

## 五、 GPU功耗控制

cuda8.0 中提供了 nvidia-smi 的接口来控制和查看GPU的相关信息。  
一些重要的选项有 -pm -e -p -r -ac -rac -pl

### 1. 直接控制GPU功率上限（粗粒度）

$ sudo nvidia-smi -pl 250 #设置GPU功率上限为250w  
  
Power limit for GPU 0000:06:00.0 was set to 250.00 W from 250.00 W.  
Power limit for GPU 0000:84:00.0 was set to 250.00 W from 250.00 W.  
All done.

### 2. 控制GPU频率（细粒度）

* 查看可选频率信息

$ nvidia-smi -q -i 1 -d SUPPORTED\_CLOCKS  
  
==============NVSMI LOG==============  
  
Timestamp : Thu Jul 13 15:02:43 2017  
Driver Version : 375.26  
  
Attached GPUs : 2  
GPU 0000:84:00.0  
 Supported Clocks  
 Memory : 715 MHz  
 Graphics : 1328 MHz  
 Graphics : 1316 MHz  
 Graphics : 1303 MHz  
 Graphics : 1290 MHz  
 Graphics : 1278 MHz  
 Graphics : 1265 MHz  
 Graphics : 1252 MHz  
 Graphics : 1240 MHz  
 Graphics : 1227 MHz  
 Graphics : 1215 MHz  
 Graphics : 1202 MHz  
 Graphics : 1189 MHz  
 Graphics : 1177 MHz  
 Graphics : 1164 MHz  
 Graphics : 1151 MHz  
 Graphics : 1139 MHz  
 Graphics : 1126 MHz  
 Graphics : 1113 MHz  
 Graphics : 1101 MHz  
 Graphics : 1088 MHz  
 Graphics : 1075 MHz  
 Graphics : 1063 MHz  
 Graphics : 1050 MHz  
 Graphics : 1037 MHz  
 Graphics : 1025 MHz  
 Graphics : 1012 MHz  
 Graphics : 999 MHz  
 Graphics : 987 MHz  
 Graphics : 974 MHz  
 Graphics : 961 MHz  
 Graphics : 949 MHz  
 Graphics : 936 MHz  
 Graphics : 923 MHz  
 Graphics : 911 MHz  
 Graphics : 898 MHz  
 Graphics : 885 MHz  
 Graphics : 873 MHz  
 Graphics : 860 MHz  
 Graphics : 847 MHz  
 Graphics : 835 MHz  
 Graphics : 822 MHz  
 Graphics : 810 MHz  
 Graphics : 797 MHz  
 Graphics : 784 MHz  
 Graphics : 772 MHz  
 Graphics : 759 MHz  
 Graphics : 746 MHz  
 Graphics : 734 MHz  
 Graphics : 721 MHz  
 Graphics : 708 MHz  
 Graphics : 696 MHz  
 Graphics : 683 MHz  
 Graphics : 670 MHz  
 Graphics : 658 MHz  
 Graphics : 645 MHz  
 Graphics : 632 MHz  
 Graphics : 620 MHz  
 Graphics : 607 MHz  
 Graphics : 594 MHz  
 Graphics : 582 MHz  
 Graphics : 569 MHz  
 Graphics : 556 MHz  
 Graphics : 544 MHz

* 查看当前频率

$ nvidia-smi -q -i 1 -d CLOCK  
  
==============NVSMI LOG==============  
  
Timestamp : Thu Jul 13 15:04:03 2017  
Driver Version : 375.26  
  
Attached GPUs : 2  
GPU 0000:84:00.0  
 Clocks  
 Graphics : 405 MHz  
 SM : 405 MHz  
 Memory : 715 MHz  
 Video : 835 MHz  
 Applications Clocks  
 Graphics : 1189 MHz  
 Memory : 715 MHz  
 Default Applications Clocks  
 Graphics : 1189 MHz  
 Memory : 715 MHz  
 Max Clocks  
 Graphics : 1328 MHz  
 SM : 1328 MHz  
 Memory : 715 MHz  
 Video : 1328 MHz  
 SM Clock Samples  
 Duration : 0.16 sec  
 Number of Samples : 3  
 Max : 1189 MHz  
 Min : 405 MHz  
 Avg : 431 MHz  
 Memory Clock Samples  
 Duration : 0.16 sec  
 Number of Samples : 3  
 Max : 715 MHz  
 Min : 715 MHz  
 Avg : 715 MHz  
 Clock Policy  
 Auto Boost : N/A  
 Auto Boost Default : N/A

* 设置GPU频率

$ sudo nvidia-smi -ac 715, 1088 #设置Graphics时钟频率为1088MHz  
  
Applications clocks set to "(MEM 715, SM 1088)" for GPU 0000:06:00.0  
Applications clocks set to "(MEM 715, SM 1088)" for GPU 0000:84:00.0  
All done.