技術解説

ECO な HPC：CARMA DEVKIT を試して

総合情報基盤センター 准教授 布村紀男

GPU(graphics processing unit)は消費電力当たりの演算性能が高く、スーパーコンピュータの低消費電力化に向けたアクセラレータとし広く知られるようになっている。倍精度浮動小数点演算性能の向上や ECC メモリに対応した製品の供給により HPC（high performance computing）での利用条件が整っている。CPU でもマルチコア化や低消費電力化技術が進展している。スマートフォンや 7 インチタブレットなどに使用される省電力 ARM ベース CPU と GPU を組み合わせた開発キットの試用について取り上げる。

キーワード：ARM®アーキテクチャ，Tegra®3，CPU/GPU ハイブリッド，CUDA®，低消費電力

１．はじめに

ウィキペディアによると、Green500 は世界で最もエネルギー消費効率の良いスーパーコンピュータ(スパコン)を定期的にランク付け評価しているとある。2012 年 6 月の結果では、1 位～20 位までを組み込み用途向けプロセッサを搭載した IBM 社の BlueGine/Q が独占している。ARM アーキテクチャを採用したプロセッサも携帯機器などのへの組み込みに適した低消費電力が特徴である。最近ではモバイル機器以外にもノート PC や省電力サーバへの採用が検討されている。そうした中、スパコンへの取り組むとして、スペインの Barcelona Supercomputing Center は、ARM ベースのクアッドコアプロセッサ「Tegra 3」搭載のスパコンを開発している[1]。この ARM スパコンでは、計算速度は世界トップクラスとはいかないが、エネルギー消費の効率化が期待できる。いずれは Green500 に登場するかもしれない。本稿では、SECO 社が提供している Tegra3(CPU)と Quadro100M(GPU)を搭載した CARMA 開発キット[2]の試用を報告する。

２．製品仕様

今回試した開発キット本体の写真を図 1 に示す。手前が GPU、奥には CPU が配置されている。GPU には冷却ファン、CPU にはヒートシンクが装着されている。製品仕様を表-1 に示す。

図 1 CARMA Development kit (SECO 社)

表-1 開発キット(CARMKit)製品仕様

CPU	NVIDIA®Tegra®3 Quad-CoreARM® Cortex A9
GPU	NVIDIA®Quadro® 1000M with 96 CUDA® Cores
CPU:Memory	2 GB
GPU:Memory	2 GB
Peak Performance	270 GFlops single procession
CPU - GPU Interface	4x PCIe Gen1 link
Network	1x Gigabit Ethernet
Storage	1x SATA 2.0 Connector
USB	3x USB 2.0
Display	HDMI
Software	Linux Ubuntu Derivative OS CUDA Tool Kit

３．動作確認

起動は、電源ボタンを押して待つこと 37 秒後、ログイン画面が表示される。しばらくすると自動的にログインする(図 2)。ユーザ名は ubuntu となっている。ウィンドウマネージャは軽量で高速な OpenBox が使われている。

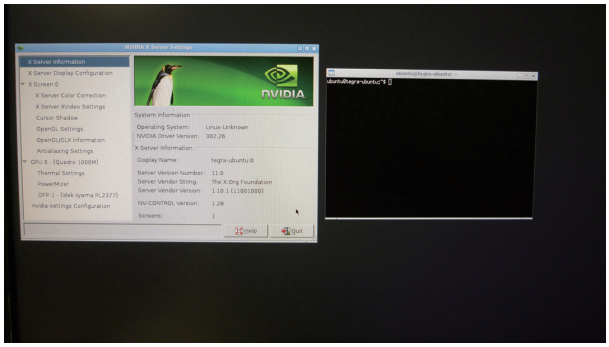

図 2 Openbox ウィンドウマネージャ

開発キットにインストールされているコンパイル済み CUDA_sample を動かしてみた。図 3 に N-body と oceanFFT を動かしている様子を示す。

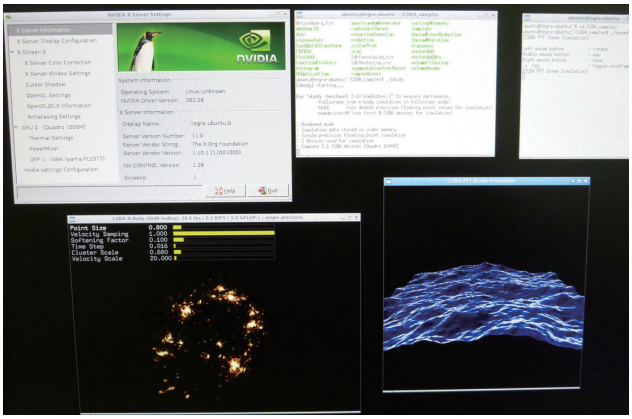

図 3 CUDA サンプルプログラム実行画面
(N-body, oceanFFT)

４．性能評価

最初に CUDA_sample にある DeviceQuery の出力結果を図 4 に示す。次にベンチマークソフト UnixBench[3]により調べた結果を表-2 に示す。比較として総合情報基盤センターでサービスしている仮想サーバの結果(Intel(R) Xeon(R) CPU X5670 @ 2.93GHz 1CPU)を表-3 に示す。ここで Index 値は Baseline の指標となる「SPARCstation 20 Model61」の基準を 10 とした場合の実測値の倍率を表す。最終行の System Benchmark Index Socre が総合的なベンチマークの値である。CARMkit の性能は「203.3」で、仮想サーバの値「1735.5」よりはかなり小さい。UNIX サーバの用途として期待できない。

```
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Found 1 CUDA Capable device(s)

Device 0: "Quadro 1000M"
  CUDA Driver Version / Runtime Version          4.2 / 4.2
  CUDA Capability Major/Minor version number:    2.1
  Total amount of global memory:                 2048 MBytes (2147155968 bytes)
  ( 2) Multiprocessors x ( 48) CUDA Cores/MP:    96 CUDA Cores
  GPU Clock rate:                                1400 MHz (1.40 GHz)
  Memory Clock rate:                             900 Mhz
  Memory Bus Width:                              128-bit
  L2 Cache Size:                                 131072 bytes
  Max Texture Dimension Size (x,y,z)             1D=(65536), 2D=(65536,65535), 3D=(2048,2048,
  Max Layered Texture Size (dim) x layers        1D=(16384) x 2048, 2D=(16384,16384) x 2048
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1536
  Maximum number of threads per block:           1024
  Maximum sizes of each dimension of a block:    1024 x 1024 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 65535
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and execution:                 Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
```

図 4 DeviceQuery の出力

表-2 UnixBench の値

Test	Score	Unit	Time	Iters.	Baseline	Index
Dhrystone 2 using register variables	4578754.3	lps	10.1 s	7	116700.0	392.4
Double-Precision Whetstone	884.7	MWIPS	10.1 s	7	55.0	160.9
Execl Throughput	170.0	lps	29.9 s	2	43.0	39.5
File Copy 1024 bufsize 2000 maxblocks	112891.9	KBps	30.0 s	2	3960.0	285.1
File Copy 256 bufsize 500 maxblocks	34619.2	KBps	30.0 s	2	1655.0	209.2
File Copy 4096 bufsize 8000 maxblocks	249846.5	KBps	30.0 s	2	5800.0	430.8
Pipe Throughput	250737.5	lps	10.1 s	7	12440.0	201.6
Pipe-based Context Switching	48756.3	lps	10.1 s	7	4000.0	121.9
Process Creation	1696.2	lps	30.0 s	2	126.0	134.6
Shell Scripts (1 concurrent)	483.7	lpm	60.1 s	2	42.4	114.1
Shell Scripts (8 concurrent)	264.4	lpm	60.1 s	2	6.0	440.7
System Call Overhead	703638.9	lps	10.1 s	7	15000.0	469.1
System Benchmarks Index Score:						203.3

表-3 仮想サーバの値

Test	Score	Unit	Time	Iters.	Baseline	Index
Dhrystone 2 using register variables	18403807.6	lps	10.0 s	7	116700.0	1577.0
Double-Precision Whetstone	1811.3	MWIPS	9.2 s	7	55.0	329.3
Execl Throughput	5819.4	lps	29.7 s	2	43.0	1353.3
File Copy 1024 bufsize 2000 maxblocks	1206653.8	KBps	30.0 s	2	3960.0	3047.1
File Copy 256 bufsize 500 maxblocks	346659.8	KBps	30.0 s	2	1655.0	2094.6
File Copy 4096 bufsize 8000 maxblocks	1836943.0	KBps	30.0 s	2	5800.0	3167.1
Pipe Throughput	2885728.7	lps	10.0 s	7	12440.0	2319.7
Pipe-based Context Switching	558034.2	lps	10.0 s	7	4000.0	1395.1
Process Creation	17307.2	lps	30.0 s	2	126.0	1373.6
Shell Scripts (1 concurrent)	8697.3	lpm	60.0 s	2	42.4	2051.2
Shell Scripts (8 concurrent)	1092.5	lpm	60.0 s	2	6.0	1820.8
System Call Overhead	4814525.8	lps	10.0 s	7	15000.0	3209.7
System Benchmarks Index Score:						1737.5

続いては CUDA のサンプルプログラム N-body を使用しての計算性能結果を表-4 に示す。比較として以前計測した 9800GTX+の結果[4]も併せて記載する。

表-4 CUDA N-body ベンチマーク

N	Quadro 1000M (96core)	9800GTX+ (128core)
1024	94.800 GFlops	84.593 GFlops
4096	94.755 GFlops	356.992 GFlops
16384	94.738 GFlops	364.917 GFlops

製品仕様によればパフォーマンスのピーク値は270GFlops である。しかしここでの N-body のベンチマークでは半分にも満たない結果であった。表-5 に計算時の消費電力と効率を参考値として示す。単精度計算ではあるが 1W あたり 5GFlops を超える高効率であることがわかる。

表-5 N-body 実行時の消費電力と効率

N	消費電力(W)	効率(GFlops/W)
1024	17.1	5.54
2048	17.3	5.48
4096	16.7	5.67
8192	17.3	5.48

PC の消費電力は[5]によれば、

1. 一般的なノート PC 20～30W
2. 省電力 PC　50W 未満
3. 一般的な PC 50～150W
4. 消費電力の大きなグラフィックカードや CPU などを使用している PC 150W～

とある。この開発キットでは計算を実行しない通常時は 11～12W、GPU も利用した計算でも最大 20W 未満であった。ノート PC よりも低い値に抑えられおり、優れていることがわかる。

４．クロスコンパイル開発環境の構築

CARMkit は ARM ベース CPU のため x86 機ではクロスコンパイル開発環境を構築する。

(1) OS のインストール

まずは、Ubuntu Linux 11.04 を導入から始める。32bit 版か 64bit 版か迷うところである。CARMkit は 32bit であるので、64bit 版をインストールした場合は、32bit の開発ライブラリをインストールする必要がある。

(2) ARM 用 gcc コンパイラのインストール

```
$ sudo apt-get install gcc-4.5-arm-linux-gnueabi
g++-4.5-arm-linux-gnueabi
```

(3) CUDA ツールキットのインストール

SECO 社の Web サイト[2]から CUDA ツールキットをダウンロードしてインストールを行う。

```
$ sudo sh cuda-linux-ARMv7-rel-4.2.10-13489154.run
```

(4) PATH の追加

コンパイラ nvcc のインストール PATH を個人環境.bashrc にを追加し、source コマンドにて PATH 設定を更新させる。

```
export PATH=/usr/local/cuda/bin:$PATH
$ source .bashrc
```

(5) CUDA コンパイラの確認

```
$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2012 NVIDIA Corporation
Built on Tue_Jul_17_14:48:12_PDT_2012
Cuda compilation tools, release 4.2, V0.2.1221
```

(6) テストコードのコンパイル

開発環境は x86 マシンであるので、ARM ベースの CARMkit 上で動作させるには、コンパイルオプション -target-cpu-arch=ARM の指定が必要である。さらに CUDA-ツールキットが 64bit の場合は、-m32 を指定すると 32bit 環境に対応できる。Web サイト[5]に掲載されていた GPU と CPU の両方での行う演算の比較するソースコード gpu_test.cu を使ってコンパイルを実行した。

```
$nvcc -target-cpu-arch ARM --compiler-bindir
/usr/bin/arm-linux-gnueabi-gcc-4.5 -m32  -o gpu_test
gpu_test.cu
/usr/lib/gcc/arm-linux-gnueabi/4.5.2/../../../../arm-linux-gnueabi
/bin/ld: warning: libc.so, needed by
/usr/lib/gcc/arm-linux-gnueabi/4.5.2/libgcc_s.so.1, not found
(try using -rpath or -rpath-link)
```

警告メッセージが表示されるものの、実行ファイル gpu_test は作成される。

(8) CARMKit 上での動作確認

CARMkit ではネットワーク機能(Ethernet)が利用できる。x86 環境でクロスコンパイルしてできた実行ファイル gpu_test をネットワーク経由で scp コマンドによりコピーする。

```
$ scp gpu_test Ubuntu@192.168.0.151:~
```

テストプログラムの実行は SSH を使ってリモートログイン、もしくは CARMKit で直接ログインして行う。図 5 に実行結果を示す。2 列目が GPU, 3 列目が CPU の結果である。同じ値を出力しており、問題なく動作することが確認できた。

```
ubuntu@tegra-ubuntu:~$ ./gpu_test
1 1.01999998092651367187500000000000000000000 1.01999998092651367187500000000000000000000
2 1.04039990901947021484375000000000000000000 1.04039990901947021484375000000000000000000
3 1.06120789051055908203125000000000000000000 1.06120789051055908203125000000000000000000
4 1.08243203163146972656250000000000000000000 1.08243203163146972656250000000000000000000
5 1.10408067703247070312500000000000000000000 1.10408067703247070312500000000000000000000
6 1.12616229057312011718750000000000000000000 1.12616229057312011718750000000000000000000
7 1.14868545532226562500000000000000000000000 1.14868545532226562500000000000000000000000
8 1.17165911197662353515625000000000000000000 1.17165911197662353515625000000000000000000
9 1.19509232044219970703125000000000000000000 1.19509232044219970703125000000000000000000
10 1.21899414062500000000000000000000000000000 1.21899414062500000000000000000000000000000
```

図 5

5．gfortran と OpenMPI の導入

CARMKit には開発環境のパッケージがインストールされていない。そのため CARMkit 上での開発に必要なソフトウェアやライブラリを導入することになる。gcc は導入済みなので gfortran と openmpi のインストールを紹介する。

(1) gfrotran のインストール

gfortran はパッケージが入手できるのでそれを導入する。ここでは apt-get コマンドを使ってインストールする。

```
$sudo apt-get install gfortran
```

(2) OpenMPI のインストール

OpenMPI の Ver.1.6.2 は ARM ベース CPU をサポートしており Web サイト[7]からソースを入手して導入を行う。

```
$ tar xvfz openmpi-1.6.2.tar.gz
$ cd openmpi-1.6.2
$ ./configure
$ sudo make –j 4 install
```

環境変数として/usr/local/bin を PATH にそして LD_LIBRARY_PATH に/usr/local/lib を追加しておく。図 6 と 7 に OpenMP と OpenMPI を用いたプログラム実行の様子を示す。

```
ubuntu@tegra-ubuntu:~/Test$ cat -n omp_test.f90
     1  program omp
     2
     3    use omp_lib
     4    write(*,*) 'Start'
     5   !$omp parallel
     6    write(*,*) 'Hello OMP world'
     7   !$omp end parallel
     8    write(*,*) 'End'
     9  end program

ubuntu@tegra-ubuntu:~/Test$ gfortran -fopenmp -o omp_test omp_test.f90
ubuntu@tegra-ubuntu:~/Test$ export OMP_NUM_THREADS=4
ubuntu@tegra-ubuntu:~/Test$ ./omp_test
 Start
 Hello OMP world
 Hello OMP world
 Hello OMP world
 Hello OMP world
 End
```

図 6

```
ubuntu@tegra-ubuntu:~/Test$ mpicc -o mpi_test mpi_test.c
ubuntu@tegra-ubuntu:~/Test$ mpiexec -n 4 mpi_test
Process 0 on tegra-ubuntu
Process 1 on tegra-ubuntu
Process 3 on tegra-ubuntu
Process 2 on tegra-ubuntu
pi is approximately 3.1416009869231249, Error is 0.0000083333333318
wall clock time = 0.004842
ubuntu@tegra-ubuntu:~/Test$
```

図 7

6．おわりに

ARM ベース CPU と GPU のハイブリッド HPC 向けの開発キットについて駆け足で述べてみた。高性能と低消費電力に向けての取り組みは HPC に限らずサーバ、デスクトップ PC そしてノート PC に共通の課題である。2013 年の初頭から NVIDIA Tegra4 クアッドコア Cortex-A15 72GPU コアのモバイルプロセッサの発表もあり今後活発化するこうした技術動向にも注目したい。

参考文献

[1] http://www.bsc.es/

[2] CARMA DEVKIT
http://www.seco.com/carmakit

[3] byte-unixBench
http://code.google.com/p/byte-unixbench

[4] 富山大学総合情報基盤センター広報，Vol.6, p.63-66 (2009)

[5] http://www.geocities.jp/arcus_270/b/ecopc.htm

[6] http://cudamusing.blogspot.jp/2012/10/setting-up-carma-kit.html

[7] OpenMPI
http://www.open-mpi.org/