B-004

OpenCL を用いたマルチ GPU 環境の構築

Multi GPU computer system using OpenCL

祢津 佑 * 南 昇吾 * 魚谷 智志 * 山本 未来呂 * 増田 信之 *
Tasuku Netsu Shogo Minami Satoshi Uotani Mikuro Yamamoto Nobuyuki Masuda

1. まえがき

近年，高い演算性能を安価に得られる並列計算ハードウェアとして GPU(Graphics Processing Unit) が注目されている．GPU は一般的に画像処理に用いられるが，数値計算に応用することができる．このことを GPGPU(General purpose computing on GPU) と呼ぶ．GPU は, CPU に比べ計算コアが多く並列計算が得意であるため, 数値計算の高速化が期待できる．また，GPU は比較的安価であるため複数用いる場合があり，このことをマルチ GPU と呼ぶ．マルチ GPU は単一の GPU に比べ高速化が期待できるが，現在では GPU のベンダを統一する必要性がある．これは, GPU を用いて数値計算させるときに使用する開発環境が統一されていないからである．

現在, 主流の GPGPU の開発環境は NVIDIA 社から提供されている CUDA(Compute Unified Device Architecture) である．しかし, CUDA は NVIDIA 社の GPU のみに対応した開発環境であり, 他社の GPU に用いることができない欠点がある．一方, GPGPU の開発環境は上記に示した CUDA 以外にも OpenCL(Open Computing Language) がある．OpenCL は, OpenGL で有名な Khronos Group によって策定されており, CUDA と違いベンダに依存しない開発環境である．

そこで, 本研究では開発環境の OpenCL を用いて, ベンダ違いのマルチ GPU 環境下でシミュレーションを行った．計算題材は, CGH(Computer Generated Hologram) を用いた．

2. GPGPU の開発環境

現在, GPU のベンダは NVIDIA 社と AMD 社がある．この 2 社の GPU を用いて GPGPU を行う際, その開発環境は使用する GPU によって制限されている．NVIDIA 社の GPU であれば CUDA, AMD 社の GPU であれば ROCm(Radeon Open Compute) である．そのため, マルチ GPU で GPGPU を行う場合は同じ GPU ベンダ同士の組み合わせでしかできない．しかし, 各ベンダの GPU にはそれぞれ特徴が違うため, その特徴をうまく組み合わせることで高速化の期待ができる．

GPGPU の開発環境には，上記に述べた CUDA と AMD APP 以外にも OpenCL という開発環境がある．OpenCL は, ヘテロジニアス型並列計算環境で利用できる共通フレームワークであり, 正確には開発環境ではない．しかし, OpenCL で作成したフレームワークは OpenCL に対応したプロセッサであればベンダに依存せず利用できる [1]．このことから, OpenCL を用いることでベンダ違いのマルチ GPU ができると予想できる．また, 近年では NVIDIA 社, AMD 社の GPU は OpenCL に対応している．

3. CGH

CGH とは, Computer-Generated Hologram の略で計算機合成ホログラムのことを指し, ホログラフィ技術の応用例の 1 つある．ホログラムは物体の三次元情報を記録した干渉縞のことであり, 計算によって求めることが可能である．しかし, CGH は計算量が膨大であるため高速化が求められている [2]．

ホログラムの画素値 I は, 参照光の波長 λ と物体光の振幅 A_a, ホログラムの画素間隔 p, 各物体点の座標 (x_a, y_a, z_a), 物体点の総数 N とすると, 以下の (1) 式を用いて求めることができる．

$$I(x_i, y_i) = \sum_{a=0}^{N-1} A_a \cos\left(\frac{2\pi}{\lambda} p\sqrt{R}\right) \quad (1)$$

$$R = (x_a - x_i)^2 + (y_a - y_i)^2 + z_a^2 \quad (2)$$

上記の式から計算量が大きくなる要素は, 作成するホログラムの画素数 M と物体点数 N である．よって, 計算量 O は以下のようになる．

$$O = M \times N \quad (3)$$

一方, (1)(2) 式からホログラムの計算は周辺の画素の影響を受けることなく別々に求めることができることがわかる．このことから, CGH は非常に並列計算がしやすい題材であり, GPGPU を用いて高速化しやすいことがわかる．よって, 本稿は CGH を計算題材とした．ただし, 今回は (1)(2) 式ではなくフレネル近似ができる条件下でホログラムを作成したので, フレネル近似式を用いた以下の計算式を用いた．

$$I(x_i, y_i) \simeq \sum_{a=0}^{N-1} \cos\left(\frac{\pi p}{\lambda} \frac{(x_a - x_i)^2 + (y_a - y_i)^2}{z_a}\right) \quad (4)$$

4. 実験結果

本研究では, CGH を異なる 2 台の GPU を用いて計算させて計算時間を測定した．2 台の GPU には, それぞれ同一のデータを送り, 同一の計算を同時にさせた．ホログラムの計算部分以外は CPU に処理をさせたので, 本研究で測定する計算時間は (4) 式の計算時間である．作成するホログラムは各 GPU につき 1 枚なので, 計 2 枚を作成するプログラムを本実験で使用した．また, 今回は物体点数 $N = 11,646$ 点の tyranno を用いて, 画素数 1,024×1,024 のホログラムを作成した．その使用した tyranno のデータを図 1 に, 作成したホログラムを図 2 に示す．

*東京理科大学基礎工学部

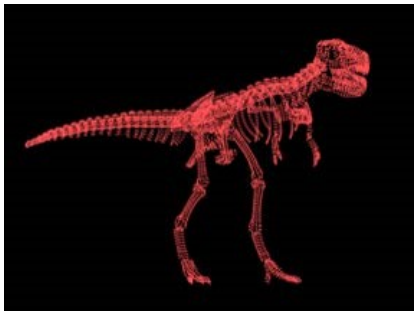

図 1: tyranno の CG データ

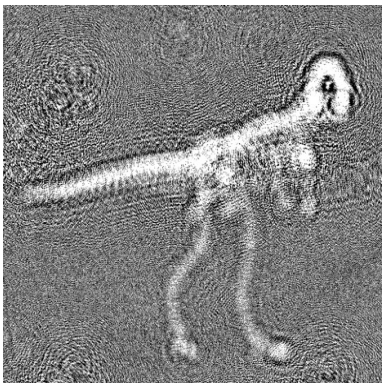

図 2: tyranno のホログラム

4.1 実験環境

本実験で使用したソフトウェアの開発環境を表 1 に，GPU の性能を表 2 に示す．計算時間の計測には，gettimeofday, OpenCL の clGetEventProfilingInfo を用いた．CPU-GPU 間のメモリはグローバルメモリを用い，カーネル起動時間に関しては今回に限り無視する．そのため, 純粋な各 GPU の計測時間のみを測定した．また，使用するカーネルは全く同じものを使用したので GPU のコア数などに依存せず, global size を x 方向と y 方向共に 1,024, local size を x 方向と y 方向共に 16 と固定してすべて単精度で計算している．計算時間の計測は，プログラムを 20 回実行させ, その平均をとったものを今回の結果とする．

表 1: 開発環境

CPU	AMD A10-7890K 4.10 GHz
メモリ	16 GB (8 GB × 2)
GPU	AMD Fiji Radeon R9 Fury NVIDIA GeForce GTX 1080
OS	CentOS Linux 7.3.1611
GPGPU の開発環境	ROCm 1.6 (OpenCL 2.0) CUDA-8.0 (OpenCL 1.2)

表 2: 各 GPU の性能

GPU	1080	Fury
コア数	2,560	3,584
ベースクロック [MHz]	1,607	1,000
メモリ量 [GB]	8	4
理論性能 [TFLOPS]	8.873	7.168

4.2 計測結果と考察

シミュレーション結果を表 3 に示す．この結果より，プログラムの実行時間が各 GPU の計算時間の和よりも速いことから，同一 PC 上において各 GPU はほぼ同時に動作していると分かる．これは，R9 Fury のカーネル実行命令と GTX 1080 のカーネル実行命令との時間差を gettimeofday で測定した際に，20 回の平均が 0.5308 [msec] であったことから同様のことがいえる．

表 3: 計算時間の平均値

GPU	各 GPU の計算時間	プログラムの実行時間
R9 Fury	0.2709 sec	0.2728 sec
GTX 1080	0.2335 sec	

5. まとめと今後の課題

本稿では, OpenCL を用いて数値計算の高速化の検討を行った．本研究の結果から, OpenCL を用いて同一 PC 上でベンダ違いのマルチ GPU ができることがわかり, また, GPU の性能に依存した結果が得られることがわかった．このことから, GPU の組み合わせにより更なる高速化が期待できることがわかった．

今回は, global size と local size の並列数や使用するメモリなどを固定して同一のプログラムでシミュレーションした．今後の課題としては, その並列数やメモリを各 GPU に適したものに設定し更なる高速化を検討する．

参考文献

[1] 伊藤智義, “GPU プログラミング入門”, 講談社, 2013

[2] Tomoyoshi Shimobaba, Tomoyoshi Ito, Nobuyuki Masuda, Yasuyuki Ichihashi, “Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL”, Opt. Express, Vol.18, pp. 9955-9960 (2010).