B-024

ギガビット・イーサネットを搭載したマルチ GPU クラスタシステムによる計算機合成ホログラムの計算高速化

Fast computation of computer-generated hologram using multi-GPU cluster system with gigabit ethernet

三宮 廣海 † 中山 弘敬 ¶ 角江 崇 §
Hiromi Sannomiya Hirotaka Nakayama Takashi Kakue
下馬場 朋禄 § 伊藤 智義 § 髙田 直樹 ‡
Tomoyoshi Shimobaba Tomoyoshi Ito Naoki Takada

1. はじめに

ホログラフィは三次元物体から発せられる物体光を忠実に記録・再生できる唯一の方法である．視覚的疲労がなく，立体像を様々な角度から見ることが可能である．コンピュータで計算したホログラムを計算機合成ホログラム（CGH：Computer-Generated Hologram）と呼ぶ．CGH による三次元物体の再生技術は電子ホログラフィと呼ばれ，“究極の三次元テレビ”になるものと期待されている．しかし，CGH は計算量が膨大であること，1 画素が 1 μm 以下の高精細な電子表示デバイスが必要となることなど多くの問題が存在し，実用化には至っていない．

電子ホログラフィを用いた三次元テレビの実現には CGH を秒間 30 枚以上表示する必要があり，CGH 計算のために高い演算能力を必要とする[1]．近年，GPU（Graphics Processing Unit）の浮動小数点演算性能とコストパフォーマンスは著しく向上している．CGH 計算は使用するデータ量に比べ演算量が多く並列化に向いている．2006 年に CUDA（Compute Unified Device Architecture）と呼ばれる GPU プログラミングの統合開発環境がリリースされ，GPU を用いたシステム開発は容易になり [2]，GPU を用いた CGH 計算に関する研究が報告された [3]．カラー電子ホログラフィの計算高速化の研究もなされている[4,5]．複数の GPU を搭載したマルチ GPU 環境 PC を複数台使用したマルチ GPU クラスタシステムによる CGH 計算の高速化についての研究が報告されている [6,7].

複数の空間光変調器（SLM：Spatial Light Modulator）とマルチ GPU クラスタによる CGH 計算の高速化についても報告されている [8]．膨大な画素数からなる CGH において，分割された CGH 毎に GPU が割り当てられ計算される．計算された CGH はそれぞれの GPU に接続された SLM に直接出力されるため，スケーラビリティが高くなる．しかし SLM を複数用いるため，非常に高価で大規模なシステムとなる．また，光学系の位置調整も容易ではない．

これに対して，Full HD の解像度を持つ SLM を 1 つ用いたマルチ GPU クラスタによる電子ホログラフィは，コストパフォーマンスもよく，光学系も小規模で実用的である．しかし，Full HD の解像度を持つ CGH を複数の GPU で計算高速化を実現することは，PC 間の転送時間がボトルネックとなり容易ではない．転送時間のボトルネックを打開するため，InfiniBand を用いたマルチ GPU クラスタシステムによるリアルタイム電子ホログラフィが提案されている [7]．しかし，InfiniBand は高価であり，汎用性の高い Gigabit Ethernet を用いて実現することが望ましい．

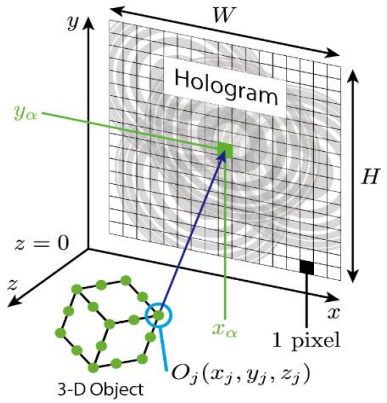

図 1　CGH 計算の座標系

本論文では，PC 間で転送するデータ量を低減することで，汎用性の高いギガビット・イーサネットを搭載したシステムによる CGH 計算の高速化を提案する．最終的に，12 枚の GPU（NVIDIA GeForce GTX 1080 Ti)を CGH 計算に用いて，約 20 万点からなる三次元物体の CGH を 30 fps で再生することに成功した．このとき，実効性能は約 95.90TFLOPS となり，理論性能に対して約 75%の性能を発揮している．また，CPU（Intel Core i7 7800X）に比べて約 1519 倍の計算高速化を実現した．

2. 計算機合成ホログラム (CGH)

図 1 に CGH 計算の座標系を示す．三次元物体を点で表し，物体を構成する点数をN_pとする．そのとき，ホログラム面上の点(x_α, y_α)における光の強度は，次式となる[3].

$$I(x_\alpha, y_\alpha) = \sum_{j=1}^{N_p} \cos\theta \tag{1}$$

$$\theta = \frac{\pi}{\lambda z_i}\left\{(x_\alpha - x_j)^2 + (y_\alpha - y_j)^2\right\} \tag{2}$$

ここで，変数 α はホログラム点を表す．物体点の座標を(x_j, y_j, z_j)とした．λは三次元情報の記録に使用される参照光の波長である．

CGH 上の 1 画素の光の強度を求めるには，式(2)を$j = 1$から N_p まで繰り返し計算する必要がある．よって，1 枚の CGH を作成するには，ホログラムの解像度を $W \times H$ とすると，その計算量は $(W \times H) \times N_p$ に比例する．

† 高知大学大学院総合人間自然科学研究科
‡ 高知大学教育研究部自然科学系理工学部門
§ 千葉大学大学院工学院
¶ 国立天文台シミュレーションプロジェクト

3. 提案手法

3.1 システム構成

図 2 に本論文で使用するマルチ GPU クラスタシステムの概略図を示す．本論文では GPU を 13 枚用いる．CGH 計算ノードでは，各 GPU で動画の各フレームの CGH を計算する．計算後，CGH 表示ノードに CGH 画像を転送する．CGH 表示ノードでは，CGH 計算ノードで作成された CGH 画像を空間光変調器(SLM)に次々と表示する．本システムでは CGH 計算ノードの GPU ボード（GPU 1-12）で計算された CGH を，ネットワークを介して CGH 表示ノード（PC 0）に転送する必要がある．

3.2 GPU クラスタによる CGH 計算

1 つの SLM に表示する CGH を，多数の GPU を用いて高速に計算を行う処理を図 3 に示す．図 3 では，CGH 計算に用いる GPU ボード N 枚と CGH 表示に用いる GPU ボード 1 枚の合計 N+1 枚の GPU ボードを使用する場合を示している．CGH 表示に用いる GPU ボードを GPU 0 とする．また，CGH 計算に用いる N 枚の GPU ボードを GPU 1～GPU N とする．1 枚の GPU ボードで三次元動画の 1 つのフレームの CGH を計算する．

図 3 のように，三次元動画の最初のフレーム (Frame 1) の CGH を GPU 1 で計算する．次に，2 番目のフレーム (Frame 2) の CGH を GPU 2 で計算する．同様に，N 番目のフレームの CGH を GPU N で計算する．N+1 番目のフレーム (Frame N+1) の CGH は GPU 1 で CGH を計算する．それ以降のフレームの CGH についても同様に各 GPU ボードに割り当てて計算を行う．各 GPU で計算された CGH は PC 0 へ送る．

PC 0 は Frame 1 の CGH を受け取り次第，直ちに GPU 0 がその CGH を SLM に描画する．次に，Frame 1 の CGH が SLM に表示されている間に，PC 0 は GPU 2 で計算された Frame 2 の CGH を受け取る．Frame 1 の CGH が SLM に一定時間表示された後，GPU 0 は Frame 2 の CGH を SLM に表示する．これを繰り返し，図 3 に示すように処理がなされる．図 3 の処理を図 2 に示すシステムに実装した．図 3 の処理を実現するために，MPI (Message Passing Interface) を使用した．

3.3 GPU 間の転送時間

CGH によるリアルタイム動画再生を実現するには，1 秒間に 30 枚の CGH を表示しなければならない．つまり，CGH 表示時間間隔が約 33ms 以内でなければならない．ネットワークに Gigabit Ethernet を使用した場合，1 枚の CGH データ転送時間は次のようになる．ここでは，CGH の解像度を 1,920×1,024，1 画素あたりのデータを 32 bit とした．

$$\frac{32[bit]\times 1,920[pixel]\times 1,024[pixel]}{1\times 10^{9}[bps]}\approx 63[ms] \qquad (3)$$

式(3)からわかるように，Gigabit Ethernet では GPU 間での通信がボトルネックとなる．そこで，高速なネットワークとして InfiniBand を搭載したシステムが提案された[7]．InfiniBand は高い信頼性，可用性，保守性を持つ．また，メモリバンド幅が広域帯で，低レイテンシ，高スループットといった特徴がある．転送レート QDR（Quad data rate）

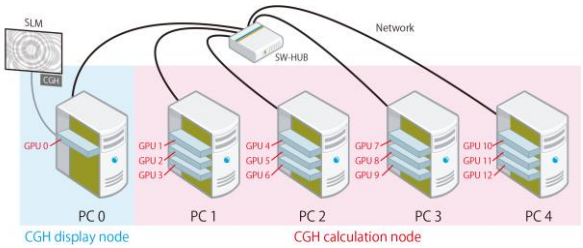

図 2　GPU クラスタシステムの概要図

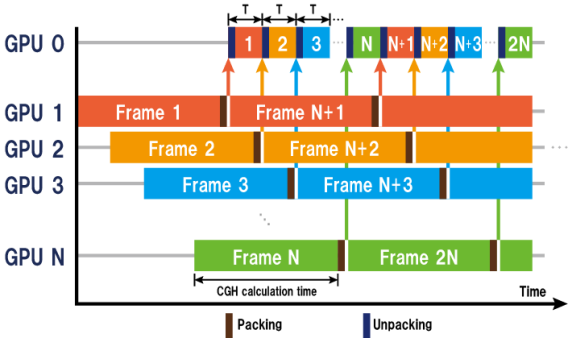

図 3　CGH 計算処理のアルゴリズム

の HCA およびスイッチングハブが使用された．InfiniBand QDR の実効転送レートは 32Gbps であるが，実効速度を測ったところ，約 25Gbps であった．したがって，1 枚の CGH 転送時間は次のように見積もることができる．

$$\frac{32[bit]\times 1,920[pixel]\times 1,024[pixel]}{25\times 10^{9}[bps]}\approx 3[ms] \qquad (4)$$

これにより通信のボトルネックが解消され，リアルタイム三次元動画再生が可能となる．

しかしながら，InfiniBand は高価であり，汎用性の点からも Gigabit Ethernet を用いて実現することが望ましい．そこで，本論文では，通信を行う前に 3.4 節で述べるバイナリ CGH のパッキング処理を行う．これにより表示ノードへ転送するデータ量が 1/32 に低減される．表示ノードでは，パッキングされた転送データを受信後，アンパッキング処理によりバイナリ CGH を復元し，空間光変調器（SLM）へ表示する．転送データ量が 1/32 になると，1 枚の CGH データ転送時間は次のように見積もることができる．

$$\frac{1[bit]\times 1,920[pixel]\times 1,024[pixel]}{1\times 10^{9}[bps]}\approx 2[ms] \qquad (5)$$

パッキング処理により Gigabit Ethernet でも通信速度のボトルネックを解消することが可能となる．

3.4 転送データの低減

バイナリ CGH は，最初に式(1),(2)から CGH 上の各画素の光強度 I を求める．CGH 上の各画素の光強度 I は，float 型の数値データ（32 bit）で表される．バイナリ CGH では，光強度 I の値が 0 より小さいところを黒，それ以外を白とし，黒と白の 2 色のバイナリ CGH 画像を作成する．よって，バイナリ CGH の各画素におけるデータは，2 値（1 bit）で表すことが可能である．

2 値データを 1 bit の並びとしてパッキング処理（図 4）を行い，unsigned int 型の変数に格納する．この処理により，CGH 計算ノードから CGH 表示ノードへの転送データ”

Packed data”は，1 画素あたり光強度 I の 32 bit から 1 bit となり 1/32 に低減される．パッキング処理は次の手順で行う（図 4）．

1. CGH 計算ノードにおいて，三次元物体点の位置座標から,CGH の各画素の光強度 I を式 (1) ,(2)より求める．
2. CGH 上の各画素において，光強度 I の値を参照し，光強度 I が 0 以上のときを“1”，それ以外を“0”とする．
3. 2.での値“0”もしくは“1”を，ビットシフトにより変数“Packed data”に 1 bit ずつ格納していく．すべての画素に対して，同様に行う．

マルチ GPU クラスタシステムにおいて，CGH 計算ノードで転送データ“Packed data”を作成し，CGH 表示ノードへ送る．CGH 表示ノードでアンパッキング処理（図 5）によりバイナリ CGH を作成し，SLM へ出力する．アンパッキング処理は次の手順で行う（図 5）．

1. 受信したデータを変数“Packed data”に格納し，先頭から 1 bit ずつ参照する．
2. 1.で参照した先頭ビットの値が“1”であれば，その画素を白とする．参照した先頭ビットの値が“0”であれば，その画素を黒とする．
3. 変数“Packed data”において，2 番目のビットをビットシフトにより変数の先頭に移動させる．

1～3 の処理を繰り返すことで，バイナリ CGH が復元される．なお，パッキング・アンパッキング処理は GPU によって高速に行われる．

4. 結果

GPU ボードとして NVIDIA GeForce GTX 1080 Ti を使用する(表 1)．表 1 に示す Theoretical performance は理論性能を示している．1 クロックで積和演算を実行できることから，1 クロックで 2 演算行うものとして理論性能を求めた．図 2 に示すシステムにおいて，各ノードのスペックを表 2 に示す．ネットワークとして Gigabit Ethernet を使用した．図 2 における SLM として Epson 社製プロジェクタ（EMP-TW1000）に搭載されている LCD パネルを使用した．本手法により 1,920×1,024 画素の CGH を GPU 0 で表示する CGH の表示時間間隔 T(図 3)を表 3 に示す．表 3 は，三次元物体を構成する物体点数に対して，図 2 に示された CGH 計算ノードの GPU ボードの枚数を用いたときの CGH 表示時間間隔 T を示している．三次元物体を構成する物体点数に比例にして CGH 表示時間間隔 T は増加している．なお，Full HD の解像度は 1,920×1,080 画素であるが，1,920×1,024 画素を用いた．これは GPU の演算性能を十分引き出すためである[6]．表 4 は表 3 の CGH 表示時間間隔 T から換算したフレームレートを示す．CGH 計算ノードの 12 枚の GPU ボードを用いたとき，204,800 点から構成される三次元物体の CGH を 1 秒間に約 30 枚表示できることを示しており，リアルタイムで動画再生できることを意味する．表 3 より求めた本手法による実効性能を表 5 に示す．ここで，式(2)において，$\pi/\lambda z_j$ をあらかじめ計算しておく．また，cos 関数は GPU の SFU (Special Function Unit)により 1 クロックで計算される．よって，式(1)，(2)の単精度浮動小数点演

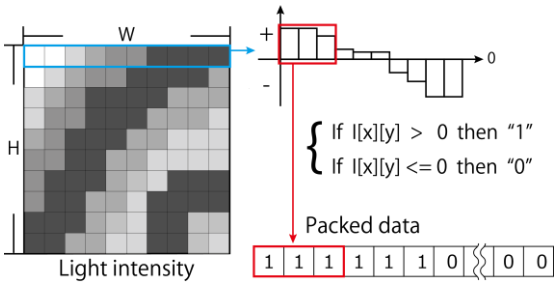

図 4　バイナリ CGH のパッキング処理

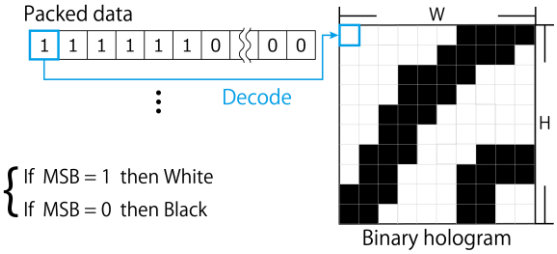

図 5　バイナリ CGH のアンパッキング処理

表 1　GeForce GTX 1080 Ti の仕様

GPU base clock	1,480 MHz
CUDA cores	3,584
Memory config	12 GB GDDR5
Memory clock	11.0 Gbps
Memory bandwidth	484.0 GB/s
Theoretical performance	10.61 TFLOPS

表 2　本システムに用いた PC の仕様

CPU	Intel Core i7 7800X (Clock speed: 3.5 GHz, 6-core, 12-thread)
Main memory	DDR4-2666 8 GB
Mother board	ASUS WS X299
OS	Linux (CentOS 7.3 x86_64)
Software	NVIDIA CUDA 10.0 SDK, OpenGL, MPICH 3.2

表 3　CGH 表示時間間隔

物体点数	CGH 表示時間間隔 T [ms]				
	1GPU	3GPUs	6GPUs	9GPUs	12GPUs
10,240	21.44	7.34	3.74	2.88	2.44
51,200	103.74	34.62	17.20	11.49	8.65
102,400	200.84	68.17	33.75	22.62	16.96
153,600	304.94	101.5	50.48	33.77	25.28
204,800	404.42	133.96	67.05	44.80	33.59

表 4　フレームレート

物体点数	フレームレート [fps]				
	1GPU	3GPUs	6GPUs	9GPUs	12GPUs
10,240	46.64	136.24	267.38	347.22	409.84
51,200	9.64	28.89	58.14	87.03	115.61
102,400	4.98	14.67	29.63	44.21	58.96
153,600	3.28	9.85	19.81	29.61	39.56
204,800	2.47	7.46	14.91	22.32	29.77

算数は，物体点数が 1 ($N_p = 1$)のとき 7 演算とし，式(1)の $j = 1 \sim N_p$に対する総和の演算数も考慮して実効性能を求めた．表 5 より，1 枚の GPU ボードで CGH 計算した場合，約 7.97 TFLOPS の演算速度を実現した．また，本手法により計算ノードの 12 枚の GPU ボードを用いたとき，約 95.90 TFLOPS の演算速度を達成した．表 1 の Theoretical performance より，理論性能に対して約 75%の性能を発揮していることがわかる．表 6 に 1 枚の GPU ボードで計算した場合に対する本手法による高速化を示す．計算ノードの 12 枚の GPU ボードを CGH 計算に使用した場合，1 枚の GPU ボードの速度に対して約 12 倍の高速化を実現している．表 7 に 1 個の CPU（Intel Core i7 7800X）による CGH 計算時間と本手法を用いた計算ノード 12 枚の GPU ボードによる CGH 計算時間との比較を示す．ここで，表 7 に示す CPU による CGH 計算時間には SLM への CGH 描画処理は含まない．CPU による CGH 計算プログラムは C 言語で作成し，Intel C コンパイラ v18.0.3（最適化オプション:-O3 -xCORE-AVX512 -mtune=skylake-x）を使用し，Open MP により 12 スレッドで計算した．本手法による計算ノードの 12 枚の GPU ボードを用いた CGH 計算は，CPU に比べ約 1519 倍の計算高速化を実現した．図 6 にパッキング処理（図 4）を行った場合と，パッキング処理を行わなかった場合の本システムにおける CGH 表示時間間隔の比較を示す．パッキング処理を行った場合，三次元物体を構成する物体点数に比例した CGH 表示時間間隔となっている．一方，パッキング処理を行わなかった場合，CGH 表示時間間隔はどの物体点数においても約 69 ms となっており，CGH の転送時間がボトルネックとなっていることが分かる．

表 5　実効性能

物体点数	実効性能 [TFLOPS]				
	1GPU	3GPUs	6GPUs	9GPUs	12GPUs
10,240	7.51	21.94	43.06	55.92	66.01
51,200	7.76	23.26	46.82	70.09	93.10
102,400	8.02	23.63	47.72	71.20	94.97
153,600	7.92	23.79	47.86	71.54	95.57
204,800	7.97	24.05	48.04	71.90	95.90

表 6　1 GPU に対する複数 GPU の CGH 計算高速化

物体点数	Speed Up (vs. 1 GPU)			
	3GPUs	6GPUs	9GPUs	12GPUs
10,240	2.92	5.73	7.44	8.79
51,200	3.00	6.03	9.03	11.99
102,400	2.95	5.95	8.88	11.84
153,600	3.00	6.04	9.03	12.06
204,800	3.02	6.03	9.03	12.04

表 7　CPU に対する本手法による CGH 計算時間の比較

物体点数	CGH 計算時間 [ms]		高速化率
	CPU	本手法	
10,240	2586.23	2.44	1059.93
51,200	12814.90	8.65	1481.49
102,400	25604.77	16.96	1509.72
153,600	38304.46	25.28	1515.21
204,800	51013.64	33.59	1518.72

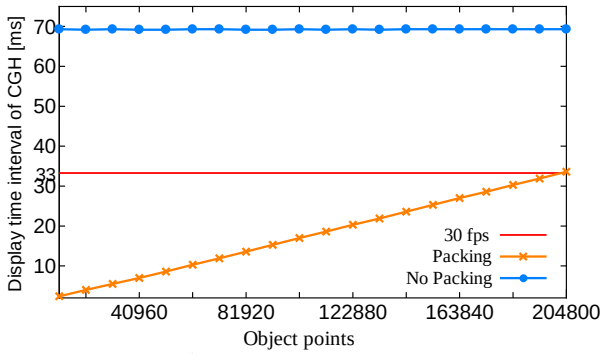

図 6　パッキング処理の有無による CGH 表示時間間隔

5. まとめ

PC 間で転送するデータ量を低減することで，汎用性の高いギガビット・イーサネットを搭載したシステムによる CGH 計算の高速化を提案し，CGH 計算の高速化を行った．12 枚の GPU（NVIDIA GeForce GTX 1080 Ti)を CGH 計算に用いて，約 20 万点からなる三次元物体の CGH を 30 fps で再生することに成功した．実効性能は約 95.90TFLOPS となり，理論性能に対して約 75%の性能を発揮した．また，CPU（Intel Core i7 7800X）に比べて約 1519 倍の計算高速化を実現した．

謝辞

本研究の一部は，日本学術振興会の科研費・基盤研究(C) (課題番号 18K11399) および公益財団法人電気通信普及財団平成 29 年度研究調査助成によって行なわれた．

参考文献

[1] T. Sugie, T. Akamatsu, T. Nishitsuji, R. Hirayama, N. Masuda, H.Nakayama, Y. Ichihashi, A. Shiraki, M. Oikawa, N. Takada, Y. Endo, T.Kakue, T. Shimobaba, T. Ito, "High-performance parallel computing for next-generation holographic imaging", Nature Electronics, 1, pp.254–259 (2018).

[2] NVIDIA, NVIDIA CUDA C Programming Guide ver.5.0, NVIDIA (2012).

[3] N. Masuda, T. Ito, T. Tanaka, A. Shiraki, and T. Sugie, “Computer generated holography using a graphics processing unit,” Opt. Express 14, pp.603- 608 (2006).

[4] H. Araki, N. Takada, S. Ikawa, H. Niwase, Y. Maeda, M. Fujiwara, H. Nakayama, M. Oikawa, T. Kakue, T. Shimobaba, and T. Ito, "Fast time-division color electroholography using a multiple-graphics processing unit cluster system with a single spatial light modulator", Chinese Optics Letters, Vol. 15 Issue 12, pp.120902-(2017).

[5] H. Araki, N. Takada, H. Niwase, S. Ikawa, M. Fujiwara, H. Nakayama, T. Kakue, T. Shimobaba, T. Ito, "Real-time time-division color electroholography using a single GPU and a USB module for synchronizing reference light," Applied Optics, Vol.54, Issue 34, pp.10029-10034 (2015).

[6] H. Niwase, N. Takada, H. Araki, H. Nakayama, A. Sugiyama, T. Kakue, T. Shimobaba, T. Ito, “Real-time spatiotemporal division multiplexing electro-holography with a single graphics processing unit utilizing movie features,” Optics Express, Vol. 22, Issue 23, pp.28052-28057 (2014).

[7] H. Niwase, N. Takada, H. Araki, Y. Maeda, M. Fujiwara, H. Nakayama, T. Kakue, T. Shimobaba, T. Ito, “Real-time electroholography using a multiple-graphics processing unit cluster system with a single spatial light modulator and the InfiniBand network,” Optical Engineering, Vol. 55, Issue 9, 093108 (2016).

[8] N. Takada, T. Shimobaba, H. Nakayama, A. Shiraki, N. Okada, M. Oikawa, N. Masuda and T. Ito, “Fast high-resolution computer-generated hologram computation using multiple graphics processing unit cluster system,” Applied Optics 51, pp. 7303-7307 (2012).