GPU クラウドにおけるコンテナ並列処理性能の評価

2013SE256 安田悠人　2013SE257 安田圭吾
指導教員：河野浩之

1　はじめに

NVIDIA クラウドコンピューティング[*1]によれば，GPU クラウドコンピューティングはディープラーニングによりビッグデータを分析，解析することでイノベーションを起こすための高度な計算処理能力を，CPU に比べはるかに少ないインスタンスでコストを削減して実現できる新時代の計算エンジンである．

本論文では，文献 [1] よりオーバーヘッドを抑えるために有効なコンテナ型仮想化基盤を用いる．また，文献 [2] では GPU を一般アプリケーションの高速化に利用する目的の研究が盛んに行われていること，文献 [3] ではコンテナ型仮想化基盤“Docker”[*2]のコンテナ上からホストのグラフィックボードに搭載された GPU にアクセスできることがそれぞれ述べられている．我々は，コンテナ型仮想化基盤において，GPU 並列処理性能の比較が先行研究で行われていないことに着目し，コンテナ上でベンチマークを並列実行する環境を構築して，GPU クラウドとしての可用性を評価する．

2　仮想化基盤の性能比較及び GPGPU に関する先行研究

本章の 2.1 節で仮想化基盤上とホストマシン上のコンピュータリソースの処理性能比較について述べ，2.2 節で Docker コンテナにおける GPU ベンチマークについて述べ，2.3 節で各論文の比較を行う．

2.1　仮想化基盤上とホストマシン上のコンピュータリソースの処理性能比較についての先行研究

文献 [1] では，ハイパーバイザ型仮想化基盤として KVM，コンテナ型仮想化基盤として LXC，Docker，新しいタイプの軽量な仮想化基盤としてクラウドコンピューティング専用設計の OSv を選び，4 つの異なる技術を CPU，Disk I/O などについて詳細な性能測定を行っている．コンテナ型の LXC や Docker は各測定においてオーバーヘッドはほぼ無視できるほどごくわずかであるという結果を得ている．

文献 [2] の仮想化基盤上における GPU パススルー測定に関する先行研究では，KVM，Xen，VMWare ESXi 及び LXC において，CUDA 並びに OpenCL のアプリケーションを使用して各仮想化基盤における GPU のパススルーを測定し比較している．Xen と VMWare では，仮想マシンへの GPU パススルーはベースシステムの 98〜100% のパフォーマンスを達成しており，さらに Linux コンテナである LXC や，ハイパーバイザ型である KVM も一貫してネイティブに近い結果を得ている．

2.2　Docker における GPU ベンチマーク関する先行研究

文献 [3] では，Docker コンテナにおいて，CUDA Toolkit や OpenCL などのアクセラレータライブラリを使用し，GPU への直接アクセスを実現することを可能にし，GPU のオーバーヘッドが極めて小さいことを示している．

マラリアの罹患率をモデル化する “SAMPO” シミュレーションでは，OpenCL API を用いてホストの GPU にアクセスしており，ほぼホストと同等のランタイムを実現している．また，行列積アルゴリズムである “SGEMM” でも，パフォーマンス損失が 0% となった結果を得ている．

2.3　先行研究の比較

表 1 に先行研究の比較を示す．

表 1　先行研究の比較

文献	実験内容	結果
[1]	KVM，LXC，Docker，OSv での CPU，Disk I/O などの性能測定	コンテナ型仮想化基盤 (LXC，Docker) の CPU オーバーヘッドは 1% 程度とごくわずか
[2]	KVM，Xen，VMWare ESXi，LXC での GPU パススルー測定	どの仮想化基盤も一貫してネイティブに近い結果を出す
[3]	Docker での GPU パススルー測定	パフォーマンス損失はほぼ 0

3　コンテナ上での GPU サポート

今回構成するコンテナ型仮想化基盤は，NVIDIA Docker を用いる．Docker は公開レジストリより，既存の Image をプルダウンし，ローカルホストで Docker を実行することができる．NVIDIA Docker は，nvidia-docker repository[*3] から導入し，コンテナを構築する．また，GPU で演算処理を実行するために “CUDA”[*4]を導入する．図 1 に NVIDIA Docker のアーキテクチャを示す．従来は Server 上のグラフィックボードに搭載された GPU を Host OS が認識し，演算処理するが，CUDA Driver をホストマシンに導入し，nvidia/cuda Image(CUDA Toolkit) を使用したコンテナを Docker Engine により展開することによりコンテナからホストマシンの GPU にアクセスできる．

[*1] http://www.nvidia.co.jp/object/gpu-cloud-computing-jp.html
[*2] https://www.docker.com/
[*3] https://github.com/NVIDIA/nvidia-docker
[*4] https://developer.nvidia.com/cuda-zone

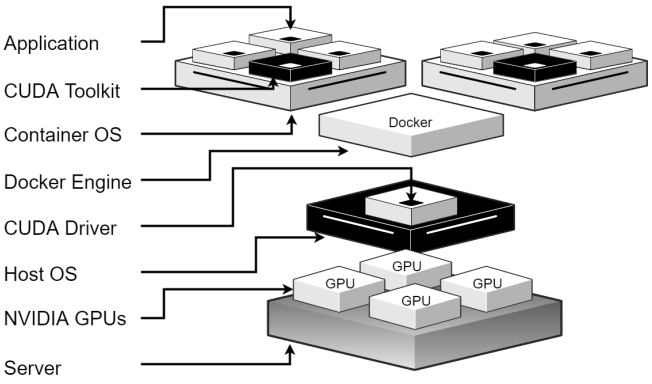

図 1　NVIDIA Docker のアーキテクチャ

また，今回使用するベンチマークツールは，文献 [4] において動作検証が行われているベンチマークプログラム “Fast N-Body Simulation with CUDA” を用い，GPU の処理性能を計測する．N-Body は，CUDA により実装された N 体シミュレーションである．N 体シミュレーションとは，重力多体系を N 個の質点で表現し，粒子間の相互作用重力を計算，運動方程式を時間積分し，系の時間発展を追うものである．

以上より，GPU における並列処理性能を評価するため，2.2 節でのネイティブ - コンテナ間でのパフォーマンス損失を NVIDIA Docker により確認する．さらに，実社会で GPU クラウドとしての活用を念頭に置き，複数のコンテナを起動，ベンチマークツールを並列実行した際の，GPU の処理性能をベンチマークする．

そこで，ネイティブ及び複数コンテナを同時に演算処理した場合のベンチマーク手法について示す．まず，ネイティブ - コンテナ間の GPU オーバーヘッドを確認するため，CUDA を導入したネイティブ環境上とコンテナ上でベンチマークする．次に複数コンテナを起動し，ベンチマークツールを並列実行し，GPU に負荷を掛けていく．その際，性能を比較するためにコンテナ数を変化させ，各コンテナをベンチマークする．

4　コンテナ並列ベンチマークの流れ

本実験における環境を，図 2，図 3 に示す．図 2 のようにネイティブ環境におけるベンチマーク環境を構築し，図 3 のようにコンテナ型仮想化基盤上におけるベンチマーク環境を構築する．

ベンチマークは，GTX750Ti，GTX960，GTX1060 の 3 つの GPU で行う．GPU の基本スペックは表 2 の通りである．N 体の数は 1.00×10^3，5.00×10^3，1.00×10^4，5.00×10^4，1.00×10^5 の 5 パターン，コンテナ数は 1〜5 に変化させて順に並列処理する．N 体の数を変化させるのは，“GPU Gems 3 - Chapter 31. Fast N-Body Simulation with CUDA”[*5] で N 体の数が少ないと安定な 1 秒間あたりの単精度浮動小数点演算回数 FLoating-point Operations Per Second[FLOPS] のスコアを出せないことを確認するためである．加えて，“GTX 1080 CUDA performance on Linux (Ubuntu 16.04) preliminary results (nbody and NAMD)”[*6] を参考に，2.56×10^5 でもベンチマークし，我々が実験を行う 5 パターンの N 体の数でも GPU が十分な処理性能を出せているか調べる．

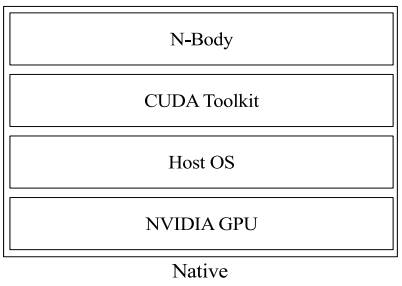

図 2　ネイティブ上のベンチマーク環境

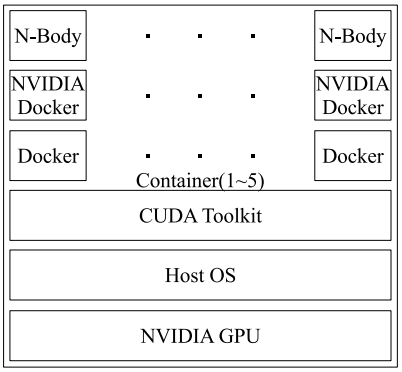

図 3　コンテナ内のベンチマーク環境

表 2　使用する GPU の基本スペック

	GTX750Ti	GTX960	GTX1060
Launch Price[USD]	149	199	249
TDP[W]	60	120	120
CUDA Cores	640	1024	1280

以降，表 3 のように条件 a〜f にそれぞれ N 体の数を指定し，ベンチマークを行う．ベンチマークは，各々 100 回ずつ実行したうち，完全に同じタイミングで開始できていないことを考慮し，はじめとおわりそれぞれ 10 回ずつを除いた 80 回のデータをベンチマークスコアに採用する．また，並列処理を行った場合は，すべてのコンテナからベンチマークログを回収し，平均を取ったものをベンチマー

[*5] http://http.developer.nvidia.com/GPUGems3/gpugems3_ch31.html

[*6] https://www.pugetsystems.com/labs/hpc/GTX-1080-CUDA-performance-on-Linux-Ubuntu-16-04-preliminary-results-nbody-and-NAMD-803/

クスコアに採用する．

本研究で使用した物理マシンの環境 (図 1 の Server) を表 4 に示す．3 台のグラフィックボードは同一筐体で差し替える．

表 3 実験パターン・条件

条件	N 体の数
a	1.00×10^3
b	5.00×10^3
c	1.00×10^4
d	5.00×10^4
e	1.00×10^5
f	2.56×10^5

表 4 物理マシンの環境

OS	Ubuntu 16.04.1 LTS 64bit
Linux Kernel Version	4.4.0-53-generic
CPU	Intel Core i5 6600K
GPU	表 2
RAM	16GB DDR4
Storage	128GB SSD
Docker Version	1.12.3
CUDA Version	8.0.44

5 ベンチマーク結果

本章の 5.1 節で N 体の数による GPU 処理性能について述べ，5.2 節で GPU の安定な処理性能でのネイティブ - コンテナ間のオーバーヘッドについて述べ，5.3 節で GPU の処理性能が安定な条件においてコンテナ数を変化させた際の性能比較について述べ，5.4 節で GTX1060 における N 体の数を条件 a〜e でコンテナ並列処理した際の性能比較について述べる．

5.1 N 体の数による GPU 処理性能の比較

図 4 はネイティブ環境での N 体の数による GPU 処理性能の変化についてのグラフである．いずれの GPU も N 体の数が少ない条件 a では十分な処理性能を出せていない．その一方で，GTX750Ti では条件 d〜f で，GTX960 では条件 d〜f で，GTX1060 では条件 e〜f で安定な 1 秒間あたりの単精度浮動小数点演算回数 FLoating-point Operations Per Second[FLOPS] のスコアを出せている．

表 5 にベンチマークで得られたデータの平均値 (図 4)± 標準偏差 (%) を示し，表 6 に GTX750Ti，GTX960，GTX1060 における条件 f と比べたときの条件 a〜e の処理性能を示す．条件 e では，条件 f の値に近似している．また，GTX1060 では，条件 c は条件 e と比べ 85.742% の処理性能を得ている．

以降，条件 e で GPU が十分な処理性能を引き出していることとみなす．

表 5 ネイティブ環境のベンチマークスコアの標準偏差

条件	GTX750Ti	GTX960	GTX1060
a	$339.455 \pm 0.390\%$	$384.727 \pm 0.139\%$	$318.589 \pm 0.133\%$
b	$821.726 \pm 1.500\%$	$1349.980 \pm 2.938\%$	$1581.828 \pm 0.055\%$
c	$834.169 \pm 0.832\%$	$1659.217 \pm 0.134\%$	$2463.018 \pm 0.669\%$
d	$886.222 \pm 0.479\%$	$1794.073 \pm 0.309\%$	$2781.246 \pm 0.530\%$
e	$892.935 \pm 0.227\%$	$1823.246 \pm 0.478\%$	$2872.596 \pm 0.559\%$
f	$892.847 \pm 0.100\%$	$1826.299 \pm 0.358\%$	$2856.118 \pm 0.329\%$

表 6 条件 f と比べたときの条件 a〜e の性能比較

条件	GTX750Ti	GTX960	GTX1060
a	38.019%	21.066%	11.155%
b	92.034%	73.919%	55.384%
c	93.428%	90.085%	86.237%
d	99.258%	98.235%	97.379%
e	100.001%	99.833%	100.577%

5.2 条件 e によるネイティブ - コンテナ間のオーバーヘッド

ここで，条件 e においてネイティブ - コンテナ間のオーバーヘッドを NVIDIA Docker を用いて確認し，表 7 に示す．2.2 節の GPU パススルー測定に関する先行研究から得られる，パフォーマンス損失がほぼ 0 である結論を我々の実験環境でも裏付けることができた．

表 7 ネイティブ - コンテナ間のオーバーヘッド

	GTX750Ti	GTX960	GTX1060
オーバーヘッド	99.918%	99.817%	99.542%

5.3 条件 e によるコンテナ数の性能比較

図 5 は条件 e におけるコンテナ数を 1〜5 に変化させて並列処理した際の性能変化についてのグラフである．GTX1060 を例に取ると，コンテナ数 2〜5 でスコアはそれぞれ 53.026%，34.275%，26.677%，20.917% となり，GPU が上限の性能を出しているような条件下では，コンテナ数とベンチマークスコアが反比例することが読み取れる．

5.4 GTX1060 におけるコンテナ並列処理の性能比較

図 6 は GTX1060 において条件 a〜e，コンテナ数 1〜5 で並列処理した際の性能変化のグラフであり，表 8 に GTX1060 において条件 a〜e によるコンテナ数 1 と比べたときのコンテナ数 2〜5 の GPU 処理性能を示す．

表 8 GTX1060 のコンテナ並列処理の性能比較

コンテナ数	条件 a	条件 b	条件 c	条件 d	条件 e
2	99.383%	97.393%	94.522%	66.684%	53.026%
3	98.821%	95.759%	92.795%	42.113%	34.275%
4	98.890%	95.247%	92.837%	30.478%	26.677%
5	99.785%	94.714%	91.112%	26.480%	20.917%

条件 a〜c の場合ではコンテナ数を増加させても GPU 処理性能がほぼ低下しないことが確認できる．

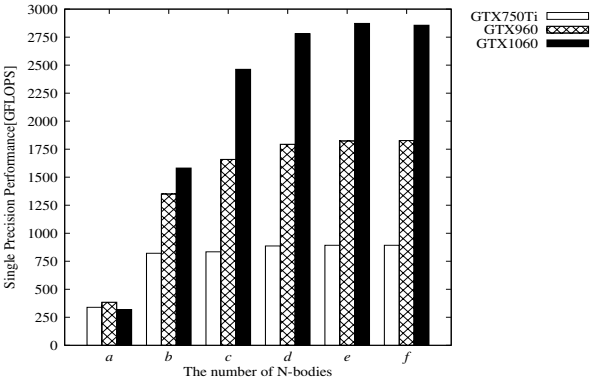

図 4 N 体の数による GPU 処理性能の変化

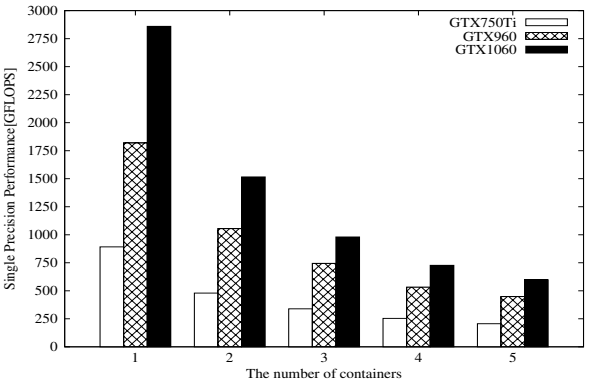

図 5 条件 e によるコンテナ並列処理の性能変化

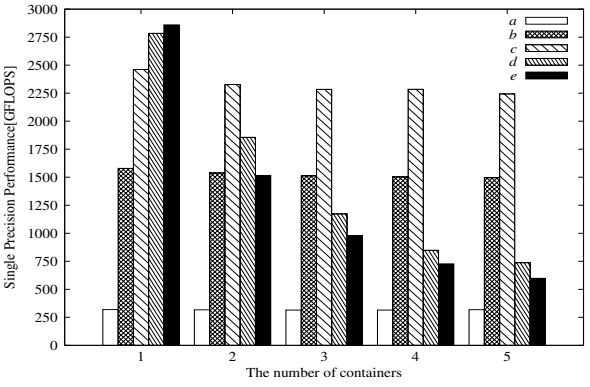

図 6 GTX1060 のコンテナ並列処理の性能変化

6 考察

本研究では，GPU クラウド環境を想定したコンテナの並列処理性能を NVIDIA Docker を使用し，ベンチマークした．その結果，GTX750Ti のコンテナ数 1 比のベンチマークスコアは，コンテナ数 5 の場合，条件 a〜e で 100.046%，89.245%，97.366%，25.882%，23.081% となった．また，GTX1060 の GTX750Ti 比のベンチマークスコアは，コンテナ数 5 の場合，条件 a〜e で 93.559%，204.908%，276.234%，321.021%，290.447% となった．

そして，5.4 節より条件 a〜c の場合，GPU クラウドを構築することができるという結果を得ることができた．とりわけ，5.1 節及び 5.4 節からは，GTX1060 で条件 c コンテナ数 5 の場合，条件 e コンテナ数 1 と比べ，78.443% の処理性能を得ていることが分かった．

一方で，5.4 節より条件 d〜e の場合にベンチマークスコアが低下することに関して，GPU は 1 つの命令で複数のデータを処理する SIMD(Single Instruction Multiple Data) 形式のため，GPU のコア数が表 2 のように CPU に比べ圧倒的に多いが，GPU の演算処理特性上 CPU のような並列処理効果を得られず，図 6 及び表 8 のような結果となったのではないかと我々は考える．

7 むすび

本研究より，GPU クラウドで様々なプログラムが混在している環境において，条件を満たせば十分なコンテナ並列処理性能が得られることが明らかになったが，GPU の演算処理特性を理解し，性能をより引き出せるような並列処理手法を検討する必要がある．今後の課題として，CPU との処理性能の比較やヘテロジニアスな環境における処理性能評価を行うことが挙げられる．また，Docker マルチホストネットワーク機能を活用した，ハードウェアリソースの有効活用を検討することも有意義であると考える．

参考文献

[1] R. Morabito, J. Kjllman, M. Komu, “Hypervisors vs. Lightweight Virtualization: a Performance Comparison,”In Proceedings of IEEE International Conference on Cloud Engineering, pp.386-393, 2015.

[2] J. P. Walters, A. J. Younge, D. I. Kang, K. T. Yao, M. Kang, S. P. Crago, G. C. Fox,“GPU Passthrough Performance: A Comparison of KVM, Xen, VMWare ESXi, and LXC for CUDA and OpenCL Applications,” In Proceedings of IEEE International Conference on Cloud Computing, pp.636-643, 2014.

[3] N. Haydel, S. Gesing, I. Taylor, G. Madey, A. Dakkak, S. G. D. Gonzalo, “Enhancing the Usability and Utilization of Accelerated Architectures via Docker,”In Proceedings of IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp.361-367, 2015.

[4] 株式会社 エルザジャパン& HPC クラウド事業部, “NEC 社製サーバー「Express5800/R120d-2M」と NextIO 社製「vCORE Express 2090」の動作検証報告書,” ELSA Japan Inc., https://www.express.nec.co.jp/linux/distributions/confirm/ihv/R120d-2M_vCORE_Express2090.pdf, 2012.