jh220046

次世代演算加速装置とそのファイル IO に関する研究

塙 敏博（東京大学）

概要

GPU クラスタにおいて，実アプリケーションを効率よく実行するためには GPU に対するデータの入出力の考慮が必要であり，データ転送と演算のオーバラップ，転送レイテンシの短縮を工夫する必要がある．そこで本研究では，GPU 上データの直接ファイル IO である GPUDirect Storage (GDS)，あるいは計算とファイル IO のオーバラップの両者を容易に取り扱い可能にする手法を確立し，様々なファイル入出力特性を持つ実アプリケーションにおいて GPU-ファイル IO 間の処理を効率化することを目的とする．さらにファイル IO を考慮したベンチマークを実装する．今年度は，N 体シミュレーション，都市気候シミュレーション，機械学習トレーニングのそれぞれについて GDS の利用について実装検討，一部については予備評価を行った．その結果，GDS は多くの場合に高い性能を示し，mdx では VM 環境であっても GDS が非 GDS に対して最大 1.67 倍の性能を示した．

1 共同研究に関する情報

1.1 共同研究を実施した拠点名

- 東京大学 情報基盤センター
- 名古屋大学 情報基盤センター
- mdx

1.2 課題分野

- 大規模計算科学課題分野

1.3 共同研究分野 (HPCI 資源利用課題のみ)

- 超大規模情報システム関連研究分野

1.4 参加研究者の役割分担

塙 敏博 (東大・代表): 全体取りまとめ，GPU 通信，機械学習
建部 修見 (筑波大・副代表): ストレージ技術，機械学習
三木 洋平 (東大): 宇宙物理コード
佐藤 拓人 (筑波大): City-LES コード
星野 哲也 (東大 → 名古屋大): GPU プログラミング
河合 直聡 (東大): 数値アルゴリズム
中島 研吾 (東大): 計算科学アプリケーション
住元 真司 (東大): ストレージ技術
小林 諒平 (筑波大): GPU 通信，IO
藤田 典久 (筑波大): GPU 通信，IO
朴 泰祐 (筑波大): GPU 通信，IO
平賀 弘平 (筑波大): ストレージ技術、機械学習
小山 創平 (筑波大): ストレージ技術、機械学習

2 研究の目的と意義

HPC システムにおいて，アクセラレータとして GPU が広く利用されているが，実アプリケーションでは本質的に GPU に対するデータの入出力の考慮が必要であり，データ転送

と演算のオーバラップ，転送レイテンシの短縮を工夫する必要がある．近年 NVIDIA によって，GPUDirect RDMA をファイル IO に拡張した GPUDirect Storage (GDS) が提供されるようになり、IO オーバヘッドの短縮が期待される．本研究は，GPU 上データの直接ファイル IO，あるいは計算とファイル IO のオーバラップの両者を容易に取り扱い可能にする手法を確立し，様々なファイル入出力特性を持つ実アプリケーションにおいて GPU-ファイル IO 間の処理を効率化することを目的とする．さらにこの成果を，最先端共同 HPC 基盤施設 (JCAHPC) の次期システム “Oakforest-PACS II (OFP-II)” の設計に反映し，OFP-II および次世代 GPU クラスタにおいて利用可能な，ファイル IO を考慮したベンチマークを実装する．

3 当拠点公募型研究として実施した意義

本研究では，計算科学の実アプリにおけるファイル IO の観点で，宇宙物理コードの GOTHIC，気象コードの City-LES を対象にし，各分野の専門家を交えて研究を実施した．さらに PyTorch などの機械学習フレームワークも対象にしている．本研究で利用した NVIDIA A100 GPU は，Wisteria/BDEC-01 Aquarius，mdx で採用されている．当初は，Wisteria/BDEC-01 Aquarius および不老 Type II において GDS の利用を想定していたが，GPU ドライバやストレージの制約等により実現できなかった．一方で，mdx においては，VM を採用しており，ハードウェアとしては抽象化され，必ずしもデバイスに対して最適な構成であるか不可視な部分はあるものの，必要に応じた OS や所望のドライバ構成を取ることができ，実際に性能の改善が見られることも確認できた．本研究では，別途 AMD EPYC Milan CPU, NVIDIA A100 80GB PCIe GPU を搭載する GPU サーバ，および筑波大学計算科学研究センターに導入された，Intel Sapphire Rapids CPU, NVIDIA H100 80GB PCIe GPU を搭載する Pegasus システムも使用して実施している．

4 前年度までに得られた研究成果の概要

該当なし

5 今年度の研究成果の詳細

5.1 N 体シミュレーション

今年度は，GPU 向けに最適化された重力ツリーコード `GOTHIC` (Miki & Umemura 2017; Miki 2019) に，GDS を用いたファイル出力機能を組み込み，銀河衝突シミュレーションにおける性能評価を実施した．HDF5 13.x において導入された Virtual File Driver（VFD）を利用する HDF5 GDS VFD[*1]を東京大学情報基盤センターが保有する実験システムに導入し，`GOTHIC` からは HDF5 経由で GDS を利用するよう実装した．通常の実装（Listing 1）との違いは，HDF5 ファイルをオープンする際に GDS を利用するための適切なプロパティを指定するということ（Listing 2）であり，これによって GPU 上のアドレス空間を直接 HDF5 の write 関数に指定することで GDS 機能を用いたファイル出力が可能となる．

アンドロメダ銀河周辺の構造をよく再現できる銀河衝突シミュレーション（Kirihara, Miki et al. 2017，全粒子数 16 777 216，約 18 万ステップ）において，GDS を利用して 1024 粒子分のデータだけを全タイムステップ出力する

[*1] `https://github.com/hpc-io/vfd-gds`

Listing 1　通常の HDF5 ファイルのオープン方法.

```
target = H5Fcreate(filename, H5F_ACC_TRUNC, H5P_DEFAULT,
    H5P_DEFAULT);
```

Listing 2　GDS を有効にした HDF5 ファイルのオープン方法.

```
#include "H5FDgds.h"

fapl = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fapl_gds(fapl, MBOUNDARY_DEF, FBSIZE_DEF, CBSIZE_DEF);
target = H5Fcreate(filename, H5F_ACC_TRUNC, H5P_DEFAULT, fapl);
```

という実験を行った．この際に，各ステップにおけるデータを個別のファイルに出力すると総ファイル数が膨大となるため，全体のスナップショットファイル出力（数千ステップに一度の頻度，全 400 ファイル）ごとに一つのファイルに書き込むこととし，HDF5 のグループ機能を用いて各タイムステップのデータが容易に取り出せるような実装としている．

使用した GPU は NVIDIA A100 80GB PCIe であり，データ出力先は KIOXIA CM6-V 3.2TB KCM61VUL3T20 である．ソフトウェア環境としては，CUDA 11.7, HDF5 1.13.1, HDF5 Nvidia GPUDirect Storage VFD 1.0.1 を用いた．図 1 に，1024 粒子分のデータ出力に GDS を用いた場合のデータ出力時間を赤点で，用いなかった場合のデータ出力時間を黒点で示した．データ出力時間はステップごとに大きく変動するものの，GDS の使用によって概ね 2 倍程度高速化されている．GOTHIC は block timestep を採用した重力ツリーコードであるため，重力計算に要する時間はステップごとに 2 桁程度変動する．今回の実験では全ステップで出力データサイズは一定としたが，各ステップで実行された重力計算による影響で，ステップごとのデータ出力時間も変動した可能性がある．また，はじめの 1-2 万ステップ程度に対してそれ以降のステップにおいては時間刻みの階層数が増加しており，こうした重力計算の挙動の変化が図 1 における振る舞いと関連している可能性がある．

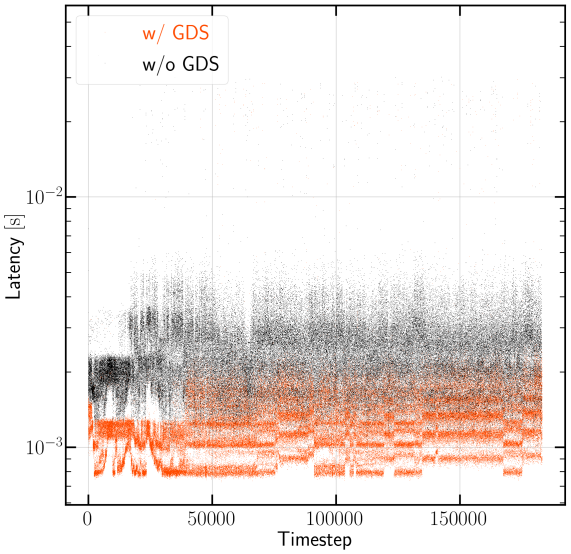

図 1　GDS 使用の有無によるデータ出力時間の比較．データ出力に要する時間をステップごとに示した．赤点が GDS 使用時，黒点が GDS 不要時のデータ出力時間であり，GDS の使用によって 2 倍程度高速になっている．

5.2 都市気象シミュレーション

今年度は, City-LES の最適化・GDS への適用を見越して, City-LES の流体計算の一部と出力部を取り出したベンチマークソフトの開発を行った．従来 City-LES に関するベンチマークソフトは City-LES の出力パターンを模したデータ出力部のみを取り出したものしかなく, 計算と出力のバランスや計算と出力のオーバーラップの効果を簡易的に測定できるものがなかった．本年度開発したベンチマークは, City-LES の気流計算部のうち移流拡散過程や Poisson 方程式の求解部といった基本的な部分と, 計算結果の出力部のみを切り出したコードである．これを用いることで, 本研究で実装を試みる最適化の効果のより効率的な議論が期待できる．

このベンチマークソフトをベースに, 出力のノンブロッキング化の方法を検討した．City-LES の出力は NetCDF ファイルであり, NetCDF-4 ライブラリを使用している．NetCDF-4 ライブラリにはノンブロッキング I/O が実装されていないため, 計算と出力のオーバーラップが困難であった．そこで, 使用するライブラリをノンブロッキング I/O の API が実装されている Parallel NetCDF に変更し, 出力をノンブロッキング API (nfmpi_iput_vara_*) に置き換えた．これによって, 出力するすべての変数の書き出しを非同期に行うことが可能となった．しかし, この API を用いる場合, 実際の書き出しは nfmpi_wait で一括して行われることがわかった．単純に出力部のライブラリを置き換えるだけでは, 出力のオペレーションの数は減らせるが, 計算と出力のオーバーラップは実現できず, wait 部で出力待ちが発生してしまう．これを防ぐために, OpenMP を用いて出力用のスレッドを生成しスレッド並列化して出力をオーバーラップすることとした．この実装では, City-LES 本体が MPI と OpenMP のハイブリッド並列コードで多数のモジュールからなるコードである点を考慮し, 流体計算部を可能な限り改変しないようにした．そのため, 入れ子並列処理を用いて, 全体を計算スレッド・出力スレッドの 2 スレッド並列に, 計算スレッド内を従来の実装通り, 計算開始時に指定した数でスレッド並列するようにした (Listing 3)．ベンチマークソフトへの実装に先立ち, 単純化したテストプログラムで Listing 3 の実装を行い, 出力部が隠蔽できることを確認した．ただし, ベンチマークプログラムに同様の機能を実装した場合, 流体計算部と出力部の両スレッドで MPI 通信が発行されるため, その管理などのための一部の変更を追加する必要がある．

5.3 機械学習トレーニング

今年度は，機械学習のフレームワークにおける GDS の適用状況について調査および環境構築を行った．

NVIDIA によって公開されている Data Loading Library (DALI: `https://github.com/NVIDIA/DALI`) は、データの読み込みおよび前処理を行うライブラリであり，画像、ビデオ、オーディオ等のデータセットを効率よく行う機能を提供する．PyTorch, Tensorflow など機械学習向けのフレームワークでは，データローダやデータイテレータを備えており, 通常, データロードとそれに伴うデコードや画像の整形などの処理は CPU で行われる．DALI はこれらの機能をオーバーライドして，GPU 上にオフロードする．さらに、DALI は、入力パイプラインのスループットを向上させるため，プリフェッチ、並列実行、バッチ処理などの機能を持ち，ユーザからは透過的に処理が行われる．DALI は TensorFlow, PyTorch, MXNet, PaddlePaddle をサポートしており，学習や推

Listing 3　テストプログラムにおける出力オーバーラップの実装例.

```
io_flag = 0
!$OMP parallel, num_threads( 2 )
 do n = total timesteps
   !$OMP single
   if(io_flag /= 2)then
     !$OMP parallel, num_threads( N )
     *Fluid simulation part*
     N1 = n
   end if
   !$OMP end single nowait
   if(io_flag == 0) then
      !$OMP barrier
   end if
   !$OMP sections firstprivate(data)
     !$OMP section
     if(n=output_step) then
         *output part*
         io_flag = 2
         !$ call omp_set_lock(lock1)
         end if
      if(io_flag == 2 .and. n == N1)then
         io_flag = 0
         !$ call omp_unset_lock(lock1)
      end if
   !$OMP end sections nowait
end do
```

論のワークフローに対して高い移植性を備えている．この中で，GDS は DALI のデータローダにおいて用いられている．

機械学習ベンチマーク MLPerf において，Training のうち，画像分類，物体検出，音声認識など一部のベンチマークについては DALI がすでに組み込まれていることを確認している．

一方，Kvikio (`https://github.com/rapidsai/kvikio`) は，NVIDIA によって提供されているライブラリであり，cuFile によって提供される GDS の機能を，C++ や Python から容易に使えるようにしたものである．Kvikio では，ホストおよび GPU から，GDS の利用可能の有無に関わらずシームレスに動作する機構を提供している．DALI とは独立に開発されているものと考えられ，両者を組み合わせてより高い性能を実現できる可能性がある．

一方で，GDS 基本性能を評価する目的で環境構築を実施し，mdx の VM 上で GDS 実験環境を構築した．VM の構成は図 2 に示すように，mdx GPU ノードのうち，1GPU, 18

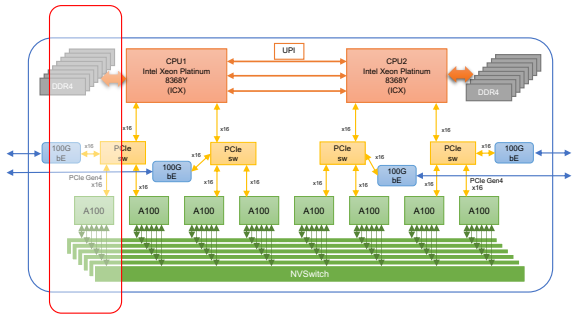

図 2　mdx における GDS 環境

vCPUs であり，OS は Ubuntu 20.04.5 を用いた．GPU は PCIe pass through により接続されており，CUDA 11.8 の環境を用いた．また，ストレージへの接続は SR-IOV により 100Gbit Ethernet で接続され，Lustre ドライバには lustre-2.14.0_ddn50 を用いた．従って，VM の環境ではあるが，GPU およびネットワークの IO インタフェースについては，ベアメタルのように直接ハードウェアをアクセスできる構成である．

NVMe SSD による Lustre FS (mdx では /fast) について gdsio ベンチマークによる測定を行った．Lustre 上ではファイルに対して最大の striping をセットし，ファイルサイズ 10GB, iosize 1MB, 3 回の試行のうち，最も良い結果を採用した．結果を表 1 に示す．

評価の結果，GDS の場合が非 GDS（CPU への転送を行う）場合に比べていずれも高い性能を示した．18 スレッドを使った write の場合で最大 1.7 倍の高い性能を示した．

表 1　gdsio による性能

スレッド数	mode	Read (GiB/s)	Write (GiB/s)
1	nonGDS	0.77	0.80
	GDS	0.92	0.92
2	nonGDS	1.46	1.54
	GDS	1.72	1.80
4	nonGDS	2.90	3.02
	GDS	3.51	3.61
8	nonGDS	5.31	4.62
	GDS	6.22	6.18
16	nonGDS	6.57	5.45
	GDS	8.98	8.39
18	nonGDS	6.53	5.75
	GDS	8.91	9.61

6　今年度の進捗状況と今後の展望

6.1　N 体シミュレーション

今年度目標としていた内容である (1) GDS を用いての GPU からの直接データ出力，(2) HDF5 の機能を用いてのファイル数削減については実装および動作試験が完了した．また，データを超高頻度に出力しても N 体シミュレーションの性能が極端に低下することはないことが分かった．

一方で，今年度使用していた Wisteria-Aquarius や不老 Type II といったスパコン上においては，OS や各種ドライバのバージョンなどの制約から GDS が動作可能とならなかったため，スパコン上での動作試験は実施できなかった．こうしたスパコン上での動作試験および性能評価は，次年度以降に取り組んでいく．

6.2　都市気象シミュレーション

今年度は, City-LES の性能測定・出力部の改良のためのベンチマークソフトの実装を行なった．また, 出力の API 変更やマルチスレッド化による計算 – 出力のオーバーラップについて検討した．その結果, API 変更による出力部の挙動の変化や, テストプログラムにおけるマルチスレッド化による出力の隠蔽が実現できることも確認できた．今後は, テストプログラム

で動作が確認できた出力の隠蔽をベンチマークプログラムへの実装を継続予定である．また, task 機能を用いたマルチスレッド化も検討する予定である．さらには, City-LES はすべての流体計算部が OpenACC によって GPU 対応済であるので, 流体計算を GPU で, 出力を CPU で行い両者をオーバーラップさせる実装にも取り組む予定である．これら 3 パターンのオーバーラップ法の性能を相互比較し, 実アプリに実装するのに最適な方法を検討する．

6.3 機械学習トレーニング

今年度は，機械学習フレームワークにおける GDS の利用方法について調査および環境構築を行った．DALI は MLPerf における Training のリファレンス実装として用いられている TensorFlow, PyTorch, MXNet のいずれもサポートしており，画像分類，物体検出，音声認識のベンチマークはすでに DALI に対応した実装が存在している．一方，kvikio を使うことで性能の改善が見られる場合もあり，詳細な性能評価については来年度実施予定である．

7 研究業績一覧（発表予定も含む）

国内会議発表 （査読なし）

- 富永 瑞己, 塙 敏博, 三木 洋平, 「GPU クラスタにおけるファイル IO 性能評価」、情報処理学会ハイパフォーマンスコンピューティング研究会, 2023 年 8 月 (発表予定)