KIOXIA

White Paper

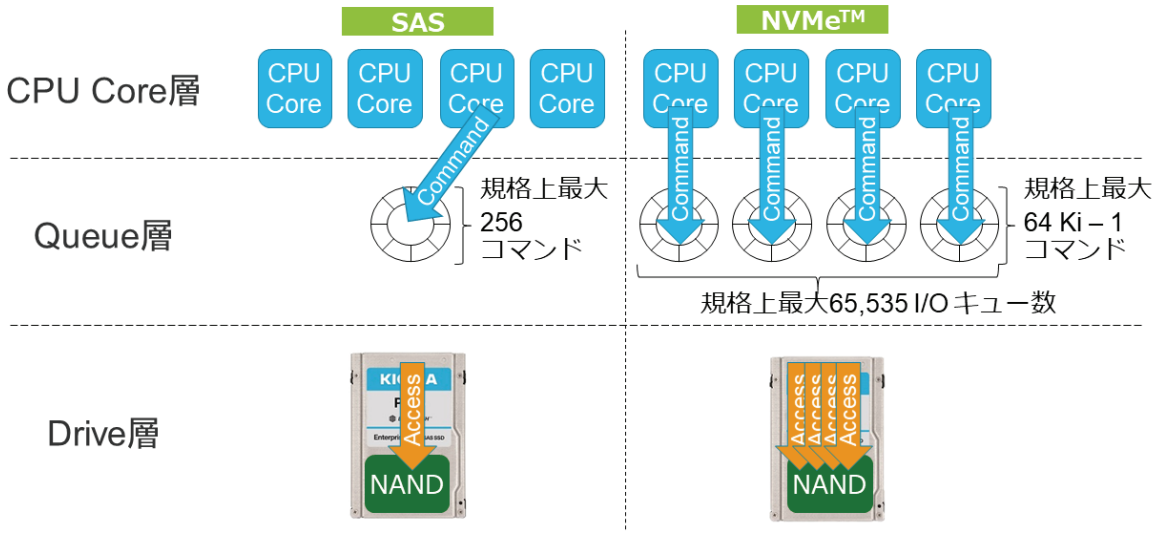
# GPUDirect Storageの性能検証とNVMe-oF™の有効性について

KIOXIAは、「記憶」で世界をおもしろくするというミッションのもと、様々な記憶媒体ストレージを提供しています。ストレージとAIのどこに関係があるのかと思われるかもしれませんが、弊社は高速なストレージがAIに貢献できると考え、これまでさまざまなコラボレーションをしてきました。例えば、AIによる漫画の執筆やクイズの自動生成などです。

いっぽう、弊社が扱う製品としては、部品としてさまざまな製品に組み込まれるフラッシュメモリー、PCやサーバーなどで用いられるSSD、家電などで用いられるメモリーカード、USBドライブなどがあります。これらの製品や技術から、AI、機械学習の高速化に関連するNVMe™ SSDの技術についてご紹介します。

まず、従来のSCSI SASとNVMe™の違いを説明します。左のSASの方式では、一つのI/Oキューに対して入出力(IO)の要求を出せるようになっています。これは、HDDのようなドライブが主な時代は十分だったのですが、NANDフラッシュメモリーが記憶メディアとして使われるようになると、フラッシュメモリーが発揮できる性能を十二分に活かしきれないようになってきました。
それに対して定義された規格が、右側に記載したNVMe™の方式です。これにより、複数のqueueから並列にI/O要求を出し、記憶メディアにも並列でアクセスできるようになりました。このように並列度を高めることで、NANDフラッシュメモリーの性能を生かせるようになっています。

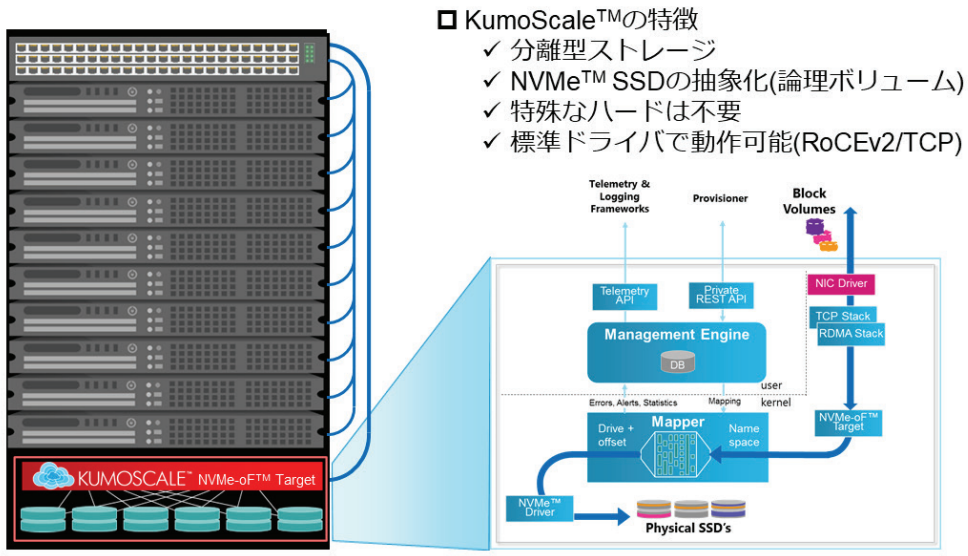
**NVMe™ SSDのメリット**




このNVMe™ SSDは、高速なローカルストレージとして活用するのが一般的です。いっぽう、今後も標準化団体によって更なる高性能化が見込まれるNVMe™ SSDを最大限ご活用いただくため、KIOXIAではKumoScale™ソフトウェアを開発しました。次に、KumoScale™について紹介します。

KumoScale™は、NVMe-oF™によるストレージ・ディスクアグリゲーションを提供するソフトウェアの製品です。KumoScale™は、高速なデータパスのみではなく、管理機能をはじめ、テレメトリ連携、オーケストレーションツールとの連携なども提供します。またストレージサーバとして構築・運用するにあたり、特殊なハードウェアやソフトウェアは必要なく、NICとLINUX™のインボックス・ドライバーがあれば、KumoScale™をNVMe-oF™ストレージとして使うことができます。

### NVMe-oF™ターゲットソフトウェア: KumoScale™の製品戦略



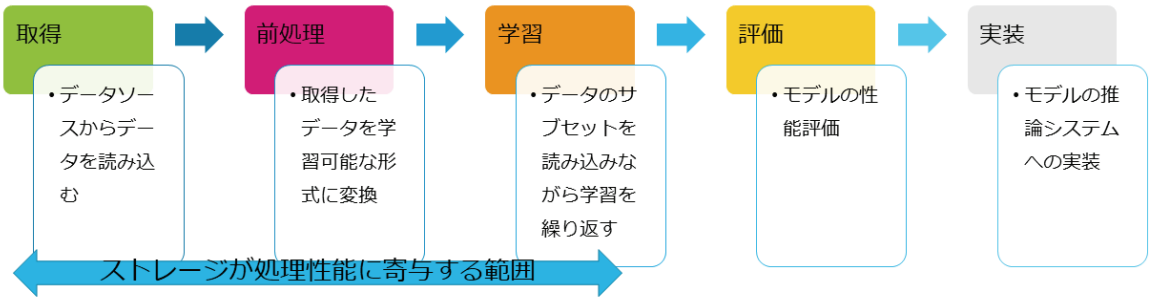

さて、先に申し上げましたように、KIOXIAはAI、機械学習に高速ストレージの製品・技術で貢献できると考えていますが、まだ一般にはそういった環境を実際に構築し運用するには、情報が不足し実績が少ないと思われます。

### KIOXIAが考える従来の機械学習環境の課題

**高性能GPUと高速ストレージの組み合わせは容易に構築できるがそれが認知されていない**

**高速ネットワークストレージの構築は容易になってきている**

• これまで高価な専用ストレージシステムが必要であった大規模・高速GPUアプリケーションのワークフローが、NVMe™/NVMe-oF™を用いることでスモールスタートが可能となった。

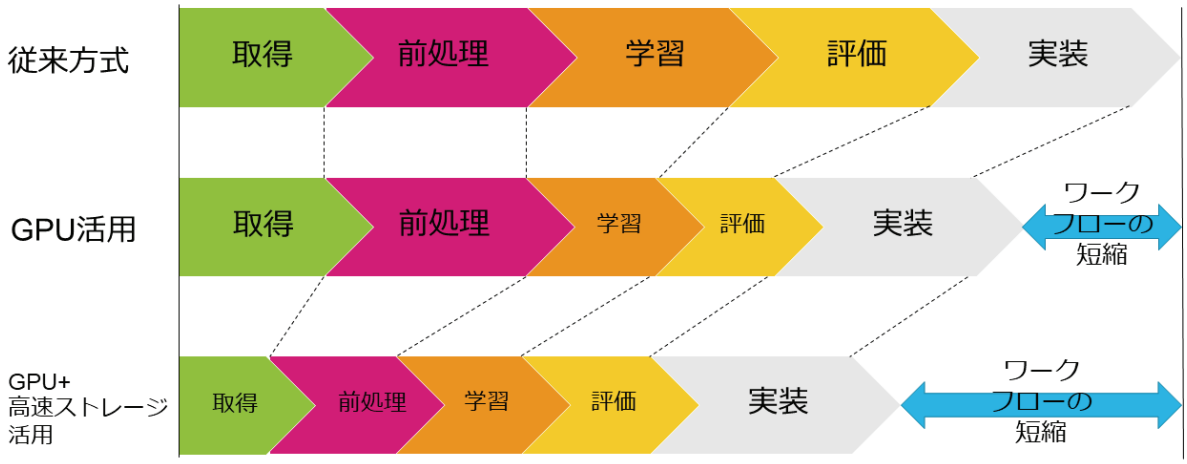



本稿では、取得、前処理、学習、評価、実装のワークフローの中で、ストレージの処理性能が、ワークフローの処理時間を短縮することができると考え、機械学習のワークフローの中で高速ストレージの製品・技術を利用するメリットを検証しました。次のスライドで具体的に例を示します。

この図では、ワークフロー中の各処理時間を矢印の長さで表現しています。一番上のCPUを用いた機械学習ワークフローを基準とすると、中段のケース、GPUを用いたケースでは、学習、評価の処理時間を短くすることができます。さらに高速ストレージを用いることで、下段のように取得や前処理の時間短縮に貢献できると考えています。

**高速ストレージの活用で期待される効果: ワークフロー処理時間の比較**

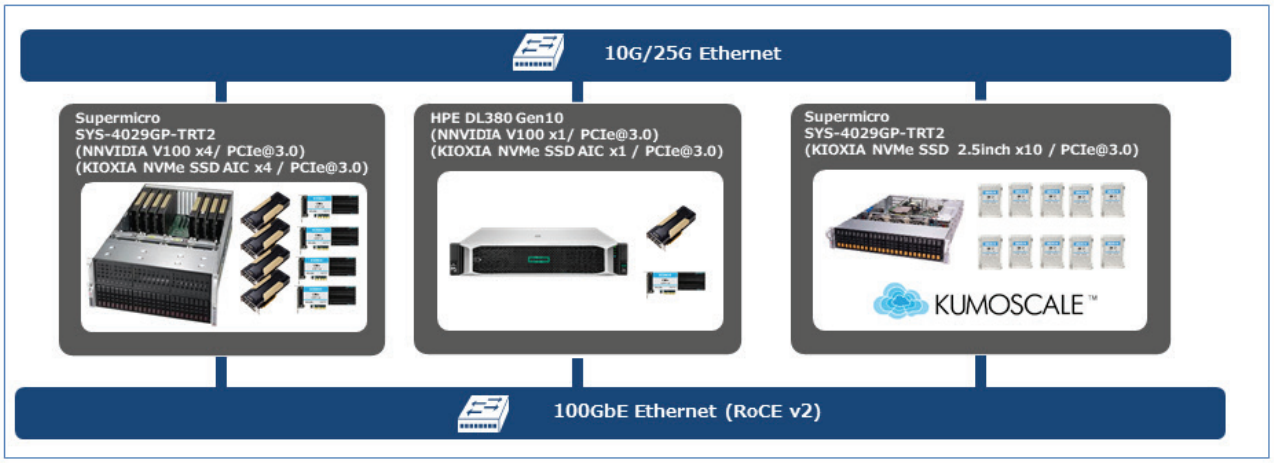
• GPUおよび高速ストレージを活用することで、機械学習ワークフローの広い範囲での改善が可能となる。




ところが、実際には、自社内に機械学習環境でのストレージ活用の知見が少なく、高速ストレージを用いたワークフロー改善などの効果測定ができていませんでした。

さらに、近年GPUダイレクト技術の一つGPUDirect Storageの登場により、更なる機械学習ワークフローの改善が期待されます。

今回検証した環境のリソースの構成の概要がこちらです。高スペックなGPUサーバーにローカルのNVMe™を搭載。RoCE v2に対応した100GbE Ethernetを接続しローカルNVMe™とKumoScale™を用いたNVMe-oF™でのNVIDIA GPUDirect Storageの性能検証を行いました。




次に、GPUDirect Storage環境構築の要点について説明します。

検証を開始した当初から、GPUDirect Storage環境構築事例はかなり限られていました。NVMe™を用いた測定方法を参照したくても、当時はほとんど情報がありませんでした。また、必要な構成に関しても、GPUを使うということ以外に不明な点が多々ありました。ベンチマーク実行方法に関しても、どういったツールがあるか、どうすれば効果を示せるかという点で理解ができていませんでした。

構築にあたって、我々の経験から留意点をご紹介します。まずGPUサーバーに関しては、PCIe®の内部トポロジーの理解が不可欠です。構成を誤ると、同じサーバーでも性能が通常のI/O方式よりも悪くなるということがありました。また、PCIe®が関係するということもあり、BIOSでの設定も関わってきます。
そして、ソフトウェアの構成の組み合わせの考慮も重要です。各機能階層ごとのソフトウェアのバージョンに関して、対応状況を確認する必要がありました。具体的な例を説明します。

### GDS環境構築での留意点

■GPUサーバーの選定
→ GPU、NVMe™、NICがサーバー内部でどの様に接続配置されているかを理解する必要性

■BIOS(EFI)レベルからの設定
→ IOMMU (Input-Output Memory Management Unit)やPCI ACS (Peripheral Component Interconnect Access Control Services)の無効化などサーバーベンダーごとに異なる設定が必要

■ソフトウェアの組合わせ
→kernelモジュール(ドライバ)、ライブラリ、MLフレームワークとの組合せの整合をとる必要
1. GDS kernelモジュールに対応したCUDA Toolkitのバージョン
2. MLフレームワークに合わせたCUDA Toolkitのバージョン
3. NVMe-oF™を利用する場合は、MLNX OFEDのバージョン
etc…

nvidia-fsに関する情報として、バージョン10.0以降対応という記載が一般的です。ところが、実際にはバージョン11.4からGPUDirect Storageに関するパッケージとして、nvidia-fs、gds-toolsが提供されていました。GPUDirect Storageの開発が活発なため、上記のような変更をドキュメントなどでチェックすることが重要です。

### GDSソフトウェアの組合わせの検討例

■GDSカーネルモジュール：nvidia-fsの導入要件
https://github.com/NVIDIA/gds-nvidia-fs より

```
Requirements
    •NVIDIA Tesla or Quadro class GPUs based on Pascal, Volta, Turing or Ampere
    •NVMe/NVMeOF storage devices or supported distributed filesystem
    •Linux kernel between 4.15.0.x and 5.4.0.x
    •MOFED 5.1 or above
    •cuda toolkit 10.0 and above
    •GPU display driver >= 418.40
```

■実際にはCUDA Toolkit 11.4以降からGDSに関するパッケージが提供されている
✓関連パッケージ：nvidia-fs, gds-tools
→ 当初導入していたCUDA Tooklit 11.2からバージョンを変更

**現在、GDSは開発が盛んに行われているため、小まめにドキュメントをチェックする必要有り。**

今回のベンチマークのソフトウェア構成を示します。

- GPUシステム全般
  - ✓ OS: Ubuntu Server 20.04.4 LTS
  - ✓ NVIDIA Driver: 470.57.02
  - ✓ CUDA Toolkit: 11.4.1

- GDS関連
  - ✓ nvidia-fs : 2.7.50
  - ✓ gds-tools : 1.0.1.3 (gdsio version :1.5)*

    * gds-tools package provides binaries for data verification, GDS config verification and a GPU based synthetic IO benchmarking tool.

- NVMe-oF™関連
  - ✓ MLNX OFED : 5.4-1.0.3.0
  - ✓ KumoScale™ : 3.19

今回実施したベンチマークについて紹介します。ベンチマークツールに関しては、gds-toolsに含まれるgdsioを用いることにしました。このgdsioを異なるブロックサイズ、モードに対して実行するgds_perf.shというスクリプトもgds-toolsに含まれており、このスクリプトを用いて測定結果を集計しました。
測定結果はこのスライドに示したようなCSV形式で出力されるので、その中からGPUDirect Storageと従来のCPUを経由したI/Oを比較するグラフを描きました。

測定環境については、以下の構成で実行しています。

**ベンチマーク実行環境**

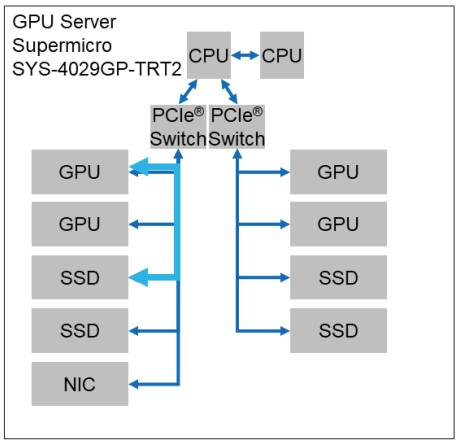
- GPUサーバ
  - ✓ Supermicro® SYS-4029GP-TRT2
  - ✓ CPU: Intel® Xeon® Gold 6148 CPU @ 2.40GHz (2 CPU x 20 core)
  - ✓ Memory: 384GB
  - ✓ 100 GbE NIC: NVIDIA ConnectX-5
  - ✓ Local SSD: NVMe™ SSD KIOXIA CM5 (1.6 TB) x 4
  - ✓ GPU: NVIDIA GV100GL x 4

- ターゲットサーバ
  - ✓ Supermicro® AS-2113S-WN24RT (PCIe® 3.0)
  - ✓ CPU: AMD EPYC™ 7702 64-Core Processor
  - ✓ Memory: 512 GB
  - ✓ NIC: NVIDIA ConnectX-6 (100 GbEシングルポートのみ使用)
  - ✓ SSD: NVMe™ SSD KIOXIA CM6 x 10

- ネットワークスイッチ
  - ✓ NVIDIA SN2700
  - ✓ Onyx 3.7.1200

まず、ローカルNVMe™ SSD1台からGPU1台に対するGPUDirect Storageの結果を紹介します。スライド右側にGPUサーバーの内部トポロジーを示しています。このGPUサーバーでは、2つのCPUのうち、ひとつのCPU配下に2つのPCIe® スイッチがあり、それぞれにGPU、SSD、NICが接続されています。

### Case Local-1: ローカルNVMe™ SSD 1台から1GPUへ

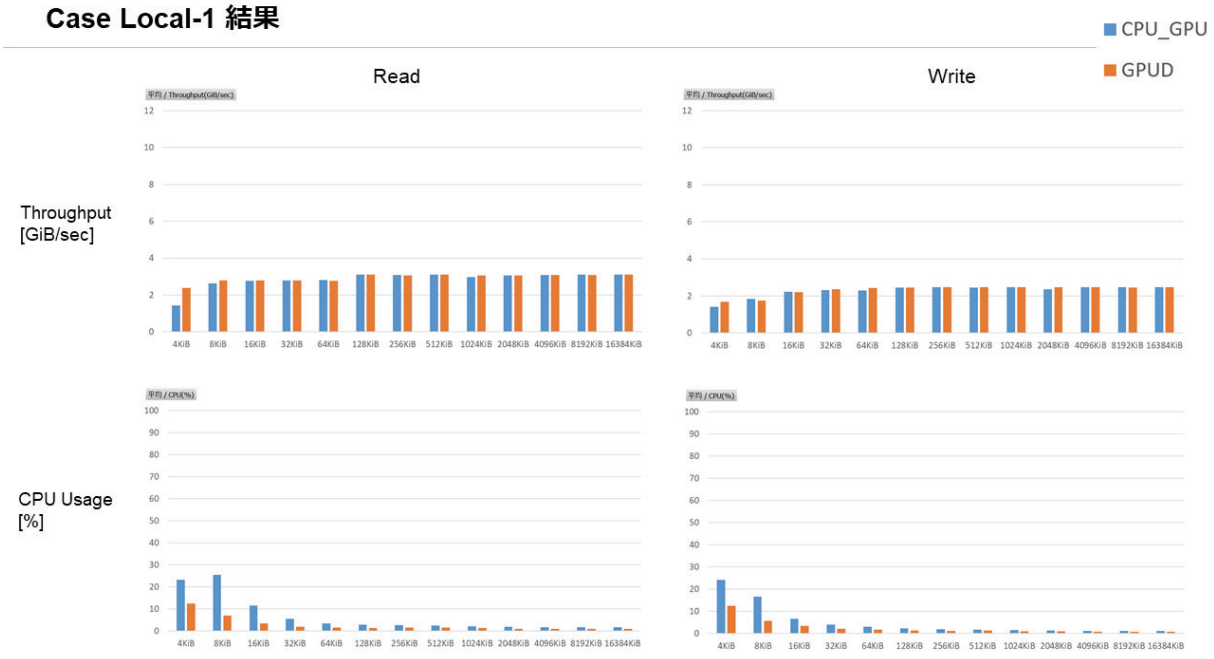



ローカルNVMe™ SSDの測定では、同じPCIe® Switch配下のGPUとNVMe™ SSDが用いられるように指定して実行しています。例えばですが、ここで誤って右側のPCIe® Switch配下のGPUから左側のPCIe® Switch配下のNVMe™ SSDを指定してGPUDirect Storageを実行すると、期待した性能が得られないので、注意が必要です。具体的なGPUとストレージの指定方法としては、GPUに関しては環境変数、ストレージに関してはスクリプトのデータ格納ディレクトリとしてSSDをマウントしたディレクトリを指定します。

測定結果は次の通りです。この表では、左側がRead、右側がWrite、上段がスループット、下段がCPU使用率となっています。グラフの読み方について説明します。まず青い棒グラフは、従来のCPUを使ったI/Oの結果を示しています。オレンジがGPUDirect Storageを用いたI/O性能の結果を示しています。

グラフのX軸はI/Oブロックサイズとなっていて、左側が4KiB、右側が16MBとなっています。縦軸に関しましてスループットは最大12GiB/secで最大値を取っており、CPU使用率に関しては100%が最大値となっています。まず、こちらのローカルの結果から分かることは、スループットに関しましてはRead、Writeともに4KiBのところで、GPUDirect Storageの方が優位な結果が出ています。しかし、それ以外のブロックサイズでは、結果がほとんど変わらないように見えます。一方、CPU使用率に関しましては、ほとんどのケースでGPUDirect Storageの方がCPU使用率が低いということが見て取れます。これにより、GPUDirect Storageの効果として、性能向上に加えて、CPUリソースの削減という目的が果たせていることが分かります。

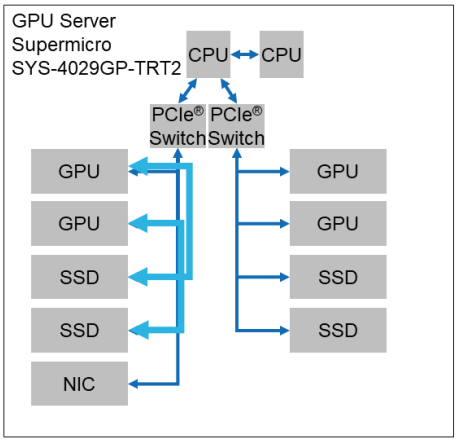
### Case Local-1 結果



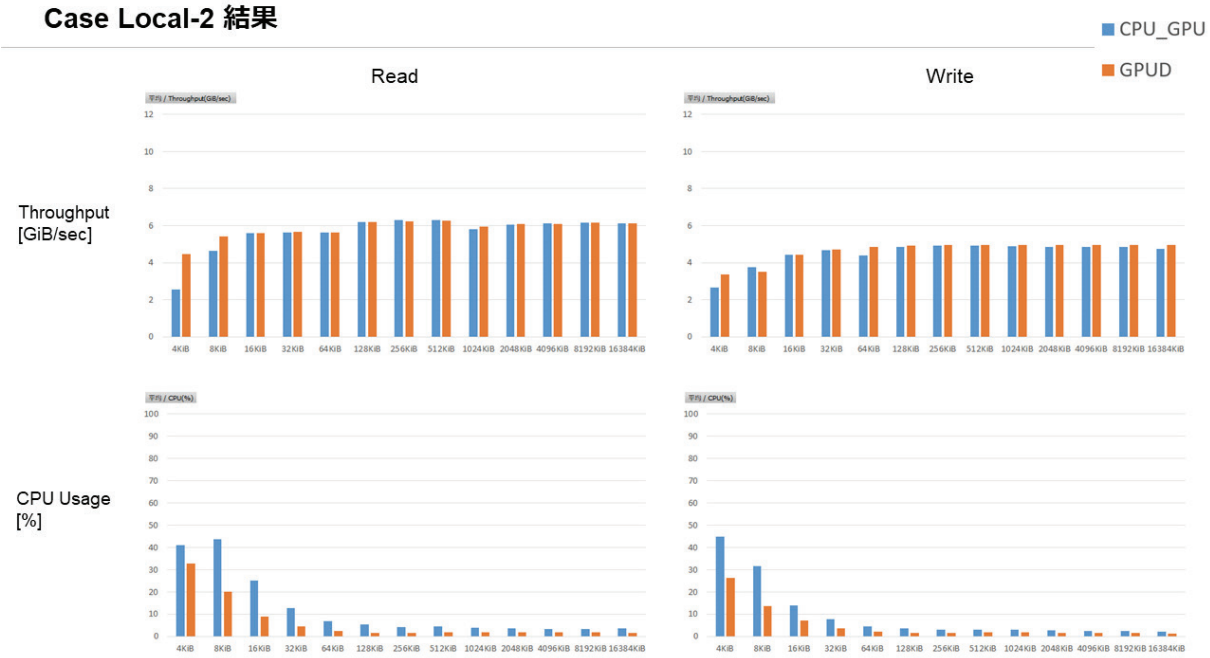
次に、2台のローカルNVMe™ SSDから2台のGPUへGPUDirect Storageを実行した結果を示します。

こちらは、先ほどの1台のSSD、NVMe™ SSDから1台のGPUに対して実行したケースに対して、スループットに関しましては、おおよそ倍の性能が出るようになっています。
そして、CPU使用率に関しましても先ほど同様に、GPUDirect Storageの方がCPU使用率が低いという結果が出ています。

**Case Local-2: ローカルNVMe™ SSD 2台から2GPUへ**



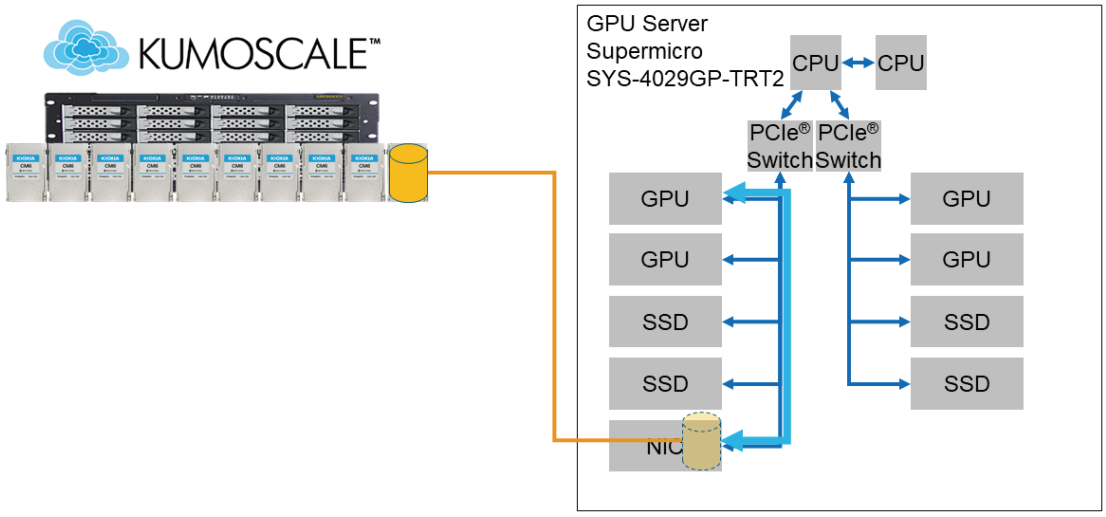

**Case Local-2 結果**




次にNVMe-oF™での測定です。こちらは最初の計測と同様に、1台のGPUから1台のSSDへGPUDirect Storageを実行する構成ですが、データはGPUサーバーのNICを介してKumoScale™とやり取りすることになります。

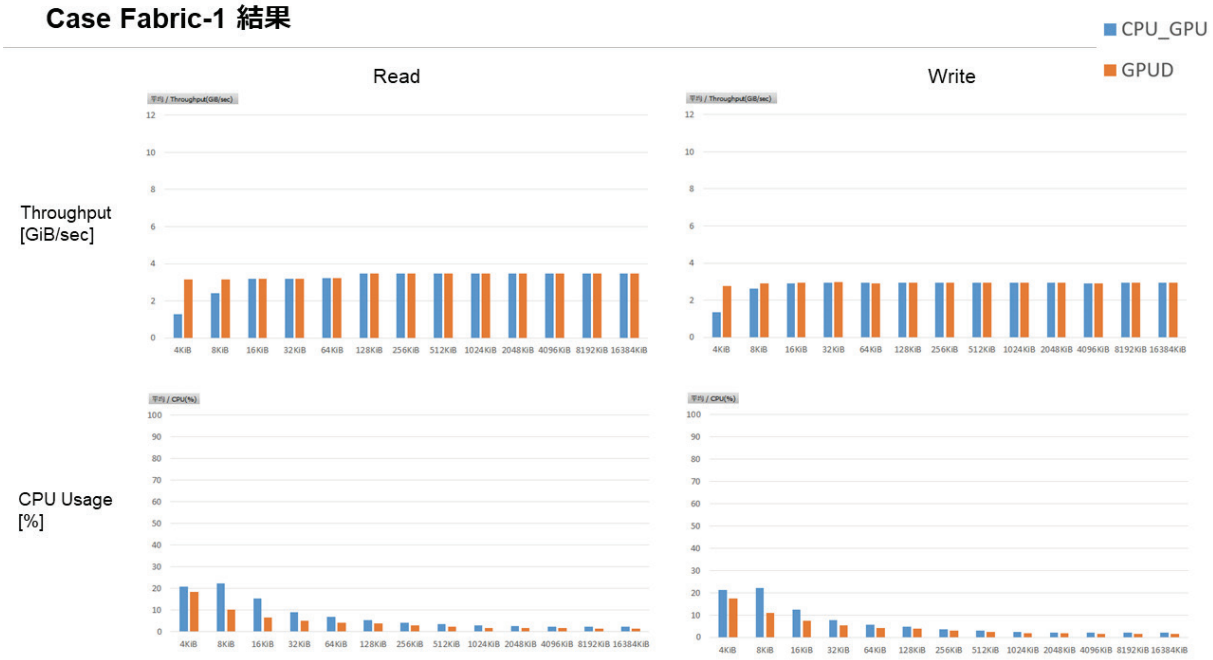
先程の構成図と同様に右側にGPUサーバーのトポロジー、左側にKumoScale™が稼働するNVMe-oF™ターゲットサーバーを記載しました。この構成では、同じPCIe® Switch配下のGPUとNICが用いられるように指定して実行しています。ローカルストレージと同様の注意点として、右側のPCIe® Switch配下のGPUから、左側のPCIe® Switch配下のNICを指定して、GPUDirect Storageを実行すると、期待した性能が得られないので注意が必要です。この場合のGPUとストレージの指定方法は、ローカルストレージの場合と同様、環境変数とディレクトリで指定します。

## Case Fabric-1: NVMe™ SSD 1台からNVMe-oF™で1GPUへ




NVMe-oF™でのGPUDirect Storageの計測結果を示します。ローカルSSDでの計測時と同じような結果になっています。4KiBの小サイズのブロックI/Oに関してはスループット、スループットに関してはGPUDirect Storageが優位な点が見えます。しかし、それ以外の部分に関しましては、ほぼ同じような結果が得られています。CPU使用率に関しましては、GPUDirect Storageのケースがリソースの使用が低い、と良い結果であることが見えています。
ただし、こちらはローカルの時ほどの差が出ていません。これはNVMe-oF™では、ネットワークの処理が入るため、CPUの使用率がローカルに比べて下がらないということが考えられます。

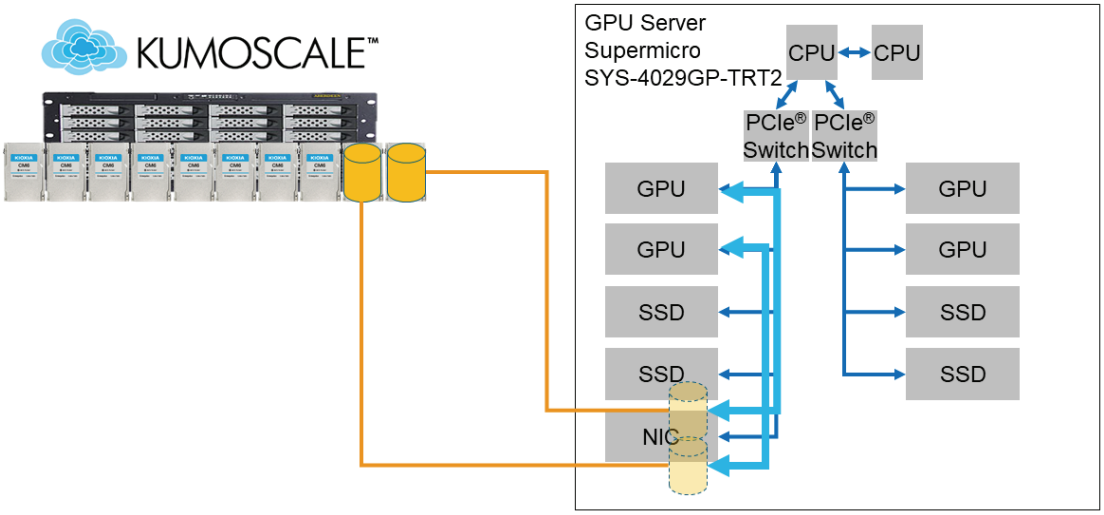
## Case Fabric-1 結果



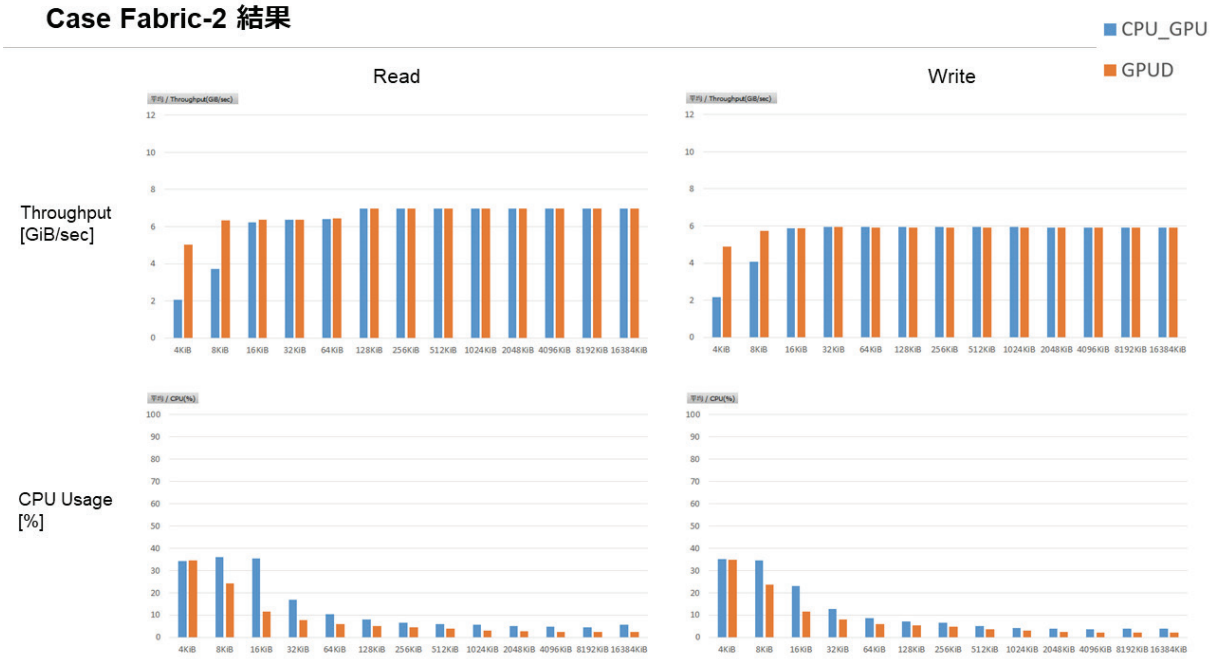

次に、NVMe™ SSD 2台からNVMe-oF™で2台のGPUに対してGPUDirect Storageを実行した結果です。

こちらも傾向としては、ローカルの時と同様に2台に増やしたことにより、1台の時に比べて倍の性能が出ているというところが見えます。CPU使用率に関してもGPUDirect Storageの方が有利に見えています。

## Case Fabric-2: NVMe™ SSD 2台からNVMe-oF™で2GPUへ並列

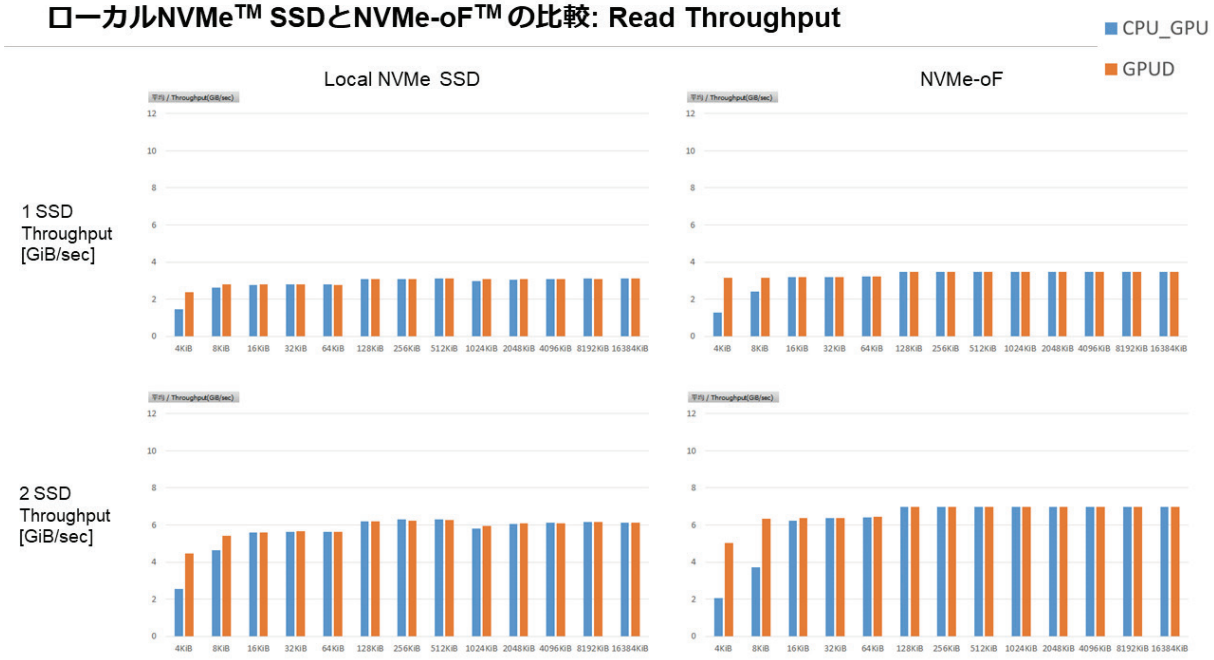



## Case Fabric-2 結果

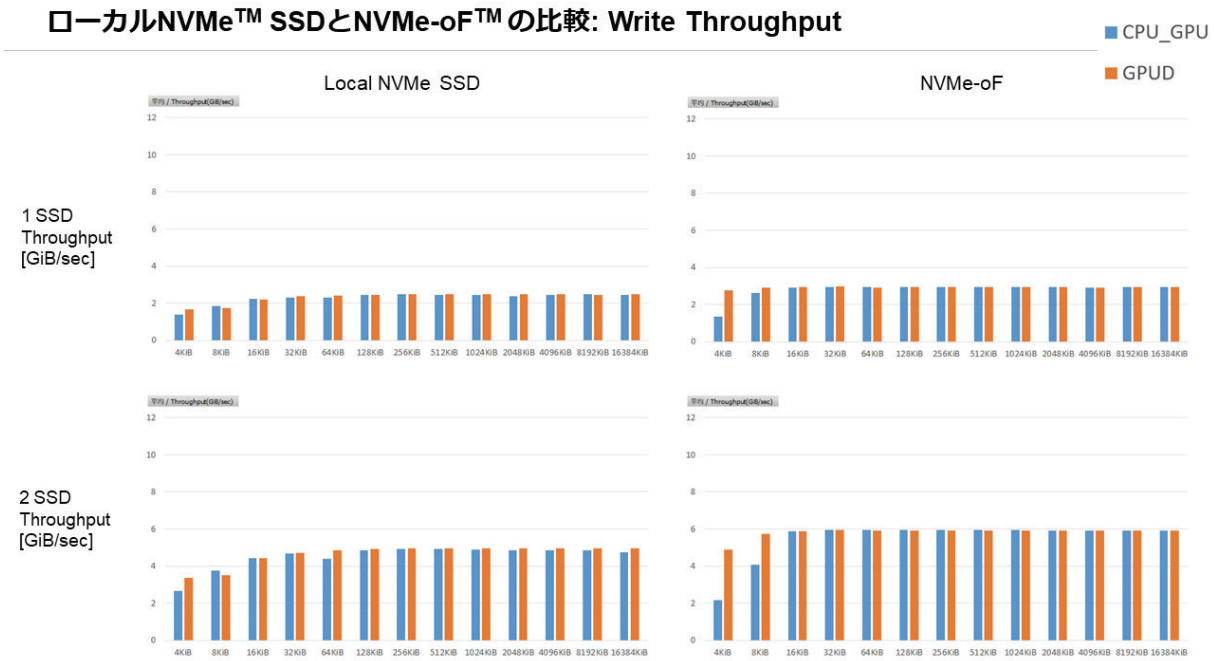



次に、ローカルのNVMe™ SSDとNVMe-oF™での計測結果を比較したいと思います。なお、この測定ではローカル側とNVMe-oF™側の使用ドライブはどちらもPCIe® Gen3接続での比較になっています。

まず、Readスループットの比較です。
左側がローカル、右側がNVMe-oF™の結果です。上の段がSSD1台、下の段がSSD2台の結果です。左と右でどちらも差があまりなくI/Oできたという結果が見えています。これは期待通りの結果なのですが、KumoScale™によってストレージを物理的に集約した上で、ネットワーク経由でストレージを提供してローカルと同等の性能を提供できていると言えます。

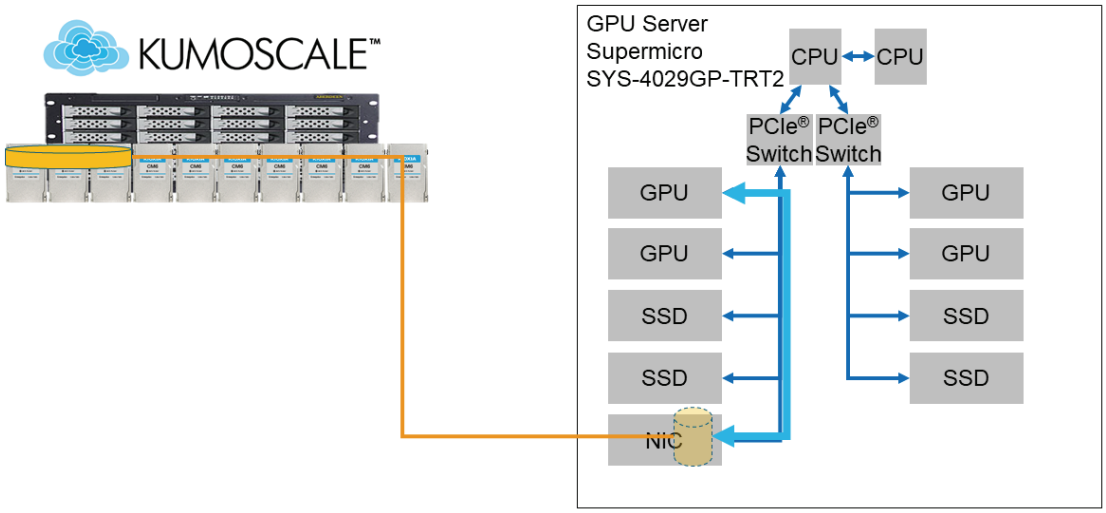



Writeのスループットに関しても同様の傾向が見て取れます。



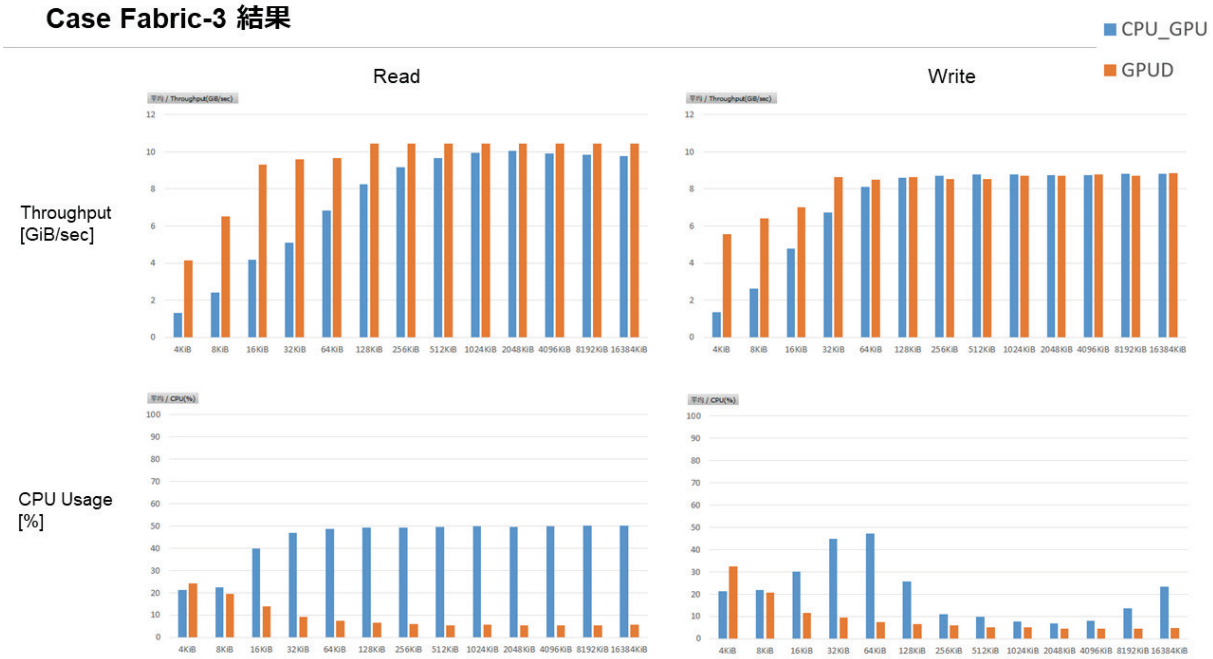

次に、KumoScale™から3台のSSDを用いて、ストライプを組んだネームスペースと、一つのGPUの間での測定結果を紹介します。

## Case Fabric-3: NVMe™ SSD 3台からNVMe-oF™で1GPUへ




まずReadのスループットに関しては、通常のCPUを介したI/Oとの差が大きくなっていることが見て取れます。なお、128KiB以上のスループットに関しましては、100 ギガビットEthernet（100GbE）の上限に達しているため、これ以上のスループットは出して出せない状況です。Writeに関しましても先程の1台のSSDよりも大きな差が出せていると思います。
CPUの使用率に関しては、概ねGPUDirect Storageの方が低く出ていますが、4KiBのところだけCPU使用率がGPUDirect Storageの方が大きくなっています。

## Case Fabric-3 結果



次に、GPUDirect Storageアプリケーション連携のベンチマークについてです。

今回、機械学習でのGPUDirect Storageの効果を明確にするために、機械学習でのワークフローにおける前処理に着目。既にGPUDirect Storageに対応しているデータ分析ライブラリーの中からRAPIDSでのベンチマークを実施しました。

## GDSによる効果の測定について

機械学習では、繰り返し処理になる学習の処理において、高速ストレージによるデータの出し入れの効率向上により貢献できると想定される。GDSを用いることにより、さらにCPUを介さずにデータの出し入れの効率が図れ、向上を実現できる可能性がある。

上記に加え、より機械学習の実ワークフローに近い形の評価、測定が必要と考え、アプリケーション連携環境でのGDS評価を実施した。今回の計測ではGDS対応済みのデータ分析ライブラリを用いてGDSの効果を測定した。


RAPIDSにはいくつものAPIが含まれており、この中でもデータ分析によく用いられるpandasライクなインターフェースを持つcuDFという機能を使って、データの入力出力の違いを測定しました。cuDFでは、4種類のデータ形式に対応しておりますが、GPUDirect Storageのファイル入力ファイル出力はそれぞれ対応が異なるので注意が必要となります。

＊2022年7月現時点の状況です。今後、開発が進捗することでソフトウェア環境が執筆時点と変わる可能性があります。

## NVIDIA RAPIDSの利用

- RAPIDSはNVIDIAが主体として開発しているデータサイエンスのワークフローをGPUで実行するためのソフトウェアスイーツ
  https://www.nvidia.com/ja-jp/deep-learning-ai/software/rapids/

- cuDFはpandasライクなインターフェースを持ちDataFrameをGPU上で扱えるようするためのRAPIDSに含まれるライブラリ
  - ✓ GDSに対応した入出力をサポート
    https://docs.rapids.ai/api/cudf/stable/basics/io-gds-integration.html

  - ✓ cuDFでGDSがサポートされているファイル形式

| データ形式 | ファイル入力 | ファイル出力 |
|---|---|---|
| Apache Avro | read_avro | |
| Apache Parque | read_parquet | to_parquet |
| Apache ORC | read_orc | to_orc |
| CSV | | to_csv |

ベンチマークに用いたpandasとRAPIDSのバージョンを示します。全てではありませんが、多くの場合、インポートのpandasからcuDFに変更することで、ほぼ同じ記述、処理が使え比較的容易にGPU化することが可能です。
例として、簡単なソースコードの比較を載せています。青の文字がpandasで、緑の文字がcuDFに変更した箇所となっております。このコードでは、2箇所の変更でGPUへの対応が可能となりました。

## データ分析ベンチマークのソフトウェア構成

- CPUで動作
  - ✓ pandas : 1.3.5
  - ✓ NumPy : 1.21.6

- GPUに対応
  - ✓ RAPIDS (cuDF): 22.04.00-stable
  - ✓ CuPy : 10.4.0

- pandas と RAPIDS の比較
  - ✓pandasからRAPIDS(cuDF)への書き換えは比較的に容易
  - ✓RAPIDSへの書き換えが行えればGDSも利用可能

| pandasの例 | RAPIDS(cuDF)の例 |
|---|---|
| `import pandas as pd`<br>`df = pd.read_parquet( “parquet形式のデータ")`<br>`df["列名"].fillna(False, inplace=True)`<br>`df.to_csv("csv形式の出力")` | `import cudf`<br>`df = cudf.read_parquet( "parquet形式のデータ")`<br>`df["列名"].fillna(False, inplace=True)`<br>`df.to_csv("csv形式の出力")` |

次に、こちらが今回使用したデータセットとデータ処理の内容になります。処理に関しては、Parquet形式のデータを読み込み、簡単な処理の後、統合してCVS形式で出力するものとなっています。
緑のハイライトになっている処理が、データの入出力の処理となり、GPUDirect Storageの効果が期待される部分となります。
表の番号は処理番号を表わしており、処理の時にデータがNVMe™SSD、システムメモリー、GPUメモリーのどこに排出されて処理をしているかを示しています。

## RAPIDSを用いた測定

- 使用したデータ「VTuber 1B: Live Chat and Moderation Statistics」
  - ✓ 学術目的のNLPデータセット(自然言語処理)
  - ✓ Open Data Commons Public Domain Dedication and License (PDDL) v1.0
  - ✓ KaggleおよびGithubで公開されているスタンダードバージョン
  - ✓ サイズが 65.63MB ~ 4.77GB のParquet形式データ

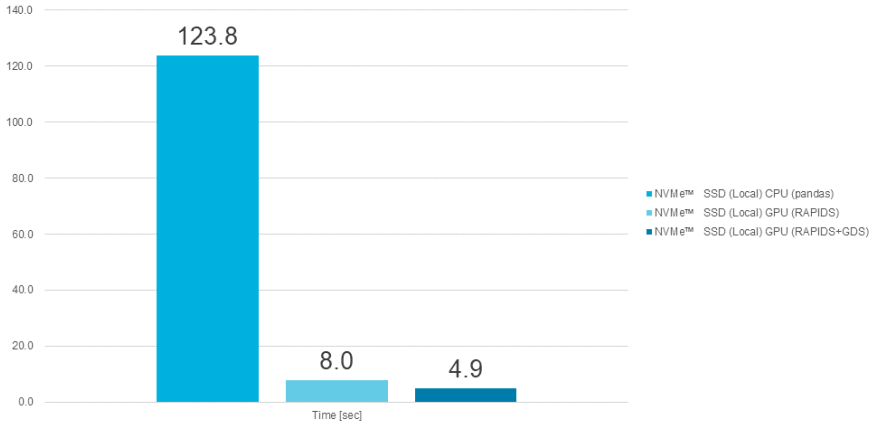
- データ処理の内容
  1. 2つのParquet形式データ(主データ約2.8 GB)をDataFrameへ読み込む
  2. 片方のDataFrameのフラグ列にTrueを挿入
  3. 2つのDataFrameを結合
  4. フラグ列の欠損値をFalseに置換
  5. 最終的なDataFrameをCSVファイルへ書き出す

| [データの配置] | NVMe™ | システムメモリ | GPUメモリ |
|---|---|---|---|
| pandas | 1, 5 | 1, 2, 3, 4, 5 | |
| RAPIDS | 1, 5 | 1, 5 | 1, 2, 3, 4, 5 |
| RAPIDS + GDS | 1, 5 | | 1, 2, 3, 4, 5 |

ベンチマークの結果です。一回の処理当たりの実行時間の平均で、値が少ない方が良い結果となります。GPUサーバーのローカルにあるNVMe™ SSDを使用した場合の結果です。

### RAPIDSを用いた測定：GDSの評価

- CPUベースのpandasに比べ、RAPIDSで大きく処理時間の短縮が可能。
- GDSにより更なる短縮が可能。

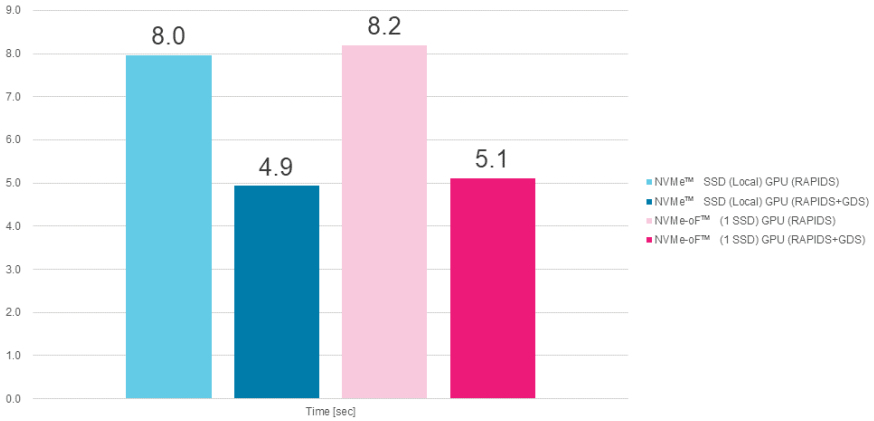



pandasのCPU処理からRAPIDSでGPU処理に変更することで、大きく性能が向上しています。真ん中がRAPIDSのみで、右側がRAPIDSに加えてGPUDirect Storageを有効にした場合の結果です。こちらからGPUDirect Storageが有利に働いていて、より少ない値で結果が出ています。
GPUDirect Storageの有無で値の差は少ないですが、データ分析ではこの処理を何度も繰り返すことになるので、僅かな差はどんどん大きくなっていきます。

次に、ローカルのNVMe™ SSDとNVMe-oF™との比較です。左側がローカルNVMe™ SSD、右側がNVMe-oF™でGDS I/Oで示した結果同様、大きな差はなく、遜色のない比較結果となっています。

### RAPIDSを用いた測定：NVMe™ SSDとNVMe-oF™

- NVMe-oF™でのRAPIDSおよびGDSはローカルNVMe™ SSDに対してわずかなオーバヘッドで実行可能。

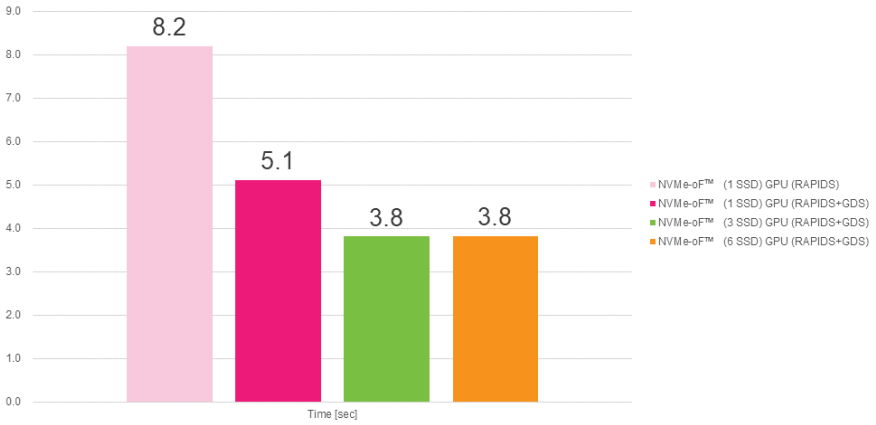


さらに、KumoScale™を使用し、NVMe™ SSDをストライピング構成にすることでI/O性能を向上させることができます。この特徴的な機能により、台数を増やすことで1台のNVMe™よりも高性能な結果が得られます。GDS I/Oのように、ベンチマーク専用のソフトで得られた結果同様、実用的な処理でもこの効果が得られていることが分かります。

これはPCIe®スロットの理論性能限界まで可能で、ローカルのNVMe™ SSDよりも良い性能を求める場合に有効と考えられます。

### RAPIDSを用いた測定: KumoScale™を用いたストライプボリューム評価

- 1 SSDに比べて3 SSDのストライプにより処理時間をさらに削減可能。
- ネットワークがボトルネックとなる環境のため、3 -> 6 SSDでも処理時間の短縮は見られなかった。




今回の検証では、GPUDirect Storageを使用することで、機械学習ワークフローにおける処理時間の改善効果を測定することが目的でした。I/Oベンチマークツールにより、GPUへのデータ転送性能が向上する点、CPUの使用率が減ることが確認できました。それに加え、より実アプリケーションに近い形として、データの前処理においてGPUDirect Storage対応ライブラリーを活用することで、従来CPUを用いて実行していた前処理の実行時間を大幅に短縮できることを確認しました。

当初想定した高速ストレージを活用することで、機械学習ワークフローの処理時間短縮が実現できることを示せたと思います。今回の検証結果により、機械学習環境に高速ストレージを用いることは、もはや特別なものではないことがわかりました。GPUDirect Storageの将来性に関しては、今後も機械学習ワークフローのさまざまなステップに関わるツールがGPUDirect Storageに対応することで、さらなるワークフロー処理時間の短縮が期待できるでしょう。

### まとめ

今回の検証では、I/OベンチマークツールによりGPUへのデータ転送性能が向上する点、CPUの使用率が減ることが確認できた。

また、データの前処理において、GDS対応ライブラリを活用することで、従来CPUを用いて実行していた前処理の実行時間を大幅に短縮できることを確認した。

今後も機械学習ワークフローの様々なステップにかかわるツールがGDSに対応することで、さらなる処理時間の短縮が期待できる。

今回のような機械学習ワークフロー処理時間の改善に貢献するKIOXIAの製品について、ご紹介します。今回の検証で実際にGPUDirect Storageの効果が確認されたローカルストレージとしてのNVMe™ SSD、それに加え、低遅延モデルのNVMe™ SSDもあります。同じく今回の検証により、機械学習の性能向上の効果が確認されたストレージ・ディスアグリゲーションを実現するNVMe-oF™ ソフトウェア製品のKumoScale™があります。

加えて、SSD単体でNVMe-oF™に対応したEthernet接続型SSDがあります。こちらはKumoScale™とは異なる方式ですが、スケーラブルなNVMe-oF™の高速ストレージ環境を構築できます。また、このEthernet接続型SSDもGPUDirect Storageに対応済みです。

### KIOXIA製品ご紹介

KumoScale™: https://business.kioxia.com/ja-jp/ssd/kumoscale-software.html
SSD: https://business.kioxia.com/ja-jp/ssd.html


NVMe™ SSD

KUMOSCALE™
NVMe-oF™ソフトウェア

Ethernet SSD
(NVMe-oF™対応SSD)

*AMDおよびEPYC は、Advanced Micro Devices, Inc. の商標です。
*Intel および Xeonは、Intel Corporationまたはその関連会社の商標です。
*Linux は、米国およびその他の国におけるLinus Torvaldsの商標です。
*Supermicroは、米国およびその他の国における Super Micro Computer, Inc. または関連会社の商標または登録商標です。
*その他記載されている社名・商品名・サービス名などは、それぞれ各社が商標として使用している場合があります。