いまいちどストレージプールを考える

Hiroshi Aiko, Enterprise Product Expert, Enterprise Marketing, NVIDIA

JLUG 2023 – Dec 1st , 2023

サービスデマンドの変化とストレージ需要の変化

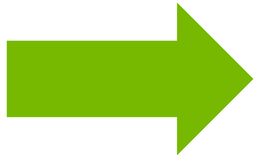
リッチなクラウドサービス
リッチなコンテンツサービス

- ✓ テキスト中心のコミュニケーションから
  画像や動画を駆使したコミュニケーションへ
- ✓ 5G等、リッチなモバイル環境の展開
  - AR/VRの実用化、業務応用など

AIに対応した新しいサービス

- ✓ GPUを使用可能なクラウドコンピューティング
- ✓ 学習用のビッグデータの取り扱い
- ✓ ハイパフォーマンスコンピューティング需要の拡大

今までにない高速な記憶域

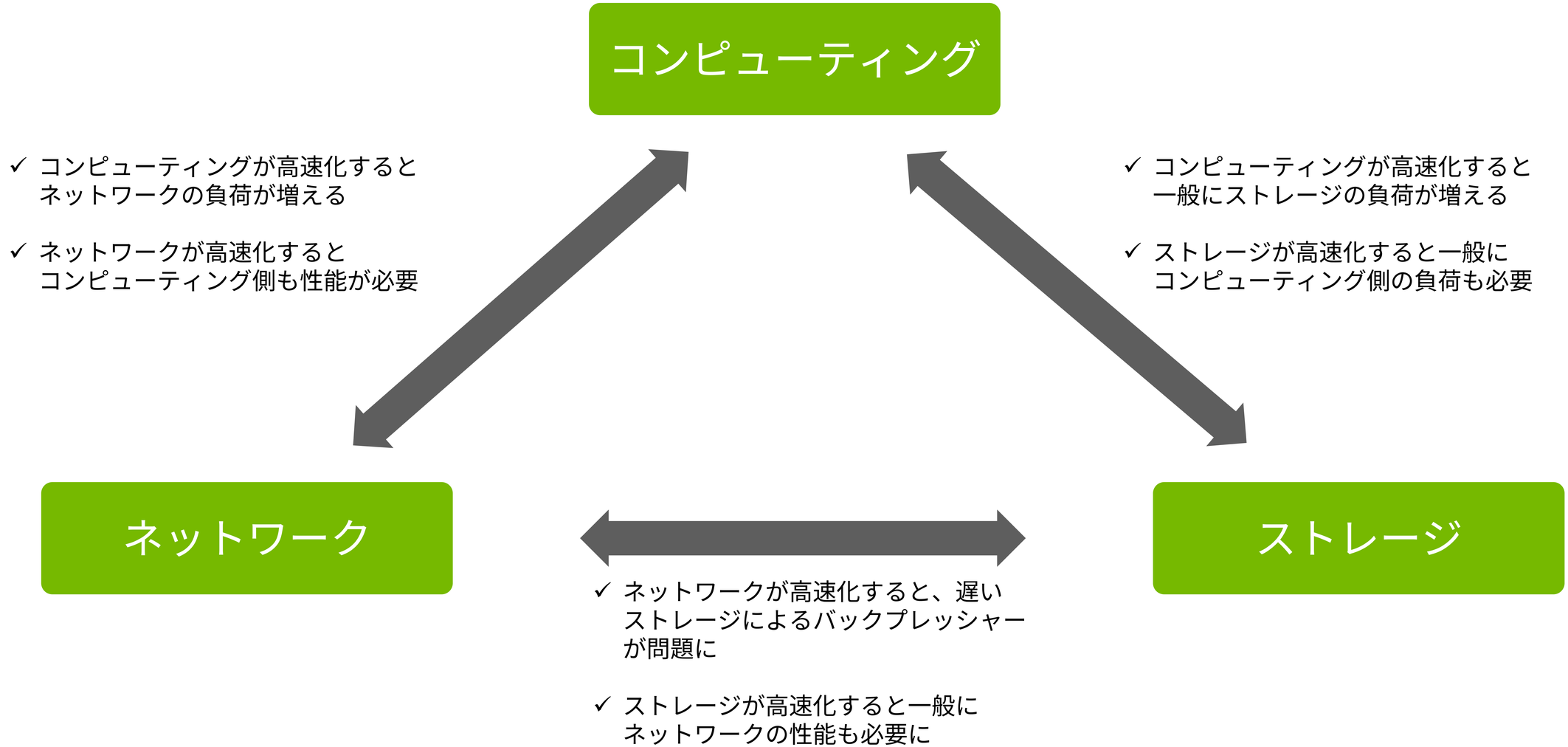


より大容量のデータの扱い

より高速なデータの流れ

システム性能の3竦み

どの要素が変化しても、他の2要素に影響が及ぶ



「速い」とは

「速い」のはどれだろう？

バイク便

トヨタ・ダイナU600（ヤマト運輸仕様）by NagoPIYO (from Wikipedia), CC 表示-継承 4.0

宅配便

船便

速いとは？

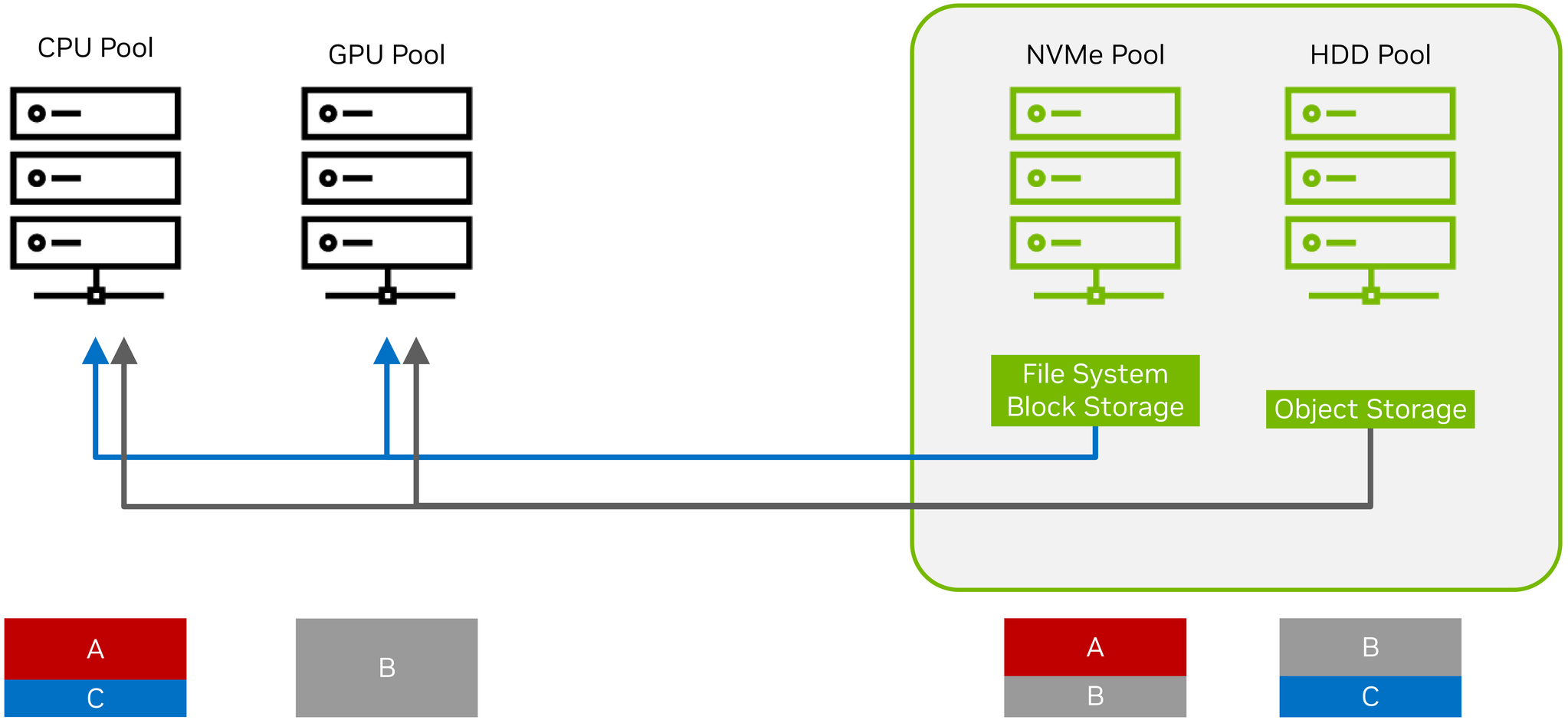
✓依頼から入手までの時間が短い？
✓単位時間あたりに送れる貨物の量が多い？

ありがちな誤解

- ストレージのベンチマークが良ければ、その分システム性能も上がる
- ストレージはローカルに置くのが一番速い
- ストレージに比べてCPUやメモリは十分に速い

今日は、このあたりの内容を頭の片隅に置きながら
ストレージプールについて考えていきます

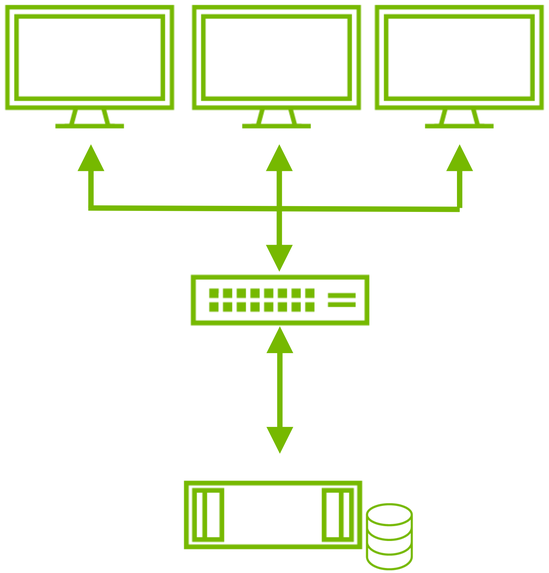
改めてストレージプール



「速い」ストレージプールやインフラ

InfiniBandを利用したストレージプールのすすめ

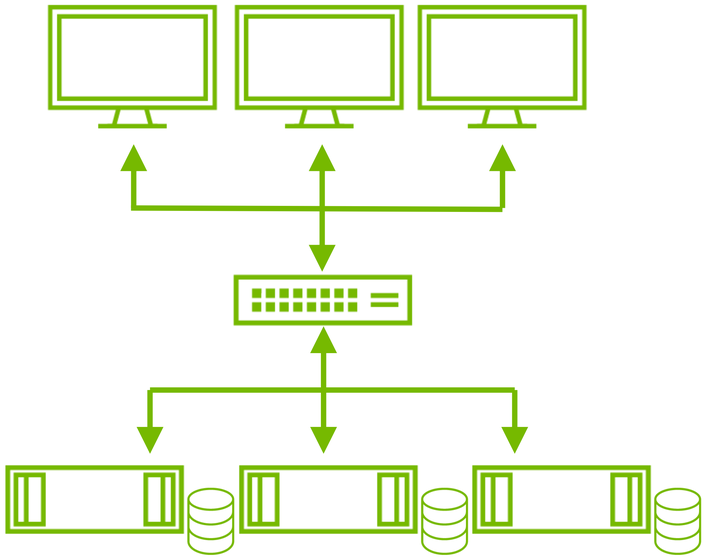
NAS



SMB(Samba) / NFS

~50Gbps

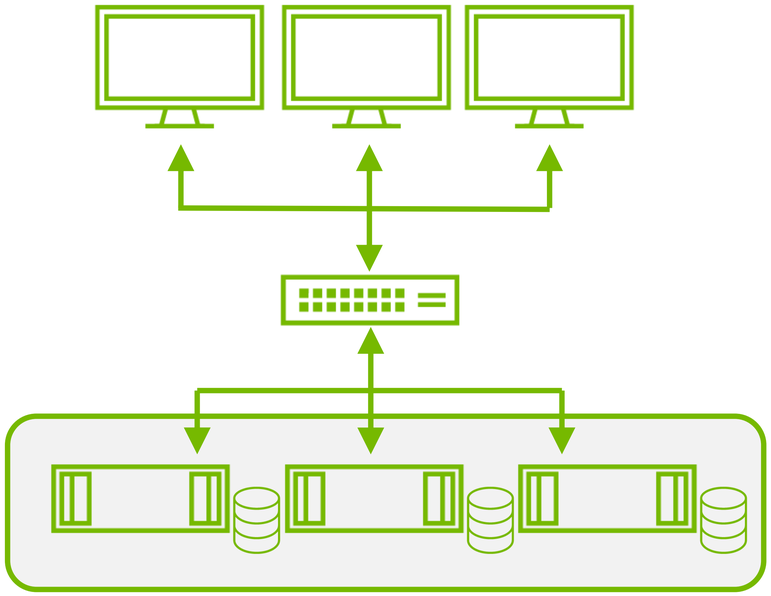
SAN



NVMe-oF

~10M IOPS

SDS



NVMe-oF　　Luster/GPFSなど

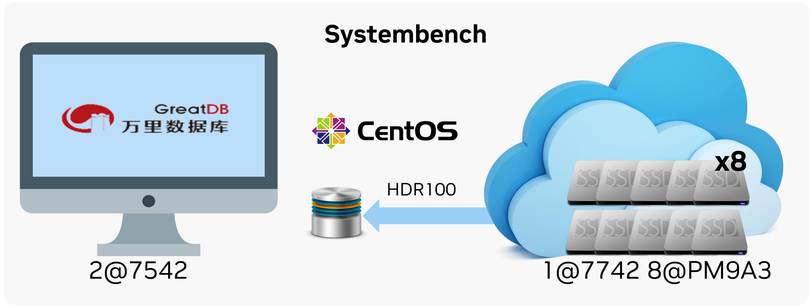
~20M IOPS (or more)

✓ ノードにストレージが不要。ストレージを集中管理可能。
✓ スケールメリットのある高速ストレージを共有

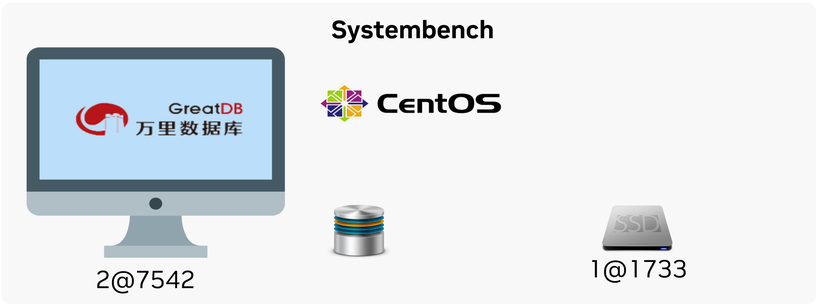
✓ ノードのCPUやGPUに負荷をかけない
✓ 個別にストレージを積むより多くの場合低コストでグリーン

ローカルストレージ vs IB プールストレージ — GreatSQL*の場合

IB-POOLING-RDMA-100G



25% Coverage	50% Coverage	75% Coverage
11096.98	12594.84	12360.47



Local-1SSD

5992.6	8372.38	10964.87
120%	50.4%	12.7%

* GreatSQL は MySQL からフォークされています

NVIDIA

なぜこういうことになる？

- ローカルストレージからの読み込みは、割とメモリコピーに伴うCPU負荷が高い
  - Coverageが低い状態で性能が伸びない理由の一つ
- RDMAや高性能ネットワークの活用でネットワークが十分「高速」なら、中途半端なローカルストレージよりリモートストレージのほうが低負荷で高速
- ローカルストレージの読み込みは、CPU負荷も高く、データベース等のアプリケーションの観点では最終的な性能を引っ張ってしまう
  - ベンチマークだけではシステム性能が図れない理由

CPUが介在するメモリ転送は、よく設計されたRDMA通信より負荷が高い
ローカルストレージやキャッシュはシステム的には必ず速いとは限らない

GPUを活用するサービスからみたIO

GPU Direct Storageおさらい

GPUを活用したサービスが増えてきましたが。。。。

GPUから見た、ストレージIOの問題

✓ストレージIOはCPUメモリに転送されるため、相対的に広帯域とは言えない CPU – GPU のDMA転送を繰り返す必要がある

✓Storage → CPU、CPU→GPUと2回のDMAをCPUメモリに対して行うことになり、CPUバスの帯域が半減する可能性がある

✓バウンスバッファが必要となり、メモリの消費効率が落ちる

ストレージから直接GPUメモリにロードできれば、これらの無駄は一気に省ける

GPU Direct Storage の登場

GPU Direct Storage

1

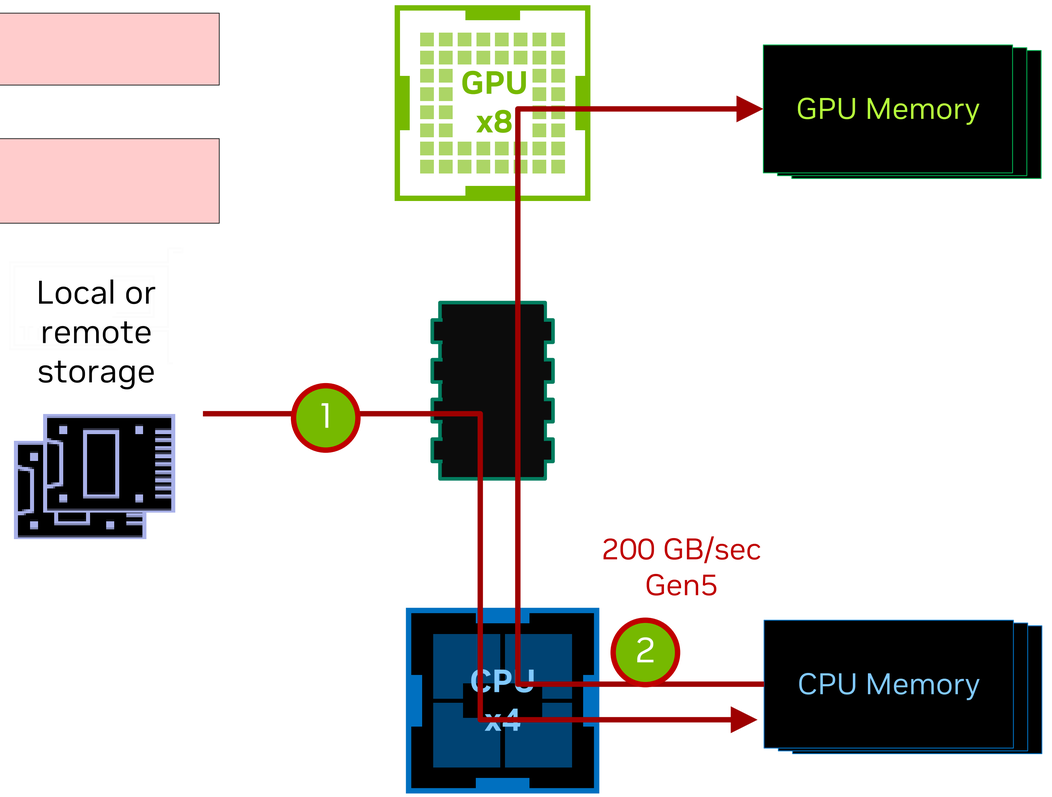
```
pread (fd, cpubuf, 8192, 0);
```

2

```
cudaMemcpy(gpubuf, cpubuf, 8192,
cudaMemcpyHostToDevice);
```

従来のファイルシステムアクセス

2回のDMA操作
バウンスバッファの必要性
CPU - PCI 間の帯域が実質半分に？



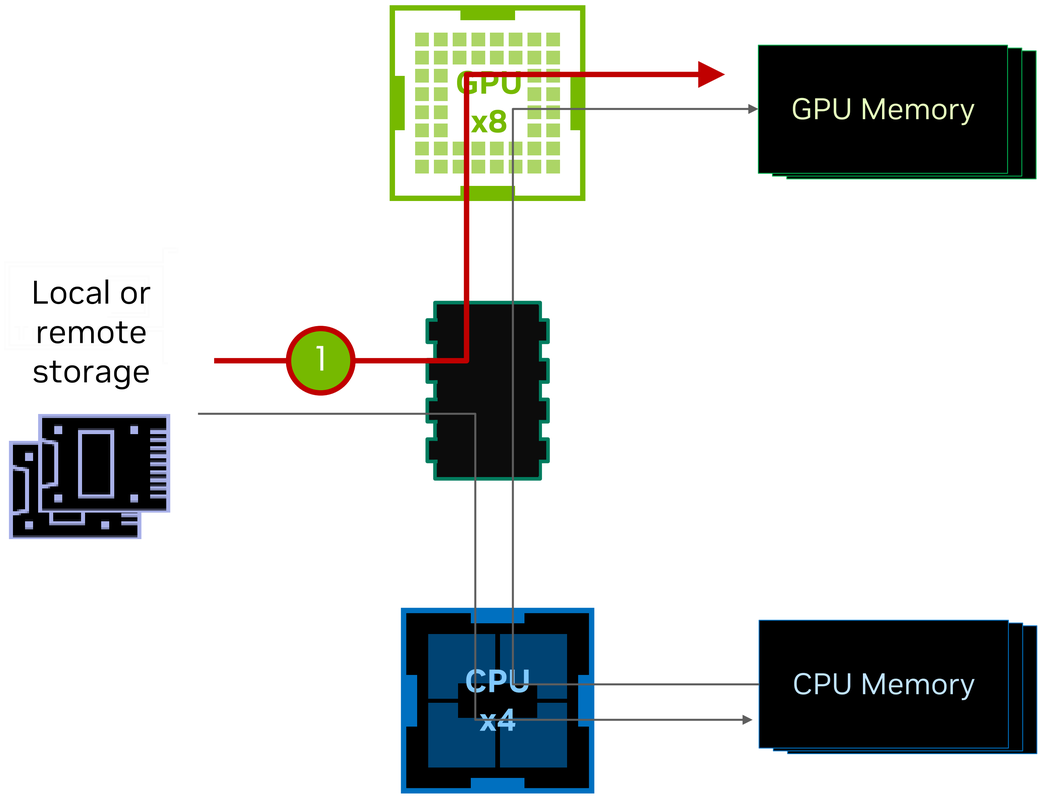
GPU Direct Storage

1

```
cuFileRead(fd, &gpubuf, 8192, 0, 0)
```

GPU Direct Storage

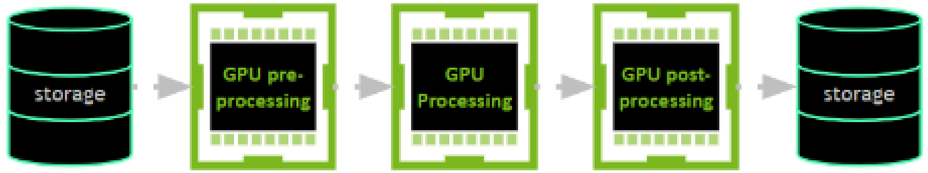
単一の DMA 操作
バウンスバッファ不要
CPU - PCI から独立した転送



GPUDirect Storage - ユースケース例

IO パフォーマンス高速化

Pre-Stack Time Migration	1.19x
Reverse Time Migration	1.3x - 6.3x
HPC Visualization	8x
DeepCAM Inference	4.6x
Merlin NVTabular	1.4x
Genotyping	3x-6x
Seismic Simulation	2.5x
RAPIDS/cuCIM	11x
RAPIDs cuDF	4.5x
KvikIO/Zarr	2.9x
DeepCAM DL Training	1.05x
Cosmoflow DL Training	1.2x



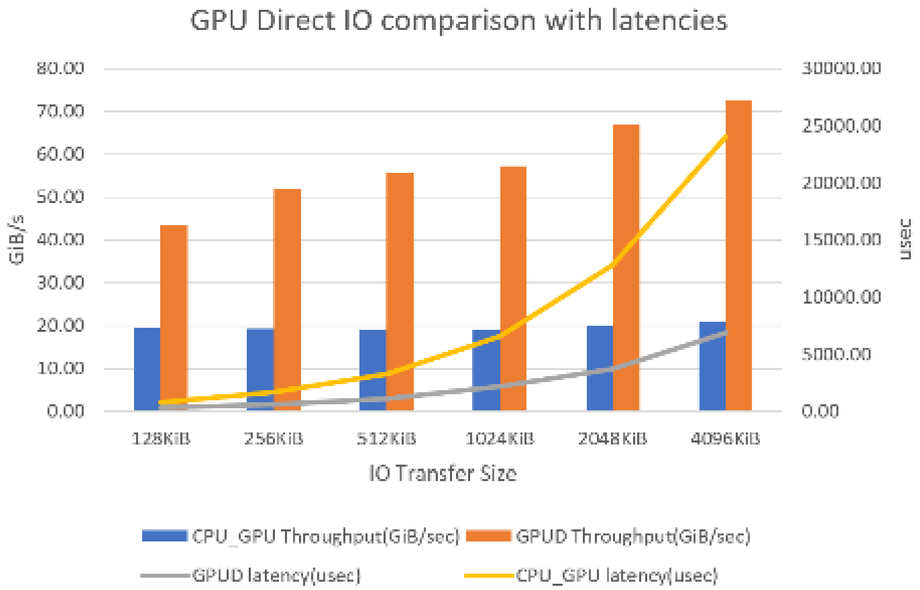
Keep workflow & data on GPU

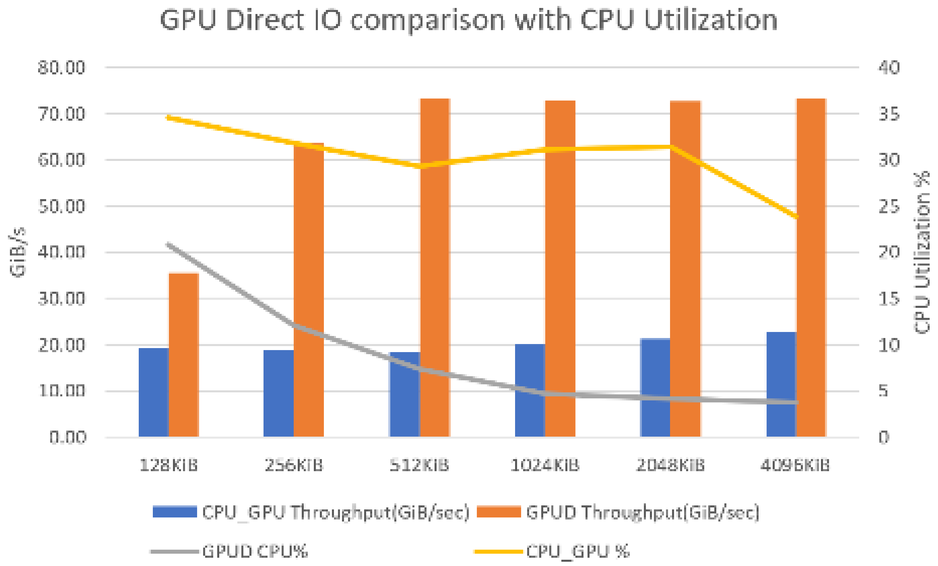
利点	要求事項
CUDAでのFile IO. 特別なハードウェアなし	CTKの利用
1.2x - 8x の帯域向上	システムトポロジ(PCIe Switch)
GPU メモリへの高速なIO	GPUDirect / Vidmem のみ
最高 1/3 のCPU 使用率低減	O_DIRECT のみ

多くのGPUアプリケーションは IOセンシティブであり、大量のデータを処理する

GPUDirect Storage パフォーマンス

GDSIO



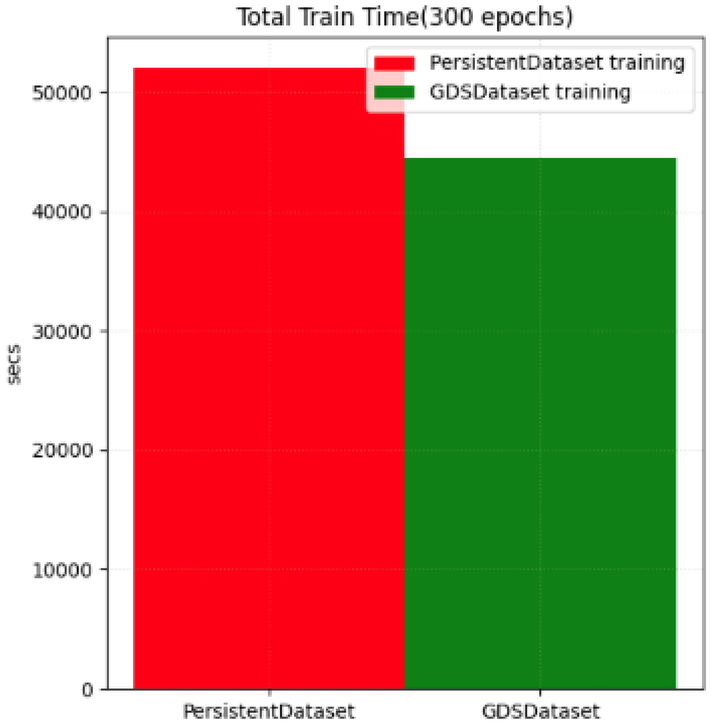


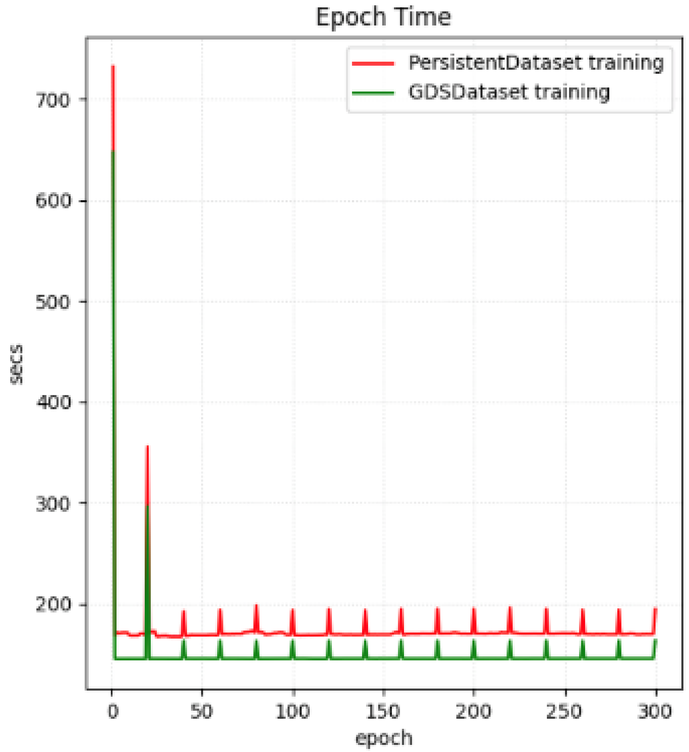
Note : 16 Threads per GPU; Total 128 threads for read

最大3.5倍の高い帯域と50%の遅延の削減

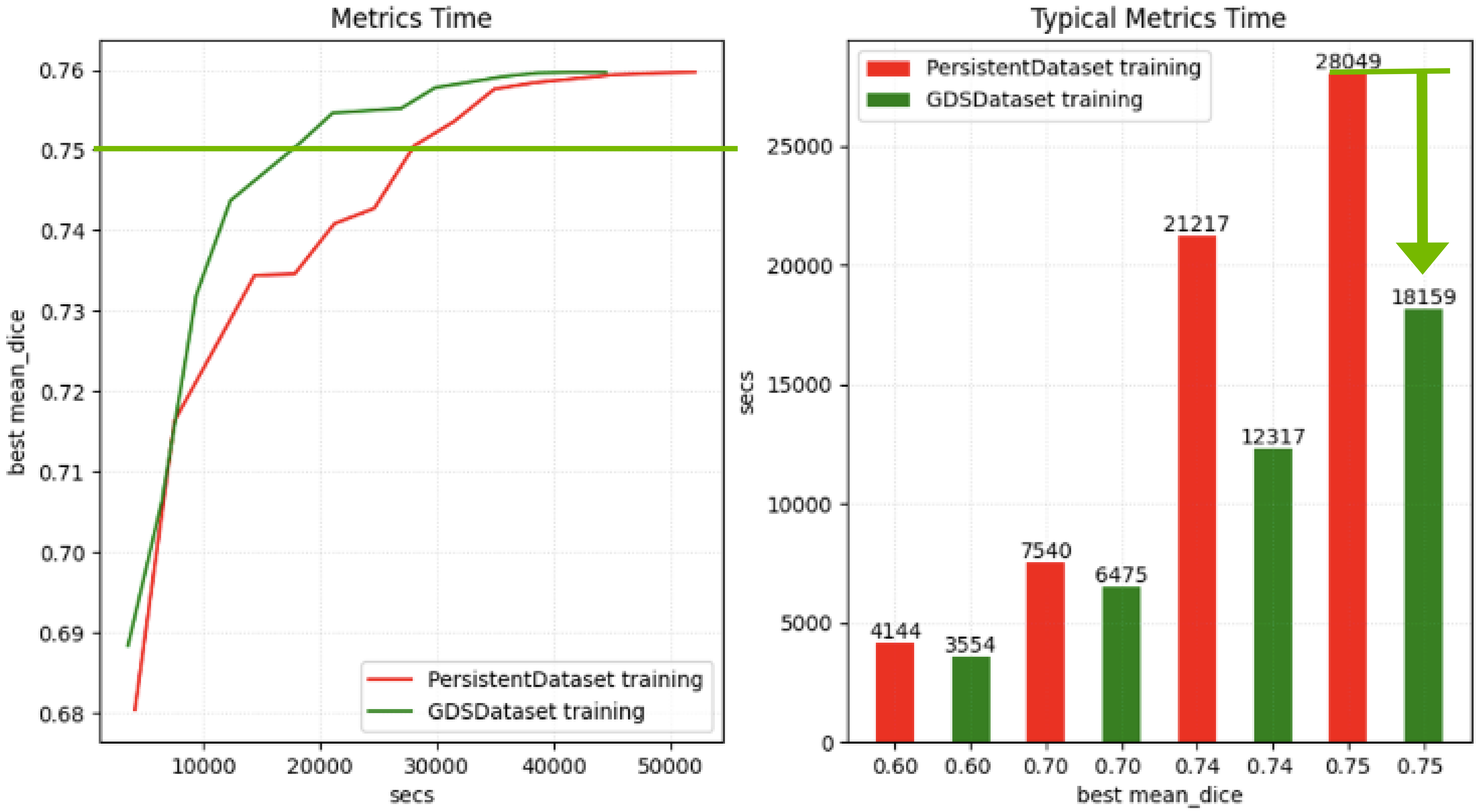
MONAIにおけるGDS

GDSとcuFile APIの利用で、通常のキャッシュメカニズムと比べてトータル10%の高速化





GDS vs Persistent Caching Dataset



GDSによりモデルは0.75 mean_dice に18159 秒で到達
Persistent Caching Dataset 使用時の 28049 秒に比べ **1.54倍** 高速化

まとめ

- サービス要求の変遷に伴い、ストレージの果たす役割はより重要になっています。
- しかしながら、ストレージのみに着目して調整等を図っても、システム全体としてみた場合には必ずしも適切な性能を発揮することはできません。
- 性能要求に対して安直に性能の向上が見込まれなくなった現代においては、その果たすアプリやサービスに合わせたシステムレベルでの考察が必要です。
- 特にスケールが必要なストレージや、GPUを活用する場合など、お困りがございましたらNVIDIAにもご相談ください。