# LLM ロード高速化ライブラリ fastsafetensors の AI アクセラレータ適用拡大に向けて

吉村 剛[1,a)]　千葉 立寛[1,b)]

## 1. はじめに

AI アクセラレータ (例：Intel Gaudi [8], IBM Spyre [7]) の利用は NVIDIA GPU 利用と比較しより良い価格対性能効率や電力消費効率が見込まれる。本研究の主題である fastsafetensors は、safetensors ファイルの NVIDIA GPU メモリへのロードが非効率であることを問題意識として、それを改善する手法として提案されている [2], [11]。モデル初期化は NVIDIA GPU 固有でなく全てのアクセラレータで必要な処理であることから、fastsafetensors と類似した手法は必要になると考えられる。

## 2. fastsafetensors

fastsafetensors は safetensors ファイルで配布される訓練済みモデルの重みのロードを高速化のオープンソースライブラリとして公開されている。safetensors フォーマットからテンソルオブジェクトをデシリアライズする際、テンソルオブジェクト転送のバッチ・並列化を行い、テンソル並列化時のデータ分割に集団通信 API を再利用して高性能 NVLink インターコネクトを活用し、GPUDirect により冗長なコピーを減らす。それにより数百ギガバイトを超える言語モデルであっても vLLM [9]、 SGLang [12] 等の推論ランタイムの起動にかかる時間を最小化する。その結果、AI アプリの開発やテストの効率化、llm-d [5] のオートスケーリングなどのより実践的かつ複雑な手法を効率化する。fastsafetensors はすでに vLLM の拡張機能として組み込まれており、PyPi パッケージインストール後にサーバー起動オプションを指定すると利用できる [1], [10]。

[1] IBM 東京基礎研究所
[a)] tyos@jp.ibm.com
[b)] chiba@jp.ibm.com

## 3. 課題

開発当初の fastsafetensors の最大の制約は機械学習フレームワークとして Pytorch を利用することと、NVIDIA CUDA 環境で利用することを前提に実装されたことにある。具体的には Pytorch はテンソルメタデータの交換やデータ変換の操作に主に利用され、CUDA はテンソル操作のカーネルおよび複数 GPU 間の通信時に呼び出されている。

しかし、提案された設計の本質は特定アクセラレータや機械学習フレームワークの機能を前提にしたものではない。高速化へ最大の貢献となっていたテンソル転送のバッチ・並列化する方式は機械学習フレームワークとは独立して標準化された DLPack [6] の機能を活用したものである。NVLink の活用も、標準化された MPI のインターフェース (broadcast, scatter) を再利用している。GPUDirect は GPU メモリと他 PCI Express デバイス間での peer-to-peer DMA を利用するための手法であり、PCI Express デバイス一般で同様の機能は実現できるものとなっている。実装の煩雑さから実現が容易でなかった場合でも、ホスト DRAM を利用した DMA で代替機能として実現することもできる。

## 4. 機械学習フレームワーク呼び出しの抽象化

本発表では、AI アクセラレータ適応に必要となる、Pytorch などの機械学習フレームワークや CUDA のようなアクセラレータによる汎用目的計算のためのランタイムといった機械学習フレームワーク呼び出しを抽象化する方法を紹介する。具体的には、サポートする値の型、テンソルオブジェクトのメタデータ操作、並列プロセスグループ、テンソルオブジェクト新規確保操作の 4 つを抽象化する必要がある。

| 種別 | 型名 | 追加型変換 |
|---|---|---|
| 真偽値 | BOOL | 無 |
| 符号あり整数値 | I8, I16, I32, I64 | 無 |
| 符号なし整数値 | U8, U16, U32, U64 | 無 |
| 浮動小数点数 | FP16, FP32, FP64 | 無 |
| 8 ビット浮動小数点数 | F8_E5M2, F8_E4M3 | 有 |

**表 1** サポートする型

fastsafetensors は内部で機械学習フレームワークに合わせて型変換する API を利用するものの、特定機械学習フレームワークへの依存を減らすために自前の型を自前のテンソルクラスのラッパと関連付けて管理・受け渡しする。DLPack がネイティブでサポートしているものはそのまま利用できるものの、低精度のものは一旦８ビット整数としてロードし、その後のゼロコピーでの追加型変換で対処する。現在サポートする型は表 1 となる。サポートする値の型は通常の 64 ビットの浮動小数展から近年提案された低精度のものまで存在する。Pytorch では新しいバージョンになるにしたがってサポートする型は増えているため、利用できるものを動的に検査する必要がある。

テンソルオブジェクトはデータが数バイトから数百メガバイトの大きさがあるため、メタデータとして内部で受け渡しする必要がある。特に、メタデータ操作により内部状態を変更する場合の処理を抽象化し、複数フレームワークでの異なるテンソルの扱いを吸収する。主にテンソルのホスト・デバイス間転送 (to)、コピー (clone, detach)、ゼロコピーでの形状置換や型の変更 (view)、メモリが物理的に連続であるかのチェック (is_contiguous) が必要となる。

並列プロセスグループはテンソル並列化での分割やコピーの高速化に直接利用される。基本的な情報、たとえば参加プロセスの数 (size) や自分自身のランク (rank) がまず必要となる。また、集団通信のラッパ (broadcast, scatter) とポイント間通信 (send, recv) を提供する。

テンソルオブジェクトの新規確保の抽象化では、機械学習フレームワークの低レベルなメモリ操作をラップすることができる。ここでは基本的なフレームワークレベルの情報取得 (get_device, get_dtype_size, get_runtime_ver, support_fp8) に加え、アクセラレータメモリのバイト数指定の割り当て・デバイスポインタ取得 (alloc_tensor_memory)、新規テンソルオブジェクトの確保 (get_empty_tensor, concat_tensor, randn) 、DLPack によるデバイスメモリのテンソル変換 (from_dlpack) を提供する。

これらのインターフェースを実装すれば、NVIDIA GPU と同等の safetensors ロード最適化を異なる AI アクセラレータや機械学習フレームワークでも実現できる。この抽象化により、実際に Pytorch ではない機械学習フレームワーク PaddlePaddle であったり [3]、AMD ROCm [4] のサポートが可能となっている。

## 5. 今後の展望

今後はより多くの機械学習フレームワークの対応 (Tensorflow など) や、AI アクセラレータ対応 (IBM Spyre など) を考えている。ただし、そうした方向性では CUDA アプリケーションとは異なる実行モデル (データフロー型や AI コンパイラ利用) も吸収する必要がある。

### 参考文献

[1] : fastsafetensors - PyPi, https://pypi.org/project/fastsafetensors/ (2025).
[2] : High-performance safetensors model loader, https://github.com/foundation-model-stack/fastsafetensors (2025).
[3] : PArallel Distributed Deep LEarning: Machine Learning Framework from Industrial Practice, https://github.com/PaddlePaddle/Paddle (2025).
[4] : ROCm Software, https://www.amd.com/en/products/software/rocm.html (2025).
[5] Cedric Clyburn, C. N.: What is llm-d and why do we need it?, https://www.redhat.com/en/blog/what-llm-d-and-why-do-we-need-it (2025).
[6] DLPack contributors: Welcome to DLPack's documentation! — DLPack 0.6.0 documentation, https://dmlc.github.io/dlpack/latest/ (2022).
[7] IBM: IBM Introduces the Spyre Accelerator for Commercial Availability, https://newsroom.ibm.com/2025-10-07-ibm-introduces-the-spyre-accelerator-for-commercial-availability (2025).
[8] Intel: IntelⓇGaudiⓇAI Accelerator Products, https://www.intel.com/content/www/us/en/products/details/processors/ai-accelerators/gaudi.html (2025).
[9] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H. and Stoica, I.: Efficient Memory Management for Large Language Model Serving with PagedAttention, *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*, p. 611–626 (2023).
[10] vLLM: Loading Model weights with fastsafetensors, https://docs.vllm.ai/en/stable/models/extensions/fastsafetensor.html (2025).
[11] Yoshimura, T., Chiba, T., Sethi, M., Waddington, D. and Sundararaman, S.: Speeding up Model Loading with Fastsafetensors, *IEEE 18th International Conference on Cloud Computing (CLOUD '25)*, pp. 163–174 (2025).
[12] Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C. and Sheng, Y.: SGLang: Efficient Execution of Structured Language Model Programs (2024).