次世代HPC・AI開発支援拠点形成事業における東京科学大学の令和7年度の取組

2026/03/30

HAIRDESCシンポジウム

Institute of SCIENCE TOKYO
東京科学大学
横田理央

東京科学大TSUBAME4.0スパコンの概要

Institute of SCIENCE TOKYO

計算ノード：
HPE Cray XD665 ×**240台**
H100 GPU 計960台
総演算性能：
- **66.8 PFlops（倍精度）**
- **952 PFlops（FP16）**

共有ストレージ：
HPE Cray ClusterStor E1000
総容量：
- 44 PByte（ハードディスクベース）
- 327 TByte（SSDベース）

30ラックに格納
- 計算ノード 23ラック
- ストレージ・管理 7ラック

横浜キャンパスG4-A棟に設置

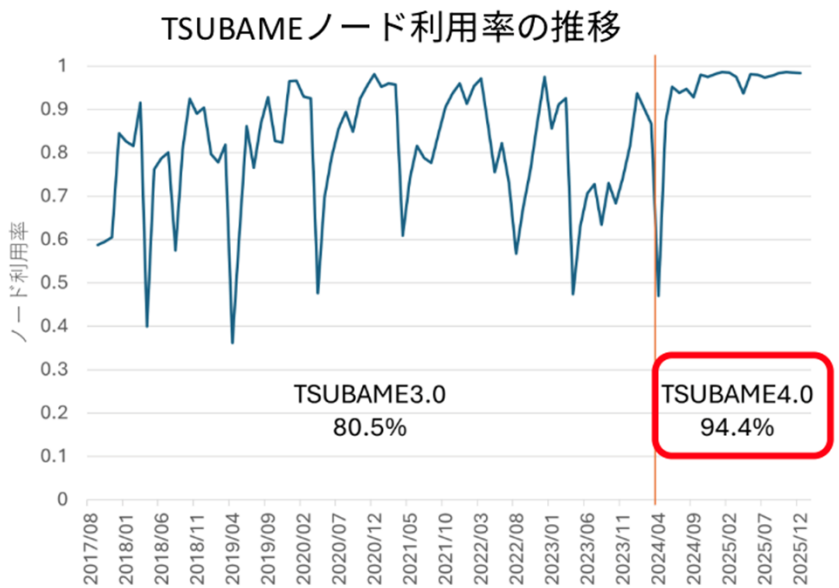
運用予定期間：
2024/4/1～2030/3/31
あと4年間運用予定

TSUBAME4.0の現況

Institute of SCIENCE TOKYO

ポイント

- **高い需要（利用率98.6%）**
  - 古くからGPUユーザコミュニティを形成
  - 現在の資源利用量の4割はAI関係
- 学内利用を優先（学外は30%を上限）しており、
  学外利用分は8月に売切れ(FYR6,R7実績)
  - FYR8分も早期に資源確保される傾向、6月売切見込み
  - 増強分ではこれを緩和し、**学外の資源提供量比率を50%以上に引き上げる**



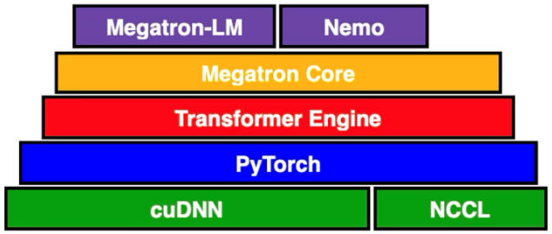
本学の強み

- 2006年より(旧東工大時代含め)アクセラレータ・GPU搭載のTSUBAMEスパコンシリーズを運用
- 蓄積した知見を活かしつつ、TSUBAME4.0を低コスト・高効率で安定運用
  - GPUスパコンの計算資源を無駄なく使いつくす運用モデルを構築済み
    （ノードを分割して、GPUを使わない／低使用のユーザーのジョブを詰め込むことで効率化）
  - 財務面でも、一歩進んだ収益モデルを提案
    （運営経費だけでなく導入コストも利用料金に反映）
- TSUBAME4.0の電力設備・冷却設備に余裕

R7年度の活動成果

Institute of SCIENCE TOKYO

深層学習ソフトウェアスタック整備

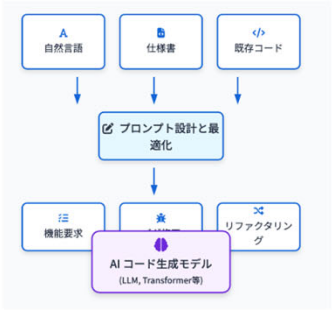


検証済み組合せの標準スタック（PyTorch + cuDNN + NCCL 等）を定義

コンテナで再現可能ビルド（バージョン固定・依存解決）

分散学習テンプレート（SLURM + torchrun など）と実行手順

生成AIを用いたGPU移行の支援



移行ワークフロー確立：コード診断→候補抽出→AI提案→レビュー→CIテスト

Fortran/ループ・カーネルのGPU化プロンプト＆変換パターン集を整備

機密対応：閉域RAG（HPCI文書・事例参照）＋ログ/権限制御

先進的GPU利用のためのソフトウェア技術

深層学習のマルチGPU 3D並列時の性能をさらに改善

低精度演算技術の進展・応用拡大（例：Ozaki Scheme＋コレスキー分解）

GPU性能ポータビリティフレームワークの検証、移植の手間の洗い出し（Kokkos and/or OpenMP target）

GPU環境の利用体制・基盤整備

TSUBAME4.0利用体制
- HAIRDESC共同研究の枠で利用可能

異種GPU実験環境整備
- GH200 NVL2ノード
- Blackwell Max-Q + x86

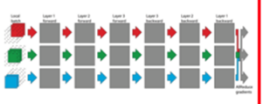
深層学習ソフトウェアスタック整備

Institute of SCIENCE TOKYO

分散並列深層学習

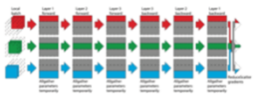
- 多くのAI利用者はノードを跨いだ学習でつまづいている
- これまで：データ並列、テンソル並列、パイプライン並列
- これから：コンテキスト並列、エキスパート並列

データ並列 (DP)



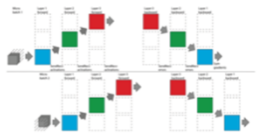
データ：分散
モデル：冗長
通信内容：勾配
通信形式：AllReduce
通信頻度：ステップ毎
長所：実装が簡単
短所：ラージバッチ問題
　　メモリ消費量

ZeRO (FSDP)



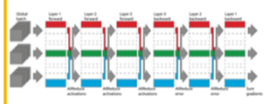
データ：分散
モデル：一時的に分散
通信内容：勾配＋重み
通信形式：ReduceScatter
　　＋AllGather
通信頻度：層毎
長所：実装が簡単
　　省メモリ
短所：ラージバッチ問題

パイプライン並列 (PP)

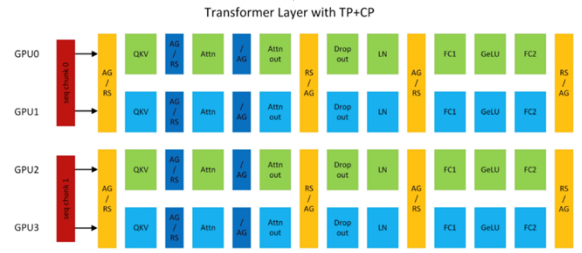


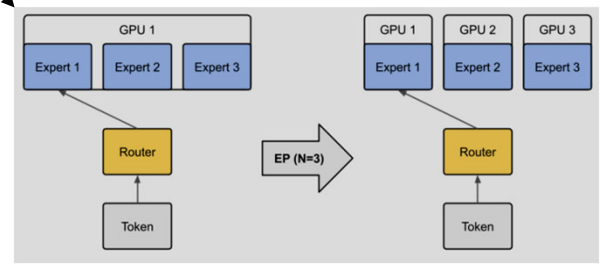
データ：冗長
モデル：分散
通信内容：活性
通信形式：SendRecv
通信頻度：層毎
長所：省メモリ
　　演算量低減
短所：パイプラインバブル

テンソル並列 (TP)

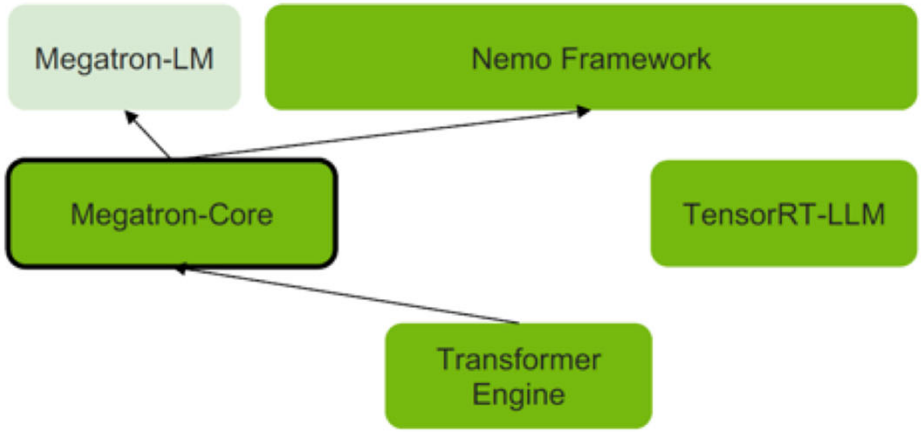


データ：冗長
モデル：分散
通信内容：活性
通信形式：AllReduce
通信頻度：層毎
長所：省メモリ
　　演算量低減
短所：通信オーバーヘッド
　　オーバーラップ不可
　　実装が複雑





分散並列深層学習のフレームワーク



Product Reference

- **NeMo Framework**：

エンタープライズユーザーが実験、学習、デプロイを行える、大規模なモデルコレクションを備えた使いやすいOOTB FW。

- **Megatron-LM**：

Megatron-Coreを使用して独自のLLMフレームワークを構築する為の軽量リファレンスフレームワーク

- **Megatron-Core**：

大規模なTransformerモデルをトレーニングするためのGPU最適化技術用ライブラリ。

- **Transformer Engine**：

Hopper, Ada, Blackwell GPUの為のTransformerモデルのアクセラレーションライブラリ。

- **TensorRT-LLM**：

NVIDIA GPU上の最新の大規模言語モデルで最適な推論パフォーマンスを実現

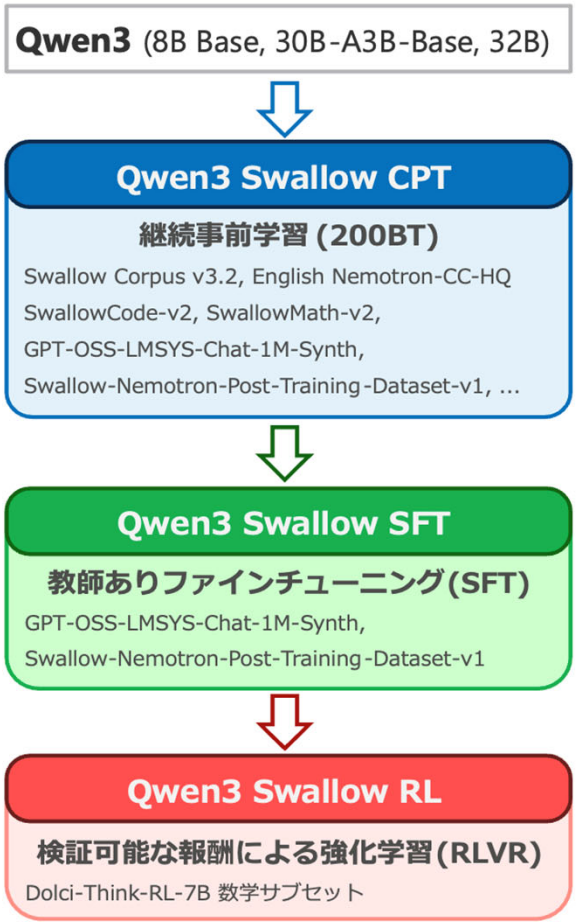
Swallowの学習：CPT → SFT → RLVR

Institute of SCIENCE TOKYO

学習パイプライン

- Qwen3 および gpt-oss を出発点として、継続事前学習（CPT）、教師ありファインチューニング（SFT）、強化学習（RL）の3段階を実施
- 推論型教師 LLM（gpt-oss-120b）で合成した推論過程付き応答をCPTおよびSFTで学習、深い推論を発現させる
- 強化学習で推論能力をさらに改善

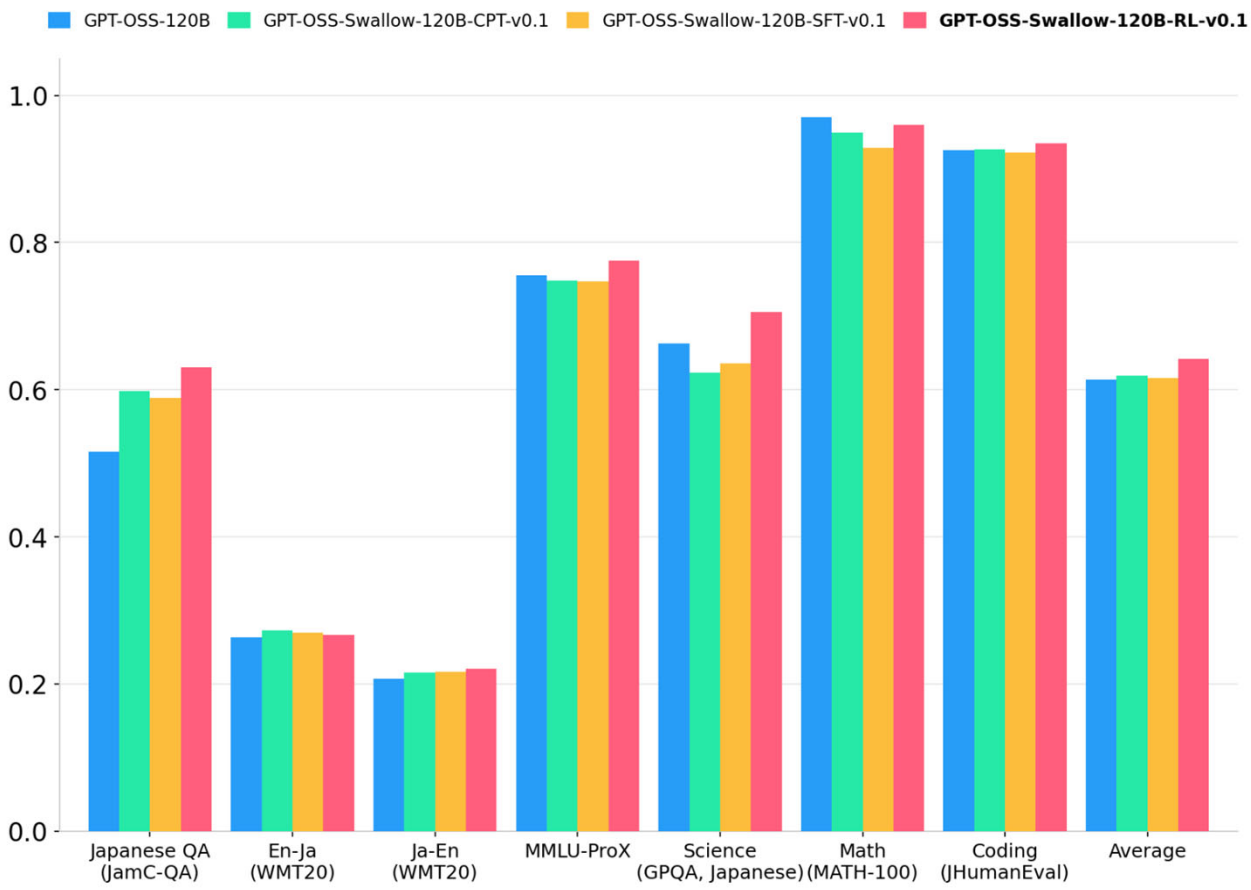
ステージ	学習内容
継続事前学習（CPT）	日英テキスト、QAテキスト、数学・コード、推論過程付き指示応答
指示チューニング（SFT）	汎用対話およびSTEMにおける推論過程付き応答を蒸留
強化学習（RL）	数学設問の正誤による検証可能な報酬



Swallowの成果

2.4M MODEL DOWNLOADS | 551K DATASET DOWNLOADS | 132 MODELS | 19 DATASETS

Institute of SCIENCE TOKYO



GPT-OSS-Swallow

1. **CPT** (Continual Pre-Training)
2. **SFT** (math, code, general)
3. **RLVR** (math)

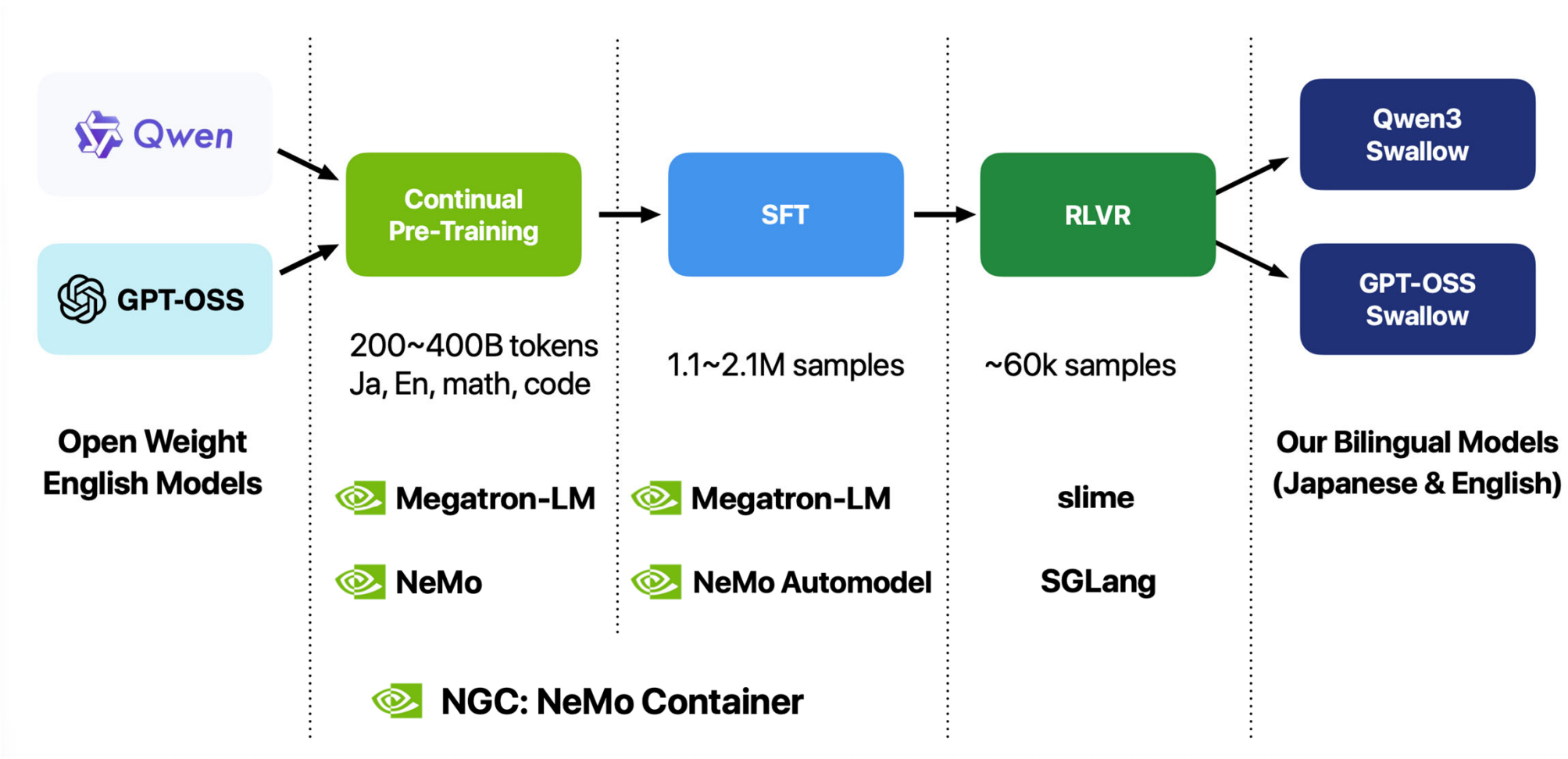
Japanese Perf: **up** ↑

English Perf: **maintained**

GPT-Swallow:
https://swallow-llm.github.io/gptoss-swallow.en.html

Swallowで用いたフレームワーク

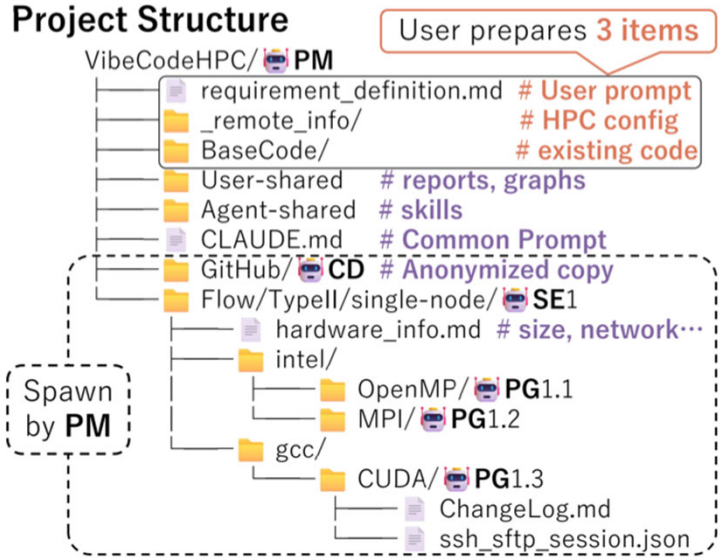
Institute of SCIENCE TOKYO

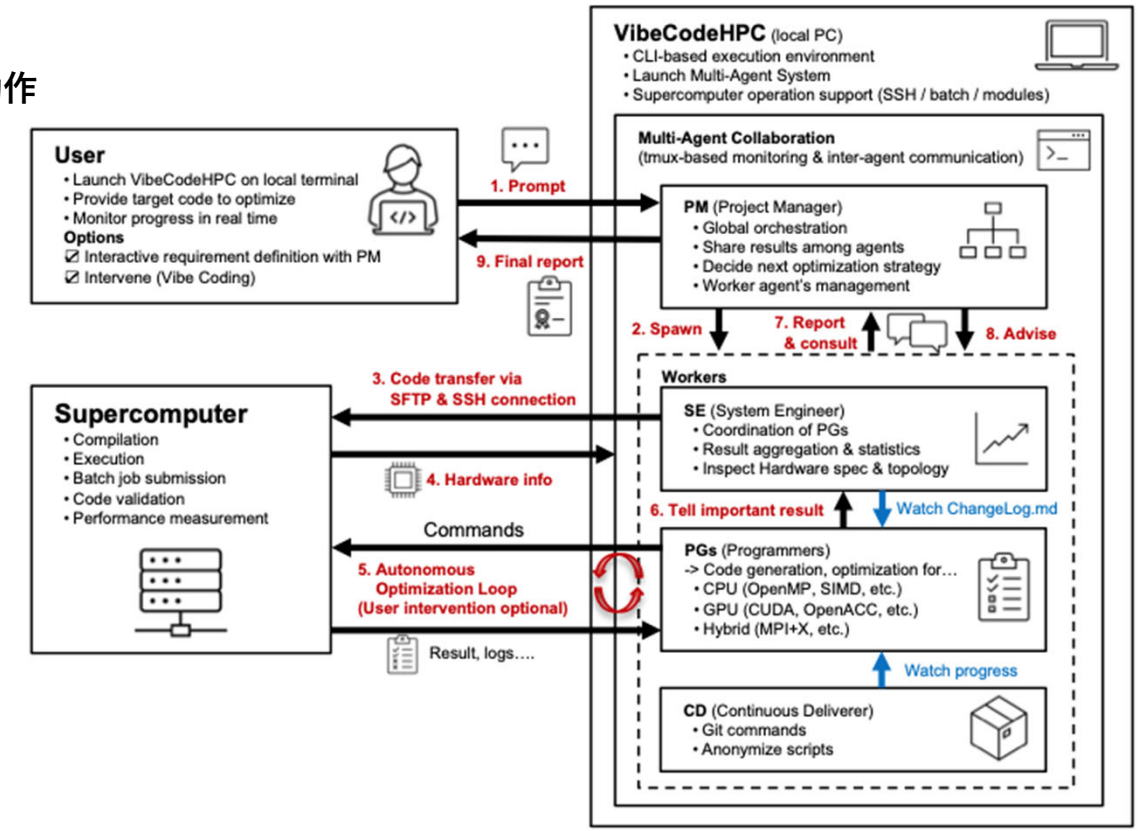


生成AIを用いたGPU移行の支援

Institute of SCIENCE TOKYO

- 複数のAI coding CLIが tmux を使って連携
- 外部オーケストレーションフレームワークなしで動作
- 階層型のマルチエージェント
- 進化型の探索を用いる





https://arxiv.org/abs/2510.00031

生成AIを用いたGPU移行の支援

Institute of SCIENCE TOKYO

コンテキスト圧縮：古い情報を要約・圧縮して新しい処理のための空きを作る
→「cuBLASは禁止」などの制約を忘れないように工夫する必要がある

マルチエージェント、単独エージェントで3回実験

単独エージェントでは

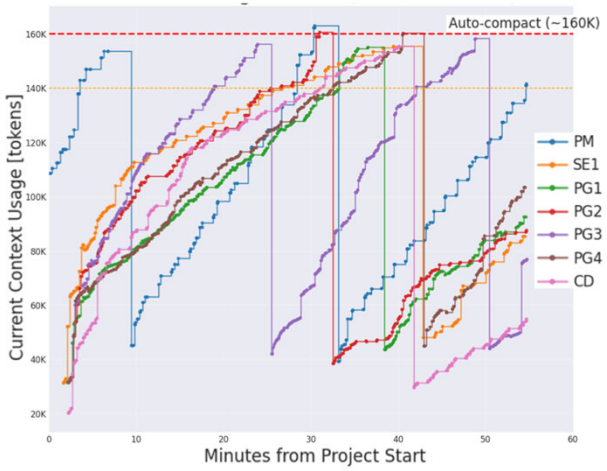
・途中で「もう終わった」と思い込んで探索を止める

・禁止ライブラリ使用を見落とす

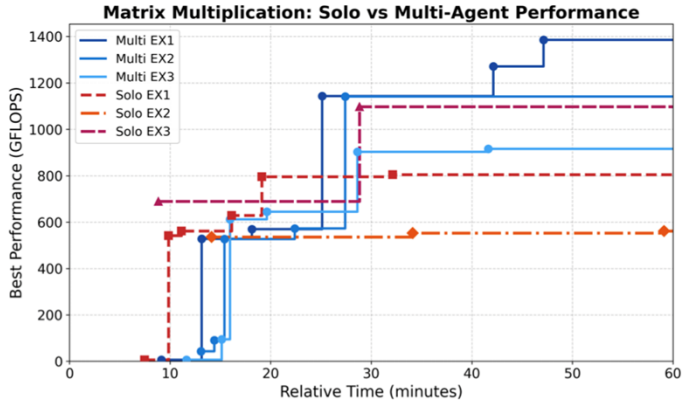
・文脈が散って最適化に集中できない

などの問題が観測された

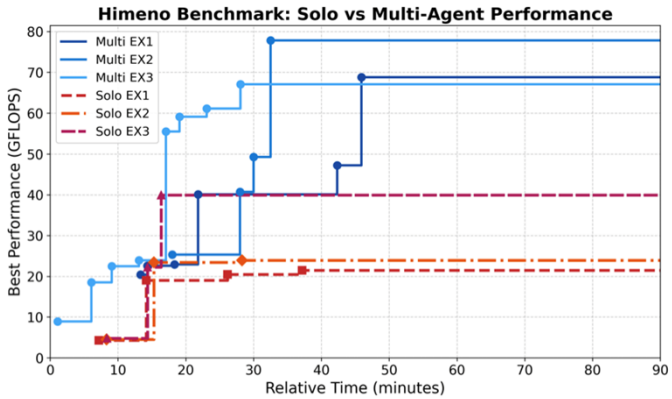
コンテキスト圧縮



行列積の性能向上



姫野ベンチの性能向上



先進GPU利用技術：
ポータビリティフレームワークの検証

Institute of SCIENCE TOKYO

kokkos

- CPUむけ並列プログラムからGPUへ低エフォートで移植する手法の探索
- NVIDIA GPU・AMD GPU・マルチコアCPUに、単一プログラムでの対応を謳うフレームワークが登場している ➔ ORNLなどによるKokkosフレームワークを用いた移植を試行
  - C++ラムダ式をベースに並列化対象コードを記述
  - Kokkos APIの下層にCUDA・ROCm・OpenMPバックエンドを持ち、移植性担保

逐次プログラムをKokkos対応させる変更点の洗い出し

```
    :
double *A;
double *B;
double *C;
    :
int matmul()
{
    const int m_ = m;
    const int n_ = n;
    const int k_ = k;
    const int lda = m_;
    const int ldb = k_;
    const int ldc = m_;

    using ExecSpace = Kokkos::DefaultExecutionSpace;
    using MemSpace = typename ExecSpace::memory_space;

    using HostView = Kokkos::View<double*, Kokkos::HostSpace,
                                  Kokkos::MemoryTraits<Kokkos::Unmanaged
                                      >>;

    HostView hA(A, (size_t)m_ * (size_t)k_);
    HostView hB(B, (size_t)k_ * (size_t)n_);
    HostView hC(C, (size_t)m_ * (size_t)n_);

    Kokkos::View<double*, MemSpace> dA("dA", (size_t)m_ * (size_t)k_)
        ;
    Kokkos::View<double*, MemSpace> dB("dB", (size_t)k_ * (size_t)n_)
        ;
    Kokkos::View<double*, MemSpace> dC("dC", (size_t)m_ * (size_t)n_)
        ;

    Kokkos::deep_copy(dA, hA);
    Kokkos::deep_copy(dB, hB);
    Kokkos::deep_copy(dC, hC);

    Kokkos::parallel_for(
        "matmul",
        Kokkos::MDRangePolicy<ExecSpace, Kokkos::Rank<2>>({0, 0}, {m_
            , n_}),
        KOKKOS_LAMBDA(const int j, const int i) {
            double sum = 0.0;
            for (int l = 0; l < k_; ++l) {
                sum += dA(i + l * lda) * dB(l + j * ldb);
            }
            dC(i + j * ldc) += sum;
        }
    );

    Kokkos::deep_copy(hC, dC);

    return 0;
}
    :
```

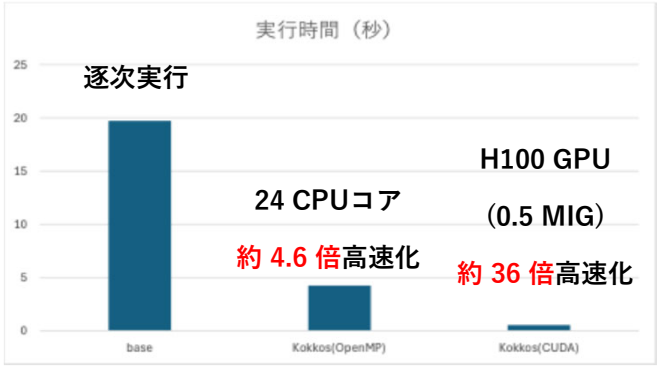
データ転送

大域変数をローカル定数としてコピー

i, j を並列化

[] を() に書き換える

miniWeatherミニアプリ（約900行）をKokkos対応改造（一部AI利用）し、TSUBAME4.0で初期評価



miniWeather の実行時間の比較

先進GPU利用技術：
低精度演算技術の応用発展

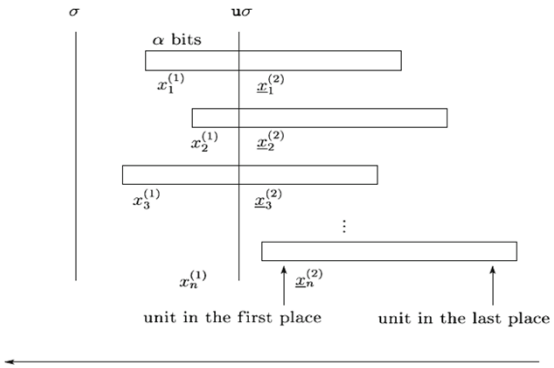
- 最新・将来のGPUアーキテクチャがAIを重視する中、倍精度(FP64)性能が弱い製品が増加

  例：Blackwell Max-QのFP64性能1.7TFlops ⇔FP32の1/64、FP4の1/1024 ☹

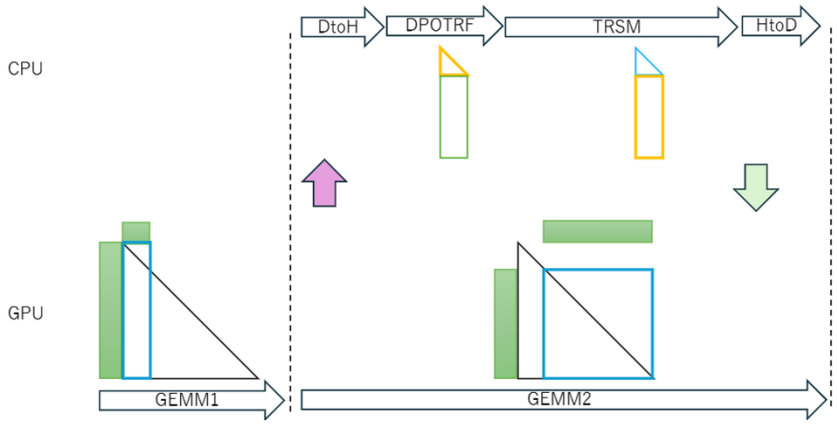
- 低精度演算を利用してFP64相当の行列積を実現する、**尾崎スキームI・II**に注目が集まる

➔ 尾崎スキームを応用した**密行列コレスキー分解演算**の最適化実装を構築中

➔ 半正定値最適化ソルバー等へ統合予定



[Ozaki et al. 2012]より



コレスキー分解中のカーネル種類を分担

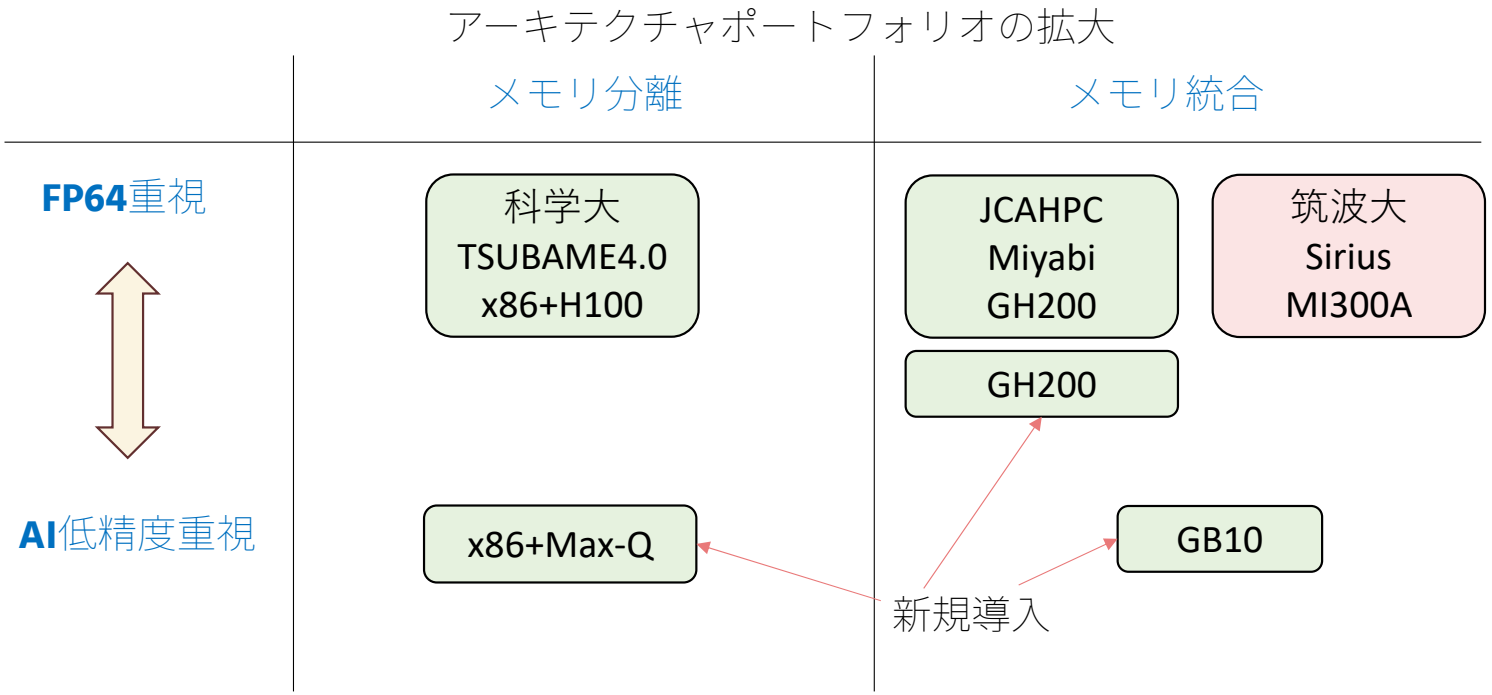
- 行列積カーネル(GEMM) ➔ 低精度GPU + 尾崎スキーム
- POTRF/TRSM ➔ マルチコアCPU上でFP64演算

GPU/CPU演算のオーバラップやブロックサイズ自動調整

GPU環境の環境・利用体制整備

- GPUマシンのアーキテクチャの細分化が発生
  - NVIDIA / AMD / (Intel)
  - メモリ構成：CPU・GPUメモリ分離型 / 統合型
  - 演算精度: FP64重視 / AI低精度重視

多種アーキテクチャに対応した研究開発を推進するため、参加機関のスパコンの利用体制整備に加え、（小規模）異種ノードを導入



GPUプログラミング教材の整備

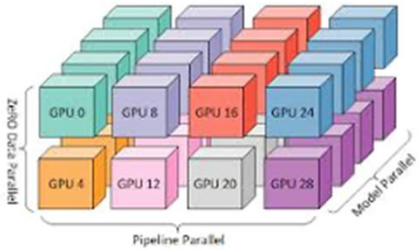
Institute of SCIENCE TOKYO

- R7年度：
  - 初級編（単一GPUプログラミング）の資料確認・コメント
  - 中級編（マルチGPUプログラミング）の資料確認・コメント
  - パフォーマンスチューニング編（プロファイリング）の資料確認・コメント
- R8年度：
  - ライブラリ編の作成をメインで担当
    - BLAS, FFT のようなHPCアプリの実装に多用される計算カーネルを GPU でどのように使いこなせば良いかをコード例を交えながら解説する

R8年度の活動予定

Institute of SCIENCE TOKYO

深層学習ソフトウェアスタック整備

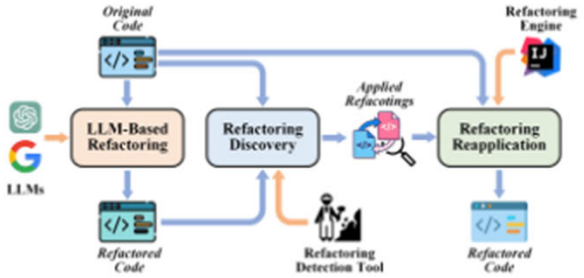


分散並列フレームワークの整備
・データ並列
・テンソル並列
・パイプライン並列
・コンテクスト並列
・エキスパート並列

性能検証ベンチ（通信・I/O・学習スループット）＋公開レポート

日本語トラブルシュート＆サンプルNotebookを整備

生成AIを用いたGPU移行の支援



ClaudeのMCP(エージェント)やSkillなどの機能を駆使した開発

異なる言語への「書き換え」ではなく、元の言語のまま「リファクタリング」を行う

変換後コードの検証テンプレ（単体/回帰/数値誤差/性能回帰）

実証案件を通じた事例化（目安：10件）とガイド公開

先進的GPU利用のためのソフトウェア技術

低精度演算・圧縮技術の進化と応用

Ozaki Scheme
H-Matrix, LoRA
PTQ, 0-1 Adam
：

→

LLM 学習/推論
リアルタイム推論
SDPソルバー
：

AI向け = FP64がほぼないGPUの応用分野を拡大
例：Ozaki SchemeのSDPソルバーへの応用

異種GPU移行の手段探索

多様化するGPUアーキへのソフトウェア移行コストの評価・改良

フレームワーク：Kokkos, Raja, OpenACC
OpenMP target …

×

GPUアーキ：メモリ統合 or メモリ分離
NVIDIA or AMD or …