小口研究室　研究紹介　(2025年度)
(お茶の水女子大学理学部情報科学科)

エッジ環境への適用に向けたWebAssembly runtime上でのLLM推論の性能評価
（研究担当:重松　蒼乃）

研究背景

- LLMの普及：
  クラウド実行にはプライバシー・遅延・コストの課題⇔エッジ環境はデバイスのアーキテクチャが多様
- ➢ アーキテクチャ非依存なバイナリフォーマット **WebAssembly（WASM）** に着目

一度コンパイルすれば異なるCPUアーキテクチャ上で実行可能であり、エッジデバイスの多様性に対応

検証環境

サーバ	FUJITSU PRIMERGY CX2570 M1
OS	Rocky Linux 9.4
CPU	Intel Xeon E5-2697 v3 @ 2.60GHz × 2
コア数	28コア（14コア×2ソケット）
メモリ	128GB
コンテナ	Docker 27.3.1
ベースイメージ	Ubuntu 24.04 LTS

実験概要

WASM環境でのLLM推論におけるオーバーヘッドを定量評価
- ➢ WASMのエッジLLM実行基盤としての実用性を検証

LlamaEdge	性能比較	llama-cpp-python
WasmEdge (WASM runtime) 上でllama.cppを実行 バージョン：WasmEdge 0.16.1 + GGML plugin b7090 プロンプト：llama-3-chat	⟷	python バインディングから llama.cppをネイティブに実行 バージョン： v0.3.16 プロンプト：create_chat_completion()

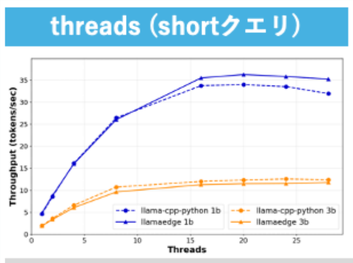
※ llama.cpp： C/C++で実装された軽量なLLM推論ライブラリ

- 検証モデル：
  - Llama-3.2-1B-Instruct-Q5_K_M.gguf
  - Llama-3.2-3B-Instruct-Q5_K_M.gguf
- 検証パラメータ：

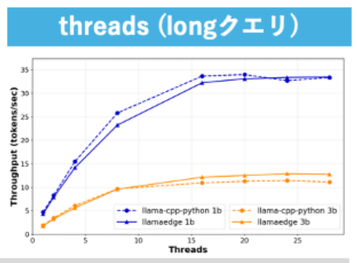
テスト名	検証値	クエリ
threads (short)	[1,2,4,8,16,20,24,28]	What is machine learning?
threads (long)	[1,2,4,8,16,20,24,28]	Explain the history of machine learning, its main algorithms, applications, and future trends in detail.
ctx-size	[128,256,512,1024,2048, 4098,8192,16384]	Explain the history of machine learning, its main algorithms, applications, and future trends in detail.

検証結果

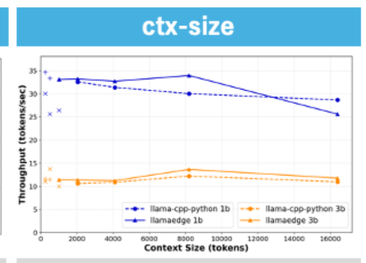
threads (shortクエリ)



threads (longクエリ)



ctx-size



llama-cpp-pythonに対し、LlamaEdgeが約0.87倍～1.11倍程度の性能差
- 1B shortや3B longのthreads≥16：LlamaEdgeの性能が上回る
- llama-cpp-pythonのcontext-switchesはLlamaEdgeの約4.7倍 (threads=16)
- ➢ OpenMPのスリープ移行閾値であるGOMP_SPINCOUNTを変動させる検証により、token間のOpenMPスレッドのスリープ移行が原因と特定

バインディング層の処理コストの差がtoken間の待機時間に影響し、スリープ移行の有無を左右している可能性

LlamaEdgeが多くの条件で優勢
- ctx=16384でllama-cpp-pythonが逆転
- 性能低下時にIPC低下を確認
- ➢ メモリ配置の偏りが要因の可能性

単一NUMAノード構成のエッジ環境における影響は限定的

まとめと今後の課題

- WASMを用いたLLM推論(LlamaEdge)とネイティブ実行(llama-cpp-python)の性能比較を実施
- WASMランタイム自体のオーバーヘッドは1.0～1.3倍に収まることを確認
  - ➢ バインディング層やメモリ配置の最適化により、さらなる性能改善の余地がある
- **WASMはエッジ環境でのLLM推論基盤として十分実用的**
- 今後はエッジデバイス上やオーケストレーション基盤上での検証を行う

NPB を用いたHPC 環境におけるSingularity コンテナ性能の調査
（研究担当:山口　夢津美）

研究背景

- AI をはじめとする計算需要の増加に伴い，高性能計算 (HPC) の重要性が高まっている
- アプリケーションが開発されるソフトウェア環境の複雑化が進むにつれ，柔軟性の高いHPCソフトウェア環境への需要も増加
- HPC環境において移植性，柔軟性，再現性の高い実行環境を実現する技術として，コンテナ技術が注目されている
- コンテナは仮想化技術の一種であるためHPCのベアメタル環境に比べて性能劣化の可能性が懸念されている

▶ Singularityを使用したA64FX上にて性能の定量的評価を行い
HPC分野におけるコンテナの実用可能性を明らかにする

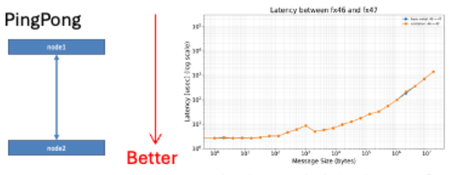
実験概要

- 使用ベンチマーク
- Intel MPI Benchmarks (IMB)
  - 通信性能を調査するマイクロベンチマーク
  - MPI-1 Benchmarksを使用
- NAS Parallel Benchmarks (NPB)
  - 異なる特徴を持つ並列アプリケーションのベンチマークプログラム群
  - NPB3.4.3とNPB3.4.3-MZを使用
- 実行環境
- Bare metal: ベアメタル
- container: コンテナ
- cpbin_container: コンテナ（ベアメタルでコンパイル, NPBのみ）

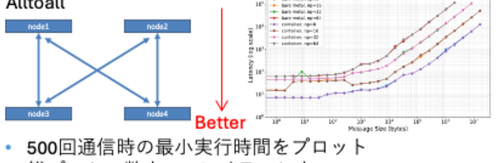
実行環境	
アーキテクチャ	A64FX (aarch64) x 8ノード
OS	Rocky Linux 8.10 (Bare metal) Rocky Linux 8.9 (container / cpbin_container)
通信ライブラリ	Open MPI 4.1.6, UCX
インターコネクト	Infiniband EDR 100Gbps

Intel MPI Benchmarks (IMB) 実行結果

- 性能指標とベンチマークの特徴
  - メッセージサイズは0から2^24bytes
  - 通信回数を各メッセージあたり500回に設定
- PingPong



  - fx46-47間での500回通信時の平均実行時間をプロット
  - ベアメタルとコンテナで性能差はほとんどない
  - 2MiBbytesでコンテナ性能わずかに劣化
- Alltoall



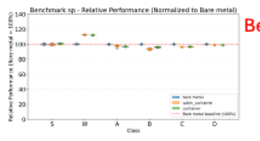
  - 500回通信時の最小実行時間をプロット
  - 総プロセス数大　➔レイテンシ大
  - わずかに性能差が生じることがある

実験概要

- 性能指標とベンチマークの特徴
  - 10回連続実行した時の Mop/s total の平均値
  - ベアメタルでの性能を100とした時の相対評価を使用
  - コンテナ起動時間は含まない

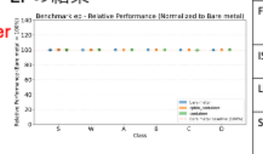
ベンチマーク名	特性	インテンシブ
EP	- 通信がほとんどない	- CPUインテンシブ
MG	- 構造化された長距離通信が必要	- ネットワークインテンシブ - メモリインテンシブ
CG	- 共役勾配法 - メモリの不規則アクセス	- ネットワークインテンシブ - メモリインテンシブ
FT	- 高速フーリエ変換を用いた，3次元偏微分方程式の解法	- ネットワークインテンシブ
IS	- 大型整数ソート - ランダムメモリアクセス	- ネットワークインテンシブ - メモリインテンシブ
LU	- CFDのwavefront法 - 小さなデータを多数回通信	- ネットワークインテンシブ(レイテンシ)
SP/BT	- CFDのマルチパーティションアルゴリズム - SPはスカラ計算，BTはブロック計算	- ネットワークインテンシブ(帯域幅)
LU-MZ	- LUのマルチゾーン版 - プロセス間通信の他にスレッド並列を使用	- ネットワークインテンシブ(レイテンシ) - 負荷均衡インテンシブ
SP-MZ/BT-MZ	- SP/BTのマルチゾーン版 - プロセス間通信の他にスレッド並列を使用	- ネットワークインテンシブ(帯域幅) - CPUインテンシブ(主にBT)

- SPの結果



Better

- EPの結果



- SP考察
  - クラスWでコンテナ２種類の性能が良い
  - クラスA，B，Cにてコンテナ２種類の性能が劣化
- EP考察
  - 3つの環境間の差なし
  - 10回実行におけるばらつきなし

➡ 演算処理能力に差はない

- 全体考察
  - cpbin_containerとcontainerの差はない
    ➔コンパイル環境が性能に与える影響なし
  - コンテナ環境の性能が上回る場合の追加調査が必要

まとめと今後の課題

- まとめ
- IMB
  - メッセージサイズ大 → 性能差縮小 の傾向あり
  - pingpong・alltoallでは環境間の性能差がほとんどない
- NPB
  - クラスが大きくなると性能差縮小
  - ネットワーク＋メモリインテンシブ
  - → コンテナ環境の性能が劣化する傾向
- 今後の課題
  - 性能差が生じているベンチマークの原因調査
  - ファイル I/O など他の要素に着目した評価を実施
  - コンテナのような仮想環境で発生しうる性能ポータビリティの検証
  - より実用的なアプリケーションプログラムを用いた評価

エッジデバイスで活用できる小規模言語モデルの実装に向けた検討
（研究担当:川村　碧葵）

研究背景

■近年モバイルフォンやIoT機器などのエッジデバイスが広く普及、個人のデータが日常的に記録・蓄積
■個人情報を漏洩リスクのある外部サーバに送信せずに、端末内で処理することが求められている
■情報漏洩対策やデータ主権の観点から、海外のストレージサービスを利用せず、国内でデータ保存をする動きが進んでいる
■エッジデバイスの性能向上により、エッジデバイス上でAIモデルが利用可能
■LLM(大規模言語モデル)は高性能だが、膨大な計算資源と大規模なデータセットを必要とし、エッジデバイス環境での活用には課題がある
■軽量化された言語モデルSLM(小規模言語モデル)は限られた環境でも高精度な自然言語処理が可能
■エッジデバイス上で個人情報を活用する小規模言語モデルの実装に向けた検討を行う

ファインチューニングとRAG

■ ファインチューニング
- 大規模言語モデルに特定分野の情報をさらに学習させることによって、モデルを構築
- 特定分野の専門知識をモデル内に取り込むことでより専門的な問い合わせへの精度向上が期待できる
- 分類チューニングは特定のクラスラベルを認識するように訓練。訓練中に遭遇したモデルの予測に限定
- インストラクションチューニングは特定の指示を使った一連のタスクで言語モデルを訓練。自然言語のプロンプトで表されたタスクを理解して実行する能力を向上

■ RAG
- 外部の知識源から関連文書を検索し、その情報を基にLLMが応答を生成する技術
- LLM単体では対応が難しい最新情報や非公開文章に基づく質問応答が可能
- LLMが事実に基づかない情報を生成するハルシネーションの抑制にも効果がある
- チャンク分割方法や検索精度に応答品質が大きく依存するため、ベクトルデータベース設計や検索戦略の最適化が重要
- ベクトル検索はデータを数値のベクトルに変換し類似度を計算する手法。調べたい情報が明確な場合は関連情報が妨げになり、目的にたどり着くのが遅くなる可能性あり
- キーワード検索はデータベースの中にある全ての文字を対象として、キーワードを検索する手法。文字列と一致する内容を探しているだけで、言葉を理解している訳ではない
- ハイブリッド検索は複数の検索方法を組み合わせることで、検索結果の精度や関連性を向上させる手法

データについて

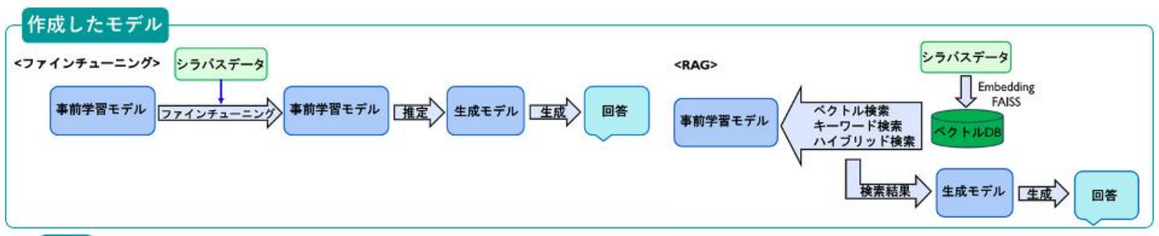
■ 使用したデータについて
- 鈴鹿工業高等専門学校令和6年度の5学科分pdf版シラバスから614科目の情報を使用
- 全データはデータ項目を絞っていない5,400KBのデータ。データ項目は「学校名,開校年度,授業科目，科目番号，科目区分，授業形態，単位の種別と単位数，開設学科，対象学年，開設期，週時間数，教科書/教材，担当教員，到達目標，ルーブリック，概要，授業の進め方・方法，注意点，授業の属性・履修上の区分，授業計画，モデルコアカリキュラムの学習内容と到達目標，評価割合」
- 限定データはデータ項目を絞った383KBのデータ。データ項目を絞ったシラバスデータの項目は「授業科目，開設学科，教科書，担当教員，到達目標」
- Accuracy(正解率)を求める為に、「OOの担当教員名は誰ですか？」という問い合わせと担当教員名のファイルを作成

実験

■ローカルデータに対するファインチューニングとRAGの評価実験
● インターネット上に上がっているシラバスデータをローカルデータとして使用し「OOの担当教員は誰ですか？」という問い合わせに対して、担当教員名を生成
● データ項目を絞った場合と絞らなかった場合に対して、ファインチューニング手法やRAGの検索方法・検索結果の上位何件を用いて回答を生成するか、の違いのAccuracy(正解率)と実行時間を調べて比較した

1. ファインチューニングの評価実験
- 文章分類モデルとして、日本語事前学習モデルであるcl-tohoku/bert-base-japanese-v3、モデルの実装及び学習にはHuggingFace Transformers及びPyTorchを使用
- 分類チューニングでは、入力をシラバス本文とし、出力を担当教員名とする教師あり分類タスクとして定式化。教員名はLabelEncoderにより、整数ラベルへ変換し、分類問題としてファインチューニング
- インストラクションチューニングでは、各シラバス本文の先頭に「次のシラバスから担当教員名を答えてください。」という自然言語の指示文を付与し、指示付き入力としてモデルに与えた
2. RAGの評価実験
- 埋め込み表現には、日本語事前学習モデルであるcl-tohoku/bert-base-japanese-v3を使用
- モデルの推論処理にはPyTorchを用い，トークン埋め込みの平均プーリングによって文書ベクトルを生成
- 検索結果を基に，Ollamaを用いてローカル環境で日本語対応のLLM，lucas2024/gemma-2-2b-jpn-it:q8_0で回答を生成

作成したモデル



結果

■ローカルデータに対するファインチューニングとRAGの評価実験
1. ファインチューニングの評価実験
- 表１より学習していない場合や分類チューニングでは正解率が０であったが、インストラクションチューニングを行うと数件正解したと分かる。また、データ量が少ない方が正解率が高かった
2. RAGの評価実験
- 表2はデータ項目を絞っていないシラバスデータに対するRAGの結果であるのに対し、表３はデータ項目を絞ったシラバスデータに対するRAGの結果である
- 表2,3でベクトル検索の方が多くの条件で高い正解率を示していたため、ベクトル検索:キーワード検索＝0.7:0.3の重みを設定し、ハイブリッド検索スコアリングを再査定した結果が表4である
- 表４より重み付けを行ったハイブリッド検索では、多くの条件で正解率が向上したが、元々ハイブリッド検索の方が正解率が高かった条件では、重み付けによりわずかに正解率が低下する場合があった

表1：ファインチューニングの結果

手法	Accuracy(正解数/総数)
学習していない結果	0.0000 (0/124)
分類チューニング 全データ	0.0000 (0/124)
分類チューニング 限定データ	0.0000 (0/124)
インストラクションチューニング 全データ	0.0323 (4/124)
インストラクションチューニング 限定データ	0.0565 (7/124)

表2：全データを用いた場合のRAGの結果

k	検索手法	Accuracy (正解数/総数)	実行時間[s]
3	ベクトル検索	0.0161(2/124)	298.32
	キーワード検索	0.0242(3/124)	297.71
	ハイブリッド検索	0.0081(1/124)	303.91
5	ベクトル検索	0.0242(3/124)	304.71
	キーワード検索	0.0081(1/124)	296.57
	ハイブリッド検索	0.0081(1/124)	305.42
10	ベクトル検索	0.0323(4/124)	309.46
	キーワード検索	0.0000(0/124)	300.94
	ハイブリッド検索	0.0000(0/124)	315.47
15	ベクトル検索	0.0484(6/124)	317.66
	キーワード検索	0.0000(0/124)	309.95
	ハイブリッド検索	0.0403(5/124)	330.08

表3：限定データを用いた場合のRAGの結果

k	検索手法	Accuracy (正解数/総数)	実行時間 [s]
3	ベクトル検索	0.2258(28/124)	74.80
	キーワード検索	0.0323(4/124)	37.89
	ハイブリッド検索	0.2177(27/124)	82.77
5	ベクトル検索	0.2581(32/124)	94.35
	キーワード検索	0.0403(5/124)	41.45
	ハイブリッド検索	0.2984(37/124)	113.90
10	ベクトル検索	0.3145(39/124)	140.72
	キーワード検索	0.0565(7/124)	42.04
	ハイブリッド検索	0.3065(38/124)	194.68
15	ベクトル検索	0.3145(39/124)	192.42
	キーワード検索	0.1532(19/124)	49.03
	ハイブリッド検索	0.2016(25/124)	280.30

表4：ベクトル検索とキーワード検索の重みを0.7:0.3としたハイブリッド検索によるRAGの結果

k	検索手法	シラバスデータ Accuracy (正解数/総数)	データ項目を絞ったシラバスデータ Accuracy (正解数/総数)
3	ベクトル検索	0.0161(2/124)	0.2258(28/124)
	キーワード検索	0.0242(3/124)	0.0323(4/124)
	ハイブリッド検索	0.0161(2/124)	0.2258(28/124)
5	ベクトル検索	0.0242(3/124)	0.2581(32/124)
	キーワード検索	0.0081(1/124)	0.0403(5/124)
	ハイブリッド検索	0.0242(3/124)	0.2581(32/124)
10	ベクトル検索	0.0323(4/124)	0.3145(39/124)
	キーワード検索	0.0000(0/124)	0.0565(7/124)
	ハイブリッド検索	0.0323(4/124)	0.3145(39/124)
15	ベクトル検索	0.0484(6/124)	0.3145(39/124)
	キーワード検索	0.0000(0/124)	0.1532(19/124)
	ハイブリッド検索	0.0484(6/124)	0.3226(40/124)

まとめと今後の課題

まとめ
■今回のようにデータ量が少ないローカルデータを活用する場面では、モデルを再学習させるファインチューニングよりも、RAGにより外部知識として参照させる方が、情報を有効的に活用できると考えられる

今後の課題
■より多いデータ量でファインチューニングを行い、データ量によるファインチューニングとRAGの特性についてさらに実験を行う
■それらを踏まえてエッジデバイス上で個人情報を活用できるLLMを実装し、評価

https://www.is.ocha.ac.jp/~oguchi_lab/　　oguchi@is.ocha.ac.jp