AI PC により大規模な LLM 開発がデスクトップで可能に

700 億のパラメーターを持つ LLM をクラウドだけで開発しない

Tony Mongkolsmai インテル コーポレーション ソフトウェア・アーキテクト / テクニカル・エバンジェリスト

大規模言語モデル（LLM）の普及により、計算能力に対するニーズが高まっています。OpenAI、Anthropic などの強力なソリューションはクラウドで実行されますが、小規模なシステムで実行できるモデルも増えています。ローカルで実行できるかどうかは、計算能力とメモリー容量の両方に依存します。モデルには通常、モデルの大きさと計算の複雑さを示すパラメーター・サイズがあります。パラメーター・サイズが大きくなると、ユーザーは計算能力の要件だけでなく、メモリー要件の課題にも直面します。トレーニングや推論を効率良く実行するには、LLM をデバイス（通常は GPU）のメモリーにロードする必要があります。コンシューマー向け GPU のメモリーは最大 24GB に制限されており、その多くは 16GB 未満のメモリーを搭載しています。企業が計算能力の要件とモデル精度のバランスを取れるように、Meta の Llama* モデルと Microsoft の Phi モデルにはさまざまなパラメーター・サイズが用意されています。例えば、Hugging Face* では Llama* モデルが最も人気があります。Llama* 2 には 7B、13B、70B のサイズがあり、Llama* 3 には 8B と 70B のサイズがあります。7B、8B、および 13B モデルは、多くのハイエンドのコンシューマー GPU で量子化と最適化を使用して実行できます。70B モデルは通常、コンシューマー GPU には大きすぎます。

大きな LLM を使用してローカルで開発する

一般に私は、ソリューションを開発するためデータセンターやクラウドにログインすることは好みません。ローカルシステムで開発するほうがレイテンシーとセキュリティーの面では優れていますが、LLM のサイズと計算能力の要件がローカル開発の妨げとなります。この課題を解決する 1 つのオプションは、エンタープライズ・グレードの GPU を搭載したワークステーションを導入することですが、コストが高くなる可能性があります。幸い、インテル® Core™ Ultra プロセッサー・ソリューションを出荷しているスモール・フォームファクター・ベンダーが解決策を提供しています。

私は最近、インテル® Arc™ 統合 GPU（iGPU）内蔵のインテル® Core™ Ultra 155H プロセッサーを搭載した Asus* NUC 14 Pro システムを購入しました。このシステムの斬新な点は、iGPU がシステム RAM の最大半分を GPU メモリーとして使用できることです。最大 96GB の DDR5-5600 DRAM で構成すると、システムはワークステーションの数分の 1 のコストで最大の 70B Llama* モデルを実行できます。

システムの設定

ハードウェア・セットアップ

Asus* NUC 14 Pro のセットアップは簡単でした。48GB DDR5-5600 SODIMM DRAM モジュール 2 本を差し込み、M.2 20x80 NVMe* SSD を取り付けるだけでした。テストでは Windows* と Linux* をデュアルブートにしたいと考えており、Meta* Llama* 3 70B モデルは約 550GB のディスク容量を必要とするため、少なくとも 4TB の SSD が必要でした。ハードウェアのセットアップにかかった時間は 5 分未満です。

オペレーティング・システムのインストール

オペレーティング・システムのインストールも比較的簡単でした。まず、Windows* 11 をインストールしましたが、これにはハードウェア・イーサネット接続が必要でした。Wi-Fi* はそのままでは機能しないため、ネットワーク・インストールの回避策はありませんでした。Windows* をインストール後、Asus のウェブサイトから最新のドライバーをダウンロードしてインストールし、Windows* Update を行いました。

次に、Ubuntu* 22.04.4 をインストールしましたが、問題なく動作しました。Linux* を起動後、指示（英語）に従って最新のユーザーモード GPU ドライバーをインストールしました。

ヒント：Windows* と Linux* からアクセスできる比較的大きな共有ドライブを用意するとよいでしょう。Windows* と Linux* で LLM をテストする場合、モデルを共有ドライブに配置し、同じモデルが両方のパーティションでスペースを占有しないようにする必要があります。

LLM の実行

LLM のソフトウェア・スタックは急速に変化しており、さまざまな選択肢があります。ここでは、従来の Hugging Face*/PyTorch* ワークフローと、人気の llama.cpp オープンソース・フレームワークを使用して Llama* 3 70B をテストしますが、将来的にはより大規模なデータセンター・システムで実行することを想定しています。ここでは、Windows* のワークフローについて説明しますが、このプラットフォームとソフトウェア・スタックは Linux* でも動作します。

基本構成

モデル実行の AI 固有の部分に入る前に、Windows* の環境を準備する必要があります。いくつかのパッケージをインストールします。

1. Windows* Visual Studio* 2022 Community Edition
2. conda-forge（英語）
3. インテル® oneAPI ベース・ツールキット

PyTorch* で実行

PyTorch* を使用して LLM を実行するには、通常、Hugging Face*（英語）にあるオープンソース・ライブラリー、API、モデルを使用します。

PyTorch* と Hugging Face* API のインストール

PyTorch* 環境と必要な Hugging Face* ライブラリーをセットアップするには、`conda-forge` Miniforge プロンプトを起動し、次のコマンドを実行します。

```
# conda-forge で conda 環境を作成して有効化
conda create -n llm python=3.11 libuv
conda activiate llm
# PyTorch* および Hugging Face* ライブラリーとインテルの PyTorch* 向け LLM ライブラリーをインストール
pip install --pre --upgrade ipex-llm[xpu] --extra-index-url
https://pytorch-extension.intel.com/release-whl/stable/xpu/us/
```

ここでは、インテル® GPU 上で Hugging Face* API のより高性能な実装を提供するインテルの PyTorch* LLM ライブラリーを使用して依存関係をインストールしました。ライブラリーの依存関係ツリーは、PyTorch* とその他の必要な Hugging Face* ライブラリーもインストールします。

モデルの取得

ここでは、Hugging Face* からダウンロード可能な最新の Llama* 3 モデルである、Llama* 70B をテストします。Llama* モデルは Meta からの承認が必要なので、https://huggingface.co/meta-llama/Meta-Llama-3-70B（英語）からアクセスを申請し、ライセンスに同意する必要があります。承認されると、次の操作を実行できます。

```
# Hugging Face* CLI のインストール
pip install huggingface-cli
# CLI を使用して Hugging Face* にログイン
git clone https://huggingface.co/meta-llama/Meta-Llama-3-70B
```

前述のように、モデルはかなり大きく、551GB のディスク容量を消費するため、ダウンロードには時間がかかる場合があります。

推論の実行

PyTorch* と Hugging Face* API を使用して推論を実行するのは簡単です。ほかの GPU スクリプトとわずかに異なる部分があることが分かります (緑色でハイライトされた箇所)。ここでは、インポートにインテルのライブラリーが使用されており、`to` 関数のターゲットデバイスは「`cuda`」ではなく「`xpu`」です。

```
from ipex_llm.transformers import AutoModelForCausalLM
```

```
...

model = AutoModelForCausalLM.from_pretrained(model_path,
      load_in_4bit=True,
      optimize_model=True,
      trust_remote_code=True,
      use_cache=True)

# モデルを GPU アクセラレーターに移動
model = model.half().to('xpu')

# トークナイザーをロード
tokenizer = AutoTokenizer.from_pretrained(model_path, trust_remote_code=True)

...

# 予測トークンを生成
with torch.inference_mode():
      prompt = get_prompt(args.prompt, [], system_prompt=DEFAULT_SYSTEM_PROMPT)
      input_ids = tokenizer.encode(prompt, return_tensors="pt").to('xpu')

      # 推論を開始
      output = model.generate(input_ids,
            eos_token_id=terminators,
            max_new_tokens=args.n_predict)
      torch.xpu.synchronize()
      output = output.cpu()
      output_str = tokenizer.decode(output[0], skip_special_tokens=False)
```

完全なコードはこちら（英語）を参照してください。スクリプトの実行は簡単です。次に示すように、いくつかのパラメーター（モデル、プロンプト、出力サイズなど）を渡します。

```
python llama3.py --repo-id-or-model-path D:\models\Meta-Llama-3-70B-Instruct --prompt "Tell
me if or why AI is important to the future of humanity" --n-predict 51
```

```
(llm) C:\Users\tonym\dev>python llama3.py --repo-id-or-model-path D:\models\Meta-Llama-3-70B-Instruct --prompt "Tell me if or why AI is impor
tant to the future of humanity" --n-predict 512
C:\Users\tonym\miniforge3\envs\llm\Lib\site-packages\torchvision\io\image.py:13: UserWarning: Failed to load image Python extension: 'Could n
ot find module 'C:\Users\tonym\miniforge3\envs\llm\Lib\site-packages\torchvision\image.pyd' (or one of its dependencies). Try using the full
path with constructor syntax.'If you don't plan on using image functionality from `torchvision.io`, you can ignore this warning. Otherwise, t
here might be something wrong with your environment. Did you have `libjpeg` or `libpng` installed before building `torchvision` from source?
  warn(
2024-05-31 11:04:40,189 - INFO - intel_extension_for_pytorch auto imported
Loading checkpoint shards: 100%|████████████████████████████████████████████| 30/30 [00:02<00:00, 11.49it/s]
2024-05-31 11:04:43,305 - INFO - Converting the current model to sym_int4 format.....
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attent
ion_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attent
ion_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
Inference time: 322.889187097549444 s
-------------------- Prompt --------------------
<|begin_of_text|><|start_header_id|>user<|end_header_id|>

Tell me if or why AI is important to the future of humanity<|eot_id|><|start_header_id|>assistant<|end_header_id|>


---------- Output (skip_special_tokens=False) ----------
<|begin_of_text|><|begin_of_text|><|start_header_id|>user<|end_header_id|>

Tell me if or why AI is important to the future of humanity<|eot_id|><|start_header_id|>assistant<|end_header_id|>

What a great question! Artificial Intelligence (AI) is indeed crucial to the future of humanity, and here's why:

**1. Solving Complex Problems**: AI can help tackle some of humanity's most pressing challenges, such as:
        * Climate change: AI can analyze vast amounts of environmental data to identify patterns and predict climate-related events.
        * Healthcare: AI can aid in disease diagnosis, personalized medicine, and drug discovery.
        * Energy: AI can optimize energy consumption, predict energy demand, and improve energy storage.

**2. Improving Efficiency and Productivity**: AI can automate repetitive tasks, freeing humans to focus on creative, high-value work, leading
 to:
        * Increased productivity: AI can streamline processes, reducing time and resources spent on mundane tasks.
        * Enhanced customer experience: AI-powered chatbots and virtual assistants can provide 24/7 support.

**3. Enabling New Industries and Jobs**: AI is creating new opportunities and transforming existing ones, such as:
        * Autonomous vehicles: AI is driving the development of self-driving cars, trucks, and drones.
        * Cybersecurity: AI-powered systems can detect and respond to threats more effectively than traditional methods.
        * Data Science: AI has created a new field of data analysis and interpretation.

**4. Enhancing Human Life**: AI can improve the quality of life for individuals, particularly:
        * Elderly care: AI-powered systems can assist with daily tasks, provide companionship, and monitor health.
        * Accessibility: AI-driven tools can help people with disabilities, such as speech-to-text systems and image recognition.

**5. Space Exploration and Colonization**: AI will play a critical role in:
        * Spacecraft navigation: AI can optimize routes, detect anomalies, and make autonomous decisions.
        * Planetary colonization: AI can help establish and maintain human settlements on other planets.

**6. Cybersecurity and Defense**: AI can help protect against cyber threats, detect anomalies, and enhance national security:
        * Threat detection: AI-powered systems can identify potential threats and respond in real-time.
        * Autonomous defense: AI can enable autonomous defense systems to respond to threats without human intervention.

**7. Education and Skills Development**: AI can revolutionize education by:
        * Personalized learning: AI can tailor educational content to individual students' needs and abilities.
        * Skills development: AI can help identify skill gaps and provide targeted training recommendations.
```

Python* 出力には、モデル・チェックポイントのロード、モデルの量子化、モデルからのプロンプトと出力の表示など、Hugging Face* ユーザーが期待するすべての標準的な情報が表示されます。

llama.cpp で実行

llama.cpp（英語）プロジェクトは、純粋な C++ 実装を使用して LLM 推論を可能にします。さまざまな GPU とフレームワークのバックエンドをサポートしています。テストには SYCL* バックエンドを使用します。

ビルド

最初に、リポジトリーをクローンします。

```
git clone https://github.com/ggerganov/llama.cpp.git
```

次に、依存関係の cmake（英語）、mingw-w64（英語）、および インテル® oneAPI ベース・ツールキットをインストールします。依存関係をインストールしたら、SYCL* を使用して llama.cpp を簡単にビルドできます。

```
# oneAPI 環境の初期化
"C:\Program Files (x86)\Intel\oneAPI\setvars.bat" intel64
# llama.cpp ビルドの設定
cmake -B build -G "MinGW Makefiles" -DLLAMA_SYCL=ON -DCMAKE_C_COMPILER=icx
-DCMAKE_CXX_COMPILER=icx -DCMAKE_BUILD_TYPE=Release
# llama.cpp 実行ファイルのビルド
cmake --build build --config Release -j
```

バイナリーは `build/bin/main` に出力されます。

モデルの取得

llama.cpp プロジェクトは、LLM を高速にロードするため GGUF（GPT-Generated Unified Format、GPT によって生成された統一形式）を使用します。この形式にはセキュリティー上の懸念があるため、信頼できるソースからの GGUF ファイルを使用していることを確認してください。この記事では、Hugging Face* LMStudio（英語）リポジトリー（ここをクリックしてダウンロード）から、Llama* 3 70B モデルの INT4 量子化バージョンをダウンロードしました。ディスク上で 42.5GB あるため、モデルを実行すると、同程度の GPU メモリーが使用されます。

推論の実行

実行ファイルとモデルを用意できたら、モデルとプロンプトを使用して llama.cpp を呼び出し、すべてのモデルレイヤーを GPU にオフロードします。

```
build\bin\main.exe --model d:\models\Meta-Llama-3-70B-Instruct-Q4_K_M.gguf --prompt "Tell me
if or why AI is important to the future of humanity" --n-gpu-layers 999
```

Hugging Face*/PyTorch* ワークフローとは異なり、llama.cpp では事前に量子化されたバージョンのモデルをロードします。llama.cpp 出力には、いくつかのモデル・パラメーター、モデルのロード方法、デバイス上のモデルの特性が表示されます。モデルには 81 のレイヤーがあり、インテル® Core™ Ultra プロセッサー上の iGPU で実行されます。モデルは SYCL* ベースの計算パスで 81 のレイヤーを実行するため、40GB のメモリーを使用していることが分かります。最後に、Hugging Face*/PyTorch* 実装に似たプロンプトと出力が表示されます。

これが素晴らしい理由

いくつもの複雑なコンポーネントを組み込んだ、さまざまなアプリケーションやツールを構築してきた経験から、ローカルでできることが多ければ多いほど、開発プロセスを効率化できると実感しています。ワークフローを理解していれば、クラウドにモデルをデプロイするのは簡単です。

また、作業方法を理解していれば、エンタープライズ・インフラストラクチャーにもモデルを容易にデプロイできます。LLM をローカルで実行できることの素晴らしい点は、これらすべてを理解しなくても、すぐにテストできることです。いずれソリューションのデプロイが課題になりますが、ソリューションを設計している間は心配する必要がありません。

このソリューションのもう 1 つの利点は、小さなモデルであっても、クライアント・システムでは量子化を使用して実行できることが多いことです。これにより、モデルの精度が低下する可能性がありますが、メモリー容量を増やすことで、より高精度の量子化を使用してモデルをテストしたり、量子化されていないモデルを使用してモデルの機能を適切に判断できます。

LLM の設計、使用、統合に関連したソフトウェア開発は、増え続けるソフトウェア・ソリューションにとって優先事項です。これらのソリューションに対応する計算能力の要件とメモリー要件は、最高のコンシューマー・システムでさえも満たすことが難しく、できるだけローカルに保持したいという開発者の要望も考慮すると、開発者の生産性を最適化する上で大きな課題が生じます。

幸いなことに、96GB の DRAM で構成されたインテル® Arc™ 統合 GPU と小型フォームファクター PC を組み合わせることで、この課題を解決できます。このシステムでは、ハイエンドのコンシューマー向けディスクリート GPU でも実行できないモデルをローカルで実行できます。これらすべてが、約 1,200 米ドル（執筆時点）という非常にリーズナブルな価格で実現できます。何よりも、前述のソフトウェアは oneAPI 統合スタック上に構築されているため、ローカルで開発し、どこにでもデプロイできる利点があります。