llama.cpp を使用したインテル® GPU 上での LLM 実行

新しい SYCL* バックエンドを活用する

Zhang Jianyu、Meng Hengyu、Hu Ying、Luo Yu,、Duan Xiaoping、Majumder Abhilash
インテル コーポレーション

オープンソース・プロジェクトの llama.cpp（英語）は、人気が高まっている軽量の LLM フレームワークです。高いパフォーマンスとカスタマイズ性により、開発者、研究者、愛好家の活気に満ちたダイナミックなコミュニティーに成長しました。開始から約 1 年が経過した現在、GitHub* プロジェクトには 600 人以上の貢献者、52,000 個のスター、1,500 個のリリース、7,400 個のフォークがあります。最近のコードマージにより、サーバー向けおよびコンシューマー向けのインテル® GPU を含む、より多くのハードウェアがサポートされました。インテル® GPU のほかに、CPU（x86 および ARM）とその他のベンダーの GPU がサポートされています。

オリジナルの実装は Georgi Gerganov 氏によって作成されました。このプロジェクトは主に教育を目的としており、マシンラーニング用のテンソル・ライブラリーである ggml ライブラリー（英語）の新機能を開発するためのものです。

最近の更新により、インテルはより多くのデバイスで推論を可能にし、「あらゆるところに AI」をより多くのユーザーに提供しています。llama.cpp は C で記述されているため高速であり、ほかにも魅力的な機能を備えています。

- 16 ビット浮動小数点サポート
- 整数量子化サポート（4 ビット、5 ビット、8 ビットなど）
- サードパーティーの依存関係なし
- 実行時のメモリー割り当てなし

インテル® GPU 向けの SYCL* バックエンド

ggml には、さまざまなハードウェアをサポートして最適化するいくつかのバックエンドがあります。SYCL* バックエンドの開発には、さまざまなベンダーの GPU をサポートする oneAPI（英語）の SYCL*（ダイレクト・プログラミング言語）と oneMKL（高性能 BLAS ライブラリー）が利用されています。SYCL* は、ハードウェア・アクセラレーターの生産性を向上させるプログラミング・モデルです。これは、純粋な C++17 をベースとした、単一ソースの組込みのドメイン固有言語です。

SYCL* バックエンドは、すべてのインテル® GPU をサポートしており、次のデバイスで検証済みです。

- インテル® データセンター GPU マックス・シリーズおよびインテル® データセンター GPU フレックス・シリーズ
- インテル® Arc™ ディスクリート GPU
- インテル® Core™ Ultra プロセッサーに内蔵されたインテル® Arc™ GPU
- 第 11、12、13 世代インテル® Core™ プロセッサーに内蔵された iGPU

llama.cpp がインテル® GPU をサポートしたことで、何百万ものコンシューマー・デバイスが Llama* で推論を実行できるようになりました。OpenCL*（CLBlast）バックエンドと比較すると、SYCL* バックエンドはインテル® GPU でのパフォーマンスが大幅に向上しています。また、将来的には CPU や AI アクセラレーターを搭載したほかのプロセッサーなど、より多くのデバイスもサポートされる予定です。SYCL* バックエンドの使用方法については、「llama.cpp for SYCL」（英語）を参照してください。

SYCL* バックエンドを使用してインテル® GPU 上で LLM を実行する

詳細なガイドは、「llama.cpp for SYCL」（英語）を参照してください。SYCL* と oneAPI でサポートされているすべてのインテル® GPU で実行できます。サーバーおよびクラウドユーザーは、インテル® データセンター GPU マックス・シリーズおよびインテル® データセンター GPU フレックス・シリーズで実行できます。クライアント・ユーザーは、インテル® Arc™ GPU またはインテル® Core™ プロセッサーに内蔵されている iGPU で試すことができます。第 11 世代インテル® Core™ プロセッサー以降に内蔵されている iGPU をテストしました。古い iGPU でも動作しますが、パフォーマンスは低くなります。

唯一の制約はメモリーです。iGPU はホストの共有メモリーを使用し、dGPU は独自のメモリーを使用します。llama2–7b-Q4 モデルでは、80 以上の EU（第 11 世代インテル® Core™ プロセッサー以降）を搭載した iGPU を使用し、4.5GB を超える共有メモリーを有効にすることを推奨します（ホストメモリーの合計は 16GB 以上で、メモリーの半分を iGPU に割り当てることができます）。

インテル® GPU ドライバーのインストール

Linux* と Windows*（WSL 2）の両方がサポートされています。Linux* の場合、開発とテストに使用された Ubuntu* 22.04 を推奨します。

Linux*:

```
sudo usermod -aG render username
sudo usermod -aG video username
sudo apt install clinfo
sudo clinfo -l
```

出力例 :

```
Platform #0: Intel(R) OpenCL Graphics -- Device #0: Intel(R) Arc(TM) A770 Graphics
```

または

```
Platform #0: Intel(R) OpenCL HD Graphics -- Device #0: Intel(R) Iris(R) Xe Graphics \
[0x9a49\]
```

Windows* :「インテル® GPU ドライバーのインストール」を参照してください。

oneAPI ランタイムを有効にする

最初に、インテル® oneAPI ベース・ツールキットをインストールして、SYCL* コンパイラーと oneMKL を入手します。次に、oneAPI ランタイムを有効にします。

- Linux* : `source /opt/intel/oneapi/setvars.sh`
- Windows* : `"C:\Program Files (x86)\Intel\oneAPI\setvars.bat" intel64`

`sycl-ls` を実行して、レベルゼロデバイスが 1 つ以上存在することを確認します。`[ext_oneapi_level_zero:gpu:0]` のように、少なくとも 1 つの GPU が存在することを確認します。

ワンクリックでビルドします。

- Linux* : `./examples/sycl/build.sh`
- Windows* : `examples\sycl\win-build-sycl.bat`

上記のスクリプトには、oneAPI ランタイムを有効にするコマンドが含まれています。

ワンクリックでサンプルを実行する

llama-2–7b.Q4_0.gguf（英語）をダウンロードして、`models` フォルダーに保存します。

- Linux* : `./examples/sycl/run-llama2.sh`
- Windows* : `examples\sycl\win-run-llama2.bat`

上記のスクリプトには、oneAPI ランタイムを有効にするコマンドが含まれています。レベルゼロ GPU の ID が `0` でない場合は、スクリプト内のデバイス ID を変更してください。デバイス ID は以下のコマンドで確認できます。

- Linux* : `./build/bin/ls-sycl-device` **または** `./build/bin/main`
- Windows* : `build\bin\ls-sycl-device.exe` **または** `build\bin\main.exe`

まとめ

llama.cpp の SYCL* バックエンドは、LLM 開発者とユーザーがすべてのインテル® GPU を利用できるようにします。インテルのラップトップの場合は iGPU が、ゲーミング PC の場合はインテル® Arc™ GPU が、クラウド VM の場合はインテル® データセンター GPU マックス・シリーズまたはインテル® データセンター GPU フレックス・シリーズが搭載されているかを確認してください。そして搭載されている場合は、インテル® GPU で llama.cpp による LLM の魔法のような機能をお楽しみください。インテル® GPU でさらに多くの機能と最適化を利用できるようにするため、SYCL* バックエンドを試して貢献してください。llama.cpp プロジェクトから、クロスプラットフォーム開発向けの oneAPI プログラミング・モデルを学ぶことができます。