日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2023年5月16日火曜日

OpenAI WhisperのC++による実装を試してみる

以前にOpenAIのWhisperをOracle Cloudの無料枠で使用できるAmpere A1のコンピュート・インスタンスで動かす方法について、記事を書いています。

OpenAI Whisperを使った文字起こしアプリの作成(2) - UbuntuへのWhisperの実装
https://apexugj.blogspot.com/2023/01/openai-whisper-2-ubuntu.html

この記事でFP16を使わないようだ、と書いているのですが、OpenAIのWhisperのGitHubリポジトリにDiscussionとして聞いている方がいらっしゃいました。

https://github.com/openai/whisper/discussions/978

上記のディスカッションに、WhisperのC++の実装を試してみては？という回答があり、以下のリポジトリが紹介されています。

https://github.com/ggerganov/whisper.cpp

質問者が試したところ、すごく速くなったとのことなので、実際に試してみました。

上記リポジトリをZIP形式（whisper.cpp-master.zip）でダウンロードし、Ampere A1のUbuntuインスタンスにログインして展開したところから始めます。

WhisperのC++による実装は**$HOME/whisper.cpp-master**以下に展開しています。

直下にあるMakefileの内容を確認すると、DiscussionではCFLAGSの部分を以下に変えてコンパイルするように回答しています。

CFLAGS += -march=armv8.2-a+fp16

2023年5月16日現在のコードでは、この部分は以下の記述に変わっています。おそらくaarch64（Ampereのアーキテクチャ）では、上記の変更しなくてもFP16が使われるようになったと思われます。一応、-mcpu=natvieを-march=armv8.2-a+fp16に変更して速度差を確認しましたが、違いはありませんでした。

```
ifneq ($(filter aarch64%,$(UNAME_M)),)
        CFLAGS   += -mcpu=native
        CXXFLAGS += -mcpu=native
endif
```

そのまま、**make**を実行します。Whisperのバイナリは**main**として作成されます。

```
ubuntu@mywhisper2:~/whisper.cpp-master$ make
```

```
I whisper.cpp build info:
I UNAME_S:  Linux
I UNAME_P:  aarch64
I UNAME_M:  aarch64
I CFLAGS:   -I.               -O3 -DNDEBUG -std=c11   -fPIC -pthread -mcpu=native
I CXXFLAGS: -I. -I./examples -O3 -DNDEBUG -std=c++11 -fPIC -pthread -mcpu=native
I LDFLAGS:
I CC:       cc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
I CXX:      g++ (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0

cc  -I.              -O3 -DNDEBUG -std=c11   -fPIC -pthread -mcpu=native   -c
ggml.c -o ggml.o
g++ -I. -I./examples -O3 -DNDEBUG -std=c++11 -fPIC -pthread -mcpu=native -c
whisper.cpp -o whisper.o
g++ -I. -I./examples -O3 -DNDEBUG -std=c++11 -fPIC -pthread -mcpu=native
examples/main/main.cpp examples/common.cpp examples/common-ggml.cpp ggml.o
whisper.o -o main
./main -h

usage: ./main [options] file0.wav file1.wav ...

options:
  -h,        --help              [default] show this help message and exit
  -t N,      --threads N         [4      ] number of threads to use during
computation
  -p N,      --processors N      [1      ] number of processors to use during
computation
  -ot N,     --offset-t N        [0      ] time offset in milliseconds
  -on N,     --offset-n N        [0      ] segment index offset
  -d  N,     --duration N        [0      ] duration of audio to process in
milliseconds
  -mc N,     --max-context N     [-1     ] maximum number of text context tokens to
store
  -ml N,     --max-len N         [0      ] maximum segment length in characters
  -sow,      --split-on-word     [false  ] split on word rather than on token
  -bo N,     --best-of N         [2      ] number of best candidates to keep
  -bs N,     --beam-size N       [-1     ] beam size for beam search
  -wt N,     --word-thold N      [0.01   ] word timestamp probability threshold
  -et N,     --entropy-thold N   [2.40   ] entropy threshold for decoder fail
  -lpt N,    --logprob-thold N   [-1.00  ] log probability threshold for decoder
fail
  -su,       --speed-up          [false  ] speed up audio by x2 (reduced accuracy)
  -tr,       --translate         [false  ] translate from source language to
english
  -di,       --diarize           [false  ] stereo audio diarization
  -nf,       --no-fallback       [false  ] do not use temperature fallback while
decoding
  -otxt,     --output-txt        [false  ] output result in a text file
  -ovtt,     --output-vtt        [false  ] output result in a vtt file
  -osrt,     --output-srt        [false  ] output result in a srt file
  -olrc,     --output-lrc        [false  ] output result in a lrc file
  -owts,     --output-words      [false  ] output script for generating karaoke
video
  -fp,       --font-path         [/System/Library/Fonts/Supplemental/Courier New
Bold.ttf] path to a monospace font for karaoke video
  -ocsv,     --output-csv        [false  ] output result in a CSV file
  -oj,       --output-json       [false  ] output result in a JSON file
  -of FNAME, --output-file FNAME [       ] output file path (without file
extension)
  -ps,       --print-special     [false  ] print special tokens
  -pc,       --print-colors      [false  ] print colors
  -pp,       --print-progress    [false  ] print progress
  -nt,       --no-timestamps     [false  ] do not print timestamps
  -l LANG,   --language LANG     [en     ] spoken language ('auto' for auto-detect)
```

```
  -dl,       --detect-language   [false  ] exit after automatically detecting
language
             --prompt PROMPT     [       ] initial prompt
  -m FNAME,  --model FNAME       [models/ggml-base.en.bin] model path
  -f FNAME,  --file FNAME        [       ] input WAV file path

g++ -I. -I./examples -O3 -DNDEBUG -std=c++11 -fPIC -pthread -mcpu=native
examples/bench/bench.cpp ggml.o whisper.o -o bench
g++ -I. -I./examples -O3 -DNDEBUG -std=c++11 -fPIC -pthread -mcpu=native
examples/quantize/quantize.cpp examples/common.cpp examples/common-ggml.cpp ggml.o
whisper.o -o quantize
ubuntu@mywhisper2:~/whisper.cpp-master$
```

ディレクトリ**models**へ移動します。

WhisperのC++実装で使用するモデルをhuggingfaceからダウンロードします。実行時間の比較のためにlargeモデルを使います。

./download-ggml-model.sh large

```
ubuntu@mywhisper2:~/whisper.cpp-master/models$ ./download-ggml-model.sh large
Downloading ggml model large from 'https://huggingface.co/ggerganov/whisper.cpp'
...
ggml-large.bin                 100%
[================================================>]   2.88G  47.0MB/s    in 55s

Done! Model 'large' saved in 'models/ggml-large.bin'
You can now use it like this:

  $ ./main -m models/ggml-large.bin -f samples/jfk.wav

ubuntu@mywhisper2:~/whisper.cpp-master/models$
```

Whisper C++のサンプルとして提供されている**samples/jfk.wav**を使って、実行時間を比較してみます。

元々のWhisperの実装にパスを通して、Whisperを実行します。

export PATH=/home/ubuntu/.local/bin:$PATH
cd whisper.cpp-master/
time whisper samples/jfk.wav --language en --model large

realで**1m49.663s**という結果になっています。

```
ubuntu@mywhisper2:~$ export PATH=/home/ubuntu/.local/bin:$PATH
ubuntu@mywhisper2:~$ cd whisper.cpp-master/
ubuntu@mywhisper2:~/whisper.cpp-master$ time whisper samples/jfk.wav --language en
--model large
/usr/lib/python3/dist-packages/requests/__init__.py:89: RequestsDependencyWarning:
urllib3 (1.26.13) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
/home/ubuntu/.local/lib/python3.8/site-packages/whisper/transcribe.py:79:
UserWarning: FP16 is not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
[00:00.000 --> 00:11.000]  And so, my fellow Americans, ask not what your country
can do for you, ask what you can do for your country.

real    1m49.663s
user    4m6.096s
```

```
sys     1m20.822s
ubuntu@mywhisper2:~/whisper.cpp-master$
```

C++の実装を試してみます。

time ./main -m models/ggml-large.bin --language en samples/jfk.wav

realが**0m43.296s**で確かに速くなっています。認識されたスクリプトにも、ほぼ違いはありません。（And soの後ろのカンマの有無が異なる）

```
ubuntu@mywhisper2:~/whisper.cpp-master$ time ./main -m models/ggml-large.bin --
language en samples/jfk.wav
whisper_init_from_file_no_state: loading model from 'models/ggml-large.bin'
whisper_model_load: loading model
whisper_model_load: n_vocab       = 51865
whisper_model_load: n_audio_ctx   = 1500
whisper_model_load: n_audio_state = 1280
whisper_model_load: n_audio_head  = 20
whisper_model_load: n_audio_layer = 32
whisper_model_load: n_text_ctx    = 448
whisper_model_load: n_text_state  = 1280
whisper_model_load: n_text_head   = 20
whisper_model_load: n_text_layer  = 32
whisper_model_load: n_mels        = 80
whisper_model_load: ftype         = 1
whisper_model_load: qntvr         = 0
whisper_model_load: type          = 5
whisper_model_load: mem required  = 3557.00 MB (+   71.00 MB per decoder)
whisper_model_load: adding 1608 extra tokens
whisper_model_load: model ctx     = 2951.27 MB
whisper_model_load: model size    = 2950.66 MB
whisper_init_state: kv self size  =   70.00 MB
whisper_init_state: kv cross size =  234.38 MB

system_info: n_threads = 4 / 4 | AVX = 0 | AVX2 = 0 | AVX512 = 0 | FMA = 0 | NEON =
1 | ARM_FMA = 1 | F16C = 0 | FP16_VA = 1 | WASM_SIMD = 0 | BLAS = 0 | SSE3 = 0 |
VSX = 0 | COREML = 0 |

main: processing 'samples/jfk.wav' (176000 samples, 11.0 sec), 4 threads, 1
processors, lang = en, task = transcribe, timestamps = 1 ...


[00:00:00.000 --> 00:00:11.000]   And so my fellow Americans, ask not what your
country can do for you, ask what you can do for your country.


whisper_print_timings:     load time =  1225.63 ms
whisper_print_timings:     fallbacks =   0 p /   0 h
whisper_print_timings:      mel time =   213.42 ms
whisper_print_timings:   sample time =    23.18 ms /    27 runs (    0.86 ms per
run)
whisper_print_timings:   encode time = 39792.95 ms /     1 runs (39792.95 ms per
run)
whisper_print_timings:   decode time =  1646.82 ms /    27 runs (   60.99 ms per
run)
whisper_print_timings:    total time = 43095.16 ms

real    0m43.296s
user    2m46.227s
sys     0m1.460s
ubuntu@mywhisper2:~/whisper.cpp-master$
```

以前のテストで使用した日本語のtest.m4aファイルを使って、速度を比較してみます。

Whisper C++が処理できる音声ファイルは、サンプリング・レートが16KのWAVファイルのみとのことなので、test.m4aを変換します。

ffmpeg -loglevel -0 -y -i /home/ubuntu/test/test.m4a -ar 16000 -ac 1 -c:a pcm_s16le samples/test.wav

```
ubuntu@mywhisper2:~/whisper.cpp-master$ ffmpeg -loglevel -0 -y -i
/home/ubuntu/test/test.m4a -ar 16000 -ac 1 -c:a pcm_s16le samples/test.wav
ubuntu@mywhisper2:~/whisper.cpp-master$
```

作成したtest.wavを使って、同様に処理時間を比較します。

time whisper samples/test.wav --language ja --model large

WAVファイルに変換していますが、以前の記事と同様の処理時間です。

```
ubuntu@mywhisper2:~/whisper.cpp-master$ time whisper samples/test.wav --language ja
--model large
/usr/lib/python3/dist-packages/requests/__init__.py:89: RequestsDependencyWarning:
urllib3 (1.26.13) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
/home/ubuntu/.local/lib/python3.8/site-packages/whisper/transcribe.py:79:
UserWarning: FP16 is not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
[00:00.000 --> 00:09.000] こんにちは 初めてウィスパーを インストールしてみました これで試して
みます

real    1m44.410s
user    3m56.191s
sys     1m14.775s
ubuntu@mywhisper2:~/whisper.cpp-master$
```

Whisper C++で試してみます。

time ./main -m models/ggml-large.bin --language ja samples/test.wav

なぜか、無音声部分（9:00から11:00の間）に、おやすみなさい、と文字が起こされています。そういう言葉は話していません。また、文字起こしされた表記も微妙に違います。

速度もそれほど上がっていません。

```
ubuntu@mywhisper2:~/whisper.cpp-master$ time ./main -m models/ggml-large.bin --
language ja samples/test.wav
whisper_init_from_file_no_state: loading model from 'models/ggml-large.bin'
whisper_model_load: loading model
whisper_model_load: n_vocab       = 51865
whisper_model_load: n_audio_ctx   = 1500
whisper_model_load: n_audio_state = 1280
whisper_model_load: n_audio_head  = 20
whisper_model_load: n_audio_layer = 32
whisper_model_load: n_text_ctx    = 448
whisper_model_load: n_text_state  = 1280
whisper_model_load: n_text_head   = 20
whisper_model_load: n_text_layer  = 32
whisper_model_load: n_mels        = 80
```

```
whisper_model_load: ftype          = 1
whisper_model_load: qntvr          = 0
whisper_model_load: type           = 5
whisper_model_load: mem required  = 3557.00 MB (+   71.00 MB per decoder)
whisper_model_load: adding 1608 extra tokens
whisper_model_load: model ctx     = 2951.27 MB
whisper_model_load: model size    = 2950.66 MB
whisper_init_state: kv self size  =   70.00 MB
whisper_init_state: kv cross size =  234.38 MB

system_info: n_threads = 4 / 4 | AVX = 0 | AVX2 = 0 | AVX512 = 0 | FMA = 0 | NEON =
1 | ARM_FMA = 1 | F16C = 0 | FP16_VA = 1 | WASM_SIMD = 0 | BLAS = 0 | SSE3 = 0 |
VSX = 0 | COREML = 0 |

main: processing 'samples/test.wav' (175424 samples, 11.0 sec), 4 threads, 1
processors, lang = ja, task = transcribe, timestamps = 1 ...


[00:00:00.000 --> 00:00:09.000]  こんにちは 初めてWisperをインストール してみました これで
試してみます
[00:00:09.000 --> 00:00:11.000]  おやすみなさい


whisper_print_timings:     load time =  1232.04 ms
whisper_print_timings:     fallbacks =   0 p /   0 h
whisper_print_timings:      mel time =   211.98 ms
whisper_print_timings:   sample time =    81.84 ms /    74 runs (    1.11 ms per
run)
whisper_print_timings:   encode time = 78187.03 ms /     2 runs (39093.52 ms per
run)
whisper_print_timings:   decode time =  4492.09 ms /    72 runs (   62.39 ms per
run)
whisper_print_timings:    total time = 84394.84 ms

real    1m24.596s
user    5m30.696s
sys     0m1.495s
ubuntu@mywhisper2:~/whisper.cpp-master$
```

誤認識（？）された無声部を削除したファイルをtest9.wavとして作成し、再度確認してみました。

本家のWhisperでの結果はほぼ同じです。

```
ubuntu@mywhisper2:~/whisper.cpp-master$ time whisper samples/test9.wav --language
ja --model large
/usr/lib/python3/dist-packages/requests/__init__.py:89: RequestsDependencyWarning:
urllib3 (1.26.13) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
/home/ubuntu/.local/lib/python3.8/site-packages/whisper/transcribe.py:79:
UserWarning: FP16 is not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
[00:00.000 --> 00:08.680] 今日は初めてウィスパーをインストール してみました これで試してみます

real    1m43.988s
user    3m54.747s
sys     1m15.679s
ubuntu@mywhisper2:~/whisper.cpp-master$
```

Whisper C++で試してみます。無声部を削除すると、処理速度は短くなっています。

```
ubuntu@mywhisper2:~/whisper.cpp-master$ time ./main -m models/ggml-large.bin --
language ja samples/test9.wav
whisper_init_from_file_no_state: loading model from 'models/ggml-large.bin'
whisper_model_load: loading model
whisper_model_load: n_vocab       = 51865
whisper_model_load: n_audio_ctx   = 1500
whisper_model_load: n_audio_state = 1280
whisper_model_load: n_audio_head  = 20
whisper_model_load: n_audio_layer = 32
whisper_model_load: n_text_ctx    = 448
whisper_model_load: n_text_state  = 1280
whisper_model_load: n_text_head   = 20
whisper_model_load: n_text_layer  = 32
whisper_model_load: n_mels        = 80
whisper_model_load: ftype         = 1
whisper_model_load: qntvr         = 0
whisper_model_load: type          = 5
whisper_model_load: mem required  = 3557.00 MB (+   71.00 MB per decoder)
whisper_model_load: adding 1608 extra tokens
whisper_model_load: model ctx     = 2951.27 MB
whisper_model_load: model size    = 2950.66 MB
whisper_init_state: kv self size  =   70.00 MB
whisper_init_state: kv cross size =  234.38 MB

system_info: n_threads = 4 / 4 | AVX = 0 | AVX2 = 0 | AVX512 = 0 | FMA = 0 | NEON =
1 | ARM_FMA = 1 | F16C = 0 | FP16_VA = 1 | WASM_SIMD = 0 | BLAS = 0 | SSE3 = 0 |
VSX = 0 | COREML = 0 |

main: processing 'samples/test9.wav' (140608 samples, 8.8 sec), 4 threads, 1
processors, lang = ja, task = transcribe, timestamps = 1 ...


[00:00:00.000 --> 00:00:08.720]  こんにちは 初めてWisperをインストール してみました これで
試してみます


whisper_print_timings:     load time =  1228.44 ms
whisper_print_timings:     fallbacks =   0 p /   0 h
whisper_print_timings:      mel time =   211.81 ms
whisper_print_timings:   sample time =    20.79 ms /    24 runs (    0.87 ms per
run)
whisper_print_timings:   encode time = 37966.31 ms /     1 runs (37966.31 ms per
run)
whisper_print_timings:   decode time =  1475.80 ms /    24 runs (   61.49 ms per
run)
whisper_print_timings:    total time = 41092.65 ms

real    0m41.295s
user    2m38.073s
sys     0m1.511s
ubuntu@mywhisper2:~/whisper.cpp-master$
```

WhisperのC++の実装では、処理時間はかなり短くなりますが、Pythonの実装とまったく同じ結果が得られる、というものではなさそうです。

完

Yuji N. 時刻: 14:46

共有

ウェブ バージョンを表示

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示