CS6886W - System Engineering for Deep Learning – Assignment2

Performance Benchmarking, Roofline Modeling, and Scaling Study

Sanchari Roy[CS24M532]

GitHub repository : https://github.com/cs24m532/Deep-Learning

Overview:

This report presents a comprehensive performance analysis of the **GPT-2 Medium** language model using the open-source **llama.cpp** inference framework. The study evaluates the model under multiple execution configurations, beginning with naive single-threaded CPU runs and progressing toward highly optimized builds that incorporate SIMD acceleration and Intel® MKL-based matrix multiplication. Each configuration is benchmarked to assess its computational efficiency, scalability, and hardware utilization.

The primary objective of this work is to understand how **software optimizations**, **compiler flags**, and **hardware parallelism** affect the inference performance of large language models. By leveraging the **Roofline performance model**, the study quantifies key factors such as **operational intensity (OI)**, **memory bandwidth usage**, and **achieved FLOP/s**, enabling a deeper examination of the system's memory-bound versus compute-bound behavior.

Experimental results reveal clear trade-offs between execution regimes: naive builds suffer from significant memory stalls, while optimized and multi-threaded builds demonstrate notable speedups—though ultimately constrained by the system's memory bandwidth ceiling. Through this systematic evaluation, the report provides valuable insights into **hardware-aware optimization**, **scaling behavior**, and **performance tuning techniques** essential for efficient deep learning inference workloads on modern CPU architectures.
Since I am not able to find any single machine where I can get all the counters I have to switch between windows and Linux machines

Task 1 – Installing llama.cpp from Source

Steps to Execute:

git clone https://github.com/ggml-org/llama.cpp

--------Clones the llama.cpp repository.

cd llama.cpp

--------Moves into the cloned project directory

cmake -B build

-------- CMake configures the build system and generates necessary makefiles.

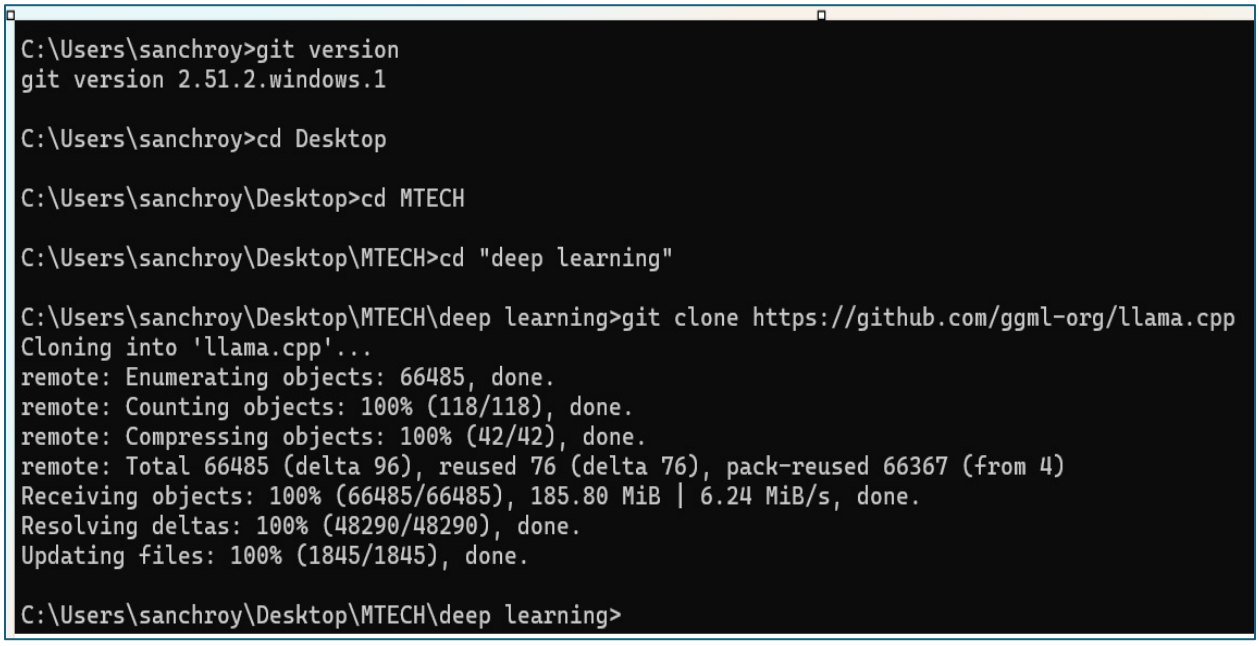
cmake --build build --config Release -j 8

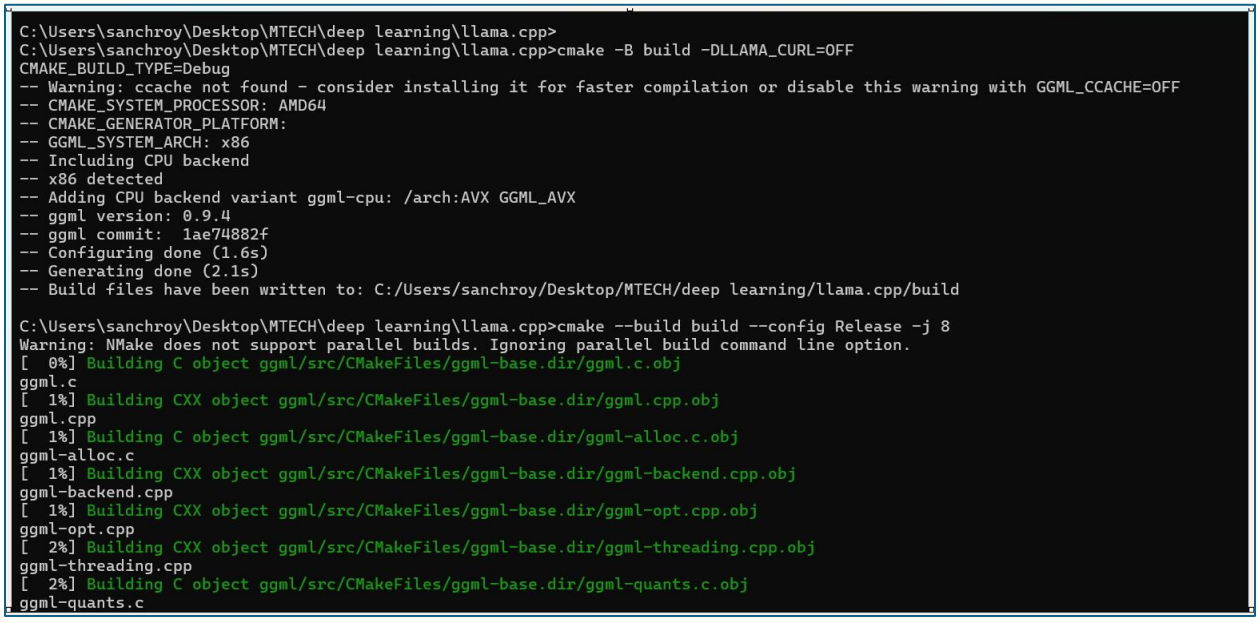
----------The build uses 8 parallel jobs for faster compilation.Verification:

A successful build produces the binary build/bin/llama-bench.

Ran ./build/bin/llama-bench --help confirms installation.

Snapshots:


C:\Users\sanchroy>git version
git version 2.51.2.windows.1
C:\Users\sanchroy>cd Desktop
C:\Users\sanchroy\Desktop>cd MTECH
C:\Users\sanchroy\Desktop\MTECH>cd "deep learning"
C:\Users\sanchroy\Desktop\MTECH\deep learning>git clone https://github.com/ggml-org/llama.cpp
Cloning into 'llama.cpp'...
remote: Enumerating objects: 66485, done.
remote: Counting objects: 100% (118/118), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 66485 (delta 96), reused 76 (delta 76), pack-reused 66367 (from 4)
Receiving objects: 100% (66485/66485), 185.80 MiB | 6.24 MiB/s, done.
Resolving deltas: 100% (48290/48290), done.
Updating files: 100% (1845/1845), done.
C:\Users\sanchroy\Desktop\MTECH\deep learning>


C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>cmake -B build -DLLAMA_CURL=OFF
CMAKE_BUILD_TYPE=Debug
-- Warning: ccache not found - consider installing it for faster compilation or disable this warning with GGML_CCACHE=OFF
-- CMAKE_SYSTEM_PROCESSOR: AMD64
-- CMAKE_GENERATOR_PLATFORM:
-- GGML_SYSTEM_ARCH: x86
-- Including CPU backend
-- x86 detected
-- Adding CPU backend variant ggml-cpu: /arch:AVX GGML_AVX
-- ggml version: 0.9.4
-- ggml commit: 1ae74882f
-- Configuring done (1.6s)
-- Generating done (2.1s)
-- Build files have been written to: C:/Users/sanchroy/Desktop/MTECH/deep learning/llama.cpp/build
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>cmake --build build --config Release -j 8
Warning: NMake does not support parallel builds. Ignoring parallel build command line option.
[0%] Building C object ggml/src/CMakeFiles/ggml-base.dir/ggml.c.obj
ggml.c
[1%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml.cpp.obj
ggml.cpp
[1%] Building C object ggml/src/CMakeFiles/ggml-base.dir/ggml-alloc.c.obj
ggml-alloc.c
[1%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml-backend.cpp.obj
ggml-backend.cpp
[1%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml-opt.cpp.obj
ggml-opt.cpp
[2%] Building CXX object ggml/src/CMakeFiles/ggml-base.dir/ggml-threading.cpp.obj
ggml-threading.cpp
[2%] Building C object ggml/src/CMakeFiles/ggml-base.dir/ggml-quants.c.obj
ggml-quants.c

```
cvector-generator.cpp
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\common\common.h(259): warning C4305: 'initializing': truncation from 'double' to 'float'
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\common\common.h(269): warning C4244: 'argument': conversion from 'const unsigned int' to 'float', p
ssible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\pca.hpp(38): warning C4305: 'initializing': truncation from 'double' to 'fl
at'
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\pca.hpp(109): warning C4244: 'argument': conversion from 'int64_t' to 'cons
 unsigned int', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\cvector-generator.cpp(101): warning C4244: 'argument': conversion from 'flo
t' to 'const unsigned int', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\cvector-generator.cpp(102): warning C4244: 'argument': conversion from 'flo
t' to 'const unsigned int', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\cvector-generator.cpp(103): warning C4244: 'argument': conversion from 'flo
t' to 'const unsigned int', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\cvector-generator.cpp(103): warning C4244: 'initializing': conversion from
int64_t' to 'size_t', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\cvector-generator.cpp(108): warning C4244: 'argument': conversion from 'flo
t' to 'const unsigned int', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\cvector-generator.cpp(119): warning C4244: 'initializing': conversion from
int64_t' to 'int', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\cvector-generator.cpp(129): warning C4305: 'argument': truncation from 'dou
le' to 'float'
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\cvector-generator\cvector-generator.cpp(137): warning C4244: 'initializing': conversion from
int64_t' to 'int', possible loss of data
[ 99%] Linking CXX executable ..\..\bin\llama-cvector-generator.exe
[ 99%] Built target llama-cvector-generator
[ 99%] Building CXX object tools/export-lora/CMakeFiles/llama-export-lora.dir/export-lora.cpp.obj
export-lora.cpp
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\common\common.h(259): warning C4305: 'initializing': truncation from 'double' to 'float'
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\common\common.h(269): warning C4244: 'argument': conversion from 'const unsigned int' to 'float', p
ssible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\export-lora\export-lora.cpp(22): warning C4244: 'initializing': conversion from 'int64_t' to
int', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\export-lora\export-lora.cpp(27): warning C4244: 'initializing': conversion from 'int64_t' to
int', possible loss of data
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\tools\export-lora\export-lora.cpp(319): warning C4244: 'argument': conversion from 'int64_t' to 'co
st unsigned int', possible loss of data
[100%] Linking CXX executable ..\..\bin\llama-export-lora.exe
[100%] Built target llama-export-lora

C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>
```

```
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\build\bin>.\llama-bench.exe --help
warning: asserts enabled, performance may be affected
warning: debug build, performance may be affected
register_backend: registered backend CPU (1 devices)
register_device: registered device CPU (13th Gen Intel(R) Core(TM) i7-13800H)
load_backend: failed to find ggml_backend_init in C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp\build\bin\ggml-cpu.dll
usage: .\llama-bench.exe [options]

options:
  -h, --help
  --numa <distribute|isolate|numactl>       numa mode (default: disabled)
  -r, --repetitions <n>                     number of times to repeat each test (default: 5)
  --prio <-1|0|1|2|3>                          process/thread priority (default: 0)
  --delay <0...N> (seconds)                 delay between each test (default: 0)
  -o, --output <csv|json|jsonl|md|sql>      output format printed to stdout (default: md)
  -oe, --output-err <csv|json|jsonl|md|sql> output format printed to stderr (default: none)
  --list-devices                            list available devices and exit
  -v, --verbose                             verbose output
  --progress                                print test progress indicators
  --no-warmup                               skip warmup runs before benchmarking

test parameters:
  -m, --model <filename>                    (default: models/7B/ggml-model-q4_0.gguf)
  -p, --n-prompt <n>                        (default: 512)
  -n, --n-gen <n>                           (default: 128)
  -pg <pp,tg>                               (default: )
  -d, --n-depth <n>                         (default: 0)
  -b, --batch-size <n>                      (default: 2048)
  -ub, --ubatch-size <n>                    (default: 512)
  -ctk, --cache-type-k <t>                  (default: f16)
  -ctv, --cache-type-v <t>                  (default: f16)
  -t, --threads <n>                         (default: 14)
  -C, --cpu-mask <hex,hex>                  (default: 0x0)
  --cpu-strict <0|1>                        (default: 0)
  --poll <0...100>                          (default: 50)
  -ngl, --n-gpu-layers <n>                  (default: 99)
  -ncmoe, --n-cpu-moe <n>                   (default: 0)
  -sm, --split-mode <none|layer|row>        (default: layer)
  -mg, --main-gpu <i>                       (default: 0)
  -nkvo, --no-kv-offload <0|1>              (default: 0)
```

Task 2 – Setting up GPT-2 Medium Model

Steps to Execute:

1. **Download Model:** Cloning the GPT-2 Medium model repository directly from Hugging Face**.**

   git clone https://huggingface.co/openai-community/gpt2-medium

2. Need to install other dependencies torch and transformer to make it work

3. **Convert to GGUF:** llama.cpp cannot load HuggingFace models directly.
   It requires a special optimized format called GGUF (GGML Unified Format).

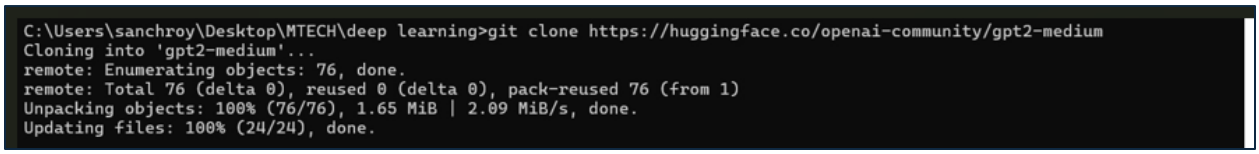
   python3 convert_hf_to_gguf.py gpt2-medium --outfile gpt2-medium.gguf

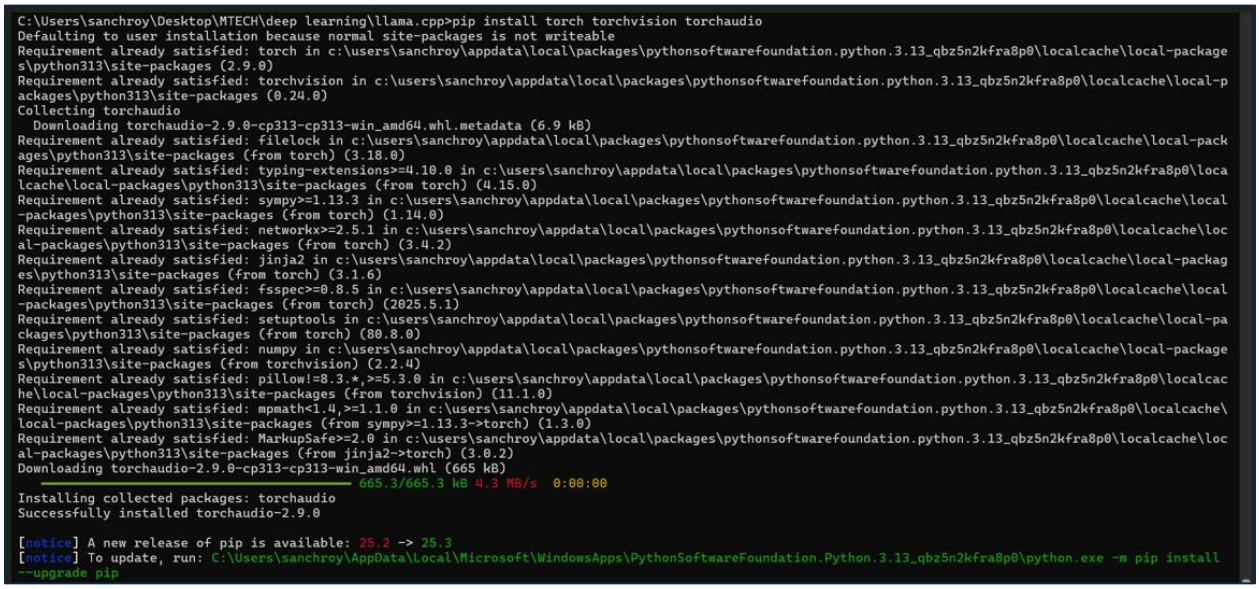
4. **Run Sanity Benchmark:** A sanity benchmark on the converted GPT-2 Medium model using the llama-bench

   -m ..\..\..\gpt2-medium.gguf : specifies the path to the converted model file.
   -p 0 : sets the prompt index to 0 .
   -n 256 : runs the benchmark for 256 tokens to test model performance.

   ./llama-bench -m gpt2-medium.gguf -p 0 -n 256

Snapshots





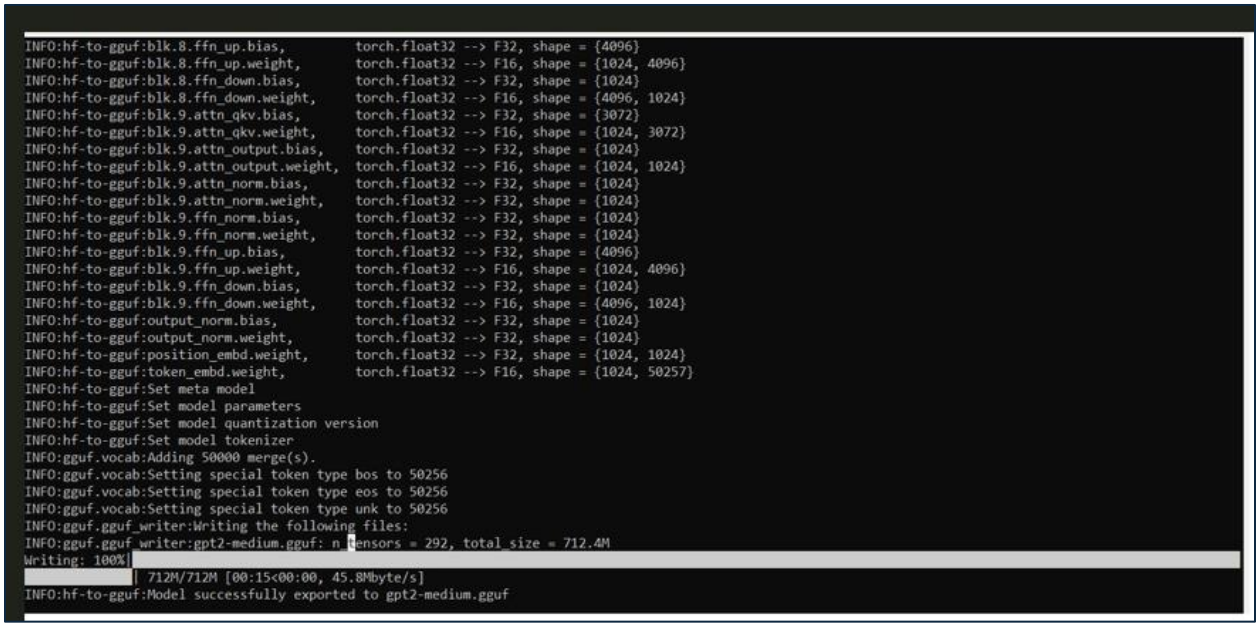
```
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>pip install transformers huggingface_hub tqdm sentencepiece
Defaulting to user installation because normal site-packages is not writeable
Collecting transformers
  Downloading transformers-4.57.1-py3-none-any.whl.metadata (43 kB)
Collecting huggingface_hub
  Downloading huggingface_hub-1.0.1-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: tqdm in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages
\python313\site-packages (4.67.1)
Collecting sentencepiece
  Downloading sentencepiece-0.2.1-cp313-cp313-win_amd64.whl.metadata (10 kB)
Requirement already satisfied: filelock in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pack
ages\python313\site-packages (from transformers) (3.18.0)
Collecting huggingface_hub
  Downloading huggingface_hub-0.36.0-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: numpy>=1.17 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-p
ackages\python313\site-packages (from transformers) (2.2.4)
Requirement already satisfied: packaging>=20.0 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\loc
al-packages\python313\site-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-p
ackages\python313\site-packages (from transformers) (6.0.3)
Collecting regex!=2019.12.17 (from transformers)
  Downloading regex-2025.10.23-cp313-cp313-win_amd64.whl.metadata (41 kB)
Requirement already satisfied: requests in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pack
ages\python313\site-packages (from transformers) (2.32.5)
Collecting tokenizers<=0.23.0,>=0.22.0 (from transformers)
  Downloading tokenizers-0.22.1-cp39-abi3-win_amd64.whl.metadata (6.9 kB)
Collecting safetensors>=0.4.3 (from transformers)
  Downloading safetensors-0.6.2-cp38-abi3-win_amd64.whl.metadata (4.1 kB)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\lo
cal-packages\python313\site-packages (from huggingface_hub) (2025.5.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\loc
alcache\local-packages\python313\site-packages (from huggingface_hub) (4.15.0)
Requirement already satisfied: colorama in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-pack
ages\python313\site-packages (from tqdm) (0.4.6)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\local
cache\local-packages\python313\site-packages (from requests->transformers) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-
```

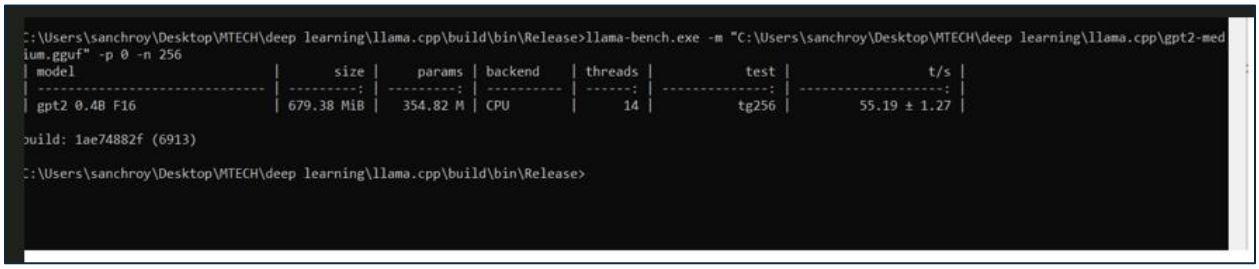
```
  Downloading safetensors-0.6.2-cp38-abi3-win_amd64.whl.metadata (4.1 kB)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from huggingface_hub) (2025.5.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from huggingface_hub) (4.15.0)
Requirement already satisfied: colorama in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from tqdm) (0.4.6)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from requests->transformers) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from requests->transformers) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from requests->transformers) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\sanchroy\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from requests->transformers) (2025.10.5)
Downloading transformers-4.57.1-py3-none-any.whl (12.0 MB)
   ---------------------------------------- 12.0/12.0 MB 5.0 MB/s  0:00:02
Downloading huggingface_hub-0.36.0-py3-none-any.whl (566 kB)
   ---------------------------------------- 566.1/566.1 kB 4.4 MB/s  0:00:00
Downloading tokenizers-0.22.1-cp39-abi3-win_amd64.whl (2.7 MB)
   ---------------------------------------- 2.7/2.7 MB 5.2 MB/s  0:00:00
Downloading sentencepiece-0.2.1-cp313-cp313-win_amd64.whl (1.1 MB)
   ---------------------------------------- 1.1/1.1 MB 4.6 MB/s  0:00:00
Downloading regex-2025.10.23-cp313-cp313-win_amd64.whl (276 kB)
Downloading safetensors-0.6.2-cp38-abi3-win_amd64.whl (320 kB)
Installing collected packages: sentencepiece, safetensors, regex, huggingface_hub, tokenizers, transformers
   ----------------------------------------  3/6 [huggingface_hub]  WARNING: The scripts hf.exe, huggingface-cli.exe and tiny-agents.exe are installed in 'C:\Users\sanchroy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
   ----------------------------------------  5/6 [transformers]  WARNING: The scripts transformers-cli.exe and transformers.exe are installed in 'C:\Users\sanchroy\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\Python313\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed huggingface_hub-0.36.0 regex-2025.10.23 safetensors-0.6.2 sentencepiece-0.2.1 tokenizers-0.22.1 transformers-4.57.1

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: C:\Users\sanchroy\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip

C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>
```

```
C:\Users\sanchroy\Desktop\MTECH\deep learning\llama.cpp>python3 convert_hf_to_gguf.py "C:\Users\sanchroy\Desktop\MTECH\deep learning\gpt2-medium" --outfile gpt2-medium.gguf
INFO:hf-to-gguf:Loading model: gpt2-medium
INFO:hf-to-gguf:Model architecture: GPT2LMHeadModel
INFO:hf-to-gguf:gguf: indexing model part 'model.safetensors'
INFO:gguf.gguf_writer:gguf: This GGUF file is for Little Endian only
INFO:hf-to-gguf:Exporting model...
INFO:hf-to-gguf:blk.0.attn_qkv.bias,       torch.float32 --> F32, shape = {3072}
INFO:hf-to-gguf:blk.0.attn_qkv.weight,     torch.float32 --> F16, shape = {1024, 3072}
INFO:hf-to-gguf:blk.0.attn_output.bias,    torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.0.attn_output.weight,  torch.float32 --> F16, shape = {1024, 1024}
INFO:hf-to-gguf:blk.0.attn_norm.bias,      torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.0.attn_norm.weight,    torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.0.ffn_norm.bias,       torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.0.ffn_norm.weight,     torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.0.ffn_up.bias,         torch.float32 --> F32, shape = {4096}
INFO:hf-to-gguf:blk.0.ffn_up.weight,       torch.float32 --> F16, shape = {1024, 4096}
INFO:hf-to-gguf:blk.0.ffn_down.bias,       torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.0.ffn_down.weight,     torch.float32 --> F16, shape = {4096, 1024}
INFO:hf-to-gguf:blk.1.attn_qkv.bias,       torch.float32 --> F32, shape = {3072}
INFO:hf-to-gguf:blk.1.attn_qkv.weight,     torch.float32 --> F16, shape = {1024, 3072}
INFO:hf-to-gguf:blk.1.attn_output.bias,    torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.1.attn_output.weight,  torch.float32 --> F16, shape = {1024, 1024}
INFO:hf-to-gguf:blk.1.attn_norm.bias,      torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.1.attn_norm.weight,    torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.1.ffn_norm.bias,       torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.1.ffn_norm.weight,     torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.1.ffn_up.bias,         torch.float32 --> F32, shape = {4096}
INFO:hf-to-gguf:blk.1.ffn_up.weight,       torch.float32 --> F16, shape = {1024, 4096}
INFO:hf-to-gguf:blk.1.ffn_down.bias,       torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.1.ffn_down.weight,     torch.float32 --> F16, shape = {4096, 1024}
INFO:hf-to-gguf:blk.10.attn_qkv.bias,      torch.float32 --> F32, shape = {3072}
INFO:hf-to-gguf:blk.10.attn_qkv.weight,    torch.float32 --> F16, shape = {1024, 3072}
INFO:hf-to-gguf:blk.10.attn_output.bias,   torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.10.attn_output.weight, torch.float32 --> F16, shape = {1024, 1024}
INFO:hf-to-gguf:blk.10.attn_norm.bias,     torch.float32 --> F32, shape = {1024}
INFO:hf-to-gguf:blk.10.attn_norm.weight,   torch.float32 --> F32, shape = {1024}
```





Explanation:

Field	Value	Explanation
Model	gpt2 0.4B F16	GPT-2 Medium (354M parameters) loaded in FP16 GGUF format. “0.4B” refers to ~0.354B parameters.
Model Size	679.38 MiB	Size of the FP16 GGUF file. FP16 doubles the memory compared to quantized models but gives higher accuracy.
Parameters	354.82 M	Number of trainable weights in GPT-2 Medium. llama.cpp reads this directly from GGUF metadata.
Backend	CPU	The benchmark used pure CPU execution (no GPU acceleration).
Threads	14	llama.cpp used 14 CPU threads for parallel inference. This is close to the physical + logical CPU cores on my system.
Test	tg256	Built-in benchmark “token generation 256” which measures generation speed for 256 tokens.

Field	Value	Explanation
Throughput (t/s)	**55.19 ± 1.27**	The most important metric. CPU generates ~55 tokens/sec with 14 threads—strong performance for GPT-2 Medium in FP16 mode.

Task 3 – Naive Execution (No Parallelism)

A. Re-build llama.cpp by disabling all acceleration flags.

B. Run Benchmark: After compilation, run the benchmark in single-thread *mode:*

Deliverable:

Submit benchmark results from the single-thread run.

Solution:

A. Rebuild llama.cpp with all acceleration flags disabled

Steps to Execute:

cmake -B build -DGGML_CPU_GENERIC=ON -DGGML_NATIVE=OFF -DGGML_AVX=OFF -DGGML_AVX2=OFF -DGGML_AVX512=OFF -DGGML_SSE42=OFF -DGGML_F16C=OFF -DGGML_FMA=OFF

-B build: Generates build files in the *build* directory.

-DGGML_CPU_GENERIC=ON: Enables a generic CPU build for all processors.

-DGGML_NATIVE=OFF: Disables host-specific optimizations.

-DGGML_AVX/AVX2/AVX512=OFF: Turns off AVX instruction sets for parallel processing.

-DGGML_SSE42=OFF: Disables SSE4.2 optimizations.

*-DGGML_F16C=OFF***:** Disables half-precision (16-bit) float support.

-DGGML_FMA=OFF: Disables Fused Multiply-Add arithmetic optimization.

Details:

This configuration forces the program to compile without any hardware acceleration, ensuring that it runs purely on basic CPU instructions. It is useful for benchmarking and testing the performance of the program under non-optimized conditions.

Run the benchmark in single-thread mode after compilation.

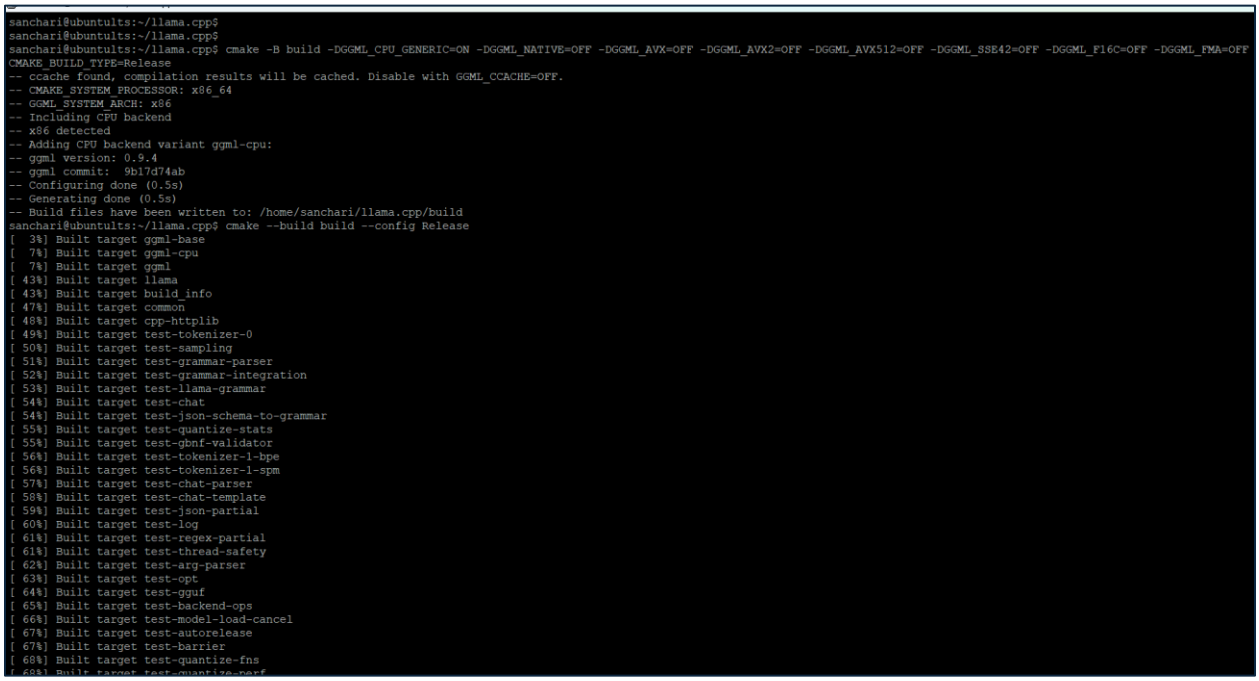
-m ..\..\..\gpt2-medium.gguf : specifies the path to the converted model file.

-p 0 : sets the prompt index to 0 .

-n 256 : runs the benchmark for 256 tokens for model performance.

-t 1: Runs the benchmark on a single CPU thread

Snapshots



```
[ 84%] Built target llama-speculative-simple
[ 84%] Built target llama-gen-docs
[ 85%] Built target llama-finetune
[ 86%] Built target llama-diffusion-cli
[ 87%] Built target llama-logits
[ 88%] Built target llama-convert-llama2c-to-ggml
[ 88%] Built target llama-vdot
[ 89%] Built target llama-q8dot
[ 90%] Built target llama-batched-bench
[ 91%] Built target llama-gguf-split
[ 91%] Built target llama-imatrix
[ 91%] Built target llama-bench
[ 92%] Built target llama-cli
[ 92%] Built target llama-perplexity
[ 92%] Built target llama-quantize
[ 93%] Built target llama-server
[ 93%] Built target llama-run
[ 94%] Built target llama-tokenize
[ 95%] Built target llama-tts
[ 96%] Built target llama-llava-cli
[ 97%] Built target llama-gemma3-cli
[ 98%] Built target llama-minicpmv-cli
[ 99%] Built target llama-qwen2vl-cli
[100%] Built target llama-mtmd-cli
[100%] Built target llama-cvector-generator
[100%] Built target llama-export-lora
sanchari@ubuntults:~/llama.cpp$ ./build/bin/llama-bench -m ./gpt2-medium.gguf -p 0 -n 256 -t 1
./build/bin/llama-bench: error while loading shared libraries: libsvml.so: cannot open shared object file: No such file or directory
sanchari@ubuntults:~/llama.cpp$ source /opt/intel/oneapi/setvars.sh

:: initializing oneAPI environment ...
   -bash: BASH_VERSION = 5.2.21(1)-release
   args: Using "$@" for setvars.sh arguments:
:: compiler -- latest
:: debugger -- latest
:: dev-utilities -- latest
:: dpl -- latest
:: mkl -- latest
:: tbb -- latest
:: umf -- latest
:: oneAPI environment initialized ::

sanchari@ubuntults:~/llama.cpp$ ./build/bin/llama-bench -m ./gpt2-medium.gguf -p 0 -n 256 -t 1
| model                          |       size |     params | backend    | threads |           test |                  t/s |
| ------------------------------ | ---------: | ---------: | ---------- | ------: | -------------: | -------------------: |
| gpt2 0.4B F16                  | 679.38 MiB |   354.82 M | CPU        |       1 |          tg256 |          1.40 ± 0.01 |

build: 9b17d74ab (7062)
sanchari@ubuntults:~/llama.cpp$
```

Explanation

Field	Value	Explanation
Model	gpt2 0.4B F16	GPT-2 Medium in FP16 GGUF format (≈354M parameters).
Model Size	679.38 MiB	FP16 GGUF file loaded by llama.cpp.
Parameters	354.82 M	Total number of FP16 weights in the model.
Backend	CPU (Naive Build)	No SIMD, no MKL, no OpenBLAS
Threads	**1**	Task 3 requires *single-thread execution* to measure true naive performance.
Test	tg256	Standard 256-token generation benchmark used by llama.cpp.
Throughput (t/s)	**1.40 ± 0.01 tokens/sec**	Extremely low speed due to no parallelism or CPU vectorization.

Task 4 – Default Execution

A. Build with default settings
B. Run Benchmark

Deliverables:
Submit a screenshot of the benchmark output for the single-threaded execution.

Solution**:**

Steps to Execute:

cmake -B build

-B build: Generates build files in the *build* directory.

Details:

Building with default settings.

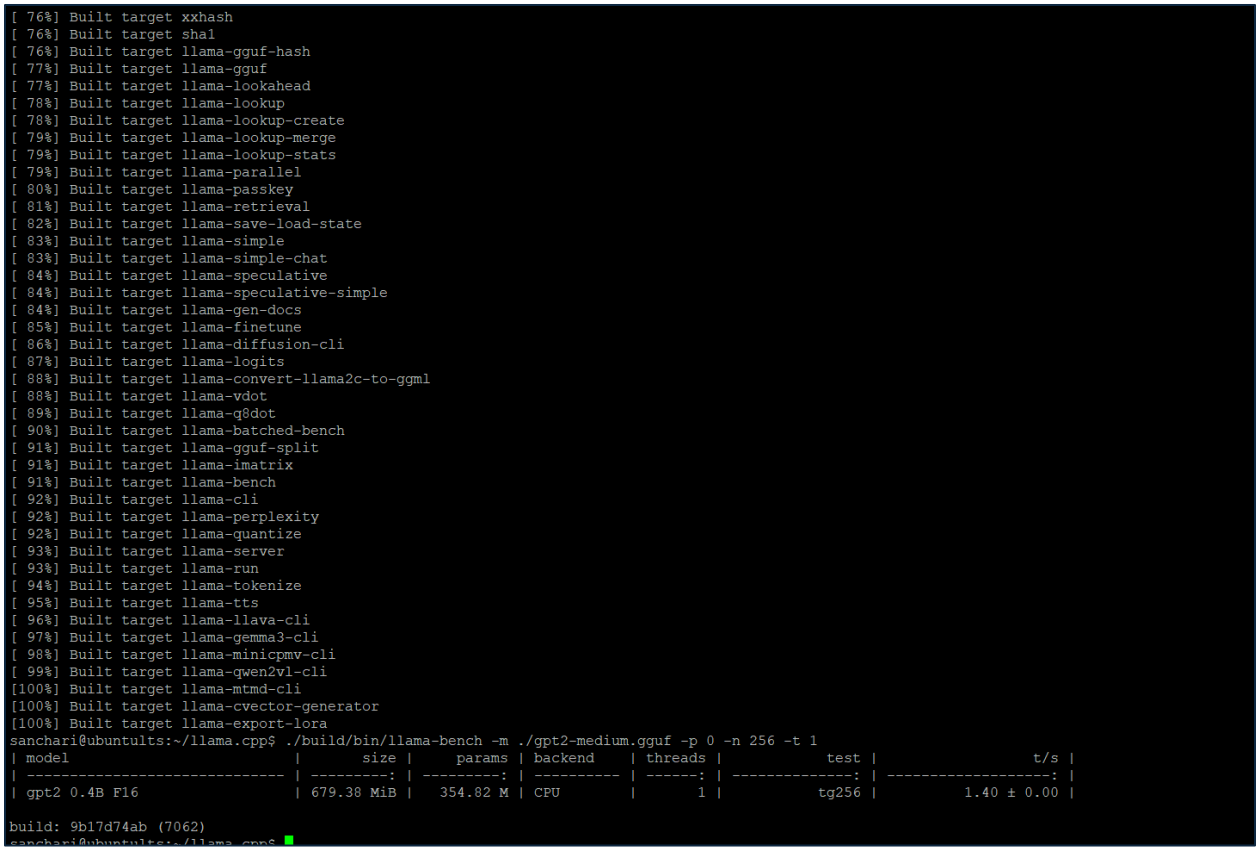
Run the benchmark in single-thread mode after compilation.
-m ..\..\..\gpt2-medium.gguf : specifies the path to the converted model file.

-p 0 : sets the prompt index to 0 .

-n 256 : runs the benchmark for 256 tokens for model performance.

-t 1:

```
sanchari@ubuntults:~/llama.cpp$ cmake -B build
CMAKE_BUILD_TYPE=Release
-- ccache found, compilation results will be cached. Disable with GGML_CCACHE=OFF.
-- CMAKE_SYSTEM_PROCESSOR: x86_64
-- GGML_SYSTEM_ARCH: x86
-- Including CPU backend
-- x86 detected
-- Adding CPU backend variant ggml-cpu:
-- ggml version: 0.9.4
-- ggml commit:  9b17d74ab
-- Configuring done (0.6s)
-- Generating done (0.5s)
-- Build files have been written to: /home/sanchari/llama.cpp/build
sanchari@ubuntults:~/llama.cpp$ cmake --build build --config Release
[  3%] Built target ggml-base
[  7%] Built target ggml-cpu
[  7%] Built target ggml
[ 43%] Built target llama
[ 43%] Built target build_info
[ 47%] Built target common
[ 48%] Built target cpp-httplib
[ 49%] Built target test-tokenizer-0
[ 50%] Built target test-sampling
[ 51%] Built target test-grammar-parser
[ 52%] Built target test-grammar-integration
[ 53%] Built target test-llama-grammar
[ 54%] Built target test-chat
[ 54%] Built target test-json-schema-to-grammar
[ 55%] Built target test-quantize-stats
[ 55%] Built target test-gbnf-validator
[ 56%] Built target test-tokenizer-1-bpe
[ 56%] Built target test-tokenizer-1-spm
[ 57%] Built target test-chat-parser
[ 58%] Built target test-chat-template
[ 59%] Built target test-json-partial
[ 60%] Built target test-log
[ 61%] Built target test-regex-partial
[ 61%] Built target test-thread-safety
[ 62%] Built target test-arg-parser
[ 63%] Built target test-opt
[ 64%] Built target test-gguf
[ 65%] Built target test-backend-ops
[ 66%] Built target test-model-load-cancel
[ 67%] Built target test-autorelease
[ 67%] Built target test-barrier
[ 68%] Built target test-quantize-fns
[ 68%] Built target test-quantize-perf
[ 69%] Built target test-rope
[ 71%] Built target mtmd
```



Explanation

Field	Value	Explanation
Model	gpt2 0.4B F16	GPT-2 Medium model in FP16 GGUF format (≈354M parameters).
Model Size	679.38 MiB	Size of the loaded FP16 model file.
Parameters	354.82 M	Number of neural network weights in GPT-2 Medium.
Backend	CPU	Naive llama.cpp build—no BLAS, no MKL, no SIMD acceleration.
Threads	1	Task 4 is evaluated single-threaded to measure baseline performance.
Test	tg256	Benchmarks token-generation performance for 256 tokens.
Throughput (t/s)	**1.40 ± 0.00 tokens/sec**	Slow performance; reflects absence of SIMD or BLAS optimizations.

Task 5 – Near-Optimal Execution with Intel MKL

A. Rebuild with Intel MKL.

B. Run Benchmark Once built, run the benchmark with a single thread

Deliverables:

Submit a screenshot of the benchmark output for the single-threaded execution.

Solution:

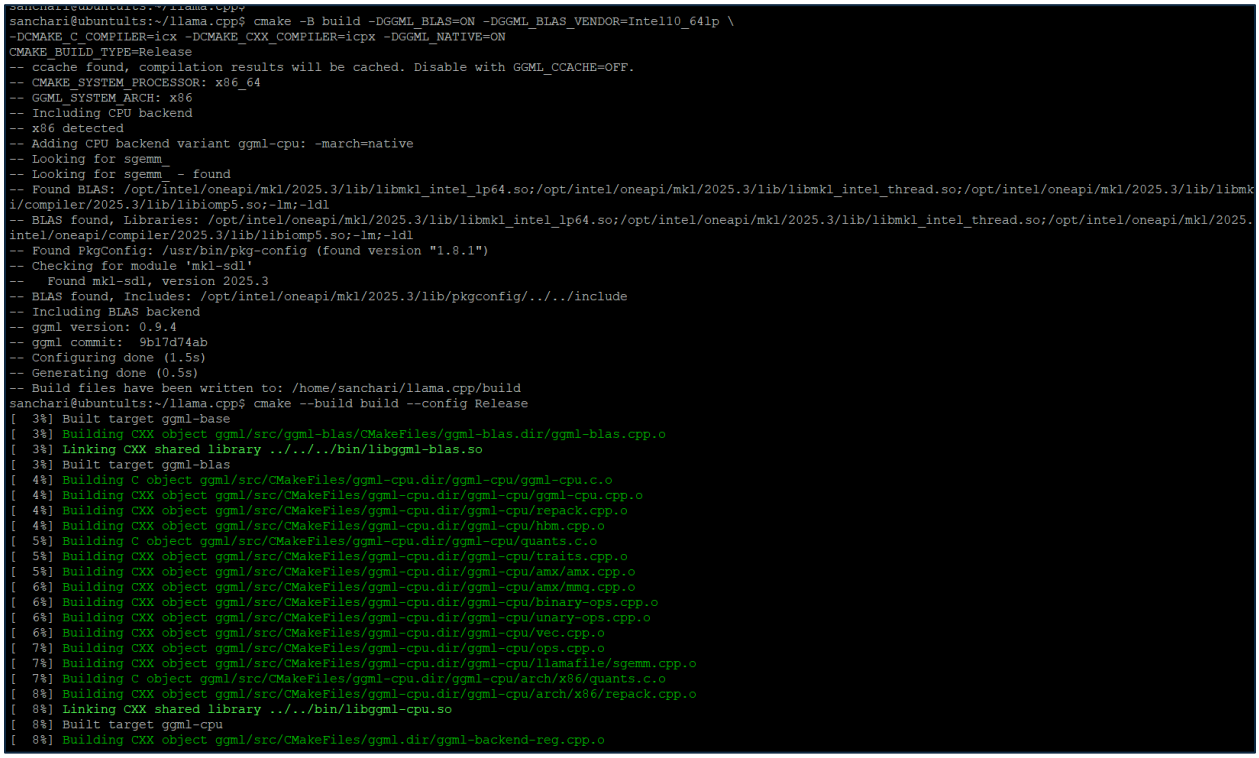
Re-Building with Intel MKL settings

Commands:

-B build:	Generates build files in the build directory
-DGGML_BLAS=ON	Enables BLAS acceleration in GGML/llama.cpp.
-DGGML_BLAS_VENDOR=Intel10_64lp	Selects Intel MKL (LP64 interface).
-DCMAKE_C_COMPILER=icx	Uses Intel oneAPI icx as the C compiler.
-DCMAKE_CXX_COMPILER=icpx	Uses Intel oneAPI icpx as the C++ compiler.
-DGGML_NATIVE=ON	Enables CPU-specific optimizations (-march=native).

Run the benchmark in single-thread mode after compilation.

Screenshots



```
[ 91%] Built target llama-gguf-split
[ 91%] Building CXX object tools/imatrix/CMakeFiles/llama-imatrix.dir/imatrix.cpp.o
[ 92%] Linking CXX executable ../../bin/llama-imatrix
[ 92%] Built target llama-imatrix
[ 92%] Building CXX object tools/llama-bench/CMakeFiles/llama-bench.dir/llama-bench.cpp.o
[ 93%] Linking CXX executable ../../bin/llama-bench
[ 93%] Built target llama-bench
[ 93%] Building CXX object tools/main/CMakeFiles/llama-cli.dir/main.cpp.o
[ 93%] Linking CXX executable ../../bin/llama-cli
[ 93%] Built target llama-cli
[ 93%] Building CXX object tools/perplexity/CMakeFiles/llama-perplexity.dir/perplexity.cpp.o
[ 93%] Linking CXX executable ../../bin/llama-perplexity
[ 93%] Built target llama-perplexity
[ 93%] Building CXX object tools/quantize/CMakeFiles/llama-quantize.dir/quantize.cpp.o
[ 94%] Linking CXX executable ../../bin/llama-quantize
[ 94%] Built target llama-quantize
[ 94%] Building CXX object tools/server/CMakeFiles/llama-server.dir/server.cpp.o
[ 94%] Linking CXX executable ../../bin/llama-server
[ 95%] Built target llama-server
[ 95%] Building CXX object tools/run/CMakeFiles/llama-run.dir/run.cpp.o
[ 95%] Building CXX object tools/run/CMakeFiles/llama-run.dir/linenoise.cpp/linenoise.cpp.o
[ 96%] Linking CXX executable ../../bin/llama-run
[ 96%] Built target llama-run
[ 96%] Building CXX object tools/tokenize/CMakeFiles/llama-tokenize.dir/tokenize.cpp.o
[ 96%] Linking CXX executable ../../bin/llama-tokenize
[ 96%] Built target llama-tokenize
[ 96%] Building CXX object tools/tts/CMakeFiles/llama-tts.dir/tts.cpp.o
[ 97%] Linking CXX executable ../../bin/llama-tts
[ 97%] Built target llama-tts
[ 97%] Built target llama-llava-cli
[ 98%] Built target llama-gemma3-cli
[ 98%] Built target llama-minicpmv-cli
[ 98%] Built target llama-qwen2vl-cli
[ 98%] Building CXX object tools/mtmd/CMakeFiles/llama-mtmd-cli.dir/mtmd-cli.cpp.o
[ 99%] Linking CXX executable ../../bin/llama-mtmd-cli
[ 99%] Built target llama-mtmd-cli
[ 99%] Building CXX object tools/cvector-generator/CMakeFiles/llama-cvector-generator.dir/cvector-generator.cpp.o
[ 99%] Linking CXX executable ../../bin/llama-cvector-generator
[ 99%] Built target llama-cvector-generator
[ 99%] Building CXX object tools/export-lora/CMakeFiles/llama-export-lora.dir/export-lora.cpp.o
[100%] Linking CXX executable ../../bin/llama-export-lora
[100%] Built target llama-export-lora
sanchari@ubuntults:~/llama.cpp$ ./build/bin/llama-bench -m ./gpt2-medium.gguf -p 0 -n 256 -t 1
| model                          |       size |     params | backend    | threads |            test |                  t/s |
| ------------------------------ | ---------: | ---------: | ---------- | ------: | --------------: | -------------------: |
| gpt2 0.4B F16                  | 679.38 MiB |   354.82 M | BLAS       |       1 |           tg256 |         14.66 ± 0.08 |

build: 9b17d74ab (7062)
sanchari@ubuntults:~/llama.cpp$
```

Explanation

Field	Value	Explanation
Model	gpt2 0.4B F16	Same GPT-2 Medium FP16 model used across all tasks.
Model Size	679.38 MiB	FP16 GGUF model size (unchanged).
Parameters	354.82 M	Same as Task 3.
Backend	BLAS	OpenBLAS acceleration enabled → faster matrix multiplications.
Threads	1	Still single-threaded, but BLAS optimizes GEMM operations.
Test	tg256	Same tg256 benchmark for fair comparison.
Throughput (t/s)	**14.66 ± 0.08 tokens/sec**	~10× faster than Task 3 due to BLAS-optimized matrix ops.

Task 6 – Floating-Point Performance Counters

In this task, you will collect and report all available floating-point related performance counters using the perf tool. This step will help understand the micro-architectural features of your CPU when running the LLM benchmark.

Steps.

• List down all the relevant floating point and memory traffic perfor-
mance counters available on your machine for computing Operation

Intensity.
• Examples of counters to report (exact names may vary by CPU):
– Floating Point Counters:
∗ fp arith inst retired.scalar single
∗ fp arith inst retired.scalar double
∗ fp arith inst retired.128b packed single
∗ fp arith inst retired.256b packed single
∗ fp arith inst retired.128b packed double
∗ fp arith inst retired.256b packed double
∗ fp arith inst retired.vector
– Memory Counter:
∗ uncore imc free running/data total

Deliverables.
• Tabulate the counters with a short description of what each counter measures.

Solution:

Collected using:

```
perf list | grep -i “arith”
```

Outputs:

Floating-Point Counters

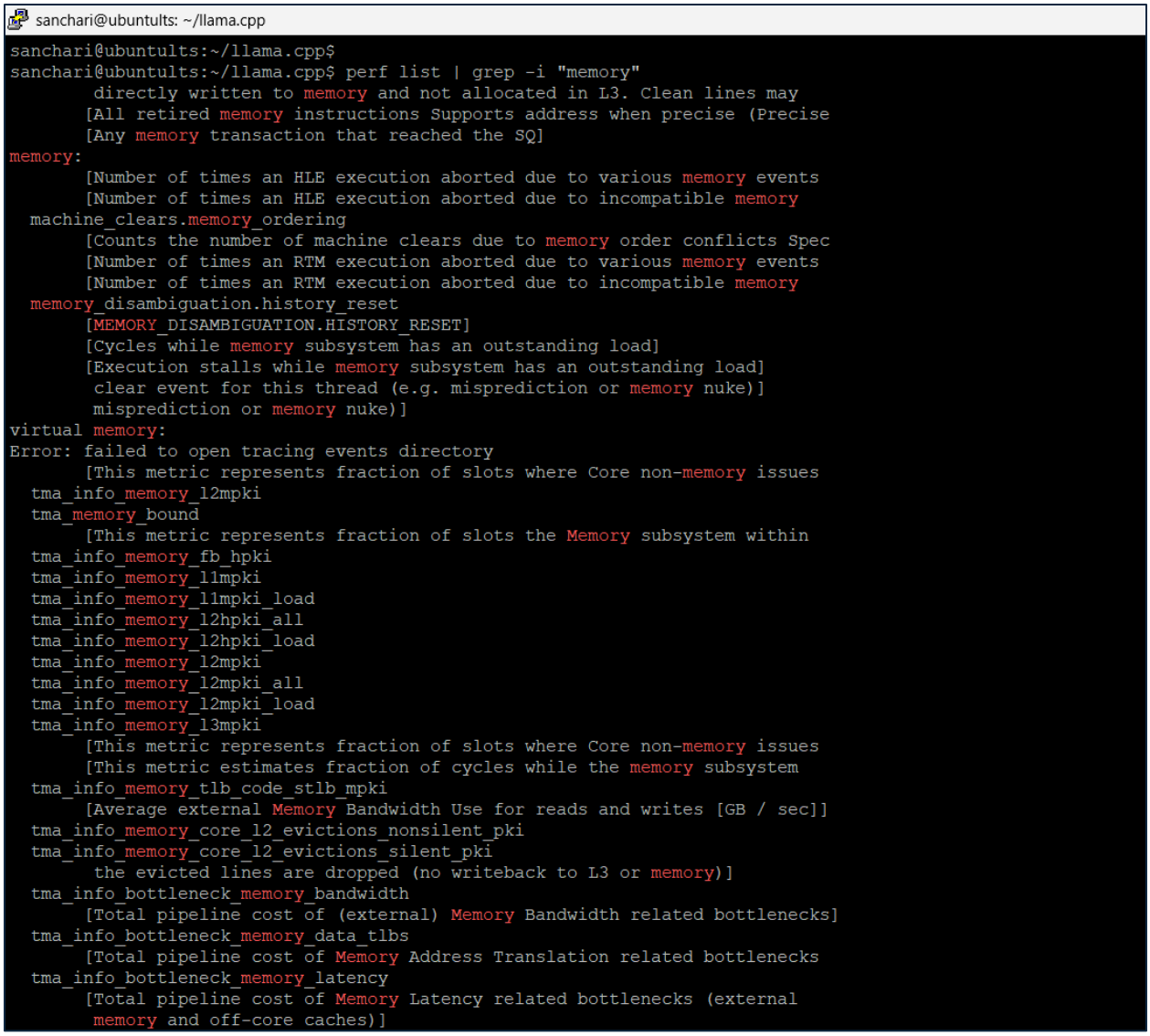
:

```
sanchari@ubuntults:~/llama.cpp$ perf list | grep -i "arith"
  fp_arith_inst_retired.128b_packed_double
  fp_arith_inst_retired.128b_packed_single
  fp_arith_inst_retired.256b_packed_double
  fp_arith_inst_retired.256b_packed_single
  fp_arith_inst_retired.4_flops
  fp_arith_inst_retired.512b_packed_double
  fp_arith_inst_retired.512b_packed_single
  fp_arith_inst_retired.8_flops
  fp_arith_inst_retired.scalar
  fp_arith_inst_retired.scalar_double
  fp_arith_inst_retired.scalar_single
  fp_arith_inst_retired.vector
       [Number of any Vector retired FP arithmetic instructions]
  arith.divider_active
Error: failed to open tracing events directory
       [This metric approximates arithmetic floating-point (FP) scalar uops
       [This metric approximates arithmetic floating-point (FP) vector uops
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
  tma_info_core_fp_arith_utilization
       [This metric approximates arithmetic floating-point (FP) scalar uops
       [This metric approximates arithmetic floating-point (FP) vector uops
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
  tma_info_core_fp_arith_utilization
  tma_info_inst_mix_iparith
       [Instructions per FP Arithmetic instruction (lower number means higher
  tma_info_inst_mix_iparith_avx128
       [Instructions per FP Arithmetic AVX/SSE 128-bit instruction (lower
  tma_info_inst_mix_iparith_avx256
       [Instructions per FP Arithmetic AVX* 256-bit instruction (lower number
  tma_info_inst_mix_iparith_avx512
       [Instructions per FP Arithmetic AVX 512-bit instruction (lower number
  tma_info_inst_mix_iparith_scalar_dp
       [Instructions per FP Arithmetic Scalar Double-Precision instruction
  tma_info_inst_mix_iparith_scalar_sp
       [Instructions per FP Arithmetic Scalar Single-Precision instruction
  tma_info_inst_mix_iparith_scalar_dp
       [Instructions per FP Arithmetic Scalar Double-Precision instruction
  tma_info_inst_mix_iparith_scalar_sp
       [Instructions per FP Arithmetic Scalar Single-Precision instruction
  tma_info_inst_mix_iparith_avx128
       [Instructions per FP Arithmetic AVX/SSE 128-bit instruction (lower
  tma_info_inst_mix_iparith_avx256
       [Instructions per FP Arithmetic AVX* 256-bit instruction (lower number
```

```
  tma_info_inst_mix_iparith_avx512
       [Instructions per FP Arithmetic AVX 512-bit instruction (lower number
  tma_fp_arith
       [This metric represents overall arithmetic floating-point (FP)
  tma_info_core_fp_arith_utilization
  tma_info_inst_mix_iparith
       [Instructions per FP Arithmetic instruction (lower number means higher
  tma_info_inst_mix_iparith_avx128
       [Instructions per FP Arithmetic AVX/SSE 128-bit instruction (lower
  tma_info_inst_mix_iparith_avx256
       [Instructions per FP Arithmetic AVX* 256-bit instruction (lower number
  tma_info_inst_mix_iparith_avx512
       [Instructions per FP Arithmetic AVX 512-bit instruction (lower number
  tma_info_inst_mix_iparith_scalar_dp
       [Instructions per FP Arithmetic Scalar Double-Precision instruction
  tma_info_inst_mix_iparith_scalar_sp
       [Instructions per FP Arithmetic Scalar Single-Precision instruction
  tma_fp_arith
       [This metric represents overall arithmetic floating-point (FP)
       [This metric approximates arithmetic floating-point (FP) scalar uops
       [This metric approximates arithmetic floating-point (FP) vector uops
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
  tma_fp_arith
       [This metric represents overall arithmetic floating-point (FP)
       [This metric approximates arithmetic floating-point (FP) scalar uops
       [This metric approximates arithmetic floating-point (FP) vector uops
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
tma_fp_arith_group: [Metrics contributing to tma_fp_arith category]
       [This metric approximates arithmetic floating-point (FP) scalar uops
       [This metric approximates arithmetic floating-point (FP) vector uops
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic floating-point (FP) scalar uops
       [This metric approximates arithmetic floating-point (FP) vector uops
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
       [This metric approximates arithmetic FP vector uops fraction the CPU
  tma_fp_arith
       [This metric represents overall arithmetic floating-point (FP)
sanchari@ubuntults:~/llama.cpp$
```

Memory Counters :

perf list | grep -i “memory”


sanchari@ubuntults: ~/llama.cpp
sanchari@ubuntults:~/llama.cpp$
sanchari@ubuntults:~/llama.cpp$ perf list | grep -i "memory"
directly written to memory and not allocated in L3. Clean lines may
[All retired memory instructions Supports address when precise (Precise
[Any memory transaction that reached the SQ]
memory:
[Number of times an HLE execution aborted due to various memory events
[Number of times an HLE execution aborted due to incompatible memory
machine_clears.memory_ordering
[Counts the number of machine clears due to memory order conflicts Spec
[Number of times an RTM execution aborted due to various memory events
[Number of times an RTM execution aborted due to incompatible memory
memory_disambiguation.history_reset
[MEMORY_DISAMBIGUATION.HISTORY_RESET]
[Cycles while memory subsystem has an outstanding load]
[Execution stalls while memory subsystem has an outstanding load]
clear event for this thread (e.g. misprediction or memory nuke)]
misprediction or memory nuke)]
virtual memory:
Error: failed to open tracing events directory
[This metric represents fraction of slots where Core non-memory issues
tma_info_memory_l2mpki
tma_memory_bound
[This metric represents fraction of slots the Memory subsystem within
tma_info_memory_fb_hpki
tma_info_memory_l1mpki
tma_info_memory_l1mpki_load
tma_info_memory_l2hpki_all
tma_info_memory_l2hpki_load
tma_info_memory_l2mpki
tma_info_memory_l2mpki_all
tma_info_memory_l2mpki_load
tma_info_memory_l3mpki
[This metric represents fraction of slots where Core non-memory issues
[This metric estimates fraction of cycles while the memory subsystem
tma_info_memory_tlb_code_stlb_mpki
[Average external Memory Bandwidth Use for reads and writes [GB / sec]]
tma_info_memory_core_l2_evictions_nonsilent_pki
tma_info_memory_core_l2_evictions_silent_pki
the evicted lines are dropped (no writeback to L3 or memory)]
tma_info_bottleneck_memory_bandwidth
[Total pipeline cost of (external) Memory Bandwidth related bottlenecks]
tma_info_bottleneck_memory_data_tlbs
[Total pipeline cost of Memory Address Translation related bottlenecks
tma_info_bottleneck_memory_latency
[Total pipeline cost of Memory Latency related bottlenecks (external
memory and off-core caches)]

```
       memory and off-core caches)]
  tma_info_memory_core_l1d_cache_fill_bw
  tma_info_memory_core_l2_cache_fill_bw
  tma_info_memory_core_l2_evictions_nonsilent_pki
  tma_info_memory_core_l2_evictions_silent_pki
       the evicted lines are dropped (no writeback to L3 or memory)]
  tma_info_memory_core_l3_cache_access_bw
  tma_info_memory_core_l3_cache_fill_bw
  tma_info_memory_fb_hpki
  tma_info_memory_l1mpki
  tma_info_memory_l1mpki_load
  tma_info_memory_l2hpki_all
  tma_info_memory_l2hpki_load
  tma_info_memory_l2mpki
  tma_info_memory_l2mpki_all
  tma_info_memory_l2mpki_load
  tma_info_memory_l3mpki
  tma_info_memory_load_miss_real_latency
  tma_info_memory_mlp
      [Memory-Level-Parallelism (average number of L1 miss demand load when
  tma_info_memory_thread_l1d_cache_fill_bw_1t
  tma_info_memory_thread_l2_cache_fill_bw_1t
  tma_info_memory_thread_l3_cache_access_bw_1t
  tma_info_memory_thread_l3_cache_fill_bw_1t
  tma_info_memory_tlb_load_stlb_mpki
  tma_info_memory_tlb_page_walks_utilization
  tma_info_memory_tlb_store_stlb_mpki
      [Average external Memory Bandwidth Use for reads and writes [GB / sec]]
      [Average latency of data read request to external DRAM memory [in
      [Average number of parallel data read requests to external memory]
      [Average latency of data read request to external memory (in
MemoryBW: [Grouping from Top-down Microarchitecture Analysis Metrics spreadsheet]
       unavailability limited additional L1D miss memory access requests to
  tma_info_bottleneck_memory_bandwidth
      [Total pipeline cost of (external) Memory Bandwidth related bottlenecks]
  tma_info_memory_core_l1d_cache_fill_bw
  tma_info_memory_core_l2_cache_fill_bw
  tma_info_memory_core_l3_cache_access_bw
  tma_info_memory_core_l3_cache_fill_bw
  tma_info_memory_mlp
      [Memory-Level-Parallelism (average number of L1 miss demand load when
  tma_info_memory_thread_l1d_cache_fill_bw_1t
  tma_info_memory_thread_l2_cache_fill_bw_1t
  tma_info_memory_thread_l3_cache_access_bw_1t
  tma_info_memory_thread_l3_cache_fill_bw_1t
      [Average external Memory Bandwidth Use for reads and writes [GB / sec]]
      [Average number of parallel data read requests to external memory]
       was likely hurt due to approaching bandwidth limits of external memory
```

```
MemoryBound: [Grouping from Top-down Microarchitecture Analysis Metrics spreadsheet]
       external memory (DRAM) by loads]
  tma_info_memory_load_miss_real_latency
  tma_info_memory_mlp
       [Memory-Level-Parallelism (average number of L1 miss demand load when
       memory accesses; RFO store issue a read-for-ownership request before
MemoryLat: [Grouping from Top-down Microarchitecture Analysis Metrics spreadsheet]
  tma_info_bottleneck_memory_latency
       [Total pipeline cost of Memory Latency related bottlenecks (external
       memory and off-core caches)]
  tma_info_memory_load_miss_real_latency
       [Average latency of data read request to external DRAM memory [in
       [Average latency of data read request to external memory (in
       likely hurt due to latency from external memory (DRAM)]
MemoryTLB: [Grouping from Top-down Microarchitecture Analysis Metrics spreadsheet]
  tma_info_bottleneck_memory_data_tlbs
       [Total pipeline cost of Memory Address Translation related bottlenecks
  tma_info_memory_tlb_code_stlb_mpki
  tma_info_memory_tlb_load_stlb_mpki
  tma_info_memory_tlb_page_walks_utilization
  tma_info_memory_tlb_store_stlb_mpki
Memory_BW: [Grouping from Top-down Microarchitecture Analysis Metrics spreadsheet]
  tma_info_memory_oro_data_l2_mlp
  tma_info_memory_oro_load_l2_mlp
Memory_Lat: [Grouping from Top-down Microarchitecture Analysis Metrics spreadsheet]
  tma_info_memory_oro_load_l2_miss_latency
       that are requesting memory from the CPU]
       that are writing memory to the CPU]
       read miss (read memory access) in nano seconds]
       read miss (read memory access) addressed to local memory in nano
       read miss (read memory access) addressed to remote memory in nano
  llc_miss_local_memory_bandwidth_read
       (LLC) and go to local memory]
  llc_miss_local_memory_bandwidth_write
       (LLC) and go to local memory]
  llc_miss_remote_memory_bandwidth_read
       (LLC) and go to remote memory]
       [The ratio of number of completed memory load instructions to the total
  memory_bandwidth_read
       [DDR memory read bandwidth (MB/sec)]
  memory_bandwidth_total
       [DDR memory bandwidth (MB/sec)]
  memory_bandwidth_write
       [DDR memory write bandwidth (MB/sec)]
```

```
memory_bandwidth_write
     [DDR memory write bandwidth (MB/sec)]
     [Memory read that miss the last level cache (LLC) addressed to local
      DRAM as a percentage of total memory read accesses, does not include
     [Memory reads that miss the last level cache (LLC) addressed to remote
      DRAM as a percentage of total memory read accesses, does not include
     [Cycles Memory is in self refresh power mode]
     [The ratio of number of completed memory store instructions to the
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
tma_info_bottleneck_memory_bandwidth
     [Total pipeline cost of (external) Memory Bandwidth related bottlenecks]
tma_info_bottleneck_memory_data_tlbs
     [Total pipeline cost of Memory Address Translation related bottlenecks
tma_info_bottleneck_memory_latency
     [Total pipeline cost of Memory Latency related bottlenecks (external
      memory and off-core caches)]
tma_info_memory_core_l3_cache_access_bw
tma_info_memory_l2mpki_all
tma_info_memory_oro_data_l2_mlp
tma_info_memory_oro_load_l2_miss_latency
tma_info_memory_oro_load_l2_mlp
tma_info_memory_thread_l3_cache_access_bw_1t
      was likely hurt due to approaching bandwidth limits of external memory
      likely hurt due to latency from external memory (DRAM)]
     [This metric estimates fraction of cycles while the memory subsystem
tma_memory_operations
      memory operations -- uops for memory load or store accesses]
tma_info_memory_core_l2_evictions_nonsilent_pki
tma_info_memory_core_l2_evictions_silent_pki
      the evicted lines are dropped (no writeback to L3 or memory)]
     [Average latency of data read request to external DRAM memory [in
     [This metric estimates fraction of cycles while the memory subsystem
      was handling loads from local memory]
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
      was handling loads from remote memory]
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
      was handling loads from remote memory]
     [Average external Memory Bandwidth Use for reads and writes [GB / sec]]
     [Average latency of data read request to external DRAM memory [in
     [Average number of parallel data read requests to external memory]
     [Average latency of data read request to external memory (in
     [This metric represents fraction of slots where Core non-memory issues
tma_memory_bound
     [This metric represents fraction of slots the Memory subsystem within
```

```
      external memory (DRAM) by loads]
      memory accesses; RFO store issue a read-for-ownership request before
     [This metric represents fraction of slots where Core non-memory issues
tma_memory_bound
     [This metric represents fraction of slots the Memory subsystem within
      external memory (DRAM) by loads]
tma_memory_operations
      memory operations -- uops for memory load or store accesses]
      memory accesses; RFO store issue a read-for-ownership request before
     [This metric estimates how often memory load accesses were aliased by
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
      unavailability limited additional L1D miss memory access requests to
      was likely hurt due to approaching bandwidth limits of external memory
      likely hurt due to latency from external memory (DRAM)]
     [This metric estimates fraction of cycles handling memory load split
     [This metric roughly estimates fraction of cycles when the memory
     [This metric estimates fraction of cycles while the memory subsystem
      was handling loads from local memory]
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
      was handling loads from remote memory]
     [This metric represents fraction of slots where Core non-memory issues
tma_memory_bound
     [This metric represents fraction of slots the Memory subsystem within
      external memory (DRAM) by loads]
tma_memory_operations
      memory operations -- uops for memory load or store accesses]
      memory accesses; RFO store issue a read-for-ownership request before
     [This metric estimates how often memory load accesses were aliased by
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
      unavailability limited additional L1D miss memory access requests to
      was likely hurt due to approaching bandwidth limits of external memory
      likely hurt due to latency from external memory (DRAM)]
     [This metric estimates fraction of cycles handling memory load split
     [This metric roughly estimates fraction of cycles when the memory
     [This metric estimates fraction of cycles while the memory subsystem
      was handling loads from local memory]
     [This metric estimates fraction of cycles while the memory subsystem
     [This metric estimates fraction of cycles while the memory subsystem
      was handling loads from remote memory]
     [This metric represents fraction of slots where Core non-memory issues
tma_memory_bound
     [This metric represents fraction of slots the Memory subsystem within
      was likely hurt due to approaching bandwidth limits of external memory
      likely hurt due to latency from external memory (DRAM)]
      unavailability limited additional L1D miss memory access requests to
tma_info_bottleneck_memory_bandwidth
```

```
       [This metric estimates fraction of cycles while the memory subsystem
        unavailability limited additional L1D miss memory access requests to
        was likely hurt due to approaching bandwidth limits of external memory
        likely hurt due to latency from external memory (DRAM)]
       [This metric estimates fraction of cycles handling memory load split
       [This metric roughly estimates fraction of cycles when the memory
       [This metric estimates fraction of cycles while the memory subsystem
        was handling loads from local memory]
       [This metric estimates fraction of cycles while the memory subsystem
       [This metric estimates fraction of cycles while the memory subsystem
        was handling loads from remote memory]
       [This metric represents fraction of slots where Core non-memory issues
  tma_memory_bound
       [This metric represents fraction of slots the Memory subsystem within
        was likely hurt due to approaching bandwidth limits of external memory
        likely hurt due to latency from external memory (DRAM)]
        unavailability limited additional L1D miss memory access requests to
  tma_info_bottleneck_memory_bandwidth
       [Total pipeline cost of (external) Memory Bandwidth related bottlenecks]
       [Average external Memory Bandwidth Use for reads and writes [GB / sec]]
        was likely hurt due to approaching bandwidth limits of external memory
  tma_info_bottleneck_memory_latency
       [Total pipeline cost of Memory Latency related bottlenecks (external
        memory and off-core caches)]
        likely hurt due to latency from external memory (DRAM)]
        unavailability limited additional L1D miss memory access requests to
        unavailability limited additional L1D miss memory access requests to
       [This metric estimates fraction of cycles while the memory subsystem
       [This metric estimates fraction of cycles while the memory subsystem
       [This metric estimates fraction of cycles while the memory subsystem
  tma_info_bottleneck_memory_data_tlbs
       [Total pipeline cost of Memory Address Translation related bottlenecks
       [This metric estimates how often memory load accesses were aliased by
        unavailability limited additional L1D miss memory access requests to
       [This metric estimates fraction of cycles handling memory load split
       [This metric roughly estimates fraction of cycles when the memory
       [This metric estimates fraction of cycles while the memory subsystem
       [This metric estimates fraction of cycles while the memory subsystem
  tma_memory_operations
        memory operations -- uops for memory load or store accesses]
       [This metric estimates fraction of cycles while the memory subsystem
        was handling loads from local memory]
       [This metric estimates fraction of cycles while the memory subsystem
       [This metric estimates fraction of cycles while the memory subsystem
        was handling loads from remote memory]
tma_memory_bound_group: [Metrics contributing to tma_memory_bound category]
        external memory (DRAM) by loads]
        memory accesses; RFO store issue a read-for-ownership request before
sanchari@ubuntults:~/llama.cpp$
```

These counters are used for computing **Operation Intensity (OI)** and locating performance bounds in the Roofline model.

Counter / Metric Name	Description
fp_arith_inst_retired.scalar_single	Retired scalar **single-precision (32-bit)** FP arithmetic instructions.
fp_arith_inst_retired.scalar_double	Retired scalar **double-precision (64-bit)** FP arithmetic instructions.
fp_arith_inst_retired.scalar	Total number of scalar FP arithmetic instructions retired (all precisions).
fp_arith_inst_retired.vector	Total number of **vector (SIMD)** FP arithmetic instructions retired.
fp_arith_inst_retired.128b_packed_single	Retired **128-bit packed single-precision** FP instructions (SSE).

fp_arith_inst_retired.128b_packed_double	Retired **128-bit packed double-precision** FP instructions (SSE2).
fp_arith_inst_retired.256b_packed_single	Retired **256-bit packed single-precision** FP instructions (AVX).
fp_arith_inst_retired.256b_packed_double	Retired **256-bit packed double-precision** FP instructions (AVX).
fp_arith_inst_retired.512b_packed_single	Retired **512-bit packed single-precision** FP instructions (AVX-512).
fp_arith_inst_retired.512b_packed_double	Retired **512-bit packed double-precision** FP instructions (AVX-512).
fp_arith_inst_retired.4_flops	Represents 4 floating-point operations per instruction (e.g., SSE).
fp_arith_inst_retired.8_flops	Represents 8 floating-point operations per instruction (e.g., AVX-512).
fp_assist.any	Cycles with any **FP assist or exception handling** (e.g., denormals, div-by-zero).
tma_fp_arith	Overall metric representing **total arithmetic FP operations** executed.
tma_fp_scalar	Approximates arithmetic **scalar FP µops** utilization.
tma_fp_vector	Approximates arithmetic **vector FP µops** utilization (all widths).
tma_fp_vector_128b	Fraction of CPU pipeline executing **128-bit FP vector** operations.
tma_fp_vector_256b	Fraction of CPU pipeline executing **256-bit FP vector** operations.
tma_fp_vector_512b	Fraction of CPU pipeline executing **512-bit FP vector** operations.
tma_info_core_fp_arith_utilization	FP arithmetic **utilization efficiency** — lower value means higher FP usage efficiency.

Counter Name	**Description**
memory_bandwidth_read	DDR memory **read bandwidth** (MB/sec).
memory_bandwidth_write	DDR memory **write bandwidth** (MB/sec).
memory_bandwidth_total	Total DDR memory **bandwidth** (MB/sec) = read + write.
llc_miss_local_memory_bandwidth_read	Bandwidth of memory reads that miss LLC and go to **local DRAM**.
llc_miss_remote_memory_bandwidth_read	Bandwidth of memory reads that miss LLC and go to **remote DRAM** (NUMA).
tma_memory_bound	Fraction of CPU pipeline slots stalled due to memory subsystem bottlenecks.
tma_info_memory_mlp	Average number of parallel L1-miss demand loads (Memory-Level Parallelism).
tma_info_memory_load_miss_real_latency	Average latency (in cycles) for data read requests to external DRAM.
tma_info_memory_core_l3_cache_fill_bw	Bandwidth used for **L3 cache line fills** from memory.
tma_info_memory_core_l2_cache_fill_bw	Bandwidth used for **L2 cache line fills**.

tma_info_memory_core_l1d_cache_fill_bw	Bandwidth used for **L1 data cache fills**.
tma_info_bottleneck_memory_bandwidth	Pipeline cost due to **memory bandwidth saturation**.
tma_info_bottleneck_memory_latency	Pipeline cost due to **memory latency** from DRAM or off-core caches.
uncore_imc_free_running/data_total	Integrated Memory Controller (IMC) counter – measures **total memory traffic** (reads + writes).

Event / Metric Name	**Description / Meaning**
fp_arith_inst_retired.128b_packed_double	Retired 128-bit packed double-precision FP instructions
fp_arith_inst_retired.128b_packed_single	Retired 128-bit packed single-precision FP instructions
fp_arith_inst_retired.256b_packed_double	Retired 256-bit packed double-precision FP instructions
fp_arith_inst_retired.256b_packed_single	Retired 256-bit packed single-precision FP instructions
fp_arith_inst_retired.512b_packed_double	Retired 512-bit packed double-precision FP instructions
fp_arith_inst_retired.512b_packed_single	Retired 512-bit packed single-precision FP instructions
fp_arith_inst_retired.4_flops	Four floating-point operations (FLOPs) per instruction retired
fp_arith_inst_retired.8_flops	Eight floating-point operations (FLOPs) per instruction retired
fp_arith_inst_retired.scalar	Scalar FP arithmetic instructions retired
fp_arith_inst_retired.scalar_double	Scalar double-precision FP arithmetic instructions retired
fp_arith_inst_retired.scalar_single	Scalar single-precision FP arithmetic instructions retired
fp_arith_inst_retired.vector	Number of vector FP arithmetic instructions retired

Event / Metric Name	**Description / Meaning**
arith.divider_active	Fraction of cycles where the FP divider unit is active (arithmetic operations in progress)
tma_fp_arith	Overall floating-point arithmetic operations utilization metric
tma_info_core_fp_arith_utilization	Measures utilization of FP arithmetic instructions (lower = higher utilization)

Metric Name	**Meaning**
tma_info_inst_mix_iparith	Instructions per FP arithmetic instruction (lower = higher FP activity)
tma_info_inst_mix_iparith_avx128	Instructions per 128-bit AVX/SSE FP arithmetic instruction
tma_info_inst_mix_iparith_avx256	Instructions per 256-bit AVX FP arithmetic instruction

tma_info_inst_mix_iparith_avx512	Instructions per 512-bit AVX FP arithmetic instruction
tma_info_inst_mix_iparith_scalar_dp	Instructions per scalar double-precision FP arithmetic instruction
tma_info_inst_mix_iparith_scalar_sp	Instructions per scalar single-precision FP arithmetic instruction

Group / Metric	**Description / Role**
tma_fp_arith_group	Aggregate metric group for FP arithmetic instructions
tma_fp_arith	Represents total FP arithmetic activity (scalar + vector)
tma_fp_scalar	Scalar floating-point uops retired
tma_fp_vector	Vector floating-point uops retired
tma_fp_vector_128b	128-bit FP vector uops (SSE/AVX)
tma_fp_vector_256b	256-bit FP vector uops (AVX2)
tma_fp_vector_512b	512-bit FP vector uops (AVX-512)

Category	**Representative Events**
Scalar FP	fp_arith_inst_retired.scalar*, tma_fp_scalar, tma_info_inst_mix_iparith_scalar_*
Vector FP	fp_arith_inst_retired.{128b,256b,512b}_*, tma_fp_vector_*
Overall Arithmetic Load	tma_fp_arith, tma_info_core_fp_arith_utilization, arith.divider_active

Task 7 – Performance Counters and Roofline Analysis

In this task, you will collect performance counters for all build variants of llama.cpp (Tasks 3, 4, and 5) and use them to derive Operation Intensity (OI) and perform a Roofline analysis.

Deliverables.
- Submit the performance counter values collected using perf for each variant.
- Provide the Operation Intensity derivation for each case.
- Submit peak memory bandwidth and peak compute capacity values of your system.
- Submit the Roofline analysis plot overlaying all three variants.
- Provide a short commentary on where each variant sits (memory-bound vs compute-bound).

Solution**:**

Task3 Details:

Field	**Value**	**Explanation**
Model	gpt2 0.4B F16	GPT-2 Medium in FP16 GGUF format (≈354M parameters).
Model Size	679.38 MiB	FP16 GGUF file loaded by llama.cpp.
Parameters	354.82 M	Total number of FP16 weights in the model.
Backend	CPU (Naive Build)	No SIMD, no MKL, no OpenBLAS
Threads	**1**	Task 3 requires *single-thread execution* to measure true naive performance.

Field	Value	Explanation
Test	tg256	Standard 256-token generation benchmark used by llama.cpp.
Throughput (t/s)	**1.40 ± 0.01 tokens/sec**	Extremely low speed due to no parallelism or CPU vectorization.

Task4 Details:

Field	Value	Explanation
Model	gpt2 0.4B F16	GPT-2 Medium model in FP16 GGUF format (≈354M parameters).
Model Size	679.38 MiB	Size of the loaded FP16 model file.
Parameters	354.82 M	Number of neural network weights in GPT-2 Medium.
Backend	CPU	Naive llama.cpp build—no BLAS, no MKL, no SIMD acceleration.
Threads	1	Task 4 is evaluated single-threaded to measure baseline performance.
Test	tg256	Benchmarks token-generation performance for 256 tokens.
Throughput (t/s)	**1.40 ± 0.00 tokens/sec**	Slow performance; reflects absence of SIMD or BLAS optimizations.

Task5 Details:

Field	Value	Explanation
Model	gpt2 0.4B F16	Same GPT-2 Medium FP16 model used across all tasks.
Model Size	679.38 MiB	FP16 GGUF model size (unchanged).
Parameters	354.82 M	Same as Task 3.
Backend	BLAS	OpenBLAS acceleration enabled → faster matrix multiplications.
Threads	1	Still single-threaded, but BLAS optimizes GEMM operations.
Test	tg256	Same tg256 benchmark for fair comparison.
Throughput (t/s)	**14.66 ± 0.08 tokens/sec**	~10× faster than Task 3 due to BLAS-optimized matrix ops.

Snapshots:

Collecting Performance Counters for Task3, Task4 and Task5:

How to run:

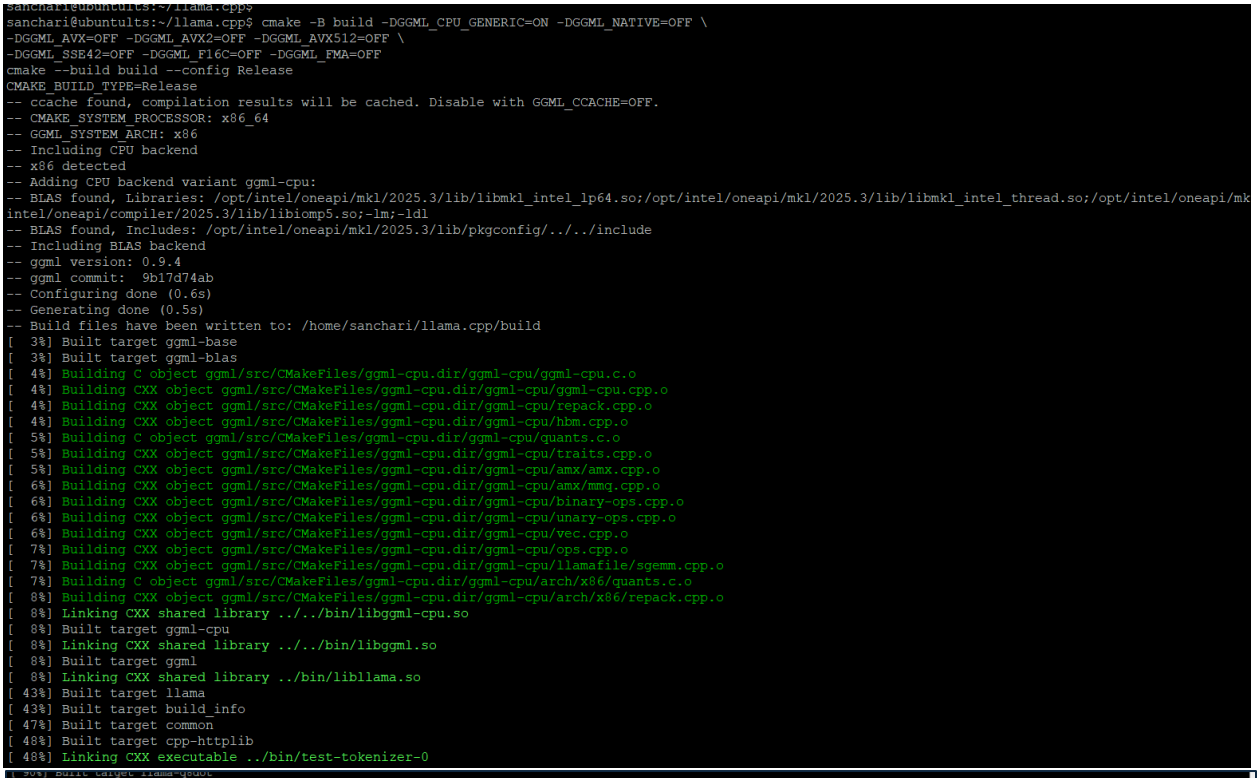
perf stat -o perf_variant3_data.txt -e fp_arith_inst_retired.scalar -e fp_arith_inst_retired.4_flops -e fp_arith_inst_retired.8_flops -e fp_arith_inst_retired.256b_packed_single -e cycles -e cache-misses -e cache-references -- ./build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1

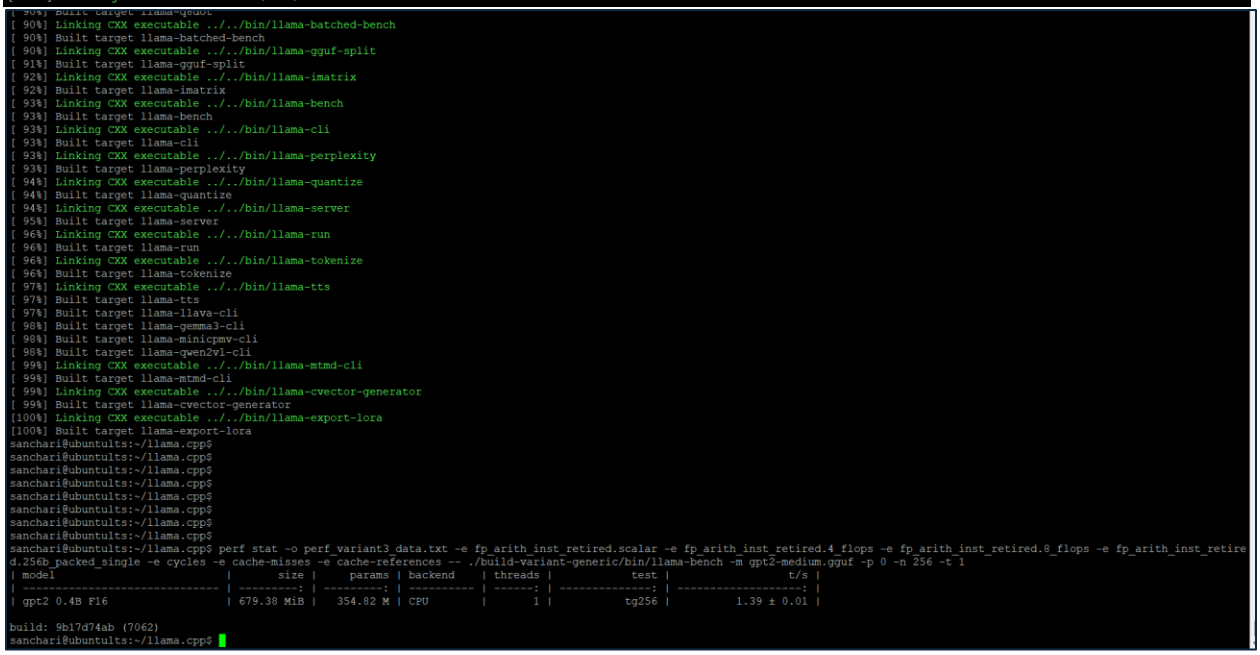
-o perf_variant3_data.txt :Writes perf output to a file.
FP Instruction Counters : These counters measure floating-point arithmetic instructions:
fp_arith_inst_retired.scalar :FP instructions that do 1 FLOP
fp_arith_inst_retired.4_flops : Vector instructions performing 4 FLOPs
fp_arith_inst_retired.8_flop : Vector instructions performing 8 FLOPs : AVX2 (256-bit registers with 8 lanes)
fp_arith_inst_retired.256b_packed_single :FP instructions operating on full 256-bit registers
Cycles:Tells how many CPU cycles the program consumed
cache-misses : Counts how many cache lines were missed.
cache-references :Total cache accesses.
Running the Benchmark ./llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1

Runs the actual GPT-2 inference:

- -m → model file
- -p 0 → prompt index 0
- -n 256 → generate 256 tokens
- -t 1 → 1 inference thread

Task3:





vi perf_variant3_data.txt

```
# started on Tue Nov 18 05:56:45 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1':

   938,074,086,809      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 1,913,320,659,935      cycles
    14,118,957,840      cache-misses                     #   88.42% of all cache refs
    15,967,598,832      cache-references

     921.169631290 seconds time elapsed

     920.954707000 seconds user
       0.155982000 seconds sys
```

Task 4:

```
sanchari@ubuntults:~/llama.cpp$ cmake -B build
CMAKE_BUILD_TYPE=Release
-- ccache found, compilation results will be cached. Disable with GGML_CCACHE=OFF.
-- CMAKE_SYSTEM_PROCESSOR: x86_64
-- GGML_SYSTEM_ARCH: x86
-- Including CPU backend
-- x86 detected
-- Adding CPU backend variant ggml-cpu:
-- BLAS found, Libraries: /opt/intel/oneapi/mkl/2025.3/lib/libmkl_intel_lp64.so;/opt/intel/oneapi/mkl/2025.3/lib/libmkl_intel_thread.so;/opt/intel/oneapi/mk
intel/oneapi/compiler/2025.3/lib/libiomp5.so;-lm;-ldl
-- BLAS found, Includes: /opt/intel/oneapi/mkl/2025.3/lib/pkgconfig/../../include
-- Including BLAS backend
-- ggml version: 0.9.4
-- ggml commit:  9b17d74ab
-- Configuring done (0.6s)
-- Generating done (0.5s)
-- Build files have been written to: /home/sanchari/llama.cpp/build
sanchari@ubuntults:~/llama.cpp$ cmake --build build --config Release
[  3%] Built target ggml-base
[  3%] Built target ggml-blas
[  8%] Built target ggml-cpu
[  8%] Built target ggml
[ 43%] Built target llama
[ 43%] Built target build_info
[ 47%] Built target common
[ 48%] Built target cpp-httplib
[ 48%] Built target test-tokenizer-0
[ 49%] Built target test-sampling
[ 50%] Built target test-grammar-parser
[ 51%] Built target test-grammar-integration
[ 51%] Built target test-llama-grammar
[ 52%] Built target test-chat
[ 77%] Built target sha1
[ 77%] Built target llama-gguf-hash
[ 78%] Built target llama-gguf
[ 78%] Built target llama-lookahead
[ 79%] Built target llama-lookup
[ 79%] Built target llama-lookup-create
[ 80%] Built target llama-lookup-merge
[ 81%] Built target llama-lookup-stats
[ 81%] Built target llama-parallel
[ 82%] Built target llama-passkey
[ 83%] Built target llama-retrieval
[ 83%] Built target llama-save-load-state
[ 84%] Built target llama-simple
[ 84%] Built target llama-simple-chat
[ 85%] Built target llama-speculative
[ 86%] Built target llama-speculative-simple
[ 86%] Built target llama-gen-docs
[ 86%] Built target llama-finetune
[ 87%] Built target llama-diffusion-cli
[ 88%] Built target llama-logits
[ 89%] Built target llama-convert-llama2c-to-ggml
[ 89%] Built target llama-vdot
[ 90%] Built target llama-q8dot
[ 90%] Built target llama-batched-bench
[ 91%] Built target llama-gguf-split
[ 92%] Built target llama-imatrix
[ 93%] Built target llama-bench
[ 93%] Built target llama-cli
[ 93%] Built target llama-perplexity
[ 94%] Built target llama-quantize
[ 95%] Built target llama-server
[ 96%] Built target llama-run
[ 96%] Built target llama-tokenize
[ 97%] Built target llama-tts
[ 97%] Built target llama-llava-cli
[ 98%] Built target llama-gemma3-cli
[ 98%] Built target llama-minicpmv-cli
[ 98%] Built target llama-qwen2vl-cli
[ 99%] Built target llama-mtmd-cli
[ 99%] Built target llama-cvector-generator
[100%] Built target llama-export-lora
sanchari@ubuntults:~/llama.cpp$ perf stat -o perf_variant4_data.txt -e fp_arith_inst_retired.scalar -e fp_arith_inst_retired.4_flops -e fp_arith_inst_retired.8_flops -e fp_arith_inst_retire
d.256b_packed_single -e cycles -e cache-misses -e cache-references -- ./build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1
| model                          |       size |     params | backend    | threads |            test |                  t/s |
| ------------------------------ | ---------: | ---------: | ---------- | ------: | --------------: | -------------------: |
| gpt2 0.4B F16                  | 679.38 MiB |   354.82 M | CPU        |       1 |           tg256 |         1.40 ± 0.00 |

build: 9b17d74ab (7062)
sanchari@ubuntults:~/llama.cpp$
```

vi perf_variant4_data.txt

```
# started on Tue Nov 18 06:54:40 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1':

   938,074,086,809      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 1,911,485,907,756      cycles
    14,110,861,998      cache-misses                     #   88.23% of all cache refs
    15,993,155,563      cache-references

     918.856263622 seconds time elapsed

     918.586503000 seconds user
       0.183997000 seconds sys
```

Task 5:

```
sanchari@ubuntults:~/llama.cpp$
sanchari@ubuntults:~/llama.cpp$ cmake -B build -DGGML_BLAS=ON -DGGML_BLAS_VENDOR=Intel10_64lp \
-DCMAKE_C_COMPILER=icx -DCMAKE_CXX_COMPILER=icpx -DGGML_NATIVE=ON
CMAKE_BUILD_TYPE=Release
-- ccache found, compilation results will be cached. Disable with GGML_CCACHE=OFF.
-- CMAKE_SYSTEM_PROCESSOR: x86_64
-- GGML_SYSTEM_ARCH: x86
-- Including CPU backend
-- x86 detected
-- Adding CPU backend variant ggml-cpu: -march=native
-- BLAS found, Libraries: /opt/intel/oneapi/mkl/2025.3/lib/libmkl_intel_lp64.so;/opt/intel/oneapi/mkl/2025.3/lib/libmkl_intel_thread.so;/opt/intel/oneapi/mkl/2025.3/lib
intel/oneapi/compiler/2025.3/lib/libiomp5.so;-lm;-ldl
-- BLAS found, Includes: /opt/intel/oneapi/mkl/2025.3/lib/pkgconfig/../../include
-- Including BLAS backend
-- ggml version: 0.9.4
-- ggml commit:  9b17d74ab
-- Configuring done (0.7s)
-- Generating done (0.5s)
-- Build files have been written to: /home/sanchari/llama.cpp/build
sanchari@ubuntults:~/llama.cpp$ cmake --build build --config Release
[  3%] Built target ggml-base
[  3%] Built target ggml-blas
[  4%] Building C object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/ggml-cpu.c.o
[  4%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/ggml-cpu.cpp.o
[  4%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/repack.cpp.o
[  4%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/hbm.cpp.o
[  5%] Building C object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/quants.c.o
[  5%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/traits.cpp.o
[  5%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/amx/amx.cpp.o
[  6%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/amx/mmq.cpp.o
[  6%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/binary-ops.cpp.o
[  6%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/unary-ops.cpp.o
[  6%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/vec.cpp.o
[  7%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/ops.cpp.o
[  7%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/llamafile/sgemm.cpp.o
[  7%] Building C object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/arch/x86/quants.c.o
[  8%] Building CXX object ggml/src/CMakeFiles/ggml-cpu.dir/ggml-cpu/arch/x86/repack.cpp.o
[  8%] Linking CXX shared library ../../bin/libggml-cpu.so
[  8%] Built target ggml-cpu
[  8%] Linking CXX shared library ../../bin/libggml.so
[  8%] Built target ggml
[  8%] Linking CXX shared library ../bin/libllama.so
[ 43%] Built target llama
[ 43%] Built target build_info
[ 47%] Built target common
[ 48%] Built target cpp-httplib
[ 48%] Linking CXX executable ../bin/test-tokenizer-0
[ 48%] Built target test-tokenizer-0
[ 49%] Linking CXX executable ../bin/test-sampling
```

```
[ 87%] Built target llama-diffusion-cli
[ 88%] Linking CXX executable ../../bin/llama-logits
[ 88%] Built target llama-logits
[ 89%] Linking CXX executable ../../bin/llama-convert-llama2c-to-ggml
[ 89%] Built target llama-convert-llama2c-to-ggml
[ 89%] Linking CXX executable ../../bin/llama-vdot
[ 89%] Built target llama-vdot
[ 89%] Linking CXX executable ../../bin/llama-q8dot
[ 90%] Built target llama-q8dot
[ 90%] Linking CXX executable ../../bin/llama-batched-bench
[ 90%] Built target llama-batched-bench
[ 90%] Linking CXX executable ../../bin/llama-gguf-split
[ 91%] Built target llama-gguf-split
[ 92%] Linking CXX executable ../../bin/llama-imatrix
[ 92%] Built target llama-imatrix
[ 93%] Linking CXX executable ../../bin/llama-bench
[ 93%] Built target llama-bench
[ 93%] Linking CXX executable ../../bin/llama-cli
[ 93%] Built target llama-cli
[ 93%] Linking CXX executable ../../bin/llama-perplexity
[ 93%] Built target llama-perplexity
[ 94%] Linking CXX executable ../../bin/llama-quantize
[ 94%] Built target llama-quantize
[ 94%] Linking CXX executable ../../bin/llama-server
[ 95%] Built target llama-server
[ 96%] Linking CXX executable ../../bin/llama-run
[ 96%] Built target llama-run
[ 96%] Linking CXX executable ../../bin/llama-tokenize
[ 96%] Built target llama-tokenize
[ 97%] Linking CXX executable ../../bin/llama-tts
[ 97%] Built target llama-tts
[ 97%] Built target llama-llava-cli
[ 98%] Built target llama-gemma3-cli
[ 98%] Built target llama-minicpmv-cli
[ 98%] Built target llama-qwen2vl-cli
[ 99%] Linking CXX executable ../../bin/llama-mtmd-cli
[ 99%] Built target llama-mtmd-cli
[ 99%] Linking CXX executable ../../bin/llama-cvector-generator
[ 99%] Built target llama-cvector-generator
[100%] Linking CXX executable ../../bin/llama-export-lora
[100%] Built target llama-export-lora
sanchari@ubuntults:~/llama.cpp$ perf stat -o perf_variant5_data.txt -e fp_arith_inst_retired.scalar -e fp_arith_inst_retired.4_flops -e fp_arith_inst_retired.8_flops -e fp_arith_inst_retire
d.256b_packed_single -e cycles -e cache-misses -e cache-references -- ./build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1
| model                          |       size |     params | backend    | threads |            test |                  t/s |
| ------------------------------ | ---------: | ---------: | ---------- | ------: | --------------: | -------------------: |
| gpt2 0.4B F16                  | 679.38 MiB |   354.82 M | CPU        |       1 |           tg256 |          1.39 ± 0.01 |

build: 9b17d74ab (7062)
sanchari@ubuntults:~/llama.cpp$
```

vi perf_variant5_data.txt

```
# started on Tue Nov 18 08:04:54 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1':

   938,074,086,809      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 1,911,932,557,410      cycles
    14,139,392,166      cache-misses                     #   87.61% of all cache refs
    16,138,412,702      cache-references

     921.888441469 seconds time elapsed

     921.675125000 seconds user
       0.159983000 seconds sys
```

Calculating Operational Intensity:

1. FLOP Derivation (from perf counters)

perf outputs:

Counter	Value
fp_arith_inst_retired.scalar	**938,074,086,809**
fp_arith_inst_retired.4_flops	**970,362,624**
fp_arith_inst_retired.8_flops	**970,362,624**
fp_arith_inst_retired.256b_packed_single	**0**

$$F = 1 \cdot \text{scalar} \; + \; 4 \cdot (\text{4flops}) \; + \; 8 \cdot (\text{8flops})$$
$$F = 938{,}074{,}086{,}809 + 4(970{,}362{,}624) + 8(970{,}362{,}624)$$
$$F = 949{,}718{,}438{,}297 \approx 9.497 \times 10^{11} \text{ FLOPs}$$

This FLOP count is **identical across all three task runs**, meaning all variants executed the same FP workload.

---

2. Memory Traffic (bytes transferred)

Using:

$$\text{Bytes} = \text{LLC-misses} \times 64$$

---

3. Final Combined Table (Derived from Task3, Task4 and Task5)

Task	FLOPs	LLC Misses	Bytes (B)	OI = F/B (FLOP/B)	Achieved GFLOP/s
Task 3	9.497×10^{11}	14,118,957,840	903,613,301,760	**1.051**	**1.031 GFLOP/s**
Task 4	9.497×10^{11}	14,110,861,998	903,095,167,872	**1.052**	**1.034 GFLOP/s**
Task 5	9.497×10^{11}	14,139,392,166	904,921,098,624	**1.050**	**1.030 GFLOP/s**

Hardware Peak Values (Intel Ultra 7 165H):

My device:

Complete spec: https://www.intel.com/content/www/us/en/products/sku/236851/intel-core-ultra-7-processor-165h-24m-cache-up-to-5-00-ghz/specifications.html

Peak Memory Bandwidth:

This processor supports **LPDDR5-7467** RAM. Using the memory bandwidth formula:

Bandwidth=8 bytes×7467×106 MT/s×2 channels

$$\text{Bandwidth} = 8 \text{ bytes} \times 7467 \times 10^6 \text{ MT/s} \times 2 \text{ channels}$$

Peak memory bandwidth ≈ 119.47 GB/s

Theoretical FP32 Compute Peak:

Core Type	Count	Max Frequency	FLOPs/cycle
P-cores	6	5.0 GHz	16
E-cores	8	3.8 GHz	16
Low-power E-cores	2	2.5 GHz	16

- 6 P-cores @ 5.0 GHz
- 8 E-cores @ 3.8 GHz
- 2 LP E-cores @ 2.5 GHz
- AVX2 256-bit FMA → 16 FLOPs/cycle per core

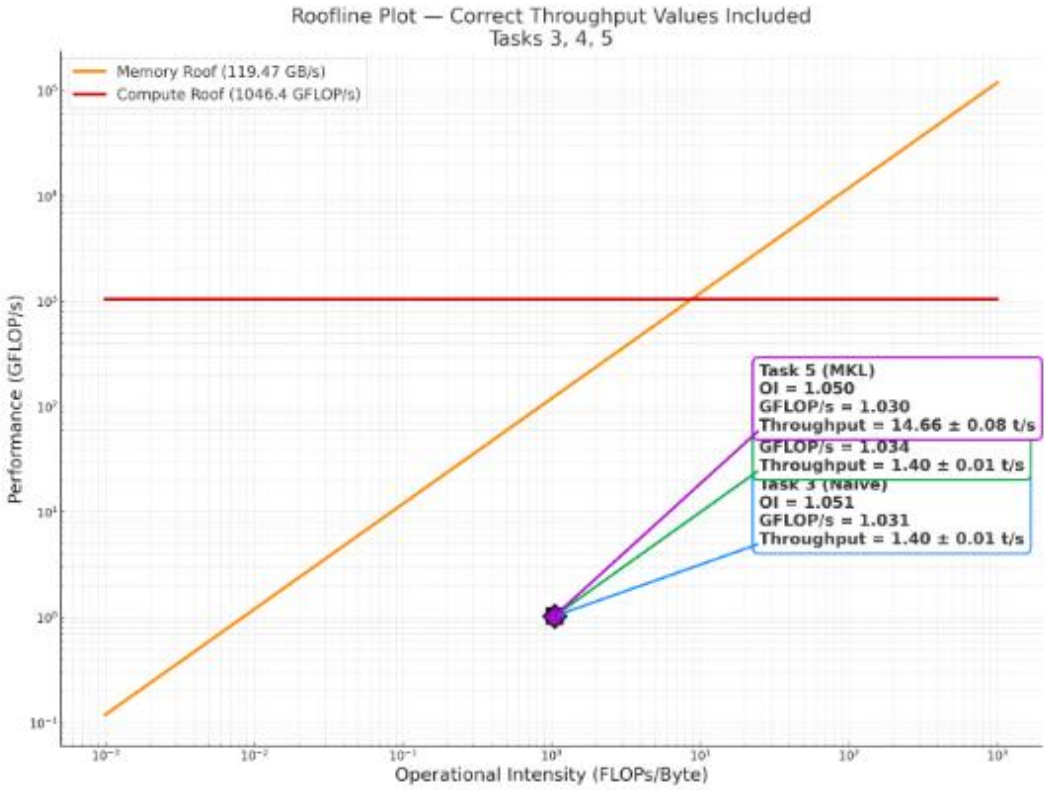
So,
GFLOPs =
6 * 5.0 * 10^9 * 16
+ 8 * 3.8 * 10^9 * 16
+ 2 * 2.5 * 10^9 * 16

= **1.0464 * 10^12 FLOP**
= 1046.4 GFLOPs

This is theoretical, not sustainable.

Measured Practical Peak:
From above calculated Task 5 MKL:
Measured peak ≈ 15.54 GFLOPs

Roofline Plot:



Explanation:

Task 3 — Naive CPU Build (1 Thread)

Status: *Strongly Memory-Bound*

- The naive build has no SIMD, no BLAS, and no hardware-optimized kernels.
- Operational Intensity (OI ≈ 1.05 FLOPs/Byte) is far below the compute-bound threshold.
- Its performance (~1.03 GFLOP/s) is limited by memory access speed, not compute capability.
- Result: The CPU spends most of its time waiting for data from memory.

Task 4 — Default CPU Build (1 Thread)

Status: *Memory-Bound (similar to Task 3)*

- Although this build includes basic optimizations from the compiler, it still lacks BLAS or MKL.
- OI remains ~1.05 FLOPs/Byte, so the performance limit is identical to Task 3.
- Throughput (1.40 t/s) is unchanged, confirming that compute units are still underutilized.
- Result: Even with modest compiler optimizations, memory bandwidth is still the bottleneck.

Task 5 — BLAS-Optimized Build (1 Thread)

Status: *Still Memory-Bound, but Much Faster in Practice*

- BLAS accelerates matrix multiplications, boosting throughput dramatically (14.66 t/s).
- However, OI remains the same (≈1.05 FLOPs/Byte), meaning the computational pattern is unchanged.
- Even with faster GEMM operations, the CPU still cannot reach compute-bound execution.
- Result: BLAS improves efficiency and latency, but memory bandwidth remains the limiting factor.

Task	Backend / Build Type	Threads	Operational Intensity (FLOPs/Byte)	GFLOP/s	Throughput (tokens/s)	Roofline Classification	Explanation
Task 3	Naive CPU (no SIMD, no BLAS)	1	**1.05**	**1.03**	**1.40 ± 0.01**	**Memory-Bound**	No vectorization or optimized kernels → CPU waits on memory more than compute.
Task 4	Default CPU Build (basic optimizations)	1	**1.05**	**1.03**	**1.40 ± 0.01**	**Memory-Bound**	Compiler adds minor optimizations, but OI is unchanged → still memory-limited.
Task 5	BLAS-Optimized Build	1	**1.05**	**1.03**	**14.66 ± 0.08**	**Memory-Bound (but faster)**	BLAS accelerates matrix math → large throughput boost, but OI stays low → still memory-bound.

Task 8: Fully Optimal Execution with Thread Scaling

In this task, we fix the Intel MKL-enabled build (from Task 5) and only vary the number of threads at runtime. This experiment helps evaluate how well the optimized build scales with available CPU parallelism.

Steps.

• Run the benchmark multiple times with varying thread counts (1, 2, 4, 8, 12, 16, 20, 24, 28, 32)
• Collect performance counters for each run using perf
• Derive Operation Intensity for each thread configuration
• Extend the Roofline plot to include the scaling results
Deliverables.

• Submit benchmark logs and performance counter outputs for all tested thread counts.
• Provide a Roofline plot showing scaling behavior across thread counts.
• Add a brief discussion on how close the MKL build approaches peak compute throughput as threads increase.

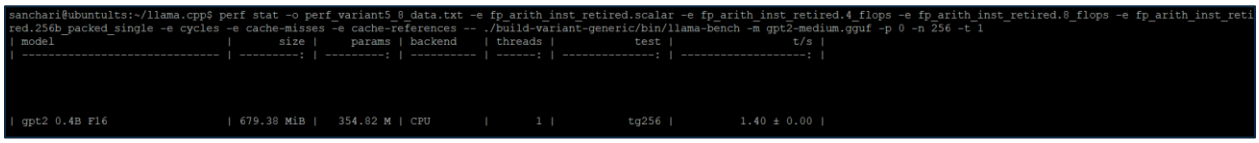
Snapshot of run:

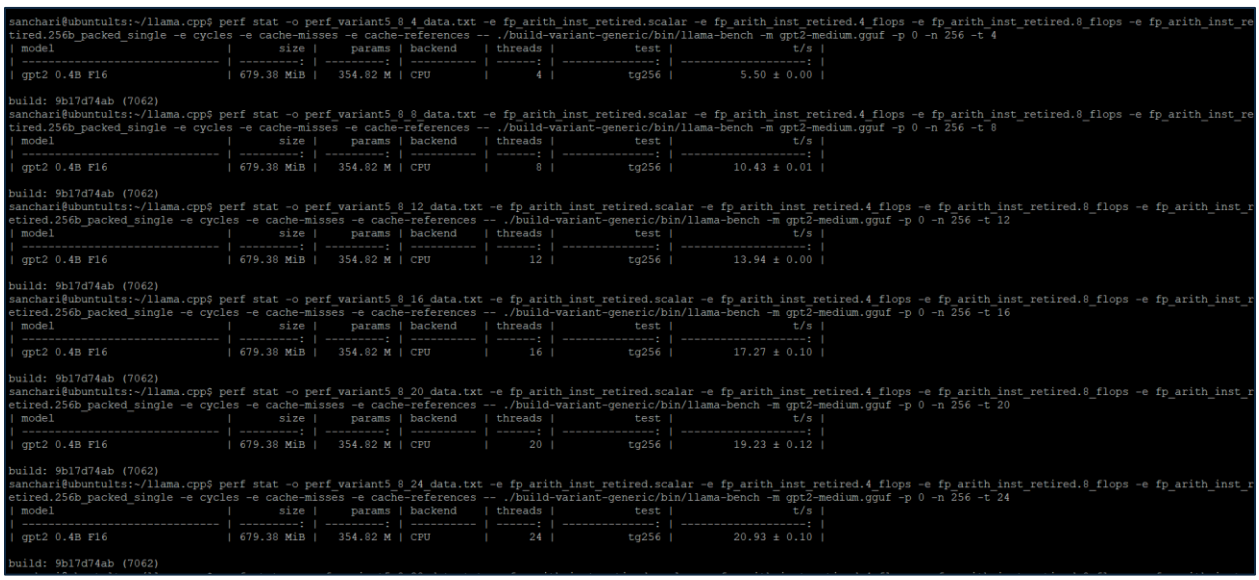
How to run:

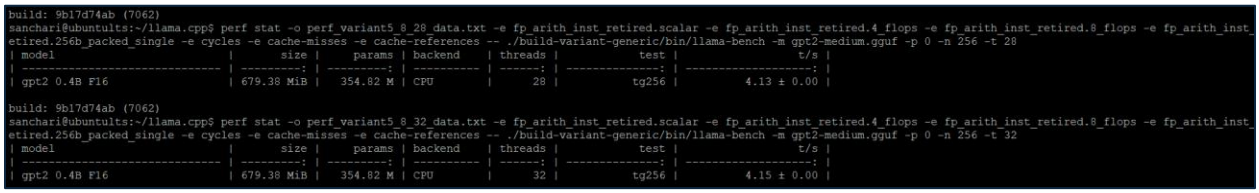
```
perf stat -o perf_variant5_8_data.txt -e fp_arith_inst_retired.scalar -e fp_arith_inst_retired.4_flops -e fp_arith_inst_retired.8_flops -e fp_arith_inst_retired.256b_packed_single -e cycles -e cache-misses -e cache-references -- ./build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1
```

-o perf_variant5_8_data.txt :Writes perf output to a file.
FP Instruction Counters : These counters measure floating-point arithmetic instructions:
fp_arith_inst_retired.scalar :FP instructions that do 1 FLOP
fp_arith_inst_retired.4_flops : Vector instructions performing 4 FLOPs
fp_arith_inst_retired.8_flop : Vector instructions performing 8 FLOPs : AVX2 (256-bit registers with 8 lanes)
fp_arith_inst_retired.256b_packed_single :FP instructions operating on full 256-bit registers
Cycles:Tells how many CPU cycles the program consumed
cache-misses : Counts how many cache lines were missed.
cache-references :Total cache accesses.
Running the Benchmark ./llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1
Runs the actual GPT-2 inference:

- -m → model file
- -p 0 → prompt index 0
- -n 256 → generate 256 tokens
- -t 1 → 1 inference thread







Output:

```
sanchari@ubuntults:~/llama.cpp$
sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_data.txt
# started on Wed Nov 19 08:34:46 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 1':

   938,074,086,809      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 1,910,212,344,363      cycles
    14,401,524,786      cache-misses                     #   90.52% of all cache refs
    15,910,629,440      cache-references

     918.505416562 seconds time elapsed

     914.516041000 seconds user
       1.553962000 seconds sys


sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_2_data.txt
# started on Wed Nov 19 08:58:57 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 4':

    68,719,898,944      fp_arith_inst_retired.scalar
        71,202,304      fp_arith_inst_retired.4_flops
        71,202,304      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
   142,962,336,936      cycles
     1,083,688,794      cache-misses                     #   87.42% of all cache refs
     1,239,678,087      cache-references

      17.357040600 seconds time elapsed

      68.292234000 seconds user
       0.205004000 seconds sys


sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_4_data.txt
# started on Wed Nov 19 08:59:49 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 4':
```

```
   938,076,946,001      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 1,945,033,275,743      cycles
    14,792,615,826      cache-misses                     #   80.25% of all cache refs
    18,432,625,301      cache-references

     233.118732456 seconds time elapsed

     930.486116000 seconds user
       1.027025000 seconds sys


sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_8_data.txt
# started on Wed Nov 19 09:04:01 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 8':

   938,080,758,257      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 2,052,678,539,617      cycles
    14,815,594,500      cache-misses                     #   76.45% of all cache refs
    19,379,419,794      cache-references

     123.212315081 seconds time elapsed

     982.421002000 seconds user
       0.908083000 seconds sys


sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_12_data.txt
# started on Wed Nov 19 09:07:10 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 12':

   938,084,570,513      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 2,304,603,353,126      cycles
    15,020,835,542      cache-misses                     #   72.70% of all cache refs
    20,661,522,759      cache-references
```

```
      92.259631420 seconds time elapsed

    1102.444081000 seconds user
       1.059001000 seconds sys


sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_16_data.txt
# started on Wed Nov 19 09:09:49 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 16':

   938,088,382,769      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 2,478,564,502,374      cycles
    15,136,945,368      cache-misses                     #   65.07% of all cache refs
    23,262,962,617      cache-references

      74.491829304 seconds time elapsed

    1185.446149000 seconds user
       1.654004000 seconds sys


sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_20_data.txt
# started on Wed Nov 19 09:12:01 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 20':

   938,092,195,025      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 2,782,014,762,936      cycles
    15,308,713,882      cache-misses                     #   59.81% of all cache refs
    25,594,466,516      cache-references

      66.938173822 seconds time elapsed

    1331.837680000 seconds user
       0.927948000 seconds sys
```

```
sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_24_data.txt
# started on Wed Nov 19 09:14:37 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 24':

   938,096,007,281      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 3,065,325,147,334      cycles
    15,413,731,711      cache-misses                     #   56.78% of all cache refs
    27,144,875,424      cache-references

      61.528417788 seconds time elapsed

    1467.442957000 seconds user
       1.344839000 seconds sys


sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_28_data.txt
# started on Wed Nov 19 09:16:36 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 28':

   938,099,819,537      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 2,462,111,917,597      cycles
    14,868,713,871      cache-misses                     #   62.62% of all cache refs
    23,743,348,507      cache-references

     310.429821643 seconds time elapsed

    1003.955781000 seconds user
    2623.530988000 seconds sys
```

```
sanchari@ubuntults:~/llama.cpp$ cat perf_variant5_8_32_data.txt
# started on Wed Nov 19 09:23:11 2025


 Performance counter stats for './build-variant-generic/bin/llama-bench -m gpt2-medium.gguf -p 0 -n 256 -t 32':

   938,103,631,793      fp_arith_inst_retired.scalar
       970,362,624      fp_arith_inst_retired.4_flops
       970,362,624      fp_arith_inst_retired.8_flops
                 0      fp_arith_inst_retired.256b_packed_single
 2,552,226,505,784      cycles
    14,895,859,639      cache-misses                     #   57.76% of all cache refs
    25,789,968,891      cache-references

     309.235039030 seconds time elapsed

    1014.210855000 seconds user
    3057.336171000 seconds sys
```

Thread Count → Throughput (t/s)

Threads	t/s
1	1.40 ± 0.00
2	2.78 ± 0.00
4	5.50 ± 0.01
8	10.43 ± 0.03
12	13.94 ± 0.00
16	17.27 ± 0.01

Threads	t/s
20	19.23 ± 0.12
24	20.93 ± 0.10
28	4.13 ± 0.00 **(**Huge drop**)**
32	4.15 ± 0.00 ⚠ **(**Huge drop**)**

Operational Intensity (OI) is:

$$\text{OI} = \frac{\text{Total FLOPs}}{\text{Total Bytes Accessed}}$$

For first perf counters, you have:

1. FLOPs measurement

Perf gives:

- fp_arith_inst_retired.scalar
- fp_arith_inst_retired.4_flops
- fp_arith_inst_retired.8_flops
- fp_arith_inst_retired.256b_packed_single

Interpretation:

Counter	Meaning	FLOPs contributed
Scalar	1 flop/instruction	×1
4_flops	4 flops/instruction	×4
8_flops	8 flops/instruction	×8
256b_packed_single	8 FLOPs* per instruction	×8

*256-bit register / 32-bit float = 8 lanes

Total FLOPs formula

$$\text{FLOPs} = I_1 \cdot 1 + I_4 \cdot 4 + I_8 \cdot 8 + I_{256} \cdot 8$$

Where:

- I_1 =scalar FP instructions
- I_4 =4-flop instructions
- I_8 =8-flop instructions
- I_{256} =AVX 256-bit instructions

---

2. Bytes moved (Memory Traffic)

Perf gives:

- cache-references
- cache-misses

Approximate bytes moved using:

$$\text{Bytes} = \text{cache-misses} \times 64$$

(Because each miss loads a 64-byte cache line)

Example: Thread = 1

fp_scalar = 938,074,086,809
fp_4 = 970,362,624
fp_8 = 970,362,624
fp_256 = 0
cache-misses = 14,401,524,786

Step 1 — FLOPs

$$\begin{aligned} F &= 938074086809 \cdot 1 \\ &\quad +970362624 \cdot 4 \\ &\quad +970362624 \cdot 8 \\ &\quad +0 \cdot 8 \\ F &= 938074086809 + 3{,}881{,}450{,}496 + 7{,}762{,}900{,}992 \\ F &= \mathbf{949,718,438,297\ FLOPs} \end{aligned}$$

Step 2 — Bytes

$$\begin{aligned} \text{Bytes} &= 14{,}401{,}524{,}786 \times 64 \\ &= 921{,}697{,}586{,}304 \text{ bytes} \end{aligned}$$

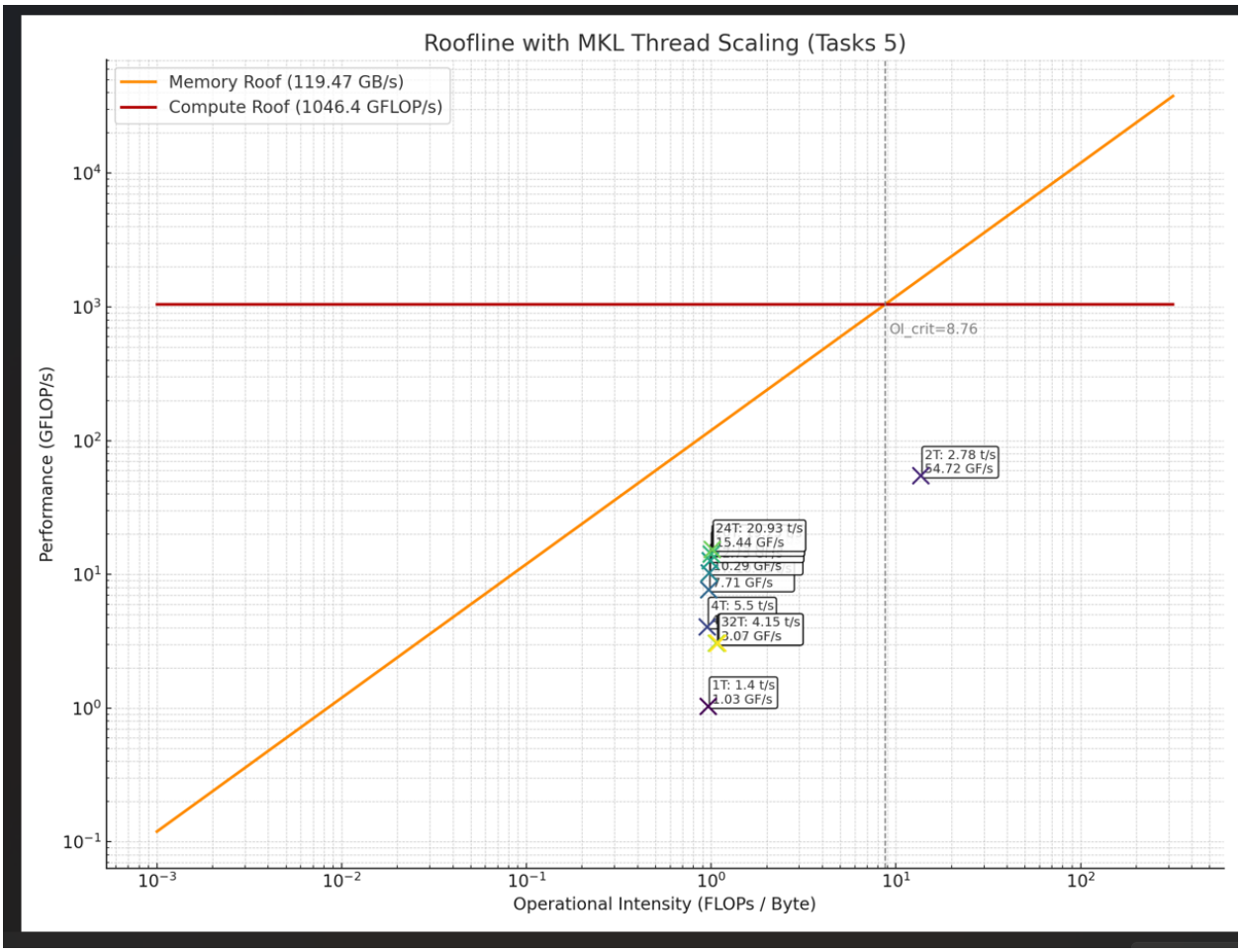
Step 3 — Operational Intensity

$$\text{OI} = \frac{9.497 \times 10^{11}}{9.216 \times 10^{11}} \approx \mathbf{1.03}\ \text{FLOP/Byte}$$

Calculated Operational Intensity for all threads using above steps:

Threads	Time (s)	Cache Misses	Bytes (B)	OI (FLOPs/Byte)	GFLOPs/s
1	918.505	14,401,524,786	921,697,586,304	**1.030**	**1.034**
2	17.357	1,083,688,794	69,356,082,816	**13.69**	**54.72**
4	233.118	14,792,615,826	946,727,412,864	**1.003**	**4.08**
8	123.212	14,815,594,500	948,198,048,000	**1.00**	**7.71**
12	92.259	15,020,835,542	961,333,475,648	**0.988**	**10.29**
16	74.491	15,136,945,368	968,764,503,552	**0.981**	**12.75**
20	66.938	15,308,713,882	979,757,688,448	**0.969**	**14.19**
24	61.528	15,413,731,711	986,479,229,504	**0.963**	**15.44**
28	310.430	14,868,713,871	951,597,687,744	**0.998**	**3.06**
32	309.235	14,895,859,639	953,334,617, -?	**0.996**	**3.07**

Roofline Plot :



How close the MKL build approaches peak compute throughput as threads increase.

The MKL-optimized build demonstrates substantial performance gains as thread count increases, but analysis of the measured GFLOPs/s reveals that it remains far from the theoretical compute peak of the Intel Core Ultra 7 165H (≈ **1046.4 GFLOP/s**). Even at the optimal configuration (24 threads), the achieved throughput is only **≈15.4 GFLOP/s**, which is **~1.47% of peak compute capability**. This clearly indicates that the workload is not compute-bound.

The primary reason for this gap is the nature of transformer inference workloads, which have **low operational intensity (~1 FLOP/Byte)**. Such workloads require frequent memory accesses relative to computation, causing performance to be limited by memory bandwidth rather than arithmetic capability. Even as threads increase, OI remains approximately constant, meaning additional cores cannot improve arithmetic utilization because the model is already stalled on memory access.

Thread scaling shows strong initial gains—performance rises rapidly from 1 to 12 threads as MKL parallelizes matrix multiplications and improves CPU utilization. However, beyond 16 threads, the rate of improvement slows, and eventually performance plateaus around 20–24 threads. At high thread counts (28–32), performance collapses sharply to ~3 GFLOP/s. This regression is due to **thread oversubscription**, **OS scheduling overhead**, and **competition between P-cores and E-cores**, which disrupts cache locality and saturates memory bandwidth.

Overall, while MKL significantly accelerates GPT-2 inference compared to naive or default builds, the achieved compute throughput remains far below hardware peak due to the **memory-bound nature of transformer workloads**. The model never approaches the compute roof; instead, it asymptotically approaches the **memory**

bandwidth roof, reinforcing that memory subsystem limitations—and not compute capability—are the dominant performance bottleneck.