Docker

1、Prerequisites（前提要求或者先决条件）

- Docker must be installed and running on your system.
- 在你的系统上Docker必须被安装和运行。
- Create a folder to store big models & intermediate files (ex. /llama/models)
- 创建一个folder用来存储大的models和一些其他的中间文件，例如llama/models

2、Images（镜像）

We have three Docker images available for this project（对于这个项目我们有三个docker镜像是可以使用的）:

1. `ghcr.io/ggerganov/llama.cpp:full`: This image includes both the main executable file and the tools to convert LLaMA models into ggml and convert into 4-bit quantization. (platforms: `linux/amd64`, `linux/arm64`)
    - 这个镜像不仅包含main executable file而且包含将LLaMA模型转化成ggml、将LLaMA转化成4-bit量化相关的工具
2. `ghcr.io/ggerganov/llama.cpp:light`: This image only includes the main executable file. (platforms: `linux/amd64`, `linux/arm64`)
    - 这个镜像仅仅包含main executable file
    - 平台可以是linux/amd64、linux/arm64
3. `ghcr.io/ggerganov/llama.cpp:server`: This image only includes the server executable file. (platforms: `linux/amd64`, `linux/arm64`)
    - 这个镜像仅包含server executable file
    - 平台可以是linux/amd64、linux/arm64

Additionally, there the following images, similar to the above(另外，下列这些镜像同上述是相类似的):

- `ghcr.io/ggerganov/llama.cpp:full-cuda`: Same as `full` but compiled with CUDA support. (platforms: `linux/amd64`)
- `ghcr.io/ggerganov/llama.cpp:light-cuda`: Same as `light` but compiled with CUDA support. (platforms: `linux/amd64`)
- `ghcr.io/ggerganov/llama.cpp:server-cuda`: Same as `server` but compiled with CUDA support. (platforms: `linux/amd64`)
- `ghcr.io/ggerganov/llama.cpp:full-rocm`: Same as `full` but compiled with ROCm support. (platforms: `linux/amd64`, `linux/arm64`)
- `ghcr.io/ggerganov/llama.cpp:light-rocm`: Same as `light` but compiled with ROCm support. (platforms: `linux/amd64`, `linux/arm64`)
- `ghcr.io/ggerganov/llama.cpp:server-rocm`: Same as `server` but compiled with ROCm support. (platforms: `linux/amd64`, `linux/arm64`)

The GPU enabled images are not currently tested by CI beyond being built. They are not built with any variation from the ones in the Dockerfiles defined in .devops/ and the GitHub Action defined in .github/workflows/docker.yml. If you need different settings (for example, a different CUDA or ROCm library, you'll need to build the images locally for now).

- 解释：目前，除了构建之外，CI 尚未对启用 GPU 的映像进行测试。它们与 .devops/ 中定义的 Dockerfile 和 .github/workflows/docker.yml 中定义的 GitHub Action 没有任何不同。如果您需要不同的设置（例如，不同的 CUDA 或 ROCm 库），您目前需要在本地构建映像。

3、Usage（使用）

The easiest way to download the models, convert them to ggml and optimize them is with the --all-in-one command which includes the full docker image.（最简单的方式就是下载models，将这些models转化为ggml并且使用--all-in-one命令进行量化，这个--all-in-one命令将会包含整个docker镜像）

Replace `/path/to/models` below with the actual path where you downloaded the models.（使用你下载的models实际路径替换/path/to/models）

```
# docker是程序名称
docker run -v /path/to/models:/models ghcr.io/ggerganov/llama.cpp:full --all-in-one "/models/" 7B
```

On completion, you are ready to play!

```
docker run -v /path/to/models:/models ghcr.io/ggerganov/llama.cpp:full --run -m /models/7B/ggml-model-q4_0.gguf -p "Building a website can be done in 10 simple steps:" -n 512
```

or with a light image:

```
docker run -v /path/to/models:/models ghcr.io/ggerganov/llama.cpp:light -m /models/7B/ggml-model-q4_0.gguf -p "Building a website can be done in 10 simple steps:" -n 512
```

or with a server image:

```
docker run -v /path/to/models:/models -p 8000:8000 ghcr.io/ggerganov/llama.cpp:server -m /models/7B/ggml-model-q4_0.gguf --port 8000 --host 0.0.0.0 -n 512
```

4、Docker With CUDA

Assuming one has the nvidia-container-toolkit properly installed on Linux, or is using a GPU enabled cloud, `cuBLAS` should be accessible inside the container.（假设已经在 Linux 上正确安装了 nvidia-container-toolkit，或者正在使用支持 GPU 的云，则应该可以在容器内访问 `cuBLAS`。）

5、Building Docker locally（在本地构建docker）

```
docker build -t local/llama.cpp:full-cuda -f .devops/full-cuda.Dockerfile .
docker build -t local/llama.cpp:light-cuda -f .devops/llama-cli-cuda.Dockerfile .
docker build -t local/llama.cpp:server-cuda -f .devops/llama-server-
cuda.Dockerfile .
```

You may want to pass in some different `ARGS`, depending on the CUDA environment supported by your container host, as well as the GPU architecture.（您可能需要传递一些不同的“ARGS”，具体取决于您的容器主机支持的 CUDA 环境以及 GPU 架构。）

The defaults are:

- `CUDA_VERSION` set to `11.7.1`
- `CUDA_DOCKER_ARCH` set to `all`

The resulting images, are essentially the same as the non-CUDA images:

1. `local/llama.cpp:full-cuda`: This image includes both the main executable file and the tools to convert LLaMA models into ggml and convert into 4-bit quantization.
2. `local/llama.cpp:light-cuda`: This image only includes the main executable file.
3. `local/llama.cpp:server-cuda`: This image only includes the server executable file.

Usage

After building locally, Usage is similar to the non-CUDA examples, but you'll need to add the `--gpus` flag. You will also want to use the `--n-gpu-layers` flag.

```
docker run --gpus all -v /path/to/models:/models local/llama.cpp:full-cuda --run -
m /models/7B/ggml-model-q4_0.gguf -p "Building a website can be done in 10 simple
steps:" -n 512 --n-gpu-layers 1
docker run --gpus all -v /path/to/models:/models local/llama.cpp:light-cuda -m
/models/7B/ggml-model-q4_0.gguf -p "Building a website can be done in 10 simple
steps:" -n 512 --n-gpu-layers 1
docker run --gpus all -v /path/to/models:/models local/llama.cpp:server-cuda -m
/models/7B/ggml-model-q4_0.gguf --port 8000 --host 0.0.0.0 -n 512 --n-gpu-layers 1
```