Expanding GGML Hardware Support using the Vulkan API

Ruben Ortlam
Low-level AI Engineering and Hacking
FOSDEM

2 February 2025

Overview

1. whoami
2. llama.cpp and GGML
3. Problem
4. Solution
5. Obstacles
6. GGML community

Background

- 2022 - Computer Science, M.Sc., Otto-von-Guericke-Universität Magdeburg
- Full time software engineer since then
- Focus on C++ and Python

llama.cpp

- Started by Georgi Gerganov to run Llama models on Apple MacBooks
- At first fully focused on CPU inference
- Acts as "playground" for GGML development

GGML

- Tensor library fully written in C/C++
- Optimized for various CPUs with intrinsics or assembly
- Based on forming a compute graph and executing it with multiple threads
- Can offload parts of or the whole graph to GPUs

Early llama.cpp troubles (2023)

Prompt processing

Very slow

Text generation

Very fast

Solution

Use a GPU to speed up matrix multiplications

GPU APIs

- CUDA (Nvidia)
- ROCm (AMD)
- OneAPI (Intel)
- OpenCL
- SYCL
- Vulkan

First attempt: OpenCL

- Existing BLAS library: CLBlast by Cedric Nugteren
- Relatively simple to implement

First attempt: OpenCL

- Existing BLAS library: CLBlast by Cedric Nugteren
- Relatively simple to implement

↓

But various driver and API limitations in OpenCL

Second attempt: Vulkan

- Very compatible
- Better hardware support than OpenCL
- A lot of complexity, but much can be avoided without the graphics part
- Operate close to hardware, while being very compatible
- Small binaries

First Obstacle: Boilerplate

Example: First steps

1. Initialize a `vk::ApplicationInfo` struct

First Obstacle: Boilerplate

Example: First steps

1. Initialize a `vk::ApplicationInfo` struct
2. Pick instance extensions

First Obstacle: Boilerplate

Example: First steps

1. Initialize a `vk::ApplicationInfo` struct
2. Pick instance extensions
3. Initialize a `vk::InstanceCreateInfo` struct

First Obstacle: Boilerplate

Example: First steps

1. Initialize a `vk::ApplicationInfo` struct
2. Pick instance extensions
3. Initialize a `vk::InstanceCreateInfo` struct
4. Create instance

First Obstacle: Boilerplate

Example: First steps

1. Initialize a `vk::ApplicationInfo` struct
2. Pick instance extensions
3. Initialize a `vk::InstanceCreateInfo` struct
4. Create instance
5. Query physical devices

First Obstacle: Boilerplate

Example: First steps

1. Initialize a `vk::ApplicationInfo` struct
2. Pick instance extensions
3. Initialize a `vk::InstanceCreateInfo` struct
4. Create instance
5. Query physical devices
6. ...

First Obstacle: Boilerplate

Example: First steps

1. Initialize a `vk::ApplicationInfo` struct
2. Pick instance extensions
3. Initialize a `vk::InstanceCreateInfo` struct
4. Create instance
5. Query physical devices
6. ...

→Solution: Hide boilerplate code in simple functions

First Obstacle: Boilerplate

Some examples:

- Instance initialization
- Buffer creation
- Shader loading
- Command buffer and queue handling
- Data copying between host and device
- Shader invocation
- ...

A lot of work!

Second Obstacle: Porting kernels to GLSL

- CUDA, ROCm and SYCL allow writing device code as C++ functions
- OpenCL and Vulkan require SPIR-V device code
- Compiled from shader code written in GLSL, embedded in the application
- No pointers in (base) GLSL

Second Obstacle: Porting kernels to GLSL

- Lots of variability in hardware feature support, handled using Vulkan extensions
- 16-bit float arithmetic extension
- Cooperative Matrix extension for Tensor Core support
- Multiple shader variants needed to accommodate hardware
- Support back to AMD GCN1, Nvidia Kepler and Intel ARC

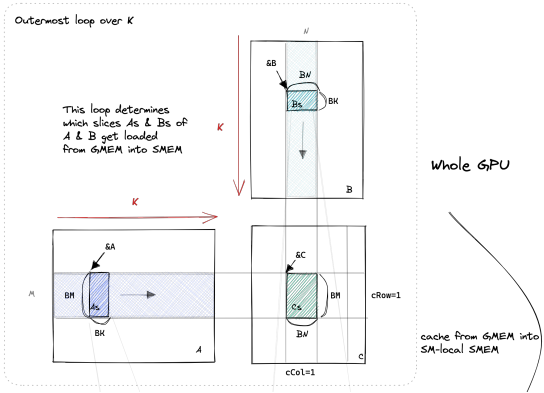
Third Obstacle: Fast Matrix Multiplication is hard

No BLAS library for Vulkan

↓

I have to do this myself

Third Obstacle: Fast Matrix Multiplication is hard



Matrix multiplication optimization on GPU, by Simon Boehm
`https://siboehm.com/articles/22/CUDA-MMM`

6 months later

It works (with some bugs)!

Driver troubles

Vulkan code is completely vendor-agnostic, right? ... right?

Testing devices

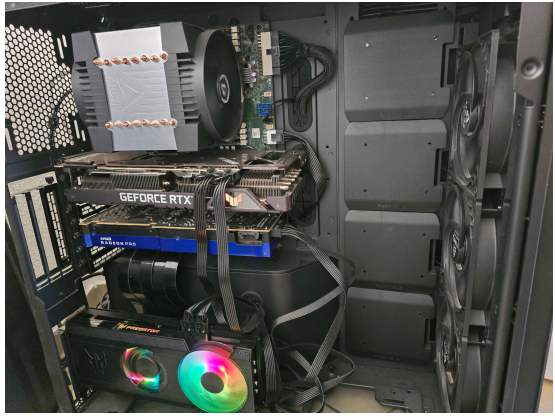
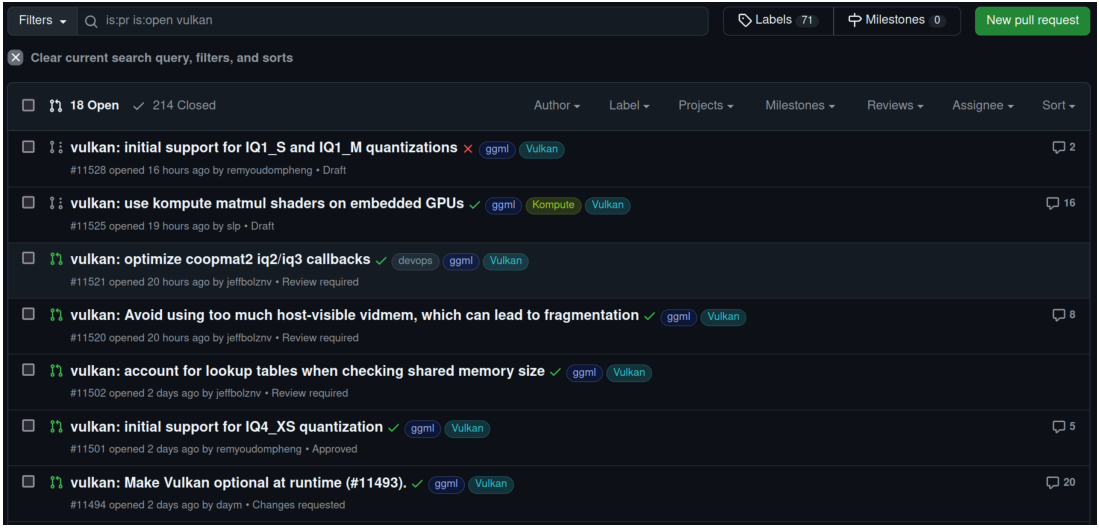


Figure: Nvidia, AMD and Intel in one server

Always more to do



Community

- All interaction happens on Github
- Communication with Issues and Discussions
- Contributions through Pull Requests
- Various kinds of contributors
  - Base team of maintainers
  - Backend maintainers
  - Various smaller contributors

Contributions welcome!

Lots of new contributions recently, for example:

- Shader optimizations and improved hardware support by Jeff Bolz (Nvidia)
- AMD GCN optimizations by netrunnereve
- Further quantization method support by remyoudompheng

Thank you

You can reach me on:

- Matrix (`@occam_razor:matrix.org`)
- Discord (`_occam`)
- Github (`https://github.com/0cc4m/`)