Rusticl

OpenCL implemented in Rust

Karol Herbst

October 5, 2022

State of OpenCL in Mesa - Clover

- Accepted in 2012 into Mesa
- Written in C++
- Targets LLVM (AMD) or NIR
- some development still happening (around 60 patches/year)
- Behaves differently compared to st/mesa
  - *PIPE_SHADER_IR_NIR_SERIALIZED*
  - Dynamic pipeloader (only user)
  - *pipe_compute_state*::*req_local_mem*
  - *pipe_grid_info*::*input* for kernel inputs
- Few users, poor support in projects using CL

Motivation

- Rust sounded like a lot of fun
- Is Rust inside Mesa feasible and sustainable?
- Is implementing existing C APIs possible in Rust?
- What are the challenges working with C bindings?
- There are actually a lot of people asking about OpenCL.

Why use Rust?

- If the compiler complains you are doing it wrong, period
- Very pleasant developing experience
- Enforces Thread and Memory Safety by design
- The threading API is a joy

Rusticl

- Merged into Mesa in Sept. 2022
- A modern OpenCL 3.0 gallium frontend written in Rust
- Entirely SPIR-V and NIR based
- Tries to be as close as possible to st/mesa
- Low API overhead
- Images are supported!
- OpenCL conformant (Intel 12th gen and radeonsi)
- Worker thread architecture, thread private pipe_context

Meson

- Nice integration into existing build system
- Rust support good enough
- No Third-party Crate support yet

Supported Drivers

- Iris
- llvmpipe
- panfrost (WIP?)
- radeonsi (WIP)
- nouveau (WIP)
- What's the minimum level of support to enable it by default?

Supported Software

- Darktable
- Gimp
- Anything not using OpenCL extensions.. probably

What is required from drivers

- Add it inside *src/gallium/targets/rusticl/meson.build*
- OpenGL Compute (*PIPE_CAP_COMPUTE*)
- *PIPE_SHADER_IR_NIR*
- some *PIPE_COMPUTE_CAPs*
  - *ADDRESS_BITS*
  - *MAX_GLOBAL_SIZE*
  - *MAX_MEM_ALLOC_SIZE*
  - *SUBGROUP_SIZE*
- *pipe_context*::*set_global_binding*
- for FULL_PROFILE
  - 128 textures
  - 64 images
  - 16 samplers
- Function calling in backend compilers

What is required from drivers Pt. 2

- *load/store_global*
- For *resource_map* (!18793)
    - *UNSYNCHRONIZED | COHERENT | PERSISTENT*
    - on UMA: *UNSYNCHRONIZED | DIRECTLY*
- !18581
    - *PIPE_CAP_SHAREABLE_SHADER*
    - *nir_intrinsic_load_kernel_input* loading from cb0 or *lower_uniforms_to_ubo*
    - *pipe_grid_info.variable_shared_mem*

Issues with Bindings

- *static inline*
- unsigned instead of enum types
- bitfield enums
- forward declared OpenCL types

Issues with other stuff

- gallium very much 32 bit specific
  - buffers can only have a 31 bit size, 2GB
  - ...
- *pipe_transfer* can't be wrapped nicely
- OpenCLs memory API
- IDE integration

TODOs

- Reduction of Kernel launch overhead !18581
- Proper implementation of *clGetKernelWorkGroupInfo*
- Improve utilization of the GPU
- Emitting less fences
- More API Validation
- More optional OpenCL features and extensions
- Function calling
- Make OpenCL "just work"
- maybe SyCL?

Q&A

- Any questions?