Vulkanised 2026

The 8th Vulkan Developer Conference
San Diego, USA | February 9-11, 2026

Vulkan Machine Learning in ggml/llama.cpp

Jeff Bolz, NVIDIA

Overview

- What are ggml and llama.cpp?
- Example workloads
- ggml-vulkan backend
- Kernel fusion optimizations
- Graph scheduling optimizations

stable-diffusion.cpp using Vulkan backend
https://github.com/leejet/stable-diffusion.cpp

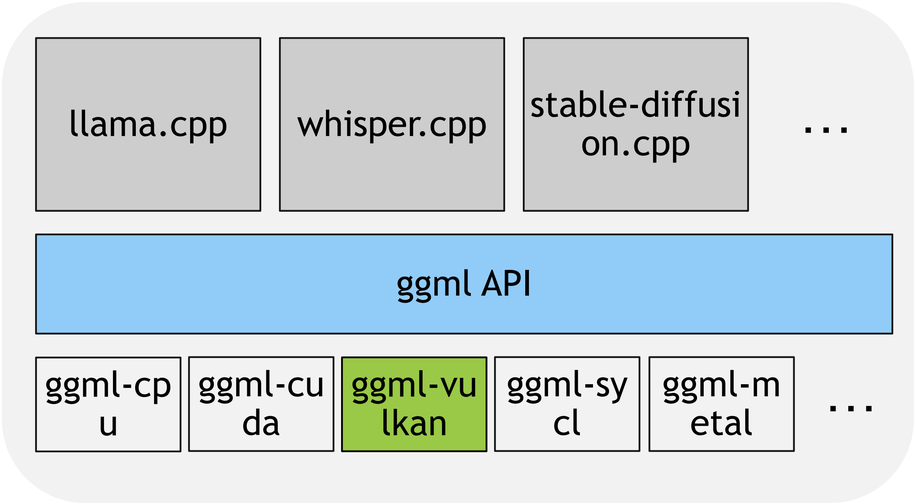
sd-cli.exe --diffusion-model qwen-image-Q8_0.gguf--vae qwen_image-vae.safetensors --qwen2vl Qwen2.5-VL-7B-Instruct-Q8_0.gguf -p "**An orange cat typing on a laptop and the laptop screen says 'Vulkan Machine Learning in ggml/llama.cpp'.**" --cfg-scale 2.5 --sampling-method euler -v --offload-to-cpu -H 512 -W 512 --diffusion-fa --flow-shift 3 --seed 2

What are ggml and llama.cpp?

- ggml is a machine learning library where you define a graph of tensor operations and then the library computes the result
  - llama.cpp implements LLMs using ggml
  - whisper.cpp implements speech to text
  - stable-diffusion.cpp implements image and video generation
  - ggml is the core library and can be integrated into other applications

- ggml has a variety of backends:
  - CPU: with various vector ISA extensions
  - GPU: Vulkan, CUDA/ROCm/MUSA, Metal, SyCL, OpenCL, WebGPU
  - NPU: CANN (Huawei), ZDNN (IBM), Hexagon (QCOM)

- Backends report which operations they support, unsupported ops fallback to the CPU implementation



Simple ggml Example

https://github.com/ggml-org/ggml/blob/master/examples/simple/simple-backend.cpp

```
// Initialize backends - create GPU and CPU backend
ggml_backend_load_all();
model.backend = ggml_backend_init_best();
model.cpu_backend = ggml_backend_init_by_type(GGML_BACKEND_DEVICE_TYPE_CPU, nullptr);

ggml_backend_t backends[2] = { model.backend, model.cpu_backend };

// Initialize sched - this will schedule the graph execution
model.sched = ggml_backend_sched_new(backends, nullptr, 2, GGML_DEFAULT_GRAPH_SIZE, false, true);

// Initialize context
size_t buf_size = ggml_tensor_overhead()*GGML_DEFAULT_GRAPH_SIZE + ggml_graph_overhead();
model.buf.resize(buf_size);

struct ggml_init_params params0 = {
    /*.mem_size   =*/ buf_size,
    /*.mem_buffer =*/ model.buf.data(),
    /*.no_alloc   =*/ true, // the tensors will be allocated later
};
struct ggml_context * ctx = ggml_init(params0);
```

Simple ggml Example

https://github.com/ggml-org/ggml/blob/master/examples/simple/simple-backend.cpp

```
// Create the graph and tensors
struct ggml_cgraph  * gf = ggml_new_graph(ctx);

model.a = ggml_new_tensor_2d(ctx, GGML_TYPE_F32, cols_A, rows_A);
model.b = ggml_new_tensor_2d(ctx, GGML_TYPE_F32, cols_B, rows_B);

struct ggml_tensor * result = ggml_mul_mat(ctx, model.a, model.b);

ggml_build_forward_expand(gf, result);

// Allocate memory for the graph
ggml_backend_sched_reset(model.sched);
ggml_backend_sched_alloc_graph(model.sched, gf);
```

Simple ggml Example

https://github.com/ggml-org/ggml/blob/master/examples/simple/simple-backend.cpp

```
// Copy input data
ggml_backend_tensor_set(model.a, matrix_A, 0, ggml_nbytes(model.a));
ggml_backend_tensor_set(model.b, matrix_B, 0, ggml_nbytes(model.b));

// Execute the graph
ggml_backend_sched_graph_compute(model.sched, gf);

// Read back the results
ggml_tensor * result = ggml_graph_node(gf, -1);
std::vector<float> out_data(ggml_nelements(result));
ggml_backend_tensor_get(result, out_data.data(), 0, ggml_nbytes(result));
```

ggml Graphs Can Be Complex

Example from gpt-oss-20b

```
node #1051 (MUL_MAT_ID):      ffn_moe_gate-18 (  45K) [Vulka        ] use=1: blk.18.ffn_gate_exps ( 134M) [Vulka        ] attn_post_norm-18 (r (  11K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1052 (    ADD_ID): ffn_moe_gate_biased- (  45K) [Vulka        ] use=1:      ffn_moe_gate-18 (  45K) [Vulka        ] blk.18.ffn_gate_exps ( 360K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1053 (MUL_MAT_ID):        ffn_moe_up-18 (  45K) [Vulka        ] use=1: blk.18.ffn_up_exps.w ( 134M) [Vulka        ] attn_post_norm-18 (r (  11K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1054 (    ADD_ID): ffn_moe_up_biased-18 (  45K) [Vulka        ] use=1:        ffn_moe_up-18 (  45K) [Vulka        ] blk.18.ffn_up_exps.b ( 360K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1055 (       GLU):  ffn_moe_weighted-18 (  45K) [Vulka        ] use=1: ffn_moe_gate_biased- (  45K) [Vulka        ] ffn_moe_up_biased-18 (  45K) [Vulka        ]
node #1056 (MUL_MAT_ID):      ffn_moe_down-18 (  45K) [Vulka        ] use=1: blk.18.ffn_down_exps ( 134M) [Vulka        ]  ffn_moe_weighted-18 (  45K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1057 (    ADD_ID): ffn_moe_down_biased- (  45K) [Vulka        ] use=1:      ffn_moe_down-18 (  45K) [Vulka        ] blk.18.ffn_down_exps ( 360K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1058 (       MUL):            node_1058 (  45K) [Vulka        ] use=4: ffn_moe_down_biased- (  45K) [Vulka        ] ffn_moe_weights_soft (   0K) [Vulka        ]
node #1063 (       ADD):            node_1063 (  11K) [Vulka        ] use=1:              (view) (  11K) [Vulka        ]     node_1058 (view) (  11K) [Vulka        ]
node #1064 (       ADD):            node_1064 (  11K) [Vulka        ] use=1:           node_1063 (  11K) [Vulka        ]     node_1058 (view) (  11K) [Vulka        ]
node #1065 (       ADD):       ffn_moe_out-18 (  11K) [Vulka        ] use=1:           node_1064 (  11K) [Vulka        ]     node_1058 (view) (  11K) [Vulka        ]
node #1066 (       ADD):             l_out-18 (  11K) [Vulka        ] use=2:      ffn_moe_out-18 (  11K) [Vulka        ]          ffn_inp-18 (  11K) [Vulka        ]
node #1067 (  RMS_NORM):              norm-19 (  11K) [Vulka        ] use=1:            l_out-18 (  11K) [Vulka        ]
node #1068 (       MUL):         attn_norm-19 (  11K) [Vulka        ] use=3:              norm-19 (  11K) [Vulka        ] blk.19.attn_norm.wei (  11K) [Vulka        ]
node #1069 (   MUL_MAT):              Qcur-19 (  16K) [Vulka        ] use=1: blk.19.attn_q.weight (  11M) [Vulka        ]         attn_norm-19 (  11K) [Vulka        ]
node #1070 (       ADD):              Qcur-19 (  16K) [Vulka        ] use=1:              Qcur-19 (  16K) [Vulka        ]   blk.19.attn_q.bias (  16K) [Vulka        ]
node #1072 (      ROPE):              Qcur-19 (  16K) [Vulka        ] use=1:   Qcur-19 (reshaped) (  16K) [Vulka        ]     Vulkan0#leaf_5#0 (   0K) [ NULL        ]
node #1073 (   MUL_MAT):              Kcur-19 (   2K) [Vulka        ] use=1: blk.19.attn_k.weight (   1M) [Vulka        ]         attn_norm-19 (  11K) [Vulka        ]
node #1074 (       ADD):              Kcur-19 (   2K) [Vulka        ] use=1:              Kcur-19 (   2K) [Vulka        ]   blk.19.attn_k.bias (   2K) [Vulka        ]
node #1076 (      ROPE):              Kcur-19 (   2K) [Vulka        ] use=1:   Kcur-19 (reshaped) (   2K) [Vulka        ]     Vulkan0#leaf_5#0 (   0K) [ NULL        ]
node #1077 (   MUL_MAT):              Vcur-19 (   2K) [Vulka        ] use=1: blk.19.attn_v.weight (   1M) [Vulka        ]         attn_norm-19 (  11K) [Vulka        ]
node #1078 (       ADD):              Vcur-19 (   2K) [Vulka        ] use=1:              Vcur-19 (   2K) [Vulka        ]   blk.19.attn_v.bias (   2K) [Vulka        ]
node #1081 (  SET_ROWS):   cache_k_l19 (view) ( 128K) [Vulka        ] use=0:       Kcur-19 (view) (   2K) [Vulka        ]    Vulkan0#leaf_34#0 (   0K) [ NULL        ]          cache_k_l19 ( 128K) [Vulka        ]
node #1083 (  SET_ROWS):   cache_v_l19 (view) ( 128K) [Vulka        ] use=0:       Vcur-19 (view) (   2K) [Vulka        ]    Vulkan0#leaf_36#0 (   0K) [ NULL        ]          cache_v_l19 ( 128K) [Vulka        ]
node #1090 (FLASH_ATTN):         __fattn__-19 (  16K) [Vulka        ] use=1: Qcur-19 (view) (perm (  16K) [Vulka        ] cache_k_l19 (view) ( ( 128K) [Vulka        ] cache_v_l19 (view) ( ( 128K) [Vulka        ]...
node #1092 (   MUL_MAT):            node_1092 (  11K) [Vulka        ] use=1: blk.19.attn_output.w (  11M) [Vulka        ]           kqv_out-19 (  16K) [Vulka        ]
node #1093 (       ADD):          attn_out-19 (  11K) [Vulka        ] use=1:           node_1092 (  11K) [Vulka        ] blk.19.attn_output.b (  11K) [Vulka        ]
node #1094 (       ADD):           ffn_inp-19 (  11K) [Vulka        ] use=2:         attn_out-19 (  11K) [Vulka        ]            l_out-18 (  11K) [Vulka        ]
node #1095 (  RMS_NORM):              norm-19 (  11K) [Vulka        ] use=1:          ffn_inp-19 (  11K) [Vulka        ]
node #1096 (       MUL):    attn_post_norm-19 (  11K) [Vulka        ] use=2:              norm-19 (  11K) [Vulka        ] blk.19.post_attentio (  11K) [Vulka        ]
node #1097 (   MUL_MAT):    ffn_moe_logits-19 (   0K) [Vulka        ] use=1: blk.19.ffn_gate_inp. ( 360K) [Vulka        ]    attn_post_norm-19 (  11K) [Vulka        ]
node #1098 (       ADD):     ffn_moe_probs-19 (   0K) [Vulka        ] use=2:    ffn_moe_logits-19 (   0K) [Vulka        ] blk.19.ffn_gate_inp. (   0K) [Vulka        ]
node #1100 (   ARGSORT):   ffn_moe_argsort-19 (   0K) [Vulka        ] use=1:     ffn_moe_probs-19 (   0K) [Vulka        ]
node #1102 (  GET_ROWS):   ffn_moe_weights-19 (   0K) [Vulka        ] use=1: ffn_moe_probs-19 (re (   0K) [Vulka        ]      ffn_moe_topk-19 (   0K) [Vulka        ]
node #1104 (  SOFT_MAX):            node_1104 (   0K) [Vulka        ] use=1: ffn_moe_weights-19 ( (   0K) [Vulka        ]
```

Kinds of Workloads

- Prompt Processing (pp):
  - Turn an (often large) input into tokens
  - Large batch size
  - Dominated by Matrix * Matrix multiplies

☑ Scales with GPU performance

- Token Generation (tg):
  - Generate tokens one at a time
  - Batch size of one
  - Matrix * Vector multiplies and other small operations

☒ Small batch size limited

- Image/Video generation:
  - Stable diffusion
  - Convolutions (often done via matrix multiplies)
  - Large batch size

☑ Scales with GPU performance

Kinds of Workloads

- Prompt Processing (pp):
  - Turn an (often large) input into tokens
  - Large batch size
  - Dominated by Matrix * Matrix multiplies

☑ Scales with GPU performance

- Token Generation (tg):
  - Generate tokens one at a time
  - Batch size of one
  - Matrix * Vector multiplies and other small operations

☒ Small batch size limited

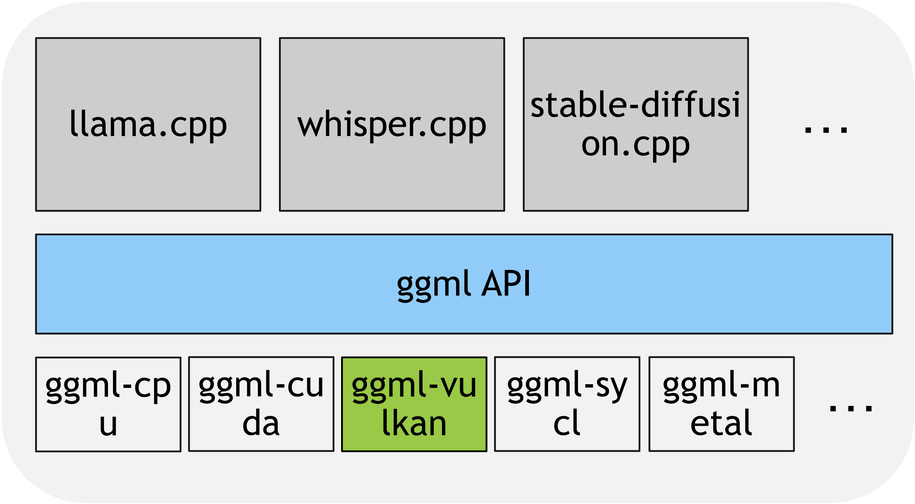
This is our focus

- Image/Video generation:
  - Stable diffusion
  - Convolutions (often done via matrix multiplies)
  - Large batch size

☑ Scales with GPU performance

The ggml-vulkan backend

- ggml-vulkan is a compute(+transfer)-only Vulkan library written using Vulkan-hpp
  - Targets a baseline of Vulkan 1.2, uses extensions if available
- Each node maps (usually) to one compute dispatch
- ggml_backend_vk_graph_compute:
  - for_each node:
    - select compute pipeline
    - compute push constants
    - bind buffers as SSBOs
    - determine number of workgroups
    - dispatch
    - pipeline barrier
- Shaders are handwritten, specialized based on types/constants
- Originally authored by @0cc4m (Ruben Ortlam)



ggml-vulkan shaders

GGML_OP_MUL shader

```
layout (push_constant) uniform parameter
{
    uint ne;
    uint ne00; uint ne01; uint ne02; uint ne03; uint nb00; uint nb01; uint nb02; uint nb03;
    uint ne10; uint ne11; uint ne12; uint ne13; uint nb10; uint nb11; uint nb12; uint nb13;
    uint ne20; uint ne21; uint ne22; uint ne23; uint nb20; uint nb21; uint nb22; uint nb23;
    uint misalign_offsets;
    float param1; float param2; int param3;
} p;

layout (binding = 0) readonly buffer A {A_TYPE data_a[];};
layout (binding = 1) readonly buffer B {B_TYPE data_b[];};
layout (binding = 2) writeonly buffer D {D_TYPE data_d[];};
layout(local_size_x = num_threads, local_size_y = 1, local_size_z = 1) in;

void main() {
    uint idx = get_idx();

    if (idx >= p.ne) {
        return;
    }
    uint i00, i01, i02, i03;
    get_indices(idx, i00, i01, i02, i03);

    data_d[get_doffset() + dst_idx(i00, i01, i02, i03)] =
        D_TYPE(FLOAT_TYPE(data_a[get_aoffset() + src0_idx(i00, i01, i02, i03)]) *
               FLOAT_TYPE(data_b[get_boffset() + src1_idx(i00, i01, i02, i03)]));
}
```

Tensor dimensions/strides

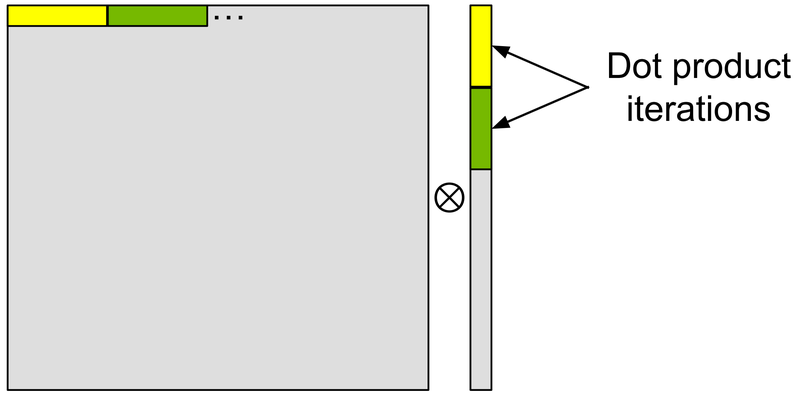
Invocation index to linear index

Linear index to 4D tensor coord

The ggml-vulkan backend

Matrix-Vector Multiplies

- Focus on spreading work across a lot of threads
  - Matrix-vector multiply computes a dot(row, vector) per workgroup
  - Many other ops also do a workgroup per row



- Matrix-vector multiplies also specialized/optimized for each quantization format
  - E.g. unrolled to do whole blocks (e.g. 256 elements)

```
#define QUANT_K_Q4_K 256
struct block_q4_K
{
    f16vec2 dm;
    uint8_t scales[3*QUANT_K_Q4_K/64];
    uint8_t qs[QUANT_K_Q4_K/2];
};
```

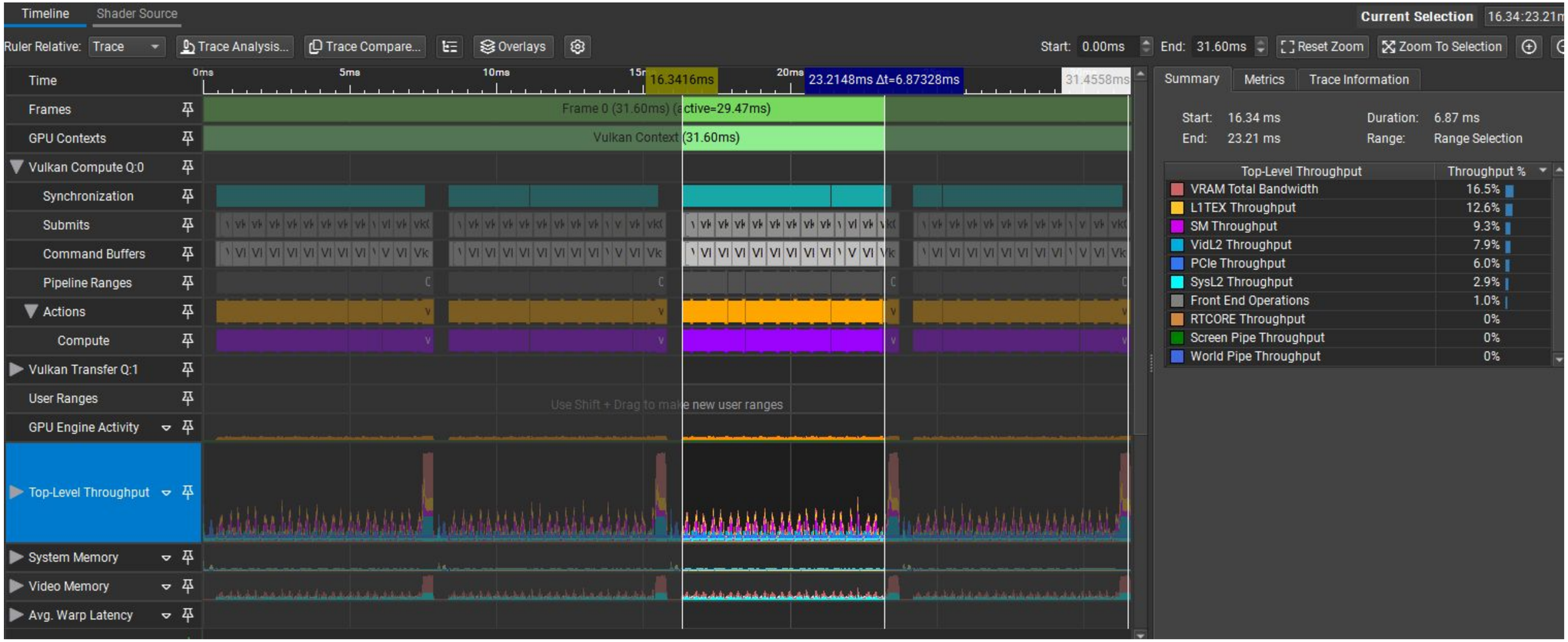
ggml Graphs Can Be Complex

Example from gpt-oss-20b

```
node #1051 (MUL_MAT_ID):      ffn_moe_gate-18 (  45K) [Vulka        ] use=1: blk.18.ffn_gate_exps ( 134M) [Vulka        ] attn_post_norm-18 (r (  11K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1052 (    ADD_ID): ffn_moe_gate_biased- (  45K) [Vulka        ] use=1:      ffn_moe_gate-18 (  45K) [Vulka        ] blk.18.ffn_gate_exps ( 360K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1053 (MUL_MAT_ID):        ffn_moe_up-18 (  45K) [Vulka        ] use=1: blk.18.ffn_up_exps.w ( 134M) [Vulka        ] attn_post_norm-18 (r (  11K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1054 (    ADD_ID): ffn_moe_up_biased-18 (  45K) [Vulka        ] use=1:        ffn_moe_up-18 (  45K) [Vulka        ] blk.18.ffn_up_exps.b ( 360K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1055 (       GLU):  ffn_moe_weighted-18 (  45K) [Vulka        ] use=1: ffn_moe_gate_biased- (  45K) [Vulka        ] ffn_moe_up_biased-18 (  45K) [Vulka        ]
node #1056 (MUL_MAT_ID):      ffn_moe_down-18 (  45K) [Vulka        ] use=1: blk.18.ffn_down_exps ( 134M) [Vulka        ]  ffn_moe_weighted-18 (  45K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1057 (    ADD_ID): ffn_moe_down_biased- (  45K) [Vulka        ] use=1:      ffn_moe_down-18 (  45K) [Vulka        ] blk.18.ffn_down_exps ( 360K) [Vulka        ]      ffn_moe_topk-18 (   0K) [Vulka        ]
node #1058 (       MUL):            node_1058 (  45K) [Vulka        ] use=4: ffn_moe_down_biased- (  45K) [Vulka        ] ffn_moe_weights_soft (   0K) [Vulka        ]
node #1063 (       ADD):            node_1063 (  11K) [Vulka        ] use=1:              (view) (  11K) [Vulka        ]     node_1058 (view) (  11K) [Vulka        ]
node #1064 (       ADD):            node_1064 (  11K) [Vulka        ] use=1:           node_1063 (  11K) [Vulka        ]     node_1058 (view) (  11K) [Vulka        ]
node #1065 (       ADD):       ffn_moe_out-18 (  11K) [Vulka        ] use=1:           node_1064 (  11K) [Vulka        ]     node_1058 (view) (  11K) [Vulka        ]
node #1066 (       ADD):             l_out-18 (  11K) [Vulka        ] use=2:      ffn_moe_out-18 (  11K) [Vulka        ]          ffn_inp-18 (  11K) [Vulka        ]
node #1067 (  RMS_NORM):              norm-19 (  11K) [Vulka        ] use=1:            l_out-18 (  11K) [Vulka        ]
node #1068 (       MUL):         attn_norm-19 (  11K) [Vulka        ] use=3:              norm-19 (  11K) [Vulka        ] blk.19.attn_norm.wei (  11K) [Vulka        ]
node #1069 (   MUL_MAT):              Qcur-19 (  16K) [Vulka        ] use=1: blk.19.attn_q.weight (  11M) [Vulka        ]         attn_norm-19 (  11K) [Vulka        ]
node #1070 (       ADD):              Qcur-19 (  16K) [Vulka        ] use=1:              Qcur-19 (  16K) [Vulka        ]   blk.19.attn_q.bias (  16K) [Vulka        ]
node #1072 (      ROPE):              Qcur-19 (  16K) [Vulka        ] use=1:   Qcur-19 (reshaped) (  16K) [Vulka        ]     Vulkan0#leaf_5#0 (   0K) [ NULL        ]
node #1073 (   MUL_MAT):              Kcur-19 (   2K) [Vulka        ] use=1: blk.19.attn_k.weight (   1M) [Vulka        ]         attn_norm-19 (  11K) [Vulka        ]
node #1074 (       ADD):              Kcur-19 (   2K) [Vulka        ] use=1:              Kcur-19 (   2K) [Vulka        ]   blk.19.attn_k.bias (   2K) [Vulka        ]
node #1076 (      ROPE):              Kcur-19 (   2K) [Vulka        ] use=1:   Kcur-19 (reshaped) (   2K) [Vulka        ]     Vulkan0#leaf_5#0 (   0K) [ NULL        ]
node #1077 (   MUL_MAT):              Vcur-19 (   2K) [Vulka        ] use=1: blk.19.attn_v.weight (   1M) [Vulka        ]         attn_norm-19 (  11K) [Vulka        ]
node #1078 (       ADD):              Vcur-19 (   2K) [Vulka        ] use=1:              Vcur-19 (   2K) [Vulka        ]   blk.19.attn_v.bias (   2K) [Vulka        ]
node #1081 (  SET_ROWS):   cache_k_l19 (view) ( 128K) [Vulka        ] use=0:       Kcur-19 (view) (   2K) [Vulka        ]    Vulkan0#leaf_34#0 (   0K) [ NULL        ]          cache_k_l19 ( 128K) [Vulka        ]
node #1083 (  SET_ROWS):   cache_v_l19 (view) ( 128K) [Vulka        ] use=0:       Vcur-19 (view) (   2K) [Vulka        ]    Vulkan0#leaf_36#0 (   0K) [ NULL        ]          cache_v_l19 ( 128K) [Vulka        ]
node #1090 (FLASH_ATTN):         __fattn__-19 (  16K) [Vulka        ] use=1: Qcur-19 (view) (perm (  16K) [Vulka        ] cache_k_l19 (view) ( ( 128K) [Vulka        ] cache_v_l19 (view) ( ( 128K) [Vulka        ]...
node #1092 (   MUL_MAT):            node_1092 (  11K) [Vulka        ] use=1: blk.19.attn_output.w (  11M) [Vulka        ]           kqv_out-19 (  16K) [Vulka        ]
node #1093 (       ADD):          attn_out-19 (  11K) [Vulka        ] use=1:           node_1092 (  11K) [Vulka        ] blk.19.attn_output.b (  11K) [Vulka        ]
node #1094 (       ADD):           ffn_inp-19 (  11K) [Vulka        ] use=2:         attn_out-19 (  11K) [Vulka        ]            l_out-18 (  11K) [Vulka        ]
node #1095 (  RMS_NORM):              norm-19 (  11K) [Vulka        ] use=1:          ffn_inp-19 (  11K) [Vulka        ]
node #1096 (       MUL):    attn_post_norm-19 (  11K) [Vulka        ] use=2:              norm-19 (  11K) [Vulka        ] blk.19.post_attentio (  11K) [Vulka        ]
node #1097 (   MUL_MAT):    ffn_moe_logits-19 (   0K) [Vulka        ] use=1: blk.19.ffn_gate_inp. ( 360K) [Vulka        ]    attn_post_norm-19 (  11K) [Vulka        ]
node #1098 (       ADD):     ffn_moe_probs-19 (   0K) [Vulka        ] use=2:    ffn_moe_logits-19 (   0K) [Vulka        ] blk.19.ffn_gate_inp. (   0K) [Vulka        ]
node #1100 (   ARGSORT):   ffn_moe_argsort-19 (   0K) [Vulka        ] use=1:     ffn_moe_probs-19 (   0K) [Vulka        ]
node #1102 (  GET_ROWS):   ffn_moe_weights-19 (   0K) [Vulka        ] use=1: ffn_moe_probs-19 (re (   0K) [Vulka        ]      ffn_moe_topk-19 (   0K) [Vulka        ]
node #1104 (  SOFT_MAX):            node_1104 (   0K) [Vulka        ] use=1: ffn_moe_weights-19 ( (   0K) [Vulka        ]
```

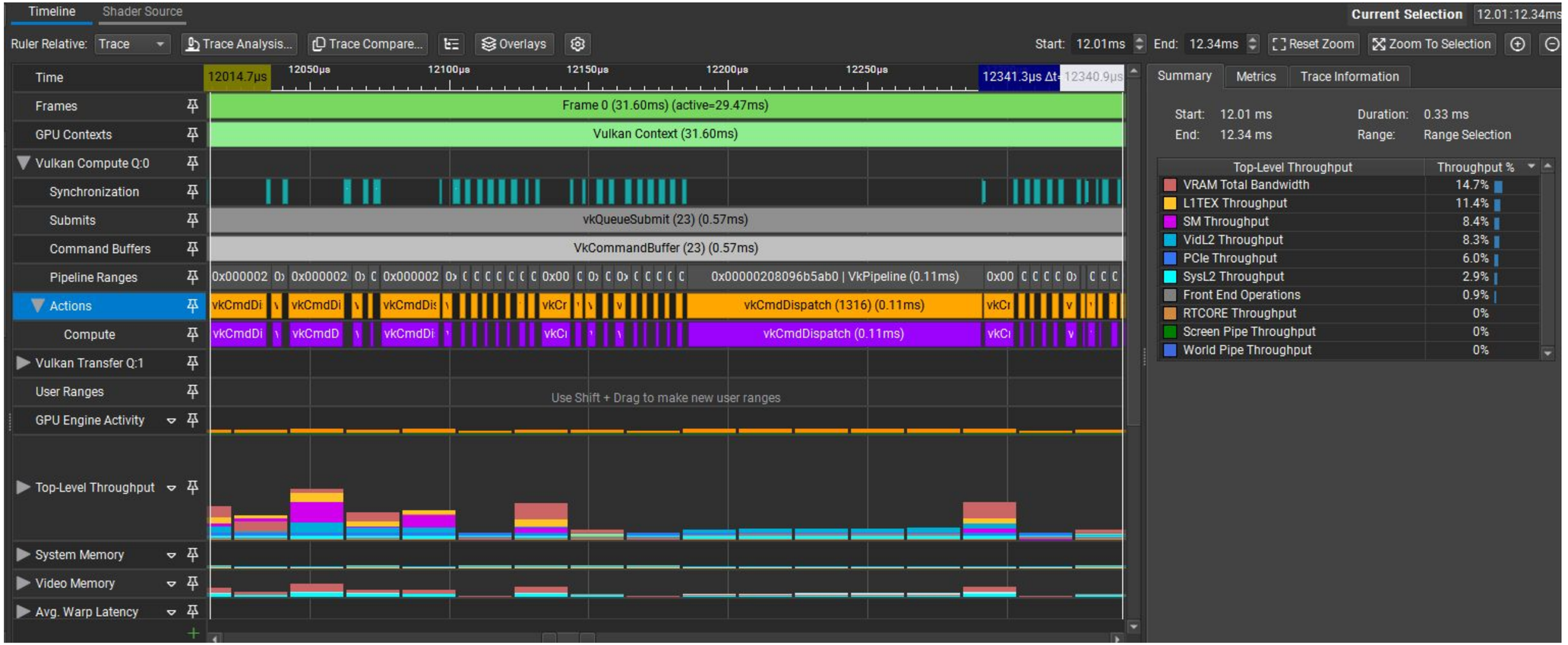
Where is the time spent?

gpt-oss-20b on RTX 5090



Where is the time spent?

gpt-oss-20b on RTX 5090




```
MUL_MAT_ID
    ADD_ID
MUL_MAT_ID
    ADD_ID
    GLU
MUL_MAT_ID
    ADD_ID
       MUL
       ADD
       ADD
       ADD
       ADD
  RMS_NORM
       MUL
    MUL_MAT
       ADD
      ROPE
    MUL_MAT
       ADD
      ROPE
    MUL_MAT
       ADD
  SET_ROWS
  SET_ROWS
FLASH_ATTN
    MUL_MAT
       ADD
       ADD
  RMS_NORM
       MUL
    MUL_MAT
       ADD
   ARGSORT
  GET_ROWS
  SOFT_MAX
```

Flash Attention GQA

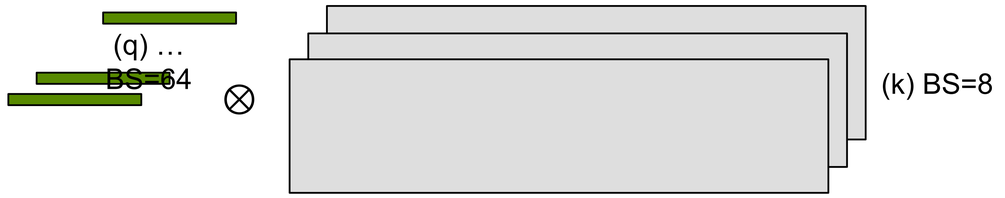
- Attention formula (from Wikipedia):

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \in \mathbb{R}^{m \times d_v}$$

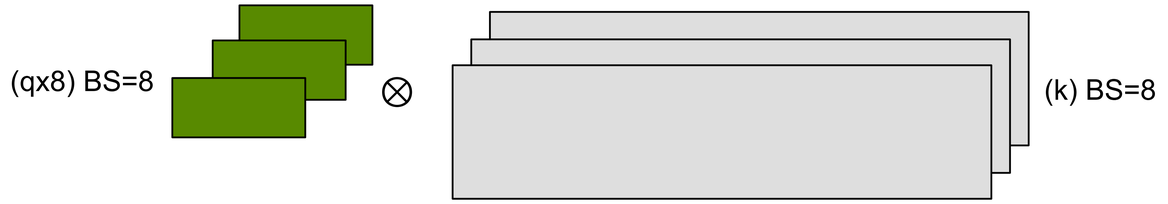
Softmax formula:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

- Example dimensions:
  - dst(64,64,1,1) = q(64,1,64,1) ⊗ k(64,4096,8,1) ⊗ v(64,4096,8,1)
  - 64 batches, each doing a (1x64 * 64x4096) * 4096x64 operation
  - There are 8 different 64x4096/4096x64 matrices, each used 8 times

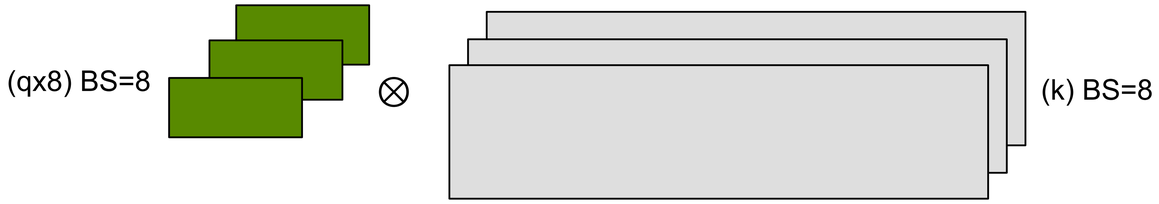


- Optimization for “Grouped Query Attention”:
  - KV cache has a certain number of “heads” (e.g. 8). Think of each head as a 2D matrix, #heads is the batch dimension
  - The network wants to compute 64 Q vectors, 8 in each head
  - Naively, this is 64 FA calculations, each with vector input/vector output
  - But, we can detect that groups use the same slice of KV, treat it as 8 matrix FA calculations on 8-high matrix

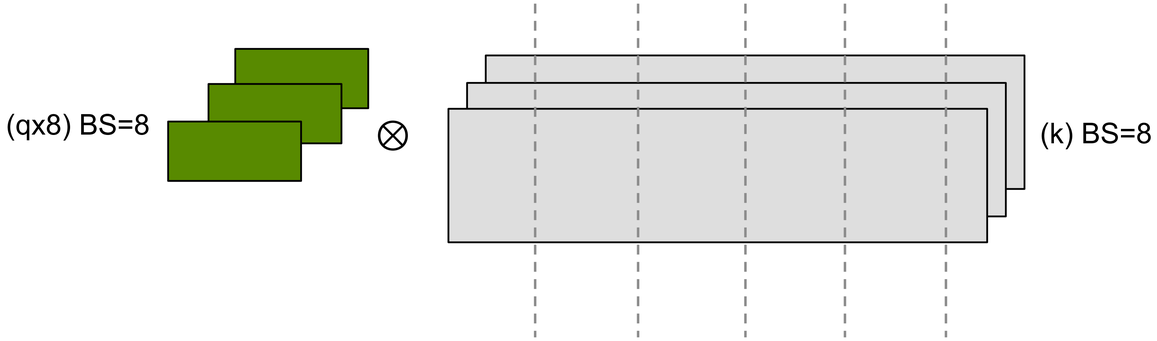


Flash Attention Split-K

- This may actually be slower!
  - Now we only have 8 workgroups
  - Only 8 SMs are busy…

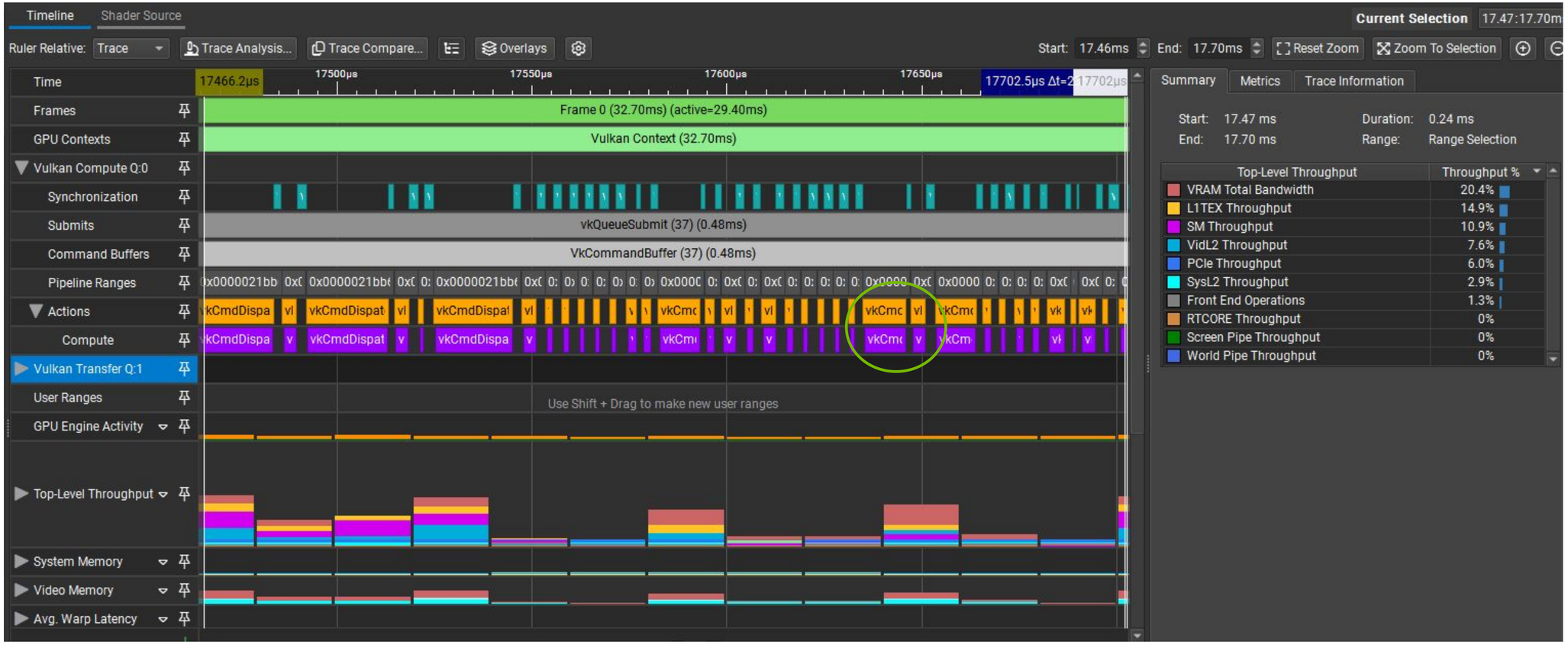


- Now let's dice up the work a better way:
  - The KV dimension is large
    - Relates to how much the model will "remember"
    - 4096 in our example – could be much larger
  - Enable "split-k" – divide matrices along the KV dimension
  - Each SM works on a different part of KV, then resolve at the end
- We can spread the work across any number of SMs
- Each SM is doing a mat-mat mul (tensor cores!)



Flash Attention (After)

gpt-oss-20b on RTX 5090




```
MUL_MAT_ID
    ADD_ID
MUL_MAT_ID
    ADD_ID
    GLU
MUL_MAT_ID
    ADD_ID
       MUL
       ADD
       ADD
       ADD
       ADD
  RMS_NORM
       MUL
   MUL_MAT
       ADD
      ROPE
   MUL_MAT
       ADD
      ROPE
   MUL_MAT
       ADD
  SET_ROWS
  SET_ROWS
FLASH_ATTN
   MUL_MAT
       ADD
       ADD
  RMS_NORM
       MUL
   MUL_MAT
       ADD
   ARGSORT
  GET_ROWS
  SOFT_MAX
```

Kernel Fusion

```
node #1067 (  RMS_NORM):              norm-19 (  11K) [Vulka         ] use=1:             l_out-18 (  11K) [Vulka         ]
node #1068 (       MUL):         attn_norm-19 (  11K) [Vulka         ] use=3:              norm-19 (  11K) [Vulka         ] blk.19.attn_norm.wei (  11K) [Vulka         ]
```

- RMS_NORM (vector normalized by root-mean-square) is very often followed by an elementwise multiply
- The RMS_NORM result is not otherwise used (see: **`use=1`**)
- Combine/Fuse the two shaders and only compute the result of the MUL
- This is just about the simplest case, but it kicks in very frequently

- This is similar to a compiler peephole optimization, but the fused shader is handwritten
- Challenges are similar to a compiler:
  - Detect sequence of instructions
  - Make sure operands/parameters of those instructions exactly match expectations
  - Make sure there are no side effects

Other Fusions

MUL_MAT + ADD (+ ADD) MUL_MAT_ID + ADD_ID (+ MUL) MUL_MAT_ID + MUL	Element-wise binary op with another tensor or two
ROPE + VIEW + SET_ROWS RMS_NORM + MUL + ROPE + VIEW + SET_ROWS	Rotate pairs of elements – remap output to the KV cache. (Almost element-wise)
ADD + ADD + … + ADD	Summing Expert results after MUL_MAT_ID
"TopK-MoE" (1): SOFT_MAX + RESHAPE + ARGSORT + VIEW + GET_ROWS (2): (1) + RESHAPE + SUM_ROWS + CLAMP + DIV + RESHAPE (3): ARGSORT + VIEW + GET_ROWS + RESHAPE + SOFT_MAX + RESHAPE (4): SIGMOID + RESHAPE + ADD + ARGSORT + VIEW + GET_ROWS + ...	Convert vectors to probabilities Sort and select top-K most probable tokens Fetch rows for those tokens, and optionally normalize

- Generally see +3-5% improvement in MoE models on RTX 5090 when a fusion is enabled

Scheduling/Barrier Optimizations

- Rather than a PipelineBarrier between each dispatch, track and defer until needed
  - Keep lists of unsynced nodes: unsynced_nodes_written, unsynced_nodes_read
  - When processing a node:
    - Check destination against unsynced written+read
    - Check all srcs against unsynced written
  - Important to check for tensors overlapping in memory – not just comparing logical tensors

```
Original graph:

903 (   MUL_MAT):              Qcur-29
905 (      ROPE):              Qcur-29
906 (   MUL_MAT):              Kcur-29
908 (      ROPE):              Kcur-29
909 (   MUL_MAT):              Vcur-29
912 (  SET_ROWS):   cache_k_l29 (view)
914 (  SET_ROWS):   cache_v_l29 (view)
921 (FLASH_ATTN):         __fattn__-29
```

```
Optimized graph:

sync MUL_MAT Qcur-29
sync ROPE Qcur-29
     MUL_MAT Kcur-29
sync ROPE Kcur-29
     MUL_MAT Vcur-29
sync SET_ROWS cache_k_l29 (view)
     SET_ROWS cache_v_l29 (view)
sync FLASH_ATTN_EXT __fattn__-29
```

Scheduling/Barrier Optimizations

- Pre-sort the graph to move independent nodes next to each other, to maximize the benefit of deferred barriers
  - Greedy algorithm to bunch together nodes that don't logically depend on each other
    - At each step, first bunch together real nodes, then bunch together "view" nodes
    - Look for fusion opportunities, and be careful not to break existing fusion opportunities
- Initially only applied for token-gen workloads, now always applied

```
Original graph:

903 (   MUL_MAT):              Qcur-29
905 (      ROPE):              Qcur-29
906 (   MUL_MAT):              Kcur-29
908 (      ROPE):              Kcur-29
909 (   MUL_MAT):              Vcur-29
912 (  SET_ROWS):   cache_k_l29 (view)
914 (  SET_ROWS):   cache_v_l29 (view)
921 (FLASH_ATTN):         __fattn__-29
```

```
Optimized graph:

sync MUL_MAT Qcur-29
     MUL_MAT Kcur-29
     MUL_MAT Vcur-29
sync ROPE Qcur-29
     ROPE Kcur-29
     SET_ROWS cache_k_l29 (view)
     SET_ROWS cache_v_l29 (view)
sync FLASH_ATTN_EXT __fattn__-29
```

Scheduling/Barrier Optimizations

gpt-oss-20b

Used to be 4 ops

```
sync 1050 MUL_MAT_ID
     1051 ADD_ID
     1052 MUL_MAT_ID
     1053 ADD_ID
sync 1054 GLU
sync 1055 MUL_MAT_ID
     1056 ADD_ID
     1057 MUL
sync 1062 ADD
     1063 ADD
     1064 ADD
     1065 ADD
sync 1066 RMS_NORM
     1067 MUL
sync 1068 MUL_MAT
     1069 ADD
     1070 MUL_MAT
     1071 ADD
     1072 MUL_MAT
     1073 ADD
(continued on the right)
```

```
sync 1078 ROPE
     1079 ROPE
     1081 SET_ROWS
     1082 SET_ROWS
sync 1089 FLASH_ATTN
sync 1091 MUL_MAT
     1092 ADD
     1093 ADD
sync 1094 RMS_NORM
     1095 MUL
sync 1097 MUL_MAT
     1098 ADD
sync 1100 ARGSORT
     1101 VIEW
     1102 GET_ROWS
     1103 RESHAPE
     1104 SOFT_MAX
     1105 RESHAPE
```

Down to 12 syncs out of 35 nodes!

Scheduling/Barrier Optimizations

Qwen3

Used to be 4 ops

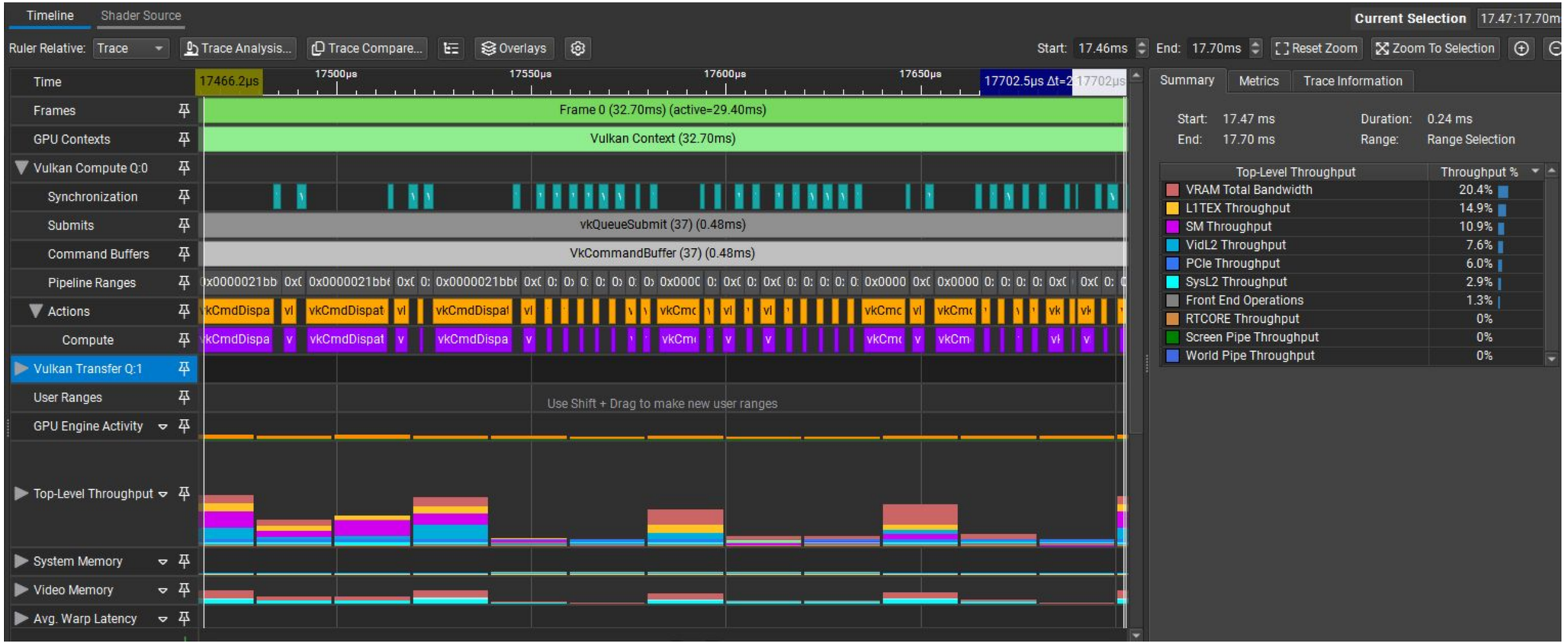
```
sync 1807 MUL_MAT_ID ffn_moe_gate-28
     1808 MUL_MAT_ID ffn_moe_up-28
sync 1809 GLU ffn_moe_weighted-28 SWIGLU split
sync 1810 MUL_MAT_ID ffn_moe_down-28
     1811 MUL node_1812
sync 1820 ADD node_1821
     1821 ADD node_1822
     1822 ADD node_1823
     1823 ADD node_1824
     1824 ADD node_1825
     1825 ADD node_1826
     1826 ADD ffn_moe_out-28
     1827 ADD l_out-28
sync 1828 RMS_NORM norm-29
     1829 MUL attn_norm-29
sync 1830 MUL_MAT Qcur-29
     1831 MUL_MAT Kcur-29
     1832 MUL_MAT Vcur-29
(continued on the right)
```

Down to 12 syncs out of 38 nodes!

```
sync 1837 RMS_NORM norm-29
     1838 MUL Qcur_normed-29
     1839 ROPE Qcur-29 rope mode: 2
     1840 RMS_NORM norm-29
     1841 MUL Kcur_normed-29
     1842 ROPE Kcur-29 rope mode: 2
     1843 VIEW Kcur-29 (view)
     1844 SET_ROWS cache_k_l29 (view)
     1845 SET_ROWS cache_v_l29 (view)
sync 1852 FLASH_ATTN_EXT __fattn__-29
sync 1854 MUL_MAT node_1855
     1855 ADD ffn_inp-29
sync 1856 RMS_NORM norm-29
     1857 MUL ffn_norm-29
sync 1859 MUL_MAT ffn_moe_logits-29
sync 1860 SOFT_MAX ffn_moe_probs-29
     1861 RESHAPE ffn_moe_probs-29 (reshaped)
     1862 ARGSORT ffn_moe_argsort-29
     1863 VIEW ffn_moe_topk-29
     1864 GET_ROWS ffn_moe_weights-29
     1865 RESHAPE ffn_moe_weights-29 (reshaped)
     1866 SUM_ROWS ffn_moe_weights_sum-29
     1867 CLAMP ffn_moe_weights_sum_clamped-29
     1868 DIV ffn_moe_weights_norm-29
     1869 RESHAPE ffn_moe_weights_norm-29 (reshaped)
```

Before Fusion/Barrier Optimizations

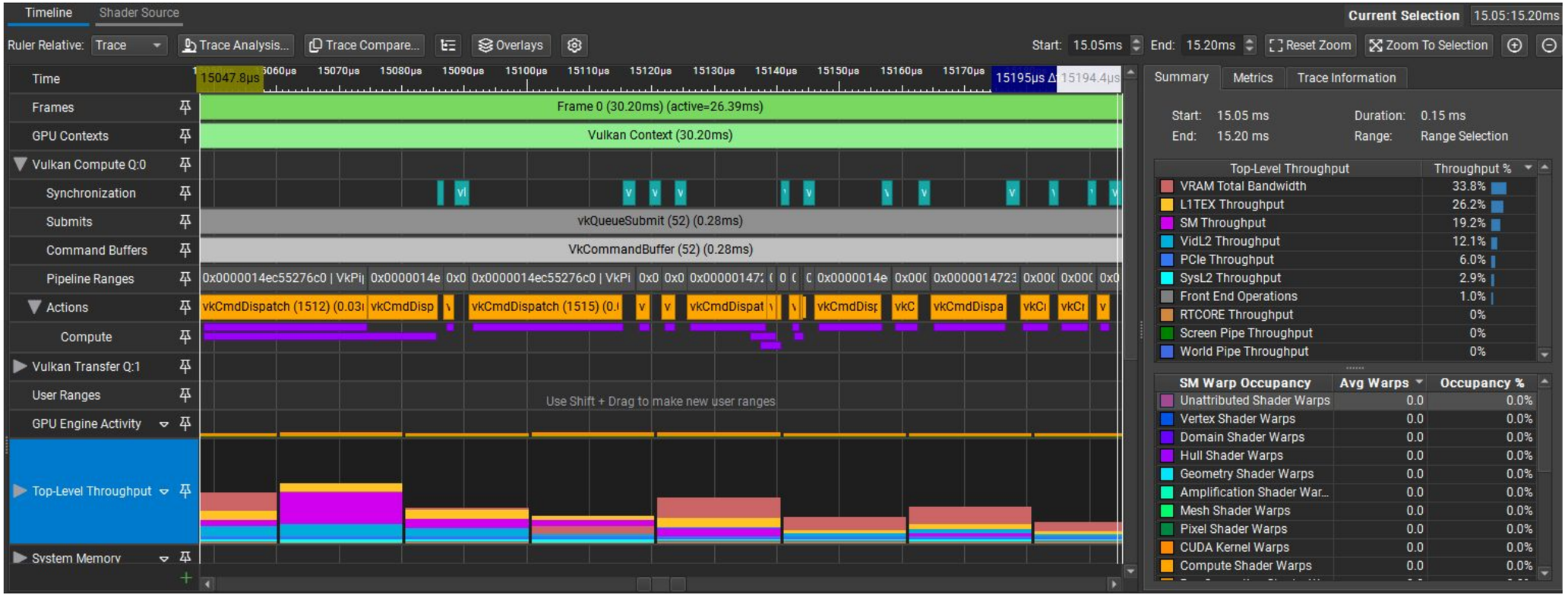
gpt-oss-20b on RTX 5090




```
1050 MUL_MAT_ID
1051 ADD_ID
1052 MUL_MAT_ID
1053 ADD_ID
1054 GLU
1055 MUL_MAT_ID
1056 ADD_ID
1057 MUL
1062 ADD
1063 ADD
1064 ADD
1065 ADD
1066 RMS_NORM
1067 MUL
1068 MUL_MAT
1069 ADD
1070 MUL_MAT
1071 ADD
1072 MUL_MAT
1073 ADD
1078 ROPE
1079 ROPE
1081 SET_ROWS
1082 SET_ROWS
1089 FLASH_ATTN
1091 MUL_MAT
1092 ADD
1093 ADD
1094 RMS_NORM
1095 MUL
1097 MUL_MAT
1098 ADD
1100 ARGSORT
1101 VIEW
1102 GET_ROWS
1103 RESHAPE
1104 SOFT_MAX
1105 RESHAPE
```

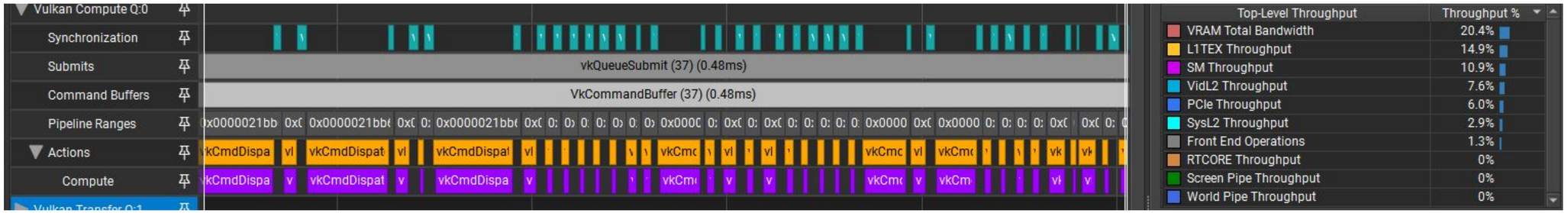
After Fusion/Barrier Optimizations

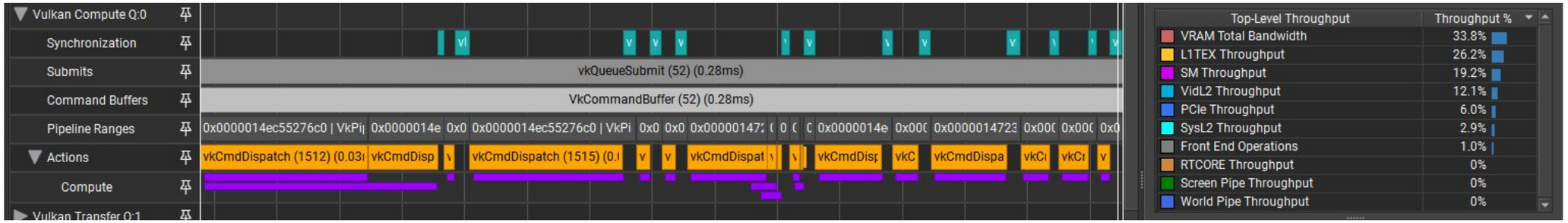
gpt-oss-20b on RTX 5090



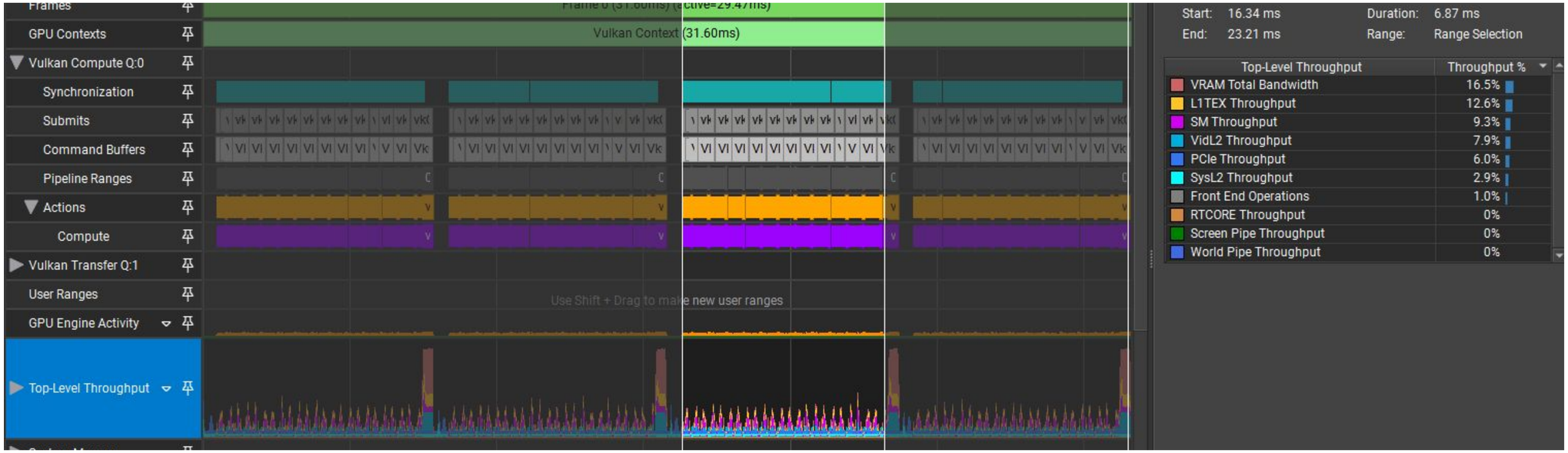

```
sync 1050 MUL_MAT_ID
     1051 ADD_ID
     1052 MUL_MAT_ID
     1053 ADD_ID
sync 1054 GLU
sync 1055 MUL_MAT_ID
     1056 ADD_ID
     1057 MUL
sync 1062 ADD
     1063 ADD
     1064 ADD
     1065 ADD
sync 1066 RMS_NORM
     1067 MUL
sync 1068 MUL_MAT
     1069 ADD
     1070 MUL_MAT
     1071 ADD
     1072 MUL_MAT
     1073 ADD
sync 1078 ROPE
     1079 ROPE
     1081 SET_ROWS
     1082 SET_ROWS
sync 1089 FLASH_ATTN
sync 1091 MUL_MAT
     1092 ADD
     1093 ADD
sync 1094 RMS_NORM
     1095 MUL
sync 1097 MUL_MAT
     1098 ADD
sync 1100 ARGSORT
     1101 VIEW
     1102 GET_ROWS
     1103 RESHAPE
     1104 SOFT_MAX
     1105 RESHAPE
```

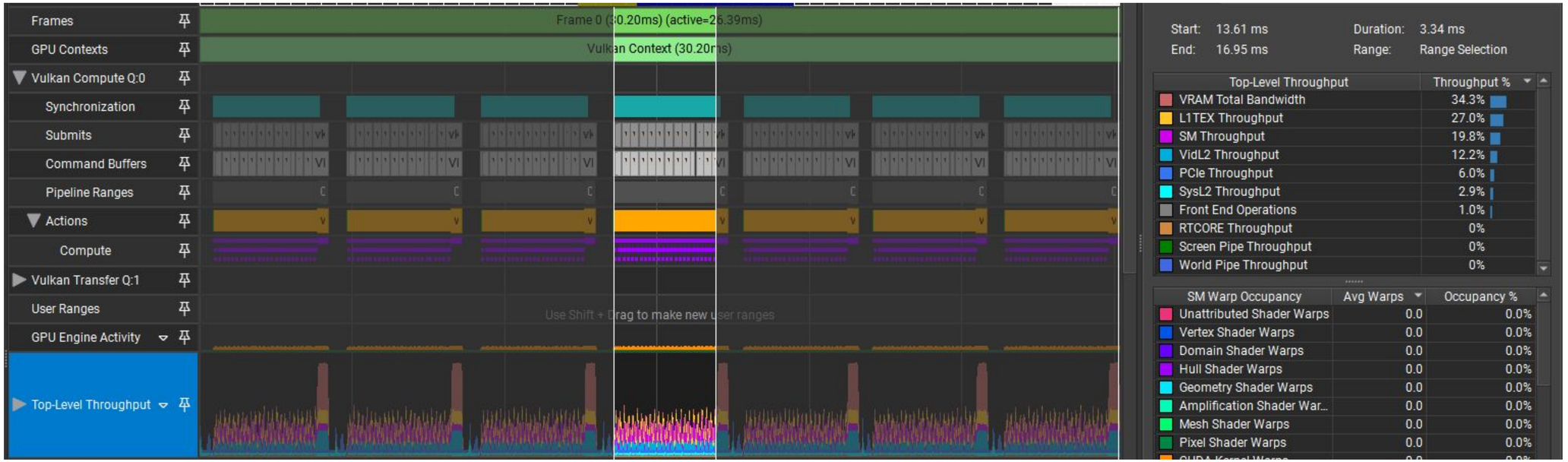
Before/After



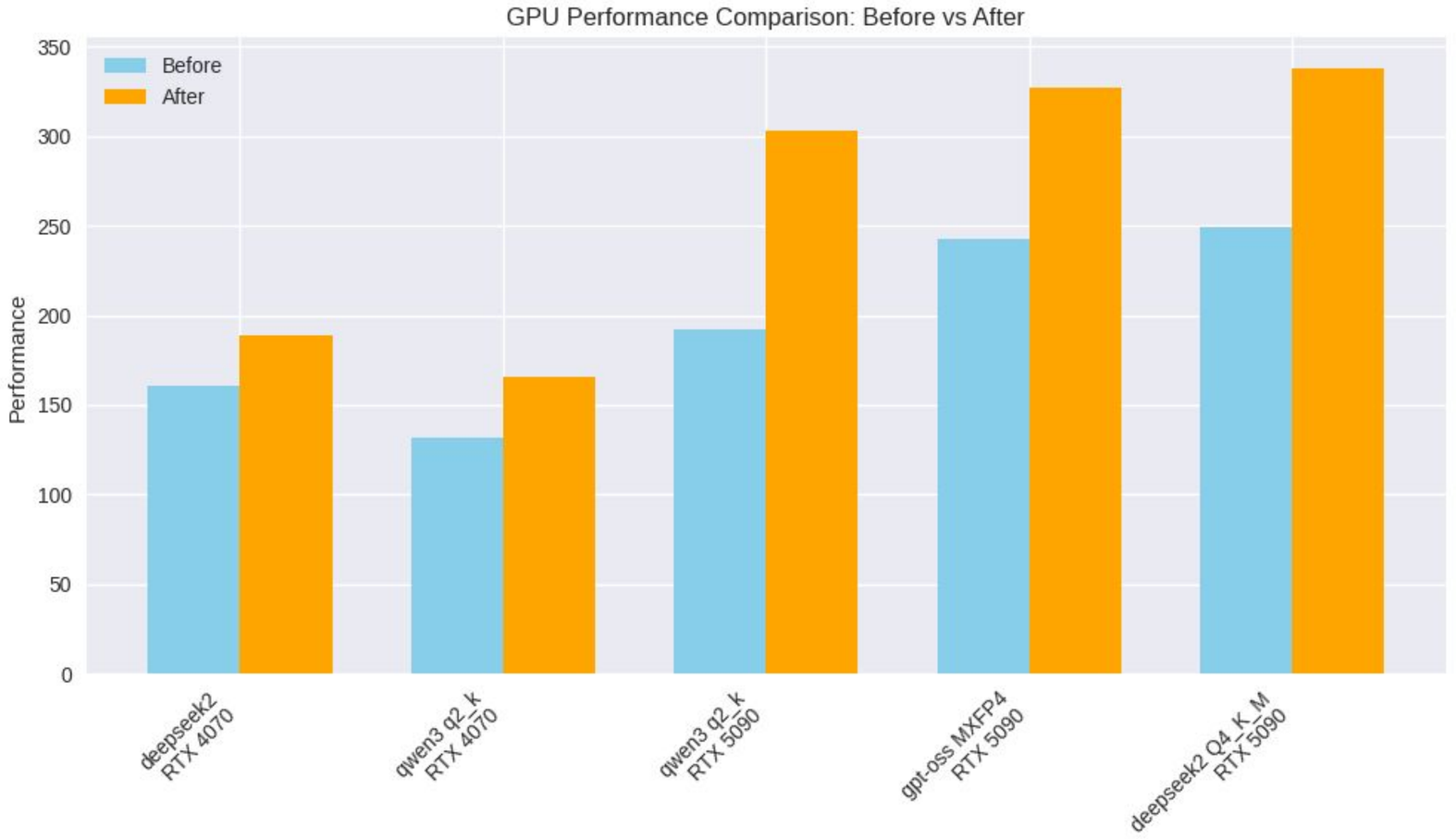


Before/After





Performance



Conclusion

- >50% speedup from initial fusion/scheduling work
- Possible future directions
  - As MoEs take over, we should continue to fuse aggressively
  - The current design using descriptor bindings is clumsy for fusion – use BDA instead?
  - Generate fused shaders on the fly?

- Try it yourself in https://github.com/ggml-org/llama.cpp/!


NVIDIA