Optimizing IREE to Match llama.cpp

An introduction to IREE optimization for newbies through a benchmark journey

ML2

What is IREE

- A retargetable MLIR-based compiler for ML-Programs
- Takes various frontend inputs(Pytorch, JAX, ONNX, etc.) then execute on different backends (x86, ARM, NVIDIA GPU, AMD GPU, etc.)

Why choose IREE as a test bed for MLIR

- **Team goal: Build real AI compilation intuition - not just tutorials, but shipping code.**
- **Why IREE?**
  - MLIR-native, end-to-end
  - Right scale for newcomers
  - Abstractions on hardwares
- **Outcome we wanted:**
  - shared compiler intuition + contributions that live beyond this talk

Benchmarking IREE on TinyLamma

- **Reasoning behind choosing TinyLamma over Llama-3.1-8B instruct**
    - Previously worked on Llama-3.1-8B instruct
    - Switched to TinyLamma for faster iteration
- **Without NVIDIA / AMD GPUs**
    - Targeting edge/CPU deployments

Basic Benchmark Results (Compared to Lamma.cpu)

- Raw one step forward function: no optimization during inference given
    - 23000 ms / token
- llama.cpu
    - 84 ms / token
- 250x speed difference 🤯

What Actually Worked For Debugging

- CPU, Kernel Profiling didn't help
- Instead, Tracy is your friend
  - Use Tracy to check the slowest frame
  - Then use `--mlir-print-ir-after-all` to check the MLIR per passes
- Diffing the intermediate MLIR files

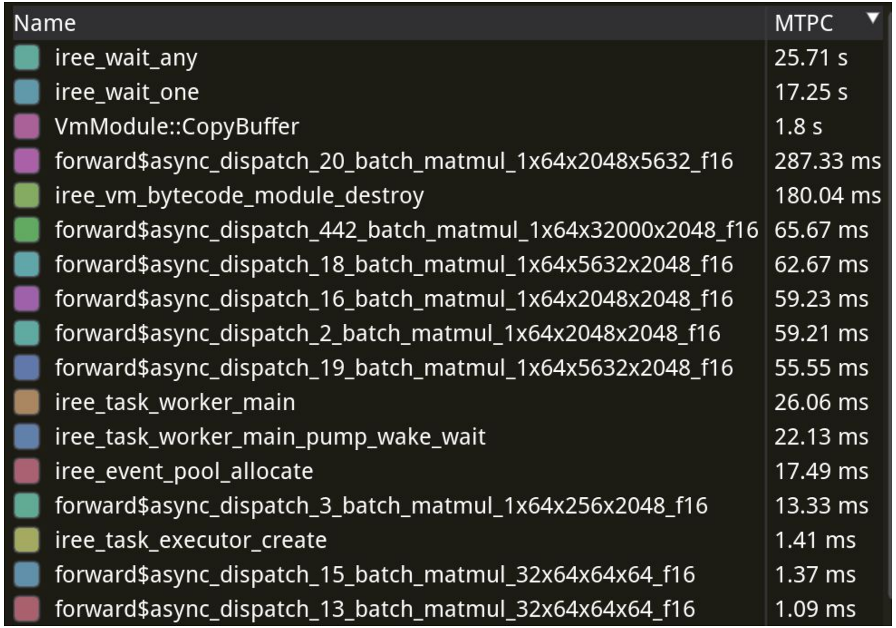
What Actually Worked For Debugging

- **MLIR printing**
  - Huge MLIR Constants in tensor operations deteriorates debugging experience
    - 4GB sized MLIR file
  - Tip. Use DenseResourceElementsAttr to reduce the IR sizes. (or some way to bind the parameters externally) - IREE has NamedParameter Attribute in its flow dialect.

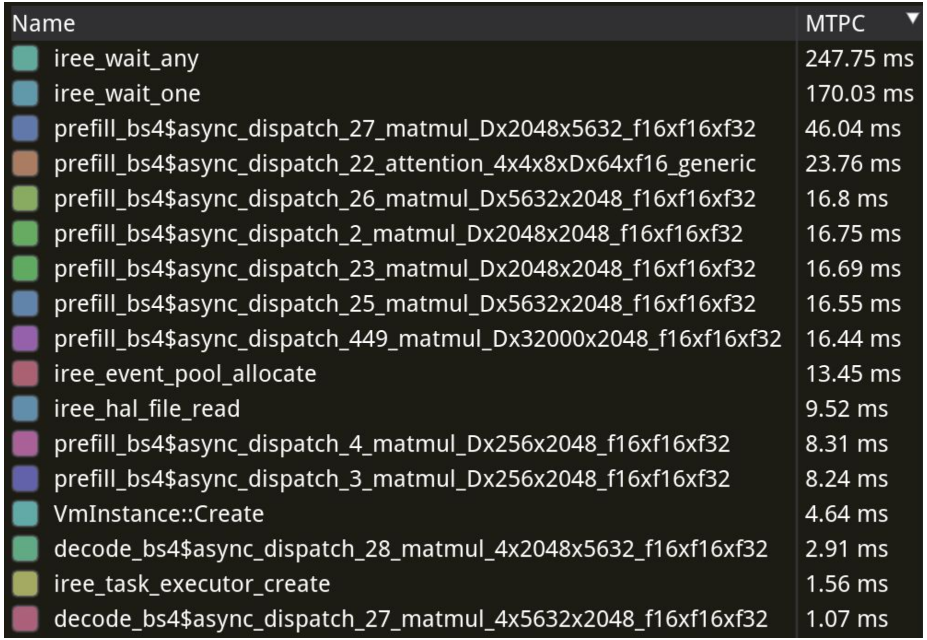
Actual optimizations that helped

- **KV-Cache with paged attention**
  - 23000m/s => 421 m/s
  - Still slower to llama.cpp (84 ms / token)
- **Better than any of the mlir optimizations and options**

Tracy Results



Name	MTPC
iree_wait_any	25.71 s
iree_wait_one	17.25 s
VmModule::CopyBuffer	1.8 s
forward$async_dispatch_20_batch_matmul_1x64x2048x5632_f16	287.33 ms
iree_vm_bytecode_module_destroy	180.04 ms
forward$async_dispatch_442_batch_matmul_1x64x32000x2048_f16	65.67 ms
forward$async_dispatch_18_batch_matmul_1x64x5632x2048_f16	62.67 ms
forward$async_dispatch_16_batch_matmul_1x64x2048x2048_f16	59.23 ms
forward$async_dispatch_2_batch_matmul_1x64x2048x2048_f16	59.21 ms
forward$async_dispatch_19_batch_matmul_1x64x5632x2048_f16	55.55 ms
iree_task_worker_main	26.06 ms
iree_task_worker_main_pump_wake_wait	22.13 ms
iree_event_pool_allocate	17.49 ms
forward$async_dispatch_3_batch_matmul_1x64x256x2048_f16	13.33 ms
iree_task_executor_create	1.41 ms
forward$async_dispatch_15_batch_matmul_32x64x64x64_f16	1.37 ms
forward$async_dispatch_13_batch_matmul_32x64x64x64_f16	1.09 ms

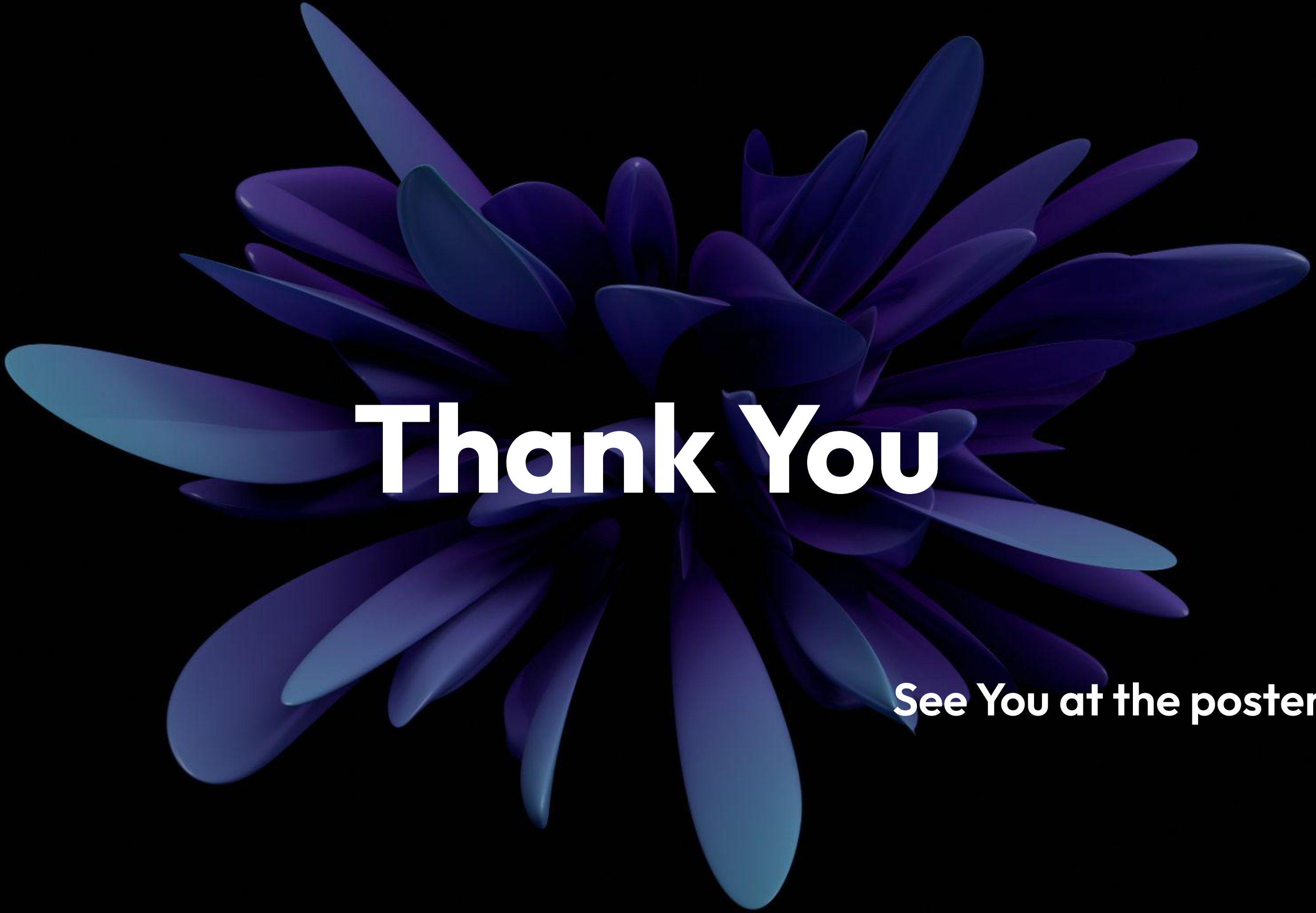


Name	MTPC
iree_wait_any	247.75 ms
iree_wait_one	170.03 ms
prefill_bs4$async_dispatch_27_matmul_Dx2048x5632_f16xf16xf32	46.04 ms
prefill_bs4$async_dispatch_22_attention_4x4x8xDx64xf16_generic	23.76 ms
prefill_bs4$async_dispatch_26_matmul_Dx5632x2048_f16xf16xf32	16.8 ms
prefill_bs4$async_dispatch_2_matmul_Dx2048x2048_f16xf16xf32	16.75 ms
prefill_bs4$async_dispatch_23_matmul_Dx2048x2048_f16xf16xf32	16.69 ms
prefill_bs4$async_dispatch_25_matmul_Dx5632x2048_f16xf16xf32	16.55 ms
prefill_bs4$async_dispatch_449_matmul_Dx32000x2048_f16xf16xf32	16.44 ms
iree_event_pool_allocate	13.45 ms
iree_hal_file_read	9.52 ms
prefill_bs4$async_dispatch_4_matmul_Dx256x2048_f16xf16xf32	8.31 ms
prefill_bs4$async_dispatch_3_matmul_Dx256x2048_f16xf16xf32	8.24 ms
VmInstance::Create	4.64 ms
decode_bs4$async_dispatch_28_matmul_4x2048x5632_f16xf16xf32	2.91 ms
iree_task_executor_create	1.56 ms
decode_bs4$async_dispatch_27_matmul_4x5632x2048_f16xf16xf32	1.07 ms

Take Aways

- Decode ≠ Prefill
- Observe before optimize
- Keep the IR small

ML2



Thank You

See You at the poster session