# ACCELERATING GEN-AI WORKLOADS BY ENABLING RISC-V MICROKERNEL SUPPORT IN IREE

Adeel Ahmad, Ahmad Tameem Kamal, Nouman Amir, Bilal Zafar, Saad Bin Nasir
10xEngineers

## ABSTRACT

This project enables RISC-V microkernel support in IREE, an MLIR-based machine learning compiler and runtime. The approach begins by enabling the lowering of MLIR contraction ops to microkernel calls for the RISC-V target within the IREE pass pipeline, followed by the development of hand-optimized and vectorized microkernels tailored to RISC-V. The performance gains are compared with upstream IREE and Llama.cpp for the Llama-3.2-1B-Instruct model.

## METHODOLOGY

Matrix multiplication is the core computation in Generative AI models as shown in Figure 1

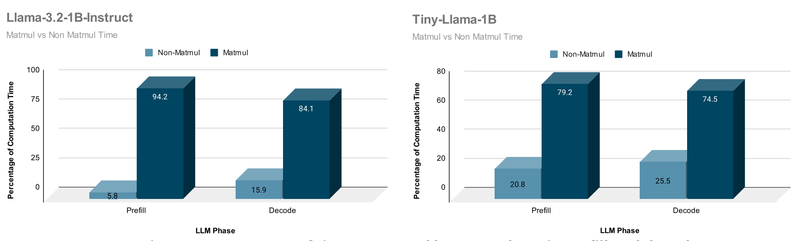
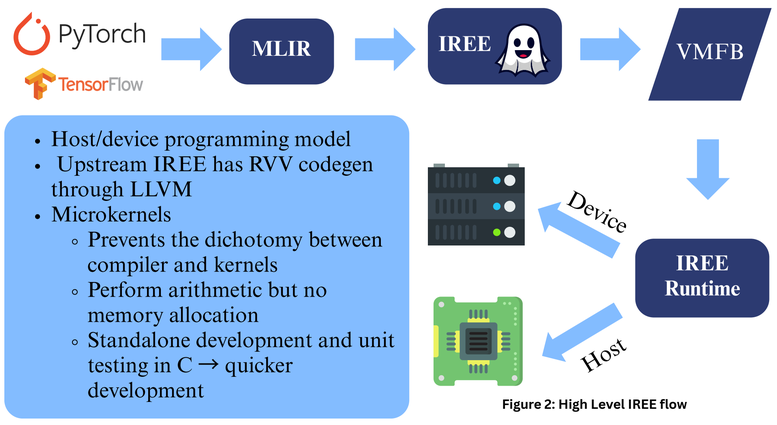



**Figure 1 : Percentage of time consumed by Matmul ops in prefill and decode stages of Llama-3.2-1B-Instruct (left) and Tiny-Llama-1B (right)**

We implemented matrix multiplication ukernels (micro-kernels) for RISC-V target in IREE[1], an MLIR[2] based compiler, and runtime.




- Host/device programming model
- Upstream IREE has RVV codegen through LLVM
- Microkernels
  - Prevents the dichotomy between compiler and kernels
  - Perform arithmetic but no memory allocation
  - Standalone development and unit testing in C → quicker development

**Figure 2: High Level IREE flow**

In IREE, Linalg contraction ops are replaced with mmt4d[3] op, which is subsequently lowered to the ukernel call.

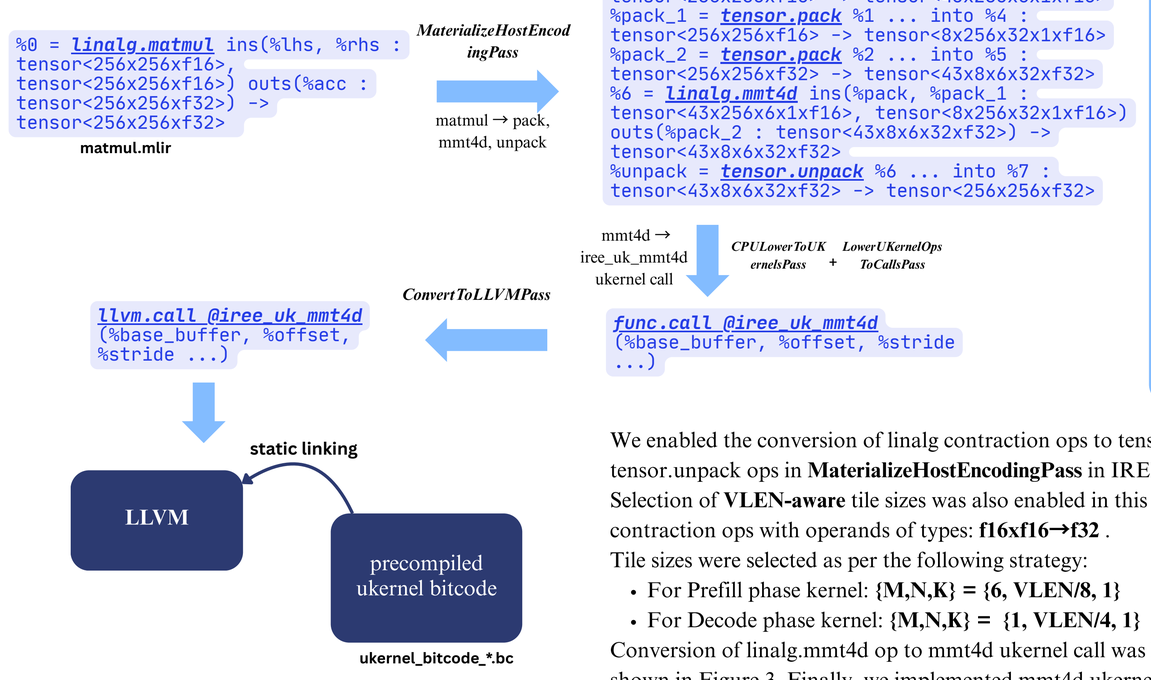



**Figure3: Transformation of linalg.matmul op in IREE**

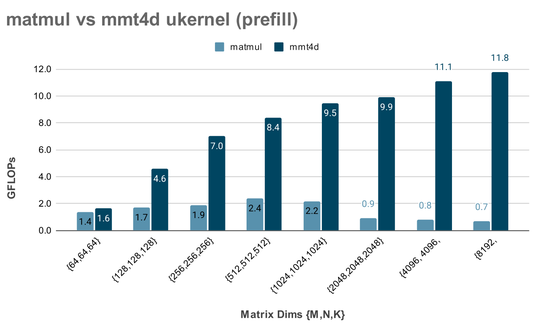
- In the case of plain MLIR linalg.matmul op, memory accesses are non-contiguous, which leads to a low cache hit rate when processing large matrices.
- To reduce the number of accesses to non-contiguous memory locations, **we need to reorder data**. IREE uses MLIR **tensor.pack** ops for this.
- After packing, elements **within each tile are stored contiguously in memory, and the tiles themselves are laid out contiguously with respect to one another.**
- MLIR linalg.mmt4d op operates on these packed matrices. This results in efficient cache utilization, and performance acceleration.

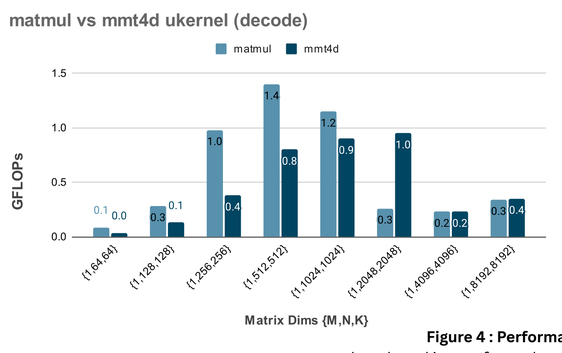
We enabled the conversion of linalg contraction ops to tensor.pack, linalg.mmt4d and tensor.unpack ops in **MaterializeHostEncodingPass** in IREE pass pipeline for **RISC-V** target. Selection of **VLEN-aware** tile sizes was also enabled in this pass for RISC-V target, for linalg contraction ops with operands of types: **f16xf16→f32** .
Tile sizes were selected as per the following strategy:

- For Prefill phase kernel: **{M,N,K} = {6, VLEN/8, 1}**
- For Decode phase kernel: **{M,N,K} = {1, VLEN/4, 1}**

Conversion of linalg.mmt4d op to mmt4d ukernel call was handled by subsequent passes as shown in Figure 3. Finally, we implemented mmt4d ukernels targeting RVV that resulted in LLM performance acceleration on RISC-V hardware.

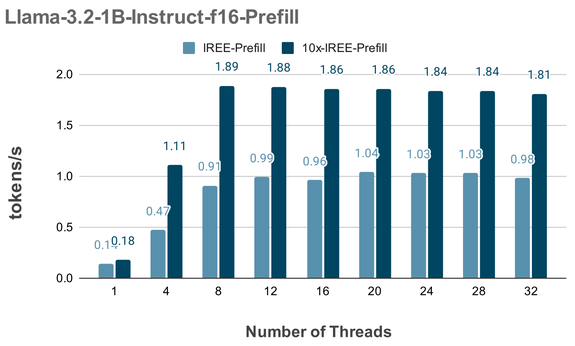
## RESULTS




MMT4D ukernel for prefill stage performs consistently better than matmul for a range of matrix sizes

- At kernel level, mmt4d for decode seems to cause performance degradation
- Smaller matrices → less impact of cache misses → pack op cost dominates in mmt4d
- Here, **>60% of execution time is consumed by pack op, in real models pack cost can be evaded leading to performance boost!**



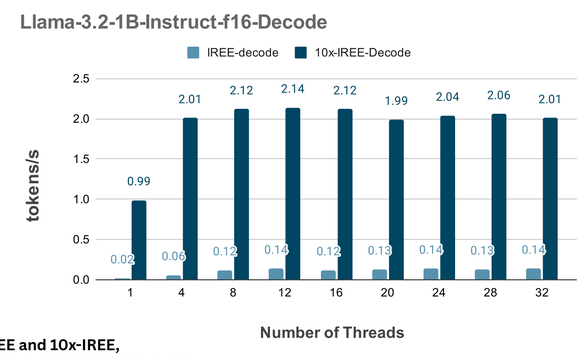




**Figure 4 : Performance comparison between upstream IREE and 10x-IREE,** benchmarking performed on MILK-V Jupiter board with 1.6GHz x 8 RVV cores, VLEN=256, RVA22

At LLM level, **2x performance improvement** was observed in the prefill phase for multi-threaded runs

At LLM level,

- **50x** perfromance improvement was observed in decode phase for single threaded runs
- **17x** perfromance improvement was observed in decode phase for multi-threaded runs

| Phase | Threads | Llama.cpp (toks/s) | IREE upstream (toks/s) | 10x-IREE (toks/s) |
|---|---|---|---|---|
| Prefill | 1 | 0.04 | 0.14 | 0.18 |
| | 8 | 0.11 | 0.91 | 1.89 |
| Decode | 1 | 0.03 | 0.02 | 0.99 |
| | 8 | 0.07 | 0.12 | 2.12 |

**Table 1: Performance comparison between Llama.cpp, upstream IREE and 10x-IREE for Llama-3.2-1B-Instruct model,** benchmarking performed on MILK-V Jupiter board with 1.6GHz x 8 RVV cores, VLEN=256, RVA22

**86.5%** of the computation time is consumed by **pack ops operating on weight tensors**.
These ops are executed only once, during prefill and aren't executed in decode at all, leading to higher performance gains in decode.
These pack ops can be evaluated at **compile-time** by enabling **const-eval** optimization in IREE, which will lead to further accleration of prefill phase.

## CONCLUSION

Custom matrix multiplication micro-kernels, targeting RVV, were implemented in IREE, resulting in **2x improvement in prefill phase performance and 50x improvement in decode** phase performance. One-time data prepacking cost is incurred on the first token generation during the prefill, and can be evaded if const-eval optimization is implemented in IREE.

## REFERENCES

[1] IREE. Accessed: Feb. 7, 2025. Feb. 2025. url: **https://iree.dev/**
[2] Chris Lattner et al. MLIR: A Compiler Infrastructure for the End of Moore's Law. 2020. arXiv: 2002.11054 [cs.PL]. url: **https://arxiv.org/abs/2002.11054**
[3] Matrix Multiplication with MMT4D. Accessed: Feb. 7, 2025. Oct. 2021. url: **https://iree.dev/community/blog/2021-10-13- matrix-multiplication-with-mmt4d/**

RISC-V SUMMIT EUROPE

10x ENGINEERS