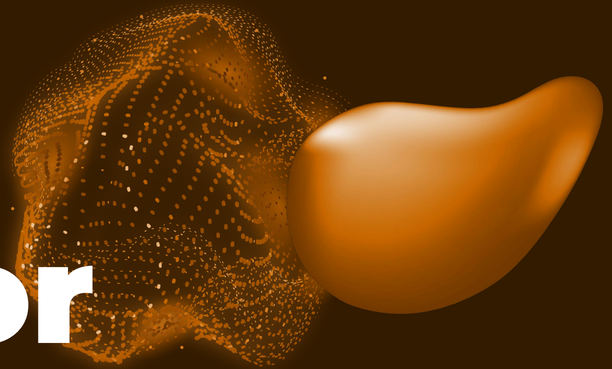
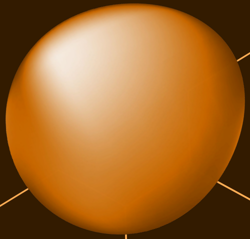


The Quest for Unification:

A Survey of Hardware-Agnostic Machine Learning Systems



Paul Dutton

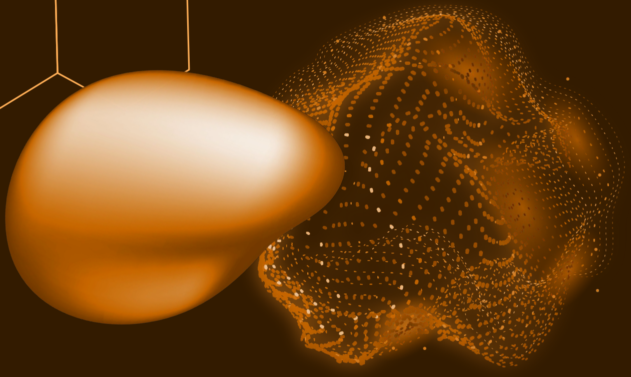
TABLE OF CONTENTS

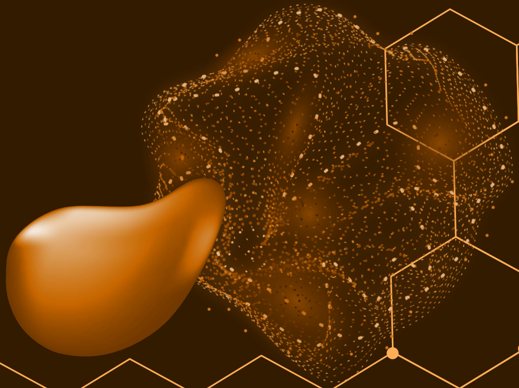
01	**Current State**	Two Language Problem, Competing Frameworks, Hardware
02	**“The Hardware Lottery”**	Hardware / software chicken and egg
03	**Modern Solutions**	Frameworks, Compilers, IRs, DSLs
04	**Ad for Chris Lattner**	Modular / Max / Mojo
05	**Takeaways & Questions**	Why care? What’s the future?

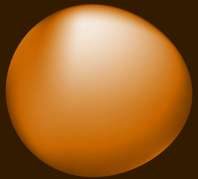
01

Introduction

What is the current state of AI software and hardware?







It’s a black box and/or hot glue

Apache TVM Path From Software To Hardware

Apache TVM: Compiler Framework

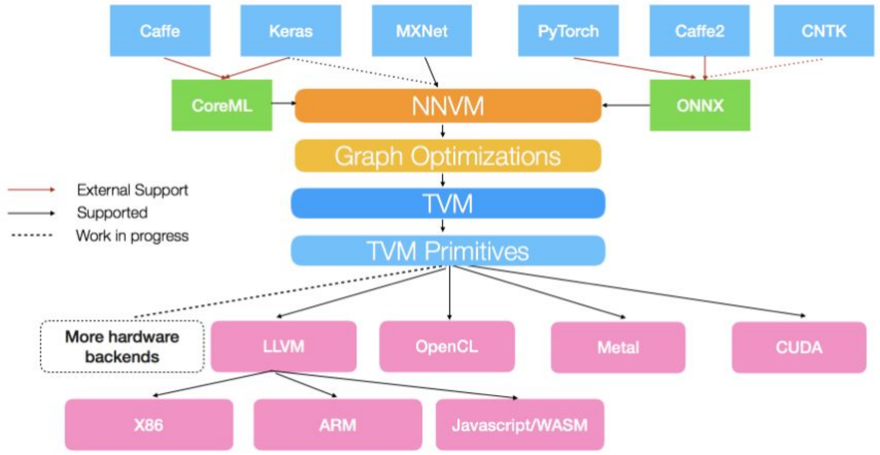
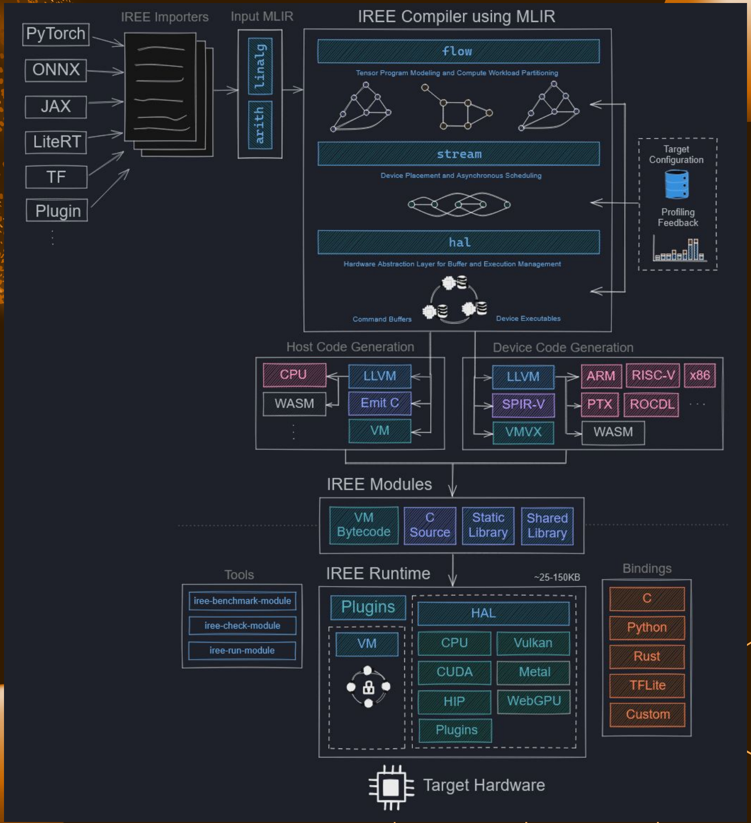


Fig. 1. What a modern ecosystem might look like. Apache TVM. (https://tvm.apache.org/2017/10/06/nnvm-compiler-announcement).

IREE: Path From Software To Hardware



Two Language Problem

Python

High level, easy
SUPER SLOW
Calls into C++/ etc.

C++ / CUDA / PTX

Requires good
programming skills,
powerful, tedious

FLOPs go BRRR

New Hardware

TPUs, LPUs	Google, Groq
Intel Loichi	SNNs, not tensors
Who knows?	We should future proof!

How has monopoly shaped our industry?

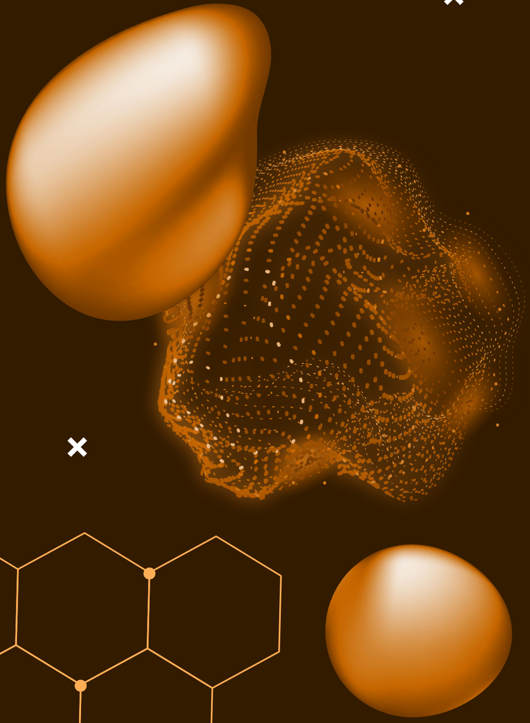
Lack of competition

Shared space

02

“The Hardware Lottery”

How has hardware shaped research?



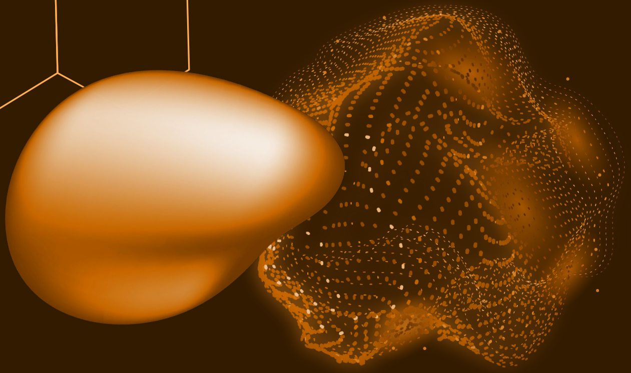
> “This essay introduces the term hardware lottery to describe when a research idea wins because it is suited to the available software and hardware and not because the idea is superior to alternative research directions”

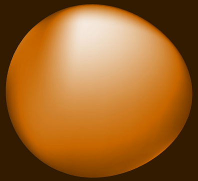
—Sara Hooker, Google Brain 2020

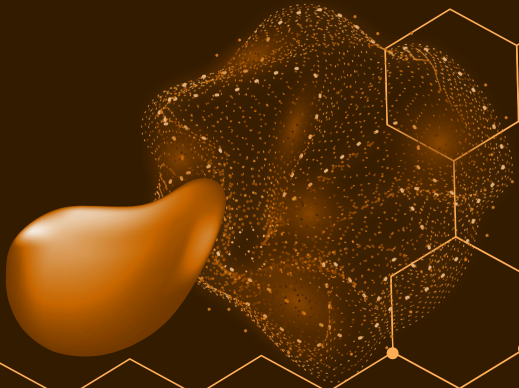
03

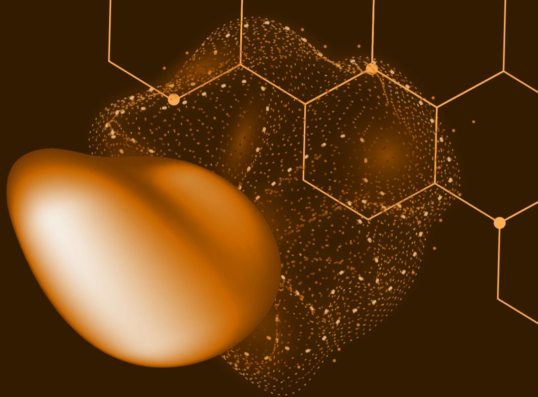
Current State

How do we deal with this now?



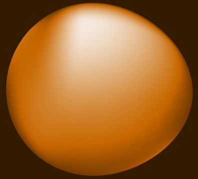






Compilers

IRs like MLIR, Auto-Tuning

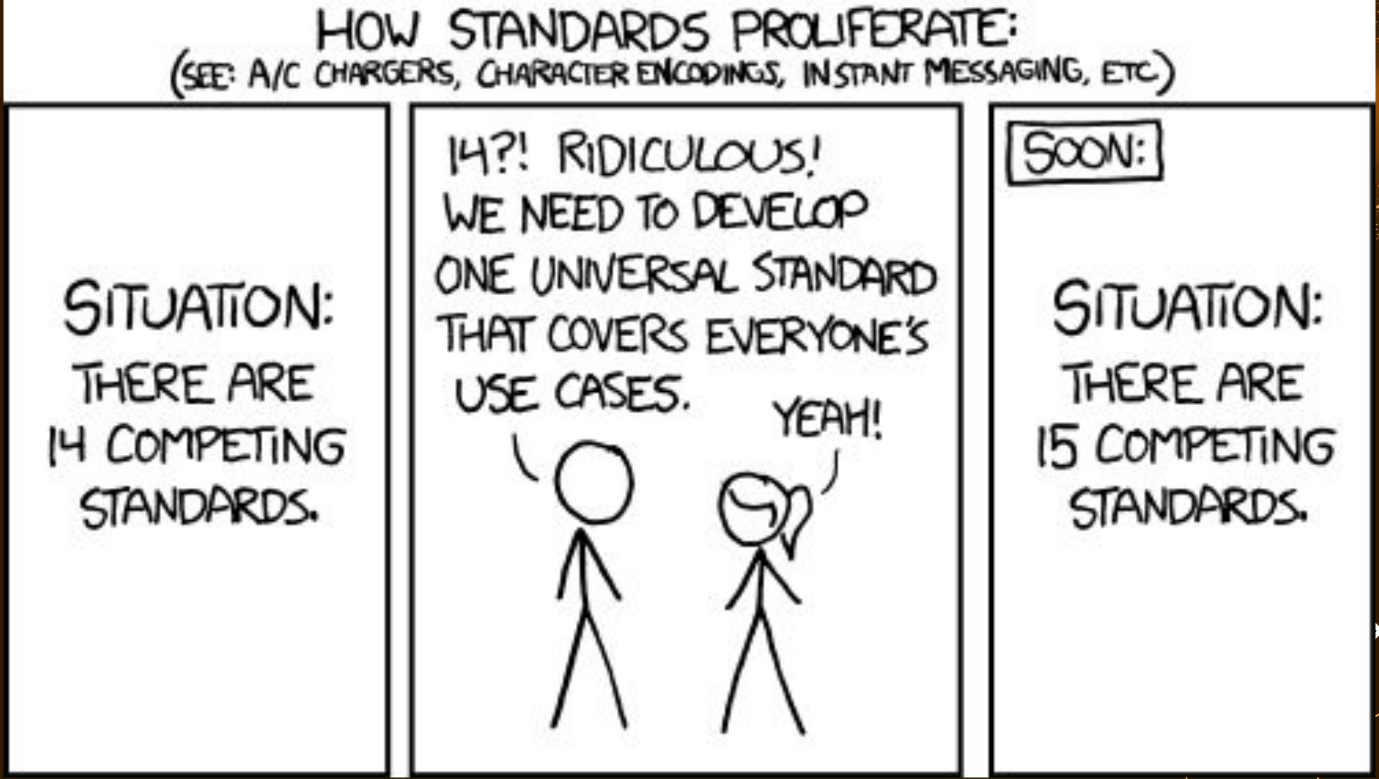


[e]DSLs

Domain Specific Languages like Triton

Open Source

Design by committee failures, coordination
failures, fragmentation

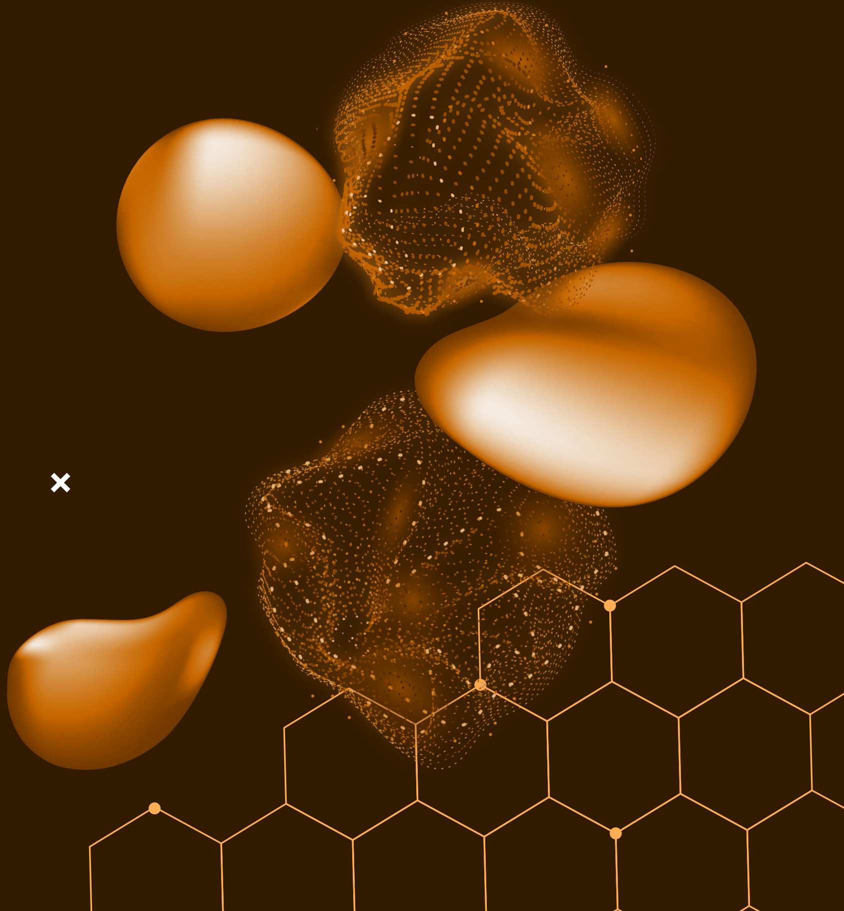

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

This Means Vendor Lock-In

PTX for a single Nvidia GPU

OpenCL example - no Tensor Core support

Switching frameworks



04

Modular / MAX / Mojo

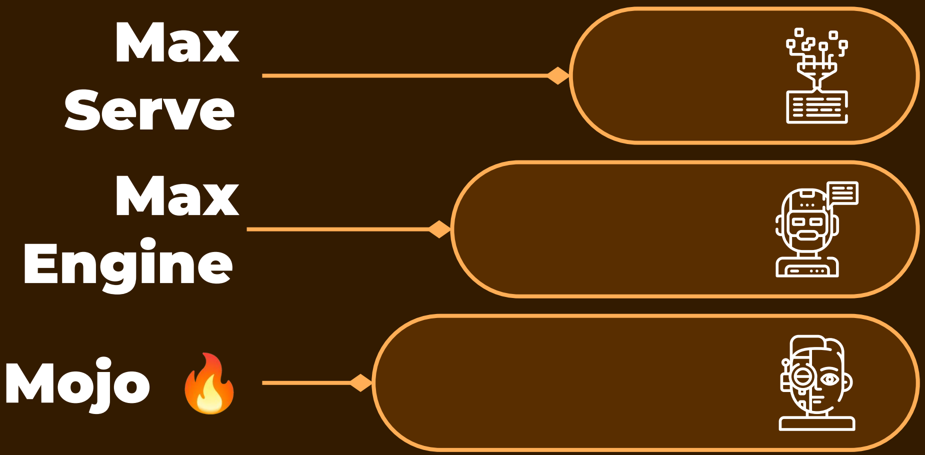
Chris Lattner

Chris Lattner

2000	**LLVM**	IR, Clang
2010	**Swift**	Replaced Obj-C
2017	**TF & TPU**	Google

Compilers **Languages** **Hardware**

Modular



Scheduling, Batching, Cloud or Local Deployment

Max Serve

Graph compilers, libraries, tools

Max Engine

Python Superset - FAST

Mojo 🔥

Mojo🔥

Python Superset

No need to learn C++ / Rust / CUDA. Use Python libraries!

MLIR Coherency

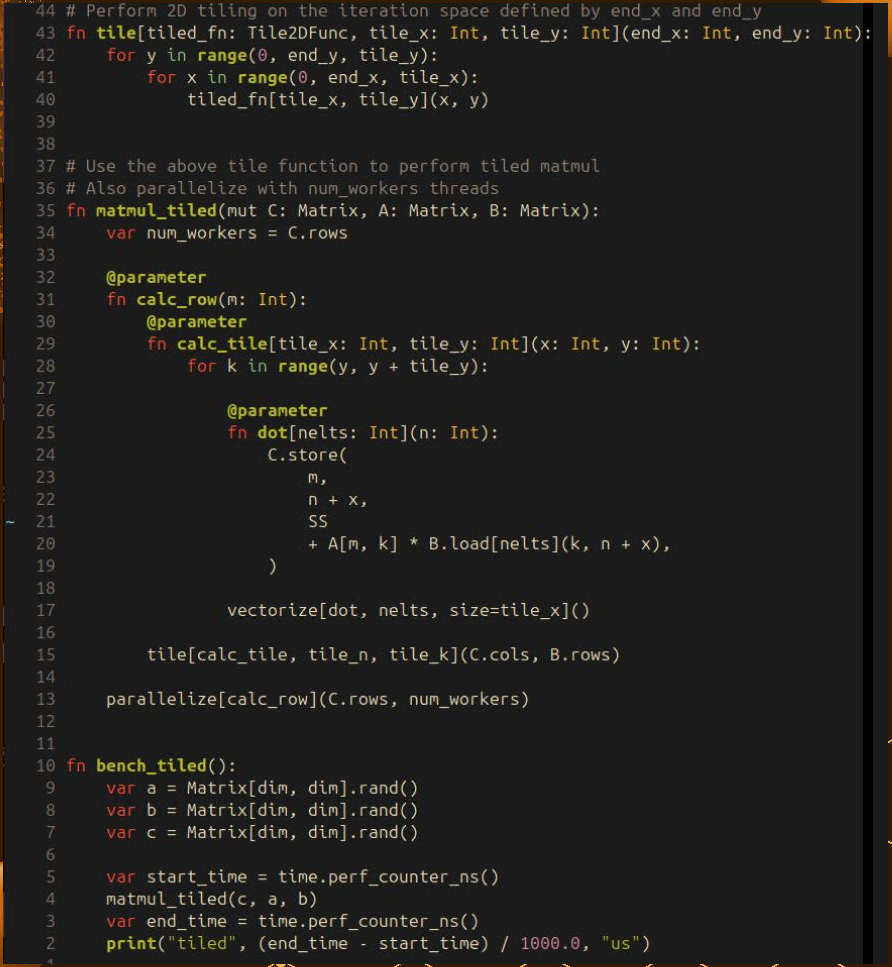
Optimizations are easy across ALL stages

Typed, Compiled!

“Up to 65,000x faster than Python”

Easy to write code! Performant!

Auto vectorizing
Auto parallelizing
Native SIMD
Easy tiling
GPU support just added
Only *slightly* verbose

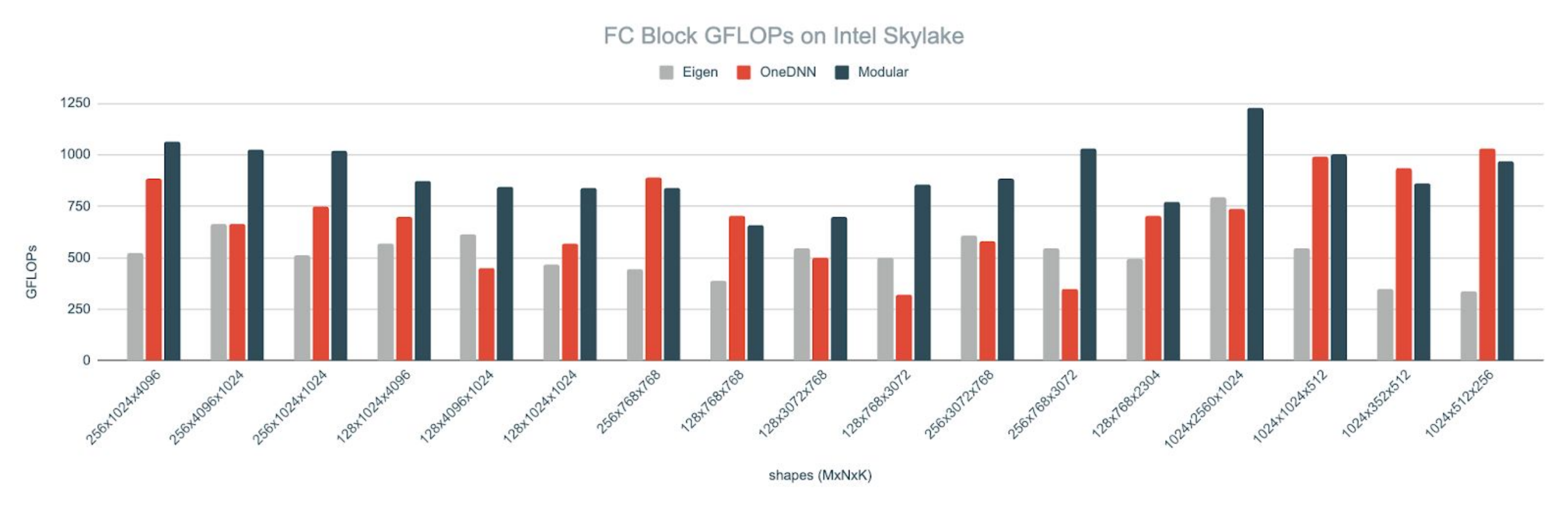


```
# Perform 2D tiling on the iteration space defined by end_x and end_y
fn tile[tiled_fn: Tile2DFunc, tile_x: Int, tile_y: Int](end_x: Int, end_y: Int):
    for y in range(0, end_y, tile_y):
        for x in range(0, end_x, tile_x):
            tiled_fn[tile_x, tile_y](x, y)


# Use the above tile function to perform tiled matmul
# Also parallelize with num_workers threads
fn matmul_tiled(mut C: Matrix, A: Matrix, B: Matrix):
    var num_workers = C.rows

    @parameter
    fn calc_row(m: Int):
        @parameter
        fn calc_tile[tile_x: Int, tile_y: Int](x: Int, y: Int):
            for k in range(y, y + tile_y):

                @parameter
                fn dot[nelts: Int](n: Int):
                    C.store(
                        m,
                        n + x,
                        SS
                        + A[m, k] * B.load[nelts](k, n + x),
                    )

                vectorize[dot, nelts, size=tile_x]()

        tile[calc_tile, tile_n, tile_k](C.cols, B.rows)

    parallelize[calc_row](C.rows, num_workers)


fn bench_tiled():
    var a = Matrix[dim, dim].rand()
    var b = Matrix[dim, dim].rand()
    var c = Matrix[dim, dim].rand()

    var start_time = time.perf_counter_ns()
    matmul_tiled(c, a, b)
    var end_time = time.perf_counter_ns()
    print("tiled", (end_time - start_time) / 1000.0, "us")
```

Matrix Multiplication 1024 x 1024

Language / Compiler	Notes	MicroSeconds
Python 3.12	Naive	50,800,000
Python 3.12	Numpy, Transpose	1,175,000
Mojo	Vectorized, Parallelized	8,300
C / Zig CC	Naive	1,950,000
C / Zig CC	-O3, OpenMP, Transpose	27,000

SOTA: MatMul & Relu



Critiques

“Ownership”

This memory management paradigm can be tough

New Keywords

Mojo is “Python ++” & WIP

Parameters

Separation of comptime is new to most

MAX System

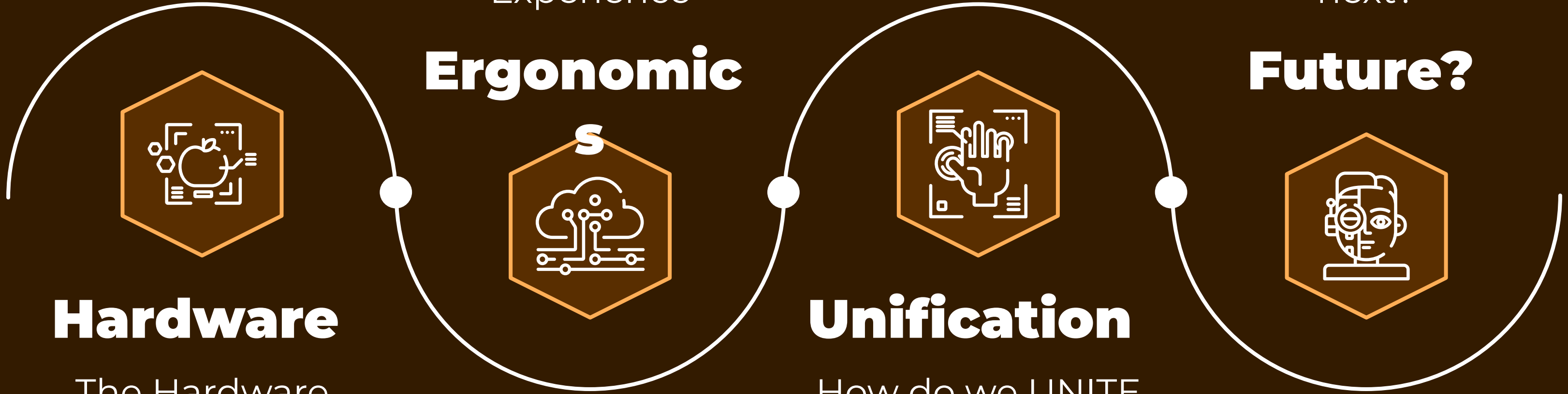
Graph building here is a new system / library to learn

05

Takeaways

Chris Lattner-senapai notice me :3 UwU

Takeaways



Hardware

The Hardware Lottery

Ergonomics

Developer Experience

Unification

How do we UNITE research?

Future?

What’s coming next?

THANKS!

DO YOU HAVE ANY QUESTIONS?

paul.dutton@my.utsa.edu

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

References

Sara Hooker, “The Hardware Lottery”. 2020. https://arxiv.org/abs/2009.06489

https://tvm.apache.org/2017/10/06/nnvm-compiler-announcement

https://iree.dev/#project-architecture

https://xkcd.com/927/

Modular: The world's fastest unified matrix multiplication

https://static.wikia.nocookie.net/monopoly/images/4/41/Monopoly_2D_Art_Render.png/revision/latest?cb=20220109225628