Inference

Model to Metal

@steeve


TRINITRON
PLAYER SELECT
U.S.S.R.
CHINA
JAPAN
U.S.A.
U.S.A.
INDIA
JAPAN
1P
2P
BRAZIL
TRAINING
INFERENCE
2P
1P
POWER
CONTROL
SONY

TRAINING

Research

One of something

More is better

Fast iteration

Python = ❤️

INFERENCE

Production

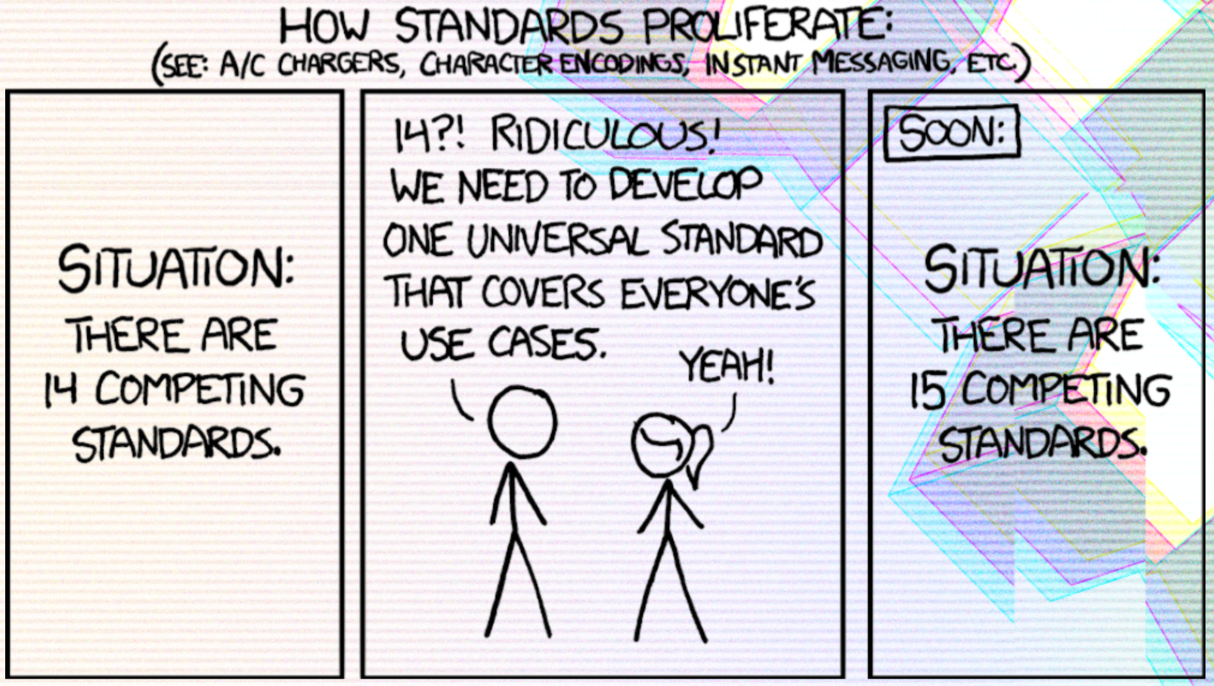
Billions of somethings

Less is better

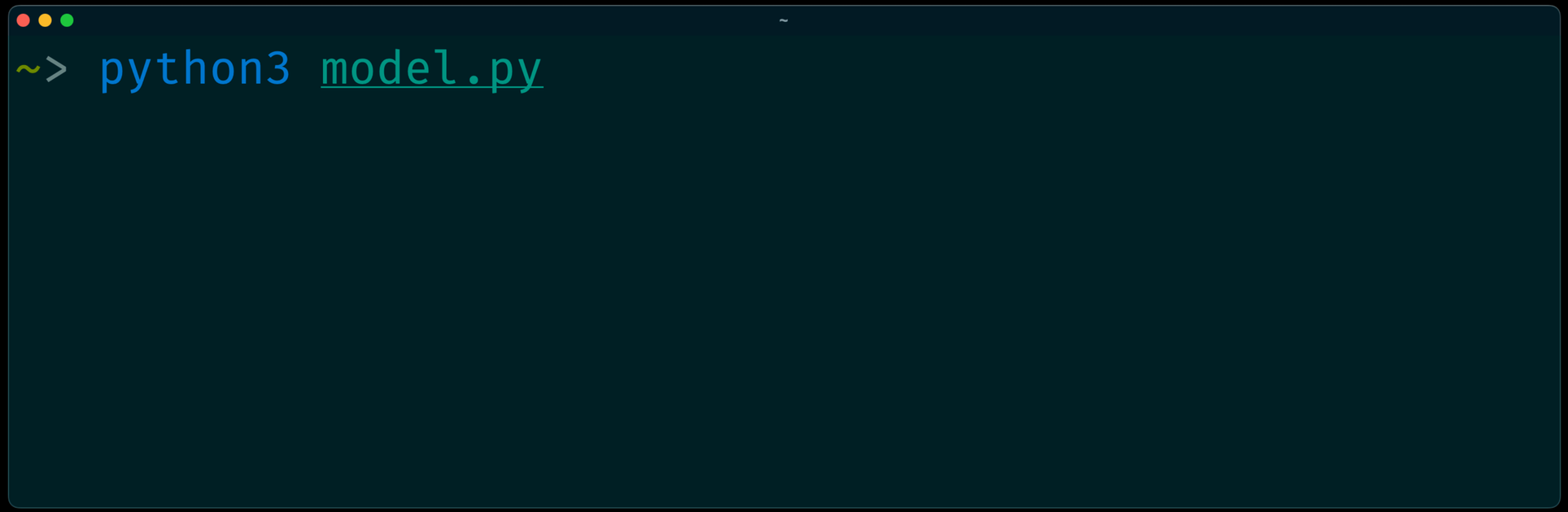
Predictable latency

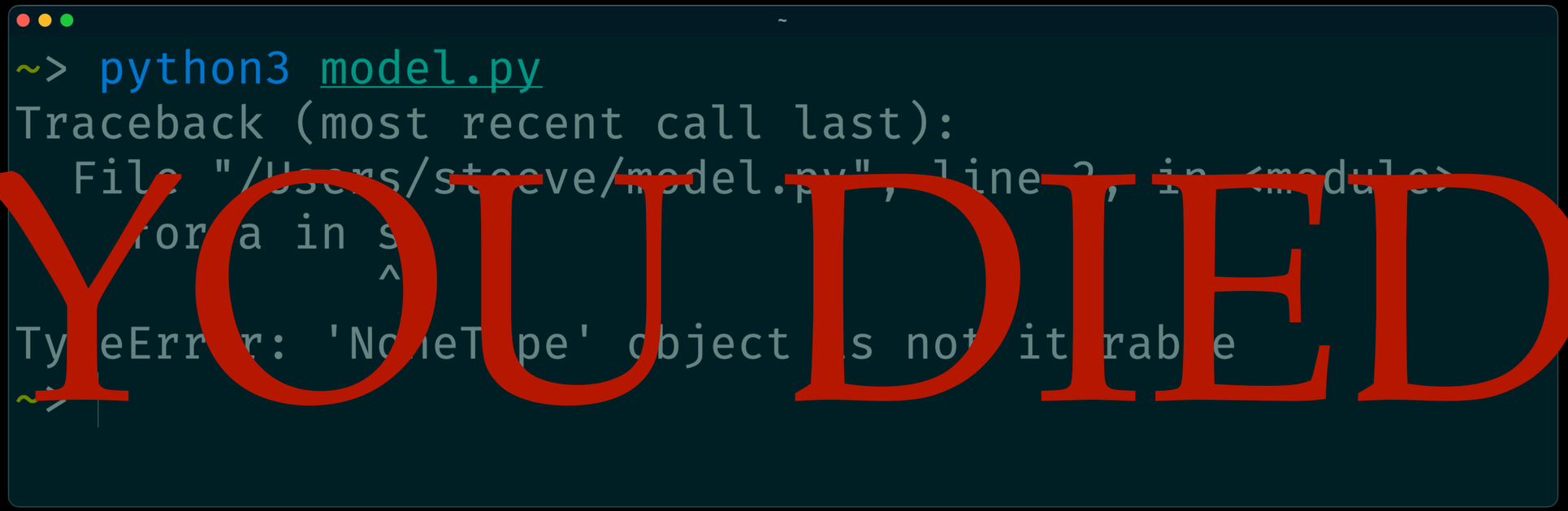
Python = 🫠

Why do we need another framework?




TRINITRON
CONTINUE? 7
THEY SAY INFERENCE IS THE BEST,
BUT THAT CHANGES THE MOMENT
YOU TRAIN .
POWER
CONTROL
SONY

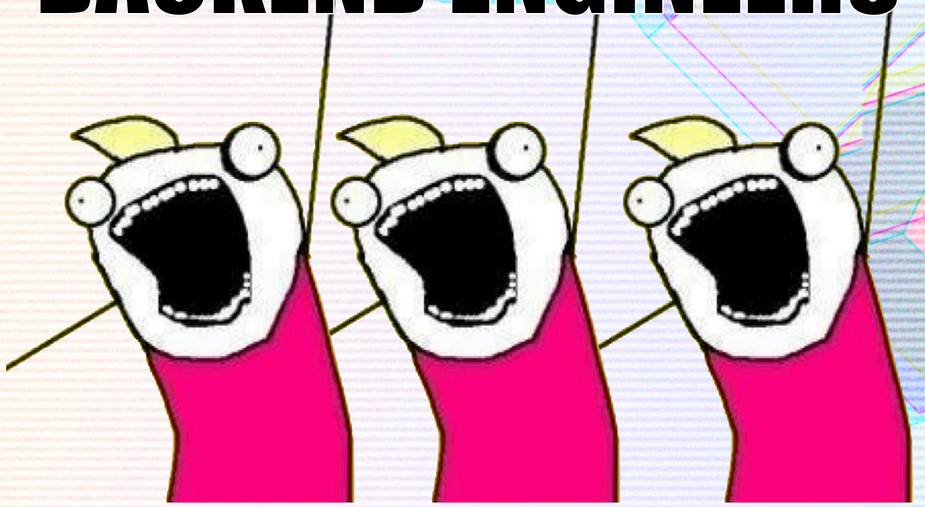

~
~> python3 model.py



YOU DIED

WHO ARE WE ?

AI FLAVORED
BACKEND ENGINEERS



WHAT DO WE WANT ??

ACCELERATOR AGNOSTICITY

CROSS COMPILING

RUNTIME DISTROLESS

SANDBOXING

PARALLEL IO

MODEL COMPILATION

K8S

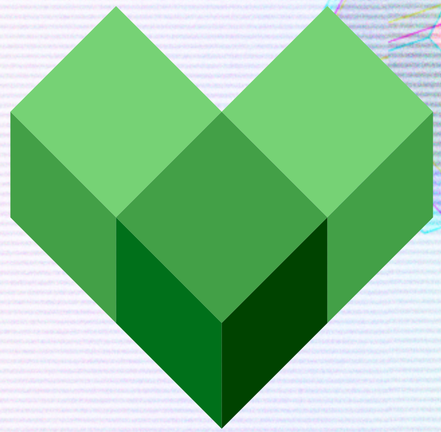
STATIC TYPING


ZML

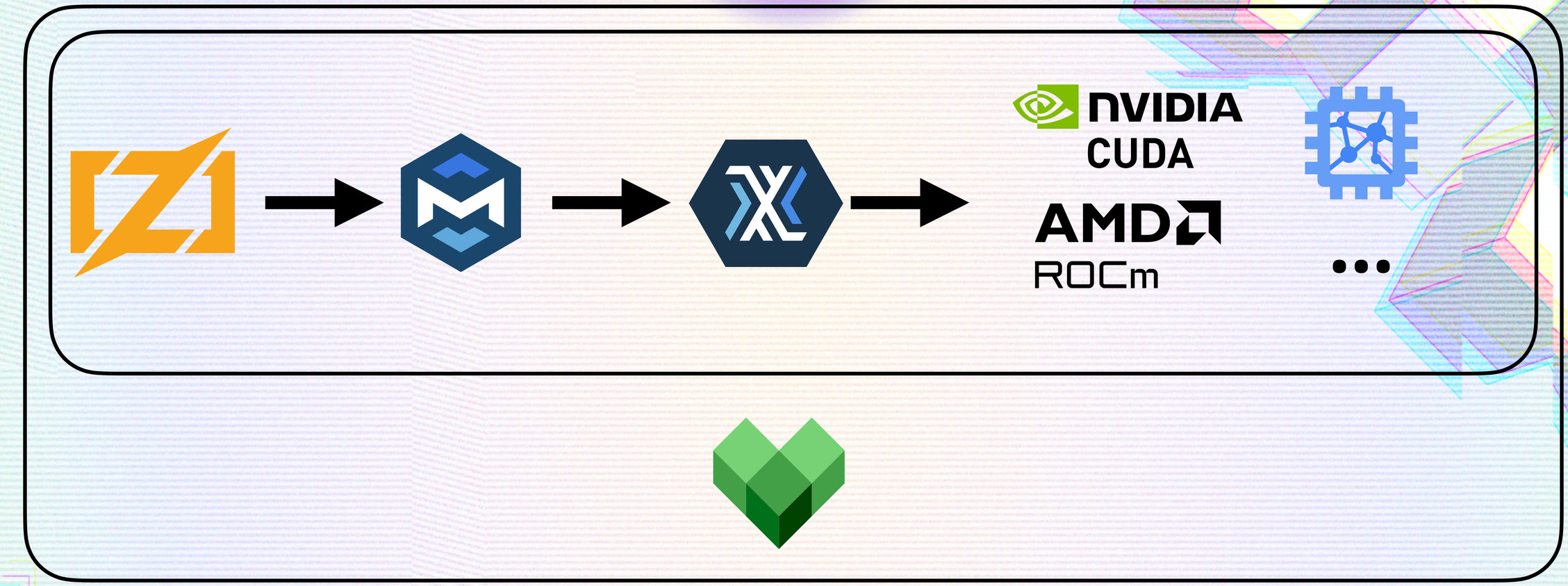
ZIG

MLIR

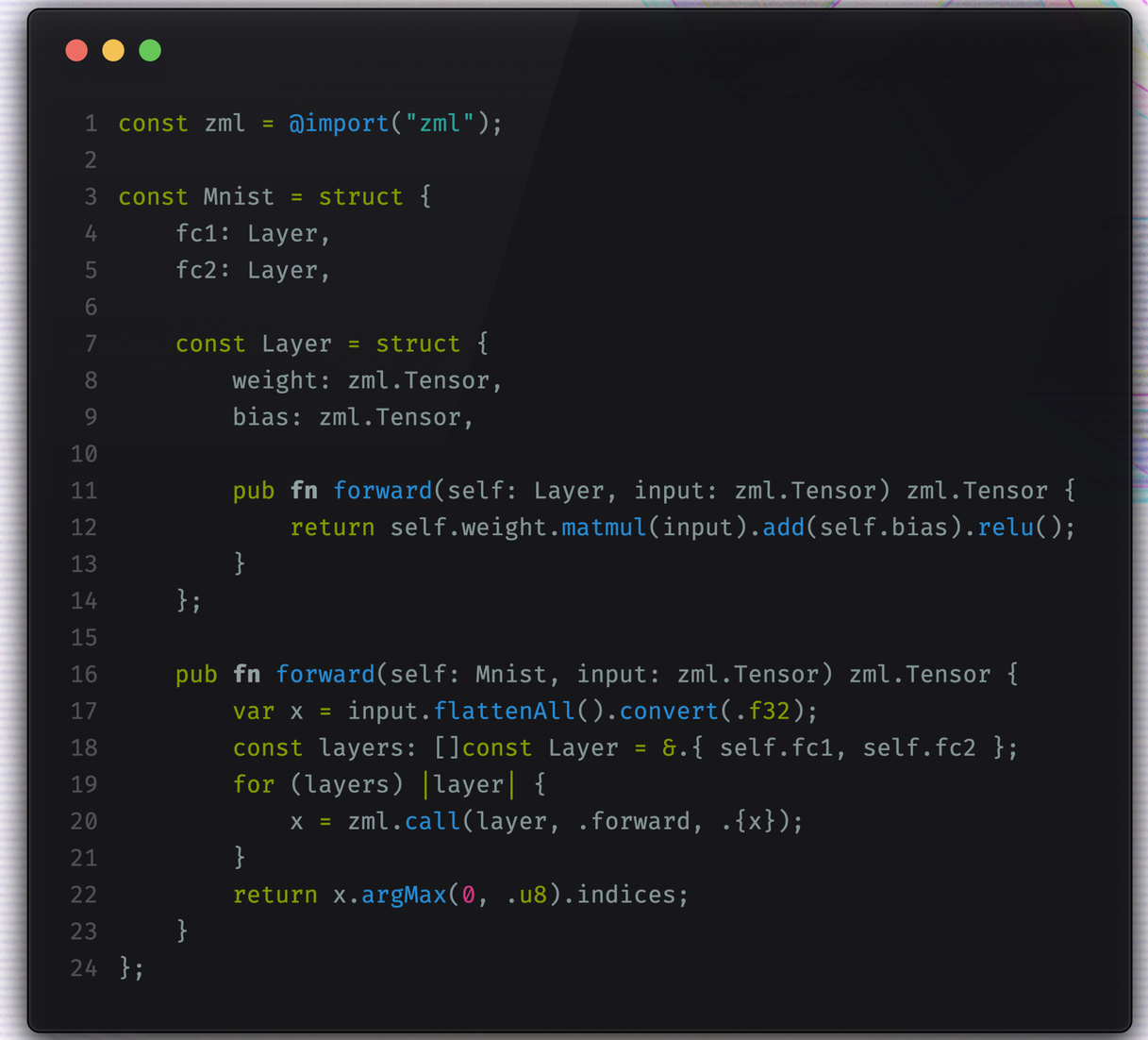
OPENXLA



BAZEL


NVIDIA
CUDA
AMD
ROCm
...

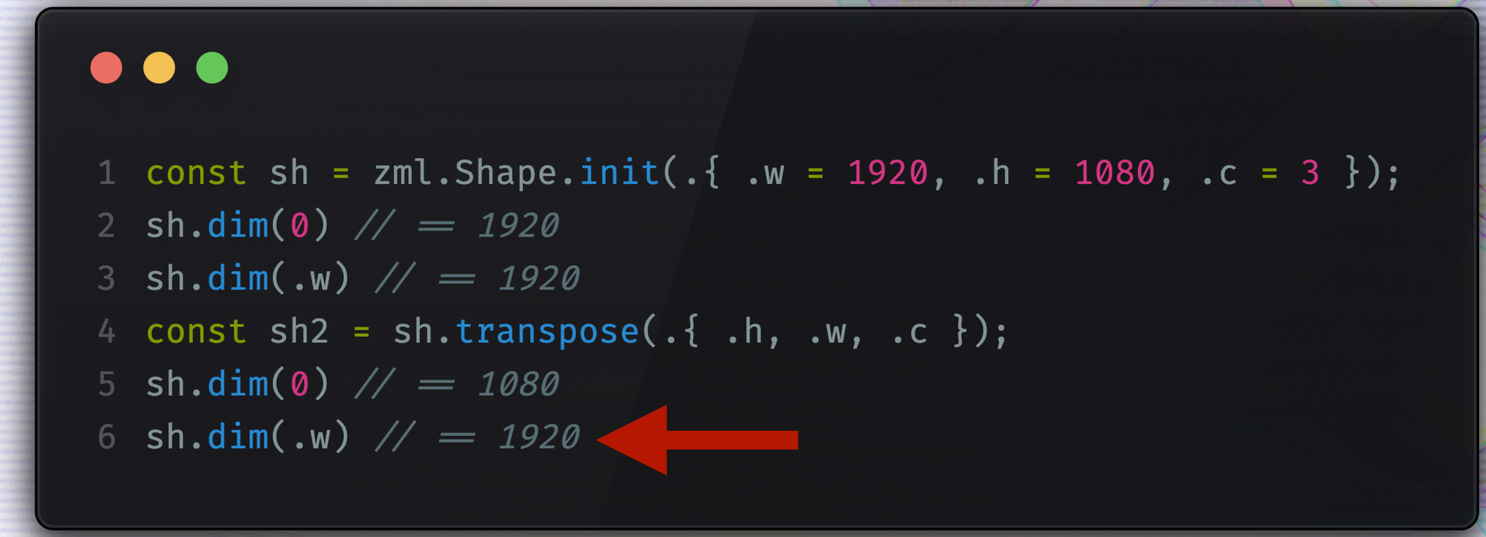

Inference *only*
Zig as the frontend
Zero Python
Modular code
Focus on maintainability



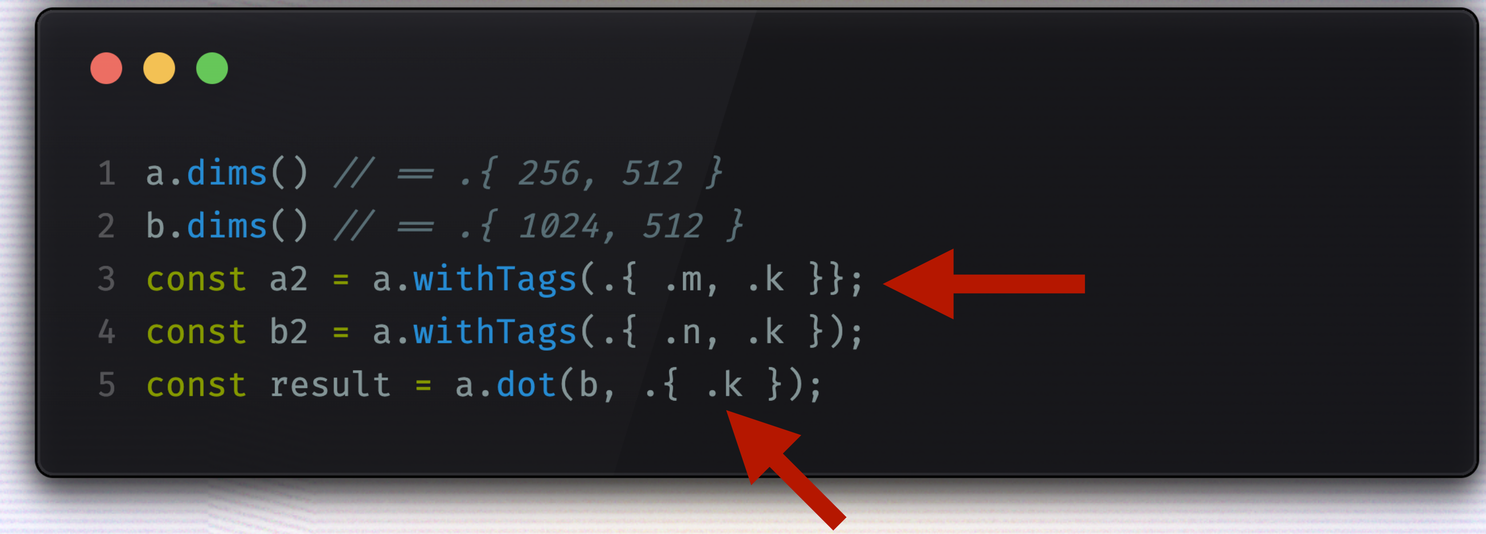
```zig
const zml = @import("zml");

const Mnist = struct {
    fc1: Layer,
    fc2: Layer,

    const Layer = struct {
        weight: zml.Tensor,
        bias: zml.Tensor,

        pub fn forward(self: Layer, input: zml.Tensor) zml.Tensor {
            return self.weight.matmul(input).add(self.bias).relu();
        }
    };

    pub fn forward(self: Mnist, input: zml.Tensor) zml.Tensor {
        var x = input.flattenAll().convert(.f32);
        const layers: []const Layer = &.{ self.fc1, self.fc2 };
        for (layers) |layer| {
            x = zml.call(layer, .forward, .{x});
        }
        return x.argMax(0, .u8).indices;
    }
};
```

Axis tagging
Propagated to IR/compiler
Much less tensor formatting
Who needs matmul anyway?

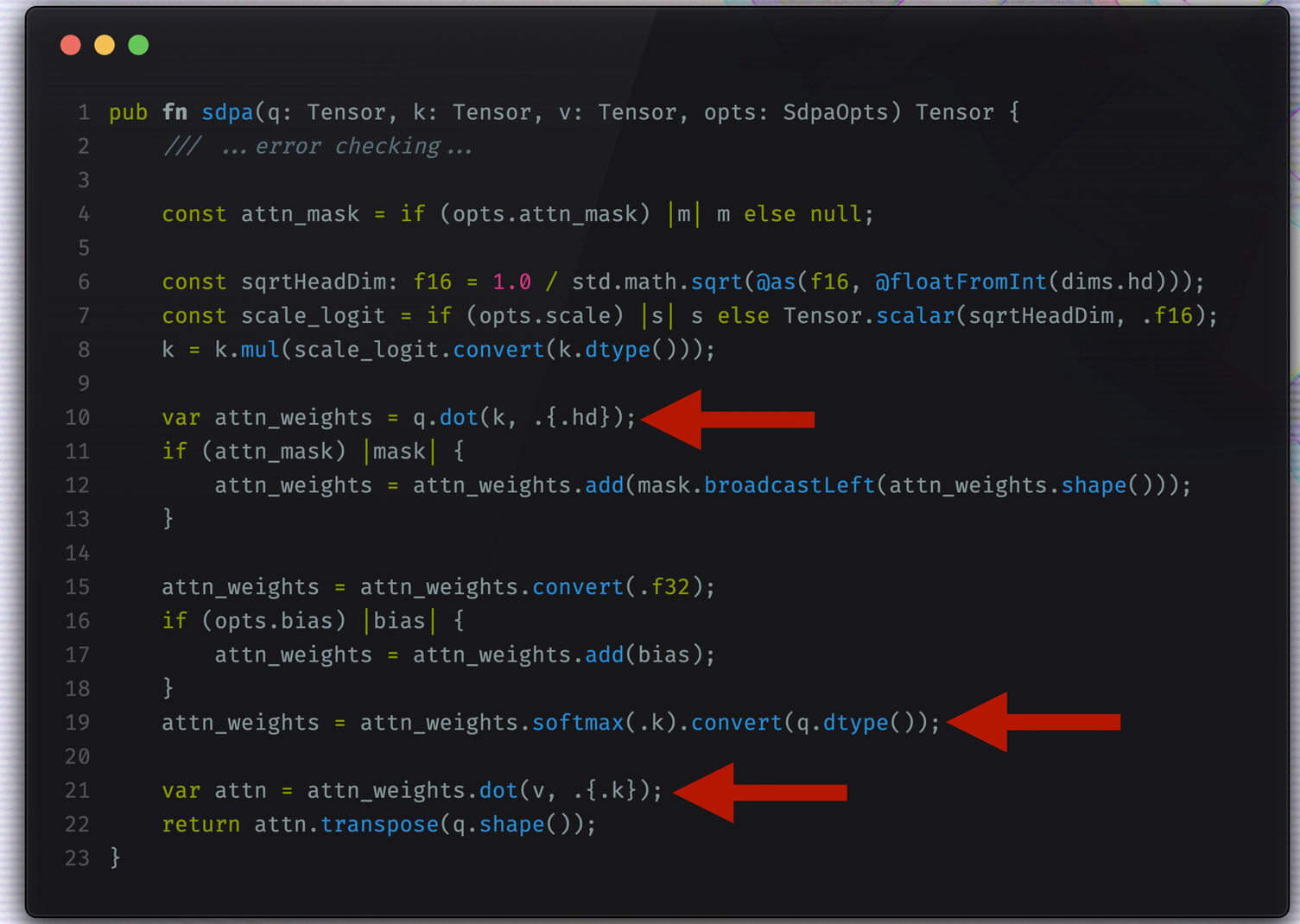


```zig
const sh = zml.Shape.init(.{ .w = 1920, .h = 1080, .c = 3 });
sh.dim(0) // == 1920
sh.dim(.w) // == 1920
const sh2 = sh.transpose(.{ .h, .w, .c });
sh.dim(0) // == 1080
sh.dim(.w) // == 1920
```



```zig
a.dims() // == .{ 256, 512 }
b.dims() // == .{ 1024, 512 }
const a2 = a.withTags(.{ .m, .k });
const b2 = a.withTags(.{ .n, .k });
const result = a.dot(b, .{ .k });
```


pub fn sdpa(q: Tensor, k: Tensor, v: Tensor, opts: SdpaOpts) Tensor {
/// ...error checking...
const attn_mask = if (opts.attn_mask) |m| m else null;
const sqrtHeadDim: f16 = 1.0 / std.math.sqrt(@as(f16, @floatFromInt(dims.hd)));
const scale_logit = if (opts.scale) |s| s else Tensor.scalar(sqrtHeadDim, .f16);
k = k.mul(scale_logit.convert(k.dtype()));
var attn_weights = q.dot(k, .{.hd});
if (attn_mask) |mask| {
attn_weights = attn_weights.add(mask.broadcastLeft(attn_weights.shape()));
}
attn_weights = attn_weights.convert(.f32);
if (opts.bias) |bias| {
attn_weights = attn_weights.add(bias);
}
attn_weights = attn_weights.softmax(.k).convert(q.dtype());
var attn = attn_weights.dot(v, .{.k});
return attn.transpose(q.shape());
}

OpenXLA Founding Members

NVIDIA

AMD

Alibaba Group
阿里巴巴集团

arm

GRAPHCORE

intel

Hugging Face

cerebras

Meta

Google

amazon
web services

Apple

XLA optimizing compiler

Kernel fusion

Coalesced memory allocation

Autotuning

Mature and battle tested IR



Cross compiling

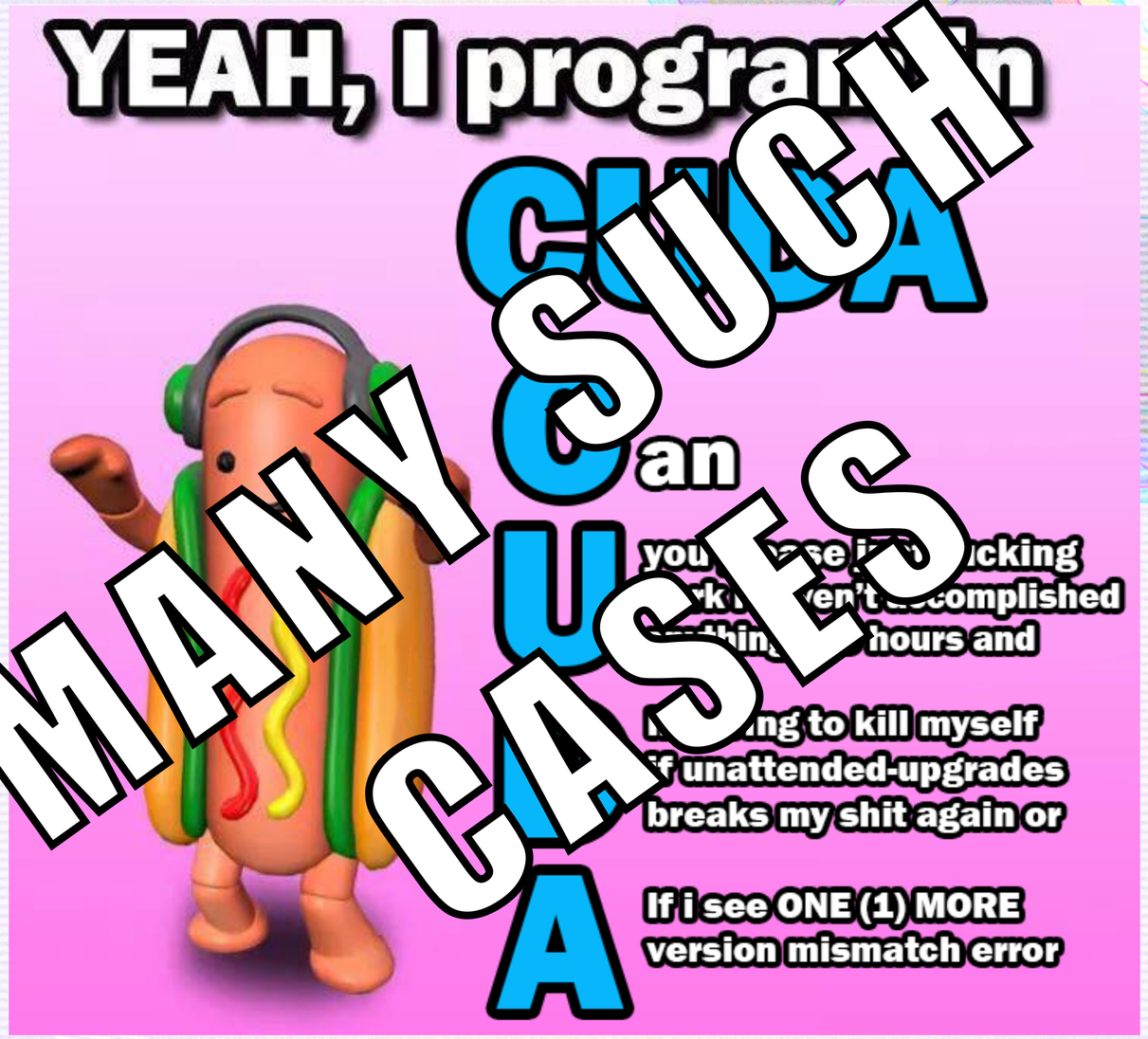
Runtime trimming/sandboxing

Self deployable archives

OCI images

K8S deployments



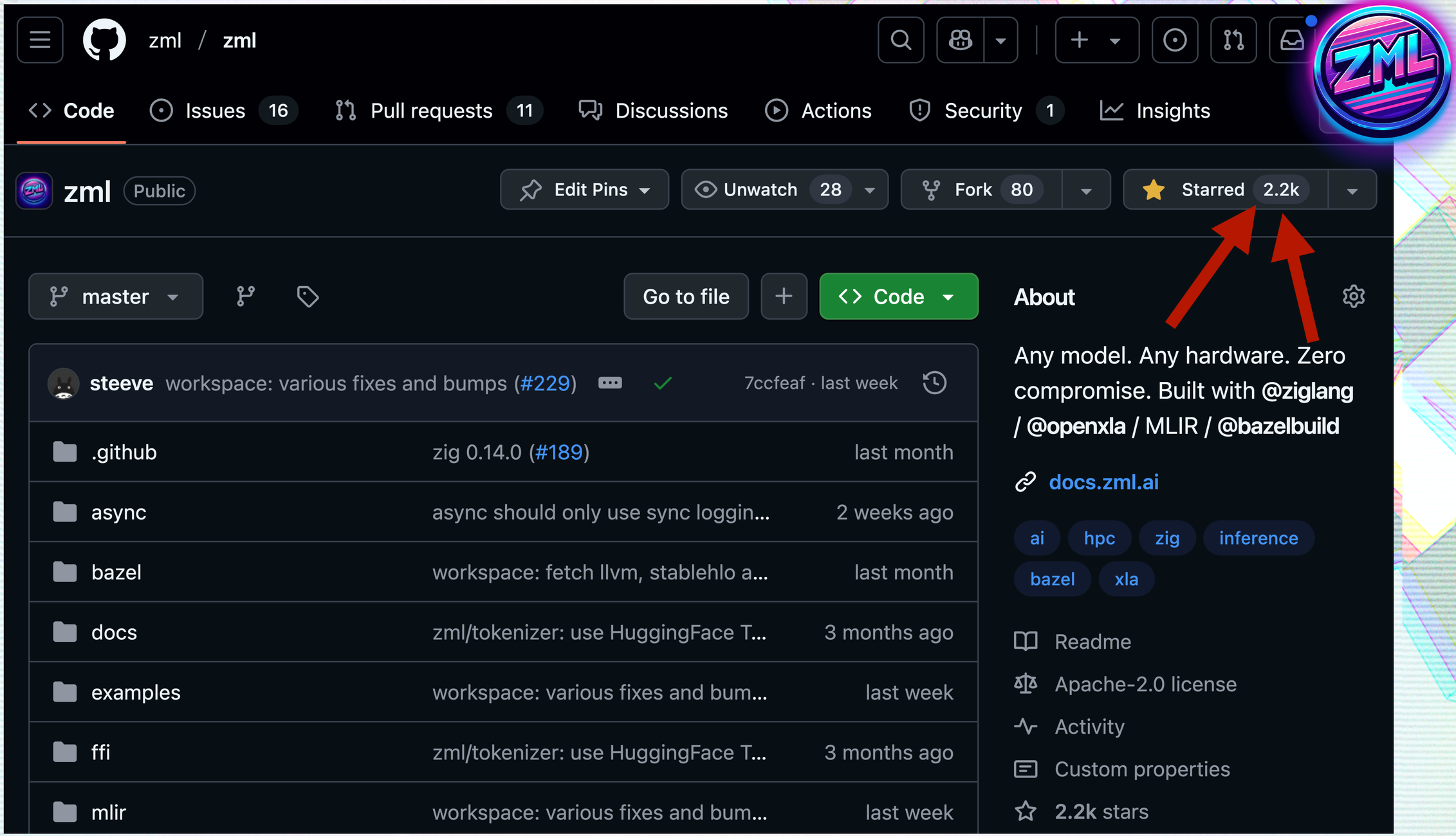

YEAH, I program in
CUDA
an
you
cking
hours and
ng to kill myself
f unattended-upgrades
breaks my shit again or
If i see ONE (1) MORE
version mismatch error
MANY SUCH CASES


PROVISIONING? Where we're going we don't need PROVISIONING

```
$ bazel run -c opt //mnist --@zml//runtimes:cuda=true
```

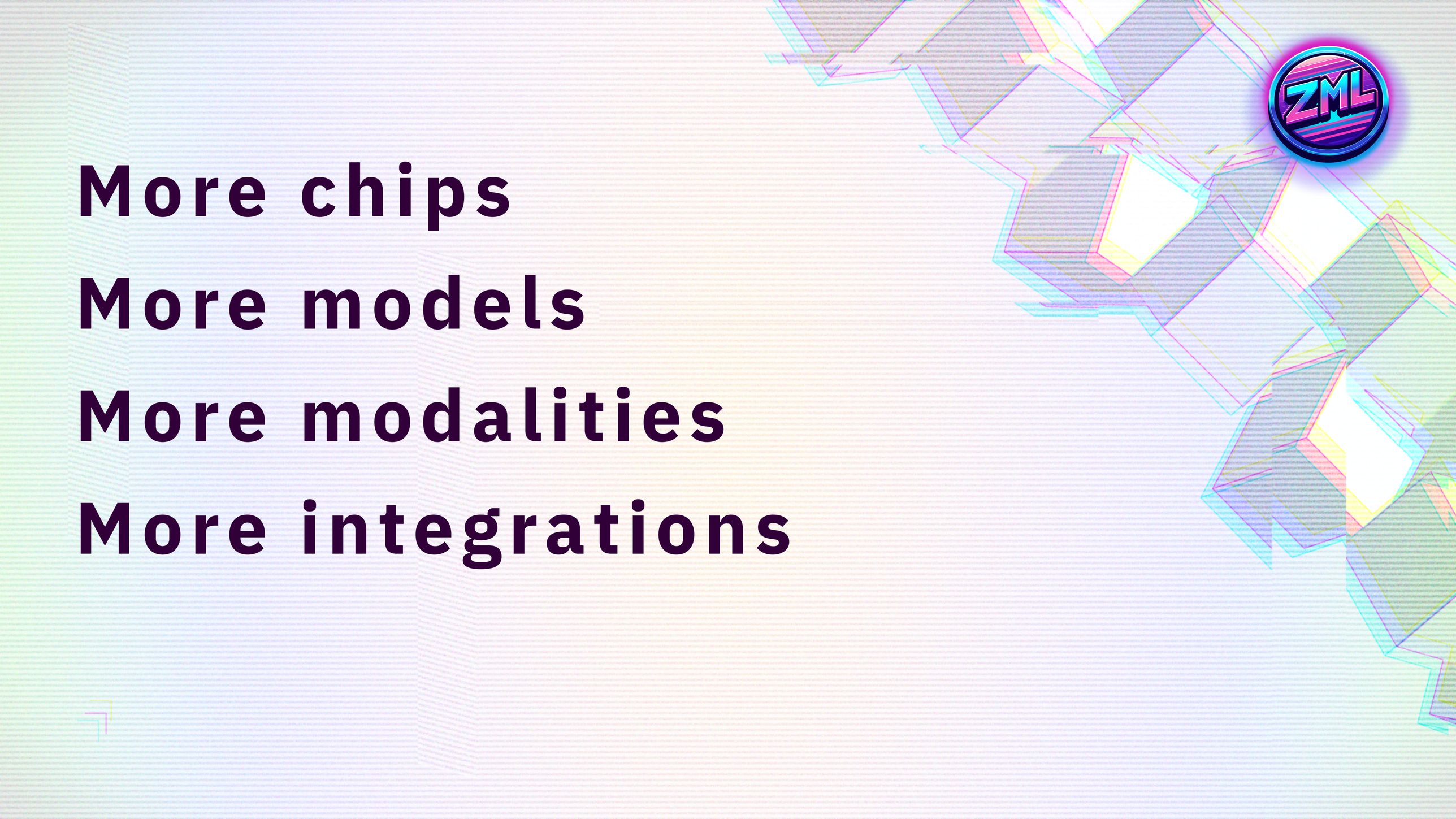
```
$ bazel build -c opt //mnist:archive --@zml//runtimes:rocm=true --platforms=@zml//:linux_amd64
```

```
$ bazel run -c opt //mnist:push --@zml//runtimes:cuda=true --@zml//runtimes:tpu=true
```


zml / zml
Code
Issues 16
Pull requests 11
Discussions
Actions
Security 1
Insights
zml Public
Edit Pins
Unwatch 28
Fork 80
Starred 2.2k
master
Go to file
Code
steeve workspace: various fixes and bumps (#229) 7ccfeaf · last week
.github zig 0.14.0 (#189) last month
async async should only use sync loggin... 2 weeks ago
bazel workspace: fetch llvm, stablehlo a... last month
docs zml/tokenizer: use HuggingFace T... 3 months ago
examples workspace: various fixes and bum... last week
ffi zml/tokenizer: use HuggingFace T... 3 months ago
mlir workspace: various fixes and bum... last week
About
Any model. Any hardware. Zero compromise. Built with @ziglang / @openxla / MLIR / @bazelbuild
docs.zml.ai
ai hpc zig inference bazel xla
Readme
Apache-2.0 license
Activity
Custom properties
2.2k stars


Yann LeCun
@ylecun
ZML: a high-performance AI inference stack that can parallelize and run deep learning systems on lots of different hardware.
It's out of stealth, impressive, and open source.
ZML @zml_ai · Sep 17
github.com/zml/zml
3:11 PM · Sep 17, 2024 · 204.9K Views
27
259
1.7K
895

What’s next ?



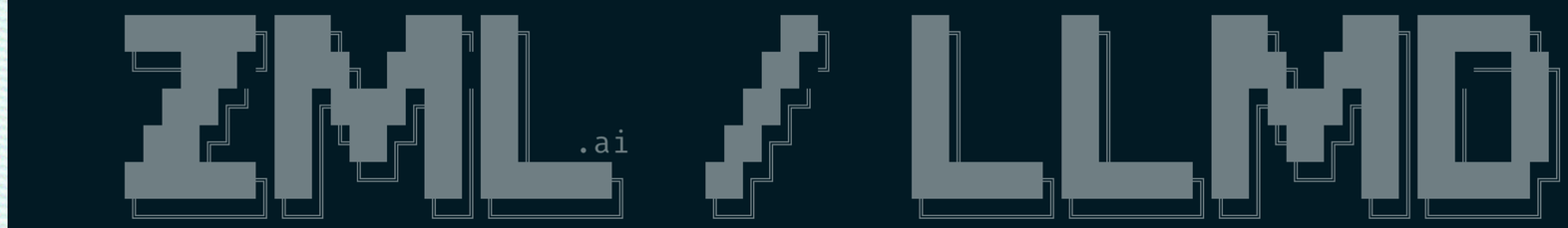
More chips

More models

More modalities

More integrations

Serving Stack




```
info(pjrt): Loaded library: libpjrt_cpu.so
info(pjrt): Loaded library: libpjrt_rocm.so
2025-04-28 16:25:43.764975: I xla/service/service.cc:152] XLA service 0x1b3c60b0 initialized for platform ROCM (this do
2025-04-28 16:25:43.765008: I xla/service/service.cc:160]   StreamExecutor device (0): AMD Radeon Graphics, AMDGPU ISA
info(zml/context): Available Platforms:
info(zml/context):    •  cpu
info(zml/context):    •  cuda
info(zml/context):    ✅ rocm (AUTO-SELECTED)
info(zml/context):         ◦ #0: AMD Radeon Graphics
info(zml/context):    •  tpu
info(zml/context):    •  neuron
debug(zml/io): Loading shard: model-00004-of-00004.safetensors
debug(zml/io): Loading shard: model-00001-of-00004.safetensors
debug(zml/io): Loading shard: model-00002-of-00004.safetensors
debug(zml/io): Loading shard: model-00003-of-00004.safetensors
info(llmd):     Loading Llama weights from ../zml++huggingface+Meta-Llama-3.1-8B-Instruct/model.safetensors.index.json.
info(zml): Compiling llama.LlamaLM.forward with { Shape({s=64,u32}), Shape({s=64,u32}), Shape({s=64,u32}), meta.MapType
,hd=128,bf16}), .v = Shape({layer=32,b=64,k=256,h=8!,hd=128,bf16}), .layer_index = Shape({u32}) }, meta.MapType(tensor.
tablehlo.RngAlgorithm.Type.DEFAULT } }
info(zml): Compiling llama.LlamaLM.forward with { Shape({s=320,u32}), Shape({s=66,u32}), Shape({s=66,u32}), meta.MapTyp
!,hd=128,bf16}), .v = Shape({layer=32,b=64,k=256,h=8!,hd=128,bf16}), .layer_index = Shape({u32}) }, meta.MapType(tensor
stablehlo.RngAlgorithm.Type.DEFAULT } }
```

zmlai/llmd

By zmlai • Updated 3 months ago

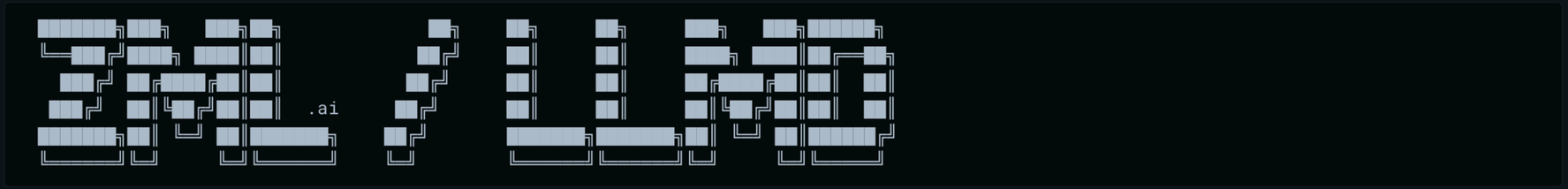
LLMD is a high performance LLM server, built by ZML.ai

IMAGE

LANGUAGES & FRAMEWORKS | MACHINE LEARNING & AI | WEB SERVERS

6 282

Overview Tags



LLMD - High Performance LLM Inference Server

This is a **technical preview** of LLMD, the ZML.ai homegrown high-performance LLM inference server. It is not intended for production use. Please report any issues to the ZML team.

LLMD is a work in progress and may change significantly in the future.
Some core features are purposedly disabled.

Get blazing-fast inference with a tiny 2.4GB container that works on both NVIDIA and AMD GPUs.

Preview Features

- **Easy Setup** - Just mount your model and run
- **Cross-Platform GPU Support** - Single container works on both NVIDIA and AMD GPUs
- **Lightweight** - Only 2.4GB container size
- **OpenAI API Compatible** - Drop-in replacement for OpenAI endpoints
- **High Performance** - Optimized for fast inference

Quick Start


TRY IT TODAY
$ git clone https://github.com/zml/zml
$ cd zml
$ bazel run //examples/llama \
--config=release \
--@zml//runtimes:cuda=true -- \
--hf-model-path=$HOME/Llama-3.1-8B-Instruct
\
--prompt="What is the capital of France?"
@zml_ai