WASMCON

BETTER TOGETHER

Machine Learning in Fastly's Compute@Edge

Andrew Brown (Intel), Matthew Tamayo-Rios (Fastly)

#WasmCon

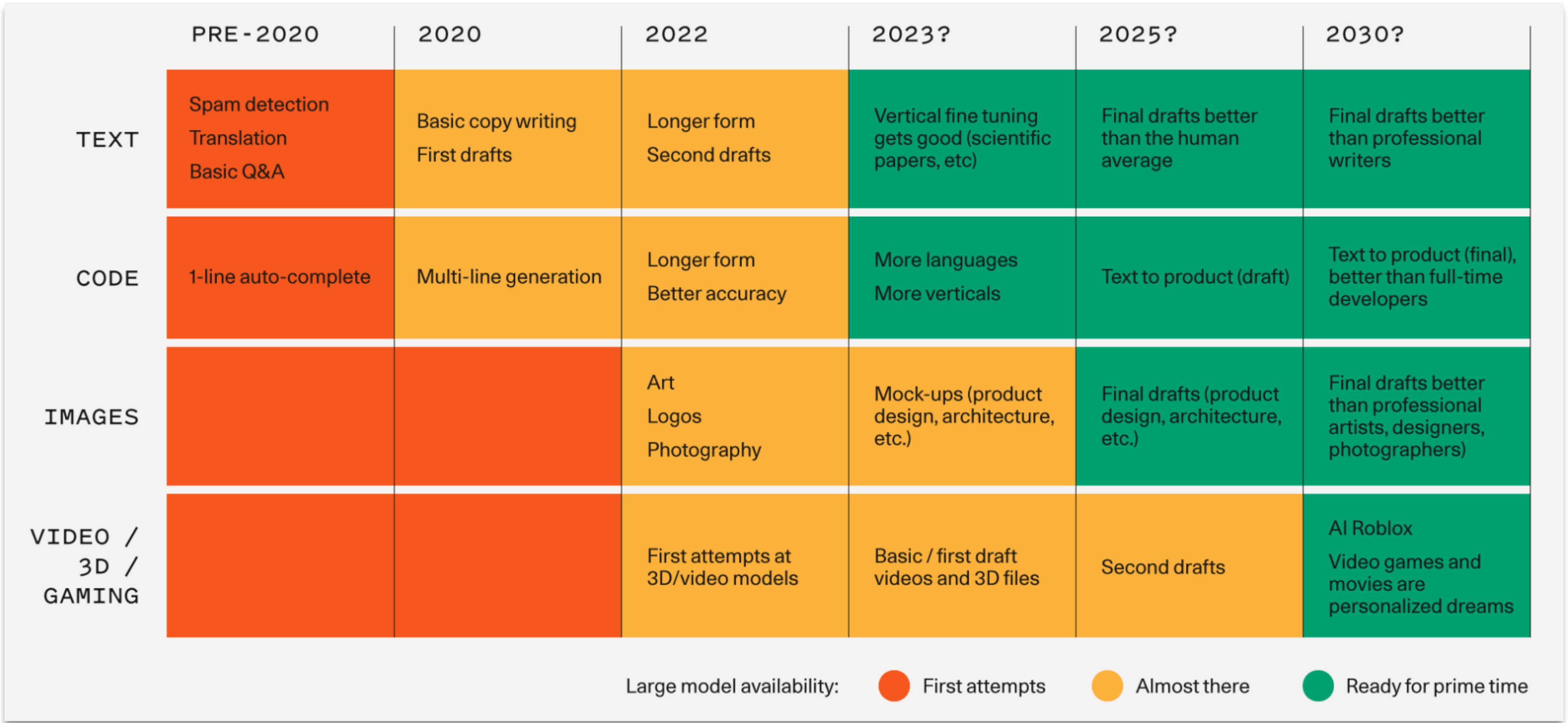
WASMCON
BETTER TOGETHER

Prompt: “AI will save the world, fire robot”

AI will “make everything we care about better.”

“Why AI Will Save the World” Mark Andreesen (June 2023)

Rise of Generative AI



	PRE-2020	2020	2022	2023?	2025?	2030?
TEXT	🟧 Spam detection Translation Basic Q&A	🟨 Basic copy writing First drafts	🟨 Longer form Second drafts	🟩 Vertical fine tuning gets good (scientific papers, etc)	🟩 Final drafts better than the human average	🟩 Final drafts better than professional writers
CODE	🟧 1-line auto-complete	🟨 Multi-line generation	🟨 Longer form Better accuracy	🟩 More languages More verticals	🟩 Text to product (draft)	🟩 Text to product (final), better than full-time developers
IMAGES	🟧	🟧	🟨 Art Logos Photography	🟨 Mock-ups (product design, architecture, etc.)	🟩 Final drafts (product design, architecture, etc.)	🟩 Final drafts better than professional artists, designers, photographers)
VIDEO / 3D / GAMING	🟧	🟧	🟨 First attempts at 3D/video models	🟨 Basic / first draft videos and 3D files	🟨 Second drafts	🟩 AI Roblox Video games and movies are personalized dreams

Large model availability: 🟧 First attempts 🟨 Almost there 🟩 Ready for prime time

"Generative AI: A Creative New World" Sequoia Capital (Sep. 2022)

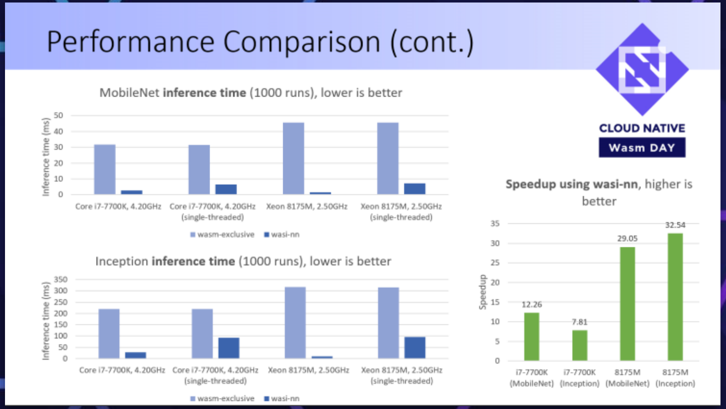
Agenda

- Explain wasi-nn
  - how do we efficiently use neural networks in Wasm?
- Build your own FaaS + ML
  - What is FaaS? How does it work?
  - How do we deploy and scale this simply and securely?
- Named models + process isolation
  - What are the key changes needed for performance and security?
- Demo
- Q&A

wasi-nn: why?

WASMCON
BETTER TOGETHER

PERFORMANCE


Performance Comparison (cont.)

HW NEEDED

- ✗ full-width SIMD
- ✗ special instructions (AMX, VNNI)
- ✗ GPUs, TPUs, NPUs…

FAST ECOSYSTEM

new models

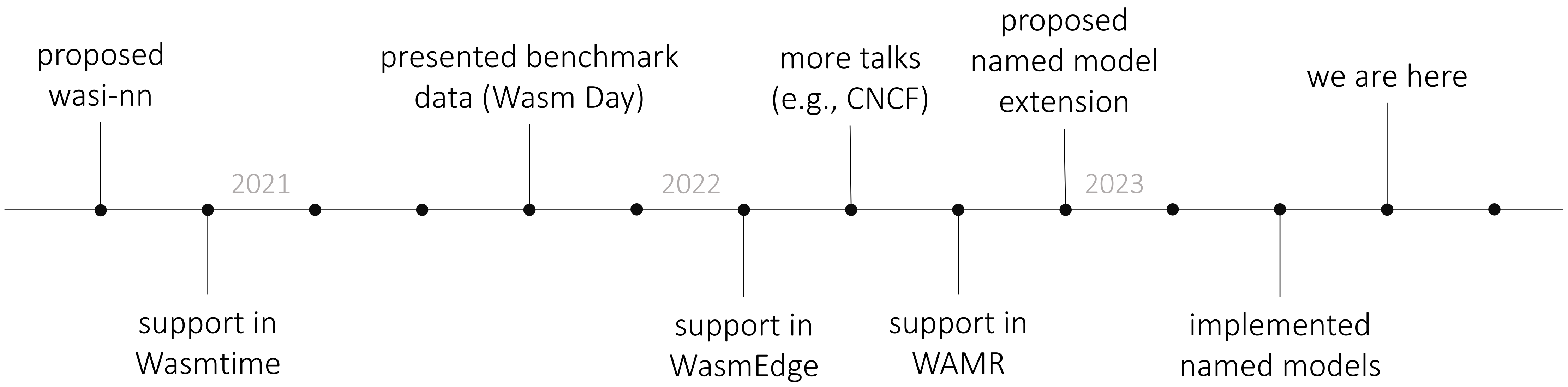
new operators

new tensor types

new model encodings

wasi-nn: an inference API using native ML capabilities for any ML framework

wasi-nn: evolution



wasi-nn: how does it work?

```rust
let graph = ...load([model, weights])?;
let mut context =
  graph.init_execution_context()?;
let input = ...;
context.set_input(0, input)?;
context.compute()?;
let mut output = vec![0f32; ...];
context.get_output(0, &mut output[..])?;
```

- defined in WITX
- high-level, model-builder (80%)
- spec → bindings → engines

recent work:
https://github.com/WebAssembly/wasi-nn

- defined in WIT
- simpler (#43, #48...)

```rust
let graph = ...load_by_name("foo")?;
let input = ...;
let output =  graph.compute([input])?;
```

Build your own FaaS + ML

WASMCON
BETTER TOGETHER

DEPLOYMENT

☑ need a registry!

☑ framework configuration

☑ model caching?

USER EXPERIENCE

☑ document available models

☑ tensor conversions

☑ explain framework errors

PERFORMANCE

☑ latency

☑ resource consumption

SECURITY

☑ protect infrastructure

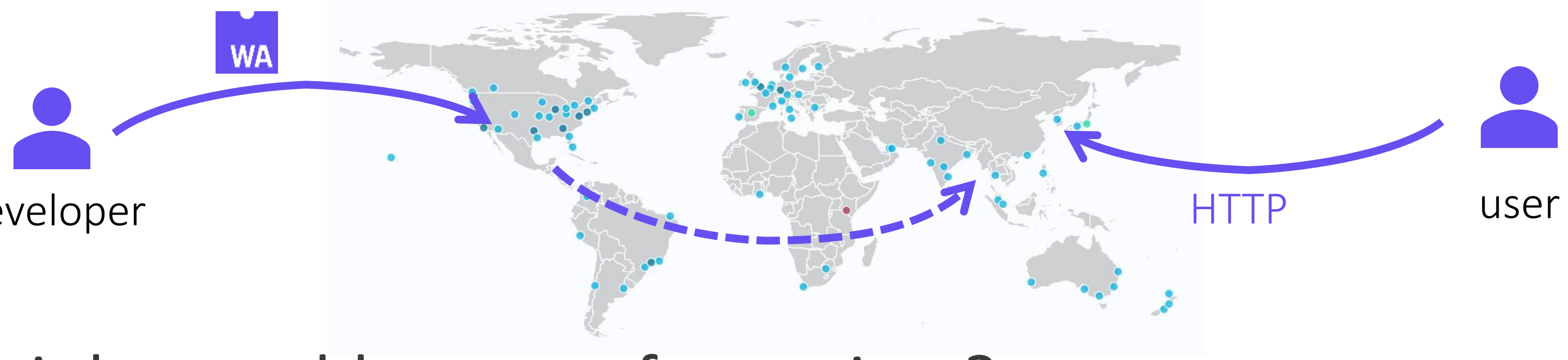
☑ protect other tenants

building blocks for FaaS + ML

Compute@Edge: what is it?

Fastly Compute@Edge makes it easy to securely deploy, run, and scale globally distributed WebAssembly modules.

- Serverless request based architecture so no need to manage infrastructure
- Run low latency use cases at the edge close to user

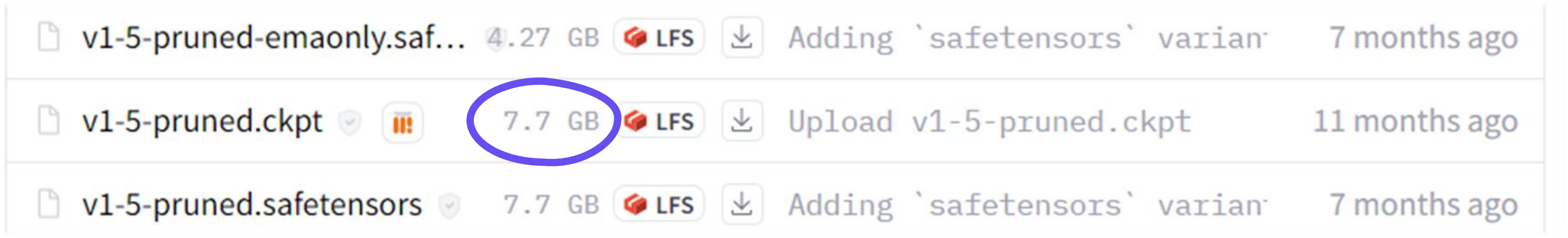


How hard can it be to add support for wasi-nn?

- Spoiler alert: it took some work…

Compute@Edge: considerations

- Wasmtime event loop is sensitive to expensive guest processes
- Models can exceed compute, memory, and module size limits



v1-5-pruned-emaonly.saf...	4.27 GB	LFS	Adding `safetensors` varian	7 months ago
v1-5-pruned.ckpt	7.7 GB	LFS	Upload v1-5-pruned.ckpt	11 months ago
v1-5-pruned.safetensors	7.7 GB	LFS	Adding `safetensors` varian	7 months ago

- Tuning for ML workloads is different than for traditional edge compute
- Limited sandboxing when directly linking framework libraries in process
- Stateless serverless model means each request must load model

problem: decouple model lifecycle from FaaS request handling

Build your own FaaS + ML: performance

current

with named models

before instantiation

load and configure model with name "foo"

during execution

high cost... for every request!

```
let bytes = fetch(...);
```

```
let graph = load(bytes);
```

```
graph.compute(...);
```

```
let graph = load_by_name("foo");
```

```
graph.compute(...);
```

Build your own FaaS + ML: security

- How to “limit the blast radius” of malicious models, resource hogging?

HW-based

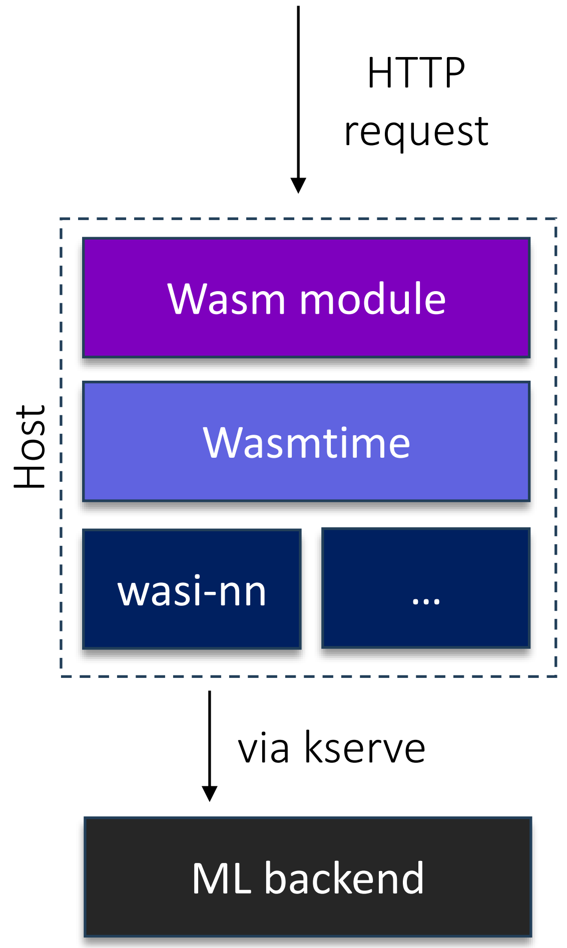
- explored memory protection keys (MPK)
- 16 keys to mark pages as protected
- downside: only protects CPU pages

Process-based

- implemented backend in separate process
- OS-provided, mostly configuration-driven
- GPU drivers isolate process memory
- slight downside: IPC overhead

Demo: kserve backend + async host APIs

- Allows running models where it makes the most sense
  - Small models locally on CPU
  - Larger models on specially tuned machines
- Manage lifecycle of models outside of Wasmtime
- Async host APIs allow Wasmtime to continue handling incoming requests while waiting on inference.
- Out of process model loading and execution means easier sandboxing



WASMCON
BETTER TOGETHER

DEMO

Q&A

Contact us!

- If you want to try this out in Compute@Edge:
  - email Matthew Tamayo-Rios, mtr@fastly.com
- To discuss changes to the specification or implementation questions:
  - open issues at https://github.com/WebAssembly/wasi-nn or
  - email Andrew Brown, andrew.brown@intel.com

WASMCON

BETTER TOGETHER