Mojo in HEP Frameworks

CERN openlab Summer Student Programme Lightning talks

Abbas Naim (AUB)

Supervisor: Axel Naumann (CERN)

Table of Contents

1. Why port to Mojo
2. Patatrack for Evaluation
3. What are we looking for in a language vs what mojo has
4. What have we ported so far
5. How does mojo stack up compared to cpp in performance
6. What is mojo lacking

Why?

What do we have, today?

HEP frameworks use C++, almost exclusively

Often >100 devs collaborate on >10 MLOC / framework

Needs expertise that is harder and harder to get

github.com/isocpp/logos

What Do We Want?

What do we want, also in 10 years?

Fast to run! 10% faster means we save a lot of money (or can do more physics)

Easy to learn, open source, significant ecosystem, not a single vendor, memory safe

GPU support, multi-threading support, “math” support; typesafe, memory-safe

Let’s dream: interface to Python! To C++! Extensive tooling support!

Patatrack

Proved to be a good test bed for languages and concurrency libraries

Similar to CMSSW code

Modules: **Beamspot**, **Clusterizer, RecHits**, **PixelTriplets**, **VertexFinding**, **Validation**

What have we ported

Mojo CUDA-dev port created: **119 files so far**

CPU framework port + tests finished (event parallelism)

CUDA framework + runtime test ported: Mojo runs on GPU (NVIDIA & AMD) natively

Build Times, Source & Binary Size

Missing ~1500 lines of C++ code — so fairly comparable

Building only PixelTrack C++ code (none of its dependencies) vs building Mojo code

4-core CERN OpenStack VM	C++	Mojo
Build Time	-j1: 110 sec -j4: 60 sec	10 seconds
Binary Size	300 kB	400 kB
Source Size	1.0 MB	900 kB

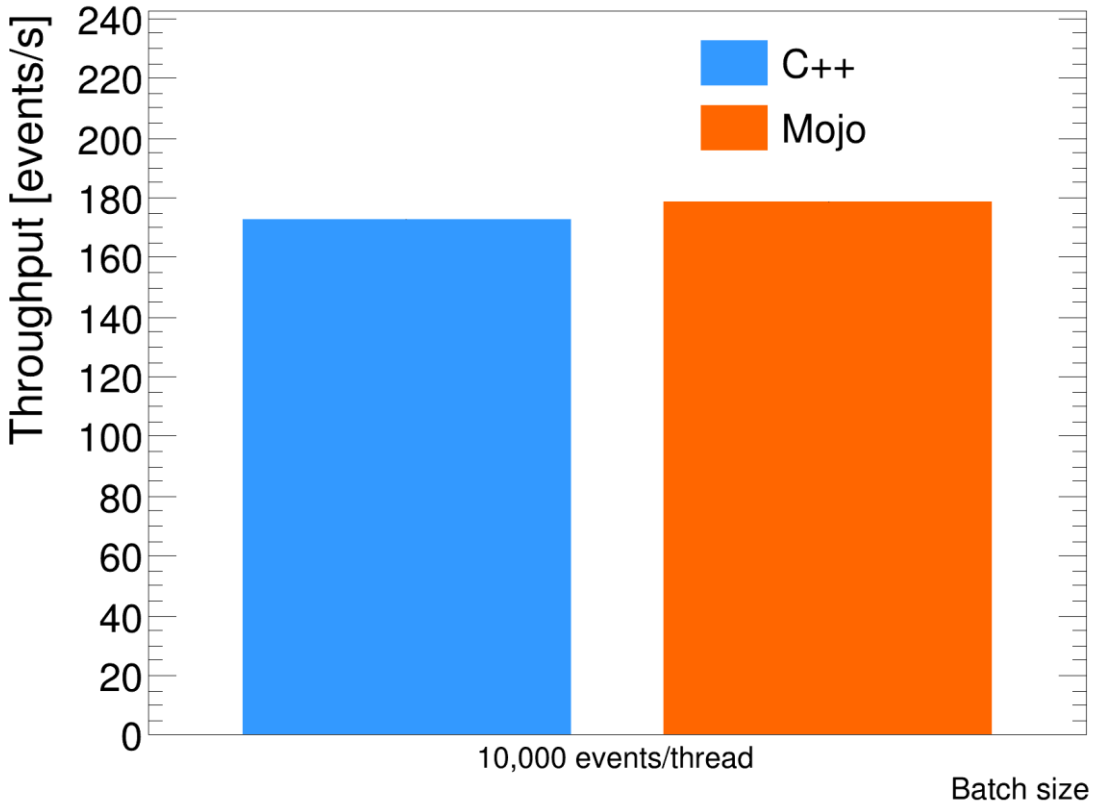
Photo by Ümit Yıldırım on Unsplash

Serial Performance

Result: **Mojo is 3% (0..15%*) faster**

⇒ We haven't even started optimizing it yet ⇐

*: On some Intel CPUs, likely due to GCC vs llvm. Thanks, CERN openlab & Philippe Canal, Fermilab!

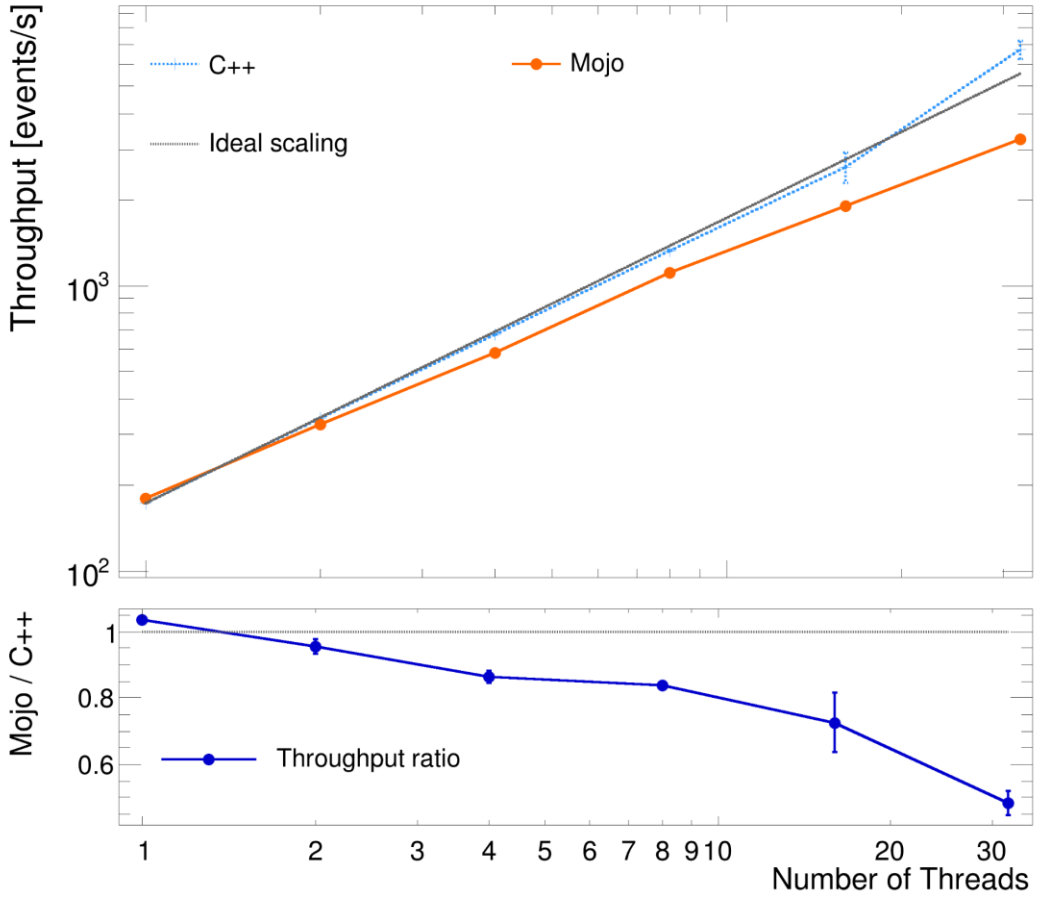


MT Performance

Result: notable scaling problems

MT language + runtime support: officially not yet designed nor implemented

Take-home: MT works but language needs to grow



Metrics: What is “Good”?

How difficult is it to port? Syntax, libraries, tooling ecosystem

What is the language’s featureset?

Support channels, documentation, fora, LLMs

And of course: **performance**

Some (Not So Fun) Experiences

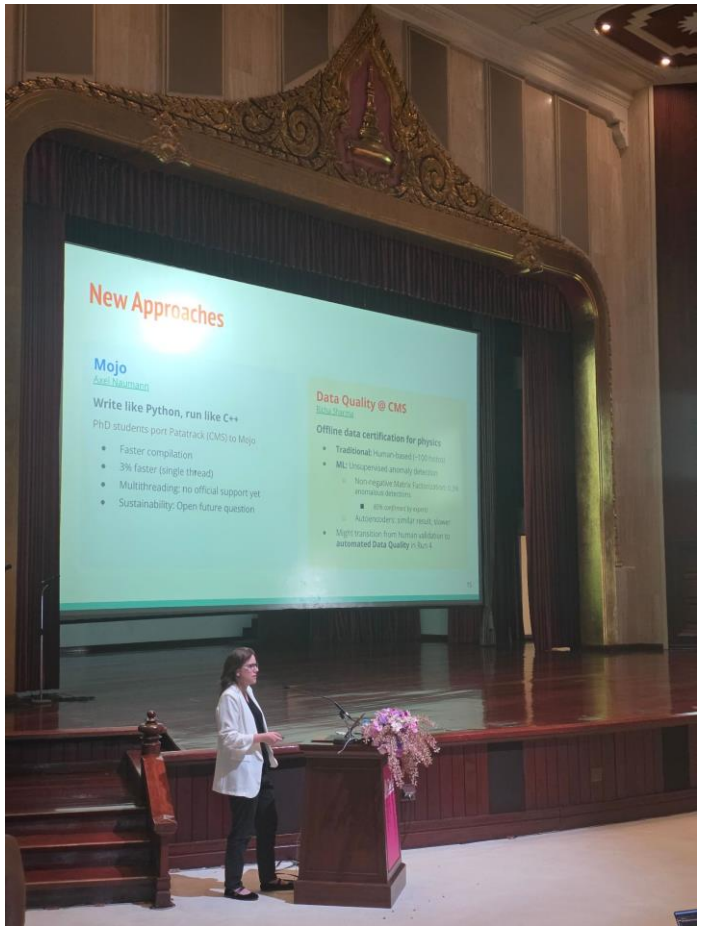
```
var done = Event(ctx)

producer.enqueue_function[my_producer_kernel](grid_dim=64, block_dim=256)
done.record(producer)  # marks "everything above on `producer` is done"

done.synchronize()
```

CHEP

Mojo’s company acquired by Qualcomm

Version 1.0 released

Compiler still “to be open sourced in summer”



References

Patatrack Mojo repo

Mojo and its roadmap

ACAT 2024 presentation on Mojo

…and just select “Mojo” in Compiler Explorer