Analyzing Compilation and Parallelization Strategies in JAX and PyTorch

R244 mini-project discussion

Why compare JAX and PyTorch?

Both dominant frameworks for ML/scientific computing, but different design philosophies

JAX: Functional, transform-based (jit, grad, vmap), explicit control over compilation

PyTorch: Eager execution, but with growing JIT compiler infrastructure (torch.compile = Dynamo + Inductor)

Which one should you use? When? Why?

Personal motivation

Interested in computation for scientific workflows = data processing (moving tensors around) + model building (e.g. generative models)

Which one should you use? When? Why?

Also a chance for me to explore how compilation works in the Python ecosystem more broadly, and with machine learning frameworks in particular

“How do we go from model.train() to where the rubber meets the road?”

What will I investigate?

Compilation Pipeline: How do JAXPR/XLA vs. Dynamo/Inductor differ in:

1. Tracing and graph construction strategies
2. Recompilation triggers and caching behavior
3. Handling dynamic shapes and control flow

Data Parallelism: What are the tradeoffs between:

1. JAX's functional approach (vmap/pmap/pjit)
2. PyTorch's DDP/FSDP imperative constructs
3. (if possible) Multi-GPU scaling behavior

How will I investigate?

- Build microbenchmarks for common operations (matmul, control flow, nested loops)
- Develop profiling tools to inspect IRs
- Measure: compilation overhead, execution speedup, memory usage
- If time permits: case study on flow matching for molecular generation

Plan and Timeline

Next 2 Weeks

- Literature review on JAX/XLA and PyTorch compilation stacks
- Set up development environment with both frameworks, set with GPU access
- Initial microbenchmark suite for basic operations (matmul, loops)
- Implementing IR inspection tools to visualize JAXPR and FX graphs
- Testing control flow compilation (jax.lax.scan vs. torch.compile with loops)
- Profiling compilation overhead vs. execution time tradeoffs

Final 2 Weeks

- Complete parallelism comparison (vmap vs. DDP) on multi-GPU setup if possible
- Benchmark dynamic shapes and recompilation behavior
- Document "performance cliffs" and optimal use patterns
- (Stretch goal) Test frameworks on flow matching model for molecules
- Write final report with benchmark results and recommendations

Feedback I would like

1. What would you like to know about JAX vs PyTorch? (or other frameworks?) What can I investigate that could help you?
    a. Since many of you will be working on machine learning or ML systems things
2. Does anyone have previous experience with profiling compilers? What did you measure and how?
    a. Mojo for JAX/Pytorch? https://www.modular.com/mojo
    b. Nsys profiling for CPU <> GPU memory transfer