Opportunities And Challenges Of Machine Learning Accelerators In Production

2019 USENIX Conference On Operational Machine Learning

Rajagopal Ananthanarayanan, Peter Brandt, Manasi Joshi, Maheswaran Sathiamoorthy
{ananthr, pbrandt, manasi, nlogn}@google.com
Agenda

- Intro
- Production Deployment
- Architectural Specialization - TPU Case Study
- Developer Experience
- Opportunities and Future Directions
- Summary
Intro
Big Trends Shaping the Future of Compute

End of Moore’s Law

- End of Line? 2X / 20 yrs (3%/yr)
- Amdahl’s Law => 2X / 6 yrs (12%/yr)
- End of Dennard Scaling => Multicore 2X / 3.5 yrs (23%/yr)
- CISC 2X / 3.5 yrs (22%/yr)
- RISC 2X / 1.5 yrs (52%/yr)

Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018. Based on SPECintCPU.

Rise of Deep Learning

- AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

Source: https://blog.openai.com/ai-and-compute/
Demand for ML Accelerators has Exploded

Google has built and deployed 3 generations of TPUs now widely used across the company.

NVIDIA has built an ML accelerator datacenter business with $2.93B revenue in fiscal year 2019, up 52% YoY.

AI chip startups have proliferated.

Cloud TPU v3 Pod: 100+ petaflops, 32 TB HBM
ML Accelerators drive Product Innovation

Assistant’s Wavenet human-sounding voice serves all production traffic on TPUs (article, related Cloud product).

The original version of WaveNet took 1 second to generate just 0.02 seconds of audio.

The new WaveNet model is 1,000 times faster and can produce 20 seconds of higher quality audio in 1 second.
Production Deployment
ML Accelerators are typically connected to a host CPU.

- **Your VM**
  - Runs data pre-processing

- **Host VM**
  - CPU
  - Runs linear algebra for deep learning

- **ML Accelerator**
  - Connected via PCI

- **Network**
The ratio between ML Accelerators and their host CPU machines is generally fixed / hard to dynamically change.

If the CPU host can’t perform data processing, shuffling, transformation, etc. at a high throughput, the ML Accelerator may be underutilized.

ML Accelerators are getting faster at a quicker rate than CPUs! Depending on the ML workload, it is easy for systems to be bottlenecked on CPU or I/O. APIs like `tf.data` can help.
Utilizing Diverse Hardware Pools Efficiently

Sharing is “easy” with homogeneous hardware

Today, hardware pools are much more heterogeneous

Help me!
Utilizing Diverse Hardware Pools Efficiently

Job scheduling constraints are increasingly complex. A few techniques that work in practice:

- Smart queueing of training workloads with a dominant resource fairness policy¹
- Proportional sharing instead of job priority on a large pool of resources
- Bin packing multiple models on a single device
- Trading off inference latency and query cost constraints

Resource Planning

Timelines for designing new hardware and deploying it to data centers stretch over several years. Demand can be unpredictable.

Decide what you are optimizing for! There is tension between ensuring flexibility for launches / research and maximizing “utilization”.

- Utilization can have many definitions e.g. financial impact, examples per second, model convergence, doing “useful work”.
- Machine-level instrumentation can be a good coarse metric e.g. power usage of individual chips.

Accurately simulating future hardware performance for computationally intensive models can greatly help planning.
Architectural Specialization – TPU Case Study
Architectural Specializations

- **Instruction sets for linear algebra**
  - Large matrix multiply unit
  - VLIW instructions
  - Restricted precision data types e.g. bfloat16, quantized integers

- **Memory hierarchy**
  - Operate over block-oriented data
  - Buffers to efficiently feed data

- **Specialized networks and data paths**
  - Fast synchronization and AllReduce

- **Separate hosts with generalized CPUs**
  - Implement language runtime and operating system services
TPU V1 Architecture

Designed for high performance linear algebra and deep learning at scale.

Source: https://cacm.acm.org/magazines/2018/9/230571-a-domain-specific-architecture-for-deep-neural-networks/fulltext
TPU V2 Pod Interconnect

Pod configuration can scale ~linearly and act as a unified accelerator on large problems of billions of parameters.

“All-reduce” weights updates on a 2-D toroidal mesh network

As easy to use as a single node

See animation at tpudemo.com
**Software Stack**

Computation is defined as a directed acyclic graph (DAG) to optimize an objective function.

- Graphs are defined in high-level language e.g. Python
- Executed (in parts or fully) on available low-level devices e.g. CPU, GPU, TPU
- Tensors of data flow through the graph
- Gradients are computed automatically!
Software Stack

TFX, TF Hub, TF Serving, TF Transform

Specify a model using common high-level abstractions

Keras and TF Estimator

Train, eval, export, and serve in production with template code for common classes of models

TensorFlow

Run compute graph ops partially or completely on ML accelerator hardware

XLA Just-in-time Compiler

TF graphs go in

Optimized assembly comes out (illustrated w/ x86 assembly)

0x00000000  movq  (%rdx), %rax
0x00000003  vmovaps (%rax), %xmm0
0x00000007  vmulps  %xmm0, %xmm0, %xmm0
0x0000000b  vmovaps %xmm0, (%rdi)
Developer Experience
Tips and tricks

- Smart decompose by dividing ops across host CPU and accelerators - compatibility vs. suitability
- Balance large batch sizes (beneficial for data parallelism) and slow ramp up of learning rate for training (to compensate for potential poor quality)
- Report errors through multiple levels of abstractions

Development and production workflow

- Continuously track model quality during training to catch regressions, and gate model deployment
- Continuously benchmark inference load tests - optimize batch size for the highest throughput vs. compromising query costs or latency
Opportunities and Future Directions
Modeling Enhancements

- Multitask learning
- Transfer learning
- Multimodal models
- Sparse features
- Larger datasets and features
Infrastructure Improvements

- Language and compilers
  - Built-in support for accelerators e.g. JAX, Julia
- Distribution strategies for large-scale systems
  - Mirroring variables across workers
- Better support for balanced system throughput
  - Input, output, and intermediate checkpoints
Summary
Summary

- ML workloads are computationally intensive and fast growing
- Investment in Domain Specific Architectures is increasing
  - Instruction sets, systems architecture
- Diverse ML programming environments are proliferating
  - Interpreted, compiled, graph analysis, auto differentiation
  - Ensure good programming / developer experience
- Optimizing data center deployment is critical
- Ultimately, there is a lot of innovation enabling new types of ML models and products!