Lecture: Commercial Efforts

- Topics: Google TPU v1/2/3 (inference and training),
  Tesla FSD (inference),
  NVIDIA Volta (inference and training),
  Graphcore (training)
Google TPU Intro

• Google’s effort to design a custom chip in 15 months to improve performance-cost for their datacenter workloads

• Plugs into a server, similar to a GPU card; out-performs the GPU by 15x; potential to go even higher

• Architecturally, we’ve seen many of these ideas before; the Google insight makes this a compelling read
Google Relevant Workloads

- Only 5% are CNNs
- Note the Ops/Weight

<table>
<thead>
<tr>
<th>Name</th>
<th>LOC</th>
<th>Layers</th>
<th>Nonlinear function</th>
<th>Weights</th>
<th>TPU Ops / Weight Byte</th>
<th>TPU Batch Size</th>
<th>% of Deployed TPUs in July 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP0</td>
<td>100</td>
<td>FC 5</td>
<td>ReLU</td>
<td>20M</td>
<td>200</td>
<td>200</td>
<td>61%</td>
</tr>
<tr>
<td>MLP1</td>
<td>1000</td>
<td>Conv 4</td>
<td>ReLU</td>
<td>5M</td>
<td>168</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>LSTM0</td>
<td>1000</td>
<td>Vector 24</td>
<td>sigmoid, tanh</td>
<td>52M</td>
<td>64</td>
<td>64</td>
<td>29%</td>
</tr>
<tr>
<td>LSTM1</td>
<td>1500</td>
<td>Pool 34</td>
<td>sigmoid, tanh</td>
<td>34M</td>
<td>96</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>CNN0</td>
<td>1000</td>
<td>Total 58</td>
<td>ReLU</td>
<td>8M</td>
<td>2888</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>CNN1</td>
<td>1000</td>
<td>FC 16</td>
<td>ReLU</td>
<td>100M</td>
<td>1750</td>
<td>32</td>
<td>5%</td>
</tr>
</tbody>
</table>

Table 1. Six NN applications (two per NN type) that represent 95% of the TPU’s workload. The columns are the NN name; the number of lines of code; the types and number of layers in the NN (FC is fully connected, Conv is convolution, Vector is self-explanatory, Pool is pooling, which does nonlinear downsizing on the TPU; and TPU application popularity in July 2016. One DNN is RankBrain [Cla15]; one LSTM is a subset of GNM Translate [Wu16]; one CNN is Inception; and the other CNN is DeepMind AlphaGo [Sil16][Jou15].
TPU Architecture

Weights are pre-loaded during previous phase and inputs flow left to right.
Causes of Speedup

• GPUs use 16b FP; TPU uses 8b integer ops (addition is 13x more energy-efficient and multiplication is 6x)

• GPUs focus on throughput and not as much on latency

• GPUs have features (caches, bpreds) that introduce response time variance

• TPU uses a systolic array to exploit reuse
### Server Parameters

<table>
<thead>
<tr>
<th>Model</th>
<th>mm²</th>
<th>nm</th>
<th>MHz</th>
<th>TDP</th>
<th>Measured</th>
<th>TOPS/s</th>
<th>GB/s</th>
<th>On-Chip Memory</th>
<th>Dies</th>
<th>DRAM Size</th>
<th>TDP</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Idle</td>
<td>Busy</td>
<td>8b</td>
<td>FP</td>
<td></td>
<td></td>
<td></td>
<td>Idle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Busy</td>
</tr>
<tr>
<td>Haswell E5-2699 v3</td>
<td>662</td>
<td>22</td>
<td>2300</td>
<td>145W</td>
<td>41W</td>
<td>145W</td>
<td>2.6</td>
<td>1.3</td>
<td>51</td>
<td>51 MiB</td>
<td>2</td>
<td>256 GiB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NVIDIA K80 (2 dies/card)</td>
<td>561</td>
<td>28</td>
<td>560</td>
<td>150W</td>
<td>25W</td>
<td>98W</td>
<td>--</td>
<td>2.8</td>
<td>160</td>
<td>8 MiB</td>
<td>8</td>
<td>256 GiB (host) + 12 GiB x 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPU</td>
<td>NA*</td>
<td>28</td>
<td>700</td>
<td>75W</td>
<td>28W</td>
<td>40W</td>
<td>92</td>
<td>--</td>
<td>34</td>
<td>28 MiB</td>
<td>4</td>
<td>256 GiB (host) + 8 GiB x 4</td>
</tr>
</tbody>
</table>

Table 2. Benchmarked servers use Haswell CPUs, K80 GPUs, and TPUs. Haswell has 18 cores, and the K80 has 13 SMX processors. Figure 10 has measured power. The low-power TPU allows for better rack-level density than the high-power GPU. The 8 GiB DRAM per TPU is Weight Memory. GPU Boost mode is not used (Sec. 8). SECDEC and no Boost mode reduce K80 bandwidth from 240 to 160. No Boost mode and single die vs. dual die performance reduces K80 peak TOPS from 8.7 to 2.8. (*The TPU die is ≤ half the Haswell die size.*)
Roofline Model

![Roofline Model Graph]

- **TPU Log-Log**
- **TeraOps/sec (log scale)**
- **Operational Intensity: Ops/weight byte (log scale)**

Legend:
- **Roofline**
- **LSTM0**
- **LSTM1**
- **MLP1**
- **MLP0**
- **CNN0**
- **CNN1**
## Response Times

<table>
<thead>
<tr>
<th>Type</th>
<th>Batch</th>
<th>99th% Response</th>
<th>Inf/s (IPS)</th>
<th>% Max IPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>16</td>
<td>7.2 ms</td>
<td>5,482</td>
<td>42%</td>
</tr>
<tr>
<td>CPU</td>
<td>64</td>
<td>21.3 ms</td>
<td>13,194</td>
<td>100%</td>
</tr>
<tr>
<td>GPU</td>
<td>16</td>
<td>6.7 ms</td>
<td>13,461</td>
<td>37%</td>
</tr>
<tr>
<td>GPU</td>
<td>64</td>
<td>8.3 ms</td>
<td>36,465</td>
<td>100%</td>
</tr>
<tr>
<td>TPU</td>
<td>200</td>
<td>7.0 ms</td>
<td>225,000</td>
<td>80%</td>
</tr>
<tr>
<td>TPU</td>
<td>250</td>
<td>10.0 ms</td>
<td>280,000</td>
<td>100%</td>
</tr>
</tbody>
</table>

TPU can meet the 7 ms constraint while allowing high batching factor ➔ higher utilization, thruput, energy-efficiency
Next Steps for TPU

• Better circuit optimizations to improve clock speed

• Add support for GDDR5 memory to get further 3x speedup (will increase area or reduce buffer capacity)

• Add support for sparsity

• Add support for power-down features for better energy proportionality

• Can get even higher performance with software tuning
Take-Home

• 99th percentile response time is important; has a secondary impact on batching/throughput/energy

• Relevance of LSTMs and MLPs

• Relative to GPU: half area, 25x more MACs, 4x more mem, 15x higher speed for inference, 30x higher perf/watt (TCO)

In summary, the TPU succeeded because of the large—but not too large—matrix multiply unit; the substantial software-controlled on-chip memory; the ability to run whole inference models to reduce dependence on host CPU; a single-threaded, deterministic execution model that proved to be a good match to 99th-percentile response time limits; enough flexibility to match the NNs of 2017 as well as of 2013; the omission of general-purpose features that enabled a small and low power die despite the larger datapath and memory; the use of 8-bit integers by the quantized applications; and that applications were written using TensorFlow, which made it easy to port them to the TPU at high-performance rather than them having to be rewritten to run well on the very different TPU hardware.
TPU v2

- A pod is a cluster of 64 TPU v2’s connected with a torus topology
- A cloud resource that can execute Tensorflow inference & training tasks
- Each TPU v2 has 4 chips and 64GB HBM
- Likely consumes 2x more power (heat sinks, fans, etc.)
- Their largest language translation model takes a day to train on 32 GPUs; takes 6 hrs on 8 TPUs, i.e., 16x better than GPUs on training
- They introduce their own bfloat16 format which has more training accuracy than FP half-precision; bfloat16 simply drops the 16 least significant mantissa bits
TPU v2

- Each MXU can do 16K multiply ops
- Inputs/outputs are 32 bits, but the multiply is bfloat16
- 45 TOPs/s

Reference: https://cloud.google.com/tpu/docs/system-architecture
Next TPUs

From Jeff Dean’s NIPS’17 talk:

- Will very low precision (1-4 bits) be effective for all workloads?
- How to handle sparsity and embeddings?
- What batch size should the accelerator be designed for?
- Will SGD continue to be the dominant training algorithm?

TPU 3.0

- Announced May 2018
- 8x more powerful
- Liquid cooled
Tesla FSD

- Tesla’s custom accelerator chip, shipping in cars since April 2019
- FSD sits behind the glovebox, consumes 72W
- 18 months for first design, next generation out in 2 years
Detection and tracking are two of the heavy-hitters and are DNN based
NN Accelerator Chip (NNA)

- Goals: under 100 W (2% impact on driving range, cooling, etc.), 50 TOPs, batch size of 1 for low latency, GPU support as well, security/safety.

- Security: all code must be attested by Tesla

- Safety: two completely independent systems on the board that verify every output

- The FSD 2.5 design (GPU based) consumes 57 W, the 3.0 design consumes 72 W, but is 21x faster (72 TOPs)

- 20% saving in cost by designing their own chip
Motivational Data

- Data from Horowitz et al. (ISCA 2010): in a regular CPU, 0.1pJ for 32b add, 6pJ for regfile, 39pJ for control, 25pJ for icache, i.e., the add is 0.15% of total power.
- In an ideal accelerator, 100% of total power is in the ALU.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Energy per operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>8b integer Add</td>
<td>0.03 pJ</td>
</tr>
<tr>
<td>32b integer Add</td>
<td>0.10 pJ</td>
</tr>
<tr>
<td>8b integer Mult</td>
<td>0.20 pJ</td>
</tr>
<tr>
<td>32b integer Mult</td>
<td>3.00 pJ</td>
</tr>
<tr>
<td>16b FP Add</td>
<td>0.40 pJ</td>
</tr>
<tr>
<td>32b FP Add</td>
<td>0.90 pJ</td>
</tr>
<tr>
<td>16b FP Mult</td>
<td>1.00 pJ</td>
</tr>
<tr>
<td>32b FP Mult</td>
<td>4.00 pJ</td>
</tr>
<tr>
<td>SRAM 64b 32KB array</td>
<td>20 pJ</td>
</tr>
<tr>
<td>DRAM 64b</td>
<td>2000 pJ</td>
</tr>
</tbody>
</table>
NNA Pipeline

- Inputs from 8 cameras, Radar, GPS, ultrasonic, maps, driver, etc.
- Camera serial interface: 2.5B pixels/sec
- On-chip network moves inputs to LPDDR4: 128b@4.2 Gb/s = 68GB/s
- Includes: video encoder, image signal processor, 600 Gflop GPU, and 12-core 2.2 GHz CPU, hardware for ReLU and pooling layers
- Most importantly: 2 NN accelerator cores, each with 96x96 grid of MACs and 32MB SRAM, 2 GHz, 36 TOPs per core

Image Source: Tesla
NNA Specs

- 37.5mm x 37.5mm BGA package, 2116 balls, 260 mm$^2$ die, 12.5K C4 bumps, 12 metal layers, 14nm FinFET CMOS
- 250M gates, 6B transistors, tested to AEC Q100 standards (auto)
- A basic NN for their narrow camera needs 35 GOPs: the 12-core CPU only allows 1.5 fps, GPU allows 17 fps, 2.5 FSD allows 110 fps, and NNA allows 2300 fps (need to handle 8 cameras).
- 99.7% operations are dot-products and 0.3% are pooling and relu (have to speed up the latter since you're speeding up the former by 10,000x)
- The SRAM is enough to store all the feature maps passing from one layer to the next; every cycle, they read 256B Activations and 128B weights, with 128B of output written back to SRAM
- Compiler optimizations further help save power
- NNA cores consume 15W (out of 72W FSD power)
Graphcore

• Targets graph-connected workloads, DNNs being the prime example
• Primary philosophy is to reduce data movement and dark silicon
• Already deploying racks of Graphcores with high throughput and no DRAM
• Core design principles:
  • A memory-centric die that achieves high efficiency by keeping memory local
  • Communication and compute are provisioned for peak power and the two are serialized for highest efficiency
  • Re-compute data instead of storing it (no DRAM!)
Graphcore

- With 200 W budget and an 800 mm$^2$ chip, only 1/3 of the chip can run ALUs at 1.5GHz
- DDR4: 320pJ/B, 256GB @ 64GB/s costs 20W
- HBM2 on interposer: 64pJ/B, 16GB @ 900GB/s costs 60W
- Monolithic SRAM on chip: 256MB: 10pJ/B, 6TB/s @ 60W
- Distributed SRAM: 1000 256KB banks is 1pJ/B, 60 TB/s @ 60W
- This SRAM has a power density that is 25% of logic power.

Image Source: Graphcore, NIPS'17
Graphcore vs. GPU

DRAM on interposer
180W GPU + 60W HBM2

16GB @ 64pJ/B
900GB/s

Distributed SRAM on chip
2x IPU (75W logic + 45W ram)

600MB @ 1pJ/B
90,000GB/s

Image Source: Graphcore, NIPS’17
Graphcore

- 4 Colossus chips in one 1U IPU-Machine (see pic)
- 16nm chip, 1000 independent processors per chip
- No attached DRAM
- Mixed-precision fp stochastic arithmetic (16b mult, 32b accum)
- Only about 4 TOPs per chip, but exceeds TPU2 and Volta without large batching
- A rack with 32 1U machines and 4 chips = 500 TOPS
NVIDIA Volta GPU

- 640 tensor cores
- Each tensor core performs a MAC on 4x4 tensors
- Throughput: 128 FLOPs x 640 x 1.5 GHz = 125 Tflops
- FP16 multiply operations
- 12x better than Pascal on training and 6x better on inference
- Basic matrix multiply unit – 32 inputs being fed to 64 parallel multipliers; 64 parallel add operations

\[
D = \begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\
A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\
A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\
A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3}
\end{pmatrix} + \begin{pmatrix}
B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\
B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\
B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\
B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3}
\end{pmatrix}
\]

FP16 or FP32

References

• “In-Datacenter Performance Analysis of a Tensor Processing Unit”, N. Jouppi et al., ISCA 2016

• Graphcore: https://www.graphcore.ai/posts/introducing-the-graphcore-rackscale-ipu-pod

• VOLTA: http://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

• Tesla Autonomy Day, FSD description: https://youtu.be/Ucp0TTmvqOE?t=4301

• Blog post on TPU: https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu#closeImage