# ECE 571 – Advanced Microprocessor-Based Design Lecture 24

Vince Weaver

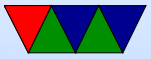
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

26 April 2018

# Announcements

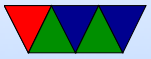
- Don't forget projects, due next Thursday
  If you need benchmarks, or need to borrow equipment, do it now, remember I'll be out of town Monday and Tuesday.
- No class Tuesday
- Will try to get homeworks graded soon.



# Reading 1

**Volta: Performance and Programmability**

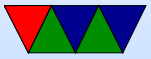
by Choquette, Giroux, and Foley



# First some Extra notes

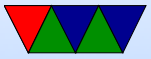
`https://devblogs.nvidia.com/inside-volta/`

Blog post. Marketing info.



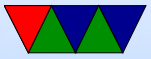
# Volta

- Tesla V100 – Volta GV100 GPU – convergence HPC and AI
- GeForce vs Quadro vs Tesla
- High-end Tesla has gone from Tesla K40 (kepler) to Tesla M40 (Maxwell) to Tesla P100 (Pascal) to Tesla V100 (Volta)
- $10,000, 14 TFlop/peak
- 21.1 billion transistors, 815$mm^2$
- For comparison, Broadwell-EP 7.2 billion, 306$mm^2$ (?

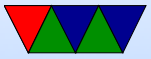


verify)

- TSMC 12nm FFN (finfet fully-functional network?)
- Better than Tesla P100 (Pascal?)
  - 50% more energy efficient
  - 12x TFLOPS in tensor training
  - Combined L1 data cache and shared memory
- 6 NVLink at 25GB/s for total of 300GB/s (not sure how that math works)
- Cache coherence with IBM Power 9
- HBM2 Memory, 16GB, 900GB/s peak. 1.5x better than pascal, 95% utilization in some workloads

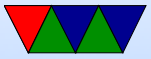


- Multi-process service?
- Unified memory, share pages with processor
- Co-operative Groups, better management of threads that communicate
- Can run up to 300W but has power-capping
- Hardware design
  - Six graphics processing clusters
  - 84 Volta streaming multiprocessors
  - 42 texture-processing clusters (each with two sms)
  - 8 512-bit memory controllers
  - Each SM has 64 FP32, 64 INT32, 32 FP64 cores, 8



tensor cores, and four texture units

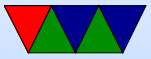
- Volta SM (Streaming Multiprocessor)
  - Mixed precision FP16/FP32 cores for "deep learning"
  - 64 FP32 and 32 FP64 cores per SM
  - Partitioned into 4 processing blocks, each with 16 FP32, 8 FP64, 16 INT32, two tensor cores, L0 icache, 1 warp scheduler, 1 dispatch unit, 64kB register file
  - Separate FP32 and INT32 units so can execute at same time
- Tensor cores
  - Good for machine learning



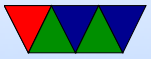
- 4x4x4 matrix processing

$$D = A \times B + C$$

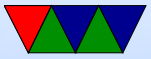
  Where A, B, C, D are 4x4 matrices (A and B are fp16 and C and F can be 16 or 32)
- Enhanced L1 Data Cache and Shared Memory?
- Independent Thread Scheduling
  - Older cores execute group of 32-threads (a warp) as SIMT. All threads share same program counter. If things diverge, the ones on the other path just not run (leaving idle execution units)



- Pascal made this better by trying to merge things together if possible, running if/else at same time, but loss of concurrency (threads might be executing earlier than expected) so cannot communicate inter-thread
- Volta allows much more complicated dependency based threading, adding a new `__syncwarp` to force a sync

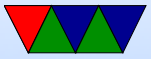


# Actual Reading

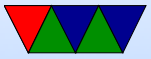


# NVLINK

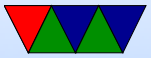
- allows to read HBM on other GPUs or system memory (Power)
- On Pascal was asymmetric – CPU could not access the GPU mem except via PCIe
- On Volta, A power9 and GPU can both initiate transactions.
- Shared memory address space.
- Still NUMA like
- Atomic operations



- Memory coherence – CPU can hold GPU data in its caches and GPU will snoop as necessary
- GPU can also keep CPU data in its caches
- GPU cache is write-through
- Power9 L3-cache is 120 MBytes?
  Makes it hard to track what is in it.
  Probe filter.
- GMMU. Has own TLB.
- 25.78125 GB/s. Six lanes, *bi-directional* so 309Gbytes/s
- Powersaving – NVLink is always running and a bit power

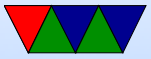


hungry, but can shut down lanes to save power

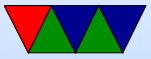


# SM Core

- Twice the insn schedulers, simpler issue rules (easier to target)
- Large, fast, L1 cache
- Improve energy efficiency by 50%
- uarch
- sub-core
  - issue one warp insn per clock from L0 icache
  - Issue to branch unit, math dispatch, tensor, memIO
  - Four datapaths, INT32, FP32, FP64, MUFU (misc)

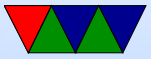


- 4x4x4 tensor core

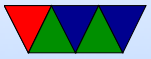


# L1 Cache / Shared Memory

- Four SM sub-cores send MIO requests
- Texture or unified shared.
- Texture one pixel-quad per clock
- L1 dcache 128k, loads/stores for 32 thread per clock, 128 bytes/clock (4 times better than pascal)
- Streaming cache?
- Up to 96k of the 128k can be used as shared (scratchpad?)
- Pascal cache hits had same latency as misses, SMEM

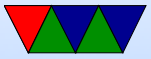


was faster than L1 (L1 was just easier for programmer)
Volta SMEM/L1 much closer



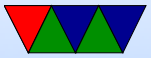
# Concurrent Programming

- As described before
- Prior to Volta only lock-free concurrent algorithms worked
- Forward progress guarantee



# Tensor Cores

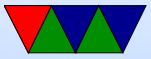
- 640 tensor cores, 8 per SM
- 64 fp ops per clock
- GV100 120 tensor TFlops



# Reading 2

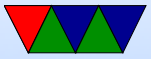
**NVIDIA Announces Jetson TX1 - A Tegra X1 Module & Development Kit**

by Ryan Smith



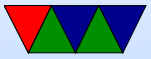
# Tegra Security Notes

- Just released nvidia tegra flaw on switch
- `http://www.theregister.co.uk/2018/04/23/nvidia_tegra_flaw/`
- Bug in ROM, bypass security to boot custom on Nintendo Switch
- Buffer overflow in USB code

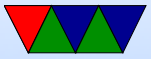


# Jetson TX-1

- We used this board for the homeworks, released in November 2015.
- Tegra X1 SoC
- 64-bit ARM Cortex A57 CPUs (4 cores) (actually big little with Cortex A53 as well)
- "1-TFLOP" 256-core Maxwell class GPU
  When run double-precision hpl_CUDA get around 7 GFLOPS which is even less than the CPU gets, although the fan doesn't come on at all.

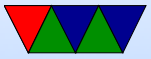


- 4GB LPDDR4 memory (25.6 GB/s)
- Full featured ITX-size I/O carrier board: gigabit Ethernet, wifi, USB, GPIOs, camera, PCIe slot



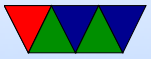
# Jetson TX-1 notes

- A pain to upgrade it
- Newer models have support for reading out power via on-board sense resistor and ADC. Mine is too old



# Jetson TX-2 notes

- NVIDIA Pascal GPU
- Quad A57, but HMP Dual Denver?
  Denver is code-morphing Transmeta like, takes ARMv8 instructions and translates on fly to a VLIW architecture?
- 8GB 128bit LPDDR4 59.7GB/s
- 6 cameras (why?)
- PCIe
- 32MB eMMC
- Also CANBUS



- Gigabit Ethernet