Lecture 20: Systems Today

Principles of Computer Systems

Spring 2019

Stanford University

Computer Science Department

Instructors: Chris Gregg and Philip Levis

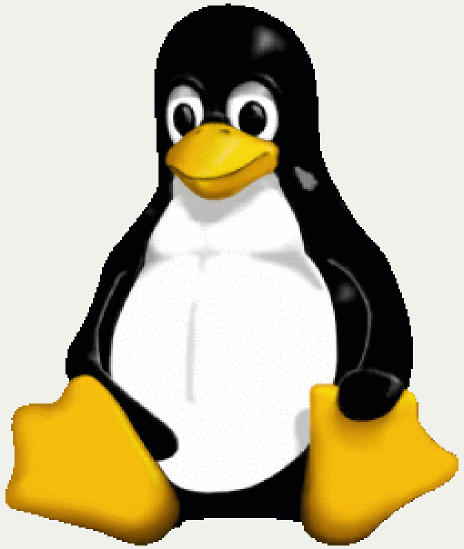
PDF of this presentation

Lecture 20: Systems Today

We wanted to finish up the quarter with a look at the relevance of systems in today's computing world. It is hard to think of a modern computing task that hasn't needed high quality systems ideas to make them relevant. Whether it is for speeding up specific computing tasks (e.g., for AI), improving networking performance, scaling to datacenter-sized computing centers, or making robust secure, redundant storage solutions, systems has had a major role. We are going to look at the following technologies and discuss how systems has played its part in their use:

- Mainstream industry use of Linux
- Google's Tensor Processing Units (TPUs)
- Google's open-source TensorFlow library
- Amazon's Elastic Cloud Compute (EC2)
- Amazon's Simple Storage Service (S3)
- The Internet Backbone
- Systems programming for entertainment (e.g., game design, movie animation, etc.)

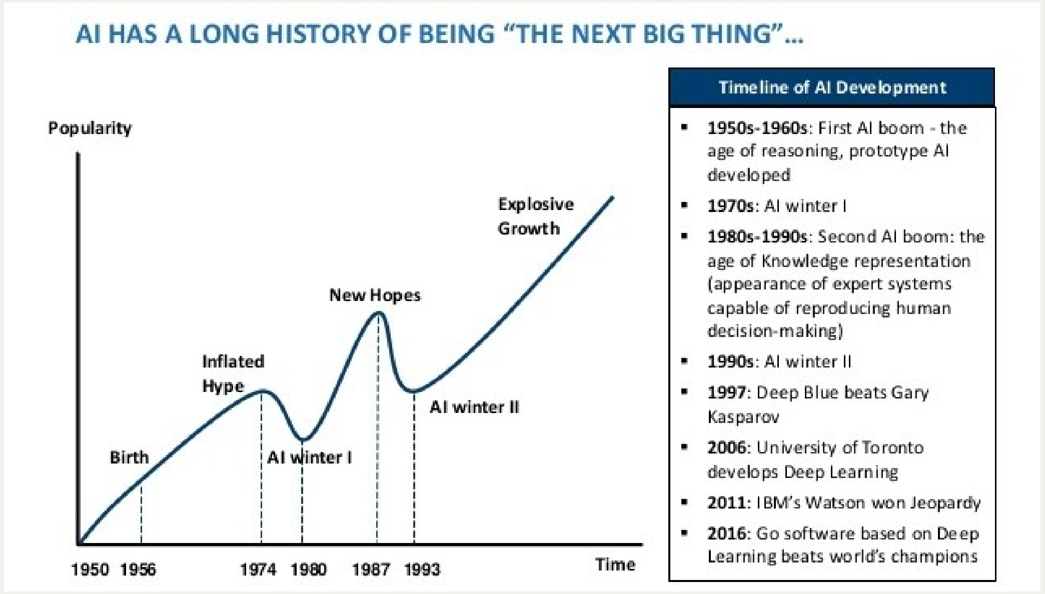
Lecture 20: Systems Today: Mainstream industry use of Linux



We have used Linux exclusively in class, and Linux is used in most of the companies you will be working at, at least for servers in the back-end. Virtually all datacenters use Linux for the back end, (even Microsoft data centers).

The fact that Linux is open source and has hooks into the lowest level of the kernel (indeed, you can look at the code for the Linux code whenever you want) means that it is ripe for getting peak performance when you need it -- this is important in many businesses and technologies that rely on huge numbers of highly connected computers (e.g., companies that use lots of AI).

Lecture 20: Systems Today: Hardware (particularly for AI)



Timeline of AI Development

- **1950s-1960s**: First AI boom - the age of reasoning, prototype AI developed
- **1970s**: AI winter I
- **1980s-1990s**: Second AI boom: the age of Knowledge representation (appearance of expert systems capable of reproducing human decision-making)
- **1990s**: AI winter II
- **1997**: Deep Blue beats Gary Kasparov
- **2006**: University of Toronto develops Deep Learning
- **2011**: IBM's Watson won Jeopardy
- **2016**: Go software based on Deep Learning beats world's champions

image source: https://www.actuaries.digital/2018/09/05/history-of-ai-winters/

- As you know from taking CS 107/107e and CS 110, systems programming is often very close to the hardware. Why might this matter if you are going into a field that doesn't seem to rely on hardware? Well, the field probably *does* rely on hardware, even if you don't consciously think about it!
- Fast hardware is critical for Artificial Intelligence, for example. In the 1970s and again in the 1980s, there were two "AI winters" -- times when AI research slowed to a crawl, despite many good ideas. Part of the reason for the slowdowns were due to the fact that computers were just too slow to handle the algorithms that were being developed.

Lecture 20: Systems Today: Hardware - scaling

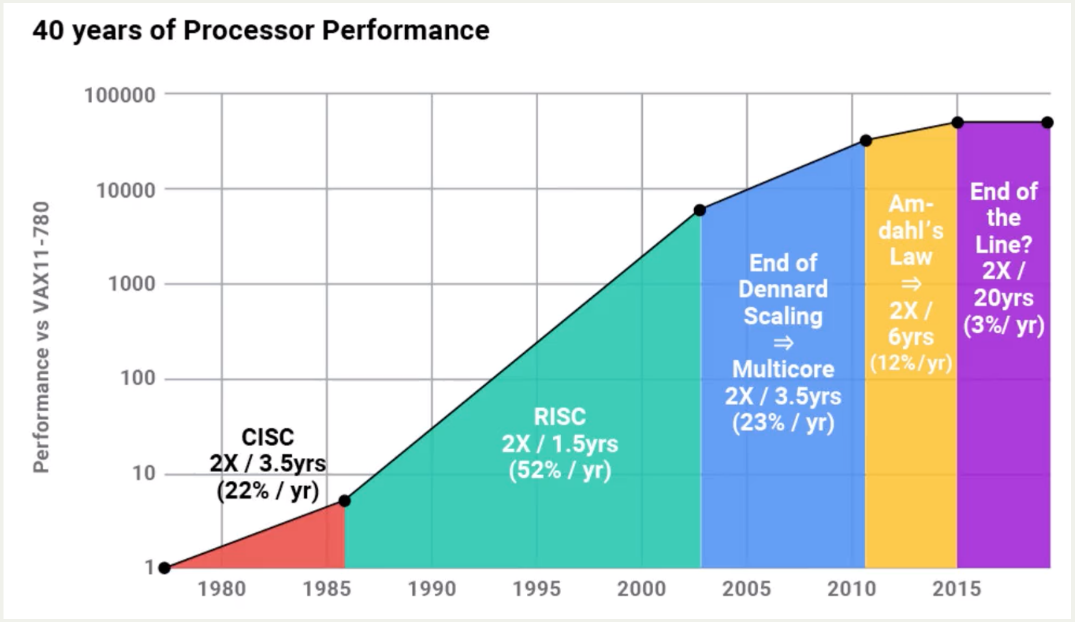


image source: https://storage.googleapis.com/nexttpu/index.html

John Hennessy and David Patterson, Computer Architecture, A Quantitative Approach, 6/e 20118

- Processor improvement has been absolutely incredible over the past 40 years. The graph above (note the log scale!) shows *Moore's Law*, which was a statement by Gordon Moore in a 1965 paper where he described a doubling every year in the number of components per integrated circuit. This (roughly) translates to processor performance, and it has been a remarkably good prediction -- until the last decade or so.
- In 1974, Robert Dennard, at IBM, recognized that with smaller transistors, *energy density* stayed about the same, so power use stays in proportion with area. If chips get too hot, they melt, so this was a good thing, and it lasted...until about a decade ago.

Lecture 20: Systems Today: Hardware - Multicore

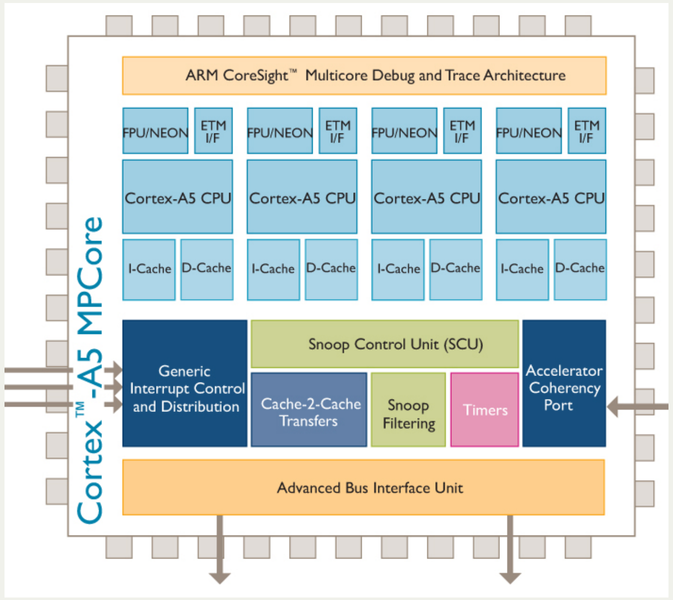
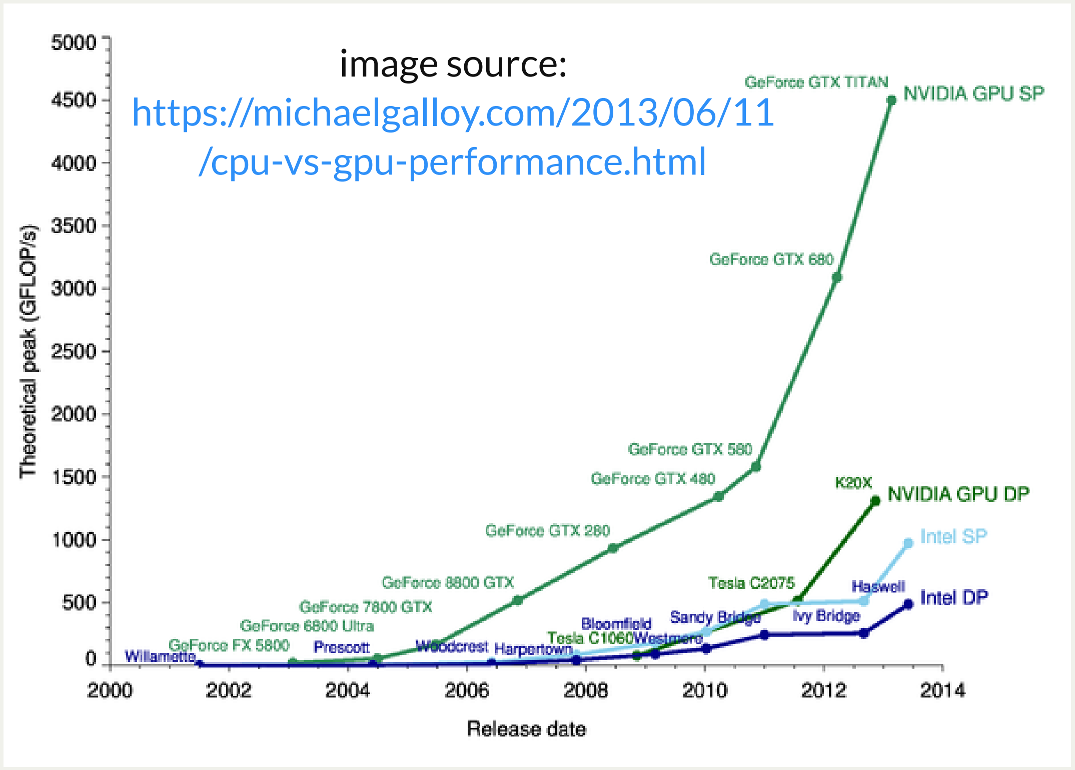


image source: https://phys.org/news/2009-10-arm-cortex-a5-power-efficient-cost-effective-multicore.html

- In about 2005, chip companies (Intel, AMD, ARM) switched their main processor model to *multicore* processors, which helped with the energy problem, but only to a certain extent. OS and application developers had to figure out how to efficiently utilize multiple cores, too (this is a systems programming challenge, as you have found out this quarter!)
- However, there is still the problem of *Wirth's Law:* that software is getting slower more rapidly than hardware is becoming faster (see also: The Great Moore's Law Compensator (TGMLC))
- The average CPU now had 2, 4, 8, or more cores (up to about 64 for very expensive servers)

Lecture 20: Systems Today: Hardware - the rise of GPGPU



- Multicore CPU computing was great, but for highly parallel tasks, GPUs became the king
- GPU companies (primarily NVIDIA and AMD) were building graphics processors with hundreds (and now thousands) of less-powerful cores, but with the ability to perform parallel calculations extremely quickly. Pixel-based graphics can be parallelized, but it turns out that so can many non-graphics problems.

- In 2007, NVIDIA released the *Compute Unified Device Architecture (CUDA)*, an API that allows developers to run general purpose code on a GPU (GPGPU). Other compute frameworks (e.g., OpenCL) also started around this time, and gave programmers with highly parallel workloads a new tool for fast processing.
- AI and machine learning researches, with their highly parallel workloads (e.g., neural network backpropogation) started utilizing GPUs for their work.

Lecture 20: Systems Today: Google's Tensor Processing Units (TPU)

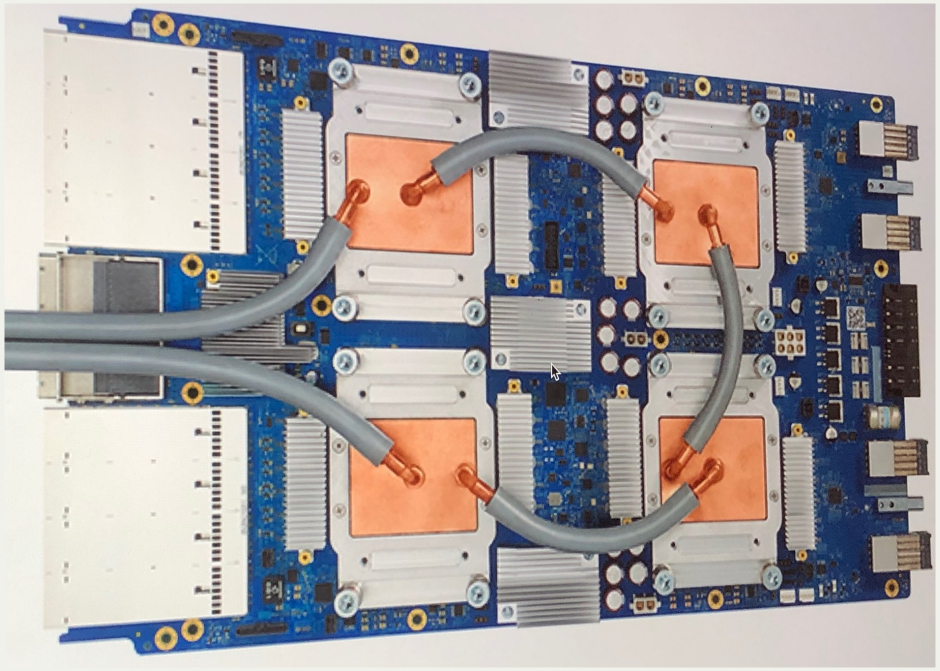


Image: Google's Tensor Processing Unit 3.0
https://en.wikipedia.org/wiki/Tensor_processing_unit#/media/File:Tensor_Processing_Unit_3.0.jpg

- It turns out that GPUs aren't the best tool for AI and machine learning, and in 2016, Google announced the *Tensor Processing Unit*, a chip specifically designed for Google's *TensorFlow* machine learning framework.

- GPUs have thousands of cores, but TPUs have thousands of *Matrix Multiplier Units (MXUs)*. The original TPU had a grid of 256 x 256 8-bit multipliers, which perform a key step many machine learning algorithms. The TPU also had 24MB of on-chip RAM to facilitate the calculations, and 4MB of 32-bit registers.
- By the way -- what are the pipes for on the card above?

Lecture 20: Systems Today: Google's Tensor Processing Units (TPU)

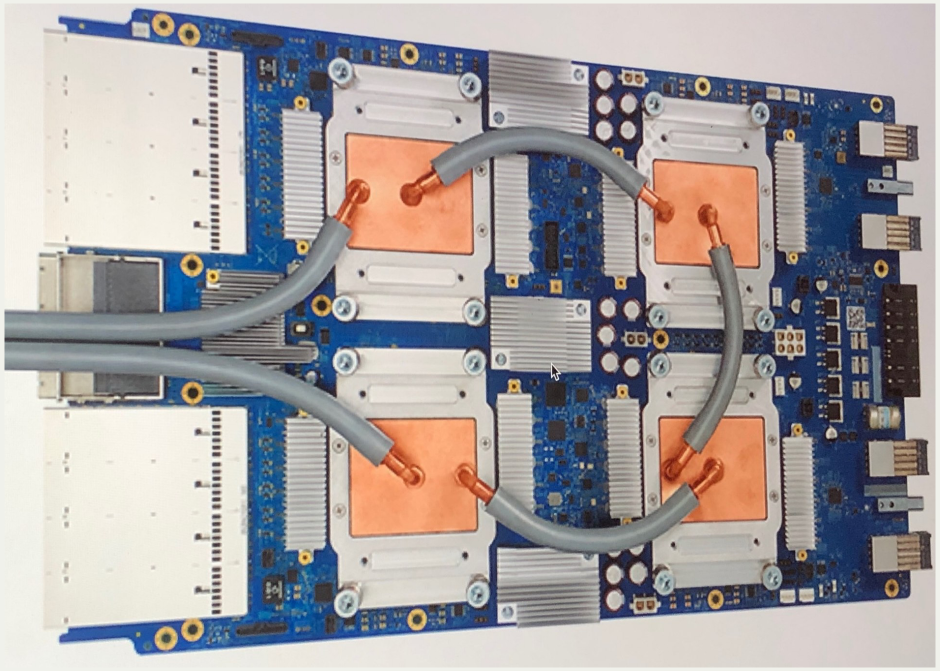
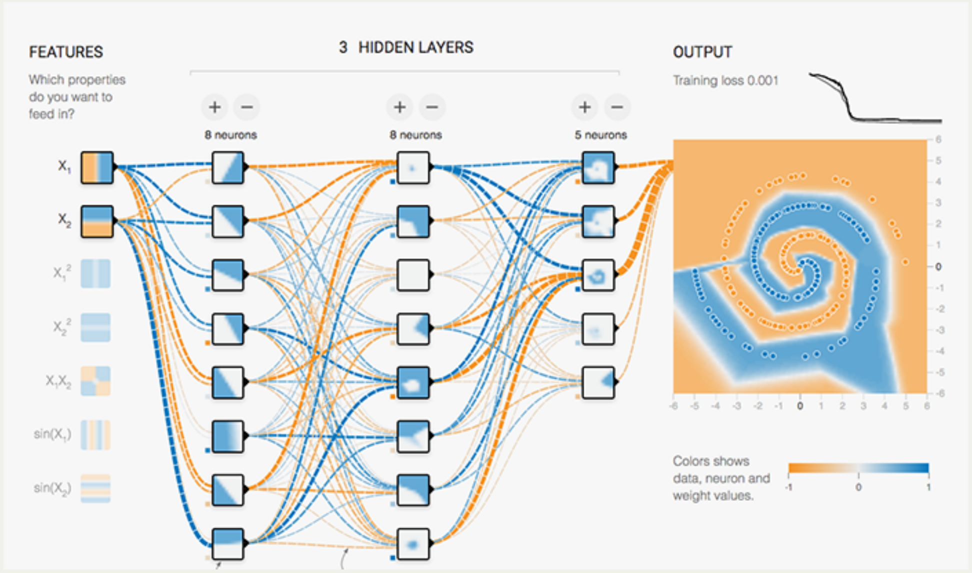


Image: Google's Tensor Processing Unit 3.0
https://en.wikipedia.org/wiki/Tensor_processing_unit#/media/File:Tensor_Processing_Unit_3.0.jpg

- It turns out that GPUs aren't the best tool for AI and machine learning, and in 2016, Google announced the *Tensor Processing Unit*, a chip specifically designed for Google's *TensorFlow* machine learning framework.

- GPUs have thousands of cores, but TPUs have thousands of *Matrix Multiplier Units (MXUs)*. The original TPU had a grid of 256 x 256 8-bit multipliers, which perform a key step many machine learning algorithms. The TPU also had 24MB of on-chip RAM to facilitate the calculations, and 4MB of 32-bit registers.
- By the way -- what are the pipes for on the card above? **Liquid Cooling. 65K multipliers running simultaneously can get pretty hot.**

Lecture 20: Systems Today: Google's Tensor Processing Units (TPU)


FEATURES
Which properties do you want to feed in?
3 HIDDEN LAYERS
8 neurons
8 neurons
5 neurons
OUTPUT
Training loss 0.001
Colors shows data, neuron and weight values.

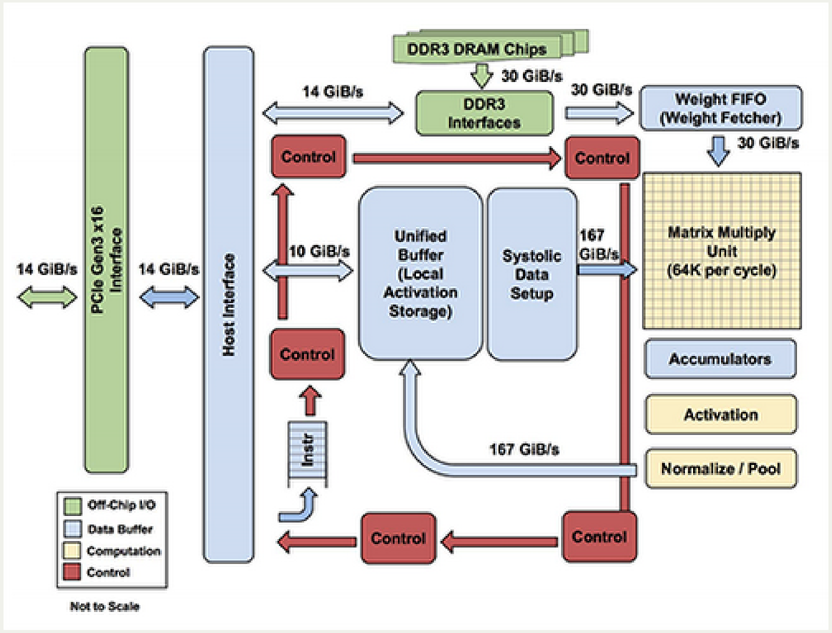
A simple neural network (click to try)
https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

- Neural networks need three basic operations to work:
  - *Multiply* for input data and associated weights
  - *Add* for aggregating the data
  - A way to apply an activation function to the result.

- This is primarily a *matrix multiplication* (at least the first two steps are), and therefore having 64KB of 8-bit multipliers is helpful.
- Wait -- "8-bit"? Why not 32-bit or 64-bit floating point?
- Well, it turns out that you can *quantize* data used in machine learning algorithms to allow calculations on 8-bit integers, simplifying the hardware:

"*If it's raining outside, you probably don't need to know exactly how many droplets of water are falling per second — you just wonder whether it's raining lightly or heavily. Similarly, neural network predictions often don't require the precision of floating point calculations.*" (same link as above)

Lecture 20: Systems Today: Google's Tensor Processing Units (TPU)

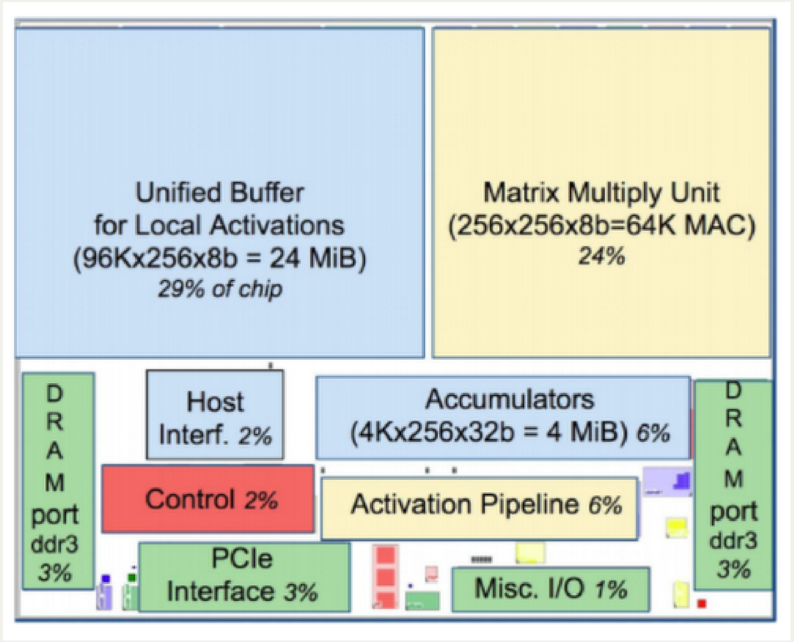


TPU Block Diagram

https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

- The design of the hardware is more than just throwing multipliers and memory on a chip.
- The designers of the TPU had to consider memory bandwidth, which is super important -- all the data flowing through the chip has to be passed efficiently.
- The choice of the instruction set is critical, as well.
  - Interestingly, although the general trend for processor architectures has been *Reduced Instruction Set Computer (RISC)* design, the TPU uses a *Complex Instruction Set Computer (CISC)* design, primarily because the multiply-and-add function is so important to the purpose of the chip that it made more sense to have a complex instruction to perform those calculations.
  - There are a dozen high-level instructions, such as Read_Host_Memory, Write_Host_Memory, Read_Weights, MatrixMultiply, and Activate (apply activation functions)

Lecture 20: Systems Today: Google's Tensor Processing Units (TPU)

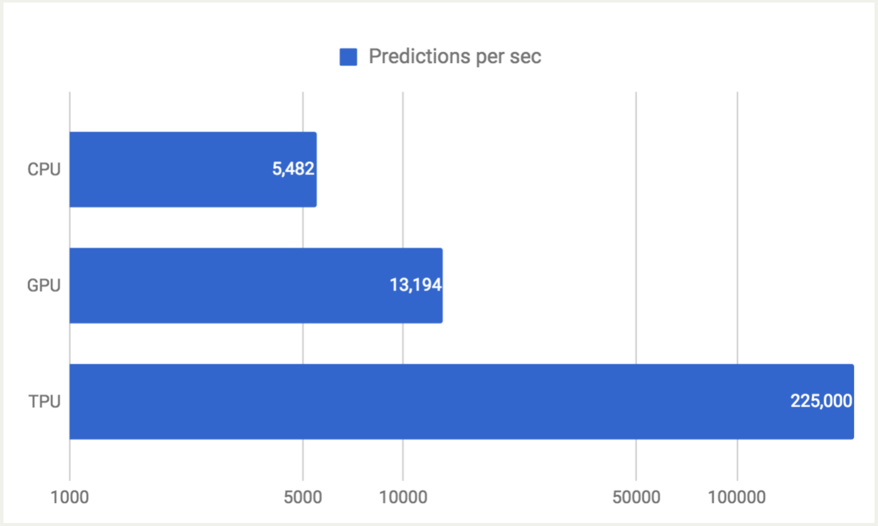

Unified Buffer for Local Activations (96Kx256x8b = 24 MiB) 29% of chip
Matrix Multiply Unit (256x256x8b=64K MAC) 24%
DRAM port ddr3 3%
Host Interf. 2%
Accumulators (4Kx256x32b = 4 MiB) 6%
DRAM port ddr3 3%
Control 2%
Activation Pipeline 6%
PCIe Interface 3%
Misc. I/O 1%

Floor Plan of TPU die(yellow = compute, blue = data, green = I/O, red = control)

https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

- The TPU floor plan is interesting -- much of the chip is dedicated to memory and computation, and a fair amount is dedicated to I/O. A very small amount is dedicated to control, which is different than CPUs and GPUs.
- The chip is also half the size of CPUs and GPUs.

- Does the TPU perform well against CPUs and GPUs? It seems that it does, by a large margin (this is TPU 1.0):
- Why? It is *application specific*. It was designed explicitly to perform neural network calculations. The instruction set (CISC), the quantization, and the matrix processor contribute to the performance for neural network calculations.



Lecture 20: Systems Today: Google's Tensor Processing Units (TPU)

Image: https://cloud.google.com/tpu/

Oh...one other thing: TPUs can work together in *pods* inside a data center.

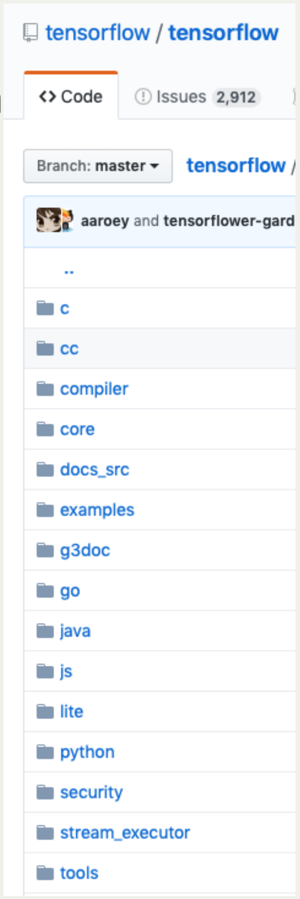
The latest version also handles floating point operations

Cloud TPU v3 Pod (beta)
100+ petaflops
32 TB High Bandwidth Memory
2-D toroidal mesh network

What's a *petaflop?* FLOPs are *"floating point operations per second"*, so a petaflop is 10^{15} floating point operations per second. Your laptop is in the 10s of Gigaflops (10^{10}) range, so a TPU Pod can perform roughly 10 million times as many floating point operations per second than your laptop...but it can't browse Reddit.

Lecture 20: Systems Today: Google's TensorFlow Framework

While we are on Google and their machine learning tools, let's talk about TensorFlow for a moment. TensorFlow is an open source library for machine learning that works on many platforms (Linux, Windows, MacOS, Raspberry Pi, Android and Javascript (!)) and on many devices (CPUs, GPUs, TPUs). It is written in Python, C++, and CUDA, meaning that there is a ton of systems that goes into it (esp. when interfacing with GPUs using CUDA). In fact, if you look at the GitHub repository, you'll see lots of language-specific folders:



tensorflow / tensorflow

<> Code | Issues 2,912

Branch: master ▾ tensorflow /

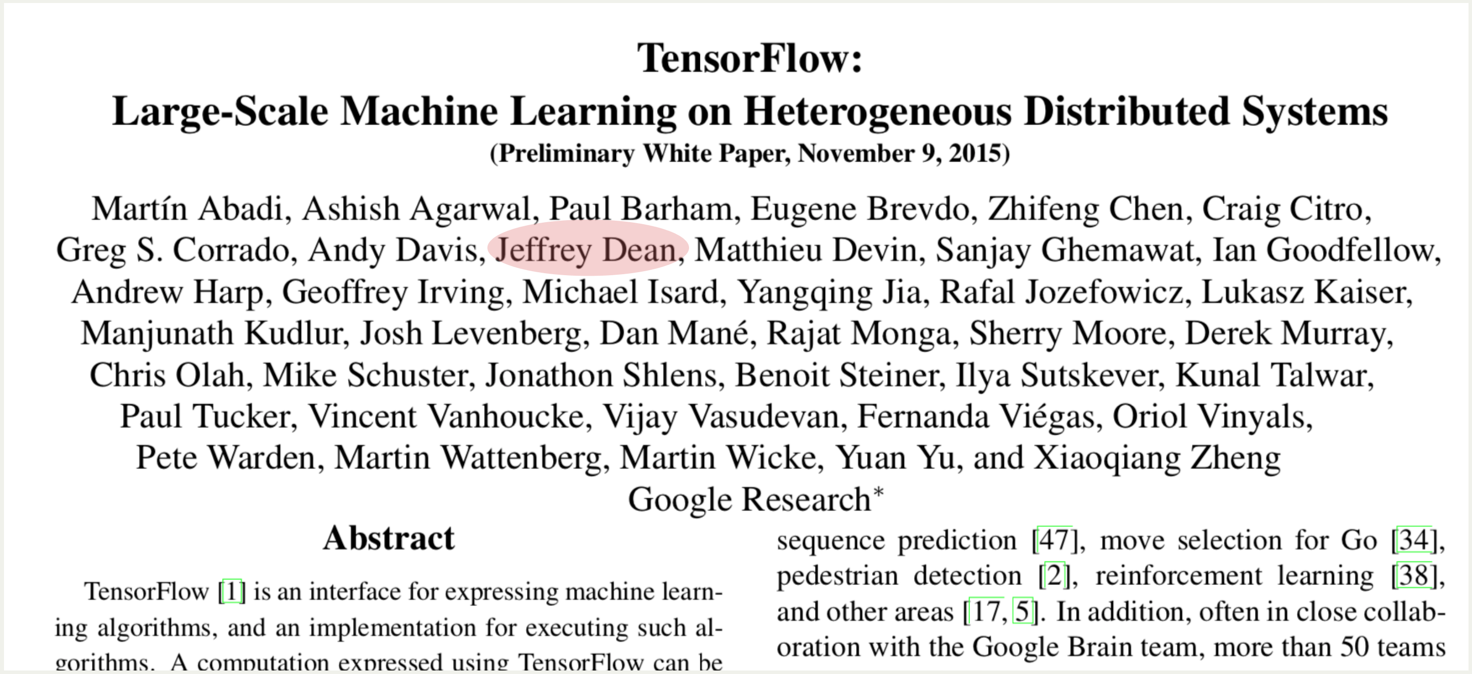
aaroey and tensorflower-gard

..
c
cc
compiler
core
docs_src
examples
g3doc
go
java
js
lite
python
security
stream_executor
tools

In other words, if you want to work on the TensorFlow library, you probably have to understand a lot of different languages, and be well-versed in systems.

Oh -- as Phil has mentioned before, the lead of Google AI, Jeff Dean (remember the MapReduce paper?) is a systems person, and he has played a big role in TensorFlow.

Lecture 20: Systems Today: Google's TensorFlow Framework



TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*

Abstract

TensorFlow [1] is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be

sequence prediction [47], move selection for Go [34], pedestrian detection [2], reinforcement learning [38], and other areas [17, 5]. In addition, often in close collaboration with the Google Brain team, more than 50 teams

http://download.tensorflow.org/paper/whitepaper2015.pdf

Lecture 20: Systems Today: Amazon’s Elastic Cloud Compute (EC2)

What if you want to run a server? What do you do?

- You could buy a computer, hire a hosting service, and hook it up.
  - You would (likely) have to manage it yourself, upgrade the software, manage users, etc. Probably not fun.
- You could rent a server from a hosting service
  - They will manage most of it for you. This works pretty well.

Let's say you take one of the options above and you set up TheCutestKittenPics.com, and all of the sudden it becomes a huge hit. Your web server gets slammed -- what do you do?

- You call up your hosting company and you either send them another server (or five), or you rent more servers from them. While you are making those decisions and handling the logistics, you are losing hundreds of dollars per hour on missed adorable kitten t-shirt sales. Boo!

Those were the options (mostly) until Amazon launched its Elastic Cloud Computer (EC2) service in 2006. EC2 allows users to run virtual machines or bare-hardware servers in the cloud, and it enables websites (for example) to scale quickly and easily.

Lecture 20: Systems Today: Amazon's Elastic Cloud Compute (EC2)

You might be surprised at some of the companies and other organizations that use EC2: Netflix, NASA, Slack, Adobe, Reddit, and many more.

Amazon (and its systems and IT people) handle all of the following:

- Infrastructure and Reliability
- Security
- Software (mostly Linux based)
- Storage
- Networking (to include static IP addresses)

You have complete *sudo* access to the machines, and if you break a virtual machine, you just start up a new one. If you need GPUs for your workload? EC2 has those, too.

Amazon also offers its *Lambda* service, which is a *serverless compute* solution -- you don't manage any servers, and you simply upload your code to Amazon, they run it, and they give you back the results.

Lecture 20: Systems Today: Amazon's Simple Storage Service (S3)

Amazon also offers a storage solution for Internet users, called the *Simple Storage Service (S3)*.

S3 provides the following features:

- Scalable storage (MB to TB and beyond)
- Backup and recovery
- Disaster recovery
- Data archives
- Data lakes for analytics (a *data lake* is generally data in raw form)

Amazon touts the following about S3:

- It is scalable
- It has high availability, with a guaranteed 99.99% uptime (which does leave 43 minutes per month for the system to be down)
- It has *low latency* (meaning that accessing a piece of data is fast)
- It has upwards of 99.999999999% durability (i.e., your data will not be destroyed)

All of the above features require a well-oiled system, and Amazon employs many systems people to innovate and to keep things running smoothly.

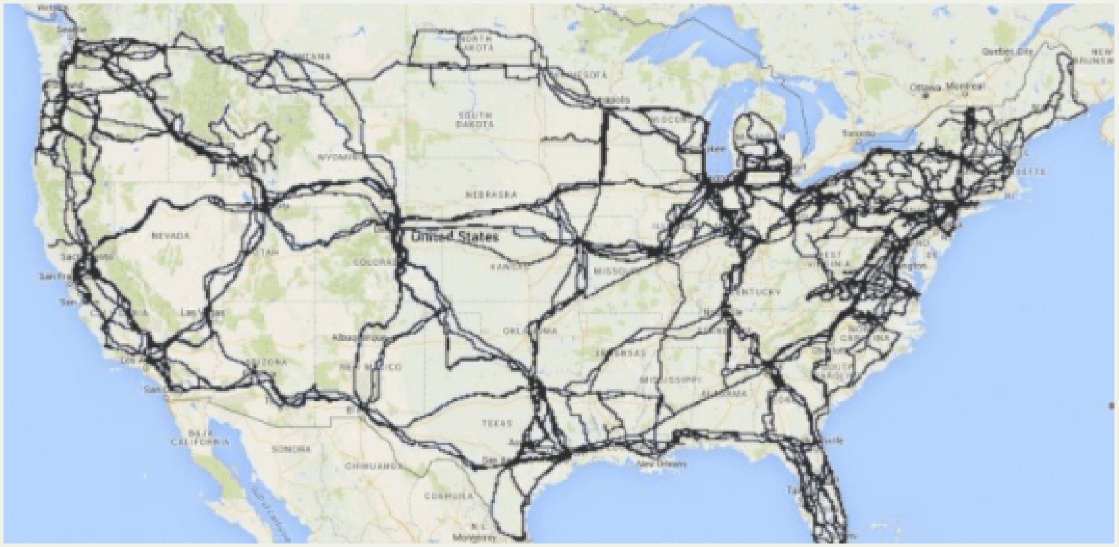
Lecture 20: Systems Today: The Internet Backbone

Who owns the Internet?

Lecture 20: Systems Today: The Internet Backbone

Who owns the Internet? No one!

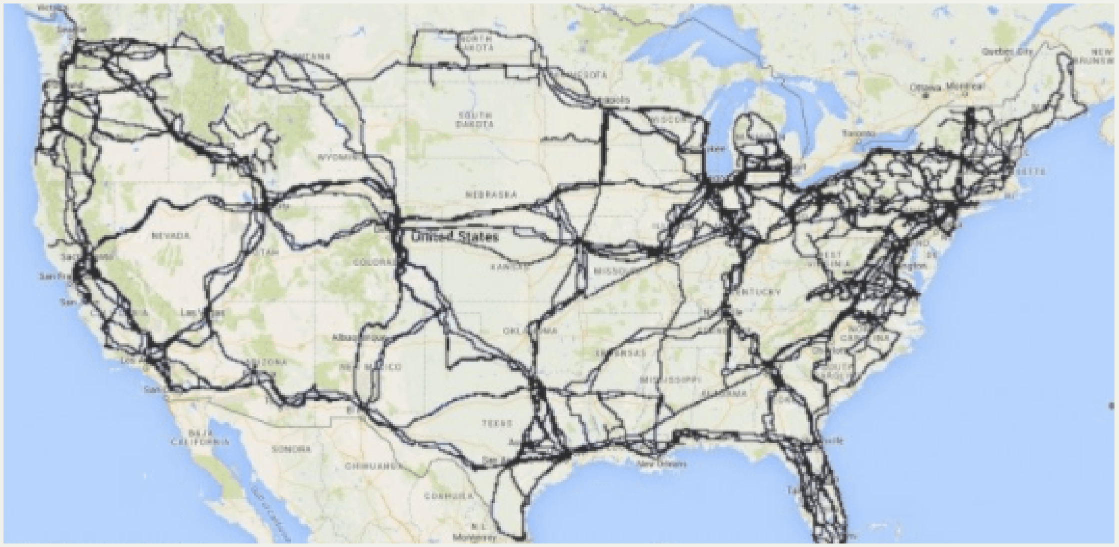
Well...lots of companies own data lines (mostly fiber optic) that connect the world through the Internet.



Source (overplayed over map of U.S.): InterTubes: A Study of the US Long-haul Fiber-optic Infrastructure

"The map shows the paths taken by the long-distance fiber-optic cables that carry Internet data across the continental U.S. The exact routes of those cables, which belong to major telecommunications companies such as AT&T and Level 3, have not been previously publicly viewable, despite the fact that they are effectively critical public infrastructure" (https://www.eteknix.com/first-detailed-public-map-u-s-internet-infrastructure-released/)

Lecture 20: Systems Today: The Internet Backbone

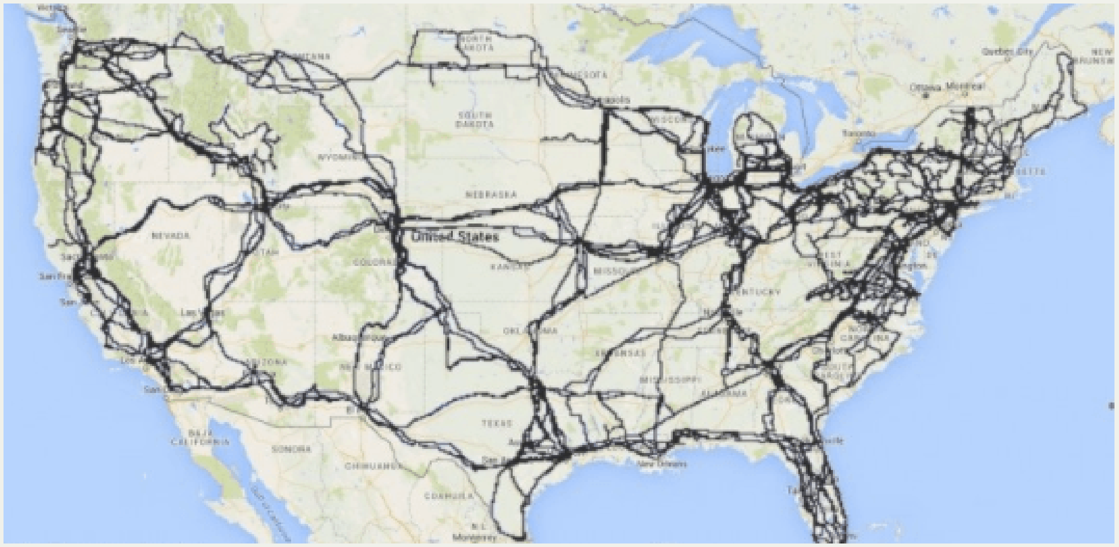


The so-called *Internet Backbone* is a nebulous route between major routers in the world, which allow Internet traffic to flow.

Many of the owners of the network connections are telephone companies, mainly because the Internet grew up using the same physical locations (telephone poles, underground cable runs, etc.). So, AT&T owns a lot of the network, as do Verizon, Sprint, and CenturyLink.

- Backbone providers have *peering agreements* that allows for the use of other companies' networks, and allows the handoff of data. There is usually not a fee for this, as the companies are charging their customers. This has led to issues where one company over-uses the network (e.g., Netflix pays backbone company Level 3, but there is so much traffic that other backbone companies want some money for it).

Lecture 20: Systems Today: The Internet Backbone



- The internet backbone needs to be fast -- throughput for fiber links are often 100Gbps. The Stanford wired network runs at 1Gbps, and your WiFi generally caps out at 450Mbps if you are lucky).
- The internet backbone also needs to be secure -- it is a prime candidate for attack.
- This is one reason why it is difficult to find out where the actual locations for the routers and centers.
- Some countries keep close control over inbound and outbound connections, and sometimes they cut off access outside the country completely (e.g., during the Arab Spring, Egypt cut off access for five days).
- There are many systems challenges -- just creating the hardware for a network that can handle 100Gpbs (or greater) traffic is difficult. For a 100Gbps network, a computer has to process a byte every 8×10^{-11} seconds, or a byte every 8 picoseconds...that is difficult.

Lecture 20: Systems Today: Systems Programming for Entertainment

- There is a tremendous amount of systems programming that goes into game development, animation, and other computer-based entertainment (e.g., simulators, virtual reality, etc.)
- In 1995, when *Toy Story* came out, it was reported that Pixar had 117 computers running 24 hours a day, and rendering an individual frame took between 45 minutes to 30 hours. There were a total of 114,240 frames to render.
- Part of Pixar's success was in its creation (by, among others, Stanford professor Pat Hanrahan) of the systems tools, networks, storage, and rendering software, called *Renderman*.
- Of course, animation has progressed, too -- for *Toy Story 4*, it can still take 60 to 160 hours to render one frame. Computers have gotten faster, and rendering software has gotten better, but the problem has scaled, too (i.e., Pixar is always trying to make certain parts of the animation more realistic, such as flowing hair).
- For computer games, the animation and 3-D views have gotten so complex that low-level systems programming has been necessary to create game engines that work fast enough to provide 60 frames-per-second play.
- Hardware designers for game consoles are always tweaking the hardware to make things run faster, and this is a systems challenge.

Lecture 20: Systems Today: Final Thoughts

- Systems programming is critical for most of the computing that is done today, from web browsing and email to huge AI and machine learning projects going on at many tech companies.
- Systems programming often supports other, higher-level tasks, but getting the systems part wrong can be devastating to the overall project.
- Systems are evolving every day, and as more advanced hardware comes online, systems programmers are going to have to learn new tools and methods. For example, quantum computing is coming online, and that will take a completely different type of systems programming.

- While you may or may not do much more systems programming after CS 110, hopefully you have an appreciation for the underlying systems nature of whatever you do end up doing.