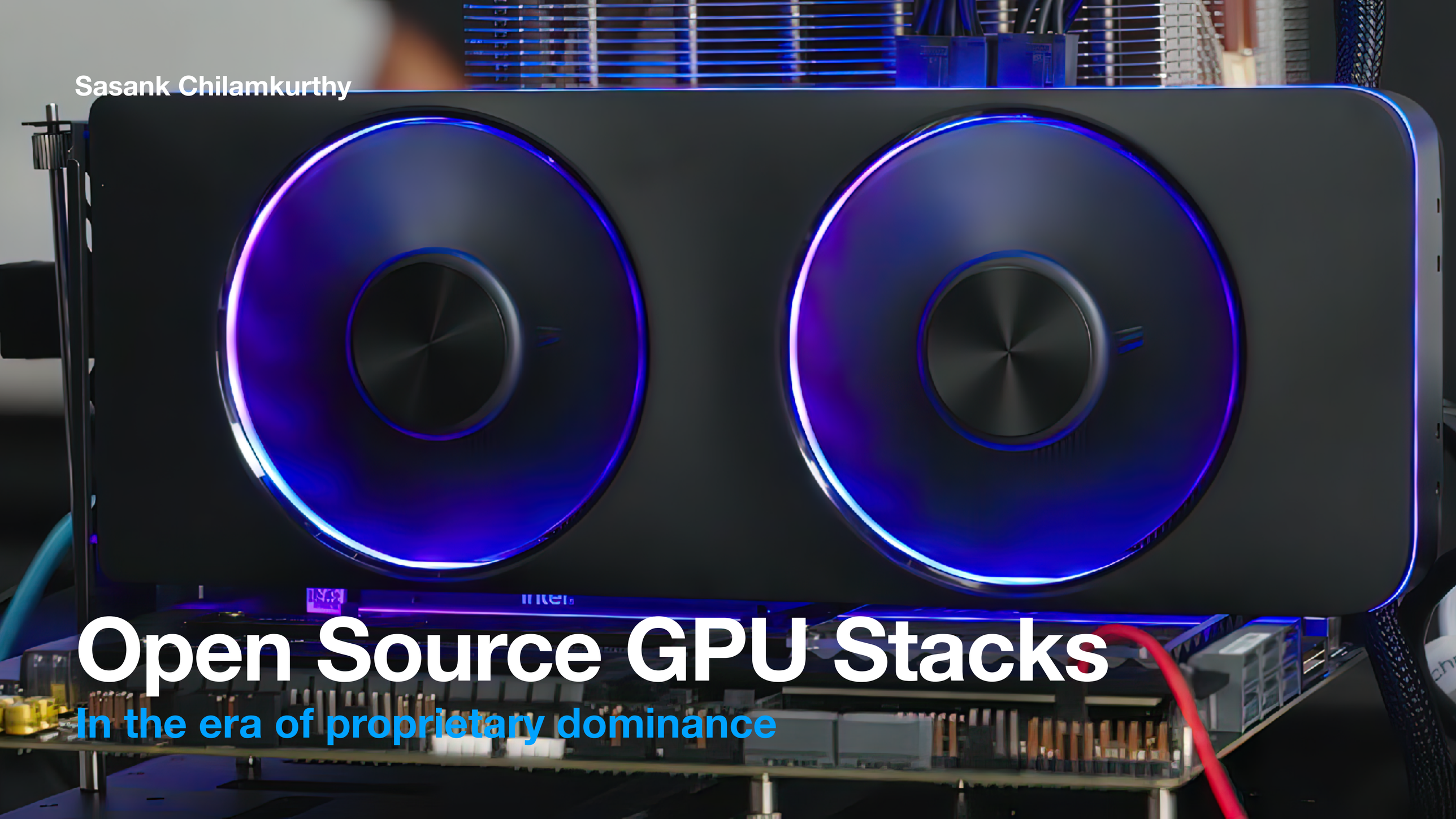
Sasank Chilamkurthy



Open Source GPU Stacks

In the era of proprietary dominance

Outline

- Proprietary dominance
- Computers
- Break the wall

Proprietary dominance

465.07 USD

+402.04 (637.85%) ↑ past 5 years

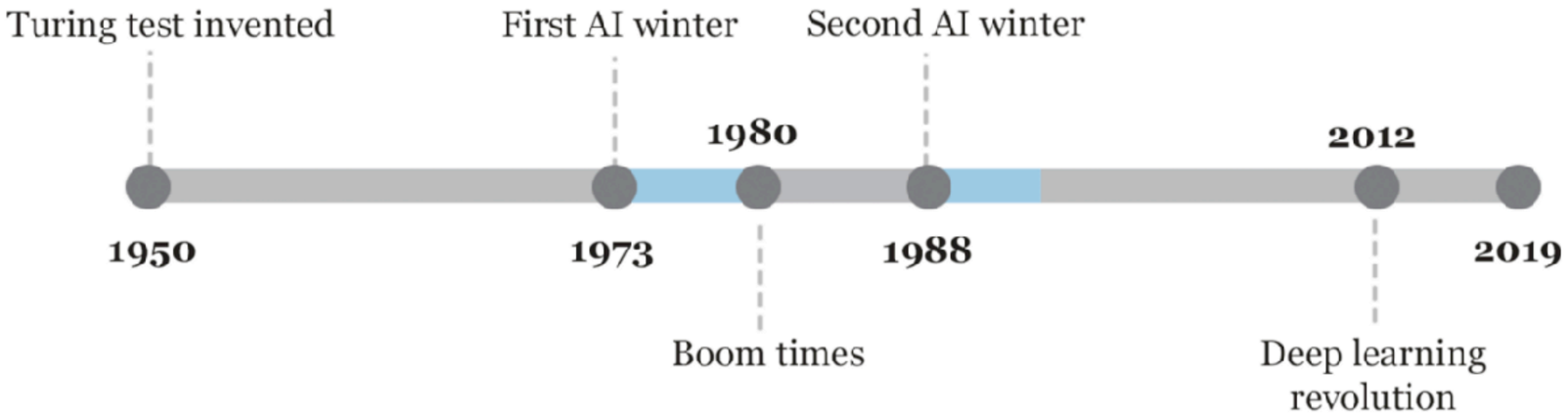
Closed: Aug 1, 7:59 PM EDT • Disclaimer

After hours 461.95 −3.12 (0.67%)

1D | 5D | 1M | 6M | YTD | 1Y | 5Y | Max

Open	464.60	Mkt cap	1.15T	CDP score	B
High	469.00	P/E ratio	241.70	52-wk high	480.88
Low	460.27	Div yield	0.034%	52-wk low	108.13

History of AI



Open research and open source code	Well, this hackathon
Huge amount of data	Opensource ImageNet
Huge amount of compute	GPUs

It all started with AlexNet

Neural Information Processing Systems
https://proceedings.neurips.cc › paper › 4824-i...

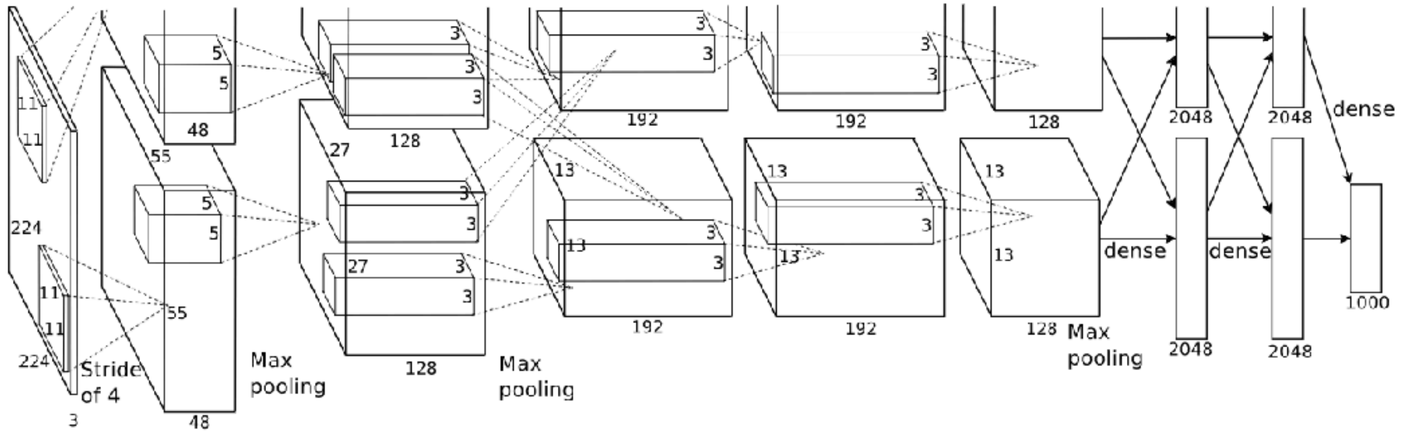
ImageNet Classification with Deep Convolutional Neural ...

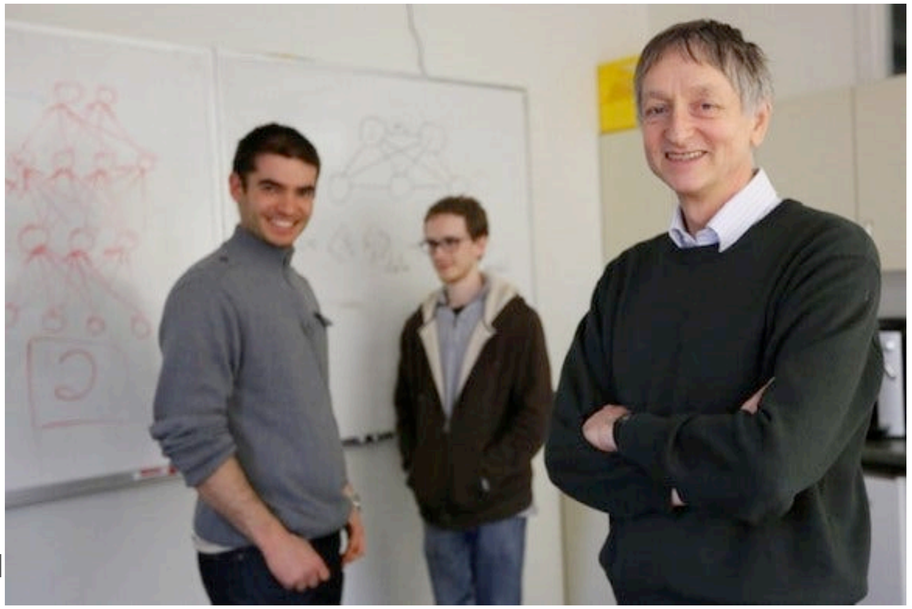
by A Krizhevsky · Cited by 119294 — We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the…
9 pages

Year 2012
100k citations!

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.





https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

AI runs on GPUs

- AI = matrix multiplications, which is massively parallelizable
- GPUs are great at parallel programming
- CPU < 32 cores/threads, GPU> 4000 cores/threads!
- CPU is 10x slower, at least
- Impractical to train or even run any reasonable AI model outside GPUs and ASICs

CUDA is de facto standard

- CUDA is C-like language to program a GPU
- All AI programs are written in Nvidia’s GPGPU language CUDA
- Works only on Nvidia GPUs
- Therefore AI stuff runs only on Nvidia GPUs
- AI hardware is **monopoly** because of lack of good compilers!

AlexNet was done in CUDA of course

contain enough labeled examples to train such models without severe overfitting.

The specific contributions of this paper are as follows: we trained one of the largest convolutional neural networks to date on the subsets of ImageNet used in the ILSVRC-2010 and ILSVRC-2012 competitions [2] and achieved by far the best results ever reported on these datasets. We wrote a highly-optimized GPU implementation of 2D convolution and all the other operations inherent in training convolutional neural networks, which we make available publicly[1]. Our network contains a number of new and unusual features which improve its performance and reduce its training time, which are detailed in Section 3. The size of our network made overfitting a significant problem, even with 1.2 million labeled training examples, so we used several effective techniques for preventing overfitting, which are described in Section 4. Our final network contains five convolutional and three fully-connected layers, and this depth seems to be important: we found that removing any convolutional layer (each of which contains no more than 1% of the model's parameters) resulted in inferior performance.

In the end, the network's size is limited mainly by the amount of memory available on current GPUs

Project

Source

Issues

Wikis

Downloads

cuda-convnet

High-performance C++/CUDA implementation of convolutional neural networks

Note July 18, 2014: * I've released an update to `cuda-convnet`, called cuda-convnet2. The two main new features are faster training on Kepler-generation GPUs and support for multi-GPU training.

This is a fast C++/CUDA implementation of convolutional (or more generally, feed-forward) neural networks. It can model arbitrary layer connectivity and network depth. Any directed acyclic graph of layers will do. Training is done using the back-propagation algorithm.

Fermi-generation GPU (GTX 4xx, GTX 5xx, or Tesla equivalent) required.

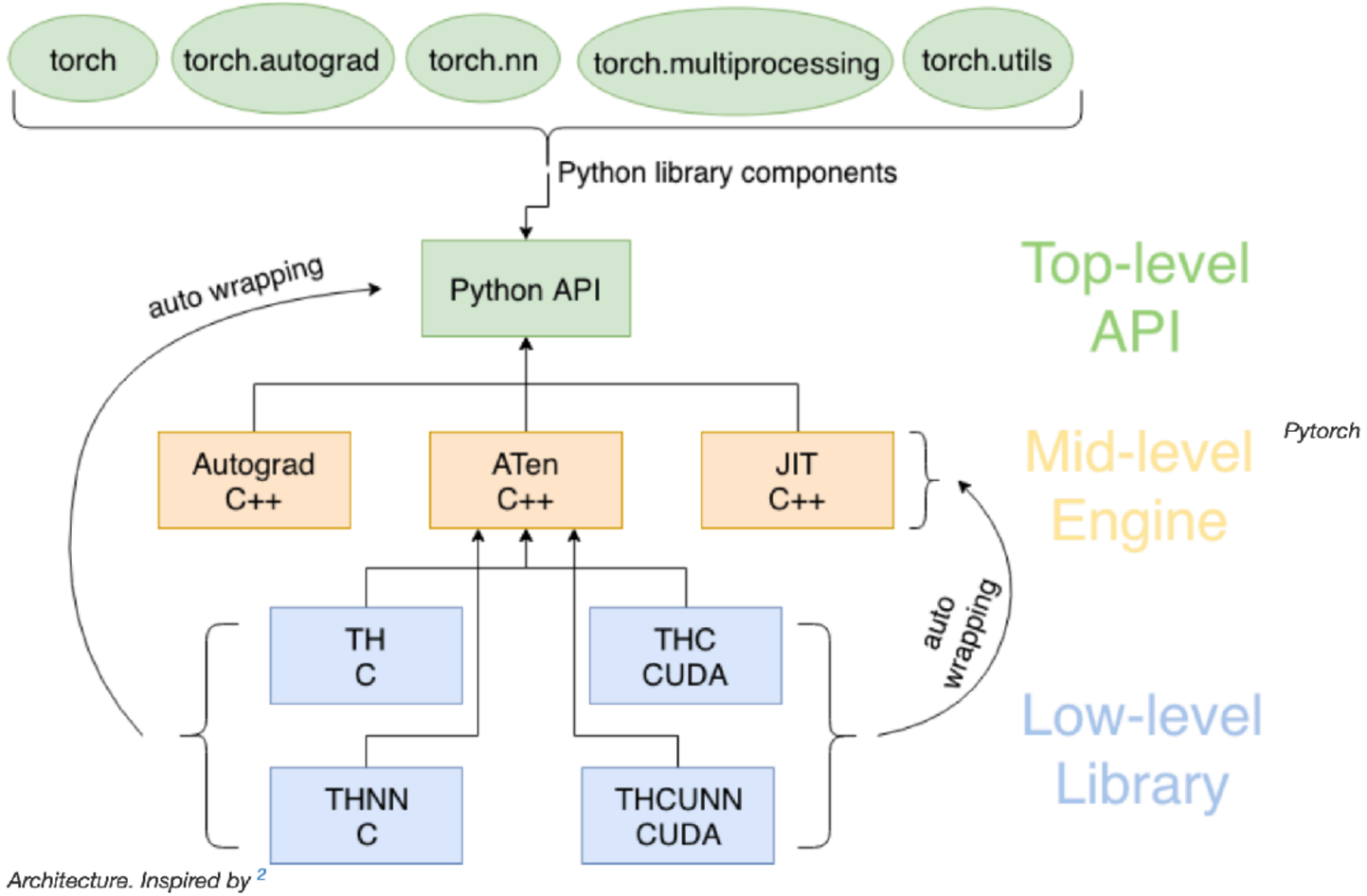
Documentation

- Compiling – how to check out and compile this code.
- Data -- what kind of data this net can train on.
- LayerParams – how to specify an architecture for the net.
- NeuronTypes – types of hidden unit nonlinearities.
- TrainingNet – how to train the net.
- Options – the command-line arguments that the net takes.
- ViewingNet -- how to look inside the checkpoints saved by the net.
- CheckingGradients -- how to numerically test the gradients for correctness.

PyTorch dominates AI frameworks

Written in C++ & CUDA but with Python API

The main structure of PyTorch in a architectural view is shown in the figure below.

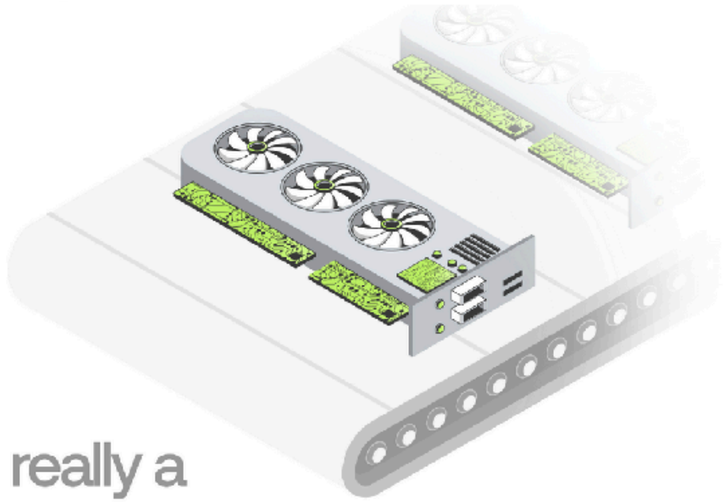


Architecture. Inspired by [2]

https://se.ewi.tudelft.nl/desosa2019/chapters/pytorch/#fn:3

Nvidia H100 GPUs: Supply and Demand

July 2023 · Updated: August 2023



Is there really a

Bottleneck?

How Many GPUs Are Needed?

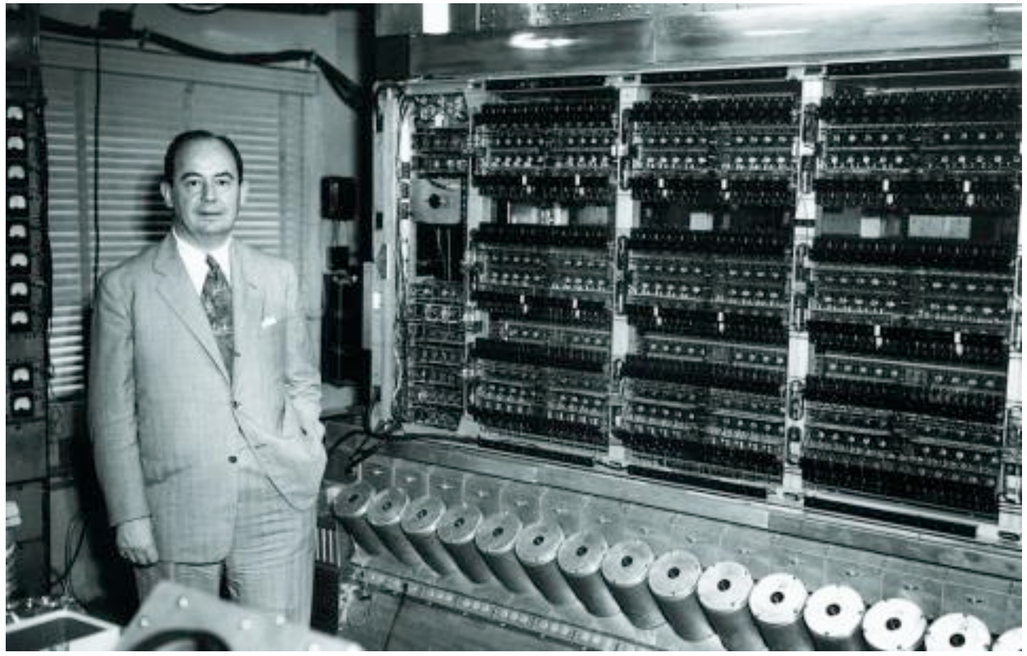
- GPT-4 was likely trained on somewhere between 10,000 to 25,000 A100s.[20]
- Meta has about 21,000 A100s, Tesla has about 7,000 A100s, and Stability AI has about 5,000 A100s.[21]
- Falcon-40B was trained on 384 A100s.[22]
- Inflection used 3,500 H100s for their GPT-3.5 equivalent model.[23]

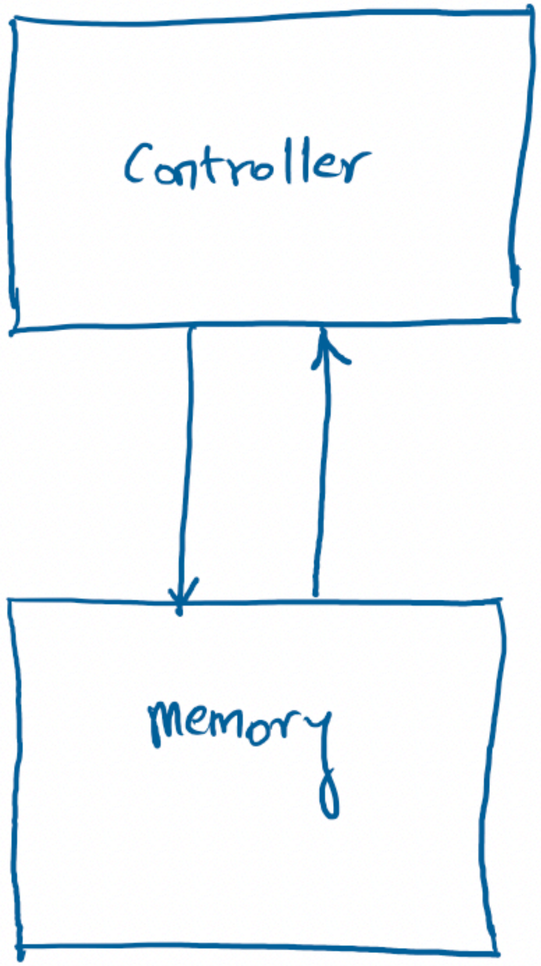
Specifications	
GPU	8x NVIDIA H100 Tensor Core GPUs
GPU memory	640GB total
Performance	32 petaFLOPS FP8
NVIDIA® NVSwitch™	4x
System power usage	10.2kW max
CPU	Dual Intel® Xeon® Platinum 8480C Processors 112 Cores total, 2.00 GHz (Base), 3.80 GHz (Max Boost)
System memory	2TB
Networking	4x OSFP ports serving 8x single-port NVIDIA ConnectX-7 VPI > Up to 400Gb/s InfiniBand/Ethernet 2x dual-port QSFP112 NVIDIA ConnectX-7 VPI > Up to 400Gb/s InfiniBand/Ethernet
Management network	10Gb/s onboard NIC with RJ45 100Gb/s Ethernet NIC Host baseboard management controller (BMC) with RJ45
Storage	OS: 2x 1.92TB NVMe M.2
Internal storage:	8x 3.84TB NVMe U.2

500k USD / DGX H100
30k USD/Card
Almost a million H100s ordered for the next year

Computer Architecture

von Neumann architecture



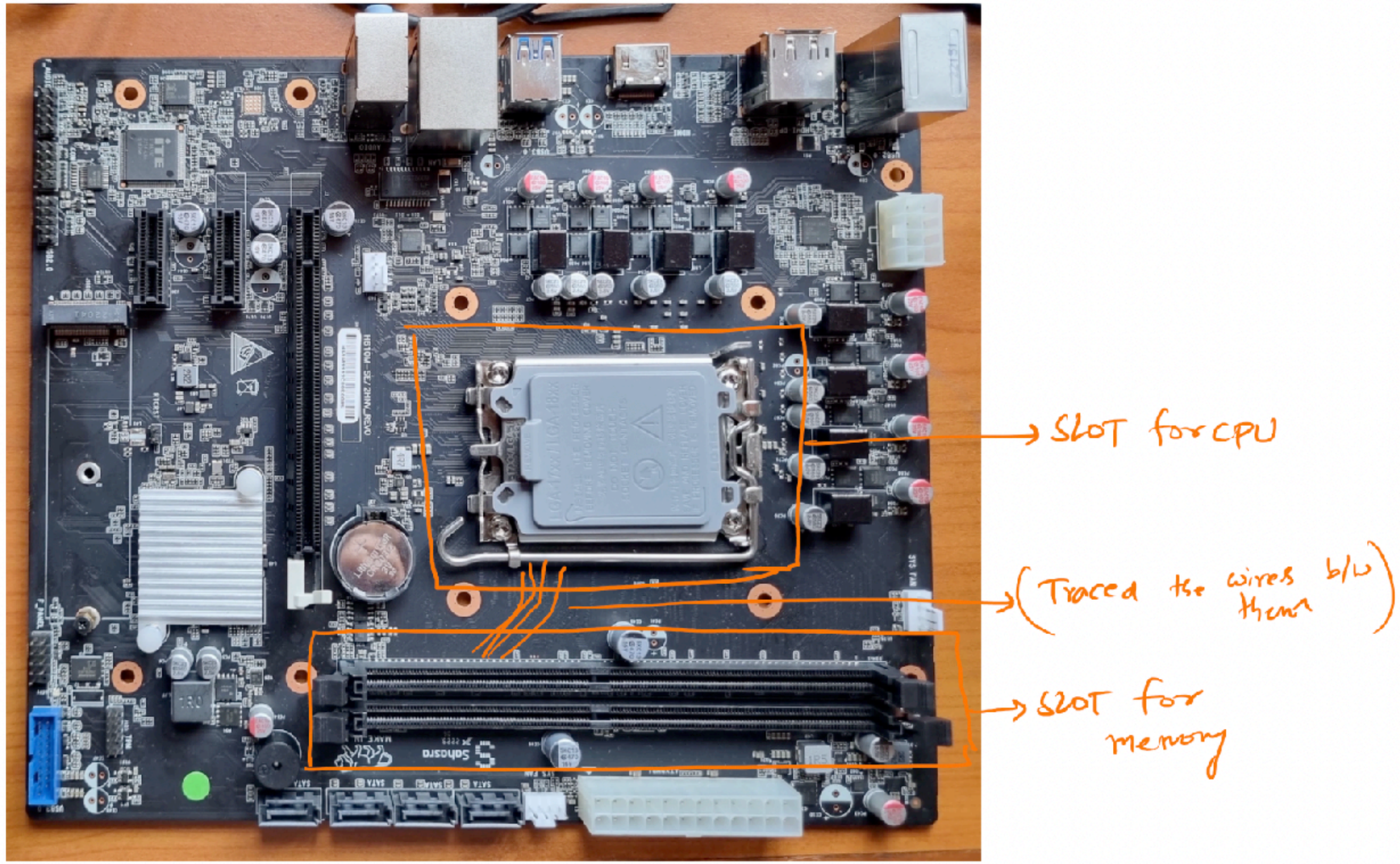


Interprets part of memory as program and executes it on data

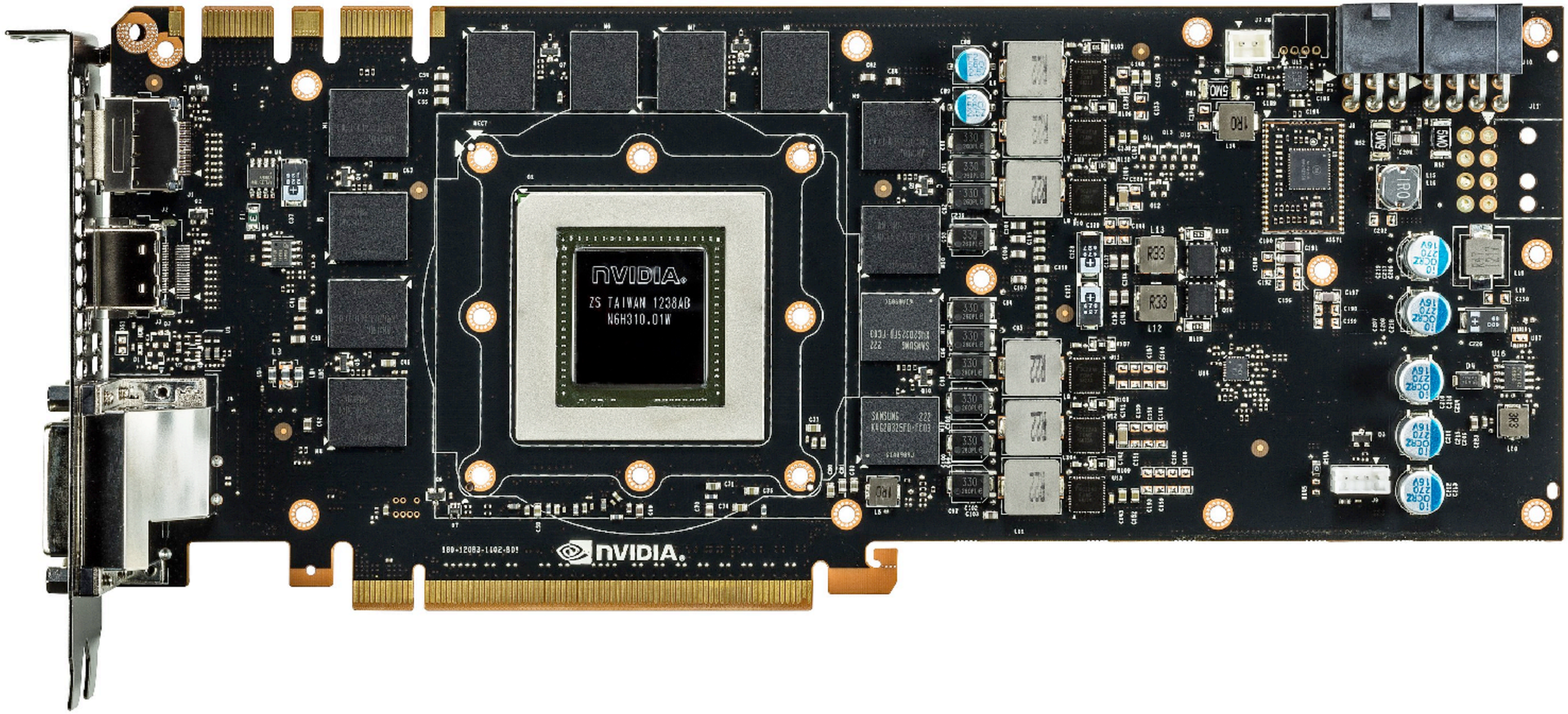
Both program and data is stored together

https://chsasank.com/llm-system-design.html

Basis of all Computers



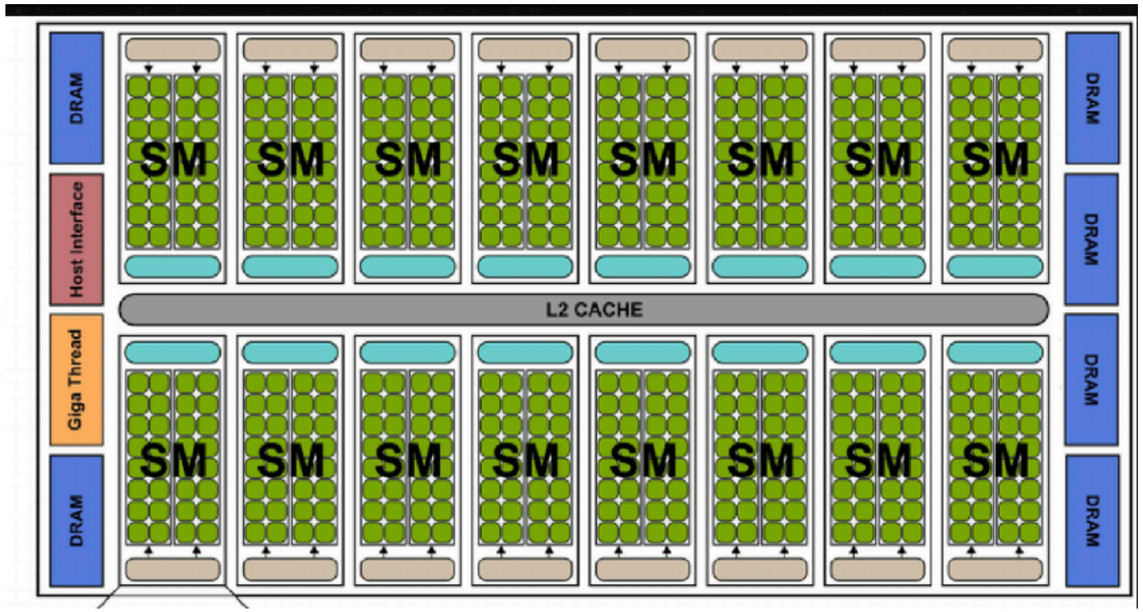
Basis of all GPUs

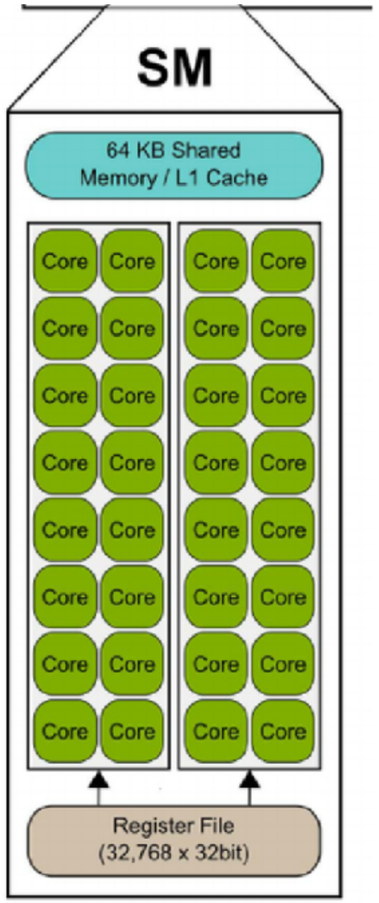


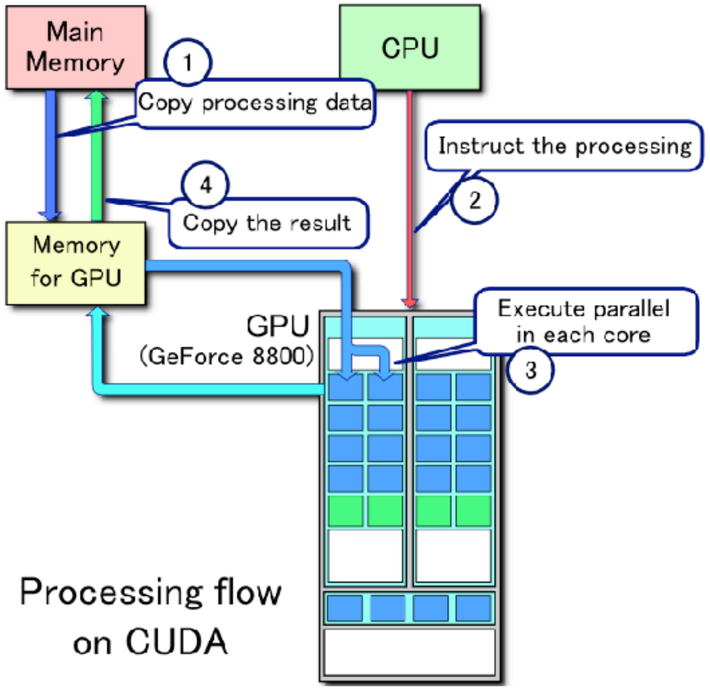
https://simple.wikipedia.org/wiki/Video_card

GPU Architecture

GPU = Multi core processors with support for hardware support for multi threading



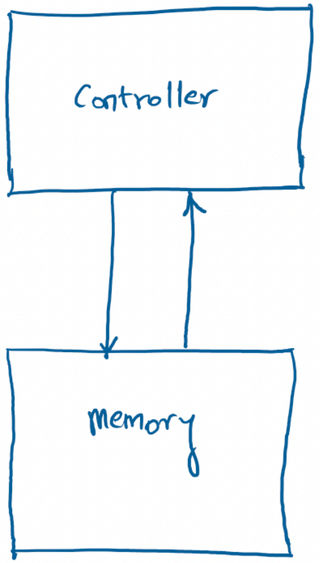




Note Memory Hierarchy

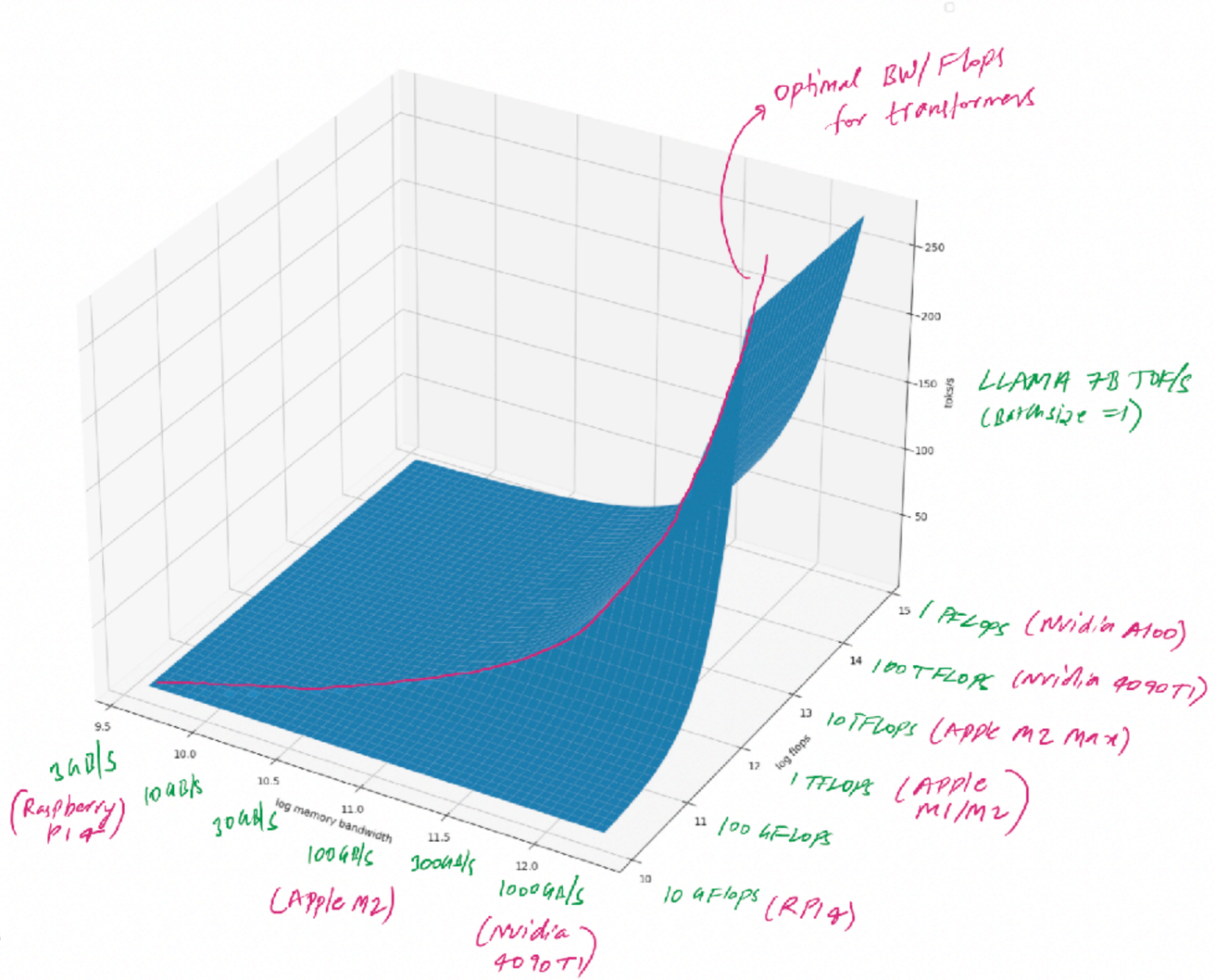
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0061892

Optimal hardware design



Interprets part of memory as program and executes it on data

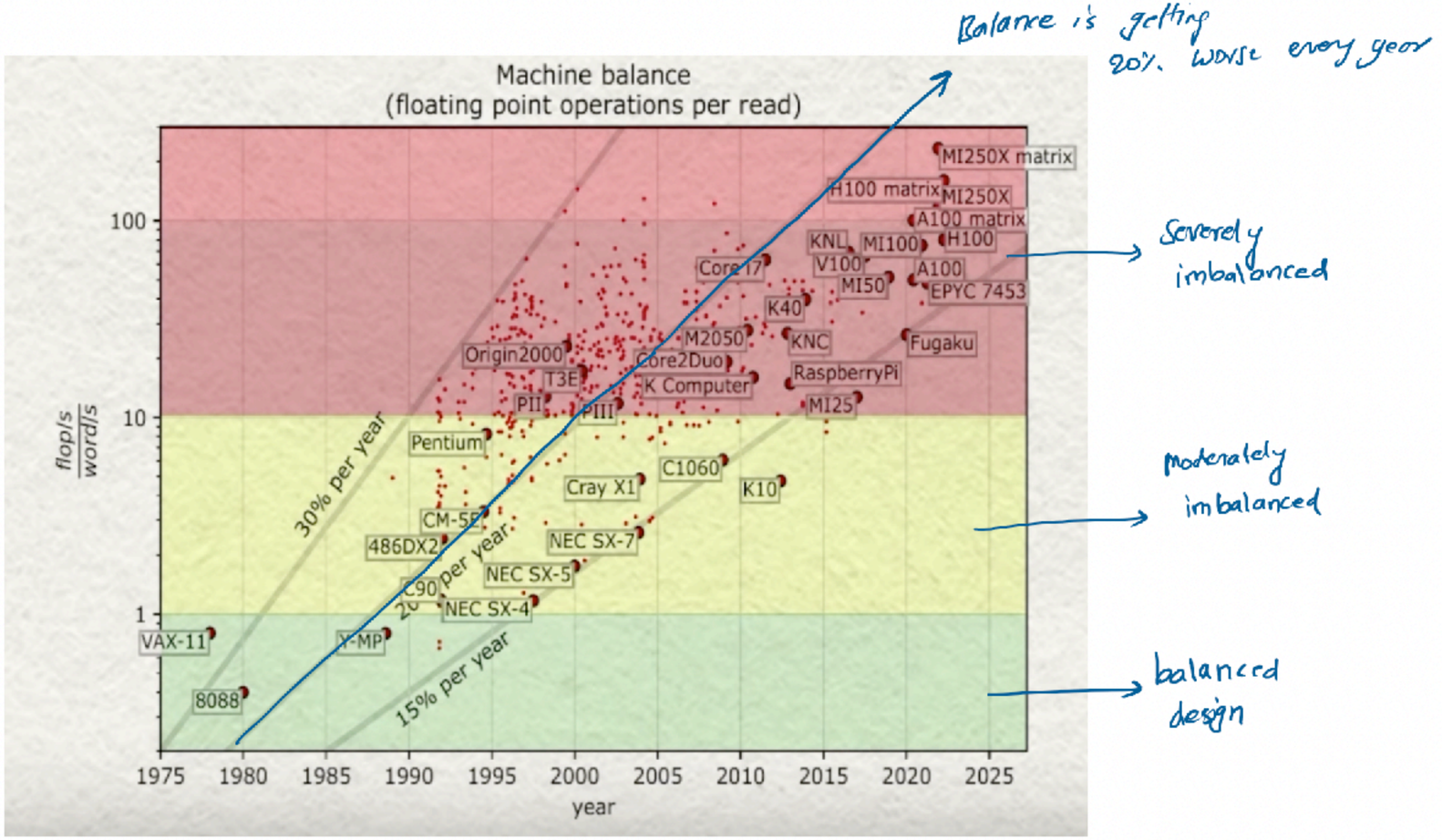
Both program and data is stored together



Optimal FLOPs/BW = Batch Size

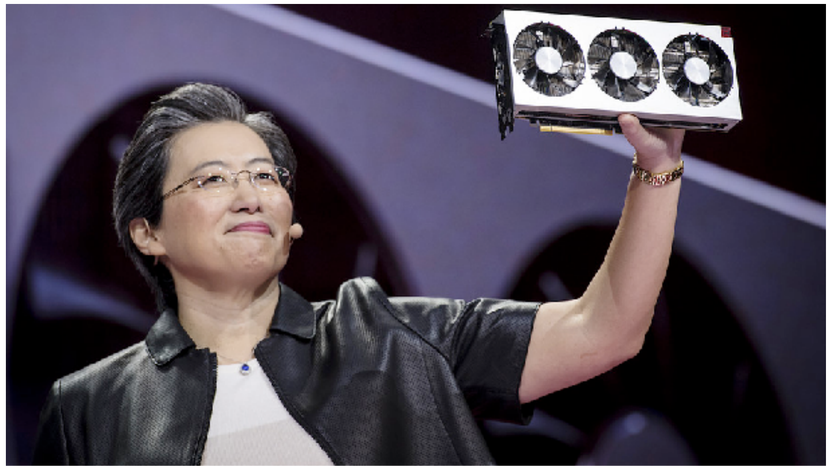
FLOPs/BW is getting worse

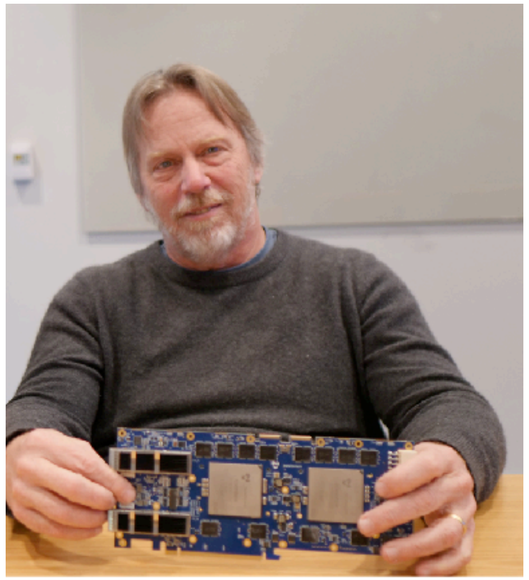
We need newer designs for inference

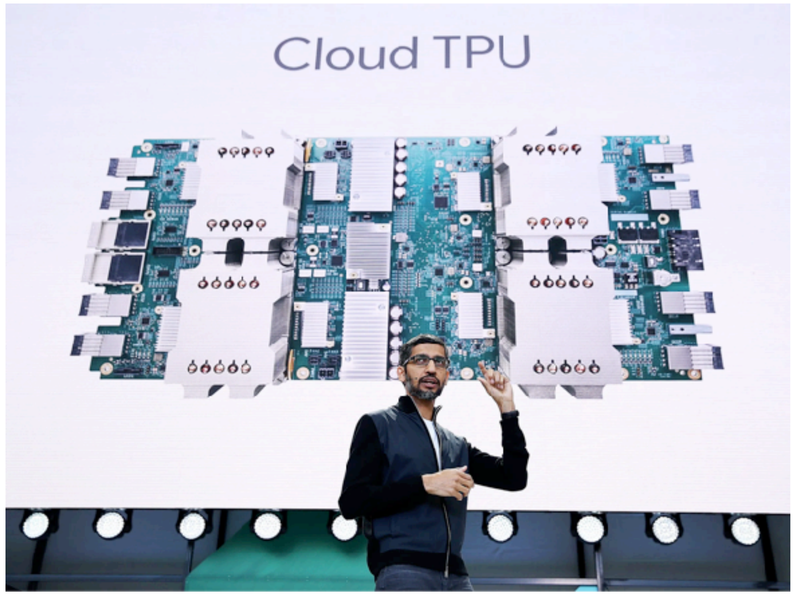


AI Chips

Cambrian Explosion



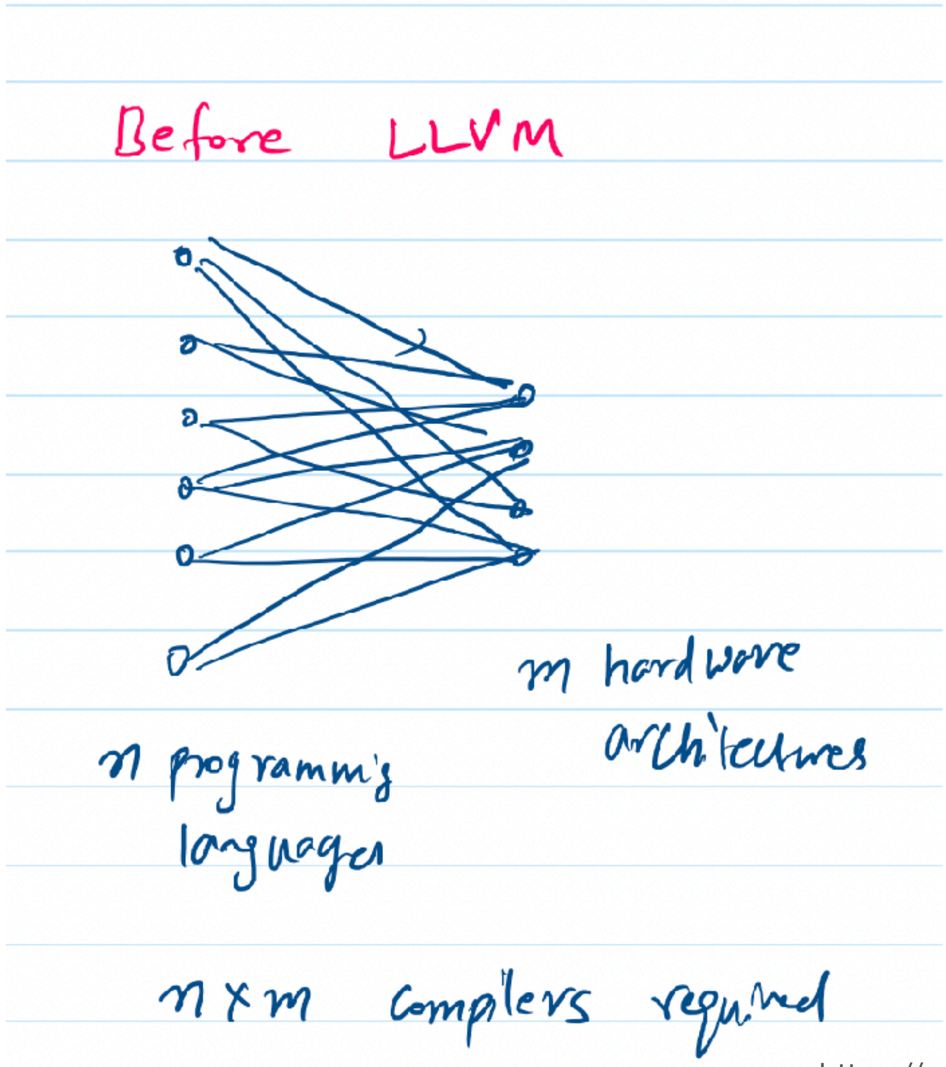


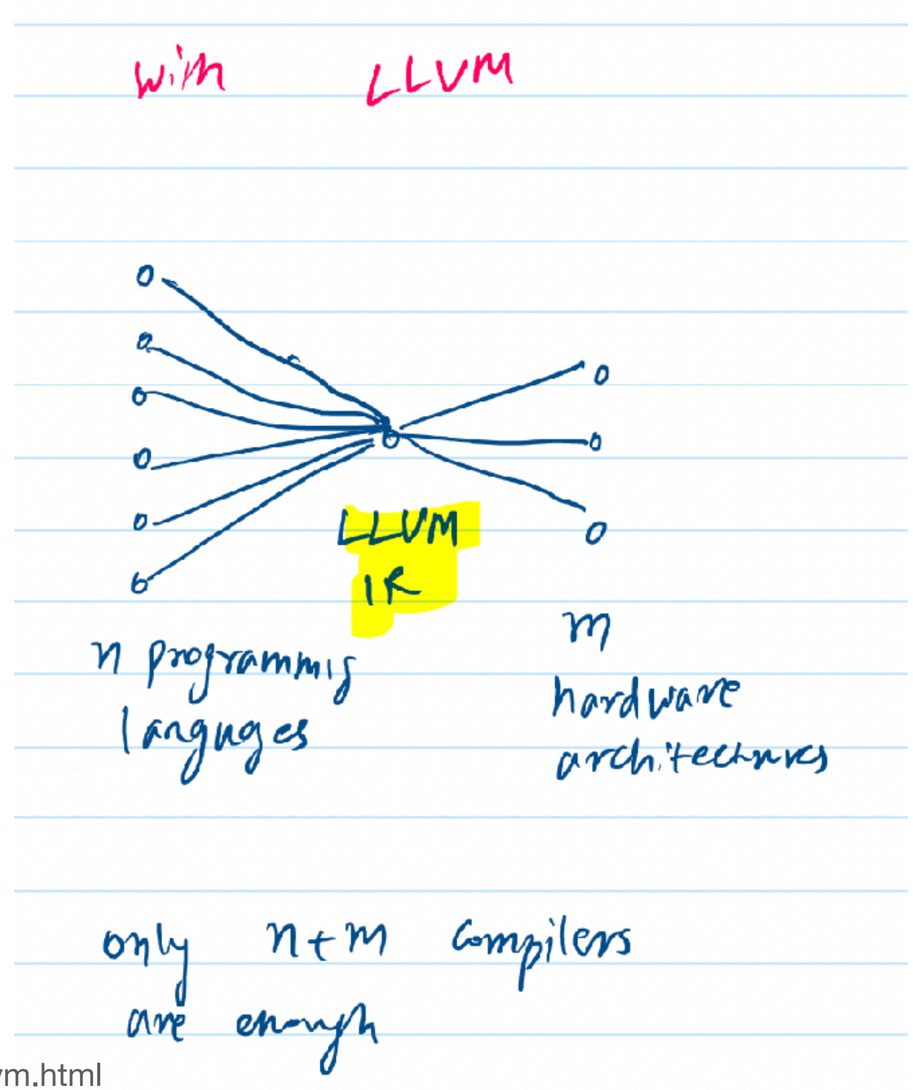



Break the wall

How to support newer architectures?

Common intermediate representation for all programming languages and hardware

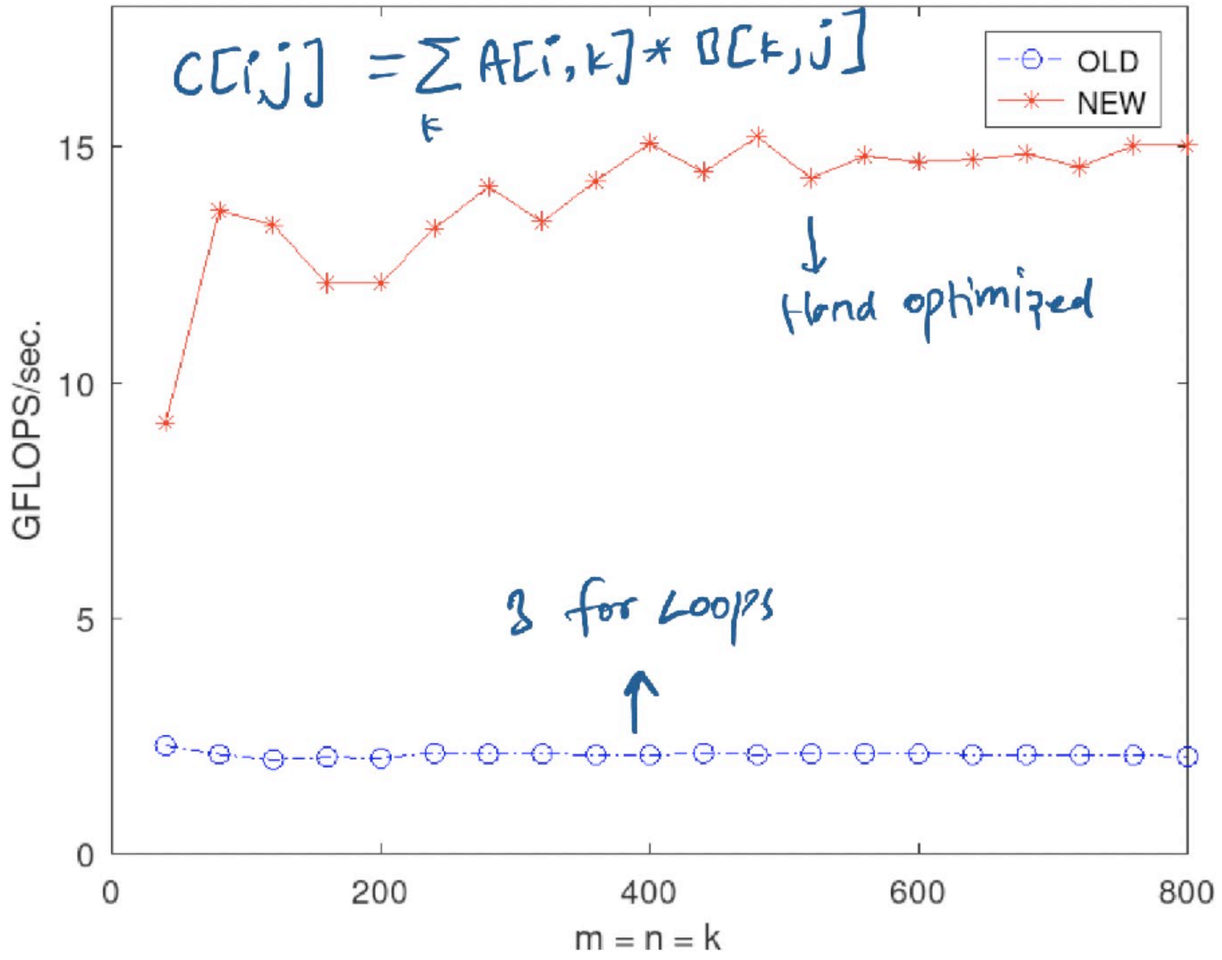




https://aosabook.org/en/v1/llvm.html

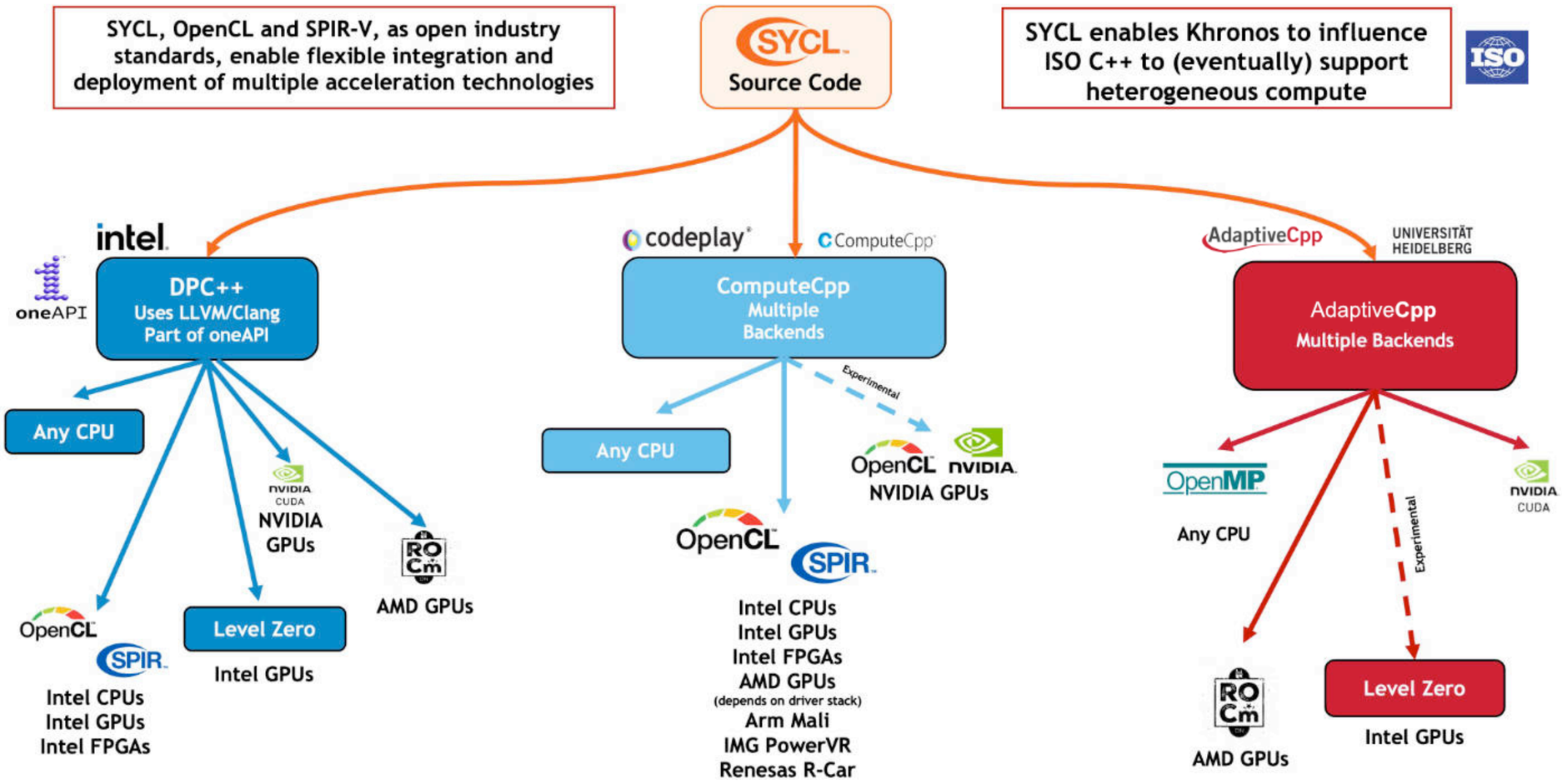
Kernels are hard

Another point of lock-in



SYCL: A Portable Alternative to CUDA

Standard, not implementation



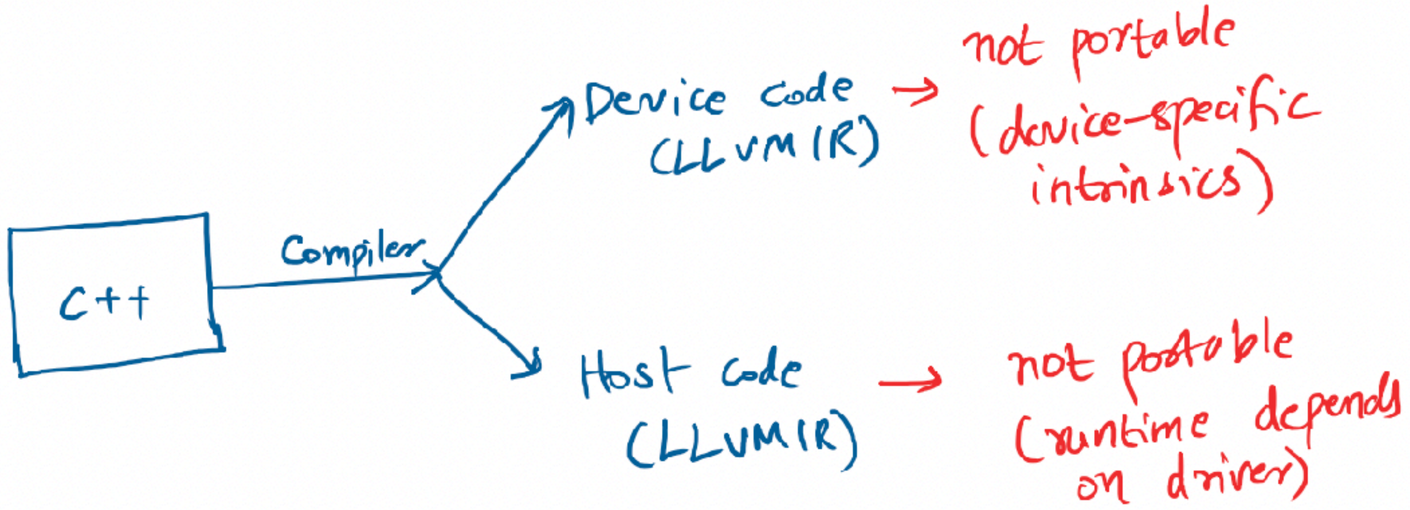
SYCL Works Great

GPU	Matrix Size	PortBlas GFLOP/s	Vendor Libraries GLOP/s	PortBlas/ Vendor
Nvidia GTX 1650M	1024	1284	1483	87%
	2048	2299	2700	85%
	4096	2475	1889	131%
AMD Van Gogh	1024	451	889	51%
	2048	911	689	132%
	4096	989	1199	82%
Intel Arc 770	1024	7210	5271	137%
	2048	8473	1511	561%
	4096	8408	16425	51%

https://chsasank.com/portblas-portable-blas-across-gpus.html

LLVM IR doesn’t work

For GPUs



Sasank Chilamkurthy
@sasank51

In search for portable CUDA alternative, I found that LLVM doesn't really cut it as intermediate representation. Read why in my latest post:
chsasank.com/intermediate-r...

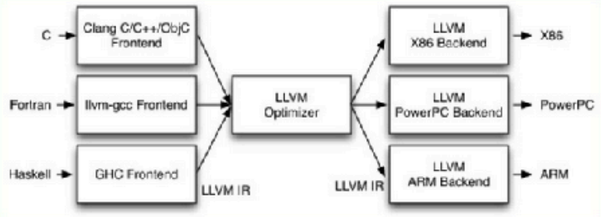
Intermediate Representations for GPUs: LLVM Does Not Cut it

Sasank Chilamkurthy | 05 April 2024 | 11 minutes to read.

- Compilers are like dragons, and wrapping my head around their complexity has been challenging. Adding to the challenge, I’ve chosen a particularly tough topic within this complexity: AI compilers. What sets AI apart are GPUs and matrix multiplication kernels. In this post, I will talk about compilers for GPUs and will leave matrix multiplication kernels to another post. In this post, we will examine LLVM compiler framework for CPUs and contrast it with for GPUs. We’ll show that LLVM is not a reasonable IR for GPU.

How LLVM works

A good review of architecture of LLVM can be found in the book The Architecture of Open Source Applications. I reproduce a key diagram from the LLVM chapter below for reference:



3:31 PM · Apr 5, 2024 · **29.3K** Views

View post engagements

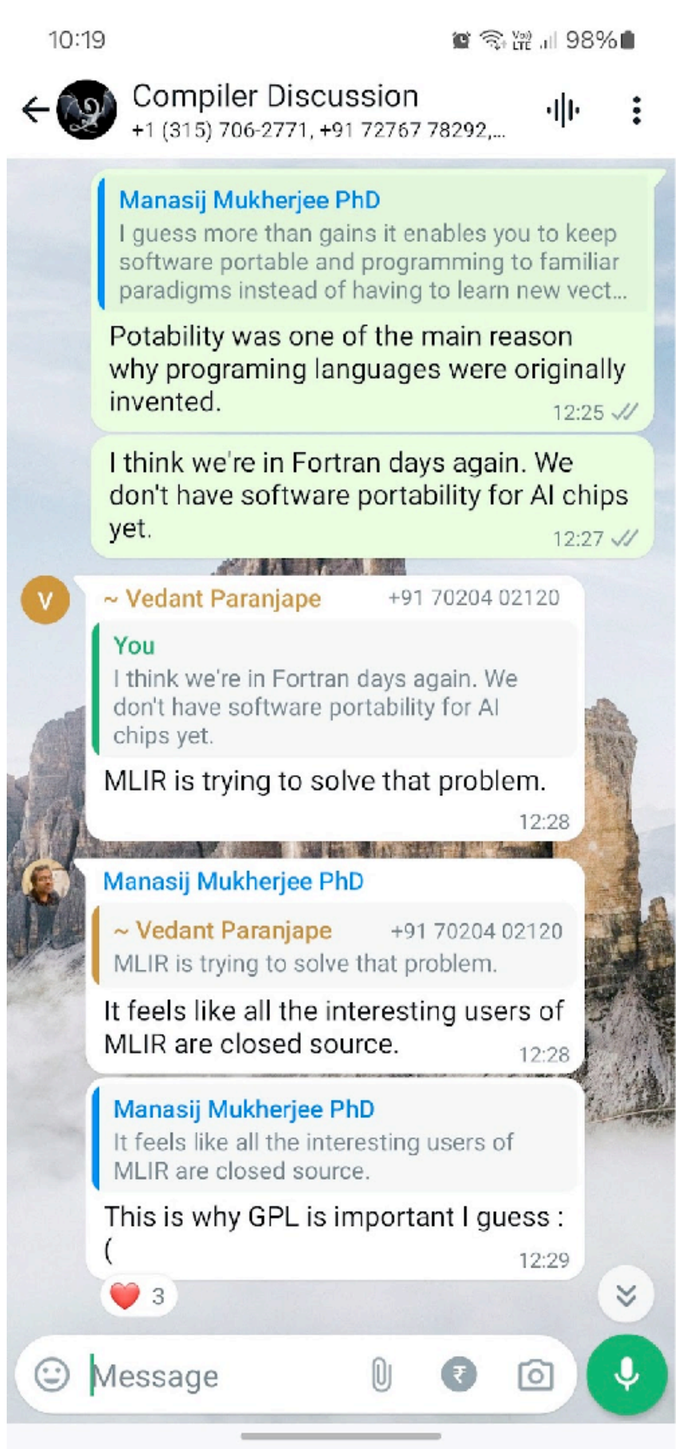
11 53 272 215

GPL License

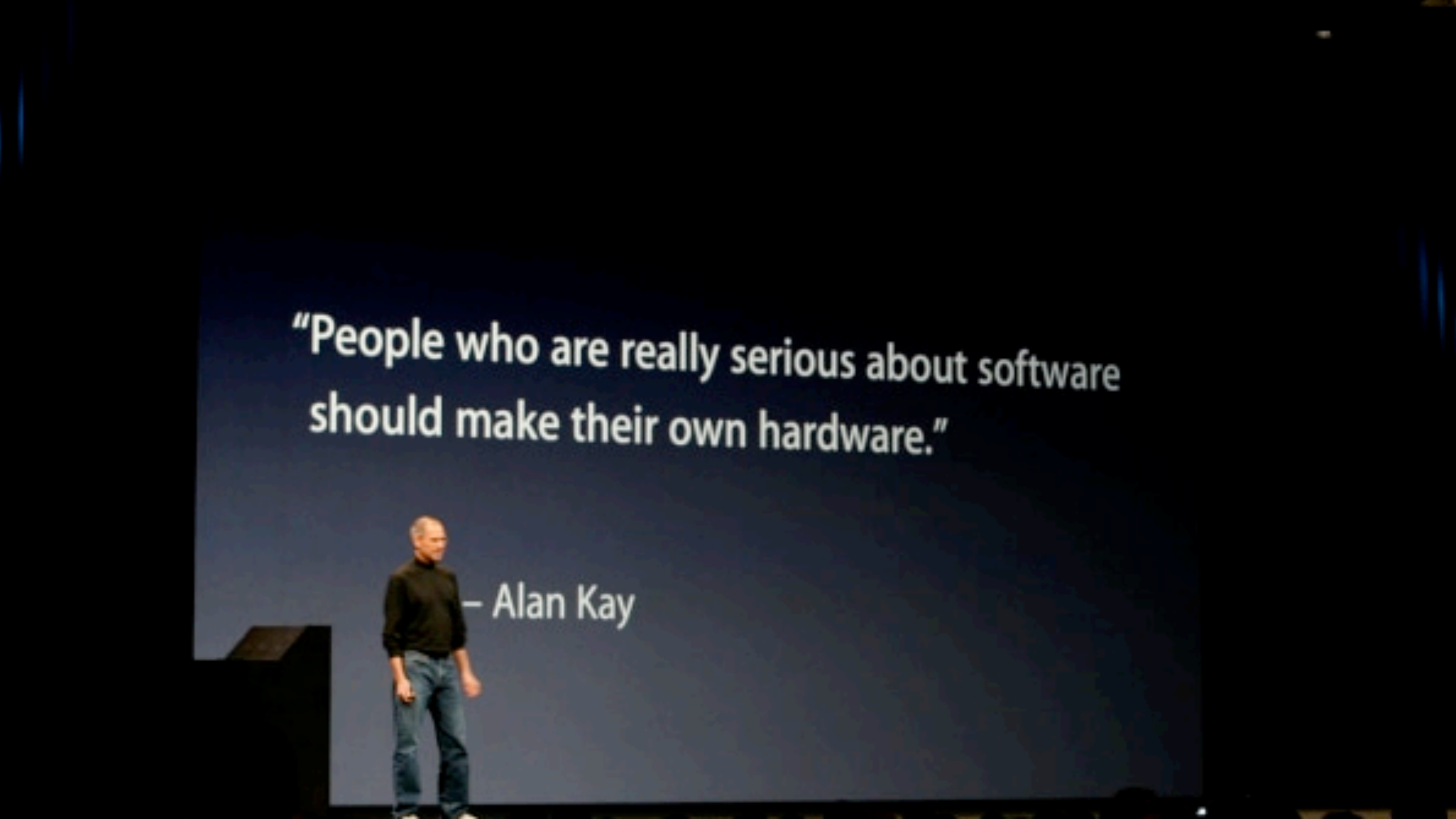
Free the software

GPL v3 Free Software

Free as in Freedom



https://twitter.com/elonmusk/status/1765387202953937224


"People who are really serious about software
should make their own hardware."
– Alan Kay

I build hardware

VON NEUMANN
AI