TVM Stack

前景如何？现在入还来得及么？

Wei Wu

2019-07-20

结论

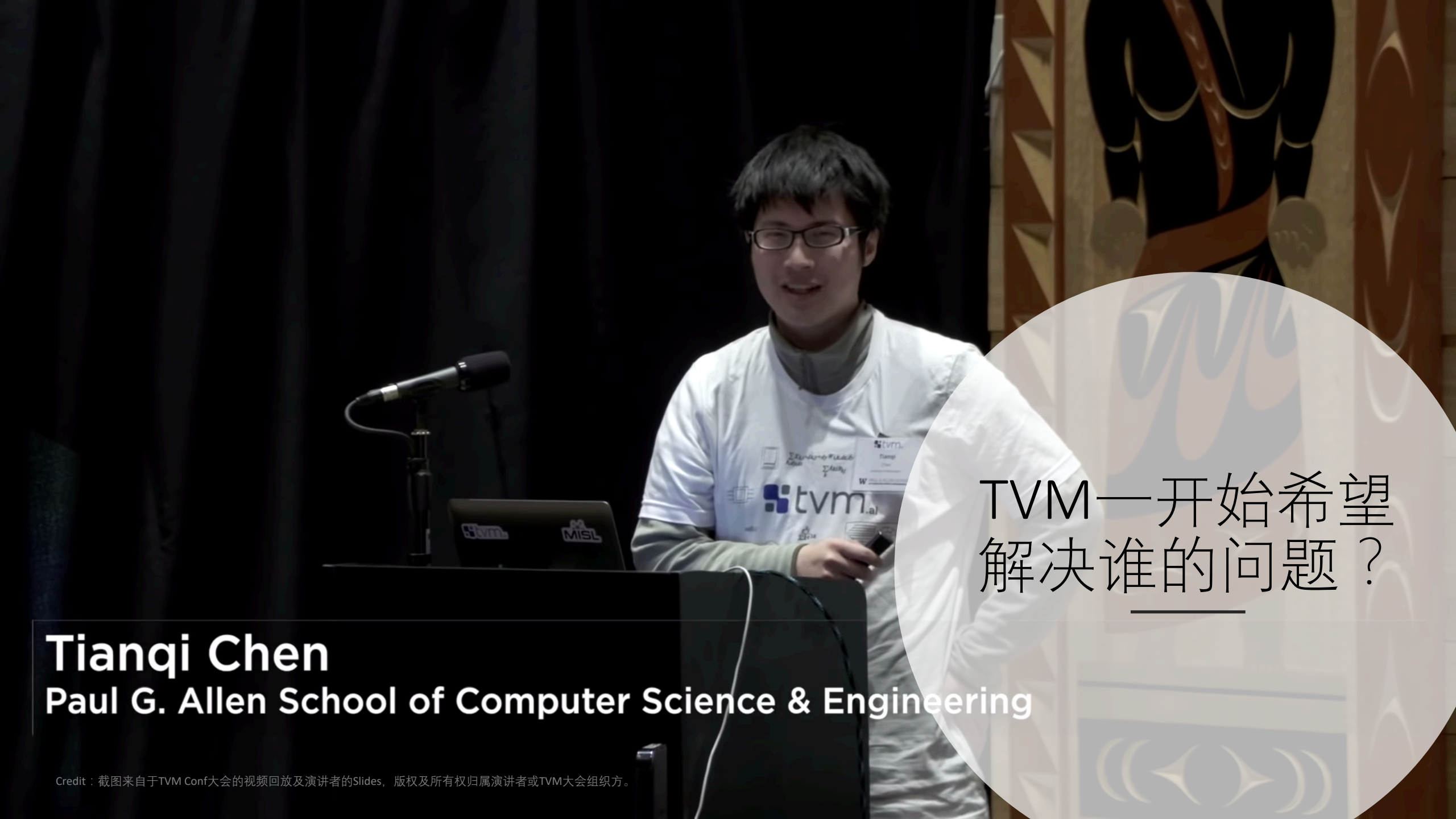
- 来得及，快上车！

结论

- 来得及，快上车！
- 我个人很看好，
- 是我司（重德智能）重点方向

TVM 是什么

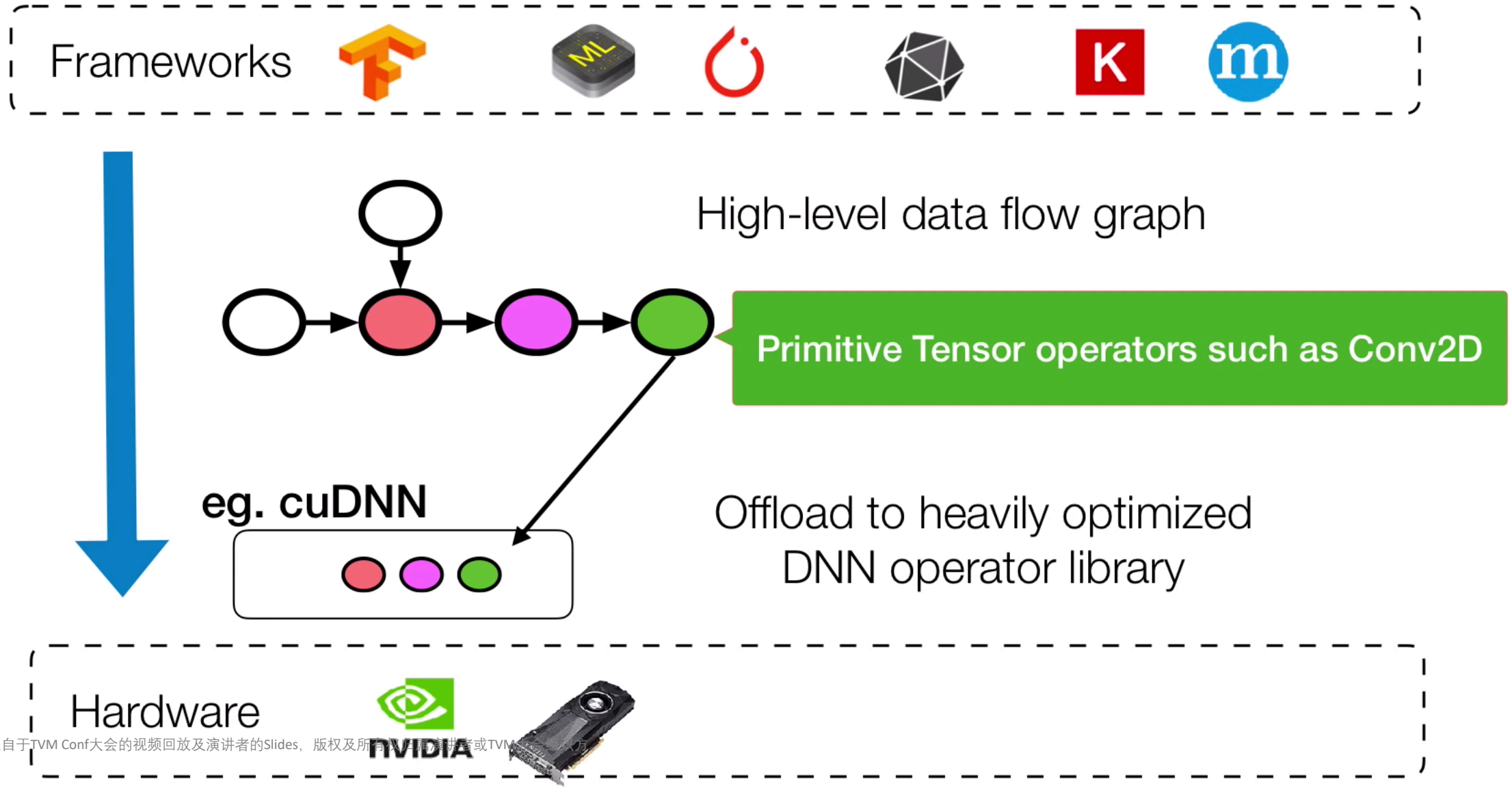
- 首先要知道
  - 深度学习、机器学习
  - 神经网络的部署（推理 or 应用 or inference）
- 没有TVM的时候，部署的问题是什么？
- TVM一开始希望解决谁的问题？



TVM一开始希望解决谁的问题？

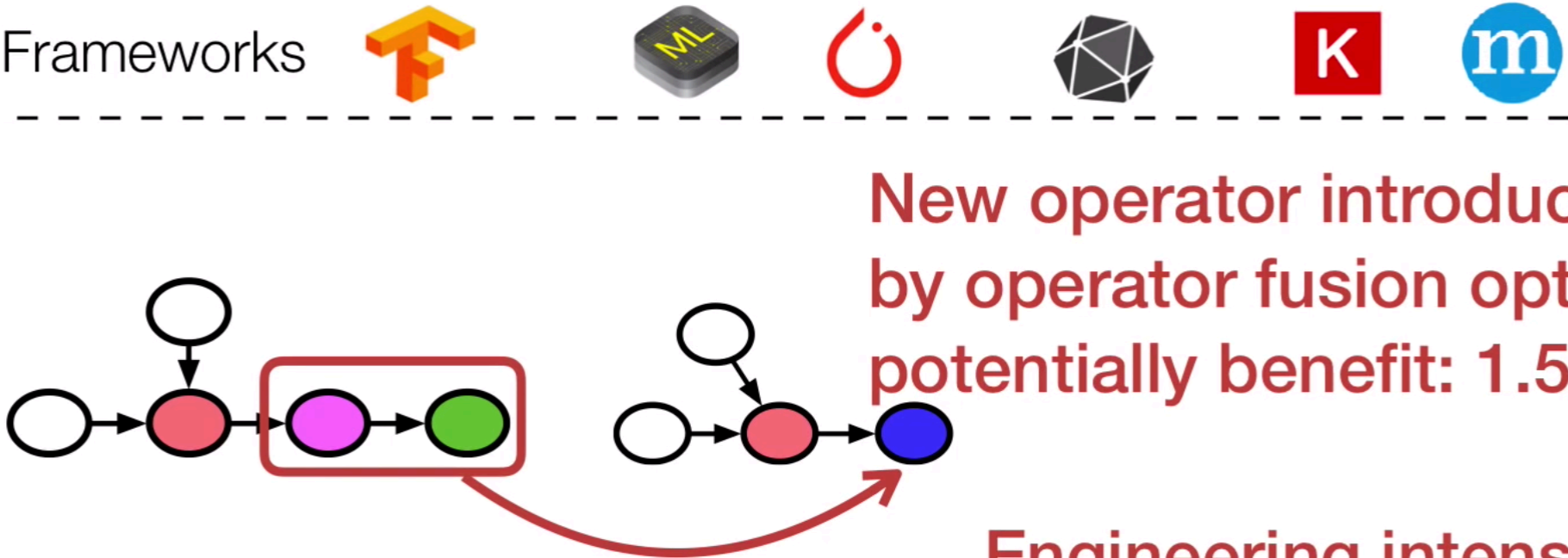
Credit：截图来自于TVM Conf大会的视频回放及演讲者的Slides，版权及所有权归属演讲者或TVM大会组织方。

Existing Approach



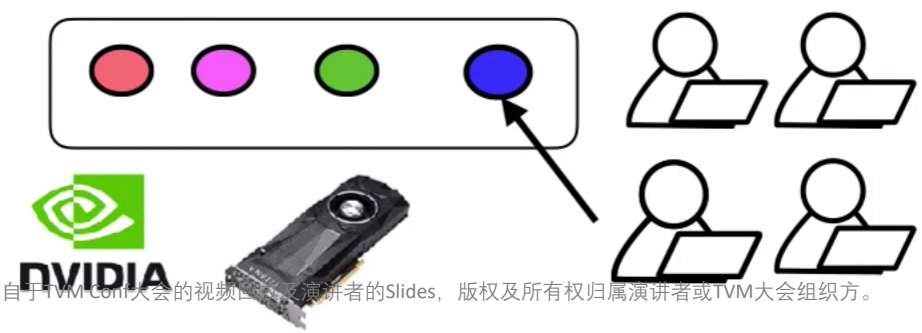
Limitations of Existing Approach

Frameworks

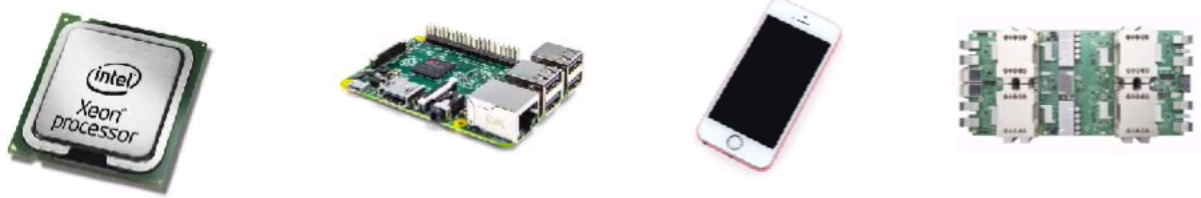


New operator introduced
by operator fusion optimization
potentially benefit: 1.5x speedup

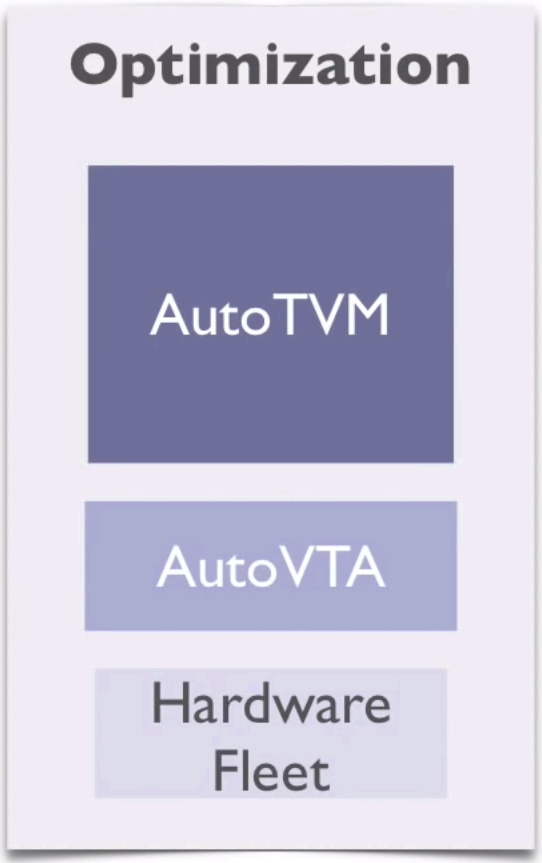
cuDNN



Engineering intensive



Automated by Machine Learning



tvm Automated by Machine Learning

说到底，TVM长什么样？

- (转入 https://tvm.ai) 跟 Tensorflow 是不是有点像？
- TOPI 是什么东西？
- Halide、Halide IR 是什么东西？
- Relay 是什么东西？

几个概念，就差不多不慌了

- 这是编译器行为，类似于构建CFG（也有叫 compute graph）
- 调度类似于调整了CFG
- 优化主要是【存储访问】和【计算部件装填】
- Autotuning（及其ML形式）都可以归为搜索问题（非连续）

Relay

- **Relay** is the high level **IR** of the **TVM** stack.
- Generalize computation graphs to **differentiable** programs.
- Enables whole-program optimization for deep learning.
- Composed of new **IR**, **auto-diff**, **optimizer**, and **backends.**
- **Relay** is open source.

Relay: A New IR for Machine Learning Frameworks

Jared Roesch jroesch@cs.uw.edu
Steven Lyubomirsky sslyu@cs.uw.edu
Logan Weber weberlo@cs.uw.edu
Josh Pollock joshpoll@cs.uw.edu
Marisa Kirisame jerry96@cs.uw.edu
Tianqi Chen tqchen@cs.uw.edu
Zachary Tatlock ztatlock@cs.uw.edu

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle, WA, USA

Abstract

Machine learning powers diverse services in industry including search, translation, recommendation systems, and security. The scale and importance of these models require that they be efficient, expressive, and portable across an array of heterogeneous hardware devices. These constraints are often at odds; in order to better accommodate them we propose a new high-level intermediate representation (IR) called Relay. Relay is being designed as a purely-functional, statically-typed language with the goal of balancing efficient compilation, expressiveness, and portability. We discuss the goals of Relay and highlight its important design constraints. Our prototype is part of the open source NNVM compiler framework, which powers Amazon's deep learning framework MxNet.

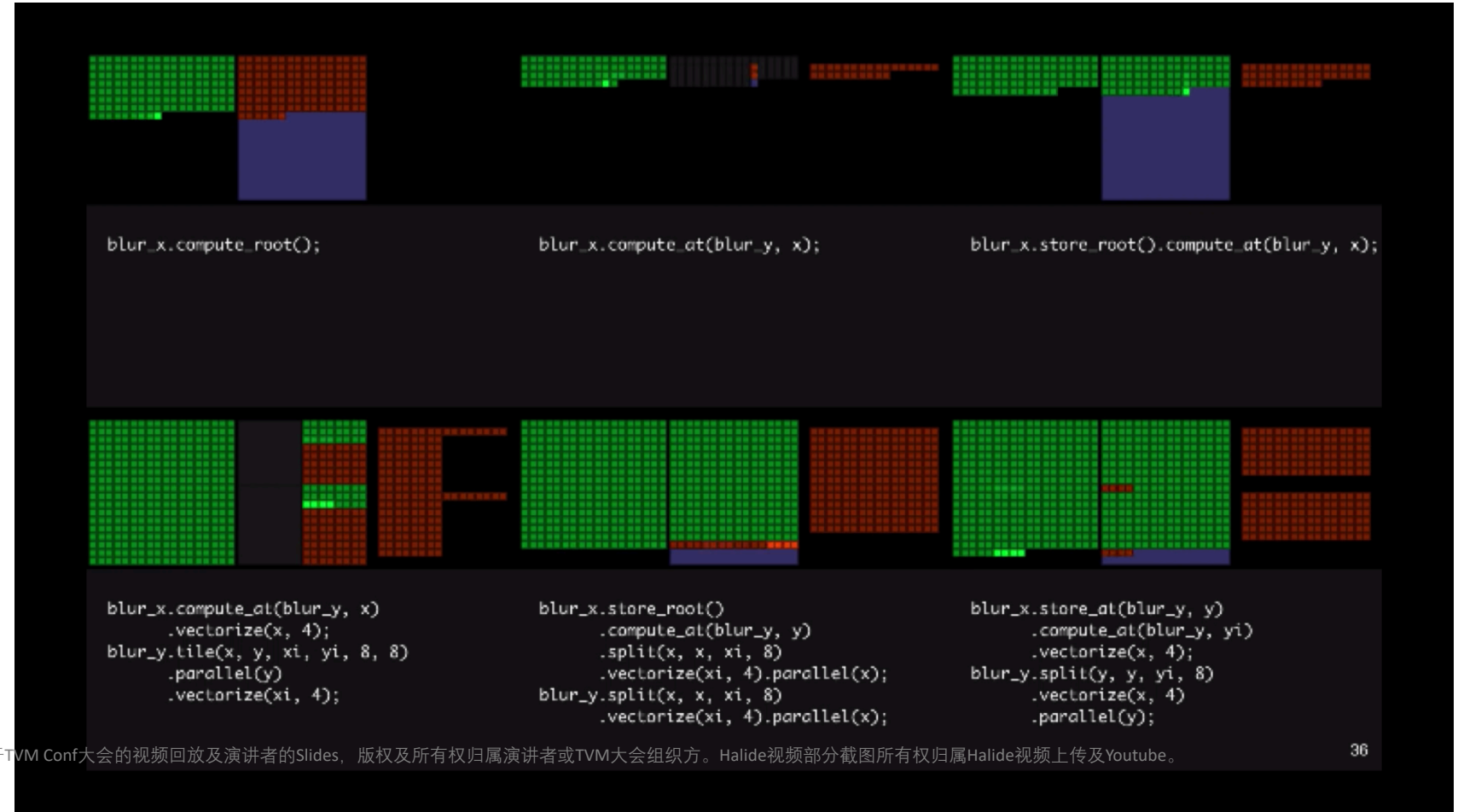
1 Introduction

Machine learning (ML) has dramatically reshaped computer vision [17, 21], natural language processing, robotics [14], and computational biology and is continuing to gain traction in new areas, including program synthesis [6]. Deep learning (DL) in particular has driven progress in these areas and is now powering diverse services in industry, including search, translation, recommendation systems, and security. Hardware diversity is growing almost as quickly. DL models are deployed not only in the cloud, but also on a myriad of devices, from off-the-shelf CPUs and GPUs to specialized smartphone chips and IOT edge devices.

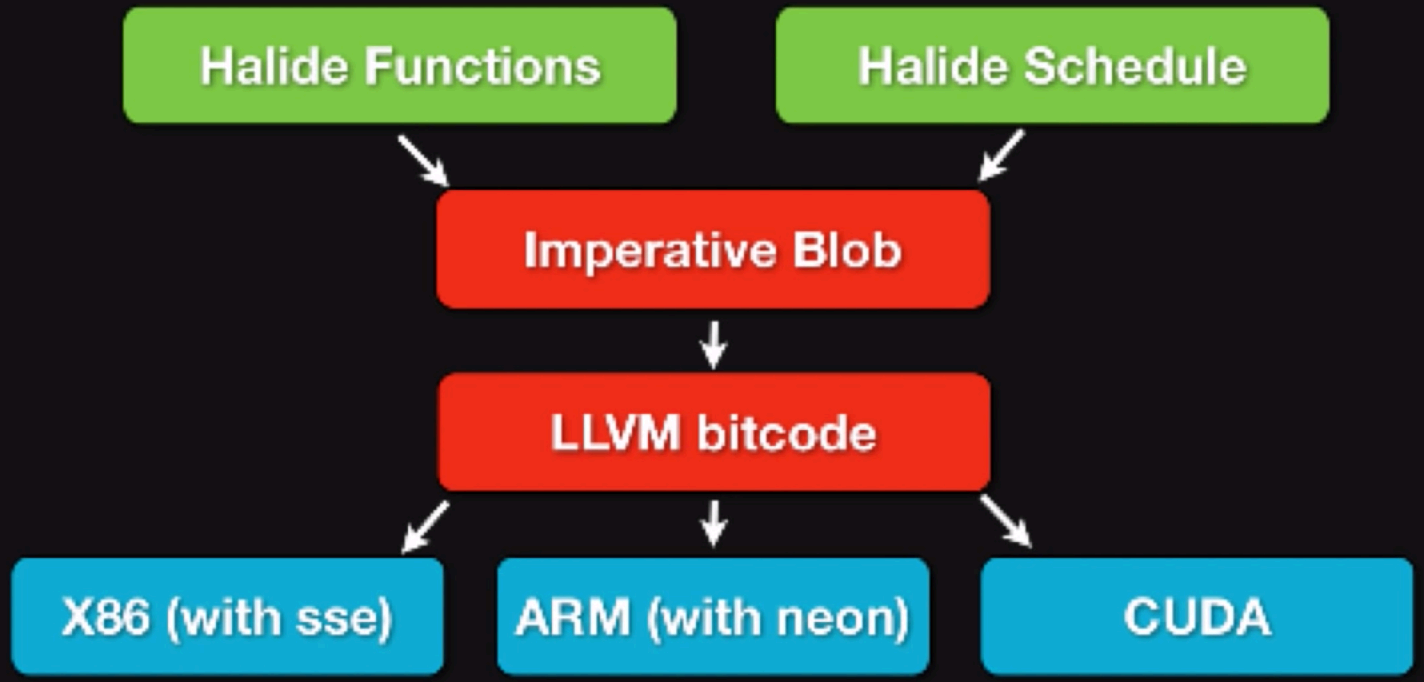
The rise of GPUs for DL compute has made deep learning both possible and scalable for many tasks, but a new generation of applications with greater compute demands

TVM的一些概念和代码借用了Halide

- （转入Halide的介绍视频）



The Halide Compiler



Local Laplacian Filters

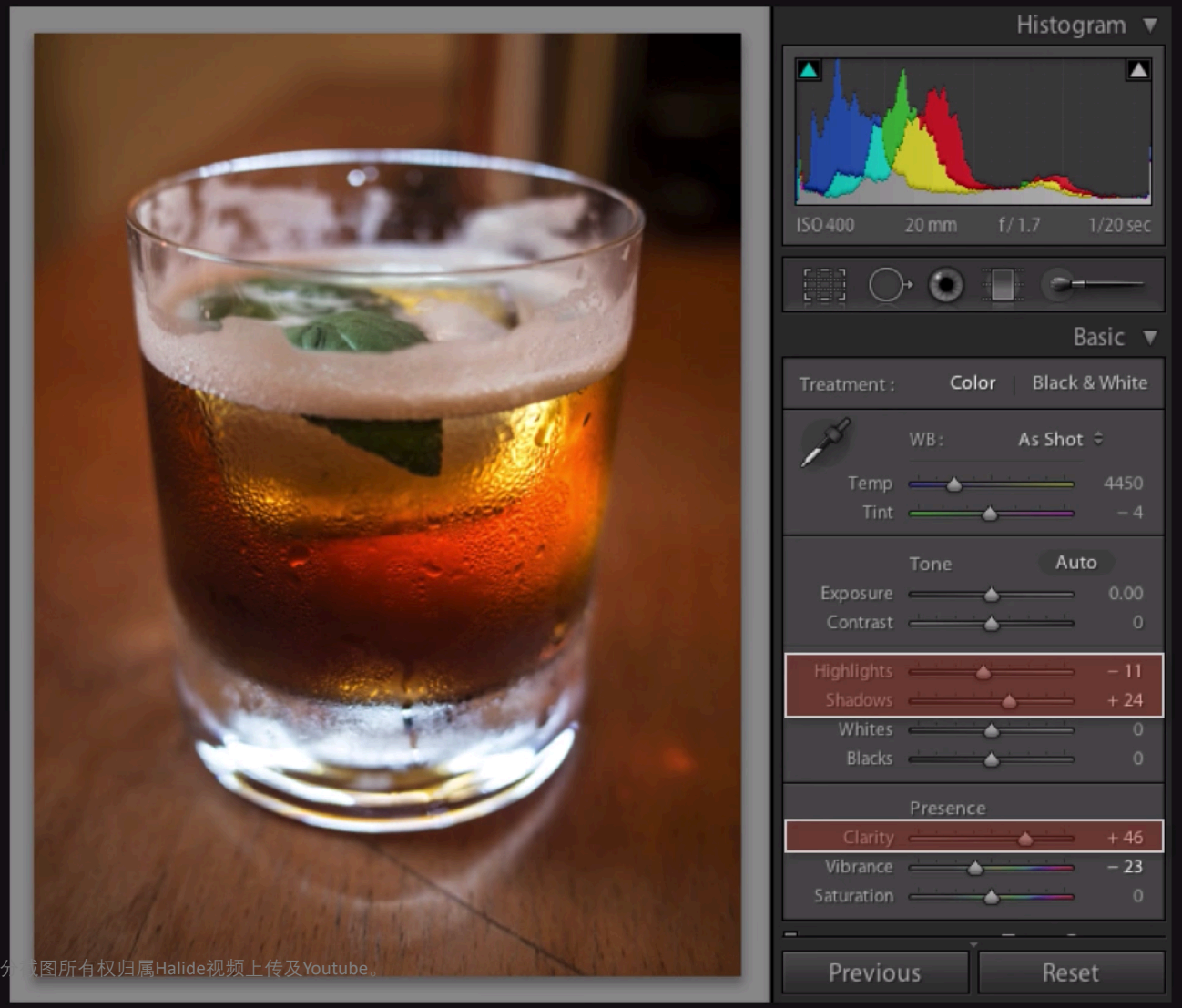
prototype for Adobe Photoshop Camera Raw / Lightroom

Reference: 300 lines C++

Adobe: 1500 lines
3 months of work
10x faster *(vs. reference)*

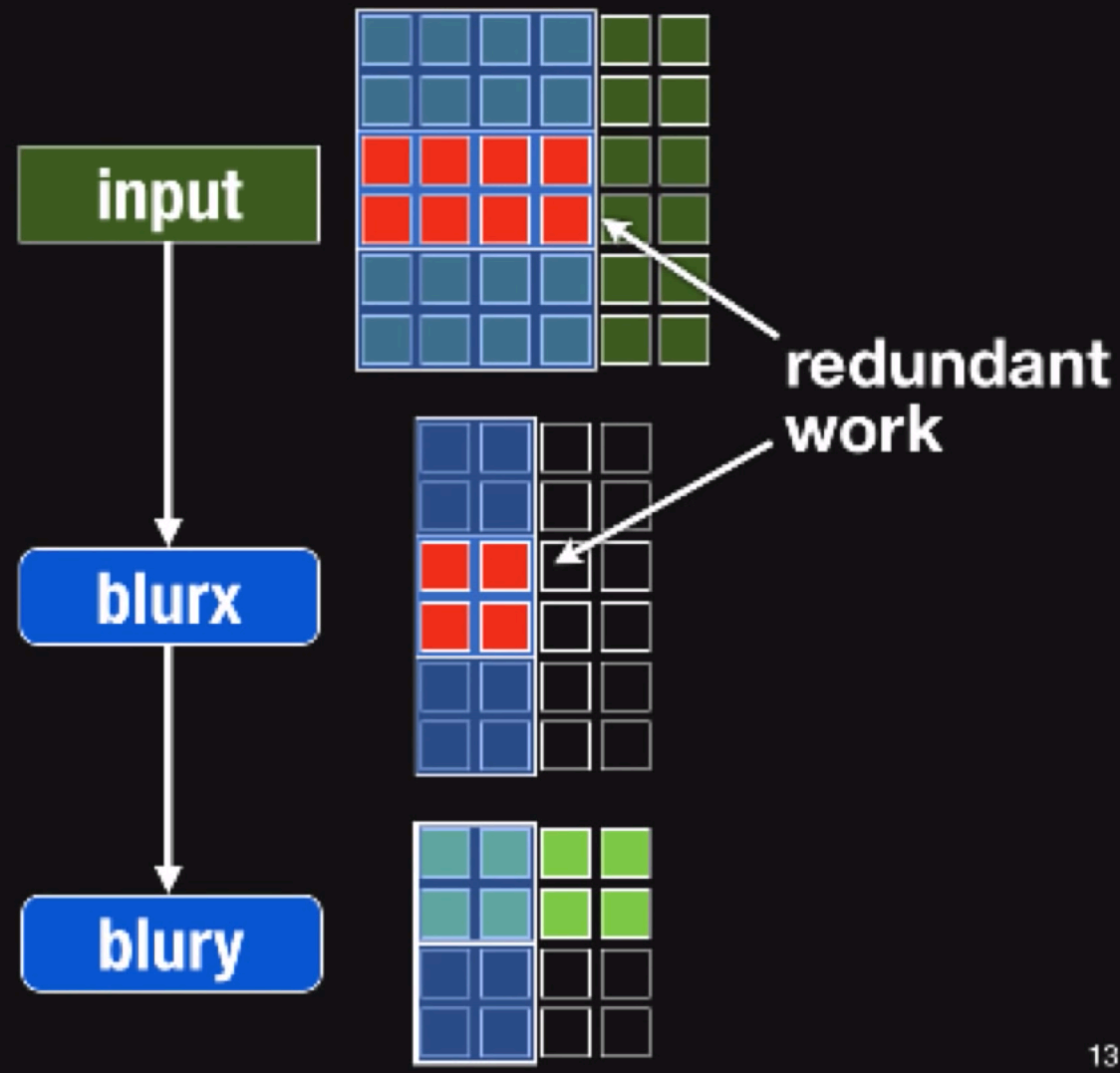
Halide: 60 lines
1 intern-day

20x faster (vs. reference)
2x faster (vs. Adobe)



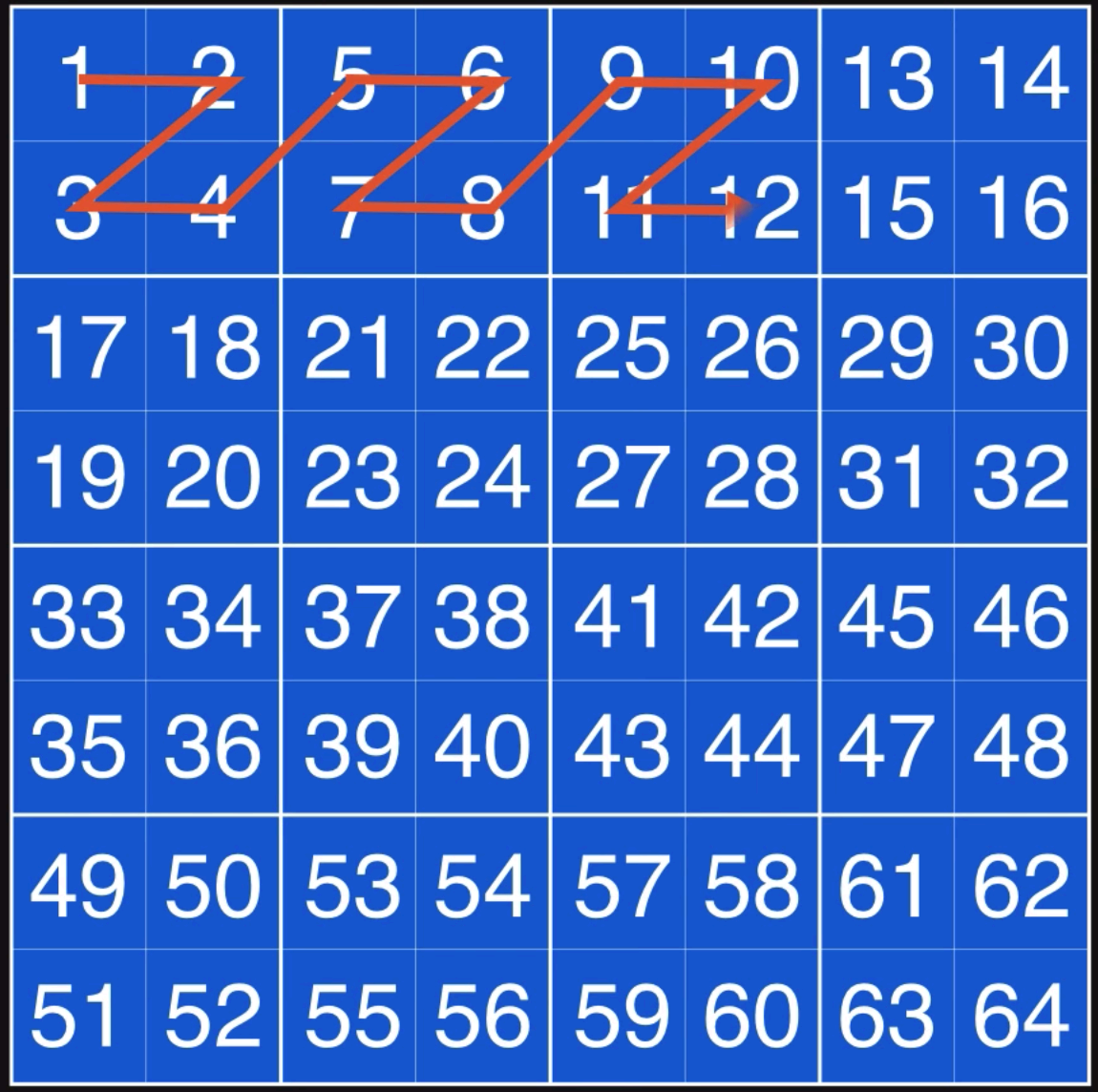
Fusion is a complex tradeoff

```
void box_filter_3x3(const Image &in, Image &blury) {
  __m128i one_third = _mm_set1_epi16(21846);
 #pragma omp parallel for
 for (int yTile = 0; yTile < in.height(); yTile += 32) {
   __m128i a, b, c, sum, avg;
   __m128i blurx[(256/8)*(32+2)]; // allocate tile blurx array
   for (int xTile = 0; xTile < in.width(); xTile += 256) {
     __m128i *blurxPtr = blurx;
     for (int y = -1; y < 32+1; y++) {
       const uint16_t *inPtr = &(in[yTile+y][xTile]);
       for (int x = 0; x < 256; x += 8) {
         a = _mm_loadu_si128((__m128i*)(inPtr-1));
         b = _mm_loadu_si128((__m128i*)(inPtr+1));
         c = _mm_load_si128((__m128i*)(inPtr));
         sum = _mm_add_epi16(_mm_add_epi16(a, b), c);
         avg = _mm_mulhi_epi16(sum, one_third);
         _mm_store_si128(blurxPtr++, avg);
         inPtr += 8;
       }
     }
     blurxPtr = blurx;
     for (int y = 0; y < 32; y++) {
       __m128i *outPtr = (__m128i *)(&(blury[yTile+y][xTile]));
       for (int x = 0; x < 256; x += 8) {
         a = _mm_load_si128(blurxPtr+(2*256)/8);
         b = _mm_load_si128(blurxPtr+256/8);
         c = _mm_load_si128(blurxPtr++);
         sum = _mm_add_epi16(_mm_add_epi16(a, b), c);
         avg = _mm_mulhi_epi16(sum, one_third);
         _mm_store_si128(outPtr++, avg);
}}}
```



The schedule defines order & parallelism within stages

Split x by 2,
Split y by 2.
Serial y_{outer},
Serial x_{outer},
Serial y_{inner},
Serial x_{inner}



Halide: decouple algorithm from schedule

Algorithm: *what* is computed
Schedule: *where* and ***when*** it's computed

Easy for programmers to build pipelines
simplifies algorithm code
improves modularity

Easy for programmers to specify & explore optimizations
fusion, tiling, parallelism, vectorization
can't break the algorithm

Easy for the compiler to generate fast code

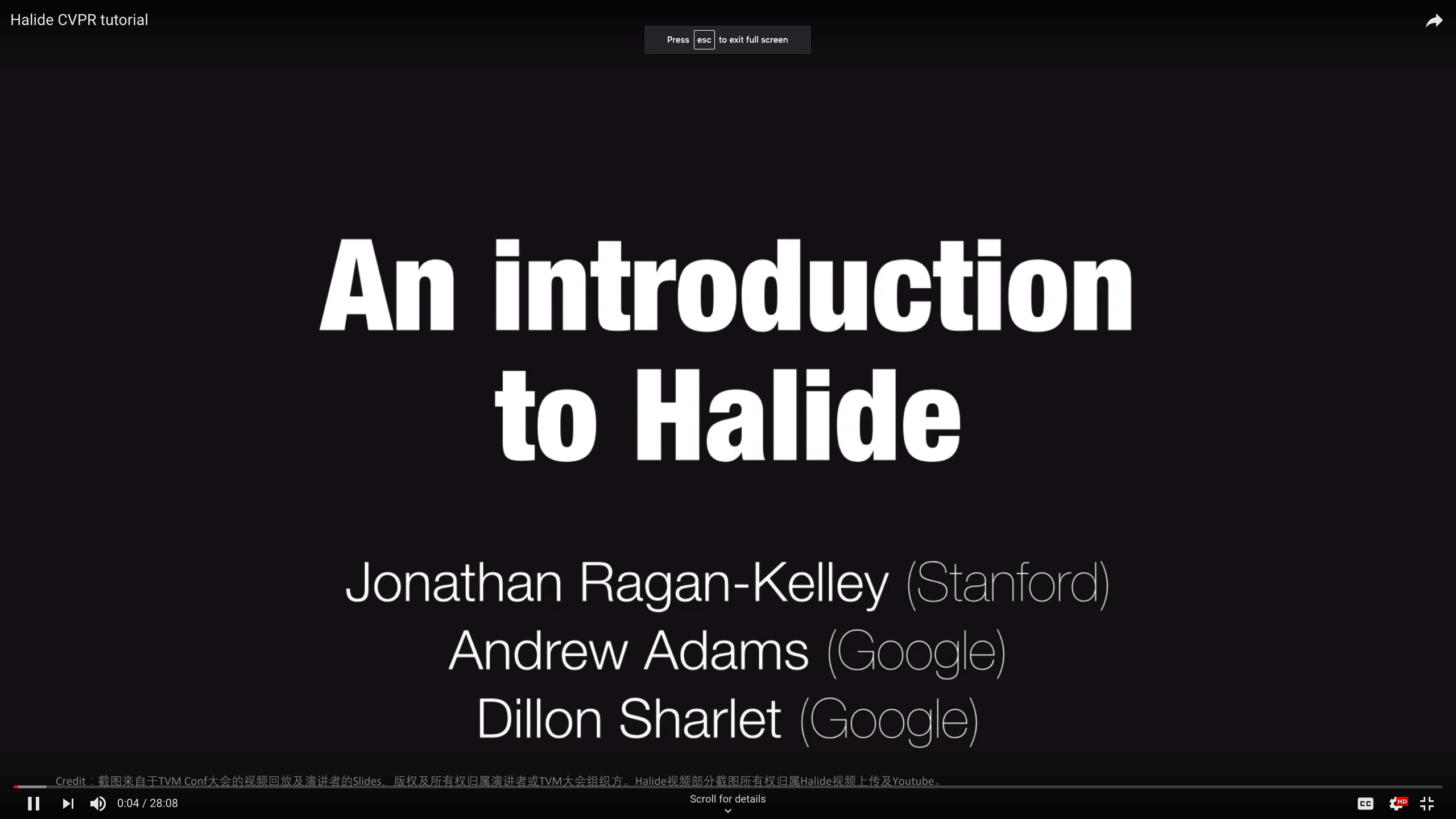
The algorithm defines pipelines as pure functions

Pipeline stages are *functions* from coordinates to values

Execution order and storage are unspecified

3x3 blur as a Halide *algorithm*:

```
Var x, y; Func blurx, blury;
blurx(x, y) = (in(x-1, y) + in(x, y) + in(x+1, y))/3;
blury(x, y) = (blurx(x, y-1) + blurx(x, y) + blurx(x, y+1))/3;
```



1st TVM and Deep Learning Compilation Conference

Opening Keynote Session

December 12, 2018

tvm

Time	
9:00	Keynote – SAMPL, Apple, Amazon, Huawei. [Video] [Slides]
10:15	TVM Stack Overview – Tianqi Chen, UW. [Video] [Slides]
10:45	Deep Learning Compilation at Amazon – Yida Wang, Amazon. [Video] [Slides]
11:05	**break**
11:25	AutoTVM & Device Fleet – Eddie Yan, UW. [Video] [Slides]
11:45	VTA Open & Flexible Deep Learning Accelerator – Thierry Moreau, UW. [Video] [Slides]
12:05	Fast & Faster Privacy-Preserving ML in Secure Hardware Enclaves – Nick Hynes, UC Berkeley/Oasis Labs. [Video] [Slides]
12:20	**Lunch (boxed lunches will be provided)**
13:30	Spatial: A Language and Compiler for Application Accelerators – Kunle Olukotun/Raghu Prabhakar, Stanford & SambaNova. [Video] [Slides]
13:50	Machine Programming – Justin Gottschlich, Intel. [Video] [Slides]
14:10	PlaidML Stripe: Polyhedral IR + Model-guided Optimization – Brian Retford, Intel. [Video] [Slides]
14:25	Relay: a high level differentiable IR – Jared Roesch, UW. [Video] [Slides]
14:45	Scalable Distributed Training with Parameter Hub: A Whirlwind Tour – Liang Luo, UW. [Video] [Slides]

15:05	The HammerBlade: An ML-Optimized Supercomputer for ML and Graphs – Michael Taylor, UW. [Video] [Slides]
15:20	**break, contributors meetup**
15:50	TVM @ FB – Andrew Tulloch, Facebook. [Video] [Slides]
16:10	Inference Architectures @ Xilinx – Graham Schelle, Xilinx. [Video] [Slides]
16:30	**Lightning talks session**
	Efficient Voice Activity Detection via Binarized Neural Networks – Matthai Philipose, Microsoft. [Video] [Slides]
	Heterogenous Bitwidth Binarization: Weird Operators with Big Benefits – Josh Fromm, UW. [Video] [Slides]
	Generating Fast Operators for Binarizable Networks – Meghan Cowan, UW. [Video] [Slides]
	OpenCL Backend for FPGA – Morita Kazutaka, NTT, Japan. [Video] [Slides]
	Build Your Own VTA Design with Chisel – Luis Vega, UW. [Video] [Slides]
	µTVM: Deep Learning on Bare-Metal Devices – Pratyush Patel, UW. [Video] [Slides]
	Supporting TVM on RISC-V Architectures – Jenq-Kuen Lee, NTHU, Taiwan. [Video] [Slides]
	Bring Your Own Datatypes – Gus Smith, UW. [Video] [Slides]
	AutoScheduler for TVM – Lianmin Zheng, SJTU. [Video][Slides]

	Data Visualization with Vega-Lite and Altair – Dominik Moritz , UW. [Video][Slides]
	TVM on Hexagon DSP – Krzysztof Parzyszek , Qualcomm. [Video][Slides]
	Sharing, Protection, and Compatibility for Reconfigurable Fabric with AmorphOS – Ahmed Khawaja, UT Austin. [Video] [Slides]
17:35	TVM & the Apache Software Foundation – Markus Weimer, Microsoft and Apache Software Foundation. [Slides]
18:15 to 20:00	*Social (drinks, food)*

Follow us on Twitter

Follow @ApacheTVM

Tweets by ApacheTVM

Register

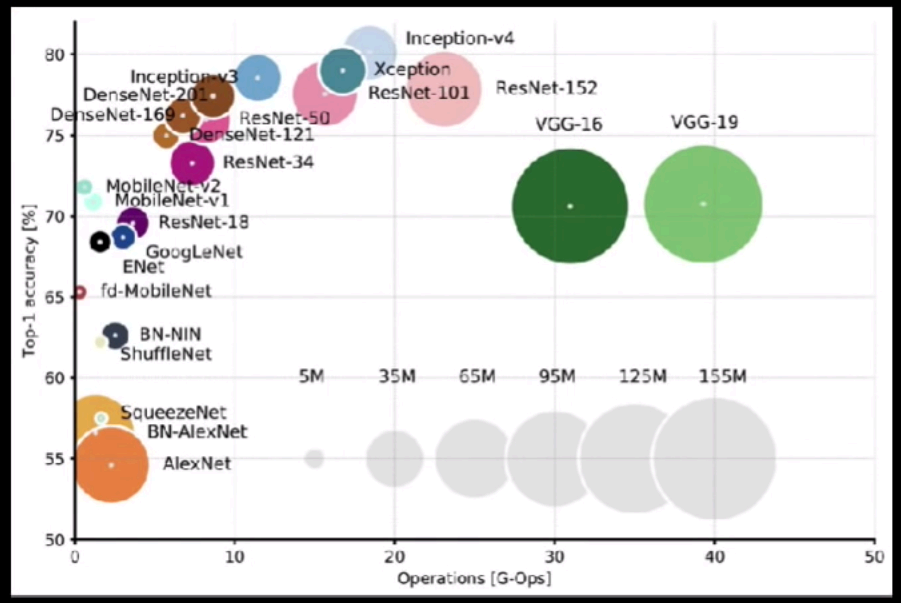
Note that if you pre-registered, you do not need to register again.

Click to Register

Software era: Problem to solve → Write code → Run on fast machine

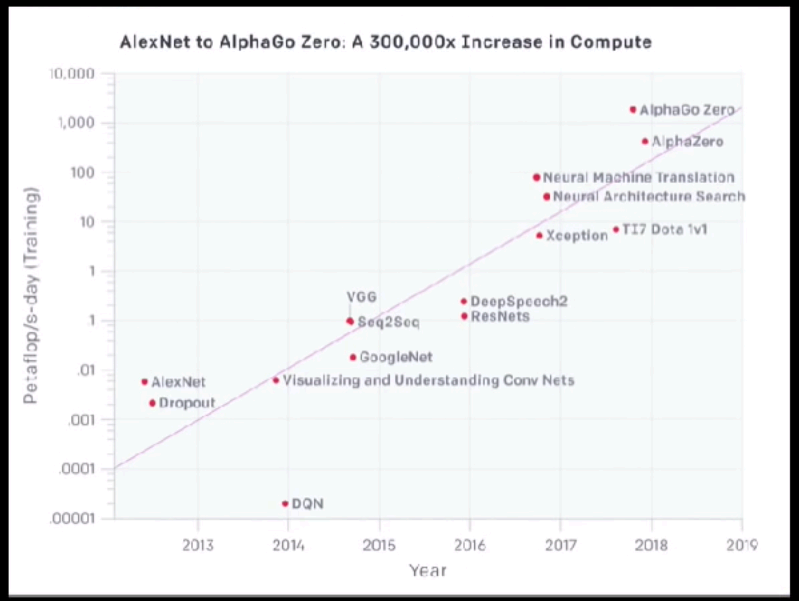
Machine learning era: Problem to solve → Data + model templates → Train on *fastest* machine → Inference on fast & cheap machine

Model size and compute cost growing fast

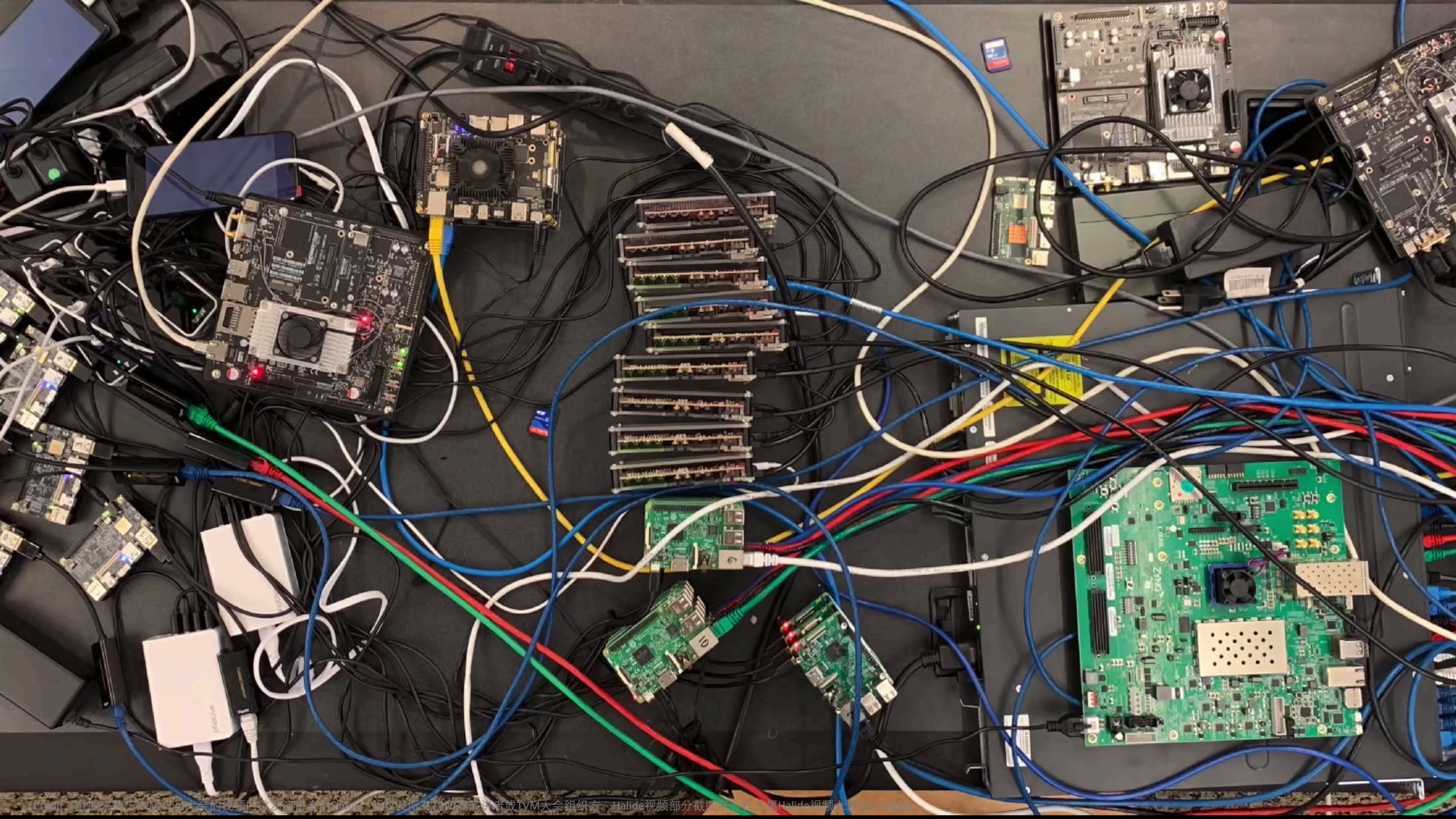


by Eugenio Culurciello

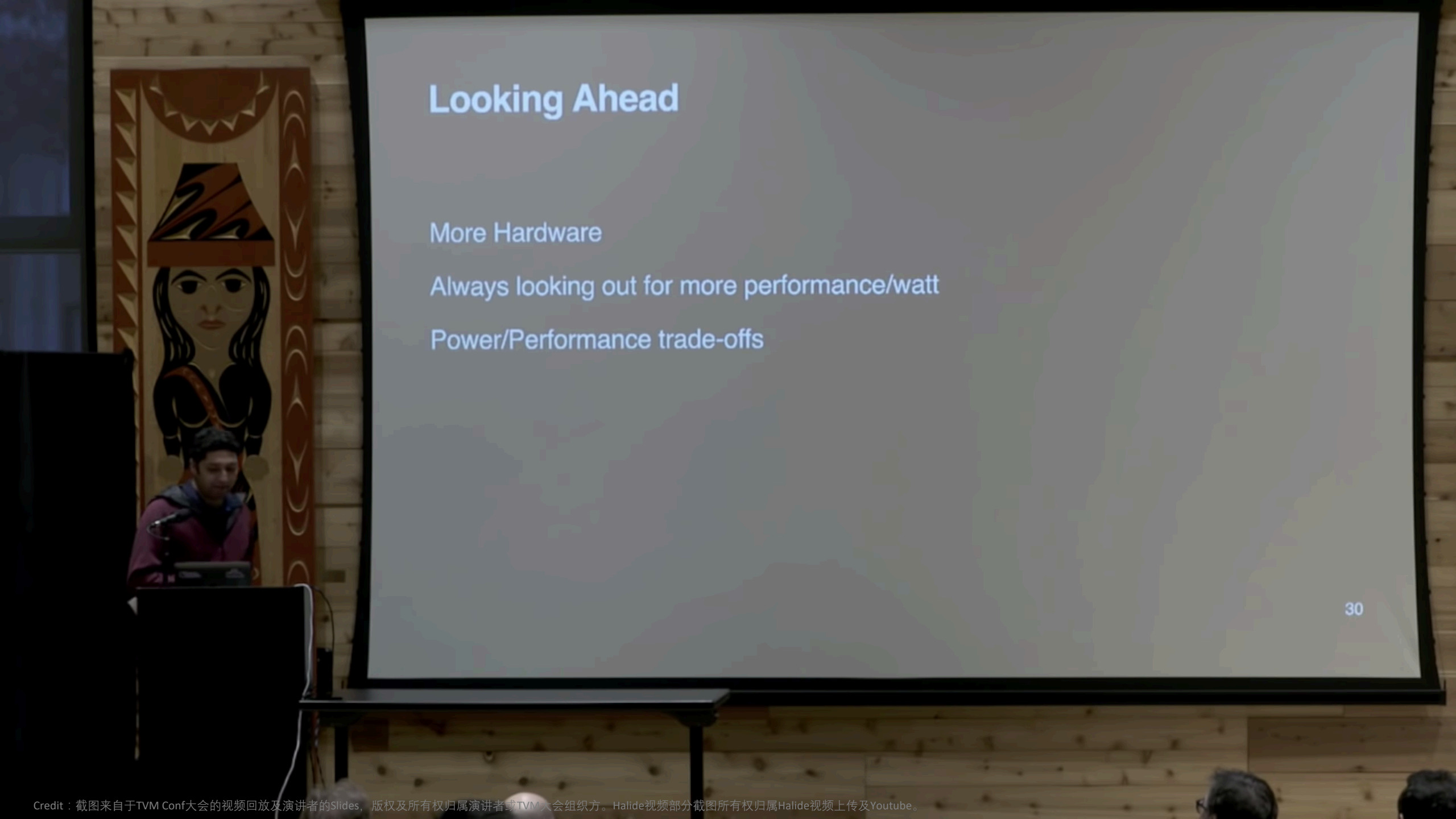
Training costs growing exponentially



by Open AI



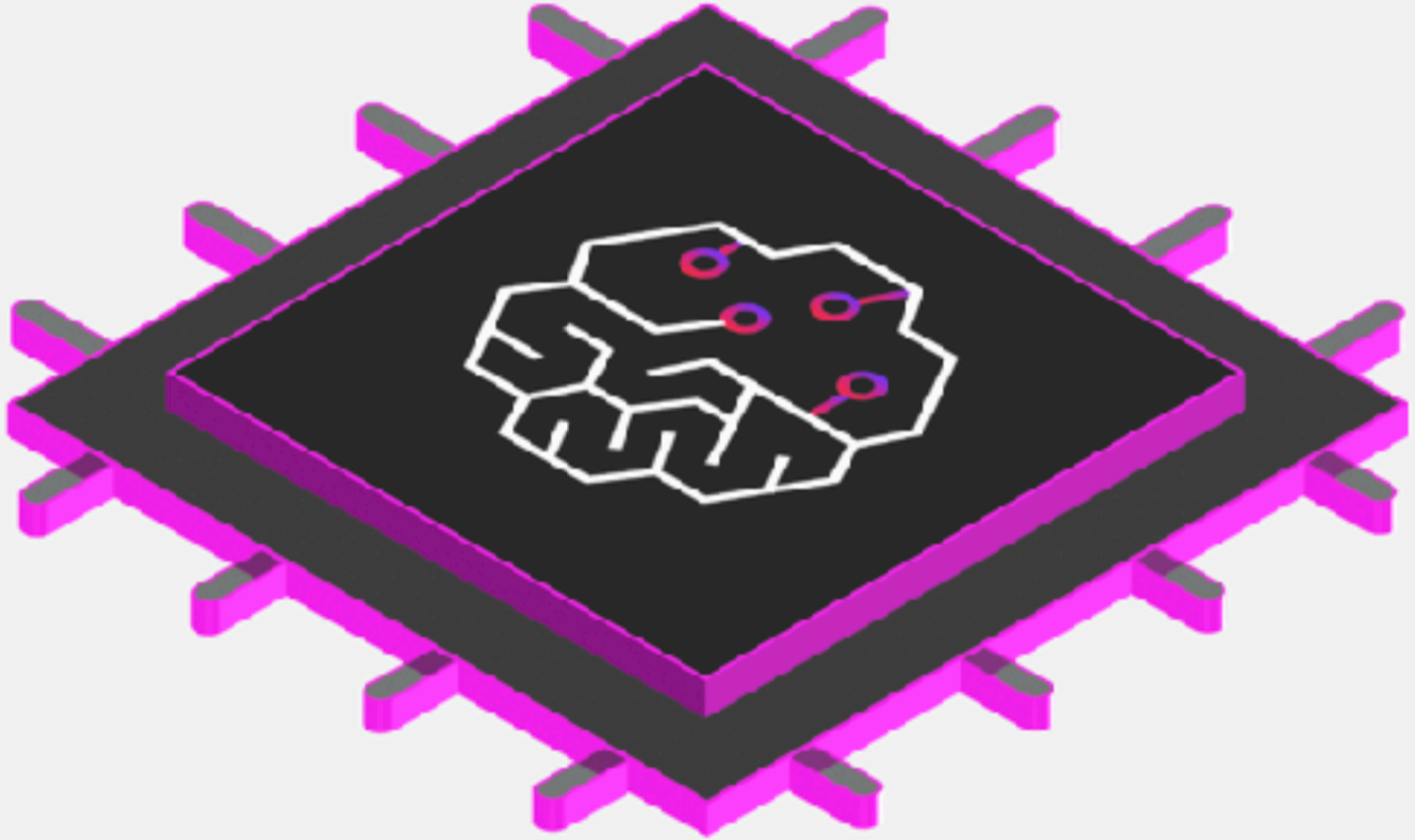
[Credit] 截图来自于TVM Conf大会的视频直播及演讲者的Slides，版权及解释权归演讲者或TVM大会组织方。Halide视频部分截图所有权归属Halide视频上传者及Youtube。



How is AWS using TVM?

- As a back-end for Apache MXNet
  - To deploy easily onto edge devices
  - To improve performance on target hardware
- As an optimizer for Amazon AI services
  - Amazon Rekognition: To improve end-to-end latency
  - Amazon Alexa: To increase resource efficiency on Echo/Dot

ASICs – AWS inferentia



Open source compilers have transformed our industry

In the age of domain-specialized systems…

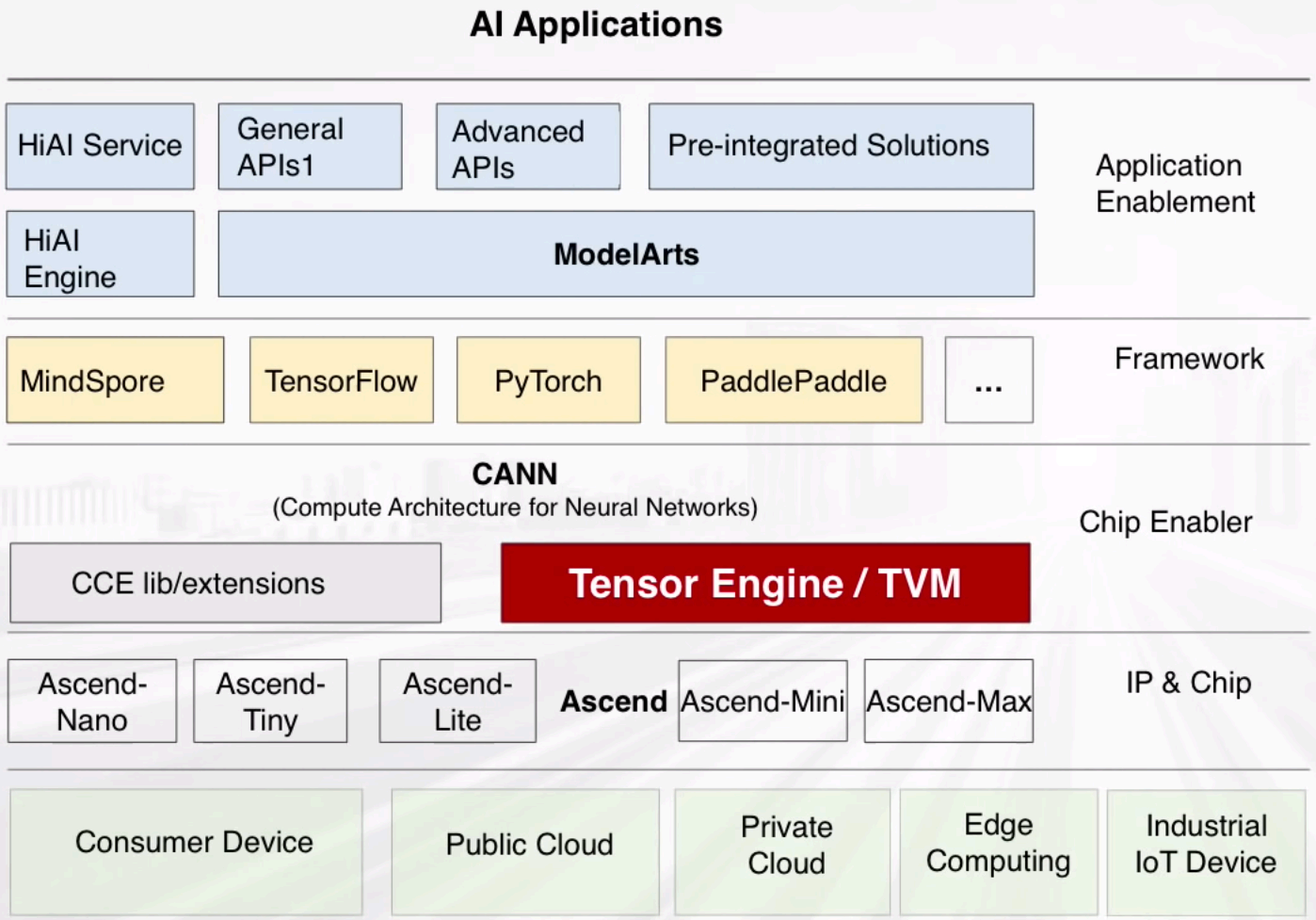
Specialized compiler stack
for Deep Learning

tvm

First major open source
compiler collection

LLVM: Higher-Level IR, new
optimizations, easier extensibility

TVM on Huawei's AI portfolio



Application enabling:
Full-pipeline services(ModelArts), hierarchical APIs, and pre-integrated solutions

MindSpore:
Unified training and inference framework for device, edge, and cloud (both standalone and cooperative)

CANN:
Chip operators library and highly automated operators development toolkit

Ascend:
AI chip series based on unified scalable architecture

HUAWEI

Huawei's Contributions on TVM

8 Contributors:

kun-zh, sgrechanik-h, libing4752, derisavi-huawei, solin319, ehsanmok, gaoxiong-1, jiacunjiang1215

4 Reviewers:

Srkreddy1238 , PariksheetPinjari909 , siju-Samuel , Xqdan

We are working on:

1.Huawei Ascend ASIC support.
2.Front end to support Darknet, ONNX.
3.Optimization on Auto-TVM, IR extensions.
4.Tensorize, cache read/write, access_ptr API.

In the future we will try to:

1.Codegen for fused operators.
2.NLP support.
3.More optimization.
4.Training Operators.

TVM on Hexagon™ DSP

- Background, goals, and motivations:
  - Qualcomm® Hexagon DSP with Hexagon Vector eXtensions (HVX).
  - Compiler development: LLVM + Hexagon.
  - Ease of use and accessibility of Hexagon hardware features.
  - Reuse of technology: TVM = ML + LLVM.
- TVM on Hexagon now:
  - Hexagon as device and compilation target.
  - Execute computational kernels on simulator.
- TVM on Hexagon future:
  - Quantized data.
  - Offload subgraphs.
  - Hexagon support upstream.
  - Everything runs flawlessly using minimum resources.

Auto-scheduler for TVM

Lianmin Zheng

Shanghai Jiao Tong University

Credit：截图来自于TVM Conf大会的视频回放及演讲者的Slides，版权及所有权归属演讲者或TVM大会组织方。Halide视频部分截图所有权归属Halide视频上传及Youtube。

对哪些地方感兴趣？直接上手吧！