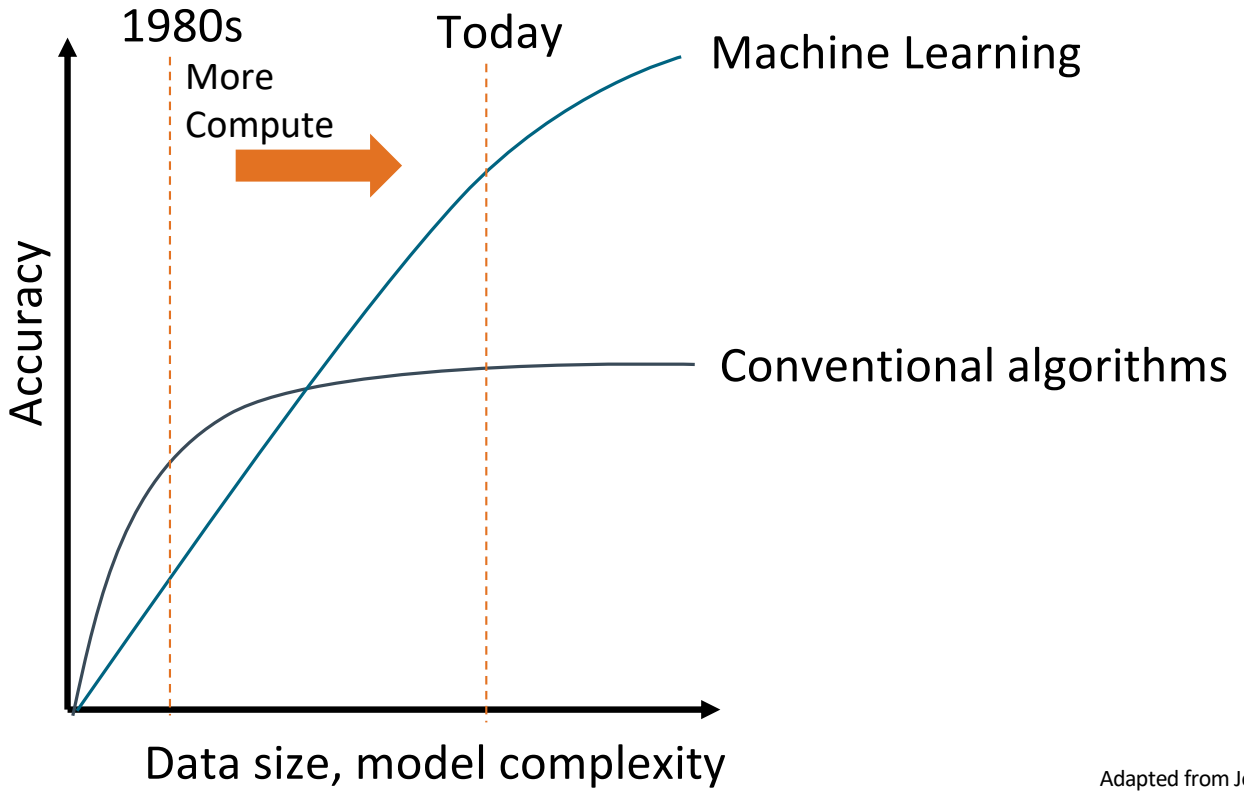
Accelerating Software 2.0

Yaqi Zhang

Principal Engineer

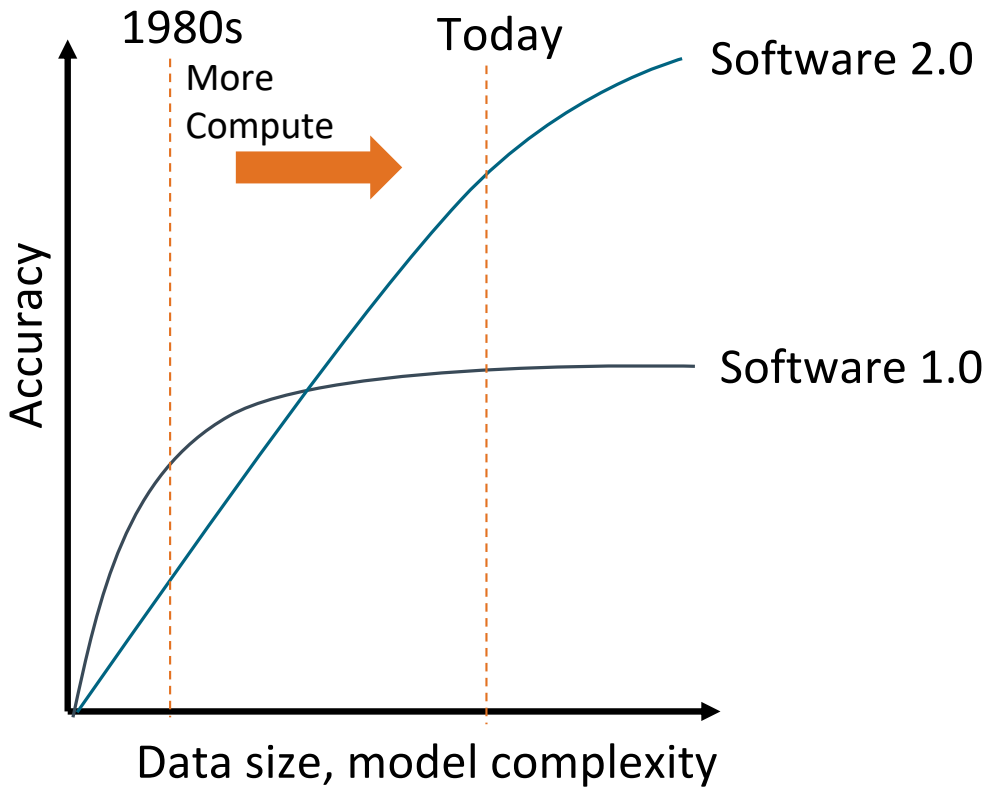
SambaNova SYSTEMS

Machine Learning Today



Adapted from Jeff Dean
HotChips 2017

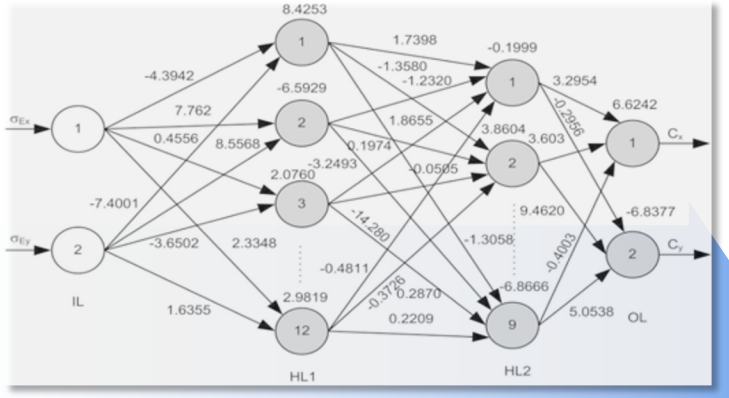
Machine Learning Today



Adapted from Jeff Dean
HotChips 2017

Evolving Nature of Computational Models

```
template<typename InputIterator1, typename InputIterator2, typename NumericT>
NumericT  inner_product(InputIterator1 start1,
                        InputIterator1 end1,
                        InputIterator2 start2,
                        NumericT       startval) const {
  for (; start1 != end1; ++start1, ++start2)
    startval += ( (*start1) * (*start2) );
  return startval;
}
// ..
vector_type vec1, vec2;
vector_type::numeric_t val;
val = inner_product(vec1.begin(), vec1.end(), vec2.begin(),
                    vector_type::numeric_t(0));
```



Software 1.0

- Programmer input: code (C++, etc.)
- Solution encoded in composed algorithms
- Deterministic computations
- Only has a single correct result

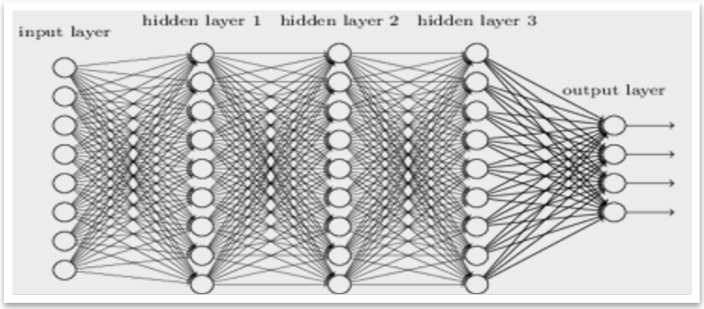
Software 2.0

- Programmer input: training data
- Solution encoded in trained weights
- Probabilistic models
- Results need only be statistically correct

Software 2.0 is Replacing Software 1.0

Easier to build and deploy

- Build products faster
- Predictable runtimes and memory use: easier qualification



2016: Google introduces GNMT (Google's Neural Machine Translation) changing its translation code from **500k imperative LoC (Software 1.0)** to **500 lines of TensorFlow (Software 2.0) while improving accuracy**

https://jack-clark.net/2017/10/09/import-ai-63-google-shrinks-language-translation-code-from-500000-to-500-lines-with-ai-only-25-of-surveyed-people-believe-automationbetter-jobs
https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

Challenge in Software 2.0

- Labeled Data
- Privacy
- AI Bias

...

- **Compute and Memory Complexity**

Training

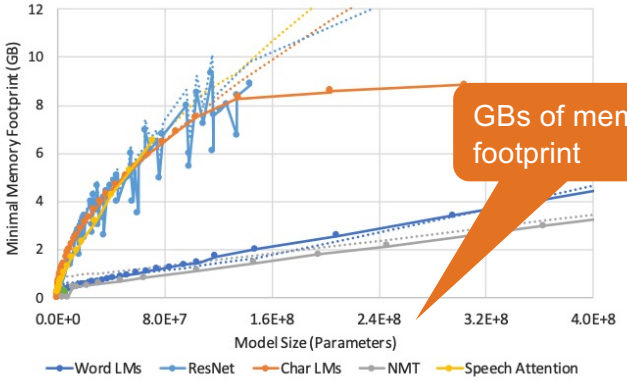
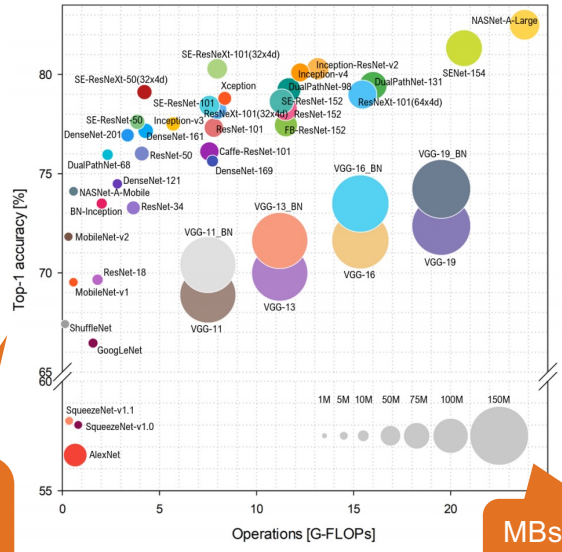


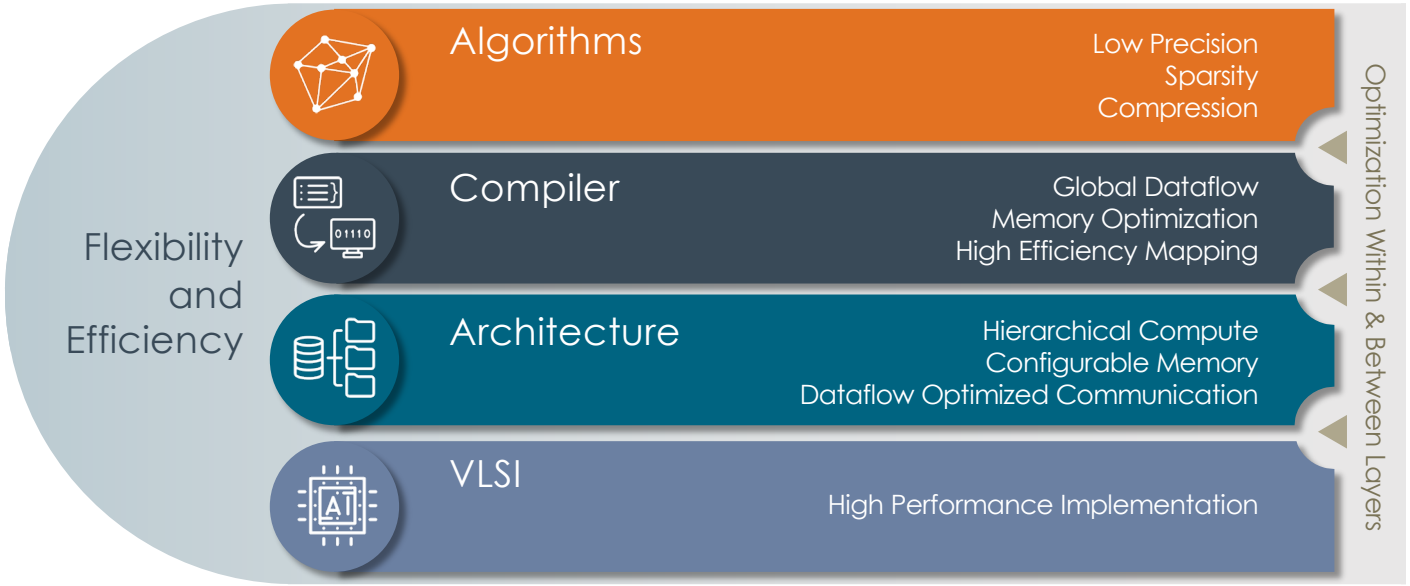
Figure 10. Empirical and asymptotic minimal memory footprint as model size grows (note: fixed subbatch size).

Inference



Our Solutions for Software 2.0

Full stack co-engineering yields optimizations where best delivered with the highest impact



Our Products

Full Stack Software + Hardware ML Solutions

- On-Prem DataScale System
  - 3D DataScale tour: https://sambanova.ai/products/
- Dataflow-as-a Service

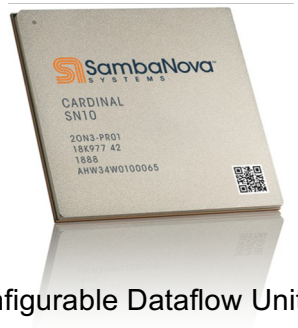
DataScale System

Open Standards, Disruptive Technology, Easy to Deploy

Open Standards Connectivity: Ethernet, PCI Express

Open Source Frameworks: PyTorch, TensorFlow

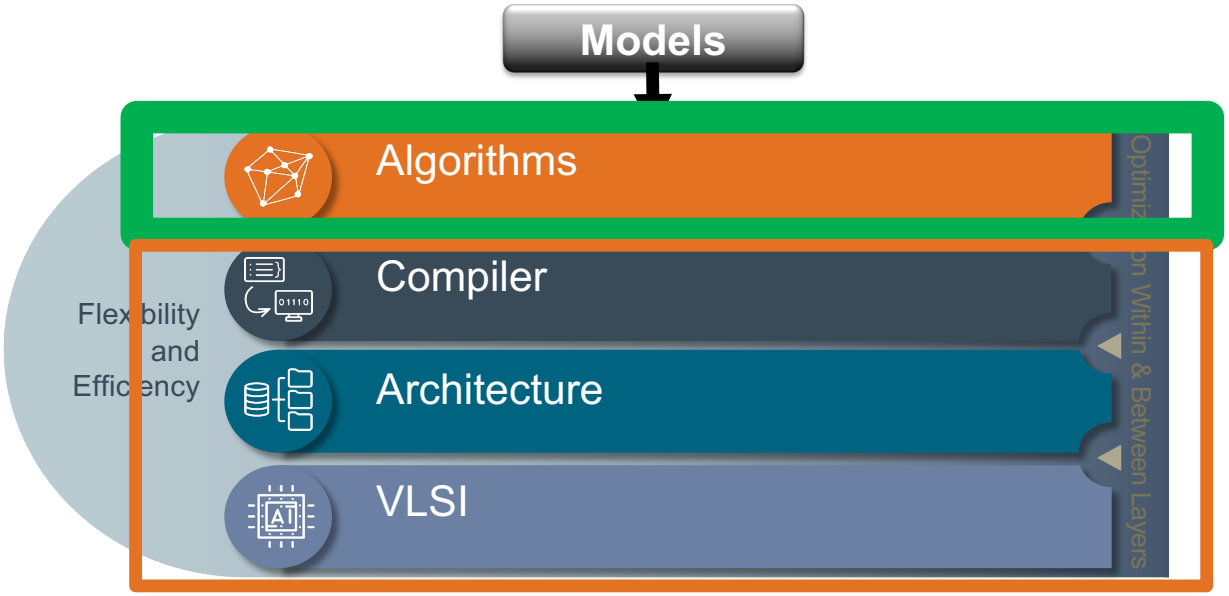
Open Source OS: Linux



Reconfigurable Dataflow Unit (RDU)

The SambaNova Systems Advantage

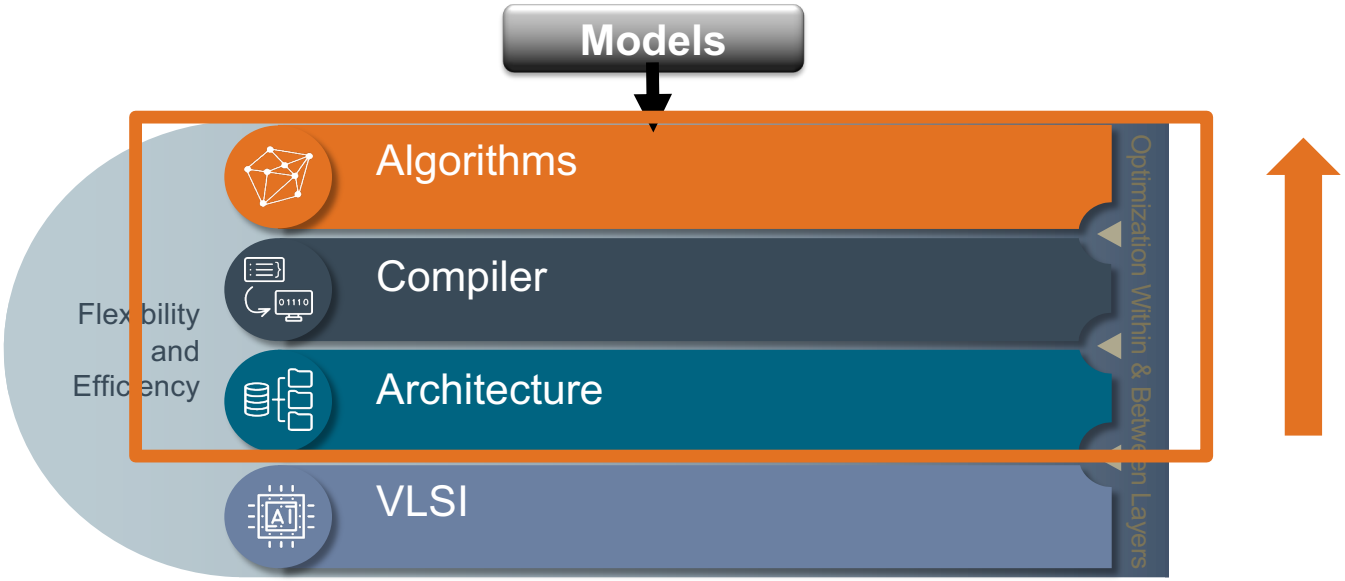
Achieve high model accuracy with fast execution



High model accuracy

Fast execution

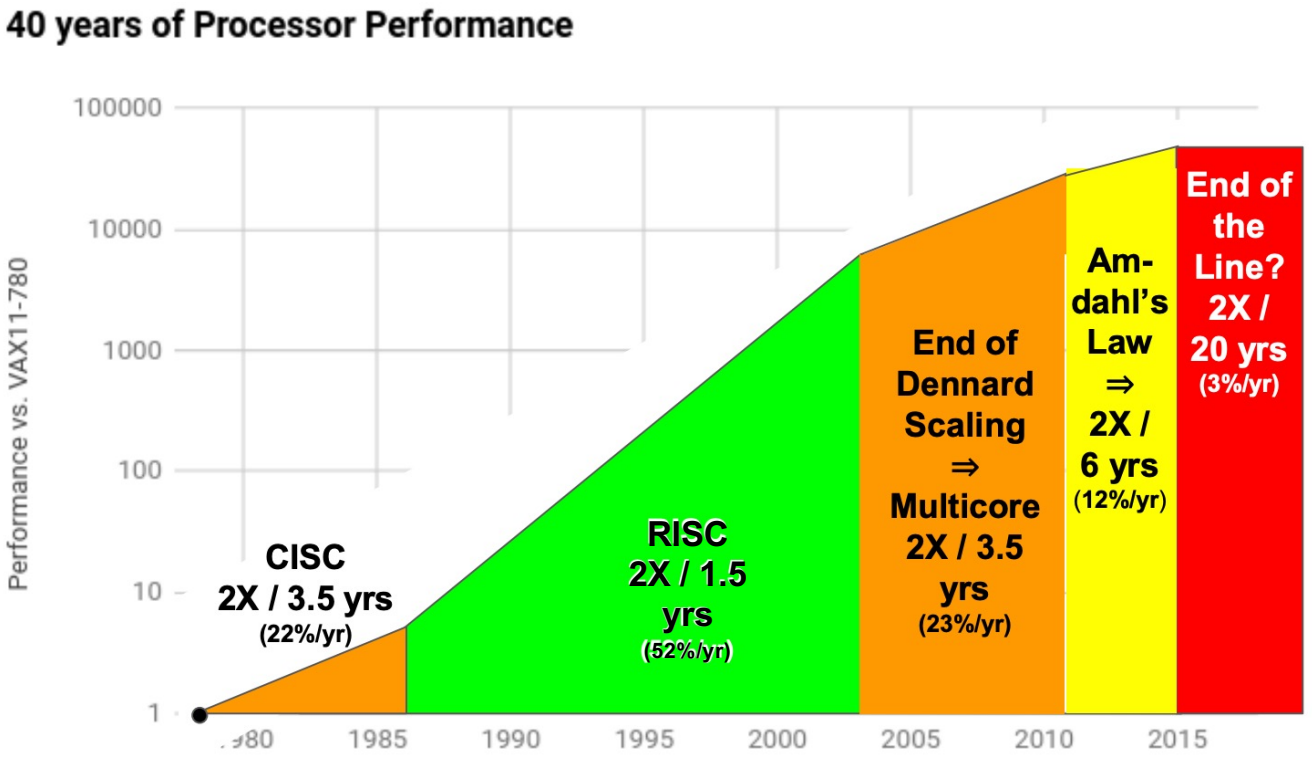
Outline



Architecture

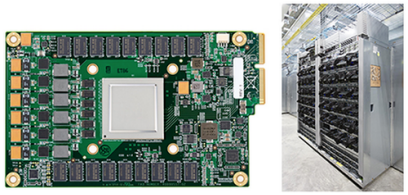
Efficiency vs. Flexibility

Slowdown of Performance Scaling



Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

How do we get the next 1000x speedup?



Tensor Processing Unit

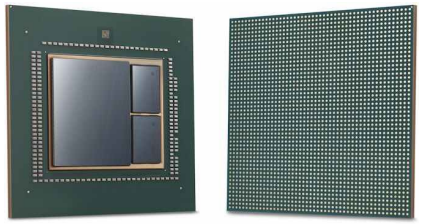
45TFLOPS

GPGPU V100

14TFLOPS

Microsoft Brainwave
on Stratix 10

8.6TFLOPS



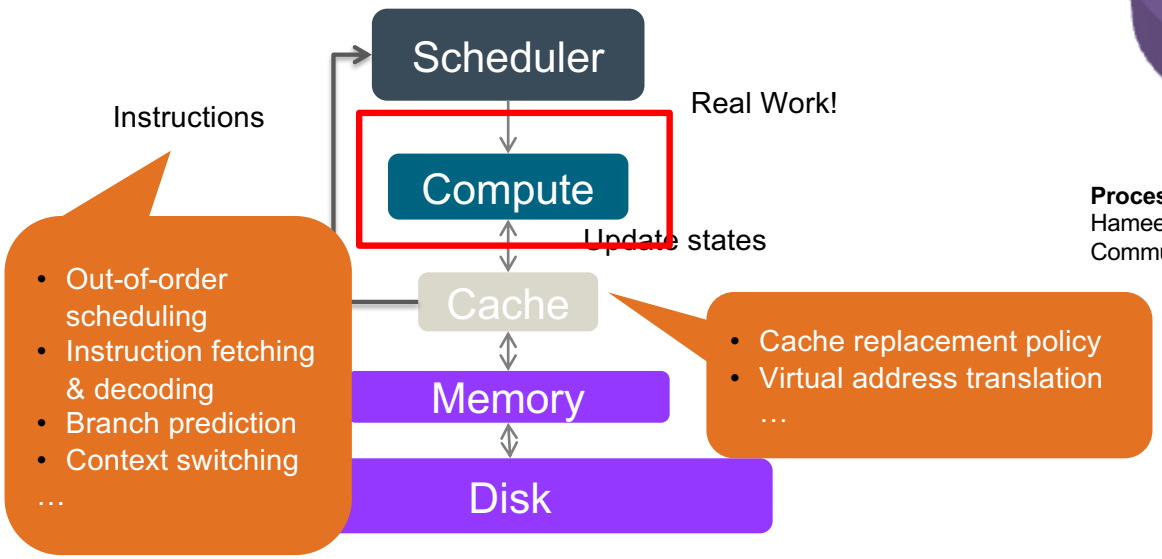
Baidu & Samsung
Kunlun AI accelerator

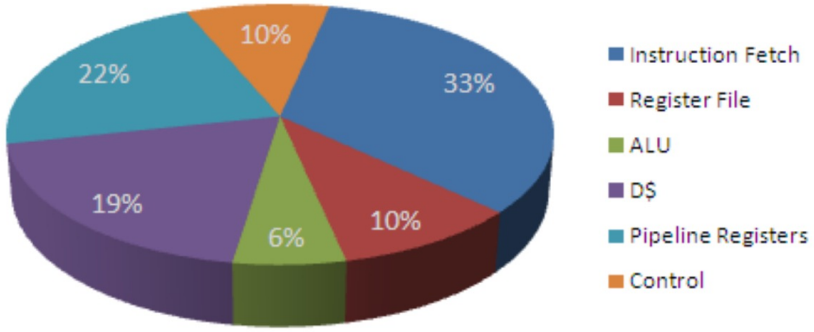
260TOPS

- **Specialized Hardware Accelerators!**

General Purpose Computing

- von Neumann architectures (processor architectures)

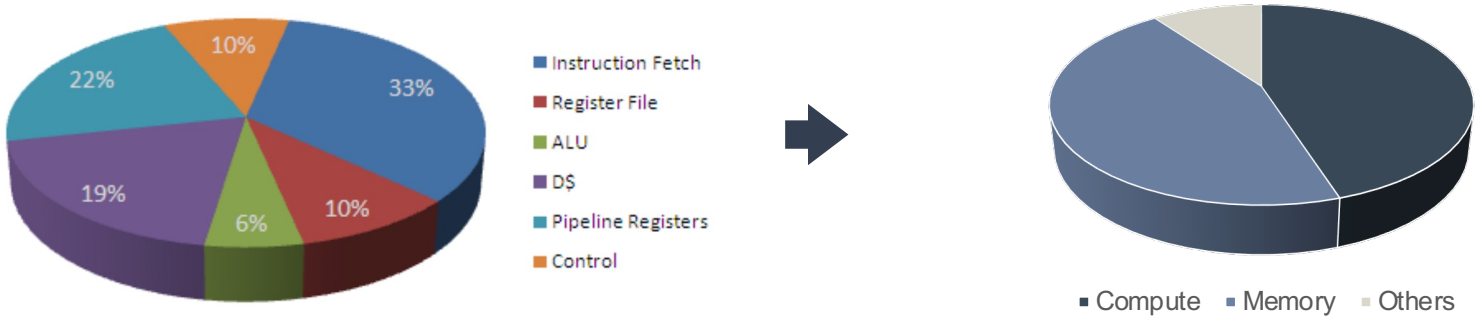




Processor Energy Breakdown
Hameed et al, Understanding Sources of Inefficiency in General-Purpose Chips, Communications of the ACM

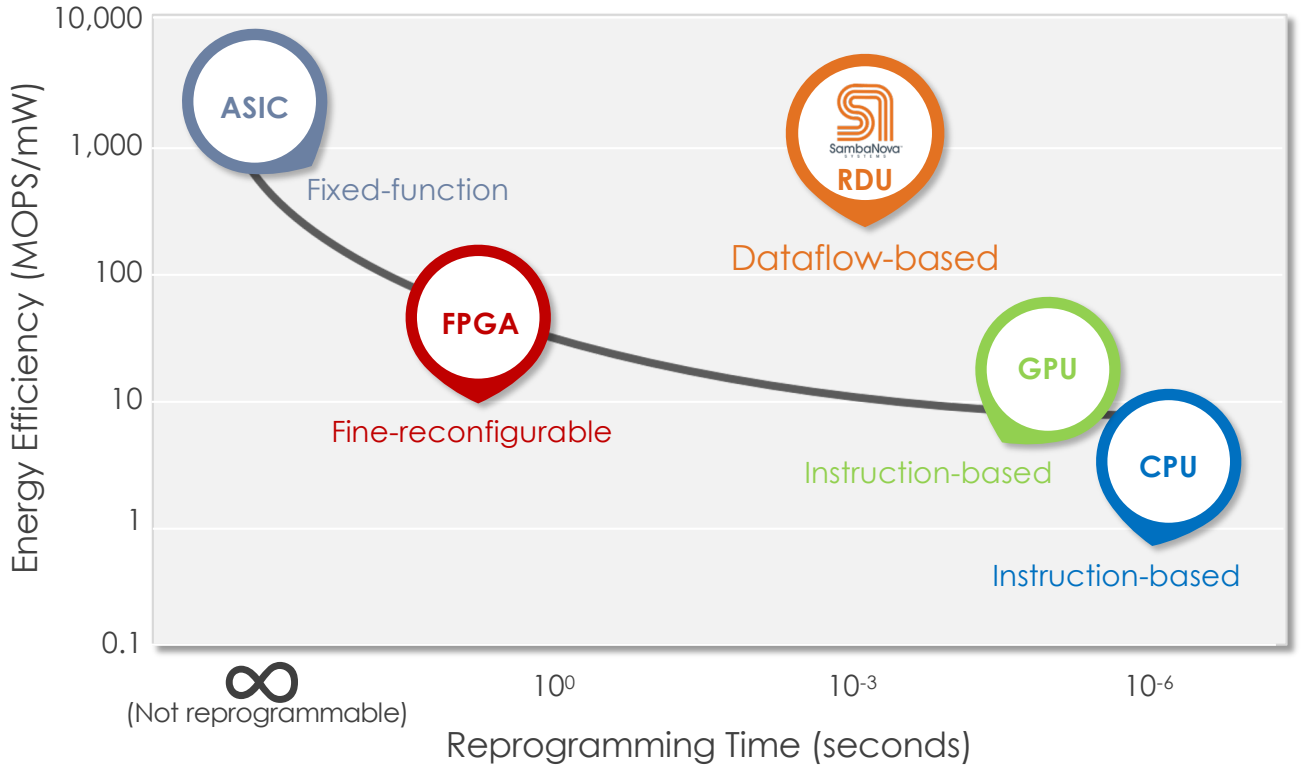
Specialization

- Reevaluate how to spend the transistor budget!

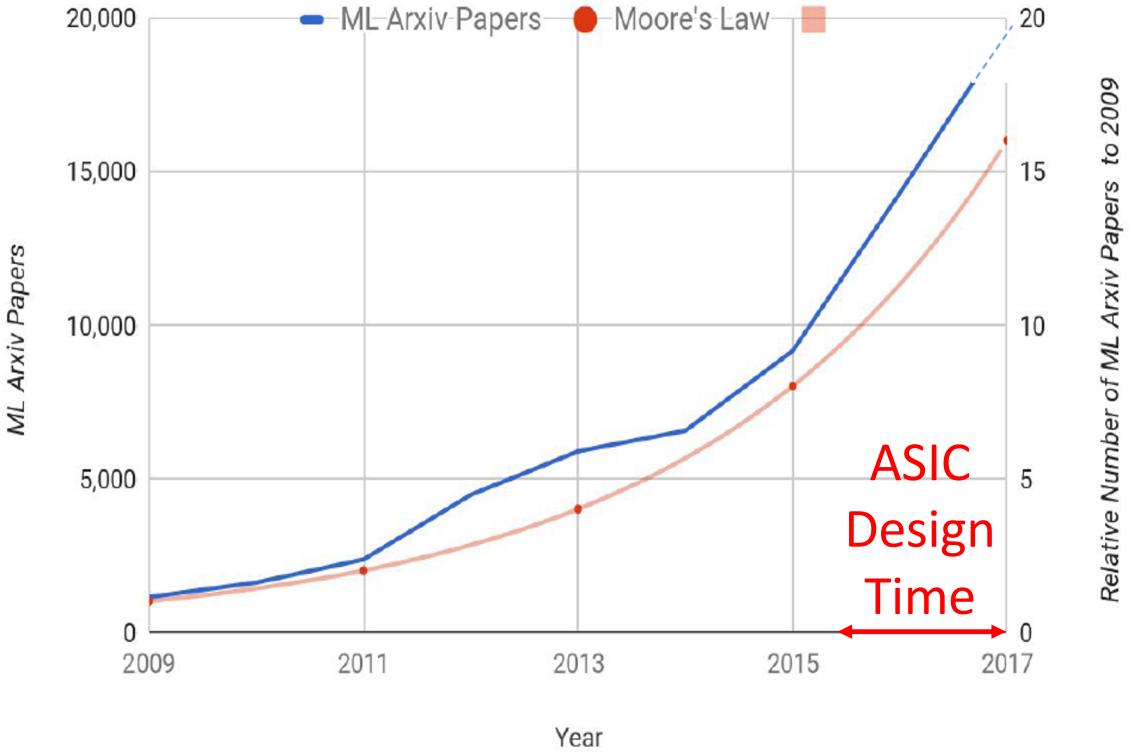


Uncompromised Programmability and Efficiency

Breaking out of the programmability vs. efficiency tradeoff curve



What to Accelerate?



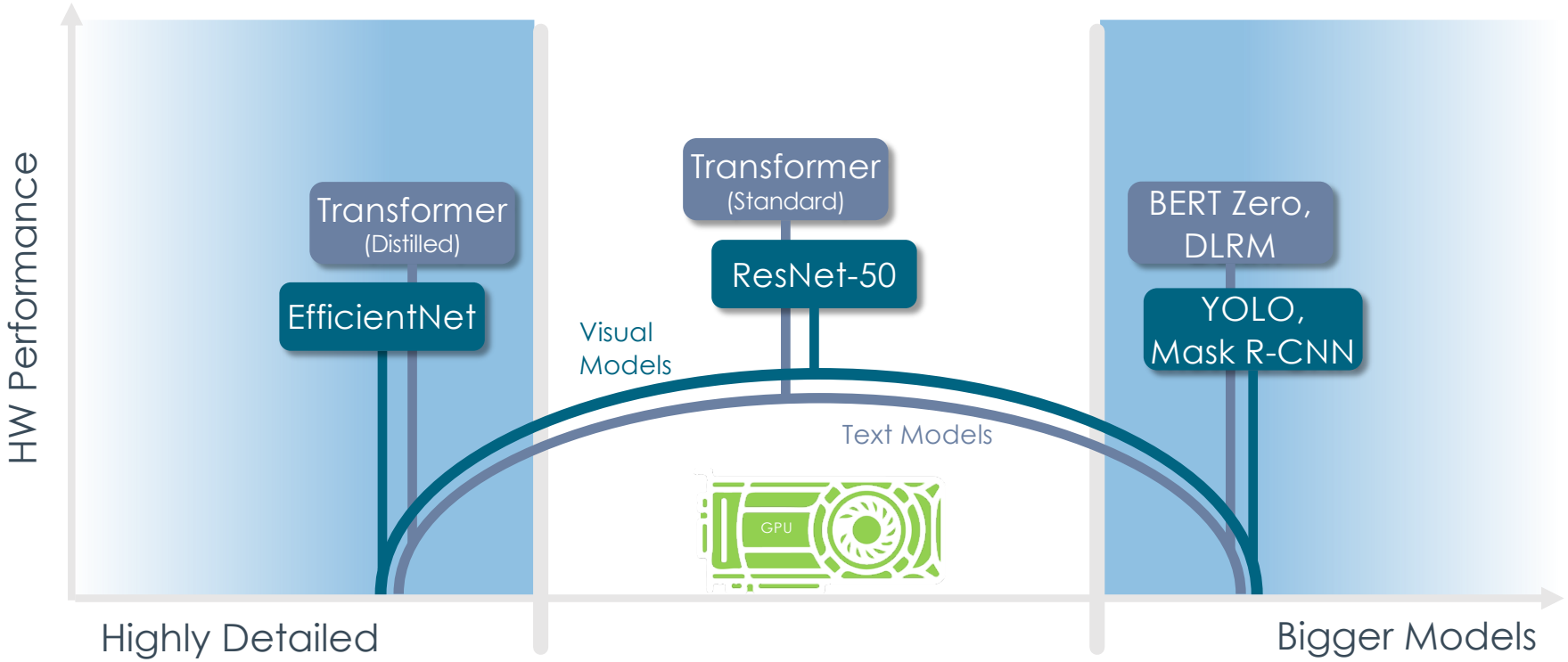
Need Configurable Accelerators

Natural Language Processing (NLP)
Used for speech recognition, translation, etc. (e.g. Siri):

LSTM (2016)
Transformer (2017)
ELMo (2018)
BERT (May 2019)
ERNIE (May 2019)
RoBERTa (July 2019)
ALBERT (Sept 2019)

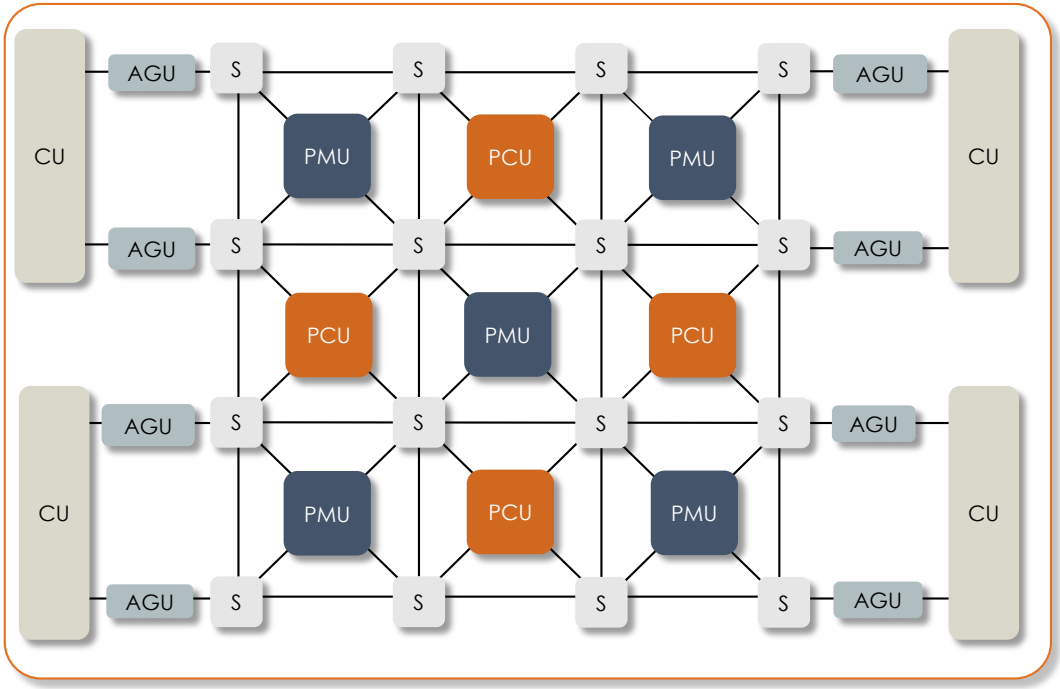
Adapted from Jeff Dean
Scaled ML 2018

Yesterday's Goldilocks Zone is Constraining Progress



Architecture: Reconfigurable Dataflow Unit (RDU)

Array of reconfigurable compute, memory and communication



- **Statically** configured datapath
- **PCU**: Pattern compute unit
  - Vectorized SIMD pipeline
- **PMU**: Pattern memory unit
  - On-chip address calculation + process 1 vector of data / cycle
- Explicitly managed on-/off-chip transfer
- Global interconnect
- Dataflow execution model
- Ultra high
  - Compute FLOPS
  - On-chip memory BW
  - On-chip network BW

SambaNova Systems Cardinal SN10 RDU

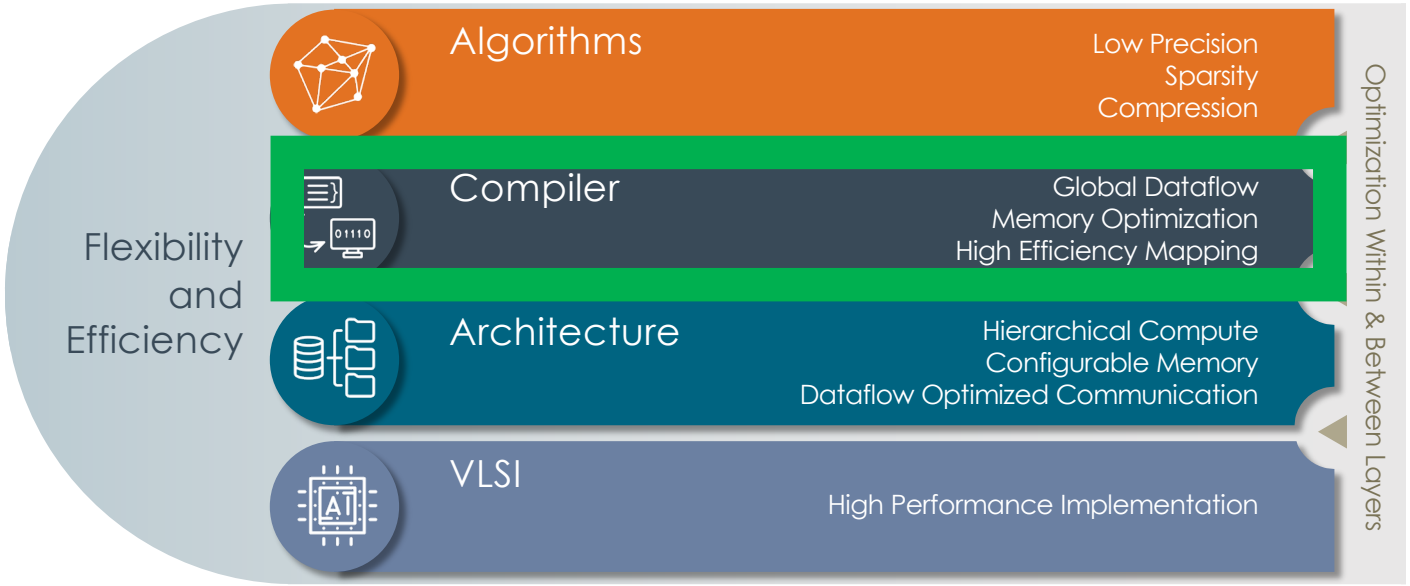
- First Reconfigurable Dataflow Unit (RDU)
- TSMC 7nm
- 40B transistors
- 50 Km of wire
- 100s of TFLOPS
- 100s MB on chip
- Direct interfaces to TBs off chip

Compiler

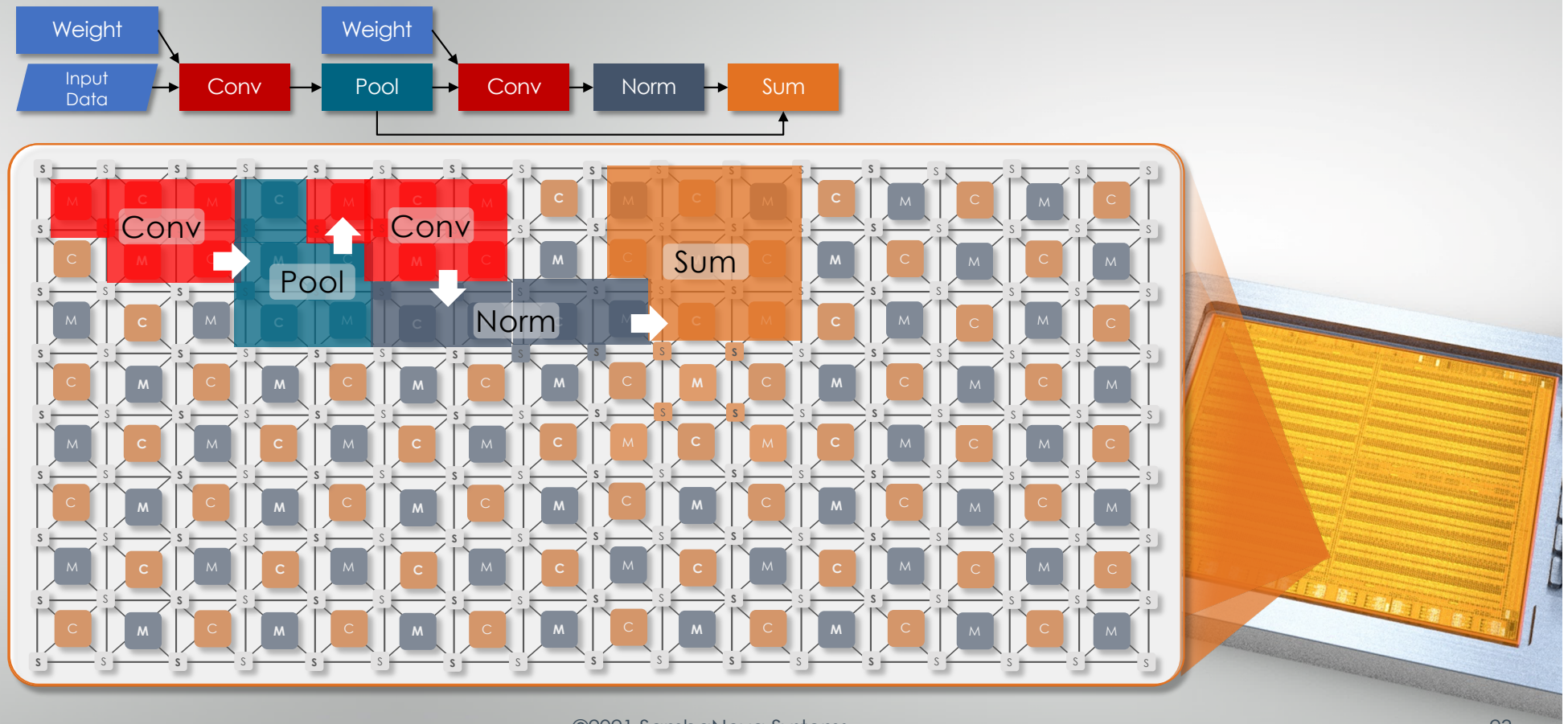
Scaling performance with programmability

The SambaNova Systems Advantage

Full stack co-engineering yields optimizations where best delivered with the highest impact

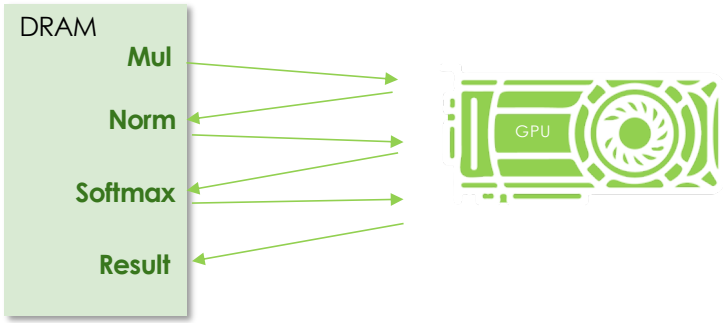


Rapid Dataflow Compilation to RDU

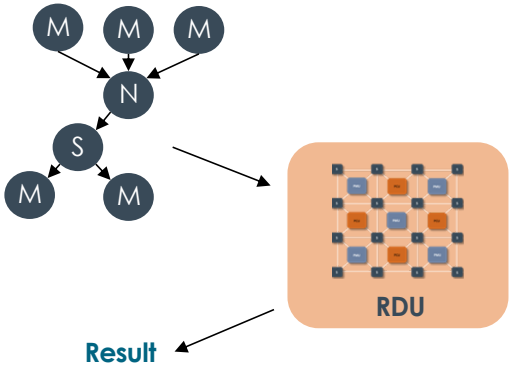


Spatial Dataflow Within an RDU

The old way: kernel-by-kernel

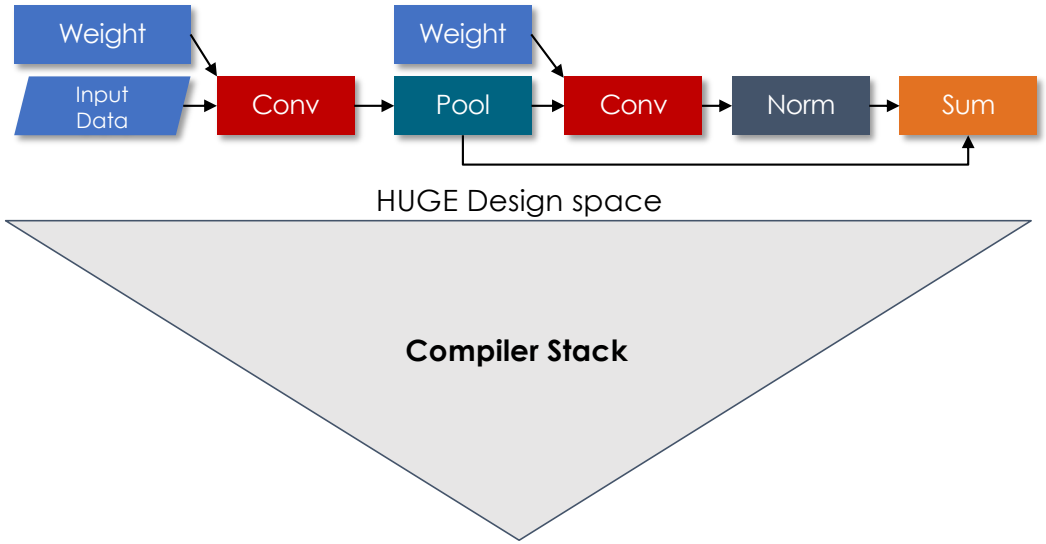


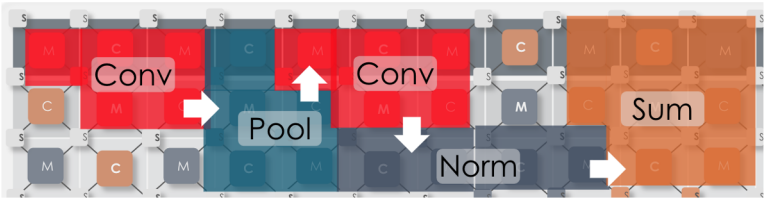
The Dataflow way: spatial



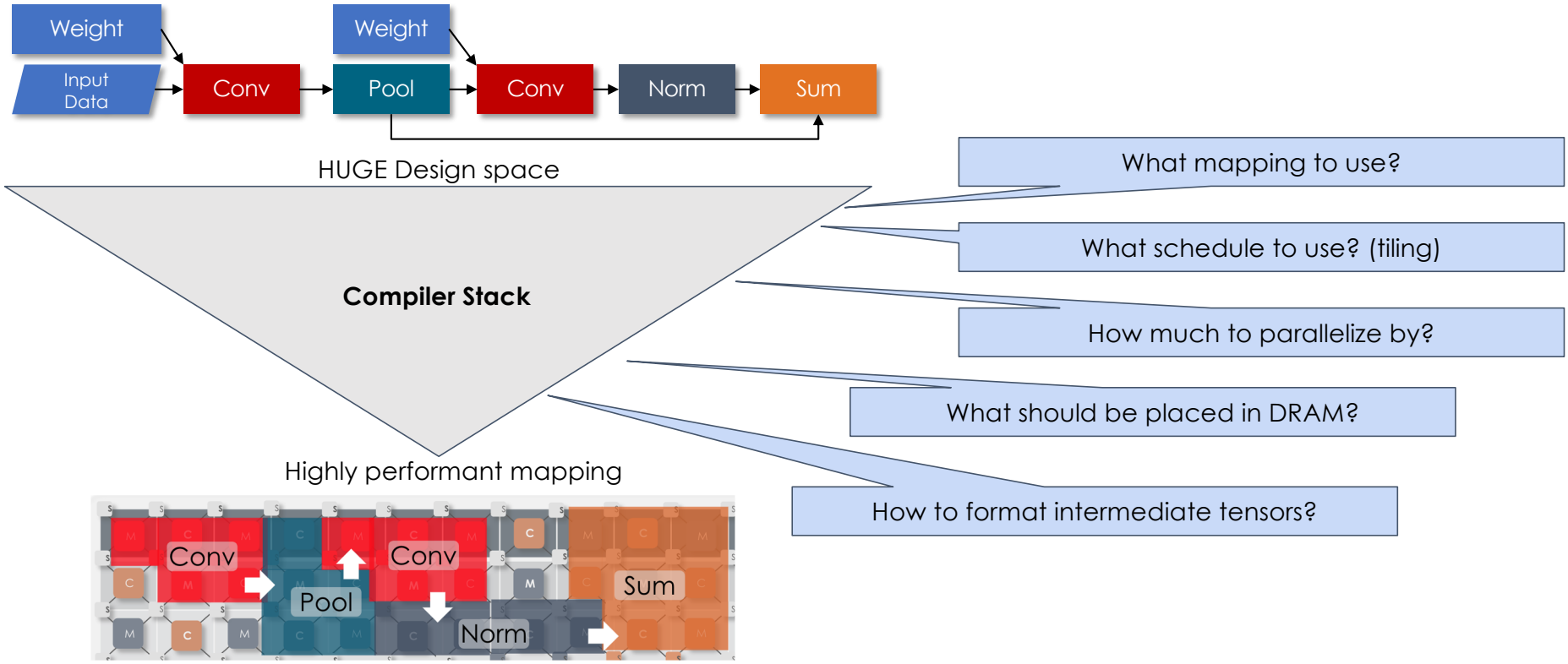
RDU programming eliminates overhead and maximizes utilization

Rapid Dataflow Compilation to RDU

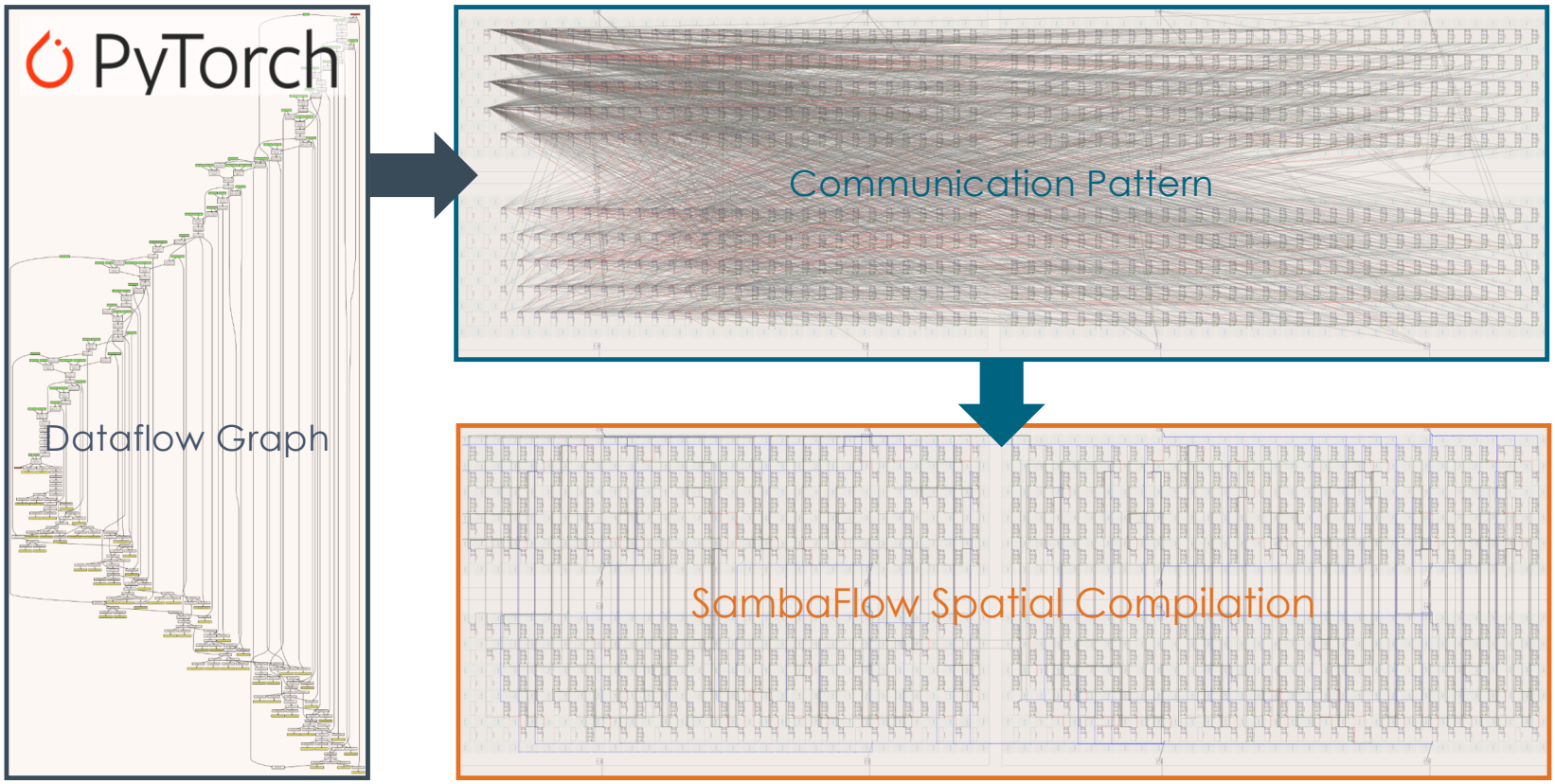




Rapid Dataflow Compilation to RDU



Placement & Routing for RDUs



Algorithms

High model accuracy

Outline

- Pure low-precision training

  Matching FP32 training accuracy with only efficient 16-bit compute units

- Asynchronous pipeline parallelization

  Up to 7X higher chip utilization than synchronous methods

- Enabling model innovations

  Enabling efficient model exploration / deployment across app. domain

Algorithm
Part 1. Pure Low-precision Training

Enabling higher compute efficiency

Low Precision (< 32-bit) Training

Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to $+1$ or -1

Matthieu Courbariaux[*1] MATTHIEU.COURBARIAUX@GMAIL.COM
Itay Hubara[*2] ITAYHUBARA@GMAIL.COM
Daniel Soudry[3] DANIEL.SOUDRY@GMAIL.COM
Ran El-Yaniv[2] RANI@CS.TECHNION.AC.IL
Yoshua Bengio[1,4] YOSHUA.UMONTREAL@GMAIL.COM

[1]Université de Montréal
[2]Technion - Israel Institute of Technology
[3]Columbia University
[4]CIFAR Senior Fellow
*Indicates equal contribution. Ordering determined by coin flip.

Recurrent Neural Networks With Limited Numerical Precision

Joachim Ott[*], Zhouhan Lin[‡], Ying Zhang[‡], Shih-Chii Liu[*], Yoshua Bengio[‡†]
[*]Institute of Neuroinformatics, University of Zurich and ETH Zurich
ottj@ethz.ch, shih@ini.ethz.ch
[‡]Département d'informatique et de recherche opérationnelle, Université de Montréal
[†]CIFAR Senior Fellow
{zhouhan.lin, ying.zhang}@umontreal.ca

Training Deep Neural Networks with 8-bit Floating Point Numbers

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen and Kailash Gopalakrishnan
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{nwang, choij, danbrand, cchen, kailash}@us.ibm.com

Higher system efficiency, minimal impact on acc. for **specific models**

Efficiency of Low Precision Compute Units (16 vs. 32-bit)[1, 2]

Benefits of low-precision DL:

- higher FLOP density
- higher tensor/weights capacity
- higher tensor/weights bandwidth

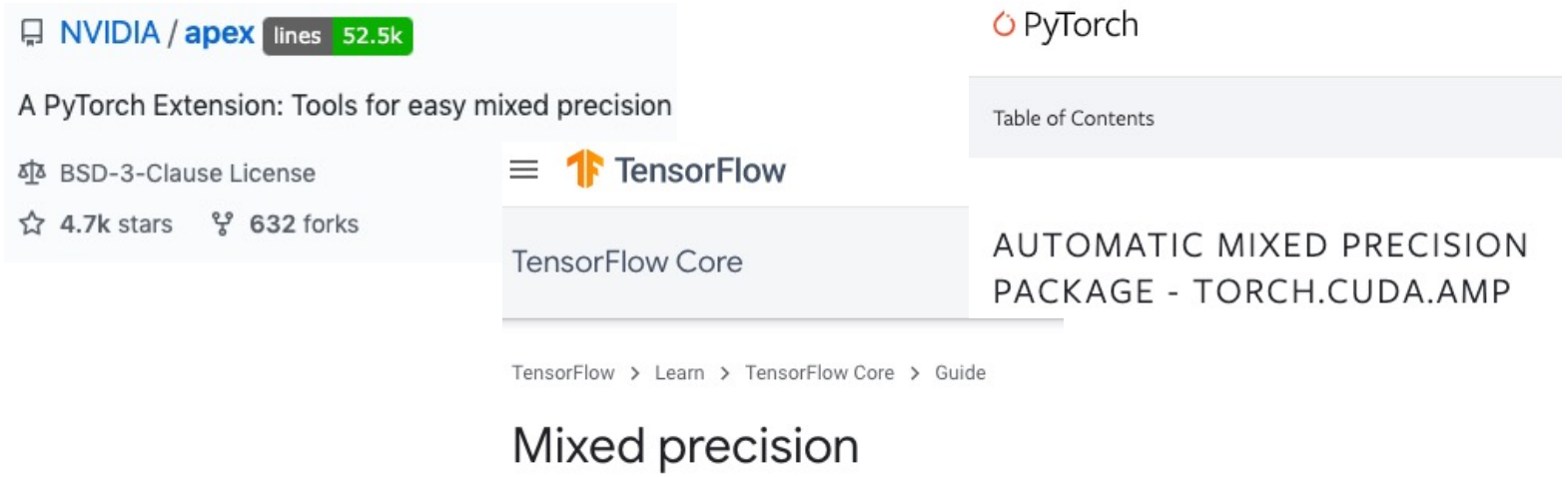
1.5X lower chip area

3X higher energy efficiency

1.5X higher throughput

1. Horowitz. ISSCC 2014
2. Galal et. al. ISCA 2013

Mixed Precision for *Generic* DL Training (16 + 32 bits)



Folklore:

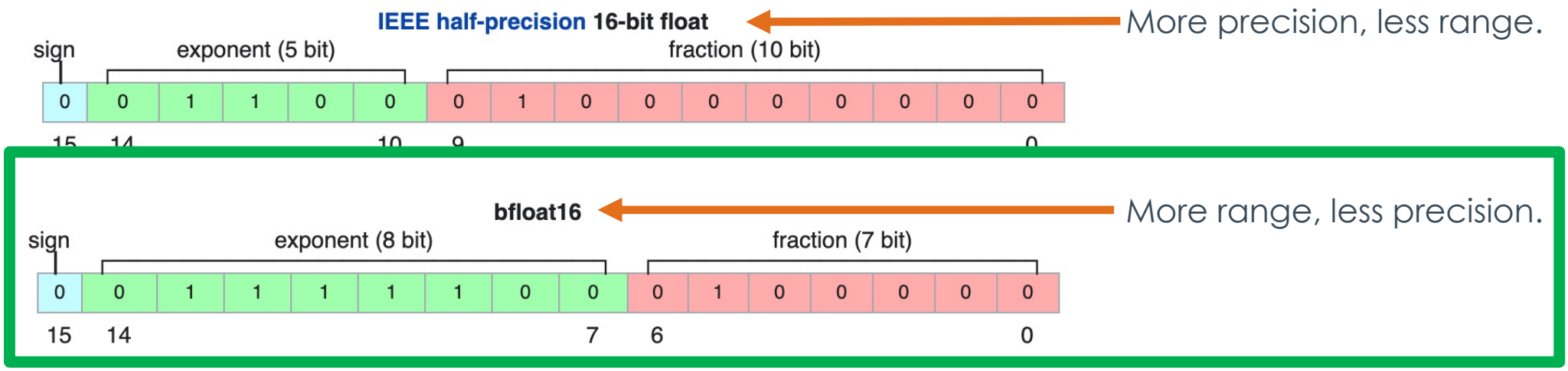
16-bit precision alone is not enough to maximize model acc.

Can we support ***only 16-bit compute units*** *on accelerators*

&

achieve model acc. matching 32-bit training?

A bit about precision

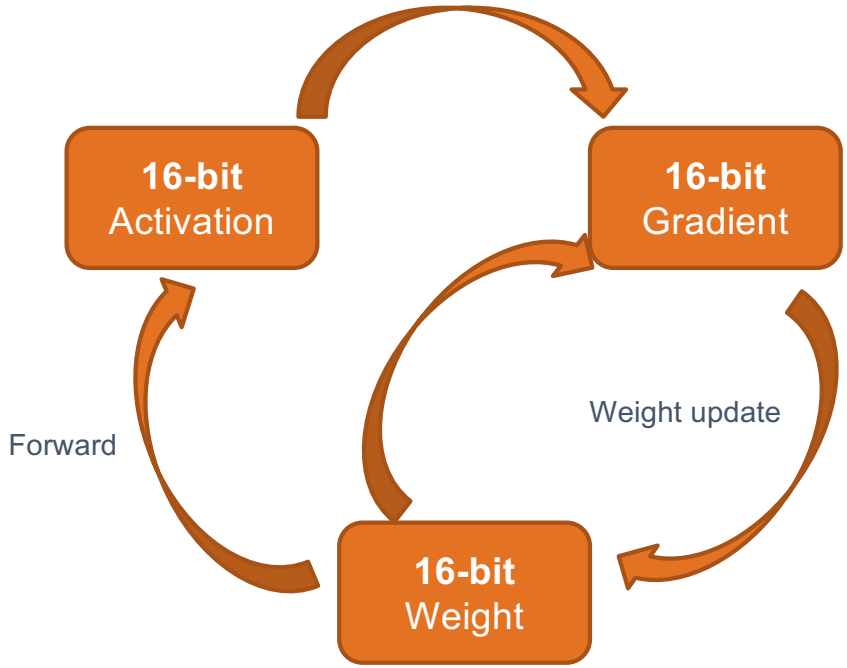


"Neural networks are far more sensitive to the size of the exponent than that of the mantissa."

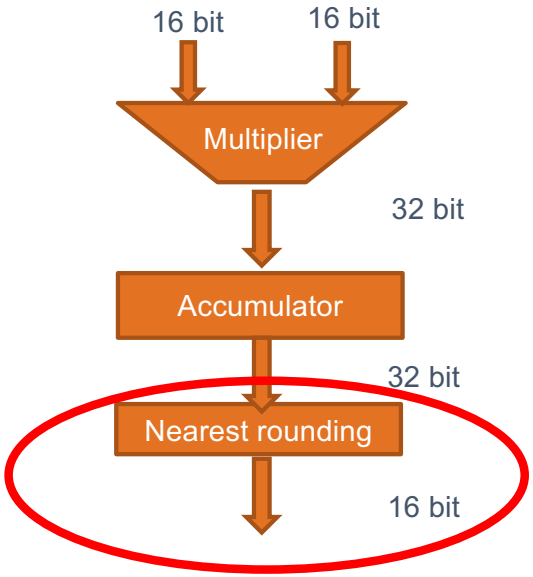
Can BFloat16 **only training** match the quality 32-bit floating point training?

Pure 16-bit (BFloat16) Training

Data Flows

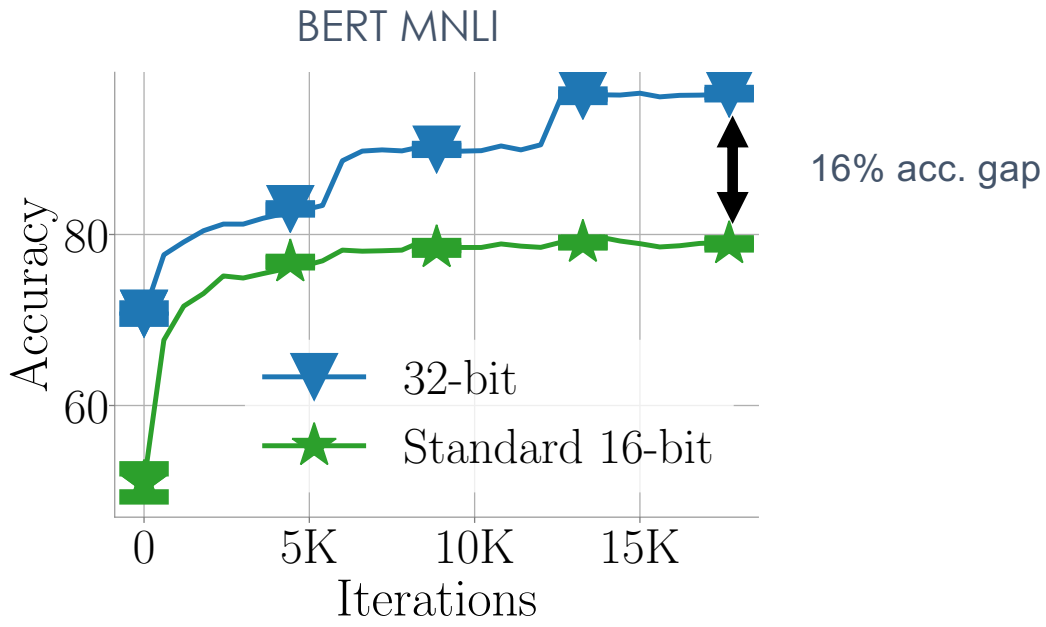


Multiply-Accumulation Units



Primary source of numerical error

The Accuracy Challenge



Standard pure 16-bit training degrades model accuracy

The Devil: Nearest Rounding(NR) for Model Weight Updates

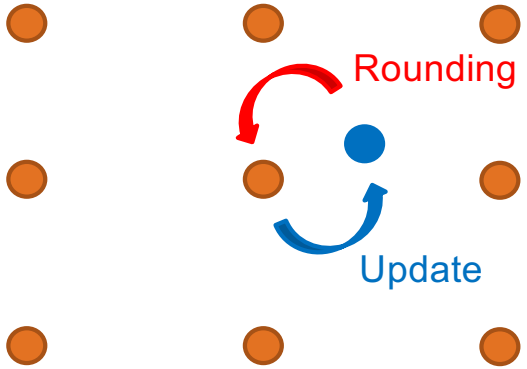
$$\boldsymbol{w}_{t+1} = \boldsymbol{Q}\left(\boldsymbol{w}_t - \alpha \nabla f_{\sigma(t)}(\boldsymbol{w}_t)\right)$$

Model weight

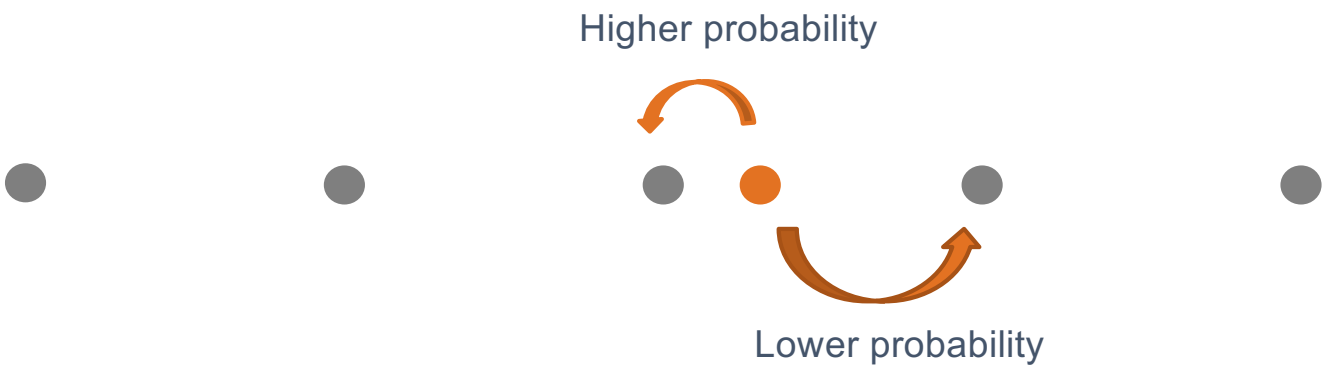
Nearest Rounding

Minibatch

Model weights halt when updates becomes small



Stochastic Rounding to the Rescue



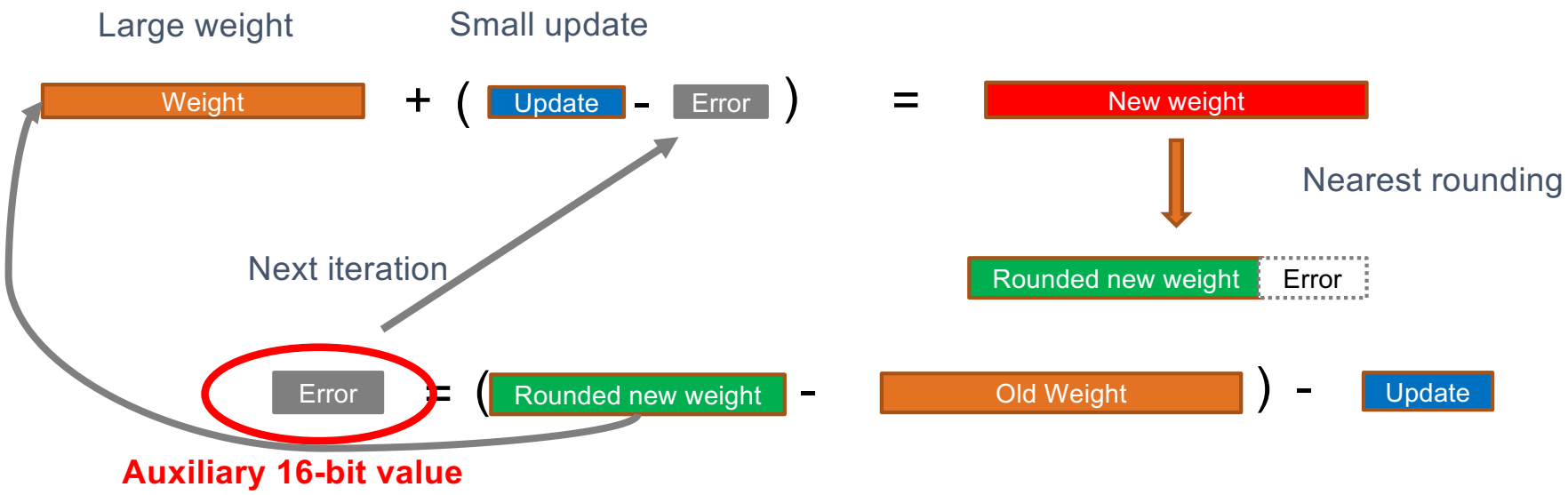
Example: 2.3 rounds to 2 with 70% probability and rounds to 3 with 30% probability

Intuition

The expectation of unbiased estimates is as accurate as weights w/o rounding

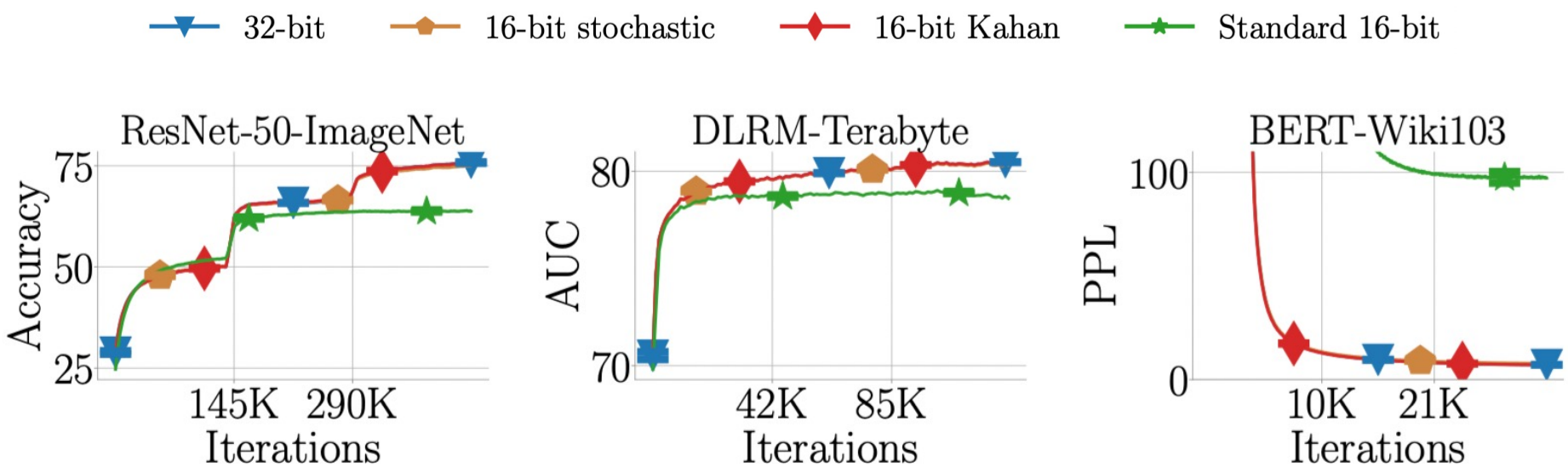
Kahan Summation as Alternative Enhancement

Auxiliary 16-bit values to track and correct weight update errors from nearest rounding error

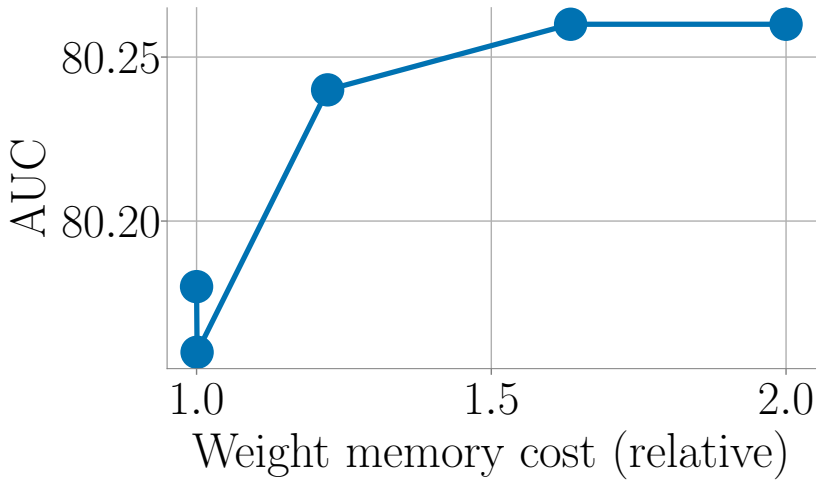


Experiment:

Pure 16-bit training can match 32-bit training in model acc.



Experiment: Efficiency-accuracy Trade-off



Kahan accumulation squeezes accuracy at the cost of memory

Revisiting BFloat16 Training [arXiv]

Summary

With support for

&

Weight + (Update + Error) = New weight

Stochastic rounding

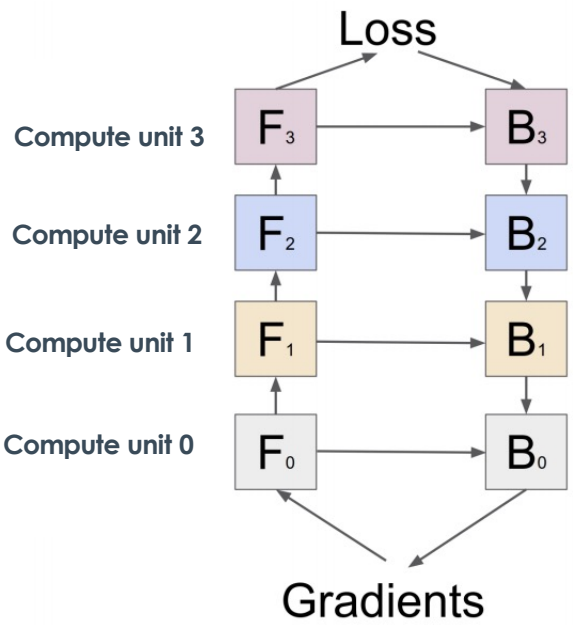
Kahan summation

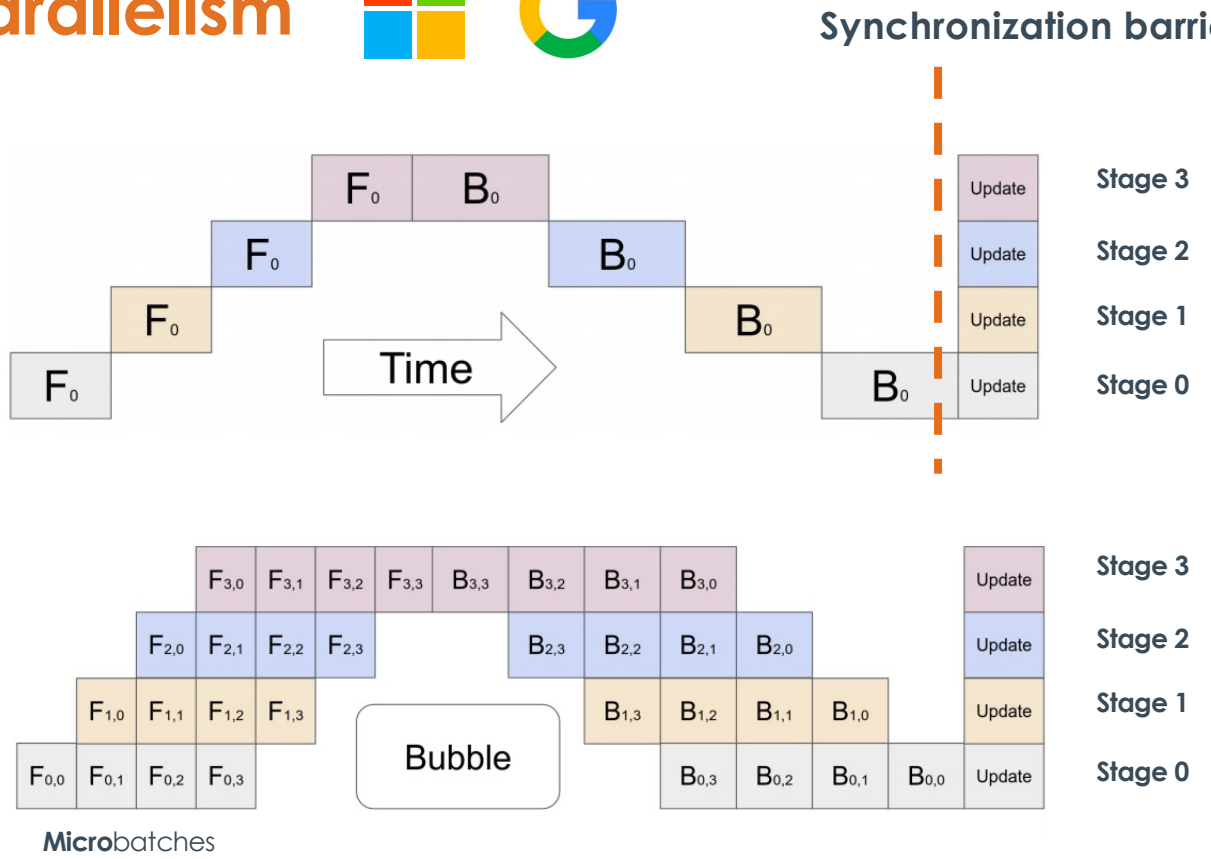
Accelerators with only 16-bit compute units can match acc. of 32-bit training

Part 2. Async. Pipeline Parallelization:

Enabling higher compute utilization

Model (Pipeline) Parallelism





1. Huang et. Neurips 2019

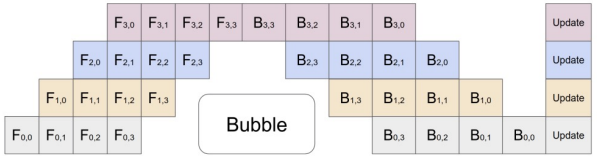
Model (Pipeline) Parallelism: Are we there yet?

Conventional processor pipeline

# of pipeline stages ↑ Throughput ↑

Model training pipeline with synchronization barrier

# of pipeline stages ↑ Utilization ↓

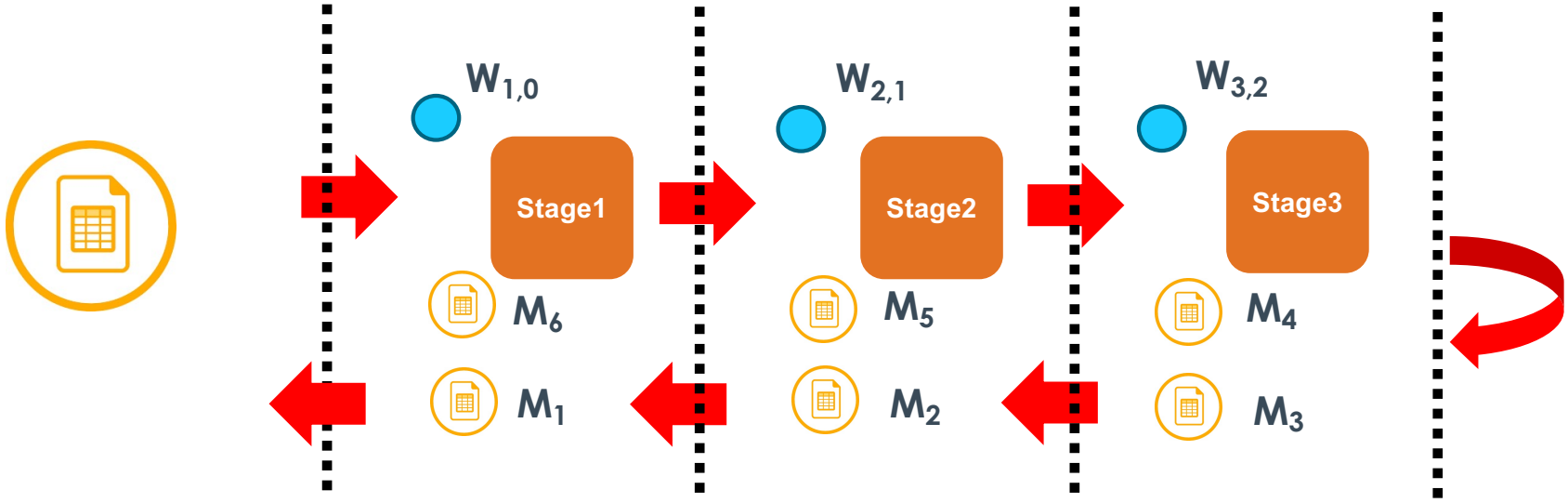


How much utilization do we really need to sacrifice?

Async. Pipeline Parallelism Steady State

$W_{i,j}$ Stage i weight after j-th update

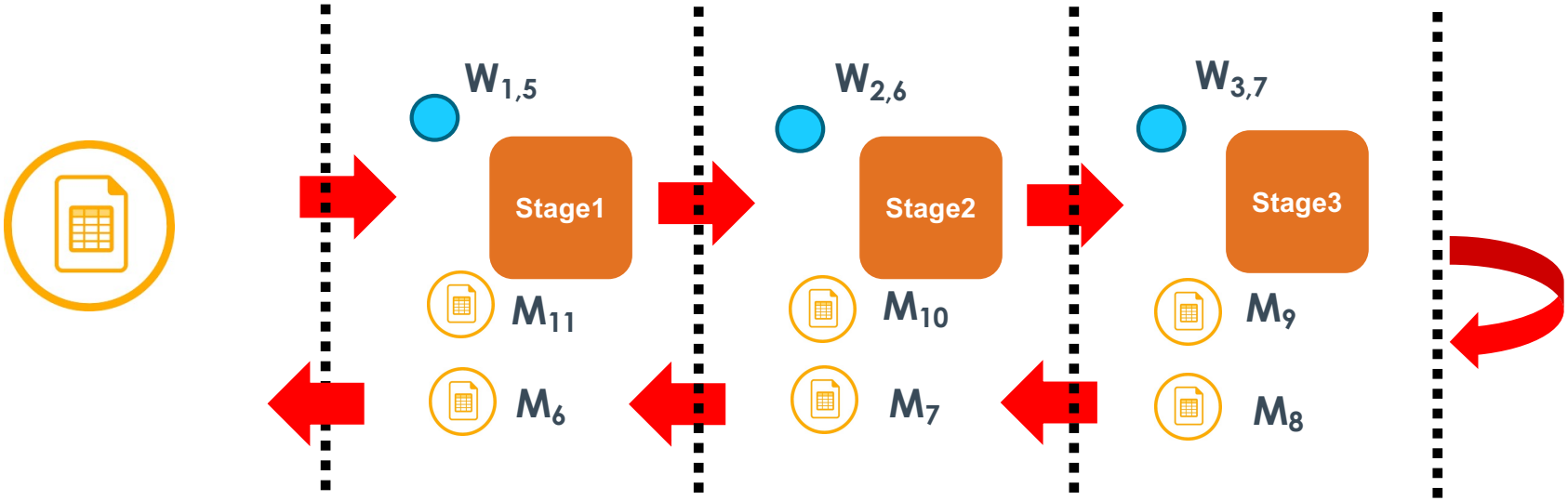
M_i i-th minibatch



Goal: No hardware sacrifices!

Async. Pipeline Parallelism Steady State

$W_{i,j}$ Stage i weight after j-th update M_i i-th minibatch

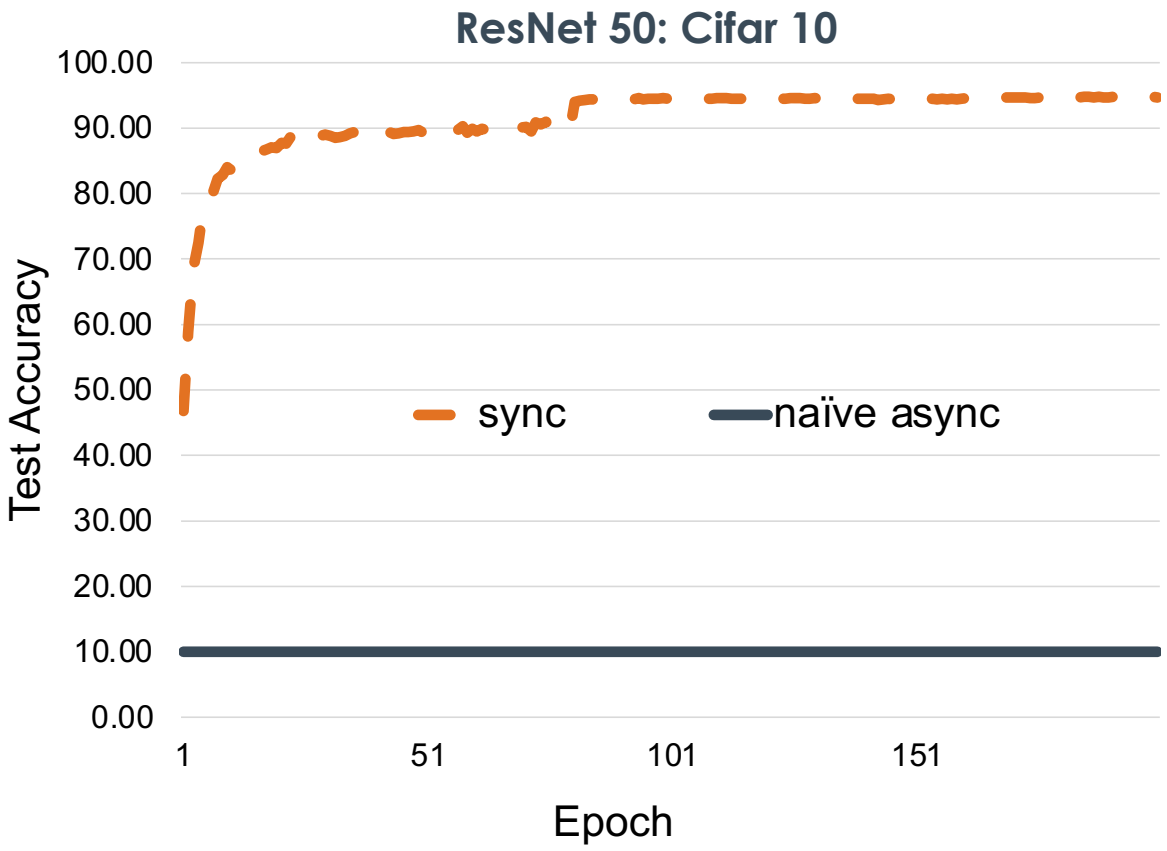


M_6 uses $W_{1,0}$ for forward and $W_{1,5}$ for backward: delay = 5

M_6 uses $W_{3,4}$ for forward and $W_{3,5}$ for backward: delay = 1

Panic: Introduces different **asynchrony** (delays) at different stages.

Houston, we have a problem.



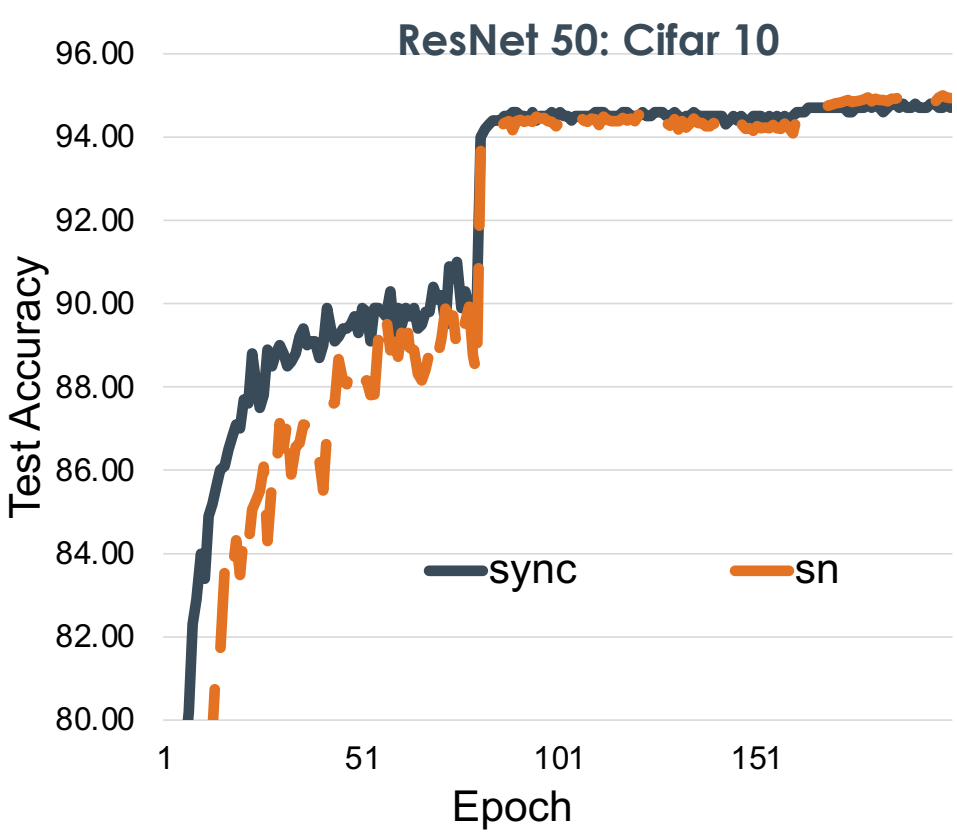
Key Insight: Scale your learning rate proportional to the delay.

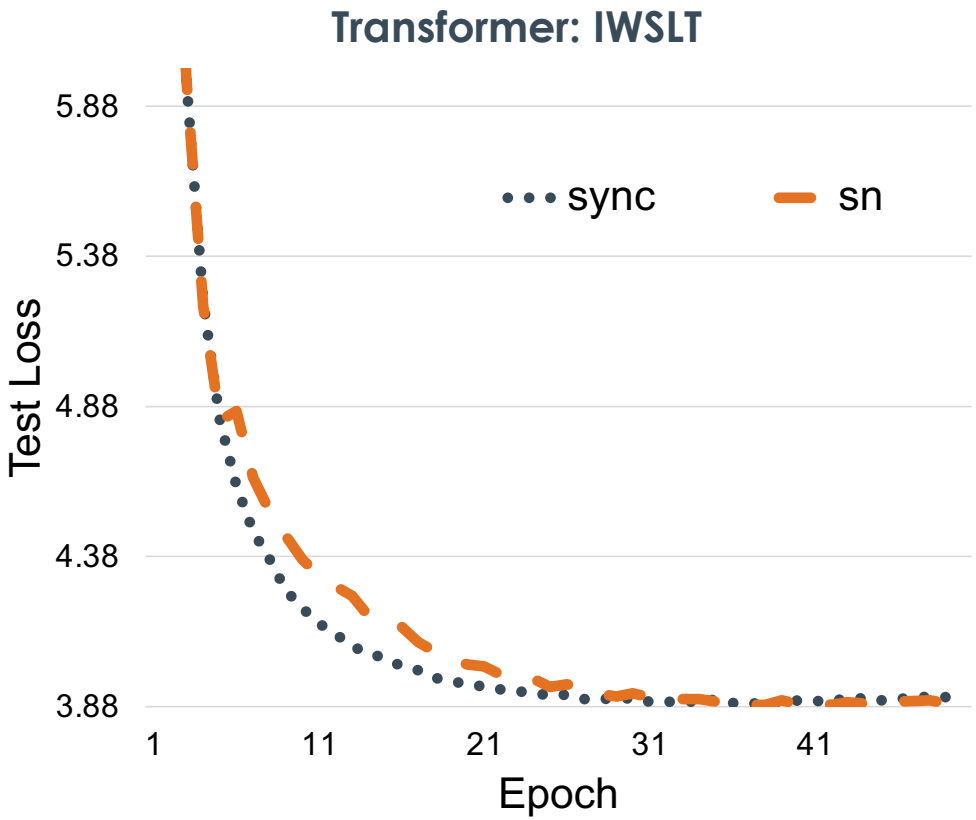
$$\alpha = \min\left(\alpha_{\text{sync}}, \frac{C}{\tau_i}\right)$$

Chris De Sa

PipeMare: Arxiv '20

Maximize efficiency with no accuracy compromise



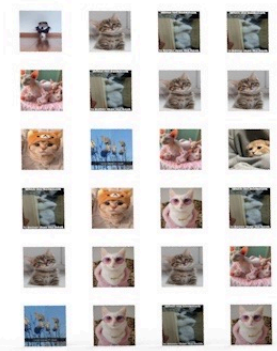


Part 3. Model Innovations:

Powered by our architecture and algorithm

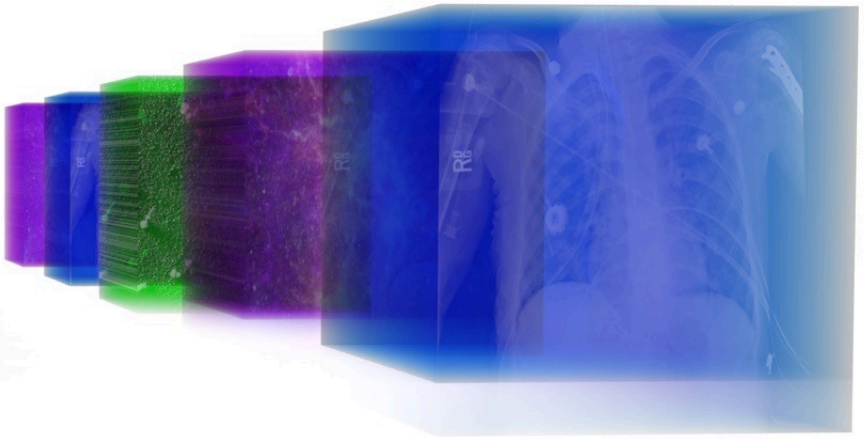
Computer Vison

Evolution of high-resolution Deep Learning



Low-resolution
(e.g. cats)

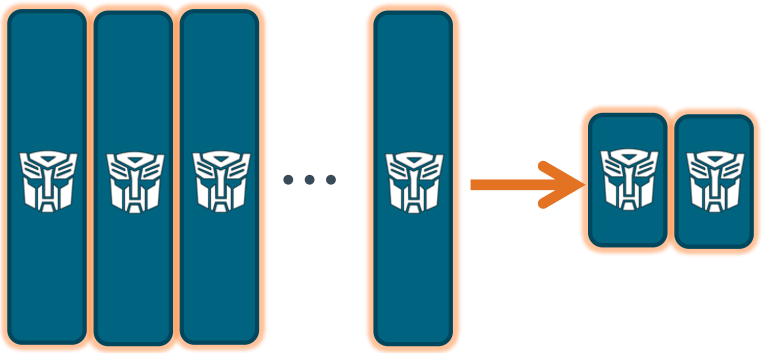
4k images
(e.g. Autonomous driving)



50k x 50k
(e.g. astronomy,
medical imaging, virus, …

Natural Language Processing

Breakthrough efficiency in NLP model online deployment



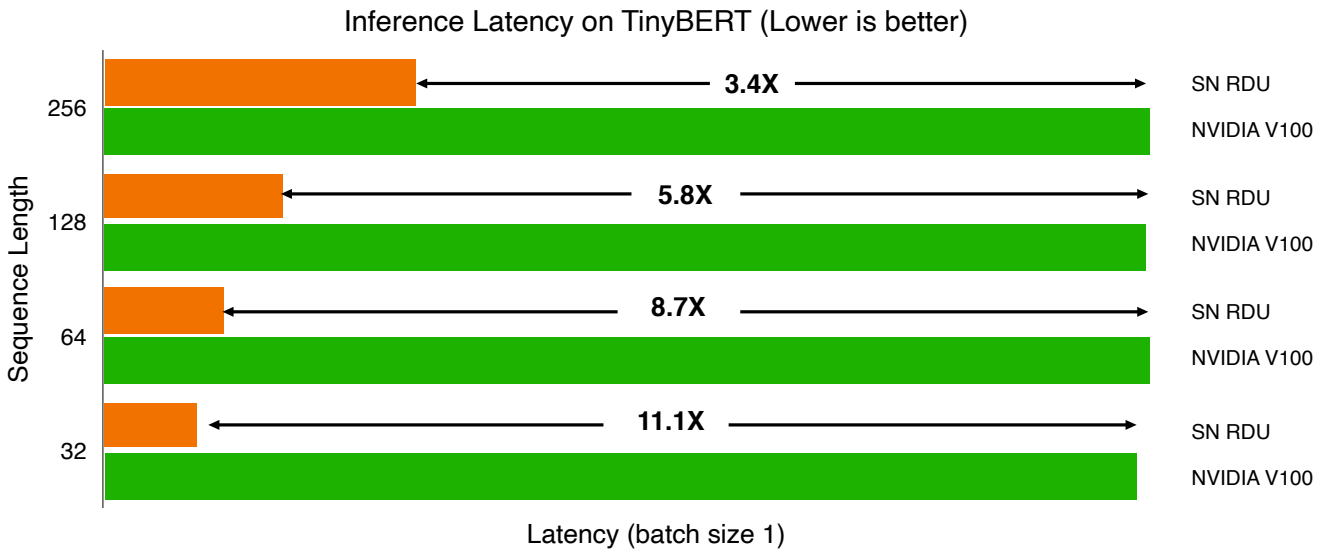
Distilled tiny Bert model

+

Siri, what is the weather in SF?

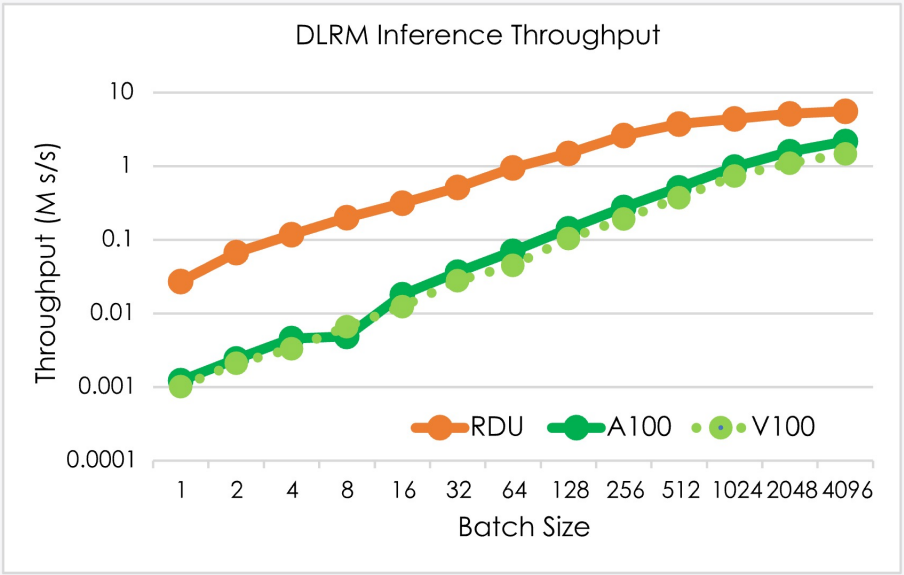
Short sequence input

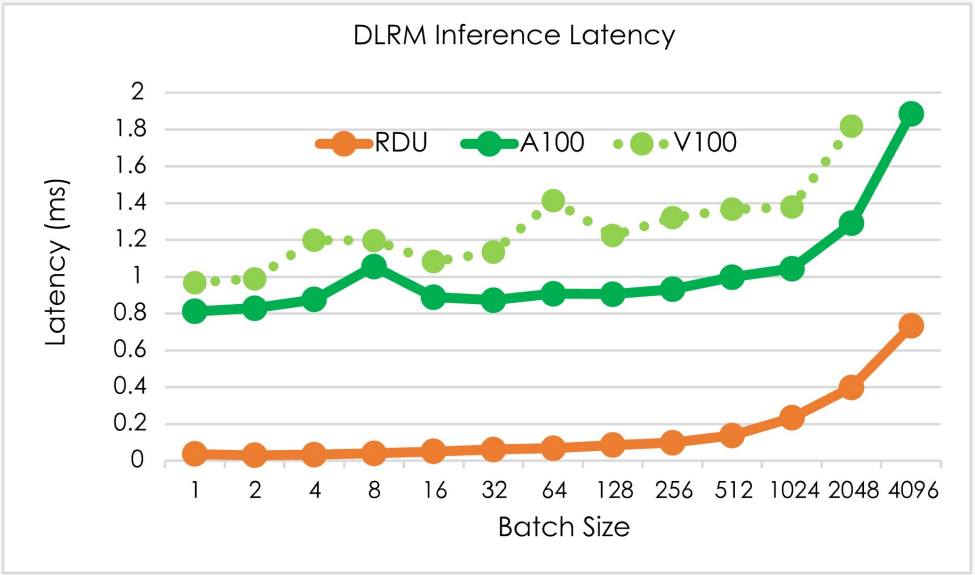
Breakthrough Efficiency in NLP Model Online Deployment



Enable up to 11X speedup for online training and inference

Facebook Deep-learning Recommendation System (DLRM)





Small Batch Size Inference

We're hiring: **sambanova.ai/career**

- ML and Applications
- Compiler
- Library/Mapping
- System Software
- DevOps/Release Engineering
- Computer Architecture
- VLSI
- **Office**: Palo Alto & Austin & wherever you are!

Join us to shape the future together

Thank You!

sambanova.ai

sambanova-systems

@SambaNovaAI

SambaNovaAI