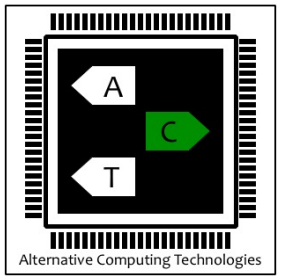
Scale-Out Acceleration for Machine Learning

Jongse Park†
Hardik Sharma†
Divya Mahajan†
Joon Kyung Kim†
Preston Olds†
Hadi Esmaeilzadeh†‡

Alternative Computing Technologies (ACT) Lab
Georgia Institute of Technology†
University of California, San Diego‡



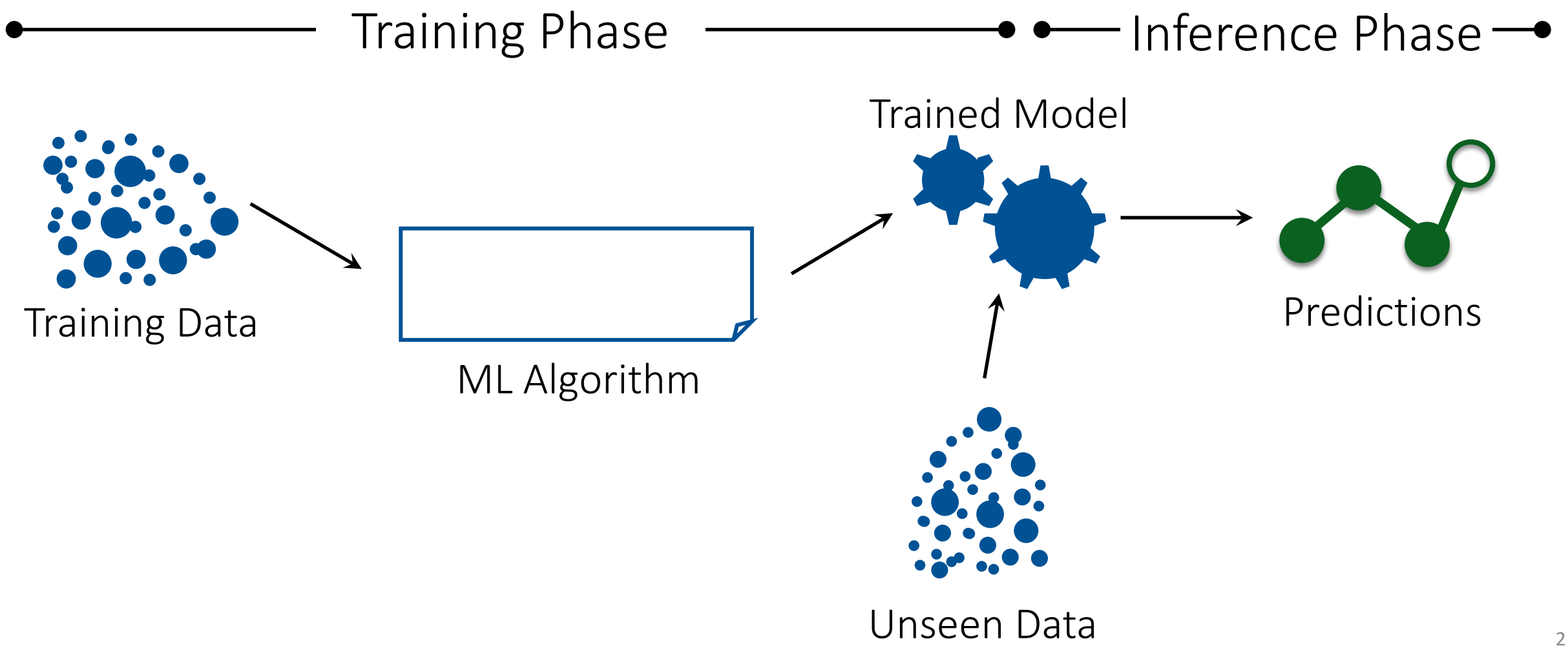
MICRO50

Data grows at an unprecedented rate

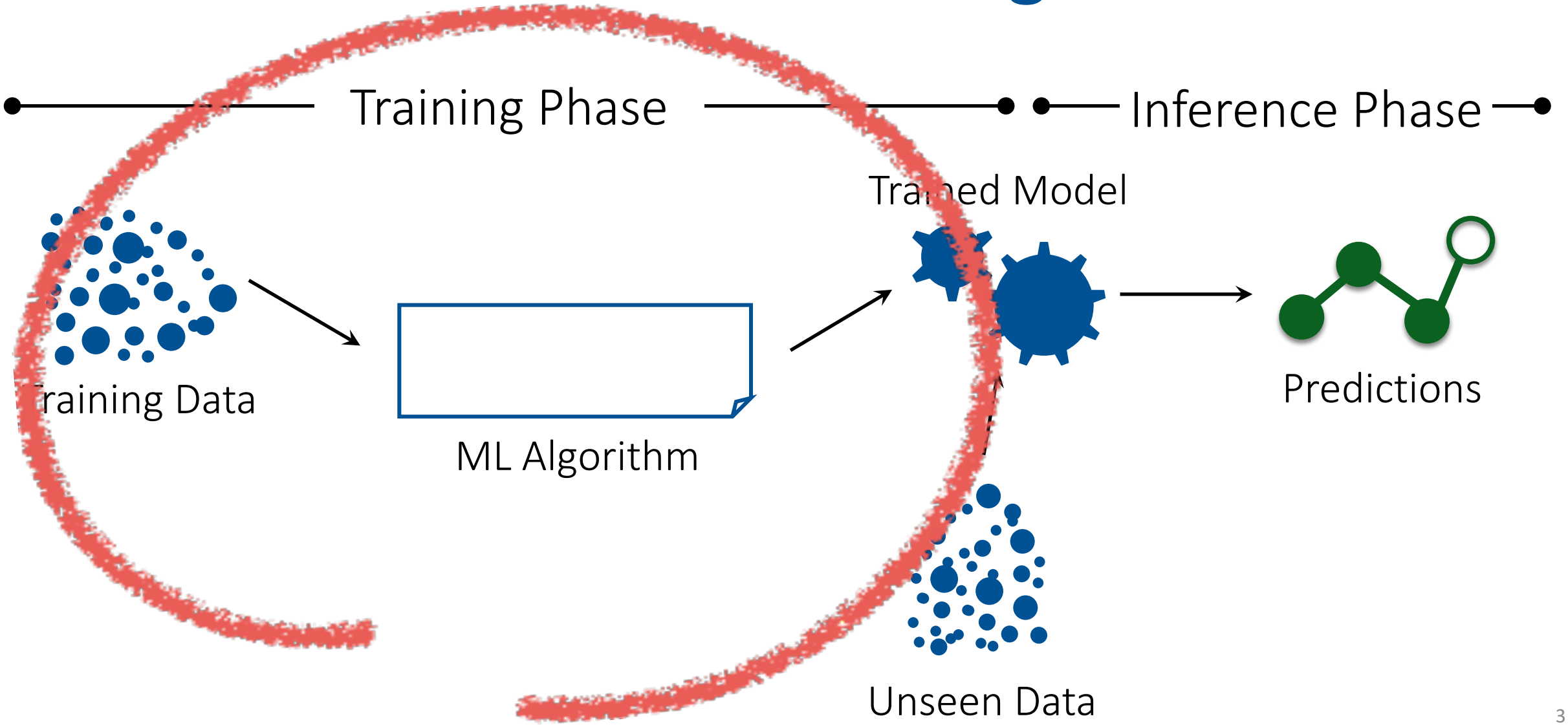
1 minute over the internet

facebook	**900,000** *Logins*
	46,200 *Posts Uploaded*
	452,000 *Tweets Sent*
	$751,522 *Spent Online*
Spotify	**40,000** *Hours Listened*
NETFLIX	**70,107 Hours** *Watched*
	156 Million *Emails Sent*
Google	**3.5 Million** *Search Queries*
	1.8 Millions *Snaps Created*
GET IT ON Google Play / Download on the App Store	**342,000** *Apps Downloaded*

Machines learn to extract insights from data

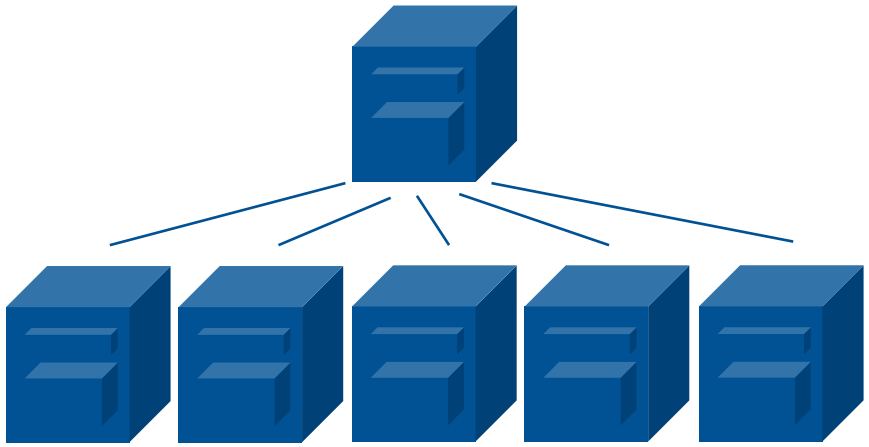


Machines learn to extract insights from data



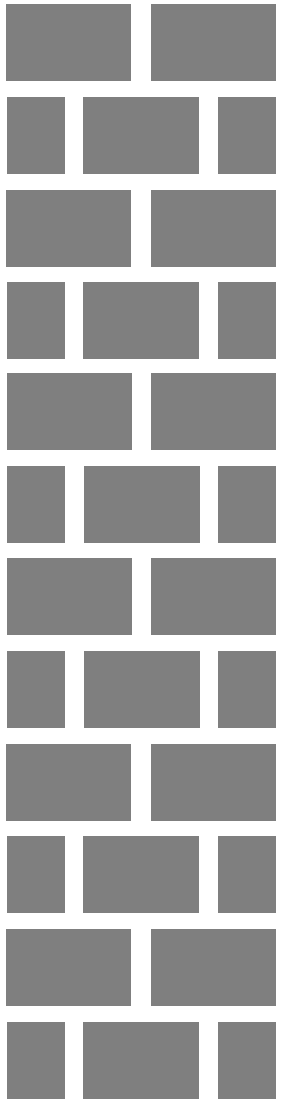
Two disjoint solutions for ML training

Distributed Computing

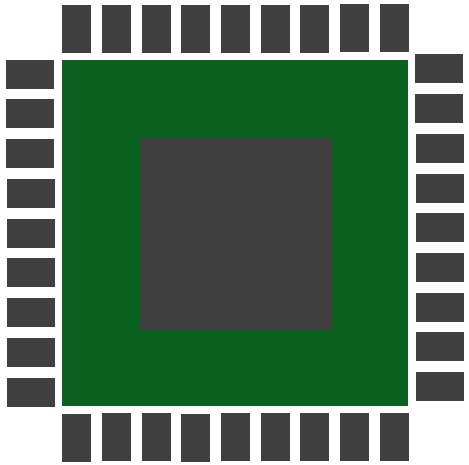


Spark MLlib TensorFlow Caffe2

mahout PaddlePaddle H2O.ai



Single-Node FPGA/ASIC Accelerators



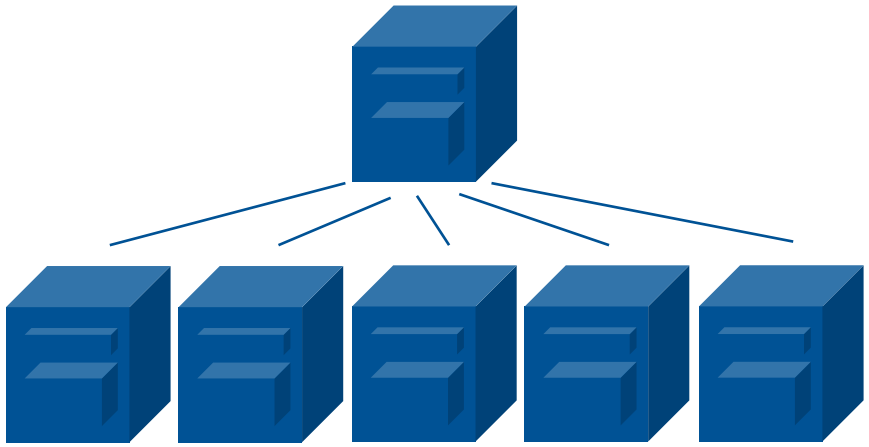
SBCCI'03 FCCM'09 FCCM'10 AHS'11

CVPRW'11 DAC'12 ASPLOS'14 ASPLOS'15

HPCA'15 HPCA'16 MICRO'16

Two disjoint solutions for ML training

Distributed Computing

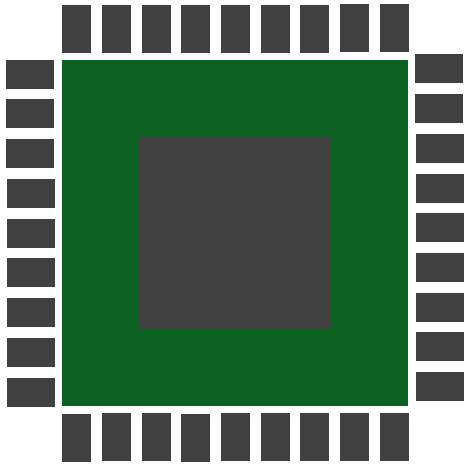


Apache Spark MLlib TensorFlow Caffe2

mahout PaddlePaddle H_2O.ai

CoSMIC

Single-Node
FPGA/ASIC Accelerators



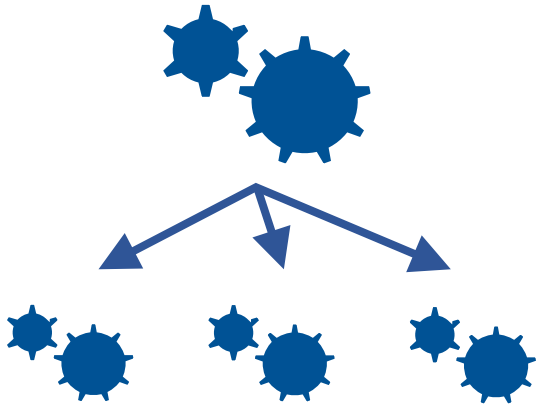
SBCCI'03 FCCM'09 FCCM'10 AHS'11

CVPRW'11 DAC'12 ASPLOS'14 ASPLOS'15

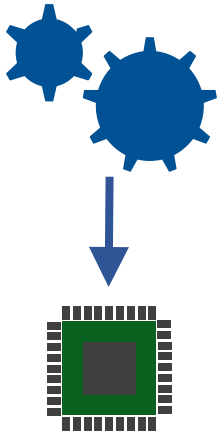
HPCA'15 HPCA'16 MICRO'16

Challenges

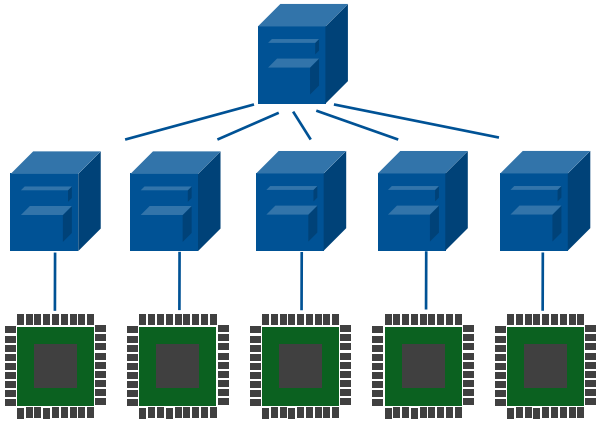
1. How to distribute ML training?



2. How to design customizable accelerators?

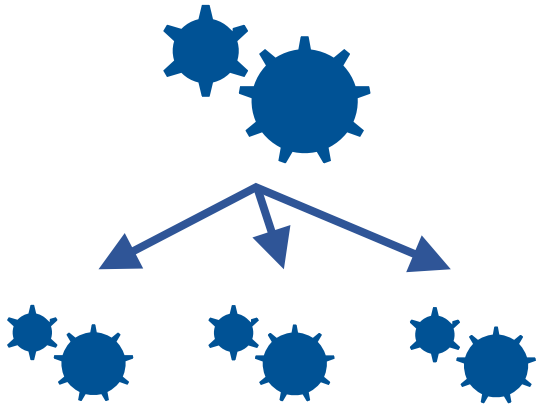


3. How to reduce the overhead of distributed coordination?

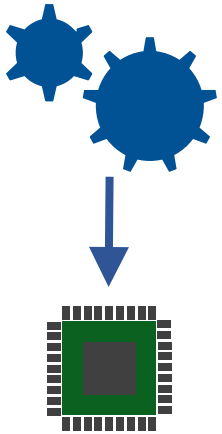


Challenges

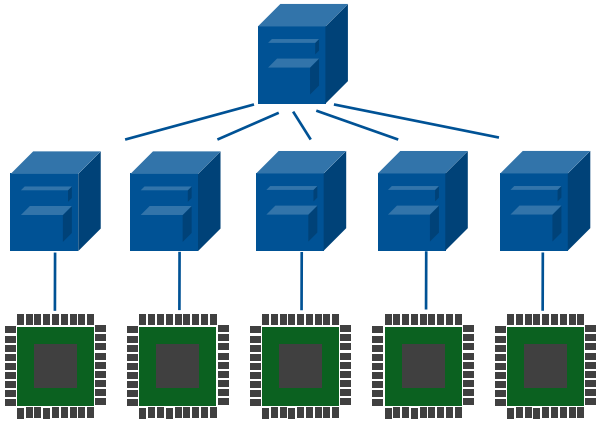
1. How to distribute ML training?



2. How to design customizable accelerators?



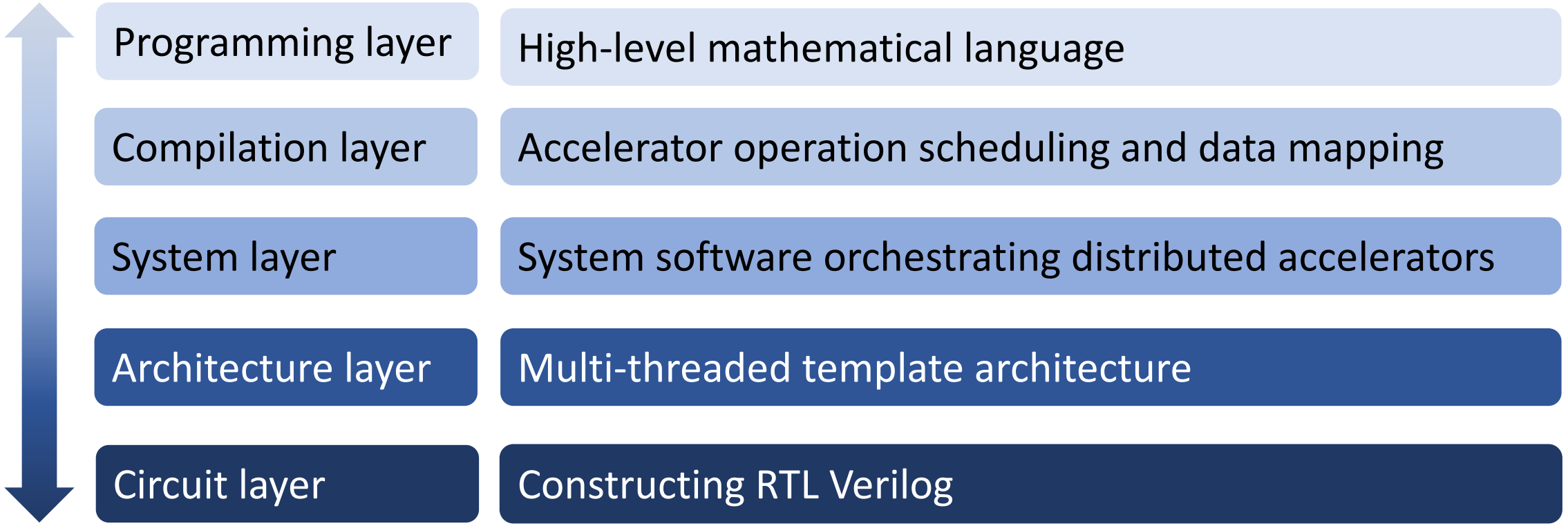
3. How to reduce the overhead of distributed coordination?



We need a full stack

CoSMIC

Computing Stack for ML Acceleration In the Cloud



CoSMIC workflow

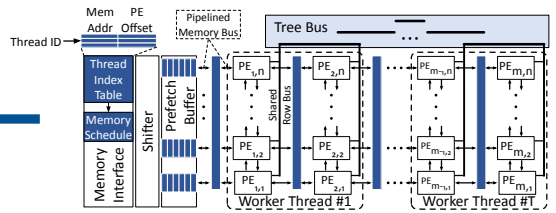
Model Specification

1 Translator

Dataflow Graph

2 Planner

Multi-threaded Template Architecture



Accelerator Architecture

3 Compiler

Operation Schedule/Map

4 Constructor

RTL Verilog for Accelerator

Circuit Constraints

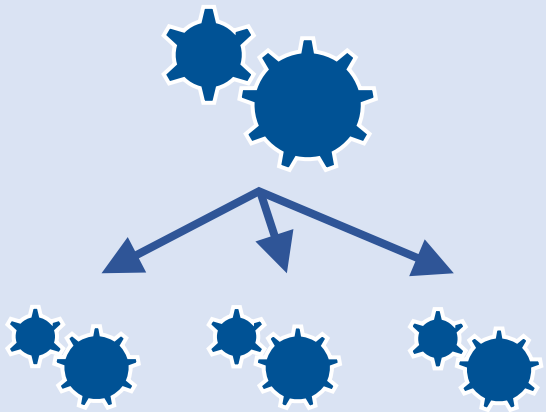
System Specification

5 System Director

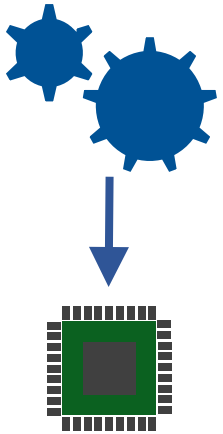
System Software

Challenges

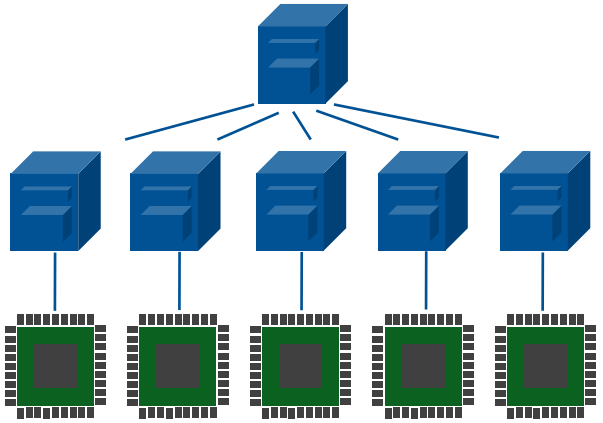
1 How to distribute ML training?



2 How to design customizable accelerators?



3 How to reduce the overhead of distributed coordination?



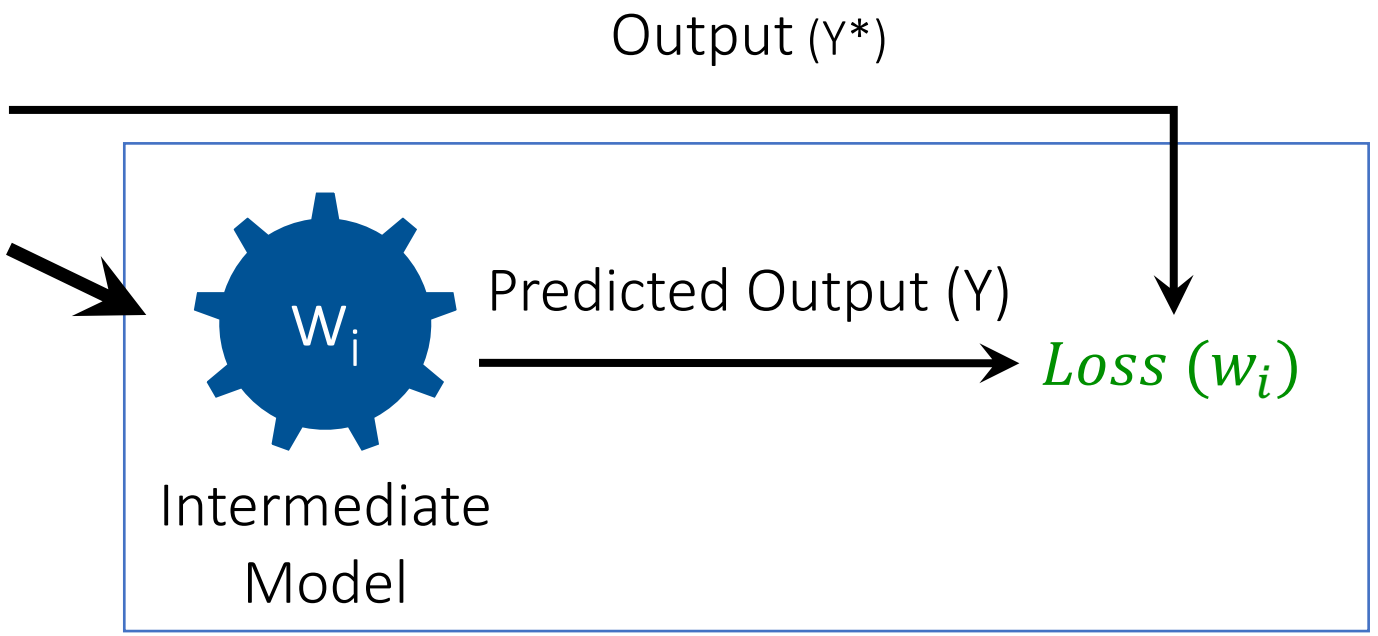
We need a full stack

❶ How to distribute?

Understanding machine learning

Training Data

Input (X)	Output (Y*)
25,1,76,0	1
6,43,9,93	6
:	:
23,56,2,0	12
12,0,9,0	0



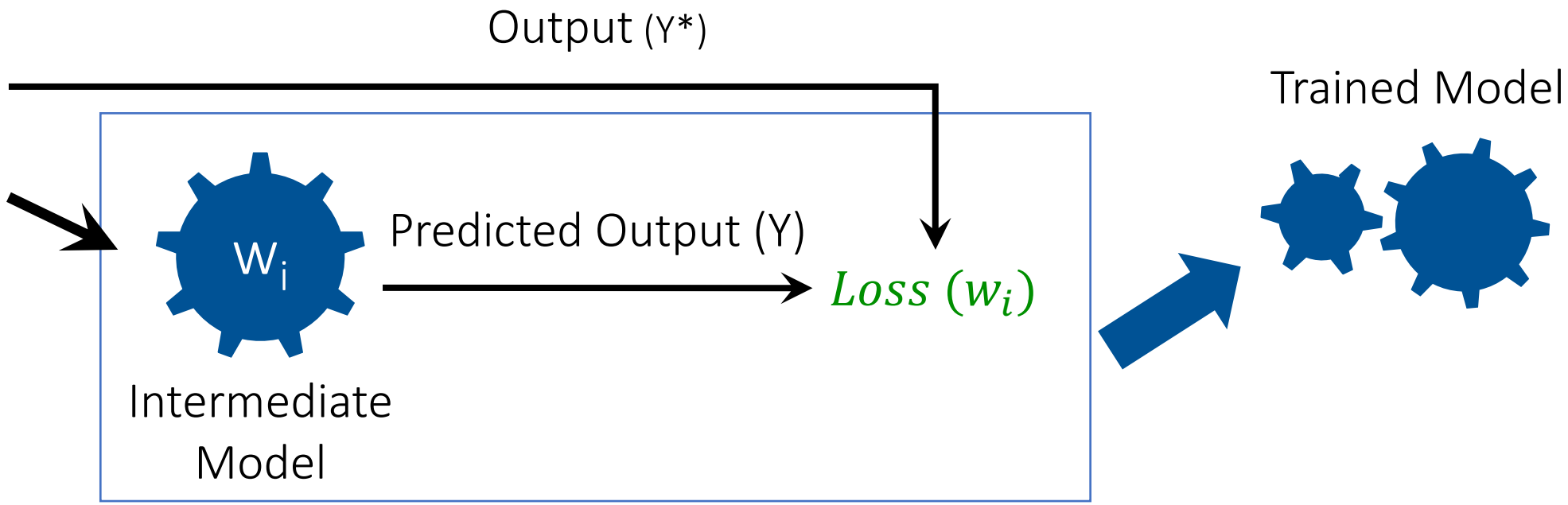
$$Loss(w_i) = \sum_i \|Y - Y^*\|$$

Algorithmic commonalities

Learning is solving an iterative optimization problem!

Training Data

Input (X)	Output (Y*)
25,1,76,0	1
6,43,9,93	6
:	:
23,56,2,0	12
12,0,9,0	0

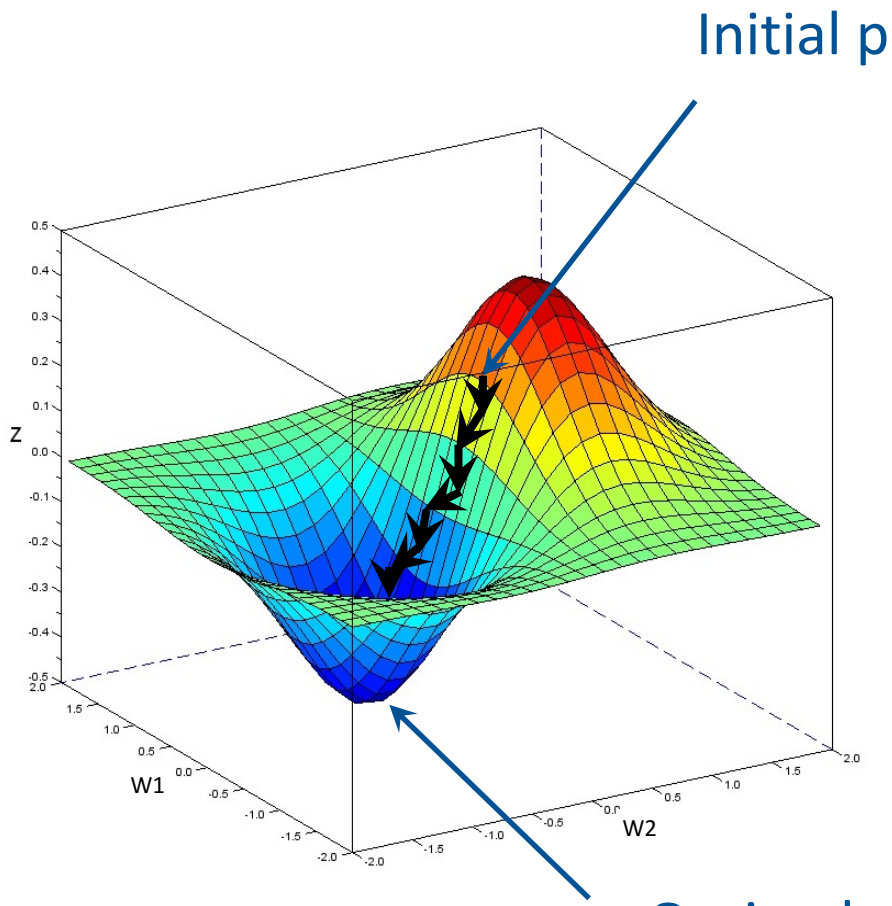


$$find(w_i) \ni \{Loss(w_i) = \sum_i \|Y - Y^*\|\}\ is\ minimized$$

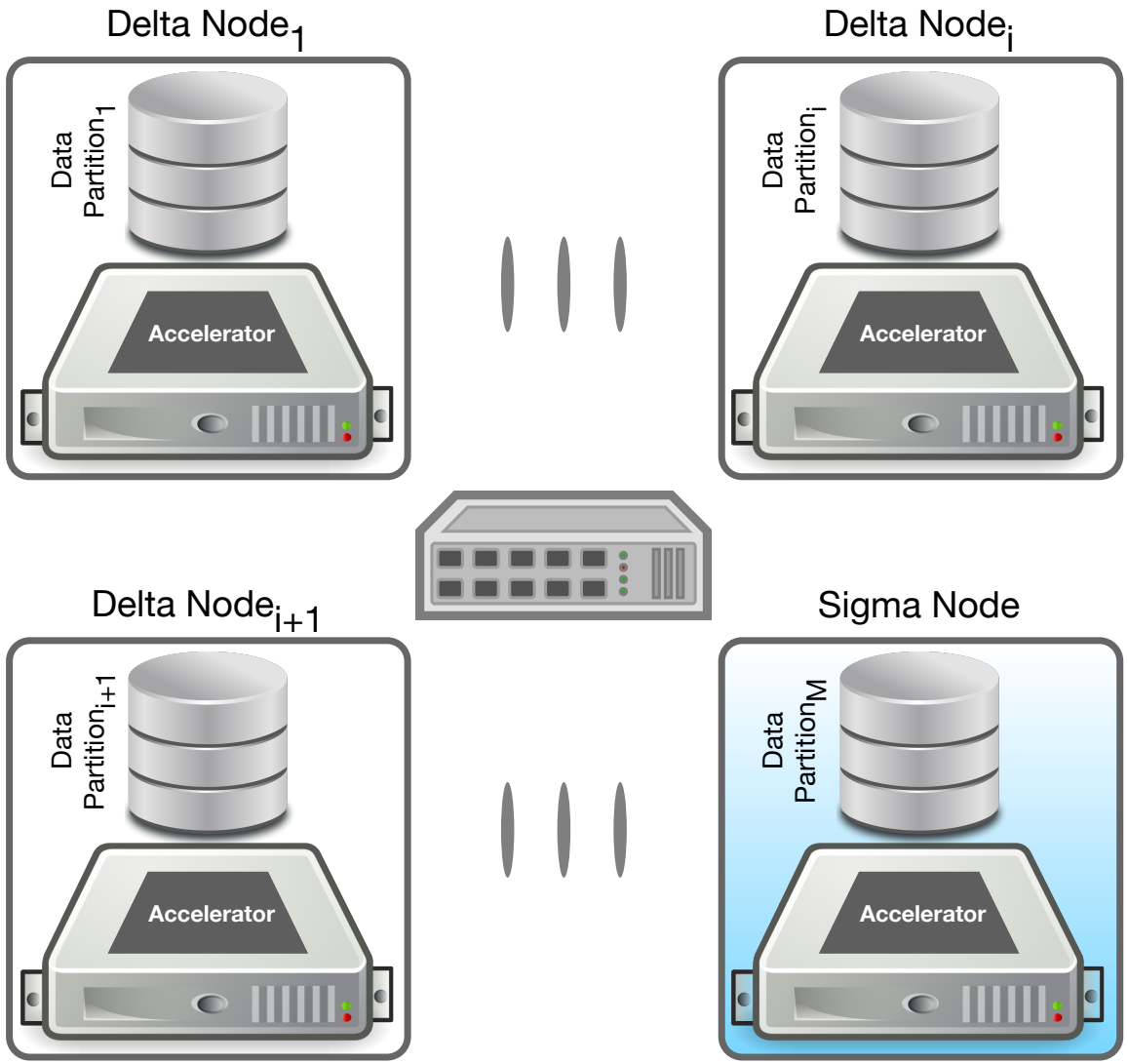
Stochastic gradient descent solver

$$Loss\ function\ (w_i) = f\left(w_1^{(t)}, w_2^{(t)}, \ldots\ldots, w_m^{(t)}\right)$$

$$w_i^{(t+1)} = w_i^{(t)} - u \times \frac{\partial f\left(w_1^{(t)}, w_2^{(t)}, \ldots\ldots, w_m^{(t)}\right)}{\partial w_i^{(t)}}$$



Leverage linearity of differentiation for distributed learning



$$\Delta W_1^{(t+1)} = \frac{\delta f}{\delta W^{(t)}}$$

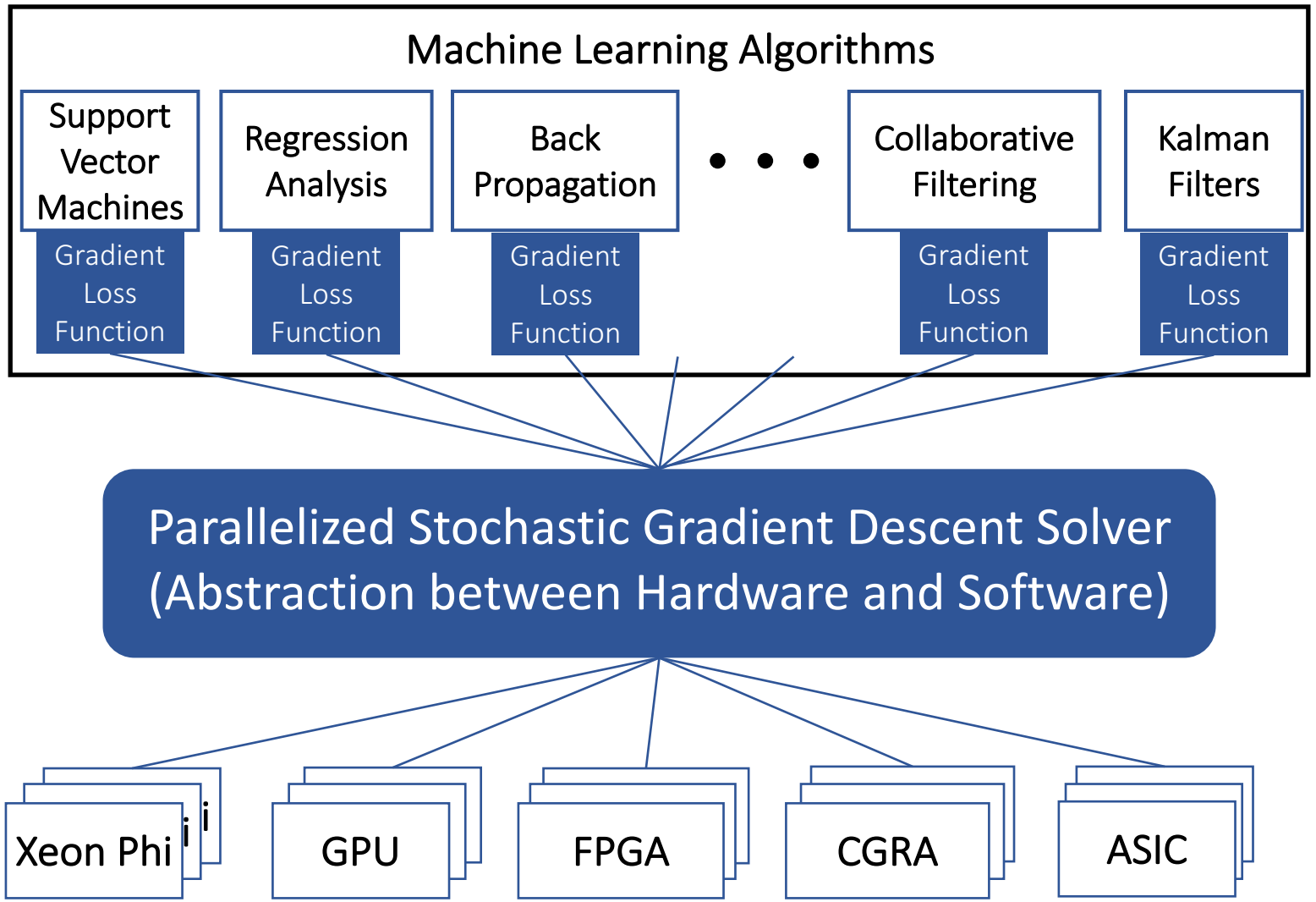
$$\Delta W_i^{(t+1)} = \frac{\delta f}{\delta W^{(t)}}$$

$$\Delta W_{i+1}^{(t+1)} = \frac{\delta f}{\delta W^{(t)}}$$

$$\Delta W_M^{(t+1)} = \frac{\delta f}{\delta W^{(t)}}$$

$$\Delta W^{(t+1)} = \frac{\sum_i \Delta W_i^{(t+1)}}{M}$$

Abstraction between algorithm and scale-out acceleration system



CoSMIC programming

Math formulations

$$\frac{\partial loss}{\partial w_i} = (\sum_i w_i x_i - y) x_i$$

$$\Delta\, w^{(t+1)} = \frac{\sum_j \Delta w_j^{(t+1)}}{n}$$

CoSMIC programming

```
h = sum[i](w[i] * x[i]);
d = h – y
g[i] = d * x[i]
```

```
aggregator(n) {
        w[i] = (sum[j](w[i])) / n;
}
```

CoSMIC compilation

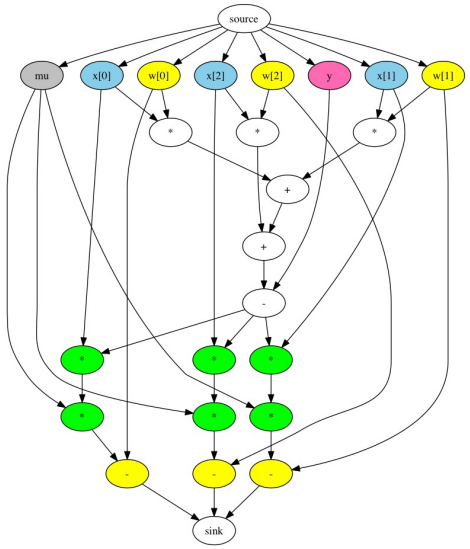
CoSMIC programming

```
h = sum[i](w[i] * x[i]);
d = h – y
g[i] = d * x[i]
```

```
aggregator(n) {
        w[i] = (sum[j](w[i])) / n;
}
```

CoSMIC compilation

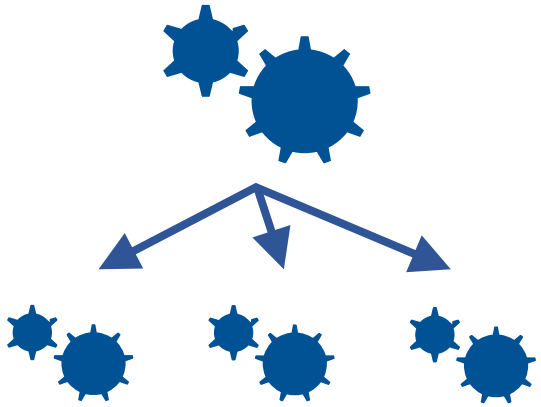
Dataflow graph



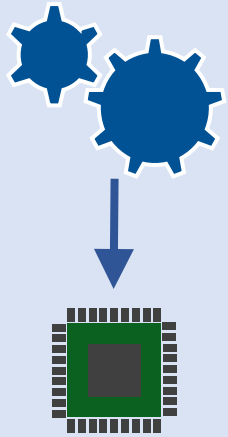
Software aggregator

Challenges

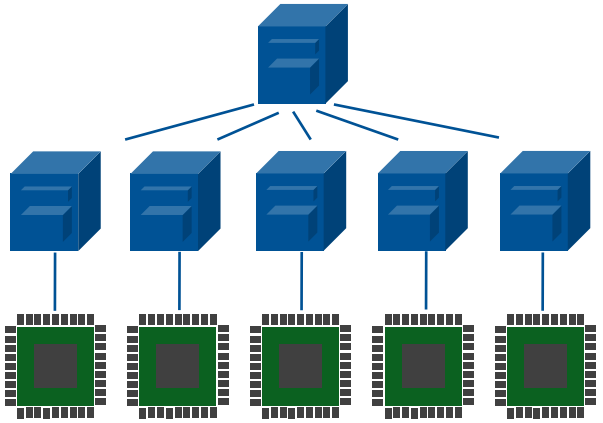
1. How to distribute ML training?



2. How to design customizable accelerators?



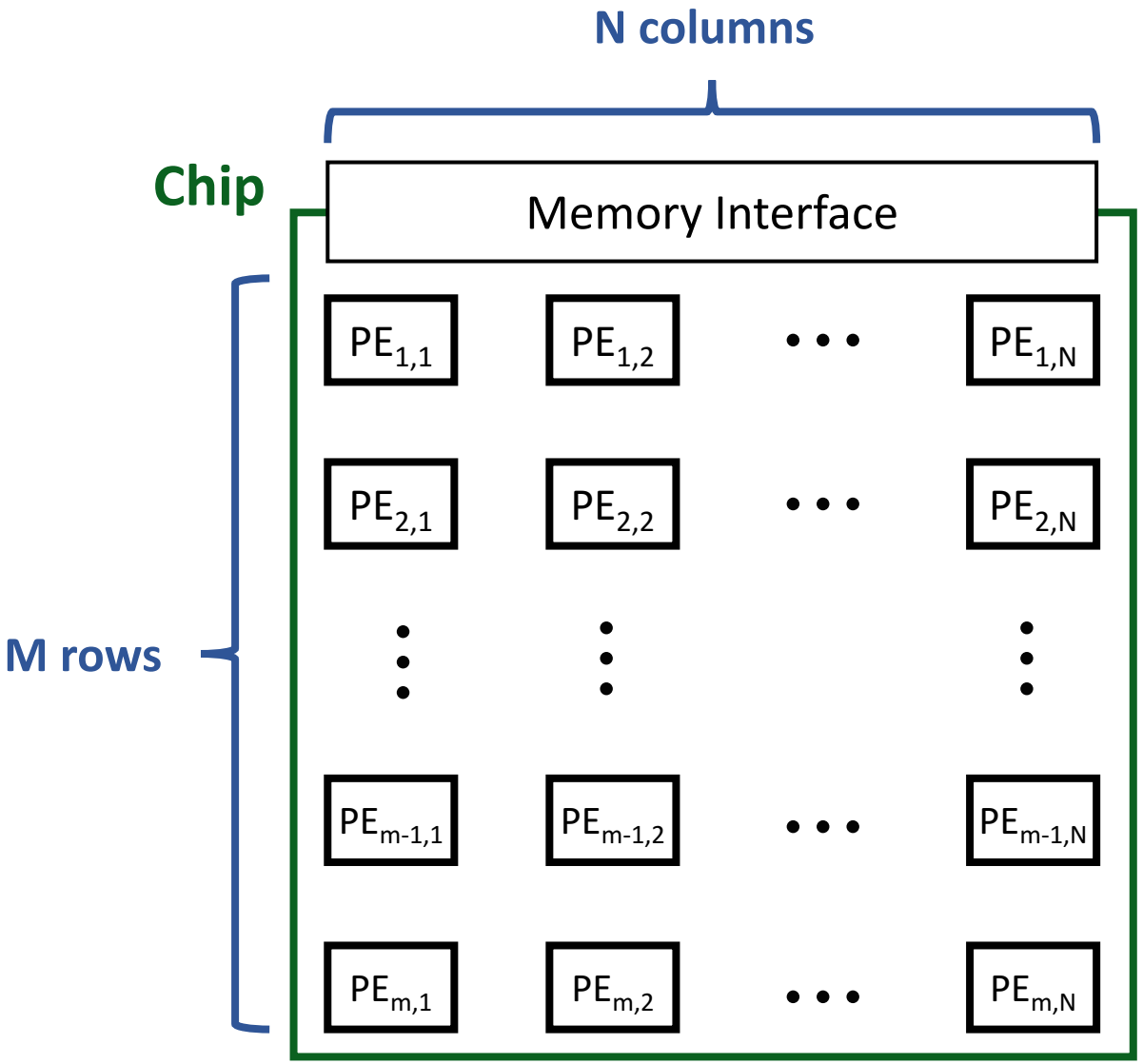
3. How to reduce the overhead of distributed coordination?



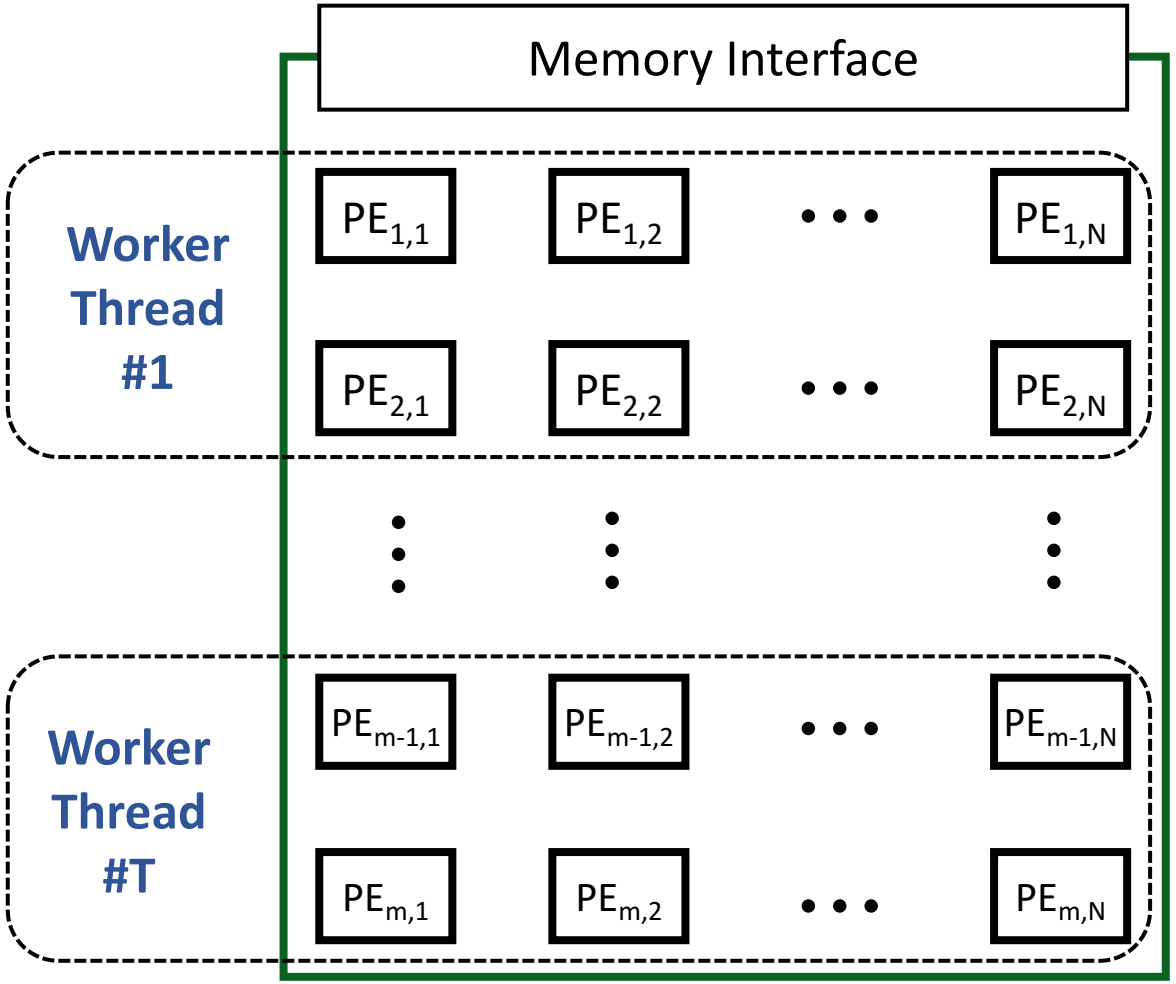
We need a full stack

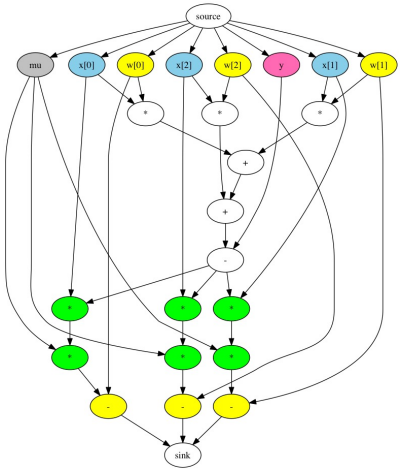
❷ How to design customizable accelerator?

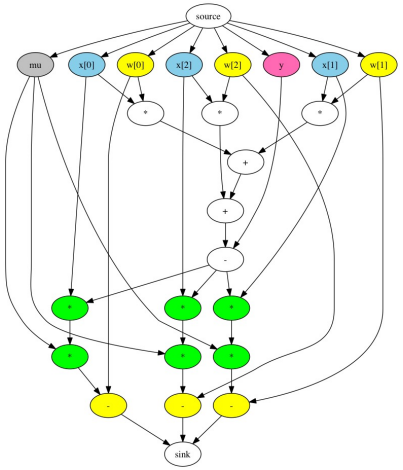
Template architecture



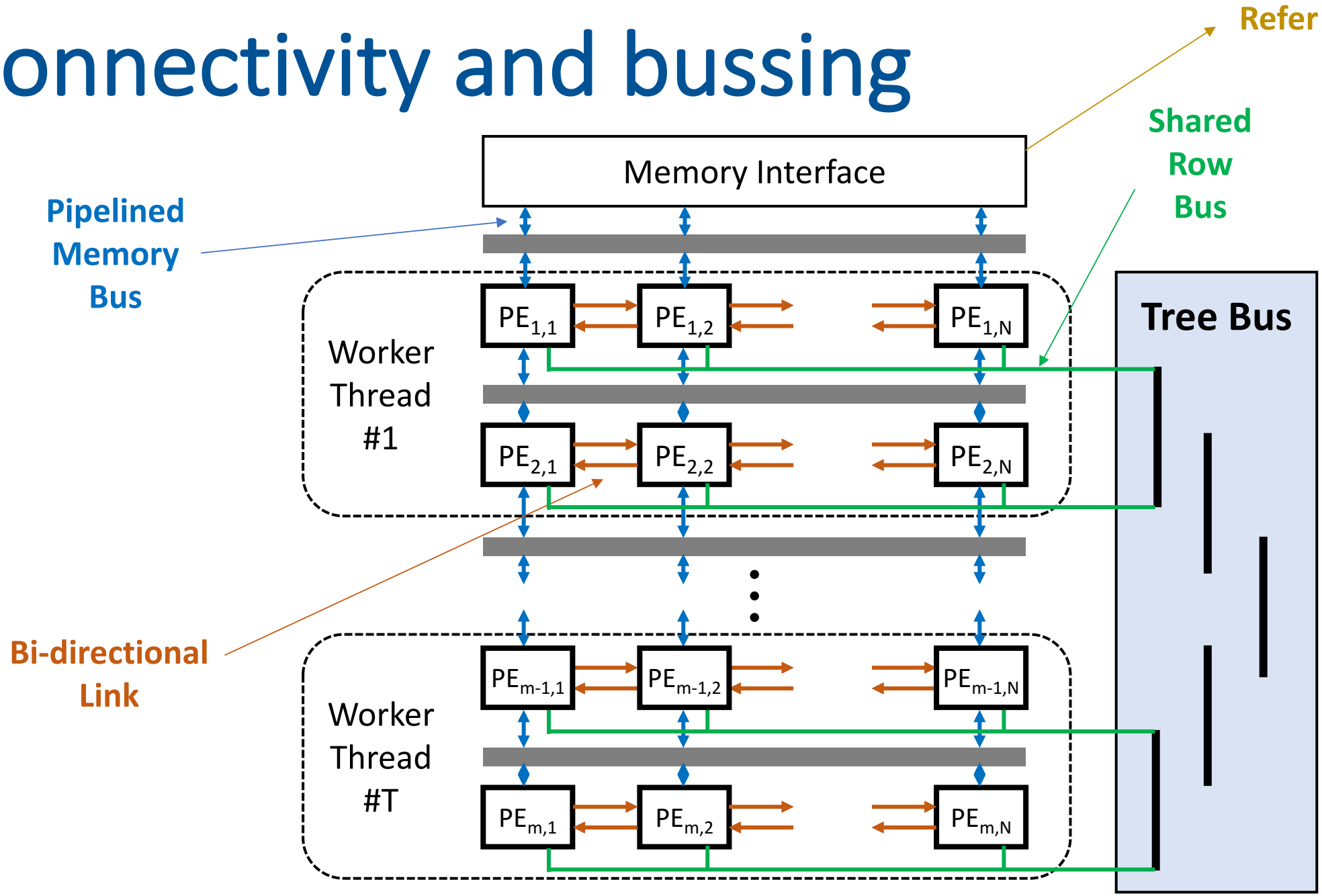
Multi-threading acceleration



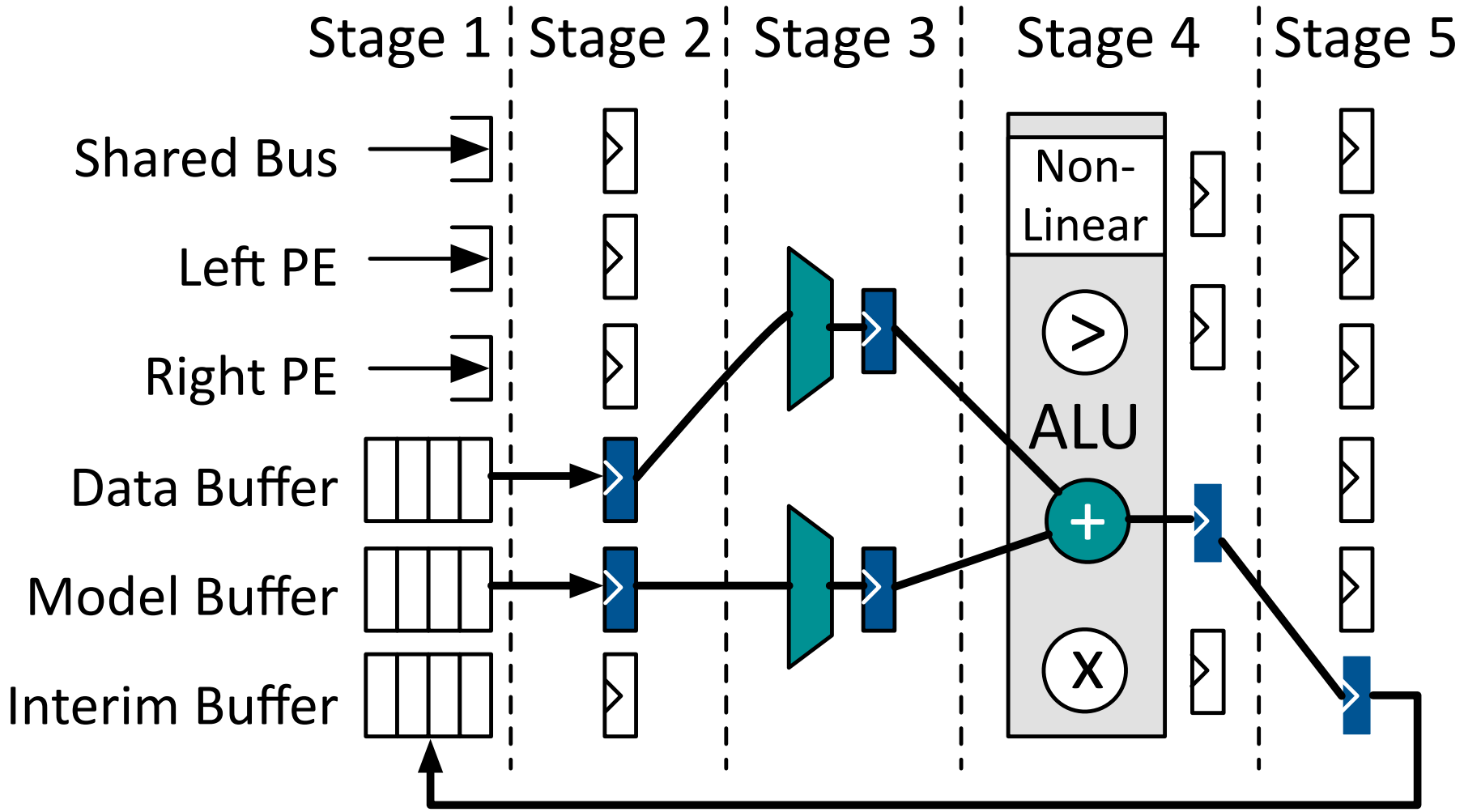




Connectivity and bussing

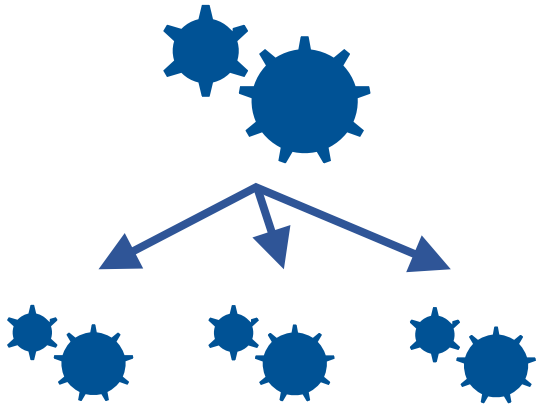


PE architecture

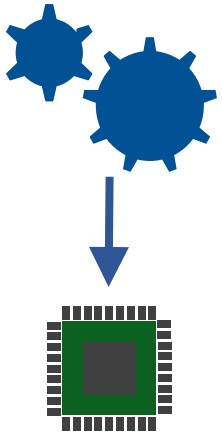


Challenges

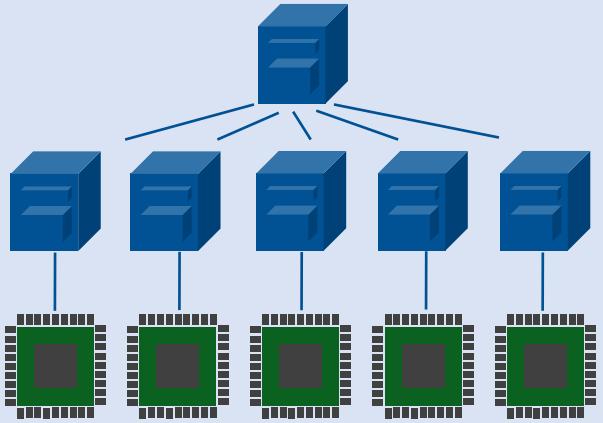
1. How to distribute ML training?



2. How to design customizable accelerators?



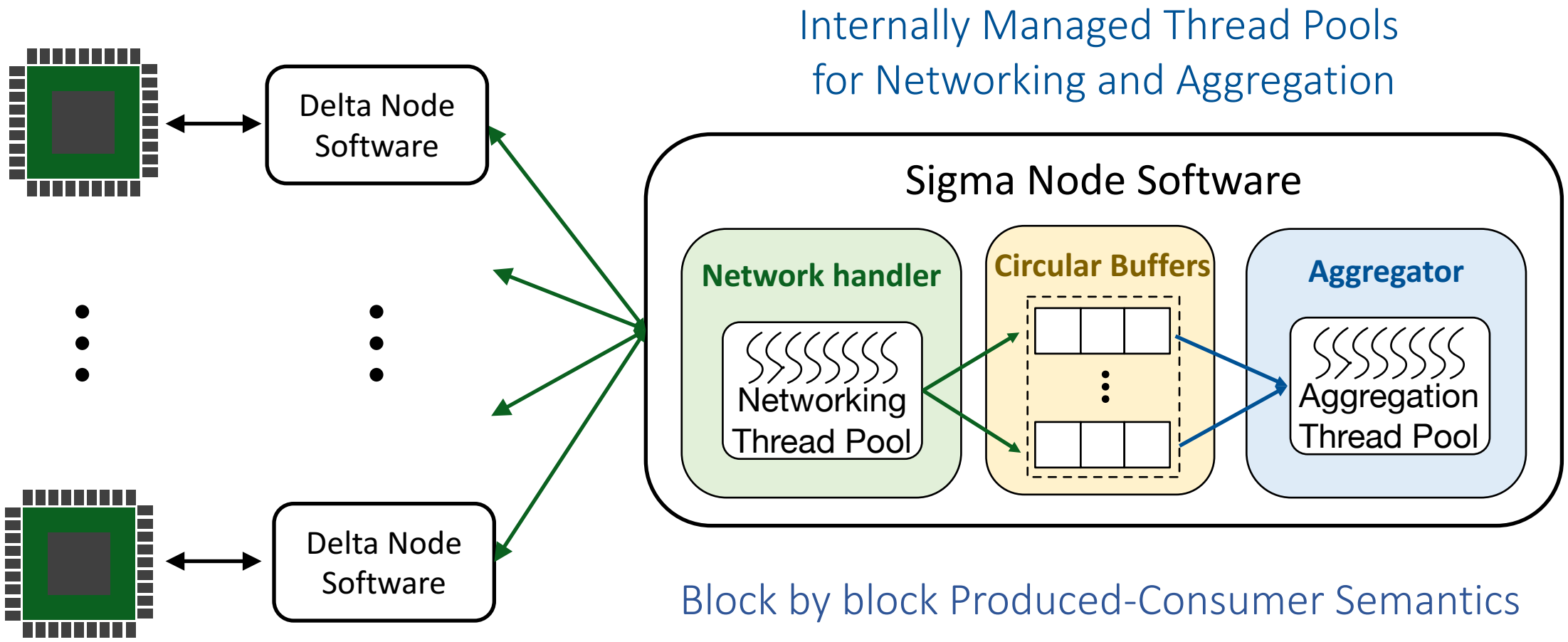
3. How to reduce the overhead of distributed coordination?



We need a full stack

❸ How to reduce overhead of distributed coordination?

Specialized system software in CoSMIC



Benchmarks for distributed learning

Name	Description	Algorithm	Model Size (KB)	Training Data Size (GB)	Lines of Code
mnist	Handwritten digit recognition	Backpropagation	2,432 KB	2.9 GB	55
acoustic	Speech recognition modeling		1,527 KB	5.6 GB	55
stock	Stock price prediction	Linear Regression	31 KB	14.7 GB	23
texture	Image texture recognition		64 KB	17.9 GB	23
tumor	Tumor classification	Logistic Regression	8 KB	10.4 GB	22
cancer1	Prostate cancer diagnosis		24 KB	13.5 GB	22
movielens	Movielens recommender system	Collaborative Filtering	1,176 KB	0.6 GB	42
netflix	Netflix recommender system		2,854 KB	2.0 GB	42
face	Human face detection	Support Vector Machine	7 KB	15.9 GB	27
cancer2	Cancer diagnosis		28 KB	20.0 GB	27

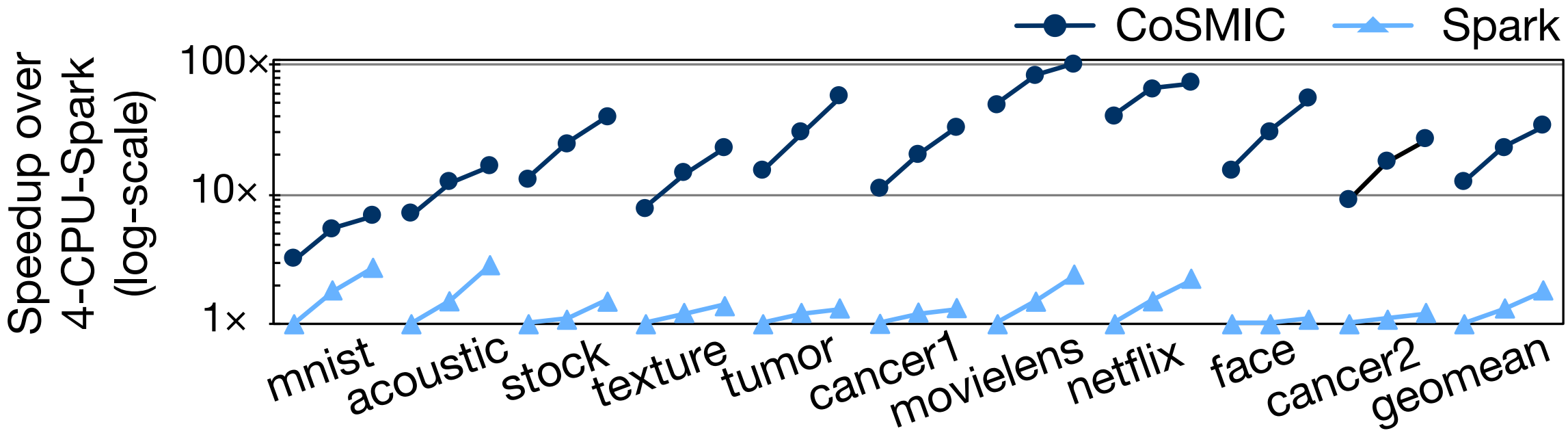
List of results

1. Performance comparison with Spark
2. Breakdown of performance between computation and system coordination
3. Performance comparison of FPGA-CoSMIC with Programmable ASIC and GPUs
4. Power Efficiency comparison between FPGA, Programmable ASICs and GPUs
5. Sensitivity to mini-batch size, compute units, memory bandwidth
6. Design space exploration for multithreading
7. Comparison of template architecture with prior accelerator designs

List of results

1. Performance comparison with Spark
2. Breakdown of performance between computation and system coordination
3. Performance comparison of FPGA-CoSMIC with Programmable ASIC and GPUs
4. Power Efficiency comparison between FPGA, Programmable ASICs and GPUs
5. Sensitivity to mini-batch size, compute units, memory bandwidth
6. Design space exploration for multithreading
7. Comparison of template architecture with prior accelerator designs

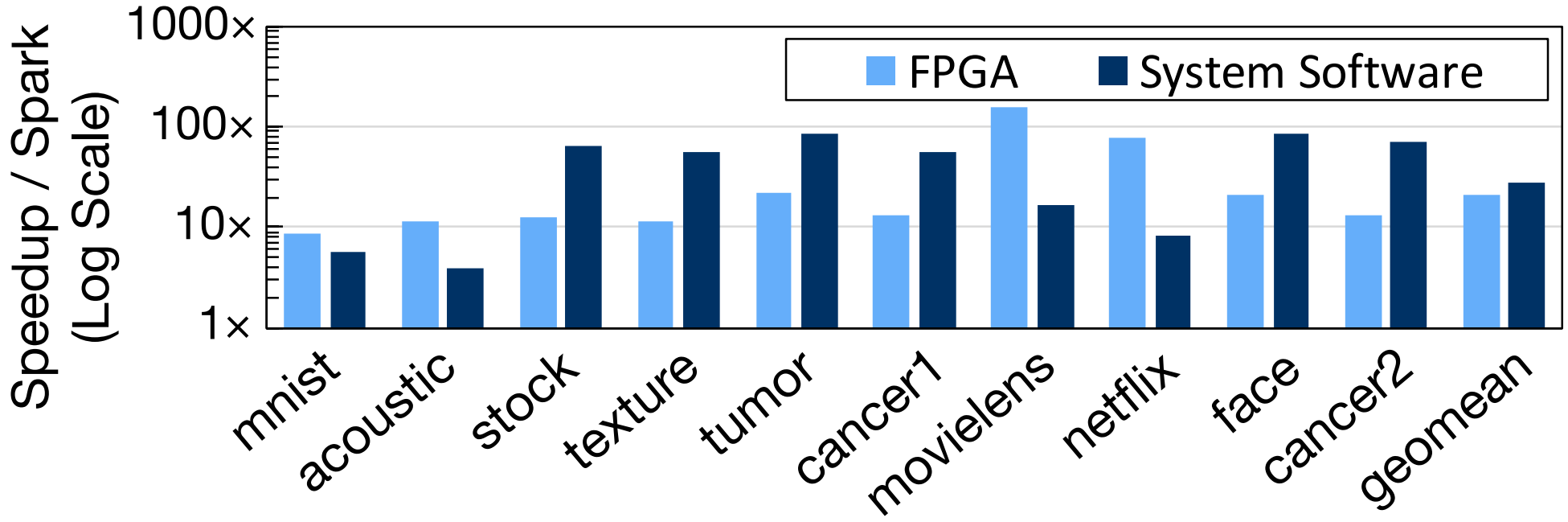
Speedup in comparison with Spark



16-node CoSMIC with UltraScale+ FPGAs offer 18.8× speedup over 16-node Spark with Xeon E3 Skylake CPUs

Scaling from 4 to 16 nodes with CoSMIC yields 2.7× improvement while Spark offers 1.8×.

Speedup breakdown between computation and system coordination

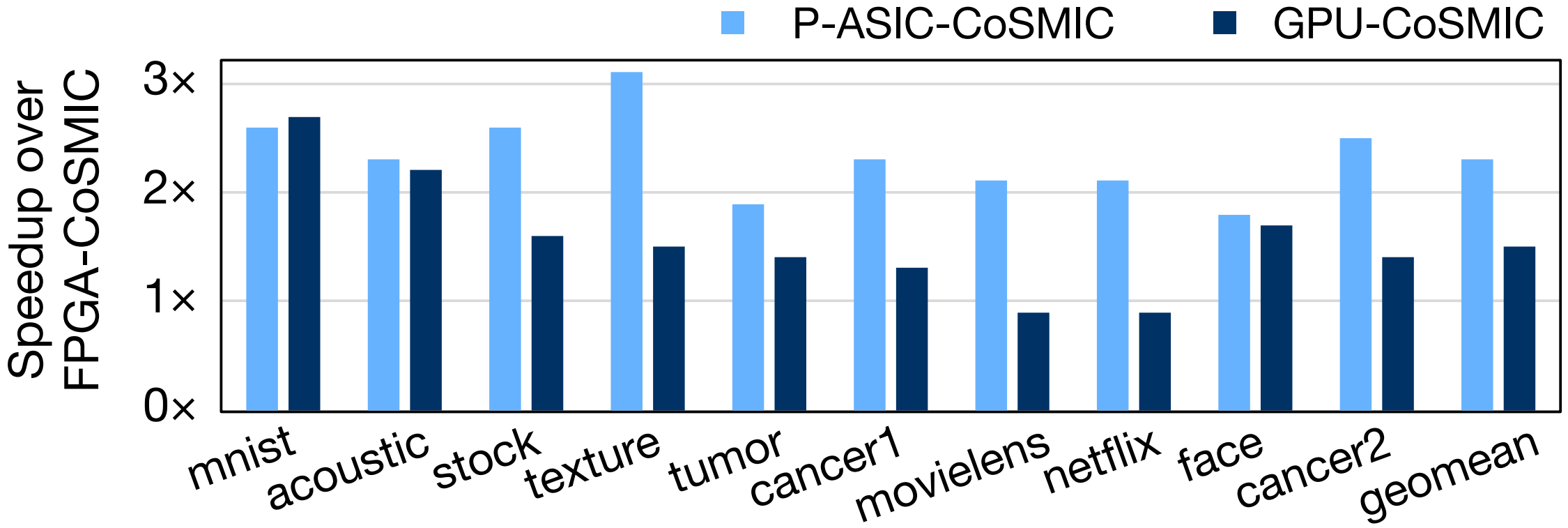


CoSMIC system software makes system coordination (34%) 28.4× faster

CoSMIC FPGA hardware makes computation (66%) 20.7× faster

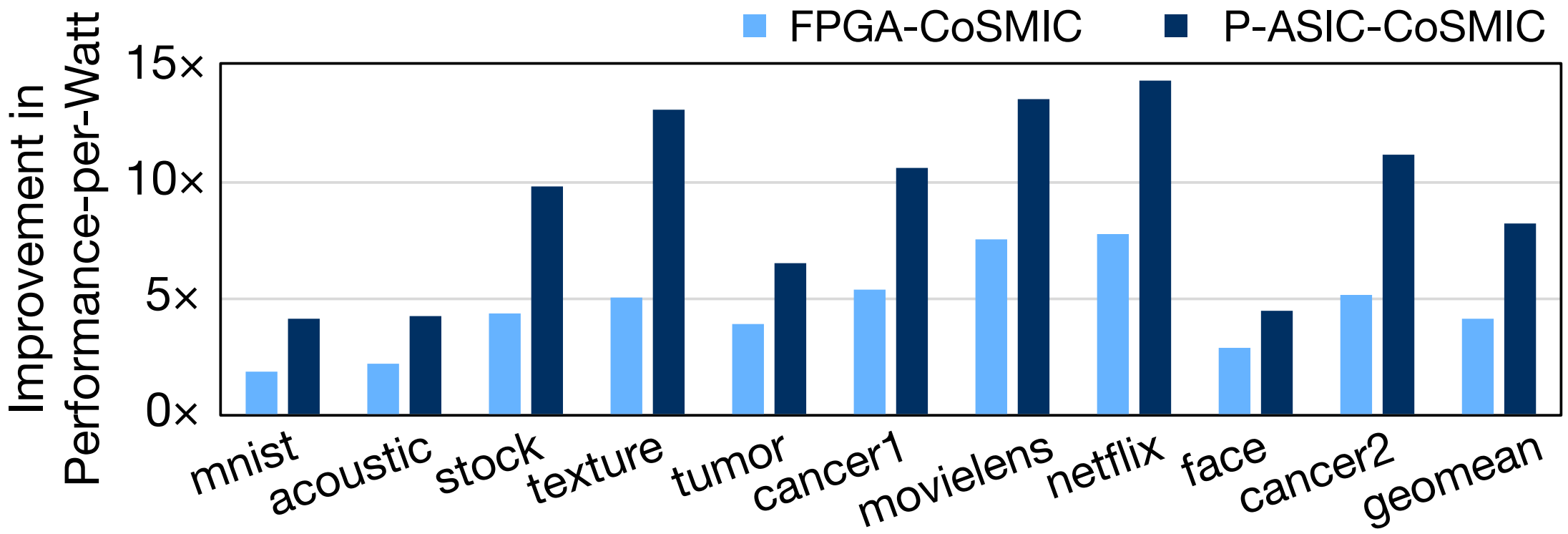
The overall speedup is 22.8× speedup

Hardware is not enough, we need a novel system stack



P-ASIC, and GPU provide **2.3× and 1.5× extra speedup** over FPGA CoSMIC, which is **22.8×** faster than Spark

Performance-per-Watt comparison



With CoSMIC FPGAs and P-ASICs provide 4.2× and 8.2× higher Performance-per-Watt than GPUs

Conclusion

CoSMIC

Full-stack solution with an algorithmic angle

What is next

Reduce communication

Specialize the networking stack

Design more template architecture