TECHNISCHE HOCHSCHULE DEGGENDORF

Datacenter Network Programming

In-Network Support for Distributed ML

Slides inspired by

SwitchML
Scaling Distributed Machine Learning with In-Network Aggregation

Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan Ports, Peter Richtarik

Do Switches Dream of Machine Learning?
And other In-Network Computing Stories

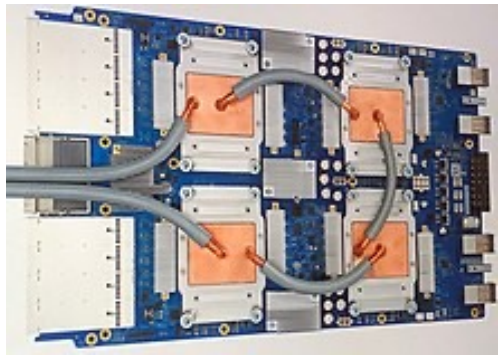
Noa Zilberman
noa.zilberman@eng.ox.ac.uk

Joint work with Changgang Zheng (Oxford), Yuta Tokusashi, Zhaoqi Xiong, Thanh T. Bui, Siim Kaupmees (Cambridge), Tu Dang, Pietro Bressana, Fernando Pedone (USI), Jackson Woodruff (Edinburgh), Murali Ramanujam (UCLA), Antoine Bernabeu (École Centrale de Nantes), Shay Vargaftik , Yaniv Ben-Itzhak (VMWare), & Robert Soulé (Yale).

Machine Learning and AI acceleration

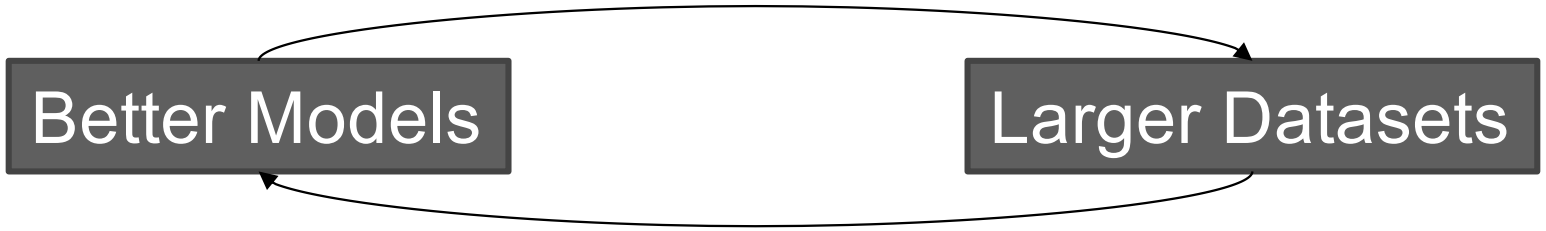
- **General purpose GPU**
  - Parallel computing on large amount of data
  - Requires large batches for efficient processing
  - Overhead to move data to/from GPU
  - GPU Good for training, Inference typically done on normal CPU

- **Tensor Processing Unit (TPU)**
  - Custom ASIC dedicated mostly for inference
    - e.g. AlphaGo → TPU inside Google
    - E.g. Integrated into new apple chipsets
  - Data transfer accounts up to 75% of processing time



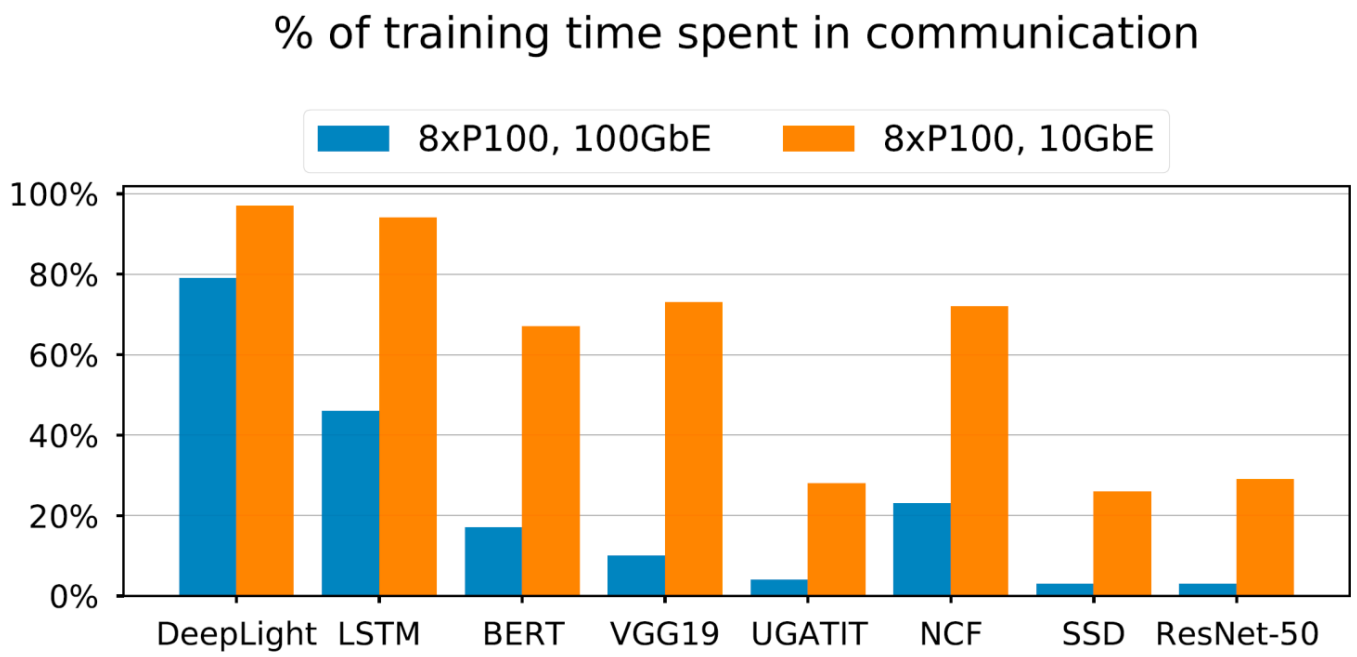
Machine Learning

- **Scalability Issues**
  - AI Models become larger → one server too slow to train
  - Datasets become larger → one server too small memory to hold data
- **Requirements**
  - Clusters with hundreds of servers
  - Each with AI accelerators
  - Reduce training time by distributed training

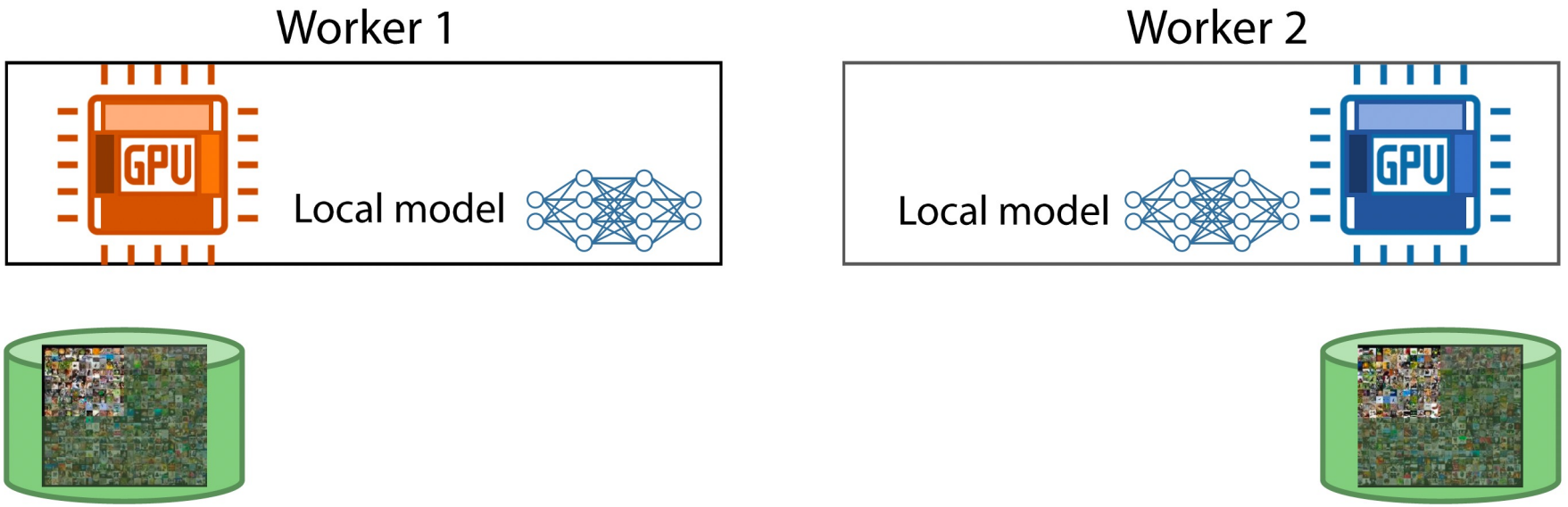


Scaling Distributed Machine Learning

- **Multiple worker nodes**
  - Network needs to scale properly for not to be the bottleneck

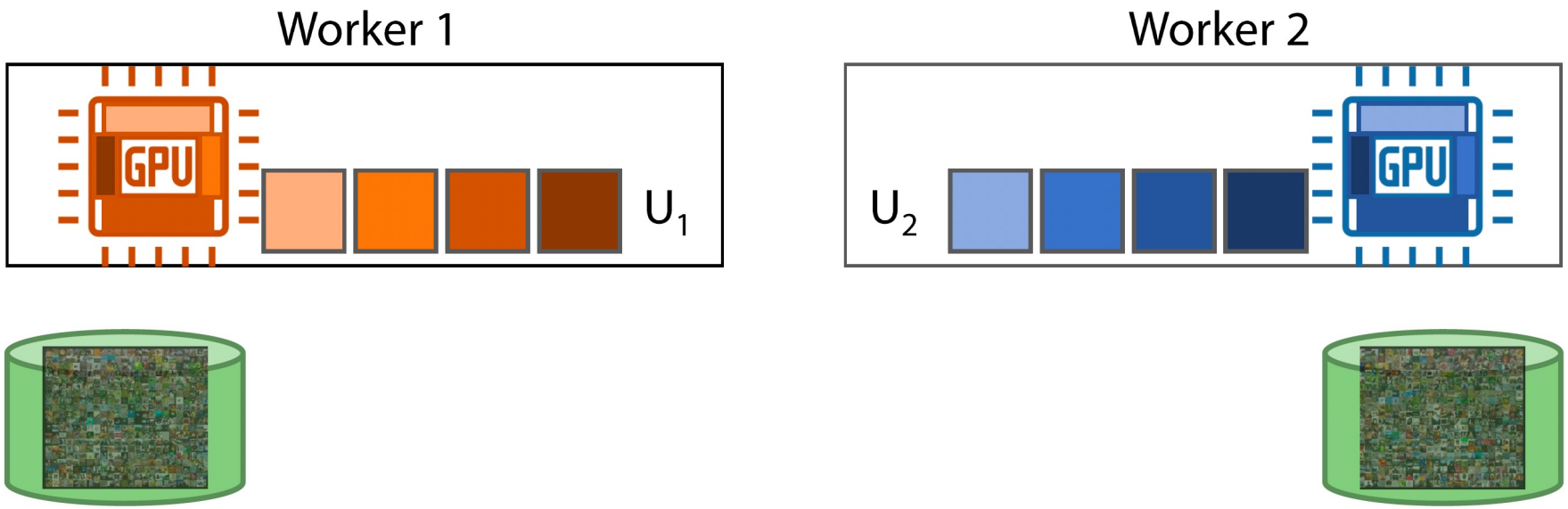


Data Parallel



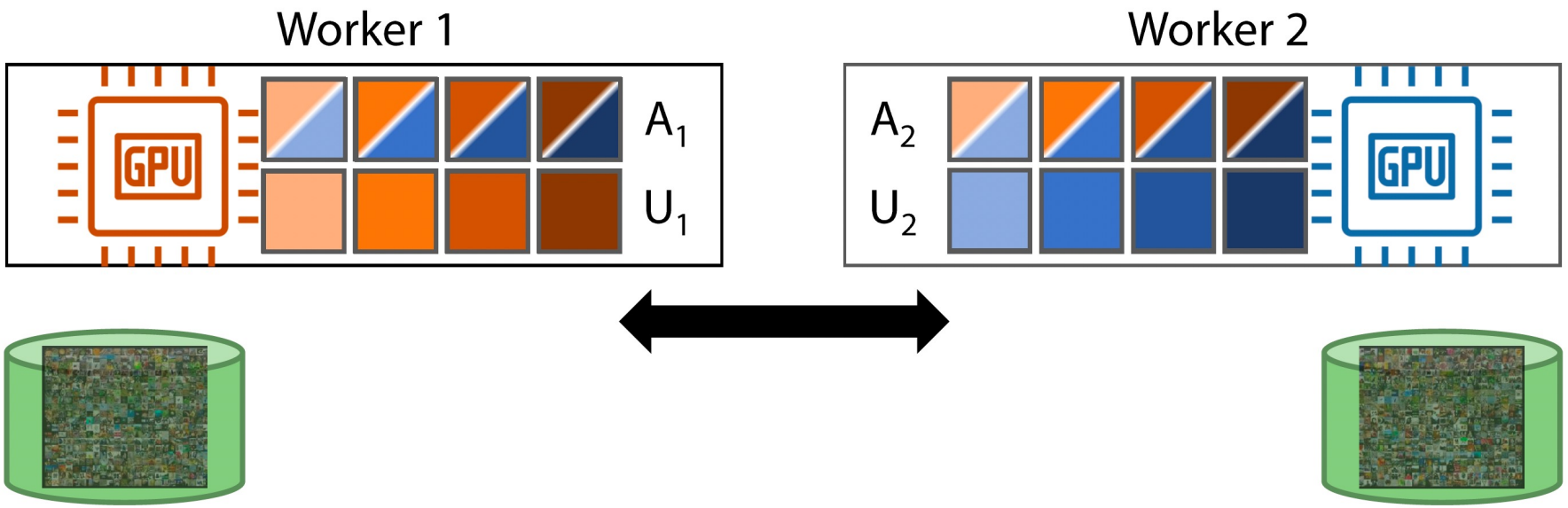
- Input data opf the training set is distributed over many machines
- Each machine trains a local model
- At each iteration, need to compute the sum of model updates across all workers. Updates are aggregated and added to form model parameters of the next iteration
- Each model update requires model parameter exchanges over the network

Data Parallel



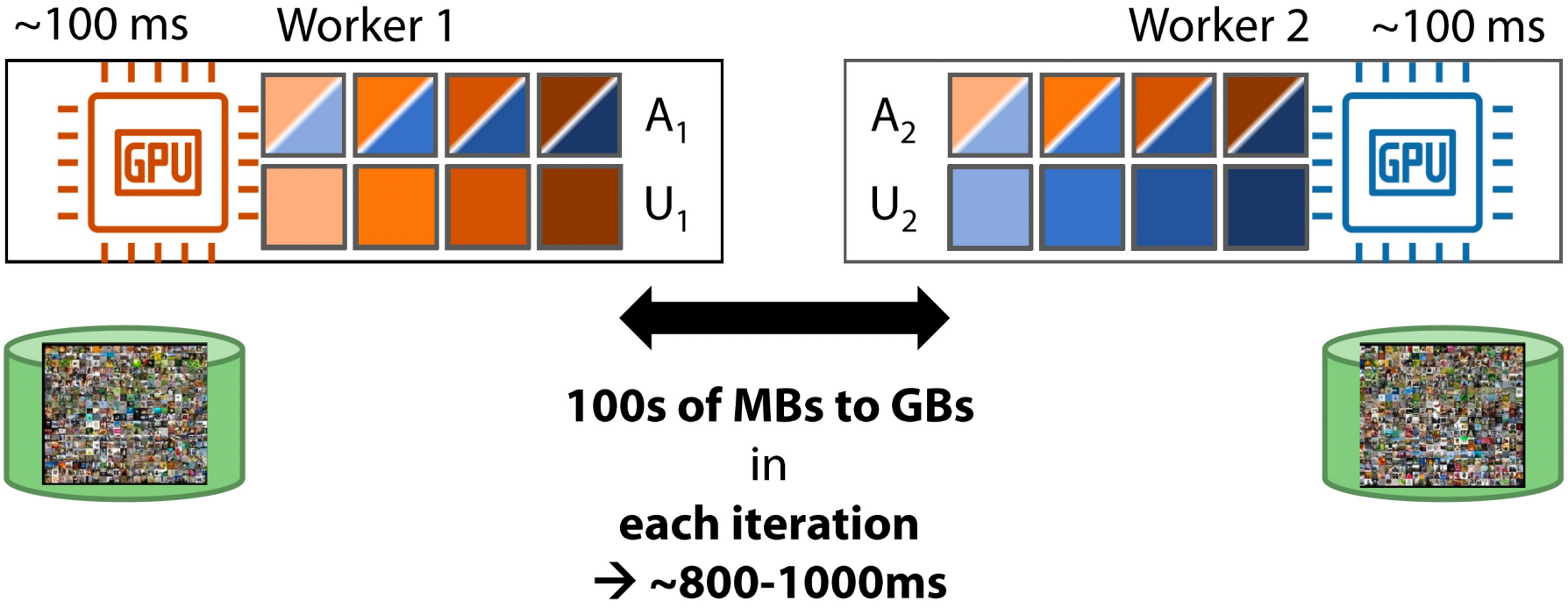
- **U_i: Model update computed by worker i**

Data Parallel



- **A_i: Aggregated Model Update of Worker i**

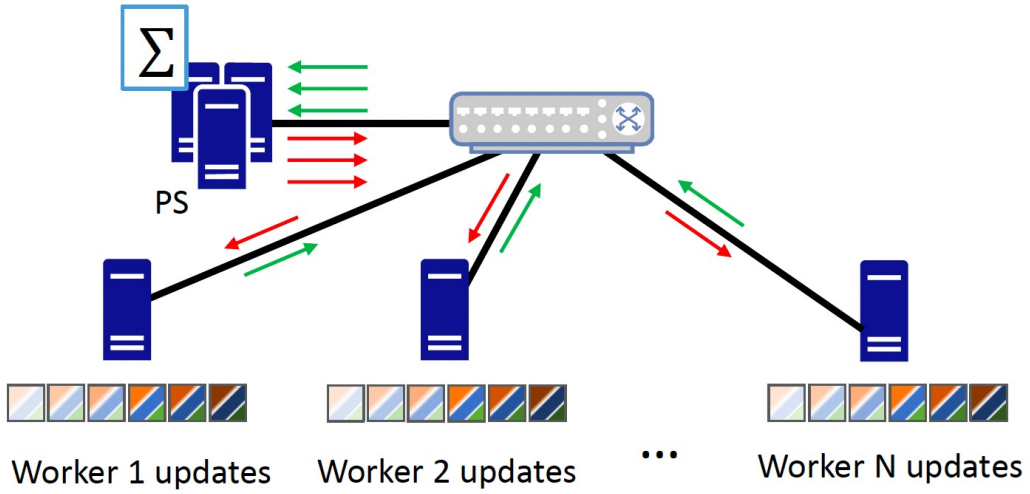
Data Parallel



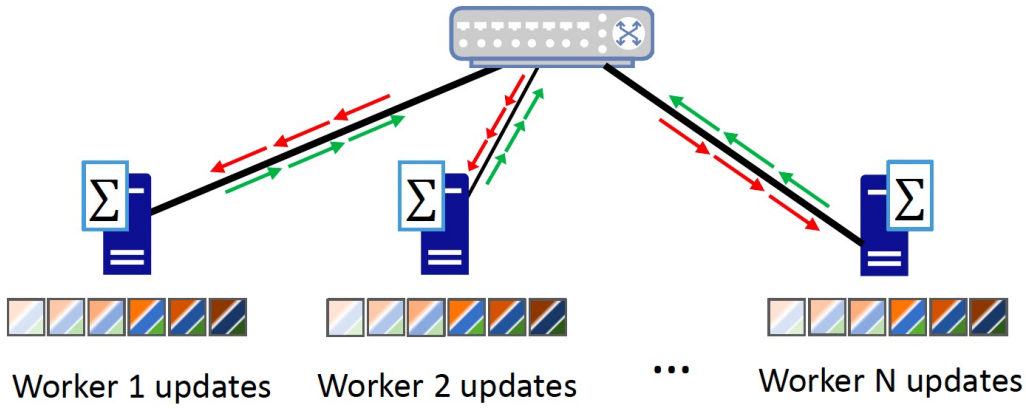
- **What is the problem?**
  - Intensive all-to-all communication → network is the bottleneck

Different approaches

- **Parameter Server (PS)**
  - Worker compute model update
  - Send to PS, which aggregates and redistributes
  - Model is sharded across multiple PS nodes

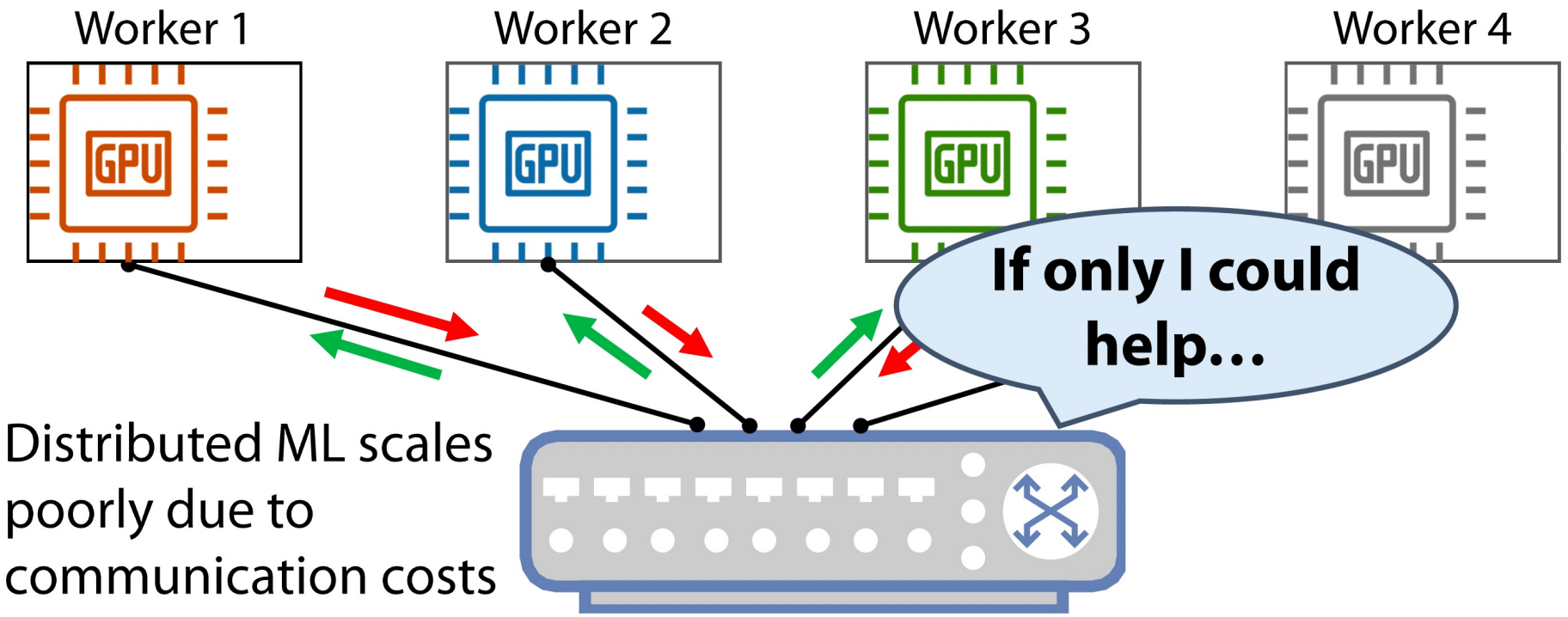


- **All-Reduce (Ring)**
  - Workers communicate over overlay network
  - Each worker communicates to next neighboring worker forming a ring
  - Latency grows with ring size



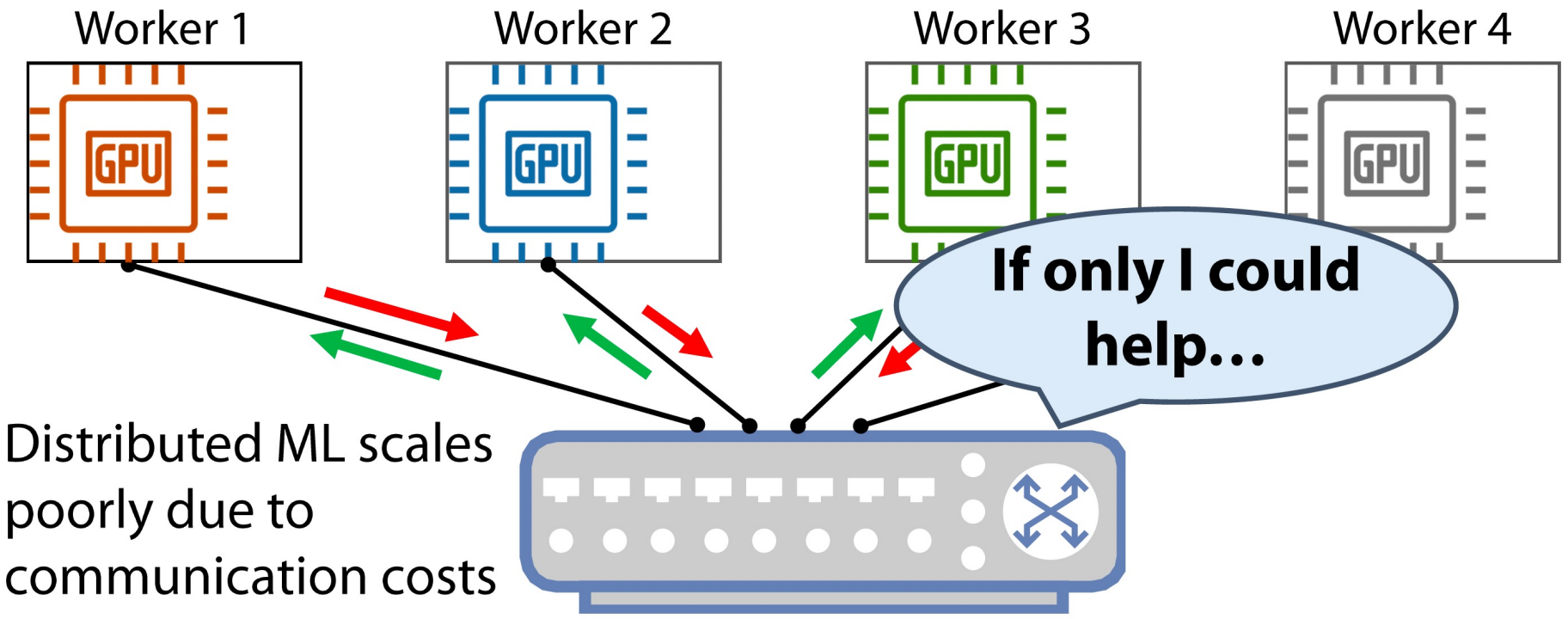
How to keep models synchronized?

- **How to reduce communication costs with P4 programmable switch?**



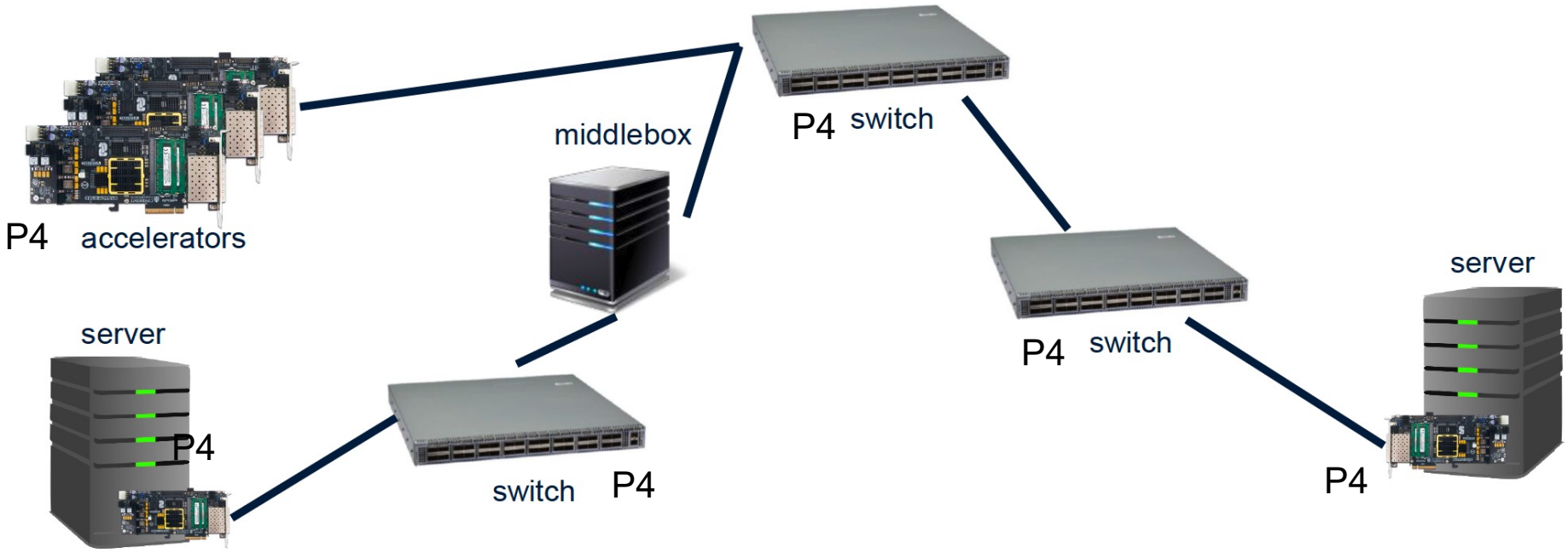
How to keep models synchronized?

- **How to reduce communication costs?**
  → Aggregate model updates in-network



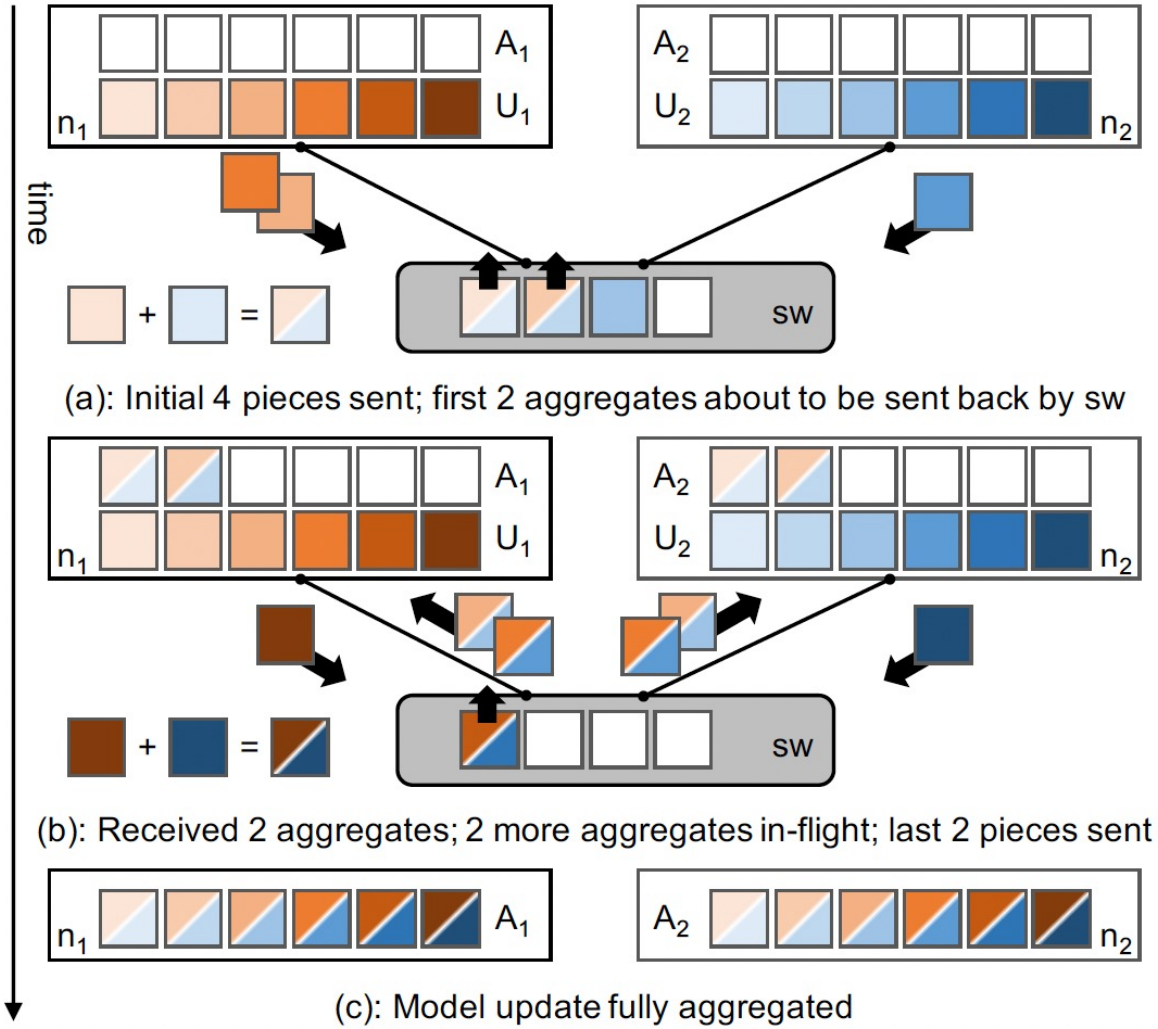
In-Network Computing?

- **What is it? Execution of native host applications within the network using standard network devices**
  - x10 latency, x10000 throughput, x1000 power efficiency



In-network aggregation

- Workers stream pieces of model updates in coordinated fashion
- E.g. each worker can have at most 4 outstanding packets to match slots in switch
- Switch aggregates updates and multicasts back values, which are collected into aggregated model upodates

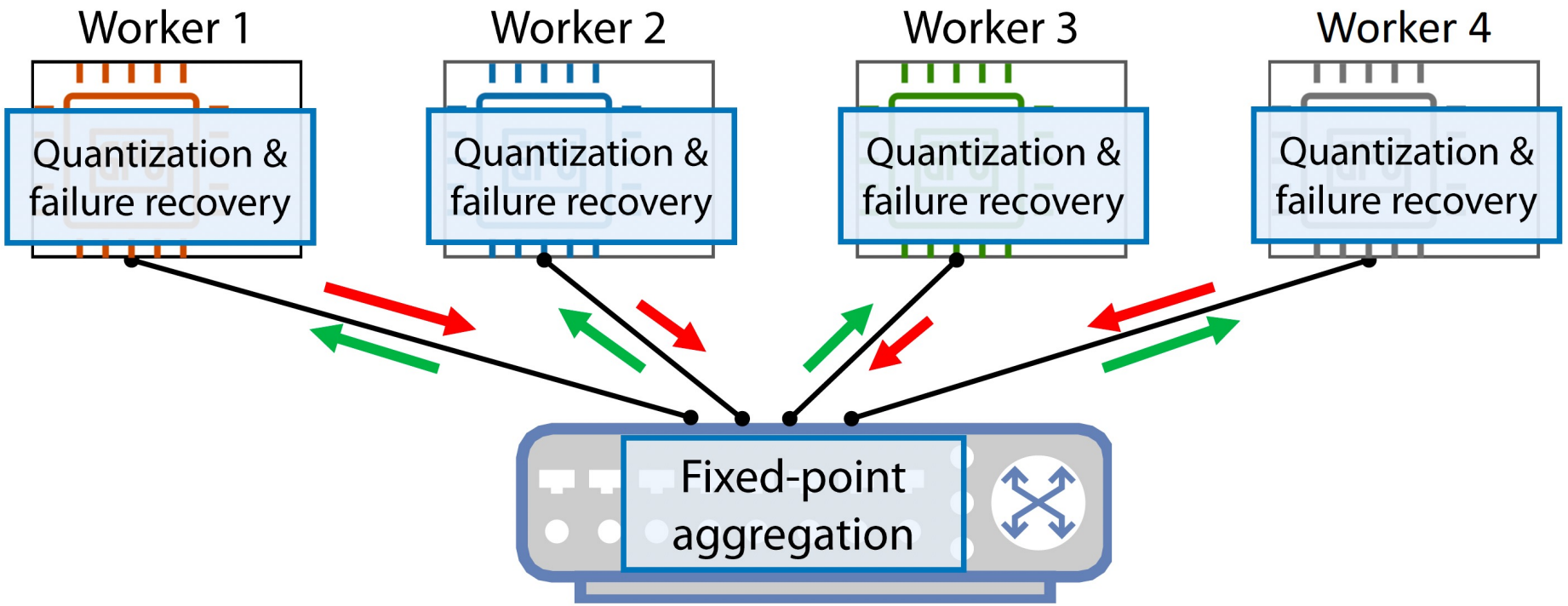


SwitchML

- **Challenges**
  - No floats (but ML-model needs floats??)
  - Limited storage
  - Limited computation
  - Packet loss

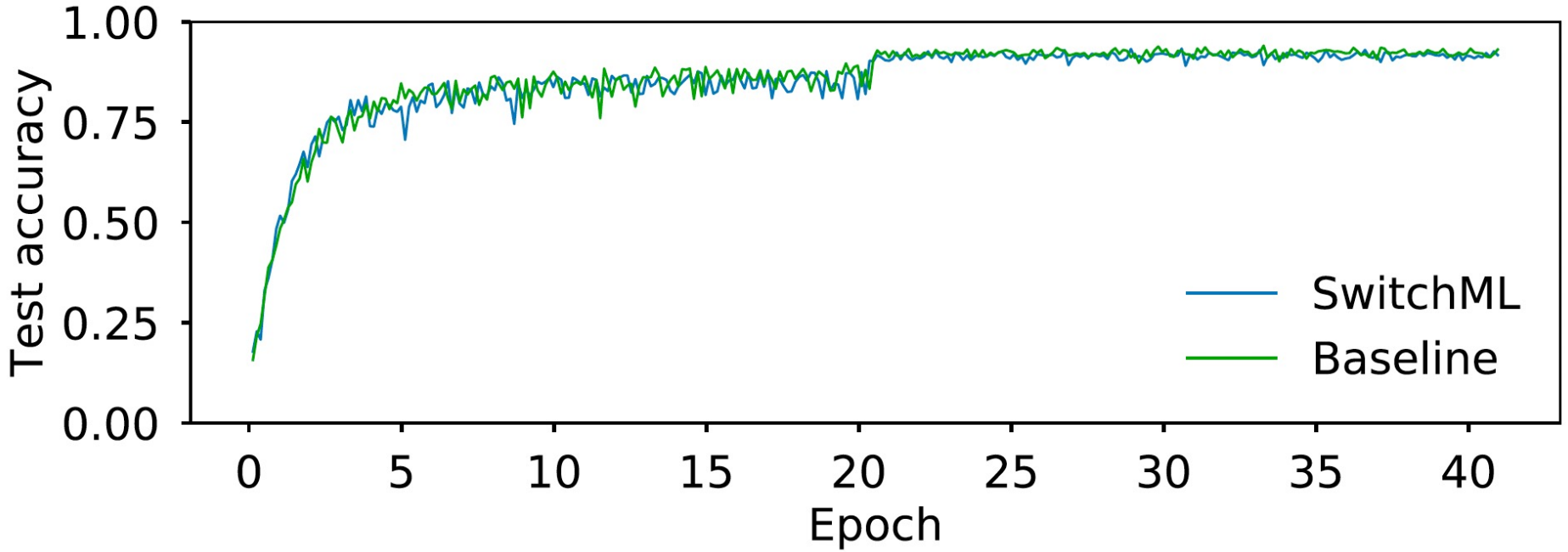
- **Design**
  - Combined switch-host architecture
  - Pool-based streaming aggregation
  - Integer quantization
  - Failure recovery protocol
  - In-switch RDMA implementation → This is nice!

Combined switch-host architecture



Quantization

- **Quantization allows training to similar accuracy in a similar number of iterations as an unquantized network**



Implementation

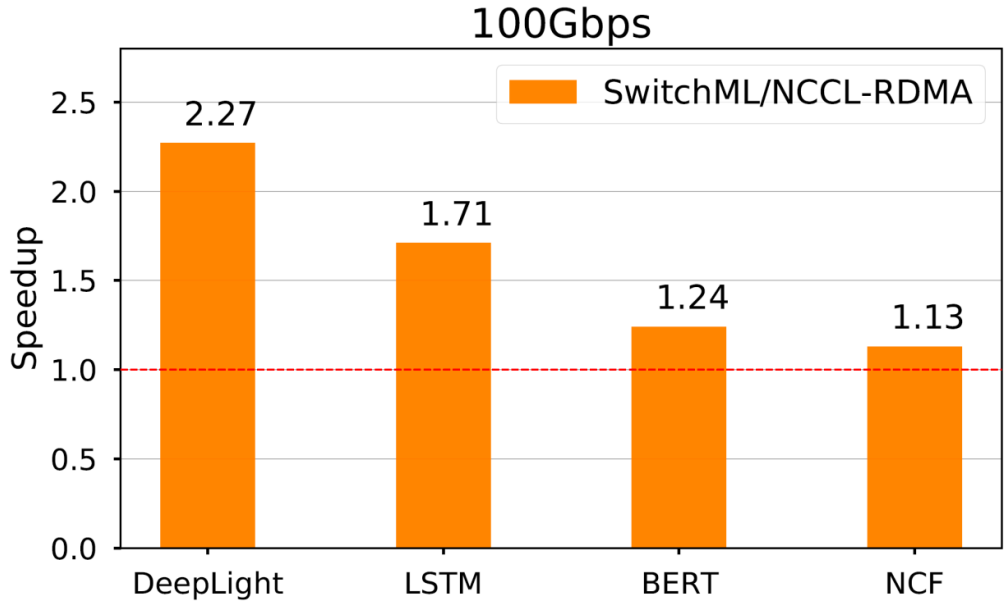
- Switch P4 Program for Intel Tofino
- End-host C++ library providing all-reduce API functionality
- Integration with PyTorch, TensorFlow and Horovod

- Github.com/p4lang/p4app-switchML

- Standard ML benchmarks
- Microbenchmarks for aggregation testing

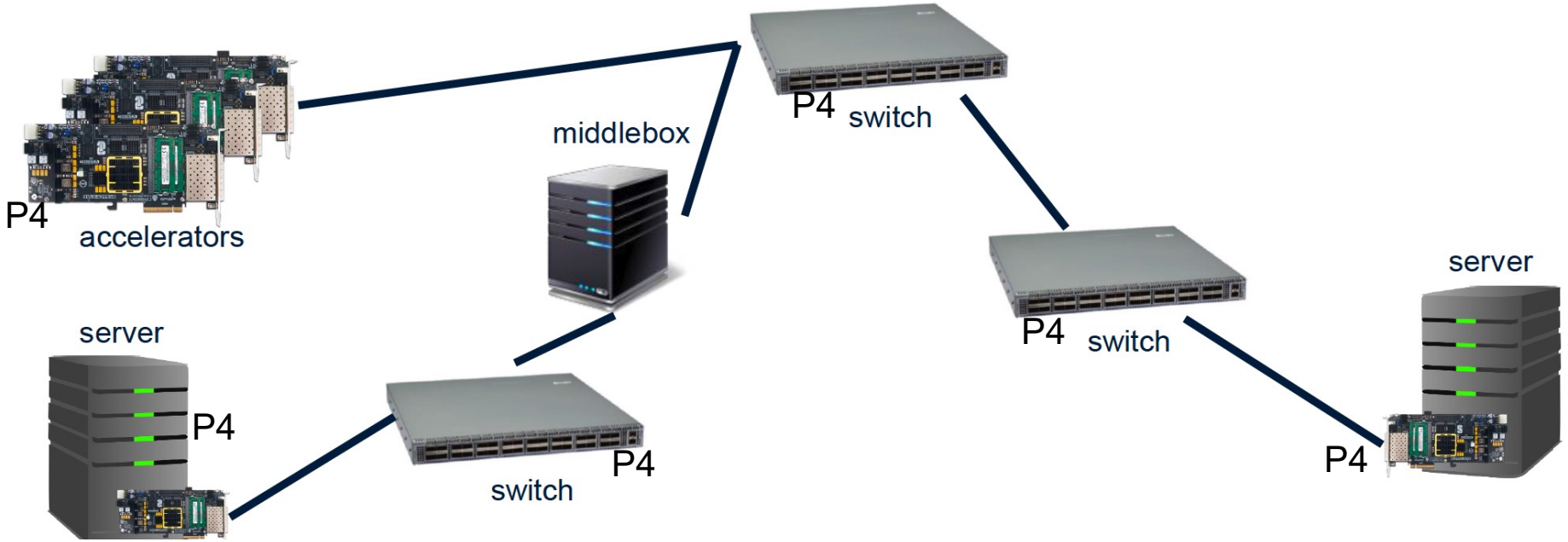
Results

- **Speedup of training throughput up to 2.27x on 100 Gbps networks**
- **Speedup higher with faster GPUs that reduce the computation/communication ratio**



In-Network Computing for ANN Acceleration?

- **Can switches directly run ML pipelines?**
  - At KAU we implemented ANN in SmartNIC and microC
- **E.g. In-network traffic classification**
  - Packets processed anyhow by switches, why not use ML-pipeline for inference (Note: This is NOT training but classification)

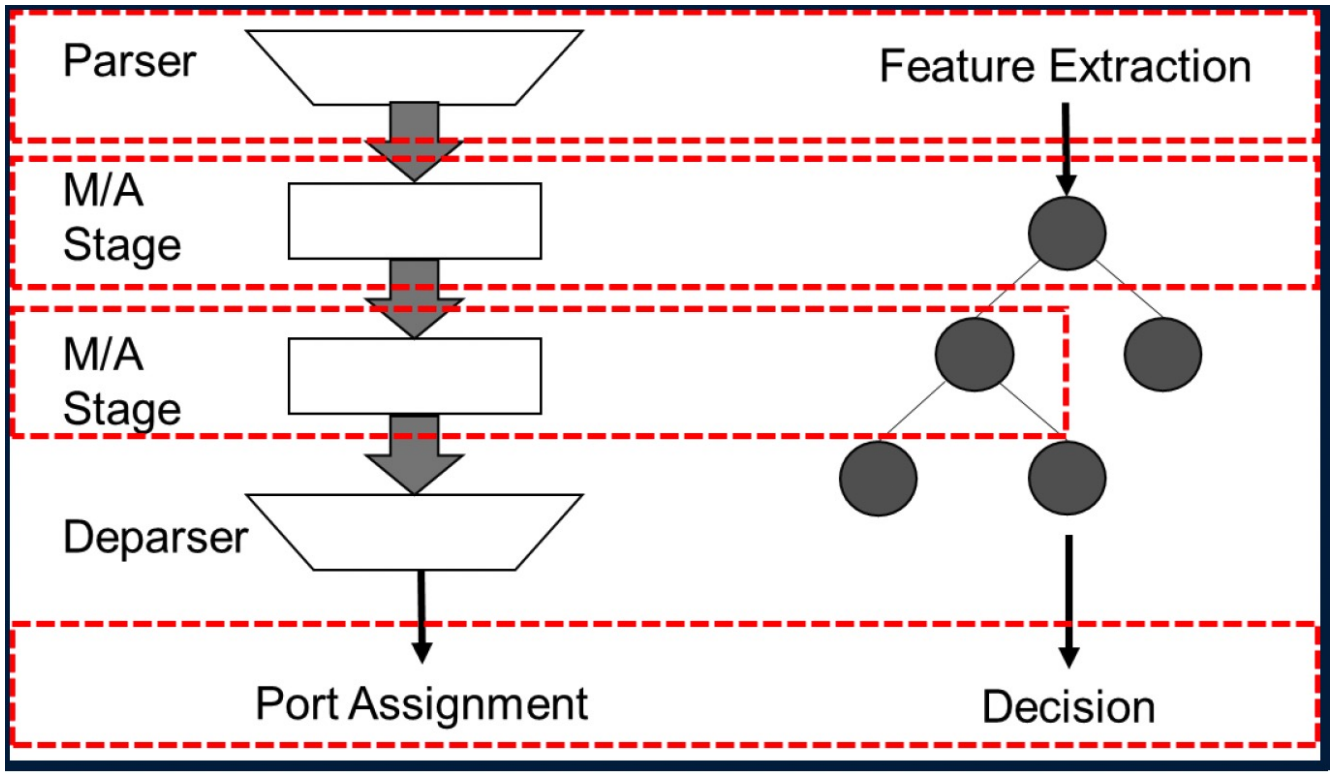


Observation

- **Can implement Neural Network on FPGA**
  - Programming model significantly different → Verilog, VHDL
  - HLS (in C) leads to reduced efficiency
    - https://towardsdatascience.com/neural-network-inference-on-fpgas-d1c20c479e84
  - Expensive, low throughput
  - how to deal with floats → quantization → errors → reduce classification performance
  - Some efforts to develop NN using just integers or binary numbers
  - Failed on network switching ASICS
    - BNN
    - 96 parallel neurons
    - 4096 neurons layer → 2.5% throughput
    - → D. Sanvito et al, “ Can the Network be the AI Accelerator ?”, 2018

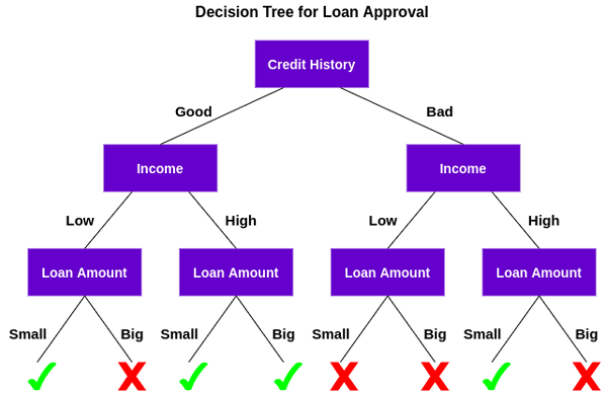
Observation

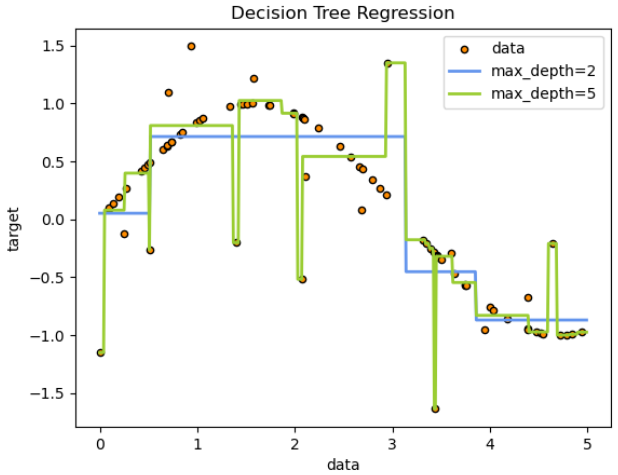
- **P4 based ethernet switch is similar to decision tree**



Decision trees

- **Decision trees, e.g. random forest**
  - non-parametric supervised learning method
  - for classification and regression.
  - predicts the value of a target variable by learning simple decision rules inferred from data features
  - Tree depth decides goodness of model
  - Random forest → multiple decision trees
- **Mapping to RMT is intuitive**
  - In every stage, a set of conditions are applied to the features, their result lead to different tree branches
  - Conditions can be implemented in P4
  - Tree depth and conditions define number of pipeline stages required
  - Can be optimized, see paper





Challenge

- **Map SVM, K-means, etc. to switch architecture**
  - SVM
    - n features, k classes
    - m=k*(k-1)/2

$$\begin{cases} a_1x_1 + b_1x_2 + ...z_1x_n + d_1 = 0 \\ a_2x_1 + b_2x_2 + ...z_2x_n + d_2 = 0 \\ ... \\ a_mx_1 + b_kx_2 + ...z_mx_n + d_m = 0 \end{cases}$$

  - Naïve Bayes
    - feature x_i

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\Big(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\Big)$$

$$\hat{y} = arg\,max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

  - K-means

$$D_i = \sqrt{(x_1 - c_1^i)^2 + (x_2 - c_2^i)^2 + ..(x_n - c_n^i)^2}$$

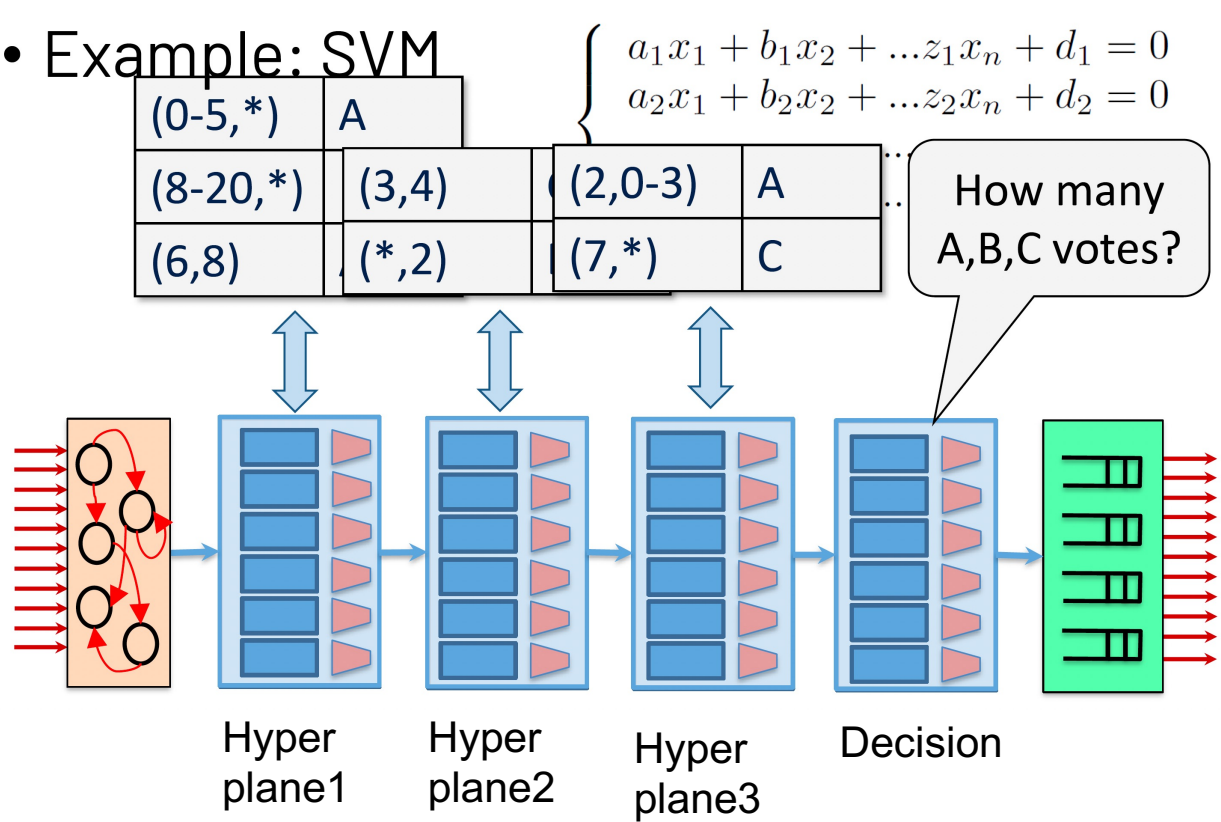
Challenge

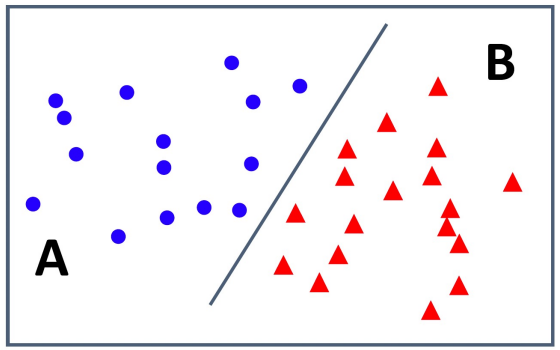
- **Map SVM, K-means, etc. to switch architecture**
- **Required operations**
  - Add/subtract
  - Multiply/divide
  - Power/root
  - Exponents
  - …
- **But in P4 available:**
  - Add/subtract
  - Xor
  - Bitshift
- **→ need to approximate with look-up tables**

Store classification, not the calculated values

- Example: SVM

$$\begin{cases} a_1x_1 + b_1x_2 + \ldots z_1x_n + d_1 = 0 \\ a_2x_1 + b_2x_2 + \ldots z_2x_n + d_2 = 0 \end{cases}$$

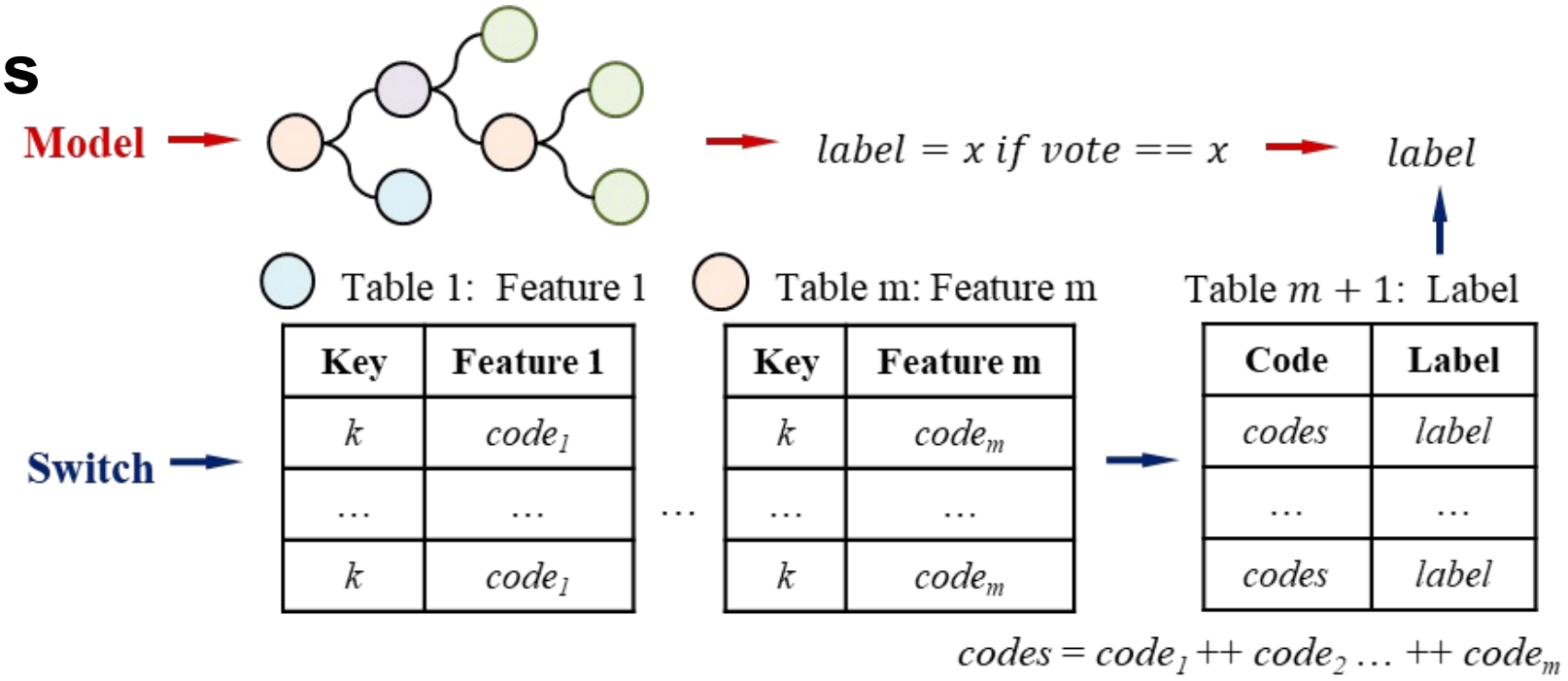




One table per hyperplane, indicating on which side a given input is. Key is set of features. Action is vote (inside or outside). Finally, all votes are counted, highest count wins

Mapping different models to switch data plane

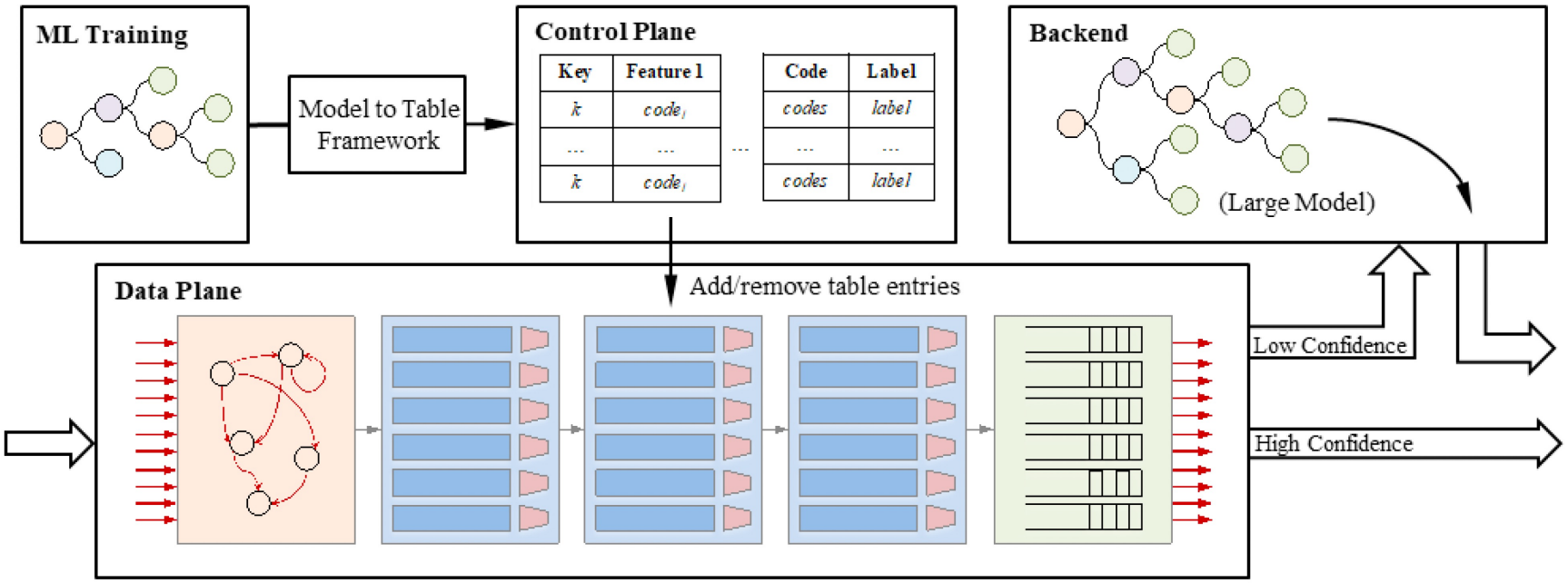
- Decision trees
- SVM
- K-Means
- Naïve Bayes



Building a framework

- **Different options**
  - Limited tables, key action width

	Classifier	A table per...	Key	Action	Last stage
1.	Decision Tree (1)	Feature	Feature's value	Feature's code word	Table, Decoding code words
2.	SVM (1)	Class (hyperplane)	All features	Vote	Logic/table, Votes counting
3.	SVM (2)	Feature	Feature's value	Calculated vector	Logic, hyperplanes calculation
4.	Naïve Bayes (1)	Class & feature	Feature's value	Probability	Logic, highest probability
5.	Naïve Bayes (2)	Class	All features	Probability	Logic, highest probability
6.	K-means (1)	Class & feature	Feature's value	Square distance	Logic, overall distance
7.	K-means (2)	Cluster	All features	Distance from core	Logic, distance comparison
8.	K-means (3)	Feature	Feature's value	Distance vectors	Logic, overall distance



Evaluation

- **IoT Traffic Classification**
  - Dataset: UNSW IoT traces: Static data, sensors, audio, video, other traffic
  - There are no features that are there to help identifying traffic types
    - No MAC, IP adresses,…
    - But 11 features in total, e.g. port, ethertype,
  - Traffic of certain type forwarded on certain port
- **Measurements:**
  - Decision tree achieves
    - 94% accuracy with a depth of 11
    - 85% with a depth of 5
  - Increased switch table/memory utilization compared to reference switch implementation due to additional tables needed

Model	# tables	Logic Util.	Memory Util.
Reference Switch	3	15%	33%
Decision Tree	6	27%	40%
SVM (1)	11	34%	53%
Naïve Bayes (2)	5	30%	44%
K-means	5	30%	44%

Summary

- **In-Network support for Machine Learning**
  - Help to accelerate distributed training by in-network aggregation of distributed models
- **Can map ANN structure to P4 pipeline for In-network traffic classification**
  - Challenge is in how to map the model to the switch architecture
  - Works only for small and stateless models
- **Future Work**
  - Increasing model size
  - Distributing models on multiple switches
  - What are new use-cases?