Scaling Deep Learning on Multi-GPU Servers

Peter Pietzuch

(with Alexandros Koliousis, Luo Mai, Pijika Watcharapichat, Matthias Weidlich, Paolo Costa)

Imperial College London

http://lsds.doc.ic.ac.uk
<prp@imperial.ac.uk>

SICS – Sweden – September 2018

Deep Neural Networks (DNN) at Scale

Image classification

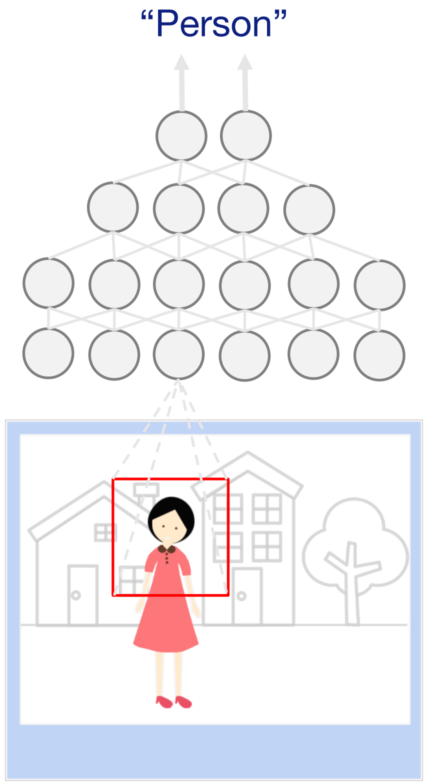
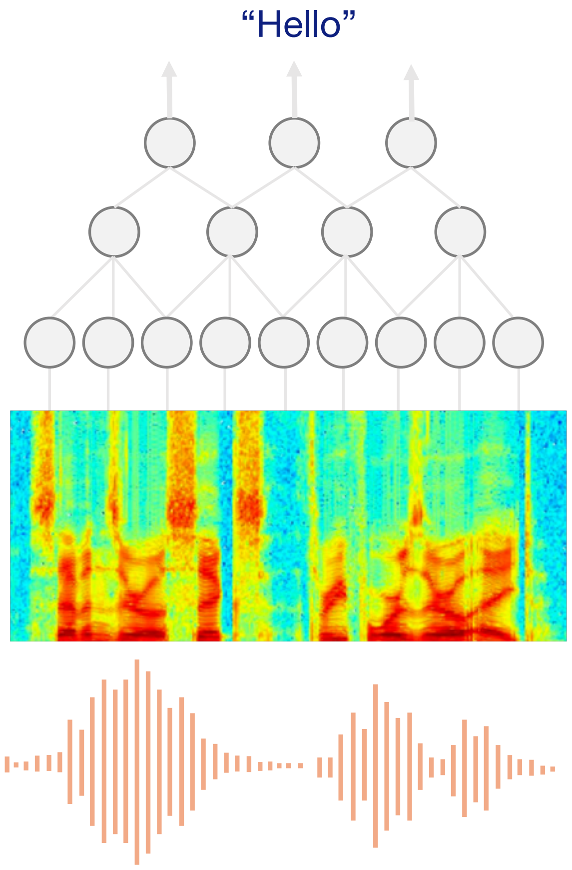


Image data

Train DNNs with a lot of data

Speech recognition



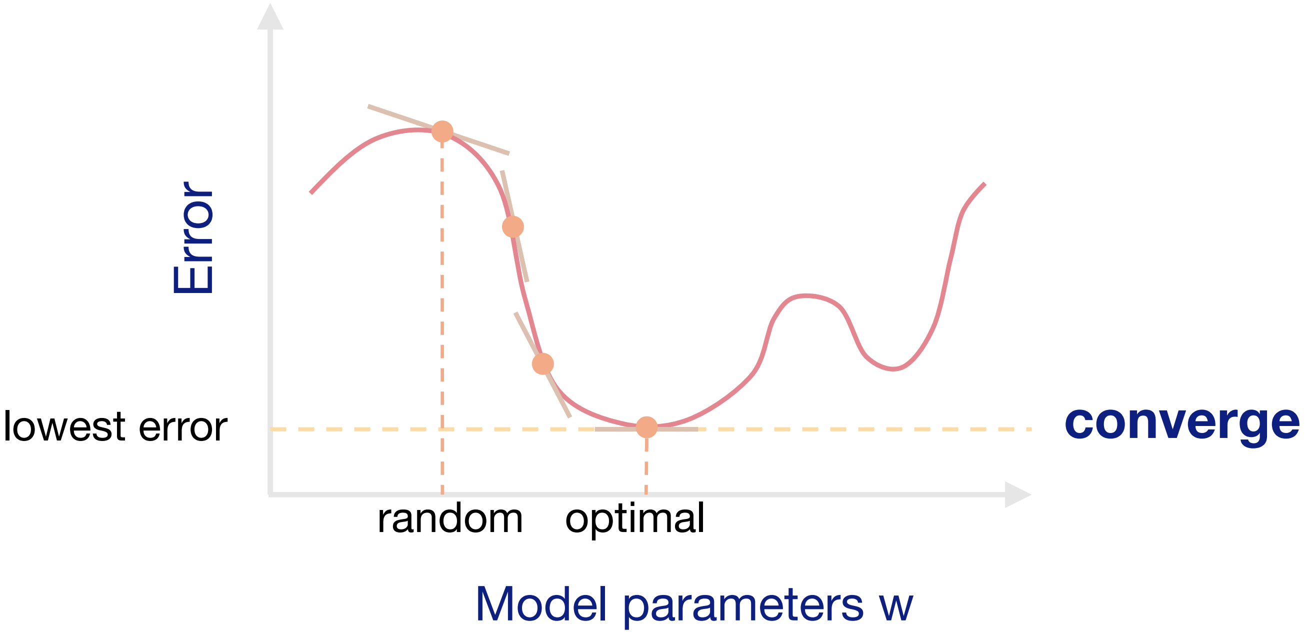
Audio data

Training DNNs with Stochastic Gradient Descent

Goal: Obtain DNN model that minimises classification error

Stochastic Gradient Descent (SGD):

- Consider mini-batch of training data
- Iteratively calculate gradients and update model parameters w



The Problem With Large Batch Sizes

"Training with large mini-batches is bad for your health. More importantly, it's bad for your test error. Friends don't let friends use mini-batches larger than 32."

– Y. LeCun, @ylecun, April 26, 2018

Scaling DNN Training on GPUs

Manager:

Need a highly-accurate image classification model ASAP

Highly-Paid Data Scientist:

OK. I'll use ResNet-50. It will take ½ month on 1 GPU

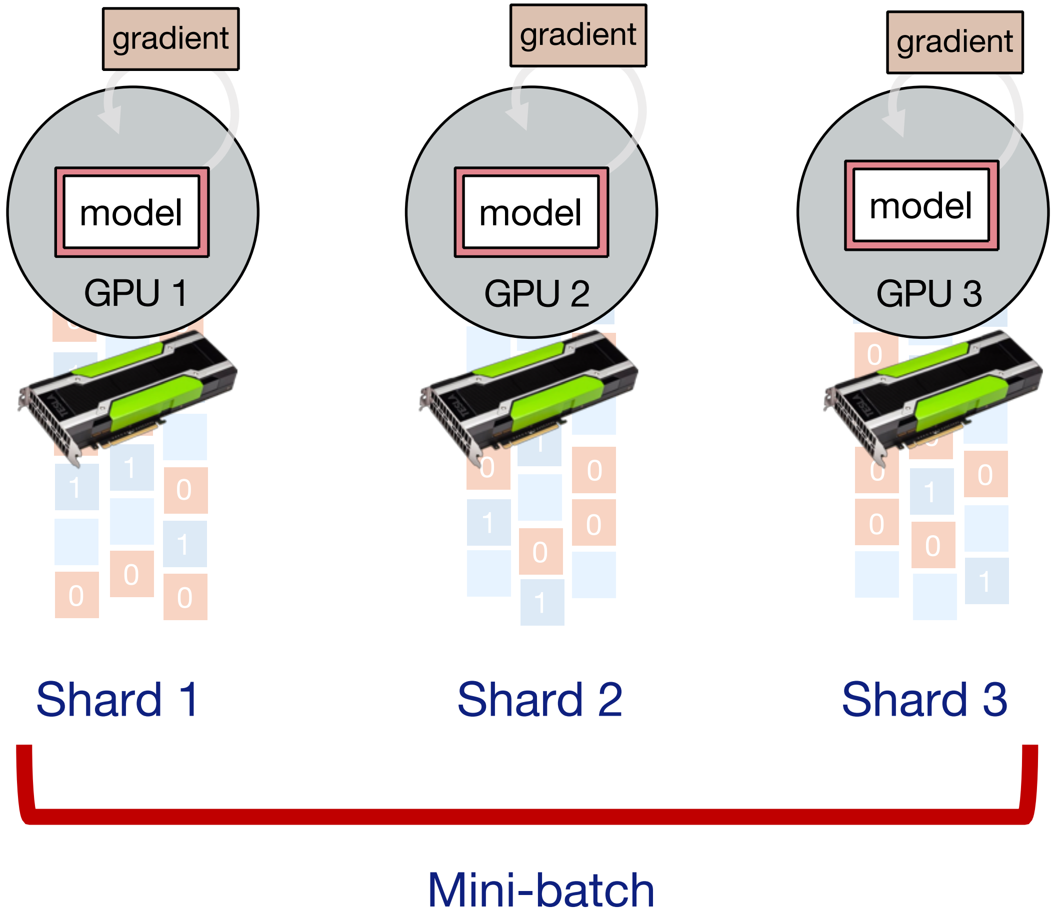
M:

Throw more hardware at the problem! Need this sooner.

HPDS: …

Parallel DNN Training

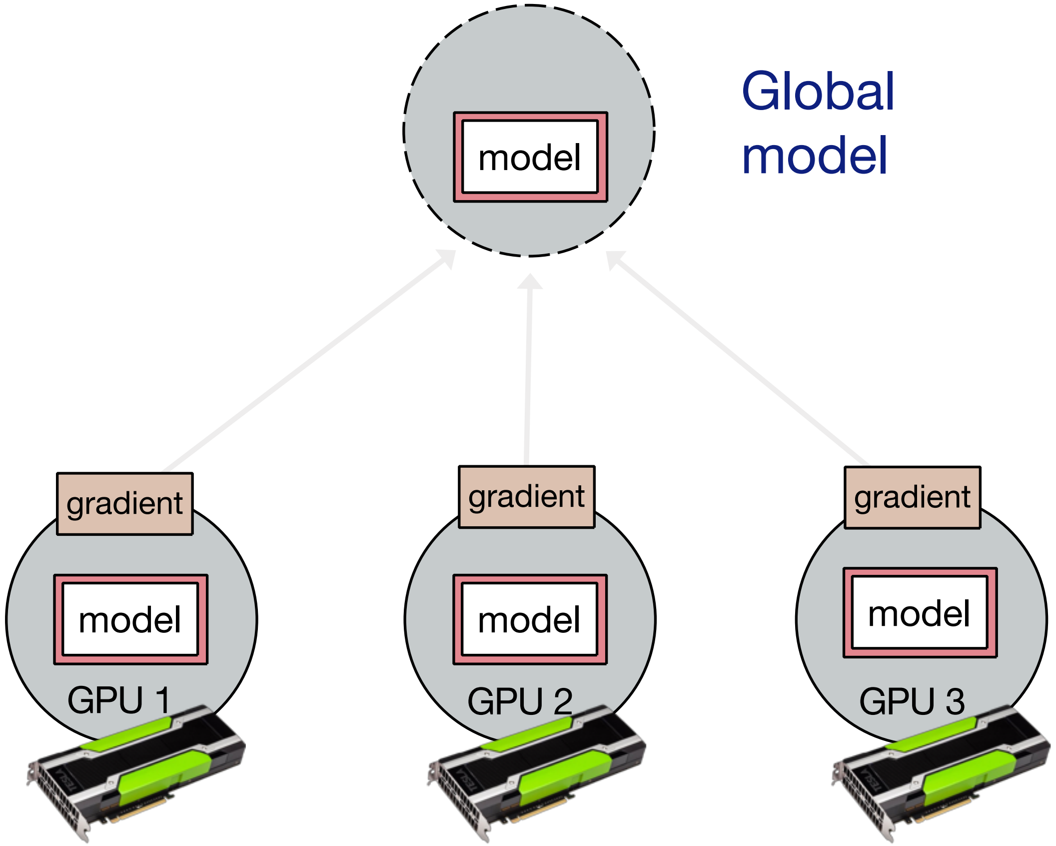
With large training datasets, speed up by calculating gradients in parallel



Model replicas diverge over time

Synchronisation Among GPUs

Parameter server: Maintains global model

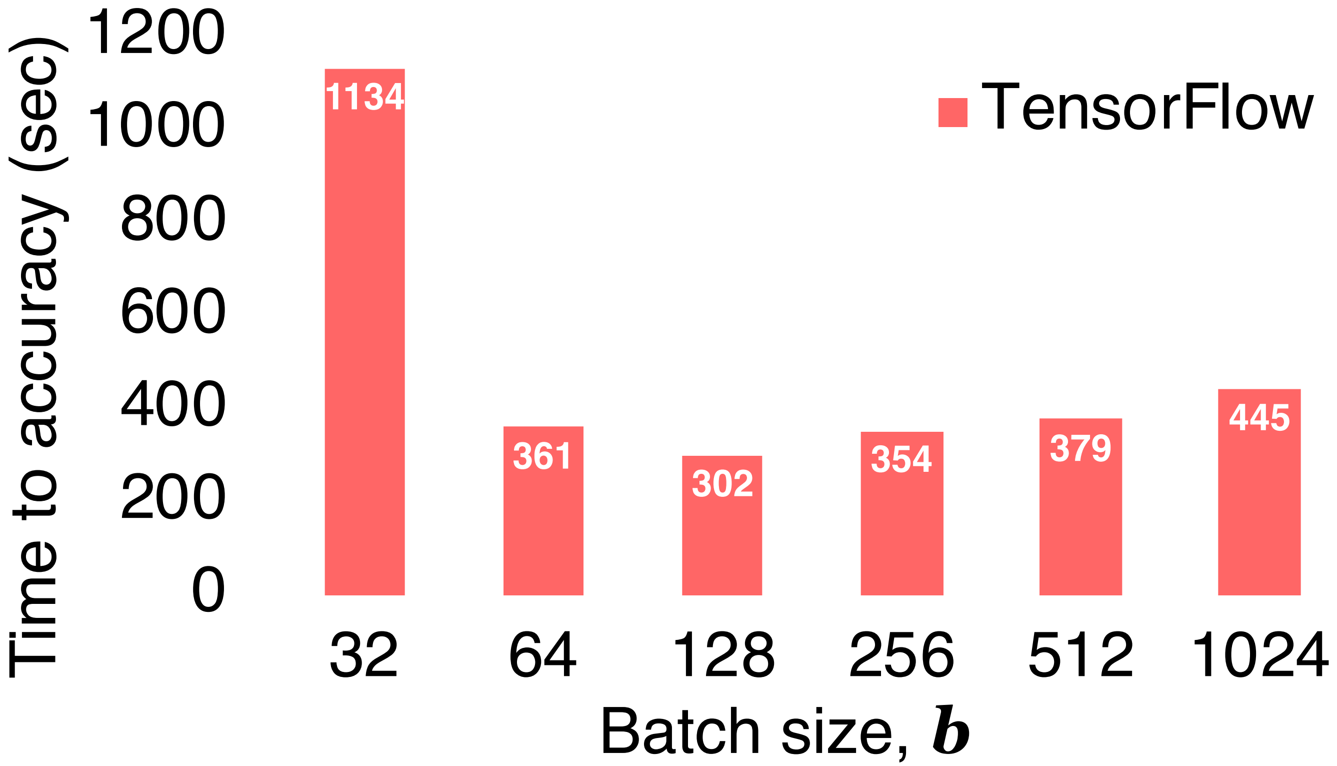


GPUs:

1. Send gradients to update global model
2. Synchronise local model replicas with global model

What is the Best Batch Size?

ResNet-32 on Titan X GPU



Intuition for Small Batch Sizes

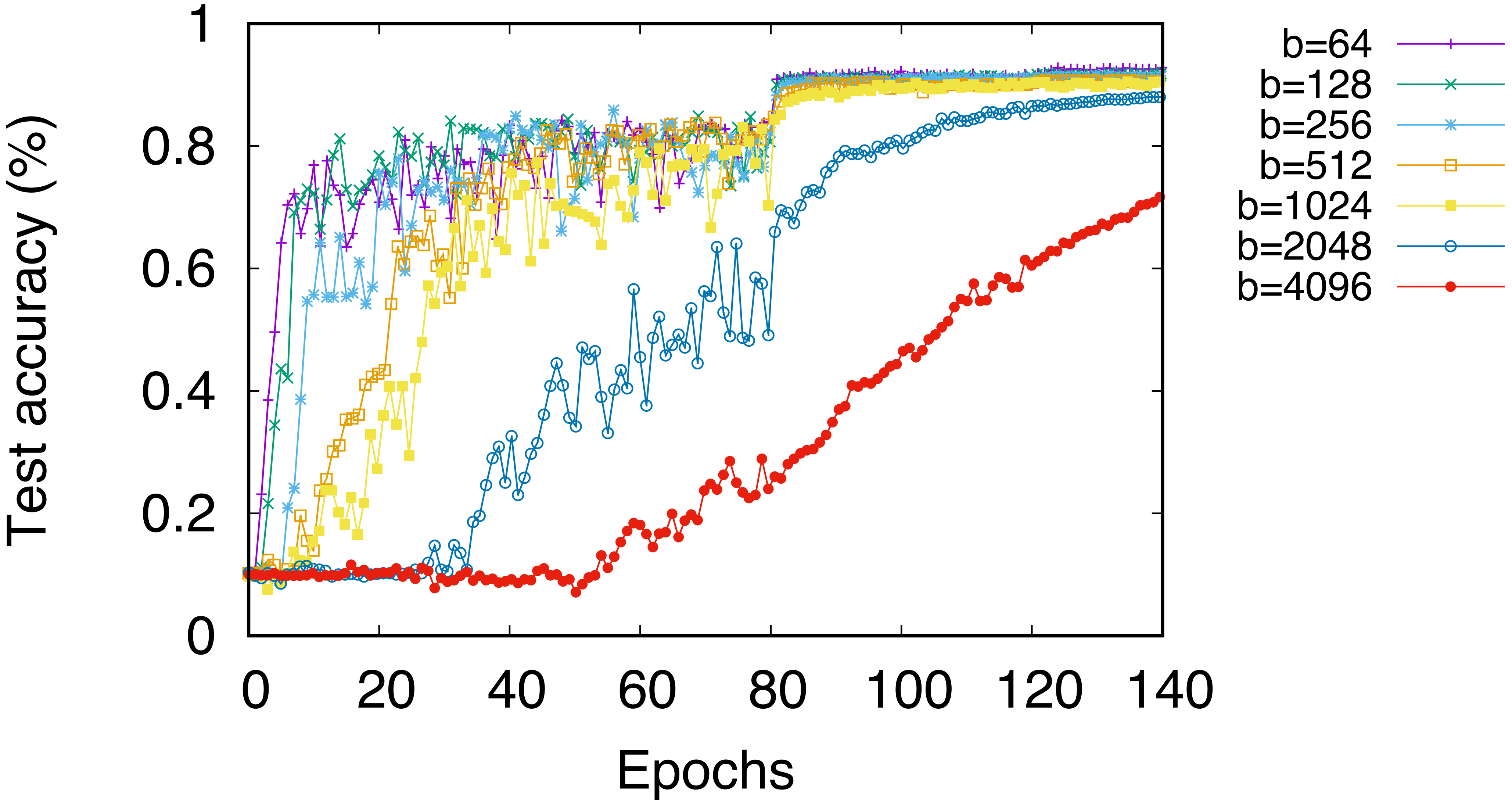
Practical considerations:

More frequent model updates minimise bias to initial conditions faster (i.e. initial weights are forgotten faster)

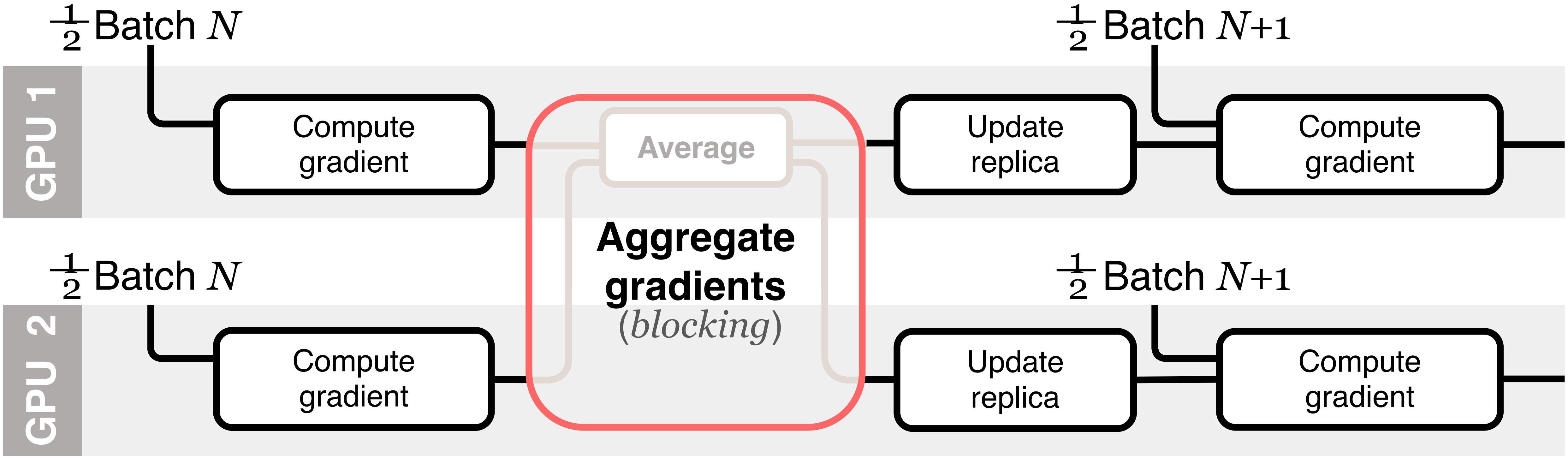
Theoretical considerations:

Small batch sizes explore better the wider minima, which are known to exhibit better test accuracy

Statistical Efficiency Needs Small Batch Sizes



Hardware Efficiency Needs Large Batch Sizes



Keep **work per GPU constant => scale batch size** with #GPUs

Hardware & Statistical Efficiency

“Training with large mini-batches is bad for your health. More importantly, it’s bad for your test error. Friends don’t let friends use mini-batches larger than 32.”

– Y. LeCun, @ylecun, April 26, 2018

The only reason practitioners increase batch size is **hardware efficiency**

But best batch size depends on both
hardware efficiency & **statistical efficiency**

Limits of Scaling DNN Training

HPDS: Managed to train it on 100s of GPUs in 1h!

M: Great! Make it faster. Use as many resources as you need!

HPDS: Can't :(Beyond this point statistical efficiency collapses. Actually, I straggled to maintain it up to this point…

M: …

The story continues. New hardware becomes available. Improved communication between workers. Latest results train ResNet-50 in just few minutes. But the problem remains.

Hyper-Parameter Tuning

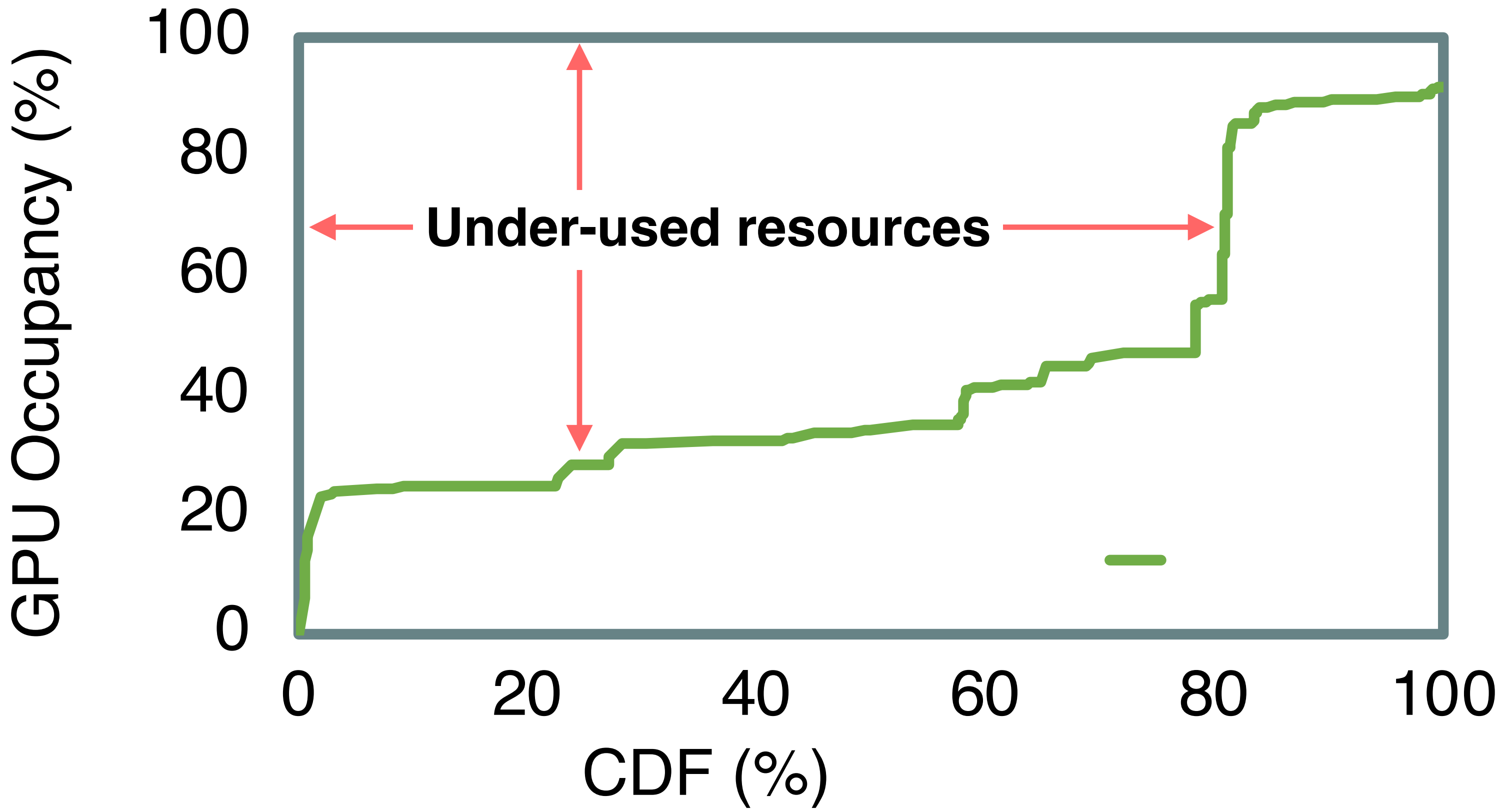
The Fundamental Challenge of GPU Scaling

"If batch size could be made arbitrarily large while still training effectively, then training is amenable to standard weak scaling approaches. However, if the training rate of some models is restricted to small batch sizes, then we will need to find other algorithmic and architectural approaches to their acceleration."

– J. Dean, D. Patterson, X. […], "The Golden Age", IEEE Micro

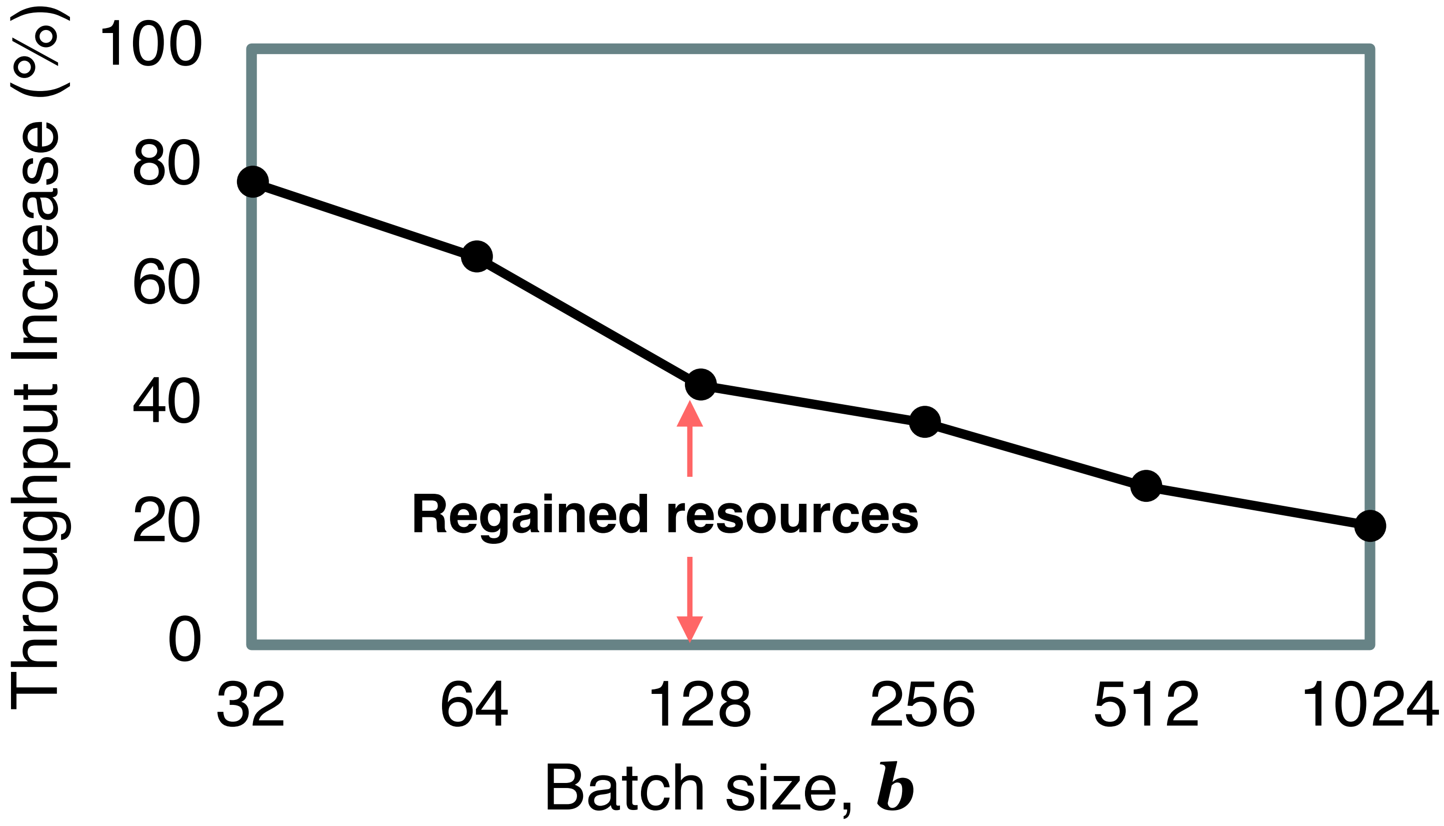
How to design a system that can scale training with multiple GPUs even when the preferred batch size is small?

Problem: Small Batch Sizes Underutilise GPUs



Idea: Train Multiple Model Replicas per GPU

Fully exploits task parallelism on GPU



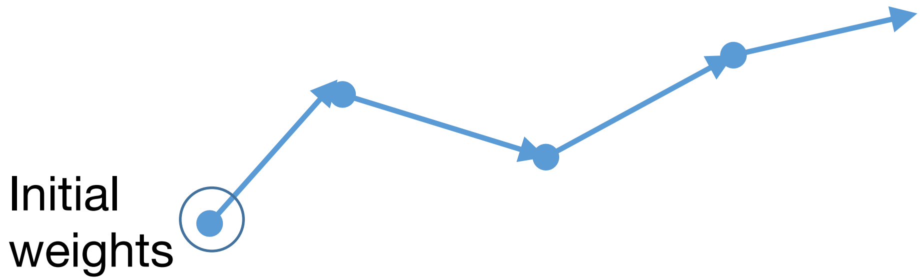
But we now need to synchronise large number of model replicas...

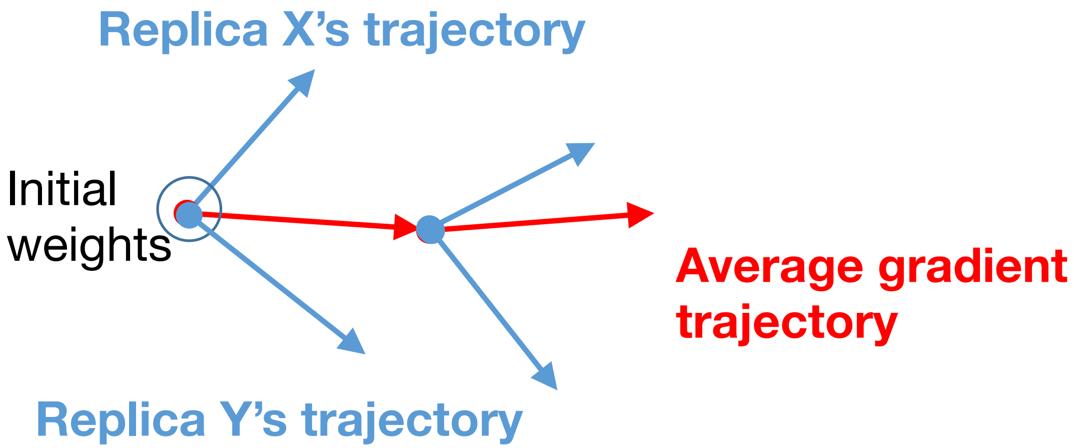
How to Synchronise Many Model Replicas?

Synchronised SGD:

- Average gradients of multiple workers
- Start next training with average model

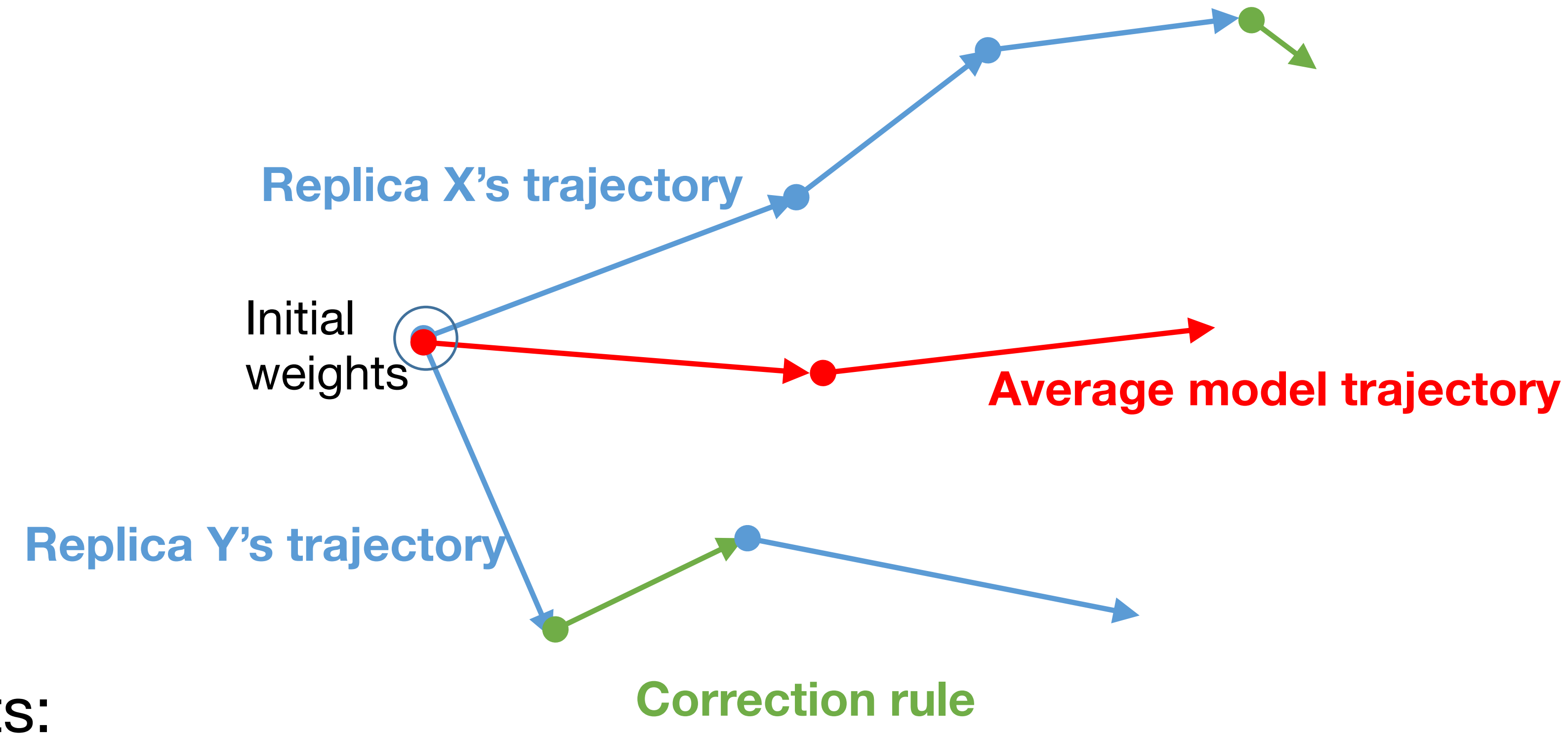
Workers start next exploration from same point in weight space





Idea: Sample Multiple Nearby Points in Space

Synchronisation using Elastic Averaging SGD (EA-SGD)

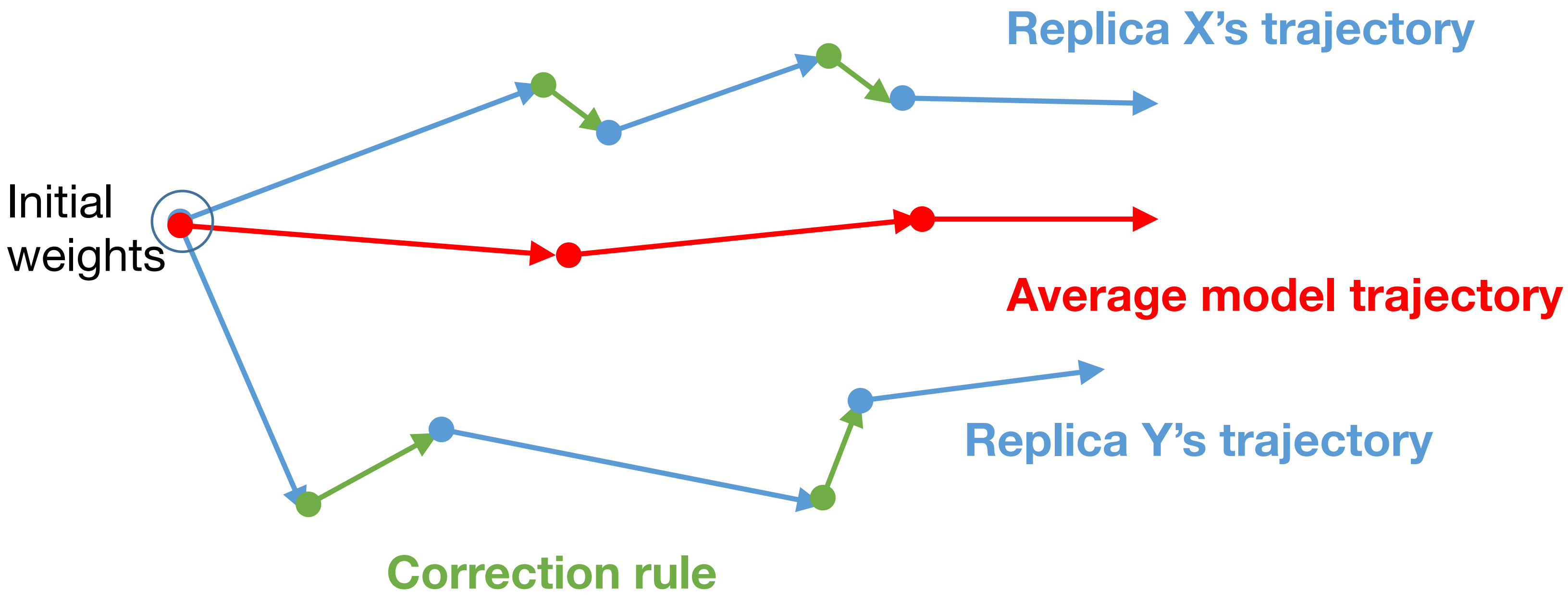


Benefits:

1. Each replica reuses learning set-up of small batch size
2. Increased exploration through parallelism
3. Average replica helps with direction to good minima

When To Apply Corrections?

Synchronously apply corrections to model replicas

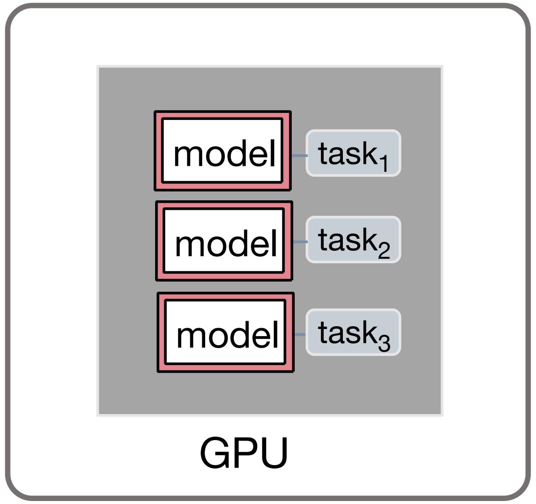


Other techniques

– Control impact of average model using **momentum**
– **Auto-tune number** of model replicas

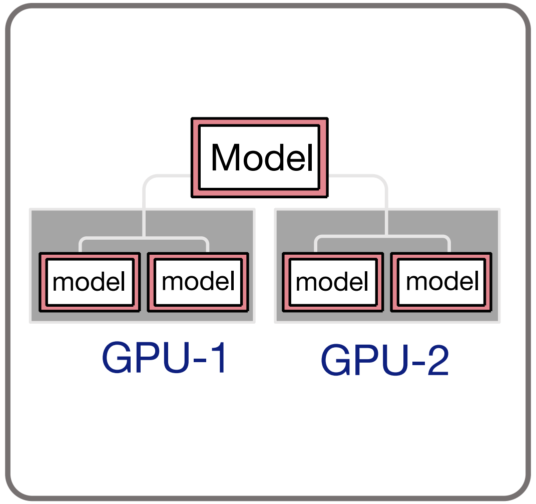
Crossbow: Multi-GPU Deep Learning System

(1) Train multiple models in parallel



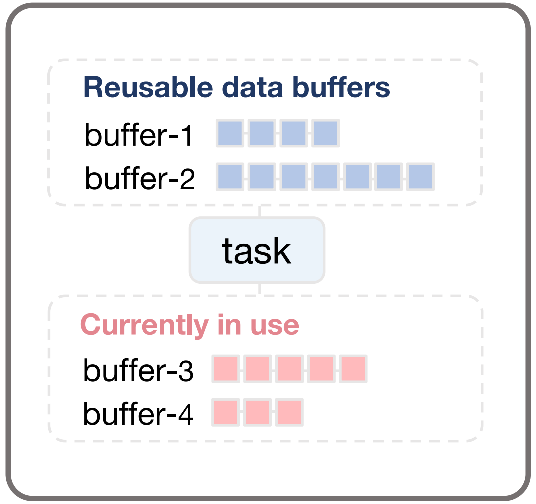
Execute concurrent tasks to leverage GPU concurrency

(2) Synchronous hierarchical model averaging



Minimise amount of data transfer among different GPUs

(3) Efficient fine-grained task engine



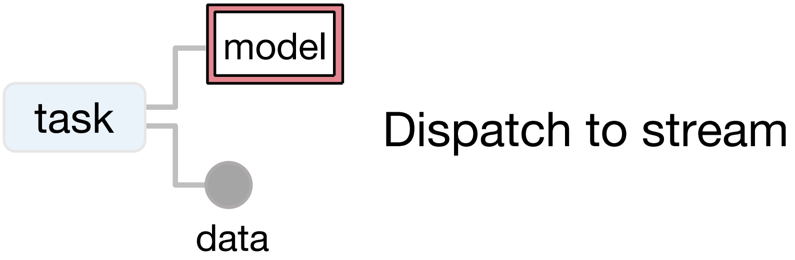
Schedule fine-grained compute & synchronisation tasks

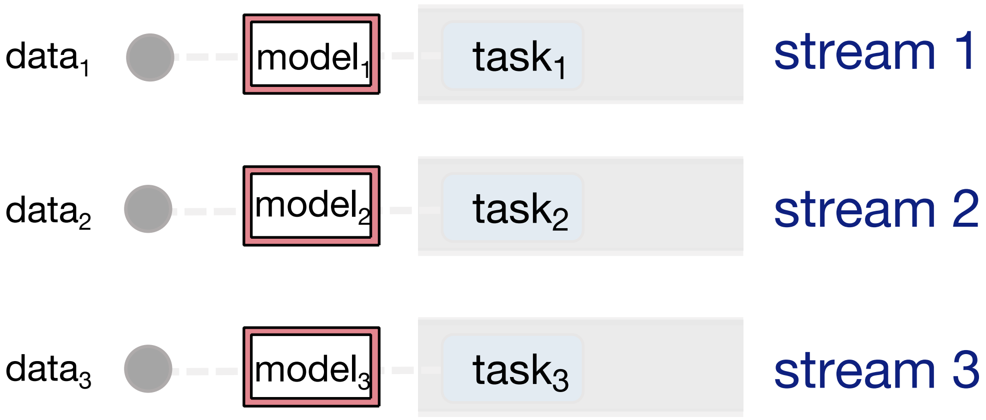
GPU Parallelism with Multiple Model Replicas

On each GPU, use **different streams** for training tasks

Task: series of operations based on computation of model layers

- Task associated with
  (a) model replica and
  (b) batch of training data

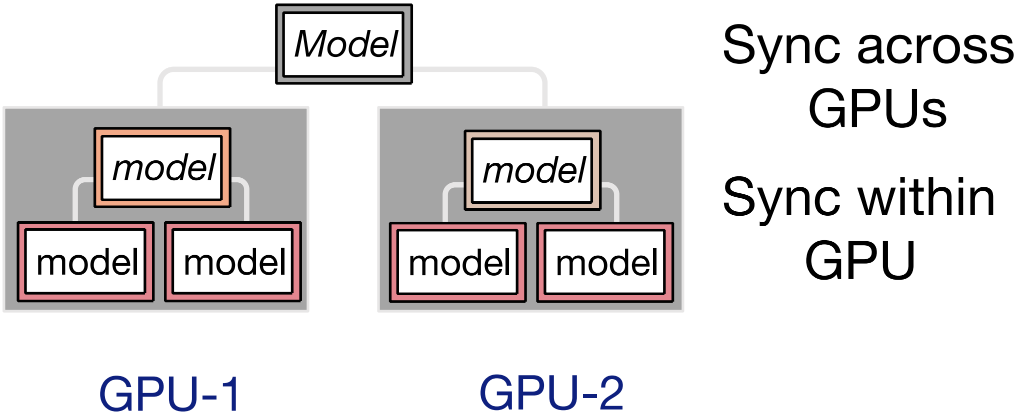




Synchronous Hierarchical Elastic Model Averaging

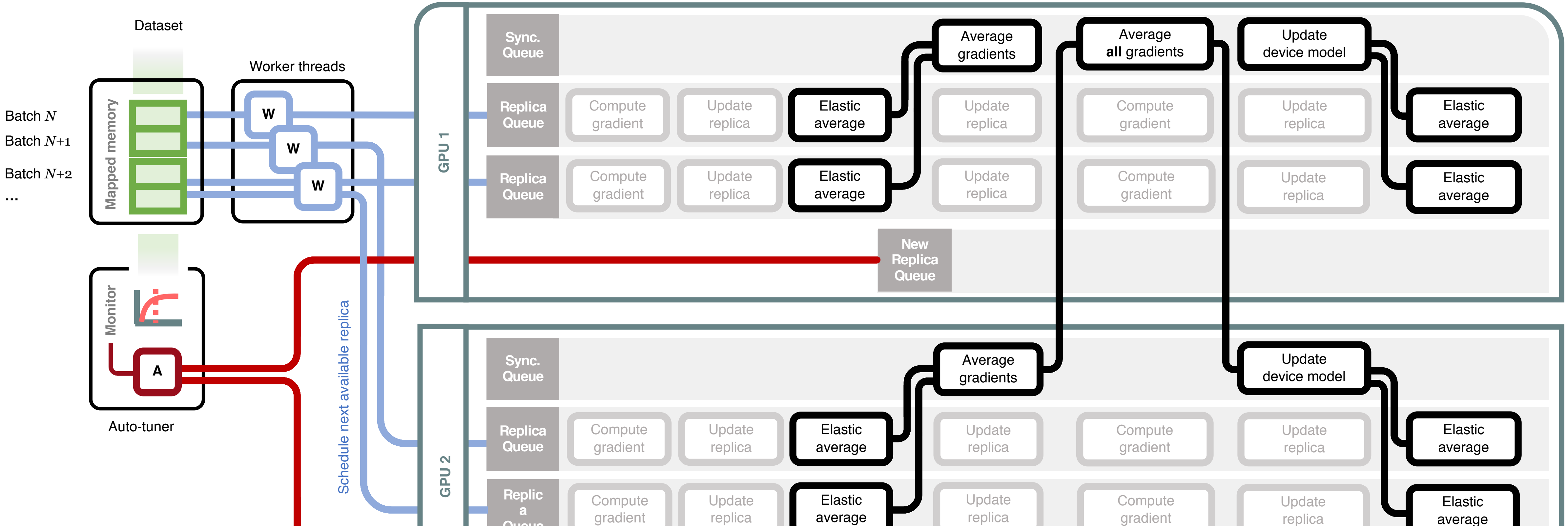
Intra-GPU, cross-GPU, and CPU-GPU communication have different performance characteristics

Crossbow uses **hierarchical** aggregation to avoid bottlenecks



CrossBow Task Execution Engine

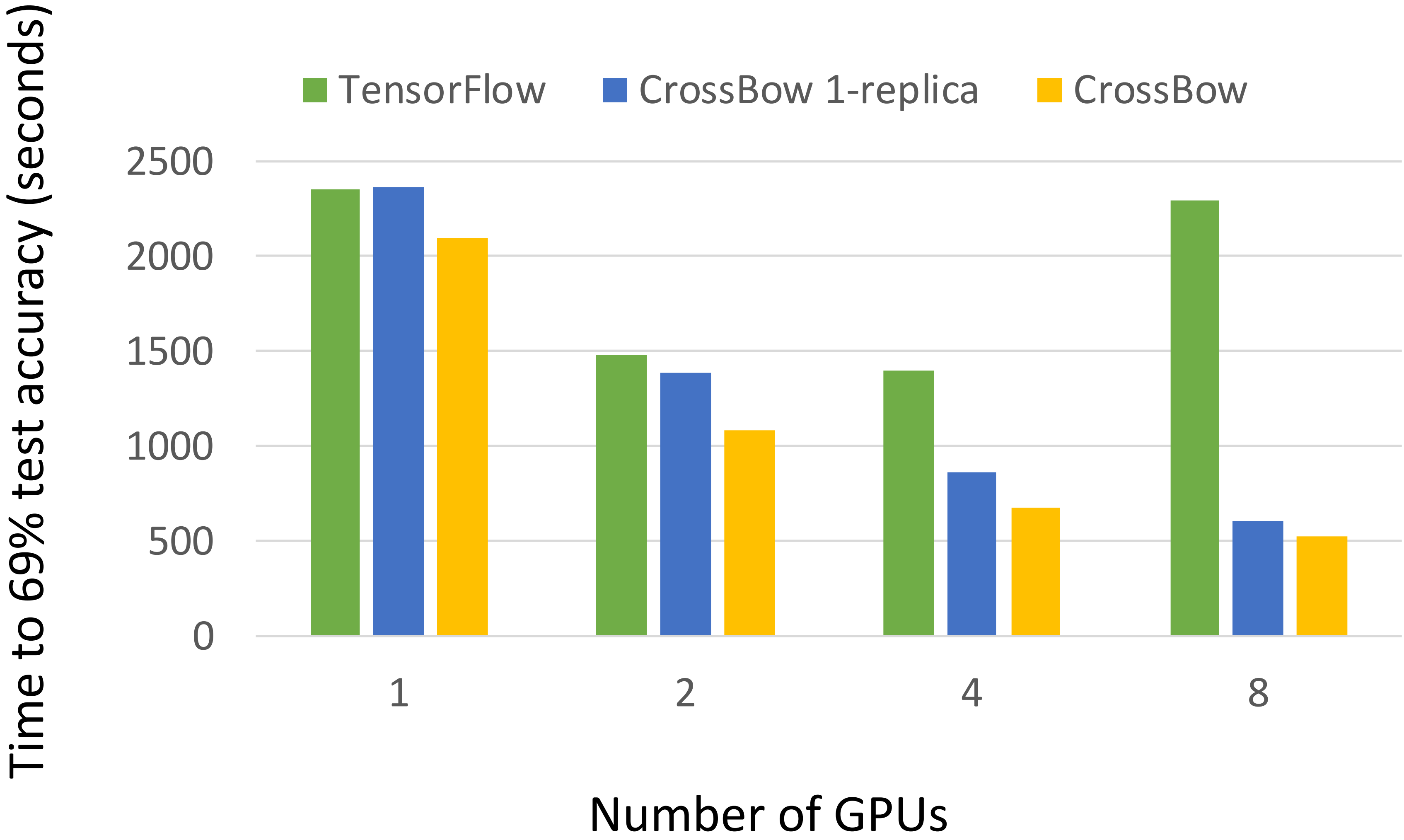
Many small tasks for (1) replica training and (2) synchronisation



CrossBow has GPU/CPU task engine for multiplexing & overlapping small tasks

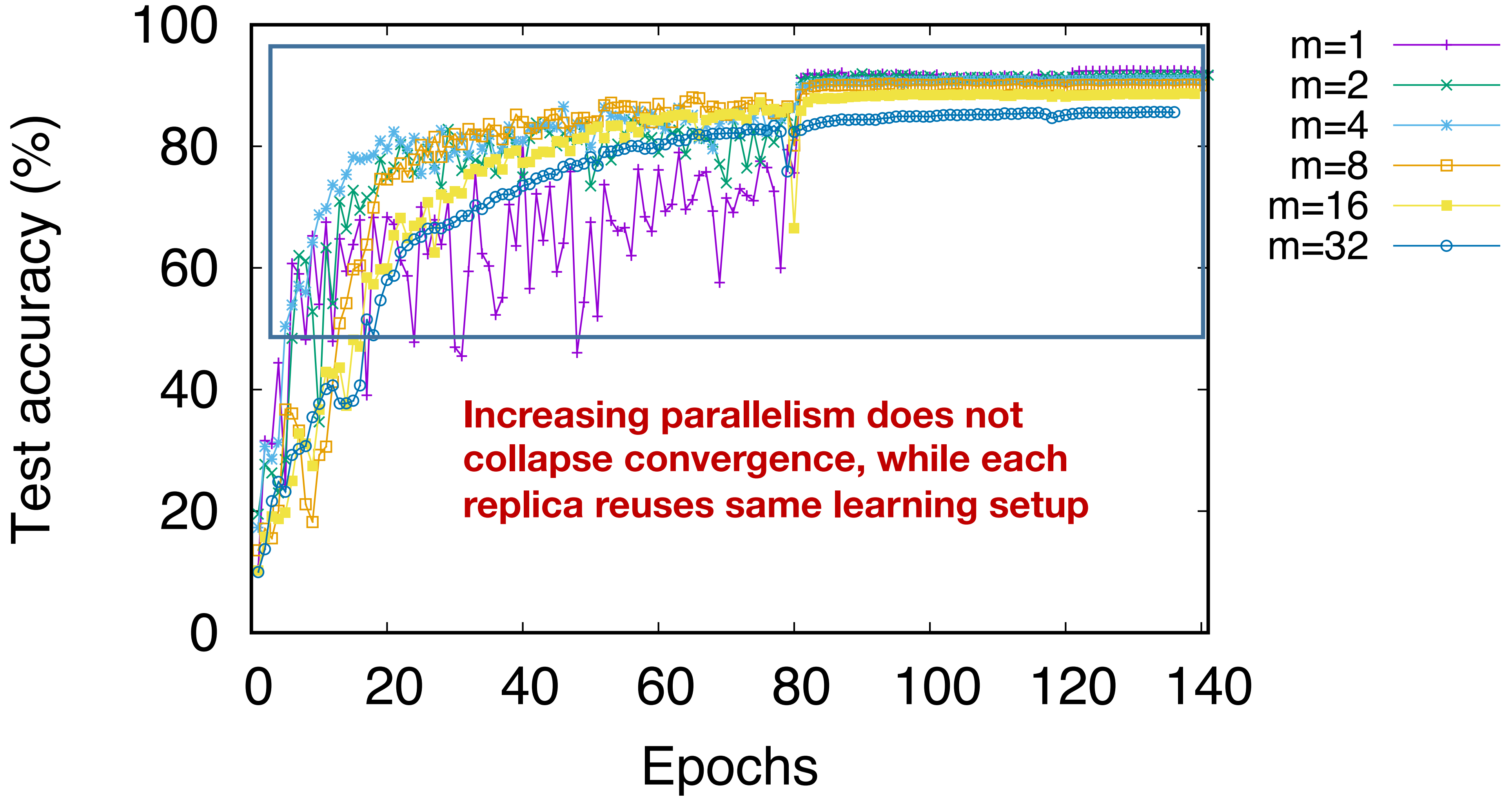
CrossBow: Benefit of Synchronous Model Averaging

VGG-16 with Cifar-100 dataset on Titan X GPUs



CrossBow: Statistical Efficiency with Many Models

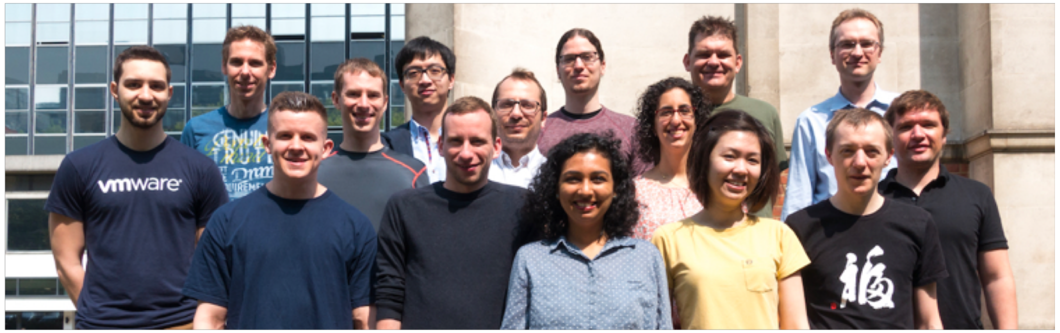
ResNet-50 with ImageNet dataset on Titan X GPUs



Summary: Scaling Deep Leaning on Multi-GPU Servers

Need to make **training throughput** independent from **hyper-parameters**
- Current need for hyper-parameter tuning too complex
- Need new designs for deep learning systems

Crossbow: Scaling DNN training with small batch sizes on many GPUs
- **Multiple model replicas** per GPU for high hardware efficiency
- **Synchronous hierarchical model averaging** for high statistical efficiency
- Requires new CPU/GPU task engine design



Thank You — Any Questions?

Peter Pietzuch
https://lsds.doc.ic.ac.uk — prp@imperial.ac.uk