Caffe con Troll:

Shallow Ideas to Speed Up **Deep** Learning

Stefan Hadjis[1], Firas Abuzaid[1], Ce Zhang[1,2],Christopher Ré[1]

[1]Stanford University, [2]University of Wisconsin-Madison

github.com/HazyResearch/CaffeConTroll

Outline

Motivation

- CPU / GPU gap?

4 Shallow ideas for ***FLOP-proportional scheduling***

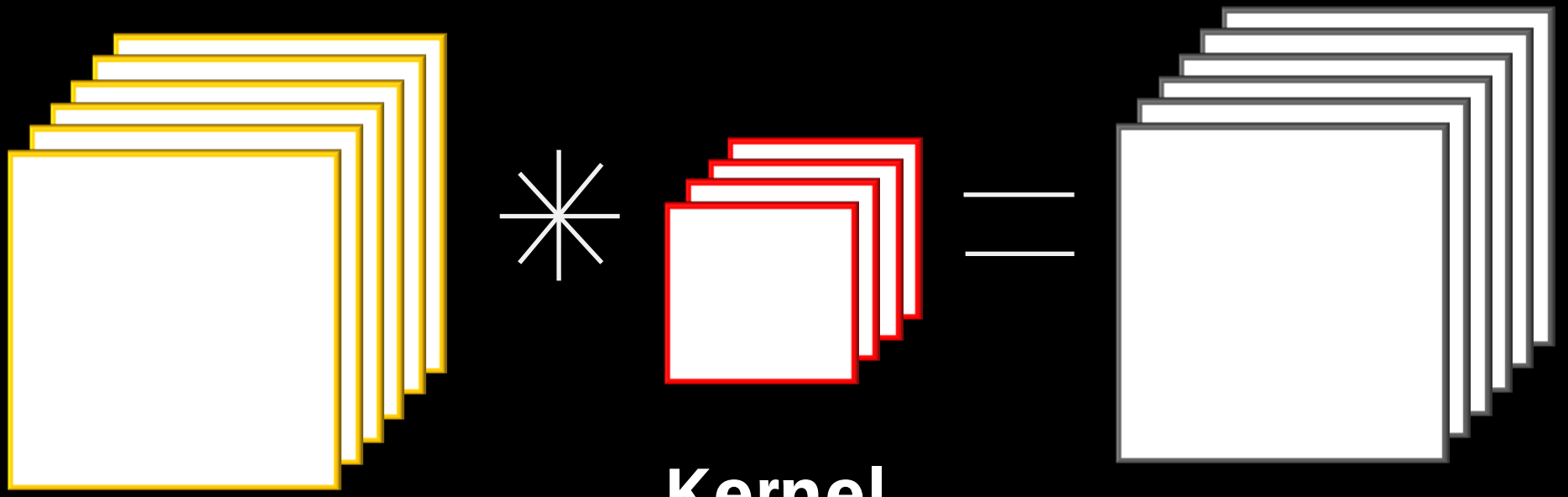
- Order of magnitude speedup on CPU
  - Close CPU / GPU gap
  - Operate all devices proportional to their FLOPS
- *Lets us use CPUs + GPUs together!*

What's next

- New optimizations

Convolutional Neural Nets

- 70-90% of time spent doing **Convolutions**



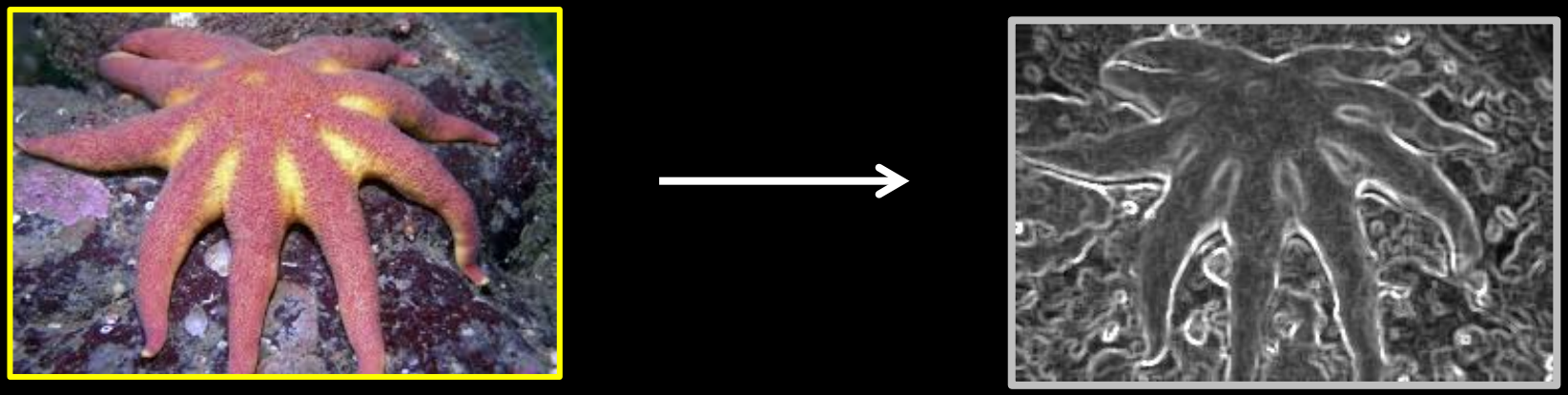
Input Images

Kernel (Convolution Weights)

Output Images

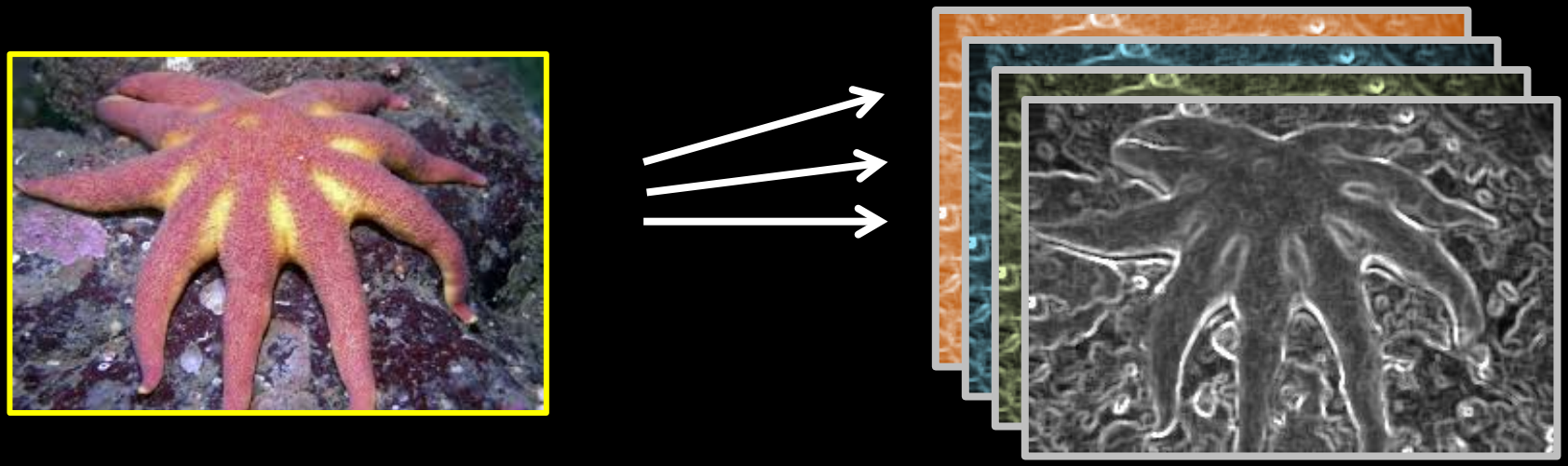
Convolutional Neural Nets

- 70-90% of time spent doing **Convolutions**



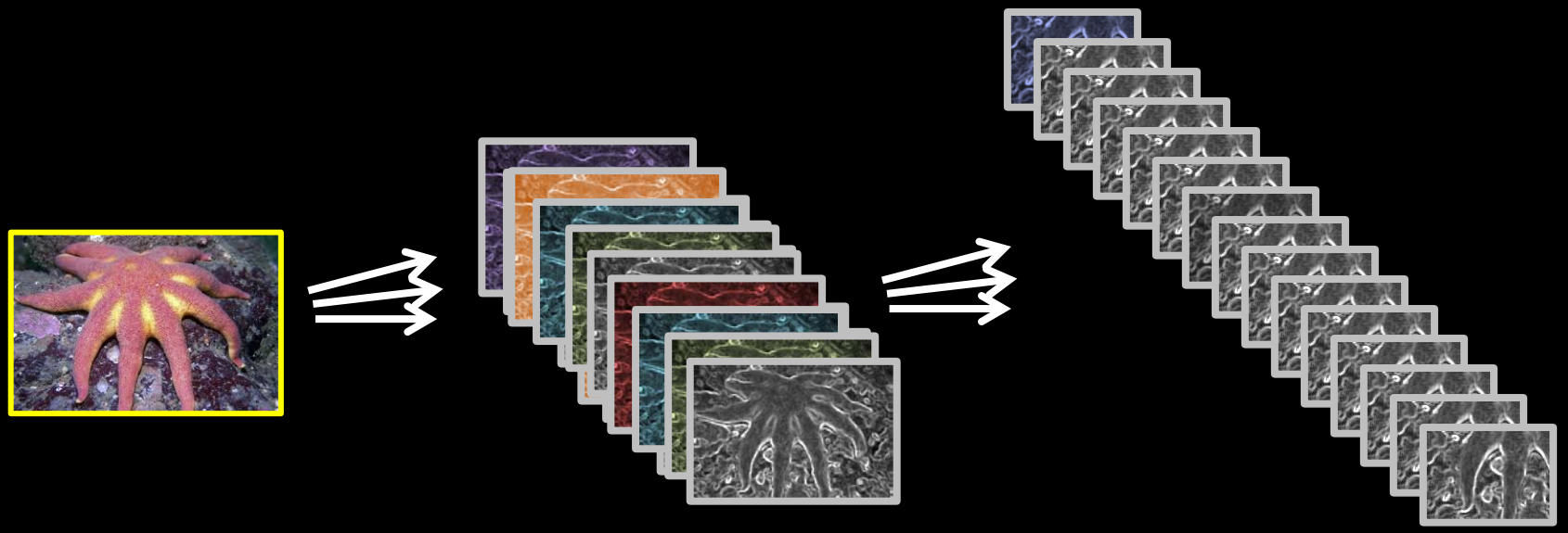
Convolutional Neural Nets

- 70-90% of time spent doing **Convolutions**



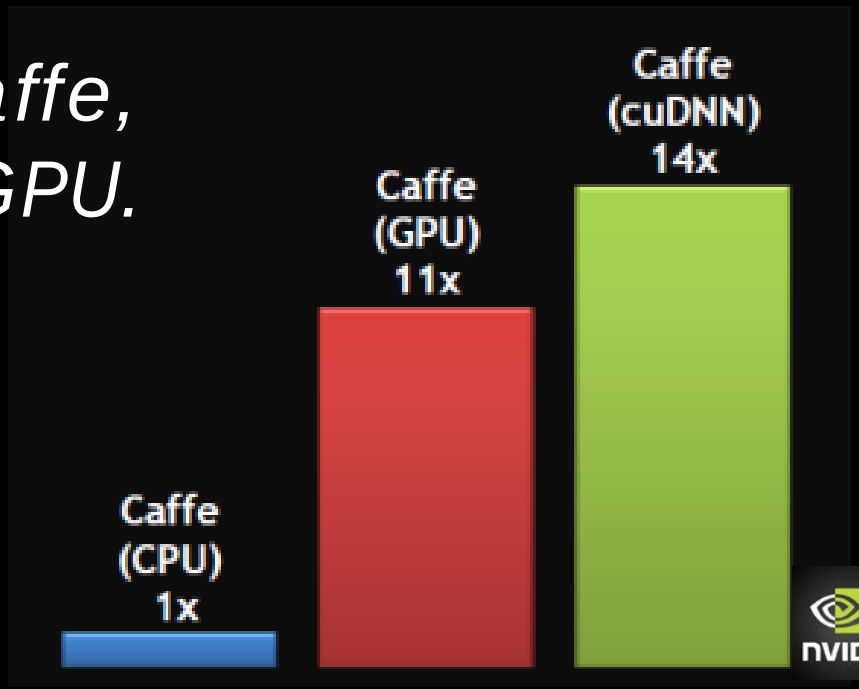
Convolutional Neural Nets

- 70-90% of time spent doing **Convolutions**



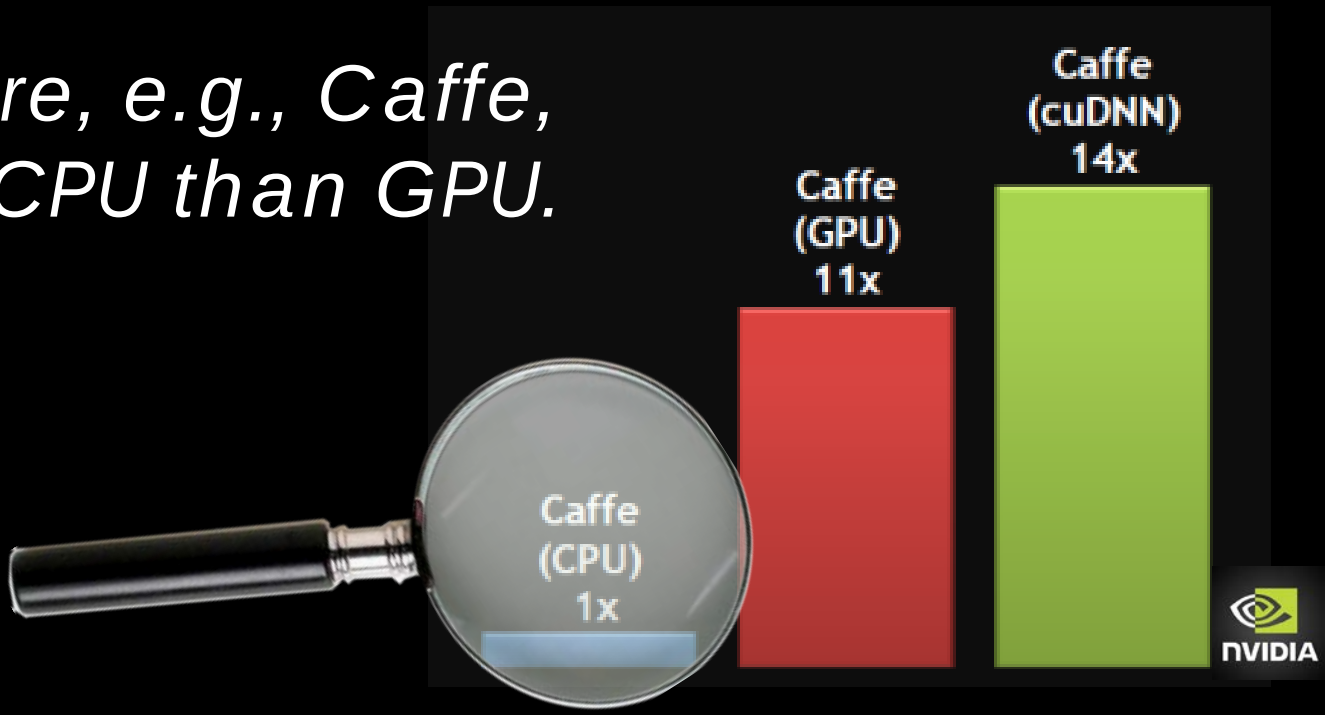
CPU or GPU?

Existing software, e.g., Caffe, 10x slower on CPU than GPU.



CPU or GPU?

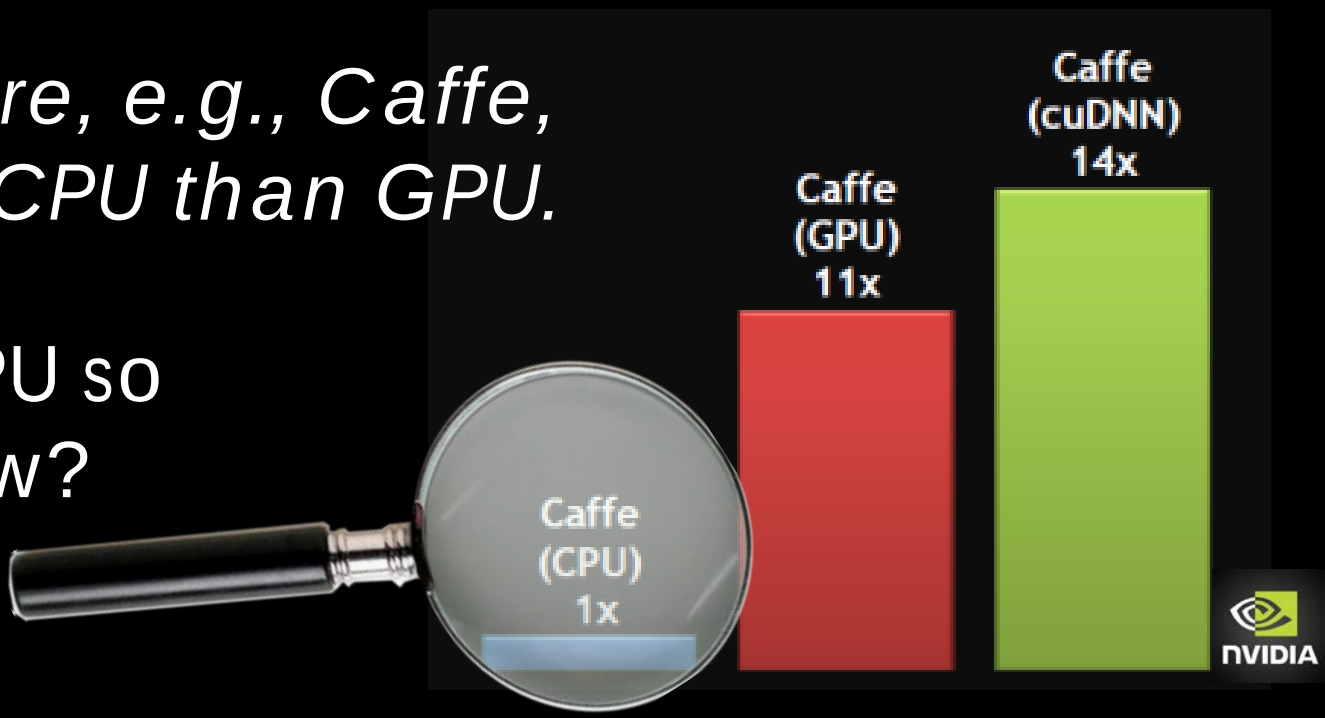
Existing software, e.g., Caffe, 10x slower on CPU than GPU.



CPU or GPU?

Existing software, e.g., Caffe, 10x slower on CPU than GPU.

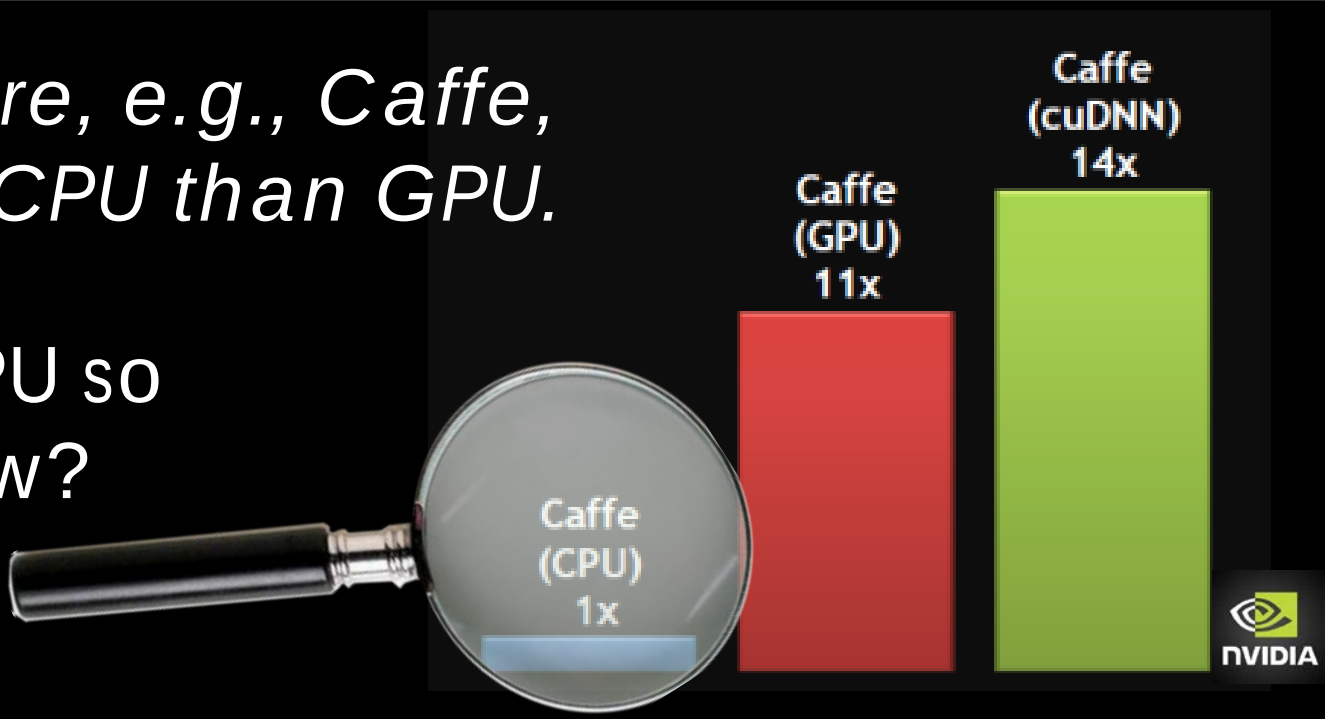
Why is the CPU so Relatively slow?



CPU or GPU?

Existing software, e.g., Caffe, 10x slower on CPU than GPU.

Why is the CPU so Relatively slow?



EC2: c4.4xlarge

8 cores@2.90GHz

0.7TFlops

EC2: g2.2xlarge

1.5K cores@800MHz

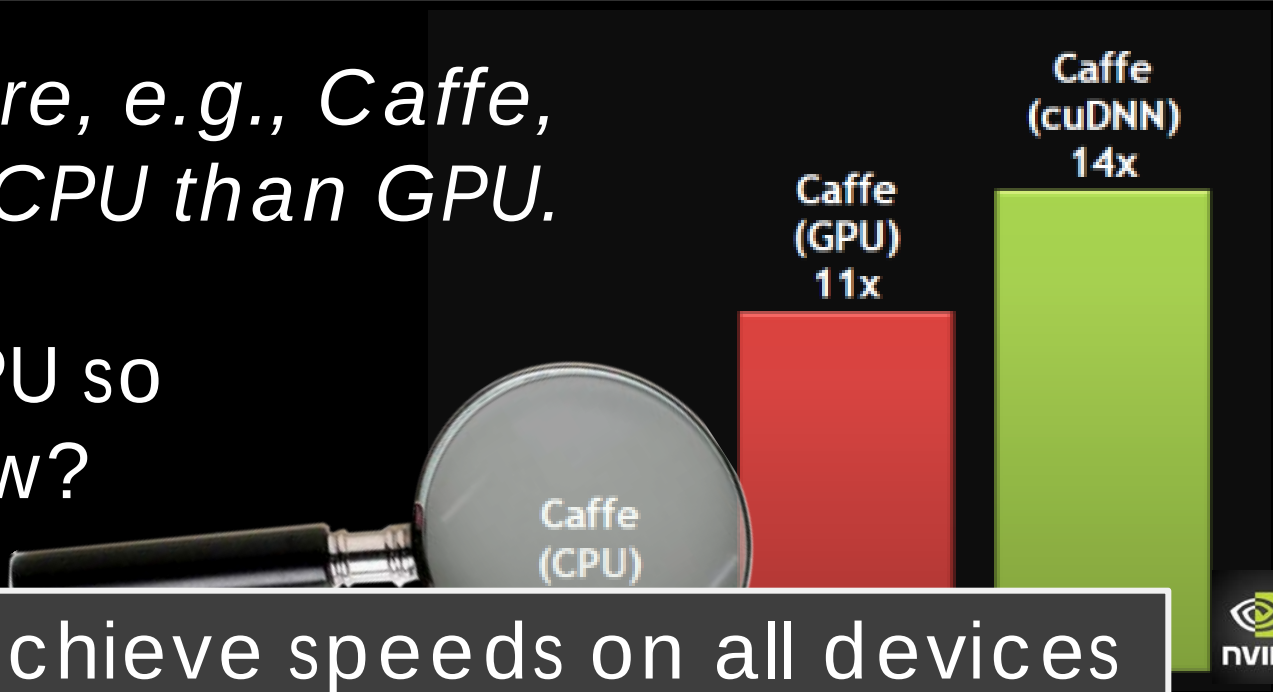
1.2TFlops

Not a 10x gap? Can we close this?

CPU or GPU?

Existing software, e.g., Caffe, 10x slower on CPU than GPU.

Why is the CPU so Relatively slow?



Goal: Achieve speeds on all devices proportional to their FLOPS

➔ Then use CPU + GPU together!

0.7TFlops

1.2TFlops

Not a 10x gap? Can we close this?

github.com/HazyResearch/CaffeConTroll

4 shallow ideas
described in 4 pages

4 Simple Ideas

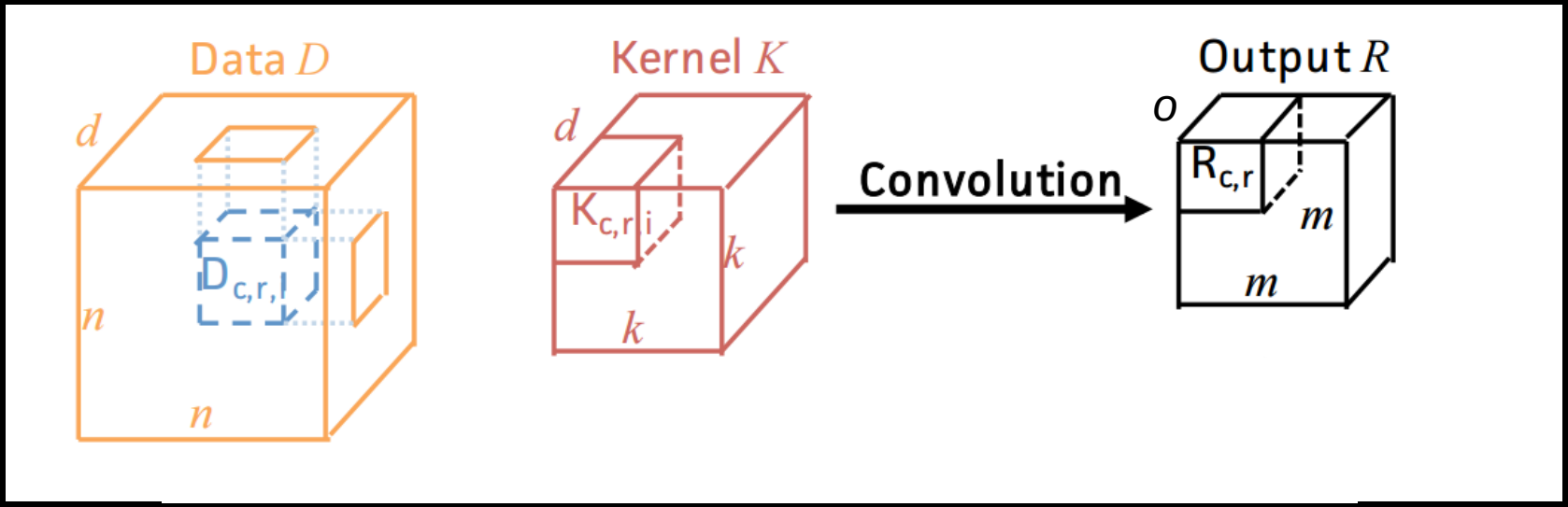
1. Understanding “Lowering”

2. Fusion of Lowering and GEMM

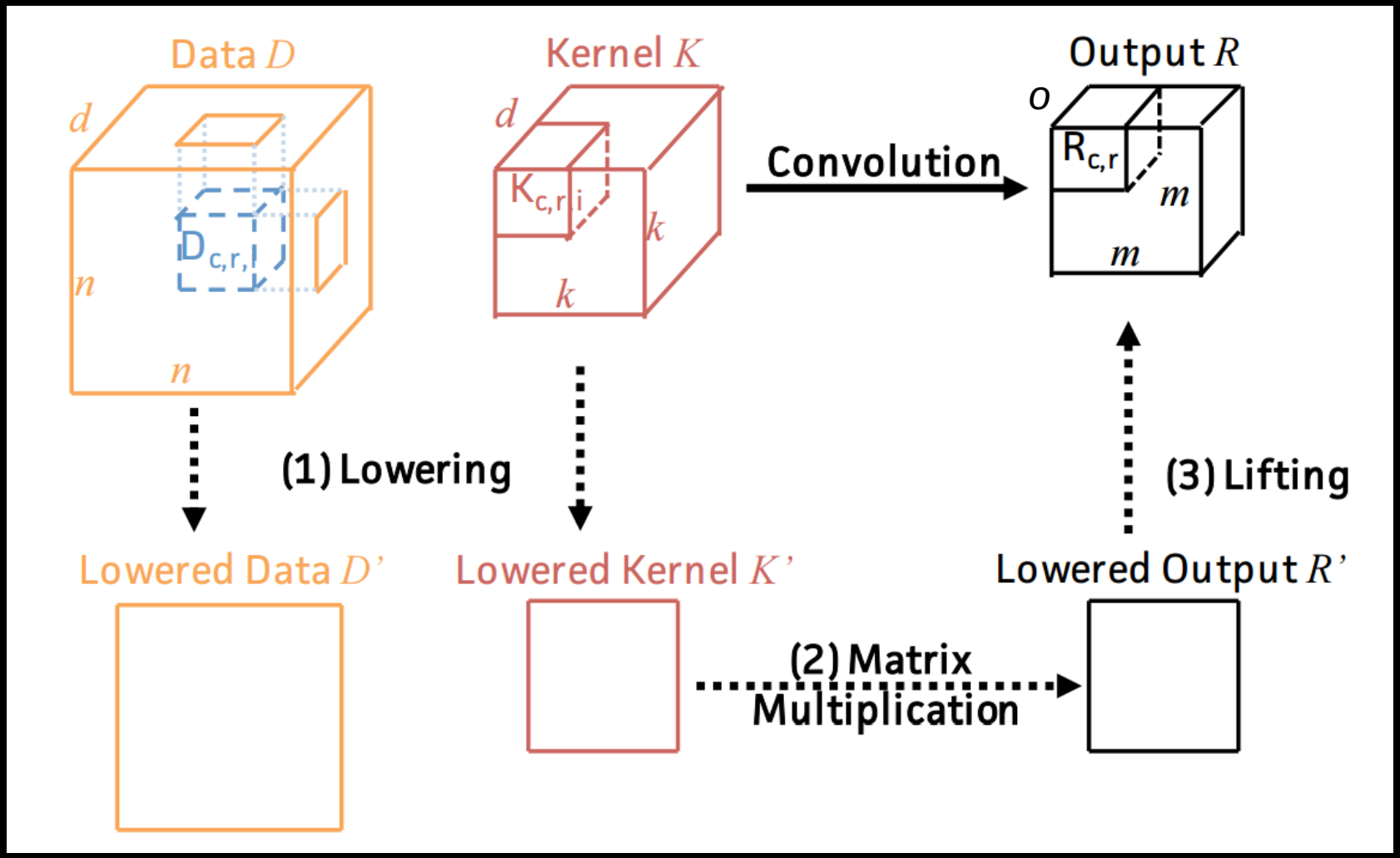
3. Parallel Batching, Blocking, SIMD

4. FLOP-Proportional Scheduling
   → Use both CPU + GPU for further speedups

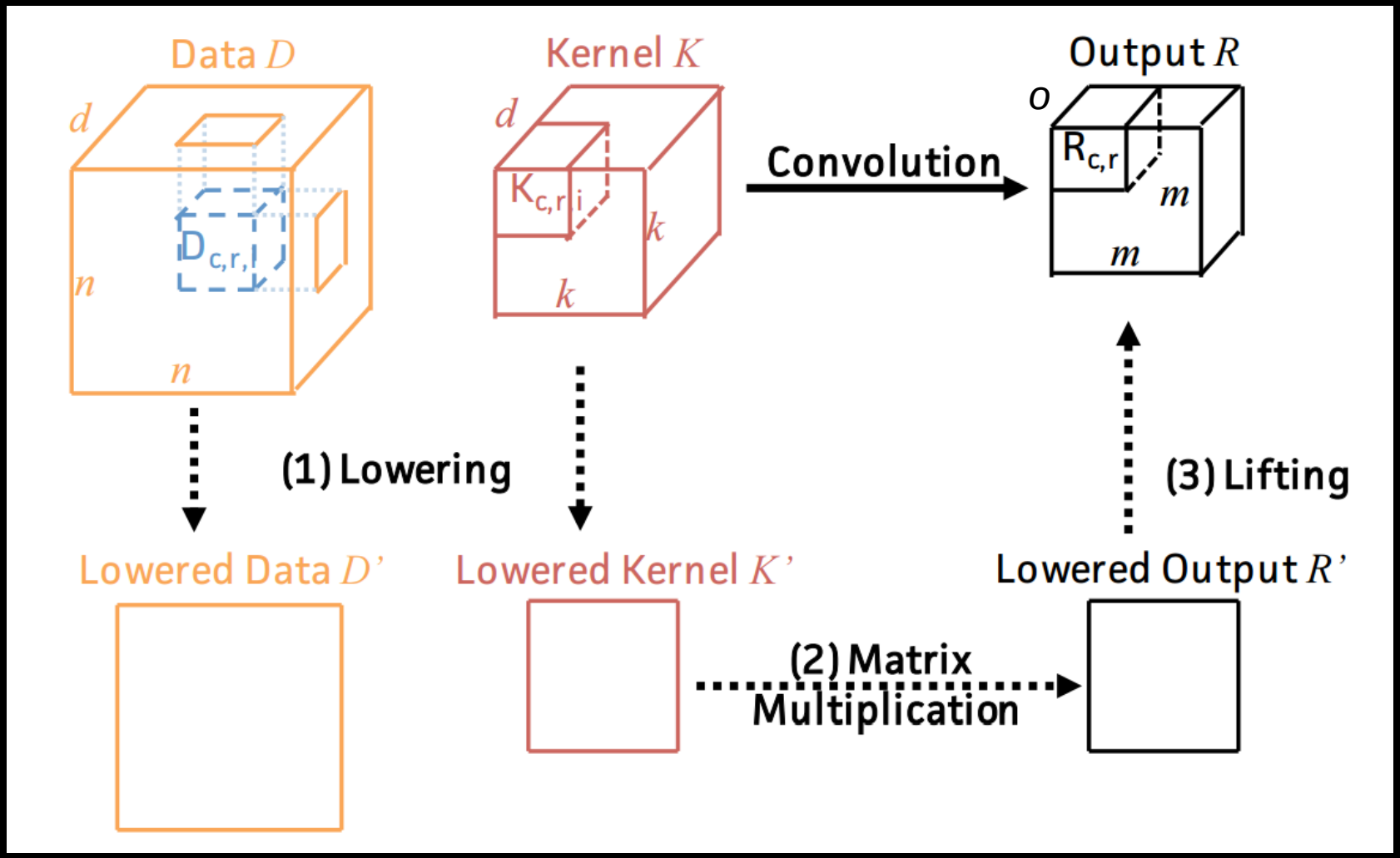
Lowering: Tensors to Matrix Multiply



Lowering: Tensors to Matrix Multiply

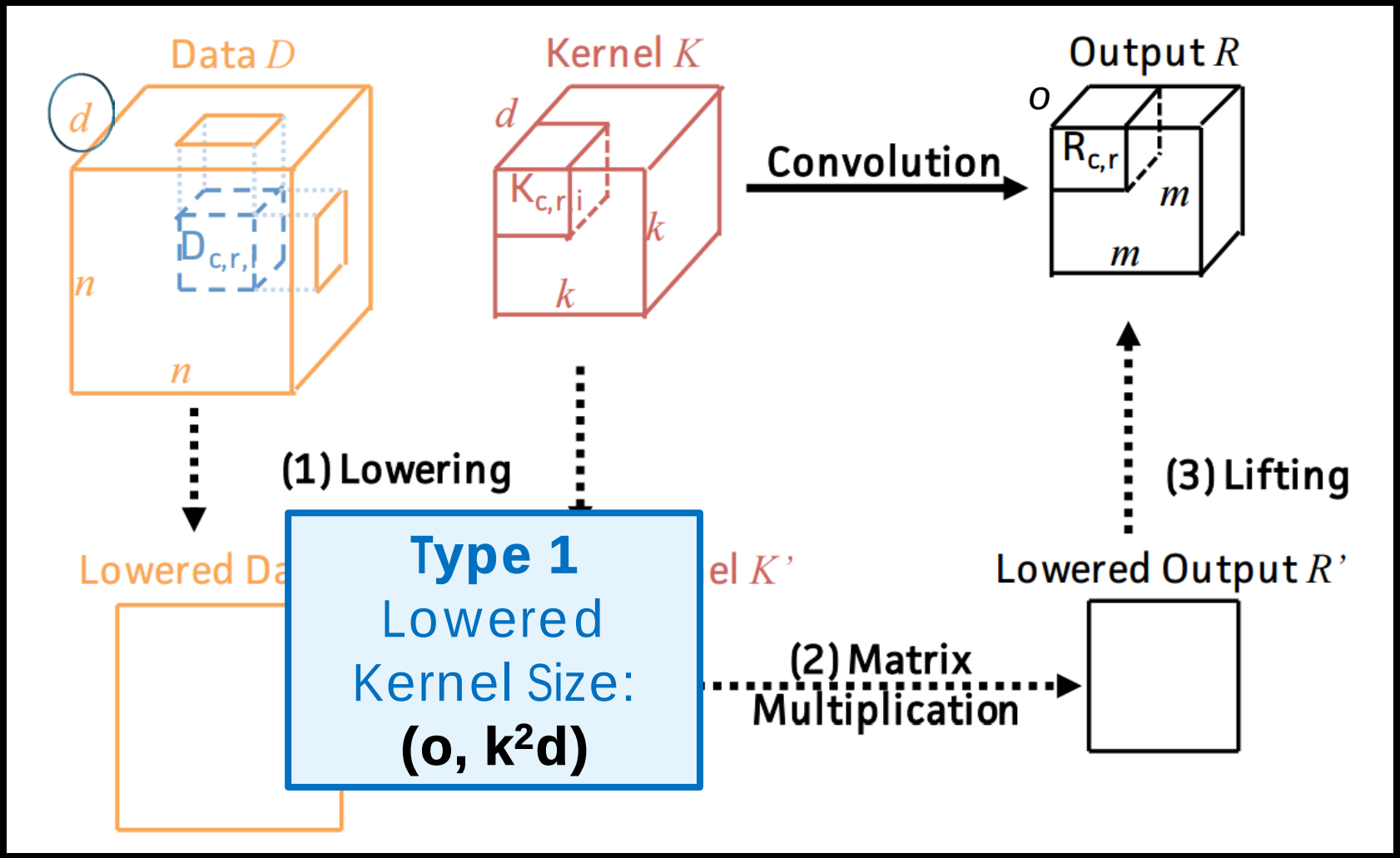


Lowering: Tensors to Matrix Multiply



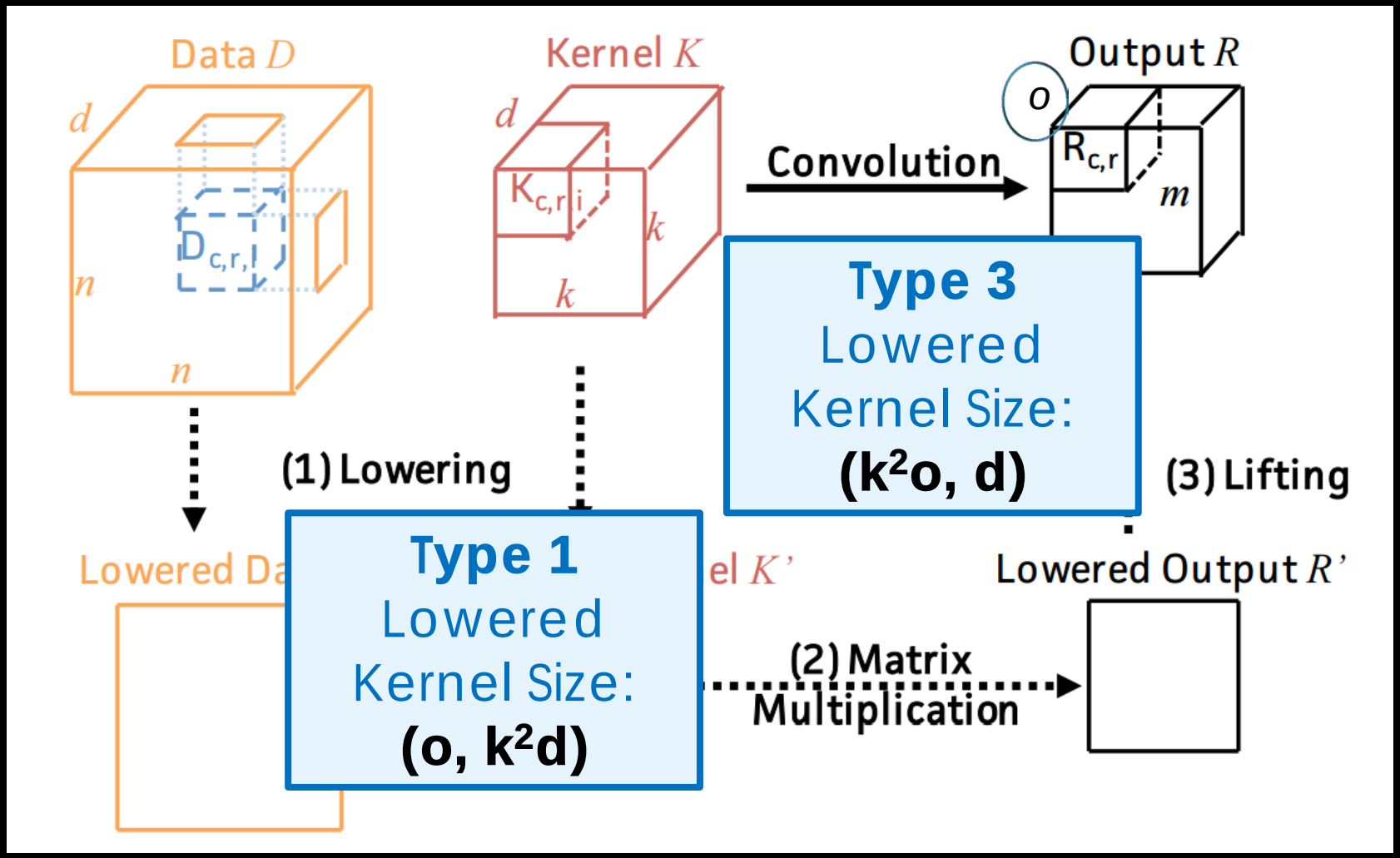
Shallow Idea 1. ***3 ways:*** *Replicate on lowering, Replicate lifting, or a little of both.*

Lowering: Tensors to Matrix Multiply



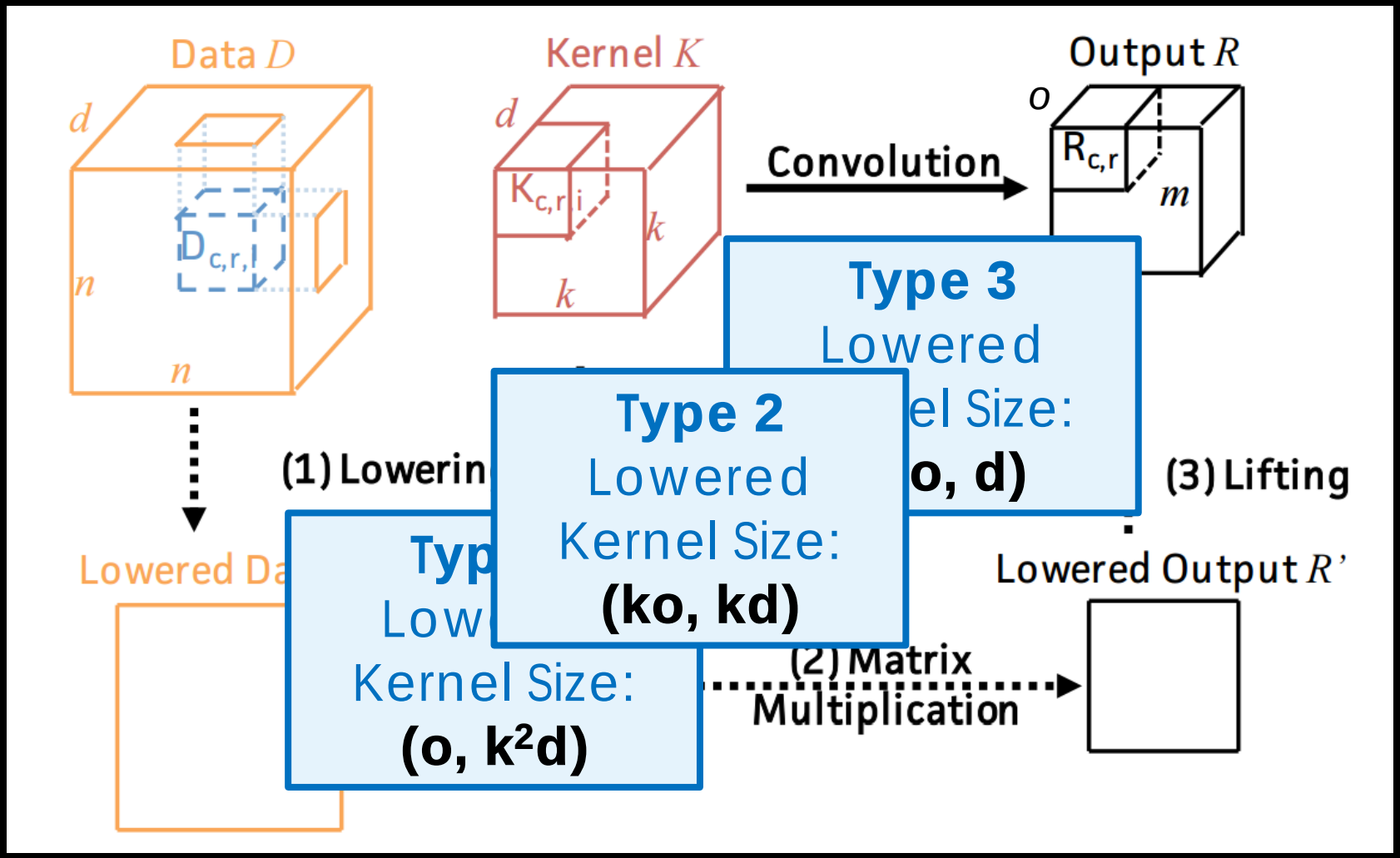
Shallow Idea 1. ***3 ways:*** *Replicate on lowering, Replicate lifting, or a little of both.*

Lowering: Tensors to Matrix Multiply



Shallow Idea 1. ***3 ways:*** *Replicate on lowering, Replicate lifting, or a little of both.*

Lowering: Tensors to Matrix Multiply

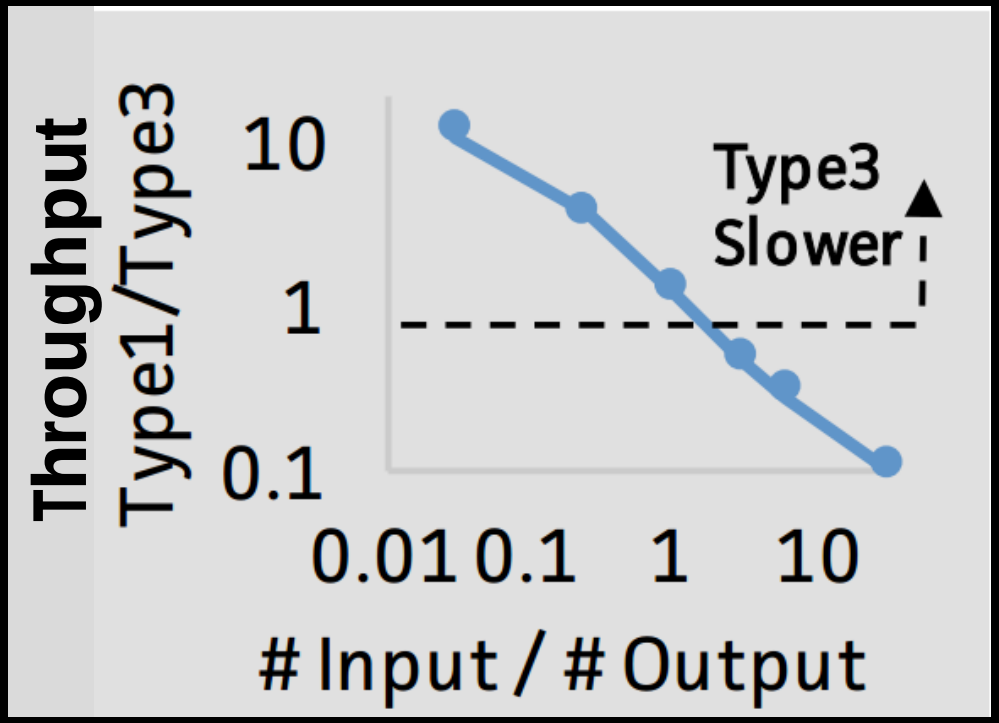


Shallow Idea 1. ***3 ways:*** *Replicate on lowering, Replicate lifting, or a little of both.*

3 Types of Lowering

Replicating Input (Type 1) is faster than replicating output (Type 3) when

#Input Channels < #Output Channels



Most conv layers *increase* depth

Replicating **Input** is usually fastest

3 Types of Lowering

Replicating Input (Type 1) is faster than replicating output (Type 3) when

#Input Channels < #Output Channels

Shallow idea 2:

Preliminary results also show 60% speedup by fusing lowering and GEMM

CPU Speedup: Batching

If the amount of data in GEMM call is too small, BLAS is not at peak FLOPS.

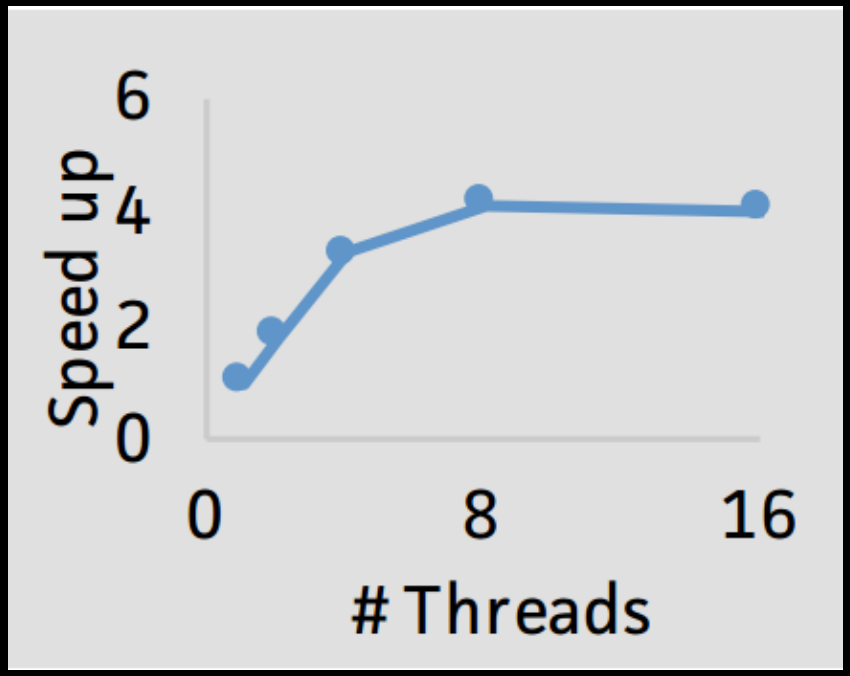
***Shallow idea 3**: Batch more data to give a chance to effectively **block** in GEMM ("fill" the L2 and L3 of all cores), and lower batches in parallel*

➔ *Not always possible to batch on GPU*

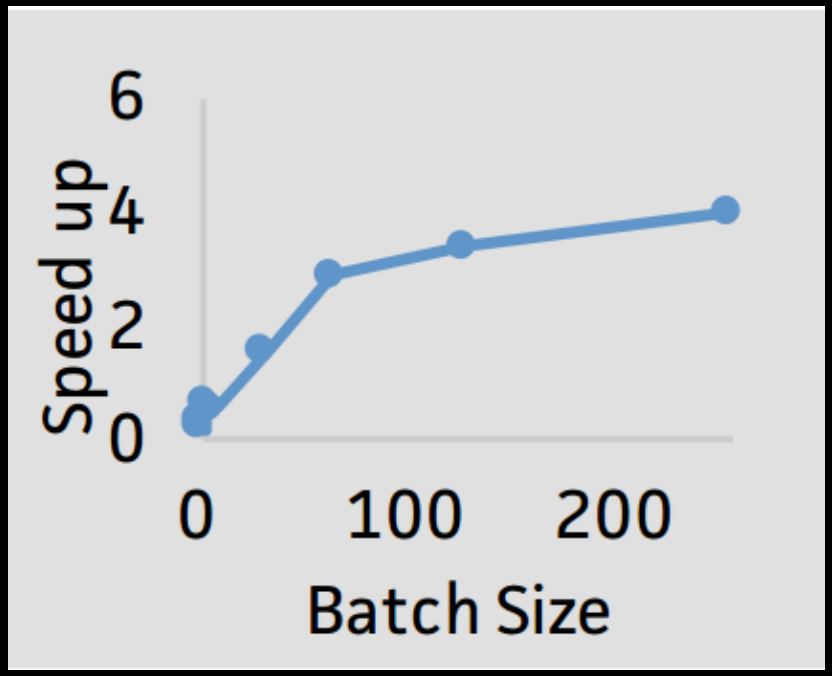
CPU Speedup: Batching

If the amount of data is small, BLAS is not CPU bound.

Effect on more threads and batch size on CPU GEMM kernel:



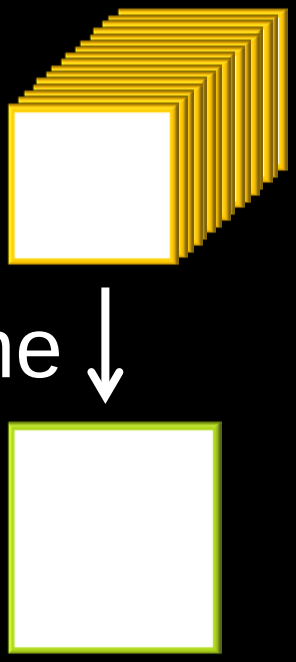
Batch size 256



8 Threads

CPU Speedup: Parallel Batch Partitions

Batching -- Caffe



Lowering

- 1 image at a time

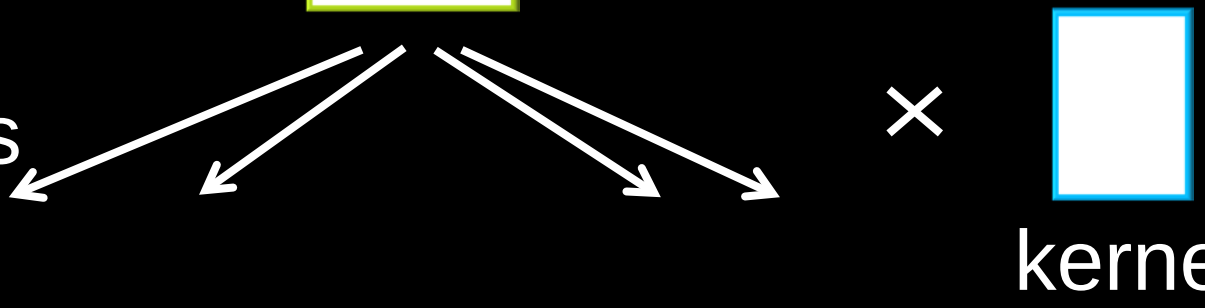
Batching -- Caffe

Lowering

- 1 image at a time

GEMM

- All cores



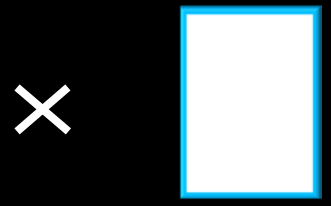
Batching -- Caffe

Lowering

- 1 image at a time

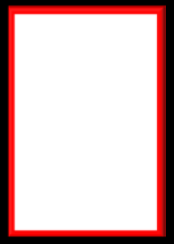
GEMM

- All cores



kernel

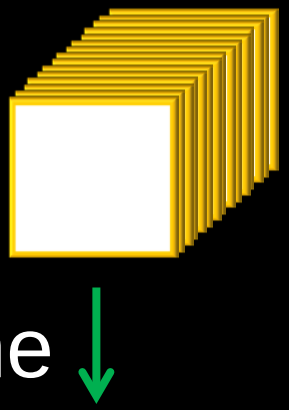
Output



Batching -- Caffe

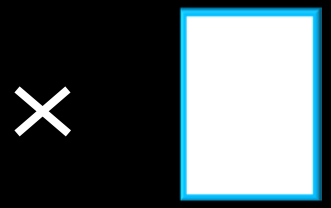
Lowering

- 1 image at a time



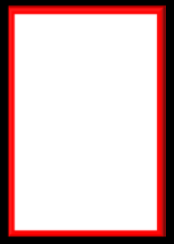
GEMM

- All cores

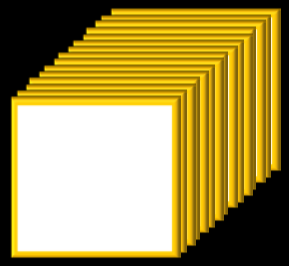


kernel

Output



Batching -- Caffe

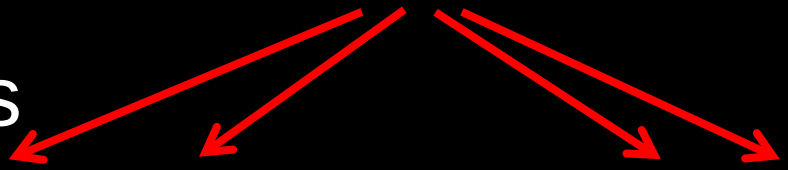


Lowering

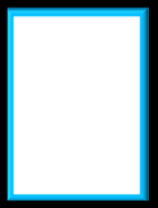
- 1 image at a time

GEMM

- All cores

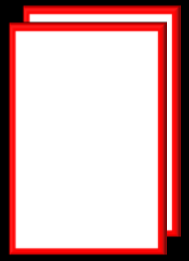


×



kernel

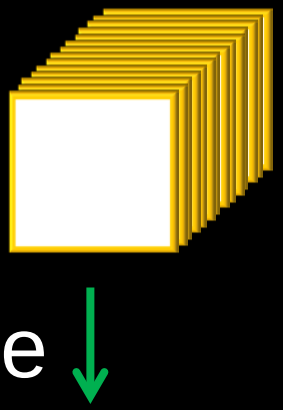
Output



Batching -- Caffe

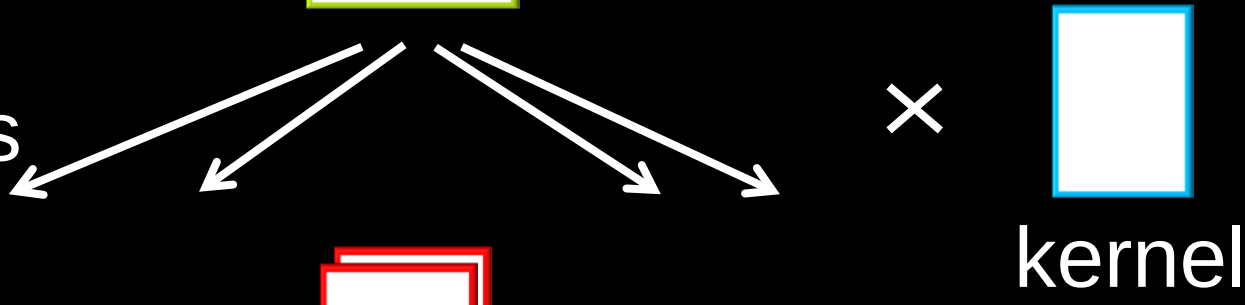
Lowering

- 1 image at a time



GEMM

- All cores

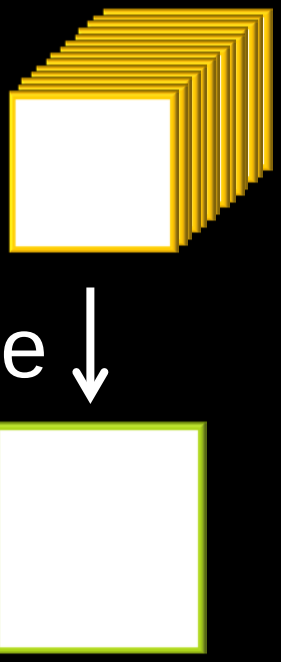


Output

Batching -- Caffe

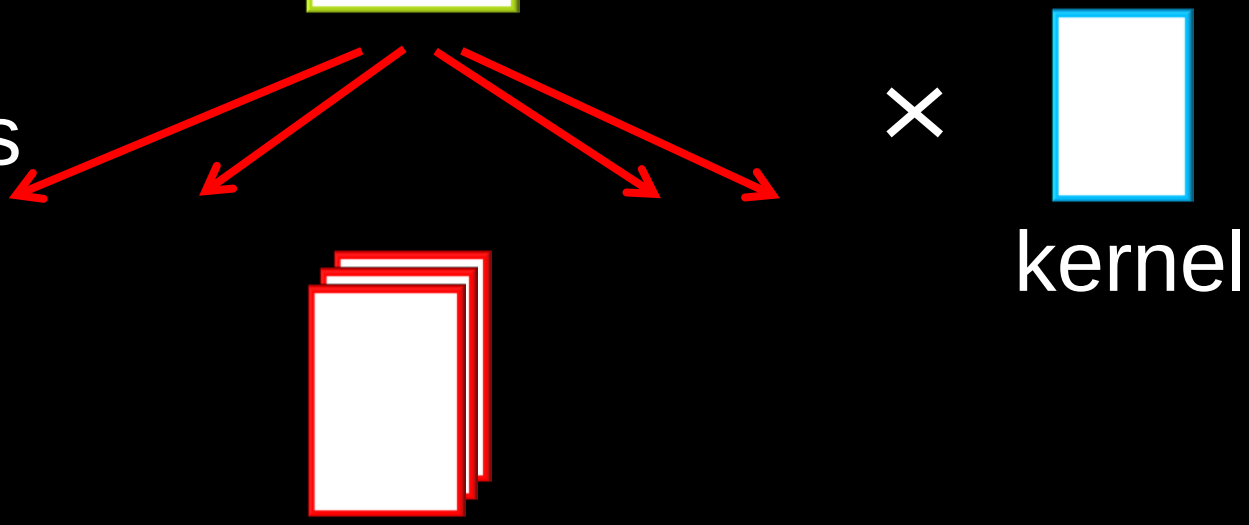
Lowering

- 1 image at a time



GEMM

- All cores



Output

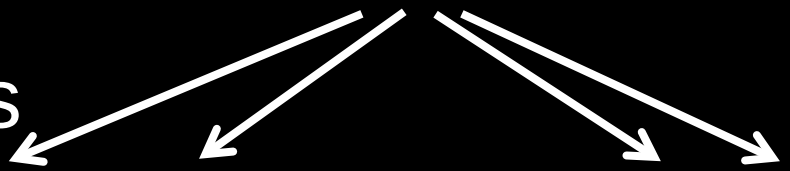
Batching -- Caffe

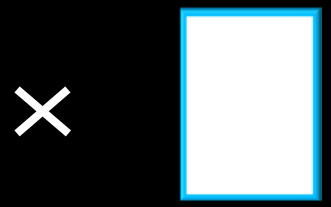
Lowering

- 1 image at a time

GEMM

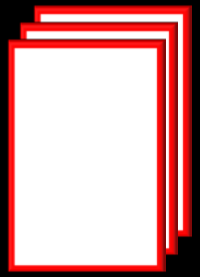
- All cores





kernel

Output



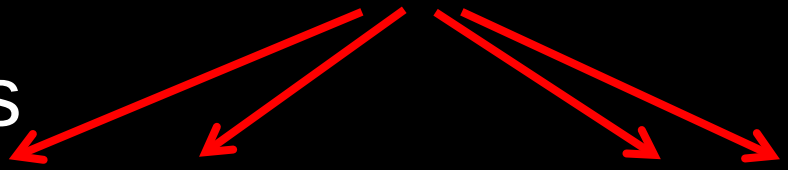
Batching -- Caffe

Lowering

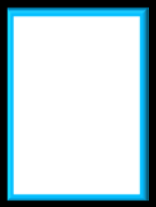
- 1 image at a time

GEMM

- All cores

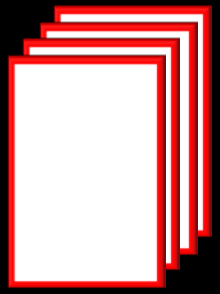


×

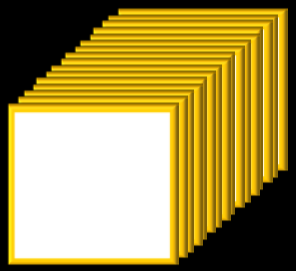


kernel

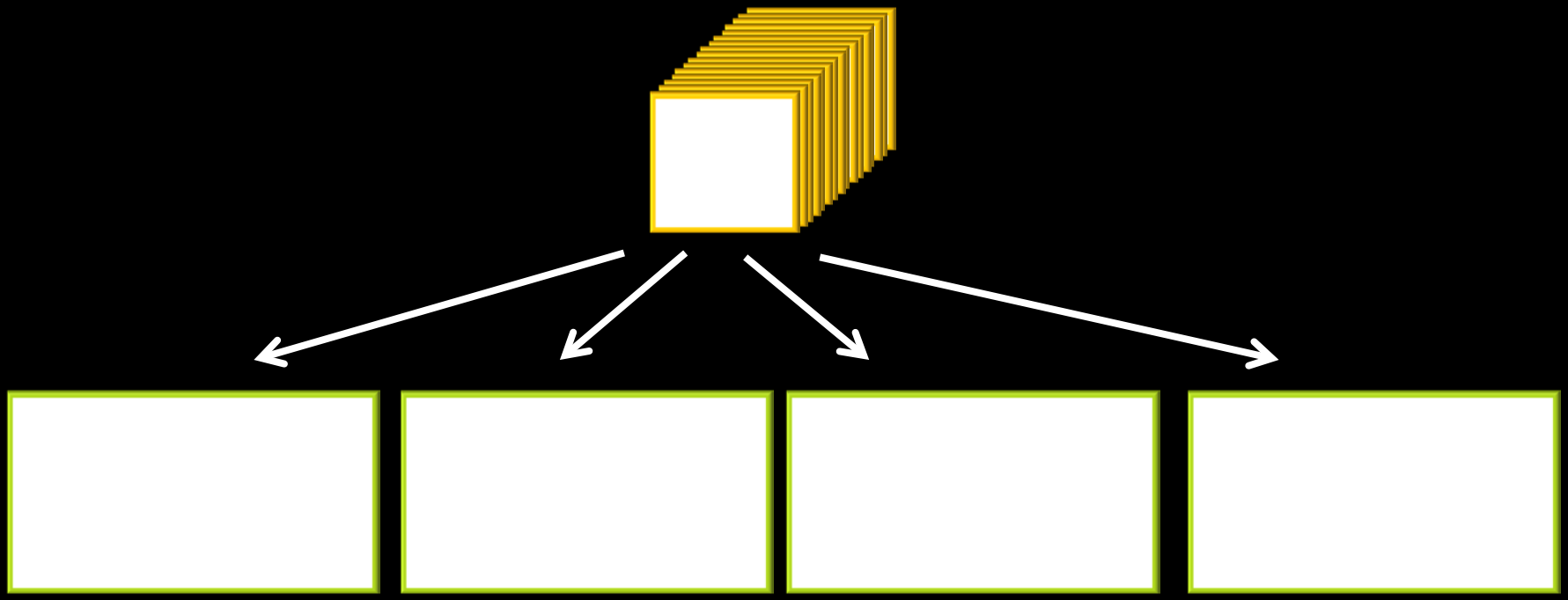
Output



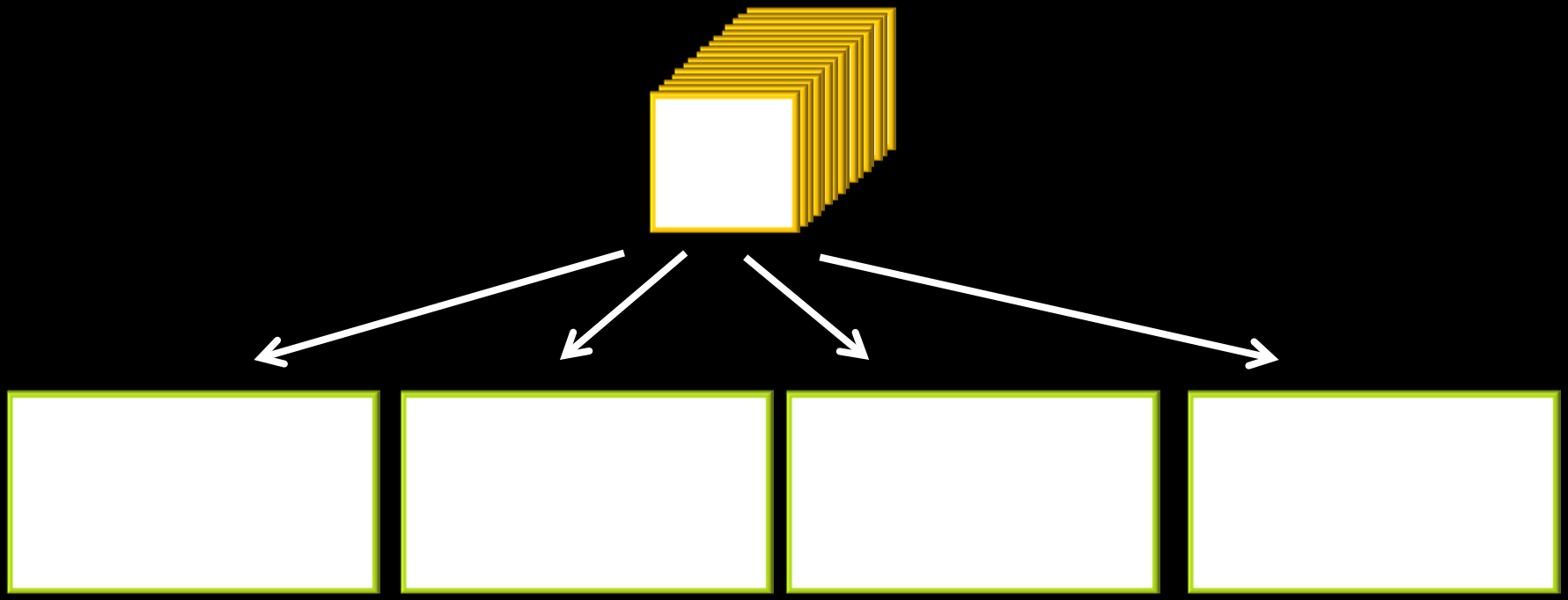
Batching -- CcT



Batching -- CcT

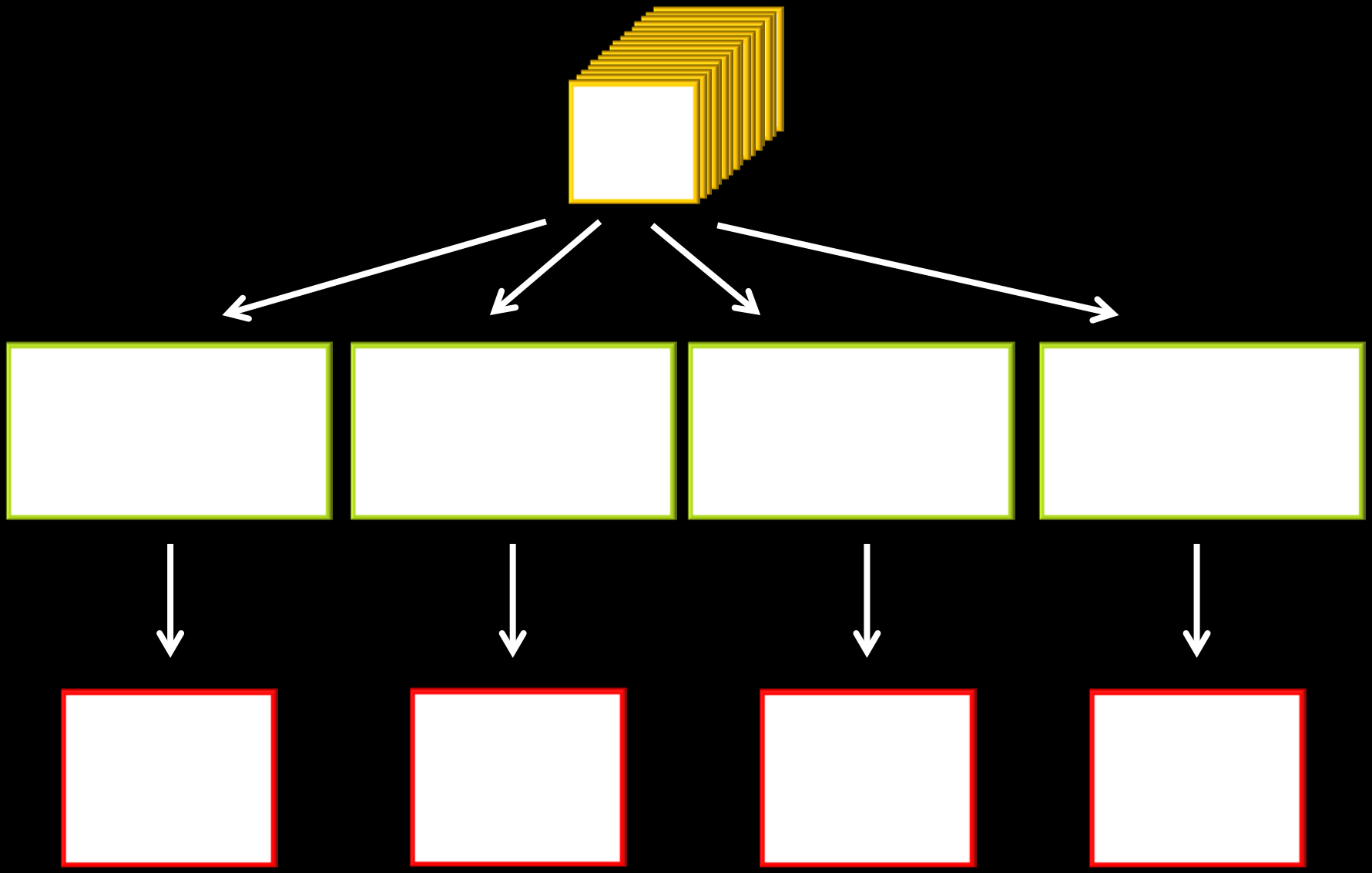


Batching -- CcT

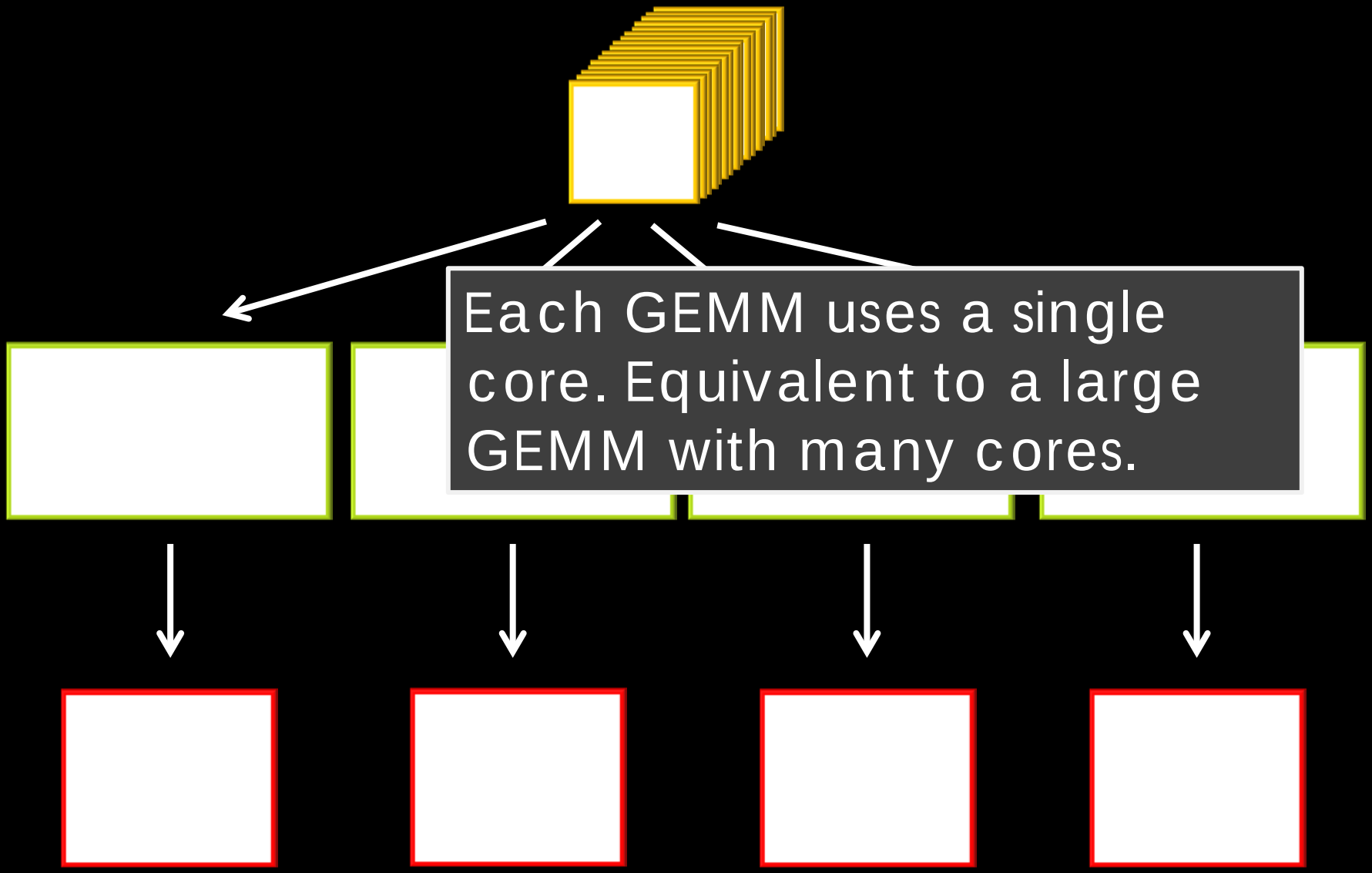


- **Partition** data and lower in parallel
- Use **batching** within each partition
  - All matrices are larger, enabling blocking optimizations and making GEMM CPU bound

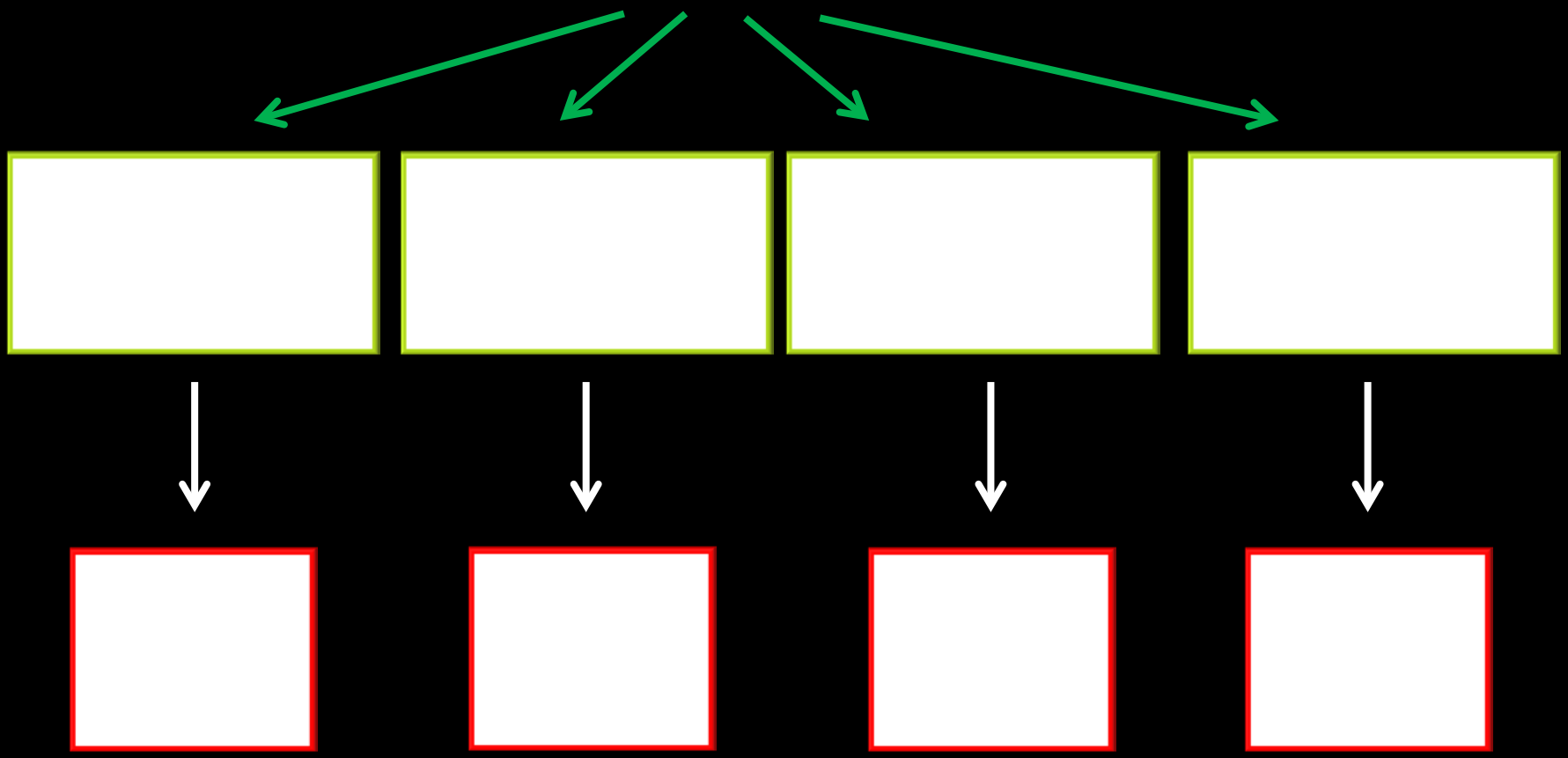
Batching -- CcT



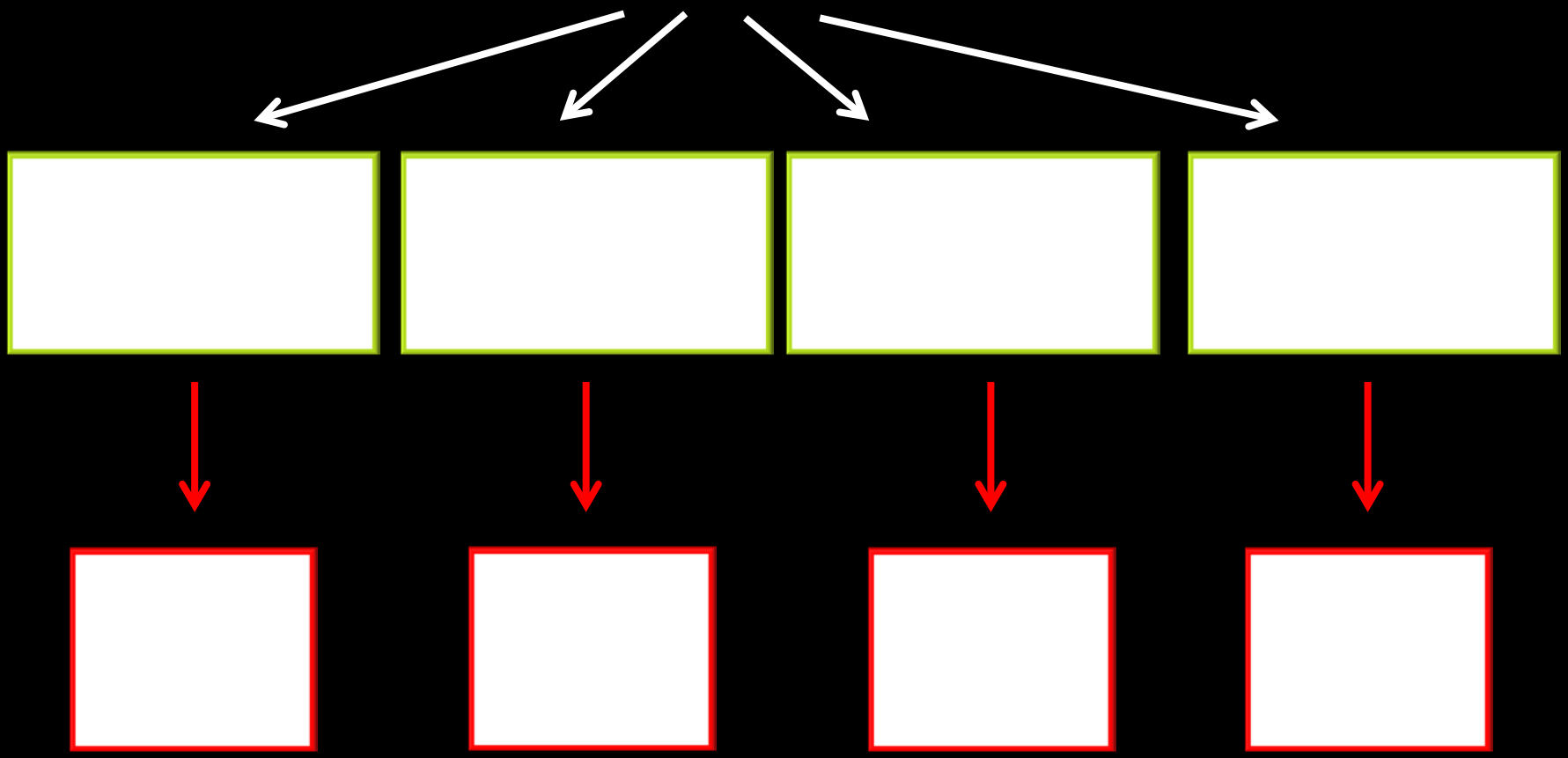
Batching -- CcT



Batching -- CcT

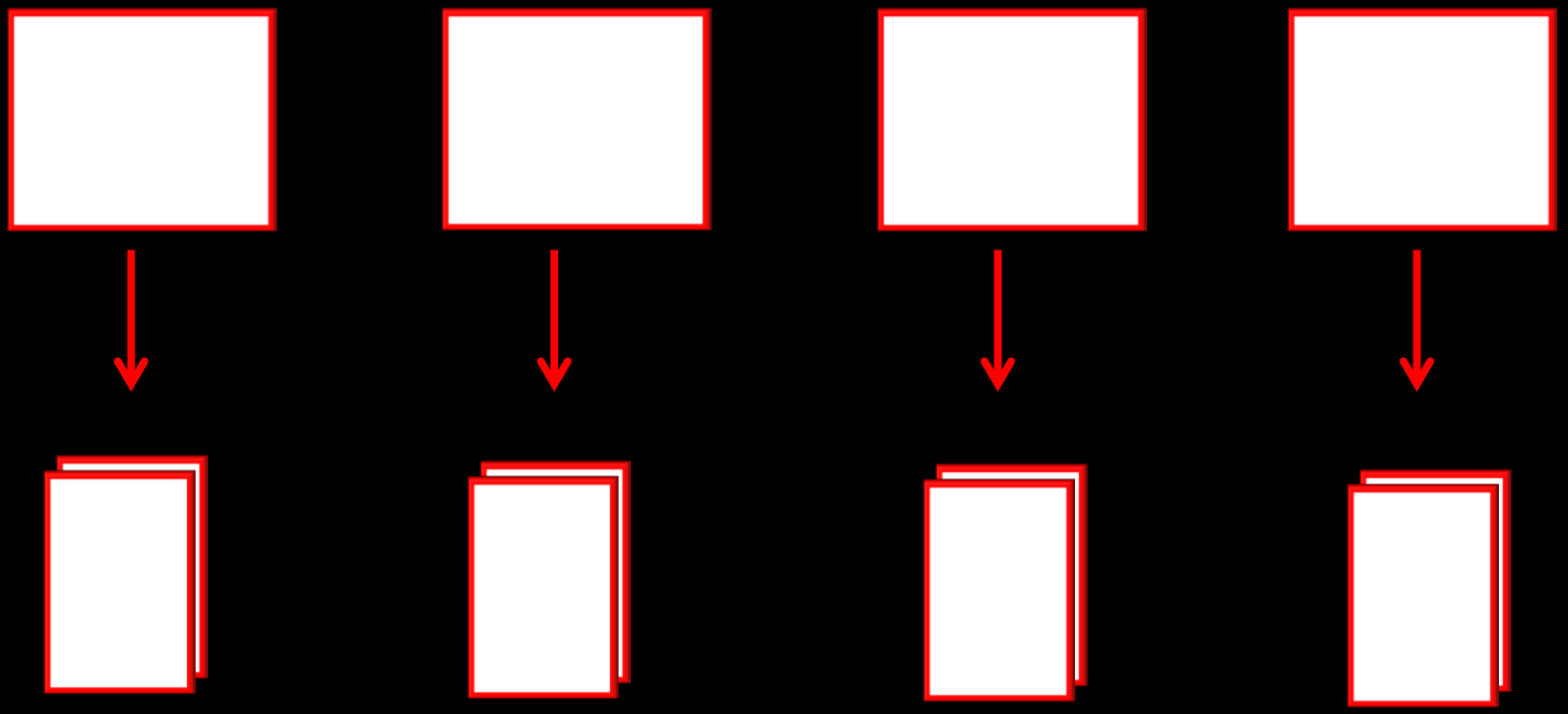


Batching -- CcT

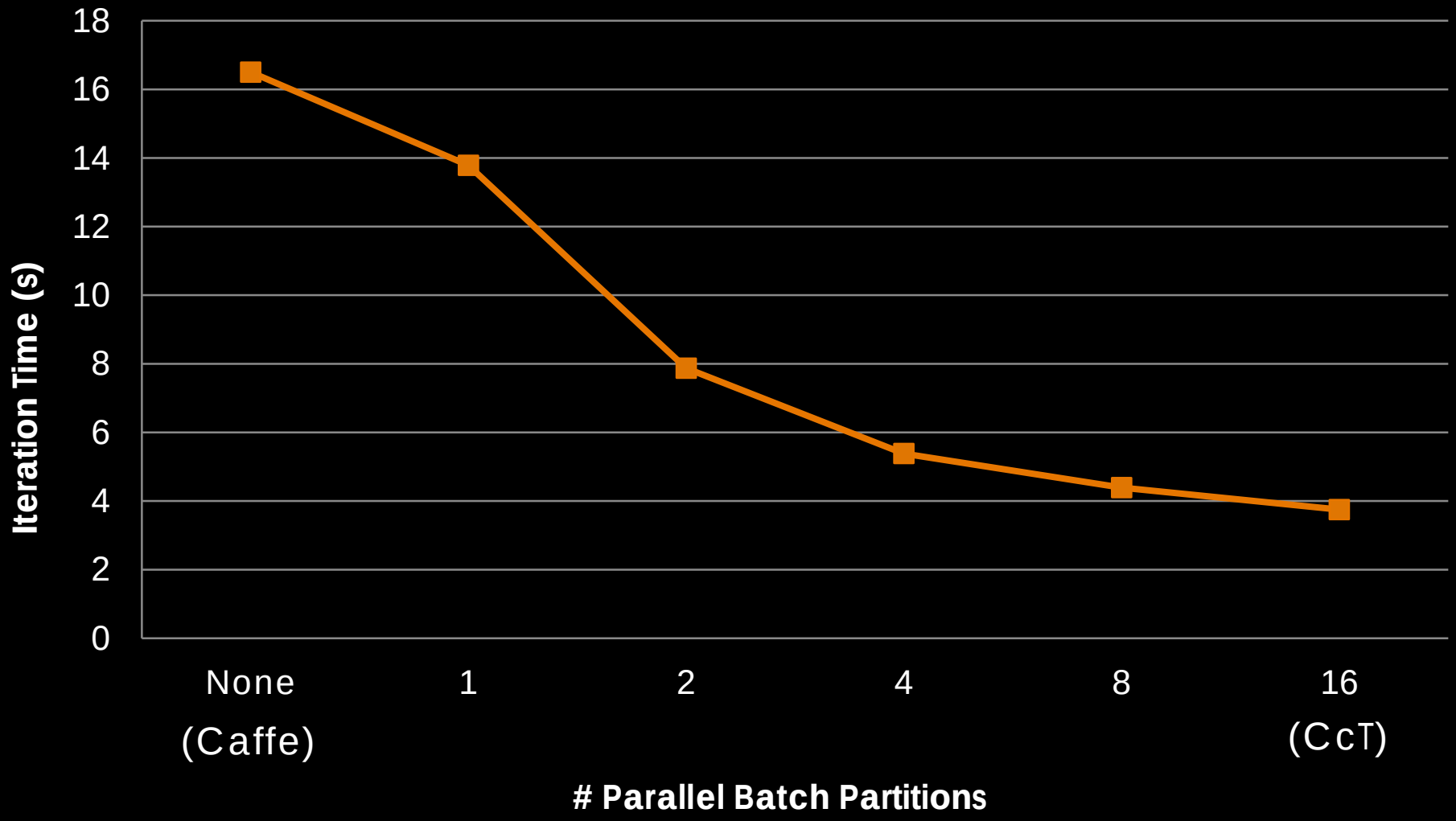


Batching -- CcT

Final step to remap the output

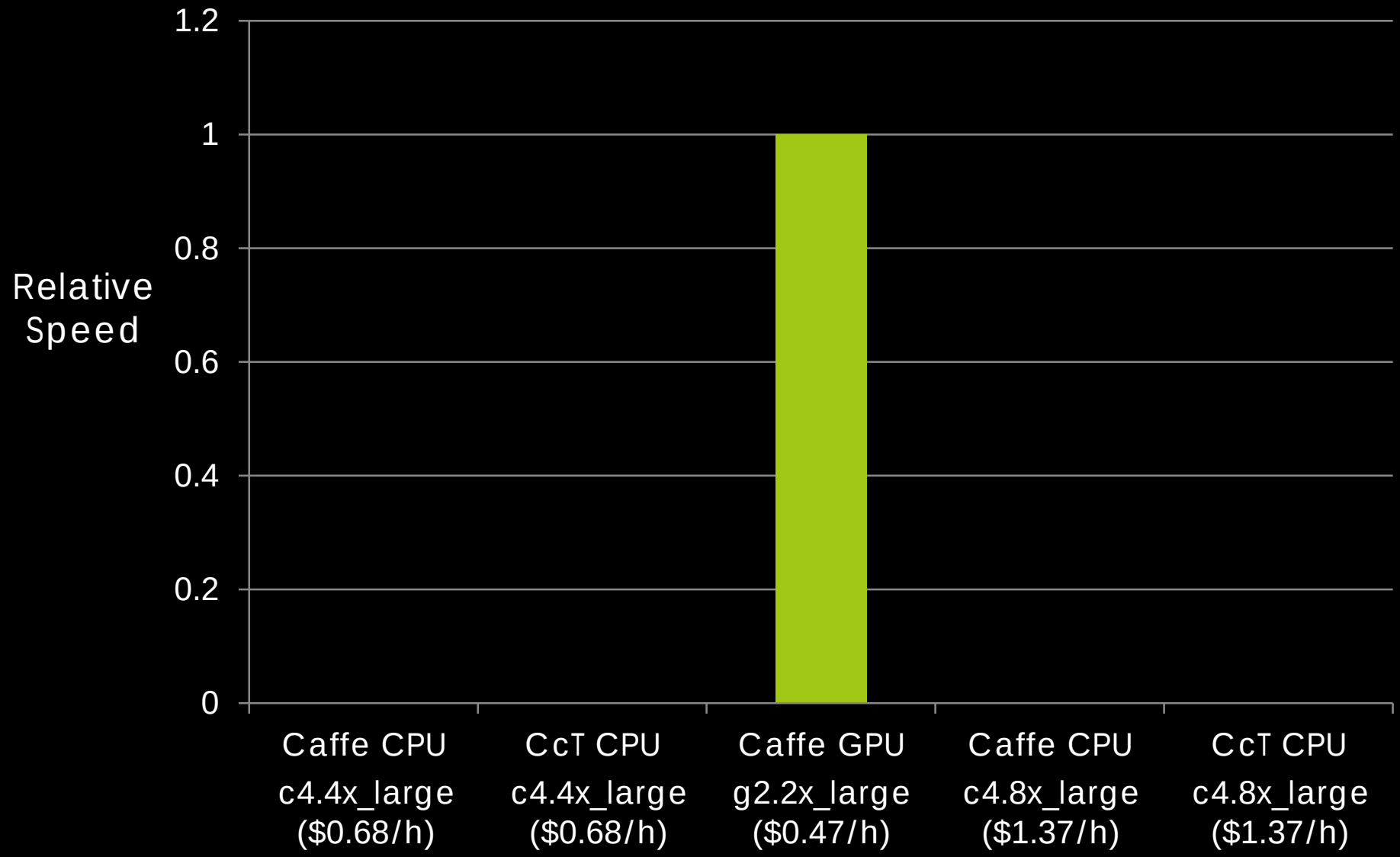


CPU Batching Speedup (full AlexNet)

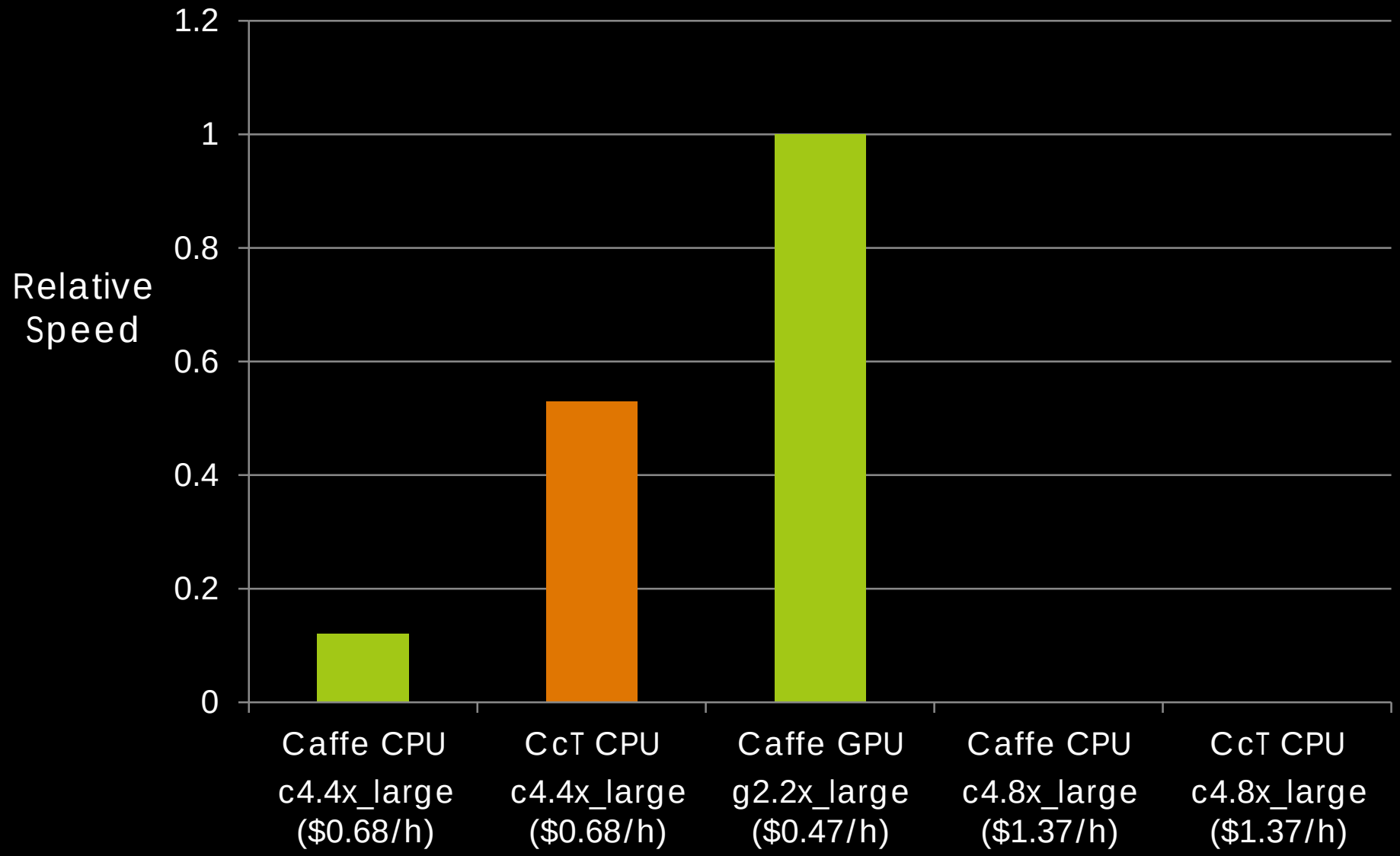


EC2 c4.4xlarge instance ($0.68/hour), end-to-end *"AlexNet", batch size 256*

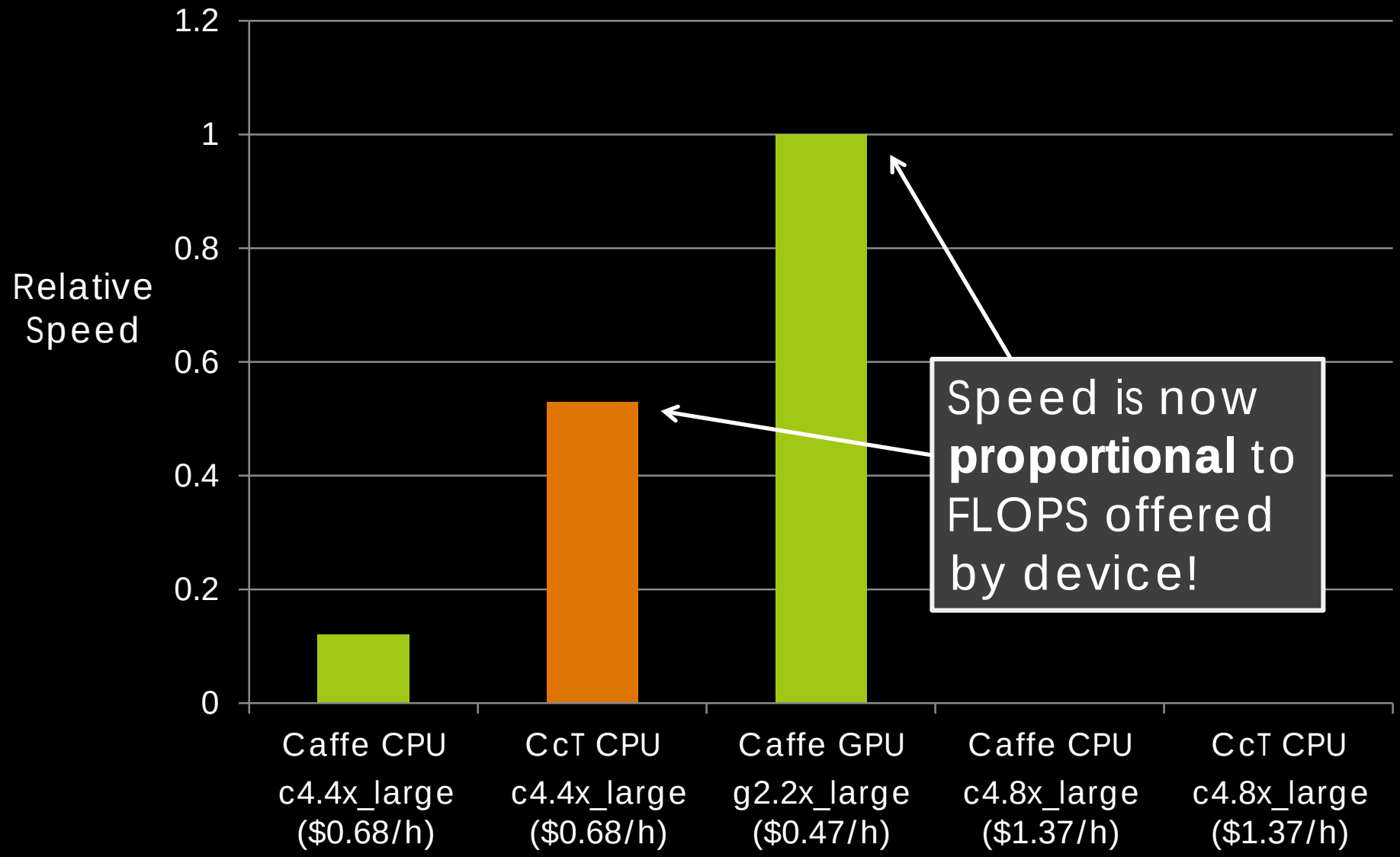
CPU Batching Speedup (full AlexNet)



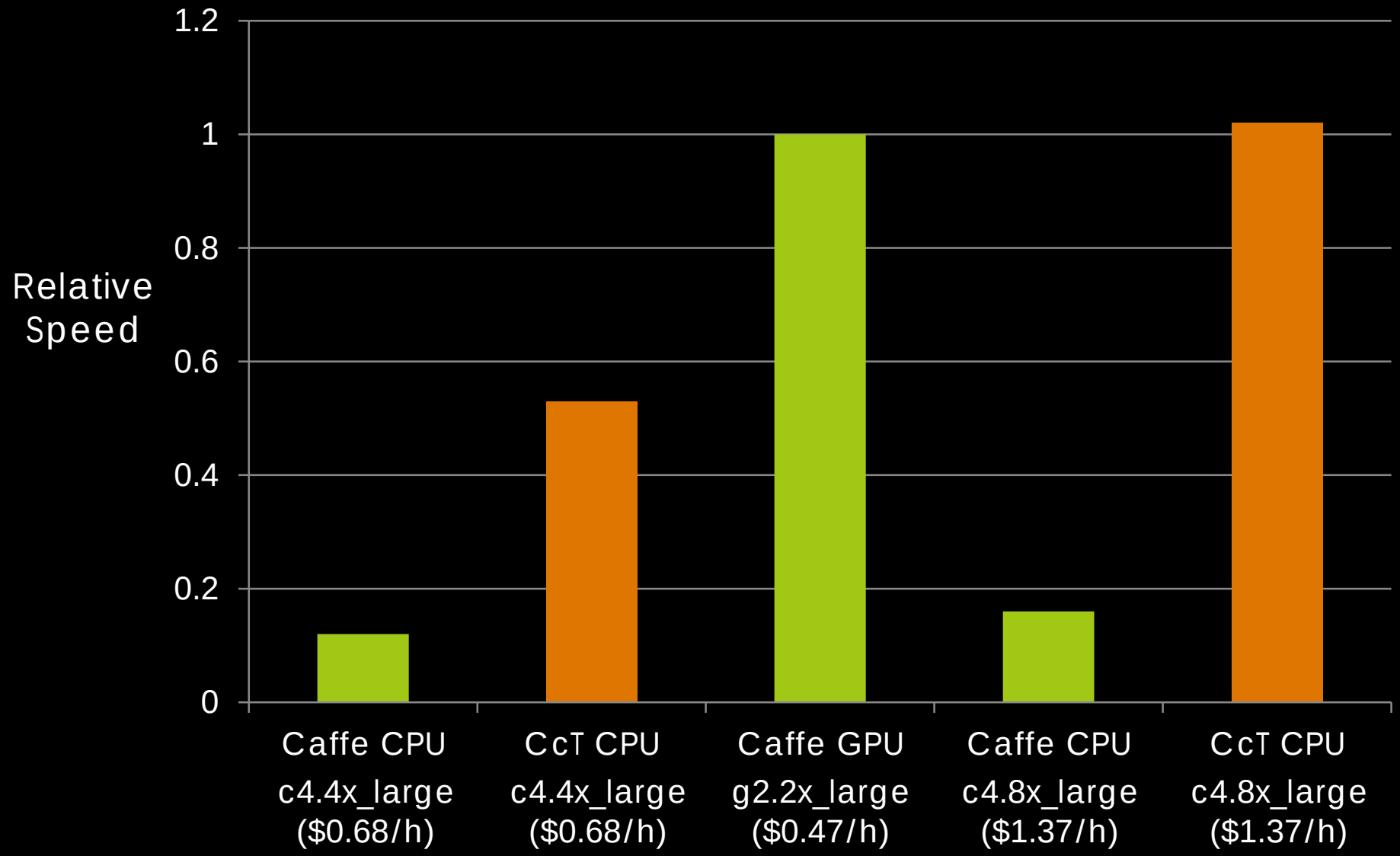
CPU Batching Speedup (full AlexNet)



CPU Batching Speedup (full AlexNet)

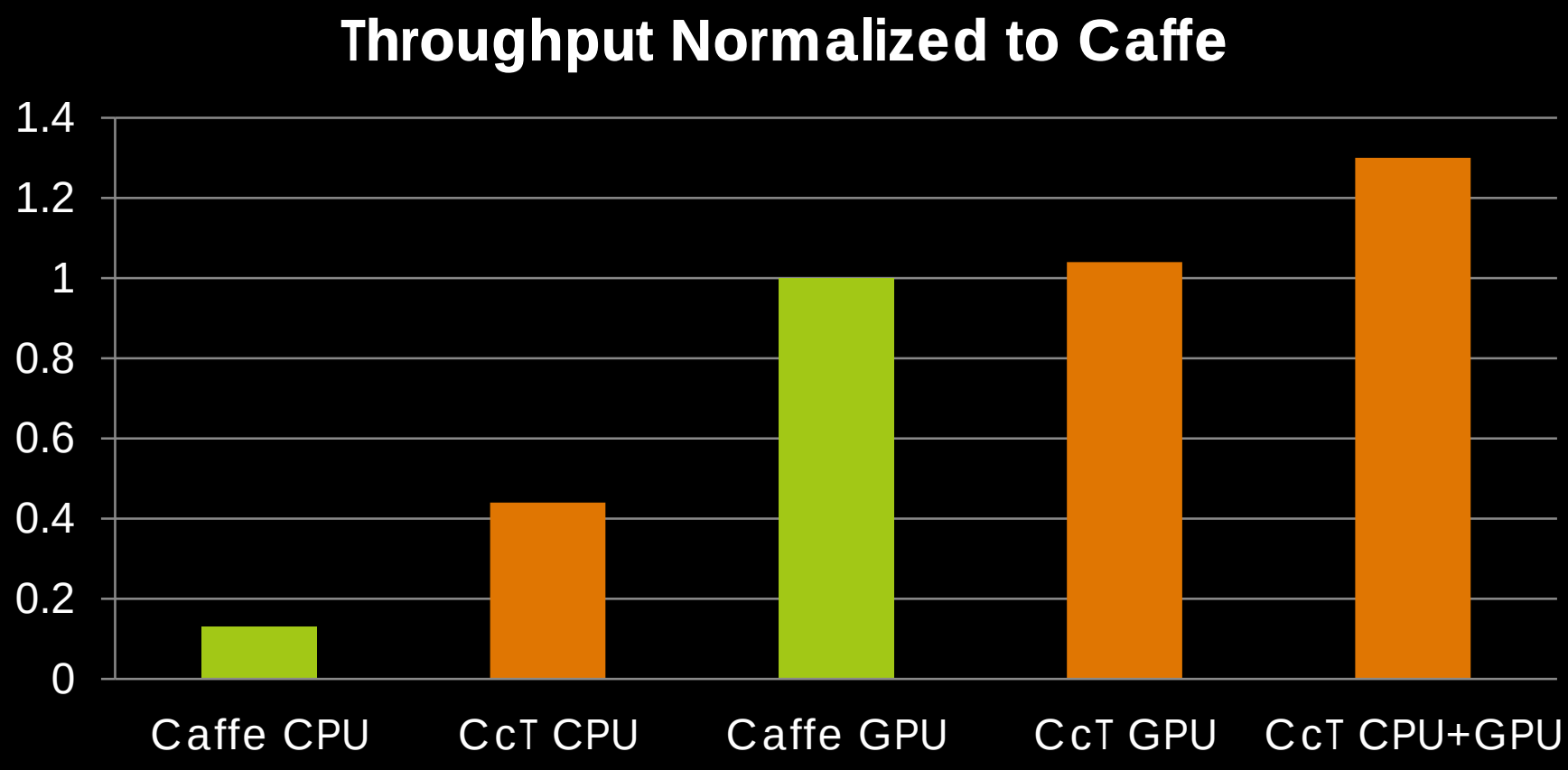


CPU Batching Speedup (full AlexNet)



CPU + GPU (Data Parallel)

Shallow idea 4: *FLOP Proportional Scheduling.*

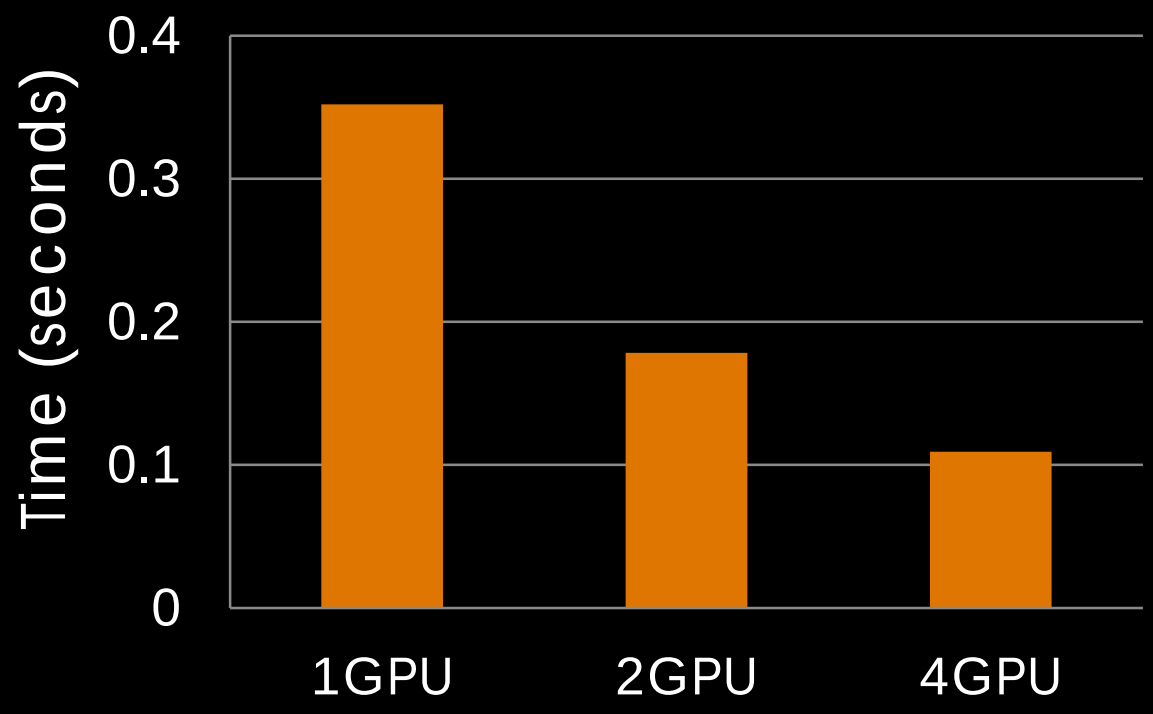


Run on EC2 g2.2xlarge instance ($0.47/hour) *for Layer 1 of popular "AlexNet"*

Multiple GPUs and Multiple Machines.

Flop Proportional Scheduling allows us to distribute the computation in a device-agnostic way.

AlexNet Conv1 Layer



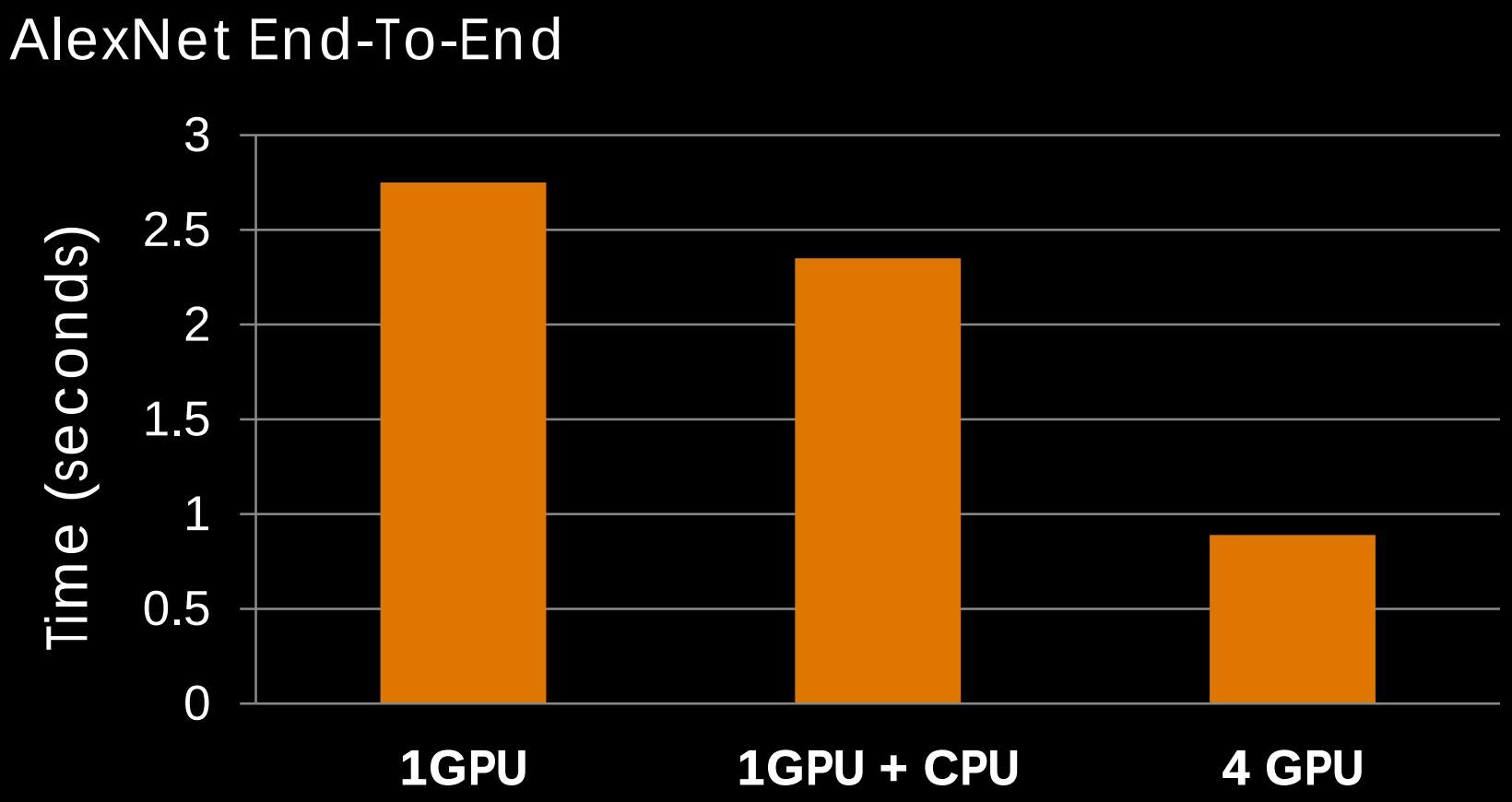
We have applied CcT to a single ***4-GPU EC2 instance*** *(announced last month!)*

Next we are working on a cluster of these instances!

Run on EC2 g2.8xlarge instance

Multiple GPUs and Multiple Machines.

Flop Proportional Scheduling allows us to distribute the computation in a device-agnostic way.



Run on EC2 g2.8xlarge instance

Trying CcT

- VMs (EC2 + Azure) available with CcT installed
- What's next?
  - Multiple Machines
  - New optimizations

github.com/HazyResearch/CaffeConTroll

Summary

- CPU + GPU can work together!
  - Close CPU gap
  - Operate both near peak FLOPS
- FLOP proportional scheduling
  - Next: Scale to distributed setting
- Questions?

github.com/HazyResearch/CaffeConTroll