Achieving Scalability in a k-NN multi-GPU Network Server with Centaur

Amir Watad
Technion

Alexander Libov
Amazon

Ohad Shacham
Yahoo! Labs

Edward Bortnikov
Yahoo! Labs

Mark Silberstein
Technion

“A leader is best when people barely know he exists”

Lao Tzu

CPU Accelerators

“A leader is best when people barely know he exists”

Lao Tzu

Today's Task

- Design a Network Server
- with Low Latency
- and High Throughput
- utilizing Many GPUs

Today's Takeaway

- Using a CPU to manage accelerators can severely limit their performance.

- There is a way to remove CPU from management role, allowing accelerators to show their full potential.

Agenda

Example Application: k-NN Network Server 🌐

CPU-Driven Server Design 👎🏽

Analysis 🔬

GPU Centric Design (Centaur) 👍🏽

Benchmarks 📊

Agenda

Example Application: k-NN Network Server

🌐

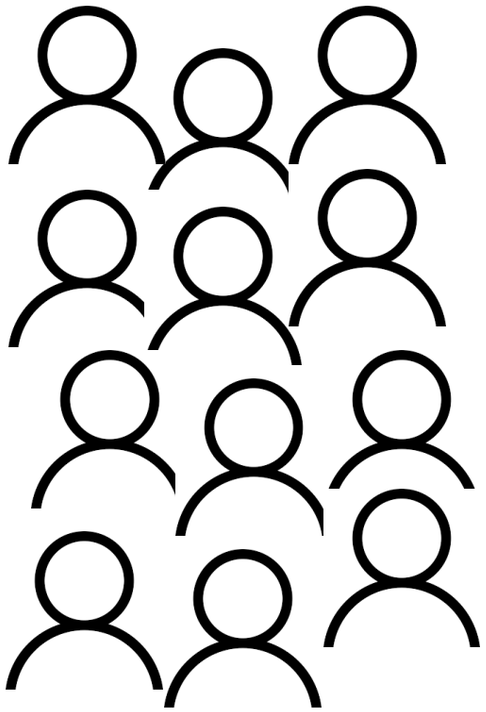
CPU-Driven Server Design 👎🏾

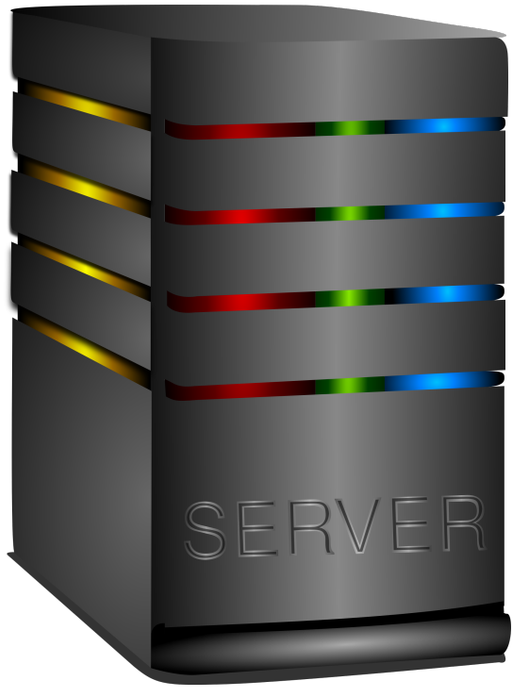
Analysis 🔬

GPU Centric Design (Centaur) 👍🏾

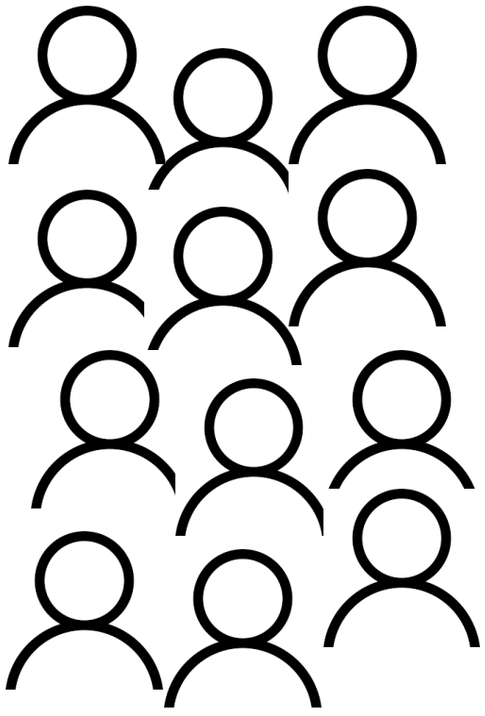
Benchmarks 📊

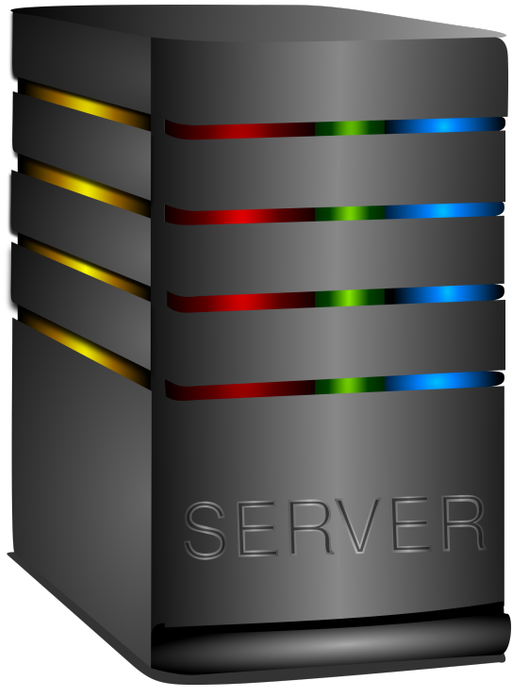
Example
k-NN Server





Example
k-NN Server

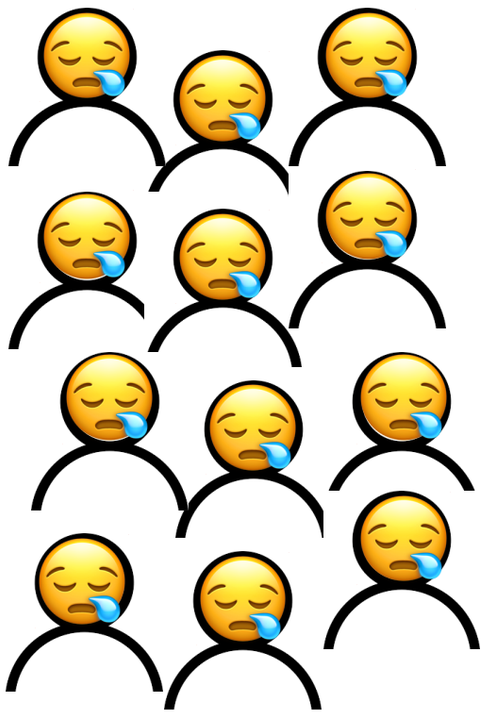


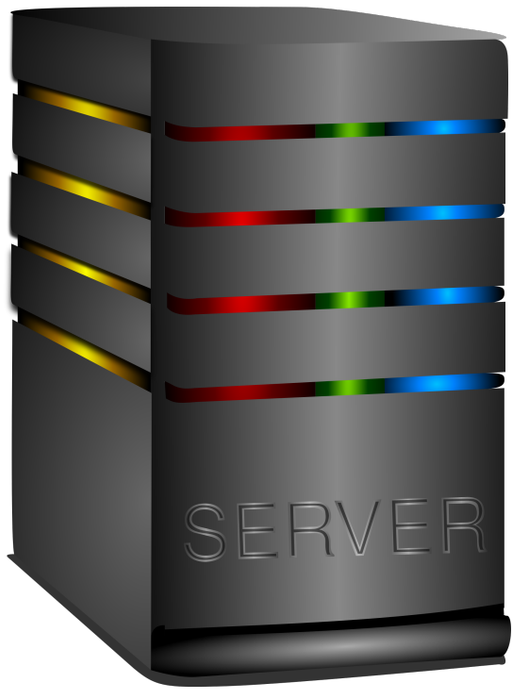


We need to serve all these folks at the same time

i.e. **High Throughput**

Example
k-NN Server



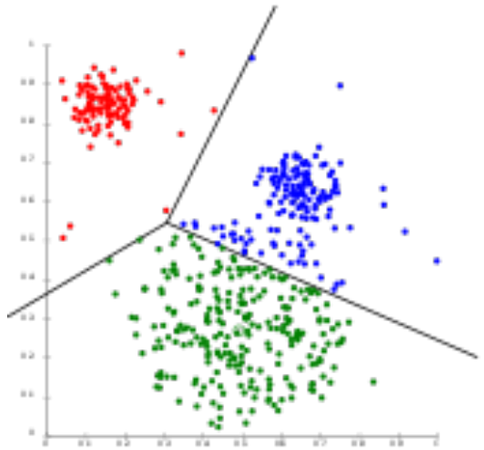


Users hate to wait.
We need to be low latency.
~1-2 msec for the backend.

Hierarchical k-NN

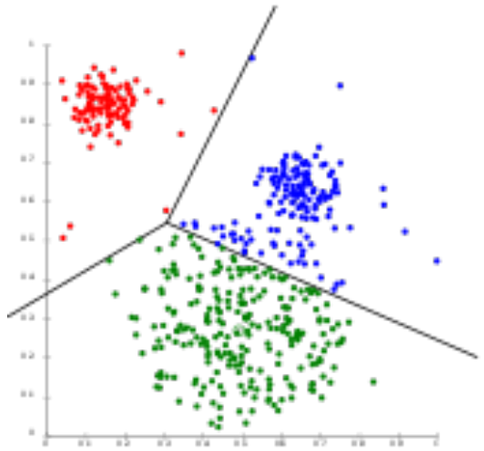
- A family of Algorithms
- Multi-Level Search
- Pros: Faster
- Cons: Approximate Results

Hierarchical k-NN



- Pre-split collection into clusters (“Albums”)
- **filter**: Find nearest W clusters.
- **search**: In each cluster, find nearest K pictures.
- **reduce**: Find closes K among the W * K.

Hierarchical k-NN



- Pre-split collection into clusters (“Albums”)
- **filter**: Find nearest W clusters.
- **search**: In each cluster, find nearest K pictures.
- **reduce**: Find closes K among the W * K.

We Need More Compute

We Need More Compute

No problem

We Need More Compute

No problem







Agenda

~~Example Application: k-NN Network Server~~ 🌐

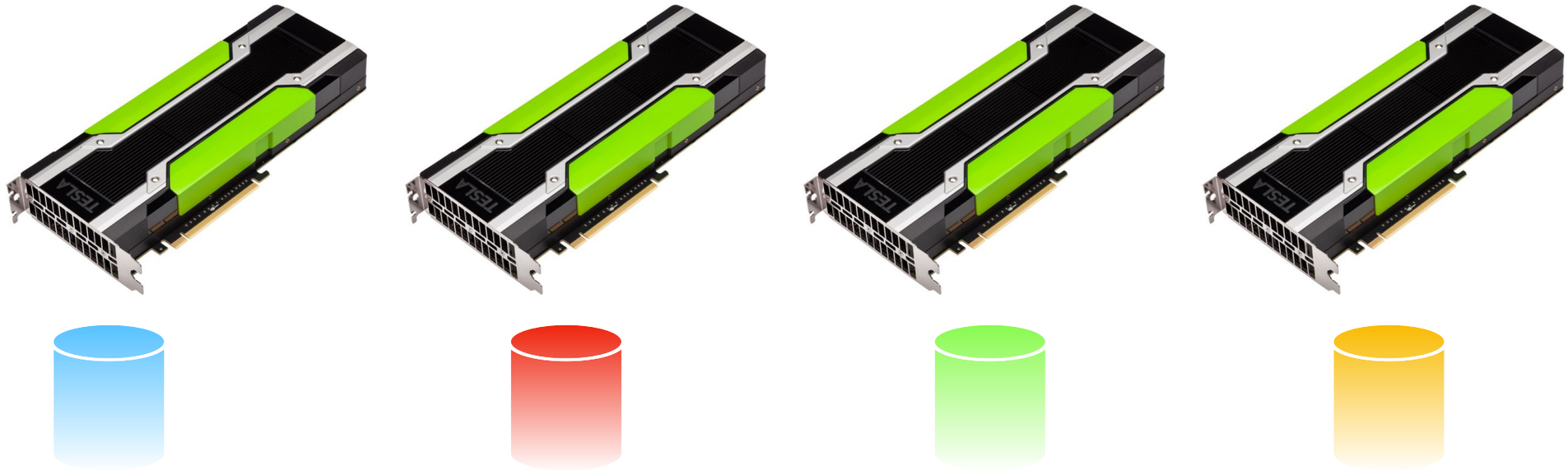
CPU-Driven Server Design 👎🏽

Analysis 🔬

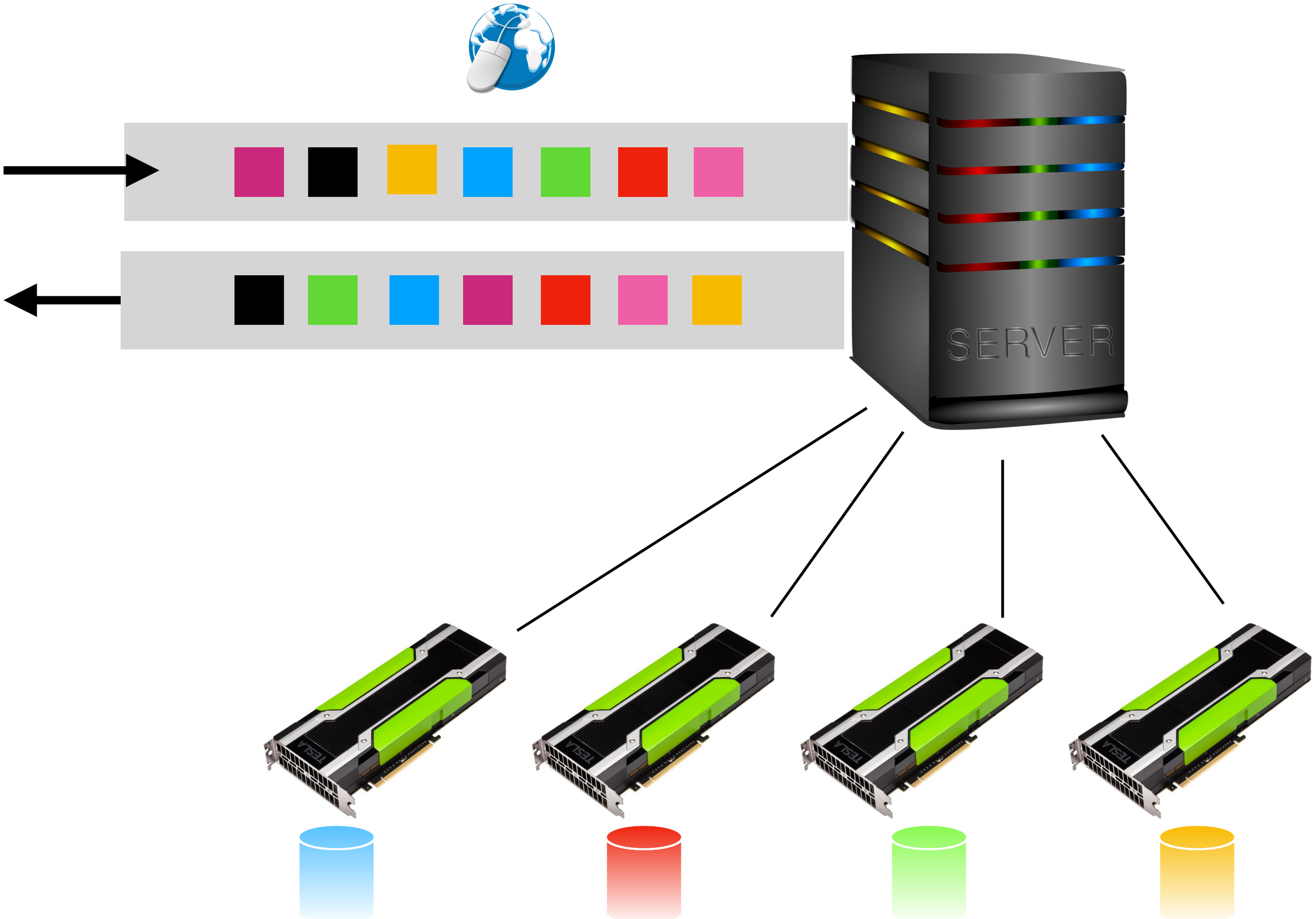
GPU Centric Design (Centaur) 👍🏽

Benchmarks 📊


SERVER
TESLA
TESLA
TESLA
TESLA

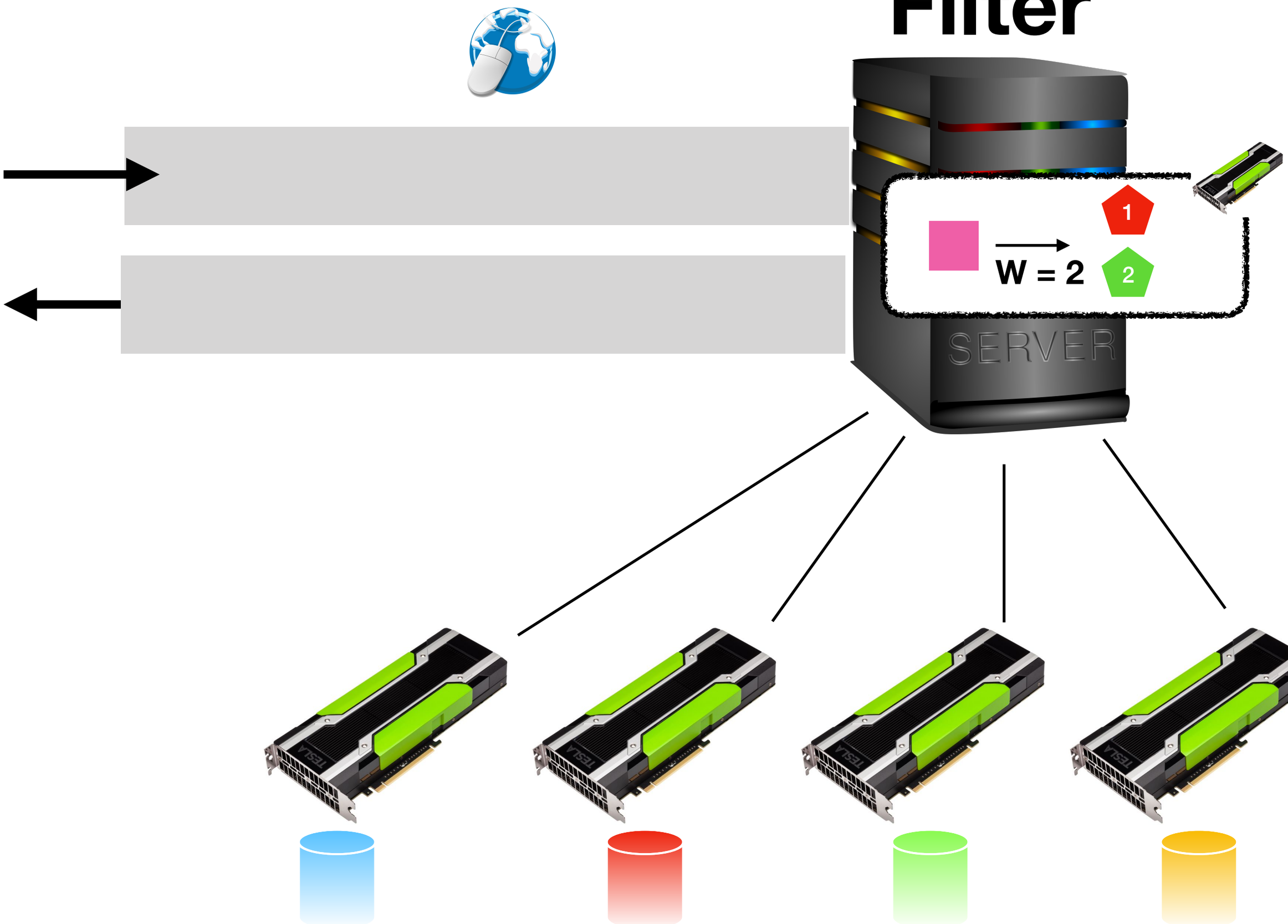


Dataset is split between the GPUs

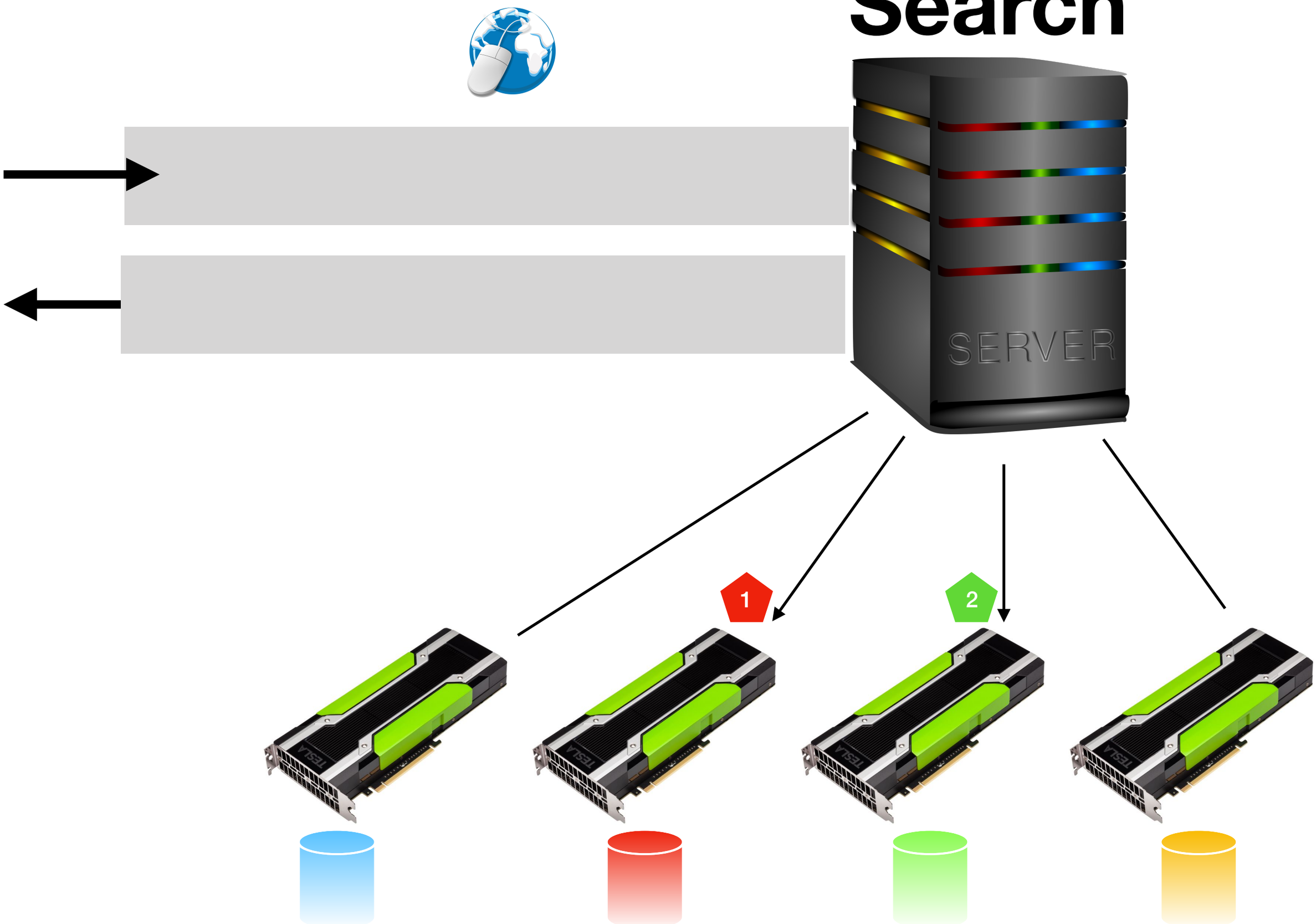

SERVER
TESLA
TESLA
TESLA
TESLA

Receive

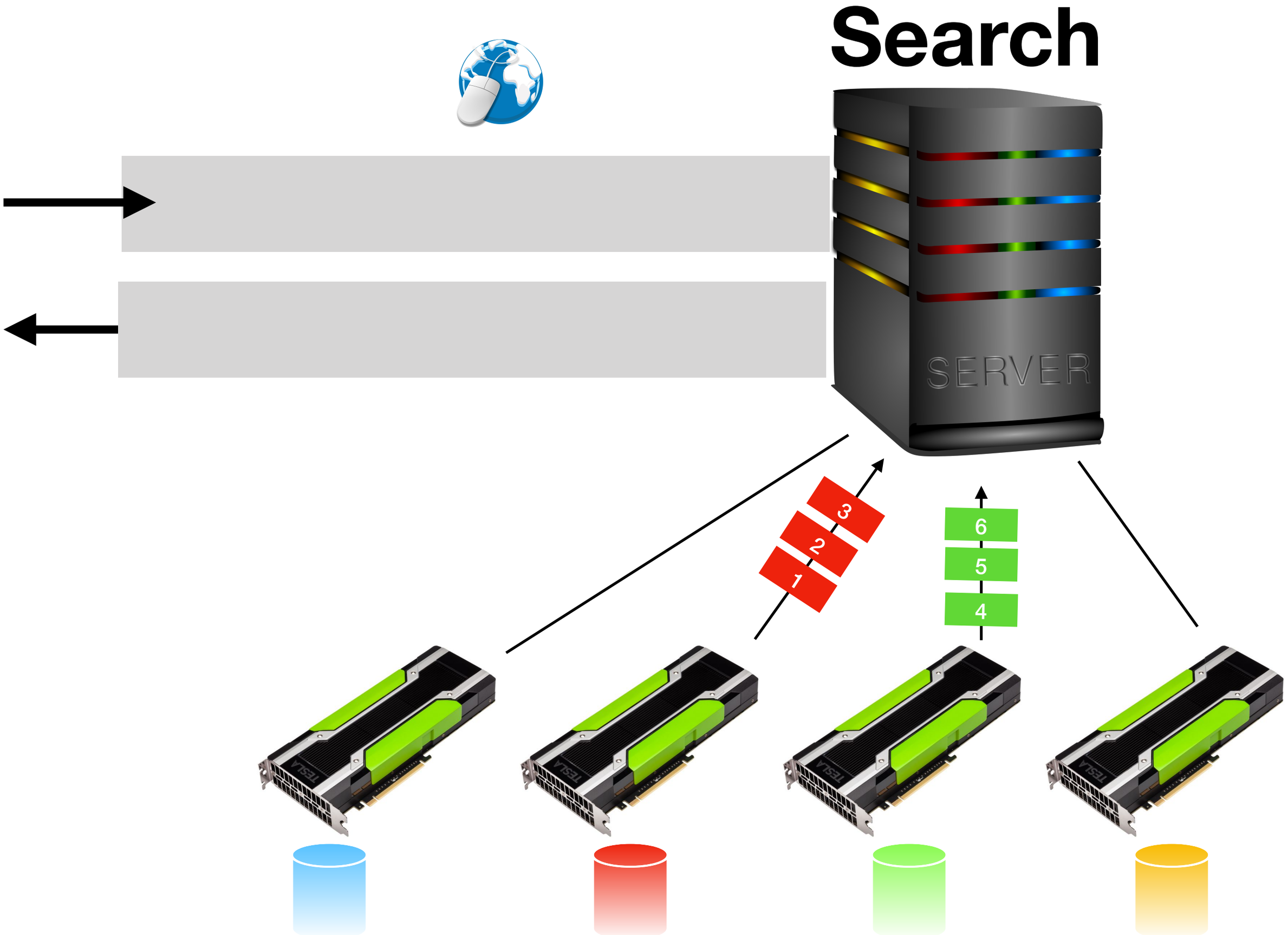
Filter



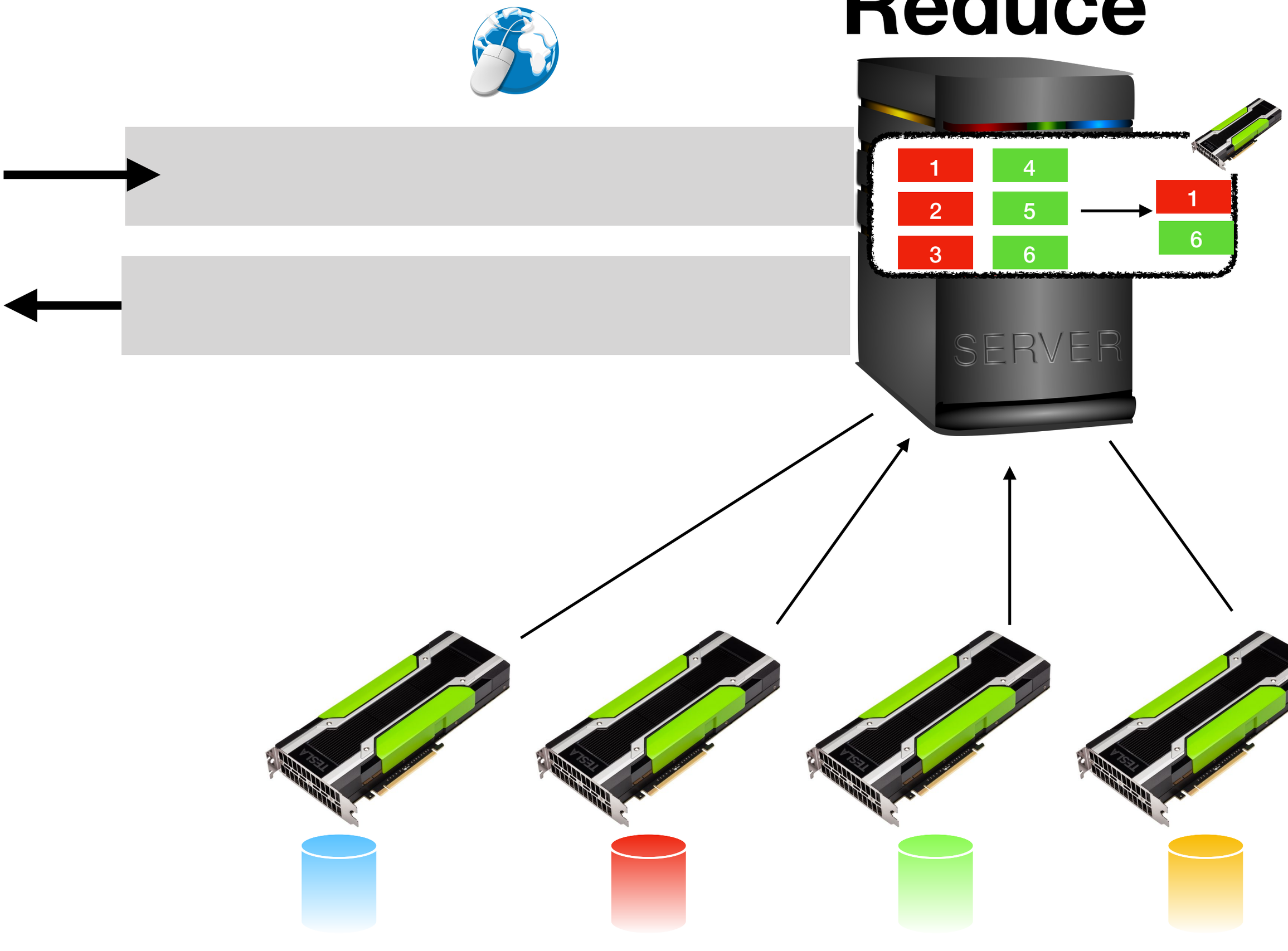
Search



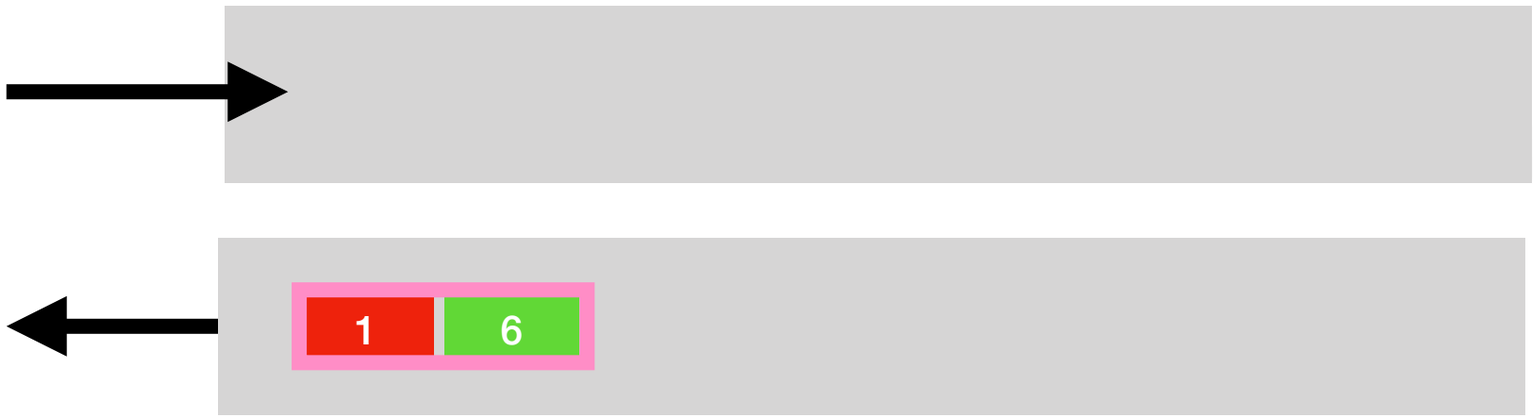
Search

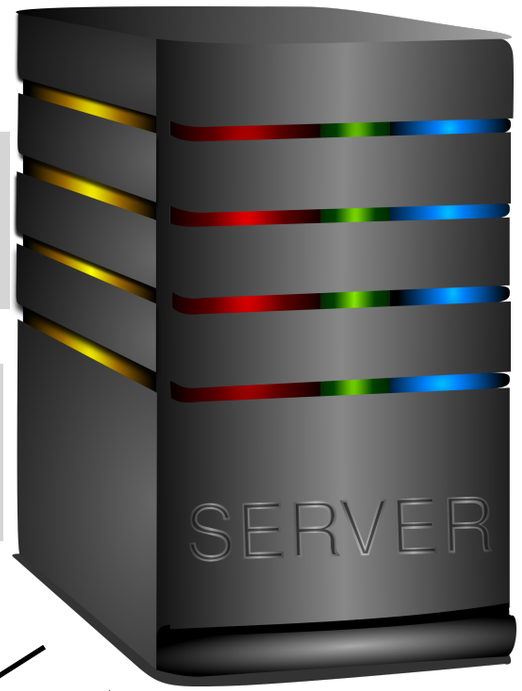


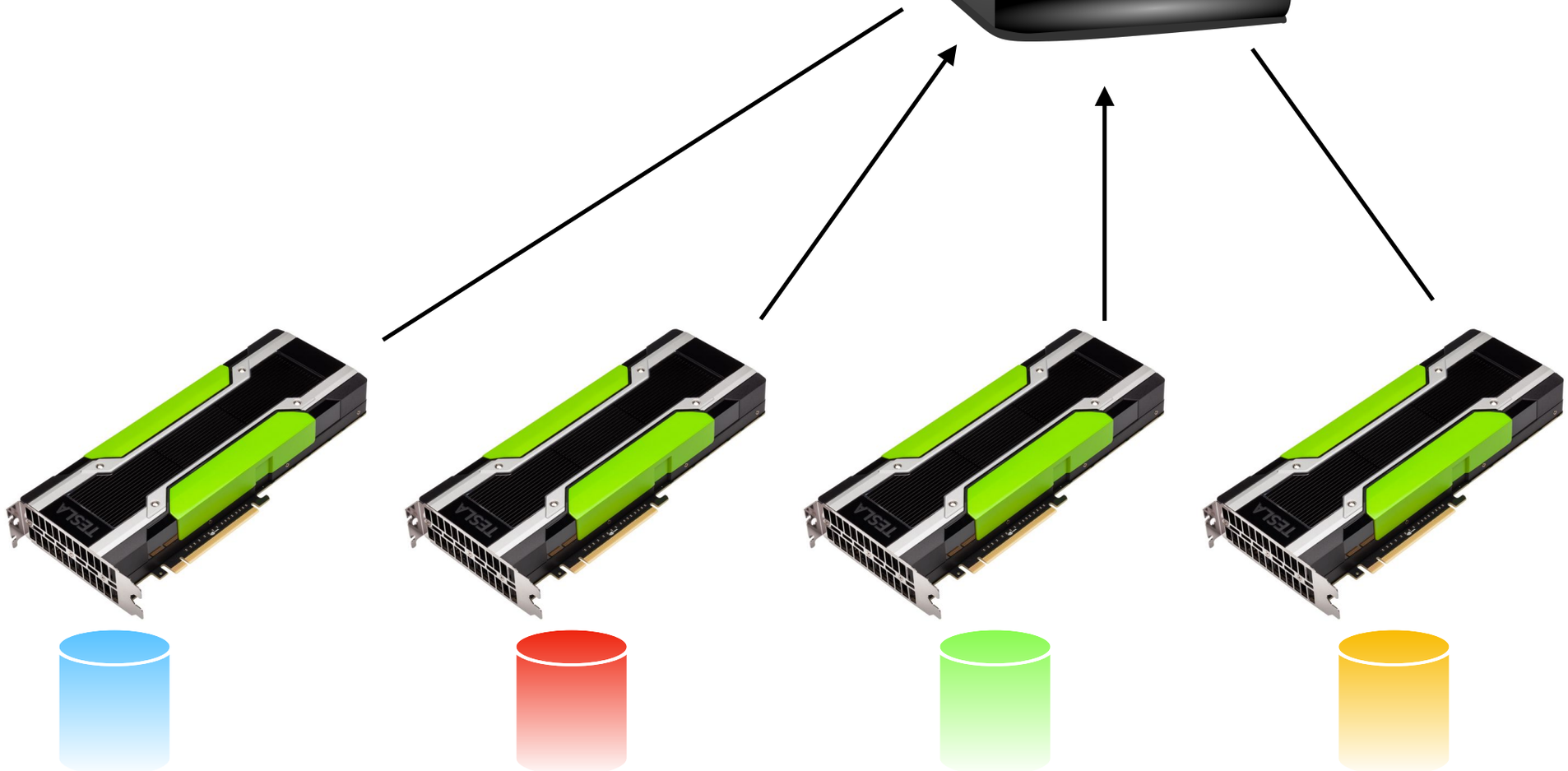
Reduce



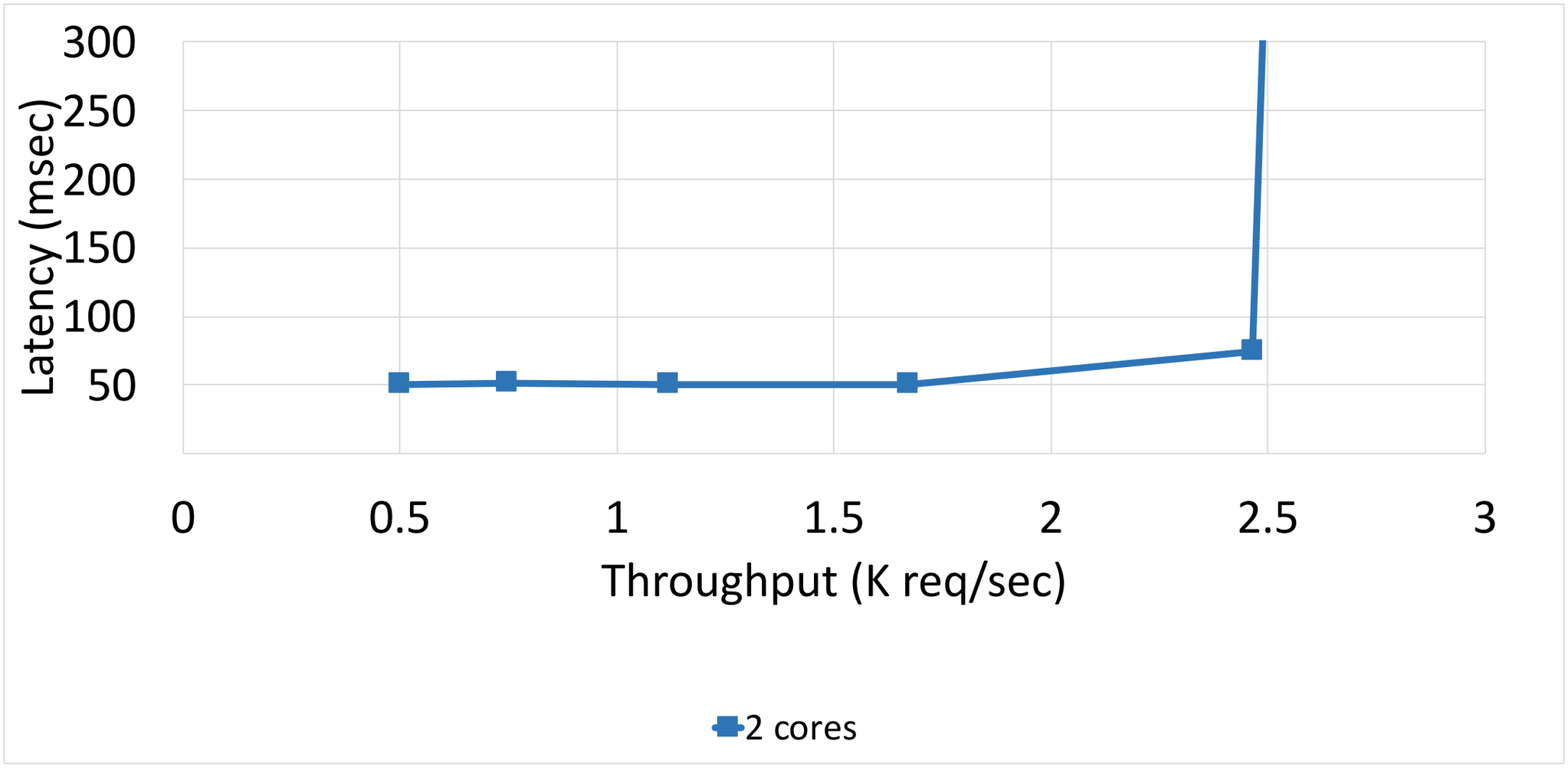
Send





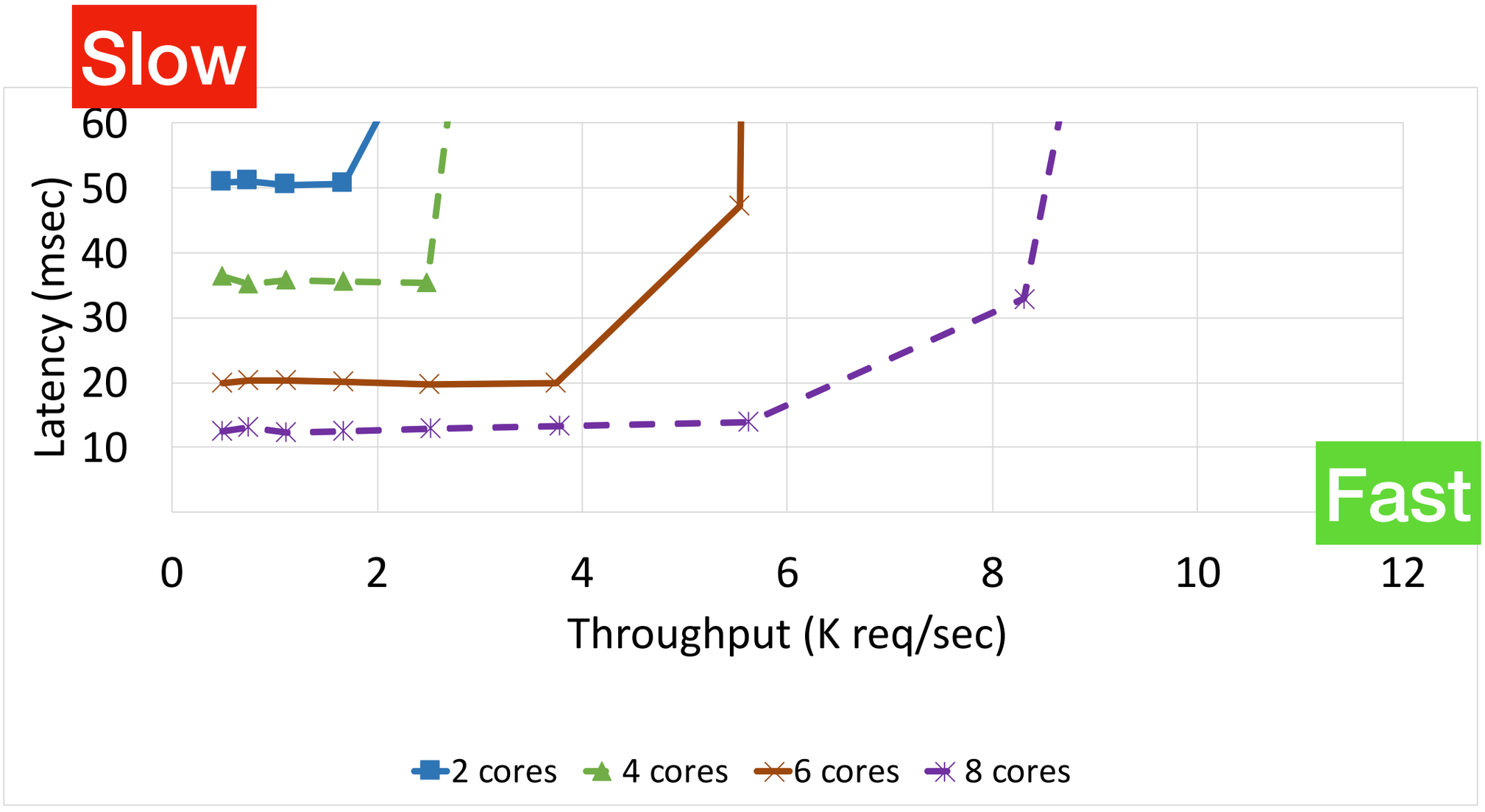


Let’s Run It

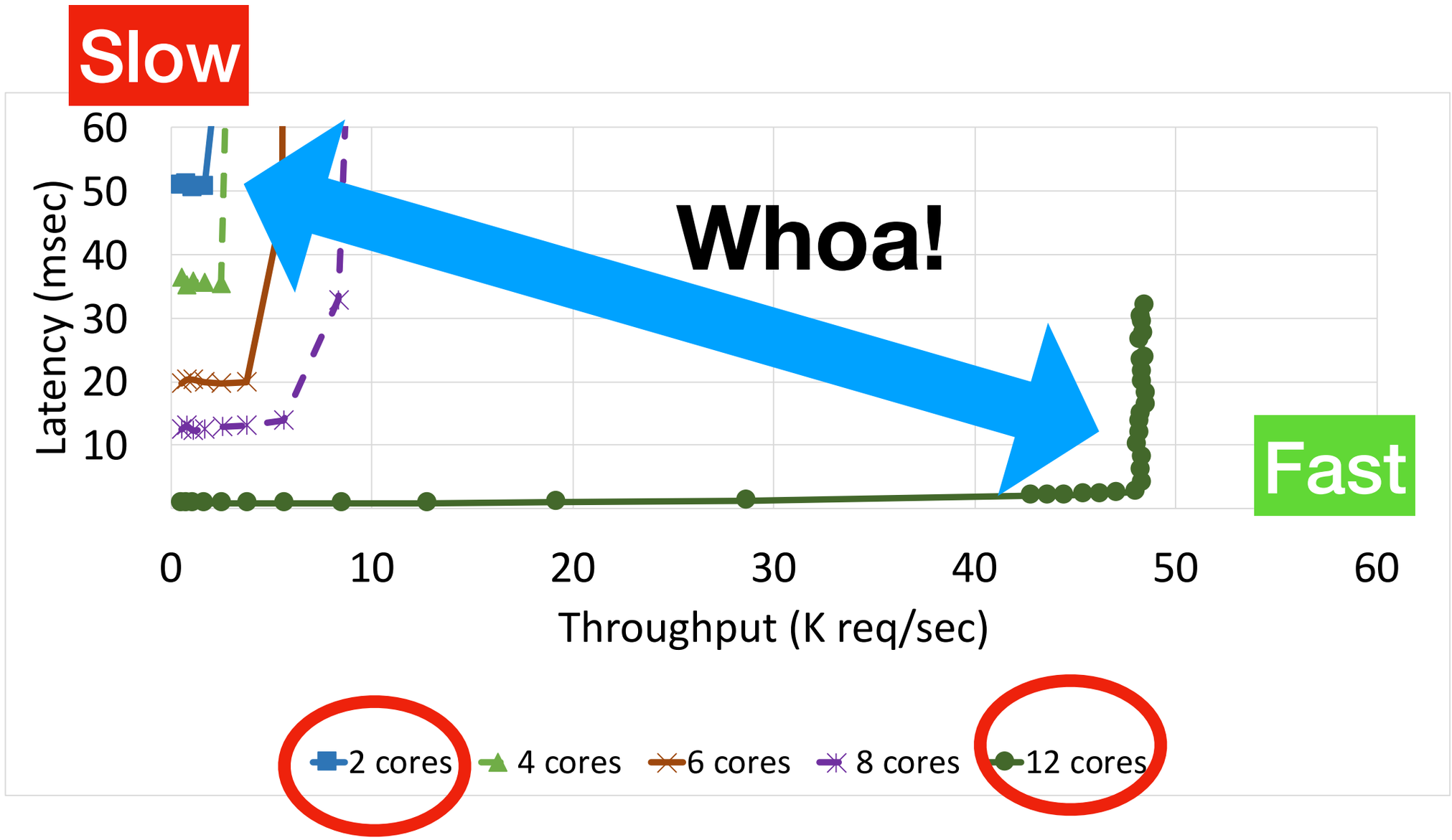


6 GPUs

Let’s Add More CPUs



Even More CPUs

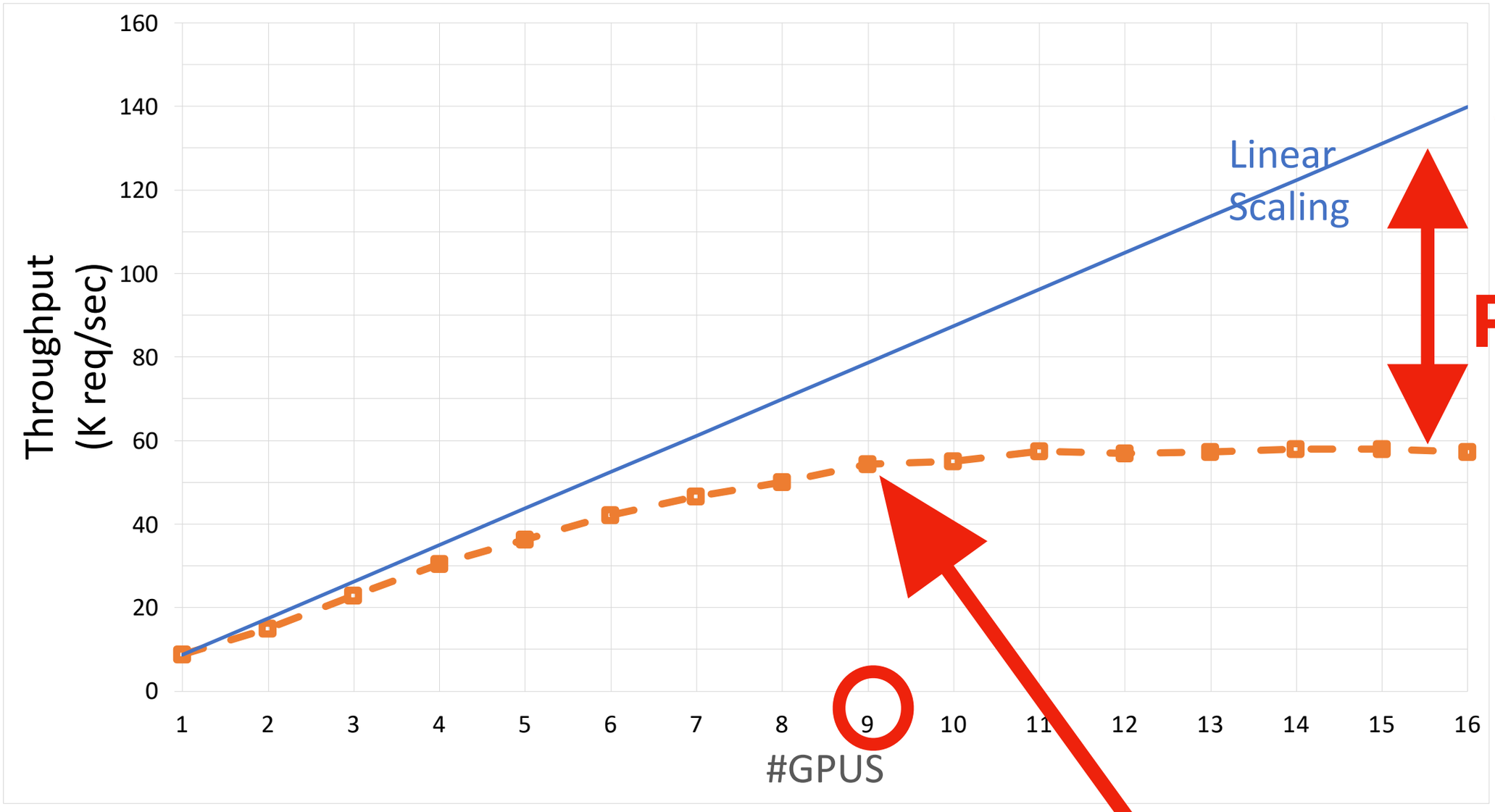


Takeaway

Amount of CPUs in the system limits the performance we can get from GPUs.

Now that we are using all the CPUs in our system, let's add even more GPUs.

More GPUs

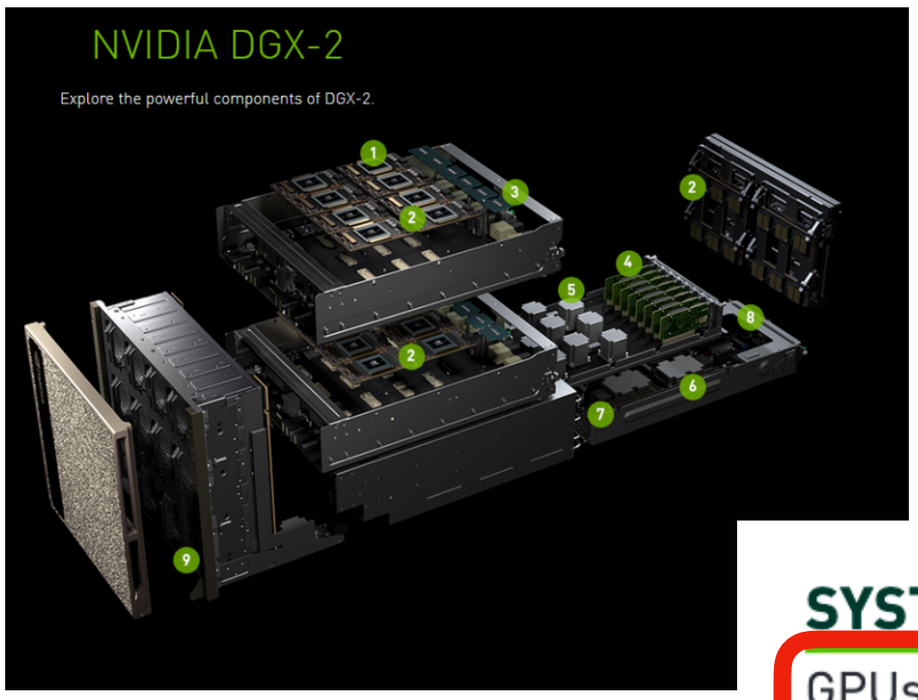


12 CPUs

Takeaway

System's Scalability w.r.t #GPUs is capped by amount of CPUs

We are not Alone



48 CPUs
vs.
16 GPUs
!!

SYSTEM SPECIFICATIONS

GPUs	**16X NVIDIA® Tesla® V100**
GPU Memory	**512GB total**
Performance	**2 petaFLOPS**
NVIDIA CUDA® Cores	**81920**
NVIDIA Tensor Cores	**10240**
NVSwitches	**12**
Maximum Power Usage	**10kW**
CPU	**Dual Intel Xeon Platinum 8168, 2.7 GHz, 24-cores**
System Memory	**1.5TB**

Agenda

~~Example Application: k-NN Network Server~~ 🌐

~~CPU-Driven Server Design~~ 👎🏽

Analysis 🔬

GPU Centric Design (Centaur) 👍🏽

Benchmarks 📊

What Went Wrong?

- Implementation?

- A fundamental issue with the design?

Analysis

How far can we scale the system given a single CPU core?

Assumptions 1

- CPU needs time to invoke a GPU kernel.
- This is just the invocation time, during which the CPU is completely blocked.
- We measured $t_{\text{invoke}} = 5\text{usec}$

Assumptions 2

- CPU needs time to query the status of a GPU kernel.
  - This is just the query time, during which the CPU is completely blocked.
  - We measured $t_{\text{query}} = 3\text{usec}$
- First query always succeeds (optimistic)

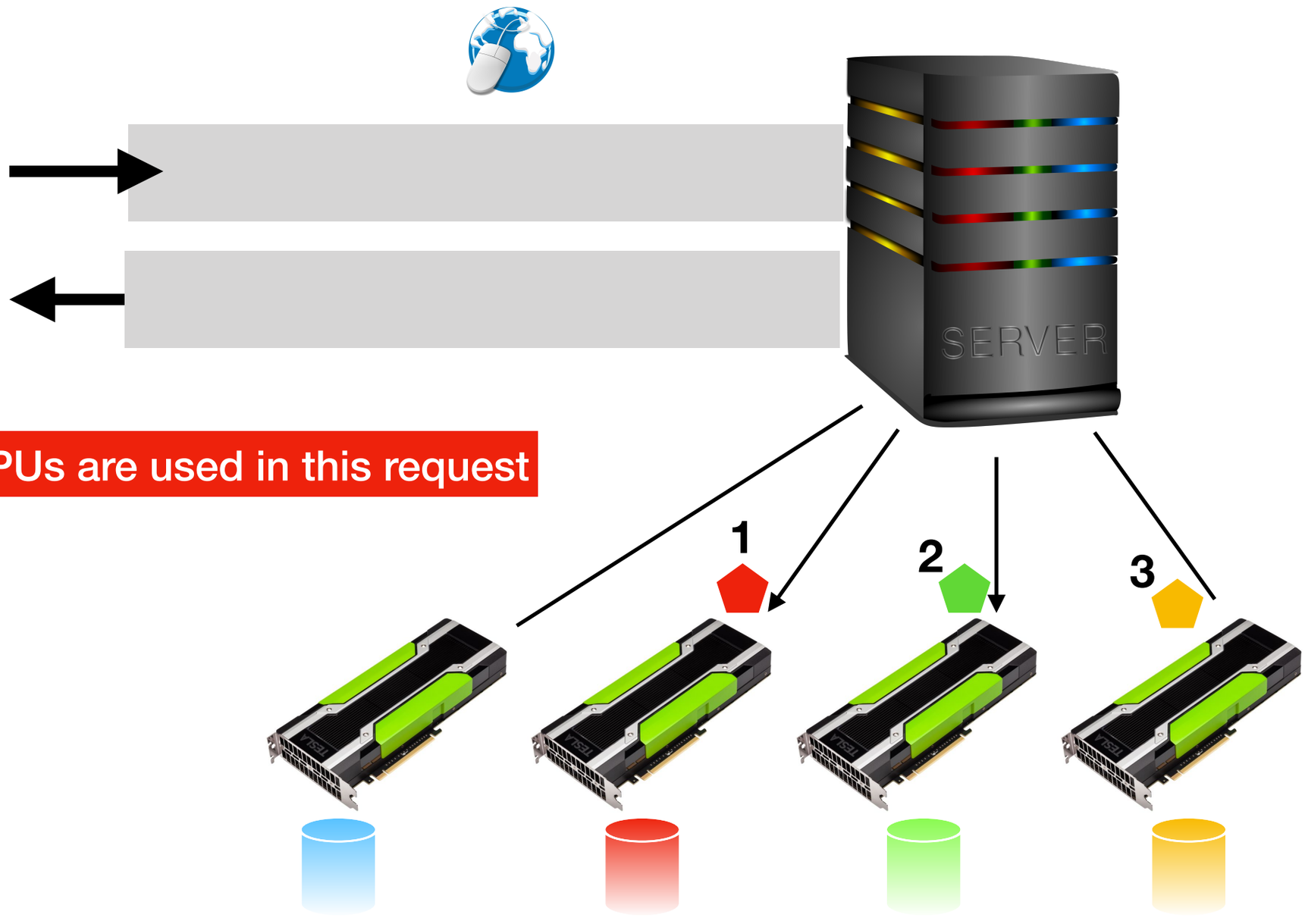
Assumptions 3

- No memory movements (optimistic)
- CPU does not do any computations (optimistic)

CPU Sustained Throughput

$$\text{Throughput} = \frac{1}{\text{kernels per request} \times \text{time to invoke}}$$

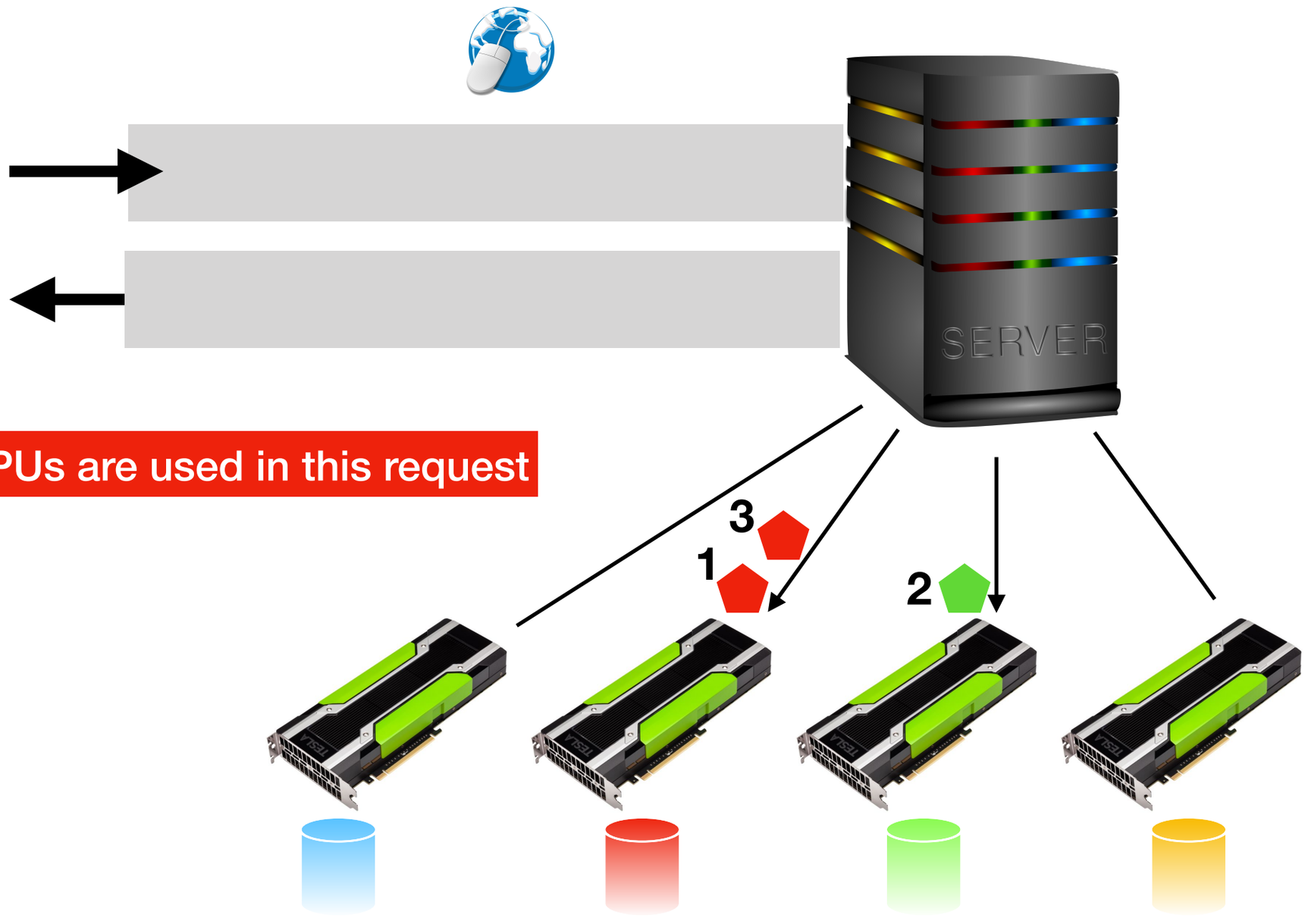
How Many Kernel Invocations Per Request



W = 3

3 GPUs are used in this request

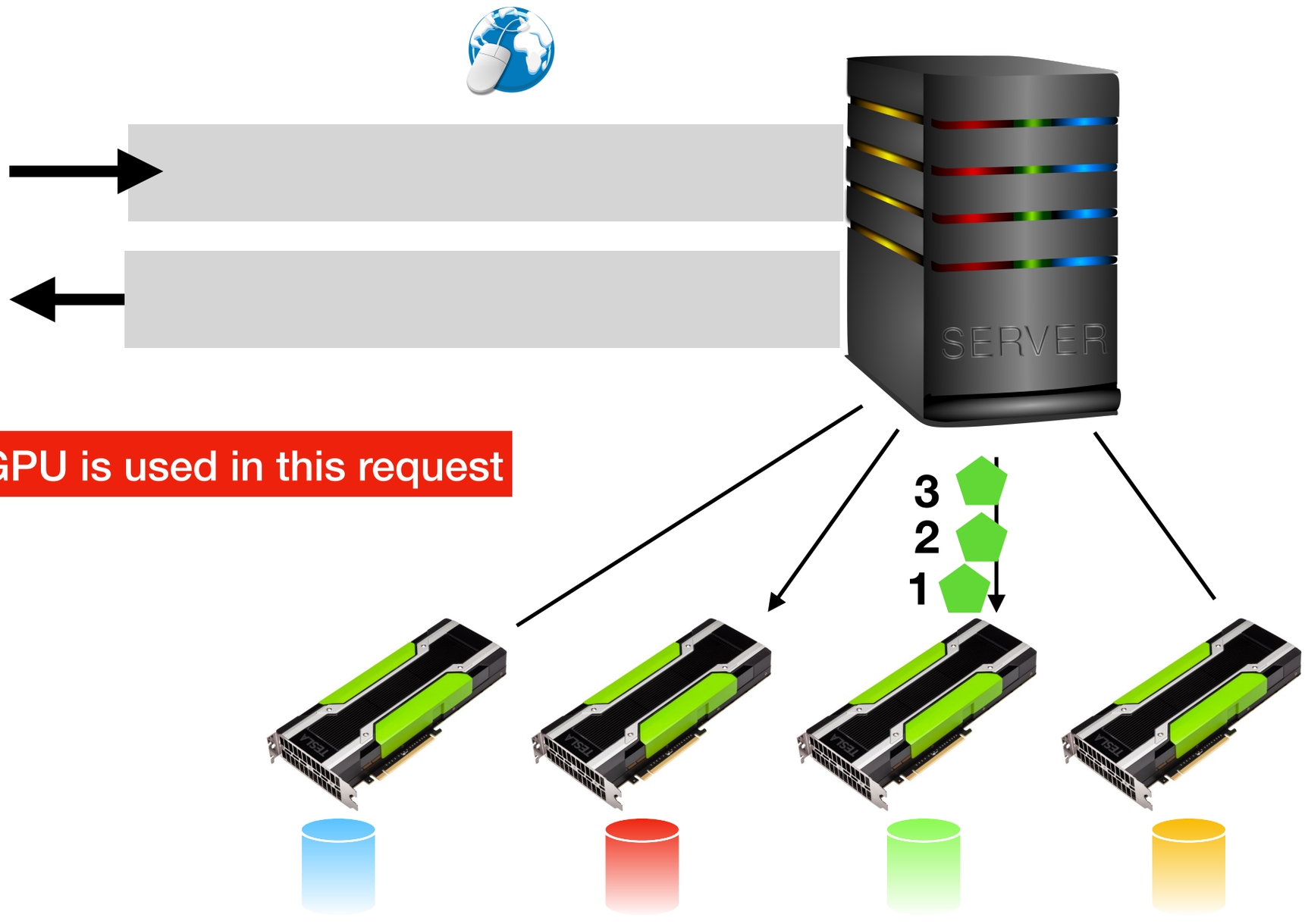
How Many Kernel Invocations Per Request



W = 3

2 GPUs are used in this request

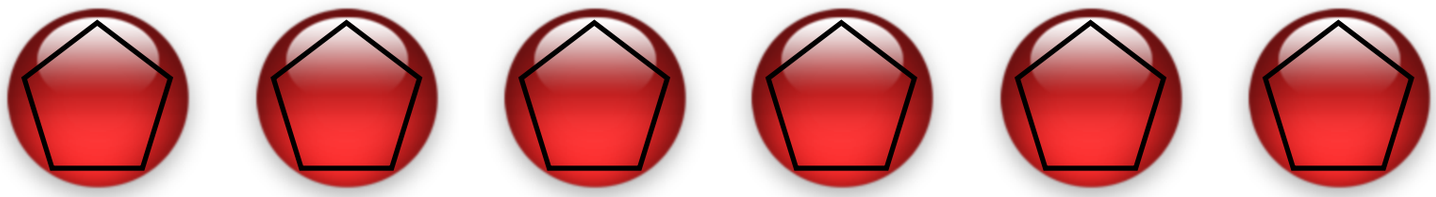
How Many Kernel Invocations Per Request



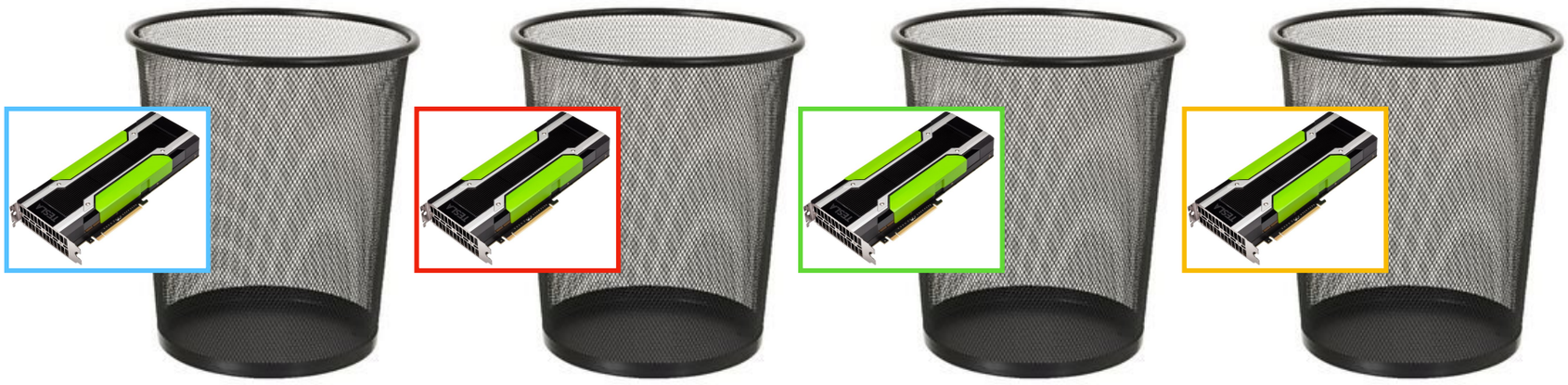
W = 3

1 GPU is used in this request

#GPUs Accessed Per Request



W balls



N bins

- Expected number of non-empty bins: $N\left(1-\left(1-\frac{1}{N}\right)^{W}\right)$

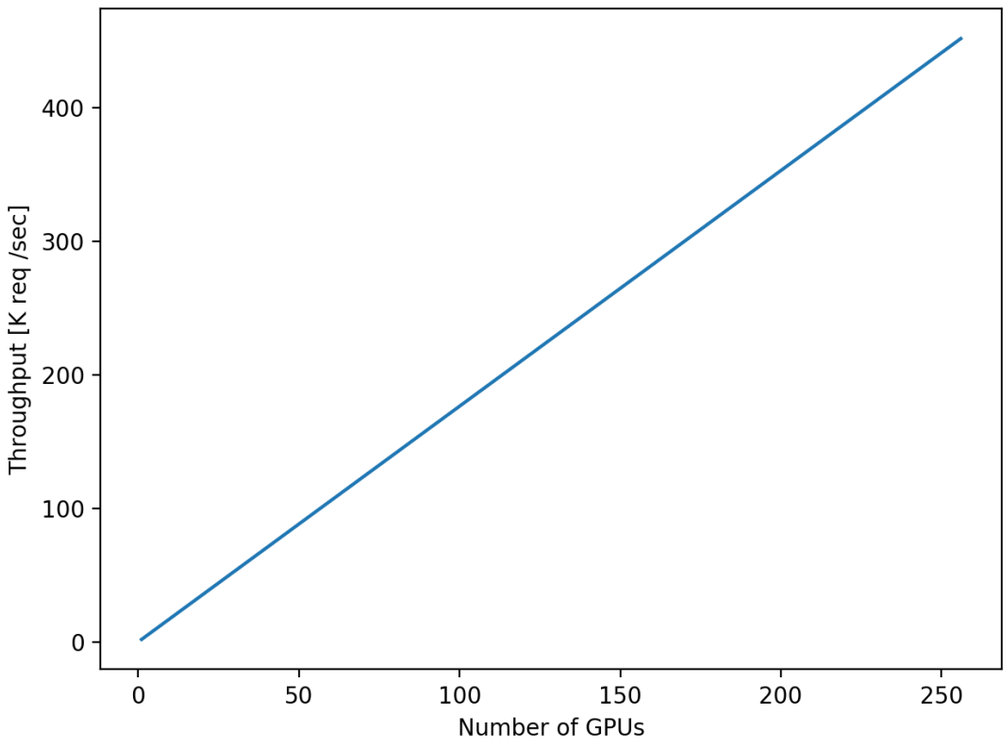
Sustained Throughput for Multiple Requests

$$N_{\text{invocations}} = N\left(1 - \left(1 - \frac{1}{N}\right)^{W}\right)$$

$$\text{Throughput} = \frac{1}{N_{\text{invocations}}}$$

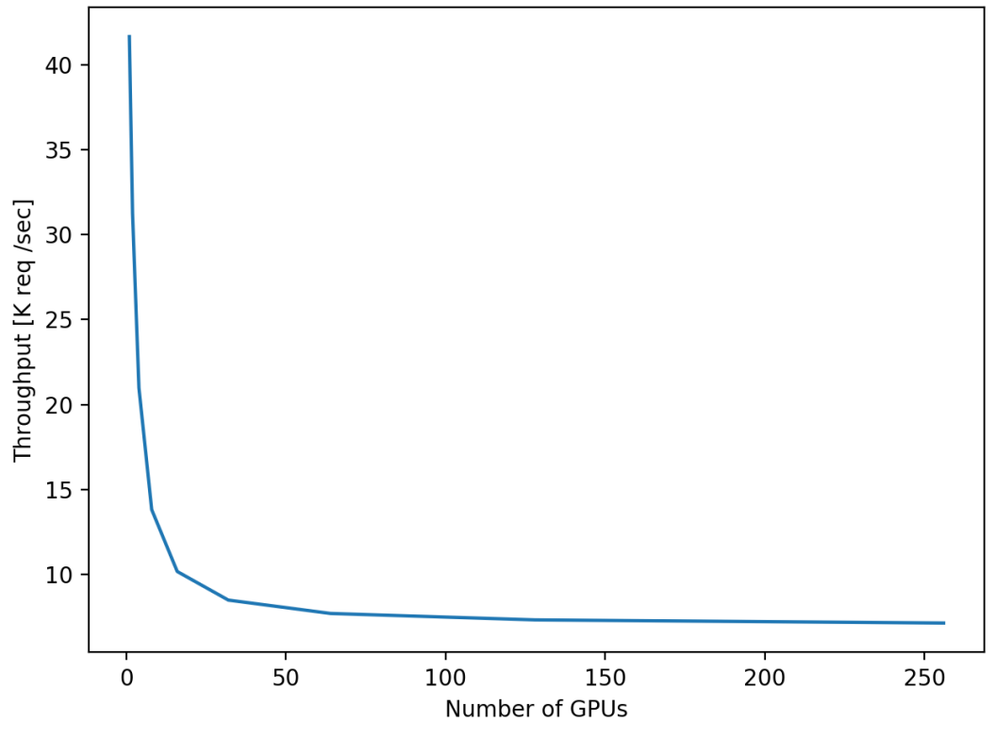
Overall Server Throughput

GPUs

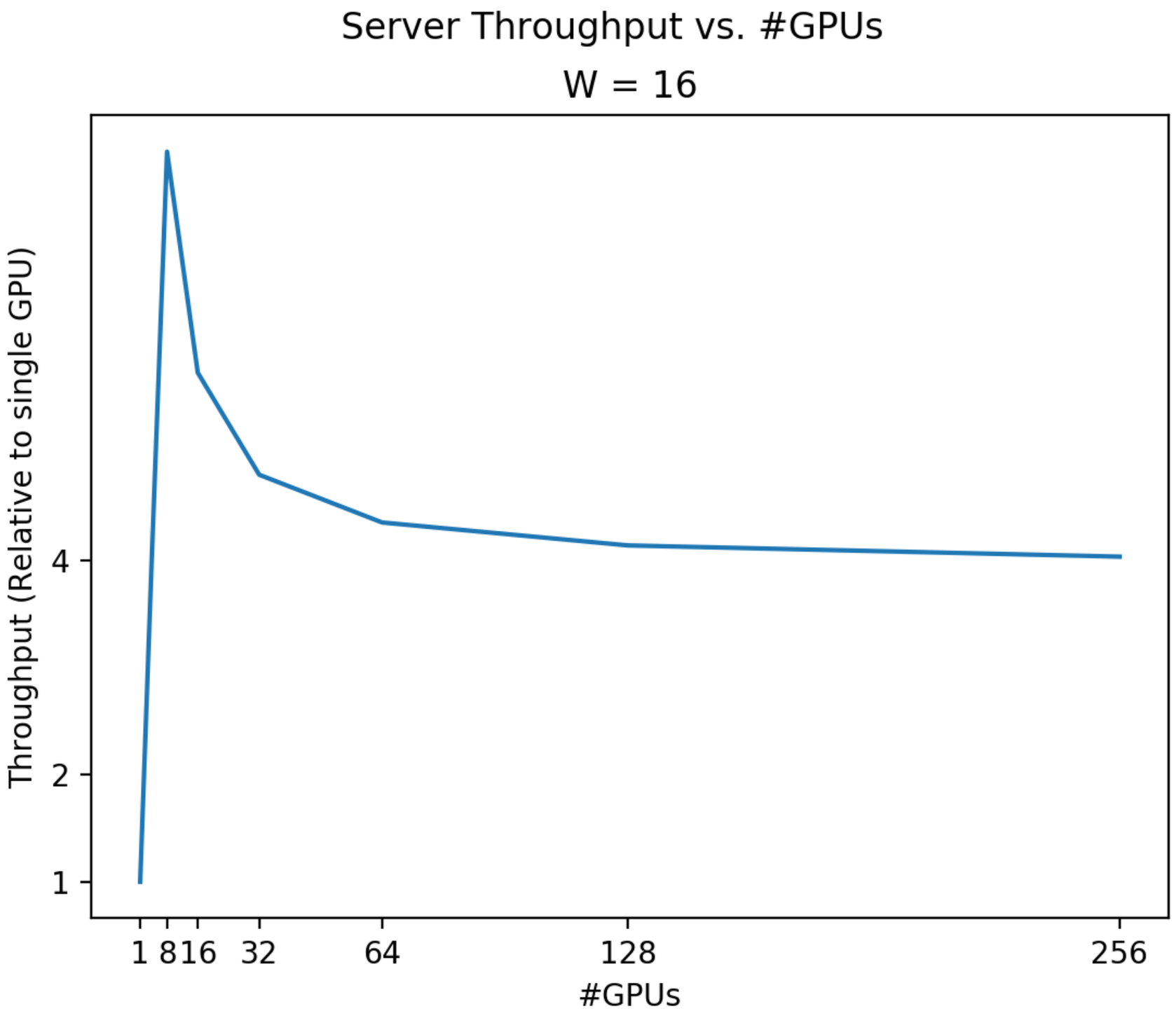


+

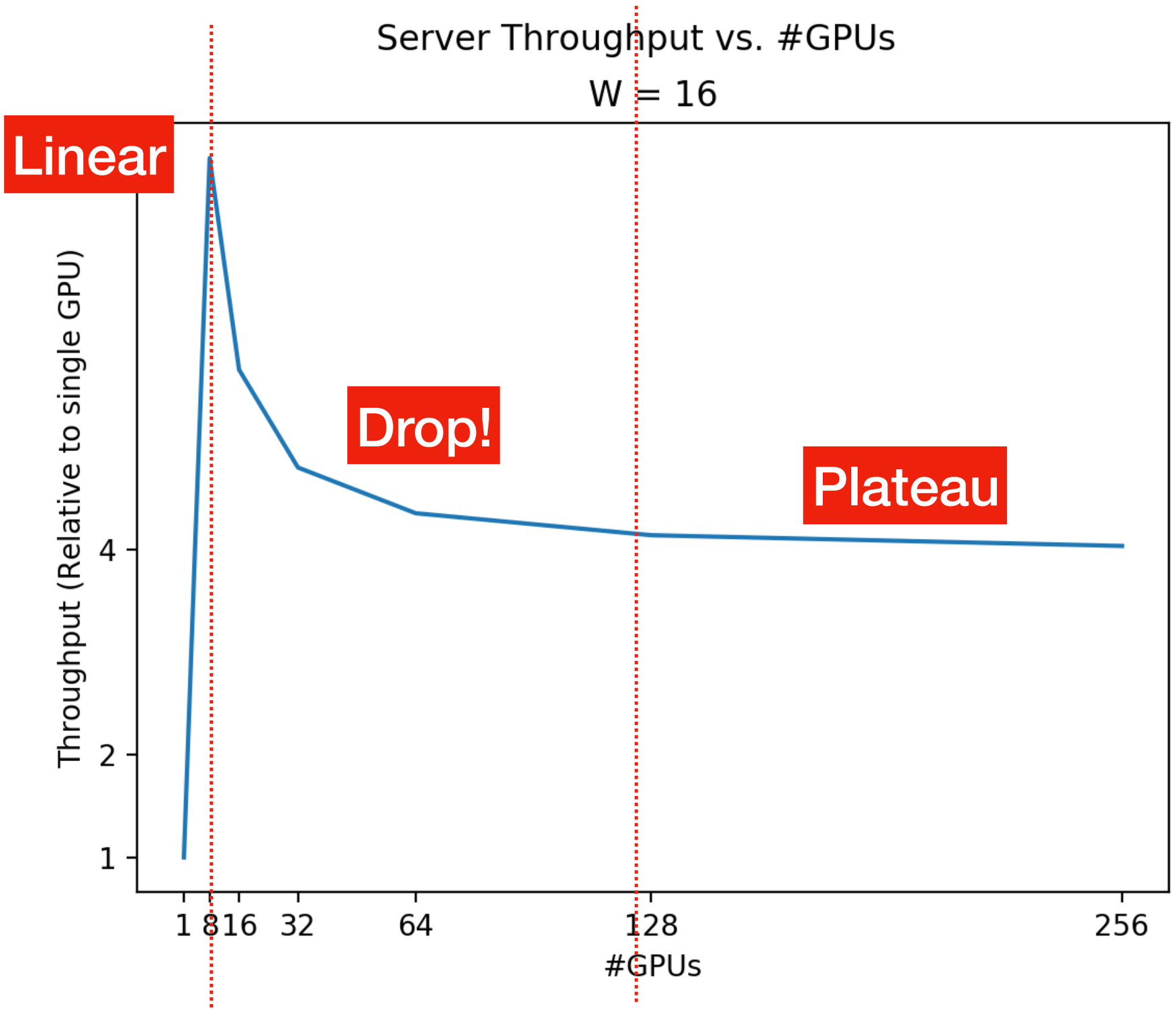
CPU



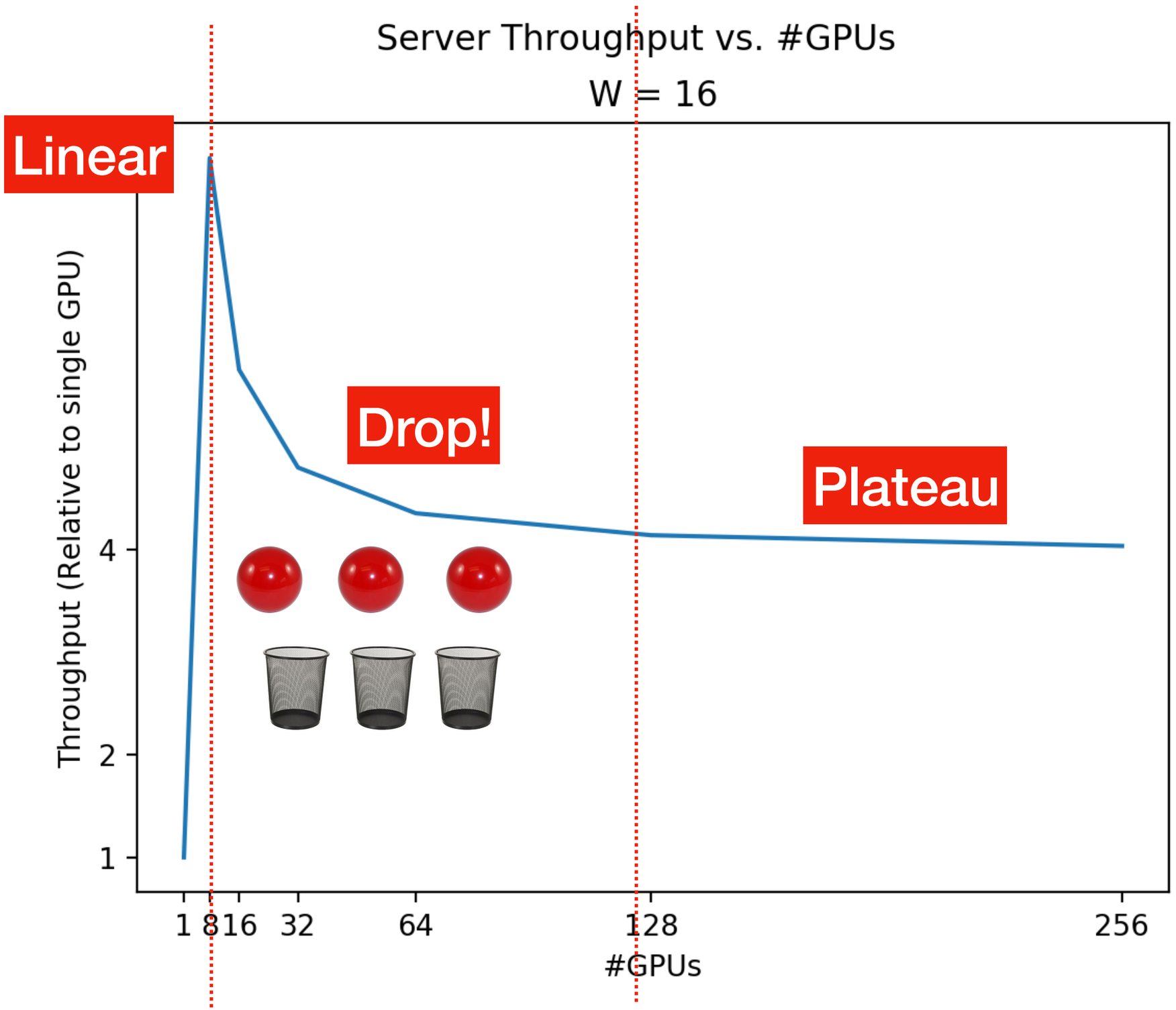
Overall Server Throughput



Overall Server Throughput



Overall Server Throughput



Takeaway

More Thorough Analysis in the Paper

The need for more CPUs as we add GPUs
is an inherent problem in CPU-Driven Server Design

Agenda

~~Example Application: k-NN Network Server~~ 🌐

~~CPU-Driven Server Design~~ 👎🏽

~~Analysis~~ 🔬

GPU Centric Design (Centaur) 👍🏽

Benchmarks 📊

Centaur: Removing CPU from Management

Why did we Need CPU?

1. Network send/recv

Why did we Need CPU?

1. Network send/recv
2. Invoke computations on GPUs

Why did we Need CPU?

1. Network send/recv
2. Invoke computations on GPUs
3. Communication and Coordination between GPUs

“CPU-less” Networking

GPUnet (OSDI 2014):

Network sockets for the GPU

Why did we Need CPU?

1. ~~Network send/recv~~
2. Invoke computations on GPUs
3. Communication and Coordination between GPUs

Compute Invocation

Persistent Kernels:

```
my_kernel() {
    while (1) {
        in = pop_task(); //from queue or socket
        out = compute(in);
        push_task(out); //via queue or socket
    }
}
```

Why did we Need CPU?

1. ~~Network send/recv~~
2. ~~Invoke computations on GPUs~~
3. Communication and Coordination between GPUs

GPU-GPU Communication and Coordination

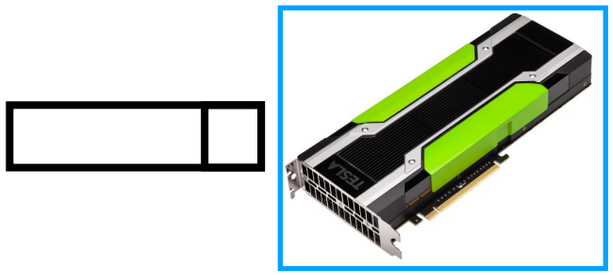
gpipes (Introduced in this paper)

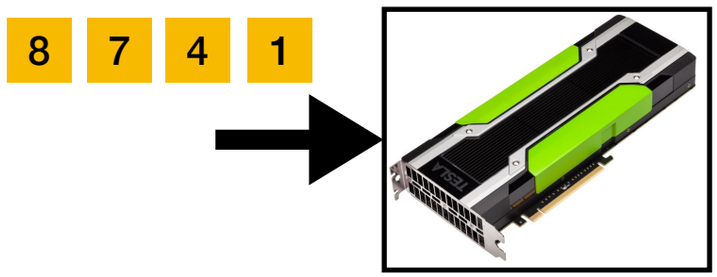
Why did we Need ~~CPU~~?

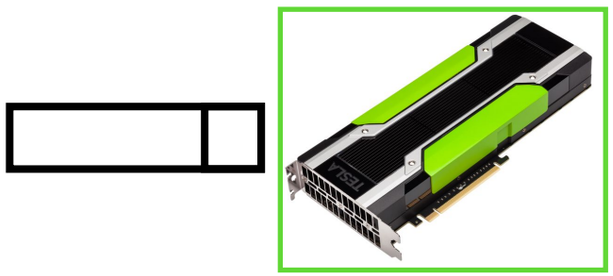
1. ~~Network send/recv~~
2. ~~Invoke computations on GPUs~~
3. ~~Communication and Coordination between GPUs~~

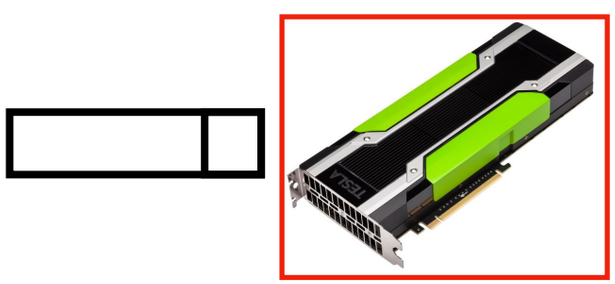
Centaur's Design

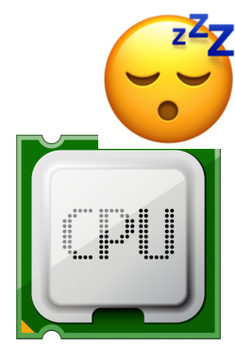
filter **search** **reduce**









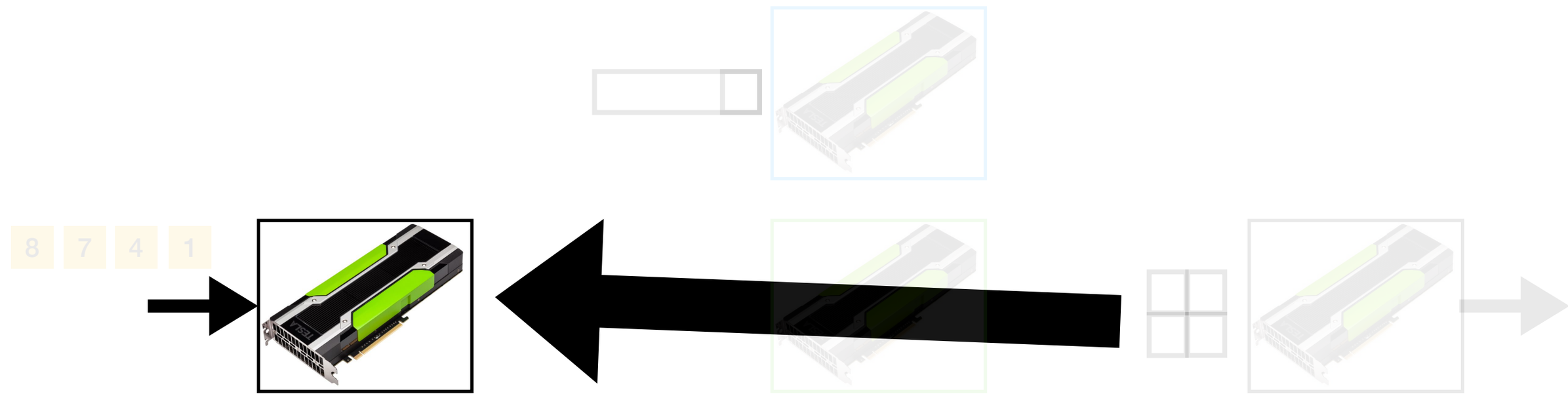


Components:
Network Requests

filter search reduce

Components: Persistent Kernels

filter search reduce



Filter's persistent kernel

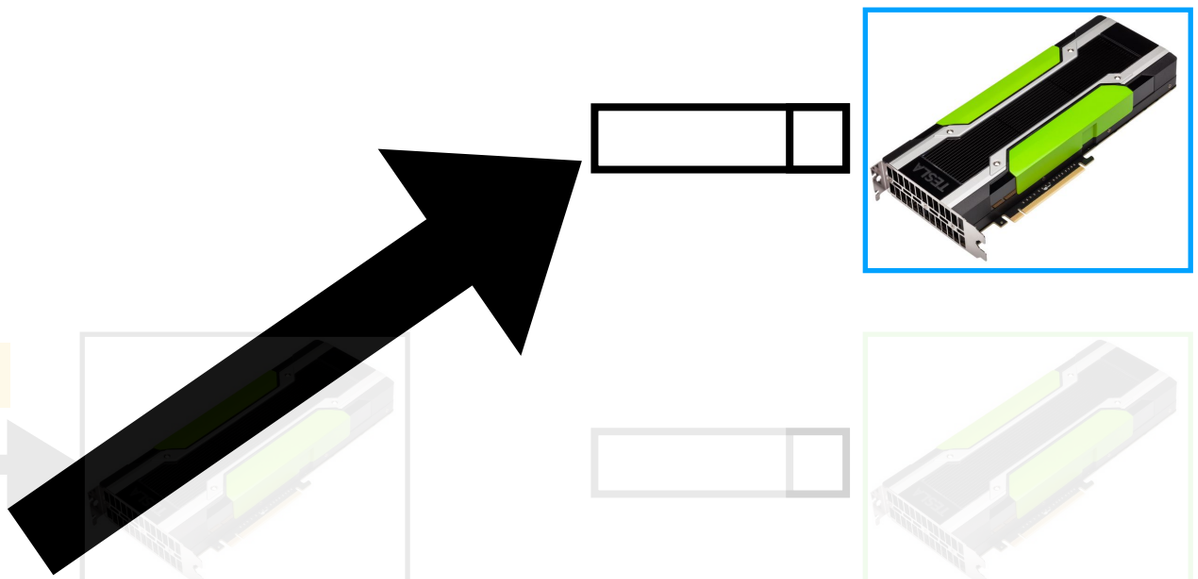
Receives requests using GPUnet

Components:
1-1 gpipes

filter

search

reduce



8 7 4 1

1-1 gpipe:
from filter to search

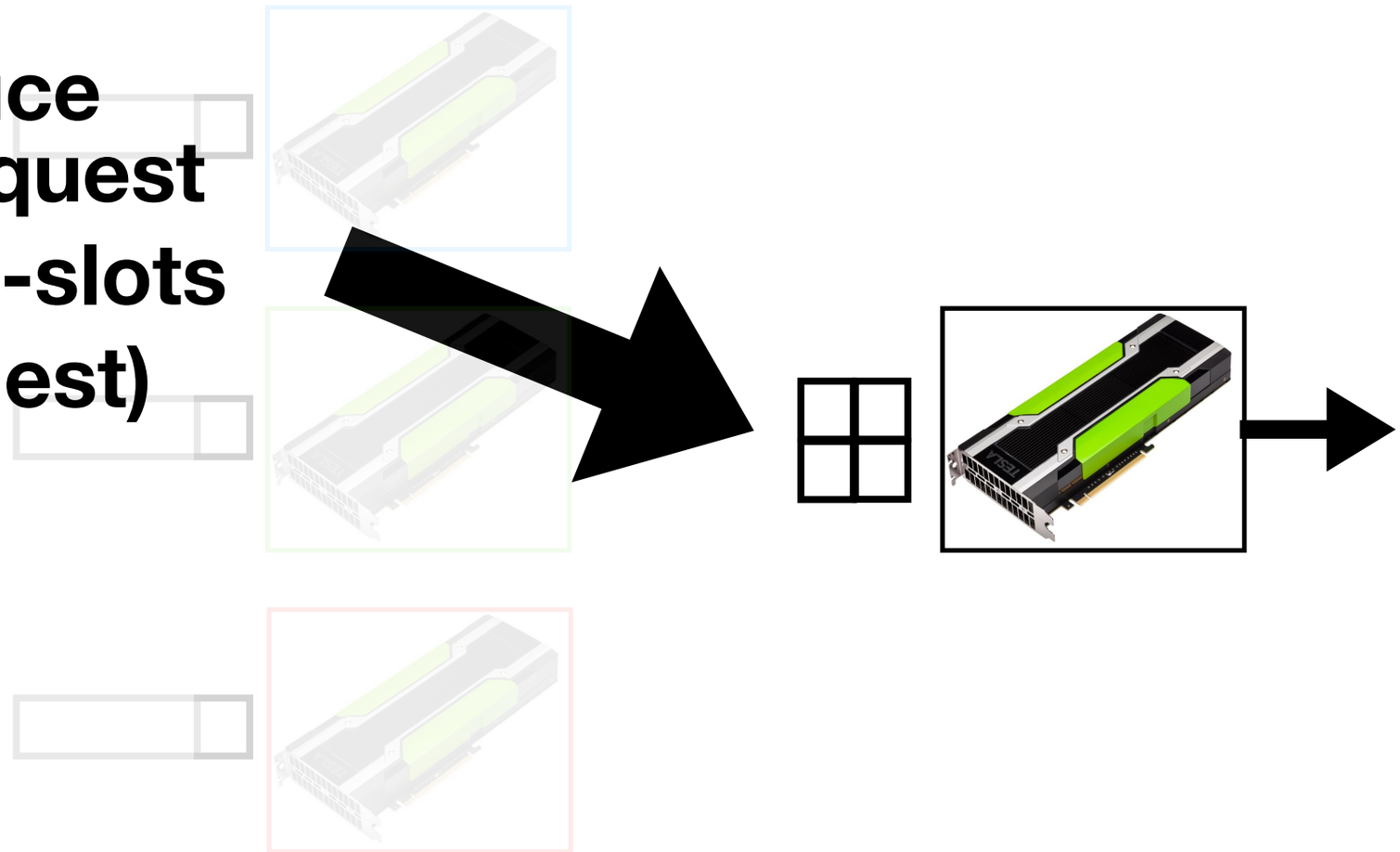
Components: Reduce gpipe

filter search reduce

reduce gpipe:

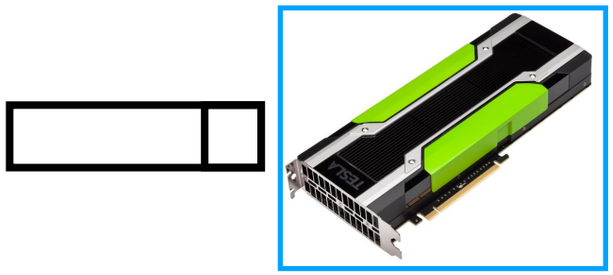
From search to reduce

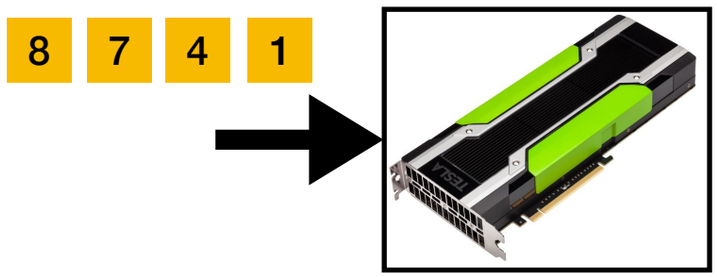
- **Slot per user request**
- **Broken into sub-slots (1 per sub-request)**

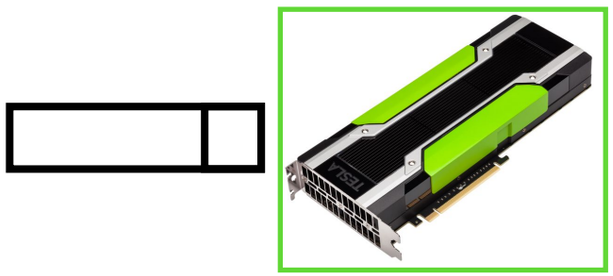


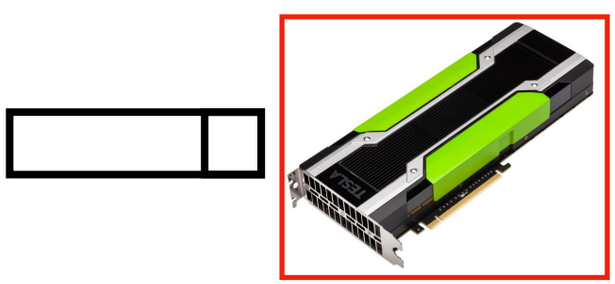
Centaur's Design

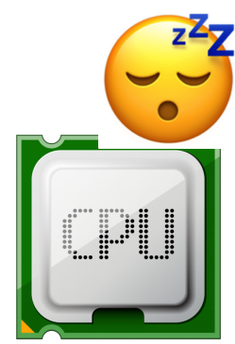
filter **search** **reduce**











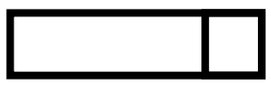
gpipes
1-1 gpipe 

- Single Producer - Single Consumer queue
  - Avoids the need for atomics across PCIe

- Placed in Consumer’s memory
  - Avoids PCIe read round trip

- Uses NVIDIA’s GPUDirect

gpipes

1-1 gpipe 

- Single Producer - Single Consumer queue
  - Avoids the need for atomics across PCIe
- Placed in Consumer's memory
  - Avoids PCIe read round trip
- Uses NVIDIA's GPUDirec

More in the paper:
e.g. How we worked around GPUDirect's limitation of 8 peers max.

gpipes:
Reduce gpipe ⊞

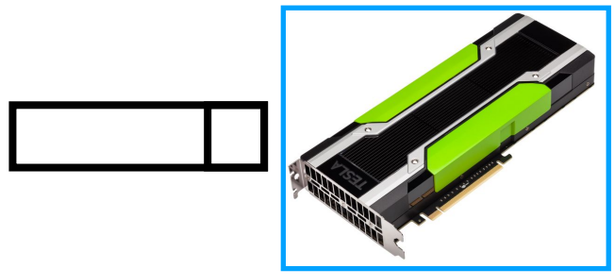
- Multiple Producers - Single Consumer

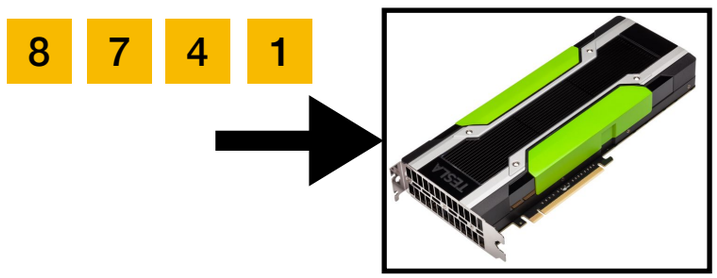
more in the paper

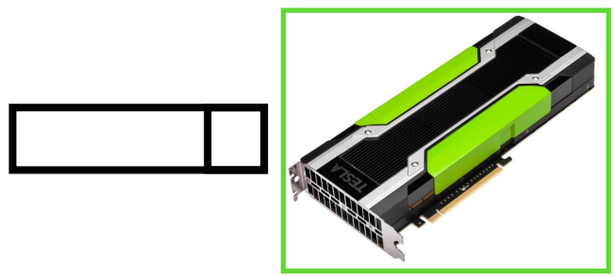
- Reservation mechanism
  - Avoids the need for PCIe atomics
  - Prevents reordering deadlocks

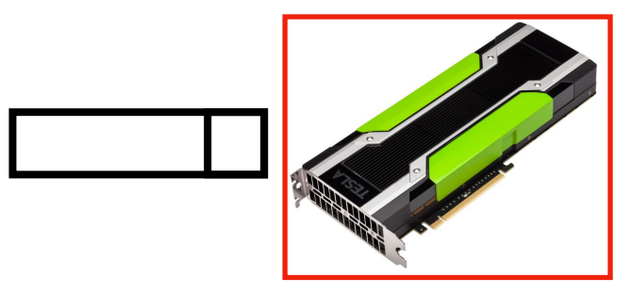
Running Example (1 Request)

filter **search** **reduce**



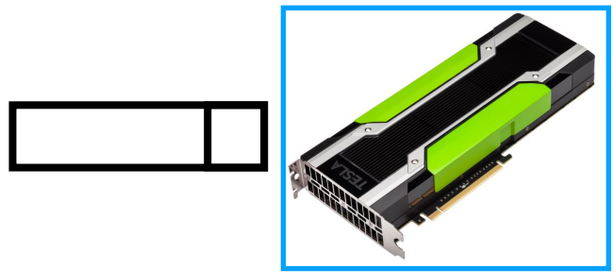


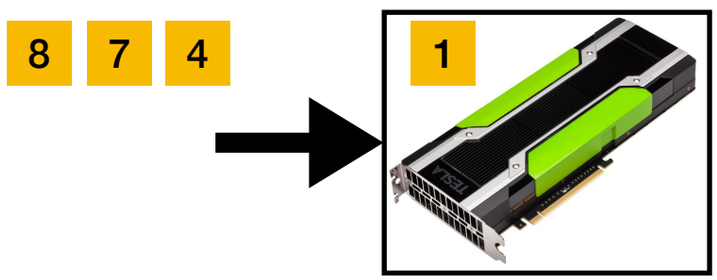


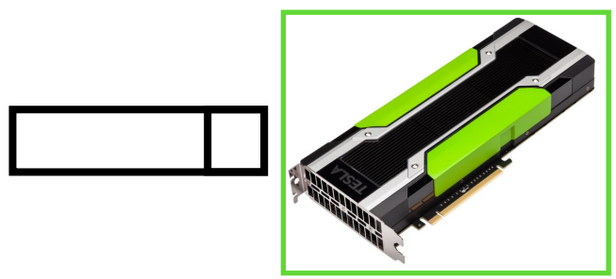


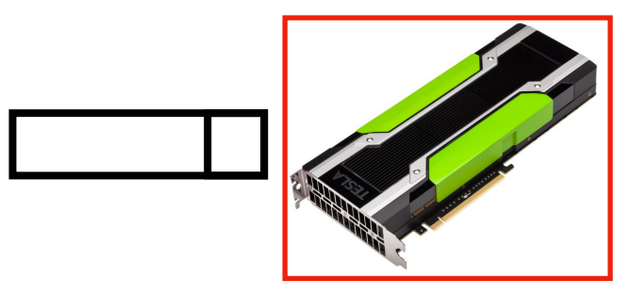
Running Example (1 Request)

filter **search** **reduce**



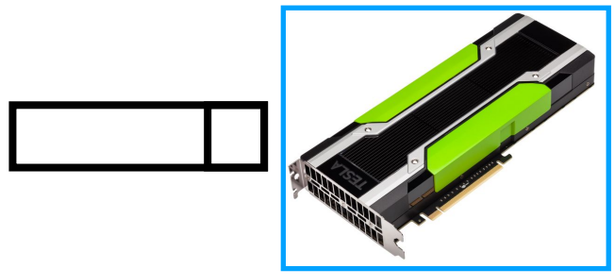




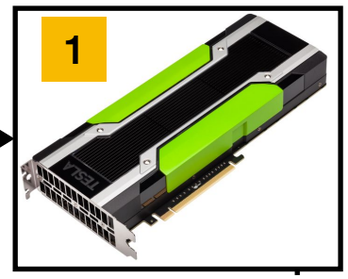


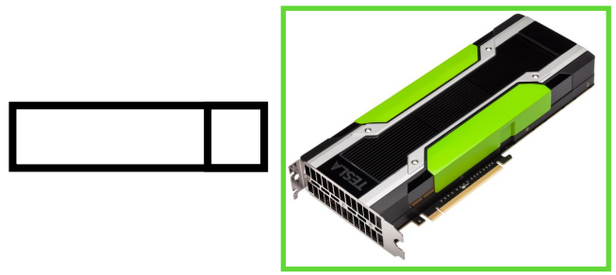
Running Example (1 Request)

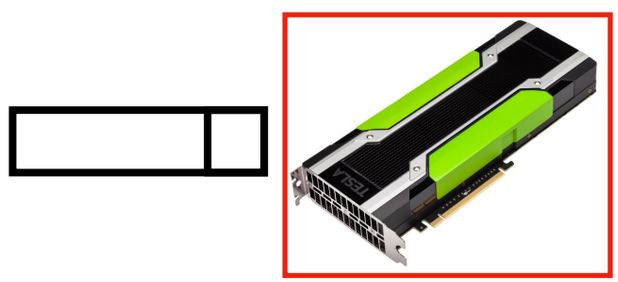
filter **search** **reduce**



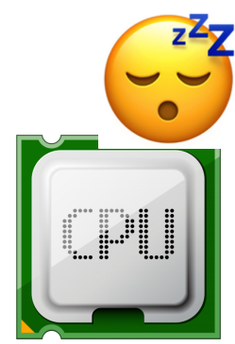
8 7 4





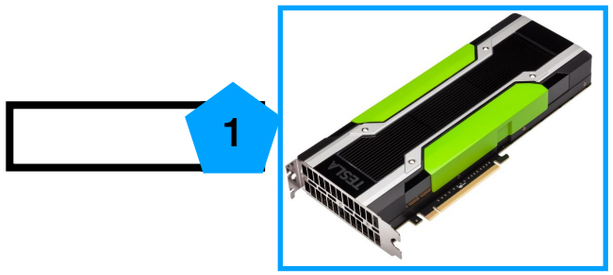


reserve

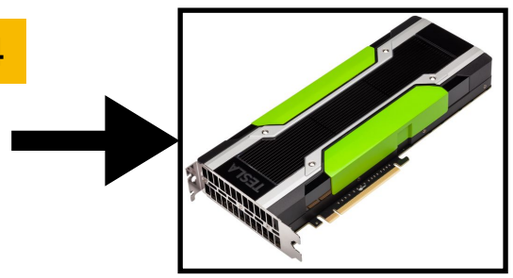


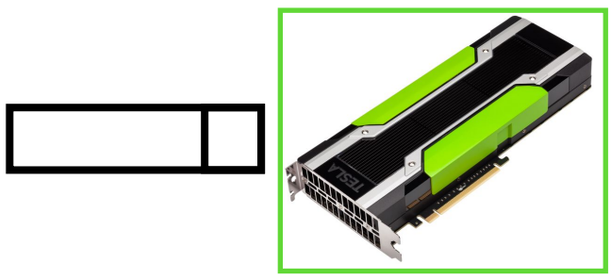
Running Example (1 Request)

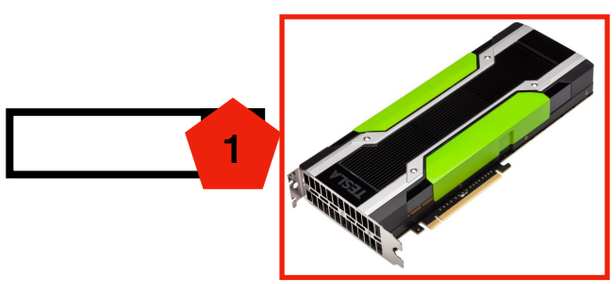
filter **search** **reduce**



8 7 4

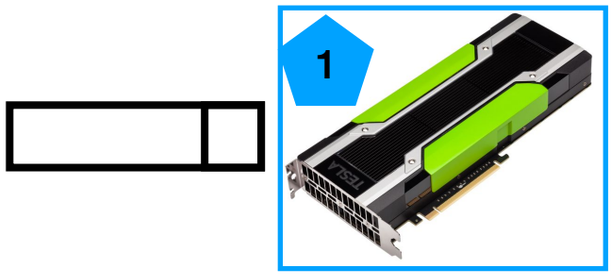




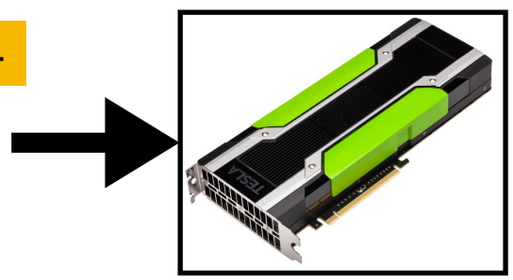


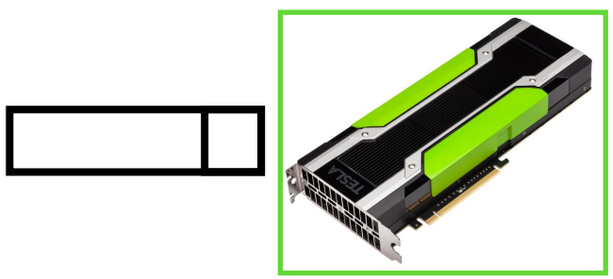
Running Example (1 Request)

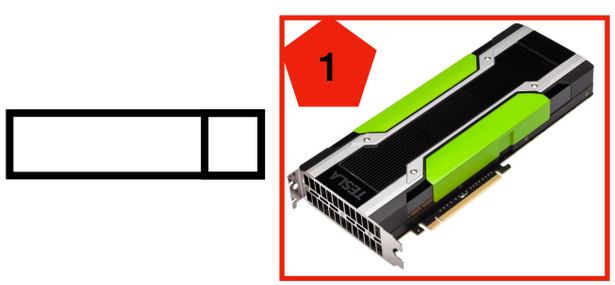
filter **search** **reduce**



8 7 4

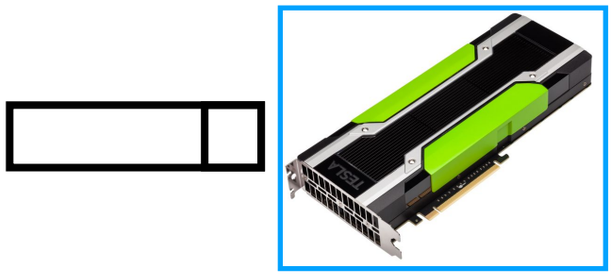


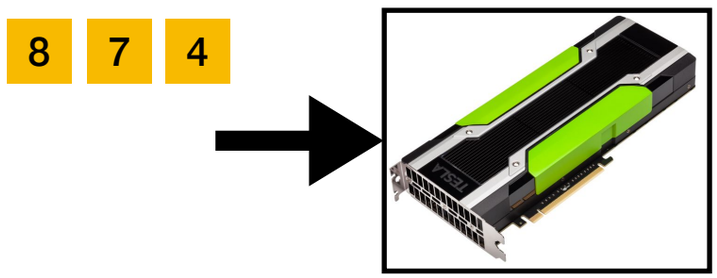


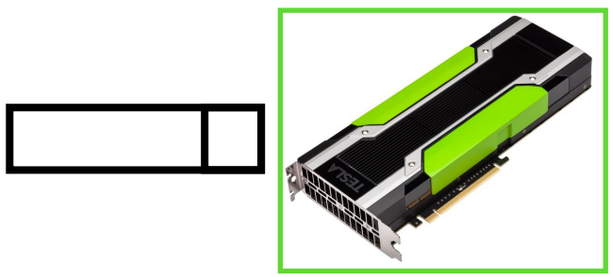


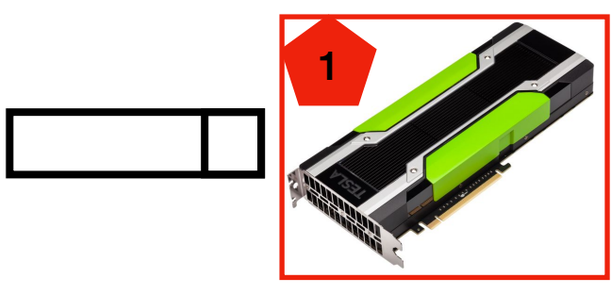
Running Example (1 Request)

filter **search** **reduce**



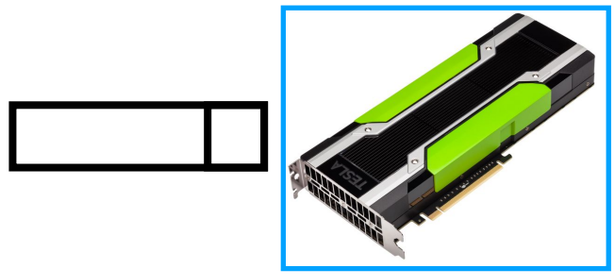


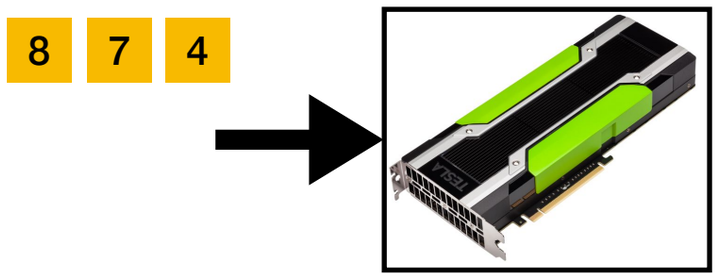


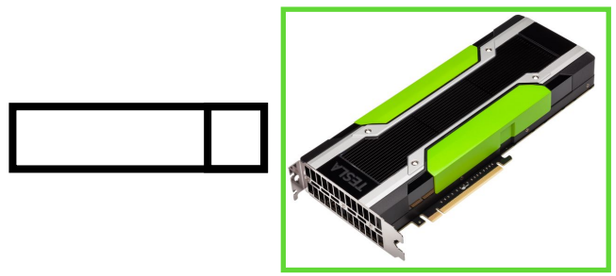


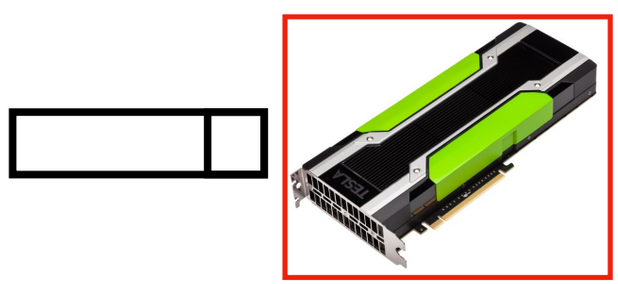
Running Example (1 Request)

filter **search** **reduce**



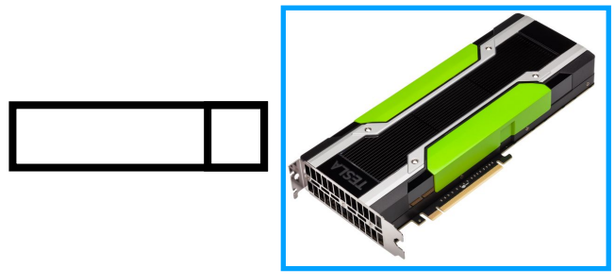


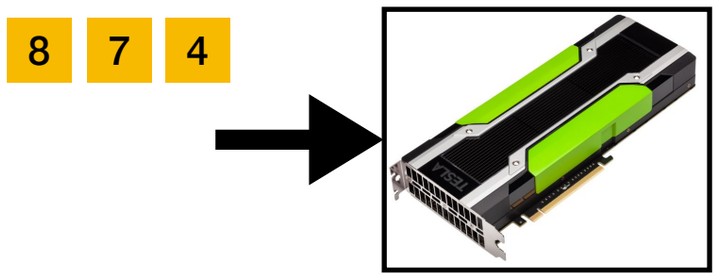


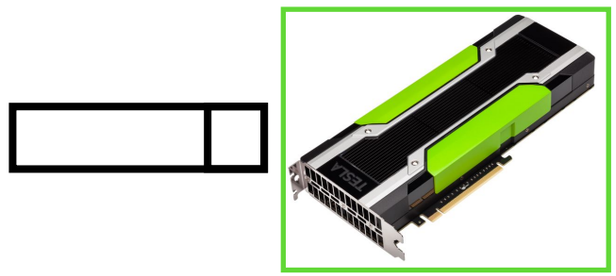


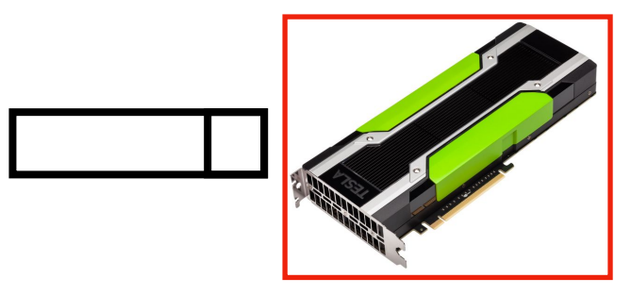
Running Example (1 Request)

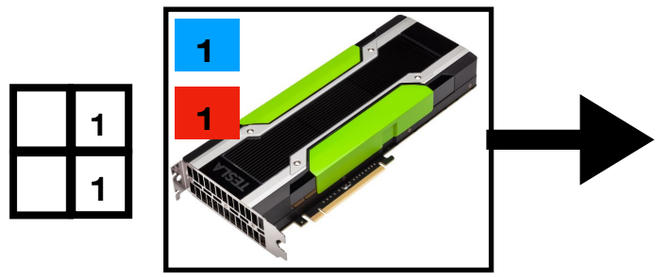
filter **search** **reduce**





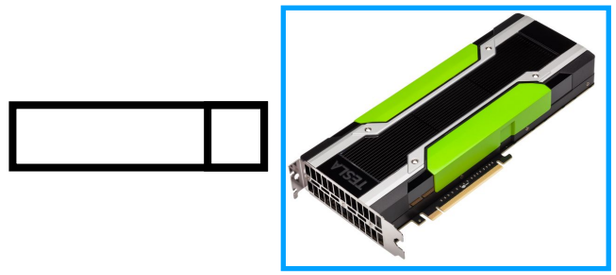




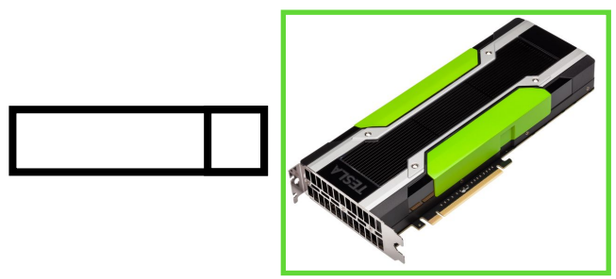


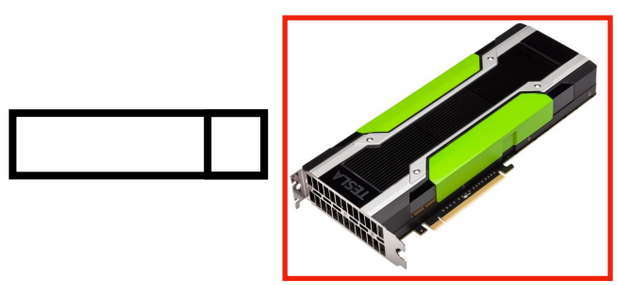
Running Example (1 Request)

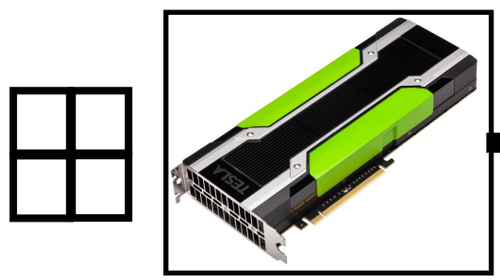
filter **search** **reduce**



8 7 4



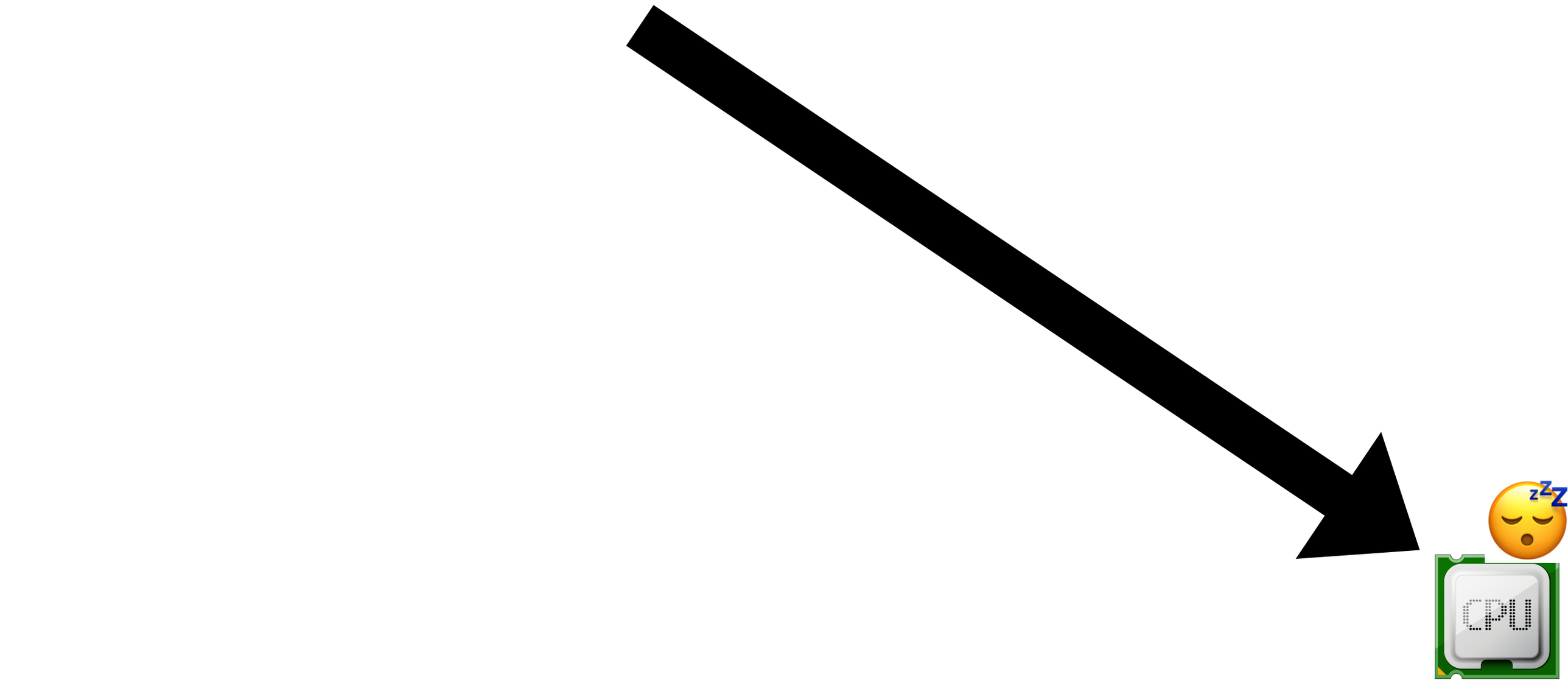




1

Where is the CPU?

Where is the CPU?



CPU's role

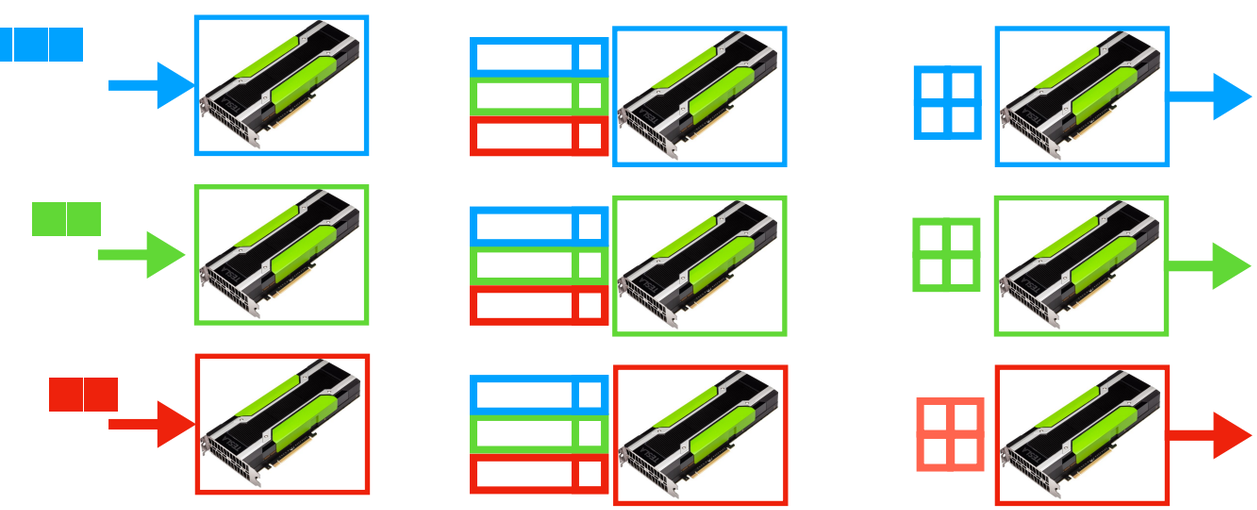
Do the setup

Then leave

More in the paper

Using same GPUs for all stages

Scalability of 1-1 gpipes as number of GPUs grow



Invocation order of kernels to take advantage of GPU scheduler

Agenda

~~Example Application: k-NN Network Server~~ 🌐

~~CPU-Driven Server Design~~ 👎🏾

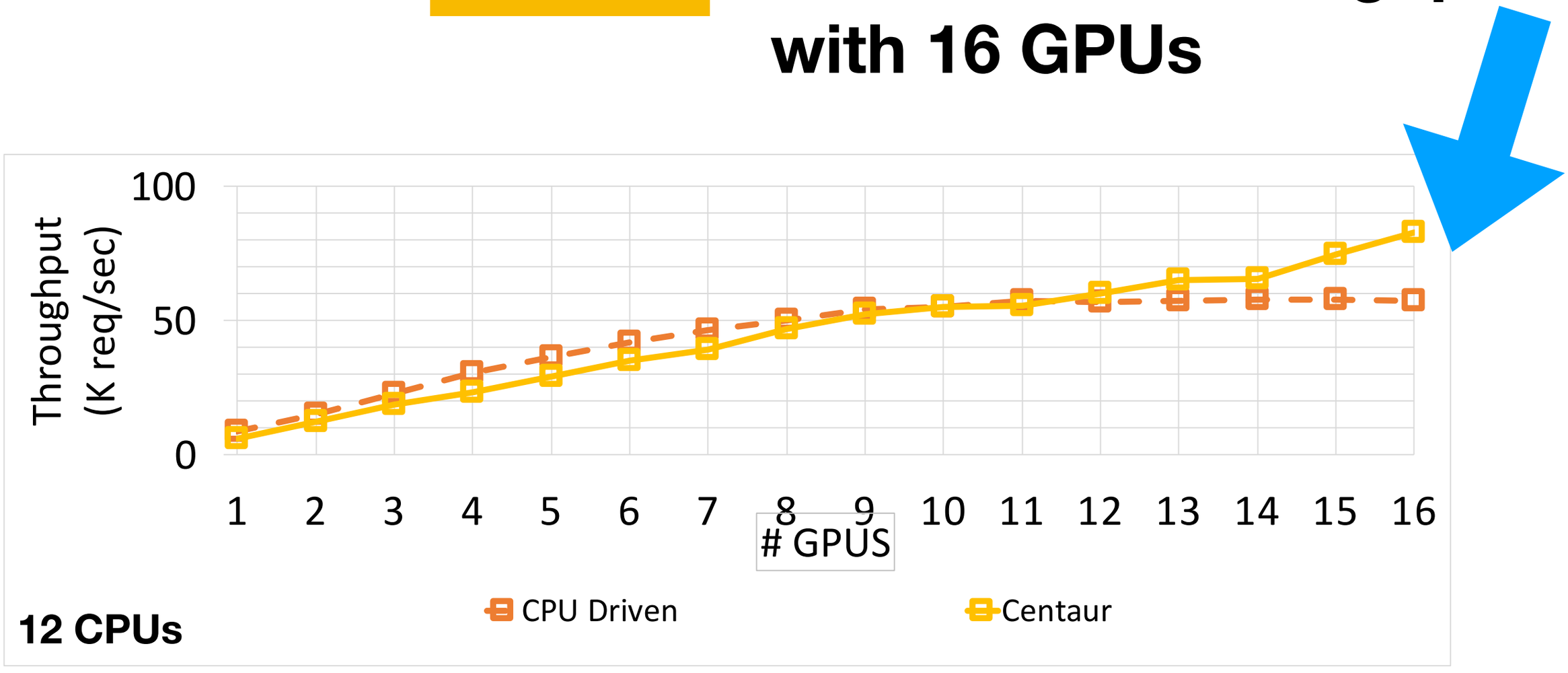
~~Analysis~~ 🔬

~~GPU Centric Design (Centaur)~~ 👍🏾

Benchmarks 📊

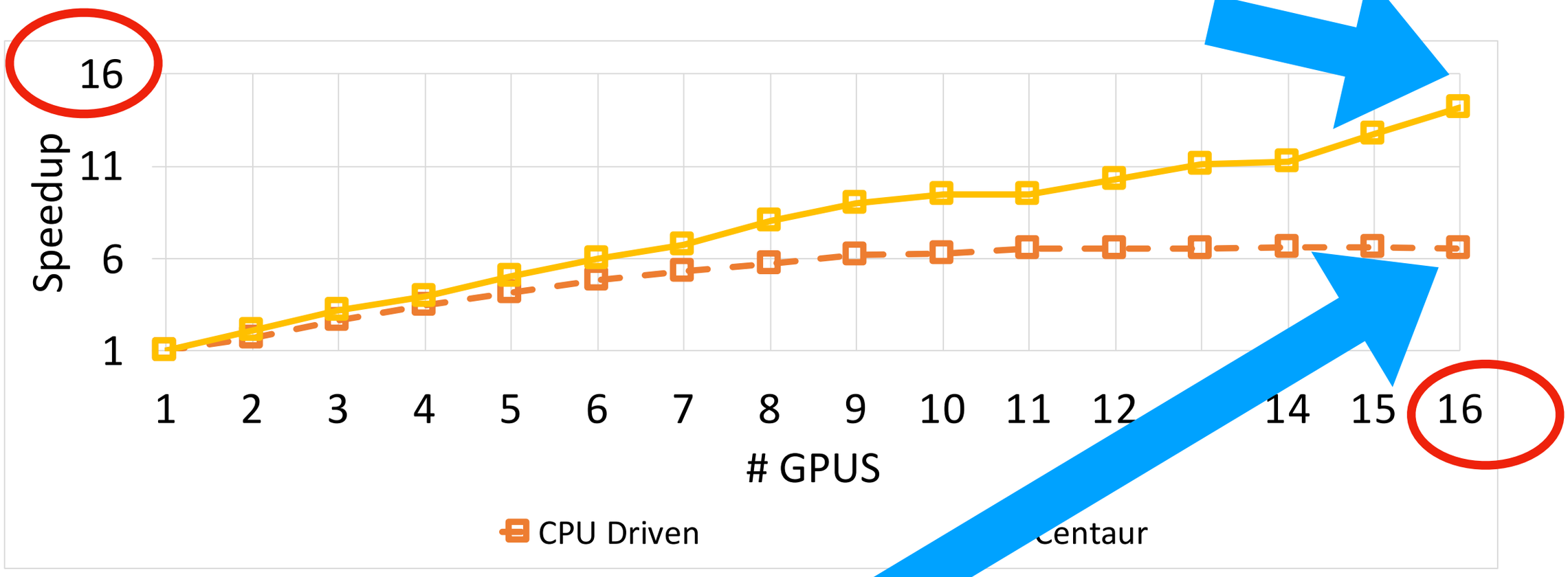
Throughput vs. GPUs

Centaur is ~45% more throughput with 16 GPUs



Throughput Scaling

Centaur: Almost perfect scaling

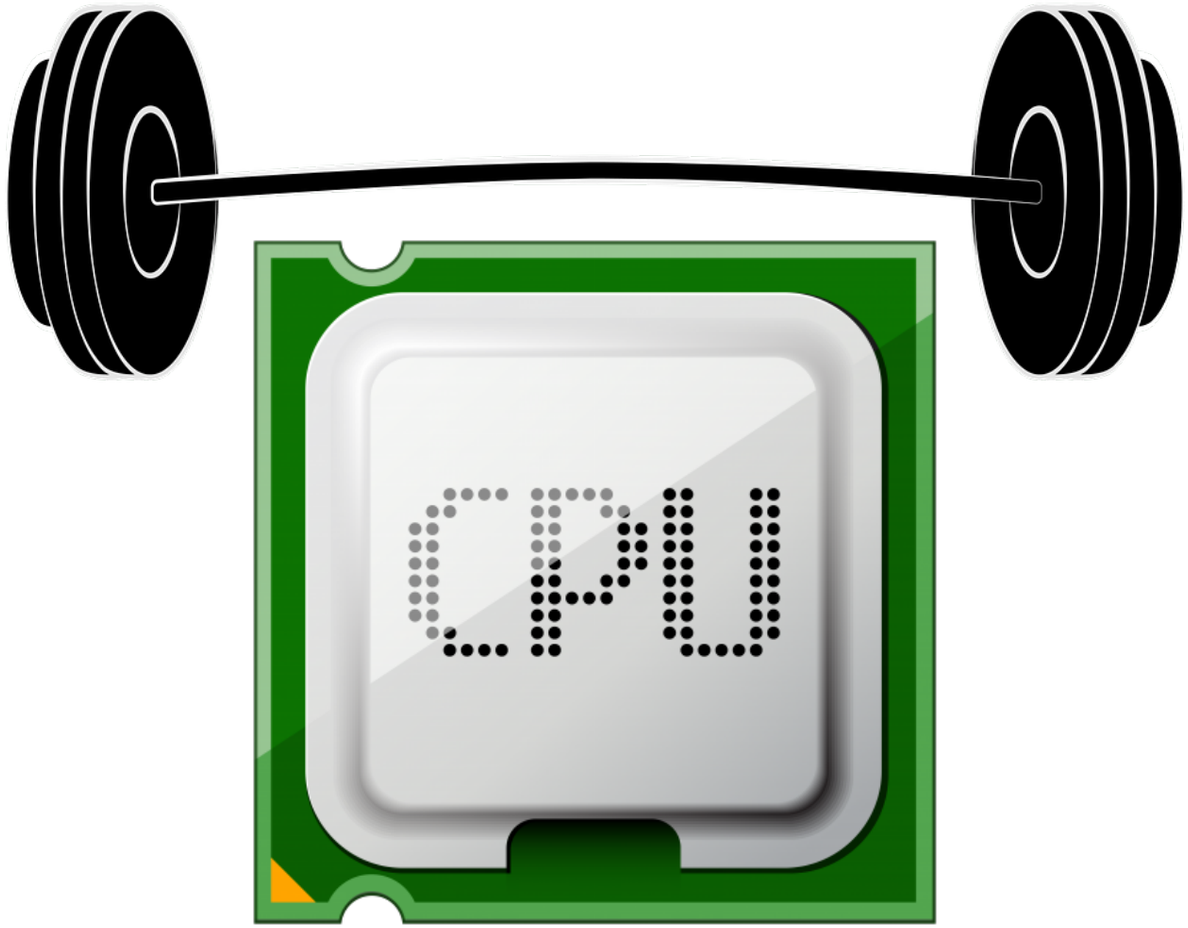


CPU-Driven: 40% of max

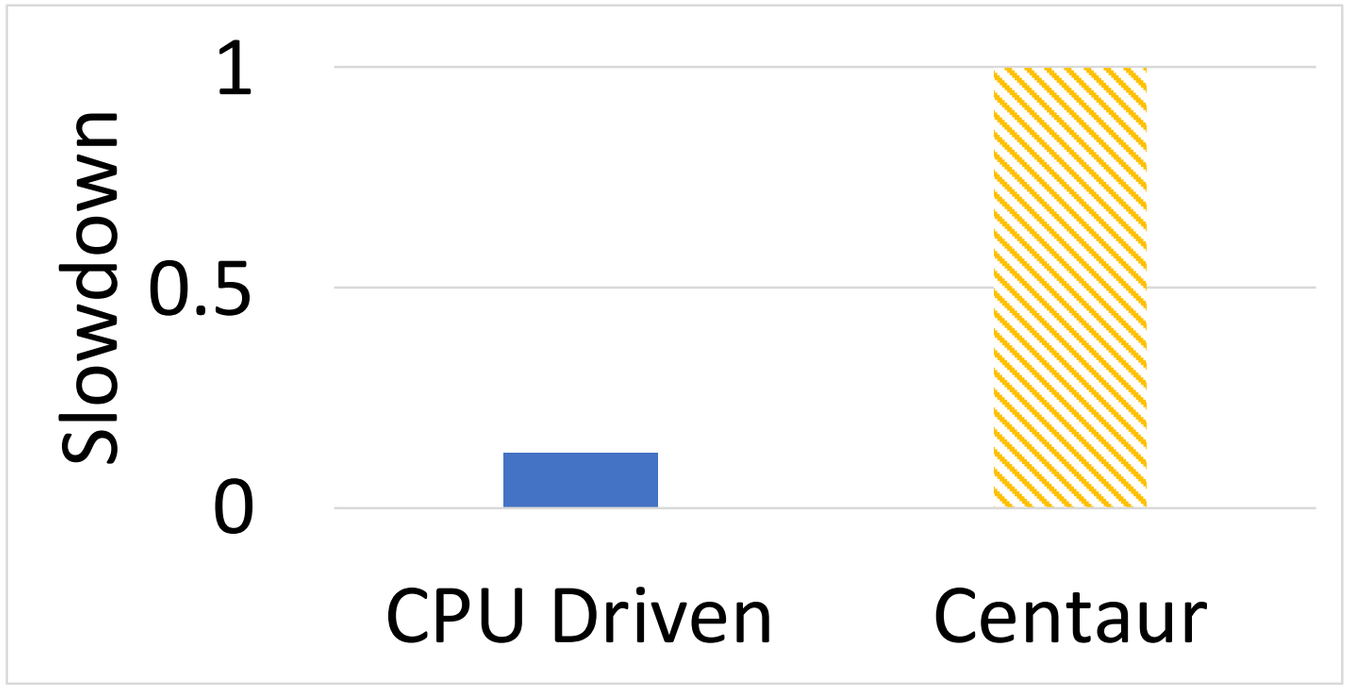
Takeaway

Removing the CPU from management allows the GPUs to show their full potential.

Co-Running with a CPU Workload



Network Server's Throughput



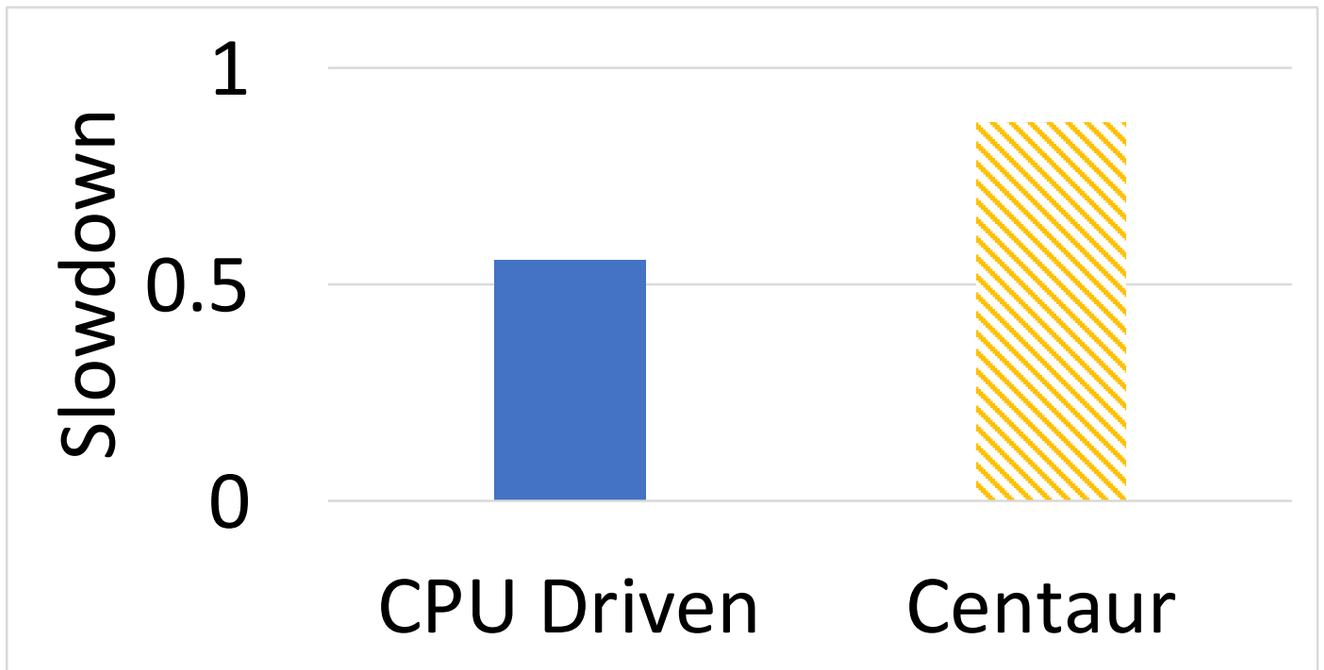
Centaur: Not Affected

CPU Driven: Significantly Slowed Down

CPU's Compute Throughput

Centaur: CPU almost unaffected

CPU Driven: CPU at ~half potential



Agenda

- ~~Example Application: k-NN Network Server~~ 🌐
- ~~CPU-Driven Server Design~~ 👎🏾
- ~~Analysis~~ 🔬
- ~~GPU Centric Design (Centaur)~~ 👍🏾
- ~~Benchmarks~~ 📊

In the Paper

Analysis of Batching Server: Latency vs. Load Imbalance

Effect of CPU Frequency

Effect of Compute Intensity

And More

Thank You