Fun with GPUs.

Vadim Markovtsev

source{d}

Machine Learning for
Large Scale Code Analysis

Plan

- NVIDIA GPGPU architecture
- CUDA environment
- multi-GPU

GPU architecture

NVIDIA GPGPU architecture

Multiple Instruction Multiple Data - CPU

- Maximum flexibility
- Low number of threads (<100)
- Low performance on parallel-friendly tasks

NVIDIA GPGPU architecture

source{d}

Multiple Instruction Multiple Data - CPU

Single Instruction Multiple Data - CPU

- SSE, AVX, etc. - all those that Go does not support
- Intel AVX512 - in Xeon Phi - 16 parallel float32 ops
- Worse flexibility
- Good performance on parallel-friendly tasks

NVIDIA GPGPU architecture

Multiple Instruction Multiple Data - CPU

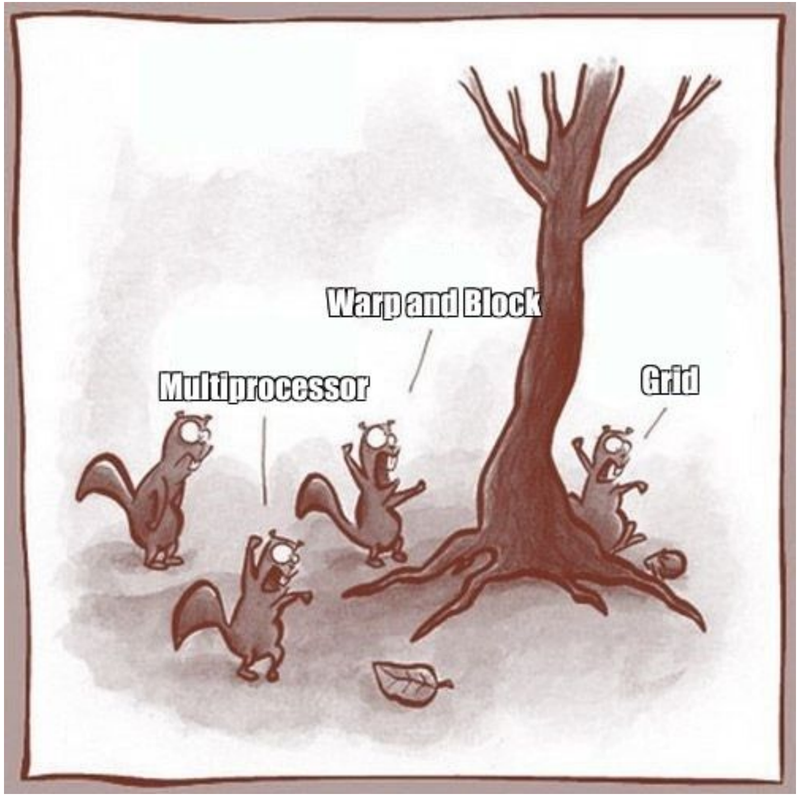
Single Instruction Multiple Data - CPU

Brain teasing mix of both - NVIDIA GPU 🤯

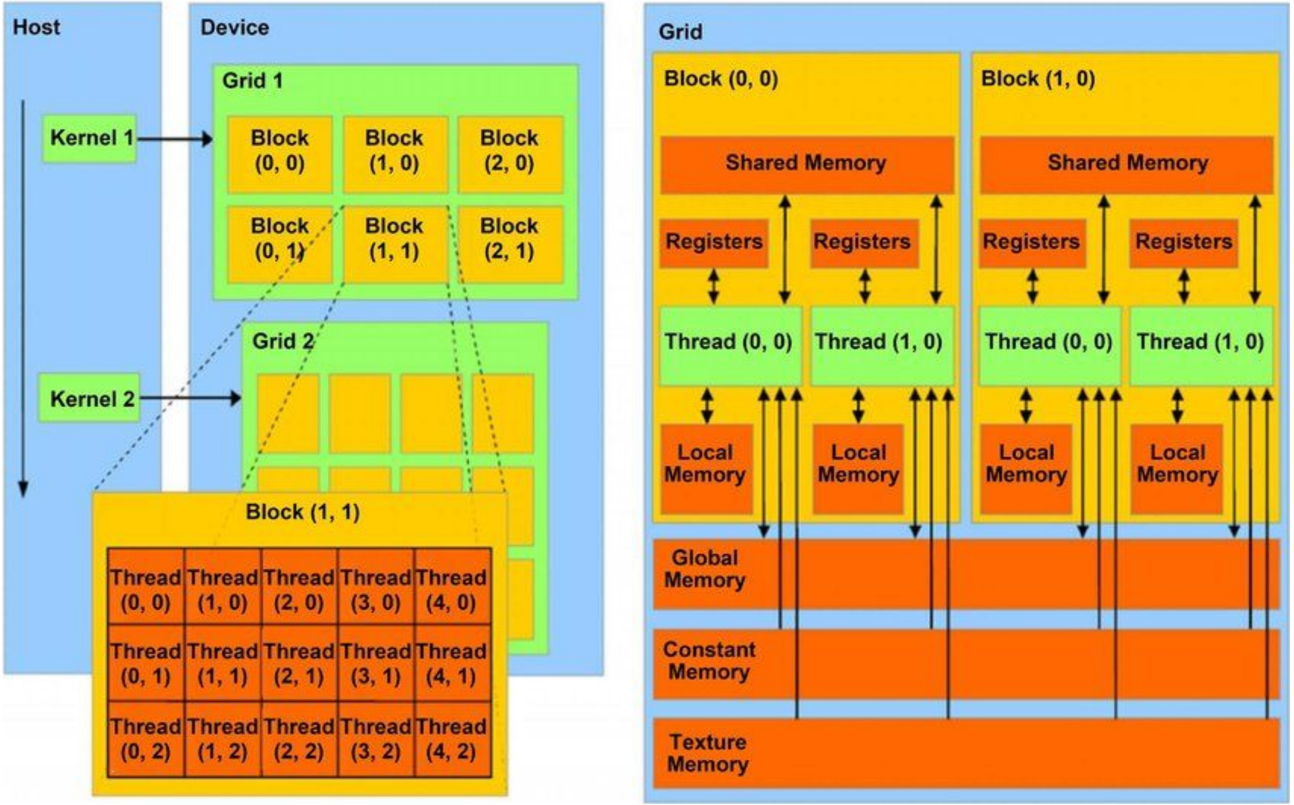
- Even worse flexibility
- Awesome performance on parallel-friendly tasks

NVIDIA GPGPU architecture

source{d}

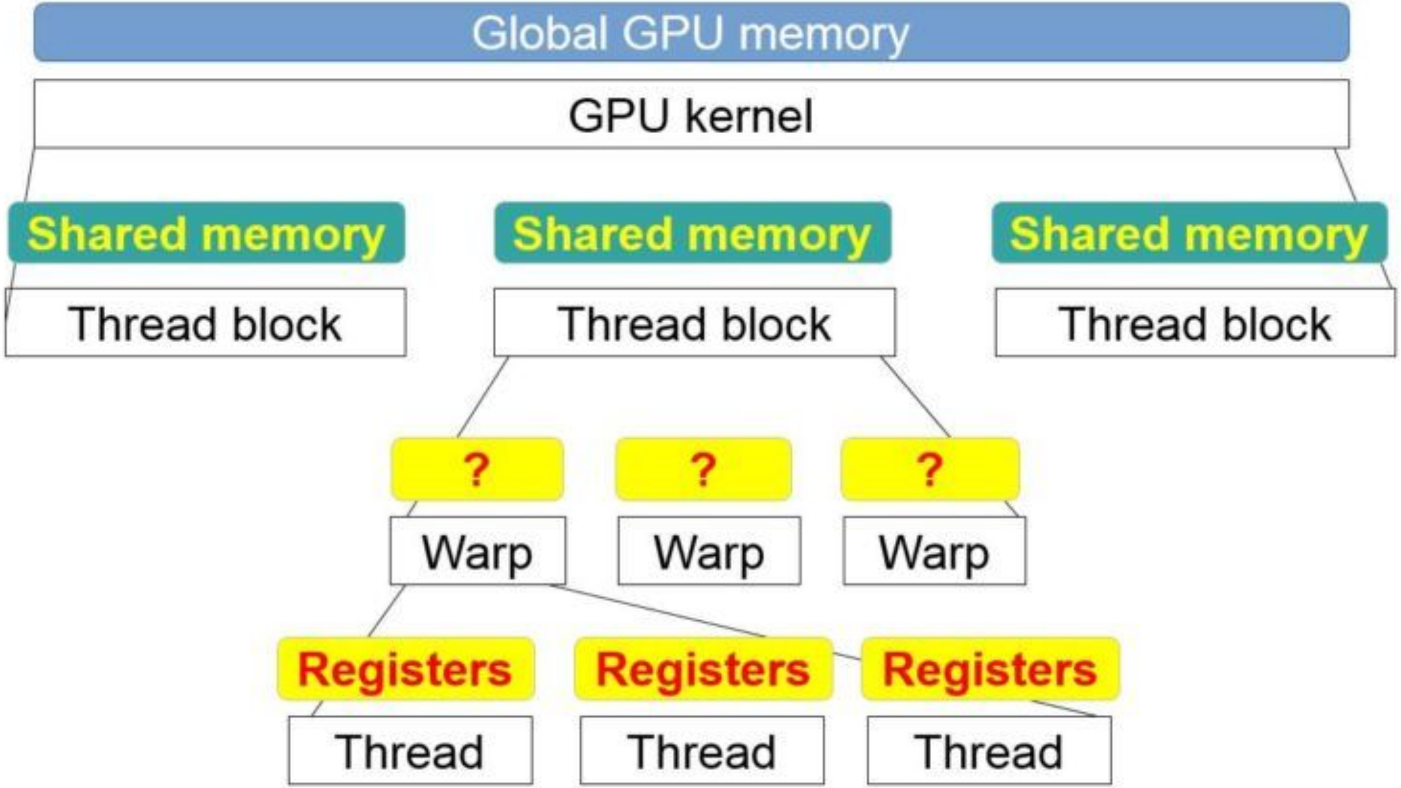


NVIDIA GPGPU architecture

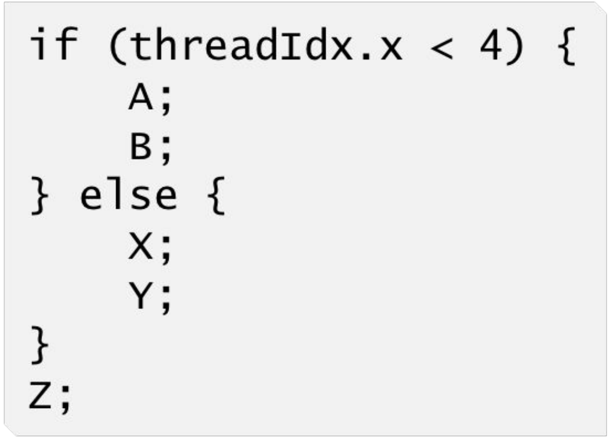


NVIDIA GPGPU architecture

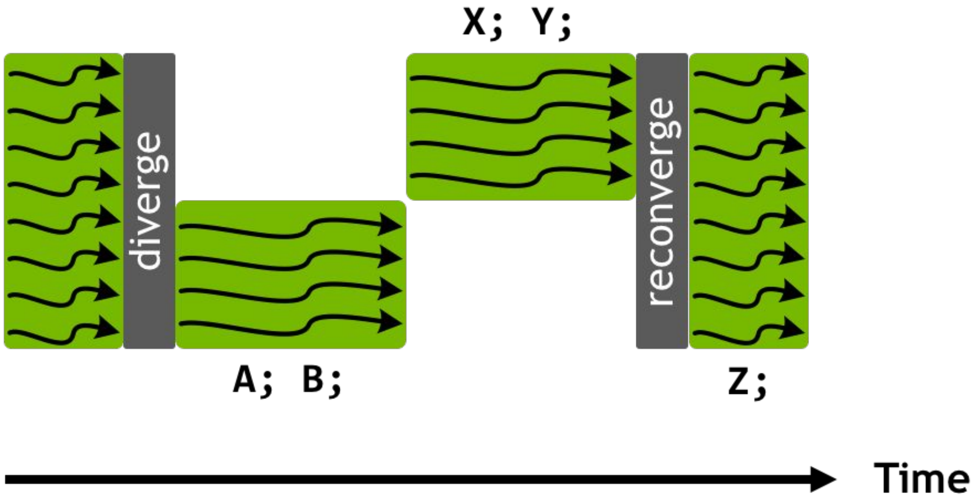
source{d}



NVIDIA GPGPU architecture

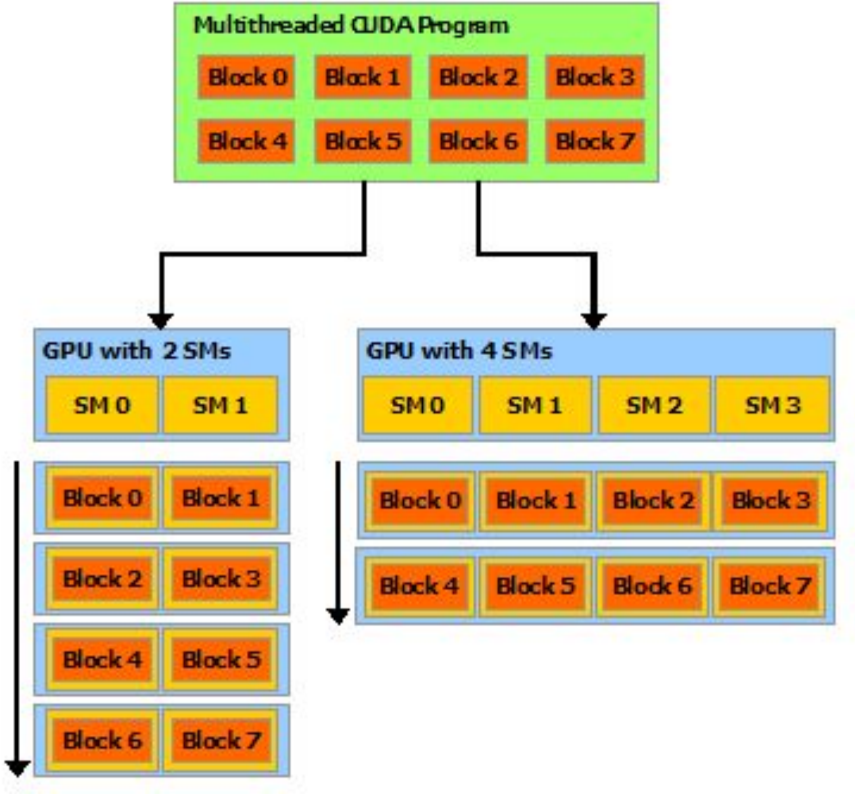


```
if (threadIdx.x < 4) {
        A;
        B;
} else {
        X;
        Y;
}
Z;
```



NVIDIA GPGPU architecture

source{d}



NVIDIA GPGPU architecture

GPU awesomeness for GPGPU = number of CUDA cores + memory size

GTX 1080 Ti (ML cluster, March 2017, $700): **3584, 11 GB**

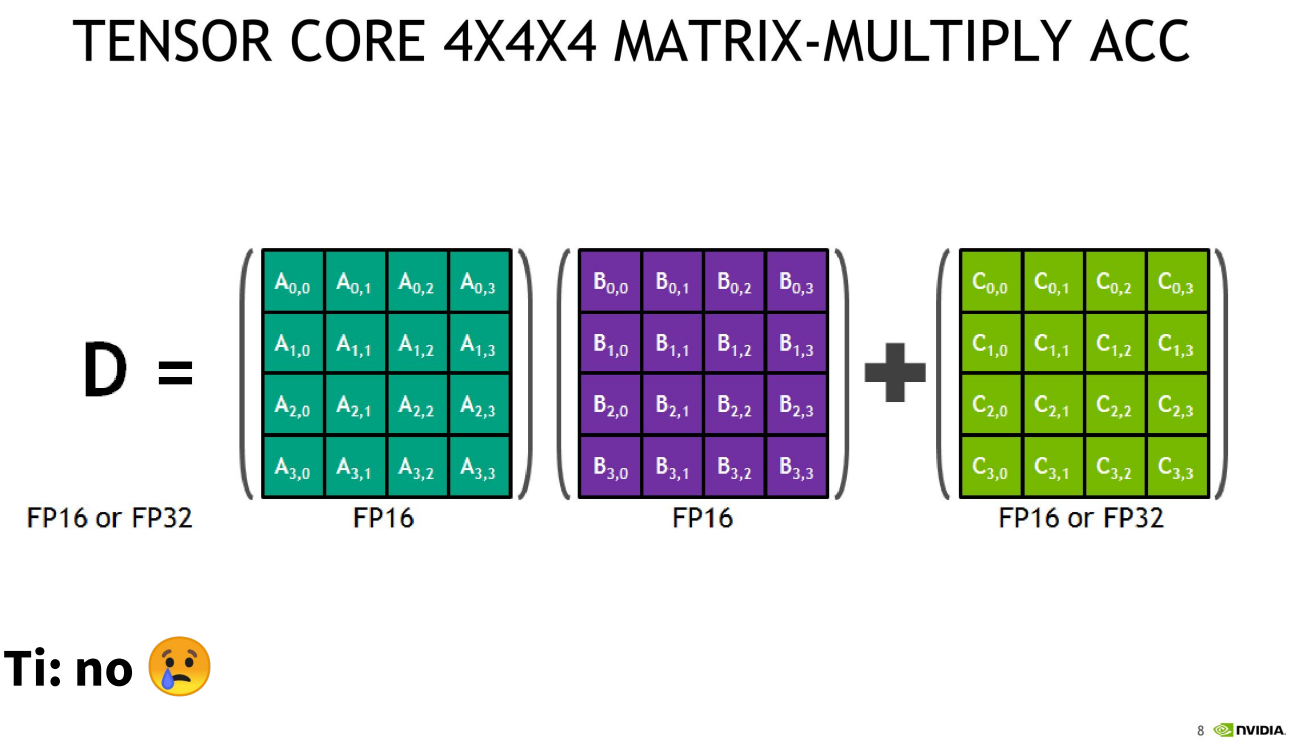
RTX 2080 Ti (September 2018, $1000): **4352, 11 GB**

Titan RTX (September 2018, $2500): **4608, 24 GB**

Tesla V100 (June 2017, $9000): **5120, 32 GB**

NVIDIA GPGPU architecture

source{d}



TENSOR CORE 4X4X4 MATRIX-MULTIPLY ACC

$$\underset{\text{FP16 or FP32}}{D} = \underset{\text{FP16}}{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}} \underset{\text{FP16}}{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}} + \underset{\text{FP16 or FP32}}{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}}$$

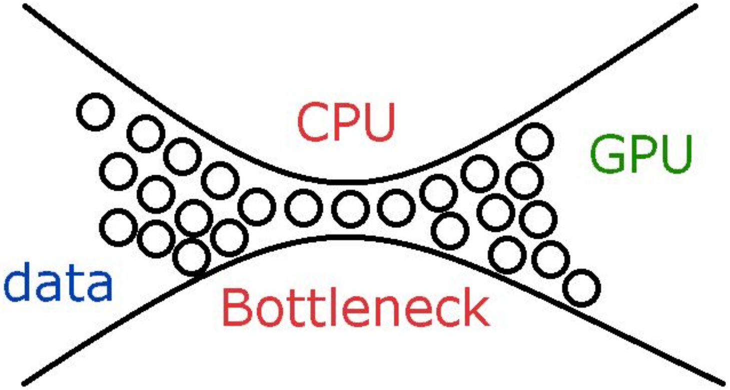
GTX 1080 Ti: no 😢

NVIDIA GPGPU architecture

Memory bandwidth:

GTX 1080 Ti: GDDR5X **484 GByte/s**

XPS 9380: LPDDR3 **43Gbyte/s** (via bandwidth)



NVIDIA GPGPU architecture

Best suited for:

- Brute force on gigabytes
- Few memory accesses, many calculations
- Aggregations

Poorly suited for:

- Serialized algorithms
- Algorithms with complex data dependencies
- Unordered memory-intensive

NVIDIA GPGPU architecture

👍Dense matrix multiplication

😐Sparse matrix multiplication

👍Image filters

👍Complex hashing

👎Diff

👎Levenshtein distance

👎Compression

😐Shortest path

😐Connected components

😐PageRank

😐Sorting (nlog^{2}n)

👍Linear search

👍Crypto

👍Physics simulation

source{d}

CUDA environment

CUDA environment

```
__global__ void addKernel(int* c, const int* a, const
int* b, int size) {
  int i = blockIdx.x * blockDim.x + threadIdx.x;
  if (i < size) {
    c[i] = a[i] + b[i];
  }
}
```

CUDA environment

```
void addWithCuda(int* c, const int* a, const int* b, int size) {
    int* dev_a = nullptr;
    int* dev_b = nullptr;
    int* dev_c = nullptr;
    // Allocate GPU buffers for three vectors (two input, one output)
    cudaMalloc((void**)&dev_c, size * sizeof(int));
    cudaMalloc((void**)&dev_a, size * sizeof(int));
    cudaMalloc((void**)&dev_b, size * sizeof(int));
    // Copy input vectors from host memory to GPU buffers.
    cudaMemcpy(dev_a, a, size * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(dev_b, b, size * sizeof(int), cudaMemcpyHostToDevice);
```

CUDA environment

```
    // Launch a kernel on the GPU with one thread for each element.

    // 2 is number of computational blocks and (size + 1) / 2 is a number of threads in a block

    addKernel<<<2, (size + 1) / 2>>>(dev_c, dev_a, dev_b, size);

    // cudaDeviceSynchronize waits for the kernel to finish, and returns

    // any errors encountered during the launch.

    cudaDeviceSynchronize();

    // Copy output vector from GPU buffer to host memory.

    cudaMemcpy(c, dev_c, size * sizeof(int), cudaMemcpyDeviceToHost);

    cudaFree(dev_c);

    cudaFree(dev_a);

    cudaFree(dev_b);

}
```

CUDA environment

source{d}

```
.cu (almost C++) -> PTXAS -> Driver -> GPU
```

CUDA environment

Two native APIs:

- Driver API - `libcuda.so` shipped with the NVIDIA driver
- Runtime API - `libcudart.so` *not* shipped with the NVIDIA driver
    - "Install CUDA"

CUDA environment

CUDA includes...

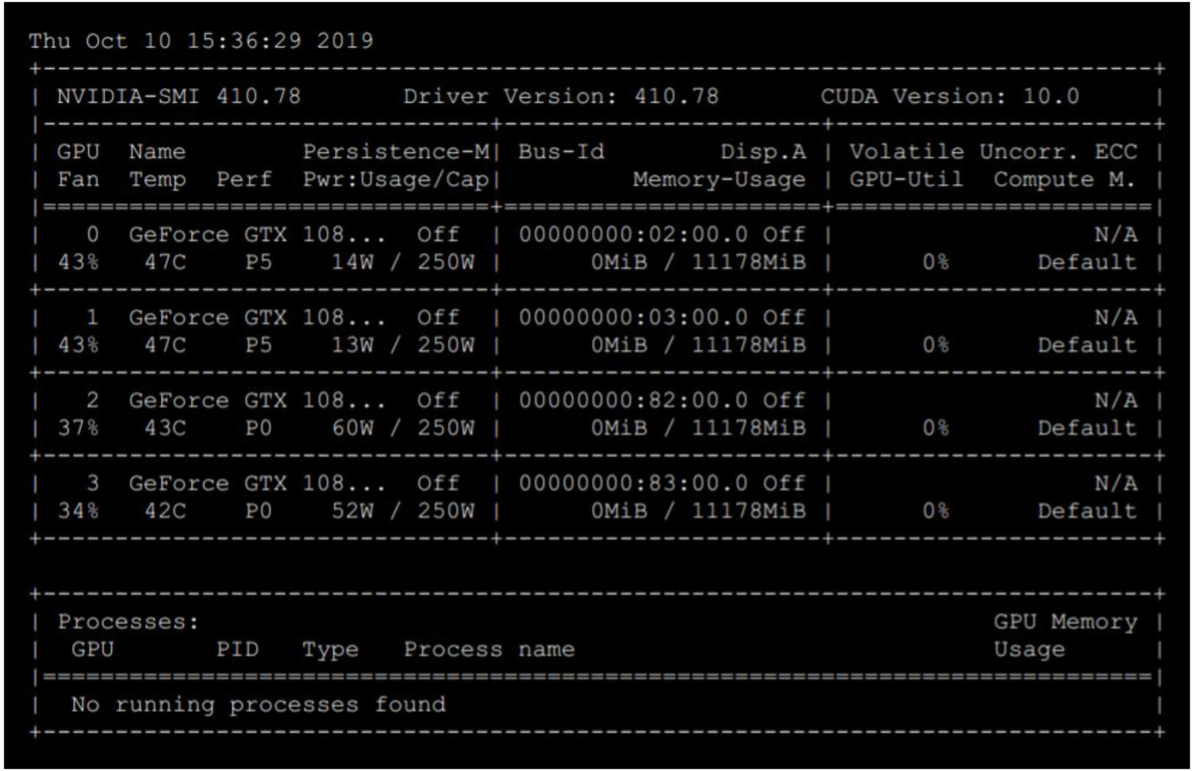
- cuFFT
- cuSPARSE
- cuSOLVER
- cuBLAS
- cuRAND
- nvJPEG
- nvGRAPH

external libs

- cuDNN
- NCCL

CUDA environment

nvidia-smi



```
Thu Oct 10 15:36:29 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.78       Driver Version: 410.78       CUDA Version: 10.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 108...  Off  | 00000000:02:00.0 Off |                  N/A |
| 43%   47C    P5    14W / 250W |      0MiB / 11178MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 108...  Off  | 00000000:03:00.0 Off |                  N/A |
| 43%   47C    P5    13W / 250W |      0MiB / 11178MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GeForce GTX 108...  Off  | 00000000:82:00.0 Off |                  N/A |
| 37%   43C    P0    60W / 250W |      0MiB / 11178MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GeForce GTX 108...  Off  | 00000000:83:00.0 Off |                  N/A |
| 34%   42C    P0    52W / 250W |      0MiB / 11178MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

CUDA environment

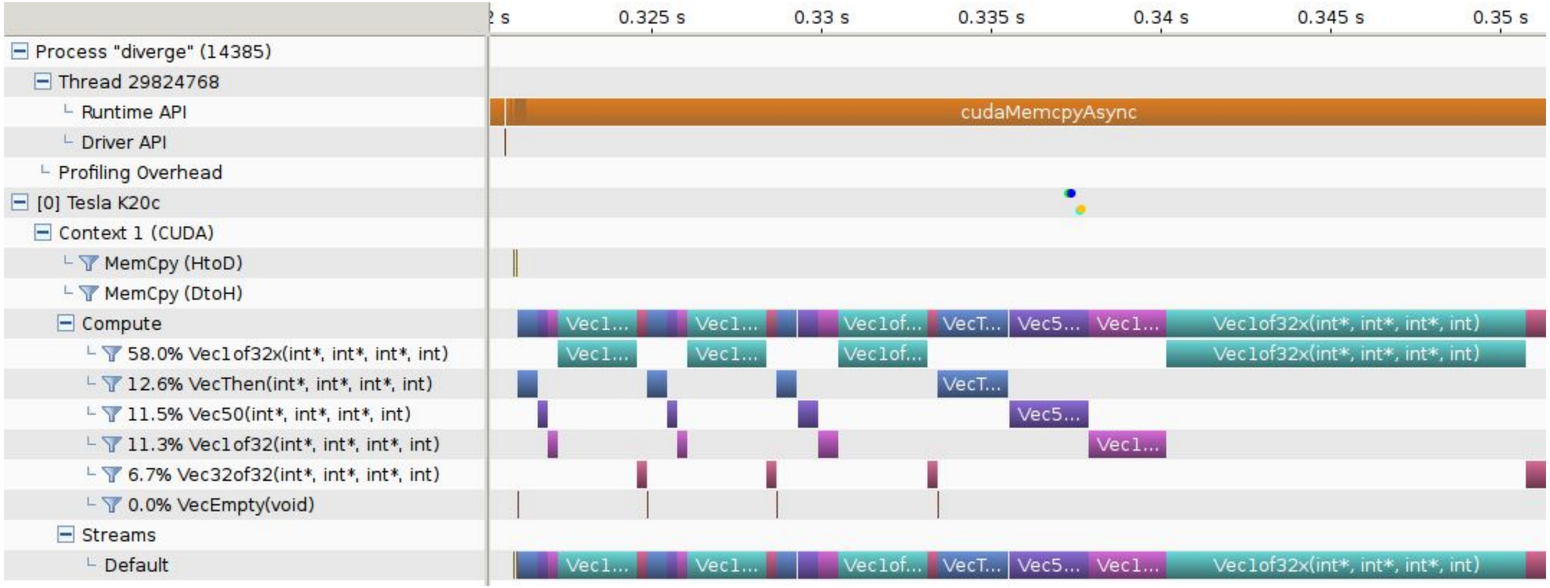
CUDA environment

cuda-memcheck

- Dynamic program analysis (like valgrind)
- Memory access
- Races
- Synchronization errors

CUDA environment

nvprof



CUDA environment

Frameworks:

- Tensorflow
- Python: cupy/cupy: `import cupy as numpy`
- Go Driver API: gorgonia/cu - renamed APIs
- Go Driver API: barnex/cuda5 - panics on errors

CUDA environment

```
import tensorflow as tf
print(tf.reduce_sum(tf.convert_to_tensor([0, 1, 2])).numpy())
```

CUDA environment

```
2019-10-10 19:28:52.224720: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcuda.so.1
2019-10-10 19:28:52.247101: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 0 with properties:
name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.62 pciBusID: 0000:02:00.0
2019-10-10 19:28:52.248367: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 1 with properties:
name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.62 pciBusID: 0000:03:00.0
2019-10-10 19:28:52.249596: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 2 with properties:
name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.62 pciBusID: 0000:82:00.0
2019-10-10 19:28:52.250883: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 3 with properties:
name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.62 pciBusID: 0000:83:00.0
2019-10-10 19:28:52.251200: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamicl ibrary libcudart.so.10.0
2019-10-10 19:28:52.252649: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10.0
2019-10-10 19:28:52.254000: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcufft.so.10.0
2019-10-10 19:28:52.254378: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcurand.so.10.0
2019-10-10 19:28:52.256244: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcusolver.so.10.0
2019-10-10 19:28:52.257719: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcusparse.so.10.0
2019-10-10 19:28:52.262122: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudnn.so.7
2019-10-10 19:28:52.271414: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1746] Adding visible gpu devices: 0, 1, 2, 3
```

CUDA environment

```
2019-10-10 19:28:52.272033: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not
compiled to use: AVX2 FMA

2019-10-10 19:28:52.297657: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2099785000 Hz

2019-10-10 19:28:52.301471: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x688b880 executing computations on platform Host. Devices:

2019-10-10 19:28:52.301500: I tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor device (0): Host, Default Version

2019-10-10 19:28:52.930088: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x68edbb0 executing computations on platform CUDA. Devices:

2019-10-10 19:28:52.930116: I tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor device (0): GeForce GTX 1080 Ti, Compute Capability 6.1

2019-10-10 19:28:52.930140: I tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor device (1): GeForce GTX 1080 Ti, Compute Capability 6.1

2019-10-10 19:28:52.930148: I tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor device (2): GeForce GTX 1080 Ti, Compute Capability 6.1

2019-10-10 19:28:52.930154: I tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor device (3): GeForce GTX 1080 Ti, Compute Capability 6.1
```

CUDA environment

```
2019-10-10 19:28:52.930154: I tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor device (3): GeForce GTX 1080 Ti, Compute Capability 6.1

2019-10-10 19:28:52.947207: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1159] Device interconnect StreamExecutorwith strength 1 edge matrix:

2019-10-10 19:28:52.947222: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1165]      0 1 2 3

2019-10-10 19:28:52.947246: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 0:   N Y N N

2019-10-10 19:28:52.947252: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 1:   Y N N N

2019-10-10 19:28:52.947258: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 2:   N N N Y

2019-10-10 19:28:52.947264: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 3:   N N Y N

2019-10-10 19:28:52.952011: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1304] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:0 with 10313 MB memory) => physical GPU (device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:02:00.0,
compute capability: 6.1)

2019-10-10 19:28:52.953481: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1304] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:1 with 10470 MB memory) => physical GPU (device: 1, name: GeForce GTX 1080 Ti, pci bus id: 0000:03:00.0,
compute capability: 6.1)

2019-10-10 19:28:52.954857: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1304] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:2 with 10470 MB memory) => physical GPU (device: 2, name: GeForce GTX 1080 Ti, pci bus id: 0000:82:00.0,
compute capability: 6.1)

2019-10-10 19:28:52.956627: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1304] Created TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:3 with 10470 MB memory) => physical GPU (device: 3, name: GeForce GTX 1080 Ti, pci bus id: 0000:83:00.0,
compute capability: 6.1)
```

CUDA environment

source{d}

3

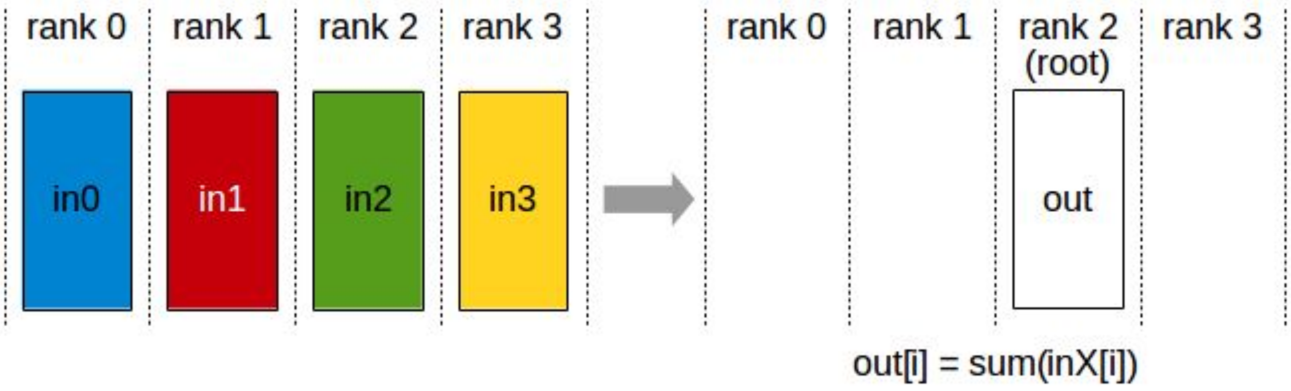
CUDA environment

github.com/NVIDIA/cuda-samples

multi-GPU

multi-GPU

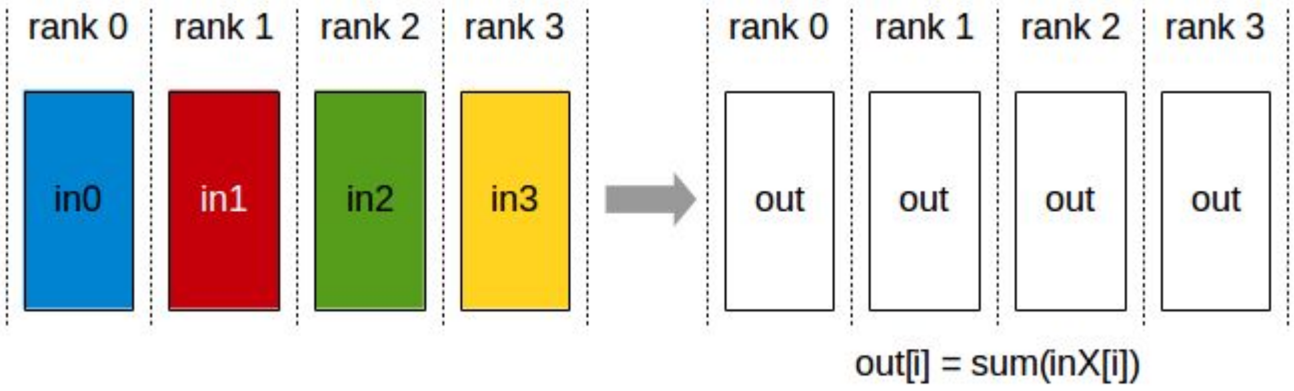
Reduce



multi-GPU

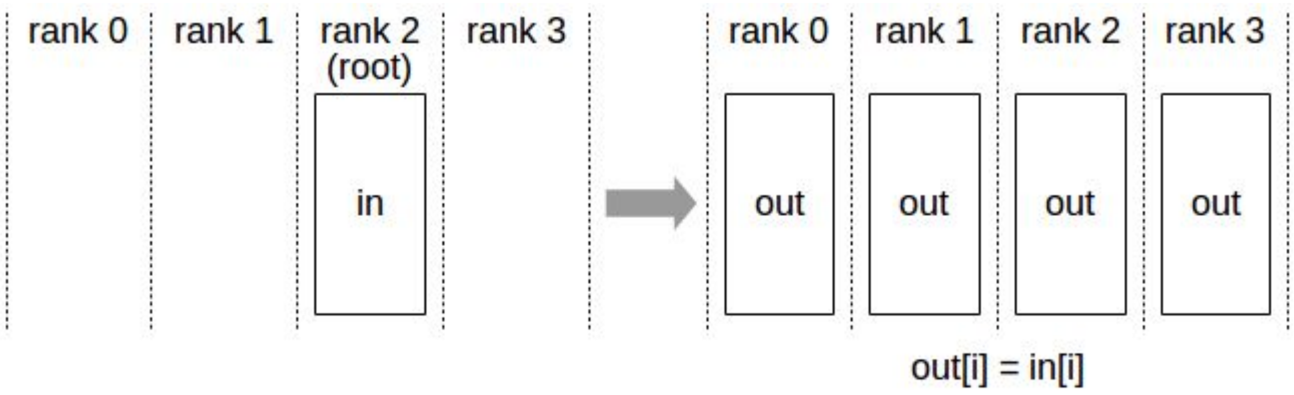
source{d}

AllReduce



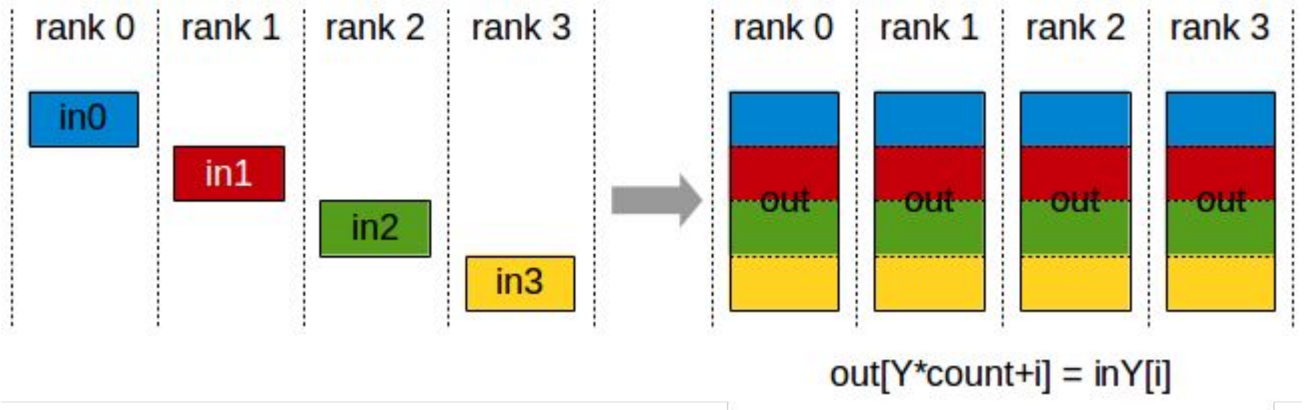
multi-GPU

Broadcast



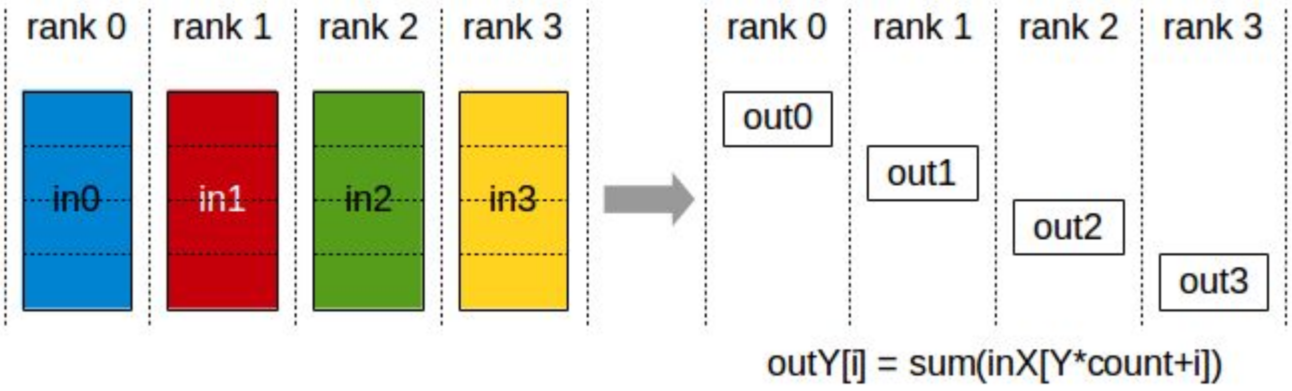
multi-GPU

AllGather

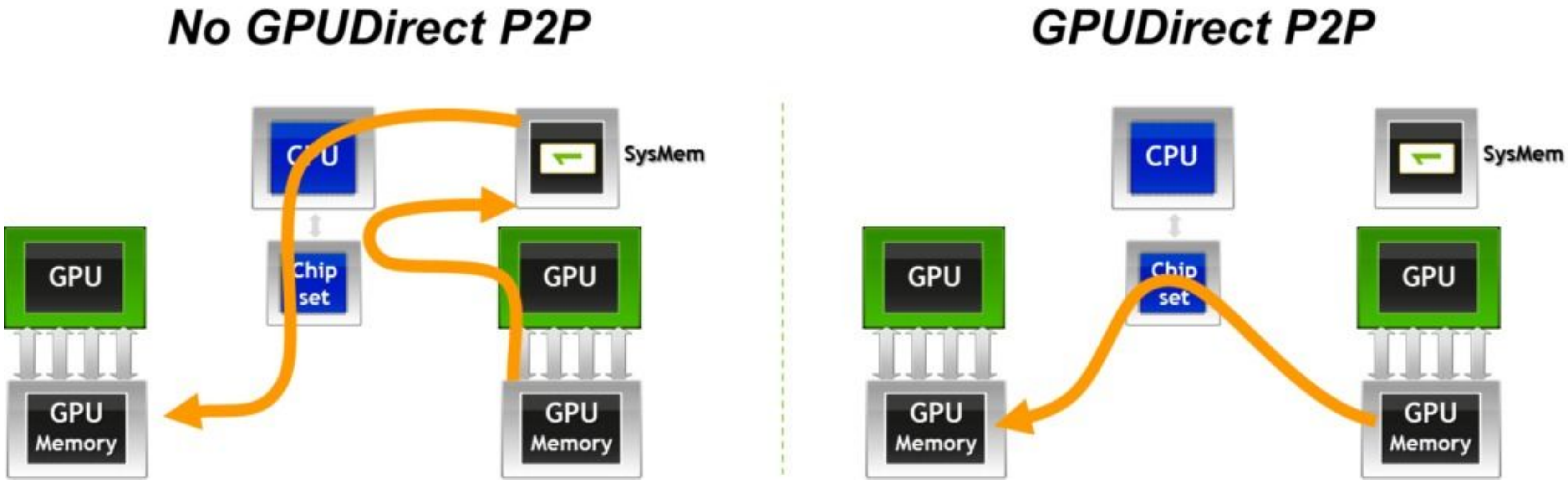


multi-GPU

ReduceScatter

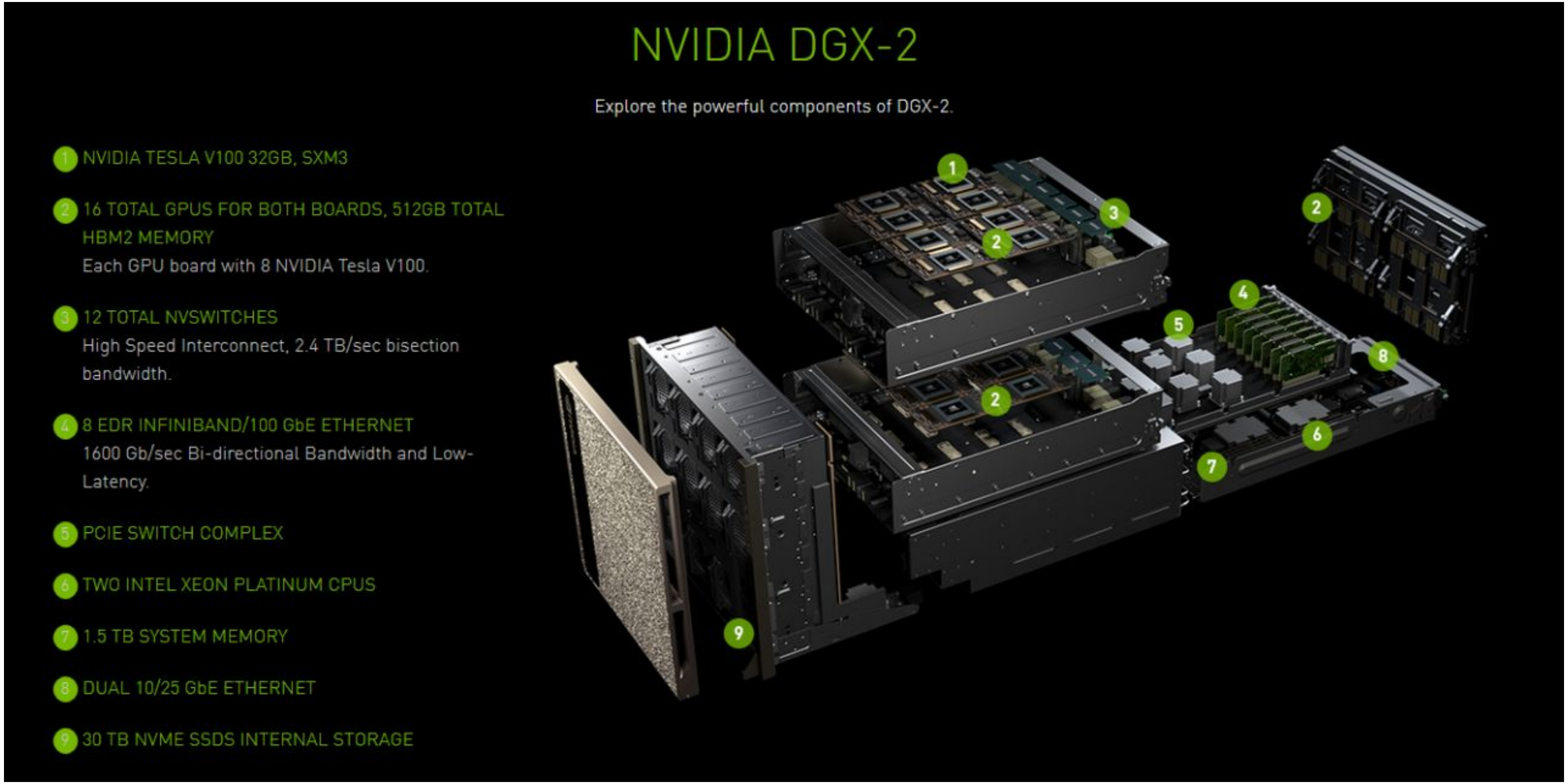


multi-GPU


No GPUDirect P2P
CPU
SysMem
Chip set
GPU
GPU Memory
GPU
GPU Memory
GPUDirect P2P
CPU
SysMem
Chip set
GPU
GPU Memory
GPU
GPU Memory

multi-GPU

source{d}



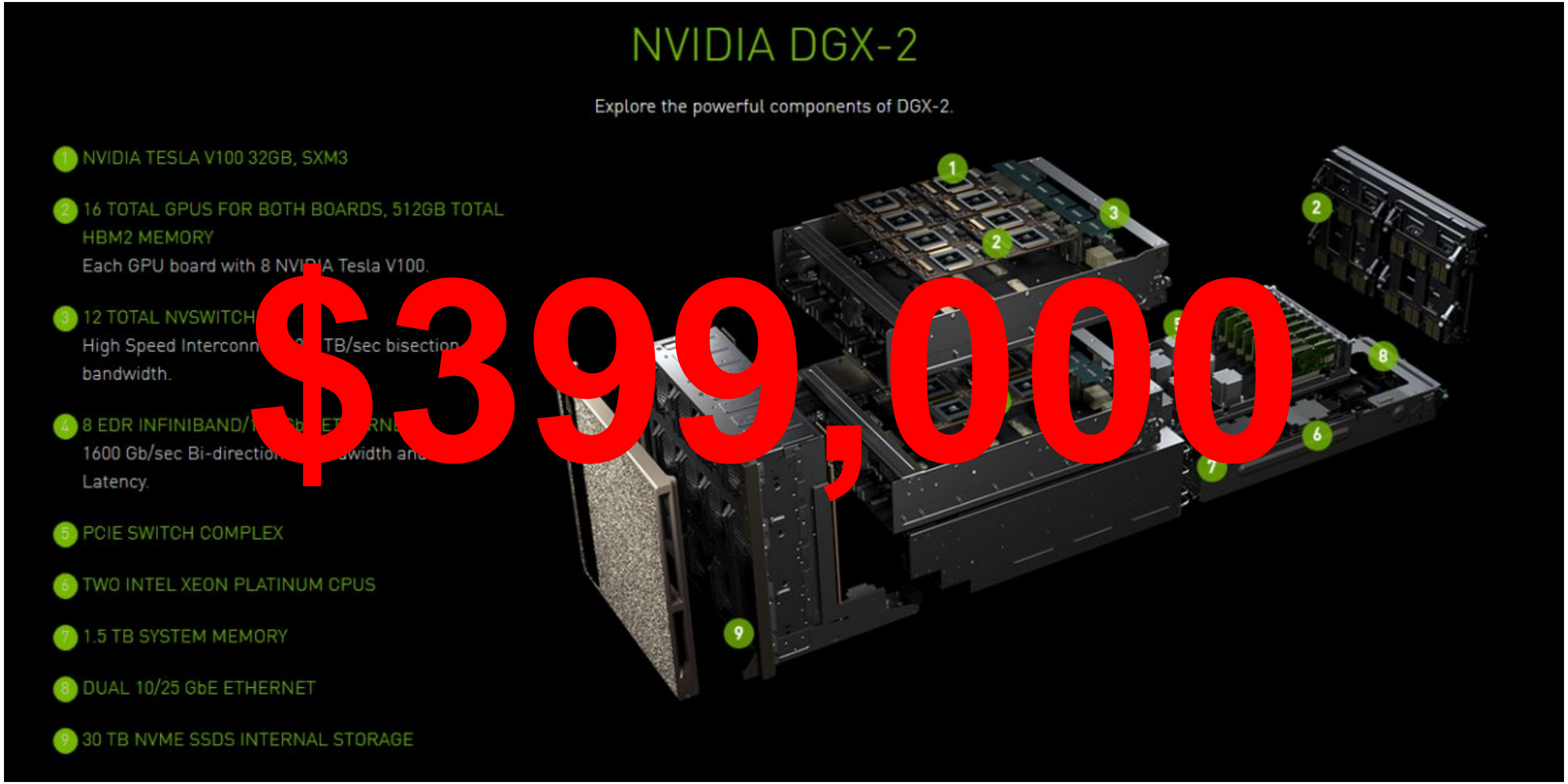
NVIDIA DGX-2

Explore the powerful components of DGX-2.

1. NVIDIA TESLA V100 32GB, SXM3
2. 16 TOTAL GPUS FOR BOTH BOARDS, 512GB TOTAL HBM2 MEMORY
   Each GPU board with 8 NVIDIA Tesla V100.
3. 12 TOTAL NVSWITCHES
   High Speed Interconnect, 2.4 TB/sec bisection bandwidth.
4. 8 EDR INFINIBAND/100 GbE ETHERNET
   1600 Gb/sec Bi-directional Bandwidth and Low-Latency.
5. PCIE SWITCH COMPLEX
6. TWO INTEL XEON PLATINUM CPUS
7. 1.5 TB SYSTEM MEMORY
8. DUAL 10/25 GbE ETHERNET
9. 30 TB NVME SSDS INTERNAL STORAGE

multi-GPU

source{d}



NVIDIA DGX-2

Explore the powerful components of DGX-2.

1. NVIDIA TESLA V100 32GB, SXM3
2. 16 TOTAL GPUS FOR BOTH BOARDS, 512GB TOTAL HBM2 MEMORY
   Each GPU board with 8 NVI A Tesla V100.
3. 12 TOTAL NVSWITCH
   High Speed Interconn TB/sec bisection bandwidth.
4. 8 EDR INFINIBAND/1 Gb ET RN
   1600 Gb/sec Bi-directio width an Latency.
5. PCIE SWITCH COMPLEX
6. TWO INTEL XEON PLATINUM CPUS
7. 1.5 TB SYSTEM MEMORY
8. DUAL 10/25 GbE ETHERNET
9. 30 TB NVME SSDS INTERNAL STORAGE

$399,000

multi-GPU

Scalable

Link

Interface

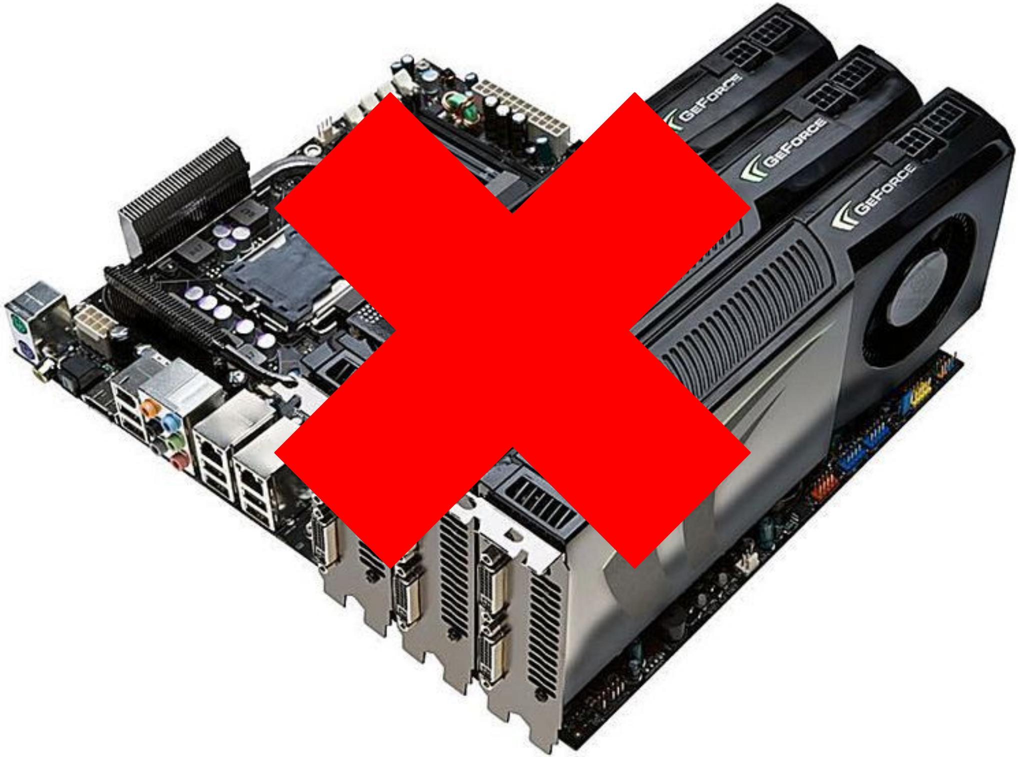
multi-GPU

source{d}

Scalable

Link

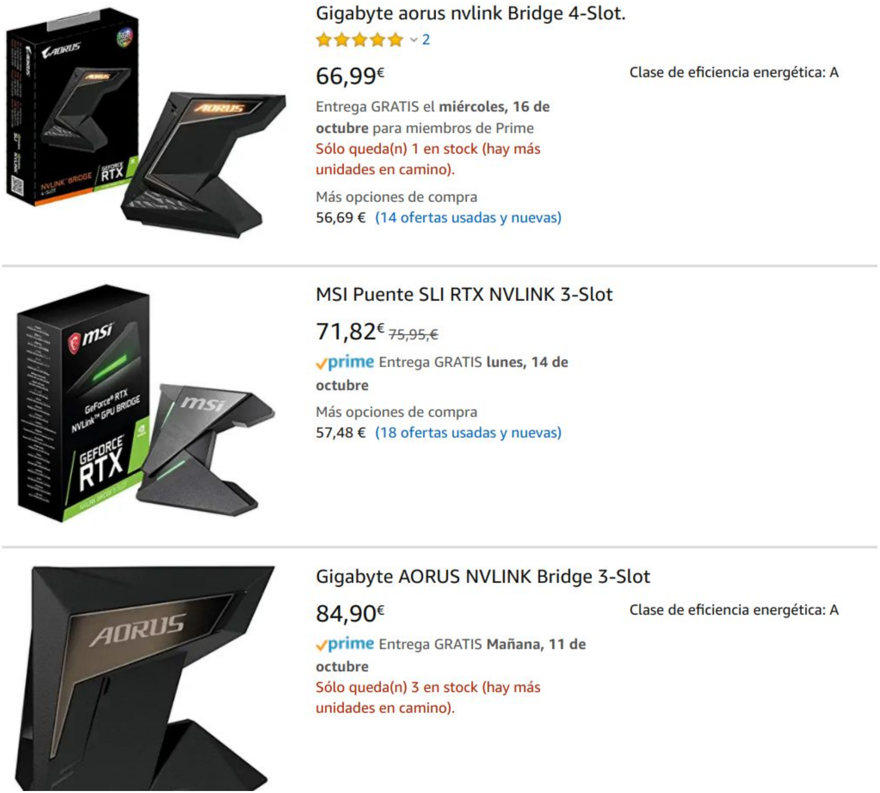
Interface



multi-GPU

NVLink

multi-GPU



Gigabyte aorus nvlink Bridge 4-Slot.

★★★★★ 2

66,99€

Clase de eficiencia energética: A

Entrega GRATIS el **miércoles, 16 de octubre** para miembros de Prime

Sólo queda(n) 1 en stock (hay más unidades en camino).

Más opciones de compra

56,69 € (14 ofertas usadas y nuevas)

MSI Puente SLI RTX NVLINK 3-Slot

71,82€ ~~75,95 €~~

✓prime Entrega GRATIS **lunes, 14 de octubre**

Más opciones de compra

57,48 € (18 ofertas usadas y nuevas)

Gigabyte AORUS NVLINK Bridge 3-Slot

84,90€

Clase de eficiencia energética: A

✓prime Entrega GRATIS **Mañana, 11 de octubre**

Sólo queda(n) 3 en stock (hay más unidades en camino).

multi-GPU

source{d}

NVLink

multi-GPU

PCI Express regular vs peer to peer, ML cluster*:

Speed: 20 GB/s vs 20 GB/s

Latency: 12us vs 1.0us

*1 month to achieve this

multi-GPU

blog.sourced.tech/post/multi-gpu-deep-learning

Thank you!

source{d}

Machine Learning for Large Scale Code Analysis

sourced.tech · github.com/src-d · @sourcedtech · blog.sourced.tech