GPU programming made easy with OpenMP

Aditya Nitsure (anitsure@in.ibm.com)

Pidad D'Souza (pidsouza@in.ibm.com)

IBM

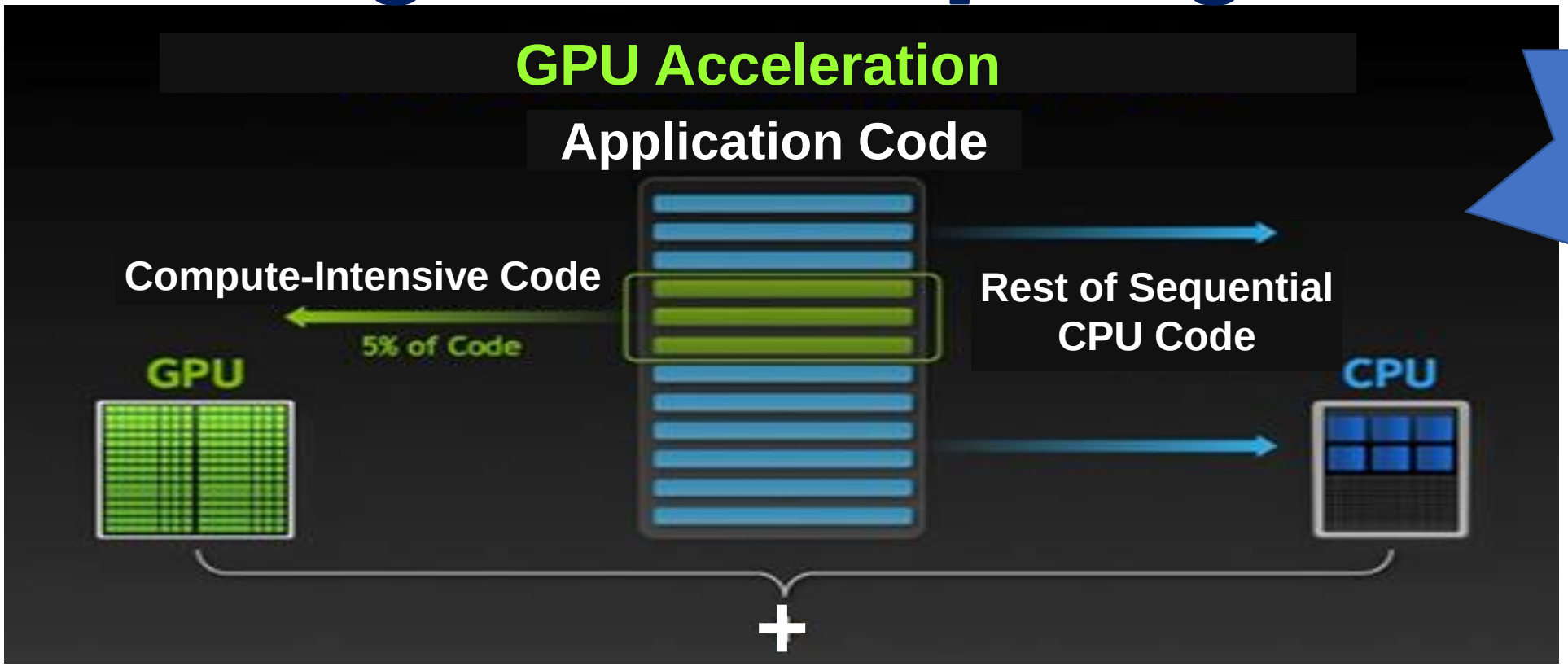
Tutorial Overview

- Heterogeneous system overview
- Introduction to parallel programming
- OpenMP programming on CPU and GPU
- Profiling and monitoring
- HPC application performance

- Hands-on

Heterogenous Systems

Heterogenous Computing



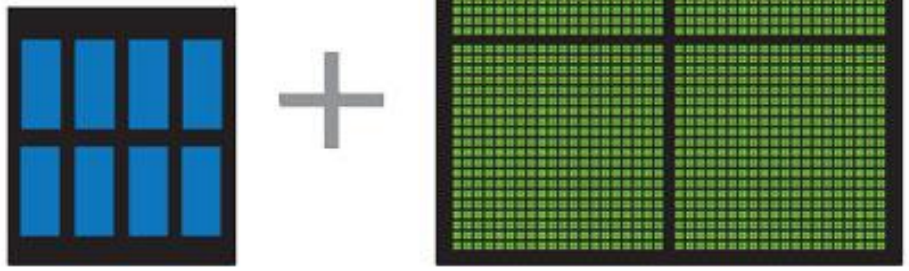
Maximize performance and energy efficiency

CPU

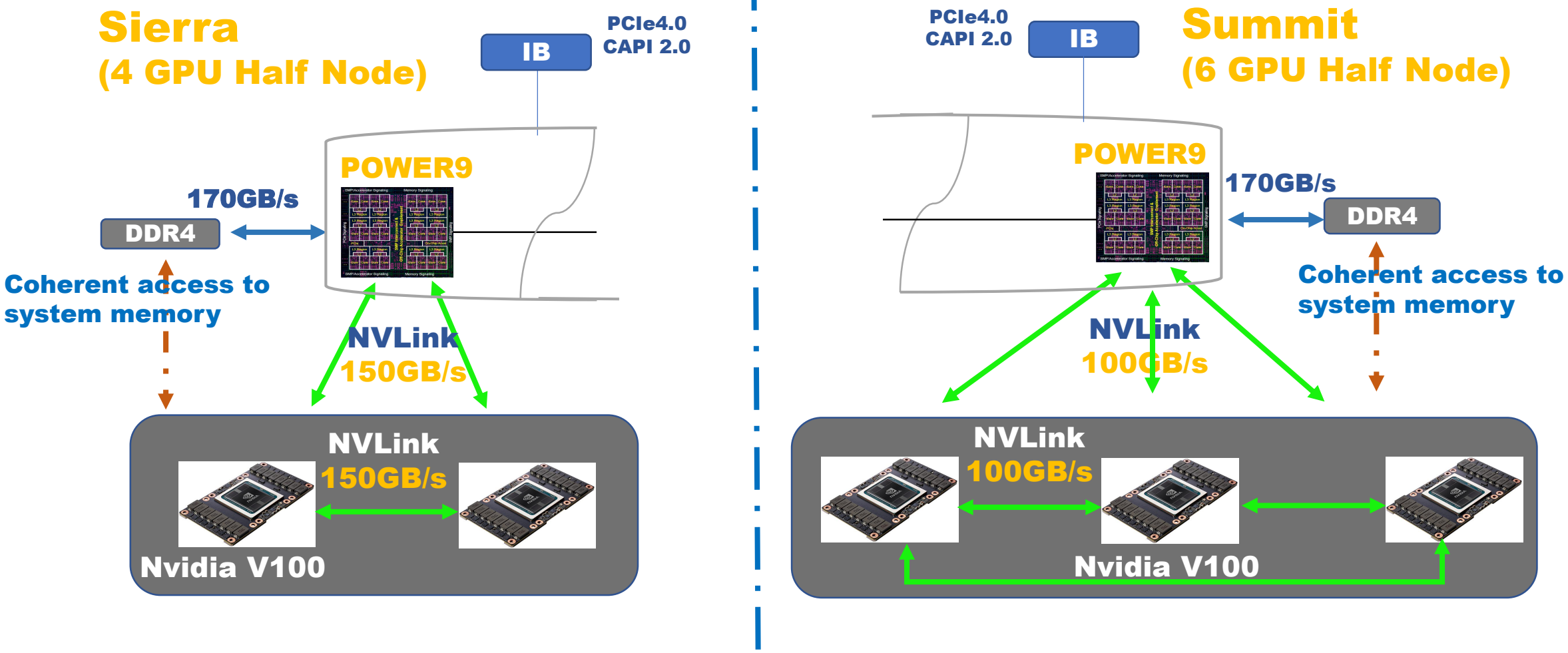
- Large and broad instruction set to perform complex operations

GPU

- High throughput – Massive parallelization through large number of cores
- Specialized for SIMD/SIMT

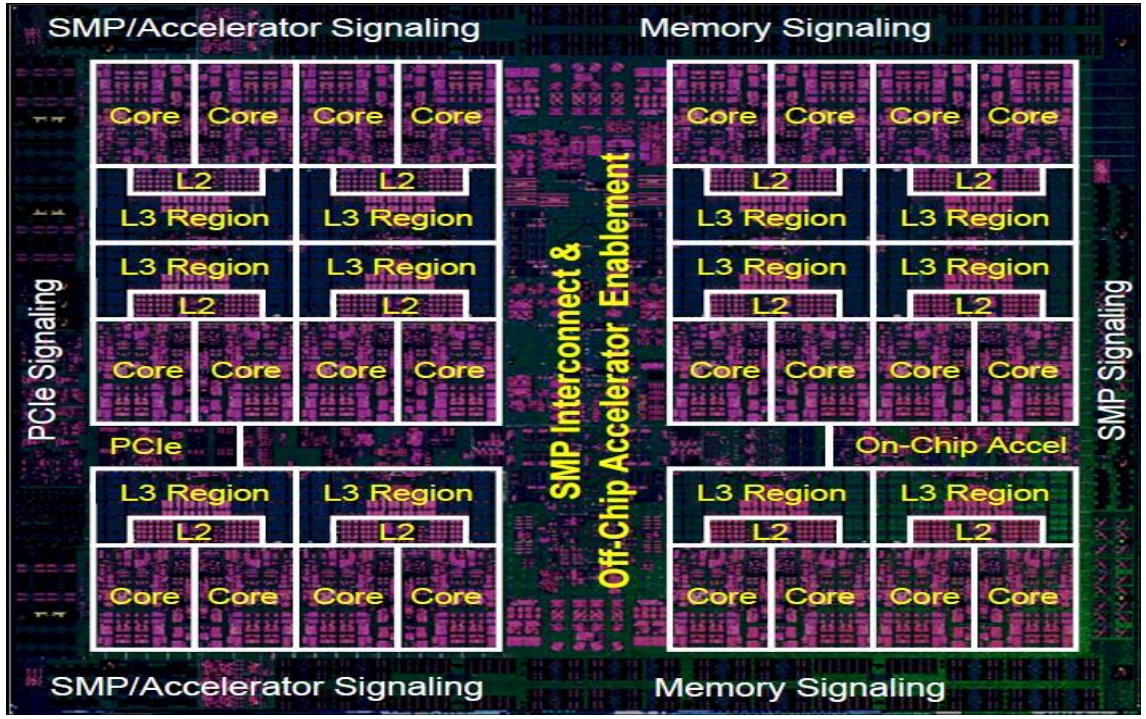


Summit and Sierra Supercomputer configurations



- CPU and GPU co-operate in execution of work
- GPU coherently access to CPU memory

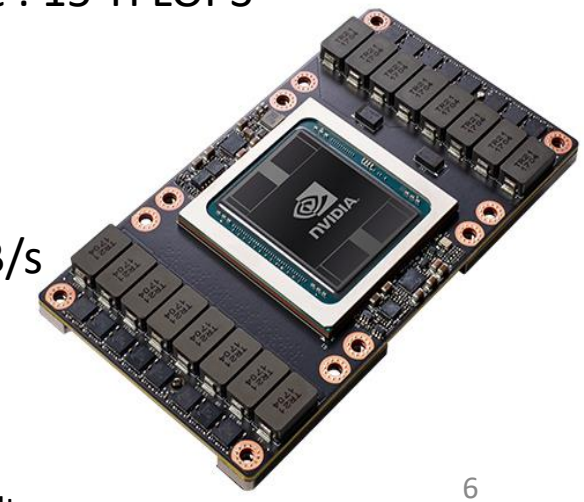
IBM Power9 Processor



- 14nm finFET semiconductor
- Stronger thread performance
- POWER ISA 3.0
- Enhanced Cache Hierarchy
- NVIDIA NVLink 2.0 with 25GB/s per link b/w
- I/O System – PCIe Gen4
- Improved device performance & reduced energy
- Nominal & Peak freq: 2.8GHz & 3.8GHz

Nvidia Tesla V100 GPU

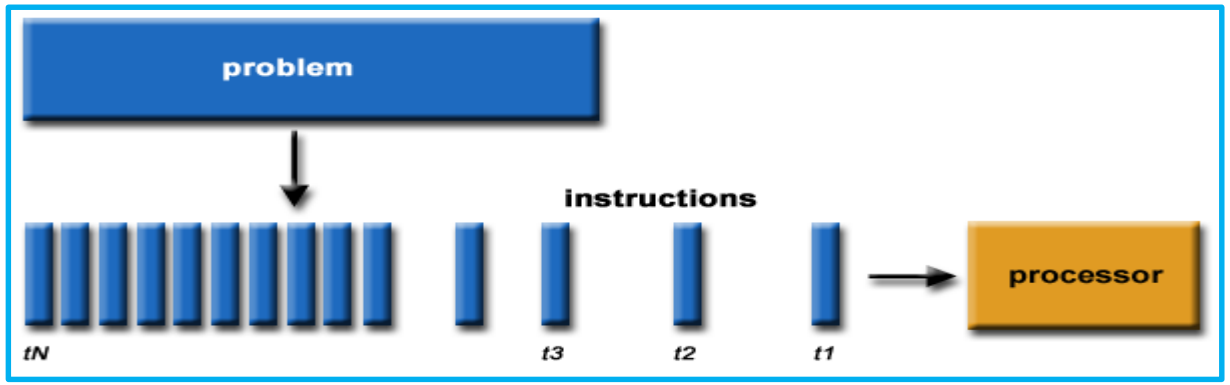
- Designed for AI Computing and HPC
- Second-Generation NVLink™
- HBM2 Memory: Faster, Higher Efficiency
- Enhanced Unified Memory and Address Translation Services
- Maximum Performance and Maximum Efficiency Modes
- Number of SM/cores : 80/5120
- Double Precision Performance : 7.5 TFLOPS
- Single Precision Performance : 15 TFLOPS
- 125 Tensor TFLOPS
- GPU Memory : 16 or 32 GB
- Memory bandwidth : 900 GB/s



https://devblogs.nvidia.com/inside-volta

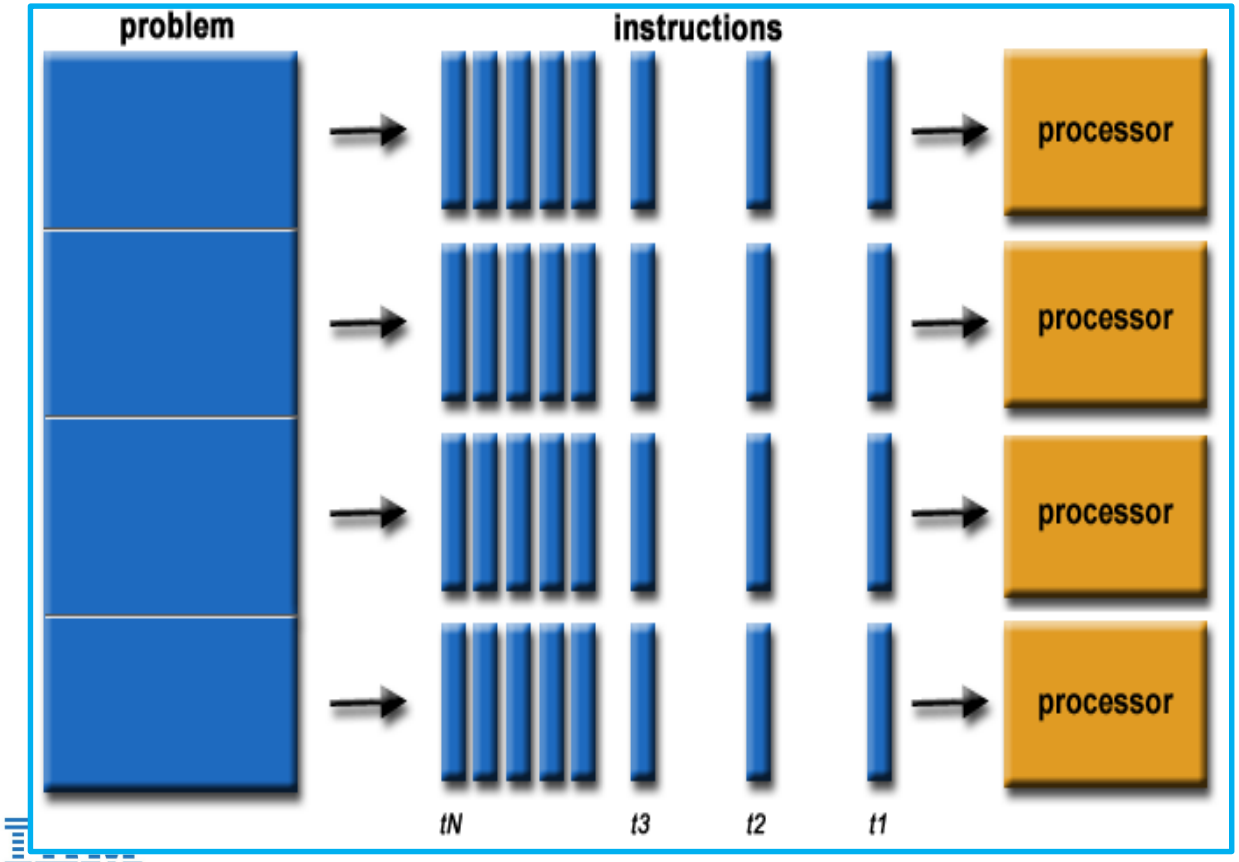
Introduction to Parallel computing

What is Parallel Computing?



Sequential Computing

- Sequential execution of series of instructions
- Only one computing resource



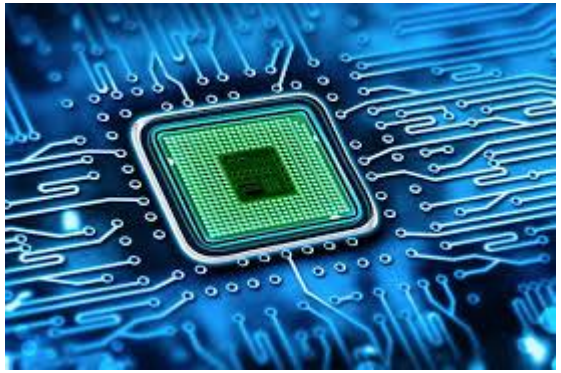
Parallel Computing

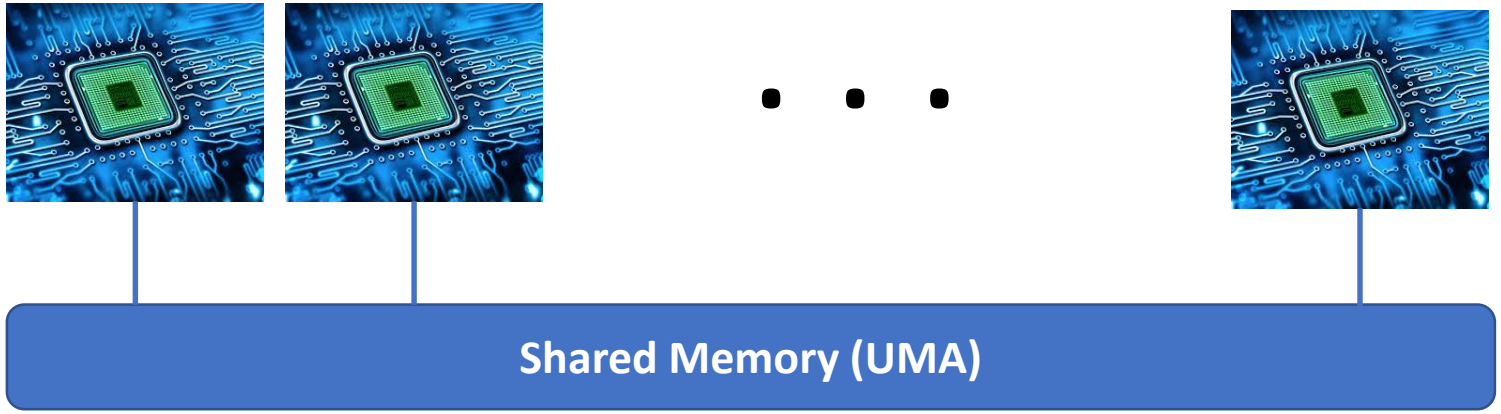
Simultaneous use of multiple compute resources to solve a computational problem

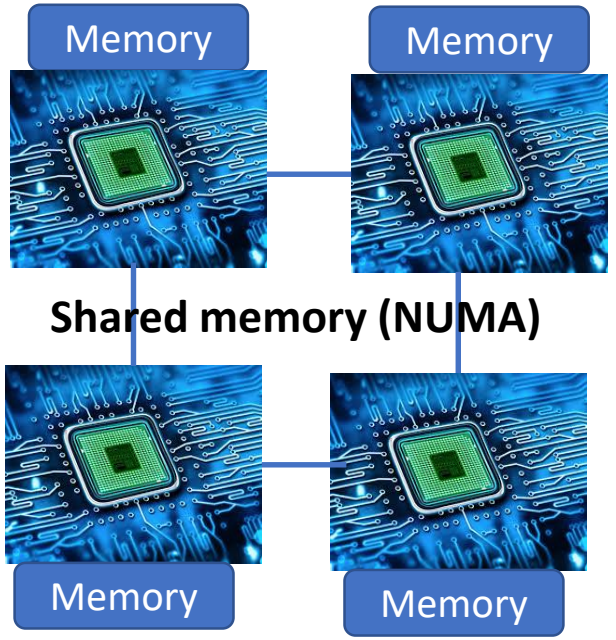
- Divided problem into discrete sub-problems
- Execute sub-problems in in simultaneously on different compute resource
- More than one compute resource

https://computing.llnl.gov/tutorials/parallel_comp/

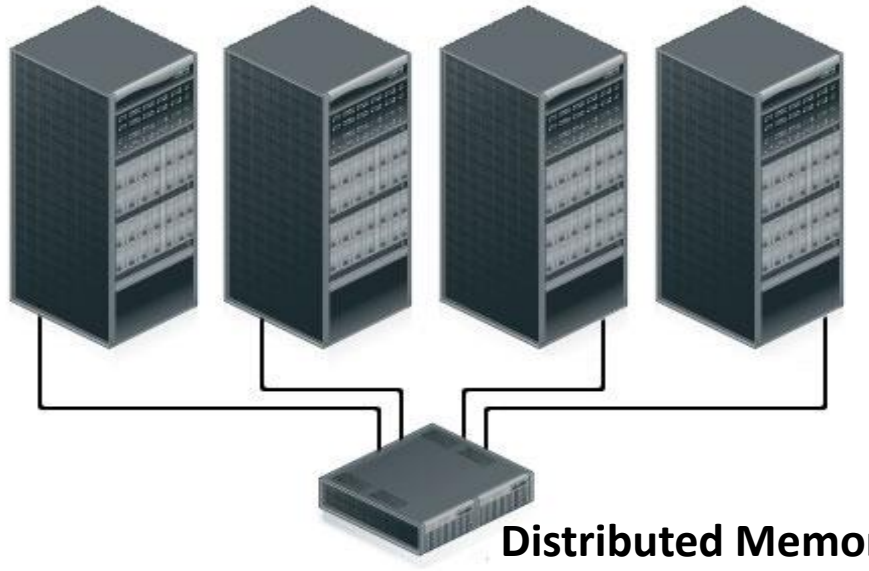
Memory Architecture







Cluster of nodes



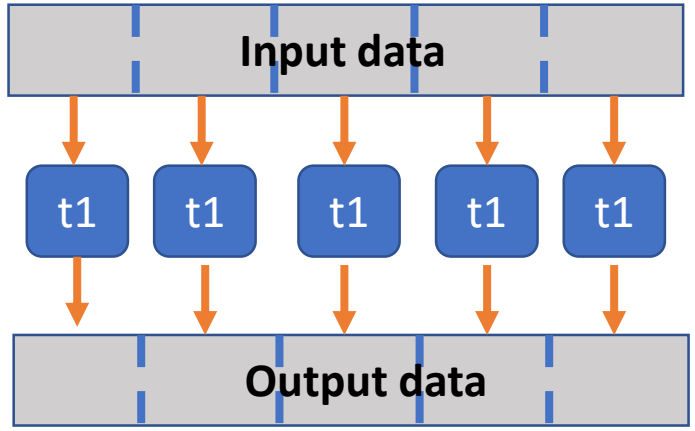
Parallel Computing

Memory Architecture

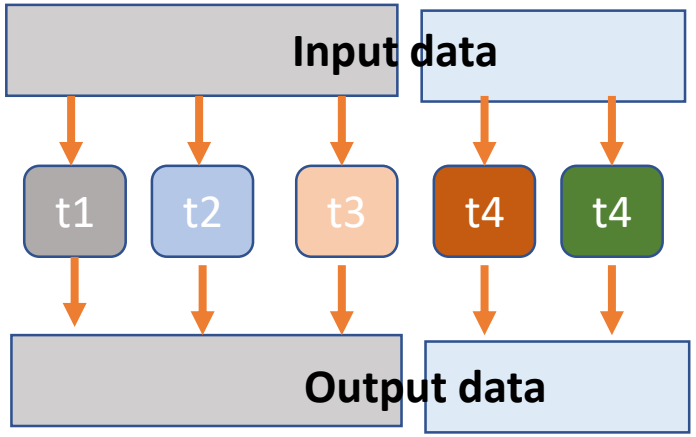
Shared memory
- Memory shared by cooperating processes (SMP)
- Requires Synchronization
- High throughput

Distributed Memory
- Message passing
- Messages passed between processes
- Highly scalable

Data Parallelism
- Partition Data and process
- Distribute data on different processing elements
- SIMD/SPMD
- Scale throughput of processing



Task Parallelism
- Each compute element perform different task
- Co-ordinate to solve a specific problem



Parallel Computing

Scaling

Weak Scaling
- Fixed problem size per processor
- Solve larger problem with same time

Strong Scaling
- Fixed problem size
- Time reduction with more compute processors
- Highly scalable

Proprietary and Open Source Compiler Offerings supporting Acceleration Enabled Programing Models

CUDA

Key Features:

- Gives direct access to the GPU instruction set
- Supports C, C++ and Fortran
- Generally achieves best leverage of GPUs for best application performance

OpenCL

MPI – Message passing

OpenMP

Key Features:

- OpenMP 4.0 introduces offloading and support for heterogeneous CPU/GPU
- Leverage existing OpenMP high level directives support

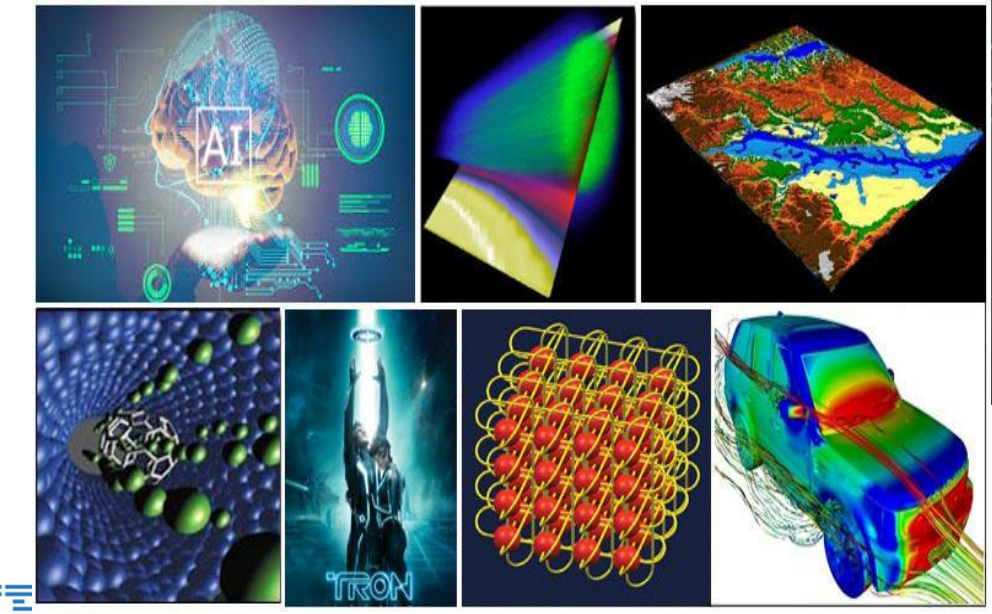
OpenACC
Directives for Accelerators

Key Features:

- Designed to simplify Programing of heterogeneous CPU/GPU systems
- Directive based parallelization for accelerator device

Applications of Parallel Computing

Science &
Engineering
Artificial Intelligence



Analytics
Realtime simulations
Modelling

OpenMP

Monitoring and Profiling tools

Monitoring and Profiling tools

Monitoring

- mpstat, vmstat – CPU and memory utilization
- numastat – numa memory statistics
- top/htop – real-time view of system usage

Profiling

- Perf record/report – CPU profiling
- nvprof – GPU profiling

```
[aditya@hpc numastat -m | grep "Node\|MemUsed"
                          Node 0          Node 8        Node 252        Node 253        Node 254        Node 255           Total
MemUsed                104872.31        57992.06           11.19           11.19           11.19         1455.19       164353.12
```

CPU memory

GPU memory

Monitoring and Profiling tools

nvidia-smi

```
[aditya@hpcnw4 ompeval]$ nvidia-smi
Sun Apr 21 03:03:12 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.64                 Driver Version: 396.64                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off |                    0 |
| N/A   42C    P0   153W / 300W |   1539MiB / 15360MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000004:05:00.0 Off |                    0 |
| N/A   38C    P0    37W / 300W |     11MiB / 15360MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000035:03:00.0 Off |                    0 |
| N/A   35C    P0    36W / 300W |     11MiB / 15360MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000035:04:00.0 Off |                    0 |
| N/A   41C    P0    38W / 300W |     11MiB / 15360MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     54063      C   ./matmul_gpuoffload_cl                      1528MiB |
+-----------------------------------------------------------------------------+
```

Also check “nvidia-smi –query-gpu” more monitoring options

Monitoring and Profiling tools

nvprof

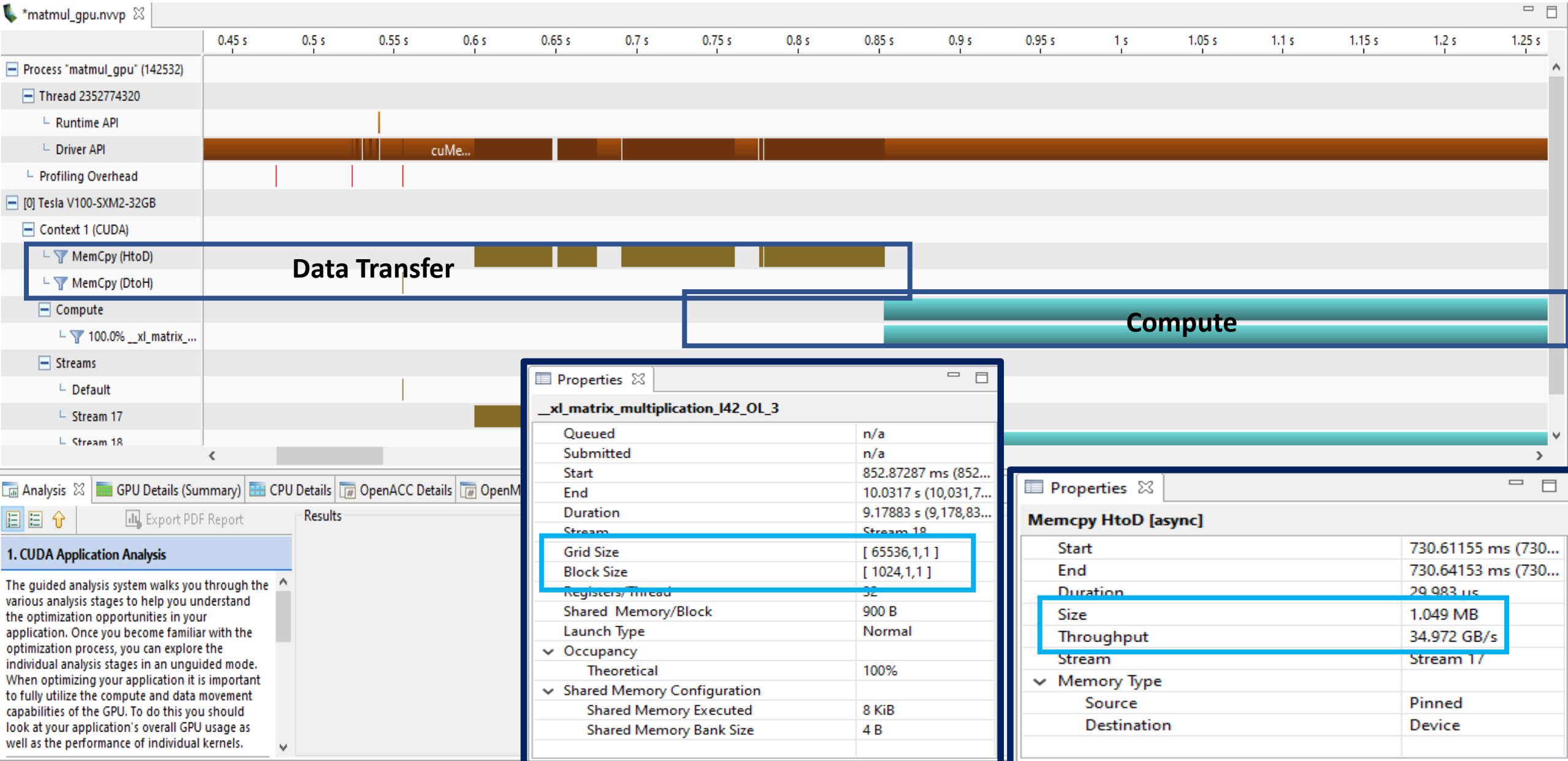
- The *nvprof* is command-line profiling tool which enables you to collect and view profiling data
- Using *nvprof* one can collect –
  - kernel execution time
  - memory transfers
  - memory set and CUDA API calls
  - events or metrics for CUDA kernels

NVVP (NVIDIA Visual Profiler)

- The Visual Profiler displays a timeline of your application's activity on both the CPU and GPU so that one can identify opportunities for performance improvement.
- Visualize profile data collected from *nvprof*

- More documentation can be found @ https://docs.nvidia.com/cuda/profiler-users-guide/index.html

Nvidia Visual Profiler

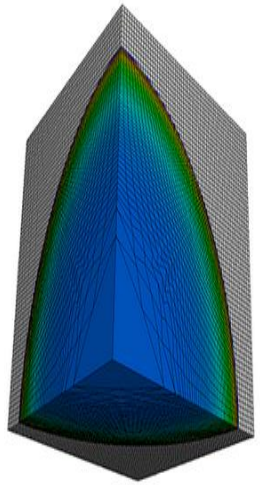


HPC Application performance with OpenMP GPU offloading

LULESH : OpenMP 4.5 GPU Offload

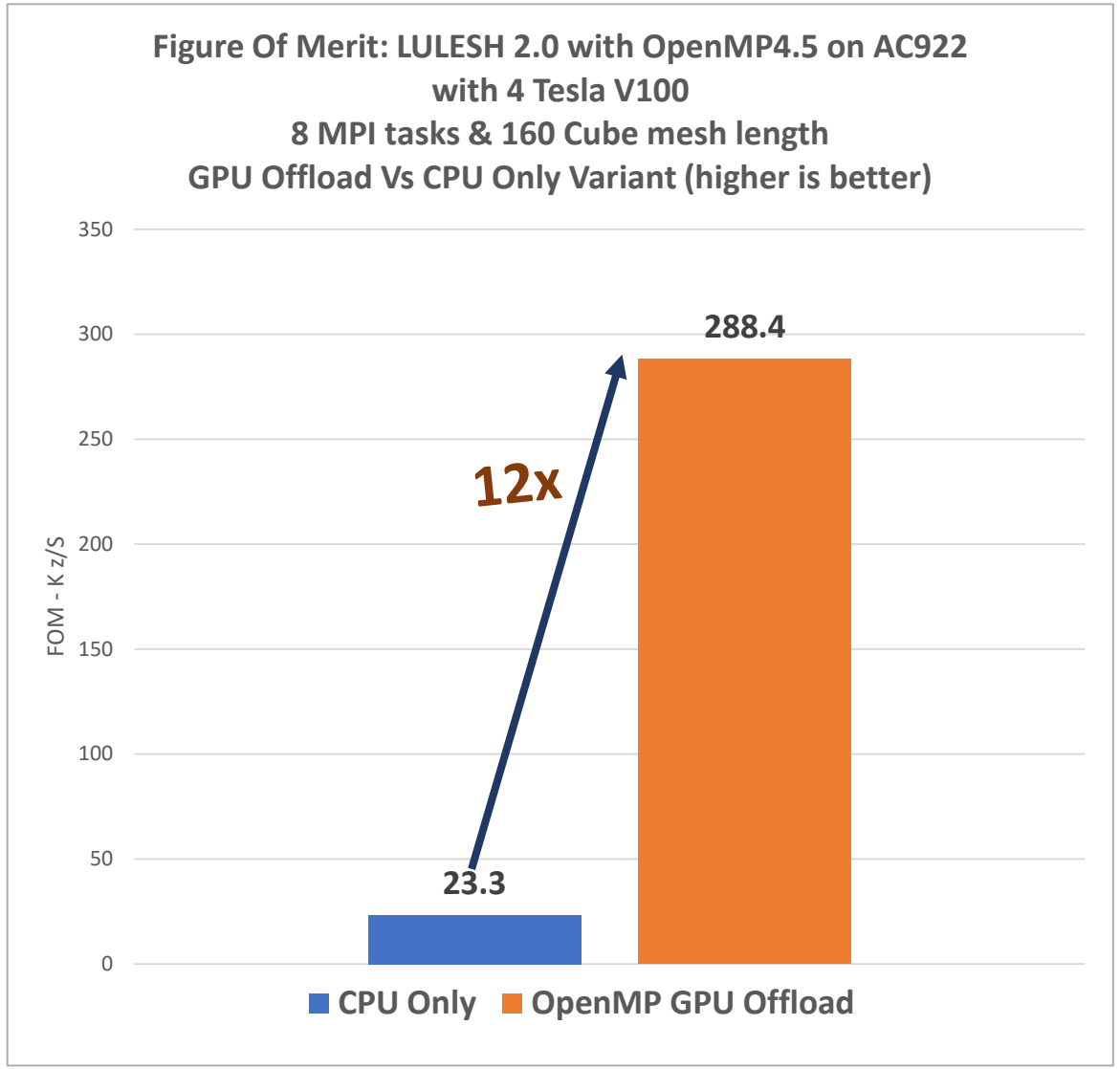
OpenMP 4.5 GPU Offload for LULESH (Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics) application can run:

- 12x faster compared to CPU only variant Implementation in reaching FOM (Figure Of Merit)

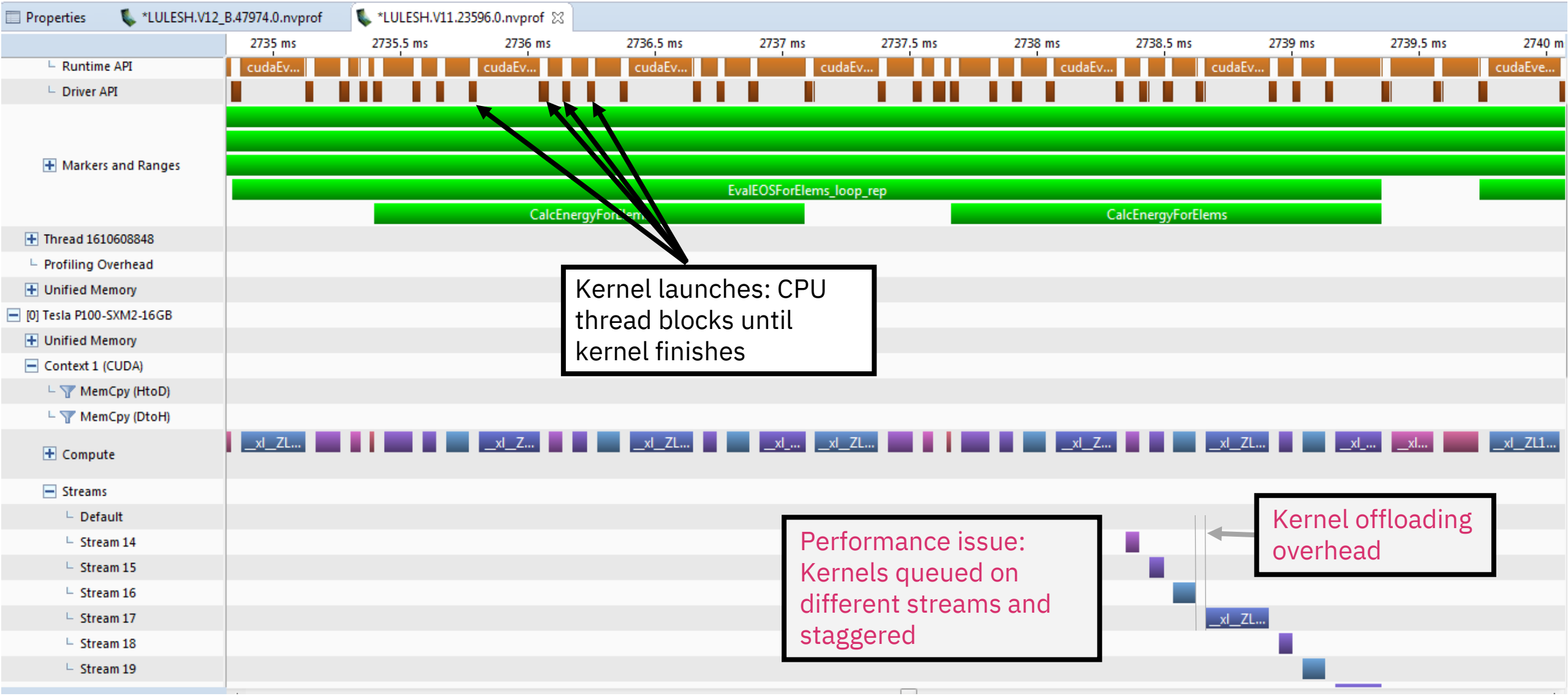


LULESH

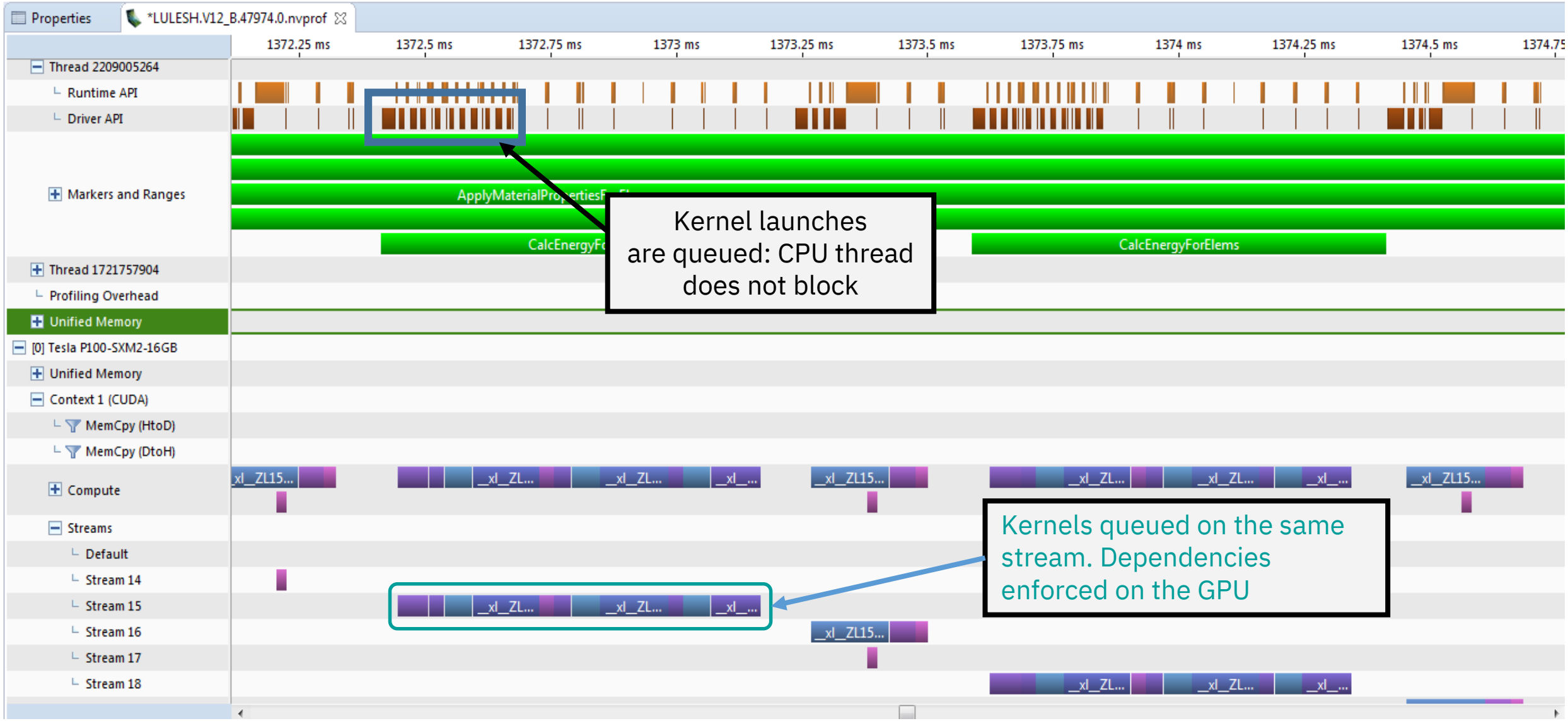
A shock hydrodynamics mini-app for computer simulations leveraging high performance computing of a wide variety of science and engineering problems that describes the motion of materials relative to each other when subject to forces.

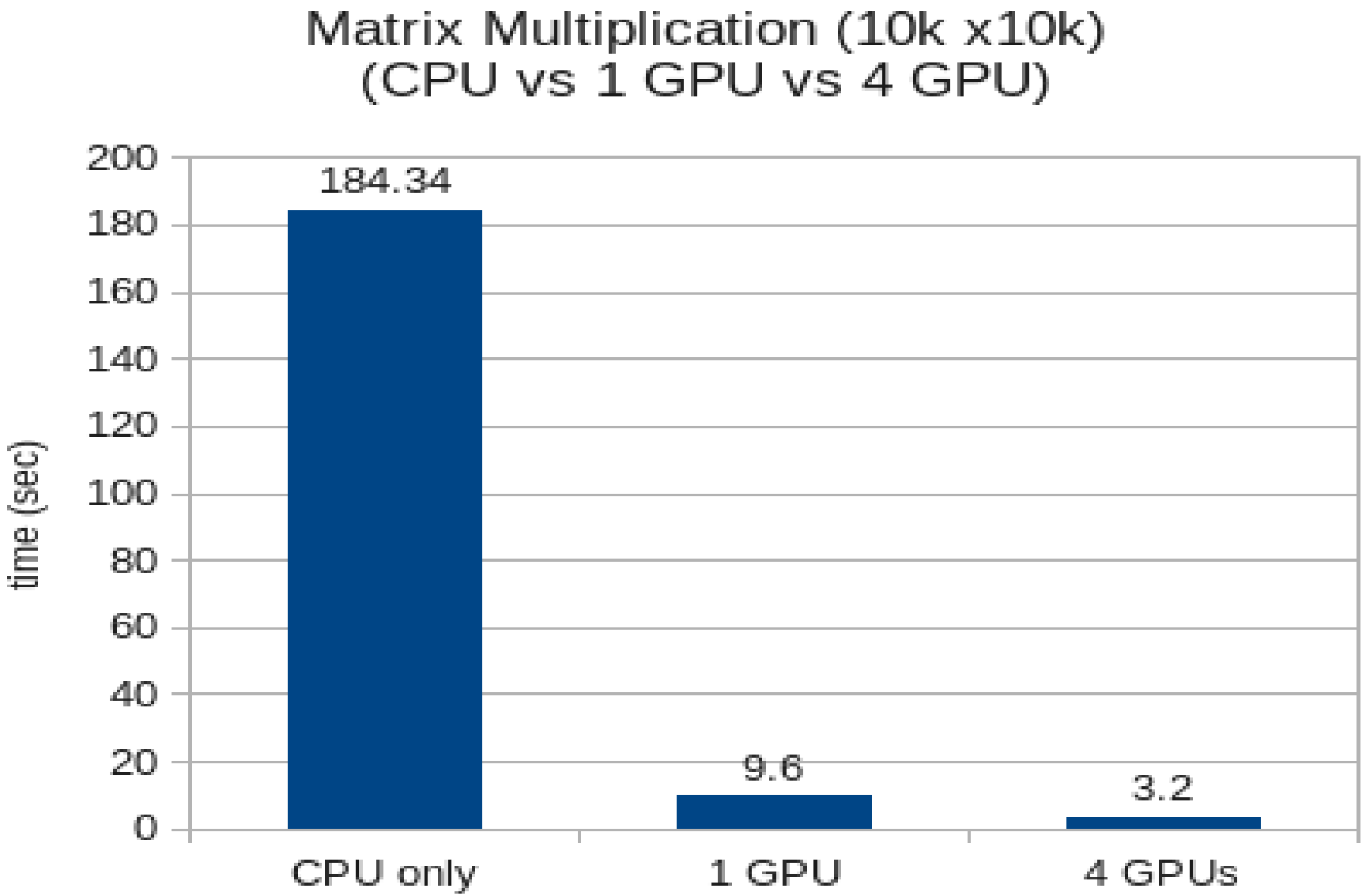


LULESH Profile: CPU-GPU Synchronous Execution



LULESH Profile – Synchronous Version




Matrix Multiplication (10k x10k)
(CPU vs 1 GPU vs 4 GPU)
time (sec)
200
180
160
140
120
100
80
60
40
20
0
184.34
9.6
3.2
CPU only
1 GPU
4 GPUs

References

- GPU programming made easy with OpenMP on IBM POWER (https://developer.ibm.com/articles/gpu-programming-with-openmp)
- Offloading computations to the NVIDIA GPUs (https://www.ibm.com/support/knowledgecenter/en/SSXVZZ_16.1.0/com.ibm.xlcpp161.lelinux.doc/proguide/offloading.html)
- Parallel Computing(https://computing.llnl.gov/tutorials/parallel_comp)
- Hands on with OpenMP4.5 and Unified Memory:
  - Part I(https://link.springer.com/chapter/10.1007%2F978-3-319-65578-9_1)
  - Part II (https://link.springer.com/chapter/10.1007%2F978-3-319-65578-9_2)

Thank You