Performance of 3-D FFT Using Multiple GPUs with CUDA 4

Akira Nukada

Tokyo Institute of Technology

GPGPU

- High performance, high memory bandwidth
- High efficiencies in power, space, price

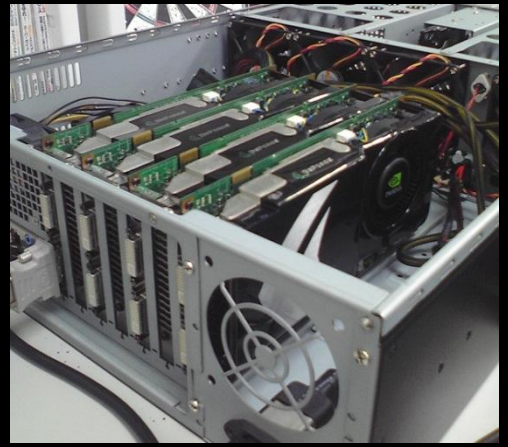
- Complicated programming
- Capacity of device memory
- Requires parallelism

- Need optimizations in algorithm, data placement, ….

Next stage of GPGPU

We need multiple GPUs for

- More computing power
- More device memory



There is no SLI for GPGPU, so we have to manage data distribution, load-balancing, and synchronizations between GPUs.

Multi-GPU programming

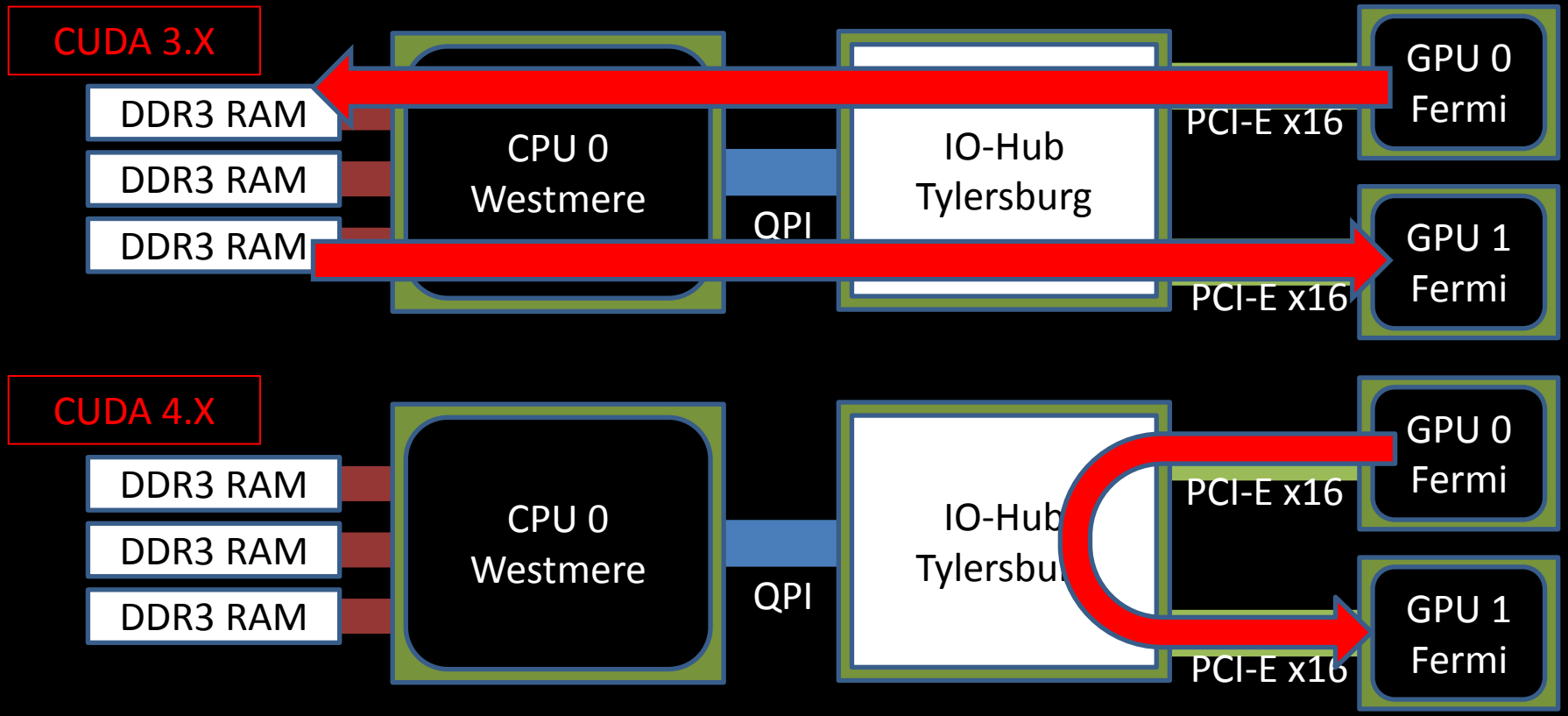
- Similar to MPI applications (distributed memory)
  - Easy for developers with MPI experience
  - Not easy to achieve expected performance

CUDA 4.0

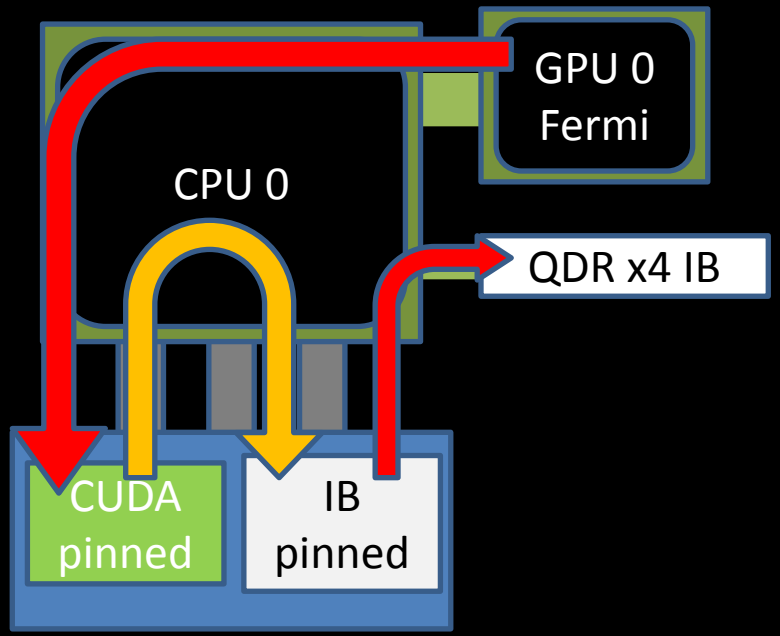
- New features for multi-GPU applications
  - Direct P2P via PCI-E network
  - NIC interoperability with InfiniBand HCAs

These improve the data transfer speed between GPUs

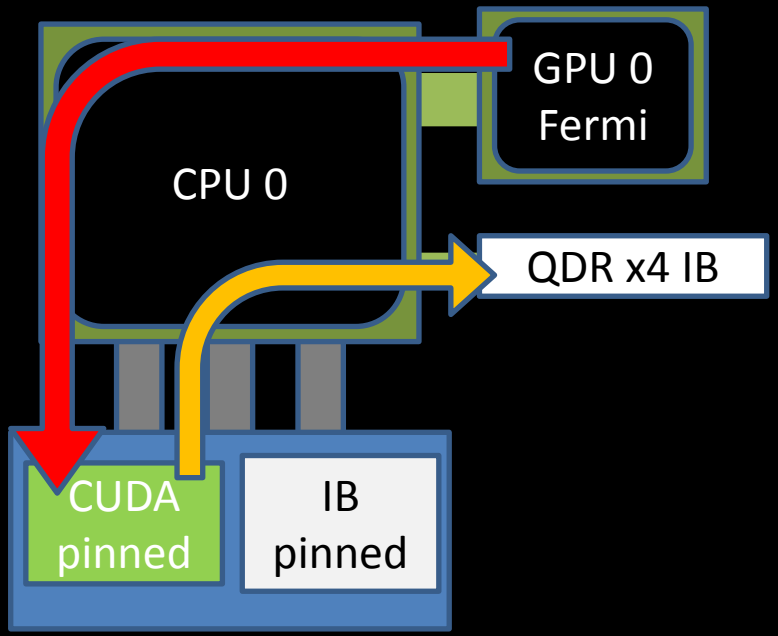
Intra-node


CUDA 3.X
DDR3 RAM
DDR3 RAM
DDR3 RAM
CPU 0
Westmere
QPI
IO-Hub
Tylersburg
PCI-E x16
PCI-E x16
GPU 0
Fermi
GPU 1
Fermi
CUDA 4.X
DDR3 RAM
DDR3 RAM
DDR3 RAM
CPU 0
Westmere
QPI
IO-Hub
Tylersbu
PCI-E x16
PCI-E x16
GPU 0
Fermi
GPU 1
Fermi

Inter-node : CUDA 3.X

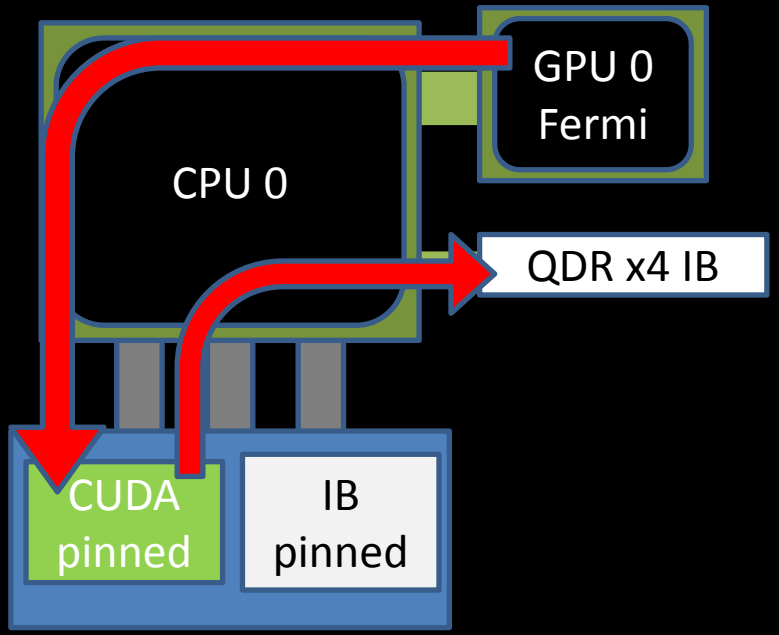


Memory copy on host



Disable DMA access by IB

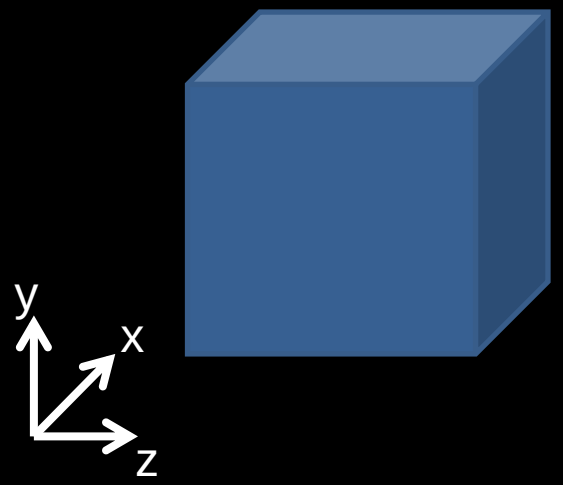
Inter-node : CUDA 4.X



Allows DMA access by IB

3-D FFT

Performs 1-D FFT for each dimension of 3-D data.



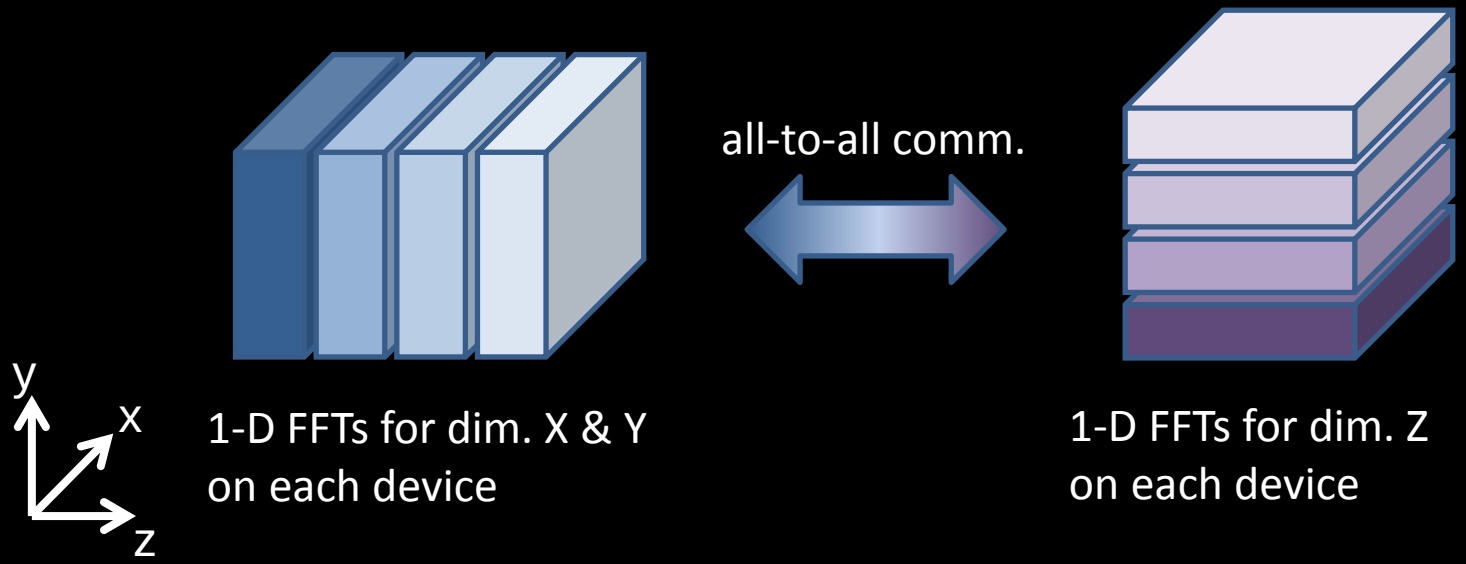
Memory-intensive computation

- O(N log N) ops for O(N) data

Several FFT implementations for CUDA

- NVIDIA CUFFT library
- Our NukadaFFT library
- others…

3-D FFT on distributed memory



All-to-all comm. in FFT

Parallel FFT typically requires 1~3 all-to-all comm.

The number depends on the data distribution before and after the transform.

$$(NX, NY, NZ/P) \Leftrightarrow (NX, NY/P, NZ)$$

before after

In such a case, only 1 all-to-all comm. is required.

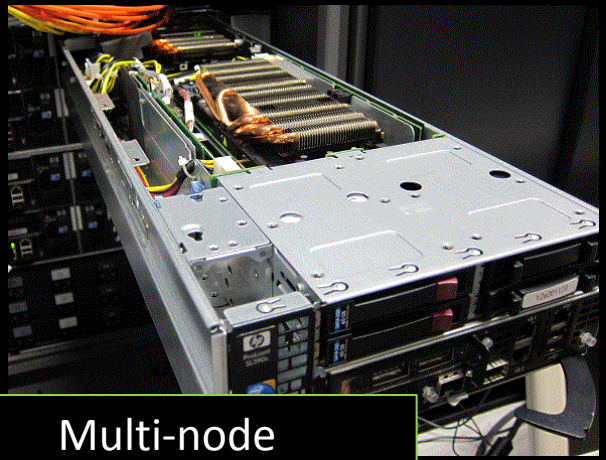
Many applications allow this:

- 3D-RISM, Molecular simulation. (Motivation of this work)
- Convolution based computations.

TSUBAME 2.0 Compute Node

Thin node (HP ProLiant SL390s G7)

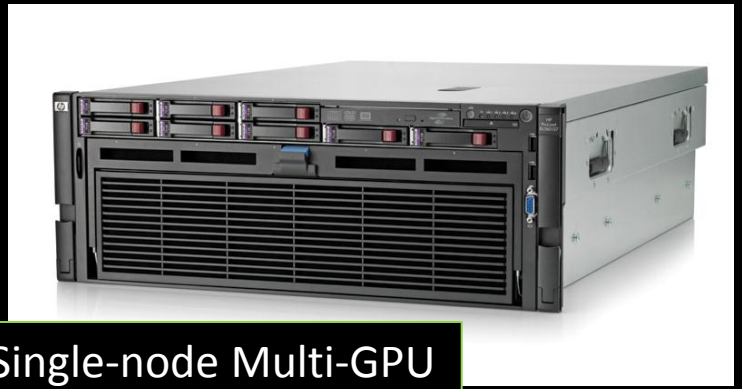
- Intel Xeon X5670 (Westmere-EP) x2
- NVIDIA Tesla M2050 x3
- Mellanox QDR x4 InfiniBand HCA x2



Multi-node

Medium node (HP ProLiant DL580 G7)

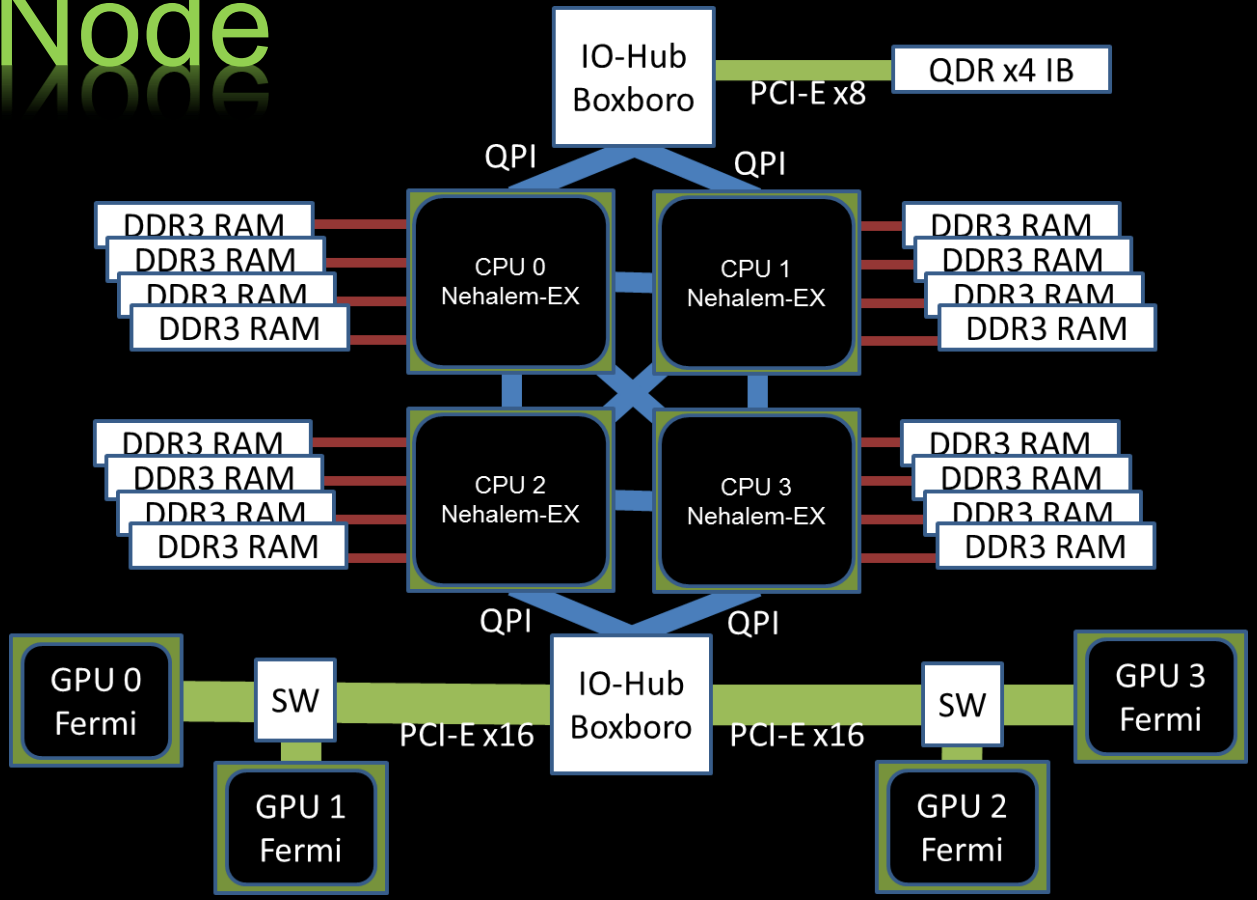
- Intel Xeon X7550 (Nehalem-EX) x4
- NVIDIA Tesla M2070 x4
  (NextIO vCORE Express 2070)
- Mellanox QDR x4 InfiniBand HCA x1



Single-node Multi-GPU

SINGLE NODE MULTI GPUS….

Medium Node


IO-Hub
Boxboro
PCI-E x8
QDR x4 IB
QPI
QPI
DDR3 RAM
DDR3 RAM
DDR3 RAM
DDR3 RAM
CPU 0
Nehalem-EX
CPU 1
Nehalem-EX
DDR3 RAM
DDR3 RAM
DDR3 RAM
DDR3 RAM
DDR3 RAM
DDR3 RAM
DDR3 RAM
DDR3 RAM
CPU 2
Nehalem-EX
CPU 3
Nehalem-EX
DDR3 RAM
DDR3 RAM
DDR3 RAM
DDR3 RAM
QPI
QPI
GPU 0
Fermi
SW
PCI-E x16
IO-Hub
Boxboro
PCI-E x16
SW
GPU 3
Fermi
GPU 1
Fermi
GPU 2
Fermi

Four scheduling

via host memory

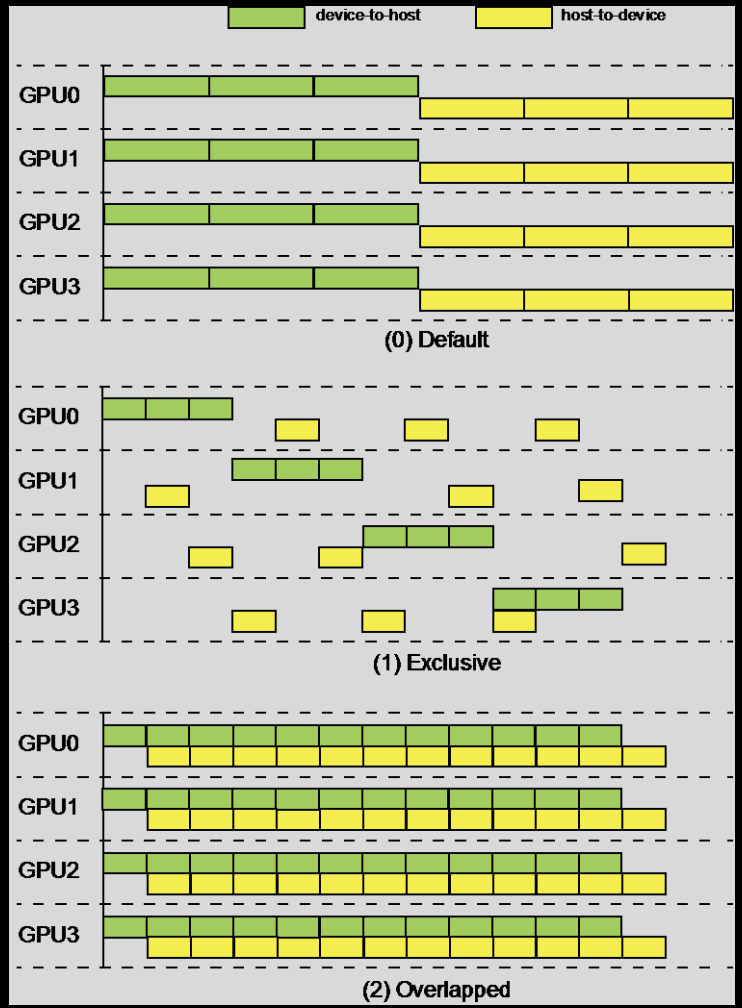
1. Default
2. Exclusive
3. Overlapped

Direct P2P

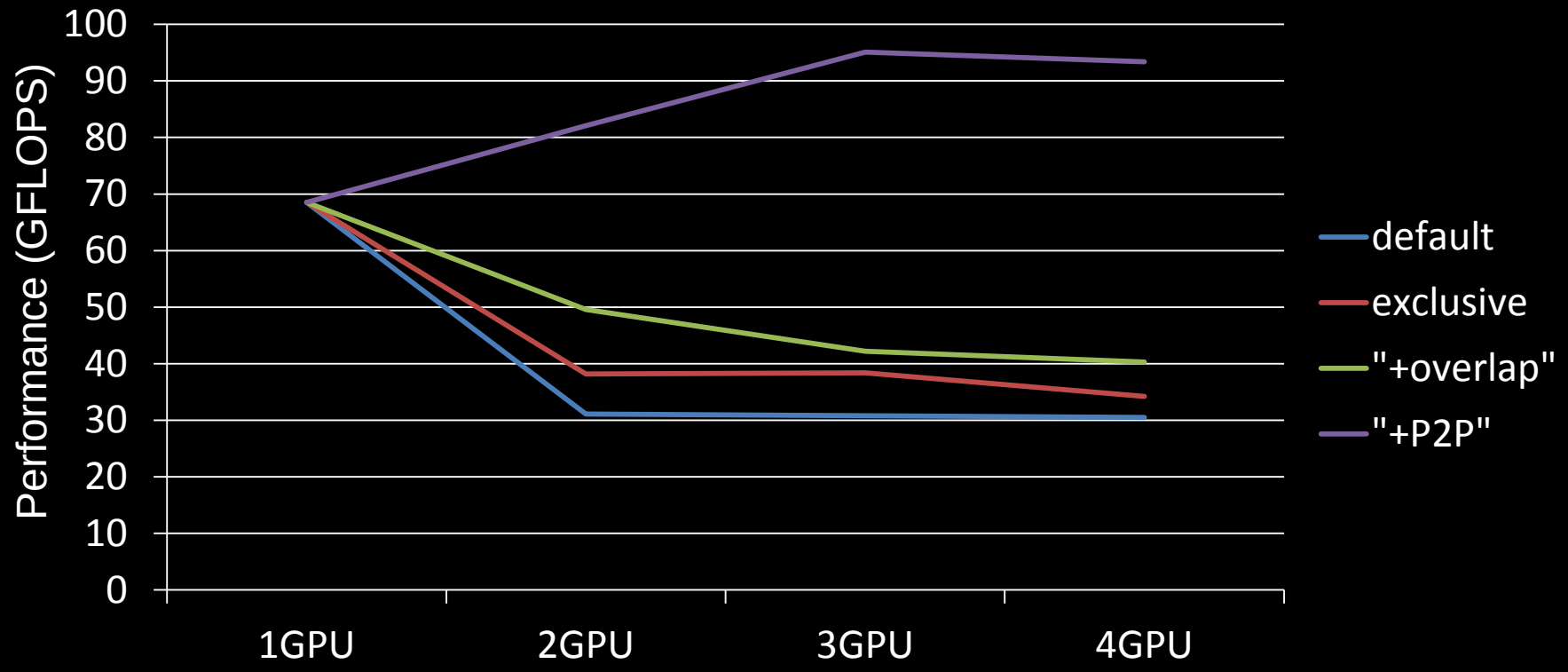
4. P2P

PCI-E & QPI are bi-directional ;

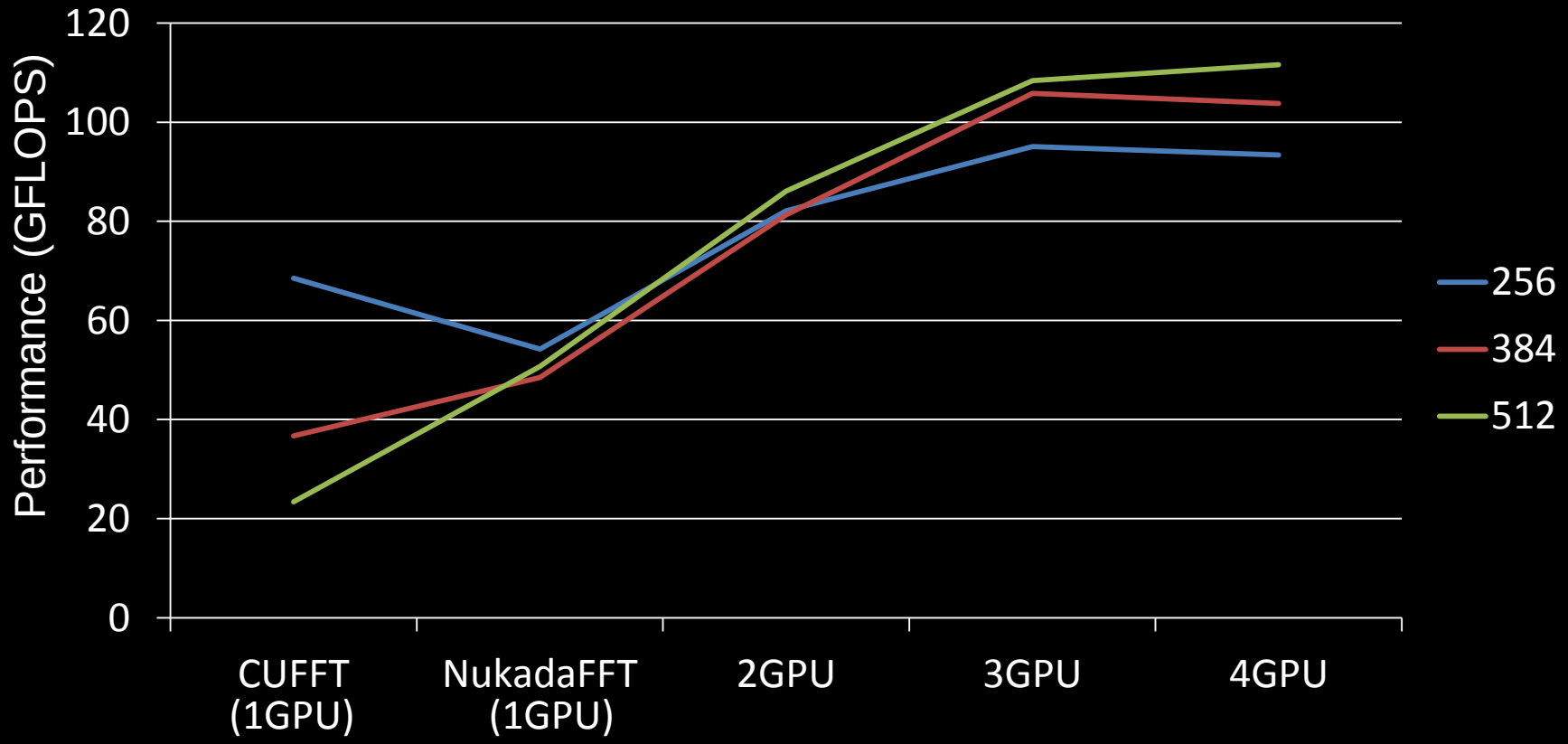
Important to fill both direction.



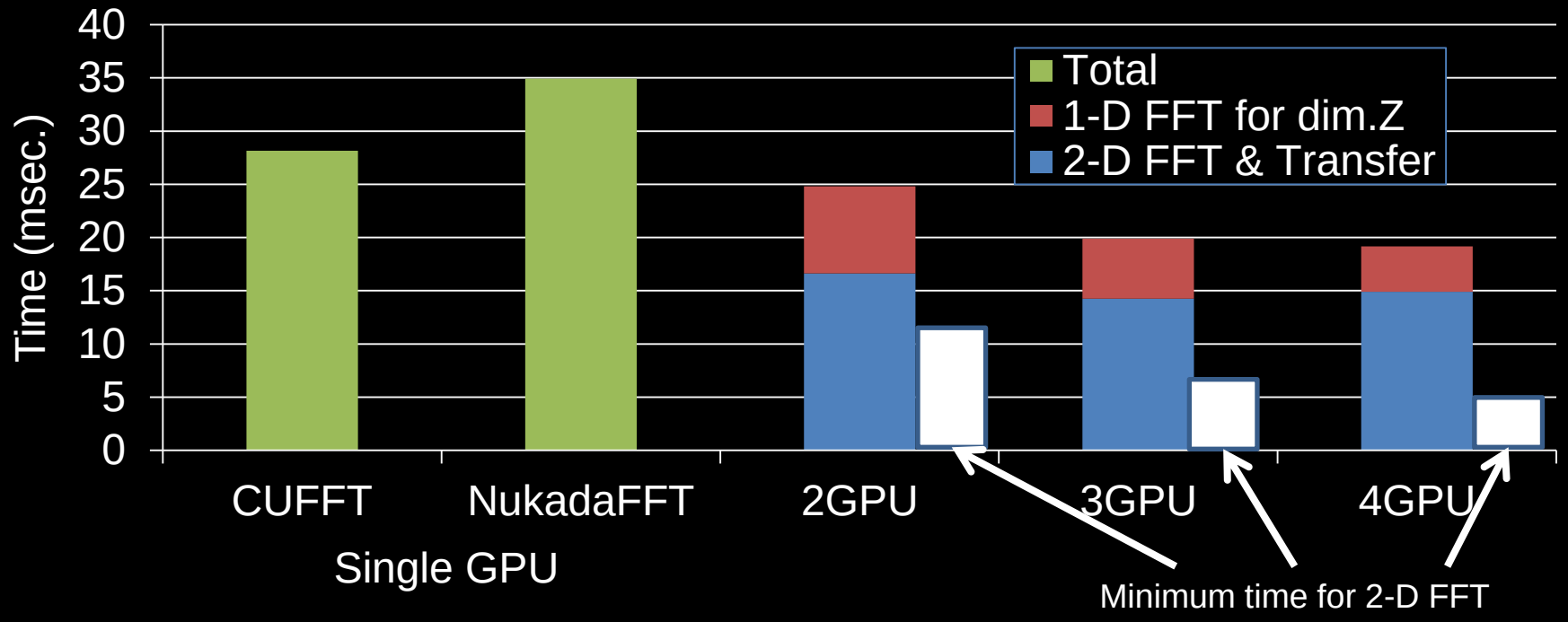
Performance of 3-D FFT using different scheduling policies (256x256x256, DP)



Performance of 3-D FFT using P2P

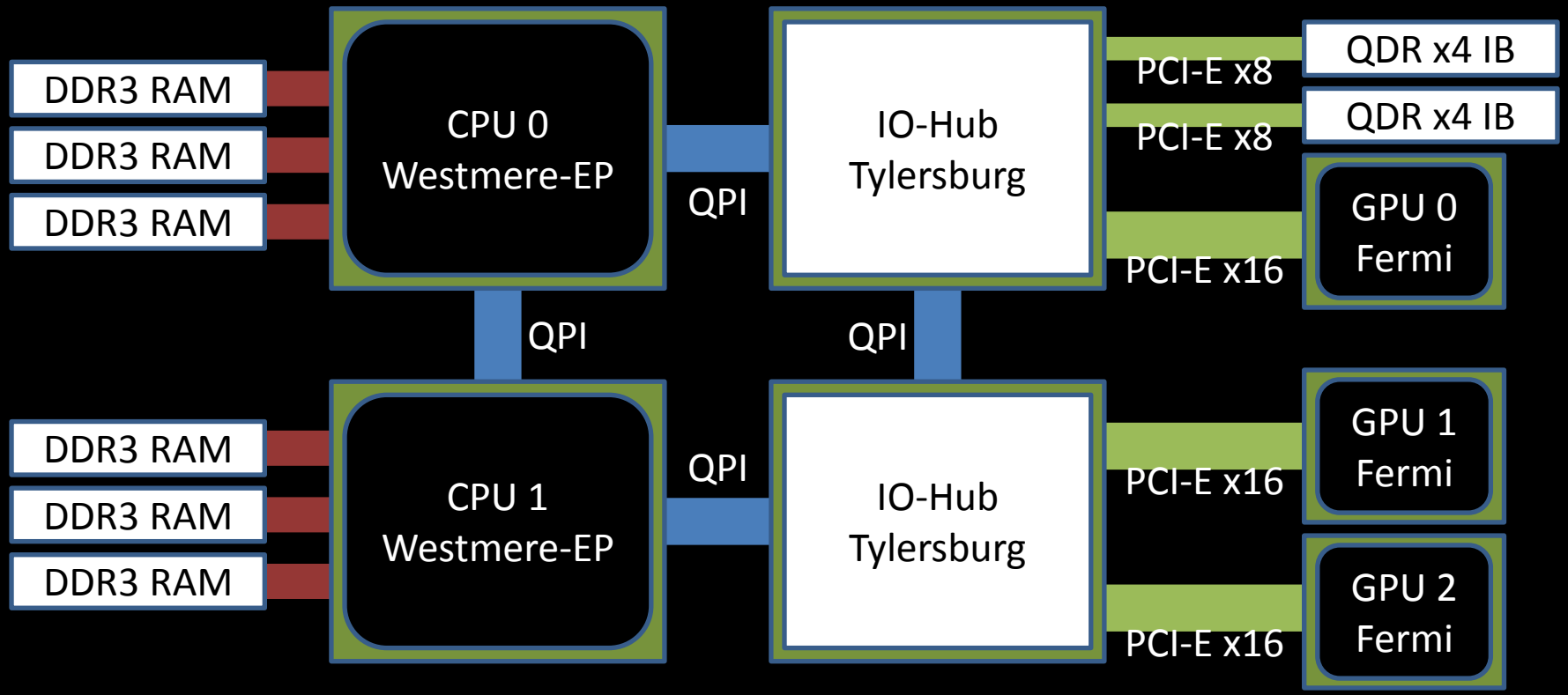


Breakdown of 256^3 3-D FFT



MULTIPLE NODES….

Thin Node


DDR3 RAM
DDR3 RAM
DDR3 RAM
CPU 0
Westmere-EP
QPI
IO-Hub
Tylersburg
PCI-E x8
QDR x4 IB
PCI-E x8
QDR x4 IB
GPU 0
Fermi
PCI-E x16
QPI
QPI
DDR3 RAM
DDR3 RAM
DDR3 RAM
CPU 1
Westmere-EP
QPI
IO-Hub
Tylersburg
PCI-E x16
GPU 1
Fermi
GPU 2
Fermi
PCI-E x16

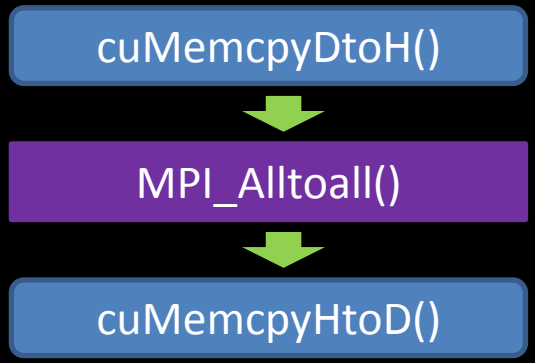
Comm. between GPUs on multi-node

Three steps of DMA transfers

(1) Device-to-host by CUDA API

(2) Between node by MPI

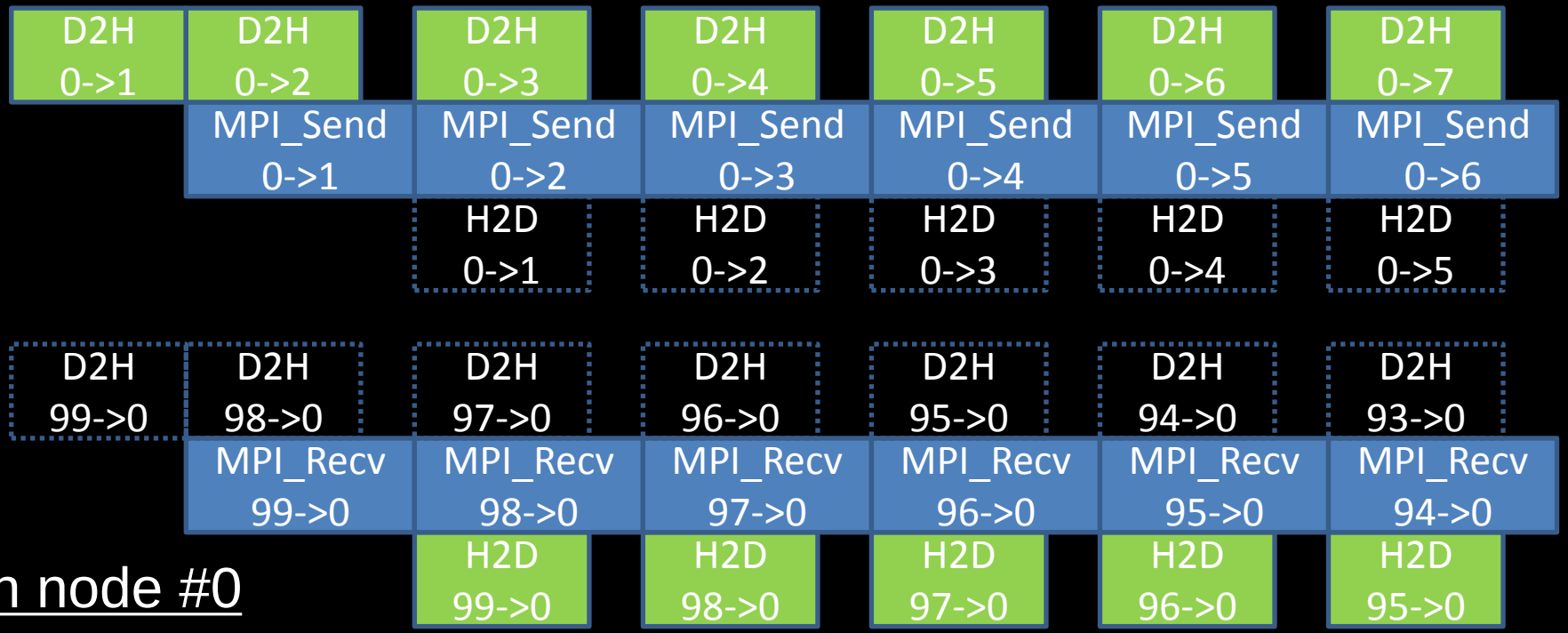
(3) Host-to-device by CUDA API



All-to-all between P nodes performs (P – 1) Comm. between GPUs.

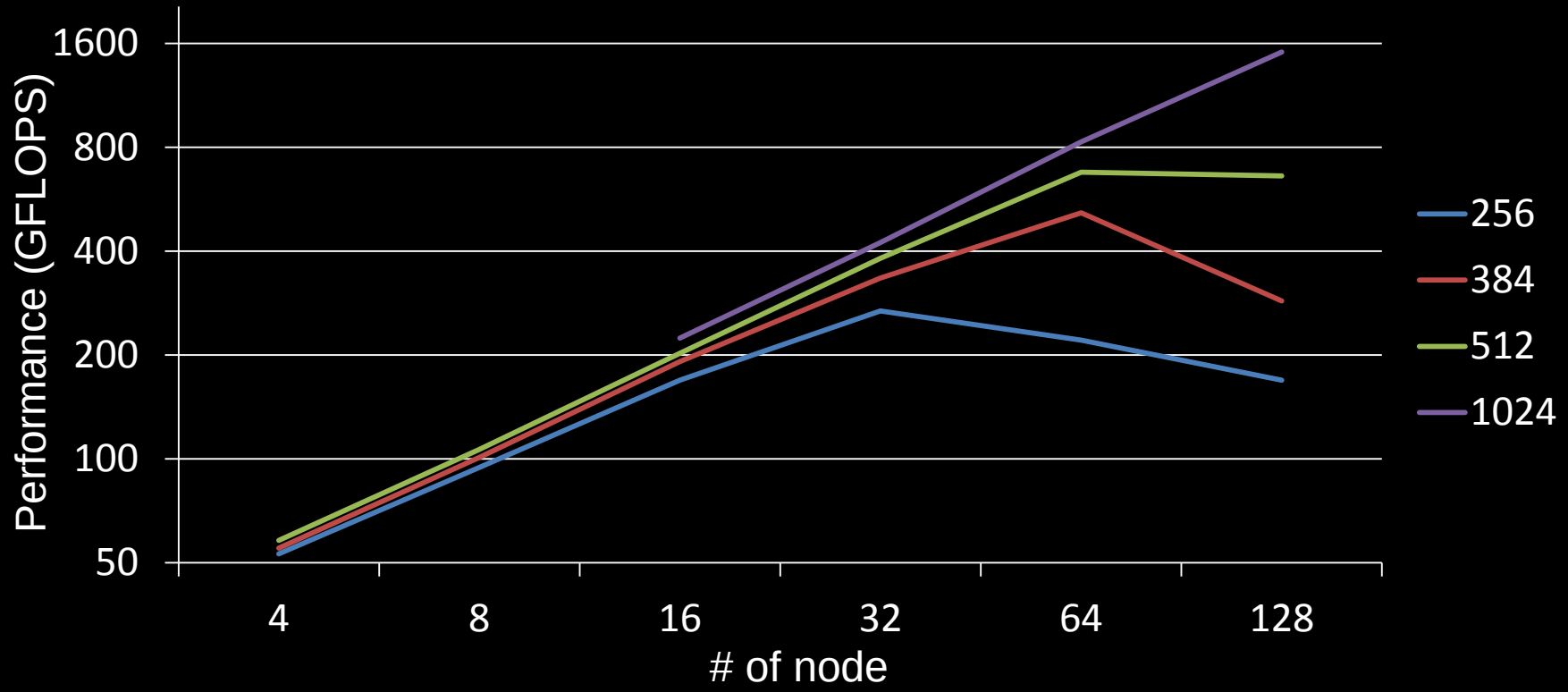
Those steps should be pipelined, because all DMA controllers can work simultaneously.

MPI-CUDA all-to-all comm.



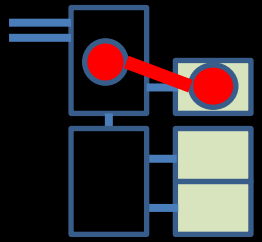
On node #0

3-D FFT using MPI (1GPU/node)

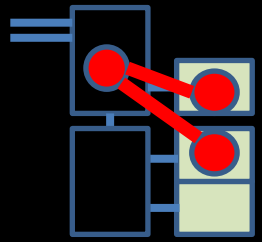


Multi-GPU per node, multiple nodes.

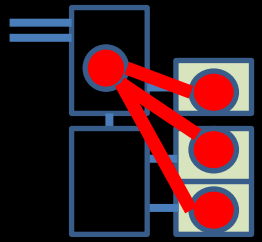
(1) Hybrid : multiple GPU per process



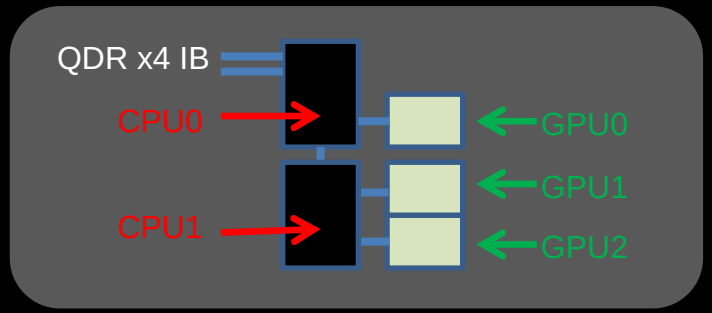
1GPU (0)



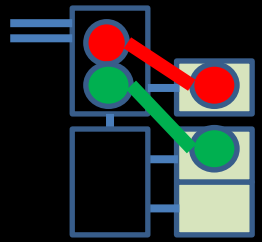
2GPU (0)



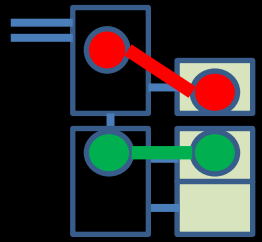
3GPU (0)



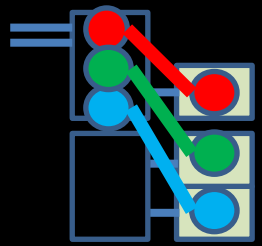
(2) Flat: one GPU per process



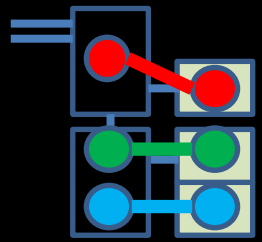
2GPU (0-0)



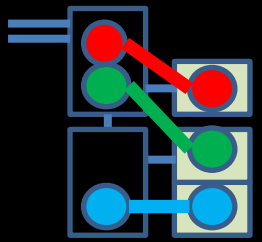
2GPU (0-1)



3GPU (0-0-0)

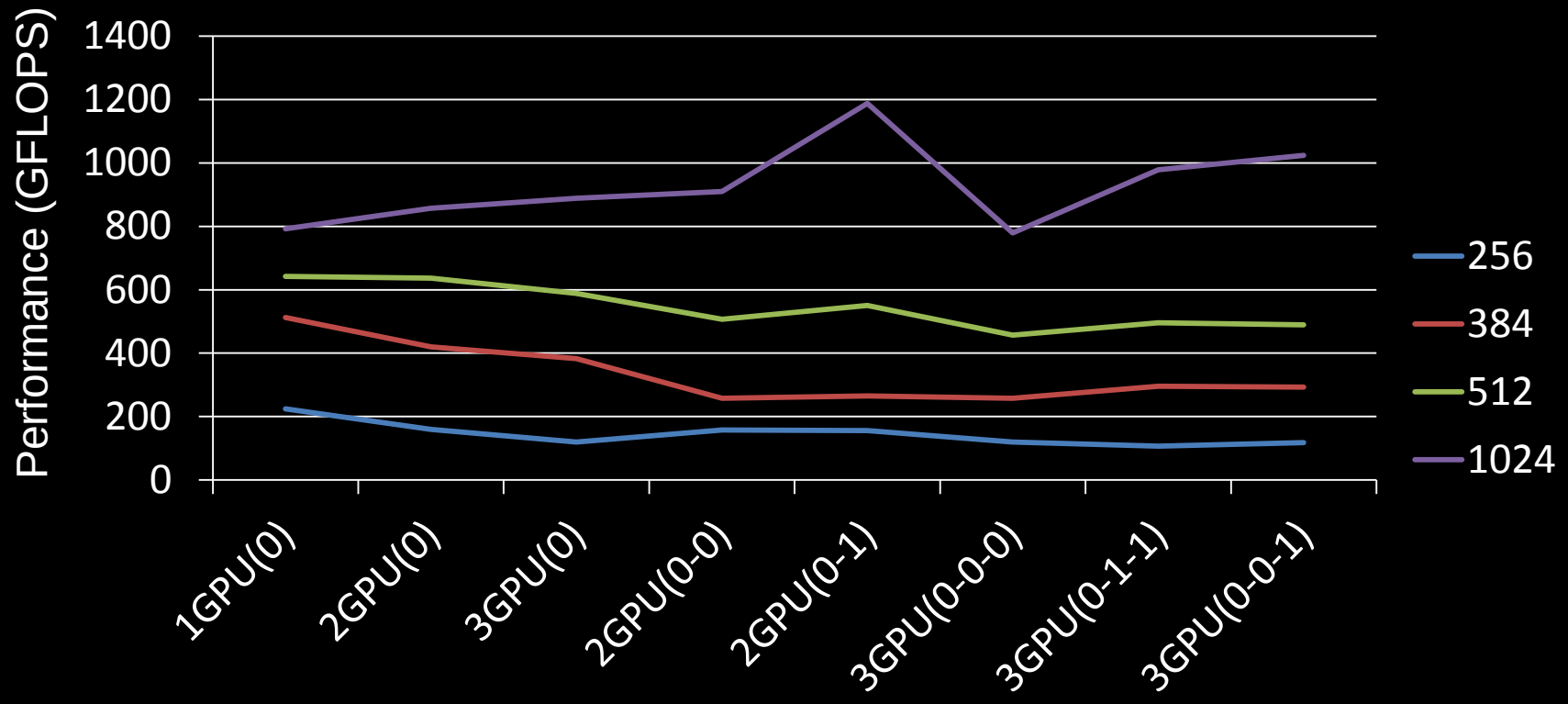


3GPU (0-1-1)



3GPU (0-0-1)

Performance with 64 nodes



Summary

- Speed-up of 3-D FFT using multi-GPU
  - Up to 2X using 4 GPUs (single node)
  - Up to 4X using 32GPUs (32nodes)
- Small message in all-to-all limits the scalability
- Use of multi-GPU per node
  - Decrease performance for small data size
  - Increase performance for large data size