# 多GPU并行编程

## 需要用到MPI\_CUDA编程的场合

* 数据量太大，超过了单个GPU的内存；
* 单个节点的计算时间过长；
* 使用GPU加速已有的MPI并行程序；
* 使单节点多GPU程序可以在多计算节点间通信。

## CUDA-aware MPI

MPI简介

MPI标准定义了信息传送API，包括点对点通信以及集合操作，如reduction。下面的C程序将”Hello, there”从进程0发送到进程1. MPI进程通常称为”rank”，通过调用MPI\_Comm\_rank()获得rank，source.c如下：

#include <stdio.h>

#include <string.h>

#include <mpi.h>

int main(int argc, char \*argv[])

{

char message[20];

int myrank, tag=99;

MPI\_Status status;

/\* Initialize the MPI library \*/

MPI\_Init(&argc, &argv);

/\* Determine unique id of the calling process of all processes participating

in this MPI program. This id is usually called MPI rank. \*/

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myrank);

if (myrank == 0) {

strcpy(message, "Hello, there");

/\* Send the message "Hello, there" from the process with rank 0 to the

process with rank 1. \*/

MPI\_Send(message, strlen(message)+1, MPI\_CHAR, 1, tag, MPI\_COMM\_WORLD);

} else {

/\* Receive a message with a maximum length of 20 characters from process

with rank 0. \*/

MPI\_Recv(message, 20, MPI\_CHAR, 0, tag, MPI\_COMM\_WORLD, &status);

printf("received %s\n", message);

}

/\* Finalize the MPI library to free resources acquired by it. \*/

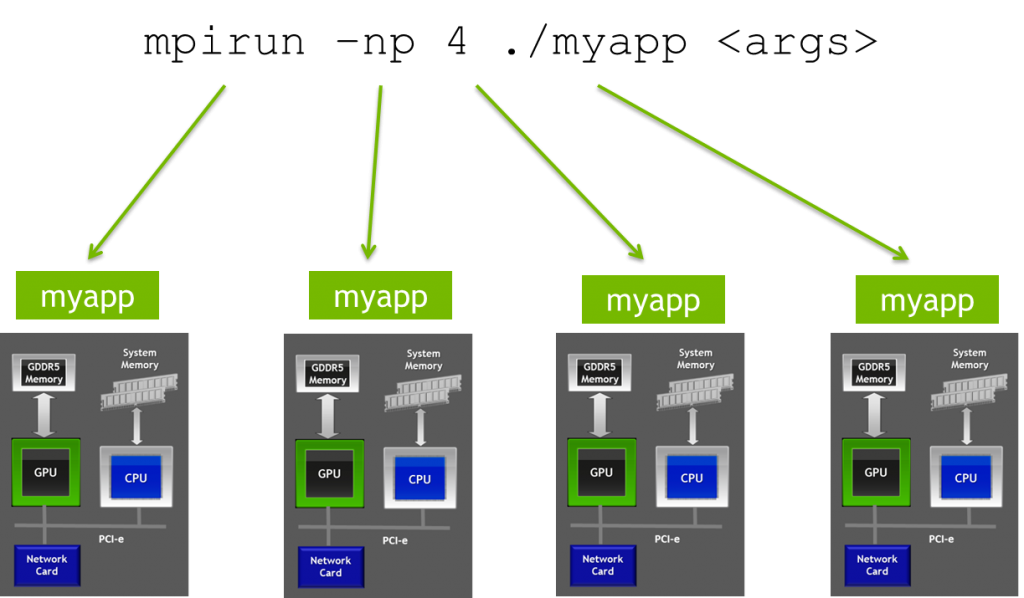
MPI\_Finalize();

return 0;

}

编译：mpicc source.c -o myapp

运行：



什么是CUDA-aware MPI？

一般情况下，MPI传递指针给host内存，然后使用cudaMemcopy将GPU缓存数据传输给host内存。CUDA-aware MPI可以直接发送和接收GPU缓存间的数据。没有CUDA-aware MPI，需要使用cudaMemcpy，通过主机内存交换GPU缓存：

//MPI rank 0

cudaMemcpy(s\_buf\_h,s\_buf\_d,size,cudaMemcpyDeviceToHost);

MPI\_Send(s\_buf\_h,size,MPI\_CHAR,1,100,MPI\_COMM\_WORLD);

//MPI rank 1

MPI\_Recv(r\_buf\_h,size,MPI\_CHAR,0,100,MPI\_COMM\_WORLD, &status);

cudaMemcpy(r\_buf\_d,r\_buf\_h,size,cudaMemcpyHostToDevice);

有了CUDA-aware MPI，无需上面的交换，GPU缓存直接传递给MPI：

//MPI rank 0

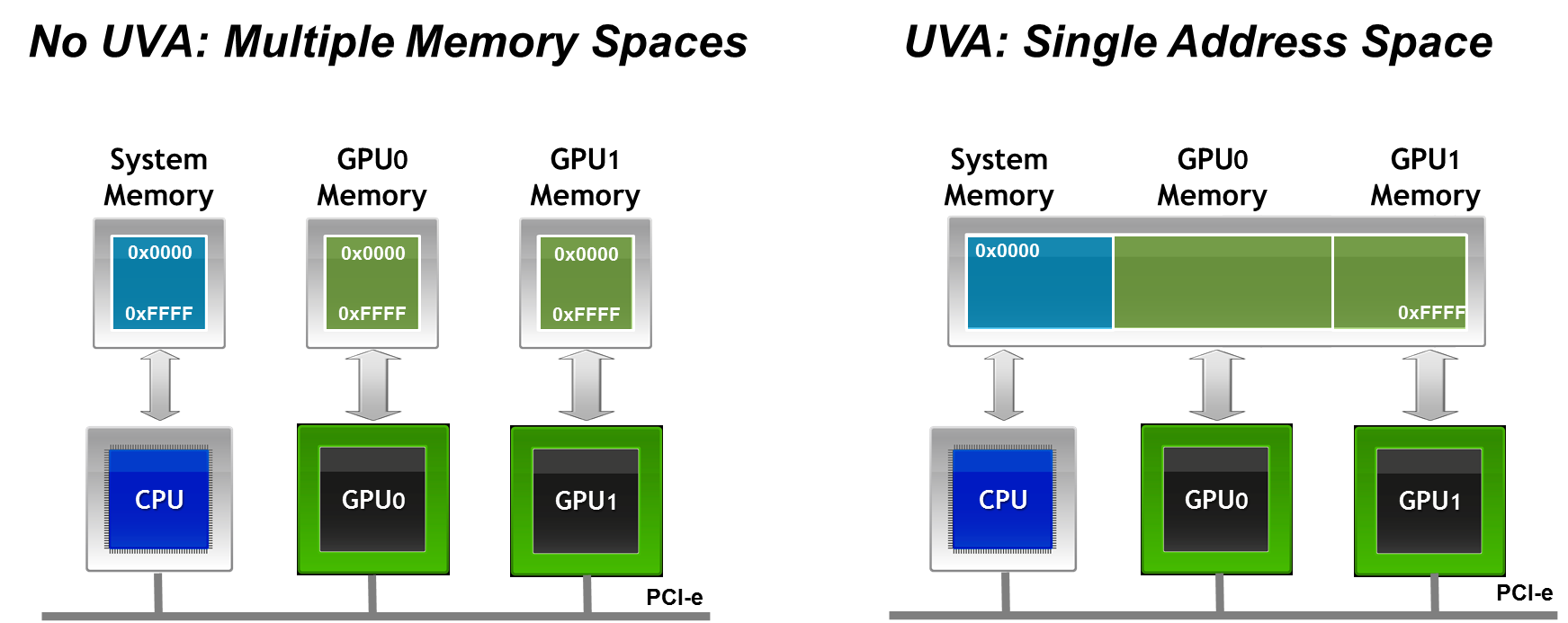
MPI\_Send(s\_buf\_d,size,MPI\_CHAR,1,100,MPI\_COMM\_WORLD);

//MPI rank n-1

MPI\_Recv(r\_buf\_d,size,MPI\_CHAR,0,100,MPI\_COMM\_WORLD, &status);

CUDA-aware MPI如何工作？

CUDA 4.0开始支持Unified Virtual Addressing (UVA) (CUDA 4.0, Compute Capability 2.0 and later GPUs)简化了CUDA-aware MPI，所有的CPU和GPU内存统一为一个地址空间。



使用UVA，缓存位置基于地址的MSB可以确定，因此无需改变MPI的API。

CUDA-aware MPI不仅让CUDA+MPI程序使用更容易，还是程序更高效，有2个原因：

* Operations that carry out the message transfers can be pipelined
* CUDA-aware MPI takes advantage of best GPUDirect technology available

支持CUDA-aware MPI库

* [MVAPICH2](https://developer.nvidia.com/mvapich) (ver.???)
* open mpi (1.7 or later)

。。。 。。。

## GPUDirect技术

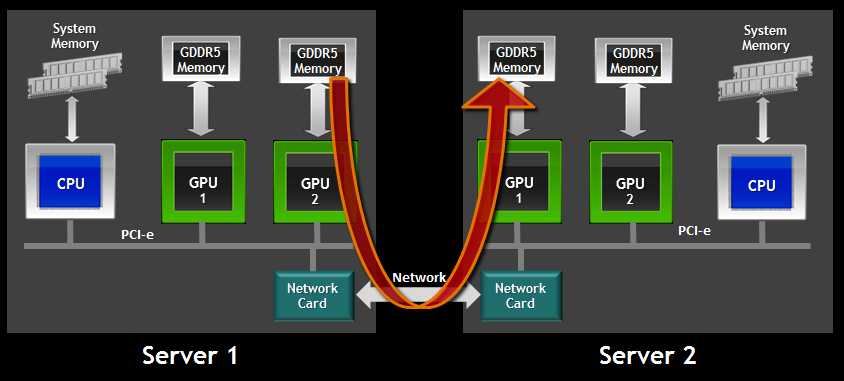
Using GPUDirect, multiple GPUs, third party network adapters, solid-state drives (SSDs) and other devices can directly read and write CUDA host and device memory, eliminating unnecessary memory copies, dramatically lowering CPU overhead, and reducing latency, resulting in significant performance improvements in data transfer times for applications running on NVIDIA Tesla™ and Quadro™ products.

NVIDIA GPUDirect technologies provide high-bandwidth, low-latency communications with NVIDIA GPUs. GPUDirect is an umbrella name used to refer to several specific technologies. In the context of MPI the GPUDirect technologies cover all kinds of inter-rank communication: intra-node, inter-node, and RDMA inter-node communication. （GPUDirect是一类技术的总称。）

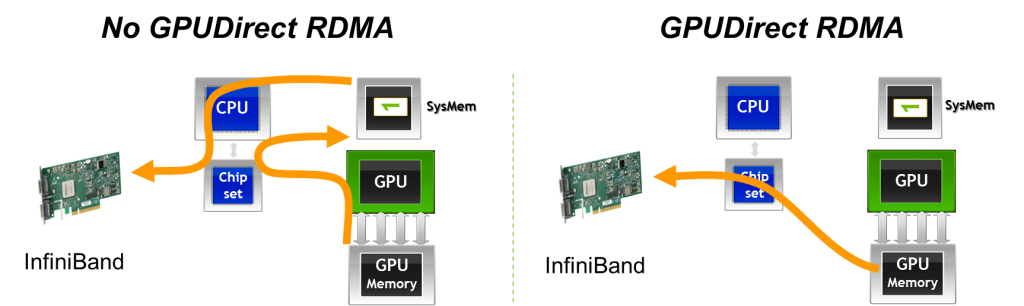
（1）GPUDirect RDMA （计算节点间）

The newest GPUDirect feature, introduced with CUDA 5.0, is support for Remote Direct Memory Access (**RDMA**), with which buffers can be directly sent from the GPU memory to a network adapter without staging through host memory.（GPU与数据交换机之间的直接数据传递）

Eliminate CPU bandwidth and latency bottlenecks using remote direct memory access (RDMA) transfers between GPUs and other PCIe devices, resulting in significantly improved MPISendRecv efficiency between GPUs and other nodes)

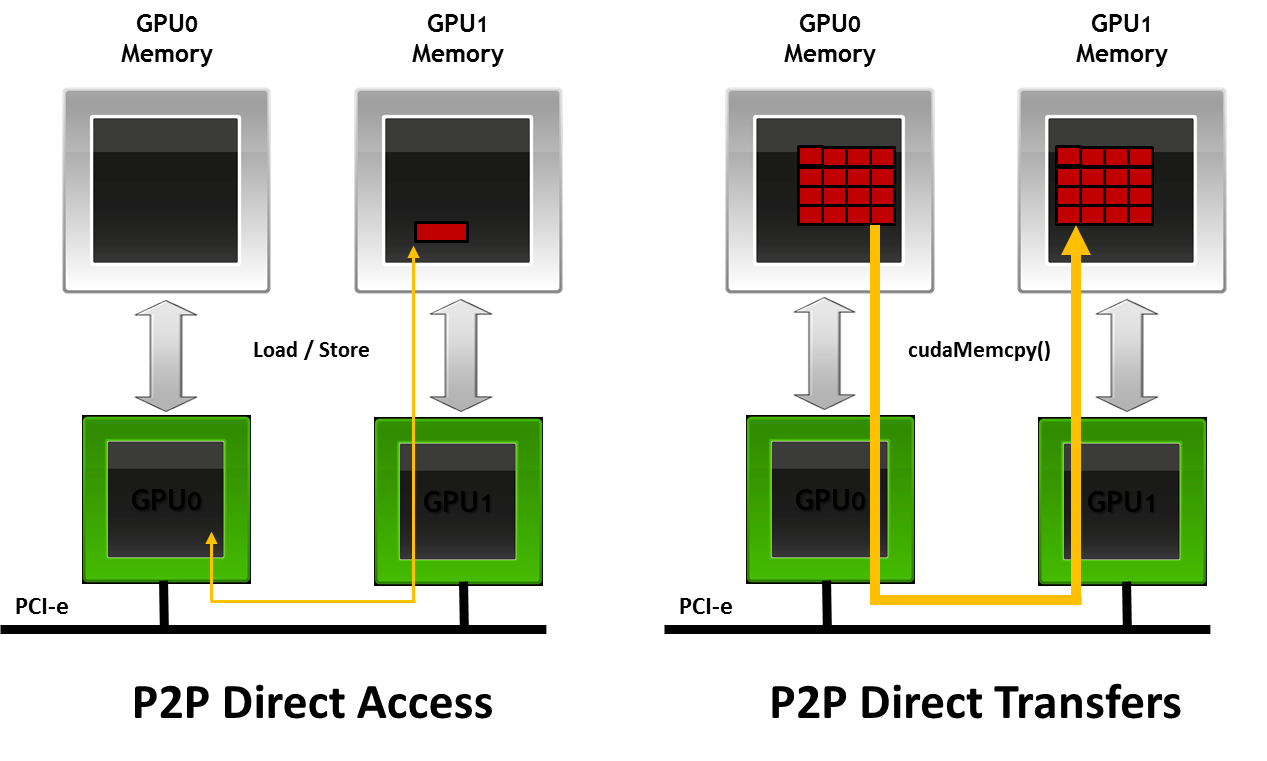


GPUDirect™ Support for RDMA, Introduced with [CUDA 5 (2012)](https://developer.nvidia.com/cuda-toolkit)



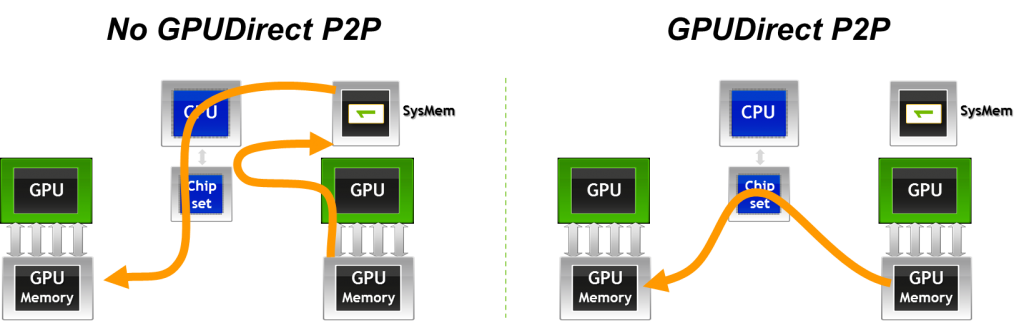
（2）Peer-to-Peer Transfers between GPUs（计算节点内GPU间的直接数据传递）

Use high-speed DMA transfers to copy data between the memories of two GPUs on the same system/PCIe bus.



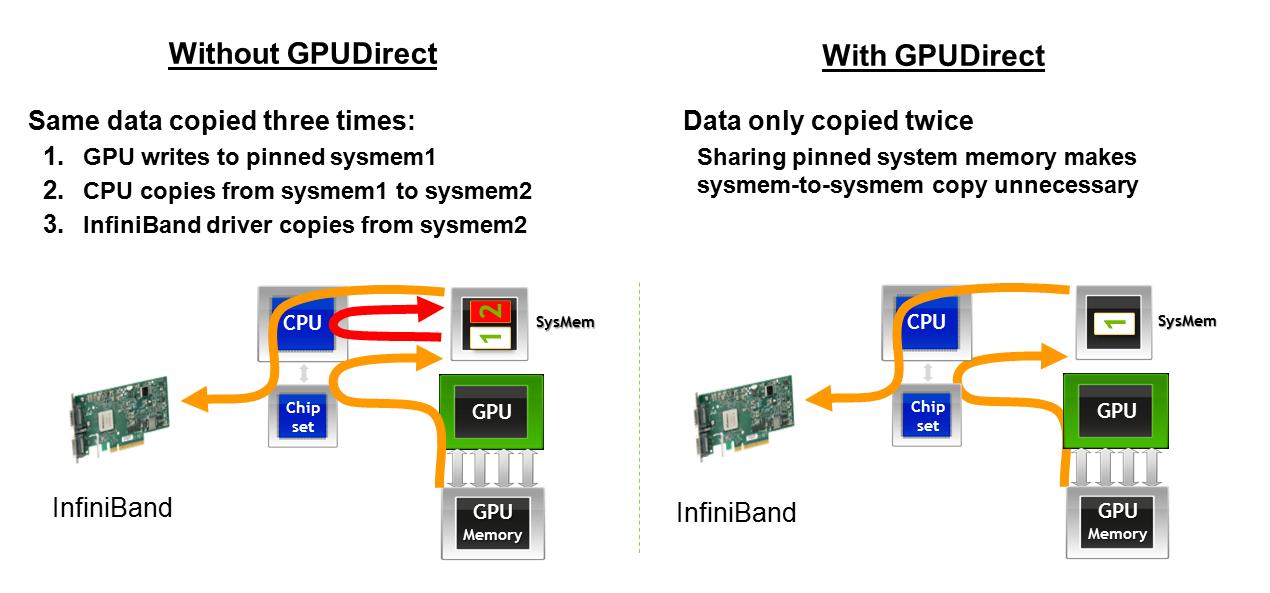
NVIDIA GPUDirect Peer-to-Peer (P2P) Communication Between GPUs on the Same PCIe Bus (2011)

Another variant is GPUDirect for Peer-to-Peer (P2P) transfers, which was introduced with CUDA 4.0 and can accelerate intra-node communication. Buffers can be directly copied between the memories of two GPUs in the same system with GPUDirect P2P.



（3）GPUDirect (Pined memory)

GPUDirect for accelerated communication with network and storage devices was the first GPUDirect technology, introduced with CUDA 3.1. This feature allows the network fabric driver and the CUDA driver to share a common pinned buffer in order to avoids an unnecessary memcpy within host memory between the intermediate pinned buffers of the CUDA driver and the network fabric buffer.



GPUDirect™ Shared Memory (2010)