PGI® COMPILERS & TOOLS

PROGRAMMING GPU SUPERCOMPUTERS TEN YEARS FROM NOW

Michael Wolfe, March 26, 2018

NVIDIA.

PGI — THE NVIDIA HPC SDK

Fortran, C++, C Compilers

- Optimizing, SIMD Vectorizing, OpenMP

Accelerated Computing Features

- OpenACC Directives, CUDA Fortran

Multi-Platform Solution

- X86-64 and OpenPOWER Multicore CPUs
- NVIDIA Tesla GPUs
- Supported on Linux, macOS, Windows

MPI/OpenMP/OpenACC Tools

- Debugger
- Performance Profiler
- Interoperable with DDT, TotalView

NVIDIA

PGI®

The Compilers & Tools
for Supercomputing

NVIDIA CUDA Compute Unified Device Architecture

Programming Guide

Version 1.0

6/23/2007

SC07 is the International Conference for High Performance Computing, Networking, Storage and Analysis

SEARCH

About | Registration | Conference | Technical Program | Exhibits | News & Press | Travel

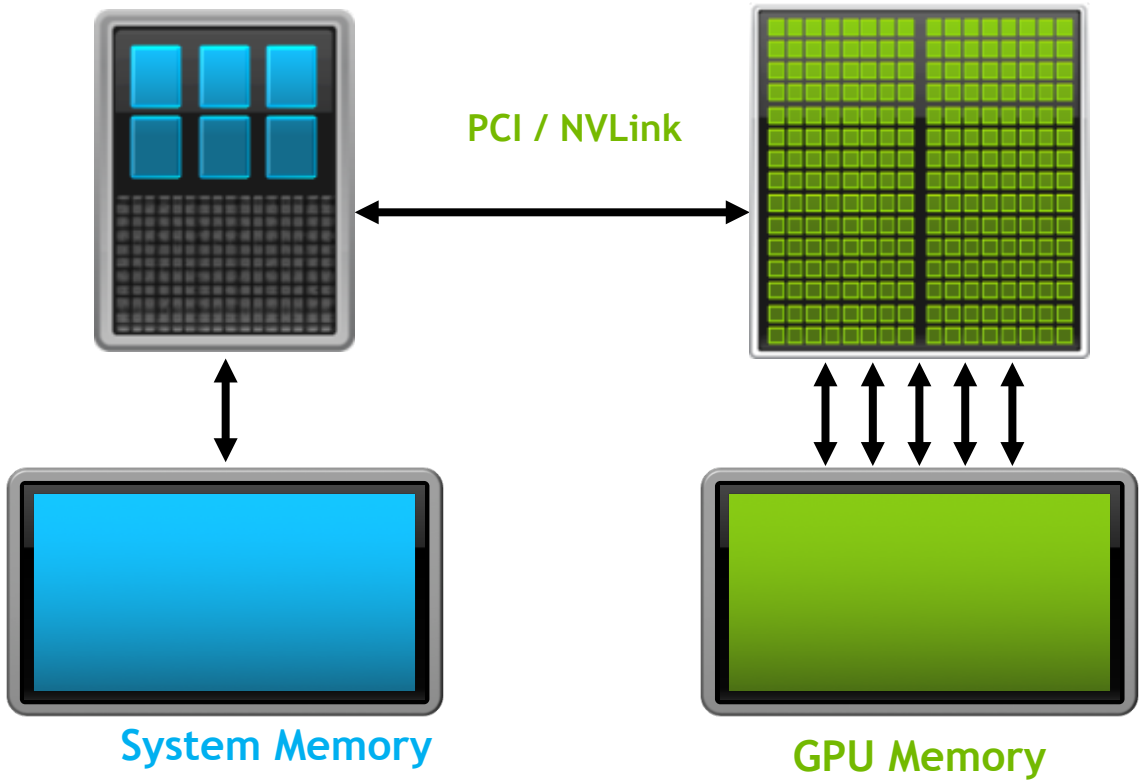
- Overview
- Awards

Tutorials

The SC07 Tutorials Program gives attendees the opportunity to explore a wide variety of important topics related to high performance computing, networking, and storage.

		and Workflow on Grids	Sabula	
8:30AM - 5:00PM	S05	High Performance Computing on GPUs with CUDA	Massimiliano Fatica, David P. Luebke, Ian A. Buck, John D. Owens, Mark J. Harris, John E. Stone, James C. Phillips, Bernard Deschizeaux	A3
8:30AM - 5:00PM	S06	Introduction to Globus	Jennifer M. Schopf, Ben Clifford, Ravi	A20

GPU COMPUTING



data management

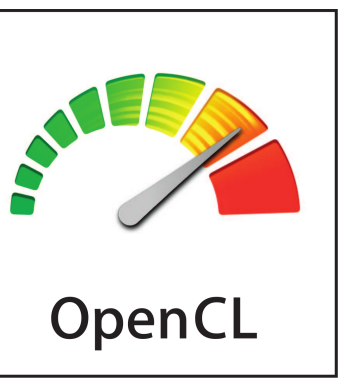
compute management

compute optimization

asynchronous operation

PROGRAMMING GPUS TODAY

OpenACC



OpenMP

libraries

data management

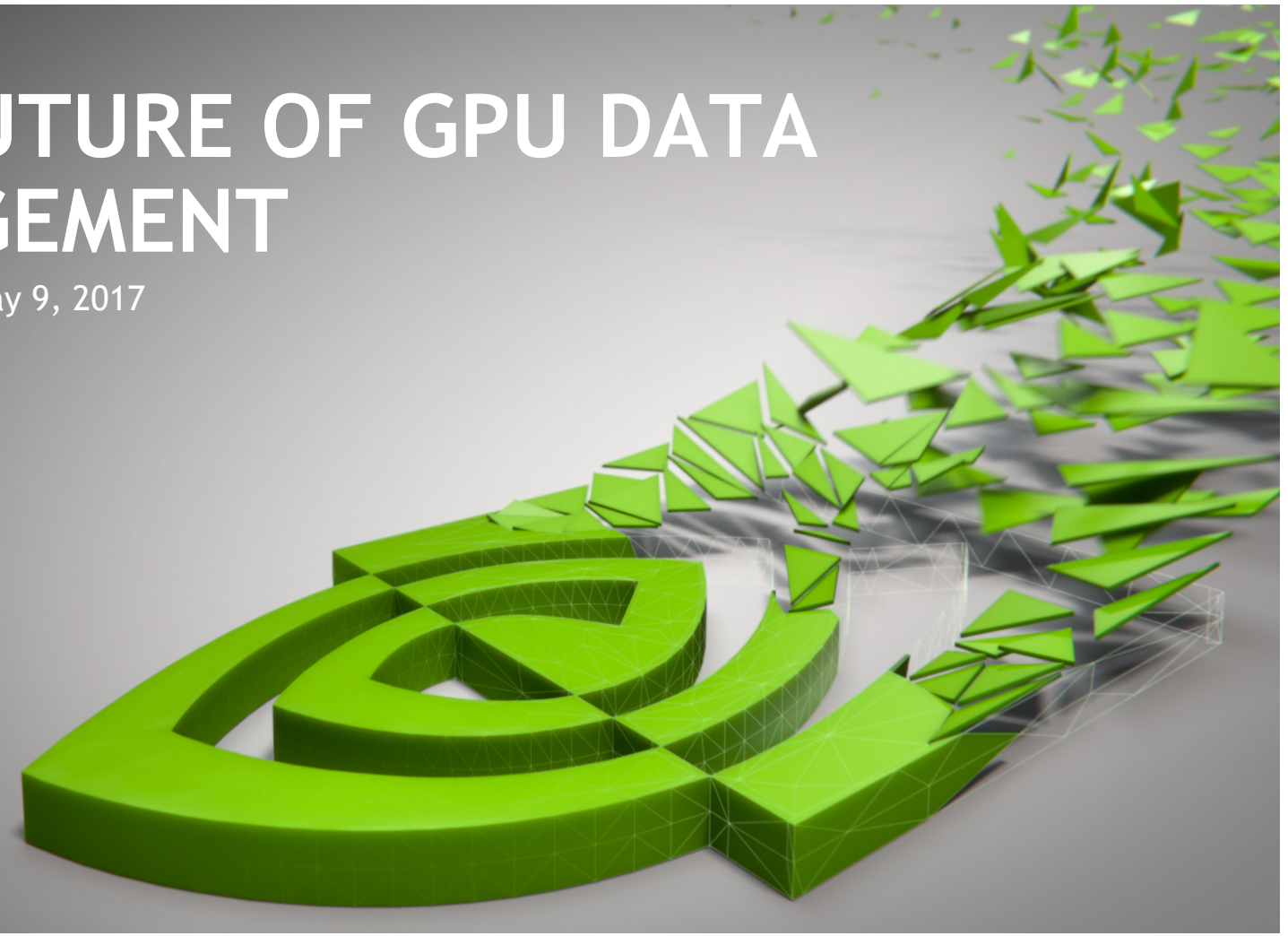
compute management

compute optimization

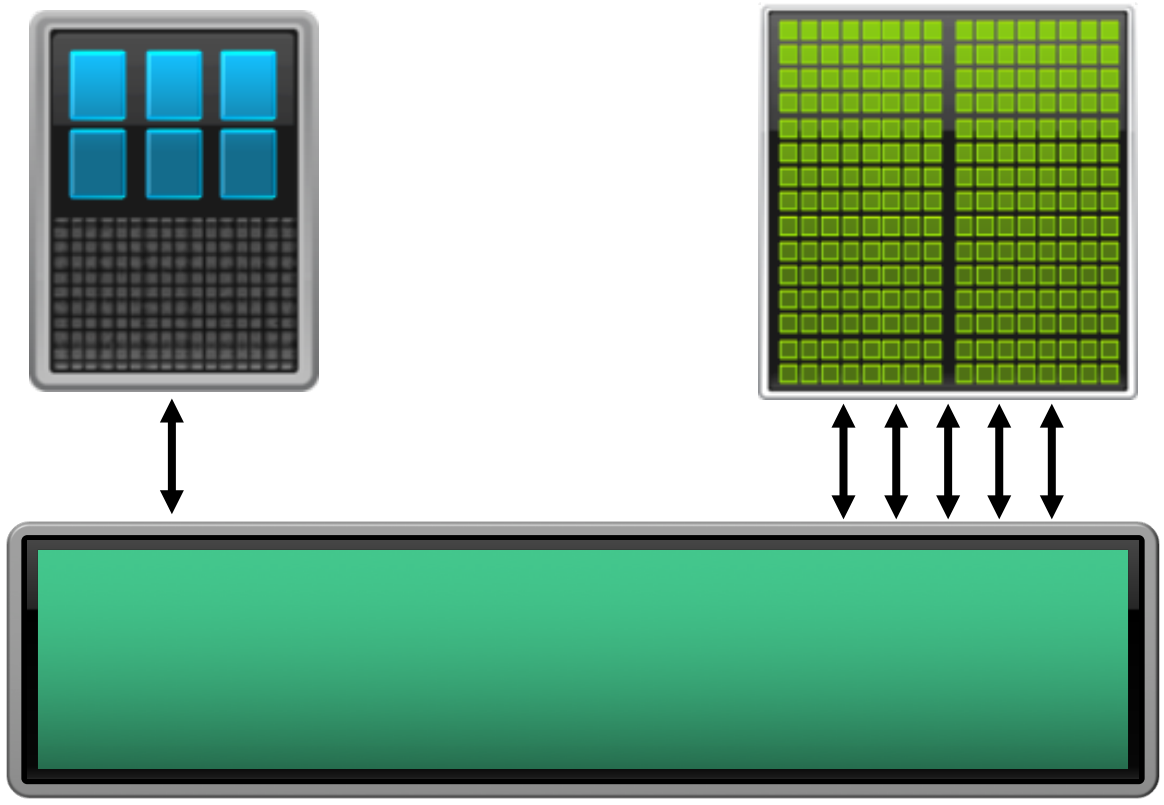
asynchronous operation

THE FUTURE OF GPU DATA MANAGEMENT

Michael Wolfe, May 9, 2017



GPU COMPUTING



data management

compute management

compute optimization

asynchronous operation

GPU PROGRAMMING

Focusing on HPC In Ten Years

This presentation contains predictions and forward-looking statements

All opinions are those of the presenter

GPU PROGRAMMING

Focusing on HPC In Ten Years

C++

Fortran

GPU PROGRAMMING

Focusing on HPC In Ten Years

C++

```
for_each_n(par, 0, n, [&](int i){
    y[i] += a*x[i];
});
```

Fortran

```
do concurrent (i = 1:n)
  y(i) = y(i) + a*x(i)
enddo
```

COMPUTE MANAGEMENT

Focusing on HPC In Ten Years

All procedures compiled for both host and device execution

- System calls
- Function pointers, virtual function tables
- Language runtime fully ported to device

Statically linking host and device code to same address space

Dynamic linking

COMPUTE OPTIMIZATION

Focusing on HPC In Ten Years

Lots of creative work to be done here:

- Selecting which code sections to leave on the host
- Optimizing compute schedules (threadIdx.x/blockIdx.x mappings)
- Optimizing data layouts
- Launching asynchronous kernels

KNOWN HARD PROBLEMS

- Spread work across CPU as well as GPU (load balancing)
- Spread work across multiple GPUs

UNKNOWN UNKNOWNS

What capabilities will GPUs have in ten years

2007

- texture cache: dedicated texture cache, hardware interpolation

2017

- tensor cores: 4x4 HP matmul with 4x4 SP matadd

2027

- ?

GPU PROGRAMMING

Focusing on HPC In Ten Years

C++

```
for_each_n(par, 0, n, [&](int i){
    y[i] += a*x[i];
});
```

Fortran

```
do concurrent (i = 1:n)
  y(i) = y(i) + a*x(i)
enddo
```

Choosing which parallel constructs to offload?
Optimizing the mapping to GPU thread and grid indices?
Controlling placement of data in memory spaces?
Mapping to asynchronous operations?
Satisfying Lisa the Library Writer as well as Phil the Physicist?