4TH RTC4AO
OBSERVATOIRE DE PARIS DECEMBER 19TH - 21ST 2016

Programming heterogeneous architectures
Introduction to Pascal architecture & Unified Memory

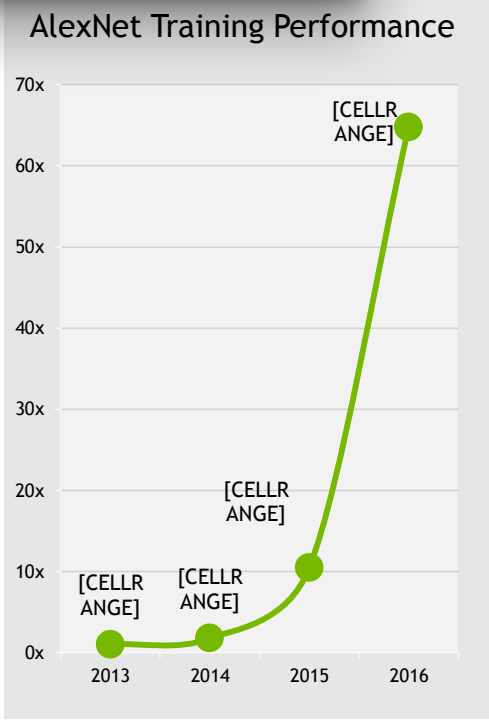
François Courteille |Principal Solutions Architect, NVIDIA |fcourteille@nvidia.com

ONE ARCHITECTURE BUILT FOR BOTH DATA SCIENCE & COMPUTATIONAL SCIENCE

Pascal- 5 Miracles

NVIDIA DGX-1

NVIDIA DGX SATURNV



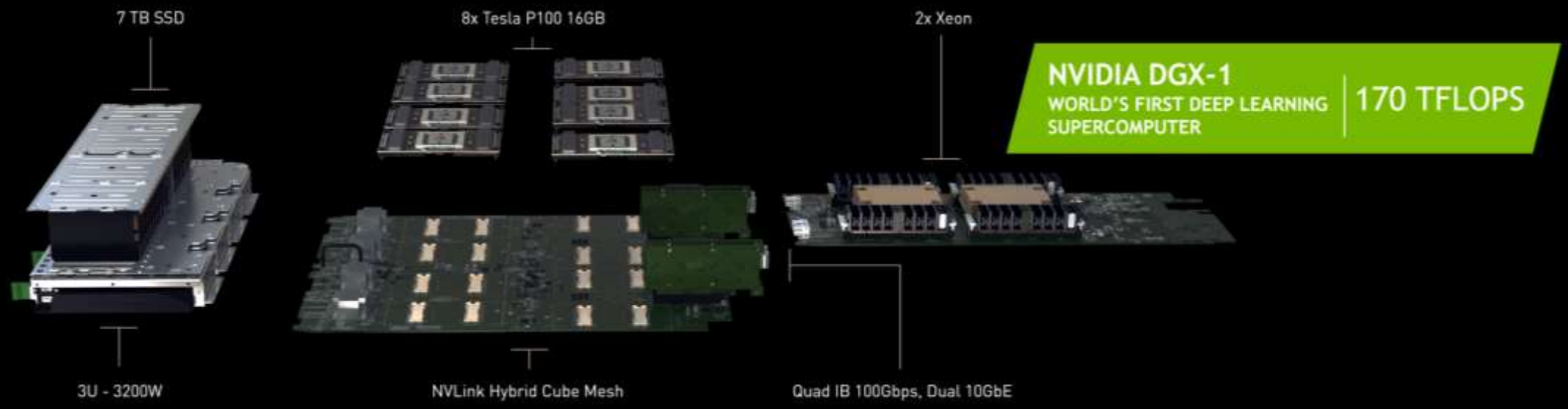
65x in 3 Years

AGENDA

The Tesla Platform : architecture and future
Rapid software development for heterogeneous architecture

TESLA PLATFORM

NVIDIA DGX-1 DEEP LEARNING SYSTEM



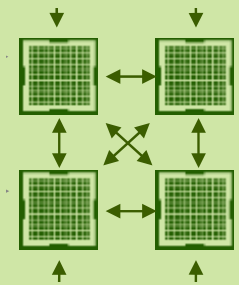
INTRODUCING TESLA P100

New GPU Architecture to Enable the World's Fastest Compute Node

Pascal Architecture

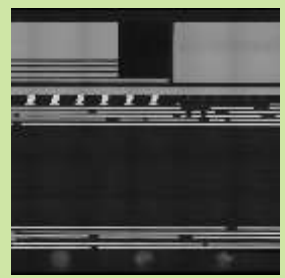
Highest Compute Performance

NVLink



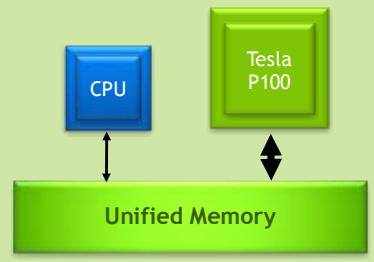
GPU Interconnect for Maximum Scalability

HBM2 Stacked Memory



Unifying Compute & Memory in Single Package

Page Migration Engine



Simple Parallel Programming with 512 TB of Virtual Memory

Tesla P100 ACCELERATORS

	Tesla P100	Tesla P40	Tesla P4
Compute	5.3 TF DP · 10.6 TF SP · 21.2 TF HP	11.8 TF SP · 47 TOP/s INT8	5.4 TF SP · 21.8 TOP/s INT8
Memory	HBM2 16GB: 732 GB/s | HBM2 12GB: 549 GB/s	GDDR5: 346 GB/s · 24 GB	GDDR5: 192 GB/s · 8GB
Interconnect	NVLink + PCIe Gen3	PCIe Gen3	PCIe Gen3
Programmability	Page Migration Engine Unified Memory	Page Migration Engine Unified Memory	Page Migration Engine Unified Memory
Power	300W SXM2 | 250W PCIe	250W	50W & 75W

(Compute #s based on boost clocks; P100 compute #s given for SXM2 version)

IEEE 754 FLOATING POINT ON GP100

3 sizes, 3 speeds, all fast

Feature	New Half precision	Single precision	Double precision
Layout	s5.10	s8.23	s11.52
Issue rate	pair every clock	1 every clock	1 every 2 clocks
Subnormal support	Yes	Yes	Yes
Atomic Addition	Yes	Yes	New Yes

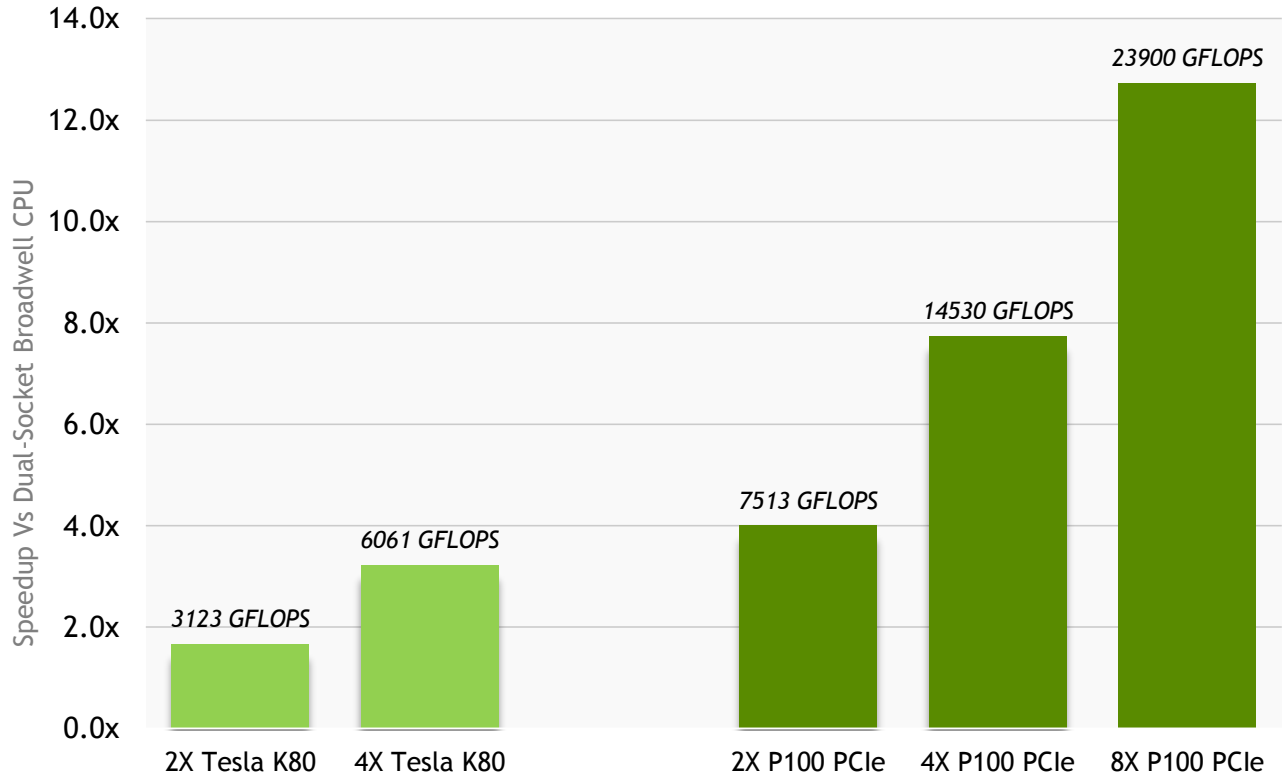
LINPACK

Benchmark
Measures floating point computing power

Accelerated Features	Metric	Scalability
All	GFLOPS	Multi-GPU, Multi Node

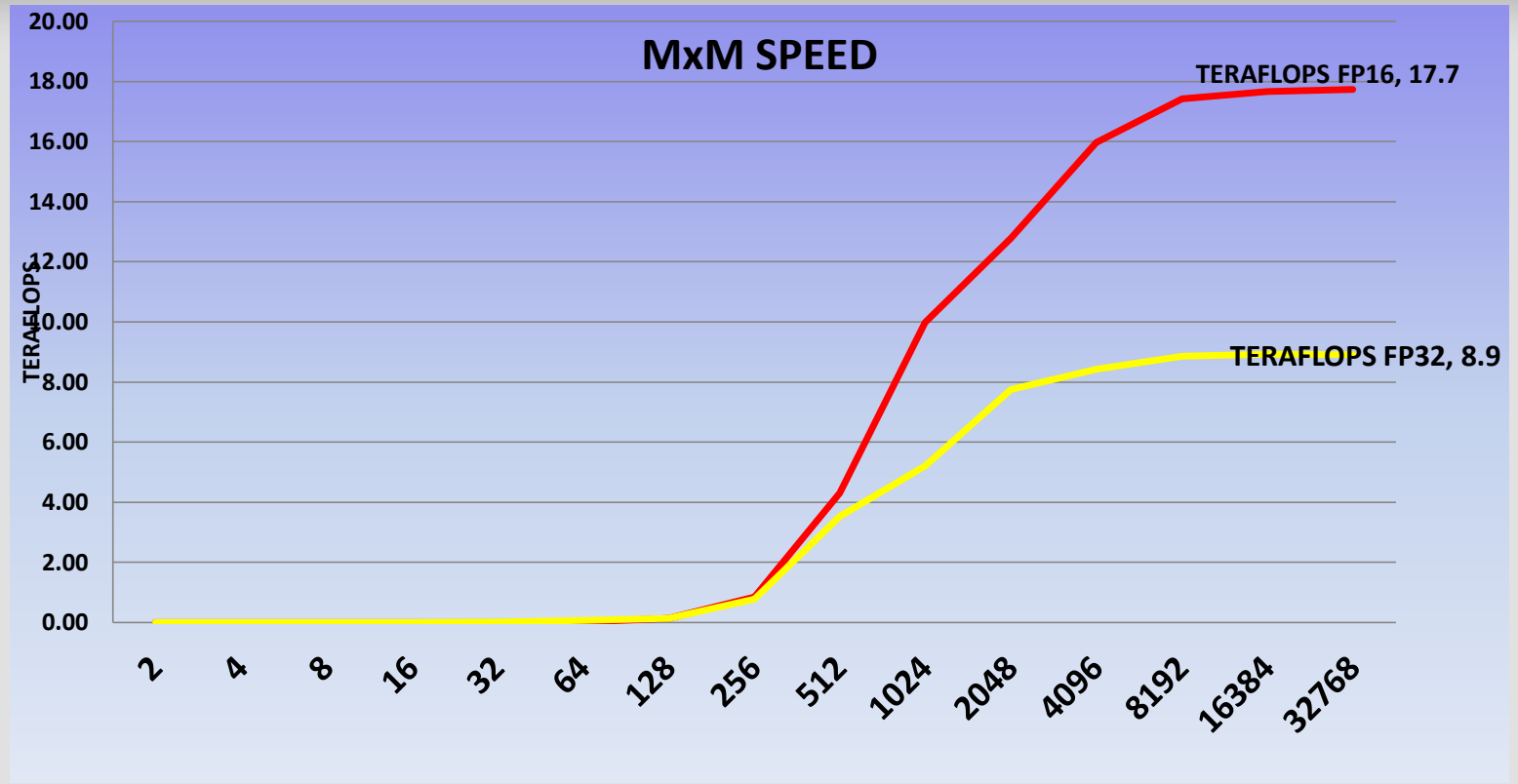
https://www.top500.org/project/**linpack**/

Linpack 2.1
Speedup Vs Dual-Socket CPU Server



CPU Server: Dual Xeon E5-2699 v4@2.2GHz (44-cores)
GPU Servers: Dual Xeon E5-2699 v4@2.2GHz (44-cores) with Tesla K80s or P100s PCIe
CUDA Version: CUDA 8.0.44
Dataset: HPL.dat

MxM PERFORMANCE (CUBLAS XGEMM)



NVLINK

NVLINK

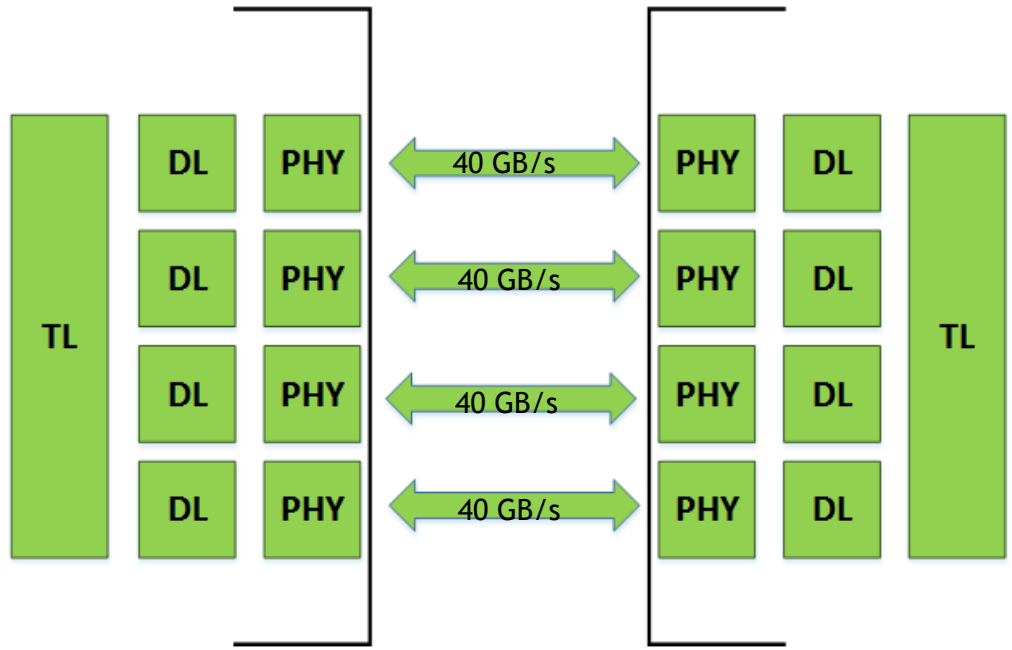
P100 supports 4 NVLinks

Up to 94% bandwidth efficiency

Supports read/writes/atomics to peer GPU

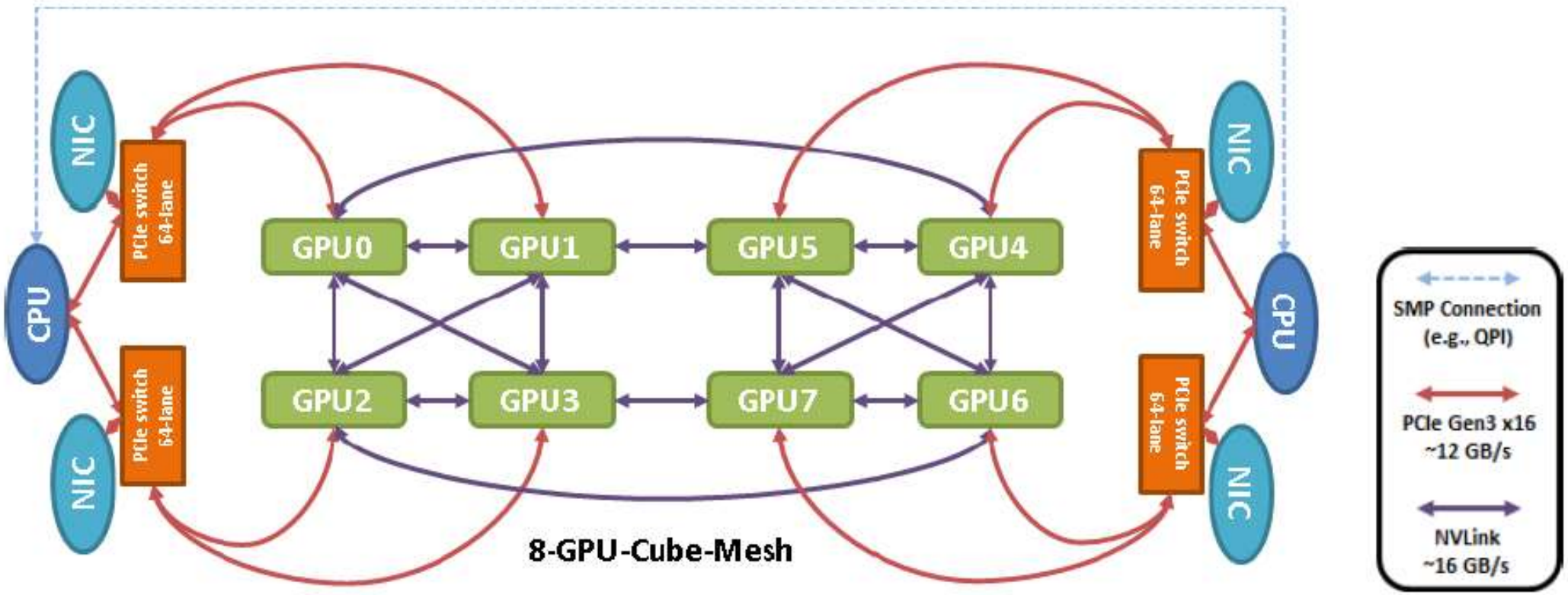
Supports read/write access to NVLink-enabled CPU

Links can be ganged for higher bandwidth



NVLink on Tesla P100

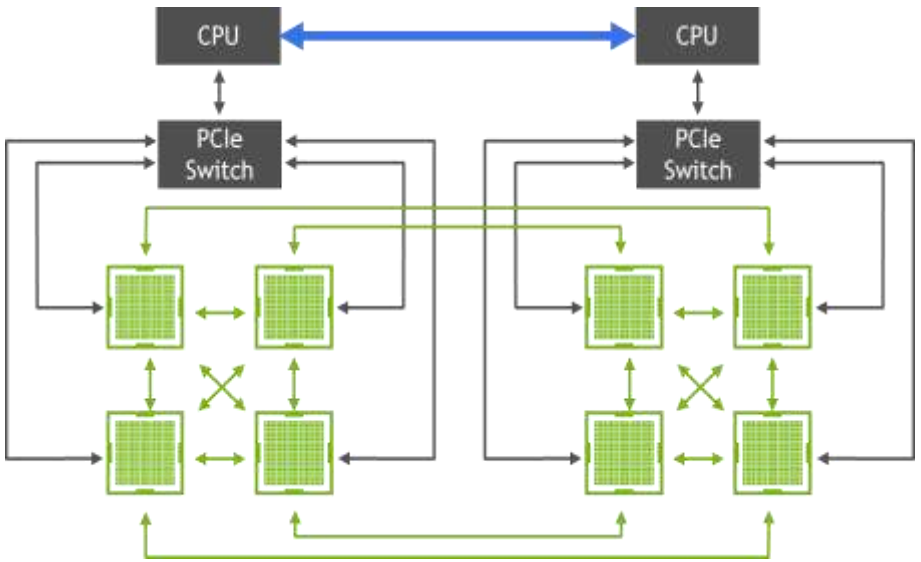
GPU-TO-GPU NVLINK TOPOLOGY



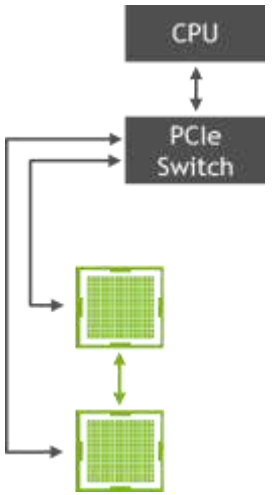
For the 8-GPU-Cube-Mesh topology, there is no need to use PCIe for any GPU-to-GPU communications (whether point-to-point or collective).

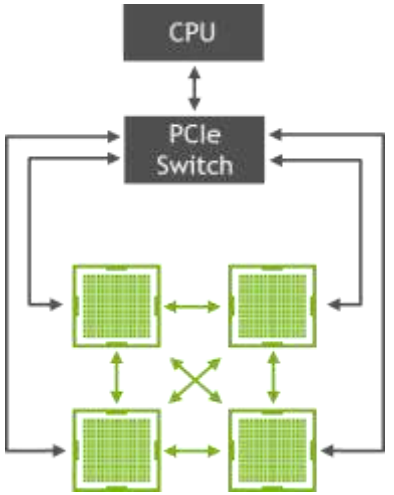
MOST SCALABLE, MOST VERSATILE

8-GPU Cube Mesh with NVLINK



Best in class scaling with 8-GPU for
Wide variety of workloads





Easy to partition into logical systems of
two or four GPUs

NVLINK TO CPU

IBM Power Systems Server S822LC (codename “Minsky”)

2x IBM Power8+ CPUs and 4x P100 GPUs

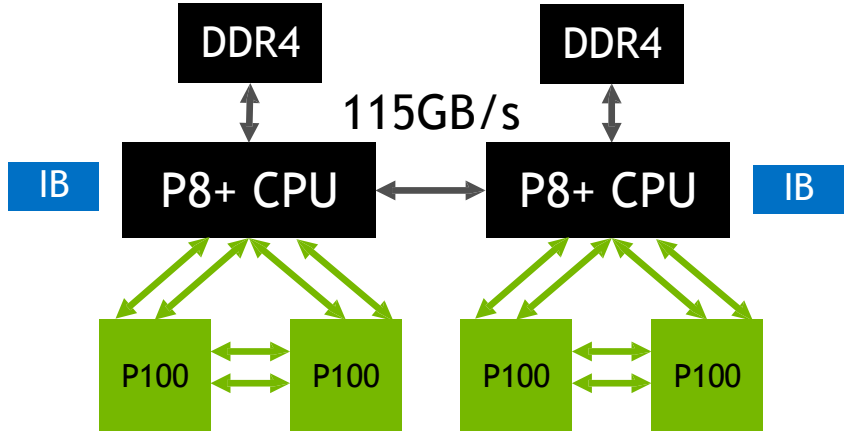
80 GB/s per GPU bidirectional for peer traffic

80 GB/s per GPU bidirectional to CPU

115 GB/s CPU Memory Bandwidth

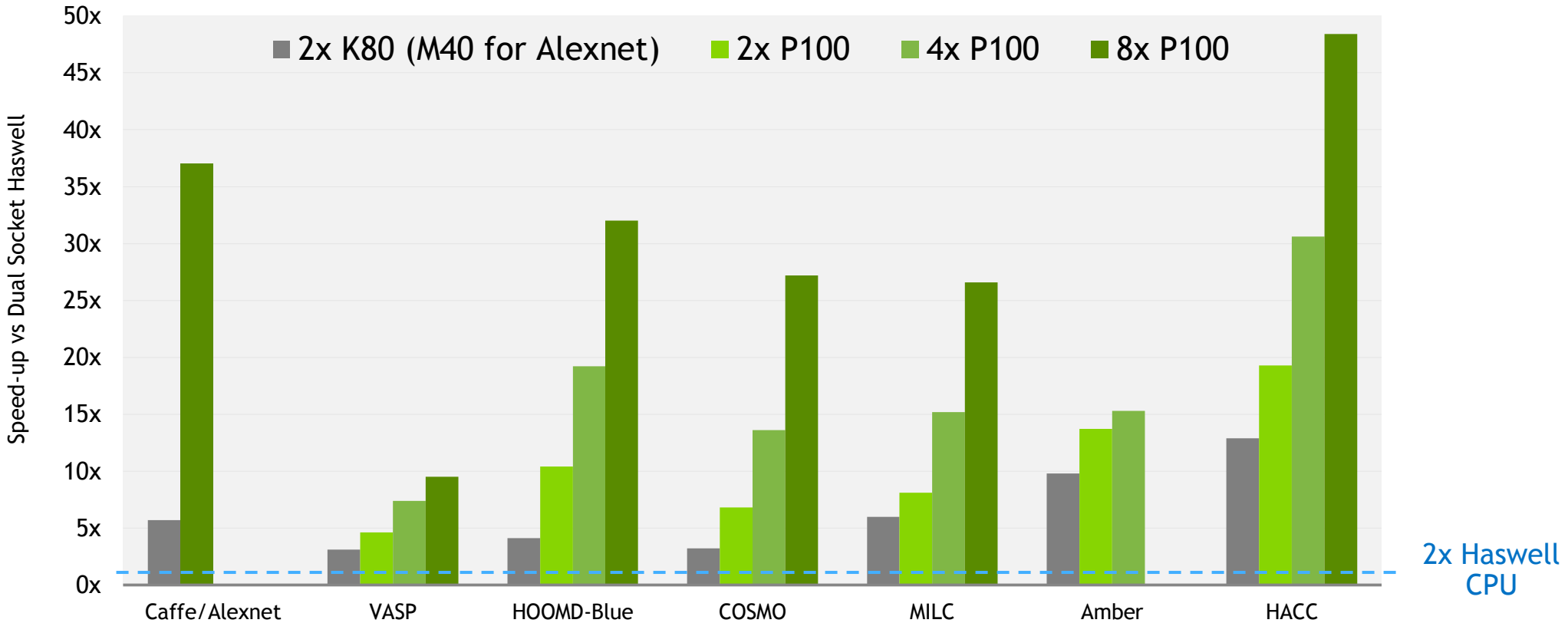
Direct Load/store access to CPU Memory

High Speed Copy Engines for bulk data movement

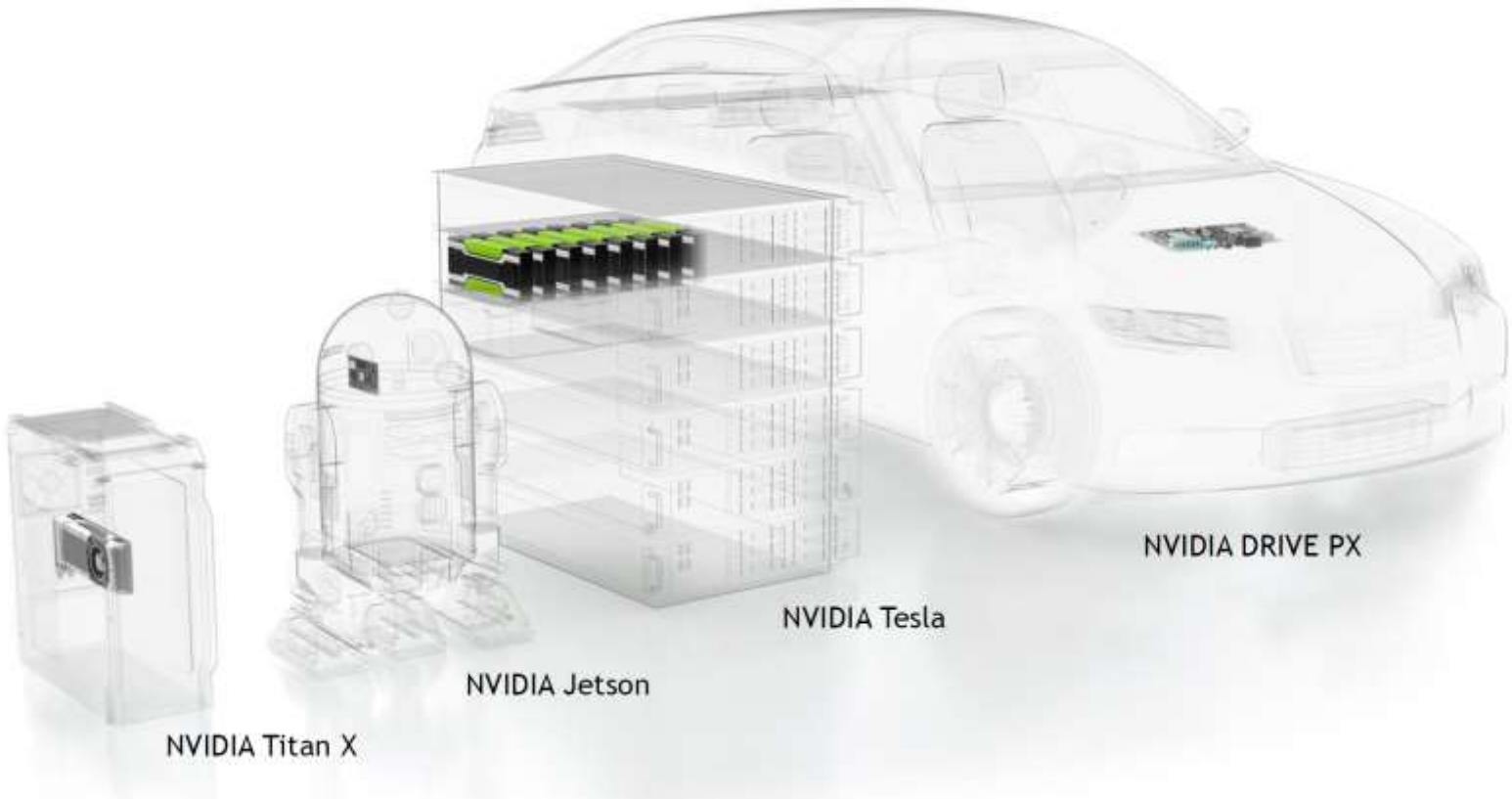


TESLA P100 PERFORMANCE DELIVERED

NVLink for Max Scalability, More than 45x Faster with 8x P100

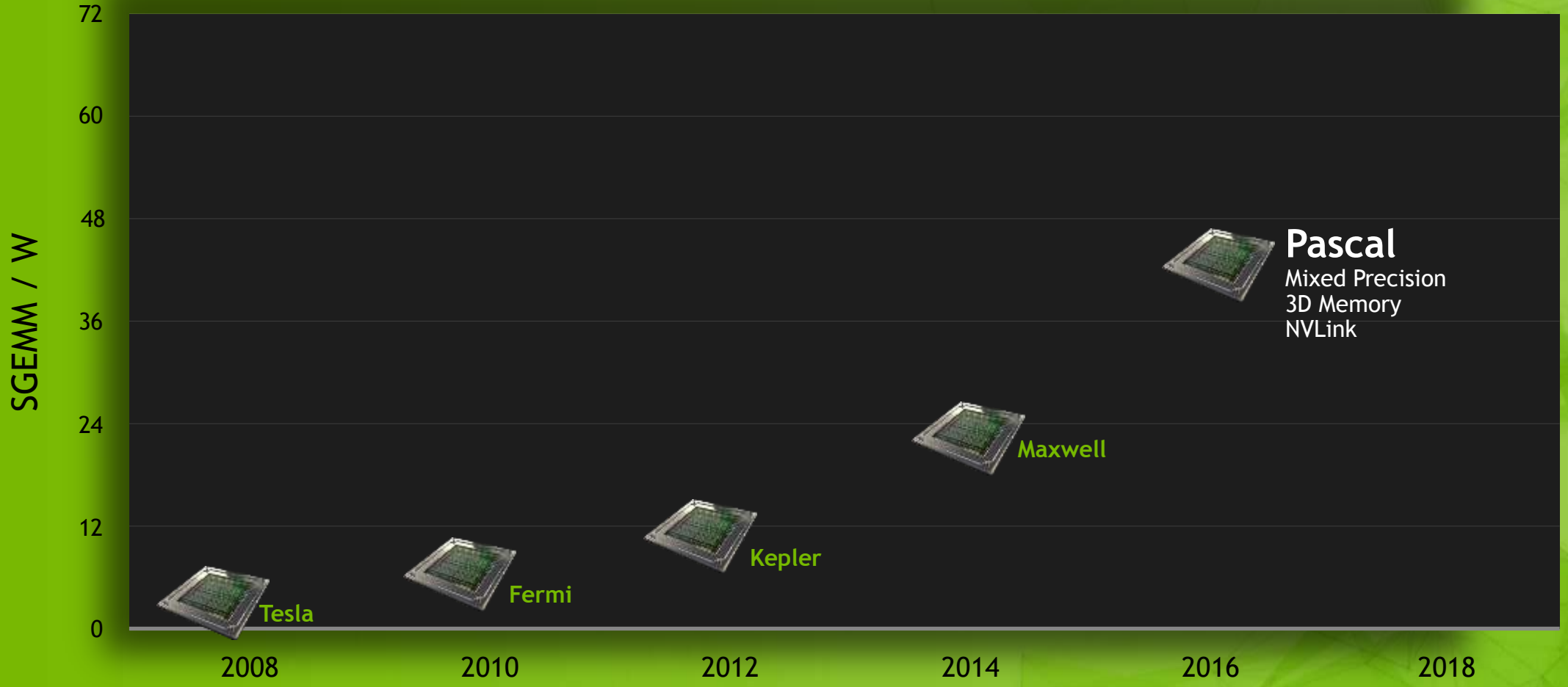


ONE ARCHITECTURE — END-TO-END AI

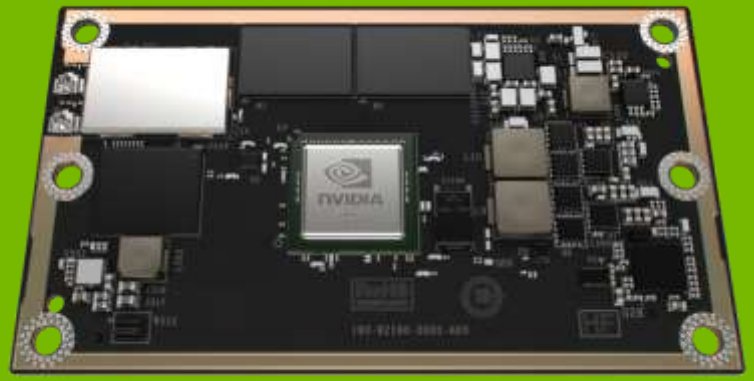


CUDA + Linux throughout the stack.

GPU ROADMAP

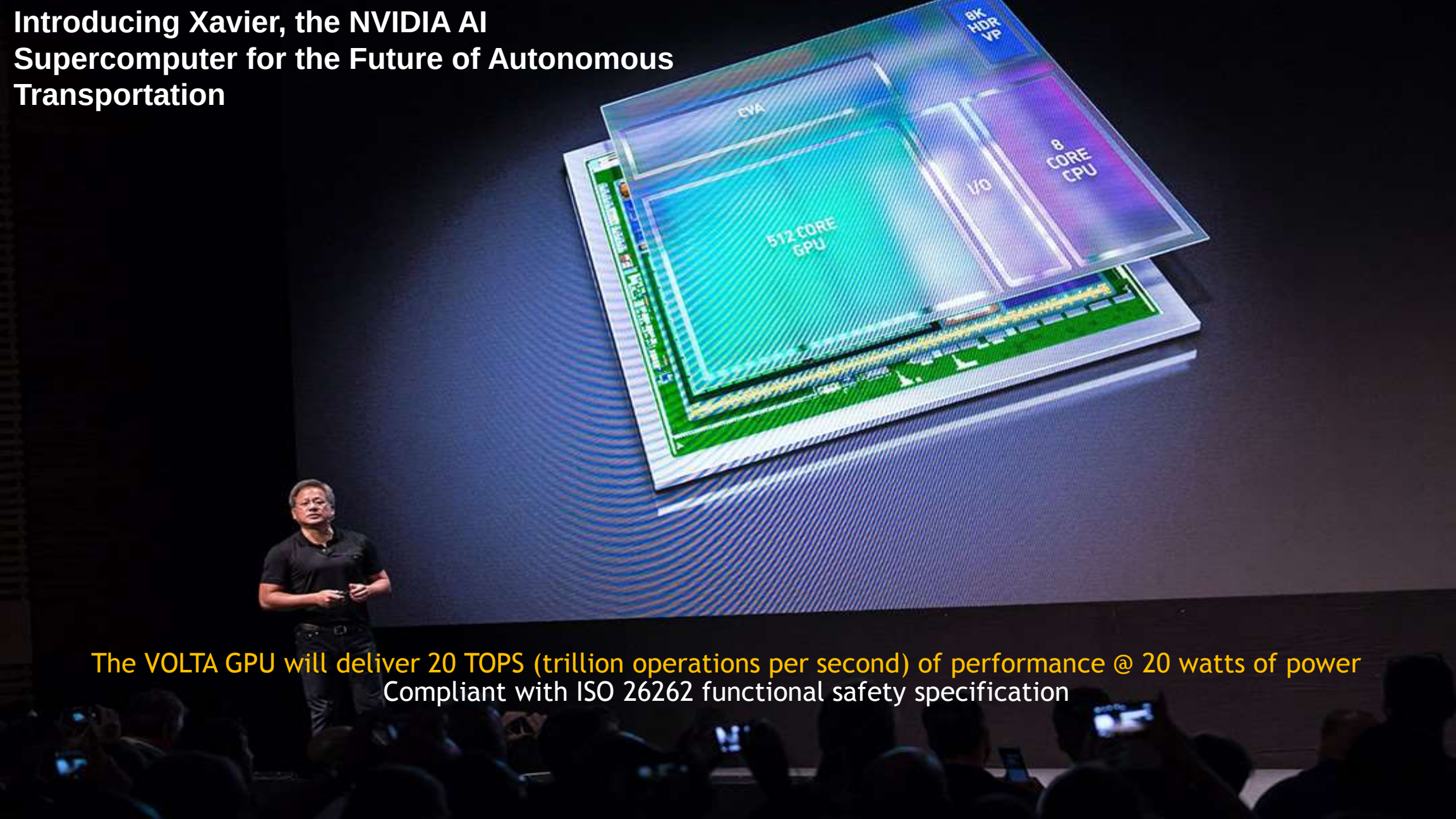

SGEMM / W
72
60
48
36
24
12
0
2008
2010
2012
2014
2016
2018
Tesla
Fermi
Kepler
Maxwell
Pascal
Mixed Precision
3D Memory
NVLink

Jetson TX1



	JETSON TX1
GPU	1024 GFLOPS 256-core Maxwell
CPU	4x 64-bit ARM A57 CPUs | 1.6 GHz
Memory	4 GB LPDDR4 | 25.6 GB/s
Video decode	4K 60Hz H.264 / H.265
Video encode	4K 30Hz H.264 / H.265
CSI	Up to 6 cameras | 1400 Mpix/s
Display	2x DSI, 1x eDP 1.4, 1x DP 1.2/HDMI
Wi-Fi	802.11 2x2 ac
Networking	1 Gigabit Ethernet
PCI-E	Gen 2 1x1 + 1x4
Storage	16 GB eMMC, SDIO, SATA
Other	3x UART, 3x SPI, 4x I2C, 4x I2S, GPIOs
Power	10-15W, 6.6V-19.5VDC
Size	50mm x 87mm

Introducing Xavier, the NVIDIA AI Supercomputer for the Future of Autonomous Transportation



The VOLTA GPU will deliver 20 TOPS (trillion operations per second) of performance @ 20 watts of power

Compliant with ISO 26262 functional safety specification

RAPID SOFTWARE DEVELOPMENT ON HETEROGENEOUS SYSTEMS

MORE C++ PARALLEL FOR LOOPS

GPU Lambdas Enable Custom Parallel Programming Models

Sandia National Laboratories

Kokkos

https://github.com/kokkos

```
Kokkos::parallel_for(N, KOKKOS_LAMBDA (int i) {
  y[i] = a * x[i] + y[i];
});
```

Lawrence Livermore National Laboratory

RAJA

https://e-reports-ext.llnl.gov/pdf/782261.pdf

```
RAJA::forall<cuda_exec>(0, N, [=] __device__ (int i) {
  y[i] = a * x[i] + y[i];
});
```

Hemi

Hemi

CUDA Portability Library

http://github.com/harrism/hemi

```
hemi::parallel_for(0, N, [=] HEMI_LAMBDA (int i) {
  y[i] = a * x[i] + y[i];
});
```

OPENACC

World's Only Performance Portable Programming Model for HPC

Add Simple Compiler Hint

```
main()
{
  <serial code>
  #pragma acc kernels
  {
    <parallel code>
  }
}
```

Simple

ARM

PEZY

POWER

Sunway

x86 CPU

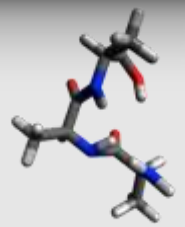
x86 Xeon Phi

NVIDIA GPU

Portable

LSDALTON

Simulation of molecular energies



Quicker Development

Lines of Code Modified

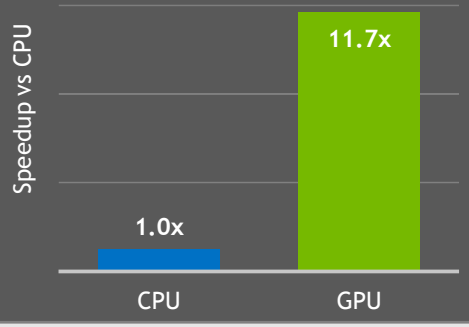
<100 Lines

# of Weeks Required

1 Week

Big Performance

CCSD(T) Module, Alanine-3
Titan System: AMD CPU vs Tesla K20X



Powerful

OpenACC for Multicore CPUs & GPUs

```
!$acc kernels loop
 do j = 1, m
   do i = 1, n
      a(j,i) = b(j,i)*alpha + c(i,j)*beta
   enddo
enddo
```

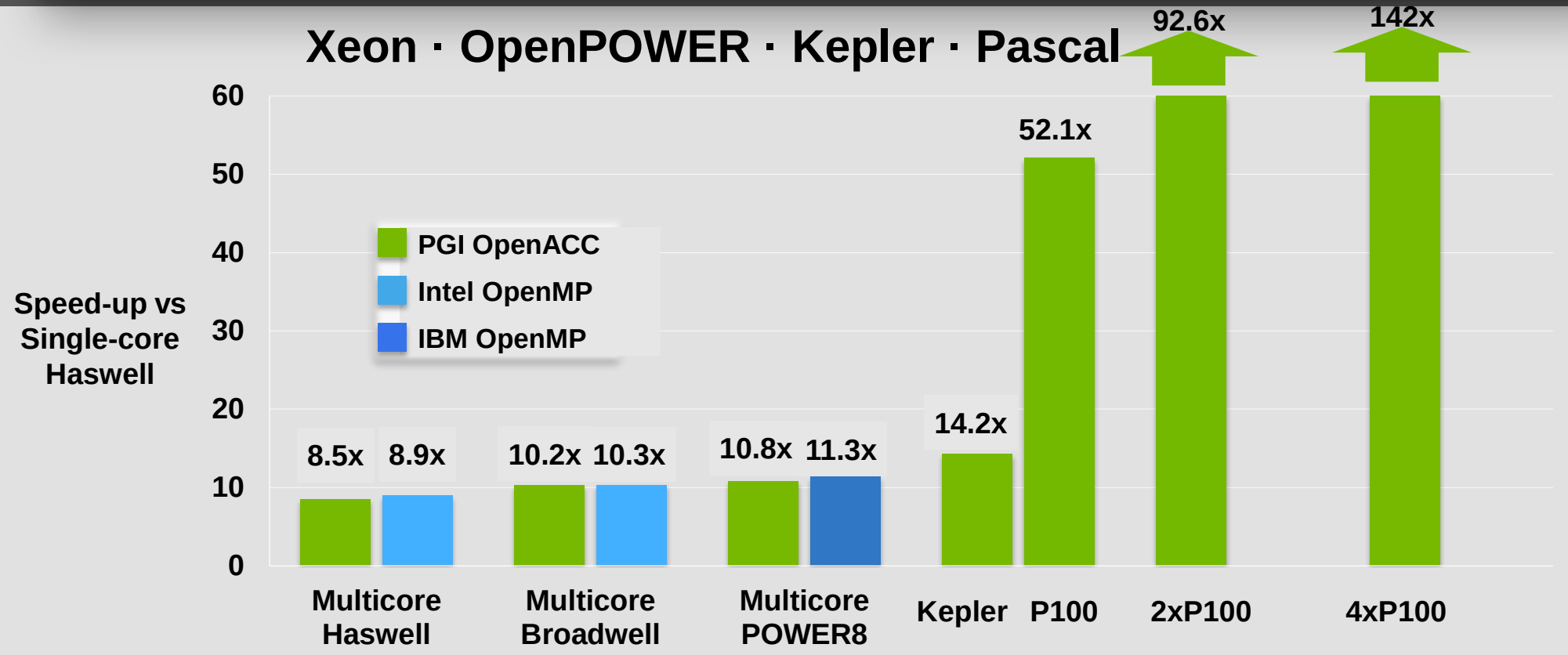
CPU

```
% pgfortran a.f90 -ta=multicore -c -Minfo
sub:
  10, Loop is parallelizable
      Generating Multicore code
      10, !$acc loop gang
  11, Loop is parallelizable
```

GPU

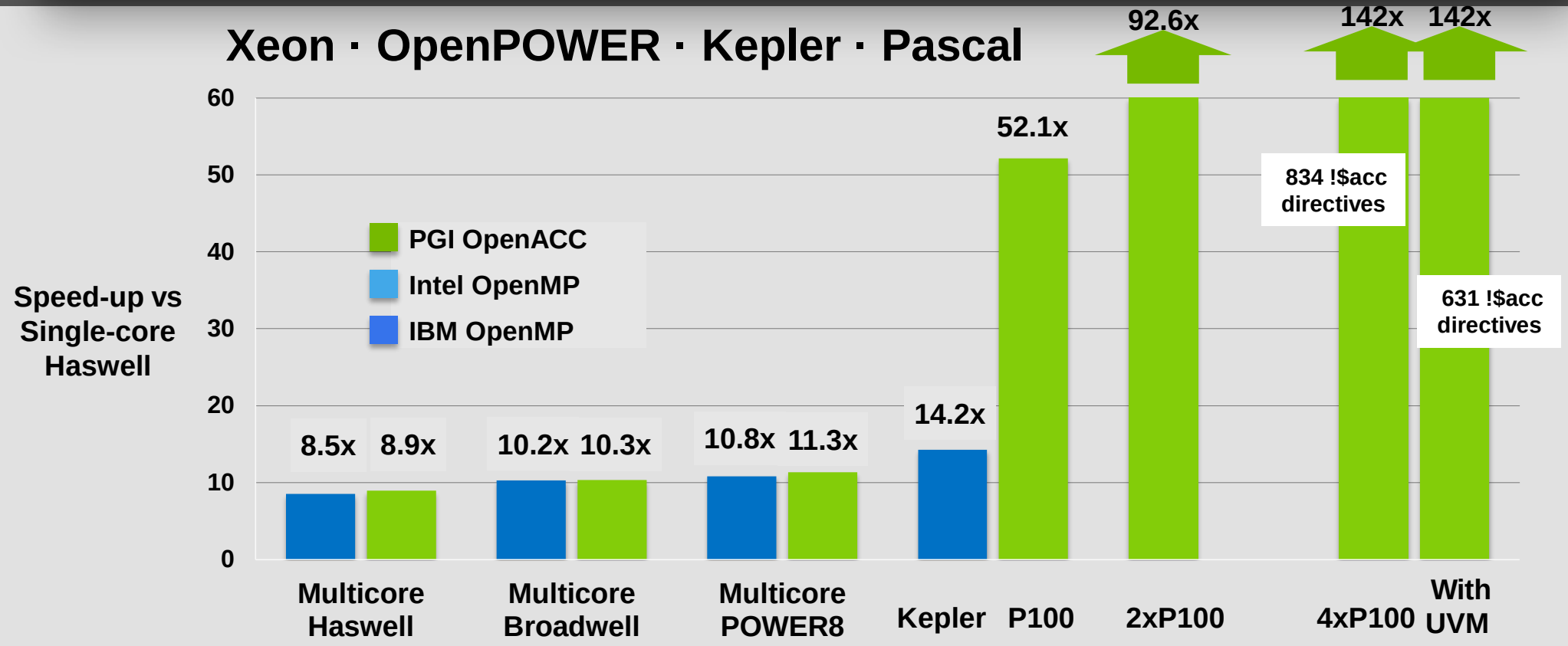
```
% pgfortran a.f90 -ta=tesla -c -Minfo
sub:
  10, Loop is parallelizable
  11, Loop is parallelizable
      Accelerator kernel generated
      Generating Tesla code
      10, !$acc loop gang, vector(4)
      11, !$acc loop gang, vector(32)
```

OPENACC PERFORMANCE PORTABILITY - CLOVERLEAF 1.3



AWE Hydrodynamics CloverLeaf mini-App, bm32 data set

OPENACC PERFORMANCE PORTABILITY - CLOVERLEAF 1.3



AWE Hydrodynamics CloverLeaf mini-App, bm32 data set

OPENACC FOR EVERYONE

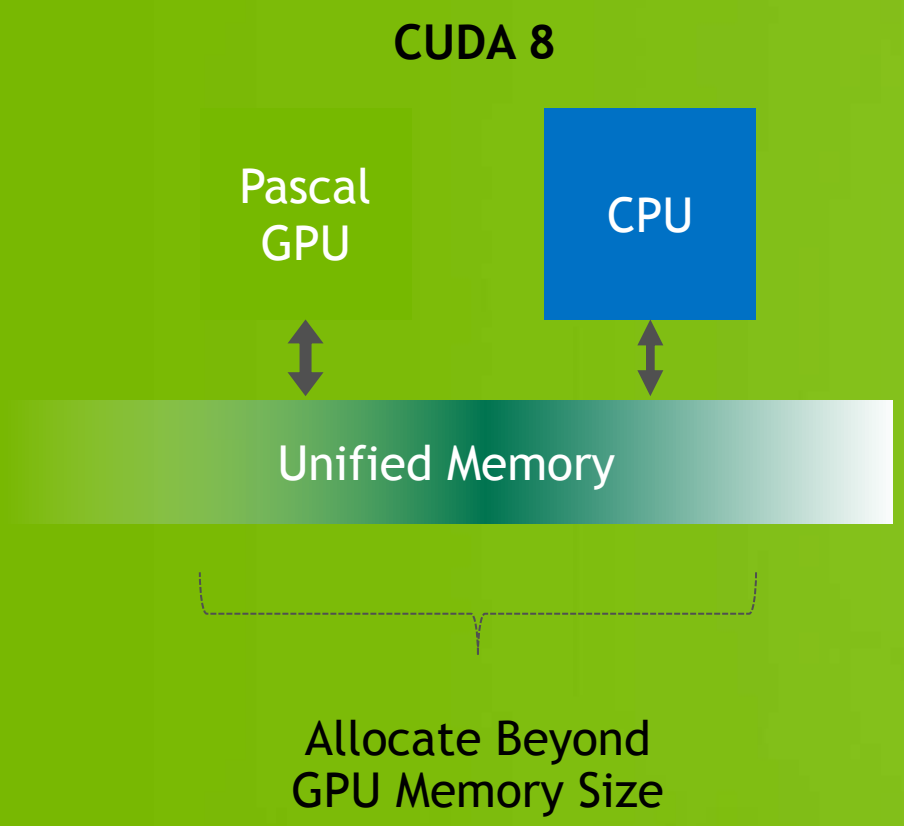
New PGI Community Edition Now Available

	FREE PGI Community Edition	PGI Professional Edition	PGI Enterprise Edition
PROGRAMMING MODELS OpenACC, CUDA Fortran, OpenMP, C/C++/Fortran Compilers and Tools	✓	✓	✓
PLATFORMS x86, OpenPOWER, NVIDIA GPU	✓	✓	✓
UPDATES	1-2 times a year	6-9 times a year	6-9 times a year
SUPPORT	User Forums	PGI Support	PGI Enterprise Services
LICENSE	Annual	Perpetual	Volume/Site

UNIFIED MEMORY

PASCAL UNIFIED MEMORY

Large datasets, simple programming, High Performance



Enable Large Data Models

- Oversubscribe GPU memory
- Allocate up to system memory size

Tune Unified Memory Performance

- Usage hints via cudaMemAdvise API
- Explicit prefetching API

Simpler Data Access

- CPU/GPU Data coherence
- Unified memory atomic operations

PAGE MIGRATION ENGINE

Support Virtual Memory Demand Paging

- 49-bit Virtual Addresses
  - Sufficient to cover 48-bit CPU address + all GPU memory
- GPU page faulting capability
  - Can handle thousands of simultaneous page faults
- Up to 2 MB page size
  - Better TLB coverage of GPU memory

CUDA 8 UNIFIED MEMORY — EXAMPLE

Allocating 4x more than P100 physical memory

```
void foo() {

  // Allocate 64 GB
  char *data;
  size_t size = 64*1024*1024*1024;
  cudaMallocManaged(&data, size);
}
```

64 GB unified memory allocation on P100 with 16 GB physical memory

Transparent – No API changes

Works on Pascal & future architectures

HOW UNIFIED MEMORY WORKS ON PASCAL

Servicing CPU *and* GPU Page Faults

GPU Code

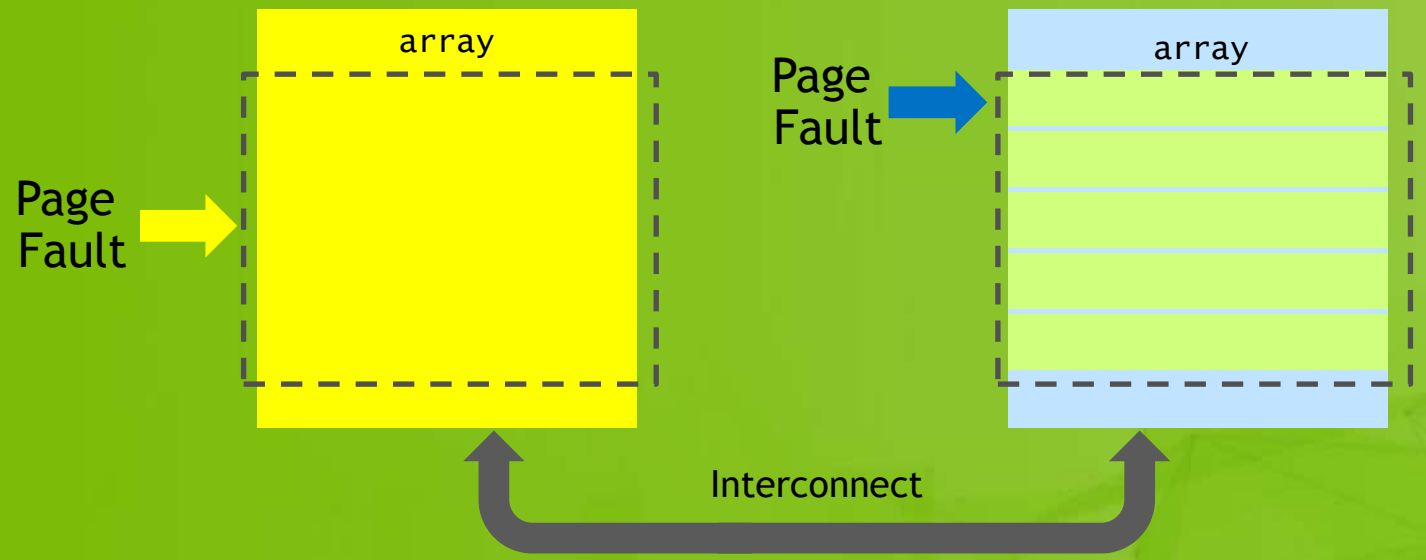
```
__global__
void setValue(char *ptr, int index, char val)
{
  ptr[index] = val;
}
```

CPU Code

```
cudaMallocManaged(&array, size);
memset(array, size);
setValue<<<...>>>(array, size/2, 5);
```

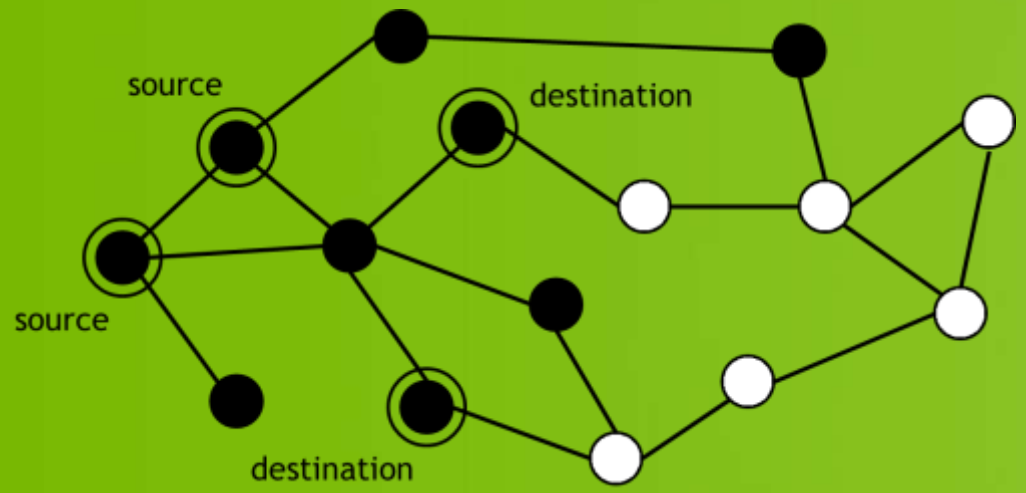
GPU Memory Mapping

CPU Memory Mapping

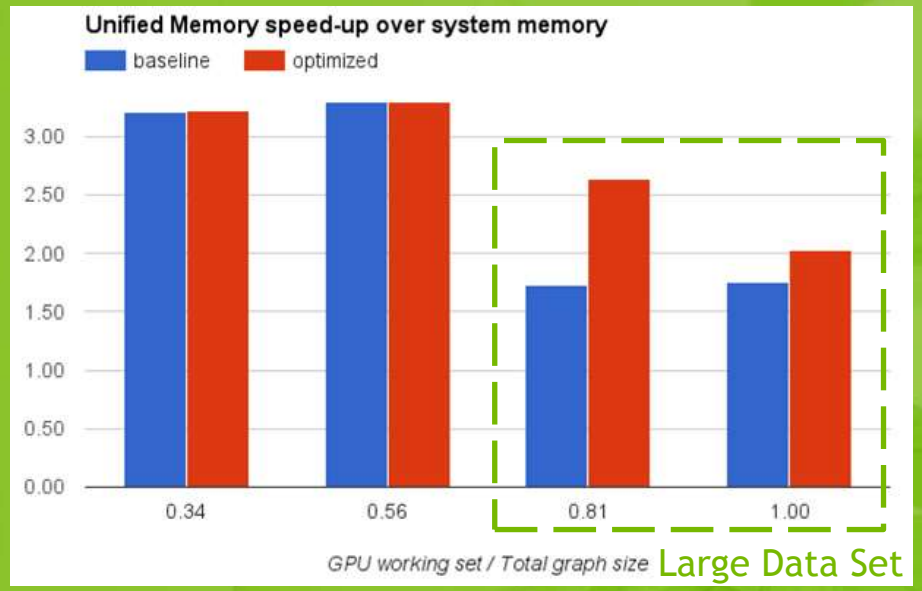


USE CASE: ON-DEMAND PAGING

Graph Algorithms

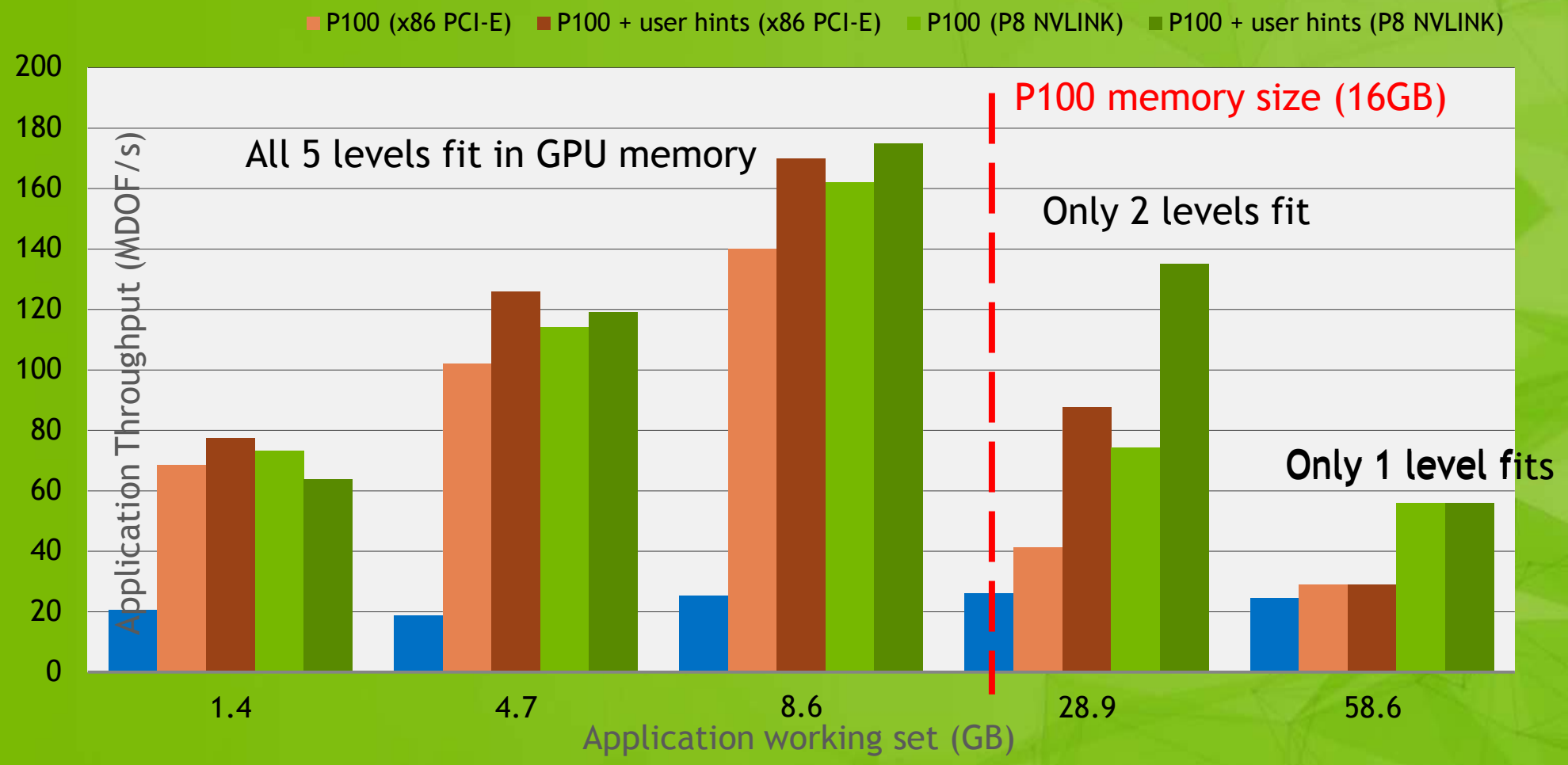


Performance over GPU directly accessing host memory (zero-copy)



Baseline: migrate on first touch
Optimized: best placement in memory

OUT-OF-CORE AMR COMPUTATIONS WITH UNIFIED MEMORY ON P100



x86 CPU: Intel E5-2630 v3, 2 sockets of 10 cores each with HT on (40 threads)

NVIDIA.

GREAT PERFORMANCE WITH UNIFIED MEMORY

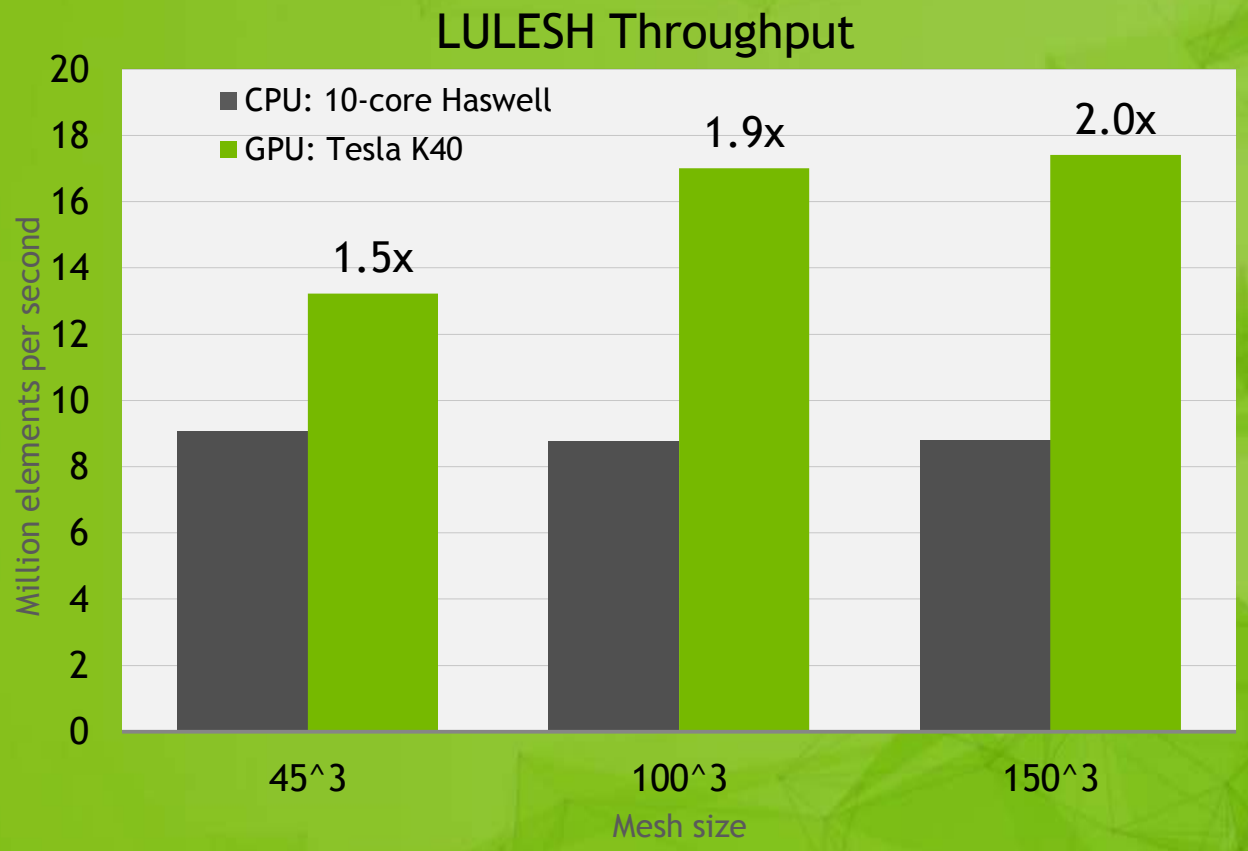
RAJA: Portable C++ Framework for parallel-for style programming

RAJA uses Unified Memory for heterogeneous array allocations

Parallel forall loops run on device

"Excellent performance considering this is a "generic" version of LULESH with no architecture-specific tuning."

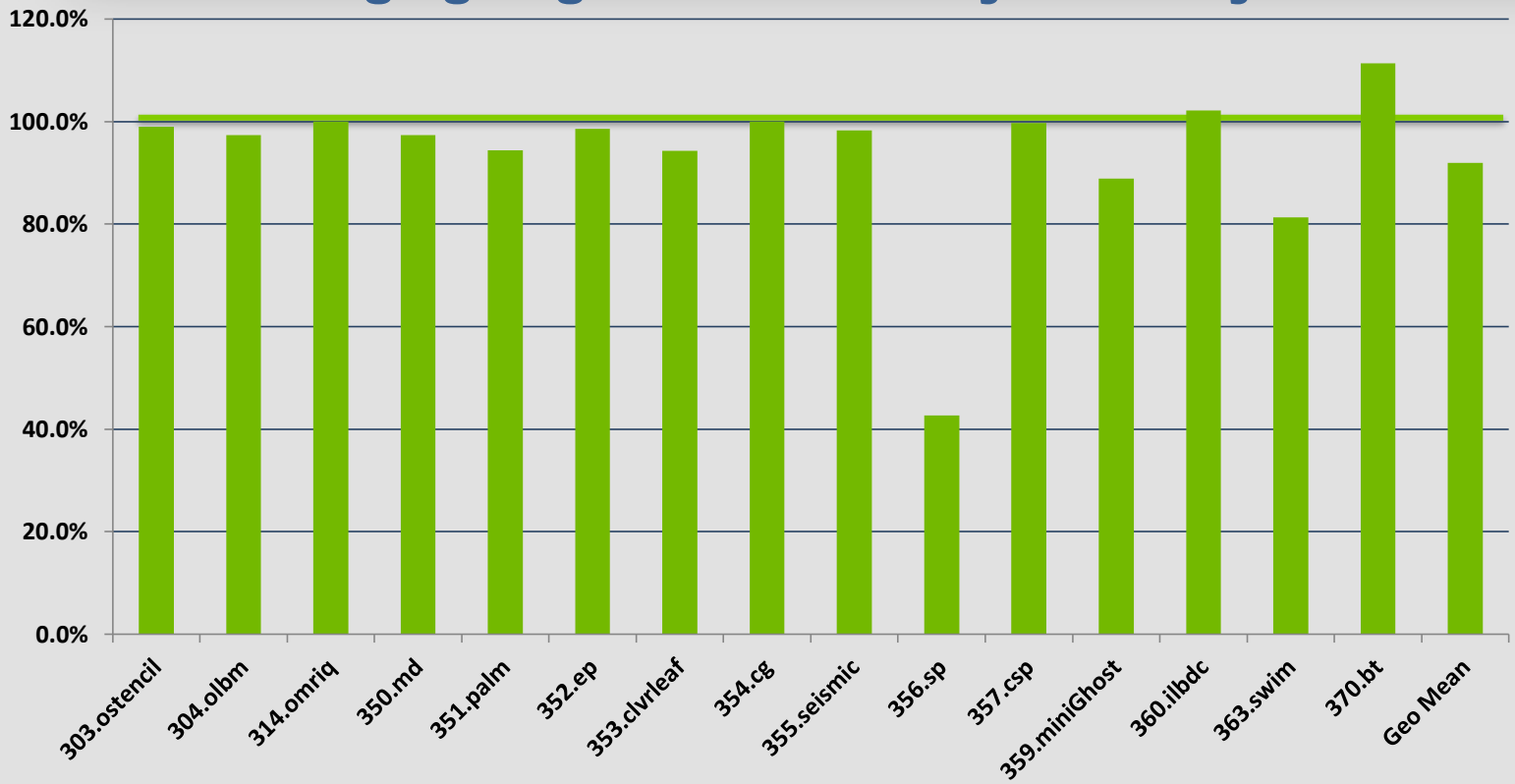
-Jeff Keasler, LLNL



GPU: NVIDIA Tesla K40, CPU: Intel Haswell E5-2650 v3 @ 2.30GHz, single socket 10-core

OpenACC and CUDA Unified Memory on Minsky

P100 Paging Engine Moves All Dynamically Allocated Data



100% = Directive-based Data Movement

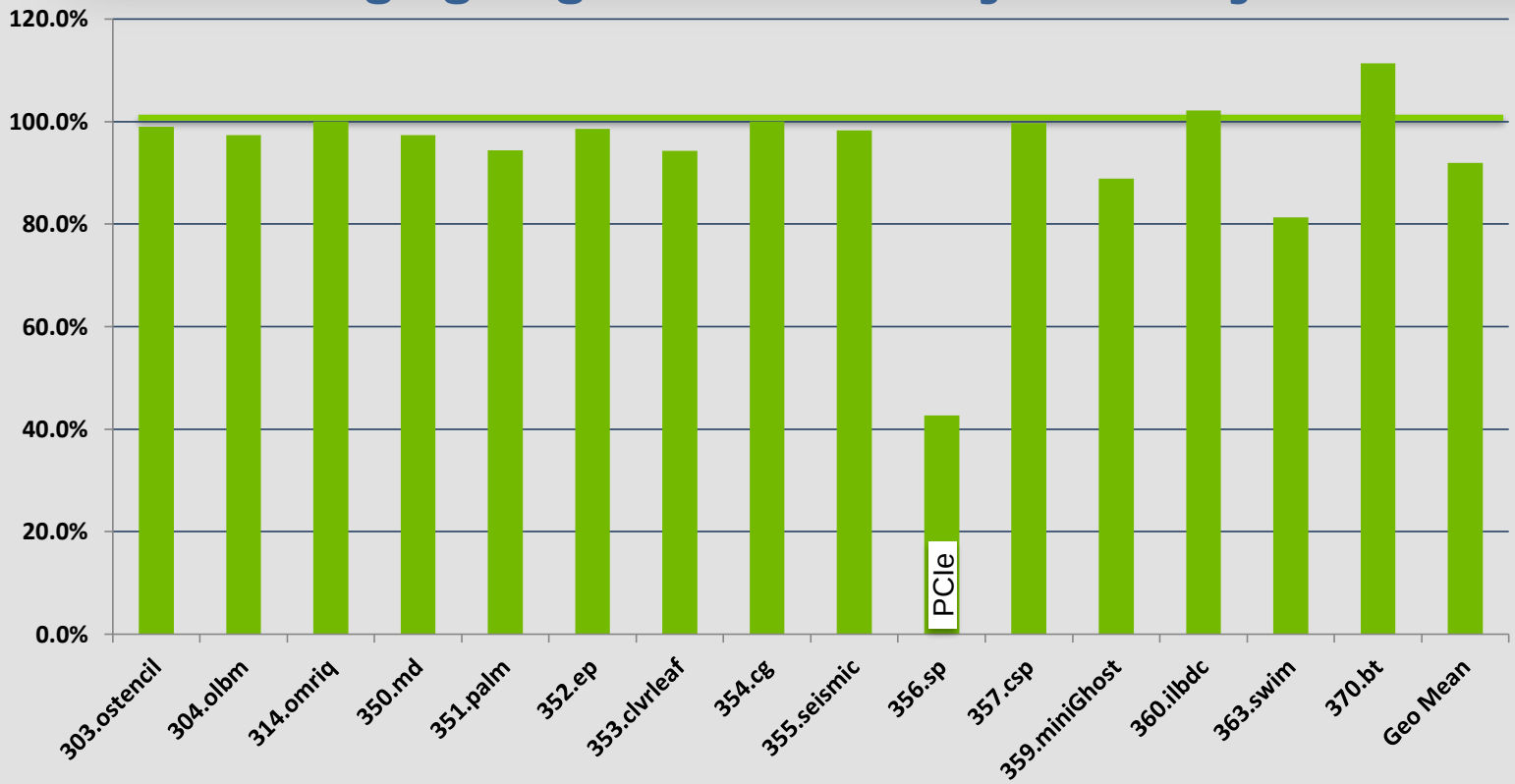
= OpenACC + CUDA Unified Memory

- Using CUDA 8.0 Unified memory with OpenACC via -ta=tesla:managed
- CUDA UM performance averages within 10% of programmer-directed OpenACC data movement
- 356.sp anomaly, Fortran automatics

PGI 16.10 Compilers OpenACC SPEC ACCEL™ 1.0 performance measured December, 2016. SPEC® and the benchmark name SPEC ACCEL™ are registered trademarks of the Standard Performance Evaluation Corporation.

OpenACC and CUDA Unified Memory on Minsky

P100 Paging Engine Moves All Dynamically Allocated Data



100% = Directive-based Data Movement

= OpenACC + CUDA Unified Memory

- Using CUDA 8.0 Unified memory with OpenACC via -ta=tesla:managed
- CUDA UM performance averages within 10% of programmer-directed OpenACC data movement
- 356.sp anomaly, Fortran automatics; *perf can be recovered w/pool allocator*

PGI 16.10 Compilers OpenACC SPEC ACCEL™ 1.0 performance measured December, 2016. SPEC® and the benchmark name SPEC ACCEL™ are registered trademarks of the Standard Performance Evaluation Corporation.

OPENACC AND CUDA UNIFIED MEMORY

INDEPENDENT CLAUSE

```
while ( err > tol && iter < iter_max ) {
  err=0.0;

#pragma acc kernels
{
  #pragma acc loop independent
  for( int j = 1; j < n-1; j++) {
    for(int i = 1; i < m-1; i++) {

      Anew[j*m+i] = 0.25 * (A[j*m+i+1] + A[j*m+i-1] +
                            A[(j-1)*m+i] + A[(j+1)*m+i]);

      err = max(err, abs(Anew[j*m+i] - A[j*m+i]));
    }
  }

  #pragma acc loop independent
  for( int j = 1; j < n-1; j++) {
    for( int i = 1; i < m-1; i++ ) {
      A[j*m+i] = Anew[j*m+i];
    }
  }
}

  iter++;
}
```

Tell compiler that it's safe to parallelize

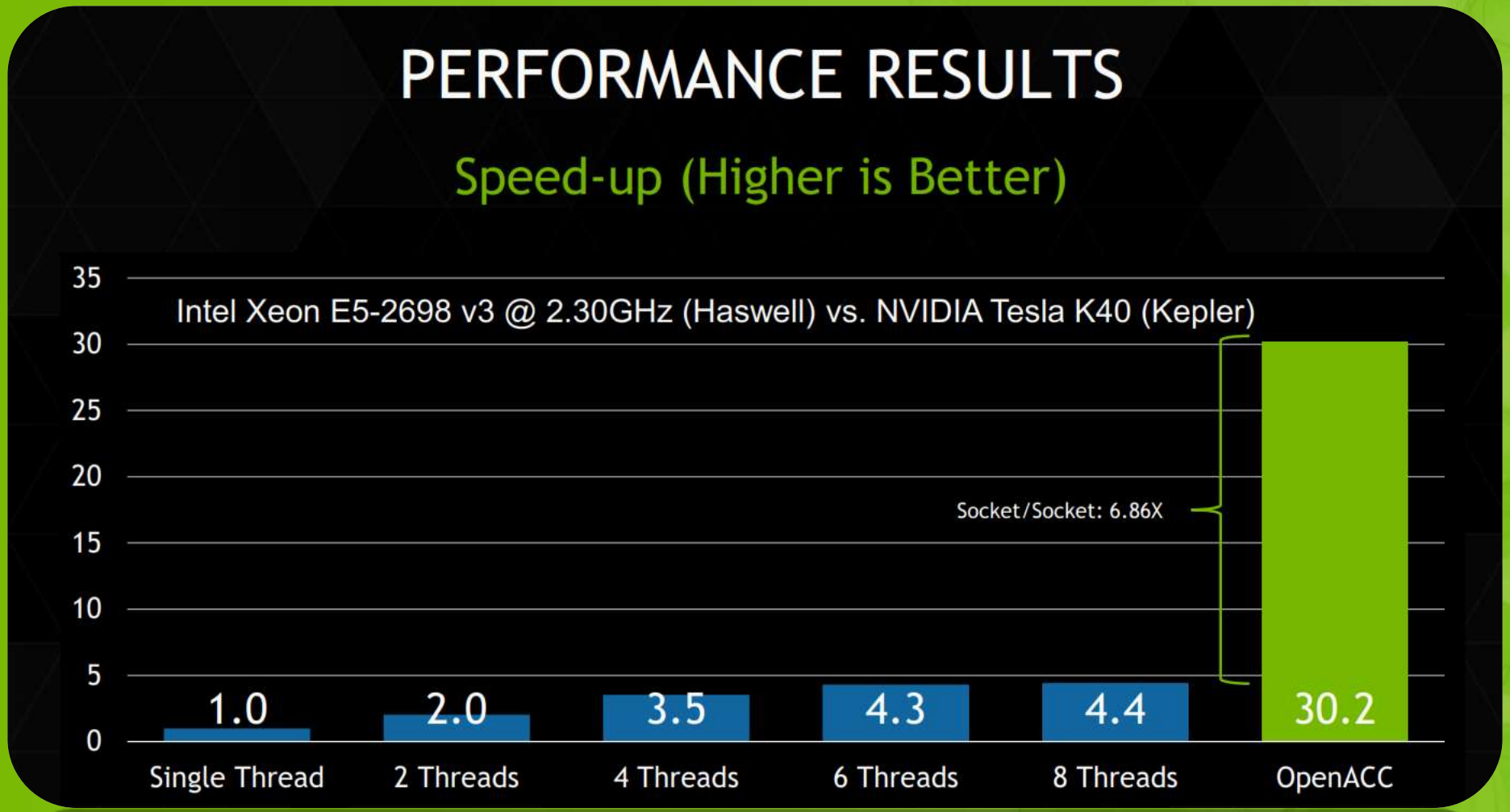
Tell compiler that it's safe to parallelize

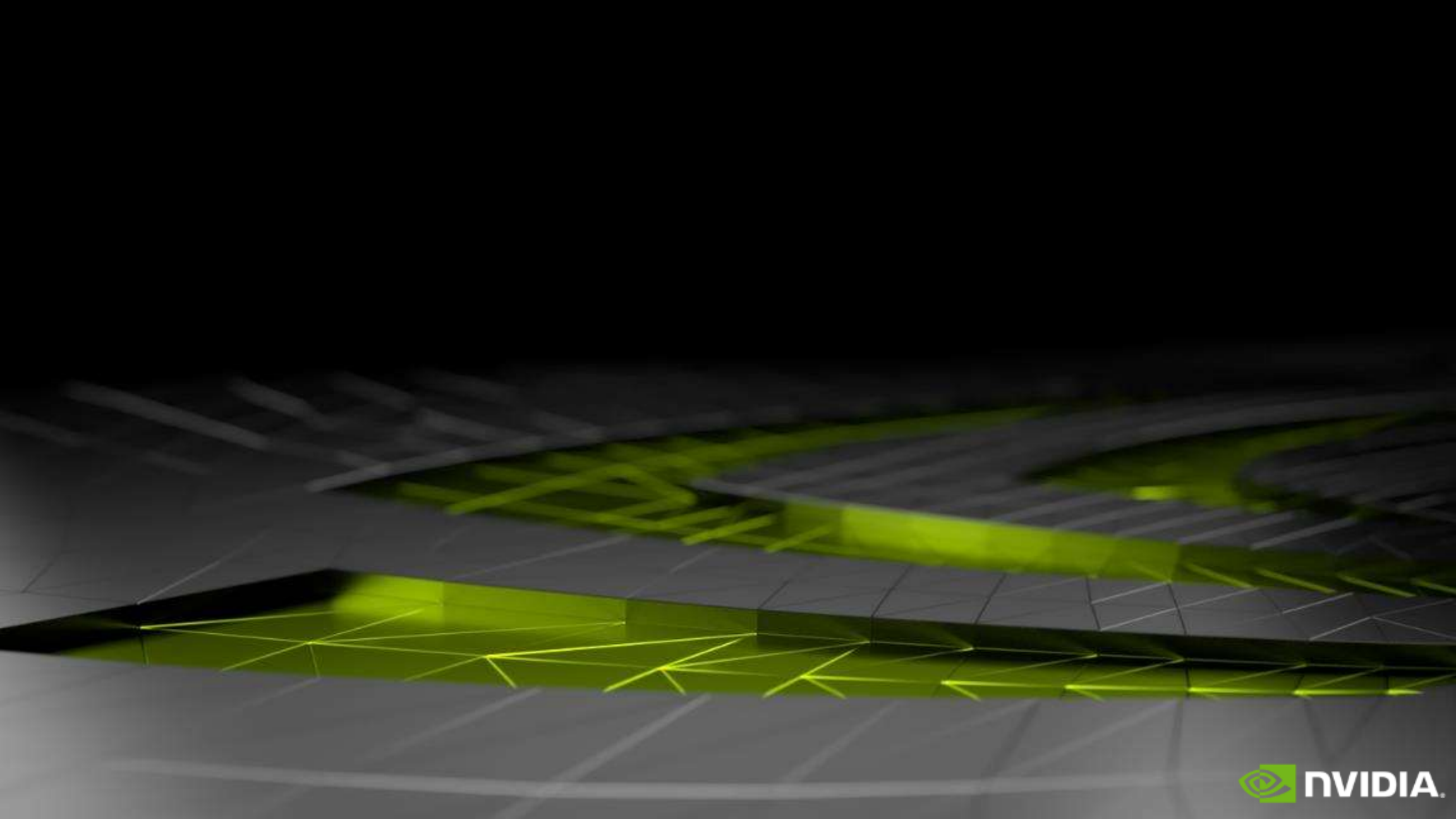
OPENACC AND CUDA UNIFIED MEMORY

BUILDING THE CODE

```
$ pgcc -fast -acc -ta=tesla:managed -Minfo=all laplace2d.c
main:
     83, Generating copyout(Anew[:])
         Generating copy(A[:])
     86, Loop is parallelizable
     87, Loop is parallelizable
         Accelerator kernel generated
         Generating Tesla code
         86, #pragma acc loop gang /* blockIdx.y */
         87, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
         92, Max reduction generated for error
     97, Loop is parallelizable
     98, Loop is parallelizable
         Accelerator kernel generated
         Generating Tesla code
         97, #pragma acc loop gang /* blockIdx.y */
         98, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
```

OPENACC AND CUDA UNIFIED MEMORY




NVIDIA.