Scientific Collaboration - GENCI, IBM, NVIDIA, Mellanox

Introduction To Ouessant OpenPOWER Platform

Stéphane CHAUVEAU	NVIDIA	schauveau@nvidia.com
François COURTEILLE	NVIDIA	fcourteille@nvidia.com
Pascal VEZOLLE	IBM	vezolle@fr.ibm.com
Nicolas TALLET	IBM	nicolas.tallet@fr.ibm.com

IBM Client Center

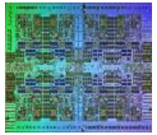
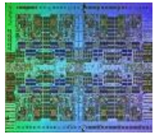
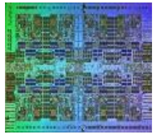
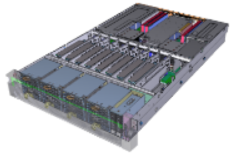
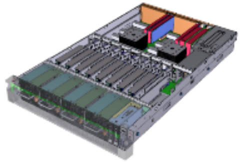
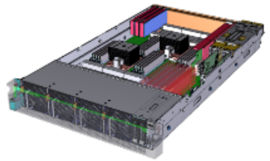
OpenPOWER Scientific Collaboration - GENCI & IBM/NVIDIA/Mellanox

- **Objectives of the Collaboration**
  - OpenPower Technology Assessment for Multi-Petaflops Deployment in 2017
    - Analyze applications affinity and requirements
    - Port applications to OpenPower platform
    - Analyze applications performance
    - Organize workshops and exchanges with application developers
  - Experiment OpenPower w/GPU Prototype Systems in 2015/2016
  - Evaluate Programming Models (OpenMP, OpenACC)
  - Provide Feedback To Development Teams

OpenPOWER Technical Architecture

Hardware Features

IBM OpenPOWER Accelerated Computing Roadmap

Mellanox Interconnect Technology	ConnectX-4 EDR Infiniband PCIe Gen3	ConnectX-4 EDR Infiniband CAPI over PCIe Gen3	ConnectX-5 HDR Infiniband Enhanced CAPI over PCIe Gen4
NVIDIA GPUs	Kepler PCIe Gen3	Pascal NVLink SXM2	Volta Enhanced NVLink SXM2
IBM CPUs	POWER8 	POWER8 w/NVLink  NVLink	POWER9  Enhanced NVLink
	2015	2016	2017
Server	 **Firestone**	 **Minsky**	 **Witherspoon**

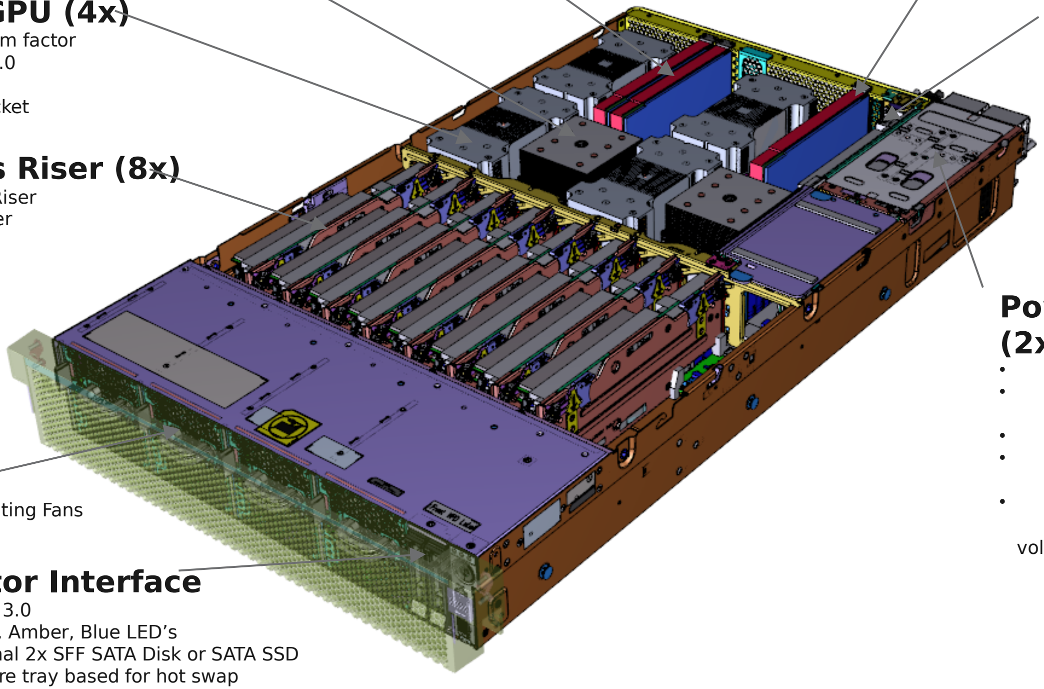
IBM Power Systems S822LC ‘Minsky’ w/NVIDIA Tesla P100

- **2 POWER8 CPUs**
- **Up to 1TB DDR4 memory**
- **Up to 4 Tesla P100 GPUs**

- **1st Server with POWER8 with NVLink Technology**

- **Only architecture with CPU:GPU NVlink**

IBM Power System S822LC 'Minsky' (8335-GTB)



P8 with NVLink Processor (2x)
- 190W Sort

PCIe slot (2x)
- Gen3 PCIe
- HHHL Adapter

PCIe slot (1x)
- Gen3 PCIe
- HHHL Adapter

NVidia GPU (4x)
- SXM2 form factor
- NVLink 1.0
- 300 W
- 2 per socket

Service Controller Card
- Firestone BMC Content
- Daughter card due to planar space constraints

Memory DIMM's Riser (8x)
- 4 IS DDR4 DIMMs per Riser
- Single Centaur per Riser
- 32 IS DIMM's total

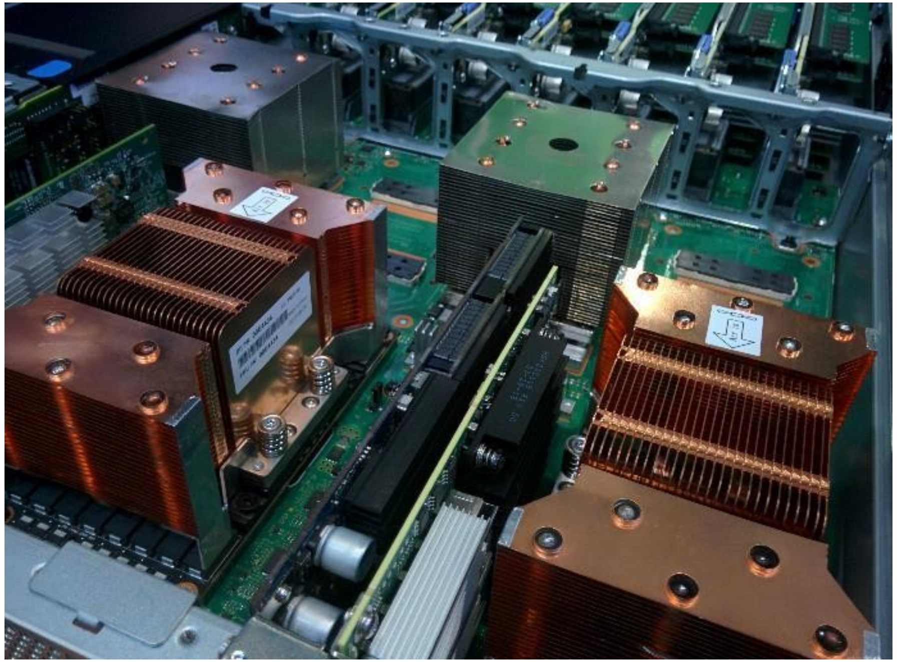
Power Supplies (2x)
- 1300W
- Configuration limits for redundancy
- Hot Swap
- Common Form Factor Supply
- Evaluating 1900W, multi-input voltage power supply

Cooling Fans
- 80mm Counter- Rotating Fans
- Hot swap

Operator Interface
- 1 USB 3.0
- Green, Amber, Blue LED's
- Optional 2x SFF SATA Disk or SATA SSD
- Disk are tray based for hot swap

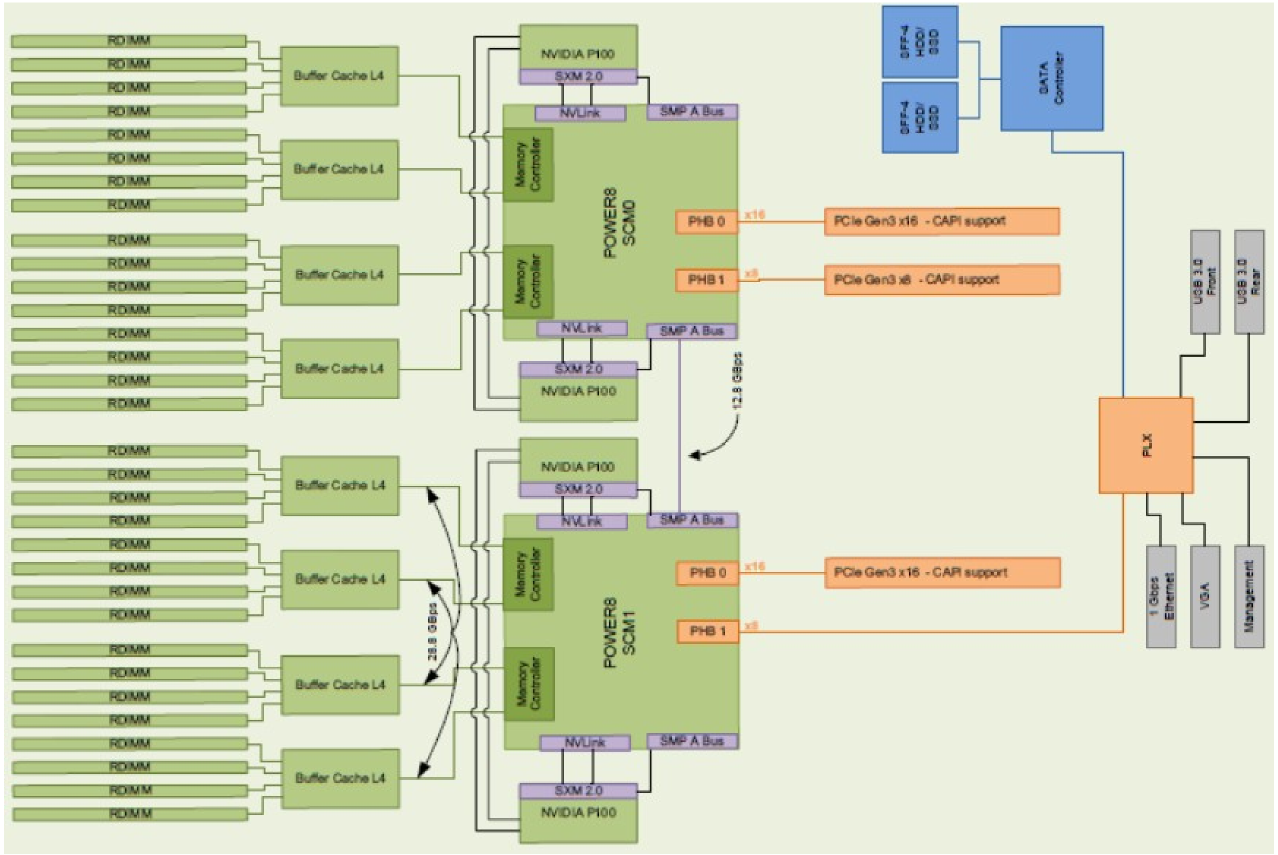
IBM Power System S822LC 'Minsky' (8335-GTB)

Sockets	2 x POWER8
Physical Cores	20
Hardware Threads (Logical Cores)	20 [SMT Off] 40 [SMT 2] 80 [SMT 4] 160 [SMT 8]
CPU Frequency	2.86 GHz [Nominal] 4.03 GHz [Turbo]
Memory Capacity	Up To 1 TB
Memory Bandwidth (Peak)	230 GB/s
DP Performance (Peak)	468 Gflops
GPUs	Up To 4 x NVIDIA Tesla P100
Link	NVLink
Link Bandwidth	20 GB/s
DP Performance (Peak)	**4.9 TFlops**



Logical System Diagram

- Two Single-Chip Modules (SCM)
- 4 Memory Riser Cards per SCM
  - Buffer Chips for L4 Cache
  - 4 RDIMM Slots
- Max. Capacity: 32 Memory DIMMs (1024 GB)
- 4 GPU Sockets (300 W Max.)
- 3 PCIe Gen3 Slots
- Dedicated PCI Bus:
  - Integrated SATA Controller: Up to 2 SATA Drives
  - Integrated Ethernet
  - Integrated USB Port

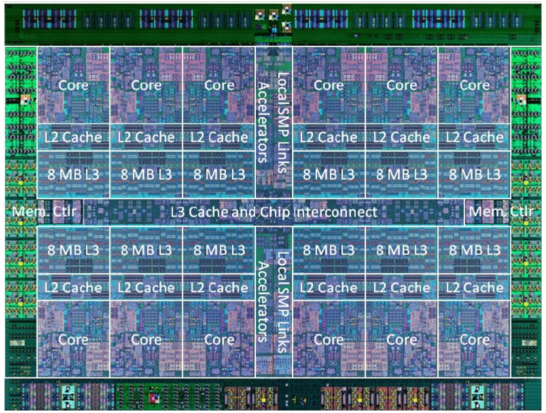


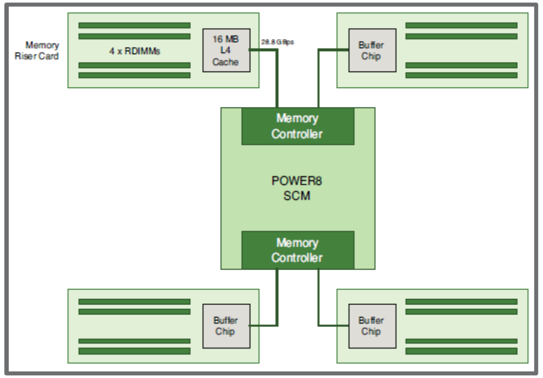
Simultaneous Multi-Threading (SMT)

- Allows a single physical processor core to dispatch simultaneously instructions from more than one hardware thread context
  - With SMT, each POWER8 core can present up to 8 hardware threads
  - Because there are multiple hardware threads per physical processor core, additional instructions can run at the same time
- SMT benefit highly depends on the workload
- Changing SMT mode does not require reboot

Mode	Logical Cores
Single Thread	20
SMT2	40
SMT4	80
SMT8	160

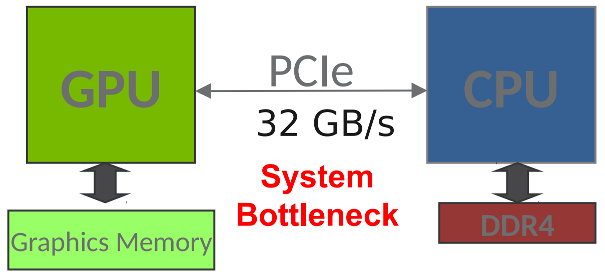
Memory Sub-System | L3 & L4 Cache

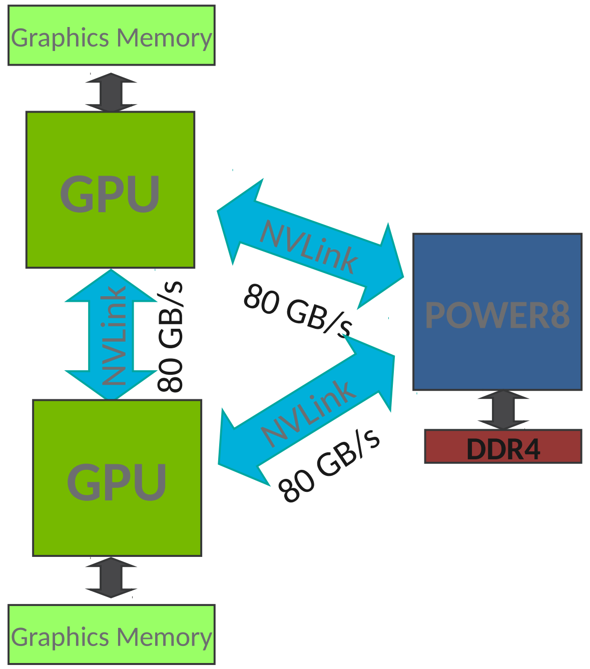




- L3 Cache
  - eDRAM on the processor die
  - Each processor core is associated with a fast 8-MB local region of L3 cache (FLR-L3)
    - But can also access other L3 cache regions as shared L3 cache
- L4 Cache
  - Memory buffer on the memory riser cards
  - Each memory buffer contains 16 MB of L4 cache
    - Up to 128 MB of L4 cache by using all memory riser cards

POWER8 w/NVLink: 2.5x Faster CPU-To-GPU Connection

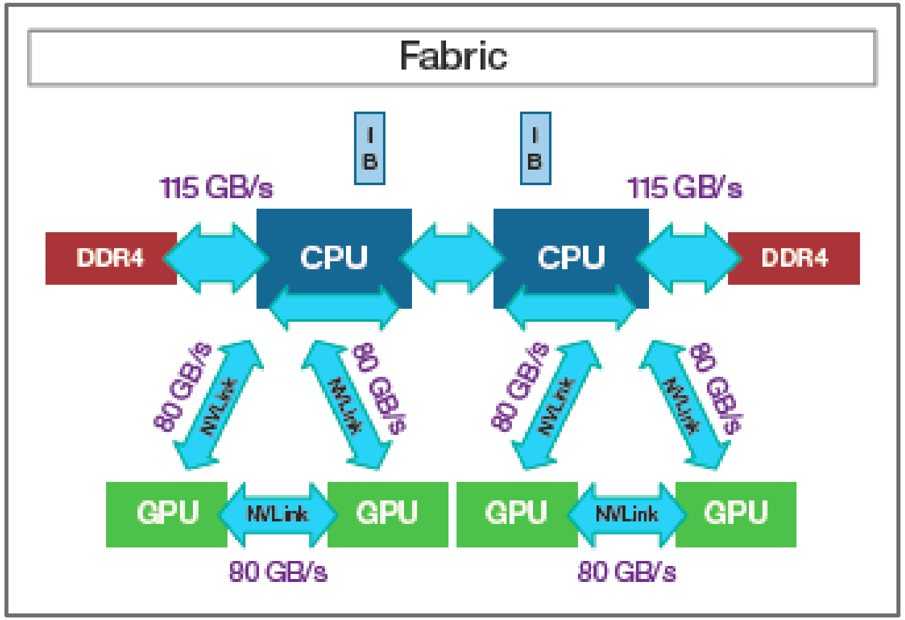


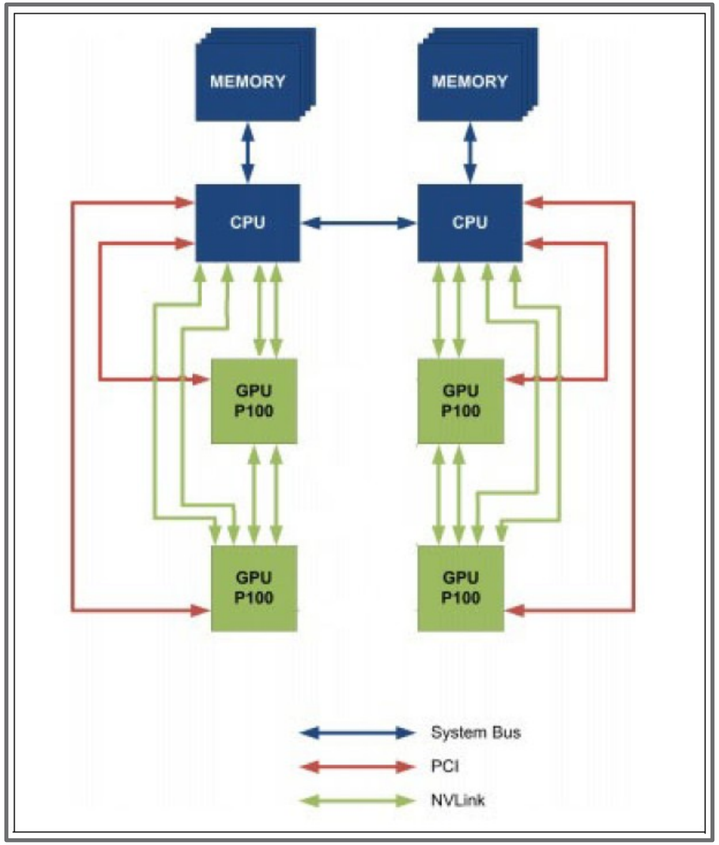


GPUs Limited by PCIe Bandwidth From CPU-System Memory

NVLink Enables Fast Unified Memory Access between CPU & GPU Memories

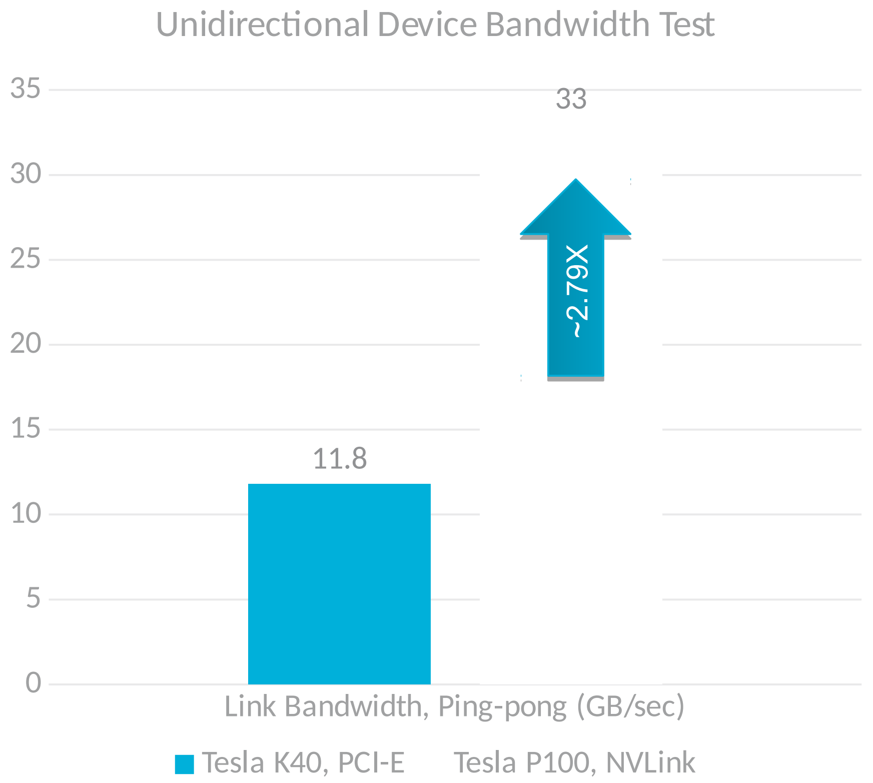
Comprehensive Fabric Architecture





Unique Bandwidth Dividend: NVLink from CPU:GPU

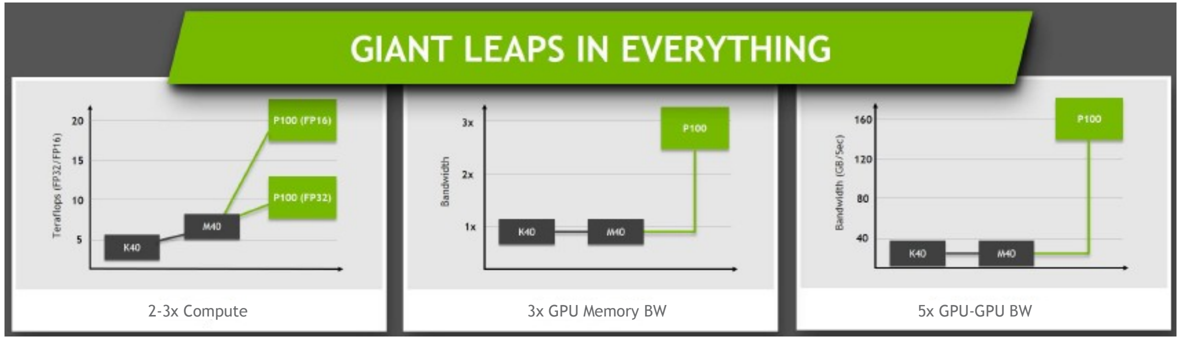
- **POWER8 with NVLink: the only processor with NVLink from CPU to GPU**
  - Delivering in excess of 2.5x bandwidth in testing ***today***
  - NVLink bus delivering higher efficiency than PCI-E links (82.5% vs 74% of peak)
  - No code changes to start leveraging bus (CUDA 8.0 and go)
- **Platform for developers needing Bandwidth for the foreseeable future**
  - POWER8 with NVLink ships in 2016
  - Xeon E5-2600 Series CPUs retain PCI-E x16 3.0 connectivity through 2017

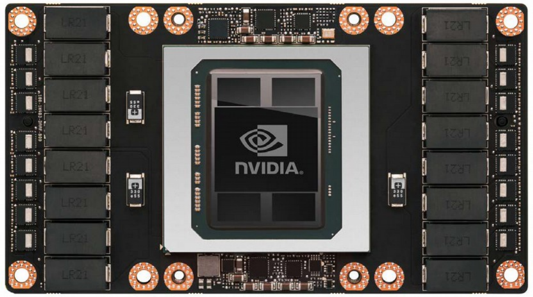


Typical Ping-pong PCI-E device bandwidth: ~74% of theoretical 16GB unidirectional max

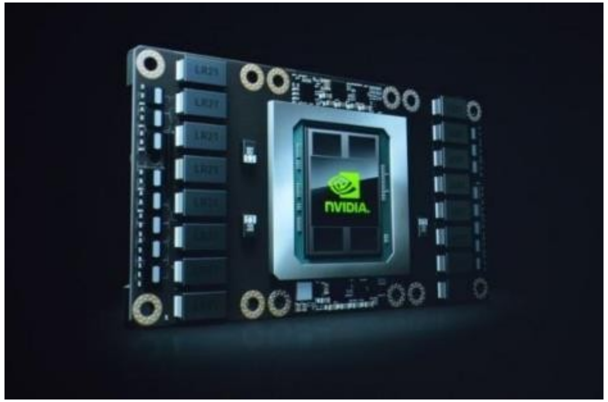
Accelerated Performance: NVIDIA Tesla P100 Pascal GPUs

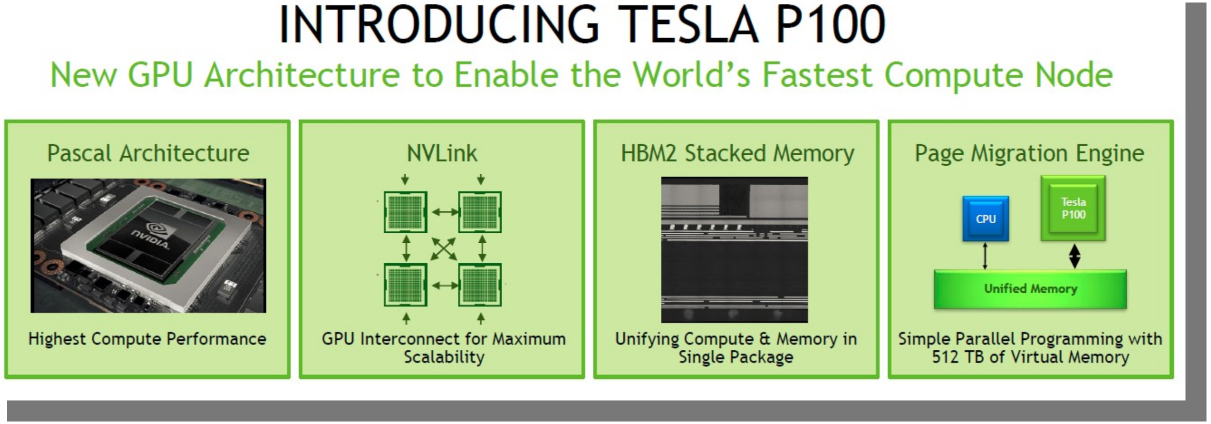
	Tesla P100	Tesla K80	Tesla K40
DP TFLOPS	5.3 TFLOPS	2.91 TFLOPS (Max Boost)	1.4 TFLOPS
SP TFLOPS	10.6 TFLOPS (21.2 TFLOPS Half Precision)	8.74 TFLOPS (Max Boost)	4.3 TFLOPS
Memory Bandwidth	720 GB/sec	480 GB/sec (2x 240GB)	288 GB/sec
Memory Capacity	16 GB	24 GB (2x 12GB)	**12 GB**





NVIDIA Tesla P100 Architecture Details





INTRODUCING TESLA P100

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

Pascal Architecture	NVLink	HBM2 Stacked Memory	Page Migration Engine
Highest Compute Performance	GPU Interconnect for Maximum Scalability	Unifying Compute & Memory in Single Package	Simple Parallel Programming with 512 TB of Virtual Memory

CPU
Tesla P100
Unified Memory

HPC Software Stack

Software Components

Comprehensive HPC Software Stack

Operating System

- Red Hat Enterprise Linux 7.3 LE

Application Development

- Advance Toolchain 10.0 [Including GCC 6.2]
- CUDA Toolkit 8.0
- GCC 4.8
- IBM XL C/C++ 13.1.5
- IBM XL Fortran 15.1.5
- LLVM Clang & XLFlang [Beta]
- PGI Accelerator 16.10

MPI Libraries

- IBM XL C/C++, IBM XL Fortran:
  - IBM Parallel Environment RTE 2.3 [Phase Out]
  - IBM Spectrum MPI 10.1 [New]
- PGI Accelerator:
  - Open MPI 1.10

Scientific Libraries

- IBM ESSL 5.5
- IBM Parallel ESSL 5.3
- IBM XL MASS

Performance Analysis

- IBM Parallel Performance Toolkit for POWER 2.3

Workload Management

- IBM Spectrum LSF 10.1

Data Management

- IBM Spectrum Scale 4.1

Network Management

- Mellanox Unified Fabric Manager (UFM)

Engineering and Scientific Subroutine Library (ESSL)

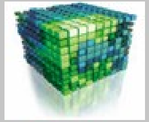
ESSL

- Serial & SMP highly-tuned mathematical subroutines
- 30+ SMP CUDA BLAS3 subroutines
  - POWER8 SMP / GPU / Hybrid (CPU+GPU) support
  - Leverage ESSL BLAS + NVIDIA cuBLAS
  - Support multiple GPUs
  - Support problem sizes larger than GPU Memory

Parallel ESSL

- 150+ SPMD highly-tuned mathematical subroutines
  - L2/L3 PBLAS, Linear Algebric Equations, Fourier Transforms…

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
- PGI/NVIDIA Compiler
- CUDA C/C++ for Power via XL NVCC

OpenACC Directives for Accelerators

Key Features:

- Designed to simplify Programing of heterogeneous CPU/GPU systems
- Directive based parallelization for accelerator device
- PGI/NVIDIA Compiler
- OpenACC/gcc

OpenMP

Key Features:

- OpenMP 4.0 introduces offloading and support for heterogeneous CPU/GPU
- Leverage existing OpenMP high level directives support
- IBM XL Compiler
- Open Source LLVM OpenMP Compiler

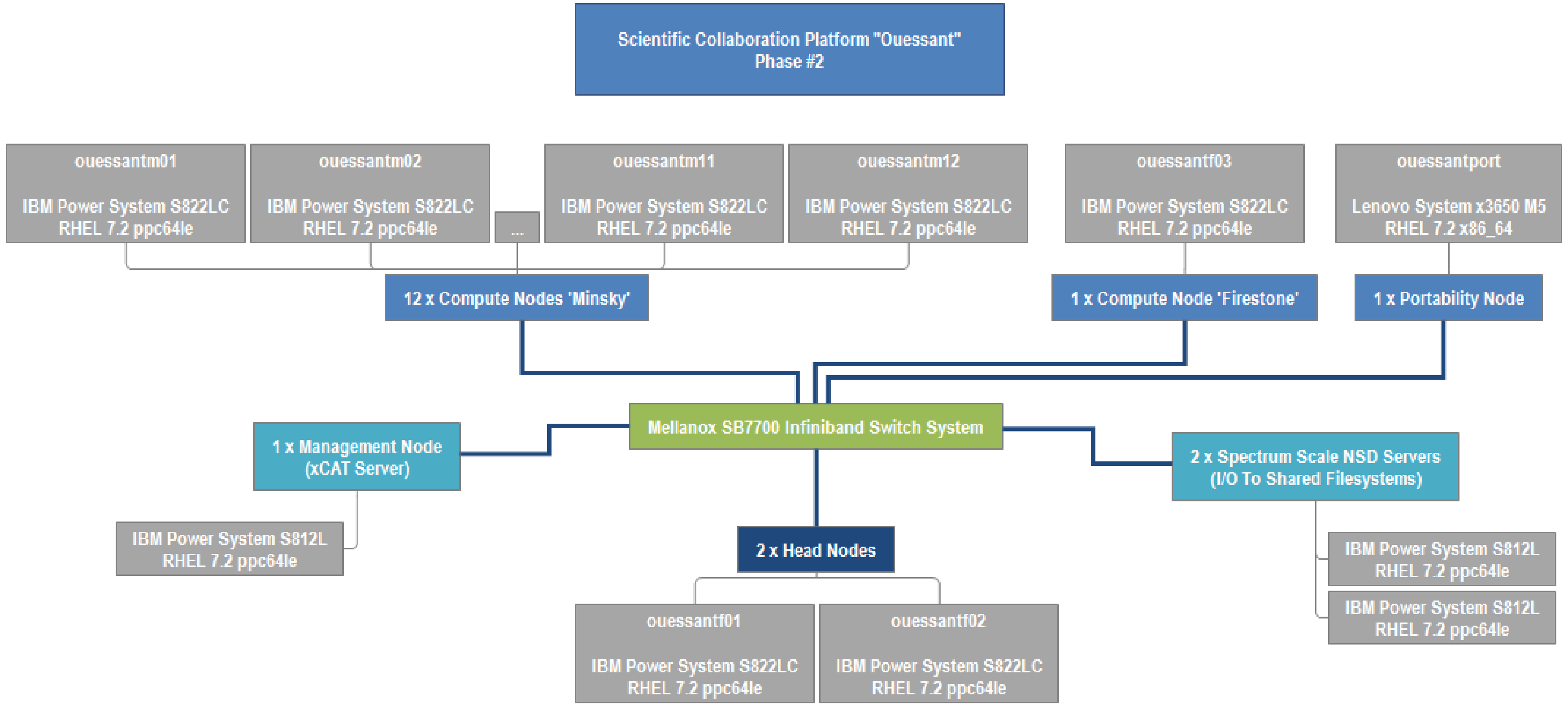
Supported GPU Offloading Features by Compiler Family (1Q 2017)

Compiler	OpenACC	OpenMP
GCC 4	-	4.0 w/o Offload
GCC 5	Experimental 2.0a No Offload on ppc64le	4.0 w/o Offload
GCC 6	Partial 2.0a	Partial 4.0
IBM XL	-	4.0+4.5
LLVM	-	4.0+4.5
PGI	Full 2.0a+2.5	-

Ouessant Platform

Scientific Collaboration Prototype

Platform Architecture


Scientific Collaboration Platform "Ouessant" Phase #2
ouessantm01 IBM Power System S822LC RHEL 7.2 ppc64le
ouessantm02 IBM Power System S822LC RHEL 7.2 ppc64le
...
ouessantm11 IBM Power System S822LC RHEL 7.2 ppc64le
ouessantm12 IBM Power System S822LC RHEL 7.2 ppc64le
ouessantf03 IBM Power System S822LC RHEL 7.2 ppc64le
ouessantport Lenovo System x3650 M5 RHEL 7.2 x86_64
12 x Compute Nodes 'Minsky'
1 x Compute Node 'Firestone'
1 x Portability Node
Mellanox SB7700 Infiniband Switch System
1 x Management Node (xCAT Server)
IBM Power System S812L RHEL 7.2 ppc64le
2 x Spectrum Scale NSD Servers (I/O To Shared Filesystems)
IBM Power System S812L RHEL 7.2 ppc64le
IBM Power System S812L RHEL 7.2 ppc64le
2 x Head Nodes
ouessantf01 IBM Power System S822LC RHEL 7.2 ppc64le
ouessantf02 IBM Power System S822LC RHEL 7.2 ppc64le

System Topology: CPU

SMT=0 [Off]

Socket #0										Socket #1									
0	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152

SMT=2

Socket #0										Socket #1									
0	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152
1	9	17	25	33	41	49	57	65	73	81	89	97	105	113	121	129	137	145	153

SMT=4

Socket #0										Socket #1									
0	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152
1	9	17	25	33	41	49	57	65	73	81	89	97	105	113	121	129	137	145	153
2	10	18	26	34	42	50	58	66	74	82	90	98	106	114	122	130	138	146	154
3	11	19	27	35	43	51	59	67	75	83	91	99	107	115	123	131	139	147	155

SMT=8

Socket #0										Socket #1									
0	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152
1	9	17	25	33	41	49	57	65	73	81	89	97	105	113	121	129	137	145	153
2	10	18	26	34	42	50	58	66	74	82	90	98	106	114	122	130	138	146	154
3	11	19	27	35	43	51	59	67	75	83	91	99	107	115	123	131	139	147	155
4	12	20	28	36	44	49	60	68	76	84	92	100	108	116	124	132	140	148	156
5	13	21	29	37	45	50	61	69	77	85	93	101	109	117	125	133	141	149	157
6	14	22	30	38	46	51	62	70	78	86	94	102	110	118	126	134	142	150	158
7	15	23	31	39	47	52	63	71	79	87	95	103	111	119	127	135	143	151	159

System Topology: GPU

- **2 GPU Devices / Socket**
  - Socket #0
    - GPU0
    - GPU1
  - Socket #1
    - GPU2
    - GPU3
- **“Same Socket” Data Exchanges**
  - Avoid going through SMP link between POWER8 CPUs

```
[user@host ~]$ nvidia-smi topo --matrix
        GPU0    GPU1    GPU2    GPU3    mlx5_0  mlx5_1  CPU Affinity
GPU0     X      NV2     SOC     SOC     SOC     SOC     0-79
GPU1    NV2      X      SOC     SOC     SOC     SOC     0-79
GPU2    SOC     SOC      X      NV2     SOC     SOC     80-159
GPU3    SOC     SOC     NV2      X      SOC     SOC     80-159
mlx5_0  SOC     SOC     SOC     SOC      X      PIX
mlx5_1  SOC     SOC     SOC     SOC     PIX      X

Legend:

  X   = Self
  SOC  = Connection traversing PCIe as well as the SMP link between CPU
sockets(e.g. QPI)
  PHB  = Connection traversing PCIe as well as a PCIe Host Bridge
(typically the CPU)
  PXB  = Connection traversing multiple PCIe switches (without traversing
the PCIe Host Bridge)
  PIX  = Connection traversing a single PCIe switch
  NV#  = Connection traversing a bonded set of # NVLinks
```

Job Submission:
IBM Spectrum LSF

Typical Submission Script

```
#!/bin/bash

#BSUB –a p8aff(1,1,1,pack)
#BSUB -cwd ~/helloworld
#BSUB -e ~/helloworld/stderr
#BSUB -J HelloWorld
#BSUB -n 2
#BSUB -o ~/helloworld/stdout
#BSUB -q compute
#BSUB -R "span[ptile=2]"
#BSUB -W 00:05

mpirun ~/helloworld/helloworld.exe
```

Directives

Option	Value	Purpose
-cwd	<Path>	Execution Directory
-e	<File>	stderr File
-J	<Job Name>	Job Name
-n	<# MPI Tasks>	Total Number of MPI Tasks
-o	<File>	stdout File
-q	<Queue>	Target Queue
-R	“span[ptile=<ppn >]”	Resource Specification: Number of Tasks per Node
-W	HH:MM	Run Limit
-x		**Travail exclusif**

User Commands

Command	Argument	Purpose
bjobs		List active jobs (waiting, in progress)
	-u { <User ID> | all }	Restrict to specified user
	-X	List associated resources
bkill	<Job ID>	Kill job
bpeek	<Job ID>	Display job stdout/stderr
	-f	Refresh display in real time
bqueues		List existing queues
bstatus	<Job ID>	Display job status
bsub	< <Submission Script>	**Submit job into queue**

Execution Environment

Setting	Purpose
Task Placement	Set target compute node for each MPI task (among all allocated hosts)
Processor Affinity	Set list of allowed CPUs for each MPI task
GPU Resource Requirement	Express GPU resources requirement for job submission
GPU Affinity	Set list of allowed GPUs for each MPI task

Task Placement

- **Default Policy: “Group Round Robin”**
  - ‘ptile’ MPI tasks per allocated compute node
  - One allocated compute node after the other until all MPI tasks have been placed
- **Alternative Policy**
  - Specified through environment variable ‘LSB_TASK_GEOMETRY’
  - Example:
    - export LSB_TASK_GEOMETRY="{(0,3)(1,4)(2,5)}«
      - Tasks #0 & #3 placed on node #1
      - Tasks #1 & #4 placed on node #2
      - Tasks #2 & #5 placed on node #3

Processor Affinity: Principles

- **Purpose**
  - Avoid as much as possible resource sharing
  - Avoid Linux Scheduler to move processes / threads between CPU cores
- **Management**
  - Manual
    - Through LSF Affinity String (Resource Requirement)
  - Automated
    - Through LSF Application (esub)
    - New in Spectrum LSF 10.1
- **Tips'n Tricks**
  - Checking Required!
    - ALWAYS check applied processor affinity with the help of monitoring tools
      - htop, nmon

IBM Client Center

Processor Affinity: Spectrum LSF Support

- **Application [Automated = Easy Way]**
  - Syntax
    - #BSUB -a p8aff(num_threads_per_task, SMT, cpus_per_core, distribution_policy)
      - cpus_per_core: # logical CPUs used per physical core
      - distribution_policy = { pack | balance }
  - Reference
    - https://goo.gl/VMTtNq

- **Affinity String [Manual = Hard Way]**
  - Syntax
    - #BSUB -R affinity[affinity_string]
      - {core|thread}(n):cpubind={core|thread}:distribute={balance|pack}
  - Reference
    - https://goo.gl/5v6Qmu

Processor Affinity: Standard Examples

Execution Configuration	‘p8aff’ esub Options
2 MPI Tasks x 10 Threads Per Task	#BSUB -a “p8aff(10,1,1,pack)”
4 MPI Tasks x 5 Threads Per Task	#BSUB -a “p8aff(5,1,1,pack)”
10 MPI Tasks x 2 Threads Per Task	#BSUB -a “p8aff(2,1,1,pack)”
20 MPI Tasks x 1 Thread Per Task	#BSUB -a “p8aff(1,1,1,pack)”
20 MPI Tasks x 2 Threads Per Task	#BSUB -a “p8aff(2,2,2,pack)”
20 MPI Tasks x 4 Threads Per Task	#BSUB -a “p8aff(4,4,4,pack)”
20 MPI Tasks x 8 Threads Per Task	#BSUB -a “p8aff(8,8,8,pack)”

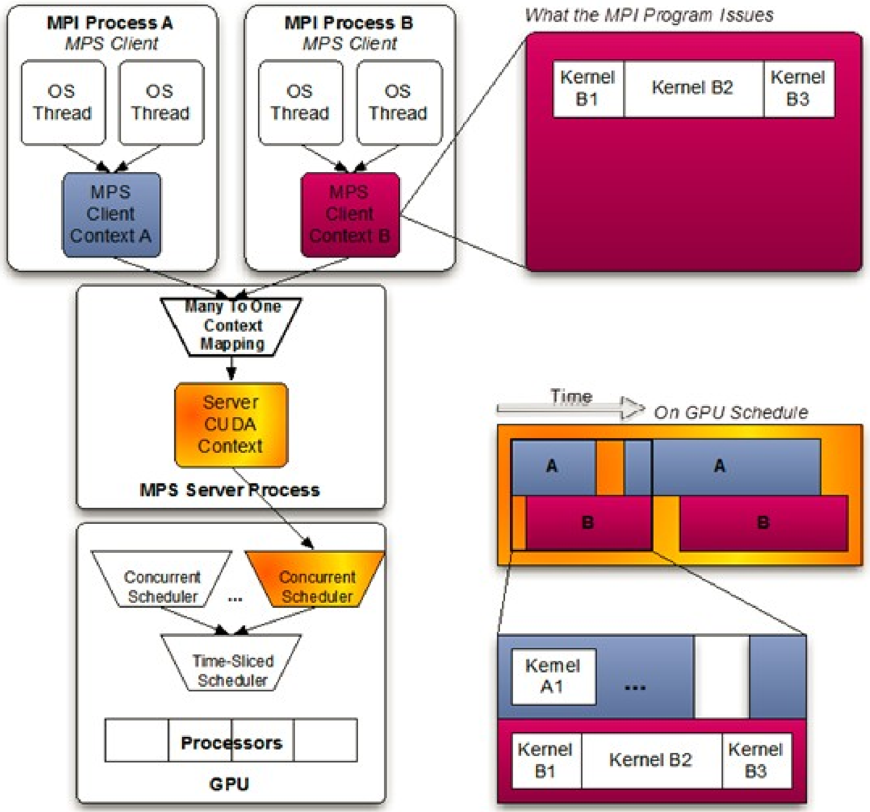
GPU Resource Requirement: GPU Compute Mode

- **Available Compute Modes**
  - DEFAULT
    - Equivalent to: Shared
  - EXCLUSIVE_PROCESS
    - Only one process allowed to run
  - EXCLUSIVE_THREAD
    - Only one thread allowed to run
  - PROHIBITED
    - Run not allowed
- **GPU-Specific**
  - GPUs can have different Compute Modes

```
[user@host ~]$ nvidia-smi --query --display=COMPUTE

==============NVSMI LOG==============

Timestamp                                    : Tue Nov  1
19:59:57 2016
Driver Version                               : 361.93.02

Attached GPUs                                : 4
GPU 0002:01:00.0
    Compute Mode                             : Default

GPU 0003:01:00.0
    Compute Mode                             : Default

GPU 0006:01:00.0
    Compute Mode                             : Default

GPU 0007:01:00.0
    Compute Mode                             : Default
```

GPU Resource Requirement: Multi-Process Service (MPS)

- MPS Benefits
  - GPU Utilization
    - w/o MPS: Single process may underutilize the compute and memory-bandwidth capacity
    - w/MPS: kernel and memcopy operations from different processes can overlap on the GPU, achieving higher utilization and shorter running times
  - Reduced On-GPU Context Storage
    - w/o MPS: each CUDA process using a GPU allocates separate storage and scheduling resources on the GPU
    - w/MPS: server allocates one copy of GPU storage and scheduling resources shared by all of its clients
  - Reduced GPU Context Switching
    - w/o MPS: when processes share the GPU their scheduling resources must be swapped on and off the GPU
    - w/MPS: server shares one set of scheduling resources between all of its clients, eliminating the overhead of swapping when the GPU is scheduling between those clients



GPU Resource Requirement: Spectrum LSF Support

- **Resource Requirement Syntax**
  - #BSUB -R “rusage[{ngpus_excl_p|ngpus_shared}=<num_gpus>]”
- **Limitation**
  - Spectrum LSF does not manage GPU Compute Mode
    - Statically defined by System Administrator
    - Current state can be examined by the following command
      - lsload -l ngpus:ngpus_excl_p:ngpus_shared <host>

Execution Configuration	Resource Requirement
4 GPUs, Shared Mode	#BSUB -R “rusage[ngpus_shared=2]”
2 GPUs, Exclusive Mode	#BSUB -R “rusage[ngpus_excl_p=2]”
4 GPUs, Exclusive Mode	#BSUB -R "rusage[ngpus_excl_p=4]

GPU Affinity: Principles

- **Purpose**
  - Avoid as much as possible resource sharing
  - Prefer GPU direct access (Avoid going through CPU-To-CPU link)
- **Management**
  - Manual
    - Through environment variable setting (CUDA_VISIBLE_DEVICES)
  - Automated
    - Forthcoming… hopefully!
- **Tips'n Tricks**
  - Checking Required!
    - ALWAYS check applied processor affinity with the help of monitoring tools
      - gpustat, nvidia-smi pmon

- **Objective**
  - Set CUDA_VISIBLE_DEVICES environment variable with proper value
    - Default value set by Spectrum LSF must be overriden
  - Variable should have distinct values for each MPI task
    - At least for exclusive GPU access
- **Suggested Solution**
  - Initialize CUDA_VISIBLE_DEVICES value based on MPI rank
  - Actual logic might depend on execution configuration
    - # MPI Tasks, Processor Affinity
- **Warning**
  - CUDA_VISIBLE_DEVICES=‘’ (null value)
    - Means no GPU assigned to the task
    - Triggers error message at first GPU access

End of Presentation

End Of Presentation