OpenPOWER Platform

Hardware Architecture, HPC Software Stack, Job Submission

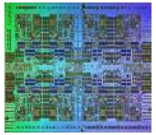
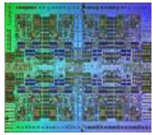
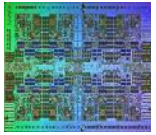
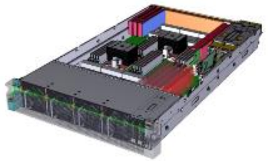
Nicolas Tallet | IBM | nicolas.tallet@fr.ibm.com

IBM Client Center

OpenPOWER Technical Architecture

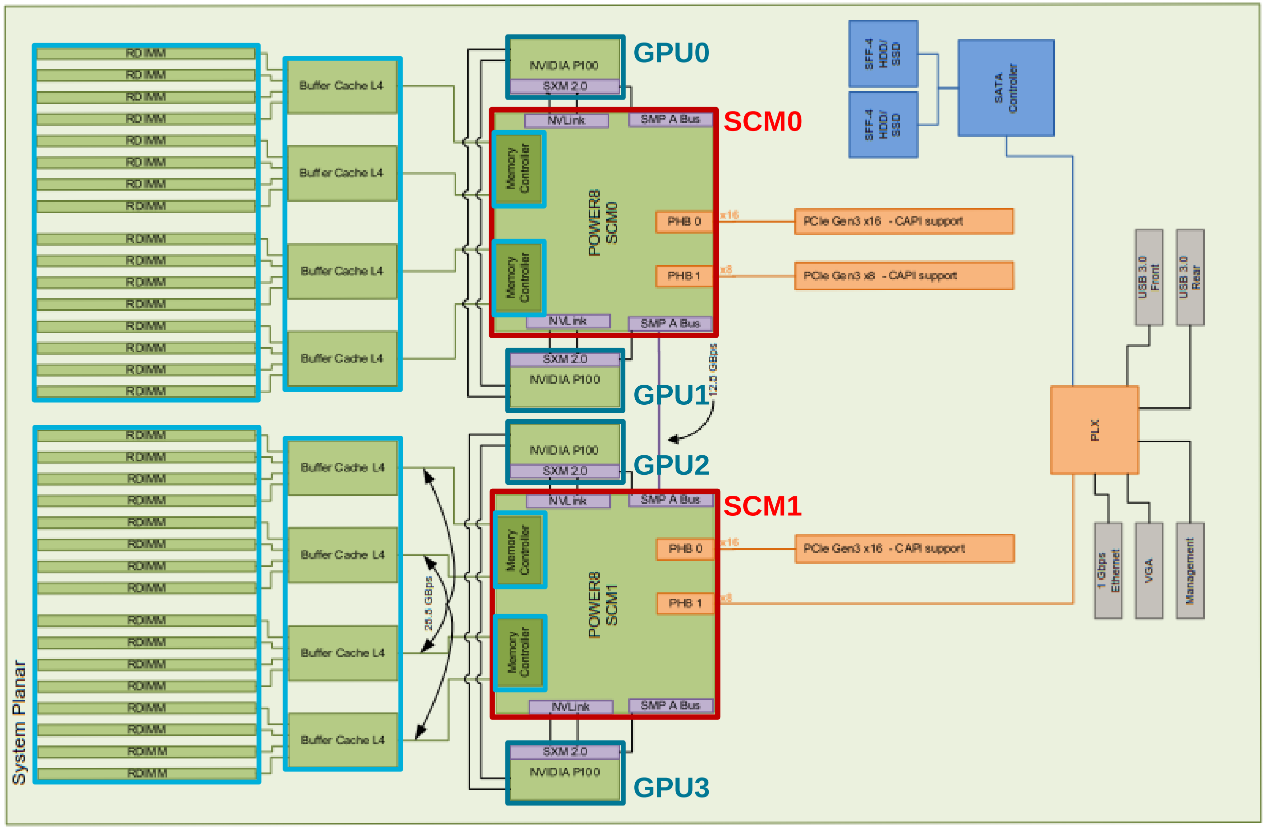
Hardware Features

IBM OpenPOWER Accelerated Computing Roadmap

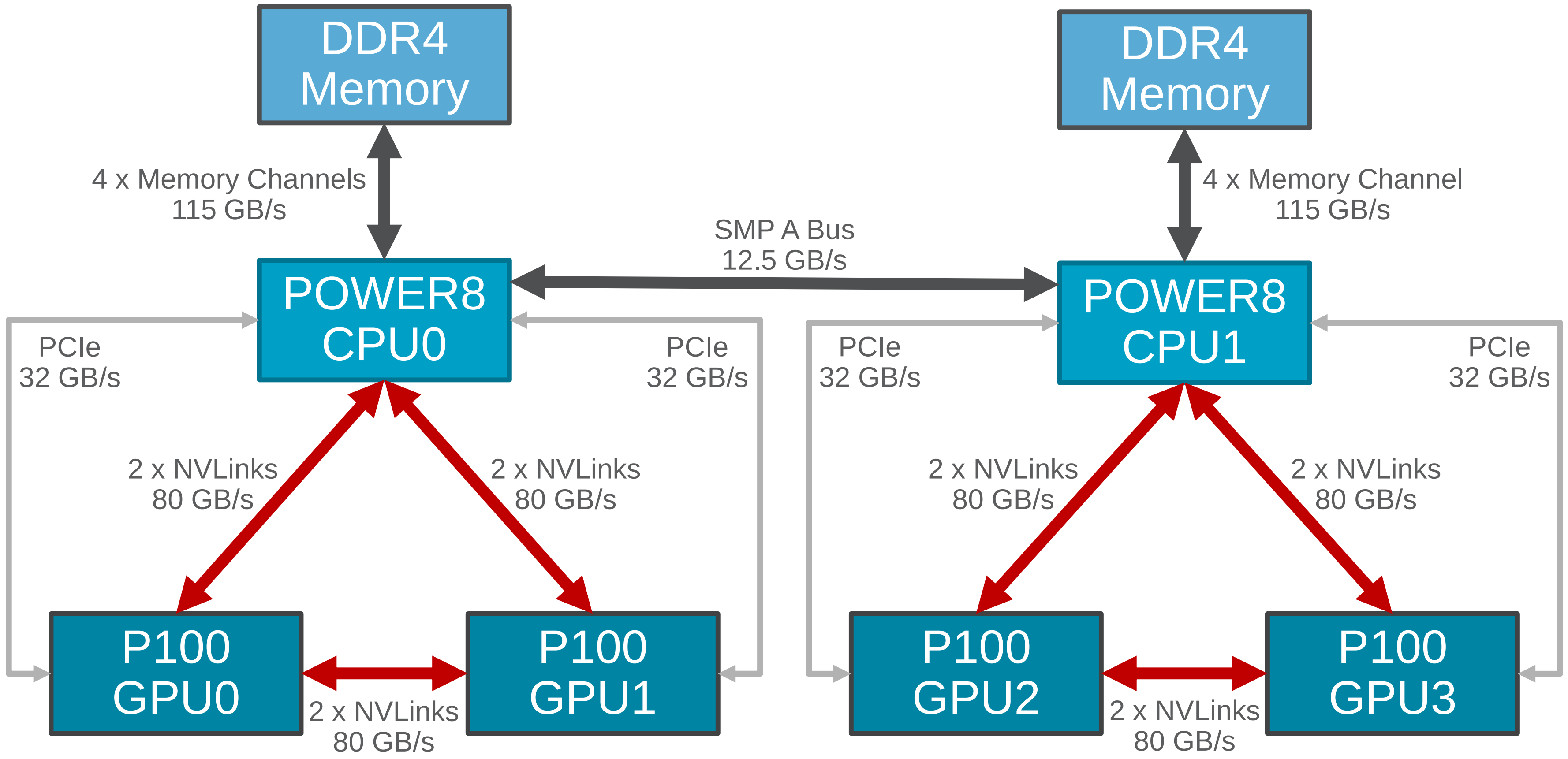
	2015	2016	2017
Mellanox Interconnect Technology	ConnectX-4 EDR Infiniband PCIe Gen3	ConnectX-4 EDR Infiniband CAPI over PCIe Gen3	ConnectX-5 HDR Infiniband Enhanced CAPI over PCIe Gen4
NVIDIA GPUs	Kepler PCIe Gen3	Pascal NVLink SXM2	Volta Enhanced NVLink SXM2
IBM CPUs	POWER8 	POWER8 w/NVLink  NVLink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

System Architecture: Overview

- Two Single-Chip Modules (SCM)
- Memory Subsystem
  - L3 Cache
    - 8MB local region / core (FLR-L3)
    - Can also be accessed other L3 cache regions as shared L3 cache
  - L4 Cache
    - On Memory Riser card
    - 16MB buffer / card
    - 128 MB cache by using all cards
  - Main Memory
    - 2 Memory Controllers / SCM
    - 2 Memory Riser Cards / Controller
    - 4 RDIMM Slots / Riser Card
    - 32 DIMMs Max. = 1TB Max.
- GPU Subsystem
  - 4 GPU Sockets (300 W Max.)
- Buses
  - 3 PCIe Gen3 Slots
  - Dedicated PCI Bus:
    - Integrated SATA Controller
    - Integrated Ethernet
    - Integrated USB Port



System Architecture: Data Links



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

System Topology: CPU

SMT=1 [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

```
$ 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
```

IBM Power System S822LC ‘Minsky’ (8335-GTB)

POWER8 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

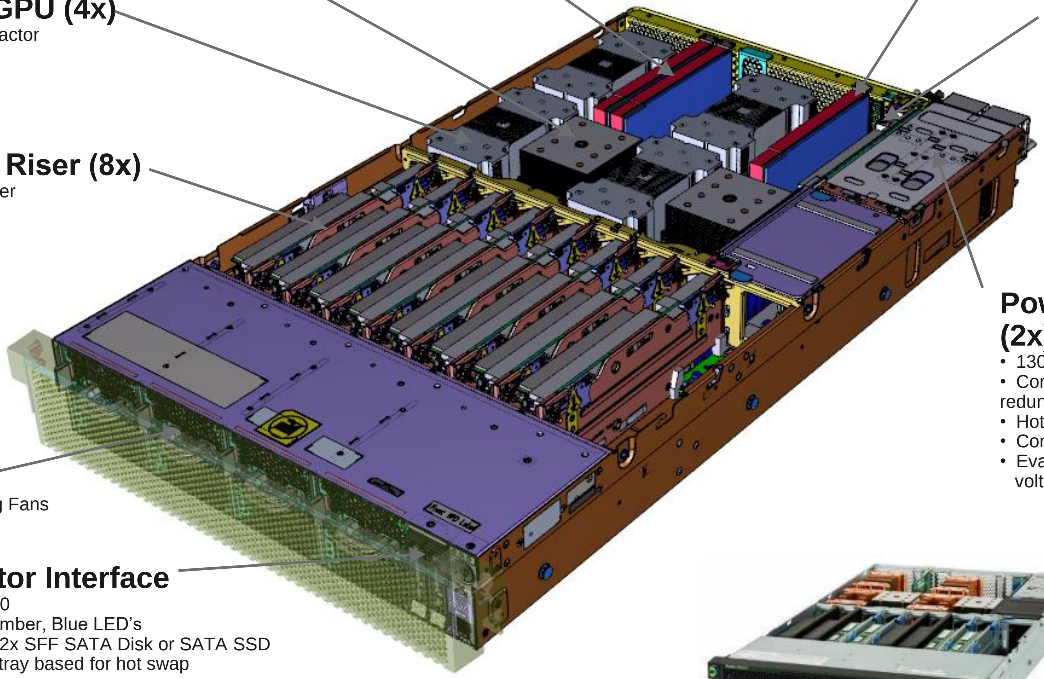
Service Controller Card
- Firestone BMC Content
- Daughter card due to planar space constraints

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

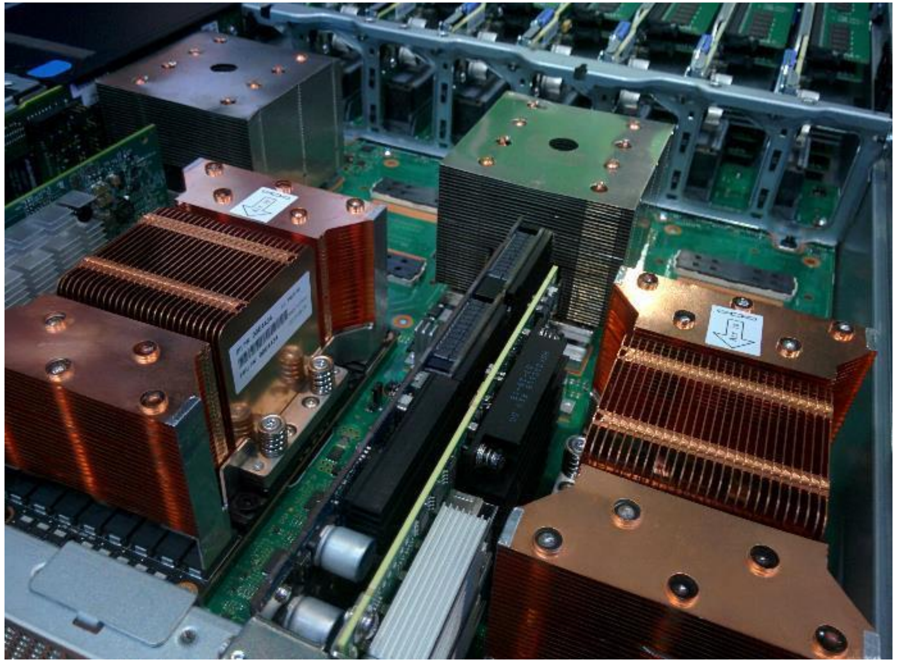
Cooling Fans
- 80mm Counter- Rotating Fans
- Hot swap

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

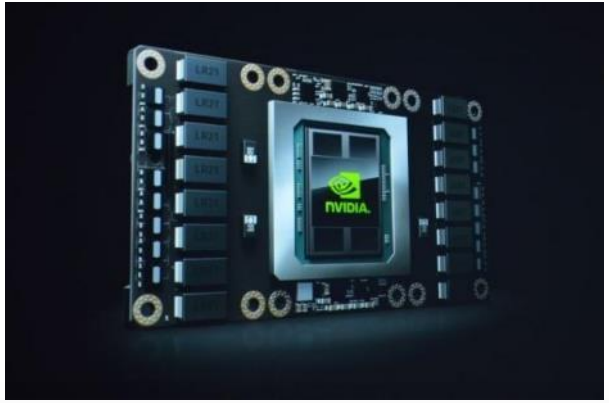


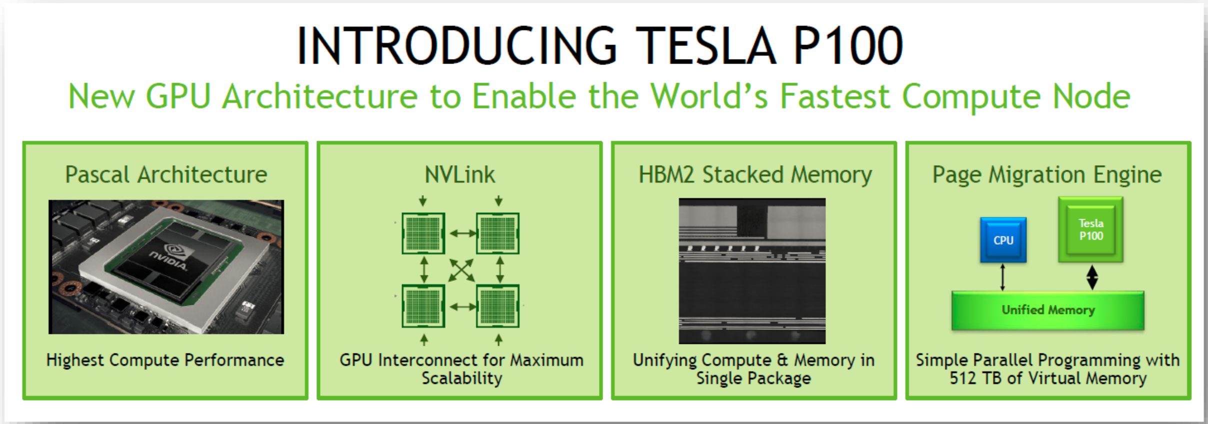
IBM Power System S822LC ‘Minsky’ (8335-GTB)

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 (Bidirectional)	40 GB/s
DP Performance (Peak)	4.9 TFlops



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 9.0 [GCC 5.4], 10.0 [6.3], 11.0 [7.2]
- CUDA Toolkit 8.0
- GCC 4.8
- **IBM XL C/C++** 13.1.5 & **13.1.6 [Beta 13]**
- **IBM XL Fortran** 15.1.5 & **15.1.6 [Beta 13]**
- LLVM Clang & XLFlang [Beta]
- **PGI Accelerator** 17.07 & **17.09**

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)

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

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

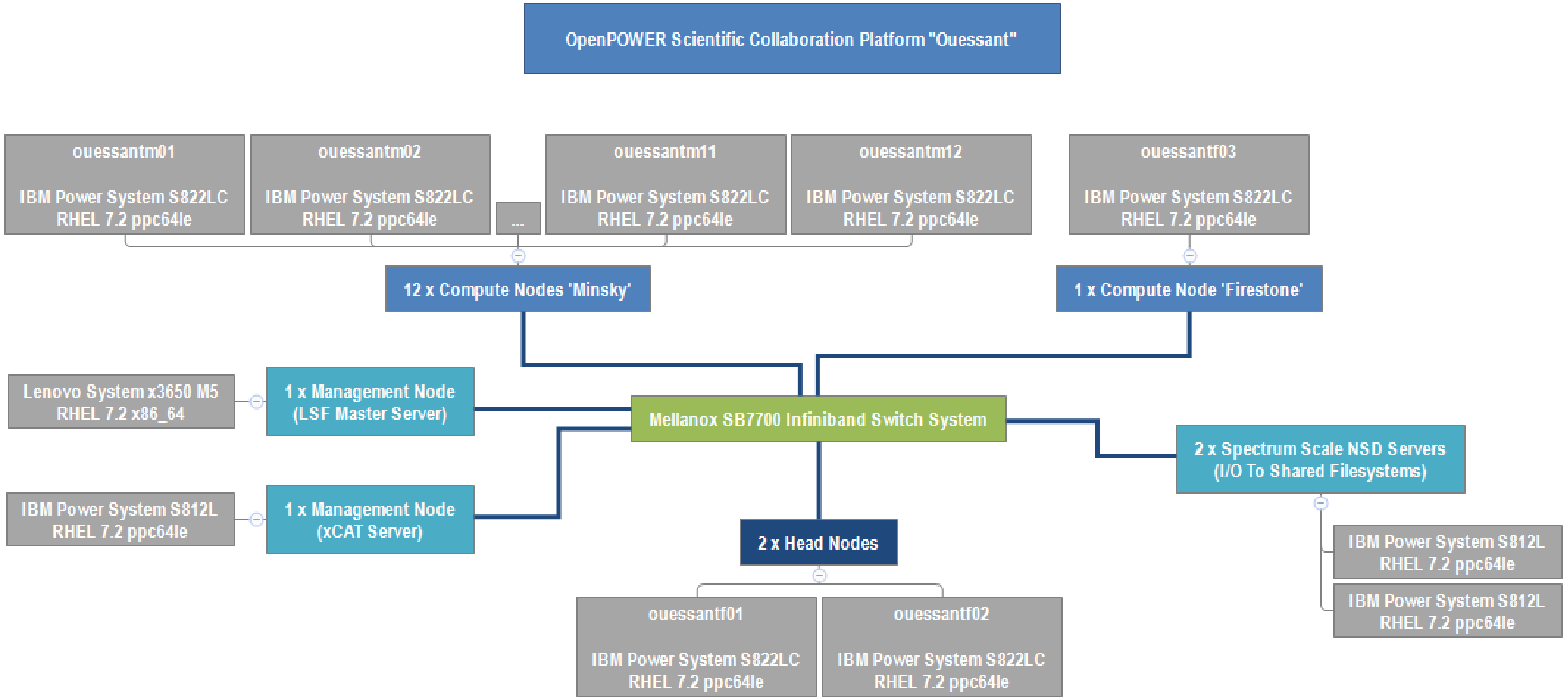
Target Compilation Environments & Associated Programming Models

Compiler	MPI Library	Programming Models
IBM XL C/C++ IBM XL Fortran	IBM Spectrum MPI	CUDA, OpenMP
PGI Accelerator	IBM Spectrum MPI (Open MPI)	CUDA, OpenACC
GCC	IBM Spectrum MPI	It’s Complicated ☺
LLVM	IBM Spectrum MPI	CUDA, OpenMP

Ouessant Platform

Scientific Collaboration Prototype

Platform Architecture


OpenPOWER Scientific Collaboration Platform "Ouessant"
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
12 x Compute Nodes 'Minsky'
1 x Compute Node 'Firestone'
Lenovo System x3650 M5 RHEL 7.2 x86_64
1 x Management Node (LSF Master Server)
Mellanox SB7700 Infiniband Switch System
2 x Spectrum Scale NSD Servers (I/O To Shared Filesystems)
IBM Power System S812L RHEL 7.2 ppc64le
1 x Management Node (xCAT Server)
2 x Head Nodes
IBM Power System S812L RHEL 7.2 ppc64le
IBM Power System S812L RHEL 7.2 ppc64le
ouessantf01 IBM Power System S822LC RHEL 7.2 ppc64le
ouessantf02 IBM Power System S822LC RHEL 7.2 ppc64le

Job Submission
& Execution Environment

Execution Environment

Setting	Purpose
Task Placement	Define target compute node among all allocated hosts for each application process
Processor Affinity	Define allowed CPU cores for each application process
GPU Resource Requirement	Define GPU resource requirement for the whole job
GPU Affinity	Define associated GPU for each application process

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)
- **Tips'n Tricks**
  - Checking Required!
    - ALWAYS check applied processor affinity

Processor Affinity: Spectrum LSF Support

- **LSF Application [Semi-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

- **LSF 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: CPU - A Few Useful Examples 1/2

p8aff(10,8,1,balance)

CPUs/Cores	Core #01	Core #02	Core #03	Core #04	Core #05	Core #06	Core #07	Core #08	Core #09	Core #10	Core #11	Core #12	Core #13	Core #14	Core #15	Core #16	Core #17	Core #18	Core #19	Core #20
CPU #0	P00.t00	P00.t01	P00.t02	P00.t03	P00.t04	P00.t05	P00.t06	P00.t07	P00.t08	P00.t09	P01.t00	P01.t01	P01.t02	P01.t03	P01.t04	P01.t05	P01.t06	P01.t07	P01.t08	P01.t09
CPU #1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

p8aff(1,8,1,balance)

CPUs/Cores	Core #01	Core #02	Core #03	Core #04	Core #05	Core #06	Core #07	Core #08	Core #09	Core #10	Core #11	Core #12	Core #13	Core #14	Core #15	Core #16	Core #17	Core #18	Core #19	Core #20
CPU #0	P00	P01	P02	P03	P04	-	-	-	-	-	P05	P06	P07	P08	P09	-	-	-	-	-
CPU #1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

p8aff(1,8,1,pack)

CPUs/Cores	Core #01	Core #02	Core #03	Core #04	Core #05	Core #06	Core #07	Core #08	Core #09	Core #10	Core #11	Core #12	Core #13	Core #14	Core #15	Core #16	Core #17	Core #18	Core #19	Core #20
CPU #0	P00	P01	P02	P03	P04	P05	P06	P07	P08	P09	-	-	-	-	-	-	-	-	-	-
CPU #1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Processor Affinity: CPU - A Few Useful Examples - 2/2

p8aff(2,8,2,balance)

CPUs/Cores	Core #01	Core #02	Core #03	Core #04	Core #05	Core #06	Core #07	Core #08	Core #09	Core #10	Core #11	Core #12	Core #13	Core #14	Core #15	Core #16	Core #17	Core #18	Core #19	Core #20
CPU #0	P00.t00	P01.t00	P02.t00	P03.t00	P04.t00	-	-	-	-	-	P05.t00	P06.t00	P07.t00	P08.t00	P09.t00	-	-	-	-	-
CPU #1	P00.t01	P01.t01	P02.t01	P03.t01	P04.t01	-	-	-	-	-	P05.t01	P06.t01	P07.t01	P08.t01	P09.t01	-	-	-	-	-
CPU #2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

p8aff(2,8,2,balance)

CPUs/Cores	Core #01	Core #02	Core #03	Core #04	Core #05	Core #06	Core #07	Core #08	Core #09	Core #10	Core #11	Core #12	Core #13	Core #14	Core #15	Core #16	Core #17	Core #18	Core #19	Core #20
CPU #0	P00.t00	P01.t00	P02.t00	P03.t00	P04.t00	P05.t00	P06.t00	P07.t00	P08.t00	P09.t00	P10.t00	P11.t00	P12.t00	P13.t00	P14.t00	P15.t00	P16.t00	P17.t00	P18.t00	P19.t00
CPU #1	P00.t01	P01.t01	P02.t01	P03.t01	P04.t01	P05.t01	P06.t01	P07.t01	P08.t01	P09.t01	P10.t01	P11.t01	P12.t01	P13.t01	P14.t01	P15.t01	P16.t01	P17.t01	P18.t01	P19.t01
CPU #2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

p8aff(1,8,1,balance)

CPUs/Cores	Core #01	Core #02	Core #03	Core #04	Core #05	Core #06	Core #07	Core #08	Core #09	Core #10	Core #11	Core #12	Core #13	Core #14	Core #15	Core #16	Core #17	Core #18	Core #19	Core #20
CPU #0	P00	P02	P04	P06	P08	P10	P12	P14	P16	P18	P20	P22	P24	P26	P28	P30	P32	P34	P36	P38
CPU #1	P01	P03	P05	P07	P09	P11	P13	P15	P17	P19	P21	P23	P25	P27	P29	P31	P33	P35	P37	P39
CPU #2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CPU #7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

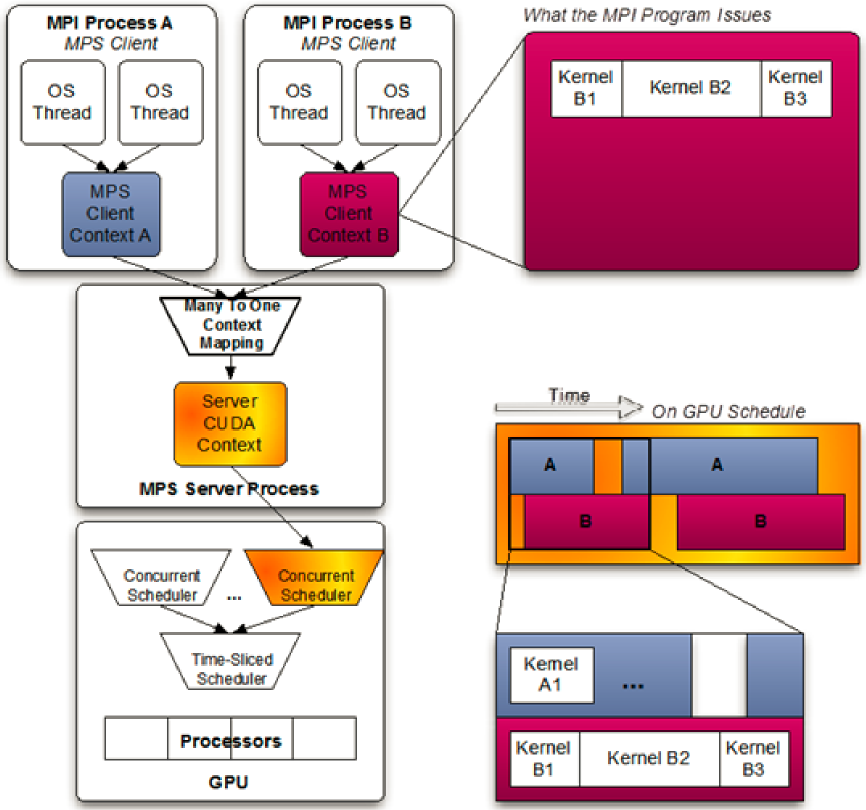
GPU Resource Requirement: GPU Compute Mode

- **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

```
$ 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

- **Syntax**
  - #BSUB -gpu “num=<num_gpus>:mode={exclusive_process|shared*}:mps={no*|yes}:j_exclusive={no|yes*}”
    - j_exclusive: Is GPU resource exclusive to user job or shared between multiple jobs?

Execution Configuration	Resource Requirement
4 GPUs, Shared Mode	#BSUB -gpu “num=4:mode=shared:mps=no:j_exclusive=yes]”
4 GPUs, Exclusive Mode, w/o MPS	#BSUB -gpu “num=4:mode=exclusive_process:mps=no:j_exclusive=yes]”
4 GPUs, Exclusive Mode, w/MPS	#BSUB -gpu “num=4:mode=exclusive_process:mps=yes:j_exclusive=yes]”

GPU Affinity: Principles

- **Purpose**
  - Avoid as much as possible resource sharing (favor exclusive GPU access)
  - Prefer GPU direct access (Avoid going through SMP link)
- **Management**
  - Manual Specification
    - Initialize CUDA_VISIBLE_DEVICES value based on MPI rank
      - Actual logic might depend on execution configuration (# MPI Tasks, Processor Affinity)
  - Automated Specification
    - Forthcoming… hopefully!
    - Intersection between Workload Scheduler and MPI Library
- **Classical Issue**
  - Null value for CUDA_VISIBLE_DEVICES triggers error message at first offloaded code section
    - Current region was compiled for:
      NVIDIA Tesla GPU sm30 sm35
      Available accelerators:
      device[1]: Native X86 (CURRENT DEVICE)
      The accelerator does not match the profile for which this program was compiled

GPU Affinity: Manual Management

- **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

Helper Script (For Easier Execution Environment Management)

- **Ideal Situation**
  - Workload Scheduler as unique orchestrator
    - Comprehensive management of SMT Mode, Compute Mode, Affinity…
- **Current Situation**
  - Some very good progress
  - Still a bit of manually-defined user control
- **Helper Scripts Collection**
  - /pwrlocal/ibmccmpl/bin/task_prolog.sh
    - Visible GPU Devices Selection
    - Additional Features (for free!)
      - Perf Profiling Startup
      - nvprof Profiling Startup

Typical Submission Script

```
#!/bin/bash

#BSUB -a p8aff(5,8,1,balance)
#BSUB -cwd ~/hpl
#BSUB -e hpl.%J.log
#BSUB -gpu "num=4:mode=exclusive_process:mps=no:j_exclusive=yes"
#BSUB -J HPL
#BSUB -n 4
#BSUB -o hpl.%J.log
#BSUB -R "span[ptile=4]"
#BSUB -W 01:00

mpirun -prot -report-bindings /pwrlocal/ibmccmpl/bin/task_prolog.sh -
devices auto ~/hpl/xhpl.exe
```

End Of Presentation

Reference

Spectrum LSF Directives

Option	Value	Purpose
-cwd	<Path>	Execution Directory
-e	<File>	stderr File
-gpu	“num=<num>:mode={exclusive_process|shared}:mps={no|yes}:j_exclusive={no|yes}”	
-J	<Job Name>	Job Name
-n	<# MPI Tasks>	Total Number of MPI Tasks
-o	<File>	stdout File
-R	“span[ptile=<ppn >]”	Resource Specification: Number of Tasks per Node
-W	HH:MM	Run Limit
-x		Travail exclusif

IBM Client Center

Spectrum LSF 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