APPLICATION USER GUIDE
SCIENTIFIC COLLABORATION PLATFORM
VERSION 16.12.1

Ludovic ENAULT
Nicolas TALLET
Pascal VEZOLLE

1. General Information

Ouessant constitutes the technical platform for the Scientific Collaborationaround OpenPOWER technologies established between GENCI on one side, and IBM, Mellanox et NVIDIA on the other side.

The platform is based on IBM Power System S822LC compute nodes, which general characteristics are:

- 12 IBM Power System S822LC « Minsky » compute nodes.
- 20 physical cores, and up to 160 logical cores (hradware threads) per node.
- 4 NVIDIA Tesla P100 GPU per node.
- Peak computing performance:
    - 470 GFlops for CPU subsystems.
    - 20 TFlops for GPU subsystems.
- 128 GB of memory per node, i.e. soit 6 GB / physical core.
- Dedicated IBM Spectrum Scale (GPFS) filesystem with 128 TB shared storage capacity.

The applications to be executed on Ouessant must be able to take advantage of the GPU accelerators in order to fully benefit from the OpenPOWER architecture.

Several development languages will be available to achieve GPU offloading:

- Development through OpenMP directives
- Development through OpenACC directives
- Development through direct CUDA language

These different acceleration models will be studied as part of the the Scientific Collaboration activities that will be performed on the platform.

2. QUICKSTART

ONE PAGE REFERENCE FOR IMPATIENT USERS ☺

2.1. COMPILATION ENVIRONMENT SELECTION

- IBM XL C/C++ & IBM XL Fortran:

```
[user@host ~] ml xlc xlf
[user@host ~] which xlc xlf
/opt/ibm/xlC/13.1.5/bin/xlc
/opt/ibm/xlf/15.1.5/bin/xlf
```

- PGI Accelerator C/C++/Fortran:

```
[user@host ~] ml pgi
[user@host ~] which pgcc pgfortran
/opt/pgi/linuxpower/16.10/bin/pgcc
/opt/pgi/linuxpower/16.10/bin/pgfortran
```

- IBM XL C/C++ & IBM XL Fortran + IBM Spectrum MPI:

```
[user@host ~] ml xlc xlf smpi
[user@host ~] mpicc -show
xlc_r -I/opt/ibm/spectrum_mpi/include -pthread -L/opt/ibm/spectrum_mpi/lib -lmpi_ibm
[user@host ~] mpif90 -show
xlf90_r -I/opt/ibm/spectrum_mpi/include -qthreaded -I/opt/ibm/spectrum_mpi/lib/XL -
L/opt/ibm/spectrum_mpi/lib -lmpiprofilesupport -lmpi_usempi -lmpi_mpifh -lmpi_ibm
```

- IBM XL C/C++ & IBM XL Fortran + IBM Parallel Environment Runtime Edition (PE RTE):

```
[user@host ~] ml xlc xlf perte
[user@host ~] mpicc -show
/opt/ibm/xlC/13.1.5/bin/xlc_r -Xlinker --no-as-needed -F:xlc_r  -q64 -Xlinker --allow-shlib-undefined -
Xlinker --enable-new-dtags -Xlinker -rpath -Xlinker /opt/ibmhpc/pe2303/mpich/gnu/lib64  -
I/opt/ibmhpc/pe2303/mpich/gnu/include64 -I/opt/ibmhpc/pe2303/base/include -
L/opt/ibmhpc/pe2303/mpich/gnu/lib64 -lmpi
[user@host ~]$ mpif90 -show
/opt/ibm/xlf/15.1.5/bin/xlf_r  -F:xlf_r  -q64 -Wl,--no-as-needed -Wl,--allow-shlib-undefined -Wl,--enable-
new-dtags -Wl,-rpath -Wl,/opt/ibmhpc/pe2303/mpich/xlf/lib64  -I/opt/ibmhpc/pe2303/mpich/xlf/include64 -
I/opt/ibmhpc/pe2303/base/include64 -L/opt/ibmhpc/pe2303/mpich/xlf/lib64 -lmpi -lmpigf -ldl
```

- PGI Accelerator C/C++/Fortran + Open MPI

```
[user@host ~] ml pgi ompi
[user@host ~] mpicc -show
pgcc -I/opt/pgi/linuxpower/2016/mpi/openmpi-1.10.2/include -Wl,-rpath -
Wl,/opt/pgi/linuxpower/2016/mpi/openmpi-1.10.2/lib -Wl,--enable-new-dtags -
L/opt/pgi/linuxpower/2016/mpi/openmpi-1.10.2/lib -lmpi
[user@host ~] mpif90 -show
pgfortran -I/opt/pgi/linuxpower/2016/mpi/openmpi-1.10.2/include -I/opt/pgi/linuxpower/2016/mpi/openmpi-
1.10.2/lib -Wl,-rpath -Wl,/opt/pgi/linuxpower/2016/mpi/openmpi-1.10.2/lib -Wl,--enable-new-dtags -
L/opt/pgi/linuxpower/2016/mpi/openmpi-1.10.2/lib -lmpi_usempif08 -lmpi_usempi_ignore_tkr -lmpi_mpifh -lmpi
```

2.2. JOB SUBMISSION

- Choose Target Queue:

Queue Name	Target Nodes	Purpose
compute	ouessantm[01-12]	Default computation queue
interactive_firestone	ouessantf03	Interactive queue on Firestone node

Queue Name	Target Nodes	Purpose
interactive_minsky	ouessantm01	Interactive queue on Minsky node

- Build Job Submission File:

```
[user@host ~] cat myjob.sh
#!/bin/bash

#BSUB -cwd /pwrhome/login
#BSUB -e /pwrhome/login/myjob.log
#BSUB -J HelloWorld
#BSUB -n 20
#BSUB -o /pwrhome/login/myjob.log
#BSUB -q compute
#BSUB -R "affinity[core(1):cpubind=core:distribute=pack]"
#BSUB -R "span[ptile=20]"
#BSUB -W 00:05

mpiexec /pwrhome/login/myjob.bin
```

- Submit Job:

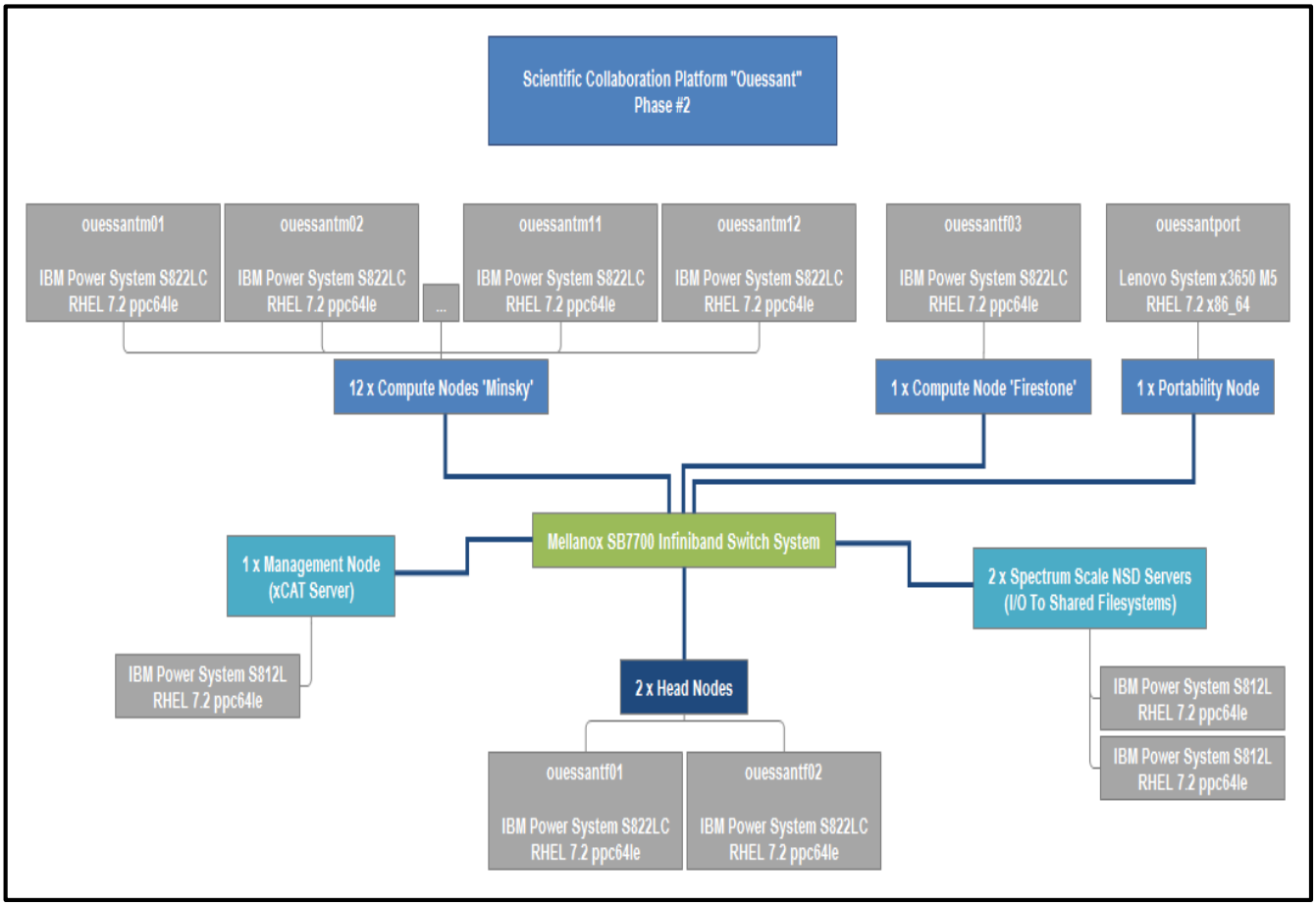
```
bsub < job.sh
```

3. Computation Environment

3.1. Hardware Configuration

3.1.1. Architecture

The platform is made of the following hardware components:



The systems constituting the platform have the following role in this environment:

System	Type	Role
ouessantf[01-02]	IBM Power System S822LC "Firestone"	Login Node
ouessantf03	IBM Power System S822LC "Firestone"	Compute Node
ouessantm[01-12]	IBM Power System S822LC "Minsky"	Compute Node

3.1.2. IBM POWER SYSTEM S822LC "FIRESTONE" & "MINSKY" COMPUTE NODES

The IBM Power System S822LC compute nodes have the following technical characteristics:

	8335-GTA Firestone	8335-GTB Minsky
# CPUs	2	2
Model	POWER8	POWER8
Physical Cores	10	10
Logical Cores / Physical Threads	10-80	10-80
Clock Frequency: Nominal	2.93 GHz	2.93 GHz
Clock Frequency: Max. (Turbo)	3.49 GHz	3.49 GHz
Memory	128 GB	128 GB
Memory Controllers	2	2
Memory Channels	4	4
L3 Cache	8 MB	8 MB
L4 Cache	16 MB / Buffer Chip	16 MB / Buffer Chip
Memory Bandwidth	230 GB/s	230 GB/s
Peak Performance (Double Precision)	235 GFlops	235 GFlops
# GPUs	2	4
Model	Tesla K80	Tesla P100
CUDA Cores	4992	3584
Memory	24 GB	16 GB
Peak Performance (Double Precision)	1.87 Tflops	5.3 TFlops

Simultaneous Multi-Threading (SMT)

The S822LC compute nodes can be configured in 4 distinct SMT modes. Each SMT mode defines the number of logical cores (hardware threads) available on the node:

SMT Mode	# Logical Cores (# Physical Threads)
0 (Off)	20
2	40
4	80
8	160

SMT mode configuration does not require the system to be restarted ; therefore, SMT mode configuration can be performed on-the-fly.

Note: In order to make things easier for the users, it has been decided that compute nodes would ne statically configured in SMT=8 mode at start, with no possibility for the users to alter this configuration.

3.1.3. EDR INFINIBAND INTERCONNECTION

High performance interconnection between the compute nodes is achieved through an EDR Infiniband network (100 Gb/s).

3.2. REMOTE ACCESS

A SSH session can be opened on the platform through the following hostname:

Hostname
ouessant.idris.fr

This hostname is redirected by default to the ouessantf01 login node. Redirection automatically switches to a different login node in case this one is temporarily unavailable.

3.3. FILESYSTEMS

The following filesystems are available on the platform:

Path	Purpose	User Quota
/pwrhome /linkhome	Home Directories	10 GB 150K inodes
/pwrlocal	Shared Software	-
/pwrwork	Scratch Filesystem	-

3.4. SOFTWARE STACK

The following software packages are available on the platform:

Category	Software Package	Version
Compilers	CUDA Toolkit	8.0
	GCC [Standard]	4.8
	GCC [Advance Toolchain 10.0]	6.2
	IBM XL C/C++	13.1.4

		13.1.5
	IBM XL Fortran	15.1.4
		15 .1.5
	LLVM C/C++	N/A
	LLVM Fortran	N/A
	PGI Accelerator	16.10
MPI Libraries	IBM Parallel Environment (PE) Runtime Edition (RTE)	2.3
	Open MPI	2.0
	IBM Spectrum MPI	10.1
Scientific Libraries	IBM ESSL	5.5
	IBM PESSL	5.3
Performance Analysis	IBM Parallel Environment (PE) Developer Edition (DE)	2.3

Other libraries will be made available upon users request: FFTW, HDF5, LAPACK, ScaLAPACK…

Note : IBM XL C/C++ is fully compatible with the C++ 11 standard.

Note : IBM Spectrum MPI is expected to become the default choice for MPI library starting from Q4 2016. IBM Spectrum MPI is based on Open MPI 2.0.

3.5. LMOD ENVIRONNEMENT MODULES

3.5.1. PRINCIPLE

Lmod provides a simple and reliable way to dynamically load / unload the various software components available on the platform.

Lmod takes care of performing the user environment dynamic configuration by automatically altering the associated environment variables (LD_LIBRARY_PATH, MANPATH, PATH…).

3.5.2. USAGE

Lmod usage is based on the following command:

Command	Action
ml avail module avail	Display available modules

ml whatis <module> module whatis <module>	Display module information
ml list module list	Display currently-loaded modules
ml <module> module load <module>	Load specified module
ml unload <module> module unload <module>	Unload specified module
ml switch <mod1> <mod2> module switch <mod1> <mod2>	Substitue mod1 by mod2
ml purge module purge	Unload all currently-loaded modules (fully purge user environment)

3.5.3. MODULES HIERARCHY

Modules are spread into a three-level hierarchy:

Level	Purpose
level0	Compilers, Basic Tools
level1	Software depending on a given compiler Example : Serial libraries
level2	Software depending on a given compiler AND a given MPI library Example : Parallel libraries

The module hierarchy sets the dependencies existing between the software packages, and defines the sequence for loading modules by the user. More precisely :

- Only level-0 modules are available to the user at first.
- Level-1 modules become available once the user has loaded a compiler module.
- Level-2 modules become available once the user has loaded a MPI library module.

4. COMPILATION

4.1. COMPILERS

The invocation commands of the compilers are the following:

Compilateur	C	C++	Fortran
GCC	gcc	g++	gfortran
IBM XL	xlc_r	xlC_r	xlf_r xlf90_r xlf95_r xlf2003_r xlf2008_r
LLVM	clang	clang++	xlflang
PGI	pgcc	pgCC	pgf90

Note : The commands specified above constitute the « thread-safe » variants of the IBM XL compilers. It is recommended to exclusively use these variants, even in the case of non-multithreaded applications.

The table below provides the main equivalences between the options of the different compilers available on the platform:

Purpose	GCC	IBM XL	LLVM	PGI
Architecture	-mcpu=power8	-qarch=pwr8	-target powerpc64le-unknown-linux-gnu -mcpu=pwr8	-m64
Disabled Optimization	-O0	-O0 -qnoopt	-O0	-O0
Optimization Levels	-O / -O1 -O2 -O3 -Ofast	-O -O2 -O3 -O4 -O5	-O0 -O2 -O3 -Os	-O1 -O -O2 -O3 -O4
Recommended Optimization	-O3 -mcpu=power8 -funroll-loops	-O3 [-qipa -qhot]		
Profile-Guided Optimization (PGO)	-fprofile-generate -fprofile-use	-qpdf1 -qpdf2	-fprofile-instr-generate -fprofile-instr-use	-Mpfi -Mpfo
Inter-Procedural Optimization	-flto	-qipa		
Vectorization	-ftree-vectorize	-qsimd=auto		-Mvect
OpenMP Support	-fopenmp	-qsmp=omp	-fopenmp	-mp
OpenMP/OpenACC Target		-qoffload -qxflag=sm_60	-fomptargets=nvptx64-nvidia-cuda	-acc -ta:tesla:cc60
Automatic Parallelization	-floop-parallelize-all	-qsmp=auto		
Loop Optimization	-fpeel-loops -funroll-loops	-qhot	-funroll-loops	
Debugging Symbols	-g	-g	-g	

4.1.1. IBM XL

The IBM XL compilers allow the following optimization levels:

Options	Purpose
-O0 -qnohot -qnosmp -qstrict	Disable any optimization
-O2	Limited optimization (safe)
-O3 -qstrict	Medium optimization without modification of the code semantics
-O3	Medium optimization with potential modifications of the code semantics
-O4	Agressive optimization + Light Inter-Procedural Analysis (IPA)
-O5	Equivalent to -O4 level with deeper IPA

Note : Increasing the optimization level tends to lead to a potentially significant increase in compilation time. It is therefore recommended to keep limited optimization levels while porting an application onto the platform.

For a given optimization level, the IBM XL compilers automatically introduce additional options by default. These options can be in turn complemented with other additional options.
The table below details the main possible combinations:

Optimization Level	Additional Options Introduced By Default	Additional Options	Other Options with Potential Benefits
-O0	-qsimd=auto	-qarch	
-O2	-qmaxmem=8192 -qsimd=auto -qstrict=all	-qarch -qtune	-qmaxmem=-1 -qhot=level=0
-O3	-qhot=noarraypad:level=0 :novector:fastmath -qnostrict -qmaxmem=-1 -qsimd=auto	-qarch -qtune	
-O4	-qarch=auto -qcache=auto -qhot=noarraypad:level=1 :vector:fastmath -qipa -qmaxmem=-1 -qnostrict -qsimd=auto -qtune=auto	-qarch -qcache -qtune	-qsmp=auto
-O5	[Idem -O4] -qipa=level=2	-qarch -qcache -qtune	-qsmp=auto

4.1.2. PGI ACCELERATOR

It is highly recommended to specify the GPU target explicitely, through the following options:

Component	Option
Compute Capability	-ta={ cc37 | cc60 }
CUDA	-ta=cuda8.0

4.2. MPI WRAPPERS

The invocation commands of the MPI wrappers are the following:

MPI Library	C	C++	Fortran
IBM PE	mpcc mpicc	mpCC mpiCC	mpfort mpifort
Open MPI	mpicc	mpic++ mpiCC mpicxx	mpifort
Spectrum MPI	mpicc	mpicxx	mpif90

Note : Les wrappers MPI portent strictement le même nom indépendamment du compilateur utilisé. La sélection d'un compilateur donné est réalisée par définition de variables d'environnement propres à la bibliothèque MPI. Sur la plate-forme Ouessant, cette définition est réalisée automatiquement dans les modules de la bibliothèque MPI, mais peut évidemment être manuellement ajustée.

Note : L'option '-show' permet de vérifier quel est le compilateur utilisé lors de l'appel du wrapper. Elle est donc très utile pour valider que c'est bien le compilateur attendu qui est invoqué.

Example : Checking the Compiler called by the MPI Wrapper

```
[login@ouessant ~]$ mpcc -show
/opt/ibm/xlC/13.1.5/bin/../bin/xlc_r -Xlinker --no-as-needed -F:xlc_r  -q64 -Xlinker --allow-shlib-
undefined -Xlinker --enable-new-dtags -Xlinker -rpath -Xlinker /opt/ibmhpc/pe2300/mpich/gnu/lib64 -Xlinker
-rpath -Xlinker /opt/ibmhpc/pe2300/ppe.poe/gnu/lib64  -I/opt/ibmhpc/pe2300/mpich/gnu/include64 -
I/opt/ibmhpc/pe2300/base/include -L/opt/ibmhpc/pe2300/mpich/gnu/lib64 -lmpi -
L/opt/ibmhpc/pe2300/ppe.poe/gnu/lib64 -lpoe
```

5. JOB SUBMISSION

The Workload Scheduler is responsible for:

- Allocating computing resources to user jobs dynamically.
- Executing the user jobs on the allocated computing resources.

Following the acquisition of the Platform Computing company by IBM in 2012, the default workload scheduler became IBM Spectrum LSF, which substitues IBM LoadLeveler.

5.1. IBM SPECTRUM LSF COMMANDS

The main commands to interact with the workload scheduler are the following:

Command	Argument	Purpose
bjobs		Display active user jobs (waiting or executing)
	-u { <User ID> | all }	Specify user
bkill	<Job ID>	
bpeek	<Job ID>	Display user job stdout/stderr
	-f	Refresh display in realtime
bqueues		Display submission queues
bstatus	<Job ID>	Display user job status
bsub	-app <Application Profile>	Call a specific application profile
	< <Submission Script>	Submit user job into a submission queue

Note : One specificity of BM Spectrum LSF is related to the buffering of the stdout / stderr streams during job execution, before final writing into the location specified by the user. Therefore, during execution, target files remain empty, and the following command allows visualizing the progress of the execution : 'bpeek -f'.

5.2. SUBMISSION SCRIPT

A submission script allows submitting user jobs through the workload scheduler.

A submission script is made of two distinct sections:

- Section #1: Directives
  The header consists in a set of IBM Spectrum LSF directives.
  These directives are introduced by the following prefix: '#BSUB'.
  These directives are specified through the following syntax : <Option> <Value>.
- Section #2: Shell Script
  The script in itself is made of standard Bash commands.
  For parallel executions, the script involves a call to the parallel submission command - which depends on the MPI library used.

The main IBM Spectrum LSF directive keywords are the following:

Option	Value	Purpose
-cwd	<Path>	Execution directory
-e	<File>	stderr file
-J	<Job Name>	User job name
-n	<# MPI Tasks>	Total number of MPI tasks
-o	<File>	stdout file
-q	<Queue>	Target queue
-R	"span[ptile=<ppn>]"	Number of MPI tasks per node
	"affinity[<level>(#):cpubind=<level>:distribute=<policy>]"	Affinity management
-W	HH:MM	Runlimit
-x		Exclusive job

Note : IBM Spectrum LSF uses an automatically-created directory inside the user Home Directory in order to temporarily store the elements related to a submitted job. The location of this directory is the following:

${HOME}/.lsbatch

5.3. ENVIRONMENT VARIABLES

When a user job starts, IBM Spectrum LSF initializes a set of environment variables. These variables can be used in the submission script if necessary.

The main environment variables are the following:

Variable	Value	Origin
LS_SUBCWD	Current Working Directory	LSF directive
LSB_DJOB_HOSTFILE	Absolute path to hostfile	Defined by sbatchd
LSB_DJOB_NUMPROC	Number of allocated slots	Defined by sbatchd
LSB_DJOB_RANKFILE	Absolute path to rankfile	Defined by sbatchd
LSB_ERRORFILE	stderr file	LSF directive
LSB_EXECHOSTS	List of allocated nodes	Defined by sbatchd
LSB_JOBFILENAME	Submission script	Defined by sbatchd
LSB_JOBID	User job ID	Defined by sbatchd
LSB_HOSTS	List of allocated nodes	Defined by sbatchd
LSB_JOBNAME	User job name	LSF directive

Variable	Value	Origin
LSB_MCPU_HOSTS	List of allocated nodes including number of slots	Defined by sbatchd
LSB_OUTPUTFILE	stdout file	LSF directive
LSB_USER_BIND_CPU_LIST	CPU list for processor affinity	Defined by sbatchd

5.4. QUEUES

The queues constitute the waiting lists that are available to receive the user job submissions.

Each queue has a set of properties which defines, in particular,

- The target compute nodes.
- The maximum number of nodes that can be allocated.
- The maximum number of tasks that can be run on a node.
- The maximum runlimit.

5.5. TASK PLACEMENT & PROCESSOR AFFINITY

It is essential to distinguish the two notions:

- Task placement defines the mechanism for distributing each MPI task onto a given compute node.
- Processor affinity defines the mechanism for pinning each MPI task (and each associated thread) onto one specific CPU (or range of CPUs) inside a given compute node.

5.5.1. TASK PLACEMENT

By default, task placement is performed according to the "group round-robin" policy: each compute node gets as many MPI tasks as specified as number of tasks per node, one compute node after the other.

> Note: The actual task placement can be checked through reading the rankfile of the job. The absolute path of this rankfile is provided by the LSB_DJOB_RANKFILE environment variable.

An alternate task placement can be requested by using the following environment variable: LSB_TASK_GEOMETRY.

This environment variable allows the user to specify the group of MPI tasks that will coexist on the same compute node.

Example : Specifying an Alternate Task Placement through LSB_TASK_GEOMETRY Environment Variable

```
export LSB_TASK_GEOMETRY="{(0,3)(1,4)(2,5)}"
```

The result of such a geometry would be the following:

- MPI Tasks #0 and #3 on node #1
- MPI Tasks #1 and #4 on node #2
- MPI Tasks #2 and #5 on node #3

> Note: The geometry does not define the identify for nodes #1, #2 and #3. This assignment remains under the responsibility of the workload manager.

> Note: It is also possible to specify an alternate task placement by using a mechanism which is specific to the MPI library used for the parallel execution :
>
> - Open MPI: By building a customized rankfile and passing it as an argument to the 'mpirun' command.

- Parallel Environment: By building a customized hostfile and pointing the MP_HOSTFILE environment variable to it.

5.5.2. PROCESSOR AFFINITY

The processor affinity is defined through the 'affinity' directive of the '-R' option. Its general syntax is based on the following architecture:

```
#BSUB -R affinity[“<pu_type>(#):cpubind=<level>:distribute=<policy>”]
```

Sub-Option	Value	Purpose
<pu_type>(#)	numa socket core thread	Type and number of Processor Units assigned to each task
cpubind=<level>	numa socket core thread	Task binding policy
distribute=<policy>	balance pack	Task distribution onto the Processor Units inside a compute node

6. DEBUGGING

6.1. GNU DEBUGGER

The GNU Debugger is installed on the login nodes and on the compute nodes.

The GNU Debugger is useful for:

- Performing interactive debugging - for a single process execution (or on a limited number of processes).
- Analyzing 'core' files generated by an execution that terminated with a memory dump.

Memory Dump 'core' Files Analysis

The following command allows opening a memory dump 'core' file with GDB :

```
gdb <binary> <coredump>
```

In the GDB session, the 'backtrace' command makes it possible to inspect the backtrace of routine calls.

Example : Analysis of a Memory Dump 'core' File (EMMA Application)

```
[login@ouessant ~]$ gdb ~/emma/1.2/bin/emmacpu coredir.0/core.93667
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-80.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "ppc64le-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from ~/emma/1.2/bin/emmacpu...done.
[New LWP 93667]
 [New LWP 93711]
[New LWP 93723]
[New LWP 93681]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/power8/libthread_db.so.1".
Core was generated by `~/emma/1.2/bin/emmacpu param.'.
Program terminated with signal 11, Segmentation fault.
#0  0x00001000279e1b78 in ?? ()
Missing separate debuginfos, use: debuginfo-install glibc-2.17-105.el7.ppc64le libgcc-4.8.5-4.el7.ppc64le
libibverbs-1.1.8mlnx1-OFED.3.1.1.0.0.ppc64le libmlx4-1.0.6mlnx1-OFED.3.1.1.0.0.ppc64le libmlx5-1.0.2mlnx1-
OFED.3.1.1.0.5.ppc64le libnl-1.1.4-3.el7.ppc64le libstdc++-4.8.5-4.el7.ppc64le numactl-libs-2.0.9-
5.el7_1.ppc64le xorg-x11-drv-nvidia-devel-352.59-1.el7.ppc64le
 (gdb) backtrace
#0  0x00001000279e1b78 in ?? ()
#1  0x0000000010033030 in PoissonMgrid (level=-952000048, param=0x10000022b5b0 <MPIDI_SendDoneCB>,
firstoct=0x1000000f250c <MPIR_Barrier_intra+684>, cpu=0x10000032c770, stencil=0x14, stride=-952000048,
tsim=6.72623263e-44)
    at /pwrwork/login/emma/1.2/000008/000000_build/work/EMMA-develop/src/poisson_utils.c:1346
#2  0x0000000010033898 in PoissonSolver (level=-951999664, param=0x10028044442, firstoct=0x3fffc741a2b0,
cpu=0x10000032c770, stencil=0x1000000f21fc <MPIR_Barrier_impl+172>, stride=-951999664, aexp=0)
    at /pwrwork/login/emma/1.2/000008/000000_build/work/EMMA-develop/src/poisson_utils.c:1546
#3  0x0000000010023bac in Advance_level (level=-951998944, adt=0x0, cpu=0x3f8a502100000000,
param=0x405335436a8046bf, firstoct=0xc01eb0a6e0000000, lastoct=0xbff6dc31c0000000, stencil=0x1000006d1370
<_pthread_cleanup_pop_restore>,
    gstencil=0x0, rstencil=0x10009c180010, ndt=0x1000598dbc0, nsteps=-951998944, tloc=0) at
/pwrwork/login/emma/1.2/000008/000000_build/work/EMMA-develop/src/advanceamr.c:442
#4  0x000000001000350c in main (argc=2, argv=0x3fffc7437688) at
/pwrwork/login/emma/1.2/000008/000000_build/work/EMMA-develop/src/emma.c:1949
(gdb)
```

7. PERFORMANCE ANALYSIS & OPTIMIZATION

7.1. MONITORING TOOLS

The following tools are available to perform a monitoring of the compute nodes during execution:

- htop
- nmon

Note: These tools are particularly useful to check the validity of the task placement and its associated processor affinity on the allocated compute nodes.

7.2. PERF

The Linux standard Perf tool makes it possible to perform performance profiling of a specific process.

Perf mainly relies on the following commands:

Command	Option	Purpose
perf annotate		Annotate source code with events
perf list		List supported hardware counters
perf record		Record events
	-e rNNN	Include specified hardware counter
	-g	Include Call Graph
perf report		Display event by process / routine / …
	-g	Include Call Graph

Performance Data Collection

The performance data collection is triggered by the 'perf record' command.

The command can apply to a binary execution :

```
perf record <binary>
```

The command can also apply to a given process designated by its PID:

```
perf record --pid=<pid>
```

Data collection produces an output file named 'perf.data'.

Performance Reporting

The 'perf report' command produces a by-routine performance report:

```
perf report --input perf.data > perf.report
```

The 'perf annotate' command generates line-by-line annotations inside the routines:

```
perf annotate --input perf.data > perf.annotate
```

7.3. OPROFILE

OProfile is an Open-Source project including a statistical tool for performance profiling. OProfile is available inside the Linux distribution itself, or through the 'IBM SDK for LoP' (up-to-date version with full POWER8 support).

OProfile mainly relies on the following commands:

Command	Purpose
ophelp	List available hardware counters
operf	Record events
opreport	Display performance report
opannotate	Annotate source code with events

Performance Data Collection

Performance data collection is achieved through the 'operf' command:

```
[login@ouessant ~]$ operf <binary>
```

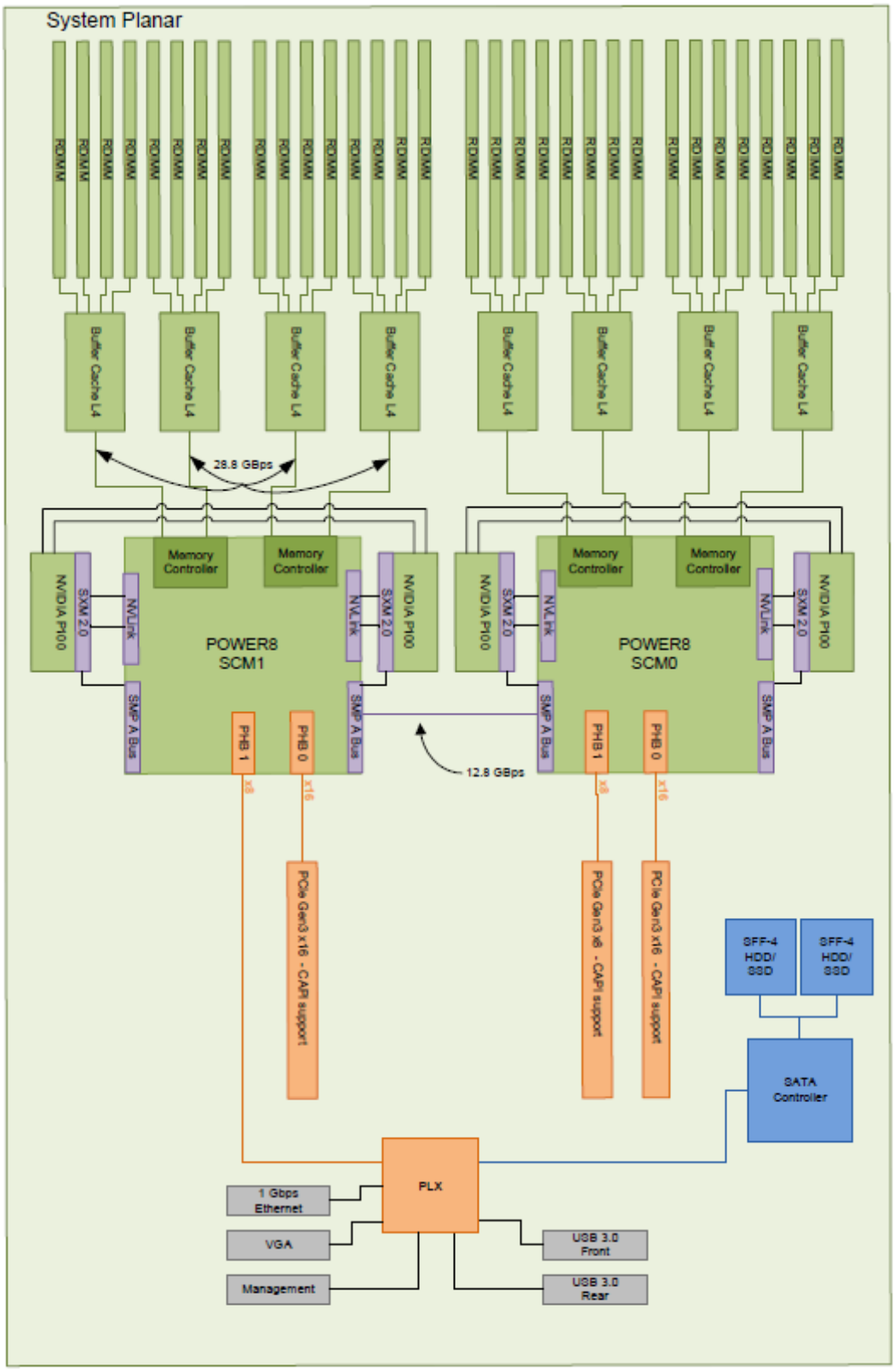
Performance Data Reporting

Performance data reporting is achieved through the 'opreport' command:

```
[login@ouessant ~]$ opreport
CPU_CLK_UNHALT...|
  samples|      %|
------------------
    25202 100.000 cg.W.x
CPU_CLK_UNHALT...|
  samples|      %|
------------------
    18509 73.4426 cg.W.x
     5190 20.5936 libiomp5.so
     1443  5.7257 no-vmlinux
       37  0.1468 ld-2.19.so
       12  0.0476 libifcoremt.so.5
        7  0.0278 libc-2.19.so
        2  0.0079 libpthread-2.19.so
```

8. REFERENCE

8.1. IBM POWER SYSTEM S822LC 'MINSKY' LOGICAL ARCHITECTURE



8.2. IBM SPECTRUM LSF SAMPLE SUBMISSION SCRIPT

Parallel Execution: 2 Compute Nodes, 40 Tasks Total, 20 Tasks / Node

```
#!/bin/bash

#BSUB -cwd /pwrhome/login/helloworld
#BSUB -e /pwrhome/login/helloworld/stderr
#BSUB -J HelloWorld
#BSUB -n 40
#BSUB -o /pwrhome/login/helloworld/stdout
#BSUB -q compute
#BSUB -R "affinity[core(1):cpubind=core:distribute=pack]"
#BSUB -R "span[ptile=20]"
#BSUB -W 00:05

mpiexec /pwrhome/login/helloworld/helloworld
```

9. TIPS & TRICKS

9.1. PHYSICAL CORES / LOGICAL CORES

The 'ppc64_cpu' command displays the relationship between physical cores and logical cores (hardware threads) of the system:

```
[login@ouessant ~]$ ppc64_cpu --info
Core   0:    0*    1*    2*    3*    4*    5*    6*    7*
Core   1:    8*    9*   10*   11*   12*   13*   14*   15*
Core   2:   16*   17*   18*   19*   20*   21*   22*   23*
Core   3:   24*   25*   26*   27*   28*   29*   30*   31*
Core   4:   32*   33*   34*   35*   36*   37*   38*   39*
Core   5:   40*   41*   42*   43*   44*   45*   46*   47*
Core   6:   48*   49*   50*   51*   52*   53*   54*   55*
Core   7:   56*   57*   58*   59*   60*   61*   62*   63*
Core   8:   64*   65*   66*   67*   68*   69*   70*   71*
Core   9:   72*   73*   74*   75*   76*   77*   78*   79*
Core  10:   80*   81*   82*   83*   84*   85*   86*   87*
Core  11:   88*   89*   90*   91*   92*   93*   94*   95*
Core  12:   96*   97*   98*   99*  100*  101*  102*  103*
Core  13:  104*  105*  106*  107*  108*  109*  110*  111*
Core  14:  112*  113*  114*  115*  116*  117*  118*  119*
Core  15:  120*  121*  122*  123*  124*  125*  126*  127*
Core  16:  128*  129*  130*  131*  132*  133*  134*  135*
Core  17:  136*  137*  138*  139*  140*  141*  142*  143*
Core  18:  144*  145*  146*  147*  148*  149*  150*  151*
Core  19:  152*  153*  154*  155*  156*  157*  158*  159*
```

Dans l'exemple ci-dessus, le système est configuré en mode SMT8. Chacun des 20 cœurs physiques ('Core 0' à 'Core 19') comporte donc 8 cœurs logiques (threads matériels).

9.2. PRE-PROCESSING (MACROS) OPTIONS WITH IBM XL FORTRAN

The IBM XL Fortran does not support the standard syntax related to pre-processing options.

With the IBM XL Fortran compiler, it is required to add the '-WF' option to introduce macros, as shown in the example below:

```
xlf90_r -qfree=f90 '-WF,-Dmacro1=1,-Dmacro2' a.F
```

9.3. IBM XL SMP LIBRARY THREAD STACK SIZE

By default, the IBM XL SMP library has a very limited OpenMP stack size value (4 MB).

It is therefore frequent to alter this default value through the following environment variable setting:

```
export XLSMPOPTS="stack=16777216"
```

Note: Stack size must be specified in Bytes.

9.4. IBM SPECTRUM LSF & OPEN MPI INTEGRATION

The proper integration between IBM Spectrum LSF et Open MPI requires:

- Usage of '-rf <rankfile' argument in the Open MPI execution command:

```
mpiexec -rf ${LSB_RANK_HOSTFILE} <Binary>
```

- Reference to a specific application profile when submitting job into Spectrum LSF:

```
bsub -app ompi < job.sh
```

9.5. IBM SPECTRUM LSF INTERACTIVE SESSION

IBM Spectrum LSF allows opening an interactive session onto the compute nodes through the following command:

```
bsub -I -q compute /bin/bash
```

Obviously, the interactive session will wait for the requested resource allocation to be possible before the session can actually open.

Note: Though IBM Spectrum LSF offers this technical possibility, there is no guarantee at this stage that this configuration will be allowed on the Ouessant platform, as this might violate security rules that are part of the IDRIS policy.

10. Additional Documentation

10.1. IBM POWER8

Implementing an IBM High-Performance Computing Solution on IBM POWER8

http://www.redbooks.ibm.com/redbooks/pdfs/sg248263.pdf

Performance Optimization and Tuning Techniques for IBM Processors, including IBM POWER8

http://www.redbooks.ibm.com/redbooks/pdfs/sg248171.pdf

10.2. HPC Software Stack

GNU Compiler Collection

https://gcc.gnu.org/onlinedocs/gcc-6.2.0/gcc/

IBM Spectrum LSF

http://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/

http://www.redbooks.ibm.com/abstracts/redp5048.html?Open

IBM XL C/C++ for Linux

http://www-01.ibm.com/support/knowledgecenter/SSXVZZ/

IBM XL Fortran for Linux

http://www-01.ibm.com/support/knowledgecenter/SSAT4T/

Open MPI

https://www.open-mpi.org/doc/v1.10/

http://www.open-mpi.org/faq/

Parallel Environment Developer Edition (PE DE)

http://www-01.ibm.com/support/knowledgecenter/SSFK5S/

Parallel Environment Runtime Edition (PE RTE)

http://www-01.ibm.com/support/knowledgecenter/SSFK3V/

10.3. Performance Analysis

OProfile

http://oprofile.sourceforge.net/

perf

https://perf.wiki.kernel.org/

http://www.brendangregg.com/perf.html

11. CHANGE LOG

Version 16.02.1

Initial Version

Version 16.02.2

Modification: Task Placement
Modification: Processor Affinity
Modification: Additional Documentation

Version 16.03.1

Addition: IBM Power System S822LC Compute Nodes: Simultaneous Multi-Threading (SMT)
Addition: Performance Analysis: OProfile
Addition: Additional Documentation: POWER8 Links, Perf, OProfile
Modification: Filesystems: Removal /pwrwork
Modification: Filesystems: Home Directories Quota
Modification: Queues
Modification: MPI Wrappers
Modification: Performance Analysis: Perf
Modification: Examples Anonymization: Removal IP Addresses, Usernames, Hostnames

Version 16.05.1

Addition: '-WF' Option for IBM XL Fortran compiler
Addition: IBM Platform LSF & Open MPI Integration
Modification : Compilation Options Equivalences Table

Version 16.10.1

Addition: Minsky Nodes Information
Modification : Translation to English

Version 16.11.1

Addition: GPU Target Specification with PGI Accelerator
Addition: Multiple Tips'n Tricks: IBM XL SMP Thread Stack Size, IBM Spectrum LSF Interactive Sessions
Modification : IBM Power System S822LC Logical Diagram

Version 16.12.1

Modification : Full Refresh Following RHEL 7.3 Migration