DE LA RECHERCHE À L'INDUSTRIE

Slurm at CEA

Site Report

www.cea.fr

SLURM User Group - September 2014 | F. Belot, F. Diakhaté, A. Roy, M. Hautreux

Slurm at CEA
Agenda

Supercomputing projects

Developments evaluation methodology

Xeon Phi accelerators with Slurm

Current activities

Slurm at CEA

Supercomputing projects

Slurm at CEA
Supercomputing projects

TERA

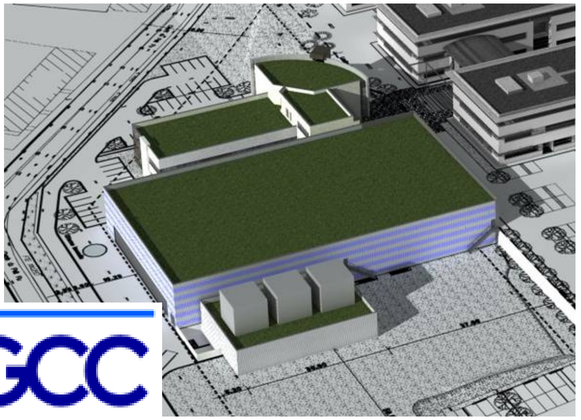
- Project started in 1998
  - Part of the Simulation Project for French Nuclear Deterrence
- **Tera-100** supercomputer
  - Installed in 2010
  - 1,25 PF/s
  - Owned and operated by **CEA**
- Hosted at **CEA** Defense computing center

Slurm at CEA

Supercomputing projects

PRACE (PaRtnership for Advanced Computing in Europe)

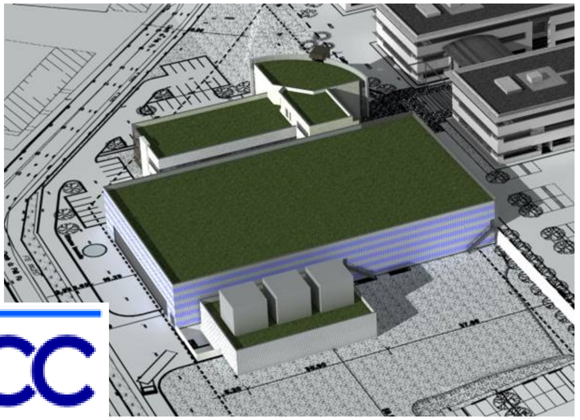
- Project Started in 2007
- **Curie** Supercomputer
  - First French Tier-0 supercomputer for the **PRACE** project
    - 2 stages installation in 2011-2012
    - 1.6 PF/s
  - Owned by **GENCI**
    (Grand Equipement National pour le Calcul Intensif)
  - Operated by **CEA**
- Hosted at the **TGCC**
  « Très Grand Centre de calcul du CEA »
  - CEA computing facility



Slurm at CEA
Supercomputing projects

CCRT (Computing Center for Research and Technology)

- French Industrial and research partners shared computing center
  - Hosted by CEA/DAM/DIF since 2003
- **Airain** Supercomputer
  - CCRT-C machine
    - 3rd phase of the CCRT project, installed in 2012
    - 200 TF/s
  - Operated by **CEA**
- Hosted at the **TGCC** « Très Grand Centre de calcul du CEA »
  - CEA computing facility



Slurm at CEA
Supercomputing projects

TERA+

- CEA PRACE prototypes
  - Connected to the PRACE infrastructure, Grid services and community
- CEA R&D Platform
  - Autonomous computing center reflecting the production systems
- CEA R&D Testbeds
  - Evaluation of novel HW/SW solutions
  - Developments/Improvements of SW stacks

Slurm footprint

- All major clusters introduced since 2009 and operated by CEA
  - Tera+ : fortoy, inti
  - Tera : Tera-100, Visualization cluster
  - PRACE : curie
  - CCRT : airain
- Mostly based on slurm-2.6.x branches
  - Rebuilt with Depth Oblivious FairSharing and Multiple-Slurmd for Xeon Phi

Support

- SLURM supported by the supercomputer vendor for large machines
  - One single vendor for now : BULL
- Level 3 support on the R&D cluster fortoy
  - Provided by SchedMD LLC

Slurm at CEA

Devs evaluation methodology

Slurm at CEA
Devs evaluation methodology

Requirements

- Validate new features and / or bug fixes
  - Multiple versions of Slurm can be targeted
- Validate at scale for targeted production clusters
  - Different Node types : # cores, memory, features, ...
  - Different Cluster Composition : # nodes, node types, ...
  - Different Network Topology : # of levels, width, ...
  - Different accounting schemas
  - ...

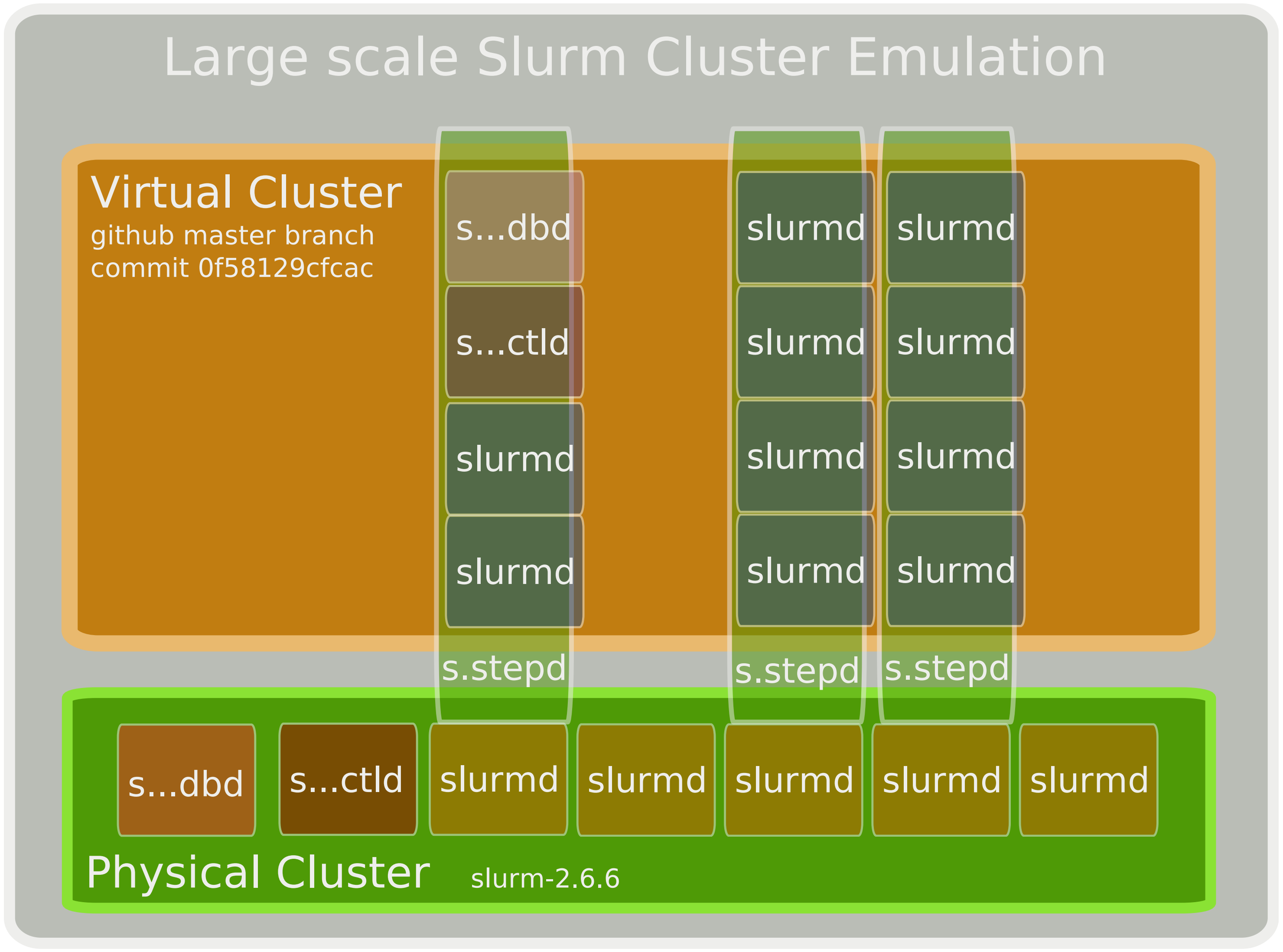
Constraints

- R&D platforms used for other things than just Slurm Dev
  - Can not be broken every time because of Slurm Dev/Validation
- Validation platform coherent with production platforms
  - Still an old Slurm version (2.6.x)

Current validation solution

- A kind of Virtual Slurm Cluster
  - Based on the multiple-slurmd feature
- More precisely a kind of Nested Virtual Slurm Cluster
  - Started inside a classic Slurm Allocation
- Providing a userspace method to validate devs
  - Nested Virtual Slurm Clusters running as standard parallel applications
  - Any user/dev can operate its own validation Slurm Clusters
    - Based on any dev branch and any Slurm versions
  - Can be used for continuous integration with third party tools (jenkins, ...)
- Virtual Cluster sizes only limited by the underlying physical cluster
  - Large physical clusters helping to emulate very(-very) large Virtual Clusters
- Some particular features still hard to validate this way
  - Cpuset/Cgroup cores confinement and affinity requires coherent underlying HW
    - Or virtualized nodes hosting the underlying « physical » Slurm Cluster

Slurm at CEA
Devs evaluation methodology



Slurm Cluster Emulation

- A set of different scripts launched inside a Slurm allocation to
  - Sync a source branch into a temporary directory
  - Compile the source branch (with multiple-slurmd)
    (And install it in a global directory)
  - Create mandatory working directories if necessary
  - Enhance template configuration files based on currently allocated nodes
  - Clean state and log files if requested
  - Start the nested Slurm components
  - Spawn an interactive shell or execute a script accessing the new cluster

Slurm Cluster Emulation Usage

```
[hautreux@leaf50 utilit]$ salloc -n 16 -c 8 --exclusive
salloc: Granted job allocation 413542
[hautreux@leaf50 utilit]$

[hautreux@leaf50 utilit]$ echo $SLURM_NODELIST
leaf[1519-1522,1533-1535,1537-1543,1545,1547]
[hautreux@leaf50 utilit]$
```

Slurm at CEA
Devs evaluation methodology

```
[hautreux@leaf50 utilit]$ ./launcher_dual.sh -h
usage: launcher_dual.sh [-x] [-ISL] [-m ctrl[comp][dbd]] [-i virtual_node_ids]
                         [-b baseport] [-s sources_dir] [-g|G] [-D nodedesc]
                         [-P partdesc] [-C configure_args] [-c conf_dir] prefix

    -x          Isolate slurmctld on its own physical node

    -I          Interactive mode
    -S          Clean state directories
    -L          Clean log directories

    -m mode     define which Slurm components to launch
                ctrl: controller (mandatory)
                comp: compute nodes (optional)
                dbd: slurmdbd (optional)

    -G          Force compilation stage
    -g          Force fast compilation stage (no configure)

[hautreux@leaf50 utilit]$
```

Slurm at CEA

Devs evaluation methodology

```
[hautreux@leaf50 utilit]$ ./launcher_dual.sh -GISL -i 10000-12047 -m ctrlcomp \
                          -c ../cconfs/layouts/ -C "--enable-debug --enable-developer" \
                          -s /scratch/hautreux/slurm-stg/ /scratch/hautreux/sandbox/

[INFO] Syncing sources dir in /scratch/hautreux/sandbox//sources
[INFO] Syncing configuration dir in /scratch/hautreux/sandbox//sources/etc/slurm
[INFO] Configuring, compiling and installing SLURM in /scratch/hautreux/sandbox/
...
[INFO] Cleaning past states from /scratch/hautreux/sandbox//tmp
[INFO] Cleaning past logs from /scratch/hautreux/sandbox//var/log/*
[INFO] Modifying slurm configuration
[INFO] Modifying slurm cgroup configuration
[INFO] Generating slurm topology configuration
[INFO] Launching the simulation cluster
[INFO] Entering interactive mode. Exit to stop the simulation
[hautreux@leaf50 utilit]$ sinfo
PART.      AVAIL  TIMELIM   NODES  STA  NODELIST
physical   up    infinite    16   idle leaf[1519-1522,1533-1535,1537-1543,1545,1547]
virtual*   up    infinite  2048   idle leaf[10000-12047]
[hautreux@leaf50 utilit]$
```

Slurm at CEA

Devs evaluation methodology

```
[hautreux@leaf50 utilit]$ srun -n 2048 -N 2048 hostname | sort | uniq -c
    128 leaf1519
    128 leaf1520
    128 leaf1521
    128 leaf1522
    128 leaf1533
    128 leaf1534
    128 leaf1535
    128 leaf1537
    128 leaf1538
    128 leaf1539
    128 leaf1540
    128 leaf1541
    128 leaf1542
    128 leaf1543
    128 leaf1545
    128 leaf1547
[hautreux@leaf50 utilit]$ exit
[INFO] Ending the simulation
[hautreux@leaf50 utilit]$
```

Slurm at CEA

Xeon Phi accelerators with SLURM

Requirements

- Integrate Nodes with Xeon Phi into existing R&D/Production clusters
  - Bull B515 blades
    - 2 Xeon sockets + 2 Xeon Phi Cards
- Provide access to both Native and Accelerator modes
  - For hybrid usage : Host + Xeon Phi
  - For direct usage : Xeon Phi as standard compute nodes

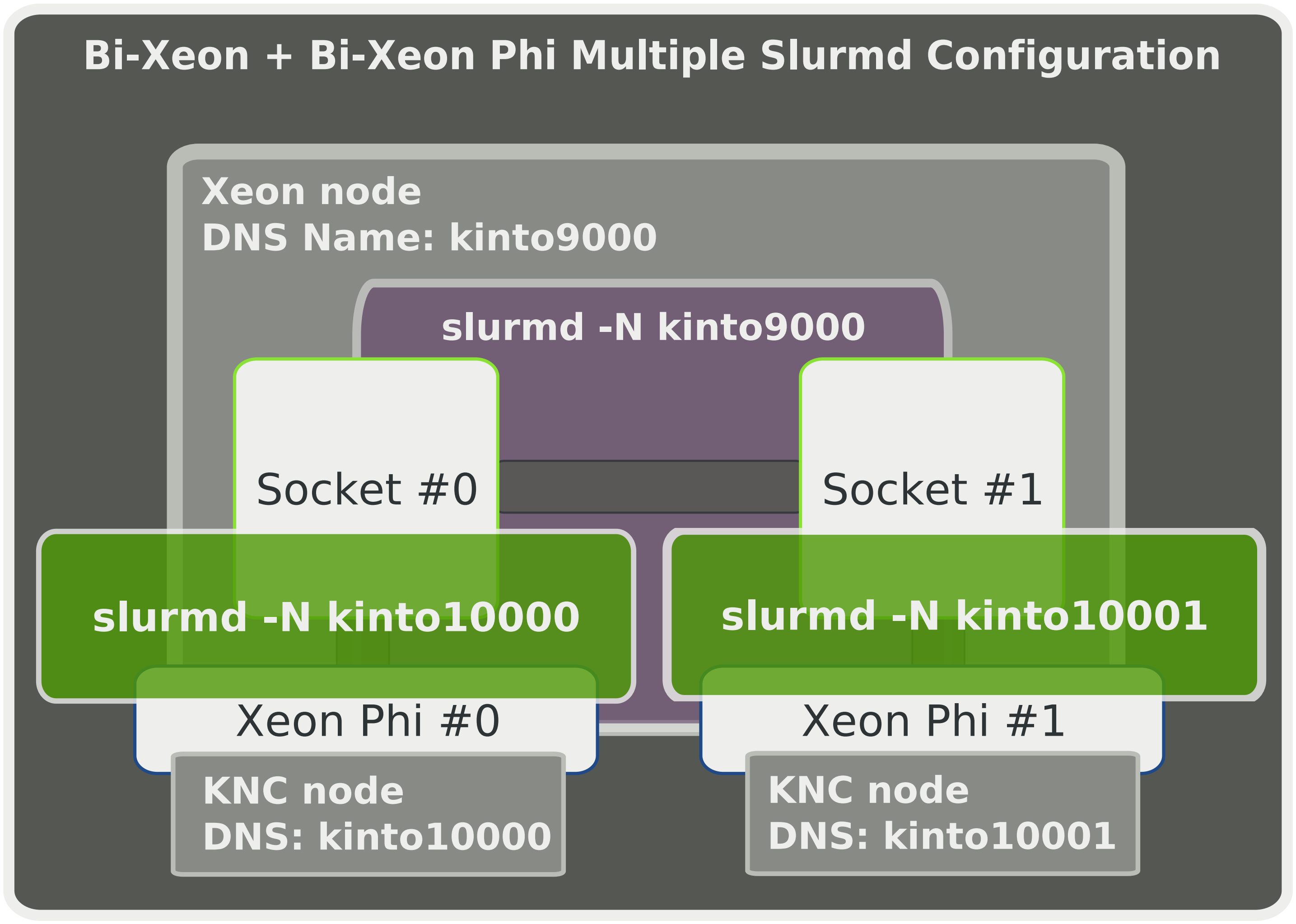
Constraints

- Existing Xeon Phi support in Slurm seems not compatible with our requirements
  - Whether viewed as accelerators like GPUs
  - Or in native mode using a cross-compiled Slurm version

Current selected solution

- A different approach than existing proposals
- Relying on multiple-slurmd
  - Definitely one of our favorite feature of Slurm !
- Allowing to access both modes
  - Using the (salloc+)srun commands to use the hybrid host+accelerator
  - Using the salloc+mpirun (IntelMPI) commands to use the native mode
    - Hydra (Intel Launcher) using ssh to connect to the KNC nodes
- Imposing a split of the original node with 2 Xeon PHI into 2 virtual nodes
  - Corresponding to 2 * (Xeon + Xeon Phi) nodes
- Keeping the original node into the Slurm configuration too
  - But not allowing it to be used in a partition
  - It could however provide the hybrid with 2 Phi mode....
    - But would require to drain the virtual Phi nodes to avoid concurrency

Current selected solution



Solution Usage

```
[hautreux@kinto50 ~]$ grep kinto /etc/slurm/slurm.conf
NodeName=kinto[9000-9001] NodeAddr=kinto[9000-9001] Sockets=2 \
                CoresPerSocket=8 ThreadsPerCore=1 RealMemory=45000 \
                Weight=20 State=UNKNOWN
NodeName=kinto[10000-10001] NodeHostName=kinto9000 \
                Port=[27001-27002] Sockets=1 CoresPerSocket=60 \
                 ThreadsPerCore=4 RealMemory=8000 State=UNKNOWN
NodeName=kinto[10002-10003] NodeHostName=kinto9001 \
                Port=[27001-27002] Sockets=1 CoresPerSocket=60 \
                 ThreadsPerCore=4 RealMemory=8000 State=UNKNOWN
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ sinfo -p knc
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
knc          up 3-00:00:00      2   idle kinto[10000-10003]
[hautreux@kinto50 ~]$
```

Slurm at CEA

Xeon Phi accelerators with SLURM

```
[hautreux@kinto50 ~]$ salloc -p knc -n 1 --exclusive
salloc: Granted job allocation 1043529
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ squeue -u hautreux
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
1043529       knc     bash hautreux  R       0:11      1 kinto10000
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ srun hostname
kinto9000
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ srun cat /proc/self/cgroup
2:freezer:/slurm_kinto10000/uid_123456/job_1043529/step_1
1:cpuset:/slurm_kinto10000/uid_123456/job_1043529/step_1
[hautreux@kinto50 ~]$
```

Slurm at CEA

Xeon Phi accelerators with SLURM

```
[hautreux@kinto50 ~]$ salloc -p knc -n 2 -N 2 --exclusive
salloc: Granted job allocation 1043530
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ srun -l hostname | sort -n
0: kinto9000
1: kinto9000
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ srun -l --cpu_bind=none cat /proc/self/status \
                                | grep Cpus_allowed_list | sort -n
0: Cpus_allowed_list:   0-7
1: Cpus_allowed_list:   8-15
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ srun -l env | grep OFFLOAD | sort -n
0: OFFLOAD_DEVICES=0
1: OFFLOAD_DEVICES=1
[hautreux@kinto50 ~]$
```

Slurm at CEA

Xeon Phi accelerators with SLURM

```
[hautreux@kinto50 ~]$ salloc -p knc -n 240 -N 4
salloc: Granted job allocation 1043612
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ srun -n 4 -N 4 -l hostname | sort -n
0: kinto9000
1: kinto9000
2: kinto9001
3: kinto9001
[hautreux@kinto50 ~]$

[hautreux@kinto50 ~]$ mpirun -np 1 uname -a
Linux kinto10000 2.6.38.8-g5f2543d #2 SMP .... k1om k1om k1om GNU/Linux
[hautreux@kinto50 ~]$ mpirun hostname | sort | uniq -c
     60 kinto10000
     60 kinto10001
     60 kinto10002
     60 kinto10003
[hautreux@kinto50 ~]$
```

Slurm at CEA

Current activities

Slurm at CEA
Current activities

Handle current requirements

- Reservations accounting, Jobs deadline, SELinux Policies...

Be ready for the next generation clusters

- R&D works in collaboration with Bull
  - Detailed in previous presentations (power capping, hierarchical comms, ...)

Evaluate new usages for R&D

- Launch virtual nodes executing Slurm Jobs (PCOCC project)
  - Starting one VM per allocated node
  - Bootstrapping an overlay network to link VMs together (OpenVswitch)
  - Exposing a nested Slurm Cluster for further parallel executions
  - VM based Checkpoint/Restart of the whole “Virtual Private Cluster”

Thank you for your attention

Questions ?