CEA Computing Center

Focus on Tera1000

Using slurm at CEA
CEA/DAM/DIF

Military Application Division of the French Atomic Energy Commission

- Ile de France, Paris Area division of CEA
- Bruyères-le-chatel (30km south of Paris)
- Involved in 3 major HPC projects
CCRT

- French Industrial and research partners shared computing center
- Project started in 1998
- Hosted at TGCC “Tres grand centre de calcul du CEA”
- Cobalt Supercomputer
  - 1422 nodes bi-sockets 28 cores Intel® Xeon® E5 Broadwell
  - 128 Go memory per node
  - InfiniBand EDR interconnect.
GENCI

- Project Started in 2007

- Hosted at TCCC “Tres grand Centre de Calcul du CEA”
  - Owned by GENCI “Grand Equipement National pour le Calcul Intensif”
  - European research project
    PRACE “Partnership for Advanced Computing in Europe”

- Irene Supercomputer
  - 1656 x 24 cores Intel Skylake 8168 bi-sockets nodes
  - 192 GB of DDR4 memory / node
  - InfiniBand EDR interconnect.
  - and
  - 666 x monosocket 68 cores Intel KNL 7250
  - 96 GB memory/node + 16 GB MCDRAM memory/node,
  - Bull eXascale Interconnect (BXI).
TERA

- Hosted at CEA Defense computing Center
- Project started in 1998
  - Part of the Simulation project for French Nuclear Deterrence

- TERA-1: 5 Tflops
- TERA-10: 60 Tflops
- TERA-100: 1000 Tflops
- TERA-1000: 12PFlops
TERA

TERA-1K supercomputer

- Installed between 2015 and 2016
- 2 clusters T1K1.2 and T1K2.2:
  - ~2200 nodes of bi-sockets 32 cores INTEL haswell
  - ~125GB memory per node
  - Infiniband Interconnect

- ~8000 nodes monosocket 68 cores in 29 Bull Sequana with Intel KNL
- ~190GB memory + 16GB MCDRAM per node
- Bull eXascale Interconnect (BXI).
- 12000 TFlops
Focus on TERA1K2.2
Focus on TERA1000

Architecture T1K2.2

- **Computing Islands x 29**
  - ATOS/BULL Sequana “islands” of 276 nodes (8004 nodes) managed by 2 machines named ISMA
  - 30th island contains 24 Skylakes nodes completed with KNL
  - BULL/ATOS BXI Interconnect

- **Service Island**
  - 2 ISMA nodes
  - 16 CEA services nodes
  - 5 service nodes (lustre router, gateway) x 29 (for each compute island)

- **Management Island**
  - 2 management nodes named TOMA for managing total ISMA nodes
Focus on TERA1000

Architecture T1K2.2

- 92 blades: 276 Intel KNL
- 10 GbEthernet
- 5 service nodes for each compute island
- 145 = 29x5: IP gateway, lustre routers
- CEA service nodes
- INTERCONNECT BXI
- SERVICE ISLAND
- MANAGEMENT ISLAND
- COMPUTE ISLAND 1
- COMPUTE ISLAND 16
- COMPUTE ISLAND 15
- COMPUTE ISLAND 29
Focus on TERA1000

Architecture T1K2.2

- ATOS/BULL Sequana (computing island x 29)

- Up to 96 blades x 3 nodes: 288 nodes
- T1K2.2: 276 computing nodes
- 2X service nodes “ISMA”: executing virtual machines managing its island
- 12 L1 switch BXI
- 12 L2 switch BXI
- Hydraulic cooling
Focus on TERA1000

Architecture T1K2.2

- Computing nodes

- Blade of 3 computing nodes with
  - Intel KNL Xeon Phi 6250 x 68 cores
  - 192 GB DDR memory
  - 16GB MCDRAM
  - diskless 15 islands
  - disk full 10 islands
  - 4 islands PLX (shared disks across 3 nodes)
  - Hydraulic cooling
Focus on TERA1000

Architecture T1K2.2

Interconnect ATOS/BULL BXI – 72 L3 switches, 29x12 L2, L1 switches
Where is Slurm on T1K2.2 ?

Slurm on T1K 2.2

- Virtual machines mslurm on ISMA of service island
- HA between the 2 ISMA on service island managed by pacemaker (ATOS/BULL SCS5)
- Resource groups:
  - grp-mslurm (fs-data-mslurm, slurmd, munge)
  - grp-mslurmdbd (data-slurmdbd, mariadb, slurmdbd)

- TCP communications between controller and slurmd nodes are routed through service nodes of each island
Using Slurm at CEA
Using slurm at CEA

Slurm footprint

Versions and support

- All major clusters introduced since 2009 and operated by CEA
  
  TERA : tera-1000 (T1K)
  GENCI : irene
  CCRT : cobalt

- slurm-16.05.x for GENCI and CCRT, slurm-17.11.x for TERA
  Migrating mysql_slurmdbd (size 6.5GB) from 16 to 17 : 1h10mns on virtual machine
  Slurm from SCS5 – ATOS/BULL
  Completed with CEA patches

- SLURM supported by the supercomputer vendor for the 3 HPC projects
  BULL/ATOS
CEA Slurm customization

- CEA uses in production
  - System confinement
  - Feature flags
  - Lua plugins: filesystem licensing, dynamic user's setting, check node health

- CEA studies
  - Lua plugin: requested switch
  - Forward algorithm
  - MPMD
System Confinement

- Operating System noise reduces by its isolation
- Improve Performance of parallel applications
- Mechanism for core specialization:
  - consumable resource required:
    * `SelectType=select/cons_res` in `slurm.conf`
    * `CoreSpecPlugin=core_spec/none` in `slurm.conf` (except for cray)
  - cgroup required:
    * `TaskPlugin=task/cgroup` in `slurm.conf`
    * `ConstrainCores=yes` in `cgroup.conf`
System Confinement

- Nodes definition, in slurm.conf, needs specialized cores
  
  \textit{CPUSpecList=<comma separated list of CPU Ids>}

- Use in conjunction at boot time with kernel parameter isolating cpus from kernel scheduling
  
  \textit{isolcpus=<comma separated list of CPU Ids>}

- Result with 4 specialized cores: ~10% better performance
  
  \textit{srun -core-spec=0 -n 64 -p knl my\_pi\_hybrid\_loop: 64s}
  
  \textit{srun -n 64 -p knl my\_pi\_hybrid\_loop: 58s}
Feature flags

- Uses for nodes validation after maintenance

- Allow to point some nodes in a global node partition instead of creation of multiple partitions

- Nodename=node1 Feature=island,flag2 in slurm.conf

- Use specific flagged nodes in a global partition

  srun --constraint="islandN" -p validation mybinary
Using Slurm at CEA

Filesystems Licensing Lua Plugin

- Interfer with slurm_job_submit setting job_desc.licenses : job_submit.lua

- Allow live maintenance on filesystem by suspending only concerned jobs

- Only new jobs asking for maintained/unsane filesystems will wait

- Simple configuration in slurm.conf

  ```
  Licenses=fs_name_1:10000,fs_name_2:10000
  ```

- Worth precising which filesystem code will use at launch:

  ```
  srun –licenses=fs_name_1 hostname
  ```
Using Slurm at CEA

Dynamic User's settings Lua Plugin

- User can change kernel parameter only for his job: perf.lua
  - Interfer with slurm_spank_init, init_post_opt, job_prolog_epilog
  - `srun --perf_enable`:
    - allow usage of perf tool or vtune by allowing access to hardware processor counters
    ```
    echo > /proc/sys/kernel/perf_event_paranoid
    ```
  - `srun --want_thp`:
    - allow usage of transparent hugepage
    ```
    echo > /sys/kernel/mm/transparent_hugepage/enabled
    ```
- Options are honored only in exclusive mode
Check node health Lua Plugin

- Drain sick nodes according to dmesg found pattern: check-dmesg.lua

  - Interfer with slurm_spank_init, slurm_spank_task_exit

  - Patterns = {"corrupted-journal" = "Error was encountered while opening journal"}

  - Rules = {
    ["corrupted-journal"] = {
      trigger ="count > 0",
      log_info ="Journal is corrupted",
      action="drain_node('%hostname','Corrupted journal')"
    }
  }

- `sinfo -RI`

<table>
<thead>
<tr>
<th>REASON</th>
<th>USER</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrupted journal</td>
<td>slurm</td>
<td>drain</td>
<td>machine</td>
</tr>
</tbody>
</table>
Request switch setting Lua Plugin

- « --req-switches=auto » for avoiding large jobs spreading across fabric switches via job_submit.lua
  - Jobs with big numbers of nodes might be spread across several interconnected switches, preferable to wait for nodes on the same switch availability
  - Interfer with slurm_job_submit setting job_desc.req_switch
  - Description of nodes distribution across switches in slurm_jobsubmit_config.lua
  - Automatic computation and setting of minimum –switch option for asked resources (if –switch not set by user)
  - Waiting time defined in slurm.conf
Forward Algorithm

Current Forward Algorithm

Restart slurmctld: 2 forwards

Restart slurmctld: 1 forward + 15 threads for send

_fwd_tree_thread: example
Treewidth=2
Forward Algorithm

Restart slurmctld: 6 forward + 2 threads send

_example Treewidth=2

_fwd_tree_thread:

Tree sending_msg forward 17 along 18-31 failed
Tree sending_msg forward 18 along 19-24 failed
Tree sending_msg forward 25 along 26-31 succeeded
Tree sending_msg forward 19 along 20-21 failed
Tree sending_msg forward 22 along 23-24 succeeded
_start_msg_tree_internal 20-21

16 nodes 16 nodes
Our need “heterogeneous job” : example IO/PROXY with dedication of nodes with job packs

`salloc -n 3 -N 3 -p compute --mpi_combine=yes : -n 2 -N 2 --mpi_combine=yes -p compute : -N 1 -p IOPRoxy --mpi_combine=no`

`srn run -n 3 --pack-group=0 mpi_binary --args solver1 : -n 2 --pack-group=1 mpi_binary --args solver2`

- **Pack 0**: `mpi_binary --args solver1`  
  + mount `SLURM_NODELIST_PACK2@SLURM_RESV_PORTS_PACK2:/lustre_files_system`

- **Pack 1**: `mpi_binary --args solver 2`

- **Pack 2**: `my_user's I/O SERVICE listening on SLURM_RESV_PORTS_PACK2 (TCP+RDMA)`  
  read/write request from mpi_binary solver1 from pack 0

- **MPI_COMM_WORLD**: rank [0-4]
**MPMD**

So MPMD needs are:

- \(\text{mpi\_combine=\text{yes/no}}\) with both setting across heterogeneous jobs

- SLURM jobs pack environment variables such as:
  \(\text{SLURM\_NODELIST\_PACK\_GROUP\_*, SLURM\_RESV\_PORTS\_PACK\_GROUP\_*}\)

**MPMD tests in 17.11.6**

- Unrecognized option \'\(--\text{mpi\_combine}\)\'
  
  Release notes "Remove srun's --mpi-combine option (always combined)"

- SLURM\_PACK environments: SLURM\_RESV\_PORTS not available

- Will see in 18?
Questions ?
Thank you for your attention