CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Preparing Slurm for use on the Cray XC30

CUG 2013
Napa Valley, California

Stephen Trofinoff and Colin McMurtrie
Swiss National Supercomputing Centre (CSCS)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Agenda

- Port of Slurm to XC30 architecture
- New BASIL v1.3 Interface
- New BASIL v1.3 Features
  - nppcu
  - `QUERY(SUMMARY)`
- Decoupling Slurm from the XC30
- New Task Affinity SPANK plugin

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Slurm

A relatively "simple" open-source resource management system

Three primary SLURM objectives
1. Allocate exclusive/non-exclusive access to resources to users
2. Provide framework for starting, executing and monitoring of work on these allocations
3. Use queues to manage contention

We wrote the Cray port so that
- Slurm sits on top of Cray's ALPS
- Uses the BASIL interface to communicate with ALPS
- BASIL is backwards compatible

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Slurm on the XC30 – Initial Port

XC30 came with a new version of ALPS
- BASIL updated to v1.3

However out initial port was easy
- Took advantage of the backward compatibility
- Used the BASIL 1.2 interface
- Slurm only need modification to a couple of lines

Exposed a bug on the new 2256-node XC30
- Jobs limited to 2047 nodes in size
- Used Slurm XML debugging to help investigate
  - Created by CSCS in 2012 specifically to debug problems of this type
- Result – simply too small a type for a variable
- Fixed – use 32-bit instead of 16-bit integer
- Patch included in Slurm v2.5.5

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Slurm on the XC30 – New BASIL v1.3 Interface

New and restructured XML

- XML hierarchy in `QUERY(INVENTORY)` response reordered
- Two new components
  - `Sockets`
  - `Compute Units`
- Four new elements
  - `SocketArray`
  - `Socket`
  - `ComputeUnitArray`
  - `ComputeUnit`
- ProcessorArray moved down one level
  - Now below `ComputeUnit`
- SegmentArray moved down one level
  - Now below `Socket`
- Some attributes moved to different elements
- New `nppcu` (*number of processors per compute unit*) argument to `QUERY(RESERVE)`

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – nppcu

Implementation of new `nppcu` feature

- nppcu = Number of Processors Per Compute Unit
- Argument to `QUERY(RESERVE)` request
- Corresponds to "`-j`" option to `aprun`
- Controls a job's "view" of the available processors
- `CR_ONE_TASK_PER_CORE` in `slurm.conf`
  - Implies nppcu = 1
- `--ntasks-per-core` job option overrides any default
- Default of 0 (if not using above `slurm.conf` setting)
  - 0 indicates to ignore and use all processors
- Code changes
  - Added `CR_ONE_TASK_PER_CORE` check
  - Added `ntasks-per-core` check
  - Adjustment of calculation of number of nodes
  - Adjusted `mppwidth`

New BASIL v1.3 Features – nppcu (cont.)

Malformed Job Problem

- Required number of processors per node between nppcu and actual node limit
- Job would "slip through the cracks"
  - ALPS can't run the job
  - Slurm doesn't flag them as illegal
  - Slurm backfiller would grind to a halt
- Fix
  - Rewrote one internal Slurm function
  - Adjusted values used for error check
  - Problem solved!

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – nppcu (cont.)

`squeue` reports wrong number of nodes for pending jobs

- Also due to new `nppcu` functionality
- Testing showed this to be a *cosmetic* bug
- `slurmctld` returns
  - number of allocated nodes
  - total number of CPUs (npcus)
- Pending jobs have no "allocated" nodes yet
- Code estimated number of nodes by computing
  - ncpus/(**max** CPUs per node)
- Fix
  - Added similar `nppcu`-based adjustments
  - Problem solved!

- `nppcu` functionality (including bug fixes) will be in Slurm v2.6

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – `QUERY(SUMMARY)`

New `QUERY(SUMMARY)` method

- Attempts to reduce overhead of ever-larger inventory responses
- Provides compact listing of
  - Up/down nodes
  - Up/down accelerators

- Cray's suggested use model
  - Call `Inventory` at startup
  - Subsequently only call `Summary` unless state changes
  - If a state change is detected, call full `Inventory`

- Problem: Does NOT provide job **reservation information**
  - Slurm uses reservation info from `Inventory` as well
  - Any job found in ALPS and NOT in Slurm is "orphan"
  - Slurm requests release of "orphan" jobs
  - No check => potential waste of resources

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – `QUERY(SUMMARY)`(cont.)

Despite limitations we decided to explore possible use anyhow

- Suggested use
  1. Could simply ignore potential "orphans" = bad idea
  2. Could call: `Inventory` 1x, `Summary` Nx, `Inventory` 1x, …
  3. Could replace a certain subset of `Inventory` calls with `Summary`
- Decided to try Option 3
  - Simplicity
  - Time constraints
- Searched code for call paths to `Inventory` invocation
- Multiple callers identified
- However tracing showed two most common paths
  1. *`_attempt_backfill`* – Called when scheduling a job
  2. *`schedule`* – Periodically called

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – `QUERY(SUMMARY)`(cont.)

Resource manager should synchronise before placing new job

- Therefore, we chose the `schedule` path
- Had to distinguish between callers at `Inventory` invocation
  - Due to time constraints, crude hack performed
    - ✧ Apply a mask to a global variable
- Use of new XML `Accelerator` element tag caused some minor issues
  - Another crude hack used
    - ✧ Set static global variable

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – `QUERY(SUMMARY)`(cont.)

Only had time for simple tests

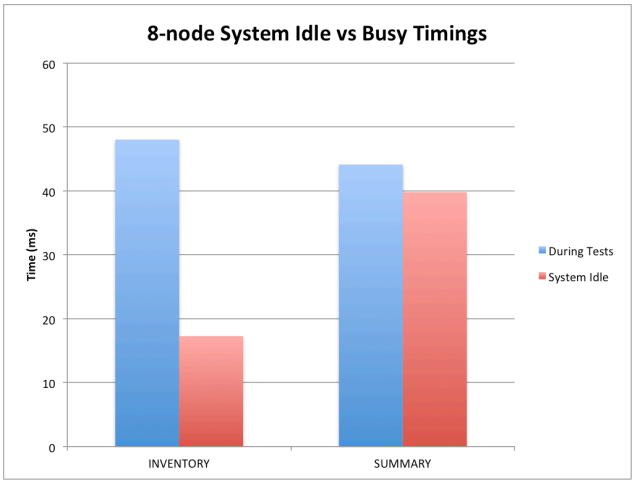
- Used Slurm timing macros to time each BASIL request
- Provided a very rough idea of relative performance
- Timers use standard *`gettimeofday`* system functions
- Ran between 500 and 1500 jobs
  - Various sizes
  - Various wall-clock times
- Ran on both small 8-node and larger 2256-node systems

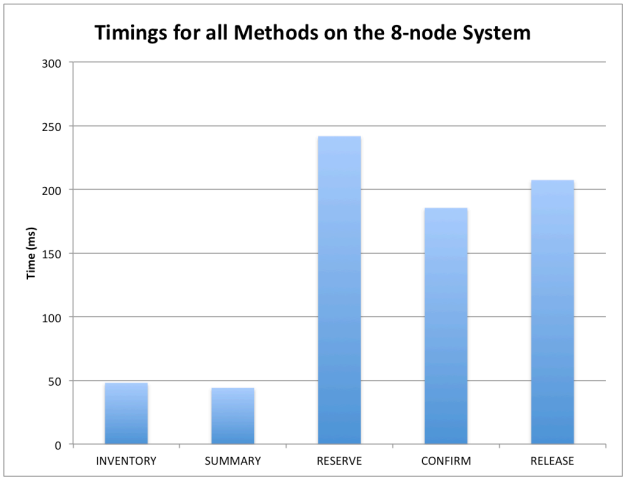
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – `QUERY(SUMMARY)` (cont.)

Results for 8-node system

- If system idle, `Inventory` time approached `Summary`
- Could be simply on small system, less info in `Inventory`
- Could be that no reservations means less work for `Inventory`



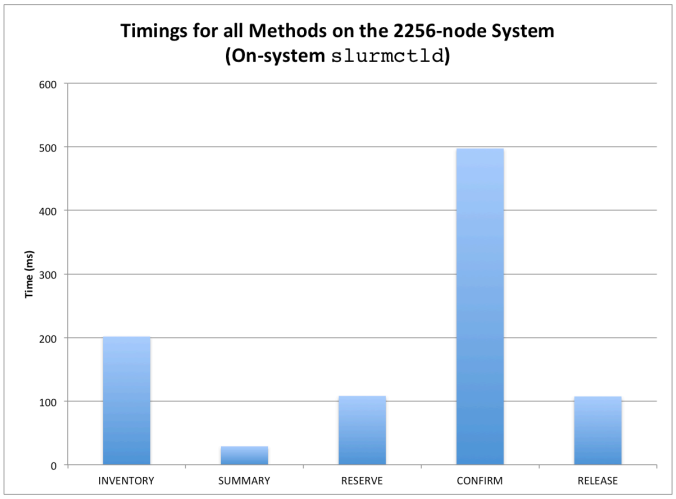


CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – `QUERY(SUMMARY)` (cont.)

Results for 2256-node system

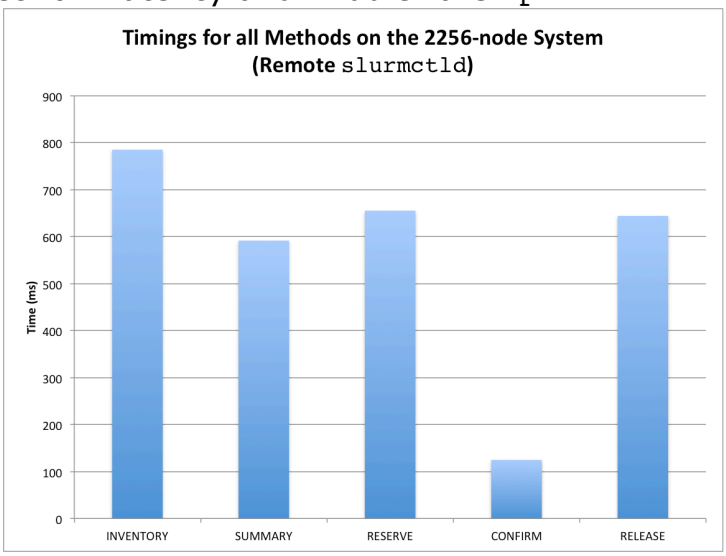
- `slurmctld` on-system
- `Summary` takes much less time than `Inventory`
- `Confirm` takes much longer than `Inventory`
- `Reserve` and `Release` take much longer than `Summary`



New BASIL v1.3 Features – QUERY(SUMMARY)(cont.)

Results for 2256-node system

- `Slurmctld` off-system
- All requests now take hundreds of milliseconds
  - Network latency and middleware `apbasil`



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New BASIL v1.3 Features – QUERY(SUMMARY)(cont.)

QUERY(SUMMARY) conclusions

- Results a bit mixed
- Does not provide all the functionality of `Inventory`
- Appears to dramatically reduce time consumed by `Inventory`
  - Needs more in-depth analysis
  - Potential speed up would warrant some further exploration
- Tricky to implement
- Would be nice to have a complimentary method such as *`QUERY(RESVSUMMARY)`*

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Decoupling Slurm from the XC30

Goals

- Attempt to free up frontend resources
  - On-system these are selected Service Nodes
- Slurm stays up when the main system is down
  - Presents persistent interface to users
  - Users will already be using esLogin nodes which are also decoupled from the main system

Relatively easy to implement

- Fully qualify DNS names in the `slurm.conf`
- Various paths in `slurm.conf` had to be checked to see if they made sense
- Same state directory must be mounted
  - On the esLogin nodes
  - On XC30 service nodes where daemons run

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Decoupling Slurm from the XC30 (cont.)

Most significant change

- Needed to write intermediary `apbasil` to pass communication between remote `slurmctld` and `apbasil` on main system
  - Relatively small
  - Currently a few hundred lines of C code
  - Able to reuse some of the Slurm pipe code (`popen2`)
- Work is on-going
  - Need to confirm persistence of interface when main system goes down

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New Task Affinity SPANK plugin

Slurm provides an elegant interface to enhance functionality
- *Slurm Plugin Architecture for Node and Job (K)control* or SPANK
- Plugins are stackable and easy to administer
- Easy API

Internal request from User Support
- Wanted similar task affinity mappings as ALPS provides
- Slurm can use affinity masks but these deemed unsuitable
- Created new affinity module
  - Based on an older one from LLNL

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New Task Affinity SPANK plugin (cont.)

New *`reduced-auto-affinity`* module has the following binding pattern
- Unique fat mask for each task of each node of a job
- Each fat mask has one processor per software thread of the task
- Each processor assigned to a task will be adjacent to each other
- Only one processor per core used
  - Effectively binding at core level
- Each task confined to one socket
- Multiple tasks can share a socket if all of them completely fit
- Any violation of this policy causes job to be rejected

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

New Task Affinity SPANK plugin (cont.)

Example of use:

- Given a 2x8x2 node layout
- Create a batch job file (`job.sh`) as follows:

```
#!/usr/bin/ksh
#SBATCH --n 4
#SBATCH --N 1
#SBATCH --cpus-per-task=4

srun --reduced-auto-affinity=on,v --cpu_bind=no my.exe
```

- To run the command line would simply be:

```
$ sbatch job.sh
```

```
reduced-auto-affinity: local task 0: CPUs: 0-3
reduced-auto-affinity: local task 2: CPUs: 8-11
reduced-auto-affinity: local task 1: CPUs: 4-7
reduced-auto-affinity: local task 3: CPUs: 12-15
```


CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Q&A

E-mail: trofinoff@cscs.ch
colin@cscs.ch