ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

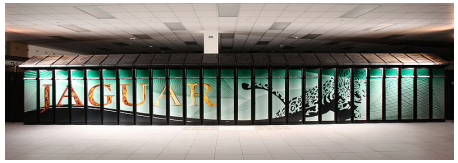
# Porting SLURM to the Cray XT and XE

Neil Stringfellow and Gerrit Renker

# Background

# Cray XT/XE basics

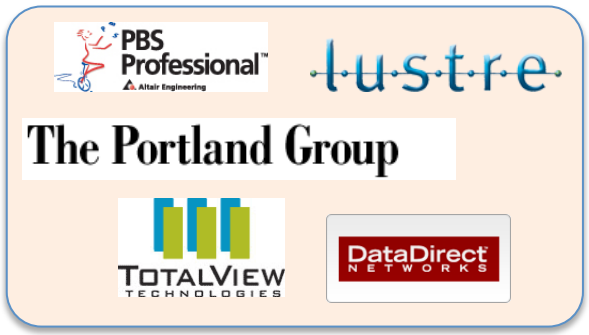
- Cray XT systems are among the largest in the world
  - 9 out of the top 30 machines on the top500 list June 2010 are XTs
  - Number 1 system on top500 June 2010 is a Cray XT5 installation
  - New Cray XE line is expected to maintain this position for Cray
- Machines consist of a number of diskless compute nodes and a set of service nodes for external connectivity
- All nodes run Linux
  - Cray runs a lightweight version of Linux on the compute nodes
- Nodes are connected with a proprietary interconnect in a 3D torus configuration
  - Small systems may be connected in a 2D torus or mesh
  - *All nodes are part of the torus, both compute nodes and service nodes*






# The Cray XT series – in the beginning …

- The first Cray XT3 systems were delivered in 2005 with a strictly defined set of hardware and software
  - DDN disk controllers for I/O
  - Lustre file system
  - PGI compilers
  - Totalview debugger (optional)
  - PBSPro batch system



# The Cray XT series – in the beginning …

- The first Cray XT3 systems were delivered in 2005 with a strictly defined set of hardware and software
  - DDN disk controllers for I/O
  - Lustre file system
  - PGI compilers
  - Totalview debugger (optional)
  - PBSPro batch system
- Over time the hardware and software ecosystem has diversified
  - DDN/LSI/Panasas file servers/systems

# The Cray XT series – in the beginning …

- The first Cray XT3 systems were delivered in 2005 with a strictly defined set of hardware and software
  - DDN disk controllers for I/O
  - Lustre file system
  - PGI compilers
  - Totalview debugger (optional)
  - PBSPro batch system
- Over time the hardware and software ecosystem has diversified
  - DDN/LSI/Panasas file servers/systems
  - PGI/Pathscale/GNU/Intel/Cray compilers

# The Cray XT series – in the beginning …

- The first Cray XT3 systems were delivered in 2005 with a strictly defined set of hardware and software
  - DDN disk controllers for I/O
  - Lustre file system
  - PGI compilers
  - Totalview debugger (optional)
  - PBSPro batch system
- Over time the hardware and software ecosystem has diversified
  - DDN/LSI/Panasas file servers/systems
  - PGI/Pathscale/GNU/Intel/Cray compilers
  - Totalview/DDT debuggers

# The Cray XT series – in the beginning …

- The first Cray XT3 systems were delivered in 2005 with a strictly defined set of hardware and software
  - DDN disk controllers for I/O
  - Lustre file system
  - PGI compilers
  - Totalview debugger (optional)
  - PBSPro batch system
- Over time the hardware and software ecosystem has diversified
  - DDN/LSI/Panasas file servers/systems
  - PGI/Pathscale/GNU/Intel/Cray compilers
  - Totalview/DDT debuggers
  - PBSpro/Torque-Moab/LSF batch systems

# The Cray XT series – in the beginning …

- The first Cray XT3 systems were delivered in 2005 with a strictly defined set of hardware and software
  - DDN disk controllers for I/O
  - Lustre file system
  - PGI compilers
  - Totalview debugger (optional)
  - PBSPro batch system
- Over time the hardware and software ecosystem has diversified
  - DDN/LSI/Panasas file servers/systems
  - PGI/Pathscale/GNU/Intel/Cray compilers
  - Totalview/DDT debuggers
  - PBSpro/Torque-Moab/LSF batch systems
- *SLURM is missing from this list*

# Cray history at CSCS

- CSCS runs a total of 6 different Cray systems
  - 3 production machines (one XT5 and two XT4)
  - 3 test and development machines (XE6, XT5, XT3)
- Cray XT systems have been in CSCS for five years
  - A 1100 core Cray XT3 machine was delivered in June 2005
- Cray XT3 machines ran the Catamount microkernel on the compute nodes with “yod” as the job launcher
- Early systems were delivered with PBSpro and a *primitive* scheduler
  - Base scheduler provided with systems in 2005 was unsuitable for batch work on large machines

# Building on the base PBSpro distribution …

- Cray (Jason Coverston) wrote a scheduler in Tcl to plug in to PBSpro
  - Had a simple backfilling algorithm
  - Provided advanced reservations
  - Had a simple formula for priority calculation
  - Was the base on which all subsequent scheduling algorithms were added
  - Tcl was chosen based on success at Pittsburgh Supercomputing Centre

# Building on the base PBSpro distribution …

- Cray (Jason Coverston) wrote a scheduler in Tcl to plug in to PBSpro
  - Had a simple backfilling algorithm
  - Provided advanced reservations
  - Had a simple formula for priority calculation
  - Was the base on which all subsequent scheduling algorithms were added
  - Tcl was chosen based on success at Pittsburgh Supercomputing Centre
- CSCS added to the scheduler over time, to produce a very efficient scheduling policy
  - Large jobs run with high priority and little delay
  - System utilisation is over 90%
  - Has special features that are required for operational weather forecasting

# Building on the base PBSpro distribution …

- Cray (Jason Coverston) wrote a scheduler in Tcl to plug in to PBSpro
  - Had a simple backfilling algorithm
  - Provided advanced reservations
  - Had a simple formula for priority calculation
  - Was the base on which all subsequent scheduling algorithms were added
  - Tcl was chosen based on success at Pittsburgh Supercomputing Centre
- CSCS added to the scheduler over time, to produce a very efficient scheduling policy
  - Large jobs run with high priority and little delay
  - System utilisation is over 90%
  - Has special features that are required for operational weather forecasting
- Ported to XT4/ALPS in 2008:
  - Tcl works great for custom backfilling
  - Scripting proved a great asset of PBS
  - … but ALPS problems created downtime

# Building on the base PBSpro distribution …

- Cray (Jason Coverston) wrote a scheduler in Tcl to plug in to PBSpro
  - Had a simple backfilling algorithm
  - Provided advanced reservations
  - Had a simple formula for priority calculation
  - Was the base on which all subsequent scheduling algorithms were added
  - Tcl was chosen based on success at Pittsburgh Supercomputing Centre
- CSCS added to the scheduler over time, to produce a very efficient scheduling policy
  - Large jobs run with high priority and little delay
  - System utilisation is over 90%
  - Has special features that are required for operational weather forecasting
- Ported to XT4/ALPS in 2008:
  - Tcl works great for custom backfilling
  - Scripting proved a great asset of PBS
  - … but ALPS problems created downtime
- CSCS has a source code licence for PBSpro
  - Allows for fixes if the distribution is broken
  - … but patches supplied back to Altair might not be included in the next release

# Motivation for change …

- Mid-2010 Altair announced that they will no longer be supporting Tcl plugin
  - Could convert to Python plugin, but opportune moment to consider alternatives
- Many other Cray sites use Torque/Moab
  - Brief testing was not particularly successful at CSCS
  - Torque has the same background as PBSpro
- *CSCS will make its next major procurement in 2012, so we need to look at the best options in this timeframe for whatever architecture we choose*

# Selection of SLURM

- We considered SLURM to be the best candidate to look at for the future
  - Designed for large scale systems from the outset (and CSCS current XT5 systems is reasonably large)
  - In use on a variety of large systems around the world
    - Investment in SLURM will reap benefits whichever large machine we chose in the future
  - Can be deployed on all current systems at CSCS
    - We would like to have a common batch environment for all our machines
  - Open source
  - Responsive and active developers
- *Cray itself and several Cray customers have expressed interest*

  … so …
- We have been experimenting with SLURM since spring 2010
- First working port implemented early June 2010

# Cray systems at CSCS


- We have a set of 3 production systems
  - *rosa* – XT5, 20 cabinets, 12 core, PBSpro
    - 22,128 core machine, national service for academic research

# Cray systems at CSCS


- We have a set of 3 production systems
  - *rosa* – XT5, 20 cabinets, 12 core, PBSpro
    - 22,128 core machine, national service for academic research
  - *buin* – XT4, 3 cabinets, 4 core, PBSpro
    - 1,040 core machine, production system for operational weather forecasting


# Cray systems at CSCS

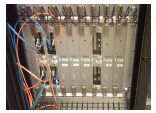

- We have a set of 3 production systems
  - *rosa* – XT5, 20 cabinets, 12 core, PBSpro
    - 22,128 core machine, national service for academic research
  - *buin* – XT4, 3 cabinets, 4 core, PBSpro
    - 1,040 core machine, production system for operational weather forecasting
  - *dole* – XT4, 2 cabinets, 4 core, PBSpro
    - 688 core machine, failover system for weather forecasters


# Cray test systems at CSCS


- We have 3 test and development systems
- *palu* – XE6, 2 cabinets, 24 core per node, slurm
  - Newest CSCS machine
  - Has 4224 compute cores and 5.6 terabytes of memory
  - First XE6 system shipped out of Cray
  - Now considered a *production-development* machine

# Cray test systems at CSCS

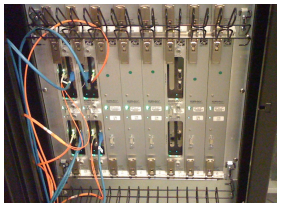

- We have 3 test and development systems
- *palu* – XE6, 2 cabinets, 24 core per node, slurm
  - Newest CSCS machine
  - Has 4224 compute cores and 5.6 terabytes of memory
  - First XE6 system shipped out of Cray
  - Now considered a *production-development* machine
- *gele* – XT5, 1 chassis, 12 core per node, slurm
  - A single chassis version of the large XT5 Rosa
- *fred* – XT3, 1 chassis, dual core node, slurm
  - Test system for the Meteo machines



# ALPS and BASIL

# Overview of Cray job placement architecture

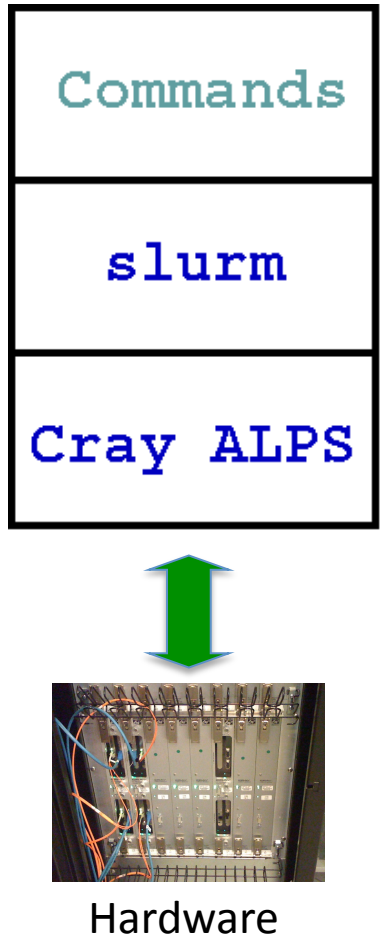
- No *direct access* to hardware



Hardware

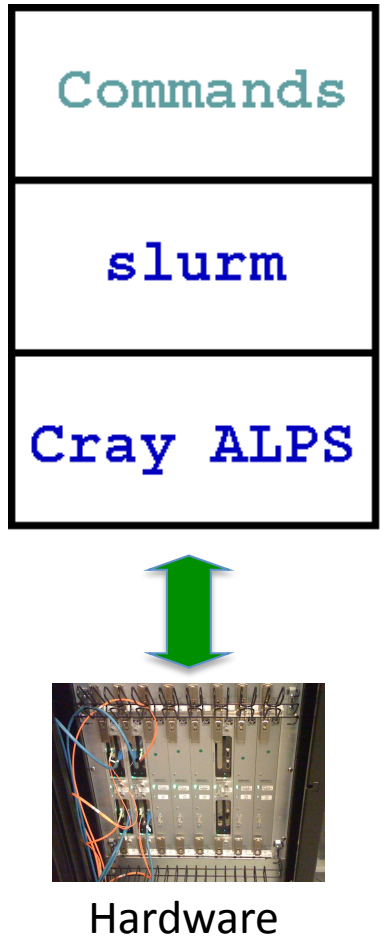
# Overview of Cray job placement architecture

- No *direct access* to hardware
- Cray provides ALPS *interface*
  - allocates resources
  - house-keeping
  - node health checking



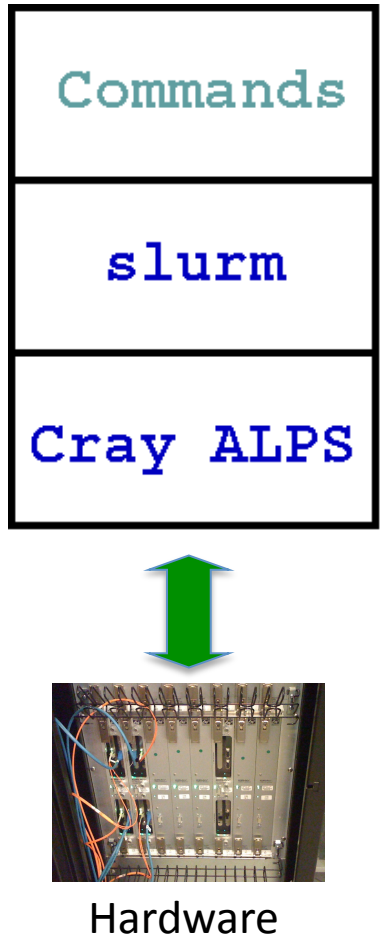
# Overview of Cray job placement architecture

- No *direct access* to hardware
- Cray provides ALPS *interface*
  - allocates resources
  - house-keeping
  - node health checking
- ALPS is the whole system
  - Application Level Placement Scheduler
  - 6 daemons plus supporting programs
  - Single daemon running on each compute node



# Overview of Cray job placement architecture

- No *direct access* to hardware
- Cray provides ALPS *interface*
  - allocates resources
  - house-keeping
  - node health checking
- ALPS is the whole system
  - Application Level Placement Scheduler
  - 6 daemons plus supporting programs
  - Single daemon running on each compute node
- Basil is the XML interface used by ALPS
  - Batch Application Scheduler Interface Layer



# ALPS/Basil layer 1/2

- XML-RPC interface with 4 methods
  - XML query via stdin to /usr/bin/apbasil
  - XML reply from stdout of apbasil

# ALPS/Basil layer 1/2

- XML-RPC interface with 4 methods
  - XML query via stdin to /usr/bin/apbasil
  - XML reply from stdout of apbasil
- QUERY - INVENTORY
  - lists compute nodes (service nodes not listed)
  - node attributes (cpus, mem, state, allocation)
  - lists current ALPS resource allocations

# ALPS/Basil layer 2/2

- RESERVE - allocate resources
  - allocate by mpp{width,depth,nppn} (PBS)
  - or use explicit nodelist (slurm)
  - only Basil 3.1 lists nodes selected by ALPS

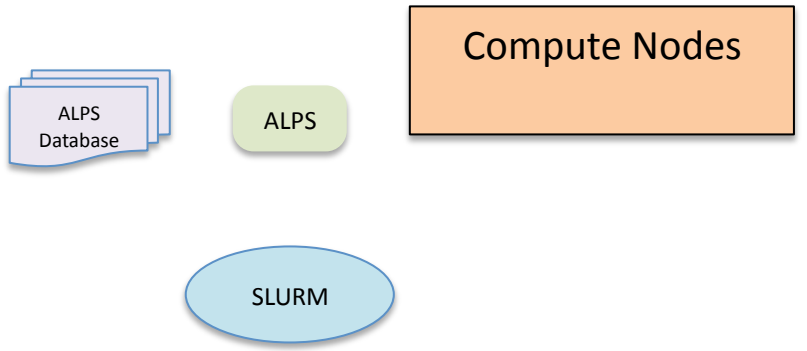
# ALPS/Basil layer 2/2

- RESERVE - allocate resources
  - allocate by mpp{width,depth,nppn} (PBS)
  - or use explicit nodelist (slurm)
  - only Basil 3.1 lists nodes selected by ALPS
- CONFIRM - commit allocation
  - must happen ca. 120 seconds after RESERVE
  - to clear state if anything hangs or crashes

# ALPS/Basil layer 2/2

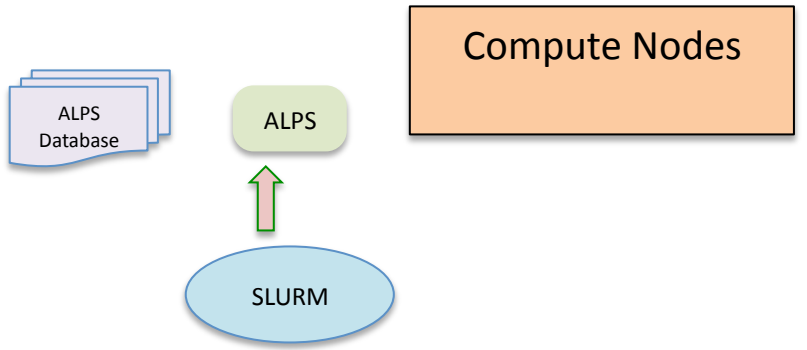
- RESERVE - allocate resources
  - allocate by mpp{width,depth,nppn} (PBS)
  - or use explicit nodelist (slurm)
  - only Basil 3.1 lists nodes selected by ALPS
- CONFIRM - commit allocation
  - must happen ca. 120 seconds after RESERVE
  - to clear state if anything hangs or crashes
- RELEASE - orderly return of resources

# ALPS Job launch



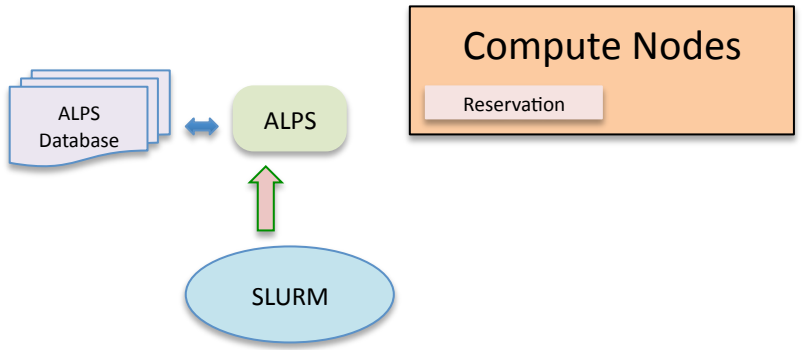
# ALPS Job launch

- Batch system makes a request for resources from ALPS



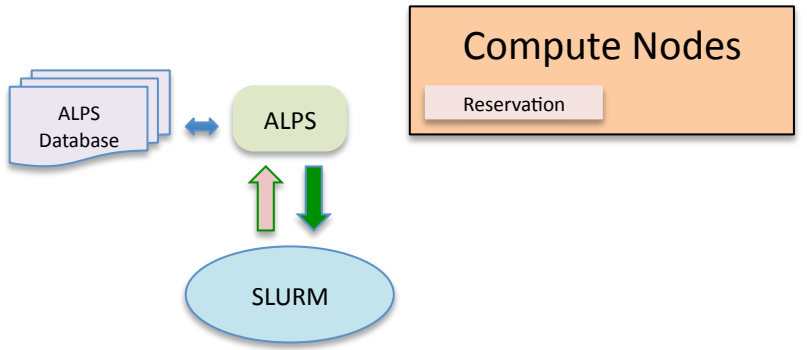
# ALPS Job launch

- Batch system makes a request for resources from ALPS



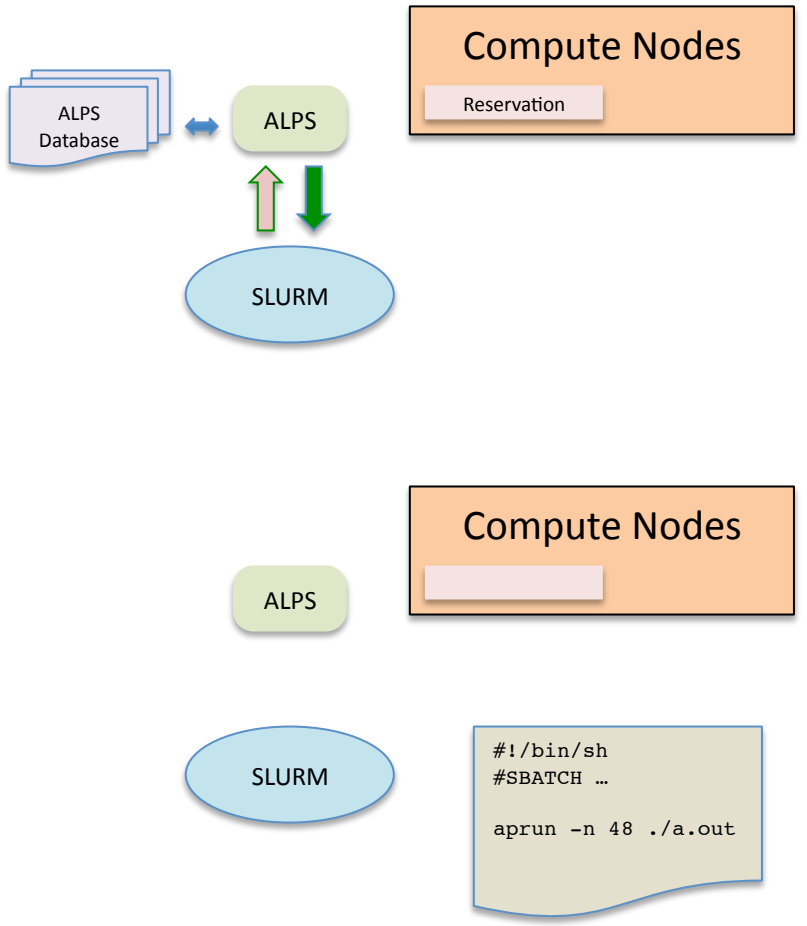
# ALPS Job launch

- Batch system makes a request for resources from ALPS
- Confirmed resource reservation allows job to run



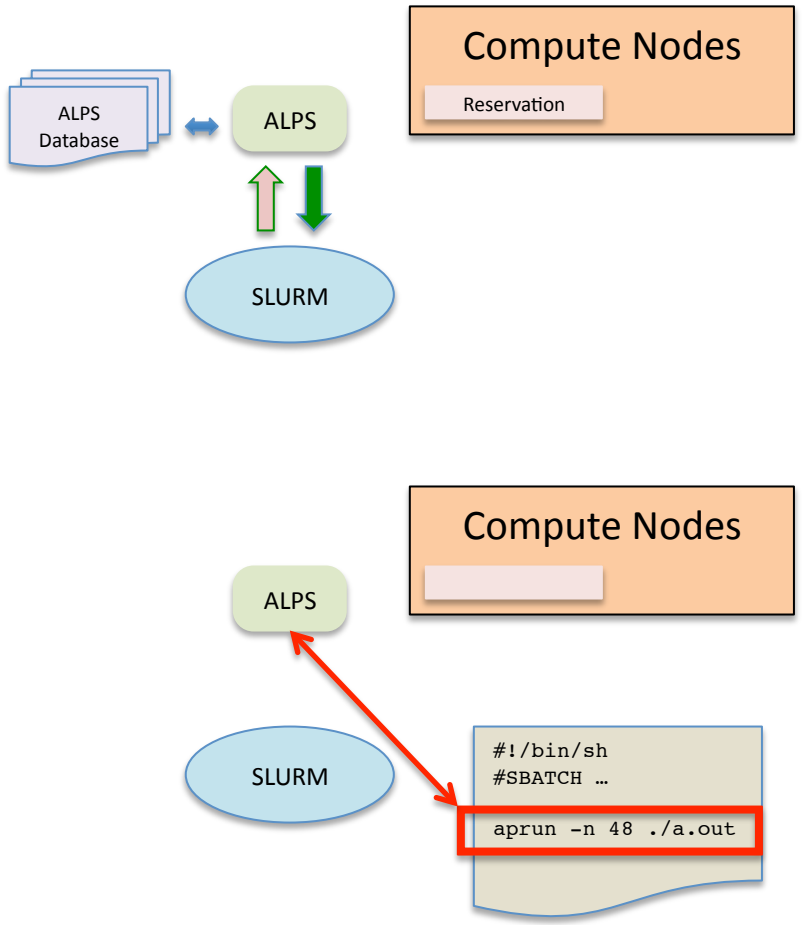
# ALPS Job launch

- Batch system makes a request for resources from ALPS
- Confirmed resource reservation allows job to run
- SLURM scripts are actually running on Cray service nodes



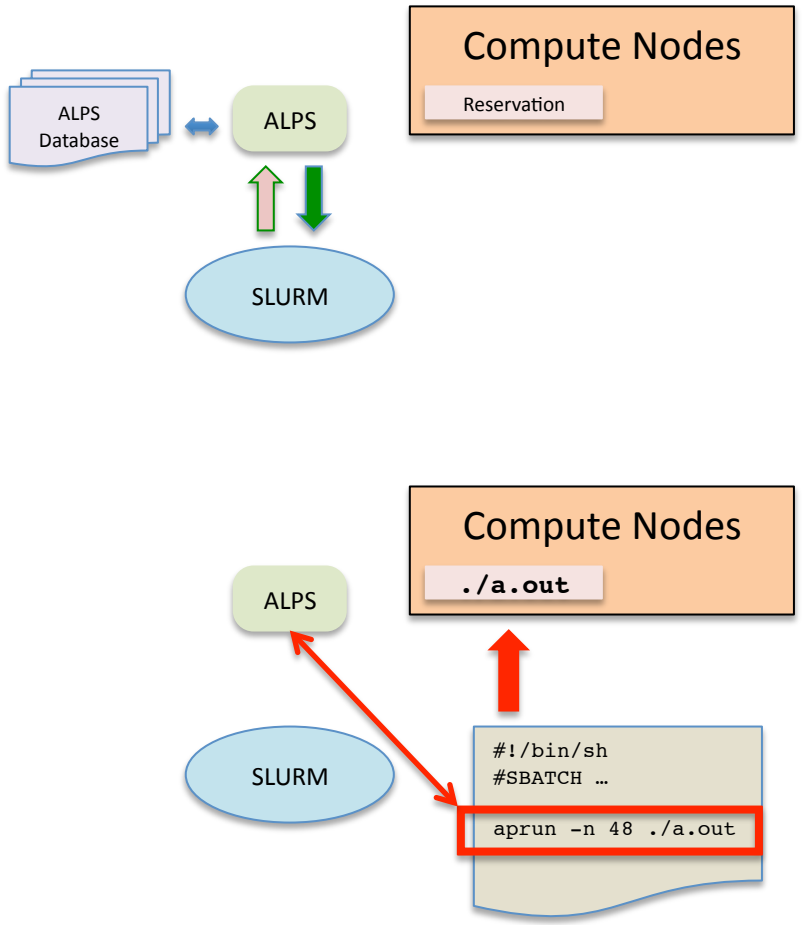
# ALPS Job launch

- Batch system makes a request for resources from ALPS
- Confirmed resource reservation allows job to run
- SLURM scripts are actually running on Cray service nodes
- Batch systems only manage allocations through ALPS layer



# ALPS Job launch

- Batch system makes a request for resources from ALPS
- Confirmed resource reservation allows job to run
- SLURM scripts are actually running on Cray service nodes
- Batch systems only manage allocations through ALPS layer
- Within job, aprun launches and manages the execution of the application



# Current Status

# Cray restrictions/complications

- Cray nodes cannot be shared between jobs
  - due to an ALPS aprun restriction
  - hence select/linear (by node) allocation
  - no select/cons_res in foreseeable future

# Cray restrictions/complications

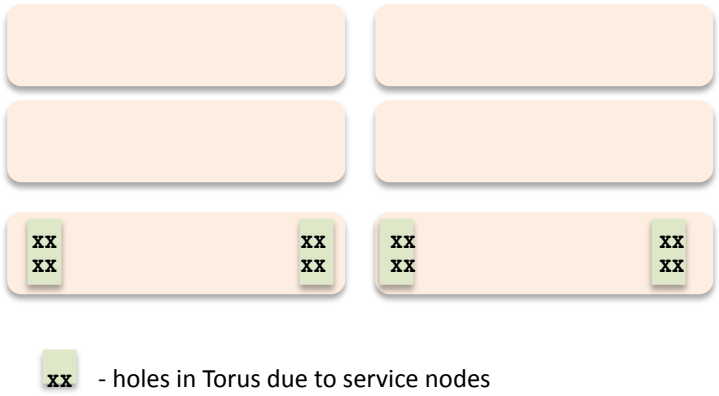
- Cray nodes cannot be shared between jobs
  - due to an ALPS aprun restriction
  - hence select/linear (by node) allocation
  - no select/cons_res in foreseeable future
- Hardware range keeps expanding
  - SeaStar / Gemini ASICs (also cabling)
  - 4 SeaStar cabling variants for 2D/3D tori

# Cray restrictions/complications

- Cray nodes cannot be shared between jobs
  - due to an ALPS aprun restriction
  - hence select/linear (by node) allocation
  - no select/cons_res in foreseeable future
- Hardware range keeps expanding
  - SeaStar / Gemini ASICs (also cabling)
  - 4 SeaStar cabling variants for 2D/3D tori
- Torus is unlike BlueGene
  - may have odd dimensions (e.g. 1 x 12 x 16)
  - always holes in the torus due to service nodes
    - The Torus itself is complete with communication ASICs, but some ASICs are not connected to compute nodes
  - E.g. CSCS Cray XT5 Rosa is a 16x10x12 Torus with holes

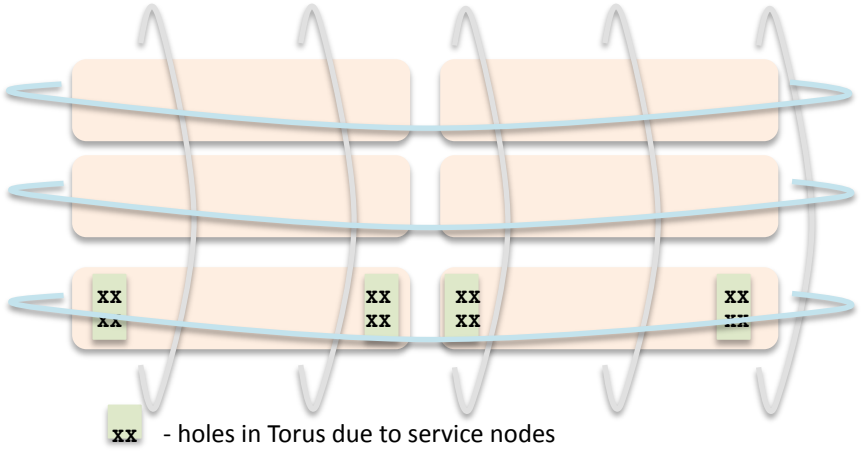
# E.g. 2D Torus with holes

- Our new Cray XE6 “palu” has a 2D torus topology
  - Nodes are arranged in a 16x12 grid
  - Two nodes with two NICs share a Gemini ASIC connection to the torus
  - Gemini communication topology is 16x6



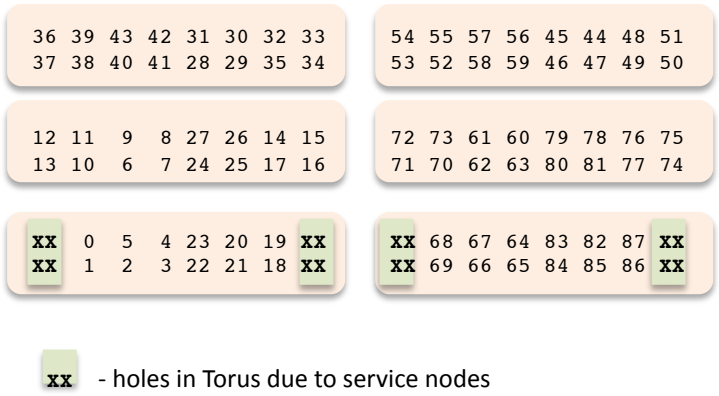
# E.g. 2D Torus with holes

- Our new Cray XE6 “palu” has a 2D torus topology
  - Nodes are arranged in a 16x12 grid
  - Two nodes with two NICs share a Gemini ASIC connection to the torus
  - Gemini communication topology is 16x6



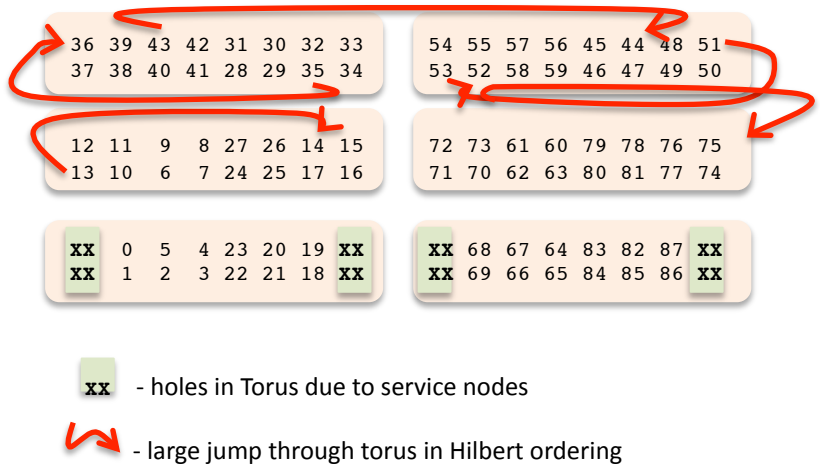
# E.g. 2D Torus with holes

- Our new Cray XE6 “palu” has a 2D torus topology
  - Nodes are arranged in a 16x12 grid
  - Two nodes with two NICs share a Gemini ASIC connection to the torus
  - Gemini communication topology is 16x6
- Hilbert curve ordering algorithm used in Slurm does not work well with this type of topology



# E.g. 2D Torus with holes

- Our new Cray XE6 “palu” has a 2D torus topology
  - Nodes are arranged in a 16x12 grid
  - Two nodes with two NICs share a Gemini ASIC connection to the torus
  - Gemini communication topology is 16x6
- Hilbert curve ordering algorithm used in Slurm does not work well with this type of topology




We could not fully think of it as a Blue Gene

# CSCS libasil library

- abstraction layer for Basil 1.0/1.1/1.2/3.1 XML
  - 3 parsers to handle version differences
  - switches to highest-supported Basil version
  - adapts parameter passing to current Basil version
- talks to SDB database to
  - resolve (X, Y, Z) coordinates
  - distinguish Gemini/SeaStar
- currently still external (--with-alpslib=...)
  - could it be shipped under 'contribs/'?
  - otherwise will be merged into select/cray

# Working status

- We have slurm working on 2 XT and 1 XE system
  - Over 15,000 jobs have been run so far
- All test systems have 2D torus
  - looking for collaborators to test on 3D
  - expect different node ranking
- Runs in frontend mode with single slurmd
  - hence srun disabled
  - use sbatch/salloc + aprun/mpirun
- OpenMPI supported on XT
  - now supports BASIL_RESERVATION
  - XE support still waiting to be done

# Working status

- select/cray plugin thanks to Danny
  - defers to select/linear for actual work
  - is being populated with ALPS-specific code
- Most salloc/sbatch options supported
  - mapped into corresponding ALPS parameters
  - --mem, -n, -N, --ntasks-per-node, --cpus-per-task
  - unusable options blocked (--share / --overcommit)
- An ALPS inventory every HealthCheckInterval

# Node ordering

- ALPS uses its own ALPS_NIDORDER ranking
  - database ordering - can be reconfigured
  - slurm picks up this ordering for select/linear
  - but reconfiguration requires ALPS/slurm restart
- Node ranking generalizes Hilbert ordering
  - dynamically “plug in” different orderings
  - ranking done by topology or node selection plugin


Outlook

# Integration work ongoing

- Patches for revision during Oct/Nov
- Discussions ongoing
- Need decisions from slurm main developers
  - libasil in contrib or put into select/cray?
  - interface details for node ranking
  - ...

# Would be good to have...

- Frontend mode with multiple slurmds
  - for (ALPS) load-balancing
    - actually ALPS appears to be the main bottleneck, not slurm
    - would be good to avoid single point of failure

# Would be good to have...

- Frontend mode with multiple slurmds
  - for (ALPS) load-balancing
    - actually ALPS appears to be the main bottleneck, not slurm
    - would be good to avoid single point of failure
- Scripted scheduler interface (lua?)

# Would be good to have...

- Frontend mode with multiple slurmds
  - for (ALPS) load-balancing
    - actually ALPS appears to be the main bottleneck, not slurm
    - would be good to avoid single point of failure
- Scripted scheduler interface (lua?)
- Would like to port our backfilling algorithm
  - very high utilisation (> 90%) on rosa
  - requires large array to make “schedule plan”

# Would be good to have...

- Frontend mode with multiple slurmds
  - for (ALPS) load-balancing
    - actually ALPS appears to be the main bottleneck, not slurm
    - would be good to avoid single point of failure
- Scripted scheduler interface (lua?)
- Would like to port our backfilling algorithm
  - very high utilisation (> 90%) on rosa
  - requires large array to make “schedule plan”
- We are also interested in GRES for GPU cluster

# Bon appetit