Refactoring ALPS: ALPS Common Libraries and SLURM Plugins

SLURM User Group Meeting
September 2013

Abstract

ALPS Common Libraries and SLURM Plugins

A move towards offering "native" Workload Manager support on Cray systems

One of the hallmarks of the Cray Linux Environment is the Cray Application Level Placement Scheduler (ALPS). ALPS is a resource placement infrastructure used on all Cray systems. Developed by Cray, ALPS addresses the size, complexity, and unique resource management challenges presented by Cray systems. It works in conjunction with workload management tools such as SLURM to schedule, allocate, and launch applications. ALPS separates policy from placement, so it launches applications but does not conflict with batch system policies. The batch system interacts with ALPS via an XML interface. Over time, the requirement to support more and varied platform and processor capabilities, dynamic resource management and new workload manager features has led Cray to investigate alternatives to provide more flexible methods for supporting expanding workload manager capabilities on Cray systems. This presentation will highlight Cray's plans to expose low level hardware interfaces by refactoring ALPS to allow 'native' workload manager implementations that don't rely on the current ALPS interface mechanism.

The ALPS Refactoring Project

- **Part I**
  - Background
  - Motivation, Goals & Objectives
  - Native SLURM
- **Part II**
  - Workload Manager Interface Unification

Cray 'ALPS' Interface

ALPS
Basil

Adaptive COMPUTING

allinea

Altair

grid engine

IBM Platform LSF

slurm
workload manager

TOTALVIEW
TECHNOLOGIES

ALPS: Background

The ALPS Suite

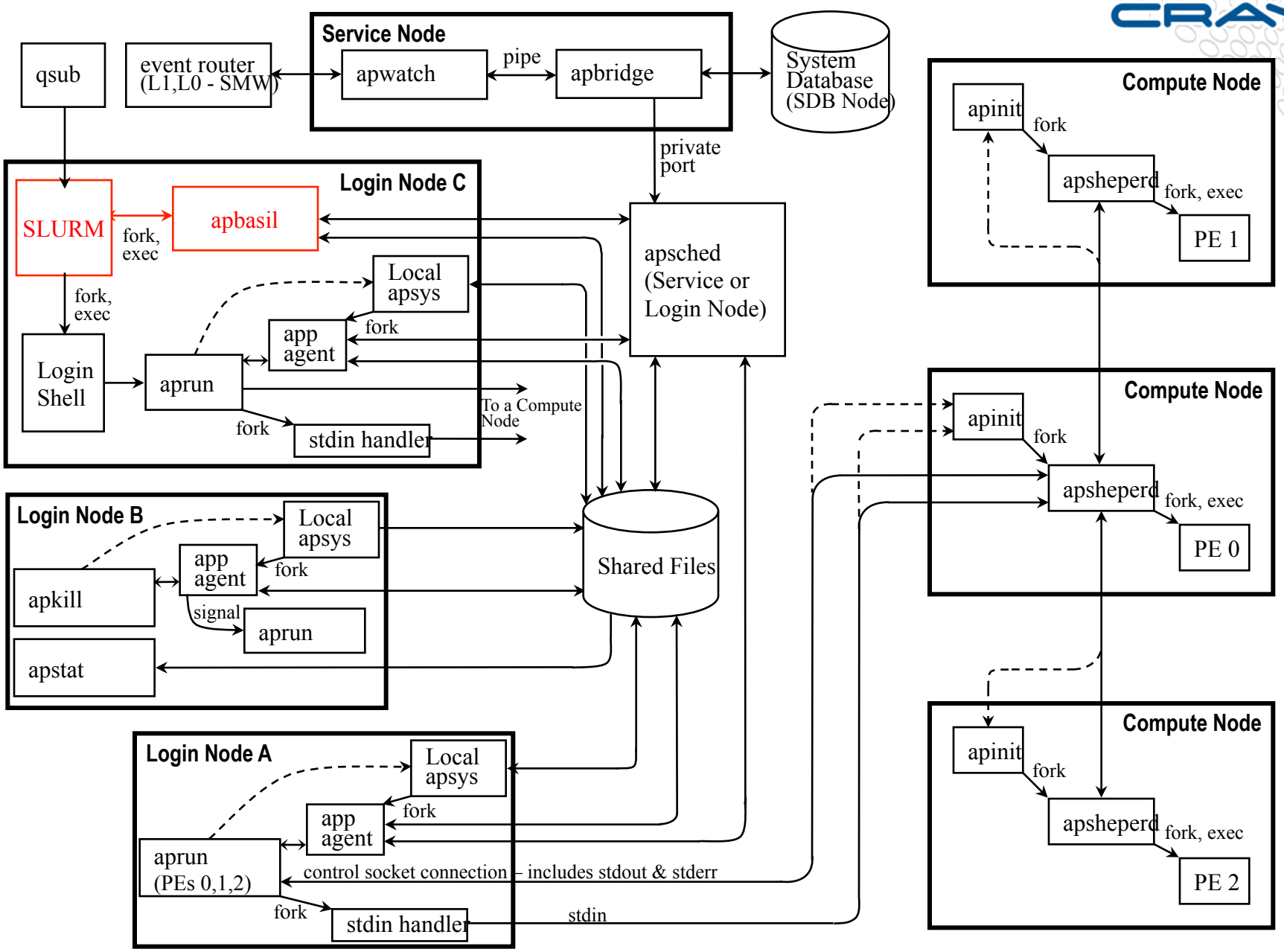
Clients

- aprun – Application submission
- apstat – Application status
- apkill – Signal delivery
- apmgr – Administration interface
- apbasil – Workload manager interface

Servers

- apsys – Client interaction on login nodes
- apinit – Process management on compute nodes
- apsched – Reservations and placement
- apbridge – System data collection
- apwatch – Event monitoring

Tightly integrated…


qsub
event router
(L1,L0 - SMW)
Service Node
apwatch
pipe
apbridge
System
Database
(SDB Node)
private
port
Login Node C
SLURM
fork,
exec
apbasil
fork,
exec
Local
apsys
fork
app
agent
Login
Shell
aprun
fork
stdin handler
To a Compute
Node
apsched
(Service or
Login Node)
Compute Node
apinit
fork
apsheperd
fork, exec
PE 1
Compute Node
apinit
fork
apsheperd
fork, exec
PE 0
Login Node B
Local
apsys
app
agent
fork
apkill
signal
aprun
apstat
Shared Files
Compute Node
apinit
fork
apsheperd
fork, exec
PE 2
Login Node A
Local
apsys
fork
app
agent
aprun
(PEs 0,1,2)
control socket connection – includes stdout & stderr
fork
stdin handler
stdin

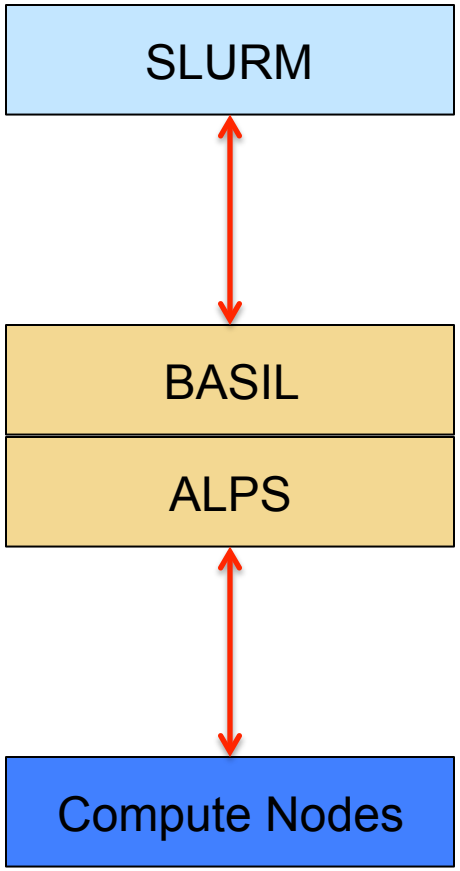
SLURM on Cray systems

- **The version of SLURM that is offered is the current open source version available on the SchedMD/SLURM web page.**
  - Latest version validated on Cray systems is **2.6**
- **Basic WLM functions**
  - **This version supports most Cray features, but is a subset**
    - Uses the Cray ALPS Workload Manager interface
    - Cray/ALPS doesn't support all of the SLURM capabilities
    - SLURM doesn't support all of the Cray/ALPS capabilities
- **Cray has contract(s) to add enhancements to SLURM for Cray**
  - These and existing Cray enhancements will be pushed upstream to SchedMD to be included in open source SLURM repository

“Hybrid” SLURM Architecture for Cray

SLURM

- Prioritizes queue(s) of work and enforces limits
- Decides when and where to start jobs
- Terminates job when appropriate
- Accounts for jobs and job steps
- No daemons on compute nodes



ALPS

- Allocates and releases resources for jobs
- Launches tasks
- Monitors node health
- Manages node state
- Has daemons on compute nodes
- Manages Cray network resources

SLURM is a scheduler layer above ALPS and BASIL, not currently a replacement

ALPS Refactoring: Motivation

Evolving system requirements driving changes

- Workload manager role not “just launching jobs”
- The role of a resource manager is managing job’s resource requirements *throughout* job
- The resource manager’s work only starts when a job begins processing
- The information a resource manager needs is constantly changing
- Resources a job needs are constantly changing
- Resiliency is an application’s responsibility with system’s assistance

From SLURM User Group Meeting Keynote Address, 2011

ALPS Refactoring

Cray Goals and Objectives - Long term

- Meet market requirements to offer variety of WLMs
  - SLURM
  - Adaptive Computing MOAB
  - Altair PBS Pro
  - IBM/Platform LSF
  - GridEngine
- Design/implement in a way beneficial to other Cray projects and business groups
- Allow 3rd Party Workload Managers to work “out of the box” and unmodified

ALPS Refactoring

Cray Goals and Objectives – Long term

- Leverage full capabilities of 3rd party WLM
- Provide access to Cray platform specific hardware (similar to Linux clusters) and let WLM manage resources
- Reduce support costs
  - 3rd party porting/integrating WLM to Cray
  - Cray QA of 3rd party WLMs

The ALPS Refactoring Project

- Phase I
  - Client/Server ALPS --> common library functions split apart from ALPS
- Phase II
  - “Native” SLURM
- Phase III
  - Workload Manager Interface Unification

ALPS Refactoring: "Native" SLURM

ALPS Transition

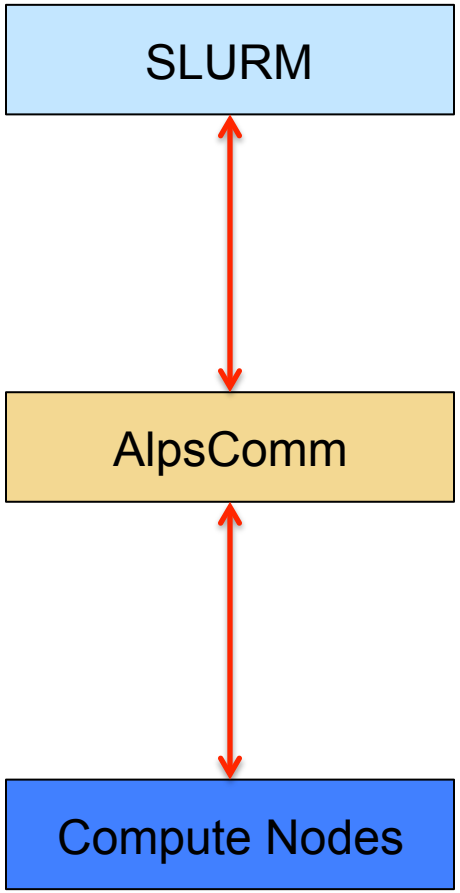
ALPS Refactoring

- **Develop a "native" SLURM implementation**
  - Cray developed plugins to provide following services:
    - Dynamic node state change information
    - System topology information
    - Congestion management information for HSS
    - Protection key and protection domain management
    - Node Health Check support
    - Network performance counter management
    - PMI port assignment management (when more than one application per compute node)
  - Working with SchedMD on implementation

“Native” SLURM Architecture for Cray

SLURM

- Prioritizes queue(s) of work and enforces limits
- Decides when and where to start jobs
- Terminates job when appropriate
- Accounts for jobs and job steps
- Allocates and releases resources for jobs



SLURM

- Launches tasks
- Monitors node health
- Manages node state
- Has daemons on compute nodes
- Plugin changes to:
  - Select
  - Switch
  - Task
  - Job Container (new)

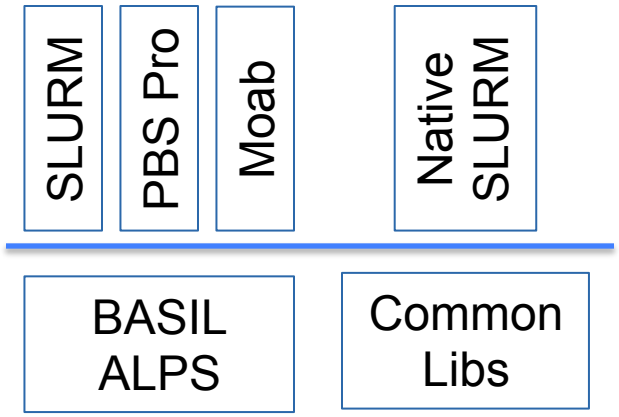
AlpsComm

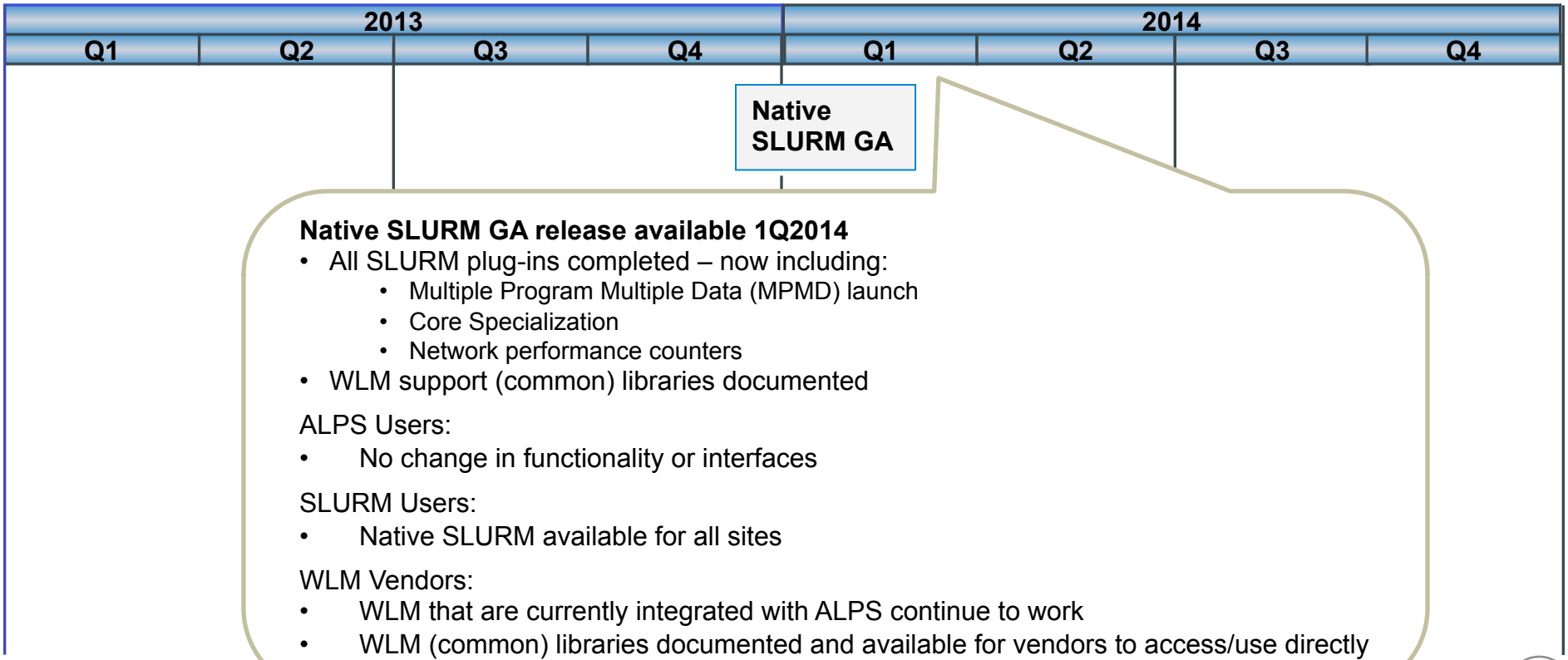
- Low level interfaces for network management

ALPS Refactoring: Native SLURM

- Plugins will be open source
- 'Native' SLURM Limited Access release, December 2013
- 'Native' SLURM General Availability release, 1Q2014

WLM Roadmap: Phase 2a





Native SLURM GA release available 1Q2014

- All SLURM plug-ins completed – now including:
  - Multiple Program Multiple Data (MPMD) launch
  - Core Specialization
  - Network performance counters
- WLM support (common) libraries documented

ALPS Users:

- No change in functionality or interfaces

SLURM Users:

- Native SLURM available for all sites

WLM Vendors:

- WLM that are currently integrated with ALPS continue to work
- WLM (common) libraries documented and available for vendors to access/use directly

Workload Manager Interface Unification

Workload Manager Interface Unification

Cray Goals and Objectives

- Meet market requirements to offer variety of WLMs
- **Allow 3rd Party Workload Managers to work "out of the box" and unmodified**
- Leverage full capabilities of 3rd party WLMs
- Create opportunities for WLM vendors to provide customer specific, value-add features

Workload Manager Interface Unification

Objective: Support WLMs “out-of-the-box”

- Surveyed major MPI implementations
- SLURM MPI launch support common to all

MPI Launcher Comparison

Launcher	OpenMPI (orte)	MVAPICH (hydra)	MPICH2 (hydra)	Intel MPI (hydra)	Platform MPI	Cray MPI
ssh	y	y	y	y	y	n
rsh	y	y	y	y	y	n
SLURM	y	y	y	y	y	y
tm_spawn (PBS, M/T)	y	y	y	y	n	n
qrsh (SGE)	n	y	y	y	n	n
blaunch (LSF)	y	y	y	y	n	n
poe (IBM LL)	n	y	y	y	n	n
ALPS1	y	n	n	n	n	y
ALPS3	y	n	n	n	n	y

Workload Manager Interface Unification

Using SLURM as application launcher

- **Selected workload manager provides resource scheduling management and monitoring**
  - WLM daemon runs on each compute node
  - Obtains node configuration status from WLM daemon
- **SLURM provides:**
  - Application launch services (SLURM job name derived from WLM job ID)
  - Node selection using WLM assignments
  - Enforcement of WLM prescribed resource limits using kernel cpusets and cgroups

Why use SLURM as launcher?

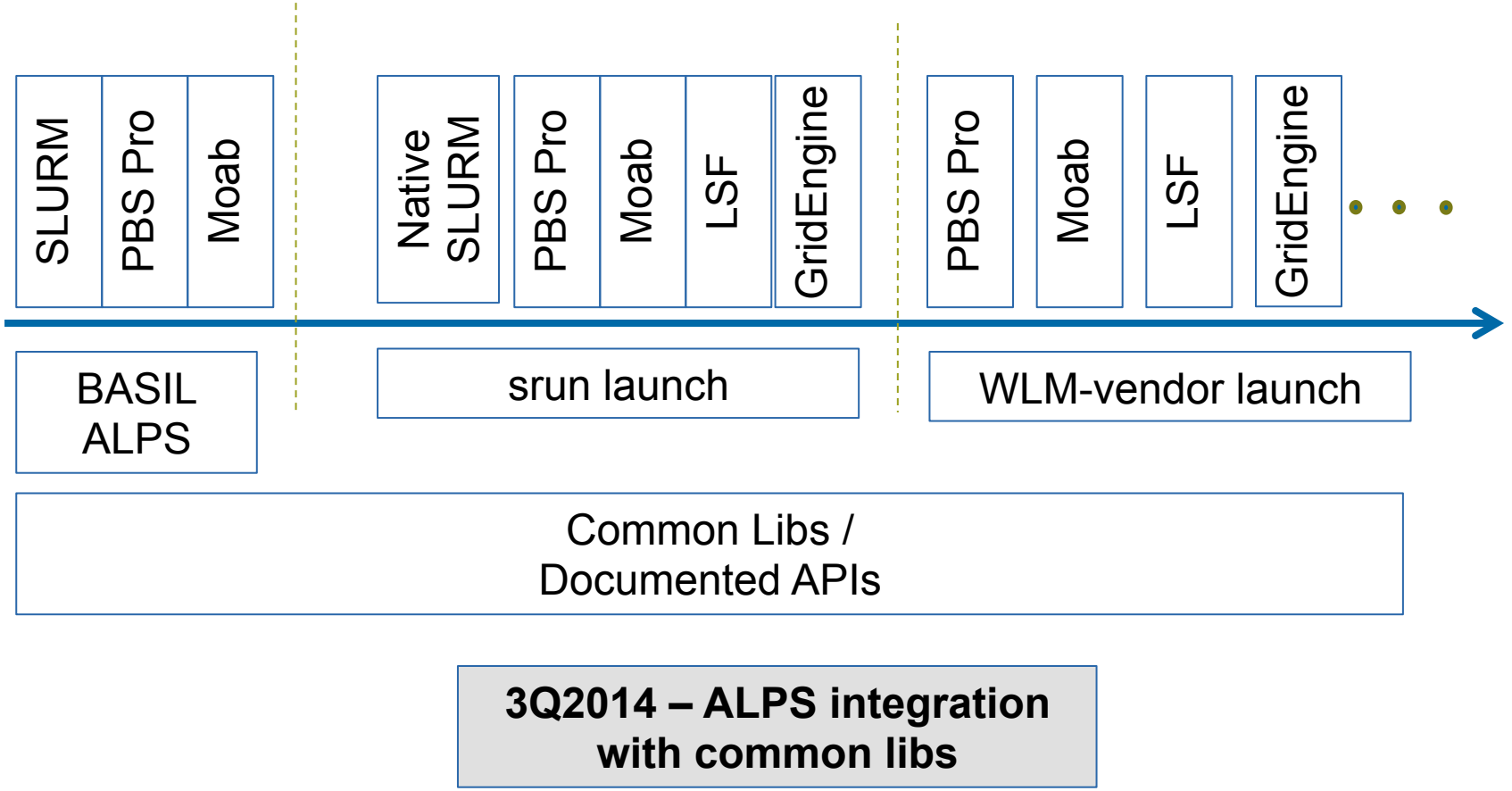
- **Don't have to implement interfaces to low-level AlpsCommon interfaces**
  - Works "out of the box"
  - Allows "native" WLM functionality (most if not all resource management, scheduling and reporting functionality)
- **Majority of 'mpirun' implementations already interface to SLURM as launcher ('srun')**

Workload Manager Interface Unification

Native WLM implementation using low-level AlpsCommon library interfaces

- Direct control over resource allocation and management
- WLM provided application launch mechanism

3Q2014: Supported operating modes



3Q2014 – ALPS integration with common libs

Questions?

Thank You!