ADAC8

Cray Shasta Orchestration for HPC and Cloud

CRAY
a Hewlett Packard Enterprise company

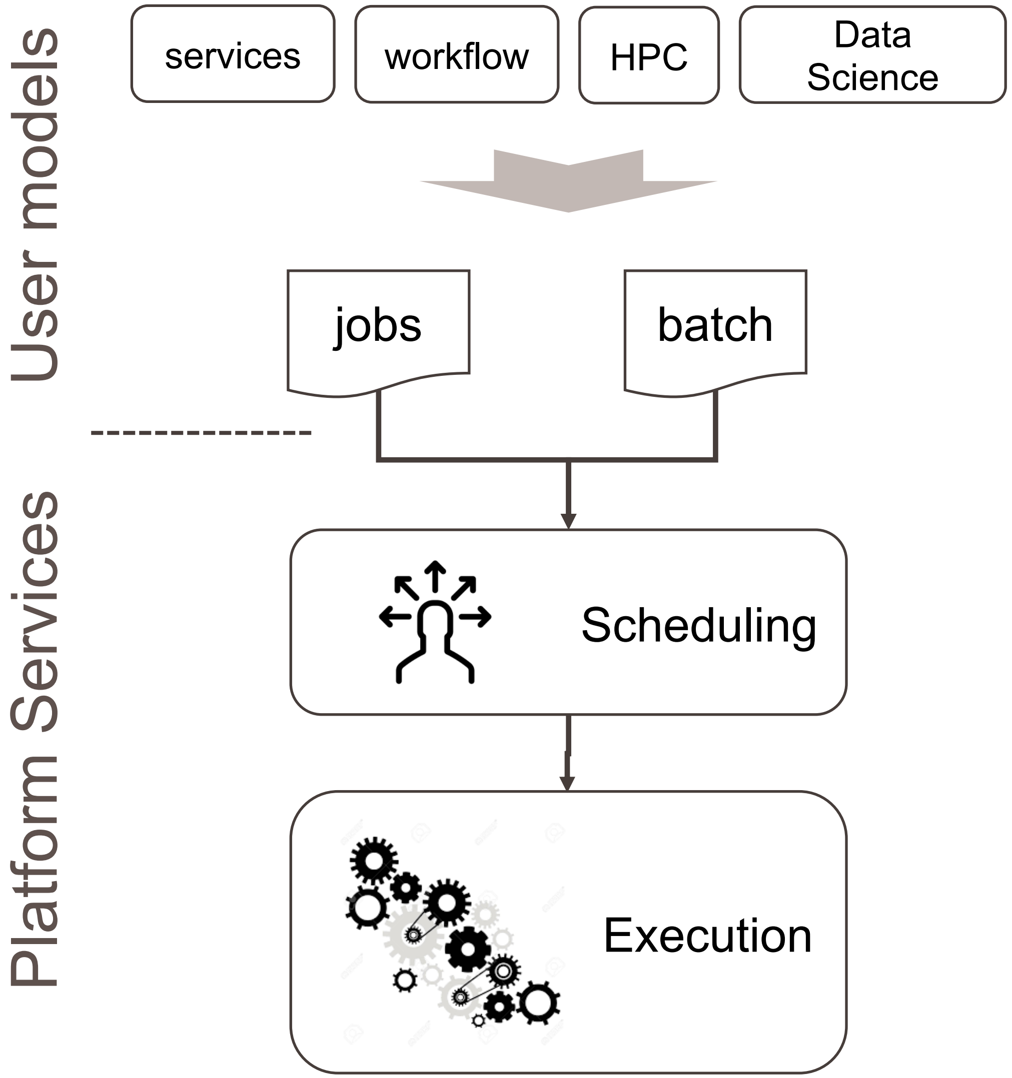
jsparks@cray.com

Agenda

- Overview
- Scheduling
  - Traditional HPC
  - Cloud Orchestration
- Emerging Technologies
  - Workflow agents
  - Capsules
- Q&A

Overview

- Objective: execute traditional batch jobs and cloud-native services & workflows
- Fully utilize system resources
- Three components
  - User models
  - Scheduling services
  - Execution services



HPC Orchestration User Cases

- Kubernetes Use Cases (for HPC)
  - Projects need to express their workflows and supporting services in a portable cloud-native way
  - Third-party tools will increasingly assume compatibility with the kubernetes platform
  - Emerging workflow tools assume kubernetes or strongly support it
    - Argo: container native workflow engine for parallel jobs
    - Kube-batch: batch ‘like’ scheduler for kubernetes
    - Others: kubeflow, nextflow, airflow, volcano

SCHEDULING

Scheduler Technologies

CRAY
a Hewlett Packard Enterprise company

Meta-schedulers
- Mesos, …(GridWay)
- Attributes:
  - Broker resources
  - Global resource view

HPC Schedulers
- WLM: Slurm, PBS, Flux, etc.
- Attributes:
  - Placement
  - Scheduling polices
  - Resource controls

Container Orchestrators
- Kubernetes, Swarm, Nomad, etc.
- Attributes:
  - Lifecycle Management
  - Monitoring
  - Placement

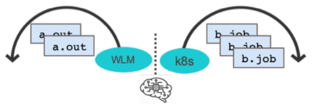
Orchestration and Workload Management

partitioned ← … → converged

Separate environments	Containers via a WLM	Orchestration job scheduling features	Run both natively in shared environment
• Deploy containerized applications separate from HPC applications • Benefit: Avoid disrupting existing environments • Challenge: Siloed partitions	• Use workload manager to instantiate containers • Benefit: Containerized workloads with minimal disruption to their environment. • Challenge: no access to orchestration features	• Use existing scheduling facilities in orchestrator • Benefit: Leverage innovation in orchestration tools • Challenge: Interaction with traditional HPC and relevant schedulers increases complexity and potentially decreases performance	• Orchestration and workload manager co-exist on the same cluster • Benefit: With this approach, WLM acts as a resource manager, making resources available to the Orchestrator. • Challenge: Difficulty of sharing resources

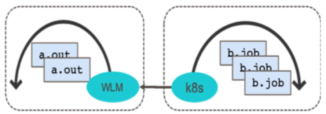
Multi-framework Scheduling Options

- A – **Independent schedulers**



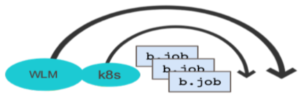
  - Admin manages scheduling domains – fixed allocations
  - Rebalancing requires reboot
  - Inefficient use of system resources

- C – **Master worker scheduler**



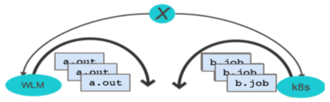
  - Corporative scheduling environment
  - Authoritative scheduling – single system view
  - Requires bridging master and subordinate schedulers

- B – **Subordinate scheduler**



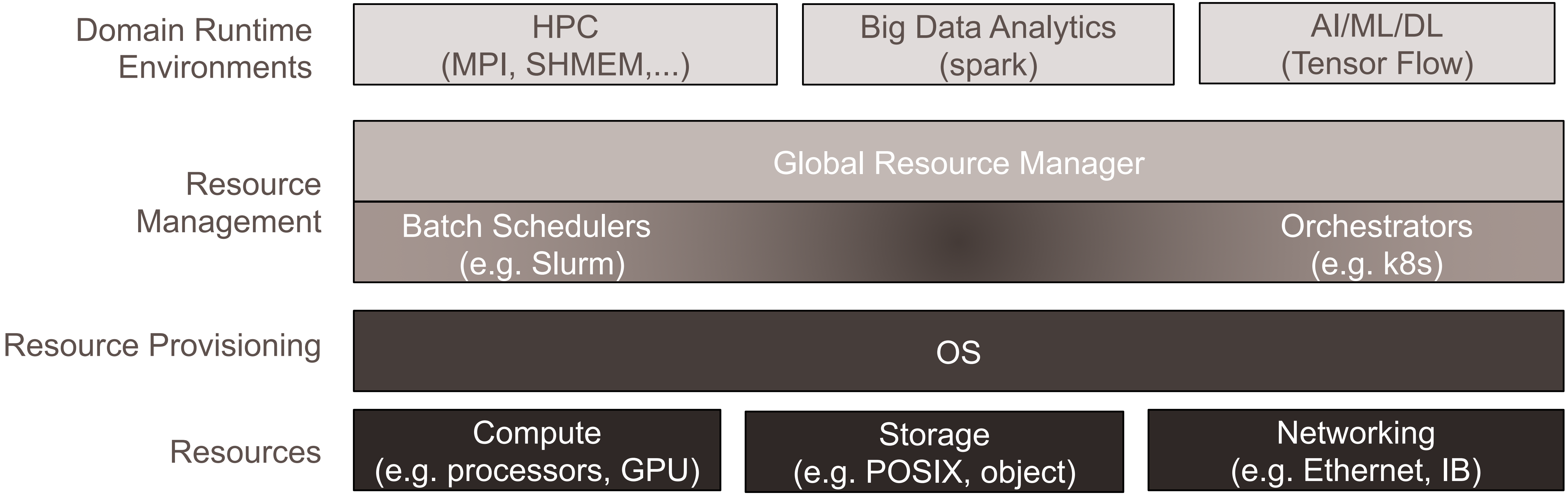
  - User defined framework
  - Outer scheduler owns resources, Inner uses resources
  - Independent/adversarial schedule policies
    - Inefficient use of system resources

- D – **Meta-scheduler environment**



  - Supports multiple workload paradigms
  - Authoritative scheduling – single system view
  - Requires scheduler modifications

Converged System

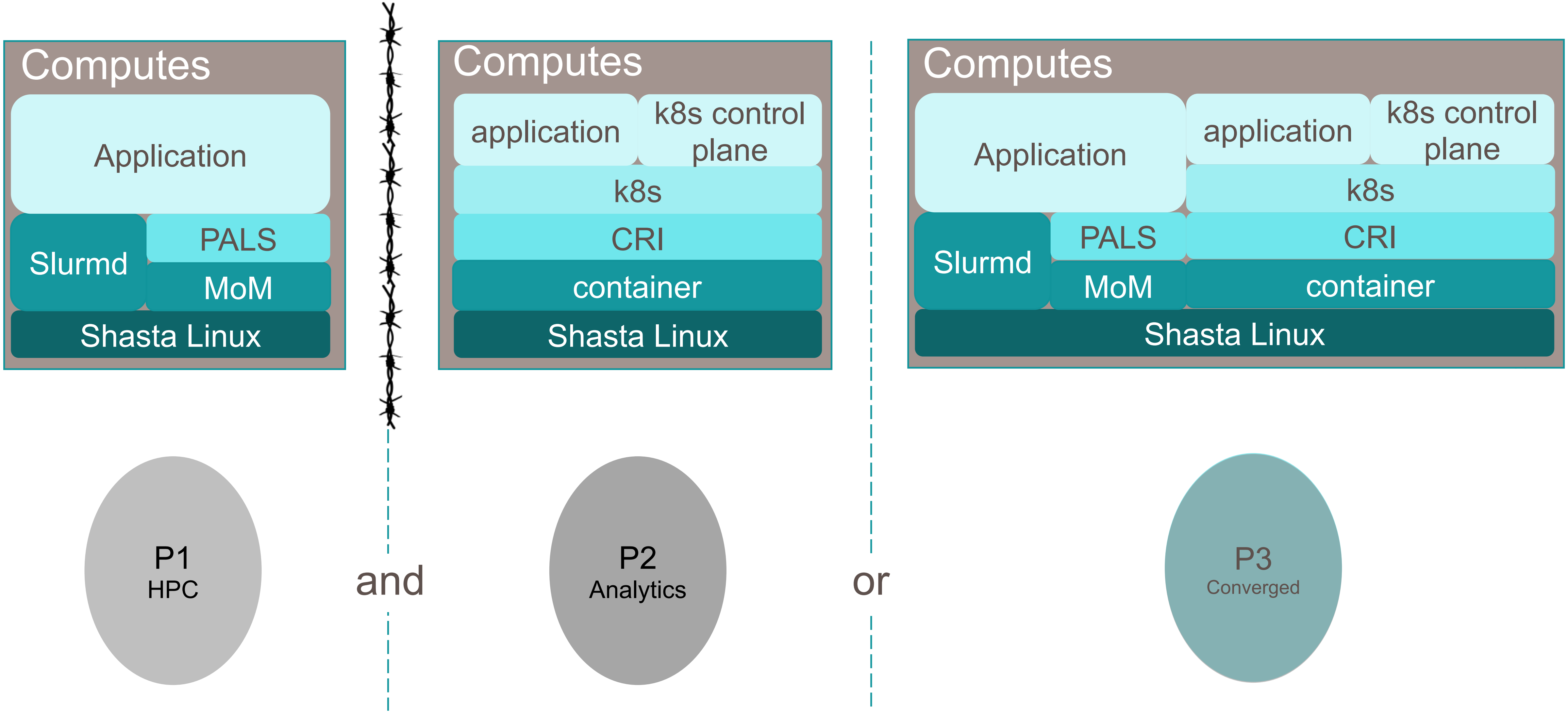


Assumptions in Orchestration Services

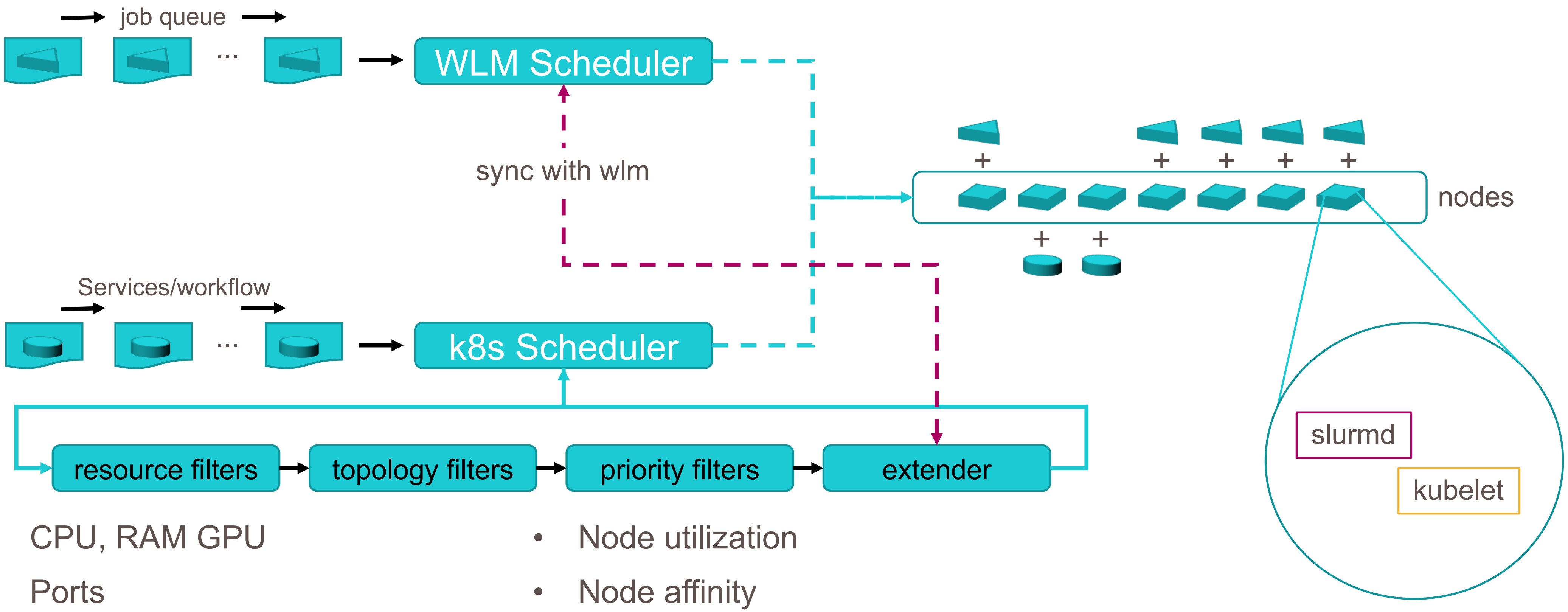
- Typically used in cloud environments – not HPC
- The user “owns” the resources
- Focused on deploying a set of services and keeping them in a good state
  - Tends to assume there is always enough resource (or it can be quickly provisioned – elastic consumption)
  - Persistent, long-running (no walltime)
  - Not HPC application focused[†]
- Definition of a consumable can be extensible

† kube-batch offers some HPC semantics, job priorities, polices

Shasta Provisioned Stacks



K8S – Extension Scheduler



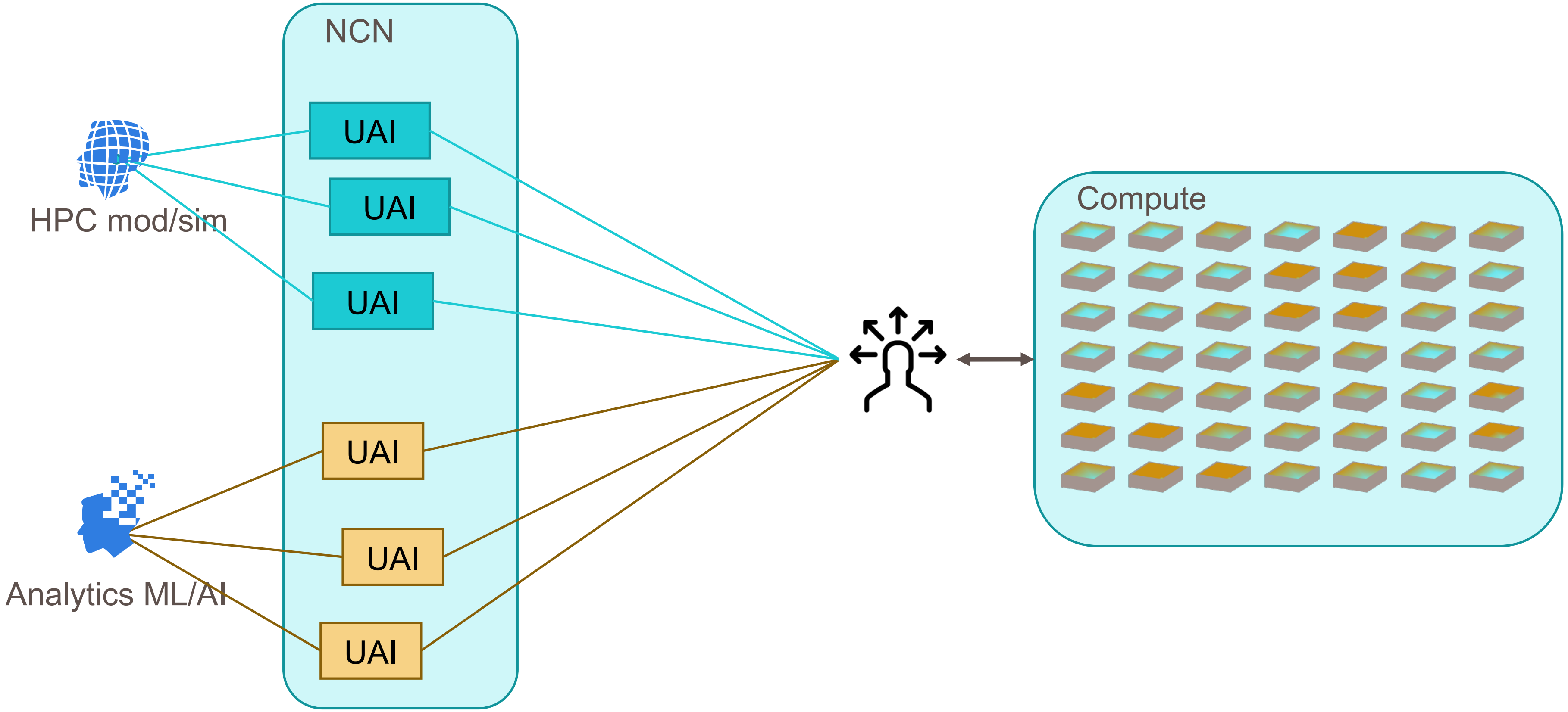
- CPU, RAM GPU
- Ports
- RAM, disk pressure

- Node utilization
- Node affinity
- Taint toleration

- Constraints
- Affinity
- Taints

- WLM decision/reservation

Shasta User View – Hybrid-Scheduling



Scheduling Opportunities and Challenges

- Job Type Characteristics
  - Parallel jobs – gang scheduling (kube-batch)
  - High throughput (million jobs) – scheduler latency
  - Interactive and batch
- Resource Dependencies
  - Accelerators, networks, storage – resource plugins
- Different Scheduling Options
  - On demand vs. queue
  - Workflow agents – argo, kube-flow, nextflow, etc.

CAPSULES

Introduction

What are Capsules

- A Capsule is a declarative specification of a desired runtime environment
- Aims to provide a common abstraction for running a wide variety of workloads
  - Reduce cognitive overheads of running across differing schedulers and orchestrators
    - i.e. don't force end users to be K8S/Slurm/PBS experts
  - Focus has been on enabling interactive supercomputing

Capsules – Design Goals

- Declarative
  - Define what the user wants
  - Capsules handles details of launching that on the underlying platform
- User Friendly
  - Inspired by common tools, such as conda, docker, git
- Portable
  - Define a capsule once, run on any supported hardware/software (within reason)
- Extensible
  - Extensible architecture allows vendor (and customers) to quickly support new functionality

Capsules – Example ML Workflow

```
> capsule create my-workflow
> capsule open my-workflow
> capsule add payload tensorflow
> capsule edit payload add --config name simple-tf
> capsule edit payload set --image default tensorflow/tensorflow
> capsule edit payload remove --args "--user_str1=hello"
> capsule edit payload add --config trainingScript simple.py
> capsule edit payload set --data home `pwd` /home/trainer Directory false all
> capsule close
> capsule launch my-workflow
> … wait for job to complete …
> capsule kill my-workflow
```

- Open a capsule for editing, make changes and close it to save those changes
- Then launch the capsule, wait for results and clean up when done

Capsules – Example HPC Workflow

a Hewlett Packard Enterprise company

```
> capsule create mpi-example
> capsule open mpi-example
> capsule add payload hpc-job
> capsule edit payload add --config command /lus/<username>/a.out
> capsule edit payload add --resource instance nodes 4 \
    --resource instance per-node 4 --resource memory per-node 4G
> capsule close
> capsule launch mpi-example --attach
> … wait for job to complete …
> capsule kill my-workflow
```

- Very similar workflow to previous example, just different config options
- This time using **--attach** to directly attach ourselves to the capsule
  - In this example this will result in monitoring the job output file

FORWARD LOOKING STATEMENTS

This presentation may contain forward-looking statements that involve risks, uncertainties and assumptions. If the risks or uncertainties ever materialize or the assumptions prove incorrect, the results of Hewlett Packard Enterprise Company and its consolidated subsidiaries ("Hewlett Packard Enterprise") may differ materially from those expressed or implied by such forward-looking statements and assumptions. All statements other than statements of historical fact are statements that could be deemed forward-looking statements, including but not limited to any statements regarding the expected benefits and costs of the transaction contemplated by this presentation; the expected timing of the completion of the transaction; the ability of HPE, its subsidiaries and Cray to complete the transaction considering the various conditions to the transaction, some of which are outside the parties' control, including those conditions related to regulatory approvals; projections of revenue, margins, expenses, net earnings, net earnings per share, cash flows, or other financial items; any statements concerning the expected development, performance, market share or competitive performance relating to products or services; any statements regarding current or future macroeconomic trends or events and the impact of those trends and events on Hewlett Packard Enterprise and its financial performance; any statements of expectation or belief; and any statements of assumptions underlying any of the foregoing. Risks, uncertainties and assumptions include the possibility that expected benefits of the transaction described in this presentation may not materialize as expected; that the transaction may not be timely completed, if at all; that, prior to the completion of the transaction, Cray's business may not perform as expected due to transaction-related uncertainty or other factors; that the parties are unable to successfully implement integration strategies; the need to address the many challenges facing Hewlett Packard Enterprise's businesses; the competitive pressures faced by Hewlett Packard Enterprise's businesses; risks associated with executing Hewlett Packard Enterprise's strategy; the impact of macroeconomic and geopolitical trends and events; the development and transition of new products and services and the enhancement of existing products and services to meet customer needs and respond to emerging technological trends; and other risks that are described in our Fiscal Year 2018 Annual Report on Form 10-K, and that are otherwise described or updated from time to time in Hewlett Packard Enterprise's other filings with the Securities and Exchange Commission, including but not limited to our subsequent Quarterly Reports on Form 10-Q. Hewlett Packard Enterprise assumes no obligation and does not intend to update these forward-looking statements.

QUESTIONS?

CRAY®
a Hewlett Packard Enterprise company

jsparks@cray.com

cray.com

@cray_inc

linkedin.com/company/cray-inc-/