

Bebida Improvements and Optimizations

REGALE Project WP2

Michael Mercier - Ryax Technologies



1



2

2023

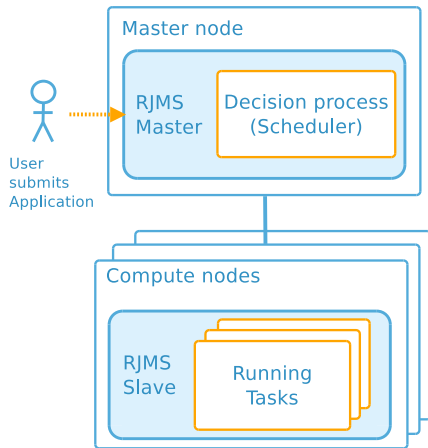
Resources and Jobs Management System (RJMS)

a.k.a. Batch scheduler

- ▶ Shares resources between users
- ▶ Resources are CPU, Memory, Network bandwidth, ...
- ▶ Conflicting objectives:
 - ▶ Minimize waiting time
 - ▶ Maximize jobs throughput
 - ▶ Maximize cluster utilization
- ▶ Examples:
 - ▶ HPC: SLURM, OAR, PBS, ...
 - ▶ Big Data: Hadoop YARN, Kubernetes ...

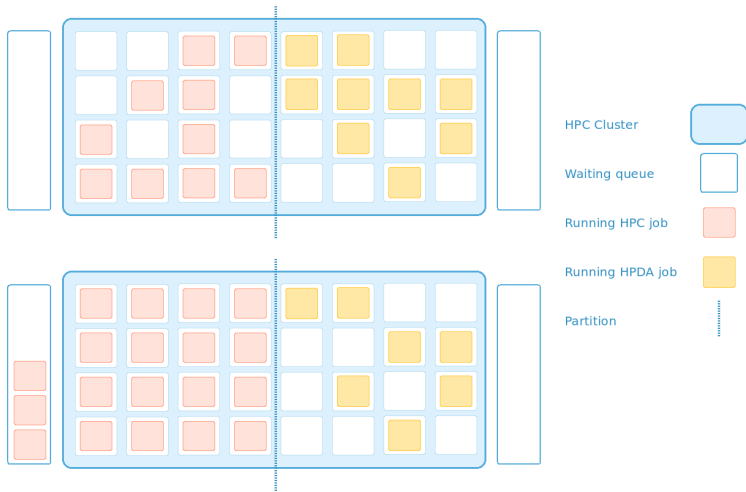


RJMS are complex:
SLURM and YARN \simeq 400k lines of code



What's happening today

Static partitioning



Static cluster partitioning:

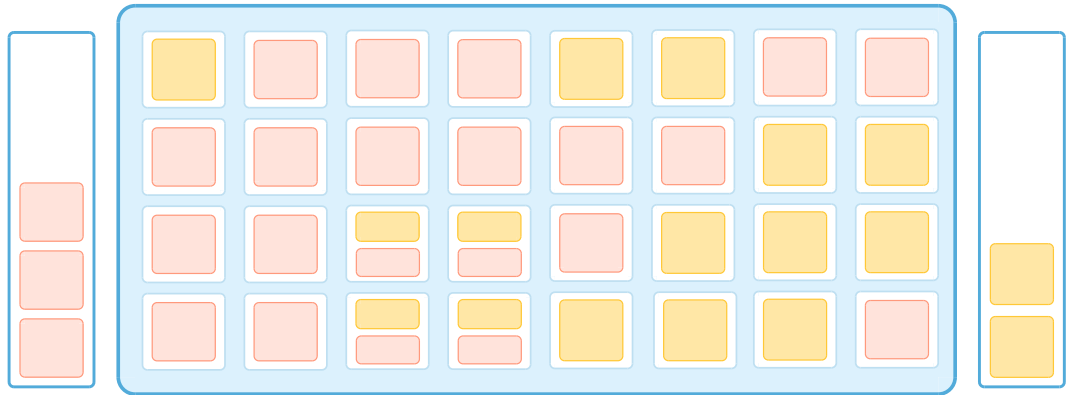
- ▶ Simple workload discrimination
- ▶ No node sharing

Leads to resources waste!

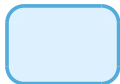
- ▶ When one queue is filling up
- ▶ Available nodes on the other partition are not used

What we want to achieve

Dynamic sharing



HPC Cluster



Running HPC job



Waiting queue



Running HPDA job



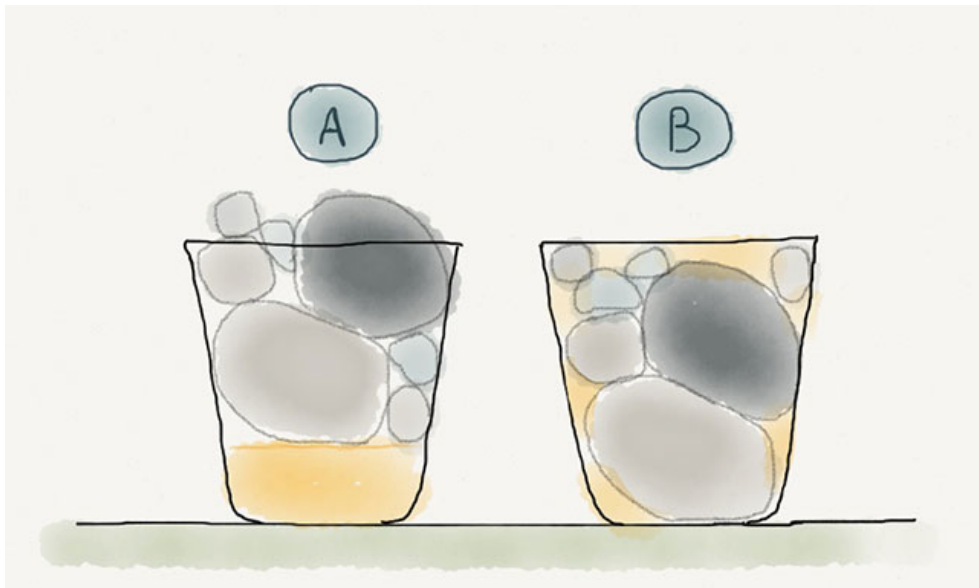
An analogy

Is the Jar Full?



A Scheduling Problem

Solutions



Big Data and HPC workload Collocation

At the Resource Management Level

Jar: HPC Hardware Cluster

- ▶ Resources (CPU cores)
- ▶ Time (seconds)

Stones: HPC Jobs

- ▶ Static resource allocation
- ▶ Static time allocation

Big Data and HPC workload Collocation

At the Resource Management Level

Jar: HPC Hardware Cluster

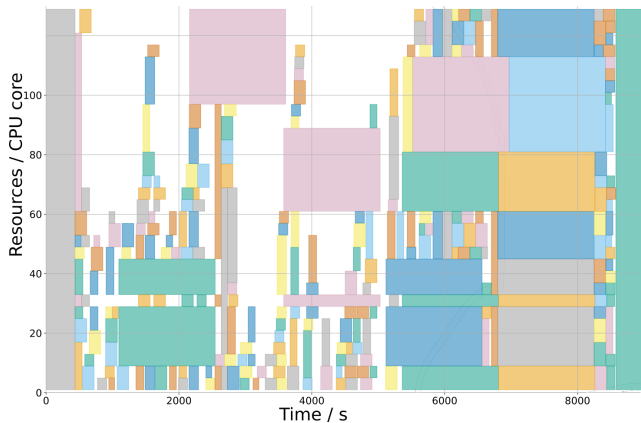
- ▶ Resources (CPU cores)
- ▶ Time (seconds)

Stones: HPC Jobs

- ▶ Static resource allocation
- ▶ Static time allocation

Sand: Big Data Applications

- ▶ Dynamic resource allocation
- ▶ Resilient to resource preemption



Existing Approaches for Collocation

Related Works

BigData in HPC job

- ▶ Setup/teardown at every job
- ▶ Need to import/export data

Two-level Scheduling

- ▶ Dynamically resources sharing negotiation
- ▶ Increase operational cost
- ▶ New API implementation required

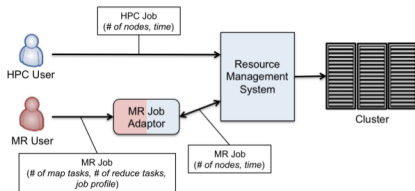
Examples: Mesos, Omega, Grid Engine

The major problem is complexity:
=> from 5k to 50k lines of code

RJMS adaptor

- ▶ HPC RJMS provides resources to Big Data
- ▶ Convert BigData resource requests
- ▶ Walltime problem

Examples: Neves et al. (Euro-Par 2012),
Intel YARN adaptor



Keep it Simple

Requirements

We want our solution to:

1. **Use existing tools unmodified** (RJMS + Applications)
2. **Work for any RJMS**
3. **Transparent for users**
4. **Minimal disturbance to HPC jobs** from Big Data applications
5. **Leverage Big Data dynamicity and resilience**

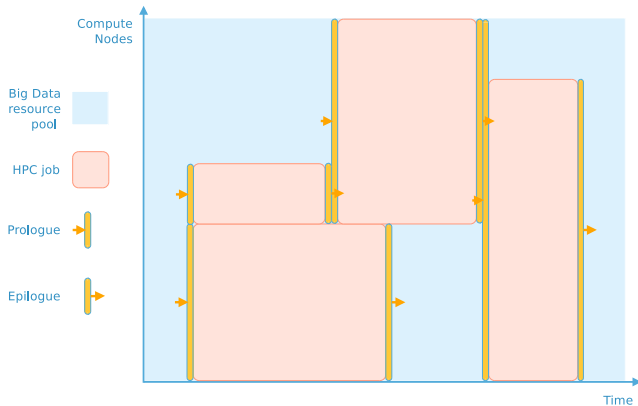
BeBiDa: Best effort Big Data on HPC cluster

Using HPC idle resources for Big Data analytics

HPC: job's prolog/epilog (50 LoC)

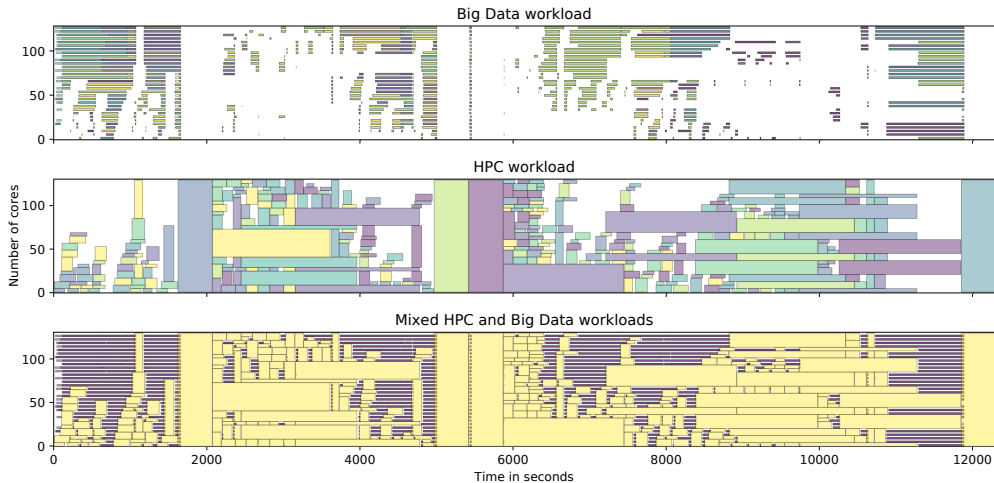
**BigData: resources
{de}commission**

1. *Initially:*
Nodes are attached to the Big Data resource pool
2. *When an HPC job starts:*
Prologue decommissions the resources
3. *When an HPC job finishes:*
Epilogue is giving resources back



BeBida Working Example

From 70% to 100% utilization



Resource Preemption Impact on Big Data

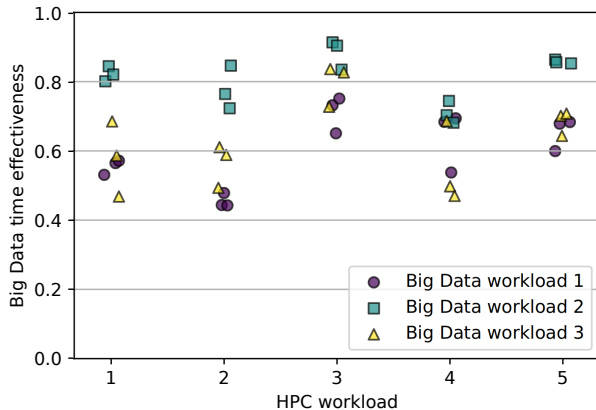
Measure effective computation time

Time Effectiveness

$$E = \frac{\text{Useful computation time}}{\text{Total computation time}}$$

Variability root causes:

- ▶ HPC workloads shape
- ▶ Big Data workload applications sensitivity
- ▶ Coincidence between "holes" and large applications
- ▶ System noise



Overhead on HPC Workloads

With and without BeBiDa

Mean Execution Time

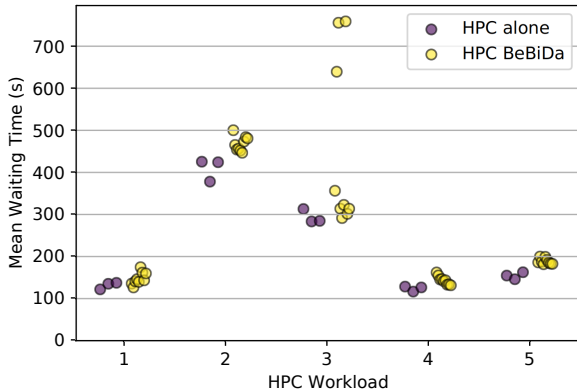
+6% on average with BeBiDa

Mean Waiting Time

- ▶ epilog/prolog: < 16s
- ▶ 26% (1min) to 1% (30min) of the jobs execution time
- ▶ +17% on average with BeBiDa

Variability root causes:

- ▶ Scheduling anomalies
- ▶ Network contentions
- ▶ DFS daemon



Key Characteristics

Advantages and Drawbacks

Performance impact

- ▶ HPC execution time +6% on average
- ▶ HPC waiting time +17% on average
- ▶ High variability in Big Data efficiency (68% on average)

BeBiDa main advantages

- ▶ Use any existing RJMSs without modification
 - ▶ Transparent for users
 - ▶ HPC workload has priority
 - ▶ Only \simeq 50 lines of code + configuration
- 100 to 1000 times less code**
- ▶ Increase the cluster utilization
 - ▶ Uses resources that would be idle otherwise

Fully reproducible experiments and analysis:

<https://gitlab.inria.fr/mmercier/bebida/>

REGALE Use cases

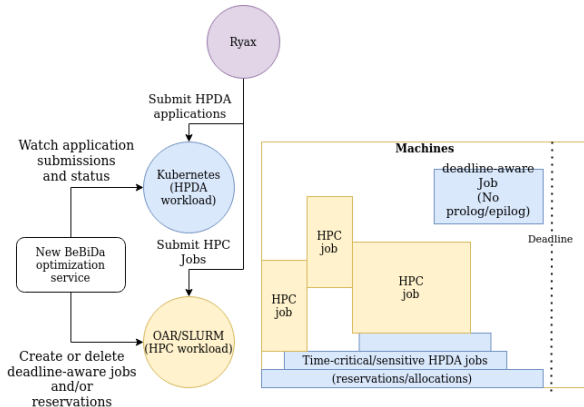
and Optimizations

Spark Application for:

1. Hydrological flow analysis (Pilot 4)
2. Network security analysis (Pilot 3)

We want more guarantees:

- ▶ **Deadline aware (1):**
Run before a user-defined deadline
- ▶ **Time critical (2):**
Run as fast as possible



Bebida Optimization Service

Roadmap

- ☒ First implementation with simple heuristic (Ti'Punch):
based on threshold on the number of pending job in the BDA queue, create HPC jobs that will stay in the BDA resource pool.
- ☒ support for K8s (BDA)
- ☒ support Slurm over SSH (HPC)
- ☒ Full testing environment
- ☒ Handle BDA app early termination (cancel HPC job if not used anymore)
- ☐ **Support for OAR (HPC)**
- ☐ Handle HPC job termination
- ☐ Improve heuristic using BDA app time and resource requirements

Hackathon Plan

Respond to

- ▶ Which version of OAR to use?
=> 2 or 3?
- ▶ How to create a container-based testbed for Bebida with OAR and K8s?
=> NXC or docker compose?
- ▶ What is the best way to implement the deadline?
=> Advanced reservation? Deadline aware jobs?
- ▶ What is the best way to implement the time-critical?
=> Priorities? Co-system? Walltime-less?
- ▶ How to connect to OAR?
=> REST API or SSH + commands?

Let's Hack ! B-)