Optimal, Multi-Dimensional Resource Provisioning for Scientific Workloads

(MidTerm / Work-In-Progress)

Remo Andreoli, PhD Student

Tommaso Cucinotta, Supervisor

Marco Mambelli, FNAL Supervisor

Sant'Anna School of Advanced Studies

The provisioning problem at Fermilab

- **Scientists** submit **jobs** to the batch system (GlideinWMS)
- GlideinWMS has access to multiple **sites** (grids, cloud providers, local datacenter, …)
- Each site provides a catalog of **machine types** (Vms, containers, physical hosts, …)
- GlideinWMS matches:
    - Job Requirements
    - Resource Capabilities of a machine type
- GlideinWMS reserves a machine based on the matching result
- GlideinWMS schedules the jobs on the provisioned machine

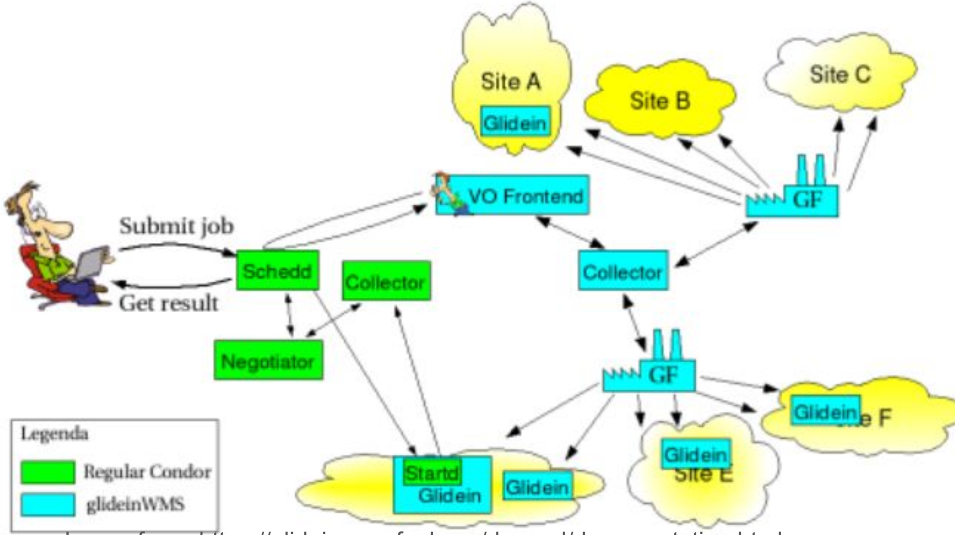


Image from: https://glideinwms.fnal.gov/doc.prd/documentation.html

Optimal resource provisioning challenge

- As of now, GlideinWMS provisions the jobs according to their HW requirements
- However, it is unable to make sophisticated provisioning decisions:
  - How to schedule the jobs to get the results in the **shortest amount of time**?
  - How to provision the jobs with the **minimal amount of rental cost**?
- Conflicting objectives: generally you **pay more** to **wait less**
- **W.I.P.** : Let's consider rental cost only

Classical approach: Mixed-integer Linear Programming

- Translate the real-world problem in to a mathematical model of form:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{Z}^n}{\text{maximize}} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

- **x** is a vector of incognitas (called the decision variables)
- **c** is a vector of coefficients for the objective
- A and **b** are a matrix and vector, respectively, of coefficients that define the search space
- Exploit MILP solver for a guaranteed optimal solution to your problem

Scalability issues with MILP

- Our provisioning problem is related to the classical bin-packing problem:

Pack items of different sizes into a finite number of bins, each of a fixed given capacity, in a way that minimizes the number of bins used

- But with an additional dimension (We don’t know the capacity of the “bin” a priori)
- The bin-packing problem is **NP-hard:**
  - No known polynomial-time algorithm to solve it
  - Requires an exhaustive search and evaluation of numerous potential solutions

System model (1)

- Set of jobs to be provisioned
- Properties of a job:

$$\forall j \in J : \begin{cases} R_j^{CPU} \in \mathbb{R}^+ & \text{Number of CPU cores required} \\ R_j^{MEM} \in \mathbb{R}^+ & \text{Megabytes of memory required} \\ R_j^{GPU} \in \{0,1\} & \text{GPU needed} \\ D_j \sim Pareto(x_m; \alpha) & \text{Job duration, where } x_m = 1h \end{cases}$$

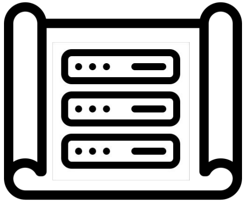
- In our case, a job is (part of) a simulation

System model (2)

- Set of machine types (blueprints) to be instantiated
- Properties of a blueprint:

$$\forall b \in B : \begin{cases} C_b^{CPU} \in \mathbb{R}^+ & \text{Number of CPU cores available} \\ C_b^{MEM} \in \mathbb{R}^+ & \text{Megabytes of memory available} \\ C_b^{GPU} \in \{0, 1\} & \text{GPU available} \\ C_b \in \mathbb{R}^+ & \text{Rental cost (usd/min)} \\ P_b \in \mathbb{R}^+ & \text{Maximum rental period (min)} \\ Q_b \in \mathbb{N} & \text{Waiting time (min)} \end{cases}$$

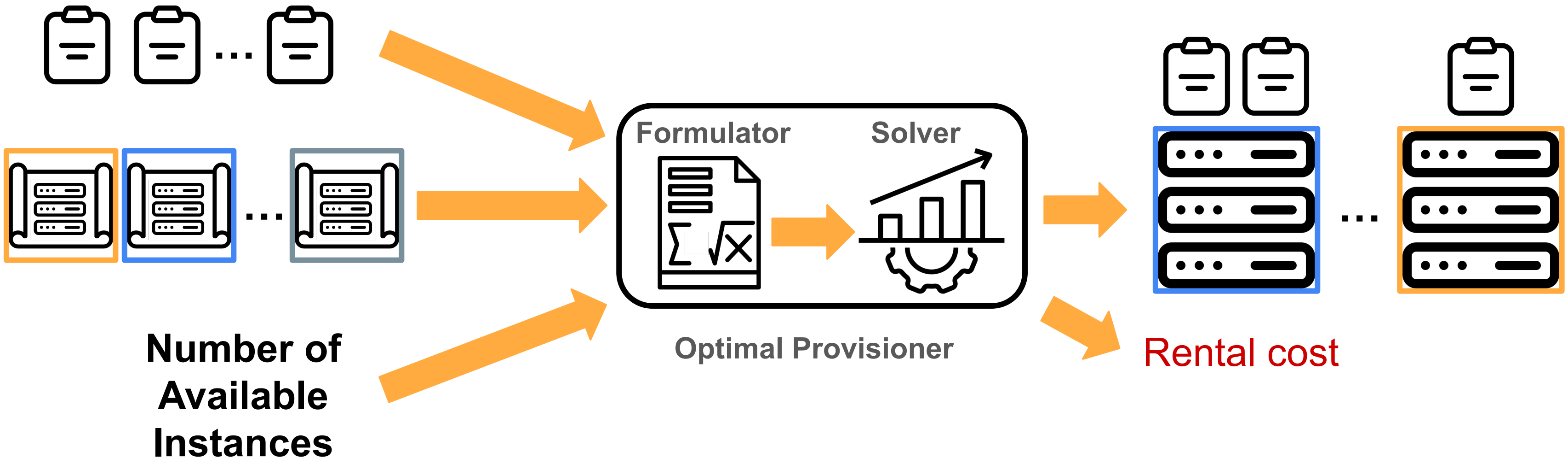
W.I.P. Currently not in use



System model (3)

- Set of “plain” instances on which to place the jobs
- Blueprint vs instance:
  - A blueprint defines a **machine type**
    (i.e., an AWS EC2 catalog entry, an OpenStack flavor)
  - An instance is the **realization/instantiation** of a blueprint
    (i.e., the EC2 instance, the Openstack server)

Optimal Provisioner Workflow



- problem formulator: Pyomo
- MILP solver: Gurobi

Decision Variables

$$\forall j \in J, \forall i \in I : x_{j,i} = \begin{cases} 1 & \text{Job } j \text{ is placed on Instance } i \\ 0 & \text{otherwise} \end{cases}$$

$$|\{x\}| = N_J \cdot N_I$$

$$\forall i \in I : y_i = \begin{cases} 1 & \text{Instance } i \text{ is in use} \\ 0 & \text{otherwise} \end{cases} \equiv \bigvee_{j \in J} x_{j,i}$$

$$|\{y\}| = N_I$$

$$\forall i \in I, \forall b \in B : z_{i,b} = \begin{cases} 1 & \text{Instance } i \text{ is of Blueprint b} \\ 0 & \text{otherwise} \end{cases}$$

$$|\{z\}| = N_I \cdot N_B$$

$$\forall i \in I : \theta_i \in \mathbb{R}^+ = \text{Cost of running Instance } i \text{ (in dollar)}$$

$$|\{\theta\}| = N_I$$

Problem Constraints (1)

- Avoid double placement
- Assign a blueprint type only if the instance is in-use

$$\sum_{i \in I} x_{j,i} = 1 \qquad \forall j \in J$$

$$x_{j,i} \leq y_i \qquad \forall j \in J, \forall i \in I$$

$$\sum_{j \in J} x_{j,i} \geq y_i \qquad \forall i \in I$$

$$\sum_{b \in B} z_{i,b} = y_i \qquad \forall i \in I$$

SCUOLA SUPERIORE SANT'ANNA PISA

Problem Constraints (2)

- Characterization of the monetary decision variables

$$\theta_i \geq x_{j,i} \cdot D_j \cdot \sum_{b \in B} z_{i,b} \cdot C_b \qquad \forall j \in J, \forall i \in I$$

Problem Constraints (3)

- Satisfy hardware requirements

$$\sum_{j\in J} R_j^{CPU} \cdot x_{j,i} \leq \sum_{b\in B} C_b^{CPU} \cdot z_{i,b} \qquad \forall i \in I$$

$$\sum_{j\in J} R_j^{MEM} \cdot x_{j,i} \leq \sum_{b\in B} C_b^{MEM} \cdot z_{i,b} \qquad \forall i \in I$$

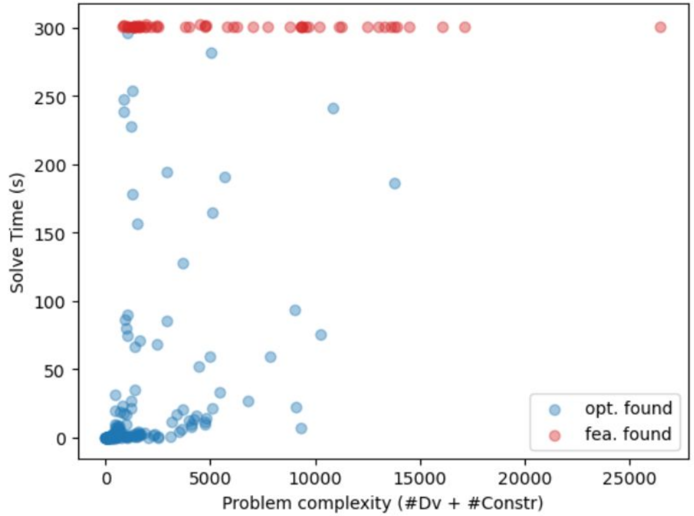
$$x_{j,i} \leq \sum_{b\in B} (C_b^{GPU} \cdot z_{i,b}) + (1 - R_j^{GPU}) \qquad \forall j \in J, \forall i \in I$$

Problem Formulation

$$
\begin{aligned}
&\text{minimize} && \sum_{i\in I}\theta_i && \\
&\text{subject to} && \sum_{i\in I} x_{j,i} = 1 && \forall j \in J \\
& && x_{j,i} \leq y_i && \forall j \in J, \forall i \in I \\
& && \sum_{j\in J} x_{j,i} \geq y_i && \forall i \in I \\
& && \sum_{b\in B} z_{i,b} = y_i && \forall i \in I \\
& && \sum_{j\in J} R_j^{CPU} \cdot x_{j,i} \leq \sum_{b\in B} C_b^{CPU} \cdot z_{i,b} && \forall i \in I \\
& && \sum_{j\in J} R_j^{MEM} \cdot x_{j,i} \leq \sum_{b\in B} C_b^{MEM} \cdot z_{i,b} && \forall i \in I \\
& && x_{j,i} \leq \sum_{b\in B} (C_b^{GPU} \cdot z_{i,b}) + (1 - R_j^{GPU}) && \forall j \in J, \forall i \in I \\
& && \theta_i \geq x_{j,i} \cdot D_j \cdot \sum_{b\in B} z_{i,b} \cdot C_b && \forall j \in J, \forall i \in I \\
& && x_{j,i} \in \{0,1\} && \forall j \in J, \forall i \in I \\
& && y_i \in \{0,1\} && \forall i \in I \\
& && z_{i,b} \in \{0,1\} && \forall i \in I, \forall b \in B \\
& && \alpha \in \{0,1\} &&
\end{aligned}
$$

Experiments: scalability (cost only)

- #JOBS in [1:50]
- #BLUEPRINTS in [1:10]
- #INSTANCES in {#JOBS/3, #JOBS/4, #JOBS/5}



Server (56 cores)

What now?

- Experiment with realistic job set (i.e., not randomly generated)
- Experiment with warm-start to speed-up solve time
- Compare with heuristics and approximation algorithms
  - A feasible solution may still be “enough” to make a “good” provisioning decision

What’s next?

- Integration with GlideinWMS
  - (+comparison with current heuristic, which only take into account the CPU requirement)
- Multi-objective optimization (cost + time, together)

Questions?