Autonomic Management of Idle HPC Resource Harvesting

LIG WAX GLSI

Quentin GUILLOTEAU,* Éric RUTTEN,* Bogdan ROBU,**
Olivier RICHARD*

*Université Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
**Université Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab

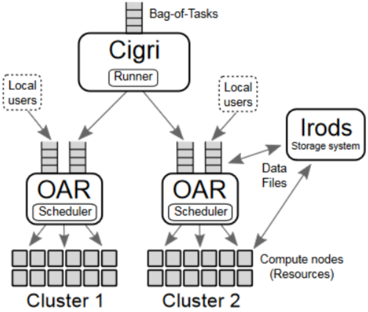
2022-06-30

Context: High Performance Computing

Idle HPC Resources $\Longrightarrow$ Lost Computing Power $\rightsquigarrow$ **How to Harvest** ?

One Solution: *CiGri*

- **bag-of-tasks**: many, multi-parametric
- **Best-effort Jobs**: Lowest priority
- **Objective**: Collect grid idle resources



Problem

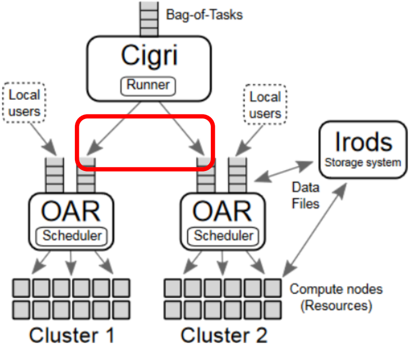
$\nearrow$ Harvesting $\Longrightarrow$ $\nearrow$ Perturbations (e.g., I/O) $\rightsquigarrow$ **Trade-off**

$\hookrightarrow$ Unpredictability $\Longrightarrow$ **runtime management**

CiGri: Submission Loop (1/2)

Algorithm 1: Current Solution

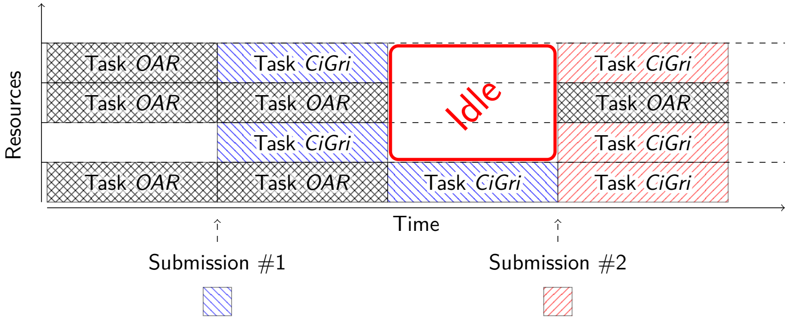
rate = 3;
increase_factor = 1.5;
while *tasks not executed in b-o-t* **do**
  if *no task running* **then**
    submit *rate* tasks;
    $rate = \min(rate \times increase\_factor, 100)$;
  end
  while *nb of tasks running > 0* **do**
    sleep during 30 sec;
  end
end



CiGri: Submission (2/2)

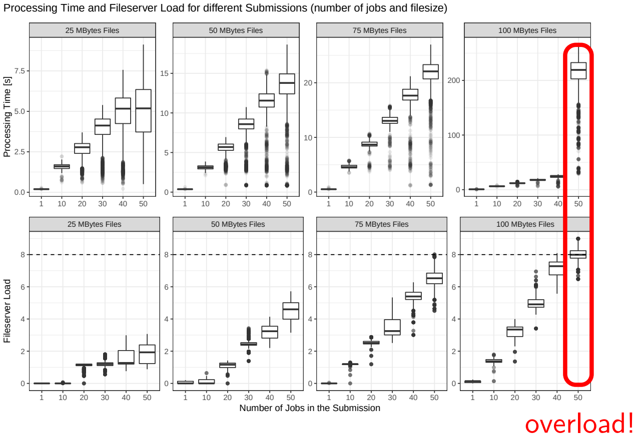
The Issue

Must wait for termination of the previous submission to submit again
↪ reduce overload but introduce **underutilization** of the resources



Degradation of the File System Performances

$\nearrow$ Jobs $\Longrightarrow$ $\nearrow$ I/O $\Longrightarrow$ $\nearrow$ Delay for users $\rightsquigarrow$ **Perturbations**



Sensor

- `loadavg`
- linear relation
- shows limits of FS
- estimation of perturbations

Runtime management

Autonomic Computing and the MAPE-K Loop

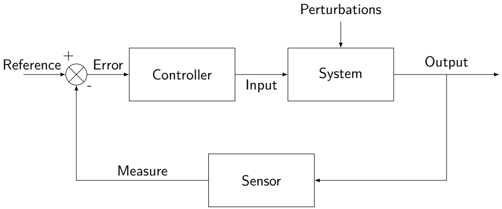
Auto-regulating Systems given **high-level objectives**

<u>Phases</u>: **M**onitor $\rightsquigarrow$ **A**nalyse $\rightsquigarrow$ **P**lan $\rightsquigarrow$ **E**xecute (with **K**nowledge)

Control Theory (Feedback Control Loop)

Regulate the behaviour of dynamical systems

$\hookrightarrow$ Interpretation of the MAPE-K Loop



Our Global Problem and Objectives

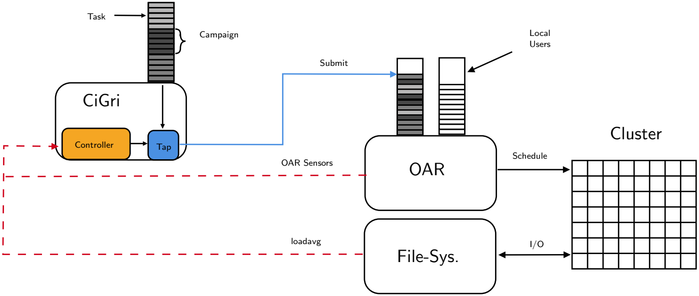
Objective

Harvest Idle Resources in a **non-intrusive** way

- max cluster utilization
- min perturbations

Means

- Instrumentation
  - Actuator: #jobs to submit, ...
  - Sensor: RJMS WQ, FS Load, ...
- Controllers (PID, RST, MFC, ...)
- Experimental Validation



PI: What are we looking for

First, **a Model ...** (i.e., how does the system behave (Open-Loop))

$$\mathbf{y}(k+1) = \sum_{i=0}^{k} a_i \mathbf{y}(k-i) + \sum_{j=0}^{k} b_j \mathbf{u}(k-j)$$

... then **a (PID) Controller** (i.e., the Closed-Loop behavior)

$$Output = \mathbf{K}_p \times Error_k + \mathbf{K}_i \times \sum_k Error_k + \mathbf{K}_d \times (Error_k - Error_{k-1})$$

Sensors & Actuators

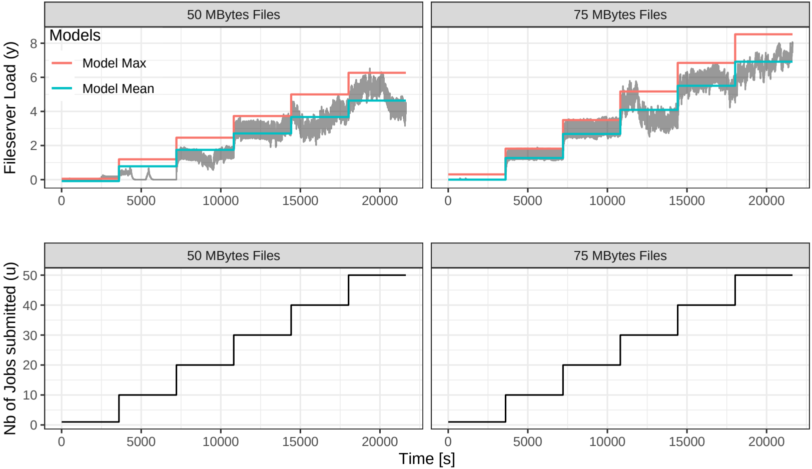
- Actuator: #jobs to sub $\rightsquigarrow$ $\mathbf{u}$
- Sensor: FS Load $\rightsquigarrow$ $\mathbf{y}$
- Error: $Reference - Sensor$

Method

1. Open-Loop expe (fixed $\mathbf{u}$)
2. Model parameters (a_i, b_j)
3. Choice controller behavior $(\mathbf{K}_*)$

PI: Open-Loop and Identification

System Identification and (Linear) Model Fitting

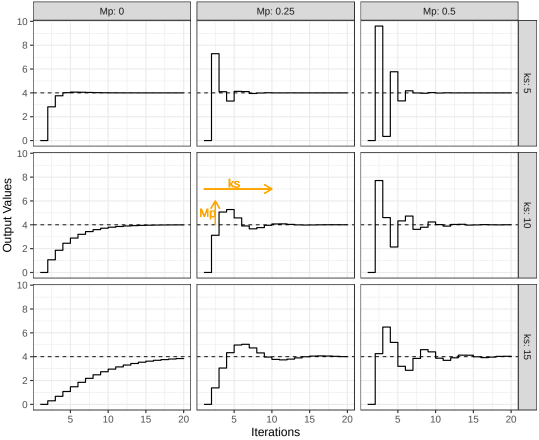


$$y_{ss} = \alpha + \beta_1 f + \beta_2 u + \gamma f u$$

PI: Closed-Loop Behavior

Open-Loop Experiments $\Longrightarrow$ Model (1st order) $\mathbf{y}(k+1) = a\mathbf{y}(k) + b\mathbf{u}(k)$ $\Longrightarrow$ Controller Gains $\mathbf{K}_p, \mathbf{K}_i, \mathbf{K}_d,$

Closed loop behaviour of our system for different values of (ks, Mp)



Controller Gains are ...

functions of the model and

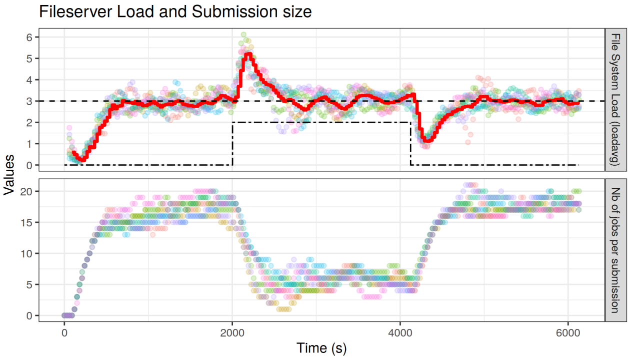
- k_s: max **time** to steady state
- M_p: max **overshoot** allowed

Non-Intrusive Harvesting

- no overshoot
- but "fast" response

Experimental Setup

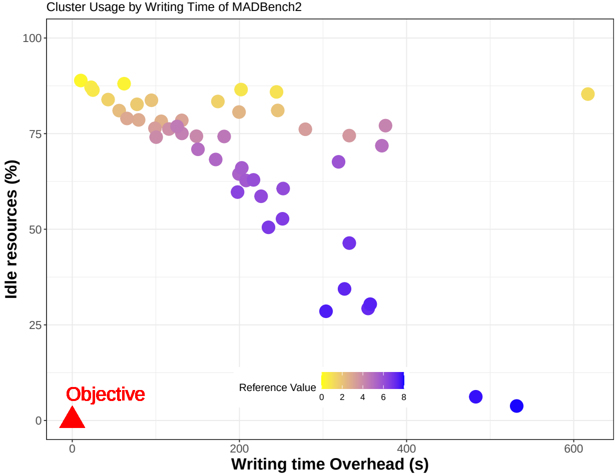
- Experiments done on Grid'5000
- Emulation of a 100 node cluster
- 2 Intel Xeon E5-2630 v3
- CiGri jobs: `sleep` + write



Synthetic Load

- Pure step
- Observe the ctlr behavior:
  - response
  - oscillations

Trade-Off: Harvesting vs. Perturbing



$\nearrow$ Harvesting $\Longrightarrow$ $\nearrow$ Perturbations $\rightsquigarrow$ **Trade-off** (Reference Value)

Conclusion & Perspectives

Reminder of the Objective

Collect **max idle resources** with **min perturbations**

Results

- Dynamical harvesting of the resources
- Trade-off between the harvesting and the perturbations

Perspectives

- Coordination with the scheduler for prediction
- Reusability of the controllers?
- Consider other resource harvesting approaches