Non-Intrusive Harvesting of Idle Grid Resources with a Control Based Approach

13th JLESC Workshop

Quentin GUILLOTEAU* Olivier RICHARD* Eric RUTTEN* Bogdan ROBU**

*Université Grenoble Alpes, CNRS, INRIA, Grenoble-INP, LIG
**Université Grenoble Alpes, CNRS, Grenoble-INP, GIPSA-Lab

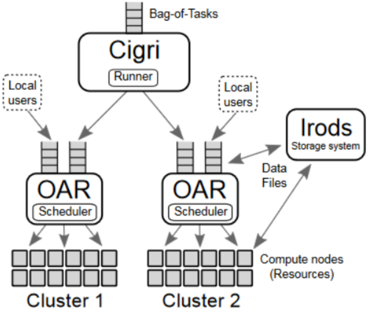
2021-12-15

Context

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**

Runtime Management and Feedback Loops

Autonomic Computing and the MAPE-K Loop

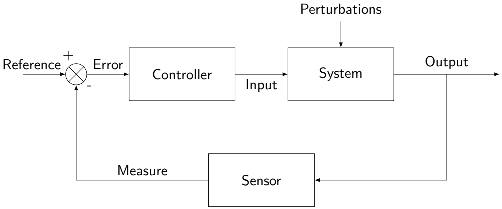
Auto-regulating Systems given **high-level objectives**

Phases: **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 Problem and Objectives

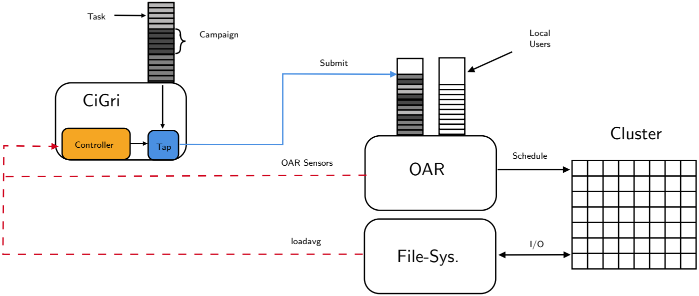
Objective

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



What we are 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 behaviour)

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

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 behaviour $(\mathbf{K}_*)$

Open-Loop Expes: Relation between Input & Output

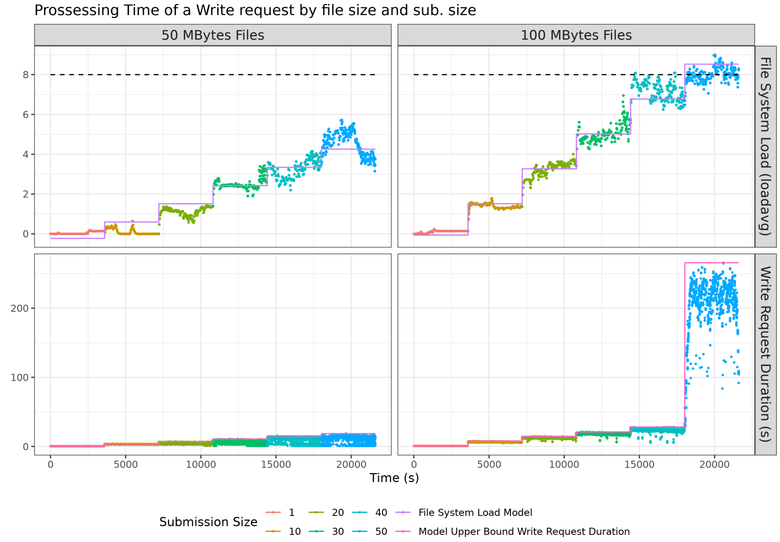


Figure: Identification of the **Model** and the **Degenerative Cases**

Parameters of the Controller

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

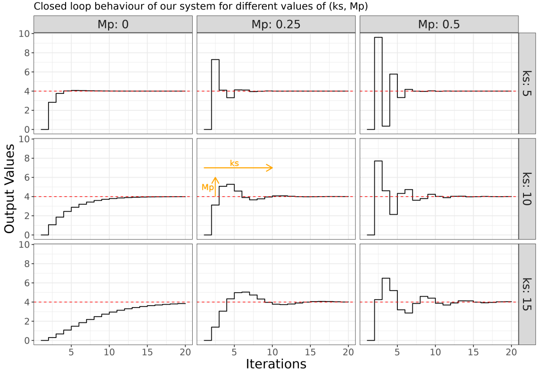


Figure: Closed-Loop Behaviour for different Parameters

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 & Synthetic Load

Experimental Setup

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

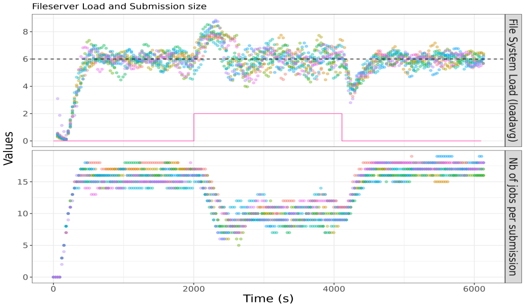


Figure: Controller Response to a Synthetic Load

Synthetic Load

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

Trade-Off: Harvesting vs. Perturbating

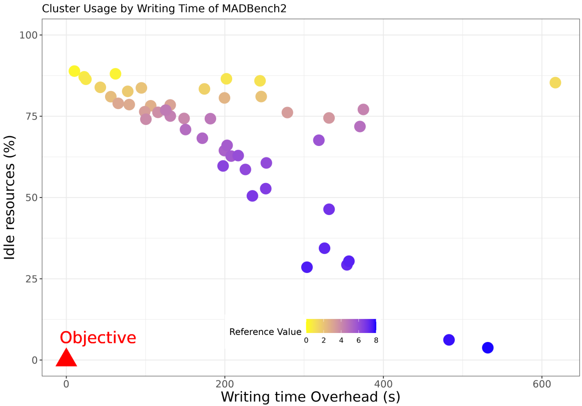


Figure: Relation between the Harvesting and the Perturbations based on the Reference value (**bottom-left is better**)

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

Conclusion and 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 & Cooperations Opportunities

- Coordination with the scheduler (OAR)
- Reproductibility of experiments (Nix)
- Consider other resource harvesting approaches