TOWARDS A CONTROL-THEORY APPROACH FOR MINIMIZING UNUSED GRID RESOURCES

Emmanuel Stahl †, Agustín Gabriel Yabo ‡, Olivier Richard ‡, Bruno Bzeznik ‡, Bogdan Robu †, and Eric Rutten ‡

†Université Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble France
‡Université Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble France

June 11, 2018

1 MOTIVATIONS

Outline

1. **Motivations**
2. **Background**
3. **Analysis and modeling**
4. **Experimental validation**

Motivations

- **need for autonomic administration in HPC**
  - **variability** at execution time
    e.g., access times (cache); values in data (iterations depth)
  - less predictable → requires more **runtime management**
- **context : CiGri**
  - simple, lightweight, scalable and fault tolerant **grid system**
  - **exploits unused resources** of clusters to run smaller jobs
  - injected as much as global system allows (i.e. avoid overloading)
- **feedback loop**
  - measure system condition
  - decide **whether** unused resources useable
  - decide **appropriate amount** of jobs to inject

Control for Computing Systems

- **Autonomic Computing**, **self-adaptive** software and hardware
  - **feedback loop** for the control of adaptations
  - complementary to Computing for Control Systems design in embedded, real-time systems
- **typical examples** in data-centers (Cloud, HPC), embedded systems
  - sensors : CPU load, occupation of queues, end of task, ...
  - actuators : add/remove servers, DVFS, start task, ...
  - objectives : regulate resources (energy, memory) & QoS, stay in safe logical states, ...
- **new application domain for Control Techniques and Theory**
  - lack of established models usable for control of computing
  - different aspects : discrete, continuous, stochastic

Contributions

- **first results** in regulating injection of jobs into a grid system
  - to maximize its utilization (minimize unused resources)
  - while avoiding overshoot of bounds leading to overload
- **method** in several phases
  1. **analysis** of the system and the overload problems to be tackled
     identification of its dynamics
  2. **design of a feedback controller**, with P and PI regulation
  3. **implementation** and experimental results,
     comparing new control scheme with previous, *ad hoc* solution
- **perspectives** multi-disciplinary
  - control : more elaborate techniques
  - HPC : more resources and features autonomically managed
  - Autonomic Computing : coordination of locally managed systems

2

BACKGROUND

Outline

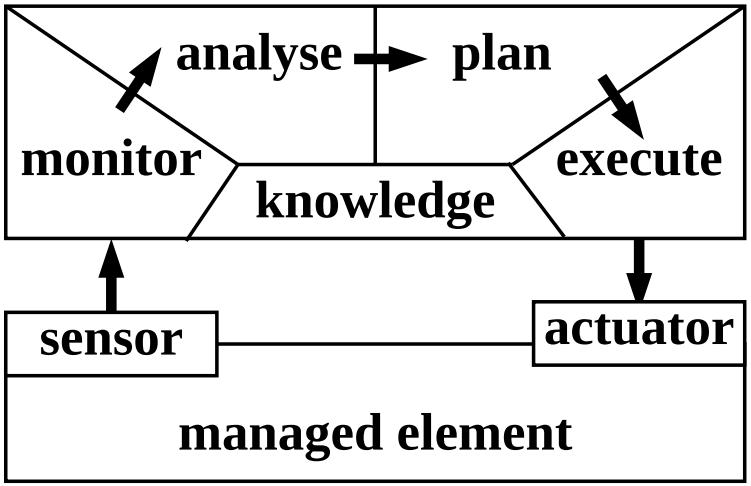
1. **Motivations**
2. **Background**
3. **Analysis and modeling**
4. **Experimental validation**

Feedback loops (i)

Autonomic Computing and self-adaptive systems :

self-* properties : self-configuration, -optimization, -healing, -protection

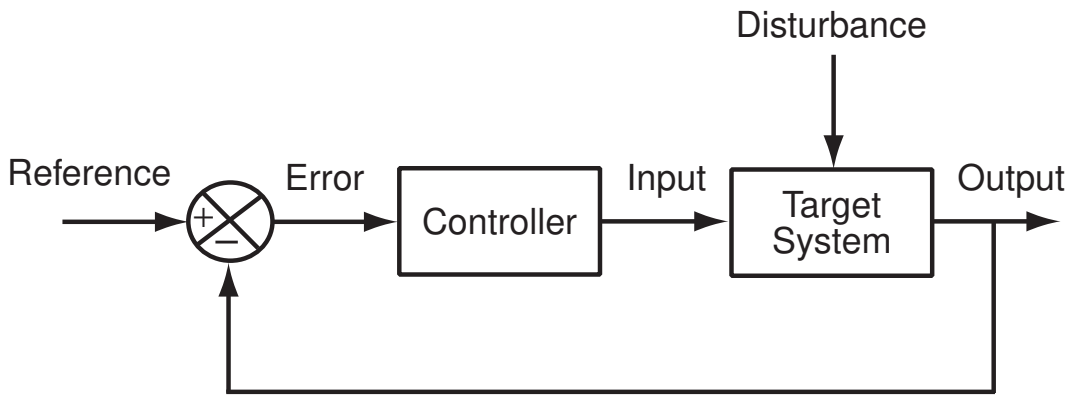
MAPE-K feedback loop :



- Monitor (through probes)
- Analyze (extract relevant information)
- Plan (transform decisions into adaptation actions)
- Execute (perform adaptation actions)
- using Knowledge (reified representation, data-base)

Feedback loops (ii)

Control-based approaches



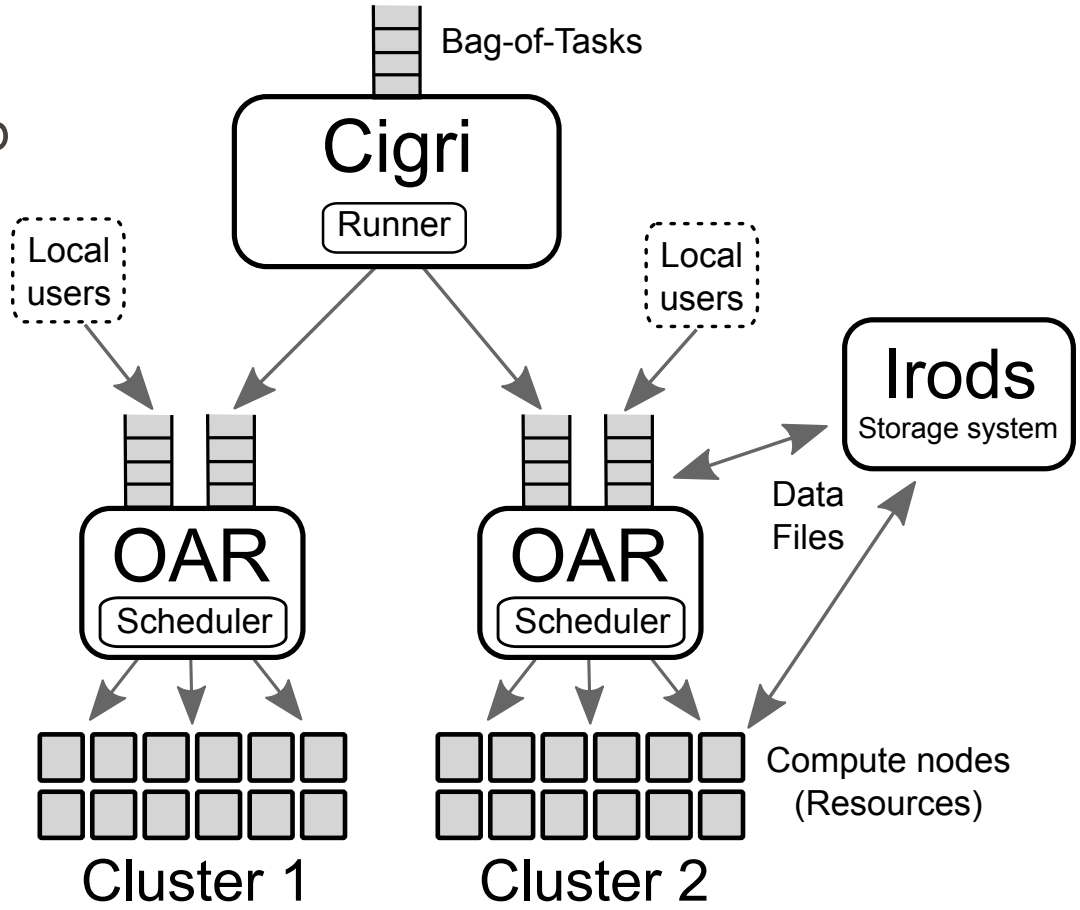
- *output* : measurable indicator of system state
- *reference* : objective value to which to drive the *output*.
- *error* : difference between *reference* and *output*
- *input* : action to modify *output* regulated to match *reference*
- *disturbance* : affects system's dynamics in unpredictable way.

Controllers

- **Basic control functions** : the classical PID
  - P : proportional to the error
  - I : integral over time of past values ; memory
  - D : derivative : current "speed" ; possible future trends
- **others**
  - MPC : model predictive control
  - discrete event systems : Petri nets, automata, ...
  - stochastic : Markov Decision Processes, ...
  - ...

The CiGri grid system

- **Overall architecture**
  - **OAR** : Ressource Job Management ; batch scheduler
  - **Irods** : distributed storage management
  - **CiGri** : exploits unused resources in clusters



Administration problems considered

- **job submission management**
  tap mechanism implemented in the Runner component of CiGri

Algorithm 2.1: JOB SUBMISSION( )

main
rate ← 3
increase factor ← 1.5
while jobs left in the bag-of-tasks
do
- launch *rate* amount of jobs
- **while** jobs running > 0 **and** *not*(*timeout*)
  - **do** sleep
- *rate* ← min(*rate* ∗ *increase factor*, 100)

essentially : submit with acceleration until bound

- **Other side problems**
  - Cluster overload, too many jobs: a *stress factor* is defined
  - Irods overload, when too many requests are submitted.
  - Particular storage resource overload, individual server.

3

ANALYSIS AND MODELING

Outline

1. **Motivations**
2. **Background**
3. **Analysis and modeling**
4. **Experimental validation**

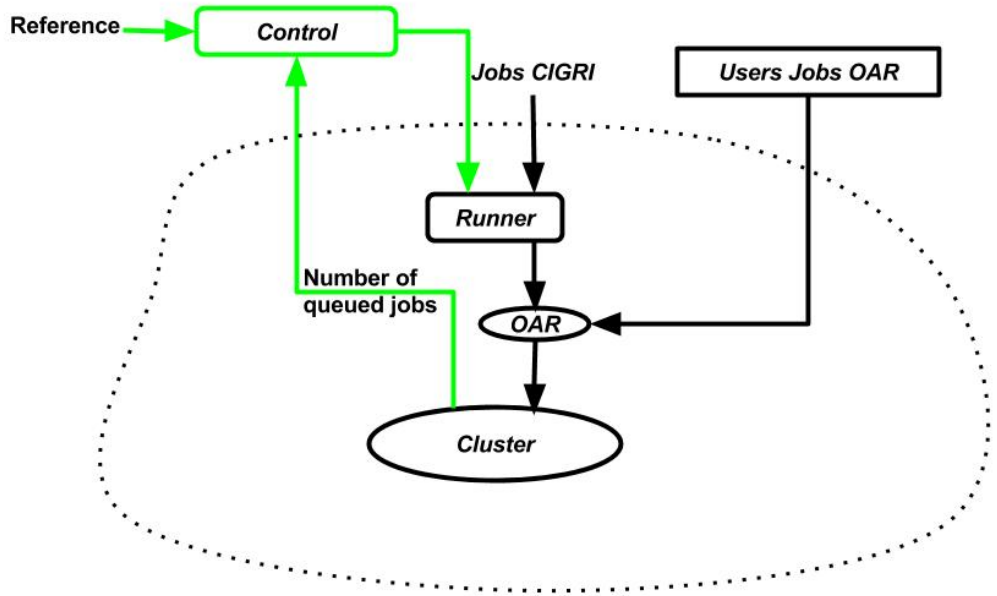
Method

classical approach in control :

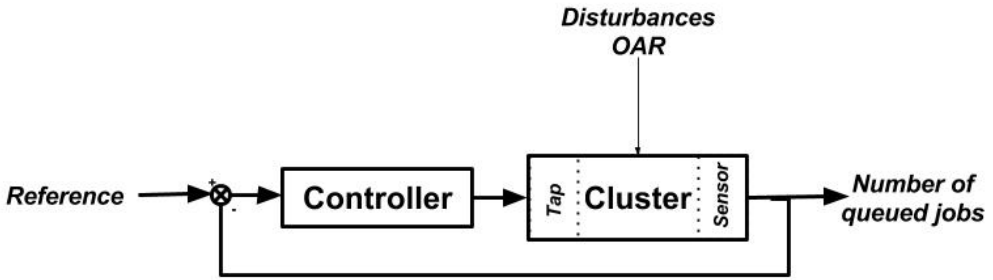
1. **informal analysis of the system**
   extract the feedback loop :
   identify **sensors and actuators**
   formulate the **control objective**
2. **open loop testing**
   measures in the absence of regulation
   acquire some **quantitative parameters** of the system behavior
3. **design of the control law**
   heart of the Autonomic Manager
   in this case simple **P and PI regulators**

Informal analysis of the system

- **considered administration loop**



- **control loop**

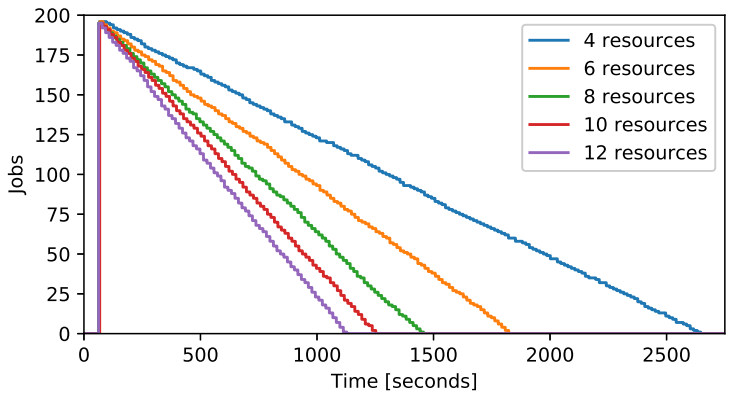


Open loop testing

to have a quantitative idea about the natural behavior of the system

Open loop testing

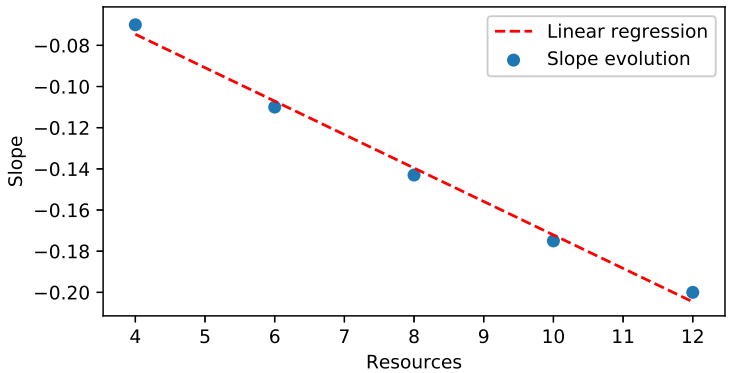
for model identification



- number of jobs is fixed (200 jobs)
  number of resources varies

  the larger the resources, the steeper the slope of queue drain

- number of resources is fixed
  number of jobs in the queue varies



  relation between slope of the draining queue and number of resources

Design of the control law (i)

A Proportional controller :

$$tap = K_p * error$$

computation of the gain K_p : we approximate system behavior

as a second order differential equation

expressed through its Laplace transformation

$$H(p) = 1 - \frac{K_p}{(1 + T_1\, p)(1 + T_2\, p)}$$

Design of the control law (ii)

A PI (Proportional Integral) controller

$$tap(t) = K_p * error(t) + K_i \int_0^t error(\tau) d\tau$$

advantage w.r.t. P controller :

- no steady state error (i.e. more exact)
- less oscillations

approximating integral of the error as cumulative sum of previous values

$$tap(t) = K_p\, error(t) + K_i \sum_{k=0}^{t} error(k) \Delta t_k$$

4

EXPERIMENTAL VALIDATION

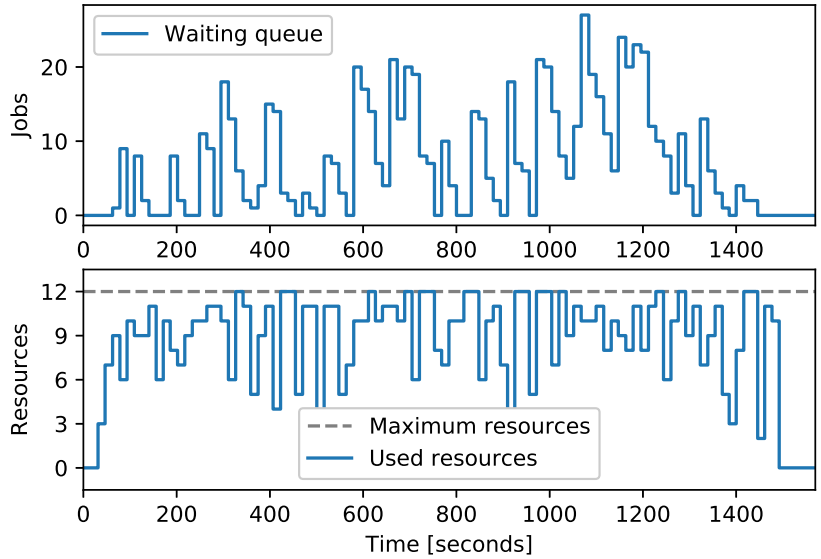
Outline

1. **Motivations**
2. **Background**
3. **Analysis and modeling**
4. **Experimental validation**

Open loop and P control

- **Open loop behavior (i.e. original algorithm)**

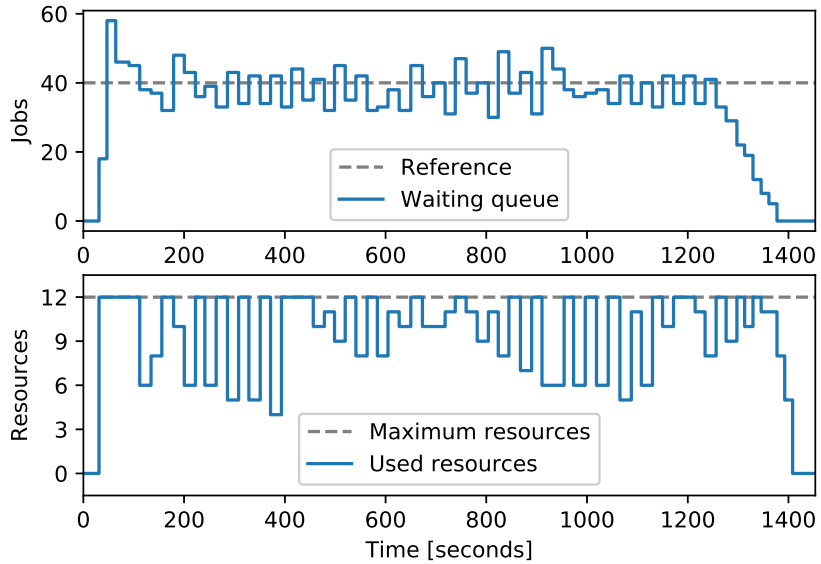
  400 jobs campaign
  running on a 12 resources cluster



- **Closed loop behavior with the P controller**

  shorter completion time

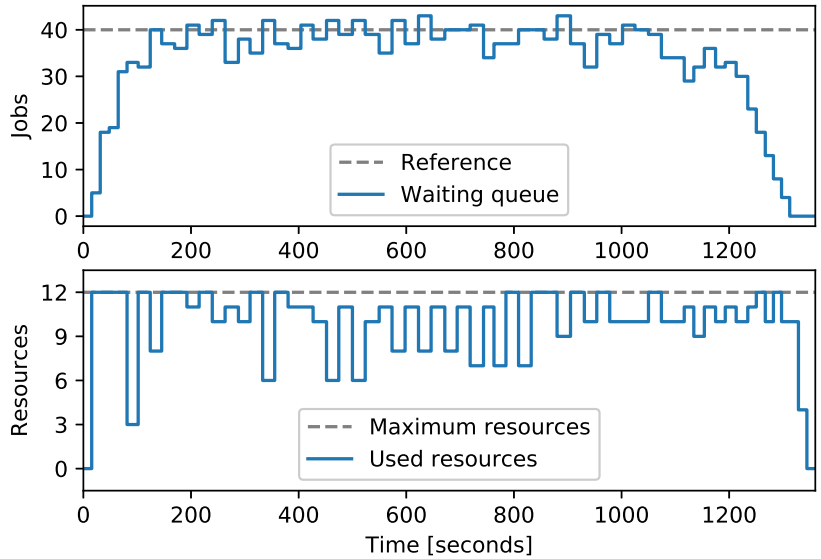
  average improvement
  in cluster utilization of 5%



PI control

- **Closed loop behavior with PI controller**
  shorter completion time

  average improvement
  in cluster utilization of 8%



- **Comparison of performance** among the solutions
  for the same campaign of 400 jobs in a 12 resources cluster

	w/o control	P control	PI control
Total time	1445 sec	1361 sec	1313 sec
Cluster usage	77%	82%	85%

Perturbations

Test in presence of perturbations :

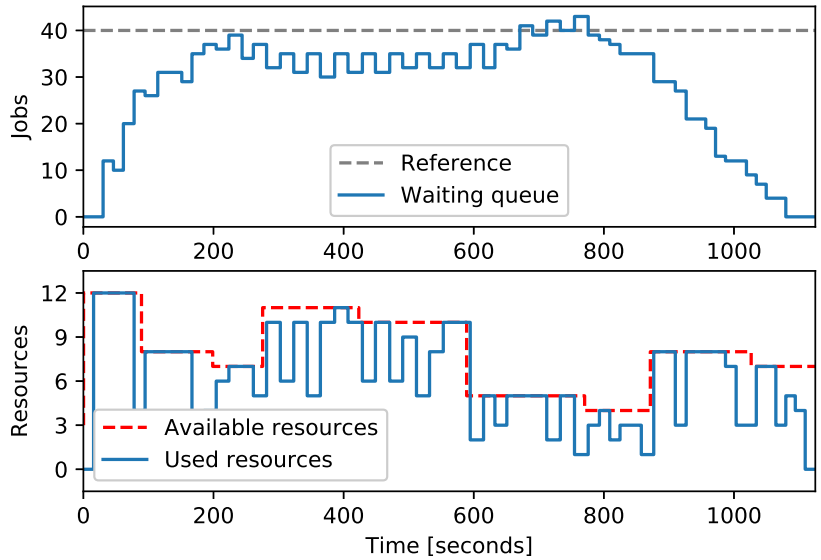
dealing with unknown external factors

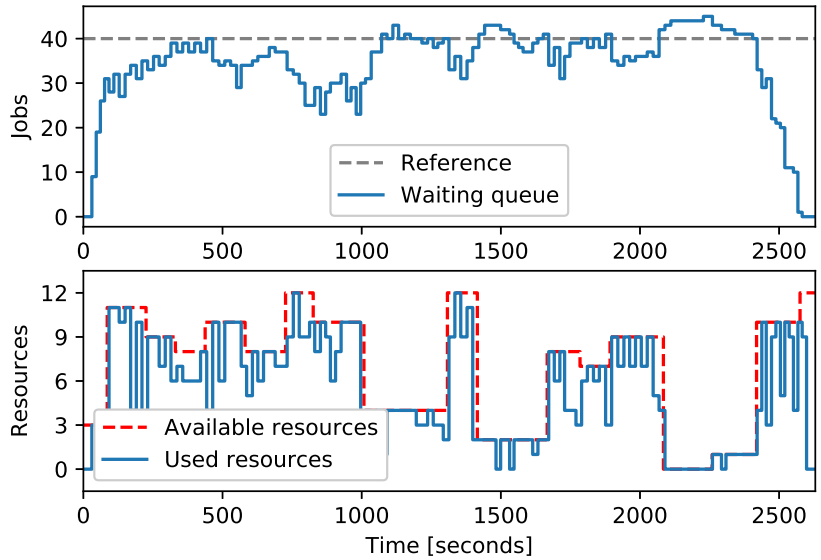
e.g. submitting higher priority jobs

to dynamically vary the amount

of available resources

- **for a 200 jobs campaign**
- **for a 400 jobs campaign**





Outline

1. **Motivations**
2. **Background**
3. **Analysis and modeling**
4. **Experimental validation**
5. **Conclusion and Perspectives**

Conclusion and Perspectives

- **Results**
  - **preliminary results** addressing autonomic administration in HPC
  - automated resource management **using classical Control Theory**
  - experimentally **validated**
- **Perspectives**
  multi-disciplinary
  - more elaborate **control** (MPC, predictive)
  - other **HPC** problems e.g. storage overload
  - **multiple loops, hierarchies & their coordination**
  - complementarity w.r.t. other techniques (scheduling, machine learning, ...)