Sustaining Performance While Reducing Energy Consumption: A Control Theory Approach

CTRL-A seminar, Grenoble

Sophie CERF*, Raphaël BLEUSE*, Valentin REIS**, Swann PERARNAU**, Éric RUTTEN*

*Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG

**Argonne National Laboratory

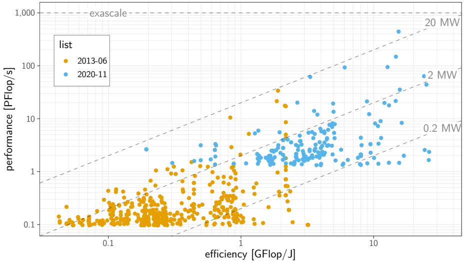
2021-04-13

Energy efficiency in production HPC systems

Current and future HPC systems:

- Toward exascale
- Heterogeneous compute nodes

Green500



Reducing energy consumption while preserving performance

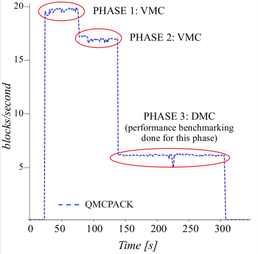
Energy efficiency in production HPC systems

Challenges to performance-per-watt efficiency

- Growing complexity of scientific workloads
- Specificity of applications behavior
- Various processor characteristics and performance
- Exogenous limits on progress

Need for dynamic perspective

- Avoid fine-grained modeling
- Robustness to execution context
- Handling phased behavior
  - data, compute, I/O



(Ramesh et al. 2019)

Dynamic Power Management

Global Objectives

- Sustain execution time
- Minimize energy usage

The Runtime Perspective

- Sustain application progress
- Minimize power usage

Actuator and Sensor

Power regulation DVFS (Imes et al. 2015; Imes et al. 2019)
DDCM (Bhalachandra et al. 2015)
RAPL (David et al. 2010; Rotem et al. 2012)

Application behavior Measuring progress with heartbeats (Ramesh et al. 2019)

Related Works

On power regulation in HPC

Different objective or static schema

(Eastep et al. 2017) application-oblivious monitoring

On using control theory for power regulation

Applications: web servers (Abdelzaher et al. 2008), cloud (Zhou et al. 2016), real-time systems (Imes et al. 2015)

Metrics: RAPL (Imes et al. 2019; Lo et al. 2014)
Progress metric (Santriaji et al. 2016)

Our contribution

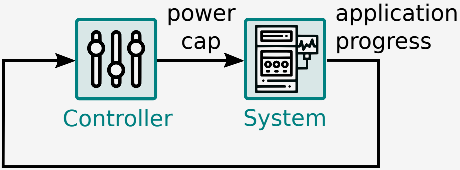
Leveraging **RAPL**'s powercap using **control theory** with **progress** objectives in **HPC** application systems.

Outlines

Autonomic Computing Approach

The Autonomic Computing approach...

- Periodically monitor application progress
- Choosing at runtime a suitable power cap for processors



. . . using Control Theory

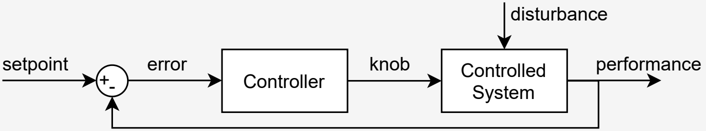
How Non-intrusive supervising

Why Stability, accuracy, transient performance (Hellerstein et al. 2004)

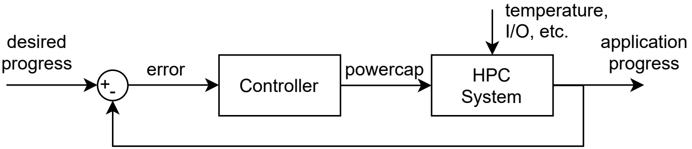
Principle of Control Theory

Feedback loops

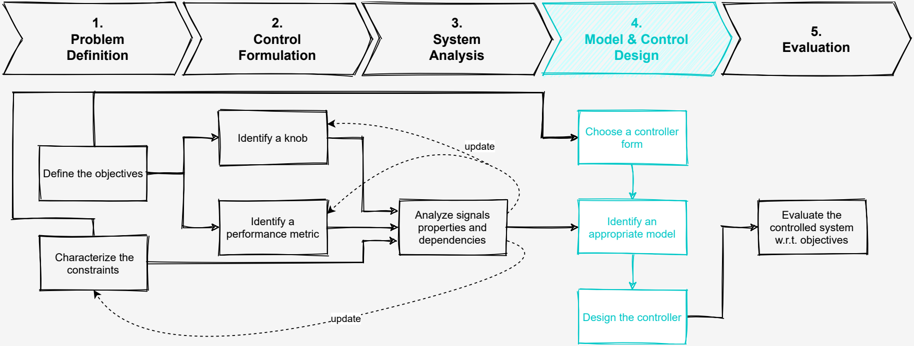
Mesure **performance** and react according to the **error** w.r.t. the desired **setpoint** by leveraging system's **knob**.



Power control in HPC

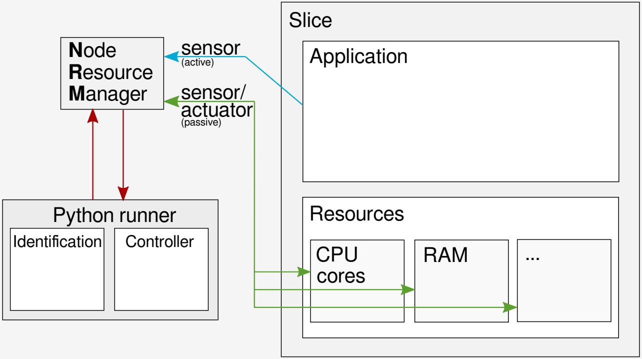


Control Theory Methodology



Software Architecture

Software Stack Argo NRM resource management framework



Platform 3 clusters from Grid5000 with various nb. of sockets

Benchmark STREAM (Desrochers et al. 2016)

Control Formulation

From post-mortem metrics to dynamic measures and knobs

Power actuator

RAPL's power limitation (David et al. 2010):

$$\mathrm{pcap}(t_i)$$

Performance sensor

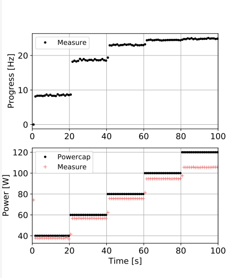
Application's progress (Ramesh et al. 2019): median heartrate

$$\mathrm{progress}(t_i) = \underset{\forall k,\, t_k \in [t_{i-1}, t_i[}{\mathrm{median}} \left(\frac{1}{t_k - t_{k-1}} \right)$$

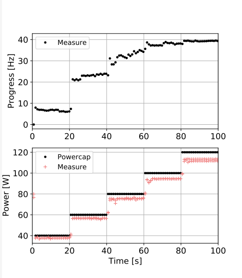
Progress is correlated with execution time.
Pearson coefficient resp.0.97, 0.80 and 0.80 for `gros`, `dahu` and `yeti`.

Uncontrolled System Analysis

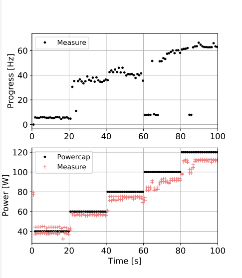
Knob variations impact on performance metric



gros

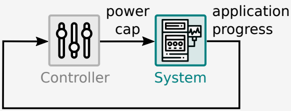


dahu



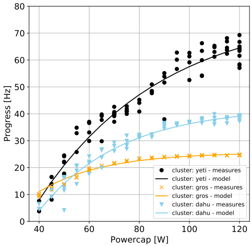
yeti

- Poor RAPL actuator accuracy
- Power cap leverages progress with
  - non linearities, saturations, and noise
- Presence of external factors



Modeling

Static Characteristic: looking at the time averaged behavior



$$\text{progress} = K_L \left(1 - e^{-\alpha(a \cdot \text{pcap} + b - \beta)}\right)$$

a, b: characterizing RAPL actuator

K_L, α, β: cluster- and application-specific

- Handling non-linearity:

$$\text{pcap}_L = -e^{-\alpha(a \cdot \text{pcap} + b - \beta)}$$

$$\text{progress}_L = \text{progress} - K_L$$

Dynamic perspective

$$\text{progress}_L(t_{i+1}) = \frac{K_L(t_{i+1} - t_i)}{t_{i+1} - t_i + \tau} \cdot \text{pcap}_L(t_i) + \frac{\tau}{t_{i+1} - t_i + \tau} \cdot \text{progress}_L(t_i)$$

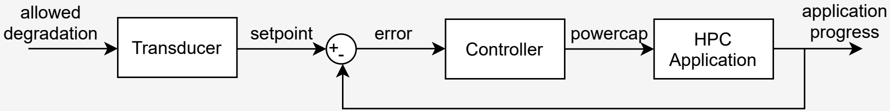
Shape set by control theory, parameters optimized offline

Control Law

Objective Allowed degradation ϵ

Setpoint $\text{setpoint} = (1 - \epsilon) \cdot \text{progress}_{\text{max}}$

Error $e(t_i) = \text{setpoint} - \text{progress}(t_i)$



Proportional Integral Controller

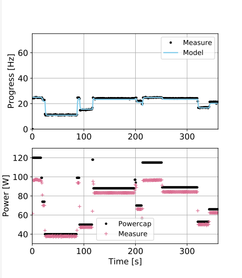
$$\text{pcap}_L(t_i) = (K_I(t_i - t_{i-1}) + K_P) \cdot e(t_i) - K_P \cdot e(t_{i-1}) + \text{pcap}_L(t_{i-1})$$

K_P and K_I are based on the model parameters

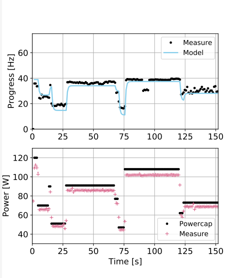
Experimental Evaluation

Measure of the Model Accuracy

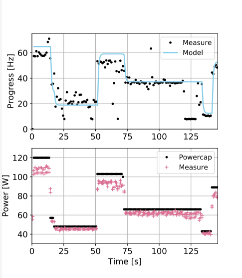
Not a prediction model but used to tune the controller



gros



dahu



yeti

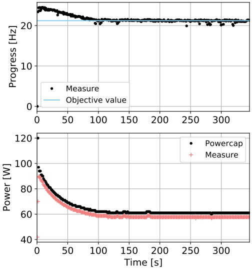
Observations

- Good accuracy.
- The model performs better on clusters with few sockets.

Experimental Evaluation

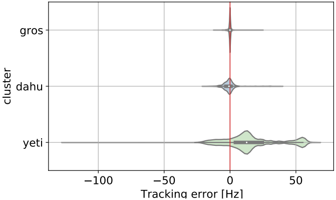
Time-local behavior

Illustration



- Progress reaches the objective level $\epsilon = 0.15$

Analysis



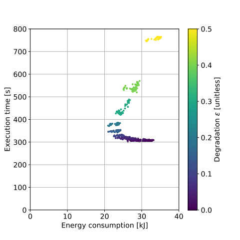
`gros`, `dahu` unimodal, centered near 0, narrow dispersion

`yeti` 2nd mode (model limitations at approx. 10Hz)

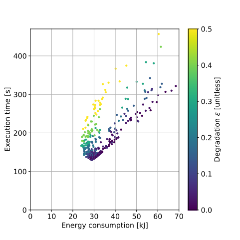
Experimental Evaluation

Post-mortem analysis

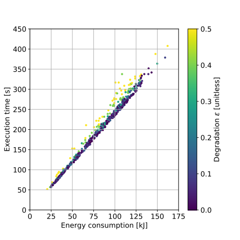
12 degradation levels, min. 30 repetitions each



gros



dahu



yeti

Pareto Front

gros, dahu Family of trade-off from 0% to 15% degradation level
gros with $\epsilon = 0.1$: -22% energy, +7% execution time

yeti no front, no negative impact of the controller

Discussions

Exploring trade-offs

- Easily configured behavior of the controller

Model limitations

- Cluster- and application-specific parameters and model
- Non-linearities
- Unmodeled progress drop
  - nb. of packages, NUMA architecture, exogenous temperature events

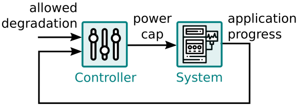
Control solutions considered

- Adaptive Control
- Actuation distribution
- Adding sensors & temperature disturbance anticipation

Conclusion

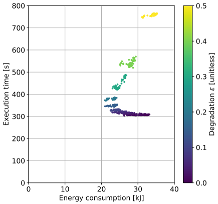
Objective Reducing energy consumption while sustaining performance

Approach Dynamic power regulation using Control Theory



Contributions

- Control methodology for HPC systems
- Offline model identification
- Controller design
- Experimental validation on several clusters



References I

T. Abdelzaher et al., "Introduction to Control Theory And Its Application to Computing Systems," in *Performance Modeling and Engineering* (2008), pp. 185–215.

K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, Second (International Society of Automation, 1995).

S. Bhalachandra et al., "Using Dynamic Duty Cycle Modulation to Improve Energy Efficiency in High Performance Computing," in IPDPS workshops (2015).

H. David et al., "RAPL: Memory Power Estimation and Capping," in ISLPED (2010), pp. 189–194.

S. Desrochers et al., "A Validation of DRAM RAPL Power Measurements," in MEMSYS (2016), pp. 455–470.

J. Eastep et al., "Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions," in ISC, Vol. 10266, Lecture Notes in Computer Science (2017), pp. 394–412.

J. L. Hellerstein et al., *Feedback Control of Computing Systems*, (Wiley, 2004).

C. Imes et al., "CoPPer: Soft Real-time Application Performance Using Hardware Power Capping," in ICAC (2019), pp. 31–41.

References II

C. Imes et al., "POET: A Portable Approach to Minimizing Energy Under Soft Real-time Constraints," in RTAS (2015), pp. 75–86.

D. Lo et al., "Towards Energy Proportionality for Large-Scale Latency-Critical Workloads," in ISCA (2014), pp. 301–312.

S. Ramesh et al., "Understanding the Impact of Dynamic Power Capping on Application Progress," in IPDPS (2019), pp. 793–804.

E. Rotem et al., "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge," IEEE Micro **32**, 20–27 (2012).

M. H. Santriaji and H. Hoffmann, "GRAPE: Minimizing Energy for GPU Applications with Performance Requirements," in MICRO (2016), 16:1–16:13.

Y. Zhou et al., "CASH: Supporting IaaS Customers with a Sub-core Configurable Architecture," in ISCA (2016), pp. 682–694.

Model and Controller Parameters

Description	Notation	Unit	gros	dahu	yeti
RAPL slope	a	[1]	0.83	0.94	0.89
RAPL offset	b	[W]	7.07	0.17	2.91
	α	[1/W]	0.047	0.032	0.023
power offset	β	[W]	28.5	34.8	33.7
linear gain	K_L	[Hz]	25.6	42.4	78.5
time constant	τ	[s]	1/3	1/3	1/3
	τ_{obj}	[s]	10	10	10
lower power limit	$pcap^{MIN}$	[W]	40	40	40
higher power limit	$pcap^{MAX}$	[W]	120	120	120
	τ_{obj}	[s]	10	10	10

Controller Parameters Computation

K_P and K_I are based both on the model parameters K_L and τ and on a tunable parameter τ_{obj} (Åström et al. 1995):

$$K_P = \tau/(K_L \cdot \tau_{\text{obj}})$$

$$K_I = 1/(K_L \cdot \tau_{\text{obj}})$$

with τ_{obj} defining the desired dynamical behavior of the controlled system. The controller is chosen to be nonaggressive:

$$\tau_{\text{obj}} = 10\,\text{s} > 10\tau$$

.