Energy vs Responsiveness Tradeoffs in EASY Backfilling

Millian Poquet

DATAMOVE team
LIG laboratory
Univ. Grenoble Alpes, Inria
millian.poquet@inria.fr

2017–06–27

Context: Computing Platforms

HPC Platforms

- Exascale around 2023
- Energy: locking point

Smaller-Scale Platforms

- ↑ in *small* companies
- Energy: $$

How To Reduce Energy Consumption?

- Energy-efficient machines/cooling system
- DVFS
- **Shutting machines down**
- ...

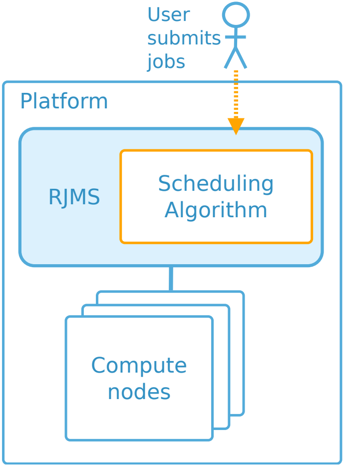
Why focus on the shutdown?

- Can be used on most platforms
- Significant potential gains
- Compatible with DVFS

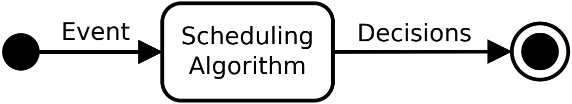
Platform Management

Resources and Jobs Management Systems (RJMS)

- AKA batch scheduler
- Orchestrates resources
  - Implements scheduling policies
  - Manages parallel jobs
  - Enforces energy policy
- Examples: SLURM, OAR, TORQUE, PBS...



Online Scheduling Algorithm



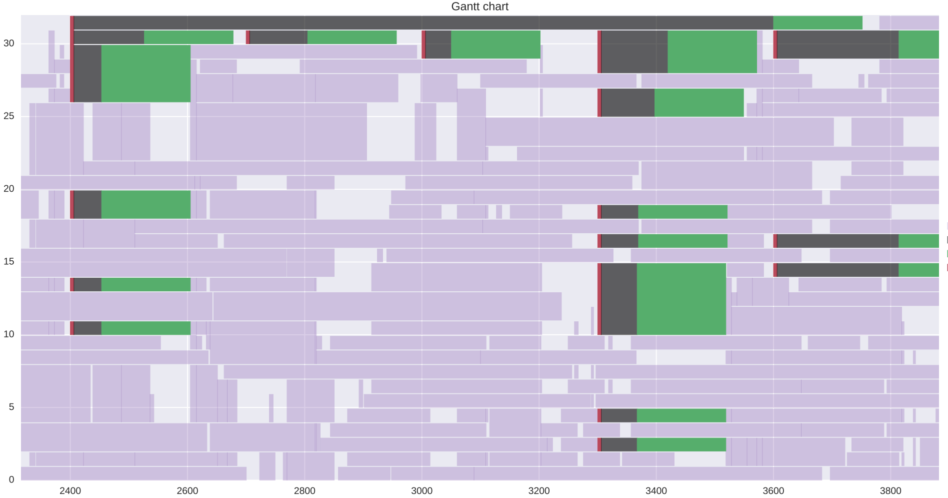
Events

- Job submission/termination
- Resource state alteration (switched ON/OFF, DVFS...)
- (Periodically)

Decisions

- Execute jobs (where?)
- Change resource state (ON, OFF, DVFS...)

Schedule (Gantt Chart)



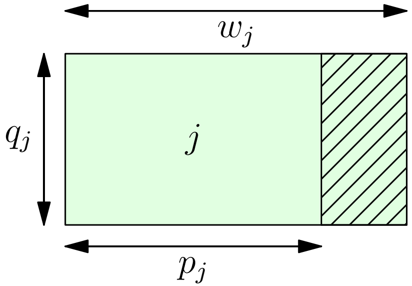
Outline

Workload Definition

$W = \{j_1, j_2, j_3, \ldots\}$. **Unknown** $|W|$

Job j definition:

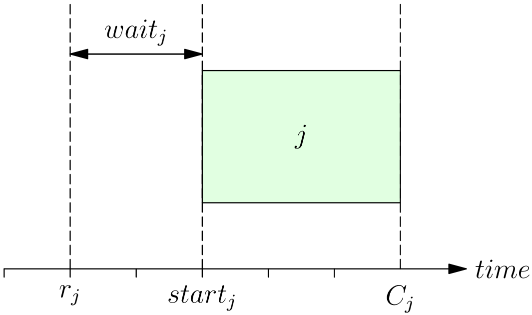
- Submission time r_j (release date). **Unknown** in advance
- Processing time p_j. **Unknown** in advance
- Requested time $w_j \geq p_j$. **Known** at submission time
- Number of requested resources q_j. **Known** at submission time
- ...



More Job-Related Notations

Once the job has been computed:

- Starting time $start_j$
- Completion time C_j
- Waiting time $wait_j = start_j - r_j$



Platform Definition

Platform: ordered set M of identical machines

- $t_{on \to off}$, switching OFF time (s)
- $t_{off \to on}$, switching ON time (s)
- $p_m(t)$, electrical consumption at time t (W)

$$p_M(t) = \sum_m \int_{min(s_j)}^{max(C_j)} p_m(t)\ dt$$

State	Power (W)
computing	p_{comp}
idle	p_{idle}
off	p_{off}
$on \to off$	$p_{on \to off}$
$off \to on$	$p_{off \to on}$

Hypotheses:

- $p_{off} \ll p_{idle} < p_{comp}$
- $p_{off} < p_{* \to *} \leq p_{comp}$

Problem Definition

Input:

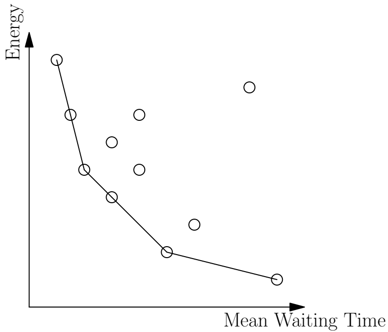
- Workload W of $|W|$ jobs
- Platform M of $|M|$ machines

Compute W on M,
minimizing:

- Total Consumed Energy
- Mean Waiting Time (QoS)

$$E = \sum_{m} \int_{min(s_j)}^{max(C_j)} p_m(t)\ dt$$

$$MWT = \frac{1}{|W|} \sum_{j} wait_j$$



Desired Properties

Results:

- High energy savings
- Low performance loss
- Robustness, predictability...

Constraints:

- Scalability
- No further job knowledge required
- Low #switch
- Ease of implementation

Some Related Work

Theoretical:

- DVFS/shutdown models&algo [Albers, 2010]
- Markov Chains [Herlich and Karl, 2012]

Practical:

- DVFS/shutdown in SLURM [Georgiou et al., 2015]
- Energy budget in EASY [Dutot et al., 2016a]
- Applications [Etinski et al., 2012]

Overprovisioning:

- Max throughput, power budget [Sarood et al., 2014]

Algorithms Overview

- Based on EASY backfilling
- Called on *classical* events **and** every T seconds
- Study interactions of two main mechanisms

Opportunistic Shutdown

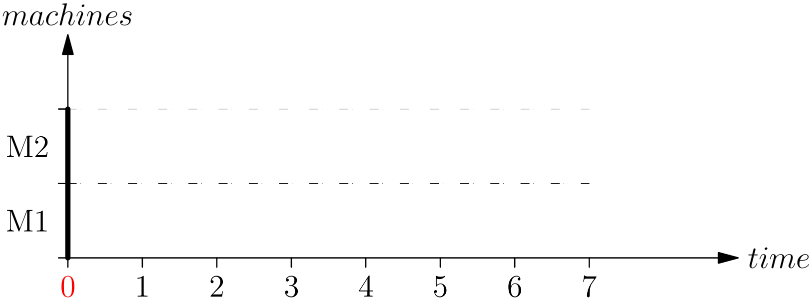
- Machine idle for $t \geq t_{idle}$ seconds $\rightarrow$ switched off

Adjusting the number of usable machines

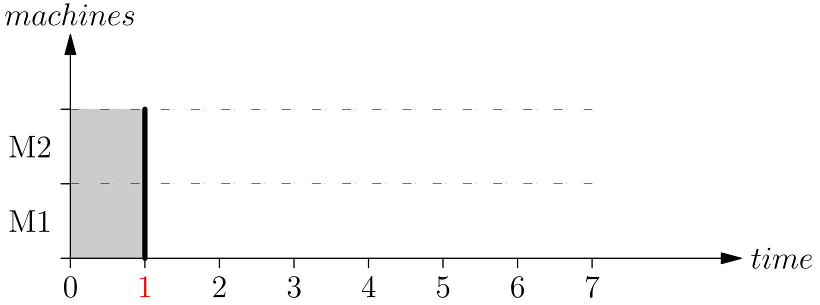
- Statically, avoid using more than $f \cdot |M|$ machines
- Dynamically, depending on system *unresponsiveness*

If the priority job **do requires** more machines, they will be switched-on.

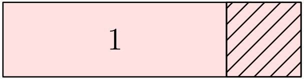
Easy Backfilling Example



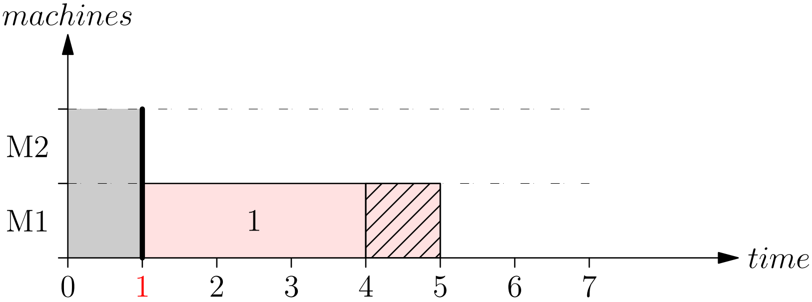
Easy Backfilling Example



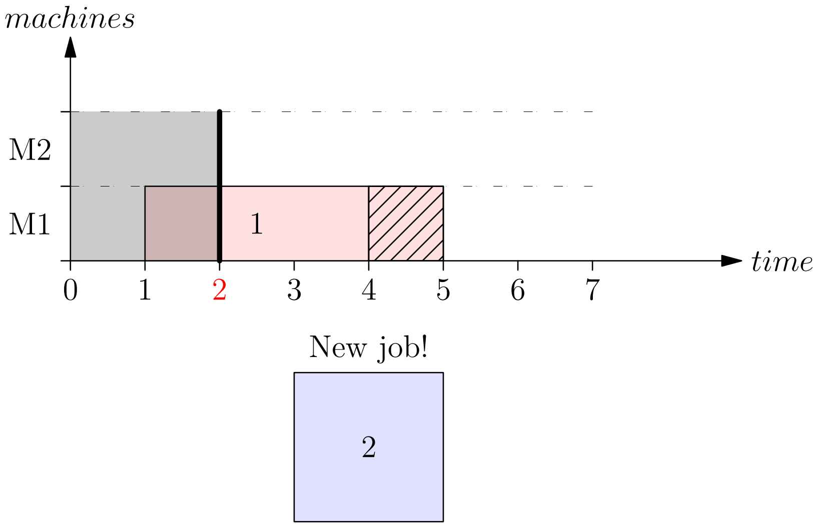
New job!



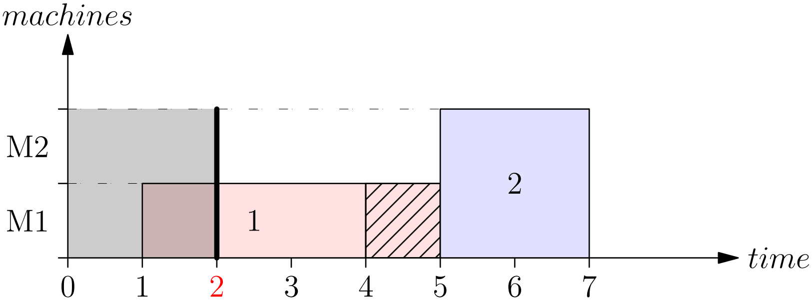
Easy Backfilling Example



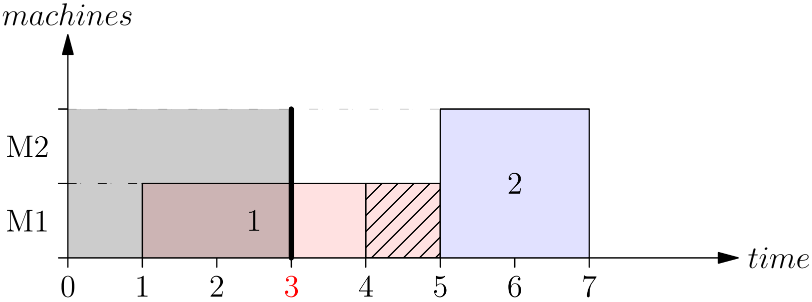
Easy Backfilling Example



Easy Backfilling Example

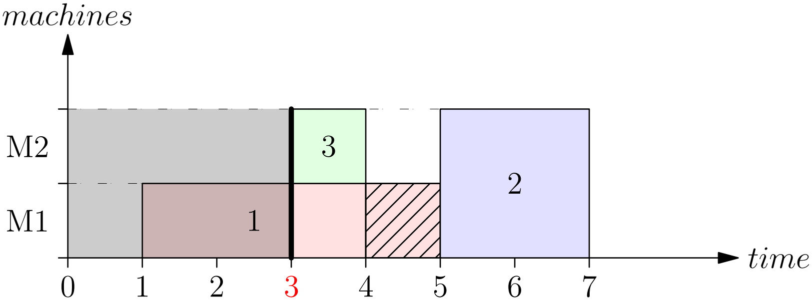


Easy Backfilling Example

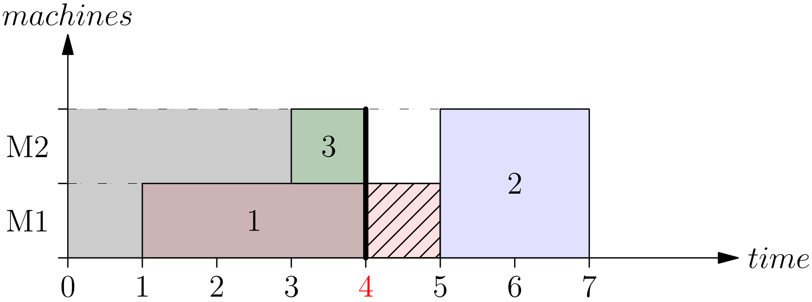


New job!

Easy Backfilling Example

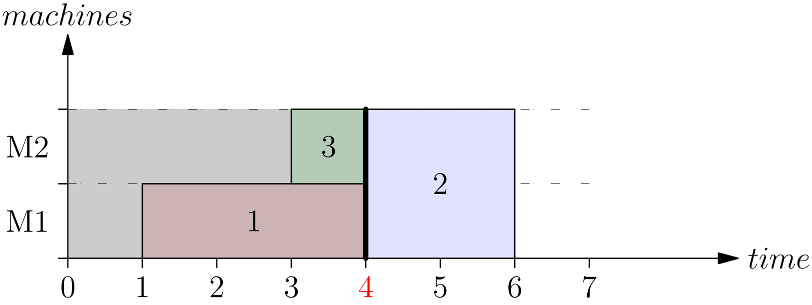


Easy Backfilling Example

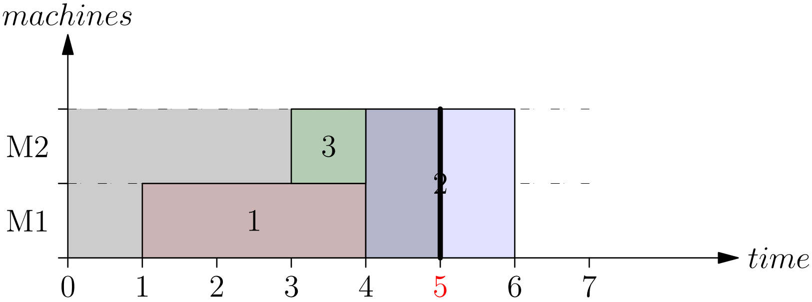


Jobs 1 and 3 finished

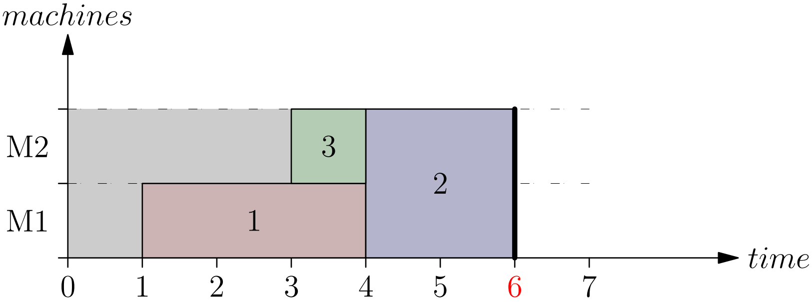
Easy Backfilling Example



Easy Backfilling Example



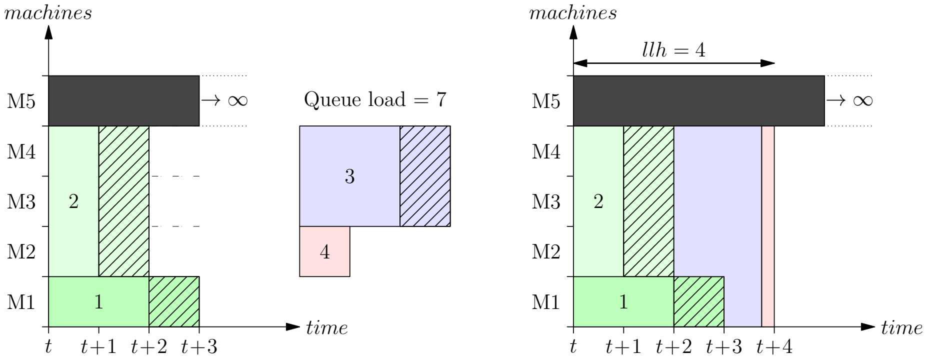
Easy Backfilling Example



How to Estimate Unresponsiveness? Liquid Load Horizon

Required time to dump current load in the provisional schedule.

$$\text{Load} = \sum_j q_j \times w_j$$

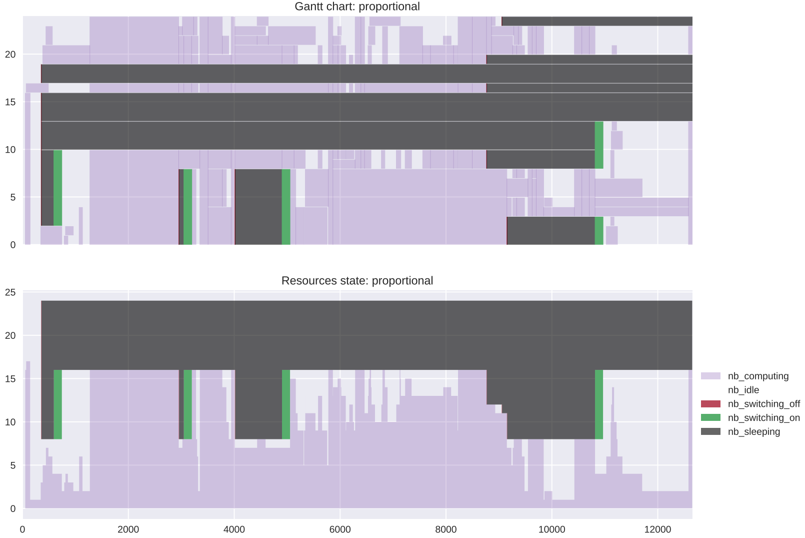


Opportunistic Shutdown

$T = 300$. $t_{idle} = 0$.

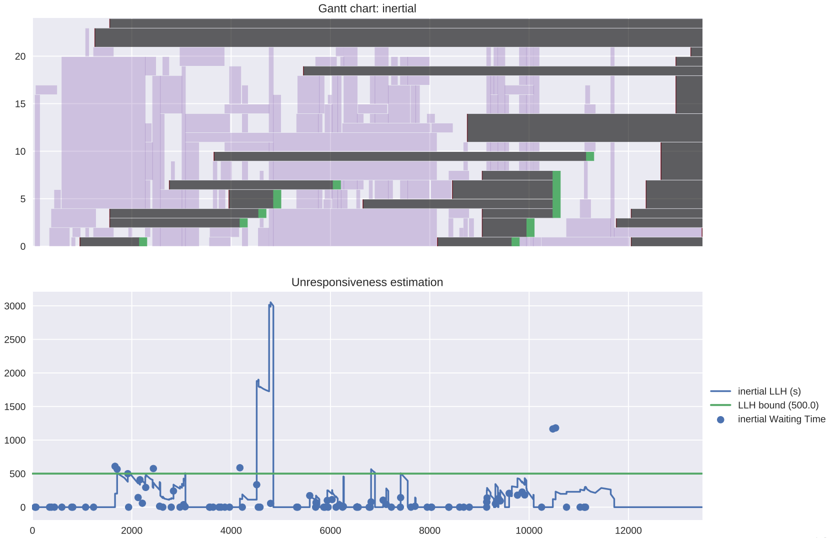
Static Adjustement

$T = 300$. 8 usable machines instead of 24.



Dynamic (Inertial) Adjustement

$T = 300.\ \bar{\upsilon}_{ub} = 500$ s. $f(x) = 2x$.



Inertial + Opportunistic

$T = 300.\ \bar{\upsilon}_{ub} = 500 \text{ s}.\ f(x) = 2x.\ t_{idle} = 0.$

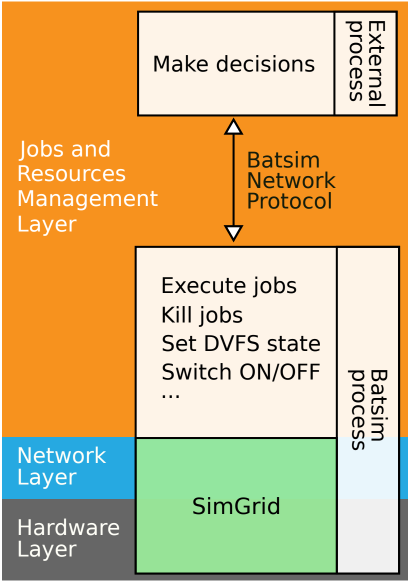
Experimental Setup

Simulation:

- Batsim (SimGrid)
- Batsched (C++)

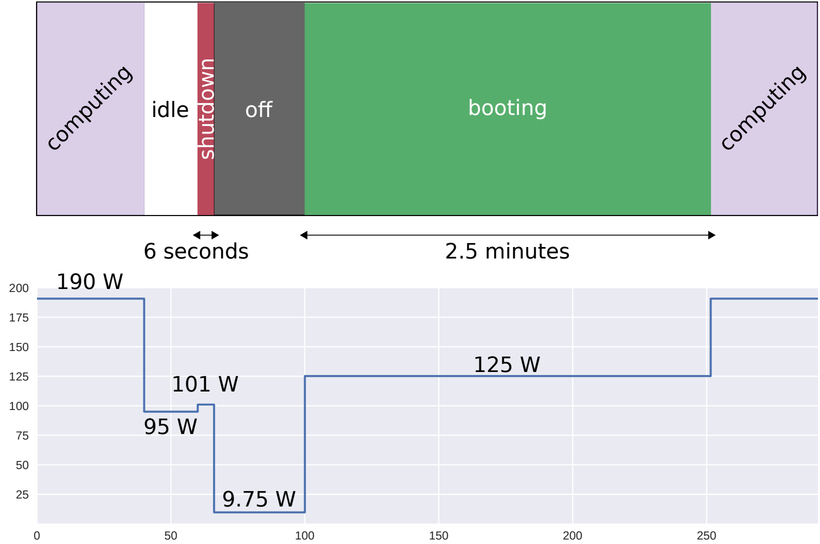
Workloads:

- KTH SP2, SDSC SP2
- Kept valid jobs ($w_j > r_j$)
- 11, 24 months → assess robustness
- Periodic utilization → room to save energy



Experimental Setup (platform)

Homogeneous. $|M| \in \{100, 128\}$. G5K Taurus [Dutot et al., 2016a].



Experimental Setup (exploration space)

Shared by all algorithms	
Workloads	KTH_SP2, SDSC_SP2
Platform	homogeneous240
Shared by Proportional and Inertial	
T (s)	60, 120, 300, 600
t_{idle} (s)	0, 30, 60, 600, 6000, $+\infty$
Make run decisions on period	true, false
Proportional-specific	
ρ	1.00, 0.95, 0.90, 0.85
Inertial-specific	
$f(n)$	$n+1$, $n \times 2$
$\bar{\upsilon}_{ub}$ (s)	$1 \cdot 10^4$, $1 \cdot 10^5$, $2 \cdot 10^5$
Allow future switches	true, false

All these parameters combinations have been tested

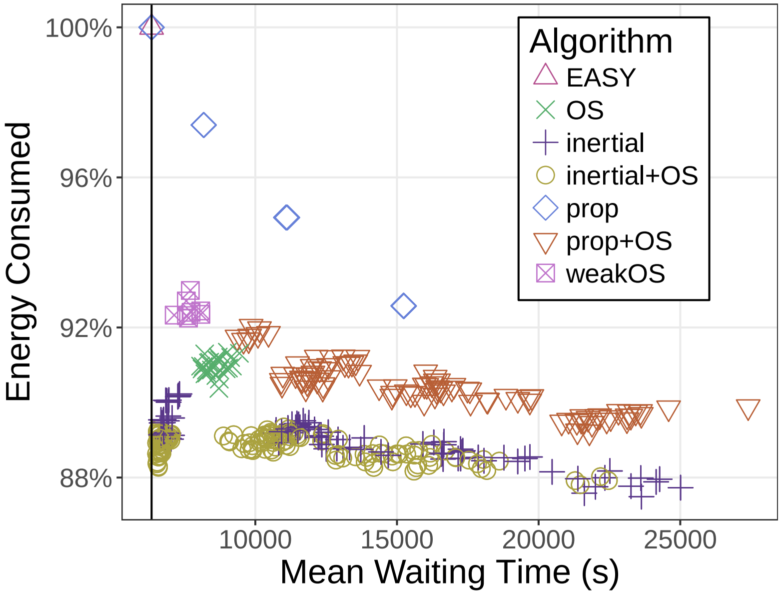
Algorithm Nomenclature

Opportunistic shutdown aggressiveness:

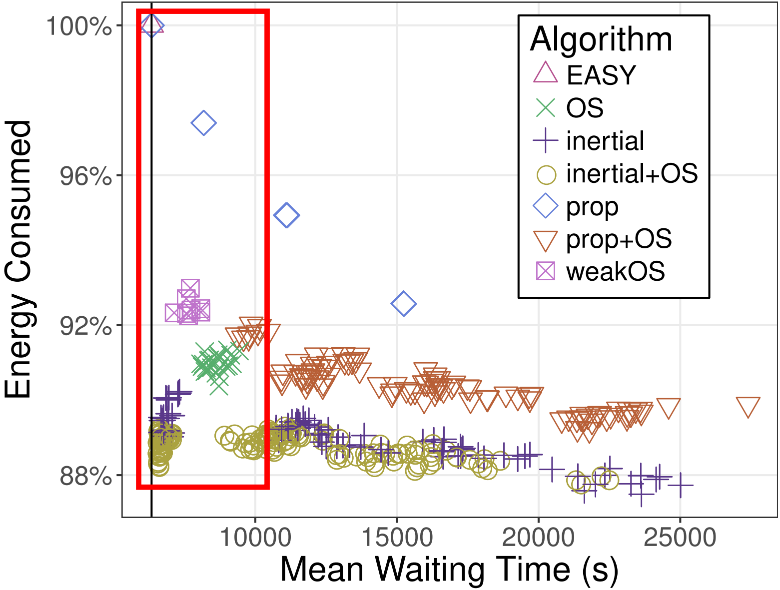
- strong: $t_{idle} \in \{ 0, 30, 60, 600 \}$
- weak: $t_{idle} \in \{ 6000, +\infty \}$

Name	Opp.?	Proportional?	Inertial?
EASY			
weakOS	weak		
prop	weak	✓	
inertial	weak		✓
OS	strong		
prop+OS	strong	✓	
inertial+OS	strong		✓

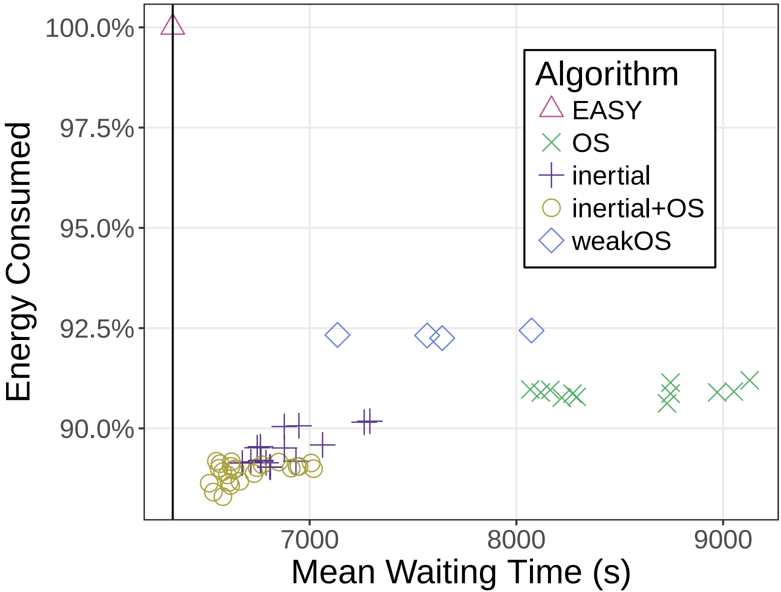
Energy / Mean Waiting Time (KTH_SP2)



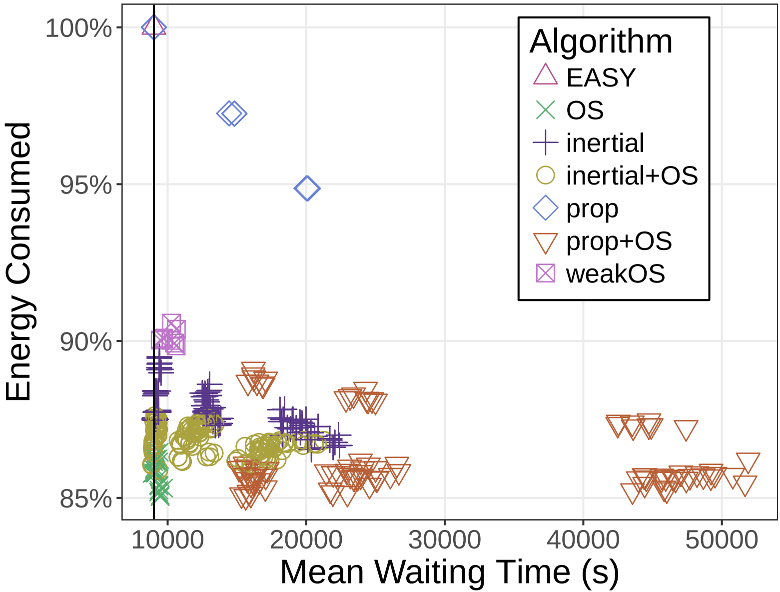
Energy / Mean Waiting Time (KTH_SP2)



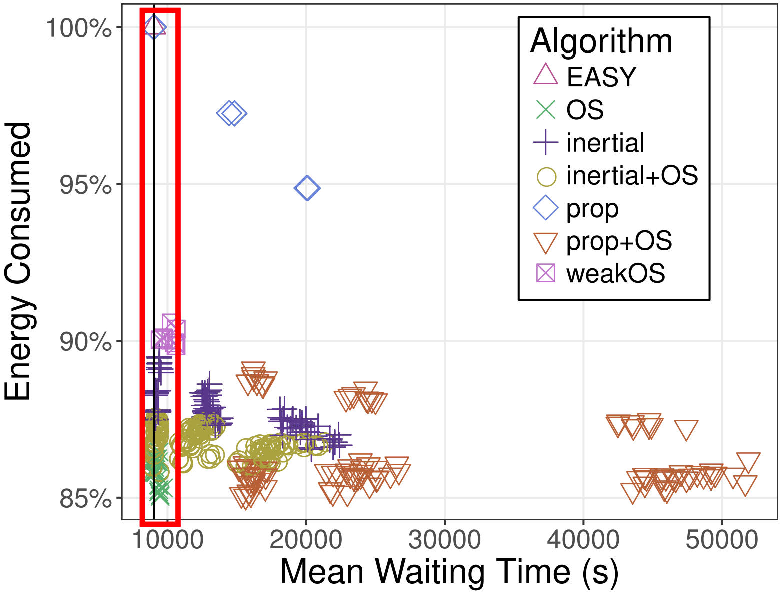
Energy / Mean Waiting Time (KTH_SP2)



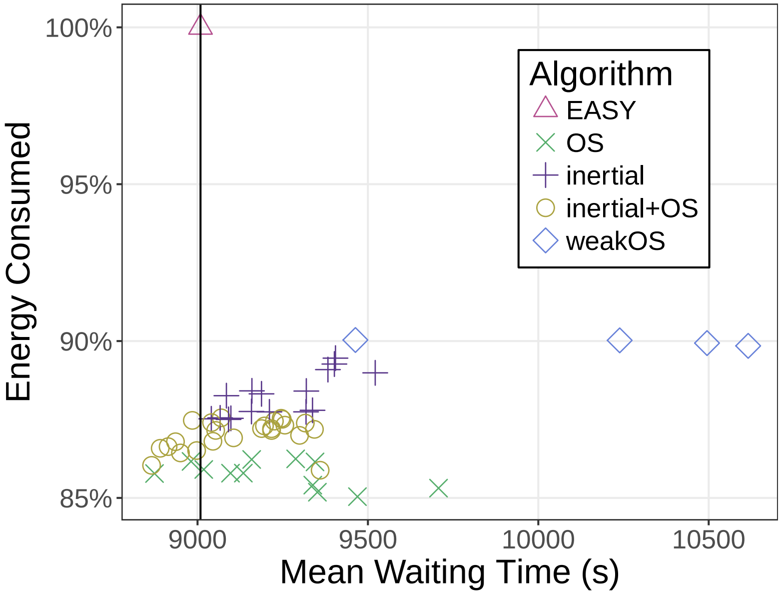
Energy / Mean Waiting Time (SDSC_SP2)



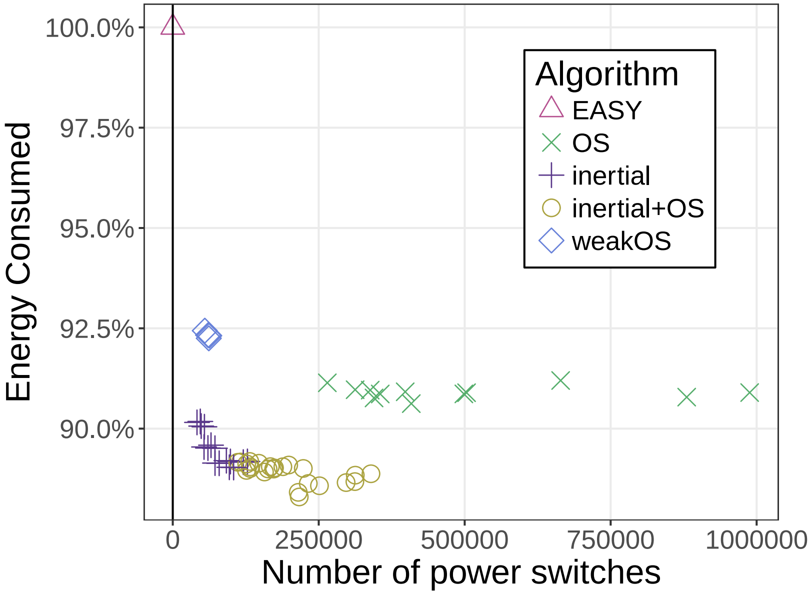
Energy / Mean Waiting Time (SDSC_SP2)



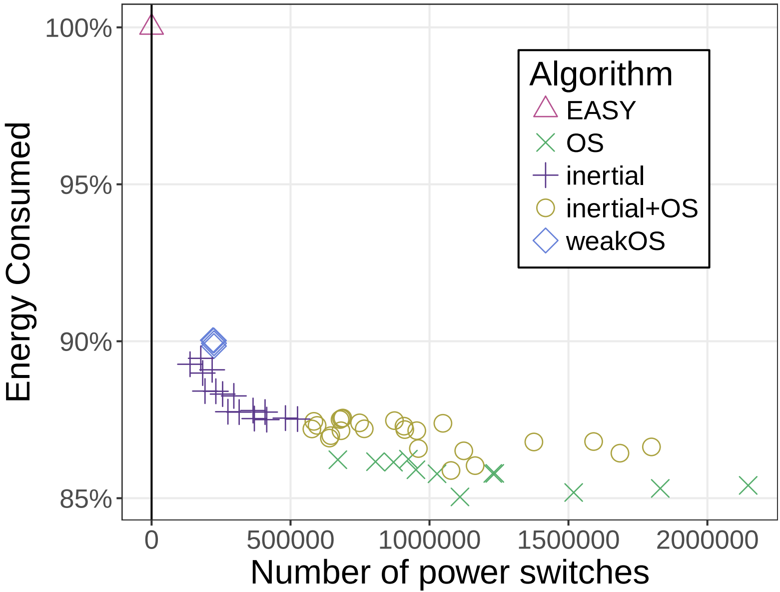
Energy / Mean Waiting Time (SDSC_SP2)



Number of Switches (KTH_SP2)



Number of Switches (SDSC_SP2)



Conclusion

Inertial shutdown:

- Energy/Performance tradeoffs
- Same order of energy savings as OS
- Low mean performance loss
- No max performance loss (not the case of OS)
- Low #switch
- Stable, predictable

Future work:

- Communication
- EASY constraints?

Thanks!

Batsim: https://github.com/oar-team/batsim
Experiment: https://gitlab.inria.fr/batsim/article-cluster17

Contact

millian.poquet@inria.fr

Inertial Shutdown

Parameters:

- $f : \mathbb{N} \to \mathbb{N}$, the inertia function
- $\bar{\upsilon}_{ub}$, the unresponsiveness mean threshold

Idea:

- Based on Easy Backfilling
- Estimates the system unresponsiveness at each event
- Do switches periodically, computing MU: the *mean* unresponsiveness since last periodic call
  - +MU → switch some machines ON
  - -MU → switch some machines OFF

Inertial Shutdown: Idertia state

$state \in \{sedating, awakening\}$ is stored
Initially, $state = awakening$

At each periodic call i:

- $(\tilde{\upsilon}_i \geq \overline{\upsilon}_{ub}) \implies$ state set to awakening. Decision made immediately.
- Otherwise,
  - $(state = awakening) \wedge (\tilde{\upsilon}_i \leq \tilde{\upsilon}_{i-1}) \implies$ state set to $sedating$. No decision made now.
  - $(state = sedating) \wedge (\tilde{\upsilon}_i > \tilde{\upsilon}_{i-1}) \implies$ state set to $awakening$. No decision made now.
  - Otherwise, decision made immediately.

Inertial Shutdown: Decision

Decision: Switch *nb* machines (ON/OFF depending on *state*)

S_a, switchable machines at i
S_e, switched machines since $i-1$ (for inertia reasons)

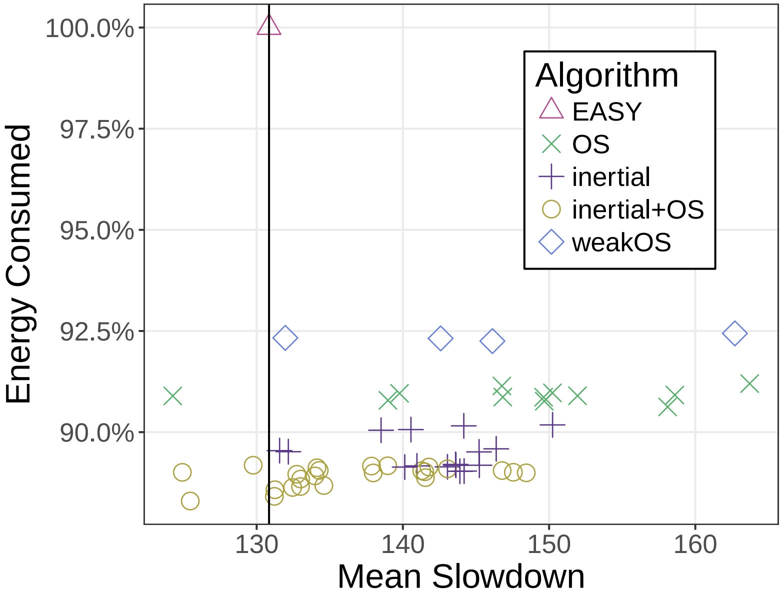
Switch at least 1 machine, without doing the impossible:

$$nb = min\left(max\left(f(|S_e|), 1 \right), |S_a| \right)$$

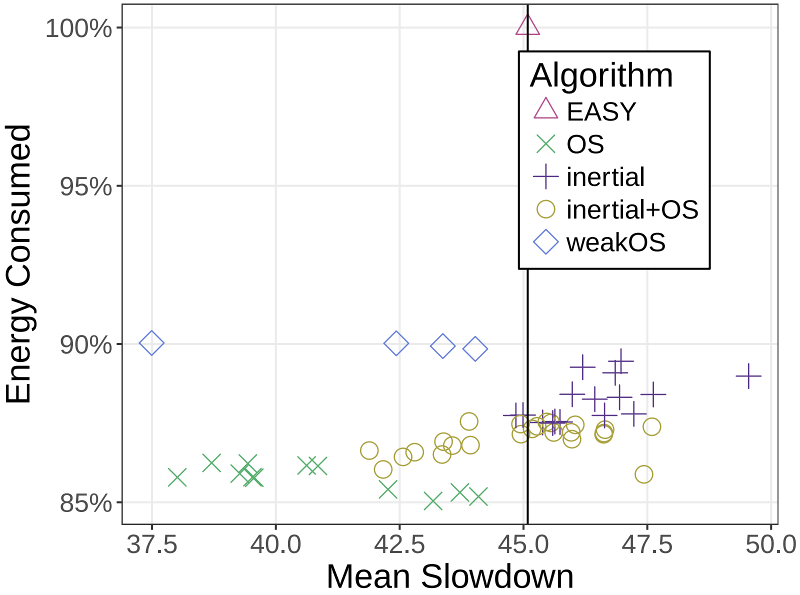
Energy information

Variable	Simulator	Scheduler
$t_{on \rightarrow off}$	151.52	$152 + 5$
$t_{off \rightarrow on}$	6.1	$6.1 + 5$
p_{off}	9.75	9.75
p_{idle}	95	95
p_{comp}	190.738	190.738
$p_{on \rightarrow off}$	100.997	101.640
$p_{off \rightarrow on}$	125.174	125.197

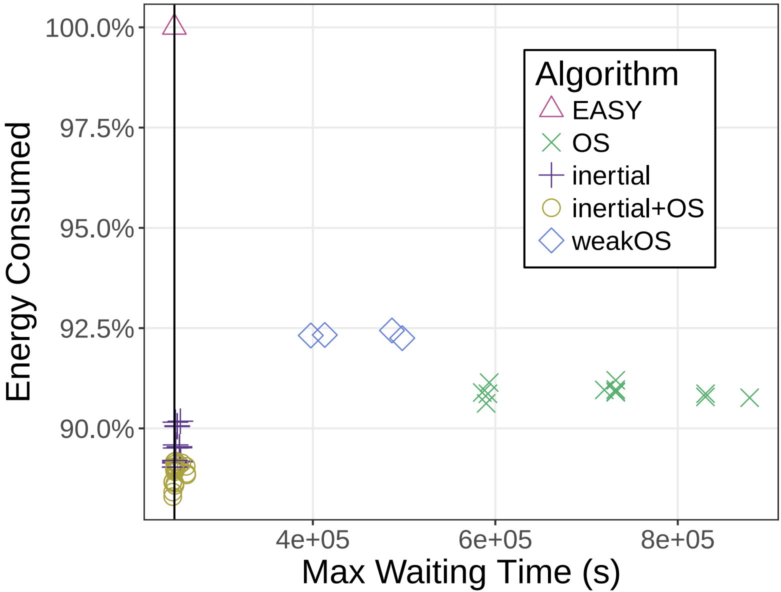
Energy / Mean Slowdown (KTH_SP2)



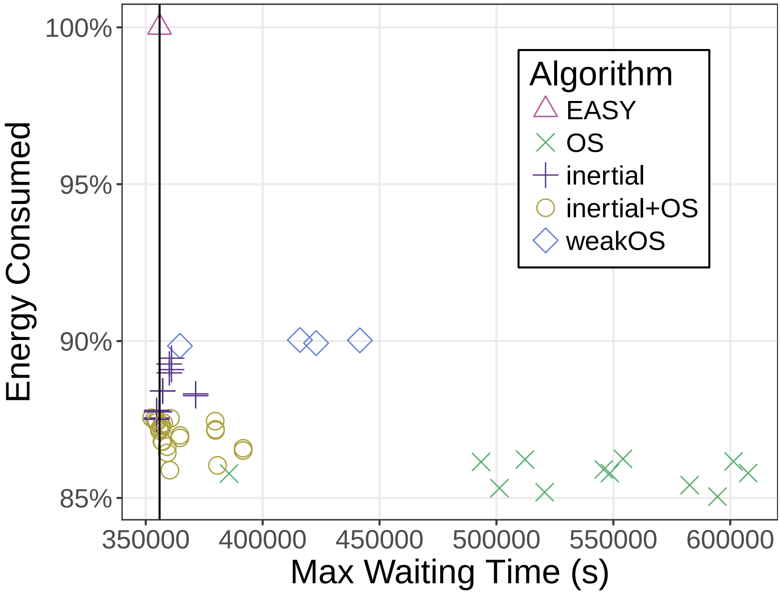
Energy / Mean Slowdown (SDSC_SP2)



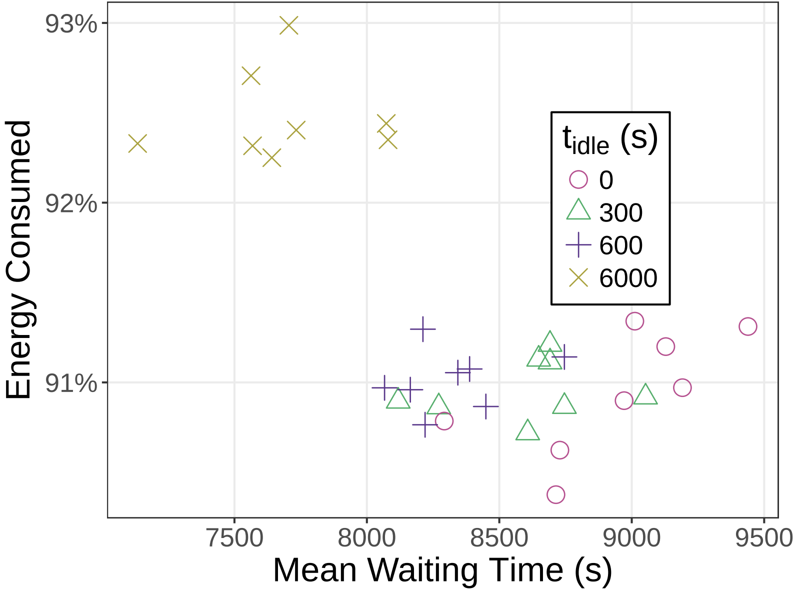
Energy / Max Waiting Time (KTH_SP2)



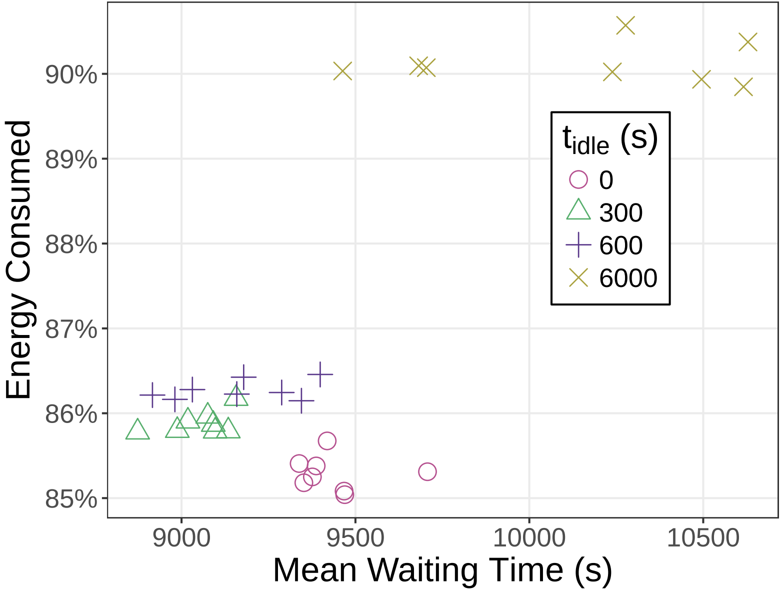
Energy / Max Waiting Time (SDSC_SP2)



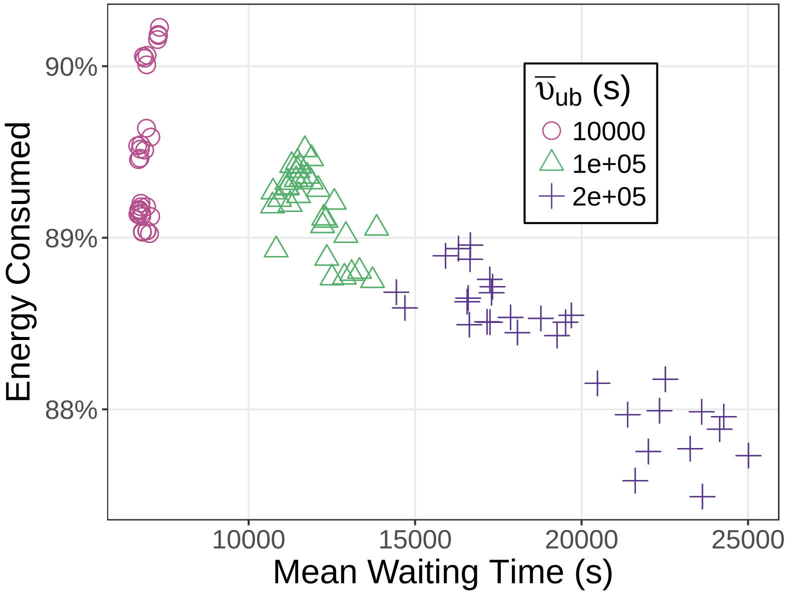
Opportunistic: Impact of t_{idle} (KTH_SP2)



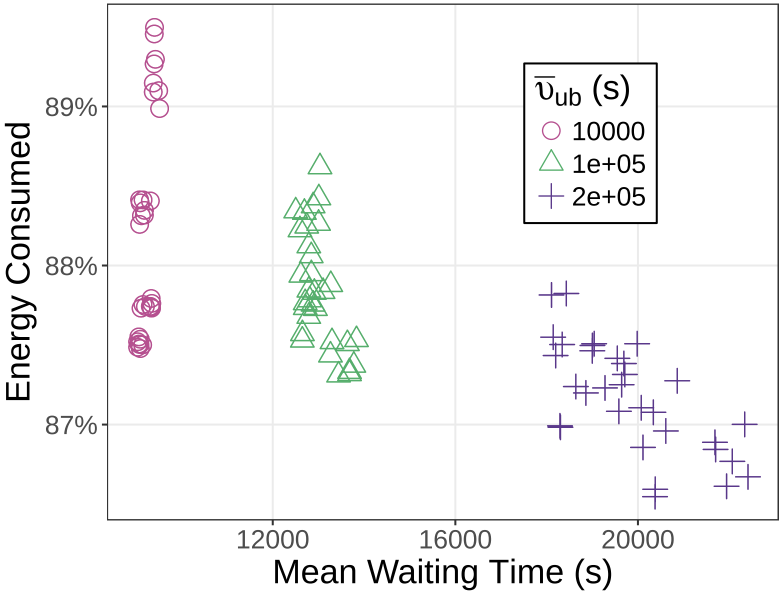
Opportunistic: Impact of t_{idle} (SDSC_SP2)



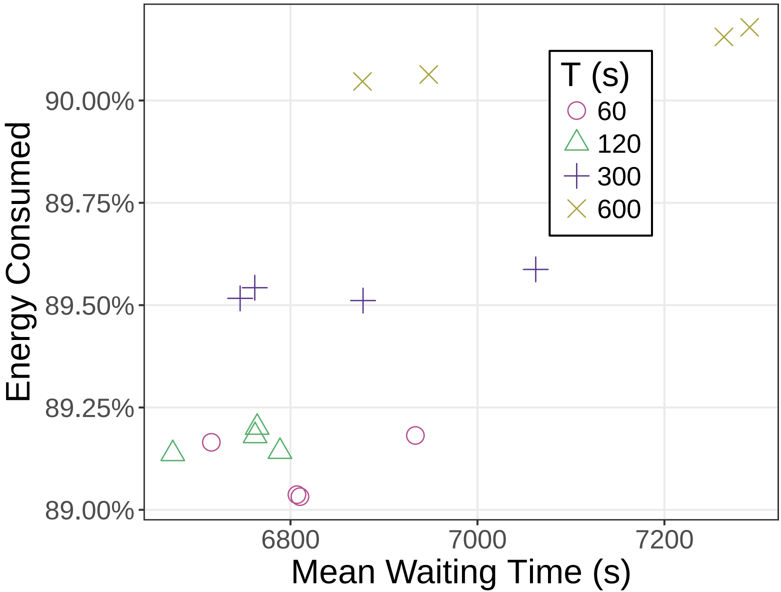
Inertial: Impact of $\overline{\upsilon}_{ub}$ (KTH_SP2)



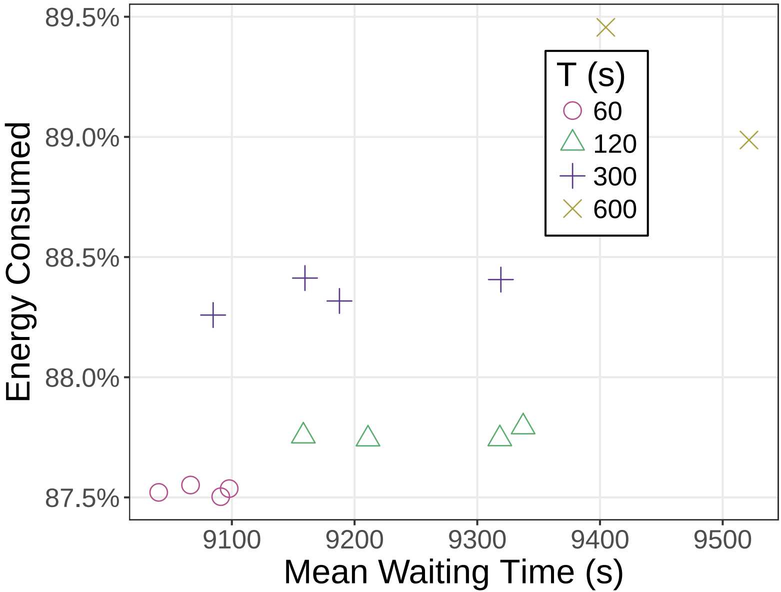
Inertial: Impact of $\overline{\upsilon}_{ub}$ (SDSC_SP2)



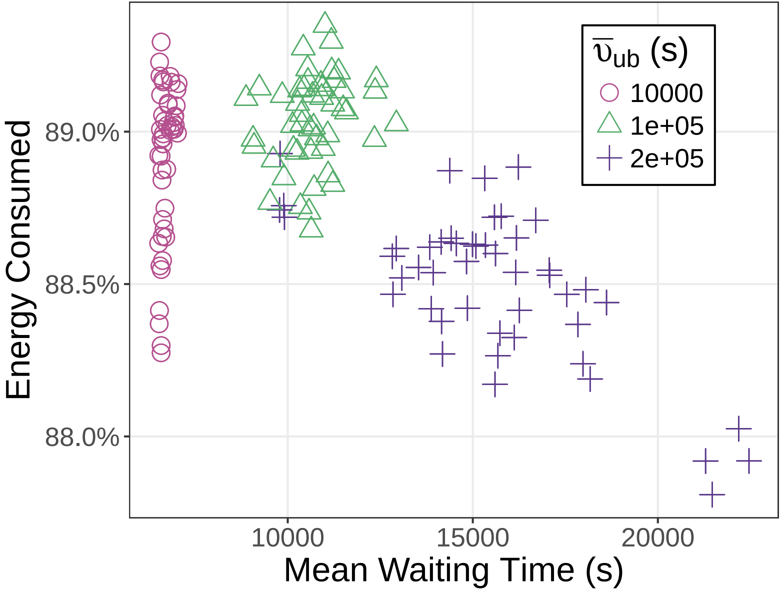
Inertial: Impact of T (KTH_SP2)



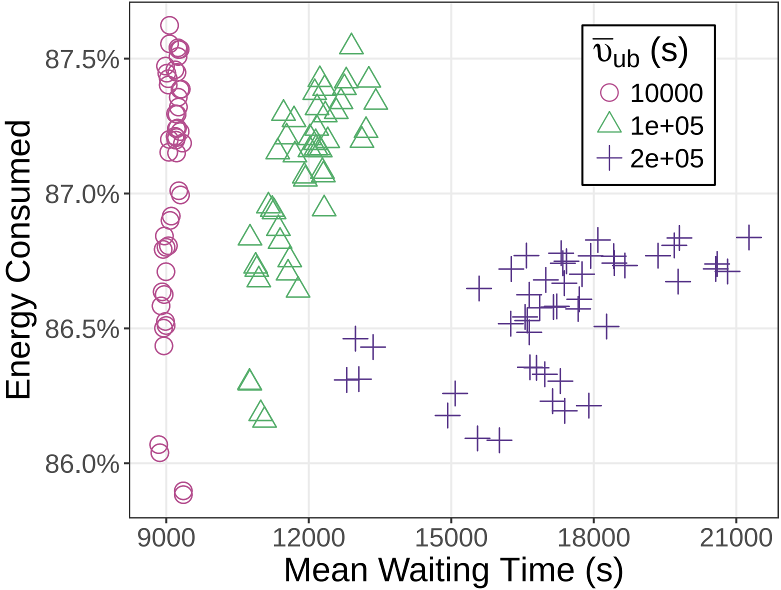
Inertial: Impact of T (SDSC_SP2)



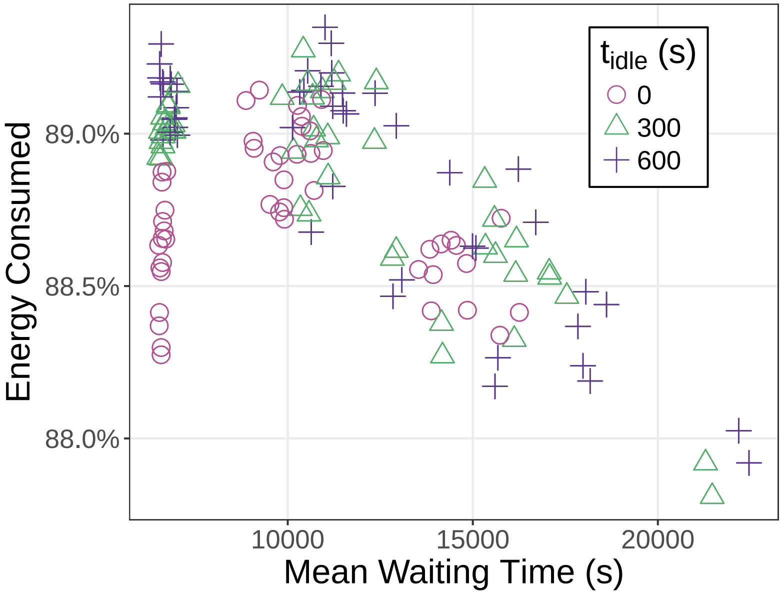
Inertial+Opportunistic: Impact of $\overline{\upsilon}_{ub}$ (KTH_SP2)



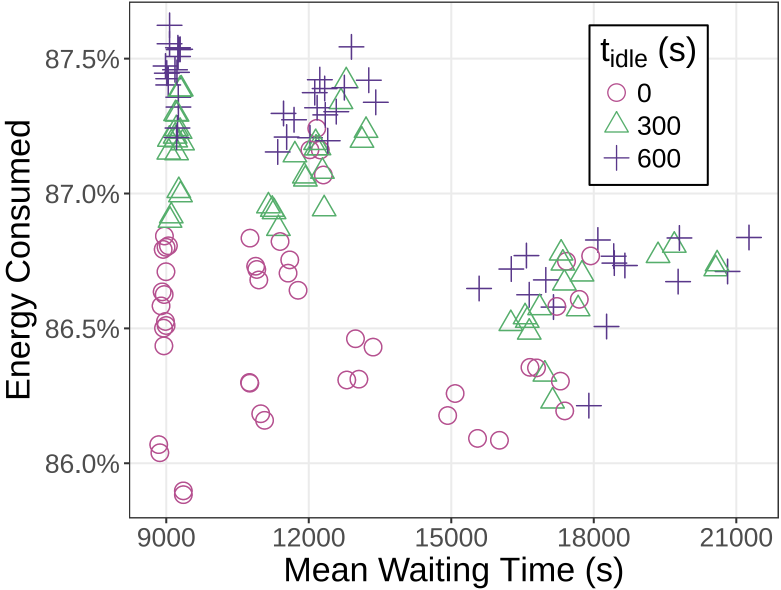
Inertial+Opportunistic: Impact of $\overline{\upsilon}_{ub}$ (SDSC_SP2)



Inertial+Opportunistic: Impact of t_{idle} (KTH_SP2)



Inertial+Opportunistic: Impact of t_{idle} (SDSC_SP2)



References I

Albers, S. (2010).
Energy-efficient algorithms.
Communications of the ACM, 53(5):86–96.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014).
Versatile, scalable, and accurate simulation of distributed applications and platforms.
Journal of Parallel and Distributed Computing, 74(10):2899–2917.

References II

Cho, S. and Melhem, R. G. (2010).
On the interplay of parallelization, program performance, and energy consumption.
IEEE Transactions on Parallel and Distributed Systems, 21(3):342–353.

Dongarra, J., Beckman, P., Moore, T., Aerts, P., Aloisio, G., Andre, J.-C., Barkai, D., Berthou, J.-Y., Boku, T., Braunschweig, B., et al. (2011).
The international exascale software project roadmap.
International Journal of High Performance Computing Applications, 25(1):3–60.

References III

Dutot, P.-F., Georgiou, Y., Glesser, D., Lefevre, L., Poquet, M., and Rais, I. (2016a).
Towards energy budget control in hpc.
In *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*. IEEE.

Dutot, P.-F., Mercier, M., Poquet, M., and Richard, O. (2016b).
Batsim: a realistic language-independent resources and jobs management systems simulator.
In *20th Workshop on Job Scheduling Strategies for Parallel Processing*.

Dutot, P.-F., Poquet, M., and Trystram, D. (2017).
Gitlab repository of the present article.

References IV

Dutot, P.-F., Rzadca, K., Saule, E., Trystram, D., et al. (2009).
Multi-objective scheduling.
Introduction to scheduling, pages 219–251.

Etinski, M., Corbalán, J., Labarta, J., and Valero, M. (2012).
Understanding the future of energy-performance trade-off via dvfs in hpc environments.
Journal of Parallel and Distributed Computing, 72(4):579–590.

Feitelson, D. (2017).
Parallel workload archive.

References V

Feitelson, D. G. (2001).
Metrics for parallel job scheduling and their convergence.
In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 188–205. Springer.

Feitelson, D. G. (2015).
Workload modeling for computer systems performance evaluation.
Cambridge University Press.

Feitelson, D. G. (2016).
Resampling with feedback—a new paradigm of using workload data for performance evaluation.
In *European Conference on Parallel Processing*, pages 3–21. Springer.

References VI

Feitelson, D. G., Tsafrir, D., and Krakov, D. (2014).
Experience with using the parallel workloads archive.
Journal of Parallel and Distributed Computing, 74(10):2967–2982.

Georgiou, Y., Glesser, D., and Trystram, D. (2015).
Adaptive Resource and Job Management for Limited Power Consumption.
In *International Parallel and Distributed Processing Symposium Workshop (IPDPS) Workshop*.

References VII

Herlich, M. and Karl, H. (2012).
Average and competitive analysis of latency and power consumption of a queuing system with a sleep mode.
In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, e-Energy '12, pages 14:1–14:10, New York, NY, USA. ACM.

Mu'alem, A. W. and Feitelson, D. G. (2001).
Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling.
TPDS.

oar team (2017a).
Batsched gitlab repository.

References VIII

oar team (2017b).
Batsim gitlab repository.

Patki, T., Lowenthal, D. K., Rountree, B. L., Schulz, M., and de Supinski, B. R. (2016).
Economic viability of hardware overprovisioning in power-constrained high performance computing.
In *Proceedings of the 4th International Workshop on Energy Efficient Supercomputing*, pages 8–15. IEEE Press.

Ruiz, C., Harrache, S., Mercier, M., and Richard, O. (2015).
Reconstructable Software Appliances with Kameleon.
SIGOPS Oper. Syst. Rev., 49(1):80–89.

References IX

Sarood, O., Langer, A., Gupta, A., and Kale, L. (2014).
Maximizing throughput of overprovisioned hpc data centers under a strict power budget.
In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 807–818. IEEE Press.

Snowdon, D. C., Ruocco, S., and Heiser, G. (2005).
Power management and dynamic voltage scaling: Myths and facts.
In *Proceedings of the 2005 workshop on power aware real-time computing*, volume 12.