Multi-objective resources optimization
Performance- and Energy-aware HPC and Clouds

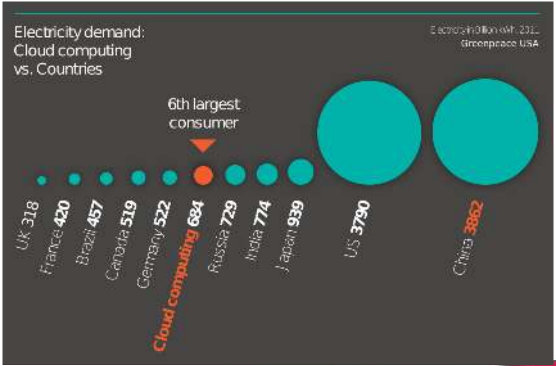
Georges Da Costa

Yerevan, Armenian National Academy of Sciences

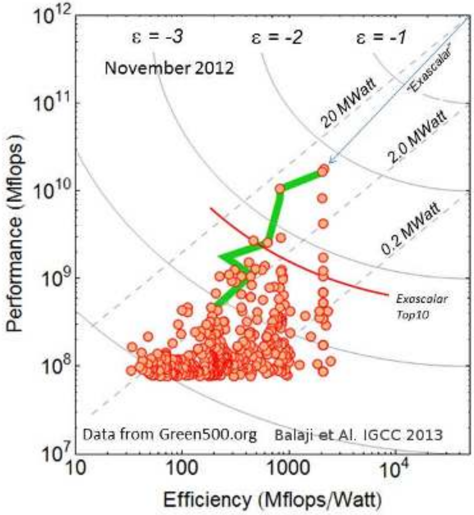
Erasmus+

IT impact on electricity

- Recent datacenters: 40000 servers, 500000 services (virtual machines). Google, Facebook > 1 million servers
- One major power consumer
    - 2000 : 70 TWh
    - 2007 : 330 TWh, 2% of CO_2 world production
    - 2011 : 6^{th} electricity consumer in the world
    - 2020 : 1000 TWh
- Rising
    - 2014 to 2016: 90% of datacenters will need hardware upgrades



How to supply electricity?



- China Telecom Inner Mongolia Information Park
  - 150 MW
- Tianhe-2
  - 17.8MW

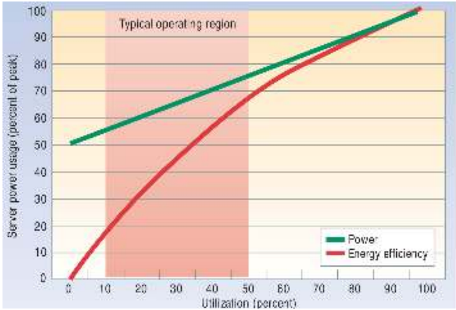
Sustainable datacenters

- Action can be done at several different levels
  - Hardware level: changing servers or cooling system
    - If entropy is constant, theoretical energy consumtion is 0 !
  - Application level: rewrite applications while changing paradigm*or library
  - Middleware level: manages servers and services/applications
- Middleware: minimal cost, maximal impact
  - OpenStack: 30% of market share in 2014
  - OpenSource solutions: 43% (+72% in 2 years)

* Georges et al. *Exascale machines require new programming paradigms and runtimes*, SFI journal, 2015

Low utilization = high electrical waste

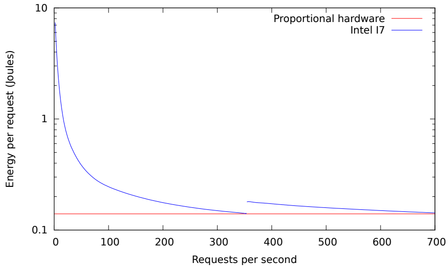
- In large organizations, computers are usually working between 10 to 50% load
- Idle power is half of max power



Barroso 2007

Low utilization = high electrical waste

- In large organizations, computers are usually working between 10 to 50% load
- Idle power is half of max power
- Problem: On low load, Watt/Request is bad

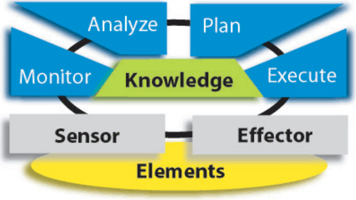


Current methods

- Current methods:
  - On high load, consolidation.
  - On high number of requests, overhead is spread on lots of nodes
  - But wasted Watts continue to add-up
- What do we want ?
  - proportional computing
  - idle load = 0W

Middlewares

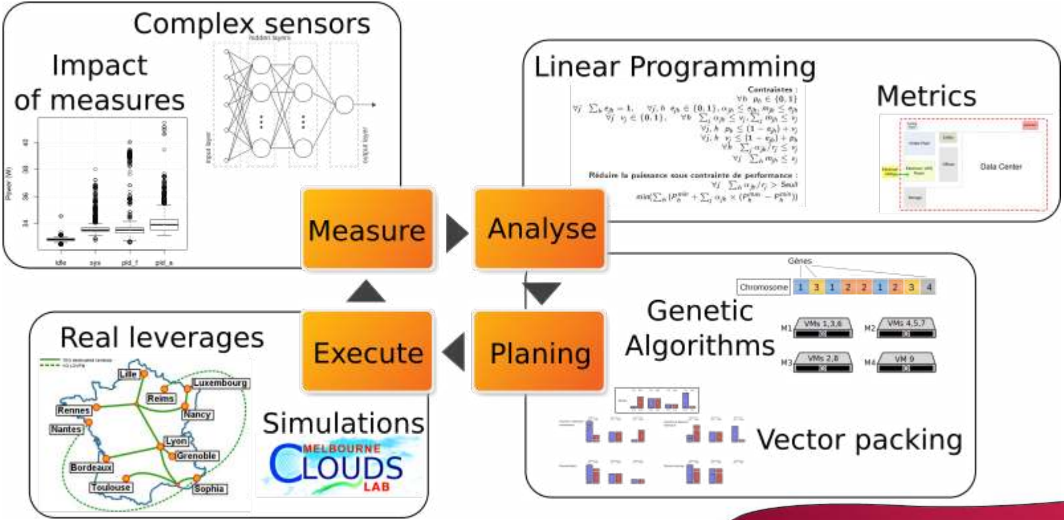
- Two goals:
  - Managing (needs, errors, faults, overheating)
  - Optimizing (Energy, performance)
- Leverages
  - Switching on/off, DVFS
  - Migration (x86/ARM)*, reduction of allocated resources, suspend
- Methods
  - Often in the real world: Humans or rules
  - In research: autonomic loop



MAPE-K loop ©IBM

* Violaine et al., *Big, Medium, Little : Reaching Energy Proportionality with Heterogeneous Computing Scheduler*, Parallel Processing Letters, journal, 2015.

Autonomic loop



Outline: How to efficiently manage a datacenter

- Efficiently?
  - It is necessary to be able to compare (**models & metrics**)
- Managing means deciding
  - **Measure** tools
  - **Evaluation tools** : Experiments, simulation
  - Exact approaches and heuristics for **decision**
- **Evolution** of datacenters
  - Datacenter federations
  - Multi-levels optimization

Plan

Model a system

To manage a system, we need to:

- Know all possible actions
- Know which is(are) the best one(s)

It can be translated into:

- Modeling impact and means (time, energy,...) of these actions
- Being able to compare two scenarios

Impact of leverages, an example with DVFS

Dynamic electric power consumed by a CMOS component:

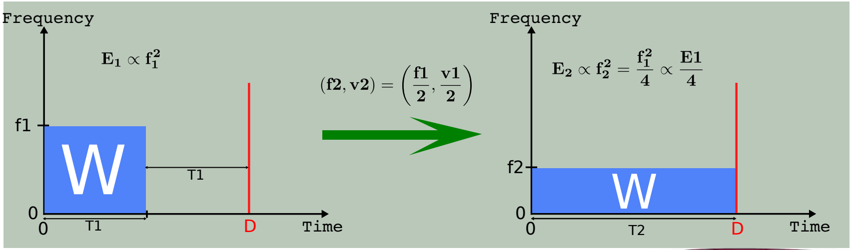
$P_{cmos} = C_{eff} \times V^2 \times f$

with, C_{eff} the effective capacitance *, V the voltage and f the frequency

* physical quantity: capacity of a component to resist to the change of voltage between its pins

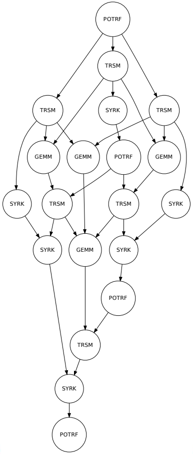
Energy consumed for each tasks:

$$E = P * T \propto T * V^3 \text{ , avec } V \propto f \text{ et } T \propto 1/F, \text{ alors } E \propto f^2$$



Dedicated hardware

- HPC applications
  - Old method: Communication and computation overlap
  - New method: Communication, complex computation, highly parallel computation,...
- Dedicated hardware
  - Dedicated hardware for each sub-task to improve overlap



Heterogeneous heterogeneous landscape

- Hardware
  - architecture : arm, GPU, FPGA
  - generation : 2014, 2015
  - In-generation : I3 I5 I7, Xeon, ...
  - And all except processor : Memory type and hierarchy, storage, network, ...
- Reconfiguration
  - DVFS, ALR (dvfs for network), ...
- Application
  - Different applications have different impact: Memory bound, cpu-intensive,...
  - Different implementation of the same API also

Some examples of dedicated hardware

- Top500
  - Tianhe-2 : 16,000 nodes, each build of two Intel Ivy Bridge Xeon and three Xeon Phi coprocessors
- European project MontBlanc
  - 2160 ARM Cortex-A15 @ 1.7 GHz dual core CPU and 1080 ARM Mali T-604 GPU
- HP MoonShot project
  - CPUs, APUs, GPUs, DSPs, and FPGAs
- Task dedicated hardware
  - Deep Learning (NVIDIA DGX-1, Intel Xeon Phi Knights Mill)

Dedicated heterogeneity is at all scale

- Dark Silicon
  - Ongoing research
  - Mostly on processor
  - Switch off unused processors units
- Heterogeneous on-die cores
  - Big.LITTLE ARM : Cortex A7 + Cortex A15, 20μs migration
  - NVIDIA Optimus : CPU-integrated GPU + Full-fleged GPU, 1/5th frame migration
- Same problems
  - Motherboard facilities (bus, network,...) always on and less dynamic
  - Baseline energy-costs are high

Example of long-term organic grow

Number of processors for each type on Grid'5000 (total 2116)

AMD Opteron 2218	100	Intel Xeon E5-2620	8	Intel Xeon E5420	68
AMD Opteron 250	158	Intel Xeon E5-2620 v3	12	Intel Xeon E5440	92
AMD Opteron 6164 HE	168	Intel Xeon E5-2630	40	Intel Xeon E5520	254
		Intel Xeon E5-2630 v3	350	Intel Xeon E5620	56
		Intel Xeon E5-2630L	32	Intel Xeon E7450	52
		Intel Xeon E5-2650	8	Intel Xeon L5335	44
		Intel Xeon E5-2660	44	Intel Xeon L5420	332
		Intel Xeon E5-2660 v2	16	Intel Xeon X3440	144
				Intel Xeon X5570	50
				Intel Xeon X5670	88

IT departments evolve

- Large institutions are build over years
- Smallest one do not necessary change hardware, only buy new one
- True also for scientists
  - Keep old habits
  - Sometime use all servers even oldest one
- True also for size
  - Small dedicated clusters for particular tasks

Even models are complex

Electrical power models for a single server:

- Classical : linear (error E∼10-15%)

$$Power = P_{min} + Load \times (P_{max} - P_{min})$$

Even models are complex

Electrical power models for a single server:

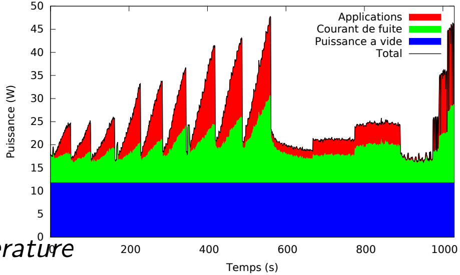
- Classical : linear (error E∼10-15%)
- Finer : Processor voltage/frequency (E∼5-9%)

$$Power = P_{min} + Load \times \alpha Voltage^2 Frequency$$

Even models are complex

Electrical power models for a single server:

- Classical : linear (error E∼10-15%)
- Finer : Processor voltage/frequency (E∼5-9%)
- Even finer: Processor temperature (E∼4-7%)



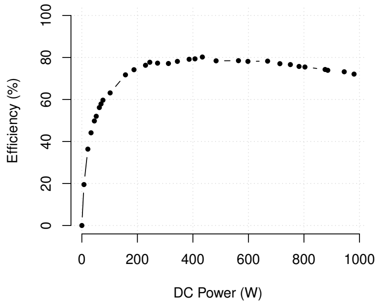
$$Power = P_{min} + Load \times \alpha Voltage^2 Frequency + \lambda Temperature$$

Even models are complex

Electrical power models for a single server:

- Classical : linear (error E∼10-15%)
- Finer : Processor voltage/frequency (E∼5-9%)
- Even finer: Processor temperature (E∼4-7%)
- Do not forget about bias: **power supply unit E∼2-3%**, cooling, ...

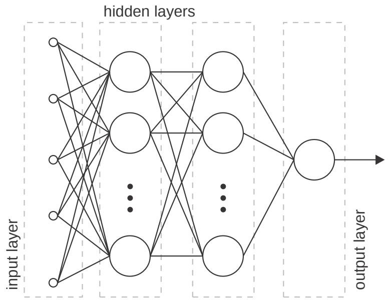
$$Power_{DC} = \omega_0 + \omega_1 Power_{AC} + \omega_2 Power_{AC}^3$$



Even models are complex

Electrical power models for a single server:

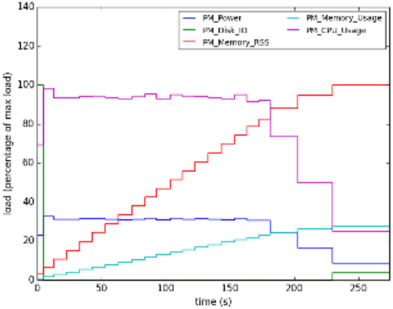
- Classical : linear (error E∼10-15%)
- Finer : Processor voltage/frequency (E∼5-9%)
- Even finer: Processor temperature (E∼4-7%)
- Do not forget about bias: **power supply unit E∼2-3%**, cooling, ...
- Learning methods (neural networks, E∼2%) *



* Leandro et al., *Towards a generic power estimator*, CSRD journal, 2015

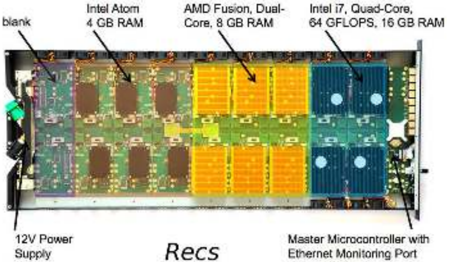
Modeling a datacenter is a complex task

- Large number of elements
  - Applications
    - Process: Traces, high-level monitoring then abstraction



Modeling a datacenter is a complex task

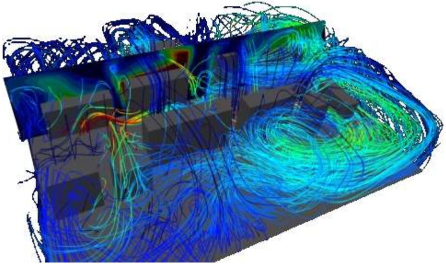
- Large number of elements
  - Applications
    - Process: Traces, high-level monitoring then abstraction
  - Servers



Recs

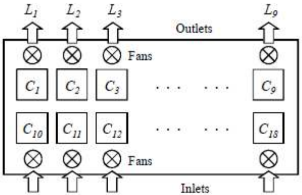
Modeling a datacenter is a complex task

- Large number of elements
  - Applications
    - Process: Traces, high-level monitoring then abstraction
  - Servers
  - Infrastructure



Modeling a datacenter is a complex task

- Large number of elements
  - Applications
    - Process: Traces, high-level monitoring then abstraction
  - Servers
  - Infrastructure
- And their interactions
  - Thermal (D-Matrix)*
  - Between applications

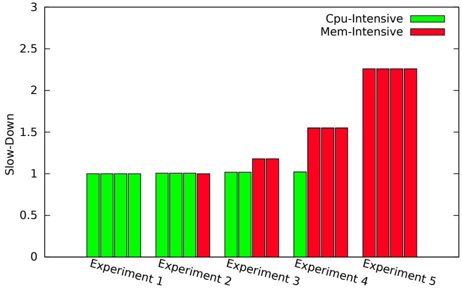


$$d_{x,k} = \begin{cases} 1, & \text{if } x = k \\ 0.84, & \text{if } x = k + 9 \\ 0, & \text{otherwise} \end{cases}$$

* Hong Yang et al.,*Energy-efficient and thermal-aware resource management for heterogeneous datacenters*, SUSCOM journal, 2014.

A “simple” interaction of applications

- Two types of mono-thread applications
  - Application 1 : Cpu-Intensive, limited by the processor
  - Application 2 : Mem-Intensive, limited by memory
- Execution on a quad-core
  - Applications 1 : Independent
  - Applications 2 : Strong cross-impact



Metrics : A complex landscape

- HPC
  - Improve performance, throughput
  - Steady and known workload
- Cloud systems
  - Improve cost efficiency
  - Varying workload, difficult to predict
- Two main questions :
  - How to program*them at large scale?
  - How to manage them at runtime?

* Georges et al. *Exascale machines require new programming paradigms and runtimes*, SFI journal, 2015

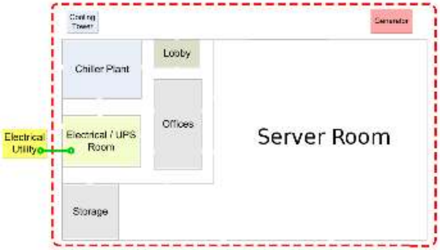
Metrics

- Direct values:
  - Processor and memory load, power, temperature,...
  - Objective: Does the system works? Comparing two datacenters, middleware, software,...
- 40000 servers, 500000 services → Need of simple metrics
  - Consumption **and** performance
- Difficult to standardize, mainly performance
  - Depends of the service, its implementation,...
- Classical metric: PUE

Georges et al. *Data Centres Sustainability Cluster Activities Task 3*. Rapport de recherche 3. European Commission, 2014

PUE : Power Usage Effectiveness

- Ratio Total electricity/IT electricity
- Mean value: 1.7 in 2014
- Standard initiated by GreenGrid
- Where does the IT part stops?
  - Power Supply Unit? Fans on the motherboard? Processor?
- Useful only in a very specific case



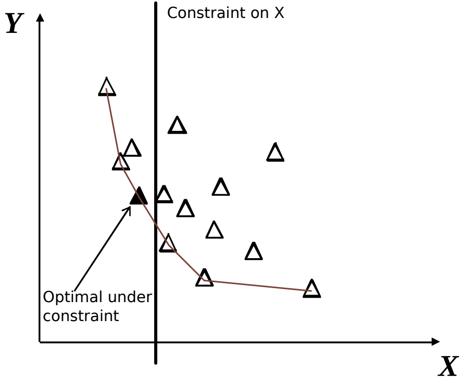
PUE : Power Usage Effectiveness

- Ratio Total electricity/IT electricity
- Mean value: 1.7 in 2014
- Standard initiated by GreenGrid
- Where does the IT part stops?
  - Power Supply Unit? Fans on the motherboard? Processor?
- Useful only in a very specific case

- Constant overhead (100), IT part 100 to 200 depending of the load
- For the same service provided by two softwares
  1. Mean load 75%
     PUE = 275/175 = 1.57
  2. Mean load 100%
     PUE = 300/200 = 1.5

Problem of multi-objective

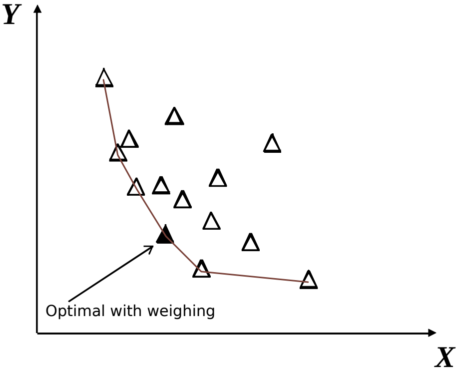
- Impossible to define in absolute, need a context, a goal
- Formalize simple metrics* : Dynamism, Energy, Performance, Resilience
- Several classical methods
  - Constraint optimization



* Tom et al., *Quality of Service Modeling for Green Scheduling in Clouds*, SUSCOM journal, 2014.

Problem of multi-objective

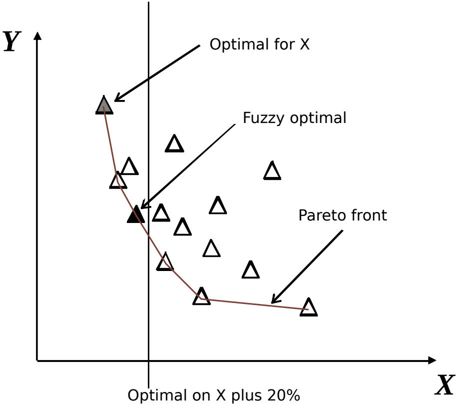
- Impossible to define in absolute, need a context, a goal
- Formalize simple metrics* : Dynamism, Energy, Performance, Resilience
- Several classical methods
  - Constraint optimization
  - Objective weighing



* Tom et al., *Quality of Service Modeling for Green Scheduling in Clouds*, SUSCOM journal, 2014.

Problem of multi-objective

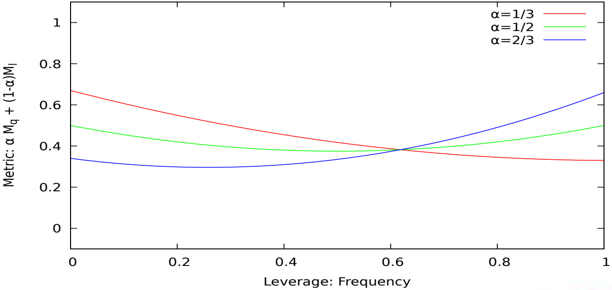
- Impossible to define in absolute, need a context, a goal
- Formalize simple metrics* : Dynamism, Energy, Performance, Resilience
- Several classical methods
  - Constraint optimization
  - Objective weighing
  - Fuzzy weighing† (Constraining by relaxation of optimal)



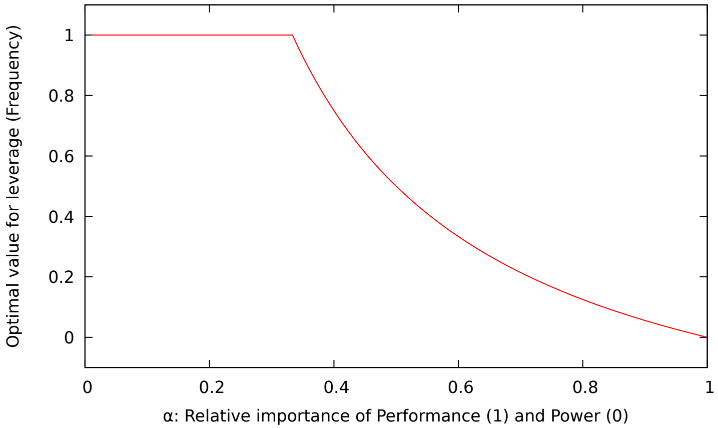
† Hong Yang et al. *Energy-efficient and thermal-aware resource management for heterogeneous datacenters*, SUSCOM journal, 2014.

Example: Performance and Power

- Two metrics, one linear, one quadratic
  - M_q is the power, so it is quadratic in function of the frequency (x)
  - M_l is the performance, so it is linear in function of the frequency (x)
- $Obj_\alpha(x) = \alpha M_q + (1-\alpha)M_l = \alpha x^2 + (1-\alpha)(1-x)$
- α is the weighing coefficient



How to choose α ?



Optimal value of frequency in function of α value.

- $Obj_\alpha(x) = \alpha M_q + (1-\alpha) M_l = \alpha x^2 + (1-\alpha)(1-x)$
- $\alpha = 0$: Max frequency
- $\alpha = 1$: Min frequency
- In-between... Voodoo !

Test suite

- Two main categories:
  - Dedicated suites (Web services, database, HPC,...)
  - Generic suites
- Scientific, Infrastructure manager: Black-Box applications
- The system must be the same in all cases
- Maximum coverage test-suite
  - Same resources/Different power
  - Different resources/Same power

Georges et al., *Energy- and Heat-aware HPC Benchmarks*, EuroEcoDc workshop, 2013

Conclusion

Study of a system

- For which use?
- Three notions are linked to provide the answer:
  - Balance of the precision front of the models*
  - Objective function used for comparing
  - Scenarios used to kame the comparison

* Georges et al., *Modèles fluides pour l'économie d'énergie dans les grilles par migration : une première approche*, RenPar conférence, 2009

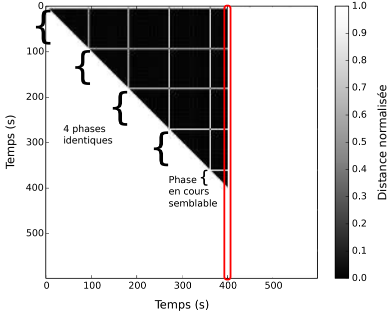
Measure Infrastructure

- Basis for taking decision
- Basis for metric evaluation
  - Classical Infrastructure (*nagios*, *ganglia*, ...)
  - Problem for scaling
    - Most values are unused of aggregated late
  - Some measures (processor, memory), but no **knowledge**
    - Need of higher level measures
    - What type of (phase of an) application
    - Electric power consumed by applications

Which (phase of an) application is running

- A phase : behavior locally regular
  - Equivalent as a constant resource consumption
  - System measures constants
- Detection then identification
  - Signature of a phase
  - Same Phase ∼ Same Impact

Landry et al., *Exploiting performance counters to predict and improve energy performance of HPC systems*, FGCS journal, 2014.



Matrix of similar system measures (WRF : Weather Research and Forecasting)

External application identification

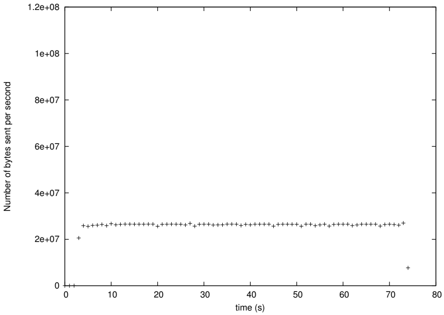
- Monitoring system values is intrusive
- Reduce the number of values monitored
- Using external values has lower impact (power, network)
- Authorize statistic tools
- Study the behavior during time

Georges et al., *Characterizing applications from power consumption : A case study for HPC benchmarks*, ICT-GLOW Symposium, 2011

External application identification

- Monitoring system values is intrusive
- Reduce the number of values monitored
- Using external values has lower impact (power, network)
- Authorize statistic tools
- Study the behavior during time

Georges et al., *Characterizing applications from power consumption : A case study for HPC benchmarks*, ICT-GLOW Symposium, 2011

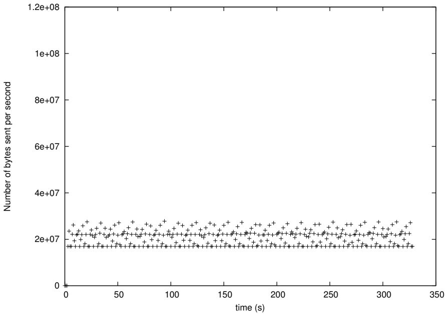


benchmark CG (NPB)

External application identification

- Monitoring system values is intrusive
- Reduce the number of values monitored
- Using external values has lower impact (power, network)
- Authorize statistic tools
- Study the behavior during time

Georges et al., *Characterizing applications from power consumption : A case study for HPC benchmarks*, ICT-GLOW Symposium, 2011

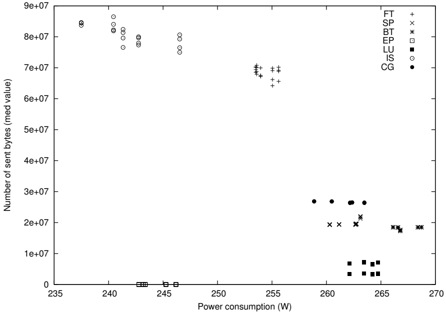


benchmark SP (NPB)

External application identification

- Monitoring system values is intrusive
- Reduce the number of values monitored
- Using external values has lower impact (power, network)
- Authorize statistic tools
- Study the behavior during time

Georges et al., *Characterizing applications from power consumption : A case study for HPC benchmarks*, ICT-GLOW Symposium, 2011



Power of servers and applications

- Watt-meters are not always available (application level: never)
- Model linking system measures with electrical power
- Analytical
  - Uses Datasheets. Very simple to put in place: PowerAPI

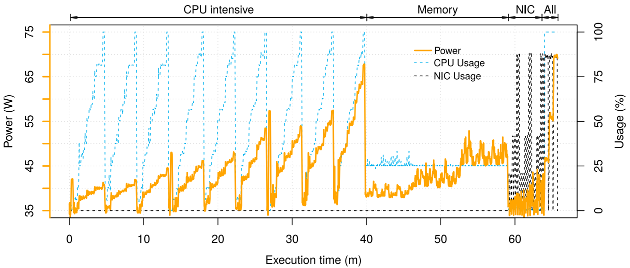
Power of servers and applications

- Watt-meters are not always available (application level: never)
- Model linking system measures with electrical power
- Learning method
  - Good coverage of the learning set and low impact of the measure

Generic synthetic load, 220 measured values (4% increase of power), 8 kept

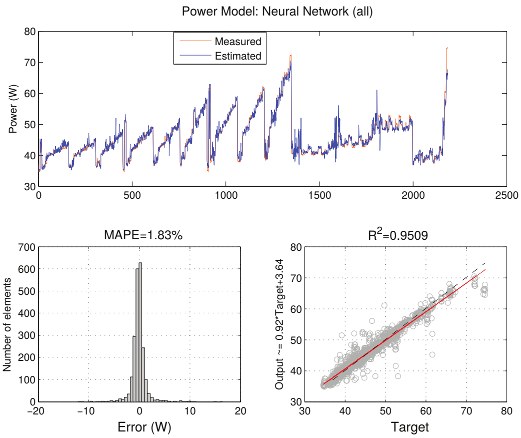
Power of servers and applications

- Watt-meters are not always available (application level: never)
- Model linking system measures with electrical power
- Learning method
  - Good coverage of the learning set and low impact of the measure

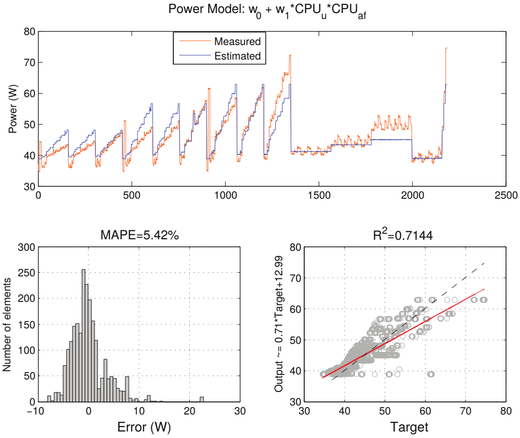


20% of values taken by HPC tests are not reached

Neural network VS capacitive model

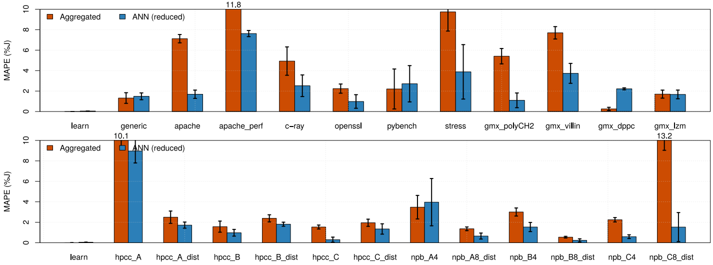


(a) ANN regression using all KPIs.



(b) Calibrated capacitive model.

Result of learning methods



Error of two models: **Aggregated** (linear regression on sum of sub-models) and **ANN** (neural network)

- Most models: error of 5%
  - Reference in the field: Rivoire et Al. *A Comparison of High-Level Full-System Power Models*

Leandro *Modeling the power consumption of computing systems and applications through Machine Learning techniques*, 2015

Conclusion

- Current approaches need a global point of view
  - Scalling
  - Latency problem
  - Lots of decisions are local (DVFS, migration,...)
- Open problems
  - Granularity of the measure: Adaptive and multi-scale
  - Spatial and Temporal independence
  - Maximal coverage test suite automatic generation

Evaluation tools: Experimentation, Simulation

- To improve, comparison is necessary
- Three main methods
  - Mathematical models
  - Simulation
  - Experiments

Linear programming

- Describe all constraints with linear equations

Example : A task is on a unique server

- Let e_{jh} the fact that task j runs on server h
- $e_{jh} = 1$ iif task j is on server h
- $\forall j, h \quad e_{jh} \in \{0, 1\}$,
  $\forall j \quad \sum_h e_{jh} = 1$

Linear programming

- Describe all constraints with linear equations
- Describe the objective as a function to minimize

Example : Minimize the total power consumed

- P_h^{stat} et P_h^{dyn} : static and dynamic power of server h (linear model)
- Let α_{jh} the processor fraction of task j on server h
- $min \sum_h (P_h^{stat} + \sum_j \alpha_{jh} P_h^{dyn})$

Linear programming

- Describe all constraints with linear equations
- Describe the objective as a function to minimize
- Formalize leverages and their impact
- Approximation of real world (quadratic phenomena)
- Exact resolution for small cases

Constraints :

$$\forall h \;\; p_h \in \{0,1\}$$
$$\forall j \;\; \sum_h e_{jh} = 1, \quad \forall j,h \;\; e_{jh} \in \{0,1\}, \alpha_{jh} \leq e_{jh}, m_{jh} \leq e_{jh}$$
$$\forall j \;\; v_j \in \{0,1\}, \quad \forall h \;\; \sum_j \alpha_{jh} \leq v_j, \sum_j m_{jh} \leq v_j$$
$$\forall j,h \;\; p_h \leq (1 - e_{jh}) + v_j$$
$$\forall j,h \;\; v_j \leq (1 - e_{jh}) + p_h$$
$$\forall h \;\; \sum_j \alpha_{jh}/r_j \leq v_j$$
$$\forall j \;\; \sum_h m_{jh} \leq v_j$$

Minimize power under performance constraints:

$$\forall j \;\; \sum_h \alpha_{jh}/r_j > Threshold$$
$$min(\sum_h (P_h^{min} + \sum_j \alpha_{jh} \times (P_h^{max} - P_h^{min}))$$

Damien et al., *Energy-Aware Service Allocation*, FGCS journal, 2011

Scaling of linear programming

- Exact method: time complexity exponential in function of integer variables
  - 6 servers, 16 tasks : 3 minutes (GLPK)

Scaling of linear programming

- Exact method: time complexity exponential in function of integer variables
  - 6 servers, 16 tasks : 3 minutes (GLPK)
- Methods of constant variables
  - Fix some variables, solve, change the fixed variables and iterate
  - Worse than the optimal but can be very fast

Scaling of linear programming

- Exact method: time complexity exponential in function of integer variables
  - 6 servers, 16 tasks : 3 minutes (GLPK)
- Methods of constant variables
  - Fix some variables, solve, change the fixed variables and iterate
  - Worse than the optimal but can be very fast
- Relax the integer constraint
  - "Better" than optimal (a task half on a server and and half on another one)

Scaling of linear programming

- Exact method: time complexity exponential in function of integer variables
  - 6 servers, 16 tasks : 3 minutes (GLPK)
- Methods of constant variables
  - Fix some variables, solve, change the fixed variables and iterate
  - Worse than the optimal but can be very fast
- Relax the integer constraint
  - "Better" than optimal (a task half on a server and and half on another one)
- Using both it is possible to have a interval

Simulation

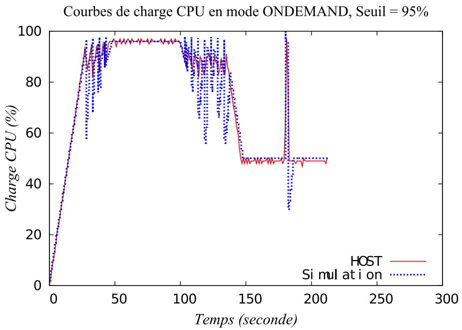
- Large number of simulators: SimGrid, DCWorms, CloudSim, ...
- Particular needs for our research
  - Cloud models (migration, Over-allocation of resources, federation†)
  - DVFS
  - Electrical power
  - Temperature
- Situation is steadily improving
  - DVFS and fine-grained management of clouds in CloudSim
  - Thermal simulation in DCWorms*
  - DVFS and energy in SimGrid

* Wojtek et al., *Energy and thermal models for simulation of workload and resource management in computing systems*, SMPT journal, 2015. †Thiam et al., *Cooperative Scheduling Anti-load balancing Algorithm for Cloud*, CCTS workshop, 2013

Adding DVFS in CloudSim

- Simulators mainly come from Grid world
  - Stability during time
  - Resources are always used at 100%
- DVFS needs to move events
- Fine-grained temporal management (1/10 s)

Tom et al., *Energy-aware simulation with DVFS*, SMPT journal, 2013



Ad-hoc simulators: Reinvent the wheel?

- Simplistic simulators
- Lets test an idea at a low cost
- Necessary to stop at the right complexity level
  - Simulator of heterogeneous architectures*
  - No network simulation
  - Example: prove the utility of heterogeneity to reach a proportional system

* Georges, *Heterogeneity: The Key to Achieve Power-Proportional Computing*, CCGrid conférence, 2013.

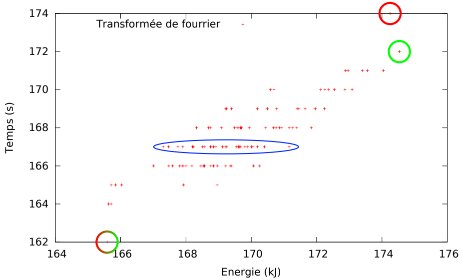
Experimentation

- A model is always an approximation
- Final validation by experiment
- Complex because of the need to have electrical measures
  - At ENS-Lyon, they where one of the first to experiment with watt-meters at large scale (GreenNet)*
- Problem of distributed measures, electrical conversions, impact of measures (performance counters)
- Reproducibility problem

* Da Costa, *The green-net framework: Energy efficiency in large scale distributed systems*, IPDPS, 2009

No stability of experiments

- *Simple* experiment of Fast Fourier Transform (NPB)
- 100 experiments on exactly the same hardware (Grid'5000)
- Large variations
  - Time: 12s, 7% (Std. Dev. 3.2s)
  - Energy: 9.3kJ, 5.5% (3kJ)
- For the same time, 167s
  - Difference of 4kJ
- Time $\neq$ Energy



Conclusion

- All tools are limited
  - Models: Approximate or optimal unreachable
  - Simulation: Approximate
  - Experimentation: Reproducibility and very sensible
- Large investment necessary
- Simulation : Quite good value for money
- Difficult to test In-Vivo

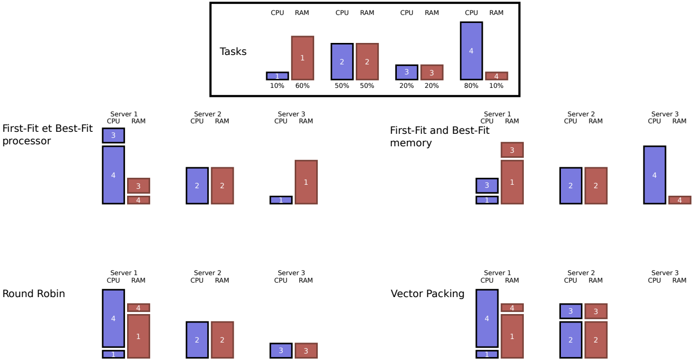
Plan

Exact Approaches and heuristics

- Two problems
  - Placement
  - Temporality
- Classical heuristics for placement
  - Greedy: Best Fit, First Fit
  - Vector Packing (*Gourmet* Greedy)
  - Genetic algorithms

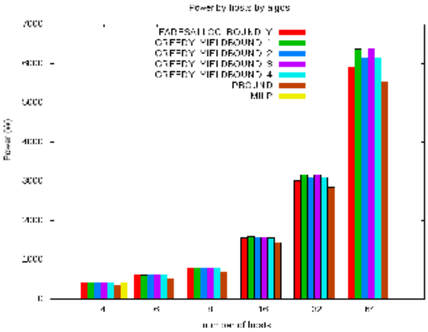
Classical greedy algorithms

- Characteristics
  - Memory
  - Processor
- Sort services
- Sort servers
- No coming back on previous decisions



Gourmet Vector packing

- 4 objectives in the sort function
  - Server is attractive from an energy point of view
  - Add the task do not overload the server
  - Server already switched on
  - The tasks brings back the balances of resources
- Time "only"in $\mathcal{O}(J \times H \ln(H))$
- But the solution of the *Gourmet* is difficult to qualify

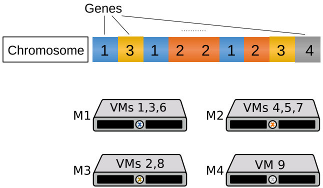


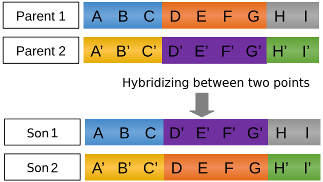
Damien et al., *Energy-Aware Service Allocation*, FGCS journal, 2012.

Genetic Algorithms

- Chromosome = Allocation
- Initial random generation
- At each generation:
  - Hybridizing and mutation
  - Sort on the objective metric
  - Keep only the best

Tom et al., *Quality of Service Modeling for Green Scheduling in Clouds*, SUSCOM journal, 2014





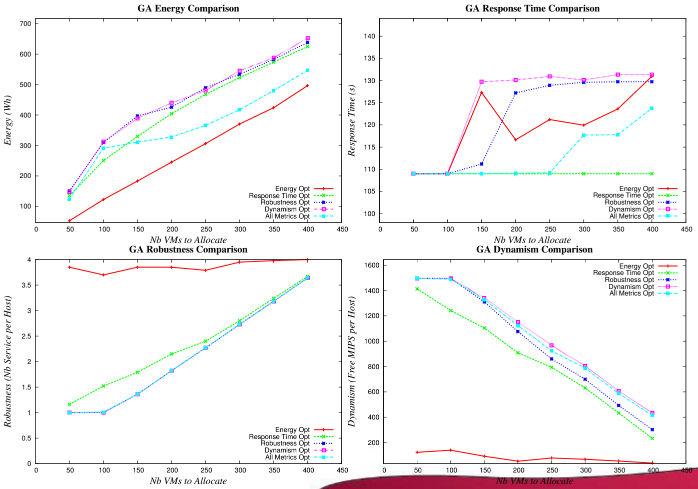
Metrics for genetic algorithms

- Contrary to with greedy, we can optimize a metric directly
- Examples of metrics
  - Energy, Performance, Resilience, Dynamism

GA Name	Coefficient applied to metrics			
	Energy	Response time	Robustness	Dynamism
GA_All	1	1	1	1
GA_Energy	1	0	0	0
GA_RespT	0	1	0	0
GA_Rob	0	0	1	0
GA_Dyn	0	0	0	1

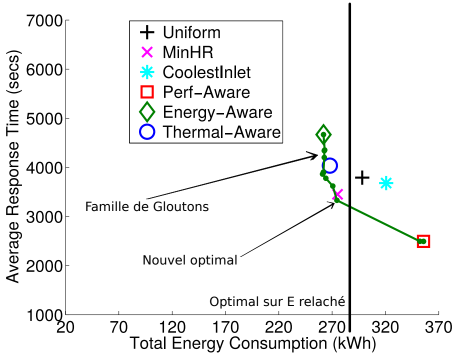
Results of the genetic algorithm

- Each algorithm is the best in its own domain (Energy)
- GA_All Very good everywhere
- 400 services on 110 servers, approximately 40s
- Taking into account a metric is already very important



Fuzzy Greedy

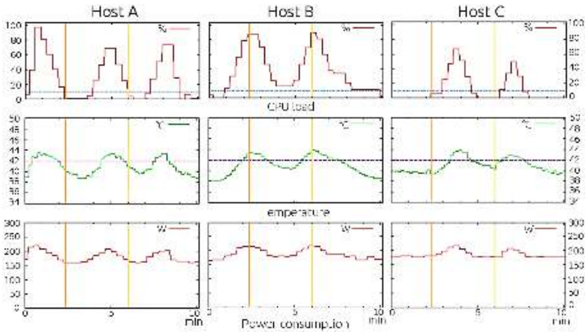
- Advantage of the G.A.: aim an objective
- Similar method for greedy algorithms
  - Families of greedy algorithms
  - Keep the best
  - Define the *best* ?
- Fuzzy multi-objective



Hong Yang et al., *Multi-Objective Scheduling for Heterogeneous Server Systems with Machine Placement*, CCGRID conférence, 2014

Why power/energy is unique

- The temporal point of view
  - Inertia due to temperature
  - Switching on/off servers
    - Over- or Under-reservation
  - Cycles are sometime good
- Non-linearities
  - Equation of power
- Feedback loops
  - Cooling system

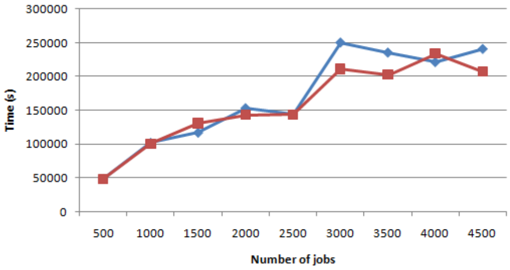


Violaine et al., *Thermal-aware cloud middleware to reduce cooling needs*, WETICE workshop, 2014

At the scale of a cloud federation

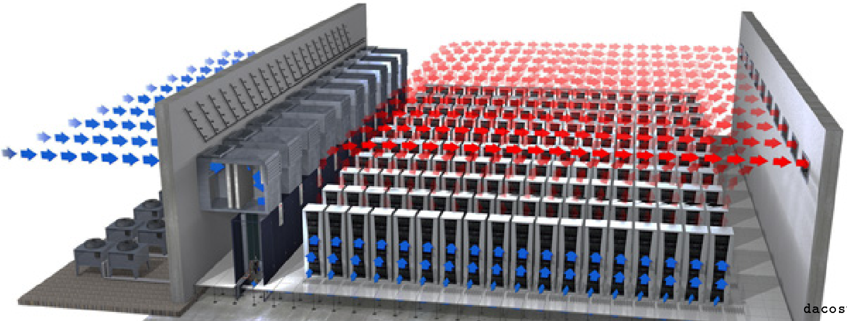
- Operator rent resources of its competitors (Telecom roaming)
- Similar method as *super-peers*
- Distributed ou centralized, similar performances

Thiam et al., *Cooperative Scheduling Anti-load balancing Algorithm for Cloud : CSAAC*, CCTS workshop, 2013



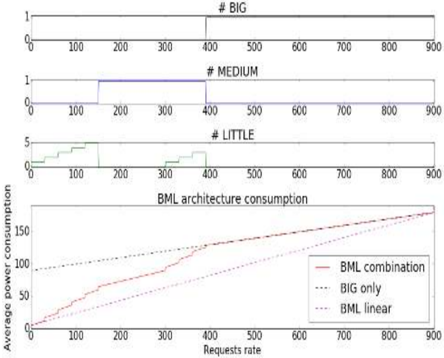
Follow the sun, Follow the moon

- Classical approach
  - Consolidation between datacenters
  - Coordinated management of Quality of Service (ex: CDN)
- Follow the state of datacenters
  - During night, less cooling cost
  - During day, more renewable energy production



At the scale of the *Data center in the box*

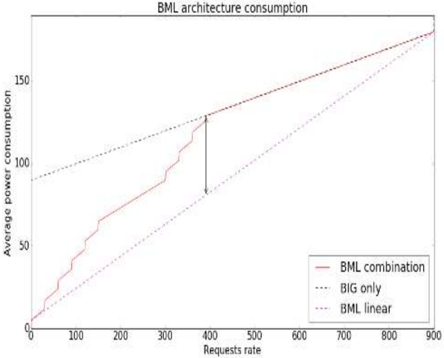
- Rack level
- Low number of services: High variance
- Perfect adaptation to the load
- Currently costly: Initial overhead
  - Cooling
  - Everything which is negative in the PUE
- *Proportional* architecture



Violaine et al., *Big, Medium, Little : Reaching Energy Proportionality with Heterogeneous Computing Scheduler*, Parallel Processing Letters, journal, 2015.

At the scale of the *Data center in the box*

- Rack level
- Low number of services: High variance
- Perfect adaptation to the load
- Currently costly: Initial overhead
  - Cooling
  - Everything which is negative in the PUE
- *Proportional* architecture



Violaine et al., *Big, Medium, Little : Reaching Energy Proportionality with Heterogeneous Computing Scheduler*, Parallel Processing Letters, journal, 2015.

Reaching energy-proportionality using heterogeneous hardware

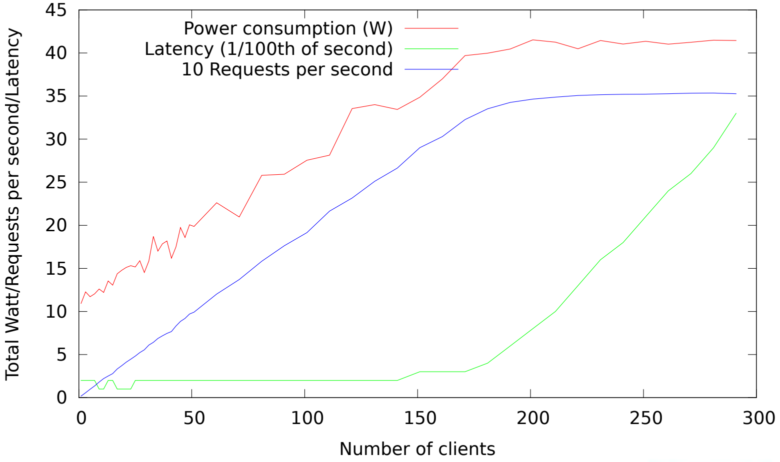
Use nodes depending on the real load (web server as example), not the peak load

Example:

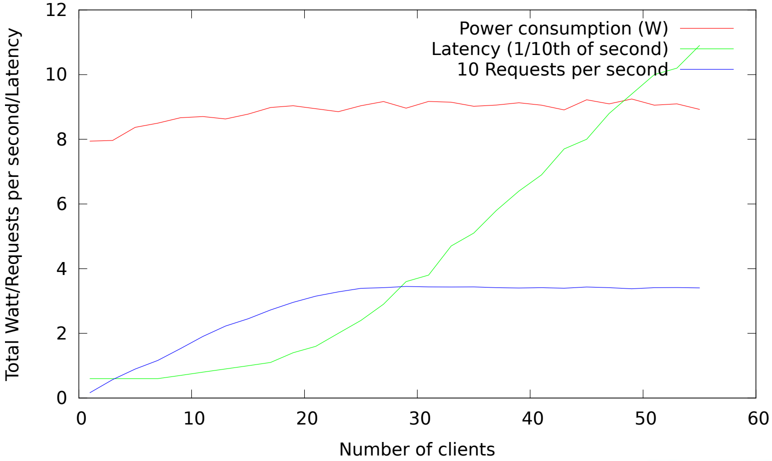
Processor	Watt range	Max request/s	Efficiency (W/r)
Intel I7	11 - 42	353	.12
Intel Atom	8 - 9	34	.26
Raspberry Pi	2.56 - 2.81	5.6	.50

Intuition: Several small node and intermediary nodes to have a multi-scale smooth curve

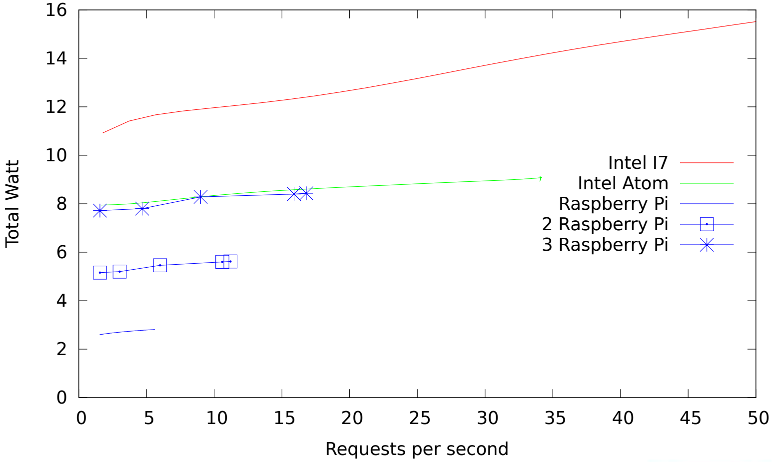
Zoom on Intel I7



Zoom on Intel Atom



Comparison


16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
35
40
45
50
Total Watt
Requests per second
Intel I7
Intel Atom
Raspberry Pi
2 Raspberry Pi
3 Raspberry Pi

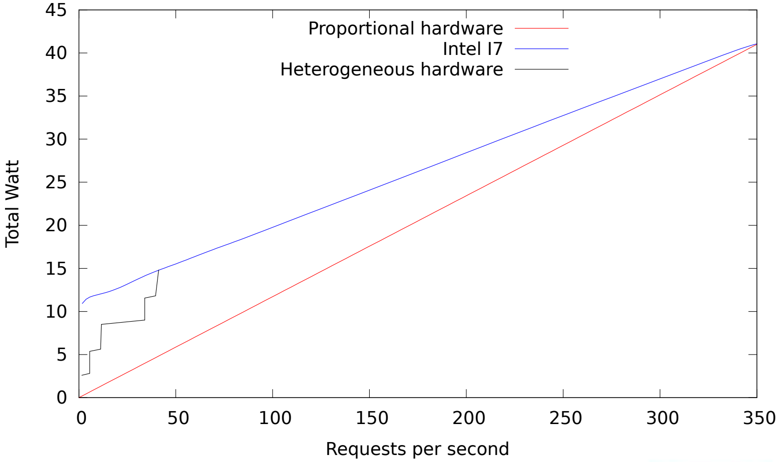
A near-linear behavior

Take the most efficient hardware as function of workload:

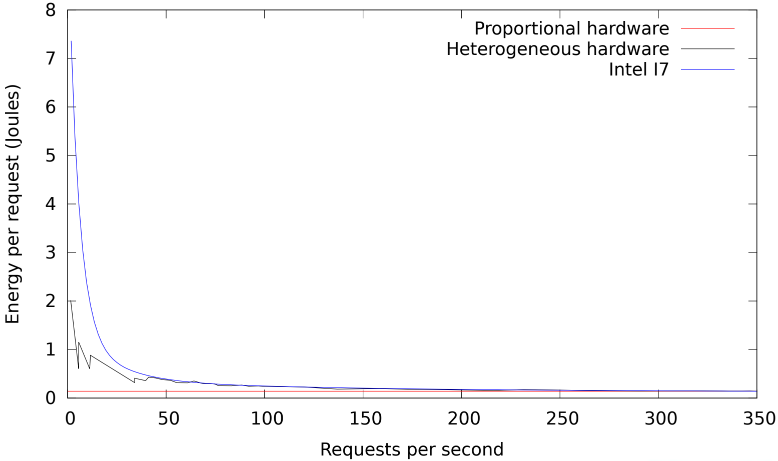
0→5 req/s	1 Raspberry Pi
5→10 req/s	2 Raspberry Pi
10→35 req/s	1 Intel Atom
35→40 req/s	1 Intel Atom + 1 Raspberry Pi
40→350 req/s	1 Intel I7

For over 350req/s, use modulo 350, and use as many I7 as necessary

Still far far away...

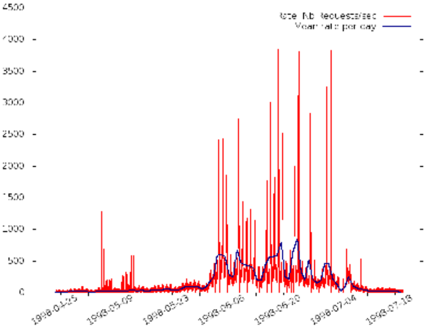


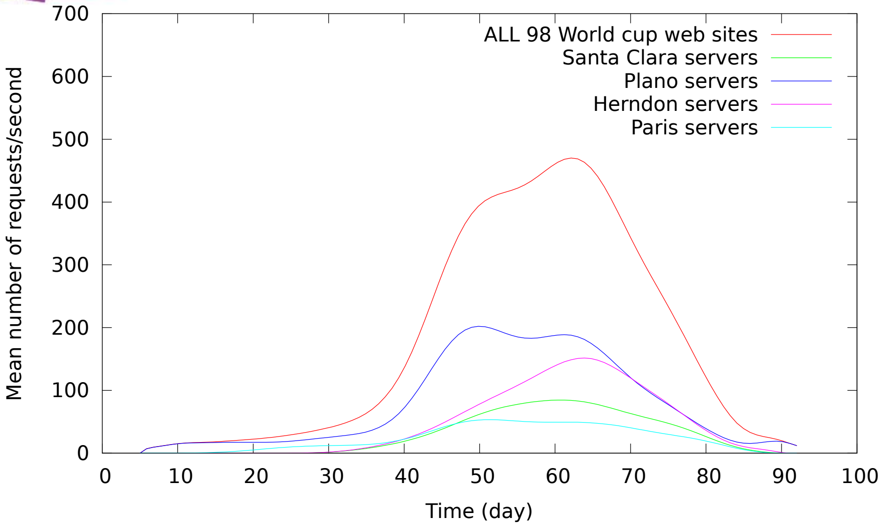
Not so far, efficiency view !



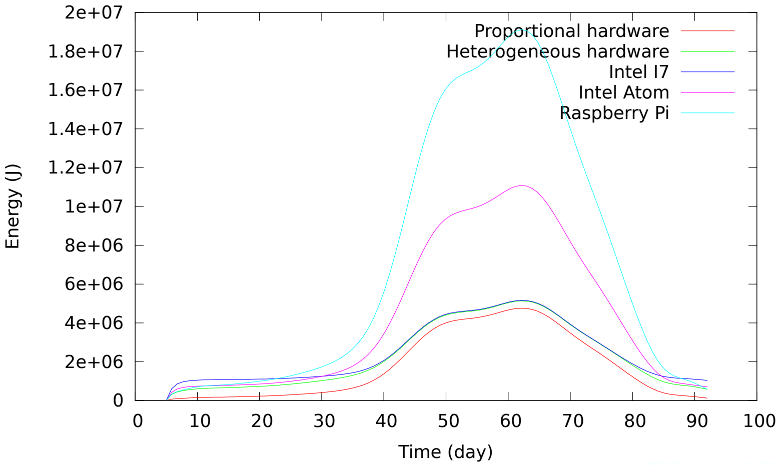
98 Football World Cup

- Available data
- 92 days of web server access logs
- Workload precise at the second level
- Four geographic locations, three in US, one in France
- Several phases
  - Low phases, first 40 days and last 10 days
  - High phase, during the competition

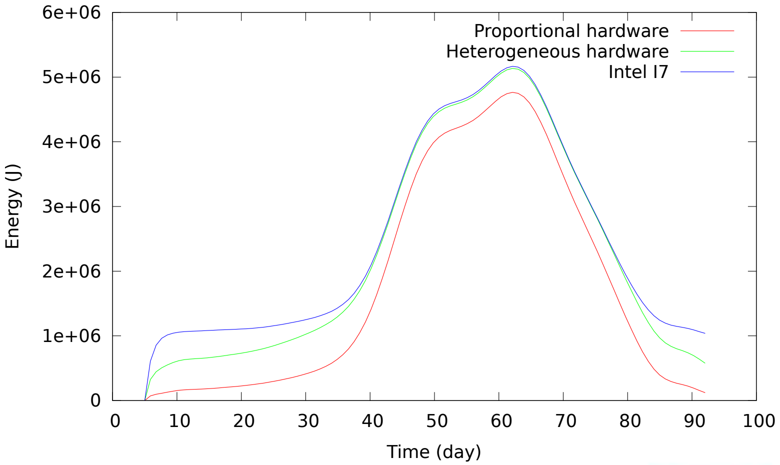



ALL 98 World cup web sites
Santa Clara servers
Plano servers
Herndon servers
Paris servers
Mean number of requests/second
Time (day)
0
100
200
300
400
500
600
700
0
10
20
30
40
50
60
70
80
90
100

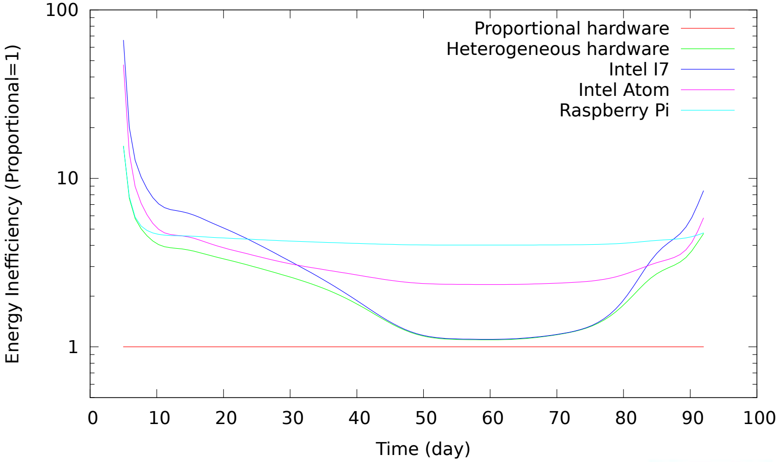
Comparison if using one single data-center



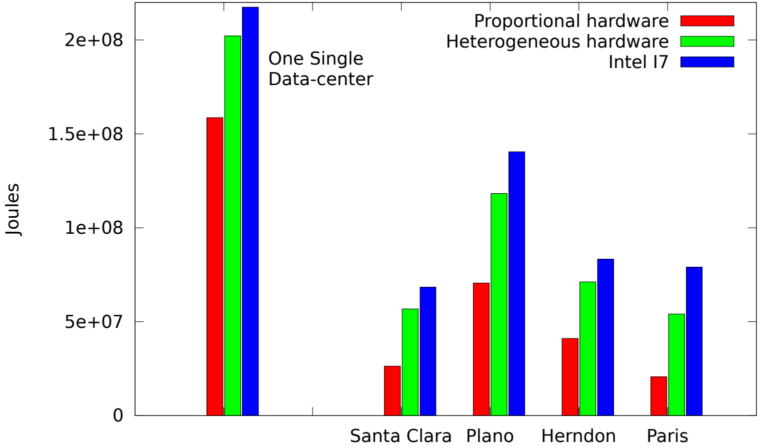
Zoom on the most efficient methods



Far reached goal: proportional computing

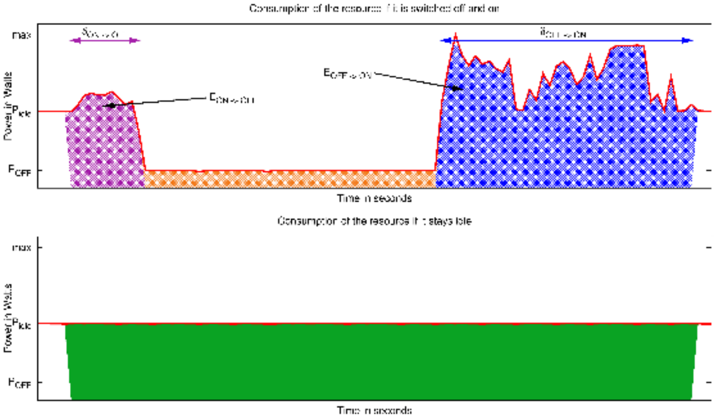


Comparison if using multiple data-centers



Adding management heterogeneity

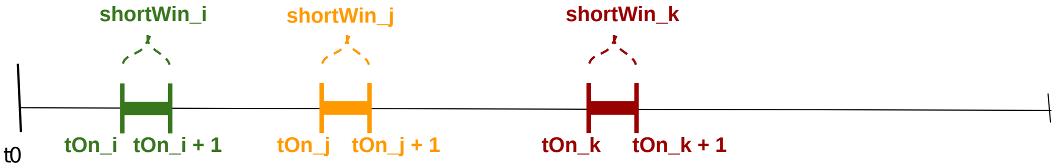
- Switching on/off nodes take time
- Switching on/off nodes consumes energy
  - For application reconfiguration
  - For switching on/off the server



Lefevre and al., Supercomputing 2008

Several methods to manage servers

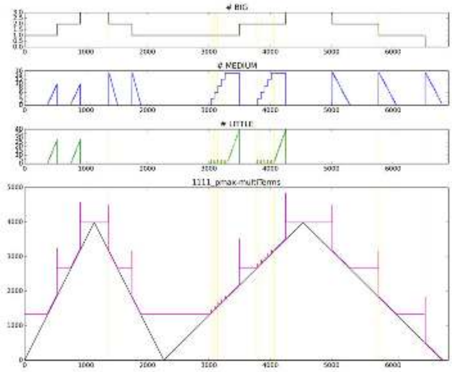
- Exact approach, linear programming
- Heuristics
  - Reactive
    - When overloaded, start new servers, when under-loaded stop some
  - Pro-active
    - Predict the future and decide which server to use



Violaine et al. *Energy Aware Dynamic Provisioning for Heterogeneous Data Centers*, SBAC-PAD 2016

Pro-active Heuristics

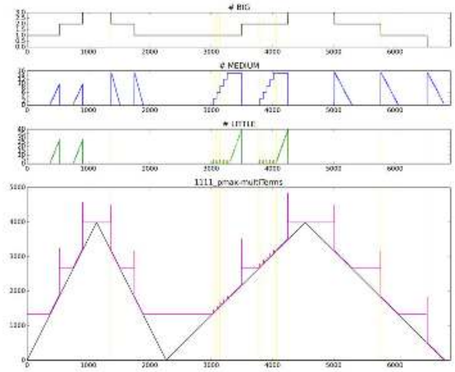
- Predict load and switch on nodes to guaranty QoS



Violaine et al., *Energy Proportionality in Heterogeneous Data Center Supporting Applications with Variable Load*, ICPDS 2016

Pro-active Heuristics

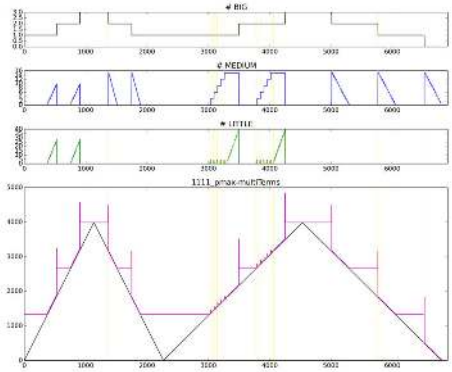
- Predict load and switch on nodes to guaranty QoS
- Switching on can be followed by switching off



Violaine et al., *Energy Proportionality in Heterogeneous Data Center Supporting Applications with Variable Load*, ICPDS 2016

Pro-active Heuristics

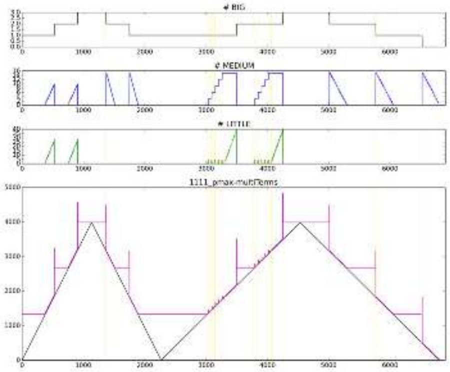
- Predict load and switch on nodes to guaranty QoS
- Switching on can be followed by switching off
- When load decrease, switch off servers accordingly



Violaine et al., *Energy Proportionality in Heterogeneous Data Center Supporting Applications with Variable Load*, ICPDS 2016

Pro-active Heuristics

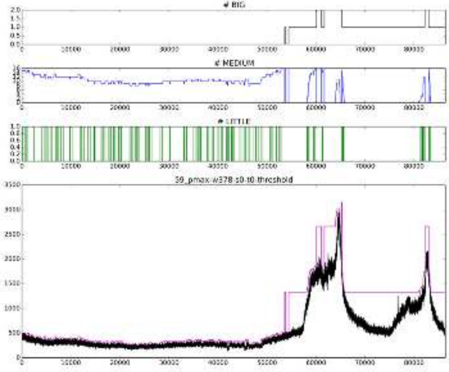
- Predict load and switch on nodes to guaranty QoS
- Switching on can be followed by switching off
- When load decrease, switch off servers accordingly
- Stay near the optimal repartition



Violaine et al., *Energy Proportionality in Heterogeneous Data Center Supporting Applications with Variable Load*, ICPDS 2016

Pro-active Heuristics

- Predict load and switch on nodes to guaranty QoS
- Switching on can be followed by switching off
- When load decrease, switch off servers accordingly
- Stay near the optimal repartition



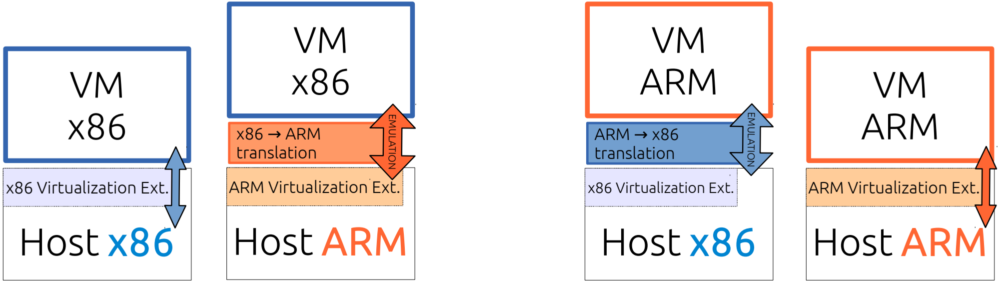
Violaine et al., *Energy Proportionality in Heterogeneous Data Center Supporting Applications with Variable Load*, ICPDS 2016

State-full distributed applications

- Classical applications are not maleable
  - Distributed databases
  - OpenMP or MPI applications
- Impossible to change the number of application instance
- Add migration time/energy to Switch on/off

Heterogeneous micro-architectures

- Instance can be migrated between architectures
- Performance depends on architecture of VM and server



- Translation slowdown ratio: 8
- Low load on ARM, high load on x86, x86 VM

Violaine and al. PPL 2005

Plan

Toward the future

- A larger number of datacenters
  - Lots of smaller ones (hybrid management)
    - The knowledge: critical resource
  - A large number of diverse sizes
  - Some larger (2016 : 6 300 000 m^2)
- Overall, datacenters will be more integrated in their environment
  - Electrical aspects
  - Thermal aspects

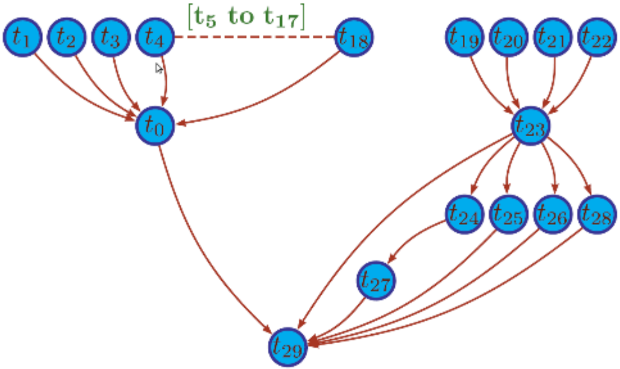
At the level of a node

- Three temporality
  - Large-grained (minute) : Optimal frequency in function of the task graph*
    - 13% of energy savings
  - Medium-grained (second) : Phase detection†
    - 20% of energy savings, 3% of time increase
  - Fined-grained (1/10s) : Frequency policy at the kernel level‡
    - 25% of energy savings, 1% of time decrease
- No coordination between the three temporality, no objectives

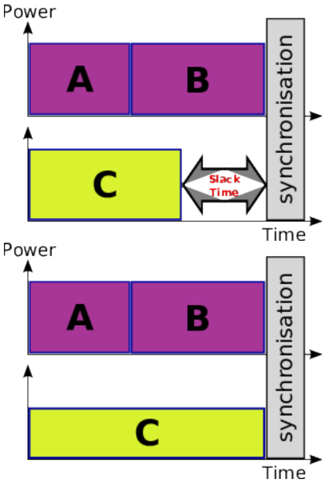
* Tom et al., *Energy-aware simulation with DVFS*, SMPT journal, 2013 †Landry et al., *Exploiting performance counters to predict and improve energy performance of HPC systems*, SUSCOM journal, 2014 ‡Georges et al., *DVFS governor for HPC: Higher, Faster, Greener*, PDP conférence, 2015

At the scale of a node: Large-grained

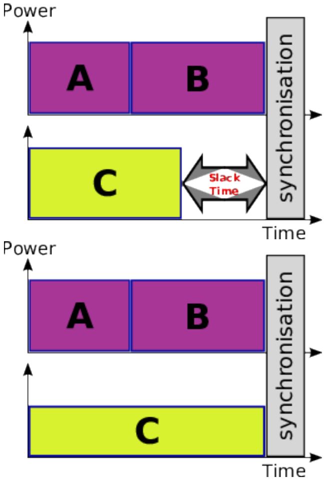
- Use of contextual external information
- Example at the scheduler level: Task DAG



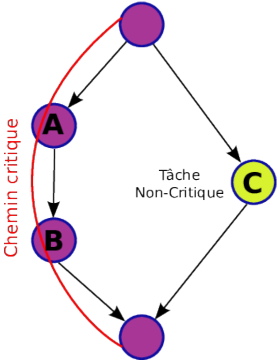
Coordination of node speeds



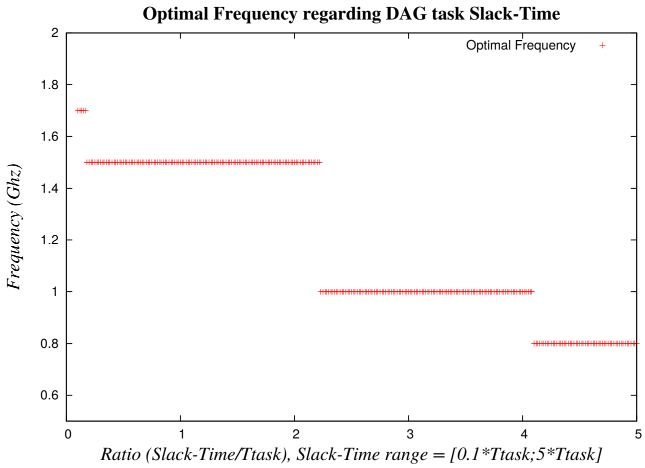
Coordination of node speeds



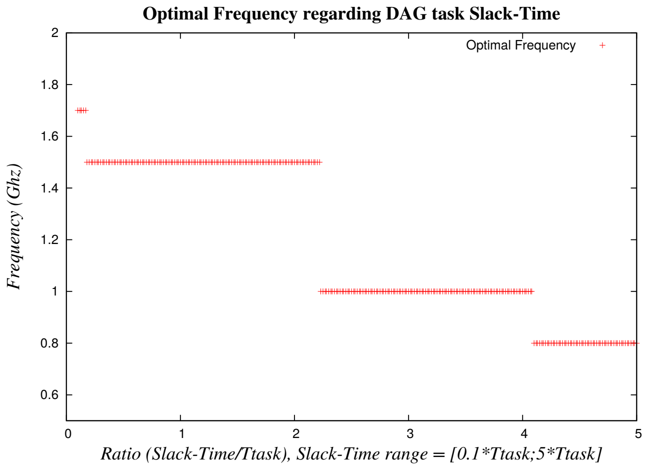
- Generalization toward critical path



Action at the node level



Action at the node level

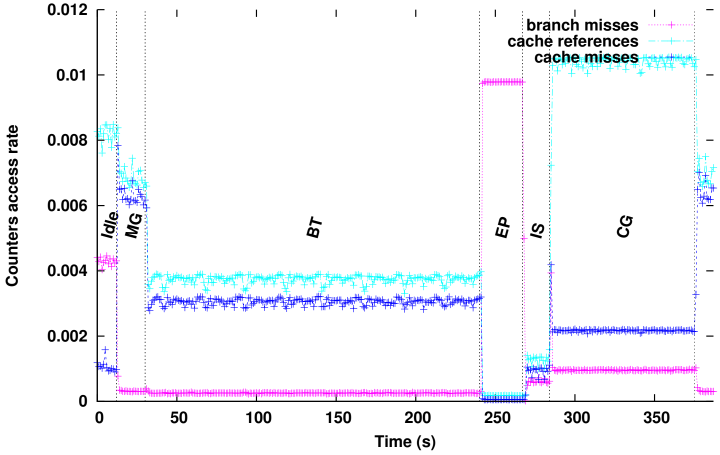


- To go further
  - Switching on/off servers
  - Manage temperature

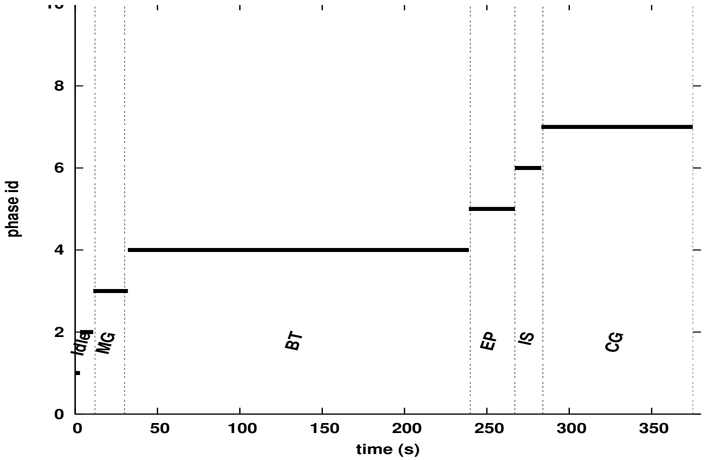
At the level of a node: Medium-grained

- React at medium latency at the level of the node
  - Change the processor frequency
  - Change the hard drive mode
  - Reconfigure the network card
- Detection of the current phase
- React in function of this profile
- Light impact on the infrastructure

Resource consumption of a complex application



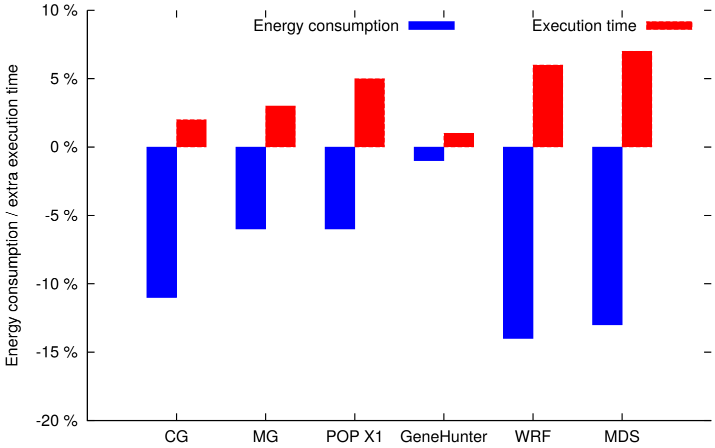
Phases where resource consumption are constant



Decision method

Phase label	Possible reconfiguration decisions
compute-intensive	switch off memory banks; send disks to sleep; scale the processor up; put NICs into LPI mode
memory -intensive	scale the processor down; decrease disks or send them to sleep; switch on memory banks
mixed	switch on memory banks; scale the processor up send disks to sleep; put NICs into LPI mode
communication intensive	switch off memory banks; scale the processor down switch on disks
IO-intensive	switch on memory banks; scale the processor down; increase disks, increase disks (if needed)

Energy and performance, 28 node



Fine-grained = DVFS ?

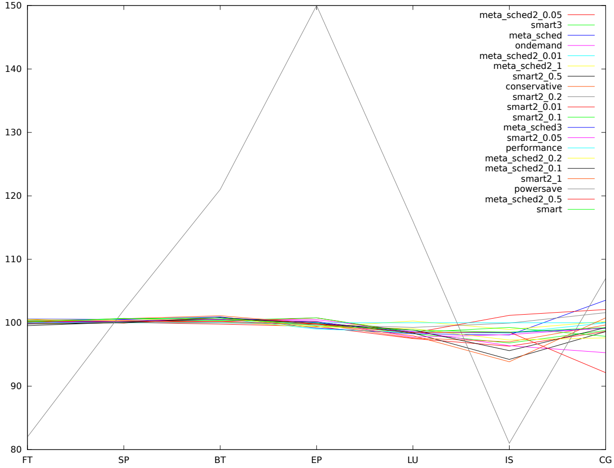
Relative values between *performance* and *ondemand* governors

Benchmark	FT	SP	BT	EP	LU	IS	CG
Time increase (%)	0	-3	-1	1	-2	2	0
Energy increase (%)	0	-3	-1	-1	-2	-1	-1

- HPC applications are rarely in Idle... Surprise !
- MPI libraries are spinning

Classical HPC benchmarks from NPB (Nas Parallel Benchmark)

DVFS = function of load



Yet DVFS has potential

Relative values between *performance* and *powersave* governors

Benchmark	FT	SP	BT	EP	LU	IS	CG
Time increase (%)	36	69	110	159	96	35	83
Energy increase (%)	-18	2	21	50	16	-19	7

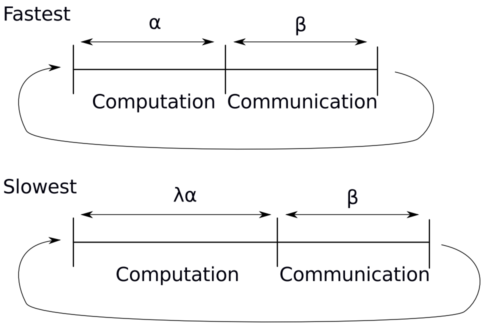
- Time rises, but up to 19% or energy consumption reduction !

HPC Hypothesis

- State of applications
  - Computing
  - Communications
  - Disk I/O
  - Idle

HPC Hypothesis

- State of applications
  - Computing
  - Communications



Decision

- Energy for maximum frequency

$$(\alpha + \beta)P_1$$

- Energy for minimum frequency

$$(\lambda\alpha + \beta)P_2$$

The processor stays at maximum frequency if it consumes less energy:

$$(\alpha + \beta)P_1 < (\lambda\alpha + \beta)P_2$$

How to measure α and β

- Difficult to measure directly α and β
  - Runtime, not code instrumentation
- Easy to measure network bandwidth (with B_m maximal bandwidth)

$$B_w = B_m \frac{\beta}{\alpha + \beta}$$

- In fact α and β are not important
  - $\frac{\alpha}{\beta}$ is needed, i.e. the ratio between *time to compute* and *time to communicate*

The great mix

Mix and serve:

$$B_w < \frac{B_m}{\lambda - 1}(\lambda - \frac{P_1}{P_2}) = B_1$$

B_1 : Bandwidth limit at maximum frequency to use or not DVFS

In the opposite direction

$$B_2 = \frac{B_m}{\lambda - 1}(\lambda \frac{P_2}{P_1} - 1)$$

B_2 : Bandwidth limit at minimum frequency to use or not DVFS

Adding an hysteresis for adding inertia

NetSched Algorithm

- Each 10^{th} of a second, do:
  - If Current_Frequency = Slowest frequency and IBR $\leq .9B_1$
    - Change frequency toward Fastest
  - If Current_Frequency = Fastest frequency and IBR $\geq 1.1B_2$
    - Change frequency toward Slowest
  - Else, do nothing

IBR : Incoming Byte Rate

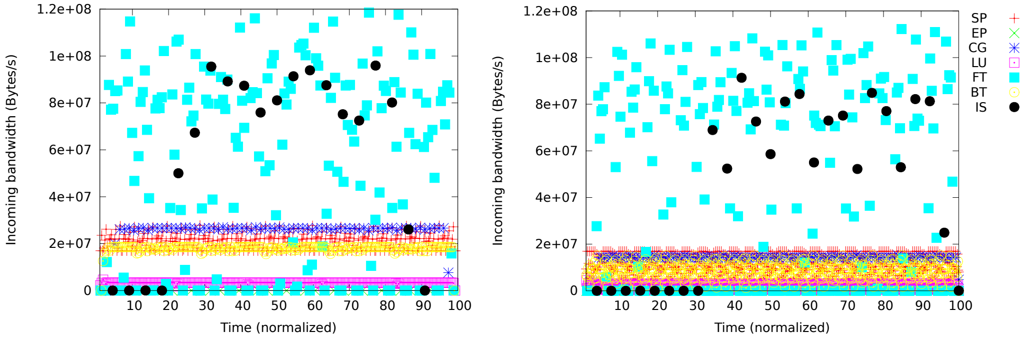
Experimental environment

- Servers (thanks Grid5000)
  - Processors : bi Dual-Core AMD Opteron (2218)
  - Memory : 8GB
  - Network card : Gigabyte Ethernet
  - Frequency : 2.6GHz and 1GHz
  - Electrical Power : $P_1 = 280W$ et $P_2 = 152W$
- Benchmark
  - 7 Nas Parallel Benchmark (NPB)
- Governors
  - Performance/Powersave/Ondemand
  - NetSched

$$1.1B_1 \simeq 7.10^7 \text{ et } 0.9B_2 \simeq 3.10^7$$

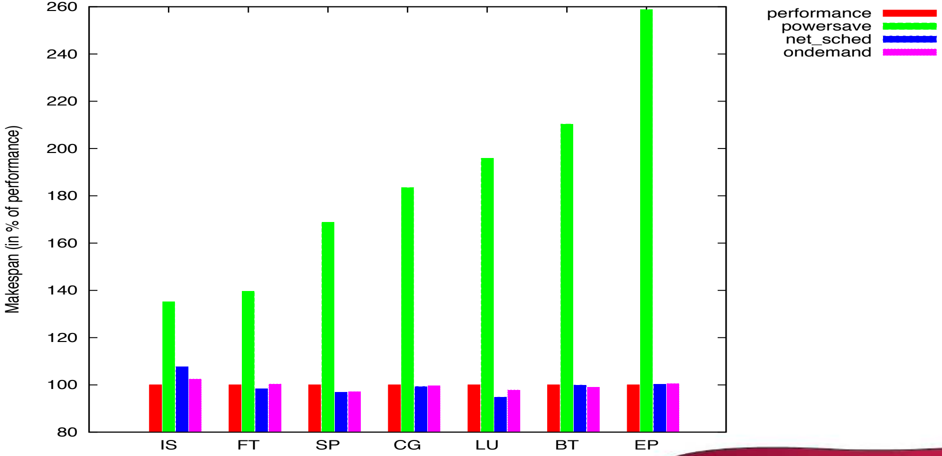
Example of execution

Network communication over time

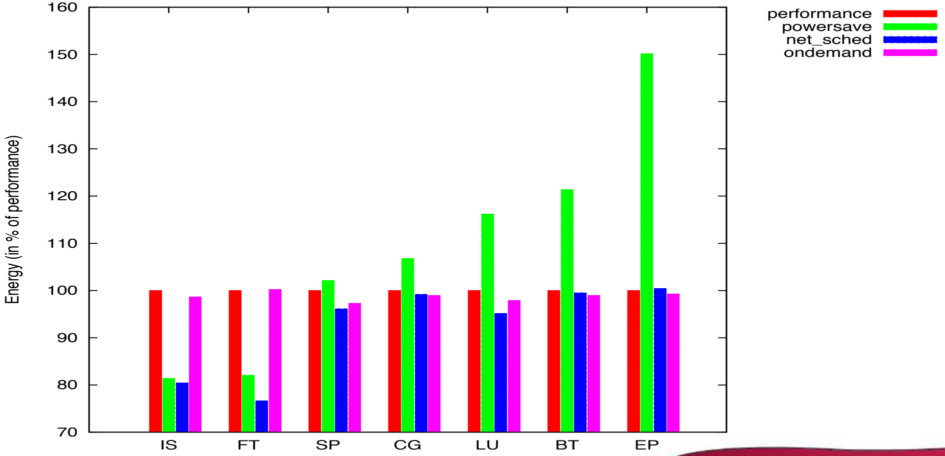


performance powersave

Results: Makespan



Results: Energy-to-solution



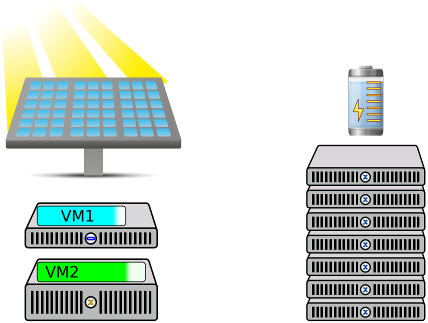
Plan

The missing link between levels

- An “handmade”work
  - A large number of inter-dependent middlewares
  - Human manipulations
- Toward a decentralized cooperation

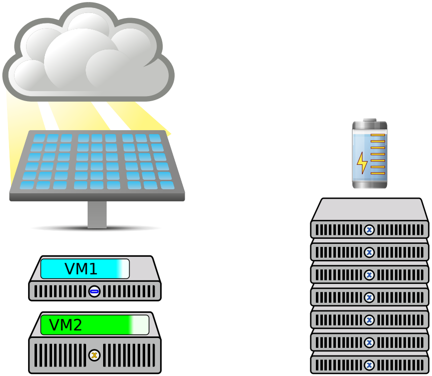
Cooperation between decision levels

1. Initial situation is stable



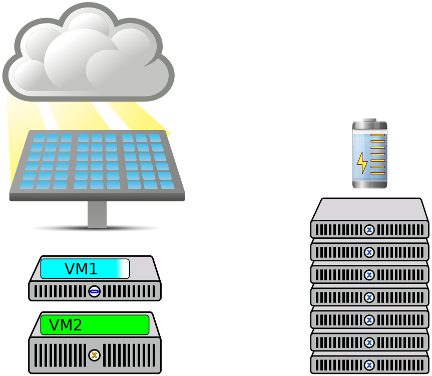
Cooperation between decision levels

1. Initial situation is stable
2. Decrease of solar production



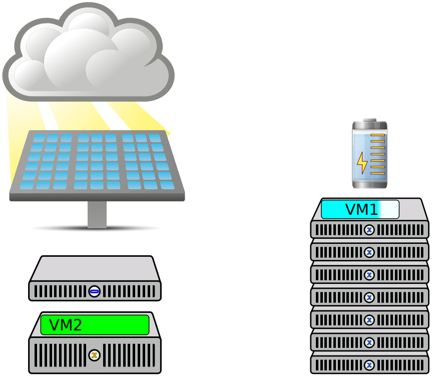
Cooperation between decision levels

1. Initial situation is stable
2. Decrease of solar production
3. Non-critical task: Aggressive DVFS
3. Critical task: unavailable dynamism



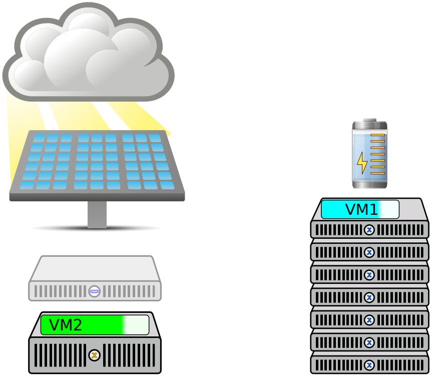
Cooperation between decision levels

1. Initial situation is stable
2. Decrease of solar production
3. Non-critical task: Aggressive DVFS
3. Critical task: unavailable dynamism
4. Critical task: looks for an adequate location



Cooperation between decision levels

1. Initial situation is stable
2. Decrease of solar production
3. Non-critical task: Aggressive DVFS
3. Critical task: unavailable dynamism
4. Critical task: looks for an adequate location
5. Switching off a server
6. Less aggressive DVFS



Open research questions

- Programming paradigms
  - Ability to describe parallelism intuitively
  - Remove the burden from developer
- Runtimes
  - Capability to adapt to particular profiles and their interactions
  - Ability to change kernels in function of context
- Communication between these two levels
- Cooperation between operators
  - Cloud federation
  - Cloud and HPC systems