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 CO₂ world production
  - 2011: 6ᵗʰ 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
  - 17.8 MW
Sustainable datacenters

- Action can be done at several different levels
  - Hardware level: changing servers or cooling system
    - If entropy is constant, theoretical energy consumption 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)

Low utilization = high electrical waste

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

Autonomic loop

Complex sensors

Impact of measures

Linear Programming

Metrics

Real leverages

Measure

Analyze

Execute

Planing

Genetic Algorithms

Simulations

Melbourne Clouds Lab

Vector packing
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

1. Autonomic loop
   - Models and Metrics
   - Measures
   - Evaluation tools
2. Decision
   - Placement
   - Cloud federation
   - Data center in the box
3. Evolution, nodes optimization
   - Large-grained
   - Medium-grained level
   - Fine-grained level
4. And beyond
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:

\[ P_{\text{cmos}} = C_{\text{eff}} \times V^2 \times f \]

with, \( C_{\text{eff}} \) 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:

\[ E = P \times T \propto T \times V^3 \], avec \( V \propto f \) et \( T \propto 1/F \), 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
  - 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, 20\(\mu\)s 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 growth

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

<table>
<thead>
<tr>
<th>Processor Type</th>
<th>Quantity</th>
<th>Processor Type</th>
<th>Quantity</th>
<th>Processor Type</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD Opteron 2218</td>
<td>100</td>
<td>Intel Xeon E5-2620</td>
<td>8</td>
<td>Intel Xeon E5420</td>
<td>68</td>
</tr>
<tr>
<td>AMD Opteron 250</td>
<td>158</td>
<td>Intel Xeon E5-2620 v3</td>
<td>12</td>
<td>Intel Xeon E5440</td>
<td>92</td>
</tr>
<tr>
<td>AMD Opteron 6164 HE</td>
<td>168</td>
<td>Intel Xeon E5-2630</td>
<td>40</td>
<td>Intel Xeon E5520</td>
<td>254</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel Xeon E5-2630 v3</td>
<td>350</td>
<td>Intel Xeon E5620</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel Xeon E5-2630L</td>
<td>32</td>
<td>Intel Xeon E7450</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel Xeon E5-2650</td>
<td>8</td>
<td>Intel Xeon L5335</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel Xeon E5-2660</td>
<td>44</td>
<td>Intel Xeon L5420</td>
<td>332</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel Xeon E5-2660 v2</td>
<td>16</td>
<td>Intel Xeon X3440</td>
<td>144</td>
</tr>
<tr>
<td>Intel Xeon E5-2630 v3</td>
<td>12</td>
<td>Intel Xeon E5420</td>
<td>68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Xeon E5440</td>
<td>92</td>
<td>Intel Xeon E5520</td>
<td>254</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Xeon E5520</td>
<td>254</td>
<td>Intel Xeon E5620</td>
<td>56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Xeon E7450</td>
<td>52</td>
<td>Intel Xeon L5335</td>
<td>44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Xeon L5420</td>
<td>332</td>
<td>Intel Xeon X3440</td>
<td>144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Xeon X5570</td>
<td>50</td>
<td>Intel Xeon X5670</td>
<td>88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Xeon X5670</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IT departments evolve

- Large institutions are built over years
- Smallest one do not necessarily 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 $E \sim 10-15\%$)

$$Power = P_{\min} + Load \times (P_{\max} - P_{\min})$$
Even models are complex

Electrical power models for a single server:

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

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

\[
\text{Power} = P_{\text{min}} + \text{Load} \times \alpha \text{Voltage}^2 \text{Frequency} + \lambda \text{Temperature}
\]
Even models are complex

Electrical power models for a single server:

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

\[ \text{Power}_\text{DC} = \omega_0 + \omega_1 \text{Power}_\text{AC} + \omega_2 \text{Power}_\text{AC}^3 \]
Even models are complex

Electrical power models for a single server:

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

* 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
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

* 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?

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

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%
     \[ PUE = \frac{275}{175} = 1.57 \]
  2. Mean load 100%
     \[ PUE = \frac{300}{200} = 1.5 \]
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)

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 ($x$)
- $Obj_\alpha(x) = \alpha M_q + (1-\alpha) M_l = \alpha x^2 + (1-\alpha)(1-x)$
- $\alpha$ is the weighing coefficient
How to choose $\alpha$?

Optimal value of frequency in function of $\alpha$ 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!

$\alpha$: Relative importance of Performance (1) and Power (0)
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

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 $\sim$ Same Impact


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
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

dacosta@irit.fr
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.

Power Model: Neural Network (all)

Power Model: $w_0 + w_1 \cdot CPU_u \cdot CPU_{af}$

MAPE = 1.83%

$R^2 = 0.9509$

MAPE = 5.42%

$R^2 = 0.7144$
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
  - Scaling
  - 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
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 $h$
- $e_{jh} = 1$ if task $j$ is on server $h$
- $\forall j, h \ e_{jh} \in \{0, 1\}$,
  $\forall j \ \sum_h e_{jh} = 1$
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 $\alpha_{jh}$ the processor fraction of task $j$ on server $h$.
- $\min \sum_{h} \left( P_{h}^{stat} + \sum_{j} \alpha_{jh} P_{h}^{dyn} \right)$
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:
\[
\forall j \sum_h e_{jh} = 1, \quad \forall j, h \ e_{jh} \in \{0, 1\}, \quad \forall j, \ h_\alpha_{jh} \leq e_{jh}, \ m_{jh} \leq e_{jh} \\
\forall j \ v_j \in \{0, 1\}, \quad \forall h \sum_j \alpha_{jh} \leq v_j, \sum_j m_{jh} \leq v_j \\
\forall j, h \ p_h \leq (1 - e_{jh}) + v_j \\
\forall j, h \ v_j \leq (1 - e_{jh}) + p_h \\
\forall h \sum_j \alpha_{jh} / r_j \leq v_j \\
\forall h \sum_j m_{jh} \leq v_j \\
\]

Minimize power under performance constraints:
\[
\forall j \sum_h \alpha_{jh} / r_j > \text{Threshold} \\
\min(\sum_h (P_h^{min} + \sum_j \alpha_{jh} \times (P_h^{max} - P_h^{min})))
\]

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 half on another one)
- Using both it is possible to have an 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

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

![Graph showing CPU charge over time](image)
Ad-hoc simulators: Reinvent the wheel?

- Simplistic simulators
- Let's 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

Experimentation

- A model is always an approximation
- Final validation by experiment
- Complex because of the need to have electrical measures
  - At ENS-Lyon, they were 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

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 ≠ 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

1. Autonomic loop
   - Models and Metrics
   - Measures
   - Evaluation tools

2. Decision
   - Placement
   - Cloud federation
   - Data center in the box

3. Evolution, nodes optimization
   - Large-grained
   - Medium-grained level
   - Fine-grained level

4. And beyond
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 $O(J \times H \ln(H))$
- But the solution of the Gourmet is difficult to qualify

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

<table>
<thead>
<tr>
<th>GA Name</th>
<th>Energy</th>
<th>Response time</th>
<th>Robustness</th>
<th>Dynamism</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA_All</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>GA_Energy</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GA_RespT</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GA_Rob</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>GA_Dyn</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
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 or 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

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

Reaching energy-proportionality using heterogeneous hardware

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

<table>
<thead>
<tr>
<th>Processor</th>
<th>Watt range</th>
<th>Max request/s</th>
<th>Efficiency (W/r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel I7</td>
<td>11 - 42</td>
<td>353</td>
<td>.12</td>
</tr>
<tr>
<td>Intel Atom</td>
<td>8 - 9</td>
<td>34</td>
<td>.26</td>
</tr>
<tr>
<td>Raspberry Pi</td>
<td>2.56 - 2.81</td>
<td>5.6</td>
<td>.50</td>
</tr>
</tbody>
</table>

Intuition: Several small node and intermediary nodes to have a multi-scale smooth curve
Zoom on Intel I7

Power consumption (W)  
Latency (1/100th of second)  
10 Requests per second
Zoom on Intel Atom

![Graph showing power consumption and latency against number of clients. The x-axis represents the number of clients ranging from 0 to 60, while the y-axis represents total wattage and requests per second ranging from 0 to 12. Lines represent power consumption (W), latency (1/10th of second), and 10 Requests per second.]

dacosta@irit.fr

60/102
Comparison

![Graph showing comparison of total wattage and requests per second for different devices: Intel I7, Intel Atom, Raspberry Pi, 2 Raspberry Pi, and 3 Raspberry Pi. The x-axis represents requests per second, ranging from 0 to 50, and the y-axis represents total watt, ranging from 0 to 16. The graph indicates a clear increase in wattage with an increase in requests per second for all devices.](image-url)
A near-linear behavior

Take the most efficient hardware as function of workload:

<table>
<thead>
<tr>
<th>req/s range</th>
<th>Hardware Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0→5</td>
<td>1 Raspberry Pi</td>
</tr>
<tr>
<td>5→10</td>
<td>2 Raspberry Pi</td>
</tr>
<tr>
<td>10→35</td>
<td>1 Intel Atom</td>
</tr>
<tr>
<td>35→40</td>
<td>1 Intel Atom + 1 Raspberry Pi</td>
</tr>
<tr>
<td>40→350</td>
<td>1 Intel I7</td>
</tr>
</tbody>
</table>

For over 350 req/s, use modulo 350, and use as many I7 as necessary.
Still far far away...

![Graph showing the relationship between total wattage and requests per second for proportional hardware, Intel I7, and heterogeneous hardware.]
Not so far, efficiency view!

Energy per request (Joules) vs. Requests per second for Proportional, Heterogeneous, and Intel I7 hardware configurations.
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
Comparison if using one single data-center

![Graph showing energy consumption over time for different hardware types.]

- Proportional hardware
- Heterogeneous hardware
- Intel I7
- Intel Atom
- Raspberry Pi

Energy (J) vs. Time (day)
Zoom on the most efficient methods

![Graph showing energy consumption over time for different hardware configurations.]

- Proportional hardware
- Heterogeneous hardware
- Intel I7
Far reached goal: proportional computing
Comparison if using multiple data-centers

![Graph showing Joules for Santa Clara, Plano, Herndon, and Paris with One Single Data-center and Heterogeneous hardware.]

Da Costa, Heterogeneity, the key to reach proportionality, CCGRID 2014

dacosta@irit.fr
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

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
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

1. Autonomic loop
   - Models and Metrics
   - Measures
   - Evaluation tools

2. Decision
   - Placement
   - Cloud federation
   - Data center in the box

3. Evolution, nodes optimization
   - Large-grained
   - Medium-grained level
   - Fine-grained level

4. And beyond
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 m²)
- 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
  - Fine-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 conference, 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

Optimal Frequency regarding DAG task Slack-Time

Frequency (Ghz) vs. Ratio (Slack-Time/Ttask), Slack-Time range = [0.1*Ttask;5*Ttask]
Action at the node level

Optimal Frequency regarding DAG task Slack-Time

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

<table>
<thead>
<tr>
<th>Phase label</th>
<th>Possible reconfiguration decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute-intensive</td>
<td>switch off memory banks; send disks to sleep; scale the processor up; put NICs into LPI mode</td>
</tr>
<tr>
<td>memory-intensive</td>
<td>scale the processor down; decrease disks or send them to sleep; switch on memory banks</td>
</tr>
<tr>
<td>mixed</td>
<td>switch on memory banks; scale the processor up; send disks to sleep; put NICs into LPI mode</td>
</tr>
<tr>
<td>communication</td>
<td>switch off memory banks; scale the processor down; switch on disks</td>
</tr>
<tr>
<td>intensive</td>
<td></td>
</tr>
<tr>
<td>IO-intensive</td>
<td>switch on memory banks; scale the processor down; increase disks, increase disks (if needed)</td>
</tr>
</tbody>
</table>
Energy and performance, 28 node

Energy consumption / extra execution time

<table>
<thead>
<tr>
<th>Tool</th>
<th>Energy consumption</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>-10%</td>
<td>0%</td>
</tr>
<tr>
<td>MG</td>
<td>-5%</td>
<td>-5%</td>
</tr>
<tr>
<td>POP X1</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>GeneHunter</td>
<td>-10%</td>
<td>-10%</td>
</tr>
<tr>
<td>WRF</td>
<td>5%</td>
<td>5%</td>
</tr>
<tr>
<td>MDS</td>
<td>10%</td>
<td>10%</td>
</tr>
</tbody>
</table>

-dacosta@irit.fr
Fine-grained = DVFS?

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>FT</th>
<th>SP</th>
<th>BT</th>
<th>EP</th>
<th>LU</th>
<th>IS</th>
<th>CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time increase (%)</td>
<td>0</td>
<td>-3</td>
<td>-1</td>
<td>1</td>
<td>-2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Energy increase (%)</td>
<td>0</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>FT</th>
<th>SP</th>
<th>BT</th>
<th>EP</th>
<th>LU</th>
<th>IS</th>
<th>CG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time increase (%)</td>
<td>36</td>
<td>69</td>
<td>110</td>
<td>159</td>
<td>96</td>
<td>35</td>
<td>83</td>
</tr>
<tr>
<td>Energy increase (%)</td>
<td>-18</td>
<td>2</td>
<td>21</td>
<td>50</td>
<td>16</td>
<td>-19</td>
<td>7</td>
</tr>
</tbody>
</table>

- 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

Fastest

\[ \alpha \quad \beta \]

Computation Communication

Slowest

\[ \lambda\alpha \quad \beta \]

Computation Communication
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 $\alpha$ and $\beta$

- Difficult to measure directly $\alpha$ and $\beta$
  - Runtime, not code instrumentation
- Easy to measure network bandwidth (with $B_m$ maximal bandwidth)

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

- In fact $\alpha$ and $\beta$ 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

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

\( B_2 \): Bandwidth limit at minimum frequency to use or not DVFS
Adding an hysteresis for adding inertia

NetSched Algorithm

- Each 10\textsuperscript{th} of a second, do:
  - If Current\_Frequency = Slowest frequency and IBR ≤ 0.9\textsuperscript{B\textsubscript{1}}
    - Change frequency toward Fastest
  - If Current\_Frequency = Fastest frequency and IBR ≥ 1.1\textsuperscript{B\textsubscript{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

Incoming bandwidth (Bytes/s) vs Time (normalized)

- SP
- EP
- CG
- LU
- FT
- BT
- IS

Performance vs Powersave

dacosta@irit.fr
Results: Makespan

![Graph showing Makespan results for different workloads.

- IS: Performance
- FT: Powersave
- SP: Net_sched
- CG: On-demand
- LU: BT: EP:

Makespan (in % of performance)

80 100 120 140 160 180 200 220 240 260]
Results: Energy-to-solution
Plan

1. Autonomic loop
   - Models and Metrics
   - Measures
   - Evaluation tools

2. Decision
   - Placement
   - Cloud federation
   - Data center in the box

3. Evolution, nodes optimization
   - Large-grained
   - Medium-grained level
   - Fine-grained level

4. And beyond
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
4. Critical task: unavailable dynamism

VM1
VM2

dacosta@irit.fr
Cooperation between decision levels

1. Initial situation is stable
2. Decrease of solar production
3. Non-critical task: Aggressive DVFS
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
4. **Critical task**: unavailable dynamism
5. **Critical task**: looks for an adequate location
6. Switching off a server
7. 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