MANCHESTER 1824

The University of Manchester

Parallel Computing: from traditional execution time minimization to multi-objective optimization

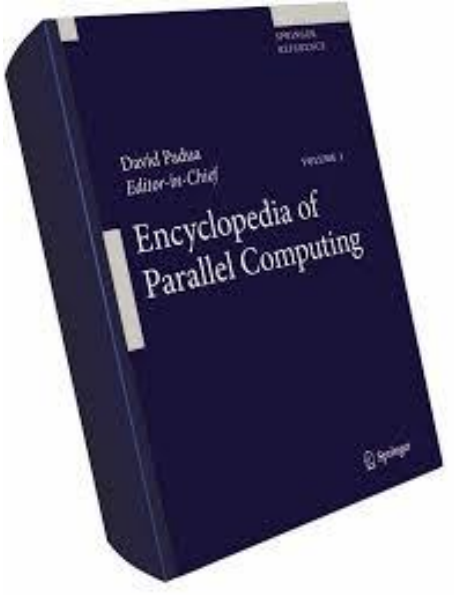
Rizos Sakellariou

but primarily thanks to the work of Ilia Pietri

(and many other people who helped over the years)

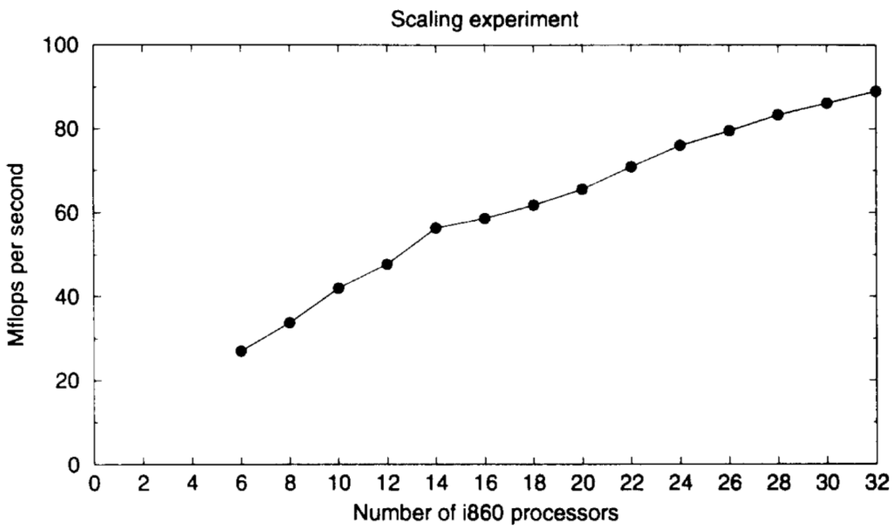
“Parallel computing is generally concerned with reducing the amount of time necessary to perform a computational task”

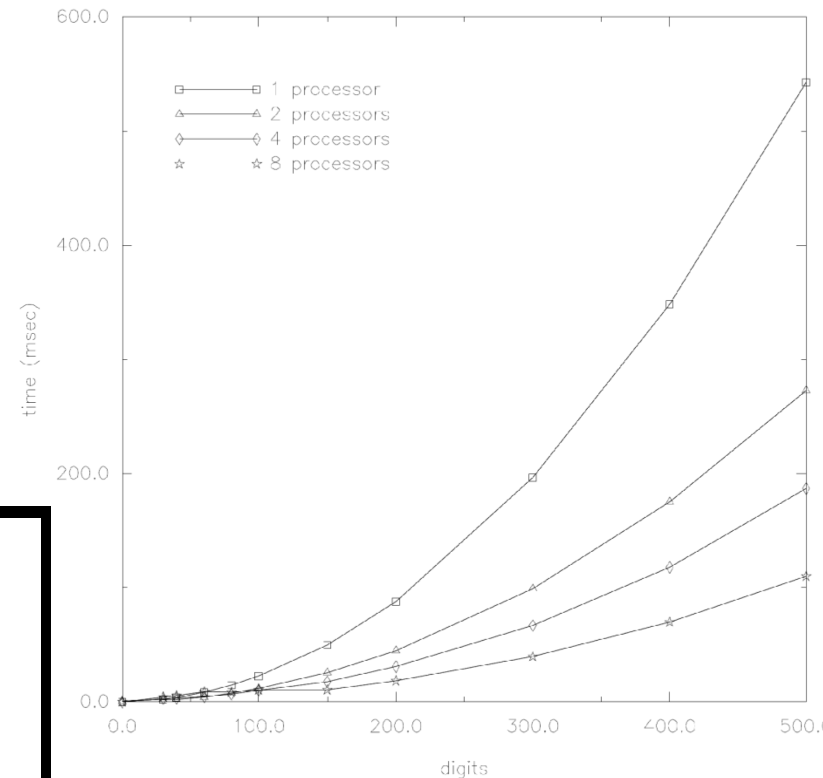
Encyclopedia of Parallel Computing
(D. Padua EiC), p. 1125

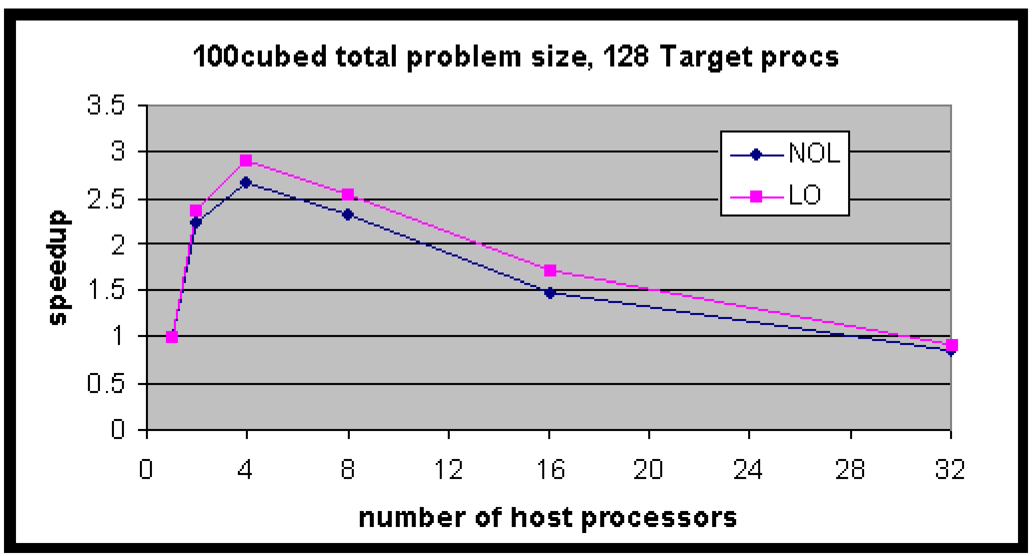


Such a reduction in time would be presented in different ways…

H.J.C. Berendsen et al. / Computer Physics Communications 91 (1995) 43–56

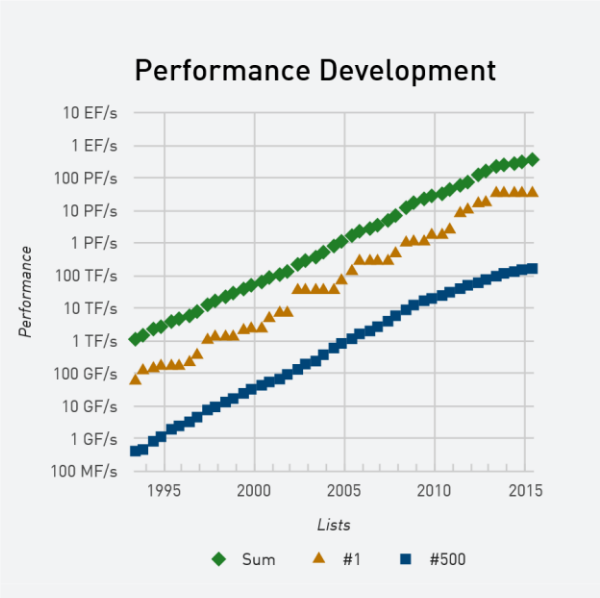






…and would drive our understanding

- Understand the inherent limitations of applications when parallelizing them:
  - Amdahl’s law
- Classify the performance of different machines
  - Top 500



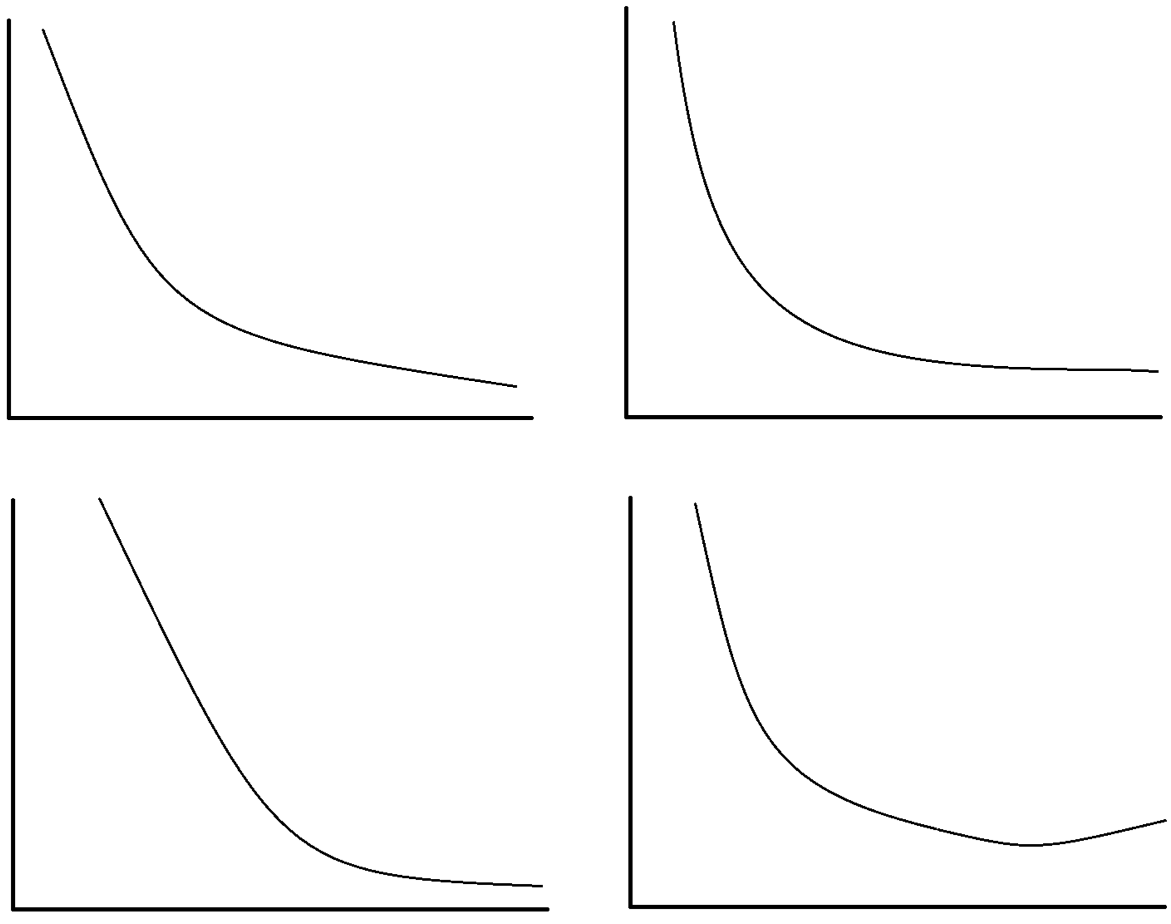
Minimizing execution time is a key goal

…but at the same time there is a cost to pay – and we cannot keep ignoring it!

- Modern machines/platforms are becoming too expensive to run.
  - Energy
- <u>Heterogeneity</u> increases the complexity of parallelization (and the search space)
  - Not simply ‘as many processors’ any more
  - We need to be able to differentiate between more costly and less costly solutions

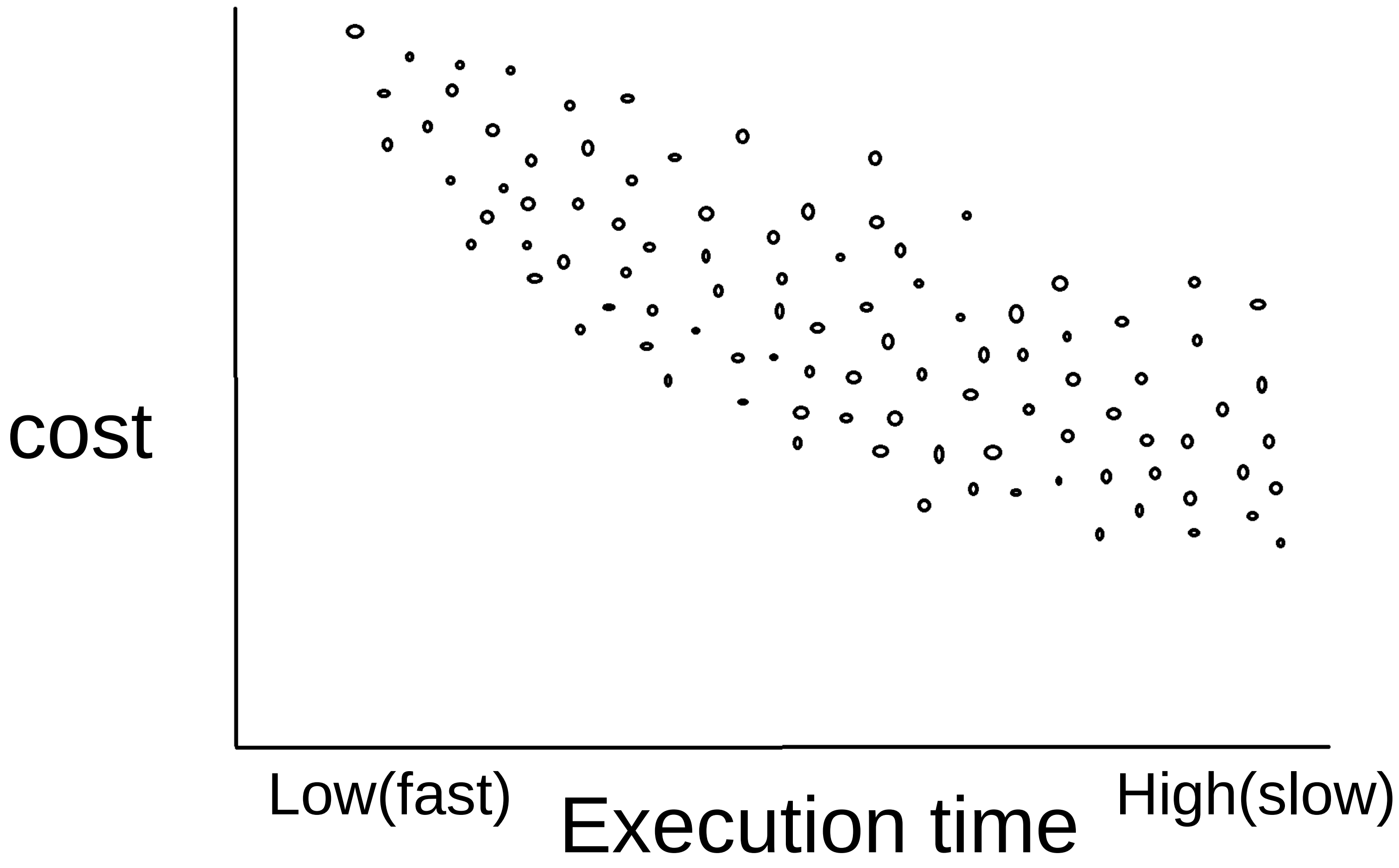
E.g., choose from fast/slow CPUs & GPUs (x% fast CPUs, y% slow CPUs, z% GPUs)

The curve of performance (y-axis) vs #processors (x-axis) differs – and we can get lots of different graphs!



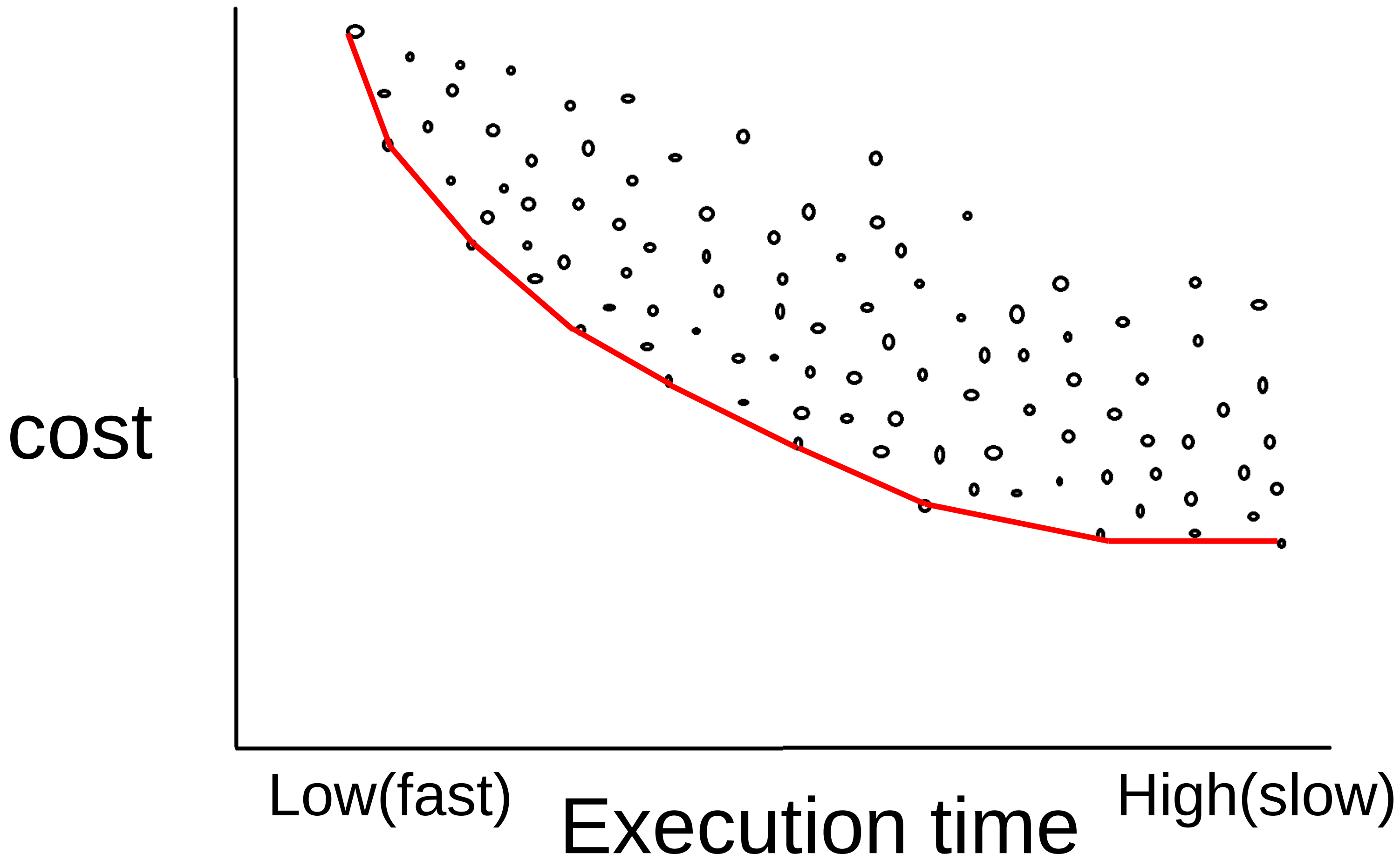
Assessing the cost of different options

We can merge all options into one graph



Assessing the cost of different options

Then, ‘best’ solutions are given by a Pareto front



Are we in good shape to find such solutions?

Essentially, the best we could do is search exhaustively or use some (basic?) intuition

A problem (motivated by Cloud providers)

CloudSigma

FEATURES PRICING ABOUT COMMUNITY BLOG LOGIN ENGLISH

2. Define your compute resources

Resource	Amount	Cost (CHF)
CPU/ GHz (1 core = 2.5GHz)	1.5	7.34
RAM/ GB	1	0.00
SSD Storage/ GB	50	0.00
Magnetic Storage/ GB	0	0.00
Data Transfer/ GB	5120	0.00
Static IPs	0	0.00
	Total cost	7.34

Calculate

2. Define your compute resources

Resource	Amount	Cost (CHF)
CPU/ GHz (1 core = 2.5GHz)	1.7	8.32

You pay depending on the CPU frequency you choose

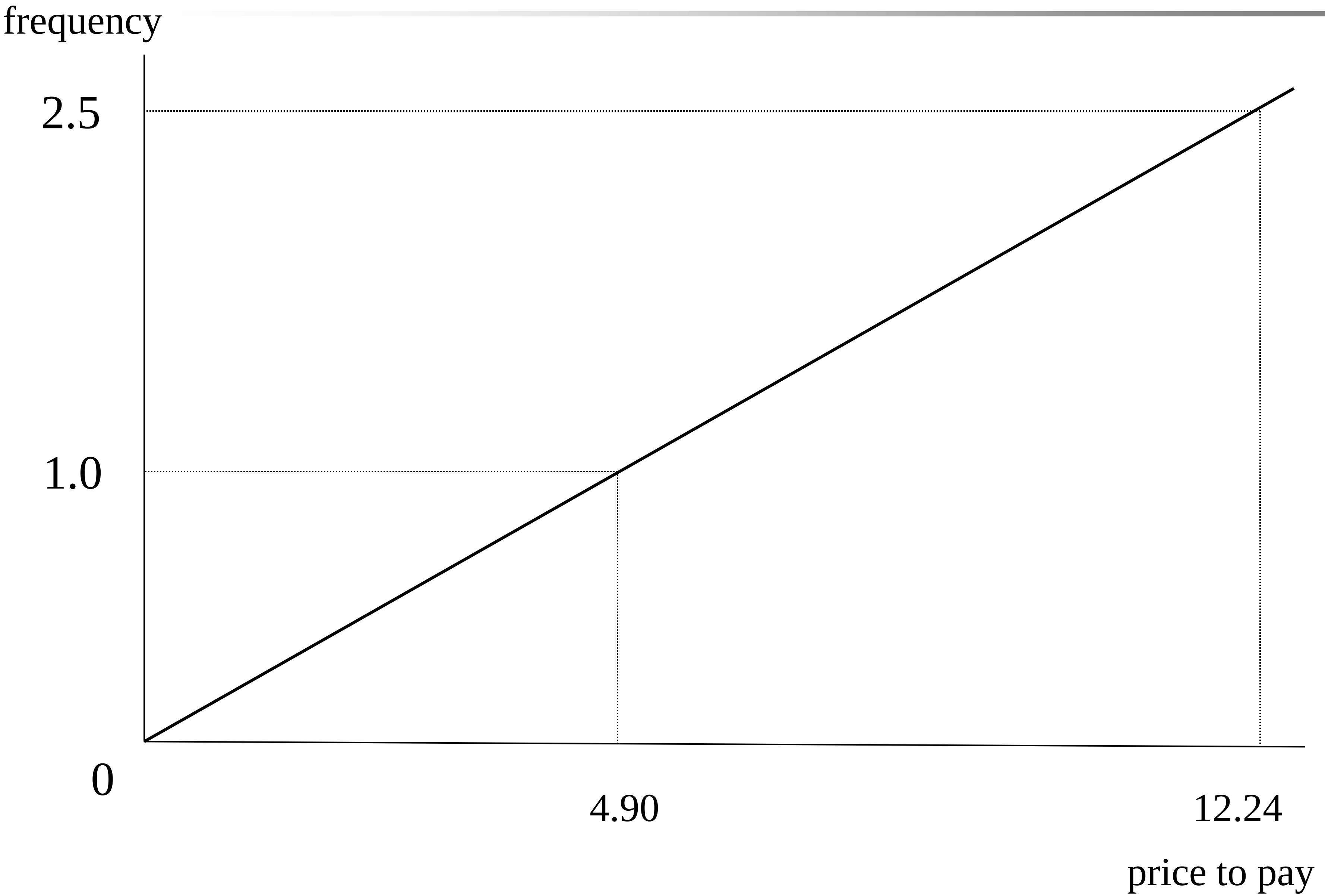
2. Define your compute resources

Resource	Amount	Cost (**CHF**)
CPU/ GHz (1 core = 2.5GHz)	1.701	8.33
RAM/ GB	1	0.00
SSD Storage/ GB	50	0.00
Magnetic Storage/ GB	0	0.00
Data Transfer/ GB	5120	0.00
Static IPs	0	0.00
	Total cost	**8.33**

Calculate

…even if you pay 0.01CHF extra for 1 MHz more!

Price vs CPU frequency chosen (per time unit)



The problem

Say that you want to use 4 CPUs for your application (to run in parallel).

Are you going to choose 4 CPUs at, say, 2GHz?

or: 1 at 2.2 GHz, 2 at 2 GHz, 1 at 1.8 GHz?

or: 1 at 2.4 GHz, 1 at 2.2 GHz, 1 at 1.8 GHz, 1 at 1.6 GHz?

and the list goes on with the number of combinations that **cost the same per time unit**…

…but could some configurations lead to faster execution time, which will make them cheaper?

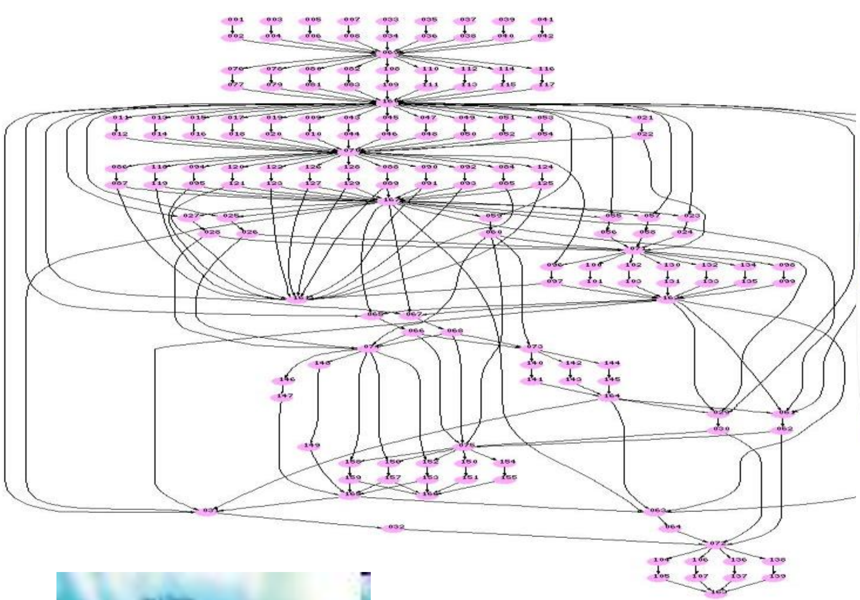
Setting the scene

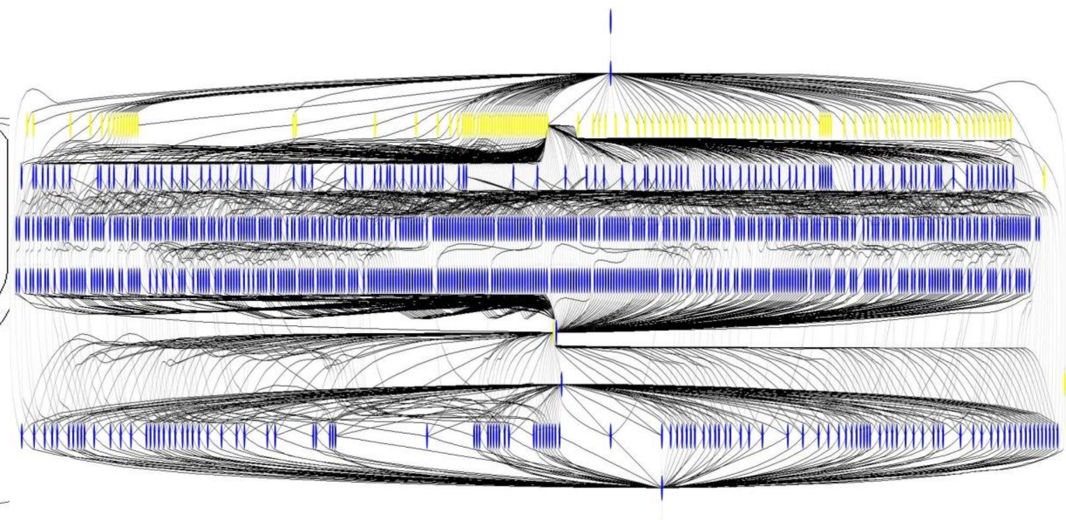
Configurations with the same average cost may lead to different execution times: how can we find the fastest/cheapest? (or at least avoid too expensive?)

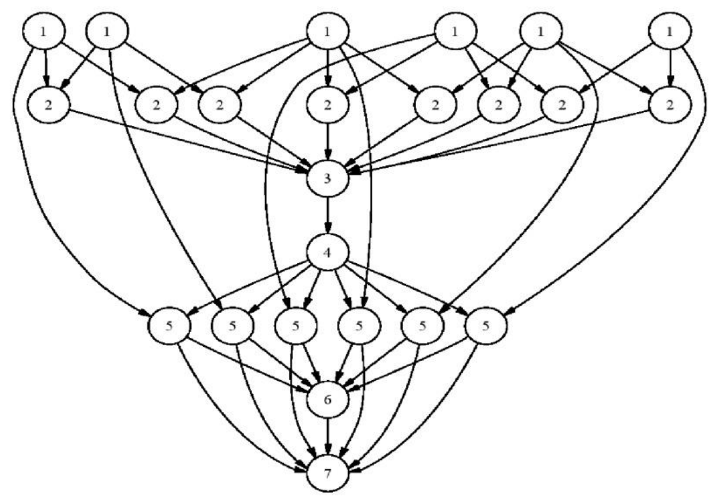
- **Work**
  - Scientific workflows (essentially Directed Acyclic Graphs)
    - DAG (nodes: computation, edges: communication)
- **Resources**
  - Cloud Computing resources at different frequencies
- **Objective**
  - Complete execution; strike a balance between cost and performance (**find the Pareto front**). Cost has two aspects: client and provider. The former are interested in monetary costs, the latter are interested in the cost running the infrastructure (energy)

Scientific Workflows

Many interesting scientific applications can be represented by DAGs

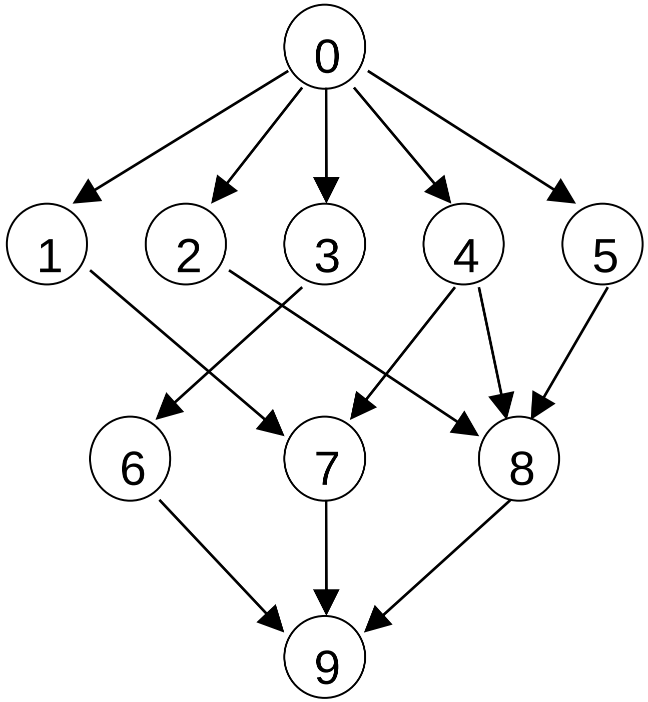


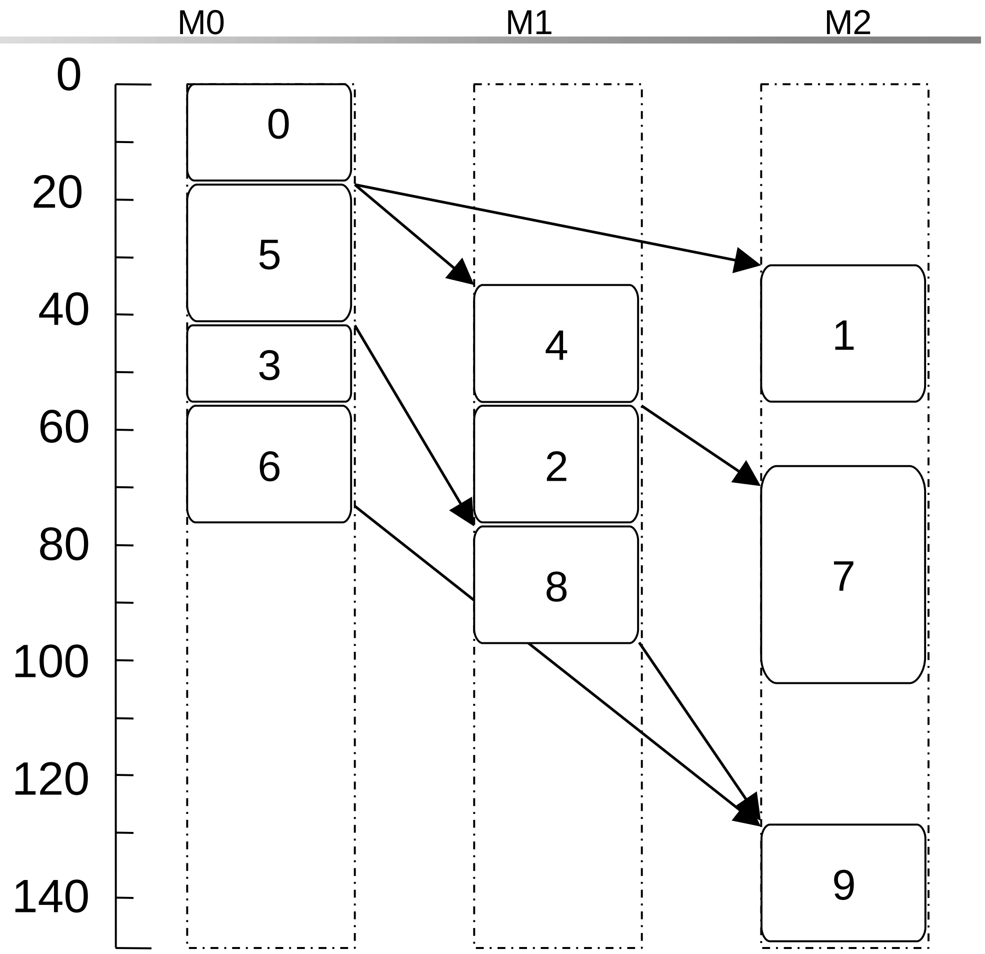




I. Taylor, E. Deelman, D. Gannon: Workflows for e-Science. Springer, 2007

A DAG, a schedule, and an old idea





Many (but not all) tasks can delay without an impact on overall execution time (e.g., 1, 7, 8, but also 2, etc) (slack/spare time)

R.Sakellariou, H.Zhao. A low-cost rescheduling policy for efficient mapping of workflows on grid systems. *Scientific Programming*, 12(4), December 2004, pp. 253-262.

Spare time & Slack

- Depend on:
  - Structure of the DAG
  - Number of resources
  - Schedule (how we map tasks onto resources)
  - Communication vs Computation
- However, unless we choose very few resources or there is an abuntance of parallelism, we'll have some spare time and slack
- **<u>They provide interesting opportunities!</u>**

  (e.g., use them to slow down without affecting overall completion time)

Furthermore…

- Every task is not affected in the same way if we change CPU frequency:
  – CPU-intensive tasks will be affected most
  – Data-intensive tasks will be affected less
- This is captured by the following formula, which gives runtime at a given frequency *f*:

$$runtime = (1 + \beta(f_{max} / f - 1)) \times runtime_{fmax}$$

where β is the CPU boundedness of a task (0 to 1)

(from: Etinski, Corbalan, Labarta & Valero, JPDC 2012)

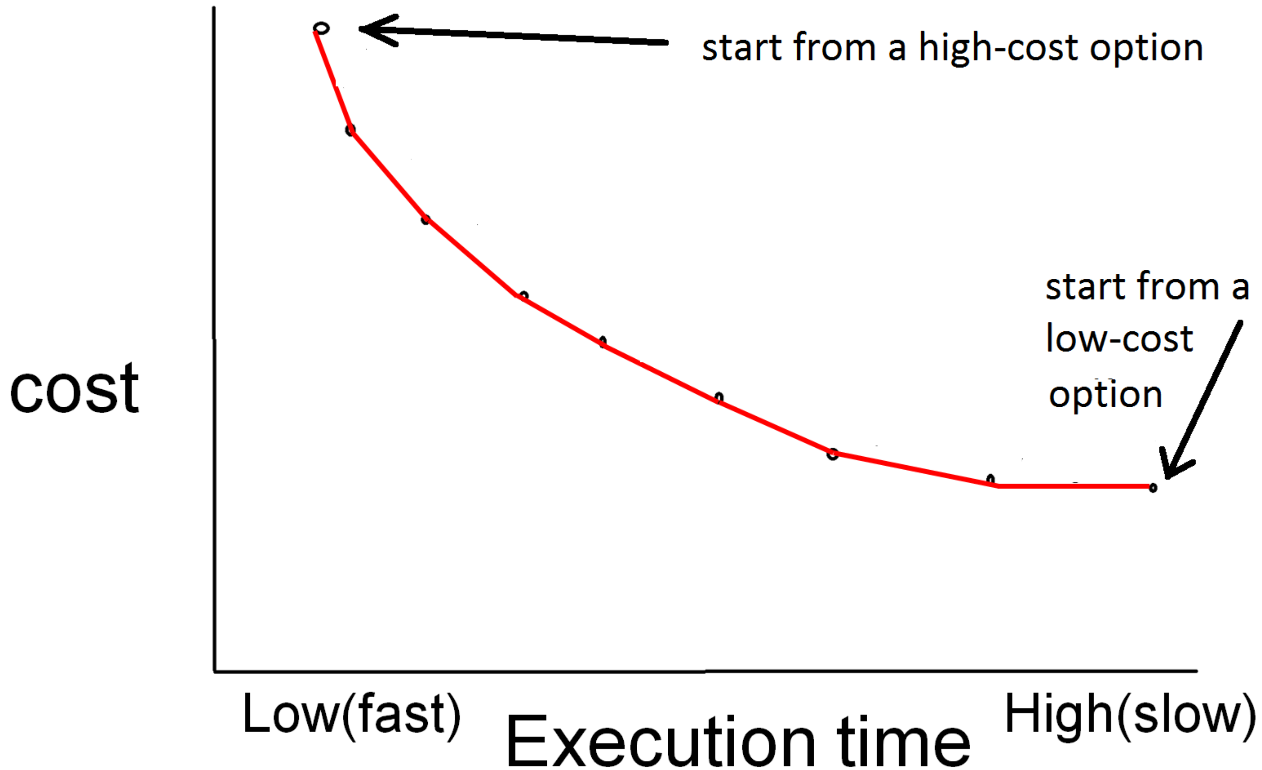
This suggests that a mix of rather fast and rather slow CPUs may be cheaper/faster than using all CPUs running at the same speed

(assuming the same average frequency overall)

The idea

- Reduce or increase CPU frequencies iteratively
  - Using the next available frequency in each iteration
  - So that cost is reduced and deadline is met

(trying iteratively to approximate the Pareto front)



Cost-based Stepwise Frequency Selection starting at Max Frequency

CSFS-Max

- Stage 1. Start with a schedule at maximum frequency.
- Stage 2. Using the next available frequency:
  - 1. Select the resources with cost savings for the new frequency.
  - 2. Update the plan using the chosen frequency for these resources.
  - 3. Accept the new plan if costs savings and continue with the same procedure (go to 1).
  - 4. Otherwise terminate.

I. Pietri, R.Sakellariou. Cost-Efficient Provisioning of Cloud Resources priced by CPU Frequency. UCC 2014 (best poster award).

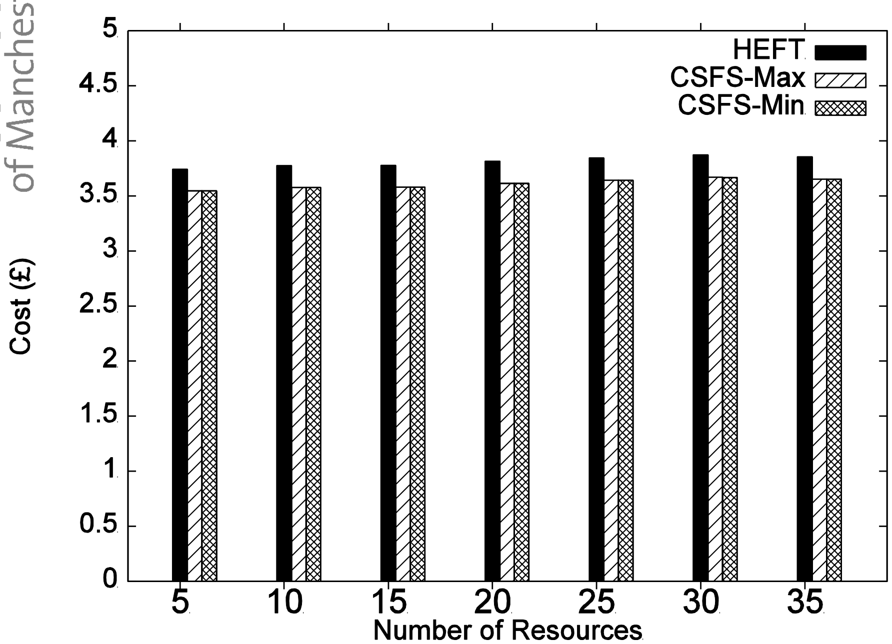
Cost-based Stepwise Frequency Selection starting at Min Frequency

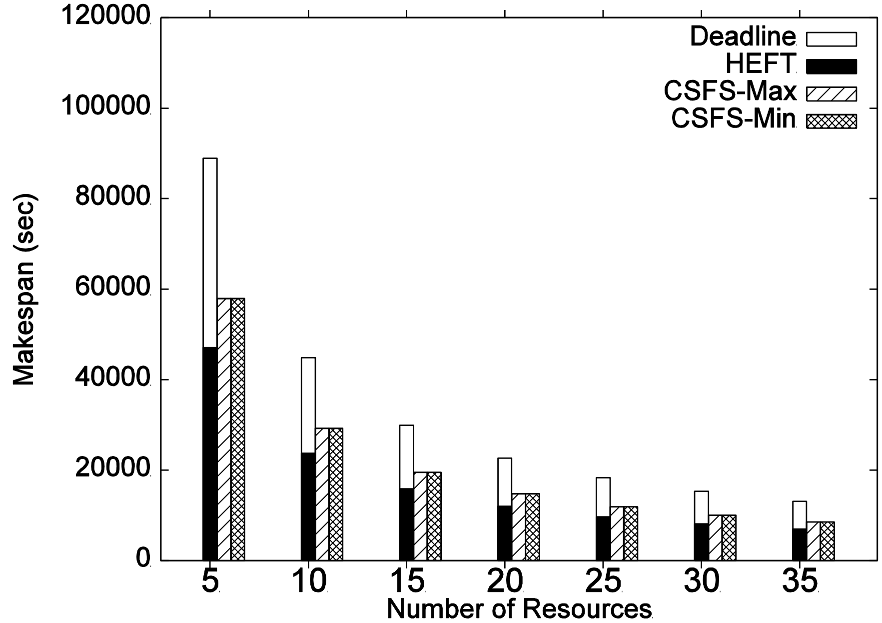
CSFS-Min

- Stage 1. Start with a schedule at minimum frequency.
- Stage 2. Using the next available frequency:
  - 1. Select the resources with makespan savings for the new frequency.
  - 2. Update the plan using the chosen frequency for these resources.
  - 3. Reject the new plan if cost is increased and deadline is already met and terminate
  - 4. Otherwise accept plan.
  - 5. Generate HEFT plan at maximum frequency if deadline cannot be met

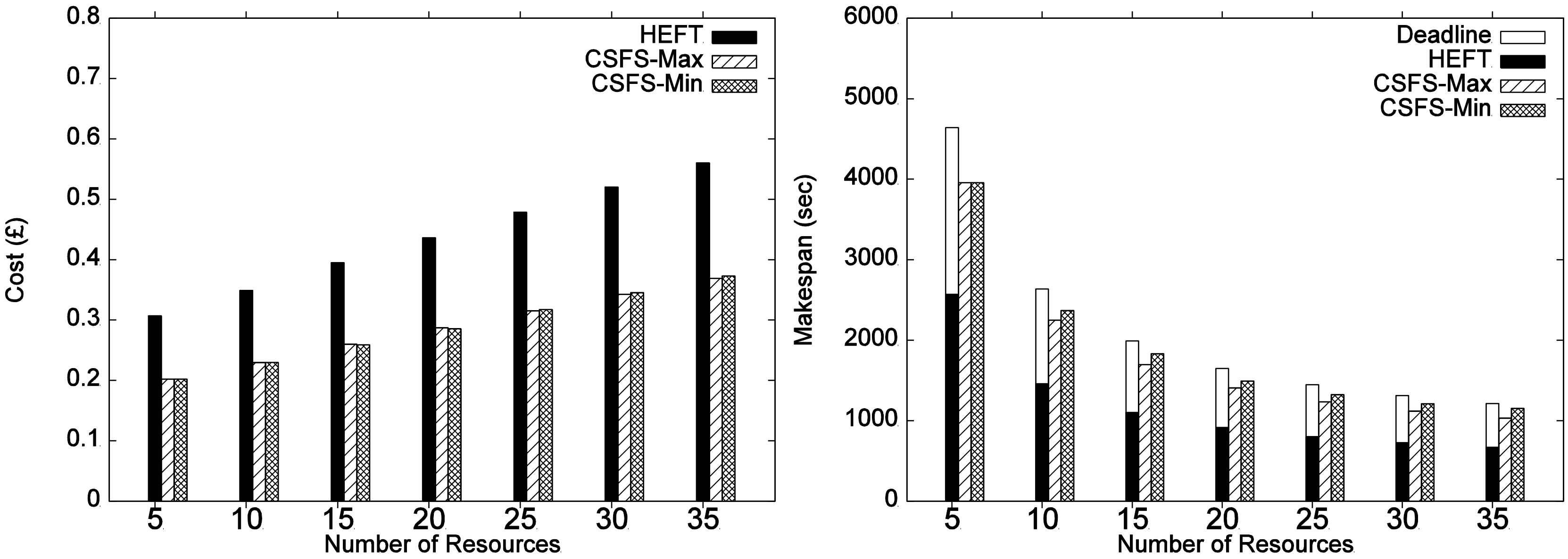
I. Pietri, R.Sakellariou. Cost-Efficient CPU Provisioning for Scientific Workflows on the Cloud. GECON 2015.

LIGO Under Linear Pricing





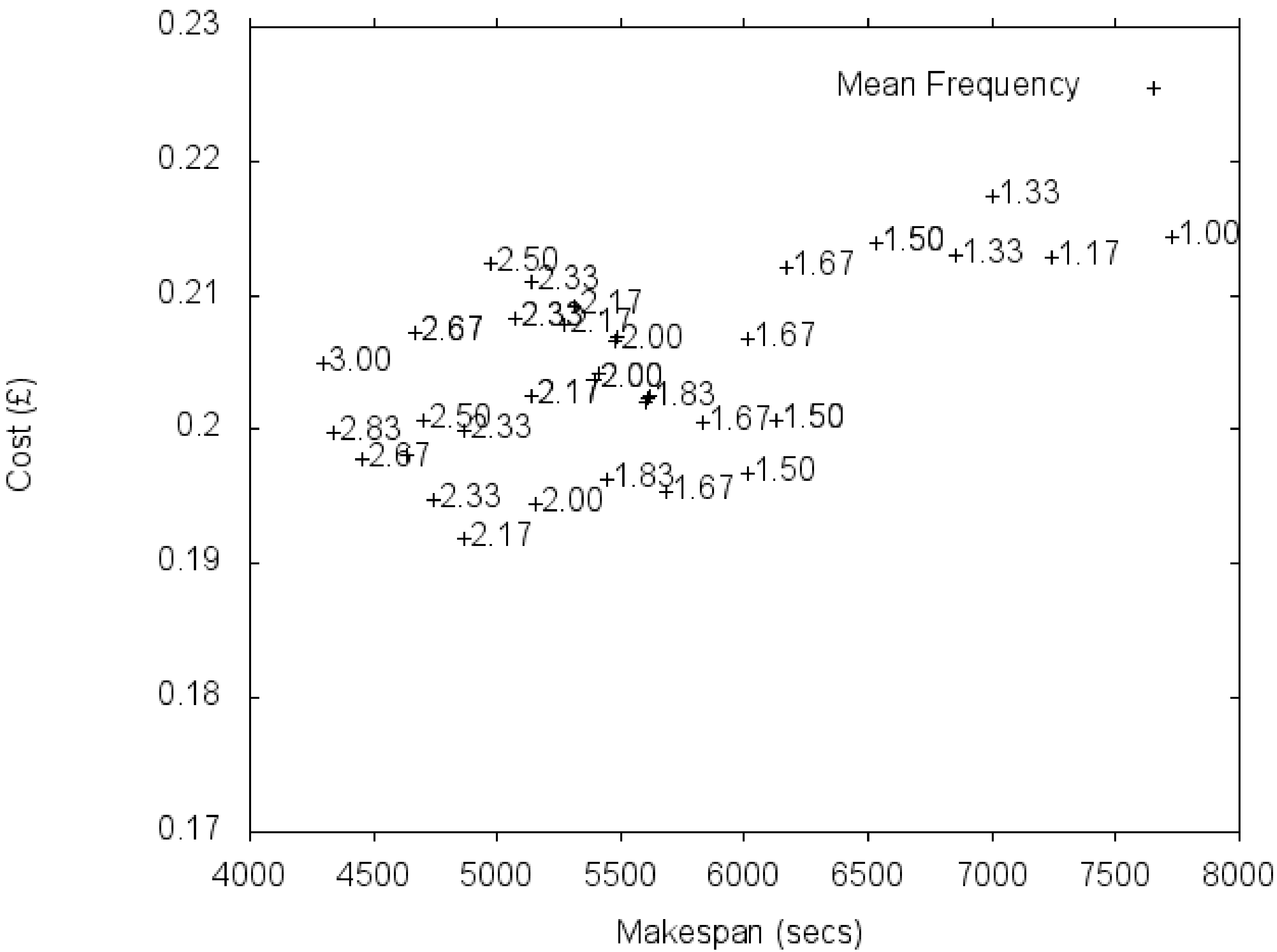
Montage under Superlinear Pricing



Some observations

- Solutions go for local optima – still they find some good mix of suitable (heterogeneous) resources.
- Approach relies on performance modelling
- Seems a good strategy to try both algorithms
- Data-intensive workflows appear to give more interesting results
- Starting from minimum frequency doesn’t perform as well – other starting points were even more problematic; other optimization approaches at UCC2015.
- Full results in the GECON2015 paper (also trying different pricing models)

Plotting some solutions with 3 resources

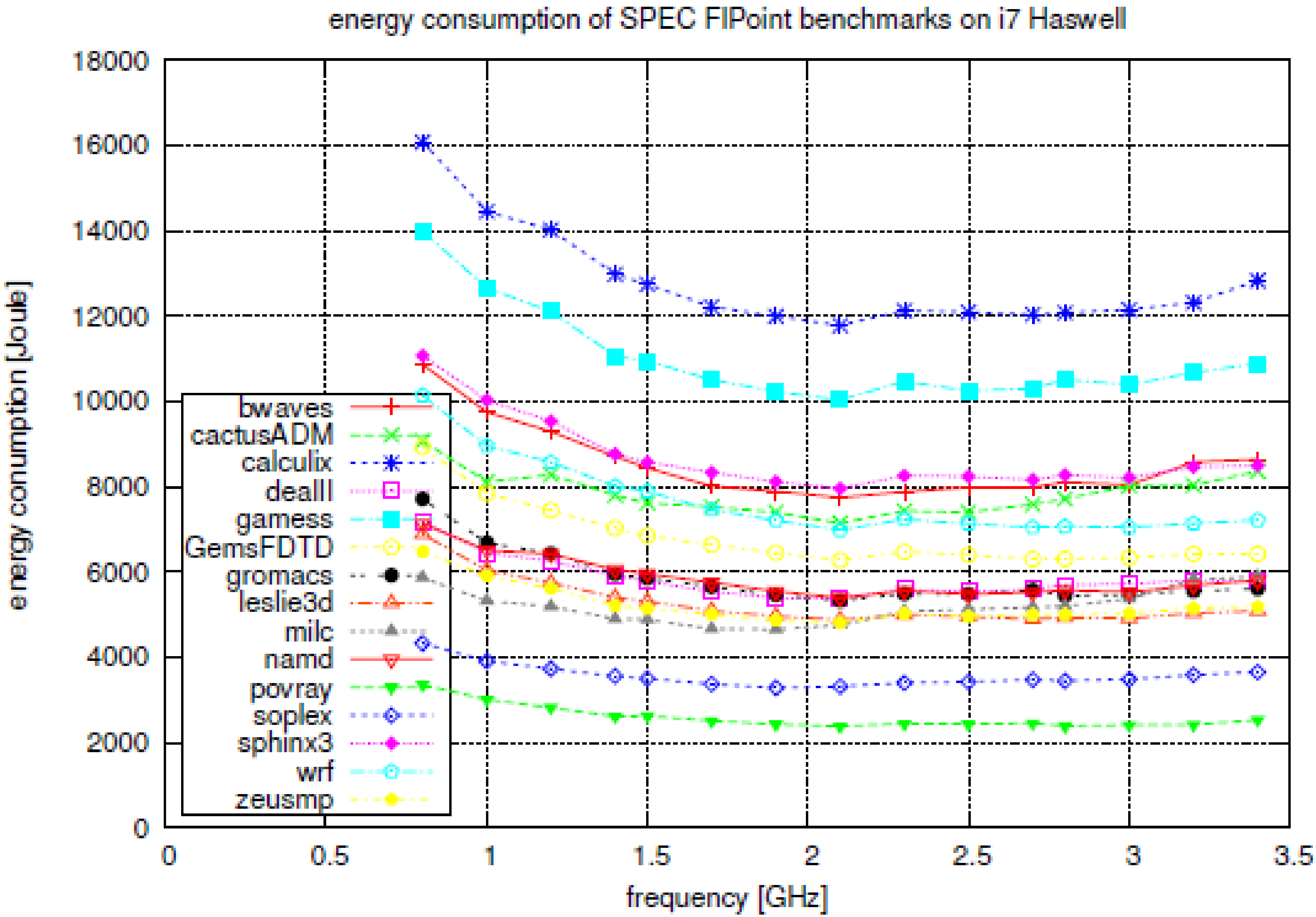


Energy vs Performance

- Cost was easy to model (based on pricing model):
  - Essentially taken as charged by provider
- Instead of cost, we can have energy
  - Difficult to model energy
- Reducing frequency requires less power but may lead to longer execution times (hence will consume more energy).

Thanks to Thomas Rauber

(presentation at the 9th Scheduling for large-scale systems workshop, Lyon, July 2014)



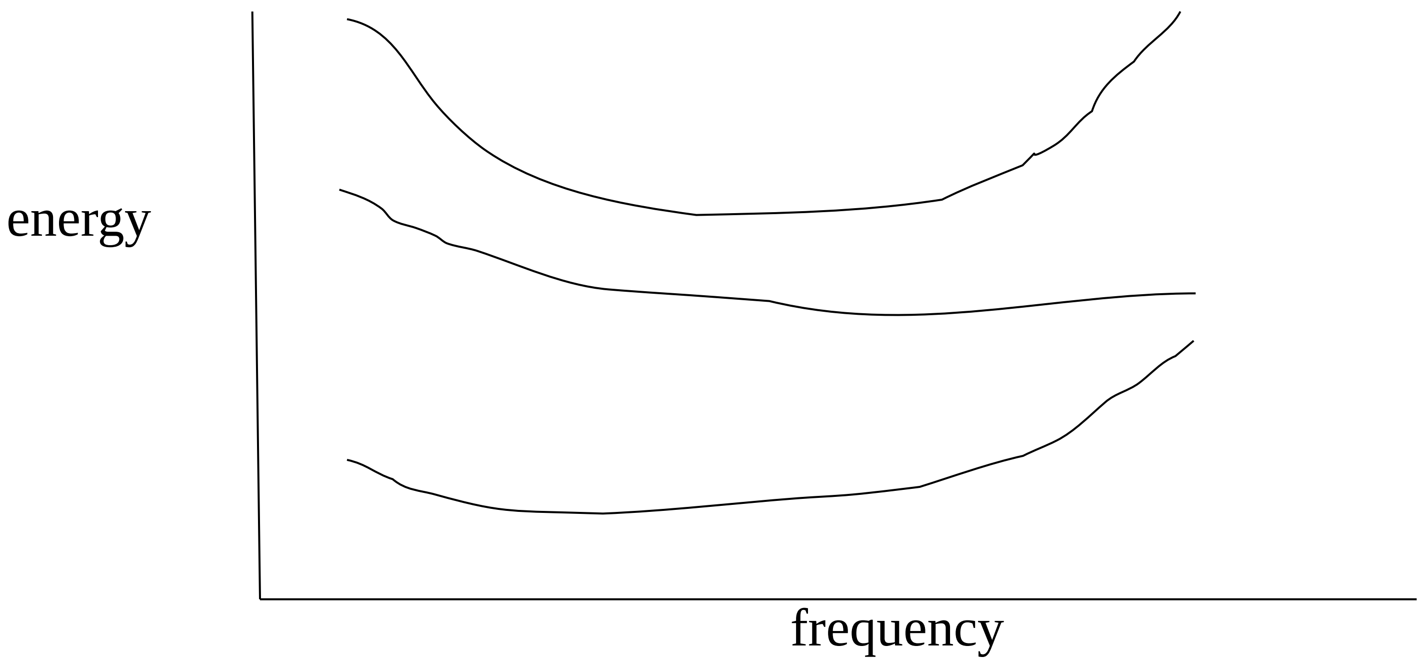
The idea – an iterative approach

- Assuming that we need to meet a deadline and minimize energy:
  - 1. Start with a schedule running at highest frequency (can be easily obtained with HEFT, etc)
  - 2. Identify the most profitable in terms of energy reduction tasks (beyond some threshold)
  - 3. Lower to the next available frequency
  - 4. Assess the impact to the whole workflow (DAG)
  - 5. Go to 2 as long as there is overall energy reduction
  - 6. Cleanup and finish.

  (Energy-aware stepwise frequency scaling – ESFS)

The intuition

- Reduce frequency by one step at a time: (i) trying to make sure that what may be the local optimum for every task (in the U-curve) is not exceeded, and (ii) assessing the overall energy consumption for the workflow.



Evaluation

- Baseline algorithms
  - EES (from CCGRID12)
  - HEFT
- Processor characteristics
  - P_{base}=152W
  - P_{dif}=15.39W
  - P_{idle}=60%P_{fmax}
  - Threshold: 0.01%
- Data from 3 workflows, 100 tasks each
  - LIGO
  - SIPHT
  - Montage

Full results in:

I. Pietri, R. Sakellariou. "Energy-Aware Workflow Scheduling Using Frequency Scaling". ICPP Workshops (PASA), 2014.

Discussion of results

- Different workflows exhibit a different behaviour
- The iterative approach can produce energy savings without missing a deadline
- Energy savings are rather small.
- The outcome is sensitive to the parameters used in the energy model. Some may be difficult to estimate / others change depending on the processor, etc.
- CPU energy is only a fraction of overall energy
- Simulation results need to be verified with real experiments

I. Pietri, R. Sakellariou. “Energy-Aware Workflow Scheduling Using Frequency Scaling”. ICPP Workshops (PASA), 2014.

Even more trade-offs

- Combining frequency scaling with VM migration/consolidation may lead to useful savings (an energy-revenue trade-off)
- Perceived-performance pricing

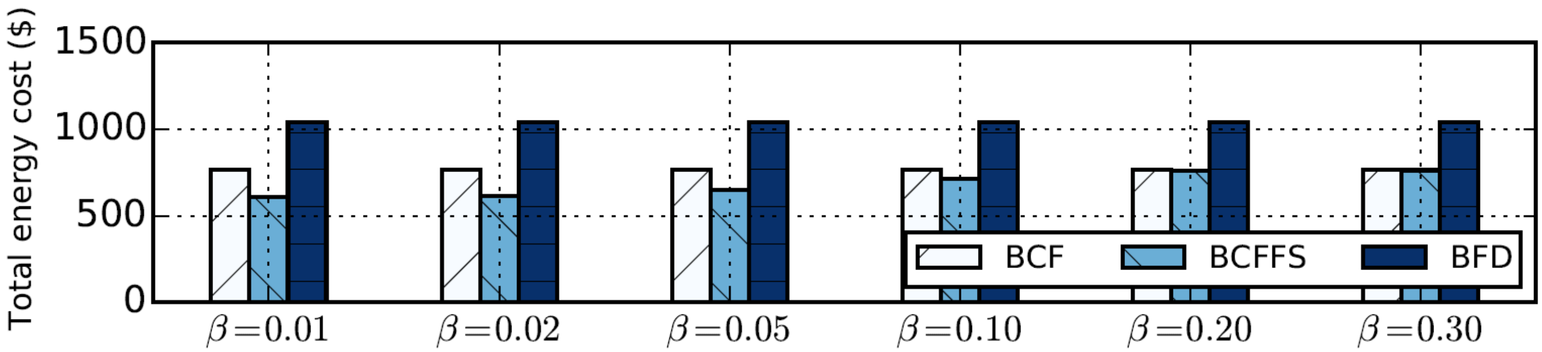


Fig. 11. Energy cost savings for VMs with different fixed CPU-boundedness.

Drazen Lucanin, Ilia Pietri, Ivona Brandic and Rizos Sakellariou. A Cloud Controller for Performance-Based Pricing. In *IEEE Cloud 2015*.

Lots of excitement for the future…

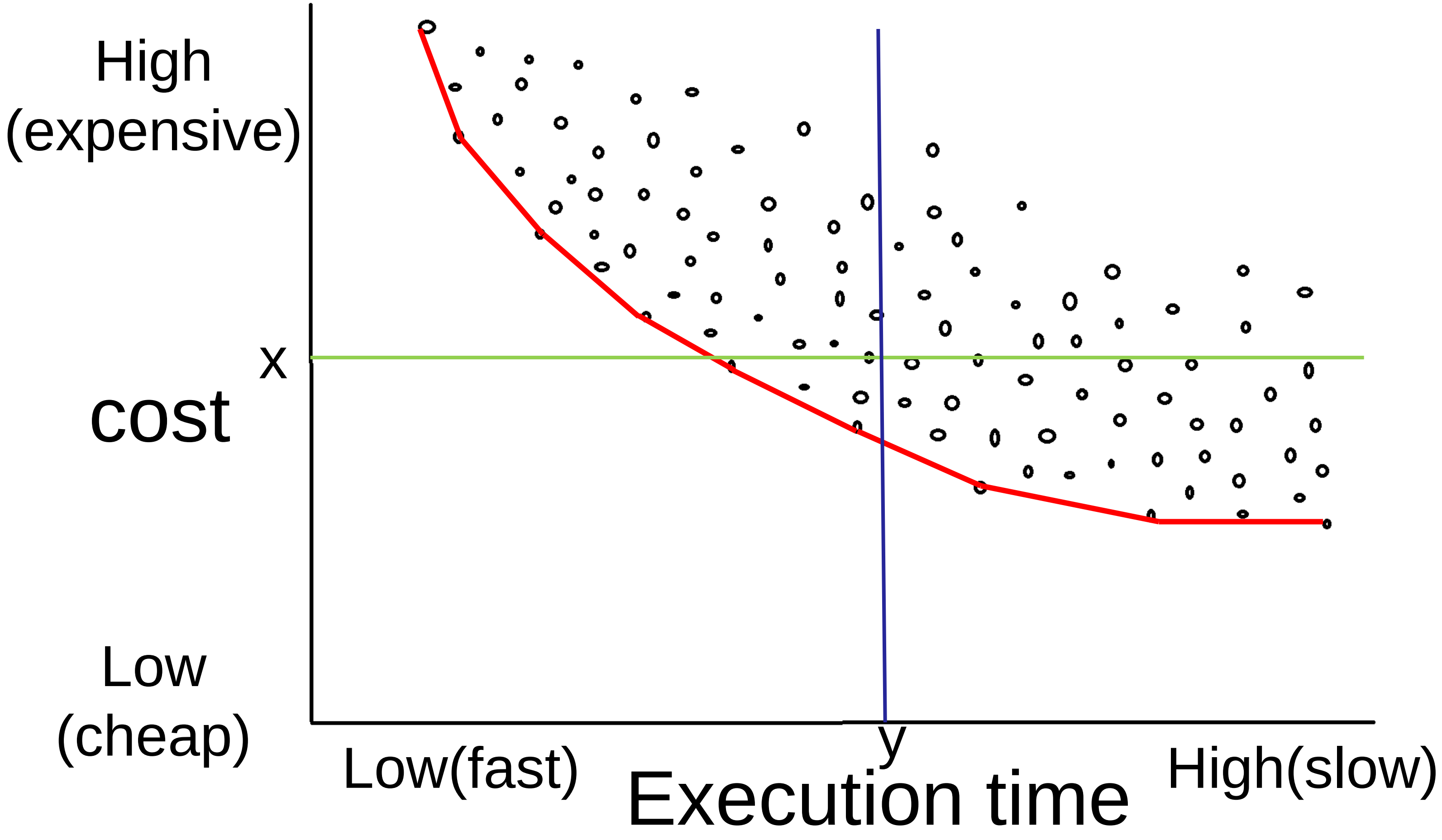
Trying to understand and appreciate all the trade-offs is a tremendous task.

However, the growing heterogeneity of modern parallel platforms and the plethora of different configurations means that we need to deal with many questions involving these trade-offs, the simplest form of which could be:

- Shall I choose 50 CPUs at 2GHz or 25 at 2.5GHz and 25 at 1.5GHz?

We need to understand our search space

- Fastest but doesn't cost more than x (green line)
- Cheapest but doesn't run in more than y (blue line)



Conclusion

- With the growing heterogeneity we need to look more carefully into the cost we pay to achieve a certain level of performance:
  - Many suboptimal solutions, but still at high-cost
  - Pareto front – multi-objective optimization
- Cost
  - Cost could be: energy, number of failures, memory (new trend 3D), etc… There are various trade-offs between them and performance.
  - Rather easy(?) to deal with a pricing model provided by somebody else.
- We need:
  - Extensive Experimentation to understand different trade-offs
  - Good Performance Models / (or at least some Rules of Thumb)
  - Optimization Techniques (multi-criteria optimization is challenging)
  - Parallelization approaches that take into account these trade-offs.
- Welcome to the growing complexity, but "*life is filled with trade-offs*"