MANCHESTER 1824

The University of Manchester

Parallel Computing in the era of the Cloud and Heterogeneous Computing

Rizos Sakellariou

with thanks to Ilia Pietri and many other students and collaborators over the years

What this talk is about

- Parallel Computing
  - Breaking a problem into sub-problems that can be solved concurrently
- The Cloud
  - On-demand computing resources
- Heterogeneous Computing
  - Systems consist of different types of processors / cores

Alternative title:

"Computational/Cloud support in the era of data-intensive applications"

What this talk is about

- Parallel Computing
  - Breaking a problem into sub-problems that can be solved concurrently
- Big Data
  - Dealing with the analysis of large data sets
- Industry 4.0
  - Take advantage of the abundance of data to make ‘smart’ decisions in industrial processes

Alternative title:

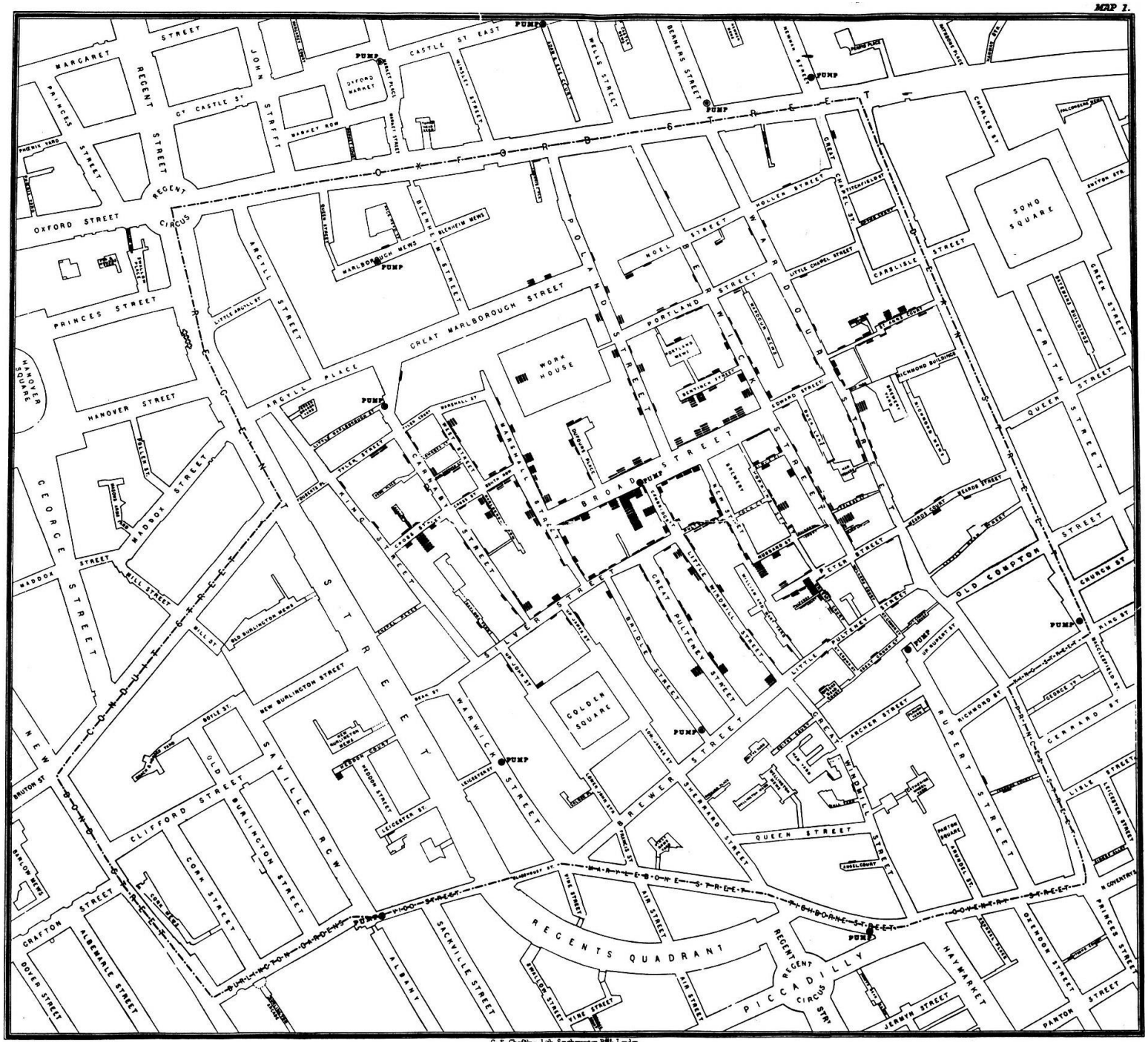
“Parallel Computing in the era of Big Data and Industry 4.0”

"What's in a name?

That which we call a rose by any other name would smell as sweet"

William Shakespeare

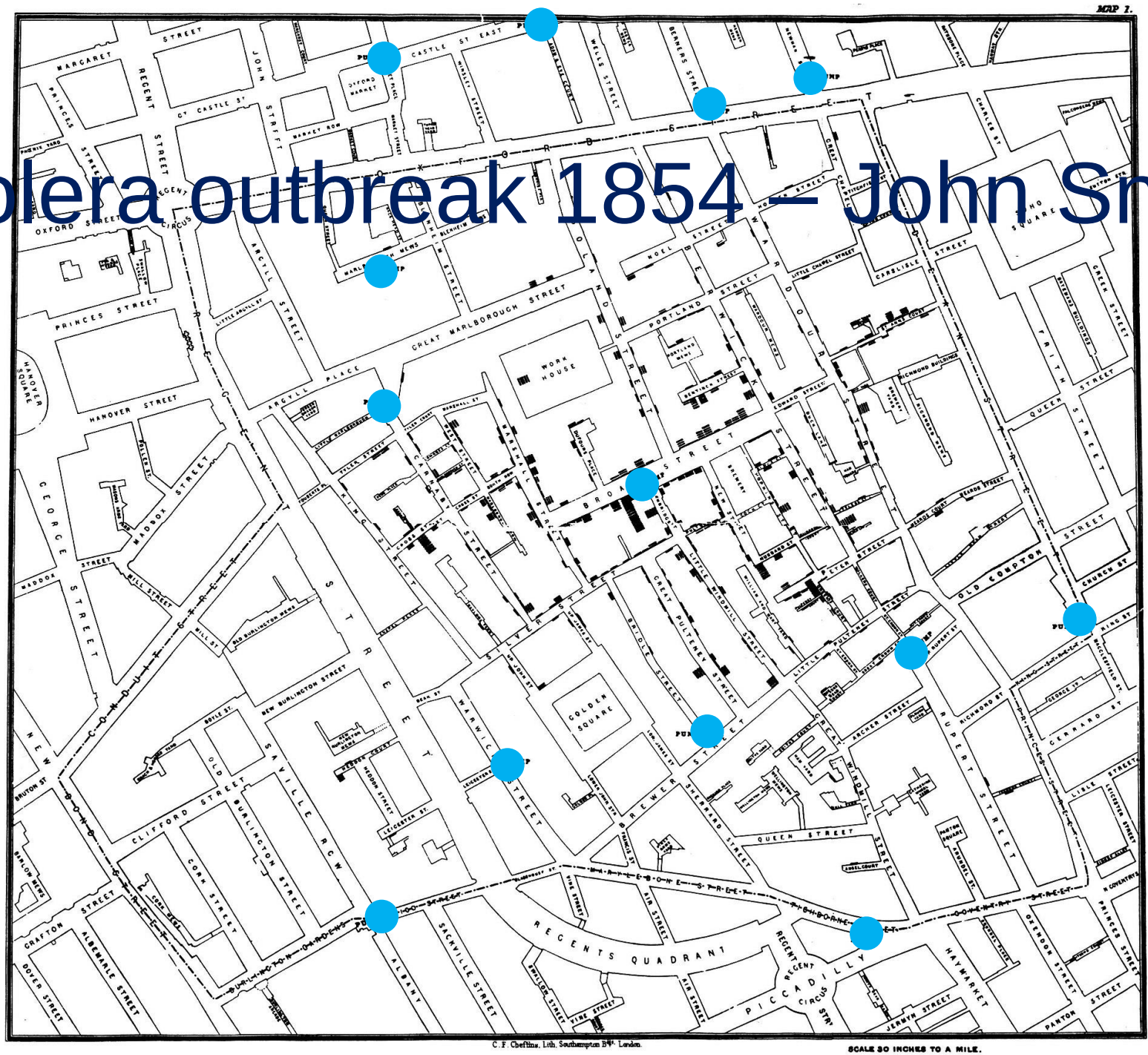
MAP 1.



C.F. Cheffins, Lith. Southampton Bldgs. London.

SCALE 30 INCHES TO A MILE.

Cholera outbreak 1854 – John Snow



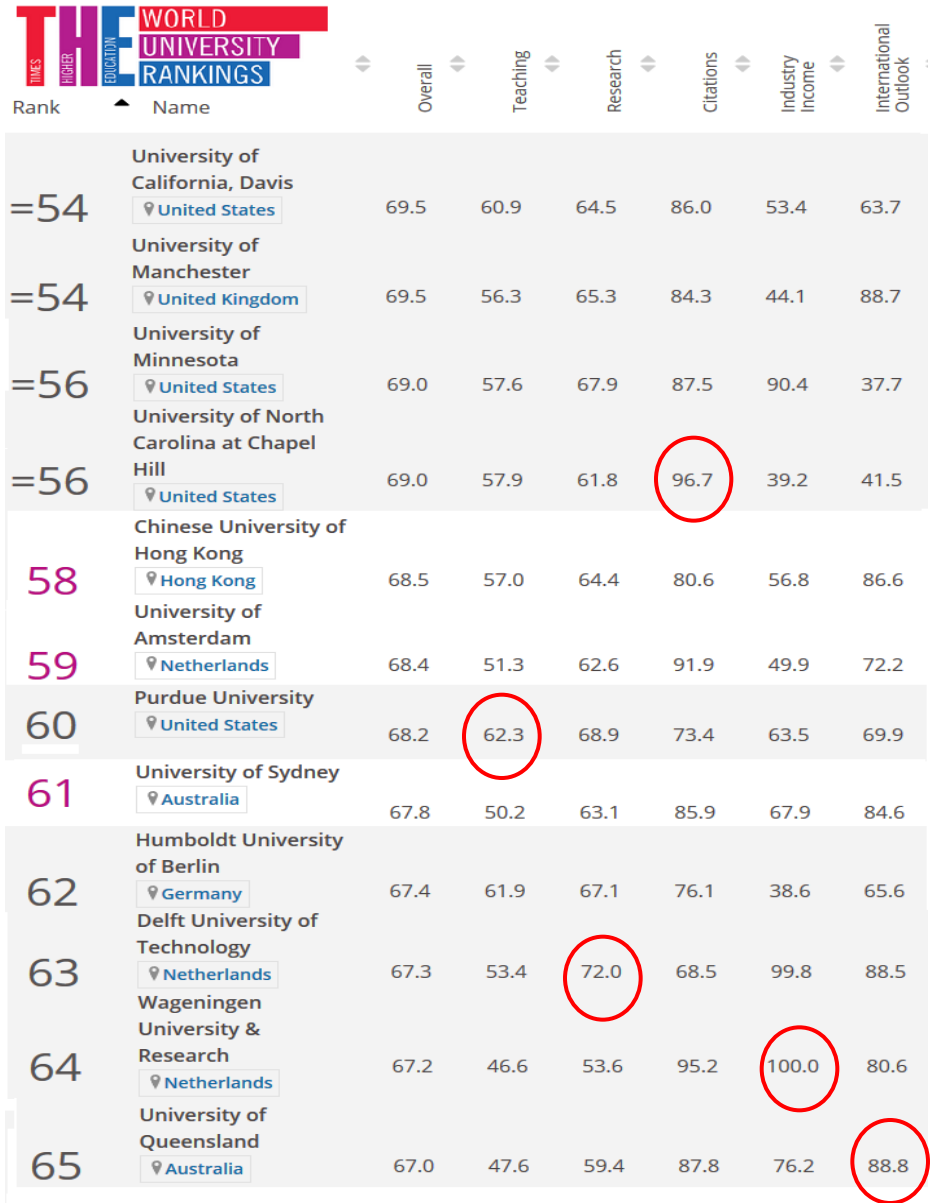
Today we call it Data Science…

"Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and **systems** to extract knowledge and insights from **data in various forms**, both structured and unstructured, similar to data mining."

(source: Wikipedia)

What this talk is about

(another way of viewing it)



THE WORLD UNIVERSITY RANKINGS

Rank	Name	Overall	Teaching	Research	Citations	Industry Income	International Outlook
=54	University of California, Davis — United States	69.5	60.9	64.5	86.0	53.4	63.7
=54	University of Manchester — United Kingdom	69.5	56.3	65.3	84.3	44.1	88.7
=56	University of Minnesota — United States	69.0	57.6	67.9	87.5	90.4	37.7
=56	University of North Carolina at Chapel Hill — United States	69.0	57.9	61.8	96.7	39.2	41.5
58	Chinese University of Hong Kong — Hong Kong	68.5	57.0	64.4	80.6	56.8	86.6
59	University of Amsterdam — Netherlands	68.4	51.3	62.6	91.9	49.9	72.2
60	Purdue University — United States	68.2	62.3	68.9	73.4	63.5	69.9
61	University of Sydney — Australia	67.8	50.2	63.1	85.9	67.9	84.6
62	Humboldt University of Berlin — Germany	67.4	61.9	67.1	76.1	38.6	65.6
63	Delft University of Technology — Netherlands	67.3	53.4	72.0	68.5	99.8	88.5
64	Wageningen University & Research — Netherlands	67.2	46.6	53.6	95.2	100.0	80.6
65	University of Queensland — Australia	67.0	47.6	59.4	87.8	76.2	88.8
1001+	Bogor Agricultural University — Indonesia	9.2–15.5	18.4	8.7	8.9	68.2	36.6

- *There is rarely a single way to define ‘best’.*
- *The answer often depends on what aspect we are interested in and how we weigh different aspects!*
- *But, sometimes, there are relatively ‘bad’ choices in every possible aspect.*
- *All the above (and more…) in the context of Cloud resource provisioning for parallel applications.*

Some background…

Parallel Computing 101 (1)

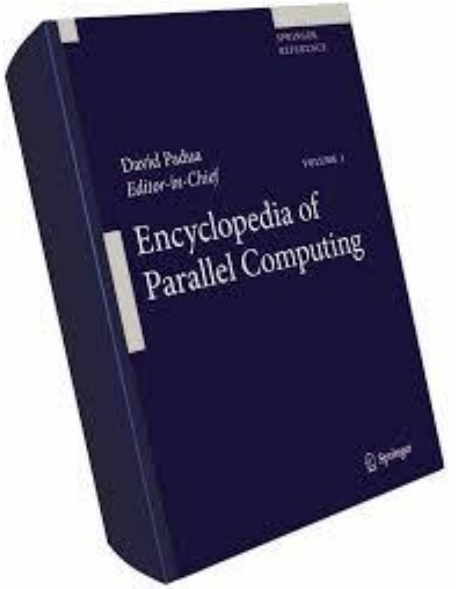
- Parallel Computing is about breaking a problem into sub-problems that can be solved concurrently.
- The concept existed even before digital computers (L. F. Richardson, “Weather Prediction by Numerical Process”, 1922)
- In early years it was rather specialized with the first parallel computers (‘supercomputers’) based on special-purpose architectures and processors trying to cope for the needs of a few computationally-demanding applications.

Parallel Computing 101 (2)

- Since the 1990s it was believed that “advances in High-End Computing find their way to the desktop” (PITAC report, 1999).
- Developments in microprocessors allowed the use of commodity parts. Developments in network technologies gave rise to the notion of **Grid Computing** from which **Utility and Cloud Computing** evolved.
- Today we live the ‘*triumph of parallel computing*’ (Schreiber, JPDC2014) as it has become mainstream: multicore, GPUs, data centers, MapReduce, etc… and lots of applications that can benefit!

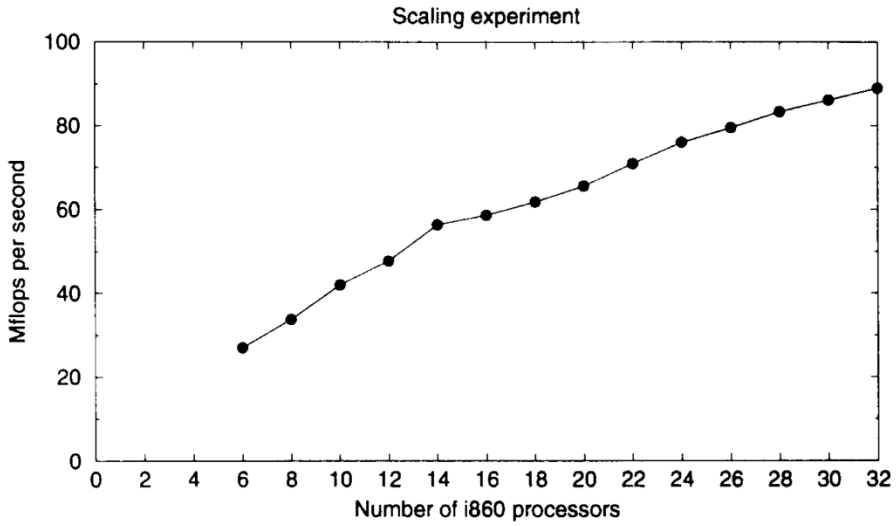
“Parallel computing is generally concerned with reducing the amount of time necessary to perform a computational task”

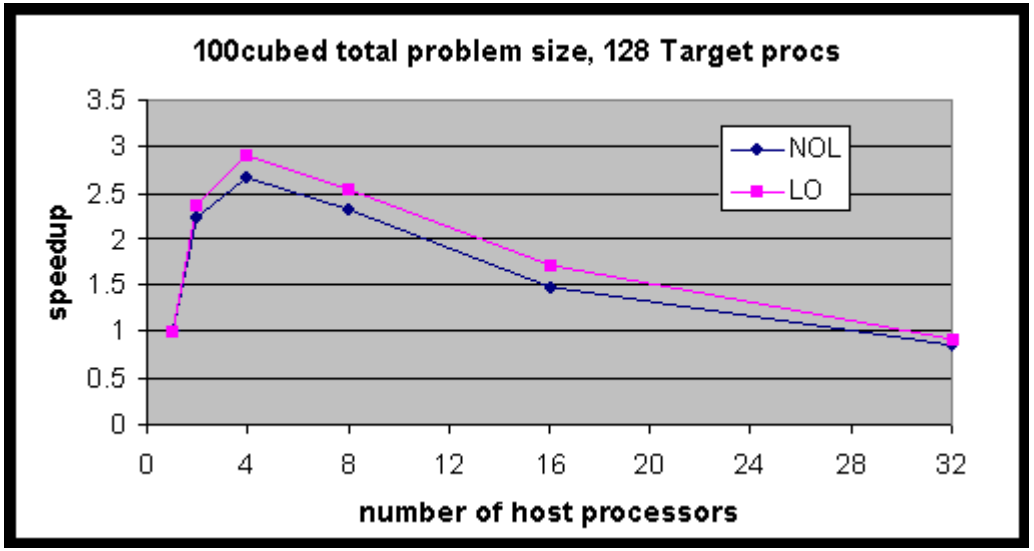
Encyclopedia of Parallel Computing, page 1125
(D. Padua EiC, published in 2011)

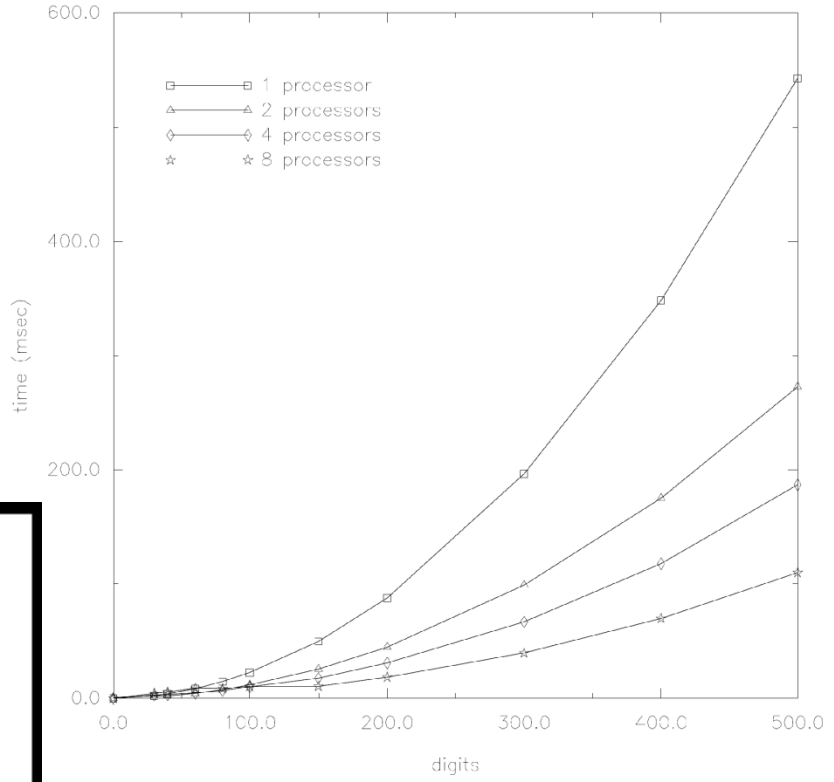


Such a reduction in time would be presented in different ways…

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



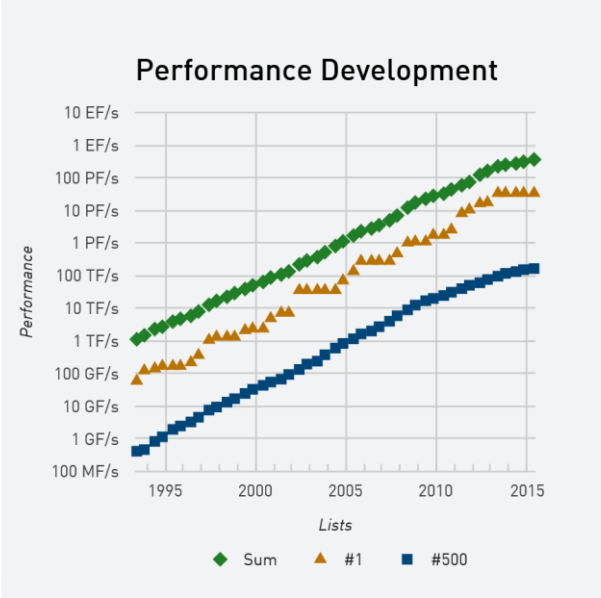




…and would drive our understanding

- Understand the inherent limitations of applications when parallelizing them:
  - Amdahl's law (or total work vs critical path)
- Classify the performance of the most powerful computers:
  - Top 500

  http://www.top500.org/



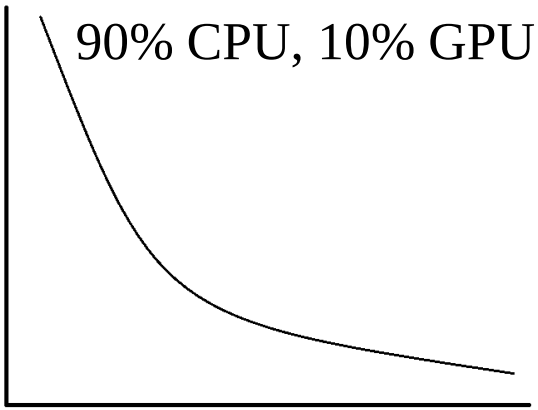
For many years, parallel computing had a single way to define 'best': minimize execution time

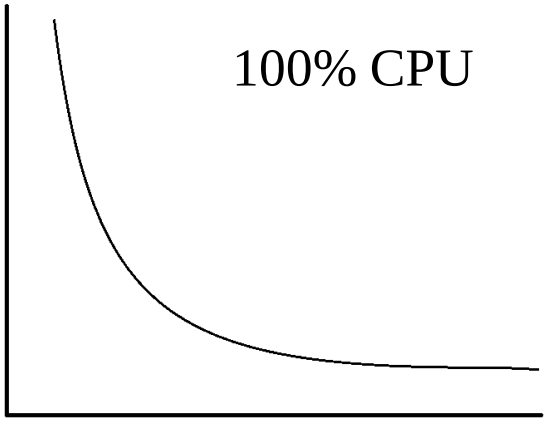
…but at the same time there is a cost to pay – and we cannot keep ignoring it!

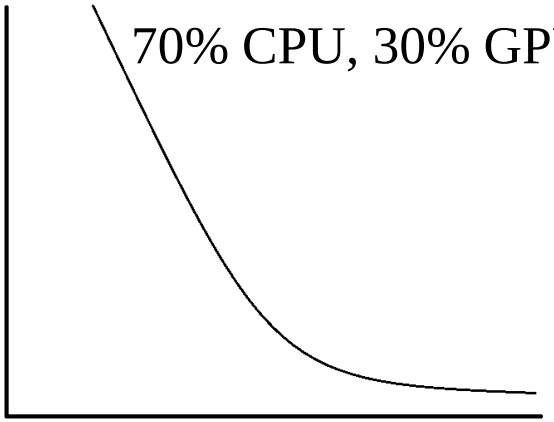
- Modern parallel machines/platforms are becoming too expensive to run.
  - Energy
- Heterogeneity increases the complexity of parallelization (and the search space)
  - Not simply 'as many processors' any more
  - There is a need to differentiate between more costly and less costly solutions

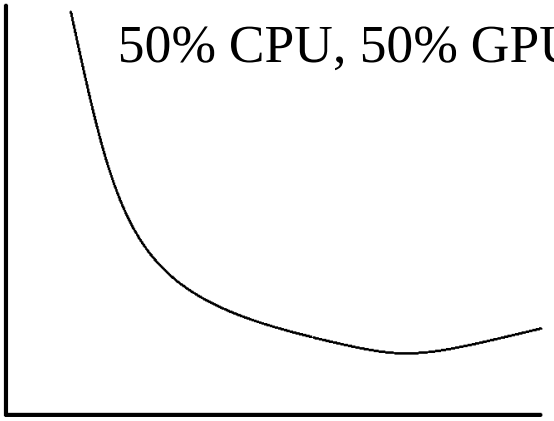
E.g., choose from fast/slow CPUs & GPUs (x% fast CPUs, y% slow CPUs, z% GPUs)

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



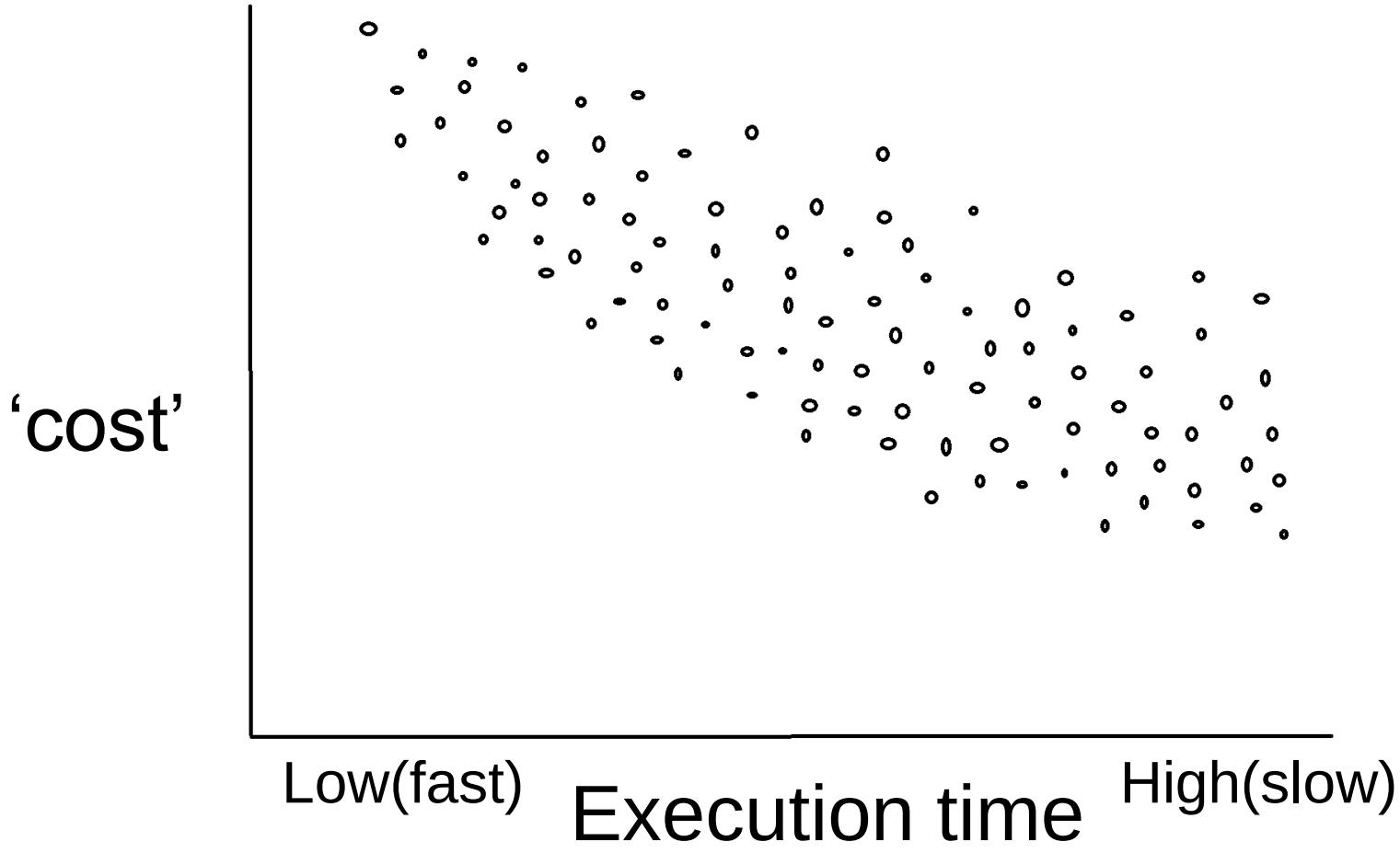






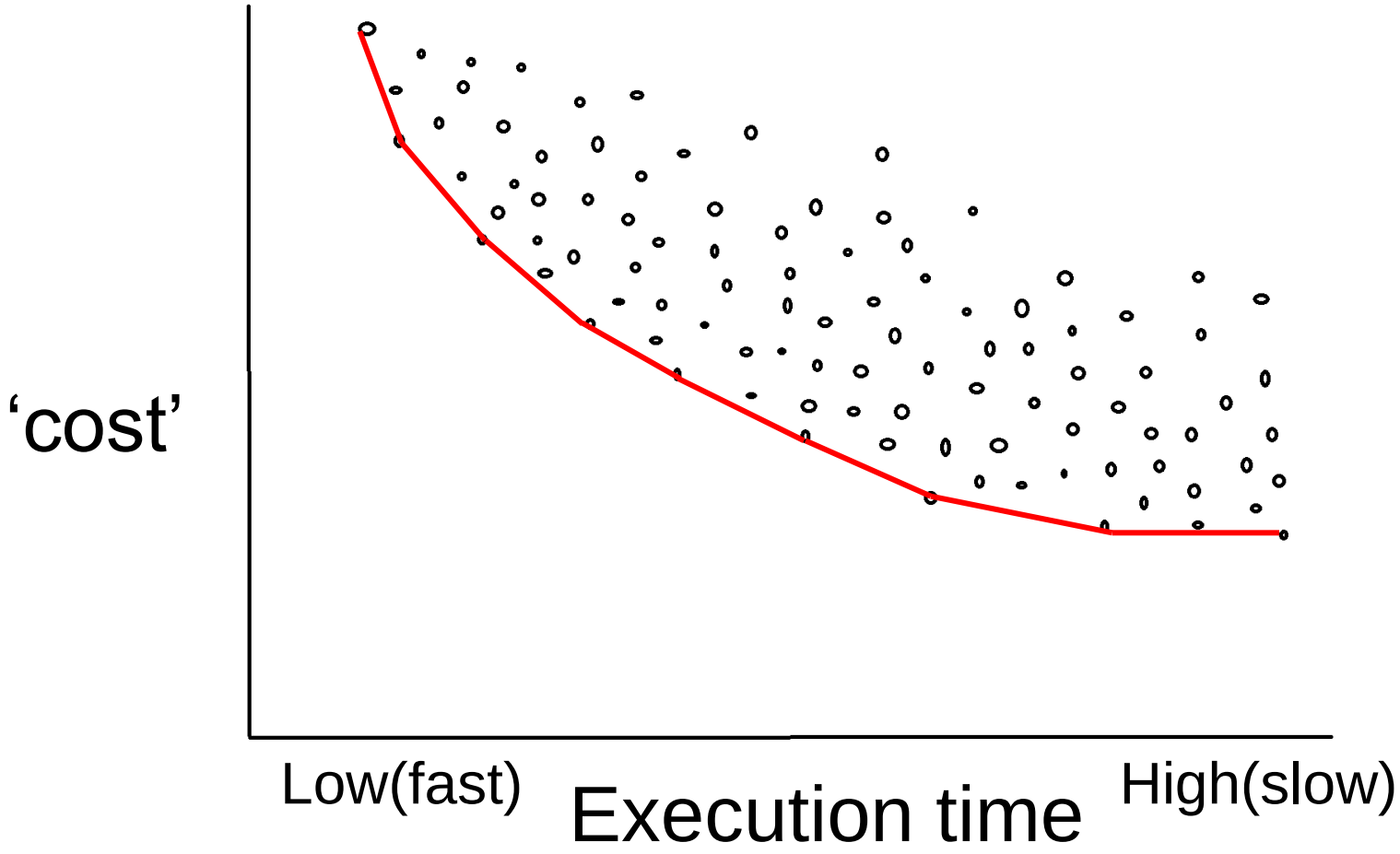
Assessing the cost of different options

We can merge all options (points in space) into one graph:



Assessing the cost of different options

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

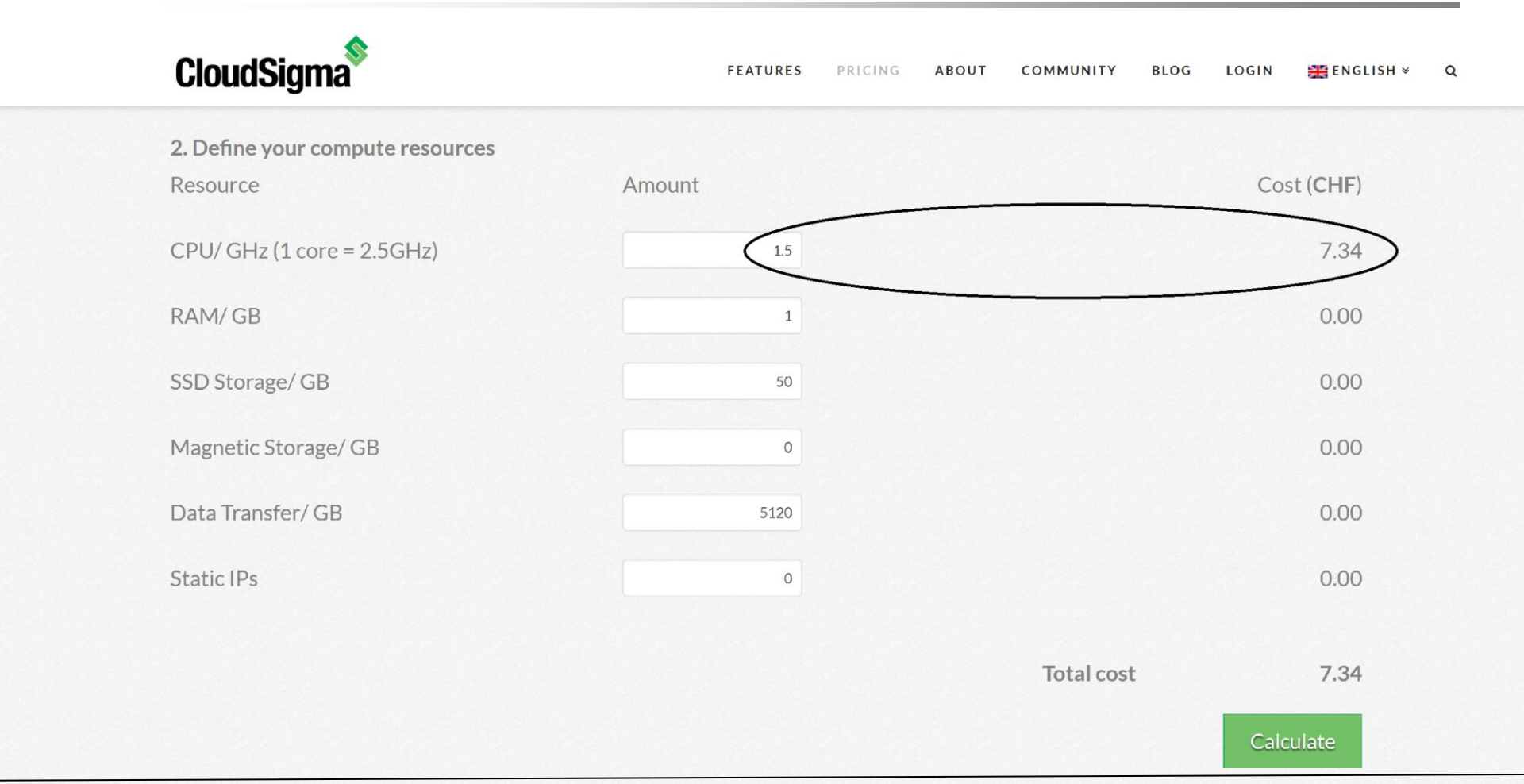


Is the current state-of-art in parallel/distributed computing research in a good position to assess such trade-offs and find a Pareto front?

The search space can be huge and we may have to limit it substantially!

Assumes a widely agreed cost model!

A problem (motivated by Cloud providers)



You pay depending on the CPU frequency you choose

2. Define your compute resources

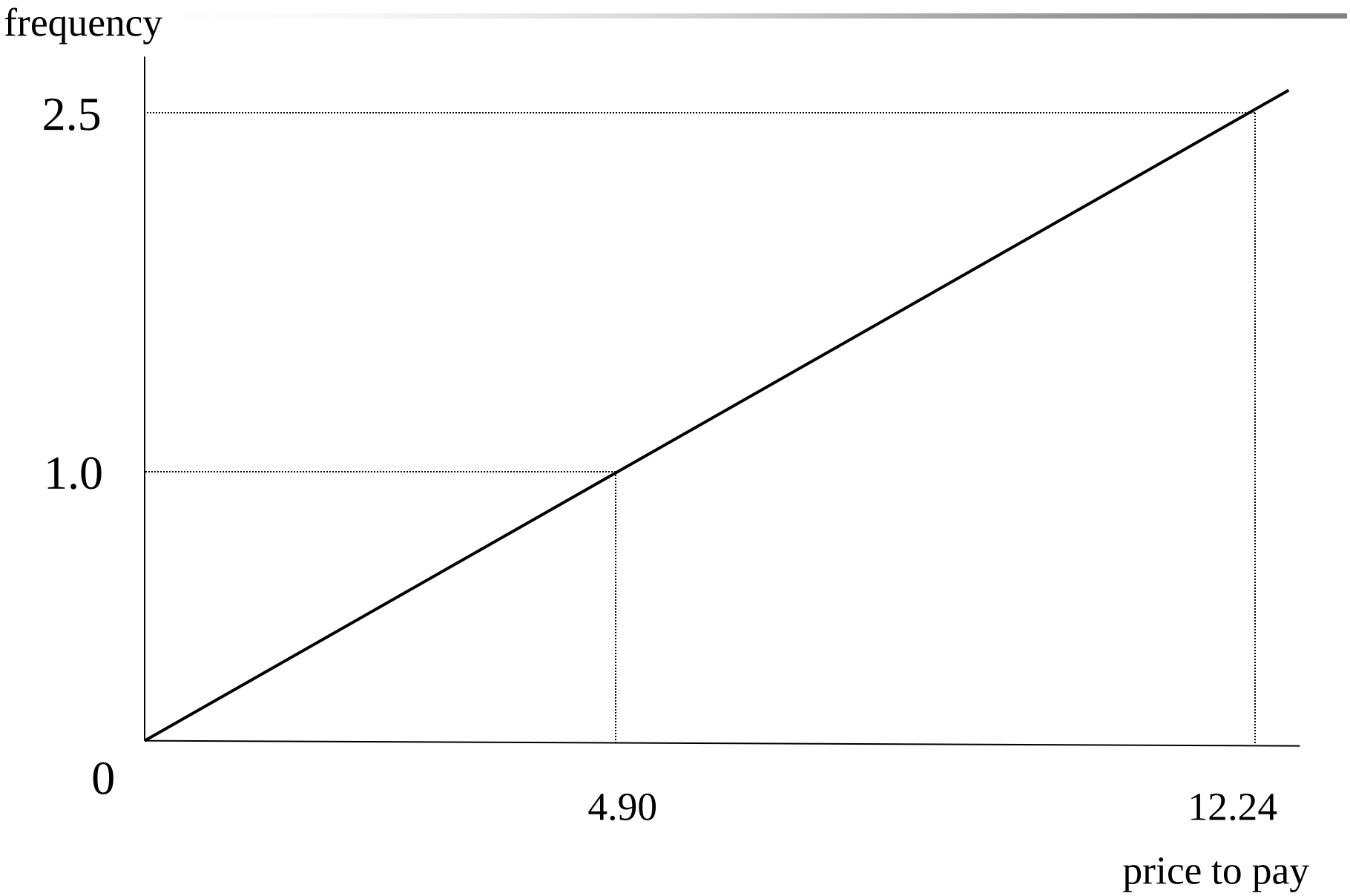
Resource	Amount	Cost (**CHF**)
CPU/ GHz (1 core = 2.5GHz)	1.701	8.33
RAM/ GB	1	0.00
SSD Storage/ GB	50	0.00
Magnetic Storage/ GB	0	0.00
Data Transfer/ GB	5120	0.00
Static IPs	0	0.00

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 know you are going 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 a huge number of combinations that **cost the same per time unit**…

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

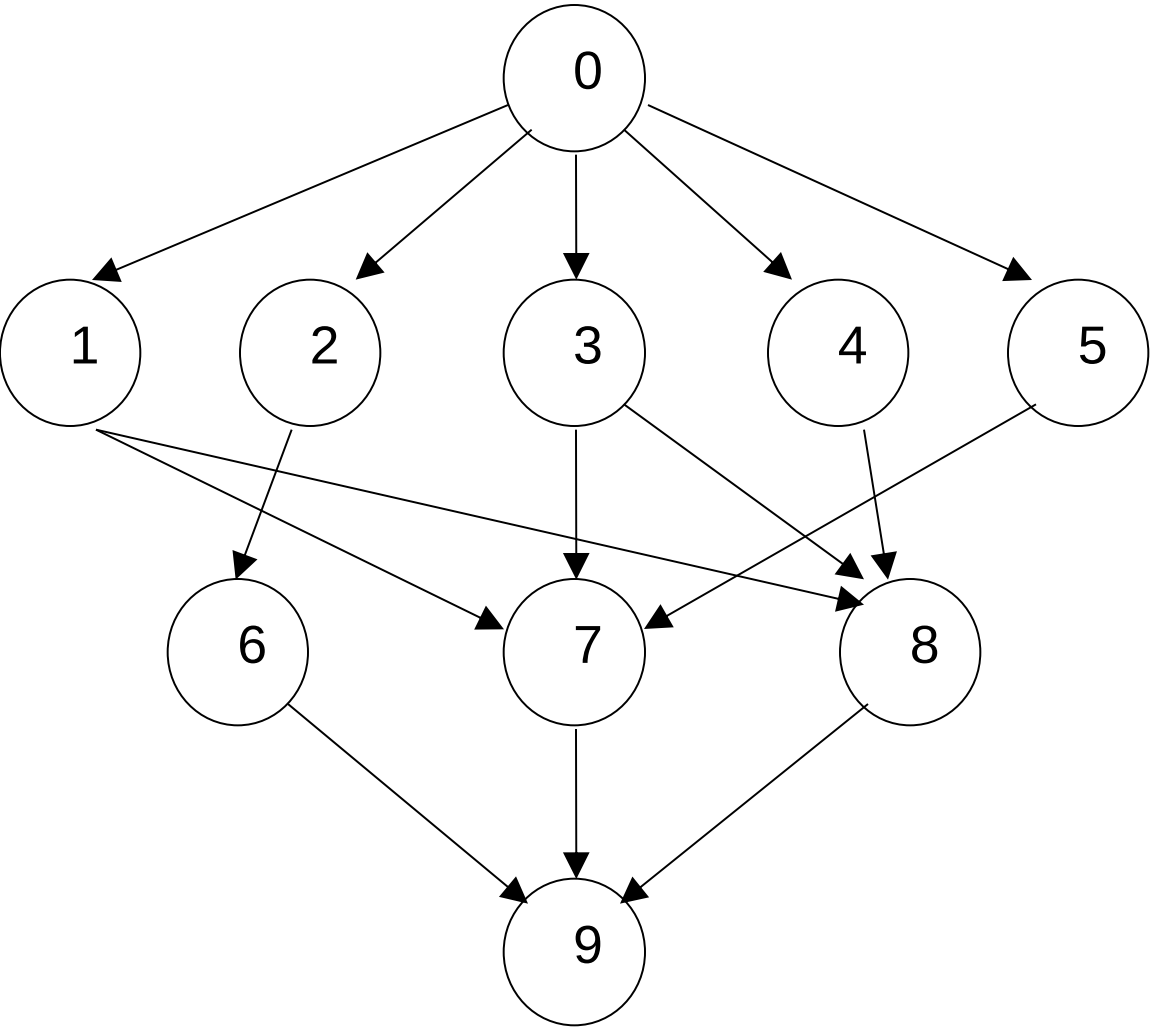
Setting the scene

Configurations with the same per unit time cost (equiv. to same average CPU frequency in the previous problem) 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 of a workflow by assigning (=**scheduling**) tasks onto multiple resources in a way that strikes 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 of the infrastructure (energy)]

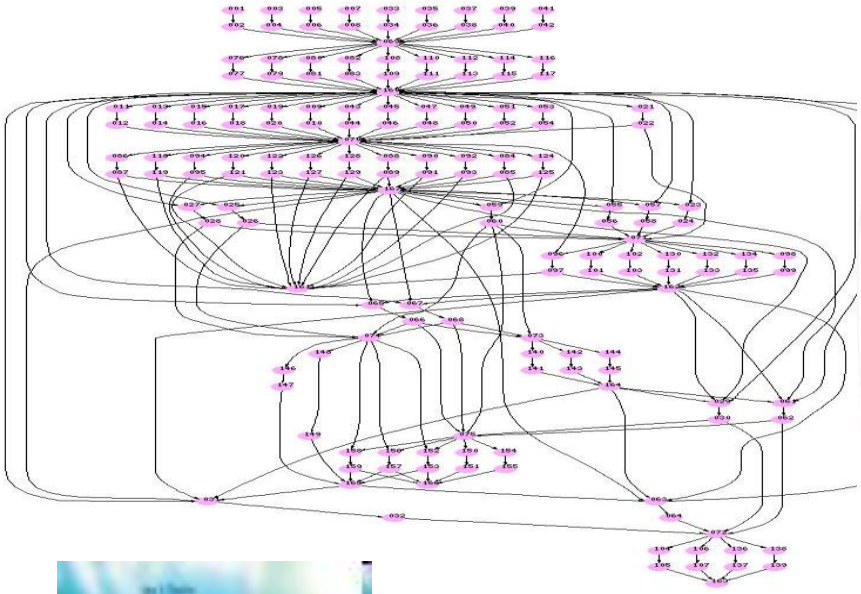
A Scientific Workflow is typically a Directed Acyclic Graph (DAG)

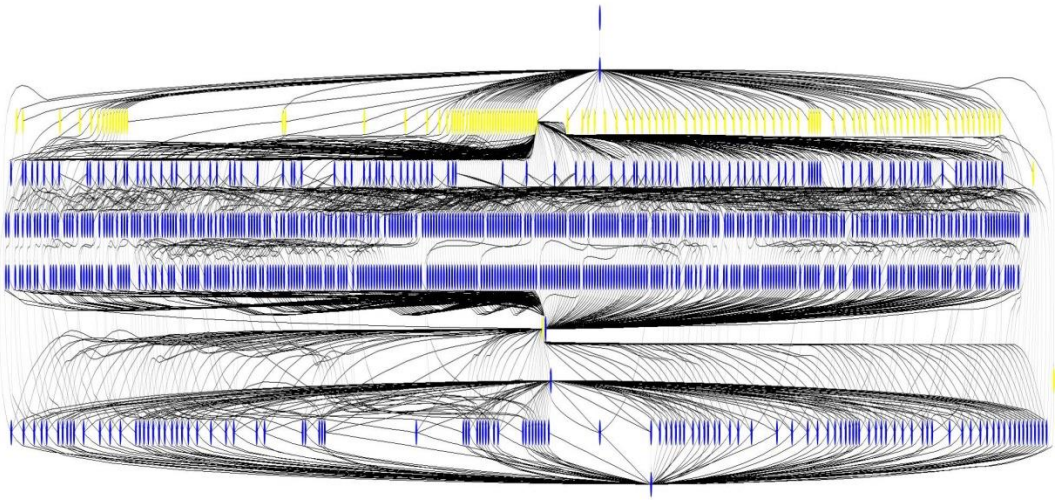


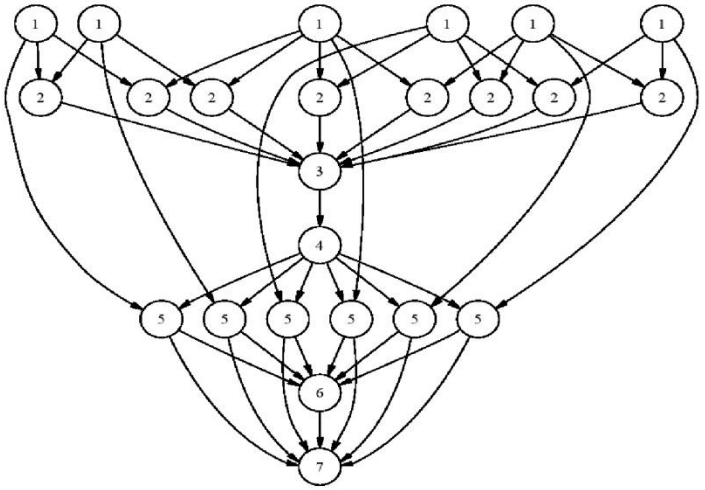
- A well-known abstraction in Computer Science
- Nodes (tasks or jobs) denote computation
- Edges denote data transfer (communication)
- Dependences between nodes (tasks) must be respected

Scientific Workflows

Many interesting (scientific) applications can be represented by DAGs



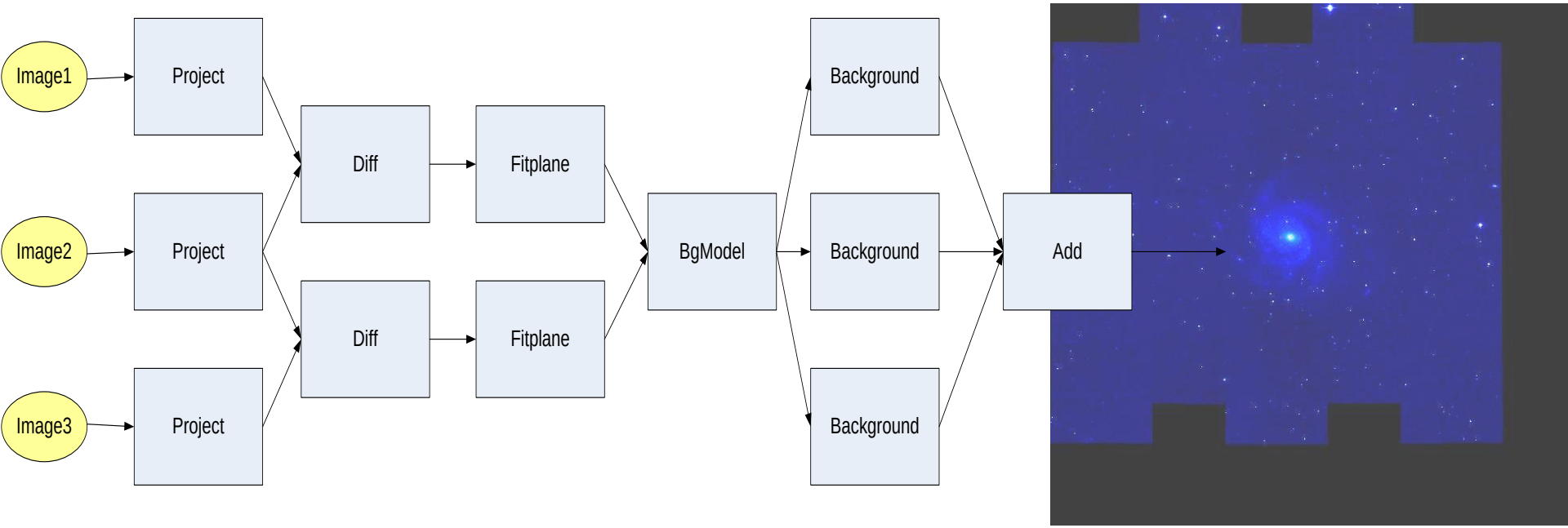




I. Taylor, E. Deelman, D. Gannon: Workflows for e-Science. Springer, 2007

Example: the Montage Workflow

- Montage: Generating science-grade mosaics of the sky (Caltech, Berriman et al.)
- http://montage.ipac.caltech.edu/



Slide courtesy: Ewa Deelman

From http://montage.ipac.caltech.edu/

The Pleiades: http://montage.ipac.caltech.edu/images/DSS_pleiades_mosaic_thumb.jpg

First announced in 2016…

https://www.ligo.caltech.edu/news/ligo20160211

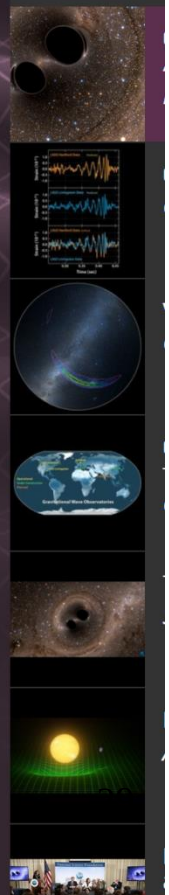
LIGO
Laser Interferometer Gravitational-Wave Observatory
Supported by the National Science Foundation
Operated by Caltech and MIT

About | Learn More | News | Gallery | Educational Resources | For Scientists | Study & Work

Image credit: The SXS (Simulating eXtreme Spacetimes) Project

Gravitational Waves Detected 100 Years After Einstein's Prediction

RELATED



The computational aspects (from https://pegasus.isi.edu/application-showcase/ligo/)

Advanced LIGO pyCBC Workflows

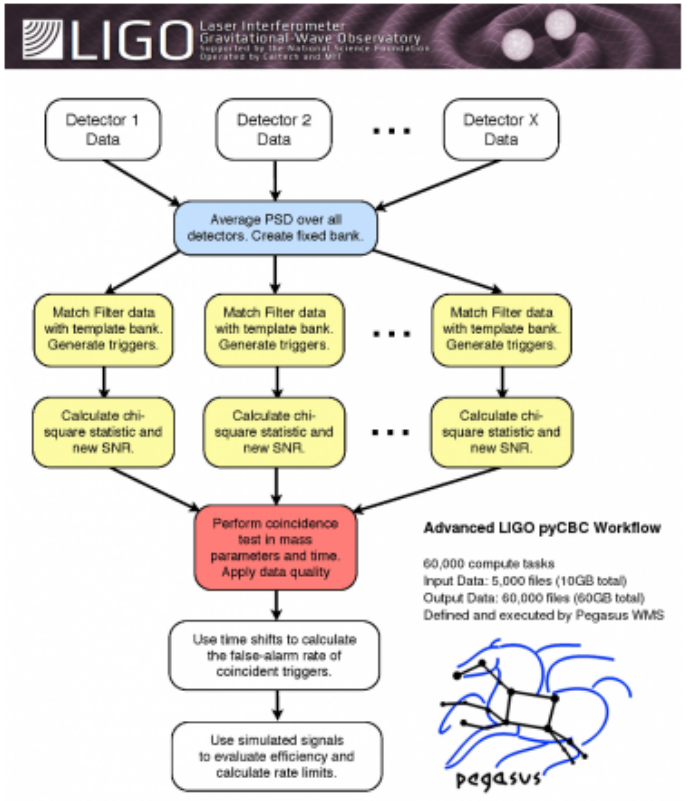
The LIGO Laboratory started the first Observing Run 'O1' with the Advanced LIGO detectors in September 2015 at a sensitivity roughly 4 times greater than Initial LIGO for some classes of sources (e.g., neutron-star binaries), and a much greater sensitivity for larger systems with their peak radiation at lower audio frequencies. The pyCBC analysis workflows are modeled as a single stage pipeline that can analyze data from both LIGO and VIRGO detectors.

These workflows are executed using Pegasus on a variety of resources such as local campus clusters, LIGO Data Grid, OSG, XSEDE in combination with Pegasus MPI Cluster and VIRGO Clusters.

Each pyCBC workflow has

- 60,000 compute tasks
- Input Data: 5000 files (10GB total)
- Output Data: 60,000 files (60GB total)

The incorporation of the Pegasus into the production pipeline has enabled LIGO users to



LIGO pyCBC Workflow. Image Credit: Samantha Usman, Duncan Brown et al

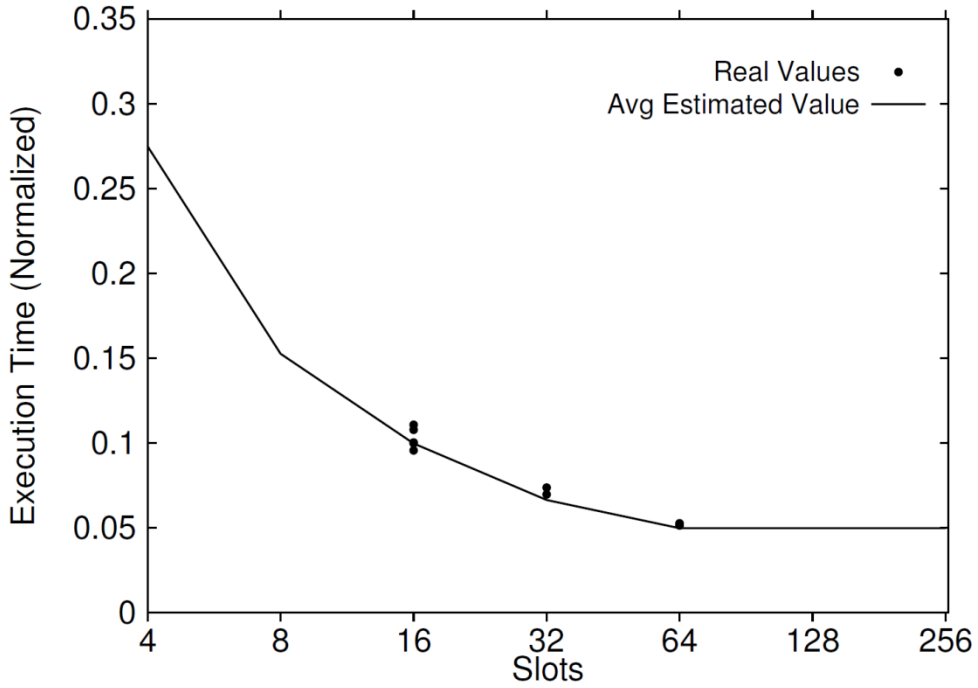
- Run an analysis workflows across sites. Analysis workflows are launched to execute on XSEDE and OSG resources with post processing steps running on LIGO Data Grid.
- Monitor and share workflows using the Pegasus Workflow Dashboard.
- Easier debugging of their workflows.
- Separate their workflow logs directories from the execution directories. Their earlier pipeline required the logs to be the shared filesystem of the clusters. This resulted in scalability issues as the load on the NFS increased drastically when large workflows were launched.
- Ability to re-run analysis later on without running all the sub workflows from start. This leverages the data reuse capabilities of Pegasus. LIGO data may need to be analyzed several times due to changed in e.g. detector calibration or data-quality flags. Complete re-analysis of the data is a very computationally intensive task. By using the workflow reduction capabilities of Pegasus, the LSC and Virgo have been able to re-use existing data products from previous runs, when those data products are suitable.

Lots of (data- & compute-intensive) scientific workflow applications available for research

https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator

The resources

- Cloud Computing resources
- How many? Let's assume the user has already made a decision on the number of nodes/slots using some performance modelling



I. Pietri, G. Juve, E. Deelman, R.Sakellariou. A Performance Model to Estimate Execution Time of Scientific Workflows on the Cloud. In 9th WORKS @ SC14, 2014.

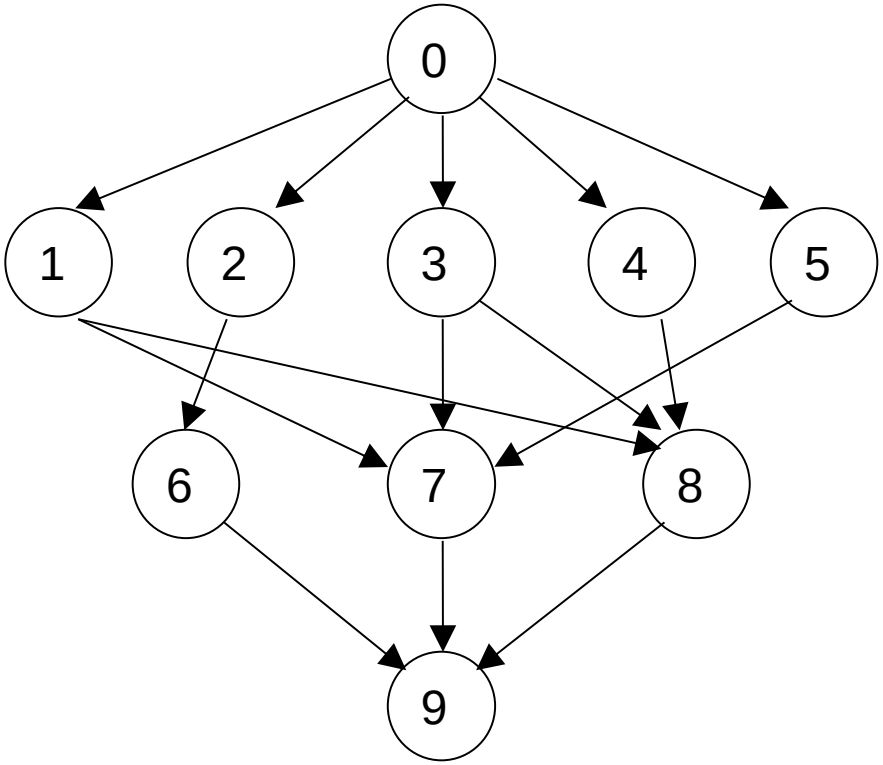
The objective

- **Cost** versus **Performance**
  - Client/User: for the least amount of money, finish as quickly as possible
  - Provider: meet user requirements at the same time minimizing the cost of running the infrastructure.

    Minimizing energy is a good target for cost from the provider's point of view.

Scheduling workflows (DAGs)

The problem: assuming a number of (possibly heterogeneous) machines, how do we decide what task to assign where (=what machine) and when to complete execution as soon as possible?



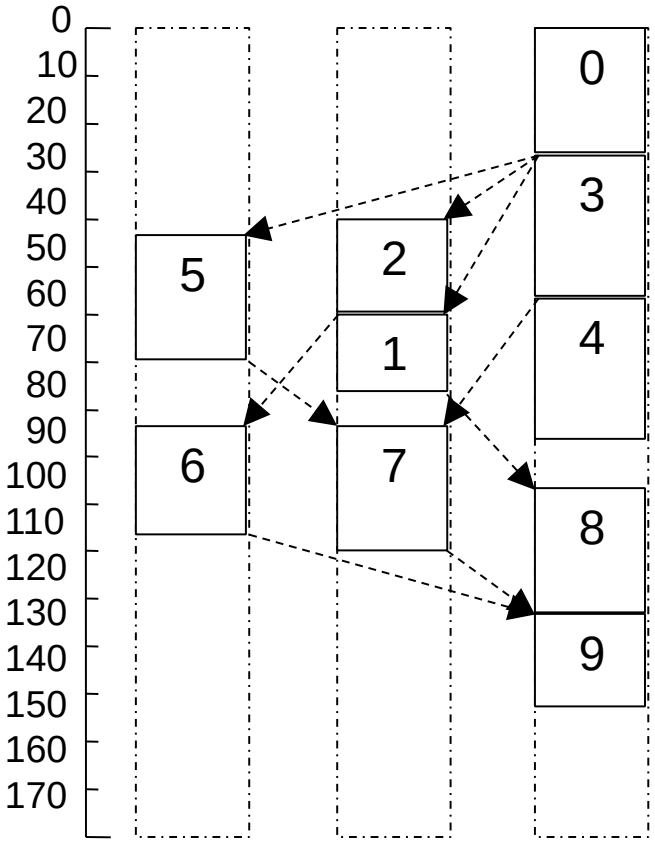
- NP-Complete problem
- Key: follow the critical path
- More than 20 heuristics in the literature for heterogeneous machines.
- Common idea: prioritize tasks with some criterion and try to allocate optimally (locally) by following this priority order.

Task execution times and communication times are assumed to be known; HEFT is a well-known heuristic that makes use of average values to compute a weight for each node and determine a scheduling order

An old observation: even small changes in the order may make a big impact on makespan

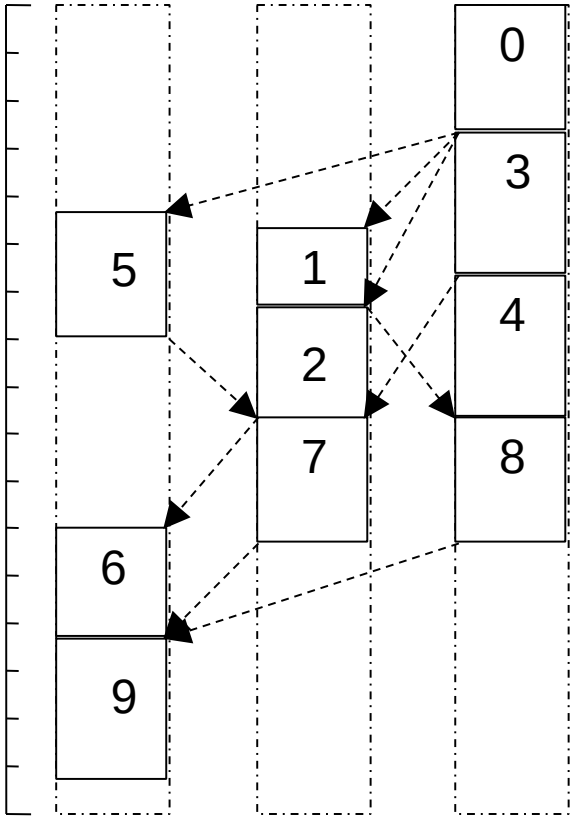
- Two schedules for two different schemes using HEFT

Worst {0, 3, 5, 2, 1, 4, 7, 8, 6, 9}



Makespan: **143**

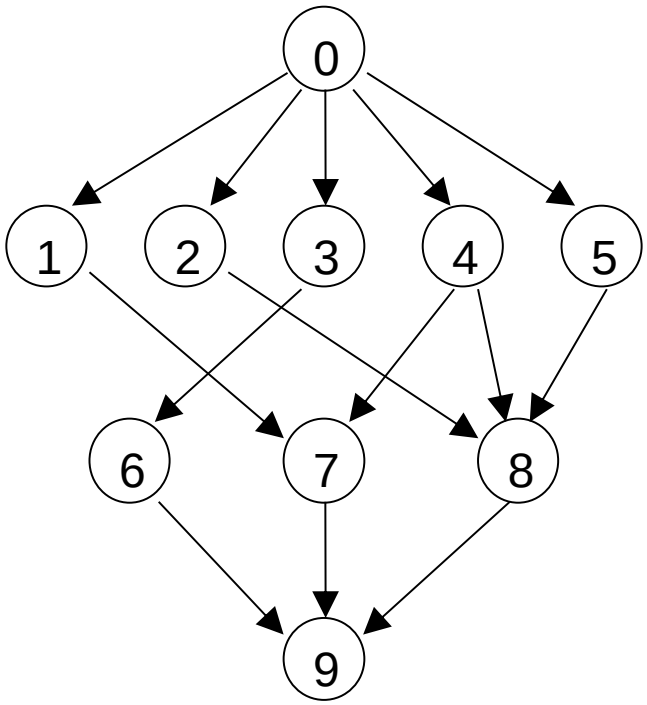
Mean {0, 3, 5, 1, 2, 4, 7, 8, 6, 9}

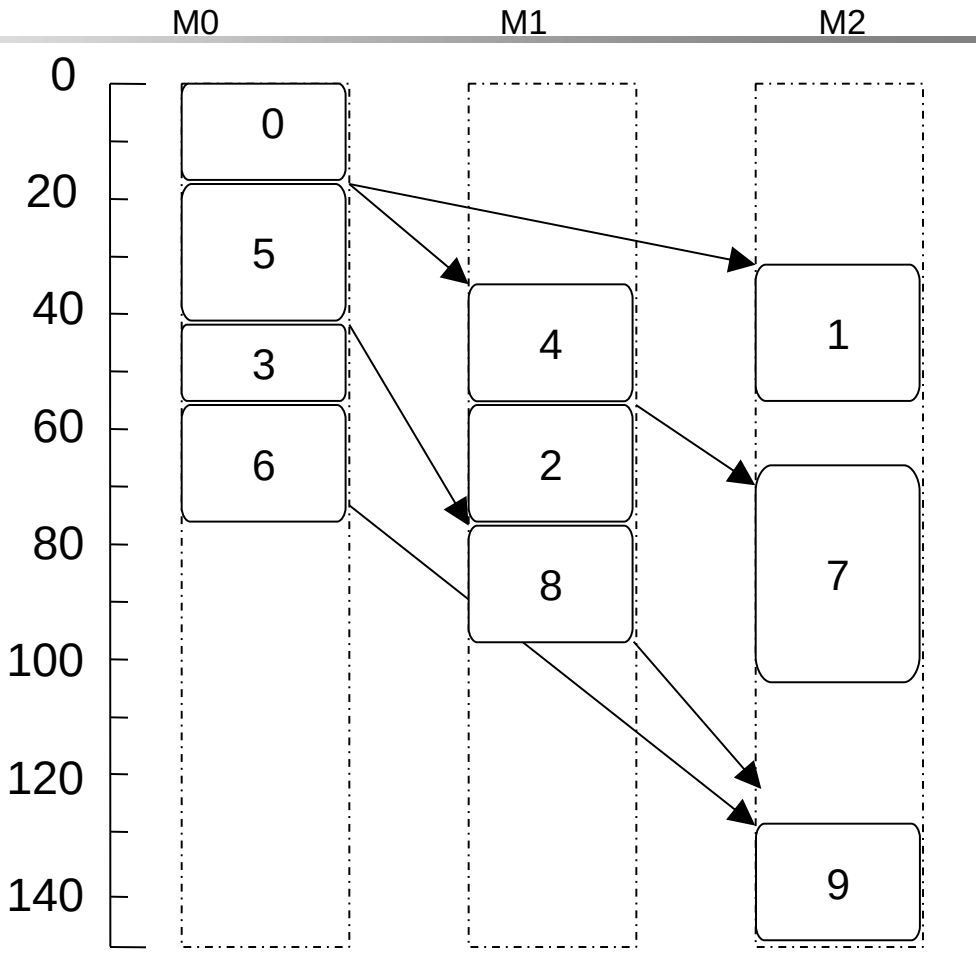


Makespan: **164**

H.Zhao,R.Sakellariou. An experimental study of the rank function of HEFT. *Euro-Par'03*.

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 (1)

- ***Spare time*** indicates the maximum time that a node, *i*, may delay without affecting the start time of an immediate successor, *j* (on the DAG or the same machine).
- ***Slack*** indicates the maximum time that a node, i, may delay without affecting the overall makespan.
  - *Slack(i)=min(slack(j)+spare(i,j))*, for all successor nodes *j* (both on the DAG and the machine)
- In the example before:
  - tasks 1, 7 and 8 have some spare time
  - tasks 2 and 4 has some slack
  - tasks 0, 5, 3, 6, 9 have zero spare time / slack.

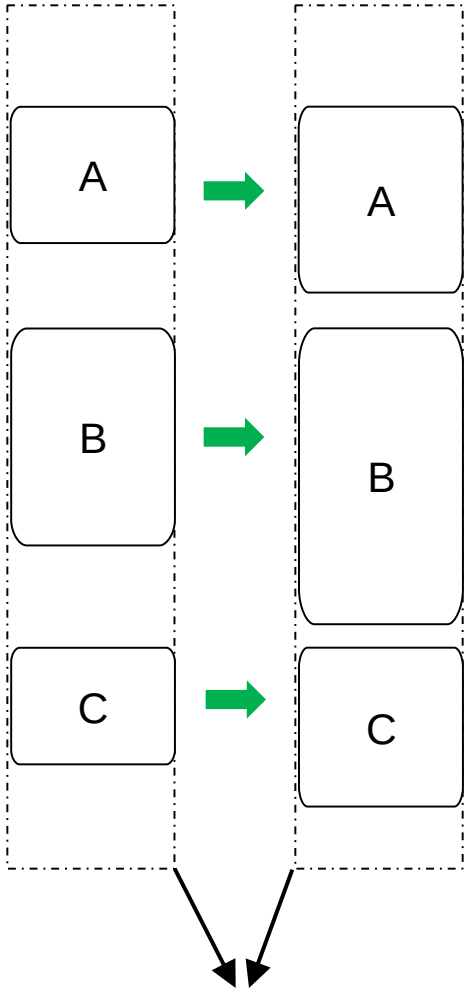
Spare time & Slack (2)

- 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 abundance of parallelism, we'll have some spare time and slack
- **One could think they provide interesting opportunities!**

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

Slowing down tasks…

- If all the tasks allocated on one machine have a spare time, which is at least 20% of each task's execution time, then we can slow down execution on that machine by 20% (this, in theory, means we can choose a CPU running at a lower frequency, which is cheaper)
- Overall execution time is not affected
- DVFS (Dynamic Voltage and Frequency Scaling) is now a common technique with many architectures.



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 can be 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

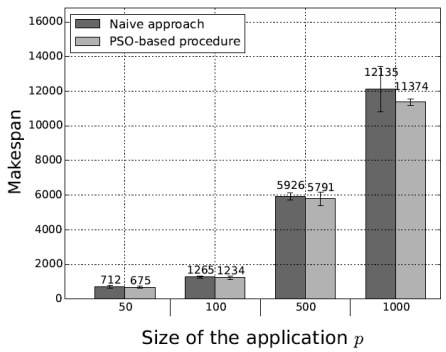
(*assuming the same average frequency overall)

A problem...

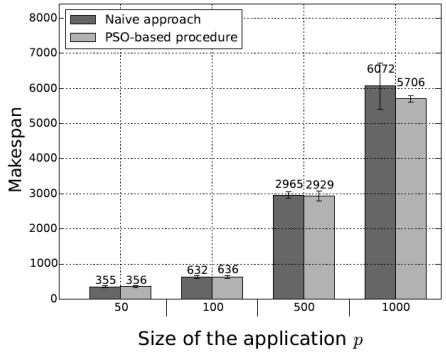
- For a given:
  - total amount of CPU frequency
  - overall number of resources
- How to split the total amount of CPU frequency to minimize execution time?
- E.g., 10GHz with 5 resources can be split:
  - Everything running at 2GHz (naïve approach)
  - Or, say, 3GHz, 2.5GHz, 2GHz, 1.5GHz, 1GHz
  - Or, …
  - (all combinations with same mean frequency)

T. Ghenez, I. Pietri, R.Sakellariou, L. F. Bittencourt, E. R.M. Madeira. A Particle Swarm Optimization Approach for Workflow Scheduling on Cloud Resources Priced by CPU Frequency. UCC 2015.

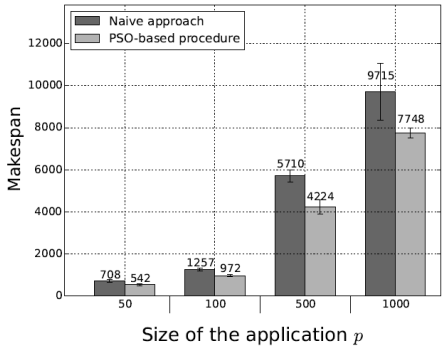
Significant gains compared to the naïve approach!



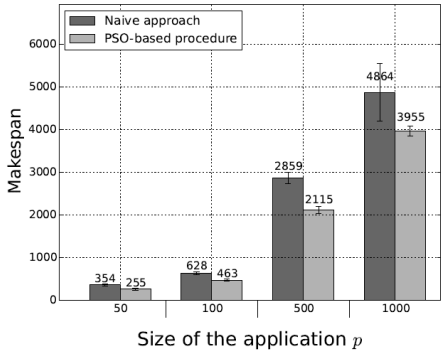
(a) Sample A



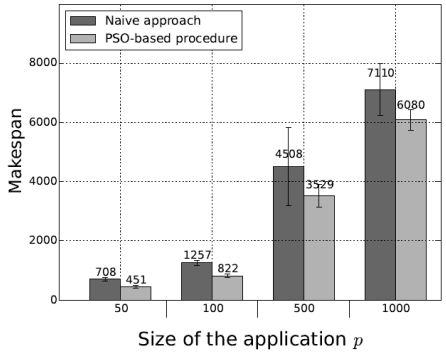
(b) Sample B



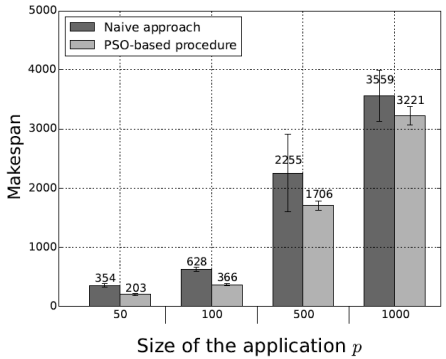
(c) Sample C



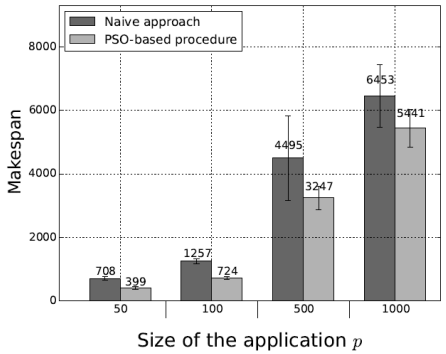
(d) Sample D



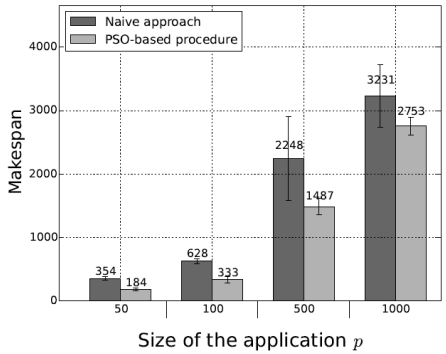
(e) Sample E



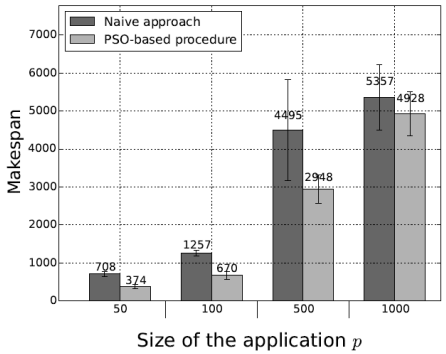
(f) Sample F



(g) Sample G



(h) Sample H



(i) Sample I

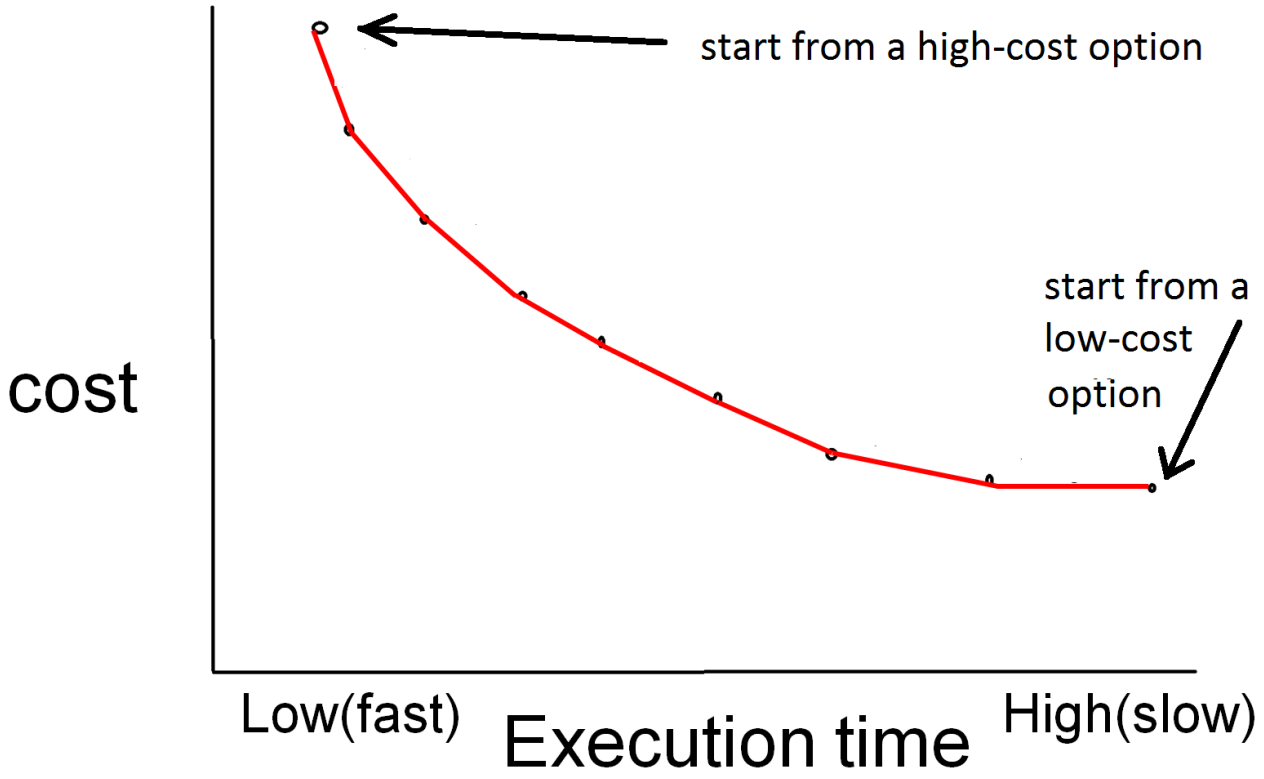
Keeping the same average frequency overall is only part of the search space.

We are interested in examining how execution time changes with cost and we'd like to find the Pareto front! (remember slide 18?)

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.

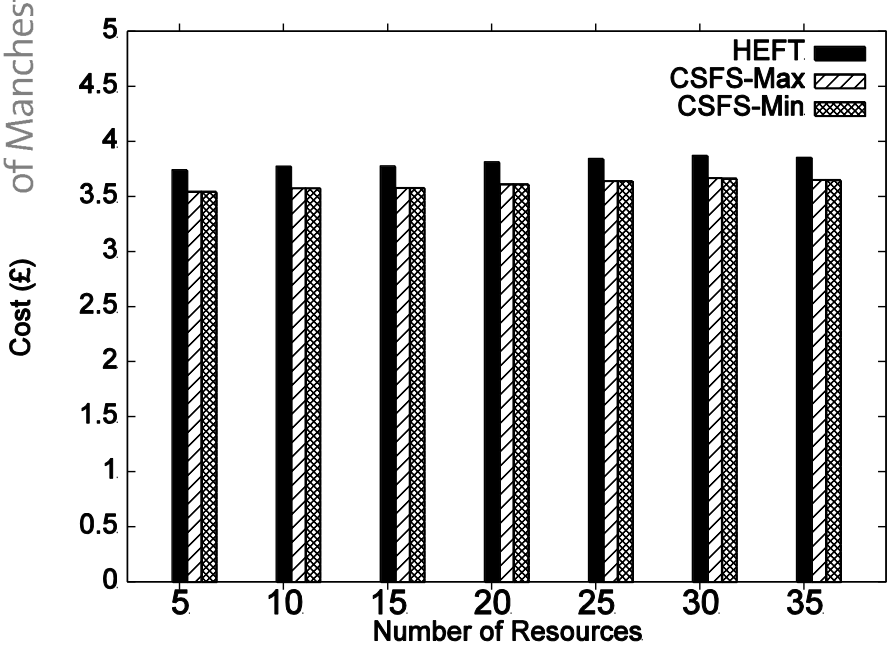
Cost-based Stepwise Frequency Selection starting at Min Frequency

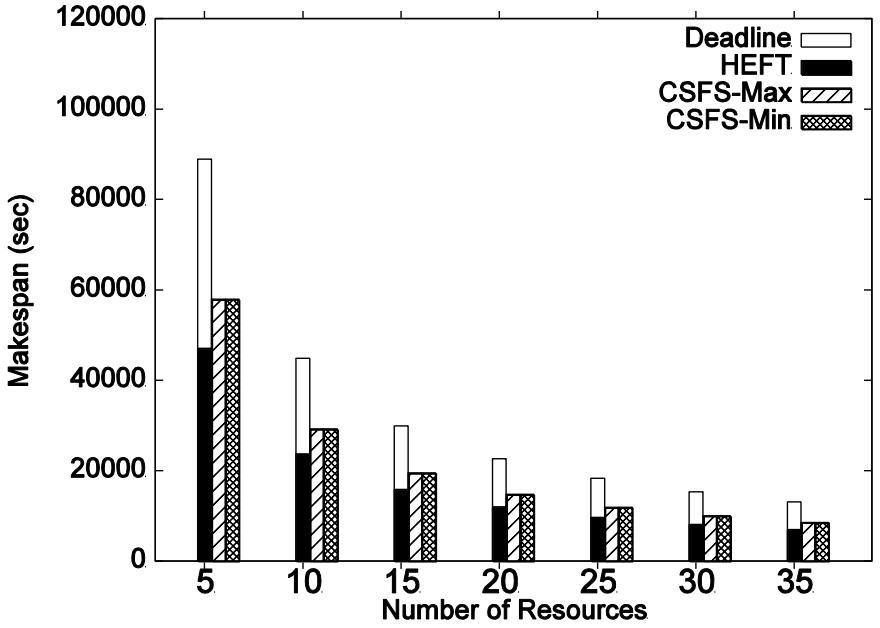
CSFS-Min

- 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

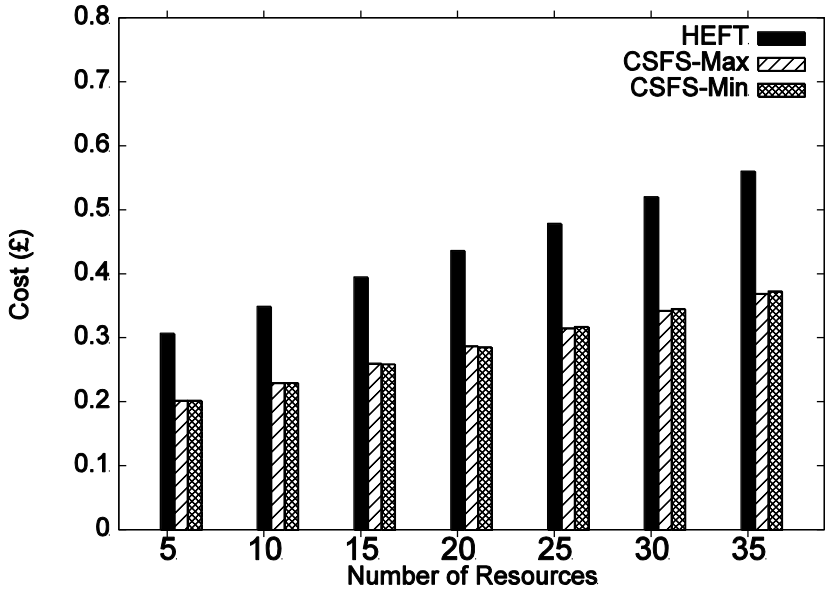
I. Pietri, R.Sakellariou. Cost-Efficient CPU Provisioning for Scientific Workflows on the Cloud. GECON 2015 (extended version to appear in FGCS)

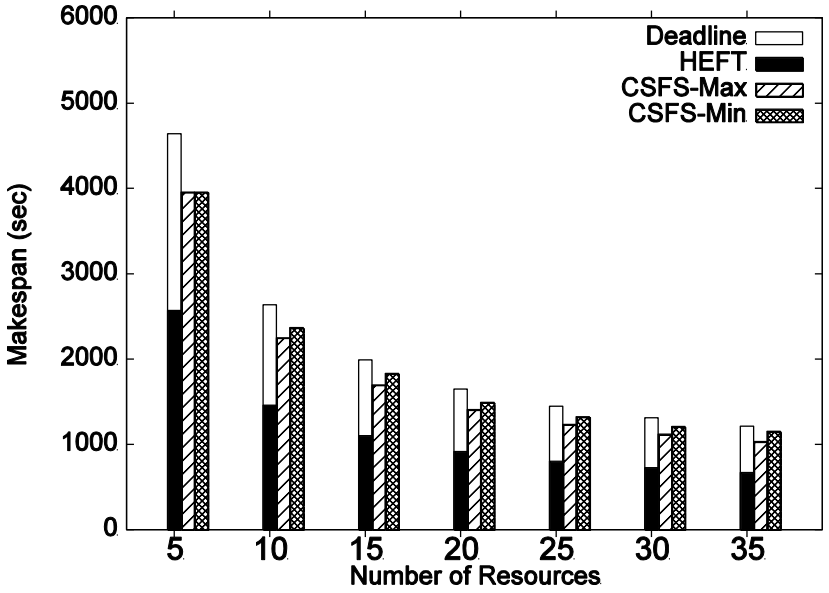
LIGO Under Linear Pricing





Montage under Superlinear Pricing

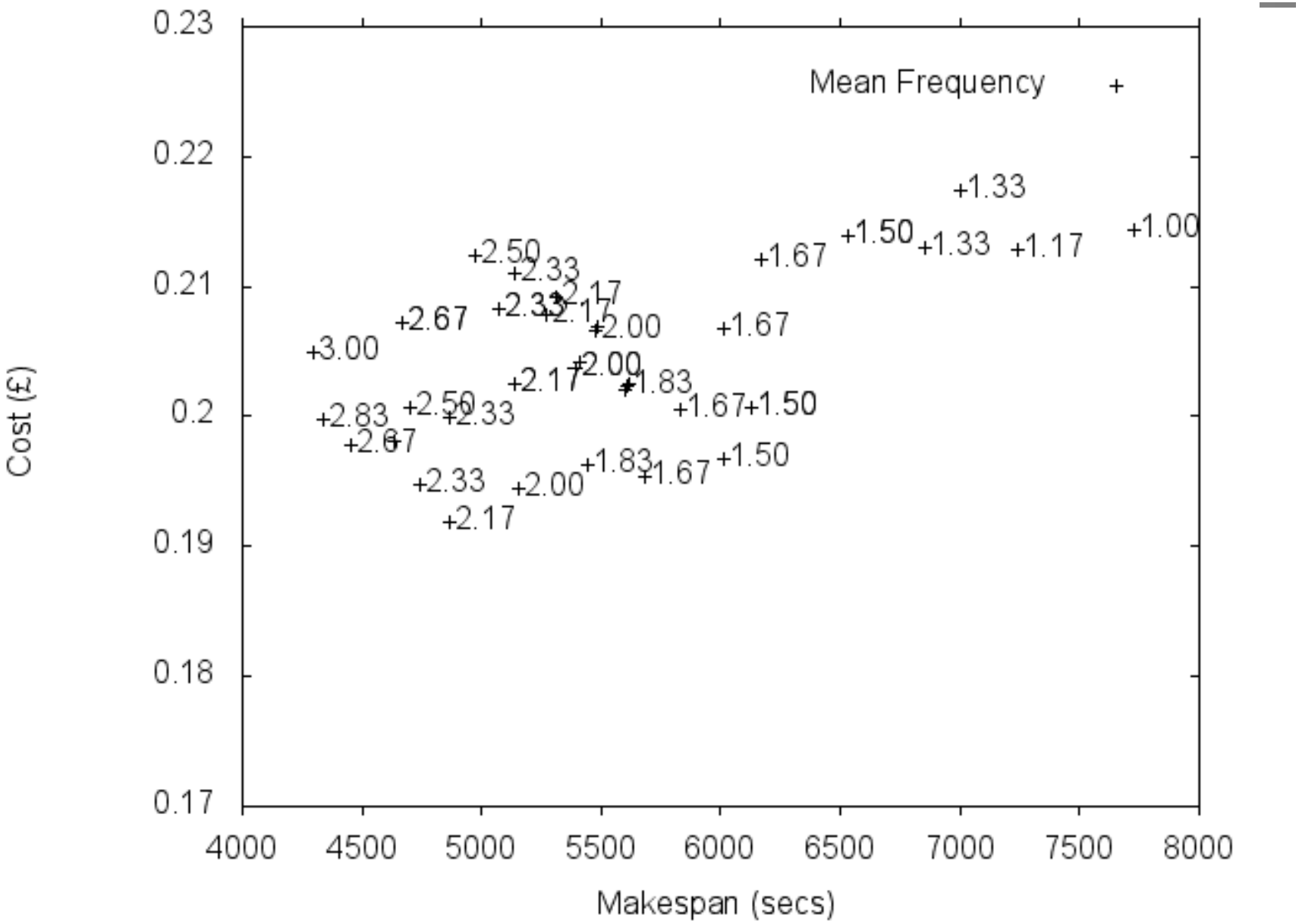




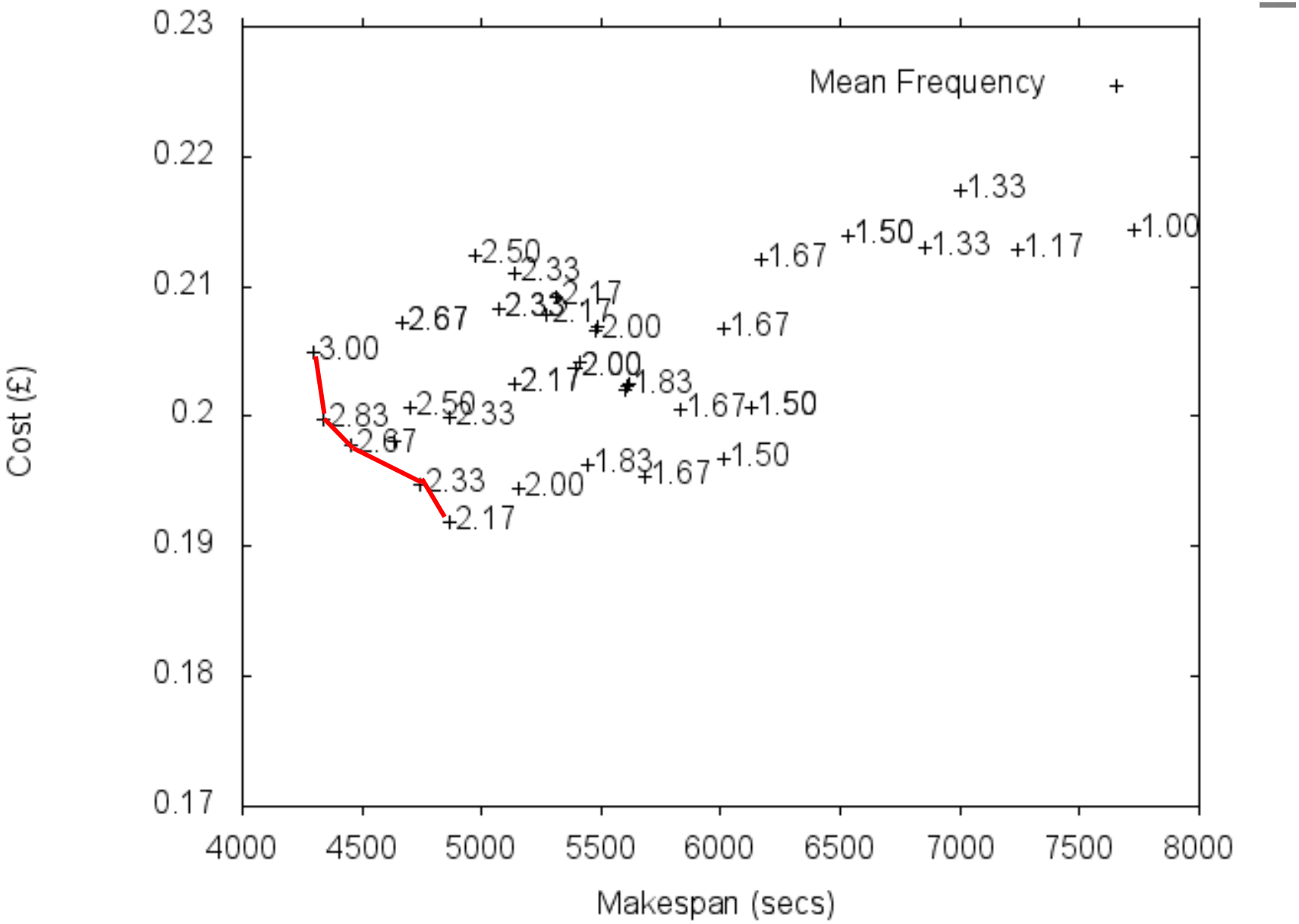
Some observations

- 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. Lots of scope for other optimization approaches.
- Full results in the GECON2015 paper (also trying different pricing models) and FGCS (to appear)

Plotting some solutions with 3 CPUresources



Plotting some solutions with 3 CPUresources

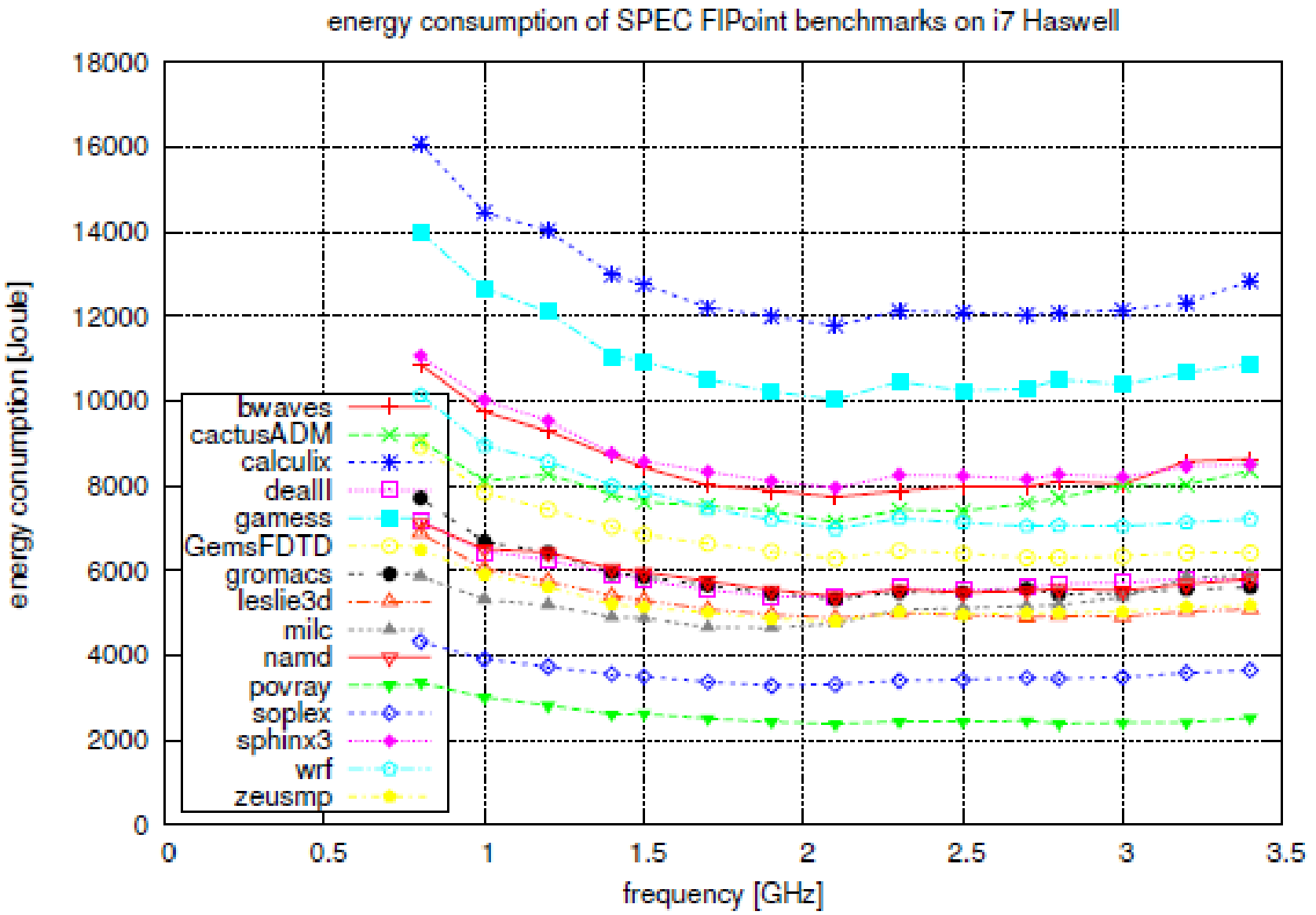


Energy Cost vs Performance

- Monetary cost was easy to model (based on pricing model):
  - Essentially taken as charged by provider
- Instead of (monetary) 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
paper in Concurrency and Computation Practice and Experience, 2015)



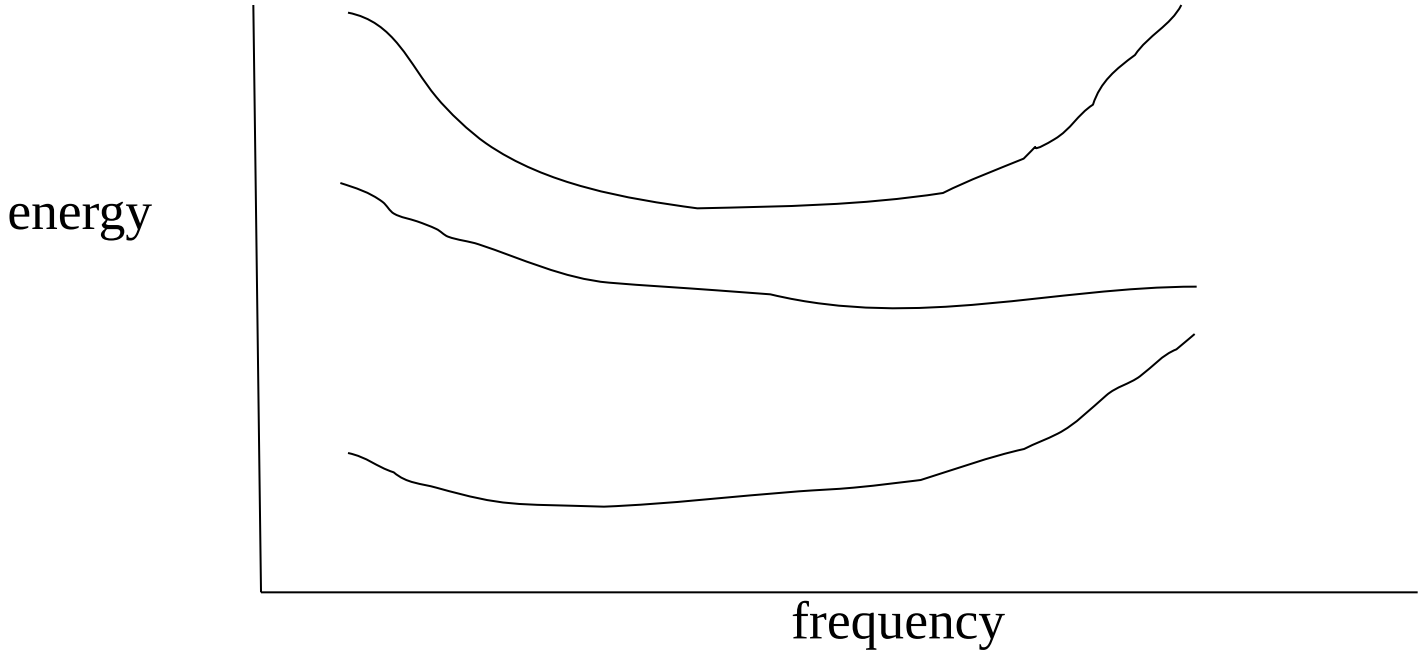
The idea – an iterative approach

- 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

- **Anecdotal evidence seems to indicate that even when the perceived utilization (based on user requests) of a Cloud system is high, the actual utilization of the system may be low.**
  - Users often request many more resources than actually needed. Providers do not particularly care as long as this translates to money!
  - Lots of scope for optimization by adjusting utilization to the workload characteristics.
    - E.g., reduce CPU frequency for I/O-intensive jobs.

Drazen Lucanin, Ilia Pietri, Ivona Brandic and Rizos Sakellariou. A Cloud Controller for Performance-Based Pricing. *IEEE Cloud 2015 & TCC* (to appear)

Even more trade-offs (I)

- Combining frequency scaling with VM migration/consolidation may lead to useful savings (an energy-revenue trade-off)
- Perceived-performance pricing, applies frequency scaling to I/O intensive jobs.

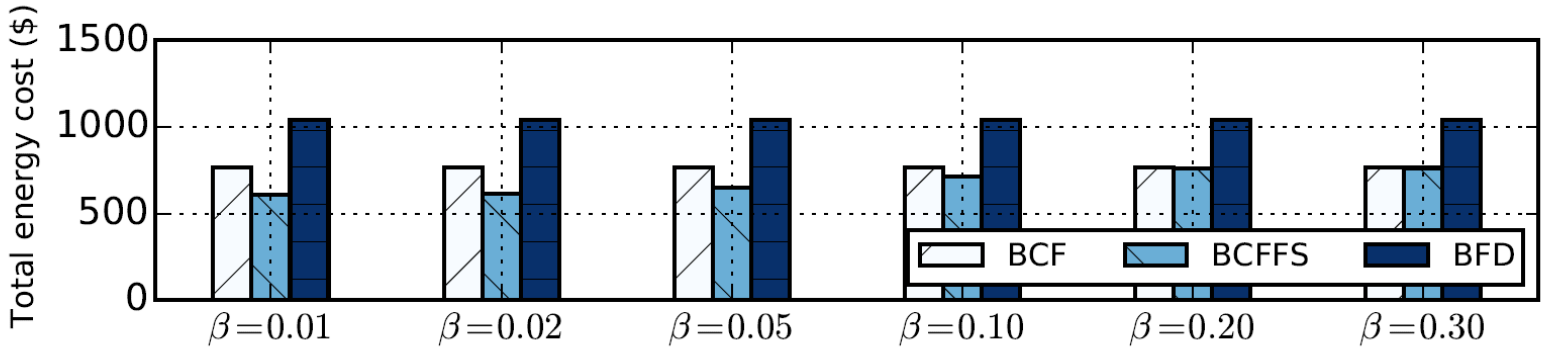


Fig. 11. Energy cost savings for VMs with different fixed CPU-boundedness.

Drazen Lucanin, Ilia Pietri, Ivona Brandic and Rizos Sakellariou. A Cloud Controller for Performance-Based Pricing. *IEEE Cloud 2015 & TCC* (to appear).

Even more trade-offs (II)

- It has been argued that there is a need “to design algorithms that communicate as little as possible” (a communication-computation trade-off)
- Scope for some small extra effort in computation if it can result in significant communication savings

(a) LIGO

nodes	HEFT solutions	Pareto solutions	Time	ComCost
	time (sec), comCost (MB)		savings (%)	
5	47380.93, 322.51			
10	24090.39, 351.26			
15	16261.31, 364.65	17497.77, 164.23	−7.60	54.96
20	12384.79, 362.44	13616.24, 165.85	−9.94	54.24
25	10018.23, 354.71			
30	8465.75, 345.42	9044.67, 167.56	−6.84	51.49
		9677.95, 167.14	−14.32	51.61
35	7371.83, 340.676	8738.92, 169.68	−18.54	50.19
		8074.22, 170.47	−9.53	49.96

(b) Epigenome

nodes	HEFT solutions	Pareto solutions	Time	ComCost
	time (sec), comCost (MB)		savings (%)	
5	331271.93, 1070.01	329805.56, 550.03	0.44	48.60
10	182615.74, 1126.06			
15	134011.90, 1136.67	145700.98, 620.73	−8.72	45.39
20	109881.86, 1094.81			
25	94212.61, 1149.43	116737.13, 650.48	−23.91	43.41
30	85539.91, 1147.92	91968.80, 716.26	−7.52	37.60
35	76567.91, 1153.18	103054.69, 663.21	−34.59	42.49
		85498.13, 725.52	−11.66	37.09

(c) Montage

nodes	HEFT solutions	Pareto solutions	Time	ComCost
	time (sec), comCost (MB)		savings (%)	
5	2570.38, 11113.82	2570.02, 11079.89	0.01	0.31
10	1468.55, 12385.99			
15	1102.11, 13015.85			
20	921.74, 13209.75	945.15, 11712.01	−2.54	11.34
25	809.87, 13319.46			
30	734.64, 13412.45	756.27, 12214.75	−2.94	8.93
35	678.79, 13471.16	700.75, 12406.85	−3.24	7.90
		700.40 12448.56	−3.18	7.59

Ilia Pietri and Rizos Sakellariou. Scheduling Data-Intensive Scientific Workflows with Reduced Communication. In *SSDBM’18*.

Controlling the trade-offs

Virtual to Physical Machine placement (VM to PM)

- Demand-based: Lots of contention (+overhead of VM migration) but better PM utilization
- Reservation-based: Fewer VM migrations but worse PM utilization
- Idea: have a ‘knob’ to choose between the two ‘extremes’ (demand- & reservation-based)
- Challenge: the same little shift of the knob does not always lead to the same change (proportionally).

A. Mosa and R. Sakellariou. Virtual Machine Consolidation for Cloud Data Centers Using Parameter-Based Adaptive Allocation, *ECBS 2017* & Dynamic Tuning for Parameter-Based Virtual Machine Placement, *ISPDC 2018*.

Problem exists in different settings

- E.g., ARM Big.Little
- System-on-chip
  - Processors coherently connected
    - ARM Cortex-A15 (Big) high performance
    - ARM Cortex-A7 (Little) power efficiency
  - Frequency selection
    - Freq. range per cluster
    - Individually per core

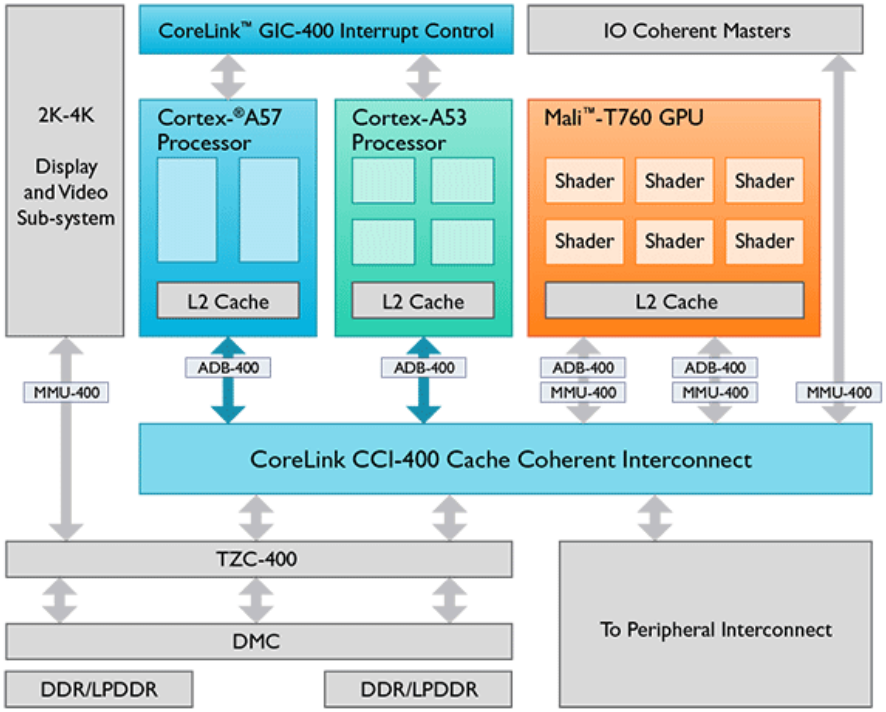
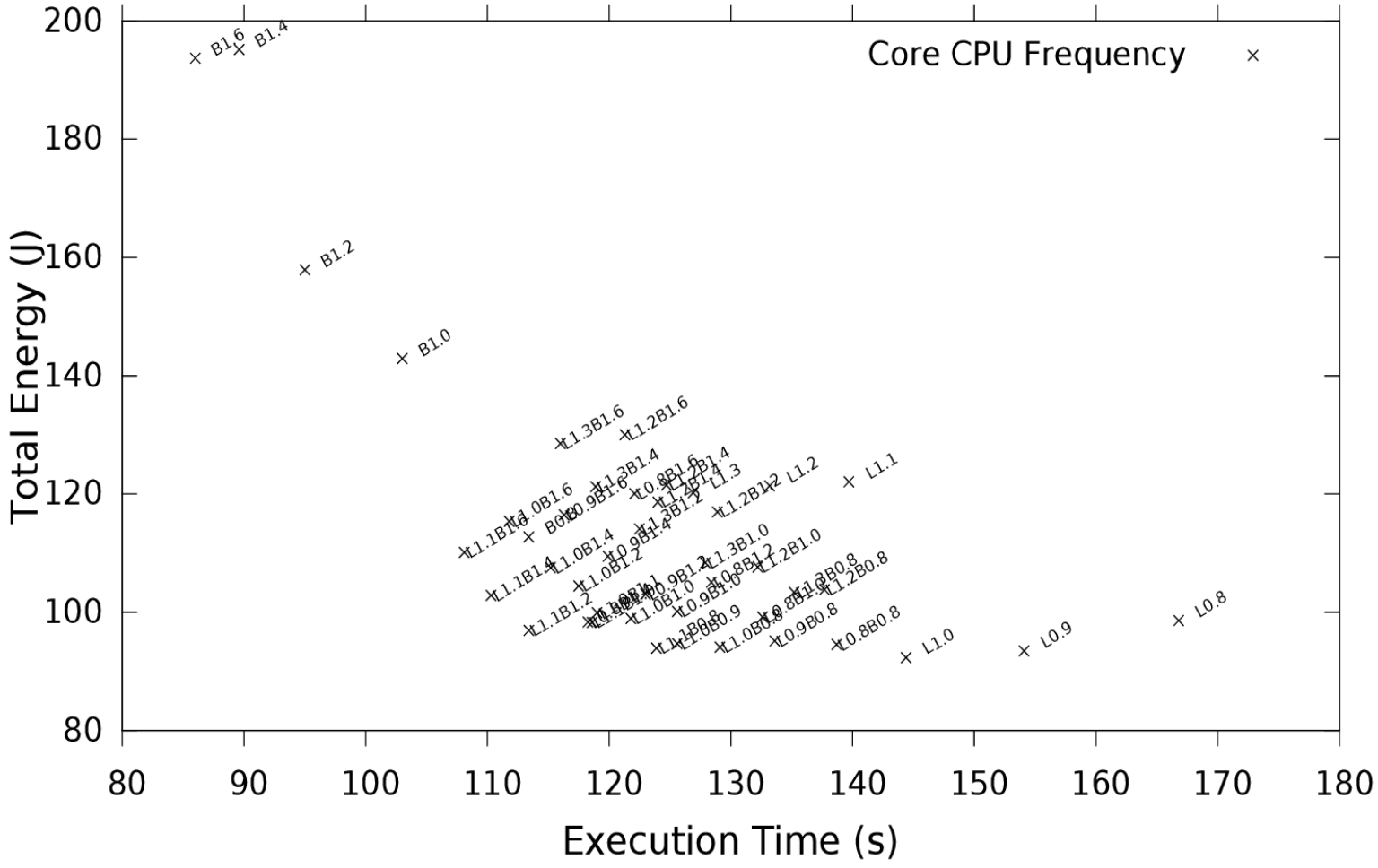


Image credit: https://www.arm.com/products/processors/biglittleprocessing.php

A small case study using Montage



Ilia Pietri, Sicong Zhuang, Marc Casas, Miquel Moreto and Rizos Sakellariou. Evaluating Scientific Workflow Execution on an Asymmetric Multicore Processor. In *HeteroPar 2017*.

to recap and consider trade-offs in the big picture…

Lots of trade-offs and excitement…

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

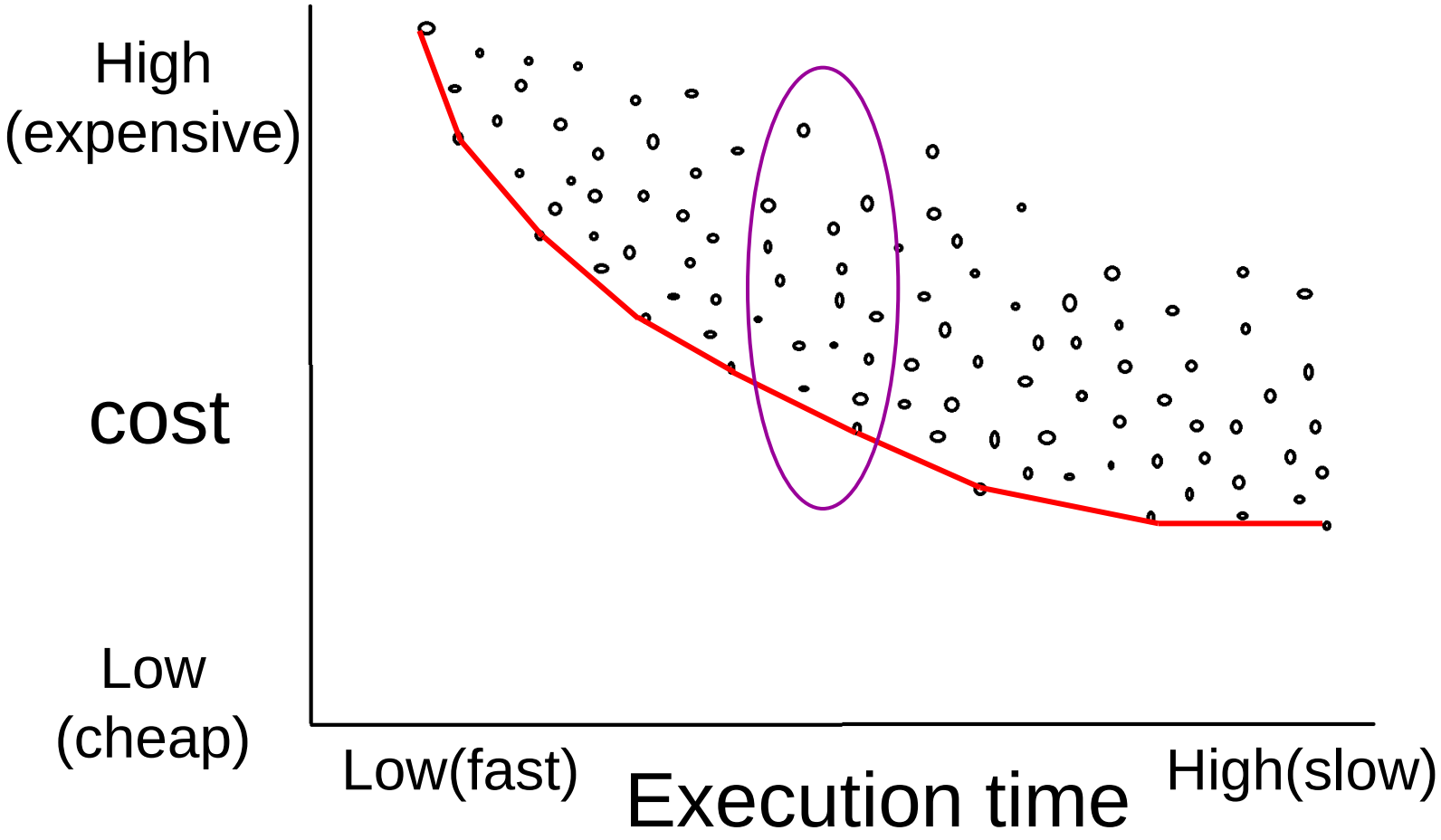
However, the **growing heterogeneity** of modern platforms and the **plethora of different configurations** suggest that somebody may need to deal with many questions involving trade-offs, the simplest form of which could be:

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

Understand the search space…

…not only part of it!

(solutions depend on requirements – don't assume equal weight)



The big picture and the challenges

- The current mindset is geared towards single-objective optimization (mostly execution time). Different objectives are often bundled as one.
- Cost
  - Cost could be: energy, number of failures, memory usage, storage, etc… There are various trade-offs between all these and performance.
  - Rather easy(?) to deal with a pricing model provided by somebody else.
- The Challenges
  - Extensive Experimentation to understand different trade-offs
  - Good Performance Models / (or at least some good Rules of Thumb)
  - Optimization Techniques (multi-criteria optimization is challenging, especially under uncertainty)
  - Software approaches that take into account different trade-offs.

More trade-offs in relation to user/business requirements with thanks to …

Industry 4.0: Background

- Next transformational change in industrial processes.
- Take advantage of the abundance of data to make ‘smart decisions’ in manufacturing (‘*smart manufacturing*’).
- Builds upon:
  - Cyber-Physical and Advanced Embedded Systems
  - Internet-of-Things
  - Data analytics
  - Cloud & High-Performance Computing

Current State

- Significant reliance on human experts
- Isolated computer systems
- Issues:
  - Systems do not scale
  - Central points of failure
  - Difficult to manage complexity
- Key challenge:
  - “System-of-systems” (different software / hardware components need to collaborate for the same goal)
  - Different trade-offs: computation, energy, storage, …

DISRUPT project

- http://www.disrupt-project.eu/
- End-users:
  - CR FIAT, ARCELIK
- Technology Providers:
  - Software AG, ATC SA, BOC GMBH, SIMPLAN AG
- Universities/Research Institutes:
  - CNR, AUEB, UNIMAN
- Research & Innovation Action (FOF), ~€3.5M, 2016-2019.
- Key goal: move the boundaries towards the Industry 4.0 vision with a specific focus on disruptions in production.

Use Cases

Management of Inbound Logistics

CRF, Automotive

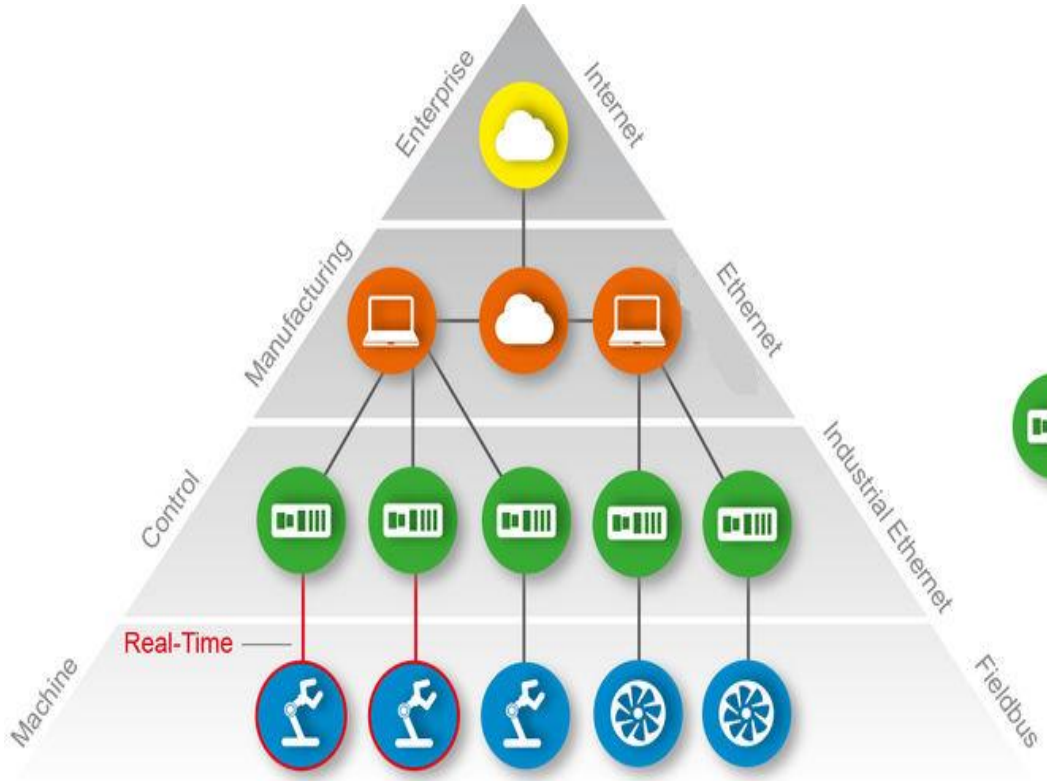
Event-enabled Production Planning and Control

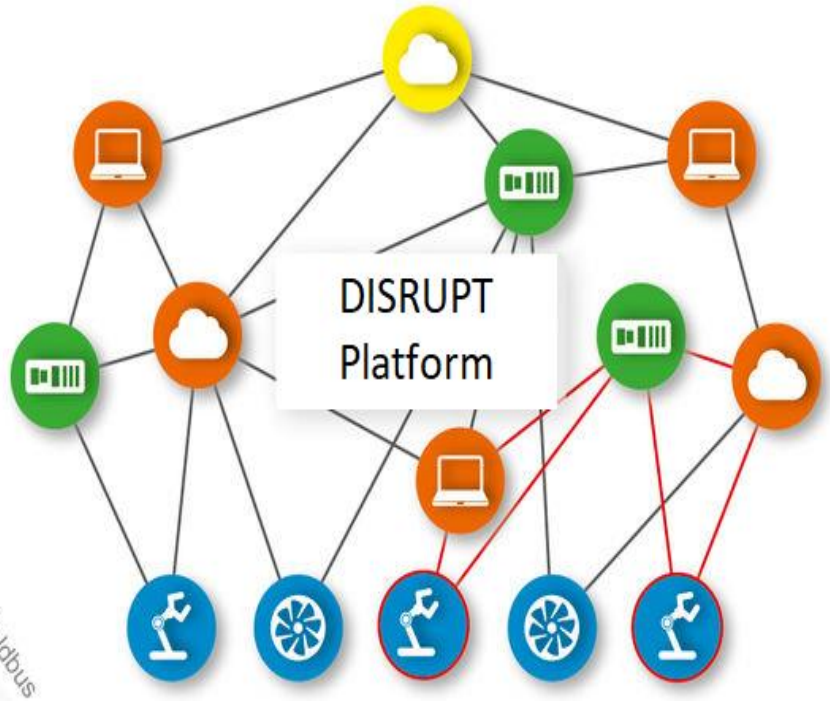
CRF, Automotive

Reconfiguration Scaling and Optimisation

ARCELIK, Home Appliances

Flattening the pyramid/hierarchy
(adjacency not necessary for data exchange)





Some observations re: compute-intensive aspects

- Computation-on-demand
  - Specific events may require significant compute capacity that needs to be made available at short notice.
  - Demands are tightly linked to the state of the system.
- Data is a bottleneck (this is not new!)
  - Adjust the rate of data production in relation to state (and anticipated risks) in the system.
- Continuous hardware/software re-configuration
  - Not only a question of achieving ‘best’ performance but a question of accommodating various user and system constraints striking a balance between different criteria.

‘Adjustable’ computation

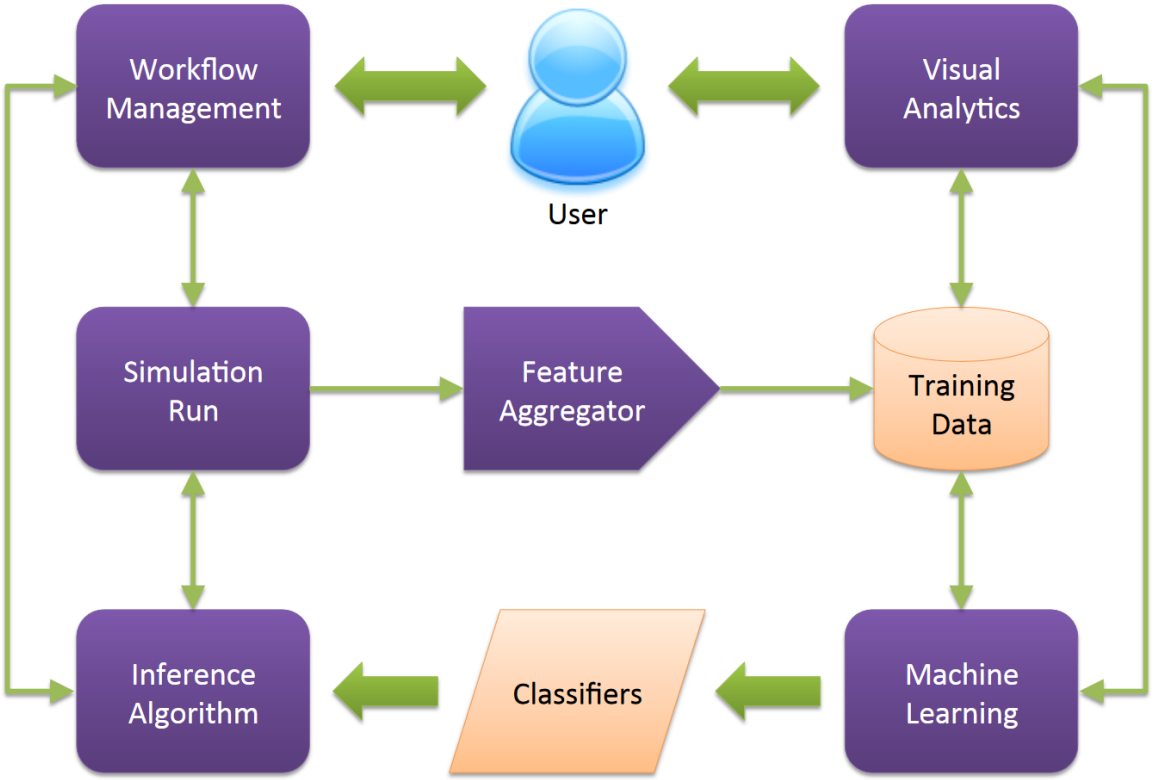


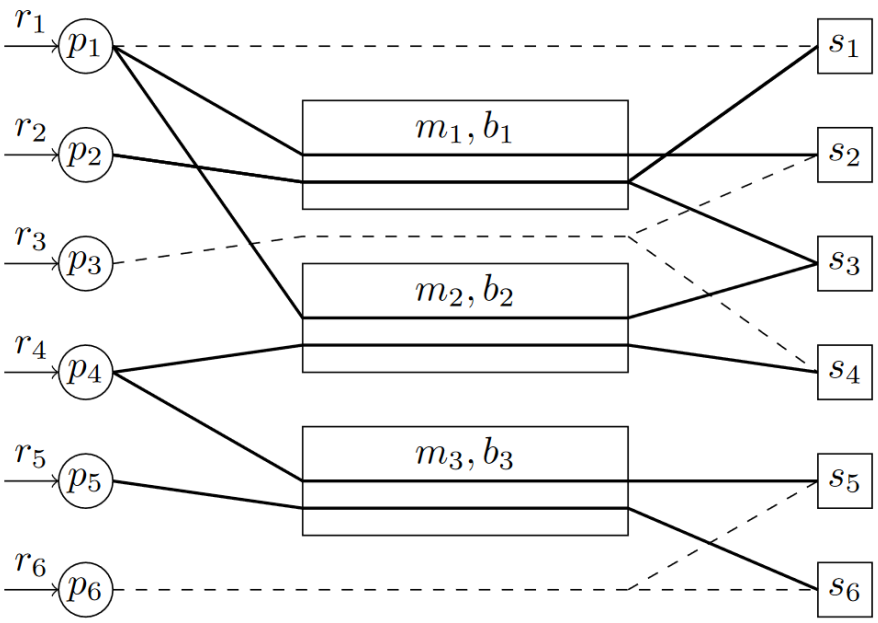
Figure 1: A high-level overview of an infrastructure for integrating machine learning into HPC simulations.

[Motivated by CFD and Lagrangian / Eulerian Simulations]

Source: R. F. da Silva et al, “A Characterization of Workflow Management Systems for Extreme-Scale Applications”, FGCS, 2018

Adjustable data production rate

- Publishers $p_{1,...}$ produce data at a rate $r_{1,...}$
- Subscribers $s_{1,...}$ aim to achieve a certain level of satisfaction
- Machines $m_{1,...}$ have a certain amount of bandwidth $b_{1,...}$
- Goal: adjust rate of production to maximize satisfaction.

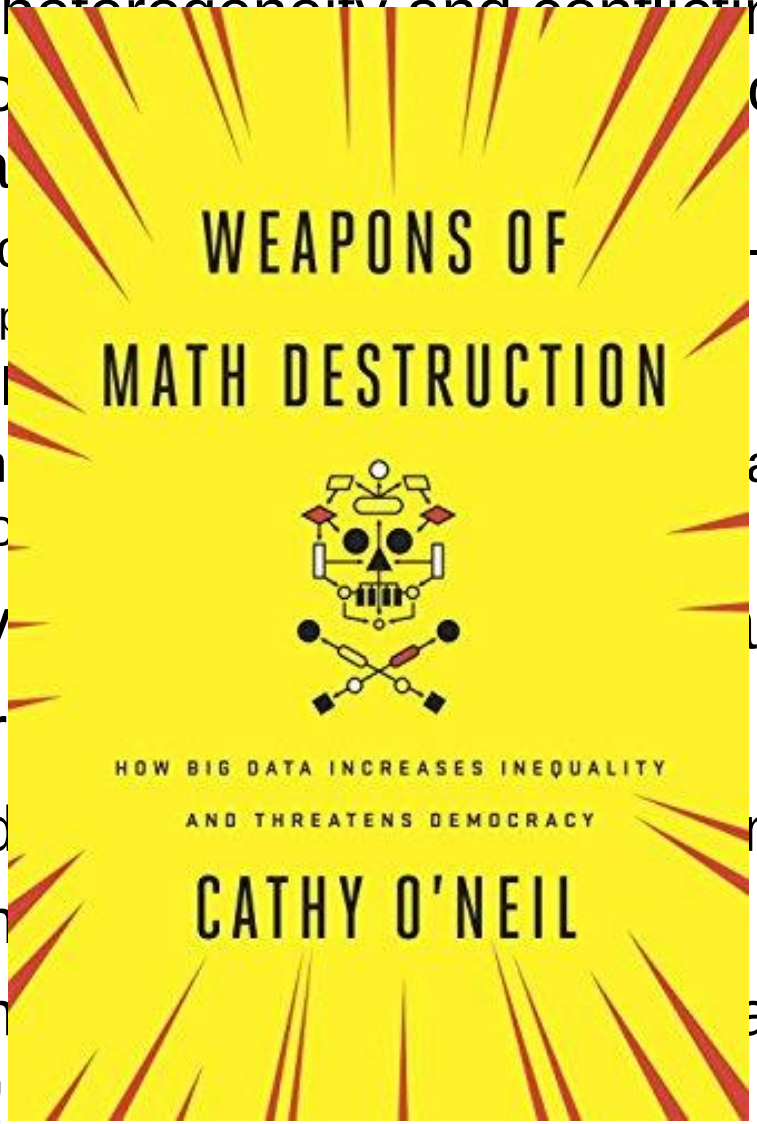


Thomas Lambert and Rizos Sakellariou. Allocation of Publisher/Subscriber Data Links on a Set of Virtual Machines. In *IEEE CLOUD 2018*.

Finally…

Conclusion

- With the growing h…ng costs, the challenge is to loc…ost one has to pay to achieve a certa…
  - Suboptimal soluti…-cost
    - This affects both p…
  - Pareto front – mul… into account all aspects
  - Many examples h…al space, in reality the problem relates to…
- Need to go bey…ches
- Implications for…
  - Cloud, Fog, Ed…mputing
  - Optimization th…
  - Software engin…ation
  - …and beyond!



Thank you!

Acknowledgements: all past students and collaborators and people who influenced thinking over the years!