MANCHESTER 1824

The University of Manchester

Scheduling Workflows with Energy Constraints

Presented by Rizos Sakellariou but thanks to students and collaborators:

Ilia Pietri, Henan Zhao, Ewa Deelman and more

Largely based on a paper to appear at the 3rd Workshop on Power-Aware Algorithms, Systems and Architectures (in conjunction with ICPP 2014)

Scheduling does matter!

Schedule: "*A plan for performing work or achieving an objective, specifying the order and allotted time for each part*" (http://www.thefreedictionary.com)

According to one view, Computer Science is the art of realising successive layers of abstraction

Scheduling: the constituent parts:

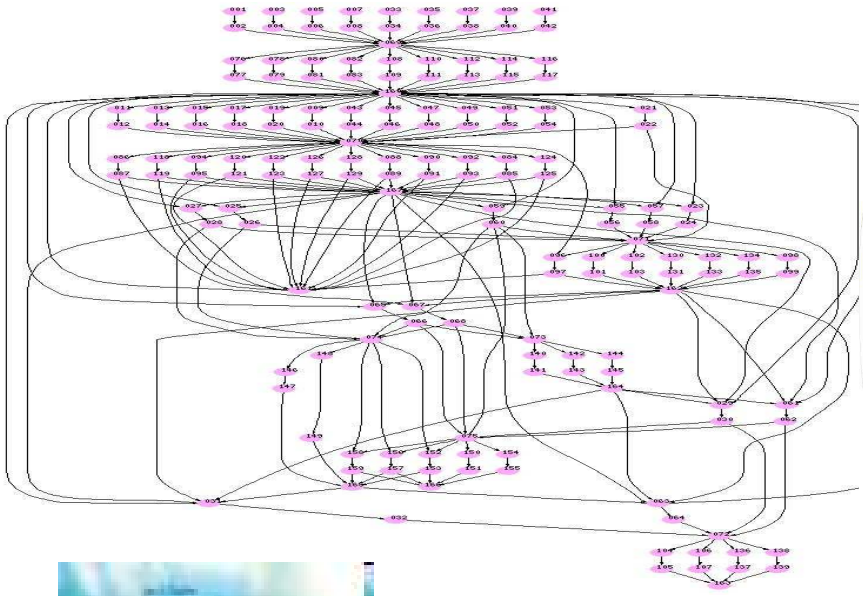
- Work
- Resources
- Objective(s)

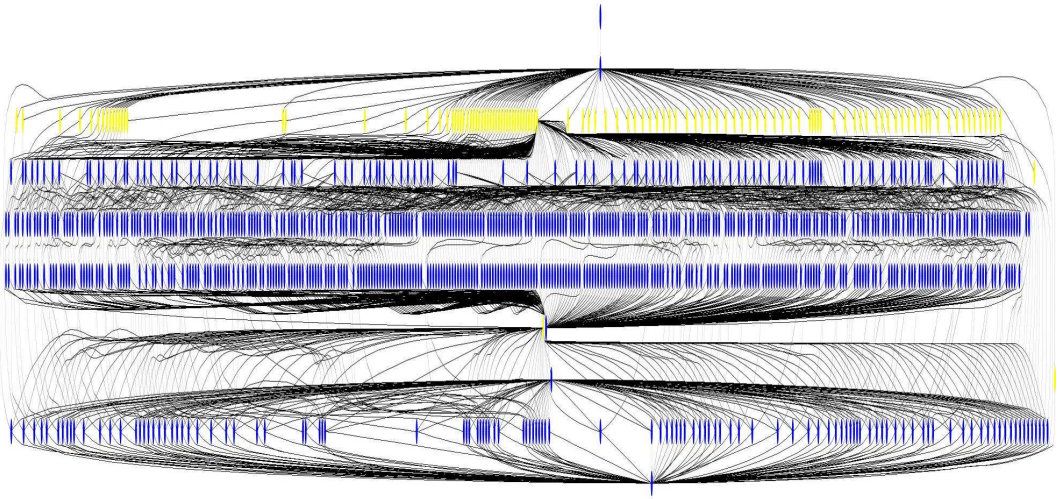
In this work…

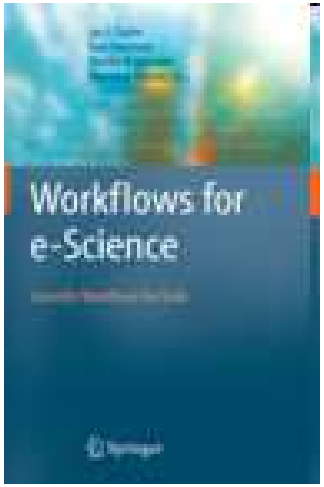
- **Work**
  - Scientific workflows
    - DAG (nodes: work, edges: communication)
- **Resources**
  - Cloud resources
- **Objective**
  - Complete execution of the workflow by a certain deadline on a number of resources
  - Minimize overall energy consumption

Scientific Workflows

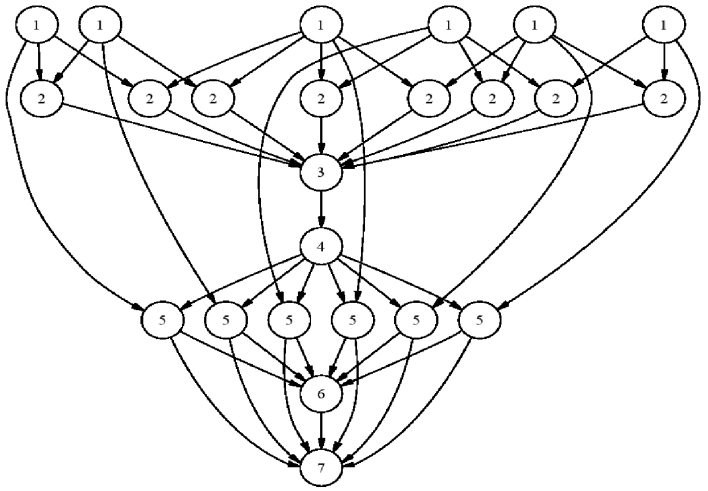
Many interesting scientific applications can be represented by DAGs





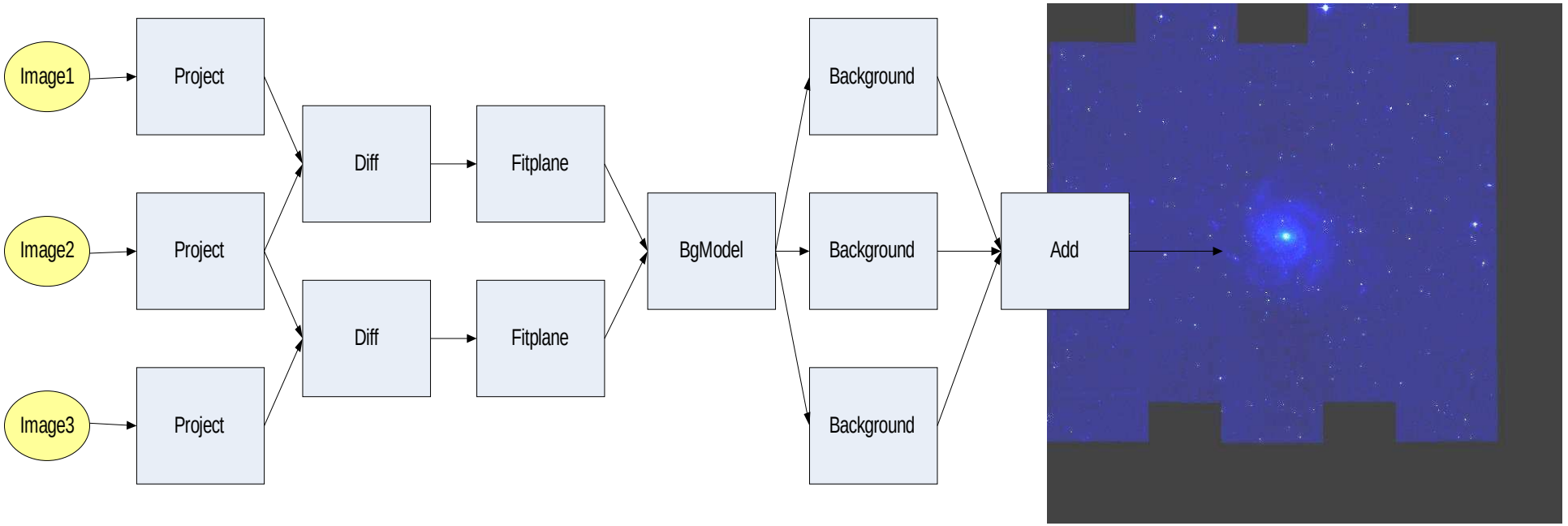


I. Taylor, E. Deelman, D. Gannon: Workflows for e-Science. Springer, 2007



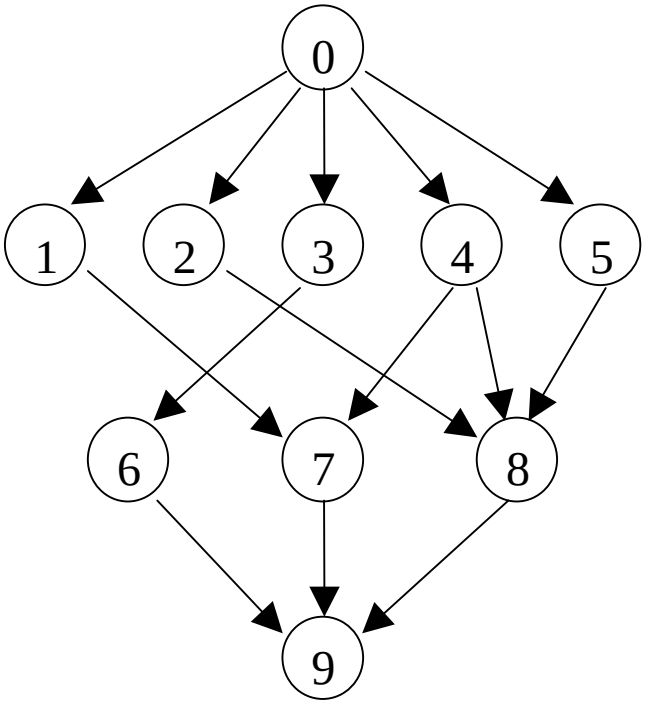
The Montage Workflow

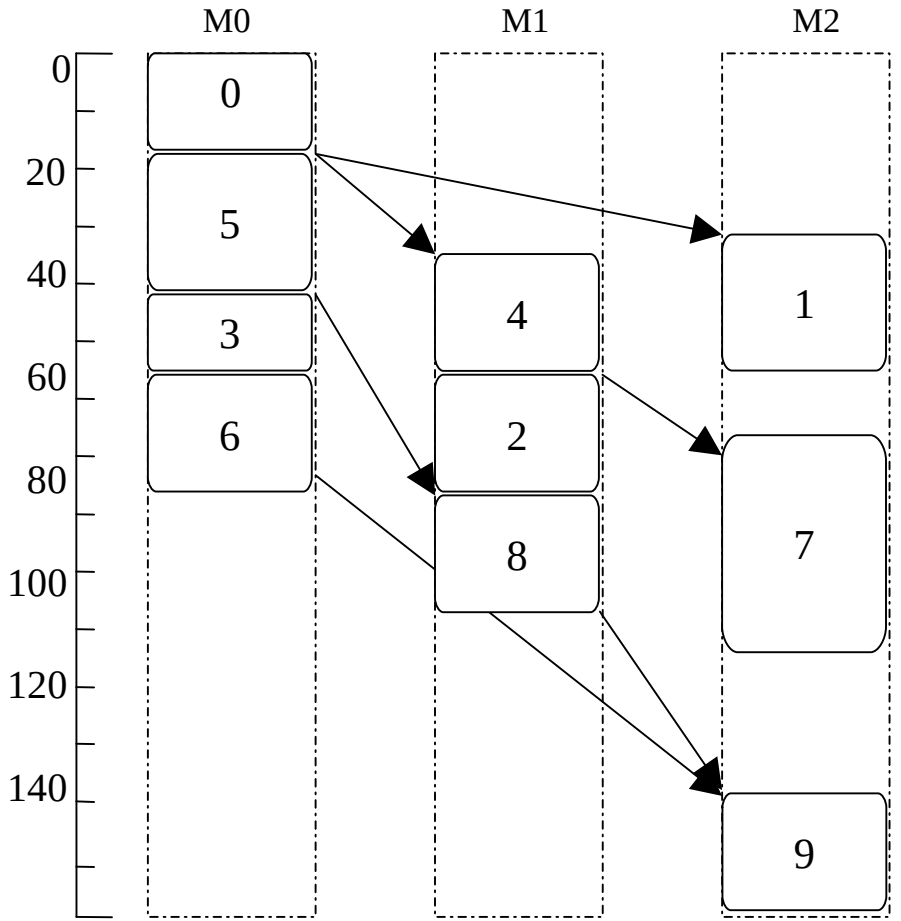
- Montage example: Generating science-grade mosaics of the sky (Bruce Berriman, Caltech)
- http://montage.ipac.caltech.edu/



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

A DAG, a schedule, and an old idea

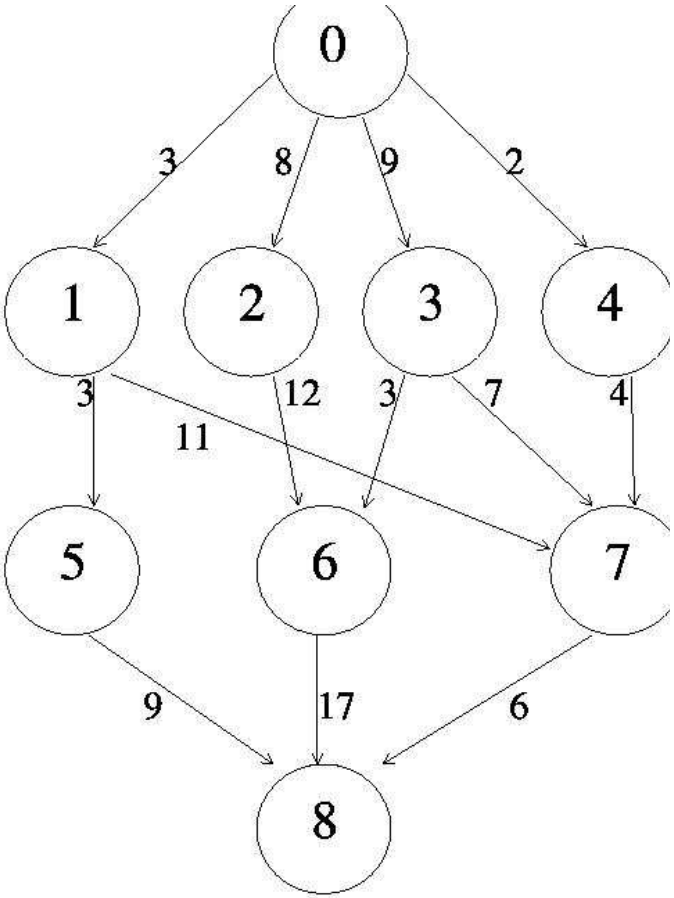


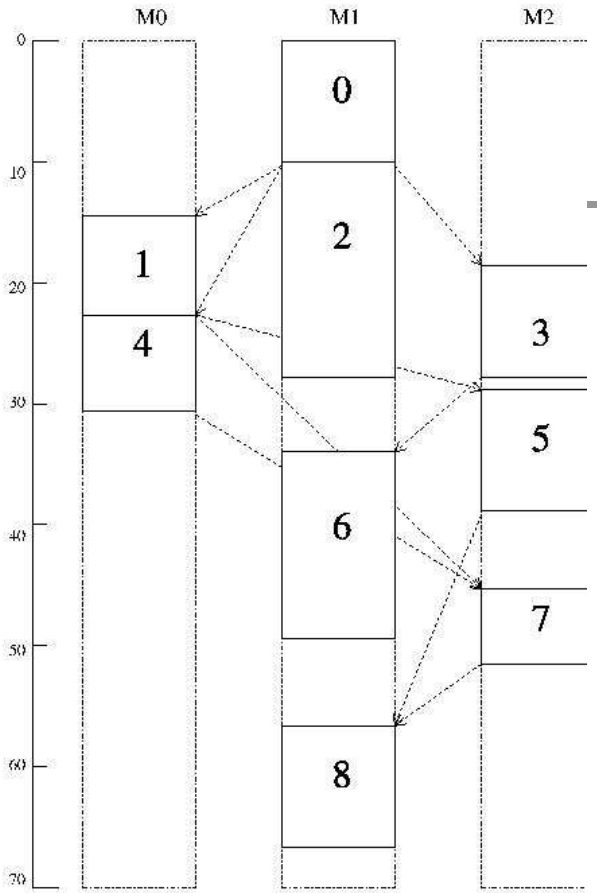


Characterize the Schedule

- ***Spare time*** indicates the maximum time that a node, *i*, may delay without affecting the start time of an immediate successor, *j*.
  - A node *i* with an immediate successor *j* on the DAG: *spare(i,j) = Start_Time(j) – Data_Arrival_Time(i,j)*
  - A node *i* with an immediate successor *j* on the same machine: *spare(i,j) = Start_Time(j) – Finish_Time(i)*
  - The minimum of the above for all successors of task *i* is the: ***Spare time*** of task *i*.

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.





Example

DAT(4,7)=40.5, ST(7)=45.5; hence, spare(4,7) = 5

FT(3)=28, ST(5)=29.5; hence, spare(3,5) = 1.5

DAT: Data_Arrival_Time, ST: Start_Time, FT: Finish_Time

Characterize the schedule (cont.)

- ***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)

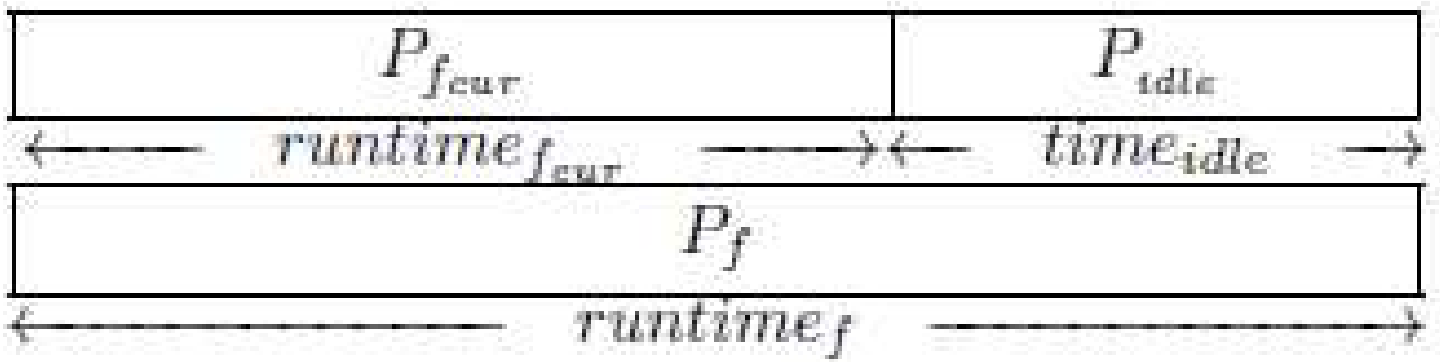
The idea

- Given a schedule (mapping of tasks onto machines)
- Given that (according to the schedule) many tasks will always have some slack
- Why don't we try to lower the frequency of the tasks with a slack so that they run up to the slack (or they use as much as possible)?
  - This should not affect overall makespan

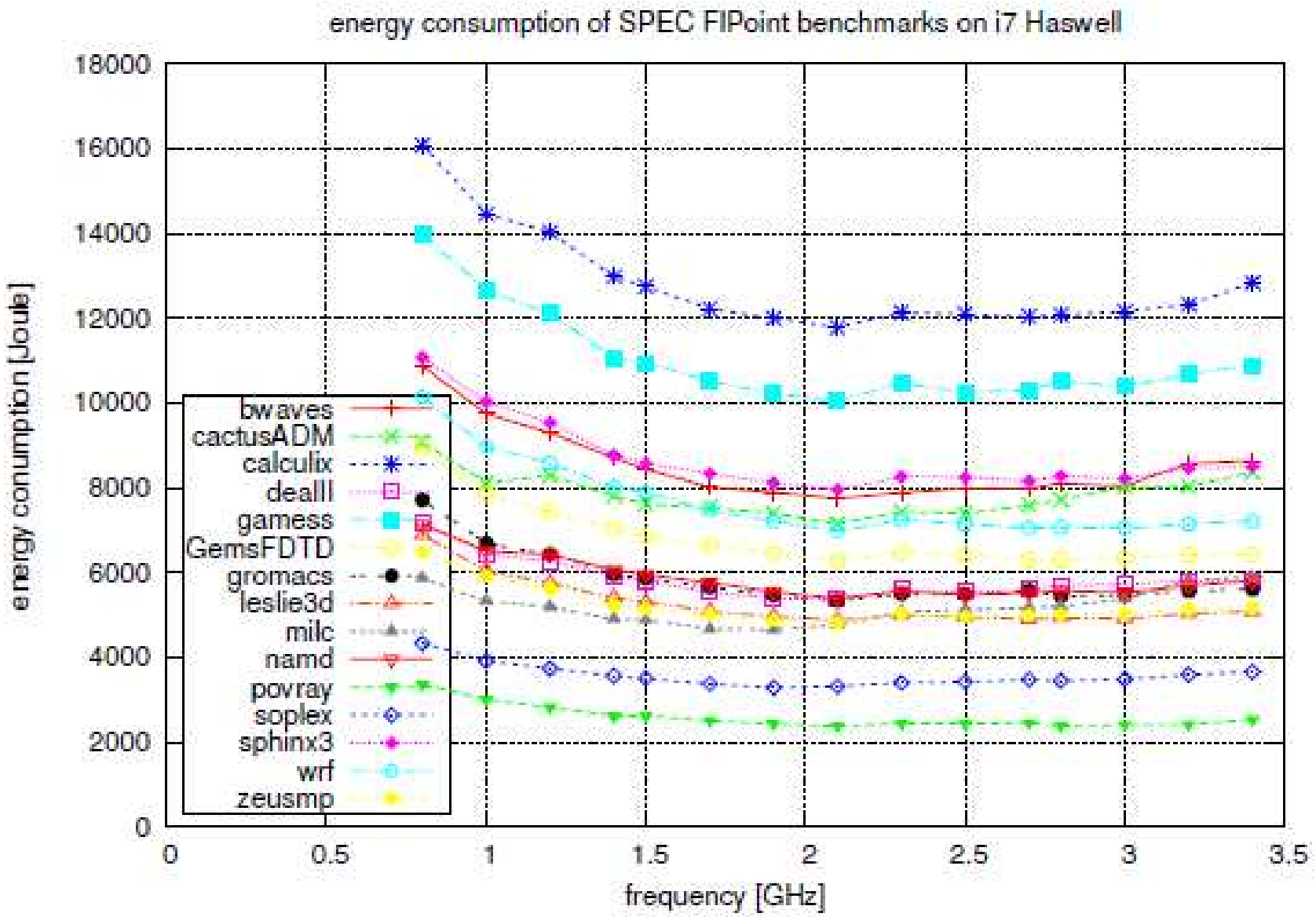
What is the catch here?

Lowering frequency does not mean we save energy!

- Running at a lower frequency will require less power, but it will take longer!
- Remember: energy is power $\times$ time



Thanks to Thomas Rauber (1st day)



In addition…

- The workflow (DAG) is a collection of tasks
- We need to take into account the energy vs frequency behaviour of each task and overall (for the whole workflow)
- Different tasks will exhibit different behaviour
- If we try to apply frequency scaling for one task we have to pay some cost for switching frequency (small, but…)

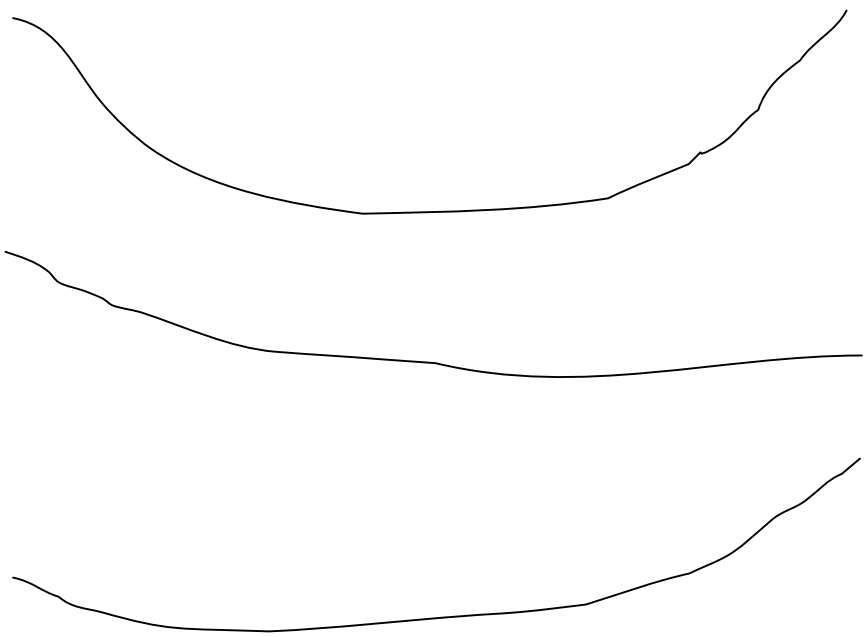
The idea

- 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: (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.



The models

- Power:

  $P_f = P_{base} + P_{dif} (f - f_{base})^3 / f_{base}$

  (Pierson & Casanova, Euro-Par 2011)

- Task execution time:

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

  (Etinski, Corbalan, Labarta, Valero, JPDC 2012)

Evaluation

- Baseline algorithms
  - EES[1]
  - HEFT
- Processor characteristics
  - P_{base}=152W
  - P_{dif}=15.39W
  - P_{idle}=60%P_{fmax}
  - Threshold: 0.01%
- Synthetic data of 3 real workflows, 100 tasks each
  - LIGO
  - SIPHT
  - Montage

fmode	**0**	**1**	**2**	**3**	**4**
Slow		1800	2000	2200	2400
Fast	1800	2000	2200	2400	2600

[2] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *Proceedings of the 12th IEEE/ACM CCGrid*. IEEE, 2012, pp. 781–786.

Results/Comments

- Simulation results assessing ESFS (and comparing with EES and HEFT) to be presented at PASA@ICPP in September.
- Comments/Criticism:
  - Simulation is not the real thing
  - Processor power is not where most of the power goes
  - Power when idle may be much less than 60% of power_max
  - Power consumption may not be constant for some frequency

Conclusion

- Energy-aware scheduling requires a good understanding of underlying energy-related aspects (or parameters), but there is lots of scope for interesting, scheduling-related problems.
- To appear at PASA@ ICPP
- (and a formula/problem): for a given *n* what is the smallest *k* so that there is an integer solution of:

$$x_1^n + x_2^n + x_3^n + \ldots + x_k^n = z^n$$

www.sucreproject.eu