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

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:
    \[
    \text{spare}(i,j) = \text{Start\_Time}(j) - \text{Data\_Arrival\_Time}(i,j)
    \]
  - A node \( i \) with an immediate successor \( j \) on the same machine:
    \[
    \text{spare}(i,j) = \text{Start\_Time}(j) - \text{Finish\_Time}(i)
    \]
  - The minimum of the above for all successors of task \( i \) is the: **Spare time** of task \( i \).

Example

$$\text{DAT}(4,7)=40.5, \text{ST}(7)=45.5; \text{hence, spare}(4,7) = 5$$

$$\text{FT}(3)=28, \text{ST}(5)=29.5; \text{hence, spare}(3,5) = 1.5$$

\text{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.
  
  \[
  \text{Slack}(i) = \min(\text{slack}(j) + \text{spare}(i,j)), \text{ for all successor nodes } j \text{ (both on the DAG and the machine)}
  \]
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 (1\textsuperscript{st} 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...)

13
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:

\[ P_f = P_{\text{base}} + P_{\text{dif}} \left( f - f_{\text{base}} \right)^3 / f_{\text{base}} \]

(Pierson & Casanova, Euro-Par 2011)

• Task execution time:

\[ \text{Runtime} = (1 + \beta \left( f_{\text{max}} / f - 1 \right) ) \text{runtime}_{f_{\text{max}}} \]

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

• Baseline algorithms
  – EES[1]
  – HEFT

• Processor characteristics
  • $P_{\text{base}} = 152\text{W}$
  • $P_{\text{dif}} = 15.39\text{W}$
  • $P_{\text{idle}} = 60\% P_{\text{fmax}}$
  • Threshold: 0.01%

• Synthetic data of 3 real workflows, 100 tasks each
  – LIGO
  – SIPHT
  – Montage

<table>
<thead>
<tr>
<th>fmode</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow</td>
<td></td>
<td>1800</td>
<td>2000</td>
<td>2200</td>
<td>2400</td>
</tr>
<tr>
<td>Fast</td>
<td>1800</td>
<td>2000</td>
<td>2200</td>
<td>2400</td>
<td>2600</td>
</tr>
</tbody>
</table>

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