Scheduling independent moldable tasks to minimize the energy consumption

Anne Benoit, Louis-Claude Canon, Redouane Elghazi, Pierre-Cyrille Héam

LIP, Ecole Normale Supérieure de Lyon, France
FEMTO-ST, Univ. Franche-Comté

Anne.Benoit@ens-lyon.fr

http://graal.ens-lyon.fr/~abenoit/

Fréjus Scheduling Workshop, June 10, 2022

A crucial issue: Energy consumption

"The internet begins with coal"

- Nowadays: more than 90 billion kilowatt-hours of electricity a year; requires 34 giant (500 megawatt) coal-powered plants, and produces huge CO_2 emissions
- Explosion of artificial intelligence; AI is hungry for processing power! Need to double data centers in next four years
  $\rightarrow$ how to get enough power?
- Failures: Redundant work consumes even more energy

Energy and power awareness $\rightsquigarrow$ crucial for both environmental and economical reasons

Yet another little scheduling problem!

We start from a classical scheduling problem for moldable tasks:

- p identical processors;
- n independent *moldable* tasks with task i executed on j processors having a known execution time of $t_{i,j}$;
- for each *moldable* task, the number of processors j must be chosen once at the beginning of the execution, as opposed to *rigid* tasks, for which the number of processors for each task is given.

Scheduling moldable tasks – Example

Example instance, with three tasks and two processors:

Task 1: $t_{1,1} = 6$ or $t_{1,2} = 5$

Task 2: $t_{2,1} = 5$ or $t_{2,2} = 3$

Task 3: $t_{3,1} = 8$ or $t_{3,2} = 4$

Example solution with makespan $C_{max} = 10$ (optimal):

Scheduling moldable tasks – Example

Example instance, with three tasks and two processors:

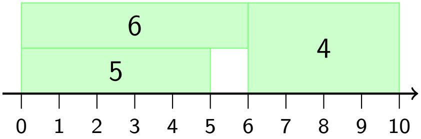
Task 1: $t_{1,1} = 6$ or $t_{1,2} = 5$

Task 2: $t_{2,1} = 5$ or $t_{2,2} = 3$

Task 3: $t_{3,1} = 8$ or $t_{3,2} = 4$

Example solution with makespan $C_{max} = 10$ (optimal):

Processor 1:
Processor 2:



Scheduling moldable tasks

This problem has been studied for the minimization of the makespan.

- It is NP-hard.
- There exist approximation algorithms for makespan minimization.
- The simpler problem with the additional constraint that all tasks must begin simultaneously is also studied (single shelf).

What can we do to minimize the energy consumption of such a schedule?

Scheduling moldable tasks

This problem has been studied for the minimization of the makespan.

- It is NP-hard.
- There exist approximation algorithms for makespan minimization.
- The simpler problem with the additional constraint that all tasks must begin simultaneously is also studied (single shelf).

What can we do to minimize the energy consumption of such a schedule?

Introducing the speed

Processors with DVFS (Dynamic Voltage and Frequency Scaling):

- Static power P_{stat} when operating
- Discrete set $S = \{s_1, s_2, \dots, s_k\}$ of possible speeds (or frequencies);
  $s_{\min} = s_1$, $s_{\max} = s_k$
- Continuous model: $S = \mathbb{R}_+^*$
- One speed per task
- Two different tasks scheduled on a same processor can be executed at different frequencies.

Now, we can formulate the problem of minimizing the energy of a schedule for moldable tasks.

Introducing the speed

Processors with DVFS (Dynamic Voltage and Frequency Scaling):

- Static power P_{stat} when operating
- Discrete set $S = \{s_1, s_2, \ldots, s_k\}$ of possible speeds (or frequencies); $s_{\min} = s_1$, $s_{\max} = s_k$
- Continuous model: $S = \mathbb{R}_+^*$
- One speed per task
- Two different tasks scheduled on a same processor can be executed at different frequencies.

Now, we can formulate the problem of minimizing the energy of a schedule for moldable tasks.

Problem formulation

We consider the MinE-Mold problem, with as input:

- p identical processors with a static power P_{stat} and a set S of possible speeds;
- n moldable tasks $\{T_1, T_2, \ldots, T_n\}$ with execution profiles $(w_{i,j})_{i \in [1,n], j \in [1,p]}$ (total work required to execute T_i on j processors);
- the execution time of T_i executed on j processors at speed s is $t_{i,j,s} = \frac{w_{i,j}}{j \times s}$.

Objective function: minimize energy consumption, which is the sum of two parts:

$$E = E_{stat} + \sum_{i \leq n} E_{i,dyn}$$

Static energy E_{stat} consumed by the processors: $E_{stat} = p \times C_{max} \times P_{stat}$, where C_{max} is the total powered up duration of the platform

Dynamic energy $E_{i,dyn}$ consumed by T_i executed on j processors at speed s: $E_{i,dyn} = j \times t_{i,j,s} \times s^{\alpha}$, where α is a constant usually between 2 and 3

Problem formulation

We consider the MinE-Mold problem, with as input:

- p identical processors with a static power P_{stat} and a set S of possible speeds;
- n moldable tasks $\{T_1, T_2, \ldots, T_n\}$ with execution profiles $(w_{i,j})_{i\in[1,n],j\in[1,p]}$ (total work required to execute T_i on j processors);
- the execution time of T_i executed on j processors at speed s is $t_{i,j,s} = \frac{w_{i,j}}{j\times s}$.

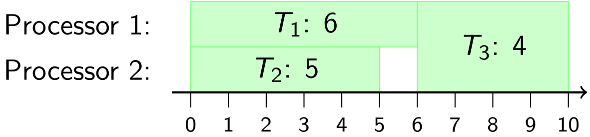
Objective function: minimize energy consumption, which is the sum of two parts:

$$E = E_{stat} + \sum_{i\leq n} E_{i,dyn}$$

Static energy E_{stat} consumed by the processors: $E_{stat} = p \times C_{max} \times P_{stat}$, where C_{max} is the total powered up duration of the platform

Dynamic energy $E_{i,dyn}$ consumed by T_i executed on j processors at speed s: $E_{i,dyn} = j \times t_{i,j,s} \times s^{\alpha}$, where α is a constant usually between 2 and 3

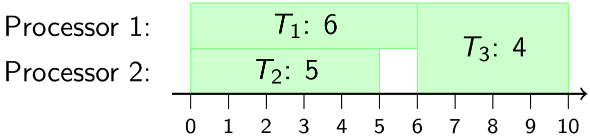
Scheduling with different speeds – Example

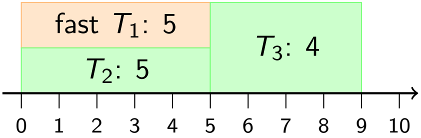


smaller makespan → less static energy
higher processing speed → more dynamic energy

same makespan → same static energy
lower processing speed → lower dynamic energy

Scheduling with different speeds – Example

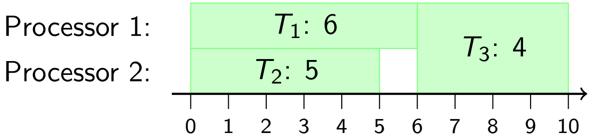


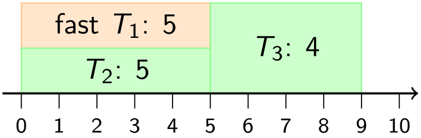


smaller makespan → less static energy
higher processing speed → more dynamic energy

same makespan → same static energy
lower processing speed → lower dynamic energy

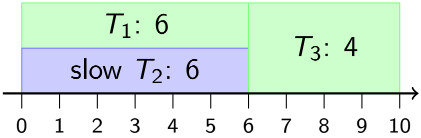
Scheduling with different speeds – Example





smaller makespan → less static energy

higher processing speed → more dynamic energy



same makespan → same static energy

lower processing speed → lower dynamic energy

Contributions

- Formulation of several problems of energy minimization for scheduling independent moldable tasks;
- Proof that MINE-MOLD is NP-complete;
- Proof of multiple approximation ratios for different algorithms solving MINE-MOLD, the approximation ratios are between 2 and 3 depending on the algorithm;
- Empirical study comparing various algorithms.

NP-completeness of MINE-MOLD

Theorem

The decision problem associated to MINE-MOLD is NP-complete.

Proof:

- Reduction from 3-PARTITION, with each processor corresponding to a different subset.
- Ensure that each task is executed on a single processor at speed 1 (both with discrete and continuous speeds)

Note that if processors can be turned off, the problem becomes trivial: use a single processor for each task and use optimal speed ($s^{opt} = \sqrt[\alpha]{\frac{P_{stat}}{\alpha-1}}$ with continuous speeds, or try all possibilities in the discrete model) $\Rightarrow$ Lower bound!

NP-completeness of MINE-MOLD

Theorem

The decision problem associated to MINE-MOLD *is NP-complete.*

Proof:

- Reduction from 3-PARTITION, with each processor corresponding to a different subset.
- Ensure that each task is executed on a single processor at speed 1 (both with discrete and continuous speeds)

Note that if processors can be turned off, the problem becomes trivial: use a single processor for each task and use optimal speed ($s^{opt} = \sqrt[\alpha]{\frac{P_{stat}}{\alpha-1}}$ with continuous speeds, or try all possibilities in the discrete model) ⇒ Lower bound!

Approximation algorithms

We first provide two approximation algorithms for the rigid case MinE-Rig, where tasks have a predefined number of processors:

- ListBased, a list-based algorithm that lists the tasks in some order, and then assigns them greedily;
- ShelfBased, a shelf-based algorithm that creates batches of tasks to be executed one after the other, with each task of a batch starting at the same time.

We then provide a way to transform approximation algorithms for the rigid case into approximation algorithms for the moldable case.

Approximation ratios with discrete speeds

In the following table, we present the approximation ratios for two algorithms for two problem variants, with discrete speeds:

Algorithm	MinE-Mold	MinE-Mold with the same speed for all tasks
ListBased	3-approximation	2-approximation
ShelfBased	3-approximation	3-approximation

The proofs for these results are based on:

- existing ratios for the makespan;
- the fact that among all the schedules these algorithms will try, static and dynamic energies will be well balanced.

Sophisticated proofs, check the paper for details!

Approximation ratios with discrete speeds

In the following table, we present the approximation ratios for two algorithms for two problem variants, with discrete speeds:

Algorithm	MinE-Mold	MinE-Mold with the same speed for all tasks
ListBased	3-approximation	2-approximation
ShelfBased	3-approximation	3-approximation

The proofs for these results are based on:

- existing ratios for the makespan;
- the fact that among all the schedules these algorithms will try, static and dynamic energies will be well balanced.

Sophisticated proofs, check the paper for details!

How do we choose $s_i \in S$

- Different algorithms have different behaviors:
  - some first choose the speeds and then schedule (usually allowing different speeds s_i for different processors)
  - others schedule and then choose speeds (usually taking the same speed s for all processors)
- Allowing different speeds for different tasks only marginally changes the energy consumption.

How do we choose $s_i \in S$

- Different algorithms have different behaviors:
    - some first choose the speeds and then schedule (usually allowing different speeds s_i for different processors)
    - others **schedule and then choose speeds** (usually taking the same speed s for all processors)
- Allowing different speeds for different tasks only marginally changes the energy consumption.

Computing the speed $s \in S$ for a schedule

We can compute a schedule at speed $s = 1$, and then compute the optimal speed of this schedule:

$$s^{\mathrm{OPT}} = \sqrt[\alpha]{\frac{p \times C_{max,s=1}}{(\alpha - 1) \times W} \times P_{stat}}$$

where W is the sum of w_{i,p_i} over all tasks.

And if $s^{\mathrm{OPT}} \notin S$ (discrete speeds), then we take one of the closest possibilities.

Computing the speed $s \in S$ for a schedule

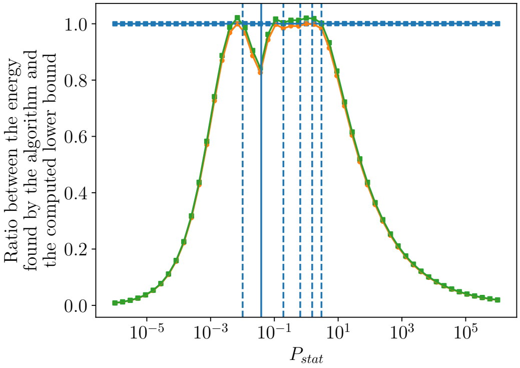
We can compute a schedule at speed $s = 1$, and then compute the optimal speed of this schedule:

$$s^{\mathrm{OPT}} = \sqrt[\alpha]{\frac{p \times C_{max,s=1}}{(\alpha - 1) \times W} \times P_{stat}}$$

where W is the sum of w_{i,p_i} over all tasks.

And if $s^{\mathrm{OPT}} \notin S$ (discrete speeds), then we take one of the closest possibilities.

Discrete vs continuous speeds: Intel Xscale



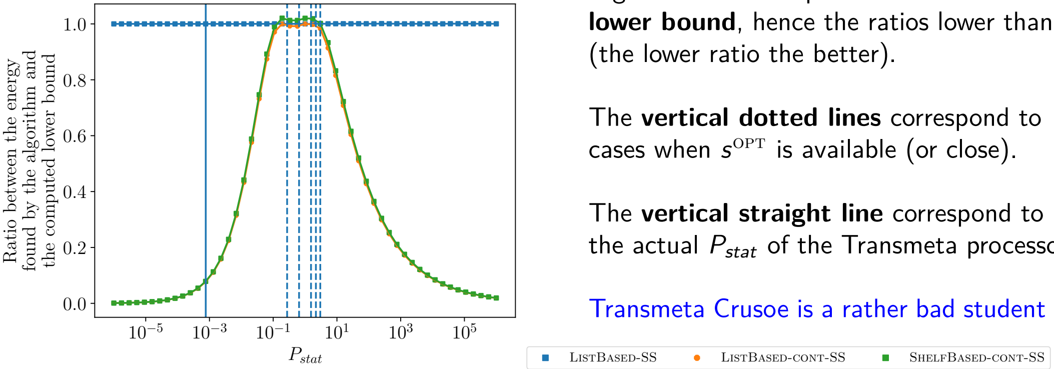
Algorithms are compared to the **discrete lower bound**, hence the ratios lower than 1 (the lower ratio the better).

The **vertical dotted lines** correspond to cases when s^{OPT} is available (or close).

The **vertical straight line** correspond to the actual P_{stat} of the Intel processor.

Intel Xscale is a rather good student ☺

Discrete vs continuous speeds: Transmeta Crusoe



Algorithms are compared to the **discrete lower bound**, hence the ratios lower than 1 (the lower ratio the better).

The **vertical dotted lines** correspond to cases when s^{OPT} is available (or close).

The **vertical straight line** correspond to the actual P_{stat} of the Transmeta processor.

Transmeta Crusoe is a rather bad student ☹

Bounding s^{OPT}

One of our algorithms ensures that, as long as no task dominates the schedule, then we actually have:

$$W \leq p \times C_{max,s=1} \leq 2 \times W$$

Also, recall that

$$s^{\mathrm{OPT}} = \sqrt[\alpha]{\frac{p \times C_{max,s=1}}{(\alpha-1) \times W} \times P_{stat}}$$

So we get

$$\sqrt[\alpha]{\frac{P_{stat}}{2 \times (\alpha-1)}} \leq s^{\mathrm{OPT}} \leq \sqrt[\alpha]{\frac{P_{stat}}{\alpha-1}}$$

Bounding s^{OPT}

Hence, we get the following bounds on s^{OPT}:

Processor	α	P_{stat}	Available speeds S	Interval of s^{OPT}
General Case	$2 \leq \alpha \leq 3$	$P_{stat} \in \mathbb{R}_+$	$\{s_1, s_2, \ldots, s_k\}$	$\left[\sqrt[\alpha]{\frac{P_{stat}}{2\times(\alpha-1)}}, \sqrt[\alpha]{\frac{P_{stat}}{\alpha-1}}\right]$
Intel Xscale	3	$\frac{60}{1550}$	$\{0.15, 0.4, 0.6, 0.8, 1\}$	$[0.21, 0.27]$
Transmeta Crusoe	3	$\frac{44}{57560}$	$\{0.45, 0.6, 0.8, 0.9, 1\}$	$[0.058, 0.073]$

Consequences of having $s^{\mathrm{OPT}} \in S$

Having access to s^{OPT} actually has an impact on the theoretical bounds of our algorithms (and it is the case with continuous speeds)

Algorithm	Approximation ratio if $s^{\mathrm{OPT}} \notin S$	Approximation ratio if $s^{\mathrm{OPT}} \in S$
ListBased	2-approximation	$2^{1-\frac{1}{\alpha}}$-approximation (e.g., $2^{1-\frac{1}{3}} \approx$ **1.59**)
ShelfBased	3-approximation	$3^{1-\frac{1}{\alpha}}$-approximation (e.g., $3^{1-\frac{1}{3}} \approx$ **2.08**)

Conclusions

- In terms of scheduling, we are already very close to the optimal energy consumption
- The best improvement we found would be to lower speeds beyond what the studied processors allow (15% to 90% energy gain depending on the processor)
- Having access to the correct speed even lowers the approximation ratio of the proposed algorithms

Future working directions

- Find more recent processor descriptions
- Conduct experiments on real HPC systems
- Extend the analysis to other energy models (e.g., change the energy formula, introduce a cost of time and energy for any speed change, ...)

Conclusions

- In terms of scheduling, we are already very close to the optimal energy consumption
- The best improvement we found would be to lower speeds beyond what the studied processors allow (15% to 90% energy gain depending on the processor)
- Having access to the correct speed even lowers the approximation ratio of the proposed algorithms

Future working directions

- Find more recent processor descriptions
- Conduct experiments on real HPC systems
- Extend the analysis to other energy models (e.g., change the energy formula, introduce a cost of time and energy for any speed change, . . .)