Energy-aware algorithms

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr

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

CR02 - 2016/2017

Speed models for DVFS

		When can we change speed?	
		Anytime	**Beginning of tasks**
Type of speeds	$[s_{min}, s_{max}]$	CONTINUOUS	-
	$\{s_1, \ldots, s_m\}$	VDD-HOPPING	DISCRETE, INCREMENTAL

- CONTINUOUS: great for theory
- Other "discrete" models more realistic
- VDD-HOPPING simulates CONTINUOUS
- INCREMENTAL is a special case of DISCRETE with equally-spaced speeds: for all $1 \leq q < m$, $s_{q+1} - s_q = \delta$

Complexity results

Minimizing energy with fixed mapping on p processors:

- CONTINUOUS: Polynomial for some special graphs, geometric optimization in the general case
- DISCRETE: NP-complete (reduction from 2-partition); approximation algorithm
- INCREMENTAL: NP-complete (reduction from 2-partition); approximation algorithm
- VDD-HOPPING: Polynomial (linear programming)

General problem: geometric programming

Reminder

For each task T_i,

- w_i is its size/work
- s_i is the speed of the processor that has task T_i assigned to
- t_i is the time when the computation of T_i ends

Objective function

$$\text{Minimize} \quad \sum_{i=1}^{n} s_i^2 \times w_i$$
$$\text{subject to (i) } t_i + \frac{w_j}{s_j} \leq t_j \text{ for each } (T_i, T_j) \in E$$
$$\text{(ii) } t_i \leq D \text{ for each } T_i \in V$$

Results for continuous speeds

- MinEnergy(G,D) can be solved in polynomial time when G is a tree
- MinEnergy(G,D) can be solved in polynomial time when G is a series-parallel graph (assuming $s_{max} = +\infty$)

TODO: Prove the lemma for forks and joins to prove that MinEnergy(G,D) can be solved in polynomial time in this case (we just need to find s_0).

Linear program for VDD-HOPPING

Definition

G, n tasks, D deadline;

$s_1, \ldots, s_m$ be the set of possible processor speeds;

t_i is the finishing time of the execution of task T_i;

$\alpha_{(i,j)}$ is the *time* spent at speed s_j for executing task T_i

This makes us a total of $n(m+1)$ variables for the system.
Note that the total execution time of task T_i is $\sum_{j=1}^{m} \alpha_{(i,j)}$.

The objective function is:

$$\min \left(\sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{(i,j)} s_j^3 \right)$$

Linear program for VDD-HOPPING

The constraints are:

- $\forall 1 \leq i \leq n,\ t_i \leq D$: the deadline is not exceeded by any task;
- $\forall 1 \leq i, i' \leq n$ s.t. $T_i \rightarrow T_{i'},\ \ t_i + \sum_{j=1}^{m} \alpha_{(i',j)} \leq t_{i'}$: a task cannot start before its predecessor has completed its execution;
- $\forall 1 \leq i \leq n,\ \sum_{j=1}^{m} \alpha_{(i,j)} \times s_j \geq w_i$: task T_i is completely executed;
- $\forall 1 \leq i \leq n,\ t_i \geq \sum_{j=1}^{m} \alpha_{(i,j)}$: each task cannot finish until all work is done.

NP-completeness for discrete speed models

Theorem

With the INCREMENTAL model (and hence the DISCRETE model), finding the speed distribution that minimizes the energy consumption while enforcing a deadline D is NP-complete.

Proof: Reduction from 2-PARTITION,

- 1 processor, n independent tasks of weight (a_i)
- 2 speeds : $s_1 = 1$, $s_2 = 2$ (increment of 1)
- $D = 3T/2$ (where $T = \frac{1}{2}\sum_{i=1}^{n} a_i$)
- $E = 5T$

Approximation results for DISCRETE and INCREMENTAL

Proposition (Polynomial-time approximation algorithms)

- *With the* DISCRETE *model, for any integer $K > 0$, the* MINENERGY*(G,D) problem can be approximated within a factor*

$$(1 + \frac{\alpha}{s_1})^2 \times (1 + \frac{1}{K})^2,$$

where $\alpha = \max_{1 \leq i < m}\{s_{i+1} - s_i\}$, in a time polynomial in the size of the instance and in K.

- *With the* INCREMENTAL *model, the same result holds where $\alpha = \delta$ ($s_1 = s_{min}$).*

Approximation results for DISCRETE and INCREMENTAL

Proposition (Comparaison to the optimal solution)

For any integer $\delta > 0$, any instance of MINENERGY*(G,D) with the* CONTINUOUS *model can be approximated within a factor $(1 + \frac{\delta}{s_{\min}})^2$ in the* INCREMENTAL *model with speed increment δ.*

Summary

- Results for CONTINUOUS, but not very practical
- In real life, DISCRETE model (DVFS)
- VDD-HOPPING: good alternative, mixing two consecutive modes, smoothes out the discrete nature of modes
- INCREMENTAL: alternate (and simpler in practice) solution, with one unique speed during task execution; can be made arbitrarily efficient

Outline

What we had:

Energy-efficient scheduling
+
frequency scaling

What we aim at:

Thanks...

...to my co-authors Guillaume Aupy, Fanny Dufossé, Paul Renaud-Goud, and Yves Robert.

Bibliography:

- On the performance of greedy algorithms for energy minimization (Benoit, Renaud-Goud, Robert, 2011)
- Reclaiming the energy of a schedule: models and algorithms (Aupy, Benoit, Dufossé, Robert, 2013)

Performance and energy optimization of concurrent pipelined applications

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr

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

CR02 - 2016/2017

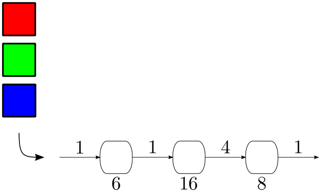
Motivations

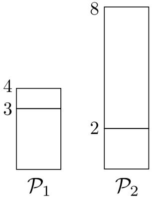
- Mapping of concurrent pipelined applications on parallel platform: practical applications, but difficult problem
- $\Rightarrow$ classification of mappings and platforms
- Energy saving is becoming a crucial problem
- Objective functions: period, latency, power
- Multi-criteria approach
- Complexity study

Outline of the talk

Outline

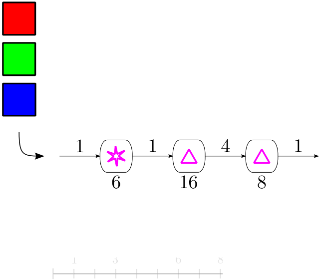
Motivating example

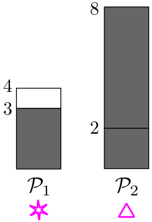




- Period: $T = 3$
- Latency: $L = 8$

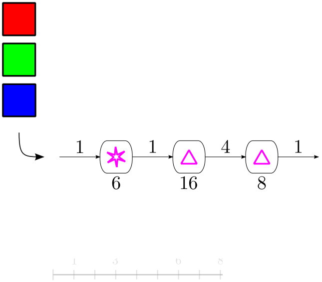
Motivating example

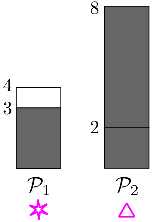




- Period: $T = 3$
- Latency: $L = 8$

Motivating example

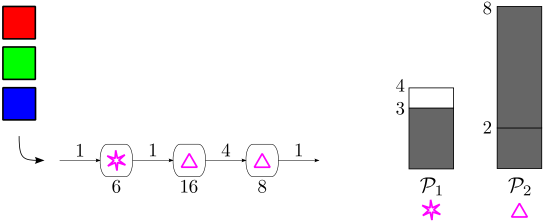




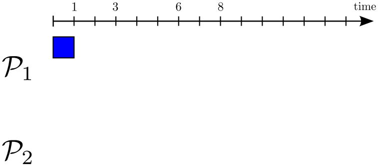
$$P = 3^3 + 8^3$$
$$= 539$$

- Period: $T = 3$
- Latency: $L = 8$

Motivating example

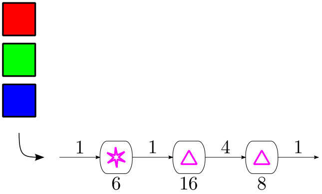


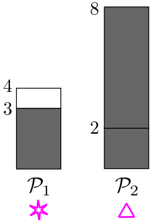
$$P = 3^3 + 8^3 = 539$$



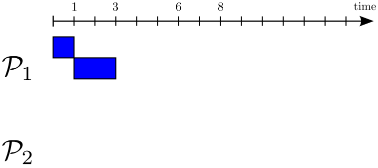
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



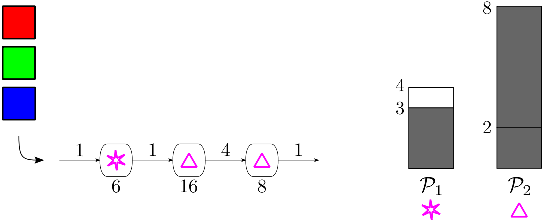


$$P = 3^3 + 8^3$$
$$= 539$$

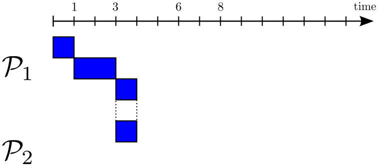


- Period: $T = 3$
- Latency: $L = 8$

Motivating example

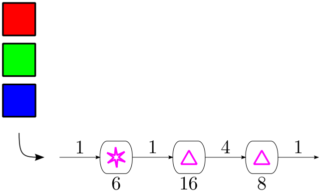


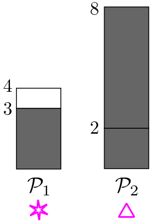
$P = 3^3 + 8^3$
$= 539$



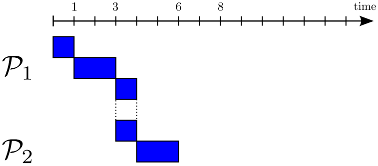
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



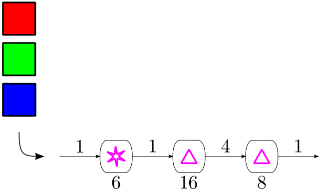


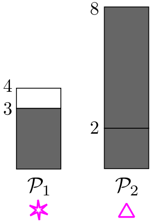
$$P = 3^3 + 8^3$$
$$= 539$$



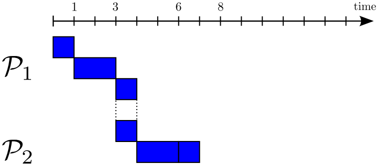
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



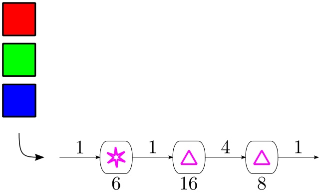


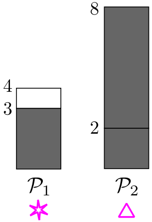
$$P = 3^3 + 8^3$$
$$= 539$$



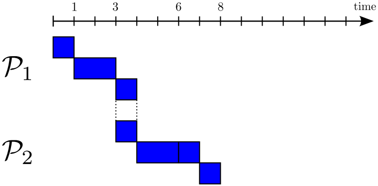
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



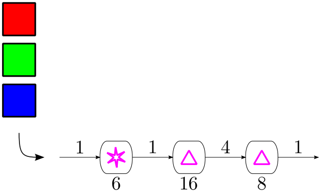


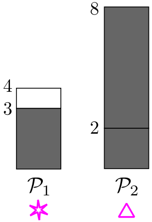
$$P = 3^3 + 8^3$$
$$= 539$$



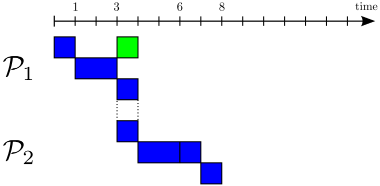
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



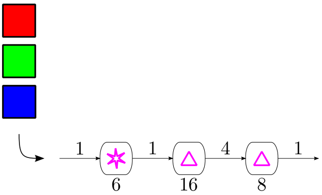


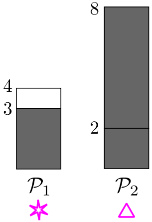
$$P = 3^3 + 8^3$$
$$= 539$$



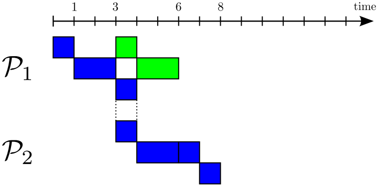
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



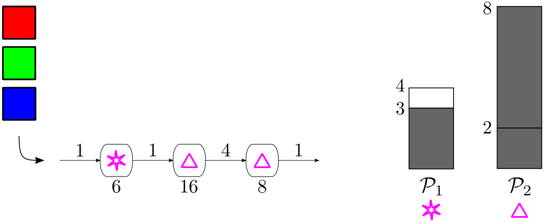


$$P = 3^3 + 8^3$$
$$= 539$$

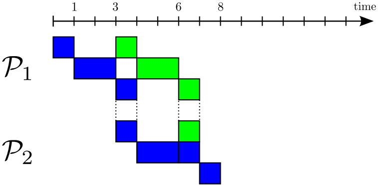


- Period: $T = 3$
- Latency: $L = 8$

Motivating example

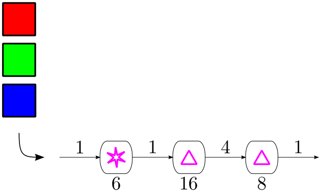


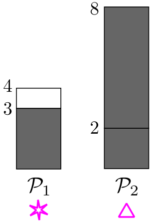
$$P = 3^3 + 8^3$$
$$= 539$$



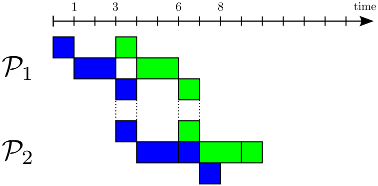
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



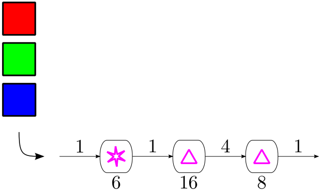


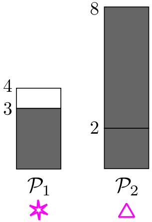
$$P = 3^3 + 8^3$$
$$= 539$$



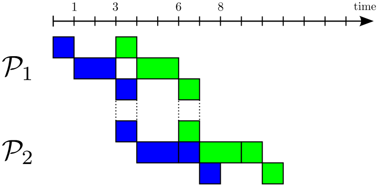
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



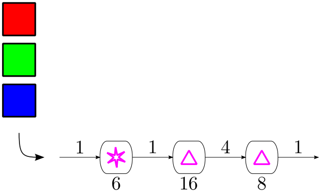


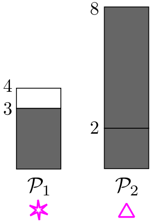
$$P = 3^3 + 8^3 = 539$$



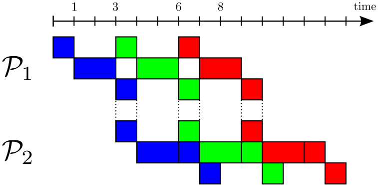
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



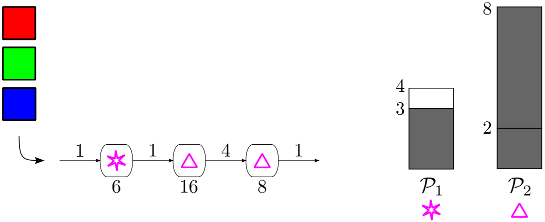


$$P = 3^3 + 8^3 = 539$$

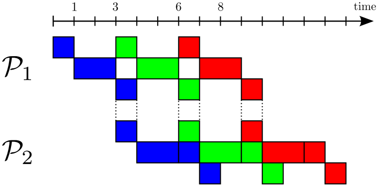


- Period: $T = 3$
- Latency: $L = 8$

Motivating example

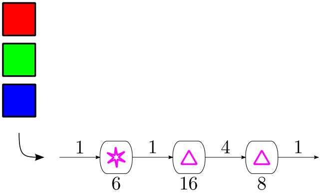


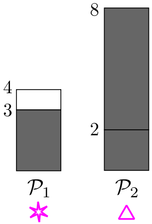
$$P = 3^3 + 8^3$$
$$= 539$$



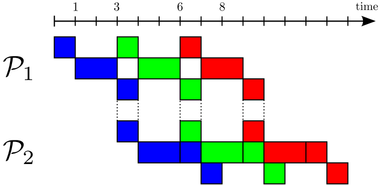
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



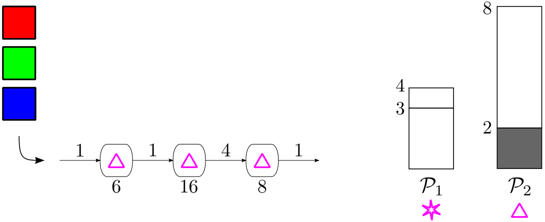


$$P = 3^3 + 8^3$$
$$= 539$$

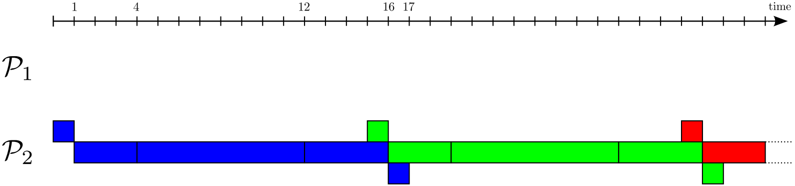


- Period: $T = 3$
- Latency: $L = 8$

Motivating example

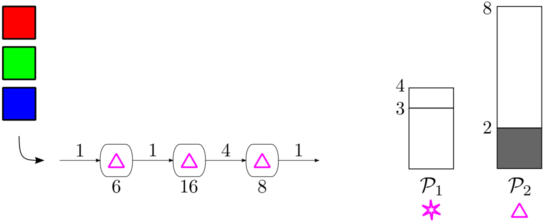


$$P = 3^3 + 8^3$$
$$= 539$$



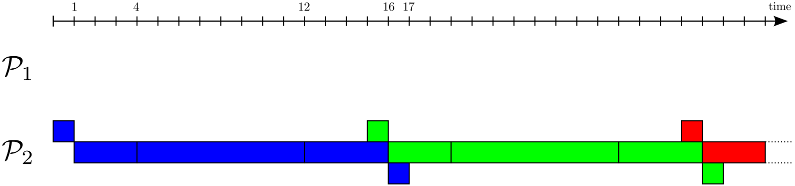
- Period: $T = 3$
- Latency: $L = 8$

Motivating example



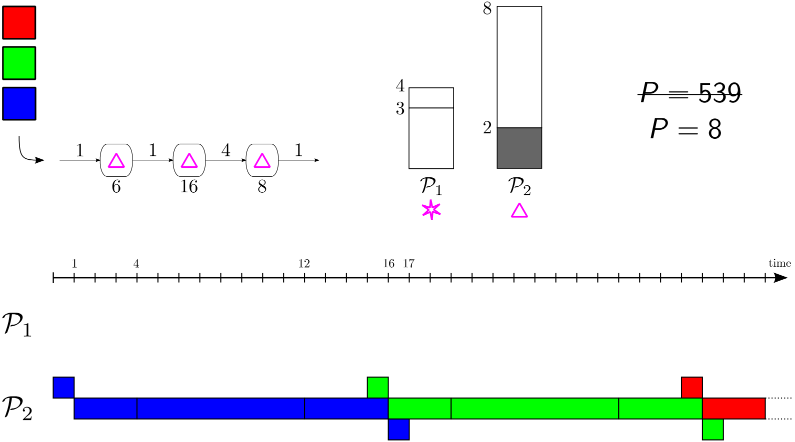
~~$P = 539$~~

$P = 8$



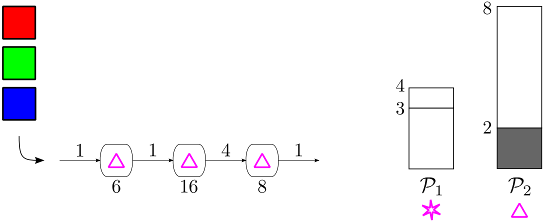
- Period: $T = 3$
- Latency: $L = 8$

Motivating example

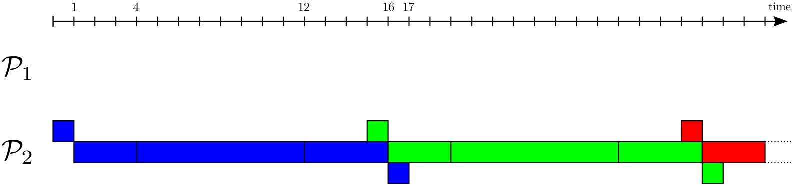


- Period: ~~$T = 3$~~ $T = 15$
- Latency: $L = 8$

Motivating example



~~$P = 539$~~
$P = 8$



- Period: ~~$T = 3$~~ $T = 15$
- Latency: ~~$L = 8$~~ $L = 17$

Applications and platform

- For an application a:
  - w_a^i: weight of stage $\mathcal{S}_a^i$
  - δ_a^i: size of outcoming data of $\mathcal{S}_a^i$

- Processors with multiple speeds (or modes): $\{s_{u,1}, \ldots, s_{u,m_u}\}$
  Constant speed during the execution
  $b_{u,v}$: bandwidth between processors $\mathcal{P}_u$ and $\mathcal{P}_v$

- Platform fully interconnected

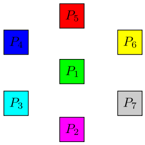
- Communications: both overlap or non-overlap model

- Three platforms types:
  1. Fully homogeneous
  2. Communication homogeneous
  3. Fully heterogeneous

Mappings

No processor sharing for both practical and theoretical reasons (security rules and NP-completeness of the execution scheduling given a mapping with a period/latency objective).

- One-to-one mapping

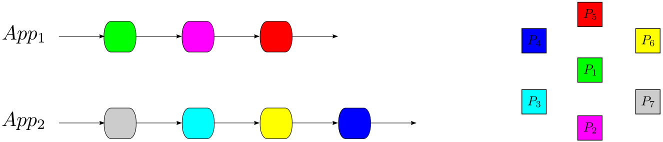


- Interval mapping

Mappings

No processor sharing for both practical and theoretical reasons (security rules and NP-completeness of the execution scheduling given a mapping with a period/latency objective).

- One-to-one mapping



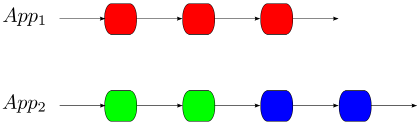
- Interval mapping

Mappings

No processor sharing for both practical and theoretical reasons (security rules and NP-completeness of the execution scheduling given a mapping with a period/latency objective).

- One-to-one mapping

- Interval mapping



Metrics

Interval mapping on a single application ; k intervals I_j of stages from $\mathcal{S}^{d_j}$ to $\mathcal{S}^{e_j}$; al assignment procedure

- Period T of an application: the minimum delay between the processing of two consecutive set of data

$$T^{(overlap)} = \max_{j \in \{1,\dots,k\}} \left(\max \left(\frac{\delta^{d_j-1}}{b_{\mathrm{al}(d_j-1),\mathrm{al}(d_j)}}, \frac{\sum_{i=d_j}^{e_j} w^i}{s_{\mathrm{al}(d_j)}}, \frac{\delta^{e_j}}{b_{\mathrm{al}(d_j),\mathrm{al}(e_j+1)}} \right) \right)$$

- Latency L of an application: time, for a data set, to go through the whole pipeline

$$L = \frac{\delta^0}{b_{\mathrm{al}(0),\mathrm{al}(1)}} + \sum_{j=1}^{m} \left(\sum_{i=d_j}^{e_j} \frac{w^i}{s_{\mathrm{al}(d_j)}} + \frac{\delta^{e_j}}{b_{\mathrm{al}(d_j),\mathrm{al}(e_j+1)}} \right)$$

- Power of a processor $\mathcal{P}_u$:

$$P(u) = P_{dyn}(s_u) + P_{stat}(u) \quad , \quad P_{dyn}(s_u) = s_u^{\alpha}$$

Optimization problems

- Minimize one criterion:
  - Period or latency: minimize $\max_a W_a \times T_a$ or $\max_a W_a \times L_a$
  - Power: minimize $\sum_u P(u)$
- Fix one criterion:
  - Fix the period or latency of each application $\rightarrow$ fix a period or latency array
  - Fix $\sum_u P(u)$
- Multi-criteria approach: minimizing 1 criterion, fixing the other ones
- Power consumption, i.e., energy per time unit
  $\Rightarrow$ combination power/period

Outline

Complexity results

Period minimization:

	proc-hom	proc-het		
	com-hom	special-app[1]	com-hom	com-het
one-to-one	polynomial (binary search)			NP-complete
interval	polynomial	NP-complete	NP-complete	

[1]special-app: com-hom & pipe-hom

Period minimization - com-hom - one-to-one

Problem: one-to-one mapping - many applications - communication homogeneous platform - period minimization

Algorithm 1: Greedy-Assignment(T)

begin
- Work with fastest N processors, numbered $\mathcal{P}_1$ to $\mathcal{P}_N$, where $s_1 \leq s_2 \leq \cdots \leq s_N$;
- Mark all stages as free;
- **for** $u = 1$ **to** N **do**
  - Pick up any free stage $\mathcal{S}_a^k$ s.t. $W_a \times \max(\frac{\delta_a^{k-1}}{b}, \frac{w_a^k}{s_u}, \frac{\delta_a^k}{b}) \leq T$;
  - Assign $\mathcal{S}_a^k$ to $\mathcal{P}_u$;
  - Mark $\mathcal{S}_a^k$ as already assigned;
  - **if** no stage found **then**
    - **return** "failure";
  - **end**
- **end**
- **return** "success";

end

Period minimization - interval

- Polynomial for fully homogeneous platforms, building upon optimal algorithm for a single application
- NP-complete even with a homogeneous application with heterogeneous processors

Period minimization - heterogeneous

NP-complete! Involved reduction from MINIMUM METRIC BOTTLENECK WANDERING SALESPERSON PROBLEM:

- Set of m cities $c_1, \dots, c_m$
- Distances $d(c_i, c_j)$ satisfying the triangle inequality
- Find a simple path from c_1 to c_m, while minimizing the maximum distance in the path

Complexity results

Period minimization:

	proc-hom	proc-het		
	com-hom	special-app[1]	com-hom	com-het
one-to-one	polynomial (binary search)			NP-complete
interval	polynomial	NP-complete	NP-complete	

Latency minimization:

	proc-hom	proc-het		
	com-hom	special-app[1]	com-hom	com-het
one-to-one	polynomial	NP-complete		NP-complete
interval	polynomial (binary search)			NP-complete

[1] special-app: com-hom & pipe-hom

Latency minimization

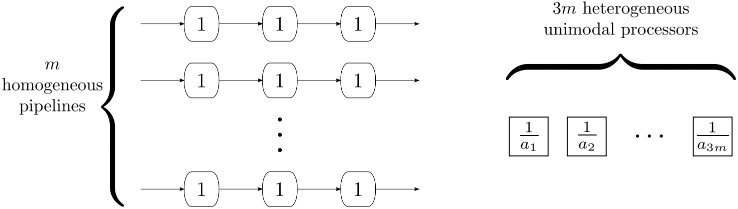
- Problem: one-to-one mapping - many applications - heterogeneous platform - no communication - homogeneous pipelines - minimize $\max_a L_a$
- Single application: greedy polynomial algorithm
- Many applications: reduction from 3-PARTITION
- 3-PARTITION:
  - Input: $3m+1$ integers $a_1, a_2, \ldots, a_{3m}$ and B such that $\sum_i a_i = mB$
  - Does there exist a partition $I_1, \ldots, I_m$ of $\{1, \ldots, 3m\}$ such that for all $j \in \{1, \ldots, m\}$, $|I_j| = 3$ and $\sum_{i \in I_j} a_i = B$?

Latency minimization (2)

- 3-PARTITION: does there exist a renumbering of a_i such that:

$$\begin{cases} a_{1,1} + a_{1,2} + a_{1,3} = B \\ a_{2,1} + a_{2,2} + a_{2,3} = B \\ \vdots \\ a_{m,1} + a_{m,2} + a_{m,3} = B \end{cases}$$

- Reduction:



Can we obtain a latency $L^0 \leq B$?

- Equivalence of problems

Outline

Complexity results

Period/latency minimization:

	proc-hom	proc-het		
	com-hom	special-app	com-hom	com-het
one-to-one or interval	polynomial	NP-complete		

Power/period minimization:

	proc-hom	proc-het		
	com-hom	special-app	com-hom	com-het
one-to-one	polynomial (minimum matching)			NP-complete
interval	polynomial	NP-complete		

Complexity results

Period/latency minimization:

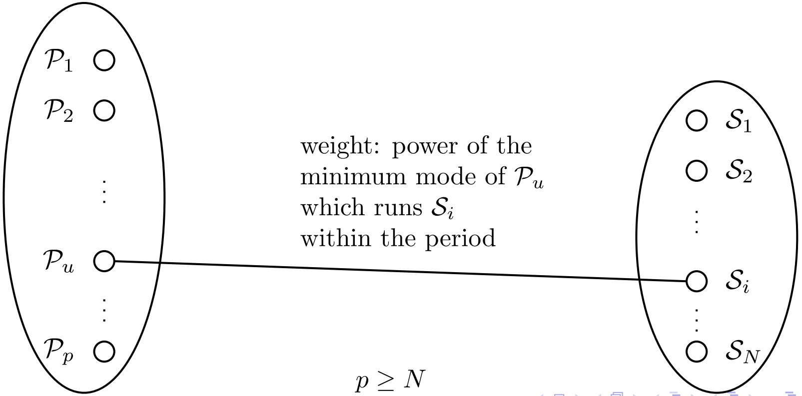
	proc-hom	proc-het		
	com-hom	special-app	com-hom	com-het
one-to-one or interval	polynomial	NP-complete		

Power/period minimization:

	proc-hom	proc-het		
	com-hom	special-app	com-hom	com-het
one-to-one	polynomial (minimum matching)			NP-complete
interval	polynomial	NP-complete		

Power/period minimization

- Problem: one-to-one mapping - many applications - communication homogeneous platform - power minimization for a given array of periods
- Minimum weighted matching of a bipartite graph



Complexity results

Period/latency minimization:

	proc-hom	proc-het		
	com-hom	special-app	com-hom	com-het
one-to-one or interval	polynomial	NP-complete		

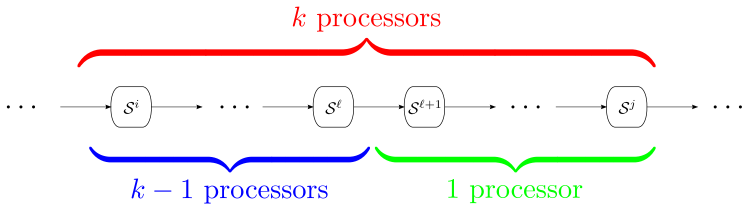
Power/period minimization:

	proc-hom	proc-het		
	com-hom	special-app	com-hom	com-het
one-to-one	polynomial (minimum matching)			NP-complete
interval	polynomial	NP-complete		

Single application

- Problem: interval mapping - single application - fully homogeneous platform - power minimization for a given period
- $P(i,j,k)$: minimum power to run stages $\mathcal{S}^i$ to $\mathcal{S}^j$ using exactly k processors → looking for $\min_{1 \leq k \leq p} P(1,n,k)$
- Recurrence relation:

$$P(i,j,k) = \min_{1 \leq \ell \leq j-1} \left(P(i,\ell,k-1) + P(\ell+1,j,1) \right)$$



Single application (2)

- $P(i,i,q) = +\infty \quad \text{if} \quad q > 1$

- $\mathcal{F}_i^j$: possible powers of a processor running the stages $\mathcal{S}^i$ to $\mathcal{S}^j$, fulfilling the period constraint

$$\mathcal{F}_i^j = \left\{ P_{dyn}(s_\ell) + P_{stat}, \max\left(\frac{\delta^{i-1}}{b}, \frac{\sum_{k=i}^j w^k}{s_\ell}, \frac{\delta^j}{b}\right) \le T, \ell \in \{1, \dots, m\} \right\}$$

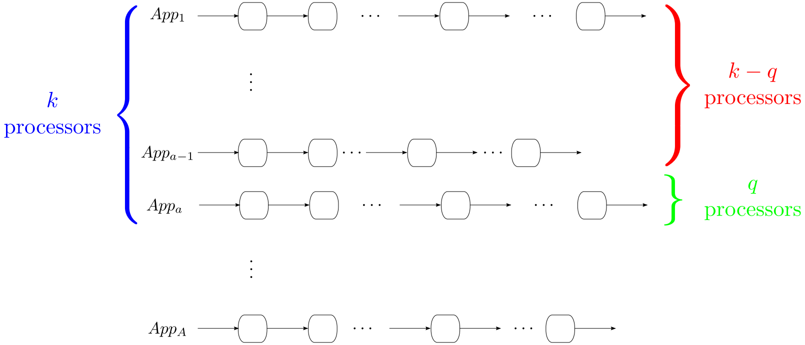
- $P(i,j,1) = \begin{cases} \min \mathcal{F}_i^j & \text{if } \mathcal{F}_i^j \neq \emptyset \\ +\infty & \text{otherwise} \end{cases}$

Many applications

- Problem: interval mapping - fully homogeneous platform - power minimization for given periods by application
- P_a^q: minimum power consumed by q processors so that the period constraint on the application a is met, found by the previous dynamic programming
- $P(a,k)$: minimum power consumed by k processors on the applications $1,\ldots,a$, unknown
- Initialization: $\forall k \in \{1,\ldots,p\} \quad P(1,k) = P_1^k$

Many applications (2)

- Recurrence: $P(a,k) = \min_{1 \leq q < k} \left(P(a-1, k-q) + P_a^q \right)$



Outline

Complexity results

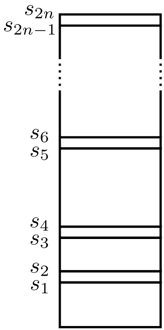
	proc-hom	proc-het		
	com-hom	special-app	com-hom	com-het
one-to-one or interval	NP-complete			

Reduction from 2-PARTITION

(Instance of 2-PARTITION: $a_1, a_2, \ldots, a_n$ with $\sigma = \sum_{i=1}^{n} a_i$)

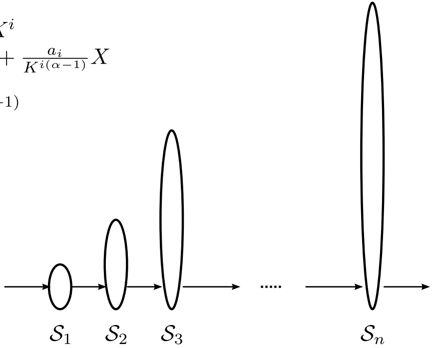
Problem instance

One-to-one mapping - fully homogeneous platform



$$\begin{cases} s_{2i-1} = K^i \\ s_{2i} = K^i + \frac{a_i}{K^{i(\alpha-1)}} X \end{cases}$$

$$w_i = K^{i(\alpha+1)}$$



$P^0 = P^* + \alpha X(\sigma/2 + 1/2)$, $L^0 = L^* - X(\sigma/2 - 1/2)$, $T^0 = L^0$ where P^* and L^* are power and latency when each $\mathcal{S}_i$ is run at speed s_{2i-1}

Main ideas

- K big enough and X small enough so that the stage $\mathcal{S}_i$ must be processed at speed s_{2i-1} or s_{2i}
- For a subset $\mathcal{I}$ of $\{1, \ldots, n\}$, if ($\mathcal{S}_i$ is run at speed s_{2i} $\Leftrightarrow i \in \mathcal{I}$),

$$P = P^* + \sum_{i \in \mathcal{I}} (\alpha a_i X + o(X)) \quad , \quad L = L^* - \sum_{i \in \mathcal{I}} (a_i X - o(X))$$

- Recall:

$$P^0 = P^* + \alpha X(\sigma/2 + 1/2) \quad , \quad L^0 = L^* - X(\sigma/2 - 1/2)$$

Outline

Conclusion

- New polynomial algorithms for a single application
- Polynomial algorithms for a single application extended to many applications
- New results of NP-completeness
- Exhaustive complexity study

- Bibliography: Models and complexity results for performance and energy optimization of concurrent streaming applications (Benoit, Renaud-Goud, Robert, 2011)