Energy-aware algorithms

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr
http://graal.ens-lyon.fr/~abenoit

CR02 - 2015/2016
Exascale platforms

- Hierarchical
  - $10^5$ or $10^6$ nodes
  - Each node equipped with $10^4$ or $10^3$ cores

- Failure-prone

<table>
<thead>
<tr>
<th>MTBF – one node</th>
<th>1 year</th>
<th>10 years</th>
<th>120 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTBF – platform of $10^6$ nodes</td>
<td>30sec</td>
<td>5mn</td>
<td>1h</td>
</tr>
</tbody>
</table>

More nodes $\Rightarrow$ Shorter MTBF (Mean Time Between Failures)

- Energy efficiency
  - Thermal power close to the one of a nuclear reactor!
  - A critical issue to address if we want to achieve Exascale.
Exascale platforms

- Hierarchical
  - 10^5 or 10^6 nodes
  - Each node equipped with 10^4 or 10^3 cores

- Failure-prone

<table>
<thead>
<tr>
<th>MTBF – one node</th>
<th>1 year</th>
<th>10 years</th>
<th>120 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTBF – platform</td>
<td>30s</td>
<td>5mn</td>
<td>1h</td>
</tr>
<tr>
<td>of 10^6 nodes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

More nodes ⇒ Shorter MTBF (Mean Time Between Failures)

- Energy efficiency
  - Thermal power close to the one of a nuclear reactor!
  - A critical issue to address if we want to achieve Exascale.

Exascale \(\neq\) Petascale \(\times 1000\)
Outline

1. Introduction and motivation: energy
2. Revisiting the greedy algorithm for independent jobs
3. Reclaiming the slack of a schedule
4. Conclusion
Energy: a crucial issue

- **Data centers**
  - 330,000,000,000 Watts hour in 2007: more than France
  - 533,000,000 tons of CO₂: in the top ten countries

- **Exascale computers** (10^{18} floating operations per second)
  - Need effort for feasibility
  - 1% of power saved → 1 million dollar per year

- **Lambda user**
  - 1 billion personal computers
  - 500,000,000,000,000 Watts hour per year

- ~ crucial for both environmental and economical reasons
Energy: a crucial issue

- Data centers
  - 330,000,000
  - 533,000,000

- Exascale computers (10^18 floating operations per second)
  - Need effort
  - 1% of power saved
  - 1 million dollar per year

- Lambda user
  - 1 billion per year
  - 500,000,000,000

- Crucial for both environmental and economical reasons
Power dissipation of a processor

- \( P = P_{\text{leak}} + P_{\text{dyn}} \)
  - \( P_{\text{leak}} \): constant
  - \( P_{\text{dyn}} = B \times V^2 \times f \)
    - constant
    - supply voltage
    - frequency

- Standard approximation: \( P = P_{\text{leak}} + f^\alpha \) \((2 \leq \alpha \leq 3)\)

- Energy \( E = P \times \text{time} \)

- **Dynamic Voltage and Frequency Scaling (DVFS)** to reduce dynamic power
  - Real life: discrete speeds
  - Continuous speeds can be emulated

- **Processor shutdown** to reduce static power
Speed models for DVFS

<table>
<thead>
<tr>
<th>Type of speeds</th>
<th>[s_{\text{min}}, s_{\text{max}}]</th>
<th>{s_1, \ldots, s_m}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>When can we change speed?</strong></td>
<td><strong>Anytime</strong></td>
<td><strong>Beginning of tasks</strong></td>
</tr>
<tr>
<td><strong>Continuous</strong></td>
<td>\text{Vdd-Hopping}</td>
<td>\text{Discrete, Incremental}</td>
</tr>
</tbody>
</table>

- **Continuous**: great for theory
- Other "discrete" models more realistic
- \text{Vdd-Hopping} simulates \text{Continuous}
- \text{Incremental} is a special case of \text{Discrete} with equally-spaced speeds: for all \(1 \leq q < m\), \(s_{q+1} - s_q = \delta\)
Outline

1. Introduction and motivation: energy
2. Revisiting the greedy algorithm for independent jobs
   - Framework
   - Related work
   - Approximation results
3. Reclaiming the slack of a schedule
4. Conclusion
Framework

- Scheduling independent jobs

- **Greedy algorithm**: assign next job to least-loaded processor

- **Two variants:**
  - **OnLine-Greedy**: assign jobs on the fly
  - **OffLine-Greedy**: sort jobs before execution
Classical problem

- $n$ independent jobs $\{J_i\}_{1 \leq i \leq n}$, $a_i =$ size of $J_i$
- $p$ processors $\{P_q\}_{1 \leq q \leq p}$
- allocation function $\text{alloc} : \{J_i\} \rightarrow \{P_q\}$
- load of $P_q = \text{load}(q) = \sum\{i \mid \text{alloc}(J_i) = P_q\} a_i$

Execution time:
$$\max_{1 \leq q \leq p} \text{load}(q)$$
**OnLine-Greedy**

**Theorem**

**OnLine-Greedy** is a $2 - \frac{1}{p}$ approximation (tight bound)

<table>
<thead>
<tr>
<th>$\mathcal{P}_1$</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{P}_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P}_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P}_4$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P}_5$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**OnLine-Greedy**  

<table>
<thead>
<tr>
<th>$\mathcal{P}_1$</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{P}_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\mathcal{P}_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\mathcal{P}_4$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\mathcal{P}_5$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Optimal solution
OffLine-Greedy is a $\frac{4}{3} - \frac{1}{3p}$ approximation (tight bound)
Power consumption

“The internet begins with coal”

- DVFS: Dynamic Voltage and Frequency Scaling
- Power at speed $s$ (continuous model):

\[ P(s) = P_{static} + \lambda \times s^3 \]
Power consumption

“The internet begins with coal”

- DVFS: Dynamic Voltage and Frequency Scaling
- Power at speed $s$ (continuous model):

\[ P(s) = P_{\text{static}} + \lambda \times s^3 \]
Outline

1. Introduction and motivation: energy

2. Revisiting the greedy algorithm for independent jobs
   - Framework
   - Related work
   - Approximation results

3. Reclaiming the slack of a schedule

4. Conclusion
Bi-criteria problem

- Minimizing (dynamic) power consumption:
  ⇒ use slowest possible speed
  \[ P_{\text{dyn}} = f^\alpha = f^3 \]

- Bi-criteria problem:
  Given bound \( M = 1 \) on execution time,
  minimize power consumption while meeting the bound
Bi-criteria problem statement

- $n$ independent jobs $\{J_i\}_{1 \leq i \leq n}$, $a_i = \text{size of } J_i$
- $p$ processors $\{P_q\}_{1 \leq q \leq p}$
- Allocation function $\text{alloc} : \{J_i\} \rightarrow \{P_q\}$
- Load of $P_q = \text{load}(q) = \sum_{\{i \mid \text{alloc}(J_i) = P_q\}} a_i$

$$(\text{load}(q))^3$$ power dissipated by $P_q$

$$\sum_{q=1}^{p} (\text{load}(q))^3$$ Power

$$\max_{1 \leq q \leq p} \text{load}(q)$$ Execution time
Same **GREEDY** algorithm . . .

- **Strategy**: assign next job to least-loaded processor

- **Natural for execution-time**
  - smallest increment of maximum load
  - minimize objective value for currently processed jobs

- **Natural for power too**
  - smallest increment of total power (convexity)
  - minimize objective value for currently processed jobs
... but different optimal solution!

- Makespan 10, power 2531.441
- Makespan 10.1, power 2488.301
Outline

1. Introduction and motivation: energy

2. Revisiting the greedy algorithm for independent jobs
   - Framework
   - Related work
   - Approximation results

3. Reclaiming the slack of a schedule

4. Conclusion
**Greedy and $L_r$ norms**

\[
N_r = \left( \sum_{q=1}^{p} (\text{load}(q))^r \right)^{\frac{1}{r}}
\]

- **Execution time** $N_\infty = \lim_{r \to \infty} N_r = \max_{1 \leq q \leq p} \text{load}(q)$
- **Power** $(N_3)^3$
Known results

\(N_2, \text{OffLine-Greedy}\)
- Chandra and Wong 1975: upper and lower bounds
- Leung and Wei 1995: tight approximation factor

\(N_3, \text{OffLine-Greedy}\)
- Chandra and Wong 1975: upper and lower bounds

\(N_r\)
- Alon et al. 1997: PTAS for offline problem
- Avidor et al. 1998: upper bound \(2 - \Theta\left(\frac{\ln r}{r}\right)\) for \(\text{OnLine-Greedy}\)
$N_3$

- Tight approximation factor for $\text{ONLINE-GREEDY}$
- Tight approximation factor for $\text{OFFLINE-GREEDY}$

- Greedy for power fully solved!
Outline

1. Introduction and motivation: energy

2. Revisiting the greedy algorithm for independent jobs
   - Framework
   - Related work
   - Approximation results

3. Reclaiming the slack of a schedule

4. Conclusion
Best-case for optimal solution

\[ P_1 \quad O \]
\[ P_2 \quad \frac{S - 0}{p - 1} \]
\[ P_3 \quad \frac{S - 0}{p - 1} \]
\[ \vdots \]
\[ P_p \quad \frac{S - 0}{p - 1} \]

\( O \) largest processor load in optimal solution, \( S = \sum_{i=1}^{n} a_i \)

\[ P_{opt} \geq O^3 + (p - 1) \left( \frac{S - O}{p - 1} \right)^3 \]
Best-case for optimal solution

\[ P_1 = \begin{array}{c} \text{O} \\ \text{O} \end{array} \]

\[ P_2 = \begin{array}{c} \frac{S_0}{p-1} \\ \frac{S_0}{p-1} \end{array} \]

\[ P_3 = \begin{array}{c} \frac{S_0}{p-1} \\ \frac{S_0}{p-1} \end{array} \]

\[ \vdots \]

\[ P_p = \begin{array}{c} \frac{S_0}{p-1} \\ \frac{S_0}{p-1} \end{array} \]

\[ O \text{ largest processor load in optimal solution, } S = \sum_{i=1}^{n} a_i \]

\[ P_{opt} \geq O^3 + (p - 1) \left( \frac{S - O}{p - 1} \right)^3 \]
Worst-case for **Greedy**

\[
P_{\text{greedy}} \leq \left( \frac{S + (p-1)a_j}{p} \right)^3 + (p-1) \left( \frac{S - a_j}{p} \right)^3
\]

**J_j** last job assigned to most loaded processor in **Greedy**
Worst-case for **GREEDY**

\[
P_1 \leq \frac{S - a_j}{p}
\]

\[
P_2 \leq \frac{S - a_j}{p}
\]

\[
P_3 \leq \frac{S - a_j}{p}
\]

\[
\vdots
\]

\[
P_p \leq \frac{S - a_j}{p}
\]

\[J_j \text{ last job assigned to most loaded processor in GREEDY}\]

\[
P_{\text{greedy}} \leq \left(\frac{S + (p - 1)a_j}{p}\right)^3 + (p - 1)\left(\frac{S - a_j}{p}\right)^3
\]
Proof sketch (1/3)

\[ P_1 \]

\[ M_1 \]

\[ u_1 = a_j \]

\[ P_2 \]

\[ M_2 \]

\[ u_2 \]

\[ P_3 \]

\[ M_3 \]

\[ P_4 \]

\[ M_4 \]

\[ u_4 \]

Notations

- \( P_1 \) maximum loaded processor in \text{Greedy}
- Load of \( P_q \): \( M_q \) before job \( J_j \), \( M_q + u_q \) final
- \( P_{\text{greedy}} = (M_1 + a_j)^3 + \sum_{q=2}^{p}(M_q + u_q)^3 \)
Notations

- For \( q \geq 2 \), rewrite \( M_q + u_q = \frac{S - M_1 - a_j}{p - 1} + v_q \)

- \( P_{\text{greedy}} = (M_1 + a_j)^3 + \sum_{q=2}^{p} \left( \frac{S - M_1 - a_j}{p - 1} + v_q \right)^3 \)

\[ f(M_1) \]
Proof sketch (3/3)

- **Show:** $f(M_1)$ strictly increasing
  - **Observe:** $M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q$
  - **Derive:** $M_1 \leq M_1^+ = \frac{S - a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
  - **Check:** if $M_1 = M_1^+$, then $v_q = 0$ for all $q$
  - **Conclude** 😊
Proof sketch (3/3)

- Show: $f(M_1)$ strictly increasing
- Observe: $M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p - 1} + v_q$
- Derive: $M_1 \leq M_1^+ = \frac{S - a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
- Check: if $M_1 = M_1^+$, then $v_q = 0$ for all $q$
- Conclude 😊
Proof sketch (3/3)

- Show: $f(M_1)$ strictly increasing
- Observe: $M_1 \leq M_q \leq M_q + u_q = \frac{S-M_1-a_j}{p-1} + v_q$
- Derive: $M_1 \leq M_1^+ = \frac{S-a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
- Check: if $M_1 = M_1^+$, then $v_q = 0$ for all $q$
- Conclude 😊
Proof sketch (3/3)

- Show: \( f(M_1) \) strictly increasing
- Observe: \( M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q \)
- Derive: \( M_1 \leq M_1^+ = \frac{S - a_j}{p} \) and \( P_{\text{greedy}} = f(M_1) \leq f(M_1^+) \)
- Check: if \( M_1 = M_1^+ \), then \( v_q = 0 \) for all \( q \)
- Conclude 😊
Proof sketch (3/3)

- Show: $f(M_1)$ strictly increasing
- Observe: $M_1 \leq M_q \leq M_q + u_q = \frac{S - M_1 - a_j}{p-1} + v_q$
- Derive: $M_1 \leq M_1^+ = \frac{S - a_j}{p}$ and $P_{\text{greedy}} = f(M_1) \leq f(M_1^+)$
- Check: if $M_1 = M_1^+$, then $v_q = 0$ for all $q$
- Conclude 😊
Approximation bound

\[
\frac{P_{\text{greedy}}}{P_{\text{opt}}} \leq \left( \frac{S+(p-1)a_j}{p} \right)^3 + (p-1) \left( \frac{S-a_j}{p} \right)^3 \frac{O^3}{(p-1)^3} + (p-1) \left( \frac{S-O}{p-1} \right)^3
\]

Agenda

- Right-hand-side is increasing with \( a_j \)
- Rewrite with \( \beta = \frac{O}{S} \in \left[ \frac{1}{p}, 1 \right] \) and bound \( a_j \):
  
  \[
  a_j \leq O \quad \text{for } \text{ONLINE-GREEDY}
  \]
  
  \[
  a_j \leq O/3 \quad \text{for } \text{OFFLINE-GREEDY}
  \]
Approximation for **ONLINE-GREEDY**

\[
\frac{P_{\text{online}}}{P_{\text{opt}}} \leq \frac{1}{p^3} \left( \left( 1 + (p - 1)\beta \right)^3 + (p - 1)(1 - \beta)^3 \right) \frac{\beta^3 + \frac{(1-\beta)^3}{(p-1)^2}}{f_p^{(\text{on})}(\beta)}
\]

**Theorem**

- \( f_p^{(\text{on})} \) has a single maximum in \( \beta_p^{(\text{on})} \in \left[ \frac{1}{p}, 1 \right] \)
- **ONLINE-GREEDY** is a \( f_p^{(\text{on})}(\beta_p^{(\text{on})}) \) approximation
- This approximation factor is tight
Approximation for OffLine-Greedy

\[
\frac{P_{\text{offline}}}{P_{\text{opt}}} \leq \frac{1}{p^3} \left( \left(1 + \frac{(p-1)\beta}{3}\right)^3 + (p-1) \left(1 - \frac{\beta}{3}\right)^3 \right) \frac{\beta^3 + \left(\frac{1-\beta}{p-1}\right)^3}{f_p^{(\text{off})}(\beta)}
\]

**Theorem**

- \(f_p^{(\text{off})}\) has a single maximum in \(\beta_p^{(\text{off})} \in \left[ \frac{1}{p}, 1 \right]\)
- OffLine-Greedy is a \(f_p^{(\text{off})}(\beta_p^{(\text{off})})\) approximation
- This approximation factor is tight
**Numerical values of approximation ratios**

<table>
<thead>
<tr>
<th>$p$</th>
<th><strong>ONLINE-GREEDY</strong></th>
<th><strong>OFFLINE-GREEDY</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.866</td>
<td><strong>1.086</strong></td>
</tr>
<tr>
<td>3</td>
<td>2.008</td>
<td>1.081</td>
</tr>
<tr>
<td>4</td>
<td><strong>2.021</strong></td>
<td>1.070</td>
</tr>
<tr>
<td>5</td>
<td>2.001</td>
<td>1.061</td>
</tr>
<tr>
<td>6</td>
<td>1.973</td>
<td>1.054</td>
</tr>
<tr>
<td>7</td>
<td>1.943</td>
<td>1.048</td>
</tr>
<tr>
<td>8</td>
<td>1.915</td>
<td>1.043</td>
</tr>
<tr>
<td>64</td>
<td>1.461</td>
<td>1.006</td>
</tr>
<tr>
<td>512</td>
<td>1.217</td>
<td>1.00083</td>
</tr>
<tr>
<td>2048</td>
<td>1.104</td>
<td>1.00010</td>
</tr>
<tr>
<td>$2^{24}$</td>
<td>1.006</td>
<td><strong>1.000000025</strong></td>
</tr>
</tbody>
</table>
Large values of $p$

Asymptotic approximation factors

- **ONLINE-GREEDY**: $\frac{4}{3}$, $1$
- **OFFLINE-GREEDY**: $2$, $1$

$\uparrow$

*optimal*
Conclusion

Contribution

- **ONLine-Greedy** and **OffLine-Greedy** for power
- Tight approximation factor for any $p$
- Extend long series of papers
- Completely solve $N_3$ minimization problem 😊

Extending to DAG workflows

- Reclaim the energy of existing list schedules
- Design (and assess) power-aware algorithms
Conclusion

Contribution

- **ONLINE-GREEDY** and **OFFLINE-GREEDY** for power
- Tight approximation factor for any $p$
- Extend long series of papers
- Completely solve $N_3$ minimization problem 😊

Extending to DAG workflows

- Reclaim the energy of existing list schedules
- Design (and assess) power-aware algorithms
Outline

1. Introduction and motivation: energy
2. Revisiting the greedy algorithm for independent jobs
3. Reclaiming the slack of a schedule
   - Models
   - Example
   - Complexity results
4. Conclusion
Motivation

- Mapping of tasks is given (ordered list for each processor and dependencies between tasks)
- If deadline not tight, why not take our time?
- Slack: unused time slots

Goal: efficiently use speed scaling (DVFS)
Outline

1. Introduction and motivation: energy
2. Revisiting the greedy algorithm for independent jobs
3. Reclaiming the slack of a schedule
   - Models
   - Example
   - Complexity results
4. Conclusion
Speed models

<table>
<thead>
<tr>
<th>Type of speeds</th>
<th>Anytime</th>
<th>Change speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>([s_{\text{min}}, s_{\text{max}}])</td>
<td>CONTINUOUS</td>
<td>-</td>
</tr>
<tr>
<td>({s_1, \ldots, s_m})</td>
<td>VDD-HOPPING</td>
<td>DISCRETE, INCREMENTAL</td>
</tr>
</tbody>
</table>

- **CONTINUOUS**: great for theory *(what we used for independent tasks!)*
- Other ”discrete” models more realistic
- **VDD-HOPPING** simulates **CONTINUOUS**
- **INCREMENTAL** is a special case of **DISCRETE** with equally-spaced speeds: for all \(1 \leq q < m\), \(s_{q+1} - s_q = \delta\)
Tasks

- DAG: $\mathcal{G} = (V, E)$
- $n = |V|$ tasks $T_i$ of weight $w_i = \int_{t_i-d_i}^{t_i} s_i(t)\,dt$
- $d_i$: task duration; $t_i$: time of end of execution of $T_i$

Figure: Parameters for $T_i$ scheduled on processor $p_j$
Assume $T_i$ is executed at constant speed $s_i$

$$d_i = \text{Exe}(w_i, s_i) = \frac{w_i}{s_i}$$

$$t_j + d_i \leq t_i \text{ for each } (T_j, T_i) \in E$$

Constraint on makespan:

$$t_i \leq D \text{ for each } T_i \in V$$
Energy to execute task $T_i$ at speed $s_i$:

$$E_i(s_i) = d_i s_i^3 = w_i s_i^2$$

→ Dynamic part of classical energy models

Bi-criteria problem

- Constraint on deadline: $t_i \leq D$ for each $T_i \in V$
- Minimize energy consumption: $\sum_{i=1}^{n} w_i \times s_i^2$
Outline

1. Introduction and motivation: energy

2. Revisiting the greedy algorithm for independent jobs

3. Reclaiming the slack of a schedule
   - Models
   - Example
   - Complexity results

4. Conclusion
Consider this DAG, with $s_{\text{max}} = 6$. Suppose deadline is $D = 1.5$.

**Figure**: Execution graph for the example.
Example

- **Continuous**: \( (s_{\text{max}} = 6) \) \( E^{(c)}_{\text{opt}} \approx 109.6 \).

With the **Continuous** model, the optimal speeds are non-rational values, and we obtain

\[
s_1 = \frac{2}{3} (3 + 35^{1/3}) \approx 4.18; \quad s_2 = s_1 \times \frac{2}{35^{1/3}} \approx 2.56;
\]

\[
s_3 = s_4 = s_1 \times \frac{3}{35^{1/3}} \approx 3.83.
\]

- **Discrete**: \( (s_1 = 2, s_2 = 5, s_3 = 6) \) \( E^{(d)}_{\text{opt}} = 170 \).

- **Incremental**: \( (\delta = 2, s_{\text{min}} = 2, s_{\text{max}} = 6) \) \( E^{(i)}_{\text{opt}} = 128 \).

- **VDD-Hopping**: \( (s_1 = 2, s_2 = 5, s_3 = 6) \) \( E^{(v)}_{\text{opt}} = 144 \).
Example

- **Continuous:** \( s_{\text{max}} = 6 \) \( E_{\text{opt}}^{(c)} \approx 109.6 \).

- **Discrete:** \( s_1 = 2, s_2 = 5, s_3 = 6 \) \( E_{\text{opt}}^{(d)} = 170 \).

  For the **Discrete** model, if we execute all tasks at speed \( s_2^{(d)} = 5 \), we obtain an energy \( E = 8 \times 5^2 = 200 \). A better solution is obtained with \( s_1 = s_3^{(d)} = 6, s_2 = s_3 = s_1^{(d)} = 2 \) and \( s_4 = s_2^{(d)} = 5 \), which turns out to be optimal.

- **Incremental:** \( \delta = 2, s_{\text{min}} = 2, s_{\text{max}} = 6 \) \( E_{\text{opt}}^{(i)} = 128 \).

- **Vdd-Hopping:** \( s_1 = 2, s_2 = 5, s_3 = 6 \) \( E_{\text{opt}}^{(v)} = 144 \).
Example

- **Continuous**: \( s_{\text{max}} = 6 \) \( E_{\text{opt}}^{(c)} \approx 109.6 \).
- **Discrete**: \( s_1 = 2, s_2 = 5, s_3 = 6 \) \( E_{\text{opt}}^{(d)} = 170 \).
- **Incremental**: \( \delta = 2, s_{\text{min}} = 2, s_{\text{max}} = 6 \) \( E_{\text{opt}}^{(i)} = 128 \).

For the Incremental model, the reasoning is similar to the Discrete case, and the optimal solution is obtained by an exhaustive search: all tasks should be executed at speed \( s_2^{(i)} = 4 \).

- **VDD-Hopping**: \( s_1 = 2, s_2 = 5, s_3 = 6 \) \( E_{\text{opt}}^{(v)} = 144 \).
Example

- **Continuous**: \( s_{\text{max}} = 6 \) \( E_{\text{opt}}^{(c)} \simeq 109.6 \).

- **Discrete**: \( s_1 = 2, s_2 = 5, s_3 = 6 \) \( E_{\text{opt}}^{(d)} = 170 \).

- **Incremental**: \( \delta = 2, s_{\text{min}} = 2, s_{\text{max}} = 6 \) \( E_{\text{opt}}^{(i)} = 128 \).

- **VDD-Hopping**: \( s_1 = 2, s_2 = 5, s_3 = 6 \) \( E_{\text{opt}}^{(v)} = 144 \).

With the VDD-Hopping model, we set \( s_1 = s_2^{(d)} = 5 \); for the other tasks, we run part of the time at speed \( s_2^{(d)} = 5 \), and part of the time at speed \( s_1^{(d)} = 2 \) in order to use the idle time and lower the energy consumption.
Example

- **Continuous:** \( s_{\text{max}} = 6 \) \( E_{\text{opt}}^{(c)} \approx 109.6 \).
- **Discrete:** \( s_1 = 2, s_2 = 5, s_3 = 6 \) \( E_{\text{opt}}^{(d)} = 170 \).
- **Incremental:** \( \delta = 2, s_{\text{min}} = 2, s_{\text{max}} = 6 \) \( E_{\text{opt}}^{(i)} = 128 \).
- **Vdd-Hopping:** \( s_1 = 2, s_2 = 5, s_3 = 6 \) \( E_{\text{opt}}^{(v)} = 144 \).
Outline

1. Introduction and motivation: energy
2. Revisiting the greedy algorithm for independent jobs
3. Reclaiming the slack of a schedule
   - Models
   - Example
   - Complexity results
4. Conclusion
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

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

- $\text{MinEnergy}(G,D)$ can be solved in polynomial time when $G$ is a tree.

- $\text{MinEnergy}(G,D)$ can be solved in polynomial time when $G$ is a series-parallel graph (assuming $s_{\text{max}} = +\infty$).
Linear program for **VDD-HOPPING**

**Definition**

\[ G, \, n \text{ tasks, } D \text{ deadline;} \]
\[ s_1, \ldots, s_m \text{ be the set of possible processor speeds;} \]
\[ t_i \text{ is the finishing time of the execution of task } T_i; \]
\[ \alpha(i,j) \text{ is the time spent at speed } s_j \text{ for executing task } T_i \]

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:

\[
\min \left( \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha(i,j) s_j^3 \right)
\]
The constraints are:

∀1 ≤ i ≤ n, ti ≤ D: the deadline is not exceeded by any task;
∀1 ≤ i, i′ ≤ n s.t. T_i → T_{i′}, ti + ∑j=1m α(i′, j) ≤ t_{i′}: a task cannot start before its predecessor has completed its execution;
∀1 ≤ i ≤ n, ∑j=1m α(i, j) × sj ≥ wi: task Ti is completely executed;
∀1 ≤ i ≤ n, ti ≥ ∑j=1m α(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 \( \text{MinEnergy}(G,D) \) 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 $\text{MinEnergy}(G,D)$ with the Continuous model can be approximated within a factor $(1 + \frac{\delta}{s_{\min}})^2$ in the Incremental model with speed increment $\delta$. 
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

1. Introduction and motivation: energy
2. Revisiting the greedy algorithm for independent jobs
3. Reclaiming the slack of a schedule
4. Conclusion
What we had:

What we aim at:

Energy-efficient scheduling + frequency scaling
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)