Resilient and energy-aware algorithms

Anne Benoit

ENS Lyon

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

CR02 - 2015/2016
Motivation

- Need of **resilient algorithms** (see classes 3-7)
- Need of **energy-aware algorithms** (see classes 8-9)

... And need to combine both! DVFS has an impact on resilience, so both problematics are linked... Also, does energy have an impact on the optimal checkpointing period?
Motivation

- Need of **resilient algorithms** (see classes 3-7)
- Need of **energy-aware algorithms** (see classes 8-9)

... And need to combine both! DVFS has an impact on resilience, so both problematics are linked... Also, does energy have an impact on the optimal checkpointing period?
Outline

1. Optimal checkpointing period: time vs energy
   - Framework
   - Optimal period for execution time
   - Optimal period for energy
   - Experiments

2. Checkpointing and energy consumption

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Energy: a crucial issue

- Data centers ("Cloud Begins with Coal", M. Mills)
  - 250 – 350 TWh in 2013
  - ≈ consumption of Turkey (242), Spain (267), or Italy (309)
  - ≈ 530 Mt of CO₂ (carbontrust) – > Canada

- ~ crucial for both environmental and economical reasons

- Coordinated periodic checkpointing: what is the optimal checkpointing period if you optimize for Energy consumption?
- Is there a tradeoff between optimizing for Energy and optimizing for Time?
Outline

1. Optimal checkpointing period: time vs energy
   - Framework
   - Optimal period for execution time
   - Optimal period for energy
   - Experiments

2. Checkpointing and energy consumption

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Power model

- $P_{\text{Static}}$: base power (platform switched on)
  - Trend: goes down (w.r.t. other powers)
- $P_{\text{Cal}}$: overhead due to CPU (computations)
- $P_{\text{I/O}}$: overhead due to file I/O (checkpoint or recovery)
- $P_{\text{Down}}$: overhead when one machine is down (rebooting)

Coordinated checkpointing

- Periodic checkpointing policy of period $T$
- Independent and identically distributed failures
- Applies to a single processor with MTBF $\mu = \mu_{\text{ind}}$
- Applies to a platform with $p$ processors with MTBF $\mu = \frac{\mu_{\text{ind}}}{p}$
  - tightly-coupled application
  - progress $\iff$ all processors available
Cost of checkpointing

**General model:** while a checkpoint is taken, computations are slowed-down: during a checkpoint of duration $C$, the same amount of computation is done as during a time $\omega C$ without checkpointing ($0 \leq \omega \leq 1$).
Outline

1. Optimal checkpointing period: time vs energy
   - Framework
   - Optimal period for execution time
   - Optimal period for energy
   - Experiments

2. Checkpointing and energy consumption

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Expected execution time

- $\mathcal{T}_{\text{base}}$ execution time without any overhead
- $\mathcal{T}_{\text{final}} = \mathcal{T}_{\text{ff}} + \mathcal{T}_{\text{fails}}$ total execution time
  - Time for fault-free execution
    \[ \mathcal{T}_{\text{ff}} = \mathcal{T}_{\text{base}} \frac{T}{T - (1 - \omega)C} \]
  - Time lost due to failures
    \[ \mathcal{T}_{\text{fails}} = \frac{\mathcal{T}_{\text{final}}}{\mu} (D + R + \text{RE-EXEC}) \]
Computation of the optimal checkpointing period in the non-blocking case:
See Course 4, Section 4: *Assessing protocols at scale*
(with $\alpha$ instead of $\omega$)

$$T_{\text{Time}}^{\text{opt}} = \sqrt{2(1 - \omega)C(\mu - (D + R + \omega C))}$$
Outline

1. Optimal checkpointing period: time vs energy
   - Framework
   - Optimal period for execution time
   - Optimal period for energy
   - Experiments

2. Checkpointing and energy consumption

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Consumed energy

\[ E_{\text{final}} = T_{\text{Cal}} P_{\text{Cal}} + T_{\text{I/O}} P_{\text{I/O}} + T_{\text{Down}} P_{\text{Down}} + T_{\text{final}} P_{\text{Static}} \]

= \left( T_{\text{base}} + \frac{T_{\text{final}}}{\mu} \left( \omega C + \frac{T^2 - C^2}{2T} + \frac{\omega C^2}{2T} \right) \right) P_{\text{Cal}}

+ \left( \frac{T_{\text{final}}}{\mu} \left( R + \frac{C^2}{2T} \right) \right) P_{\text{I/O}}

+ \frac{T_{\text{final}}}{\mu} D P_{\text{Down}} + T_{\text{final}} P_{\text{Static}}

\[ T_{\text{final}} \neq T_{\text{Cal}} + T_{\text{I/O}} + T_{\text{Down}}, \text{ unless } \omega = 0 \]

CPU and I/O activities are overlapped (and both consumed) when checkpointing
\[ P_{\text{Cal}} = \alpha P_{\text{Static}}, \quad P_{\text{I/O}} = \beta P_{\text{Static}}, \quad P_{\text{Down}} = \gamma P_{\text{Static}} \]

\[
\frac{(T - a)^2(b - \frac{T}{2\mu})^2}{P_{\text{Static}} T_{\text{base}}} E'_{\text{final}} = \frac{-abT^2}{\mu} \left( (\alpha \omega C + \beta R + \gamma D + \mu) + \frac{\alpha T^2}{2} + \frac{\alpha(1 - \omega)C^2}{2T} + \frac{\beta C^2}{2T} \right) \\
+ \frac{(T - a)(b - \frac{T}{2\mu})}{2\mu} \left( \alpha + \frac{\alpha(1 - \omega)C^2 - \beta C^2}{T} \right) - \beta C \left( b - \frac{T}{2\mu} \right)^2 \\
= T^3 \left( \frac{1}{4\mu} - \frac{1}{4\mu^2} \right) + T^2 \left( \frac{\alpha \omega C + \beta R + \gamma D + \mu}{2\mu^2} + \frac{b + \frac{a}{2\mu}}{2\mu^2} - \frac{\beta C}{4\mu^2} + \frac{1}{2\mu} \right) \\
+ T \left( \frac{-ab}{2\mu} - \frac{ab}{2\mu} + \frac{\beta C b}{\mu} - 2 \left( \frac{(\alpha(1 - \omega) - \beta)C^2}{4\mu^2} \right) - \beta C b^2 \right) \\
- \frac{ab(\alpha \omega C + \beta R + \gamma D + \mu)}{\mu} - \left( \frac{b}{2\mu} - \frac{a}{4\mu^2} \right) \left( \alpha(1 - \omega) - \beta \right) C^2 \\
+ \frac{1}{T} \left( (\alpha(1 - \omega) - \beta) \frac{C}{2\mu} - (\alpha(1 - \omega) - \beta) \frac{C}{2\mu} \right) \\
= T^2 \left( \frac{\alpha \omega C + \beta R + \gamma D}{2\mu^2} + \frac{b + \frac{\beta C}{4\mu^2}}{2\mu^2} + \frac{1}{2\mu} \right) \\
+ T \left( \frac{(\beta C - a) b}{\mu} - 2 \left( \frac{(\alpha(1 - \omega) - \beta)C^2}{4\mu^2} \right) \right) \\
- \frac{ab(\alpha \omega C + \beta R + \gamma D + \mu)}{\mu} - \beta C b^2 \\
+ \left( \frac{b}{2\mu} + \frac{a}{4\mu^2} \right) \left( \alpha(1 - \omega) - \beta \right) C^2 .
\]
\[ P_{\text{Cal}} = \alpha P_{\text{Static}}, \quad P_{\text{I/O}} = \beta P_{\text{Static}}, \quad P_{\text{Down}} = \gamma P_{\text{Static}} \]

\[ \frac{(T - a)^2 (b - T)^2}{P_{\text{Static}} T_{\text{base}}} = \frac{-abT^2}{\mu} \left( (\alpha \omega C + \beta R + \gamma D + \mu) + \frac{\alpha T + \beta T^2}{2T} \right) + \frac{\alpha T - a}{\mu} \left( \alpha + \frac{\alpha(1 - \omega - \beta)C^2}{4\mu^2} \right) - \beta C \left( b - \frac{T}{2\mu} \right)^2 \]

\[ = T^3 \left( \frac{1}{4\mu} - \frac{1}{4\mu^2} + \frac{\alpha \omega C + \beta R + \gamma D + \mu}{2\mu^2} + \frac{b + a}{2\mu} - \frac{\beta C}{4\mu^2} + \frac{1}{2\mu} \right) + T \left( \frac{\alpha T - a}{\mu} + \frac{\beta C}{\mu} - \left( \frac{b + a}{2\mu} - \frac{\beta C}{4\mu^2} \right) \left( \alpha(1 - \omega - \beta)C^2 \right) - \beta C b^2 \right) \]

\[ + \frac{1}{T} \left( \frac{\alpha(1 - \omega - \beta)}{2\mu} \frac{C}{2\mu} - \frac{\alpha(1 - \omega - \beta)}{4\mu^2} \right) \frac{C}{2\mu} + \frac{1}{T} \left( \frac{\beta C - a}{\mu} b - \frac{2}{4\mu^2} (\alpha(1 - \omega - \beta)C^2) \right) \]

We let Maple compute \( \tau_{\text{opt}}^{\text{Energy}} \)
Outline

1. **Optimal checkpointing period: time vs energy**
   - Framework
   - Optimal period for execution time
   - Optimal period for energy
   - Experiments

2. Checkpointing and energy consumption

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Parameters: power

\[ \rho = \frac{P_{\text{Static}} + P_{\text{I/O}}}{P_{\text{Static}} + P_{\text{Cal}}} = \frac{1 + \beta}{1 + \alpha} \]

- 20 Mega-watts for Exascale platform with $10^6$ nodes
- Nominal power = 20 milli-watts per node
- $1/2 \rightarrow 1/4$ of that power in static consumption

Scenario 1: \( P_{\text{Static}} = 10, \ P_{\text{Cal}} = 10, \ P_{\text{I/O}} = 100 \Rightarrow \rho = 5.5 \)

Scenario 2: \( P_{\text{Static}} = 5, \ P_{\text{Cal}} = 10, \ P_{\text{I/O}} = 100 \Rightarrow \rho = 7 \)
Parameters: resilience

- **MTBF**
  - $N = 45,208$ processors: one fault per day
  - Individual (processor) MTBF $\mu_{\text{ind}} \approx 125$ years.
  - Total number of processors $N$: from $N = 219,150$ to $N = 2,191,500 \Rightarrow \mu = 300$ min down to $\mu = 30$ min
  - $C = R = 10$ min, $D = 1$ min, and $\omega = 1/2$.  

Impact of ratio $\rho$

How much slower, if we optimize for energy instead of optimizing for time
Impact of ratio $\rho$

How much more energy consumption, if we optimize for time instead of optimizing for energy.
How much slower, if we optimize for energy instead of optimizing for time

How much more energy consumption, if we optimize for time instead of optimizing for energy
Scalability ($\rho = 5.5$)

\[ \mu = 120 \text{ min for } 10^6 \text{ nodes, } C = R = 1 \text{ min, } D = 0.1 \text{ min, } \omega = 1/2 \]
Scalability ($\rho = 7$)

$\mu = 120$ min for $10^6$ nodes, $C = R = 1$ min, $D = 0.1$ min, $\omega = 1/2$
Conclusion

- Coordinated checkpointing, non-blocking
- Different optimal periods for time and energy
- Save more than 20% of energy with 10% increase in time
- Expect more gains for large-scale platforms

- Variety of resilience and power consumption parameters 😞
- Quite flexible analytical model 😊
- Easy to instantiate for other scenarios 😊
Conclusion

- Coordinated checkpointing, non-blocking
- Different optimal periods for time and energy
- Save more than 20% of energy with 10% increase in time
- Expect more gains for large-scale platforms

- Variety of resilience and power consumption parameters 😞
- Quite flexible analytical model 😊
- Easy to instantiate for other scenarios 😊
Outline

1. Optimal checkpointing period: time vs energy

2. Checkpointing and energy consumption
   - Model for one single chunk
   - Model for multiple chunks and optimization problem
   - Solving the problems
   - Simulations

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Framework

- Execution of a divisible task ($W$ operations)
- Failures may occur
  - Transient failures
  - Resilience through checkpointing
- Objective: minimize expected energy consumption $\mathbb{E}(E)$, given a deadline bound $D$

- Probabilistic nature of failure hits: expectation of energy consumption is natural (average cost over many executions)
- Deadline bound: two relevant scenarios (soft or hard deadline)
Soft vs hard deadline

- Soft deadline: met in expectation, i.e., $\mathbb{E}(T) \leq D$ (average response time)
- Hard deadline: met in the worst case, i.e., $T_{wc} \leq D$
Outline

1. Optimal checkpointing period: time vs energy

2. Checkpointing and energy consumption
   - Model for one single chunk
   - Model for multiple chunks and optimization problem
   - Solving the problems
   - Simulations

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
One single chunk of size $W$

- Checkpoint overhead: execution time $T_C$
- Instantaneous failure rate: $\lambda$

- **First execution** at speed $s$: $T_{\text{exec}} = \frac{W}{s} + T_C$
- **Failure probability**: $P_{\text{fail}} = \lambda T_{\text{exec}} = \lambda\left(\frac{W}{s} + T_C\right)$
- In case of failure: **re-execute** at speed $\sigma$: $T_{\text{reexec}} = \frac{W}{\sigma} + T_C$
- And we assume success after re-execution

\[
E(T) = T_{\text{exec}} + P_{\text{fail}} T_{\text{reexec}} = \left(\frac{W}{s} + T_C\right) + \lambda\left(\frac{W}{s} + T_C\right)\left(\frac{W}{\sigma} + T_C\right)
\]

\[
T_{\text{wc}} = T_{\text{exec}} + T_{\text{reexec}} = \left(\frac{W}{s} + T_C\right) + \left(\frac{W}{\sigma} + T_C\right)
\]
Energy consumption, one single chunk

One single chunk of size $W$

- Checkpoint overhead: energy consumption $E_C$
- First execution at speed $s$: $\frac{W}{s} \times s^3 + E_C = Ws^2 + E_C$
- Re-execution at speed $\sigma$: $W\sigma^2 + E_C$, with probability $P_{\text{fail}}$

$$P_{\text{fail}} = \lambda T_{\text{exec}} = \lambda \left( \frac{W}{s} + T_C \right)$$

- $\mathbb{E}(E) = (Ws^2 + E_C) + \lambda \left( \frac{W}{s} + T_C \right) (W\sigma^2 + E_C)$
Outline

1. Optimal checkpointing period: time vs energy

2. Checkpointing and energy consumption
   - Model for one single chunk
   - Model for multiple chunks and optimization problem
   - Solving the problems
   - Simulations

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Multiple chunks

- Execution times: sum of execution times for each chunk (worst-case or expected)
- Expected energy consumption: sum of expected energy for each chunk

Coherent failure model: consider two chunks \( W_1 + W_2 = W \)
- Probability of failure for first chunk: \( P_{\text{fail}}^1 = \lambda \left( \frac{W_1}{s} + T_C \right) \)
- For second chunk: \( P_{\text{fail}}^2 = \lambda \left( \frac{W_2}{s} + T_C \right) \)
- With a single chunk of size \( W \): \( P_{\text{fail}} = \lambda \left( \frac{W}{s} + T_C \right) \), differs from \( P_{\text{fail}}^1 + P_{\text{fail}}^2 \) only because of extra checkpoint

Trade-off: many small chunks (more \( T_C \) to pay, but small re-execution cost) vs few larger chunks (fewer \( T_C \), but increased re-execution cost)
Optimization problem

- Decisions that should be taken before execution:
  - Chunks: how many \((n)\)? which sizes \((W_i)\) for chunk \(i\)?
  - Speeds of each chunk: first run \((s_i)\)? re-execution \((\sigma_i)\)?

- Input: \(W\), \(T_C\) (checkpointing time), \(E_C\) (energy spent for checkpointing), \(\lambda\) (instantaneous failure rate), \(D\) (deadline)
Optimization problem

- Decisions that should be taken before execution:
  - Chunks: how many \((n)\)? which sizes \((W_i)\) for chunk \(i\)?
  - Speeds of each chunk: first run \((s_i)\)? re-execution \((\sigma_i)\)胱?

- Input: \(W, T_C\) (checkpointing time), \(E_C\) (energy spent for checkpointing), \(\lambda\) (instantaneous failure rate), \(D\) (deadline)
Optimization problem

- Decisions that should be taken before execution:
  - Chunks: how many ($n$)? which sizes ($W_i$ for chunk $i$)?
  - Speeds of each chunk: first run ($s_i$)? re-execution ($\sigma_i$)?

- Input: $W$, $T_C$ (checkpointing time), $E_C$ (energy spent for checkpointing), $\lambda$ (instantaneous failure rate), $D$ (deadline)
Models

- **Chunks**
  - Single chunk of size $W$
  - Multiple chunks ($n$ and $W_i$'s)

- **Speed per chunk**
  - Single speed ($s$)
  - Multiple speeds ($s$ and $\sigma$)

- **Deadline bound**
  - Hard ($T_{wc} \leq D$)
  - Soft ($\mathbb{E}(T) \leq D$)
Outline

1. Optimal checkpointing period: time vs energy

2. Checkpointing and energy consumption
   - Model for one single chunk
   - Model for multiple chunks and optimization problem
   - Solving the problems
   - Simulations

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Single chunk and single speed

Consider first that $s = \sigma$ (single speed): need to find optimal speed

- $\mathbb{E}(E)$ is a function of $s$:
  \[ \mathbb{E}(E)(s) = (Ws^2 + E_C)(1 + \lambda(\frac{W}{s} + T_C)) \]

- Lemma: this function is convex and has a unique minimum $s^*$ (function of $\lambda, W, E_c, T_c$)
  \[ s^* = \frac{\lambda W}{6(1+\lambda T_c)} \left( \frac{-(3\sqrt{3}\sqrt{27a^2-4a-27a+2})^{1/3}}{2^{1/3}} - \frac{2^{1/3}}{(3\sqrt{3}\sqrt{27a^2-4a-27a+2})^{1/3}} - 1 \right) \]
  where $a = \lambda E_C \left( \frac{2(1+\lambda T_c)}{\lambda W} \right)^2$

- $\mathbb{E}(T)$ and $T_{wc}$: decreasing functions of $s$

- Minimum speed $s_{exp}$ and $s_{wc}$ required to match deadline $D$ (function of $D, W, T_c$, and $\lambda$ for $s_{exp}$)

  $\rightarrow$ Optimal speed: maximum between $s^*$ and $s_{exp}$ or $s_{wc}$
Consider now that \( s \neq \sigma \) (multiple speeds): two unknowns

- \( \mathbb{E}(E) \) is a function of \( s \) and \( \sigma \):
  \[
  \mathbb{E}(E)(s, \sigma) = (Ws^2 + EC) + \lambda(\frac{W}{s} + TC)(W\sigma^2 + EC)
  \]

- Lemma: energy minimized when deadline tight (both for wc and exp)
  \[
  \sigma_{\text{exp}} = \frac{\frac{\lambda W}{W_s + TC} - (1 + \lambda TC)}{D - 2TC}, \quad \sigma_{\text{wc}} = \frac{W}{(D - 2TC)s - WS}
  \]

→ Minimization of single-variable function, can be solved numerically (no expression of optimal \( s \))
General problem with multiple chunks

- Divisible task of size $W$
- Split into $n$ chunks of size $W_i$: $\sum_{i=1}^{n} W_i = W$
- Chunk $i$ is executed once at speed $s_i$, and re-executed (if necessary) at speed $\sigma_i$
- Unknowns: $n$, $W_i$, $s_i$, $\sigma_i$

$$\mathbb{E}(E) = \sum_{i=1}^{n} (W_i s_i^2 + E_C) + \lambda \sum_{i=1}^{n} \left( \frac{W_i}{s_i} + T_C \right) (W_i \sigma_i^2 + E_C)$$
Multiple chunks and single speed

With a single speed, $\sigma_i = s_i$ for each chunk

- Theorem: in optimal solution, $n$ equal-sized chunks ($W_i = \frac{W}{n}$), executed at same speed $s_i = s$
  - Proof by contradiction: consider two chunks $W_1$ and $W_2$ executed at speed $s_1$ and $s_2$, with either $s_1 \neq s_2$, or $s_1 = s_2$ and $W_1 \neq W_2$
  - $\Rightarrow$ Strictly better solution with two chunks of size $w = (W_1 + W_2)/2$ and same speed $s$

- Only two unknowns, $s$ and $n$

- Minimum speed with $n$ chunks: $s^*_{\text{exp}}(n) = W \frac{1 + 2\lambda T_C + \sqrt{4 \frac{\lambda D}{n} + 1}}{2(D - n T_C (1 + \lambda T_C))}$

$\rightarrow$ Minimization of double-variable function, can be solved numerically both for expected and hard deadline
Multiple chunks and multiple speeds

Need to find $n$, $W_i$, $s_i$, $\sigma_i$

- With expected deadline:
  - All re-execution speeds are equal ($\sigma_i = \sigma$) and tight deadline
  - All chunks have same size and are executed at same speed

- With hard deadline:
  - If $s_i = s$ and $\sigma_i = \sigma$, then all $W_i$’s are equal
  - Conjecture: equal-sized chunks, same first-execution / re-execution speeds

- $\sigma$ as a function of $s$, bound on $s$ given $n$

→ Minimization of double-variable function, can be solved numerically
Outline

1. Optimal checkpointing period: time vs energy

2. Checkpointing and energy consumption
   - Model for one single chunk
   - Model for multiple chunks and optimization problem
   - Solving the problems
   - Simulations

3. Tri-criteria problem: execution time, reliability, energy

4. Conclusion
Simulation settings

- Large set of simulations: illustrate differences between models
- **Maple** software to solve problems
- We plot relative energy consumption as a function of $\lambda$
  - The lower the better
  - Given a deadline constraint (hard or expected), normalize with the result of single-chunk single-speed
  - Impact of the constraint: normalize expected deadline with hard deadline
- Parameters varying within large ranges
Comparison with single-chunk single-speed

- Results identical for any value of $W/D$

- For expected deadline, with small $\lambda (< 10^{-2})$, using multiple chunks or multiple speeds do not improve energy ratio: re-execution term negligible; increasing $\lambda$: improvement with multiple chunks

- For hard deadline, better to run at high speed during second execution: use multiple speeds; use multiple chunks if frequent failures
Expected vs hard deadline constraint

- **Important differences for single speed models**, confirming previous conclusions: with hard deadline, use multiple speeds.

- **Multiple speeds**: no difference for small $\lambda$: re-execution at maximum speed has little impact on expected energy consumption; increasing $\lambda$: more impact of re-execution, and expected deadline may use slower re-execution speed, hence reducing energy consumption.
Outline

1. Optimal checkpointing period: time vs energy
2. Checkpointing and energy consumption
3. Tri-criteria problem: execution time, reliability, energy
   - Complexity results
   - Heuristics
   - Simulation results
4. Conclusion
Framework

- DAG: $G = (V, E)$
- $n = |V|$ tasks $T_i$ of weight $w_i$
- $p$ identical processors fully connected
- DVFS, Continuous model: interval of available continuous speeds $[s_{\text{min}}, s_{\text{max}}]$
- One speed per task
Makespan

Execution time of $T_i$ at speed $s_i$:

$$d_i = \frac{w_i}{s_i}$$

If $T_i$ is executed twice on the same processor at speeds $s_i$ and $s'_i$:

$$d_i = \frac{w_i}{s_i} + \frac{w_i}{s'_i}$$

Constraint on makespan:

end of execution before deadline $D$

(hard deadline constraint)
Reliability

- **Transient failure**: local, no impact on the rest of the system

  Transient failure rate: Poisson distribution of parameter:

  \[ \lambda(s) = \tilde{\lambda} e^{\tilde{d} \frac{s_{\text{max}} - s}{s_{\text{max}} - s_{\text{min}}}}. \]

- Reliability \( R_i \) of task \( T_i \) as a function of speed \( s_i \):

  \[ R_i(s_i) = e^{-\lambda(s_i)E_x(e(w_i,s_i))} = (1st \ order) 1 - \lambda_0 e^{-d s_i} \times \frac{w_i}{s_i} \]

- Threshold reliability (and hence speed \( s_{\text{rel}} \))

\[ R_i(s) \]

\( R_i(s_{\text{rel}}) \)

\( 1 \)

\( s_{\text{min}} \)

\( s_{\text{rel}} \)

\( s_{\text{max}} \)
Re-execution: a task is re-executed *on the same processor, just after its first execution*

With two executions, reliability $R_i$ of task $T_i$ is:

$$R_i = 1 - (1 - R_i(s_i))(1 - R_i(s'_i))$$

**Constraint on reliability:**

**Reliability:** $R_i \geq R_i(s_{rel})$, and at most one re-execution
Energy

- Energy to execute task $T_i$ once at speed $s_i$:
  \[ E_i(s_i) = w_i s_i^2 \]
  → Dynamic part of classical energy models

- With re-executions, it is natural to take the worst-case scenario:

\[ \text{ENERGY} : E_i = w_i \left( s_i^2 + s_i'^2 \right) \]
Energy and reliability: set of possible speeds

\[ w_i s_i^2 + w_i s_i^2 = 2E_i(s_i) \]

\[ w_i s_i^2 = E_i(s_i) \]
Given $G = (V, E)$
Find
- A schedule of the tasks
- A set of tasks $I = \{i \mid T_i \text{ is executed twice}\}$
- Execution speed $s_i$ for each task $T_i$
- Re-execution speed $s_i'$ for each task in $I$

such that

$$
\sum_{i \in I} w_i(s_i^2 + s_i'^2) + \sum_{i \notin I} w_i s_i^2
$$

is minimized, while meeting reliability and deadline constraints
Outline

1. Optimal checkpointing period: time vs energy
2. Checkpointing and energy consumption
3. Tri-criteria problem: execution time, reliability, energy
   - Complexity results
   - Heuristics
   - Simulation results
4. Conclusion
Complexity results

- **One speed** per task
- Re-execution **at same speed** as first execution, i.e., $s_i = s'_i$

- **Tri-Crit-Cont** is NP-hard even for a linear chain, but not known to be in NP (because of continuous model)
- Polynomial-time solution for a fork
Complexity results with **VDD-HOPPING**

- Each task is computed using **at most two different speeds**

- **Tri-Crit-Vdd** is NP-complete even for a **linear chain**
# Outline

1. Optimal checkpointing period: time vs energy
2. Checkpointing and energy consumption
3. **Tri-criteria problem: execution time, reliability, energy**
   - Complexity results
   - Heuristics
   - Simulation results
4. Conclusion
Two steps:

- Mapping (NP-hard) → List scheduling
- Speed scaling + re-execution (NP-hard) → Energy reducing

- The list scheduling heuristic maps tasks onto processors at speed $s_{\text{max}}$, and we keep this mapping in step two
- Step two = slack reclamation: use of deceleration and re-execution
Deceleration and re-execution

- **Deceleration**: select a set of tasks that we execute at speed $\max(s_{rel}, s_{\max} \frac{\max_{i=1..n} t_i}{D})$: slowest possible speed meeting both reliability and deadline constraints

- **Re-execution**: greedily select tasks for re-execution
Super-weight (SW) of a task

- SW: sum of the weights of the tasks (including $T_i$) whose execution interval is included into $T_i$’s execution interval
- SW of task slowed down = estimation of the total amount of work that can be slowed down together with that task
Selected heuristics

**A. SUS-Crit**: efficient on DAGs with low degree of parallelism
- Set the speed of every task to $\max(s_{rel}, s_{\max} \frac{\max_{i=1..n} t_i}{D})$
- Sort the tasks of every *critical path* according to their *SW* and try to re-execute them
- Sort all the tasks according to their *weight* and try to re-execute them

**B. SUS-Crit-Slow**: good for highly parallel tasks: re-execute, then decelerate
- Sort the tasks of every *critical path* according to their *SW* and try to re-execute them. If not possible, then try to slow them down
- Sort all tasks according to their *weight* and try to re-execute them. If not possible, then try to slow them down
Outline

1. Optimal checkpointing period: time vs energy
2. Checkpointing and energy consumption
3. Tri-criteria problem: execution time, reliability, energy
   - Complexity results
   - Heuristics
   - Simulation results
4. Conclusion
We compare the impact of:

- the number of processors $p$
- the ratio $D$ of the deadline over the minimum deadline $D_{\text{min}}$ (given by the list-scheduling heuristic at speed $s_{\text{max}}$)

on the output of each heuristic

Results normalized by heuristic running each task at speed $s_{\text{max}}$; the lower the better
Figure: With increasing $p$, $D = 1.2$ (left), $D = 2.4$ (right)

- A better when number of processors is small
- B better when number of processors is large
- Superiority of B for tight deadlines: decelerates critical tasks that cannot be re-executed
Summary

- Tri-criteria energy/makespan/reliability optimization problem
- Various theoretical results
- Two-step approach for polynomial-time heuristics:
  - List-scheduling heuristic
  - Energy-reducing heuristics
- Two complementary energy-reducing heuristics for **Tri-Crit-Cont**
Outline

1. Optimal checkpointing period: time vs energy
2. Checkpointing and energy consumption
3. Tri-criteria problem: execution time, reliability, energy
4. Conclusion
Resilience and energy consumption are two of the main challenges for Exascale platforms.

Revisiting checkpointing techniques for reliability while minimizing energy consumption.

Tri-criteria heuristics aiming at minimizing the energy consumption, with re-execution to deal with reliability.

... Still a lot of challenging algorithmic problems on these hot topics 😊
Conclusion

- **Resilience** and **energy consumption** are two of the main challenges for Exascale platforms.
- Revisiting **checkpointing techniques** for reliability while minimizing energy consumption.
- Tri-criteria heuristics aiming at minimizing the energy consumption, with **re-execution** to deal with reliability.

... Still a lot of challenging algorithmic problems on these hot topics 😊
Bibliography

- Optimal checkpointing period: time vs energy (Aupy, Benoit, Hérault, Robert, Dongarra, 2013)

- Energy-aware checkpointing of divisible tasks with soft or hard deadlines (Aupy, Benoit, Melhem, Renaud-Goud, Robert, 2013)

- Energy-aware scheduling under reliability and makespan constraints (Aupy, Benoit, Robert, 2012)