Resilient and energy-aware algorithms

Anne Benoit

ENS Lyon

Anne.Benoit@ens-lyon.fr

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

CR02 - 2016/2017

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

Energy: a crucial issue

- Data centers ("Cloud Begins with Coal", M. Mills)
  - $250 - 350 TWh$ in 2013
    $\approx$ consumption of Turkey (242), Spain (267), or Italy (309)
  - $\approx 530 Mt$ of CO_2 (carbontrust) $-->$ Canada
- $\leadsto$ 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

Power model

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

Meneses, Sarood and Kalé:

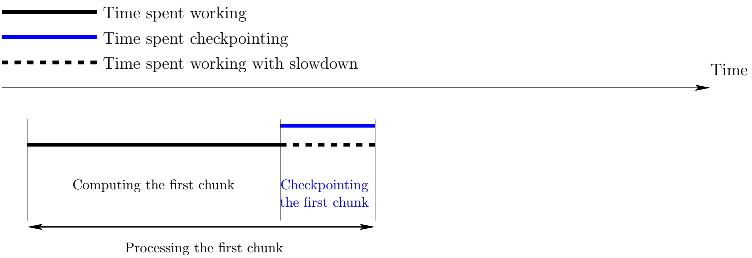
E. Meneses, O. Sarood, and L.V. Kalé, "Assessing Energy Efficiency of Fault Tolerance Protocols for HPC Systems," in Proceedings of the 2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2012), New York, USA, October 2012.

- Base power $L = \mathcal{P}_{\text{Static}}$
- Maximum power $H = \mathcal{P}_{\text{Static}} + \mathcal{P}_{\text{Cal}}$
- $\mathcal{P}_{\text{I/O}} = 0$ (and $\mathcal{P}_{\text{Down}} = 0$)

Coordinated checkpointing

- Periodic checkpointing policy of period T
- Independent and identically distributed failures
- Applies to a single processor with MTBF $\mu = \mu_{ind}$
- Applies to a platform with p processors with MTBF $\mu = \frac{\mu_{ind}}{p}$
  - tightly-coupled application
  - progress $\Leftrightarrow$ 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 ωC without checkpointing ($0 \leq \omega \leq 1$).

Outline

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

$$\mathcal{T}_{\text{fails}} = \frac{\mathcal{T}_{\text{final}}}{\mu}(D + R + \textsc{Re-Exec})$$

ALGOT: Strategy with $\mathcal{T}_{\text{Time}}^{\text{opt}}$

Computation of the optimal checkpointing period in the non-blocking case:
See Course 4, Section 4: *Assessing protocols at scale* (with α instead of ω)

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

Outline

Consumed energy

$$
\begin{aligned}
\mathcal{E}_{\text{final}} &= \mathcal{T}_{\text{Cal}}\mathcal{P}_{\text{Cal}} + \mathcal{T}_{\text{I/O}}\mathcal{P}_{\text{I/O}} + \mathcal{T}_{\text{Down}}\mathcal{P}_{\text{Down}} + \mathcal{T}_{\text{final}}\mathcal{P}_{\text{Static}} \\
&= \left(\mathcal{T}_{\text{base}} + \frac{\mathcal{T}_{\text{final}}}{\mu}\left(\omega C + \frac{T^2 - C^2}{2T} + \frac{\omega C^2}{2T}\right)\right)\mathcal{P}_{\text{Cal}} \\
&\quad + \left(\frac{\mathcal{T}_{\text{final}}}{\mu}\left(R + \frac{C^2}{2T}\right) + C\frac{\mathcal{T}_{\text{base}}}{T - (1-\omega)C}\right)\mathcal{P}_{\text{I/O}} \\
&\quad + \frac{\mathcal{T}_{\text{final}}}{\mu}D\mathcal{P}_{\text{Down}} + \mathcal{T}_{\text{final}}\mathcal{P}_{\text{Static}}
\end{aligned}
$$

$\mathcal{T}_{\text{final}} \neq \mathcal{T}_{\text{Cal}} + \mathcal{T}_{\text{I/O}} + \mathcal{T}_{\text{Down}}$, unless $\omega = 0$
CPU and I/O activities are overlapped (and both consumed) when checkpointing

ALGOE: Strategy with $\mathcal{T}^{\text{opt}}_{\text{Energy}}$

$$\mathcal{P}_{\text{Cal}} = \alpha\mathcal{P}_{\text{Static}},\ \mathcal{P}_{\text{I/O}} = \beta\mathcal{P}_{\text{Static}},\ \mathcal{P}_{\text{Down}} = \gamma\mathcal{P}_{\text{Static}}$$

$$\begin{aligned}
\frac{(T-a)^2\left(b-\frac{T}{2\mu}\right)^2}{\mathcal{P}_{\text{Static}}\mathcal{T}_{\text{base}}}\mathcal{E}'_{\text{final}} &= \frac{-ab+\frac{T^2}{2\mu}}{\mu}\left((\alpha\omega C+\beta R+\gamma D+\mu)+\frac{\alpha T}{2}+\frac{\alpha(1-\omega)C^2}{2T}+\frac{\beta C^2}{2T}\right)\\
&\quad+\frac{(T-a)(b-\frac{T}{2\mu})}{2\mu}\left(\alpha+\frac{\alpha(1-\omega)C^2-\beta C^2}{T}\right)-\beta C\left(b-\frac{T}{2\mu}\right)^2\\
&= T^3\left(\frac{1}{4\mu}-\frac{1}{4\mu}\right)+T^2\left(\frac{\alpha\omega C+\beta R+\gamma D}{2\mu^2}+\frac{b+\frac{a}{2\mu}}{2\mu}-\frac{\beta C}{4\mu^2}+\frac{1}{2\mu}\right)\\
&\quad+T\left(-\frac{ab}{2\mu}-\frac{ab}{2\mu}+\frac{\beta Cb}{\mu}-2\frac{(\alpha(1-\omega)-\beta)C^2}{4\mu^2}\right)-\beta Cb^2\\
&\quad-\frac{ab(\alpha\omega C+\beta R+\gamma D+\mu)}{\mu}-\left(\frac{b}{2\mu}-\frac{a}{4\mu^2}\right)(\alpha(1-\omega)-\beta)C^2\\
&\quad+\frac{1}{T}\left((\alpha(1-\omega)-\beta)\frac{C}{2\mu}-(\alpha(1-\omega)-\beta)\frac{C}{2\mu}\right)\\
&= T^2\left(\frac{\alpha\omega C+\beta R+\gamma D}{2\mu^2}+\frac{b}{2\mu}+\frac{a-\beta C}{4\mu^2}+\frac{1}{2\mu}\right)\\
&\quad+T\left(\frac{(\beta C-a)b}{\mu}-2\frac{(\alpha(1-\omega)-\beta)C^2}{4\mu^2}\right)\\
&\quad-\frac{ab(\alpha\omega C+\beta R+\gamma D+\mu)}{\mu}-\beta Cb^2\\
&\quad+\left(\frac{b}{2\mu}+\frac{a}{4\mu^2}\right)(\alpha(1-\omega)-\beta)C^2\ .
\end{aligned}$$

ALGOE: Strategy with $\mathcal{T}^{\text{opt}}_{\text{Energy}}$

$$\mathcal{P}_{\text{Cal}} = \alpha \mathcal{P}_{\text{Static}},\ \mathcal{P}_{\text{I/O}} = \beta \mathcal{P}_{\text{Static}},\ \mathcal{P}_{\text{Down}} = \gamma \mathcal{P}_{\text{Static}}$$

$$
\begin{aligned}
\frac{(T-a)^2 \left(b \cdots\right)^2}{\mathcal{P}_{\text{Static}} \mathcal{T}_{\text{base}}}
&= \frac{-ab + \frac{T^2}{2\mu}}{\mu} \left((\alpha\omega C + \beta R + \gamma D + \mu) + \frac{\alpha T}{\cdots} \cdots \omega) C^2}{2T} + \frac{\beta C^2}{2T} \right) \\
&\quad + \frac{\cdots (b - \frac{T}{2\mu})}{\cdots} \left(\alpha + \frac{\alpha(1 - \cdots C^2}{\cdots} \right) - \beta C \left(b - \frac{T}{2\mu} \right)^2 \\
&= T^3 \left(\frac{1}{4\mu} - \frac{1}{4\mu \cdots} \cdots \frac{\alpha\omega C + \beta R + \gamma D}{2\mu^2} + \frac{b + \frac{a}{2\mu}}{2\mu} - \frac{\beta C}{4\mu^2} + \frac{1}{2\mu} \right) \\
&\quad + T \left(\cdots - \frac{ab}{2\mu} + \frac{\beta C \cdots}{\mu} \cdots \frac{(\alpha(1-\omega) - \beta) C^2}{4\mu^2} \right) - \beta C b^2 \\
&\quad \cdots \frac{(\alpha\omega C + \beta R + \gamma D + \mu)}{\mu} - \left(\frac{b}{2\mu} \cdots \right) (\alpha(1-\omega) - \beta) C^2 \\
&\quad + \frac{1}{T} \left((\alpha(1-\omega) - \beta) \frac{C}{2\mu} - (\alpha(1 - \cdots - \beta) \frac{C}{2\mu} \right) \\
&= T^2 \left(\frac{\alpha\omega C + \beta R + \gamma D}{2\mu^2} + \frac{b}{2\mu} + \frac{a - \beta C}{4\mu^2} + \frac{1}{2\mu} \right) \\
&\quad + T \left(\frac{(\beta C - a) b}{\mu} - 2 \frac{(\alpha(1-\omega) - \beta) C^2}{4\mu^2} \right)
\end{aligned}
$$

We let Maple compute $\mathcal{T}^{\text{opt}}_{\text{Energy}}$

Outline

Parameters: power

$$\rho = \frac{\mathcal{P}_{\mathsf{Static}} + \mathcal{P}_{\mathsf{I/O}}}{\mathcal{P}_{\mathsf{Static}} + \mathcal{P}_{\mathsf{Cal}}} = \frac{1+\beta}{1+\alpha}$$

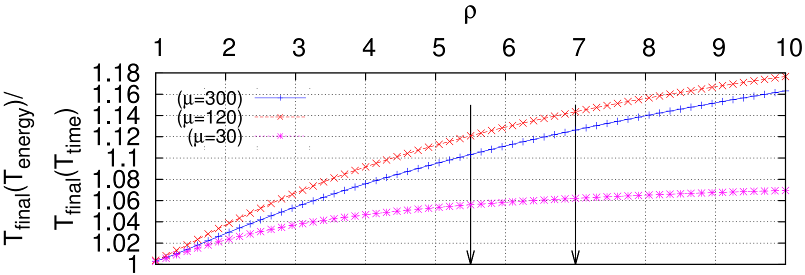
- 20 Mega-watts for Exascale platform with 10^6 nodes
- Nominal power = 20 milli-watts per node
- $1/2 \longrightarrow 1/4$ of that power in static consumption
- "I/O an order of magnitude more than computing" (J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in the 9th Int. Conf. High Performance Computing for Computational Science, 2011)

- Scenario 1: $\mathcal{P}_{\mathsf{Static}} = 10$, $\mathcal{P}_{\mathsf{Cal}} = 10$, $\mathcal{P}_{\mathsf{I/O}} = 100 \Rightarrow \rho = 5.5$
- Scenario 2: $\mathcal{P}_{\mathsf{Static}} = 5$, $\mathcal{P}_{\mathsf{Cal}} = 10$, $\mathcal{P}_{\mathsf{I/O}} = 100 \Rightarrow \rho = 7$

Parameters: resilience

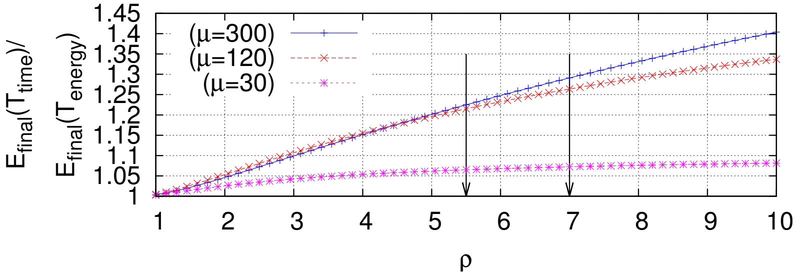
- MTBF
  - $N = 45,208$ processors: one fault per day
  - Individual (processor) MTBF $\mu_{\text{ind}} \approx 125$ years.
  - Total number of processors N: from $N = 219,150$ to $N = 2,191,500 \Rightarrow \mu = 300$ min down to $\mu = 30$ min
- $C = R = 10$ min, $D = 1$ min, and $\omega = 1/2$.

Impact of ratio ρ



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

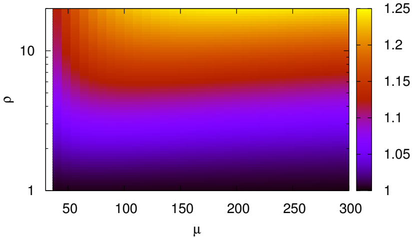
Impact of ratio ρ



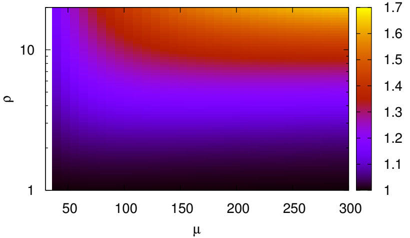
How much more energy consumption, if we optimize for time instead of optimizing for energy

ALGOT over ALGOE

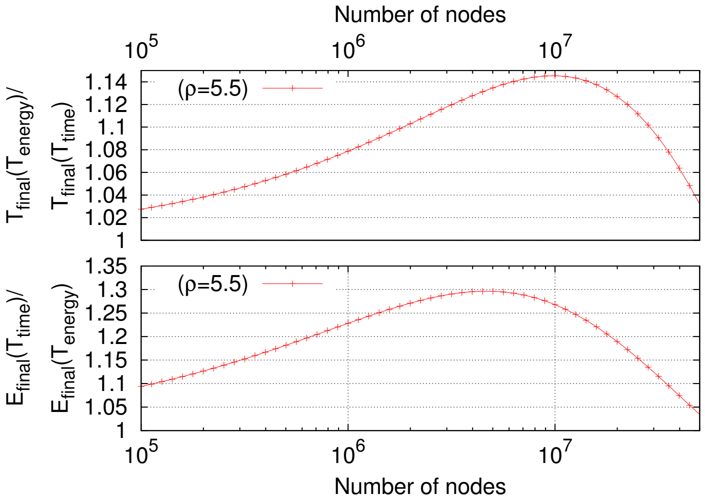
How much slower, if we optimize for energy instead of optimizing for time



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

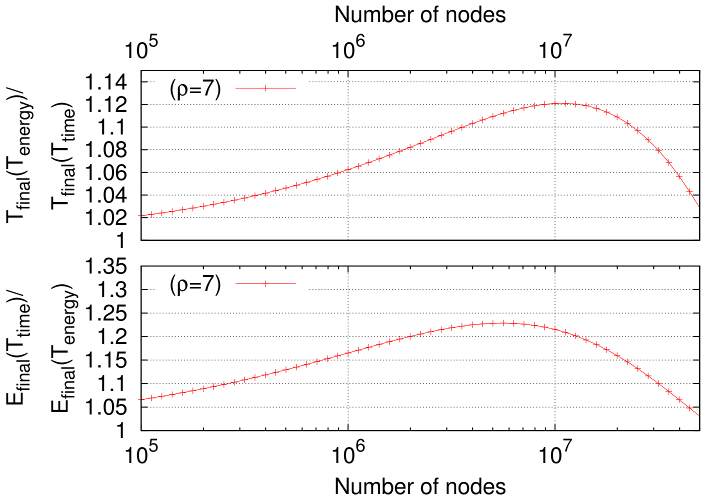


Scalability ($\rho = 5.5$)



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

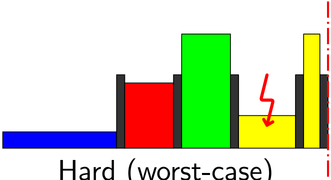
Framework

- 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)

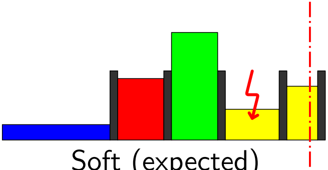
Soft vs hard deadline

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



Hard (worst-case)

VS



Soft (expected)

Outline

Execution time, one single chunk

One single chunk of size W

- Checkpoint overhead: execution time T_C
- Instantaneous failure rate: λ

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

- $\mathbb{E}(T) = T_{\text{exec}} + P_{\text{fail}} T_{\text{reexec}} = (\frac{W}{s} + T_C) + \lambda(\frac{W}{s} + T_C)(\frac{W}{\sigma} + T_C)$
- $T_{wc} = T_{\text{exec}} + T_{\text{reexec}} = (\frac{W}{s} + T_C) + (\frac{W}{\sigma} + T_C)$

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 σ: $W\sigma^2 + E_C$, with probability P_{fail} $\left(P_{\text{fail}} = \lambda T_{\text{exec}} = \lambda(\frac{W}{s} + T_C)\right)$

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

Outline

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

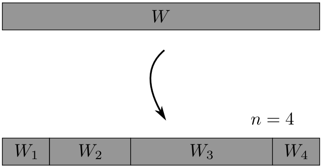
- Coherent failure model: consider two chunks $W_1 + W_2 = W$
- Probability of failure for first chunk: $P^1_{\text{fail}} = \lambda(\frac{W_1}{s} + T_C)$
- For second chunk: $P^2_{\text{fail}} = \lambda(\frac{W_2}{s} + T_C)$
- With a single chunk of size W: $P_{\text{fail}} = \lambda(\frac{W}{s} + T_C)$,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

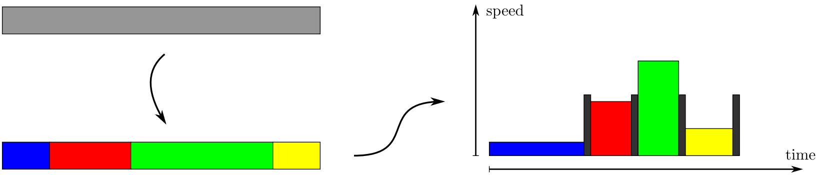
- Decisions that should be taken before execution:
  - Chunks: how many (n)? which sizes (W_i for chunk i)?
  - Speeds of each chunk: first run (s_i)? re-execution (σ_i)?

- Input: W, T_C (checkpointing time), E_C (energy spent for checkpointing), λ (instantaneous failure rate), D (deadline)



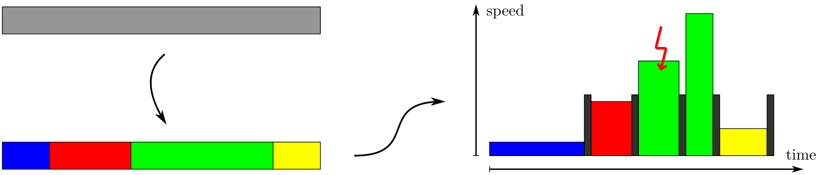
Optimization problem

- 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 (σ_i)?
- Input: W, T_C (checkpointing time), E_C (energy spent for checkpointing), λ (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 (σ_i)?
- Input: W, T_C (checkpointing time), E_C (energy spent for checkpointing), λ (instantaneous failure rate), D (deadline)



Models

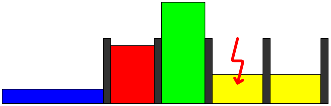
- Chunks

Single chunk of size W

VS

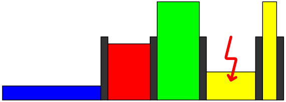
Multiple chunks (n and W_i's)

- Speed per chunk



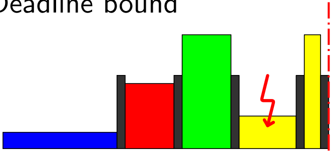
Single speed (s)

VS



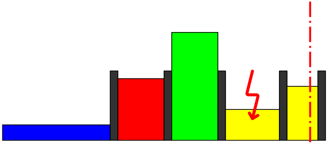
Multiple speeds (s and σ)

- Deadline bound



Hard ($T_{wc} \leq D$)

VS



Soft ($\mathbb{E}(T) \leq D$)

Outline

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 $s^\star$ (function of λ, W, E_c, T_c)
$$s^\star = \frac{\lambda W}{6(1+\lambda T_C)} \left(\frac{-(3\sqrt{3}\sqrt{27a^2-4a}-27a+2)^{1/3}}{2^{1/3}} - \frac{2^{1/3}}{(3\sqrt{3}\sqrt{27a^2-4a}-27a+2)^{1/3}} - 1 \right),$$
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 λ for s_{exp})

$\rightarrow$ Optimal speed: maximum between $s^\star$ and s_{exp} or s_{wc}

Single chunk and multiple speeds

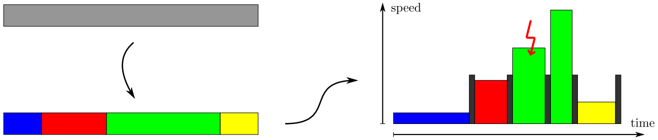
Consider now that $s \neq \sigma$ (multiple speeds): two unknowns

- $\mathbb{E}(E)$ is a function of s and σ:
$$\mathbb{E}(E)(s,\sigma) = (Ws^2 + E_C) + \lambda(\frac{W}{s} + T_C)(W\sigma^2 + E_C)$$
- Lemma: energy minimized when deadline tight (both for wc and exp)
- $\leadsto \sigma$ expressed as a function of s:
$$\sigma_{exp} = \frac{\lambda W}{\frac{D}{\frac{W}{s}+T_C} - (1+\lambda T_C)}, \quad \sigma_{wc} = \frac{W}{(D-2T_C)s - W}s$$

$\rightarrow$ 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 σ_i
- Unknowns: n, W_i, s_i, σ_i
- $$\mathbb{E}(E) = \sum_{i=1}^{n} \left(W_i s_i^2 + E_C\right) + \lambda \sum_{i=1}^{n} \left(\frac{W_i}{s_i} + T_C\right)\left(W_i \sigma_i^2 + E_C\right)$$



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: $s^{\star}_{exp}(n) = W \dfrac{1 + 2\lambda T_C + \sqrt{4\frac{\lambda D}{n} + 1}}{2(D - nT_C(1 + \lambda T_C))}$

$\rightarrow$ 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, σ_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
- σ as a function of s, bound on s given n

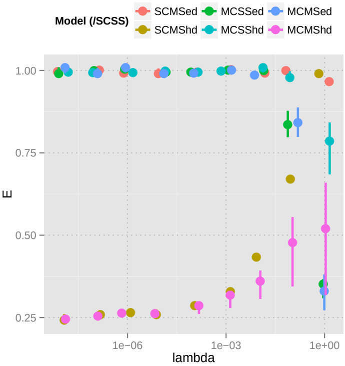
$\rightarrow$ Minimization of double-variable function, can be solved numerically

Outline

Simulation settings

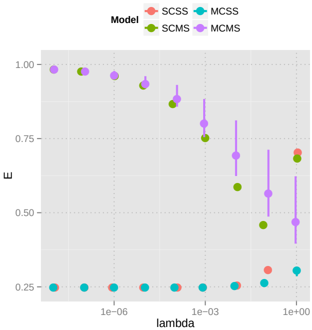
- Large set of simulations: illustrate differences between models
- Maple software to solve problems
- We plot relative energy consumption as a function of λ
  - The lower the better
  - Given a deadline constraint (hard or expected), normalize with the result of single-chunk single-speed
  - Impact 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 λ ($< 10^{-2}$), using multiple chunks or multiple speeds do not improve energy ratio: re-execution term negligible;
  increasing λ: 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 λ: re-execution at maximum speed has little impact on expected energy consumption;
  increasing λ: more impact of re-execution, and expected deadline may use slower re-execution speed, hence reducing energy consumption

Outline

Framework

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

Makespan

Execution time of T_i at speed s_i:

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

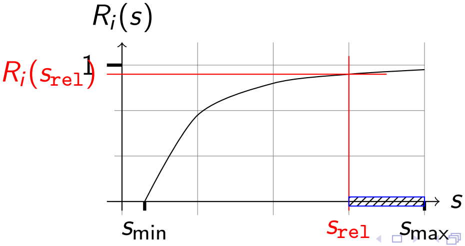
If T_i is executed twice on the same processor at speeds s_i and s_i':

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

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:
  $$\lambda(s) = \tilde{\lambda_0} e^{\tilde{d}\frac{s_{max}-s}{s_{max}-s_{min}}}.$$
- Reliability R_i of task T_i as a function of speed s_i:
  $$R_i(s_i) = e^{-\lambda(s_i)\mathcal{E}xe(w_i,s_i)} =_{(1st\ order)} 1 - \lambda_0 e^{-ds_i} \times \frac{w_i}{s_i}$$
- Threshold reliability (and hence speed $s_{\mathtt{rel}}$)



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

Energy

- Energy to execute task T_i once at speed s_i:

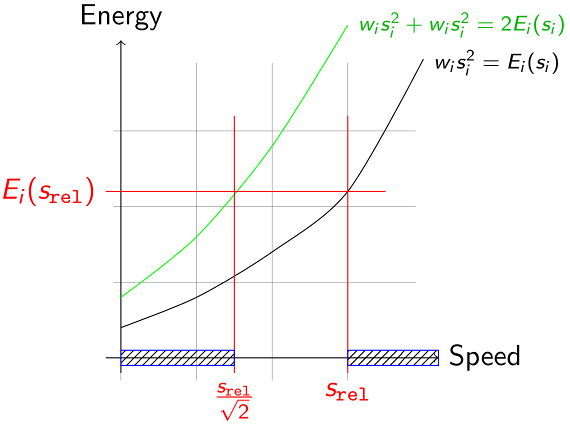
$$E_i(s_i) = w_i s_i^2$$

$\rightarrow$ Dynamic part of classical energy models

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

$$\textsc{Energy} : E_i = w_i \left(s_i^2 + s_i'^2 \right)$$

Energy and reliability: set of possible speeds



Tri-Crit-Cont

Given $\mathcal{G} = (V, E)$
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

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

Energy-reducing heuristics

Two steps:

- Mapping (NP-hard) $\rightarrow$ List scheduling
- Speed scaling + re-execution (NP-hard) $\rightarrow$ Energy reducing

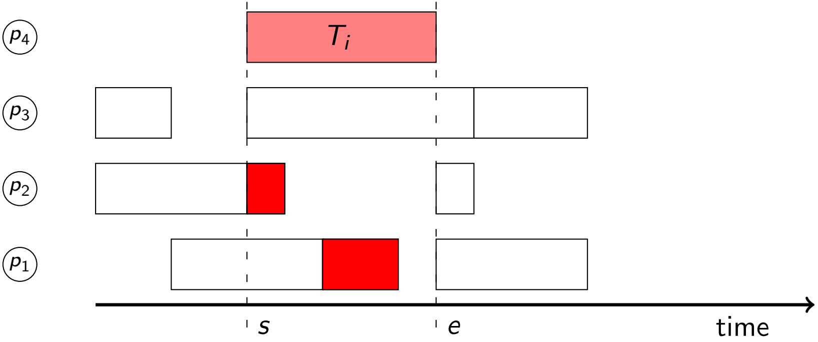
- The list scheduling heuristic maps tasks onto processors at speed $s_{\max}$, and we keep this mapping in step two
- Step two = slack reclamation: use of deceleration and re-execution

Deceleration and re-execution

- Deceleration: select a set of tasks that we execute at speed $\max(s_{\texttt{rel}}, s_{\max}\frac{\max_{i=1..n} t_i}{D})$: 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 $\max(s_{\texttt{rel}}, s_{\max} \frac{\max_{i=1..n} t_i}{D})$
  - 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

Results

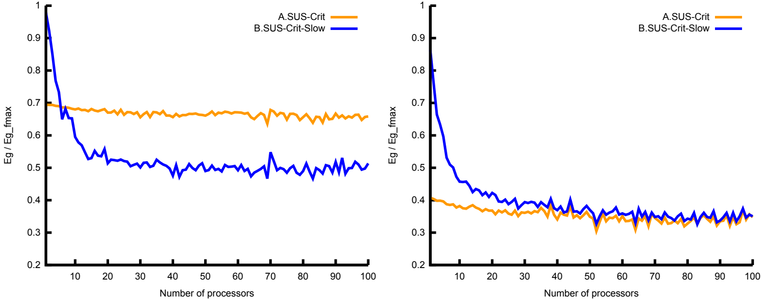
We compare the impact of:

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

on the output of each heuristic

Results normalized by heuristic running each task at speed $s_{\max}$; the lower the better

Results



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

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)