On the complexity of scheduling checkpoints for computational workflows

Yves Robert[1,2,3], Frédéric Vivien[4,1], and
Dounia Zaidouni[4,1]

1. *Ecole Normale Supérieure de Lyon*
2. *Institut Universitaire de France*
3. *University of Tennessee Knoxville*
4. INRIA

June 28, 2012

Motivation

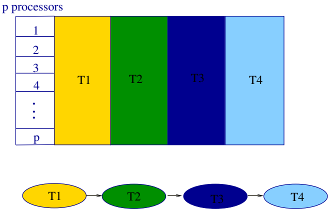
Framework

- Application task graph, a DAG where nodes represent tasks and edges correspond to dependences between them.
- Application DAG to be executed on a failure-prone platform of p identical processors.

Motivation

Framework

- Application task graph, a DAG where nodes represent tasks and edges correspond to dependences between them.
- Application DAG to be executed on a failure-prone platform of p identical processors.
- Each task is executed in parallel on the p processors.



- Resilience provided through coordinated checkpointing.

Objective and Questions

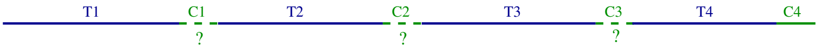
Objective : Minimizing the expectation of the total execution time.

Objective and Questions

Objective : Minimizing the expectation of the total execution time.

Questions :

- In which order should we execute the tasks?
- At the end of the execution of each task Ti, should we perform a checkpoint or should we proceed directly with the computation of another task?



State of the art

Bouguerra et al [1], Daly [3] and Young [4]:

- Periodic checkpointing strategies.

State of the art

Bouguerra et al [1], Daly [3] and Young [4]:

- Periodic checkpointing strategies.

Bouguerra, Trystram, and Wagner [2]:

- Checkpointing strategies for computational tasks with linear chains (with single processor).
- They cannot aim at minimizing the expected execution time.
- Maximizing the amount of work done before the first failure.

State of the art

Bouguerra et al [1], Daly [3] and Young [4]:

- Periodic checkpointing strategies.

Bouguerra, Trystram, and Wagner [2]:

- Checkpointing strategies for computational tasks with linear chains (with single processor).
- They cannot aim at minimizing the expected execution time.
- Maximizing the amount of work done before the first failure.

We solve the original problem that is minimizing the expected execution time (At least for Exponential failures)

Outline

1. Expected time needed to execute a work and to checkpoint it
2. Complexity of the general scheduling problem
3. Dynamic Programming algorithm for linear chains
4. Conclusion and extensions

Outline

1. Expected time needed to execute a work and to checkpoint it
2. Complexity of the general scheduling problem
3. Dynamic Programming algorithm for linear chains
4. Conclusion and extensions

Hypothesis

- Full parallelism: Each task is executed by all the p processors.

Hypothesis

- Full parallelism: Each task is executed by all the p processors.
- Poisson process: Platform failure inter-arrival times follow an Exponential distribution of parameter $\lambda = p\lambda_{proc}$.

Hypothesis

- Full parallelism: Each task is executed by all the p processors.
- Poisson process: Platform failure inter-arrival times follow an Exponential distribution of parameter $\lambda = p\lambda_{proc}$.
- $\mathcal{W}$: Duration of Work

Hypothesis

- Full parallelism: Each task is executed by all the p processors.
- Poisson process: Platform failure inter-arrival times follow an Exponential distribution of parameter $\lambda = p\lambda_{proc}$.
- $\mathcal{W}$: Duration of Work
- C: Checkpoint cost

Hypothesis

- Full parallelism: Each task is executed by all the p processors.
- Poisson process: Platform failure inter-arrival times follow an Exponential distribution of parameter $\lambda = p\lambda_{proc}$.
- $\mathcal{W}$: Duration of Work
- C: Checkpoint cost
- D : Downtime (hardware replacement by spare, or software rejuvenation via rebooting)

Hypothesis

- Full parallelism: Each task is executed by all the p processors.
- Poisson process: Platform failure inter-arrival times follow an Exponential distribution of parameter $\lambda = p\lambda_{proc}$.
- $\mathcal{W}$: Duration of Work
- C: Checkpoint cost
- D : Downtime (hardware replacement by spare, or software rejuvenation via rebooting)
- R: Recovery cost after failure

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$\mathbb{E}(T(\mathcal{W}, C, R)) =$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \mathcal{P}_{\text{succ}}(\mathcal{W} + C)\overbrace{(\mathcal{W} + C)}^{\substack{\text{Time needed}\\ \text{to compute}\\ \text{the work } \mathcal{W}}}$$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \overbrace{\mathcal{P}_{\text{succ}}(\mathcal{W} + C)}^{\substack{\text{Probability} \\ \text{of success}}} (\mathcal{W} + C)$$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \mathcal{P}_{\text{succ}}(\mathcal{W} + C)\,(\mathcal{W} + C)$$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \begin{array}{l} \mathcal{P}_{\mathrm{succ}}(\mathcal{W} + C)\,(\mathcal{W} + C) \\ + \\ (1 - \mathcal{P}_{\mathrm{succ}}(\mathcal{W} + C))\,(\mathbb{E}(T_{lost}(\mathcal{W} + C)) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}, C, R))) \end{array}$$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\mathcal{W} + C)\,(\mathcal{W} + C) \\ + \\ (1 - \mathcal{P}_{\text{succ}}(\mathcal{W} + C))\,(\underbrace{\mathbb{E}(T_{lost}(\mathcal{W} + C))}_{\substack{\text{Time elapsed} \\ \text{before the failure} \\ \text{occured}}} + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}, C, R))) \end{array}$$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\mathcal{W} + C)\,(\mathcal{W} + C) \\ + \\ (1 - \mathcal{P}_{\text{succ}}(\mathcal{W} + C))\,(\mathbb{E}(T_{lost}(\mathcal{W} + C)) + \underbrace{\mathbb{E}(T_{rec})}_{\substack{\text{Time needed} \\ \text{to perform} \\ \text{downtime} \\ \text{and recovery}}} + \mathbb{E}(T(\mathcal{W}, C, R))) \end{array}$$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\mathcal{W}+C)\,(\mathcal{W}+C) \\ + \\ (1-\mathcal{P}_{\text{succ}}(\mathcal{W}+C))\,(\mathbb{E}(T_{lost}(\mathcal{W}+C)) + \mathbb{E}(T_{rec}) + \underbrace{\mathbb{E}(T(\mathcal{W}, C, R))}_{\substack{\text{Time needed} \\ \text{to compute } \mathcal{W} \\ \text{from scratch}}}) \end{array}$$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \begin{array}{l} \mathcal{P}_{\mathrm{succ}}(\mathcal{W}+C)\,(\mathcal{W}+C) \\ + \\ \underbrace{(1-\mathcal{P}_{\mathrm{succ}}(\mathcal{W}+C))}_{\text{Probability of failure}}\,(\mathbb{E}(T_{lost}(\mathcal{W}+C)) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}, C, R))) \end{array}$$

Problem statement

Compute the expected time $\mathbb{E}(T(\mathcal{W}, C, R))$ to execute a work of duration $\mathcal{W}$ followed by a checkpoint of duration C.

- Recursive Approach :

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \begin{array}{l} \mathcal{P}_{\text{succ}}(\mathcal{W}+C)\,(\mathcal{W}+C) \\ + \\ (1-\mathcal{P}_{\text{succ}}(\mathcal{W}+C))\,(\mathbb{E}(T_{lost}(\mathcal{W}+C)) + \mathbb{E}(T_{rec}) + \mathbb{E}(T(\mathcal{W}, C, R))) \end{array}$$

Computation of $\mathbb{E}(T(\mathcal{W}, C, R))$

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \mathbb{P}_{suc}(\mathcal{W} + C)(\mathcal{W} + C)$$

$$+(1-\mathbb{P}_{suc}(\mathcal{W}+C))\left[\mathbb{E}(T_{lost}(\mathcal{W} + C)) + E(T_{rec}) + \mathbb{E}(T(\mathcal{W}, C, R))\right]$$

With an exponential failure distribution, we have :

- $\mathbb{P}_{suc}(\mathcal{W} + C) = e^{-\lambda(\mathcal{W}+C)}$

Computation of $\mathbb{E}(T(\mathcal{W}, C, R))$

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \mathbb{P}_{suc}(\mathcal{W} + C)(\mathcal{W} + C)$$

$$+(1-\mathbb{P}_{suc}(\mathcal{W}+C))\,[\mathbb{E}(T_{lost}(\mathcal{W} + C)) + E(T_{rec}) + \mathbb{E}(T(\mathcal{W}, C, R))]$$

With an exponential failure distribution, we have :

- $\mathbb{P}_{suc}(\mathcal{W} + C) = e^{-\lambda(\mathcal{W}+C)}$
- $\mathbb{E}(T_{lost}(\mathcal{W} + C)) = \int_0^\infty x\mathbb{P}(X = x | X < \mathcal{W} + C)dx$
  $\mathbb{E}(T_{lost}(\mathcal{W} + C)) = \frac{1}{\lambda} - \frac{\mathcal{W}+C}{e^{\lambda(\mathcal{W}+C)}-1}$

Computation of $\mathbb{E}(T(\mathcal{W}, C, R))$

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \mathbb{P}_{suc}(\mathcal{W} + C)(\mathcal{W} + C)$$

$$+(1-\mathbb{P}_{suc}(\mathcal{W}+C))\left[\mathbb{E}(T_{lost}(\mathcal{W} + C)) + E(T_{rec}) + \mathbb{E}(T(\mathcal{W}, C, R))\right]$$

With an exponential failure distribution, we have :

- $\mathbb{P}_{suc}(\mathcal{W} + C) = e^{-\lambda(\mathcal{W}+C)}$
- $\mathbb{E}(T_{lost}(\mathcal{W} + C)) = \int_0^\infty x\mathbb{P}(X = x | X < \mathcal{W} + C)dx$
  $\mathbb{E}(T_{lost}(\mathcal{W} + C)) = \frac{1}{\lambda} - \frac{\mathcal{W}+C}{e^{\lambda(\mathcal{W}+C)}-1}$
- $\mathbb{E}(T_{rec}) = e^{-\lambda R}(D+R)+(1-e^{-\lambda R})(D+\mathbb{E}(T_{lost}(R))+\mathbb{E}(T_{rec}))$

Computation of $\mathbb{E}(T(\mathcal{W}, C, R))$

$$\mathbb{E}(T(\mathcal{W}, C, R)) = \mathbb{P}_{suc}(\mathcal{W} + C)(\mathcal{W} + C)$$

$$+(1-\mathbb{P}_{suc}(\mathcal{W}+C))\,[\mathbb{E}(T_{lost}(\mathcal{W} + C)) + E(T_{rec}) + \mathbb{E}(T(\mathcal{W}, C, R))]$$

With an exponential failure distribution, we have :

- $\mathbb{P}_{suc}(\mathcal{W} + C) = e^{-\lambda(\mathcal{W}+C)}$
- $\mathbb{E}(T_{lost}(\mathcal{W} + C)) = \int_0^{\infty} x\mathbb{P}(X = x|X < \mathcal{W} + C)dx$
  $\mathbb{E}(T_{lost}(\mathcal{W} + C)) = \frac{1}{\lambda} - \frac{\mathcal{W}+C}{e^{\lambda(\mathcal{W}+C)}-1}$
- $\mathbb{E}(T_{rec}) = e^{-\lambda R}(D+R)+(1-e^{-\lambda R})(D+\mathbb{E}(T_{lost}(R))+\mathbb{E}(T_{rec}))$

$$\mathbb{E}(T(\mathcal{W}, C, R)) = e^{\lambda R}\left(\frac{1}{\lambda} + D\right)\left(e^{\lambda(\mathcal{W}+C)} - 1\right)$$

Outline

1. Expected time needed to execute a work and to checkpoint it
2. Complexity of the general scheduling problem
3. Dynamic Programming algorithm for linear chains
4. Conclusion and extensions

Problem statement

The general scheduling problem is :

- Given a time bound K, can we find an ordering for the execution of several independent tasks, and decide after which tasks to checkpoint, so that the expected execution time does not exceed K?

Problem statement

The general scheduling problem is :

- Given a time bound K, can we find an ordering for the execution of several independent tasks, and decide after which tasks to checkpoint, so that the expected execution time does not exceed K?

Proposition

Consider n independent tasks, T_1, ..., T_n, with task T_i of duration $\mathcal{W}_i$ for $1 \leq i \leq n$. All checkpoint and recovery times are equal to C, and there is no downtime ($D = 0$). The problem to schedule these tasks, and to decide after which tasks to checkpoint, so as to minimize the expected execution time, is NP-complete in the strong sense.

Proof of NP-completeness

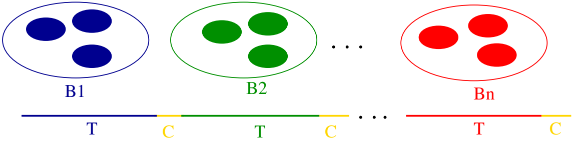
We use a reduction from 3-PARTITION, which is NP-complete in the strong sense.

Proof of NP-completeness

We use a reduction from 3-PARTITION, which is NP-complete in the strong sense.

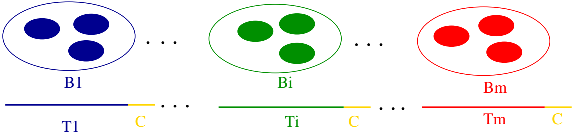
- General instance $\mathcal{I}_1$ of 3-PARTITION:
  - $3n$ integers $a_1, \ldots, a_{3n}$;
  - $\sum_{1 \leq j \leq 3n} a_j = nT$, and $\frac{T}{4} < a_j < \frac{T}{2}$ for $1 \leq j \leq 3n$,
  - Does there exist a partition in n subsets $B_1, \ldots, B_n$ of $\{a_1, \ldots, a_{3n}\}$ such that for all $1 \leq i \leq n$, $\sum_{a_j \in B_i} a_j = T$.

Note that necessarily in any solution, each B_i has cardinal 3.



Proof of NP-completeness

- Instance $\mathcal{I}_2$ of our problem :
  - $3n$ independent tasks: $task_1$, ..., $task_{3n}$, $task_i$ being of size $\mathcal{W}_i = a_i$.
  - Does there exist a partition in m subsets $B_1, \ldots, B_m$
  - $\sum_{i=1}^{m} T_i = nT$ and m checkpoints.
  - We let: $\lambda = \frac{1}{2T}$, $\quad C = R = \frac{1}{\lambda}(\ln(2) - \frac{1}{2})$, and $D = 0$, $\quad K = n\frac{e^{\lambda C}}{\lambda}(e^{\lambda(T+C)} - 1)$.



Outline

1. Expected time needed to execute a work and to checkpoint it
2. Complexity of the general scheduling problem
3. Dynamic Programming algorithm for linear chains
4. Conclusion and extensions

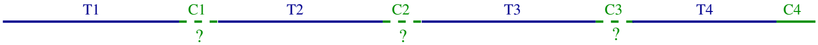
Problem statement

We want to compute the optimal expected execution time, that is:

- the expectation $\mathbb{E}$ of the time needed to process all the tasks of an applications whose DAG is a linear chain.

Problem:

- Decide whether to checkpoint or not after the completion of each given task.



Dynamic programming

T1	? C1	T2	? C2	T3	? C3	T4	C4
	0		0		0		1
	0		0		1		1
	0		1		0		1
	0		1		1		1
	1		0		0		1
	1		0		1		1
	1		1		0		1
	1		1		1		1

DPMAKESPAN(1,4)

Dynamic programming

Algorithm 1: *DPMAKESPAN*(x, n)

if $x = n$ **then**
  return $(\mathbb{E}(T(\mathcal{W}_n, C_n, R_{n-1})), n)$
$best \leftarrow \mathbb{E}(T(\sum_{i=x}^{n} \mathcal{W}_i, C_n, R_{x-1}))$
$numTask \leftarrow n$
for $j = x$ *to* $n - 1$ **do**
  $(exp\_succ, num\_Task) \leftarrow DPMAKESPAN(j + 1, n)$
  $Cur \leftarrow exp\_succ$
    $+ \mathbb{E}(T(\sum_{i=x}^{j} \mathcal{W}_i, C_j, R_{x-1}))$
  if $Cur < best$ **then**
    $best \leftarrow Cur$
    $numTask \leftarrow j$
return $(best, numTask)$

Linear complexity

Proposition

Algorithm1 provides the optimal solution for a linear chain of n tasks. Its complexity is $O(n^2)$.

Outline

1. Expected time needed to execute a work and to checkpoint it
2. Complexity of the general scheduling problem
3. Dynamic Programming algorithm for linear chains
4. Conclusion and extensions

Extensions

- General model of checkpointing costs:
  - The checkpoint after a task T_i may depend on T_i and on some other tasks that have been executed since the last checkpoint.
- Alleviating the *full parallelism* assumption:
  - Variable parallelism.
  - Ressource allocation problem.
- Using general failure laws than Exponential distributions:
  - First difficulty: Approximating the failure distribution of a platform of p processors.
  - Second difficulty: Estimating the expected execution time of a work W.

Conclusion

Important results:

- Closed-form formula for the expected execution time of a computational workflows followed by its checkpoint (using Exponential failure distribution).
- The strong NP-hardness of the problem for independent tasks and constant checkpoint costs.
- Dynamic programming algorithm for linear chains of tasks with arbitrary checkpoint costs.

Bibliography

M.-S. Bouguerra, T. Gautier, D. Trystram, and J.-M. Vincent.
A flexible checkpoint/restart model in distributed systems.
In *PPAM*, volume 6067 of *LNCS*, pages 206–215, 2010.

M.-S. Bouguerra, D. Trystram, and F. Wagner.
Complexity analysis of checkpoint scheduling with variable costs.
Computers, IEEE Transactions on, 2012.

J. T. Daly.
A higher order estimate of the optimum checkpoint interval for restart dumps.
Future Generation Computer Systems, 22(3):303–312, 2004.

J. W. Young.
A first order approximation to the optimum checkpoint interval.
Communications of the ACM, 17(9):530–531, 1974.