An optimal algorithm for scheduling checkpoints with variable costs

Mohamed Slim Bouguerra, Denis Trystram, Frédéric Wagner

An optimal algorithm for scheduling checkpoints with variable costs

Mohamed-Slim Bouguerra, Denis Trystram and Frédéric Wagner
Grenoble University and INRIA

1 Introduction

1.1 Context and Motivation

Since the last decade, computing systems turn to large scale parallel platforms composed of thousands of processors. Many actual applications run on such systems for long duration, up to several days or weeks. Recently, statistic studies about failures on high performance computing platforms emphasize that the mean time between failures may not exceed few hours [1]. Thus, it is necessary to develop efficient strategies providing a safe and reliable completion of applications. This may be achieved through redundancy [6] or by storing intermediate computation states on reliable external devices [3]. Saved states are then used to restart computations from the last checkpoint. This last approach called *checkpointing* is one of the most popular fault tolerance technique in parallel systems.

1.2 Brief review of related works

Young proposed a first order approximation to determine the optimal interval between checkpoints that minimizes the expected lost time before failure

[9]. This result was established considering that checkpoints are periodically scheduled and failures arrivals follow a Poisson's process. Daly [2] extended Young's result and proposed a higher order approximation under the same hypothesis. Ling et al. [8] introduced a new variational technique that gives a first order approximation of the checkpoint frequency for minimizing the expected lost time before a failure. This method is based on the first order truncation of the Taylor expansion of the failure distribution under the hypothesis that checkpoints do not alterate the probability of failure of the application. Recently, Dohi et al. described in [5] a numerical algorithm using the Brender's classical fixed-point theorem considering a distribution with increasing failure rate property. Hence, the proposed algorithm converges to a near optimal solution. Toueg et al. [7] provided a dynamic programming technique aiming at minimizing the expected completion time considering any failure law. Unfortunately, this model expresses only an upper bound for the expected completion time.

Notice that all these previous works except the last one consider that the checkpoint cost is constant. They provide different approximation solutions for determining the intervals between checkpoints under some restricted assumptions: infinite execution time, preemption, limited failure laws. Moreover, most of the proposed algorithms do not provide any guarantee on the time needed to reach a good solution.

1.3 Contributions

We consider a parallel system where users submit sequential jobs. Each node in this machine goes down after a random duration. Jobs are scheduled on nodes with a given predefined scheduling policy. Then, checkpointing is used to minimize the lost work due to the crash of nodes. Hence, a job is consid-

ered as saved if a checkpoint is successfully completed after its execution. However, a blind checkpoint can lead to an expensive overhead, while low frequency of checkpoints can lead to a huge amount of lost computation due to failures. We consider the expected wasted time as a suitable measure for dependability. We provide an optimal scheduling policy for checkpoints that minimizes this objective.

The checkpoint scheduling problem is usually formulated as a complex non-linear optimization based on a continuous objective function with unknown number of optimization variables. Unfortunately most of the time, such formulations lead to intractable problems. The main contribution of our work is to propose a new formulation for the checkpoint scheduling problem of pre-allocated jobs with arbitrary length. This formulation aims at minimizing the wasted time considering the variability on the checkpoint costs and the time to detect failures. To our knowledge such a generic formulation does not exist. We provide optimal algorithms for any failure laws and for arbitrary checkpoint costs.

2 Formulation of the problem

2.1 Model

We consider a computing platform composed of a collection of identical processors. After a failure, all the computation results since the last checkpoint are lost. Failures on different nodes are supposed to be identically distributed and without propagation. The random duration until failure is usually described by a general distribution failure law denoted by $F(t)$ and the corresponding density $f(t)$ with a finite mean. The failures are supposed to be *transient* (i.e. as soon as the failure is detected, the processor

can restart the computation) and their durations depend on the detection policy.

Sequential jobs are submitted to the different processors with a given pre-allocation policy. For a given processor, n jobs of length p_j are allocated in a predefined execution order. It is worth to notice that as jobs are independent, an optimal strategy for one processor leads to a global optimal strategy for whole computing platform. Without loss of generality, the jobs are indexed by their execution order.

We focus on the minimization of the wasted time. It is considered as the accumulation of three terms that are described below (see Figure 1). First, the *lost computation time* denoted by L is the lost time since the last checkpoint until the actual failure time. We introduce the re-execution ratio α as a speed-up ratio for the lost computations before the failure. This may be useful in some domain, for instance in transactional databases, the jobs correspond to transactions where the time needed to re-execute the lost transactions due to failure is usually less than the actual running time of the job ($0 < \alpha \leq 1$). Second, the *transitory failure time* denoted by T is the elapsed time from the actual failure time until the time of the failure detection. Upon a failure, a reparation or job migration is possible to fix the problem as soon as the failure is detected. We consider here that a detection mechanism is included into the checkpoints. More precisely, if the processor does not write its checkpoint on the stable storage at the checkpoint scheduled time, this processor is considered as failed. The proposed model can be used even when time to detect a failure is negligible by introducing a boolean parameter β ($\beta = 0$ when the failure detection is ignored). It is interesting to notice that to our knowledge there exists no related works taking into account the time needed to detect failures. Third, the *cumu-*

lative checkpoint overhead (denoted by σ) is the accumulation of the time spent in checkpoints phases before the failure. As a summary the expected wasted time denoted by W is expressed as: $W = \alpha L + \beta T + \sigma$.

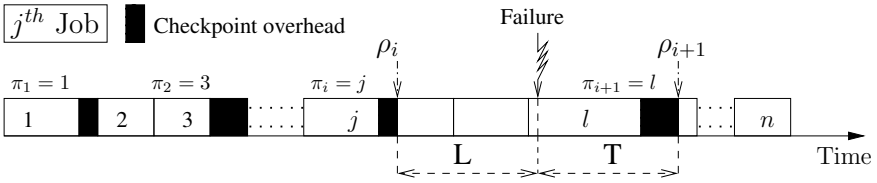

Figure 1: Typical execution on one processor

2.2 Formal description

Formally the problem inputs are a set of n pre-allocated jobs with different durations p_j and checkpoint costs c_j for storing the state just after the completion of job j. The output is a checkpoints schedule defined by a vector $\pi = (\pi_0, \pi_1, \cdots, \pi_k)$ of checkpointing times where π_i is the job index after which the i^{th} checkpoint is performed. For convenience we introduced an artificial initial checkpoint π_0 set to 0. Without loss of generality, we suppose that a checkpoint should be scheduled just after the last job. This assumption is usually considered as an acknowledgment for the application completion. Let τ_l denotes the useful computing time from the beginning until the l^{th} job such that $\tau_l = \sum_{j=1}^{l} p_j$ with $\tau_0 = 0$. Then, let σ_i denote the cumulative checkpoint overhead from the beginning until the i^{th} checkpoint. It is defined by $\sigma_i = \sum_{s=0}^{i} c_{\pi_s}$ ($\sigma_0 = 0$). We denote by $\rho_i = \tau_{\pi_i} + \sigma_i$ the absolute time from the beginning until the completion of the i^{th} checkpoint after the job π_i. Then, by conditioning on the time t when a failure happens between two consecutive checkpoints (i^{th} and $i{+}1^{th}$), such that $\rho_i < t \leq \rho_{i+1}$ the expected wasted time can be written as follows. Let Δ be the conditional

wasted time if a failure happens between two consecutive checkpoints:

$$\Delta(\pi_i, \pi_{i+1}) = \int_{\rho_i}^{\rho_{i+1}} [\sigma_i + \alpha(t - \rho_i) + \beta(\rho_{i+1} - t)] f(t) dt, \;\; 0 \leq i < n. \quad (1)$$

Let W denote the total expected wasted time considering a given checkpoint scheduling policy. Then, the general expression of the wasted time is given in the following expression.

$$W(\pi) = \sum_{i=0}^{k-1} \Delta(\pi_i, \pi_{i+1}). \quad (2)$$

3 Algorithm Description

We present in this section an algorithm for scheduling the checkpoints with variable costs for any distribution of failures and prove its optimality. This algorithm is based on dynamic programming. It is pseudo-polynomial in the general case and fully polynomial when checkpoint costs are constant.

Variable checkpoint costs: Let Π_l^θ denote the set of valid checkpoints schedules for the l first jobs ($l \leq n$) being given integer θ that verify the following constraint.

$$\pi = (\pi_0, \pi_1, \cdots, \pi_k) \in \Pi_l^\theta \qquad \text{if and only if} \qquad \sum_{i=0}^{k} c_{\pi_i} = \theta \text{ and } \pi_k = l.$$

This means that the cumulative checkpoint overhead is exactly θ ($\sigma_k = \theta$). and the last checkpoint is scheduled after the l^{th} job. Thus, there exists at least one valid optimal schedule $\pi \in \Pi_l^\theta$, such that $W_l^{\theta *} = \min_{\pi \in \Pi_l^\theta} W(\pi)$, where $W_l^{\theta *}$ denotes the minimal reachable wasted time for jobs up to l under the previous constraint.

Given a schedule $\pi \in \Pi_l^\theta$, we recall that a checkpoint is scheduled just after the last job ($\pi_k = l$). We denote by r the job index of the penultimate checkpoint ($\pi_{k-1} = r$). Let $\pi' = (\pi_0, \pi_1, \cdots, \pi_{k-1})$. From Expression 2 the expected wasted time for the first l jobs can be expressed as $W(\pi) = W(\pi') + \Delta(r, l)$. This relation may be interpreted in the following way: the scheduling solution for k checkpoints considering only the l first jobs with cumulative checkpoint overhead $\sigma_k = \theta$ can be obtained from a scheduling solution for $k-1$ checkpoints with cumulative checkpoint overhead $\sigma_{k-1} = \theta - c_l$ for jobs up to r. Based on this observation we are able to establish the following proposition.

Proposition 1. *$\pi \in \Pi_l^\theta$ and $W(\pi) = W_l^{\theta*}$ implies that $\pi' \in \Pi_r^{\theta - c_l}$ and $W(\pi') = W_r^{\theta - c_l *}$.*

Proof. Let $\pi \in \Pi_l^\theta$, we have $\sum_{i=0}^{k-1} c_{\pi_i} = \theta - c_l$ and $\pi_{k-1} = r$, thus $\pi' \in \Pi_r^{\theta - c_l}$.
Suppose now that $W(\pi') > W_r^{\theta - c_l *}$.
Thus, it exists a valid schedule $\pi'' \in \Pi_r^{\theta - c_l}$ such that $W(\pi'') < W(\pi')$.
Therefore, $W(\pi'') + \Delta(r, l) < W(\pi) \Leftrightarrow W(\pi'') + \Delta(r, l) < W_l^{\theta*}$, which leads to the a contradiction, thus we have $W(\pi') \leq W_r^{\theta - c_l *}$.
As $W_r^{\theta - c_l *}$ is the optimal, we obtain $W(\pi') = W_r^{\theta - c_l *}$. □

Let W^* denote the global minimal wasted time for the general problem without constraints. $W^* = \min_{\theta \in \Theta} W_n^{\theta*}$ denotes the minimum taken over all possible integer values of θ between $\min_j(c_j)$ and $n \max_j(c_j)$ (this range is denoted by Θ). Using a recursion on the last checkpoint, we can directly derive from Proposition 1 that:

$$W_l^{\theta*} = \min_{1 \leq r < l} \left(W_r^{\theta - c_l *} + \Delta(r, l) \right), \qquad 1 \leq l \leq n,\ \theta \in \Theta. \tag{3}$$

This equation is a dynamic programming scheme. This algorithm leads to

an optimal schedule. It is pseudo-polynomial time in $O(n^3 \times \max_j(c_j))$.

Constant checkpoint costs: Using a similar analysis, we show that when checkpoint costs are constant, the previous algorithm can be simplified to an optimal polynomial algorithm. It is clear that when checkpoints are constant the cumulative checkpoint overhead is a multiple of k ($\sigma_k = kc$) where c is the checkpoint cost. Thus, given a checkpoint scheduling policy, the cumulative checkpoint overhead can be only characterized by the number of checkpoints k. Hence, in this case the range Θ of k is reduced to integer interval $[1..n]$. Using Proposition 1, the optimal scheduling is obtained after computing all the $W_l^{k^*}$ for $1 \leq k \leq n$ and $1 \leq l \leq n$, using the following recursive equation.

$$W_l^{k^*} = \min_{k-1 \leq r < l} \left(W_r^{k-1^*} + \Delta(r,l) \right), \qquad 1 \leq l \leq n,\ 1 \leq k \leq n. \tag{4}$$

This algorithm is polynomial, its complexity of this algorithm is in $O(n^3)$.

4 Concluding remarks

We have presented in this paper a new combinatorial approach for scheduling checkpoints of a set of independent jobs in parallel platforms. We have proposed an optimal algorithmic solution for the general problem (arbitrary failure laws and variable checkpoint costs) based on dynamic programming. We believe that this work could serve as a good starting point for many further studies. For instance, it is clear that for non-preallocated jobs, checkpoint scheduling and job scheduling are strongly interleaved. Thus, the dynamic programming scheme can be used as a basic step for designing a mixed solution. Another extension of this work is to include advanced

failure detection mechanisms into the proposed scheme.

References

[1] F. Cappello. Fault Tolerance in Petascale/Exascale Systems: Current Knowledge, Challenges and Research Opportunities. *Int. J. High Perform. Comput. Appl.*, 23(3):212–226, 2009.

[2] J. T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Gener. Comput. Syst.*, 22(3):303–312, 2006.

[3] E. N. Elnozahy and J. S. Plank. Checkpointing for peta-scale systems: A look into the future of practical rollback-recovery. *IEEE Trans. Dependable Secur. Comput.*, 1(2):97–108, 2004.

[4] S. Garg, Y. Huang, C. Kintala, and K.S. Trivedi. Minimizing completion time of a program by checkpointing and rejuvenation. In *Proceedings of the ACM SIGMETRICS*, pages 252–261, 1996.

[5] T. Ozaki, T. Dohi, and N. Kaio. Numerical computation algorithms for sequential checkpoint placement. *Perform. Eval.*, 66(6):311–326, 2009.

[6] E. Saule and D. Trystram. Analyzing scheduling with transient failures. *Information Processing Letters*, 109(11):539–542, 2009.

[7] S. Toueg and Ö. Babaoğlu. On the optimum checkpoint selection problem. *SIAM J. Comput.*, 13(3):630–649, 1984.

[8] X.Lin Y.Ling, J.Mi. A variational calculus approach to optimal checkpoint placement. *IEEE Trans. Comput.*, 50(07):699–708, 2001.

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