Towards Optimal Multi-Level Checkpointing

Anne Benoit, Aurélien Cavelan, Valentin Le Fèvre, Yves Robert, Hongyang Sun

Towards Optimal Multi-Level Checkpointing

Anne Benoit, Aurélien Cavelan, Valentin Le Fèvre, Yves Robert, Hongyang Sun

Abstract—We provide a framework to analyze multi-level checkpointing protocols, by formally defining a k-level checkpointing pattern. We provide a first-order approximation to the optimal checkpointing period, and show that the corresponding overhead is in the order of $\sum_{\ell=1}^{k} \sqrt{2\lambda_\ell C_\ell}$, where λ_ℓ is the error rate at level ℓ, and C_ℓ the checkpointing cost at level ℓ. This nicely extends the classical Young/Daly formula on single-level checkpointing. Furthermore, we are able to fully characterize the shape of the optimal pattern (number and positions of checkpoints), and we provide a dynamic programming algorithm to determine the optimal subset of levels to be used. Finally, we perform simulations to check the accuracy of the theoretical study and to confirm the optimality of the subset of levels returned by the dynamic programming algorithm. The results nicely corroborate the theoretical study, and demonstrate the usefulness of multi-level checkpointing with the optimal subset of levels.

Index Terms—resilience, fail-stop errors, multi-level checkpointing, optimal pattern.

1 Introduction

Checkpointing is the de-facto standard resilience method for HPC platforms at extreme-scale. However, the traditional single-level checkpointing method suffers from significant overhead, and multi-level checkpointing protocols now represent the state-of-the-art technique. These protocols allow different levels of checkpoints to be set, each with a different checkpointing overhead and recovery ability. Typically, each level corresponds to a specific fault[1] type, and is associated to a storage device that is resilient to that type. For instance, a two-level system would deal with (i) transient memory errors (level 1) by storing key data in main memory; and (ii) node failures (level 2) by storing key data in stable storage (remote redundant disks).

We consider a very general scenario, where the platform is subject to k levels of faults, numbered from 1 to k. Level ℓ is associated with an error rate λ_ℓ, a checkpointing cost C_ℓ, and a recovery cost R_ℓ. A fault at level ℓ destroys all the checkpoints of lower levels (from 1 to $\ell-1$ included) and implies a roll-back to a checkpoint of level ℓ or higher. Similarly, a recovery of level ℓ will restore data from all lower levels. Typically, fault rates are decreasing and checkpoint/recovery costs are increasing when we go to higher levels: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k$, $C_1 \leq C_2 \leq \cdots \leq C_k$, and $R_1 \leq R_2 \leq \cdots \leq R_k$.

The idea of multi-level checkpointing is that checkpoints are taken for each level of faults, but at different periods. Intuitively, the less frequent the faults, the longer the checkpointing period: this is because the risk of a failure striking is lower when going to higher levels; hence the expected re-execution time is lower too; one can safely checkpoint less frequently, thereby reducing failure-free overhead (checkpointing is useless in the absence of fault). There are several natural approaches to implement multi-level checkpointing. The first option is to use *independent checkpointing periods* for each level. This option raises several difficulties, the most prominent one being overlapping checkpoints. Typically, we need to checkpoint different levels in sequence (e.g., writing into memory before writing onto disk), so we would need to delay some checkpoints, which might not be possible in some environments, and which would introduce irregular periods. The second option is to synchronize all checkpoint levels by nesting them inside a *periodic pattern* that repeats over time, as illustrated in Figure 1(a). In this figure, the **pattern** has five computational **segments**, each followed by a level-1 checkpoint. A segment is a chunk of work between two checkpoints, and a pattern consists in segments and checkpoints. The second and fifth level-1 checkpoints are followed by a level-2 checkpoint. Finally, the pattern ends with a level-3 checkpoint. When using patterns, a checkpoint at level ℓ is always preceded by checkpoints at all lower levels 1 to $\ell-1$, which makes good sense in practice (e.g., with two levels, main memory and disk, one writes the data into memory before transferring it to disk).

Using periodic patterns simplifies the orchestration of checkpoints at all levels. In addition, repeatedly applying the same pattern is optimal for on-line scheduling problems, or for jobs running a very long (even infinite) time on the platform. Indeed, in this scenario, we seek the best pattern, i.e., the one whose overhead is minimal. The *overhead* of a pattern is the price per work unit to pay for resilience in the pattern; hence minimizing overhead is equivalent to optimizing platform throughput. For a pattern $\mathrm{P}(W)$ with W units of work (the cumulated length of all its segments), the overhead $H(\mathrm{P(W)})$ is defined as the ratio of the pattern's expected execution time $\mathbb{E}(\mathrm{P(W)})$ over its total work W minus 1:

$$H(\mathrm{P}(W)) = \frac{\mathbb{E}(\mathrm{P}(W))}{W} - 1. \quad (1)$$

If there were neither checkpoint nor fault, the overhead would be zero. Determining the optimal pattern (with minimal overhead), and then repeatedly using it until job completion, is the optimal approach with Exponential failure distributions and long-lasting jobs. Indeed, once a pattern is successfully executed, the optimal strategy is to re-execute the same pattern. This is because of the memoryless property of exponential distributions: the history of failures has no impact on the solution, so if a pattern is optimal at some point in time, it stays optimal

• *Anne Benoit, Aurélien Cavelan, Valentin Le Fèvre, Yves Robert, and Hongyang Sun are with Ecole Normale Supérieure de Lyon & INRIA, France. Yves Robert is also with University of Tennessee Knoxville, USA. Contact: Anne.Benoit@ens-lyon.fr.*

1. We use the terms *fault*, *failure* and *error* indifferently.

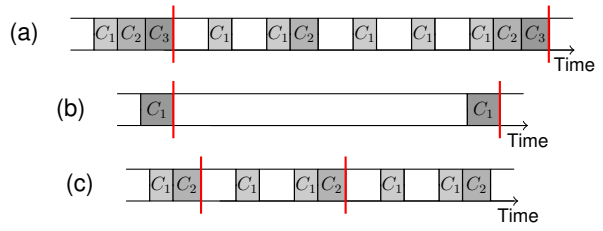


Figure 1: Checkpointing patterns (highlighted using red bars) with (a) $k = 3$, (b) $k = 1$, and (c) $k = 2$ levels.

later in the execution, because we have no further information about the amount of work still to be executed.

The difficulty of characterizing the optimal pattern dramatically increases with the number of levels. How many checkpoints of each level should be used, and at which locations inside the pattern? What is the optimal length of each segment? With one single level (see Figure 1(b)), there is a single segment of length W, and the Young/Daly formula [18], [6] gives $W^{\text{opt}} = \sqrt{\frac{2C_1}{\lambda_1}}$. The minimal overhead is then $H^{\text{opt}} = \sqrt{2\lambda_1 C_1} + O(\lambda_1)$ [3].

With two levels, the pattern still has a simple shape, with N segments followed by a level-1 checkpoints, and ended by a level-2 checkpoint (see Figure 1(c)). Recent work [8] shows that all segments have same length in the optimal pattern, and provides mathematical equations that can be solved numerically to compute both the optimal length W^{opt} of the pattern and its optimal number of segments. However, no closed-form expression is available, neither for W^{opt}, nor for the minimal overhead H^{opt}.

With three levels, no optimal solution is known. The pattern shape becomes quite complicated. Coming back to Figure 1(a), we identify two sub-patterns ending with a level-2 checkpoint. The first sub-pattern has 2 segments while the second one has 3. The memoryless property does not imply that all sub-patterns are identical, because the state after completing the first sub-pattern is not the same as the initial state when beginning the execution of the pattern. In the general case with k levels, the shape of the pattern will be even more complicated, with different-shaped sub-patterns (each ended by a level $k-1$ checkpoint). In turn, each sub-pattern may have different-shaped sub-sub-patterns (each ended by a level $k-2$ checkpoint), and so on. The major contribution of this work is to provide an analytical characterization of the optimal pattern with an arbitrary number k of checkpointing levels, with closed-form formulas for the pattern length W^{opt}, the number of checkpoints at each level, and the optimal overhead H^{opt}. In particular, we obtain the following beautiful result:

$$H^{\text{opt}} = \sum_{\ell=1}^{k} \sqrt{2\lambda_\ell C_\ell} + O(\Lambda), \tag{2}$$

where $\Lambda = \sum_{\ell=1}^{k} \lambda_\ell$. However, we point out that this analytical characterization relies on a first-order approximation, so it is valid only when resilience parameters C_ℓ and R_ℓ are small in front of the platform Mean Time Between Failures (MTBF) $\mu = 1/\Lambda$. Also, the optimal pattern has rational number of segments, and we use rounding to derive a practical solution. Still, Equation (2) provides a lower bound on the optimal overhead, and this bound is met very closely in all our experimental scenarios.

Finally, in many practical cases, there is no obligation to use all available checkpointing levels. For instance, with $k = 3$ levels, one may choose among four possibilities: level 3 only, levels 1 and 3, levels 2 and 3, and all levels 1, 2 and 3. Of course, we still have to account for all failure types, which translates into the following:

- level 3: use $\lambda_3 \leftarrow \lambda_1 + \lambda_2 + \lambda_3$;
- levels 1 and 3: use λ_1 and $\lambda_3 \leftarrow \lambda_2 + \lambda_3$;
- levels 2 and 3: use $\lambda_2 \leftarrow \lambda_1 + \lambda_2$ and λ_3;
- all levels: use λ_1, λ_2 and λ_3.

Our analytical characterization of the optimal pattern leads to a simple dynamic programming algorithm for selecting the optimal subset of levels.

The rest of this paper is organized as follows. Section 2 surveys the related work. Section 3 is the heart of the paper and shows how to compute the optimal pattern as well as the optimal subset of levels. Section 4 is devoted to simulations assessing the accuracy of the first-order approximation. Finally, Section 5 provides concluding remarks and hints for future work.

2 Related work

Given the checkpointing cost and platform MTBF, classical formulas due to Young [18] and Daly [6] are well known to determine the optimal checkpointing period in the single-level checkpointing scheme. However, this method suffers from the intrinsic limitation that the cost of checkpointing/recovery grows with failure probability, and becomes unsustainable at large scale [9], [4] (even with diskless or incremental checkpointing [15]).

To reduce the I/O overhead, various two-level checkpointing protocols have been studied. Vaidya [17] proposed a two-level recovery scheme that tolerates a single node failure using a local checkpoint stored on a parter node. If more than one failure occurs during any local checkpointing interval, the scheme resorts to the global checkpoint. Silva and Silva [16] advocated a similar scheme by using memory to store local checkpoints, which is protected by XOR encoding. Di et al. [8] analyzed a two-level checkpointing pattern, and proved equal-length segments in the optimal solution. They also provided mathematical equations that can be solved numerically to compute the optimal pattern length and number of segments. Benoit et al. [3] relied on disk checkpoints to cope with fail-stop failures and memory checkpoints coupled with error detectors to handle silent data corruptions. They derived first-order approximation formulas for the optimal pattern length and the number of memory checkpoints between two disk checkpoints.

Some authors have also generalized two-level checkpointing to account for an arbitrary number of levels. Moody et al. [14] implemented this approach in a three-level Scalable Checkpoint/Restart (SCR) library. They relied on a rather complex Markov model to recursively compute the efficiency of the scheme. Bautista-Gomez et al. [2] designed a four-level checkpointing library, called Fault Tolerance Interface (FTI), in which partner-copy and Reed-Solomon coding are employed as two intermediate levels between local and global disks. Based on FTI, Di et al. [7] proposed an iterative method to compute the optimal checkpointing interval for each level with prior knowledge of the application's total execution time. Hakkarinen and Chen [11] considered multi-level diskless checkpointing for

tolerating simultaneous failures of multiple processors. Balaprakash et al. [1] studied the trade-off between performance and energy for general multi-level checkpointing schemes.

While all of these works relied on numerical methods to compute the checkpointing intervals at different levels, this paper is the first one to provide explicit formulas on the optimal parameters in a multi-level checkpointing protocol (up to first-order approximation as in Young/Daly's classical result).

3 Computing the Optimal Pattern

This section computes the optimal multi-level checkpointing pattern. We first state our assumptions in Section 3.1, and then analyze the simple case with $k = 2$ levels in Section 3.2, before proceeding to the general case in Section 3.3. Finally, the algorithm to compute the optimal subset of levels is described in Section 3.4.

3.1 Assumptions

In this paper, we assume that failures from different levels are independent[2]. For each level ℓ, the arrival of failures follows *Poisson* process with error rate λ_ℓ. In order to deal with the interplay of failures from different levels, we make use of the following well-known properties of independent Poisson processes [10, Chapter 2.3].

Property 1. *During the execution of a segment with length w, let X_ℓ denote the time when the first level-ℓ error strikes. Thus, X_ℓ is a random variable following an Exponential distribution with parameter λ_ℓ, for all $\ell = 1, 2, \ldots, k$.*

(1). Let X denote the time when the first error (of any level) strikes. We have $X = \min\{X_1, X_2, \ldots, X_k\}$, which follows an Exponential distribution with parameter $\Lambda = \sum_{\ell=1}^{k} \lambda_\ell$. The probability of having an error (from any level) in the segment is therefore $P(X \leq w) = 1 - e^{-\Lambda w}$.

(2). Given that an error (from any level) strikes during the execution of the segment, the probability that the error belongs to a particular level is proportional to the error rate of that level, i.e., $P(X = X_\ell | X \leq w) = \frac{\lambda_\ell}{\Lambda}$, for all $\ell = 1, 2, \ldots, k$.

Moreover, we assume that error rates of different levels are of the same order, i.e., $\lambda_\ell = \Theta(\Lambda)$ for all $\ell = 1, 2, \ldots, k$, and that errors only strike during the computations, while checkpointing and recovery are error-free. Indeed, the durations of checkpoints and recoveries are generally small compared to the pattern length, so the probability of a failure striking during these operations is low. It has been shown in [3] that removing this assumption does not impact the first-order approximation of the pattern overhead.

3.2 Optimal two-level pattern

We start by analyzing the two-level pattern shown in Figure 1(b). The goal is to determine a first-order approximation to the optimal pattern length W, the number n of level-1 checkpoints in the pattern, as well as the length $w_i = \alpha_i W$ of the i-th segment, for all $1 \leq i \leq n$, where $\sum_{i=1}^{n} \alpha_i = 1$.

2. In practice, failures from different checkpointing levels can exhibit potential correlation [12], [7]. Consideration of correlated failures is beyond the scope of this paper.

3.2.1 With a single segment

We first consider a special case of the two-level pattern, in which only a single segment is present, i.e., $n = 1$. The result establishes the order of the optimal pattern length W^{opt}, which will be used later for analyzing the general case. Recall that $\Lambda = \lambda_1 + \lambda_2$ and, for convenience, let us also define $C = C_1 + C_2$. The following proposition shows the expected time of such a pattern with fixed length W.

Proposition 1. *The expected execution time of a two-level pattern with a single segment and fixed length W is*

$$\mathbb{E} = W + C + \frac{1}{2}\Lambda W^2 + O(\max\{\Lambda^2 W^3, \Lambda W\}).$$

Proof. We can express the expected execution time of the pattern recursively as follows:

$$\mathbb{E} = P\left(\mathbb{E}^{\text{lost}}(W, \Lambda) + \frac{\lambda_1}{\Lambda}(R_1 + \mathbb{E}) + \frac{\lambda_2}{\Lambda}\Big(R_2 + R_1 + \mathbb{E}\Big)\right) + (1 - P)(W + C), \tag{3}$$

where $P = 1 - e^{-\Lambda W}$ denotes the probability of having a failure (either level-1 or level-2) during the execution of the pattern based on Property 1.1, and $\mathbb{E}^{\text{lost}}(w_i, \Lambda)$ denotes the expected time lost when such a failure occurs. In this case, and based on Property 1.2, if the failure belongs to level 1, which happens with probability $\frac{\lambda_1}{\Lambda}$, we can recover from the latest level-1 checkpoint (R_1). Otherwise, the failure belongs to level 2 with probability $\frac{\lambda_2}{\Lambda}$, and we need to first recover from the latest level-2 checkpoint (R_2) before restoring the level-1 checkpoint (R_1). In both cases, the entire pattern needs to be re-executed again. Finally, if no error (of any level) strikes, which happens with probability $1 - P$, the pattern is completed after W time of execution followed by the time C to perform the two checkpoints, which are assumed to be error-free.

From [13, Equation (1.13)], the expected time lost when executing a segment of length W with error rate Λ is

$$\mathbb{E}^{\text{lost}}(W, \Lambda) = \frac{1}{\Lambda} - \frac{W}{e^{\Lambda W} - 1}. \tag{4}$$

Substituting Equation (4) into Equation (3) and solving for $\mathbb{E}$, we get:

$$\mathbb{E} = \left(e^{\Lambda W} - 1\right)\left(\frac{1}{\Lambda} + R_1 + \frac{\lambda_2}{\Lambda} R_2\right) + C_1 + C_2, \tag{5}$$

which is an exact formula on the expected execution time of the pattern. Now, using Taylor series to expand $e^{\Lambda W} = 1 + \Lambda W + \frac{\Lambda^2 W^2}{2} + O(\Lambda^3 W^3)$ while assuming $W = \Theta(\Lambda^{-x})$, where $0 < x < 1$, we can re-write Equation (5) as

$$\mathbb{E} = W + \frac{1}{2}\Lambda W^2 + C_1 + C_2 + O(\Lambda^2 W^3) + \left(\Lambda W + \frac{\Lambda^2 W^2}{2} + O(\Lambda^3 W^3)\right)\left(R_1 + \frac{\lambda_2}{\Lambda} R_2\right).$$

Since recovery costs (R_1, R_2) are assumed to be constants, and error rates $(\lambda_1, \lambda_2, \Lambda)$ are in the same order, the expected execution time can be expressed as follows:

$$\mathbb{E} = W + C_1 + C_2 + \frac{1}{2}\Lambda W^2 + O(\Lambda^2 W^3) + O(\Lambda W),$$

which completes the proof of the proposition. □

From Proposition 1, the expected execution overhead of the

pattern can be derived as

$$H = \frac{C}{W} + \frac{1}{2}\Lambda W + O(\max\{\Lambda^2 W^2, \Lambda\}).$$

Assume that the platform MTBF $\mu = 1/\Lambda$ is large in front of the resilience parameters, and consider the first two terms of H: the overhead is minimized when the pattern has length $W = \Theta(\Lambda^{-1/2})$, and in that case both terms are in the order of $\Theta(\Lambda^{1/2})$, so we have $H = \Theta(\Lambda^{1/2}) + O(\Lambda)$. Indeed, the last term $O(\Lambda^2 W^2) = O(\Lambda)$ becomes negligible compared to $\Theta(\Lambda^{1/2})$. Hence, the optimal pattern length W^{opt} can be obtained by balancing the first two terms in H, which gives

$$W^{\text{opt}} = \sqrt{\frac{2C}{\Lambda}} = \Theta(\Lambda^{-1/2}), \tag{6}$$

and the optimal execution overhead becomes

$$H^{\text{opt}} = \sqrt{2\Lambda C} + O(\Lambda). \tag{7}$$

Remarks. Unlike in single-level checkpointing, the checkpoint to roll back to in a two-level pattern depends on which type of error strikes first. Under first-order approximation and assuming that the resilience parameters are small compared to the platform MTBF and pattern length, the formulas shown in Equations (6) and (7) reduce exactly to Young/Daly's classical result by aggregating the error rates and checkpointing costs of both levels.

3.2.2 With multiple segments

We now consider the general two-level pattern with multiple segments, and derive the optimal pattern parameters. As in the single-segment case, we start with a proposition showing the expected time to execute a two-level pattern with fixed parameters.

Proposition 2. *The expected execution time of a given two-level pattern is*

$$\mathbb{E} = W + nC_1 + C_2 + \frac{1}{2}\left(\lambda_1 \sum_{i=1}^{n} \alpha_i^2 + \lambda_2\right) W^2 + O(\Lambda^{1/2}).$$

Proof. We first prove the following result (by induction) on the expected time $\mathbb{E}_i$ to execute the i-th segment of the pattern (up to the level-1 checkpoint at the end of the segment):

$$\mathbb{E}_i = w_i + C_1 + \frac{\lambda_1}{2} w_i^2 + \lambda_2 \left(\frac{w_i^2}{2} + \sum_{j=1}^{i-1} w_j w_i\right) + O(\Lambda^{1/2}). \tag{8}$$

According to the result with a single segment, we know that the optimal pattern length and hence the segment length are in the order of $O(\Lambda^{-1/2})$, which implies that $\mathbb{E}_i = w_i + O(1)$.

For the ease of analysis, we assume that there is a hypothetical segment at the beginning of the pattern with length $w_0 = 0$ (hence no need to checkpoint). For this segment, we have $\mathbb{E}_0 = w_0 = 0$, satisfying Equation (8). Suppose the claim holds up to $\mathbb{E}_{i-1}$. Then, $\mathbb{E}_i$ can be recursively expressed as follows:

$$\begin{aligned}\mathbb{E}_i = P_i \Bigg(& \mathbb{E}^{\text{lost}}(w_i, \Lambda) + \frac{\lambda_1}{\Lambda}(R_1 + \mathbb{E}_i) \\ & + \frac{\lambda_2}{\Lambda}\Big(R_2 + R_1 + \sum_{j=1}^{i-1} \mathbb{E}_j + \mathbb{E}_i\Big)\Bigg) \\ & + (1 - P_i)(w_i + C_1),\end{aligned} \tag{9}$$

where $P_i = 1 - e^{-\Lambda w_i}$ denotes the probability of having a failure (either level-1 or level-2) during the execution of the segment, and $\mathbb{E}^{\text{lost}}(w_i, \Lambda)$ denotes the expected time lost when such a failure occurs.

Equation (9) is very similar to Equation (3), except when a level-2 failure occurs we need to re-execute all the segments (up to segment i) that have been executed so far. Following the derivation of Proposition 1 and applying $\mathbb{E}_j = w_j + O(1)$ for $j = 1, 2, \ldots, i-1$, we can derive the first-order approximation of $\mathbb{E}_i$ as follows:

$$\begin{aligned}\mathbb{E}_i &= w_i + C_1 + \frac{1}{2}\left(\lambda_1 w_i^2 + \lambda_2 w_i^2 + 2\lambda_2 w_i \sum_{j=1}^{i-1} \mathbb{E}_j\right) + O(\Lambda^{1/2}) \\ &= w_i + C_1 + \frac{1}{2}\left(\lambda_1 w_i^2 + \lambda_2 w_i^2 + 2\lambda_2 w_i \sum_{j=1}^{i-1}\Big(w_j + O(1)\Big)\right) + O(\Lambda^{1/2}) \\ &= w_i + C_1 + \frac{1}{2}\left(\lambda_1 w_i^2 + \lambda_2\Big(w_i^2 + 2\sum_{j=1}^{i-1} w_j w_i\Big)\right) + O(\Lambda^{1/2}).\end{aligned} \tag{10}$$

Since the level-2 checkpoint at the end of the pattern is also assumed to be error-free, we can compute the expected execution time of the pattern as

$$\begin{aligned}\mathbb{E} &= \sum_{i=1}^{n} \mathbb{E}_i + C_2 \\ &= W + nC_1 + C_2 + \frac{1}{2}\left(\lambda_1 \sum_{i=1}^{n} \alpha_i^2 + \lambda_2\right) W^2 + O(\Lambda^{1/2}),\end{aligned}$$

since $\sum_{i=1}^{n} w_i^2 + 2\sum_{i=1}^{n}\sum_{j=1}^{i-1} w_j w_i = \left(\sum_{i=1}^{n} w_i\right)^2 = W^2$. □

Theorem 1. *A first-order approximation to the optimal two-level pattern is characterized by*

$$n^{opt} = \sqrt{\frac{\lambda_1}{\lambda_2} \cdot \frac{C_2}{C_1}}, \tag{11}$$

$$\alpha_i^{opt} = \frac{1}{n^{opt}} \quad \forall i = 1, 2, \ldots, n^{opt}, \tag{12}$$

$$W^{opt} = \sqrt{\frac{n^{opt} C_1 + C_2}{\frac{1}{2}\left(\frac{\lambda_1}{n^{opt}} + \lambda_2\right)}}, \tag{13}$$

where n^{opt} is the number of segments, $\alpha_i^{opt} W^{opt}$ is the length of the i-th segment, and W^{opt} is the pattern length.
The optimal pattern overhead is

$$H^{opt} = \sqrt{2\lambda_1 C_1} + \sqrt{2\lambda_2 C_2} + O(\Lambda). \tag{14}$$

Proof. For a given pattern with a fixed number n of segments, $\sum_{i=1}^{n} \alpha_i^2$ is minimized subject to $\sum_{i=1}^{n} \alpha_i = 1$ when $\alpha_i = \frac{1}{n}$ for all $i = 1, 2, \ldots, n$. Hence, we can derive the expected execution overhead from Proposition 2 as follows:

$$H = \frac{nC_1 + C_2}{W} + \frac{1}{2}\left(\frac{\lambda_1}{n} + \lambda_2\right) W + O(\Lambda). \tag{15}$$

For a given n, the optimal work length can then be computed from Equation (15), and it is given by $W^{\text{opt}} = \sqrt{\frac{nC_1 + C_2}{\frac{1}{2}\left(\frac{\lambda_1}{n} + \lambda_2\right)}}$. In that case, the execution overhead becomes

$$H = \sqrt{2\left(\frac{\lambda_1}{n} + \lambda_2\right)(nC_1 + C_2)} + O(\Lambda), \tag{16}$$

which is minimized as shown in Equation (14) when n satisfies Equation (11). Indeed, $2\left(\frac{\lambda_1}{n^{\text{opt}}}+\lambda_2\right)(n^{\text{opt}}C_1+C_2) = 2\lambda_1 C_1 + 2\lambda_2 C_2 + 4\sqrt{\lambda_1\lambda_2 C_1 C_2} = (\sqrt{2\lambda_1 C_1}+\sqrt{2\lambda_1 C_1})^2$. In practice, since the number of segments can only be a positive integer, the optimal solution is either $\max(1, \lfloor n^{\text{opt}} \rfloor)$ or $\lceil n^{\text{opt}} \rceil$, whichever leads to a smaller value of the convex function H as shown in Equation (16). □

Remarks. Consider the example given in [8] with $C_1 = R_1 = 20$, $C_2 = R_2 = 50$, $\lambda_1 = 2.78 \times 10^{-4}$ and $\lambda_2 = 4.63 \times 10^{-5}$. The optimal solution[3] provided by [8] gives $n^{\text{opt}} = 3.83$, $W^{\text{opt}} = 1362.49$ and $H^{\text{opt}} = 0.1879$, while Theorem 1 suggests $n^{\text{opt}} = 3.87$, $W^{\text{opt}} = 1378.27$ and $H^{\text{opt}} = 0.1735$, which is quite close to the exact optimum. The difference in overhead is due to the negligence of lower-order terms in the first-order approximation. We point out that the solution provided by [8] relies on numerical methods to solve rather complex mathematical equations, whose convergence is not always guaranteed, and it is only applicable to two levels. Our result, on the other hand, is able to provide fast and good approximation to the optimal solution when the error rates are sufficiently small, and it can be readily extended to an arbitrary number of levels, as shown in the next section.

3.3 Optimal k-level pattern

In this section, we derive the first-order approximation to the optimal k-level pattern by determining its length W, the number N_ℓ of level-ℓ checkpoints for all $1 \le \ell \le k$, as well as the positions of all checkpoints in the pattern.

3.3.1 Observations

Before analyzing the optimal pattern, we make several observations. First, we can obtain the orders of the optimal length and pattern overhead as shown below (recall that $\Lambda = \sum_{\ell=1}^{k} \lambda_\ell$).

Observation 1. *Consider the simplest k-level pattern with a single segment of length W. We can conduct the same analysis as in Section 3.2.1 to show that the optimal pattern length satisfies $W^{opt} = \Theta(\Lambda^{-1/2})$, and the corresponding overhead satisfies $H^{opt} = \Theta(\Lambda^{1/2})$.*

From the analysis of the two-level pattern, we can also observe that the overall execution overhead of any pattern comes from two distinct sources defined below.

Observation 2. *There are two types of execution overheads for a pattern:*

(1). Error-free overhead*, denoted as o_{ef}, is the total cost of all the checkpoints placed in the pattern. For a given set of checkpoints, the error-free overhead is completely determined regardless of their positions in the pattern.*

(2). Re-executed fraction overhead*, denoted as o_{re}, is the expected fraction of work that needs to be re-executed due to errors. The re-executed fraction overhead depends on both the set of checkpoints and their positions.*

For example, in the two-level pattern with n level-1 checkpoints and given values of α_i for all $i = 1, 2, \ldots, n$, the two types of overheads are given by $o_{\text{ef}} = nC_1 + C_2$ and $o_{\text{re}} = \frac{1}{2}\left(f_1 \sum_{i=1}^{n} \alpha_i^2 + f_2\right)$, where $f_\ell = \frac{\lambda_\ell}{\Lambda}$ for $\ell = 1, 2$. Assuming that checkpoints at all levels have constant costs and that the error rates at all levels are in the same order, then both o_{ef} and o_{re} can be considered as constants, i.e., $o_{\text{ef}} = O(1)$ and $o_{\text{re}} = O(1)$.

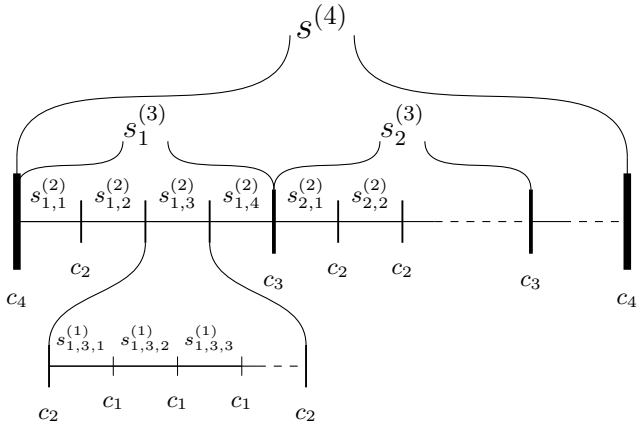


Figure 2: Example of a 4-level pattern. Here, we let $c_\ell = C_1|C_2|\cdots|C_\ell$ denote the succession of checkpoints from level 1 to level ℓ.

A trade-off exists between these two types of execution overheads, since placing more checkpoints generally reduces the re-executed work fraction when an error strikes, but it can adversely increase the overhead when the execution is error-free. Therefore, in order to achieve the best overall overhead, a resilience algorithm must seek an optimal balance between o_{ef} and o_{re}.

For a given pattern with fixed overheads o_{ef} and o_{re}, we can make the following observation based on Propositions 1 and 2, which partially characterizes the optimal pattern.

Observation 3. *For a given pattern (with fixed o_{ef} and o_{re}), the expected execution time is given by*

$$\mathbb{E} = \underbrace{W + o_{\text{ef}}}_{\substack{\text{error-free}\\ \text{execution time}}} + \underbrace{\Lambda W}_{\substack{\text{expected}\\ \#\text{ errors}}} \cdot \underbrace{o_{\text{re}} W}_{\substack{\text{re-executed work}\\ \text{in case of error}}} + O(\Lambda^{1/2}), \tag{17}$$

and the optimal pattern length and the resulting expected execution overhead of the pattern are

$$W^{opt} = \sqrt{\frac{o_{\text{ef}}}{\Lambda \cdot o_{\text{re}}}}, \tag{18}$$

$$H^{opt} = 2\sqrt{\Lambda \cdot o_{\text{ef}} \cdot o_{\text{re}}} + O(\Lambda). \tag{19}$$

Equation (19) shows that the trade-off between o_{ef} and o_{re} is manifested as the product of the two terms. Hence, in order to determine the optimal pattern, it suffices to find the pattern parameters (e.g., n and α_i) that minimize $o_{\text{ef}} \cdot o_{\text{re}}$.

3.3.2 Analysis

We now extend the analysis to derive the optimal multi-level checkpointing patterns. Generally, for a k-level pattern, each computational segment $s^{(\ell)}_{i_{k-1},\ldots,i_\ell}$ can be uniquely identified by its level ℓ as well as its position $\langle i_{k-1}, \ldots, i_\ell \rangle$ within the multi-level hierarchy. For instance, in a four-level pattern, the segment $s^{(2)}_{1,3}$ denotes the third level-2 segment inside the first level-3 segment of the pattern (see Figure 2). Note that a segment can contain multiple sub-segments at the lower levels (except

3. The original optimal solution of [8] considers faults in checkpointing but not during recoveries. We adapt its solution to exclude faults in checkpointing so to be consistent with the model in this paper for a fair comparison. The results reported herein are based on this modified solution.

for bottom-level segments) and is a sub-segment of a larger segment at a higher level (except for top-level segments). The entire pattern can be denoted as $s^{(k)}$, which is the only segment at level k.

For any segment $s^{(\ell)}_{i_{k-1},\ldots,i_\ell}$ at level ℓ, where $1 \le \ell \le k$, let $w^{(\ell)}_{i_{k-1},\ldots,i_\ell}$ denote its length. Hence, we have $w^{(\ell+1)}_{i_{k-1},\ldots,i_{\ell+1}} = \sum_{i_\ell} w^{(\ell)}_{i_{k-1},\ldots,i_\ell}$ and $w^{(k)} = W$. Also, let $n^{(\ell)}_{i_{k-1},\ldots,i_\ell}$ denote the number of sub-segments contained by $s^{(\ell)}_{i_{k-1},\ldots,i_\ell}$ at the lower level $\ell-1$. We have $n^{(1)}_{i_{k-1},\ldots,i_1} = 1$ for all $i_{k-1},\ldots,i_1$. For convenience, we further define

$$\alpha^{(\ell)}_{i_{k-1},\ldots,i_\ell} = \frac{w^{(\ell)}_{i_{k-1},\ldots,i_\ell}}{W}$$

as the fraction of the length of segment $s^{(\ell)}_{i_{k-1},\ldots,i_\ell}$ inside the pattern, and define N_ℓ to be the total number of level-ℓ segments in the entire pattern. Therefore, we have $N_k = 1$, $N_{k-1} = n^{(k)}$, and in general

$$N_\ell = \sum_{i_{k-1},\ldots,i_{\ell+1}} n^{(\ell+1)}_{i_{k-1},\ldots,i_{\ell+1}}.$$

The following proposition shows the expected time to execute a given k-level pattern.

Proposition 3. *The expected execution time of a given k-level pattern is*

$$\mathbb{E} = W + \sum_{\ell=1}^{k-1} N_\ell C_\ell + C_k + \frac{W^2}{2}\left(\sum_{\ell=1}^{k}\lambda_\ell \sum_{i_{k-1},\ldots,i_\ell}\left(\alpha^{(\ell)}_{i_{k-1},\ldots,i_\ell}\right)^2\right) + O(\Lambda^{1/2}).$$

Proof. We show that the expected time to execute any segment $s^{(h)}_{i_{k-1},\ldots,i_h}$ at level h, where $1 \le h \le k$, satisfies the following (without counting the time to execute all the checkpoints inside the segment):

$$\begin{aligned}\mathbb{E}^{(h)}_{i_{k-1},\ldots,i_h} &= w^{(h)}_{i_{k-1},\ldots,i_h} + \frac{W^2}{2}\left(\sum_{\ell=1}^{h}\lambda_\ell \sum_{i_{h-1},\ldots,i_\ell}\left(\alpha^{(\ell)}_{i_{k-1},\ldots,i_\ell}\right)^2\right)\\ &+ \Lambda_{[h+1,k]}\left(\frac{\left(w^{(h)}_{i_{k-1},\ldots,i_h}\right)^2}{2} + w^{(h)}_{i_{k-1},\ldots,i_h}\sum_{j_h=1}^{i_h-1}\mathbb{E}^{(h)}_{i_{k-1},\ldots,j_h}\right)\\ &+ w^{(h)}_{i_{k-1},\ldots,i_h}\sum_{\ell=h+2}^{k}\Lambda_{[\ell,k]}\sum_{j_{\ell-1}=1}^{i_{\ell-1}-1}\mathbb{E}^{(\ell-1)}_{i_{k-1},\ldots,j_{\ell-1}} + O(\Lambda^{1/2}),\end{aligned} \tag{20}$$

where $\Lambda_{[x,y]} = \sum_{\ell=x}^{y}\lambda_\ell$ and, if $x > y$, we define $\Lambda_{[x,y]} = 0$. The proposition can then be proven by setting $\mathbb{E} = \mathbb{E}^{(k)} + \sum_{\ell=1}^{k-1} N_\ell C_\ell + C_k$, since checkpoints are assumed to be error-free.

We now prove Equation (20) by induction on the level h. For the base case, i.e., when $h = 1$, consider a segment $s^{(1)}_{i_{k-1},\ldots,i_1}$ at the first level. Following the proof of Proposition 2 (in particular, Equation (9)), we can express its expected execution time $\mathbb{E}^{(1)}_{i_{k-1},\ldots,i_1}$, as

$$\begin{aligned}\mathbb{E}^{(1)}_{i_{k-1},\ldots,i_1} = P^{(1)}_{i_{k-1},\ldots,i_1}\Bigg(&\mathbb{E}^{\text{lost}}\big(w^{(1)}_{i_{k-1},\ldots,i_1},\Lambda\big)\\ &+\frac{\lambda_1}{\Lambda}\Big(R_1 + \mathbb{E}^{(1)}_{i_{k-1},\ldots,i_1}\Big)\\ &+\frac{\lambda_2}{\Lambda}\Big(\sum_{j=1}^{2}R_j + \sum_{j_1=1}^{i_1}\mathbb{E}^{(1)}_{i_{k-1},\ldots,j_1}\Big)\\ &+\frac{\lambda_3}{\Lambda}\Big(\sum_{j=1}^{3}R_j + \sum_{j_2=1}^{i_2-1}\mathbb{E}^{(2)}_{i_{k-1},\ldots,j_2} + \sum_{j_1=1}^{i_1}\mathbb{E}^{(1)}_{i_{k-1},\ldots,j_1}\Big)\\ &\quad\vdots\\ &+\frac{\lambda_k}{\Lambda}\Big(\sum_{j=1}^{k}R_j + \sum_{j_{k-1}=1}^{i_{k-1}-1}\mathbb{E}^{(k-1)}_{j_{k-1}} + \sum_{j_{k-2}=1}^{i_{k-2}-1}\mathbb{E}^{(k-2)}_{i_{k-1},j_{k-2}}\\ &\qquad\qquad + \cdots + \sum_{j_1=1}^{i_1}\mathbb{E}^{(1)}_{i_{k-1},\ldots,j_1}\Big)\Bigg)\\ &+\big(1 - P^{(1)}_{i_{k-1},\ldots,i_1}\big)w^{(1)}_{i_{k-1},\ldots,i_1},\end{aligned} \tag{21}$$

where $\Lambda = \sum_{\ell=1}^{k}\lambda_\ell$ is the total rate of all error sources, and $P^{(1)}_{i_{k-1},\ldots,i_1} = 1 - e^{\Lambda\cdot w^{(1)}_{i_{k-1},\ldots,i_1}}$ denotes the probability of having an error (from any level) during the execution of the segment. Simplifying Equation (21) and solving for $\mathbb{E}^{(1)}_{i_{k-1},\ldots,i_1}$ we get:

$$\begin{aligned}\mathbb{E}^{(1)}_{i_{k-1},\ldots,i_1} &= w^{(1)}_{i_{k-1},\ldots,i_1} + \frac{W^2}{2}\Lambda_{[1,k]}\left(\alpha^{(1)}_{i_{k-1},\ldots,i_1}\right)^2\\ &+ w^{(1)}_{i_{k-1},\ldots,i_1}\sum_{\ell=2}^{k}\Lambda_{[\ell,k]}\sum_{j_{\ell-1}=1}^{i_{\ell-1}-1}\mathbb{E}^{(\ell-1)}_{i_{k-1},\ldots,j_{\ell-1}} + O(\Lambda^{1/2})\\ &= w^{(1)}_{i_{k-1},\ldots,i_1} + \frac{W^2}{2}\lambda_1\left(\alpha^{(1)}_{i_{k-1},\ldots,i_1}\right)^2\\ &+ \Lambda_{[2,k]}\left(\frac{\left(w^{(1)}_{i_{k-1},\ldots,i_1}\right)^2}{2} + w^{(1)}_{i_{k-1},\ldots,i_1}\sum_{j_1=1}^{i_1-1}\mathbb{E}^{(1)}_{i_{k-1},\ldots,j_1}\right)\\ &+ w^{(1)}_{i_{k-1},\ldots,i_1}\sum_{\ell=3}^{k}\Lambda_{[\ell,k]}\sum_{j_{\ell-1}=1}^{i_{\ell-1}-1}\mathbb{E}^{(\ell-1)}_{i_{k-1},\ldots,j_{\ell-1}} + O(\Lambda^{1/2}),\end{aligned}$$

which satisfies Equation (20).

Suppose Equation (20) holds up to any segment $s^{(h)}_{i_{k-1},\ldots,i_h}$ at level h. Following the proof of Proposition 2 (in particular, the derivation of Equation (10)), we can show by induction that $\mathbb{E}^{(h)}_{i_{k-1},\ldots,i_h} = w^{(h)}_{i_{k-1},\ldots,i_h} + O(1)$. Hence, for segment $s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}$ at level $h+1$, we have:

$$\begin{aligned}\mathbb{E}^{(h+1)}_{i_{k-1},\ldots,i_{h+1}} &= \sum_{i_h}\mathbb{E}^{(h)}_{i_{k-1},\ldots,i_h}\\ &= \sum_{i_h} w^{(h)}_{i_{k-1},\ldots,i_h} + \frac{W^2}{2}\left(\sum_{\ell=1}^{h}\lambda_\ell\sum_{i_h,\ldots,i_\ell}\left(\alpha^{(\ell)}_{i_{k-1},\ldots,i_\ell}\right)^2\right)\\ &+ \Lambda_{[h+1,k]}\sum_{i_h}\left(\frac{\left(w^{(h)}_{i_{k-1},\ldots,i_h}\right)^2}{2} + w^{(h)}_{i_{k-1},\ldots,i_h}\sum_{j_h=1}^{i_h-1}w^{(h)}_{i_{k-1},\ldots,j_h}\right)\end{aligned}$$

$$+\sum_{i_h} w^{(h)}_{i_{k-1},\ldots,i_h} \sum_{\ell=h+2}^{k} \Lambda_{[\ell,k]} \sum_{j_{\ell-1}=1}^{i_{\ell-1}-1} \mathbb{E}^{(\ell-1)}_{i_{k-1},\ldots,j_{\ell-1}} + O(\Lambda^{1/2})$$

$$= w^{(h+1)}_{i_{k-1},\ldots,i_{h+1}} + \frac{W^2}{2}\left(\sum_{\ell=1}^{h}\lambda_\ell \sum_{i_h,\ldots,i_\ell}\left(\alpha^{(\ell)}_{i_{k-1},\ldots,i_\ell}\right)^2\right)$$
$$+ \Lambda_{[h+1,k]}\frac{\left(w^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2}{2}$$
$$+ w^{(h+1)}_{i_{k-1},\ldots,i_{h+1}} \sum_{\ell=h+2}^{k} \Lambda_{[\ell,k]} \sum_{j_{\ell-1}=1}^{i_{\ell-1}-1} \mathbb{E}^{(\ell-1)}_{i_{k-1},\ldots,j_{\ell-1}} + O(\Lambda^{1/2})$$

$$= w^{(h+1)}_{i_{k-1},\ldots,i_{h+1}} + \frac{W^2}{2}\left(\sum_{\ell=1}^{h+1}\lambda_\ell \sum_{i_h,\ldots,i_\ell}\left(\alpha^{(\ell)}_{i_{k-1},\ldots,i_\ell}\right)^2\right)$$
$$+ \Lambda_{[h+2,k]}\left(\frac{\left(w^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2}{2} + w^{(h+1)}_{i_{k-1},\ldots,i_{h+1}} \sum_{j_{h+1}=1}^{i_{h+1}-1} \mathbb{E}^{(h+1)}_{i_{k-1},\ldots,j_{h+1}}\right)$$
$$+ w^{(h+1)}_{i_{k-1},\ldots,i_{h+1}} \sum_{\ell=h+3}^{k} \Lambda_{[\ell,k]} \sum_{j_{\ell-1}=1}^{i_{\ell-1}-1} \mathbb{E}^{(\ell-1)}_{i_{k-1},\ldots,j_{\ell-1}} + O(\Lambda^{1/2}).$$

Hence, Equation (20) also holds for any segment at level $h+1$. This completes the proof of the proposition. □

Proposition 3 shows that, for a given k-level checkpointing pattern, the error-free overhead o_{ef} and the re-executed fraction overhead o_{re} are given as follows:

$$o_{\text{ef}} = \sum_{\ell=1}^{k-1} N_\ell C_\ell + C_k, \tag{22}$$

$$o_{\text{re}} = \frac{1}{2}\sum_{\ell=1}^{k} f_\ell \sum_{i_{k-1},\ldots,i_\ell}\left(\alpha^{(\ell)}_{i_{k-1},\ldots,i_\ell}\right)^2, \tag{23}$$

where $f_\ell = \frac{\lambda_\ell}{\Lambda}$. According to Observation 3, it remains to find parameters of the pattern such that $o_{\text{ef}} \cdot o_{\text{re}}$ is minimized.

To derive the optimal pattern, we first consider the case where o_{ef} is fixed, i.e., the set of checkpoints is given. The following proposition shows the optimal value of o_{re}.

Proposition 4. *For a k-level checkpointing pattern, suppose the number N_ℓ of checkpoints at each level ℓ is given, i.e., the error-free overhead o_{ef} is fixed (as in Equation (22)). Then, the optimal value of the re-executed work overhead is given by*

$$o^{opt}_{\text{re}} = \frac{1}{2}\left(\sum_{\ell=1}^{k-1}\frac{f_\ell}{N_\ell} + f_k\right), \tag{24}$$

and it is obtained when all the checkpoints of each level are equally spaced in the pattern.

Proof. According to Equation (23), which shows the value of o_{re} for the entire pattern, we can define the corresponding overhead for each level-h segment $s^{(h)}_{i_{k-1},\ldots,i_h}$ recursively as follows:

$$o_{\text{re}}\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right) = \frac{f_h}{2}\cdot\left(\alpha^{(h)}_{i_{k-1},\ldots,i_h}\right)^2 + \sum_{i_{h-1}} o_{\text{re}}\left(s^{(h-1)}_{i_{k-1},\ldots,i_{h-1}}\right),$$

with $o_{\text{re}}\left(s^{(0)}_{i_{k-1},\ldots,i_0}\right) = 0$ by definition.

For each segment $s^{(h)}_{i_{k-1},\ldots,i_h}$, we also define $N_\ell\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right)$ to be the total number of level-ℓ segments it contains, with $\ell \le h$. We will show that the optimal value $o^{\text{opt}}_{\text{re}}\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right)$ for the segment satisfies:

$$o^{\text{opt}}_{\text{re}}\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right) = \frac{1}{2}\left(\sum_{\ell=1}^{h}\frac{f_\ell}{N_\ell\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right)}\right)\left(\alpha^{(h)}_{i_{k-1},\ldots,i_h}\right)^2, \tag{25}$$

and it is achieved when its level-ℓ checkpoints are equally spaced, for all $\ell \le h-1$. The proposition can then be proven by setting $o^{\text{opt}}_{\text{re}} = o^{\text{opt}}_{\text{re}}\left(s^{(k)}\right)$, since $N_\ell\left(s^{(k)}\right) = N_\ell$, $N_k = 1$, and $\alpha^{(k)} = 1$.

Now, we prove Equation (25) by induction on the level h. For the base case, i.e., when $h = 1$, we have $o_{\text{re}}\left(s^{(1)}_{i_{k-1},\ldots,i_1}\right) = \frac{f_1}{2}\cdot\left(\alpha^{(1)}_{i_{k-1},\ldots,i_1}\right)^2$ by definition, and it satisfies Equation (25), because $N_1\left(s^{(1)}_{i_{k-1},\ldots,i_1}\right) = 1$. Suppose Equation (25) holds for any segment $s^{(h)}_{i_{k-1},\ldots,i_h}$ at level h. Then, for segment $s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}$ at level $h+1$, we have:

$$o_{\text{re}}\left(s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right) = \frac{f_{h+1}}{2}\cdot\left(\alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2 + \sum_{i_h} o^{\text{opt}}_{\text{re}}\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right)$$
$$= \frac{f_{h+1}}{2}\cdot\left(\alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2 + \frac{1}{2}y, \tag{26}$$

where $y = \sum_{i_h} x^{(h)}_{i_{k-1},\ldots,i_h}\cdot\left(\alpha^{(h)}_{i_{k-1},\ldots,i_h}\right)^2$, and $x^{(h)}_{i_{k-1},\ldots,i_h} = \sum_{\ell=1}^{h}\frac{f_\ell}{N_\ell\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right)}$. To minimize $o_{\text{re}}\left(s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)$ as shown in Equation (26), it suffices to solve the following minimization problem:

$$\text{minimize}\ \ y = \sum_{i_h} x^{(h)}_{i_{k-1},\ldots,i_h}\cdot\left(\alpha^{(h)}_{i_{k-1},\ldots,i_h}\right)^2,$$
$$\text{subject to}\ \ \sum_{i_h}\alpha^{(h)}_{i_{k-1},\ldots,i_h} = \alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}.$$

Since y is clearly a convex function of $\alpha^{(h)}_{i_{k-1},\ldots,i_h}$, we can readily get, using Lagrange multiplier [5], the minimum value of y as follows:

$$y_{\min} = \frac{1}{\sum_{i_h} 1/x^{(h)}_{i_{k-1},\ldots,i_h}}\cdot\left(\alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2, \tag{27}$$

which is obtained at

$$\widetilde{\alpha}^{(h)}_{i_{k-1},\ldots,i_h} = \frac{1/x^{(h)}_{i_{k-1},\ldots,i_h}}{\sum_{j_h} 1/x^{(h)}_{i_{k-1},\ldots,j_h}}\cdot\alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}. \tag{28}$$

Let us define $z = \sum_{i_h} 1/x^{(h)}_{i_{k-1},\ldots,i_h}$. We now need to solve the following maximization problem:

$$\text{maximize}\ \ z = \sum_{i_h}\frac{1}{\sum_{\ell=1}^{h}\frac{f_\ell}{N_\ell\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right)}},$$
$$\text{subject to}\ \sum_{i_h} N_\ell\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right) = N_\ell\left(s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right), \forall \ell = 1,\ldots,h.$$

Again, z is a convex function of $N_\ell\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right)$, and it can be

shown to be maximized when

$$N_\ell\left(s^{(h)}_{i_{k-1},\ldots,i_h}\right) = \frac{N_\ell\left(s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)}{n^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}}, \quad \forall \ell = 1,\ldots,h,$$

which gives $\widetilde{\alpha}^{(h)}_{i_{k-1},\ldots,i_h} = \frac{1}{n^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}} \cdot \alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}$ according to Equation (28). This implies that all level-ℓ checkpoints are also equally spaced inside segment $s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}$, for all $\ell \leq h$. The maximum value of z in this case is

$$z_{\max} = \frac{1}{\sum_{\ell=1}^{h} \frac{f_\ell}{N_\ell\left(s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)}},$$

and the optimal value of $y_{\min}$ according to Equation (27) is then given by

$$\begin{aligned} y^{\text{opt}}_{\min} &= \frac{1}{z_{\max}} \left(\alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2 \\ &= \left(\sum_{\ell=1}^{h} \frac{f_\ell}{N_\ell\left(s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)}\right) \left(\alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2. \end{aligned}$$

Substituting $y^{\text{opt}}_{\min}$ into Equation (26), we get the optimal value of $o_{\text{re}}\left(s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)$ as follows:

$$\begin{aligned} o^{\text{opt}}_{\text{re}}\left(s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right) &= \frac{f_{h+1}}{2} \cdot \left(\alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2 + \frac{1}{2} y^{\text{opt}}_{\min} \\ &= \frac{1}{2}\left(\sum_{\ell=1}^{h+1} \frac{f_\ell}{N_\ell\left(s^{(h+1)}_{i_{k-1},\ldots,i_h}\right)}\right) \left(\alpha^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}\right)^2. \end{aligned}$$

This shows that Equation (25) also holds for segment $s^{(h+1)}_{i_{k-1},\ldots,i_{h+1}}$ at level $h+1$ and, hence, completes the proof of the proposition. □

We are now ready to characterize the optimal k-level pattern. The result is stated in the following theorem.

Theorem 2. *A first-order approximation to the optimal k-level pattern and its overhead are characterized by*

$$W^{opt} = \sqrt{\frac{2\left(\sum_{\ell=1}^{k-1} N^{opt}_\ell C_\ell + C_k\right)}{\sum_{\ell=1}^{k-1} \frac{\lambda_\ell}{N^{opt}_\ell} + \lambda_k}}, \tag{29}$$

$$N^{opt}_\ell = \sqrt{\frac{\lambda_\ell}{C_\ell} \cdot \frac{C_k}{\lambda_k}}, \quad \forall \ell = 1,2,\ldots,k-1, \tag{30}$$

$$H^{opt} = \sum_{\ell=1}^{k} \sqrt{2\lambda_\ell C_\ell} + O(\Lambda). \tag{31}$$

Proof. From Observation 3, Equation (22) and Proposition 4, we know that the optimal pattern can be obtained by minimizing the following function:

$$F = o_{\text{ef}} \cdot o^{\text{opt}}_{\text{re}} = \frac{1}{2}\left(\sum_{\ell=1}^{k-1} N_\ell C_\ell + C_k\right)\left(\sum_{\ell=1}^{k-1} \frac{f_\ell}{N_\ell} + f_k\right).$$

We first compute the optimal number of checkpoints at each level using a *two-phase iterative* method. Towards this end, let us define

$$\begin{aligned} o_{\text{ef}}(h) &= \sum_{\ell=h}^{k-1} N_\ell C_\ell + C_k, \\ o^{\text{opt}}_{\text{re}}(h) &= \frac{1}{2}\left(\sum_{\ell=h}^{k-1} \frac{f_\ell}{N_\ell} + f_k\right). \end{aligned}$$

In the first phase, we set initially

$$F(1) = o_{\text{ef}}(1) \cdot o^{\text{opt}}_{\text{re}}(1).$$

The optimal value of N_1 that minimizes $F(1)$ can then be obtained by setting

$$\begin{aligned} \frac{\partial F(1)}{\partial N_1} &= C_1 o^{\text{opt}}_{\text{re}}(1) - o_{\text{ef}}(1)\frac{f_1}{2N_1^2} \\ &= C_1\left(\frac{f_1}{2N_1} + o^{\text{opt}}_{\text{re}}(2)\right) - \left(N_1 C_1 + o_{\text{ef}}(2)\right)\frac{f_1}{2N_1^2} \\ &= C_1 o^{\text{opt}}_{\text{re}}(2) - o_{\text{ef}}(2)\frac{f_1}{2N_1^2} = 0, \end{aligned}$$

which gives $N^{\text{opt}}_1 = \sqrt{\frac{f_1}{C_1} \cdot \frac{o_{\text{ef}}(2)}{2o^{\text{opt}}_{\text{re}}(2)}}$. Substituting it into $F(1)$ and simplifying, we can get the value of F after the first iteration as

$$F(2) = \frac{1}{2}\left(\sqrt{f_1 C_1} + \sqrt{o_{\text{ef}}(2) \cdot o^{\text{opt}}_{\text{re}}(2)}\right)^2.$$

Repeating the above process, we can get the optimal value of F after $k-1$ iterations as

$$F^{\text{opt}} = F(k) = \frac{1}{2}\left(\sum_{\ell=1}^{k} \sqrt{f_\ell C_\ell}\right)^2, \tag{32}$$

and the optimal value of N_ℓ as

$$N^{\text{opt}}_\ell = \sqrt{\frac{f_\ell}{C_\ell} \cdot \frac{o_{\text{ef}}(\ell+1)}{2o^{\text{opt}}_{\text{re}}(\ell+1)}}, \quad \forall \ell = 1,2,\ldots,k-1. \tag{33}$$

In the second phase, we first compute from Equation (33)

$$\begin{aligned} N^{\text{opt}}_{k-1} &= \sqrt{\frac{f_{k-1}}{C_{k-1}} \cdot \frac{o_{\text{ef}}(k)}{2o^{\text{opt}}_{\text{re}}(k)}} \\ &= \sqrt{\frac{f_{k-1}}{C_{k-1}} \cdot \frac{C_k}{f_k}} \\ &= \sqrt{\frac{\lambda_{k-1}}{C_{k-1}} \cdot \frac{C_k}{\lambda_k}}. \end{aligned}$$

Substituting it into N^{opt}_{k-2}, we obtain:

$$\begin{aligned} N^{\text{opt}}_{k-2} &= \sqrt{\frac{f_{k-2}}{C_{k-2}} \cdot \frac{o_{\text{ef}}(k-1)}{2o^{\text{opt}}_{\text{re}}(k-1)}} \\ &= \sqrt{\frac{\lambda_{k-2}}{C_{k-2}} \cdot \frac{N^{\text{opt}}_{k-1} C_{k-1} + C_k}{\frac{\lambda_{k-1}}{N^{\text{opt}}_{k-1}} + \lambda_k}} \\ &= \sqrt{\frac{\lambda_{k-2}}{C_{k-2}} \cdot \frac{\sqrt{\frac{\lambda_{k-1}}{\lambda_k} C_{k-1} C_k} + C_k}{\sqrt{\lambda_{k-1}\lambda_k \frac{C_{k-1}}{C_k}} + \lambda_k}} \end{aligned}$$

$$= \sqrt{\frac{\lambda_{k-2}}{C_{k-2}} \cdot \frac{C_k\left(\sqrt{\frac{\lambda_{k-1}}{\lambda_k}\cdot\frac{C_{k-1}}{C_k}}+1\right)}{\lambda_k\left(\sqrt{\frac{\lambda_{k-1}}{\lambda_k}\cdot\frac{C_{k-1}}{C_k}}+1\right)}}$$
$$= \sqrt{\frac{\lambda_{k-2}}{C_{k-2}}\cdot\frac{C_k}{\lambda_k}}.$$

Repeating the above process iteratively, we can compute the optimal values of N_ℓ^{opt} for $\ell = k-3,\ldots,2,1$, as given in Equation (30) by using values of $N_{k-1}^{\text{opt}},\ldots,N_{\ell+1}^{\text{opt}}$.

The optimal pattern length, according to Equation (18), can be expressed as $W^{\text{opt}} = \sqrt{\frac{o_{\text{ef}}}{\Lambda\cdot o_{\text{re}}^{\text{opt}}}}$, which turns out to be Equation (29) with the optimal values of N_ℓ^{opt}.

The optimal overhead, according to Equations (19) and (32), can be expressed as $H^{\text{opt}} = 2\sqrt{\Lambda\cdot F^{\text{opt}}}+O(\Lambda)$, which gives rise to Equation (31). This completes the proof of the theorem. □

Since Proposition 4 shows that all the checkpoints of each level are equally spaced in the pattern, we can readily obtain the following corollary.

Corollary 1. *In an optimal k-level pattern, the number of level-ℓ checkpoints between any two consecutive level-$(\ell+1)$ checkpoints is given by*

$$n_\ell^{opt} = \frac{N_\ell^{opt}}{N_{\ell+1}^{opt}} = \sqrt{\frac{\lambda_\ell}{\lambda_{\ell+1}}\cdot\frac{C_{\ell+1}}{C_\ell}}. \tag{34}$$

for all $\ell = 1,\ldots,k-1$.

Remarks. The optimal k-level pattern derived in this section has a *rational* number of segments, while the optimal *integer* solution could be much harder to compute. In Section 4, we use rounding to derive a practical solution. Still, Equation (31) provides a lower bound on the optimal overhead, which is met very closely in all our experimental scenarios.

3.4 Optimal subset of levels

The preceding section characterizes the optimal pattern by using k levels of checkpoints. In many practical cases, there is no obligation to use all available levels. This section addresses the problem of selecting the optimal subset of levels in order to minimize the overall execution overhead.

3.4.1 Checkpoint cost models

So far, we have assumed that all the checkpoint costs are fixed under a multi-level checkpointing scheme. In practice, the checkpoint costs may vary depending upon the implementation, and upon the subset of selected levels. In order to determine the optimal subset, we identify the following two checkpoint cost models:

- **Fixed independent costs.** The checkpoint cost C_ℓ at level ℓ is the cost paid to save data at level ℓ, *independently* of the subset of levels used. In this model, the checkpoint costs stay the same for all possible subsets.
- **Incremental costs.** The checkpointing cost C_ℓ at level ℓ is the *additional* cost paid to save data when going from level $\ell-1$ to ℓ. In this model, the checkpoint cost at a particular level depends on the subset of levels selected.

For example, with $k=2$ levels and $C_1=10, C_2=20$, two subsets are possible: $\{1,2\}$ and $\{2\}$. In the fixed independent cost model, these costs will stay unchanged regardless of the subset chosen. In the incremental cost model, since C_2 is the *additional* cost paid after C_1 is done, when using subset $\{2\}$, i.e., only placing level-2 checkpoints in the pattern, we need to adjust its cost as $C_2' = 10+20 = 30$. In both cases, once the subset is decided, the checkpoint costs at the selected levels can be computed and therefore considered as fixed constants. The theoretical analysis presented in Section 3.3 can then be used to compute the optimal pattern.

But how to determine the optimal subset of levels? Consider again the example with $k=2$ levels. In the incremental cost model, Equation (31) suggests that the optimal solution (ignoring lower-order terms) uses both levels if and only if

$$\sqrt{2\lambda_1 C_1}+\sqrt{2\lambda_2 C_2} \le \sqrt{2\left(\lambda_1+\lambda_2\right)\left(C_1+C_2\right)}$$
$$\Leftrightarrow \quad 0 \le \left(\sqrt{\lambda_1 C_2}-\sqrt{\lambda_2 C_1}\right)^2,$$

which is always true when assuming $\lambda_1 \ge \lambda_2$ and $C_1 \le C_2$. We can easily apply the same argument to show that the optimal subset must contain *all* levels available as long as all checkpoint costs are positive.

In the fixed independent cost model, however, it is not clear whether all available levels should be used. Consider the same example with $k=2$ levels, and define $\alpha = \frac{\lambda_2}{\lambda_1}$ and $\beta = \frac{C_2}{C_1}$. The optimal solution uses both levels if and only if

$$\sqrt{2\lambda_1 C_1}+\sqrt{2\lambda_2 C_2} \le \sqrt{2\left(\lambda_1+\lambda_2\right)C_2}$$
$$\Leftrightarrow \quad 4\alpha\beta \le (\beta-1)^2,$$

which is not true when $\alpha = 0.5$ and $\beta = 2$. In this case, using only level-2 checkpoints leads to a smaller overhead.

3.4.2 Dynamic programming algorithm

In the fixed independent cost model, the optimal subset of levels in a general k-level pattern could well depend on the checkpoint costs and error rates of different levels. One can enumerate all 2^{k-1} possible subsets and select the one that leads to the smallest overhead. The following theorem presents a more efficient dynamic programming algorithm when the number k of levels is large.

Theorem 3. *Suppose there are k levels of checkpoints available and their costs are fixed. Then, the optimal subset of levels to use can be obtained by dynamic programming in $O(k^2)$ time.*

Proof. Let $\mathcal{S}^{\text{opt}}(h) \subseteq \{0,1,\ldots,h\}$ denote the optimal subset of levels used by a pattern that is capable of handling errors up to level h, and let $H^{\text{opt}}(h)$ denote the corresponding optimal overhead (ignoring lower-order terms) incurred by the pattern. Define $\mathcal{S}^{\text{opt}}(0) = \emptyset$ and $H^{\text{opt}}(0) = 0$. Recall that $\Lambda_{[x,y]} = \sum_{\ell=x}^{y}\lambda_\ell$. We can compute $H^{\text{opt}}(h)$ using the following dynamic programming formulation:

$$H^{\text{opt}}(h) = \min_{0\le\ell\le h-1}\left\{H^{\text{opt}}(\ell)+\sqrt{2\Lambda_{[\ell+1,h]}C_h}\right\}, \tag{35}$$

and the optimal subset is $\mathcal{S}^{\text{opt}}(h) = \mathcal{S}^{\text{opt}}(\ell^{\text{opt}})\bigcup\{h\}$, where ℓ^{opt} is the value of ℓ that yields the minimum $H^{\text{opt}}(h)$.

The optimal subset of levels to handle all k levels of errors is then given by $\mathcal{S}^{\text{opt}}(k)$ with the optimal overhead $H^{\text{opt}}(k)$. The complexity is clearly quadratic in the total number of levels. □

4 SIMULATIONS

In this section, we conduct a set of simulations whose goal is threefold: (i) to verify the accuracy of the first-order approximation; (ii) to confirm the optimality of the subset of levels found by the dynamic programming algorithm; and (iii) to evaluate the performance of our approach and to compare it with other multi-level checkpointing algorithms. After introducing the simulation setup in Section 4.1, we proceed in two steps. First, in Section 4.2, we instantiate the model with realistic parameters from the literature and run simulations for all possible subsets of levels and roundings. Then, in Section 4.3, we instantiate the model with different test cases from the recent work of Di et al. [7], [8] on multilevel checkpointing and compare the overheads obtained with three approaches: (a) Young/Daly's classical formula; (b) our first-order approximation formula; and (c) Di et al.'s iterative/optimal algorithm. The simulator code is publicly available at *http://perso.ens-lyon.fr/aurelien.cavelan/multilevel.zip*, so that interested readers can experiment with it and instantiate the model with parameters of their own choice.

4.1 Simulation setup

Checkpoint and recovery costs both depend on the volume of data to be saved, and are mostly determined by the hardware resource used at each level. As such, we assume that recovery cost for a given level is equivalent to the corresponding checkpointing cost, i.e., $R_\ell = C_\ell$ for $1 \leq \ell \leq k$ (unless specified otherwise). This a common assumption [14], [7], even though in practice the recovery cost can be expected to be *somewhat* smaller than the checkpoint cost [7], [8]. All costs are fixed and independent (as discussed in Section 3.4.1).

The simulator is fed with k levels of errors and their MTBFs $\mu_\ell = 1/\lambda_\ell$, as well as the resilience parameters C_ℓ and R_ℓ. For each of the 2^{k-1} possible subsets of levels (the last level is always included), we take the optimal pattern given in Theorem 2 and Corollary 1, and then try all possible roundings (floor and ceiling) based on the optimal (rational) number of checkpoints (n_ℓ^{opt} given in Equation (34)). For each rounding, we compare the following three overheads:

- **Simulated overhead**, obtained by running the simulation 10000 times and averaging the results;
- **Corresponding theoretical overhead**, obtained from Equations (19), (22) and (24) using the integer solution that corresponds to the rounding;
- **Theoretical lower bound**, obtained from Equation (31) with the optimal rational solution.

In the following, we associate Young/Daly's classical formula, defined as $W^{opt} = \sqrt{\frac{2C}{\Lambda}}$, with the highest checkpointing level available, i.e., $C = C_k$. Note that in this case, Young/Daly's formula and Equation (29) can be used interchangeably, and the corresponding theoretical overhead is obtained with $H^{opt} = \sqrt{2\Lambda C}$.

4.2 Assessing accuracy of first-order approximation

In this section, we run simulations with two sets of parameters, described in Table 1. For each set of parameters, we consider all possible subsets of levels. Then, for each subset, we compute the optimal pattern length and number of checkpoints to be used at each level. We show the accuracy of our approach in both scenarios, and we confirm the optimality of the subset of levels returned by the dynamic programming algorithm.

Table 1: Sets of parameters (A) and (B) used as inputs for simulations.

Set	From	Level	1	2	3	4
(A)	Moody et al. [14]	C (s)	0.5	4.5	1051	-
		MTBF (s)	5.00e6	5.56e5	2.50e6	-
(B)	Balaprakash et al. [1]	C (s)	10	30	50	150
		MTBF (s)	3.60e4	7.20e4	1.44e5	7.20e5

4.2.1 Using set of parameters (A)

The first set of parameters (shown in set (A) of Table 1) corresponds to the Coastal platform, a medium-sized HPC system of 1104 nodes at the Lawrence Livermore National Laboratory (LLNL). The Coastal platform has been used to evaluate the Scalable Checkpoint/Restart (SCR) library by Moody et al. [14], who provided accurate measurements for the checkpoint costs using real applications (given in the first row of Table 1). There are $k = 3$ levels of checkpoints. First-level checkpoints are written to the local RAMs of the nodes, and this is the fastest method ($0.5s$). Second-level checkpoints are also written to local RAMs, but small sets of nodes collectively compute and store parity redundancy data, which takes a little while longer ($4.5s$). Lastly, Lustre is used to store third-level checkpoints onto the parallel file system, which takes significantly longer time ($1051s$). Failures were analyzed in [14], and the error rates are given in the second row of Table 1. Note that the error rate at level 2 is higher than those of levels 1 and 3.

Results: Table 2 and Figure 3 present the simulation results. Table 2 shows, from left to right, the subset of levels used, the number of checkpoints computed by our first-order approximation formula for each possible rounding (N_1, N_2, N_3), the corresponding optimal pattern length ($W^{\text{opt}}(s)$), the simulated overhead (Sim. Ov.), the corresponding theoretical overhead (Th. Ov.), the absolute and relative differences of these two overheads (Ab. Diff. = 100 $\times$ (Sim. Ov. - Th. Ov.), and Rel. Diff. = 100 $\times$ (Sim. Ov. - Th. Ov.)/Sim. Ov.), and finally the theoretical lower bound for this subset (Th. L.B.).

With $k = 3$, there are four possible subsets of levels, and both the best simulated overhead and the corresponding theoretical overhead are achieved for the subset $\{2, 3\}$, with $N_2 = 35$ and $N_3 = 1$ (highlighted in **bold** in the table). First, the difference between the simulated and theoretical overheads is very small, with a difference $< 0.7\%$ in absolute values, and a relative difference ranging from 2.9% (for subset $\{1, 2, 3\}$) to 8.14% (for subset $\{3\}$), which shows the accuracy of the first-order approximation for this set of parameters. The simulated overhead is always higher than the theoretical one, which is expected, because the first-order approximation ignores some lower-order terms. Next, we observe that, for each subset, all roundings of the number of checkpoints yield similar overheads on this platform, and the difference between the best and worst roundings is almost negligible.

Furthermore, using the best subset ($\{2, 3\}$) improves the overhead by over 50% compared to using level-3 checkpoints alone (as in Young/Daly's result). This is indeed the subset returned by the dynamic programming algorithm, and the result matches closely the minimum theoretical lower bound. Finally, comparing our result to the one obtained by the optimal two-level algorithm by Di et al. [8] on this best subset, we see that the simulated overheads are similar under the optimal subset, as the patterns found using both approaches share the same number of checkpoints and the pattern lengths are also almost identical.

Table 2: Simulation results using set of parameters (A).

Levels	N_1	N_2	N_3	W^{opt} (s)	Sim. Ov.	Th. Ov.	Abs. Diff.	Rel. Diff.	Th. L.B.
{3}	-	-	1	2.96e4	7.74e-2	7.11e-2	0.63%	8.14%	7.11e-2
{1,3}	14	-	1	3.09e4	7.40e-2	6.85e-2	0.55%	7.43%	6.85e-2
	13	-	1	3.09e4	7.39e-2	6.85e-2	0.54%	7.31%	
{2,3}	-	35	1	7.27e4	**3.44e-2**	**3.33e-2**	0.11%	3.20%	**3.33e-2**
	-	34	1	7.25e4	3.46e-2	**3.33e-2**	0.13%	3.76%	
{1,2,3}	33	33	1	7.27e4	3.46e-2	3.35e-2	0.11%	3.18%	3.35e-2
	32	32	1	7.24e4	3.45e-2	3.35e-2	0.10%	2.90%	

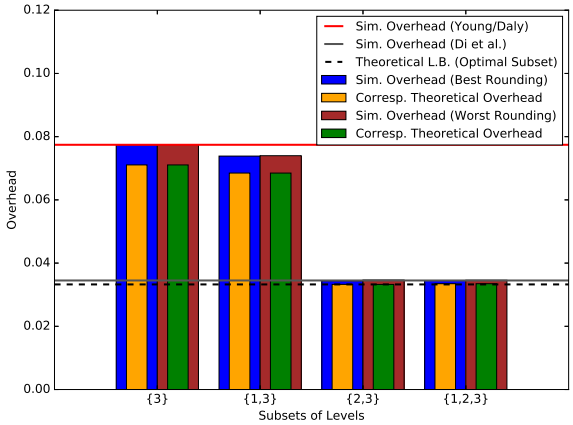


Figure 3: Simulated and (corresponding) theoretical overheads for all possible subsets of levels with the best and worst roundings for each subset using set of parameters (A).

4.2.2 *Using set of parameters (B)*

The second set of parameters correspond to the execution of the LAMMPS application on the large BG/Q platform Mira at the Argonne National Laboratory (ANL) [1]. The parameters are presented in set (B) of Table 1. In this setting, the Fault Tolerance Interface (FTI) [2] was used, which has four checkpoint levels ($k = 4$): Local checkpoint; Local checkpoint + Partner-copy; Local checkpoint + Reed-Solomon coding; and PFS-based checkpoint. The MTBFs correspond to the failure rates typically observed for petascale HPC applications [2], [14], [7].

Table 3: Simulation results using set of parameters (B).

Levels	N_1	N_2	N_3	N_4	W^{opt} (s)	Sim. Ov.	Th. Ov.	Abs. Diff.	Rel. Diff.	Th. L.B.
{4}	-	-	-	1	2.45e3	1.43e-1	1.22e-1	1.9%	13.3%	1.22e-1
{1,4}	5	-	-	1	3.79e3	1.18e-1	1.05e-1	1.3%	11.0%	1.05e-1
	4	-	-	1	3.61e3	1.18e-1	1.05e-1	1.3%	11.0%	
{2,4}	-	5	-	1	6.00e3	1.11e-1	1.00e-1	1.1%	9.9%	1.00e-1
{3,4}	-	-	11	1	1.55e4	9.96e-2	9.02e-2	0.94%	9.44%	9.01e-2
	-	-	10	1	1.44e4	9.91e-2	9.01e-2	0.90%	9.08%	
{1,2,4}	9	3	-	1	6.41e3	1.11e-1	1.03e-1	0.8%	7.2%	1.02e-1
	6	2	-	1	5.21e3	1.13e-1	1.04e-1	0.9%	8.0%	
	6	3	-	1	5.84e3	1.11e-1	1.03e-1	0.8%	7.2%	
	4	2	-	1	4.74e3	1.17e-1	1.05e-1	1.2%	10.3%	
{1,3,4}	21	-	7	1	1.58e4	9.72e-2	8.99e-2	0.73%	7.51%	**8.96e-2**
	18	-	6	1	1.40e4	9.82e-2	**8.98e-2**	0.84%	8.55%	
	14	-	7	1	1.04e4	**9.68e-2**	9.01e-2	0.67%	6.92%	
	12	-	6	1	1.26e4	9.85e-2	9.04e-2	0.81%	8.22%	
{2,3,4}	-	16	4	1	1.70e4	1.07e-1	9.75e-2	0.95%	8.9%	9.68e-2
	-	12	3	1	1.36e4	1.04e-1	9.73e-2	0.67%	6.4%	
	-	12	4	1	1.47e4	1.05e-1	9.68e-2	0.82%	7.8%	
	-	9	3	1	1.17e4	1.05e-1	9.75e-2	0.75%	7.1%	
{1,2,3,4}	24	8	4	1	1.66e4	1.09e-1	1.00e-1	0.9%	8.2%	9.92e-2
	18	6	3	1	1.32e4	1.08e-1	9.99e-2	0.81%	7.5%	
	12	4	4	1	1.15e4	1.11e-1	1.03e-1	0.8%	7.2%	
	9	3	3	1	9.17e3	1.14e-1	1.05e-1	0.9%	7.9%	
	16	8	4	1	1.51e4	1.08e-1	9.95e-2	0.85%	7.9%	
	12	6	3	1	1.20e4	1.09e-1	1.00e-1	0.9%	8.3%	
	8	4	4	1	1.05e4	1.16e-1	1.05e-1	1.1%	9.5%	
	6	3	3	1	8.33e3	1.19e-1	1.08e-1	1.1%	9.2%	

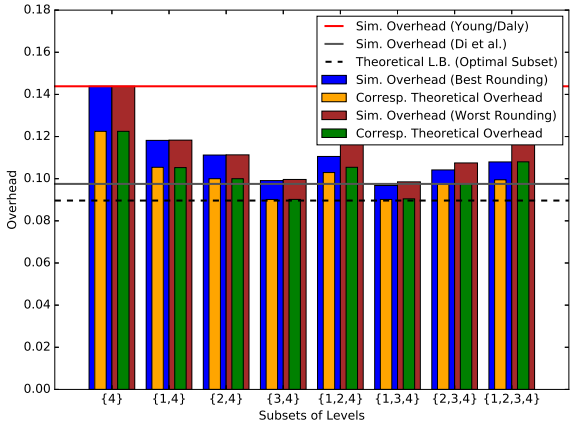


Figure 4: Simulated and (corresponding) theoretical overheads for all possible subsets of levels with the best and worst roundings for each subset using set of parameters (B).

Results: Table 3 and Figure 4 present the simulation results for this set of parameters. There are 8 possible subsets of levels. As before, we observe that the theoretical overhead is always slightly smaller than the simulated one, with an absolute difference of less than 2%, and a relative difference between 6-14%, demonstrating the accuracy of the model. Again, the results are very close to the theoretical lower bound. For this platform, the simulated overheads vary from 9.68% (with optimal subset of levels $\{1, 3, 4\}$ found by the dynamic programming algorithm) to 14.3% (with level-4 checkpoints alone). For a given subset of levels, the rounding does not play a significant role, as W^{opt} is also adjusted accordingly (increased or decreased) as a result of rounding. For instance, we observe that, for subset $\{1, 2, 3, 4\}$, the numbers of checkpoints at levels 1 and 2 are halved for the third rounding compared to the first rounding in Table 3, but W^{opt} is also reduced by 31%, so that for the same amount of work, the number of checkpoints does not change by much. We can also see that the pattern length W^{opt} for the smallest overhead is around $10400s$, but only $2450s$ for the largest overhead. In fact, the largest pattern lengths are obtained for the *highest cumulated checkpoint cost*, which turns out to be $830s$ for $\{2, 3, 4\}$ with $N_2 = 16, N_3 = 4, N_4 = 1$, and for $\{1, 2, 3, 4\}$ with $N_1 = 24, N_2 = 8, N_3 = 4$ and $N_4 = 1$. This is because using more checkpoints both increases the error-free overhead and reduces the time lost due to re-executions upon errors. As a consequence, and to mitigate the aforementioned overhead, the length of the pattern increases (e.g., $W^{opt} = 17000s$ for $\{2, 3, 4\}$ and $W^{opt} = 16600s$ for $\{1, 2, 3, 4\}$). And the converse is also true: when using fewer checkpoints, the error-free overhead decreases and the time lost upon errors increases. In order to compensate, the pattern length decreases (e.g., $W^{opt} = 8330s$ for $\{1, 2, 3, 4\}$ with $N_1 = 6, N_2 = 3, N_3 = 3$ and $N_4 = 1$).

We note that, in this case, our first-order solution slightly outperforms the iterative method by Di et al. [7] on multi-level checkpointing (with a simulated overhead of 9.68e-2 compared to 9.75e-2). The reason is that their algorithm computes a solution under the independent checkpointing model, i.e., checkpoints at different levels are taken according to different independent periods. However, it is not clear how such a model can be implemented in practice due to the difficulties as explained in Section 1 and the different options to rollback to a checkpoint in case of a fault. Therefore, we transformed their result to a pattern-based solution by rounding the differ-

Table 4: Set of parameters (C) used as input for simulations.

Set (C), from Di et al. [7]					
	Level	1	2	3	4
Case #A	$C\ (s)$	8	10	80	90
	$R\ (s)$	8	10	80	90
	MTBF (s)	2160	1440	8640	21600
Case #B	$C\ (s)$	1	20	60	70
	$R\ (s)$	1	10	30	35
	MTBF (s)	864	864	1080	1440

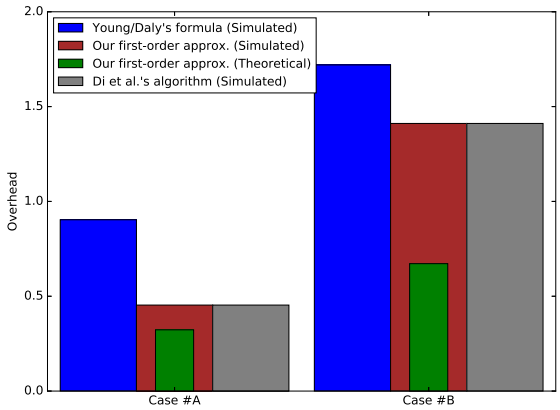


Figure 5: Performance comparison of the three different approaches using two cases from Di et al. [7].

ent numbers of checkpoints obtained using their algorithm to create equal number of checkpoints at level $\ell - 1$ between two consecutive level-ℓ checkpoints. Although the best rounding is selected here for comparison, the result can still change drastically the number of checkpoints computed by their initial rational solution without changing the pattern length, thus increasing the overhead.

4.3 Comparing performance of different approaches

In this section, we conduct simulations using settings from Di et al.'s recent work on multilevel checkpointing, which comprises two cases with four levels [7] and eight cases with two levels [8], thus covering a wide range of configurations. For each case, we compare the performance of three different approaches: (a) Young/Daly's classical formula; (b) our first-order approximation formula; and (c) Di et al.'s iterative algorithm.

4.3.1 Using set of parameters (C)

We first run simulations for Cases #A and #B, whose parameters are presented in Table 4. These parameters are based on the FTI multilevel checkpointing model and have been used by Di et al. [7] to evaluate the performance of their approach. Note that the recovery cost is about half that of the checkpointing cost in Case #B.

In their work, Di et al. considered independent checkpointing periods, as opposed to the nested method based on periodic patterns (as discussed in Section 1). Although they provided an optimal solution, an iterative approach was used to compute it numerically in contrast to the simple formula we propose in this paper. Recall that using independent checkpointing periods allows checkpoints at different levels to be taken simultaneously, which can hardly be done in practice. Adapting their solution to our model results in rational numbers of checkpoints, and we again use rounding to resolve this issue. We find that, using the best roundings for both approaches, their solution turns out to be very similar to ours (with the same number of checkpoints, and close periods with $< 1\%$ difference).

Table 5: Set of parameters (D) used as input for simulations.

Set (D), from Di et al. [8]							
	Level	1	2		Level	1	2
Case 1	$C\ (s)$	20	50	Case 5	$C\ (s)$	10	40
	MTBF (s)	3600	21600		MTBF (s)	432	2160
Case 2	$C\ (s)$	20	50	Case 6	$C\ (s)$	100	20
	MTBF (s)	1728	8640		MTBF (s)	432	2160
Case 3	$C\ (s)$	20	100	Case 7	$C\ (s)$	40	200
	MTBF (s)	864	4320		MTBF (s)	288	1440
Case 4	$C\ (s)$	10	40	Case 8	$C\ (s)$	50	300
	MTBF (s)	864	4320		MTBF (s)	216	1440

Results: Figure 5 presents the overheads for both cases. First, we observe that Di et al.'s optimal iterative algorithm has almost identical performance to our solution, with a simulated overhead around 45% for Case #A and 140% for Case #B under both approaches. However, using Young/Daly's formula to checkpoint only at the highest level yields significantly worse overheads (around 90% for Case #A and 170% for Case #B). Overall, our solution is as good as Di et al.'s optimal numerical one (but has much less complexity), and it is up to 45% better than Young/Daly's formula in Case #A and 30% better in Case #B.

Note that the corresponding theoretical overhead of our solution is close to the simulated one for Case #A, but starts to diverge for Case #B. This is because first-order approximation is only accurate when the resilience parameters and pattern length are small compared to the MTBF, which is no longer true for Case #B. Specifically, we have:

- In Case #A, the optimal subset of levels is $\{2, 4\}$. The optimal pattern has length $W^{\text{opt}} = 1052s$ and consists of $N_2 = 8$ level-2 checkpoints followed by $N_4 = 1$ level-4 checkpoint, meaning that we have a level-2 checkpoint every $131.5s$ of computation. So a level-2 checkpoint is saved every $141.5s$ and a level-4 checkpoint is saved every 1222s. On the other hand, the combined MTBF for errors at levels 1 and 2 (handled by level-2 checkpoints) is $864s$ and the combined MTBF for errors at levels 3 and 4 (handled by level-4 checkpoints) is $6171s$. Hence, we have $\frac{141.5}{864} = 0.164$ and $\frac{1222}{6171} = 0.198$, which are reasonably small, making our solution accurate.
- In Case #B, the optimal subset of levels is $\{1, 4\}$, and the optimal pattern has $W^{\text{opt}} = 223s$, $N_1 = 5$ and $N_4 = 1$. Thus, we have a level-1 checkpoint every $44.6s$ of computation. So a level-1 checkpoint is saved every $45.6s$ and a level-4 checkpoint is saved every $298s$. The MTBF for errors at level 1 is $864s$ and the combined MTBF for errors at levels 2, 3 and 4 (handled by level-4 checkpoints) is $360s$. Thus, we have $\frac{44.6}{864} = 0.052$, which is fine, but $\frac{298}{360} = 0.828$, which is too high and essentially makes the first-order solution inaccurate.

Despite the difference between the theoretical and simulated overheads under Case #B, the proximity of our solution to Di et al.'s optimal numerical solution nevertheless shows the usefulness of first-order approximation for determining the optimal multi-level checkpointing patterns.

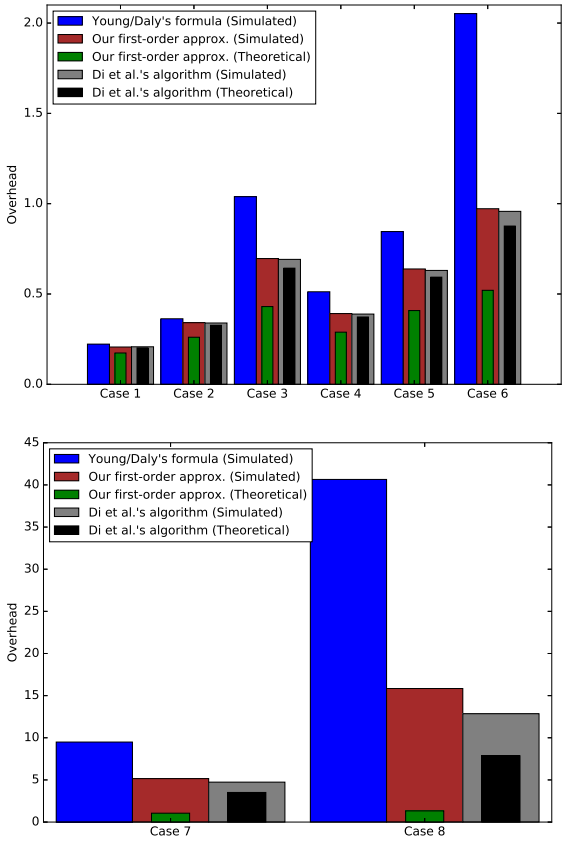


Figure 6: Performance comparison of the three different approaches using 8 cases from Di et al. [8].

4.3.2 *Using set of parameters (D)*

Finally, we run simulations for eight cases, whose parameters are presented in Table 5. These parameters have been used by Di et al. [8] to evaluate their two-level checkpointing model, and as such, each case consists of only two checkpointing levels. In their work, the authors proposed an optimal solution by solving complex mathematical equations using numerical method. Again, for each case, we compare the simulated overheads obtained with the three different approaches.

In this set of parameters, the MTBF has a large variation, ranging from more than 1 hour (Case 1) to less than 4 minutes (Case 8). Similarly, the checkpointing costs vary from $10s$ (Cases 4 and 5) to $300s$ (Case 8). Note that Cases 7 and 8 have both very short MTBFs and very high checkpointing costs, resulting in a lot of errors and recoveries. In particular, the checkpointing cost at level 2 in Case 8 ($300s$) is larger than the MTBF at level 1 ($216s$).

Results: Figure 6 presents the simulation results for the eight cases. First, we observe that the optimal algorithm by Di et al. only yields a slightly better simulated overhead compared to our simple first-order approximation solution (by less than 2% in Cases 1 to 6). However, our solution always improves significantly over Young/Daly's formula, from 2% (Case 1) up to 100% (Case 6). Due to their short MTBFs, Cases 7 and 8 stand out and incur much higher overheads compared to the first six cases (thus their results are presented in a separate plot). Still, considering Case 8, we are able to improve over Young/Daly's solution by as much as 2500% (in absolute value of the overhead), and we are off the optimal simulated overhead by only 300%. In addition, Figure 6 shows the theoretical overheads obtained both with our formula and the solution provided by Di et al. in [8]. As expected, our first-order approximation remains accurate when the MTBF is large, as in Cases 1, 2 and 4. However, it becomes less accurate with shorter MTBFs and higher error rates, especially in Cases 7 and 8 (which do not represent healthy HPC platforms).

4.4 Summary of results

From the simulation results, we conclude that first-order approximation remains a valuable performance model for evaluating checkpointing solutions in HPC systems (as long as the error rates stay reasonably low). We have demonstrated, through an extensive set of simulations with a wide range of parameters, the usefulness of multi-level checkpointing (over using only one level of checkpoints) with significantly reduced overheads. The results also corroborate the analytical study by showing the benefit of selecting an optimal subset of levels among all the levels available. Overall, our approach achieves the optimal or near-optimal performance in almost all cases, except when the MTBF is too small, in which case even the optimal solution yields an unacceptably high overhead (e.g., Case 8 of Table 5).

5 Conclusion

This paper has studied multi-level checkpointing protocols, where different levels of checkpoints can be set; lower levels deal with frequent errors that can be recovered at low cost (for instance with a memory copy), while higher levels allow us to recover from all errors, such as node failures (for instance with a copy in stable storage). We consider a general scenario with k levels of faults, and we provide explicit formulas to characterize the optimal checkpointing pattern, up to first-order approximation. The overhead turns out to be of the order of $\sum_{\ell=1}^{k} \sqrt{2\lambda_\ell C_\ell}$, which elegantly extends Young/Daly's classical formula.

The first-order approximation to the optimal k-level checkpointing pattern uses rational numbers of checkpoints, and we prove that all segments should have equal lengths. We corroborate the theoretical study by an extensive set of simulations, demonstrating that greedily rounding the rational values leads to an overhead very close to the lower bound. Furthermore, we provide a dynamic programming algorithm to determine those levels that should be selected, and the simulations confirm the optimality of the subset of levels returned by the dynamic programming algorithm.

The problem of finding a first-order optimal pattern with an integer number of segments to minimize the overhead remains open. It may well be the case that such an integer pattern is not periodic at each level and uses different-length segments. However, the good news is that the rounding of the rational solution provided in this paper seems quite efficient in practice.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments, which have greatly improved the quality of this paper. This research was funded in part by the European project SCoRPiO, by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR), by the PIA ELCI project, and by the ANR RESCUE project. Yves Robert is with Institut Universitaire de France.

REFERENCES

[1] P. Balaprakash, L. A. B. Gomez, M.-S. Bouguerra, S. M. Wild, F. Cappello, and P. D. Hovland. Analysis of the tradeoffs between energy and run time for multilevel checkpointing. In *Proc. PMBS'14*, 2014.

[2] L. Bautista-Gomez, S. Tsuboi, D. Komatitsch, F. Cappello, N. Maruyama, and S. Matsuoka. FTI: High performance fault tolerance interface for hybrid systems. In *Proc. SC'11*, 2011.

[3] A. Benoit, A. Cavelan, Y. Robert, and H. Sun. Optimal resilience patterns to cope with fail-stop and silent errors. Research report RR-8786, INRIA, 2015. Available at graal.ens-lyon.fr/~yrobert/rr8786.pdf. Short version appears in IPDPS'16.

[4] G. Bosilca et al. Unified model for assessing checkpointing protocols at extreme-scale. *Concurrency and Computation: Practice and Experience*, 2013.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

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

[7] S. Di, M. S. Bouguerra, L. Bautista-Gomez, and F. Cappello. Optimization of multi-level checkpoint model for large scale HPC applications. In *Proc. IPDPS'14*, 2014.

[8] S. Di, Y. Robert, F. Vivien, and F. Cappello. Toward an optimal online checkpoint solution under a two-level HPC checkpoint model. *IEEE Trans. Parallel & Distributed Systems*, 2016, preprint available on the IEEE digital library.

[9] K. Ferreira, J. Stearley, J. H. I. Laros, R. Oldfield, K. Pedretti, R. Brightwell, R. Riesen, P. G. Bridges, and D. Arnold. Evaluating the Viability of Process Replication Reliability for Exascale Systems. In *Proc. SC'11*, pages 44:1–44:12, 2011.

[10] R. G. Gallager. *Stochastic Processes: Theory for Applications*. Cambridge University Press, New York, NY, USA, 2014.

[11] D. Hakkarinen and Z. Chen. Multilevel diskless checkpointing. *IEEE Transactions on Computers*, 62(4):772–783, 2013.

[12] E. Heien, D. Kondo, A. Gainaru, D. LaPine, B. Kramer, and F. Cappello. Modeling and tolerating heterogeneous failures in large parallel systems. In *Proc. ACM/IEEE Supercomputing'11*. ACM Press, 2011.

[13] T. Hérault and Y. Robert, editors. *Fault-Tolerance Techniques for High-Performance Computing*, Computer Communications and Networks. Springer Verlag, 2015.

[14] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski. Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System. In *Proc. SC'10*, 2010.

[15] J. Plank, K. Li, and M. Puening. Diskless checkpointing. *IEEE Trans. Parallel Dist. Systems*, 9(10):972–986, 1998.

[16] L. Silva and J. Silva. Using two-level stable storage for efficient checkpointing. *IEE Proceedings - Software*, 145(6):198–202, 1998.

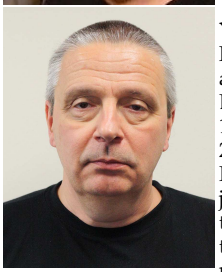
[17] N. H. Vaidya. A case for two-level distributed recovery schemes. *SIGMETRICS Perform. Eval. Rev.*, 23(1):64–73, 1995.

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

Anne Benoit received the PhD degree from Institut National Polytechnique de Grenoble in 2003, and the Habilitation à Diriger des Recherches (HDR) from École Normale Supérieure de Lyon (ENS Lyon) in 2009. She is currently an associate professor in the Computer Science Laboratory LIP at ENS Lyon, France. She is the author of 38 papers published in international journals, and 78 papers published in international conferences. She is the advisor of 7 PhD theses. Her research interests include algorithm design and scheduling techniques for parallel and distributed platforms, and also the performance evaluation of parallel systems and applications, with a focus on energy awareness and resilience. She is Associate Editor of IEEE TPDS, JPDC, and SUSCOM. She is the program chair of several workshops and conferences, in particular she is program chair for HiPC'2016, program co-chair for ICPP'2017, and technical papers chair for SC'2017. She is a senior member of the IEEE, and she has been elected a Junior Member of Institut Universitaire de France in 2009.

Aurélien Cavelan completed his master at the University of Orléans in 2014, and then moved to École Normale Supérieure de Lyon (ENS Lyon), where he is currently a PhD candidate under the supervision of Anne Benoit and Yves Robert. As part of completing his PhD, he is also a visiting student at Argonne National Laboratory, where he is working with Franck Cappello on optimizing the execution of large application workflows subject to faults. His main topics of interest include fault tolerance, energy efficiency and scheduling techniques for large scale platforms.

Valentin Le Fèvre received the M.S. degree in fundamental computer science from École Normale Supérieure de Lyon (ENS Lyon, France) in 2016, where he also received his B.S. degree in 2014. He is mainly interested in high-performance computing, in particular resilience and scheduling problems.



Yves Robert received the PhD degree from Institut National Polytechnique de Grenoble. He is currently a full professor in the Computer Science Laboratory LIP at ENS Lyon. He is the author of 7 books, 147 papers published in international journals, and 219 papers published in international conferences. He is the editor of 11 book proceedings and 13 journal special issues. He is the advisor of 30 PhD theses. His main research interests are scheduling techniques and resilient algorithms for large-scale platforms. Yves Robert served on many editorial boards, including IEEE TPDS and JPDC. He was the program chair of HiPC'2006 in Bangalore, IPDPS'2008 in Miami, ISPDC'2009 in Lisbon, ICPP'2013 in Lyon and HiPC'2013 in Bangalore. He is a Fellow of the IEEE. He has been elected a Senior Member of Institut Universitaire de France in 2007 and renewed in 2012. He has been awarded the 2014 IEEE TCSC Award for Excellence in Scalable Computing, and the 2016 IEEE TCPP Outstanding Service Award. He holds a Visiting Scientist position at the University of Tennessee Knoxville since 2011. Contact him at Yves.Robert@inria.fr.

Hongyang Sun received his Ph.D. in Computer Science from Nanyang Technological University, Singapore in 2011. Prior to that, he completed his B.Eng. in Computer Engineering from the same university, and his M.Sc. in Computer Science from the National University of Singapore under the Singapore-MIT Alliance program. He has held research position at the Toulouse Institute Computer Science Research, France. Currently, he is a Postdoc Researcher at ENS de Lyon and INRIA, France. His main research interests include high-performance computing, scheduling and resource management, energy efficiency, design and analysis of algorithms, and fault tolerance. He has published over 30 peer-reviewed papers in international journals and conferences, and has served in the program committee of SBAC-PAD'15, IPDPS'16.