Coping with silent and fail-stop errors at scale by combining replication and checkpointing

Anne Benoit[a], Aurélien Cavelan[b], Franck Cappello[c], Padma Raghavan[d], Yves Robert[a,e], Hongyang Sun[d,*]

[a] *Ecole Normale Superieure de Lyon & INRIA, France*
[b] *University of Basel, Switzerland*
[c] *Argonne National Laboratory, USA*
[d] *Vanderbilt University, USA*
[e] *University of Tennessee Knoxville, USA*

Abstract

This paper provides a model and an analytical study of replication as a technique to cope with silent errors, as well as a mixture of both silent and fail-stop errors on large-scale platforms. Compared with fail-stop errors that are immediately detected when they occur, silent errors require a detection mechanism. To detect silent errors, many application-specific techniques are available, either based on algorithms (e.g., ABFT), invariant preservation or data analytics, but replication remains the most transparent and least intrusive technique. We explore the right level (duplication, triplication or more) of replication for two frameworks: (i) when the platform is subject to only silent errors, and (ii) when the platform is subject to both silent and fail-stop errors. A higher level of replication is more expensive in terms of resource usage but enables to tolerate more errors and to even correct some errors, hence there is a trade-off to be found. Replication is combined with checkpointing and comes with two flavors: *process replication* and *group replication*. Process replication applies to message-passing applications with communicating processes. Each process is replicated, and the platform is composed of process pairs, or triplets. Group replication applies to black-box applications, whose parallel execution is replicated several times. The platform is partitioned into two halves (or three thirds). In both scenarios, results are compared before each checkpoint, which is taken only when both results (duplication) or two out of three results (triplication) coincide. Otherwise, one or more silent errors have been detected, and the application rolls back to the last checkpoint, as well as when fail-stop errors have struck. We provide a detailed analytical study for all of these scenarios, with formulas to decide, for each scenario, the optimal parameters as a function of the error rate, checkpoint cost, and platform size. We also report a set of extensive simulation results that nicely corroborates the analytical model.

1. Introduction

Triple Modular Redundancy, or TMR [38], is the standard fault-tolerance approach for critical systems, such as embedded or aeronautical devices [1]. With TMR, computations are executed three times, and a majority voting is conducted to select the correct result out of the three available ones. Indeed, if two or more results agree, they are declared correct, because the probability of two or more errors leading to the same wrong result is assumed so low that it can be ignored. While triplication seems very expensive in terms or resources, anybody sitting in a plane would heartily agree that it is worth the price.

On the contrary, duplication, let alone triplication, has a bad reputation in the High Performance Computing (HPC) community: Who would be ready to waste half or two-thirds of the

☆A preliminary version [6] of this paper has appeared in the Proceedings of the ACM Workshop on Fault-Tolerance for HPC at Extreme Scale, June 2017.
*Corresponding author

precious computing resources? However, despite its high cost, several authors have been advocating the use of duplication in HPC in the recent years [45, 53, 28, 30]. In a nutshell, this is because platform sizes have become so large that fail-stop errors are likely to strike at a high rate during application execution. More precisely, the MTBF (Mean Time Between Failures) μ_P of the platform decreases linearly with the number of processors P, since $\mu_P = \frac{\mu_{\text{ind}}}{P}$, where μ_{ind} is the MTBF of each individual component (see Proposition 1.2 in [35]). Take $\mu_{\text{ind}} = 10$ years as an example. If $P = 10^5$ then $\mu_P \approx 50$ minutes and if $P = 10^6$ then $\mu_P \approx 5$ minutes: from the resilience's point of view, scale is the enemy. Given any value of μ_{ind}, there is a threshold for the number of processors above which the platform throughput will decrease [27, 42, 45, 30]: the platform MTBF becomes so small that the applications experience too many failures, hence too many recoveries and re-execution delays, to progress efficiently. All this explains why duplication has been considered for HPC applications despite its cost. The authors in [30] propose *process replication* by which each process in a parallel MPI (Message Passing Interface) application is duplicated on multiple physical processors while maintaining synchronous execution of the replicas. This approach is effective because the MTBF of a set of two replicas (which is the average delay for failures to strike *both* processors in the replica set) is much larger than the MTBF of a single processor.

Process replication may not always be a feasible option, since its features must be provided by the application. Some prototype MPI implementations [30, 31] are convincing proofs of concept and do provide such capabilities. However, many other programming frameworks (not only MPI-like frameworks, but also concurrent objects, distributed components, workflows, algorithmic skeletons) do not provide an equivalent to transparent process replication for the purpose of fault-tolerance, and enhancing them with transparent replication may be non-trivial. When transparent replication is not (yet) provided by the runtime system, one solution could be to implement it explicitly within the application, but this is a labor-intensive process especially for legacy applications. Another approach introduced in [17] is *group replication*, a technique that can be used whenever process replication is not available. Group replication is agnostic to the parallel programming model, and thus views the application as an unmodified black box. The only requirement is that the application be startable from a saved checkpoint file. Group replication consists in executing multiple application instances concurrently. For instance, two distinct P-process application instances could be executed on a $2P$-processor platform. At first glance, it may seem paradoxical that better performance can be achieved by using group duplication. After all, in the above example, 50% of the platform is "wasted" to perform redundant computation. The key point here is that each application instance runs at a smaller scale. As a result each instance can use lower checkpointing frequency, and can thus have better parallel efficiency in the presence of faults, when compared to a single application instance running at full scale. In some cases, the application makespan can then be comparable to, or even shorter than that obtained when running a single application instance. In the end, the cost of wasting processor power for redundant computation can be offset by the benefit of reduced checkpointing frequency. Furthermore, in group replication, once an instance saves a checkpoint, the other instance can use this checkpoint immediately to "jump ahead" in its execution. Hence, group replication is more efficient than the mere independent execution of several instances: each time one instance successfully completes a given "chunk of work", all the other instances immediately benefit from this success. To implement group replication, the runtime system needs to perform the typical operations needed for system-assisted checkpoint/restart: determining checkpointing frequencies for each application instance, causing checkpoints to be saved, detecting application failures, and restarting an application instance from a saved checkpoint after a failure. The only additional feature is that the system must be able to stop an instance and let it resume execution from a checkpoint file produced by another instance as soon as it is produced.

Process or group replication has been mainly proposed in HPC to cope with fail-stop errors. However, another challenge is represented by silent errors, or silent data corruption, whose threat can no longer be ignored [41, 54, 39]. There are several causes of silent errors, such as cosmic radiation, packaging pollution, among others. Silent errors can strike the cache and memory (bit flips) as well as CPU operations; in the latter case they resemble floating-point errors due

to improper rounding, but have a dramatically larger impact because any bit of the result, not only the low-order mantissa bits, can be corrupted. In contrast to a fail-stop error whose detection is immediate, a silent error is identified only when the corrupted data leads to an unusual application behavior. Such detection latency raises a new challenge: if the error struck before the last checkpoint, and is detected after that checkpoint, then the checkpoint is corrupted and cannot be used for rollback. To address the problem of detection latency in silent errors, many application-specific detectors, or verification mechanisms, have been proposed (see Section 2 for a survey). It is not clear, however, whether a special-purpose detector can be designed for each scientific application. In addition, application-specific verification mechanisms can only protect an application from certain types of errors, and fail to provide accurate and efficient detection of all silent errors. In fact, providing such detectors for scientific applications has been identified as one of the hardest challenges[1] towards extreme-scale computing [15, 16].

Altogether, silent errors call for revisiting replication in the framework of scientific applications executing on large-scale HPC platforms. Because replication is now applied at the process level, scale becomes an even harder-to-fight enemy. Processor count ranges to about 10^5 on the K-computer and TaihuLight systems. The number of processors could increase further to 10^6 (hence 10^6 or more processes) on Exascale systems, with billions of threads [24]. In addition, the probability of several errors striking during an execution can get significant, depending upon whether or not circuit manufacturers increase significantly the protection of the logic, latch/flip-flops and static arrays in the processors. In a recent paper [47], the authors consider that with significant more protection (more hardware, more power consumption), the FIT[2] rate for undetected errors on a processor circuit could be maintained to around 20. But without additional protection compared to the current situation, the FIT rate for undetected errors could be as high as 5,000 (or 1 error every 200,000 hours). Combining 10 millions of devices with this FIT rate would result in a silent error in the system every 72 seconds. This work aims at providing a quantitative assessment of the potential of duplication and triplication to mitigate such a threat.

The main contributions of this work are summarized as follows:

- We develop an analytical model to study the performance of all replication scenarios to cope with silent errors, namely, duplication, triplication, and more for both process and group replications;
- We derive closed-form formulas for the optimal checkpointing period, the optimal process count, as well as the expected application speedup as a function of error rate, checkpoint cost, and platform size;
- We develop an extended model and derive the corresponding closed-form formulas when the platform is subject to both fail-stop and silent errors;
- We conduct a comprehensive set of simulations whose results corroborate the analytical studies.

The rest of the paper is organized as follows. We first survey the related work in Section 2. We then focus on silent errors only and introduce the performance model in Section 3. We derive the general expected execution time in Section 4. The analysis for process replication is presented in Section 5, followed by the analysis for group replication in Section 6. Section 7 extends the previous results to the more general framework where the platform is confronted with both fail-stop and silent errors. Section 8 is devoted to the simulation results. Finally, we provide concluding remarks and directions for future work in Section 9.

[1] More generally, trustworthy computing, which aims at guaranteeing the correctness of the results of a long-lasting computation on a large-scale supercomputer, has received considerable attention recently [14].

[2] The Failures in Time (FIT) rate of a device is the number of failures that can be expected in one billion (10^9) device-hours of operation.

2. Related work

We survey the related work in this section. We start with replication for HPC applications in Section 2.1 and cover application-specific silent-error detectors in Section 2.2.

2.1. Replication for fail-stop errors

Checkpointing policies have been widely studied. We refer to [35] for a survey of various protocols and the derivation of the Young's and Daly's formula [51, 22] for the optimal checkpointing periods. Recent advances include multi-level approaches [39, 23, 7], or the use of SSD or NVRAM as secondary storage [16]. Combining replication with checkpointing has been proposed in [45, 53, 28] for HPC platforms, and in [36, 50] for grid computing.

The use of redundant MPI processes is analyzed in [29, 30, 18]. In particular, the work by Ferreira et al. [30] has studied the use of process replication for MPI applications, using 2 replicas per MPI process. They provide a theoretical analysis of parallel efficiency, an MPI implementation that supports transparent process replication (including failure detection, consistent message ordering among replicas, etc.), and a set of experimental and simulation results. Partial redundancy is studied in [26, 48, 49] (in combination with coordinated checkpointing) to decrease the overhead of full replication. Adaptive redundancy is introduced in [32], where a subset of processes is dynamically selected for replication. Thread-level replication has been investigated in [52, 21, 43], which target process-level replication in order to detect (and correct) silent errors striking in all communication-related operations. Finally, Ni et al [40] introduce process duplication to cope with both fail-stop and silent errors. Their pioneering work contains many interesting results but differs from this work as follows: (i) they limit themselves to perfectly parallel applications while we investigate speedup profiles that obey Amdahl's law; (ii) they do not investigate triplication; and (iii) they compute an upper bound on the optimal period and do not determine optimal processor counts.

2.2. Silent error detection and correction

Application-specific information enables ad-hoc solutions, which dramatically decrease the cost of error detection. Algorithm-based fault tolerance (ABFT) [34, 12, 46] is a well-known technique, which uses checksums to detect up to a certain number of errors in linear algebra kernels. Unfortunately, ABFT can only protect datasets in linear algebra kernels, and it must be implemented for each different kernel, which incurs a large amount of work for large HPC applications. Other techniques have also been advocated. Benson, Schmit and Schreiber [10] compare the result of a higher-order scheme with that of a lower-order one to detect errors in the numerical analysis of ODEs and PDEs. Sao and Vuduc [44] investigate self-stabilizing corrections after error detection in the conjugate gradient method. Bridges et al. [33] propose linear solvers to tolerant soft faults using selective reliability. Elliot et al. [25] design a fault-tolerant GMRES capable of converging despite silent errors. Bronevetsky and de Supinski [13] provide a comparative study of detection costs for iterative methods.

Recently, several silent error detectors based on data analytics have been proposed, showing promising results. These detectors use several interpolation techniques such as time series prediction [11] and spatial multivariate interpolation [3, 4, 5]. Such techniques offer large detection coverage for a negligible overhead. However, these detectors do not guarantee full coverage; they can detect only a certain percentage of corruptions (i.e., partial verification with an imperfect recall). Nonetheless, the accuracy-to-cost ratios of these detectors are high, which makes them interesting alternatives at large scale. Similar detectors have also been designed to detect silent errors in the temperature data of the Orbital Thermal Imaging Spectrometer (OTIS) [20].

Again, all the papers quoted in this section provide application-specific detectors, while our approach is agnostic of the application characteristics. The only information required is whether we can use process replication. If not, we can view the application as a black box and use group replication.

Table 1: List of Notations.

Parameters	
T	Length (or period) of a pattern
P	Number of processes allocated to an application
n	Number of (process or group) replicas
$S(P)$	Speedup function of an application
$H(P)$	Error-free execution overhead
$\mathbb{E}_n(T,P)$	Expected execution time of a pattern
$\mathbb{H}_n(T,P)$	Expected execution overhead of a pattern
$\mathbb{S}_n(T,P)$	Expected speedup function of a pattern
$\lambda = \frac{1}{\mu_{\text{ind}}}$	Silent error rate of an individual process
$\mathbb{P}_n(T,P)$	Silent error probability of a pattern
C	Checkpointing cost
R	Recovery cost
V	Verification cost (comparison of replicas)

3. Model

This section presents the analytical model for evaluating the performance of different replication scenarios when the platform is subject to silent errors. Table 1 summarizes the main notations for this framework. The extension to both fail-stop and silent errors will be discussed in Section 7, and additional notations will be introduced then for the extended framework.

The model presented here is a classical one, similar to those of the literature for replication [30], only with a different objective (quantifying replication for silent errors). Recall that μ_{ind} denotes the MTBE of an individual processor or process[3] of the system, and let $\lambda = \frac{1}{\mu_{\text{ind}}}$ denote the silent error rate of the processor. The error rate for a collection of P processors is then given by $\lambda_P = \frac{1}{\mu_P} = \frac{P}{\mu_{\text{ind}}} = \lambda P$ [35]. Assuming that the error arrivals follow *Exponential* distribution, the probability that a computation is hit by a silent error during time T on P processes is given by $\mathbb{P}(T,P) = 1 - e^{-\lambda P T}$.

Consider long-lasting HPC applications that execute for hours or even days on a large-scale platform. Resilience is enforced by the combined use of replication and periodic checkpointing. Before each checkpoint, the results of different replicas are compared. Only when both results (for duplication) or two out of three results (for triplication) coincide[4], in which case a *consensus* is said to be reached, the checkpoint is taken. Otherwise, silent errors are assumed to have been detected, and they cannot be corrected through consensus. The application then rolls back to the last checkpoint. There are two different types of replications:

- *Process replication*: Each process of the application is replicated, and the results of different processes are independently compared. A rollback is needed when at least one process fails to reach a consensus;
- *Group replication*: The entire application (as a black box) is replicated, and the results of all replicas (as a whole) are compared. A rollback is needed when these group replicas fail to reach a consensus.

The computational chunk between two checkpoints is called a *periodic pattern*. For a replication scenario with n replicas, the objective is to minimize the expected total execution time (or makespan) of an application by finding the optimal pattern parameters:

- T: length (or period) of the pattern;

[3]We assume that each process is executed by a dedicated processor. Hence, we will use "processor" and "process" interchangeably. We also use MTBE instead of MTBF to emphasize that we deal with (silent) errors, not failures.

[4]For $n > 3$ replicas, the results of k replicas should coincide, where $2 \leq k < n$ is a design parameter set by the system to control the level of reliability. $k = \lfloor \frac{n}{2} \rfloor + 1$ is a widely-used choice (majority voting).

- P: number of processes allocated to the application.

Indeed, for long-lasting applications, it suffices to focus on a single pattern, since the pattern repeats itself over time. To see this, let W_{total} denote the total amount of work of the application and suppose the application has an error-free speedup function $S(P)$ when executed on P processors. In this paper, we focus on a speedup function that obeys Amdahl's law[5]:

$$S(P) = \frac{1}{\alpha + \frac{1-\alpha}{P}} \ , \tag{1}$$

where $\alpha \in [0, 1]$ denotes the sequential fraction of the application that cannot be parallelized. For convenience, we also define $H(P) = \frac{1}{S(P)}$ to be the execution overhead. For a pattern of length T and run by P processes, the amount of work done in a pattern is therefore $W_{\text{pattern}} = T \cdot S(P)$, and the total number of patterns in the application can be approximated as $m = \frac{W_{\text{total}}}{W_{\text{pattern}}} = \frac{W_{\text{total}}}{T \cdot S(P)} = \frac{W_{\text{total}}}{T} H(P)$. Now, let $\mathbb{E}_n(T, P)$ denote the expected execution time of the pattern with n replicas in either replication scenario. Define $\mathbb{H}_n(T, P) = \frac{\mathbb{E}_n(T,P)}{T} H(P)$ to be the expected execution overhead of the pattern, and $\mathbb{S}_n(T, P) = \frac{1}{\mathbb{H}_n(T,P)}$ the expected speedup. The expected makespan of the application can then be written as $\mathbb{E}_{\text{total}} \approx \mathbb{E}_n(T, P)m = \mathbb{E}_n(T, P)\frac{W_{\text{total}}}{T} H(P) = \mathbb{H}_n(T, P) \cdot W_{\text{total}} = \frac{W_{\text{total}}}{\mathbb{S}_n(T,P)}$. Since W_{total} is fixed, this shows that the optimal expected makespan can be achieved by minimizing the expected execution overhead of a pattern, or equivalently, maximizing its expected speedup.

Now, we describe a model for the costs of checkpoint, recovery and consensus verification. First, the checkpoint cost clearly depends on the protocol and storage type. Note that only the result of one replica needs to be checkpointed, so the cost does not increase with the number of replicas. To save the application's memory footprint M to the storage system using P processes, we envision the following two scenarios:

- $C = O(\frac{M}{\tau_{io}})$: In this case, checkpoints are being written to the remote storage system, whose bandwidth is the I/O bottleneck. Here, τ_{io} is the remote I/O bandwidth.
- $C = O(\frac{M}{\tau_{net} P})$: This case corresponds to in-memory checkpoints, where each process stores $\frac{M}{P}$ data locally (e.g., on SSDs). Here, τ_{net} is the process network bandwidth.

The recovery cost is assumed to be the same as the checkpointing cost, i.e., $R = C$, as it involves the same I/O operations. This is a common assumption [39], although practical recovery cost can be somewhat smaller than the checkpoint cost [23]. Finally, verifying consensus is performed by communicating and comparing $\frac{M}{P}$ data stored on each process, which can be executed concurrently by all process pairs (or triplets). Hence, the verification cost satisfies $V = O(\frac{M}{P})$. Overall, we use the following general expression to account for the combined cost of verification and checkpoint/recovery:

$$V + C = c + \frac{d}{P} \ , \tag{2}$$

where c and d are constants that depend on the application memory footprint, checkpointing protocol, network or I/O bandwidth, etc. Here, $c = 0$ corresponds to the in-memory checkpointing scenario discussed above. Finally, we assume that checkpoints, recoveries, and consensus verifications are all protected from silent errors[6].

[5]The model is generally applicable to other speedup functions as well.

[6]Disk operations (remote checkpoint/recovery) are typically protected by RAID [19] or I/O integrity protection mechanisms [37], and in-memory operations (local checkpoint and verification) can be performed on protected memory space or using redundancy. In both cases, the costs of protecting these operations can be well captured by the constants in the general expression of Equation (2).

4. Expected execution time

In this section, we compute the expected execution time of a periodic pattern, which will be used in the next two sections to derive the optimal pattern parameters.

Theorem 1. *The expected time to execute a periodic pattern of length T using P processes and n replicas can be expressed as*

$$\mathbb{E}_n(T,P) = T + V + C + \frac{\mathbb{P}_n(T,P)}{1-\mathbb{P}_n(T,P)}\left(T + V + R\right), \tag{3}$$

where $\mathbb{P}_n(T,P)$ denotes the probability that the execution fails due to silent errors striking during the pattern, thereby forcing to roll back to the last checkpoint.

Proof. Since replicas are synchronized, we can express the expected execution time as follows:

$$\mathbb{E}_n(T,P) = T + V + \mathbb{P}_n(T,P)\cdot\big(R + \mathbb{E}_n(T,P)\big) + \big(1-\mathbb{P}_n(T,P)\big)\cdot C\ . \tag{4}$$

First, the pattern of length T is executed followed by the verification (through comparison and/or voting), which incurs cost V. With probability $\mathbb{P}_n(T,P)$, the pattern fails due to silent errors. In this case, we need to re-execute the pattern after performing a recovery from the last checkpoint with cost R. Otherwise, with probability $1-\mathbb{P}_n(T,P)$, the execution succeeds and the checkpoint with cost C is taken at the end of the pattern.

Now, solving for $\mathbb{E}_n(T,P)$ from Equation (4), we can obtain the expected execution time of the pattern as shown in Equation (3). □

Remarks. Theorem 1 is applicable to both process replication and group replication. The only difference lies in the computation of failure probability $\mathbb{P}_n(T,P)$, which depends on the replication scenario and on the number of replicas n.

5. Process replication

In this section, we consider process replication. We first derive the optimal computing patterns when each process of the application is duplicated (Section 5.1) and triplicated (Section 5.2), respectively. Finally, we generalize the results to an arbitrary but constant number of replications per process under a general process replication framework (Section 5.3).

5.1. Process duplication

We start with process duplication, that is, each process has two replicas. The following lemma shows the failure probability of a given computing pattern in this case.

Lemma 1. *Using process duplication, the failure probability of a computing pattern of length T and with P processes is given by*

$$\mathbb{P}_2^{\text{prc}}(T,P) = 1 - e^{-2\lambda TP}\ . \tag{5}$$

Proof. With duplication, errors cannot be corrected (no consensus), hence a process fails if either one of its replicas fails or both replicas fail. In other words, there is an error if the results of both replicas do not coincide (we neglect the quite unlikely scenario with one error in each replica leading to the same wrong result). Let $\mathbb{P}_1^{\text{prc}}(T,1) = 1 - e^{-\lambda T}$ denote the probability of a single process failure. Therefore, we can write the failure probability of any duplicated process as follows:

$$\begin{aligned}\mathbb{P}_2^{\text{prc}}(T,1) &= \binom{2}{2}\mathbb{P}_1^{\text{prc}}(T,1)^2 + \binom{2}{1}\left(1-\mathbb{P}_1^{\text{prc}}(T,1)\right)\mathbb{P}_1^{\text{prc}}(T,1)\\ &= \left(1-e^{-\lambda T}\right)^2 + 2e^{-\lambda T}\left(1-e^{-\lambda T}\right)\\ &= 1 - e^{-2\lambda T}\ .\end{aligned}$$

Now, because we have P independent processes, the probability that the application gets interrupted by silent errors is the probability that at least one process fails because of silent errors, which can be expressed as:

$$\begin{aligned}\mathbb{P}_2^{\text{prc}}(T,P) &= 1 - \mathbb{P}(\text{“No process fails”}) \\ &= 1 - \left(1 - \mathbb{P}_2^{\text{prc}}(T,1)\right)^P \\ &= 1 - e^{-2\lambda PT} \ .\end{aligned}$$

□

Using the failure probability in Lemma 1, we derive the optimal computing pattern for process duplication as shown in the following theorem. Recall that the application speedup follows Amdahl's law as shown in Equation (1) and the cost of verification and checkpoint is modeled by Equation (2).

Theorem 2. *A first-order approximation to the optimal number of processes for an application with 2 replicas per process is given by*

$$P_{\text{opt}} = \min\left\{\frac{Q}{2}, \left(\frac{1}{2}\left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{\lambda c}\right)^{\frac{1}{3}}\right\} \ , \tag{6}$$

where Q denotes the total number of available processes in the system. The associated optimal checkpointing period and the expected speedup function of the application are

$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{V+C}{2\lambda P_{\text{opt}}}\right)^{\frac{1}{2}} \ , \tag{7}$$

$$\mathbb{S}_2^{\text{prc}}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1 + 2\big(2\lambda(V+C)P_{\text{opt}}\big)^{\frac{1}{2}}} \ . \tag{8}$$

Proof. First, we can derive, from Theorem 1 and Lemma 1, the expected execution time of a pattern with length T and P duplicated processes as follows:

$$\begin{aligned}\mathbb{E}_2^{\text{prc}}(T,P) &= T + V + C + \left(e^{2\lambda PT} - 1\right)(T + V + R) \\ &= T + V + C + 2\lambda PT\,(T + V + R) + o(\lambda PT^2) \ .\end{aligned}$$

The second equation above is obtained by applying Taylor series to approximate $e^z = 1+z+o(z)$ for $z < 1$, while assuming $\lambda PT = \Theta(\lambda^\epsilon)$, where $\epsilon > 0$. Now, substituting $\mathbb{E}_2^{\text{prc}}(T,P)$ into $\mathbb{H}_2^{\text{prc}}(T,P) = H(P)\frac{\mathbb{E}_2^{\text{prc}}(T,P)}{T}$, we can get the expected execution overhead as:

$$\mathbb{H}_2^{\text{prc}}(T,P) = H(P)\left(1 + \frac{V+C}{T} + 2\lambda PT + o(\lambda PT)\right) \ . \tag{9}$$

The optimal overhead can then be achieved by balancing the two terms $\frac{V+C}{T}$ and $2\lambda PT$ above, which gives the following optimal checkpointing period as a function of the process count:

$$T_{\text{opt}}(P) = \left(\frac{V+C}{2\lambda P}\right)^{\frac{1}{2}} \ . \tag{10}$$

Now, substituting $T_{\text{opt}}(P)$ back into Equation (9), we get the execution overhead as a function of the process count as follows (lower-order terms ignored):

$$\mathbb{H}_2^{\text{prc}}(P) = H(P)\left(1 + 2\big(2\lambda(V+C)P\big)^{\frac{1}{2}}\right) \ . \tag{11}$$

Note that Equations (10) and (11) hold true regardless of the form of the function $H(P)$ or the cost $V + C$. Recall that we consider Amhdal's law $H(P) = \alpha + \frac{1-\alpha}{P}$ and a cost model $V + C = c + \frac{d}{P}$. In order to derive the optimal process count, we consider two cases:

Case (1). $c > 0$ and $\alpha > 0$ are both constants: we can expand Equation (11) to be

$$\mathbb{H}_2^{\text{prc}}(P) = \alpha + 2\alpha\big(2\lambda c P\big)^{\frac{1}{2}} + \frac{1-\alpha}{P} + o(\lambda^{\frac{1}{2}}) \ . \tag{12}$$

The optimal overhead can then be achieved by setting

$$\frac{\partial \mathbb{H}_2^{\text{prc}}(P)}{\partial P} = \alpha \left(\frac{2\lambda c}{P}\right)^{\frac{1}{2}} - \frac{1-\alpha}{P^2} = 0 \ ,$$

which leads to $P^* = \left(\frac{1}{2}\left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{\lambda c}\right)^{\frac{1}{3}}$. Since the total number of processes in the system is Q and each application process is duplicated, the optimal process count is upper-bounded by $\frac{Q}{2}$ if $P^* > \frac{Q}{2}$, due to the convexity of $\mathbb{H}_2^{\text{prc}}(P)$ as shown in Equation (11). Hence, the optimal process count P_{opt} is given by Equation (6).

Case (2). $c = 0$ or $\alpha = 0$: In either case, we can see that Equation (11) becomes a decreasing function of P. Therefore, the optimal strategy is to utilize all the available Q processes, i.e., $P_{\text{opt}} = \frac{Q}{2}$, which again satisfies Equation (6), since $\left(\frac{1}{2}\left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{\lambda c}\right)^{\frac{1}{3}} = \infty$.

In either case, the expected application speedup is then given by the reciprocal of the overhead as shown in Equation (11) with the optimal process count P_{opt}. □

Remarks. For fully parallelizable applications, i.e., $\alpha = 0$, the optimal pattern on a Q-process platform is characterized by

$$P_{\text{opt}} = \frac{Q}{2} \ , \quad T_{\text{opt}} = \begin{cases} \sqrt{\frac{c}{\lambda Q}} & \text{for } V + C = c \\ \frac{1}{Q}\sqrt{\frac{2d}{\lambda}} & \text{for } V + C = \frac{d}{P} \end{cases} \ ,$$

$$\mathbb{S}_2^{\text{prc}}(P_{\text{opt}}) = \begin{cases} \frac{Q}{2(1+2\sqrt{\lambda c Q})} & \text{for } V + C = c \\ \frac{Q}{2(1+2\sqrt{2\lambda d})} & \text{for } V + C = \frac{d}{P} \end{cases} \ .$$

5.2. Process triplication

Now, we consider process triplication, that is, each process has three replicas. This is the smallest number of replicas that allows an application to recover from silent errors through majority voting instead of rolling back to the last checkpoint.

Lemma 2. *Using process triplication, the failure probability of a computing pattern of length T and with P processes is given by*

$$\mathbb{P}_3^{\text{prc}}(T, P) = 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P \ . \tag{13}$$

Proof. Using triplication, if only one replica fails, the silent error can be masked by the two successful replicas. Hence, in this case, a process fails if at least two of its replicas are hit by silent errors. Let $\mathbb{P}_1^{\text{prc}}(T, 1) = 1 - e^{-\lambda T}$ denote the probability of a single process failure. Therefore, we can write the failure probability of any triplicated process as follows:

$$\begin{aligned} \mathbb{P}_3^{\text{prc}}(T, 1) &= \binom{3}{3} \mathbb{P}_1^{\text{prc}}(T, 1)^3 + \binom{3}{2} \left(1 - \mathbb{P}_1^{\text{prc}}(T, 1)\right) \mathbb{P}_1^{\text{prc}}(T, 1)^2 \\ &= \left(1 - e^{-\lambda T}\right)^3 + 3e^{-\lambda T}\left(1 - e^{-\lambda T}\right)^2 \\ &= 1 - 3e^{-2\lambda T} + 2e^{-3\lambda T} \ . \end{aligned}$$

For P independent processes, the application fails when at least one of its processes fails. Hence, we have:

$$\begin{aligned} \mathbb{P}_3^{\text{prc}}(T, P) &= 1 - \mathbb{P}(\text{“No process fails”}) \\ &= 1 - \left(1 - \mathbb{P}_3^{\text{prc}}(T, 1)\right)^P \\ &= 1 - \left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P \ . \end{aligned}$$

□

The following theorem derives the optimal computing pattern for process triplication. The proof can be found in Appendix A.

Theorem 3. *A first-order approximation to the optimal number of processes for an application with* 3 *replicas per process is given by*

$$P_{\text{opt}} = \min\left\{\frac{Q}{3}, \left(\frac{4}{3}\left(\frac{1-\alpha}{\alpha}\right)^3\left(\frac{1}{\lambda c}\right)^2\right)^{\frac{1}{4}}\right\} , \tag{14}$$

where Q denotes the total number of available processes in the system. The associated optimal checkpointing period and the expected speedup function of the application are

$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{V+C}{6\lambda^2 P_{\text{opt}}}\right)^{\frac{1}{3}} , \tag{15}$$

$$\mathbb{S}_3^{\text{prc}}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1+3\left(\frac{3}{4}\left(\lambda(V+C)\right)^2 P_{\text{opt}}\right)^{\frac{1}{3}}} . \tag{16}$$

Remarks. For fully parallelizable applications, i.e., $\alpha = 0$, the optimal pattern on a Q-process platform is characterized by

$$P_{\text{opt}} = \frac{Q}{3} , \quad T_{\text{opt}} = \begin{cases} \sqrt[3]{\frac{c}{2\lambda^2 Q}} & \text{for } V+C=c \\ \sqrt[3]{\frac{3d}{2\lambda^2 Q^2}} & \text{for } V+C=\frac{d}{P} \end{cases} ,$$

$$\mathbb{S}_2^{\text{prc}}(P_{\text{opt}}) = \begin{cases} \frac{Q}{3\left(1+3\sqrt[3]{\left(\frac{\lambda c}{2}\right)^2 Q}\right)} & \text{for } V+C=c \\ \frac{Q}{3\left(1+3\sqrt[3]{\left(\frac{3\lambda c}{2}\right)^2 \frac{1}{Q}}\right)} & \text{for } V+C=\frac{d}{P} \end{cases} .$$

Compared with duplication, the ability to correct errors in triplication allows checkpoints to be taken less frequently (i.e., larger checkpointing period). In terms of the expected speedup, triplication suffers from a smaller error-free speedup ($\frac{Q}{3}$ vs $\frac{Q}{2}$) due to the use of fewer concurrent processes to perform useful work, but also has a smaller error-induced denominator, especially on platforms with a large number of processes Q. In Section 8, we will conduct simulations to evaluate this trade-off and compare the performance of duplication and triplication.

5.3. General process replication

In this section, we consider a general resilience framework and derive the optimal pattern using n replicas per process, where n is an arbitrary constant. Moreover, let k denote the number of "good" replicas (not hit by silent errors) that is required to reach a consensus through voting. Optimistically, assuming any two replicas that are hit by silent errors will produce different results, we can set $k = 2$, i.e., at least two replicas should agree on the result to avoid a rollback. Under a more pessimistic assumption, we will need a majority of the n replicas to agree on the result, so in this case we need $k = \lfloor \frac{n}{2} \rfloor + 1$. Our analysis is independent of the choice of k.

As for duplication and triplication, for a given (n, k) pair, we can compute the failure probability of a pattern with length T and P processes as follows:

$$\begin{aligned} \mathbb{P}_{n,k}^{\text{prc}}(T,P) &= 1 - \mathbb{P}(\text{"No process fails"}) \\ &= 1 - \left(1 - \mathbb{P}_{n,k}^{\text{prc}}(T,1)\right)^P , \end{aligned} \tag{17}$$

where

$$\begin{aligned} \mathbb{P}_{n,k}^{\text{prc}}(T,1) &= \sum_{j=0}^{k-1} \binom{n}{j} \left(1 - \mathbb{P}_1^{\text{prc}}(T,1)\right)^j \mathbb{P}_1^{\text{prc}}(T,1)^{n-j} \\ &= \sum_{j=0}^{k-1} \binom{n}{j} e^{-\lambda j T} \left(1 - e^{-\lambda T}\right)^{n-j} \end{aligned} \tag{18}$$

denotes the failure probability of a single process with n replicas due to less than k of them surviving silent errors.

The following theorem shows the general result for (n, k)-process replication. The proof follows closely those of Theorems 2 and 3, and is therefore omitted here.

Theorem 4. *On a system with a total number of Q available processors, a first-order approximation to the optimal number of processes for an application with n replicas per process (k of which must concur to avoid a rollback) is given by*

$$P_{\text{opt}} = \min \left\{ \frac{Q}{n}, \left(\gamma_{n,k} \left(\frac{1-\alpha}{\alpha} \right)^{n-k+2} \left(\frac{1}{\lambda c} \right)^{n-k+1} \right)^{\frac{1}{n-k+3}} \right\} . \tag{19}$$

The associated optimal checkpointing period and the expected speedup function of the application are

$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{V+C}{\beta_{n,k} \lambda^{n-k+1} P_{\text{opt}}} \right)^{\frac{1}{n-k+2}} , \tag{20}$$

$$\mathbb{S}^{\text{prc}}_{n,k}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1 + (n-k+2) \left(\frac{(\lambda(V+C))^{n-k+1} P_{\text{opt}}}{\gamma_{n,k}} \right)^{\frac{1}{n-k+2}}} . \tag{21}$$

Here, $\beta_{n,k} = \binom{n}{k-1}(n-k+1)$ and $\gamma_{n,k} = \frac{(n-k+1)^{n-k+1}}{\binom{n}{k-1}}$.

Remarks. Theorem 4 encompasses Theorem 2 and Theorem 3 as special cases. We point out that it even holds for the case without replication, i.e., when $n = k = 1$. In this case, Theorem 4 evaluates to

$$T_{\text{opt}}(P) = \sqrt{\frac{V+C}{\lambda P}} ,$$

$$\mathbb{S}^{\text{prc}}_{1}(P) = \frac{S(P)}{1 + 2\sqrt{\lambda(V+C)P}} ,$$

which is consistent with the results obtained in [2, 8, 9], provided that a reliable silent error detector is available. However, as mentioned previously, such a detector is only known in some application-specific domains. For general-purpose computations, replication appears to be the only viable approach to detect/correct silent errors so far.

6. Group replication

In this section, we consider group replication. Recall that, unlike process replication where the results of each process from different replicas are independently compared, group replication compares the outputs of the different groups viewed as independent black-box applications. First, we make the following technical observation, which establishes the relationship between the two replication mechanisms from the resilience point of view.

Observation 1. *Running an application using group replication with n replicas, where each replica has P processes and each process has error rate λ, has the same failure probability as running it using process replication with one process, which has error rate λP and is replicated n times.*

The above observation allows us to compute the failure probability for group replication by transforming the corresponding formulas for process replication while setting $P = 1$ and $\lambda = \lambda P$. The rest of this section shows the results for duplication, triplication, and a general group replication framework. Proofs are similar to those in process replication, and are therefore omitted.

6.1. Group duplication

By applying Observation 1 on Lemma 1, we can get the failure probability for a given pattern under group duplication as follows.

Lemma 3. *Using group duplication, the failure probability of a computing pattern of length T and with P processes is given by*

$$\mathbb{P}_2^{\text{grp}}(T,P) = 1 - e^{-2\lambda TP} \ . \tag{22}$$

This leads us to the following theorem on the optimal pattern.

Theorem 5. *A first-order approximation to the optimal number of processes for an application with 2 replica groups is given by*

$$P_{\text{opt}} = \min\left\{\frac{Q}{2}, \left(\frac{1}{2}\left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{\lambda c}\right)^{\frac{1}{3}}\right\} \ , \tag{23}$$

where Q denotes the total number of available processes in the system. The associated optimal checkpointing period and the expected speedup function of the application are

$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{V+C}{2\lambda P_{\text{opt}}}\right)^{\frac{1}{2}} \ , \tag{24}$$

$$\mathbb{S}_2^{\text{grp}}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1 + 2\big(2\lambda(V+C)P_{\text{opt}}\big)^{\frac{1}{2}}} \ . \tag{25}$$

Remarks. Both the failure probability (Lemma 3) and the optimal pattern (Theorem 5) under group duplication are identical to those of process duplication (Lemma 1 and Theorem 2). Indeed, in both duplication scenarios, a single silent error that strikes any of the running processes will cause the whole application to fail.

6.2. Group triplication

Again, applying Observation 1 on Lemma 2, we can get the failure probability for a given pattern under group triplication.

Lemma 4. *Using group triplication, the failure probability of a computing pattern of length T and with P processes is given by*

$$\mathbb{P}_3^{\text{grp}}(T,P) = 1 - \left(3e^{-2\lambda TP} - 2e^{-3\lambda TP}\right) \ . \tag{26}$$

The following lemma shows that, with the same pattern length and process count, group triplication suffers from a higher failure probability than its process counterpart.

Lemma 5. $\mathbb{P}_3^{\text{grp}}(T,P) \geq \mathbb{P}_3^{\text{prc}}(T,P)$.

Proof. Let us define $x = e^{-\lambda T}$. From Equations (13) and (26), it suffices to show that

$$\left(3x^2 - 2x^3\right)^P \geq 3x^{2P} - 2x^{3P} \ .$$

We prove the above inequality by induction on the number of processes P. The base case holds trivially with $P = 1$. Now, suppose the claim holds for any $P \geq 1$. Then, we have for $P+1$:

$$\begin{aligned}
\left(3x^2 - 2x^3\right)^{P+1} &= \left(3x^2 - 2x^3\right)^P \left(3x^2 - 2x^3\right) \\
&\geq \left(3x^{2P} - 2x^{3P}\right)\left(3x^2 - 2x^3\right) \qquad \text{(By inductive hypothesis)} \\
&= 9x^{2(P+1)} - 6x^{2P+3} - 6x^{3P+2} + 4x^{3(P+1)} \\
&= 6x^{2(P+1)}\left(1 - x - x^P + x^{P+1}\right) + 3x^{2(P+1)} - 2x^{3(P+1)} \\
&= 6x^{2(P+1)}(1-x)(1-x^P) + 3x^{2(P+1)} - 2x^{3(P+1)} \\
&\geq 3x^{2(P+1)} - 2x^{3(P+1)} \ .
\end{aligned}$$

The last inequality is because $6x^{2(P+1)}(1-x)(1-x^P) \geq 0$ since $x = e^{-\lambda T} \in [0,1]$. □

The following theorem shows the optimal pattern under group triplication.

Theorem 6. *A first-order approximation to the optimal number of processes for an application with* 3 *replica groups is given by*

$$P_{\text{opt}} = \min\left\{\frac{Q}{3}, \left(\frac{1}{6}\left(\frac{1-\alpha}{\alpha}\right)^3\left(\frac{1}{\lambda c}\right)^2\right)^{\frac{1}{5}}\right\} , \tag{27}$$

where Q denotes the total number of available processes in the system. The associated optimal checkpointing period and the expected execution overhead are

$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{V+C}{6(\lambda P_{\text{opt}})^2}\right)^{\frac{1}{3}} , \tag{28}$$

$$\mathbb{S}_3^{\text{grp}}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1+3\left(\frac{3}{4}\left(\lambda(V+C)P_{\text{opt}}\right)^2\right)^{\frac{1}{3}}} . \tag{29}$$

Remarks. Compared to the result of process triplication (Theorem 3) and under the same condition (e.g., $\alpha = 0$ so both scenarios allocate the same number of $P_{\text{opt}} = \frac{Q}{3}$ processes to each replica), Theorem 6 shows that group triplication needs to place checkpoints more frequently yet enjoys a smaller execution speedup. Together with Lemma 5, it provides a theoretical explanation to the intuition that group replication cannot recover from some error combinations that its process counterpart is capable to handle, thus making process replication a superior replication mechanism provided that it can be feasibly implemented.

6.3. General group replication

Finally, we consider a general group replication framework and derive the optimal pattern using a constant number of n replica groups, out of which k of them must agree to avoid a rollback. Again, the results work for any choice of k.

Now, applying Observation 1 on Equations (17) and (18), we can compute the failure probability of a pattern with length T and P processes under a (n,k) group replication model:

$$\mathbb{P}_{n,k}^{\text{grp}}(T,P) = \sum_{j=0}^{k-1}\binom{n}{j}\left(e^{-\lambda PT}\right)^j\left(1-e^{-\lambda PT}\right)^{n-j} . \tag{30}$$

The following theorem shows the general result for this case.

Theorem 7. *On a system with a total number of Q available processors, a first-order approximation to the optimal number of processes for an application with n replica groups (k of which must concur to avoid a rollback) is given by*

$$P_{\text{opt}} = \min\left\{\frac{Q}{n}, \left(\frac{1}{\beta_{n,k}}\left(\frac{1-\alpha}{\alpha}\right)^{n-k+2}\left(\frac{1}{\lambda c}\right)^{n-k+1}\right)^{\frac{1}{2n-2k+3}}\right\} . \tag{31}$$

The associated optimal checkpointing period and the expected speedup function of the application are

$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{C+V}{\beta_{n,k}(\lambda P_{\text{opt}})^{n-k+1}}\right)^{\frac{1}{n-k+2}} , \tag{32}$$

$$\mathbb{S}_{n,k}^{\text{grp}}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1+(n-k+2)\left(\frac{1}{\gamma_{n,k}}\left(\lambda(V+C)P_{\text{opt}}\right)^{n-k+1}\right)^{\frac{1}{n-k+2}}} . \tag{33}$$

Here, $\beta_{n,k} = \binom{n}{k-1}(n-k+1)$ and $\gamma_{n,k} = \frac{(n-k+1)^{n-k+1}}{\binom{n}{k-1}}$.

Table 2: Additional Notations.

Parameters	
λ_s	Silent error rate of an individual process
λ_f	Fail-stop error rate of an individual process
Λ	Total error rate of an individual process
$\mathbb{P}_n(T,P)$	Silent error probability of a pattern
$\mathbb{Q}_n(T,P)$	Fail-stop error probability of a pattern
$\mathbb{F}_n(T,P)$	Total error probability of a pattern
$\mathbb{E}_n^{\text{lost}}(T,P)$	Expected time lost after a fail-stop error

7. Replication under both fail-stop and silent errors

In this section, we consider an extended framework where the platform is subject to both fail-stop and silent errors. We aim at assessing the impact of fail-stop errors as an additional error source. In this regard, this section extends the previous results and presents a unified model and optimal algorithmic solutions to this challenge. We first introduce some additional notations and definitions to be used in this section (Section 7.1). As in the previous analysis for silent errors, we then derive the expected execution time of a given pattern (Section 7.2). Finally, we compute the optimal patterns under process duplication and triplication (Section 7.3), followed by the case for group replication (Section 7.4).

7.1. New notations and definitions

Table 2 lists the additional notations (on top of the ones shown in Table 1) to include fail-stop errors. In the presence of both error sources, a periodic pattern can fail due to either error source. First, if fail-stop errors strike one replica (for duplication) or two out of three replicas (for triplication), then we can directly roll back to the last checkpoint, since the remaining replica(s) would prevent a consensus from being reached for silent error detection. Otherwise, if the pattern survived fail-stop errors (i.e., both replicas are intact for duplication or at least two replicas are intact for triplication), then we can compare the remaining replicas as per the standard consensus protocol for detecting and correcting silent errors.

To distinguish the two error sources, we will now use λ_s and λ_f to denote a single process's error rates for fail-stop errors and silent errors, respectively. Also, we define $\Lambda = \lambda_s + \lambda_f$ to be the total error rate of the process and further assume that both λ_s and λ_f are of the same order, i.e., $\lambda_s = O(\Lambda)$ and $\lambda_f = O(\Lambda)$. Recall that we used $\mathbb{P}_n(T,P)$ to denote the probability that a pattern fails due to silent errors. Similarly, we define $\mathbb{Q}_n(T,P)$ to be the probability that the pattern fails due to fail-stop errors and define $\mathbb{F}_n(T,P)$ to be the probability that the pattern fails due to either error source (i.e., it either fails due to fail-stop errors or survived fail-stop errors but fails due to silent errors). Lastly, we define $\mathbb{E}_n^{\text{lost}}(T,P)$ to be the expected time lost whenever the pattern fails due to fail-stop errors. In that case, let $\mathbb{P}(X=t)$ denote the probability that the failure occurs at time t since the beginning of the pattern. We know that:

$$\mathbb{P}(X \leq T) = \mathbb{Q}_n(T,P) \ .$$

and differentiating the above expression, we can get:

$$\mathbb{P}(X=t) = \frac{d\mathbb{Q}_n(t,P)}{dt} \ .$$

Therefore, we can compute the expected time lost as follows:

$$\begin{aligned}
\mathbb{E}_n^{\text{lost}}(T,P) &= \int_0^{\infty} t\mathbb{P}(X=t|X\leq T)dt \\
&= \frac{1}{\mathbb{P}(X\leq T)} \int_0^{T} t\mathbb{P}(X=t)dt \\
&= \frac{1}{\mathbb{Q}_n(T,P)} \int_0^{T} t\frac{d\mathbb{Q}_n(t,P)}{dt}dt \ .
\end{aligned}$$

Since $\int \frac{d\mathbb{Q}_n(t,P)}{dt} dt = \mathbb{Q}_n(t,P)$, integrating by parts, we can get:

$$\begin{aligned}
\mathbb{E}_n^{\text{lost}}(T,P) &= \frac{1}{\mathbb{Q}_n(T,P)} \left(\Big[t \cdot \mathbb{Q}_n(t,P) \Big]_0^T - \int_0^T \mathbb{Q}_n(t,P) dt \right) \\
&= \frac{1}{\mathbb{Q}_n(T,P)} \left(T \cdot \mathbb{Q}_n(T,P) - \int_0^T \mathbb{Q}_n(t,P) dt \right) \\
&= T - \frac{1}{\mathbb{Q}_n(T,P)} \int_0^T \mathbb{Q}_n(t,P) dt \ . \qquad (34)
\end{aligned}$$

Note that both failure probabilities $\mathbb{Q}_n(T,P)$ and $\mathbb{F}_n(T,P)$ depend on the replication scenario (process v.s. group) as well as on the number of replicas n, and so is the expected time lost $\mathbb{E}_n^{\text{lost}}(T,P)$. In the following analysis, we will derive these quantities under different replication conditions.

7.2. Expected execution time

We first compute the expected execution time of a periodic pattern under both fail-stop and silent errors. Again, the derived formula is generic in the sense that it can be applied to both replication scenarios and regardless of the number of replicas.

Theorem 8. *In the presence of both fail-stop and silent errors, the expected time to execute a periodic pattern of length T using P processes and n replicas can be expressed as*

$$\begin{aligned}
\mathbb{E}_n(T,P) = T + V + C &+ \frac{\mathbb{F}_n(T,P)}{1 - \mathbb{F}_n(T,P)} (T + V + R) \\
&+ \frac{\mathbb{Q}_n(T,P)}{1 - \mathbb{F}_n(T,P)} \left(\mathbb{E}_n^{\text{lost}}(T,P) - T - V \right) \ . \qquad (35)
\end{aligned}$$

Proof. To derive the expected execution time, we focus on four quantities (or events) and identify probabilities associated with each one of them. The following presents a recursive expression for the expected execution time:

$$\begin{aligned}
\mathbb{E}_n(T,P) = &\ \mathbb{Q}_n(T,P) \cdot \mathbb{E}_n^{\text{lost}}(T,P) \\
&+ \big(1 - \mathbb{Q}_n(T,P)\big) \cdot (T + V) \\
&+ \mathbb{F}_n(T,P) \cdot \big(R + \mathbb{E}_n(T,P)\big) \\
&+ \big(1 - \mathbb{F}_n(T,P)\big) \cdot C \ .
\end{aligned}$$

Specifically, with probability $\mathbb{Q}_n(T,P)$, an expected time of $\mathbb{E}_n^{\text{lost}}(T,P)$ is lost due to the occurrence of fail-stop errors (first line). Otherwise, with probability $1 - \mathbb{Q}_n(T,P)$, the entire pattern is executed and the (remaining) replicas are compared against each other to detect silent errors (second line). The pattern fails due to either fail-stop or silent errors with probability $\mathbb{F}_n(T,P)$, and in this case, we recover from the last checkpoint and restart the pattern again (third line). Lastly, the pattern successfully completes with probability $1 - \mathbb{F}_n(T,P)$, and the checkpoint is taken in this case (last line).

Now, solving the recursion above and rearranging terms, we can obtain the expected execution time of the pattern as shown in Equation (35). □

Remarks. The expression of $\mathbb{F}_n(T,P)$ is complicated, and will be derived below for the different scenarios. In a nutshell, the intuitive formula $\mathbb{F}_n(T,P) = \mathbb{Q}_n(T,P) + (1 - \mathbb{Q}_n(T,P))\mathbb{P}_n(T,P)$ does not hold for general n, because we have to account for the number of processes that have been struck by errors. The formula does hold for the simplest case of $n = 2$ though.

7.3. Process replication

As previously, we first consider process replication in this section, and then generalize the results to group replication in the next section.

7.3.1. Process duplication

The following lemma shows the failure probabilities of a pattern and the expected time lost for process duplication. The proof can be found in Appendix B.

Lemma 6. *Using process duplication, the probabilities that a computing pattern of length T and with P processes fails due to fail-stop errors and due to either error source are respectively*

$$\mathbb{Q}_2^{\text{prc}}(T,P) = 1 - e^{-2\lambda_f TP} \ ,$$
$$\mathbb{F}_2^{\text{prc}}(T,P) = 1 - e^{-2\Lambda TP} \ ,$$

and the expected time lost whenever the pattern fails due to fail-stop errors is approximated as

$$\mathbb{E}_2^{\text{lost}}(T,P) \approx \frac{T}{2} \ .$$

We can now derive the optimal pattern with the help of Lemma 6, and the following theorem shows the results. The proof follows closely that of Theorem 2 and is therefore omitted.

Theorem 9. *In the presence of both fail-stop and silent errors, a first-order approximation to the optimal pattern under process duplication is characterized by*

$$P_{\text{opt}} = \min\left\{\frac{Q}{2}, \left(\left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{(2\Lambda-\lambda_f)c}\right)^{\frac{1}{3}}\right\} \ ,$$
$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{V+C}{(2\Lambda-\lambda_f)P_{\text{opt}}}\right)^{\frac{1}{2}} \ ,$$
$$\mathbb{S}_2^{\text{prc}}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1+2\big((2\Lambda-\lambda_f)(V+C)P_{\text{opt}}\big)^{\frac{1}{2}}} \ .$$

7.3.2. Process triplication

We now consider process triplication. First, the following lemma gives the failure probabilities and the expected time lost. The proof can be found in Appendix C.

Lemma 7. *Using process triplication, the probabilities that a computing pattern of length T and with P processes fails due to fail-stop errors and due to either error source are respectively*

$$\mathbb{Q}_3^{\text{prc}}(T,P) = 1 - \left(3e^{-2\lambda_f T} - 2e^{-3\lambda_f T}\right)^P \ ,$$
$$\mathbb{F}_3^{\text{prc}}(T,P) = 1 - \left(3e^{-2\Lambda T} - 2e^{-3\Lambda T}\right)^P \ ,$$

and the expected time lost whenever the pattern fails due to fail-stop errors is approximated as

$$\mathbb{E}_3^{\text{lost}}(T,P) \approx \frac{2T}{3} \ .$$

Remarks. Based on Lemmas 6 and 7, we can see that, compared to process duplication, process triplication enjoys a smaller overall failure probability (with the same proof of Lemma 5), but at the same time suffers from a larger expected time lost when the pattern fails due to fail-stop errors. Intuitively, since triplication is more resilient, it takes more errors and more time to cause a rollback in this scenario.

The following theorem shows the optimal pattern for process triplication. The proof follows closely that of Theorem 3 and is again omitted.

Theorem 10. *In the presence of both fail-stop and silent errors, a first-order approximation to the optimal pattern under process triplication is characterized by*

$$P_{\text{opt}} = \min\left\{\frac{Q}{3}, \left(\left(\frac{1-\alpha}{\alpha}\right)^3 \frac{4}{(3\Lambda^2-\lambda_f^2)c^2}\right)^{\frac{1}{4}}\right\},$$

$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{V+C}{(6\Lambda^2-2\lambda_f^2)P_{\text{opt}}}\right)^{\frac{1}{3}},$$

$$\mathbb{S}_3^{\text{prc}}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1+3\left(\frac{3\Lambda^2-\lambda_f^2}{4}(V+C)^2P_{\text{opt}}\right)^{\frac{1}{3}}}.$$

7.4. Group replication and triplication

Lastly, we consider group replication with both fail-stop and silent errors. Again, by applying Observation 1, we can transform the failure probabilities from process replication to group replication for any given pattern. In fact, the same transformation can also be applied to the expected time lost, which stays the same (i.e., $\mathbb{E}_2^{\text{lost}}(T,P) = \frac{T}{2}$ for duplication and $\mathbb{E}_3^{\text{lost}}(T,P) = \frac{2T}{3}$ for triplication) regardless of whether process replication or group replication is used.

Following the analysis of Section 6, we can derive the optimal pattern for group duplication, which turns out to be the same as that of process duplication as stated in Theorem 9. The following theorem states without proof the optimal pattern for group triplication.

Theorem 11. *In the presence of both fail-stop and silent errors, a first-order approximation to the optimal pattern under group triplication is characterized by*

$$P_{\text{opt}} = \min\left\{\frac{Q}{3}, \left(\left(\frac{1-\alpha}{\alpha}\right)^3 \frac{1}{(6\Lambda^2-2\lambda_f^2)c^2}\right)^{\frac{1}{5}}\right\},$$

$$T_{\text{opt}}(P_{\text{opt}}) = \left(\frac{V+C}{(6\Lambda^2-2\lambda_f^2)P_{\text{opt}}^2}\right)^{\frac{1}{3}},$$

$$\mathbb{S}_3^{\text{grp}}(P_{\text{opt}}) = \frac{S(P_{\text{opt}})}{1+3\left(\frac{3\Lambda^2-\lambda_f^2}{4}\big((V+C)P_{\text{opt}}\big)^2\right)^{\frac{1}{3}}}.$$

Finally, we remark that the optimal patterns presented in this section under both fail-stop and silent errors can further be generalized to the scenario with an arbitrary number of replicas, as analyzed in Sections 5.3 and 6.3. We omit the analysis for this general case, as our simulation results indicate, under a wide range of practical parameters, that the optimal solution does not benefit from using more than three replicas.

8. Simulations

We conduct a set of simulations whose goal is twofold: (i) validate the accuracy of the theoretical study; and (ii) evaluate the efficiency of both process and group replication under different scenarios at extreme scale. The simulator can be freely downloaded from `http://graal.ens-lyon.fr/~yrobert/replication-code.zip`. Interested readers can instantiate their preferred scenarios or repeat the same simulations for reproducibility purpose.

8.1. Simulation setup

The simulator has been designed to simulate each process individually, and each process has its own error trace. A simulation works as follows: we feed the simulator with the model parameters Q, C, V, R, and α, and individual error rate λ (or λ_s and λ_f with both error sources), and we compute the associated optimal number of processes P_{opt} and the optimal checkpointing period $T_{\text{opt}}(P_{\text{opt}})$ using the corresponding model equations. For each run, the simulator outputs the *efficiency*, defined as $\frac{\mathbb{S}(P_{\text{opt}})}{Q}$, as well as the average number of errors and the average number of recoveries per million CPU hours of work. Then, for each of the simulated scenarios, we compare the simulated efficiency to the theoretical value, obtained using the model equations for $\mathbb{S}(P_{\text{opt}})$. As pointed out in Section 6.1, process and group duplications lead to identical patterns, so we have merged the two scenarios and compared it against process and group triplications[7].

The rest of this section presents the simulation results, most of which focus on coping with silent errors only, with the exception of Section 8.5, which considers both fail-stop and silent errors. In the following, we set the cost of recovery to be the same as the checkpoint cost (as discussed in Section 3), and the cost $V + C$ according the values of c and d as in Equation (2). We consider different system Mean Time Between Errors (MTBE), ranging from 10^6 seconds ($\approx$ 11 days) down to 10^2 seconds ($<$ 2 minutes) for $Q = 10^6$ processes, matching the numbers in [47].

8.2. Impacts of $MTBE$ and checkpoint cost

We first presents the impact of $MTBE$ on the efficiency of both duplication and triplication for three different checkpoint costs, using the same value $\alpha = 10^{-6}$ for the sequential fraction of the application (see next section for the impact of varying α). Figure 1 shows the results with a checkpoint cost of 30 minutes (i.e. $c = 1800, d = 0$), Figure 2 shows the results with a checkpoint cost of 60 seconds (i.e. $c = 60, d = 0$), and Figure 3 shows the results with $c = 0, d = 10^7$, which corresponds to a checkpoint cost of 20 seconds for duplication with $\frac{Q}{2}$ processes and 30 seconds for triplication with $\frac{Q}{3}$ processes. In addition to the efficiency, we provide the average number of errors and recoveries per million hours of work, the optimal checkpointing period $T_{\text{opt}}(P_{\text{opt}})$ and the optimal number of processes P_{opt}.

Efficiency. First, we observe from the first plot of Figures 1, 2 and 3 that the difference between the theoretical efficiency and the simulated efficiency is quite small ($<$ 5% absolute difference), which shows the accuracy of the first-order approximation. Then, with very few errors ($MTBE = 10^6$), we observe that duplication is always better than triplication. This is as expected, since the maximum efficiency for duplication is 0.5 (assuming $\alpha = 0$ and no error), while the maximum efficiency for triplication is 0.33. However, as the $MTBE$ decreases, triplication becomes more attractive and eventually outperforms duplication. With a checkpoint cost of 30 minutes (Figure 1), the $MTBE$ required is around 28 hours for process triplication to win and 20 hours for group triplication to win. With smaller checkpoint costs, such as 60 seconds (Figure 2) and 30 seconds (Figure 3), checkpoints can be more frequent and the $MTBE$ required for triplication to win is pushed down to a couple of hours and a couple of minutes, respectively.

Number of errors and recoveries. The second plot of the three figures presents the number of errors and the corresponding number of recoveries per million hours of work. First, we note that the number of errors is always higher than the number of recoveries. This is because multiple errors can occur during a period (before the checkpoint, which is the point of detection), causing a single recovery, and in the case of triplication, some errors are masked by majority voting and hence no recovery is needed. In particular, with $MTBE = 10^2$, almost half of the errors that occurred with duplication were actually hidden behind another error. Even more errors were hidden with group triplication, since one more error (in a different replica) is required to cause a recovery. Finally, (almost) all errors were hidden with process triplication, which is able to handle many errors, as long as they strike in different processes.

[7]We have also simulated process and group replications using more than 3 replicas, but the results under a wide range of practical parameters do not favor those replication scenarios. Therefore, their results are not presented.

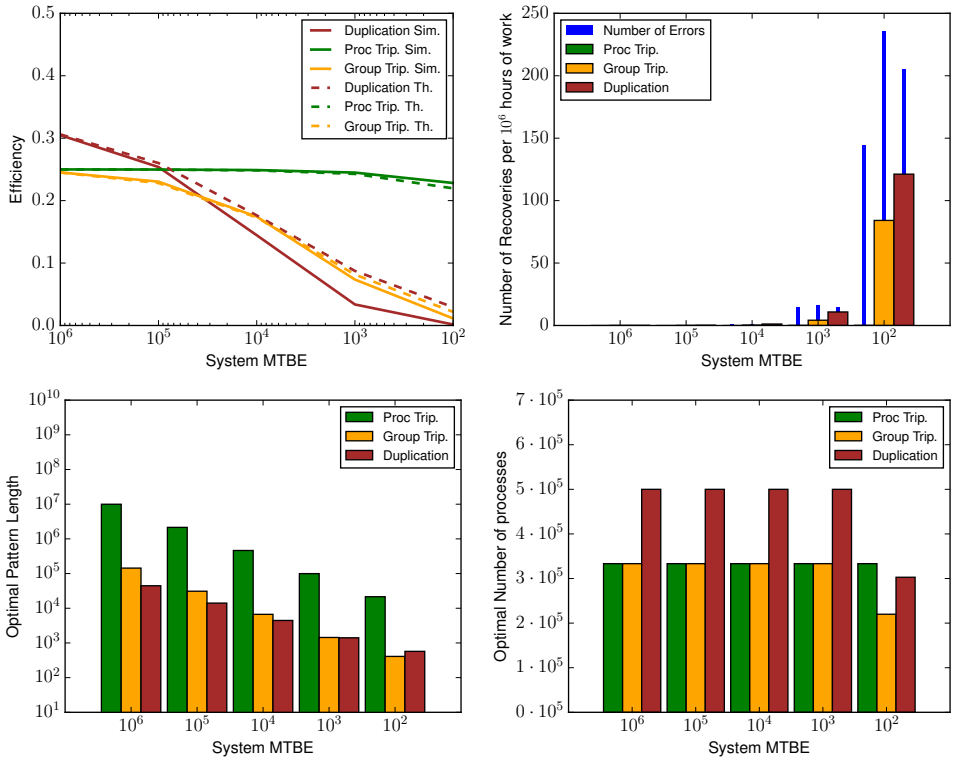

Figure 1: Impact of the system $MTBE$ on the efficiency with $c = 1800, d = 0$, and $\alpha = 10^{-6}$.

Optimal checkpointing period. The third plot of the three figures shows the optimal length of the pattern, or the checkpointing period. In order to cope with the increasing number of errors and recoveries, the optimal period becomes smaller with decreased $MTBE$. Note that the length of the period for group triplication is comparable to that for duplication, around one day when $MTBE = 10^6$ down to a couple of minutes when $MTBE = 10^2$ under different checkpoint costs. However, the length of the period for process triplication is always higher by around two orders of magnitude, from 10 to 100 days when $MTBE = 10^6$ down to a few hours when $MTBE = 10^2$.

Optimal number of processes. The last plot of the three figures shows the optimal number of processes to use. With $\alpha = 10^{-6}$, the application has ample parallelism, so the optimal number of processes is always $\frac{Q}{2} = 5 \cdot 10^5$ for duplication and $\frac{Q}{3} \approx 3.3 \cdot 10^5$ for triplication, except when $MTBE = 10^2$ and $c = 1800$, in which case the combination of large error rate and high checkpoint cost has driven the pattern to decrease the optimal number of processes (around $3 \cdot 10^5$ for duplication and around $2 \cdot 10^5$ for group triplication), in order to reduce the failure probability and to avoid the expensive recovery operations.

8.3. Impact of sequential fraction (Amdahl's Law)

Figure 4 presents two additional sets of simulation results for $\alpha = 10^{-7}$ and $\alpha = 10^{-5}$. With a small sequential fraction of $\alpha = 10^{-7}$ (top plots), the efficiency is improved ($\approx 85\%$ of the maximum efficiency for duplication and $\approx 95\%$ for triplication at $MTBE = 10^6$), and both duplication and triplication use all the processes available. On the contrary, with a relatively higher sequential fraction of $\alpha = 10^{-5}$ (bottom plots), the efficiency drops ($< 20\%$ of the maximum efficiency for duplication and $< 30\%$ for triplication at $MTBE = 10^6$). In this case, using more processes does not improve the efficiency and only contributes to increasing the number of errors. Therefore, these results suggest that with duplication or even triplication, there comes a point where it is no longer beneficial to use all available processes. In this example, when $MTBE = 10^2$, duplication

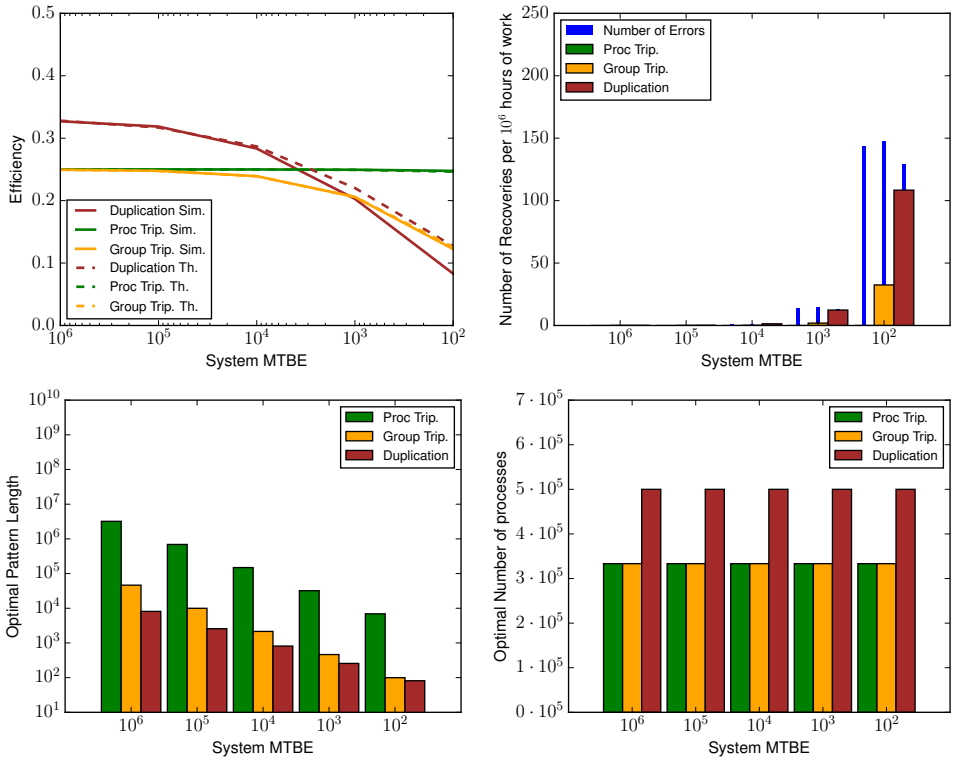

Figure 2: Impact of the system $MTBE$ on the efficiency with $c = 60, d = 0$, and $\alpha = 10^{-6}$.

and group triplication would use fewer than $2 \cdot 10^5$ processes (one fifth of the available resources). Process triplication, on the other hand, still utilizes all the resources and outperforms the other two schemes in terms of the efficiency across the whole range of system $MTBE$.

8.4. Impact of number of processes

Figure 5 shows the impact of the number of processes on the simulated efficiency of different replication scenarios. In addition, we also show (using dots) the theoretical efficiency obtained with the optimal number of processes from Theorems 2, 3 and 6. By varying the number of processes, we find that the simulated optimum (that yields the best efficiency) matches our theoretical optimal number of processes closely. We can also see that process triplication scales very well with increasing number of processes. As opposed to group triplication, which has to recover from a checkpoint if just two errors strike in two different replica groups, process triplication benefits from independent error detection and correction for all processes: from a resilience point of view, each process acts as a buffer to handle one more error; in other words, the probability that two errors strike two replicas of the same process decreases, thereby improving the efficiency.

8.5. Impact of fail-stop errors

Figure 6 focuses on the impact of fail-stop errors on the efficiency. Here, the combined system $MTBE$ (silent + fail-stop errors) is fixed to be 10^3, and we vary the percentage of errors that are fail-stop from 0% (silent errors only) to 100% (fail-stop errors only). The number of errors shown by the right plot is the total number of errors (silent + fail-stop). Note that the optimal checkpointing period and the optimal number of processes remain (almost) unchanged and are not shown. The first observation we can make is that fail-stop errors do not have a noticeable impact on the efficiency of triplication, as long as the total error rate stays constant. However, we observe a slight increase in the efficiency of duplication when there are more fail-stop errors. Indeed, the

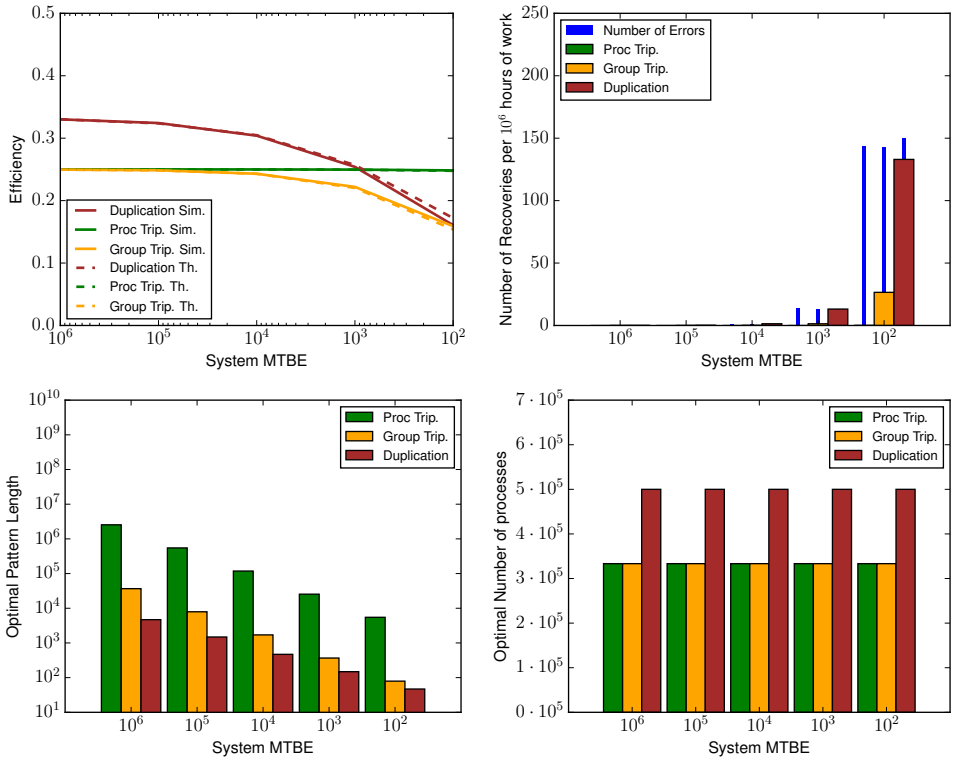

Figure 3: Impact of the system $MTBE$ on the efficiency with $c = 0, d = 10^7$, and $\alpha = 10^{-6}$.

application must roll back as soon as a fail-stop error strikes, which means that less time is lost compared to silent errors (see Lemma 6). This is confirmed by the right plot, which shows that the number of recoveries is slightly higher for duplication when there are more fail-stop errors. Overall, the combined replication and checkpointing schemes are robust enough to handle both fail-stop and silent errors, and a mixture of the two error sources in the execution does not have a significant impact on the performance.

8.6. Summary

In summary, the simulation results have shown that duplication is more efficient than triplication for high system $MTBE$ (e.g., $> 10^5$ seconds). If process triplication is available, then it is always more efficient for small system $MTBE$ (e.g., $< 10^3$ seconds): its efficiency remains stable despite the increasing number of errors. If process triplication is not available, we have shown that group triplication is slightly more efficient than duplication for small $MTBE$, but only marginal gain can be expected. Furthermore, the impact of the sequential fraction α (in Amdahl's Law) is twofold: it limits the efficiency (e.g., $< 30\%$ of the maximum with $\alpha = 10^{-5}$ for both duplication and triplication), and it is a major factor in limiting the optimal resource usage (e.g., one tenth of the platform with $\alpha = 10^{-5}$ and $Q = 10^6$ at $MTBE = 10^2$). Finally, we have observed that adding fail-stop errors does not have a significant impact on these results.

9. Conclusion

Silent errors represent a major threat to the HPC community. In the absence of application-specific detectors, replication is the only solution. Unfortunately, it comes with high cost: by definition, the efficiency is upper-bounded by 0.5 for duplication, and by 0.333 for triplication. Are these upper bounds likely to be achieved? If yes, it means that duplication should always

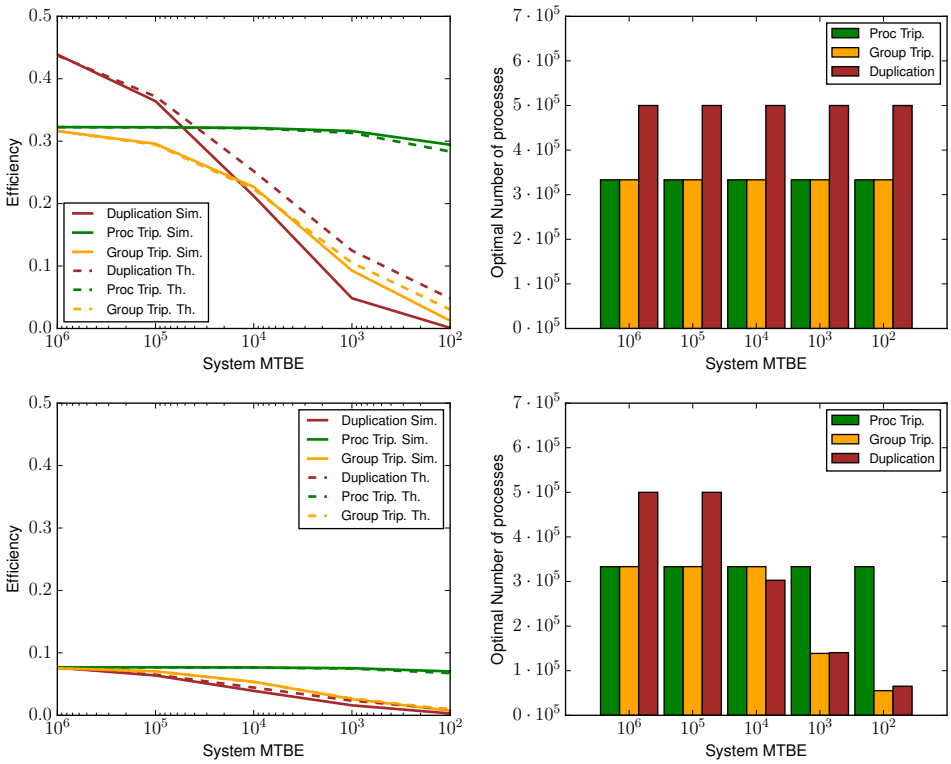

Figure 4: Impact of the sequential fraction (in Amdahl's Law) on the efficiency and optimal number of processes with $\alpha = 10^{-7}$ (top) and $\alpha = 10^{-5}$ (bottom).

be preferred to triplication. If not, it means that in some scenarios, the striking of errors is so frequent that duplication, and in particular group duplication, is not the right choice.

The major contribution of this paper is to provide an in-depth analysis of process and group duplication, and of process and group triplication. Given a level n of replication, and a set of application/platform parameters (speedup profile, total number or processes, process MTBE, checkpoint cost, etc.), we derive closed-form formulas for the optimal checkpointing period, the optimal resource usage, and the overall efficiency of the approach, when the platform is subject to both silent and fail-stop errors. This allows us to choose the best value of n. A set of simulations demonstrates the accuracy of the model and analysis. Our simulator code has been made publicly available, so that interested readers can instantiate their preferred scenario. Altogether, this paper has laid the foundations for a better understanding of using replication and checkpointing to cope with fail-stop and silent errors for HPC at scale.

An interesting topic to explore in future work is partial replication: if the application comes as a workflow whose tasks are atomic components, one could assign different replication levels (duplication, triplication or more) to different tasks, depending upon their criticality in terms of longest paths, number of successors, etc. Although partial replication has been empirically studied by some previous work [26, 48, 49], designing an optimal strategy that combines partial redundancy and checkpointing and analyzing its efficacy remain to be done.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions, which helped improve the quality of this paper. This research was supported in part by the National Science Foundation grant CCF 1719674 and Vanderbilt Institutional Fund.

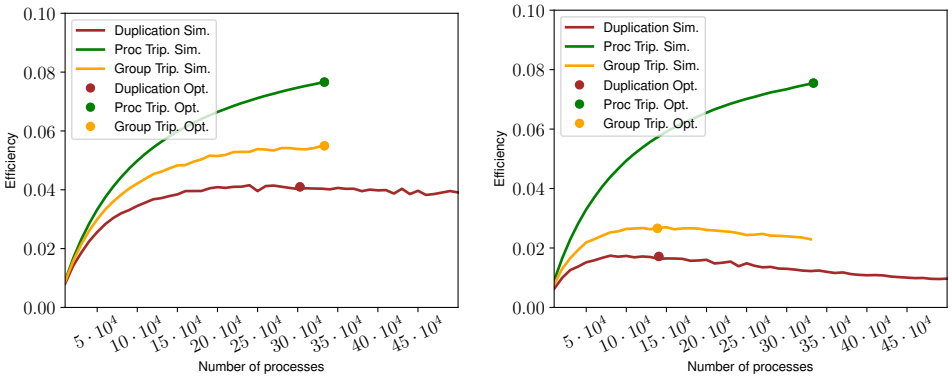

Figure 5: Impact of the number of processes on the efficiency with $MTBE = 10^4$ (left), $MTBE = 10^3$ (right), and $c = 1800$, $d = 0$, $\alpha = 10^{-5}$.

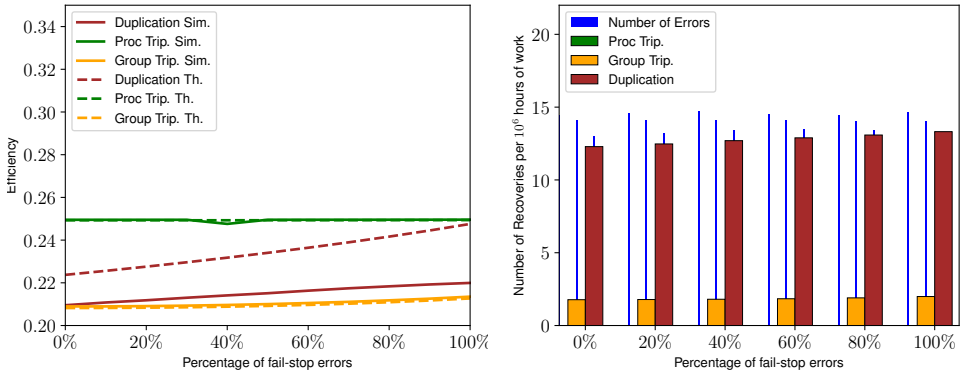

Figure 6: Impact of fail-stop errors as a percentage of the total number of errors with combined $MTBE = 10^3$, $c = 60, d = 0$, and $\alpha = 10^{-6}$.

References

[1] A. Avizienis, J. Laprie, B. Randell, and C. E. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Sec. Comput.*, 1(1):11–33, 2004.

[2] L. Bautista-Gomez, A. Benoit, A. Cavelan, S. K. Raina, Y. Robert, and H. Sun. Which verification for soft error detection? In *HiPC*. IEEE, 2015.

[3] L. Bautista Gomez and F. Cappello. Detecting silent data corruption through data dynamic monitoring for scientific applications. In *PPoPP*. ACM, 2014.

[4] L. Bautista Gomez and F. Cappello. Detecting and correcting data corruption in stencil applications through multivariate interpolation. In *FTS*. IEEE, 2015.

[5] L. Bautista Gomez and F. Cappello. Exploiting spatial smoothness in HPC applications to detect silent data corruption. In *HPCC*. IEEE, 2015.

[6] A. Benoit, A. Cavelan, F. Cappello, P. Raghavan, Y. Robert, and H. Sun. Identifying the right replication level to detect and correct silent errors at scale. In *Fault Tolerance for HPC at eXtreme Scale (FTXS) Workshop*. ACM, 2017.

[7] A. Benoit, A. Cavelan, V. L. Fèvre, Y. Robert, and H. Sun. Towards optimal multi-level checkpointing. *IEEE Transactions on Computers*, 66(7):1212–1226, 2017.

[8] A. Benoit, A. Cavelan, Y. Robert, and H. Sun. Assessing general-purpose algorithms to cope with fail-stop and silent errors. In *PMBS*. ACM, 2014.

[9] A. Benoit, A. Cavelan, Y. Robert, and H. Sun. Optimal resilience patterns to cope with fail-stop and silent errors. In *IPDPS*. IEEE, 2016.

[10] A. R. Benson, S. Schmit, and R. Schreiber. Silent error detection in numerical time-stepping schemes. *Int. J. High Performance Computing Applications*, 2014.

[11] E. Berrocal, L. Bautista-Gomez, S. Di, Z. Lan, and F. Cappello. Lightweight silent data corruption detection based on runtime data analysis for HPC applications. In *HPDC*. ACM, 2015.

[12] G. Bosilca, R. Delmas, J. Dongarra, and J. Langou. Algorithm-based fault tolerance applied to high performance computing. *J. Parallel Distrib. Comput.*, 69(4):410–416, 2009.

[13] G. Bronevetsky and B. de Supinski. Soft error vulnerability of iterative linear algebra methods. In *ICS*. ACM, 2008.

[14] F. Cappello, E. M. Constantinescu, P. D. Hovland, T. Peterka, C. Phillips, M. Snir, and S. M. Wil. Improving the trust in results of numerical simulations and scientific data analytics. White paper MCS-TM-352, ANL, 2015.

[15] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir. Toward Exascale Resilience. *Int. J. High Performance Computing Applications*, 23(4):374–388, 2009.

[16] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir. Toward Exascale Resilience: 2014 update. *Supercomputing frontiers and innovations*, 1(1), 2014.

[17] H. Casanova, M. Bougeret, Y. Robert, F. Vivien, and D. Zaidouni. Using group replication for resilience on exascale systems. *Int. Journal of High Performance Computing Applications*, 28(2):210–224, 2014.

[18] H. Casanova, Y. Robert, F. Vivien, and D. Zaidouni. On the impact of process replication on executions of large-scale parallel applications with coordinated checkpointing. *Future Generation Comp. Syst.*, 51:7–19, 2015.

[19] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Comput. Surv.*, 26(2):145–185, June 1994.

[20] E. Ciocca, I. Koren, Z. Koren, C. M. Krishna, and D. S. Katz. Application-level fault tolerance in the orbital thermal imaging spectrometer. In *PRDC*. IEEE, 2004.

[21] S. P. Crago, D. I. Kang, M. Kang, R. Kost, K. Singh, J. Suh, and J. P. Walters. Programming models and development software for a space-based many-core processor. In *4th Int. Conf. onon Space Mission Challenges for Information Technology*, pages 95–102. IEEE, 2011.

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

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

[24] J. Dongarra and al. The international exascale software project roadmap. *Int. J. High Perform. Comput. Appl.*, 25(1):3–60, 2011.

[25] J. Elliott, M. Hoemmen, and F. Mueller. Evaluating the impact of SDC on the GMRES iterative solver. In *IPDPS*. IEEE, 2014.

[26] J. Elliott, K. Kharbas, D. Fiala, F. Mueller, K. Ferreira, and C. Engelmann. Combining partial redundancy and checkpointing for HPC. In *ICDCS*. IEEE, 2012.

[27] E. Elnozahy and J. Plank. Checkpointing for peta-scale systems: A look into the future of practical rollback-recovery. *IEEE Transactions on Dependable and Secure Computing*, 1(2):97—108, 2004.

[28] C. Engelmann, H. H. Ong, and S. L. Scorr. The case for modular redundancy in large-scale high performance computing systems. In *PDCN*. IASTED, 2009.

[29] C. Engelmann and B. Swen. Redundant execution of HPC applications with MR-MPI. In *PDCN*. IASTED, 2011.

[30] 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 *SC'11*. ACM, 2011.

[31] D. Fiala, F. Mueller, C. Engelmann, R. Riesen, K. Ferreira, and R. Brightwell. Detection and correction of silent data corruption for large-scale high-performance computing. In *SC*. ACM, 2012.

[32] C. George and S. S. Vadhiyar. ADFT: An adaptive framework for fault tolerance on large scale systems using application malleability. *Procedia Computer Science*, 9:166 – 175, 2012.

[33] M. Heroux and M. Hoemmen. Fault-tolerant iterative methods via selective reliability. Research report SAND2011-3915 C, Sandia Nat. Lab., 2011.

[34] K.-H. Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Comput.*, 33(6):518–528, 1984.

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

[36] T. Leblanc, R. Anand, E. Gabriel, and J. Subhlok. VolpexMPI: An MPI Library for Execution of Parallel Applications on Volatile Nodes. In *16th European PVM/MPI Users' Group Meeting*, pages 124–133. Springer-Verlag, 2009.

[37] M. Li and P. P. C. Lee. Toward I/O-efficient protection against silent data corruptions in RAID arrays. In *30th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–12, 2014.

[38] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM J. Res. Dev.*, 6(2):200–209, 1962.

[39] A. Moody, G. Bronevetsky, K. Mohror, and B. R. d. Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *SC*. ACM, 2010.

[40] X. Ni, E. Meneses, N. Jain, and L. V. Kalé. ACR: Automatic Checkpoint/Restart for Soft and Hard Error Protection. In *SC*. ACM, 2013.

[41] T. O'Gorman. The effect of cosmic rays on the soft error rate of a DRAM at ground level. *IEEE Trans. Electron Devices*, 41(4):553–557, 1994.

[42] R. A. Oldfield, S. Arunagiri, P. J. Teller, S. Seelam, M. R. Varela, R. Riesen, and P. C. Roth. Modeling the impact of checkpoints on next-generation systems. In *24th IEEE Conf. Mass Storage Systems and Technologies*. IEEE, 2007.

[43] M. W. Rashid and M. C. Huang. Supporting highly-decoupled thread-level redundancy for parallel programs. In *14th Int. Conf. on High-Performance Computer Architecture (HPCA)*, pages 393–404. IEEE, 2008.

[44] P. Sao and R. Vuduc. Self-stabilizing iterative solvers. In *ScalA '13*, 2013.

[45] B. Schroeder and G. A. Gibson. Understanding Failures in Petascale Computers. *Journal of Physics: Conference Series*, 78(1), 2007.

[46] M. Shantharam, S. Srinivasmurthy, and P. Raghavan. Fault tolerant preconditioned conjugate gradient for sparse linear system solution. In *ICS*. ACM, 2012.

[47] M. Snir and al. Addressing failures in exascale computing. *Int. J. High Perform. Comput. Appl.*, 28(2):129–173, 2014.

[48] J. Stearley, K. B. Ferreira, D. J. Robinson, J. Laros, K. T. Pedretti, D. Arnold, P. G. Bridges, and R. Riesen. Does partial replication pay off? In *FTXS*. IEEE, 2012.

[49] O. Subasi, J. Arias, O. Unsal, J. Labarta, and A. Cristal. Programmer-directed partial redundancy for resilient HPC. In *Computing Frontiers*. ACM, 2015.

[50] S. Yi, D. Kondo, B. Kim, G. Park, and Y. Cho. Using replication and checkpointing for reliable task management in computational grids. In *SC*. ACM, 2010.

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

[52] J. Yu, D. Jian, Z. Wu, and H. Liu. Thread-level redundancy fault tolerant CMP based on relaxed input replication. In *ICCIT*. IEEE, 2011.

[53] Z. Zheng and Z. Lan. Reliability-aware scalability models for high performance computing. In *Cluster Computing*. IEEE, 2009.

[54] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, and B. Chin. IBM experiments in soft fails in computer electronics. *IBM J. Res. Dev.*, 40(1):3–18, 1996.

Appendix A. Proof of Theorem 3

Proof. From Theorem 1 and Lemma 2, and applying Taylor series while assuming $\lambda^2 PT^2 = \Theta(\lambda^\epsilon)$, where $\epsilon > 0$, we can derive the expected execution time of a pattern as follows:

$$\begin{aligned}
\mathbb{E}_3^{\text{prc}}(T,P) &= T + V + C + \frac{1 - \left(3e^{-2\lambda T} + 2e^{-3\lambda T}\right)^P}{\left(3e^{-2\lambda T} - 2e^{-3\lambda T}\right)^P}\left(T + V + R\right) \\
&= T + V + C + \left(\left(\frac{e^{3\lambda T}}{3e^{\lambda T} - 2}\right)^P - 1\right)\left(T + V + R\right) \\
&\approx T + V + C + \left(\left(\frac{1 + 3\lambda T + \frac{(3\lambda T)^2}{2}}{1 + 3\lambda T + \frac{3(\lambda T)^2}{2}}\right)^P - 1\right)(T + V + R) \\
&\approx T + V + C + \left(\left(1 + 3(\lambda T)^2\right)^P - 1\right)(T + V + R) \\
&= T + V + C + \left(\sum_{j=0}^{P}\binom{P}{j}\left(3(\lambda T)^2\right)^j - 1\right)(T + V + R) \\
&= T + V + C + 3P(\lambda T)^2(T + V + R) + o(\lambda^2 PT^3)\ .
\end{aligned}$$

The execution overhead can then be expressed as:

$$\mathbb{H}_3^{\text{prc}}(T,P) = H(P)\left(1 + \frac{V + C}{T} + 3P(\lambda T)^2 + o(\lambda^2 PT^2)\right)\ . \tag{36}$$

The optimal checkpointing period is then obtained by setting

$$\frac{\partial \mathbb{H}_3^{\text{prc}}(T,P)}{\partial T} = -\frac{V+C}{T^2} + 6\lambda^2 PT = 0 \ ,$$

which gives

$$T_{\text{opt}}(P) = \left(\frac{V+C}{6\lambda^2 P}\right)^{\frac{1}{3}} \ .$$

Substituting $T_{\text{opt}}(P)$ back into Equation (36), we get the following execution overhead (with lower-order terms ignored):

$$\mathbb{H}_3^{\text{prc}}(P) = H(P)\left(1 + 3\left(\frac{3}{4}\left(\lambda(V+C)\right)^2 P\right)^{\frac{1}{3}}\right) \ . \tag{37}$$

To derive the optimal process count, consider $V + C = c$ and $H(P) = \alpha + \frac{1-\alpha}{P}$ for $\alpha > 0$. Then, Equation (11) can be expanded as

$$\mathbb{H}_3^{\text{prc}}(P) = \alpha + 3\alpha\left(\frac{3}{4}\left(\lambda c\right)^2 P\right)^{\frac{1}{3}} + \frac{1-\alpha}{P} + o(\lambda^{\frac{2}{3}}) \ . \tag{38}$$

The optimal overhead is achieved by setting

$$\frac{\partial \mathbb{H}_3^{\text{prc}}(P)}{\partial P} = \alpha\left(\frac{3}{4}\left(\lambda c\right)^2 \frac{1}{P^2}\right)^{\frac{1}{3}} - \frac{1-\alpha}{P^2} = 0 \ ,$$

which gives rise to $P^* = \left(\frac{4}{3}\left(\frac{1-\alpha}{\alpha}\right)^3\left(\frac{1}{\lambda c}\right)^2\right)^{\frac{1}{4}}$. Now, the optimal process count is upper-bounded by $\frac{Q}{3}$. Thus, P_{opt} is given by Equation (14), which again holds true when $c = 0$ or $\alpha = 0$, and the optimal expected speedup satisfies $\mathbb{S}_3^{\text{prc}}(P_{\text{opt}}) = \frac{1}{\mathbb{H}_3^{\text{prc}}(P_{\text{opt}})}$, as shown in Equation (16). □

Appendix B. Proof of Lemma 6

Proof. Following the proof of Lemma 1, the failure probability $\mathbb{Q}_2^{\text{prc}}(T,P)$ due to fail-stop errors can be readily derived in a similar way. To derive the overall failure probability $\mathbb{F}_2^{\text{prc}}(T,P)$, let $\mathbb{Q}_1^{\text{prc}}(T,1) = 1 - e^{-\lambda_f T}$ and $\mathbb{P}_1^{\text{prc}}(T,1) = 1 - e^{-\lambda_s T}$ denote the probabilities that a single process is hit by fail-stop errors and silent errors, respectively. We first focus on the failure probability of any single duplicated process, which is derived as follows:

$$\begin{aligned}\mathbb{F}_2^{\text{prc}}(T,1) = &\binom{2}{2}\mathbb{Q}_1^{\text{prc}}(T,1)^2 + \binom{2}{1}(1-\mathbb{Q}_1^{\text{prc}}(T,1))\mathbb{Q}_1^{\text{prc}}(T,1) \\ &+ \binom{2}{0}(1-\mathbb{Q}_1^{\text{prc}}(T,1))^2\left[\binom{2}{2}\mathbb{P}_1^{\text{prc}}(T,1)^2 + \binom{2}{1}\left(1-\mathbb{P}_1^{\text{prc}}(T,1)\right)\mathbb{P}_1^{\text{prc}}(T,1)\right] \ .\end{aligned}$$

In particular, the first line represents the probability that the process fails due to fail-stop errors, and the second line represents the probability that the process survived fail-stop errors but subsequently fails due to silent errors. Recall that in this case, a single fail-stop error means that there is a failure, since we will not be able to verify whether the process has been struck by a silent error. Substituting $\mathbb{Q}_1^{\text{prc}}(T,1)$ and $\mathbb{P}_1^{\text{prc}}(T,1)$ into the equation above and simplifying gives us $\mathbb{F}_2^{\text{prc}}(T,1) = 1 - e^{-2\Lambda T}$. With P independent processes, the probability that the whole pattern fails due to either error source is therefore:

$$\begin{aligned}\mathbb{F}_2^{\text{prc}}(T,P) &= 1 - \mathbb{P}(\text{“No process fails”}) \\ &= 1 - \left(1 - \mathbb{F}_2^{\text{prc}}(T,1)\right)^P \\ &= 1 - e^{-2\Lambda PT} \ .\end{aligned}$$

Now, to compute the expected time lost due to fail-stop errors, we first approximate $\mathbb{Q}_2^{\text{prc}}(T,P) = 1 - e^{-2\lambda_f TP} \approx 2\lambda_f TP$. Substituting it into Equation (34), we get:

$$\begin{aligned}\mathbb{E}_2^{\text{lost}}(T,P) &\approx T - \frac{1}{2\lambda_f TP}\int_0^T 2\lambda_f tP dt \\ &= T - \frac{1}{T}\int_0^T t dt \\ &= \frac{T}{2}\ .\end{aligned}$$

This completes the proof of Lemma 6. □

Appendix C. Proof of Lemma 7

Proof. Again, the failure probability $\mathbb{Q}_3^{\text{prc}}(T,P)$ due to fail-stop errors can be derived by following the proof of Lemma 2. To derive the overall failure probability $\mathbb{F}_3^{\text{prc}}(T,P)$, we again consider the failure probability of any single triplicated process as follows:

$$\begin{aligned}\mathbb{F}_3^{\text{prc}}(T,1) = &\binom{3}{3}\mathbb{Q}_1^{\text{prc}}(T,1)^3 + \binom{3}{2}\left(1-\mathbb{Q}_1^{\text{prc}}(T,1)\right)\mathbb{Q}_1^{\text{prc}}(T,1)^2 \\ &+ \binom{3}{1}\left(1-\mathbb{Q}_1^{\text{prc}}(T,1)\right)^2\mathbb{Q}_1^{\text{prc}}(T,1)\cdot\left[\binom{2}{2}\mathbb{P}_1^{\text{prc}}(T,1)^2 + \binom{2}{1}\mathbb{P}_1^{\text{prc}}(T,1)(1-\mathbb{P}_1^{\text{prc}}(T,1))\right] \\ &+ \binom{3}{0}\left(1-\mathbb{Q}_1^{\text{prc}}(T,1)\right)^3\cdot\left[\binom{3}{3}\mathbb{P}_1^{\text{prc}}(T,1)^3 + \binom{3}{2}\left(1-\mathbb{P}_1^{\text{prc}}(T,1)\right)\mathbb{P}_1^{\text{prc}}(T,1)^2\right]\ .\end{aligned}$$

In the above expression, the first line represents the probability that the process fails due to fail-stop errors (at most one replica is surviving, and hence we cannot detect silent errors). The next two lines represent the probabilities of two different scenarios where the process survived fail-stop errors (by leaving at least two replicas alive), but subsequently fails due to silent errors. Evaluating it gives us $\mathbb{F}_3^{\text{prc}}(T,1) = 1 - 3e^{-2\Lambda T} + 2e^{-3\Lambda T}$. Thus, the probability that the whole pattern fails due to either error source is given by

$$\begin{aligned}\mathbb{F}_3^{\text{prc}}(T,P) &= 1 - \mathbb{P}(\text{“No process fails”}) \\ &= 1 - \left(1-\mathbb{F}_3^{\text{prc}}(T,1)\right)^P \\ &= 1 - \left(3e^{-2\Lambda T} - 2e^{-3\Lambda T}\right)^P\ .\end{aligned}$$

Finally, to compute the expected time lost, we can approximate $\mathbb{Q}_3(T,P)$ up to second-order terms as follows:

$$\begin{aligned}\mathbb{Q}_3(T,P) &= 1 - \left(3e^{-2\lambda_f T} - 2e^{-3\lambda_f T}\right)^P \\ &\approx 1 - \left(3\left(1 - 2\lambda_f T + \frac{(2\lambda_f T)^2}{2}\right) - 2\left(1 - 3\lambda_f T + \frac{(3\lambda_f T)^2}{2}\right)\right)^P \\ &= 1 - (1 - 3(\lambda_f T)^2)^P \\ &= 1 - \sum_{j=0}^{P}\binom{P}{j}(-3)^j(\lambda_f T)^{2j} \\ &\approx 3P(\lambda_f T)^2\ .\end{aligned}$$

Substituting it into Equation (34), we get:

$$\begin{aligned}\mathbb{E}_3^{\text{lost}}(T,P) &\approx T - \frac{1}{3P(\lambda_f T)^2}\int_0^T 3P(\lambda_f t)^2 dt \\ &= T - \frac{1}{T^2}\int_0^T t^2 dt \\ &= \frac{2T}{3}\ .\end{aligned}$$

This completes the proof of Lemma 7. □