Partial Redundancy in HPC Systems with Non-Uniform Node Reliabilities

Zaeem Hussain
Department of Computer Science
University of Pittsburgh
zaeem@cs.pitt.edu

Taieb Znati
Department of Computer Science
University of Pittsburgh
znati@cs.pitt.edu

Rami Melhem
Department of Computer Science
University of Pittsburgh
melhem@cs.pitt.edu

***Abstract*—We study the usefulness of partial redundancy in HPC message passing systems where individual node failure distributions are not identical. Prior research works on fault tolerance have generally assumed identical failure distributions for the nodes of the system. In such settings, partial replication has never been shown to outperform the two extremes(full and no-replication) for any significant range of node counts. In this work, we argue that partial redundancy may provide the best performance under the more realistic assumption of non-identical node failure distributions. We provide theoretical results on arranging nodes with different reliability values among replicas such that system reliability is maximized. Moreover, using system reliability to compute MTTI (mean-time-to-interrupt) and expected completion time of a partially replicated system, we numerically determine the optimal partial replication degree. Our results indicate that partial replication can be a more efficient alternative to full replication at system scales where Checkpoint/Restart alone is not sufficient.**

***Keywords*—HPC, fault tolerance, resilience, replication, checkpoint.**

I. Introduction

With the increasing scale of modern day high performance computing systems, faults are becoming a growing concern. This is simply a consequence of the increasing number of resources being used in HPC platforms. Even though, on average, individual components fail after several years, the sheer number of these components inside the system means that the entire system experiences failures at a much higher rate, usually on the order of days[1].

One of the most common techniques to deal with faults is Checkpointing and Rollback recovery. However, as system scale increases, the overall failure rate is also expected to increase, which means more checkpoints need to be taken. This causes a lot of system time to be spent writing checkpoints and recovering from failures, rather than doing useful work. As an alternative to checkpointing, [2] proposed replication to improve the system reliability at large scales. In pure replication with dual redundancy, all work is duplicated on separate hardware resources. This allows the application to continue execution even in the presence of failures as long as both processes (or nodes) that are replicas of each other have not failed. This significantly improves the mean time to interrupt (MTTI) of the system[2][3], requiring fewer checkpoints compared to the case without replication. However, it comes at the cost of system efficiency, which is capped at 50%. Hence, the argument for pure replication as a fault tolerance mechanism holds weight only at system scales at which the efficiency of checkpointing alone drops below 50%. To break the 50% efficiency barrier of pure replication, [4] studied partial replication where only a subset of application visible nodes are replicated. However, neither [4] nor any other works since then have established any range of node counts for which it makes sense to only partially replicate an execution.

The above mentioned fault tolerance techniques have traditionally been studied in the HPC community with the assumption that all of the individual nodes in a system have independent and identical (iid) failure distributions. While this does simplify the theoretical analysis, there is no experimental evidence to suggest that this assumption holds true in reality. In fact, several studies[5][6][7][8] on failures experienced by supercomputing systems have concluded that failures are spatially non-uniform. One natural approach to model such systems is to assume that individual node failure distributions are still independent, but not identical. In this work, we study the usefulness of partial redundancy for such systems. It should be noted that changing the iid failure assumption does not simply mean redoing the theoretical and numerical analysis, but rather brings up other questions as well. One such question, for example, that we answer in this paper, is: which nodes in the system should be replicated and which nodes should they be replicated with? We answer this and other questions exploring the optimal replication factor (ratio of total nodes used to the number of application visible nodes) of such systems through a combination of theoretical and numerical analyses. To the best of our knowledge, this is the first work that assumes a non-uniform failure distribution of individual nodes and provides theoretical insights into how such a system should be partially replicated. Specifically, we make the following contributions:

1) We theoretically determine, given the total number of nodes in a system with non-identical node failure distributions and the fraction of nodes to be replicated, the selection of nodes into replicated and non-replicated sets and the pairing of replicas such that system reliability is maximized.
2) Using the system reliability to compute the MTTI and average completion time of an execution, we numerically determine the optimal partial replication factor and demonstrate that optimal performance can often be achieved through partial redundancy.
3) We investigate in detail a hypothetical system involving two kinds of nodes: Good, that are less likely to

SC18, November 11-16, 2018, Dallas, Texas, USA
978-1-5386-8384-2/18/$31.00

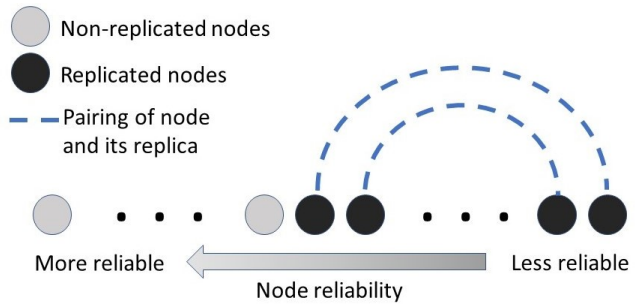

Fig. 1. Selection and pairing of replicas to maximize reliability.

fail, and Bad, that are more likely to fail. We show how different parameters affect the optimal partial replication factor. Our work provides a framework which can be used by system administrators and job schedulers to decide which nodes to replicate in systems where individual nodes' mean-time-between-failures (MTBFs), while not necessarily accurately modeled, are known to take either a high or a low value to a first order approximation.

Even at node counts where the performance of simple checkpoint/restart drops drastically, pure replication still seems like an overkill. Our work attempts to demonstrate that, instead of blindly replicating every node, a smarter choice can be made by understanding the failure characteristics of the underlying system and replicating accordingly.

The remainder of this paper is organized as follows: section II provides results on the system configuration that maximizes reliability, section III presents the mathematical details of the model and the optimization problem, sections IV, V and VI present the results of the optimization for different types of systems, section VII surveys some of the related work and section VIII concludes.

II. Maximizing Reliability

We start with the question of how, knowing the number of nodes to replicate, should the replicated nodes be selected and paired. Consider a system with N nodes with individual node failure density functions given by $h_i(t), 1 \leq i \leq N$, where $t > 0$ is the time. These functions are typically taken to be exponential or Weibull, and characterized by failure rate λ_i, where λ_i is the inverse of node i's MTBF. Individual node MTBFs can be assigned by observing their failure history. For example, works such as [5], [6] and [7] explore the spatial distribution of failures by analyzing the number of failures over time at the cabinet, cage, blade and node granularities for multiple HPC systems at the Oak Ridge National Laboratory (ORNL) and Los Alamos National Laboratory (LANL) over several years. Such analyses can be used to estimate MTBF down to the level of individual nodes or group of nodes. A more sophisticated approach to compute the reliability of individual nodes is presented in [9].

We assume without loss of generality that the nodes are ordered by their failure rates, such that $\lambda_i(t) \leq \lambda_{i+1}(t)$ for all $1 \leq i \leq N-1$. Note that the iid failure distribution assumption is a special case of this in which all λ_i's have the same value. In order to answer the question of optimal selection and pairing of replicas, it is simpler to work with the nodes' probability of survival until time t (or *reliability*) given by $g_i(t) = 1 - \int_0^t h_i(x)dx, 1 \leq i \leq N$. With the nodes sorted by increasing failure rates, we see that $g_i(t) \geq g_{i+1}(t)$ for all $1 \leq i \leq N-1$.

Assume, for now, that a particular job requires n nodes to execute in parallel, where $n \leq N$. Moreover, assume that the remaining $N-n$ nodes are to be used as replicas of some of the n nodes, in order to provide better protection from failures. We will relax these assumptions in subsequent sections to make n variable in order to explore if partial replication is beneficial at all. For now, however, we try to answer the first question: Which of the n nodes should have replicas, and how should they be paired with the other $N-n$ nodes to form node-replica pairs? We restrict ourselves to maximum dual node replication only, so $N/2 \leq n \leq N$. In such a configuration, let $a = n-(N-n) = 2n-N$ be the number of non replicated nodes, and $b = n-a = N-n$ be the number of node replica pairs, such that $a+2b = N$ and $a+b = n$. The partial replication factor, r, will thus be given by $r = (a+2b)/(a+b)$, and $1 \leq r \leq 2$. Our original question can thus be reformulated as: Given values of a and b and reliability $g_i(t), 1 \leq i \leq N$, which $2b$ out of the N nodes should be replicated and how should the replicated nodes be paired so that overall system reliability is maximized? The answer is to pick the least reliable $2b$ nodes for replication. Among those $2b$ nodes, the least reliable node should be paired with the most reliable node, and so on. This is shown in Fig 1, and formally stated in the following theorem:

Theorem. *Given a, b and an N node system ($a+2b = N$) with node reliability given by $g_i(t)$ and $g_i(t) \geq g_{i+1}(t)$ for $1 \leq i \leq N-1$, let $A \subseteq \{1,2,\ldots,N\}$, $|A| = a$, be the set of non-replicated nodes and $B = \{(j,k) \mid j,k \in \{1,2,\ldots N\} - A$ and $j \neq k\}$, $|B| = b$, be the set of node-replica pairs. Maximum overall system reliability is achieved when $A = \{1,2,\ldots,a\}$ and $B = \{(j, 2(a+b)+1-j) \mid j \in \{a+1, a+2, \ldots a+b\}\}$.*

To determine the overall reliability for a given partial replication configuration, we observe that, for a node-replica pair (j,k), application failure occurs when both nodes in the pair fail. Hence, the reliability of pair (j,k) is given by $1-(1-g_j(t))(1-g_k(t))$. For sets A and B as defined above, the overall system reliability $R(t)$ can thus be written as

$$R(t) = \prod_{i \in A} g_i(t) \prod_{(j,k) \in B} (1-(1-g_j(t))(1-g_k(t))) \quad (1)$$

For simplicity, we remove variable t and obtain

$$R = \prod_{i \in A} g_i \prod_{(j,k) \in B} (1-(1-g_j)(1-g_k)) \quad (2)$$

We prove the above theorem in two lemmas. First we prove that maximum reliability is achieved when the set of non-replicated nodes consists of the most reliable nodes.

Lemma 1. *R is maximized when $A = \{1,2,\ldots,a\}$.*

Proof: Assume by contradiction that we have a configuration in which $A \neq \{1,2,\ldots,a\}$. This means there is a node with higher reliability in the replicated set and a node with lower reliability that is not replicated. In other words, $\exists g_i$ where $i \in A$ and $i > a$ and $\exists$ a pair $(j,k) \in B$ such that

at least one of j or k is in $\{1, 2, \ldots, a\}$. Assume without loss of generality that $j \in \{1, 2, \ldots, a\}$. This means that $j < i$, and we know, from the ordering of node reliability, that $g_j \geq g_i$. The contribution of nodes i, j, k in this configuration to system reliability, R, is given by $g_i(1-(1-g_j)(1-g_k)) = g_i(g_j + g_k - g_j g_k)$. We have

$$\begin{aligned} g_i(g_j + g_k - g_j g_k) &= g_i g_j + g_i g_k - g_i g_j g_k \\ &\leq g_i g_j + g_j g_k - g_i g_j g_k \text{ (since } g_i \leq g_j) \\ &= g_j(1 - (1 - g_i)(1 - g_k)) \end{aligned} \tag{3}$$

Since $g_i(1-(1-g_j)(1-g_k)) \leq g_j(1-(1-g_i)(1-g_k))$ with equality iff $g_j = g_i$, we observe that if we exchange nodes i and j between sets A and B, while keeping everything else the same, we obtain a system with reliability R' such that $R' \geq R$. We can keep performing these exchanges as long as $A \neq \{1, 2, \ldots, a\}$. Each exchange step will either improve the system reliability, R, or keep it the same. Hence, R will be maximized when $A = \{1, 2, \ldots, a\}$. ∎

We now move to the second part of the theorem regarding the pairing of replicas. Rewriting $R = R_A R_B$ where $R_A = \prod_{i \in A} g_i$ and $R_B = \prod_{(j,k) \in B}(1-(1-g_j)(1-g_k))$, we focus solely on R_B since R_A is determined from lemma 1. Our job, then, is to show that, given $2b$ numbers $g_1 \geq g_2 \geq \cdots \geq g_{2b}$, R_B is maximized when $B = \{(j, 2b+1-j) \mid j \in \{1, 2, \ldots b\}\}$. To simplify the expressions, we will rewrite R_B in terms of the node failure probabilities, $p_i = 1-g_i$, $1 \leq i \leq 2b$ as $R_B = \prod_{(j,k) \in B}(1 - p_j p_k)$. The ordering of the failure probabilities then becomes $p_1 \leq p_2 \leq \cdots \leq p_{2b}$.

Lemma 2. *R_B is maximum when $B = \{(j, 2b + 1 - j) \mid j \in \{1, 2, \ldots, b\}\}$.*

Proof: We prove this through induction on b. When $b = 1$, there are only 2 nodes, and only one possible pairing, so $B = \{(1, 2)\}$ trivially.

For the inductive hypothesis, assume that the lemma is true for $b = k$. For $b = k + 1$, we first prove that, for R_B to be maximum, $(1, 2k + 2) \in B$. Assume by contradiction that $(1, 2k + 2) \notin B$. This means that $\exists (1, i), (j, 2k + 2) \in B$ where $i, j \in \{2, \ldots, 2b - 1 = 2k + 1\}$. Similar to lemma 1, we will show that swapping the nodes in the two pairs to get B', where $(1, 2k + 2), (i, j) \in B'$, will improve system reliability. The contribution of pairs $(1, i), (j, 2k + 2)$ to R_B is given by $(1 - p_1 p_i)(1 - p_j p_{2k+2})$. We have

$$\begin{aligned} (1 - p_1 p_i)(1 - p_j p_{2k+2}) &= 1 - p_1 p_i - p_j p_{2k+2} + p_1 p_i p_j p_{2k+2} \\ &\leq 1 - p_1 p_{2k+2} - p_i p_j + p_1 p_i p_j p_{2k+2} \\ &= (1 - p_1 p_{2k+2})(1 - p_i p_j) \end{aligned} \tag{4}$$

The inequality on the second line is obtained by noting that $p_1 \leq p_j$ and $p_i \leq p_{2k+2}$. By rearrangement inequality[10], we know that $p_1 p_i + p_j p_{2b} \geq p_1 p_{2k+2} + p_i p_j$ which leads to the inequality obtained above. This means that for any B such that $(1, i), (j, 2k+2) \in B$, we can get $R_{B'} \geq R_B$ where $B' = (B - \{(1, i), (j, 2k + 2)\}) \cup \{(1, 2b), (i, j)\}$. Using the same argument as in lemma 1, we conclude that R_B is maximum when $(1, 2k + 2) \in B$. We can thus write the maximum R_B as $R_B = (1 - p_1 p_{2k+2}) R'_B$ where R'_B is the combined reliability of all node-replica pairs other than $(1, 2k + 2)$. R'_B can also be considered as the reliability of $2k$ nodes making k pairs which, according to the inductive assumption, is maximum when the $2k$ nodes are paired as stated in the lemma. The overall reliability, R_B, is therefore maximized when $B = \{(j, 2(k + 1) + 1 - j) \mid j \in \{1, 2, \ldots k + 1\}\}$ which concludes the proof. ∎

Lemma 1 and lemma 2 combined complete the proof of the theorem.

At this point, one may also wonder if a similar result can be obtained for replication degrees greater than 2, for example if triple replication is also allowed. In that case, the only result we can obtain is the following

Lemma 3. *If B contains replica groups with degrees ≥ 2, i.e. $x \in B \rightarrow |x| \geq 2$, R is still maximized when $A = \{1, 2, \ldots, a\}$.*

Proof: The proof proceeds by contradiction in the same way as lemma 1 by taking a tuple in B which has an element i where $i \leq a$, and similarly a $j \in A$ where $j > a$. It can then be shown that swapping i and j between the two sets causes R to increase. We omit the detailed steps since they are identical to that of lemma 1. ∎

The same result, however, does not extend to the case of deciding, for example, which nodes should be doubly replicated and which should be triply replicated. In this paper, we restrict our focus to partially redundant systems where nodes are at most doubly replicated.

It should be noted that, although the proof in this section is formulated in terms of node reliabilities, the result holds for any time interval in which the relative ordering of the individual nodes' likelihoods of failure is known. This means that if, at different time intervals, the ordering of nodes based on their likelihoods of failure changes, the optimal configuration, while still determined based on the result in this section, will be different during different time intervals. Handling such configuration changes in practical settings may be possible through an adaptive method to switch replicas on the fly, as in [11]. A theoretical analysis to determine when to change the configuration, taking into consideration the cost of reconfiguring the system during execution, is beyond the scope of the current work and is left for future work. In this paper, we will only consider cases where the nodes failure densities are exponential, or Weibull with the same shape parameter. In both of these cases, the relative ordering of node reliabilities remains the same throughout the lifetime and is determined from the individual node MTBFs.

III. Expected Completion Time

In the previous section, we looked at how the nodes should be grouped into replicas when the number of nodes to be replicated is fixed. In other words, the number of application visible nodes n was already decided, a and b were then determined from the equations $a+2b = N$ and $a+b = n$, and the goal was to intelligently pick nodes to be placed in sets A and B based on their individual reliability $g_1(t) \geq g_2(t) \geq \cdots \geq g_N(t)$. In that case, it made sense to look at system reliability alone, because the number of nodes to use and the partial replication factor was fixed. In the rest of this paper, however, we attempt

to answer the more general question: Given an N node system with node reliability $g_1(t) \geq g_2(t) \geq \cdots \geq g_N(t)$, how many of the N nodes should be used and what should be the optimal partial replication factor? This makes both a and b as variables to be determined since the equations relating them to n and N become the following inequalities: $a + 2b \leq N$ and $a + b = n \geq 1$. This question cannot be answered by considering system reliability alone. Although a higher value of n will reduce the work per node due to parallelism, system reliability will go down making failures more likely. On the other hand, higher replication factors are likely to add more runtime overhead to the application, although they lead to a more resilient configuration. These trade offs can only be captured by computing the expected completion time for given a and b, and then picking the values of these variables that yield the minimum completion time.

Parameter	Description
N	Total number of system nodes
n	Number of application visible nodes
a	Number of non replicated nodes
b	Number of replica pairs
r	Partial replication factor ($1 \leq r \leq 2$)
α	Communication ratio ($0 \leq \alpha \leq 1$)
γ	Serial portion of application code
W	Work duration on single node
W_n	Work duration on n parallel nodes
W_r	Work duration with replication factor r
M	Mean Time To Interrupt (MTTI)
C	Checkpointing cost
τ	Checkpointing interval

A. Job Model

The first thing to determine, as n becomes variable, is the amount of work that will be distributed over each node and executed in parallel. Assuming that a particular job on a single node takes W units of time to finish execution, we use the following job model[12] to determine, W_n ,the time required to execute the same job on n parallel nodes without failures:

$$W_n = (1 - \gamma)W/n + \gamma W \tag{5}$$

where $0 \leq \gamma \leq 1$ represents the sequential part of the job. Smaller values of γ indicate higher potential for parallelism in the application. Larger values of γ, on the other hand, offer diminishing returns with increasing parallelism. Since increasing the value of $r = (a + 2b)/(a + b)$, while keeping $a + 2b$ fixed, means reducing $n = a + b$, higher values of γ are more favorable towards replication. In this work, most of the analysis we perform and the results we report will be for values of γ equal to, or close to, 0. This is for two reasons: i) as explained above, lower values of γ are more favorable towards no replication versus replication, and ii) HPC jobs typically should be highly parallelizable, and so γ is small in those settings. One could also use more sophisticated job models, taking into account the kind of computation and/or domain decomposition that the application performs. However, any model that is less than perfectly parallel is more favorable towards replication. Thus, by focusing on the perfectly parallel job model, we can ensure that conclusions drawn from the cases in which replication (full or partial) performs better than no-replication can be generalized to other job models as well.

B. Overhead of Replication

In addition to reducing the nodes over which work is parallelized, increasing replication also increases the overhead to message passing applications because of the communication required between replicas in order maintain consistency among them. The replication system then guarantees that every replica receives the same messages in the same order and that a copy of each message from one rank is sent to each replica in the destination rank[2]. This requires duplicating the communication of a process to its replica as well. While [2] provided estimates of overhead of replication based on implementation on a real system, that work only applies to full replication. An approach to model the overhead of partial replication was proposed in [4] using α, the ratio of application time spent in communication under no replication. According to that model, for an application executing under partial replication factor r, the time, W_r, that includes the overhead of partial replication, is given by

$$W_r = (1 - \alpha)W_n + r\alpha W_n = W_n + (r - 1)\alpha W_n \tag{6}$$

where W_n is computed using Eq. 5. The rationale provided in [4] for this model is that every message involving a replicated node will be duplicated. Hence, the additional communication overhead will be linearly related to the replication factor, r. The experimental results in [4], though, showed Eq. 6 actually underestimated the overhead of partial replication. Similarly, [13] reported overheads with replication factor of 1.5 which, in some cases, went as high as 70% of the overhead of full replication ($r = 2$). Since this would indicate that the communication overhead of replication is not linear w.r.t the replication degree, which is the assumption for Eq. 6, we update it as

$$W_r = W_n + \sqrt{r - 1}\alpha W_n \tag{7}$$

This estimate, while yielding the same overhead for full replication as Eq. 6, provides a more pessimistic overhead for partial replication compared to Eq. 6. Moreover, it matches with the experimental result of [13] on real systems since, for $r = 1.5$, the overhead will be $1/\sqrt{2} \approx 0.71\%$ of the overhead of full replication. We, therefore, use Eq. 7 to compute and add the overhead of partial replication.

C. Combining with Checkpointing

Having figured out the failure-free execution time, W_r, of a partially replicated application, we now proceed to compute the expected completion time of such an application under failures. Since even a fully replicated system is subject to failures when both nodes that are replicas of each other fail, both [2] and [4] combined checkpointing with a fully or partially replicated system. However, the checkpointing interval would be larger compared to the no replication case since it depends on the mean time to interrupt (MTTI). The MTTI, M, can be computed using the reliability as:

$$M = \int_0^\infty R(t)dt \tag{8}$$

where $R(t)$ is given by Eq. 1. Although, in subsequent sections, we will discuss closed form approximations for the MTTI for some specific cases of systems with exponential node distributions, in general it is not possible to evaluate the

integral in Eq. 8 analytically. We, therefore, resort to numerical integration to obtain the MTTI for our results.

To compute the checkpointing interval, τ, we use Daly's[14] approximation to the optimum checkpointing interval, given a checkpointing cost, C and the MTTI M. This approximation was derived for exponential failure distributions. The failure distribution of a partially replicated system is not exponential even when the node failures are exponentially distributed[15]. However, [3] observed that Daly's approximation still yields results close to optimal even when the underlying failure distribution is not exponential. Hence, we use it as a reasonable approximation for the optimal checkpointing interval.

In order to determine the expected completion time, we employ the approach used in [16] and [17] that approximates the completion time of a system with a generic failure distribution by computing the extra work done in an execution. The extra work during an execution consists of the time spent writing checkpoints and the lost work due to the failures. By considering each failure as a renewal process, the average fraction of extra time during an execution can be taken to be the same as the fraction of extra time between two consecutive failures. Let $F(t) = 1 - R(t)$ be the cumulative failure distribution and $f(t) = F'(t)$ be the failure density function. The extra time between consecutive failures will be given by

$$E(extra) = \int_0^\infty \frac{Ct}{\tau} f(t)dt + k\tau = \frac{C * M}{\tau} + k\tau \quad (9)$$

where k is the average fraction of work lost during a checkpointing interval due to failure. To compute k, note that $k\tau$ is equal to the expected time of failure within an interval of length τ. If we divide the time into segments of length τ, where segment $t_i = [\tau(i-1), \tau i)$, and $i > 0$ is an integer, we can compute the expected time of failure, f_i, within segment t_i as $f_i = \int_{\tau(i-1)}^{\tau i} tf(t)dt/p_i$ where $p_i = \int_{\tau(i-1)}^{\tau i} f(t)dt$ is the probability of failure striking in interval t_i. The average value of the failure within a τ interval will then be given by $\sum_{i=1}^{\infty}(f_i - \tau(i-1))p_i$ which we can divide by τ to obtain k. Since p_i approaches 0 as i increases, the value of this sum converges quickly, so a simple summation of the first few terms suffices. Even though the sum converges much earlier, for increased accuracy, we used the first 20 terms to compute the sum and obtain the value of k.

Once we obtain $E(extra)$, the useful work done between consecutive failures is given by $M - E(extra)$, where M is the MTTI. On average, therefore, the time it takes the system to complete a unit amount of work will be equal to $M/(M - E(extra))$. Thus, the expected time it takes to finish work W_r will be given by

$$E(W_r) = W_r \frac{M}{M - E(extra)} \quad (10)$$

where W_r is determined from Eq. 7. This is the equation that we use to compute and compare the expected completion time of a system under different partial replication factors.

D. Optimization Problem

The purpose of computing the expected completion time was to find the replication factor r that minimizes it for a given system. We thus formulate the search for r as an optimization problem as follows:

$$\begin{aligned} \underset{a,b}{\text{minimize}} \quad & E(W_r) \\ \text{subject to} \quad & a + 2b \leq N \\ & n = a + b \geq 1 \end{aligned}$$

where $r = (a + 2b)/(a + b)$ and a and b can only take nonnegative integer values. The inputs include work W, total number of system nodes N, individual node reliability functions $1 > g_1(t) \geq \cdots \geq g_N(t) > 0$, checkpointing cost C, the parameter γ, and communication ratio, α. In subsequent sections, we will discuss our findings and results about the optimal r for different kinds of systems. In our results, we report the expected completion time normalized by W_N, which is calculated from Eq. 5, and represents the time it takes to execute the job on all N nodes without replication or checkpointing and under no failures.

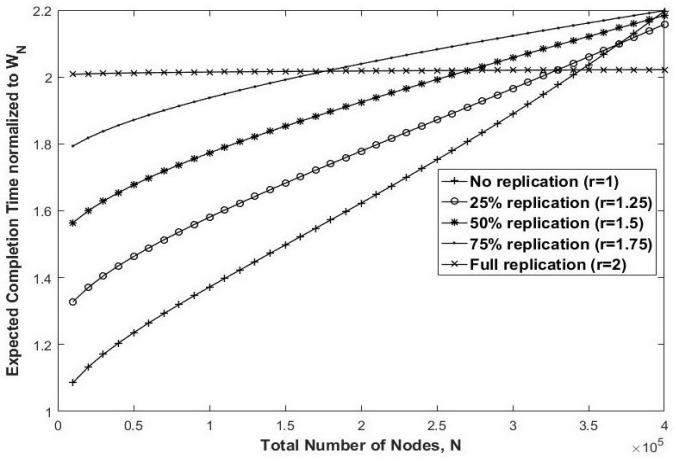

Fig. 2. Expected Completion Time for different values of r normalized by the failure-free time it takes to finish the same work on all N nodes without replication or checkpointing. Node MTBF = 5 years. Checkpointing cost is taken to be 60 seconds. $\alpha = 0$ and also $\gamma = 0$.

IV. System with IID Nodes

We start off with the simplest possible scenario, a system where the individual node failure distributions are identical. This has been the traditional assumption when analyzing fault tolerance techniques in HPC systems. Our goal is to explore whether there are cases in which partial replication, where r is strictly between 1 and 2, results in the lowest expected completion time. It should be noted that, since all nodes are identical, it does not matter which individual nodes are picked for replication or how they are paired together.

A. Exponential Distribution

We first consider a system where node failure probabilities are exponentially distributed. When taking both γ and α as 0, our optimization never yielded an optimal value of r strictly between 1 and 2 for any scenario we tested. This can also be seen in figure 2 where the expected completion time according to Eq. 10 (normalized by the time it takes to run the same job on N nodes without any fault tolerance and without failures) with different partial replication degrees is plotted against the

total number of nodes in the system. We see that the minimum time is always attained either when $r = 1$ or $r = 2$. The trend was the same for other node MTBF values, with the crossover between full and no replication occurring at higher node counts as the node MTBF increases.

We further investigate this scenario analytically with the goal of determining if $1 < r < 2$ is ever optimal for uniformly exponential node distributions when γ and α are both 0. Assuming that the configuration uses all of the system nodes N, so that $a + 2b = N$, and individual node failure rate is λ, we can write the MTTI, M, as:

$$\begin{aligned} M &= \int_0^\infty e^{-a\lambda t}(1-(1-e^{-\lambda t})^2)^b dt \\ &= 2^N \int_0^\infty (e^{-\lambda t}/2)^{N-b}(1-e^{-\lambda t}/2)^b dt \end{aligned} \tag{11}$$

Since obtaining a closed form expression for the above integral is not possible, we try to provide a closed form approximation for M. Setting $x = e^{-\lambda t}/2 \Leftrightarrow t = -ln(2x)/\lambda$ in the above expression, we get

$$M = \frac{2^N}{\lambda} \int_0^{1/2} x^{(N-b-1)}(1-x)^b dx \tag{12}$$

We employ Laplace's method of approximating integrals[18] to derive an approximation of the above expression. We can rewrite the function inside the integral as $x^{(N-b-1)}(1-x)^b = e^{(N-b-1)f(x)}$ where $f(x) = ln(x) + bln(1-x)/(k-b-1)$. Assuming $2b < N$, within the interval of integration $f(x)$ is maximum at $x = 1/2$ which is the endpoint of the integration, so the integral can be approximated as

$$\begin{aligned} \int_0^{1/2} x^{(N-b-1)}(1-x)^b dx &\approx \frac{e^{(N-b-1)f(1/2)}}{(N-b-1)f'(1/2)} \\ &= \frac{(1/2)^N}{N-2b-1} \end{aligned} \tag{13}$$

Plugging this into the expression for MTTI above we obtain $M \approx 1/\lambda(N-2b-1)$. This reasonably approximates the MTTI as long as $2b$ is not close to N, which corresponds to the full replication case. To the best of our knowledge, this is the first closed form approximation of the MTTI of a partially replicated system with exponential node failure distributions with rate λ.

Having obtained a closed form approximation for M in terms of N and b, we will infer the behavior of the expected completion time. Using Young's[19] expression for the expected completion time we get

$$E(W) = W(1 + \frac{C}{\tau} + \frac{(\tau + C)^2}{2\tau M}) \tag{14}$$

where we take $\tau = \sqrt{2CM}$ which is also Young's approximation for the optimum checkpoint interval. Assuming that a perfectly parallel job takes unit time on N nodes without checkpoints and failures, the work per node will be r units when the system is partially replicated, since $r = (a+2b)/(a+b) = N/n$. This means that $W_r = r$, and $E(W_r)$ then is given by

$$E(W_r) = r(1 + \sqrt{\frac{2C}{M}} + \frac{C}{M} + \frac{C^2}{2M\sqrt{2CM}}) \tag{15}$$

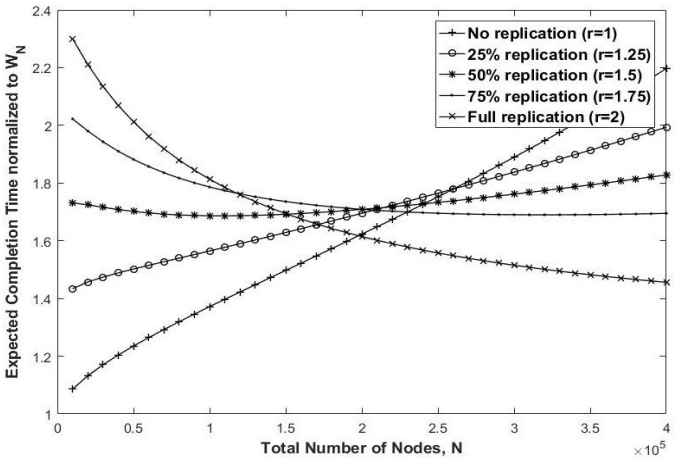

Fig. 3. Expected Completion Time for different values of r for exponential node distribution. Node MTBF = 5 years, $\gamma = 0.00001$, Checkpointing cost = 60 seconds, $\alpha = 0.2$

Since both r and M can be defined in terms of b and N, and since N is fixed, $E(W_r)$ is a function of b. By taking the first and second derivative of this expression *wrt* b, we find that this expression has no local minimum over the range $0 \leq b < N/2$ as long as $M > C$. For conciseness, we omit the calculations. This means, though, that the minimum of $E(W_r)$ occurs only at one of the endpoints of r which correspond to either no replication or full replication. While eq. 15 is an approximation for the expected completion time, this analysis supports our numerical results **that partial replication never yields optimal performance for jobs with $\alpha = \gamma = 0$ and when individual node failures are iid with exponential distributions**. This is also consistent with the findings of [15] where it was observed that in cases where replication is better than no replication, full replication offers the best performance.

When $\alpha > 0$, it may theoretically be possible to have cases where the optimal r is strictly between 1 and 2. This is because there can be cases in which the expected completion time with $\alpha = 0$ is minimized when $r = 2$, but that minimum may shift to $r < 2$ if $\alpha > 0$. That being said, we did not observe this for any values of parameters that we tried. As for when $\gamma > 0$, although it may be possible for $1 < r < 2$ to be optimal, we again did not observe any such case. Figure 3 shows one example with both $\alpha > 0$ and $\gamma > 0$. We observe that, although the crossover between full and no replication happens earlier compared to Fig. 2, partial replication again does not win against the two extremes. Hence, our conclusion from this subsection is that partial replication is almost always never optimal for systems with iid exponential node distributions.

B. Weibull Distribution

Fig. 4 shows the completion times when individual node failures are given by the Weibull distribution. In practice, values of the shape parameter between 0 and 1 are used for real world failures. In this paper, we show results with the parameter value of 0.7. Similar trends were observed with value of 0.5, but are omitted due to space limitations. In comparison to Fig. 2, we see that the crossover between full and no replication happens much earlier. Additionally, there are node counts where partial replication with degrees 1.25 and 1.5

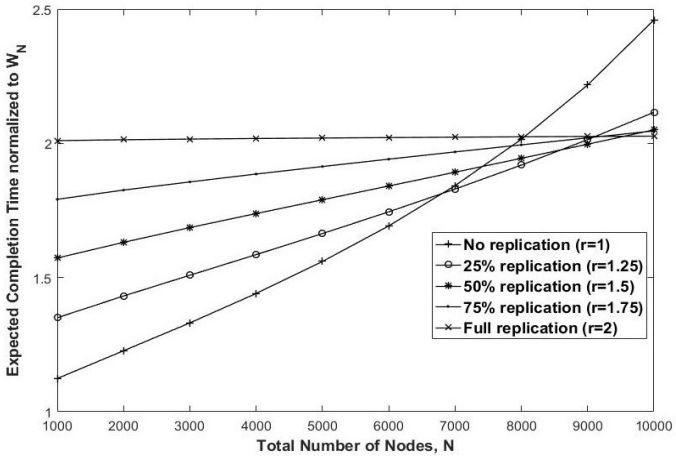

Fig. 4. Expected Completion Time for different values of r with Weibull node failures. For the distribution, shape parameter = 0.7 and MTBF = 5 years. Checkpointing cost = 60 seconds and $\alpha = \gamma = 0$.

have lowest completion times. The range of node counts for which this happens is still quite small, however. The behavior remains almost similar when α or γ are made > 0, except that the crossover points are shifted. Increasing α shifts the crossover points to the right. For example, with $\alpha = 0.2$, the crossover between no replication and $r = 1.25$ happens around 9000 nodes instead of 7000 nodes. Increasing γ, on the other hand, brings the crossover points to the left towards smaller node counts. For example, $\gamma = 10^{-5}$ causes the crossover between $r = 1$ and $r = 1.25$ to happen at 6500 nodes instead of 7000 nodes. Moreover, just like in Figure 4, there is only a very small range of node counts for which partial replication provides the lowest completion time.

Our main takeaway point from this section is that when the nodes in the system have identical failure distributions, which has been the traditional assumption in fault tolerance research for HPC, partial replication rarely provides any gains in performance against full and no replication. Depending on the number of nodes in the system, the choice should then only be between running an application under full replication or running it with no replication at all.

V. System with Two Types of Nodes

We now move one step further by considering a system where nodes are of two kinds: i) Good, which have a low probability of failure, and ii) Bad, which have a higher probability of failure. We assume that all the Good nodes have the same failure distribution and all the Bad nodes have the same failure distribution. This can be a scenario in a system where individual system nodes can be approximately divided into two categories: those which are more prone to failures and those which are less prone to failures.

Let N_G be the number of Good nodes and N_B be the number of bad nodes, such that $N_G + N_B = N$. Thanks to the main result of section II, we know that if partial replication is to be employed, we should start replicating from the lower end. Moreover, within the nodes to be replicated, pairing should be done as indicated by Figure 1. Using this knowledge, we can enumerate all possible cases for different partial replication

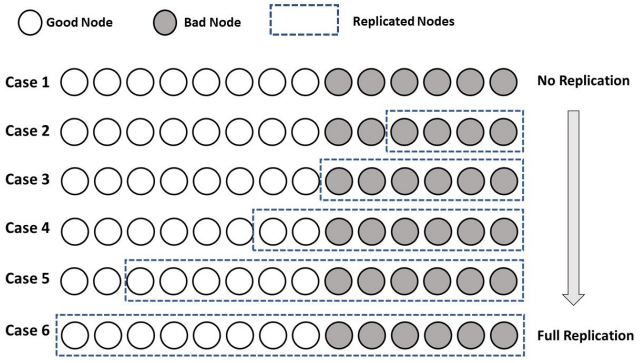

Fig. 5. Possible cases of partial replication for system with Good and Bad nodes. Nodes within the replicated set are paired according to the arrangement depicted in Figure 1. For this figure, the number of Good nodes is taken to be higher than the number of Bad nodes. If the number of Good nodes is strictly lower than the number of Bad nodes, cases 5 and 6 above will not happen.

degrees of a Good-Bad node system. This enumeration is depicted in Fig. 5. Starting from the no replication case, increasing replication degree would mean initially replicating the Bad nodes among themselves. Case 3 is the boundary of case 2, when all of the Bad nodes have been replicated. As the replication degree is further increased, some of the Good nodes enter the replicated set as well. Case 4 thus contains two kinds of replica pairs: a Good node paired with a Bad node, and a Bad node paired with a Bad node. Case 5 is again a boundary of case 4 where all replica pairs consist of a Good and a Bad node each. The full replication case contains additional node pairs depending on the difference between the number of Good and Bad nodes.

We will explore how the average completion times of these different cases fare against each other in different settings. Such an analysis can be useful for system administrators in deciding the optimal replication scheme that will result in the lowest job completion time on average, based on information about system nodes and other parameters.

A. Exponential Distribution

Assuming all the nodes in the system have exponential failure distribution, we can take the failure rate of Good nodes as λ_g and the failure rate of the Bad nodes as λ_b, where $\lambda_g \leq \lambda_b$. Since case 2 in Fig. 5 is quite similar to the partially replicated iid system in section IV, we first attempt to approximate its MTTI. For this case, we can write the reliability of the system as

$$R(t) = e^{-N_G\lambda_g t} e^{-(N_B-2b)\lambda_b t} \left(2e^{-\lambda_b t} - e^{-2\lambda_b t}\right)^b \qquad (16)$$

where $2b$ is the number of Bad nodes that are replicated. To obtain the MTTI of such a system, we can follow the same approach as in section IV to approximate the integral of $R(t)$. This yields the following approximation for the MTTI, M, of the system in case 2

$$M \approx \frac{1}{N_G\lambda_g + (N_B - 2b - 1)\lambda_b} \qquad (17)$$

This expression again reasonably approximates the MTTI as long as $2b$ is not close to N_B.

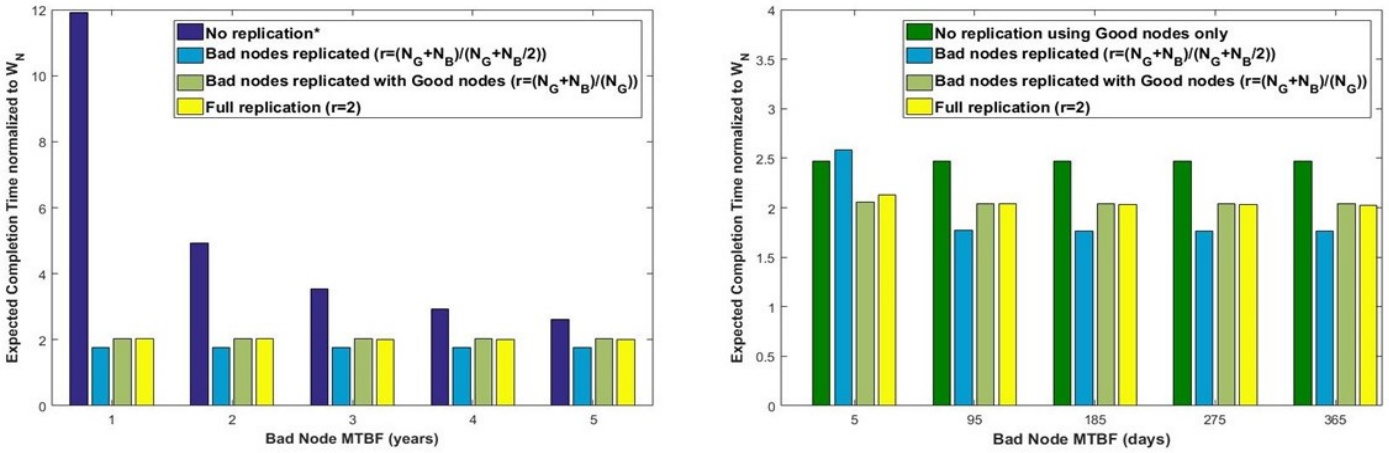

Fig. 6. Execution time of different partially replicated executions*. $N_G = 10^6$, $N_B = 8 \times 10^5$, $\lambda_g = 1/50$ years, $C = 60$ seconds and $\alpha = \gamma = 0$.
* Using Eq. 10 for the No Replication case in the left figure resulted in negative values when Bad node MTBF was low, because $E(extra)$ became greater than MTTI in those cases. We instead used Daly's[14] expression to approximate the expected completion time for No replication, which usually provides a lower value than Eq. 10. We do this only for the no replication case in the plot on the left. Expected times for the other schemes are still computed using Eq. 10.

Similar to section IV, we use eq. 17 to understand the behavior of the expected completion time of the application *wrt* b when $\alpha = \gamma = 0$. We plug M into eq. 15 along with r for this case, which is equal to $(N_G+N_B)/(N_G+N_B-b)$. Taking the first and second derivatives of the resulting expression *wrt* b, we again conclude that the function has no local minima and thus the minimum occurs only at the extremes, i.e. $b = 0$ (no replication), or $b = N_B/2$ (all Bad nodes replicated among themselves). This indicates that, between cases 1, 2 and 3, the minimum expected time can only be achieved by cases 1 and 3 for exponential node failures with $\alpha = \gamma = 0$. We again mention that while this derivation holds only for the approximations of M and expected completion time, our numerical search also never yielded any scenarios in which case 2 resulted in lower average time than both cases 1 and 3.

While we were unable to obtain an approximation of MTTI for case 4, our numerical search indicates that the minimum average completion time occurs again at the boundary cases, i.e. 3 or 5. This means that, in general, we need only consider the boundaries of partial replication in a Good-Bad node system. As an example, Figure 6 shows the expected completion time of full and no replication along with cases 3 and 5 from Figure 5. From the plot on the left in Figure 6, we see that replicating the Bad nodes among themselves (Case 3) yields the lowest completion time. Case 5, which replicates each Bad node with a Good node, offered almost the same performance as full replication. While we do not show the results with higher Bad node MTBF, we saw that no-replication started outperforming Case 3 when Bad node MTBF went above 20 years, with the same parameters as in Figure 6.

In order to find out if there can be a scenario where Bad node MTBF is so low that not using the Bad nodes, replicated or not, at all is the best performing scheme, we reduced the Bad node MTBF to the order of days and also compare with a no replicated configuration using the Good nodes only ($a = N_G$, $b = 0$). The plot on the right in figure 6 depicts the results. We see that only in the unrealistic case of individual Bad node MTBF dropping to the order of a few day does using Good nodes only outperform Case 3. We deduce from this that, as long as Bad node MTBF is larger than a few days, utilizing the Bad nodes results in lower completion time on average instead of not using them at all. Whenever Bad node MTBF is so low that using them without replication hurts application runtime, the lowest expected time can be achieved by replicating the Bad nodes among themselves and still utilizing them along with the non-replicated Good nodes.

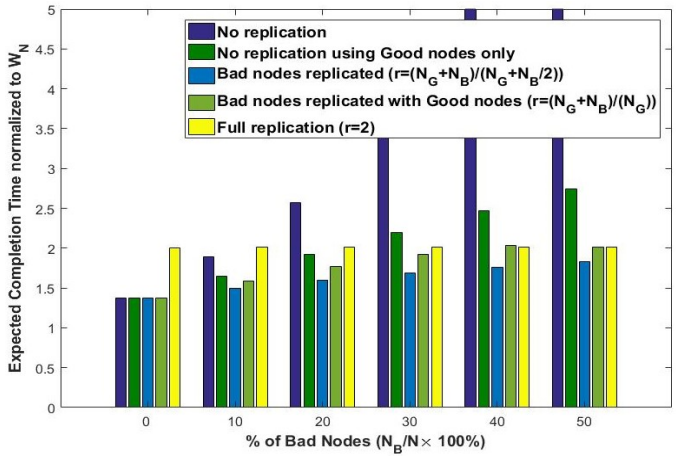

Fig. 7. Expected time vs % of Bad Nodes in the system. $N = 2 \times 10^6$. Bad Node MTBF = 5 years. Other parameters are the same as in Fig 6.

Figure 7 shows the behavior of the schemes with varying percentage of Bad nodes in the system, while the total number of nodes, N, is kept constant. When all nodes are Good, no replication is the best choice. However, as further nodes are added, no replication has a much higher normalized time. The normalized time for the no replication scheme which uses the Good nodes only also increases as % of N_B in the system increases. This is because the time is normalized by W_N which is the failure free time of running the job on all N system nodes. In all cases, however, we see that Case 3 offers the best expected completion time.

Figure 8 shows the behavior of the different partial repli-

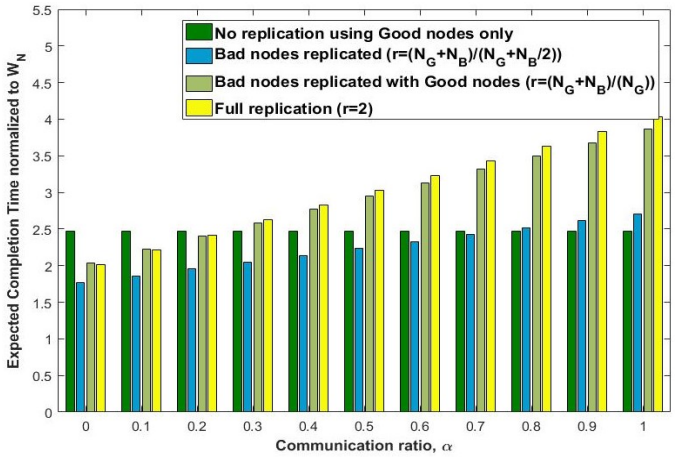

Fig. 8. Expected time for different values of α when Bad Node MTBF = 5 years. Other parameters are the same as in Figure 6. The expected time for no replication using all system nodes is much higher than all other schemes so it is omitted from the plot.

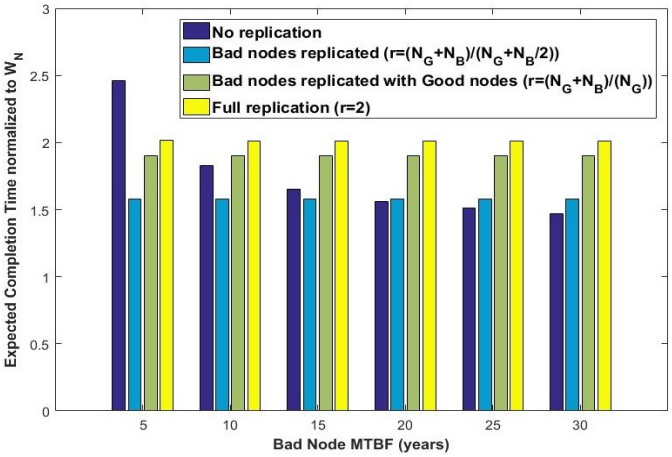

Fig. 9. Execution time of different replication schemes with Weibull node failures. $N_G = 10^4$, $N_B = 8 \times 10^3$ and Good node MTBF = 50 years. The other parameters are the same as in Fig. 6.

cation schemes for different values of α. The time for all the partial replication schemes increases with increasing α. However, since Case 3 has smaller replication factor than Cases 5 and 6, the impact of α is much smaller. Only when $\alpha \geq 0.8$ does partial replication of Case 3 start losing to no replication using Good nodes only. Hence, we can say that for most practical values of α, using the Bad nodes with full replication amongst themselves is still better than not using them at all. We do not present similar plots for the parameter γ due to lack of space. The impact of increasing γ is to favor more the cases with higher replication factor, r. Hence, as γ increases, the lowest completion time shifts from case 3 towards full replication ($r = 2$).

B. Weibull Distribution

For node failures given by the Weibull distribution, we assume that all nodes' distribution have the same shape parameter. Only the rate parameter, λ, is different for the Good and Bad nodes. With this assumption, and again taking $\lambda_g \leq \lambda_b$, the Good node will always be more reliable than the Bad node

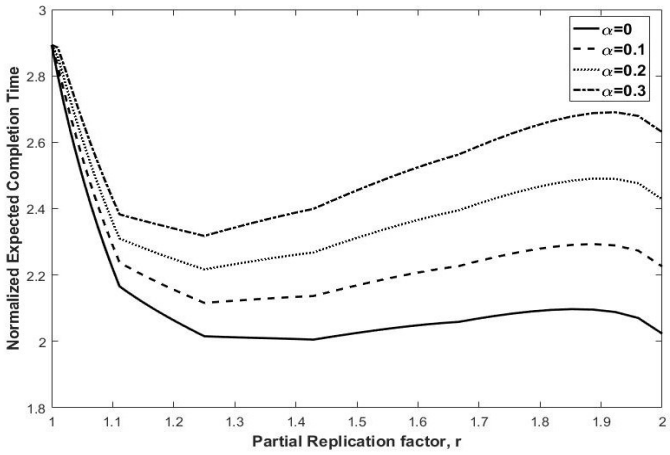

Fig. 10. Expected completion time versus r for different values of α. The values of other parameters are: $\gamma = 0$, $C = 30$ seconds and each category contains 100k nodes, for a total of 500k system nodes.

throughout its lifetime. Hence, this assumption allows us to apply the theorem of section II when deciding the pairing of nodes and so the possible partial replication schemes will still be given by Fig. 5.

Figure 9 shows the normalized runtimes of different partial replication cases similar to the exponential distribution subsection, but over a larger range of Bad node MTBFs. We again see that with lower Bad node MTBF, replicating Bad nodes among themselves yields the lowest expected completion time. Moreover, this happens at system scales much smaller than the ones for exponential distribution. We omit the plots for the cases when $\alpha > 0$. The trends, however, were the same as the ones observed for exponential distribution.

Based on the results from both exponential and Weibull distributions, we conclude this section with the following insight: If an HPC system has some nodes that are more likely to fail than others, those nodes can still be utilized to achieve performance gains. When the likelihood of failures in such Bad nodes is not too high, those nodes can simply be used alongside the rest of the system nodes to execute a job in parallel, without replication. If, however, the likelihood of failures in those nodes increases, they can be replicated among themselves and still be used along with the other system nodes to provide better performance compared to the case of not using such nodes at all.

VI. Systems beyond two categories of nodes

The optimization problem formulated in section III is capable of finding the optimal r for a system with any set of non-uniform node reliability values $g_i(t)$, as long as they maintain the ordering $g_1(t) \geq g_2(t) \geq \cdots \geq g_N(t)$. This is useful when all the individual node reliability functions are known. However, we do not present any results for such a generic system because they don't provide any interesting insights about r or the behavior of the expected completion time. We, therefore, only present one example of a system with 5 categories of nodes. The MTBFs of the five categories range from 1 to 5 years in increments of 1 year, with each category having the same number of nodes. Figure 10 shows

the normalized expected completion time, using Eq 10, versus the partial replication factor, r, for different values of α. We kept $a + 2b = N$ instead of the inequality $a + 2b \leq N$. This is because, similar to the conclusions for the Good-Bad node system, we usually found that using all the nodes in the system is beneficial, as long as the lowest MTBF nodes do not have unrealistically low values of the MTBF.

We can make several observations from Fig. 10. For all values of α, the optimal r is less than 2. For $\alpha = 0$, optimal value of $r \approx 1.42$, but for other values of α, the optimal value of $r = 1.25$. These results highlight the importance of having and utilizing a deeper understanding of the failure characteristics of the underlying system. If, for example, instead of considering the 5 categories of nodes, one took the average value of the node MTBF across 5 categories as 2.5 years, and used that to decide the replication degree, the answer would be to fully replicate the execution. However, as we can see in the figure, partially replicating the right nodes can result in lower expected completion time than full replication. In fact, if the decision to fully replicate is made without the knowledge of the different categories of nodes, the replica pairing may not be the same as that described in section II, and may lead to even higher expected completion time.

We make one final remark about the behavior versus r. We see in Fig. 10 that the curve is piecewise smooth in segments. The values of r at the boundary points of these segments correspond to the boundary cases of different partial replication configurations. So, for example, if only the nodes in the lowest MTBF category are all replicated among themselves, we get $r = 1.1$. We see in Fig. 10 that for $1 \leq r \leq 1.1$, the curve is smooth. Similarly, the next smooth segment finishes at $r = 1.25$, which is the boundary case achieved when the lowest MTBF category is fully replicated with the next lowest category. Although we do not have any analytical results about this, our investigations of multiple scenarios always yielded the optimal r on one of these boundary cases. This indicates that, in cases where node MTBFs take a small set of discrete values, rather than doing a full search for the best r, it may be a reasonable heuristic to only consider boundary cases and pick r with the lowest completion time.

VII. Related Work

Full[2] and partial[4] replication were both proposed for large scale systems when failures become frequent. A deeper analysis of pure replication and its comparison with simple checkpoint/restart was carried out in [3]. For partial replication, [15] provides a limited analysis and comparison with full and no replication. Even though our focus in this work is on systems with non-uniform failure distributions of individual nodes, section IV provides a more detailed analysis of partial replication with iid node failures. We provide theoretical results for the MTTI and evidence that partial replication is never optimal on such systems.

All of the above have assumed systems with identical nodes in their analyses. We are only aware of two works that distinguish between different failure likelihoods in the underlying hardware. [20] considers two instances of an application running on two different platforms, that execute at different speeds and are subject to different failure rates. Our work differs from it in several aspects. Firstly, the paper considers *group replication*, where a complete instance of the parallel application is executed redundantly, rather than replicating individual processes. This avoids communication between instances but a single failure causes the whole instance to fail. Secondly, the framework does not allow for partial replication. Thirdly, their work assumes a single platform failure distribution, without considering the underlying nodes in the system. [21] is closer to our work since it considers individual node failure rates. However, it only performs a post hoc analysis based on failure logs to determine which nodes have the most failures and how many of those failures could be eliminated by duplicating those nodes with spare nodes. Moreover, this work only considers the improvement in MTTI without looking at the impact on completion time. Our work provides a comprehensive theoretical framework which not only determines how the nodes should be duplicated, but also when it pays off to duplicate some nodes in the system.

While our work looks at partial redundancy in the presence of non-identical node failures, there are papers that consider the problem of selectively replicating tasks based on criticality[22][23][24]. These works replicate tasks from an application task dependence graph by measuring the criticality of an individual task. The idea of criticality is orthogonal to our task of selectively replicating *nodes* based on their individual reliability. Our work, additionally, is application agnostic since it only considers the failure distributions of individual nodes.

VIII. Conclusion

We explored partial replication for HPC systems where individual nodes have non-identical failure distributions. We provided theoretical results on the optimal way to divide the nodes into replicated and non replicated sets and to pair the nodes in the replicated sets. By computing the MTTI and expected completion time of a job executed in a partially replicated configuration, we also investigated the optimal fraction of replication. We found that, while rarely optimal for IID node failure platforms, partial replication can yield the best performance for systems comprising of nodes with different failure rates.

One direction of future work is to explore the energy/performance trade-off of partial replication. While our work has demonstrated that partial replication for systems with non-identical node failures can often provide the best performance, it should be explored if that performance gain is worth spending the extra energy on the replicated nodes. Another direction of future work is to consider a mix of jobs instead of a single job. Solving for multiple jobs will require changing the metric from completion time to average job turnaround time and will also expand the search space to include options such as starting jobs in parallel over divided resources or running them one after the other.

Acknowledgment

We are thankful to reviewers for their constructive feedback that has helped us improve the quality of this paper. This research is based in part upon work supported by the Department of Energy under contract DE-SC0014376. This research was supported in part by the University of Pittsburgh Center for Research Computing through the resources provided.

REFERENCES

[1] F. Cappello, "Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities," *The International Journal of High Performance Computing Applications*, vol. 23, no. 3, pp. 212–226, 2009.

[2] K. Ferreira, J. Stearley, J. H. 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 *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*. IEEE, 2011, pp. 1–12.

[3] H. Casanova, Y. Robert, F. Vivien, and D. Zaidouni, "Combining process replication and checkpointing for resilience on exascale systems," Ph.D. dissertation, INRIA, 2012.

[4] J. Elliott, K. Kharbas, D. Fiala, F. Mueller, K. Ferreira, and C. Engelmann, "Combining partial redundancy and checkpointing for hpc," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 615–626.

[5] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how hpc systems fail," in *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*. IEEE, 2013, pp. 1–12.

[6] S. Gupta, D. Tiwari, C. Jantzi, J. Rogers, and D. Maxwell, "Understanding and exploiting spatial properties of system failures on extreme-scale hpc systems," in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 2015, pp. 37–44.

[7] S. Gupta, T. Patel, C. Engelmann, and D. Tiwari, "Failures in large scale systems: long-term measurement, analysis, and implications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017, p. 44.

[8] S. Di, R. Gupta, M. Snir, E. Pershey, and F. Cappello, "Logaider: A tool for mining potential correlations of hpc log events," in *Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on*. IEEE, 2017, pp. 442–451.

[9] T. J. Hacker, F. Romero, and C. D. Carothers, "An analysis of clustered failures on large supercomputing systems," *Journal of Parallel and Distributed Computing*, vol. 69, no. 7, pp. 652–665, 2009.

[10] Wikipedia, "Rearrangement inequality." [Online]. Available: https://en.wikipedia.org/wiki/Rearrangement_inequality

[11] C. George and S. Vadhiyar, "Fault tolerance on large scale systems using adaptive process replication," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2213–2225, 2015.

[12] M. Bougeret, H. Casanova, M. Rabie, Y. Robert, and F. Vivien, "Checkpointing strategies for parallel jobs," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, p. 33.

[13] C. Engelmann and S. Böhm, "Redundant execution of hpc applications with mr-mpi," in *Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN)*, 2011, pp. 15–17.

[14] J. T. Daly, "A higher order estimate of the optimum checkpoint interval for restart dumps," *Future generation computer systems*, vol. 22, no. 3, pp. 303–312, 2006.

[15] J. Stearley, K. Ferreira, D. Robinson, J. Laros, K. Pedretti, D. Arnold, P. Bridges, and R. Riesen, "Does partial replication pay off?" in *Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on*. IEEE, 2012, pp. 1–6.

[16] Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Paun, and S. L. Scott, "An optimal checkpoint/restart model for a large scale high performance computing system," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–9.

[17] O. Subasi, G. Kestor, and S. Krishnamoorthy, "Toward a general theory of optimal checkpoint placement," in *Cluster Computing (CLUSTER), 2017 IEEE International Conference on*. IEEE, 2017, pp. 464–474.

[18] Wikipedia, "Laplace's method." [Online]. Available: https://en.wikipedia.org/wiki/Laplace's_method

[19] J. W. Young, "A first order approximation to the optimum checkpoint interval," *Communications of the ACM*, vol. 17, no. 9, pp. 530–531, 1974.

[20] A. Benoit, A. Cavelan, V. Le Fèvre, and Y. Robert, "Optimal checkpointing period with replicated execution on heterogeneous platforms," Ph.D. dissertation, INRIA, 2017.

[21] N. Nakka and A. Choudhary, "Failure data-driven selective node-level duplication to improve mttf in high performance computing systems," in *High Performance Computing Systems and Applications*. Springer, 2010, pp. 304–322.

[22] O. Subasi, O. Unsal, and S. Krishnamoorthy, "Automatic risk-based selective redundancy for fault-tolerant task-parallel hpc applications," in *Proceedings of the Third International Workshop on Extreme Scale Programming Models and Middleware*. ACM, 2017, p. 2.

[23] O. Subasi, G. Yalcin, F. Zyulkyarov, O. Unsal, and J. Labarta, "A runtime heuristic to selectively replicate tasks for application-specific reliability targets," in *Cluster Computing (CLUSTER), 2016 IEEE International Conference on*. IEEE, 2016, pp. 498–505.

[24] ——, "Designing and modelling selective replication for fault-tolerant hpc applications," in *Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on*. IEEE, 2017, pp. 452–457.