Selective Process Replication for Fault Tolerance in Large-scale, Heterogeneous Environments with Non-Uniform Node Failure Distribution

Longhao Li
Department of Computer Science
University of Pittsburgh
Pittsburgh, USA
Email: lol16@cs.pitt.edu

Taieb Znati
Department of Computer Science
University of Pittsburgh
Pittsburgh, USA
Email: znati@cs.pitt.edu

Rami Melhem
Department of Computer Science
University of Pittsburgh
Pittsburgh, USA
Email: melhem@cs.pitt.edu

***Abstract*—Future systems are scaling to a large number of cores. Consequently, their propensity to failure increases dramatically, making it more challenging to achieve forward progress for compute-intensive applications on a large number of cores. Pure process replication is a widely accepted technique to tolerate fail-stop errors. At extreme-scale, however, it is inadequate to achieve fault tolerance efficiently due to doubled or even tripled computational resources usage. In this paper, we propose a selective process replication model that only assigns replicas to failure-prone processes. It assumes cores fail independently, but non-identically. The simulation results show that, on average, selective replication reduces more than 35 percent of energy consumption and more than 25 percent of the time to completion comparing to full replication with 1 million cores, where 20 percent of them are failure-prone.**

Keywords–fault-tolerance; selective replication; cloud computing; heterogeneous environment.

I. INTRODUCTION

Cloud computing has been widely used as a computing platform for resource-intensive applications like *MapReduce*, which require massive data analysis [1]. Meanwhile, computing and information systems have become integral to all aspects of our society, and their significance will inevitably increase in the future. The demand of cloud computing as back-end support of these systems is growing. Exploiting the convenience of on-demand computing power and flexible, low-cost resources is the key to success. With the continuous increase in computational scale, the ability to support massive parallelism is required for the cloud computing platforms of the future.

Massive parallelism is supported by large scale systems that contain millions of computational cores. Such systems are plagued with massive energy consumption and high system failure rate. Even with the expected technology improvement, the rate of system level failures will dramatically increase with the number of computational cores increase. For example, a computing infrastructure with 200,000 cores will experience a Mean Time Between Failure (MTBF) of less than one hour, even when the MTBF of an individual core is as large as 5 years [2], [3], [4], [5].

MapReduce is a popular data processing computational model which is widely used in Cloud computing platforms. Applications based on MapReduce divide large amount of data into subloads and distribute them to small tasks that can execute in parallel and independently. The computational results will merge together after all the subtasks complete execution and produce the final results. Thus, a single failure on any of the sub-tasks may lead to delay time to completion. Also, applications may contain multiple stages of MapReduce computation. Multiple failures on different subtasks may lead to an unacceptable delay in response time. This will likely cause unpleasant experience for customers and lead to revenue reduction.

A Service Level Agreement (SLA) is a contract between a customer and a Cloud Service Provider (CSP). It specifies the response time requirement of the customer. Violations on the agreement would lead to a penalty, which would further result in revenue reduction. Also, longer completion time may cause additional energy consumption of task re-execution and poor utilization of resources. To ensure that the tasks can be accomplished before the agreed deadline, a certain level of reliability is necessary.

In order to maintain system reliability, a resilience strategy is required in such large-scale systems. Process replication fault tolerance strategy relies on redundancy in resources by replicating the entire process. The original and replicated process runs in parallel on different hardware so that if one of the processes failed, other processes would finish the task on time. Comparatively, it is unlikely that the main and replicated processes failed at the same time. Process replication requires additional energy consumption due to the replications of the same process. In large scale, however, the additional energy consumption is significant enough that need to be reduced. Thus, full replication is not suitable for the future large scale systems.

Furthermore, most studies in fault tolerance assume that failures occur independently and identically. However, due to aging and replacement, cores may appear to have different failure rates. New cores may appear to have a comparatively higher failure rate due to manufacturing defects, and old cores also have a high failure rate because the hardware is worn out. Other factors can also lead to difference in failure rates, such as working environment temperature or nodes' usage [6].

In this paper, we propose a selective replication framework for cloud computing that only replicate processes on unreliable cores. In order to minimize the energy consumption, the proposed framework only protect the cores which are prone to failure. The main contributions of this paper are the following:

- A selective replication framework for cloud computing

 ISBN: 978-1-61208-770-2

resilience that select processes on unreliable cores based on the age of the hardware.

- A simulation based experimentation and evaluation which contains sensitivity analysis of different workloads and different ratio of unreliable cores.

The rest of the paper is introduced as follows. We discuss about the related works in Section II and explain the aspects leading to heterogeneous environment in Section III. We introduce the selective replication framework in Section IV. In Section V, we discuss about the simulation setup and the evaluation results. Section VI concludes the paper and discusses about some possible future works.

II. Related Works

The field of fault tolerance in computing systems is well established, and significant advances on how to deal with faults have been achieved by different communities. Rollback and recovery are predominate mechanisms to achieve fault tolerance in current High Performance Computing (HPC) and cloud computing environments [7], [8], [9]. Upon the occurrence of a fault, recovery is achieved by restarting the computation from a safe checkpoint [8]. Both coordinated and uncoordinated checkpointing schemes rollback and recovery schemes have been proposed [7], [8], [9], [10]. The drawback of coordinated checkpointing is a lack of scalability, as it requires global process coordination [5], [11], [12], [13], [14]. Uncoordinated checkpointing has not been widely adopted in HPC environments, due to its dependency on applications [15], [16]. Multi-level checkpointing can benefit from tolerance to failure but may increase failure rates of individual nodes and increase per-node cost [17].

Process and state machine replication has long been used to provide fault tolerance in distributed [18] and mission critical systems [19]. Based on this technique, a process's state and computation are replicated across independent computing nodes. Redundancy has been proposed, to augment existing checkpointing techniques [20], [21], [22].

Study of failure modelling is another direction of research. Eric et al. found that Weibull and Log-normal are the best fitting distributions to model failure in the high performance computing system based on the history data study [23]. Nosayba and Bianca studied the impact of different factors on the reliability of HPC systems such as environmental factors and nodes' usages. Cooling system failure may cause node outages, and high usages can lead to a higher failure rate. They also find that some nodes fail more often than others even when they have the same hardware specification [6].

In general, replication requires doubling the number of cores, with increases power consumption, which might not be efficient enough for future generation exascale infrastructure. Message logging-based approaches have been proposed to harness applications' temporal computation and communication patterns to reduce the cost of checkpointing for hybrid systems. The cost of recovery, however, remains proportional to the system size and not to the degree of failure. Partial redundancy has been analytically studied for HPC environment [24]. In this work, we study a practical selective replication strategy for cloud computing environment.

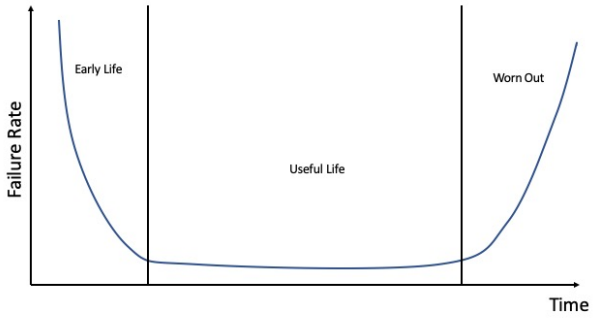

Figure 1. "Bathtub" Curve.

III. Heterogeneous Environment

As the age of computational cores increases, the likelihood of failure changes. It has been well studied and applied that the hardware reliability changes by following a "bathtub curve" as depicted in Figure 1 [25]. The life span of a hardware is divided into three periods: the early life period, the useful life period and the worn out period. The early life period is the beginning of the hardware's life span. It starts with a relatively high failure rate due to possible manufacturing defects. By fixing the defects, the failure rate continually decreases until it reaches the useful life period. The failure rate remains steady during the useful life period, and the useful life period is significantly longer than the other two life periods. When a hardware reaches the end of its life span, it start to wear out and the failure rates gradually increase. However, the increasing of failure rate is more gradual than the early life period.

Considering the hardware replacement and maintenance, systems may contain cores in different life periods and form a heterogeneity. Large-scale systems may encounter considerable amount of replacement of cores. Consequently, influence of unreliable cores in these systems cannot be neglected.

IV. Selective Replication

Process replication is one of the most popular fault-tolerance techniques. Full replication significantly reduces the failure likelihood, but also at least doubles the energy consumption and resource usage. To overcome this disadvantage, we propose a selective process replication model that only replicates processes host on error-prone cores. This is done because when compared to full replication, more cores are utilized for task computation, while processes on unreliable cores are protected. We assume that there are N number of cores assigned to the current job, and each core i has failure rate λ_i. The job has workload W. Assume each process occupies one core and executes in maximum speed. The subload is dependent on the number of main processes, M. In full replication, number of main processes occupies half of the available cores $M_f = \frac{N}{2}$, and the subload $w_f = \frac{W}{M_f}$ for each process. By selectively replicating processes, there are more cores available for main processes compared to that by using full replication, $M_s > M_f$. Therefore, the subload on each main process is reduced and so is the execution time. Consequently, the number of faults encountered may drop during execution and the time to completion may be reduced. Also, because of the reduction in redundancy, the energy

 ISBN: 978-1-61208-770-2

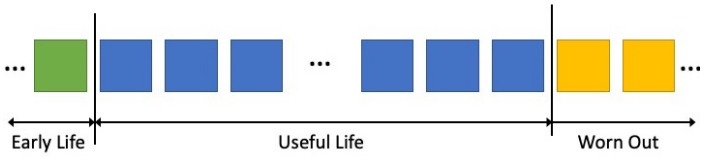

Figure 2. Cores in different life periods.

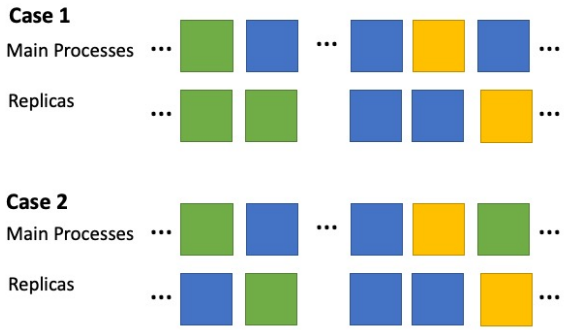

Figure 3. Cases of full replication.

consumption may be reduced. However, there are two major challenges that needs to be addressed for selective replication. The first challenge is to determine which cores need to be protected, and the second challenge is to determine which cores should be used to host replicas.

A. Replication Candidate Selection

Selecting processes that need to be replicated is a hard problem considering there are no absolute standards to determine if a core is reliable. In practice, however, we can estimate the cores' reliability by their history of failure in execution. In this paper, we estimate reliability as the failure rate based on the life span of the cores. Replicating processes on most unreliable cores is an essential guidance for candidate selection. Furthermore, determining the number of cores to replicate is a problem we need to solve. Analytical modeling is one way to determine the optimal ratio. However, it may not be the best solution if the modeling has too many assumptions. Moreover, the optimization may be compute-intensive and may not be able to produce a truly optimal solution. In our model, since cores are divided to three life periods, it may be a good approach to only replicate processes on cores in early life and worn out life period as a practical solution.

B. Replica Assignment

The goal of replica assignment is to determine which core should be used to host replicas to minimize the failure likelihood of any replication pair. It is studied that the best strategy to assign replica is pairing the most unreliable core with the most reliable core for replications [24].

We assume that there are cores in three different life periods, as shown in Figure 2. There is a small portion of cores in early life and worn out period as shown in Figure 2. Figure 3 illustrates the different cases of full replication. The obliviousness of failure rate difference may cause two unreliable cores to be paired for replication. These pairs may

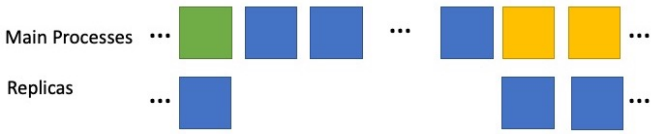

Figure 4. Selective Replication Assignment.

Algorithm 1 Algorithm for Selective Replica Assignment

Input:
$C^e = \{c_1^e, c_2^e, ..., c_K^e\}$
$C^u = \{c_1^u, c_2^u, ..., c_L^u\}$
$C^w = \{c_1^w, c_2^w, ..., c_Q^w\}$
Output: Assignment
Initialization : Assignment = $\{\}$
1: **for** $i = 1$ to K **do**
2: Assignment = Assignment $\cup \{< c_i^e, c_i^u >\}$
3: **end for**
4: **for** $i = 1$ to Q **do**
5: Assignment = Assignment $\cup \{< c_i^w, c_{L-i-1}^u >\}$
6: **end for**
7: **for** $i = K + 1$ to $L - Q$ **do**
8: Assignment = Assignment $\cup \{< c_i^u >\}$
9: **end for**
10: **return** Assignment

Figure 5. Algorithm for Selective Replica Assignment.

be less reliable than a core in useful life without replication. Additionally, the pairing of two reliable cores will lead to a high probability of energy wastage. Figure 4 depicts the proposed selective replication solution. Processes on cores in early life and worn out period are protected by replicas chosen from cores in the useful life period. The remaining cores in useful life period are utilized to complete the job by themselves. Algorithm 1, as shown in Figure 5, is used for selective replica assignment. C^e, C^u and C^w are sets of cores in early life period, useful life period and worn out period, respectively. The output is a set of tuples where each tuple is an assignment that contains either a pair of cores for replication or a single core for execution by themselves.

C. Discussion

Considering the practicality of the proposed model, it is very easy to apply it to current cloud computing infrastructures. Given the estimation of failure rate changes, which may be estimated by the past data, boundaries of different life periods are easy to determine. Therefore, it is straightforward to categorize the cores into the discussed three life periods. Consequently, the selective replica assignment algorithm is able to execute in linear time. Hence, the replica selection and assignment methods can be easily embedded with current cloud computing job scheduling algorithms for better resource allocation.

V. Experimental Evaluation

To evaluate our model, a comprehensive simulation-based experiment is conducted. To fairly compare selective replica-

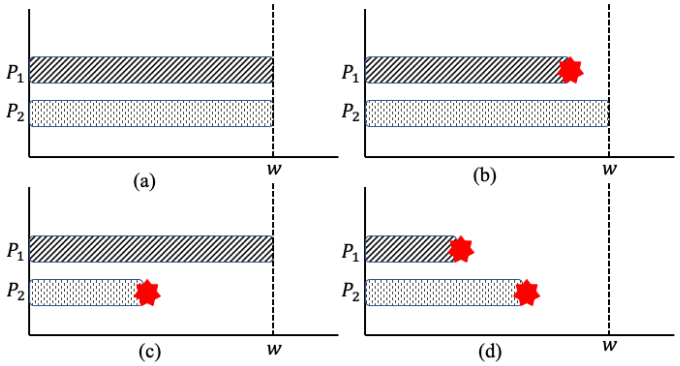

Figure 6. Failure cases on replication group of 2.

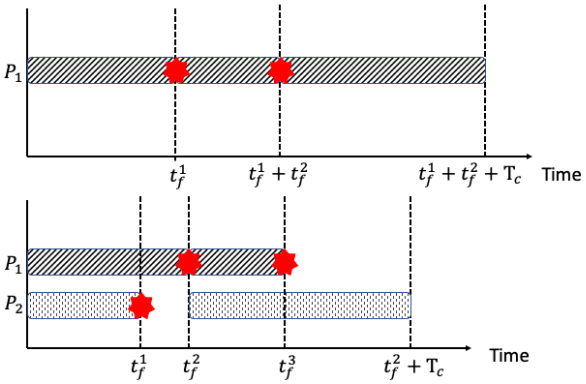

Figure 7. Failure on processes with and without replica.

tion with full replication, we simulate two baseline models – full replication with optimal replica assignment (baseline 1) and full replication with random replica assignment (baseline 2).

A. Simulation Setup

The simulation assumes that there are one million cores available ($N = 1,000,000$) such that each core hosts only one process and executes at maximum speed. To simulate the heterogeneous environment, the experiment assumes that cores are divided into three classes. These three classes represent three life periods: early life, useful life and worn out life period. We assume that the reliability of cores in each class is identical to make the simulation controllable. Furthermore, we assume that failures occur independently, but not identically. With consideration that the computational time is significantly shorter than the life span, we assume that the failure rate does not change during execution. Consequently, we make the assumption that failure follows an exponential distribution with mean set as the MTBF of the core. The MTBF of each core in these classes is 1 year, 10 years and 3 years, respectively. We varied the total workload (W) and ratio of cores in each class to observe how selective replication performed in different scenarios.

We use CSIM19 (C version) to conduct the simulation [26]. To simulate the failure time of a core, a random number is drawn from an exponential distribution with mean set as the core's MTBF. This random number represents the failure time of the core. Failure occurs only if the failure time is earlier than the failure free completion time. If failure occurs and the process has either no replica or a failed replica, the task is re-executed. We assume that the re-execution is on the repaired cores, and maintain the same fault tolerance strategy, with or without replicas. Thus, the re-execution may encounter additional failures on one subload's execution. For processes assigned a replica, there are four possible cases that may occur during the execution, as depicted in Figure 6. Case (a) is the execution of subload w without failure. Cases (b) and (c) represent the execution with one failure on the process or its replica. Case (d) has both the process and its replica failed before the task completion.

We normalize the hourly dynamic energy consumption rate of each core as $R^D = 1$, i.e., for each core, the dynamic energy consumption of executing one-hour job is 1 unit of energy consumption. Other than dynamic energy, static energy is another aspect of energy consumption. It comes from the power leakage of several components (processor, memory, etc.). We define the static energy consumption as a predefined fraction $\rho = 0.5$ of dynamic energy consumption [27]. The calculation of energy consumption is based on the execution time of each process, including execution time before failure and re-execution time. The minimum execution time is the failure free completion time of subload w. There is no maximum execution time for a subload w because it is possible to have failure in every execution and re-execution. For example, a process P_1 without replication encountered two failures during execution as depicted in Figure 7. The execution time before failure for each failed execution is t_f^1 and t_f^2, respectively. The total energy consumption of process P_1 is $E^1 = (1+\rho)R^D(t_f^1 + t_f^2 + T_c)$, where T_c is the failure free execution time of subload w. Since we represent the workload as hours of execution, the failure-free execution time is same as that of the subload, $T_c = w$. For processes with replicas, the computation of energy consumption requires to consider the failure time for both process and its replica when failures occur in both cores. The second example in Figure 7 depicts the case where both the process P_1 and its replica P_2 failed before the completion of task. The re-execution starts from t_f^2, and the replica P_2 finishes the task at $t_f^2 + T_c$. The energy consumption of this process pair is calculated as $E^{1,2} = (1+\rho)R^D t_f^3 + (1+\rho)R^D(t_f^1 + T_c)$. The total energy consumption is the sum of the energy consumption for all the processes. We assume no energy consumption after completion of the execution for each process. The time to completion of the total workload W is the maximum completion time among all the subtasks. For each setup, we simulate the job execution 100 times and report the average result to overcome the bias of randomness.

B. Sensitivity Analysis on total workload

The goal of this analysis is to evaluate the performance of the proposed model with different workloads. Heavier workload indicates longer time to completion. Intuitively, longer time to completion indicates that failures are more likely to occur during execution. We vary the total workload from 10 to 2000 million hours of works. The percentage of cores in each class is 5%, 80%, and 15%, respectively. The results of comparing selective replication with baseline model 1 and 2 are depicted in Figures 8 and 9, respectively. Compared to full replication with and without optimal replica

 ISBN: 978-1-61208-770-2

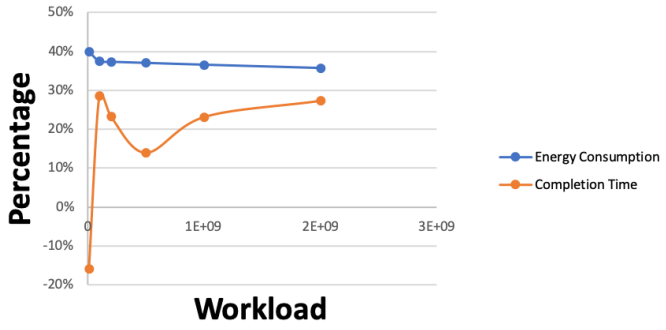

Figure 8. Comparison between selective replication with baseline 1 in different workloads.

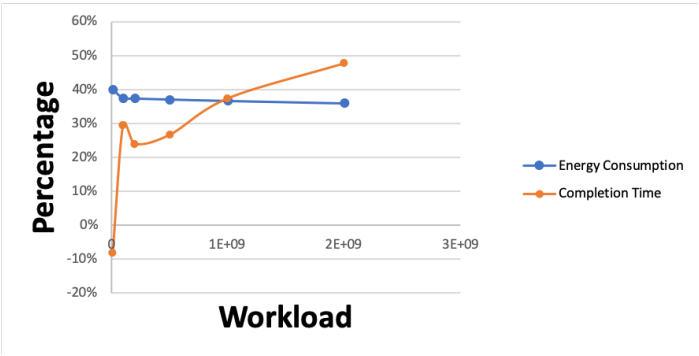

Figure 9. Comparison between selective replication with baseline 2 in different workloads.

assignment, selective replication can reduce more than 35 % of energy consumption. With 10 million hours of total work, full replication has slightly shorter response time. Selective replication model finishes the job in 23.9 hours. The optimal replica assignment with full replication completes the job in 20.6 hours, while the full replication with random replica assignment completes the job in 22.1 hours. There is about 40 % of energy consumption reduced compared to both baseline models. Due to the fairly light workload, it is very unlikely to have failure on both the process and its replica simultaneously. Therefore, the completion time of full replication is very close to the failure free completion time as $T_c = w = \frac{2W}{N} = 20$ hours. Compared to baseline 2, which is the traditional full replication model, the difference on time to completion is very marginal.

As the total workload increases, the selective replication model has a shorter response time than the baseline models. For 2000 million hours of work, selective replication achieved 48% reduction in time to completion compared to full replication with random replica assignment. It indicates that, with heavier workload, the likelihood of failure on process with replication start to increase and it causes longer total completion time. For selective replication, the total completion time is under double of failure free completion time for full replication when there are no more than 1000 million hours of work. This sensitivity analysis shows that selective replication has advantage on energy consumption with different workloads and advantage on time to completion with heavy workload in cloud computing platform.

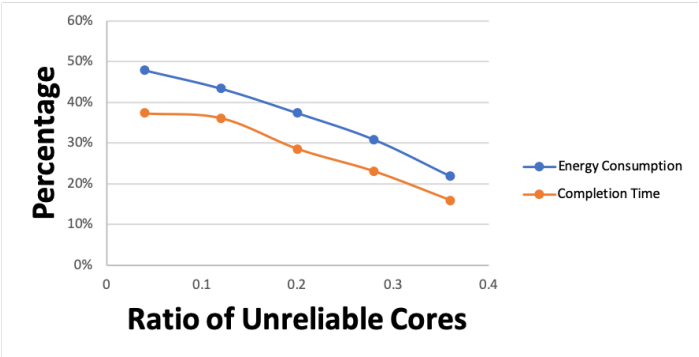

Figure 10. Comparison between selective replication with baseline 1 in different ratios of unreliable cores.

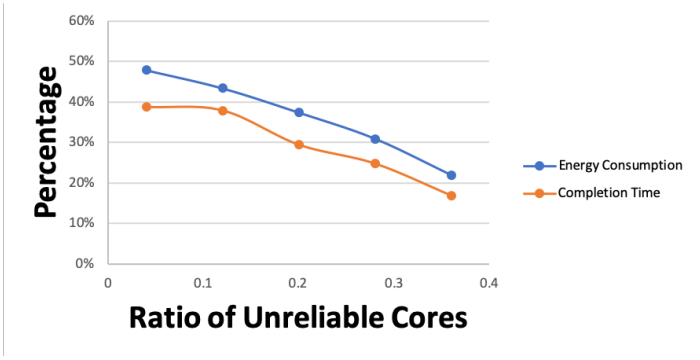

Figure 11. Comparison between selective replication with baseline 2 in different ratios of unreliable cores.

C. Sensitivity Analysis on ratio of cores in each class

The goal of this analysis is to evaluate the performance of selective replication with different percentage of unreliable cores. The total workload for this analysis is 100 million hours of work. We vary the percentage of cores in early life and worn out period to represent different scenarios. We maintain the ratio of cores in these two classes as 1:3 to ensure that the results are comparable. The results of comparing selective replication with baseline model 1 and 2 are depicted in Figures 10 and 11, respectively. When 36% of cores are unreliable (9% of cores in early life and 27% of cores in worn out period)(a relatively more unreliable scenario of the analysis), selective replication reduces about 22% of energy consumption compared to two baseline models. It also reduces about 16% and 17% of time to completion when compared to full replication with and without optimal replica assignment strategy, respectively. Considering that in our approach, only 28 % of cores are not associated with replicas, the improvements are significant. As the reliability increases, the reduction in energy consumption and time to completion increases. With only 4 % of unreliable cores, selective replication achieves about 48% of reduction in energy consumption and about 39% of reduction in time to completion when compared to the baseline model 2.

Hence, we observe that selective replication has an advantage in all scenarios with different reliabilities, and has more advantage in reliable scenarios. Also, our experiment validated that the optimal replica assignment strategy has advantage on cloud computing platform as the baseline model 1 always has shorter time to completion when compared to baseline model 2.

 ISBN: 978-1-61208-770-2

VI. Conclusion and Future Works

The major contribution of this paper was to address the problem that, when infrastructures grow to large scale, traditional full replication fault tolerance strategy requires high amount of resources. By considering the failure rate difference of computational cores of different ages, we proposed the selective replication model that only replicates processes on unreliable cores to reduce the energy consumption and response time. The results of our experimental evaluations showed that, compared to full replication, selective replication can reduce more than 35 percent of energy consumption and about 30 percent of completion time simultaneously with 100 million hours of workload and 1 million cores.

In this work, we proposed a model categorizing the cores into three different classes. We may gain more reduction in energy consumption if we increase the granularity. Our framework is a practical approach to reduce energy consumption and response time. However, it may not be the optimal solution to this problem. An optimization solution with the necessary assumptions may produce better results. Furthermore, the proposed solution estimated failure rates based on the age of the cores. Other factors such as spatial and temporal dependencies among core failures is not considered. By exploring the emerging online and offline machine learning methods, we may gain better insights into temporally and spatially correlated failures, which can then be used to predict when failures are likely to occur. Therefore, it would be possible to dynamically assign replicas to further reduce energy consumption and execution time.

Acknowledgment

This material is based in part upon work supported by the National Science Foundation under Grants Number CNS-1252306 and CNS-1253218. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. W. Services, "Overview of amazon web services," Amazon Whitepapers, 2019.

[2] B. Mills, R. E. Grant, K. B. Ferreira, and R. Riesen, "Evaluating energy saving for checkpoint/restart," in First International Workshop on Energy Efficient Supercomputing (E2SC) in conjunction with SC13: The International Conference for High Performance Computing, Networking, Storage and Analysis, Nov 2013, pp. 1–8.

[3] K. Ferreira et al., "Evaluating the viability of process replication reliability for exascale systems," in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 1–12.

[4] R. Oldfield et al., "Modeling the impact of checkpoints on next-generation systems," in Mass Storage Systems and Technologies, 2007. MSST 2007. 24th IEEE Conference on, sept. 2007, pp. 30 –46.

[5] R. Riesen et al., "Redundant computing for exascale systems," Sandia National Laboratories, no. SAND2010-8709, December 2010.

[6] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how hpc systems fail," in 2013 43rd annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE, 2013, pp. 1–12.

[7] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," ACM Comput. Surv., vol. 34, no. 3, Sep. 2002, pp. 375–408.

[8] S. Kalaiselvi and V. Rajaraman, "A survey of checkpointing algorithms for parallel and distributed computers," Sadhana, vol. 25, no. 5, 2000, pp. 489–510.

[9] B. Meroufel and G. Belalem, "Adaptive time-based coordinated checkpointing for cloud computing workfl ows," Scalable Computing: Practice and Experience, vol. 15, no. 2, 2014, pp. 153–168.

[10] J. Daly, "A model for predicting the optimum checkpoint interval for restart dumps," in Int. Conf. on Computation Science, 2003, pp. 3–12.

[11] E. N. Elnozahy and J. S. Plank, "Checkpointing for peta-scale systems: A look into the future of practical rollback-recovery," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 2, 2004, pp. 97–108.

[12] M. Bougeret, H. Casanova, Y. Robert, F. Vivien, and D. Zaidouni, "Using group replication for resilience on exascale systems," INRIA, Rapport de recherche RR-7876, Feb. 2012.

[13] P. Hargrove and J. Duell, "Berkeley lab checkpoint/restart (blcr) for linux clusters," in J. Phys. Conf. Ser, vol. 46, no. 1, 2006, p. 494.

[14] N. Losada, G. Bosilca, A. Bouteiller, P. González, and M. J. Martín, "Local rollback for resilient mpi applications with application-level checkpointing and message logging," Future Generation Computer Systems, vol. 91, 2019, pp. 450–464.

[15] G. Zheng, L. Shi, and L. V. Kalé, "FTC-Charm++: an in-memory checkpoint-based fault tolerant runtime for Charm++ and MPI," in Cluster Computing, 2004, pp. 93–103.

[16] A. Guermouche, T. Ropars, E. Brunet, M. Snir, and F. Cappello, "Uncoordinated checkpointing without domino effect for send-deterministic mpi applications," in IPDPS, May 2011, pp. 989–1000.

[17] D. Hakkarinen and Z. Chen, "Multilevel diskless checkpointing," Computers, IEEE Transactions on, vol. 62, no. 4, April 2013, pp. 772–783.

[18] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," ACM Computing Surveys (CSUR), vol. 22, no. 4, 1990, pp. 299–319.

[19] J. F. Bartlett, "A nonstop kernel," in ACM SIGOPS Operating Systems Review, vol. 15, no. 5. ACM, 1981, pp. 22–29.

[20] J. Stearley et al., "Does partial replication pay off?" in IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012). IEEE, 2012, pp. 1–6.

[21] J. Elliott et al., "Combining partial redundancy and checkpointing for hpc," in 2012 IEEE 32nd International Conference on Distributed Computing Systems. IEEE, 2012, pp. 615–626.

[22] H. Casanova, Y. Robert, F. Vivien, and D. Zaidouni, "Combining Process Replication and Checkpointing for Resilience on Exascale Systems," INRIA, Research Report RR-7951, May 2012. [Online]. Available: https://hal.inria.fr/hal-00697180 [retrieved: January, 2020]

[23] E. Heien et al., "Modeling and tolerating heterogeneous failures in large parallel systems," in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2011, p. 45.

[24] Z. Hussain, T. Znati, and R. Melhem, "Partial redundancy in hpc systems with non-uniform node reliabilities," in SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2018, pp. 566–576.

[25] W. M. Jones, J. T. Daly, and N. DeBardeleben, "Application monitoring and checkpointing in hpc: looking towards exascale systems," in Proceedings of the 50th Annual Southeast Regional Conference. ACM, 2012, pp. 262–267.

[26] H. Schwetman, "Csim19: a powerful tool for building system models," in Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304), vol. 1. IEEE, 2001, pp. 250–255.

[27] C. Rusu, R. Melhem, and D. Mossé, "Maximizing rewards for real-time applications with energy constraints," ACM Transactions on Embedded Computing Systems (TECS), vol. 2, no. 4, 2003, pp. 537–559.

 ISBN: 978-1-61208-770-2