International Journal of Research in Advanced Engineering and Technology
ISSN: 2455-0876; Impact Factor: RJIF 5.44
www.newengineeringjournal.in
Volume 3; Issue 2; May 2017; Page No. 06-08

System health monitoring and evaluation using cooperative check pointing mechanism

Ozor Godwin O, Oleka Chioma V, Nwobodo Lois O

Computer Engineering, Enugu State University of Science and Technology, Enugu, Nigeria

Abstract
Incessant failures had been recorded in medium and large scale systems in the past. Greater percentage of the damages were during runtime and partly on design. The problem is largely attributed to poor mechanism for checking and evaluating health of the system both in design and operation. To mitigate the challenges, an effective, simple but rugged fault tolerance mechanism was proposed. Cooperative checkpointing fault tolerance mechanism was investigated and modelled to report the health status of the system and recommend a real-time response to the results of the evaluation. The theoretical results was simulated in MATLAB.

Keywords: system health, cooperative checkpointing, fault, reliability

1. Introduction
Cooperative checkpointing uses global knowledge of the state of the machine to improve performance and reliability by dynamically deciding when to skip checkpoint requests made by applications. Checkpointing involves periodically saving a sufficient amount of the state of a running job to stable storage, allowing for that job to be restarted from the last successful checkpoint. Checkpoints have an associated overhead, usually dictated by the bottleneck to the stable storage system. Therefore, while there is a risk associated with not checkpointing, there is also a direct and measurable cost associated with performing the checkpoints. As systems grow larger, this overhead may increase due to the greater coordination necessary to guarantee a consistent checkpoint, more data that requires saving, and interference with other applications. Optimally, every checkpoint is used for recovery and every checkpoint is completed immediately preceding a failure. A central insight of cooperative checkpointing is that skipping checkpoints that are less likely to be used for recovery can improve reliability and performance. Application need to be recoded when they are ported to a system with different reliability characteristics. Cooperative checkpointing allows for irregular checkpoint intervals by giving the system an opportunity to skip requested checkpoints at runtime. Therefore, cooperative checkpointing may be thought of as a hybrid of application-initiated and system-initiated checkpointing.
The application requests checkpoints, and the system either grants or denies each one. Without cooperative checkpointing, all application-initiated checkpoints are taken, even if system-level considerations would have revealed that some are grossly inefficient or have a low probability of being used for recovery. If the heuristics used by the system are reasonably confident that a particular checkpoint should be skipped, a benefit is conferred to both parties. That is, the application may finish sooner or at a lower cost because checkpoints were performed at more efficient times, and the system may accomplish more useful work because fewer checkpoints were wasted. Checkpointing is the best solution for providing reliable completion of jobs on inherently unreliable hardware, though the rate at which the data sets are growing is outpacing growth in the speed of stable storage and networks. In other words, there is an I/O bottleneck facing the massive clusters when they attempt to save their state. Taken together, it is clear that standard checkpointing techniques must be reevaluated [1]. Many high-performance computing applications are executed repetitiously [2], making their behavior amendable to modeling.

2. Checkpointing
Checkpointing for computer systems has been a major area of research over the past few decades. The goal of high performance computing is to obtain maximum efficiency from given resources. System failures, and the resulting job failures, are a significant cause of degraded performance. Recently, there have been a number of studies on checkpointing based on certain failure characteristics [3], including Poisson distributions. Plank and Elwasif [4] carried out a study on system performance in the presence of real failure distributions and concluded that it is unlikely that failures in a computer system would follow a Poisson distribution.
The job workloads themselves can be of importance for the performance evaluation of high performance computing systems. The workloads considered by Plank and Elwasif for their study are artificial. Similarly, the communication/network topologies can play an important role in regard to checkpointing or job scheduling strategies for HPC systems. Tantawi and Ruschitzka [5] developed a theoretical framework for performance analysis of checkpointing schemes. In addition to considering arbitrary failure distributions, they present the concept of an equicost checkpointing strategy, which varies the checkpoint interval according to a balance between the checkpointing cost and the likelihood of failure. Such a strategy would be costly in practice, because it is expensive to checkpoint at arbitrary points in a program's execution.

3. Cooperative Checkpointing
Cooperative checkpointing is a set of semantics and policies that allow the application, compiler, and system to jointly

decide when checkpoints should be performed. Specifically, the application requests checkpoints, which have been optimized for performance by the compiler, and the system grants or denies these requests. The general process consists of three parts:

a) The application programmer inserts checkpoint requests in the code at places where the state is minimal, or where a checkpoint is otherwise efficient. These checkpoints can be placed liberally throughout the code, and permit the user to place an upper bound on the number and rate of checkpoints.
b) The compiler optimizes these requests by catching errors, removing dead variables, and assisting with optimization techniques such as incremental checkpointing. In the case of cooperative checkpointing, the compiler may move the checkpoint request to a slightly earlier point in time; this permits a number of additional performance improvements.
c) The system receives and considers checkpoint requests. Based on system conditions such as I/O traffic, critical event predictions, and user requirements, this request is either granted or denied. The mechanism that handles these requests is referred to as the checkpoint gatekeeper or, simply, the gatekeeper. The request/response latency for this exchange is assumed to be negligible.

Cooperative checkpointing appears to an observer as irregularity in the checkpointing interval. If we model failures as having an estimable MTBF, but not much else, then periodic checkpointing is sensible (even optimal). But once these failures are seen to be predictable, and other factors are considered, this irregularity allows us to do much better.

The dynamics of operations in system need prognosis of the statues. The heath assessment is based on the reliability and availability of each unit that makes up the entire system. The mean time before failure must be considered. The storage capacity of the system was considered in this work, as their will be incremental load as a result of operation or routine test which is triggered by the checkpointing embedded in the programming. We adopted rate of checkpointing as the key indicator for system health assessment as will be discussed in section 3.

4. Methodology and Analysis

Cooperative checkpointing stands tall in 'smart' assessment of the system, due to its ability to initiate operation when necessary based on the vital variable indicators' report and analysis. The reliability of the units was assessed and hence the rate of checkpointing to ascertain and predict the health statues of a system.

Reliability is defined as "the probability of a device performing its purpose adequately for the period of time intended under the operating conditions encountered":

$$R(t) = P(T > t), t \geq 0 \quad (1)$$

where T is a random variable denoting the time to failure. In other words, reliability is the probability that the value of the random variable T is greater than the mission time t.

Probability of failure, $F(t)$, is defined as the probability that the system will fail by time t:

$$F(t) = P(T \leq t), t \geq 0 \quad (2)$$
$$R(t) = 1 - F(t) \quad (3)$$
$$R(t) = e^{-\lambda t} \quad (4)$$

Where λ is the failure rate

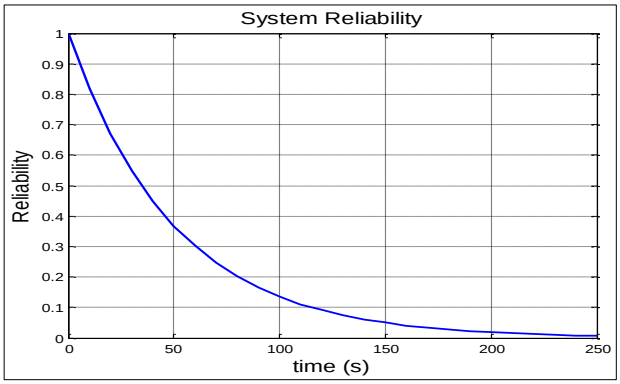

Fig 1: System Reliability curve as simulated with 0.02 as failure rate

Also the rate of cooperative checkpointing operation was simulated to ascertain the health statues of a system, hence a decision was taken based on the output.

Table 1: Rate of Cooperative Checkpointing as observed in two system

Time of Operation in a Day	System 1 (Checkpointing Recording)	System 2 (Checkpointing Recording)
1	5	2
2	6	5
3	7	8
4	5	4
5	4	3
6	6	3
7	6	2
8	6	0
9	6	3
10	5	1
11	7	3
12	8	3
13	5	2
14	4	1
15	6	1
16	7	0
17	6	1
18	7	1
19	7	0

20	7	1
21	5	1
22	5	1
23	6	1
24	5	1

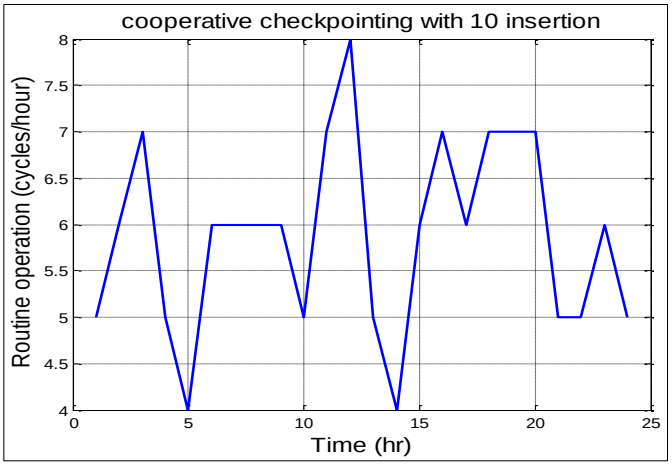

Fig 2: cooperative checkpointing with more cycles/hr

In this simulated work, 10 cooperative checkpointing insertion were considered during programming and operation of the system. From the recordings in Fig.2, it shows clearly that there was high rate of checkpointing, which indicates that the system is suspecting unhealthy situation hence 'smart' request for frequent checkpoint.

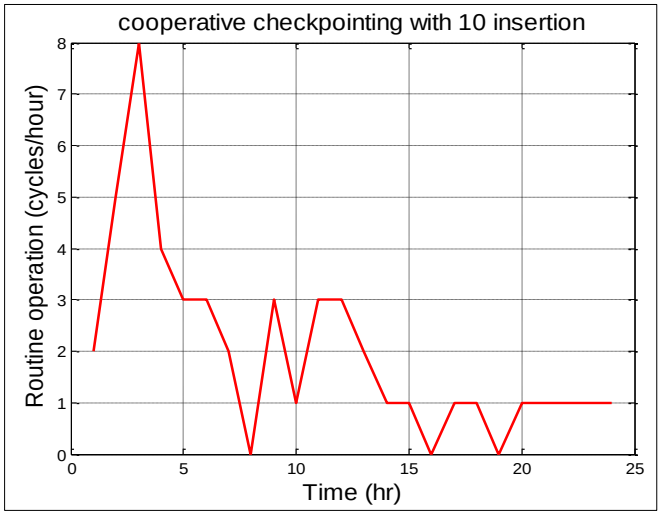

Fig 3: cooperative checkpointing with less cycles/hr

In Fig.3, the system experience less cycles of checkpointing and even skipping of some points in some operation, hence an indication of healthy system. The recording was monitored in hourly for a day.

Conclusion

It was inferred from the research that cooperative checkpointing is a viable tool for predicting the health statues of a system. The model investigated shows a level of 'intelligence' in skipping some points when sensed that the part is through, reliable and available for the operation. Frequent and incessant checkpoint is also a challenge of operation hence the need to check the operation checkpointing based on the programmed reference point.

References

1. Elmootazbellah N. Elnozahy, James S. Plank. Checkpointing for peta-scale systems: A look into the future of practical rollback-recovery. IEEE Trans. Dependable Secur. Comput. 2004; 1(2):97-108.
2. Uri Lublin, Dror G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. Parallel Distrib. Comput. 2003; 63(11):1105-1122.
3. Plank JS, Thomason MG. Processor allocation and checkpoint interval selection in cluster computing systems. Journal of Parallel and Distributed Computing. 2001; 61(11):1570-1590.
4. Plank JS, Elwasif WR. Experimental assessment of workstation failures and their impact on checkpointing systems. In The 28th Intl. Symposium on Fault-tolerant Computing, 1998.
5. Tantawi AN, Ruschitzka M. Performance analysis of checkpointing strategies. In ACM Transactions on Computer Systems. 1984; 110:123-144.