Report Number: WUCS-93-36

1993-01-01

Distributed Computing Systems and Checkpointing

Authors: Ken Wong and Mark Franklin

This paper examines the performance of synchronous checkpointing in a distributed computing environment with and without load redistribution. Performance models are developed, and optimum checkpoint intervals are determined. The analysis extends earlier work by allowing for multiple nodes, state dependent checkpoint intervals, and a performance metric which is coupled with failure-free performance and the speedup functions associated with implementation of parallel algorithms. Expressions for the optimum checkpoint intervals for synchronous checkpointing with and without load redistribution are derived and the results are then used to determine when load redistribution is advantageous.

Distributed Computing Systems and Checkpointing

Ken Wong
Mark Franklin

WUCS-93-36

July 1993

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis, MO 63130-4899

Distributed Computing Systems and Checkpointing

Ken Wong and Mark Franklin

The Computer and Communications Research Center
Washington University
One Brookings Drive, Campus Box 1115
St. Louis, MO 63130-4899
kenw@wuccrc.wustl.edu, jbf@wuccrc.wustl.edu

Presented at the

Second International Symposium on

High Performance Distributed Computing

July 21-23, 1993
Spokane, Washington

Abstract

This paper examines the performance of synchronous checkpointing in a distributed computing environment with and without load redistribution. Performance models are developed, and optimum checkpoint intervals are determined. The analysis extends earlier work by allowing for multiple nodes, state dependent checkpoint intervals, and a performance metric which is coupled with failure-free performance and the speedup functions associated with implementation of parallel algorithms. Expressions for the optimum checkpoint intervals for synchronous checkpointing with and without load redistribution are derived and the results are then used to determine when load redistribution is advantageous.

Distributed Computing Systems and Checkpointing*

Ken Wong and Mark Franklin

The Computer and Communications Research Center
Washington University, St. Louis, MO 63130

Abstract

This paper examines the performance of synchronous checkpointing in a distributed computing environment with and without load redistribution. Performance models are developed, and optimum checkpoint intervals are determined. The analysis extends earlier work by allowing for multiple nodes, state dependent checkpoint intervals, and a performance metric which is coupled with failure-free performance and the speedup functions associated with implementation of parallel algorithms. Expressions for the optimum checkpoint intervals for synchronous checkpointing with and without load redistribution are derived and the results are then used to determine when load redistribution is advantageous.

1. Introduction

The dual emerging technologies associated with gigabit networks [1] and high-speed processors (supercomputers), suggest the possibility of tackling very large, computationally intensive problems by coupling these technologies into a distributed computing environment. The large applications which make good candidates for this environment may produce results only after many hours even when multiple computers are employed. For example, several computing sites around the U.S. might be willing to cooperate (i.e., act as a distributed parallel processor) in tackling a difficult simulation problem. The computing sites may consist of computational resources from several vendors, and communcation between sites may require message transmission over long distances (thousands of miles) through several intermediate hops. Clearly, computing in this environment is much more precarious and we can expect higher resource failures rates than in a standard multiprocessor. Thus, a fundamental problem which must be addressed in this environment is that of providing effective computational progress in the face of resource failures.

* This research has been sponsored in part by funding from the NSF under Grant CCR-9021041.

In order to achieve maximum speed, the computational tasks must be assigned to the resources to exploit maximum parallelism. However, the possibility of a system failure (and therefore a complete restart) increases as larger numbers of processors are brought to bear on the application. Fault tolerant techniques must be used to insure finishing times which are comparable with fault-free performance.

One approach to providing higher reliability is to have each site periodically checkpoint by making a copy of the system state onto stable storage such as disk. When a failure occurs, each site can resume computing after it restores its system state by reading the latest checkpoint from its checkpoint storage. An important issue in such a system concerns the selection of checkpoint frequency. Checkpointing too frequently in a highly reliable system results in unnecessary overhead, while checkpointing too infrequently in a highly unreliable system results in the loss of large quantities of work — work which must be repeated after a failure.

Note also that in a system where the repair times are long, it may be beneficial to redistribute the load onto the remaining operational processors and resume computing (at a lower aggregate rate) instead of waiting for its repair. When repair is completed, the load could then be redistributed back onto the repaired processor. This approach allows for graceful performance degradation in the face of failures.

This paper examines the performance of synchronous checkpointing in a distributed computing environment with and without load redistribution. Performance models are developed, and optimum checkpoint intervals are determined. The analysis significantly extends earlier work by allowing for multiple nodes, state dependent checkpoint intervals, and a performance metric which is coupled with failure-free performance and the speedup functions associated with implementation of parallel algorithms. Expressions for the optimum checkpoint intervals for synchronous checkpointing with and without load redistribution are derived and the results are then used to determine when load redistribution is advantageous. Thus, the connected issues of checkpoint interval and load redistribution are considered and, for a given set of system and application parameters, an optimum checkpointing and

load distribution scheme can be selected.

2. Optimum checkpointing

Optimum checkpointing for the single-node case has been studied extensively [2,3,4,5,6,7,8]. Optimum checkpoint intervals have been found by maximizing availability or minimizing response time. The objective function is typically convex, and analytic or numerical solutions can be found in many cases [8]. In most cases, a transaction-oriented environment is assumed where jobs or requests arrive from a Poisson source.

The optimum selection of checkpoint intervals for the multicomputer case has been sparsely studied. Gelenbe, et. al., developed a model which included the overhead of fault detection [9]. Gelenbe assumed that the nodes were homogeneous, the load at each node was identical, the nodes were tested periodically for faults, and there was an external source of jobs. In his development, requests sent to faulty nodes were routed to operational ones. Thus, the mean input rate to a node is a function of the job source rate, the number of faulty nodes, and the precision of fault detection. Based on these assumptions, expressions for the optimum checkpoint interval and optimum testing interval were obtained.

Our models consider a related situation but with the following differences:

1) Jobs are generated internally as the result of other jobs rather than coming from an external Poisson source.
2) Synchronous checkpointing is employed rather than an asynchronous checkpointing algorithm.
3) Fault detection is not explicitly modeled.

Our motivation is driven by a desire to model applications in a scientific computation environment rather than a transaction-oriented one. These differences lead to different models, solution techniques, and results.

We begin by considering the single-node case and then extend this to multiple nodes. A node can be viewed as being in one of three states: A) available, C) checkpointing, or R) recovering. For mathematical tractability and simplicity, we assume 1) Markovian state occupancy times, 2) failures form a Poisson process, and 3) failures only occur when a node is in the available state. This is reasonable when checkpoint and recovery times are small compared to the time that the node is available between such events.

The state transition-rate diagram for this Markov process is shown in Figure 1, and the parameters in the diagram are defined in Table I. After spending on average α^{-1} time units in the available state (A), the system will enter the checkpointing state (C). It spends on average β^{-1} time units saving the system state before it reenters the available state to resume computing. Occasionally, the system will fail. On average, the system will be available for ϕ^{-1} time units before this happens. When a failure occurs, the system will spend σ^{-1} time units recovering to the failure point before resuming the computation in the available state.

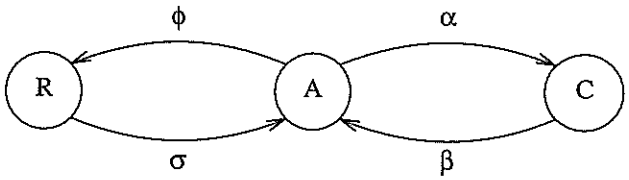

Figure 1. Single-Node State Transition Rate Diagram.

Parameter	Description
ϕ^{-1}	mean failure time
α^{-1}	mean time between checkpoints
β^{-1}	mean time to perform a checkpoint
σ^{-1}	mean recovery time

Table I. Model Parameters.

The availability (steady-state probability of being in the available state) π_A can be derived using standard CTMC (continuous-time Markov chain) techniques [10]:

$$\pi_A = \left[1 + \frac{\alpha}{\beta} + \frac{\phi}{\sigma}\right]^{-1} \tag{1}$$

Note that the mean recovery rate σ is a function of the mean checkpoint rate α since the amount of work to be repeated after a failure is related to the time between checkpoints. In order to determine how the availability π_A varies with the intercheckpoint frequency α, we must first express σ as a function of α.

The mean recovery time σ^{-1} consists of three time components: 1) a mean system repair time ρ^{-1}, 2) a mean state restoration time r, and 3) a mean recomputation time. The recomputation time is the time spent recomputing from the restored state to the failure point. If the system is never idle while in the available state, renewal theory says that the failure point will occur α^{-1} from the latest checkpoint since the failure and checkpointing processes are Poisson. If we assume that the system is never idle while in the available state, the mean recomputation time is

$$\sigma^{-1} = \rho^{-1} + r + \alpha^{-1} \tag{2}$$

The availability can now be written as:

$$\pi_A = \left[1 + \frac{\alpha}{\beta} + \phi\left(\rho^{-1} + r + \alpha^{-1}\right)\right]^{-1} \tag{3}$$

Setting the derivative of π_A with respect to α to 0 and solving for α leads us to the optimum checkpoint rate:

$$\alpha^{opt} = \sqrt{\phi\,\beta} \tag{4}$$

This corresponds to the result found in the literature [2,3]. The optimum checkpoint rate α^{opt} behaves as expected. That is, checkpoints should be taken more frequently (larger α) as failures occur more frequently (larger ϕ). Also, as checkpoints take less time (larger β), checkpoints can be taken more often (larger α) since there is less overhead associated with checkpointing.

At first, it may seem strange that the optimum checkpoint rate does not depend upon the repair and recovery parameters ρ and r. However, in the time interval between failures, the checkpoint frequency (α) controls only the overheads associated with checkpointing (through the number of checkpoints) and recomputation during recovery (through the intercheckpoint period). Changing the checkpointing frequency will not affect the repair and restore time components ρ^{-1} and r respectively.

The multi-node case offers us the opportunity to choose between several recovery methods. Consider two synchronous checkpoint recovery methods: one with load redistribution and one without load redistribution. Typically, if the down time after a failure is short, it is reasonable to wait for the failed node to recover before continuing the computation. However, if the failed node will be unavailable for a significant amount of time, it may make sense to redistribute the load among the remaining nodes, repeat the lost work, and then continue.

In addition to the three single-node model assumptions discussed earlier, we assume that 4) the fault-free speed-up curve is known, 5) the nodes are homogeneous, 6) the load is balanced across the processors, and 7) the processor utilization is approximately equal to the ratio of the speed-up to the number of processors. Assumption 4 is reasonable for many scientific computations. The fault-free speed-up is the ratio of the single-node and multi-node finishing times in a fault-free system ($T(1)$ and $T(N)$) and is defined as:

$$S(N) = \frac{T(1)}{T(N)}, \qquad N \geq 1 \tag{5}$$

Figure 2 shows two typical curves for $S(N)$. In the ideal situation, the speed-up with N processors would be N. Because of multiprocessor overheads, synchronization delays, and communication delays, however, the speed-up is generally less than N. The dashed curve indicates linear speed-up. The solid curve is approximately the same as the dashed curve for small numbers of processors, but then reaches an assymptote for larger numbers of processors. This deviation from linearity can be a result of a lack of sufficient parallelism, or heavy message traffic. We extend the speed-up measure to a faulty environment by defining the speed-up in a faulty environment $\bar{S}$ to be the ratio of the fault-free single-node finishing time $T(1)$ to the faulty multi-node finishing time $T^*(N)$; that is,

$$\bar{S}(N) = \frac{T(1)}{T^*(N)}, \qquad N \geq 1 \tag{6}$$

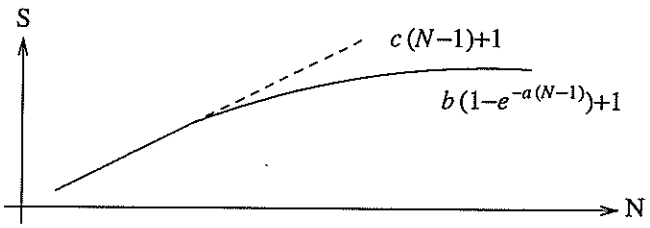

Figure 2. Speed-Up as a Function of Number of Nodes.

Assumptions 5 and 6 (homogeneous nodes and load balance) are made for illustrative purposes. Equivalently, the interfailure time, intercheckpoint time, and checkpoint duration at any node are stochastically identical to those on any other node. Although we make this last assumption for analytic tractability, the basic development would remain the same even without this last assumption, although the resulting equations would be more complex.

Assumption 7 is also made for analytic tractability. We assume that the fault-free utilization $U(N)$ is

$$U(N) = \frac{S(N)}{N} = \frac{T(1)}{T(N)}\frac{1}{N}, \qquad N \geq 1 \tag{7}$$

Note that this is equivalent to assuming that the sum of the CPU time component of all processors is equal to the CPU time of a single processor system. This can be seen by noting that $U(N)T(N)$ is the CPU time of each processor. The sum of the CPU times of all N processors is $NU(N)T(N)$ which is equal to $T(1)$ when $U(N)$ is replaced by the righthand side of Equation 7.

3. Model I — synchronous checkpointing without load redistribution

Begin by considering the simplest case: synchronous checkpointing without load redistribution. The disadvantages of this approach for systems with many nodes are the high synchronization cost and low availability when repair times are slow. In this approach, all nodes checkpoint at approximately the same time. A two-phase commit protocol might be used to synchronize the start of a checkpointing phase. Once the checkpointing is done the system is available for normal computation. If any node fails, the whole system must go through a recovery phase. After recovery, the system is again operational with N nodes.

The system is equivalent to a single-node running N times as fast when operational, but also failing N times as often. Such a system can be in one of three states: A) all nodes are available, C) all nodes are checkpointing, or R) all nodes are recovering. The availability π_A of this system can be obtained from the single-node case by noting that the mean system failure rate is now $N\phi$ instead of ϕ. Furthermore, since the nodes can be idle during the intercheckpoint period, the mean recompute time is now given by $U(N)\alpha^{-1}$ where $U(N)$ is the fault-free utilization

when there are N processors. Using equation 3, the availability is:

$$\pi_A = \left[1 + \frac{\alpha}{\beta} + N\phi\left[\rho^{-1} + r + U(N)\alpha^{-1}\right] \right]^{-1} \quad (8)$$

The objective is to maximize the availability of the system. Since the multi-node availability equation is identical to the single-node one except $N\phi$ replaced ϕ, and $U(N)\alpha^{-1}$ replaced α^{-1}, it is easy to see that the optimum checkpoint rate will be

$$\alpha^{opt} = \sqrt{N\phi\beta U(N)} \quad (9)$$

The speed-up using the optimum checkpoint frequency can now be derived. If $T(N)$ is the finishing time of the fault-free system with N processors, the finishing time of the faulty system is $T^*(N) = T(N)/\pi_A$ since π_A is the fraction of time that the faulty system is operational. From our definition of speed-up in a faulty environment,

$$\bar{S}(N) = \frac{T(1)}{T^*(N)} = \pi_A \frac{T(1)}{T(N)} = \pi_A S(N) \quad (10)$$

where $S(N)$ is the fault-free speed-up; that is, the speed-up in the faulty environment is reduced by a factor equal to the system availability. After substituting for α^{opt} (from 9) into the availability equation (8), equation 10 becomes:

$$\bar{S}(N) = \frac{S(N)}{1 + 2\sqrt{N\phi U(N)/\beta} + N\phi(\rho^{-1}+r)} \quad (11)$$

Figure 3 shows the speed-up curves for N=4 to N=64 processors for the parameters shown in Table II. Checkpoint and state restoration times are representative of times for writing and reading a few megabytes of data from moveable head disk and synchronizing the checkpoint and recovery activities. The mean failure time of 10^5 seconds is a typical value [11], while the mean repair times of 10^3 seconds and 10 seconds represent two short repair times. The dashed curve is the failure-free speed-up curve with exponential form (constants b=128 and a=1/128). The lower solid curve corresponds to the case when the mean repair time is $\rho^{-1}=10^3$ seconds. The other solid curve corresponds to the case when the mean repair time is smaller by two orders of magnitude.

Param.	Value	Description
β^{-1}	1 sec	Mean checkpoint time
a,b	1/128,128	Exponential fault-free speed-up params.
r	1 sec	Mean state restoration time
α	Optimum	Mean checkpoint rate
ϕ^{-1}	10^5 sec	Mean failure time
ρ^{-1}	{10^3,10} sec	Mean repair time

Table II. Model I Parameter Values.

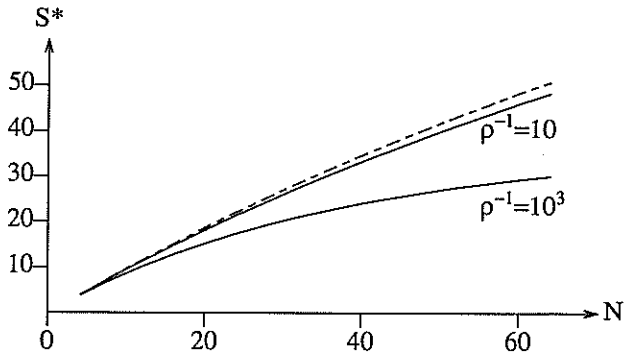

Figure 3. Speed-Up Versus Number of Processors (Model I).

4. Model II — synchronous checkpointing with load redistribution

In synchronous checkpointing without load redistribution, long recovery times due to permanent errors which can not be resolved by quick system resets forces the entire system to be idle for large time periods. In this situation, it would make more sense to work around the faulty node and redistribute the load onto operational nodes. But typically, the cost of load redistribution is significant. If the faulty node becomes operational immediately after the load is redistributed, it would have been better to wait for the faulty node to be repaired and not redistribute the load. Furthermore, in our analysis, the checkpoint interval is allowed to be dependent on the number of operational nodes. This section quantifies the overheads that justify a load redistribution and determines the optimum checkpoint intervals.

For homogeneous nodes, the system state can be defined in terms of the vector $S = (n_A, n_C, n_R, n_F)$ where the components correspond to the number of nodes in the states A, C, R, and F. These parameters are are shown in Table III. The state transition rate diagram is shown in Figure 4. In the available state $(N-k,0,0,k)$ (column 2, Figure 4), $N-k$ nodes are operational and k nodes are being repaired. When checkpointing is initiated from this state, the system enters the checkpointing state $(0,N-k,0,k)$ (column 4). After checkpointing, the system reenters the available state $(N-k,0,0,k)$. When a failure occurs, the system enters the state $(0,0,N-k-1,k+1)$ (column 1) where it downsizes (redistributes the load down to the $N-k-1$ operational nodes). After load redistribution, the system enters the available state $(N-k-1,0,0,k+1)$ (column 2) where it resumes computation with one less processor. Nodes are repaired (in parallel) while in state $(N-k,0,0,k)$. After a repair, the system is in state $(0,0,N-k+1,k-1)$ where the system takes a checkpoint and then upsizes (redistributes the load among the $N-k+1$ operational nodes).

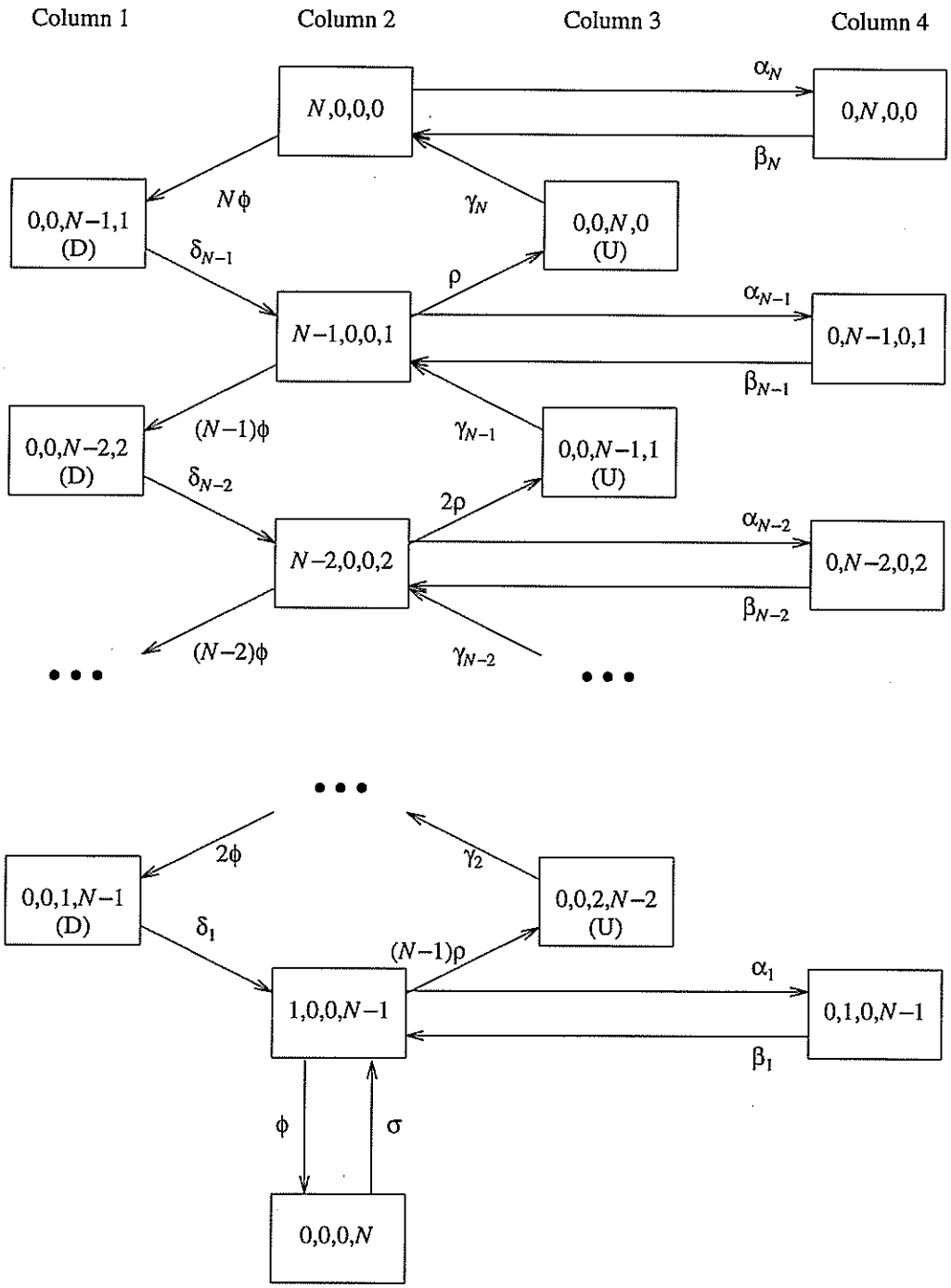

Figure 4. State Transition Rate Diagram (Synchronous Checkpointing With Load Redistribution).

Param.	Description
n_A	number of nodes in available state
n_C	number of nodes in checkpointing state
n_R	number of nodes in load redistribution state
n_F	number of failed nodes

Table III. Markov State Components.

In one case, the load is not redistributed after a failure. This occurs when there is only one operational node which then fails. In this case, the load can not be redistributed and the system must be repaired and lost work must be repeated.

The parameters in the state transition rate diagram are summarized in Table IV. The mean rates ϕ, and σ have the same interpretations as in the single-node case. In the multi-node model, the failure rate while in state $(N-k,0,0,k)$ is $(N-k)\phi$ since each of the $N-k$ operational nodes fail at a mean rate of ϕ. The mean rate α_k, $k=1,...,N$, is the multiprocessor equivalent to α and are the mean intercheckpoint rates when there are k operational nodes. The mean rate β_k, $k=1,...,N$, is the multiprocessor equivalent to β and are the mean checkpoint rates when there are k operational nodes. The two rates δ_k, $k=1,...,N-1$, and γ_k, $k=2,...,N$, are the recovery rates associated with downsizing (omitting a failed node) and upsizing (including a repaired node) respectively. During downsizing, the load must be redistributed, the state must be reloaded, and the work lost since the latest checkpoint must be repeated. The mean downsizing time is

$$\delta_k^{-1} = d_k + r_k + U(k+1)\alpha_{k+1}^{-1}, \qquad k=1,...,N-1 \quad (12)$$

where the first term is the mean load redistribution time, the second term is the mean state restoration time, and the third term is the mean recomputation time. During upsizing, a checkpoint is made and then the load is redistributed. The mean upsizing time is

$$\gamma_k^{-1} = \beta_k^{-1} + g_k + r_k, \qquad k=2,...,N \quad (13)$$

Param.	Description
N	Number of processors
k	Number of operational nodes
β_k^{-1}	Mean checkpoint time
$U(k)$	Fault-free utilization
r_k	Mean state restoration time
α_k	Optimum mean checkpoint rate
ϕ	Mean failure rate of a single node
ρ	Mean repair rate of a single node
d_k	Mean redistribution time (downsize)
g_k	Mean redistribution time (upsize)

Table IV. Model II Parameters.

Appendix I shows how the state probabilities can be computed using local balance equations and probability conservation.

Unlike the prior case, this system can continue to operate with failed nodes. However, when k nodes are under repair, the speed-up will be less than the N node case. Thus, our performance measure should account for variations in the speed-up due to node failures. We define the average speed-up to be

$$\bar{S}(N) = \sum_{k=1}^{N} S(k)\pi_{k,0,0,N-k} = \sum_{k=1}^{N} kU(k)\pi_{k,0,0,N-k} \quad (14)$$

where U(k) is the fault-free utilization when k nodes are operational. Appendix I shows that the average speed-up can be derived as

$$\bar{S}(N) = \pi_{N,0,0,0}\left[\sum_{k=1}^{N} U(k)\binom{N}{N-k}\left(\frac{\phi}{\rho}\right)^{N-k}\right] \quad (15)$$

and the optimum checkpoint interval when there are k operational nodes is

$$\alpha_k^{opt} = \sqrt{k\phi\beta_k U(k)}, \qquad k=1,...,N \quad (16)$$

Thus the kth state-dependent checkpoint rate is identical to the optimum checkpoint rate of a system with k nodes using synchronous checkpointing without load redistribution.

Figure 5 shows the speed-up curves for $N=4$ to $N=64$ processors for the parameters shown in Table V. The parameters are the same as in Table II, but augmented by the load redistribution parameters d_k and g_k. For this example, we assumed that the mean checkpoint time ($\beta_k=\beta$), and the mean load redistribution times d_k and g_k are independent of the number of operational nodes. The dashed curve is the failure-free speed-up curve with exponential form (constants $b=128$ and $a=1/128$). The two solid curves are almost identical and indicate speed-ups which are not significantly worse than the one in a fault-free environment. This contrasts with the result in Model I where a longer mean repair time affected the performance substantially. Because Model II allows work to continue after redistributing the load, the affect of faulty nodes has a smaller impact on performance than in Model I.

Parameter	Value	Description
d_k	10 *sec*	Mean redistribution time (downsize)
g_k	10 *sec*	Mean redistribution time (upsize) ($g_k=d_k$)
Others		See Table II

Table V. Example Model II Parameters.

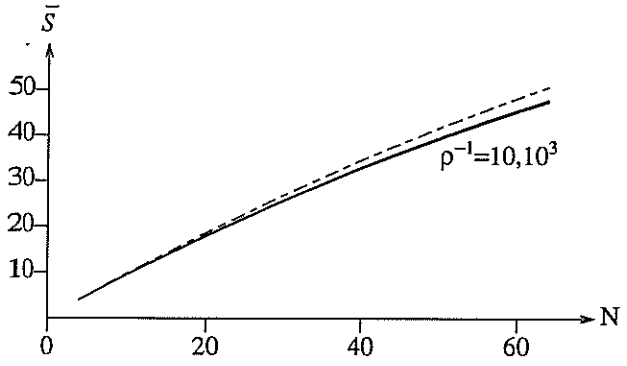

Figure 5. Speed-Up Versus Number of Processors (Model II).

5. An example

This section presents an example indicating how the expressions derived can be used to determine whether or not to employ load redistribution techniques. Table VI shows the parameter values which are used in this section's examples. In order to factor out the fault-free behavior from the examples, the fault-free utilization is assumed to be 1 ($U(k)$=1). Although $U(k)$ affects the absolute performance of the two algorithms, it will not affect their relative performance. Bracketed values in column two indicate that parameter values have been chosen from the interval indicated in brackets. The parameter values have been chosen as before to represent both typical and pedagogic cases. Both N=8 nodes and N=64 nodes have been evaluated to examine the effect of node population on performance.

Param.	Value	Description
N	[8,64]	Number of processors
β_k^{-1}	1 sec	Mean checkpoint time
$U(k)$	1.0	Fault-free utilization
r_k	1 sec	Mean state restoration time
α_k	Optimum	Mean checkpoint rate
ϕ	$[10^{-6},10^{-5}]$ sec^{-1}	Mean failure rate
ρ	$[10^{-4},10^{-1}]$ sec^{-1}	Mean repair rate
d_k	$[1,10^{+2}]$ sec	Mean redistribution time (downsize)
g_k	$[1,10^{+2}]$ sec	Mean redistribution time (upsize)

Table VI. Model II Parameter Values.

In order to focus on the fractional difference between the performance of the two algorithms, we use the ratio of the average speed-up $\bar{S}(N)$ to the ideal speed-up (N) as our performance measure and refer to this measure as the *efficiency* ε. In Model I (without load redistribution), the average speed-up is the speed-up given by Equation 11 since the system is only operational when all N processors are operational. However, in Model II (with load redistribution), the average speed-up is given by Equation 15.

$$\varepsilon = \frac{\bar{S}(N)}{N} \tag{17}$$

Figures 6a and 6b show the effect of the mean repair rate ρ on the efficiency ε for different values of the mean load redistribution time d_k=d and N=8 nodes. The solid (dashed) curves indicate the model II (I) efficiencies. Note that the horizontal axis is a log scale where ρ has been varied over four orders of magnitude.

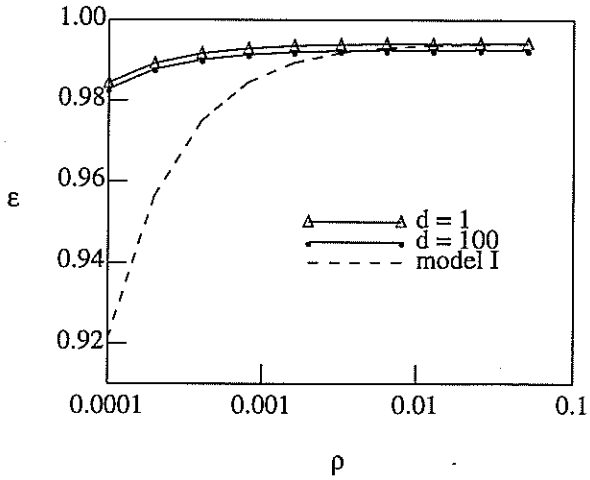

Figure 6(a). Efficiency (ε) Versus Repair Rate (ρ) (N=8 Processors, $\phi = 10^{-6}$ sec^{-1}).

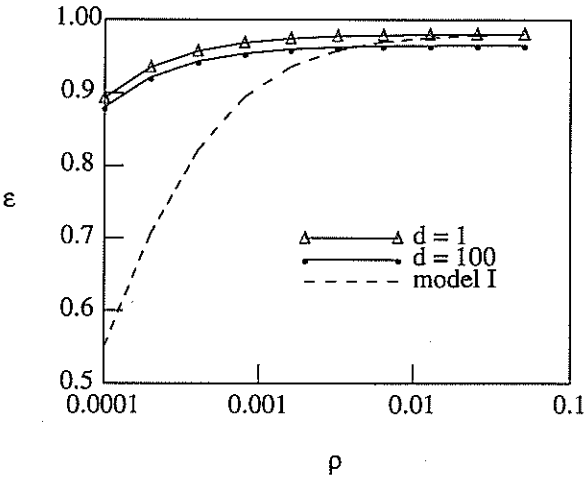

Figure 6(b). Efficiency (ε) Versus Repair Rate (ρ) (N=8 Processors, $\phi = 10^{-5}$ sec^{-1}).

Let ε_I and ε_{II} denote the efficiencies using models I and II respectively. The curves indicate that:

1) When the load redistribution cost is low (small d), the load should be redistributed for greater efficiency (i.e., $\varepsilon_{II} > \varepsilon_I$) since the system is more resilient to failures.
2) When the load redistribution cost is high (large d), the best strategy depends on the repair rate. The costs of

load redistribution must be offset by an increase in availability. Load redistribution is better ($\varepsilon_{II} > \varepsilon_I$) when repairs are slow (small ρ). Load redistribution is not better ($\varepsilon_{II} < \varepsilon_I$) when repairs are fast (large ρ).

3) The efficiency of model II is fairly insensitive to the repair rate ρ while the opposite is true of the efficiency of model I. Failures requiring a large repair time (small repair rate) are disastrous when the load can not be redistributed.

The curves also seem to suggest we should always redistribute the load since the efficiency of model I is never significantly better than that of model II. However, our next example shows that this is not always the case.

Figures 7a and 7b show the efficiency curves for the same parameter values used in the preceding example except that there are N=64 nodes instead of N=8 nodes.

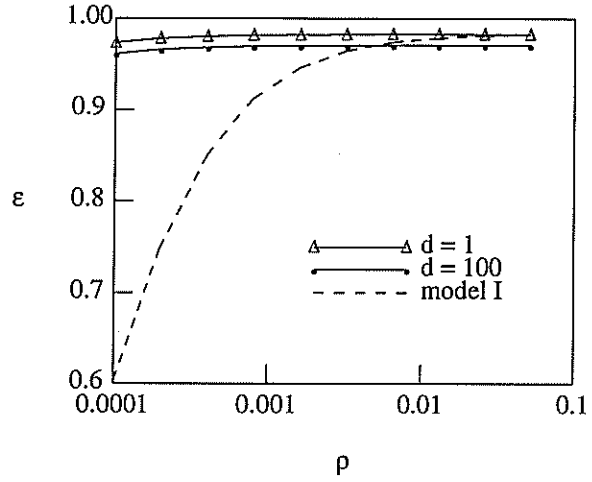

Figure 7(a). Efficiency (ε) Versus Repair Rate (ρ) (N=64 Processors, $\phi = 10^{-6}$ sec^{-1}).

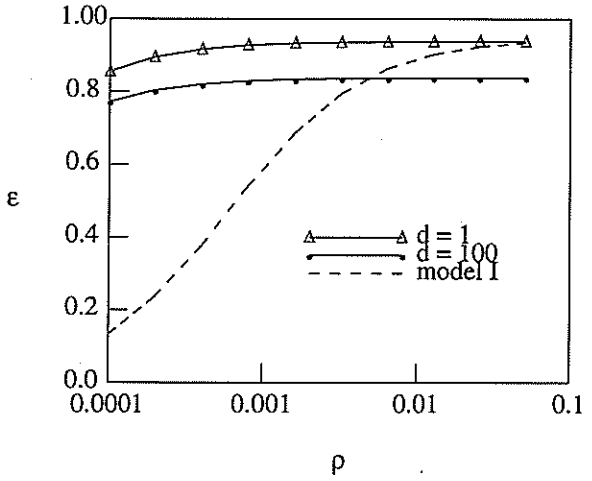

Figure 7(b). Efficiency (ε) Versus Repair Rate (ρ) (N=64 Processors, $\phi = 10^{-5}$ sec^{-1}).

The curves have the same general shapes as before but the details indicate additional features:

1) The efficiency in both models is less than before. This is due to the higher over-all system failure rate caused by a greater number of failure sources (nodes).
2) The repair rate has a more dramatic effect on the efficiency in model I than model II. For example, in order to have an efficiency in model I that is at least 0.80, the mean repair time can be no more than 250 sec (25 sec) when the failure rate is 10^{-6} sec^{-1} (10^{-5} sec^{-1}).
3) The efficiency in model II is less affected by the increased node population. It drops to only 0.80 even when the failure rate is high (10^{-5} sec^{-1}) and repair rate is low (10^{-4} sec^{-1}).
4) The load redistribution time d must be kept low in order to maintain high efficiency in model II, especially when failure rates are high. For example, when the failure rate is 10^{-5} sec^{-1}, there is a 10% difference between the efficiencies when d=1 sec and d=100 sec.

6. Conclusions and further research

The proper checkpointing strategy will be necessary to maintain high efficiency in long-running, massively parallel applications which are subject to failures. We have analyzed two checkpoint/recovery strategies for non-transaction-oriented systems: synchronous checkpointing with and without load redistribution. Optimum checkpoint rate(s) were determined analytically and have familiar forms. The models indicate operational regions where one is preferred over the other. In particular, synchronous checkpointing without load redistribution will have limited use in large node population applications because of its lack of resilience to failures. Synchronous checkpointing with load redistribution is more resilient to failures, but the load redistribution and checkpointing overheads must be kept small to maintain high efficiency. This becomes even more imperative with ever increasing node populations.

Although the material presented here assumes a symmetric system in which all nodes act identically (in the stochastic sense), it is a straightforward extension to handle the heterogeneous case or to add more state dependent considerations to the model (e.g., mean checkpoint time). Another related issue that can also be addressed in our framework is the distribution of tasks in a heterogeneous system.

Our models could easily be extended to include the possibility for handling transient and permanent failures differently. A transient failure is one in which the repair time is measured in seconds (e.g., software/hardware reset) whereas a permanent failure requires a much longer repair time which is measured in hours or even days. The response to a permanent failure might be to redistribute the load whereas the response to a transient failure might be to wait for recovery without load redistribution.

We are also in the process of modeling specific checkpointing strategies [12,13] and exploring alternative asynchronous strategies which take local snapshots and allow for partial operation during checkpoint periods.

The models presented in this paper characterize the system architecture explicitly and the task assignment indirectly through the fault-free speed-up curve. Another approach is to model the computational load by a task graph explicitly and the system architecture indirectly as affects on arc and/or node costs. We are also developing models using this alternative approach and trying to match the model parameters to real workloads (e.g., logic simulation) and use speed-up curves derived from experimental data. Our hope is that results from both approaches can be used iteratively to produce results which identify the fundamental parameters in determining task allocation in a faulty computation environment.

Appendix I

Optimum Synchronous Checkpointing With Load Redistribution

The Markov chain for Model II obeys local balance equations [14]. These equations can be written using four pairs of arcs for each row of states where row k corresponds to the states with k failed nodes:

1) the arcs coming into and going out of a downsizing recovery state (column 1);
2) the arc going out of an availabile state $(N-k,0,0,k)$ due to a failure and the arc going out of the available state $(N-k-1,0,0,k+1)$ due to a repair;
3) the arcs going into and out of an upsizing recovery state (column 3);
4) arcs going into and out of checkpoint states (column 4).

These arc pairs correspond to surfaces for states in columns 1, 2, 3, and 4 in the state transition diagram respectively. We equate the flows across these pairs of arcs. The solution to these local balance equations also satisfy the global balance equations in which the flow into each state is equal to the flow out of each state.

We first write the state probabilities for the active states (column 2) in terms of $\pi_{N,0,0,0}$. Then, all other state probabilities are written in terms of $\pi_{N,0,0,0}$. For the states in column 2 (excluding state (0,0,0,N)),

$$\pi_{N-k,0,0,k} = \pi_{N,0,0,0}\binom{N}{k}\left(\frac{\phi}{\rho}\right)^k, \quad k=1,\ldots,N-1 \tag{18}$$

For the state (0,0,0,N), and then the states in columns 1, 3, and 4:

$$\pi_{0,0,0,N} = \pi_{N,0,0,0}\left(\frac{\phi}{\rho}\right)^N \frac{N\rho}{\sigma} \tag{19}$$

$$\pi^{(D)}_{0,0,N-k,k} = \pi_{N,0,0,0}\binom{N}{k}\left(\frac{\phi}{\rho}\right)^k \frac{k\rho}{\delta_{N-k}}, \quad k=1,\ldots,N-1 \tag{20}$$

$$\pi^{(U)}_{0,0,N-k,k} = \pi_{N,0,0,0}\binom{N}{k}\left(\frac{\phi}{\rho}\right)^k \frac{(N-k)\phi}{\gamma_{N-k}}, \quad k=0,\ldots,N-1 \tag{21}$$

$$\pi_{0,N-k,0,k} = \pi_{N,0,0,0}\binom{N}{k}\left(\frac{\phi}{\rho}\right)^k \frac{\alpha_{N-k}}{\beta_{N-k}}, \quad k=0,\ldots,N-1 \tag{22}$$

Using probability conservation, we can solve for $\pi_{N,0,0,0}$. The details can be found in [15]. Since all state probabilities have been written in terms of $\pi_{N,0,0,0}$ above, we have solved for all state probabilities.

The problem now is to find checkpoint rates α_k for $k=1,\ldots,N$ operational nodes which will optimize the average speed-up $\bar{S}(N)$.

$$\bar{S}(N) = \sum_{k=1}^{N} kU(k)\pi_{k,0,0,N-k}$$

$$= \pi_{N,0,0,0}\left[\sum_{k=1}^{N} kU(k)\binom{N}{N-k}\left(\frac{\phi}{\rho}\right)^{N-k}\right] \tag{23}$$

A necessary (but not sufficient) condition for optimality of the checkpoint rates is that the partial derivatives of $\bar{S}(N)$ with respect to the checkpoint rates be zero. But since $\pi_{N,0,0,0}$ is the only term in the $\bar{S}(N)$ expression that depends on the checkpoint rates α_k, the requirement for optimality is equivalent to finding the roots of the N equations:

$$\frac{d\,\pi_{N,0,0,0}}{d\,\alpha_k} = 0, \qquad k=1,\ldots,N \tag{24}$$

Note that the recovery rates are state dependent. Recovery from the states in column 3 in which there were $N-k$ operational nodes $(k=2,\ldots,N)$ involves checkpointing and then redistributing the load (including reloading the state); that is,

$$\gamma_k^{-1} = \begin{cases} 0, & k=1 \\ \beta_k^{-1} + g_k + r_k, & k=2,\ldots,N \end{cases} \tag{25}$$

The parameters g_k and r_k, $k=1,\ldots,N$, are the mean times for load redistribution and state restoration respectively. Recovery from state $(0,0,0,N)$ involves repairing the system, reloading the state, and then repeating work lost since the latest checkpoint.

$$\sigma^{-1} = \rho^{-1} + r_1 + U(1)\alpha_1^{-1} \tag{26}$$

Recovery from the states in column 1 in which there are $N-k$ operational nodes involves redistributing the load (including reloading the state) and then repeating the work lost since the latest checkpoint. The parameter d_k, $k=1,\ldots,N-1$, is the analog to g_k in the expression for γ_k and represents the load redistribution time.

$$\delta_k^{-1} = d_k + r_k + U(k+1)\alpha_{k+1}^{-1}, \qquad k=1,\ldots,N-1 \tag{27}$$

After substituting the expressions for γ_k^{-1}, σ^{-1}, and δ_k^{-1} into the equations obtained from taking the N derivatives,

the resulting derivatives can be shown to be quadratic in α_k with one non-negative root in each case. After some algebra, the optimum checkpoint rates can be shown to be

$$\alpha_k^{opt} = \sqrt{k\phi\beta_k U(k)}, \qquad k=1,...,N \tag{28}$$

REFERENCES

1. J.S. Turner, "Design of a Broadcast Packet Switching Network," *IEEE Trans. on Comm.* **36**(6) pp. 734-743 (June 1988).
2. John W. Young, "A First Order Approximation to the Optimum Checkpoint Interval," *Communications of the ACM* **17**(9) pp. 530-531 (Sept. 1974).
3. K. Mani Chandy, James C. Browne, Charles W. Dissly, and Werner R. Uhrig, "Analytic Models for Rollback and Recovery Strategies in Data Base Systems," *IEEE Transactions on Software Engineering* **SE-1**(1) pp. 100-110 (March 1975).
4. E. Gelenbe and D. Derochette, "Performance of Rollback Recovery Systems under Intermittent Failures," *Comm. ACM* **21**(6) pp. 493-499 (June 1978).
5. Erol Gelenbe, "On the Optimum Checkpoint Interval," *Journal of the ACM* **26**(2) pp. 259-270 (Apr. 1979).
6. Asser N. Tantawi and Manfred Ruschitzka, "Performance Analysis of Checkpointing Strategies," *ACM Transactions on Computer System* **2**(2) pp. 123-144 (May 1984).
7. Kang G. Shin, Tein-Hsiang Lin, and Yann-Hang Lee, "Optimal Checkpointing of Real-Time Tasks," *IEEE Transactions on Computers* **C-36**(11) pp. 1328-1341 (Nov. 1987).
8. Victor F. Nicola and Johannes M. Van Spanje, "Comparative Analysis of Different Models of Checkpointing and Recovery," *IEEE Trans. Software Engineering* **16**(8) pp. 807-821 (Aug. 1990).
9. Erol Gelenbe, David Finkel, and Satish K. Tripathi, "Availability of a Distributed Computer System with Failures," *Acta Informatica* **23** pp. 643-655 (1986).
10. Kishor Shridharbhai Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications,* Prentice-Hall, Englewood Cliffs, New Jersey (1982).
11. X. Castillo, S. R. McConnel, and D. P. Siewiorek, "Derivation and Calibration of a Transient Error Reliability Model," *IEEE Trans. on Comm.* **C-31**(7) pp. 658-671 (July 1982).
12. Richard Koo and Sam Toueg, "Checkpointing and Rollback-Recovery for Distributed Systems," *IEEE Transactions on Software Engineering* **SE-13**(1) pp. 23-31 (January 1987).
13. Kai Li, Jeffrey F. Naughton, and James S. Plank, "An Efficient Checkpointing Method for Multicomputers with Wormhole Routing," *International Journal of Parallel Processing,* (June 1992).
14. K. Kant, *Introduction to Computer System Performance Evaluation,* McGraw Hill, New York (1992).
15. Ken Wong and Mark Franklin, "Distributed Computing Systems and Checkpointing," WUEE 92-115, Department of Electrical Engineering, Washington University (January 1992).