A review of:

**Evaluating Distributed Checkpointing Protocols**

Checkpoint/Restart (C/R) provides a very basic technique for providing fault tolerance. In this technique, a process periodically takes checkpoints (saves its state). If the process fails, then it can restart from the latest checkpoint avoiding doing the whole work again. The challenge in designing the checkpointing protocols is to ensure low overhead of such schemes.

In a distributed system, the problem is compounded due to the possibility of Domino Effect. This says that if every process takes independent checkpoints, then it is possible that no collection of them forms a consistent state. To circumvent this problem, the processes coordinate while taking checkpoints. These checkpoints are referred to as the forced checkpoints. There are two basic classes of checkpointing protocols:

- Coordinated Checkpointing: Whenever a process has to take a checkpoint, it coordinates with all the other processes in the system and takes a checkpoint to make sure that its checkpoint is part of some consistent global state.
- Communication Induced Checkpointing: In these protocols a process can take local checkpoints but on detection of some dangerous communication pattern a process is forced to take more checkpoints.

In literature many checkpointing protocols have been proposed and it becomes difficult for a user to choose which one is best for his/her application. The authors claim to provide a framework for evaluating the protocols. Their work is an extension of framework developed by Vaidya[N 95] for single processor systems. For adapting it to distributed systems, the authors have accounted for the communication costs and the possibility of rollback.

Every process is modeled as taking a sequence of checkpoints, some of which are independent and the others are forced. To model the fact that in practice processes don't stop their normal execution while taking the checkpoint, the concept of checkpoint overhead and checkpoint latency is used as done by Vaidya[N 95]. Communication overhead of an execution is estimated using the extra information it sends during each checkpoint interval. The number of rollbacks needed to find the recovery line is based on the classification of the executions by [AHFV 01]. Intuitively, this classification categorizes an execution as $k$-rollback if it requires atmost $k$ rollbacks at a process from the most recent checkpoint to find a recovery line.

The measure used to compare different protocols is called the Overhead Ratio which is basically the fractional increase in the execution time of a process due the checkpointing protocol. The overhead ratio for a protocol in $k$-rollback class is denoted by $\Gamma_k$. $\Gamma_k$ is computed using a finite-state Markov chain. We get a closed formula for $\Gamma_k$ in terms of some parameters which depend on the environment and the protocol used.

The model proposed by the authors provides a useful evaluation measure for comparing
different distributed checkpointing protocols. The model is fairly comprehensive and includes the main sources of overheads in a checkpointing protocol. It provides a generic framework for the comparison of the checkpointing protocols as opposed to the previous work which was mainly based on implementation of some protocols [MW 96] [ZB 94].

However, there are still some serious shortcomings of this paper

- A shorter version of this paper has already appeared in ICDCS 2003 [AFF 03]. This paper just adds minor details to that paper. Some of the details added are:
  - Section 2.1.1 about “Practical Considerations” which does not give any useful information.
  - It gives details like equations for computing $T_0, T_1$ etc which are not really needed after a point
  - Gives a description of the protocols for which the comparison was performed using the model suggested. This again does not add any value to the paper itself. Other additions are also of trifling nature. So this “extended” version of the paper conveys almost the same amount of information as the shorter version.

- The paper concludes by saying that a process should try to achieve small rollback propagation and avoid piggybacking messages. This conclusion doesn't require such a complicated model as this is fairly obvious. In fact there is a tradeoff between the two and from a model, one would expect to see how the two factors are correlated. Even though the model used by the authors takes most of the parameters in account, it does not add to our knowledge of the relationship between these two parameters.

- The model used for evaluation of checkpointing protocols assumes that processes take regular independent checkpoints regardless of the forced checkpoints. In [AERHM 99], Alvisi et. al demonstrated that the communication induced checkpointing protocols perform better when the local checkpoints interval is decided dynamically on the basis of the occurrence of the forced checkpoints. Similarly, the model does not take into account things like communication pattern of processes which can affect the performance of the communication induced checkpointing protocols. Although these factors may not turn out to be very important but it still leaves some shadow of doubts over the results of the paper.

- In section 3.3, the value of $r$ is assumed to be constant. Although it makes sense to have the value of $o$ and $L$ as constant for all the protocols but $r$ should differ across protocols. This is because a $k$-rollback ($k > 1$) execution would in all likelihood take more time in searching for a recovery line than a 1-rollback execution. Infact this difference could be significant in many cases and can alter the way the comparison between two protocols stands.

- The use of the approach used by the authors depends upon the classification of a checkpointing protocol in one particular rollback class. This classification may not be
easy for new protocols rendering this method inapplicable. Moreover, it appears that here we are just concerned with the worst case rollbacks of a protocol. Using the worst case bound to estimate the overhead may not give us a reasonable result. To some extent the use of decreasing probabilities takes care of this problem but the ratio of these probabilities is again dependent on the protocol and should not be chosen as constant across the protocols.

A list of detailed comments follows

Detailed Comments

0. Abstract
   • The footer on the first page should have 'shorter' instead of 'shoter'

1. Introduction

2. Preliminaries
   • In the last line of the first paragraph on page 3, the authors state the assumption that processes fail under exponential distribution. However, all the analysis has been done assuming the failures to be governed by a Poisson process.

   • The section 2.1.1 on practical considerations not needed. It does not really tell anything useful about the model or the checkpointing process in general. It unnecessarily breaks the flow between the system model and the definitions.

   • The definitions of control overhead and total checkpoint overhead should also be numbered.

   • The control overhead $M$ is defined for the protocol. So when the authors say “$\text{expectMsgsNum}(P)$ the expected number of control messages in a single checkpoint interval .....” in the 2nd last paragraph of page 5, it appears that we are talking about the total number of messages exchanged in the whole system. However, total checkpoint overhead $O$ is defined per process and uses $M$ as if it is the overhead per process. This seems inconsistent

3. The Overhead Ratio

1. The definition of $r$ in the last paragraph on page 6, appears confusing. The way $r$ has been used suggests that $r$ is the constant amount of overhead incurred by the process whenever it recovers. It does not include the computation that has to be done again by the process. This point should be made clear while defining $r$.

2. In section 3.1, the authors have given all the equations for $\Gamma_0, \Gamma_1, \Gamma_2$ and $\Gamma_3$. This
was really not needed. Equations for just one or two examples would have been enough, although one needed the Markov Chain diagrams for better understanding.

3. On page 12, the last paragraph: “Given two states s,t ...”. There should be a “,” before the sentence “The entry A(i)(j)” instead of a “.”. Moreover, the relationship between s,t and i,j have not been defined. The fact that we have been using i to denote the the current checkpoint compunds the confusion. This mistake is also present in all the rules given on the beginning of the page 11.

4. On page 15, second paragraph, the first line should have a “,” instead of a “.” after “... SMALL and BIG”.

5. The last line of page 15 should have “loss in” instead of “lose of”. Similarly, the next line on page 16 and the last line in 2nd paragraph should also have “loss”.

4. Performance Analysis of Checkpointing Protocols

1. The last line of 2nd paragraph on page 17 should be “...iff there is no Z-path ...” instead of “... iff there is no a Z-path”.

2. In section 4.1.1, the authors say that the protocol requires 8-bit control messages. It is not clear why that is required. The reason for it should have been stated more explicitly.

3. Again in section 4.1.2, it is not clear why do we need an 8-bit marker to distinguish between the runs. We just require one marker per process and that is \( \log_2(n) \).

4. In section 4.1.3, it states that \( M(FDI) = n.MR(E) \). This is incorrect as M is the overhead in terms of time whereas \( n.MR(E) \) denotes the extra information sent. The relation should have been something like \( M(FDI) = n.MR(E).8.w_b \) assuming in the dependency vector each process requires 8 bits.

5. On page 21, first paragraph, the line which says “MR=2048, BQC an BCS” should be “MR=2048, BQC and BCS”.

6. The results don't explain the wide difference between BQC and BCS in figure 14(b). Even if BQC involves a factor of \( n^2 \), the difference between the two looks really wide especially when the difference between SaS and C-L is very less even though they also have a similar interrelationship as BQC and BCS.

5. Related Work

6. Conclusion
1. The authors make a rather tall claim by saying that their model considers all the parameters that affect the execution of the checkpointing protocols. This is incorrect as for example, their model does not consider the recovery cost that is associated with obtaining the recovery line. Moreover, their model does not explicitly take into account things like the communication pattern which affects the performance of communication-induced checkpointing protocols [AERHM 99]. Another type of cost that the model overlooks is that of storing the checkpoints. A \textit{k-rollback} ($k>1$) protocol would have to store more checkpoints than a 1-rollback protocol.

References


