On Zig-zag Theorem and Rollback Dependency

Vijay K. Garg

TR-PDS-1998-013 **November 1998**

Parallel & Distributed Systems group

Department of Electrical & Computer Engineering

University of Texas at Austin

Austin, Texas 78712

On Zig-zag Theorem and Rollback Dependency

Vijay K. Garg*
Electrical and Computer Engineering Department
The University of Texas at Austin,
Austin, TX 78712.

November 18, 1998

Abstract

I present a survey of results on uncoordinated local checkpoints for the purpose of recovery from failures. Most results presented in this paper are known but the proofs and notation are different from the existing literature.

1 Introduction

Consider any distributed computation with n processes $P_1, ..., P_n$ with $n \geq 1$. Each process P_i checkpoints its local state at some intermittent interval giving rise to a sequence of local checkpoints denoted by S_i. We will assume that the initial state and the final state in any process are checkpointed. For any checkpoint c we denote by $pred.c$ the predecessor of the state c in the sequence S_i whenever it exists, i.e. when c is not the initial checkpoint. Similarly, we use $succ.c$ for the successor of the checkpoint d.

Given a set of local checkpoints, X, we say that X is consistent iff $\forall c, d \in X : c||d$. A set of local checkpoints is called global if it contains n checkpoints.

2 Zig-zag Relation

We first tackle the problem of finding a global checkpoint that contains a given set of checkpoints X. A relation called *zig-zag precedes* which is weaker (bigger) than $\rightarrow$ is useful in analysis of such problems. This relation was first defined by Netzer and Xu [NX95]. We use a different but equivalent definition. It is an easy exercise to show that our definition is equivalent to that of Xu and Netzer.

Definition 1 *The relation* zig-zag precedes, *denoted by* $\stackrel{z}{\rightarrow}$, *is the smallest relation which satisfies:*
(Z1) $c \rightarrow d$ *implies* $c \stackrel{z}{\rightarrow} d$
(Z2) $(c \rightarrow e) \wedge (pred.e \stackrel{z}{\rightarrow} d)$ *implies* $c \stackrel{z}{\rightarrow} d$.

*supported in part by the NSF Grants ECS-9414780, CCR-9520540, TRW faculty assistantship award, a General Motors Fellowship, and an IBM grant. http://maple.ece.utexas.edu

The following property of *zig-zag* relation is easy to show
(Z3) $(c \xrightarrow{z} e) \wedge (pred.e \xrightarrow{z} d)$ implies $(c \xrightarrow{z} d)$. Based on the above relation, we say that a set of local checkpoints X is z-consistent iff $\forall c, d \in X : c \not\xrightarrow{z} d$.

The following result is due to Xu and Netzer.

Theorem 1 (Netzer and Xu 95) *Given a set of local checkpoints X, there exists a consistent global checkpoint G containing X iff X is z-consistent.*

Proof: ($\Rightarrow$) We prove the contrapositive. Given c and d in X (possibly $c = d$), we show that $c \xrightarrow{z} d$ implies that there is no consistent global state G containing c and d. We prove a slightly stronger claim. We show that $c \xrightarrow{z} d$ implies that there is no consistent global state G containing c or any checkpoint preceding it and d.

The proof is based on induction on k, the minimum number of applications of $Z2$ rule to derive that $c \xrightarrow{z} d$. When $k = 0$, $c \rightarrow d$. Thus, c or any checkpoint preceding c and d cannot be part of a consistent state by the definition of consistency. Now consider the case when $c \xrightarrow{z} d$ because $\exists e : (c \rightarrow e) \wedge (pred.e \xrightarrow{z} d)$. We show that any consistent set of states Y containing c and d cannot have any checkpoint from the process containing the checkpoint e. Y cannot contain e or any state following e because $c \rightarrow e$ would imply that c causally precedes that state. Further, Y cannot contain any checkpoint previous to e because $pred.e \xrightarrow{z} d$ and the induction hypothesis ($pred.e \xrightarrow{z} d$ must have fewer applications of $Z2$ rule). Since any consistent set of checkpoints cannot contain any checkpoint from the process $e.p$ we conclude that there is no global checkpoint containing c and d.

($\Leftarrow$) It is sufficient to show that if X is not global then there exists Y strictly containing X which is z-consistent. By repeating the process the set can be made global. Further, the set is always consistent since $c \not\xrightarrow{z} d$ implies that $c \not\rightarrow d$. For any process, say P_i, which does not have a checkpoint in X, we define $e = min\{f \in S_i | \forall c \in X : f \not\xrightarrow{z} c\}$ where min is taken over the relation $\prec$. Note that the set over which min is taken is non-empty since the final checkpoint on process P_i cannot zig-zag precede any other checkpoint. We show that $Y = X \cup \{e\}$ is z-consistent. It is sufficient to show that $e \not\xrightarrow{z} e$ and $c \not\xrightarrow{z} e$ for any c in X. Since $e \xrightarrow{z} e$ and $pred.e \xrightarrow{z} c$ imply that $e \xrightarrow{z} c$ we know that $e \xrightarrow{z} e$ is not possible. Similarly, $c \xrightarrow{z} e$ and $pred.e \xrightarrow{z} c$ imply $c \xrightarrow{z} c$ which is false.

■

3 Rollback Dependency Trackability (RDT)

The concept of rollback dependency trackability was first proposed by Yi-Min Wang [Wan97]. If a computation satisfies a condition called rollback dependency trackability, then for every zig-zag path there is also a causal path. In other words, rollback dependency is then trackable by tracking causality. Formally,

Definition 2 *A checkpoint computation is called rollback dependency trackable if for all checkpoints c, d: $c \rightarrow d \equiv c \xrightarrow{z} d$.*

Again our definition is different but equivalent to that by Wang. Given a distributed computation as a trace of Mattern and Fidge's vectors [Mat89, Fid89]for each process, how can we check whether its satisfies the RDT condition ? We give two approaches below. In our discussion we use the following notation: $s.VC$ denote the vector clock in the state s, and $s.p$ denote the process of which s is a part.

3.1 Checking RDT: Using R-graph

The first approach is based on creating R-graph. Our definition of R-graph is slightly different from that used by Wang[Wan97], but has the same essential property - there is a path from c to d in the graph iff there is a zig-zag path from c to d in the computation. Each checkpoint is a node in the graph. There is an edge between two successive checkpoints between the graph. Furthermore, if $c.VC < d.VC$ then there is an edge from $succ.c$ to d. It is sufficient to put at most one edge outgoing from one checkpoint to any other process. That is, for any checkpoint c and any process P_i, different from $c.p$, let d be the minimum state in P_i with $c.VC < d.VC$, if there is such a state. We put an edge from $succ.c$ to d. Now, we directly apply Wang's definition. The computation satisfies RDT iff for any checkpoints c, d, a path from $succ.c$ to d implies $c.VC < d.VC$. This is true due to the following:

A checkpoint computation satisfies RDT

$=$ { definition of RDT }

$\forall c, d : c \rightarrow d \equiv c \stackrel{z}{\rightarrow} d$

$=$ { causal path implies zig-zag path }

$\forall c, d : c \stackrel{z}{\rightarrow} d \Rightarrow c \rightarrow d$

$=$ { definition of R-graph }

$\forall c, d : succ.c$ has a path to $d \Rightarrow c \rightarrow d$

$=$ { definition of vector clock }

$\forall c, d : succ.c$ has a path to $d \Rightarrow c.VC \rightarrow d.VC$.

Assume that there are n processes each with at most m checkpoints. Then, creating the R-graph requires $O(mn^2)$ work. Now to check the RDT condition, we can compute the transitive closure of R-graph which takes $O(m^3n^3)$ time and then check the required condition by cheking it for every edge in the transitively closed graph. This algorithm takes $O(m^3n^3)$ time overall.

3.2 Checking RDT: Using WCP algorithm

From zig-zag consistency theorem, it follows that a checkpoint computation is RDT iff for any two concurrent checkpoints (possibly same) there exists a consistent global state containing them. We use this observation to derive a more efficient algorithm based on an algorithm for weak conjunctive predicate detection [GW94]. We first derive an equivalent condition for RDT which is easier to evaluate. Given any checkpoint c we denote by $conc(c, j)$ the set of checkpoints concurrent with c on process P_j. Since all the checkpoints within a process are totally ordered the minimum of $conc(c, j)$ is well defined whenever it is non-empty. Now we are ready to present the condition.

Lemma 1 *A computation is RDT iff for any checkpoint c and any process j*

(C1) $conc(c, j)$ is non-empty, and

(C2) $min.conc(c, j) \stackrel{z}{\not\rightarrow} c$.

Proof: ($\Rightarrow$) The first conjunct holds due to zig-zag consistency theorem and the second conjunct holds since $\rightarrow$ and $\stackrel{z}{\rightarrow}$ are equivalent for RDT computation.

($\Leftarrow$) Let c and d be any two checkpoints. It is sufficient to show that $d \stackrel{z}{\rightarrow} c$ implies $d \rightarrow c$. We show the contrapositive.
$d \not\rightarrow c$
$\Rightarrow$ { C1, definition of min }
$min.conc(c, j) \preceq d$
$\Rightarrow$ { C2, definition of zig-zag relation}
$d \stackrel{z}{\not\rightarrow} c$.

■

Now we exploit the above Lemma as follows. We first focus our attention on any process P_i. For any checkpoint c in that process let $L(c)$ be the least and the greatest consistent global states containing c. If there is no consistent cut containing c, then the computation is not RDT. Otherwise, it is sufficient to verify that $L(c)[j] = min.conc(c, j)$. This verification needs to be done for all checkpoints in c. $L(c)$ can be computed using WCP algorithm in $O(n^2 m)$ time. Since $c_1 \preceq c_2$ implies that $L(c_1) \leq L(c_2)$, by a single application of WCP algorithm, the condition can be checked for all the checkpoints within a single process. Thus, the entire computation can be checked for RDT condition in $O(n^3 m)$.

An $O(n^3 m)$ algorithm can also be derived based on the following condition to check RDT.

Lemma 2 *A computation is RDT iff for any checkpoint c*
(D1) $\forall j : conc(c, j)$ is non-empty, and
(D2) $\forall j, k : min.conc(c, j) \not\rightarrow min.conc(c, k)$.

Proof: We show that the conditions C2 and D2 are equivalent. We first show that the condition C2 implies D2, or equivalently that violation of D2 leads to violation of C2. If there exists j and k such that $min.conc(c, j) \rightarrow min.conc(c, k)$, then this combined with the fact that $pred.min.conc(c, k) \rightarrow c$ gives us $min.conc(c, j) \stackrel{z}{\rightarrow} c$.

We now show that D2 implies C2. From D2, it follows that the global state consisting of $min.conc(c, j)$ from each process is a consistent global state consisting of c and $min.conc(c, j)$ for each j. By zig-zag consistency theorem it follows that $min.conc(c, j) \stackrel{z}{\not\rightarrow} c$

■

We observe that if the vector clock is incremented only on checkpoints, then given a checkpoint c, $c.VC[j] + 1$ gives minimum check point on process j which does not causally precede c. Further, $d \rightarrow c$ is equivalent to $d.VC[d.p] < c.VC[d.p]$.

References

[Fid89] C. J. Fidge. Partial orders for parallel debugging. *Proceedings of the ACM SIGPLAN/SIGOPS Workshop on Parallel and Distributed Debugging, published in ACM SIGPLAN Notices*, 24(1):183–194, January 1989.

[GW94] V. K. Garg and B. Waldecker. Detection of weak unstable predicates in distributed programs. *IEEE Transactions on Parallel and Distributed Systems*, 5(3):299–307, March 1994.

[Mat89] F. Mattern. Virtual time and global states of distributed systems. In *Parallel and Distributed Algorithms: Proc. of the International Workshop on Parallel and Distributed Algorithms*, pages 215–226. Elsevier Science Publishers B.V. (North-Holland), 1989.

[NX95] R. H. B. Netzer and J. Xu. Necessary and sufficent conditions for consistent global snapshots. *IEEE Transactions on Parallel and Distributed Systems*, 6(2):165–169, February 1995.

[Wan97] Y. M. Wang. Consistent global checkpoints that contain a given set of local checkpoints. *IEEE Transactions on Computers*, 46(4), April 1997.