Proceedings of the World Congress on Engineering 2008 Vol I
WCE 2008, July 2 - 4, 2008, London, U.K.

Transaction-Consistent Global Checkpoints in a Distributed Database System

Jiang Wu, D. Manivannan and Bhavani Thuraisingham *

Abstract—**Checkpointing and rollback recovery are well-known techniques for handling failures in distributed database systems. In this paper, we establish the necessary and sufficient conditions for the checkpoints on a set of data items to be part of a transaction-consistent global checkpoint of the distributed database. This can throw light on designing efficient, non-intrusive checkpointing techniques and transparent recovery techniques for distributed database systems.**

Keywords: Checkpointing, Recovery, Fault-tolerance, Distributed databases

1 Introduction

It is a common practice to take checkpoints of a database from time to time, and restore the database to the most recent checkpoint when a failure occurs. In order to maintain the atomicity (explained in Section 2) of transactions, it is desirable that a checkpoint records a state of the database which reflects the effect of a set of completed transactions and not the results of partially executed transactions. Such a checkpoint of the database is called a transaction-consistent checkpoint[1]. A straightforward way to take a transaction-consistent checkpoint of a distributed database is to block all newly submitted transactions and wait till all the currently executing transactions finish and then take the checkpoint. Such a checkpoint is guaranteed to be transaction-consistent, but this approach is not practical, since blocking newly-submitted transaction will dramatically increase transaction response time and this is not acceptable for the users of the database. A more efficient way would be to save (checkpoint) the state of each data item independently and periodically without blocking any transaction. However if each data item is checkpointed independently and periodically, the checkpoints of a data item may not be part of any transaction-consistent global checkpoint of the database and hence are useless.

In this paper, we address this issue and establish the necessary and sufficient conditions for a checkpoint of a data item (or the checkpoints of a set of data items) to be part of a transaction-consistent global checkpoint of the database. This result would be useful for constructing a transaction-consistent global checkpoint incrementally from the checkpoints of each individual data item. This is because, by applying this condition, we can start from an useful checkpoint of a data item and then can incrementally add checkpoints of other data items until we get a transaction-consistent checkpoint of the database. This can also throw light on designing efficient non-intrusive distributed checkpointing algorithms.

1.1 Motivation and Objectives

In a distributed system, to minimize the lost computation due to failures, the state of the processes involved in a distributed computation are periodically checkpointed. When one or processes involved in a distributed computation fails, the processes are restarted from a previously saved consistent global checkpoint. When process are independently checkpointed, the checkpoints taken may not be part of any consistent global checkpoint and hence are useless [2]. Netzer and Xu established the necessary and sufficient conditions for the checkpoint of a process to be useful. Netzer and Xu [2] introduced the notion of zigzag paths between checkpoints of processes involved in the computation. They proved that a checkpoint of a process is useful if and only if there is no zigzag path from that checkpoint to itself. In this paper, we generalize this result to database systems.

1.2 Organization of the paper

The remainder of this paper is organized as follows. In Section 2 we introduce the necessary background required for understanding the paper. Section 3 discusses related works. In Section 4 we present the necessary and sufficient conditions for a set of checkpoints on a set of data items to belong to a transaction consistent global checkpoint and prove its correctness. Section 5 concludes the

*This material is based in part upon work supported by the US National science Foundation under Grant No. IIS-0414791 and the US Department of Treasury Award #T0505060. 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 or the Department of Treasury. Jiang Wu and D. Manivannan are with the Department of Computer Science, University of Kentucky, Lexington, KY 40506 Tel/Fax: 859-257-9234/323-3740 Email:{wujiang,mani}@cs.uky.edu. Bhavani Thuraisingham is with the Department of Computer Science, University of Texas at Dallas Tel/Fax: 972-883-4738/2349 Email:bhavani.thuraisingham@utdallas.edu.

paper and also presents the future work.

2 Background

2.1 System model

We consider a model of distributed database system similar to the model in [1]. In this model, a *distributed database system* consists of a set of data items residing at various sites. Sites can exchange information via messages transmitted on a communication network, which is assumed to be reliable. The data items of the database are accessed by transactions and the transactions are controlled by transaction managers (TM) that reside on these sites. The TM is responsible for the proper scheduling of transactions using appropriate concurrency control algorithms in such a way that the integrity of the database is maintained. In addition, the data items at each site are controlled by a data manager (DM). Each DM, which is responsible for controlling access to items at its site. Each data item is checkpointed by a transaction periodically. Before a transaction takes a checkpoint of a data item it obtains an exclusive lock on the data item so no other transaction can be accessing that data item while it is checkpointed. The state of a data item changes when a transaction accesses that data item for a write operation. In order to guarantee the integrity and efficiency of transaction processing, four requirements referred to as ACID [4] must be maintained.

- Atomicity: Each transaction is executed in its entirety, or not at all executed.
- Consistency preservation: Execution of a transaction in isolation (that is, with no other transaction executing concurrently) preserves the consistency of the database.
- Isolation: Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions T_i and T_j, it appears to T_i that either T_j finished execution before T_i started, or T_j started execution after T_i finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.
- Durability: After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

In order to maintain ACID requirements and achieve maximum performance, a proper schedule of transactions need to be arranged in which the operations of various transactions are interleaved as much as possible. Given a schedule, a directed graph, referred to as precedence graph [5] or serialization graph [4], can be constructed to illustrate the procedure of all the transactions running in the database system. The serialization graph serves as an important tool to analyze transaction processing in the distributed database systems.

Each checkpoint on a data item is assigned a unique sequence number. We assume that the database consists a set of n data items $\mathbf{X} = \{x_i \mid 1 \leq i \leq n\}$. In addition, we denote by $C_i^{k_i}$ the checkpoint on x_i with sequence number k_i. The set of all checkpoints on data item x_i is denoted by $\mathbf{C_i} = \{C_i^{k_i} \mid k_i : k_i \geq 0\}$. The initial state of data item x_i is represented by checkpoint C_i^0 and a virtual checkpoint $C_i^{virtual}$ represents the last state obtained after termination of all transactions accessing data item x_i. A data item is checkpointed only after the state of the data item changes. That is, after a data item is checkpointed, it is not checkpointed again until at least one other transaction has accessed and changed the data item.

Let $\mathbf{T} = \{T_i \mid 1 \leq i \leq m\}$ be a set of transactions that access the database system. In order to analyze the relationship between checkpoints of various data items, we assume that each checkpoint of a data item x_i is taken by a special transaction, called *checkpointing transaction*. We denote by $T_{C_i^{k_i}}$ the checkpointing transaction that takes checkpoint $C_i^{k_i}$ of data item x_i. In order to maintain atomicity of transactions, $T_{C_i^{k_i}}$ is the local transaction which is required to be scheduled to access a data item when there are no other transactions accessing the data item. The set of checkpointing transactions that produce the checkpoints $\mathbf{C_i}$ is denoted by $\mathbf{T_{C_i}}$ and the set of all checkpointing transactions in the system is denoted by $\mathbf{T_C}$.

A global checkpoint of the database is a set $\mathbf{S} = \{C_i^{k_i} \mid 1 \leq i \leq n\}$ of local checkpoints consisting of one checkpoint for each data item. The set of checkpointing transactions that produce the global checkpoint $\mathbf{S}$ is denoted by $\mathbf{S_T} = \{T_{C_i^{k_i}} \mid 1 \leq i \leq n\}$. **We use $C_i^{k_i}$ and $T_{C_i^{k_i}}$ interchangeably. Sometimes, when we say a checkpoint of a data item we mean the checkpointing transaction which takes that checkpoint**.

Each regular transaction is a partially ordered set of read and/or write operations (operations are partially ordered because two adjacent read operations in a transaction are not comparable). A checkpointing transaction consists of only one operation (namely the *checkpointing operation*), an operation that is similar to a write operation which requires mutually exclusive access to the data item. Let $R_i(x_j)$ (respectively, $W_i(x_j)$) denote the read (write) operation of T_i on data item $x_j \in \mathbf{X}$ and $O_{C_j^{k_j}}(x_j)$ denote the checkpointing operation of $T_{C_j^{k_j}}$ on data item x_j. A schedule ε over $\mathbf{T} \bigcup \mathbf{T_C}$ is a family of disjoint sets of partially ordered operations of transactions in $\mathbf{T} \bigcup \mathbf{T_C}$ on the data items (one set for each data item) [1]. Let $\varepsilon(x_j)$ consist of all read, write and checkpointing operations

on x_j of all transactions in $\mathbf{T} \bigcup \mathbf{T_C}$. We denote by $<_{x_j}$ the partial order induced by all read, write, and checkpointing operations on x_j by the schedule ε over $\mathbf{T} \bigcup \mathbf{T_C}$. Given a schedule ε over $\mathbf{T} \bigcup \mathbf{T_C}$, we define the relation $<_T$ between transactions in $\mathbf{T} \bigcup \mathbf{T_C}$ with respect to a schedule ε as follows:

1) $T_i <_T T_j \Leftrightarrow (i \neq j) \wedge (\exists x_k \in X : (R_i(x_k) <_{x_k} W_j(x_k)) \vee (W_i(x_k) <_{x_k} W_j(x_k)) \vee (W_i(x_k) <_{x_k} R_j(x_k)))$.

2) $T_i <_T T_{C_j^{k_j}} \Leftrightarrow (W_i(x_j) <_{x_j} O_{C_j^{k_j}}(x_j)) \vee (R_i(x_j) <_{x_j} O_{C_j^{k_j}}(x_j))$.

3) $T_{C_i^{k_i}} <_T T_j \Leftrightarrow (O_{C_i^{k_i}}(x_i) <_{x_i} W_j(x_i)) \vee (O_{C_i^{k_i}}(x_i) <_{x_i} R_j(x_i))$.

A schedule is serial if the operations belonging to each transaction appear together in the schedule [4]. A schedule ε is serializable if the schedule has the effect equivalent to a schedule produced when transactions are run serially in some order. The concurrency control algorithm ensures that a schedule of transactions running in the distributed database system is serializable. One important kind of serialization, called conflict serializability (CSR) [4] is considered in this paper. An execution $\varepsilon \in CSR$ iff the relation $<_T$ is acyclic. A serialization order of a set of transactions with respect to a schedule ε over $\mathbf{T}$ is defined as a linear ordering of all the transactions such that if $T_i <_T T_j$ (either T_i or T_j could be checkpointing transaction), then T_i must appear before T_j in the ordering. If $\varepsilon \in CSR$, there must exist a serialization order for ε over $\mathbf{T}$ that is compatible with $<_T$.

Formal definition of a *transaction consistent* global checkpoint follows [1]:

Definition 1 *A global checkpoint of a distributed database system is said to be transaction-consistent (tr-consistent or simply consistent, for short) with respect to a set of transactions* $\mathbf{T}$ *if there exists a serialization order (which is a sequence of transactions)* $\sigma_1\sigma_2$ *for an execution* $\varepsilon \in CSR$ *of* $\mathbf{T}$ *such that the data item states represented by the global checkpoint is the same as those read by a read-only transaction* T_{CP} *after all transactions in* σ_1 *have finished execution and before any transaction in* σ_2 *has started execution.*

If the concurrency control algorithm guarantees an execution $\varepsilon \in CSR$, then the relation $<_T$ induces a directed acyclic graph (Dag) on $\mathbf{T} \bigcup \mathbf{T_C}$ and conversely. We call this graph the *global serialization graph* with respect to the schedule ε of $\mathbf{T} \bigcup \mathbf{T_C}$. For each data item, the transactions accessing that data item induce a component of the global serialization graph. The *local serialization graph* induced by the transactions in $\mathbf{T} \bigcup \mathbf{T_{C_i}}$ accessing data item x_i is denoted by $G_{x_i}(V_{x_i}, E_{x_i})$; the vertex set $V_{x_i} = \{T_k \cup T_{C_i^{k_i}} \mid T_k \in \mathbf{T}$ has accessed data item x_i; $C_i^{k_i}$ is the checkpoint of x_i taken by local checkpoint transaction $T_{C_i^{k_i}}\}$ and the edge set $E_{x_i} = \{E_{x_i}^{TT} \cup E_{x_i}^{TC} \cup E_{x_i}^{CT}\}$, where

1: $E_{x_i}^{TT} = \{(T_i, T_j) \mid T_i, T_j \in V_{x_i}; T_i <_T T_j\}$.

2: $E_{x_i}^{TC} = \{(T_j, T_{C_i^{k_i}}) \mid T_j, T_{C_i^{k_i}} \in V_{x_i};\ T_j <_T T_{C_i^{k_i}}\}$.

3: $E_{x_i}^{CT} = \{(T_{C_i^{k_i}}, T_j) \mid T_j, T_{C_i^{k_i}} \in V_{x_i};\ T_{C_i^{k_i}} <_T T_j\ \}$.

By merging the local serialization graphs $G_{x_i}(V_{x_i}, E_{x_i})$, we can construct the global serialization graph $G(V, E)$ where

$$V = \bigcup_{x_i \in X} V_{x_i}$$

and

$$E = \bigcup_{x_i \in X} E_{x_i}$$

We use the following notations throughout the paper: $T_i \longrightarrow^+ T_j$ iff there is a path from transaction T_i to T_j (T_i and/or T_j could be a checkpointing transaction). $T_i \longrightarrow T_j$ iff there is an edge from T_i to T_j (T_i or T_j could be a checkpointing transaction). Let $\sigma_1 \subseteq \mathbf{T}$ and $\sigma_2 \subseteq \mathbf{T}$ be such that $\sigma_1 \bigcap \sigma_2 = \phi$. Then by $\sigma_1 \mathbf{S_T} \sigma_2$ with respect to the serialization order induced by conflict-serializable execution ε over $\mathbf{T}$, we mean that each checkpointing transaction in $\mathbf{S_T}$ is executed after every transaction in σ_1 has been executed and before any transaction in σ_2 has started execution. In particular, if $\sigma_1 \bigcup \sigma_2 = \mathbf{T}$, then the set of checkpoints $\mathbf{S}$ taken by $\mathbf{S_T}$ is tr-consistent iff $\sigma_1 \mathbf{S_T} \sigma_2$.

Next, we make the following observations:

Observation 1 *For any checkpointing transaction* $T_{C_i^{k_i}}$, *since it accesses the data item* x_i *exclusively,* $T_{C_i^{k_i}}$ *must have a path in the* local serialization graph *either to or from any transaction* T_j *that has accessed* x_i.

Observation 2 *For any checkpointing transaction* $T_{C_i^{k_i}}$, *since it accesses the data item* x_i *exclusively, if* $T_i \longrightarrow^+ T_{C_i^{k_i}}$ *and* $T_{C_i^{k_i}} \longrightarrow^+ T_j$, *then in the* local serialization graph *induced by the operations in* $\mathbf{T} \cup \mathbf{T_C}$ *on the data item* x_i *any path from* T_i *to* T_j *must pass through* $T_{C_i^{k_i}}$.

Observation 3 *In the* local serialization graph *induced by the operations in* $\mathbf{T} \cup \mathbf{T_C}$ *on the data item* x_i, *for any checkpointing transaction* $T_{C_i^{k_i}}$ *and two other transactions* T_i *and* T_j *that have accessed* x_i, *the following holds:*

1. *If* $T_{C_i^{k_i}} \longrightarrow^+ T_j$ *and there exists* $T_i \longrightarrow^+ T_j$ *without any checkpoint along the path in the local serialization graph, then* $T_{C_i^{k_i}} \longrightarrow^+ T_i$.

2. *Similarly, if* $T_i \longrightarrow^+ T_{C_i^{k_i}}$ *and there exists a path* $T_i \longrightarrow^+ T_j$ *from* T_i *to* T_j *without any checkpoint along the path in the local serialization graph, then* $T_j \longrightarrow^+ T_{C_i^{k_i}}$.

Observations 1 and 2 are trivial. Observation 3 holds because in case 1, suppose $T_{C_i^{k_i}} \longrightarrow^+ T_i$ is not true, then $T_i \longrightarrow^+ T_{C_i^{k_i}}$ from Observation 1. Since $T_{C_i^{k_i}} \longrightarrow^+ T_j$, from Observation 2, every path in the local serialization graph from T_i to T_j must pass through $T_{C_i^{k_i}}$ from Observation 2, which contradicts our assumption that there exists a path $T_i \longrightarrow^+ T_j$ without any checkpointing transactions along the path. Similar argument can be used to prove the correctness of case 2 in Observation 3.

We make use of these Observations in the proof of the two important theorems in Section 4.

3 Related Work

The checkpointing algorithms for distributed database systems can be classified as log-oriented and dump-oriented. In log-oriented approach, periodically a dump of the database is taken and also a marker is saved at appropriate places in the log. When a failure occurs, the latest dump is restored and the operations on the log after the dump was taken is applied to the dump until the marker is reached to restore the system to a consistent state. In this approach, proper positioning of the marker in the log will produce a consistent global checkpoint. In the dump-oriented approach, checkpointing is referred to as the process of saving the state of all data items in the database (or taking a dump of the database) in such a way that it represents a transaction-consistent checkpoint of the database. The algorithms proposed in [6, 7, 8, 9] take this approach. The basic idea behind the algorithm in [6] is to divide the transactions into two groups: those before or after the checkpointing process. This algorithm is non-intrusive but requires a copy of the database stored temporarily. This temporary copy is used by transactions that could not be decided whether or not they belong to any of the two groups when the checkpointing process is going on. Pu [7] uses color method (white and black) to distinguish data items that have started checkpointing with data items that have not. Transactions accessing both white and black data items have to be aborted or delayed in order to maintain consistency, which increases transaction response time. Pilarski et al. [8] consider checkpoints as checkpoint transactions, one for each data item. In addition, a checkpoint number (CPN) is associated with each checkpoint. By comparing the CPN, forced checkpoints on data items are taken in order to maintain consistency. The previous two algorithms are coordinated algorithms, in which one process initiates and coordinates the checkpointing activity. The algorithm proposed by Baldoni et al. [9] uses a non-coordinated approach, in which no process initiates checkpointing and each data item is checkpointed independently. Like the algorithm of Pilarski et al. [8], checkpoint numbers are used to synchronize checkpointing process and forced checkpoints are taken to maintain consistency. This algorithm is fully distributed but may may suffer from large checkpointing overhead due to forced checkpoints.

4 Necessary and Sufficient Condition

First, we establish the necessary and sufficient conditions for a set of checkpoints, one from each data item to form a transaction-consistent global checkpoint.

Theorem 1 *Let* $\mathbf{T} = \{T_1, \cdots, T_m\}$ *be a set of transactions accessing the database consisting of* n *data items* $\mathbf{X} = \{x_1, \cdots, x_n\}$. *Assume that each data item is checkpointed by a checkpointing transaction that runs at the site containing the data item. Let* $\mathbf{S} = \{C_i^{k_i} \mid 1 \leq i \leq n\}$ *be a set of checkpoints, one for each data item and let* $\mathbf{S_T} = \{T_{C_i^{k_i}} \mid 1 \leq i \leq n\}$ *be the set of checkpointing transactions that produce* $\mathbf{S}$. *Let* ε *be a schedule over* $\mathbf{T}$. *Then* $\mathbf{S}$ *is a tr-consistent global checkpoint iff there is no path between any two checkpointing transactions* $\in \mathbf{S_T}$ *in the global serialization graph corresponding to the schedule* ε.

Proof: For lack of space we omit the proof of the theorem. $\diamond$

Theorem 1 is useful for verifying whether a given global checkpoint is transaction-consistent. However, this theorem does not help in constructing a transaction-consistent global checkpoint incrementally. This is because, if there is no path between two checkpoints of two different data items, it does not mean that these two checkpoints together can be part of a transaction-consistent global checkpoint. Therefore, additional restrictions need to be added in order to be able to extend a given set of checkpoints to a transaction-consistent global checkpoint. As mentioned earlier, our goal is to come up with the necessary and sufficient conditions for a set of checkpoints to be part of a transaction-consistent global checkpoint.

Next, we introduce some terminology for developing the necessary and sufficient conditions for a set of checkpoints to be part of a transaction-consistent global checkpoint. Netzer and Xu [2] introduced the concept of zigzag paths between checkpoints of a distributed computation and used it to establish the necessary and sufficient conditions for a set of checkpoints of a distributed computation to be

part of a consistent global checkpoint. We generalize their definition of zigzag paths to checkpoints in distributed database systems and use it for establishing the necessary and sufficient conditions for a set of checkpoints of a distributed database to be part of a transaction-consistent global checkpoint.

Definition 2 *Let $C_i^{k_i}$ be a checkpoint taken by the checkpointing transaction $T_{C_i^{k_i}}$, and let $C_j^{k_j}$ be another checkpoint taken by checkpointing transaction $T_{C_j^{k_j}}$. We say a zigzag path exists from $T_{C_i^{k_i}}$ to $T_{C_j^{k_j}}$ if there exists a set of transactions $\mathbf{T'} = \{T_{i_1}, T_{i_2}, \cdots, T_{i_v}\} \subseteq \mathbf{T}$ such that*

a) $T_{i_1} \in \mathbf{T'}$ is a transaction such that $T_{C_i^{k_i}} \longrightarrow T_{i_1}$ in the global serialization graph;

b) for any $T_{i_k} \in \mathbf{T'}(1 \leq k < v)$, $T_{i_{k+1}} \in \mathbf{T'}(1 < (k+1) \leq v)$ is a transaction such that

1: $T_{i_k} \longleftarrow T_{i_{k+1}}$ (we call such an edge as reverse edge);

or

2: $T_{i_k} \longrightarrow T_{i_{k+1}}$ or ($T_{i_k} \longrightarrow T_{C_w^{k_w}}$ and $T_{C_w^{k_w}} \longrightarrow T_{i_{k+1}}$ for some w);

c) $T_{i_v} \in T'$ is a transaction such that $T_{i_v} \longrightarrow T_{C_j^{k_j}}$;

Note that a path in the global serialization graph is also a zigzag path but not conversely. Next, we establish the necessary and sufficient condition formally.

Theorem 2 *A set $\mathbf{S'}$ of checkpoints, each checkpoint of which is from a different data item, can belong to the same consistent global checkpoint with respect to a serializable schedule of a set of transactions iff no checkpoint in $\mathbf{S'}$ has a zigzag path to any checkpoint (including itself) in $\mathbf{S'}$ in the global serialization graph corresponding to that schedule.*

Proof:

(If-Part:) Suppose no checkpoint in $\mathbf{S'}$ has a zigzag path to any checkpoint (including itself) in $\mathbf{S'}$. We construct a consistent global checkpoint $\mathbf{S}$ that contains the checkpoints in $\mathbf{S'}$ plus one checkpoint for each data item not represented in $\mathbf{S'}$ as follows:

- For each data item that has no checkpoint in $\mathbf{S'}$ and that has a checkpoint with a zigzag path to a member of $\mathbf{S'}$, we include in $\mathbf{S}$ its first checkpoint that has no zigzag path to any checkpoint in $\mathbf{S'}$. Such a checkpoint is guaranteed to exist because the virtual checkpoint of a data item does not have an outgoing zigzag path.
- For each data item that has no checkpoint in $\mathbf{S'}$ and that has no checkpoint with zigzag path to a member of $\mathbf{S'}$, we include its initial checkpoint (it is also the first checkpoint that has no zigzag path to any member of $\mathbf{S'}$ and there cannot be a zigzag path from any checkpoint in $\mathbf{S'}$ to this initial checkpoint).

We claim that $\mathbf{S}$ is a tr-consistent global checkpoint. From Theorem 1, it is sufficient to prove that there is no path between any two checkpoints of $\mathbf{S}$ in the global serialization graph. Suppose there is a path from a checkpoint $A \in \mathbf{S}$ to a checkpoint $B \in \mathbf{S}$. Assume that the checkpoint A was taken on data item x_i and checkpoint B was taken on data item on x_j.

Case 1: $A, B \in \mathbf{S'}$. This condition implies that a zigzag path exists from A to B, contradicting the assumption that no zigzag path exists between any two checkpoints in $\mathbf{S'}$.

Case 2: $A \in \mathbf{S} - \mathbf{S'}$ and $B \in \mathbf{S'}$. This contradicts the way $\mathbf{S} - \mathbf{S'}$ is constructed (checkpoints in $\mathbf{S} - \mathbf{S'}$ are chosen in such a way that no zigzag path exists to any member of $\mathbf{S'}$ from those checkpoints).

Case 3: $A \in \mathbf{S'}$ and $B \in \mathbf{S} - \mathbf{S'}$. B cannot be an initial checkpoint, since no checkpoint can have a path to an initial checkpoint. Then by the choice of B, B must be the first checkpoint on x_j that has no zigzag path to any member of $\mathbf{S'}$. The checkpoint preceding B on x_j, say D, must have a zigzag path to some member of $\mathbf{S'}$, say E.

Let T_u be the transaction (that accessed x_j and created the edge $T_u \longrightarrow B$) that lies on the zigzag path from A to B. Similarly, let T_v be a transaction (that accessed x_j and created the edge $D \longrightarrow T_v$) that lies on the zigzag path from D to E. Note that such transactions exist because B and D are checkpoints of data item x_j. In addition, according to our assumption that checkpoint is taken only after the state of the data item has been changed by transaction, let us suppose such transaction is T_w.

Claim: There exists a zigzag path from A to E in the global serialization graph.

Proof of the claim: Based on Observation 1, we have either $T_u \longrightarrow^+ D$ or $D \longrightarrow^+ T_u$ in local serialization graph corresponding to data item x_j. If $T_u \longrightarrow^+ D$, due to the existence of zigzag path from D to E, we get a zigzag path from A to E through T_u, D and T_v. and hence the claim. On the other hand, if $D \longrightarrow^+ T_u$, then because of the edge from T_u to B, we know that T_u must happen between D and B in the local serialization graph. Similarly, we can show that T_v must also happen between D and B in the local serialization graph. Meanwhile since T_w happen between D and B, we conclude that any path between T_u, T_v and T_w must have no checkpoints along the path since D and B are two adjacent checkpoints in the local serialization graph of x_j. Since T_w involves write

operations, which is similar to checkpointing operations, all the observations for checkpointing transactions can be used for T_w. From Observation 1, T_w must have a path to or from both T_u and T_v, therefore four cases arise:

1) If $T_u \longrightarrow^+ T_w \longrightarrow^+ T_v$, the zigzag path from A to T_u, the path $T_u \longrightarrow^+ T_w \longrightarrow^+ T_v$ and the zigzag path from T_v to E construct the zigzag path from A to E.

2) If $T_v \longrightarrow^+ T_w \longrightarrow^+ T_u$, the zigzag path from A to T_u, the reverse path $T_v \longrightarrow^+ T_w \longrightarrow^+ T_u$ and the zigzag path from T_v to E construct the zigzag path from A to E.

3) If $T_w \longrightarrow^+ T_u$ and $T_w \longrightarrow^+ T_v$, the zigzag path from A to T_u, the reverse path from T_u to T_w, the path from T_w to T_v and the zigzag path from T_v to E construct the zigzag path from A to E.

4) If $T_u \longrightarrow^+ T_w$ and $T_v \longrightarrow^+ T_w$, the zigzag path from A to T_u, the path from T_u to T_w, the reverse path from T_w to T_v and the zigzag path from T_v to E construct the zigzag path from A to E.

Since local serialization graph is a component of the global serialization graph, any local path can find its correspondence in the global serialization graph. Therefore the zigzag path from A to E must can be found in the global serialization graph. Therefore no matter what the situation is, we can always find a zigzag path from from A to E, which a contradiction to our assumptions. This is a contradiction to the assumption that no zigzag path exists between any two checkpoints in S'.

Case 4: $A \in \mathbf{S} - \mathbf{S'}$ and $B \in \mathbf{S} - \mathbf{S'}$. As in case 3, B must be the first checkpoint on x_j that has no zigzag path to any member of $\mathbf{S'}$. Then the checkpoint that precedes B on data item x_j, say D, must have a zigzag path to some member of $\mathbf{S'}$, say E. Then, as in case 3, there exists a zigzag path from A to E. This contradicts the choice of A where A is the first checkpoint on data item x_i with no zigzag path to any member of $\mathbf{S'}$.

Therefore $\mathbf{S}$, containing $\mathbf{S'}$, is a tr-consistent global checkpoint.

(Only-if Part:) Conversely, suppose there exists a zigzag path between two checkpoints in $\mathbf{S'}$ (including zigzag cycle), then we show that they cannot belong to the same transaction-consistent global checkpoint. Assume that a zigzag path exists from A to B (A could be B) and along such a path, the length of consecutive reverse edges is at most w. We use induction on w to show that A and B cannot belong to the same consistent global checkpoint.

Base case ($w = 0$): If the length of consecutive reverse edges is at most zero, the zigzag path from A to B is in fact a path from A to B. Then, from Theorem 1, A and B can not belong to the same consistent global checkpoint.

Base case ($w = 1$): Suppose the length of consecutive reverse edges along the zigzag path from A to B is at most one. Let the consecutive reverse edges with length equal to one from A to B be $T_{1,1} \longleftarrow T_{2,1}$, $\cdots$, $T_{1,u} \longleftarrow T_{2,u}$. Suppose those reverse edges are components of local serialization graph corresponding to data items $x_{1,1}, \cdots, x_{1,u}$.

Claim: $x_{1,1}, \cdots, x_{1,u}$ can not all be equal to x_i where A takes place.
Proof of claim: Suppose $x_{1,1}, \cdots, x_{1,u}$ are all equal to x_i. Then A, $T_{1,1}$, $T_{2,1}$, $\cdots$, $T_{1,u}$, $T_{2,u}$ are transactions accessing x_i (note that we use A and the checkpointing transaction that takes the checkpoint A interchangeably). From Observation 1, the following two cases arise:

1) $A \longrightarrow^+ T_{2,u}$. If this is the case, a path $A \longrightarrow^+ B$ via $T_{2,u}$ exists and hence A and B can not be part of a transaction-consistent global checkpoint, by Theorem 1.

2) $T_{2,u} \longrightarrow^+ A$. Since $T_{2,u} \longrightarrow T_{1,u}$, we must have $T_{1,u} \longrightarrow^+ A$ from Observation 3. If this is the case, when we consider the reverse edge $T_{1,u-1} \longleftarrow T_{2,u-1}$, the following two sub-cases arise:

   2.1) $A \longrightarrow^+ T_{2,u-1}$. In this case, a cycle $T_{1,u} \longrightarrow^+ A \longrightarrow^+ T_{2,u-1} \longrightarrow^+ T_{1,u}$ from $T_{1,u}$ to itself exists. However, a cycle can not exist if the schedule of $\mathbf{T} \cup \mathbf{T_C} \in CSR$.

   2.2) $T_{2,u-1} \longrightarrow^+ A$. Since $T_{2,u-1} \longrightarrow T_{1,u-1}$, we must have $T_{1,u-1} \longrightarrow^+ A$ from Observation 3. If this is the case, we need to consider the previous reverse edge $T_{1,u-2} \longleftarrow T_{2,u-2}$ in the zigzag path and make a similar argument with that edge. Proceeding like this, we will end up with a path $T_{1,1} \longrightarrow^+ A$; since $A \longrightarrow^+ T_{1,1}$, we have $A \longrightarrow^+ A$, i.e., A is on a cycle which is a contradiction to the assumption that the schedule of $\mathbf{T} \cup \mathbf{T_C} \in CSR$ is serializable.

So, our assumption that $x_{1,1}, \cdots, x_{1,u}$ are all equal to x_i is wrong and hence the proof of the claim.

Using arguments similar to the one above, we can show that $x_{1,1}, \cdots, x_{1,u}$ can not all be x_j. So far, we have proved that there must exist a data item associated with a reverse edge that is different from both x_i and x_j. Let us assume such a data item is $x_{1,p}$ with associated reverse edge as $T_{1,p} \longleftarrow T_{2,p}$. Next we prove our claim that A and B can not belong to a transaction-consistent global checkpoint.

On data item $x_{1,1}$ that both $T_{1,1}$ and $T_{2,1}$ have accessed, no checkpoint taken after $T_{1,1}$, say D_1, can be combined

with A to form a consistent global checkpoint due to the path $A \longrightarrow^+ D_1$ (from Theorem 1). Therefore, on $x_{1,1}$ we can only use some checkpoint C_1 taken before $T_{2,1}$ accessed $x_{1,1}$ to construct a transaction-consistent global checkpoint containing A. Using similar argument, on $x_{1,2}$, which both $T_{1,2}$ and $T_{2,2}$ have accessed, any checkpoint taken after $T_{1,2}$ accessed, say D_2, can not be combined with C_1 to form a consistent global checkpoint due to the path from $C_1 \longrightarrow^+ D_2$. So we have to use some checkpoint C_2 on $x_{1,2}$, which was taken before $T_{2,2}$ accessed $x_{1,2}$. Similarly, on $x_{1,p}$, which both $T_{1,p}$ and $T_{2,p}$ have accessed, we have to use some checkpoint C_p, which was taken before $T_{2,p}$ to construct a transaction-consistent global checkpoint containing A.

On the other hand, on data item $x_{1,u}$ that both $T_{1,u}$ and $T_{2,u}$ have accessed, no checkpoint taken before $T_{2,u}$, say C_u, can be combined with B to construct a transaction-consistent global checkpoint due to the path path $C_u \longrightarrow^+ B$. Therefore, on $x_{1,u}$, we can only use some checkpoint D_u taken after $T_{1,u}$ accessed $x_{1,u}$ to construct a transaction-consistent consistent global checkpoint containing B. Similarly, on $x_{1,u-1}$, which both $T_{1,u-1}$ and $T_{2,u-1}$ have accessed, any checkpoint taken before $T_{2,u-1}$, say C_{u-1} can not be combined with D_u to construct a transaction-consistent global checkpoint containing due to the path path $C_{u-1} \longrightarrow^+ D_u$. So we have to use some checkpoint D_{u-1} on $x_{1,u-1}$ that was taken after $T_{1,u-1}$ accessed $x_{1,u-1}$. Proceeding like this, on $x_{1,p}$, which both $T_{1,p}$ and $T_{2,p}$ have accessed, we have to use some checkpoint D_p, which is taken after $T_{1,p}$ accessed $x_{1,p}$, to construct a transaction-consistent global checkpoint containing B.

Thus, for data item $x_{1,p}$, we can only use a checkpoint taken before $T_{1,p}$ and $T_{2,p}$ have accessed $x_{1,p}$ to construct a transaction-consistent global checkpoint containing A; on the other hand, we can only use a checkpoint taken after $T_{1,p}$ and $T_{2,p}$ have accessed to construct a transaction-consistent global checkpoint containing B. So, for data item $x_{1,p}$, there in no checkpoint that can be combined with both A and B to construct a transaction-consistent global checkpoint. This proves the Theorem in the base case $w = 1$.

Next, assume that if there is a zigzag path from A to B which contains consecutive reverse edges with length at most k, then A and B together cannot belong to a transaction-consistent global checkpoint. We prove that a zigzag path from A to B which contains consecutive reverse edges of lengths at most $k+1$ can not belong to a transaction-consistent global checkpoint.

Suppose the series of consecutive reverse edges along the zigzag path from A to B are: $T_{1,1} \longleftarrow \cdots \longleftarrow T_{u_1,1}$ $(u_1 \leq k+1)$; $T_{1,2} \longleftarrow \cdots \longleftarrow T_{u_2,2}$ $(u_2 \leq k+1)$; $\cdots$ $T_{1,v} \longleftarrow \cdots \longleftarrow T_{u_v,v}$ $(u_v \leq k+1)$. Thus, on the zigzag path from A to B that we consider, we have consecutive reverse edges of lengths $u_1, \cdots, u_v$, $(u_i \leq k+1 \forall i)$. Each of these reverse edges should come from the local serialization graph of a data item. Suppose the reverse edges are edges of local serialization graphs corresponding to data items $x_{1,1}, \cdots, x_{u_1-1,1}, \ldots, \ldots, x_{1,v}, \cdots, x_{u_v-1,v}$ respectively. First, we show that at least one of the data items $x_{1,1}, \cdots, x_{u_1-1,1} \ldots, \ldots x_{1,v}, \cdots, x_{u_v-1,v}$ is not equal to x_i (recall that A is the checkpoint of data item x_i).

Suppose $x_{1,1}, \cdots, x_{u_1-1,1} \ldots, \ldots, x_{1,v}, \cdots, x_{u_v-1,v}$ are all same as x_i. Then $A, T_{1,1}, \cdots, T_{u_1,1} \ldots, \ldots T_{1,v}, \cdots, T_{u_v,v}$ are transactions accessing x_i. Based on Observation 1, two cases arise:

1) $A \longrightarrow^+ T_{u_v,v}$. If this is the case, a path $A \longrightarrow B$ via $T_{u_v,v}$ exists, and hence A and B together cannot be part of a transaction-consistent global checkpoint by Theorem 1.

2) $T_{u_v,v} \longrightarrow^+ A$. Because of the series of reverse edges $T_{1,v} \longleftarrow \cdots \longleftarrow T_{u_v,v}$ on x_i, from Observation 3, we have $T_{1,v} \longrightarrow^+ A$. Then, when we consider the series of reverse edges $T_{1,v-1} \longleftarrow \cdots \longleftarrow T_{u_{v-1},v-1}$, the following two sub-cases arise:

   2.1) $A \longrightarrow^+ T_{u_{v-1},v-1}$. In this case, a cycle exists $T_{1,v} \longrightarrow^+ A \longrightarrow^+ T_{u_{v-1},v-1} \longrightarrow^+ T_{1,v}$, which is a contradiction to the fact that the schedule of $\mathbf{T} \cup \mathbf{T_C} \in CSR$.

   2.2) $T_{u_{v-1},v-1} \longrightarrow^+ A$. Because of the series of reverse edges $T_{1,v-1} \longleftarrow \cdots \longleftarrow T_{u_{v-1},v-1}$ on x_i, based on Observation 3, we have $T_{1,v-1} \longrightarrow^+ A$. In this case, we need to consider the previous sequence of reverse edges $T_{1,v-2} \longleftarrow \cdots \longleftarrow T_{u_{v-2},v-2}$ and repeat the analysis similar to case 2.1) and 2.2).

Continuing this process, we will end up with a cycle in the serialization graph which is a contradiction to the fact that $\mathbf{T} \cup \mathbf{T_C} \in CSR$. This means that our assumption that $x_{1,1}, \cdots, x_{u_1-1,1} \ldots, \ldots x_{1,v}, \cdots, x_{u_v-1,v}$ are all x_i is wrong.

Using similar arguments, we can show that not all the data items $x_{1,1}, \cdots, x_{u_1-1,1} \ldots, \ldots x_{1,v}, \cdots, x_{u_v-1,v}$ can be equal to x_j. Suppose $x_{1,1}, \cdots, x_{u_1-1,1} \ldots, \ldots x_{1,v}, \cdots, x_{u_v-1,v}$ are all x_j.

So far we have proved that there must exist a data item associated with at least one reverse edge in the zigzag path from A to B that is different from both x_i and x_j. suppose such a data item is $x_{g,p}$ associated with the reverse edge $T_{g,p} \longleftarrow T_{g+1,p}$ which is one of the reverse edges in the sequence of reverse edges $T_{1,p} \longleftarrow \cdots T_{u_p,p}$. Next, we prove that A and B can not be part of a transaction-consistent global checkpoint.

On data item $x_{1,1}$ that both $T_{1,1}$ and $T_{2,1}$ have accessed, no checkpoint D_1, taken after $T_{1,1}$ has accessed $x_{1,1}$, can be combined with A to construct a consistent global checkpoint because there is a path from A to D_1. Therefore we can only use some checkpoint C_1, taken before $T_{2,1}$ on $x_{1,1}$ to construct a consistent global checkpoint containing A. On $x_{1,2}$, which both $T_{1,2}$ and $T_{2,2}$ have accessed, no checkpoint taken after T_{1_2}, say D_2, can be combined with C_1 to form a consistent global checkpoint because there is a zigzag path from C_1 to D_2 with consecutive reverse edges of length at most k. So we have to use some checkpoint C_2 on $x_{1,2}$, which was taken before $T_{2,2}$ accessed $x_{1,2}$.

Proceeding like this, on data item $x_{g,p}$, which was accessed by the transactions $T_{g,p}$ and $T_{g+1,p}$, no checkpoint D_p taken after both $T_{g,p}$ and $T_{g+1,p}$, have accessed can be combined with C_{p-1}, to construct a consistent global checkpoint due to the existence of the zigzag path of consecutive reverse edges of length at most k. So we have to use some checkpoint C_p which was taken before $T_{g,p}$ and $T_{g+1,p}$ have accessed $x_{g,p}$.

On the other hand, on $x_{1,v}$, which both $T_{1,v}$ and $T_{2,v}$ have accessed, no checkpoint C_v that was taken before $T_{2,v}$ accessed $x_{1,v}$ combined with B to construct a tr-consistent global checkpoint because C_v has a zigzag path to B with consecutive reverse edges of length at most k. Therefore, on $x_{1,v}$, we have to use some checkpoint D_v, that was taken after $T_{1,v}$ accessed $x_{1,v}$. On $x_{1,v-1}$, which both $T_{1,v-1}$ and $T_{2,v-1}$ have accessed, we can not use any checkpoint C_{v-1} that was taken before $T_{2,v-1}$ to construct a consistent global checkpoint containing D_v due to the existence of a zigzag path with consecutive reverse edges of length at most k So, we have to use some checkpoint D_{v-1} on $x_{1,v-1}$ that was taken after $T_{1,v-1}$ accessed. Proceeding like this, on $x_{g,p}$, we have to use some checkpoint D_p that was taken after $T_{g,p}$ has accessed to construct a tr-consistent global checkpoint containing D_{p-1}.

Thus, on the data item $x_{g,p}$, which is different from both x_i and x_j, no checkpoint that was taken before $T_{g,p}$ accessed can be used to construct a tr-consistent global checkpoint containing A and no checkpoint taken after $T_{g,p}$ accessed can be used to construct a tr-consistent global checkpoint containing B. Thus, $x_{g,p}$ does not have any checkpoint that can be combined with both A and B to construct a tr-consistent global checkpoint.

Therefore A and B can not belong to a consistent global checkpoint. This proves the theorem. ⋄

Corollary 1 *A checkpoint of a data item in a distributed database can be part of a tr-consistent global checkpoint of the database iff it does not lie on a zigzag cycle.*

Proof: Follows from the Theorem by taking $\mathbf{S}^{'}$ as the set containing the checkpoint. ⋄

5 Conclusion

Checkpointing has been traditionally used for handling failures in distributed database systems. If each data item is independently checkpointed, the checkpoints taken may not be useful for constructing a transaction-consistent global checkpoint of the entire database. In this paper, We have presented the necessary and sufficient condition for a set of checkpoints of a set of data items in the database to be part of a transaction-consistent global checkpoint of the distributed database.

References

[1] S. Pilarski and T. Kameda, "Checkpointing for distributed databases: starting from the basics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, pp. 602–610, 1992.

[2] R. H. B. Netzer and J. Xu, "Necessary and sufficient condition for consistent global snapshots," *IEEE Trans. Soft. Eng.*, vol. 6, pp. 274–281, 1999.

[3] S. H. Son, "An algorithm for non-interfering checkpoints and its practicality in distributed database systems," *Information Systems*, vol. 14, no. 5, pp. 421–429, 1989.

[4] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database system concepts*, 1996.

[5] M. Singhal and N. G. Shivaratri, *Advanced concepts in operating systems*, 1994.

[6] S. H. Son and A. K. Agrawala, "Distributed checkpointing for globally consistent states of databases," *IEEE Transactions on Software Engineering*, vol. 15, no. 10, pp. 1157–1167, October 1989.

[7] C. Pu, "On-the-fly, incremental, consistent reading of entire databases," in *Proceedings of the 11th Conference on Very Large Database*. Morgan Kaufman Pubs. (Los Altos, CA), Stockholm, 1985, pp. 367–375.

[8] S. Pilarski and T. Kameda, "A novel checkpointing scheme for distributed database systems," in *Proc. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Sys.*, Nashville, TN, 1990.

[9] R. Baldoni, F. Quaglia, and M. Raynal, "Consistent checkpointing for transaction systems," *The Computer Journal*, vol. 44, no. 2, 2001.