

What is claimed is:

1. A method for processing transactions in a distributed computing system including multiple processing modules, the method including:

storing data items in a data storage system accessible to multiple processes running in the distributed computing system, where the data items are totally ordered according to an ordering rule, and at least some of the processes are running on different processing modules; and

processing transactions using a plurality of the multiple processes, where processing a transaction using one of the plurality of the multiple processes includes

receiving a set of requests for accessing data items stored in the data storage system, where the requests are in a first order,

obtaining locks on the data items sequentially in the first order if each of the locks is obtained within a first time interval,

determining a second order that is consistent with the ordering rule for at least two of the locks if any of the locks is not obtained within the first time interval, and

restarting the transaction being processed, including obtaining locks on data items sequentially in the second order.

2. The method of claim 1, wherein restarting the transaction being processed includes rolling back the transaction, releasing any existing locks on the data items.

3. The method of any one of claims 1 to 2, wherein the second order is consistent with the ordering rule for at least data items for which locks were obtained within the first time interval.

4. The method of any one of claims 1 to 3, wherein the processing further includes storing information identifying positions based on the ordering rule of data items for which locks were obtained within the first time interval, and determining the second order includes determining the second order based at least in part on the stored information identifying the positions.

5. The method of any one of claims 1 to 4, wherein determining the second order includes sorting a list of operations for obtaining the locks according to the ordering rule.

6. The method of any one of claims 1 to 5, wherein the processing further includes determining whether or not the first order is consistent with the ordering rule, and waiting longer than the first time interval for locks on any data items for which a lock was not obtained within the first time interval if the first order is consistent with the ordering rule.

7. The method of any one of claims 1 to 6, wherein the processing further includes determining a priority of the process that received the set of requests relative to other processes of the plurality of the multiple processes.

8. The method of claim 7, wherein restarting the transaction being processed is performed after determining the priority.

9. The method of any one of claims 1 to 8, wherein restarting the transaction being processed is performed after determining that a state of a lock on at least one data item for which access was requested by the set of requests has changed.

10. The method of any one of claims 1 to 5, wherein the processing further includes determining whether or not the first order is consistent with the ordering rule, and restarting the transaction being processed is performed after a second time interval from determining that first order is not consistent with the ordering rule.

11. A computer-readable storage medium, storing instructions, which when executed by one or more computers, cause one or more computers to perform operations for processing transactions in a distributed computing system including multiple processing modules, the operations including:

storing data items in a data storage system accessible to multiple processes running in the distributed computing system, where the data items are totally ordered according to an ordering rule, and at least some of the processes are running on different processing modules; and

processing transactions using a plurality of the multiple processes, where processing a transaction using one of the plurality of the multiple processes includes

receiving a set of requests for accessing data items stored in the data storage system, where the requests are in a first order,

obtaining locks on the data items sequentially in the first order if each of the locks is obtained within a first time interval,

determining a second order that is consistent with the ordering rule for at least two of the locks if any of the locks is not obtained within the first time interval, and

restarting the transaction being processed, including obtaining locks on data items sequentially in the second order.

12. A distributed computing system for processing transactions, the distributed computing system including:

a plurality of processing modules; and

a data storage system configured to store data items, the data storage system accessible to multiple processes running in the distributed computing system, where the data items are totally ordered according to an ordering rule, and at least some of the processes are running on different processing modules;

wherein a plurality of the multiple processes are configured to process transactions, and processing a transaction using one of the plurality of the multiple processes includes

- receiving a set of requests for accessing data items stored in the data storage system, where the requests are in a first order,
- obtaining locks on the data items sequentially in the first order if each of the locks is obtained within a first time interval,
- determining a second order that is consistent with the ordering rule for at least two of the locks if any of the locks is not obtained within the first time interval, and
- restarting the transaction being processed, including obtaining locks on data items sequentially in the second order.

13. A method for processing transactions in a distributed computing system including multiple processing modules, the method including:

storing data items in a data storage system accessible to multiple processes running in the distributed computing system, where the data items are totally ordered according to an ordering rule, and at least some of the processes are running on different processing modules; and

processing transactions using a plurality of the multiple processes, where processing a transaction using one of the plurality of the multiple processes includes

receiving a set of requests for accessing data items stored in the data storage system, where the requests are in a first order,

obtaining locks on the data items sequentially in the first order if each of the locks is obtained within a first time interval,

determining whether or not the first order is consistent with the ordering rule if any of the locks is not obtained within the first time interval, and

restarting the transaction being processed if the first order is not consistent with the ordering rule.

14. The method of claim 13, wherein restarting the transaction being processed includes rolling back the transaction, releasing any existing locks on the data items, and obtaining locks on data items sequentially in a second order different from the first order.

15. The method of claim 14, wherein the second order is consistent with the ordering rule for at least the data items for which locks were obtained within the first time interval.

16. The method of any one of claims 14 to 15, wherein the processing further includes storing information identifying positions based on the ordering rule of data items for which locks were obtained within the first time interval, and determining the second order based at least in part on the stored information identifying the positions.

17. The method of any one of claims 14 to 16, wherein the second order is determined based on sorting a list of operations for obtaining the locks according to the ordering rule.

18. The method of any one of claims 13 to 17, wherein the processing further includes waiting longer than the first time interval for locks on any data items for which a lock was not obtained within the first time interval if the first order is consistent with the ordering rule.

19. The method of any one of claims 13 to 18, wherein the processing further includes determining a priority of the process that received the set of requests relative to other processes of the plurality of the multiple processes.

20. The method of claim 19, wherein restarting the transaction being processed if the first order is not consistent with the ordering rule is performed after determining the priority.

21. The method of any one of claims 13 to 20, wherein restarting the transaction being processed if the first order is not consistent with the ordering rule is performed after determining that a state of a lock on at least one data item for which access was requested by the set of requests has changed.

22. The method of any one of claims 13 to 20, wherein restarting the transaction being processed if the first order is not consistent with the ordering rule is performed after a second time interval from determining that first order is not consistent with the ordering rule.

23. The method of claim 10 or claim 22, wherein the second time interval is longer than the first time interval.

24. The method of any one of claims 1 to 10, 13 to 23, wherein the first time interval is less than one second.

25. The method of any one of claims 1 to 10, 13 to 24, wherein the processing further includes: holding any locks obtained on the data items until the transaction being processed is committed, aborted, or restarted; and releasing the locks on the data items when the transaction being processed is committed, aborted, or restarted.

26. The method of any one of claims 1 to 10, 13 to 25, wherein the locks on the data items include at least two states including a single unlocked state and one or more locked states.

27. The method of claim 26, wherein obtaining a lock on a data item includes changing the state of a lock associated with the data item into one of the locked states.

28. The method of any one of claims 26 to 27, wherein releasing a lock on a data item includes changing the state of a lock associated with the data item from one of the locked states into the unlocked state.

29. The method of any one of claims 26 to 28, wherein the locks on the data items include at least one exclusive locked state that permits only a single process full access to a locked data item.

30. The method of any one of claims 26 to 29, wherein the locks on the data items include at least one shared locked state that permits multiple processes read-only access to a locked data item.

31. The method of any one of claims 1 to 10, 13 to 30, wherein at least some of the multiple processes are running asynchronously from each other.

32. The method of any one of claims 1 to 10, 13 to 31, wherein the transactions include database transactions and the data items are records in a database.

33. The method of claim 32, wherein the database is an in-memory database.

34. The method of any one of claims 32 to 33, wherein the data storage system is distributed among multiple nodes of the database.

35. The method of any one of claims 1 to 10, 13 to 34, wherein at least some of the plurality of the multiple processes are running on different ones of the processing modules.

36. A computer-readable storage medium, storing instructions, which when executed by one or more computers, cause one or more computers to perform operations for processing transactions in a distributed computing system including multiple processing modules, the operations including:

storing data items in a data storage system accessible to multiple processes running in the distributed computing system, where the data items are totally ordered according to an ordering rule, and at least some of the processes are running on different processing modules; and

processing transactions using a plurality of the multiple processes, where processing a transaction using one of the plurality of the multiple processes includes

receiving a set of requests for accessing data items stored in the data storage system, where the requests are in a first order,

obtaining locks on the data items sequentially in the first order if each of the locks is obtained within a first time interval,

determining whether or not the first order is consistent with the ordering rule if any of the locks is not obtained within the first time interval, and

restarting the transaction being processed if the first order is not consistent with the ordering rule.

37. A distributed computing system for processing transactions, the distributed computing system including:

a plurality of processing modules; and

a data storage system configured to store data items, the data storage system accessible to multiple processes running in the distributed computing system, where the data items are totally ordered according to an ordering rule, and at least some of the processes are running on different processing modules;

wherein a plurality of the multiple processes are configured to process transactions, and processing a transaction using one of the plurality of the multiple processes includes

receiving a set of requests for accessing data items stored in the data storage system, where the requests are in a first order,

obtaining locks on the data items sequentially in the first order if each of the locks is obtained within a first time interval,

determining whether or not the first order is consistent with the ordering rule if any of the locks is not obtained within the first time interval, and

restarting the transaction being processed if the first order is not consistent with the ordering rule.