

WHAT IS CLAIMED IS:

1. A system, comprising:

one or more computing devices configured to:

- 5 in response to a request to register a plurality of data stores for transaction management via a multi-database logging service implemented at a provider network, including a first data store implementing a first set of data access interfaces and a second data store implementing a second set of data access interfaces, designate a particular persistent change log of the logging service to store representations of writes directed to one or
- 10 more data stores of the plurality of data stores;
- receive, at a conflict detector of the multi-database logging service, a representation of a transaction requested by a client, wherein the transaction includes at least a first write operation directed to the first data
- 15 store, and at least a second write operation directed to the second data store;
- determine, at the conflict detector, based at least in part on contents of the representation and at least in part on contents of the particular persistent change log, that the transaction is to be committed;
- 20 add, to the particular persistent change log of the multi-database logging service, one or more log records indicating the transaction;
- apply the first write operation to the first data store and the second write operation to the second data store;
- provide, in response to a first query directed to the first data store using the first
- 25 set of data access interfaces, a first response including a result of the first write operation;
- provide, in response to a second query directed to the second data store using the second set of data access interfaces, a second response including a result of the second write operation; and
- 30 provide, in response to a third query directed to the logging service via a programmatic query interface defined by the multi-database logging service, a third response including a result of one or more of: the first write operation and the second write operation.

2. The system as recited in claim 1, wherein each data store of the first data store and the second data store comprises one of: a non-relational database system, a relational database system, a storage service that implements a web services interface allowing access to unstructured data objects, an in-memory database, or an instance of a distributed cache.

5

3. The system as recited in claim 1, wherein the particular persistent change log comprises a plurality of replica nodes, wherein each node of the plurality of replica nodes stores a replica of a log record of the one or more log records at a respective storage device.

10 4. The system as recited in claim 1, wherein the one or more computing devices are further configured to:

update a materialized view in accordance with the result of at least the first write operation.

15 5. The system as recited in claim 4, wherein the one or more computing devices are further configured to:

generate, based at least in part on the result of at least the first write operation, a pre-computed result for a fourth query; and
in response to receiving the fourth query, provide the pre-computed result.

20

6. A method, comprising:

performing, by one or more computing devices:

instantiating a particular persistent change log to store representations of writes directed to one or more data stores including a first data store

25

implementing a first set of data access interfaces;

receiving a representation of a transaction requested by a client of the first data store, wherein the transaction includes at least a first write directed to a data object stored at the first data store;

30

determining, based at least in part on contents of the representation and at least in part on contents of the particular persistent change log, that the requested transaction is to be committed;

adding, to the particular persistent change log, a log record indicating the first write; and

propagating an indication of the first write from the particular persistent change log to the first data store.

7. The method as recited in claim 6, wherein the first data store does not support atomicity for a set of operations that includes more than one write, wherein the transaction requested by the client includes a second write directed to a second data object, further comprising performing, by the one or more computing devices:

subsequent to determining that the requested transaction is to be committed,
adding, to the particular persistent change log, an indication of the second write;
and
propagating an indication of the second write to the first data store.

8. The method as recited in claim 6, wherein the particular persistent change log comprises a plurality of replica nodes, wherein each node of the plurality of replica nodes stores a replica of the log record at a respective storage device.

9. The method as recited in claim 8, wherein each replica node of the plurality of replica nodes is a member of a directed acyclic graph (DAG) with an associated DAG configuration manager responsible for propagating DAG membership change messages to the plurality of replica nodes.

10. The method as recited in claim 6, further comprising performing, by the one or more computing devices:

updating a materialized view in accordance with the result of the first write.

11. The method as recited in claim 6, further comprising performing, by the one or more computing devices:

generating, based at least in part on the result of the first write, a pre-computed result for a query; and

in response to receiving the query, providing the pre-computed result.

12. The method as recited in claim 6, further comprising performing, by the one or more computing devices:

receiving a request to register the one or more data stores for transaction management by

a logging service, wherein said instantiating the particular persistent change log is responsive to said request to register the one or more data stores.

13. The method as recited in claim 6, further comprising performing, by the one or more computing devices:

providing, in response to a first query directed to the first data store using the first set of data access interfaces, a response including a result of the first write; and providing, in response to a second query directed to the particular persistent change log via a programmatic query interface of a logging service, a second response including a result of the first write.

14. The method as recited in claim 6, wherein the first data store comprises control-plane configuration information of a network-accessible service implemented at a provider network, and wherein the transaction comprises a change to a configuration of a particular resource of the network-accessible service.

15. The method as recited in claim 6, further comprising performing, by the one or more computing devices:

storing, by a snapshot manager associated with the particular persistent change log, a point-in-time snapshot of contents of at least a portion of the particular persistent change log.

16. The method as recited in claim 8, wherein the plurality of replica nodes includes a first replica node at a first data center of a provider network and a second replica node at a second data center of the provider network.

17. A non-transitory computer-accessible storage medium storing program instructions that when executed on one or more processors:

receive, at a logging service, a representation of a transaction requested by a client, wherein the transaction includes at least a first write directed to a data object stored at a first data store;

determine, based at least in part on contents of the representation at least in part on contents of a particular persistent change log, that the requested transaction is to be committed;

initiate an insertion, into the particular persistent change log, of a log record indicating the first write; and
propagate the first write to the first data store.

5 18. The non-transitory computer-accessible storage medium storing program instructions as recited in claim 17, wherein the transaction requested by the client includes a second write directed to a second data object, further comprising performing, by the one or more computing devices:

subsequent to determining that the requested transaction is to be committed,
10 initiate an insertion, to the particular persistent change log, of an indication of the second write; and
propagate the second write to a destination data store.

19. The non-transitory computer-accessible storage medium storing program instructions as
15 recited in claim 18, wherein the destination data store is the first data store, and wherein the first data store does not support atomicity for a set of operations that includes more than one write operation.

20. The non-transitory computer-accessible storage medium storing program instructions as
20 recited in claim 18, wherein the destination data store is a second data store with a different access interface than the first data store.

21. The non-transitory computer-accessible storage medium storing program instructions as
recited in claim 17, wherein the particular persistent change log comprises a plurality of replica
25 nodes, wherein each replica node of the plurality of replica nodes stores a replica of the log record at a respective storage device.

22. The non-transitory computer-accessible storage medium storing program instructions as
recited in claim 21, wherein each replica node of the plurality of replica nodes is a member of a
30 directed acyclic graph (DAG) with an associated DAG configuration manager responsible for propagating DAG membership change messages to the plurality of replica nodes.

23. A system, comprising:
one or more computing devices to:

receive, at a log-based transaction manager instantiated for one or more data stores, a transaction descriptor of a requested transaction from a client, wherein the transaction descriptor indicates, corresponding to a first client-specified logical constraint to be checked to determine whether the requested transaction is to be committed, (a) a constraint type, and (b) one or more data signatures;

determine, by the log-based transaction manager using at least some transaction records corresponding to committed transactions stored in a particular persistent log, that the requested transaction is committable with respect to a set of concurrency control requirements distinct from the first client-specified logical constraint;

identify, by the log-based transaction manager, a subset of transaction records stored in the particular persistent log to be examined for checking the first client-specified logical constraint,

compare, by the log-based transaction manager, contents of the one or more data signatures with other data signatures stored in the subset of transaction records;

determine, by the log-based transaction manager based at least in part on a result of one or more comparisons, that the first client-specified logical constraint is satisfied by the requested transaction; and

commit, by the log-based transaction manager, the requested transaction in response to determining that the requested transaction is committable with respect to the set of concurrency requirements and the first client-specified logical constraint is satisfied by the requested transaction.

24. A method, comprising:

performing, by one or more computing devices:

receiving, at a log-based transaction manager, a transaction descriptor indicating a first client-specified logical constraint on a requested transaction, the log-based transaction manager configured to enforce one or more commit requirements for transactions distinct from the first client-specified logical constraint;

identifying a subset of transaction records stored in a particular persistent log to be examined for checking the first client-specified logical constraint,

comparing contents of one or more data signatures indicated in the transaction descriptor with one or more data signatures stored in the subset of transaction records;

determining, based at least in part on a result of said comparing, that the first client-specified logical constraint is met by the requested transaction; and in response to determining that one or more commit requirements of the requested transaction have been met, and that the first client-specified logical constraint is met, committing the requested transaction.

25. A non-transitory computer-accessible storage medium storing program instructions that when executed on one or more processors:

receive, at a log-based transaction manager, a transaction descriptor indicating a first client-specified logical constraint on a requested transaction, the log-based transaction manager configured to enforce one or more commit requirements for transactions distinct from the first client-specified logical constraint;

identify a subset of transaction records stored in a particular persistent log to be examined for checking the first client-specified logical constraint,

compare contents of one or more data signatures indicated in the transaction descriptor with one or more data signatures stored in the subset of transaction records;

determine, based at least in part on a result of comparing the contents of the one or more data signatures, that the first client-specified logical constraint is met by the requested transaction; and

in response to a determination that one or more commit requirements of the requested transaction have been met, and that the first client-specified logical constraint is met, commit the requested transaction.

26. A non-transitory computer-accessible storage medium storing program instructions that when executed on one or more processors:

receive, at a log-based transaction manager, a transaction descriptor of a requested transaction from a client, wherein the transaction descriptor comprises at least (a) respective read signatures indicative of one or more objects from which data is read during the requested transaction, including a first object at a first data store and a second object at a second data store, (b) a first committed state identifier (CSI) determined based at least in part on a first logical timestamp corresponding

to a first previously-committed transaction at the first data store and a second logical timestamp corresponding to a second previously-committed transaction at the second data store, and (c) respective write signatures indicative of one or more objects to which data is to be written in response to a commit of the requested transaction;

identify, by the log-based transaction manager using the first CSI, a subset of transaction records stored in a particular persistent log to be examined for conflicts prior to committing the requested transaction, wherein the particular persistent log comprises transaction records corresponding to previously-committed transactions associated with one or more data stores including the first and second source data stores,

in response to a determination, by the log-based transaction manager, that none of the objects indicated by the read signatures of the requested transaction have been modified by a write operation indicated in the subset of transaction records, accept the requested transaction for commit.

27. A method, comprising:

performing, by one or more computing devices:

receiving a transaction descriptor of a requested transaction, wherein the transaction descriptor comprises at least (a) a read set descriptor indicative of one or more objects from which data is read during the requested transaction including a first object in a first data store and a second object in a second data store, (b) a conflict check delimiter determined based at least in part on a first logical timestamp corresponding to a first previously-committed transaction at the first data store and a second logical timestamp corresponding to a second previously-committed transaction at the second data store, and (c) a write set descriptor indicative of one or more objects to which data is to be written in response to a commit of the requested transaction;

identifying, using the conflict check delimiter, a subset of transaction records stored in a particular persistent log to be examined for conflicts prior to committing the requested transaction,

in response to determining that none of the objects indicated in the read set descriptor were modified by a write operation indicated in the subset of transaction records, accepting the requested transaction for commit.

5 28. A system, comprising:

one or more computing devices configured to:

receive a transaction descriptor of a requested transaction from a client, wherein the transaction descriptor comprises at least (a) a read set descriptor indicative of one or more read locations from which data is read during the requested transaction, including a first location in a first data store and a second location in a second data store, and (b) a conflict check delimiter determined based at least in part on a first logical timestamp corresponding to a first previously-committed transaction at the first data store and a second logical timestamp corresponding to a second previously-committed transaction at the second data store;

identify, using the conflict check delimiter, a subset of transaction records stored in a particular persistent log to be examined for conflicts prior to committing the requested transaction; and

in response to determining that none of the read locations of the requested transaction correspond to a write location indicated in the subset of transaction records, store a new transaction record in the particular persistent log indicating that the requested transaction has been committed.

25 29. A system, comprising:

one or more computing devices configured to:

instantiate a state replication group (SRG) comprising a plurality of nodes to replicate state information of a particular application, wherein at least some nodes of the SRG store a respective commit record set of the application, wherein each commit record of a commit record set comprises a commit sequence number (CSN) indicative of an order in which a state transition of the application was committed relative to other state transitions, and wherein the SRG comprises a committer node configured to commit requested state transitions;

determine, at the committer node, in response to a detection that a threshold condition has been met, that state transition processing operations of the SRG are to be suspended;

transmit, from the committer node to a fault-tolerant configuration manager of the SRG, a suspend request indicating a highest commit sequence number (HCSN) among the CSNs of the commit record set stored at the committer node;

transmit a respective suspend command from the configuration manager to one or more other nodes of the SRG including a second node, wherein the suspend command indicates the HCSN;

pause, in response to receiving a suspend command from the configuration manager, state transition processing operations at the second node;

verify, by the second node, that the second node's commit record set includes a commit record with the HCSN; and

defer, by the committer node and the second node, further processing of state transition operations until a reactivation message is received from the configuration manager.

30. A method, comprising:

performing, by one or more computing devices:

determining that state transition processing operations of a state replication group (SRG) comprising a plurality of nodes are to be suspended, wherein the SRG is designated to replicate state information comprising a respective commit record set of an application, wherein each commit record of the commit record set has an associated commit sequence number (CSN) indicative of an order in which the corresponding state transition of the application was committed at the SRG;

identifying a target CSN up to which commit record sets of one or more nodes of the SRG are to be synchronized;

transmitting a respective suspend command from a configuration manager of the SRG to the one or more other nodes of the SRG, wherein the suspend command indicates the target CSN;

verifying, by a particular node of the one or more other nodes, that a commit record corresponding to the target CSN is stored in the particular node's commit record set; and
suspending state transition processing operations by the particular node.

5

31. A non-transitory computer-accessible storage medium storing program instructions that when executed on one or more processors:

determine a target commit sequence number (CSN) to be used to synchronize state information pertaining to a particular application among a plurality of nodes of a state replication group (SRG) prior to a suspension of application state transition processing operations at the SRG, wherein the plurality of nodes includes a first node and a second node, wherein each node of the first and second nodes stores a respective commit record set of the particular application, and wherein each commit record of the set has an associated respective CSN indicative of an order in which the corresponding state transition was committed at the SRG;
store, by a configuration manager of the SRG at a persistent storage device, the target CSN;
transmit, from the configuration manager to at least one node of the first node and the second node, a respective suspend command indicating the target CSN; and
in response to an indication received at the configuration manager that, subsequent to a respective suspension of operations at the first node and the second node, the first node and the second node are available for resumption of operations,
verify that a number of available nodes of the SRG whose commit record sets include a commit record corresponding to the target CSN exceeds a threshold; and
transmit a re-activation message to at least a subset of available nodes whose commit record sets include a commit record corresponding to the target CSN.

10

15

20

25

32. A system, comprising:

one or more computing devices configured to:

30

identify, at a first non-committer node of a directed acyclic graph (DAG) of nodes designated to replicate a state of an application, at least a portion of a first replication pathway to a first committer node of the DAG, based at least in part on a first sequence of configuration-delta messages received from a fault-tolerant DAG configuration manager, wherein at least one other node

of the DAG has received a different sequence of configuration-delta messages from the fault-tolerant DAG configuration manager; transmit, from the first non-committer node, for replication at respective local storage devices of one or more nodes along the first replication pathway, a state transition message indicating (a) approval of a first candidate state transition requested by a client and (b) a sequence number indicative of an order in which the first candidate state transition is processed at the first non-committer node relative to other candidate state transitions; in response to a determination, by the fault-tolerant DAG configuration manager, that a particular node of the first replication pathway is to be removed from the DAG, initiate a transmission, from the fault-tolerant DAG configuration manager to one or more remaining nodes of the DAG, of a respective configuration-delta message indicating a removal of the particular node from the DAG, without directing the one or more remaining nodes to pause processing of state transitions.

33. A method, comprising:

performing, by one or more computing devices:

propagating a state transition message corresponding to a requested state transition from a first non-committer node of a directed acyclic graph (DAG) of replication nodes to a first committer node of the DAG, wherein the state transition message includes a sequence number indicative of an order in which the requested state transition was approved relative to other requested state transitions; in response to determining, by a DAG configuration manager, that a particular node is to be added to the DAG, initiating a propagation, from the DAG configuration manager to one or more other nodes of the DAG, of a configuration-delta message indicating the addition of the particular node to the DAG, without directing the one or more other nodes to pause processing of state transition messages.

34. A non-transitory computer-accessible storage medium storing program instructions that when executed on one or more processors implement a particular node of a DAG established to replicate state information, wherein the particular node is configured to:

receive, from a DAG configuration manager, a plurality of configuration-delta messages,
5 wherein a particular configuration-delta message of the plurality of configuration-delta messages indicates one or more of: (a) an addition of a node to the DAG, or (b) a removal of a node from the DAG;

subsequent to receiving the particular configuration-delta message from the DAG configuration manager, indicating that an immediate predecessor node from
10 which a state transition message was received at the particular node has been removed from the DAG,

identify, using at least the particular configuration-delta message, a different immediate predecessor node, and

transmit, to the different immediate predecessor node, a request for state transition
15 messages.

35. A non-transitory computer-accessible storage medium storing program instructions that when executed on one or more processors implement a configuration manager of a DAG of nodes established to replicate state information, wherein the configuration manager is configured to:

20 determine, based at least in part on a consensus protocol, that a particular node of the DAG is to be removed from the DAG, wherein the particular node occupies a particular position within the DAG;

remove, from a persistent representation of a configuration of the DAG stored at the configuration manager, the particular node; and

25 transmit, to one or more remaining nodes of the DAG, a configuration-delta message indicating that the particular node has been removed from the DAG, without requesting the one or more remaining nodes to pause processing related to state transitions of the application.