Maintaining Data Consistency in Mobile Distributed Environments

Evaggelia Pitoura and Bharat Bhargava

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
{pitoura, bb}@cs.purdue.edu
http://www.cs.purdue.edu/people/pitoura

ICDCS Presentation
June 1995

Introduction

- Cordless technology
  Portable computers
- Access to databases =>
  replication for
  - reliability
  - performance
  - cost
- Schemas for replica placement and coherency control
- Topic: a coherency schema appropriate for mobile environments

Introduction

- Motivation

  Wireless connections are:
  - slow
  - expensive
  - not widely available

  Disconnections:
  - frequent
  - predictable

- Effect on coherency control

  1. Support for autonomous operation
  2. Greater concern for bandwidth
  3. Adaptability to varying bandwidth
  4. Take into account changing locality

Introduction

- Outline of our approach

  -- Related data are grouped in clusters

  -- Cluster-wise consistency

  -- Special operations : weak, and
  strict

Outline of the rest of the talk

- Consistency Clusters
- Transaction Types
- Correctness
    - *given configuration*
    - *cluster merging*
- Other Applications
    - *large databases*
    - *multidatabases*
- Related Work
- Conclusions and Future Work

Clusters

- Distributed database

  Transactions submitted from static and mobile hosts
  access both local and remote data

- Definition of clusters

  - based on physical location
    neighbor or strongly connected hosts
  - semantics of data
    location data
  - application-dependent
  - based on access patern
    private

- Dynamic configuration

- Definition of consistency

  Intra-cluster consistent
  among clusters m-degree consistency

Clusters

- Definition of degrees

  1. set a max on the deviation from core copies
  2. limit number of transactions operating on inconsistent data
  3. limit number of versions
  4. limit number of data items that can diverge
  5. limit number of copies per data item that can diverge

- Adaptability of degree

  Degree may vary based on the available bandwidth thus providing data of varying quality or detail

  Example:
  cooperative environment -- book editing

Transactions

- Rationale:

  - maximize local processing
  - support disconnected operation
  - use network only when necessary

- Two types of operations:

weak read	W_Read[x] reads any locally available copies Value written by last weak or strict write at that cluster
strict read:	S_Read[x] reads the most up-to-date value Value written by last strict write operation
weak write:	W_Write[x] writes locally available copies Not persistent, till cluster merging
strict write:	S_Write[x] writes any copies Persistent upon commitment

Transactions

- Two types of transactions:

  weak transactions
  strict transactions

- Two versions per cluster: core and quasi copies

- Translation function h: items -> copies

  weak read local (quasi or core)
  strict read core
  weak write local (quasi)
  strict write (core or quasi)

- Commutativity of operations:

	W_Read(a)	S_Read(a)	W_Write(a)	S_Write(a)
W_Read(a)			x	x
S_Read(a)				x
W_Write(a)	x		x	x
S_Write(a)	x	x	x	x

Two operations conflict if they access the same version of a data copy and one of them is a write operation

Transactions

- Commitment:

  local (cluster) -wise for weak transactions
  global for strict

- Discussion:

  - weak transactions: local
    strict transactions may need network access
    to guarantee consistency

  - weak reads: gather information for statistics
    approximate location

  - weak writes: update local data without confirming
    those updates immediatelly

    private data
    can be compensated

Correctness: - given configuration

- Intra-cluster schedule

An intra-cluster schedule *IAS* over $T=\{T_1, \ldots, T_n\}$ is a pair (OP, <) where < is a partial ordering relations and

[1] $OP = h(\bigcup_{i=0}^{n} T_i)$ for some translation function h

[2] For each T_i and all operations op and op' in T_i, if op' < op then every operation in h(op) is related by < to every operation in h(op')

[3] All pairs of conflicting operation are related by <

[4] For all read operations $S_Read_i[x_k]$ there is at least one $S_Write_j[x_k]$ such that $S_Write_j[x_k] < S_Read_i[x_k]$ and for each $W_Read_i[x_k]$ there is at least one $Write_j[x_k]$ such that $Write_j[x_k] < S_Read_i[x_k]$

[5] If $S_Write[x] < S_Read[x]$ and $h(S_Read[x]) = S_Read[x_k]$ then $S_Write[x_k]$ in $h(S_Write[x])$

[6] If $S_Write[x_k]$ in $h(S_Write[x])$ for some strict transaction T then $S_Write[y_l]$ in $h(S_Write[y])$ for all y written by T for which there is a y_l in the cluster

Correctness: - given configuration

- Intra-cluster correctness:

(weak correctness)

Definition:

An intracluster schedule is correct if its projections on strict transactions is 1SR and each of its projection on a cluster is conflict-equivalent to a serial schedule

- both weak and strict transactions get a consistent view

- weak transactions at different clusters may get different views

(strong correctness)

Definition:

1. An intracluster schedule is correct if its projection to strict transactions is equivalent to an 1C schedule S and also conflict equivalent to a serial schedule Q such that the order of transactions in Q is consistent with the order of transactions in S

2. An intracluster schedule is correct if its projection to strict transactions is equivalent to an 1C schedule S and each of its projections Q_i on a cluster i is equivalent to a serial schedule such that the order of transactions in Q_i is consistent with the order in S

Correctness: - given configuration

- Graph characterization:

  A modified serialization graph:

  - a replicated data graph for strict transactions
  - add edges between conflicting weak in the same cluster and weak and strict

  Type of edges:

  dependency edges
  precedence edges
  interference edges

- Properties (type of edges):

1. Between weak transactions of the same cluster there may exist dependency and precedence edges. Between weak transactions at different clusters there are no edges

2. Let WT a weak transaction and ST a strict transaction. Between them the following edges are possible:
   - a dependency edge from ST to WT
   - a precedence edge from WT to ST

Correctness: - given configuration

- Theorem

  Let S an intracluster schedule. If S has an acyclic serialization graph then S is strongly correct.

- Maintaining degrees:

  - Degree expresses divergence between core and quasi copies

  1. only accept weak writes with values inside a range
  2. bound number of weak transactions at each cluster
  3. after m strict writes on a data item update quasi copies
  4. allow weak operations on a limited number of data items
  5. bound number of weak writes such that their total number over all clusters is less than a limit

Correctness: - cluster merging

- Intercluster schedule:

An intercluster schedule based on an intracluster schedule (OP, <) is a pair (OP', <') where

[1] OP' = OP

[2] for any op and op' in OP' if op < op' in the intracluster schedule then op < op' in the intercluster schedule

[3] in addition, for each $W_Write_i[x_k]$ and $S_Read_j[x_k]$ either $S_Read_j[x_k] < W_Write_i[x_k]$ or $W_Write_i[x_k] > S_Read_j[x_k]$

- Intercluster correctness:

Definition:

An intercluster schedule is correct if it is based on a correct intracluster schedule and all strict transactions have the same read-from relation as in the intracluster schedule.

Correctness: - cluster merging

- Rollback weak transactions whose writes conflict with strict transactions

- Property:

  Only weak transactions in the same cluster read values written by weak transactions in that cluster

Thus,

transaction undoing is not propagated to other clusters

- Some weak may not need to be undone

Correctness: - cluster merging

- Graph characterization:

  - add edges to reflect conflicts between weak writes and strict reads

  - add edges to force conflicts among operations on quasi and core copies to induce:

    1. First, a write order as follows:
       if T and T' read any copy of an item
       T -> T' or T' -> T
    2. Then, a strict read order as follows:
       if a strict transaction ST reads-x-from ST'
       and a weak transaction WT follows ST'
       then ST -> WT

Correctness: - cluster merging

- Graph characterization:

Theorem

Let S be an intercluster schedule.
If it has an acyclic serialization graph
then it is correct.

Other Applications

- Large databases:

scale in geographical distribution

clustering based on location

scale in the number of sites

clustering based on semantics

- Multidatabases:

local sites --> clusters

local transactions --> weak

global transactions --> strict

core copies correspond to data created after the integration

quasi copies to data prior to the integration and are updated independently at each database

weak correctness respects autonomy

to reconcile core and quasi copies use for example polytransactions

Related Work

- Network partitioning:

  - clustering is elective
  - clusters may be partly connected
  - hybrid - optimistic (weak)
    pessimistic (strict)

- Read-only transactions:

consistency requirements: a. no consistency
b. weak consistency
c. strong consistency

strict read-only (strong consistency)
weak read-only: a. (none) if ignored
b. (weak) weak correctness
c. (strong) strong correctness

currency requirements:

strict (up-to-date)
weak (depends on degree)

- epsilon- serializability

weak read-only with no consistency requirements

Related Work

- Mobile databases:
  - adaptability
  - demarcation algorithm
  - the Bayou approach
- Mobile file systems:

  Coda:
  - treats disconnections as partitions
    optimistic schema
    optimization of logs

  File interface:
  - loose and strict read operations
    1USR

  Isolation-only transactions

Conclusions

- Conclusions:

  1. support autonomous operation
     weak transactions
  2. concern for bandwidth -- adaptability
     choose between weak and strict degrees
  3. locality
     dynamic configuration

  imprecise location data

- Future work:

  1. generalize semantics of weak operations

     e.g., vertical and horizontal partitions
     complex objects
  2. performance evaluation