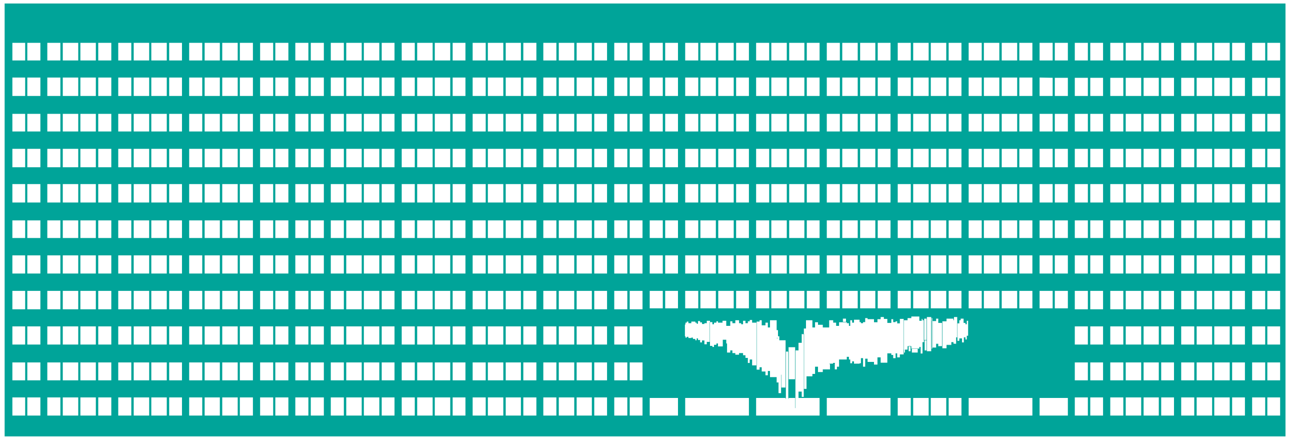
VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Limited Connectivity in Databases and WEB Systems



MS (Mobile Computing)
Lecture 4

Limited connectivity in Mobile Database Systems

Network partitioning vs. DO

- Network failure partitions the sites of a distributed database system into disconnected clusters.
- Trade-off between consistency, availability & the level of semantic knowledge
  - In network partition, transactions are executed at any partition of equal importance × in mobile computing transactions at the mobile host (considered second-class).
  - frequency of disconnections: network partitions correspond to failure behavior × some disconnections in mobile computing are foreseeable.

Hoarding & Hoard keys

- Extended database organization:
  - set of *hoard keys* along with the primary & secondary key for each relation.
  - hoard keys capture typical access patterns of mobile clients. Each hoard key partitions the relation into a set of disjoint logical horizontal fragments.
- Hoard fragments constitute the hoard granularity.
- Hoard keys can either be
  - Logical – index in DB for each logical fragments
  - Physical – mirroring physical fragments in DB.

Tentative Commit

1. Tentatively commit transactions at a disconnected unit & make their results visible to subsequent transactions in that unit.
2. Upon reconnection, a certification process is executed, during which each tentatively committed transaction is validated against an application or system defined correctness criterion.
3. If the criterion is met, the transaction is committed. Otherwise, it must be aborted, reconciled or compensated.

Isolation-only transactions

- IOT is a sequence of file access operations.
- Transaction execution is performed entirely on the client and no partial result is visible on the servers.
  - *First-class transaction* – no partitioned file access, i.e., the client machine maintains a connection for every file it has accessed. The result of a first-class transaction is immediately committed to the servers.
  - *Second-class transaction* – remains in the pending state till connectivity is restored. Result is held within the client's local cache and visible only to subsequent accesses on the same client. Guaranteed to be locally serializable among themselves.

IOT – reconnection

- First-class transaction is guaranteed to be serializable with all transactions previously resolved or committed to server.
- Second-class transaction is validated upon reconnection against one of two proposed serialization constraints:
  - global serializability
  - global certifiability
- Transactions that cannot meet these criteria are resolved incrementally, according to their local serialization order. An invalidated transaction is aborted, reexecuted or an ASR is automatically invoked.
- Manual resolution or repair is left the last resort.

Two-tier replication

- Replicated data have two versions at mobile nodes:
  - master version – records the most recent value retrieved when the site was connected.
  - tentative version – records local updates
- Two types of transactions:
  - Base transaction – works only on master data and produces master data. Base permitted only on connected sites.
  - Tentative transaction – works on local tentative data and produces tentative data. On reconnection, tentative tr. are reprocessed as base transactions. If they fail to meet an application-specific acceptance criteria, they are aborted and a message is returned mobile node.

Weak connectivity in MDBS

A) Mobile host simply submits operations to be executed on a server or an agent at the fixed network, 2 ways:

  a) each operation at a time
  b) complete transaction is one atomic unit

B) Local database processing on the mobile host

physical database design issues, e.g., appropriate fragmentation of the database & fragments allocation at fixed and mobile hosts

Transactions & weak connectivity

- *Mobile Transactions* – distributed transactions involving both mobile and fixed hosts:
    - traditional approaches (locks or timestamps)
    - open-nested transaction models: a mobile transaction → a set of relatively independent component transactions, some run only on the mobile host. Component transaction can commit without waiting for the commit of other CT.
- *Weak and Strict Transactions*:
    - Weak transactions local on mobile host, update weak copies
    - Strict transactions access strict copies (slower, permanent updates & current reads, adaptable – number of ST, …)
    - connectivity improves or application limit of copy dev. exceeded → weak copies integrated with strict.

Example: Bayou architecture

- Peer-to-peer architecture of replicated servers weakly connected to each other.
- Read-any and write-any available copy. Writes are propagated to other servers during pair-wise contracts (anti-entropy sessions).
- Sessions (instead of transactions) – sequences of read and write operation. *Session guarantees* – to avoid inconsistencies when accessing copies at different servers; (e.g. reads reflect previous writes). Adaptivity by individually selectable session guarantees, choices of committed or tentative data, age parameters on reads.
- Arbitrary disconnections

Other improvements

- *Complex Data Items* – large or complex objects split into smaller fragments, independent operation on each of the fragments.
- *Site escrow* – total number of instances of a given item is partitioned across a number of sites.
- *Fragmentation approach* – master copy of a large object at the fixed network are split into smaller (physical) fragments logically removed from the copy and loaded on the mobile host. Fragment re-assembling merge procedure.

Distributed object repositories

Organize data items as a collection of objects – unit of caching and replication. Very flexible, e.g. objects can encapsulate conflict resolution procedures. Can be built on top of an existing system.

- *Rover* – relocatable dynamic objects (RDO) exist. Each RDO has a home server that maintains the primary canonical copy. Clients import secondary copies of RDOs in their local caches.
- *Pro-motion* infrastructure – the unit of caching and replication is a compact – object encapsulating the cached data, operations for accessing them, state information, consistency rules and obligations (e.g. as deadlines).

Limited connectivity in WEB-based Systems

Cache management

- Cache management meets the needs of wired users, cache methods can trade inexpensive network bandwidth ↔ reducing storage size. Mostly used for browsing, no updates until requested page changes.
- Prefetch/Hoarding – usually based on the user's surfing history list, ~ 2% is usable according to some studies.
    - To support disconnected operation, all documents of pages visited previously or of pages expected to be visited based on the user's profile can be prefetched.
    - HTTP server can monitor access patterns and periodically augment objects with information about what objects the client should prefetch, if it requested a particular document.

Browser Sessions

- Cache methods that
  - (a) purge the cache at the end of a browser session
  - (b) let cached objects persist across sessions with updates occurring once on first reference in the session.
- Also, advanced options that cause cache objects to be updated on every reference or never to be updated.
- Cross-session persistence of cached objects is critical during disconnections permitting disconnected clients to end and start sessions as they wish.

Cache management

- *Cache Updates* – optimistic techniques looking for modifications to stored objects based on elapsed time.
    - coherency interval to measure the elapsed time permitting web users to adjust the update frequency of cached web-objects

- *Cache Validation at Reconnection* – a single asynchronous coherency check for all cached objects that are older than coherency interval, or by initiating coherency checks only on object references.

Cache misses

- An “asynchronous-disconnected” mode permitting requests to be automatically queued when connectivity is lost & resumed when connectivity is re-established. Users can issue multiple web requests without having to wait for their replies. Arriving responses are queued.

- Rover uses queued RPC to develop a non-blocking operation for browsers to allow users to click ahead. Requests are stored in the client's “operation log” that is executed at reconnection. The client is notified, when its request is satisfied.

Weak Connectivity in Web-Based Systems

- *Server Cache* – a cache in the fixed network, addition to client cache.
    - Prefetching to server cache hides some latency without affecting the wireless link.
    - Can facilitate user-specified data reductions.
- *Distillation* and *Refinement* – graphics included in web pages increases latency of their loading.
    - Distillation – highly lossy, real-time, datatype-specific compression preserving most of the semantic content of a document.
    - Refinement – user can request (a higher quality) full image (or its part).

D&R Example: Pythia

- An agent proxy
  1. distills and forwards an image to the client
  2. caches a copy of the original image locally in case the client requests refinement.

- Users can register their capabilities with the Pythia proxy → distillation can be tuned based on their configuration. Current connectivity can be set to influence the distillation process

Connection overhead

- Each request requires the browser to open a new TCP/IP socket, this increases user response time and network traffic.
  → prefetching or pipelining of web content might cause more problems than benefits, especially if usability of the prefetched objects is low (like ~2%).

- A single, long-lived TCP/IP connection between the mobile browser and the fixed network is required.

Differencing approach

Takes advantage of the fact that only small parts of cached objects are modified:

1. When an application requests an object, the object is cached at both the client and the server. Both cache a common base object.
2. When a client requests an outdated cached object (i.e. whose coherency interval has expired), the request is directed to the server's cache.
3. If this also outdated, then to the web server for a fresh copy. The server computes the difference between its local copy and the response and sends the difference to the client.
4. The client merges the difference with its cached object to create the browser's response.