Lesson 02
Data Hoarding and Caching
Large databases—kept on servers, remote computing systems, or networks

- A mobile device cannot store a large database
- Retrieving the required data from a database server during every computation—impractical due to time constraints
Hoarding (caching) of specific database in mobile devices

- A mobile device— not always connected to the server or network, neither does the device retrieve data from a server or a network for each computation.
- Rather, the device caches required specific data, which may be required for future computations, during the interval in which the device is connected to the server or network.
Hoarding of Cached Data

- Database architecture—Two-tier or multi-tier databases
- Databases reside at the remote servers and the copies of these databases are hoarded and cached at the client tier
Synchronizing the local copies at the device

- At tier 2 or tier 3, the server retrieves
- Server transmits the data record(s) to tier 1 using business logic and sends and synchronizes the local copies at the device
- Local copies function as device caches
Advantage of hoarding

- No access latency (delay in retrieving the queried record from the server over wireless mobile networks)
- The client device API has instantaneous data access to hoarded or cached data
- After a device caches the data distributed by the server, the data is hoarded at the device
Disadvantage of hoarding

- Needs maintain the consistency of the cached data with the database at the server
Distributed data caches in mobile devices

[Diagram]

© Oxford University Press 2007. All rights reserved.
Architecture of distributed data caches in mobile devices

- Similar architecture to distributed cache memory in multiprocessor systems.
- The copies cached at the devices are equivalent to the cache memories at the processors in a multiprocessor system with a shared main memory and copies of the main memory data stored at different locations.
Architecture for a distributed cache memory in multiprocessor systems
1. Using the pushed (disseminated) data records from a server
   • Caching leads to a reduced access interval as compared to the pull (on-demand) mode of data fetching
   • Also reduces the dependence on pushing precedence at the server
Caching of data records at Client device

2. Can be based on pushed ‘hot records’
3. Cost-based data replacement or caching—Caching can be based on the ratio of two parameters—access probability (at the device) and pushing rates (from the server) for each record
Cost-based data replacement

Method

- Least frequently pushed records and the pushed records having larger access time placed in the database at the device.
- This access method, therefore, use the ratio of two parameters—average access time between two successive instances of access to the record and pushing rates for the record.
Pre-fetching

- Alternative to caching of disseminated data entails requesting for and pulling records that may be required later
- Perfetched — keeping future needs in view instead of caching from the pushed records
Pre-fetching

- Reduces server load
- Reduces the cost of cache-misses
- The term ‘cost of cache-misses’ refers to the time taken in accessing a record at the server in case that record is not found in the device database when required by the device API.
Cache consistency

- Also called cache coherence
- Requires a mechanism to ensure that a database record identical at the server as well as at the device caches and that only the valid cache records are used for computations
Cache access Protocols based on Caching Invalidation Mechanisms

- Access protocols cached record at the client device invalidated
  - Due to expiry or modification of the record at the database server
Cache invalidation

- A process by which a cached data item or record becomes invalid and thus unusable because of modification, expiry, or invalidation at another computing system or server.
- Cache invalidation mechanisms are means by which the server conveys this information to client devices.
Four possible states ($M$, $E$, $S$, or $I$) of a data record $i$ at any instance at the server database and device $j$ cache.

- $E_{MW}$: Exclusive at the device (server does not modify it)
- $S_{MW}$: Shared with server or other devices
- $M_{MW}$: Modified at the device after new record from server
- $I_{MW}$: Invalidated after invalidation report from server

$E_{SW}$: Exclusive at server and the device record does not affect it
$S_{SW}$: Shared with other devices or server
$I_{SW}$: Invalidated at server, report is sent to the devices
$M_{SW}$: Modified at the server
Cache-invalidation mechanisms under the MESI protocol

- Entail that each record (line) in a cache has a tag to specify its state at any given instant and the tag is updated (modified) as soon as the state of the record changes.
MESI Protocol one of four possible tags

- Assigned cache state
  1. M— Modified (after rewriting)
  2. E— Exclusive
  3. S— Shared
  4. I — invalidated (after expiry or when new data becomes available) at any given instance.
Two-tier or multi-tier databases

Databases reside at the remote servers and the copies of these databases are cached at the client tiers

Computing API at the mobile device (first tier) uses the cached local copy
Summary

- Architecture of distributed data caches in mobile devices and a similar architecture of distributed cache memory in multiprocessor systems
- Cache Access Protocols
- Cache Invalidation Mechanisms
- MESI protocol
End of Lesson 02
Data Hoarding and Caching