A User-defined Probabilistic Data Consistency = Opportunistic Data Access + Lazy Request

Yumin Lee and Sunho Lim
Department of Computer Science, Texas Tech University

Introduction & Motivation

- Mobile Environment
  - Frequent disconnections
  - Mobility of the users
  - Battery limitation
  - Scarce wireless bandwidth

- Caching Techniques
  - Caching frequently accessed data items in a local storage
  - Relieve network traffic
  - Improve information accessibility and availability
  - Save wireless bandwidth and battery

- Challenges:
  1) Non-negligible long query delay; caused strong consistency
  2) Does every user need the same consistency?

- Problems:
  1) Is all information update sensitive?
  2) Does every user need the same consistency?

- Contribution of Our Work:
  1) How does a user-defined probabilistic data consistency affect the data access delay?
  2) How to integrate our scheme into IR and UIR methods?

IR- and UIR-based Caching Schemes

- Server
  - Broadcasts periodical IRs and UIRs
  - IR
    - A list of ids and timestamps of recently update items
    - An update history witnessed during the last \( L \cdot |w_b| \) broadcast intervals, where \( w_b \) is a broadcast window
  - UIR
    - ids and timestamps updated since the last IR

- Users
  - Must wait for the next IR/UIR whenever a query occurred
  - Long query delay (L or \( L/3 \) for IR and UIR respectively)

A User-defined Prob. Data Consistency

- A User-defined Probabilistic Data Consistency
  - Each user sets its own target consistency (\( 0 < r < 1 \))
  - Each user maintains its own current consistency (\( r_{cur} \))
  - Counting a number of invalid items within a consistency window (\( w_c \))
    \[
    r_{cur} = 1 - \frac{\sum Flag[i]}{|w|}
    \]
  - Consistency condition:
    1) Satisfied if \( r_{cur} \leq r_{\text{target}} \)
    2) Not satisfied otherwise

Opportunistic Data Access

- A query can be answered without waiting for the next IR/UIR
  - Condition:
    1) A cached item is valid
    2) Consistency Condition is satisfied (\( r_{cur} \leq r_{\text{target}} \))

Lazy Request

- A request query can be answered without sending an uplink request
  - Condition:
    1) A cached item is invalid
    2) Consistency Condition is satisfied (\( r_{cur} \leq r_{\text{target}} \))
    3) Have never been accessed before

Finite State Machine

- The opportunistic data access and lazy request framework

Performance Analysis

- A comparison of the query delay
- A comparison of cache hit ratio
- The number of opportunistic data accesses
- The number of lazy requests
- Current consistency changes over simulation time

Conclusion

- Each node can define its own consistency level flexibly and independently
- Effectively balance the data accessibility and query latency
- Each user can achieve a goal set by own consistency level