Cooperative Caching with Return on Investment

Gala Yadgar  
Technion

Michael Factor  
IBM Research

Assaf Schuster  
Technion
Rethinking Cooperative Caching

Traditionally:
- “Global” LRU management
  - Global optimization
- Central ownership: cooperation is mandatory

- What about selfish clients?
Selfish Clients: Why?

- Large scale resource consolidation
  - Storage clouds
  - Computational grids
  - Large scale data centers
  - Federated CDNs
  - Resources owned/chartered by different entities

- Limited resources
  - Buffers, energy, bandwidth, processing
Selfish Clients: How?

• Refuse to cache unneeded blocks
  – Manage cache according to their own policy
    • Traditionally assumed irrelevant

• Refuse to SERVE blocks
  – Even if they are cached
    • Traditionally assumed cost is negligible

• There is a scale of selfishness
Approaches for Selfish Clients

- Storage caching
  - Limited “private” partition
  - Load balancing
- Peer-to-Peer
  - “Tit for tat”, reputations
  - Virtual currency
- Job scheduling
  - Market based models

Insufficient
Unfair
Short term encounters
Computation and message intensive
A New Model

• New operation: SERVE
  – Added to READ, WRITE etc.
  – Define cost in terms of client’s objective function

• Measure utility derived from cache content
  – Accurate or estimate

• Selfish clients cooperate iff
  \[ \text{Utility} \ (\text{cache w/o cooperation}) < \text{Utility} \ (\text{cache with cooperation}) \]
  - Cost (total access to remote caches)
  - Cost (total SERVES to peers)
In The New Model

- When to cooperate?
- How to cooperate?
Cooperative DHT

- Distributed Hash Table [Stoica et al. 2003]
  - Distributed key assignment

- **Altruistic** clients
  - Cache: blocks they are assigned
  - SERVE: all requests
Cooperative Peer to Peer

• **BitTorrent based caching** [Cohen 2003]
  - Server tracks accesses (“tracker”)

• **Selfish clients**
  - **Cache**: LRU without replication
  - **SERVE**: peers with positive balance (“tit for tat”)

Altruistic  Selfish
Cooperative ARC

- Adaptive Replacement Cache  
  [Megiddo and Modha 2003]  
  - New blocks in L\(_1\)  
  - Useful blocks in L\(_2\)  

- **Very selfish** clients  
  - Cache: ARC with replication in L\(_2\)  
  - SERVE: peers with positive balance
Utility Based Cooperative Caching

- Use utility calculations [Yadgar et al. 2008]
  - Block accesses hinted or derived
  - Server constructs configuration

- Very selfish clients
  - Cache: by configuration
  - SERVE: by configuration

→ As long as utility increases
Decision Support Workload (TPCH)

- Uniform distribution
- Short term information
- Partial correlation

Performance

20 peers

No Coop

Altruistic

Selfish
OLTP Workload (TPCC)

- Long tail distribution
- Partial information
- Strong correlation

Cost (SERVE)

Total Cost (I/O)

20 peers
OLTP Workload

- Long tail distribution
- Partial information
- Strong correlation
“YouTube” Workload

- Heavy tail distribution
- General information
- Medium correlation
- Cheap SERVE

Performance

50 peers

Selfish Altruistic
No Coop
TPCH Queries

- Uniform distribution
- Accurate information
- Medium correlation

Queries 3, 10, and 18

Performance

No Coop

Altruistic

Selfish
Conclusions

• When to cooperate?
• How to cooperate?
Conclusions

- Blocks accessed uniformly?
  - Yes
    - Correlated workload?
      - Yes
        - Cooperate. Altruistic clients will do better.
      - No
        - Cooperate with explicit block allocation.
  - No, skewed accesses
    - Can compute utility?
      - Yes, with hints or statistics
        - Very low
        - Medium/High
          - What is the cost of SERVE?
            - Not accurately
            - Medium/High
            - Medium/high
              - Don’t cooperate
      - Not accurately
        - Very low
          - Cooperate. Selfish clients will do better.
Coming Up

- Saving energy
- Federated CDN

THANK YOU!