CS848 Paper Presentation

Scalable Query Result Caching for Web Applications

Garrod, Manjhi, Ailamaki, Maggs,
Mowry, Olston, Tomasic

PVLDB 2008

Presented by Rehan Rauf

David R. Cheriton School of Computer Science
University of Waterloo

18th January 2010

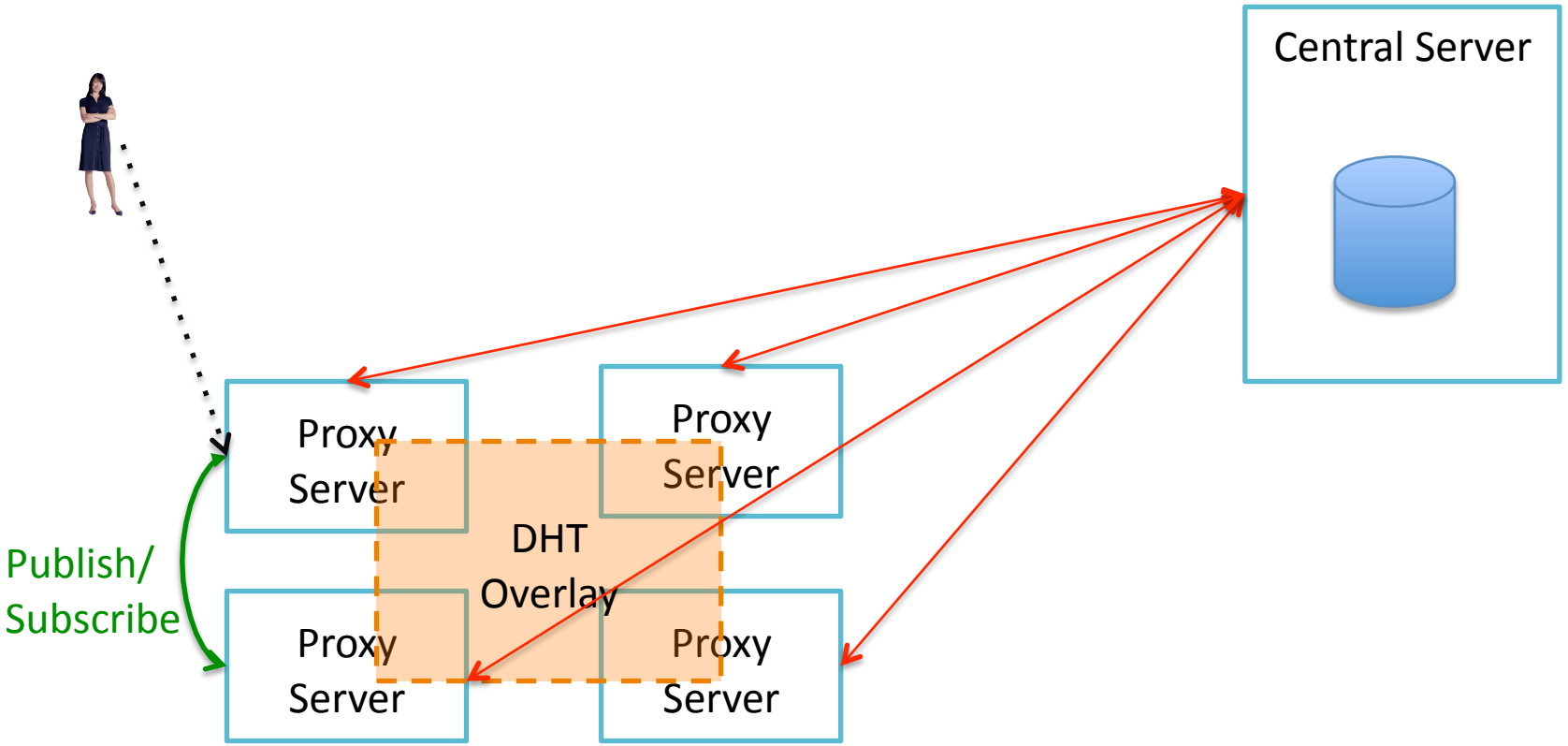
Problem

- Central database server becomes a bottle-neck in Content Distributed Networks

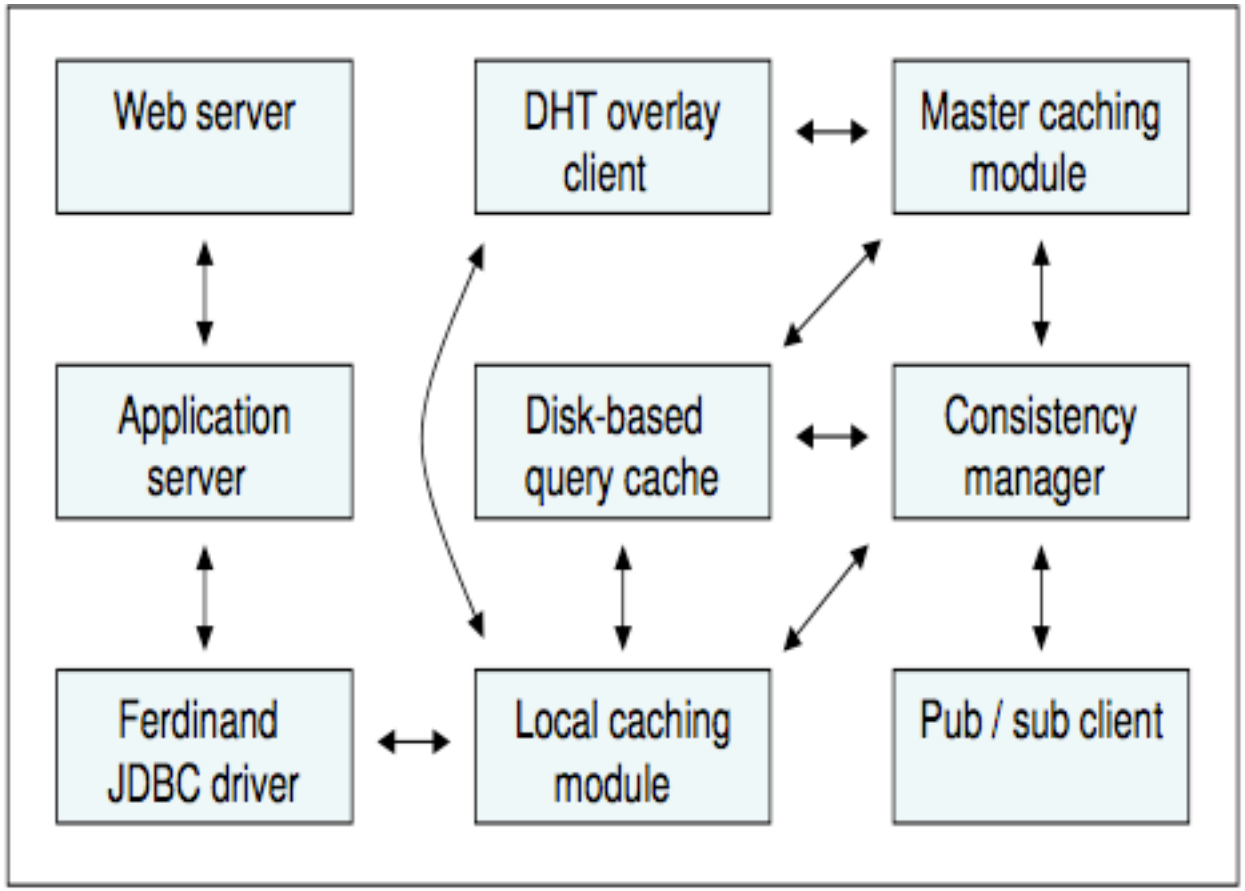
- Replication of database among multiple servers removes bottle-neck but …
    - requires low latency consistency which conflicts with low latency between user and server
    - does not scale linearly with cost.

- Traditional proxy-cache based solutions remove bottle-neck but …
    - inefficiently maintain consistency
    - scalability is limited by low cache hit-rate

High-level Architecture of Ferdinand

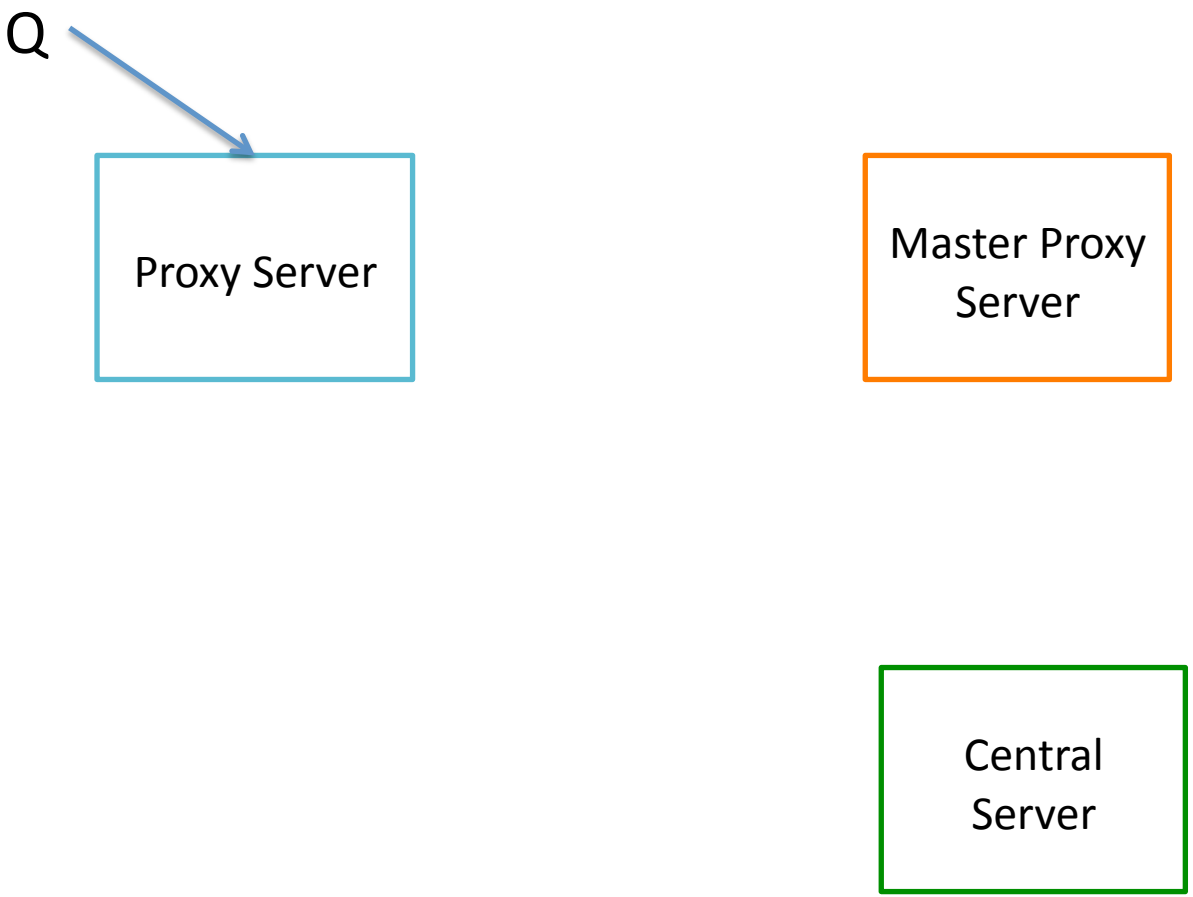


Ferdinand Proxy Server

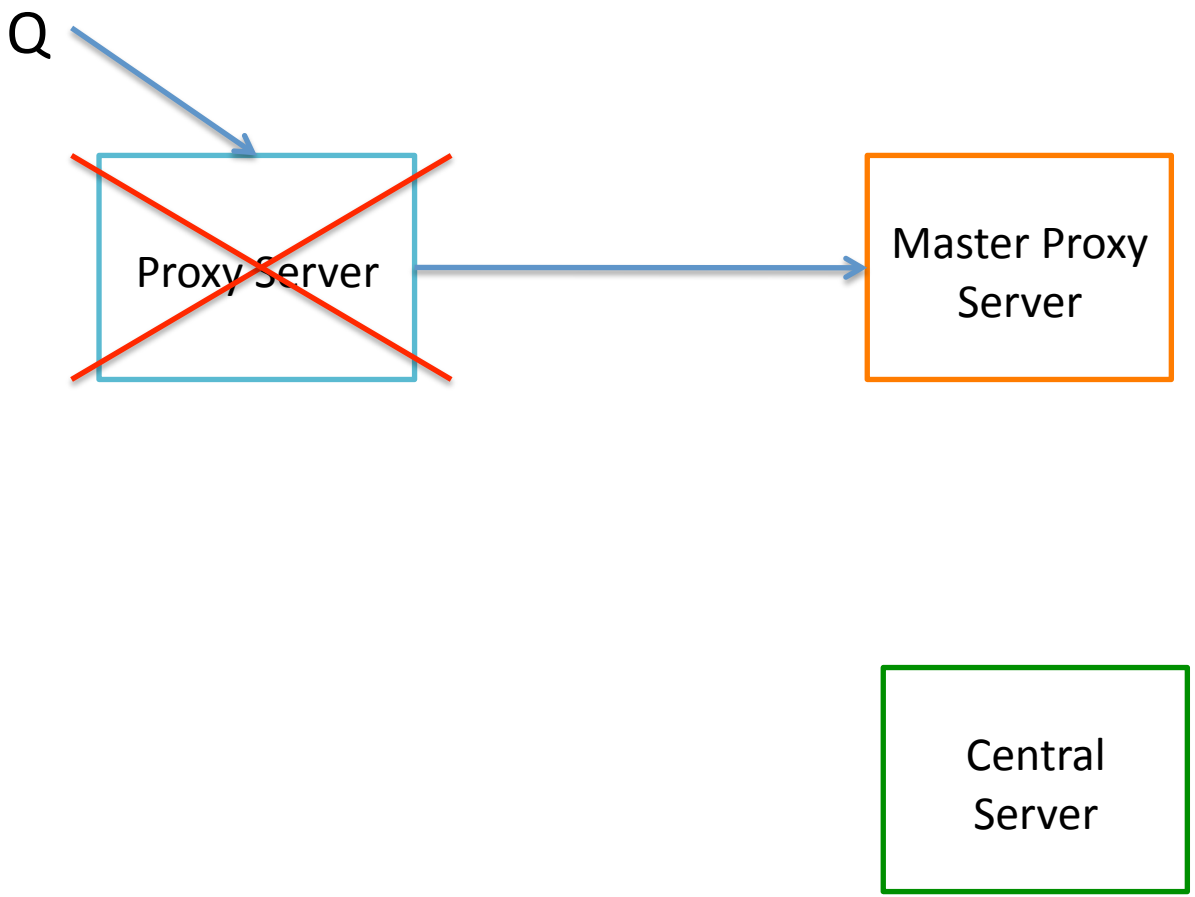


Operations

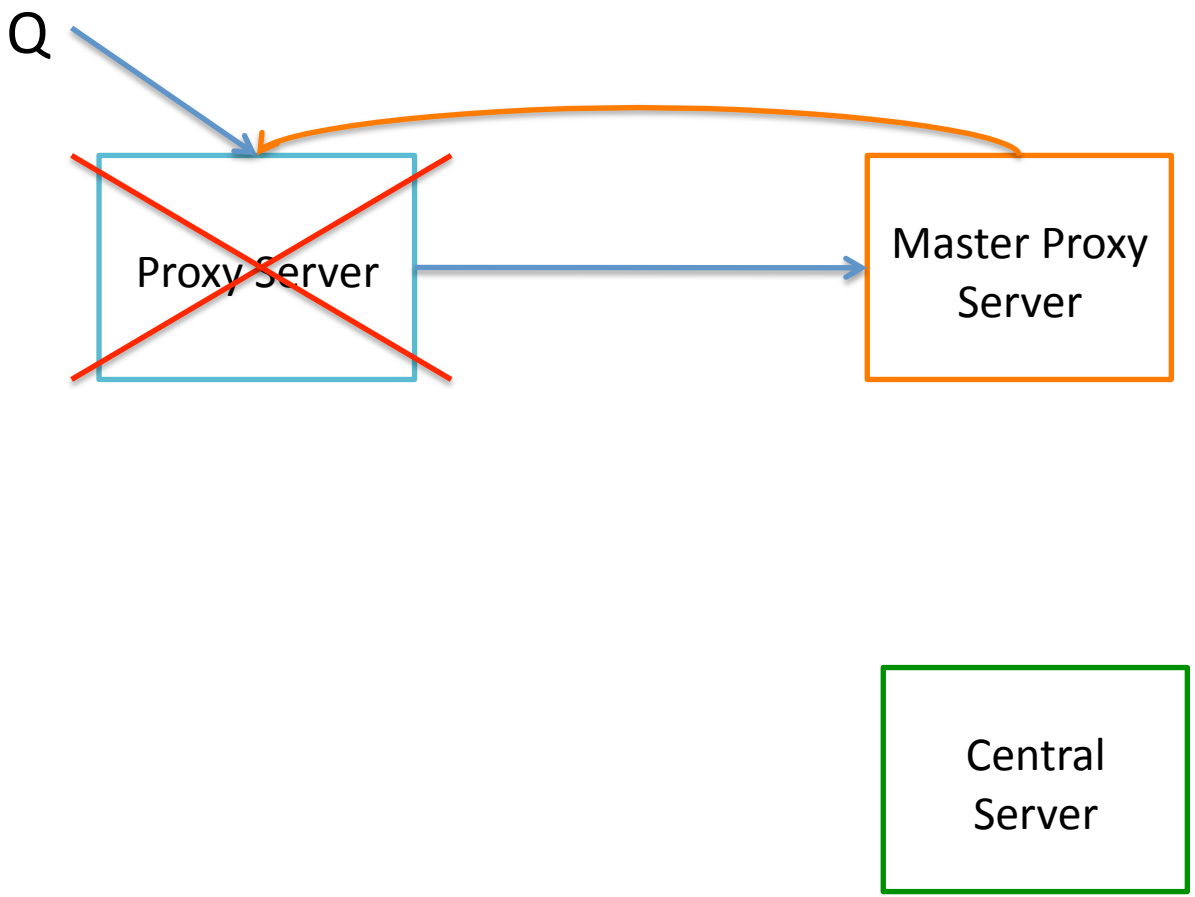
Each query has a master proxy server



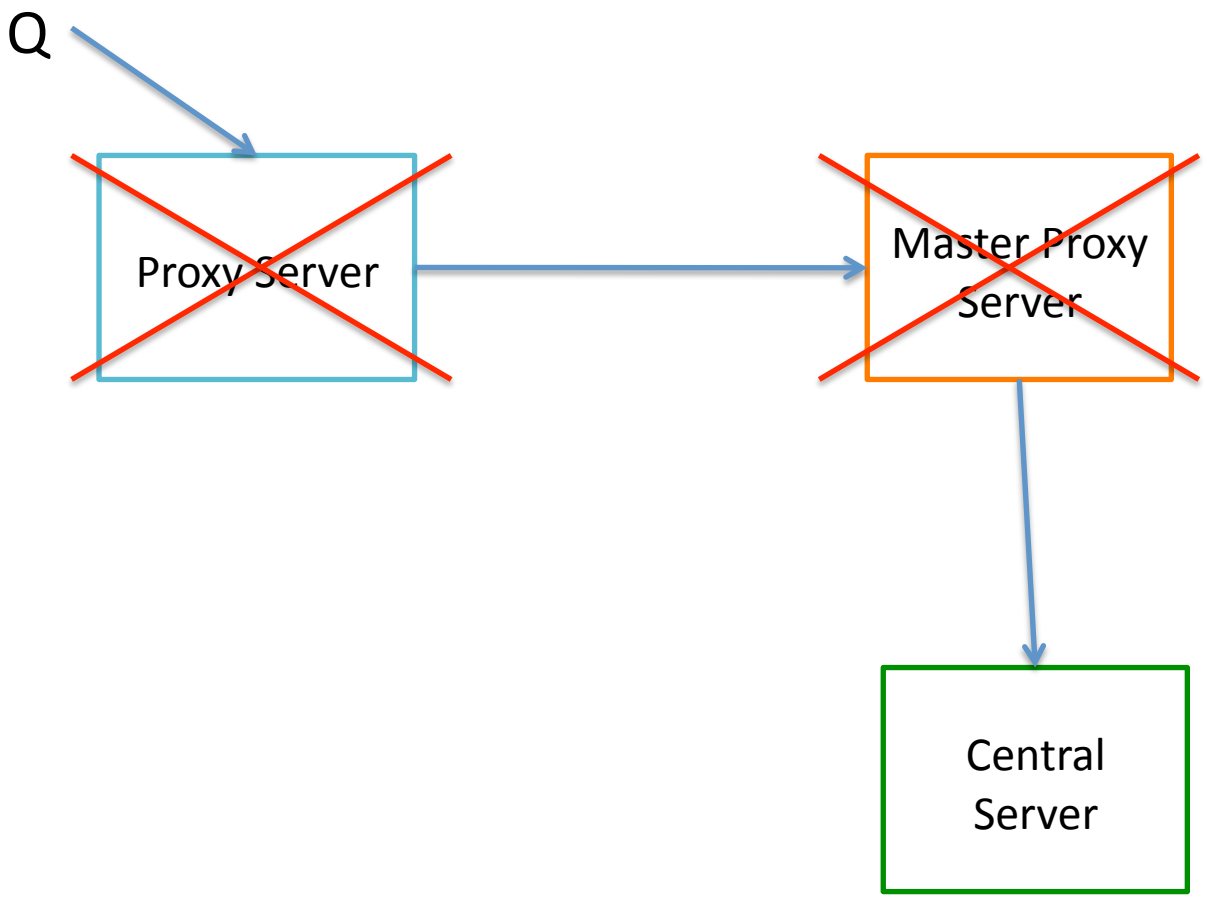
Operations



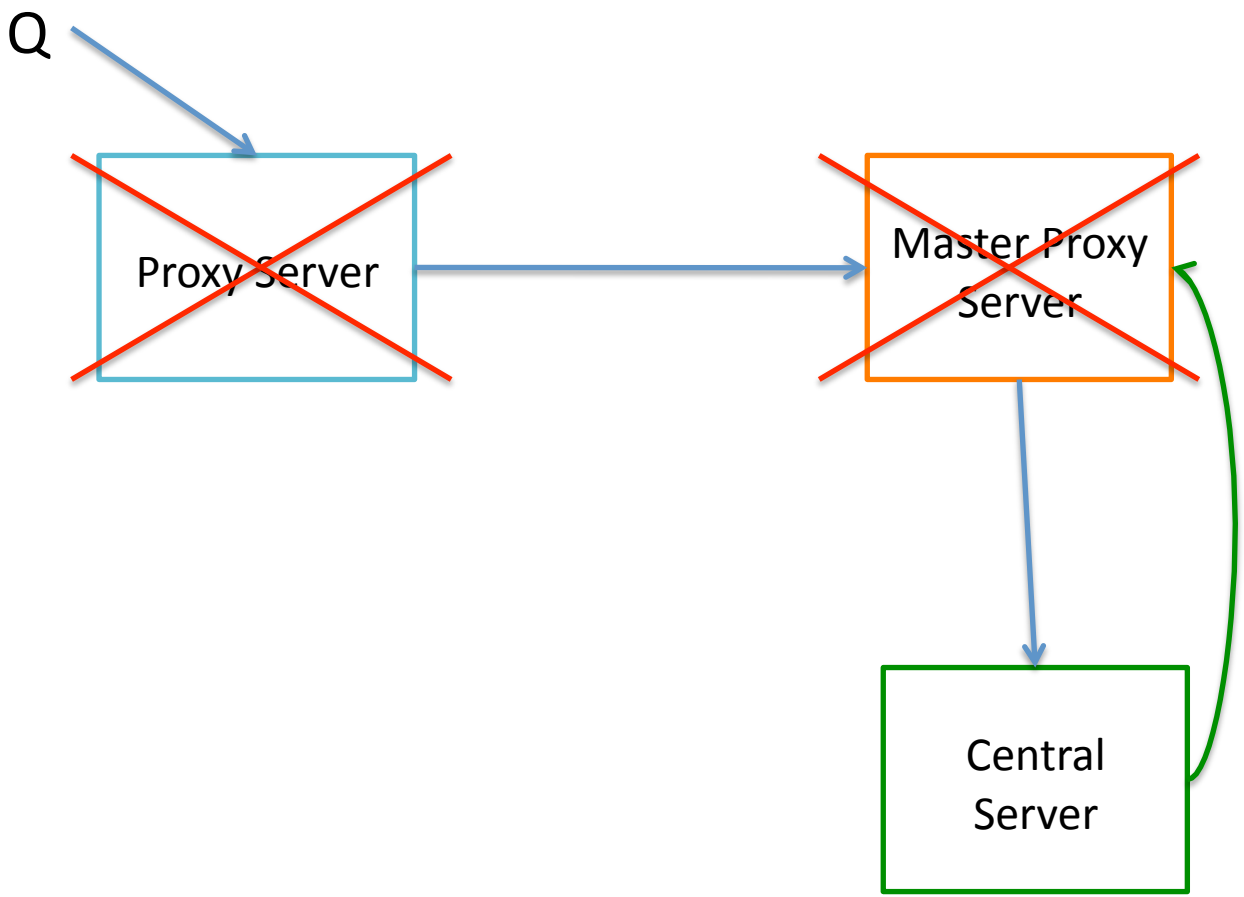
Operations



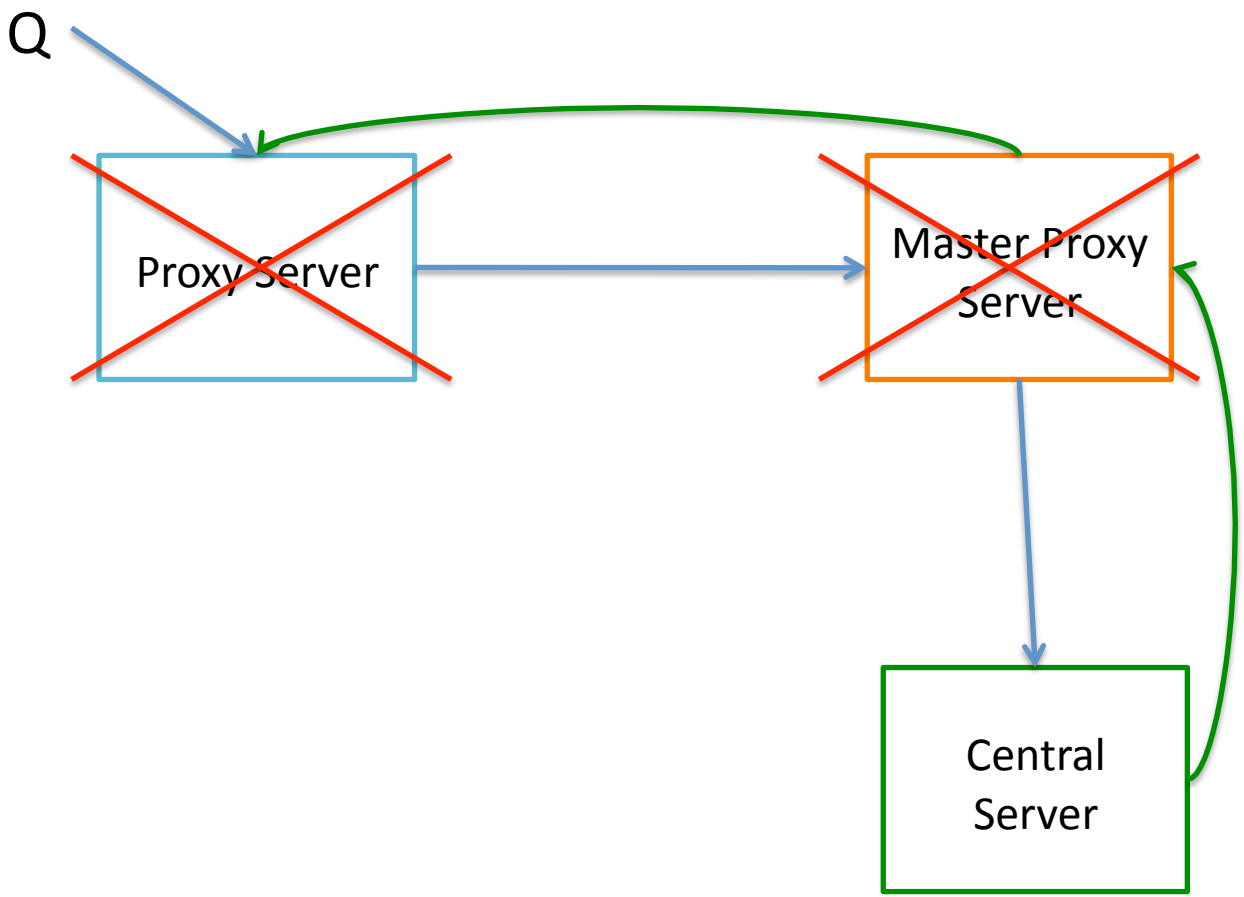
Operations



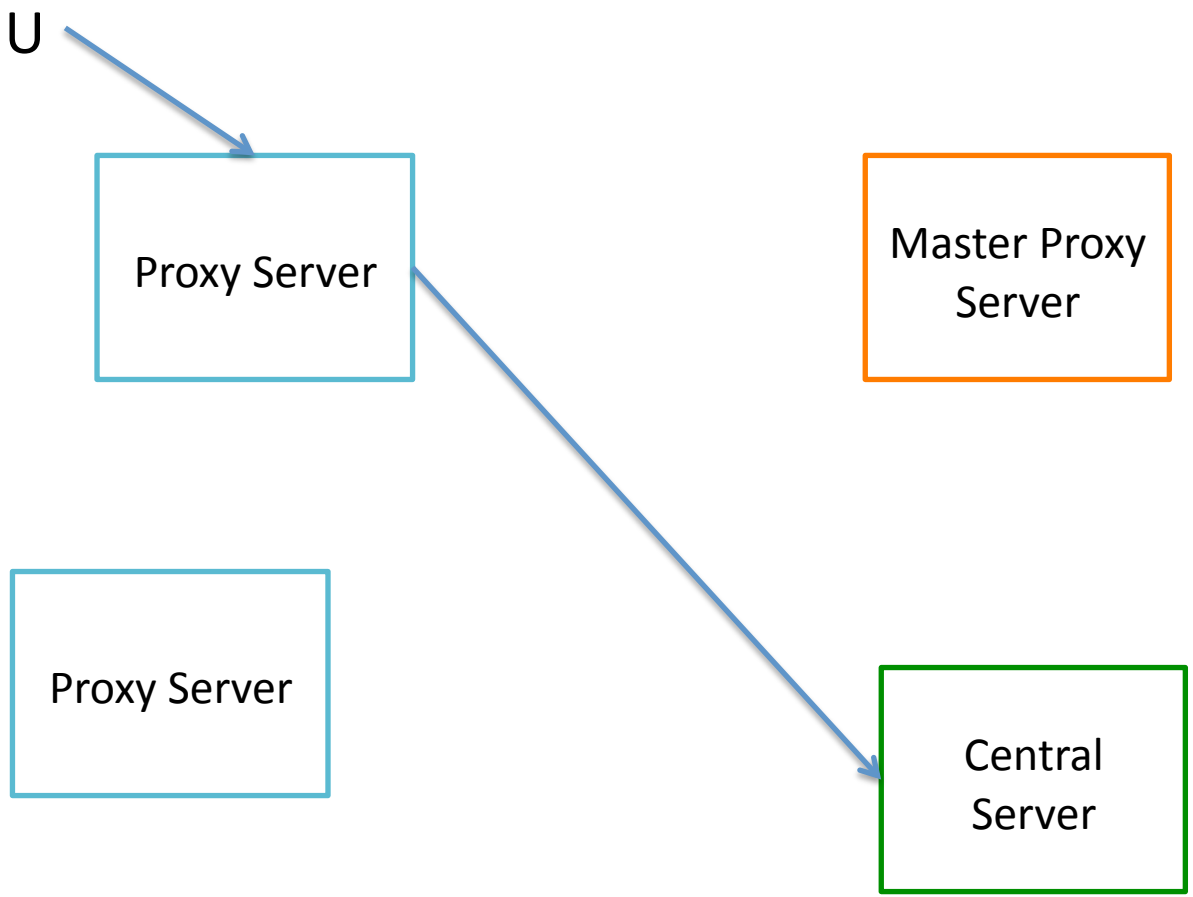
Operations



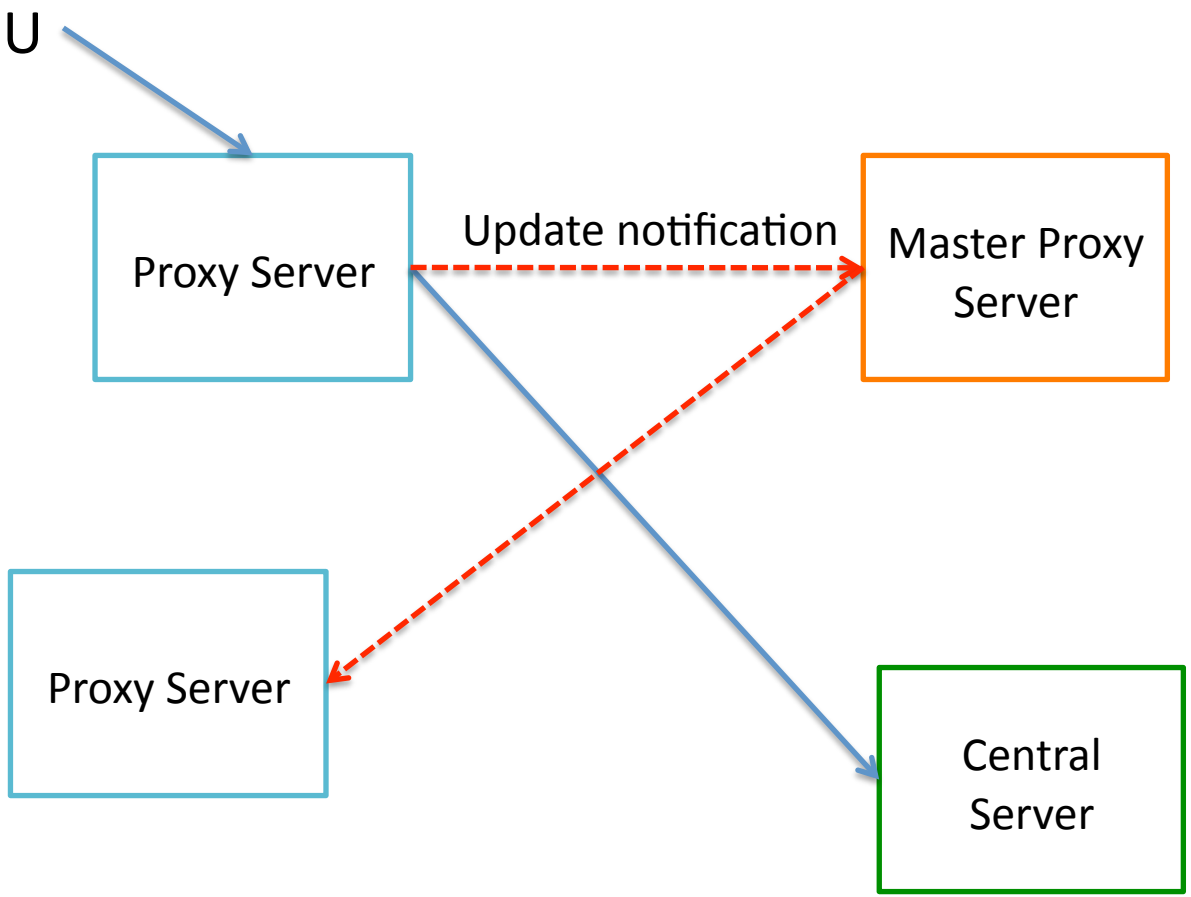
Operations



Operations


U
Proxy Server
Master Proxy Server
Proxy Server
Central Server

Operations


U
Proxy Server
Update notification
Master Proxy Server
Proxy Server
Central Server

Proxy Server Cache

• When proxy server receives update notification, the consistency module invalidates the query result.

• Even if results can be processed using previously cached results, it is not done

Advantages/ Disadvantages

- Advantages
  - High cache hit rate.
  - More work offloaded from the backend server

- Disadvantage
  - Latency cost of an over all cache miss is greater

Consistency Management

• Any proxy caching a particular query must be notified when an update affects the result of that query.

• Query Update Multicast Association (QUMA) is used to ensure this requirement.

QUMA Solution

- Multicast groups are created.
    - Offline analysis of application's template database queries is performed.
    - Independence analysis is done to determine independence of query-update pairs.

- Goals
    - Any update notification published to a group should affect each query subscribed to that group
    - Related queries should be clustered into same multicast group to reduce number of notifications.

- Each cached query subscribes to appropriate multicast groups.
- Updates are published to these groups and hence reach the appropriate proxy server.

QUMA Example

Template U1:
```
INSERT INTO inv VALUES
  (id = ?, name = ?, qty = ?,
  entry_date = NOW())
```

Template U2:
```
UPDATE inv SET qty = ?
  WHERE id = ?
```

Template Q3:
```
SELECT qty FROM inv
  WHERE name = ?
```

Template Q4:
```
SELECT name FROM inv
  WHERE entry_date > ?
```

Template Q5:
```
SELECT * FROM inv
  WHERE qty < ?
```

QUMA Solution

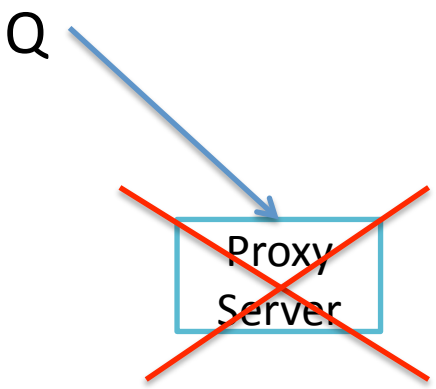
Template	Associated Multicast Groups
Template U1	{GROUPU1:NAME=?, GROUPU1}
Template U2	{GROUPU2}
Template Q3	{GROUPU1:NAME=?, GROUPU2}
Template Q4	{GROUPU1}
Template Q5	{GROUPU1, GROUPU2}

- Selection predicates are only practical when they are equality based.
- The process is not automated yet.

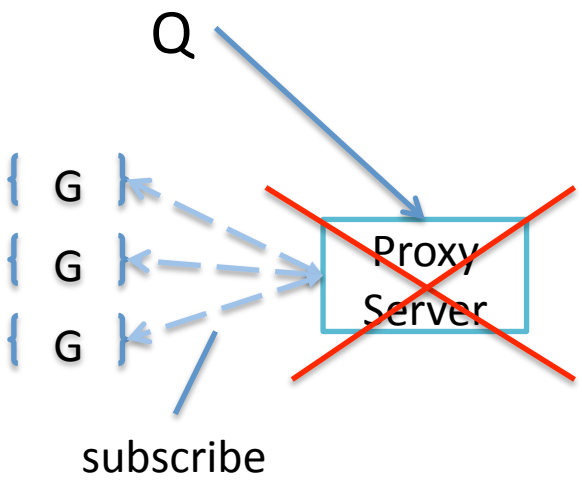
Consistency Management

- For each multicast group, there is a master multicast group.
    - used for communication with master proxies.

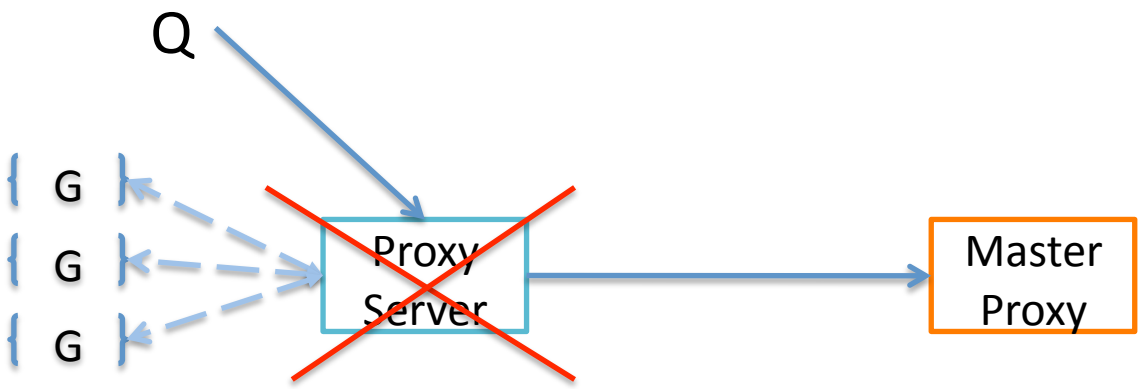
Cache Miss



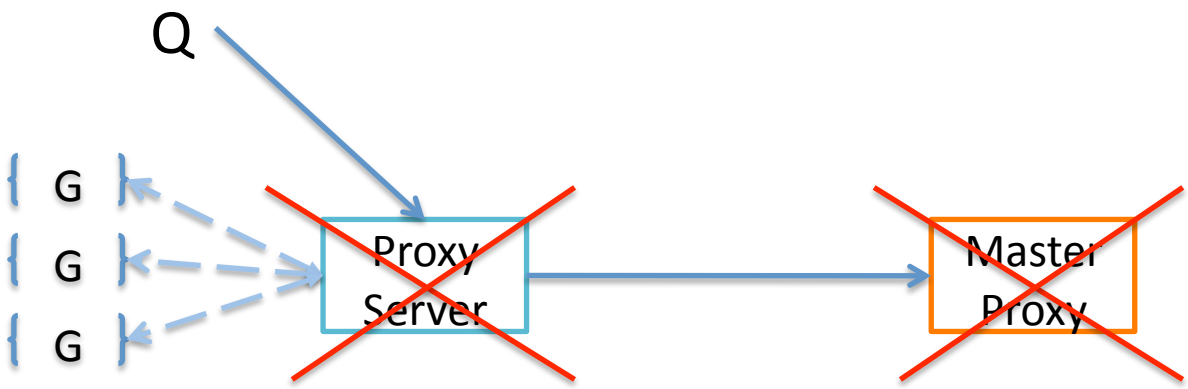
Cache Miss



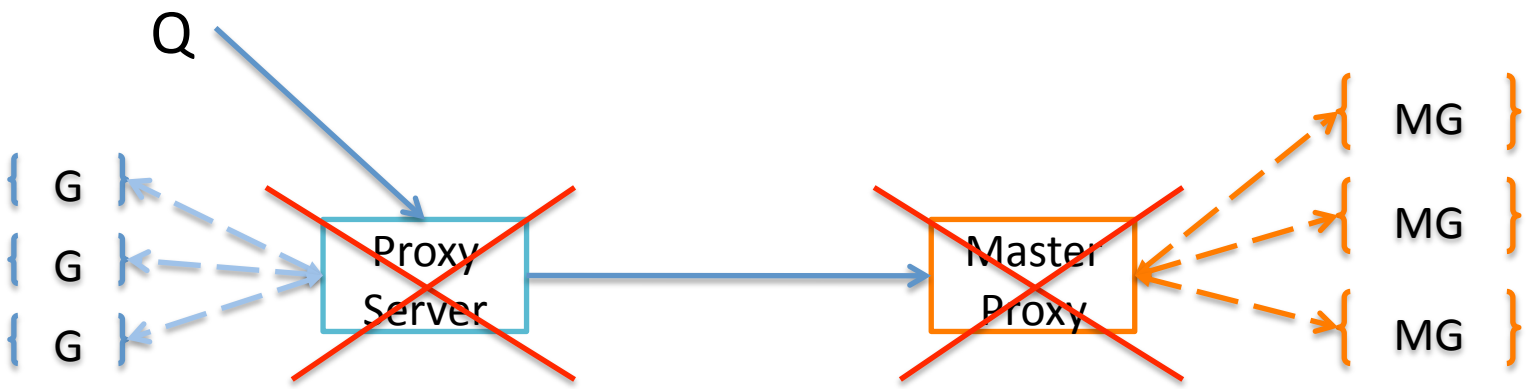
Cache Miss



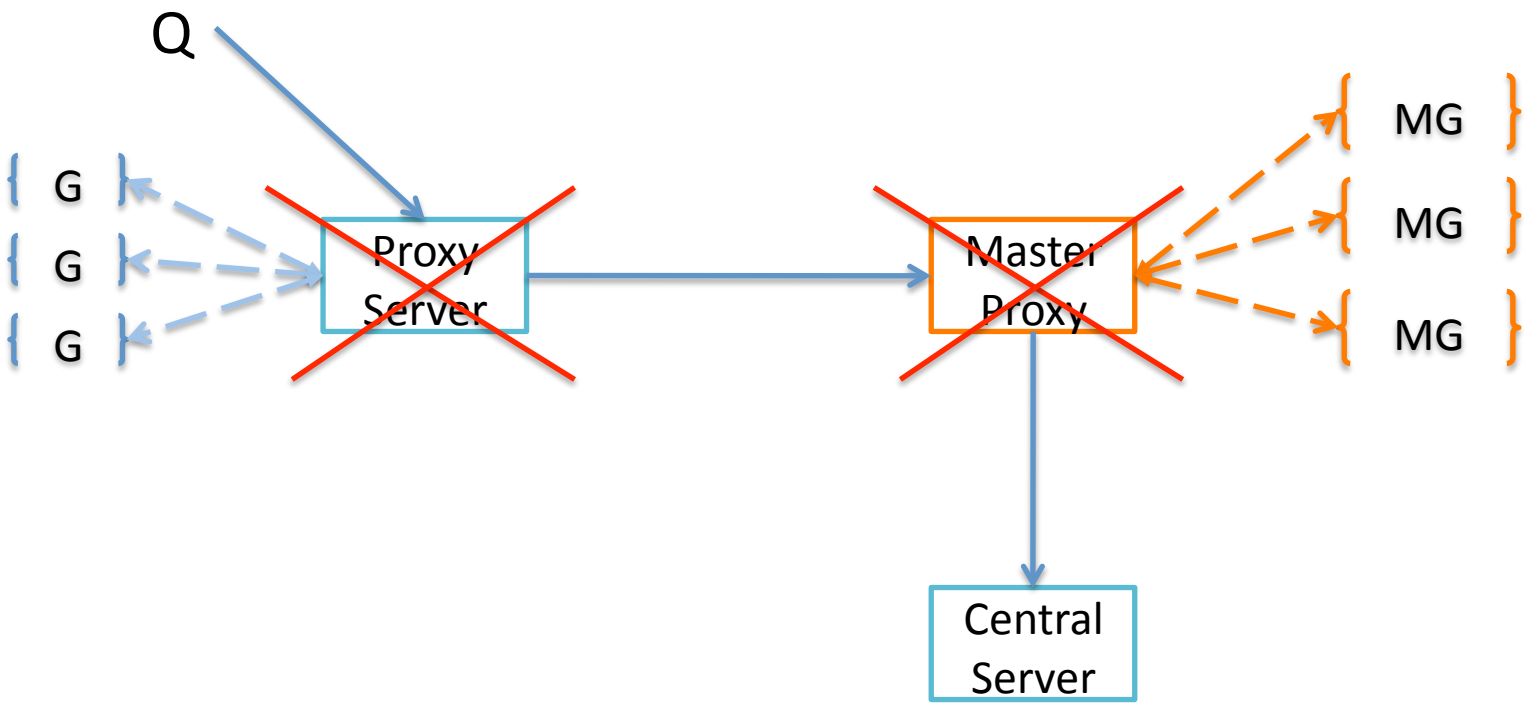
Cache Miss



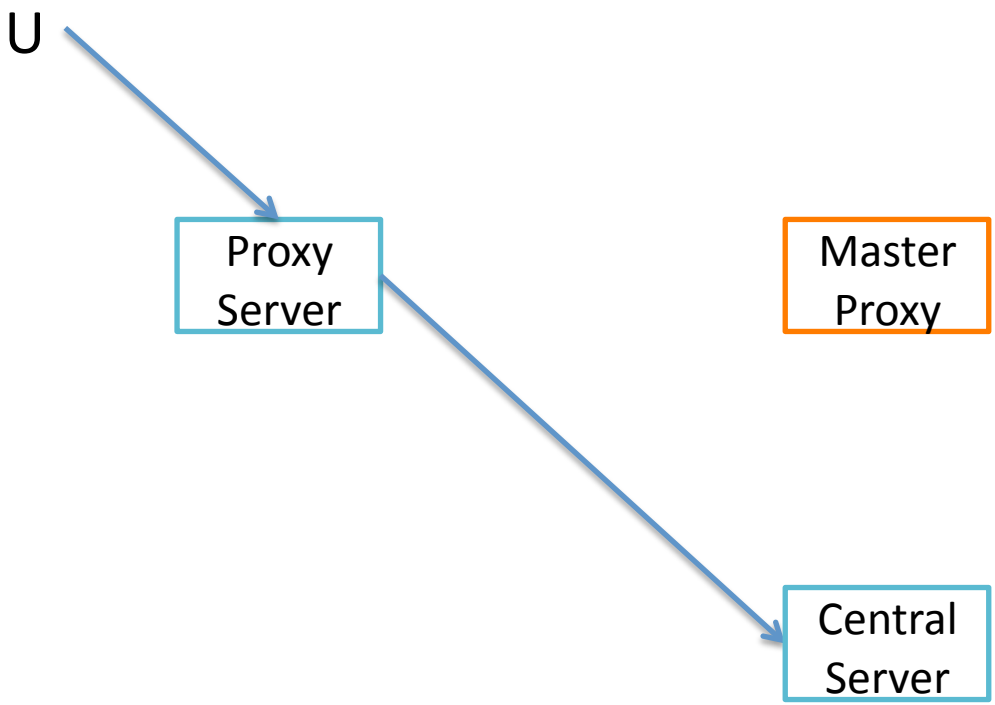
Cache Miss



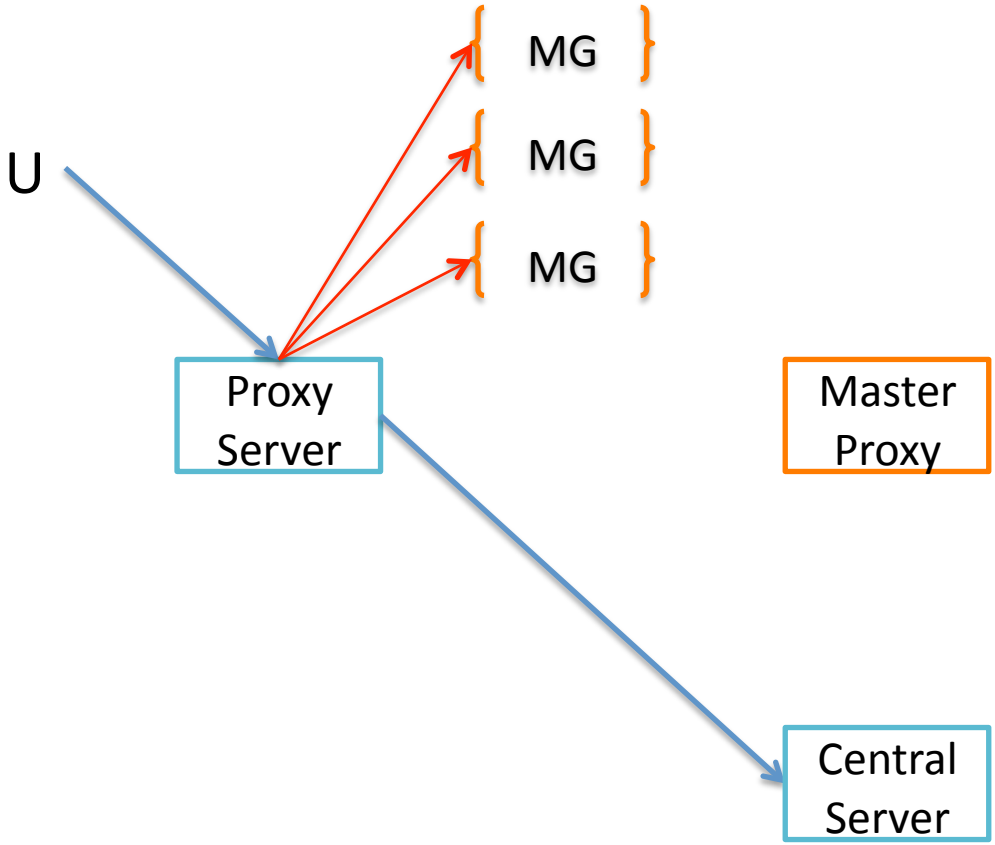
Cache Miss



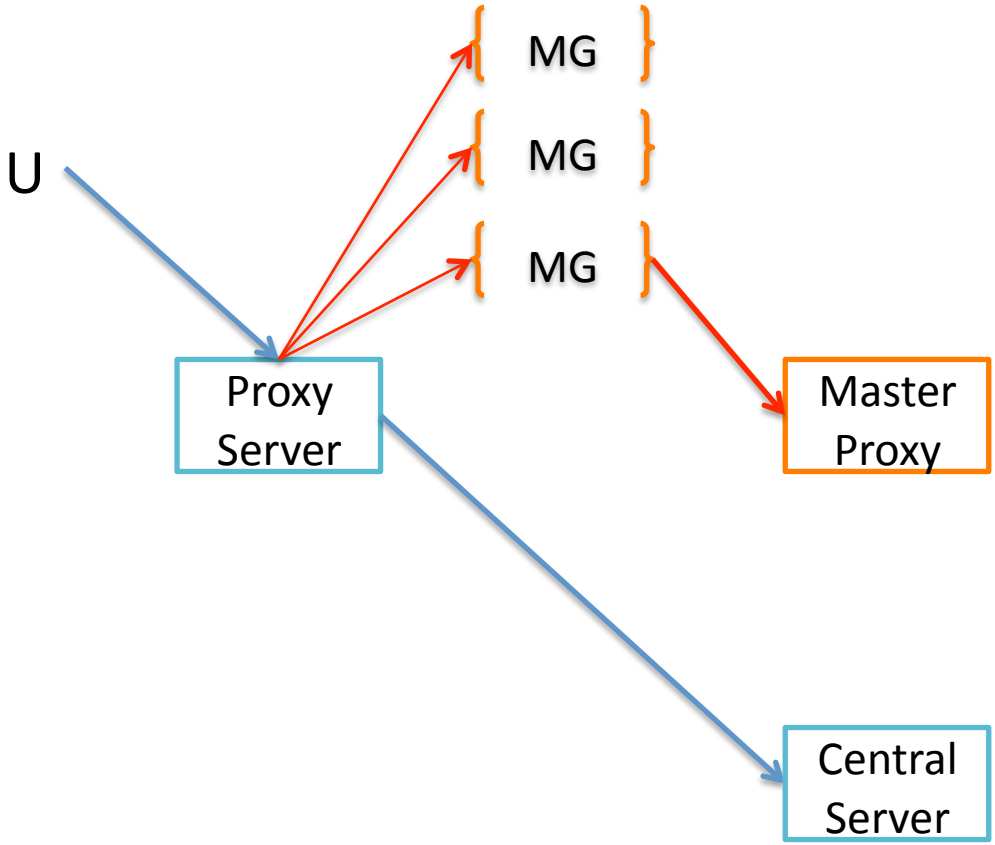
Update



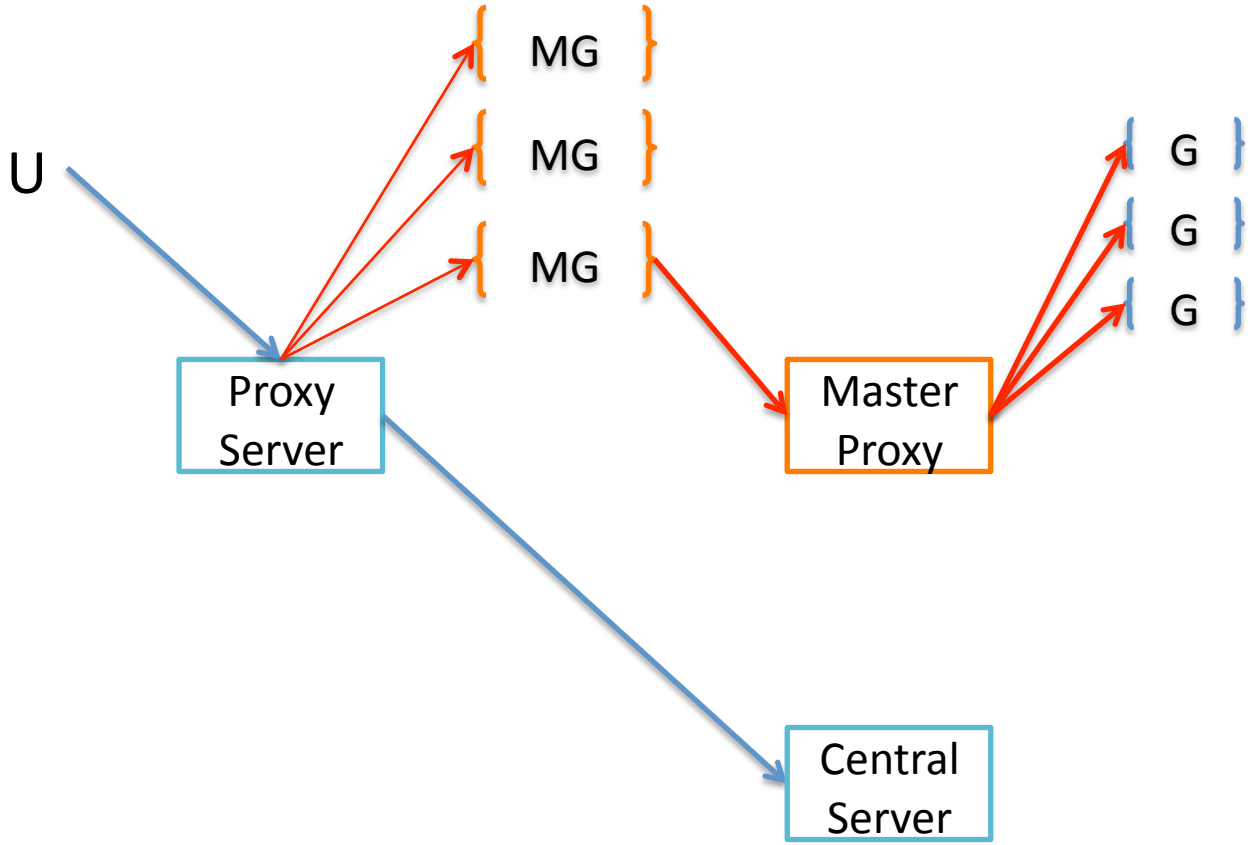
Update



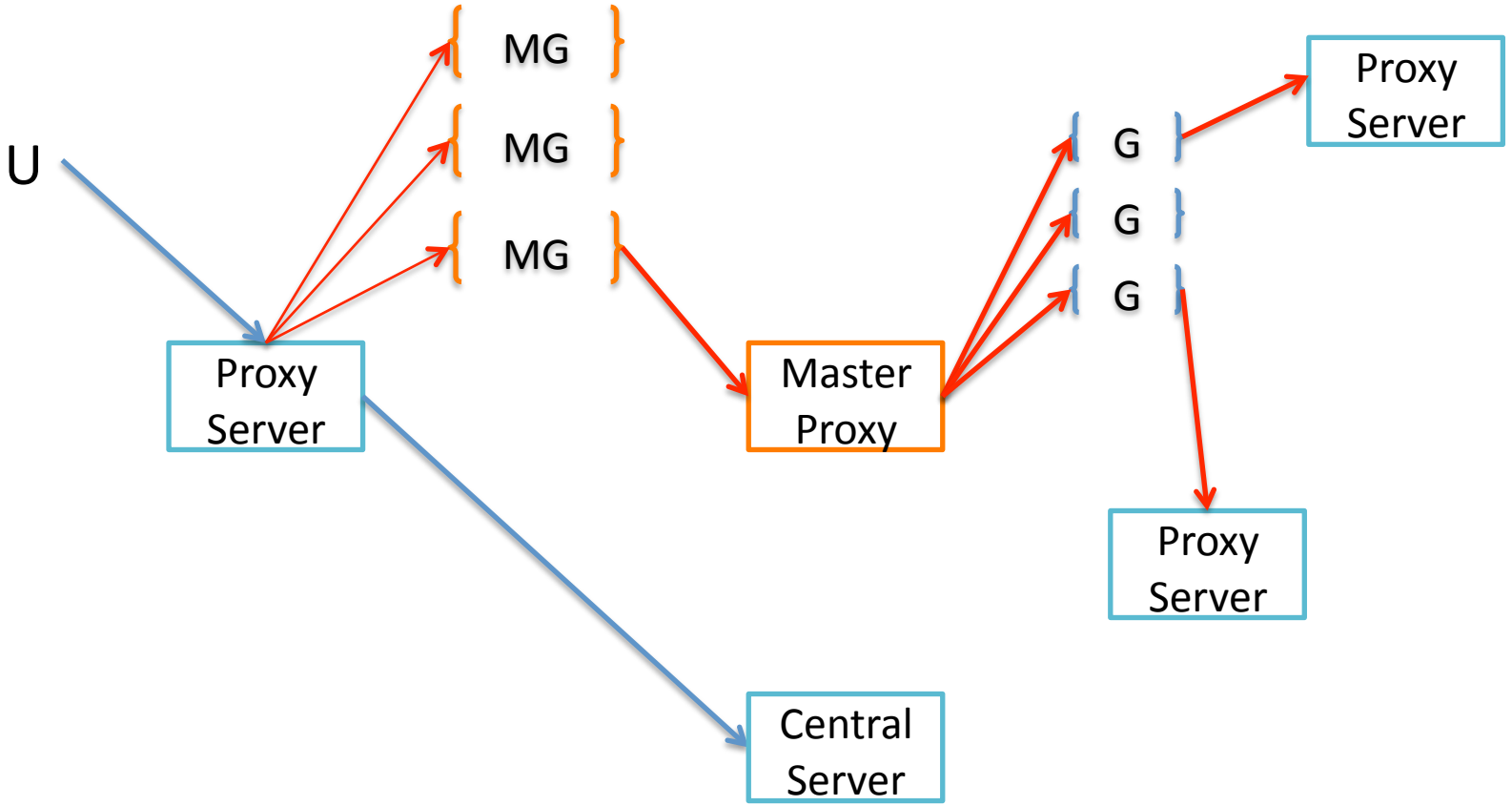
Update



Update



Update


U
MG
MG
MG
Proxy Server
Master Proxy
G
G
G
Proxy Server
Proxy Server
Central Server

Consistency Management

- Ensures that query cache remains coherent with the central DB but..
  - Guarantees full consistency only for single statement transactions.
- Requires a reliable publish/subscribe system.
- Requires serializability guarantees from central database

Implementation

- JDBC Driver
- Proxy runs Apache tomcat as static cache
- Pastry overlay as DHT (is based on PRR tree)
- Scribe for publish scribe
    - does not guarantee reliable delivery.
- Ferdinand cache map is stored in MySQL4
- Backend database is MYSQL4

Evaluation

- Performance comparison with several alternative approaches.

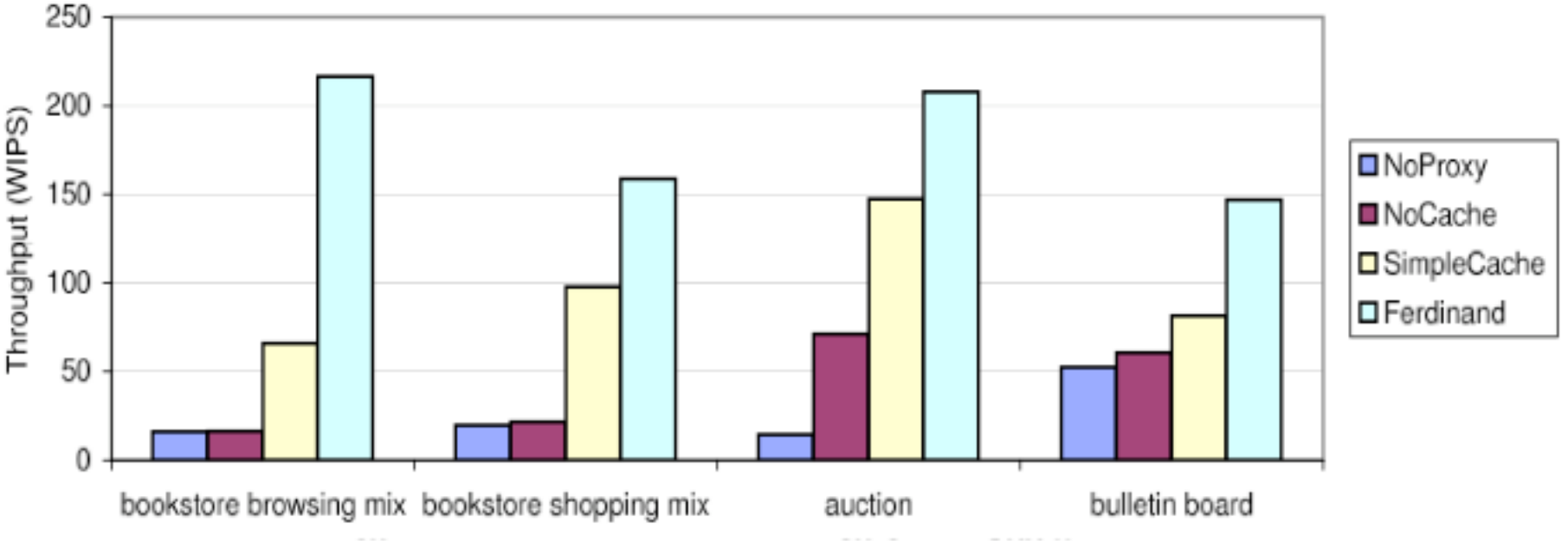
- Performance of DHT based cooperative caching in varying network latency scenarios.

- Publish/subscribe vs simple broadcast-based system

Evaluation

- Emulab testbed
- Proxy ran on 3 GHz Intel Pentium Xeon, 1GB ram, 10,000 RPM SCSI disk.
- Benchmark clients on 850 MHz client servers.
- Benchmark
    - TPC-W bookstore
    - RUBiS auction
    - RUBBos bulletin board
        - Conforms to TPC-W model of emulated browsers

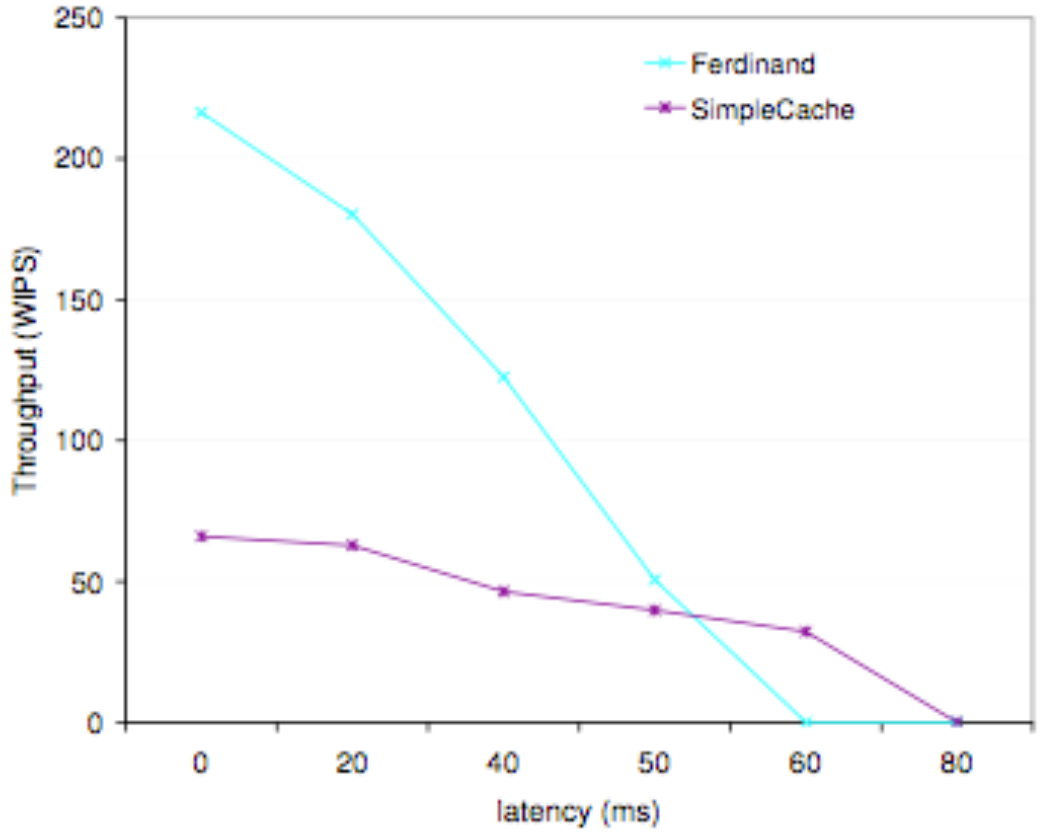
Evaluation: Comparison to other approaches



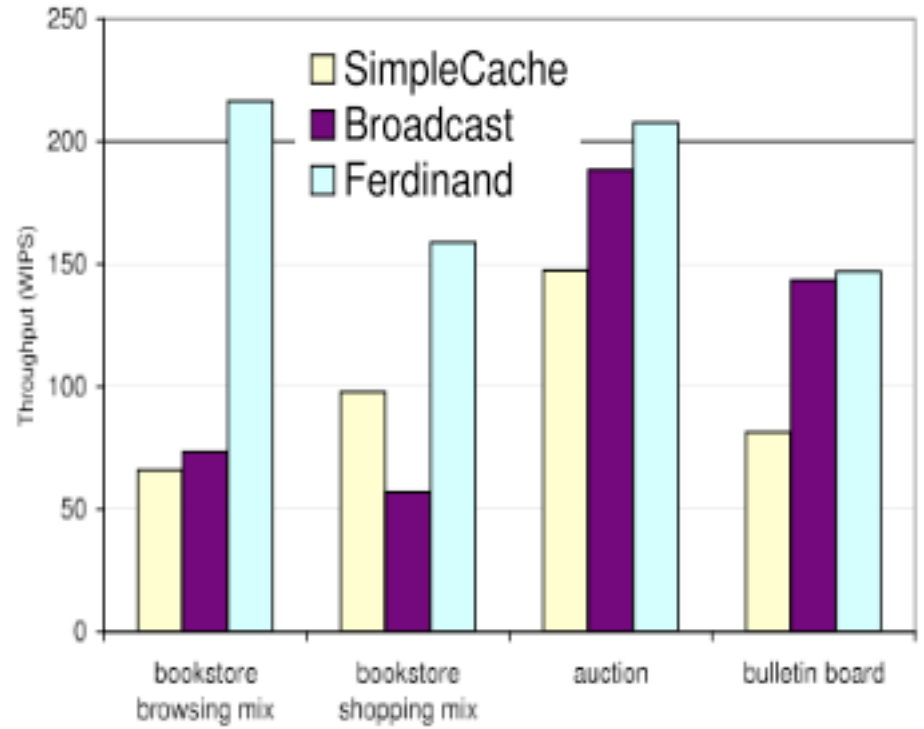
Evaluation: Cache Miss Rate

	SimpleCache	Ferdinand
bookstore browsing mix	17%	7%
bookstore shopping mix	22%	14%
auction	40%	17%
bulletin board	20%	11%

Evaluation: Throughput as a function of latency



Evaluation: Throughput compared to broadcast-based consistency



Closing Observations

- No Scalability evaluation shown.
- Not suitable for update intensive application.
- Requires offline analysis of database requests.
- Failure Scenarios need to be deal with.
- No consistency for multi statement transactions.