An Adaptive Peer-to-Peer network for Distributed Caching of OLAP Results

-by Henry Huaxin Zhang

Peer to Peer

OLAP

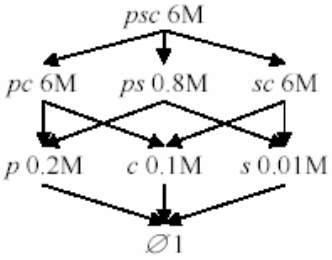
Caching

First time I see them all-in-one

Principle: Data Cube Lattice in PeerOLAP

Materialized view for OLAP query

Paths show queries can be answered



Higher level views have more information, lower level views smaller

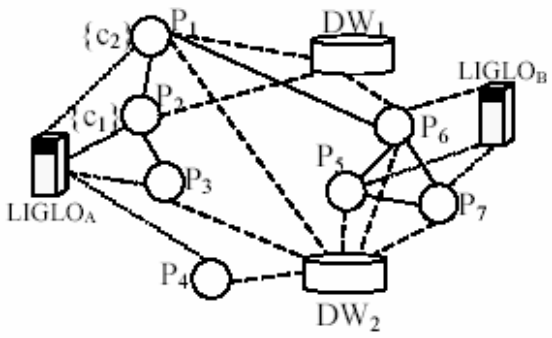
Chunked Views – better reusability, regularity, non-redundancy

Answering queries by aggregating chunks found in peers

Architecture

Dynamic P2P networks

BestPeer as Foundation



Application Layer
(Mobile Agent)

Application Layer downloaded from DW

P2P management

Cache Controller | Cache

Cache Layer is
P2P Management + Cache Management

Parameters, Policies, Options

Eager Query Processing

- Propagate request for entire C_{miss} recursively to all neighbors
- Looking for optimal path at the cost of network flooding

Lazy Query Processing

- Only propagate filtered request to most beneficial neighbor
- Avoid network messaging overhead for sub-optimal path

Trade off: Transportation cost or network messages?

Performance Evaluation (chunk-size =1M) shows EQP improves faster as

1. *Number of hops increases*
2. *Number of neighbors increase*

Parameters, Policies, Options (cont.)

Caching policy is a revision of LRU based on the ***Benefit*** of chunk

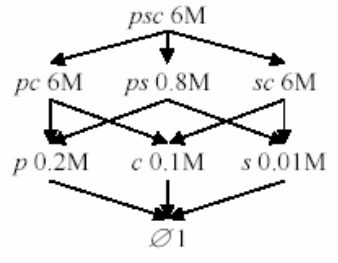
Request time + number of hops

Why not LRU/LFU?

Less frequently used chunks may be expensive to compute

Why not based on Data Cube Lattice metrics like $\frac{|D|}{n}$?

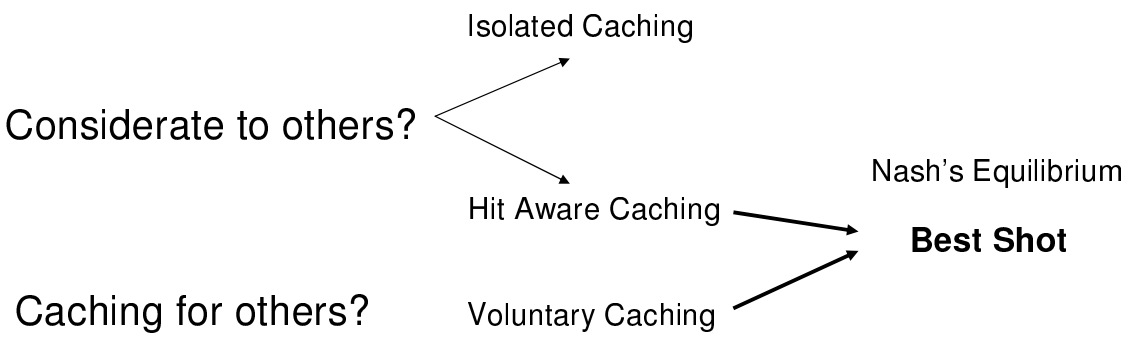
- Aggregation only in data warehouse
- We need to think of the Peer-to-Peer architecture



Parameters, Policies, Options (cont.)

Non-Semantic Caching

-- Separation of application layer from Caching layer



Parameters, Policies, Options (cont.)

Network Re-organization

Cache control level -- dynamic, not link layer, no routers

Thinking of network connection as cache problem

Why LRF? Assumptions:

- A peer contacted recently would be contacted again
- Cache chunks are of equal value in answering a question

- Infrequent / over-frequent network re-organization can hurt performance
- Finding optimal frequency is out side the scope of this paper

Clarifications

Because PeerOLAP is not distributed data warehouse …

These issues are irrelevant to PeerOLAP

- Update/insert/delete from clients
- Transactional management
- View update normally, data source inconsistency
- Data mart as alternative to cached chunks

Because PeerOLAP is a client-side approach … not proxy/server

These are minor considerations for PeerOLAP

- Query planning for efficient execution for OLAP queries
- Providing uniformed views of different data warehouses to clients

Clarifications (cont.)

These issues is/can be easily solved by PeerOLAP

- Updates inside Data Warehouse can be easily detected from the Mobile Agents downloaded from DW.
- Deletion of a peer is transparent to each peer. Adding a site is transparent to the whole system
- Lock a chunk in its cache (from being evicted) until all requests for that chunk from other peers are fulfilled

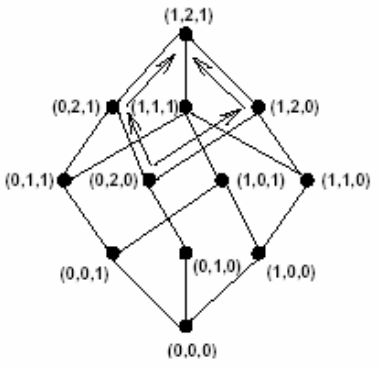
Clarifications (cont.)

This issue can NOT be easily solved by PeerOLAP

Aggregation is only allowed at the same granularity

Double Hard -- One problem nested in the other

- Find optimal path in computing missing chunks
- Finding optimal combinations of computing missing chunks from all other peers



Other concerns -- weight algorithm needs to be adjusted, but how to do it fast?

Summary

- Excellent in doing the combination
- Leave little room for improvement