Research Issues in Peer-to-Peer Data Management

- Part 2 -

Alex Coman
acoman@cs.ualberta.ca

Addressed Issues

- Designing a Super-Peer Network [1]
  - Understanding the characteristics of super-peer networks
  - Designing a language for schema mediation in a Peer Data Management System (PDMS)

Designing a Super-Peer Network [1]

- Definition:
  - a cross btw. a pure and hybrid P2P system
  - super-peer nodes act as servers for a subset of clients
  - clients pose queries through servers, but download directly from other peers
  - super-peer nodes are part of a pure P2P network

Designing a Super-Peer Network [1]

- Super-Peer network motivation
  - existence of peers with limited capabilities
  - greater responsibility assigned to capable peers
  - combines efficiency of centralized search with the autonomy, load balancing and robustness of distributed search networks

Designing a Super-Peer Network [1]

- Goal:
  - understanding the fundamental characteristics and performance trade-off
  - develop practical guidelines on designing such networks

Designing a Super-Peer Network [1]

- Contributions:
  - summarize trade-offs through “rules of thumb”
  - formulate procedure for global design
  - give guidelines for local decision making
  - introduce “K-redundancy” for improved super-peer network design
Designing a Super-Peer Network [1]

- Notions:
  - neighbors of a node: nodes with a direct open connection
  - outdegree of a node: number of neighbors
  - hop count: length of path traveled by a message
  - TTL (time to leave): how many hops a message takes
  - query reach: number of nodes that process a query

- Super-peer redundancy
  - problem: super-peer is point of failure
  - k-redundancy (replication)
    - k-nodes share the super-peer load
    - each k-node connected to all clients
    - each k-node has full index
    - each k-node connected to all cluster’s neighbors

- Evaluation model metrics:
  - load: individual and aggregate
    - incoming bandwidth
    - outgoing bandwidth
    - processing power
  - quality of results

Designing a Super-Peer Network [1]

- More about Super-peer networks
  - clients connected to a super-peer form a cluster
  - client submit query to super-peer only
  - super-peer keeps index of clients’ data
  - process incoming queries in their behalf
  - operations: query, join(login), update

- Evaluation model parameters:
  - graph type: power-law or strongly connected
  - graph size: 10k -> 30K
  - cluster size
  - avg. outdegree: 3.1 (Gnutella)
  - TTL
  - query rate
  - update rate

- Performance evaluation
  - generate an instance
  - compute expected cost of actions
    - cost of actions based on Gnutella
  - calculate cost from actions
  - repeated trial
Designing a Super-Peer Network [1]

- **Rules of thumb** (to follow while designing a super-peer P2P topology)
  - Increasing cluster size decreases aggregate load, but increases individual load
  - Super-peer redundancy is a good tradeoff
  - Maximize outdegree of super-peers
  - Minimize TTL
  - While query reach kept constant

- **Procedure for global design**
  - Starts from "rules of thumb"
  - Based on empirical evidence
  - 79% improvement in aggregate bandwidth

- **Local Decisions (guidelines)**
  - When global decisions are not possible
  - A super-peer should always accept new clients
  - A super-peer should increase its outdegree, as long as its cluster is not growing and it has enough resources to spare
  - A super-node should decrease its TTL, as long as it does not affect its reach

- **Conclusions**
  - Provide a foundation for understanding and building a good super-peer network
  - k-redundancy — decreases load

- **Observations**
  - Based on many real measurements on Gnutella
  - Global source may not be realistic
  - k=2 in all experiments
  - Node's load does not include disk (for index)
  - Combines measurements from Gnutella and OpenNap
  - Uses query model for hybrid P2P systems

- **Observations (cont.)**
  - TTL=1 in many experiments — will the results be satisfactory?
  - Minimize TTL — may lead to decreased number of results
  - Query reach may cover every node in the system -> unrealistic
  - Discovering nodes at login time -> research issues
Schema Mediation in PDMS [2]

- Data integration systems:
  - allow queries over autonomous heterogeneous systems
  - administrator defined global mediated schema
  - semantic mappings between sources and the mediated schema
- Lightweight data storage
  - simpler and cheaper tools
  - simple administration

Schema Mediation in PDMS [2]

- Data integration problems
  - comprehensive schema design
  - data sources cannot change significantly
  - concepts can be only added by administrator
  - schema evolution is heavyweight
    - difficult to extend schema
    - may brake existing queries
- Lightweight storage tools problems
  - increased functionality cost
  - if repository grows too large, it is hard to migrate to a richer tool

Schema Mediation in PDMS [2]

- **Peer Data Management System:**
  - decentralized, easily extensible data management architecture in which any user can contribute new data, schema information or even mappings between other peers’ schemas.

Schema Mediation in PDMS [2]

- Characteristics of PDMS systems
  - joining is a heavyweight operation
  - peers are likely to stay available most of the time
  - membership may be restricted
  - may provide an infrastructure for the Semantic Web

Schema Mediation in PDMS [2]

- Paper contributions:
  - focus on decentralized schema mediation and answering queries over multiple schemas
  - flexible language for mediating between peer schemas (Peer-Programming Language - PPL)
  - characterize complexity of query answering
  - describe a query reformulation algorithm
  - methods for optimizing the reformulation algorithms
  - initial set of experiments
**Schema Mediation in PDMS [2]**

- **Formalisms for schema mediation borrowed from data integration**
  - *global-as-view approach (GAV)* – mediated schema defined as set of views over data sources
  - *local-as-view approach (LAV)* – contents of data sources are described as views over the mediated schema

**Schema Mediation in PDMS [2]**

- **Assumptions**
  - peers employ relational data model
  - only select-project-join queries
  - conjunctive queries with no comparisons (<, >, ...
  - each peer defines its own relational peer schema
  - relations and attribute names unique to each peer
  - some peers are just logical mediators
  - names of stored relations are distinct than peer relations
  - a peer will obtain all possible data from anywhere in the PDMS

**Schema Mediation in PDMS [2]**

- **Mapping language**
  - types of mappings considered
    - describing data within the stored relations with respect to one or more peer relations
    - mappings between peers’ schema
  - LAV formalism: used when there are many data sources related to a mediated schema
  - PPL tries to use and extend both GAV and LAV to a general framework of interrelated peers

**Schema Mediation in PDMS [2]**

- **Defined mappings:**
  - storage description: each peer contains a set of storage descriptions that specify which data it stores by relating its stored relations to one or more peer relations
  - peer mappings: semantic glue between schemas of different peers
    - inclusion or equality mappings (e.g. $Q_1(A_1) \subseteq Q_2(A_2)$)
    - definitional mappings – datalog rules

**Schema Mediation in PDMS [2]**

- **Defining a PDMS system:**
  - set of peers
  - set of peer schemas and a mapping function from peers to schemas
  - set of stored relations at each peer
  - set of peer mappings
  - set of storage descriptions

**Complexity of query answering**

**Def:** Let $Q$ be a query over the schema of peer $A$ in a PDMS $N$, and let $D$ be an instance of the stored relations of $N$. A tuple $a$ is **certain answer** to $Q$ if $a$ is in $Q(D)$ for every data instance that is consistent with $N$ and $D$.

**Results:**
- finding all certain answers is undecidable in general case
- if $N$ includes only inclusion peer and storage descriptions and the peer mappings are acyclic, the query can be answered in polynomial time
Schema Mediation in PDMS [2]

Query Reformulation Algorithm
- **Input**: set of peer mappings, storage descriptions, query Q
- **Output**: query Q' that only refers to stored relations at the peers
  - may be very deep
  - large branching factor
- query evaluation will only produce certain answers

Schema Mediation in PDMS [2]

Query Optimizations
- memoization of nodes
- detection of dead ends and useless paths
- employ priority scheme
- first reformulations must be generated quickly (in the context where many reformulations are required)

Schema Mediation in PDMS [2]

Experiments
- lack of existing PDMS to test on
- workload generator used
- experiments in initial phase

Experimental conclusions
- finding rewritings form the rule-goal tree is bottleneck
- diameter of the PDMS is the main factor determining the size of the rule-goal tree

Observations:
- many new notions, some assumed to be known
- assumes semantic mapping is properly done
- assumes schema mediation is done
- mostly hierarchical structure of the PDMS system
- small PDMS systems (up to 96 peers in experiments)
- some peers act just as logical mediators (no data)
- PDMS is less dynamical than P2P systems
- efficient query answering only for simplified PDMSs
- query optimizations are vague
- few experiments

References


Thank you!