What is Peer-to-Peer?

- [1] - a distributed system in which participants rely on one another for service
- [2,3] - distributed computing nodes of equal roles or capabilities exchange information and services directly with each other
- [4] - a large number of nodes are pooled together to share their resources information and services

P2P – What are we looking at?

- Point of view for P2P study
  - Data organization view [1][4]
  - Information retrieval view [2][3]
- Types of P2P studied
  - Pure P2P [3][4]
  - Hybrid P2P [2]
  - General P2P [1]
What can databases do for P2P?

**Disadvantages:**
- Weak application semantics (Gnutella)
- Immense technical challenges
- Difficult to predict system's resources
- Limitations on data consistency and availability

**Issues:**
- Naïve placement of data
- Optimizations must be done in completely distributed manner
- Limitations on data consistency and availability

**Fundamental problem:**
- Placement and retrieval of data

**Data Placement Problem**
- To distribute data and work so the full query workload is answered with lowest cost under existing resources and bandwidth constraints

**Differences from DDBMS**
- No centralized schema
- No central administration

**Design choices for data placement in P2P**
- Scope of decision making: optimization scale
- Extent of knowledge sharing: amount of knowledge available at optimization time
  - Central directory (Napster)
  - Full catalog replication
  - Hierarchical organization (DNS, LDAP)
- Heterogeneity of information sources
  - Single schema
  - Limited number of schemas
  - Dynamicity of participants

**Design choices (cont)**
- Data granularity
  - Atomic granularity level
  - Hierarchical granularity level
  - Value based granularity
- Degrees of replication
  - At will
  - Only sparingly
  - Not at all
- Freshness and update consistency
  - Server pushed invalidation
  - Client initiated invalidation
  - Timeout/expiration based protocol
What can databases do for P2P?

- Piazza system
  - Hierarchy of spheres of cooperation (resource pooling and cooperative decisions)
  - Focus on dynamic data placement (data, query workload, peer membership)
    - Query optimization
      - Propagating information about views
      - Consolidating query evaluation
    - Data freshness
      - Use expiration times

Comparing hybrid P2P systems

- Advantages:
  - Large pool of information and system resources
  - Better network bandwidth utilization

- Disadvantages:
  - Network flooding (Gnutella)
  - Limited searching on single server (Napster)

Comparing hybrid P2P systems

- Issues:
  - Evaluating P2P architectures
- Contributions:
  - Several architectures for hybrid P2P servers
  - Probabilistic model for user queries and result sizes
  - Model for evaluating the performance of P2P architectures
  - Results from a real P2P application

Comparing hybrid P2P systems

- Server architectures - Chained (OpenNap)
  - Uses “local inverted lists” – subset indexing at local server level
  - Linear chain of server
  - Search local, then forward
  - Advantage:
    - fast logins
  - Disadvantage:
    - expensive if many servers are needed for query answering

Comparing hybrid P2P systems

- Server architectures - Full replication
  - uses NNTP approach – full replication
  - Complete index on all servers
  - Advantage:
    - Any server can answer query
  - Disadvantage:
    - Message overhead at login time
### Comparing hybrid P2P systems

**Server architectures:** **Hash**
- Uses “global inverted lists” – lexicographical partitioning across nodes
- Server keep complete list for words subset

**Advantage:**
- Fast query answers

**Disadvantage:**
- Message overhead at login time
- Increased bandwidth usage at query time

**Server architectures:** **Unchained** (Napster)
- Independent servers that do not communicate with each other

**Advantage:**
- Fast answers
- Linear scaling with increasing number of servers

**Disadvantage:**
- Limited coverage (partial search)

**Login strategies:**
- **Batch:**
  - Only libraries of connected users are in the index
  - Small index
  - Expensive updates
- **Incremental:**
  - User files are kept in index all the time
  - Update at login time
  - May tie user to a server
  - Filtering required during query processing

**Query Model**
- To estimate
  - The number of query results
  - The number of servers needed to answer a query
- Probability density functions
  - \( f \) – what files users like to store
  - \( g \) – what queries users like to submit
- Exponential distributions are used

\[
Q(n) = 1 - \frac{1}{n} \sum_{i=0}^{n-1} g(i) \left( \frac{n}{m} \right)^i (1 - \left( \frac{n}{m} \right)^n) \]

Probability that \( R \) or more results are found in a collection of \( n \) or fewer files

**Performance Model**
- System environment and resources
- CPU consumption
- Network consumption
- Overall performance
- Cost formulas based on studying the actions of a typical implementation and counting

**Evaluation**
- Maximum number of users supported by server
- Response time not considered
Comparing hybrid P2P systems

- Conclusions
  - Incremental strategies are better
  - Batch Chained better than Batch Unchained (Napster) at low cost
  - Query Login Ratio – very important parameter
  - Many parameters involved – hard to evaluate

Efficient search in P2P networks [3]

- P2P Definition – as before
  - System with focus on availability
  - Loose systems – persistence and availability are not guaranteed or necessary
- Advantages:
  - Self organization
  - Load balancing
  - Adaptation
  - Fault tolerance

Efficient search in P2P networks

- Problem (loose P2P systems): Inefficient search techniques
  - Generate too much load
  - Bad user experience
- Solution: reduce then number of nodes that process a query
  - Optimize bandwidth and processing

Efficient search in P2P networks

- Existing techniques:
  - Breadth first traversal (BFS) with depth limit D (Gnutella)
    - Large number of results
    - Waste of resources
  - Depth first traversal (DFT) with depth limit D (Freenet)
    - Minimal costs
    - Poor response time

Efficient search in P2P networks

- Broadcast policies:
  - Iterative Deepening – minimize resource usage
    - Multiple BFS with successively larger depth limits
    - Stop if query satisfied or max depth D reached
  - Directed BFS – minimize response time
    - BFS with neighbors selected by quality of results
      - Heuristics for node selection
        - Highest number of result in the past
        - Neighbors with lowest number of hops
        - Neighbors that has forwarded the largest number of messages
        - Neighbor with shortest message queue
  - Local indices
    - Index of all nodes within r hops
    - Each node process the data on behalf of nodes within r hops
    - Policy to specify what depths to be searched
    - Extra processing when nodes join, leaves or updates its data
Efficient search in P2P networks

• Conclusions:
  – Performance trade-off

PeerDB: A P2P based system… [4]

• P2P definition: a large number of nodes are pooled together to share their resources, information and services

• Existing systems:
  – Only file level sharing
  • Lack data management capabilities
  • Lack content based search
  – Limited in flexibility and extensibility
  – A node’s peers are statically defined

PeerDB: A P2P based system…

• P2P systems
  – Nodes join and leave network anytime
  – No predetermined global schema
  – Query based on keywords
  – Nodes may not contain the complete data (query answers)
  – Content location is by neighbor nodes

PeerDB: A P2P based system…

• DDBS
  – Nodes are added and removed in controlled manner
  – Stable nodes with shared schema
  – Complete set of answers is expected
  – The exact location to direct the query is known

PeerDB: A P2P based system…

• BestPeer
  – Platform for P2P applications
  – Integrates mobile agents and P2P technologies
  – Share partial file content
  – Share computational power
  – Dynamic reconfiguration of neighbor peers
  – Uses Location-Independent Global Names Lookup Server
  – Secure access to node’s resources

PeerDB: A P2P based system…

• Issue: propose a P2P system with data management capabilities
• PeerDB
  – Extends BestPeer
  – Node’s data managed by a DBMS
  – Data is shared without global schema
  – Query processing assisted by mobile agents
  – Nodes reconfigure themselves
PeerDB: A P2P based system...

- Architecture of PeerDB node
  - DBMS (MySQL)
  - Database agent system (DBAgent)
  - Cache manager
  - User interface

- Sharing data without shared schema
  - Meta-data for each relation and attribute
    - Local dictionary – for all relations
    - Export dictionary – for shared relations
  - Using meta-data, relations and attributes are matched, similarity degree computed
  - User is deciding if match is correct

- Query processing
  - Agent assisted
  - User selects relevant relations
  - Query types
    - Local – where it was initiated
      - Relation matching agents are sent out
      - User select desired relations
      - Data retrieval agent sent out to relevant nodes
    - Remote
      - Match based on export dictionary
      - Clone and send relation matching agents
      - Answer to data retrieval agent

- Nodes reconfiguration
  - Favors node that has provided answers most recently
  - Uses the stack distance notion

- Cache management
  - Expiration time policy
  - LRU replacement policy
  - Keeps track of multiple copied by a unique node identifier

- Conclusion
  - Promising system
  - Not much faster than Client-Server architecture as time to answer
  - Faster than Client-Server in terms of speed to obtain subset of results
  - Agent-Based Protocol performs better that Message-Based Protocol
  - Neighbors reconfiguration improves results over static neighbors

Conclusions

- Loose P2P definition
  - Includes many possible architectures
- Many research direction
  - Add database capabilities
  - Improve existing using database techniques
  - Propose new systems
- Promising paradigm
References


Thank you!