Range queries in trie-structured overlays

Anwitaman Datta, Manfred Hauswirth
Renault John, Roman Schmidt, Karl Aberer

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Funded by

Swiss National Centres of Competence in Research

Mobile Information & Communication Systems

EC FP6, IST priority “Complex System Research”
Contract no. 001935 (FET-IP)

Ever-growing global scale-free networks,
their provisioning, repair and unique functions.
Outline

• Problem Definition
• Existing Approaches
• P-Grid overview
• Our approach to range queries
  – Algorithms
  – Complexity analysis
• Experiments
• Conclusions
Problem Definition

• A range query is supposed to return all keys in an interval defined by 2 bounds

• Range queries are non-trivial search predicates for structured overlay networks

• Problems to address:
  – Scalability
    • Number of messages
    • Number of involved peers
    • Latency of queries
Existing Approaches (ring-based)

• Examples: Chord, Pastry, Bamboo, etc.
  – Uniform hashing functions
    • uniform distribution but no clustering (destroys order)
  – Good storage load balancing and efficient exact queries
  – Superimpose a prefix hash tree (PHT)
  – Less efficient because requires separate exact queries

Range query: C - H

O(log n)
Existing Approaches (CAN-based)

- Hash ranges on top of CAN
  - Exact queries are not supported/inefficient
- Maintenance of separate hash tables for keys and ranges
  - No reuse of peer resources
- Leads to bad fragmentation
  - Poor storage-load balancing or inefficient access
- CAN's search efficiency guarantees only hold for uniform space partitioning
  - Storage load is arbitrarily distributed
  - Queries are also non-uniformly distributed
Classification of existing approaches

- Uniform hashing of individual data keys to generate DHT/overlay keys
  - For example Chord, Pastry, etc.
  - Hash function balances load but destroys ordering of keys
  - Require a superimposed PHT
  - Higher maintenance and less efficient
- Hashing and caching of range predicates to generate DHT/overlay keys
  - For example approaches based on CAN
  - Hash ranges instead of keys
  - Separate hash tables for exact and range queries
- Preserving the data key ordering while generating the overlay keys
  - For example P-Grid, Mercury, Skip Graphs, etc.
  - Order-preserving hash function
  - Requires additional load-balancing methods
  - One hash table for exact and range queries
P-Grid overview

- Peers are organized in a binary trie structure
  - One node for every common prefix
  - Trie is only virtual (exists only via routing tables)
  - All nodes remain at the leaf-level (no hierarchy)
  - Designed with respect to efficient range queries [Aberer01]
- Multiple peers per key space partition
- Multiple routing entries (random choice)
- Order-preserving hash-function
  - $s_1$ is-prefix-of $s_2$ => $h(s_1)$ is-prefix-of $h(s_2)$
  - Clustering but explicit load-balancing necessary
- Logarithmic search complexity
- Even for skewed data distributions
P-Grid routing

- Keys resolved by longest prefix matching
  - Insures logarithmic search cost for skewed trees

```
query for '100'
```

```
00*  
A  1*: C, D
  01*: B
Stores data with key prefix 00

F  1*: E
Stores data with key prefix 00

01*  
B  1*: C, D
  00*: F
Stores data with key prefix 01

10*  
C  0*: A, B
  11*: E
Stores data with key prefix 10

D  0*: A, F
  11*: E
Stores data with key prefix 10

11*  
E  0*: B, F
  10*: D
Stores data with key prefix 11
```
Min-max traversal algorithm

Result set: C, D, E
Shower algorithm

Result set: C, D, E
Complexity Analysis

• Variables
  – M ... data items per key space partition
  – D ... data items in the range
  – N ... number of key space partitions
  – \( N_R \) ... number of key space partitions in the range (expected \( D/M \))
  – Depth ... maximum path length of any partition in the range

• Min-max traversal algorithm
  – Search cost and latency is \( O(\log_2 N) + N_R - 1 \)
  – Neighbor detection overhead is \( N_R O(\log_2 N) \)

• Shower algorithm
  – Search cost lower bounded by \( O(x) + N_R - 1 \) and upper bounded by \( O(x) + \min(2O(N_R), 2^{Depth-x}) \)
  – Latency between \( O(x) \) and \( O(x) + O(\text{Depth-x}) \)
  – Expected value of \( x \) is \( 0.5\log(NM/D) \)
Experiments on PlanetLab

- Implemented in Java (http://www.p-grid.org/)
- 250 PlanetLab nodes
  - Each node selects 10 items from data set
  - In total 2500 unique data items
- 2 data sets (uniform, Pareto)
- Replication factor of 5
  - On average 5 peers per key space partition
  - Peers are responsible for 50 to 100 data items
  - In total 18750 data items in the system
- Experiments designed such that 50, 100, 150, 200, 400, 800 items are returned
- One query per peer for each range for each distribution and for each algorithm
  - In total 6000 queries
  - 250 values per data point in the following figures

![Uniform data distribution](image1.png)

![Pareto data distribution](image2.png)
Message latency (hops)

- Maximum number of messages required to hit each sub-partition (whole queried interval is covered)
  - At least 3 hops to reach a responsible peer
  - Min-max hop counts increase linearly (sequential traversal)
  - Shower hop counts increase logarithmically (start at higher level)
Message cost

- Number of sent messages and number of peers involved in query forwarding/answering

- Number of hops and messages is equal for the min-max algorithm

- Both algorithms scale well for both distributions

- Shower algorithm requires more messages and involves more peers
Query latency (time)

- Latency to receive all data items of the queried data set

- Logarithmic increase for the shower algorithm
- Linear increase for the min-max algorithm
- First result items are received after 10 sec. (for both algorithms)
Result completeness

- Percentage of received items with respect to actual available items

- Independent of range size and data distribution

- Local, smaller scale experiments proved the correctness

- Observed problems
  - “Lost” messages
  - PlanetLab nodes crash

- Possible solutions
  - Increasing replication
  - Protocol adoptions (ACK messages, resending results)
Conclusions

• Range queries on top of a trie-structured DHT
• Theoretical analysis shows the efficiency
  – Independent of queried range sizes
  – Depends only on result set size
• Works in a real network environment
Thank you!

Questions?

http://www.p-grid.org/