Set Implementation over DHT Systems

Nuno Lopes   Carlos Baquero

Departamento de Informática

Universidade do Minho

Simpósio Doutoral Departamento Informática, 2005

Contents

Peer-to-Peer System Characterization

- Very large number of hosts.
- Deployment over wide-area networks.
- Dynamic host connectivity.
- Decentralized architecture.
- Highly concurrent access.

Potentially very large data storage system. . .

Distributed Hash Table Algorithms

- Scalable on very large systems.
- Efficient object location given a key.
- Common Key-Based Routing API: $route(key, message)$.

However,

- Unable to search for objects.

Search Functionality on DHTs

- Use DHT as scalable storage layer.
- Inverted Index model to allow (document) searching: $keyword \hookrightarrow document\_reference_{set}$.

Efficient set implementation over DHTs required...

Contents

Set Algorithm Implementation

- B-link tree:
  - logarithmic cost for common ops;
  - best performance for highly concurrent access.
- Block contents is reached via block pointers (references).
- Storing tree blocks on a DHT requires an unique block key generation scheme.

Client-side Interface

- Item insertion, removal and search:
  - same tree traversal pattern used, creating contention on top level blocks.
  - Relevant information is held only at leaf level.

  $\Rightarrow$ Cache non-leaf blocks at client hosts.
- Complete item retrieval.
  - sequential leaf block access.
- Set union and intersection.

Internal Tree Management Operations

- Block splitting:
  - initiated locally at overloaded block,
  - involves 3 blocks: initiator, new sibling and parent.
- Inserting child reference.
- Joining blocks and removal child references.

Contents

Using Block Replication for Fault Tolerance

- Hosts have very small session uptime and create high level of churn.
- Replication offered by DHTs work in a “best-effort” way.
- Not only that, but *route* function is also a “best-effort”.
- Our (generic) data model requires strong consistency. . .

What replication technique can be used in a P2P environment to support our requirements?

Pessimistic Replication Proposals

- Some proposals enable generic functionality using strong data consistency.
- Approaches rely on atomic multicast/consensus protocols.
- Deployable on low latency (local-area) networks.
- Not scalable for very large wide-area (P2P) systems.

Can expensive communication protocols be avoided?

Optimistic Replication

- Relaxing data consistency allows simpler communication protocols to be used.
- Replica divergence and data loss must be considered.
- Reconciliation algorithms required to bring data into a consistent state.
- However, our (generic) algorithm cannot reconciliate data (under some circumstances). . .

⇒ Solution may be using operation semantics.

Back to the Inverted Index Again

Suggestions to make it feasible:

- Restricted functionality: insertions only with re-announcing and timeout schemes.
- Define inner tree block reconciliation algorithm, reconstructing tree from scattered pieces, possibly incomplete.
- Caching index blocks at (leaf block) hosts, instead of replicating them.

Contents

Current Status

- We defined a basic distributed set structure over DHTs.
- It can be used as generic data structure as long as strong consistency primitives are available.
- On very dynamic networks, pessimistic approaches are not adequate, we must rely on optimistic semantic-aware approaches.

Future Directions

- Enhance the algorithm to support optimistic scenario.
- Implement generic data structures over DHTs and study their feasibility.

The End