CIS 700/005
Networking Meets Databases

Boon Thau Loo
Spring 2007
Lecture 4

Note: Several slides are courtesy of Ryan Huebsch (http://www.huebsch.org)

Announcements

- Paper summaries due at noon today.
- Office hours: Wed 3-4 pm (605 Levine)
  - Project proposal: due Feb 12.
- Student presenter:
  - Varun will present.

VLDB punch line: What is Very Large? Depends on Who You Are

<table>
<thead>
<tr>
<th>Single Site Clusters</th>
<th>Distributed 10’s – 100’s</th>
<th>Internet Scale 1000’s – Millions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Community</td>
<td>Network Community</td>
<td></td>
</tr>
<tr>
<td>Challenges</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▫ How to run database style queries at Internet scale?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▫ Can DB concepts influence the next Internet architecture?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Four Design Principles (I)

- Relaxed Consistency
  - ACID transactions severely limits the scalability and availability of distributed databases
  - Best-effort results
- Organic Scaling
  - Applications may start small, without a priori knowledge of size

Four Design Principles (II)

- Natural habitat
  - No CREATE TABLE/INSERT
  - No “publish to web server”
  - Wrappers or gateways allow the information to be accessed where it is created
- Standard Schemas via Grassroots software
  - Data is produced by widespread software providing a de-facto schema to utilize

Application Space

- Key properties
  - Data is naturally distributed
  - Centralized collection undesirable (legal, social, performance, etc.)
  - Homogenous schemas
  - Data is more useful when viewed as a whole
- This is the design space PIER has chosen to investigate
  - Mostly systems/algorithms challenges
  - As opposed to …
    - Enterprise Information Integration
    - Semantic Web
    - Data semantics & cleaning challenges
Who Needs Internet Scale?

Example 1: Filenames

- Simple ubiquitous schemas:
  - Filenames, Sizes, ID3 tags
  - Born from early P2P systems such as Napster, Gnutella, AudioGalaxy, etc.
  - Content is shared by “normal” non-expert users… home users
  - Systems were built by a few individuals ‘in their garages’ ➔ Low barrier to entry

Example 2: Network Traces

- Schemas are mostly standardized:
  - IP, SMTP, HTTP, SNMP log formats
  - Network administrators are looking for patterns within their site AND with other sites:
    - DoS attacks cross administrative boundaries
    - Tracking virus/worm infections
    - Timeliness is very helpful
  - Might surprise you how useful it is:
    - Network bandwidth on PlanetLab (world-wide distributed research test bed) is mostly filled with people monitoring the network status

Review: Distributed Hash Tables

- What is a DHT?
  - Take an abstract ID space, and partition among a changing set of computers (nodes)
  - Given a message with an ID, route the message to the computer currently responsible for that ID
  - Can store messages at the nodes
    - This is like a “distributed hash table”
    - Provides a put()/get() API
  - Cheap maintenance when nodes come and go
    - Logarithmic network state and hop count

PIER’s Three Uses for DHTs

- Single elegant mechanism with many uses:
  - Search: Index
    - Like a hash index
  - Partitioning: Value (key)-based routing
    - Like Gamma/Volcano
  - Routing: Network routing for QP messages
    - Query dissemination
    - Bloom filters
    - Hierarchical QP operators (aggregation, join, etc)
  - Not clear there’s another substrate that supports all these uses
Enhancements

- Additional API
  - `localScan` (namespace) – retrieve all data items stored in local storage for a particular namespace
  - `newData` (namespace) – receive a callback when new data is inserted into the local store for the namespace

Recall: Parallel Hash Join

- Given two tables A and B partitioned across N machines
  - A(@aid,jid), B(@bid,joinKey)
- How can we join them?
  - Hash-based re-partitioning based on joinKey

Join Algorithms (R Join S)

- Symmetric Hash Join
  - `localScan` on tables R & S. Rehash tuples into DHT using the join attributes.
  - Nodes receiving re-partitioned R and S tuples perform joins locally and forwards results to requestor.
- Fetch Matches
  - If there is an index on the join attribute(s) for one table (say R), use `localScan` for other table (say S) and then issue a lookup for R matches.

Join

For each of my attackers, how many sites did they attack, and how many packets were involved?

- `SELECT F.sourceIP, COUNT(DISTINCT p.*), COUNT(DISTINCT p.destIP)`
  `FROM firewalls F, packets P`
  `WHERE F.sourceIP = P.sourceIP`
  `AND F.destIP = <myIP>`
  `GROUP BY P.sourceIP`
- Symmetric Hash Join
  - Everybody routes their F and P records to hash(sourceIP)
  - Forms a tree per sourceIP, can combine tuples in each tree independently
  - Automatically parallelizes the join algorithm
  - No notion of parallelism in the code; falls out the DHT
Recall: Join Optimizations

- **Sailors** (SID, sname, rating, age)
- **Reserves** (RID, SID, boatID)
- Semi-joins
  - Let’s just ship Sailors.SID from London to Paris
  - Find all matches for Reserves at Paris.
  - Ship matching Reserves back to London to complete join
- Bloom-joins
  - Same idea, but use a bit-vector (bloom filter)
  - Some false positives

Symmetric Semi Join (SSJ)

- Both R and S are projected to save bandwidth
- The complete R and S tuples are fetched in parallel to improve latency

Query Execution in Dynamic Environments

- No major change in logic for QP!
- The state of query execution is stored in the network
- When network reorganizes,
  - Existing data repartitioned among existing and new nodes
  - New temporary data generated by query execution is automatically re-routed to the proper node
  - New nodes will resume any ongoing queries

Aggregation in a DHT

- **SELECT COUNT(*)**
  - FROM firewalls
- One common approach:
  - Everybody routes their firewall records to a particular “collector”
    - This forms a tree
    - Along the way, count up totals
    - At root, form final result
- Note: the shapes of these trees depend on the DHT topology!
- Can reason about comm costs, sensitivity to failure, influence of malefactors, etc.

Some Fun Stuff with PIER

- PIER+Gnutella Hybrid Search
- Distributed Web/Gnutella Crawler

Hybrid Peer Implementation
**PlanetLab Deployment**

1. Gnutella Leaf
2. Gnutella Ultrapeer
3. Hybrid Ultrapeer
   - Gnutella links
   - PIER links

**Distributed Web Crawler with PIER**

- P2P users donate excess bandwidth and computation resources to crawl the web.
- Organized using Distributed Hash tables (DHTs)
- DHT and Query Processor agnostic crawler:
  - Designed to work over any DHT
  - Crawler can be expressed as declarative recursive queries
  - Easy for user customization.
  - Queries can be executed over PIER, a DHT-based relational P2P Query Processor

**Gnutella Network Crawler**

- Gnutella Netwrok Crawler
  - Popular open-source file-sharing network
  - ~450,000 users today
  - Ultrapeer-based Topology
  - Queries flooded among ultrapeers
  - Leaf nodes shielded from query traffic
  - Based on multiple crawlers from 30 vantage points on PlanetLab

**Status of PIER today**

- At one time, ran on 24x7 on PlanetLab continuously for more than a year.
- Written in Java
- First to use Bamboo as DHT.
- Available for download ([http://pier.cs.berkeley.edu](http://pier.cs.berkeley.edu))
- Ryan Huebsch’s Ph.D. dissertation ([http://www.rinera.com](http://www.rinera.com))
- Some potential class projects:
  - Applications, applications, applications
  - Multi-query optimizations
  - Stream semantics to Internet-scale Query Processing
  - Adaptive query optimizations (Eddy Operator)
  - Sensor networks (Query processing in GHT)
  - Data integration to handle heterogeneous schemas

**PIER Lighthouse**

- GUI (called Lighthouse) for query entry and result display
  - Designed primarily for application developers
  - User inputs an optimized physical query plan
Other Internet-Scale Query Engines

- DHT-based
  - SDIMS (next week)
- Non-DHT based
  - Astrolabe (Cornell)
  - P2

Motivation of Knowledge Plane

- Revisit the “End-to-End principal”:
  - Add something to a network layer only when absolutely required.
  - What is good and bad about the “end-to-end” principal?
  - Today’s Internet has problems:
    - Network carries data w/o knowing what data is, or its purpose.
    - Intelligence @ edges, not in core
    - Simple, transparent network leads to user frustration when fails occur
    - High management overhead
    - Manual configuration, diagnosis, and design

Proposed Objective

- “Build a fundamentally different sort of network that can assemble itself given high level instructions, reassemble itself as requirements change, automatically discover when something goes wrong, and automatically fix a detected problem or explain why it cannot do so”

Ideals

- Self-sustaining network:
  - Less low-level human involvement
  - Adapt to new traffic patterns
  - Allow the network to make low-level decisions when high-level goals are specified
  - More effective interaction with end users and administrators

Where is the Knowledge Plane?
Challenges

* Lots and lots and lots of questions:
  * How to represent and utilize knowledge?
  * How to scale?
  * How to route knowledge?
  * How to deal with malicious or untrustworthy incentives?

* Research in the past 3 years…
  * "Information Planes":
    * A service or service component that efficiently delivers timely and relevant data about the state of the system to all the dispersed components of the system.
  * RAD Lab: [http://radlab.cs.berkeley.edu/wiki/RAD_Lab](http://radlab.cs.berkeley.edu/wiki/RAD_Lab)
    * Reliable Adaptive Distributed systems Laboratory
    * Sponsored by Sun, Google and Microsoft

Next lecture

* Public Health for the Internet (PHI):
  * CIDR ’07 paper on our reading list (optional)
  * [http://openphi.net/](http://openphi.net/)

* Next Tuesday:
  * A Scalable Distributed Information Management System (SIGCOMM ’04)