Distributed Data-Intensive Systems

CSCE 438
Texas A&M University
Department of Computer Science and Engineering
Instructor: Prof. James Caverlee

Peer-to-Peer Systems
January 28, 2014
Chord Circle

N56
N51
N48
N42
K38
N38
N32
K24, K30
N21
N8
K10
N14
N1
K54

m=6
Rule

- Consider nodes X, Y such that Y follows X clockwise
- Node Y is responsible for keys k such that H(k) in (H(X), H(Y))

stores K55, K56, ... K3

use hashed values...
e.g., N54 is node whose id hashes to 54.
Succ, pred links

N1.pred

N1.succ

m=6
Code

- X.find_succ(k)
  
  if k in (X, succ]
    return succ
  else
    return succ.find_succ(k);
Search using succ links

N51.find_succ(K52) = N56

N14.find_succ(K52)
Inserting Data

- X.DHTinsert(k, v)
  
  if k in (X, succ]
    succ.insert(k,v)
  else
    succ.DHTinsert(k, v);
Finger Table

• The i-th entry of node $n$ contains the address of successor:
  • $(n + 2^{i-1}) \mod 2^m$

finger table for N8

<table>
<thead>
<tr>
<th>Address</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>N8+1</td>
<td>N14</td>
</tr>
<tr>
<td>N8+2</td>
<td>N14</td>
</tr>
<tr>
<td>N8+4</td>
<td>N14</td>
</tr>
<tr>
<td>N8+8</td>
<td>N21</td>
</tr>
<tr>
<td>N8+16</td>
<td>N32</td>
</tr>
<tr>
<td>N8+32</td>
<td>N42</td>
</tr>
</tbody>
</table>

$m=6$
Finger Table

finger table for N8

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N8+1</td>
<td>N14</td>
</tr>
<tr>
<td>N8+2</td>
<td>N14</td>
</tr>
<tr>
<td>N8+4</td>
<td>N14</td>
</tr>
<tr>
<td>N8+8</td>
<td>N21</td>
</tr>
<tr>
<td>N8+16</td>
<td>N32</td>
</tr>
<tr>
<td>N8+32</td>
<td>N42</td>
</tr>
</tbody>
</table>

m=6
## Finger Table

Node responsible for key $8+32 = 40$
Example

find_succ(K54)
Example

findsucc(K54)
Example

N56
N51
N48
N42
N38
N32
N21
N14
N8
N1
K54

find_succ(K54)
Another Example

\[ \text{find\_succ}(K7) \]
Example

find_succ(K7)
Example

```
find_succ(K7)
N7
return N8
```

Diagram:
- Nodes: N1, N51, N56, N48, N42, N38, N32, N21, N14, N8, K7
- Connection: N8 -> K7
Adding Nodes to Circle

For now, assume nodes never die.

Need to:
1. Update links
2. Move data
New Node X Joins

- Node Y is known to be in ring
- X.join(Y)
  pred := nil;
  succ := Y.find_succ(X);
Periodic Stabilization

- X.stabilize()
  \[ Y := \text{succ.p}red; \]
  \[ \text{if } Y \text{ in } (X, \text{succ}) \text{ succ } := Y; \]
  \[ \text{succ.notify}(X); \]

- X.notify(Z)
  \[ \text{if pred } = \text{ nil OR } Z \text{ in } (\text{pred}, X) \]
  \[ \text{pred } := Z; \]
Chord Discussion

• Performance
  • Search is $O(\log n)$
  • Resilience to failures: replication to successor nodes

• Qualitative Criteria
  • Search predicates: equality of keys only
  • Global knowledge: key hashing, network origin
  • Peer autonomy: nodes have by virtue of their address a specific role in the network

• In terms of practical use, CHORD and similar structured P2P networks have not caught for consumer-oriented file sharing, but the ideas are embedded in virtually all cloud and distributed applications today
Today’s Champion: BitTorrent
Our story so far …

• Napster
• Gnutella
• Kazaa/FastTrack
• (CHORD)
• …?
So what replaced FastTrack?

- BitTorrent
- Introduced by Bram Cohen in Summer of 2001
- Just a few months after FastTrack goes online, actually
- At the time, though, there weren't any groups to host the trackers that were needed for the protocol to work
- That wouldn't change until early 2003
BitTorrent

• Motivation
  • Popular content exhibits temporal locality: flash crowds
    • E.g., slashdot effect, CNN on 9/11, new movies, new OS releases

• Big idea: allow others to download from you while you are downloading
  • Efficient fetching, not searching
  • Single publisher, many downloaders
Aside: Game Theory

• Study of strategic decision making
• Major impact on international relations, economics, biology, computer science, …

• Basic game:
  • Players
    • Information available to players
  • Actions available to players
  • Payoffs
Example Game (Prisoner’s Dilemma)

<table>
<thead>
<tr>
<th></th>
<th>Cooperate</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td>(5, 5)</td>
<td>(0, 8)</td>
</tr>
<tr>
<td>Defect</td>
<td>(8, 0)</td>
<td>(2, 2)</td>
</tr>
</tbody>
</table>
Another example ...

Majority of Class

<table>
<thead>
<tr>
<th></th>
<th>Hand</th>
<th>Fist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand</td>
<td>(20, —)</td>
<td>(0, —)</td>
</tr>
<tr>
<td>Fist</td>
<td>(30, —)</td>
<td>(10, —)</td>
</tr>
</tbody>
</table>

You
Repeated Prisoner’s Dilemma

• Play the game not just once
• But over and over
• Hopefully, we can find some mutually beneficial outcomes by cooperating

• Strategies
  • “Grim Trigger”: I will begin by cooperating, but once you defect I will defect for the rest of the game
  • “Tit for tat”: I will begin by cooperating, but then I will replicate your previous action (e.g., if you defect, then I will defect next time; if you cooperate, then I will cooperate next time)
Example Game
(Repeated Prisoner’s Dilemma)

<table>
<thead>
<tr>
<th>Player 1</th>
<th>Cooperate</th>
<th>Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooperate</td>
<td>(5, 5)</td>
<td>(0, 8)</td>
</tr>
<tr>
<td>Defect</td>
<td>(8, 0)</td>
<td>(2, 2)</td>
</tr>
</tbody>
</table>
OK, back to BitTorrent …

• Motivation
  • Popular content exhibits temporal locality: flash crowds
  • E.g., slashdot effect, CNN on 9/11, new movies, new OS releases

• Big idea: allow others to download from you while you are downloading
  • Efficient fetching, not searching
  • Single publisher, many downloaders
What exactly is BitTorrent?

- From bittorrent.org:
  - BitTorrent is a free speech tool.
  - BitTorrent gives you the same freedom to publish previously enjoyed by only a select few with special equipment and lots of money.
  - You have something terrific to publish -- a large music or video file, software, a game or anything else that many people would like to have.
  - But the more popular your file becomes, the more you are punished by soaring bandwidth costs.
  - If your file becomes phenomenally successful and a flash crowd of hundreds or thousands try to get it at once, your server simply crashes and no one gets it.
  - There is a solution to this vicious cycle: BitTorrent
  - With BitTorrent free speech no longer has a high price.
What is BitTorrent?

- Efficient content distribution system using *file swarming*.
  
  *Does not perform all the functions* of a typical p2p system, like *searching*.

- The throughput *increases* with the number of downloaders via the efficient use of network bandwidth.
File sharing

To share a file or group of files, the initiator first creates a .torrent file, a small file that contains

- Metadata about the files to be shared, and
- Information about the tracker, the computer that coordinates the file distribution.

Downloaders first obtain a .torrent file, and then connect to the specified tracker, which tells them from which other peers to download the pieces of the file.
How it works

The file to be distributed is split up into pieces and an SHA-1 hash is calculated for each piece.

```
| 0 | 1 | 2 | 3 | 4 | 5000 |
```
BT Components

The peers first obtain a metadata file for each object.

The metadata contains:

- The SHA-1 hashes of all pieces
- A mapping of the pieces to files
- Piece size
- Length of the file
- A tracker reference
BT Components

The **tracker** is a **central server** keeping a list of all peers participating in the swarm.

- A **swarm** is the set of peers that are participating in distributing the same files.
- A peer joins a swarm by asking the tracker for a peer list and connects to those peers.
**BitTorrent Lingo**

Seeder = a peer that provides the complete file.

Initial seeder = a peer that provides the initial copy.

**One who is downloading (not a derogatory term)**
Simple example

Seeder: A

Downloader B: \{1,2,3,4\} \{5\}

Downloader C: \{1,2,3\} \{5\}

User: \{1,2,3,4,5,6,7,8,9,10\}
Basic Idea

- As a leecher downloads pieces of the file, replicas of the pieces are created. *More downloads mean more replicas available*

- As soon as a leecher has a complete piece, it can potentially share it with other downloaders. Eventually each leecher becomes a seeder by obtaining all the pieces, and assembles the file. Verifies the checksum.