PART 3.
DATA STORAGE AND FLOW MANAGEMENT

Sangmi Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs435

Today’s topics

- FAQs
- Dynamo: Key-Value storage

FAQs

- Final Report of TP
  0. Title
  1. Problem description: Describe your problem and goal
  2. Description of your data: The characteristics of your data (e.g. why is it challenging?)
  3. Your approaches (Methodology)
    - Description of algorithm that you have used (e.g. description of linear regression, and how you applied the algorithm on your problem)
    - Description of your software (e.g. your design of mapreduce)
    - System architecture
  4. Discussion of your analysis
    - What did you find from the results of your analysis?
    - Evaluation of your approach (Accuracy, latency, etc.)
    - Did you find any challenges during your project? Please explain those here.
  5. Your contributions
    Please specify contributions of each team member here.

Key Value Stores: Dynamo

(1) Partitioning
Chord

Scalable Key location in Chord

- Let $m$ be the number of bits in the key/node identifiers
- Each node $n_i$ maintains,
  - A routing table with (at most) $m$ entries
  - Called the finger table

Definition of variables for node $n_i$, using $m$-bit identifiers

- $\text{finger}[i].\text{start} = (n + 2^{i}) \mod 2^{m}, 1 \leq i \leq m$
- $\text{finger}[i].\text{interval} = [\text{finger}[i].\text{start}, \text{finger}[i+1].\text{start})$
- Exception: $\text{finger}[m].\text{interval} = [\text{finger}[m].\text{start}, \text{finger}[0].\text{start}-1])$
- $\text{finger}[i].\text{succ} = \text{first node} \geq n.\text{finger}[i].\text{start}$
- Finger table
  - The Chord identifier
  - The IP address of the relevant node

- First finger of $n$ is its immediate successor on the circle

 lookup process (1/3)
- Each node stores information about only a small number of other nodes
- A node’s finger table generally does not contain enough information to determine the successor of an arbitrary key $k$
- What happens when a node $n$ does not know the successor of a key $k$?
  - If $n$ finds a node whose ID is closer than its own to $k$, that node will know more about the identifier circle in the region of $k$ than $n$ does

 lookup process (2/3)
- $n$ searches its finger table for the node $j$
  - Whose ID most immediately precedes $k$
  - Ask $j$ for the node it knows whose ID is closest to $k$

1. Go clockwise
2. Never overshoot

 lookup process (3/3)
- Request comes into node (machine) 1 to find the successor of id 4.
  - 1. Node 1 wants to find the successor of identifier 4
  - 2. Identifier 4 belongs to [3,5)
  - 3. Check succ: 3
  - 4. Node 1 asks node 3 to find successor of 4
  - 5. Successor of 4 is 0

- Request comes into node (machine) 1 to find the successor of id 1.
  - 1. Node 1 wants to find the successor of identifier 1
  - 2. Identifier 1 belongs to [7,3)
  - 3. Check succ: 0
  - 4. Node 3 asks node 0 to find successor of 1
  - 5. Successor of 1 is 1
Lookup process: example 2

0. Request comes into node 3.
1. Node 3 wants to find the successor of identifier 0
2. Identifier 0 belongs to [7,3)
3. Check succ: 0
4. Node 3 asks node 0 to find successor of 1
5. Machine is using identifier 0 as well. succ is 0.

Finger table:

- Start int succ
- 1 [1,2) 1
- 2 [2,4) 3
- 3 [3,5) 3
- 4 [4,0) 0
- 5 [5,1) 0
- 6 [6,7) 0
- 7 [7,3) 0

Finger table:

- Start int succ
- 2 [2,3) 3
- 3 [3,5) 3
- 5 [5,1) 0

Finger table:

- Start int succ
- 4 [4,5) 0
- 5 [5,7) 0
- 7 [7,3) 0

4. Request comes into node 3.
1. Node 3 wants to find the successor of identifier 0
2. Identifier 0 belongs to [7,3)
3. Check succ: 0
4. Node 3 asks node 0 to find successor of 1
5. Machine is using identifier 0 as well. succ is 0.

Finger table:

- Start int succ
- 1 [1,2) 1
- 2 [2,4) 3
- 3 [3,5) 3
- 4 [4,0) 0
- 5 [5,1) 0
- 6 [6,7) 0
- 7 [7,3) 0

Finger table:

- Start int succ
- 2 [2,3) 3
- 3 [3,5) 3
- 5 [5,1) 0

Finger table:

- Start int succ
- 4 [4,5) 0
- 5 [5,7) 0
- 7 [7,3) 0

Lookup cost

- With high probability, the number of nodes that must be contacted to find a successor in an N-node network is O(logN)

Dynamo’s partitioning

- Inspired by Consistent Hashing and Chord
- Dynamo is a zero-hop DHT
  - Each node maintains enough routing information locally
  - When a node starts for the first time
    - Chooses its set of tokens (virtual nodes in the consistent hash space)
    - Maps nodes to their respective token sets
    - Stores both tokens and nodes onto disk
  - Repeated reconciliation of the membership change
- Partitioning and placement information are propagated via the gossip-based protocol
  - Token ranges handled by its peers
  - Direct forwarding of read/write operations are possible