Load bounded Multi-Probe Consistent Hashing

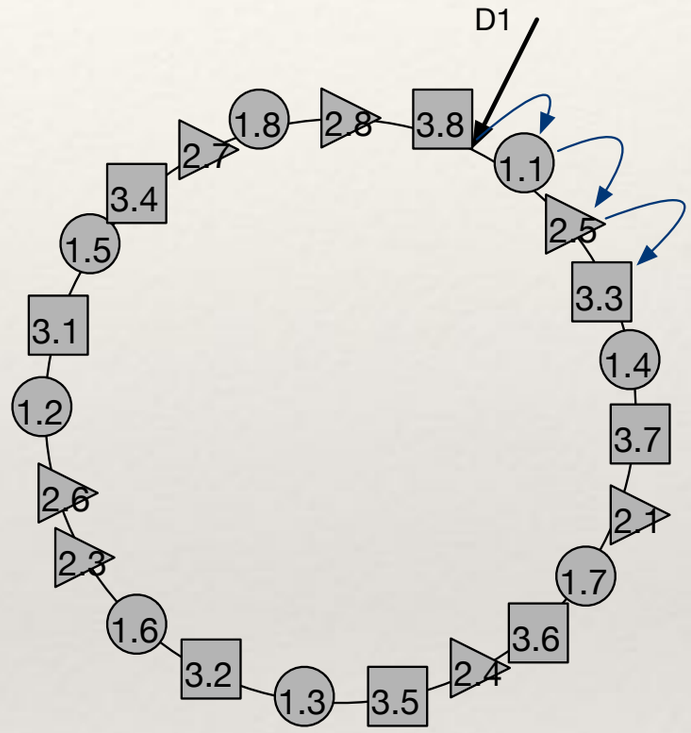
Fast, Memory Efficient and Durable Load Balancing

Wei Xie

DISCL Seminar, Apr 18th, 2017

Introduction

- Consistent hashing uses virtual nodes (virtual servers) for load balancing
- To achieve adequate balance, certain number of vnodes are required (~100s)
- Issues: memory footprint, correlated failure durability



Defining Load Balance

- Load balance is defined as the Peak to Average Ratio
- For achieving (1+e) peak-to-average ratio, it requires $V=O(\ln(N)/e^2)$ virtual nodes
- 1000 servers and 1.1 ratio, around 300,000 virtual nodes needed; 1.05 ratio, around 1,200,000 needed

Table 1: Memory Usage for Consistent Hashing

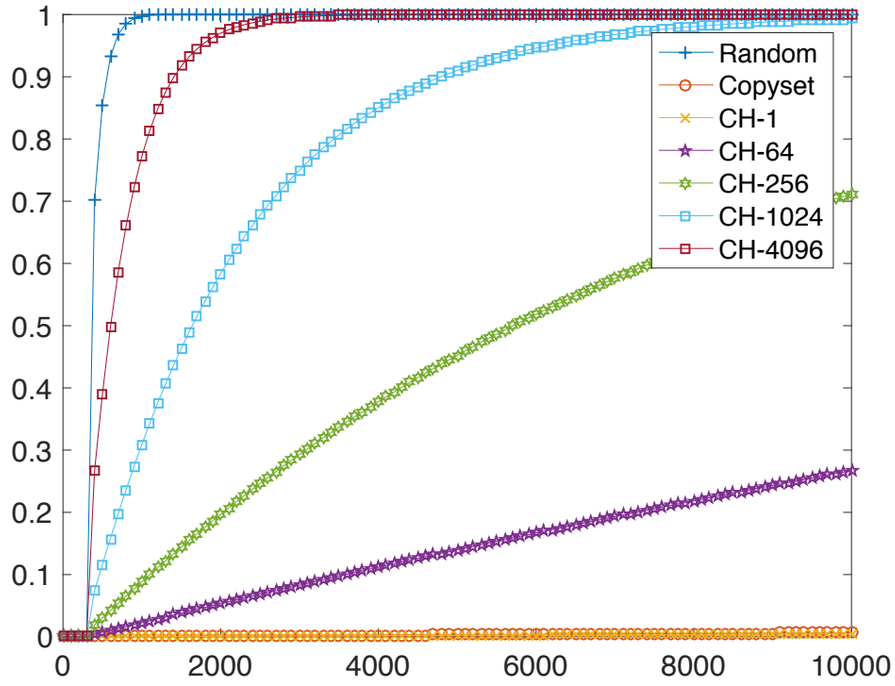
	V=100	V=1000	V=10000	V=100000
$\epsilon = 0.1$	1.4MB	21MB	281MB	3.4GB
$\epsilon = 0.05$	5.6MB	84MB	1.1GB	13.7GB

Correlated Failure Durability

- In a cluster-wide failure event such as power outage occurs, about 0.5-1% of the nodes fail to reboot
- The probability that data become unavailable determines the durability

$$1 - (1 - \frac{\binom{F}{R}}{\binom{N}{R}})^{n_{copyset}}$$

- With V virtual nodes:
  - $n_{copyset}$=O(N*V)
  - More virtual nodes, less durability

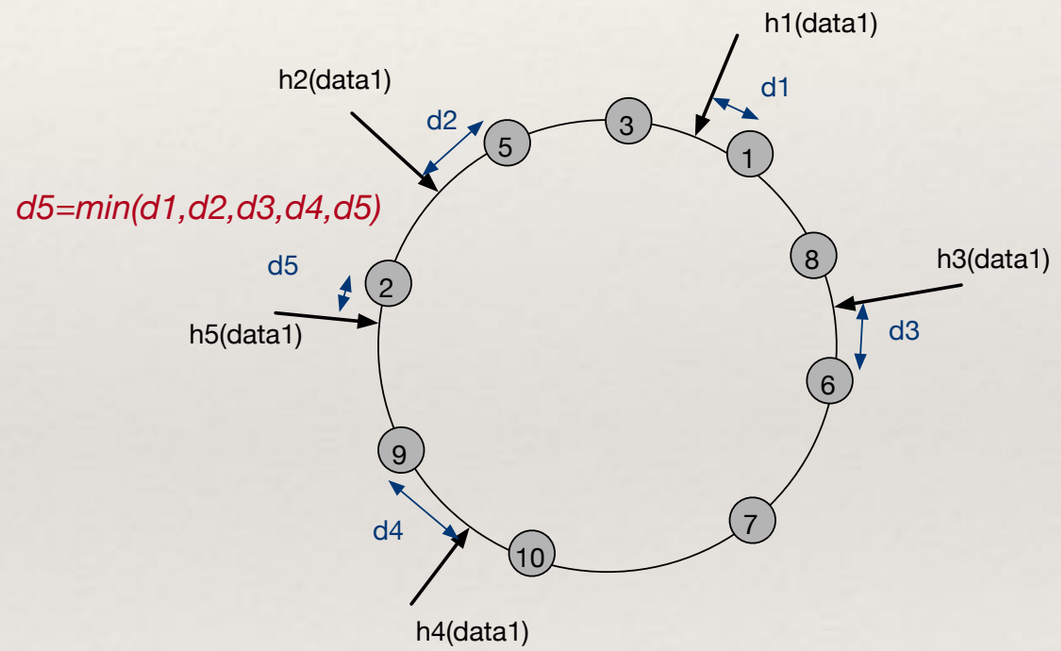


Recovery Rate

- When a server joins/leaves, data needs to be re-distributed/recovered
- The recovery rate is determined by the scatter width
- Virtual node adds scatter width as it adds different combinations of data placements

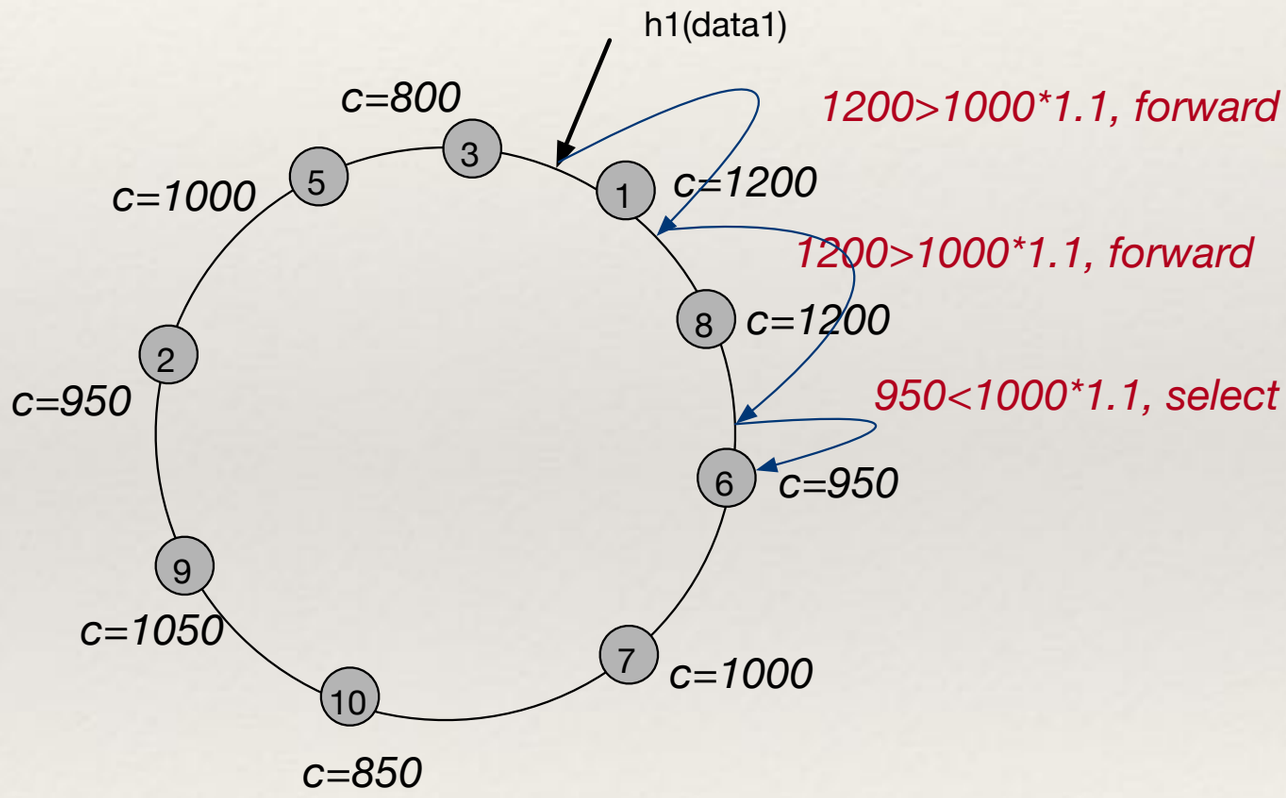
Multi-Probe Consistent Hashing

- Virtual data instead of virtual server
- Trade speed for memory footprint and durability
- Hash data ID multiple times
- PTA ratio=K/(K-1) + O(1)
- K=1+ 1/e = 11 for e=0.1, 11 times slower

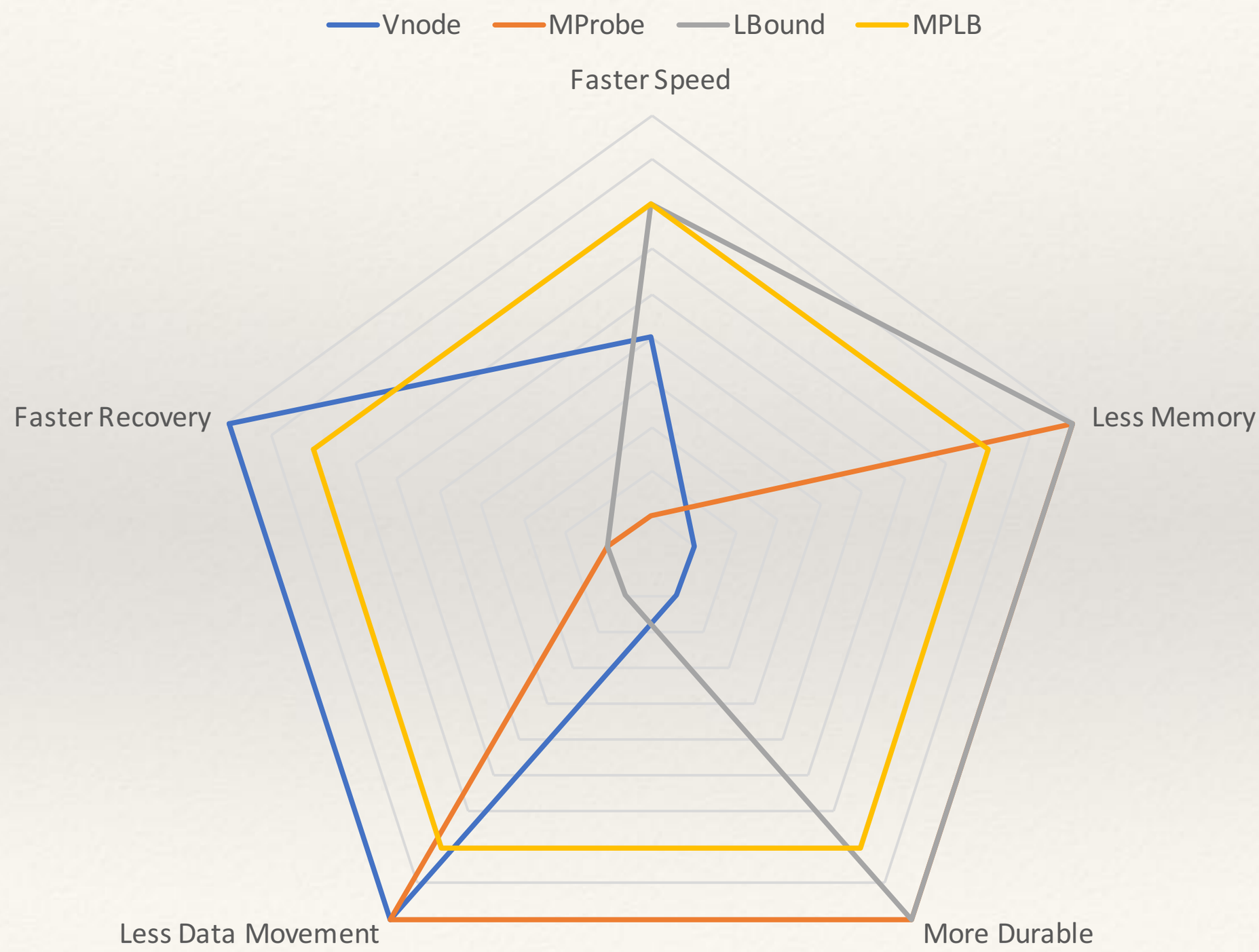


Load Bounded Consistent Hashing

- Each server's load is tracked, when overloaded forward to the next server
- Not sacrifice speed, memory, or durability, but data movement could be enormous $O(1/e^2)$



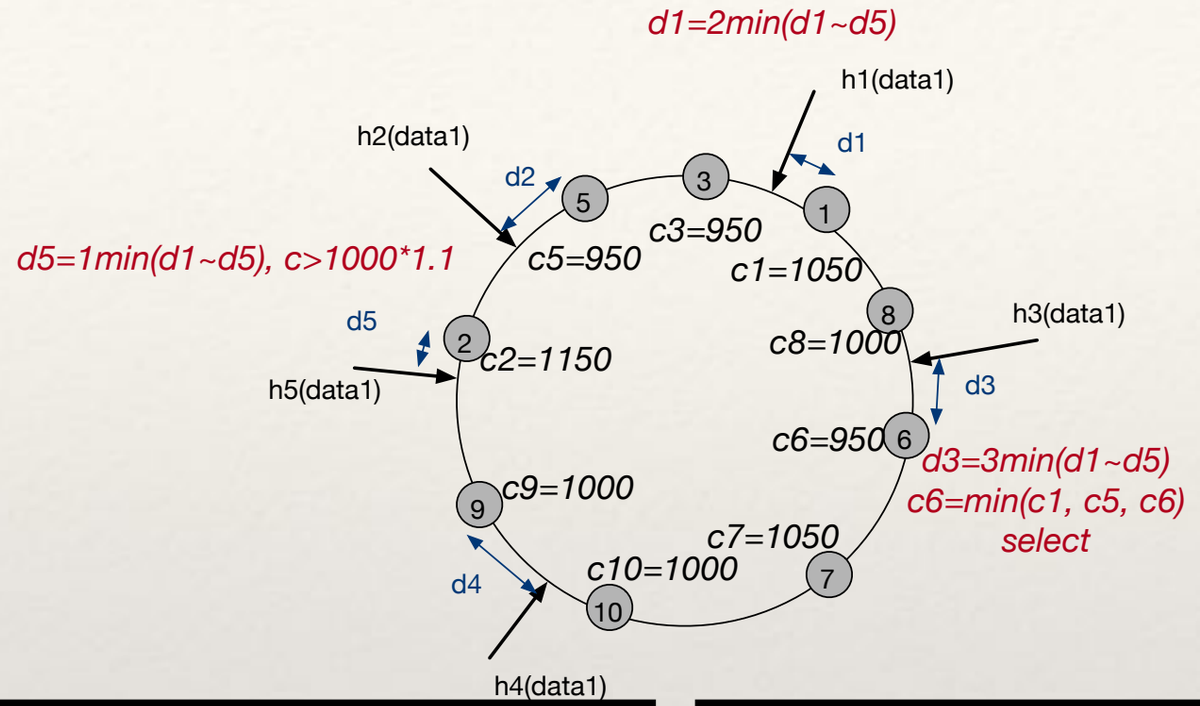
Comparison of Consistent Hashing Variants



Load Bounded Multi-Probe CH

- Goal: memory efficient, durable, reasonably fast, and small data movement
- General idea: use low-order multi-probe (i.e. K=5) to keep the assignment speed reasonable, and use a relaxed load bounding to achieve the desired load balance

Load Bounded Multi-Probe CH



- Multi-Probe

1. Get K data hashes
2. Get i that min |hash(i)-server(i)|, i=1 to K, server(i) is the successor of hash(i) on the hash ring
3. Select server(i) as the placement server

- LB-MP

1. Get K data hashes
2. Get B (bounding factor) server(i)s that have min| hash(i)-server(i)|, i=1 to K
3. Select the server in the B selected servers that has smallest load

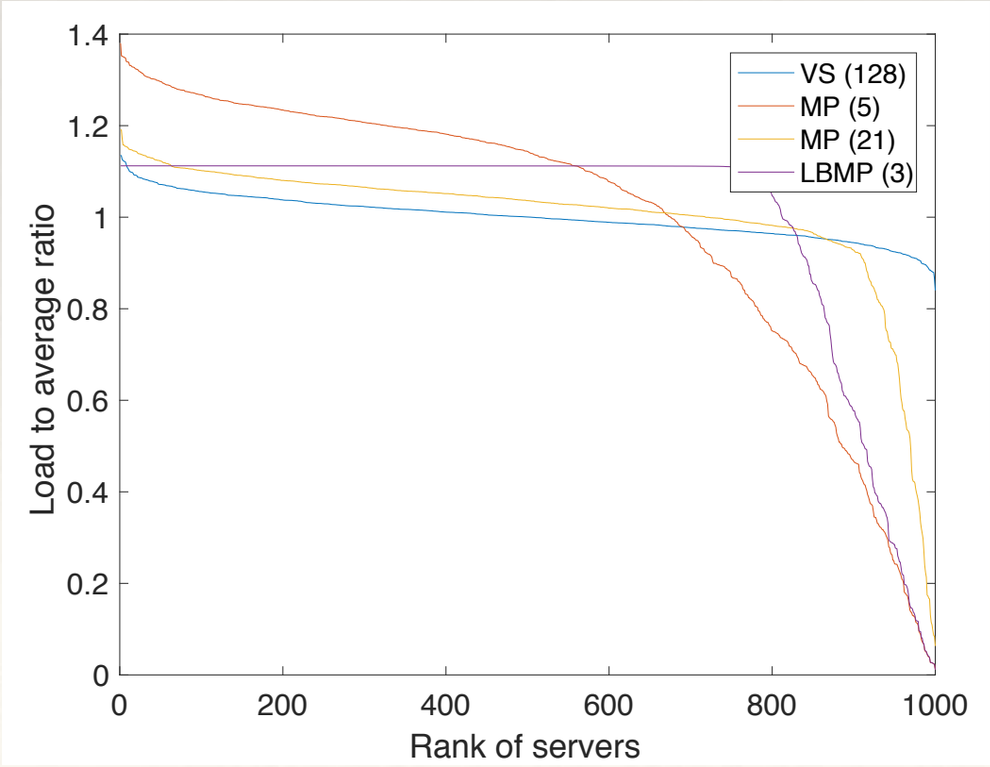
Analysis of LBMP

- Speed: the number of probing K is small
- Memory efficiency: no virtual nodes -> low memory footprint
- Durability: no virtual nodes -> less copysets -> durable
- Data movement: the bounding factor determines the data movement (B=3 means at most 3 times more data movement)

Load Balance Test

Table 2: Load balance with 1000 servers. The second column is for the peak-to-average load ratio. The other columns are for the top X-tile over average load. "VS" is short for virtual server. "MP" is short for multi-probe. "LBMP" is short for load bounded multi-probe. "LB" is short for loud bounded.

	VS (4)	VS (16)	VS (128)	MP(2)	MP(5)	MP(21)	LBMP(2)	LBMP(3)	LBMP(4)	LB(1.1)
max	4.50	1.67	1.13	2.24	1.40	1.19	1.22	1.11	1.11	1.10
99%	3.98	1.54	1.10	2.17	1.33	1.15	1.20	1.11	1.11	1.10
90%	3.31	1.26	1.05	2.13	1.29	1.10	1.14	1.11	1.11	1.10
50%	2.91	1.19	1.00	2.04	1.00	1.03	1.06	1.11	1.11	1.10



Observation:
LBMP has almost a flat curve in the first 80% servers, and suddenly drops

Assignment Time Test

Table 3: Assignment time with 1000 servers.

	VS (4)	VS (16)	VS (128)	MP(2)	MP(5)	MP(21)	LBMP(2)	LBMP(3)	LBMP(4)	LB(1.1)
Time (ns)	184.5	215.3	261	307.3	768.2	3226.7	794.3	796.3	795.3	155.2

- LBMP is 4X times faster compared to MP to achieve the same load balance, and is only 3X slower than using Virtual Node or Load Bounding (which have other limitations)

Summary

- Identified the important properties a data placement should have and the issues with existing consistent hashing algorithms
- Propose a load bounded multi-probe consistent hashing that balances speed, memory efficiency, durability, and data movement.
- Initial tests show promising results:

  optimal memory efficiency and durability with only 3X sacrifice in speed and data movement (compared to 10X in speed and 100X in data movement in existing algorithms)