TDT4225
Chapter 6 – Partitioning

Svein Erik Bratsberg
Department of Computer Science (IDI), NTNU

Partitioning concepts

- For very large datasets, or very high query throughput:
- We need to break the data up into **partitions**
  - Shards (MongoDB, ElasticSearch)
  - Region (HBase)
  - Vnode (Cassandra, Riak)
  - vBucket (CouchBase)

- **Scalability** is the main reason for partitioning
- **Load sharing** is another reason

Query execution and Partitioning

- For single partition queries (simple queries), each node can independently execute the queries
- Large, complex queries can potentially be parallelized across many nodes, and by shipping SQL and/or data across the network (**data shipping** vs **function shipping**)
- TerraData and Tandem NonStop SQL pionered some of this
- HyperKuben and Clustra came from NTNU and used hash partitioning (and replication)

Partitioning and Replication

- Also called *declustering*

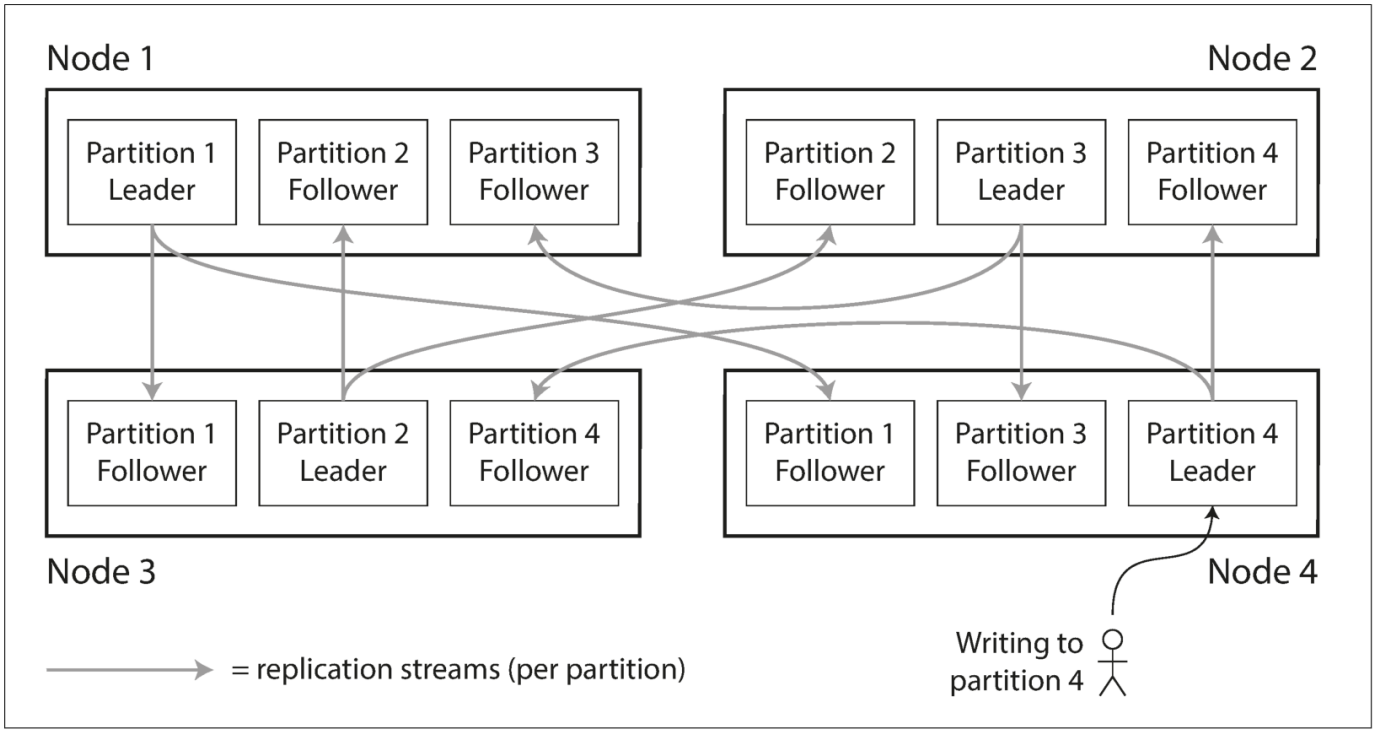


Figure 6-1. Combining replication and partitioning: each node acts as leader for some partitions and follower for other partitions.

Range Partitioning

- Partitioned by the order of the search key

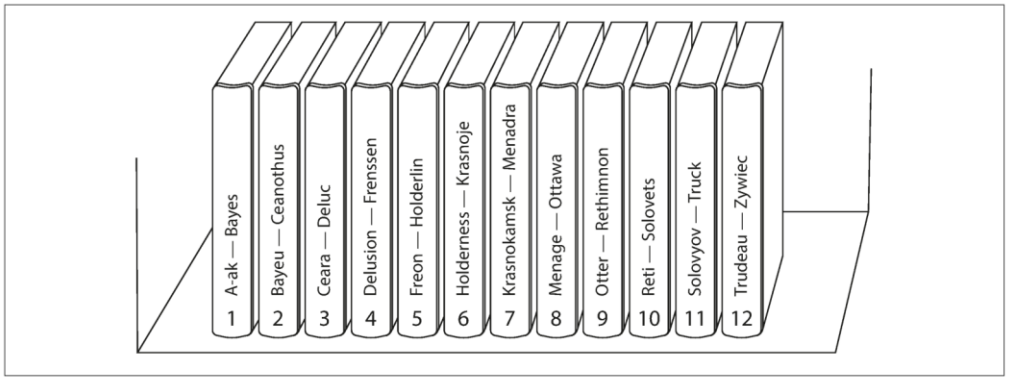


Figure 6-2. A print encyclopedia is partitioned by key range.

- May lead to some hot spots (e.g. newest items)
- Repartitioning may be necessary
- Don’t use timestamps as first key, e.g., use «sensor name» (example in text book)

Hash Partitioning

```
algorithm fnv-1 is
    hash := FNV_offset_basis do

    for each byte_of_data to be hashed
        hash := hash × FNV_prime
        hash := hash XOR byte_of_data

    return hash
```

- A good hash function takes skewed data and makes it uniformly distributed
  - Cassandra and MongoDB use MD5
  - Voldemort uses the Fowler–Noll–Vo function
- Gives good load balance, but no range scans. You have to do range scans on all partitions (MongoDB)
  - Cassandra may use *compound keys*, where only the first part is hashed

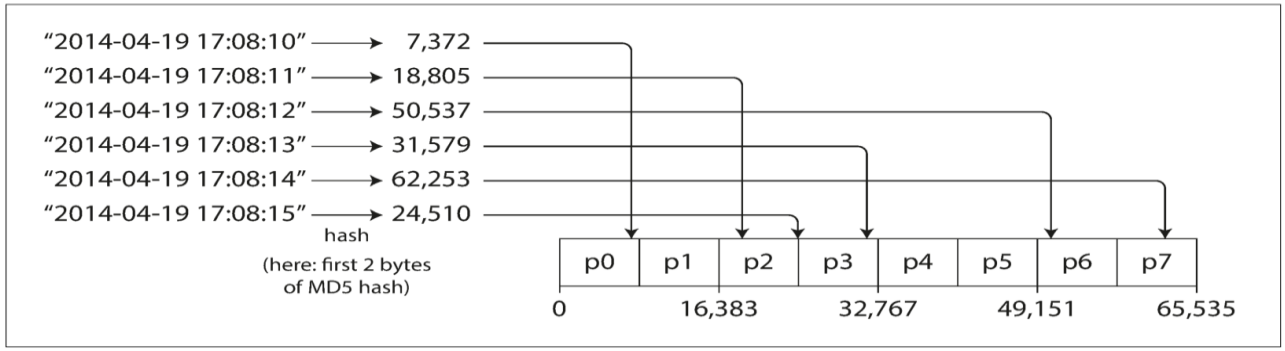


Figure 6-3. Partitioning by hash of key.

Skewed Workloads and Hot Spots

- Sometimes hashing doesn't load balance well
- All writes happen to the same key being hashed
- A technique to solve this is to let the application use an additional random key to the start or the end of the shared key
- It is also possible to use a different hash function if the current hash function doesn't work very well.

Document-Based Indexing

- Document-based indexing is also called *local indexing*
- Used in MongoDB, Riak, Cassandra, Elasticsearch, SolrCloud, and VoltDB
- Read from all nodes, writes to one

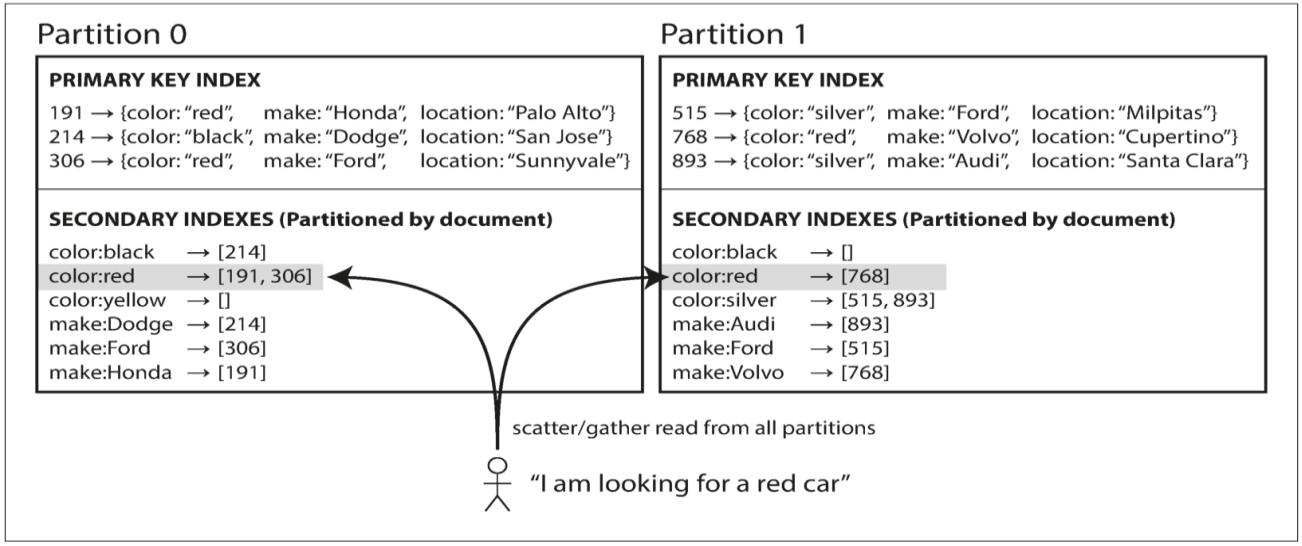


Figure 6-4. Partitioning secondary indexes by document.

Term-Based Indexing

- Term-Based Indexing (term partitioned) is also called *global indexing*
- Reads may become faster, but writes may become slower (they're not local). Updates may be done asynchronously (Amazon Dynamo).

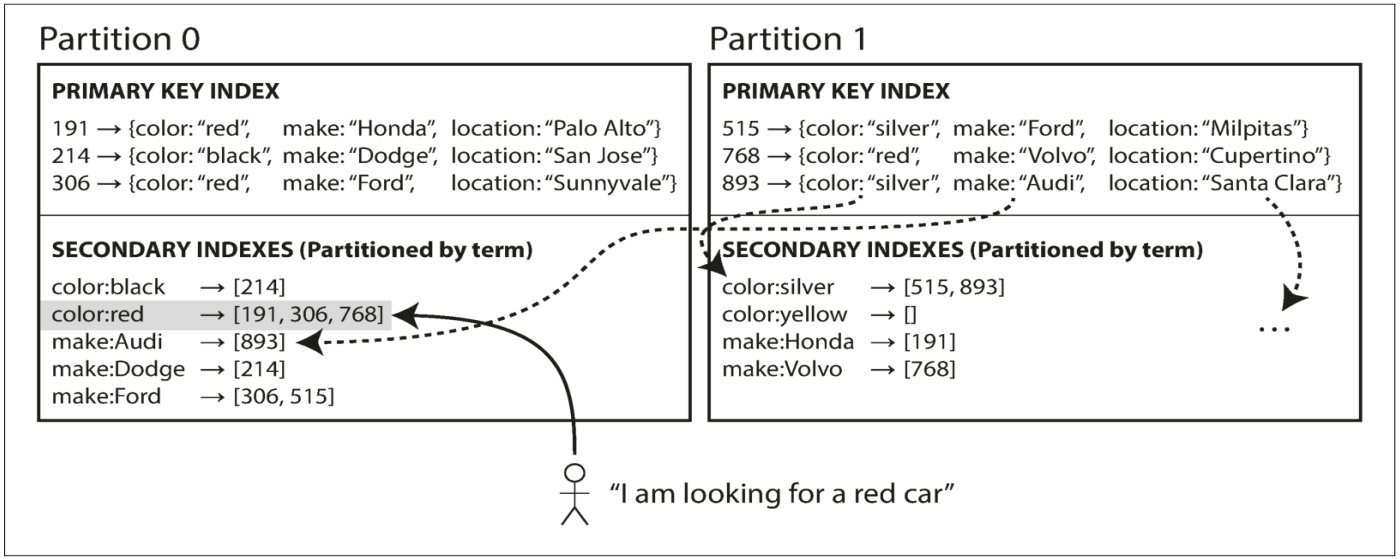
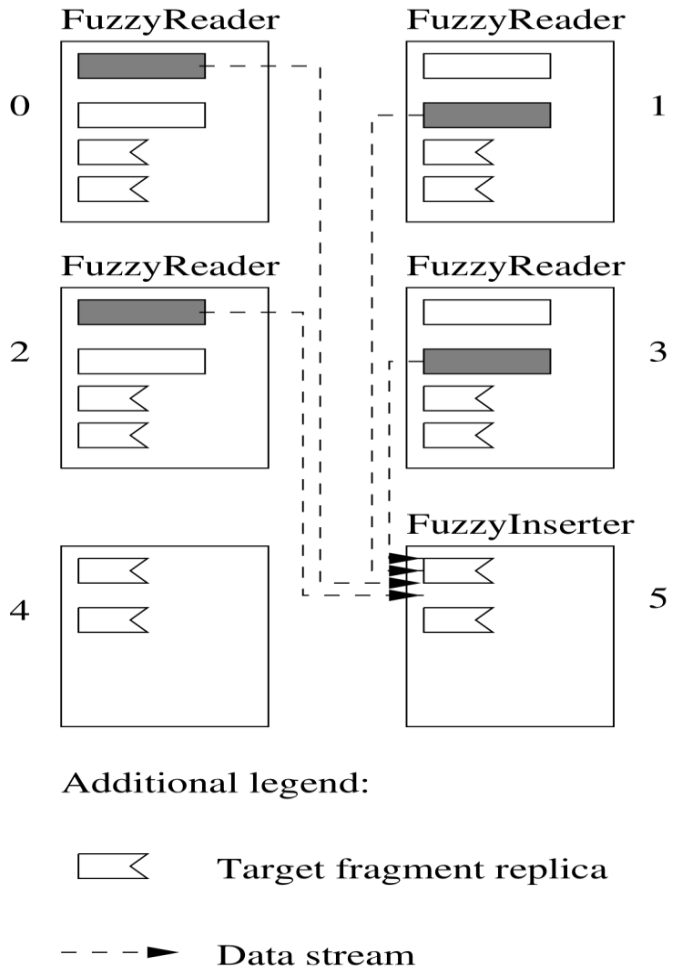


Figure 6-5. Partitioning secondary indexes by term.

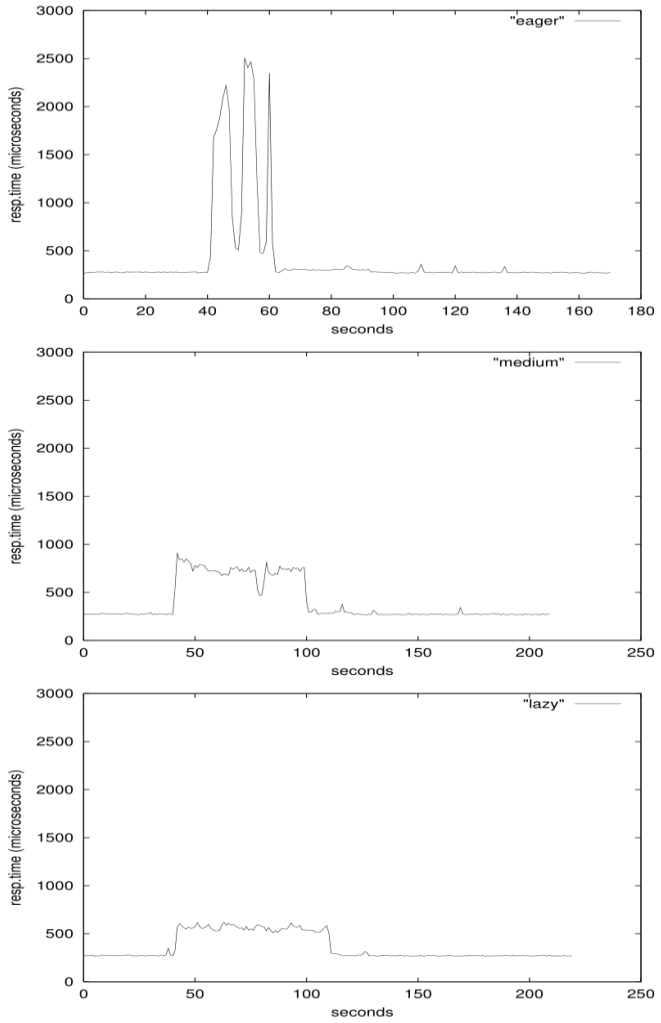
Rebalancing Partitions

- Reasons for rebalancing / repartitioning
  - Increased load
  - Increased data volume
  - Failed nodes need to be replaced
- Hashing with mod N is not a good choice
  - Need to move most records
  - All to all copy
  - Done in Clustra



Rebalancing partitions (2)

- Done in Clustra
- Will slow down concurrent queries



Rebalancing Partitions (3)

- Use a fixed, high number of partitions (Riak, Voldemort,ElasticSearch, Couchbase, ..)

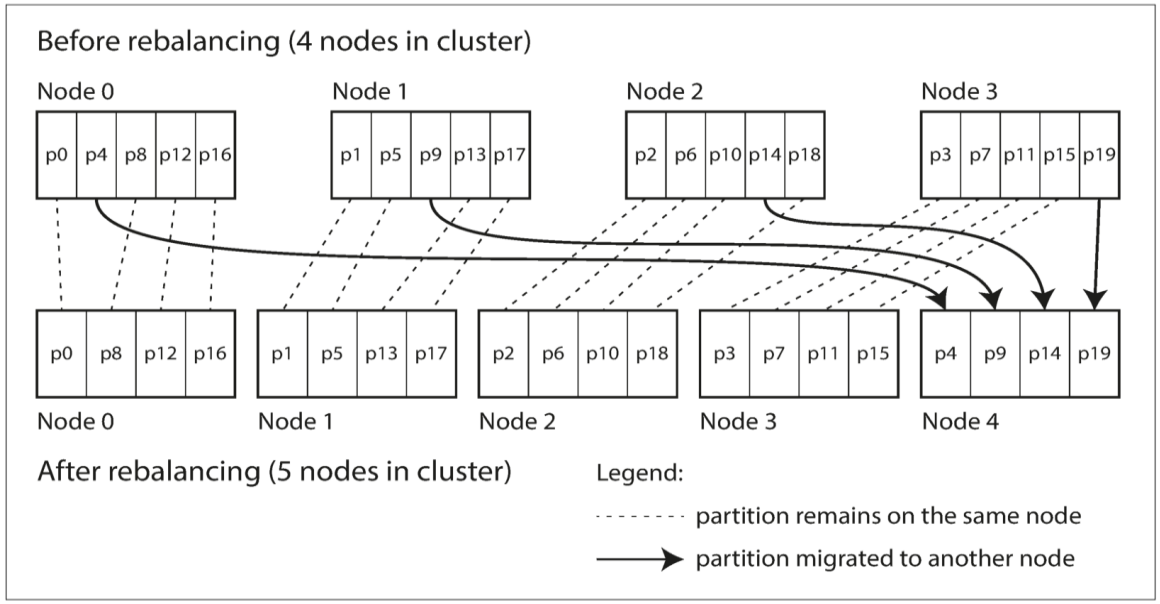


Figure 6-6. Adding a new node to a database cluster with multiple partitions per node.

Dynamic Partitioning (range part.)

- When a partition grows to exceed a configured size, it is split into two partitions
- HBase and RethinkDB create partitions dynamically
- HBase and MongoDB allow an initial set of partitions to be configured on an empty database (*pre-splitting*)
- May also be used on hash partitioning (MongoDB)
- Proportional to the size of data volume
- Proportional to the size of the number of nodes (Cassandra and Ketama)
- Operations: Automatic or manual rebalancing

Request Routing

- Three different approaches
  1. Allow clients to contact any node
  2. Send all requests from clients to a routing tier first
  3. Require that clients be aware of the partitioning and the assignment of partitions

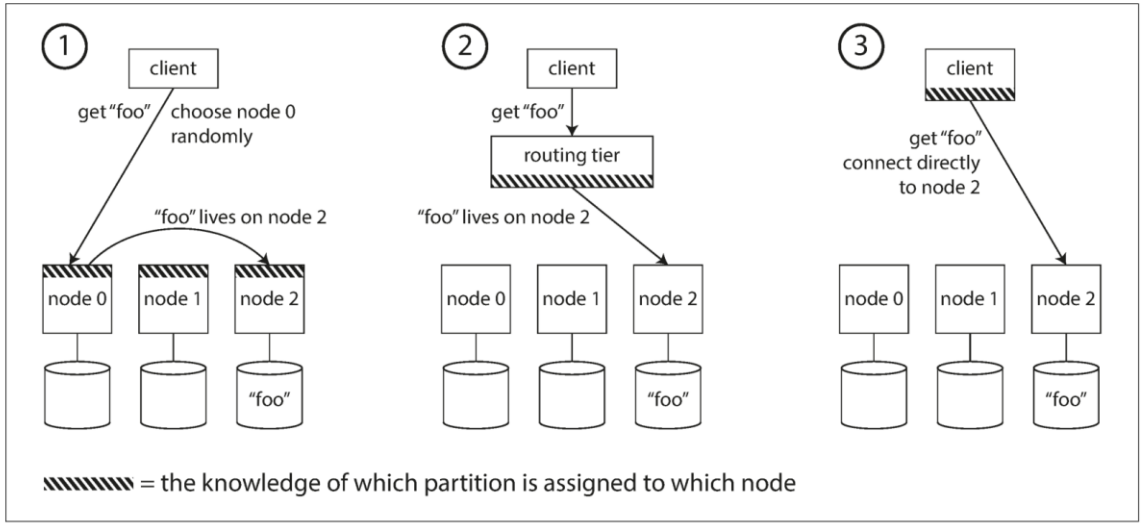
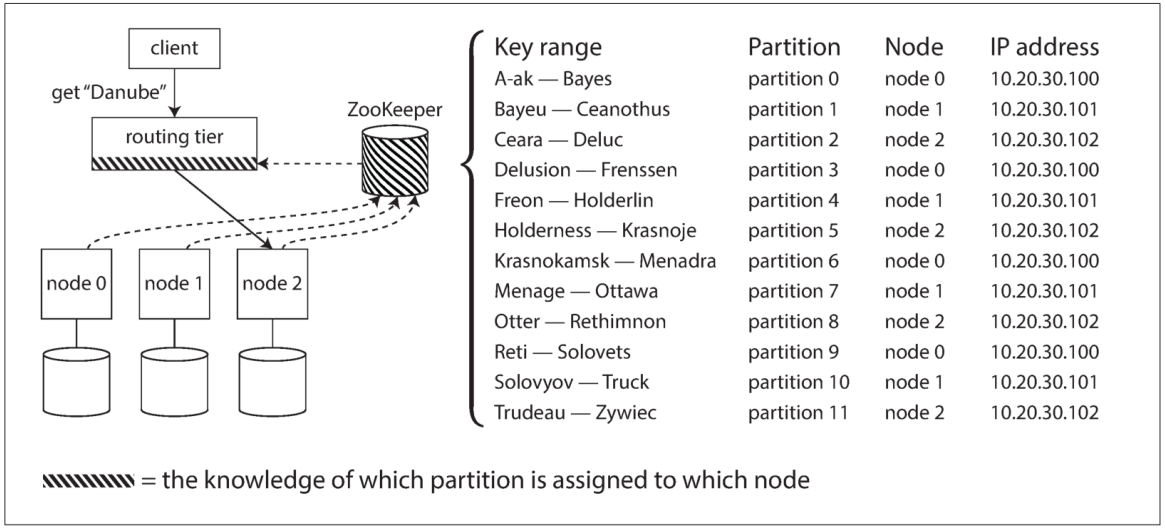


Figure 6-7. Three different ways of routing a request to the right node.

Coordination of services

- How do clients know about changes?
- ZooKeeper



Key range	Partition	Node	IP address
A-ak — Bayes	partition 0	node 0	10.20.30.100
Bayeu — Ceanothus	partition 1	node 1	10.20.30.101
Ceara — Deluc	partition 2	node 2	10.20.30.102
Delusion — Frenssen	partition 3	node 0	10.20.30.100
Freon — Holderlin	partition 4	node 1	10.20.30.101
Holderness — Krasnoje	partition 5	node 2	10.20.30.102
Krasnokamsk — Menadra	partition 6	node 0	10.20.30.100
Menage — Ottawa	partition 7	node 1	10.20.30.101
Otter — Rethimnon	partition 8	node 2	10.20.30.102
Reti — Solovets	partition 9	node 0	10.20.30.100
Solovyov — Truck	partition 10	node 1	10.20.30.101
Trudeau — Zywiec	partition 11	node 2	10.20.30.102

Figure 6-8. Using ZooKeeper to keep track of assignment of partitions to nodes.

- Gossip protocol (Cassandra and Riak)
- Virtual partition protocol (UCSB) used in Clustra

Summary

- Hash partitioning
- Key range partitioning
- Document-partitioned indexes (local indexes)
- Term-partitioned indexes (global indexes)
- Rebalancing
- Request routing
- Coordination of services