Gagnasafnsfræði

Björn Patrick Swift

November 18, 2011

Hver er maðurinn

- 2011: MSc Parallel and Distributed Computer Systems, Vrije Universiteit
- 2008: BSc Hugbúnaðarverkfræði, Háskóli Íslands

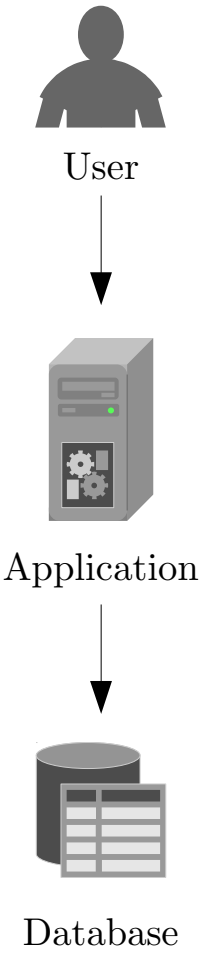
- 2011 – : Amazon AWS
- 2006 – 2009: CCP
- 2006: FRISK

Transactional

ACID properties

- Atomicity
- Consistency
- Isolation
- Durability

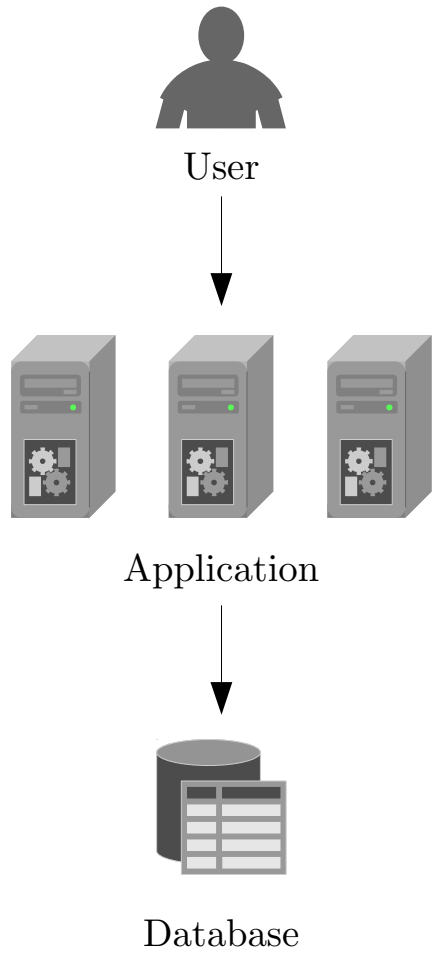
Transactions: All or nothing.



User, application, database

Application layer scales well.

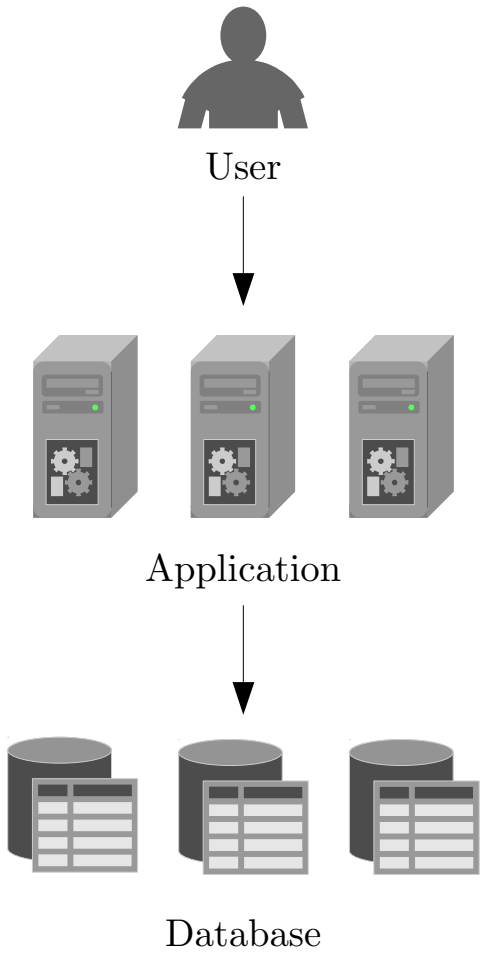
Same does not hold for the database layer



User, application, database

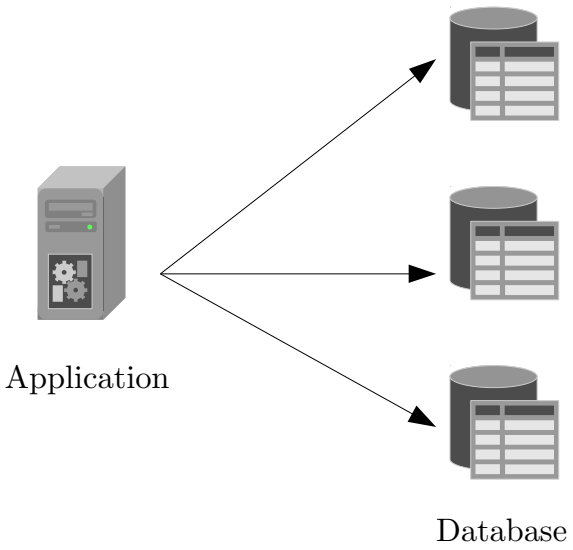
Application layer scales well.

Same does not hold for the database layer



Full replication

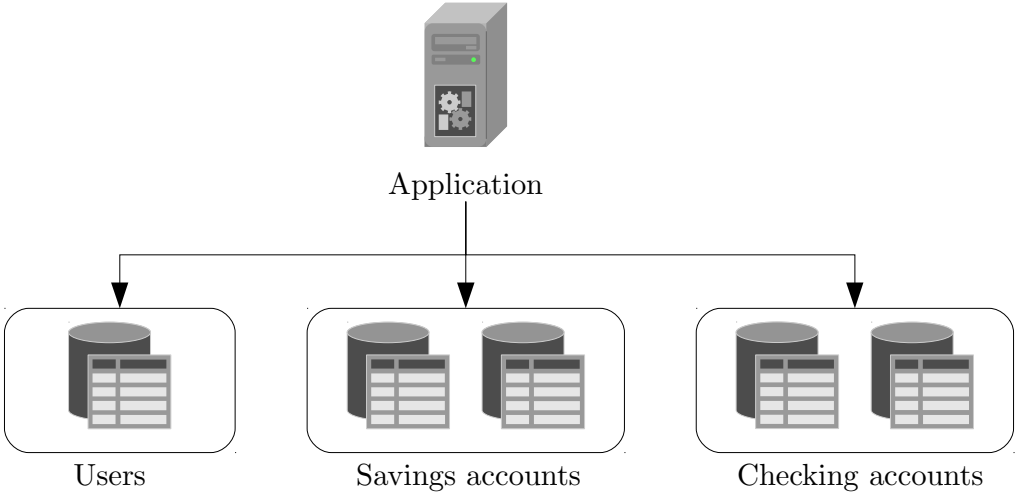
- State is fully replication across all servers
  - Writes need to be performed on all servers



Sharding

- Withdraw 100$ from checking account, deposit to savings account.
  - Operation 1: Withdraw 100$ from checking account.
  - Operation 2: Deposit 100$ to savings account.
  - Could something have changed between operations one and two?

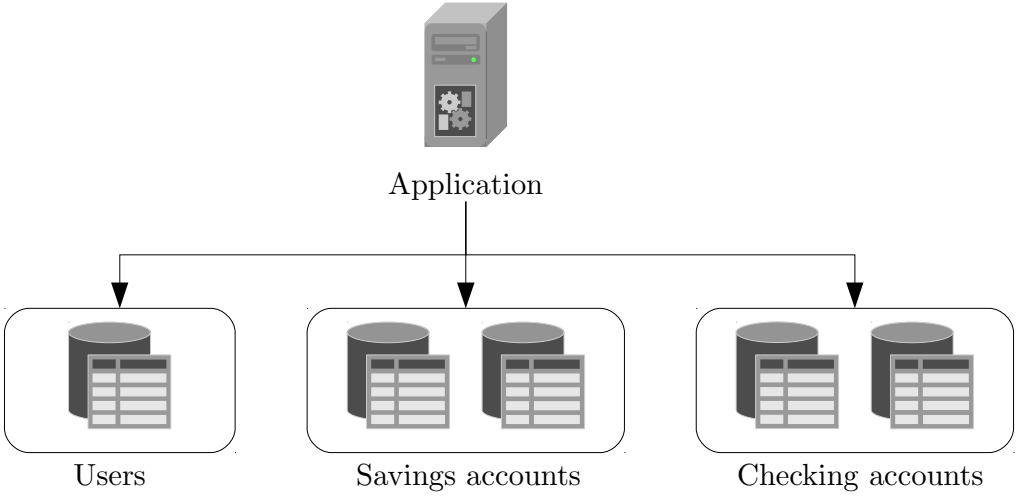
Trade-off between transactional consistency and scalability.



Sharding

- Withdraw 100$ from checking account, deposit to savings account.
  - Operation 1: Withdraw 100$ from checking account.
  - Operation 2: Deposit 100$ to savings account.
  - Could something have changed between operations one and two?

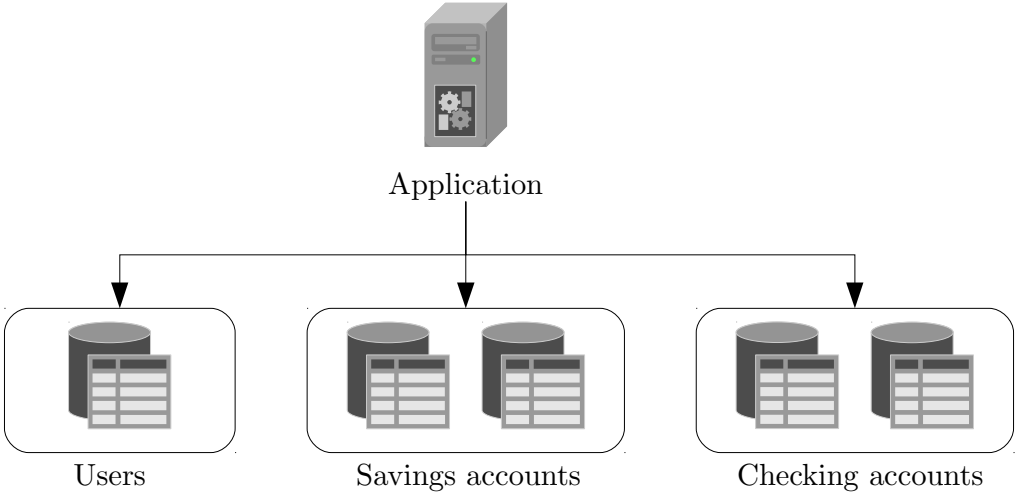
Trade-off between transactional consistency and scalability.



Sharding

- Withdraw 100$ from checking account, deposit to savings account.
  - Operation 1: Withdraw 100$ from checking account.
  - Operation 2: Deposit 100$ to savings account.
  - Could something have changed between operations one and two?

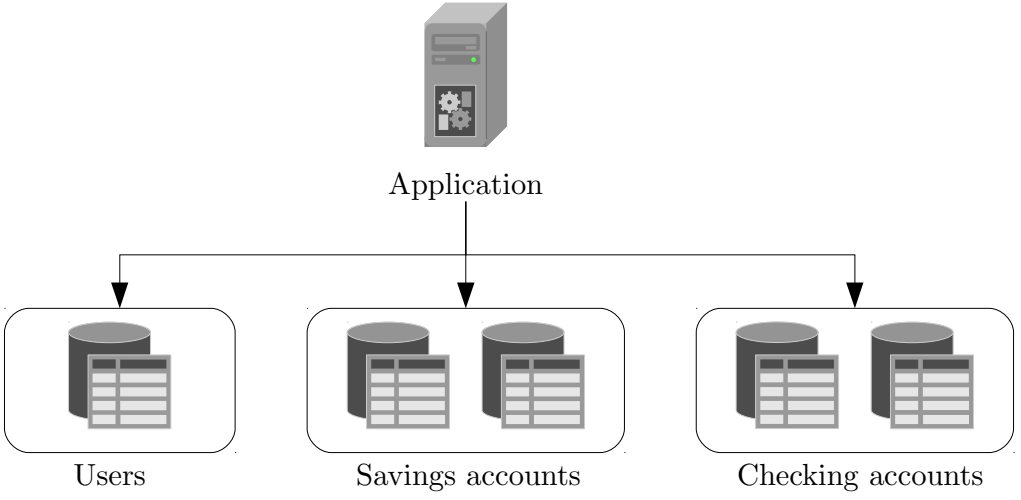
Trade-off between transactional consistency and scalability.



Sharding

- Withdraw 100$ from checking account, deposit to savings account.
  - Operation 1: Withdraw 100$ from checking account.
  - Operation 2: Deposit 100$ to savings account.
  - Could something have changed between operations one and two?

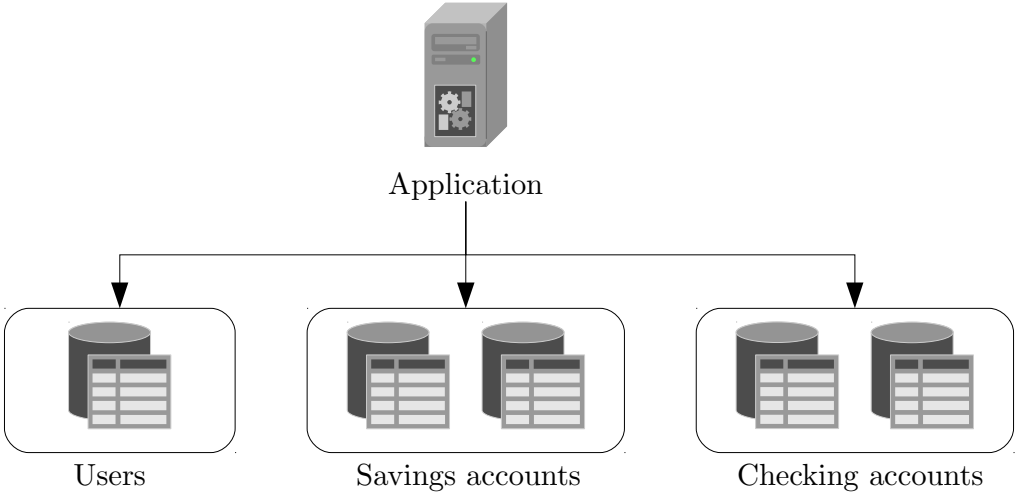
Trade-off between transactional consistency and scalability.



Sharding

- Withdraw 100$ from checking account, deposit to savings account.
  - Operation 1: Withdraw 100$ from checking account.
  - Operation 2: Deposit 100$ to savings account.
  - Could something have changed between operations one and two?

Trade-off between transactional consistency and scalability.



How far can we scale without ACID?

- Emphasis on scalability
- Several distributed data stores, both proprietary and open source.
  - BigTable, HBase
  - Dynamo, Cassandra, Riak
  - SimpleDB
  - CouchDB, MongoDB, ...
- Transactional consistency not supported.

“Practice is just around the corner”

- Consistent hashing
- Chord
- Object versioning with Vector Clocks
- Quorums
- Replica synchronization
- Gossiping

- Incremental scalability
- Symmetry
- Decentralization
- Heterogeneity
- High Availability

- Incremental scalability
- Symmetry
- Decentralization
- Heterogeneity
- High Availability

- Incremental scalability
- Symmetry
- Decentralization
- Heterogeneity
- High Availability

- Incremental scalability
- Symmetry
- Decentralization
- Heterogeneity
- High Availability

- Incremental scalability
- Symmetry
- Decentralization
- Heterogeneity
- High Availability

CAP Theorem

Of three properties of shared-data systems

- data consistency
- system availability
- tolerance to network partition

only two can be achieved at any given time.[a]

[a] Brewer, E. A. 2000. Towards robust distributed systems (abstract). In Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (July 16-19, Portland, Oregon): 7.

We need to relax some constraints.

CAP Theorem

Of three properties of shared-data systems

- data consistency
- system availability
- tolerance to network partition

only two can be achieved at any given time.[a]

[a] Brewer, E. A. 2000. Towards robust distributed systems (abstract). In Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (July 16-19, Portland, Oregon): 7.

We need to relax some constraints.

- get(key)
  - A single object
  - List of objects with conflicting versions and a *context*
- put(key, context, object)

- Objects are binary blobs, Dynamo had no schema.
- Keys are hashed using MD5.

- get(key)
  - A single object
  - List of objects with conflicting versions and a *context*
- put(key, context, object)

- Objects are binary blobs, Dynamo had no schema.
- Keys are hashed using MD5.

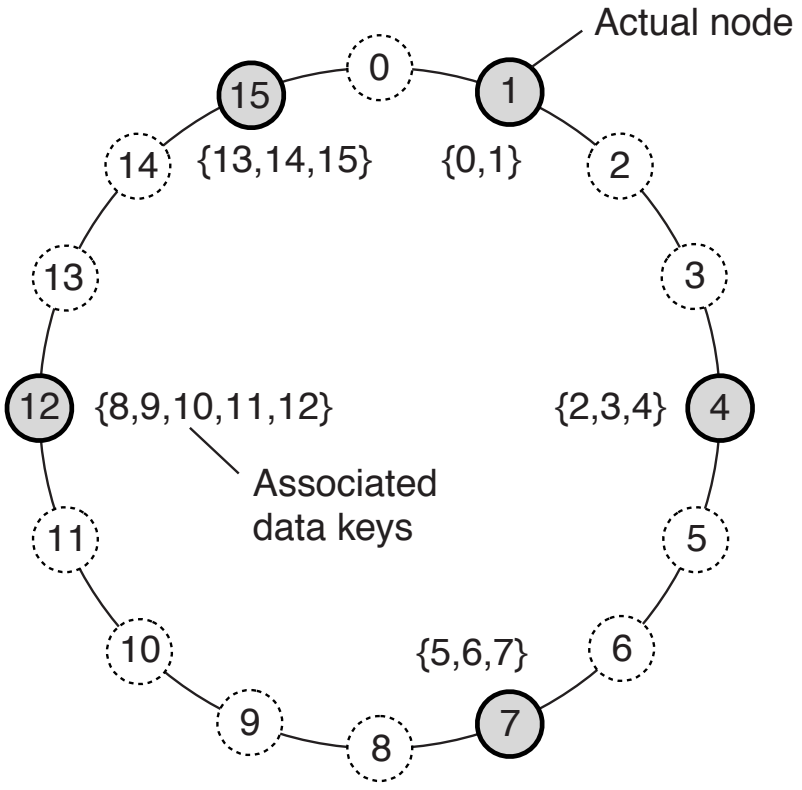
Chord system

... with a twist

- Each node gets multiple tokens in the ring

Virtual nodes address:

- Heterogeneity
- Incremental scaling



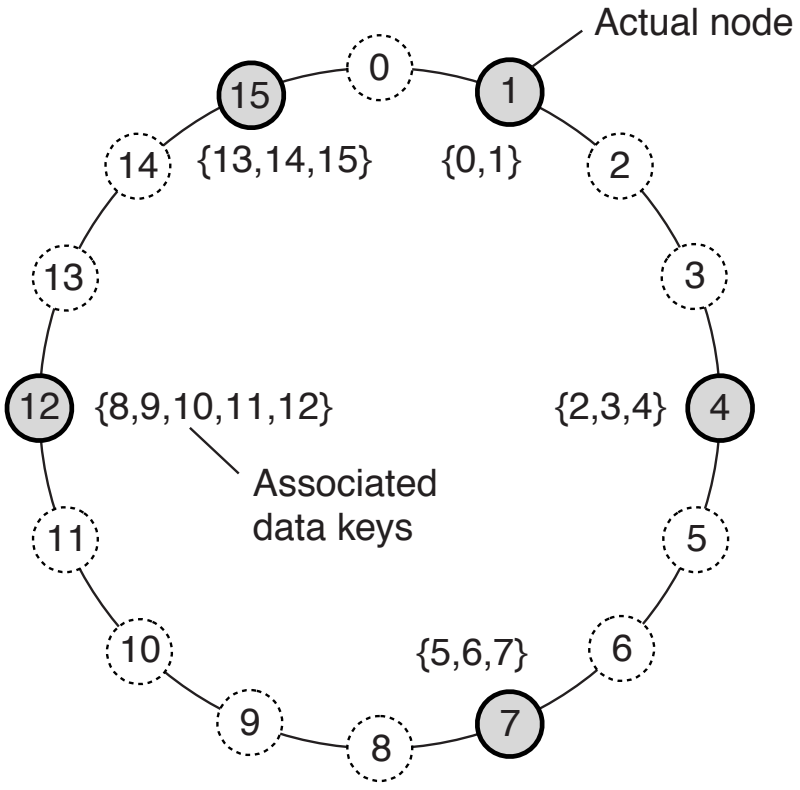
Chord system

... with a twist

- Each node gets multiple tokens in the ring

Virtual nodes address:

- Heterogeneity
- Incremental scaling



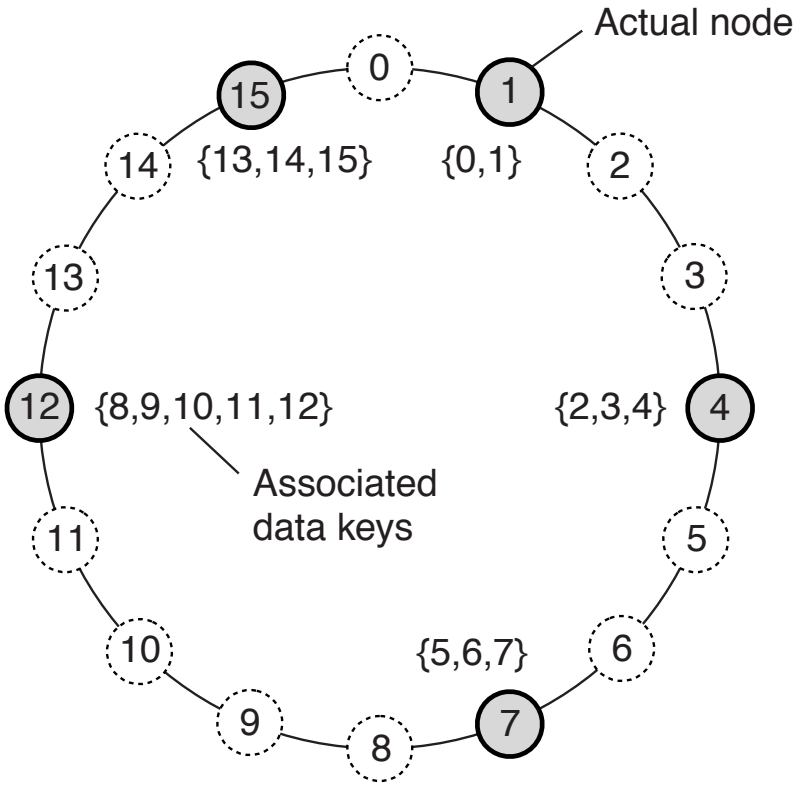
Chord system

... with a twist

- Each node gets multiple tokens in the ring

Virtual nodes address:

- Heterogeneity
- Incremental scaling



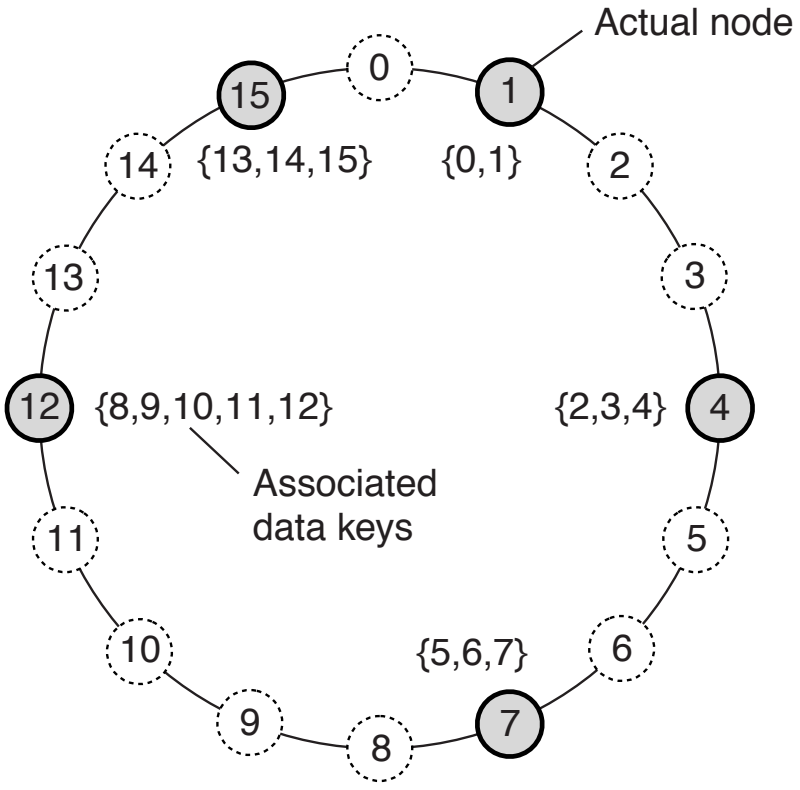
Chord system

... with a twist

- Each node gets multiple tokens in the ring

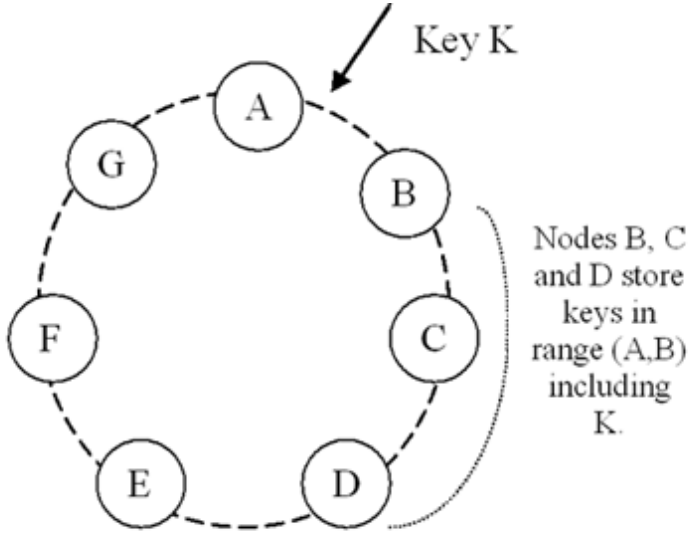
Virtual nodes address:

- Heterogeneity
- Incremental scaling



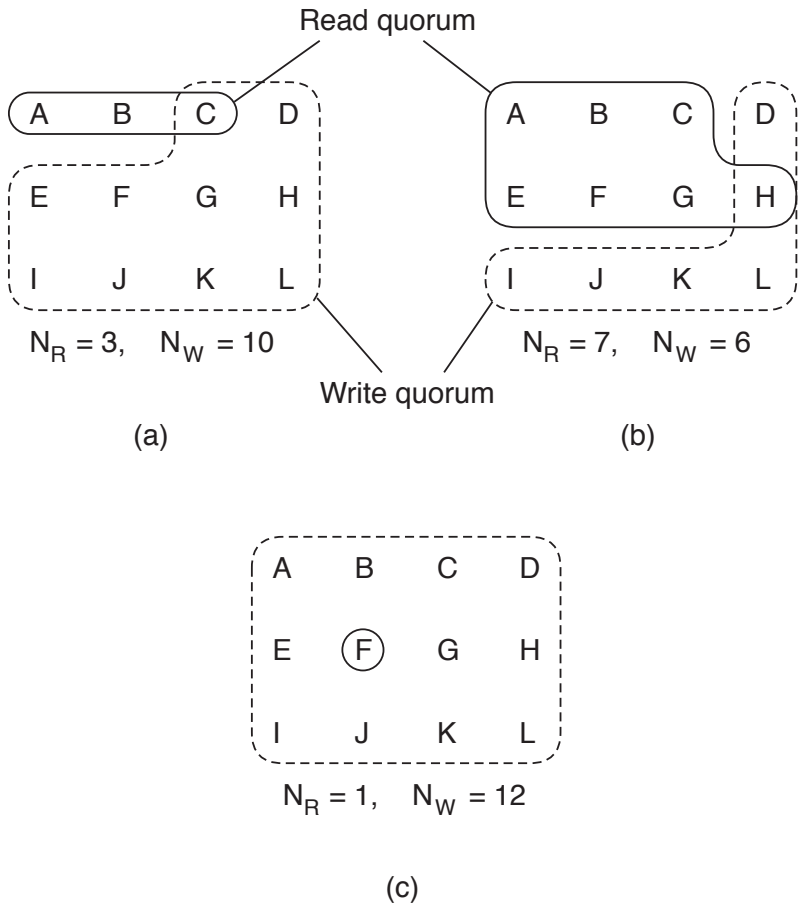
Each object is replicated to N nodes

- N-1 successors to the coordinator



Traditional quorum

- R + W > N
- W > N/2
- Dynamo allows clients to tune N, R and W

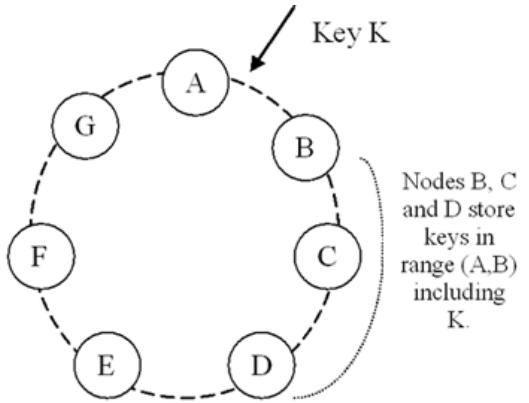


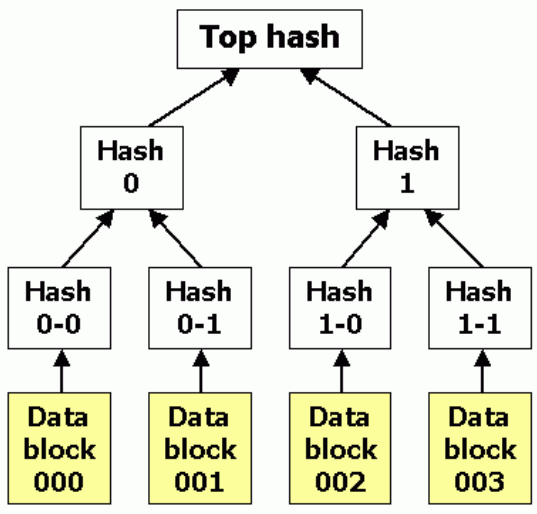
Sloppy-Quorum

- The set of nodes in N may change

Anti-entropy

- Merkel hash trees used to find “out of sync” keys



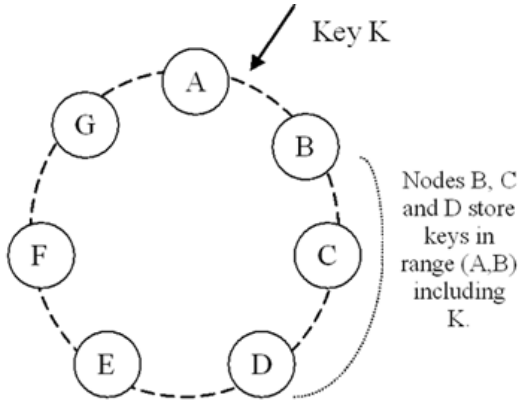


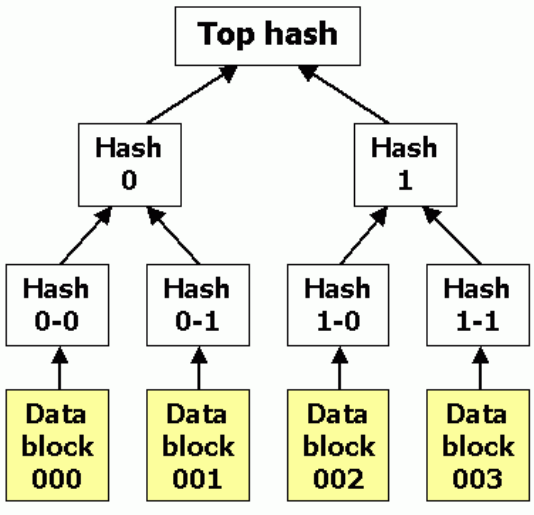
Sloppy-Quorum

- The set of nodes in N may change

Anti-entropy

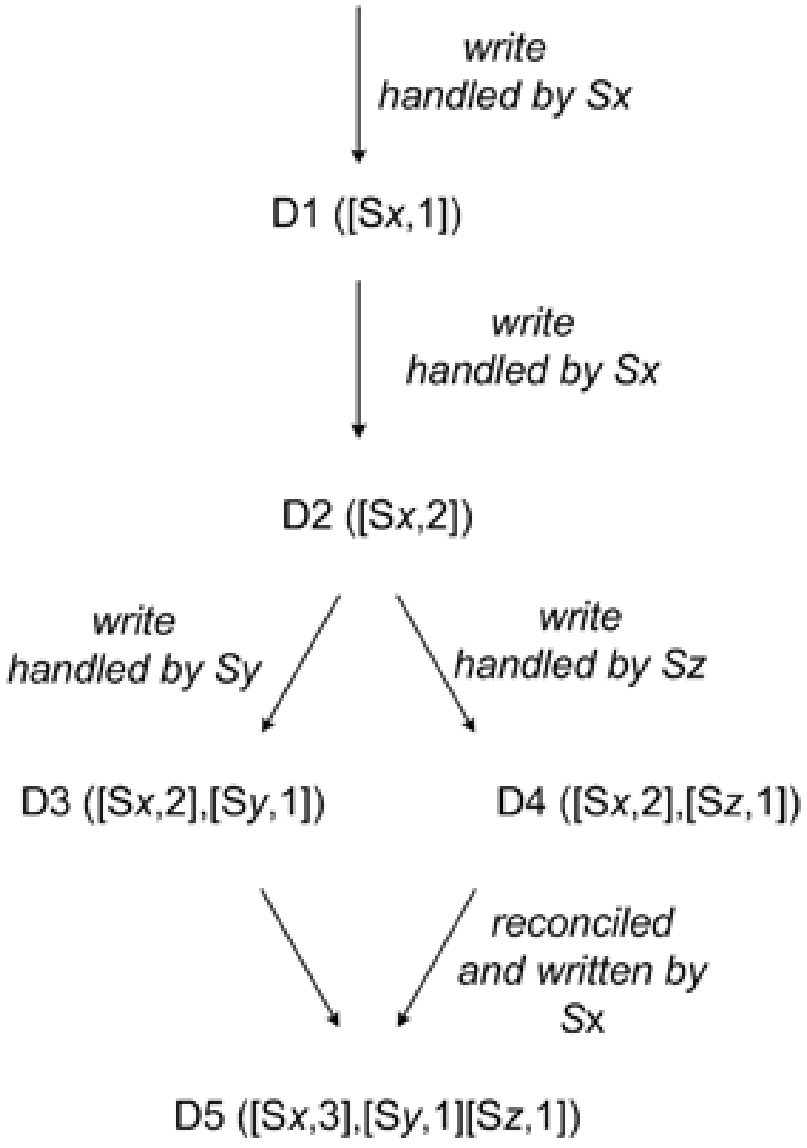
- Merkel hash trees used to find “out of sync” keys





Each object has a Vector clock

- Captures causal ordering



Conflict resolution

- The when
  - During writes?
  - During reads?
- and the who
  - By the data store?
  - By the application?

Conflict resolution

- The when
  - During writes?
  - During reads?
- and the who
  - By the data store?
  - By the application?

When do divergent version arise?

- Failure scenarios
  - Node, data center, network partitions
- Large number of concurrent writes

How frequently are divergent versions created?[1]

Versions	Requests
1	99.94%
2	0.00057%
3	0.00047%
4	0.00009%

[1] 24 hour profile of the Shopping Cart Serivce

When do divergent version arise?

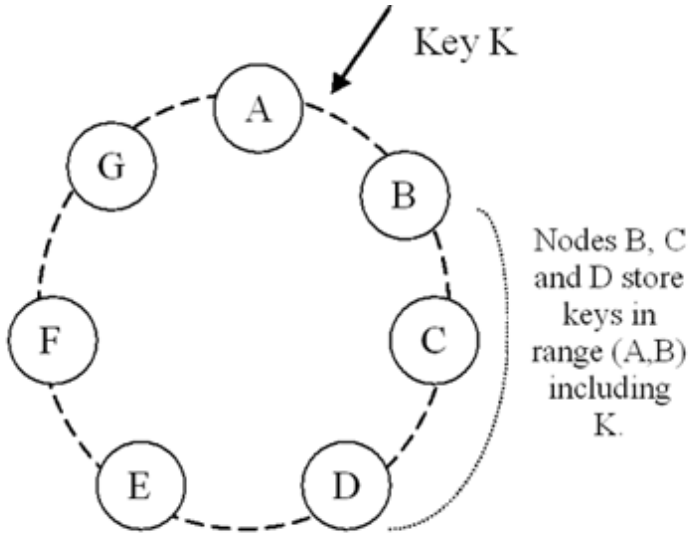
- Failure scenarios
  - Node, data center, network partitions
- Large number of concurrent writes

How frequently are divergent versions created?[1]

Versions	Requests
1	99.94%
2	0.00057%
3	0.00047%
4	0.00009%

[1] 24 hour profile of the Shopping Cart Serivce

Client-driven vs Server-driven coordination



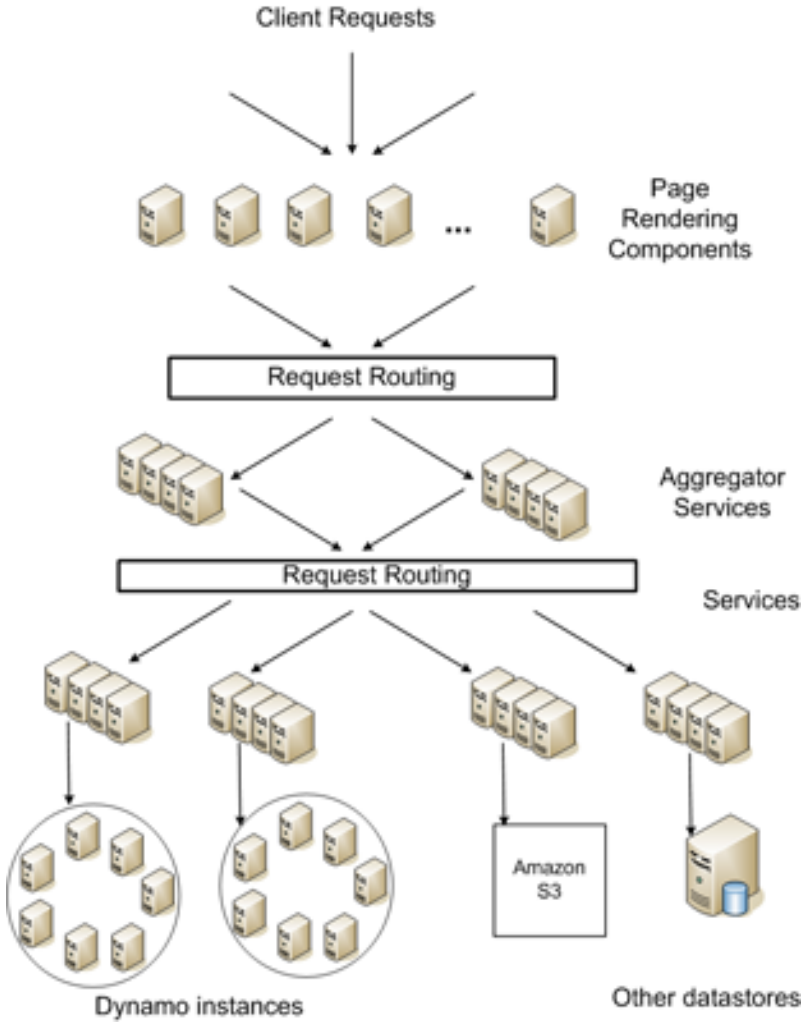
	99.9th percentile read latency	99.9th percentile write latency	Average read latency	Average write latency
Server-driven	68.9	68.5	3.9	4.02
Client-driven	30.4	30.4	1.55	1.9

Times are in milliseconds.

Amazon

- Service architecture
- Decentralized
- Loosely coupled

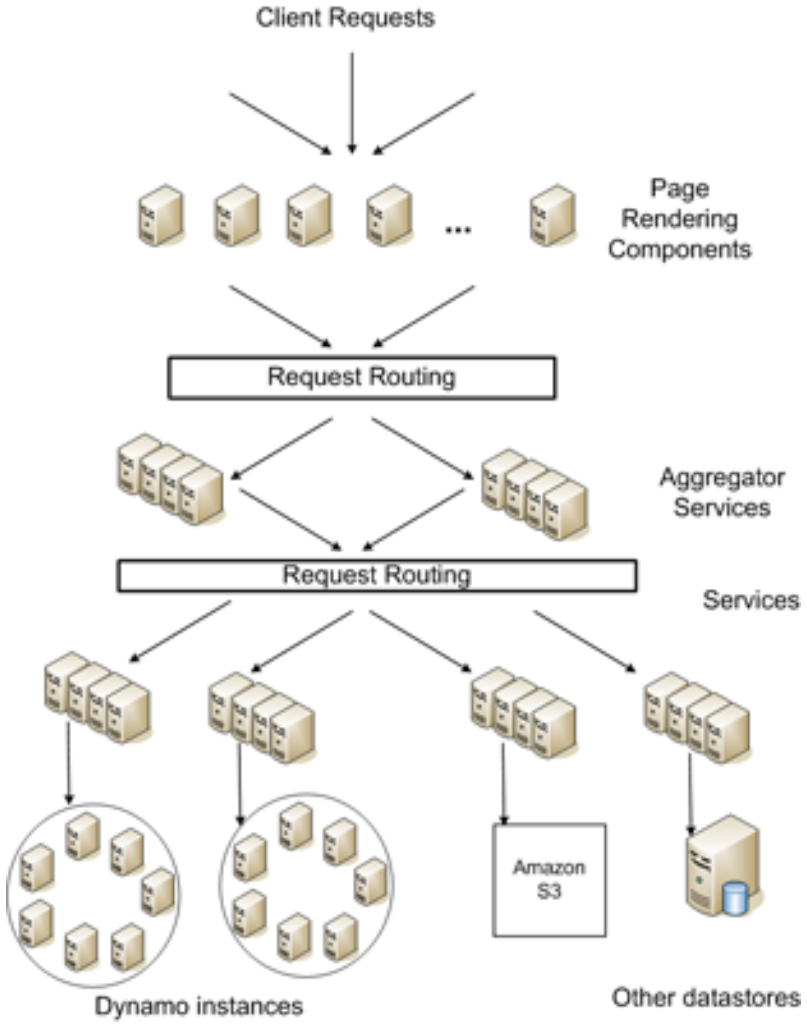
- SLA expressed and measured at the 99.9^{th} percentile
  - Optimizations not focused on averages



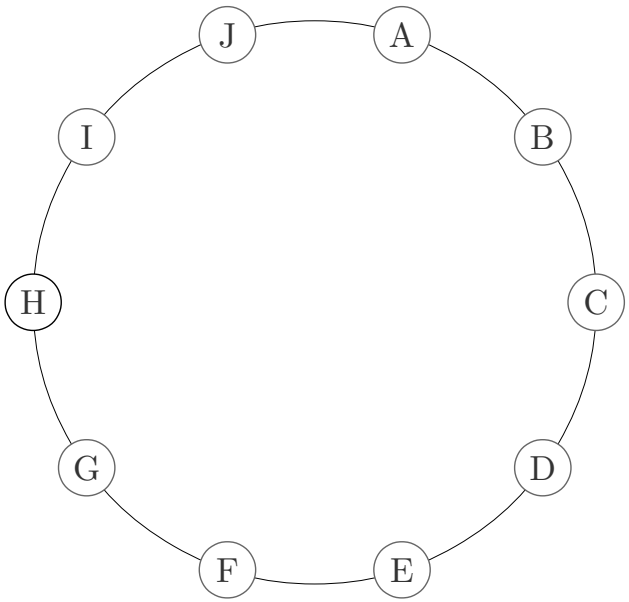
Amazon

- Service architecture
- Decentralized
- Loosely coupled

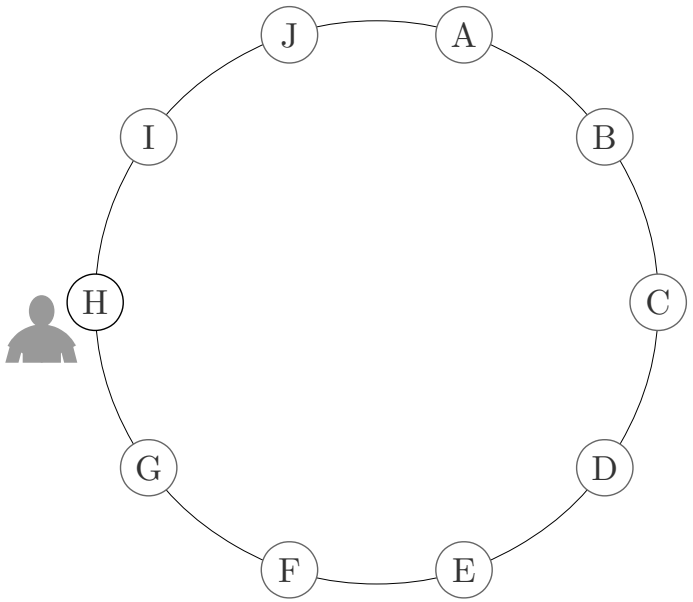
- SLA expressed and measured at the 99.9^{th} percentile
  - Optimizations not focused on averages



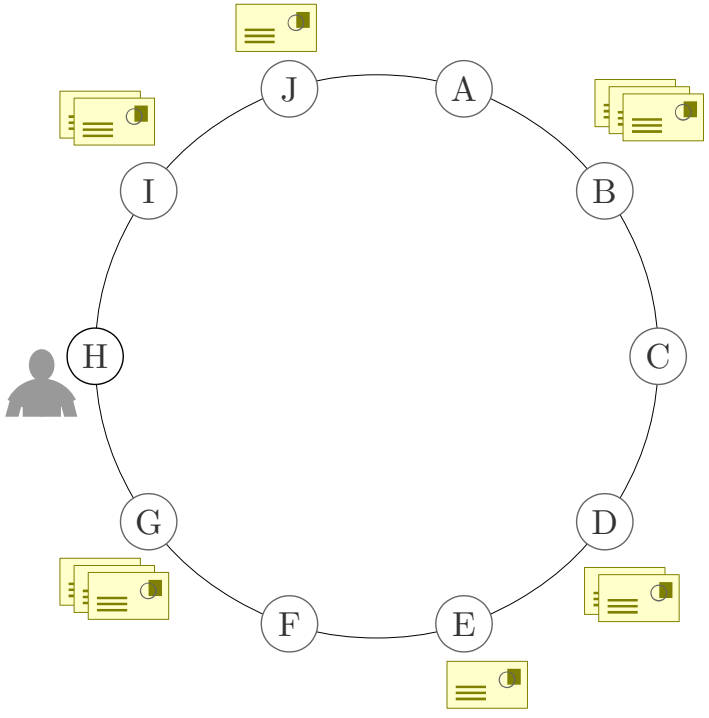
Distributed buckets



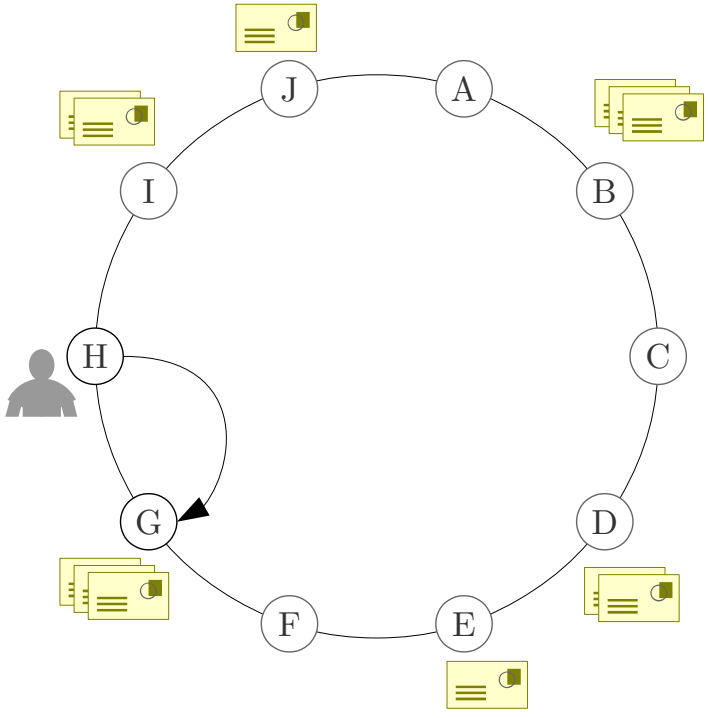
Distributed buckets



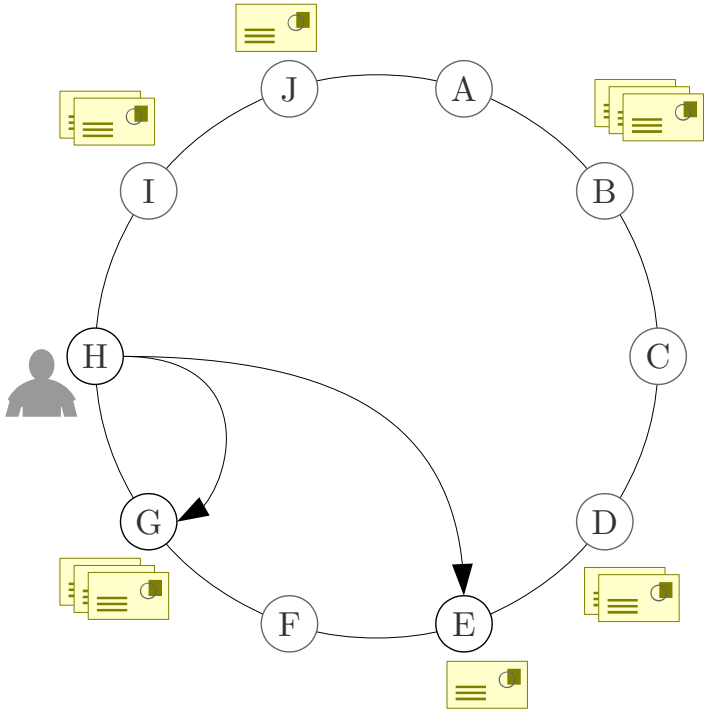
Distributed buckets



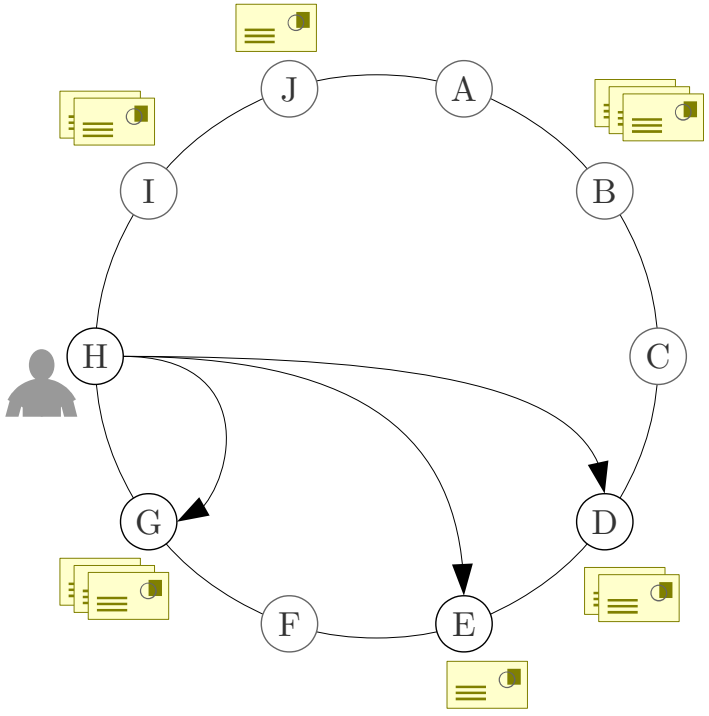
Distributed buckets



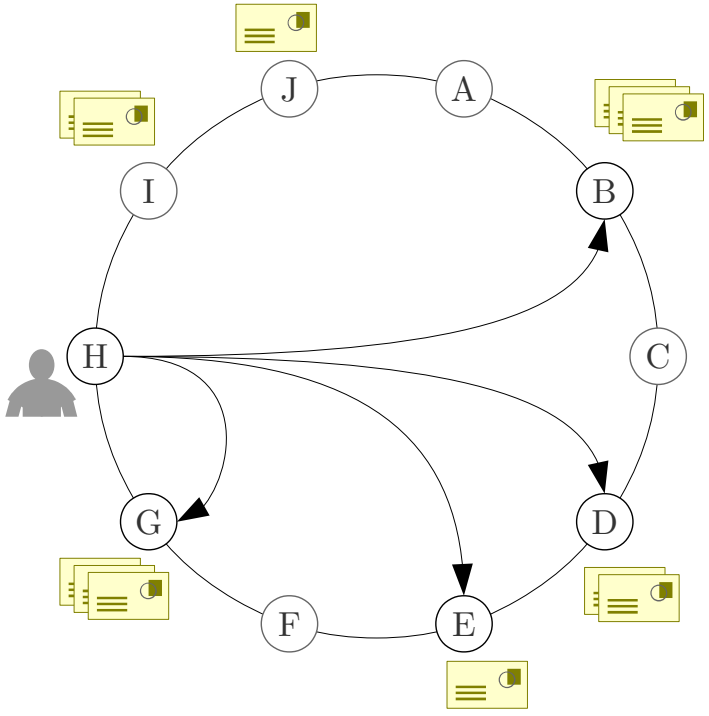
Distributed buckets



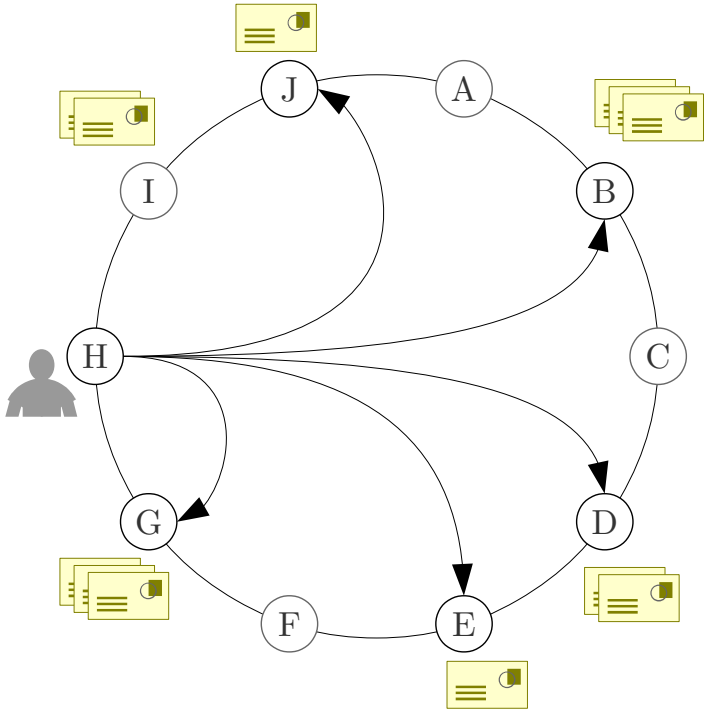
Distributed buckets



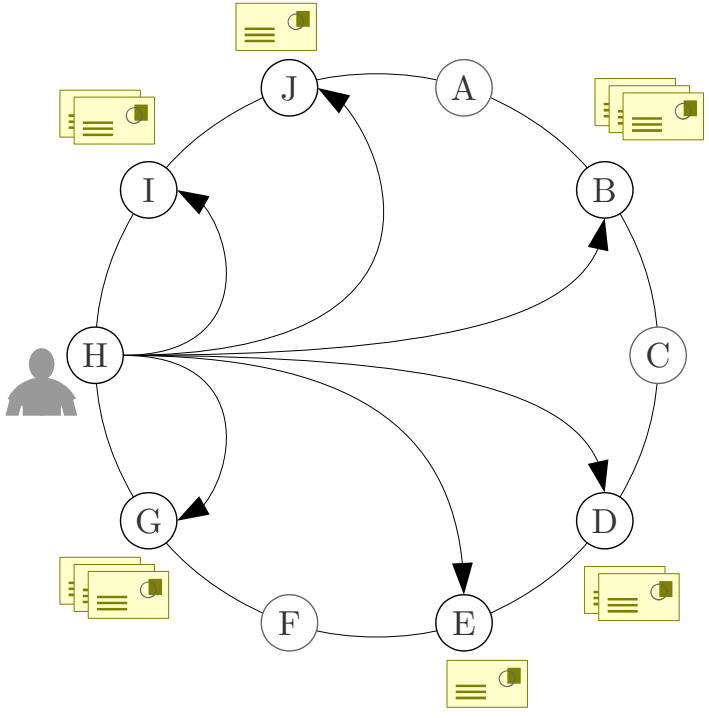
Distributed buckets



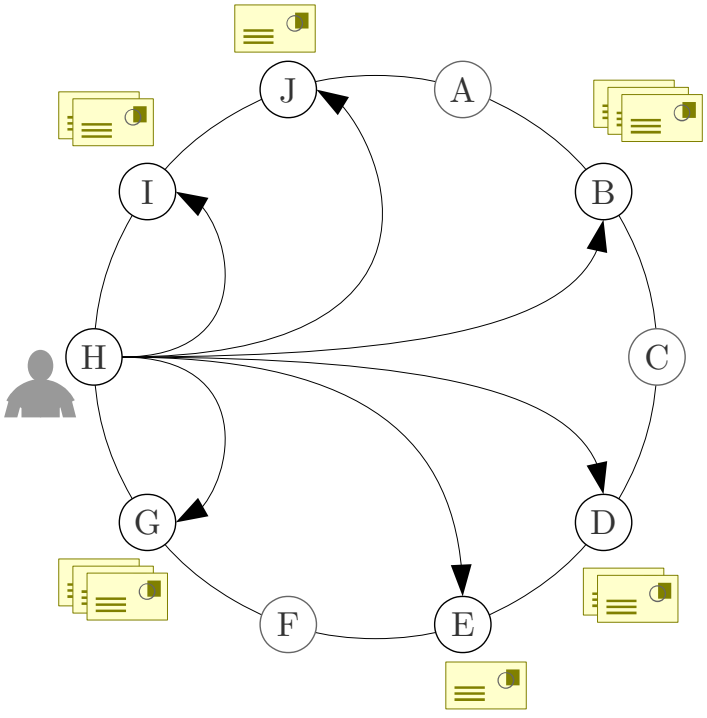
Distributed buckets



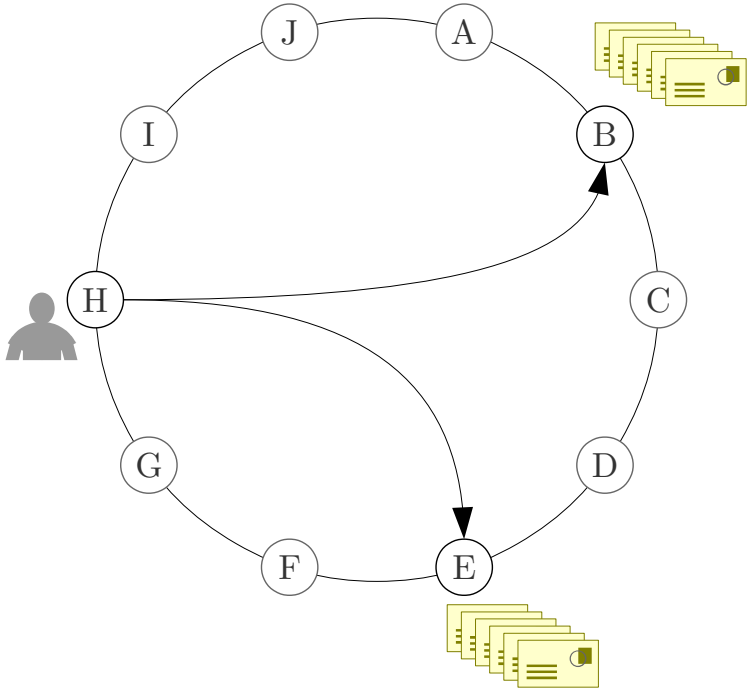
Distributed buckets



Data placement



Random



Clustered