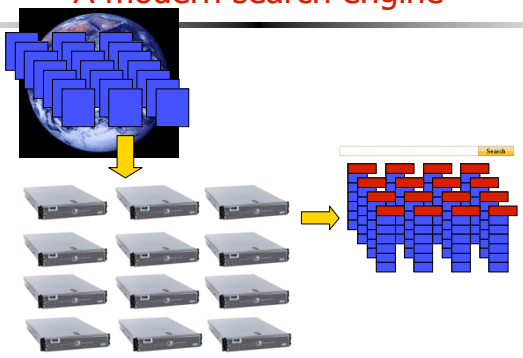
Parallel and distributed databases II

Some interesting recent systems

- MapReduce
- Dynamo
- Peer-to-peer

Then and now

A modern search engine



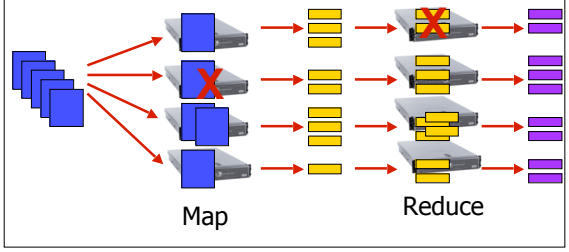
MapReduce

- How do I write a massively parallel data intensive program?
  - Develop the algorithm
  - Write the code to distribute work to machines
  - Write the code to distribute data among machines
  - Write the code to retry failed work units
  - Write the code to redistribute data for a second stage of processing
  - Write the code to start the second stage after the first finishes
  - Write the code to store intermediate result data
  - Write the code to reliably store final result data

MapReduce

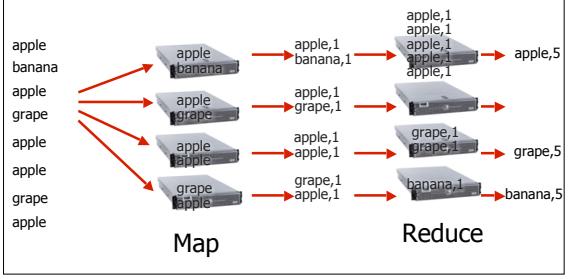
- Two phases
  - Map: take input data and map it to zero or more key/value pairs
  - Reduce: take key/value pairs with the same key and reduce them to a result
- MapReduce framework takes care of the rest
  - Partitioning data, repartitioning data, handling failures, tracking completion…

MapReduce



Example

Count the number of times each word appears on the web



Other MapReduce uses

- Grep
- Sort
- Analyze web graph
- Build inverted indexes
- Analyze access logs
- Document clustering
- Machine learning

Dynamo

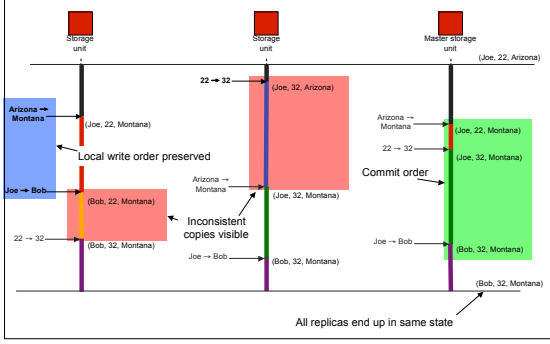
- Always writable data store

- Do I need ACID for this?

Eventual consistency

- Weak consistency guarantee for replicated data
  - Updates initiated at any replica
  - Updates eventually reach every replica
    - If updates cease, eventually all replicas will have same state
  - Tentative versus stable writes
    - Tentative writes applied in per-server partial order
    - Stable writes applied in global commit order
  - Bayou system at PARC

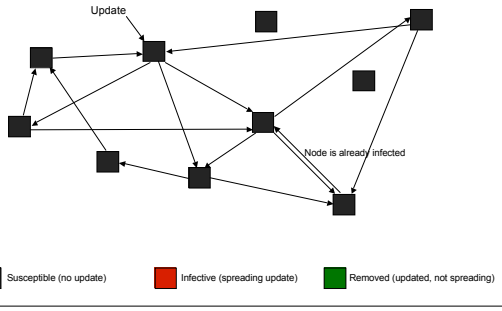
Eventual consistency



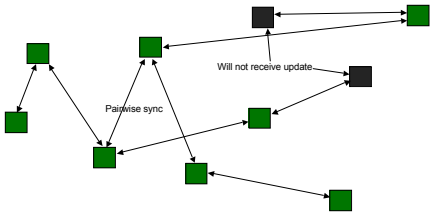
Mechanisms

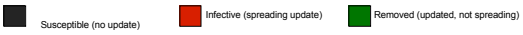
- Epidemics/rumor mongering
  - Updates are "gossiped" to random sites
  - Gossip slows when (almost) every site has heard it
- Anti-entropy
  - Pairwise sync of whole replicas
    - Via log-shipping and occasional DB snapshot
  - Ensure everyone has heard all updates
- Primary copy
  - One replica determines the final commit order of updates

Epidemic replication



Anti-entropy

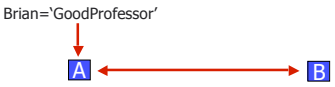




What if I get a conflict?

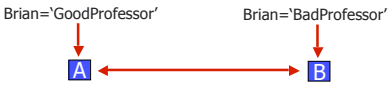
- How to detect?
  - Version vector: (A's count, B's count)

What if I get a conflict?



- How to detect?
  - Version vector: (A's count, B's count)
  - Initially, (0,0) at both
  - A writes, sets version vector to (1,0)
  - (1,0) **dominates** B's version (0,0)
  - *No conflict*

What if I get a conflict?



- How to detect?
  - Version vector: (A's count, B's count)
  - Initially, (0,0) at both
  - A writes, sets version vector to (1,0)
  - B writes, sets version vector to (0,1)
  - Neither vector dominates the other
  - ***Conflict!!***

How to resolve conflicts?

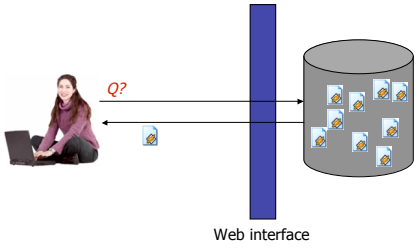
- Commutative operations: allow both
  - Add "Fight Club" to shopping cart
  - Add "Legends of the Fall" shopping cart
  - Doesn't matter what order they occur in
- Thomas write rule: take the last update
  - That's the one we "meant" to have stick
- Let the application cope with it
  - Expose possible alternatives to application
  - Application must write back one answer

Peer-to-peer

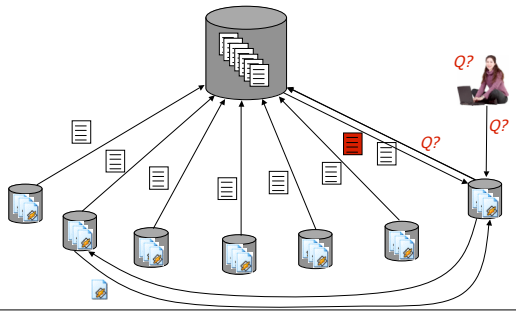
- Great technology
- Shady business model
- Focus on the technology for now

Peer-to-peer origins

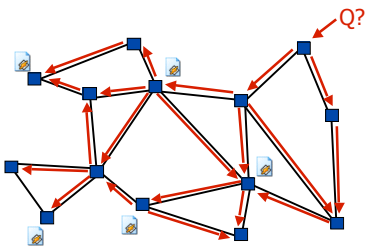
- Where can I find songs for download?



Napster



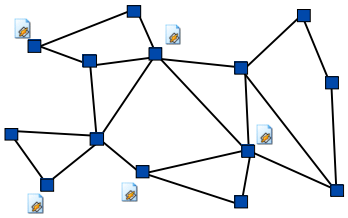
Gnutella



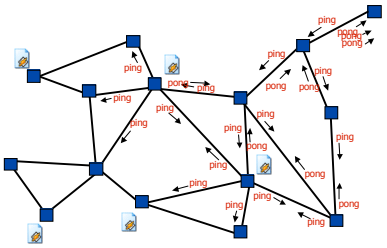
Characteristics

- Peers both generate and process messages
  - Server + client = "servent"
- Massively parallel
- Distributed
- Data-centric
  - Route queries and data, not packets

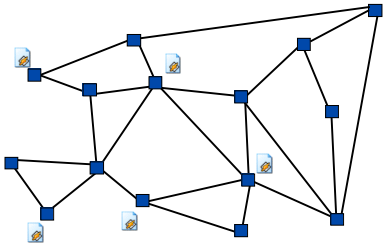
Gnutella



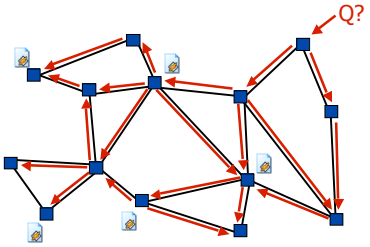
Joining the network



Joining the network

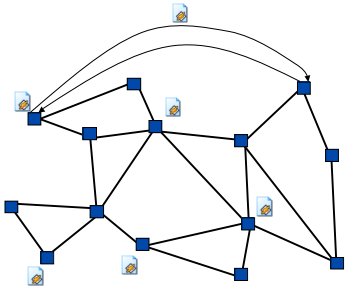


Search

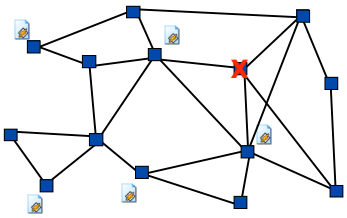


TTL = 4

Download

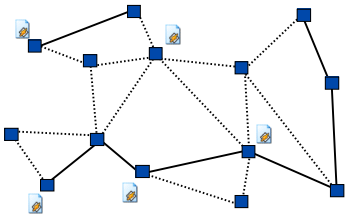


Failures



Scalability!

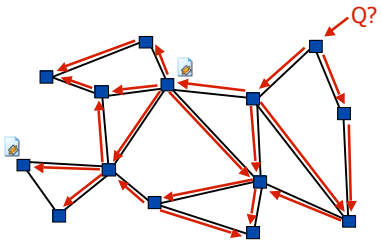
- Messages flood the network
- Example: Gnutella meltdown, 2000



How to make more scalable?

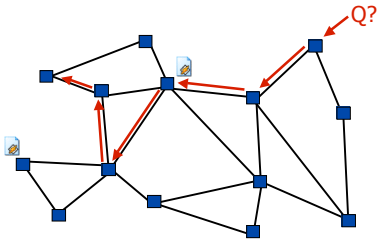
- Search more intelligently
- Replicate information
- Reorganize the topology

Iterative deepening



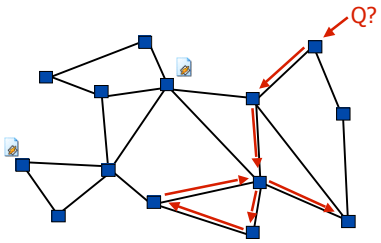
Yang and Garcia-Molina 2002, Lv et al 2002

Directed breadth-first search



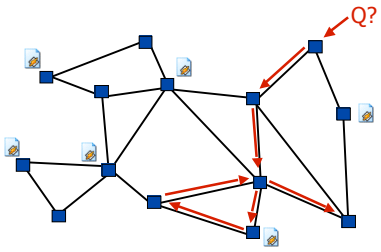
Yang and Garcia-Molina 2002

Random walk



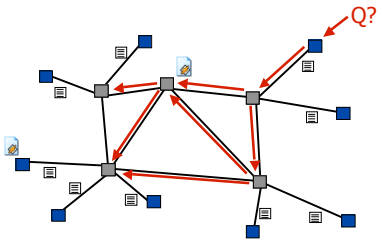
Adamic et al 2001

Random walk with replication



Cohen and Shenker 2002, Lv et al 2002

Supernodes



Kazaa, Yang and Garcia-Molina 2003

Some interesting observations

- Most peers are short-lived
  - Average up-time: 60 minutes
  - For a 100K network, this implies churn rate of 1,600 nodes per minute
  - Saroiu et al 2002
- Most peers are "freeloaders"
  - 70 percent of peers share no files
  - Most results come from 1 percent of peers
  - Adar and Huberman 2000
- Network tends toward a power-law topology
  - Power-law: n^{th} most connected peer has k/n^{α} connections
  - A few peers have many connections, most peers have few
  - Ripeanu and Foster 2002

"Structured" networks

- Idea: form a structured topology that gives certain performance guarantees
  - Number of hops needed for searches
  - Amount of state required by nodes
  - Maintenance cost
  - Tolerance to churn
- Part of a larger application
  - Peer-to-peer substrate for data management

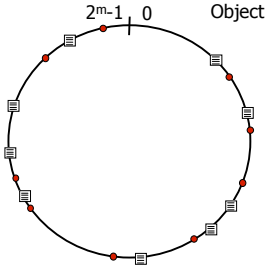
Distributed Hash Tables

- Basic operation
  - Given key K, return associated value V
- Examples of DHTs
  - Chord
  - CAN
  - Tapestry
  - Pastry
  - Koorde
  - Kelips
  - Kademlia
  - Viceroy
  - Freenet
  - …

Chord

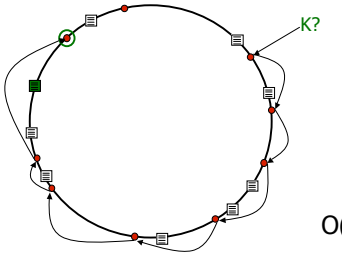
Node ID = Hash(IP)

Object ID = Hash(Key)



Stoica et al 2001

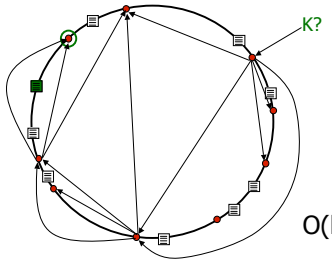
Searching



O(N)

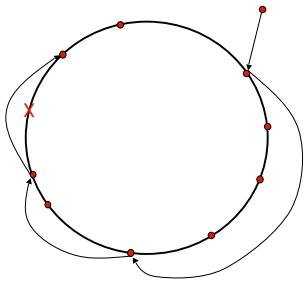
Better searching

Finger table: i^{th} entry is node that succeeds me by at least 2^{i-1}, m entries total

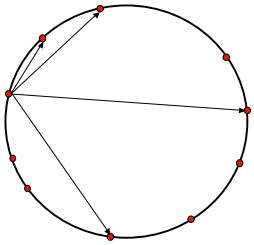


O(log N)

Joining

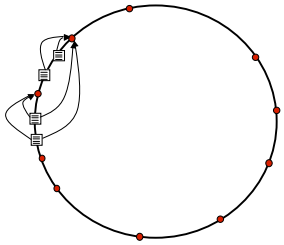


Joining

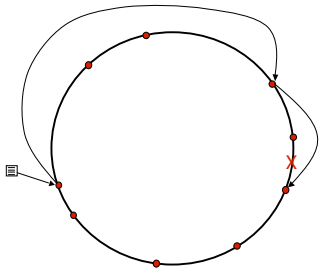


O(log^2 N)

Joining



Inserting



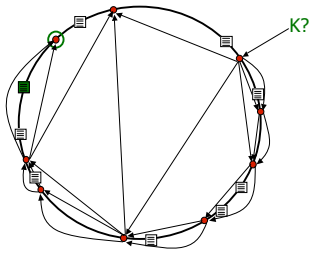
What is actually stored?

- Objects
  - Requires moving object
  - Load balancing for downloads
    - Unless some objects are "hot"
- Pointers to original objects
  - Object can stay in original location
  - Nodes with many objects can cause load imbalance
- Chord allows either option

Good properties

- Limited state
  - Finger table size: m = O(log n)
- Bounded hops
  - O(log n) for search, insert (w.h.p.)
- Bounded maintenance
  - $O(\log^2 n)$
- Robust

What if finger table is incomplete?



Issues

- Partitions
- Malicious nodes
- Network awareness