Distributed Pagerank for P2P Systems

Karthikeyan Sankarlingam,

Simha Sethumadhavan, and James C. Browne

The University of Texas at Austin

Department of Computer Sciences

Contributions

- Distributed computation of Pageranks based on asynchronous iteration
  - Application in P2P systems
  - Application on Internet scale
- Practical keyword search for P2P systems
- Very large scale asynchronous iteration computation

Overview

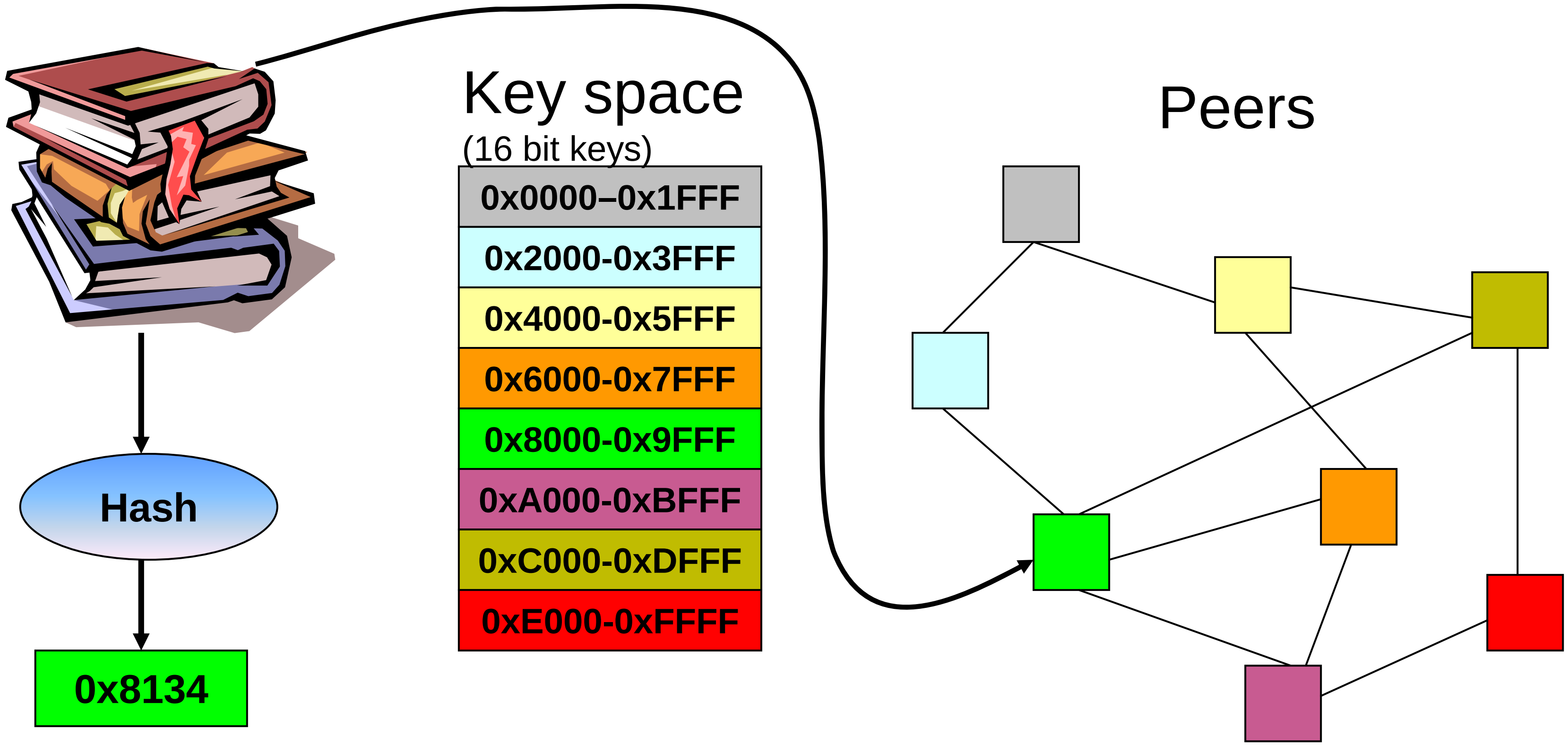
- *Motivation:* Keyword search for P2P systems
  - P2P system overview
  - State of art in keyword search
- *Approach and Solution*
  - Pageranks for P2P systems
  - Distributed computation of pageranks
  - Incremental retrieval of documents
  - Performance results

- Distributed computation of pageranks on the Internet

Peer to Peer (P2P) Systems

- P2P systems can be effective distributed storage systems
  - Efficient retrieval
  - Efficient search
- Retrieval
  - Distributed Hash Tables: Chord, CAN, Pastry, Freenet
  - Unstructured P2P systems: Gnutella, Morpheus, Kazaa
- Characteristics
  - Distributed storage, no centralized server
  - Peer-to-peer communication
  - Dynamic effects – peers enter and leave frequently

P2P Systems



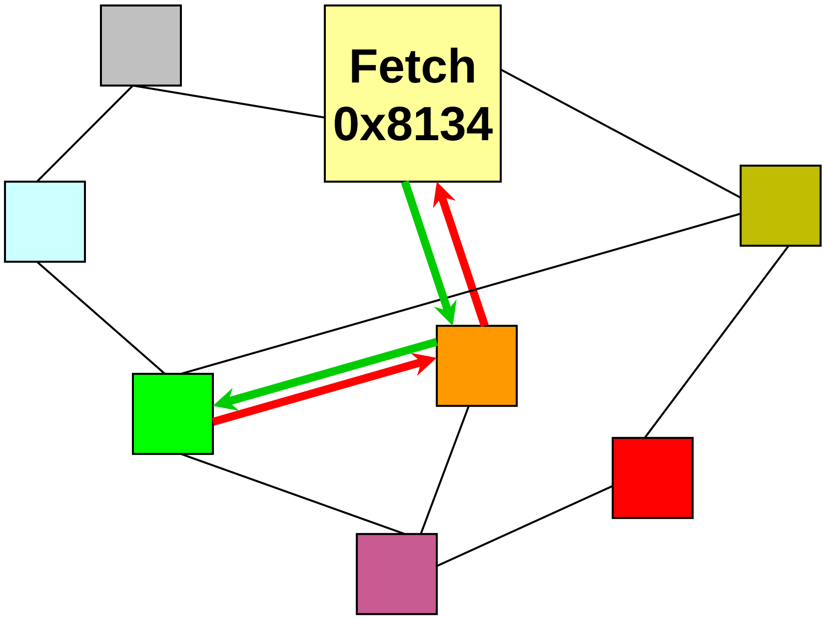
- Distributed hash tables
- Routing

P2P systems: Retrieval

Key space

0x0000–0x1FFF
0x2000-0x3FFF
0x4000-0x5FFF
0x6000-0x7FFF
0x8000-0x9FFF
0xA000-0xBFFF
0xC000-0xDFFF
0xE000-0xFFFF

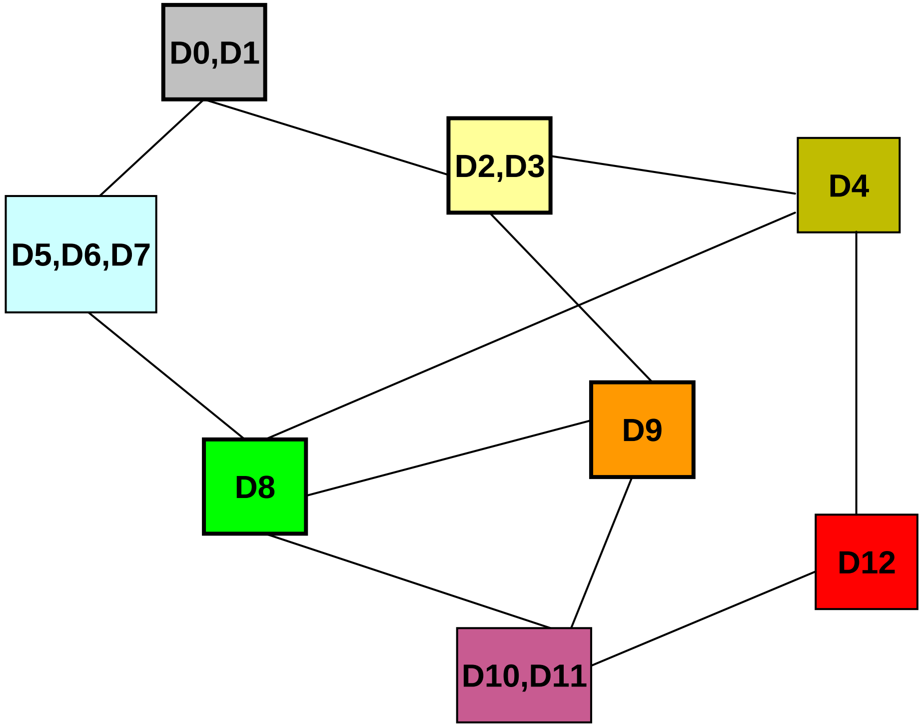
Peers



P2P systems: Search

- State of the art
  - Index based keyword search *[Reynolds and Vahdat, Gnawali]*
  - Document vectors *[Kronofol]*
  - Combinations based on these
- Problem
  - Retrieval – too many responses
  - No easy way to estimate relevance

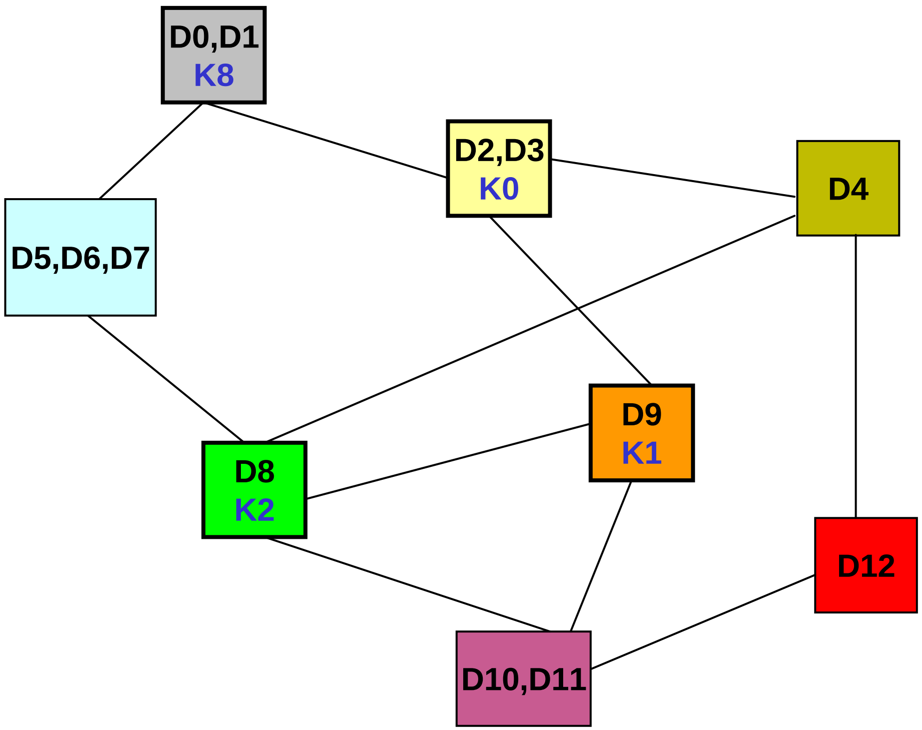
Index based keyword search



Centralized Index

Keyword	List of Doc Ids (keys)
tree	D0,D1,D2,D3
oak	D0,D1,D9
spider	D12,D11
…	
linux	D12,D11

Index based keyword search



Hashed Keyword	List of Doc Ids (keys)
K0(tree)	D0,D1,D2,D3
K1(oak)	D0,D1,D9
K2(spider)	D12,D11
…	
K8(linux)	D12,D11

- **Hash, distribute and embed the index in P2P system!**

P2P Systems: Search

- State of art
  - Index based keyword search *[Reynolds and Vahdat, Gnawali]*
  - Document vectors *[Kronofol]*
  - Combinations based on these
- Problem
  - Retrieval – too many responses
  - No easy way to estimate relevance

Solution

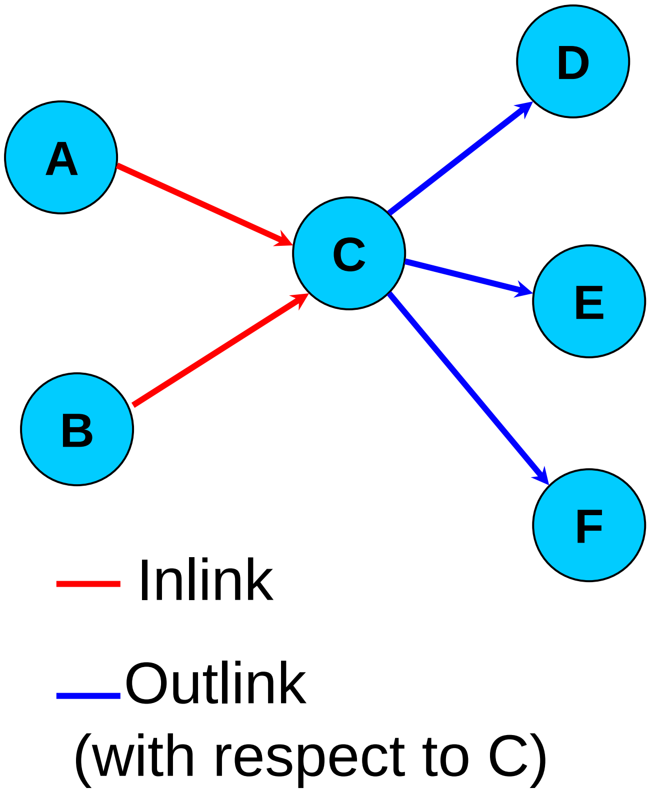
- Google's Pagerank!
- Apply Pagerank in a P2P environment
  - Give every document in the P2P system a rank
  - Use link structure
  - Incremental retrieval based on pageranks

Google's Computation of Pagerank

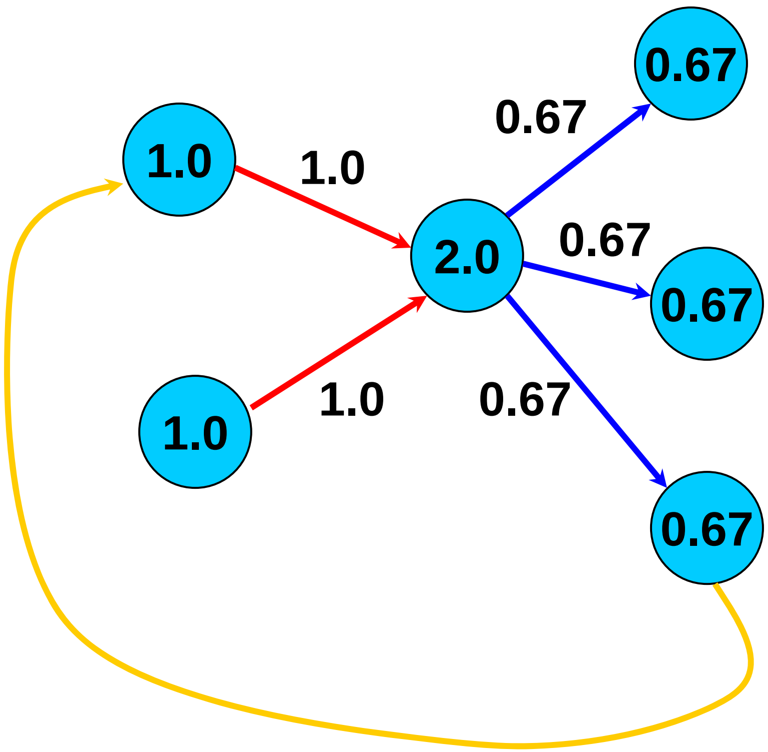
- Centralized solution
  - Uses a centralized crawler updated every 4 weeks
  - Computation farm solving a 3 billion order matrix problem
  - Computation time of 6 to 7 days
  - Methods to accelerate this have been proposed *[Kamvar et. al]*
- Challenge for a P2P implementation
  - Files are distributed
  - No crawler on P2P systems
  - No centralized computation possible
  - Peers keep entering and leaving

Pagerank

- Assign a numeric rank to every page
- Document link structure is the key



Pagerank contd.

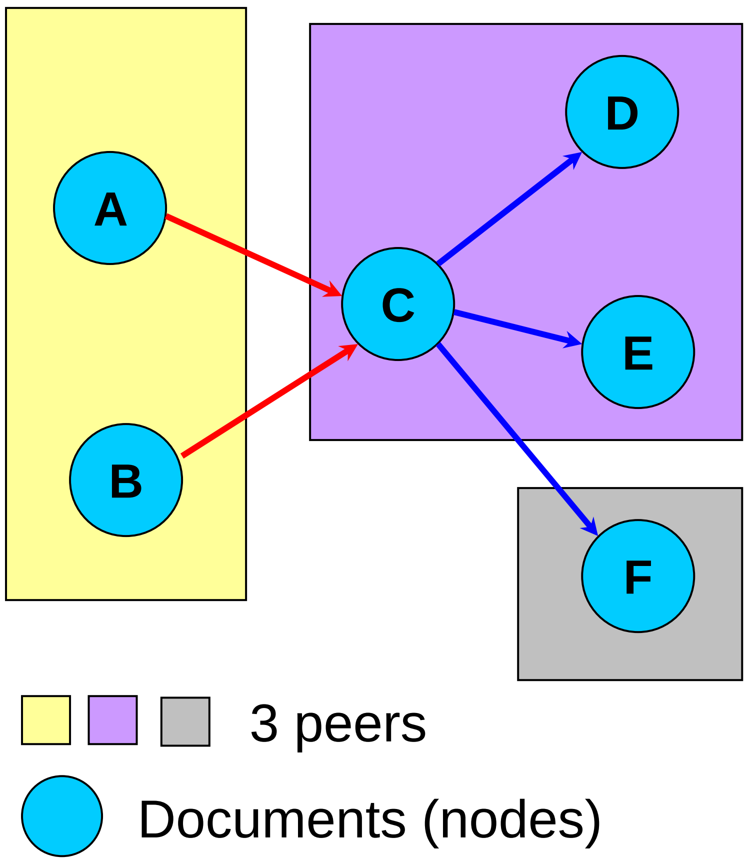


- Every page contributes equally to all its outlinks
- Pagerank of a page = sum of inlink weights
- Web graph has backedges
- Pagerank is computed iteratively
- Mathematical formulation:

$$R_{i+1} = AR_i$$

Distributed Pagerank

- Compute pagerank locally at each peer node
- Send pagerank updates to linked documents (on other peers)
- Stop when each local pagerank “converges”



Why does this process work? Asynchronous Iterations

- Pagerank is an eigenvalue computation problem *[Page et. al, Haveliwala]*
- Link matrix is sparse and diagonally dominant

- Asychronous Iterations *[Chazan & Miranker, and others]*
- Peers act as simple state machines exchanging messages

Integration with P2P systems

- **Storage:** Augment P2P system to store a rank for every document
- **Computation:** Peers must execute the distributed pagerank computation algorithm
- **Communication:** Pagerank update messages are routed based on linked document's key

- **Caching:** Optimization to save routed traffic
  - Route first message using P2P layer
  - Cache IP address for that key at sender
  - Deliver subsequent messages point to point

Dynamic systems

- Peer joins and leaves
  – Use transport layer to detect if peer unavailable
  – Buffer update messages if peer unavailable
  – Periodically retry until peer comes back
- Document insertion and deletion
  – New documents are initialized with a pagerank
  – Deleted documents send pagerank update messages with negative pagerank
- Incremental and continuously updated pageranks

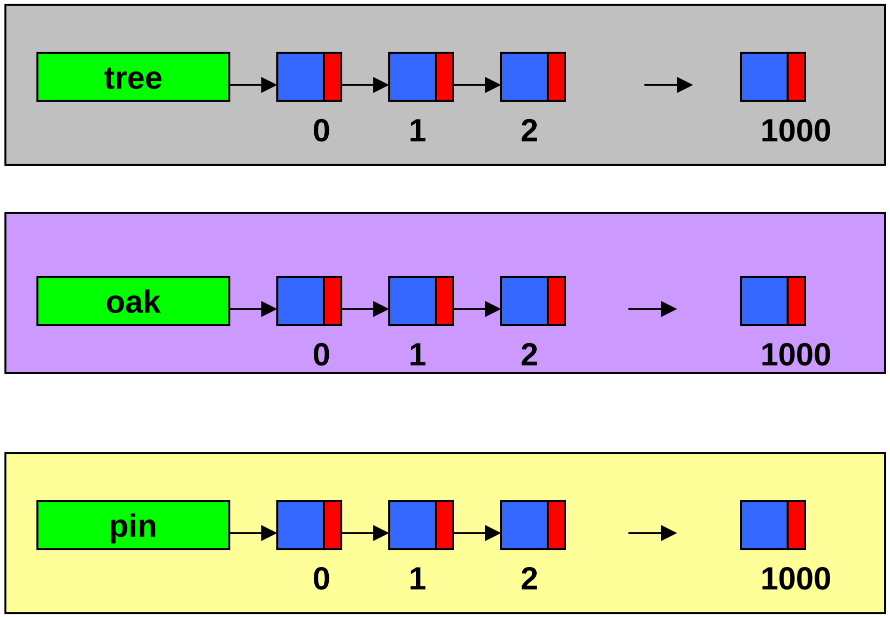
Integration with P2P search

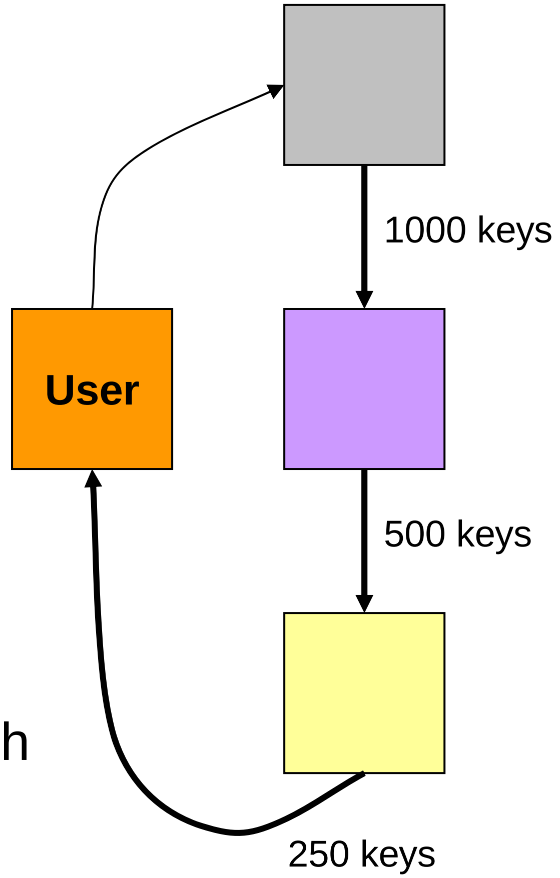
- DHT systems
  - Augment index with a pagerank field
  - Return results sorted by pagerank
  - Nodes update index with pagerank when they converge
- FASD/Freenet systems
  - Forward based on *document closeness* and pagerank

Hashed Keyword	List of Doc Ids (keys)
K0(tree)	D0{R0},D1{R1},D2{R2},D3{R3}
K1(oak)	D0{R0},D1{R1},D9{R9}
K2(spider)	D12{R12},D11{R11}
...	
K8(linux)	D12{R12},D11{R11}

{Rxx} - Pageranks

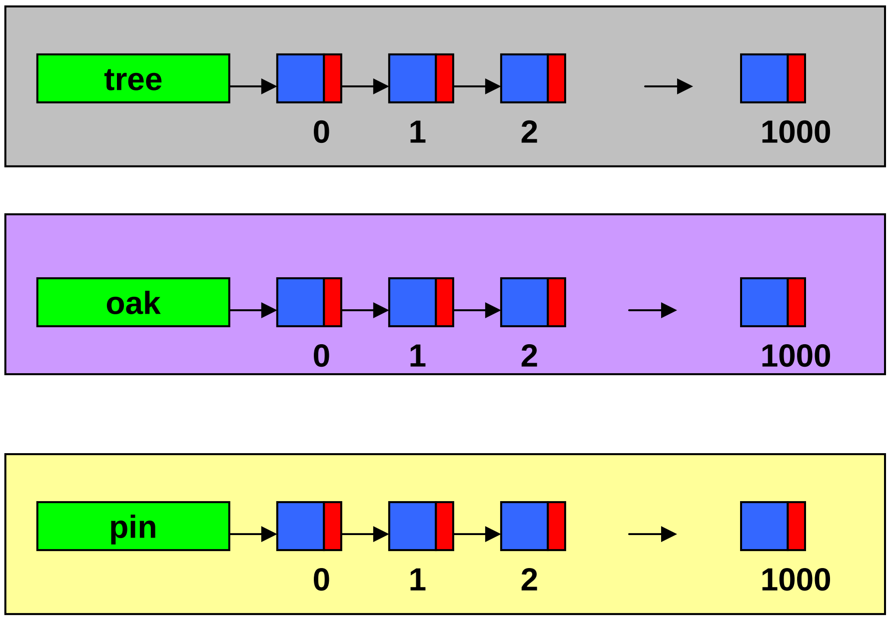
Multi-word search

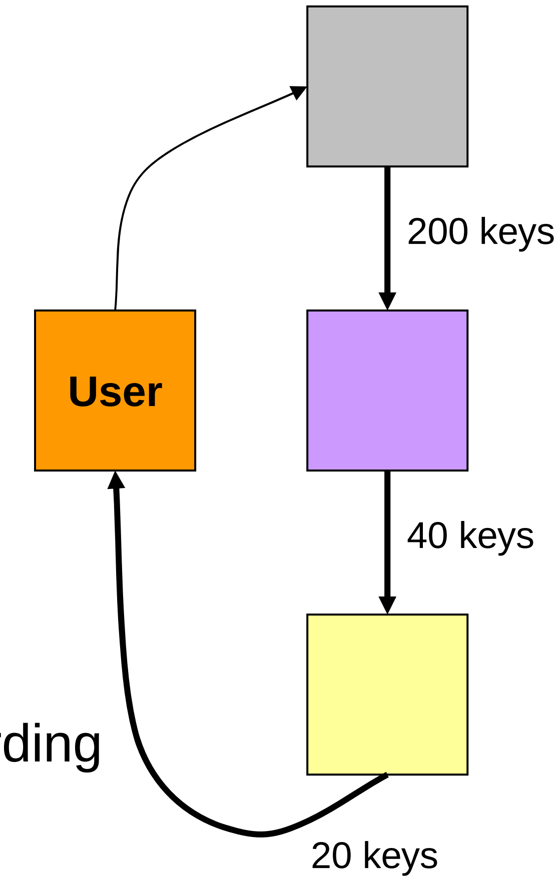




- Example Query: *tree & oak & pin*
- Many keys transferred, leading to high network traffic
- No ranking scheme

Incremental search





- Example Query: *tree & oak & pin*
- Traffic reduction: Incremental forwarding
- Quality of hits: Relevance sorting

Results

- Modeling
  - 10K, 100K, 500K, 5M document sets
  - 500 peer network
  - Simple network transfer model
  - Power law distribution for link structure:
    # nodes with degree i α *1 / i^k [Broder et. al]*

- Evaluation parameters
  - *Convergence:* How many passes?
  - *Quality of pagerank:* Error relative to a centralized scheme
  - *Message traffic:* Number of pagerank update messages
  - *Execution time and Scalability*

Results

Convergence	1. Fast convergence: ~ 100 iterations 2. 99% of documents converge to within 1% in 10 iterations
Quality of Pagerank	Very high, Over 99% have very small errors, max error typically < 0.1%
Message traffic	1. 30 msgs/doc for a 0.2 error threshold 2. 100 msgs/doc for a 10^{-6} error threshold 3. Msgs/doc. independent of # docs 4. Traffic grows logarithmically with error threshold

Results

Execution Time	Dominated by network speed		
	Error threshold	Slow n/w(32 Kb/sec)	Fast n/w (200 Kb/sec)
	0.2 10^{-3} 10^{-6}	33.7 hrs 87.9 hrs 117 hrs	5.4 hrs 14.1 hrs 18.7 hrs
Scalability	1. Convergence, quality and messages/doc independent of size 2. Execution times grows logarithmically with size		

Results: Incremental Search

- We built our own document set
- 2-word and 3-word queries synthesized using frequent terms
- 10X reduction in network traffic for 2-word queries
- 6X reduction in network traffic for 3-word queries

Conclusions

- Distributed computation of Google Pagerank
- First document ranking scheme for P2P networks
- Major benefits for keyword search
- **Performance and Scalability** demonstrated for P2P systems

P2P Internet search engine?

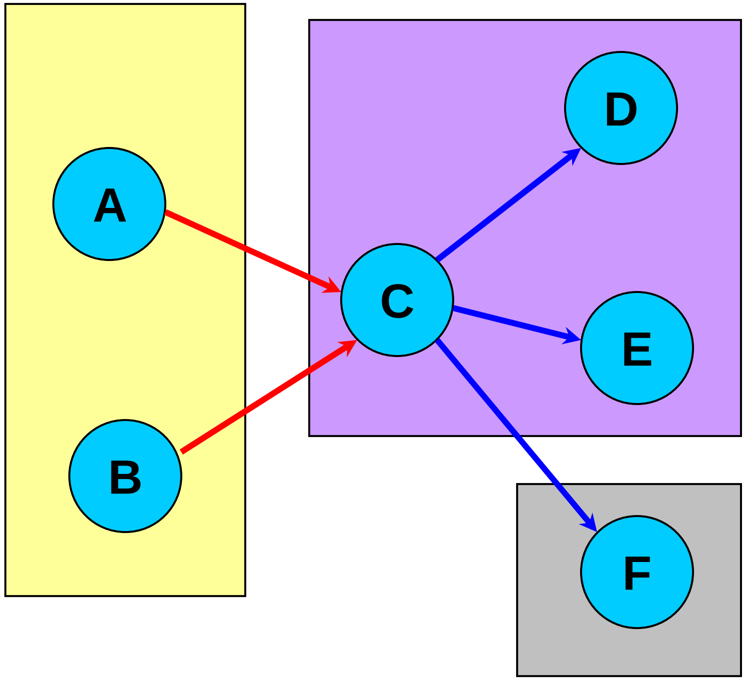
- P2P computation of Pagerank of Internet documents
  - Web servers acts as peers, exchange messages and compute pagerank
  - Pagerank becomes a “free” public commoditiy
  - Will this work?
    - With a T3 link between web space providers, 3 billion node graph can be computed in 35 days.
    - No re-crawls required!
    - Document inserts and deletes are automatically handled
- How to build a distributed Internet scale keyword index?
  - Web server implementation?

Future Work

- Implement Pagerank on a P2P system
- Use link structure to map documents
- Peer-to-peer chaotic iterations solutions should work in other domains
- Explore Internet scale application

Questions

Distributed Pagerank



Time = 0

Set A,B = 1.0

Set C,D,E = 1.0

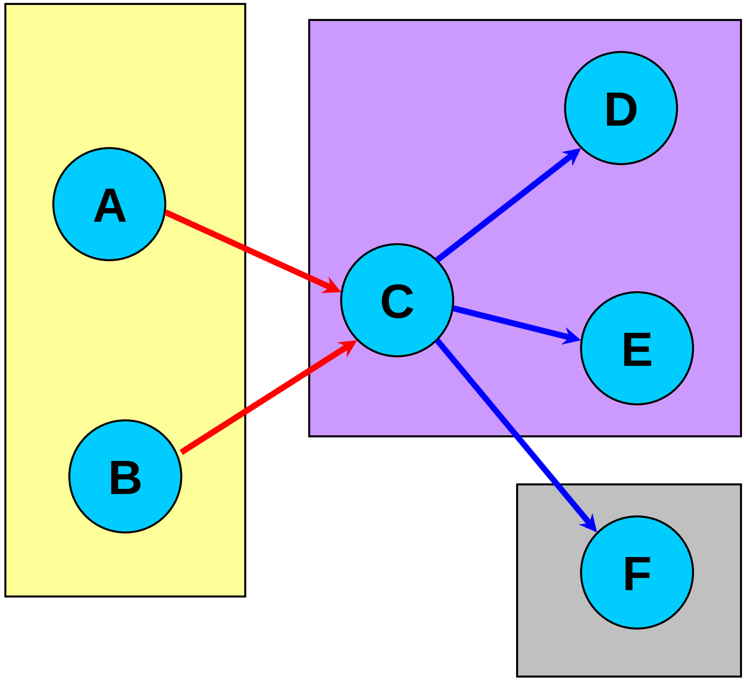
Set F = 1.0

Send updates:

From A, B to C

From C to F

Distributed Pagerank



Time = 5
C,F receive updates
Recompute page ranks

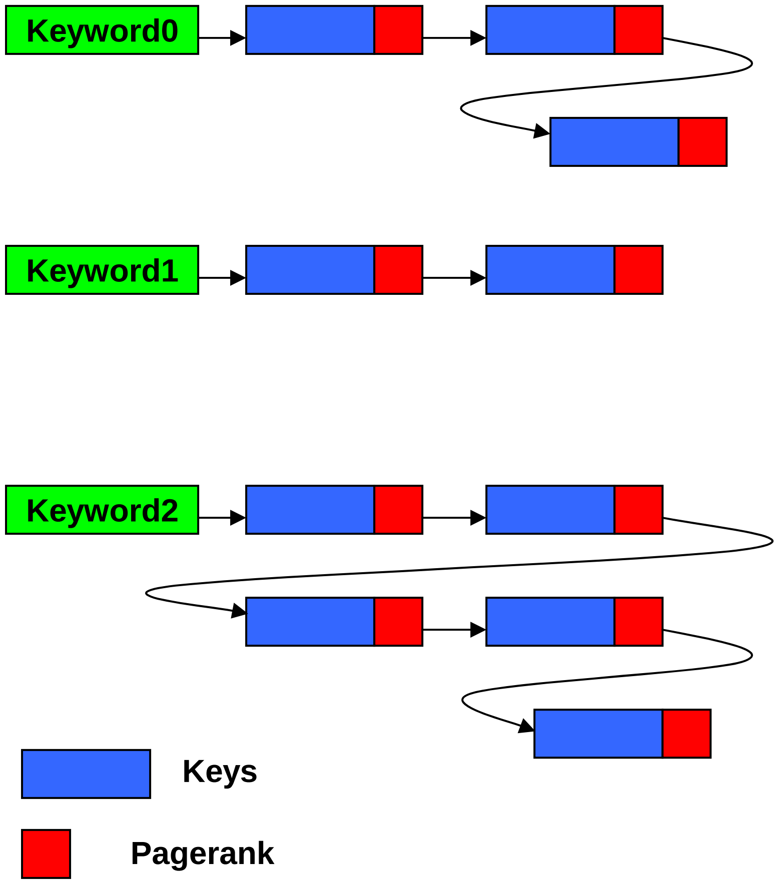
Send updates:
From C to F

Time = 17
F receives updates
Recompute page ranks

No more updates. STOP.

Integration with P2P search

- DHT systems like Chord, CAN, Pastry
  - Augment index with a pagerank field
  - Return results sorted by pagerank
- FASD/Freenet systems
  - Forward based on *document closeness* and pagerank



Convergence	1. Fast convergence: ~ 100 iterations 2. 99% of documents converge to within 1% in 10 iterations
Quality of Pagerank	Very high, Over 99% have very small errors, max error typically < 0.1%
Message traffic	1. Low: 30 (0.2 error) to 100 (10^{-6} error) msgs/doc 2. Msgs/doc. independent of # docs 3. Traffic grows logarithmically with error threshold
Execution Time	1. Low: 14 to 90 hrs based on n/w speed 2. Dominated by network speed
Scalability	1. Convergence, quality and messages/doc independent of size 2. Execution times grows logarithmically with size