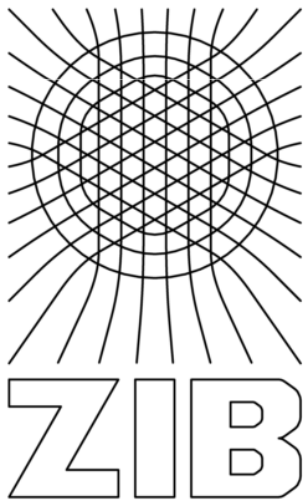


Scalable Wikipedia with Erlang



Thorsten Schütt, Florian Schintke , Alexander Reinefeld

Zuse Institute Berlin (ZIB)

onScale solutions

Scaling

Web 2.0 Hosting

1. Step



Clients

- Single Server
 - Webserver
 - DB Server



2. Step



Clients

- Single Webserver



- Single DB server

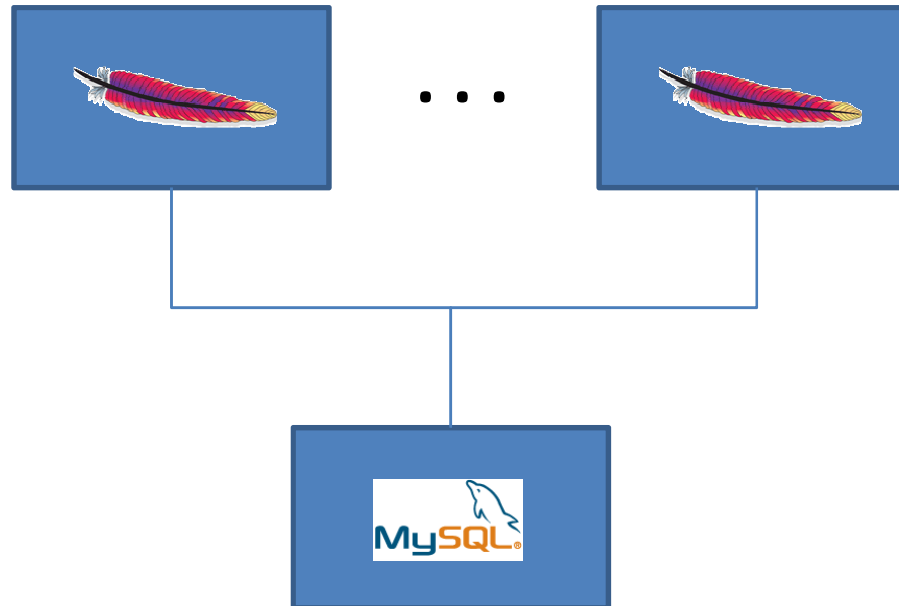


3. Step



Clients

- Load Balancer
- n Webservers



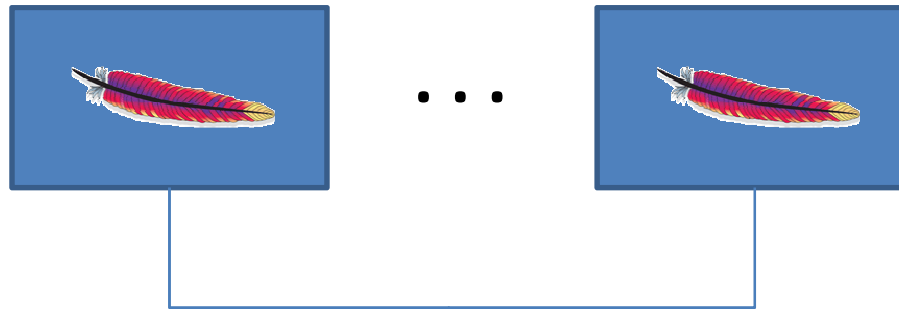
- Single DB server

4. Step



Clients

- Load Balancer
- n Webservers



- Master DB server



- Slave DB servers

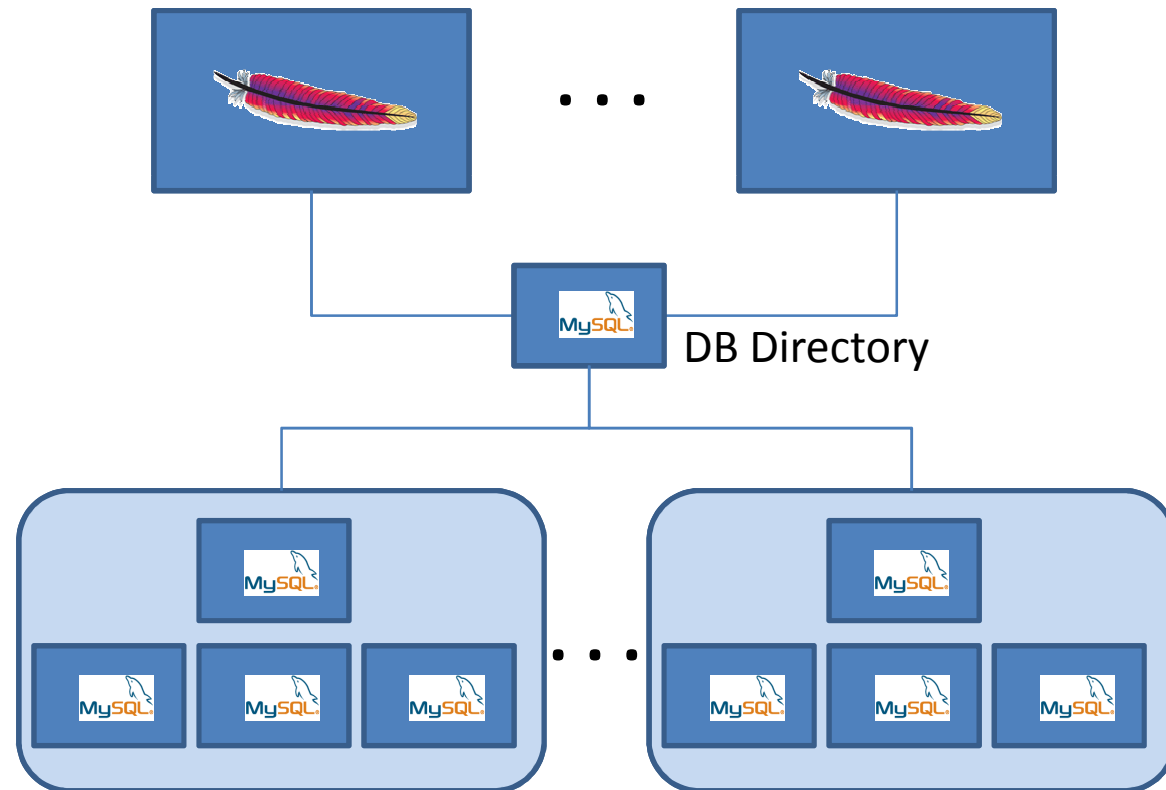


5. Step



Clients

- Load Balancer
- n Webservers
- DB partitioning
- DB clusters
- n DB servers
- memcached



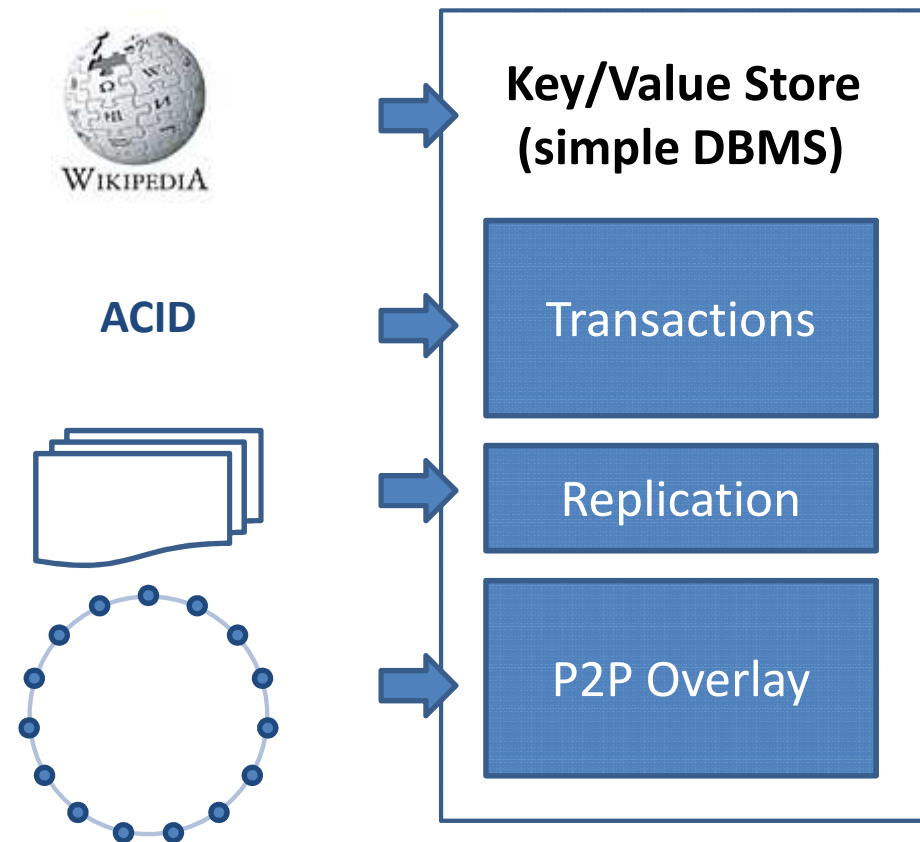
What did others do?

- SimpleDB
- BigTable
- Dynamo
- MySQL Cluster
- MapReduce
- Google FS

Our Approach: P2P in the Data Center

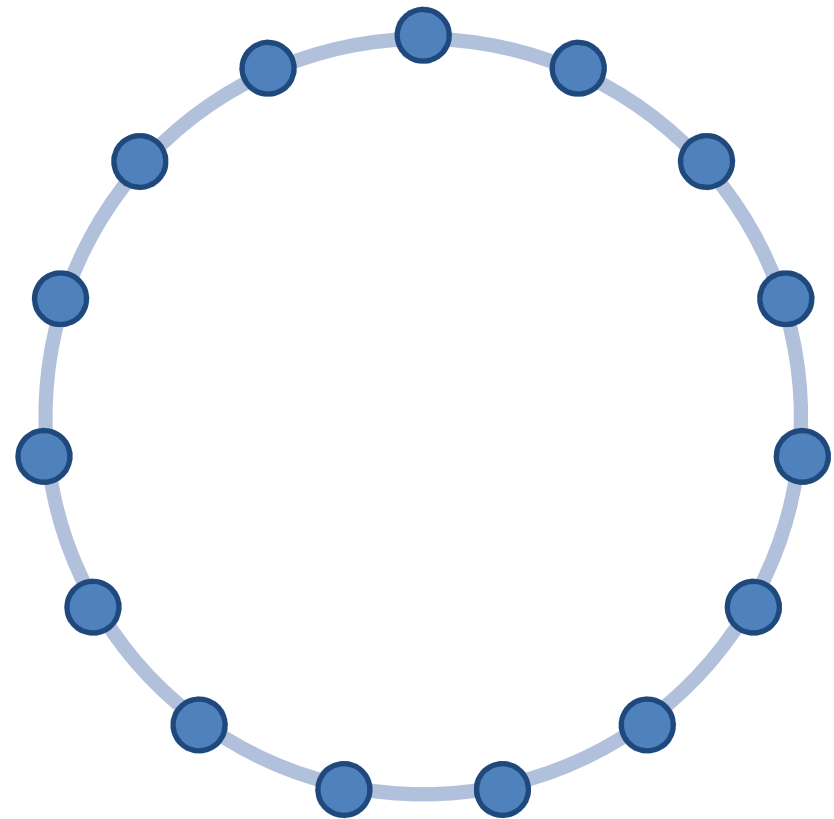


Clients



Scalable Wikipedia with Erlang

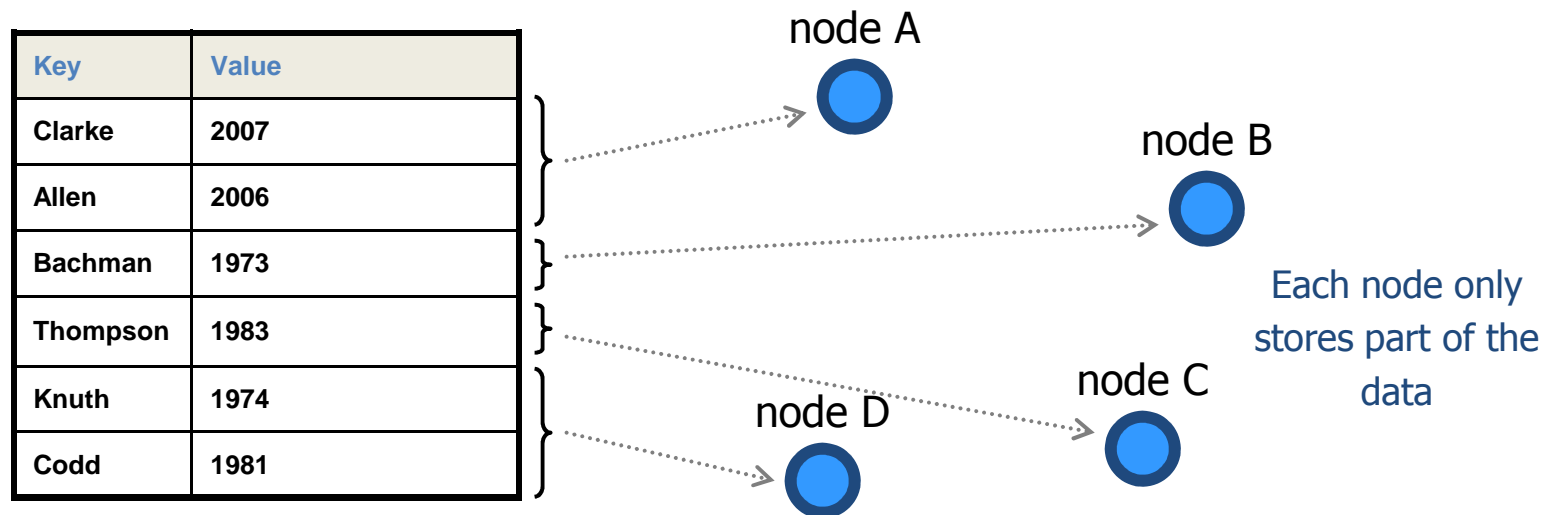
- Chord#
- Transactional Layer
- Load Balancing
- Wikipedia



Chord[#] = Chord \ Hash

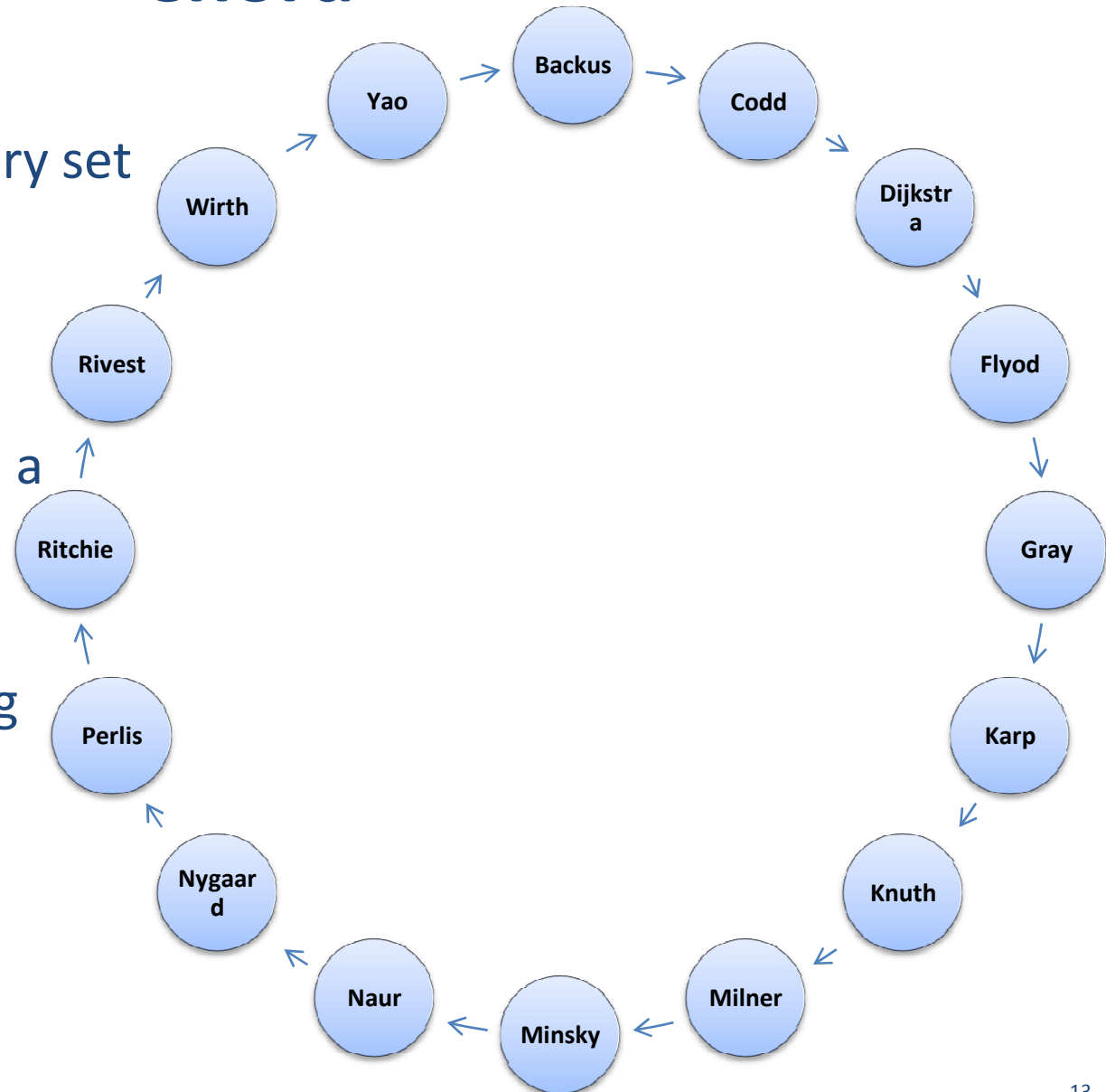
Chord#

- A *dictionary* has 3 ops:
 - insert(key, value)
 - delete(key)
 - lookup(key)
- Chord# implements a *distributed* dictionary



Chord#

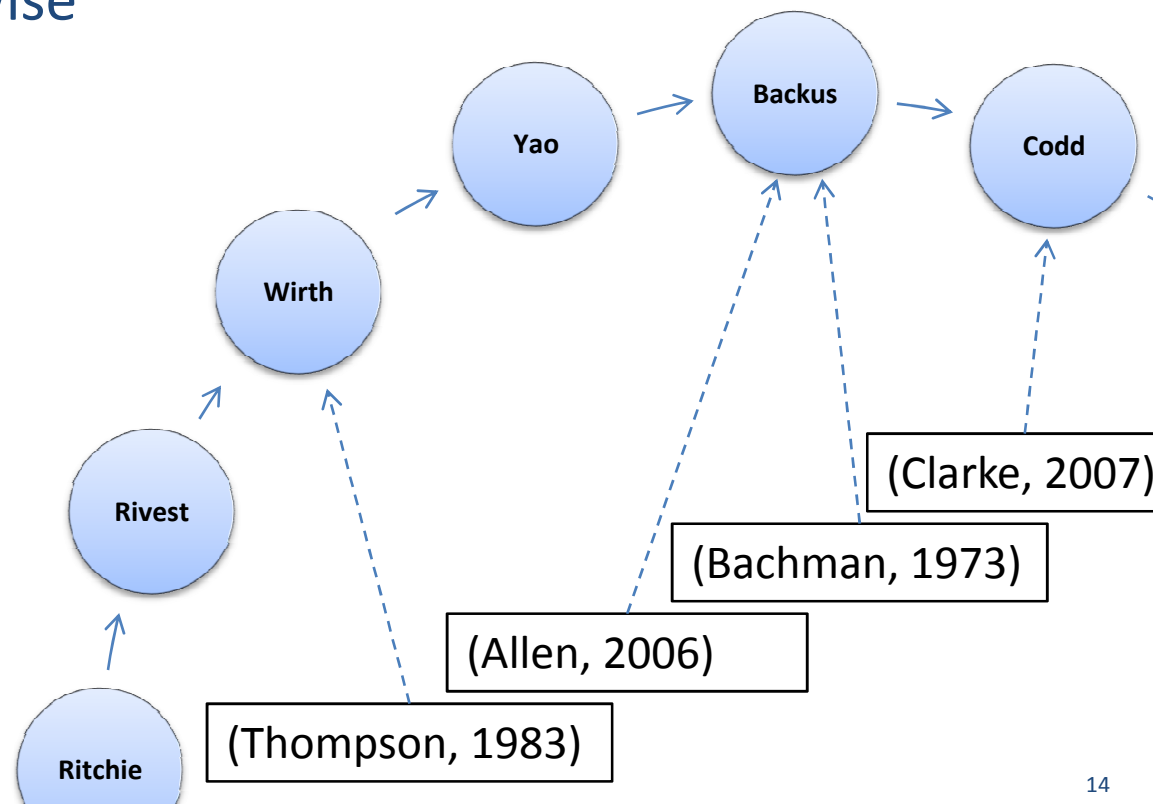
- Key Space is an arbitrary set with a total order, e.g. strings
- Every node is assigned a random key
- Nodes form logical ring over the key space



Distributed Dictionary

- Items are stored on their successor, i.e. the first node encountered in clockwise direction

- Thomson on Wirth
- Allen on Backus
- Bachman on Backus
- Clarke on Codd
- ...



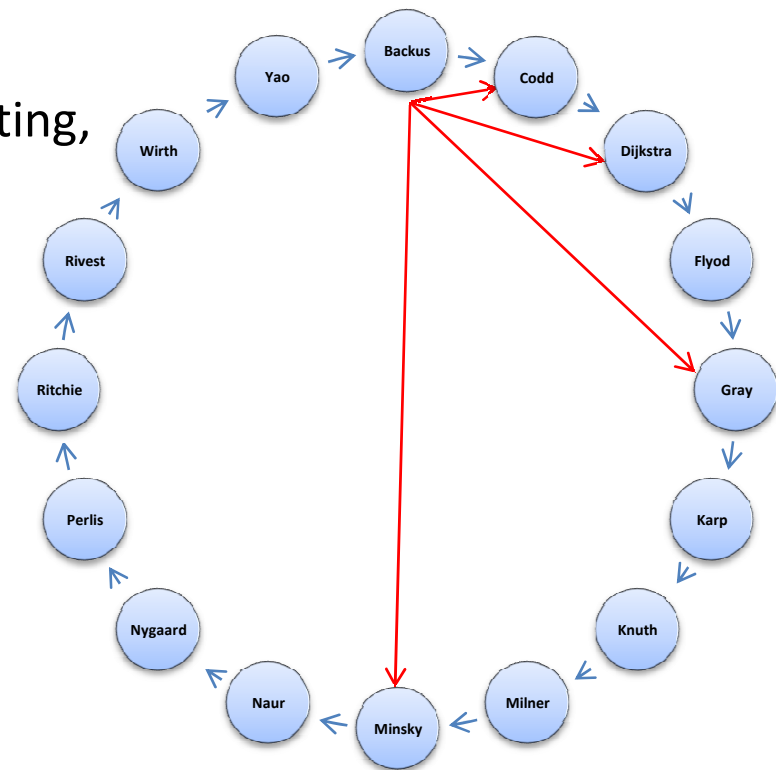
Distributed Dictionary

- Each node puts the successor and $\log_2 N$ exponentially spaced fingers in its routing table

$$pointer_i = \begin{cases} successor & : i = 0 \\ pointer_{i-1} . pointer_{i-1} & : i \neq 0 \end{cases}$$

- With each routing step using greedy routing, the distance between the current node and the destination is halved.

=> Yields $O(\log_2 N)$ hops.



Summary: Chord#

DHTs/Chord

- DHT is a fully decentralized data structure
- DHTs self-organize as nodes join, leave, and fail
- **All operations only require local knowledge**

Chord#

- Only **one** hop for calculating a routing pointer, Chord needs $\log(N)$ hops.
- **max. $\log(N)$ hops**, while Chord guarantees this only with “high probability”.
- Can adapt to imbalances in the query load; Chord can't.
- Supports range queries.

Scalable Wikipedia with Erlang

- Chord#
- Transactional Layer
- Load Balancing
- Wikipedia

ACID

Transactions on DHTs are Challenging

- high churn rate
 - nodes may leave, join, or crash at any time
→ changing responsibilities
- “crash stop” fault model
- no perfect “failure detector”
 - never know whether a node crashed or just slow network

Transactions + Replicas

START

debit (a, 100);

deposit (b, 100);

COMMIT

START

debit (a₁, 100);

debit (a₂, 100);

debit (a₃, 100);

deposit (b₁, 100);

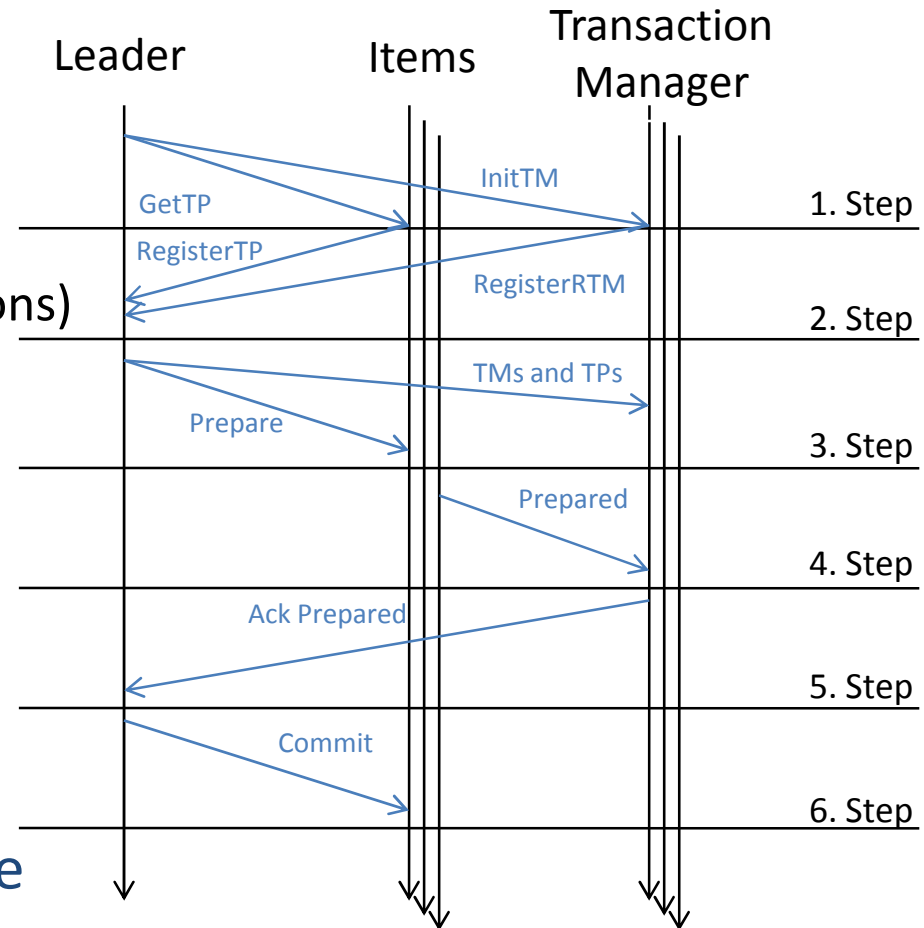
deposit (b₂, 100);

deposit (b₃, 100);

COMMIT

Adapted Paxos Commit

- Optimistic CC with fallback option
- Fast Read Operation
 - just 1 round
 - reads a majority (of the last versions) of all replicates
- Write Operation
 - 3 rounds
 - nonblocking (fallback)
- succeeds when $> f/2$ nodes alive



Summary “Transactions”

- Consistent way to update several items and their replicas
- Mitigates some of the Overlay Oddities
 - Node Failures
 - Asynchronous programming model

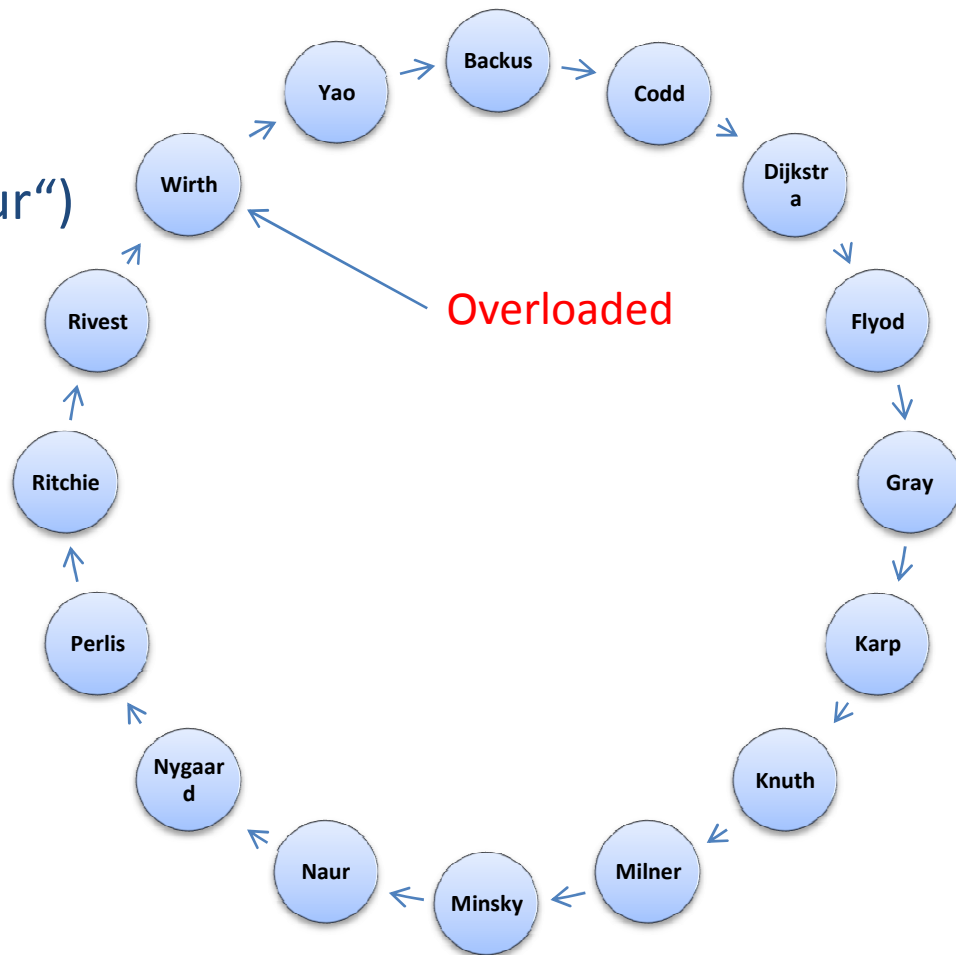
Scalable Wikipedia with Erlang

- Chord#
- Transactional Layer
- **Load Balancing**
- Wikipedia

Load-Balancing

1. Pick 2 random nodes („Naur“, „Wirth“)
2. IF $\text{Load}(\text{„Wirth“}) \gg \text{Load}(\text{„Naur“})$
 1. „Naur“ leaves system
 2. „Naur“ joins as „Tarjan“

Load can be *any* metric

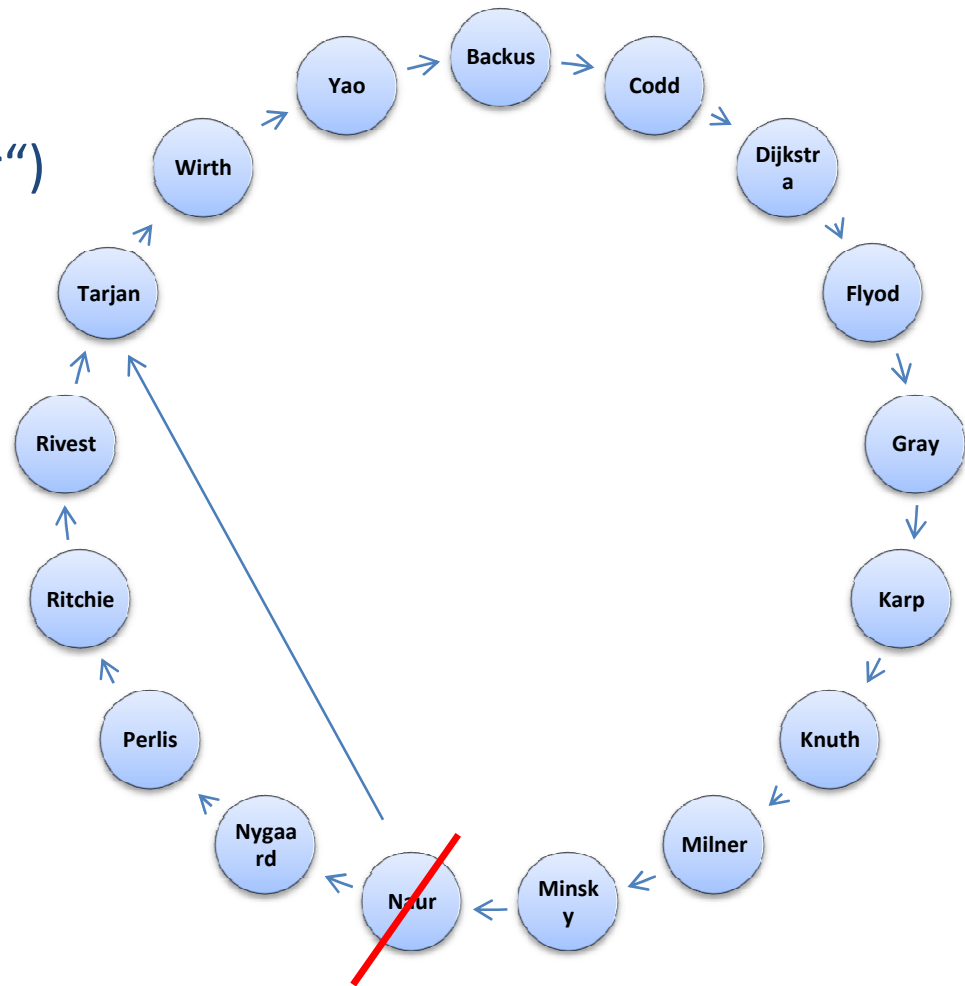


D. Karger, M. Ruhl. „Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems“. IPTPS 2004.

Load-Balancing

1. Pick 2 random nodes („Naur“, „Wirth“)
2. IF $\text{Load}(\text{„Wirth“}) \gg \text{Load}(\text{„Naur“})$
 1. „Naur“ leaves system
 2. „Naur“ joins as „Tarjan“

Load can be *any* metric



D. Karger, M. Ruhl. „Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems“. IPTPS 2004.

Multi Data-Center Scenario

- Multi-Data-Center Scenarios

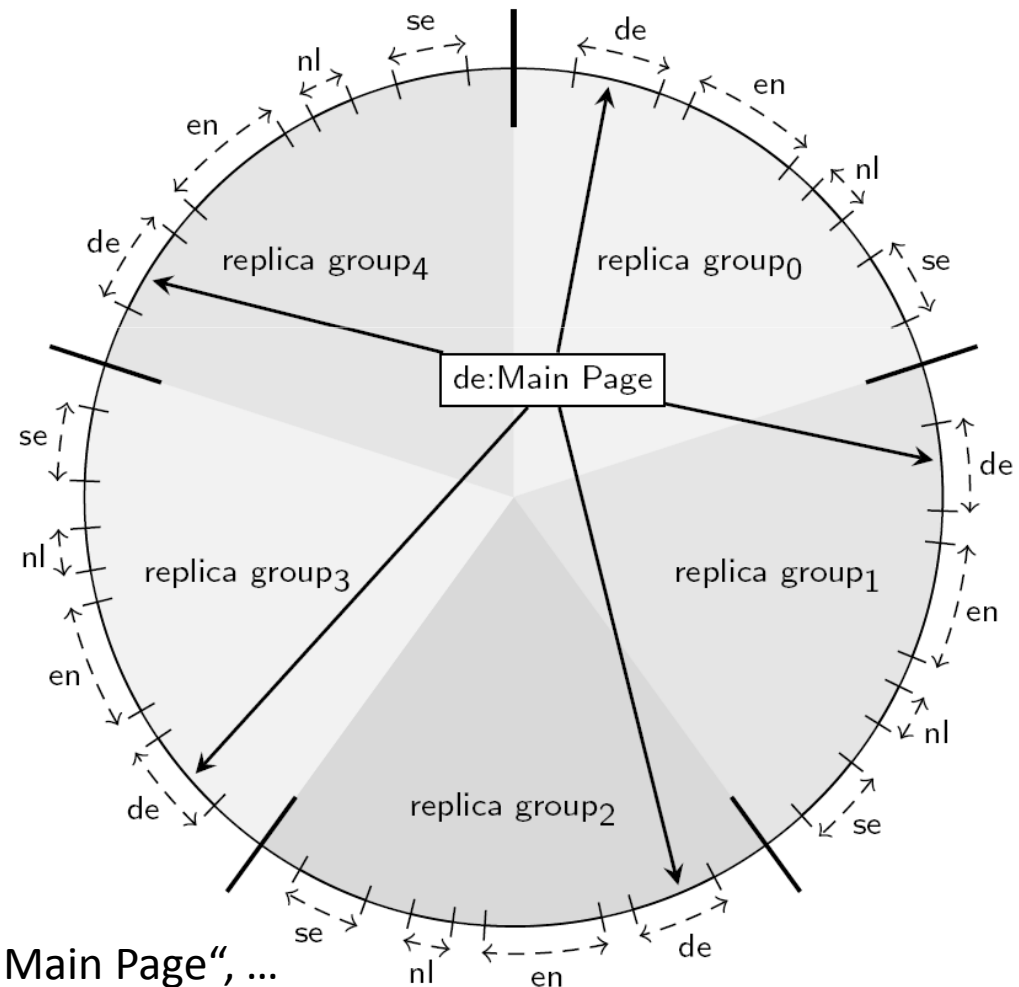
- Optimize for Latency
- Increase Availability

- Prefix Articles with

- Language
- Replicas Number

- E.g. „de:Main Page“

- 5 replicas
- 2 in Germany
- 1 in UK
- 1 in USA
- 1 in Asia



„0de:Main Page“, „1de:Main Page“, „2de:Main Page“, ...

Scalable Wikipedia with Erlang

- Chord#
- Transactional Layer
- Load Balancing
- **Wikipedia**



Wikipedia

Top 10 Web sites

1. Yahoo!
2. Google
3. YouTube
4. Windows Live
5. MSN
6. Myspace
7. **Wikipedia**
8. Facebook
9. Blogger.com
10. Yahoo!カテゴリ

Wikipedia is the top1 *open* Web Site

- source code is open
- architecture is open
- dumps available

Source: alexa.com

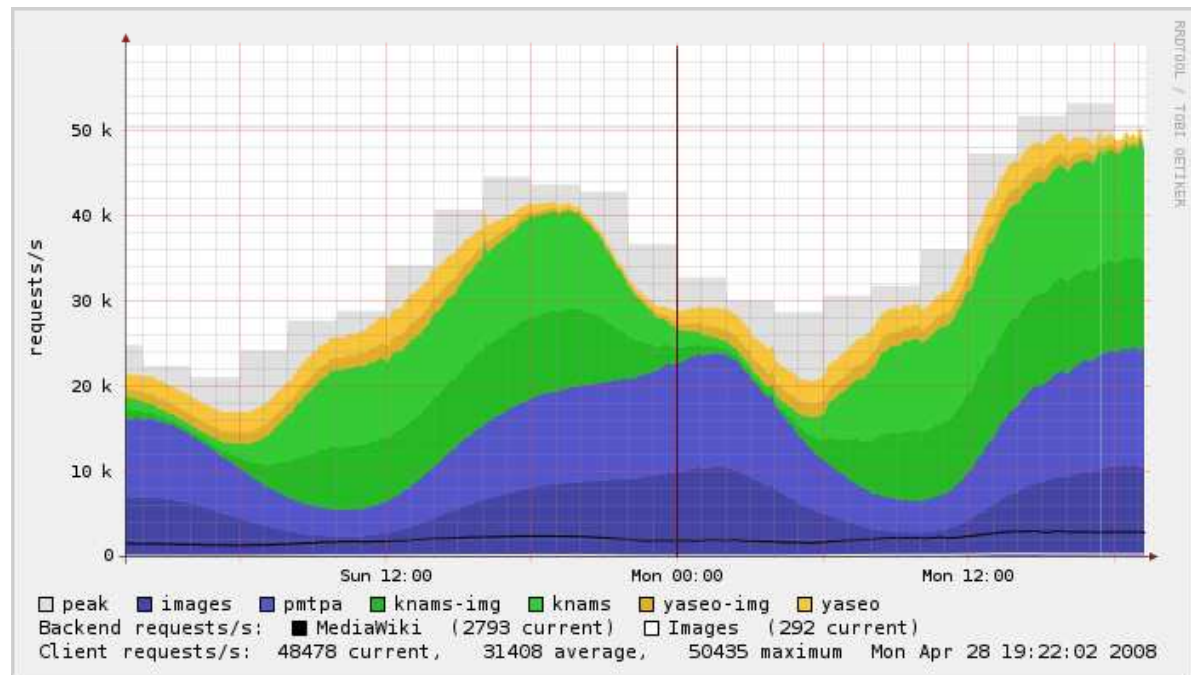
Wikipedia

Top 10 Web sites

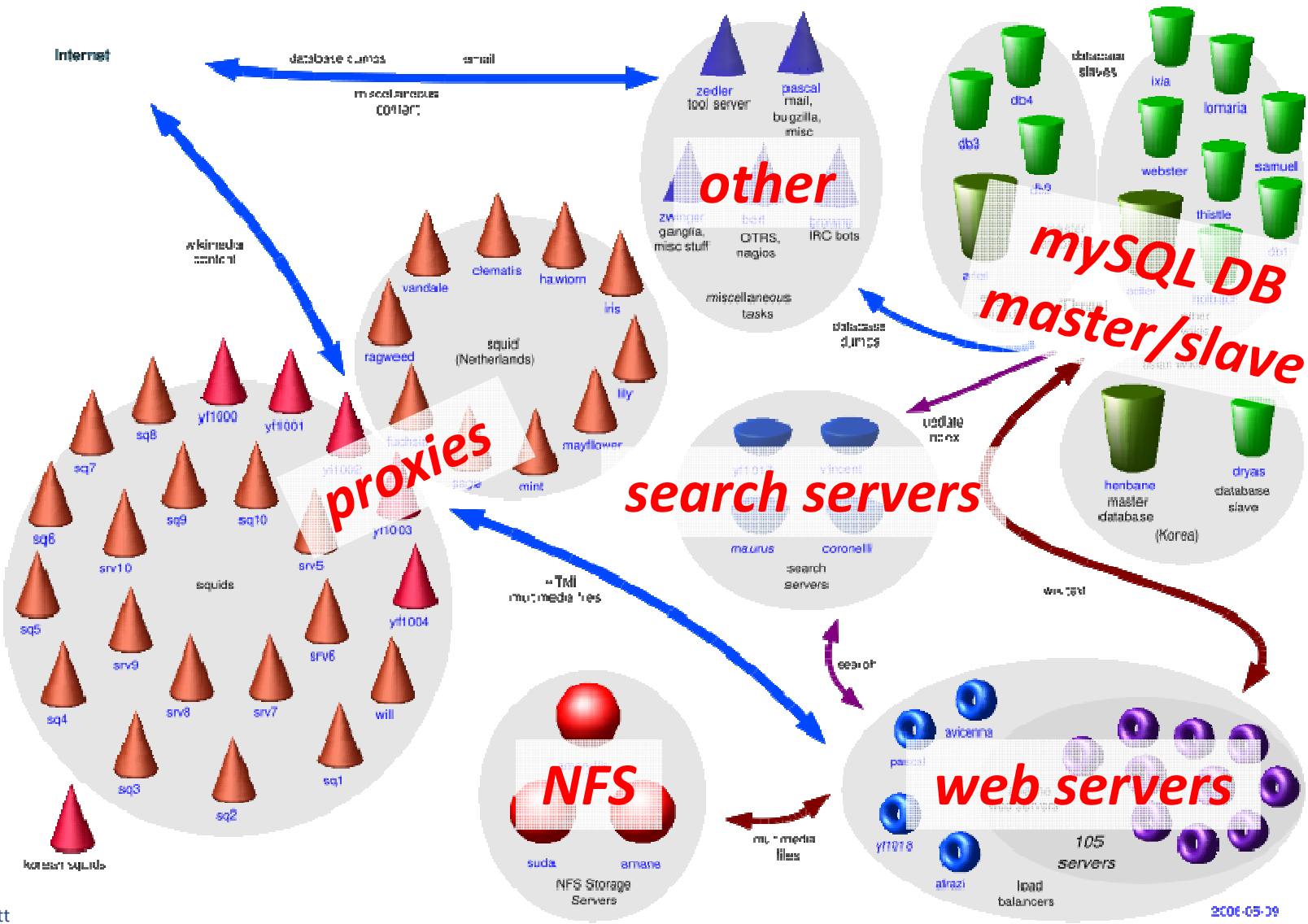
1. Yahoo!
2. Google
3. YouTube
4. Windows Live
5. MSN
6. Myspace
7. **Wikipedia**
8. Facebook
9. Blogger.com
10. Yahoo!カテゴリ

50.000 requests/sec

- 95% are answered by squid proxies
- 2,000 req./sec hit the backend



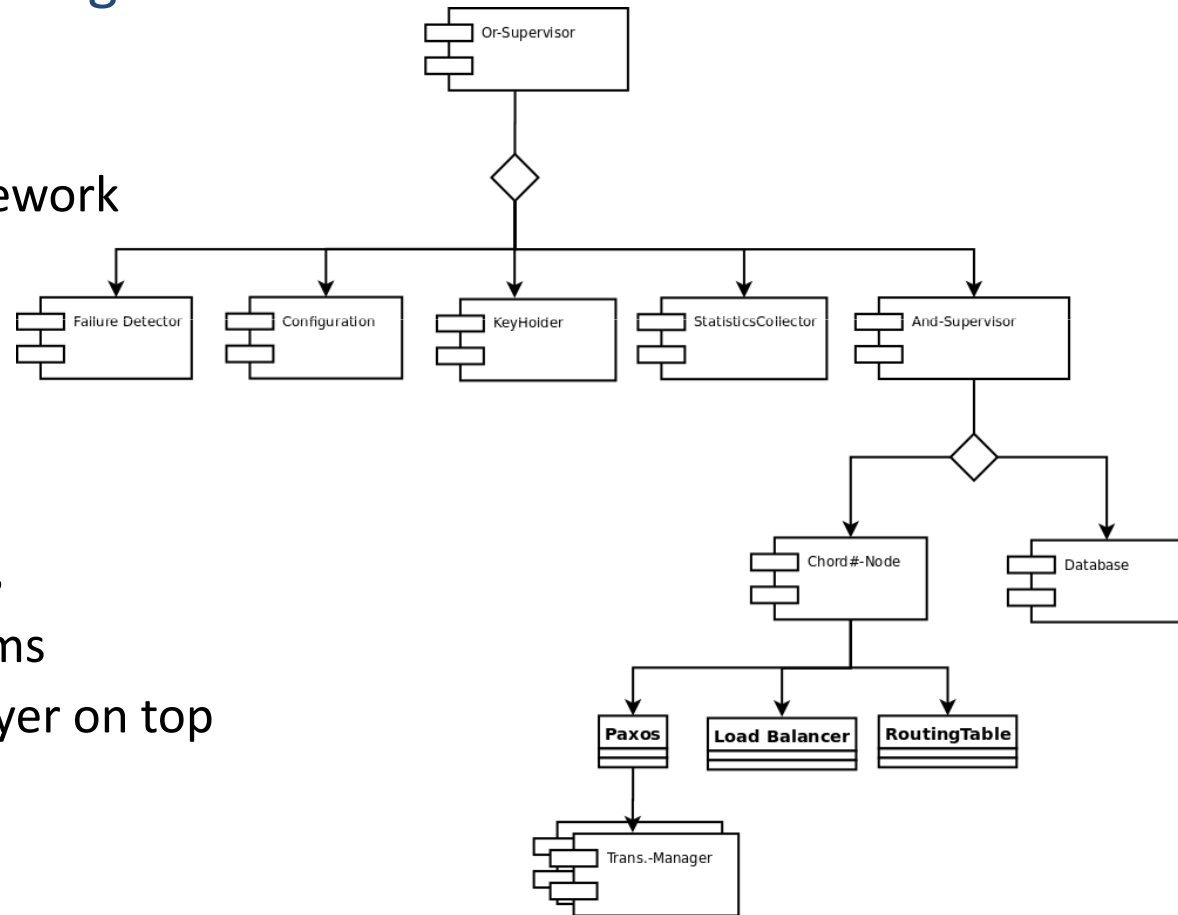
The Wikipedia System Architecture



Erlang Implementation

Overlay

- Implemented in Erlang
 - Chord#
 - Load-Balancing
 - Transaction Framework
- OTP behaviours
 - Supervisor
 - gen_server
- Distributed Erlang
 - Security Problems
 - Scalability Problems
 - > Own Transport-Layer on top of TCP



Data Model

Wikipedia

– SQL DB



Chord#

– Key-Value Store

```
CREATE TABLE /*$wgDBprefix*/page (  
  page_id int unsigned NOT  
    NULL auto_increment,  
  page_namespace int NOT NULL,  
  ...
```

Map Relations to Key-Value Pairs

- (Title, List of Versions)
- (CategoryName, List of Titles)
- (Title, List of Titles) //Backlinks

Data Model

```
void updatePage(string title, int oldVersion, string newText)
{
    //new transaction
    Transaction t = new Transaction();
    //read old version
    Page p = t.read(title);
    //check for concurrent update
    if(p.currentVersion != oldVersion)
        t.abort();
    else{
        //write new text
        t.write(p.add(newText));
        //update categories
        foreach(Category c in p)
            t.write(t.read(c.name).add(title));
        //commit
        t.commit();
    }
}
```

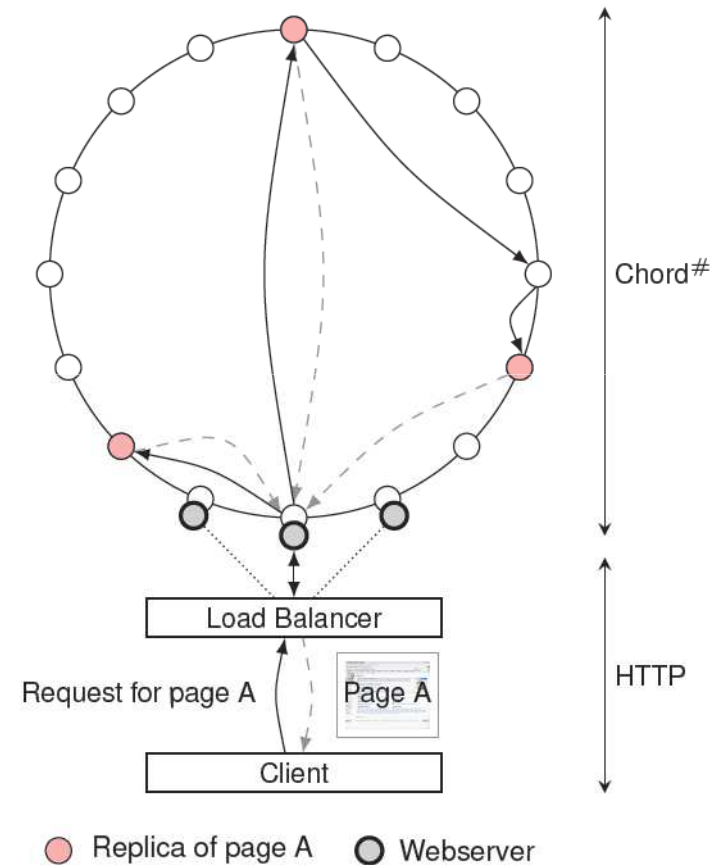
Wiki

Database:

- Chord#
- Mapping
 - Wiki -> Key-Value Store

Renderer:

- Java
 - Tomcat
 - Plog4u
- Jinterface
 - Interface to Erlang



IEEE Scale Challenge 2008

Live Demos

- Bavarian
- Simple English

- Browsing
- Editing with full History
- Category-Pages

• Deployments

- Planet-Lab
 - 20 nodes
- Cluster
 - 320 nodes in Berlin



Summary

DHT + Transactions = Scalable, Reliable, Efficient Key/Value Store

- Previously, P2P was mainly used for file sharing (read only).
- **We support consistent, distributed *write* operations.**
- Numerous applications
 - Internet databases, transactional online-services, ...

Team

- Thorsten Schütt,
- Florian Schintke,
- Monika Moser,
- Stefan Plantikow,
- Alexander Reinefeld,
- Nico Kruber,
- Christian von Prollius,
- Seif Haridi (SICS),
- Ali Ghodsi (SICS),
- Tallat Shafaat (SICS)

Detailed Descriptions

Wiki

S. Plantikow, A. Reinefeld, F. Schintke.

Transactions for Distributed Wikis on Structured Overlays.

DSOM, October 2007.

Transactions

M. Moser, S. Haridi.

Atomic Commitment in Transactional DHTs.

1st CoreGRID Symposium, August 2007.

T. Shafaat, M. Moser, A. Ghodsi, S. Haridi, T. Schütt, A. Reinefeld.

Key-Based Consistency and Availability in Structured Overlay Networks.

Infoscale, June 2008.

DHT

T. Schütt, F. Schintke, A. Reinefeld.

A Structured Overlay for Multi-dimensional Range Queries.

Euro-Par, August 2007.

T. Schütt, F. Schintke, A. Reinefeld.

Structured Overlay without Consistent Hashing: Empirical Results.

GP2PC, May 2006.

Questions