Scalaris: Reliable Transactional P2P Key/Value Store

Thorsten Schütt, Florian Schintke, Alexander Reinefeld

Zuse Institute Berlin (ZIB)
onScale solutions
Background

• CSR – Parallel and Distributed Computing
• P2P Computing, Structured Overlay Networks

• Need Simulators
  • Experiments (empirical evaluation)
  • Abstract from certain details

• Need Implementations (EU projects)
  • „production-quality“ code
Scalaris vs. Mnesia

- Distributed database
- Partitioning
- Transactions
- Key-Value Store
- More scalable

- Distributed database
- Partitioning (table fragmentation)
- Transactions
- Complex Data Model
Our Approach

Clients

Key/Value Store (simple DBMS)

ACID

Transactions

Replication

P2P Overlay
Scalaris
P2P Overlay
Chord#

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

### Key & Value

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarke</td>
<td>2007</td>
</tr>
<tr>
<td>Allen</td>
<td>2006</td>
</tr>
<tr>
<td>Bachman</td>
<td>1973</td>
</tr>
<tr>
<td>Thompson</td>
<td>1983</td>
</tr>
<tr>
<td>Knuth</td>
<td>1974</td>
</tr>
<tr>
<td>Codd</td>
<td>1981</td>
</tr>
</tbody>
</table>

Each node only stores part of the data.
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
  
  $$\text{pointer}_i = \begin{cases} 
  \text{successor} & : i = 0 \\
  \text{pointer}_{i-1} \cdot \text{pointer}_{i-1} & : i \neq 0 
  \end{cases}$$

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

$\Rightarrow$ 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.
Scalaris
Transactions
Transactions + Replicas

START

debit (a, 100);
deposit (b, 100);

COMMIT

START

debit (a_1, 100);
debit (a_2, 100);
debit (a_3, 100);
deposit (b_1, 100);
deposit (b_2, 100);
deposit (b_3, 100);

COMMIT
Adapted Paxos Commit

- Optimistic CC with fallback
- Write
  - 3 rounds
  - non-blocking (fallback)
- Read even faster
  - reads majority of replicas
  - just 1 round
- succeeds when >f/2 nodes alive
Motivation for Using Erlang

• Fit our model
  • Processes + Message Passing

• Allows simple testing/simulating
  • Light-Weight Processes
  • 100s of nodes on a laptop

• Same code runs:
  • locally and
  • on a cluster

• Simulator => Application
Distributed Erlang

Too Powerfull:
• We want to implement our own Failure Detectors
  • Part of the Algorithms

Too weak:
• SSL + X509?
• Scalability?

⇒ TCP-based „reimplementation“
  ⇒ Message statistics for free
Transactions in mnesia

\[
\text{raise(Eno, Raise) -> } \\
\text{\quad F = fun() ->} \\
\text{\quad\quad [E] = mnesia:read(employee, Eno, write),} \\
\text{\quad\quad Salary = E\#employee.salary + Raise,} \\
\text{\quad\quad New = E\#employee\{salary = Salary\},} \\
\text{\quad\quad mnesia:write(New)} \\
\text{\quad end,} \\
\text{\quad mnesia:transaction(F).}
\]
Transactions in Scalaris

F = fun (TransLog) ->
    {X, TL1} = read(TransLog, "Account A"),
    {Y, TL2} = read(TL1, "Account B"),
    if
      X > 100 ->
        TL3 = write(TL2, "Account A", X - 100),
        TL4 = write(TL3, "Account B", Y + 100)
        {ok, TL4};
      true ->
        {ok, TL2};
    end
    end,
transaction:do_transaction(F, ...).
Supervisor Tree

- **Components**
  - Chord#
  - Load-Balancing
  - Transaction Framework

- **OTP behaviours**
  - supervisor
  - gen_server
Erlang Quirks

• Flat Namespace for Processes

• Error Messages (Line Number?)
  • I can translate gcc to english (took me 2 years)

• Packages, e.g. transstore.transaction_api:do/1
  • Sub-Directories
    ○ Cover?
    ○ Dialyzer?
Wikipedia

Top 10 Web sites

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

50,000 requests/sec

- 95% are answered by squid proxies
- only 2,000 req./sec hit the backend
The Wikipedia System Architecture
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
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)
Publications

Chord#

T. Schütt, F. Schintke, A. Reinefeld.

T. Schütt, F. Schintke, A. Reinefeld.

Transactions

M. Moser, S. Haridi.
Atomic Commitment in Transactional DHTs. 1st CoreGRID Symposium, August 2007.


Wiki

S. Plantikow, A. Reinefeld, F. Schintke.
Transactions for Distributed Wikis on Structured Overlays. DSOM, October 2007.

Talks / Demos

IEEE Scale Challenge, May 2008
1st price (live demo)

http://code.google.com/p/scalaris
Questions
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“, ...