Memory: The New Disk

TU Berlin, 2010-11-12
Who?

• Tim Lossen / @tlossen
• Ruby developer
• backend developer at ....
Based in Berlin, wooga is the leading European social games developer.

We are hiring!

Currently, we are searching for:

- Senior Creative Producer (m/f)
- Vorstandsassistent (m/w) (m/w)
- Software Engineer - Graduate Position (m/f)

wooga German start-up blog deutsche-startups.de visited us and published photos of their tour.

How do you like our office in Berlin?

Hausbesuch bei wooga :: deutsche-startups.de
www.deutsche-startups.de

Deutsche-startups – eine Art Infoservice fur die Web2.0-Branche. Hausbesuch bei wooga

Friday at 1:16am

wooga We added interviews from a few of our employees to find out what they like about working at wooga. Check it out:
Requirements

• backend for facebook game
Requirements

• backend for facebook game
• 1 mio. daily users
• 100 KB data per user
Requirements

- **peak traffic:**
  - 10,000 concurrent users
  - 3,000 requests/second
Requirements

• peak traffic:
  - 10,000 concurrent users
  - 3,000 requests/second

• write-heavy workload
Relational Database
Scaling Reads

- caching (memcached)
Scaling Reads

- caching (memcached)
- read-only slaves
  - horizontal scaling
Scaling Reads

- caching (memcached)
- read-only slaves
  - horizontal scaling
- easy
Scaling Writes

- limit: 1000 writes/second on EC2
Scaling Writes

• limit: 1000 writes/second on EC2
• sharding
  - split database into parts
Scaling Writes

- limit: 1000 writes/second on EC2
- sharding
  - split database into parts

- difficult
Alternatives?
Alternatives?

- Cassandra
- CouchDB
- MongoDB
- Redis
- Riak
- Membase
- SimpleDB
- BigTable
- CouchDB
- Hypertable
Basics
# Relative Latency

<table>
<thead>
<tr>
<th>Memory</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD</td>
<td></td>
</tr>
<tr>
<td>Disk</td>
<td></td>
</tr>
</tbody>
</table>
## Relative Latency

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>1</td>
</tr>
<tr>
<td>SSD</td>
<td>1000</td>
</tr>
<tr>
<td>Disk</td>
<td></td>
</tr>
</tbody>
</table>

$x \times 1000$
## Relative Latency

<table>
<thead>
<tr>
<th></th>
<th>Memory</th>
<th>SSD</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1000</td>
<td>100 000</td>
</tr>
</tbody>
</table>

x 1000

x 100
## Pizza Delivery Time

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>x 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory</strong></td>
<td><strong>30 minutes</strong></td>
<td></td>
</tr>
<tr>
<td><strong>SSD</strong></td>
<td><strong>3 weeks</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Disk</strong></td>
<td><strong>5.5 years</strong></td>
<td></td>
</tr>
</tbody>
</table>
“Memory is the new disk, disk is the new tape.”
—Jim Gray
Rule of thumb

• random access = memory
• sequential access = disk
## Euro per GB

<table>
<thead>
<tr>
<th>Memory</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD</td>
<td>2</td>
</tr>
<tr>
<td>Disk</td>
<td>0.05</td>
</tr>
</tbody>
</table>

\[ \div 10 \]

\[ \div 40 \]
Redis
Architecture

• key-value-store
• in-memory database
  - with virtual memory
Durability

- full database dumps
Durability

- full database dumps
- append-only log
Killer Feature

• high (write) throughput
  - 30 to 150 K operations / second
Other Features

• interesting data structures
  - lists, hashes, (sorted) sets
  - atomic operations
Our Setup
Architecture

• single Redis master
  - with virtual memory
  - handles all reads / writes
Architecture

- single Redis master
  - with virtual memory
  - handles all reads / writes
- single Redis slave
  - as hot standby (for failover)
Throughput

• **redis-benchmark**

  - \(60\, \text{K ops/s} = 3.6\, \text{mio ops/m}\)
Throughput

• **redis-benchmark**
  - 60 K ops / s = **3.6 mio** ops / m

• **monitoring (rpm, scout)**
  - ca. 10 ops per request
Throughput

• redis-benchmark
  - 60 K ops / s = 3.6 mio ops / m
• monitoring (rpm, scout)
  - ca. 10 ops per request

• 200 K req / m = 2.0 mio ops / m
Capacity 1

- 100 KB / user (on disk)
- 10,000 concurrent users (peak)
Capacity 1

• 100 KB / user (on disk)
• 10,000 concurrent users (peak)

• 1 GB memory
  - (plus Redis overhead)
Capacity 2

- Redis keeps all keys in memory
- 10 mio. total users
- 20 GB / 100 mio. integer keys
Capacity 2

- Redis keeps all keys in memory
- 10 mio. total users
- 20 GB / 100 mio. integer keys
- 2 GB memory for keys
Data model

• one Redis hash per user
  - key: facebook id
Data model

• one Redis hash per user
  - key: facebook id

• store data as serialized JSON
  - booleans, strings, numbers, timestamps ...
Advantages

- efficient to swap user data in / out
Advantages

• efficient to swap user data in / out
• turns Redis into “document db”
  – atomic ops on parts
Advantages

• efficient to swap user data in / out
• turns Redis into “document db”
  – atomic ops on parts
• easy to dump / restore user data
Advice
Advice

• use the right tool for the job
Advice

- use the right tool for the job
- avoid sharding
  - as long as possible
Advice

• use the right tool for the job
• avoid sharding
  - as long as possible
• keep it simple
Q & A
Nosql Night Berlin

November 17, 19:30
newthinking store
Tucholskystr. 48
Links

• "A Conversation with Jim Gray"
• redis.io
• tim.lossen.de
• wooga.com/jobs