Who am I?

- Name: Berner Setterwall
- Co-founding CTO at Campanja
- Likes climbing, paragliding and bicycling.
- Favorite programming language prior to Erlang: Perl

Switching to Erlang

For fun and profit!

Campanja?

- Search Engine Marketing Optimization system
- Solving the problems for big advertisers
- Minimal customer facing dashboard
- Mainly using Erlang
- 20 FTE, moving into new office, expanding

A short Campanja history

- How SEM software companies start.
- Focus on short time to market
- Wanted cheap hosting

Campanja 2007 - 2010

- SEM productivity tool
- Tailored for media agencies needs
- Client software build using .NET
- Backend+API based on LAMP+Perl
- 2 - 3 FTE

Problem: Data

- Large datasets
- Queries hard to pre-compute
- Most queries useless to cache
- MySQL can be hard to scale

Problem: Concurrency

- Advertising is naturally parallel
- Need async I/O due to “slow” queries

Switching to Erlang

Google introduces “preferred pricing”

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Sign Avito.ru

Need to scale again

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Node.js

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Node.js: Hopes

- Easy to get started
- Easy to run untrusted code
- Internet says it is fast =)
- Hoping for short time to market
- Easy to be charmed trendy stuff

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Node.js: Despair

- Performance not what I hoped for
- Callback Hell
- Poor error handling

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Node.js: Example

```
var http = require('http');
var sys  = require('sys');

function DEBUG(str) {
    if (1) console.log(str);
}

var serv = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/json'});
    var post_data = "";
    DEBUG("waiting for data ..");
    request.on('data',function (chunk) {
        DEBUG("got data ..");
        post_data += chunk;
    });
    request.on('end', function() {
        output = {'hello': 'world'};
        response.end(JSON.stringify(output));
    });
});

serv.on('error',function (err) {console.log('error event: '+err)});
serv.on('clientError',function (err) {console.log('clientError event: '+err)});

Port = 1234;
serv.listen(Port)

:!node example.js
waiting for data ..
got data ..
clientError event: Error: Parse Error
```

```
$ curl -v -XPOST 'http://127.0.0.1:1234' -d 'asd ' '-HContent-Length: 12345'
* About to connect() to 127.0.0.1 port 8125 (#0)
*   Trying 127.0.0.1... connected
* Connected to 127.0.0.1 (127.0.0.1) port 8125 (#0)
> POST / HTTP/1.1
> User-Agent: curl/7.18.2
> Host: 127.0.0.1:8125
> Accept: */*
> Content-Length: 12345
> Content-Type: application/x-www-form-urlencoded
>

^C
```

Can Scala replace Node.js?

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Scala: Hopes

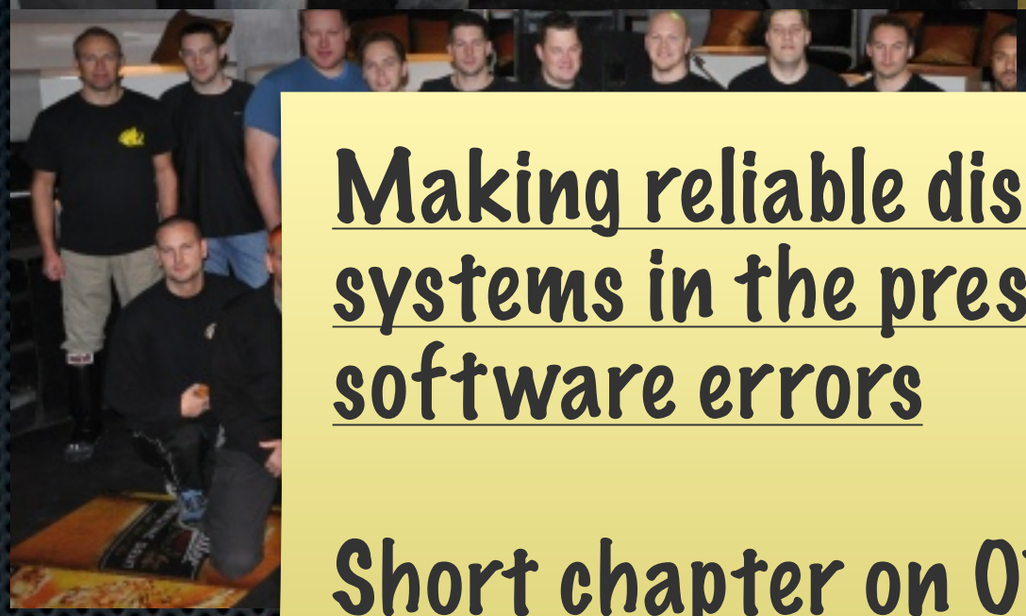
- JVM error handling better than V8 + node.js hack?
- Better support for multiple cores?
- JVM better performance?
- More mature?
- Lots of libraries to use
- Not Java :)

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Scala: Prototypes

- Performance seems better than Node.js
- Some java examples hard to translate to scala
- Need lots of classes and stuff to get things going

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011



Making reliable distributed systems in the presence of software errors

Short chapter on OTP - maybe too tired =)

Krav

This is a good way to avoid focusing on work.

Product 1

changeset: 0:afc802228540
user: Berner Setterwall <berner.sett
date: Sun Feb 13 18:34:44 2011 +0
summary: * init commit - working cam

Making reliable distributed systems in the presence of software errors

Short chapter on OTP - maybe too tired =)

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Erlang: First impressions

- Hot Code loading enables fast iteration
- Modules are simple
- Used gen_event but wrote mostly pure erlang (i.e. w/o OTP)

Product 2

```
changeset:   0:394c9c2ad8b5
user:        Berner Setterwall <berner.setterwall@campanja.com>
date:        Wed Mar 02 13:16:03 2011 +0000
summary:     * init commit ..
```

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Erlang is number one

After two successful pilots, erlang is the new language of choice for (most) projects at Campanja

Product 3

```
changeset:   0:86bf0eaae903
user:        berset
date:        Thu Apr 07 06:49:07 2011 +0000
summary:     * dirty init commit (includes inets:start() patch).
```

June 2010 July 2010 October 2010 November 2010 December 2010
January 2011 February 2011 March 2011 April 2011

Some surprises during our first year

httpc

- Switched to lhttpc

emysql

- https://github.com/Eonblast/Emysql/issues/20

merle

- Switch to erlmc

ezk

handle_call(_, From0, _) ->
From = {blocking, From0},

```
From ! {error, client_broke, CommId, Path},
```

- https://github.com/infinipool/ezk/blob/master/src/ezk_connection.erl#L428

Data?

With the parallelism problem controlled, I started looking for a new approach to our data problems.

Riak

0.14.0

Riak Eval (scala client)

```
Status:     3856016/    49731000 in 11586.6s (s:  4292.1/s r:   332.8/s) Outstanding: 45874984 Missing: 92.25%
Status:     3856042/    49732000 in 11586.6s (s:  4292.2/s r:   332.8/s) Outstanding: 45875958 Missing: 92.25%
Status:     3856071/    49733000 in 11586.7s (s:  4292.2/s r:   332.8/s) Outstanding: 45876929 Missing: 92.25%
Exception in thread "pool-25-thread-1" java.lang.OutOfMemoryError: Java heap space
        at java.nio.HeapByteBuffer.<init>(HeapByteBuffer.java:57)
        at java.nio.ByteBuffer.allocate(ByteBuffer.java:329)
        at org.apache.mina.core.buffer.SimpleBufferAllocator.allocateNioBuffer(SimpleBufferAllocator.java:44)
        at org.apache.mina.core.buffer.SimpleBufferAllocator.allocate(SimpleBufferAllocator.java:36)
        at org.apache.mina.core.buffer.IoBuffer.allocate(IoBuffer.java:218)
        at org.apache.mina.core.buffer.IoBuffer.allocate(IoBuffer.java:203)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor.read(AbstractPollingIoProcessor.java:609)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor.process(AbstractPollingIoProcessor.java:598)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor.process(AbstractPollingIoProcessor.java:587)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor.access$400(AbstractPollingIoProcessor.java:61)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor$Processor.run(AbstractPollingIoProcessor.java:969)
        at org.apache.mina.util.NamePreservingRunnable.run(NamePreservingRunnable.java:64)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1110)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:603)
        at java.lang.Thread.run(Thread.java:636)
Exception in thread "pool-7-thread-1" java.lang.OutOfMemoryError: Java heap space
Exception in thread "pool-1-thread-1" Exception in thread "pool-19-thread-1" java.lang.OutOfMemoryError: Java heap space
        at java.nio.HeapByteBuffer.<init>(HeapByteBuffer.java:57)
        at java.nio.ByteBuffer.allocate(ByteBuffer.java:329)
        at org.apache.mina.core.buffer.SimpleBufferAllocator.allocateNioBuffer(SimpleBufferAllocator.java:44)
        at org.apache.mina.core.buffer.SimpleBufferAllocator.allocate(SimpleBufferAllocator.java:36)
        at org.apache.mina.core.buffer.IoBuffer.allocate(IoBuffer.java:218)
        at org.apache.mina.core.buffer.IoBuffer.allocate(IoBuffer.java:203)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor.read(AbstractPollingIoProcessor.java:609)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor.process(AbstractPollingIoProcessor.java:598)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor.process(AbstractPollingIoProcessor.java:587)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor.access$400(AbstractPollingIoProcessor.java:61)
        at org.apache.mina.core.polling.AbstractPollingIoProcessor$Processor.run(AbstractPollingIoProcessor.java:969)
        at org.apache.mina.util.NamePreservingRunnable.run(NamePreservingRunnable.java:64)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1110)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:603)
        at java.lang.Thread.run(Thread.java:636)
Exception in thread "pool-39-thread-1" java.lang.OutOfMemoryError: Java heap space
```

Riak Eval

```
$ curl 'http://localhost:8091/riak/bucket?keys=stream'
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  159M    0  159M    0     0  93870      0 --:--:--  0:29:41 --:--:--     0

^C
```

```
root@db-bgc1:~# du -sh /var/lib/riak/bitcask/
90G   /var/lib/riak/bitcask/
root@db-bgc1:~#
```

top while reading:

```
top - 12:26:09 up 341 days, 21:23,  4 users,  load average: 6.49, 4.66, 3.44
Tasks: 133 total,   2 running, 131 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.1%us, 18.0%sy,  0.0%ni, 31.0%id, 49.9%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  16473904k total, 16392440k used,    81464k free,      424k buffers
Swap:  3855592k total,  3457700k used,   397892k free,     4056k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3421 riak      20   0 18.5g  14g 2260 S  108 94.9 966:03.30 beam.smp
  275 root      15  -5     0    0    0 R   70  0.0  14:19.60 kswapd0
 1354 root      15  -5     0    0    0 S   29  0.0  55:56.94 kjournald
```

```
=ERROR REPORT==== 1-Feb-2011::17:04:13 ===
** State machine <0.3623.0> terminating
** Last event in was timeout
** When State == initialize
**      Data  == {state,<0.3562.0>,undefined,default,undefined,undefined,
...
                    {dict,0,16,16,8,80,48,
                        {[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]},
                          []}}}},
                undefined}
** Reason for termination =
** {{badmatch,none},
    [{riak_core_util,chash_key,1},
     {riak_kv_get_fsm,initialize,2},
     {gen_fsm,handle_msg,7},
     {proc_lib,init_p_do_apply,3}]}
```

BigCouch

0.4a

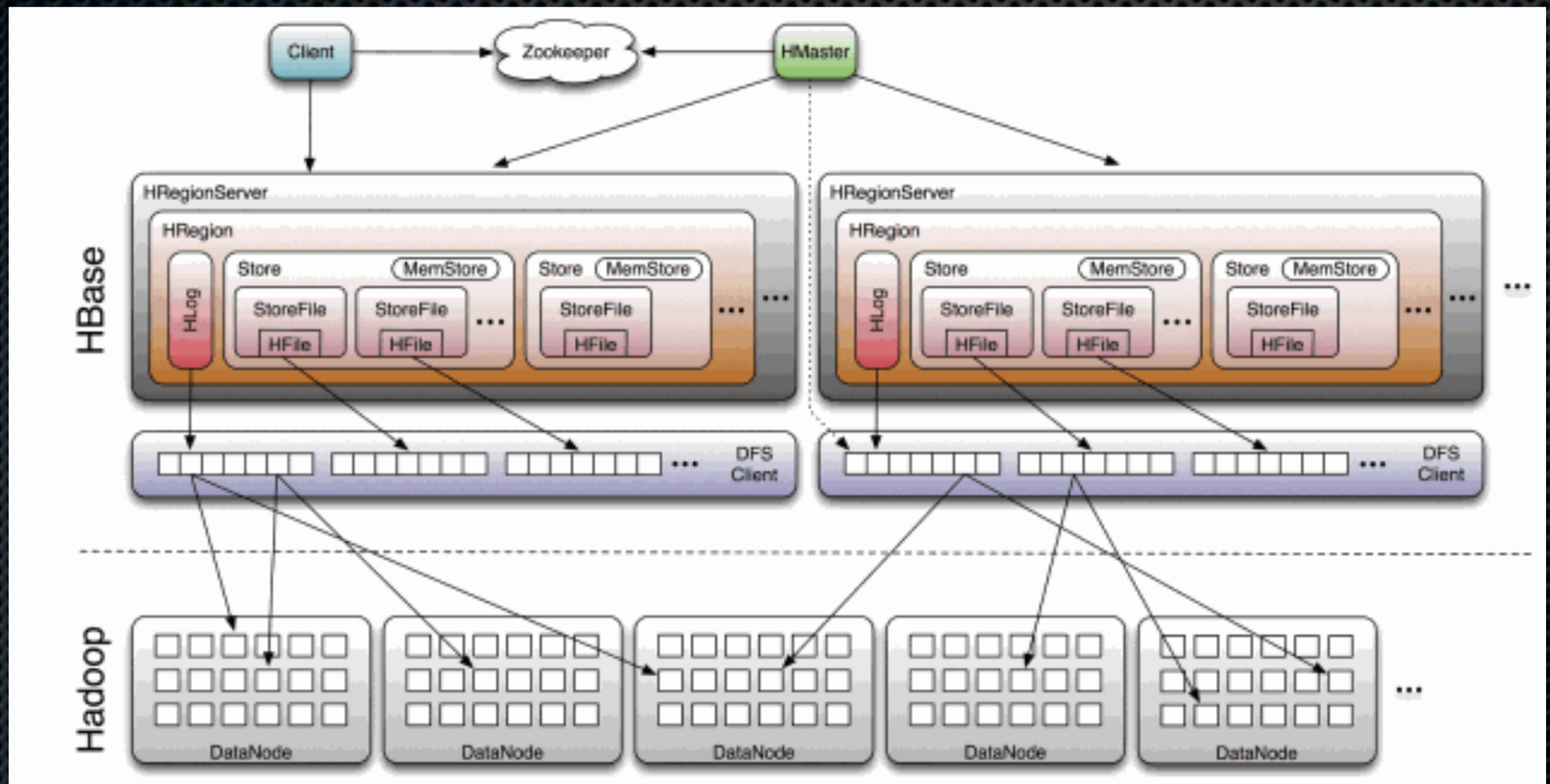
HBase

0.90

HBase in short

- First developed by Powerset to mimic Google's BigTable
- Today top-level Apache project.
- "random, realtime read/write access to your Big Data"
- "atop clusters of commodity hardware"
- "billions of rows X millions of columns"

HBase picture



HBase API

- Read
- Write
- Delete
- Atomic increments
- Scan table (in order)

HBase + Erlang?

Thrift

Need more speed

JInterface + HBase client (org.apache.hadoop.hbase.client)

JInterface + asynchbase (org.hbase.async.HBaseClient from stumbleupon)

Java fear

```
} else {  // We lost the second race.
          // Here we synchronize on two different references without any
          // apparent ordering guarantee, which can typically lead to
          // deadlocks.  In this case though we're fine, as any other thread
          // that still has a reference to `nsred_rpcs' is gonna go through
          // this very same code path and will lock `nsred_rpcs' first
          // before finding that it too lost 2 races, so it'll lock `added'
          // second.  So there's actually a very implicit ordering.
          if (can_retry_rpc) {
            synchronized (added) {  // Won't deadlock (explanation above).
              if (added.isEmpty()) {
                LOG.error("WTF?  Shouldn't happen!  Lost 2 races and found"
                          + " an empty list of NSRE'd RPCs (" + added
                          + ") for " + Bytes.pretty(region_name));
```

Pure-erlang HBase client?

asynchbase is about 12k lines of Java, how many lines erlang is that?

Summing up