Improving Running Components

Evan Weaver
Twitter, Inc.

QCon London, 2009

Many tools:

Rails

C

Scala

Java

MySQL

Rails front-end:

rendering
cache composition
db querying

Middleware:

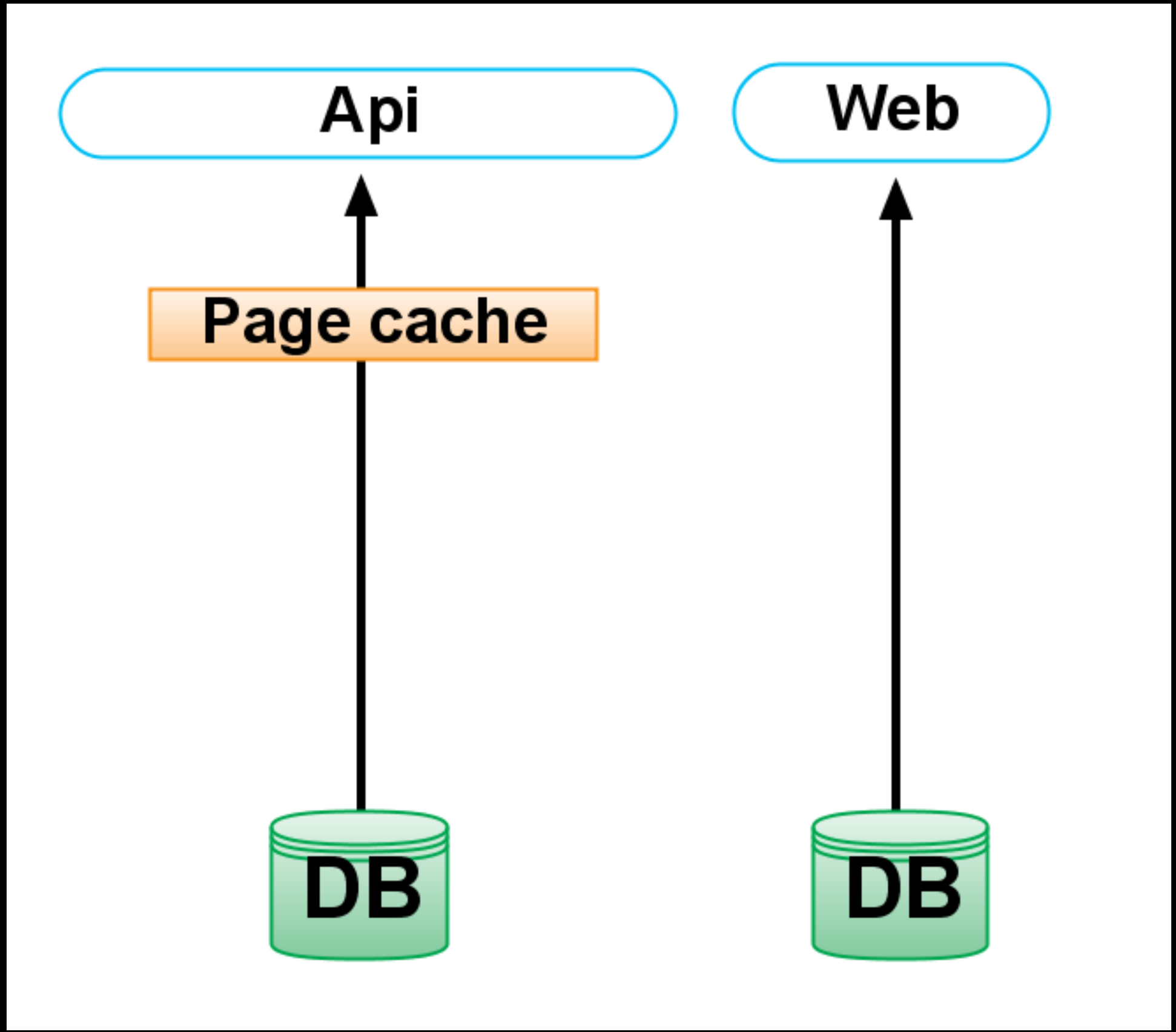
Memcached
Varnish (cache)
Kestrel (MQ)
comet server

Milestone 1: Cache policy

Optimization plan:

1. stop working
2. share the work
3. work faster

Old



Everything runs from memory in Web 2.0.

First policy change: vector cache

Stores arrays of tweet pkeys

Write-through

99% hit rate

Second policy change: row cache

Store records from the db
(Tweets and users)
Write-through
95% hit rate

Third policy change: fragment cache

Stores rendered version of tweets for the API

Read-through

95% hit rate

Fourth policy change: giving the page cache its own cache pool

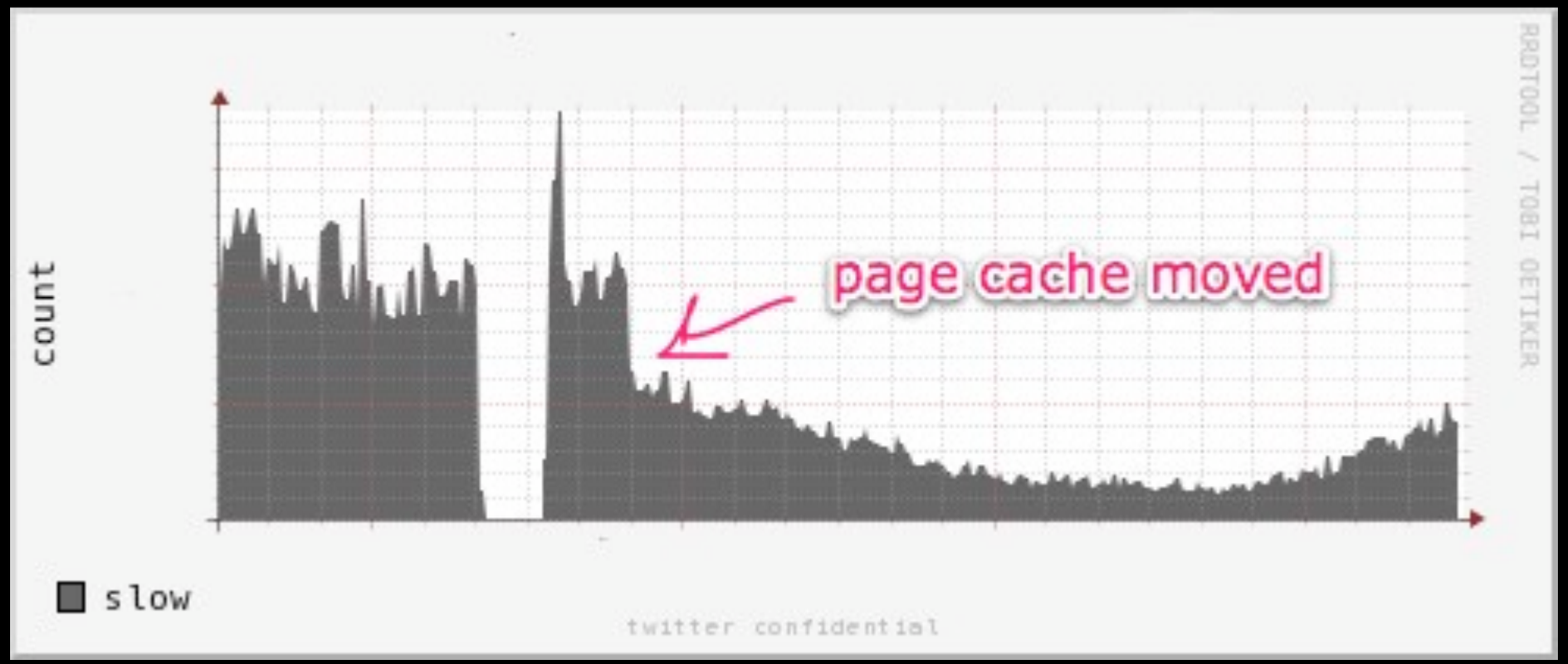
Generational keys
Low hit rate (40%)

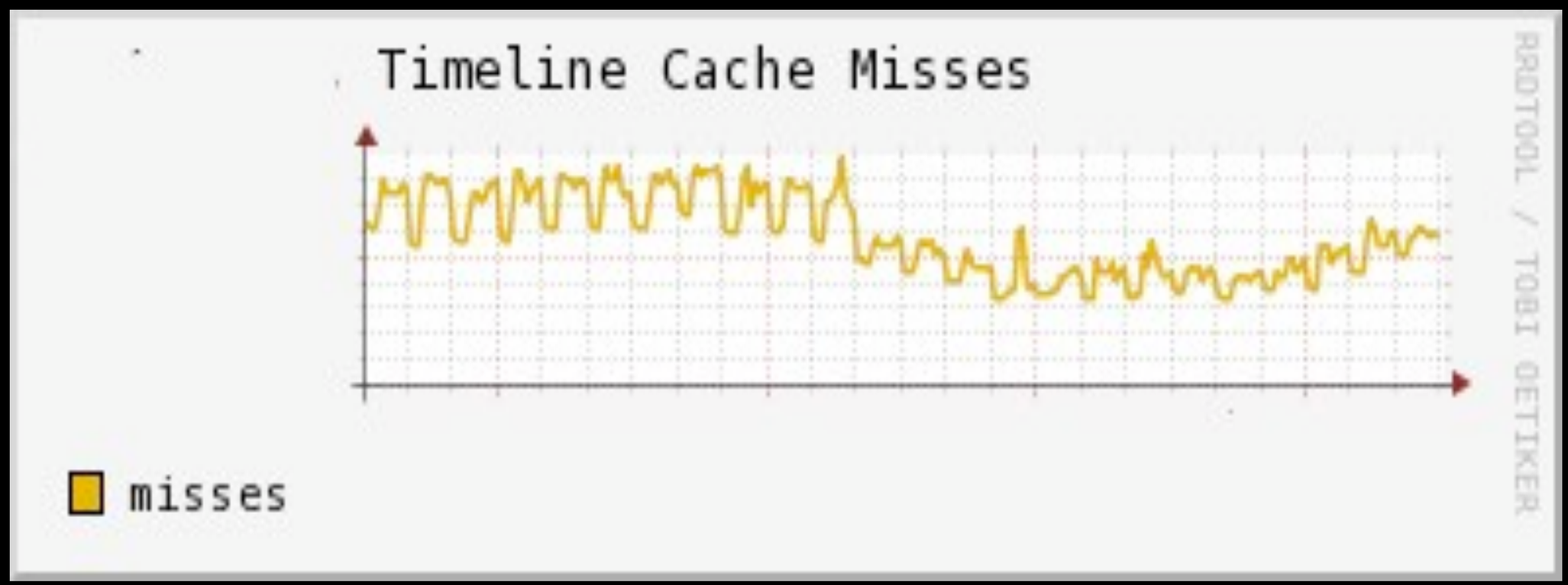
Visibility was lacking.

Peep tool
Dumps a live memcached heap

Cache only was living five hours

```
mysql> select round(round(log10(3576669 - last_read_time) * 5, 0) / 5, 1) as log, round(avg(3576669 - last_read_time), -2) as freshness, count(*),
rpad('', count(*) / 2000, '*') as bar from entries group by log order by log desc;
+------+-----------+----------+-----------------------------------------------------------------------------------------------------------------------+
| log  | freshness | count(*) | bar                                                                                                                   |
+------+-----------+----------+-----------------------------------------------------------------------------------------------------------------------+
| NULL |         0 |    13400 | *******                                                                                                               |
|  6.6 |   3328300 |      940 |                                                                                                                       |
|  6.2 |   1623200 |        1 |                                                                                                                       |
|  5.2 |    126200 |        1 |                                                                                                                       |
|  5.0 |     81100 |      343 |                                                                                                                       |
|  4.8 |     64800 |     3200 | **                                                                                                                    |
|  4.6 |     34800 |    18064 | *********                                                                                                             |
|  4.4 |     24200 |    96739 | ************************************************                                                                      |
|  4.2 |     15700 |   212865 | ****************************************************************************************************************      |
|  4.0 |     10200 |   224703 | ********************************************************************************************************************* |
|  3.8 |      6500 |   158067 | ********************************************************************************                                      |
|  3.6 |      4100 |   108034 | ****************************************************                                                                  |
|  3.4 |      2600 |    82000 | ****************************************                                                                              |
|  3.2 |      1600 |    65637 | ********************************                                                                                      |
|  3.0 |      1000 |    49267 | ************************                                                                                              |
|  2.8 |       600 |    34398 | *****************                                                                                                     |
|  2.6 |       400 |    24322 | ************                                                                                                          |
|  2.4 |       300 |    19865 | **********                                                                                                            |
|  2.2 |       200 |    14810 | *******                                                                                                               |
|  2.0 |       100 |    10108 | *****                                                                                                                 |
|  1.8 |       100 |     8002 | ****                                                                                                                  |
|  1.6 |         0 |     6479 | ***                                                                                                                   |
|  1.4 |         0 |     4014 | **                                                                                                                    |
|  1.2 |         0 |     2297 | *                                                                                                                     |
|  1.0 |         0 |     1733 | *                                                                                                                     |
|  0.8 |         0 |      649 |                                                                                                                       |
|  0.6 |         0 |      710 |                                                                                                                       |
|  0.4 |         0 |      672 |                                                                                                                       |
|  0.0 |         0 |      319 |                                                                                                                       |
+------+-----------+----------+-----------------------------------------------------------------------------------------------------------------------+
```


count
page cache moved
slow
twitter confidential
RRDTOOL / TOBI OETIKER


Timeline Cache Misses
misses
RRDTOOL / TOBI OETIKER

What does a timeline miss mean?

Container union

/home rebuild reads through your followings' profiles

Api

Web

Page cache

Fragment cache

Row cache

Vector cache

DB

DB

DB

Memcached

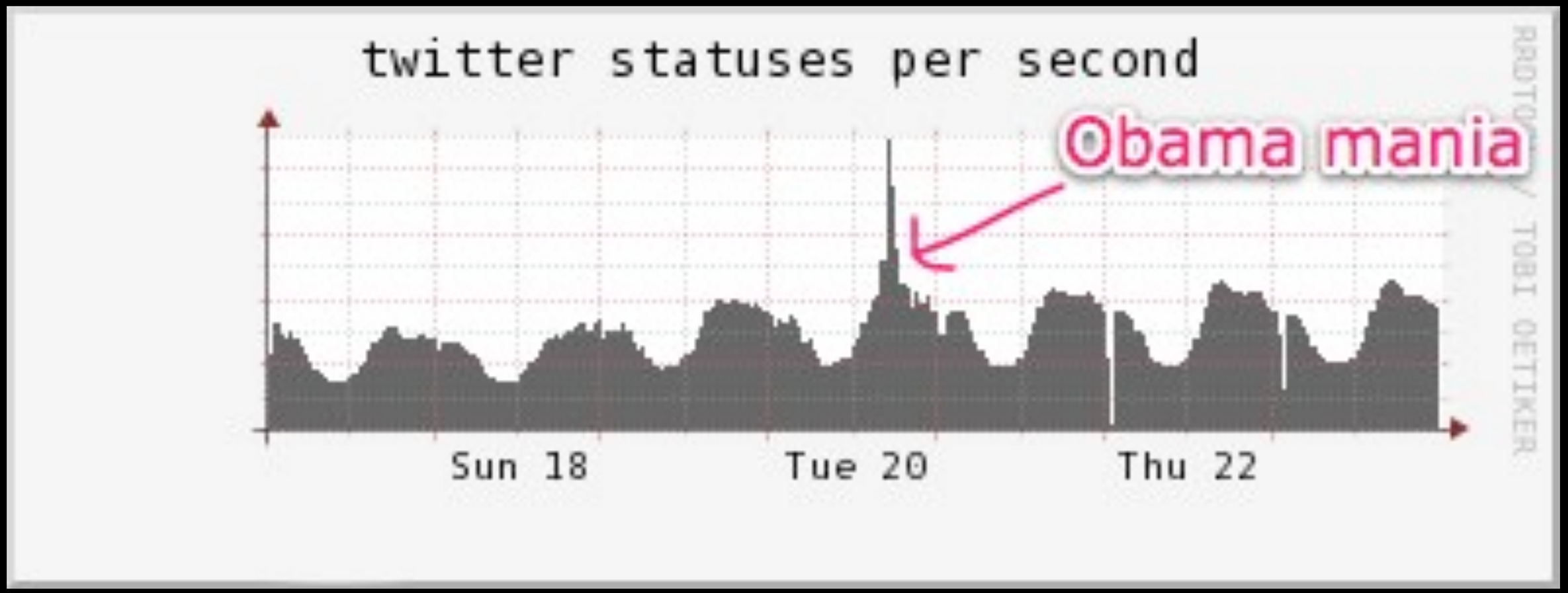
Milestone 2:
Message queue

A component with problems

Purpose in a web app:

Move operations out of the synchronous request cycle

Amortize load over time



Inauguration, 2009

Simplest MQ ever:

Gives up constraints for scalability
No strict ordering of jobs
No shared state among servers
Just like memcached
Uses memcached protocol

First version was written in Ruby

Ruby is “optimization-resistant”
Mainly due to the GC

If the consumers could not keep pace, the MQ would fill up and crash

Ported it to Scala for this reason

Good tooling
for the Java GC:

JConsole
Yourkit

Heap Memory Usage

Time Range: All

Perform GC

ordinary times:
latency = 2ms avg, 4ms max

2G serious limit

little sawteeth
are eden collections
(eden = 128M)

1.7M items
in queue
(1 item = 1KB)

rough times:
latency = 5ms avg, 35ms max

14:55 15:00 15:05 15:10 15:15

me: 2008-10-17 15:17:30
sed: 1,888,719 kbytes
ed: 2,096,960 kbytes

100% --
75% --

Poor tooling for the Ruby GC:

Railsbench w/patches
BleakHouse w/patches
Valgrind/Memcheck
MBARI 1.8.6 patches

Our Railsbench GC tunings

35% speed increase

RUBY_HEAP_MIN_SLOTS=500000
RUBY_HEAP_SLOTS_INCREMENT=250000
RUBY_HEAP_SLOTS_GROWTH_FACTOR=1
RUBY_GC_MALLOC_LIMIT=50000000
RUBY_HEAP_FREE_MIN=4096

Situational decision:

Scala is a flexible language
(But libraries a bit lacking)
We have experienced JVM
engineers

Big rewrites fail...?

Small rewrite:

No new features added

Well-defined interface

Already went over the wire

Deployed to 1 MQ host

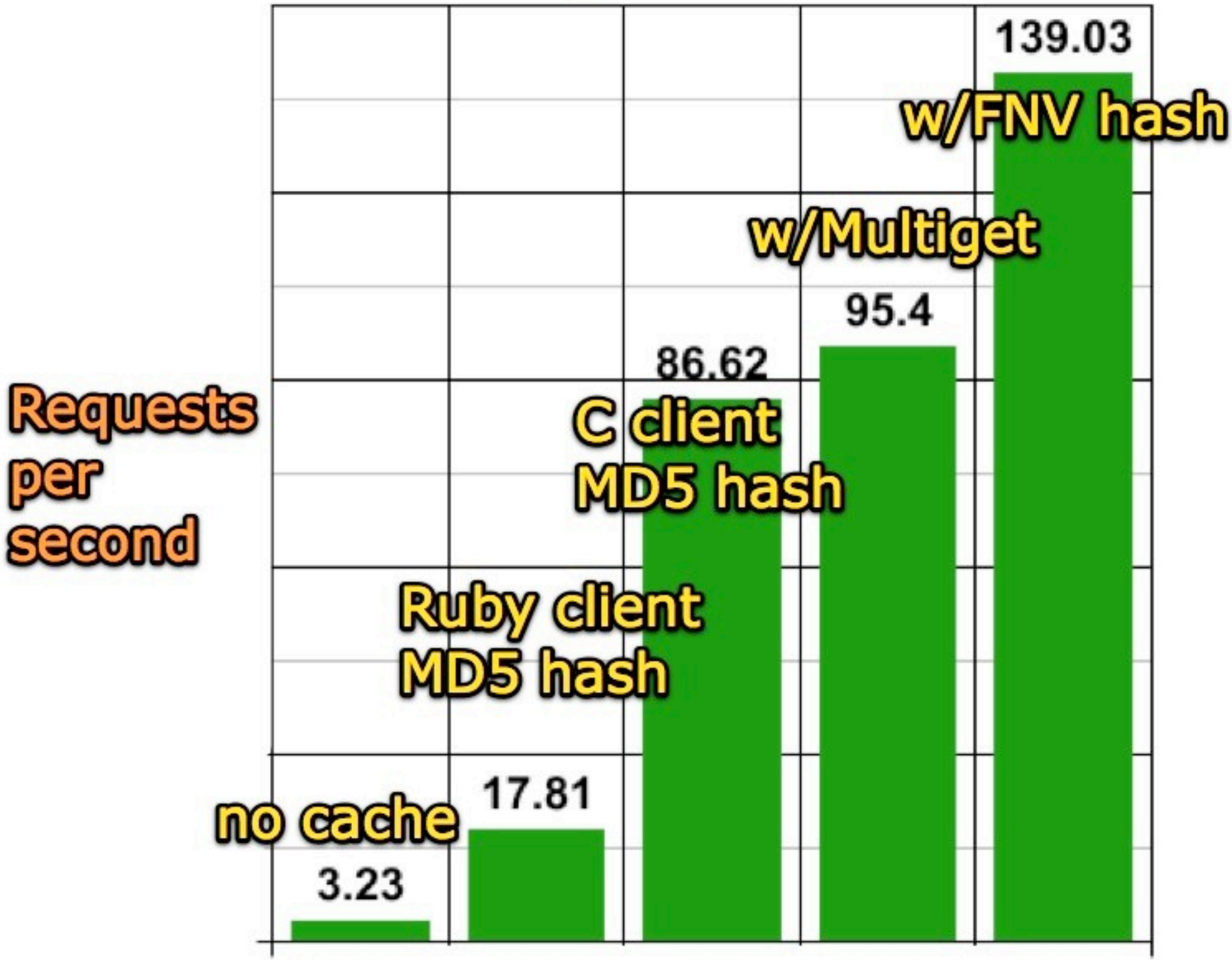
Fixed regressions

Eventually deployed to all hosts

Milestone 3:
the memcached client

Optimizing a critical path

Fragment cache



Switched to libmemcached, a new C Memcached client

We are now the biggest user and biggest 3rd-party contributor

Uses a SWIG Ruby binding I started a year or so ago

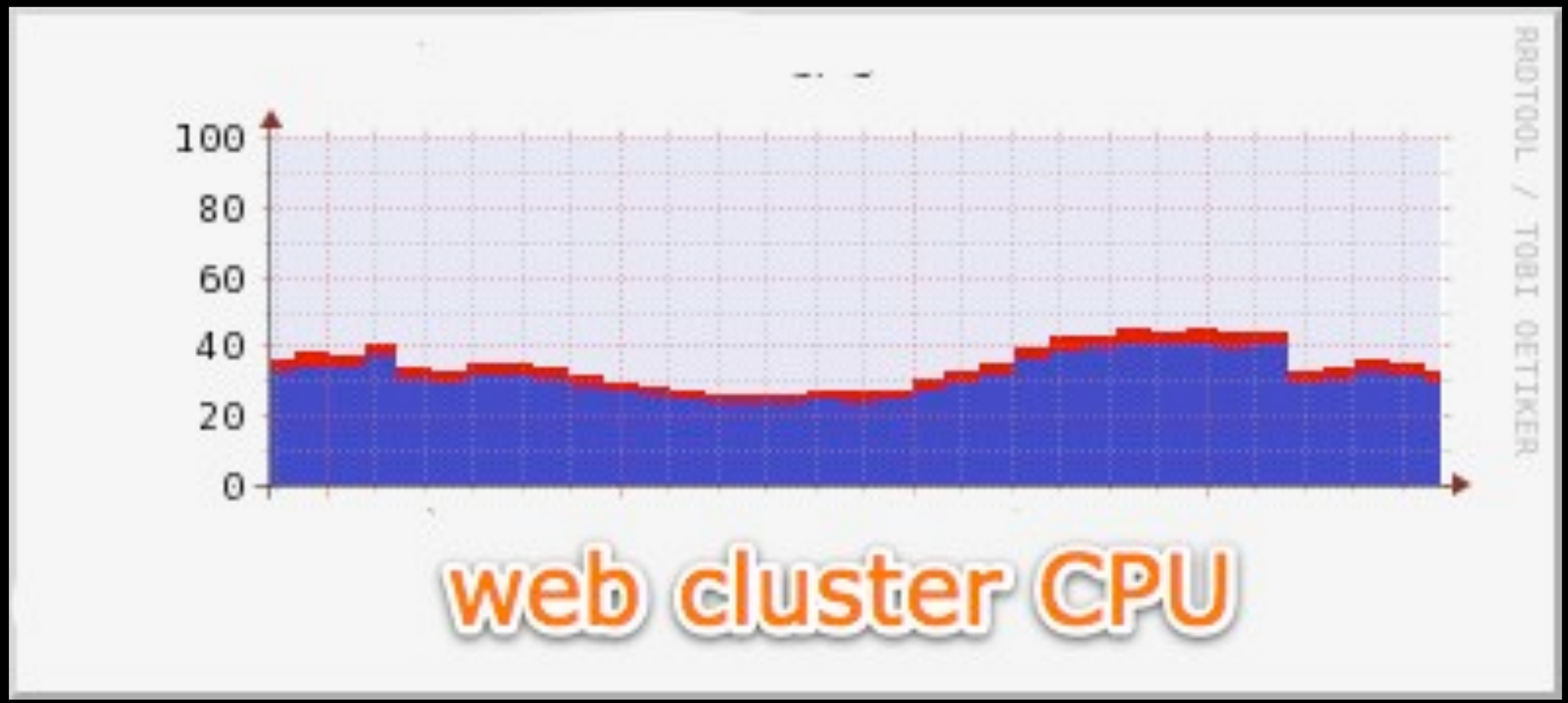
Compatibility among memcached clients is critical

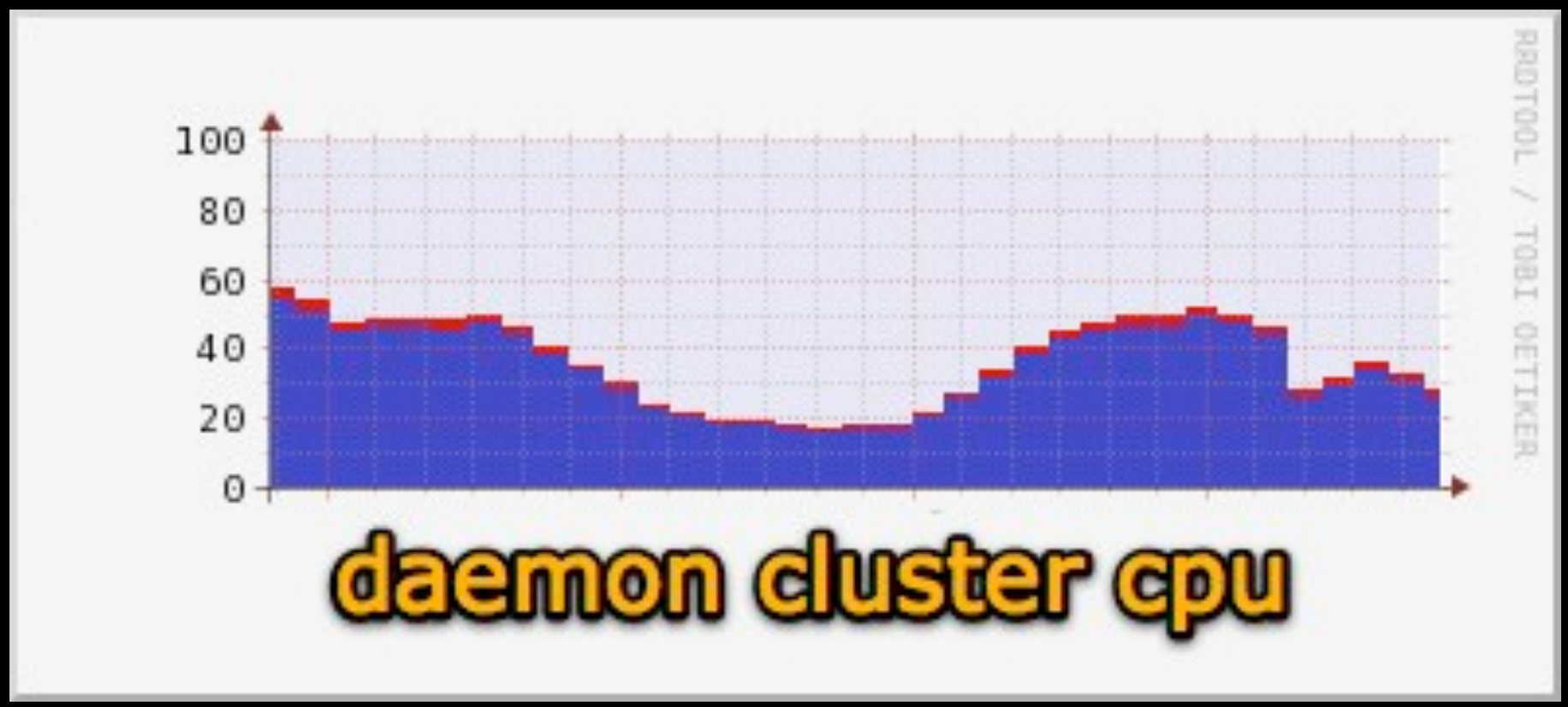
Twitter is big, and runs hot

Flushing the cache would be catastrophic

Spent endless time on backwards compatibility

A/B tested the new client over 3 months


100
80
60
40
20
0
RRDTOOL / TOBI OETIKER
web cluster CPU


100
80
60
40
20
0
RRDTOOL / TOBI OETIKER
daemon cluster cpu

MQ also benefitted

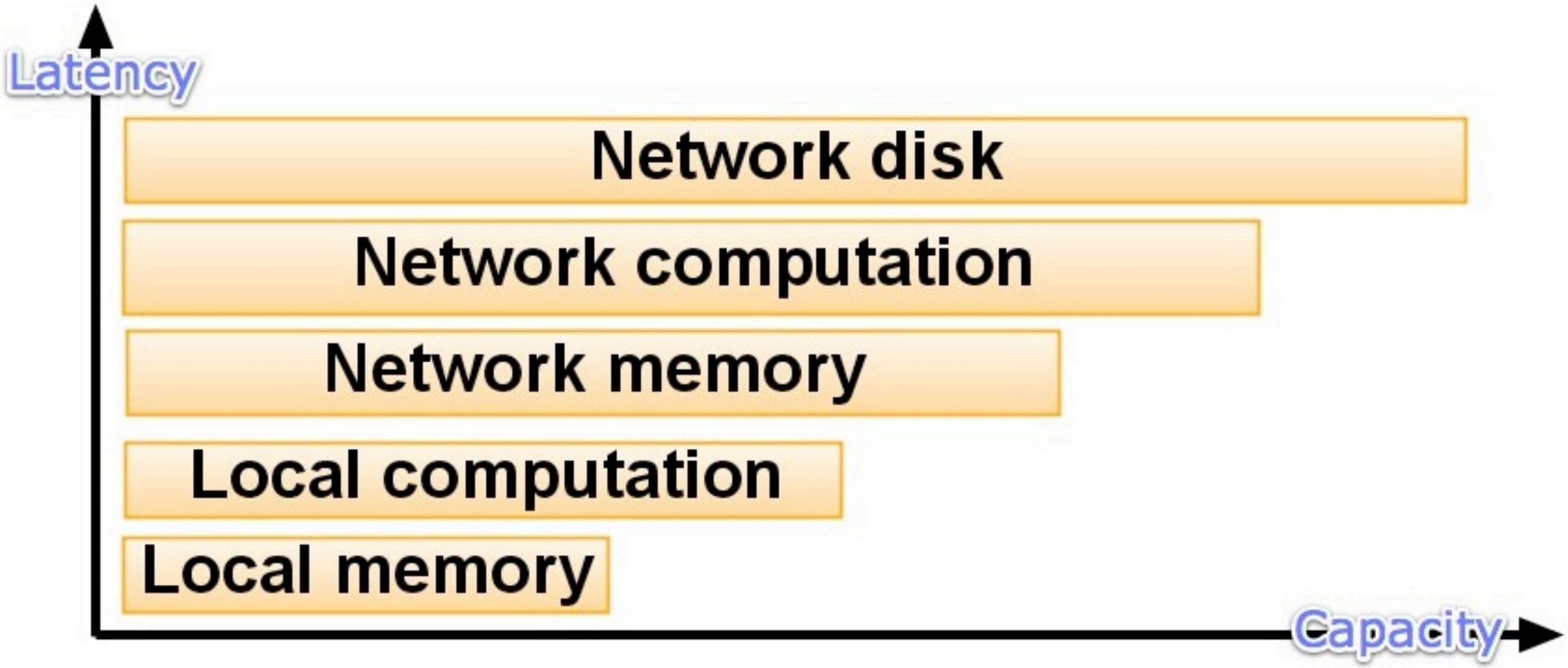
Memcached can be a generic lightweight service protocol

We also use Thrift and HTTP internally

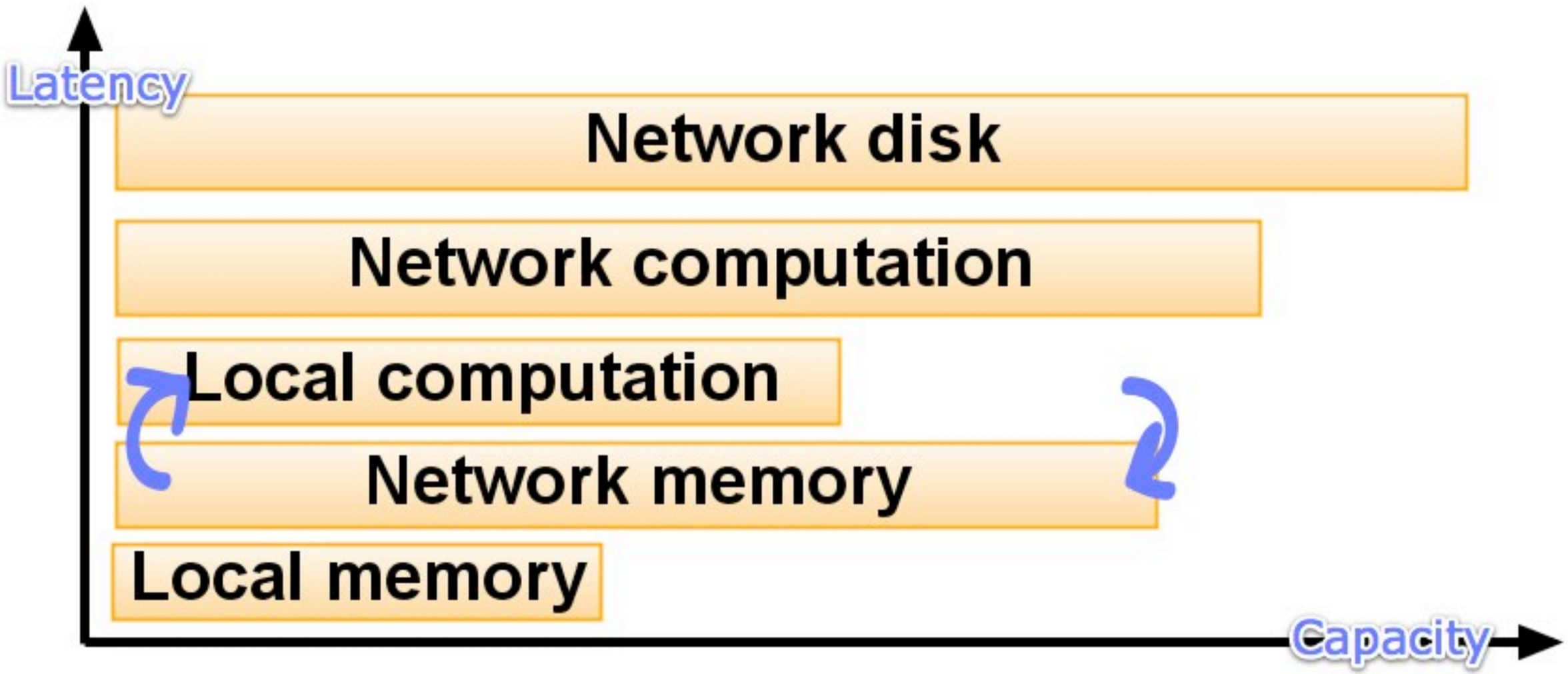
So many RPCs! Sometimes 100s of Memcached round trips per request.

"As a memory device gets larger, it tends to get slower."

Performance hierarchy is supposed to look like:



At web scale, it looks more like:



End

Links:

C tools:

- Peep http://github.com/fauna/peep/
- Libmemcached http://tangent.org/552/libmemcached.html
- Valgrind http://valgrind.org/

JVM tools:

- Kestrel http://github.com/robey/kestrel/
- Smile http://github.com/robey/smile/
- Jconsole http://openjdk.java.net/tools/svc/jconsole/
- Yourkit http://www.yourkit.com/

Ruby tools:

- BleakHouse http://github.com/fauna/bleak_house/
- Railsbench Ruby patches http://github.com/skaes/railsbench/
- MBARI Ruby patches http://github.com/brentr/matzruby/tree/v1_8_6_287-mbari

General:

- Danga stack http://www.danga.com/words/2005_oscon/oscon-2005.pdf
- Seymour Cray quote http://books.google.com/books?client=safari&id=qM4Yzf8K9hwC&dq=rapid+development&q=cray&pgis=1
- Last.fm downtime http://blog.last.fm/2008/04/18/possible-lastfm-downtime

twitter.com/evan

blog.evanweaver.com

cloudbur.st