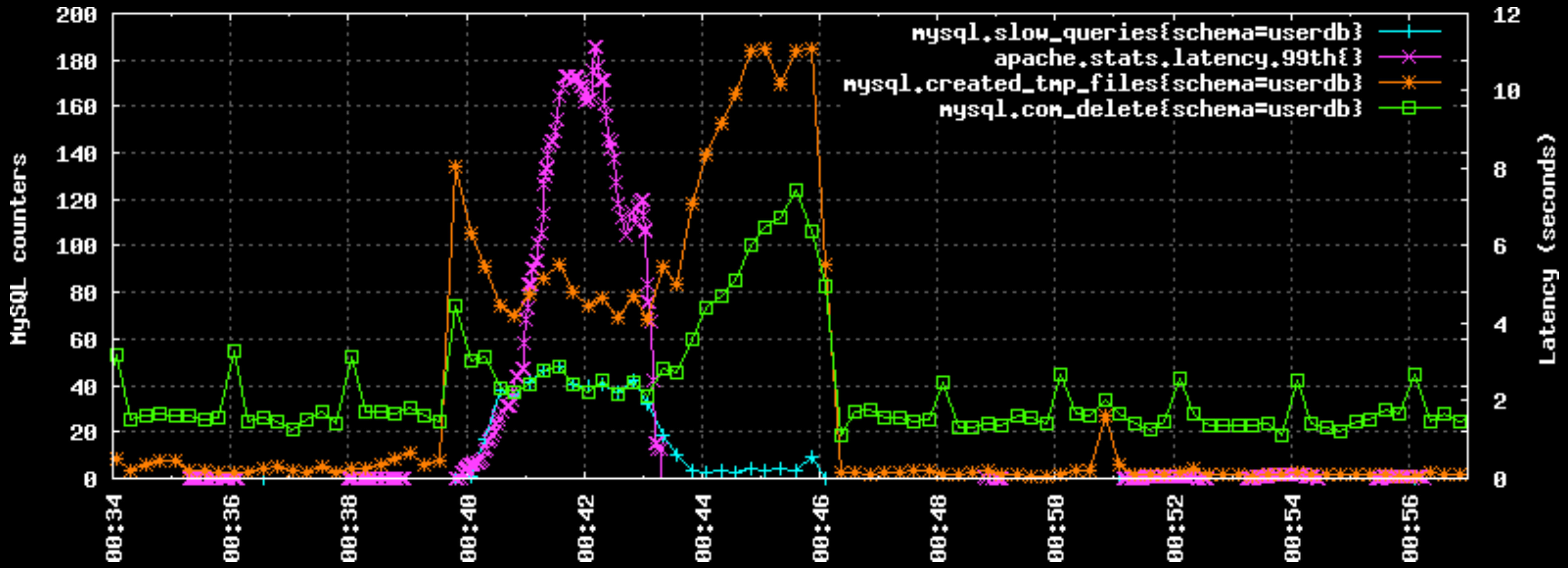
OpenTSDB

The Distributed, Scalable, Time Series Database
For your modern monitoring needs

Collect, store and serve billions of data points
with no loss of precision



15464 points retrieved, 932 points plotted in 100ms

StumbleUpon

Benoît "tsuna" Sigoure
tsuna@stumbleupon.com

Tired of 10+ year old monitoring systems?

Common problems include:

- Centralized data storage (SPoF)
- Limited storage space
- Data deteriorates over time
- Plotting a custom graph is hard
- Doesn't scale to:
  - >>10s of billions of data points
  - >1000s of metrics
  - New data every few seconds

Copyright 2010 - Richard Barber

Old Plow

OpenTSDB

- First open-source monitoring system built on an open-source distributed database
- Collect **all** the metrics you can imagine every few seconds
- Store them forever
- Retain granular data
- Make custom graphs on the fly
- Plug it into your alerting system
- Do capacity planning

Let's take a deep dive inside

203 by David Clavel

HBase

Distributed

Scalable

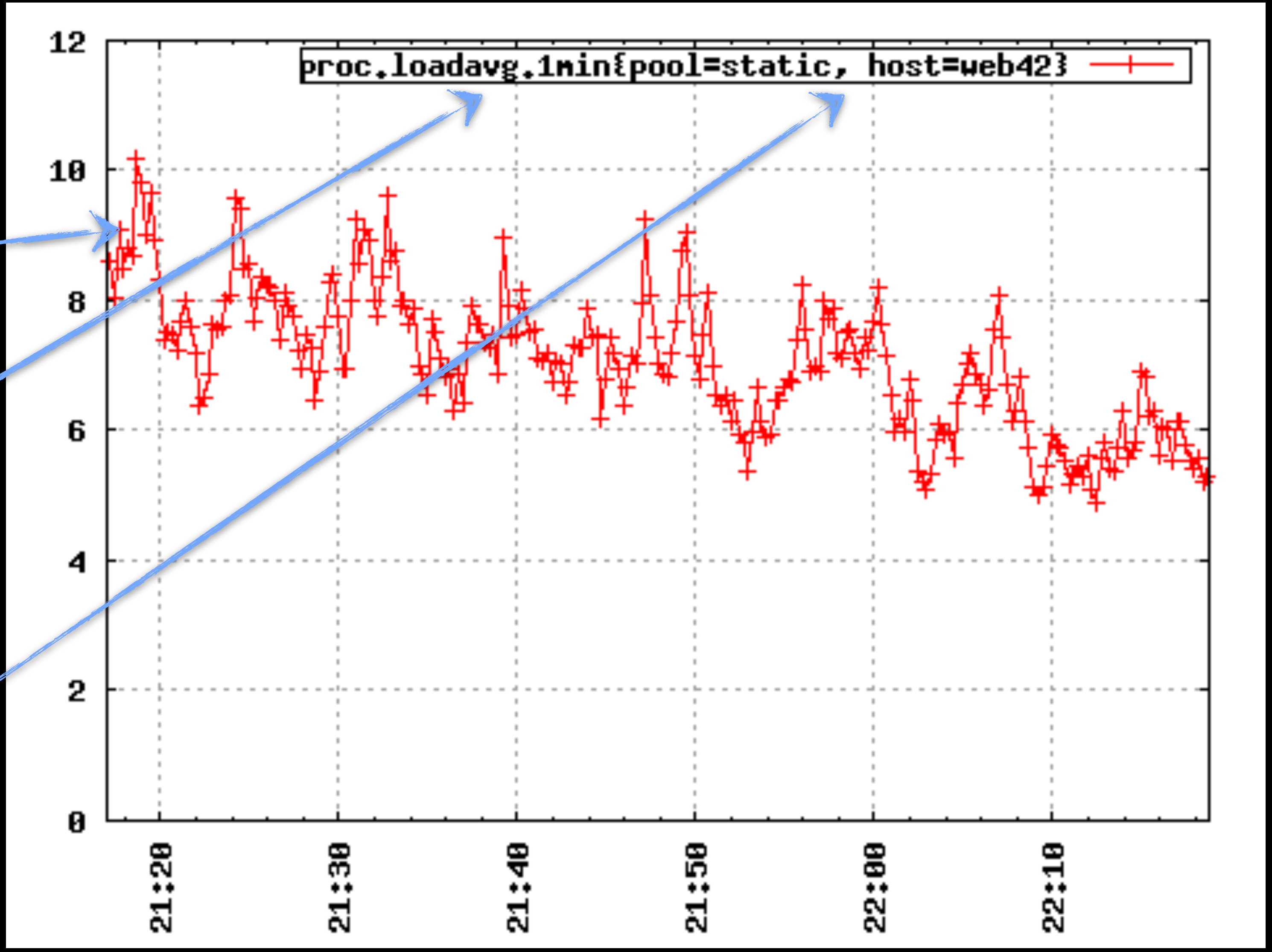
Reliable

Efficient

Bright Ideas by purplemattfish

Key concepts

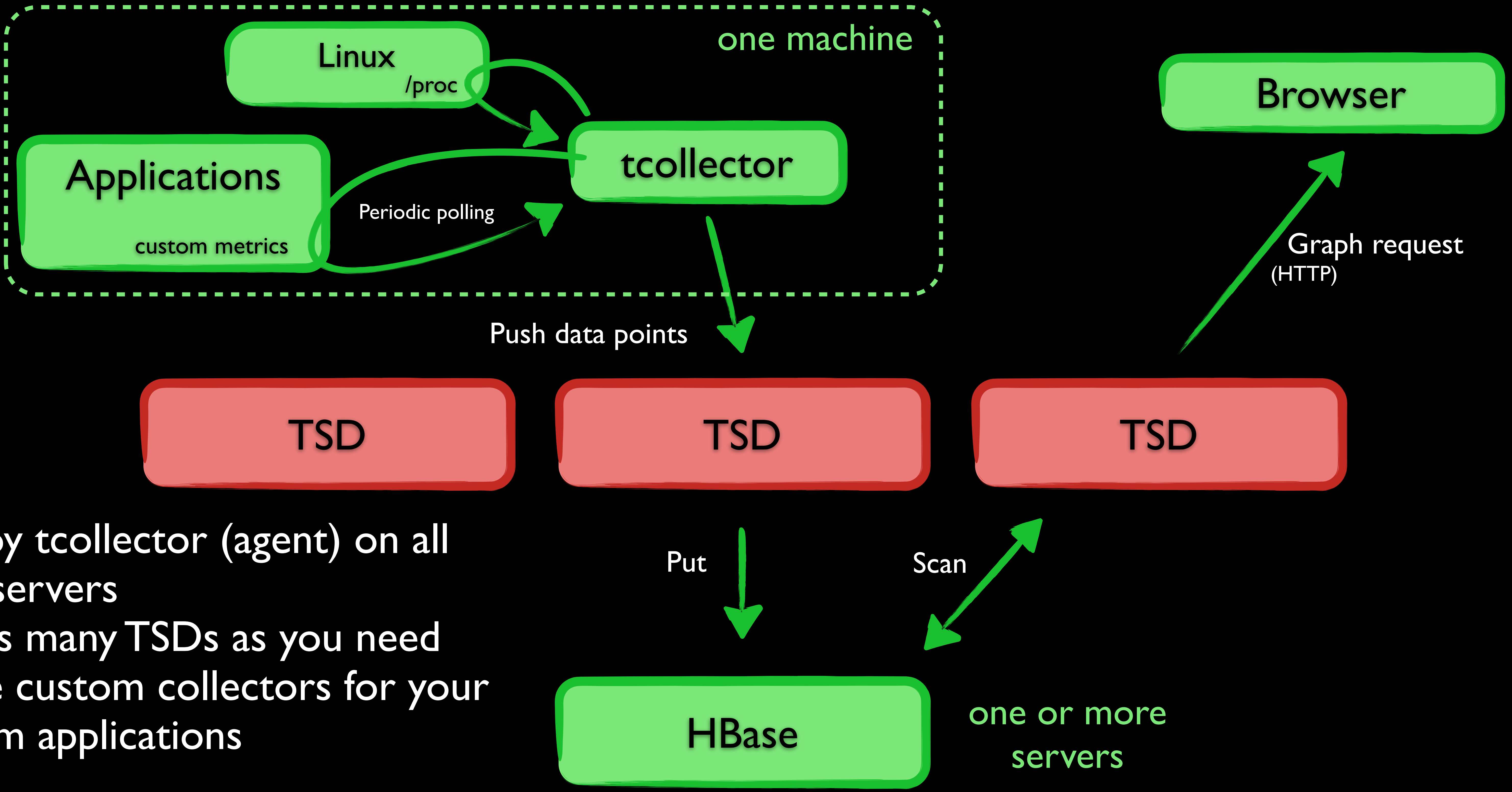
- Data Points
  (time, value)
- Metrics
  proc.loadavg.1m
- Tags
  host=web42 pool=static
- Metric + Tags = Time Series



```
put proc.loadavg.1m 1234567890 0.42 host=web42 pool=static
```

OpenTSDB's
push model

The Big Picture™



- Deploy tcollector (agent) on all your servers
- Run as many TSDs as you need
- Write custom collectors for your custom applications

12 Bytes Per Datapoint

4TB per year for 1000 machines

2 to 3 ~~12~~ Bytes Per Datapoint

What's new?

- Faster write path
- Two `fsck`-type tools
  (because sh*t happens)
- Wider rows
- More memory efficient

Misc:

- More unit tests
- Forward compatibility with future variable length encoding
- Improved build system

What's hot (just in for OSCON)

- Compacted rows / improved schema
  BETA (reduces data size by 6x, allows reading >6M points/s)

OpenTSDB @

600

(4x growth in 6 months)

~~150~~ Million Datapoints/Day

in a typical datacenter

(after 5x LZO compression)

- Over 70 billion data points stored (only 720GB on disk)
- 1 year anniversary as the main production monitoring system
- Completely replaced Ganglia + Munin + Cacti mix

Demo Time!

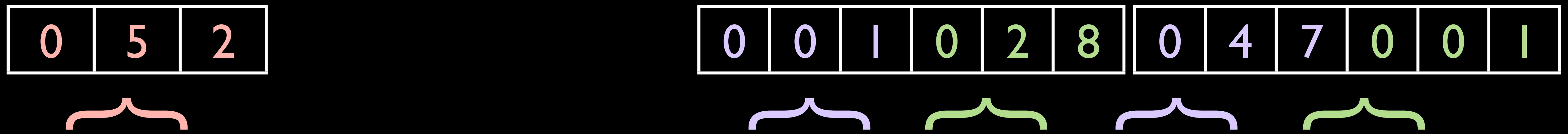
Recipe For Good Performance

- #1 rule: keep good data locality
- Know your access pattern
- Use a key structure that yields good locality for your access pattern
- Avoid wide rows with big keys and many small cells
- OpenTSDB's secret ingredient: asynchbase
  - Fully asynchronous, non-blocking HBase client
  - Written from the ground up to be thread-safe for server apps
  - Far fewer threads, far less lock contention, uses less memory
  - Provides more throughput, especially for write-heavy workloads

Table: tsdb-uid

Inside HBase

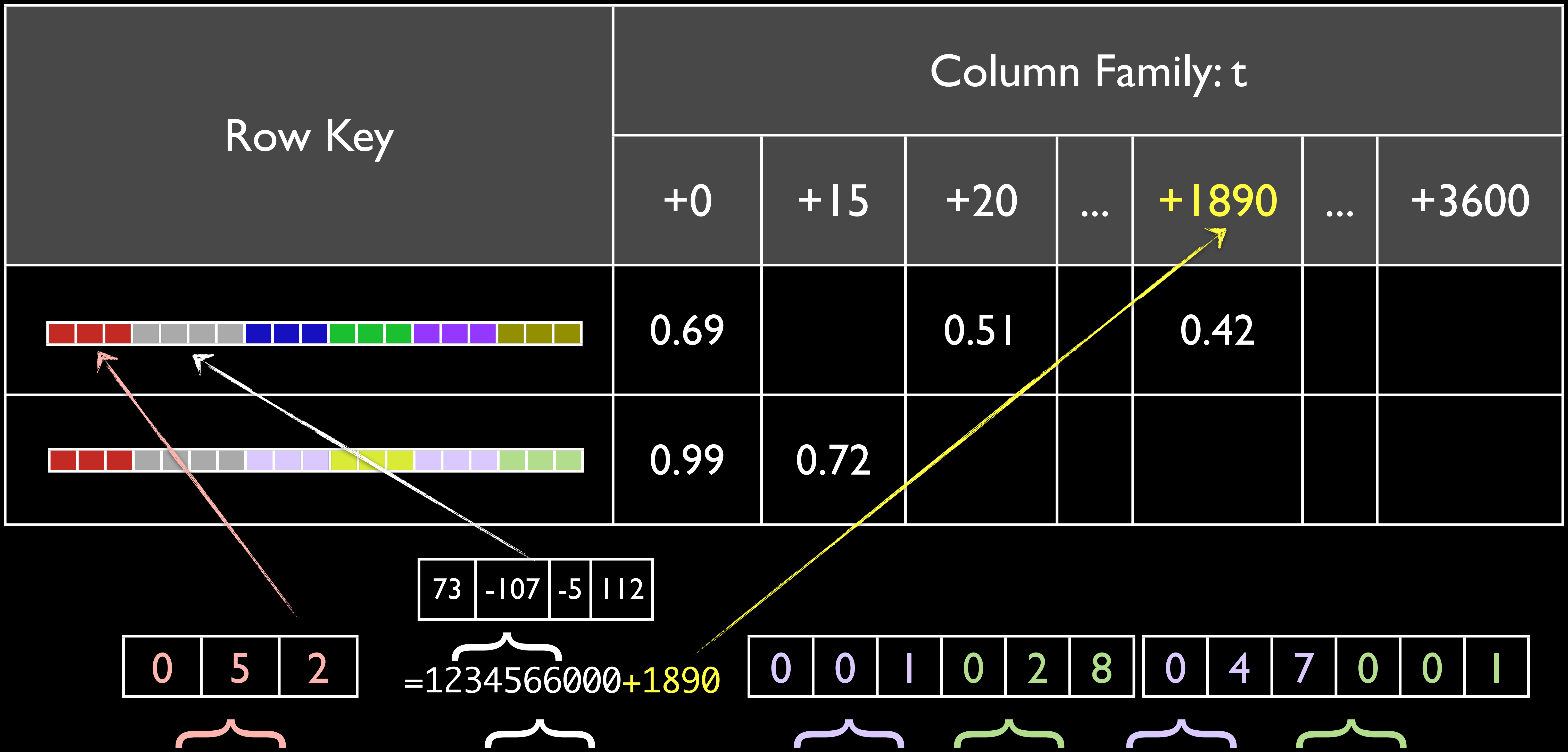
Row Key	Column Family: name			Column Family: id		
	metrics	tagk	tagv	metrics	tagk	tagv
0 0 1		host	static			
0 5 2	proc.loadavg.1m					
host					0 0 1	
proc.loadavg.1m				0 5 2		



`put proc.loadavg.1m 1234567890 0.42 host=web42 pool=static`

Table: tsdb

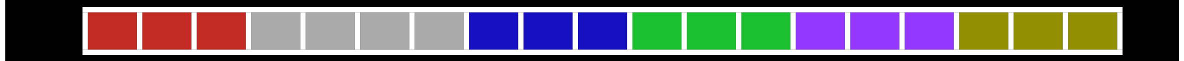
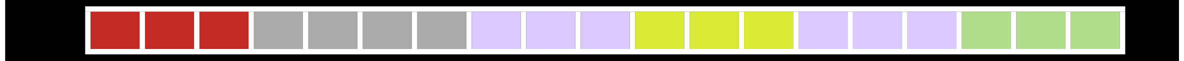
Inside HBase



Row Key	Column Family: t						
	+0	+15	+20	...	+1890	...	+3600
	0.69		0.51		0.42		
	0.99	0.72					

```
put proc.loadavg.1m 1234567890 0.42  host=web42  pool=static
```

Implications of the Schema

Row Key	Column Family: t						
	+0	+15	+20	...	+1890	...	+3600
	0.69		0.51		0.42		
	0.99	0.72					

- Queries always need data points for a metric and time range
- All data points for a given metric next to each other
- All data points for a time range next to each other
- Compact data + data locality = efficient range scans
- Tag filtering is pushed down to the HBase server

TSDB Compactions

Row Key	Column Family: t						
	+0	...	+10	...	+25	...	...
🟥🟥🟥⬜⬜⬜⬜🟦🟦🟦🟩🟩🟩🟪🟪🟪🟨🟨🟨	0.69		0.51		0.42		

TSDB Compactions

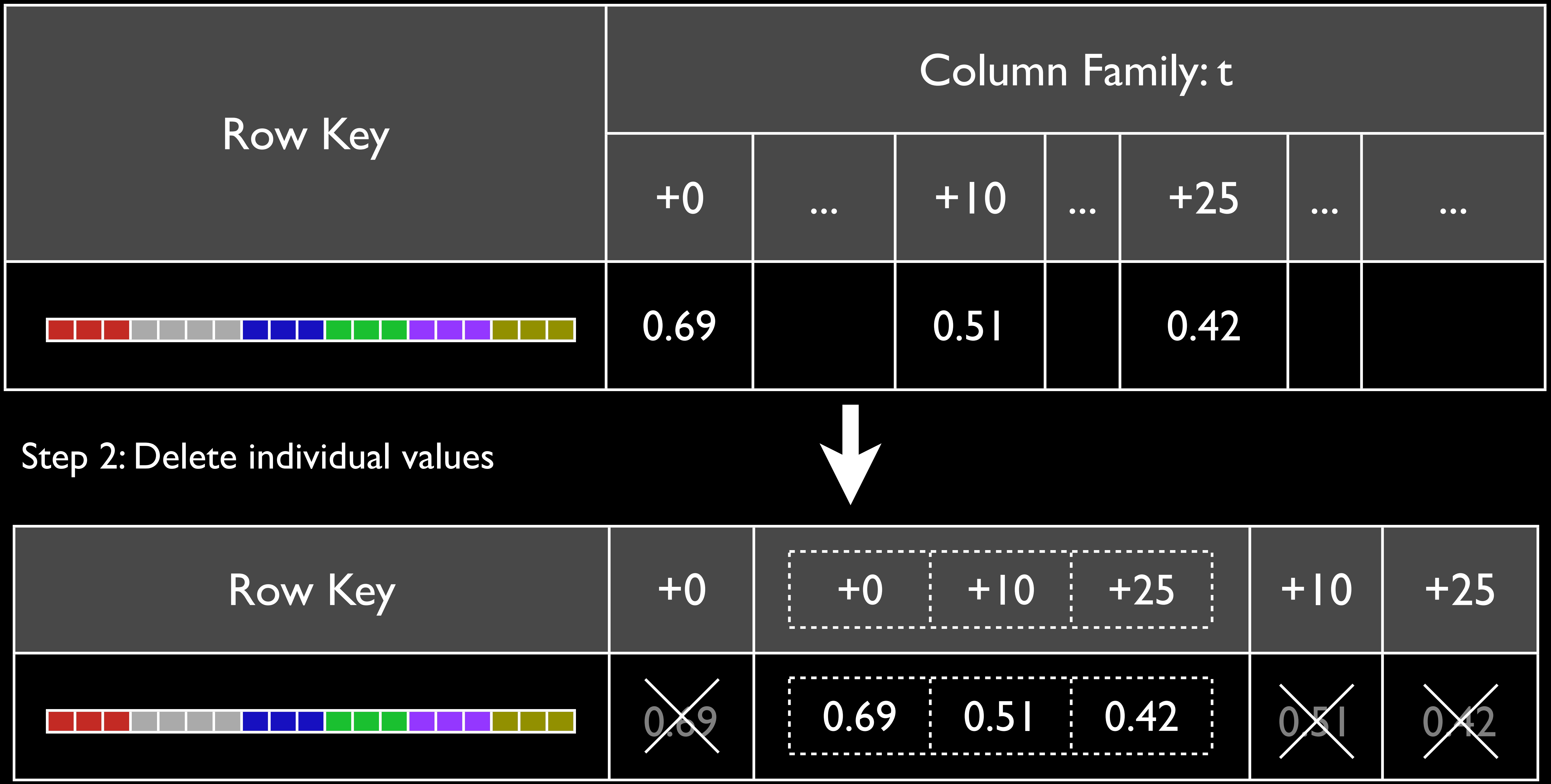
Row Key	Column Family: t						
	+0	...	+10	...	+25	...	...
🟥🟥🟥⬜⬜⬜⬜🟦🟦🟦🟩🟩🟩🟪🟪🟪🟨🟨🟨	0.69		0.51		0.42		

Step 1: Concatenate all columns and values

↓

Row Key	+0	+0 +10 +25	+10	+25
🟥🟥🟥⬜⬜⬜⬜🟦🟦🟦🟩🟩🟩🟪🟪🟪🟨🟨🟨	0.69	0.69 0.51 0.42	0.51	0.42

TSDB Compactions



Row Key	Column Family: t						
	+0	...	+10	...	+25	...	...
🟥🟥🟥⬜⬜⬜⬜🟦🟦🟦🟩🟩🟩🟪🟪🟪🟨🟨🟨	0.69		0.51		0.42		

Step 2: Delete individual values

Row Key	+0	+0 +10 +25	+10	+25
🟥🟥🟥⬜⬜⬜⬜🟦🟦🟦🟩🟩🟩🟪🟪🟪🟨🟨🟨	~~0.69~~	0.69 0.51 0.42	~~0.51~~	~~0.42~~

100% Natural, Organic Free & Open-Source

Danger in the Corn by Roger Smith

Fork me on GitHub

Fork me on GitHub

¿ Questions ?

opentsdb.net

Liked what you saw?
Set it up in 15 minutes

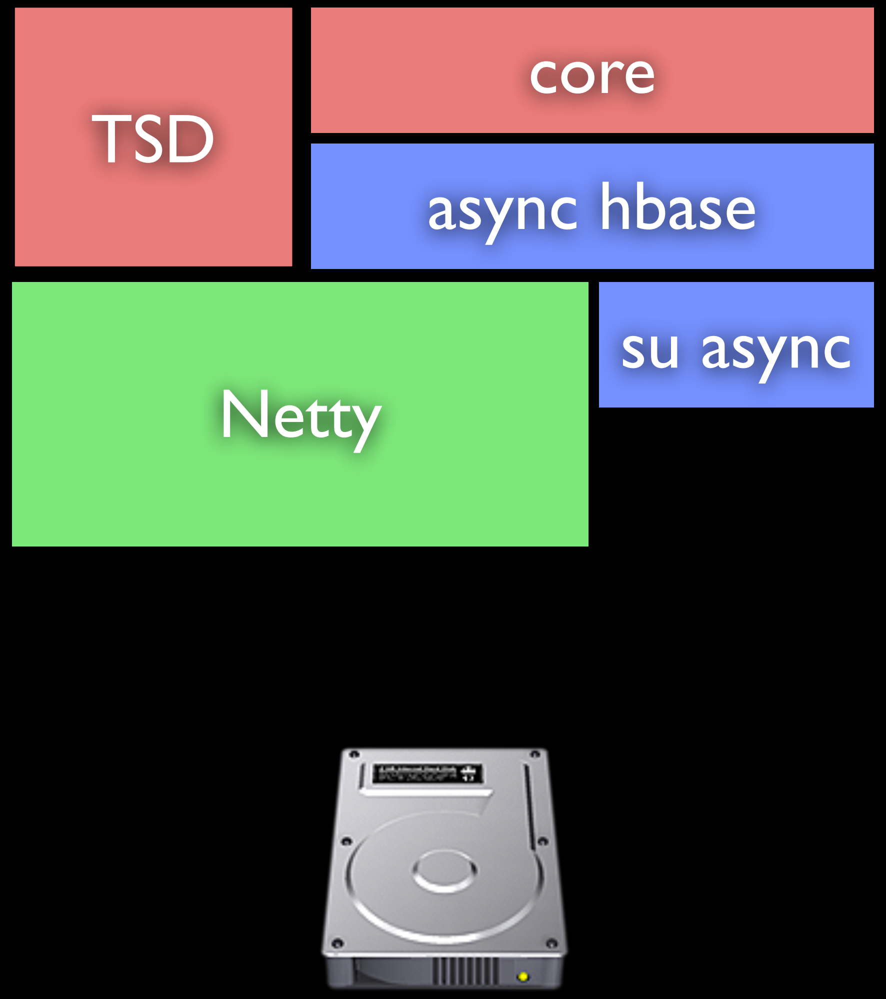
JDK + Gnuplot	1 minute (1 command)
Single-node HBase	4 minutes (3 commands)
OpenTSDB	5 minutes (5 commands)
Deploy tcollector	5 minutes

StumbleUpon

Think this is cool?
We're hiring

Benoît "tsuna" Sigoure
tsuna@stumbleupon.com

Under the Hood



TSD

core

async hbase

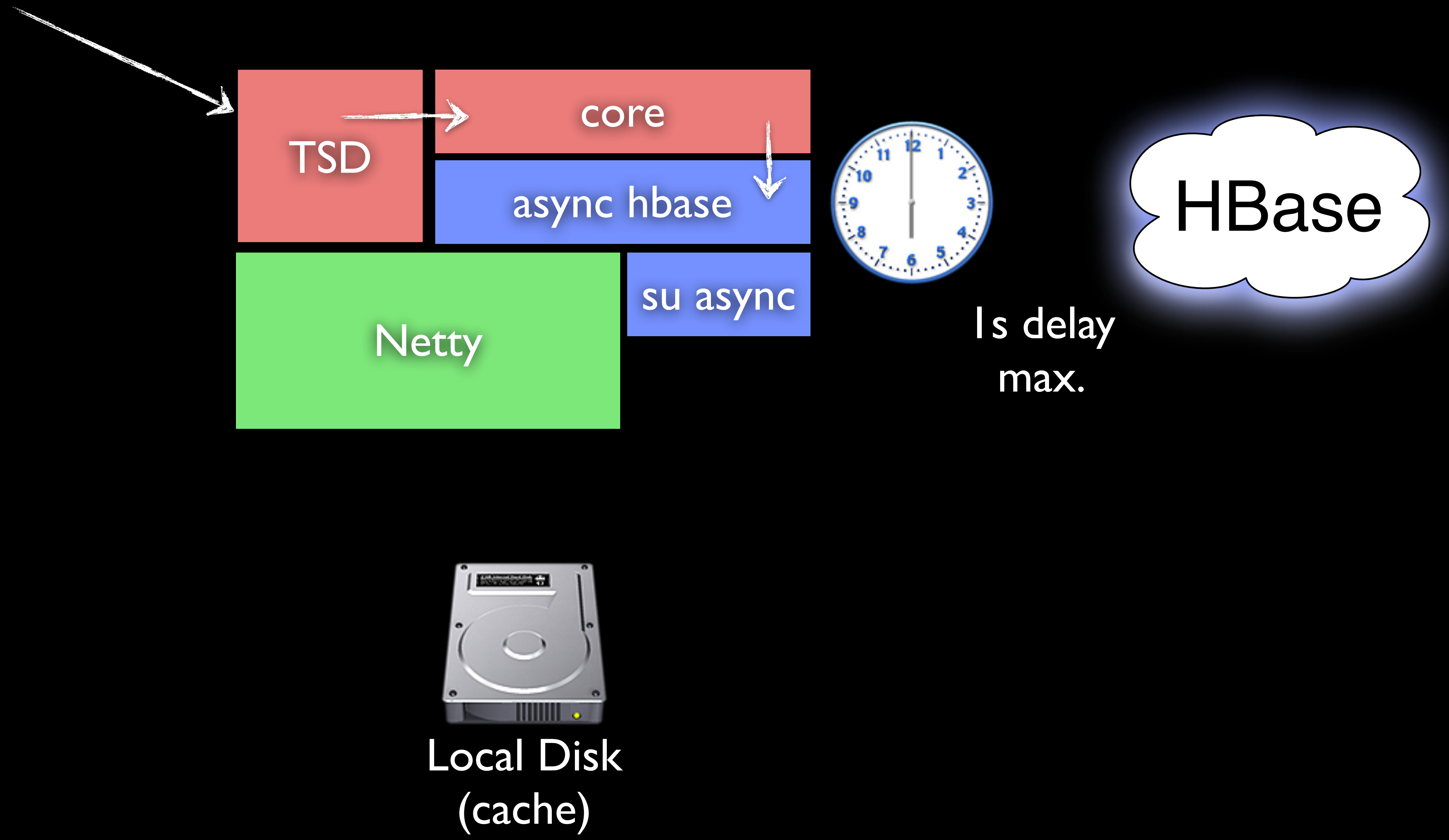
Netty

su async

Local Disk
(cache)

Under the Hood

```
put proc.loadavg.1m 1234567890 0.42 host=web42 pool=static
```

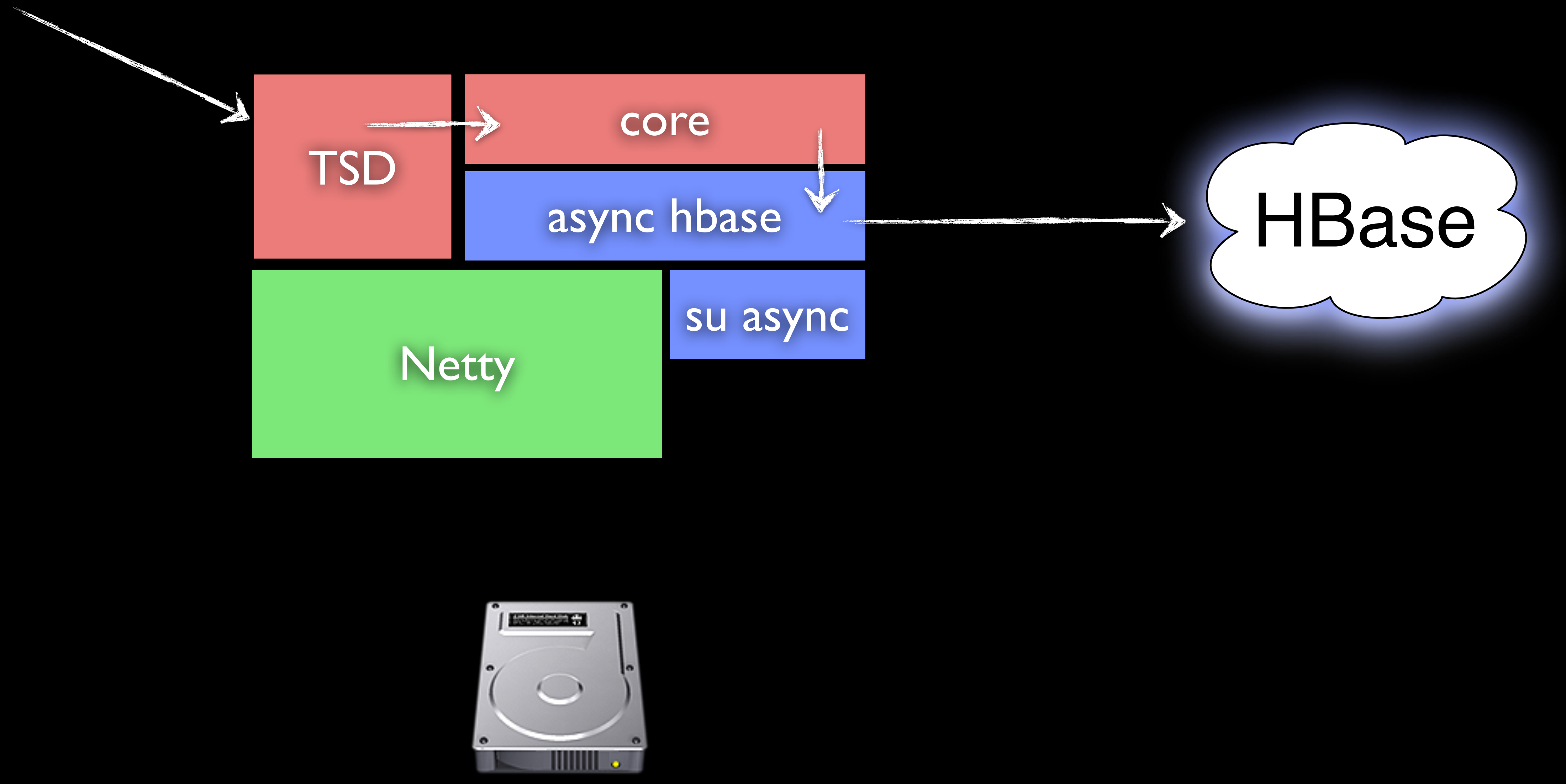


Write Path

>2000 data points / sec / core

Under the Hood

```
put proc.loadavg.1m 1234567890 0.42 host=web42 pool=static
```

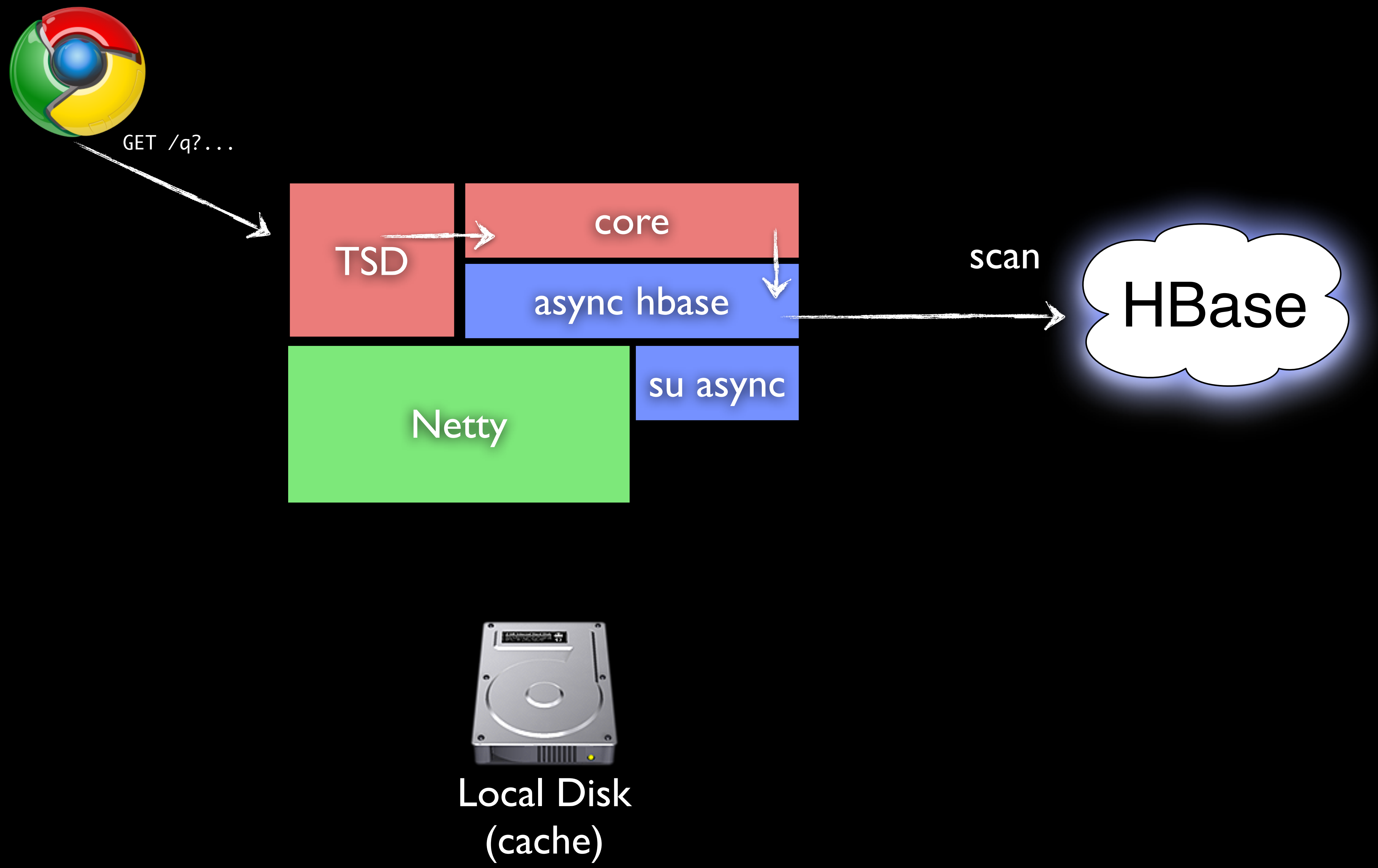


Local Disk
(cache)

Write Path

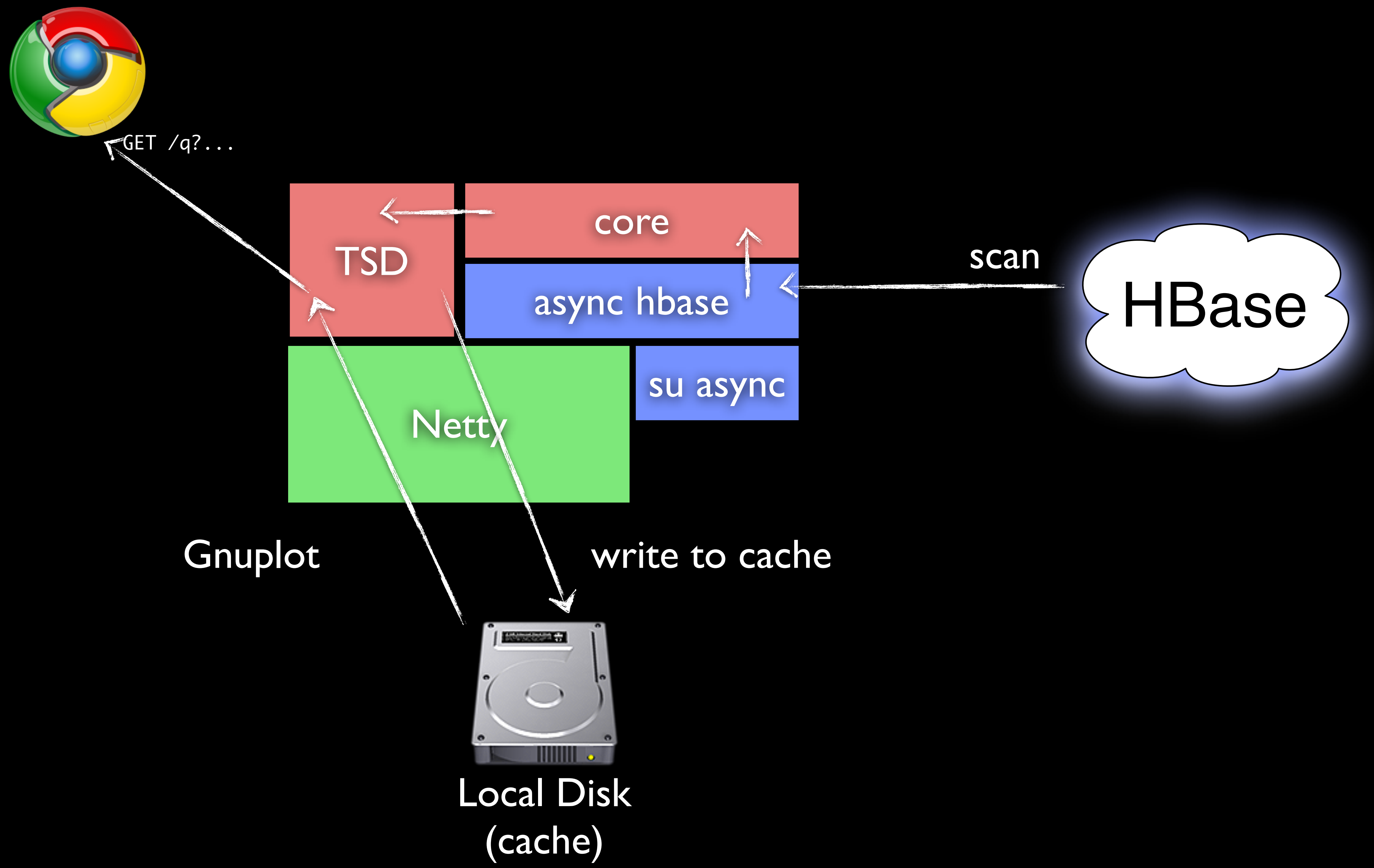
>2000 data points / sec / core

Under the Hood



Read Path

Under the Hood



Read Path