Graphite
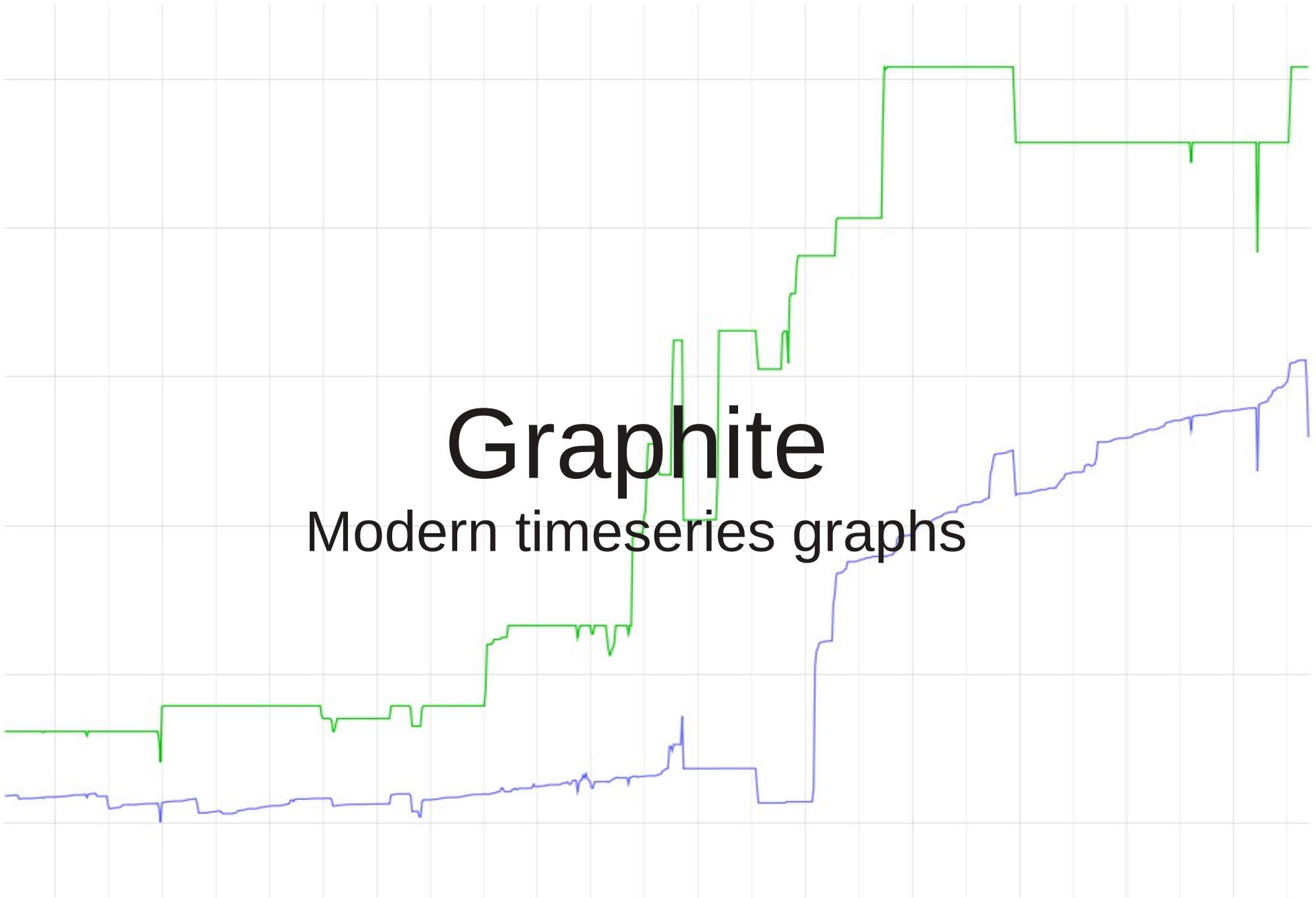
Modern timeseries graphs

Graphite basics

- Graphite is a web based graphing program for time series data.
    - http://www.graphite.wikidot.com
- Written in Python
- Simple input format
    - some.metric.name value unix_timestamp\n
- Consists of multiple separate daemons
- Has it's own storage backend
    - Like RRD, but with more features

The church of graphs
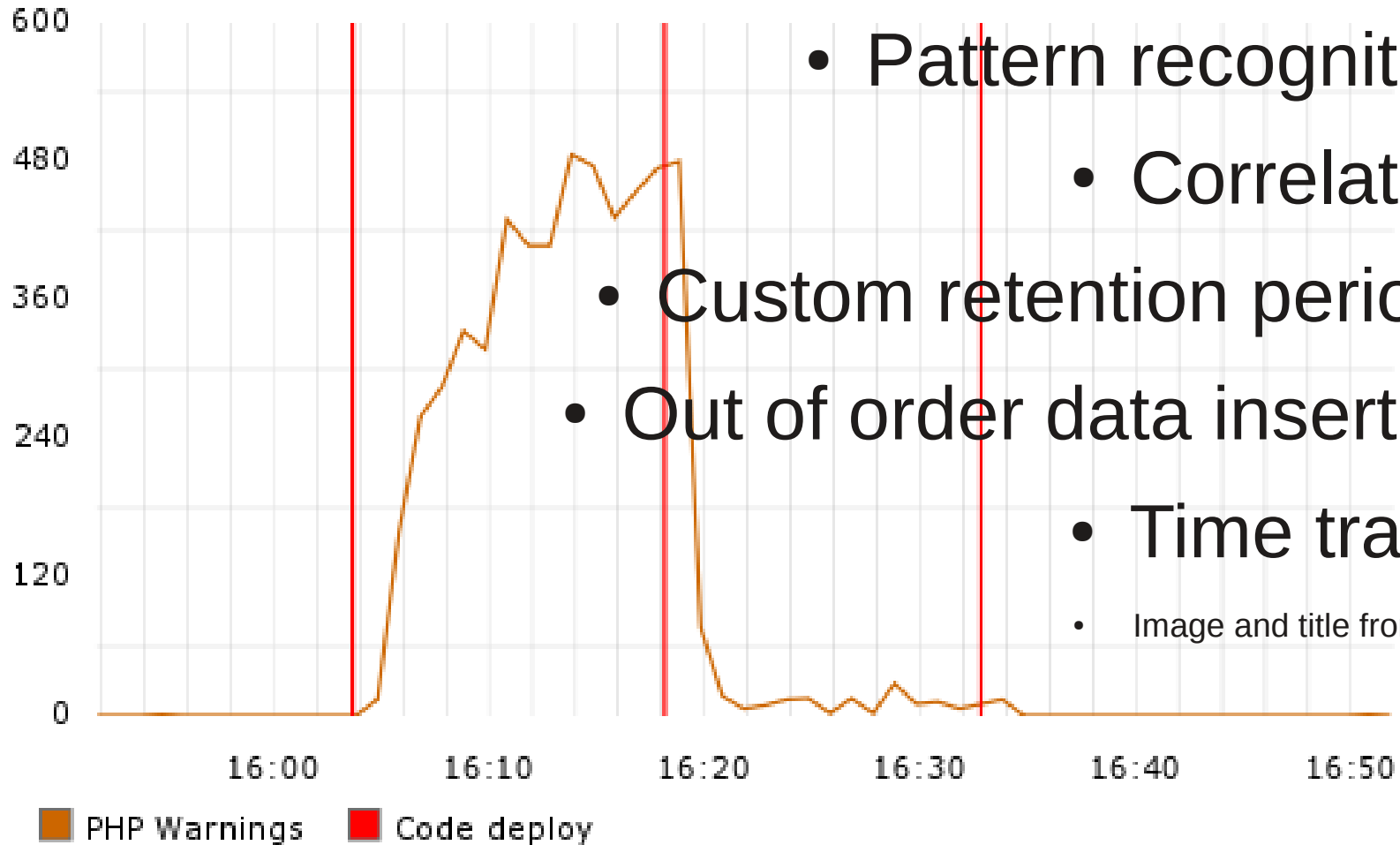
- Pattern recognition
- Correlation
- Custom retention periods
- Out of order data insertion
- Time travel
  - Image and title from Etsy

Moving parts

- Whisper/Ceres
    - The storage backend
- Webapp
    - Web frontend, and API provider
- Relaying daemons
    - Event based daemons
    - Matches input based on name
    - Relays to one or more destinations based on rules or hashing

Original production setup

- A small cluster
    - We were planning to grow slowly
- RAID 1+0 spinning disk setup
    - It works for our databases
- Ran into the IO wall
    - Spinning rust sucks at IO
    - Whisper updates force crazy seek patterns

Scaling problems

- We started with hosts in a /24 feeding one box.

- We ran into IO issues when we added the second /24.

    – On the second day

Sharding

- Added more backends

- Manual rules to split traffic coming to the Graphite setup to storage nodes

- This becomes hard to maintain and balance

Speeding up IO

- Move to 400 GB SSDs from HP in RAID 1.

- We got performance

  - Not as much as we needed

- Losing a SSD meant the host crashed

  - Negating the whole RAID 1 setup

- SSDs aren't as reliable as spinning rust in high update scenarios

Naming conventions

- None in the beginning

- We adopted

  - sys.* for system metrics

  - user.* for user testing metrics

  - Anything else that made sense

Metrics collectors

- Collectd ran into memory problems
    - Used too much RAM
- Switch to Diamond
    - Python application
    - Base framework + metric collection scripts
    - Added custom patches for internal metrics

Relaying

- We started with relays only on the cluster
  - Relaying was done based on regex matching
- Ran into CPU bottlenecks as we added nodes
  - Spun up relay nodes in each datacenter
- Did not account for organisational growth
  - CPU was still a bottleneck
- Ran multiple relays on each host
  - Haproxy used as a load balancer
  - Pacemaker used for cluster failover
- Rewrite in C
  - http://github.com/grobian/carbon-c-relay

statsd

- We added statsd early on

- We didn't use it for quite some time

    - Found that our PCI vulnerability scanner reliably crashed it

    - Patched it to handle errors, log and throw away bad input

- The first major use was for throttling external provider input

- We use this only for metrics from a couple of applications.

Business metrics

- Turns out, our developers like Graphite

- They didn't understand RRD/Whisper semantics though

  – Treat graphite queries as if they were SQL

- Create a very large number of named metrics

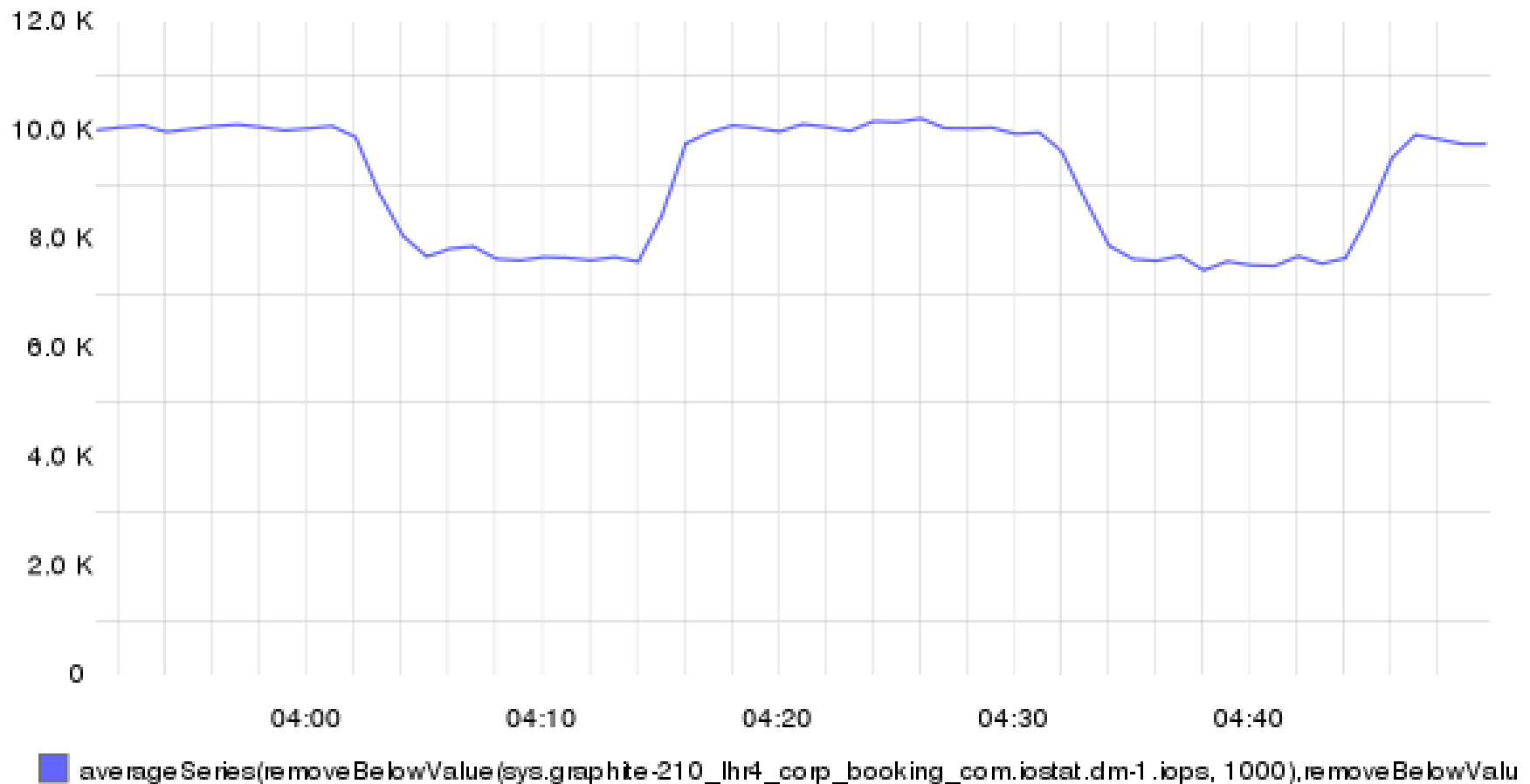 – Not much data in each metric, but the request was for 5.3TiB of space

Sharding – take 2

- Manually maintaining regexes became painful
  - Two datacenters
  - 10 backend servers
- Keeping disk usage balanced was even harder
  - We didn't know who would create metrics and when (this is a feature, not a bug)

Sharding – take 2

- Introduce hashing

- Switch from RAID 1 to RAID 0

- Store data in two locations in a ring

- Mirror rings between datacenters

- Move metrics around so we don't lose data

- Ugly shell scripts to synchronise data between datacenters.

    - http://github.com/jssjr/carbonate does the same things, but is already out there.

Current status (Disk IOPS)

averageSeries(removeBelowValue(sys.graphite-210_lhr4_corp_booking_com.iostat.dm-1.iops, 1000),removeBelowValu

Using Graphite

- Graphs
  - Time series data (default graphs)
  - Sparklines (via API)
- Dashboards
  - Developers create their own
  - Overhead displays
- Additional charting libraries
  - D3.js, Rickshaw
- Nagios
  - Trend based alerting
  - Passive checks

Current problems

- Hardware
  - CPU usage
    - Too easy to saturate
  - Disk IO
    - We saturate disks
    - Reading can get a bit … slow
  - Disks
    - SSDs die under update load
    - A disk lasts between 12 to 18 months.

More interesting problems

- Software
  - The frontend melts down at a few thousand hosts in sys.*
  - We have had problems recording data after upgrading whisper
- Horizontal scalability
  - Adding shards is hard
  - Replacing SSDs is getting a bit expensive
- People
  - Want a graph, throw the data at Graphite
  - Even if it isn't time series data or one record a day

Things we are looking at

- Second order rate of change alerting
  - Not just the trend, the rate at which it changes
- Hbase/Cassandra/RIAK for storage
- Anomaly detection
  - Skyline, etc
- Tracking even more business metrics
- Hiring people to work on such fun problems
  - Developers, Sysadmins ...
  - http://www.booking.com/jobs