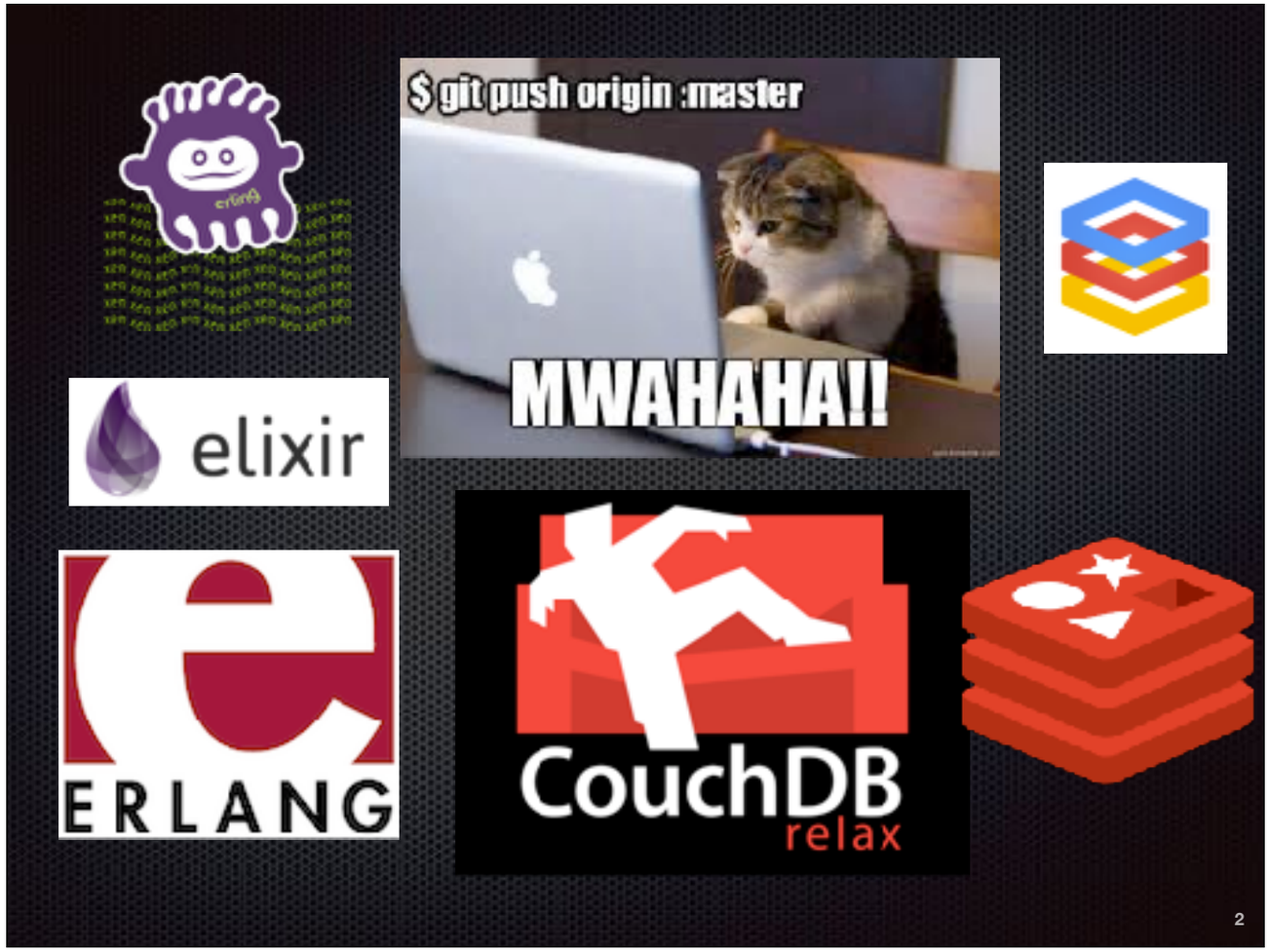
```
{
    "_id": "dch",
    "_rev": "3-814a704a7e0c8d77c4fcbdde8b7bb744",
    "founder": [
        "http://jsonified.com/",
        "http://thecouchfirm.com/"
    ],
    "from": "New Zealand",
    "mails": [
        "dch@apache.org",
        "dch@jsonified.com"
    ],
    "name": "Dave Cottlehuber",
    "passions": [
        "Telemark Skiing",
        "CouchDB Fanatic"
    ],
    "twitter": "@dch__"
}
```


Then:
- Enterprise ICT Outsourcing
- ex Storage & Backup Services Manager for HP Managed Services EMEA, 7 PB + 20k+ servers. In 2006
- Mid-level Manager
- Design, Build & Roll out Multi-year Business & Technology Transformation programmes
- Certified ITIL Master

Now:
- Open source convert
- CouchDB Fanatic
- Committer on CouchDB, CollectD projects
- ASF Member
- Erlang Addict
- Less Meetings
- Optimise for Fun

Always:
- Telemark Skiier
- Leading Edge Tech



This is my preferred development stack.
It has no front end, because I'm a poor designer.
I like functional languages because OO is too hard to understand and reason about.
I like GCE and Bare Metal Erlang but struggle to find time to use them fully.

Hipster Polyglot Monitoring

- OS: CollectD written in C
- Correlation: Riemann in Clojure
- History: Graphite (duh) Python
- Dashboards: nothing perfect yet
- Alerts: PD or AWS SES/SNS
- May contain traces of Erlang/ OTP and CouchDB

HPM is the solution I've been working on this last year for a couple of customers.

The main point today is that by using composable modules from a variety of tools, and some coding you can get exactly what you need for your stack, for a lot less effort, and faster turn-around, than a typical enterprise solution.

There are pain points; no solution is perfect of course, but real-time introspection is hard to pass up.

What's the Problem?

- Correlate events and transactions on disparate bits of infrastructure
- Even from the browser
- And in secure* networks
- In real time & dynamic
- Can we consolidate our infrastructure?
- When do we need to scale?

The problem today is that we need more flexible tools than what we can buy.
And more flexibility than what typical enterprise tools offer.

Could you correlate time spent inside your app stack handling DB calls, with the delay as seen by a specific user?
And in real time?
With metrics from *inside* the DB engine itself?

What if the questions were different tomorrow?
How can you handle that?

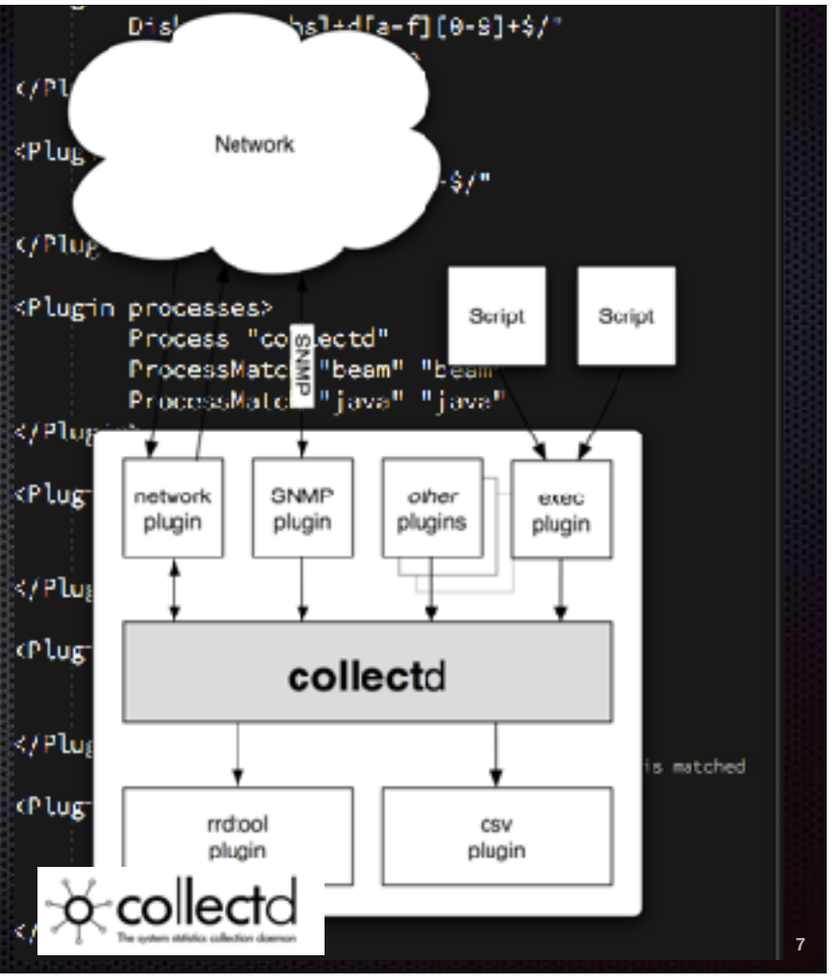
This is one of my customers, running about 15000 instances of redis and couchdb, almost everywhere & anywhere you can think of.

An ideal test case!

CollectD

- daemon written in C
- lightweight & fast
- Multicast support!
- metrics:
  - network stack
  - disk I/O
  - custom thresholds
  - cpu, load
  - virtual, swap mem ...
- misc e.g. JSON or XML via cURL
- plugins like redis, custom UNIX sockets, stdio available

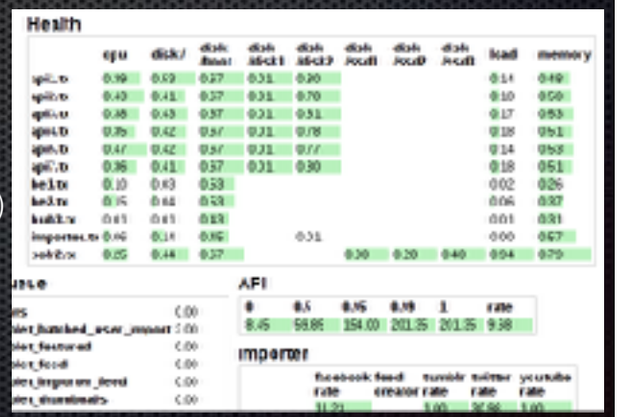
collectd.org



I'm a new committer on CollectD project, I haven't broken anything yet**.
This is my go-to tool for OS monitoring, running as a dropped privileges client per box.
There are many different ways of collecting this data, alternatives such as ganglia, brightcove's diamond stack, it really doesn't matter much what you pick.
Some interesting features that I've not used are unique support for multicast monitoring - ideal for cloud-like deployments where the monitored agent does not know of the existence of the collation agent.

Riemann

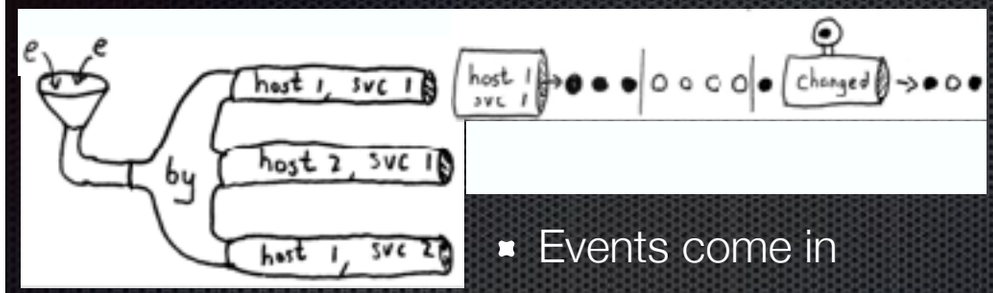
- written in Clojure, ergo Turing Compleat
- Runs on JVM, CPU bound
- Correlate all the things!
  - disparate sources (node, collectd, erlang)
  - diverse functions (filter, rollup, aggregate ..)
  - extremely lisp-y
- Speaks GPB, also HTTP & ws, JSON
- Embarassingly Extensible



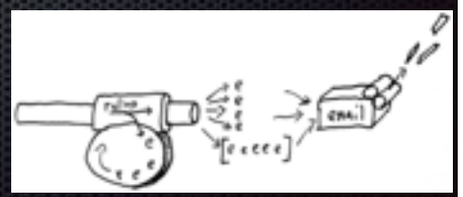
riemann.io

Riemann is an amazing event correlation engine,

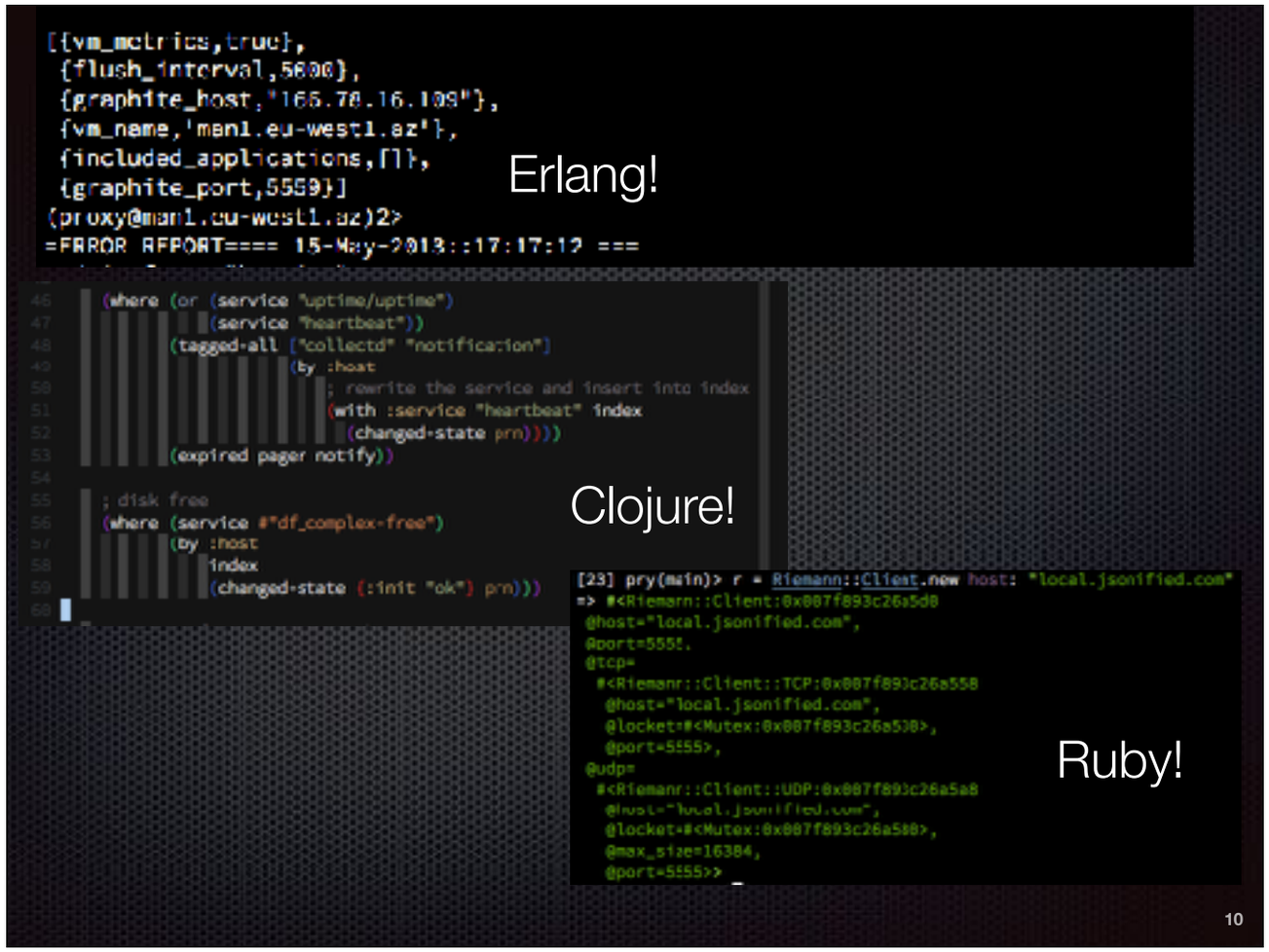
Riemann the Plumber



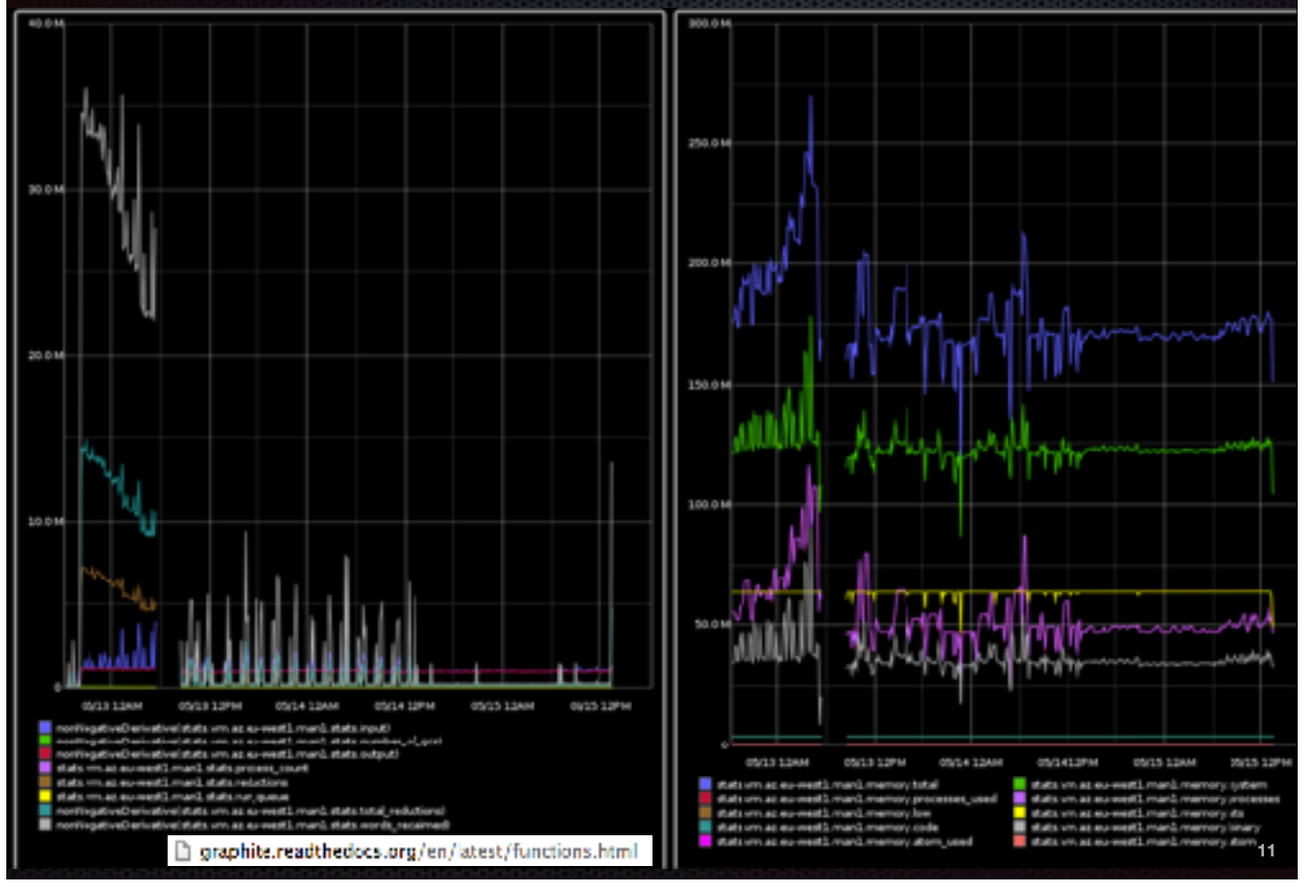
- Events come in
- We group them as we need to
- Watch for changed state
- And do Things



riemann.io/concepts.html


[{vm_metrics,true},
{flush_interval,5000},
{graphite_host,"165.70.16.109"},
{vm_name,'man1.eu-west1.az'},
{included_applications,[]},
{graphite_port,5559}]
(proxy@man1.eu-west1.az)2>
=ERROR REPORT==== 15-May-2013::17:17:12 ===
Erlang!
(where (or (service "uptime/uptime")
(service "heartbeat"))
(tagged-all ["collectd" "notification"]
(by :host
; rewrite the service and insert into index
(with :service "heartbeat" index
(changed-state prn))))
(expired pager notify))
; disk free
(where (service #"df_complex-free")
(by :host
index
(changed-state {:init "ok"} prn)))
Clojure!
[23] pry(main)> r = Riemann::Client.new host: "local.jsonified.com"
=> #<Riemann::Client:0x007f893c26a5d0
@host="local.jsonified.com",
@port=5555,
@tcp=
#<Riemann::Client::TCP:0x007f893c26a558
@host="local.jsonified.com",
@locket=#<Mutex:0x007f893c26a530>,
@port=5555>,
@udp=
#<Riemann::Client::UDP:0x007f893c26a5a8
@host="local.jsonified.com",
@locket=#<Mutex:0x007f893c26a580>,
@max_size=16384,
@port=5555>>
Ruby!

Graphite / Python



Graphite is a Time-Series database, with a round-robin storage engine and support for aggregation of values over time. It fits well with the pipeline of events streamed and cleaned through riemann and stored in graphite for later.

http://graphite.readthedocs.org/en/latest/functions.html has an incredible array of transformation functions, many duplicated & all are possible to do prior in riemann.

You can see here where the ephemeral ports issue was fixed, this graph was whipped up live during diagnosing the underlying issue.

Problems?

- Master, It's How I Roll. Mostly.
- No SmartOS support
- Bleeding edge
- Dependency Hell
- Docs??
- Java tuning & GC pauses
- Valgrind needed
- Now suprisingly stable in Prod

Credits

- You got me ribbons! http://i.imgur.com/S71Mo.jpg
- Riemann.io
- CollectD.org
- Yoda cat http://www.dodger.uselessopinions.com/stuff/images/cat_yoda.jpg
- Hamster caught eating apple http://www.media.desicolours.com/2009/july/cutehamsters15.jpg
- Ralph's balloon http://memerial.net/5014-balloon-wasnt-going-to-come-back
- Cat typing http://images.wikia.com/uncyclopedia/images/f/fa/Cat_using_computer.jpg
- Java http://www.cioal.com/wp-content/uploads/java2.png
- Piggies http://mediad.publicbroadcasting.net/p/ipr/files/201304/piggies.jpg