RYOT YAHOO! SPORTS HUFFPOSTI MAKERS Aol. TechCrunch engadget YAHOO! MAIL 1 by Aol. YAHOO! tumblr. verizon digital media services YAHOO! FINANCE BrightRoll YAHOO! GEMINI

Panoptes: Network Telemetry Ecosystem

Varun Varma, Sr. Principal Engineer

March 10, 2019

verizon

Collect, store, analyze & visualize network telemetry

verizon

10 second primer on Network Telemetry

verizon

Network Telemetry

- Collection of metrics and state from network devices
- The dominant protocol to collect telemetry is SNMP (Simple Network Management Protocol)
  - Which is unencrypted transmission over UDP
  - First defined in 1993
- APIs, Agents and Streaming Telemetry are becoming mainstream

How is this problem different?

Why not use ‘x’?

- High rate of change network
  - Static configuration is out of the question
- Primitives unique to network telemetry
  - E.g. rate conversion, enrichments
- Decoupling of collection, processing, and storage
- Python

Complexifiers

- We have to poll as pushing metrics from devices isn't supported universally
  - Polling is expensive on devices
- Vendor/Platform/OS Diversity
- Scale

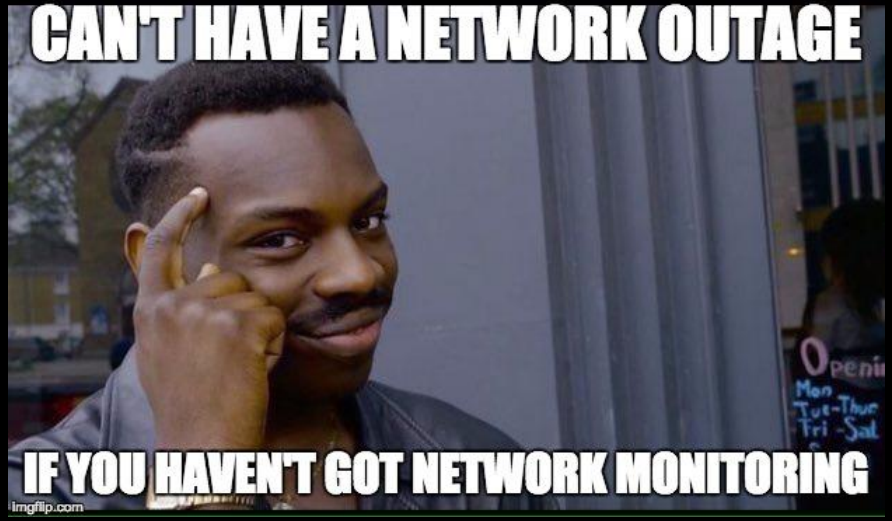
verizon

Meet Panoptes

verizon

Panoptes

- Greenfield Python based network telemetry platform
- Built @Yahoo, now Verizon Media
- Provides real time telemetry collection and analytics
- Implements discovery, enrichment, polling, distribution bus and numerous consumers



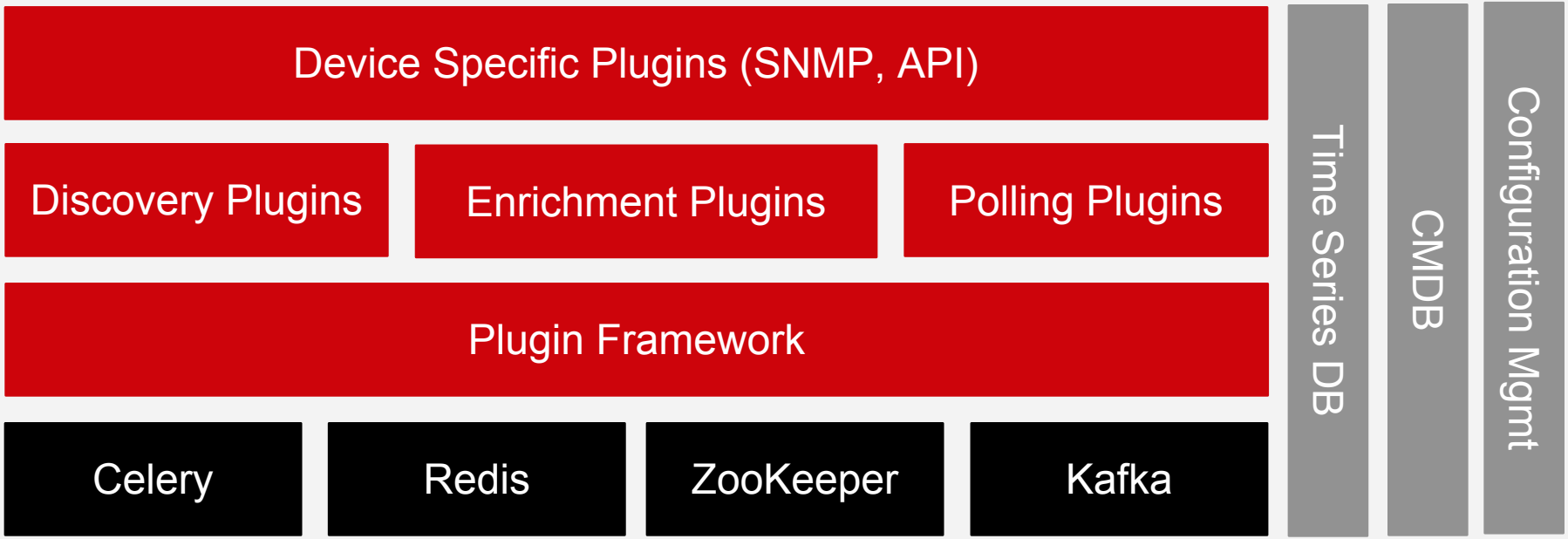
verizon

Architecture

System Requirements

- Multiple methods to collect data
  - SNMP, APIs, CLI, Streaming
- Horizontal Scalability
  - No Single Point Of Failure
- Multiple, extensible, ways to consume data
- Survive Network Partitions

Platform



verizon

Framework Requirements

- Configuration Parsing
- Logging Management
- Plugin Management
- Work Queue Management
- Message Bus
- Distributed Locking and Leader Election
- Persistence
- Caching
- Federation

Tech Stack

Framework Requirement	Choice
Language	Python
Configuration Parsing	ConfigObj
Logging	Logging Facility + rsyslog
Plugin Management	yapsy
Work Queue Management	Celery
Message Bus	kafka-python + Kafka
Distributed Locking, Leader Election	Kazoo + ZooKeeper
Persistence	OpenTSDB, Django + MySQL
Caching	redis-py + Redis
Federation	Django + MySQL

Core Concepts

Plugins

- Python classes conforming to a well defined API
- Can collect/process and transform data from any source
  - SNMP
  - API
  - CLI
  - *
- Can be of three types:
  - Discovery
  - Enrichment
  - Metrics

Resources

- Abstract representations of what should be monitored
  - In the context of network telemetry, these would usually be the network devices to monitor
- ‘Discovered’ using discovery plugins
  - Usually would talk to a Configuration Management Database but could also be from topology walks
- Have an id, endpoint and various metadata
  - For example, the vendor name or operating system version of a device would be it’s metadata
- Specified within Panoptes with a DSL
  - Example: “resource_class” = “network” AND “resource_subclass” = “switch” AND “resource_type” = “cisco” AND “resource_metadata.os_version” LIKE “12.2%”

Metrics

- Numbers that can be measured and plotted
  - Example is the bytes in/bytes out counter of an interface
- Generally fast changing or have the potential to be
- Can be collected through various means:
  - SNMP
  - API
  - CLI
  - Streaming

Enrichments

- Are metadata in addition to metrics
    - For interfaces, we collect metrics like bytes in and bytes out and enrichments like interface name and description
- Can be any data type
    - Unlike metrics which can only be numeric
- Can come from sources other than the device being monitored
    - The geo location of the device or the ASN number to name mapping

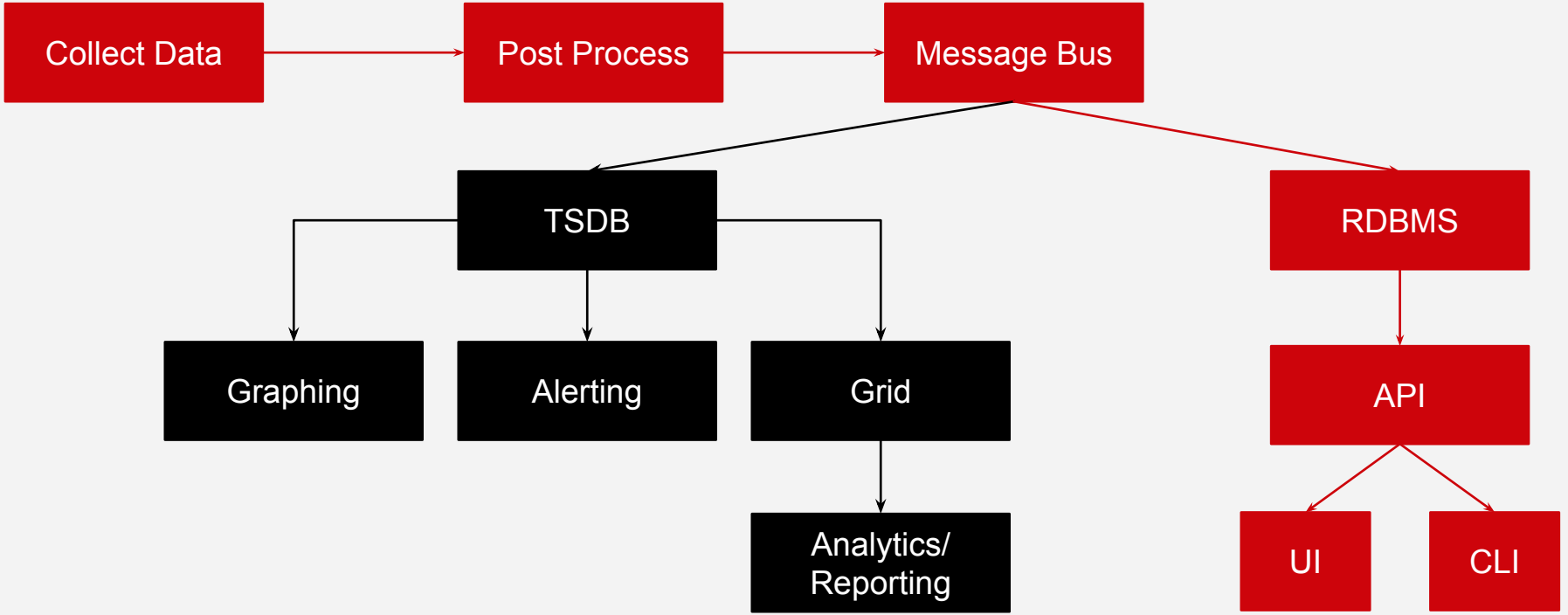
Enrichments Cont...

- Usually are more expensive to process than metrics
  - Might need complex transformations and therefore...
    - Are collected at a rate less than those for metrics
      - We collect interface metrics every 60 seconds, but enrichments every 30 minute
    - Are cached
- Allows us to scale more by being efficient about data collection

Data Encoding & Distribution

- Panoptes is a distributed system
  - Discovery, enrichment and polling are all decoupled
- Kafka and/or Redis are used to pass data between all subsystems
  - This makes it so that you can extend or introspect any subsystem
- JSON is used to encode all data within Panoptes
  - It’s non-performant but developer/operator friendly

Workflow



verizon

Scaling & Operations

Scale: Orders of Magnitude

10M

Time Series

100K

Network Interfaces

10K

Network Devices

100

Network Sites

60

Seconds

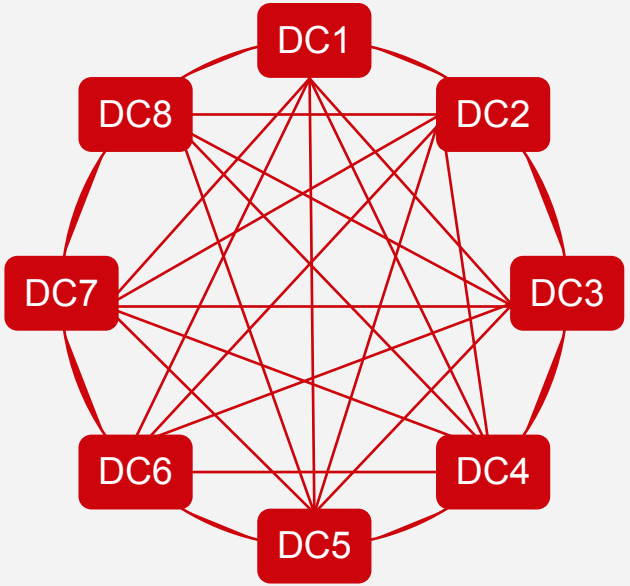
verizon

Scaling Issues

- Panoptes was built to be horizontally scalable and free of single points of failure from day one
  - Performance or high-availability are not easy to bolt on afterwards
- We chose Python to be developer friendly but it wasn't fast enough
  - High throughput actions are delegated to C extension modules
- Ditto for JSON serialization for all data
- We broke everything - Redis, ZooKeeper, Kafka
  - Redis allows 'only' 10,000 clients to be connected by default :)

Divide & Conquer: Federated API

- Due to availability concerns, each site has its own MySQL cluster
  - Telemetry data must be available during a network partition
  - Centralized telemetry store might not be reachable in all cases
- Each API endpoint acts as a tribe node
  - If a tribe node doesn't have the requested data, it returns a pointer to the node that does through a find API



verizon

Covered Systems

- Interface metrics for Arista, Cisco, Juniper, A10, Brocade
- System metrics for A10 (AX, TH), Arista EOS, Brocade TrafficWorks, Cisco IOS, Cisco IOS-XE, Cisco NX-OS, Juniper (MX, SRX)
- Functional metrics for VIPs (A10 AX, TH, Brocade), A10 LSN, Juniper SRX

Operational Experiences

- Metrics across different platforms or versions of even the same OS from vendors aren’t consistent
  - Normalizing these metrics was our single biggest time drain
- SNMP has its faults but is still ubiquitous
  - Especially in a multi-vendor, multi-platform, and multi-generational network
- Performance of APIs was much better than SNMP
- Using Kafka proved to be the right choice, we already have 3 separate consumers

Operational Experiences Cont...

- We don’t expose ‘raw’ data to external systems
  - It’s tempting to give access to external teams via Kafka, but that would lead to friction if we want to change our internals
  - Instead, we expose APIs which abstract away all our internals
- We push metrics to our in-house time series database and alerting service
  - Custom dashboard service our user base is familiar with
  - Economies of scale – no need to provision new hardware or software
- Custom UIs are useful and enabled by APIs

Performance

verizon

Throughput = Speed x Parallelism

verizon

Throughput = Speed x Parallelism x Productivity

verizon

“Optimize for your most expensive resource”

- Nick Humrich: Yes, Python is Slow, and I Don’t Care

Scaling Vertically: aka Speed

Profile it!

Our single slowest operation? JSON Schema Validation

Begin with the basics

https://wiki.python.org/moin/PythonSpeed

https://wiki.python.org/moin/PythonSpeed/PerformanceTips

- List comprehensions
- Built-ins
- Local vs. global

Tools

- cProfile
  - Built-in since Python 2.5
  - pstats lets you do slicing/dicing/reporting
  - Use with a signal handler to profile daemon processes
- objgraph
  - Hunt down memory leaks
  - Draw graphs of object counts and relations

cProfile

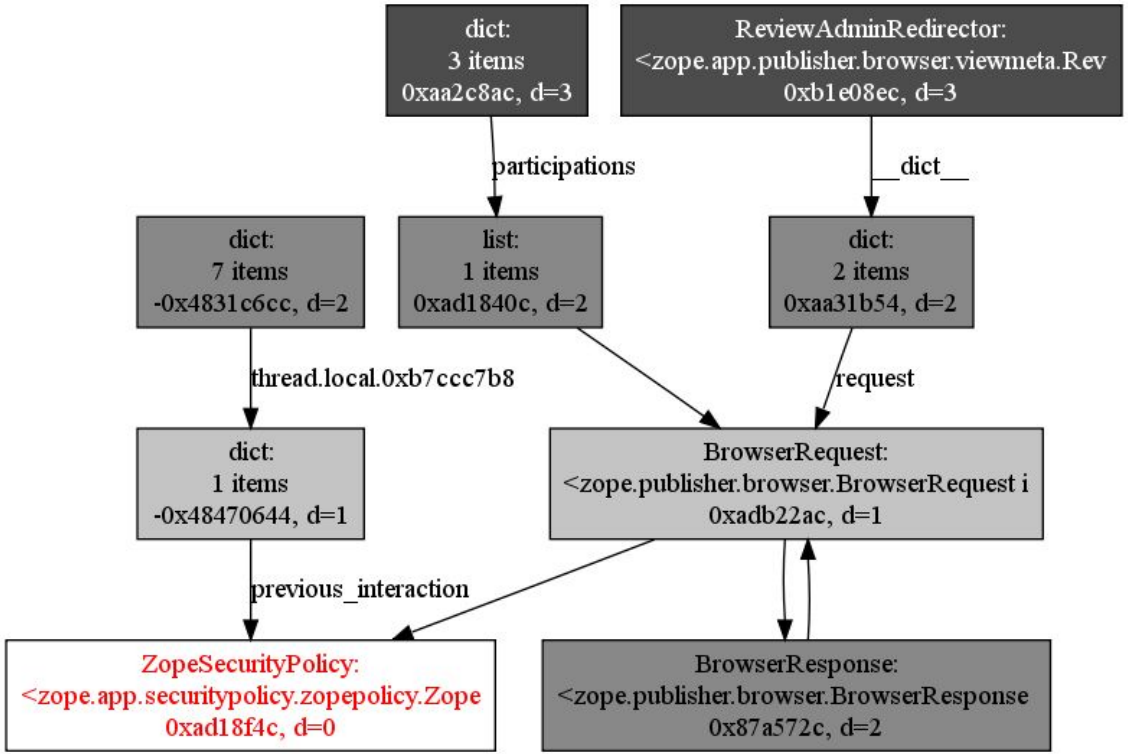
import cProfile
import re
cProfile.run('re.compile("foo|bar")', 'restats')

197 function calls (192 primitive calls) in 0.002 seconds

Ordered by: standard name

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.001	0.001	<string>:1(<module>)
1	0.000	0.000	0.001	0.001	re.py:212(compile)
1	0.000	0.000	0.001	0.001	re.py:268(_compile)
1	0.000	0.000	0.000	0.000	sre_compile.py:172(_compile_charset)
1	0.000	0.000	0.000	0.000	sre_compile.py:201(_optimize_charset)
4	0.000	0.000	0.000	0.000	sre_compile.py:25(_identityfunction)
3/1	0.000	0.000	0.000	0.000	sre_compile.py:33(_compile)

objgraph



verizon

Use C Extension Modules

cDecimal vs. Decimal (in Python < 3.3):
Pi, 64-bit, 10,000 iterations, 3.16GHz Core 2 Duo

Digits	floats	decimal	cdecimal	cdecimal-nt	gmpy
9	0.12s	17.61s	0.27s	0.24s	0.52s
19	-	42.75s	0.58s	0.55s	0.52s
38	-	-	1.32s	1.21s	1.07s
100	-	-	4.52s	4.08s	3.57s

Source: http://www.bytereef.org/mpdecimal/benchmarks.html

Cache Properties

https://github.com/pydanny/cached-property

Scaling Horizontally: aka Parallelism

Celery!

Scale across processes, CPUs, and hosts

http://www.celeryproject.org/
How Celery fixed Python's GIL problem

Choose & test dependent systems that scale horizontally

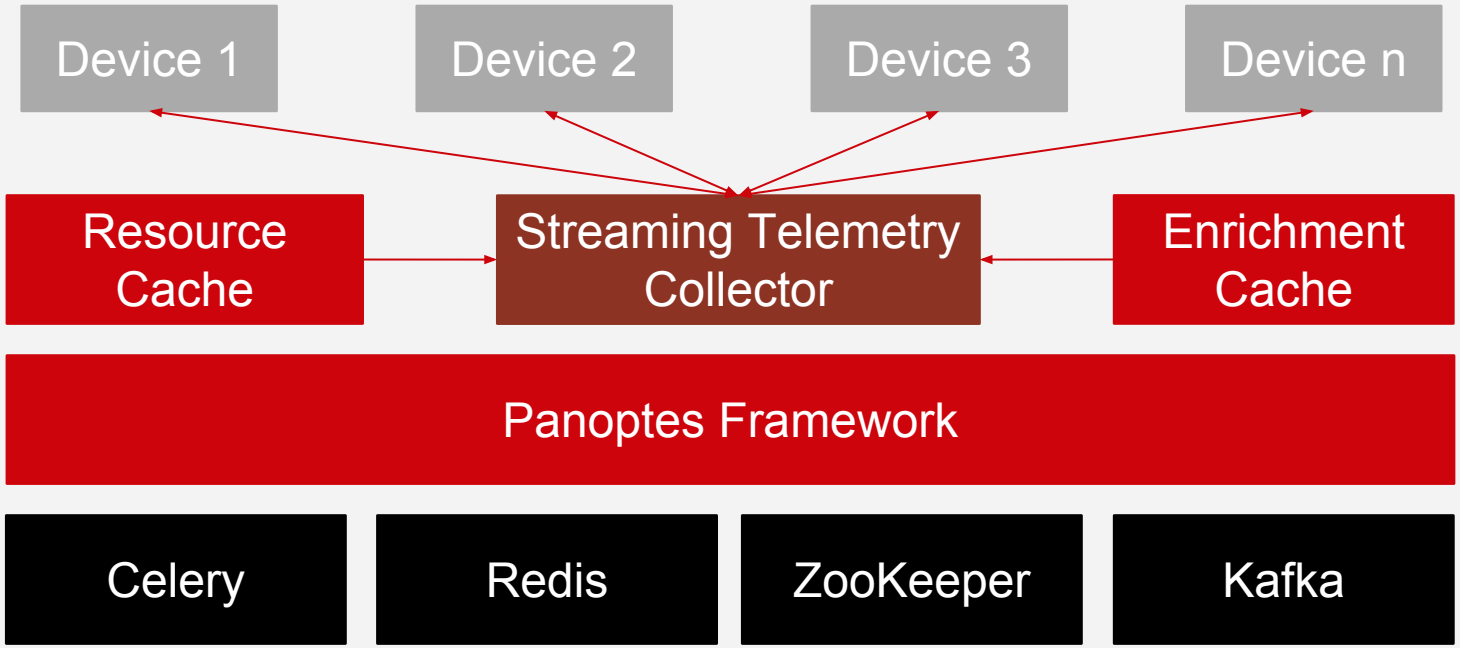
Compare system performance with all features

TBD

cython, Async I/O, More C extension modules

Future: Streaming Telemetry

Proposed Design



verizon

Pretty Pictures

APIs

Realtime - purpose specific

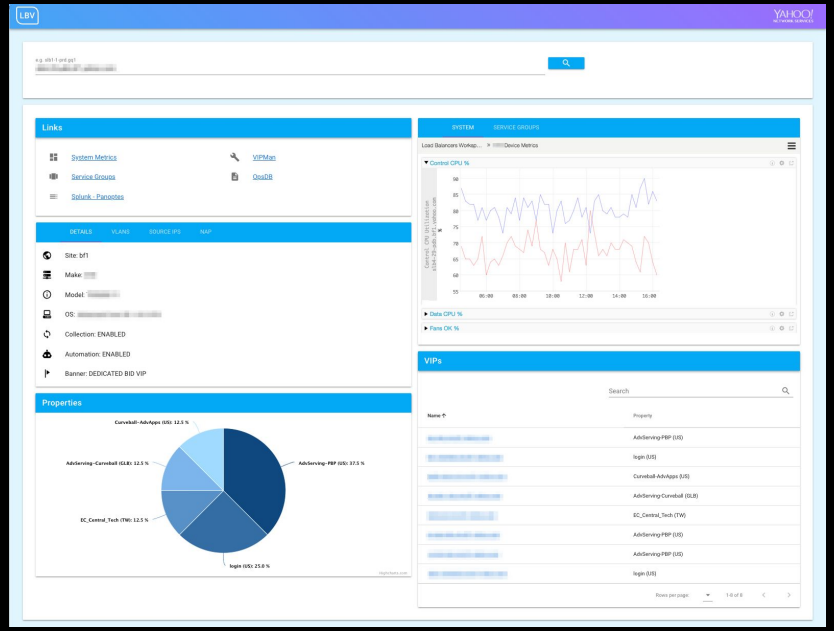

```
{
  "members_metrics": [
    {
      "load_balancer_model": "[redacted]",
      "weight": 1,
      "site": "[redacted]",
      "vip": "[redacted]",
      "load_balancer_make": "[redacted]",
      "vip_property": "[redacted]",
      "max_connections": 100000,
      "bytes_in_gauge": 802742,
      "bytes_out_gauge": 0,
      "load_balancer_name": "[redacted]",
      "polling_interval": 60,
      "active_connections_gauge": 24307,
      "vip_port": 443,
      "status": 0,
      "pool_name": "[redacted]",
      "packets_out_gauge": 0,
      "timestamp": 1496772838,
      "real_port": 443,
      "vip_type": "l3dsr",
      "packets_in_gauge": 4221,
      "cache_age": 41,
      "ip_address": "[redacted]",
      "name": "[redacted]",
      "connections_per_second_gauge": 281,
      "total_connections_counter": 746440138,
```

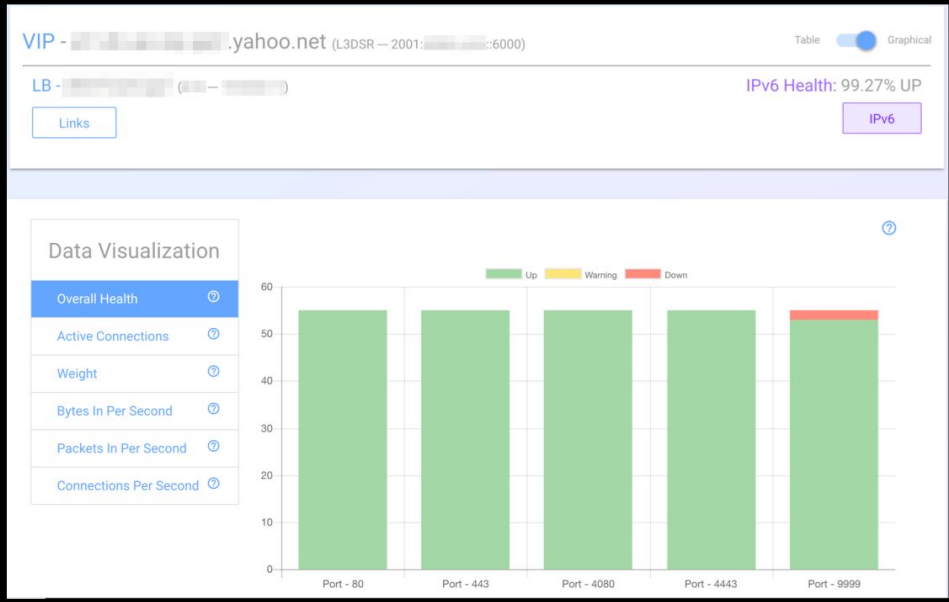

Bulk/Historical - Generic


```
[
  - {
      - aggregateTags: [
            "_aggregate",
            "resource_endpoint",
            "resource_site",
            "vip_type",
            "real_port",
            "vip_property"
        ],
      - dps: {
            1525809840: 100000
        },
        metric: "${Panoptes.network-load-balancer-vip.real_max_connections}",
      - tags: {
            vip_protocol: "tcp",
            vip_ip_address_version: "4",
            vip_port: "9999",
            real_dns_name: "[redacted]",
            vip_dns_name: "[redacted]"
        },
        __groupId__: "real_dns_name:[redacted]|vip_dns_name:[redacted]|vip_ip_address_version:4|vip_port:9999|vip_protocol:tcp"
    },
```


verizon

Custom UIs





verizon

And now: a special offer just for you...

verizon

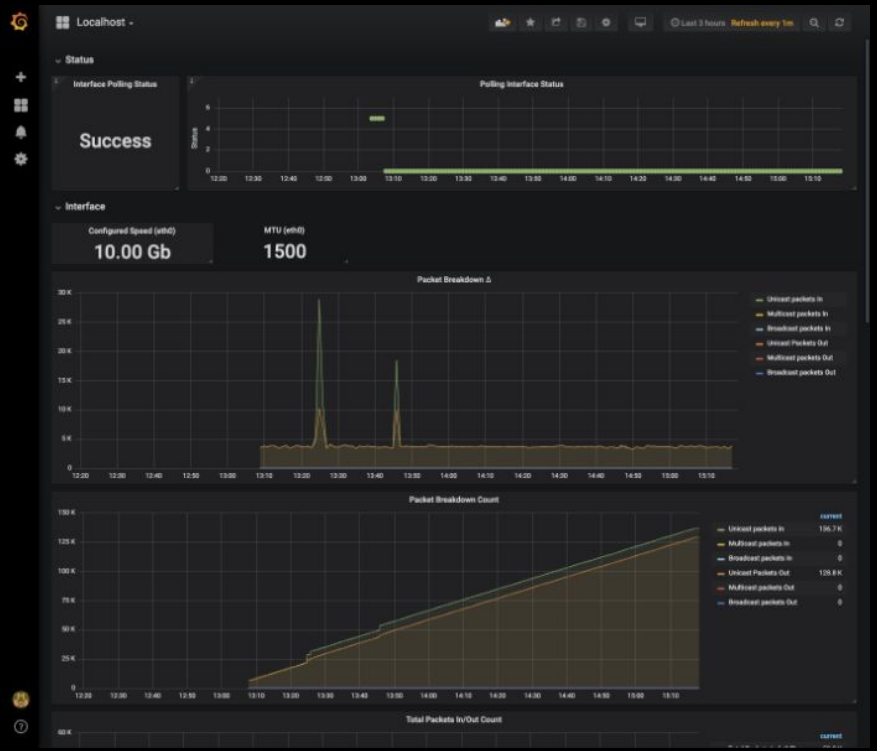
getpanoptes.io

verizon

What you get

- Docker container
- Discovery, enrichment and polling of the interfaces of the host you deploy on
- InfluxDB as the TSDB
- Grafana as the dashboarding system

Sample InfluxDB/Grafana Dashboard



verizon

Why?

- Docker container
- Discovery, enrichment and polling of the interfaces of the host you deploy on
- InfluxDB as the TSDB
- Grafana as the dashboarding system

Feedback & Contributions

- Try it out!
- Find and fix bugs
- Tell your friends, family, and colleagues
- Can be used for more than just network telemetry

Thank you

verizon

Questions?

vvarun@verizonmedia.com

verizon