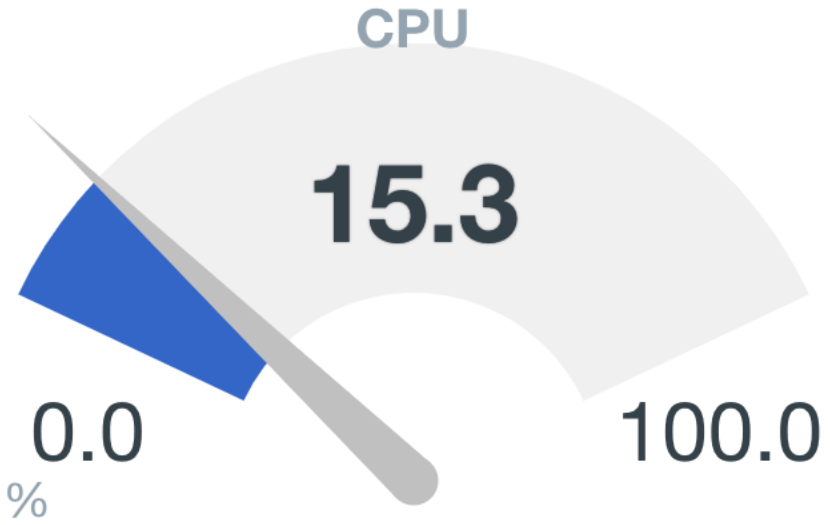
Prometheus: is it ready for Network Monitoring?

Brian Candler

Network Startup Resource Center

UNIVERSITY OF OREGON

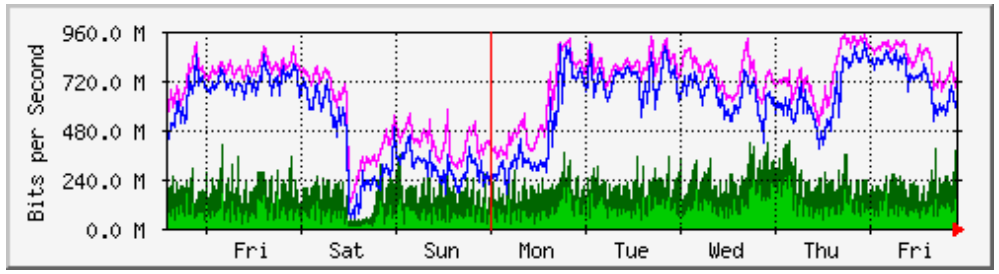
To manage our networks, we need to collect measurements (metrics)

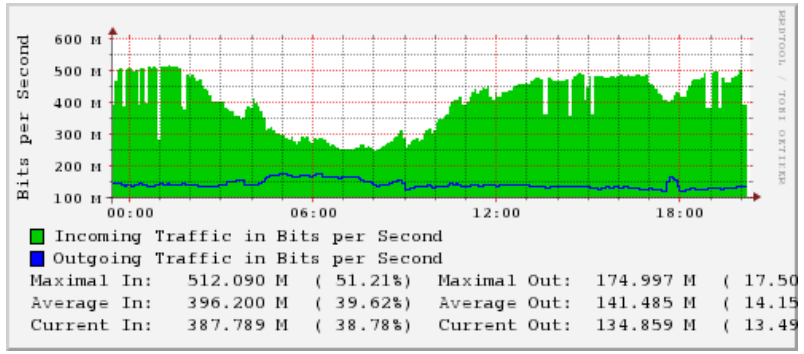


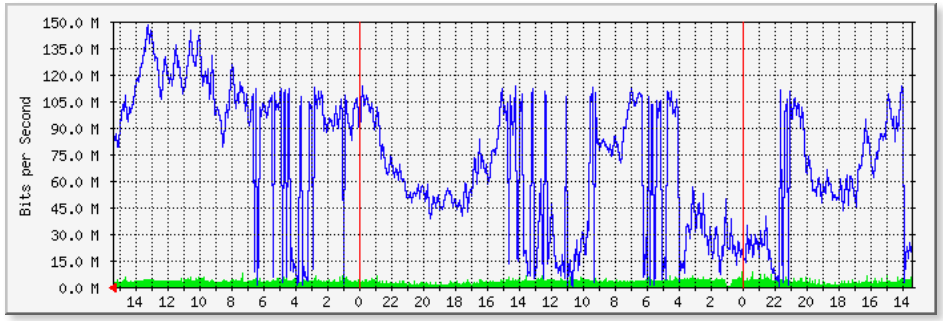
(and view historical data, identify trends, generate alerts…)

Many OSS network monitoring tools use **RRDtool** internally for storing and graphing time series data

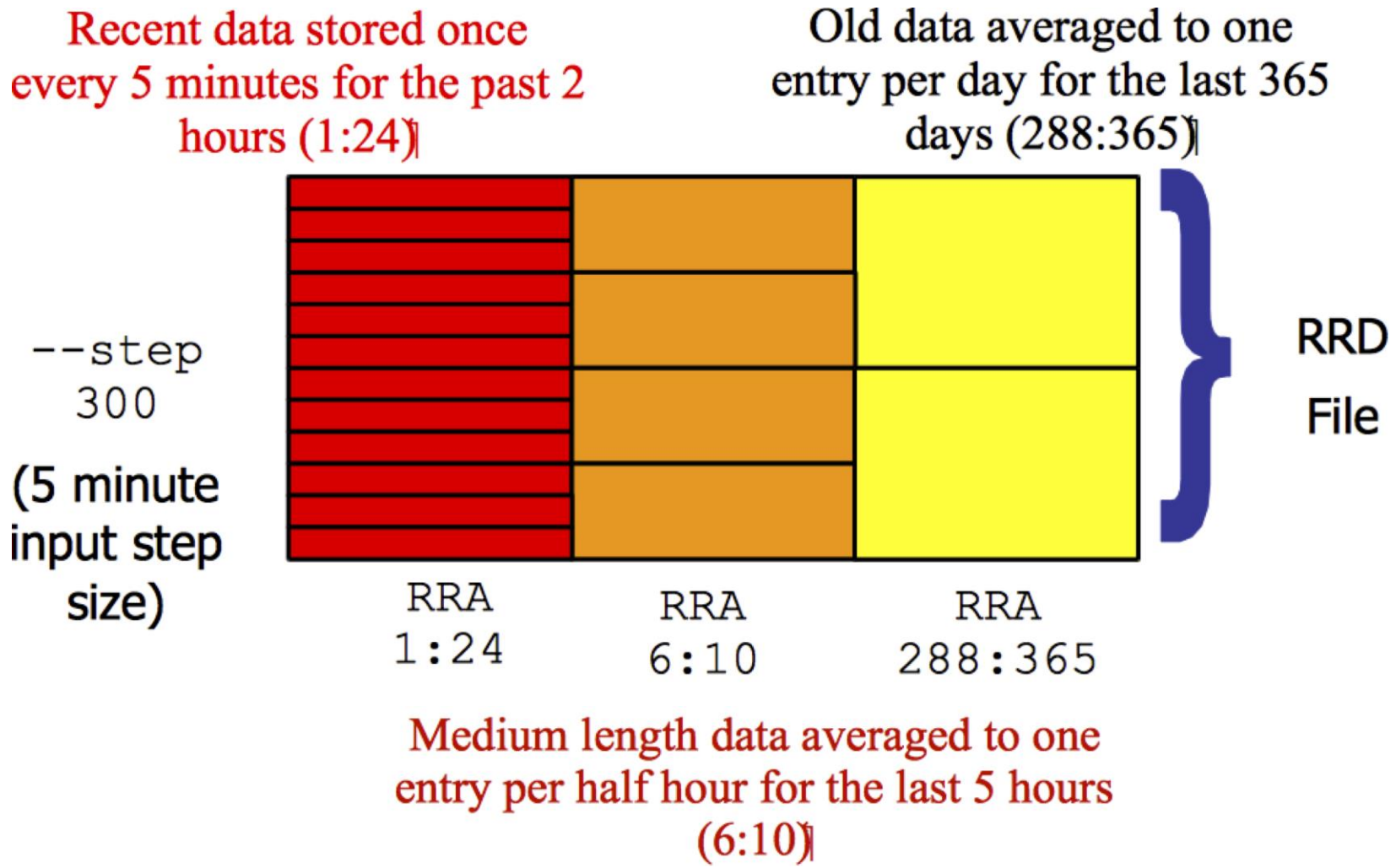
- Cacti
- Smokeping
- LibreNMS
- Check_mk
- NfSen
- Munin
- …







RRDtool database file format (example)



Problems with RRDtool

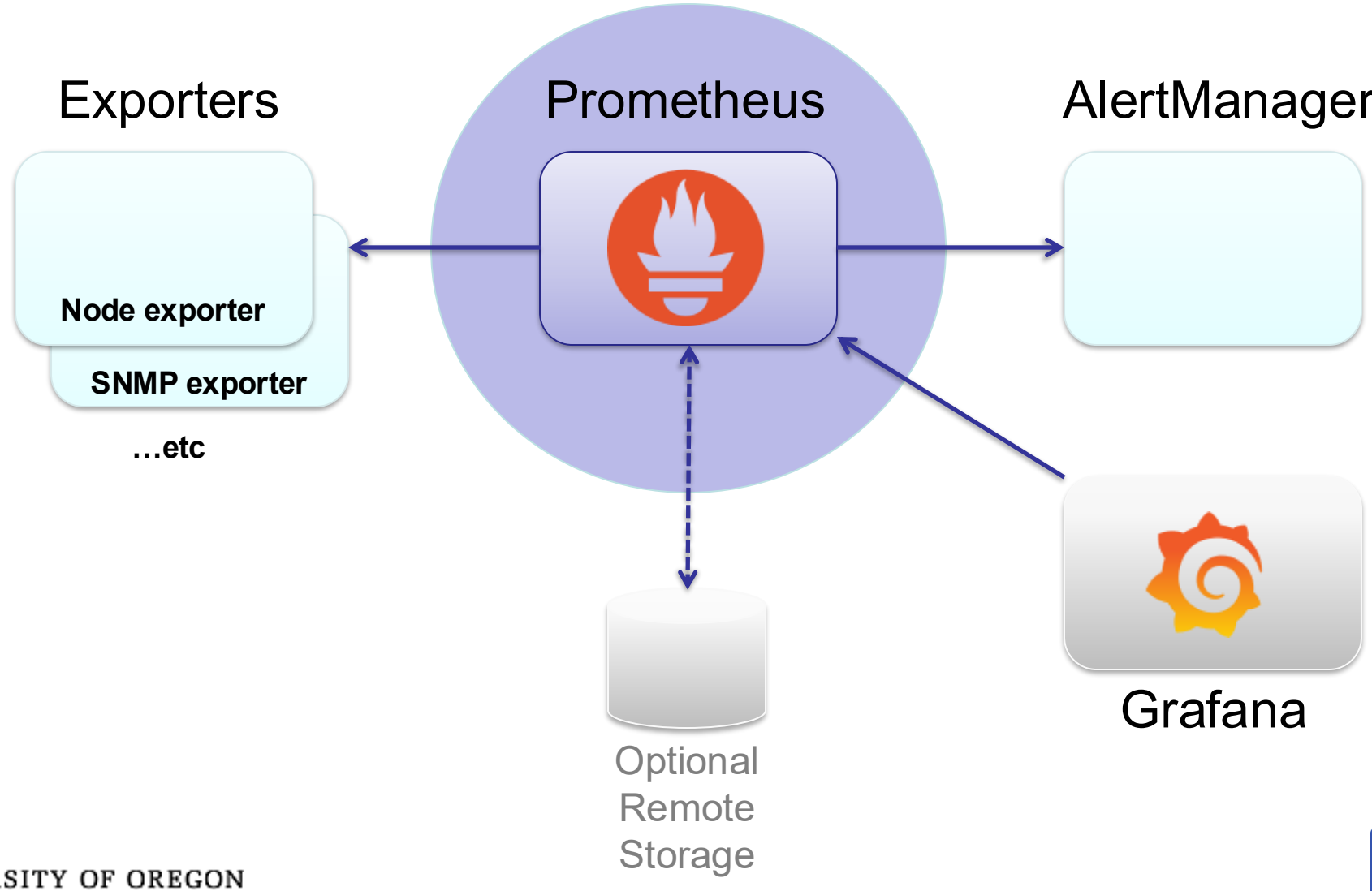
- Released in 1999, designed to minimize **disk space usage**
  - Hard disks are now 1,000 times bigger, but IOPS only slightly better
  - It's now solving the wrong problem
- Reduces resolution (throws data away) to maintain fixed file sizes
- Updates often rewrite the whole file
  - Very inefficient, and very bad for write amplification on SSDs
  - *rrdcached* tries to help
- You need to throw a lot of hardware at the problem
- A new generation of **time series databases** (TSDB) are much better optimized for this use case

InfluxDB, Cassandra, TimescaleDB, VictoriaMetrics, OpenTSDB, Clickhouse, Yottadb…

What's Prometheus?

- Highly efficient time series database
  - Average usage: 1.7 bytes per data point
  - Batched writes and Write-Ahead Log minimize I/O
- Data collection engine (scraper)
- Query language (PromQL)
- Alerting rules engine
- HTTP API and web query frontend
- Metrics exchange format (OpenMetrics)
- Sounds like what we want!

Prometheus ecosystem



Data collection

- Prometheus' background is from devops (think: kubernetes)
- Data collection is by making periodic HTTP requests to "exporters"
  - Active polling, a.k.a. "pull monitoring"
  - Can proxy it, enable TLS and authentication, etc
- HTTP response in *OpenMetrics* format
  - Plain text: easy to read, easy to debug (e.g. with curl)

```
curl -fsS localhost:9100/metrics
…
node_filesystem_avail_bytes{device="/dev/mapper/vg0-root",fstype="ext4",mountpoint="/"} 7.27789568e+09
node_filesystem_avail_bytes{device="/dev/nvme0n1p1",fstype="vfat",mountpoint="/boot/efi"} 1.1186176e+09
node_filesystem_avail_bytes{device="lxcfs",fstype="fuse.lxcfs",mountpoint="/var/lib/incus-lxcfs"} 0
…
```

Example exporters

- **node_exporter** collects *nix server stats - snmpd not needed
  - very easy to add custom metrics (textfile collector)
- **windows_exporter** for Windows servers
- **blackbox_exporter** performs Nagios-like active service checks
- Third-party exporters for Postgres, IPMI, LVM, Proxmox VE, …
- Built-in prometheus exporters in many modern applications
  - e.g. ceph, linstor, incus, netbox, zrepl, powerdns, …

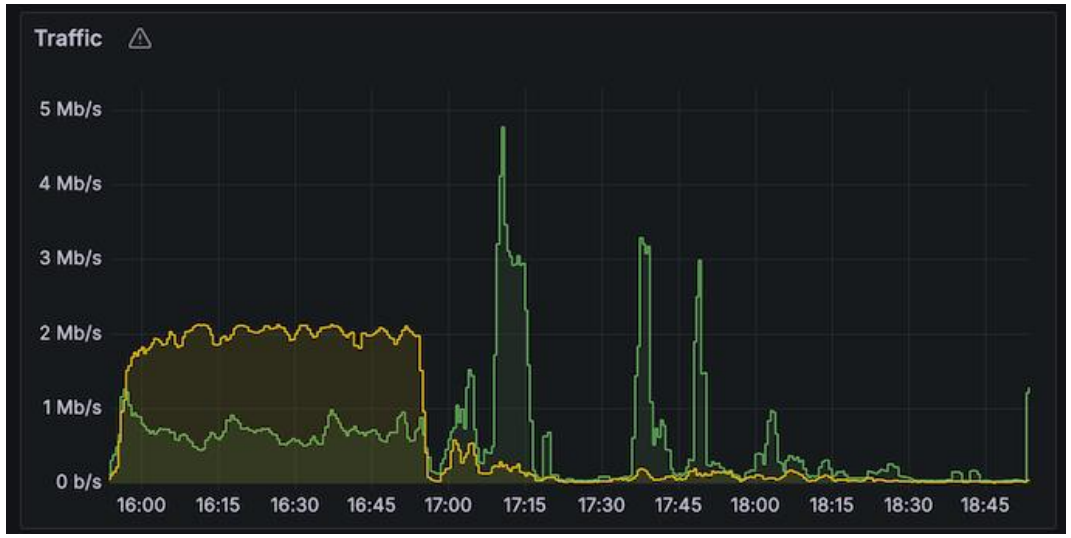
But can you use it with SNMP?

- Yes!
- Pull-mode maps nicely to SNMP polling
- **snmp_exporter** accepts HTTP scrape request, performs SNMP collection, and returns OpenMetrics

```
ifHCInOctets{instance="rtr1",ifName="gi0"} 1.7658225527e+10
ifHCInOctets{instance="rtr1",ifName="gi1"} 3.56567063e+08
ifHCOutOctets{instance="rtr1",ifName="gi0"} 3.56567063e+08
ifHCOutOctets{instance="rtr1",ifName="gi1"} 2.5711868714e+10
```

Does it work?

- Yes! Incredibly well
- In my experience, you can scrape at 1 minute intervals instead of 5 minutes, and still use around **1%** of the resources of LibreNMS! *



* Collecting only IF_MIB. Maybe unfair comparison, LibreNMS was likely collecting more MIBs

What's the catch? Rather a lot, actually.

- It's not a Network Management System, it's a kit of parts!
  - With a steep learning curve
- There's no built-in network inventory or device auto-discovery
  - It can read various types of inventory, from YAML files to cloud APIs; Prometheus calls these modules "service discovery"
  - Inventory needs to specify scrape options, e.g. explicitly which MIB(s) to poll for each SNMP device

What's the catch? (cont'd)

- SNMP collection is tricky to configure at first
  - Need "relabeling rules" to set HTTP parameters for each scrape, e.g.

```
localhost:9116/snmp?target=10.12.255.1&module=if_mib&auth=workshop_v3
```

- To use MIBs other than the supplied ones, you have to compile them into YAML using the "generator", which is also tricky
  - Compile Go code from source; isolate all the MIB interdependencies

What's the catch? (cont'd 2)

- PromQL is not like any other query language you've used
  - A key part of the learning curve
  - It works with "vectors" of values, distinguished by "labels"
  - But it's very powerful, and the same language is used for dashboards, configuring alerts (inc. using history), and ad-hoc queries

PromQL query examples

- All values of a given metric name

```
node_filesystem_avail_bytes
```

- Filter by label values or patterns

```
node_filesystem_avail_bytes{mountpoint="/home"}
node_filesystem_avail_bytes{instance=~"server[1-3]"}
```

- Filter by time series value

```
node_filesystem_avail_bytes < 100000000
```

  - Commonly used for alerting expressions
  - Note that this is not a boolean (true/false) result: it filters out values which don't meet the criteria, and passes those that do
  - If *any* value is present in the result set, an alert is fired

More query examples

- Arithmetic

  ```
  node_filesystem_avail_bytes / 1024
  node_filesystem_avail_bytes / node_filesystem_size_bytes
  ```

- Functions across time series

  ```
  sum(node_filesystem_avail_bytes)
  ```

  Returns a 1-element instant vector with the *total available size*

  ```
  min(node_filesystem_avail_bytes)
  ```

  Returns a 1-element instant vector with the *lowest available size*

- Turning counters into rates

  ```
  rate(ifHCInOctets[5m])
  ```

What's the catch? (cont'd 3)

- In many ways, you're starting from scratch for network monitoring
  - No pre-existing Grafana dashboards for SNMP *
  - No pre-existing alerting rules for common conditions, like "disk full"
- More discussion:
  - https://docs.google.com/document/d/1oEpjiWfTHF352NCAOGolwij3EIkrprCkdQmaQMpjg4M/ *"First-class network monitoring support in the Prometheus & Grafana ecosystem"*

* I published dashboards #12489 and #12492 as examples

Note: Prometheus is for metrics only

- For logs and netflow you'll need something else
  - Elasticsearch/Opensearch – *resource hungry*
  - Grafana Loki
  - VictoriaLogs
  - ClickHouse (+ OTel Collector, Akvorado, ntop-ng …)
  - OpenObserve
  - Quickwit
  - …
- But you *can* generate metrics from logs
  - e.g. counts of errors, histograms of response times

Bottom line: should you deploy this for NM?

- Yes, if…
- you have large numbers of network devices to poll (CPE, IOT); or
- you have non-SNMP metrics to collect too (e.g. application monitoring); or
- you have in-house systems or dev expertise