CERN

An overview on CINNAMON

An update on IPMI monitoring @ CERN IT

Luca Gardi

What is CINNAMON?

- stands for **C**entralized **I**PMI **N**otificatio**N** **A**nd **M**onitoring System
- provides a consistent part of CERN's DC server hardware, temperature and power monitoring
- meant as a replacement to in-band **ipmi-lemon-sensor**
- developed and introduced by *Alberto G. Molero*, presented at ASDF on the 19th Oct 2017

What does CINNAMON do?

Take a deep breath and prepare for many acronyms

What does CINNAMON do?

- catches **S**ystem **E**vent **L**ogs (**SEL**) records (= alerts that something is wrong on a node) eg: memory/CPU errors, power incidents
- collects **S**ensor **D**ata **R**epository (**SDR**) (= metrics that change over time) eg: temperatures, fans speed, voltages, currents
- makes data available to humans (ServiceNow, Grafana, InfluxDB)
- interacts with servers' **B**aseboard **M**anagement **C**ontrollers (**BMCs**) though **IPMI** messages

What is IPMI?

- stands for **I**ntelligent **P**latform **M**anagement **I**nterface
- specification led by Intel, in 1998 and supported by Cisco, DELL, HP, SuperMicro, QCT...
- works through *local bus* (ICMB) or *LAN*
- provides access to hardware sensors
- can store information in a non-volatile memory (critical events, serial numbers, model info)
- **has been adopted and required by our tender specifications**

Why IPMI?

- acts independently of the server
- it is available when servers are switched off
- homogeneous implementation across vendors
- availability of open-source tools (*ipmitool*, *ipmiutil*...)
- strong IT internal know-how
- de-facto standard in remote control

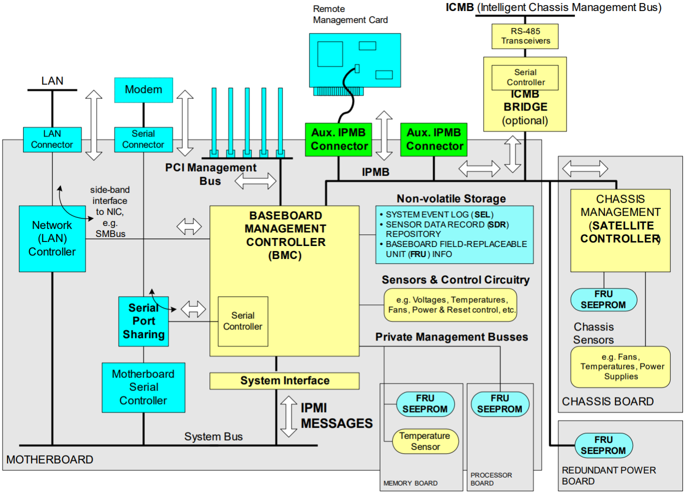


Figure: IPMI Specification, V2.0, Rev. 1.1 - section 1.7.3

System Event Logs entries

```
[root@p05798818d83430 ~]# ipmitool sel get 0002
SEL Record ID          : 0002
 Record Type           : 02
 Timestamp             : 06/25/2017 18:11:50
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Temperature
 Sensor Number         : 39
 Event Type            : Threshold
 Event Direction       : Assertion Event
 Event Data (RAW)      : 575d5d
 Trigger Reading       : 93.000degrees C
 Trigger Threshold     : 93.000degrees C
 Description           : Upper Non-critical going high
```

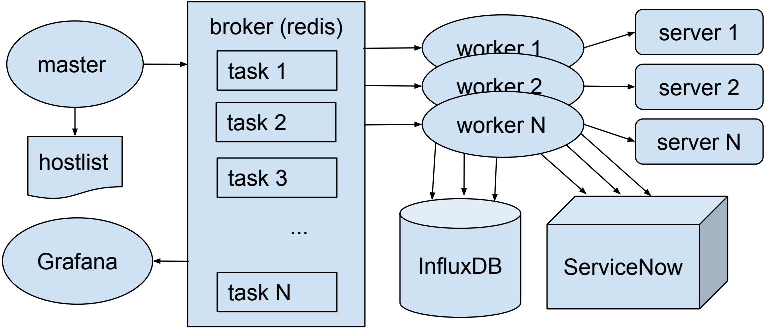
Sensor Data Repository entries

```
[root@p05798818d83430 ~]# ipmitool sdr elist
MB1_Temp         | 35h | ok  | 64.2 | 45 degrees C
MB2_Temp         | 36h | ok  | 64.1 | 49 degrees C
CPU0_Temp        | 37h | ok  |  3.1 | 43 degrees C
CPU1_Temp        | 38h | ok  |  3.2 | 41 degrees C
P0_DIMM_Temp     | 39h | ok  | 32.0 | 36 degrees C
P1_DIMM_Temp     | 3Ah | ok  | 32.1 | 33 degrees C
P5V              | 2Ah | ok  |  7.3 | 5.13 Volts
P3V3             | 15h | ok  |  7.2 | 3.39 Volts
P12V             | 29h | ok  |  7.5 | 12.10 Volts
Top_PSU_Status   | F1h | ok  | 10.1 | Presence detected
Bot_PSU_Status   | F2h | ok  | 10.2 | Presence detected
PSU_Redundancy   | F3h | ok  | 10.3 |
PSU_Input_Power  | F0h | ok  | 10.0 | 228 Watts
```

Advantages of out-of-band centralized monitoring

- no local running agent required (as opposed to ipmi-lemon-sensor)
- independence from operative systems (SLC6, CC7, C8, Windows)
- concurrent use of the *ICMB* local bus can lead to bricked nodes during *BIOS*/firmware upgrades
- local *ipmi_si* kernel driver systematic usage can cause other issues (CPU load $>=$ 100%)

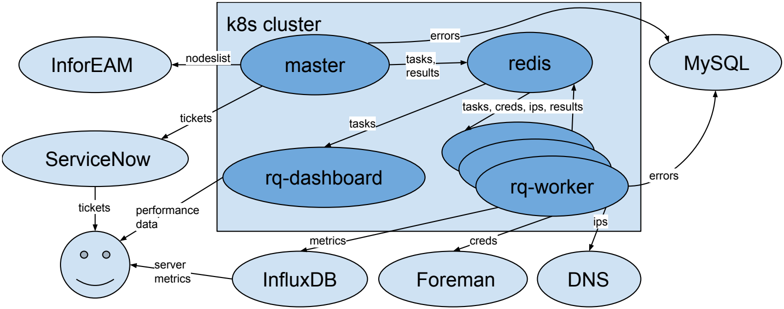
Design concept



CINNAMON enters production (2018)

- still running side-by-side with legacy lemon IPMI sensor
- containers (*docker*), based on SLC6
- still relying on LEMON/SNOW APIs, *collectd* offers grouping/de-duplication
- caching is unreliable, excessive usage of external resources (DNS, SSO, Foreman)
- credentials source of truth is now IPMIDB
- hard to troubleshoot (logs only on MySQL)
- data is available exclusively to IT-CF-FPP

Initial cluster architecture



Adoption of collectd: approach

- in order to compute a change in status and send a Notification[1], a *collectd* instance needs to be aware of the alerting state value of a metric
- workers are assigned random tasks from a nodeslist
- every worker would need to be aware of all the metrics of every monitored node [2]

[1] https://collectd.org/wiki/index.php/Notifications_and_thresholds

[2] May 2020: 34 metrics * 11000 nodes: 374000 records per instance (6 GB)

Adoption of collectd: solution

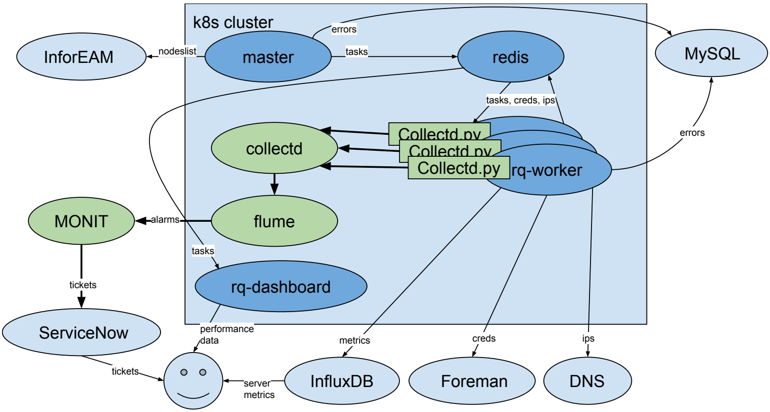
- use a stateful instance of *collectd* to coordinate the Threshold plugin alerts
- allow the worker pod to communicate directly with the *collectd* instance, implementing a Python version of *collectd* Network plugin's [3] binary protocol [4] directly in main task
- use *flume* to report threshold notifications to *MONIT* central infrastructure [5]

[3] https://collectd.org/wiki/index.php/Plugin:Network

[4] https://collectd.org/wiki/index.php/Binary_protocol

[5] https://monitdocs.web.cern.ch/monitdocs/alarms/collectd.html

Cluster architecture: evolution (I)



Adopt general services

- send SDR data to MONIT HTTP metrics sink [6]
- enhance errors and debug logging [7]
- request a private CERN ElasticSearch[8] instance for log ingestion
- get rid of our InfluxDB and MySQL instances (Database on Demand)

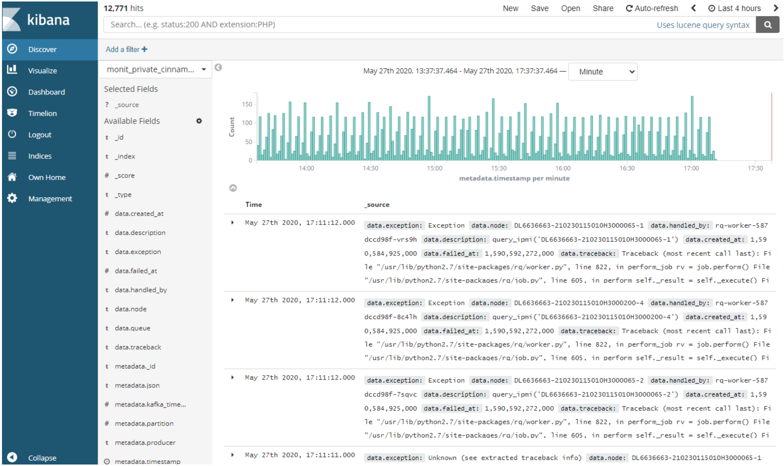
[6] https://monitdocs.web.cern.ch/monitdocs/ingestion/service_metrics.html

[7] many thanks to Luis Gonzalez for his contribution

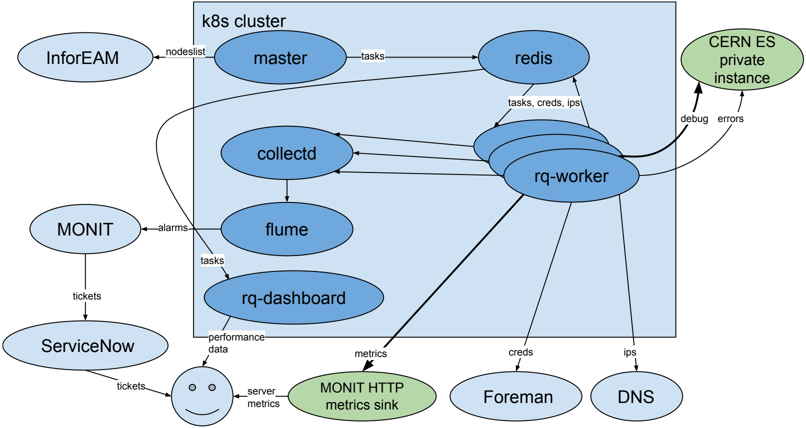
[8] https://monitdocs.web.cern.ch/monitdocs/logs/service_logs.html

Server metrics access on Grafana

CINNAMON private ES instance



Cluster architecture: evolution (II)



Credentials store restructuring

Problems:

- too many queries to Foreman APIs
- since the introduction of Ironic, Foreman doesn't retain all the credentials for the DC

Solutions:

- introduce IPMIDB-grabber (nightly credentials sync from Foreman and Ironic)
- rely solely on IPMIDB HTTP endpoint (high performance)

DNS issues: symptoms

- too many queries to CERN DNS
- caching appears to be inefficent
- very high metric drop rate (low SDR data flow but regular sweep time)
- pod restarts due to *NXDOMAIN* answers from the *CoreDNS* service

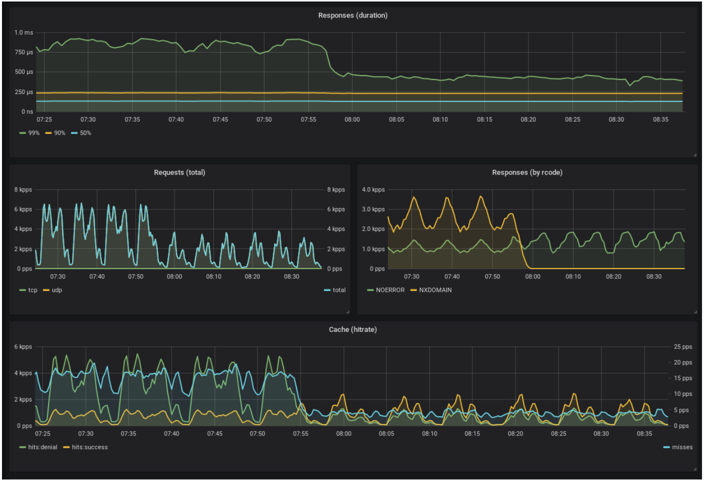
DNS issues: causes

- high NXDOMAIN:NOERROR ratio, due to the default *ClusterFirst* policy
- external DNS lookups from a pod will result in 3 futile cluster/local domain searches before searching for the bare domain name
- at our scale, this results in excessive I/O pressure on the *CoreDNS* pods, which will fall on the reliability of DNS query resolution.

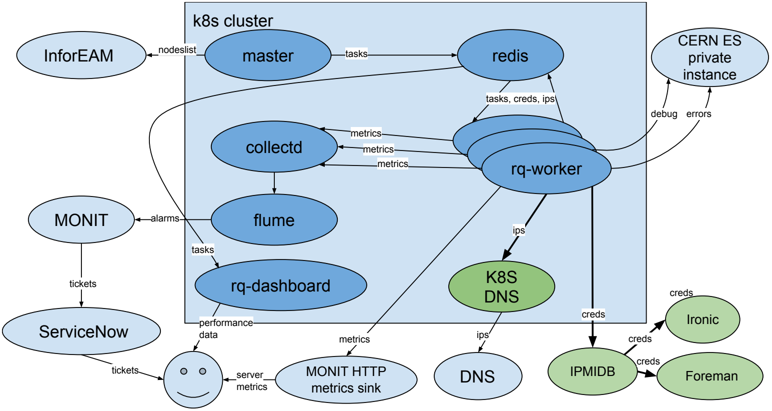
DNS issues: solutions

- increase number of *CoreDNS* replicas
- at least 4 replicas, not less than 1 every 64 cores
- enable *autopath* plugin for server-sided path resolution
- set *cache* plugin TTL to 3600s (1hr)
- rely on *CoreDNS* for caching

DNS issues: performance plot



Final cluster architecture

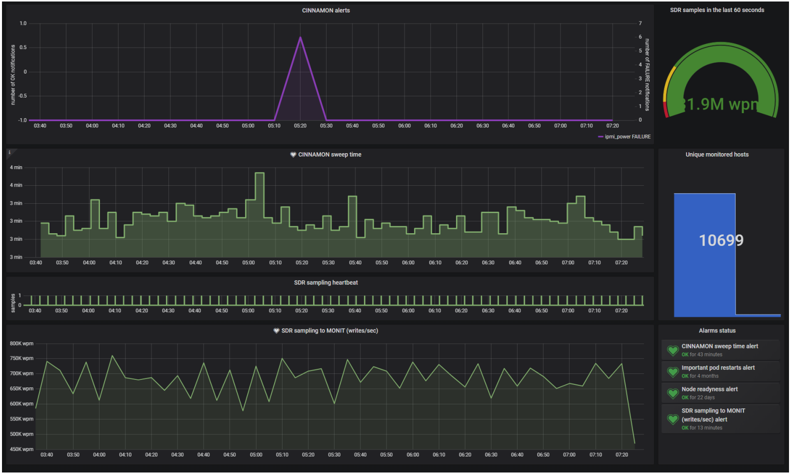


Resources usage

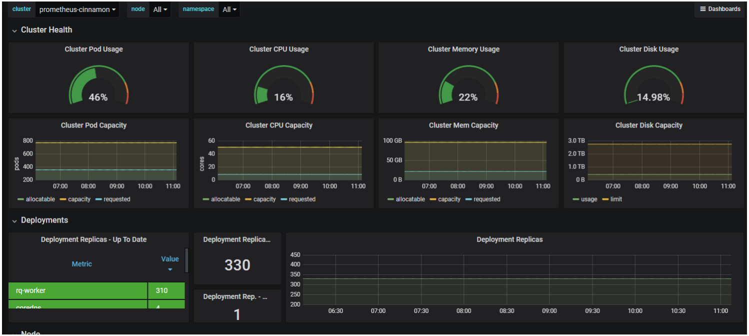
- 2 Kubernetes environments (prod, qa)
- prod: 6 m2.xlarge[9], 1 m2.medium[10] VMs
- qa: 1 m2.xlarge, 1 m2.medium VMs
- total of 59 VCPUs, 108GB RAM

[9] RAM: 14.6GB, 8 VCPUs, 80GB disk
[10] RAM: 3.7GB, 2 VCPUs, 20GB disk

Grafana dashboard



Prometheus cluster metrics



Grafana alerting

- full sweep time >6 minutes
- SDR samples sent to MONIT <10000/minute
- an important pod restarts (collectd, master, redis, flume)
- a cluster node is not in *Ready* state

Final considerations

- CINNAMON is reliable and production quality
- can grow with CERN computing requirements
- can change with CERN computing requirements
- could be a platform for all OOB centralized monitoring

Questions?

CERN

home.cern