Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Modern Monitoring Systems

An overview

Aleksander Paravac [1]
[1] *aleksander.paravac@uni-wuerzburg.de*

ISGC 2017

Overview

1 Monitoring
Environment
Operating System
Services

2 Issues with Monitoring

3 Modern approaches
Deployment

Overview

Environment

Check your environment

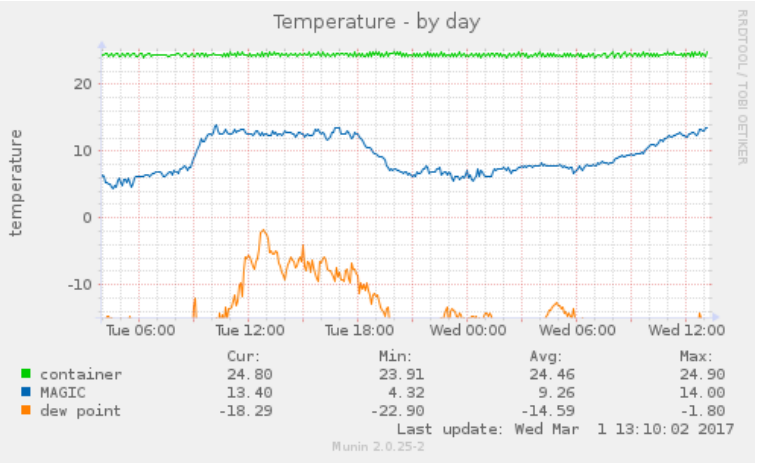
Health, e.g.

- CPU Temperatures, Fanspeeds
- Drive status (SMART values)
- Currents and Voltages (Mainboard, Power supply)

Climate

- Room temperature
- Humidity
- Dew point

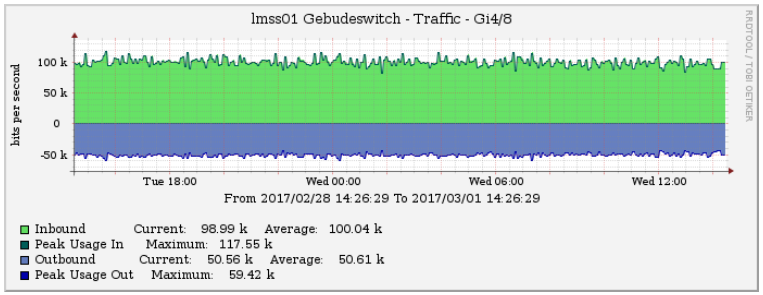
Dew point and temperature



Monitoring via Munin.

Environment

Traffic



Traffic graph displayed by Cacti.

Communication channels

Traffic, e.g.

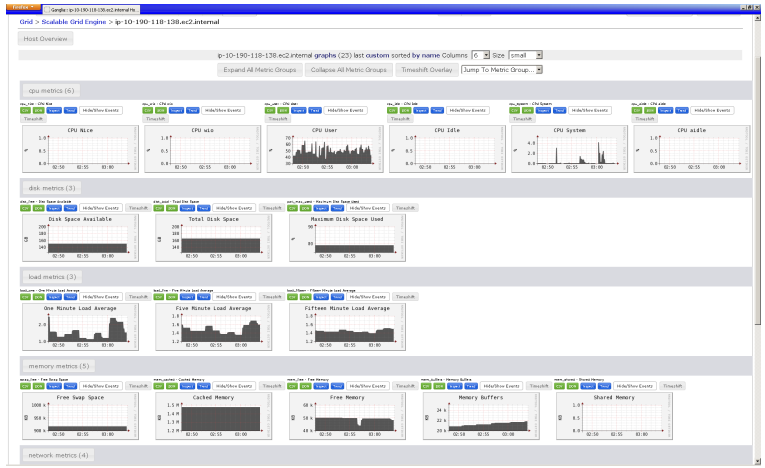
- In and outgoing traffic
- Number of connections
- Latency
- Link status

Operation System

Keep track of your OS

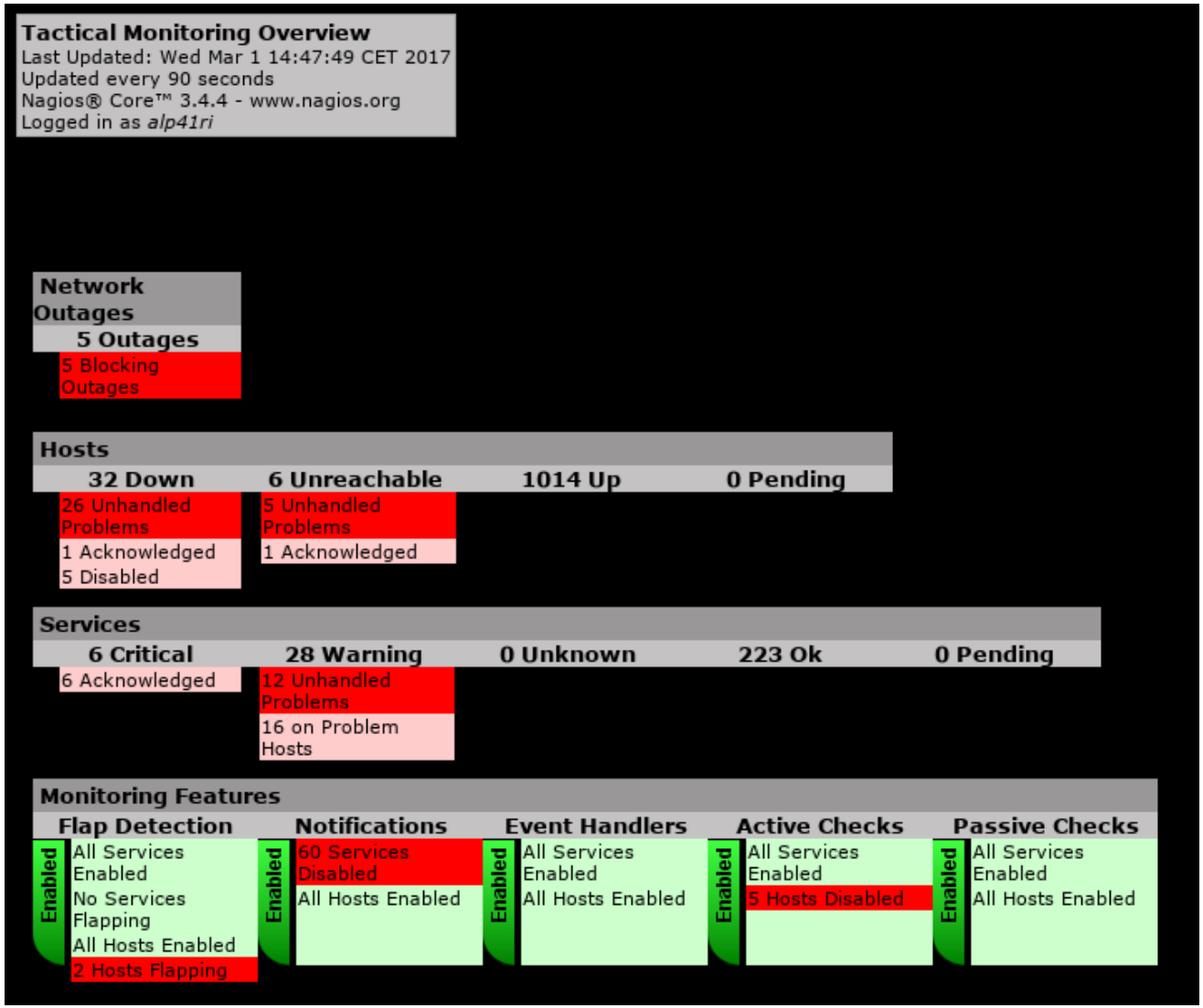
- CPU Usage / Load
- Memory Usage
- IO Usage and IRQ stats
- Number of processes
- Number of users

A lot of graphs



Statistics of your EC2 instance monitored via Ganglia.

Services


Tactical Monitoring Overview
Last Updated: Wed Mar 1 14:47:49 CET 2017
Updated every 90 seconds
Nagios® Core™ 3.4.4 - www.nagios.org
Logged in as alp41ri
Network Outages
5 Outages
5 Blocking Outages
Hosts
32 Down
26 Unhandled Problems
1 Acknowledged
5 Disabled
6 Unreachable
5 Unhandled Problems
1 Acknowledged
1014 Up
0 Pending
Services
6 Critical
6 Acknowledged
28 Warning
12 Unhandled Problems
16 on Problem Hosts
0 Unknown
223 Ok
0 Pending
Monitoring Features
Flap Detection
Enabled
All Services Enabled
No Services Flapping
All Hosts Enabled
2 Hosts Flapping
Notifications
Enabled
60 Services Disabled
All Hosts Enabled
Event Handlers
Enabled
All Services Enabled
All Hosts Enabled
Active Checks
Enabled
All Services Enabled
5 Hosts Disabled
Passive Checks
Enabled
All Services Enabled
All Hosts Enabled

Overview

Issues

RRD based tools

- Stores data in intervals of certain length, e.g. 5 minutes, 1 hour, 1 day, etc.
- Old data will be overwritten
- Missing datapoints will be interpolated
- Requires I/O syscalls which may not scale

Overview

Modern Approaches

Improvements

- Storing $\$key \rightarrow \$value$ pairs became scaleable
- Storage space got cheaper (Space for long term monitoring)
- Efficient use of client side resources instead of using server side resources
- Users now can analyse and interact with the data
- Availability of the data in different formats
- Configuration got more easily

Demonstration

+++ LIVE DEMO +++ LIVE DEMO +++

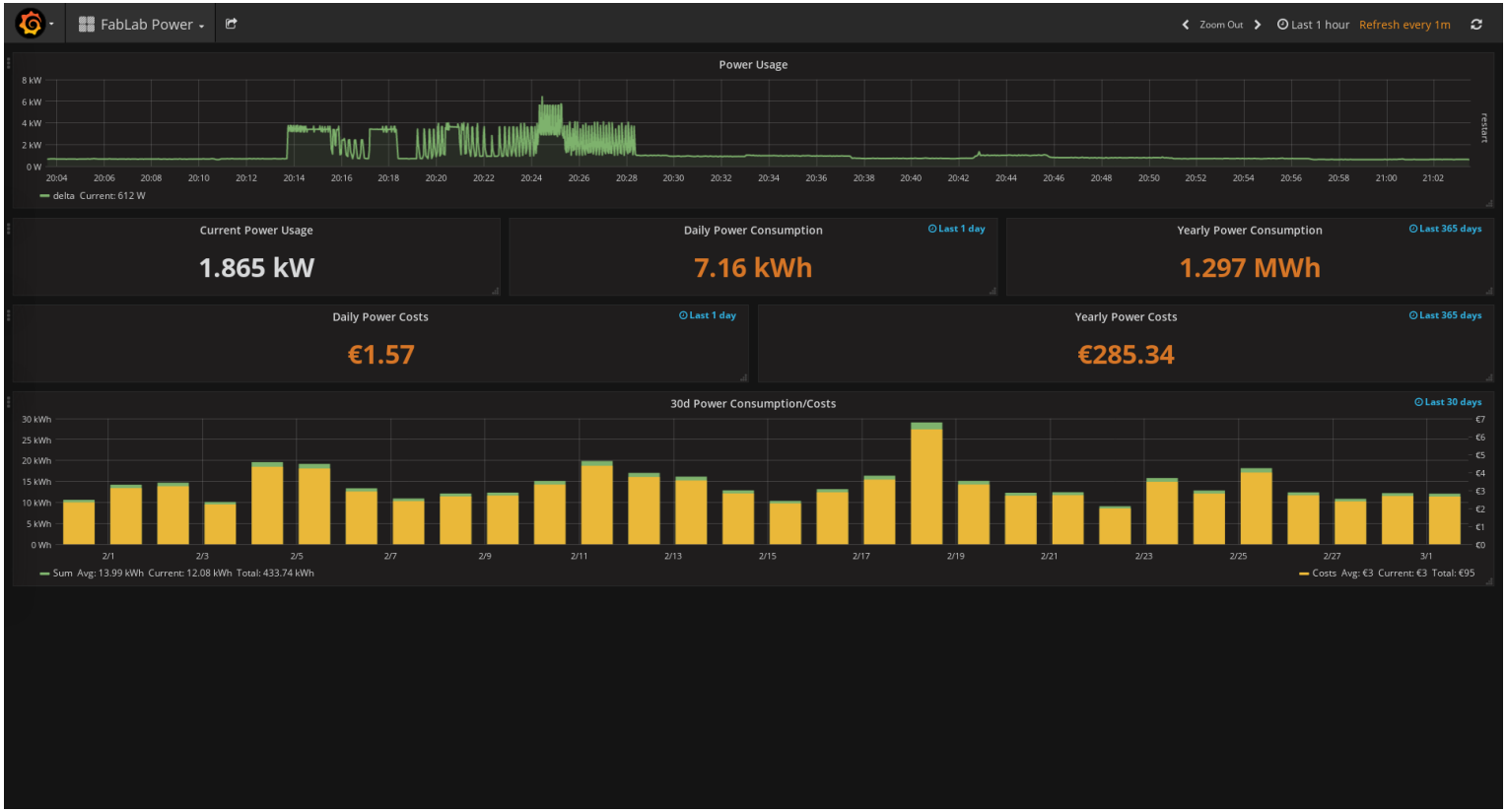
 Demotime 

+++ LIVE DEMO +++ LIVE DEMO +++

Grafana

Usual business

Grafana



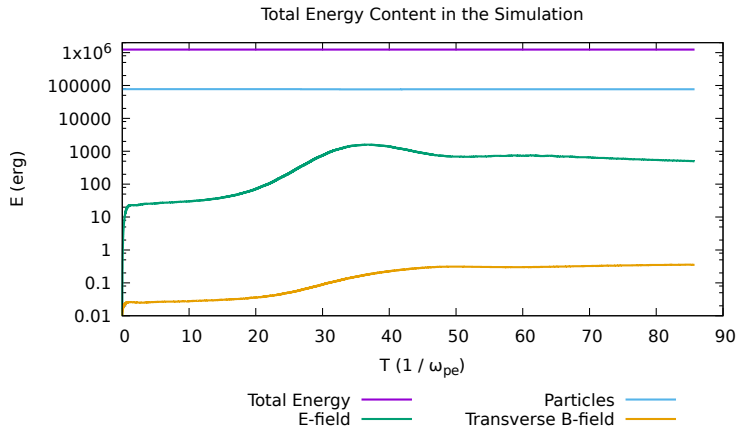
Making more out of one data series

Outlook

Look around you

- Increasing number of data due to the age of IoT
- Automatization with the help of real time monitoring
- Increasing amount of services provided by software containers
- In-application monitoring

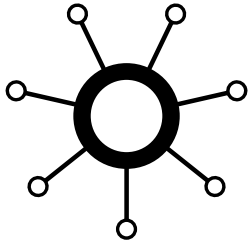
Monitoring your simulation



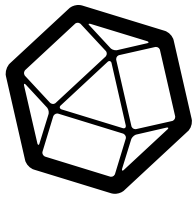
Total Energy should not change during simulation. Thanks to U. Ganse for the image.

Example

Collect data



Store data



influxdb

Visualize data

Interaction

collectd.conf

```
<Plugin network>
#       # client setup:
        Server "127.0.0.1" "25826"
#       <Server "127.0.0.1" "25826">
#               SecurityLevel Encrypt
#               Username "user"
#               Password "secret"
#               Interface "eth0"
#       </Server>
[...]
</Plugin>
```

Interaction

influxdb.conf

```
[[collectd]]
  enabled = true
  bind-address = "localhost:25826"
  database = "collectd"
  retention-policy = ""
  batch-size = 5000
  batch-pending = 10
  batch-timeout = "10s"
  read-buffer = 0
  typesdb = "/usr/share/collectd/types.db"
```

Demonstration

+++ LIVE DEMO +++ LIVE DEMO +++

 Demotime 

+++ LIVE DEMO +++ LIVE DEMO +++

Prometheus

Simple HTTP Server

```
package main

import "fmt"
import "log"
import "net/http"

func main()  {
  http.HandleFunc("/search",
    func(w http.ResponseWriter, r *http.Request) {
      fmt.Fprintf(w, "<h1>OHAI ISGC 2017</h1>")
    })
  log.Fatal(http.ListenAndServe(":2626", nil))
}
```

Prometheus

In-application monitoring

```
[...]
import "github.com/prometheus/client_golang/prometheus"

var queries = prometheus.NewCounter(prometheus.CounterOpts{
  Name: "queries",
  Help: "Queries for /search",
})

func init() {
  prometheus.MustRegister(queries)
}

func main()  {
  http.Handle("/metrics", prometheus.Handler())
  http.HandleFunc("/search",
    func(w http.ResponseWriter, r *http.Request) {
      fmt.Fprintf(w, "<h1>OHAI ISGC 2017</h1>")
      queries.Inc()
    })
  log.Fatal(http.ListenAndServe(":2626", nil))
}
```

Thank you

Thank you