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

1 Monitoring
   Environment
   Operating System
   Services

2 Issues with Monitoring

3 Modern approaches
   Deployment
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 graph displayed by Cacti.

Communication channels

Traffic, e.g.
- In and outgoing traffic
- Number of connections
- Latency
- Link status

Traffic graph displayed by Cacti.
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.
<table>
<thead>
<tr>
<th>Services</th>
<th>Monitoring Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Updated: Wed Mar 1 14:47:49 CET 2017</td>
<td></td>
</tr>
<tr>
<td>Updated every 90 seconds</td>
<td></td>
</tr>
<tr>
<td>Nagios® Core™ 3.4.4 - <a href="http://www.nagios.org">www.nagios.org</a></td>
<td></td>
</tr>
<tr>
<td>Logged in as alp4ri</td>
<td></td>
</tr>
</tbody>
</table>

## Network Outages
- 5 Outages
- 5 Blocking Outages

## Hosts
- 32 Down
- 6 Unreachable
- 1041 Up
- 0 Pending
- 26 Unhandled Problems
- 5 Unhandled Problems
- 1 Acknowledged
- 1 Acknowledged
- 5 Disabled

## Services
- 6 Critical
- 28 Warning
- 0 Unknown
- 223 Ok
- 0 Pending
- 6 Acknowledged
- 12 Unhandled Problems
- 16 on Problem Hosts

## Monitoring Features
<table>
<thead>
<tr>
<th>Flap Detection</th>
<th>Notifications</th>
<th>Event Handlers</th>
<th>Active Checks</th>
<th>Passive Checks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>No Services</td>
<td>60 Services</td>
<td>All Services</td>
<td>All Services</td>
<td>All Services</td>
</tr>
<tr>
<td>Flapping</td>
<td>Disabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>All Hosts Enabled</td>
<td>All Hosts Enabled</td>
<td>Enabled</td>
<td>All Services Enabled</td>
<td>All Services Enabled</td>
</tr>
<tr>
<td>2 Hosts Flapping</td>
<td>6 Hosts Disabled</td>
<td>Enabled</td>
<td>All Services Enabled</td>
<td>All Services Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A. Paravac (RZ Uni Würzburg) Monitoring
Overview

1 Monitoring
   Environment
   Operating System
   Services

2 Issues with Monitoring

3 Modern approaches
   Deployment
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

1 Monitoring
   Environment
   Operating System
   Services

2 Issues with Monitoring

3 Modern approaches
   Deployment
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
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.
Collect data

Store data

Visualize data
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>
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 +++

_players

Demotime

+++ LIVE DEMO +++ LIVE DEMO +++
### Simple HTTP Server

```go
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))
}
```
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