

Monitoring im 21. Jahrhundert

Sebastian 'tokkee' Harl
<sh@tokkee.org>

collectd core developer

Grazer Linuxtage 2014
04. April 2014
Graz





- Wer verwendet Nagios/Icinga/Naemon/OpenNMS/etc.?



- Wer verwendet Nagios/Icinga/Naemon/OpenNMS/etc.?
- Wer verwendet Performance-Daten seines Monitoring-Systems?



- Wer verwendet Nagios/Icinga/Naemon/OpenNMS/etc.?
- Wer verwendet Performance-Daten seines Monitoring-Systems?
- Wer basiert (den Großteil) sein(es) Monitorings auf Performance-Daten?



- Wer verwendet Nagios/Icinga/Naemon/OpenNMS/etc.?
- Wer verwendet Performance-Daten seines Monitoring-Systems?
- Wer basiert (den Großteil) sein(es) Monitorings auf Performance-Daten?
- Wer kennt **collectd**?



- Wer verwendet Nagios/Icinga/Naemon/OpenNMS/etc.?
- Wer verwendet Performance-Daten seines Monitoring-Systems?
- Wer basiert (den Großteil) sein(es) Monitorings auf Performance-Daten?
- Wer kennt **collectd**?
- Wer kennt Riemann-Monitoring?



Warum Monitoring auf Performance-Daten basieren?

Umdenken: „Was passiert?“ statt „Wie ist der Status?“

- Mehr Information als { OK, WARNING, CRIT }
- Push statt Poll → besser skalierbar
- Einfache(re) Aggregation
 - z.B. Monitoring homogener Systeme / Cluster
 - z.B. verschiedene Granularität
- Einfache(re) Korrelierung
 - z.B. Identifizierung von Hotspots
 - z.B. Identifizierung von Anomalien
- Trending → Kapazitätsplanung
- ⇒ graphische Aufbereitung sehr wichtig

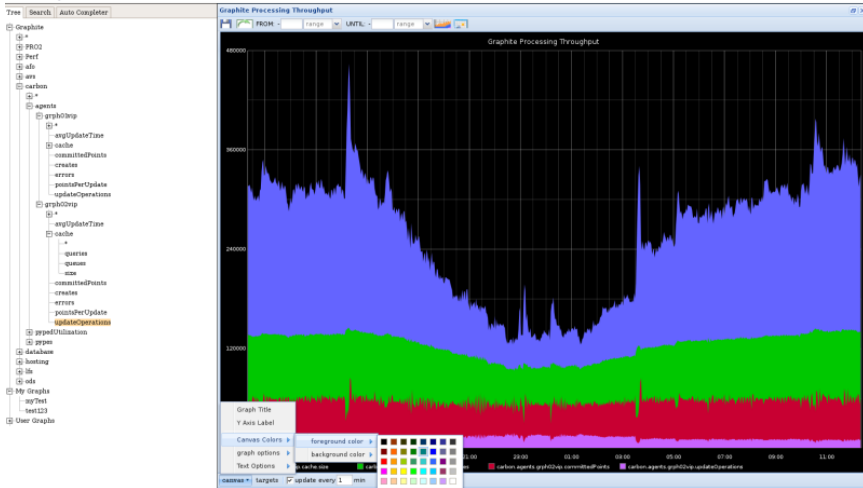
Monitoring mit collectd, Riemann und Graphite



- Graphite
 - <http://graphite.wikidot.com>
 - Echtzeit Graphing
 - Dynamisch, keine Konfiguration von Datensätzen
- **collectd**
 - <https://collectd.org>
 - Daemon zum Sammeln, Verarbeiten und Speichern von Performance-Daten
 - Sehr effizient (10 Sekunden Standardauflösung)
- Riemann
 - <http://riemann.io>
 - Stream-Verarbeitung und Monitoring
 - Schwerpunkt auf verteilte Systeme



- Datenbank für Time-Series Datensätze (Whisper-DB und Carbon-Daemon)
- Unterstützt lesen aus RRD Dateien
- Web-Frontend zur Anzeige und Konfiguration von Graphen
- Erstellen von Graphen per drag-and-drop
- Übermitteln von Daten durch ein einfaches Text-Protokoll





- Mehrere Subprojekte:
 - `git://github.com/graphite-project/ceres.git`
 - `git://github.com/graphite-project/whisper.git`
 - `git://github.com/graphite-project/carbon.git`
 - `git://github.com/graphite-project/graphite-web.git`
- Doku: <http://graphite.readthedocs.org/en/latest/>
- Installation etwas aufwändiger :-/
 - zum Glück gibt's Pakete in Debian :-)



- Alle Funktionalität steckt in Plugins
- > 100 Plugins in der Standarddistribution enthalten
 - Basis-Statistiken: CPU, Speicher, Load, etc.
 - Viele fortgeschrittene Plugins: vmem, iptables, etc.
 - Div. Erweiterungen: z.B. Carbon, JVM



- HEAD: `git://github.com/collectd/collectd`
→ Bootstrap: `./build.sh`

```
% sudo apt-get install default-jdk \  
    libprotobuf-c0-dev protobuf-c-compiler  
% wget http://collectd.org/files/collectd-5.4.1.tar.gz  
% tar xzf collectd-5.4.1.tar.gz  
% cd collectd-5.4.1
```



- Java ist lustig :-/
- `--with-perl-bindings` Workaround für Debian/Ubuntu

```
% JAVA_HOME=/usr/lib/jvm/default-java
% JAVA_LDIR=$JAVA_HOME/jre/lib/amd64/server
% ./configure --enable-java --enable-write_graphite \
  --enable-write_riemann --with-perl-bindings="" \
  JAVAC=/usr/bin/javac JAR=/usr/bin/jar \
  JAVA_CPPFLAGS=-I$JAVA_HOME/include \
  JAVA_LDFLAGS="-L$JAVA_LDIR -Wl,-rpath -Wl,$JAVA_LDIR"
# -> Zusammenfassung und ggf. Fehler beachten
% make
% sudo make install
```



Minimale Konfiguration:

```
# siehe auch collectd.conf(5)
LoadPlugin syslog # debugging!
LoadPlugin cpu
LoadPlugin write_graphite

<Plugin write_graphite>
  <Node "local">
    Host "localhost"
    Port "2003"
    Prefix "collectd"
  </Node>
</Plugin>
```




- In Clojure geschrieben und konfiguriert
- Filtern, Verarbeitung und Kombination von Event-Streams
- Einfache Erweiterung durch Clients (Kommunikation mittels Protobufs)
- Läuft auf JVMs und damit in **collectd**
→ collmann (proof-of-concept)
- Kann Daten auch zu Graphite schreiben



```
% sudo apt-get install leiningen
% git clone git://github.com/exoscale/collmann
% cd collmann
% lein uberjar
% cp collmann-*-standalone.jar /opt/collectd/lib/collectd
```



collectd.conf:

```
LoadPlugin java
<Plugin java>
    JVMArg "-Dlog4j.configuration=\
            file:///path/to/collmann/resources/\
            log4j.properties"
    JVMArg "-Djava.class.path=/opt/collectd/lib/collectd/\
            collmann-0.1.0-standalone.jar"
LoadPlugin "io.exo.collmann.core"
<Plugin "riemann">
    LoadConfig "/path/to/collmann/doc/collectd.clj"
</Plugin>
</Plugin>
```



- <http://riemann.io/howto.html>
- <http://riemann.io/api.html>

```
(where (or (service #"^api")
           (service #"^app"))
      (where (tagged "exception")
             (rollup 5 3600
                  (email "dev@foo.com")))
      (else
        (changed-state
         (email "ops@foo.com")))))
```



```
(where collectd?  
  (rate 1  
    (with {  
      :service "collman events"  
      :plugin "collmann"  
      :type "gauge"  
      :type_instance "events"  
    }  
    index collectd-index)))
```

Weitere Infos



- **sysdb** – Zusammenfassung von Informationen aus mehreren Systemen (→ siehe Vortrag morgen 15:00 Uhr in HS14)
- **collectd** – Kombination mit Nagios
<https://collectd.org/wiki/index.php/Collectd-nagios>
- **collectd** – viele weitere Kombinationsmöglichkeiten
→ statsd, logstash, Ganglia, AMQP, u.v.m.
- **Riemann** – Kombination mit Nagios (NSCA)
<http://riemann.io/api/riemann.nagios.html>
- **Nagios** – diverse Erweiterungen / Addons
<http://tokkee.org/talks/glt13-monitoring.pdf>
- <https://github.com/monitoringsucks/tools-repos>



Danke für die Aufmerksamkeit!

Fragen?



Kontakt: Sebastian "tokkee" Harl <sh@tokkee.org>
Feedback: <http://gl1t13-programm.linuxtage.at/>