

SysDB – System DataBase

Alle Systeminformationen auf einen Blick

Sebastian 'tokkee' Harl

<tokkee@sysdb.io>

**OPEN
RHEIN
RUHR**

Ein Pott voll Software

9. November 2014





HINWEIS:

SysDB ist noch in der Entwicklung
und wird noch nicht als stabil angesehen¹.

Flaming, Bashing oder andere Formen von konstruktivem
Feedback sind sehr willkommen ... oder **Mitwirkung** :-)

¹ I.S.v. Änderungen an Schnittstellen



Motivation / Hintergrund





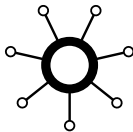
Naemon

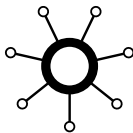
 **ICINGA**

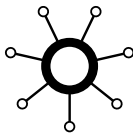




Naemon

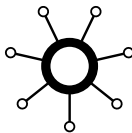






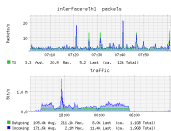
- Database Server Foo
- External Gateway
- Zooperserver

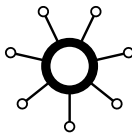




- Database Server Foo
- External Gateway
- Zooperserver

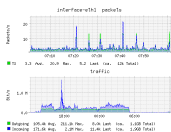
gw1.domain.tld





- Database Server Foo
- External Gateway
- Zooperserver

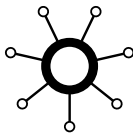
gw1.domain.tld



gw1.domain.tld

arch=amd64
lsbdistid=Debian
loc=BayArea



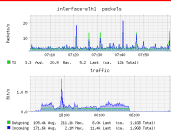


● Database Server Foo

● External Gateway

● Zooperserver

gw1.domain.tld



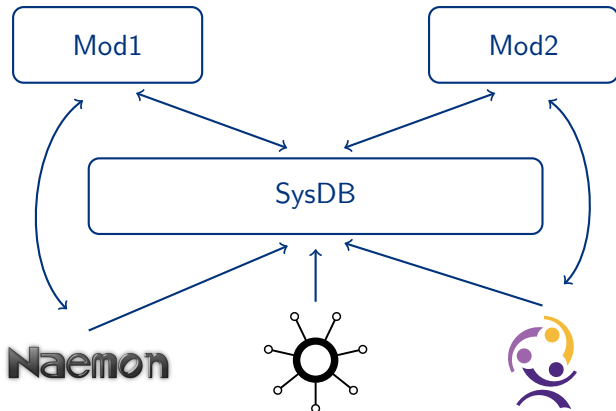
gw1.domain.tld

arch=amd64
lsbdistid=Debian
loc=BayArea



Die System DataBase







- SysDB aggregiert Informationen zu beliebigen Hardware- und Software-Systemen zu einem Gesamtüberblick
- Fokus auf Metadaten und Verknüpfung zu anderen Informationen
- Einfache Beispiele
 - Hosts und deren Eigenschaften („facts“)
 - Services und deren Eigenschaften
 - Metrics (Verweis auf Performance-Daten)
 - Monitoring Informationen
- SysDB sammelt diese Informationen von verschiedenen Systemen und korreliert zusammengehörige Informationen





- <https://sysdb.io/>, <https://github.com/sysdb>
 - CI: <https://travis-ci.org/sysdb/sysdb>
 - ~ 80% Code (Funktion) Unit-Test Coverage im Core
- BSD Lizenz
- Geschrieben in C (UI in Go WIP)
- Generische Speicherung von aggregierten Systeminformationen
- Multi-threaded, event-based client/server Architektur
- Einfach erweiterbar (Plugin API)
- Einfaches Netzwerk-Protokoll und Abfragesprache
- Großteil des Codes als Bibliothek (wiederverwendbar)





Aktuell verfügbare Backends (Kollektoren):

- **collectd::unixsock** – Abfrage von collectd (unixsock Plugin)
→ host metrics
- **mk-livestatus** – Abfrage von Monitoring Systemen (Nagios, Naemon, Icinga, Shinken) mittels Check_MK Livestatus
→ Hosts und Services (Metrics geplant)
- **puppet::store-configs** – Abfrage von Puppet
→ Host Attribute

Geplant: Passive Datensammlung (z.B. über Gearman), Foreman, PuppetDB, \$your_favorite_system (send patches!) :-)





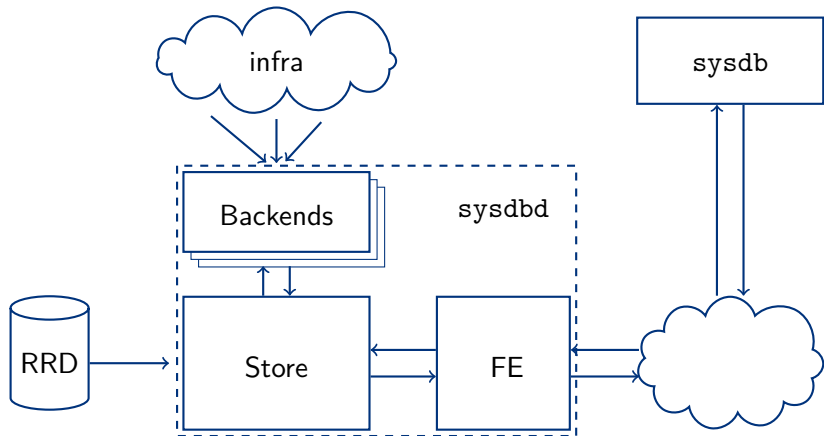
Time-series Plugins:

- `timeseries::rrdtool` – Abfrage von Timeseris aus lokalen RRD Dateien

Verschiedene Plugins:

- `cname::dns` – Normalisierung von Hostnamen mittels DNS
- `syslog` – syslog Logging







- Speichert generische Objekte (in-memory atm)
- Normalisierung von Hostnamen
- Jedes Objekt speichert Zeitpunkt der letzten Aktualisierung und automatisch berechnetes Update-Intervall

- Schnittstelle zur Datenabfrage
- Generischer Zugriff auf Timeseries Daten
- JSON als externe Repräsentation von Objekten

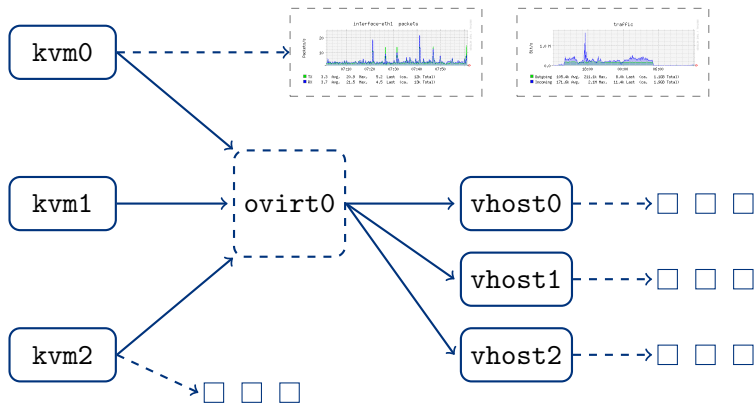




- Host – jede Art physikalischer Ressource
- Service – Dienste (im weitesten Sinne)
- Metric – Bezeichner für Metriken
 - Timeseries Daten werden **nicht** in SysDB gespeichert
- Attribute – Eigenschaften von Hosts und Services
 - verschiedene Datentypen unterstützt
- Alle Objekte speichern:
 - eindeutiger Name
 - Zeitpunkt des letzten Updates
 - Update Intervall
 - Liste von Backends



Der SysDB Store – Beispiel



(Unterstützung von Graphdaten geplant.)





- Interaktiver Client für SysDB
- Verbindet sich mit einem SysDB Daemon
- Interaktive Kommando-Shell
- Zeigt interaktive Log-Meldungen vom Server an





- Entfernt an SQL angelehnt
- Aber es ist nicht SQL¹
- Auflistung von Abfrage von Hosts, Services, Metrics
- Abfrage von Timeseries Daten (von einem Backend Data-Store)

¹ Eine DSL war die einfachste Option ;-)





- LIST hosts|services|metrics
- FETCH host <hostname>
- FETCH service|metric <hostname>.<name>
- LOOKUP hosts|services|metrics MATCHING <condition>
- ... FILTER <condition>
- TIMESERIES <host>.<metric>
START 2014-01-01 END 2014-12-31

→ <https://sysdb.io/manpages/head/sysdbql.7.html>





```
sysdb=> LIST hosts FILTER age < 2 * interval;
[ {
  "name": "monitor.lxc.tokkee.net",
  "last_update": "2014-04-03 10:26:41 +0200",
  "update_interval": "5m4s",
}, {
  "name": "puppet.lxc.tokkee.net",
  "last_update": "2014-04-05 11:04:08 +0200",
  "update_interval": "5m2s"
}]
```





```
sysdb=> LOOKUP hosts WHERE attribute['arch'] = 'amd64'
                                AND ANY service =~ 'postgres';
{ "name": "db.lxc.tokkee.net",
  "last_update": "...", "update_interval": "10s",
  "attributes": [{
    "name": "architecture", "value": "amd64",
    "last_update": "...", "update_interval": "5m3s"
  },{ ... }],
  "services": [{
    "name": "PostgreSQL Master",
    "last_update": "...", "update_interval": "5m"
  },{ ... }],
  "metrics": [{...}]
}
```





Abfrage- und Filter-Bedingungen:

```
MATCHING attribute['architecture'] = 'amd64'  
      AND ANY service =~ 'postgres'
```

```
MATCHING name =~ /\.tokkee\.net$'
```

```
FILTER age >= 3 * interval
```

```
FILTER NOT 'backend::mk-livestatus' IN backend
```



Anwendungsbeispiele





- Mächtige Web-Frontends für mehrere Backends
 - Informationen korrelieren oder annotieren
 - zentrales Dashboard
 - Integration von spezifischeren Frontends
- Abgleich diverser Backends (Monitoring)
 - Welche Hosts / Services fehlen in welchem Backend?
 - Basis für Businessprozesse: Wie ist der Gesamtstatus meiner Windows-Systeme in Rechenzentrum XYZ?
- Erweiterung von CMDBs durch dynamische Informationen



Ausblick: künftige Entwicklungen





- Bessere und mehr Integration mit anderen Systemen
- Persistenter Store: RDBMS, Graphdatenbank (?)
- Interface zur Abfrage von anderen Live-Daten (z.B. Monitoring-Status) von Backends
- Verteilte Architektur (HA und Load-Balancing)
- Erweiterbares Web-Interface
- ...






Danke für die Aufmerksamkeit
Fragen, Kommentare, Rants?



<https://sysdb.io/>, <https://github.com/sysdb>

 <https://sysdb.io/+>  <https://sysdb.io/facebook>

 @SystemDatabase

