Orchestrating a brighter world

NEC

[FEPA]

Monitoring Infrastructure

July 2017

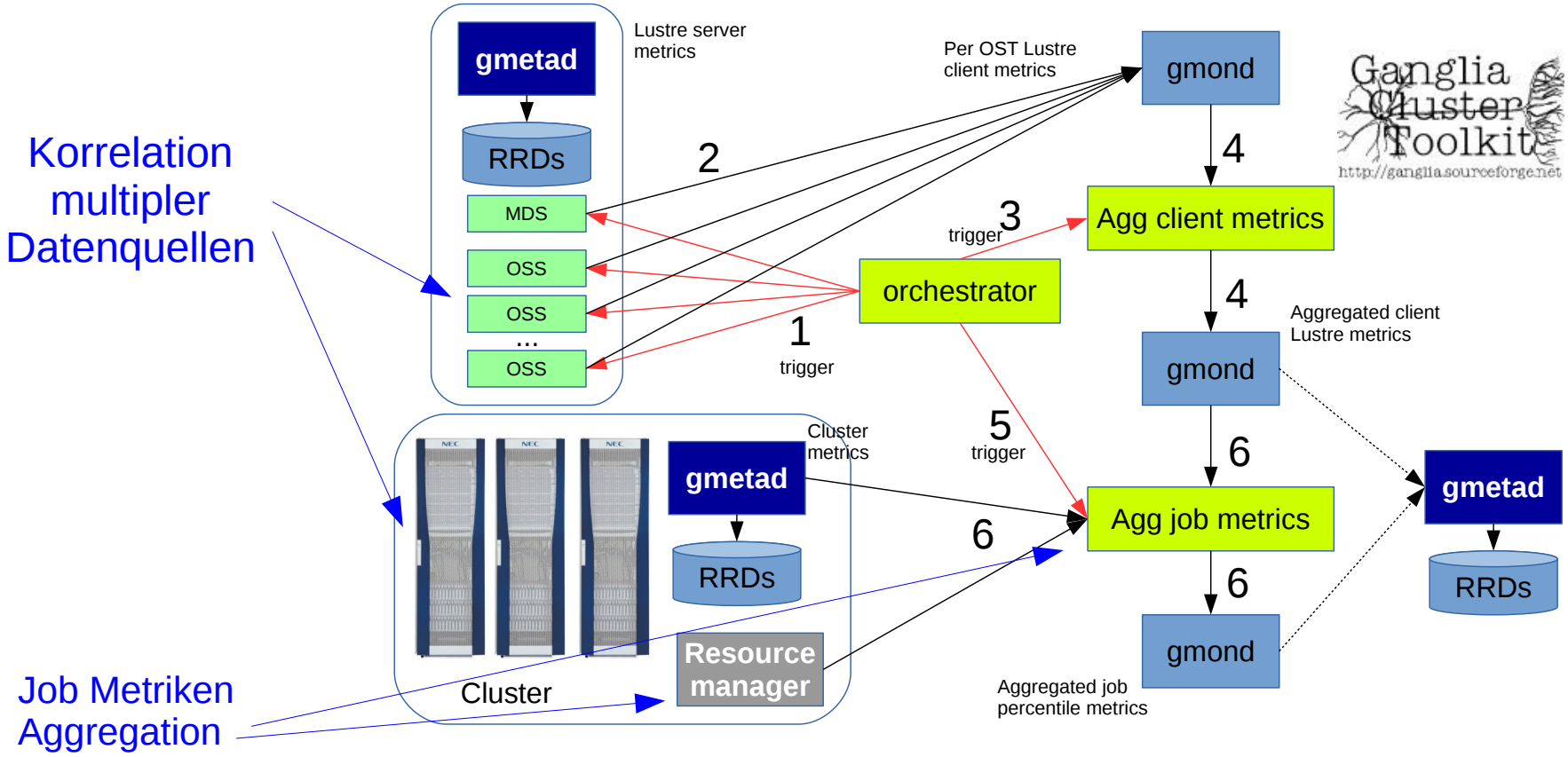
Erich Focht, Andreas Jeutter
NEC HPC Europe

Motivation: flexible Monitoring Architektur

- ***Einschränkungen erzwingen Anpassungen***
  - Zahl der Metriken
  - Zahl der Jobs
  - Zahl der Compute Nodes
  - Message Rate, Messfrequenz
  - Aggregation: CPU Last
  - Historische Daten
    - Wie lang aufbewahren?
    - Ausgedünnt oder nicht?
    - Exportieren? Pro Job?

- Ansatz:
  - **Hierarchie! Workflow! Datenmodell!**

Experiment: Workflow mit Ganglia



Experiment: Workflow mit Ganglia

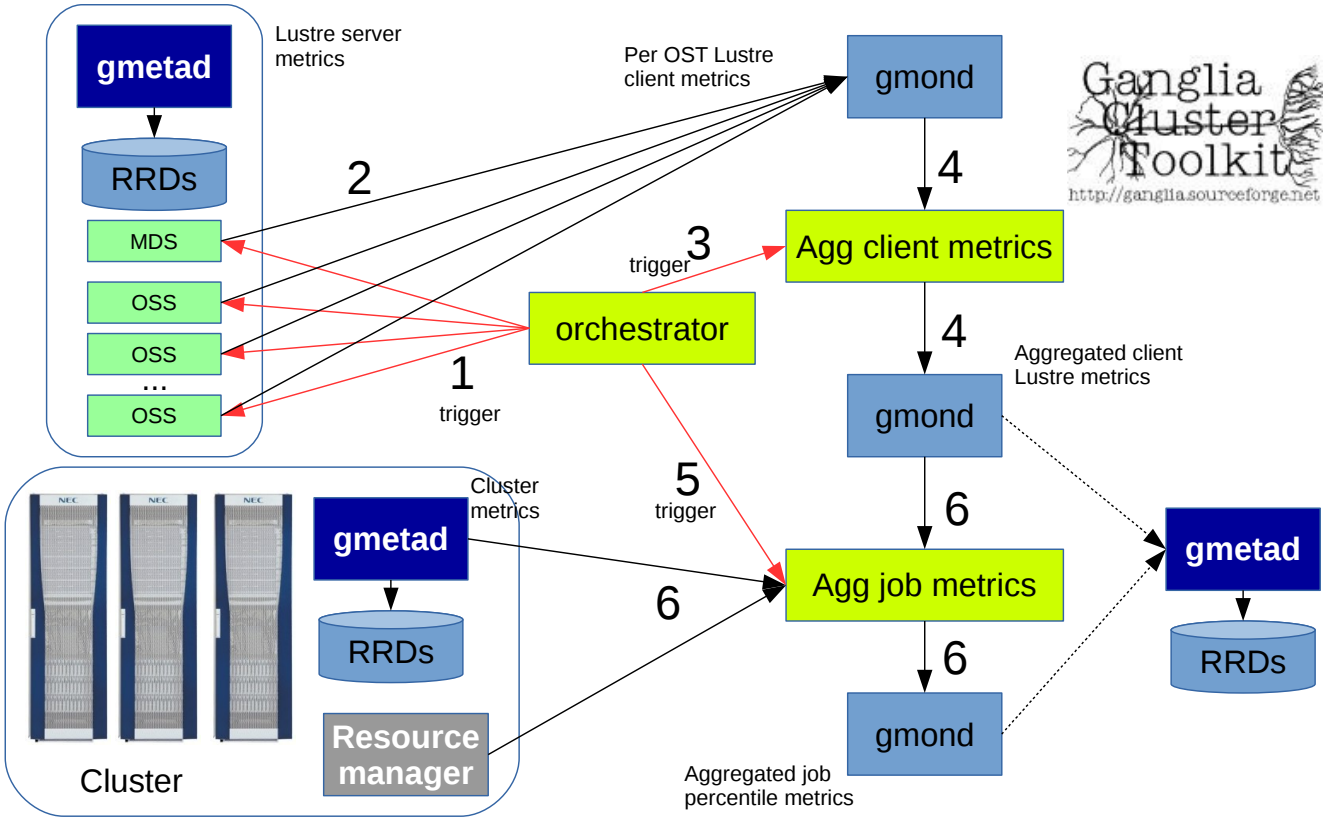
Zahl der Metriken:
(500 Nodes, 20 OSTs)

Lustre:
- 4 Metriken/Client/OST = 40000 → 2000
- 16 MDT Metriken/Client = 8000

Cluster:
- 50-200 Metriken/Node = 25k-100k

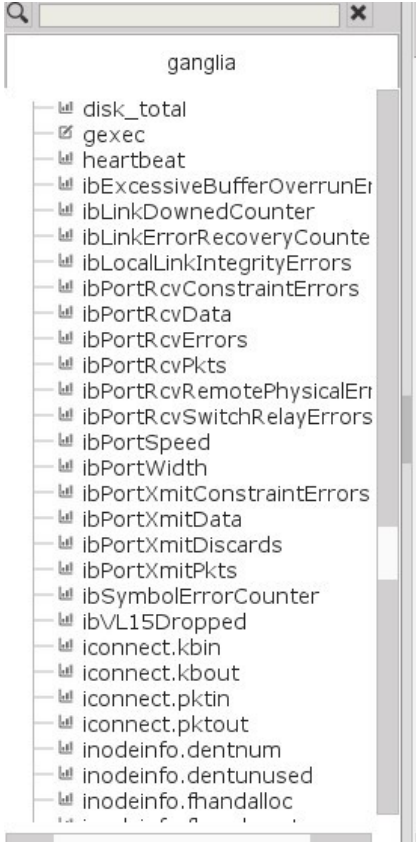
Quantilaggregation:
- LIKWID + Lustre / Job
- O(20) Metriken, 12 Werte

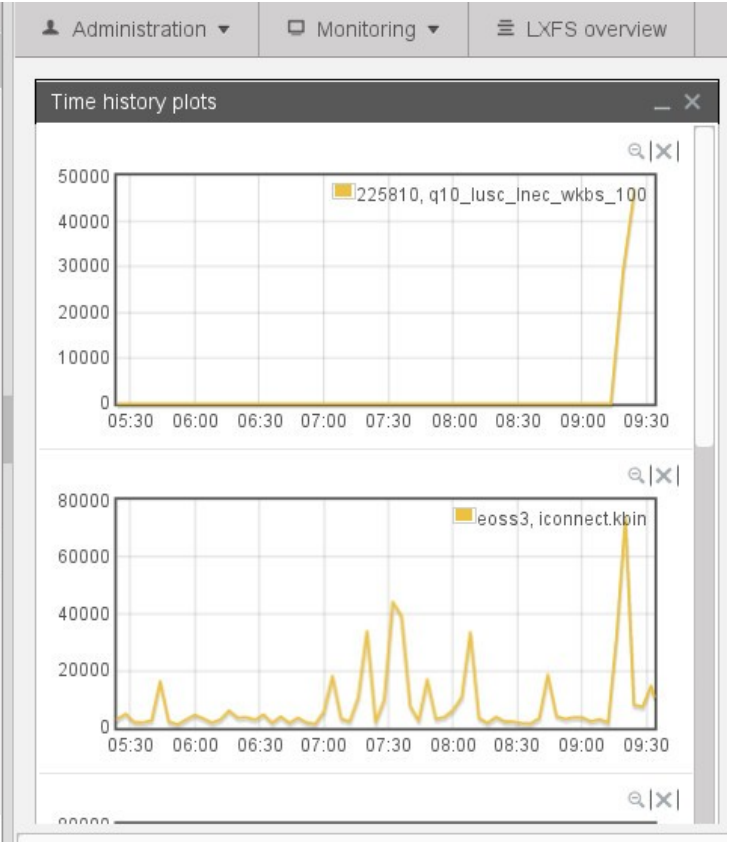
Frequenz:
- 1 – 5 Minuten



Experiment: Korrelation mit LxGUI

- Plots verschiedener Metriken
  - Aus unterschiedlichen Quellen
  - Untereinander
  - Mit gleicher Zeitskala
    - Gemeinsam zoom- und scrollbar





Workflow mit Ganglia: Probleme und Grenzen

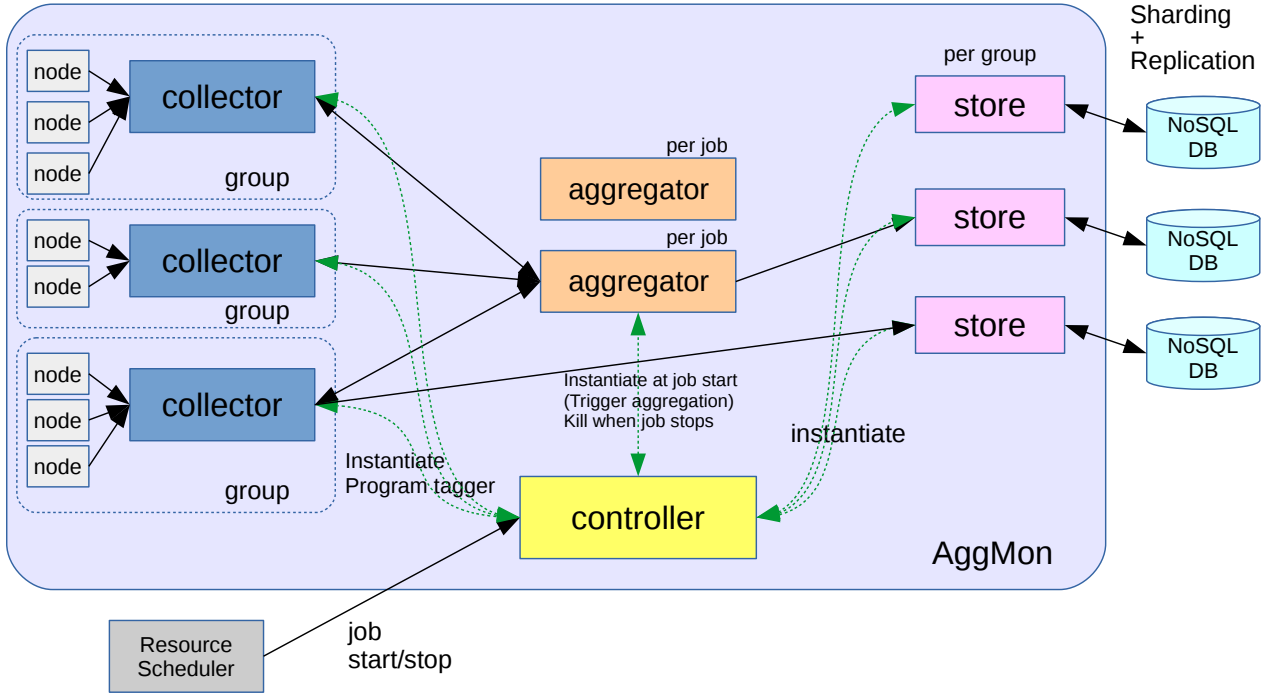
- Synchronisierte Datensammlung
  - Nötig für gute Qualität der Quantilmetriken
  - Hohe Bursts O(10k/s) im “Schritt 1” des Orchestrators, am Limit von Ganglia
- Quantilaggregation
  - Bei RRZE in der Zeit: viele kleine Jobs, 1 Node, Aggregation sinnlos
  - Python code, parallelisiert: kaum möglich die Zykluszeit von 3 Minuten einzuhalten
- Datenspeicherung auf RRDs
  - Jeder Job entspricht einem neuen Host in Ganglia
  - Jeder Wert einer Quantilmetrik entspricht einer Ganglia Metrik
  - Eine Ganglia RRD Datei belegt 630760 Bytes, unabhängig von der Zahl der Datenpunkte
  - Eine 10% Quantilmetrik: 12 Werte pro Zeitschritt
  - 20 Quantilmetriken pro Job: 12 * 20 * 630760Bytes = 144.4MiB pro Job
  - Beim RRZE im Testzeitraum: >500 Jobs / Tag : → 72.2GiB Plattenplatz / Tag
  - Management (life-cycle) der Job RRDs nötig!

Aggmon: Monitoring Framework

- Im TIMACS BMBF Projekt angefangen (v1)
  - Hierarchisch verteilt, skalierbar: Administrationshierarchie der Knoten
  - topic basiertes PubSub: AMQP
  - Eingebaute flexible Aggregation
  - eigene Time Series Datenbank
  - Python, monolythisch, GIL verhindert Skalierung
- Im FEPA Projekt weiterentwickelt (v2)
  - AMQP durch ZeroMQ ersetzt, Message Rate von O(1k-2k) hoch auf O(50k)
  - Selbst implementiertes PubSub mit flexiblem Matching (PUSH-PULL)
  - RPC Layer auf ZeroMQ: REQ-REP
  - Separate Python Prozesse statt monolythischem Daemon
    - Skalierbar, verteilt, mit ZeroMQ gekoppelt
  - Metrik Datenspeicher: NoSQL Datenbank: MongoDB
    - Eigentlich TokuMX (Percona)
  - Integration von Job Informationen
  - Datenmodell erlaubt Job- und Administrationshierarchie

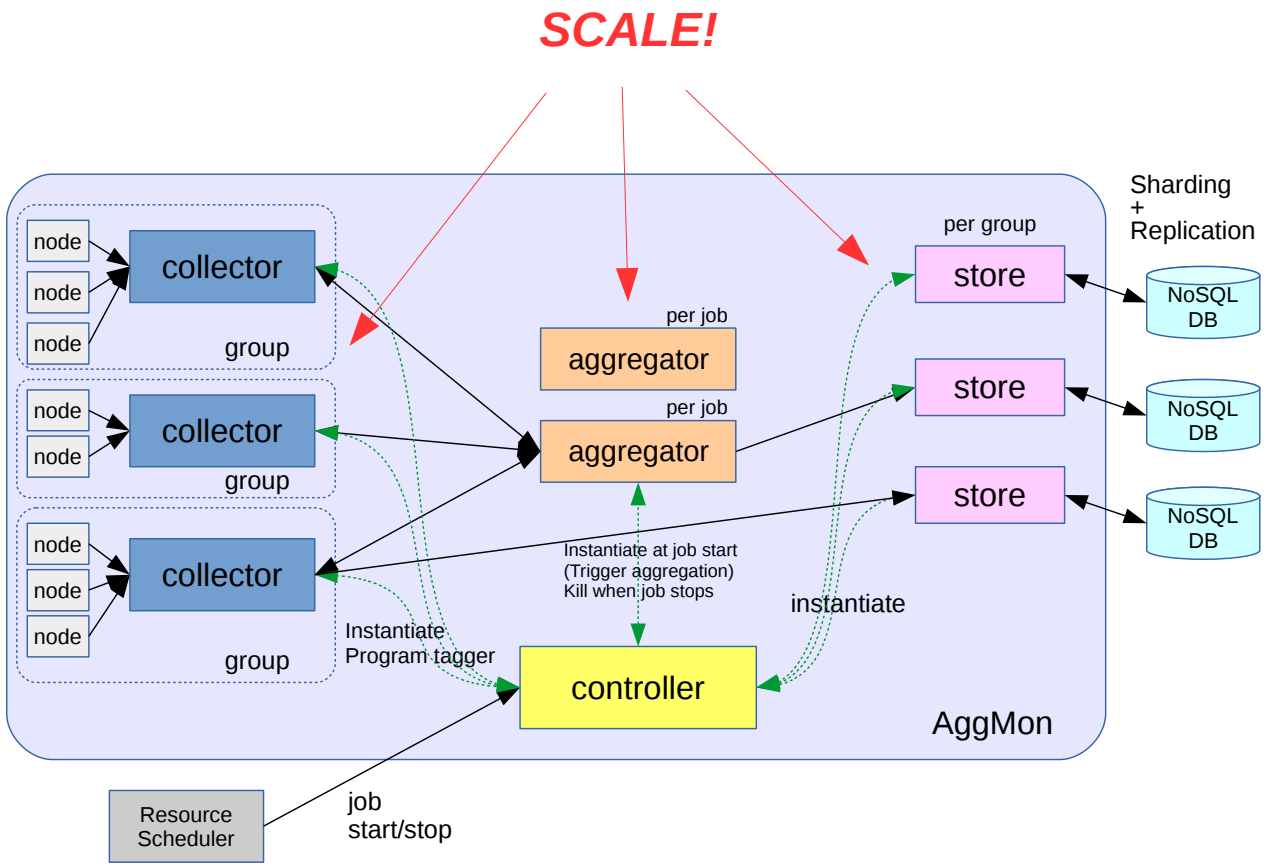
Aggmon v2: Architecture

- Components
  - Controller
  - Collector
    - Tagger
  - Aggregator
    - Statistics
      - min, max, sum, avg, percentiles
      - More? Running avgs? ...
    - Filter
    - Tagger
    - ...
  - Data Store
  - Database Backend

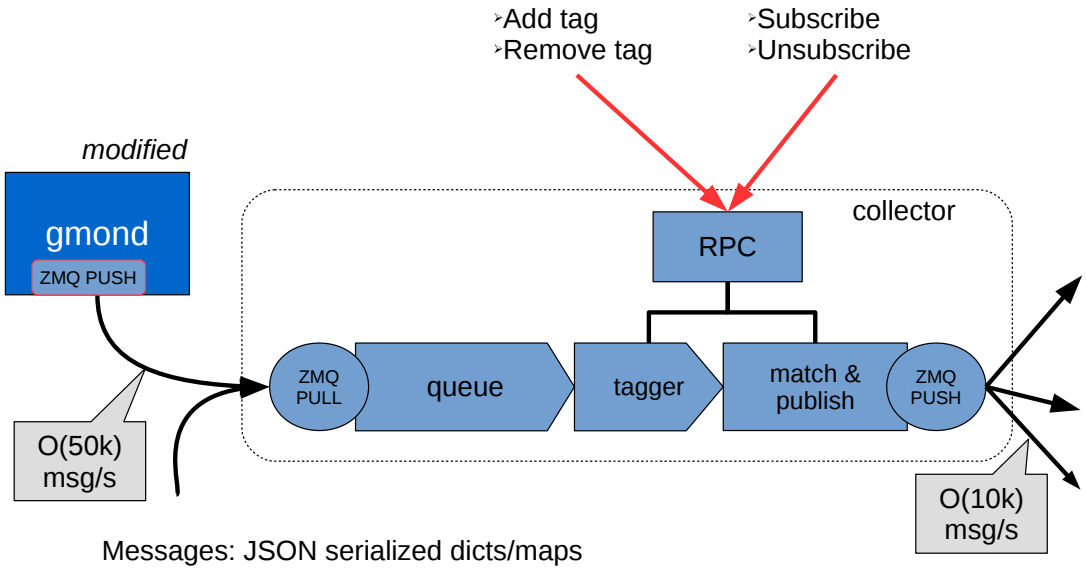


Aggmon v2: Architecture

- Components
  - Controller
  - Collector
    - Tagger
  - Aggregator
    - Statistics
      - min, max, sum, avg, percentiles
      - More? Running avgs? ...
    - Filter
    - Tagger
    - ...
  - Data Store
  - Database Backend

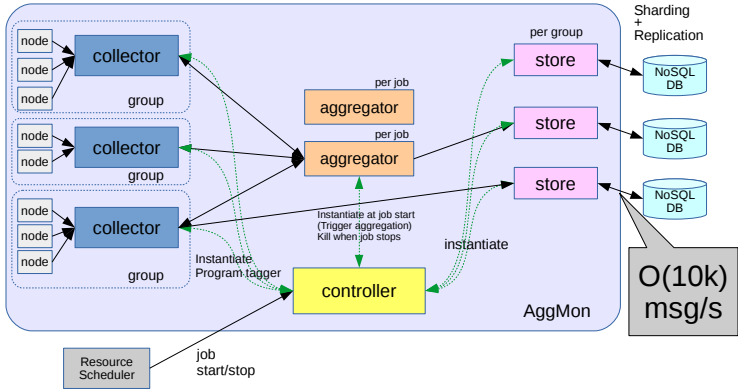


Aggmon v2: Collector



Messages: JSON serialized dicts/maps
Tagger: adds a key-value to message, based on match condition
Subscribe: based on match condition (key-value, key-value regex)

- Collector
  - Tagger
  - Matcher & Publisher
  - RPC
- Gmond
  - Integrated ZMQ metric pusher
- Diamond
  - Integrated zmq_aggmon handler



Aggmon v2: “Data Model”

- Hierarchie: bestimmt Datenfluss, verbessert evtl Übersicht!
  - Abgeleitete/aggregierte Metriken für verschiedene Hierarchiebenen!
  - Administrative Hierarchie (Gruppen, Racks, …)
  - Job Hierarchie (User, Jobs)
- Metrikformat
  - Datenerfassung
  - Im Workflow
  - In Datenbank
  - Für Visualisierung
- Parallelisierung des Datenspeichers
  - Sharding, automatisch oder manuell
  - Wo werden welche Daten gespeichert?
  - Wie indiziert man am geschicktesten?
  - Visualisierungsclient?
  - Daten extrahieren (Jobdaten)
  - Daten löschen

Aggmon v2: “Data Model”

- Hierarchie: bestimmt Datenfluss, verbes
  - Abgeleitete/aggregierte Metriken für verschiedene Hier
  - Administrative Hierarchie (Gruppen, Racks, ..)
  - Job Hierarchie (User, Jobs)
- Metrikformat
  - Datenerfassung
  - Im Workflow
  - In Datenbank
  - Für Visualisierung
- Parallelisierung des Datenspeichers
  - Sharding, automatisch oder manuell
  - Wo werden welche Daten gespei
  - Wie indiziert man am geschicktesten
  - Visualisierungsclient?
  - Daten extrahieren (Jobdaten)
  - Daten löschen

Ganglia:
Metadaten + Time Series: Attribute
Diamond:
Key + Value, Metadata encoded in Key

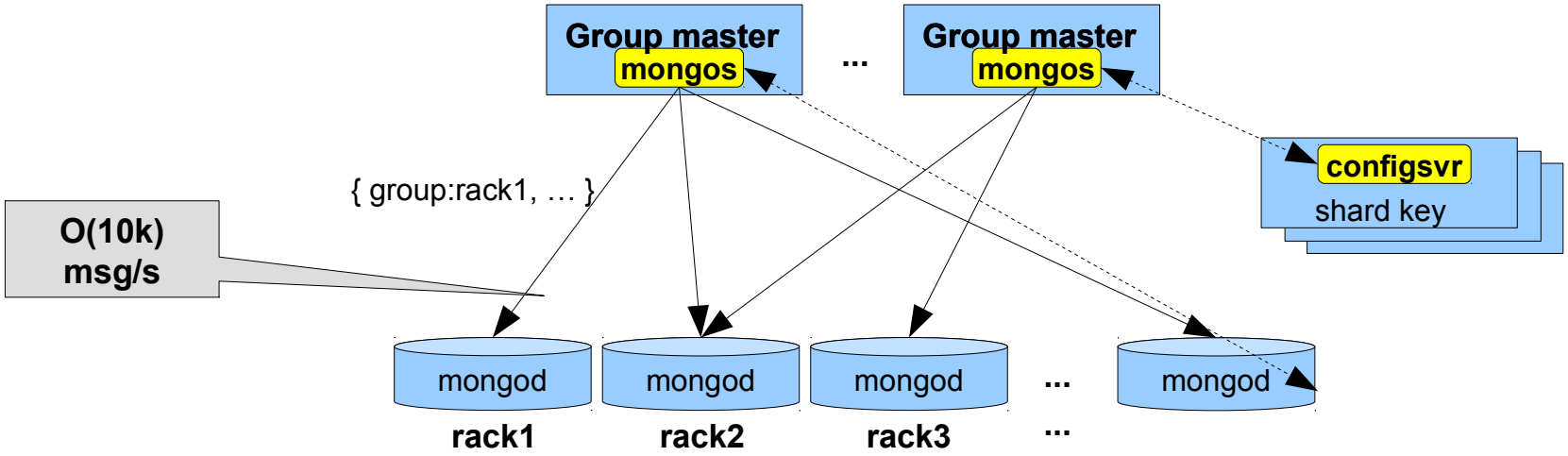
Metadata,Metrik:
Python dict, JSON encoded
Attribute, Tags
Name, Host, Timestamp, Value
Type, Origin, hpath

MongoDB, TokuMX
Metadata,Metrik:
- BSON encoded dicts
- Metadata collection
  Index: ***hpath***
- Values collection
  Indizes: ***H+N+T, J***

InfluxDB
Metrik:
- Key = flattened metadata
  eg. *servers.host.cpu.total.idle*
- Value = metric value + time

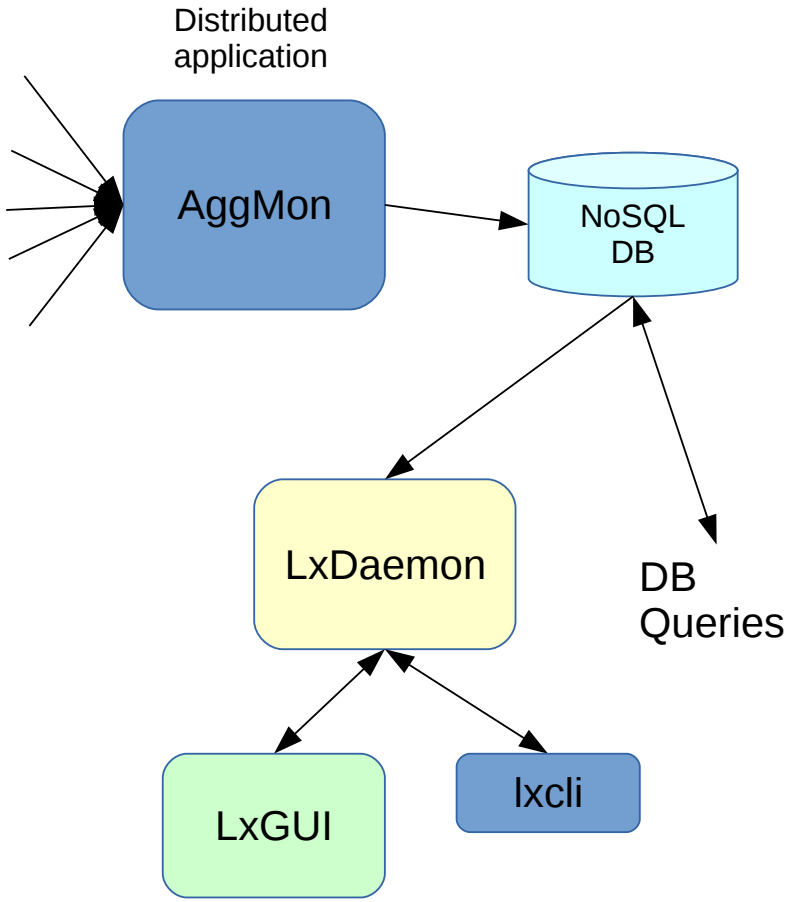
AggMon v2: Data Store

- **TokuMX**: MongoDB compatible
- Collections can be sharded
  - Spread Documents on different mongod instances
  - Shard key: group name, shard ranges define where data lives
  - Entry point: any *mongos* instance
- Replication (for example master-slave) is possible



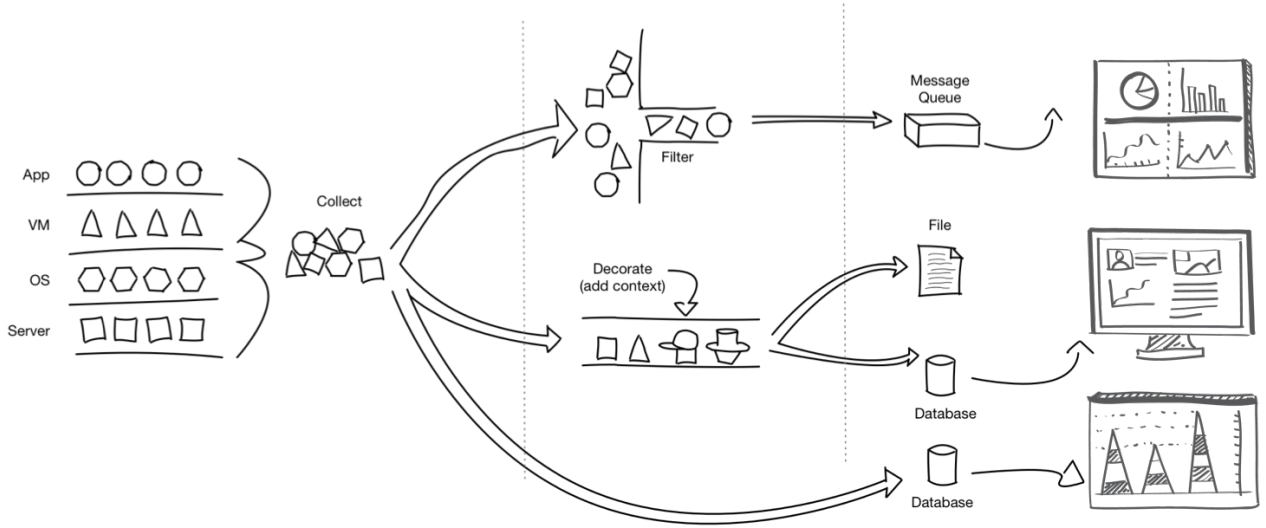
AggMon v2: Visualisierung

- Monitoring
- Data Store
- Access layer
- GUI or scripts



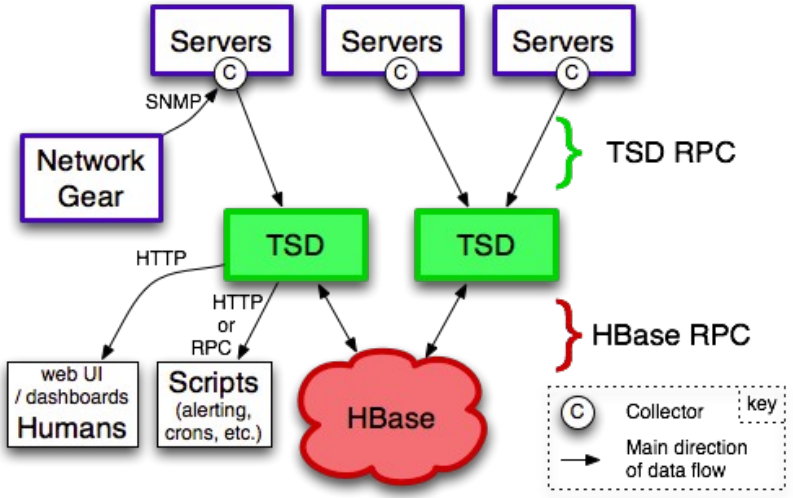
Vergleichbar: SNAP (Intel)

- Workflow manuell „programmiert”
- Fault tollerance: nicht vorhanden
- Plugins: sehr viele
- Skalierbare Datenhaltung: keine gefunden



Was noch? OpenTSDB

- Workflow: nicht wirklich
  - Externe Tools: zB collectd, statsd
  - Nicht wirklich im Konzept
- Fault tollerance: nicht vorhanden
- Parallele Datenhaltung! (Hadoop style)
- Metric format:
  - A metric name.
  - A UNIX timestamp (seconds or millisecinds since Epoch).
  - A value (64 bit integer or single-precision floating point value), a JSON formatted event or a histogram/digest.
  - A set of tags (key-value pairs) that describe the time series the point belongs to.
- Tags



- Komponenten eines Monitoring Systems sind nicht beliebig kombinierbar.
- Integration in ein Workflow ist … speziell
  - Automatisierbar?
  - Ja: zB durch Hierarchien
  - Data format?
  - Aggregatoren? Data formats?
- Verfügbarkeit: wird mit der Größe schnell sehr kompliziert
- Data Store:
  - Skalierbar? Wenige… Anforderungen genau prüfen.
  - GUI / Skripte müssen dazu passen!
- Pragmatismus ist vorteilhaft