Kwollect

Monitoring étendu d'infrastructures avec Kwollect

Simon Delamare[1] Lucas Nussbaum[2]

[1]LIP / CNRS

[2]Inria Nancy-Grand Est

Introduction

- Monitoring pour comprendre un système informatique complexe
  - Le matériel, l'application étudiée + son environnement
  - Ex: utilisation du CPU, énergie consommée, température ambiante
  - Comprendre l'effet du logiciel sur l'environnement, et inversement
- Les utilisateurs des plateformes ont besoin d'avoir accès à ces métriques
  - Parfois, mesures réalisées par les utilisateurs eux-mêmes
  - Parfois impossible (métrique non disponible depuis le système)
  - Exemple de Grid'5000/SILECS : Une plateforme pour la réalisation d'expérience à destination de la communauté système distribués, réseau, HPC, IA...
- Présentation de la solution Kwollect : Pourquoi, comment, retour d'expérience

  ... et introduire quelques technos intéressantes

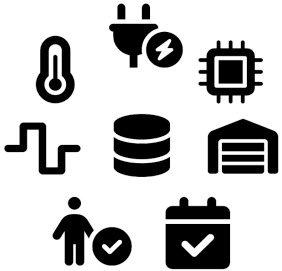
Section 1

Contexte

Motivation

Encore un outil de monitoring ? Besoins :

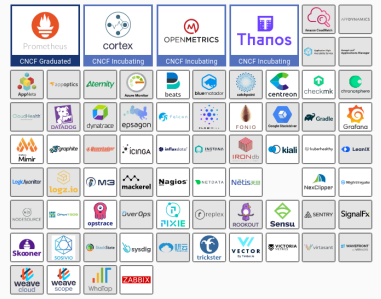
- Focus sur l'environnemental :
  - Température ambiante (via BMC type iDRAC, capteur sur PDU)
  - Consommation électrique des composants (CPU, GPU sur la BMC) ou à la prise ⇒ *SNMP, IPMI*
  - Pour Grid'5000, *Wattmetre* à 50 mesures par seconde
- Haute fréquence
- Conservation longue durée et sans perte
- Pour les utilisateurs
  - *pas uniquement pour les admin*
  - Intégré avec le *job scheduler*
  - Customisable



Outils existants

Les historiques : *Munin, Ganglia, Cacti, Collectd...*

Les modernes : *Prometheus, Influx TICK, Graphite...*



(source: landscape.cncf.io)

- Parfois liés à une solution de stockage non adaptée (RRD, Prometheus TSDB)
- Parfois uniquement « push » ou uniquement « pull »
- Souvent orientés «métriques internes aux nœuds» $\Longrightarrow$ support SNMP, IPMI, etc. en retrait
- Rarement prévu pour la haute fréquence, pour s'intégrer avec un *job scheduler*

Kwollect

Nouvel outil de monitoring des données orientées time-series

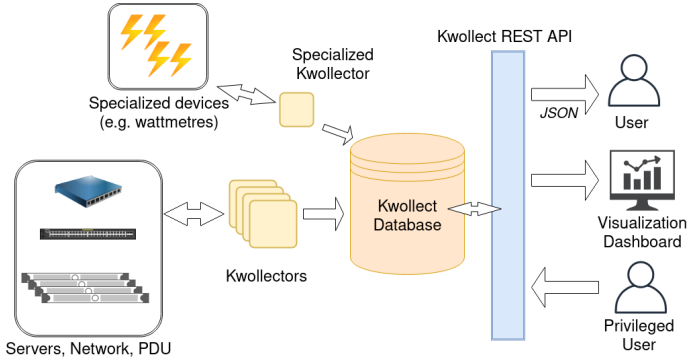
Focus: utilisateurs des infrastructures, métriques environnementales, haute fréquence

- Réutilisation de briques logicielles « sur l'étagère »
  - Peu de code maison
  - PostgreSQL au cœur

Section 2

Choix de conception

Architecture



PostgreSQL (stockage, API, logique applicative) + Kwollectors + Grafana

$\Longrightarrow$ https://gitlab.inria.fr/grid5000/kwollect

Solution pour le stockage

S'affranchir des contraintes de RRD, HDF5 $\Longrightarrow$ Une base de données !

Solution pour le stockage

S'affranchir des contraintes de RRD, HDF5 $\Longrightarrow$ Une base de données !

/!\ Besoin de haute performance lors de la manipulation de données *time-series*

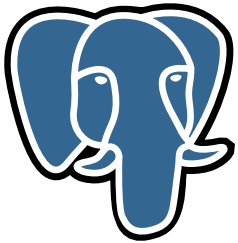
- InfluxDB, OpenTSDB, ClickHouse ?

Solution pour le stockage

S'affranchir des contraintes de RRD, HDF5 $\Longrightarrow$ Une base de données !

/!\ Besoin de haute performance lors de la manipulation de données *time-series*

- InfluxDB, OpenTSDB, ClickHouse ?



$\Longrightarrow$ PostgreSQL ! (avec l'extension TimescaleDB)

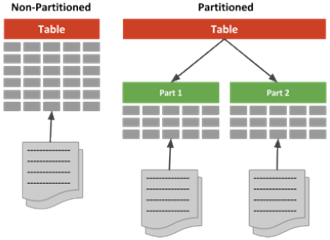
TimescaleDB

Pourquoi PostgreSQL ?

- SQL : syntaxe, cohabitation avec des données classiques
- Écosystème : clients, outils, savoir-faire, communauté, robustesse

Mais... pas optimisé pour les *time-series*

- Beaucoup de données $\Longrightarrow$ Grosses tables
  $\Longrightarrow$ Pas en RAM (les indexes) $\Longrightarrow$ Faibles perfs
- Solution : partitionnement (= on découpe les tables en sous-tables).
  Mais doit être réalisé par l'admin de la DB



(source: oracle-base.com)

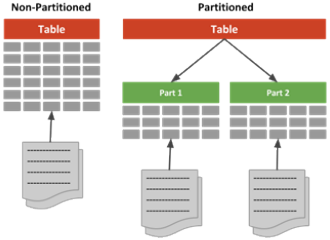
TimescaleDB

Pourquoi PostgreSQL ?

- SQL : syntaxe, cohabitation avec des données classiques
- Écosystème : clients, outils, savoir-faire, communauté, robustesse

Mais... pas optimisé pour les *time-series*

- Beaucoup de données $\Longrightarrow$ Grosses tables
  $\Longrightarrow$ Pas en RAM (les indexes) $\Longrightarrow$ Faibles perfs
- Solution : partitionnement (= on découpe les tables en sous-tables).
  Mais doit être réalisé par l'admin de la DB



(source: oracle-base.com)

TimescaleDB

- Auto partitionnement sur des périodes de temps
- Les données insérées / accédées le sont généralement sur la partition la plus «récente»
- Cette partition est suffisamment petite pour tenir en RAM $\Longrightarrow$ Performances++
- Également : compression, agrégation de données, nouvelles opérations temporelles, ...

Stockage des métriques dans la BDD PostgreSQL

Une table “metrics” qui stocke tout:

- Format des métriques ($\approx$ Prometheus) :

```
  Column   |           Type           |
-----------+--------------------------+
timestamp  | timestamp with time zone |
device_id  | text                     |
metric_id  | text                     |
value      | double precision         |
labels     | jsonb                    |
```

Des constructions SQL (vues, fonctions) pour gérer toute la «logique» de Kwollect :

- Métriques liées à **plusieurs devices**
- **API** utilisateur
- Intégration au **job scheduler**
- Activation de métriques **à la demande**

API avec PostgREST

- Service qui permet d'interagir avec une base PostgreSQL via une API Rest
  - Permet de faire l'équivalent des requêtes SQL SELECT, INSERT, etc. avec HTTP
  - Contrôle d'accès utilise les utilisateurs de la BDD
- Une API gratuite pour Kwollect !
  - requête :

```
$ curl 'http://kwollect.host:3000/rpc/get_metrics?devices=node-1,node-2&start_time=2020-01-06T13:35:00&end_time=2020-01-06T14:35:00'
```


```
{"timestamp":"2020-09-11T20:58:49.40357+02:00","device_id":"node-1","metric_id":"bmc_gpu_power_watt","value":42,"labels":{"gpu": "0"}},
{"timestamp":"2020-09-11T20:58:49.40357+02:00","device_id":"node-1","metric_id":"bmc_gpu_power_watt","value":44,"labels":{"gpu": "1"}},
{"timestamp":"2020-09-11T20:58:49.40357+02:00","device_id":"node-1","metric_id":"bmc_node_power_watt","value":648,"labels":{}},
{"timestamp":"2020-09-11T20:58:49.40357+02:00","device_id":"node-1","metric_id":"bmc_temp_ambient_celsius","value":21,"labels":{}},
{"timestamp":"2020-09-11T20:58:50.08682+02:00","device_id":"node-1","metric_id":"network_ifaceout_bytes_total","value":2654766193,"labels":
{"timestamp":"2020-09-11T20:58:50.08682+02:00","device_id":"node-1","metric_id":"network_ifacein_bytes_total","value":3883940842,"labels":{"
```


  - insertion :

```
$ curl http://kwollect.host:3000/rpc/insert_metrics \
  -H "Authorization: Bearer $TOKEN" \
  -H 'content-type: application/json' \
  -d '{"timestamp": "2020-01-06 14:00:00", "device_id": "node-1", "metric_id": "example_metric", "value": 42}'
```

Kwollector

- Kwollector : récupère des métriques et les stocke dans la BDD.
- Support de : SNMP, IPMI, exporteur Prometheus
- Python (~700 LoC), asyncio
  - Essentiellement I/O intensif

    $\Longrightarrow$ 1 seul cœur pour interroger chaque seconde 10^2 devices, avec 10^1 métriques chacun
- Accepte des fichiers de configuration du type :

```yaml
- name: idrac_power_watt
  device_id: node-1
  url: snmp://public@node-1-admin.domain.com/1.3.6.1.4.1.674.10892.5.4.600.30.1.6.1.3
  update_every: 5000
```

  (typiquement générés dynamiquement par un gestionnaire de configuration type *Ansible* ou *Puppet*)
- *kwollector-wattmetre*, pour lire et stocker les données des Wattmetre OmegaWatt

Visualisation

$\Longrightarrow$ Grafana : Facile avec PostgreSQL (bis)

Section 3

Retour d'expérience dans Grid'5000

Déploiement dans Grid'5000

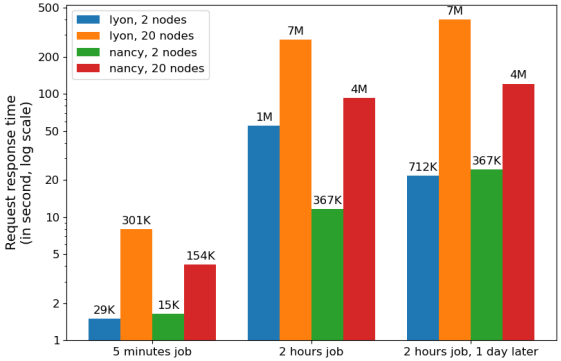
- Une instance (BDD + Kwollector + ...) déployée sur chacun de 8 sites Grid'5000
- Métriques par défaut :
  - La **consommation électrique** à la prise, chaque seconde (Wattmetre et/ou PDU)
  - Les principales **métriques des BMC** : température ambiante, consommation mesurée sur l'alim., ... (SNMP ou IPMI)
  - Le trafic sur les **équipements réseau**, chaque seconde (SNMP)
  - Les principales **métriques Prometheus** (*prometheus-node-exporter* et *dcgm-exporter* pour les GPU Nvidia)
  - Des **métriques arbitraires**, poussées par l'utilisateur depuis un nœud :

```
echo 'kwollect_custom{_metric_id="my_metric", _timestamp="1606057000"} 42' \
  >> /var/lib/prometheus/node-exporter/mymetric.prom
```

- À la demande : Mesures de consommation par les Wattmetre à 50Hz, toutes les métriques des BMC, toutes les métriques des exporteurs Prometheus, etc.

Quelques mesures de performance

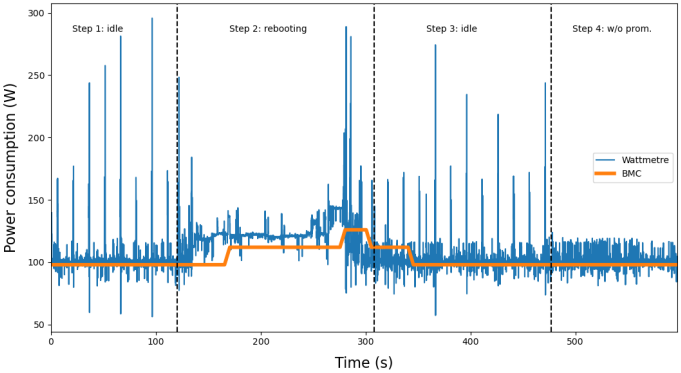
- Grid'5000@Nancy : 1350+ équipements différents monitorés, 4000 mesures/s en moyenne
- Grid'5000@Lyon : 250 équipements, 850 mesures par seconde + 20 wattmètres haute fréquence avec 50 mesures par seconde chacun, activées à la demande
- Déploiement sur des VMs «modestes» : 4 coeurs, 16Go RAM, 500Go HDD classique + SSD à Nancy



$\Longrightarrow$ $\sim 10^9$ valeurs retournées en ~1 min.

Scénario d'exemple

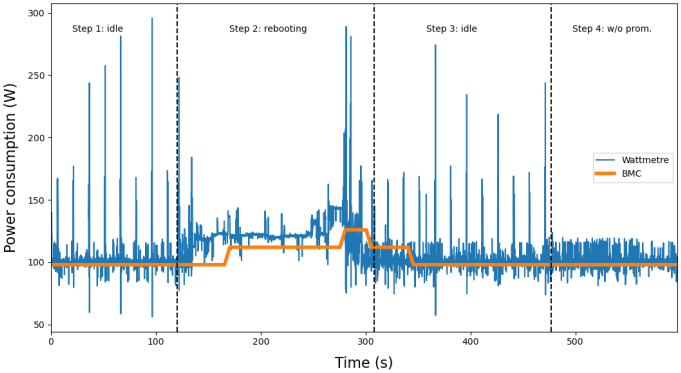
Consommation énergétique d'un noeud, mesuré sur la BMC et avec un Wattmetre



- Step 1: Au repos

Scénario d'exemple

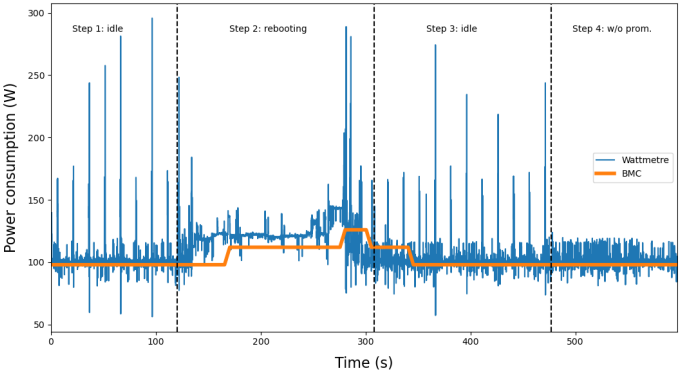
Consommation énergétique d'un noeud, mesuré sur la BMC et avec un Wattmetre



- Step 1: Au repos
- Step 2: Reboot de la machine

Scénario d'exemple

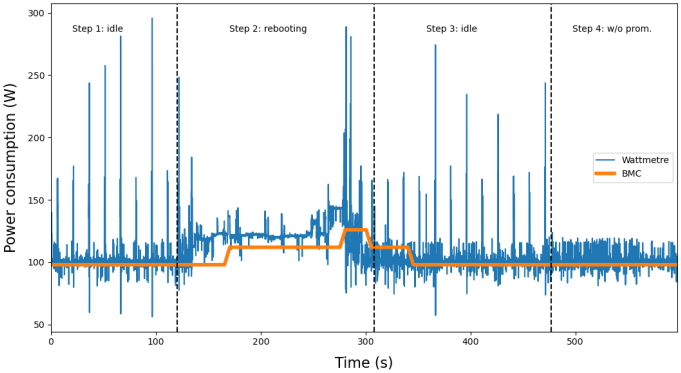
Consommation énergétique d'un noeud, mesuré sur la BMC et avec un Wattmetre



- Step 1: Au repos
- Step 2: Reboot de la machine
- Step 3: Au repos

Scénario d'exemple

Consommation énergétique d'un noeud, mesuré sur la BMC et avec un Wattmetre



- Step 1: Au repos
- Step 2: Reboot de la machine
- Step 3: Au repos
- Step 4: Arrêt de l'exporteur Prometheus

Section 4

Conclusion

Conclusion

- Kwollect : Monitoring orienté mesures environnementales et de performance, pour les infrastructures IT
  - Pour utilisateurs de Grid'5000 ou de centre de calcul $\Longrightarrow$ orienté *job*
  - Centré sur PostgreSQL, utilise aux maximum des composants logiciels existants

$\Longrightarrow$ Utile pour d'autres plateformes, dans d'autres contextes (IoT, ...)

- La suite :
  - D'avantage de métriques ! De plus en plus de capteurs...
  - Déploiement sur de nouvelles plateformes

Merci !

https://gitlab.inria.fr/grid5000/kwollect