

Dimensionnement des Pollers à l'aide de la commande `shinken-scheduler-export-data`

Sommaire

Connaitre les consommation des checks afin de pouvoir dimensionner ses Pollers quand on rajoute des hôtes
Création du tableau récapitulatif sur la consommation totale des temps CPU par royaume
Consolidation des données: utilisation d'un tableau croisé dynamique
Création du tableau croisé dynamique
Obtenir la consommation CPU totale par royaume
Sélection des royaumes en tant que lignes de notre tableau
Sélection des "Valeurs": le `cpu_time`, le temps consommé par les checks
Passer du nombres de lignes avec `"cpu_time"` à une vrai somme des temps CPU
Passer du nombre total de temps CPU consommé au nombre de CPU nécessaires
Tableau final avec le nombres de CPUs utilisés par royaume
Analyse des résultats

Description

Le module Graphite-Perfdata permet d'envoyer et stocker les métriques dans un serveur Graphite (*via Carbon*). Il est possible de modifier des paramètres via le fichier de configuration ci-dessous.

Les métriques sont associées aux UUID des éléments de Shinken. Cela permet d'éviter d'avoir à réécrire toutes les métriques suite à la modification du nom d'un hôte ou d'un check.

Pour faciliter la consultation des métriques avec des outils externes à Shinken, le module fournit une API qui permet de récupérer les relations UUID nom :

- Il peut envoyer les informations sur les éléments modifiés lors d'un chargement de configuration au serveur de métrologie.
- Il permet la récupération des relations UUID nom via un serveur qui fournit les données d'inventaire.

Pour assurer la transition vers ces mécanismes sans interruption de service, le module stocke également l'inventaire en base de données.

Cela permet d'assurer une compatibilité avec le fonctionnement historique de Shinken avec Graphite.

Cet enregistrement en base de données est voué à disparaître.

Activation du module

Le module Graphite-Perfdata est un module qui peut être activé seulement sur le démon Broker.

- L'activation du module s'effectue en ajoutant le nom de ce module dans le fichier de configuration du démon Broker.
- Pour ce faire, ouvrir le fichier de configuration du Broker à l'emplacement `/etc/shinken/brokers/nom_du_broker.cfg`, et ajouter le nom de votre module "Graphite-Perfdata".

Exemple: par défaut, nous livrons un module dont le nom est "Graphite-Perfdata":

```
define broker {
    ...
    modules          Module 1, Module 2, Module 3, Graphite-Perfdata
    ...
}
```

Pour prendre en compte le changement de configuration, redémarrer l'Arbiter:

```
service shinken-arbiter restart
```

Configuration

La configuration du module se trouve par défaut dans le fichier `/etc/shinken/modules/graphite.cfg`

- Vous trouverez aussi systématiquement un exemple dans `/etc/shinken-user-example/configuration/daemons/brokers/modules/graphite_perfdata/graphite_perfdata-example.cfg`

Exemple de fichier de configuration

```

=====
# Graphite-Perfdata
=====
# Daemons that can load this module:
# - broker (to save sla information into a mongodb database)
# This module send metrics into a graphite (carbon) server
# CFG_FORMAT_VERSION 1
=====

define module {

    # #
    #     MODULE IDENTITY     #
    # #

    # Module name [ Must be unique ]                                [ MANDATORY ]
    #
    module_name                                Graphite-Perfdata

    # Module type [ Do not edit ]                                    [ MANDATORY ]
    #
    module_type                                graphite_perfdata

    # #
    #     STORED METRICS     #
    # #

    # Restrict stored metrics
    # You can specify a list of realms whose metrics will be managed
    #
    #     Default : empty => ( Store metrics from all realms and all sub realms )
    #     ...      : list of realms => ( format is realm names, coma separated )
    #     Example : East, West
    #
    # broker_module_graphite_perfdata_realm_store_only

    # Store Warning Threshold
    #
    #     Default : Enable => 1 ( warning threshold will be stored as well as its metric value )
    #     ...      : Disable => 0 ( warning threshold will not be be stored )
    #
    # broker_module_graphite_perfdata_store_warning_threshold 1

    # Store Error Threshold
    #
    #     Default : Enable => 1 ( error threshold will be stored as well as its metric value )
    #     ...      : Disable => 0 ( error threshold will not be be stored )
    #
    # broker_module_graphite_perfdata_store_error_threshold 1

    # #
    #     NETWORK TIMEOUT     #
    # #

    # Connect Timeout
    #
    #     Default : 4 ( seconds )
    #
    # broker_module_graphite_perfdata_connect_timeout 4

    # Send Timeout
    #
    #     Default : 4 ( seconds )
    #
    # broker_module_graphite_perfdata_send_timeout 4

    # #
    #     METRIC SEND     #
    # #

    # Metrology server parameters #

    # Graphite writer ( carbon ) address
    # IP address or FQDN of carbon-cache or carbon-relay instance to send metrics to
    #
    #     Default : localhost
    #
    # broker_module_graphite_perfdata_writer_host localhost

```

```

# Graphite writer ( carbon ) port
# tcp port of carbon-cache or carbon-relay instance to send metrics to
#
#         Default : 2003
#
# broker_module_graphite_perfdata_writer_port      2003

# Resources Usage #

# Number of workers
# This module will use workers in the Broker, each worker will manage a shard of all hosts/checks.
# This parameter is used by the Broker to set the number of workers.
# Each worker will use one CPU, which will balance the metrology processing load among CPUs.
#
#         Default : 1
#
# broker_module_graphite_perfdata_writer_nb_workers 1

# Pending metrics count limit
# Maximal number of pending metrics to be sent the module keeps, before discarding new ones
# This parameter makes it possible to limit memory size needed by each worker of this module
#
#         Default : 0 => Limit is disabled ( all data is kept )
#
# broker_module_graphite_perfdata_writer_pending_nb_limit 0

# Batch mode tuning #

# Batch send interval
# Delay, in seconds, between batch send.
# In the meantime, metrics accumulate for the next batch.
# This parameter makes it possible to avoid excessive solicitations of metrology server,
# and to group metrology server writes.
#
#         Default : 10
#         ...      : 0 => Disable batch mode ( metrics are sent when module receives them )
#
# broker_module_graphite_perfdata_writer_send_interval 10

# Batch send trigger
# Number of pending metrics that triggers an immediate batch send (without waiting next interval)
# This parameter makes it possible to avoid sending too large blocks to metrology server,
# as well as consuming too much memory on the worker's module by keeping a lot of data pending.
#
#         Default : 20
#         ...      : 0 => Disable trigger
#
# broker_module_graphite_perfdata_writer_pending_nb_trigger 20

# #
#   INVENTORY      #
# #

# Inventory push #

# Activate inventory push on configuration reload
#
#         Default : Enable => 1
#         ...      : Disable => 0
#
# broker_module_graphite_perfdata_inventory_push_enable 1

# URL where inventory push is sent
#
#         Default : http://localhost/migrate
#         ...      : port can be defined in URL if needed ( http://localhost:80/migrate )
#
# broker_module_graphite_perfdata_inventory_push_url http://localhost/migrate

# Batch size
# When pushing inventory to metrology server, split data to be sent in requests sending
# only this number of elements each time
#
#         Default : 5000
#         ...      : 0 ( Push whole inventory in one request )
#
# broker_module_graphite_perfdata_inventory_push_batch_size 5000

# Inventory Server #

# Activate inventory server
#
#         Default : Enable => 1

```

```
#           ...           : Disable => 0
#
# broker__module_graphite_perfdata__inventory_server__enable 1
#
# Listen address of inventory server
#
#           Default : 127.0.0.1 => listen on loopback interface
#           ...       : 0.0.0.0 => listen on all interfaces
#           ...       : IP or FQDN => listen on this address only
#
# broker__module_graphite_perfdata__inventory_server__address 127.0.0.1
#
# Listen port of inventory server
#
#           Default : 52000
#
# broker__module_graphite_perfdata__inventory_server__port 52000
#
# Activate SSL
#
#           ...           : Enable  => 1
#           Default : Disable => 0
#
# broker__module_graphite_perfdata__inventory_server__use_ssl 0
#
# Certificate file
#
#           Default : /etc/shinken/certs/server.cert
#
# broker__module_graphite_perfdata__inventory_server__ssl_cert /etc/shinken/certs/server.cert
#
# Certificate key file
#
#           Default : /etc/shinken/certs/server.key
#
# broker__module_graphite_perfdata__inventory_server__ssl_key /etc/shinken/certs/server.key
#
# Inventory in MongoDB #
#
# Store inventory data in MongoDB
# Deprecated way for metrology server to get inventory data
#
#           Default : Enable  => 1
#           ...       : Disable => 0
#
# broker__module_graphite_perfdata__inventory_mongo__enable 1
#
# MongoDB server URL
#
#           Default : mongodb://localhost/?w=1&fsync=false
#
# broker__module_graphite_perfdata__inventory_mongo__database__uri mongodb://localhost/?w=1&fsync=false
#
# MongoDB database
#
#           Default : shinken
#
# broker__module_graphite_perfdata__inventory_mongo__database__name shinken
#
# MongoDB replica set
#
#           Default :
#
# broker__module_graphite_perfdata__inventory_mongo__database__replica_set
#
# MongoDB collection
#
#           Default : metrology_inventory
#
# broker__module_graphite_perfdata__inventory_mongo__collection metrology_inventory
#
# SSH tunneling #
#
# Secure MongoDB access with SSH tunnel
#
#           ...           : Enable  => 1
#           Default : Disable => 0
#
# broker__module_graphite_perfdata__inventory_mongo__database__use_ssh_tunnel 0
#
# SSH user to connect to
#
#           Default : shinken
#
```

```

# broker__module_graphite_perfdata__inventory_mongo__database__ssh_user shinken
# SSH key file used for authentication
#
#     Default : ~shinken/.ssh/id_rsa
#
# broker__module_graphite_perfdata__inventory_mongo__database__ssh_keyfile ~shinken/.ssh/id_rsa
#
# SSH tunnel timeout
#
#     Default : 10
#
# broker__module_graphite_perfdata__inventory_mongo__database__ssh_tunnel_timeout 10
#
# MongoDB auto reconnect #
#
# Retry number, after connection loss, before failing with an error
#
#     Default : 5
#
#
broker__module_graphite_perfdata__inventory_mongo__database__retry_connection_X_times_before_considering_an_err
or 5
#
# Delay between retries, after connection loss
#
#     Default : 5
#
# broker__module_graphite_perfdata__inventory_mongo__database__wait_X_seconds_before_reconnect 5
#
# NOTE: Change these values only if you have a MongoDB cluster and you change the
# heartbeatTimeoutSecs of your MongoDB replica set
# The value of mongodb_retention__database__wait_X_seconds_before_reconnect *
# mongodb_retention__database__retry_connection_X_times_before_considering_an_error must be
# higher than heartbeatTimeoutSecs in the rs.conf(); of your MongoDB replica set.
}

```

Détails des sections composant le fichier de configuration

Identification du module

Il est possible de définir plusieurs instances de module de type Graphite-Perfdata dans votre architecture Shinken.

- Chaque instance devra avoir un nom unique.

Nom	Type	Unité	Défaut	Commentaire
module_name	Texte	---	Graphite-Perfdata	Nous vous conseillons de choisir un nom en fonction de l'utilisation du module pour que votre configuration soit simple à maintenir. Doit être unique.
module_type	Texte	---	graphite_perfdata	Ne peut être modifié.

Métriques gérées

Nom	Type	Unité	Défaut	Commentaire
broker__module_graphite_perfdata__realm_store_only	Liste Texte	---		Par défaut, ce module sauvegarde les métriques de tout le royaume du broker et ses sous-royaumes. Vous pouvez choisir de sauvegarder dans uniquement certains royaumes. Le nom des royaumes est renseigné à la suite séparé par une virgule. Exemple: 'Realm1, Realm2, Realm3'
broker__module_graphite_perfdata__store_warning_threshold	Booléen	---	1	Permet d'activer ou désactiver la sauvegarde des seuils d'avertissements (1 pour activer, 0 pour désactiver).

broker__module_graphite_perfdata__store_error_threshold	Booléen	---	1	Permetts d'activer ou désactiver la sauvegarde des seuils critiques (1 pour activer, 0 pour désactiver).
---	---------	-----	---	--

Timeouts réseau

Nom	Type	Unité	Défaut	Commentaire
broker__module_graphite_perfdata__connect_timeout	Entier	---	4	Délai d'attente maximal avant de considérer qu'une tentative de connexion réseau a échoué
broker__module_graphite_perfdata__send_timeout	Entier	---	4	Délai d'attente maximal avant de considérer qu'un envoi de données non abouti est en erreur

Envoi des métriques

Serveur de métrologie

Nom	Type	Unité	Défaut	Commentaire
broker__module_graphite_perfdata__writer__host	IP fqdn	---	localhost	Adresse du serveur Graphite.
broker__module_graphite_perfdata__writer__port	Entier	---	2003	Port du serveur Graphite utilisé pour écrire les données.

Ressources système

Nom	Type	Unité	Défaut	Commentaire
broker__module_graphite_perfdata__writer__nb_workers	Entier	---	1	Ce module utilise des workers dans le Broker, chaque worker va gérer une part des hôtes / checks. Ce paramètre est utilisé par le Broker pour définir le nombre de workers. Chaque worker va utiliser un CPU, cela permet de répartir la charge d'écriture sur plusieurs CPU.
broker__module_graphite_perfdata__writer__pending_nb_limit	Entier		0	Limite maximale du nombre de métriques en attente d'envoi. Au delà de ce nombre, les nouvelles métriques reçues ne sont plus mise en attente et sont ignorées, ceci afin d'éviter de saturer la mémoire du module. Utiliser la valeur spéciale 0 pour une accumulation sans limite

Envoi par lot

Nom	Type	Unité	Défaut	Commentaire
broker__module_graphite_perfdata__writer__send_interval	Entier	---	10	Délai, en secondes, entre deux envois de métriques. En attendant, les métriques s'accumulent pour le prochain envoi. Ce paramètre permet d'éviter des sollicitations excessives du serveur de métrologie, et de regrouper les écritures. Utiliser la valeur spéciale 0 pour un envoi sans temporisation
broker__module_graphite_perfdata__writer__pending_nb_trigger	Entier	---	20	Nombre de métriques en attente au delà duquel on effectue un envoi, indépendamment du délai d'attente Ce paramètre permet d'éviter d'envoyer de trop gros blocs à écrire au serveur de métrologie, ainsi que de consommer trop de mémoire sur le module en gardant trop de données en attente. Utiliser la valeur spéciale 0 pour une accumulation sans limite

Inventaire (relation [UUID nom])



Si aucun outil externe n'est utilisé pour consulter les métriques (*comme Grafana par exemple*), Graphite n'aura pas besoin de l'inventaire de Shinken.

Le serveur fournissant les données d'inventaire peut alors être désactivé.

? Unknown Attachment

Ce schéma détaille les flux réseaux associés aux deux sections de configuration qui suivent.

- **En noir** le flux induit par une requête par nom aux éléments de Graphite, suivi,
 - **En vert**, par le flux vers le serveur d'inventaire, lors de la première requête, pour obtenir la table de traduction nom UUID
- **En rouge**, le flux suite à un déploiement de configuration, permettant de réinitialiser l'inventaire de Graphite

Serveur fournissant les données d'inventaire

Nom	Type	Unité	Défaut	Commentaire
<code>broker__module_graphite_perfdata__inventory_server__enable</code>	Booléen	---	1	Activer le serveur permettant de récupérer les données d'inventaire (<i>1 pour activer, 0 pour désactiver</i>).
<code>broker__module_graphite_perfdata__inventory_server__port</code>	Entier	---	52000	Port sur lequel le serveur va recevoir les requêtes
<code>broker__module_graphite_perfdata__inventory_server__address</code>	IP nom d'hôte	---	127.0.0.1	Adresse IP ou nom d'hôte sur lequel le serveur va recevoir les requêtes qui lui sont destinées. Valeurs spéciales: <ul style="list-style-type: none">• 127.0.0.1 pour écouter sur la boucle locale• 0.0.0.0 pour écouter sur toutes les interfaces réseaux de la machine

Chiffrement

Nom	Type	Unité	Défaut	Commentaire
<code>broker__module_graphite_perfdata__inventory_server__use_ssl</code>	Booléen	---	0	Chiffrer les échanges en utilisant le protocole https au lieu de http (<i>1 pour activer, 0 pour désactiver</i>)
<code>broker__module_graphite_perfdata__inventory_server__ssl_cert</code>	Texte	---	/etc/shinken/certs/server.cert	Chemin du fichier contenant le certificat.
<code>broker__module_graphite_perfdata__inventory_server__ssl_key</code>	Texte	---	/etc/shinken/certs/server.key	Chemin du fichier contenant la clé du certificat.

Envoi de l'inventaire lors d'un chargement de configuration

Nom	Type	Unité	Défaut	Commentaire
<code>broker__module_graphite_perfdata__inventory_push__enable</code>	Booléen	---	1	Activation de l'envoi des mises à jour de la configuration vers des URL Valeurs: <ul style="list-style-type: none">• 0 (<i>non</i>)• 1 (<i>oui</i>)

<code>broker__module_graphite_perfdata__inventory_push_url</code>	Liste Texte	---	http://local host /migrate	La syntaxe du paramètre est la suivante : PROTOCOLE://HOSTNAME[:PORT][BASE_URL] <ul style="list-style-type: none"> PROTOCOLE : valeurs possibles <ul style="list-style-type: none"> <code>http</code> pour des échanges non chiffrés <code>https</code> pour des échanges chiffrés HOSTNAME : nom de l'hôte ou adresse ip où envoyer les relations [UUID nom] PORT : paramètre optionnel précisant le port à contacter BASE_URL : paramètre optionnel précisant un complément d'URL à renseigner
<code>broker__module_graphite_perfdata__inventory_push_batch_size</code>	Entier		5000	Attendre d'avoir reçu ce nombre de modifications, ou la fin du chargement de la configuration pour envoyer les changements de l'inventaire Utiliser la valeur spéciale 0 pour attendre la fin du chargement de la configuration avant d'effectuer l'envoi

Stockage dans MongoDB

En plus d'envoyer les données d'inventaire au serveur Graphite, ce module peut également stocker ces données en base de données.

Cela permet au serveur Graphite d'accéder à ces informations si l'accès au serveur fournissant les données d'inventaire n'est pas encore autorisé ou configuré (voir la page [Grafana - v8.3.2](#) section "Paramètres de connexion aux serveurs d'inventaire").

Ce mode de fonctionnement est voué à disparaître, au profit d'un accès au serveur fournissant directement les données d'inventaire.



Si les serveurs de stockage des métriques ([carbon-cache](#)) sont tous configurés et autorisés à se connecter au serveur fournissant les données d'inventaire, le stockage de l'inventaire dans MongoDB peut être désactivé

Nom	Type	Défaut	Commentaire
<code>broker__module_graphite_perfdata__inventory_mongo__enable</code>	Booléen	1	Activer l'enregistrement dans MongoDB (<i>1 pour activer, 0 pour désactiver</i>).
<code>broker__module_graphite_perfdata__inventory_mongo__database__uri</code>	Texte	mongodb://localhost/?w=1&fsync=false	URL de connexion à MongoDB. Consulter la documentation de MongoDB pour la syntaxe de ce paramètre : https://docs.mongodb.com/manual/reference/connection-string/
<code>broker__module_graphite_perfdata__inventory_mongo__database__name</code>	Texte	shinken	Nom de la base de données à utiliser
<code>broker__module_graphite_perfdata__inventory_mongo__database__replica_set</code>	Texte		Nom du replica set à utiliser
<code>broker__module_graphite_perfdata__inventory_mongo__collection</code>	Texte	metrology_inventory	Nom de la collection où stocker les relations [UUID nom]

Connexion via SSH

Nom	Type	Défaut	Commentaire
<code>broker__module_graphite_perfdata__inventory_mongo__database__use_ssh_tunnel</code>	Booléen	0	Sécuriser la connexion à MongoDB en passant par un tunnel SSH (<i>1 pour activer, 0 pour désactiver</i>).

<code>broker__module_graphite_perfdata__inventory_mongo__database__use_ssh_retry_failure</code>	Entier	1	En cas d'échec lors de la création du tunnel SSH, nombre de tentatives supplémentaires effectuées
<code>broker__module_graphite_perfdata__inventory_mongo__database__ssh_user</code>	Texte	shinken	Utilisateur avec lequel établir la connexion SSH
<code>broker__module_graphite_perfdata__inventory_mongo__database__ssh_keyfile</code>	Texte	~shinken/.ssh/id_rsa	Clé SSH utilisée pour se connecter au serveur hébergeant MongoDB
<code>broker__module_graphite_perfdata__inventory_mongo__database__ssh_tunnel_timeout</code>	Entier	10	Timeout utilisé pour tester la viabilité du tunnel SSH (<i>secondes</i>)

Reprise sur panne

Nom	Type	Défaut	Commentaire
<code>broker__module_graphite_perfdata__inventory_mongo__database__retry_connection_X_times_before_considering_an_error</code>	Entier	5	Suite à une perte de connexion, nombre de tentatives pour la rétablir avant de tomber en erreur
<code>broker__module_graphite_perfdata__inventory_mongo__database__wait_X_seconds_before_reconnect</code>	Entier	5	Temporisation entre chaque tentative de reconnexion (<i>secondes</i>)

Vérification du bon fonctionnement du module

Vérification de l'état du module

Il est possible de récupérer l'état du module avec la commande suivante :

```
shinken-healthcheck
```

Voir la page suivante pour la description des messages de retour de la commande : [Shinken-healthcheck - Vérifier le bon fonctionnement de Shinken Entreprise - Chapitre - Vérification de la configuration de la sauvegarde des données de métrologie](#)

Si le module à un problème de connexion avec Graphite

Il existe une série de vérification que vous pouvez réaliser afin de tester l'accès à votre base Graphite, qui comprend :

- La vérification du processus carbon-cache,
- La vérification du port d'écoute,
- La vérification du firewall,

Voir la page suivant pour la description de comment réaliser ces vérifications : [Base de métrologie \(Graphite \) - Chapitre Vérification de carbon-cache, le demon écrivain](#)