Receiver - GLOBAL - Les logs généraux à tous les démons

Démarrage du broker

```
[XXXXXXXXXX] INFO   : [broker] Using the local log file '/var/log/shinken/brokerd.log'
[XXXXXXXXXX] INFO   : [broker] Printing stored debug messages prior to our daemonization
[XXXXXXXXXX] INFO   : [broker] System limit number of process/threads is set to maximum available: 22392
/22392
[XXXXXXXXXX] INFO   : [broker] System limit number of open files is set to maximum available: 131070/131070
[XXXXXXXXXX] INFO   : [broker] Starting HTTP daemon
[XXXXXXXXXX] INFO: [broker]
|----------------------------------------------------------------------------------------------|
[XXXXXXXXXX] INFO: [broker]                                                    broker is starting
[XXXXXXXXXX] INFO: [broker]
|----------------------------------------------------------------------------------------------|
```

Avec affichage:

- du fichier de log défini dans sa configuration (broker.ini)
- du nombres de processus/threads maximum autorisé par le système pour ce daemon
- du nombres de fichiers ouverts maximum autorisé par le système

Chargement d'une configuration

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ]
|----------------------------------------------------------------------------------------------|
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ]                                  Loading a
configuration from the arbiter
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ]
|----------------------------------------------------------------------------------------------|
```

Un tour de boucle

Début d'un tour

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ] [BROKER TIME ] [ === Loop start === ] ===-===-===-===-===-===-
===-===-===-===-===-===-===
```

Récupération des broks des schedulers et arbiters

L'arbiter envoi ses broks vers le broker:

```
[YYYY-MM-DD HH:MM:SS] INFO   : [broker-master  ] [RECEIVE BROKS] [ arbiter           ] [PERF] [ 0.000 ]s -
Add 1    broks into INTERNAL queue (new size=18) and the EXTERNAL queue (new size=18)
[YYYY-MM-DD HH:MM:SS] INFO   : [broker-master  ] [RECEIVE BROKS] [ arbiter           ]
----- 1    composed of: architecture_export_map=1
```

Le broker récupère les broks depuis un scheduler:

```
[YYYY-MM-DD HH:MM:SS] INFO   : [broker-master  ] [GET BROKS    ] [ scheduler-master   ] [PERF] [ 0.007 ]s -
Add 16   broks into INTERNAL queue (new size=16) and the EXTERNAL queue (new size=16)
[YYYY-MM-DD HH:MM:SS] INFO   : [broker-master  ] [GET BROKS    ] [ scheduler-master   ]
----- 16   composed of: host_check_result=10, host_next_schedule=6
```

Avec pour les deux cas:

- affichage du nombre de broks récupérés sur le daemon, et affichage de la taille des files d'attentes une fois rajoutés
- affichage du type de broks récupérés, ainsi que leur nombre

Envoie des broks aux modules externes

Statut des files d'envoi

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ] [MANAGE BROKS ] [ EXTERNAL MODULE ] => Number of "Broks Sets"
not eaten in MODULE queues (WebUI5-ha): 11 (WebUI3-ha): 11 (WebUI7-ha): 11 (WebUI4-ha): 11 (WebUI8-ha): 11
(WebUI1-ha): 11 (WebUI2-ha): 11 (WebUI6-ha): 11
```

A chaque tour de boucle, le broker envoie **1** ensemble de broks à chaque WebUI. 1 ensemble est composé d'autant de broks qu'il a reçus dans le tour.

Si le nombre est plus gros que 1, c'est que les WebUIs mettent du temps à digérer les ensembles.

- C'est courant au démarrage car les broks initiaux sont longs à être digéré,
- mais cela ne devrait pas arriver après.

Préparation des Broks pour l'envoi

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ] [MANAGE BROKS ] [ PREPARING BROKS    ] [PERF] [ 0.001 ]s,
preparing broks lists for INTERNAL and EXTERNAL modules
```

Chaque tour de boucle le broker préparer les listes d'envoi avec les nouveaux broks reçus.

Envoi vers les modules externes

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ] [MANAGE BROKS ] [ EXTERNAL MODULE    ] - PUSHED   [ 0.331s,
limit=5.000s ]s, EXTERNAL queue evolution: [   424 broks =>     0 broks remaining ] [   424 broks managed ]
[ Push average speed =  1928 broks/s]
```

Le broker a envoyé **424** broks en **0.331s**, et avait laissé une limite de temps de **5s** pour cet envoi (calcul basé sur la vitesse moyenne des derniers envois, ici **1928broks/s**, et une marge de sécurité).

A noter: si le nombre de broks remaining est différent de zéro, ceci signifie que le broker a reçu des broks pendant la phase d'envoi, et qu'il les enverra le prochain tour.

Envoie des broks aux modules internes (sans leur propre processus)

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ] [MANAGE BROKS ] [ INTERNAL MODULE    ] - EXECUTED [ 0.239 ]s,
INTERNAL queue evolution: [   424 broks =>   238 broks remaining ] [   424 broks managed ]
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ]                                        ----- Details of
INTERNAL modules execution time: (Graphite-Perfdata=0.071s), (Simple-log=0.010s), (sla=0.086s)
```

Le broker a fourni **424** broks aux modules internes (ceux qui n'ont pas leur propre processus), en **0.239s** au total. Ici le nombre de broks remaining est différent de zéro, ceci signifie que le broker a reçu des broks pendant la phase d'envoi, et qu'il les enverra le prochain tour.

Il fourni ensuite le détail de temps de chaque module interne.

Récupération des commandes (demande de prise en compte, demande pour relancer une vérification, etc)

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ] [MODULES      ] [ EXTERNAL COMMANDS  ] [PERF] [ 0.001 ]s Did
read 0 external commands (like recheck, set acknowledge, etc) from modules
```

Le broker récupère les commandes (comme une création de downtime, etc) et le temps que ceci lui a demandé.

Appel au modules internes chaque seconde

Chaque fin de tour, un appel est lancé vers les modules internes afin qu'ils puissent faire des actions spécifiques (par exemple vérifier un cache, vider leur éléments pas encore envoyés, etc)

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ] [MODULES      ] [ TIME IN BROKER     ] [PERF] [ 0.025 ]s All
modules "ticks" are done. Execution times by modules: (Graphite-Perfdata=0.001s), (sla=0.024s)
```

Avec:

- le temps total
- le temps passé par chaque module

Fin d'un tour

```
[YYYY-MM-DD HH:MM:SS] INFO: [broker-master  ] [BROKER TIME ] [ === Loop stop === ] [PERF] [ 0.397 ]s
```

Le broker donne le temps qu'il a passé sur ce tour de boucle. Ce dernier doit rester sous la seconde sauf pendant la phase de reception d'une nouvelle configuration où il peux dépasser ce temps.

Surcharge serveur en activité disque, ralentissant l'écriture des logs

Si le serveur hébergeant le daemon est surchargé en terme d'IO disques sur le volume qui héberge le fichier de log, alors ce dernier va mettre du temps à s'écrire et va ralentir tout le daemon. Il faut alors si c'est faisable isoler le volume des disques sur un disque moins chargé pour ne pas ralentir le daemon.

En cas de soucis vous aurez dans les lots l'entrée suivante:

```
2020-05-04 00:00:51 WARNING : [ LOGGER ]
2020-05-04 00:00:51 WARNING : [ LOGGER ]
-----------------------------------------------------------------------------------------------
2020-05-04 00:00:51 WARNING : [ LOGGER ] [ WRITING ] The log write time is very high (1.87s). Please look at
your log disk performance.
2020-05-04 00:00:51 WARNING : [ LOGGER ]
-----------------------------------------------------------------------------------------------
2020-05-04 00:00:51 WARNING : [ LOGGER ]
```

Chargement des broks initiaux par un regenerator (créateur d'objets des modules de broker) et vérifier que c'est bien la même configuration charger entre les regenerators / scheduler / arbiter

Les logs suivant permet de suivre le chargement de la configuration de supervision entre l'arbiter les schedulers jusqu'au interface : webui / livestatus / livedata

Il existe 2 types d'identifiants de configuration (représentation de la configuration)

- configuration_uuid -> uuid de configuration total généré par l'Arbiter
- configuration_part_id -> id de la partie de configuration géré par un Scheduler

Quand un module de broker avec un regenerator charge une nouvelle configuration :

[2020-05-15 16:29:49] INFO : [WebUI3] [CONFIGURATION] [NEW] [REGENERATOR] configuration part retrieved: [configuration_part_id=**configuration_part_id**, scheduler=**scheduler_name** configuration_uuid=**configuration_uuid**, arbiter=**arbiter_name** date=**creation_date**]

- **configuration_part_id** -> id de la partit de configuration géré par le Scheduler (unique par Scheduler)
- **scheduler_name** ->nom du scheduler gère cette partie de la configuration
- **configuration_uuid** -> uuid crée lors du démarrage de l'Arbiter qui correspond donc à l'id de la configuration géré par l'Arbiter
- **creation_date** -> date du démarrage de l'Arbiter
- **arbiter_name** ->nom de l'Arbiter qui a crée cette configuration

Exemple Log Broker - module WebUI3 chargement de la nouvelle configuration

```
[2020-05-15 16:29:49] INFO   : [WebUI3] [ CONFIGURATION ] [ NEW ] [ REGENERATOR ] configuration part
retrieved : [ configuration_part_id=8, scheduler=scheduler-master
configuration_uuid=fe5982b29bfb48cdadb35523799f7cec, arbiter=arbiter-master1 date=15-05-2020 16:13:40 ]
```

Quand un module de broker avec un regenerator rejet une configuration :

Dans le cas ou la configuration d'un Scheduler est déjà géré par un regenerator cas qui arrive si par exemple :
un module crash on redemande les broks initiaux donc tout les modules vont recevoir la nouvelle configuration mais ceux qui la gère déjà ne vont pas la recharger et vont log :

[2020-05-15 16:41:40] INFO : [WebUI3] [CONFIGURATION] [NEW] [REGENERATOR] No need to reload the configuration part because I already handle it [configuration_part_id=**configuration_part_id**, scheduler=**scheduler_name** configuration_uuid=**configuration_uuid**, arbiter=**arbiter_name** date=**creation_date**]

- **configuration_part_id** -> id de la partit de configuration géré par le Scheduler (unique par Scheduler)
- **scheduler_name** ->nom du scheduler gère cette partie de la configuration
- **configuration_uuid** -> uuid crée lors du démarrage de l'Arbiter qui correspond donc à l'id de la configuration géré par l'Arbiter
- **creation_date** -> date du démarrage de l'Arbiter
- **arbiter_name** ->nom de l'Arbiter qui a crée cette configuration

Exemple Log Broker - module WebUI3 chargement de la nouvelle configuration

```
[2020-05-15 16:41:40] WARNING: [WebUI3] [ CONFIGURATION ] [ NEW ] [ REGENERATOR ] No need to reload the configuration part because I already handle it [ configuration_part_id=8, scheduler=scheduler-master configuration_uuid=fe5982b29bfb48cdadb35523799f7cec, arbiter=arbiter-master1 date=15-05-2020 16:13:40 ]
```