Note d'application

Analyse des logs HAProxy avec halog

Version du document : v1.1

Dernière mise à jour : 3 septembre 2013

RULE YOUR NETWORK!®

EXCELIANCE

Objectif

Être capable d'analyser les logs générés par l'ALOHA Load-Balancer stockés dans un serveur syslog tierce.

Difficulté

Versions concernées

- ALOHA 4.2 et supérieur

Contexte

L'ALOHA Load-Balancer, utilisé en mode niveau 7, génère des lignes de logs très détaillées. Il les stocke par défaut dans un espace mémoire. Malheureusement, cet espace mémoire est limité et peut être utilisé uniquement dans le cadre d'un diagnostique instantané.
Afin d'analyser un problème qui est arrivé quelques minutes plus tôt ou pour avoir une tendance sur les temps de réponses des serveurs ou applications, ce n'est pas suffisant.
Pour analyser ces logs, HAProxy est livré avec un petit outils nommé **halog** que l'on peut installer et utiliser sur le serveur qui récupère les logs générés par l'ALOHA.

Configuration syslog de l'ALOHA Load-Balancer

Dans l'interface graphique de l'ALOHA, aller sur l'onglet **Services** puis cliquer sur l'icône d'édition  de la ligne **Trafic** du service **Syslog**.

Ajouter une ligne **server** dans la configuration, comme dans l'exemple ci-dessous :

```
server 192.168.10.26:514
```

Ensuite, cliquer sur l'icône **redémarrer**  .

L'ALOHA Load-Balancer émet deux sortes de logs :

- logs de trafic : logs de connexion HTTP et TCP, fournissant beaucoup d'informations.
  Ils sont émis en facility **local0** et en severity **info**.

– logs d'évènements : évènements survenant sur les frontends, backends, serveurs, etc...
Ils sont émis en facility **local0** et en severity **notice**.

Il serait judicieux de trier ces logs sur le système de fichier du serveur car ils pourront être utiles pour différents besoins.

Configuration serveur syslog sous Linux

syslog-ng

Syslog-ng est l'un des serveur syslog les plus puissants.
Pour **syslog-ng**, vous devez définir une source, un filtre et une destination, comme dans l'exemple ci-dessous :

```
# IP et port d'ecoute de syslog-ng
source s_net { udp(ip("192.168.10.26") port(514)); };

# ou ecrire les logs
# logs de trafic
destination d_aloha_traffic { file("/var/log/aloha/traffic.log"
        create_dirs(yes)); };
# logs d'evenements
destination d_aloha_events { file("/var/log/aloha/events.log"
        create_dirs(yes)); };

# les logs de trafic de l'ALOHA sont envoyes avec
# la facilty local0 et le niveau info
filter f_aloha_traffic { facility(local0) and level(info); };
# les logs de trafic de l'ALOHA sont envoyes avec
# la facilty local0 et le niveau notice
filter f_aloha_events { facility(local0) and level(notice); };

# ecriture des logs de trafic
log { source ( s_net ); filter(f_aloha_traffic);
        destination ( d_aloha_traffic ); };
# ecriture des logs d'evenements
log { source ( s_net ); filter(f_aloha_events);
        destination ( d_aloha_events ); };
```

rsyslog

rsyslog est l'un des serveurs syslog le plus utilisé, car installé par défaut sur les principales distributions Linux.
Pour **rsyslog**, nous devons activer l'écoute sur le réseau et router les lignes de logs de L'ALOHA.
Afin de configurer **rsyslog** pour qu'il écoute sur le réseau, décommenter les deux lignes suivantes dans le fichier **/etc/rsyslog.conf** :

```
$ModLoad imudp
$UDPServerRun 514
```

Afin de router les messages syslog vers différents fichiers, ajouter les deux lignes suivantes à la fin du fichier **/etc/rsyslog.conf** :

```
# ALOHA logs traffic with facility local0 and severity info
local0.info                     -/var/log/aloha/traffic.log
# ALOHA logs events with facility local0 and severity notice
local0.notice                   -/var/log/aloha/events.log
```

logrotate

Quand on ajoute de nouveaux fichier de logs sur un système, il est fortement conseillé de le faire tourner et de supprimer les plus anciens fichiers. C'est le rôle de **logrotate**
Créer un nouveau fichier nommé **aloha** dans le répertoire de configuration de logrotate **/etc/logrotate.d** :

```
/var/log/aloha/*.log
{
        rotate 31
        daily
        missingok
        notifempty
        delaycompress
        compress
        sharedscripts
        postrotate
                invoke-rc.d rsyslog reload > /dev/null
        endscript
}
```

Dans l'exemple ci-dessus, remplacer **rsyslog** par **syslog-ng**, en fonction du serveur syslog que vous utilisez

Installation d'HALog

HALog est un petit et très puissant outil d'analyse des logs de l'**ALOHA**.
L'installation est assez simple, comme décrit ci-dessous :

```
cd /usr/src
wget http://haproxy.1wt.eu/download/1.5/src/devel/haproxy-1.5-dev11.tar.
tar xzf haproxy-1.5-dev11.tar.gz
cd haproxy-1.5-dev11/contrib/halog
make
cp halog /usr/bin/
```

Analyse des logs ALOHA

Maintenant que nous avons HALog et les logs de l'ALOHA sur le même serveur, nous pouvons exécuter quelques analyses.

Lister les serveurs en fonction du nombre de requêtes traitées

La commande ci-dessous liste les serveurs en fonction du nombre de requêtes qu'ils ont eu à traiter. Le nom du **serveur** est préfixé par le nom du **backend**.
La huitième colonne "**tot_req**" donne le nombre de requêtes traitées par le **serveur**.

```
cat traffic.log | halog -srv -H -q |awk 'NR==1; NR > 1 {print $0 | "sort  -n -r -k 9"}' |column -t
#srv_name         1xx  2xx   3xx  4xx  5xx  other  tot_req  req_ok  pct_ok  avg_ct  avg_rt
dynamic/server1  0    3510  0    7    0    0      3517     3517    100.0   1495    1747
dynamic/server2  0    3516  0    0    0    0      3516     3516    100.0   1372    1776
```

Lister les serveurs par temps de réponse

La commande ci-dessous liste les serveurs par temps de réponse. Le nom du **serveur** est préfixé par le nom du **backend**.
Le temps de réponse est exprimé en milliseconde et la dernière colonne "**avg_rt**" donne le temps de réponse moyen pour le serveur pour toutes les URLs traitées par ce **serveur** dans ce **backend**.

```
cat traffic.log | halog -srv -H -q |awk 'NR==1; NR > 1 {print $0 | "sort  -n -r -k 12"}' |column -t
#srv_name         1xx  2xx   3xx  4xx  5xx  other  tot_req  req_ok  pct_ok  avg_ct  avg_rt
dynamic/server2  0    3516  0    0    0    0      3516     3516    100.0   1372    1776
dynamic/server1  0    3510  0    7    0    0      3517     3517    100.0   1495    1747
```

Il est recommandé de séparer le trafic statique et dynamique : vous verriez alors les temps de réponses d'une même machine pour différents types de requêtes

Lister les serveurs par le nombre d'erreur applicative : statut HTTP 5xx

La commande ci-dessous trie les serveurs par le nombre d'erreurs applicatives qu'ils ont générés. Le nom du **serveur** est préfixé par le nom du **backend**.
La sixième colonne "**5xx**" donne le nombre d'erreurs applicatives générées par le **serveur**.

```
cat traffic.log |halog -srv -H -q | awk 'NR==1; NR > 1 {print $0 | "sort  -n -r -k 6"}' | column -t
#srv_name         1xx  2xx   3xx  4xx  5xx  other  tot_req  req_ok  pct_ok  avg_ct  avg_rt
dynamic/server2  0    3516  0    0    0    0      3516     3516    100.0   1372    1776
dynamic/server1  0    3510  0    7    0    0      3517     3517    100.0   1495    1747
```

Il est recommandé d'affecter une application par backend, de cette manière, vous verriez facilement quelles applications génèrent des erreurs

Lister les serveurs par le nombre d'erreur : statut HTTP 4xx

La commande ci-dessous liste les serveurs par le nombre d'erreurs. Le nom du **serveur** est préfixé par le nom du **backend**.

```
cat traffic.log |halog -srv -H -q | awk 'NR==1; NR > 1 {print $0 | "sort  -n -r -k 5"}' | column -t
#srv_name        1xx  2xx   3xx  4xx  5xx  other  tot_req  req_ok  pct_ok  avg_ct  avg_rt
dynamic/server1  0    3510  0    7    0    0      3517     3517    100.0   1495    1747
dynamic/server2  0    3516  0    0    0    0      3516     3516    100.0   1372    1776
```

trier les URLs par temps de traitement du serveur

La commande ci-dessous permet de lister les URLs par le temps de traitement moyen, peu importe le serveur qui a traité la requête.
La sixième colonne "**okavg**" fournit le temps de traitement moyen de la réponse, en milliseconde.

```
cat traffic.log | halog -ut -H -q | column -t
#req  err  ttot     tavg  oktot    okavg  url
1004  0    6609819  6583  6609819  6583   /3s.php
2006  0    2771766  1381  2771766  1381   /health.php
2008  0    1601026  797   1601026  797    /fast.php
1004  0    1003335  999   1003335  999    /mega.php
1004  0    406830   405   406830   405    /health.html
7     0    19       2     19       2      /favicon.ico
```

Trier les URLs en erreur

La commande ci-dessous permet de trier les URLs en fonction du nombre d'erreur qu'elles ont générées, peu importe le **serveur** ou le type d'erreur.
La seconde colonne "**err**" fournie le nombre d'erreurs générées par l'URL donnée (dernière colonne).

```
cat traffic.log | halog -ue -H -q | column -t
#req  err  ttot     tavg  oktot    okavg  url
1004  0    1003335  999   1003335  999    /mega.php
2006  0    2771766  1381  2771766  1381   /health.php
1004  0    406830   405   406830   405    /health.html
7     0    19       2     19       2      /favicon.ico
2008  0    1601026  797   1601026  797    /fast.php
1004  0    6609819  6583  6609819  6583   /3s.php
```

Trier les URLs par fichier manquants : statut HTTP 404

La commande ci-dessous permet de trier les URLs en fonction du nombre d'erreur 404 qu'elles ont générées, peu importe le **serveur**.
La première colonne "**req**" fournie le nombre d'erreur 404 générée par l'URL donnée (dernière colonne).

```
cat traffic.log | halog -u -H -q -hs 404 | column -t
#req  err  ttot  tavg  oktot  okavg  url
7     0    19    2     19     2      /favicon.ico
```

Trier les URLs par nombre de requêtes

La commande ci-dessous trie les URLs par le nombre de fois qu'elles ont été appelées sur la platforme. La première colonne "**req**" fournie le nombre d'appels à l'URL.

```
cat aloha.log | halog -u -H -q | awk 'NR==1; NR > 1 {print $0 | "sort  -n -r -k 1"}' | column -t
#req  err  ttot     tavg  oktot    okavg  url
2008  0    1601026  797   1601026  797    /fast.php
2006  0    2771766  1381  2771766  1381   /health.php
1004  0    6609819  6583  6609819  6583   /3s.php
1004  0    406830   405   406830   405    /health.html
1004  0    1003335  999   1003335  999    /mega.php
7     0    19       2     19       2      /favicon.ico
```