Shinken Enterprise's distributed architecture for load balancing

The load balancing feature is very easy to obtain with Shinken.

If you use the distributed architecture for load balancing, load is typically present in 2 processes:

- pollers: they launch checks, they use a lot of CPU resources
- schedulers: they schedule, potentially lots of checks

For both, a limit of 15000 checks/5min is a reasonable goal on an average server(4 cores\@3Ghz). Scaling can be achieved horizontally by simply adding more servers and declaring them as pollers or schedulers.

Please note that the scheduler is NOT a multi-threaded process, so even if you add cores to your server, it won't change its performance.

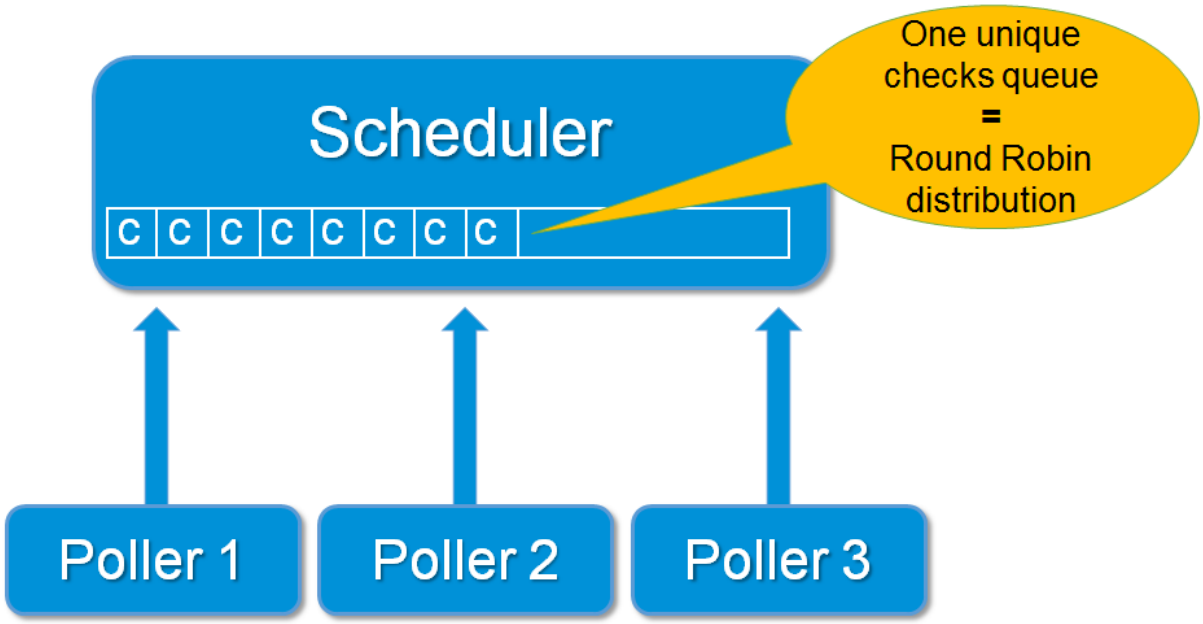
There are mainly two cases where load is a problem:

- using plugins that require lots of processing (check_esx3.pl is a good example)
- scheduling a very large number of checks

In the first case, you need to add more pollers. In the second, you need to add more schedulers. In this last case, you should also consider adding more pollers (more checks = more pollers) but that can be determined by the load observed on your poller(s).

From now, we will focus on the first case, typically installations have less than 150K checks in 5 minutes, and will only need to scale pollers, not schedulers.

How multiple pollers connect to a common scheduler



Setup a load balancing architecture with some pollers

First install the Shinken Enterprise package as usual, but in just a pollernode mode:

```
$ ./install.sh --pollernode
```

Declare the new poller on the main configuration file

Ok, now you have a brand new poller declared on your new server, server2. But server1 arbiter needs to know that it must give work to it. This is done by declaring the new poller in the etc/shinken/pollers/poller-master.cfg file.

Edit your /etc/shinken/pollers/poller-master.cfg file and define your new poller under the existing poller-1 definition (on server1):

```
#Pollers launch checks
define poller{
  poller_name poller-2
  address server2
  port 7771
}
```

Be sure to have also those lines:

```
define scheduler{
  scheduler_name scheduler-1 ; just the name
  address 192.168.0.1 ; ip or dns address of the daemon
  port 7768 ; tcp port of the daemon
}
```

Note: the address has to be 192.168.0.1 or server1 but not localhost!

When it's done, restart your Shinken Enterprise.

```
$ /etc/init.d/shinken restart
```

Check for connexion

You can look at the global shinken.log file (should be under /var/lib/shinken/shinken.log) that the new poller is started and can reach scheduler-1.
So look for lines like:

```
[All] poller satellite order: poller-2 (spare:False), poller-1 (spare:False),
[All] Trying to send configuration to poller poller-2
[All] Dispatch OK of for configuration 0 to poller poller-2
```

You can also look at the poller logs on server2.
You may have lines like that:

```
Waiting for initial configuration
[poller-2] Init de connection with scheduler-1 atHTTP://192.168.0.1:7768
[poller-2] Connexion OK with scheduler scheduler-1
I correctly loaded the modules: []
[poller-2] Allocating new fork Worker: 0
```