Scaling your installation

Planning your deployment

A monitoring system needs to meet the expected requirements. The first thing you need to do is to get your management buy-in on deploying a monitoring and data acquisition system to meet corporate goals. The second is to define the scope of the monitoring system and its particularities.

- Number of checks to supervise
- check frequency
- Method of supervising the checks Passive versus Active
- Protocol used for data acquisition (ping, SSH, NSCA, TSCA, SNMP, NRPE, NSCAweb, collectd, scripts, etc)
- Retention duration for performance data
- For each status or performance data determine if it meets the scope and goals of the project.

How Shinken Enterprise is scalable

Shinken can scale out horizontally on multiple servers or vertically with more powerful hardware. Shinken deals automatically with distributed status retention. There is also no need to use external clustering or HA solutions.

Scalability can be described through a few key metrics

- Number of hosts + checks supervised
- Number of active checks per second (type of active check having a major impact!)
- Number of checks results per second (hosts and checks combined)

And to a less extent, as performance data is not expected to overload a Graphite server instance (Which a single server can do up to 80K updates per second with millions of metrics) with a hardware RAID 10 of SSD disks.

Passive versus Active

Passive checks do not need to be scheduled by the monitoring server. Data acquisition and processing is distributed to the monitored hosts permitting lower acquisition intervals and more data points to be collected.

Active checks benefit from Shinken Enterprise's powerful availability algorithms for fault isolation and false positive elimination.

A typical large installation should make use of both types of checks.

Scaling the broker

The broker is a key component of the scalable architecture. Only a single broker can be active per scheduler. A broker can process broks (messages) from multiple schedulers. In most modern deployments, Livestatus is the broker module that provides status information to the web frontends. (Nagvis, Multisite, Thruk, etc.) or Shinken Enterprise's own WebUI module. The broker needs memory and processing power.

Dependency model

Shinken Enterprise has a great dependency resolution model. Automatic root cause isolation, at a host level, is one method that Shinken Enterprise provides. This is based on explicitly defined parent/child relationships. This means that on a check or host failure, it will automatically reschedule an immediate check of the parent(s). Once the root failure(s) are found, any children will be marked as unknown status instead of soft down.

This model is very useful in reducing false positives. What needs to be understood is that it depends on defining a dependency **tree**. A dependency tree is restricted to single scheduler. Shinken Enterprise provides a distributed architecture, that needs at least two trees for it to make sense.

Splitting trees by a logical grouping makes sense. This could be groups of checks, geographic location, network hierarchy or other. Some elements may need to be duplicated at a host level (ex. ping check) like common critical elements (core routers, datacenter routers, AD, DNS, DHCP, NTP, etc.). A typical tree will involve clients, servers, network paths and dependent checks.

Scaling the acquisition daemons

Typically pollers and Schedulers use up the most network, CPU and memory resources. Use the distributed architecture to scale horizontally on multiple commodity servers. Use at least a pair of Scheduler daemons on each server. Your dependency model should permit at least two trees, preferably 4.

Passive acquisition methods

Metrics or performance data (in a Nagios way) are embedded with check results. A check result can have zero or more performance metrics associated with it.
Theses are transparently passed off to systems outside of Shinken Enterprise using a Broker module. The Graphite broker module can easily send more than 2000 metrics per second. We have not tested the upper limit. Graphite itself can be configured to reach upper bounds of 80K metrics per second.

If a metric does not need its own check, it should be combined with a similar natured check being run on the server. checks are the expensive commodity, as they have all the intelligence like to them such as timeouts, retries, dependencies, etc. With Shinken 1.2 and fast servers, you should not exceed **60K checks*for optimum performance.

Recommended protocols for scalable passive acquisition

- Ws_Arbiter (Used by GLPI)
- NSCA (generic collection)

Log management methods

System and application logs should be gathered from servers and network devices. For this a centralized logging and analysis system is required.

Suggested centralized logging systems:

- OSSEC+Splunk
- loglogic