Presentation Artifact Descriptor
Monitoring HPC Services with CheckMK

Philip Cass
EPCC, University of Edinburgh
p.cass@epcc.ed.ac.uk

Kieran Leach
EPCC, University of Edinburgh
k.leach@epcc.ed.ac.uk

ACM Reference Format:
Philip Cass and Kieran Leach. 2019. Presentation Artifact Descriptor Monitoring HPC Services with CheckMK . In *HPCSYSPROS19 Workshop, SC19*. ACM, New York, NY, USA, 2 pages.

1 ABSTRACT

Administrative monitoring of a range of HPC systems can be time consuming and inefficient with many HPC systems being provided with their own integrated monitoring solutions and an expectation that system managers will monitor each system separately. In order to save staff time, effort and in order to improve the potential for rapid and effective response to emerging problems where systems interact, a "single pane of glass" approach is considered optimal. HPC systems typically utilise relatively boutique technology however which is commonly not monitored by existing out-of-the-box monitoring solutions.

This work details the application of `CheckMK`, a general use monitoring system, to HPC systems using non-commodity hardware and software. We focus on the development and use of "check" scripts, which at EPCC have enabled the System Administrators team to simply and reliably monitor all relevant and service-critical aspects of a variety of HPC systems through a single "pane of glass" approach.

This is the Artifact Descriptor for the artifact presented with the presented work.

2 DESCRIPTION

The artifact consists of two scripts:

(1) **`tesseract_node_check.py`** : a Python2 script that wraps for the HPE "cpower" command to provide basic status information on each compute in a HPE system. It does this by processing output from the "cpower" command and passes the data in an appropriate format to the `Panopticon CheckMK` server at EPCC. This script is run on the central management node provided with the Tesseract HPE SGI 8600 system [4].
(2) **`opa_switch_monitor.sh`** : a BASH script developed to monitor the Omnipath network on the Tesseract HPE SGI 8600 system. It uses the "opareoprt" command in order to generate monitoring information for each switch within the Tesseract Omni-Path network.

Both scripts are designed to integrate with the `CheckMK` monitoring system. In addition to the scripts, the relevant sample input and output files are also provided.

2.1 Check-list (artifact meta information)

Fill in whatever is applicable with some informal keywords and remove the rest

- **Program:** monitoring, system management, `CheckMK`, Omni-Path, custom check scripts
- **Data set:** input samples, switch list sample
- **Hardware:** HPE
- **Execution:** client node, cron, "rack leader", any node
- **Output:** output samples
- **Publicly available?:** git repository

2.2 How software can be obtained

All artifacts described here can be found in the following git repository:
https://github.com/EPCCed/hpcsystems-monitoring

2.2.1 Hardware dependencies. There are no specific hardware dependencies. This work is reproducible on any HPE system similar to Tesseract [4].

2.2.2 Software dependencies: Details of the `CheckMK` system can be found here: [2]. Details on how to set up `CheckMK` can be found in the official guide: [3].

2.2.3 Imported check details.

- RAID check: https://github.com/HeinleinSupport/check_mk_extensions/tree/master/hpsa
- Lustre performance check: http://wiki.lustre.org/Lustre_Monitoring_and_Statistics_Guide#check_mk_and_Graphite

3 EXPERIMENT WORKFLOW

The custom monitoring scripts described above are setup in one of two ways:

(1) Scripts are run directly by the `CheckMK` agent on the client node.
(2) A `cron`, or other automatic process, runs the relevant script and data is output to an appropriate spool directory that `CheckMK` reads.

Method (1) is used by the `tesseract_node_check.py` script whilst method (2) is used by the `opa_switch_monitor.sh` script.

The `tesseract_node_check.py` script should be reproducible on any similar HPE system - or in fact any system where the "cpower" command provides suitably formatted output. With `CheckMK` client installed on the management node this script can be place in the relevant directory (`/var/lib/checkmk` on our system) and discovered at the CheckMK server as described at [1]. Along with the script itself we have included a sample of the node listing maintained by the HPE software stack for dsh (/etc/dsh/group/ice-compute):

```
tesseract_node_check_sample_input
```

This is used to determine the base node list for monitoring.

This opa_switch_monitor.sh script is run on the "rack leader" for rack 1 of the system - this rack leader also acts as one of two fabric managers for the Omni-Path network. This script could however be run from any node from which the "opareport" command can interrogate the network.

This script outputs to the /var/lib/check_mk_agent/spool directory - this location may vary depending on the OS and version of CheckMK involved. This script should be simply reproducible on any system with an Omnipath network and CheckMK. A cron or other automatic mechanism should be used to run the script at an appropriate interval. Along with this script we have included a sample of the switch list used to identify which switches should be monitored:

```
opa_switch_monitor_switch.list
```

4 EVALUATION AND EXPECTED RESULT

Sample output from the tesseract_node_check.py script when run manually can be found in:

```
tesseract_node_check_sample_output
```

Note this is the output for the same 12 nodes listed in the sample input.

Sample output generated by the opa_switch_monitor.sh script can be found in:

```
opa_switch_monitor_output
```

5 NOTES

A useful outline of developing custom checks in the manner used at EPCC can be found here: [1].

REFERENCES

[1] [n. d.]. CheckMK: local checks. https://checkmk.com/cms_localchecks.html.
[2] [n. d.]. CheckMK: official website. https://checkmk.com/.
[3] [n. d.]. CheckMK: set up guide. https://checkmk.com/cms.html.
[4] [n. d.]. Tesseract system (DiRAC). https://dirac.ac.uk/resources/.