MONITORING HPC SERVICES WITH CHECKMK

Monitoring practice at EPCC

Presenter: Craig Manzi

Authors: Philip Cass, Kieran Leach

Problem statement

- EPCC manages a variety of HPC and research computing services in addition to critical support infrastructure.
- EPCC sysadmins spent a lot of time tracking the state of various systems; problem detection and diagnosis typically requires looking in multiple locations:
  - Time intensive, difficult and requires a constant wide awareness.
  - Difficult to effectively diagnose new systems where team members are typically under pressure to get things up and running in short timeframe.
- We needed a “single pane of glass” approach.

Our solution – CheckMK and locally developed checks

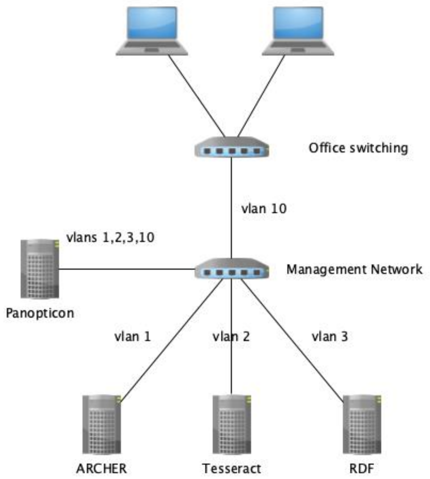
- Originally a Nagios extension, now a Nagios derivative monitoring system.
- Many checks (both Nagios and CheckMK) available already.
  - CPU, Memory, Filesystem, Interface status etc.
- Simple to create new checks
  - DDN controller, Lustre, PBS etc.
- Very simple to add new hosts, and can alter check parameters from the central user interface
- CheckMK server first installed at EPCC in 2015. Now core to our service management for HPC services.
- Since 2015 this has allowed us to provide bespoke integrated monitoring solutions for a variety of HPC technologies.

https://checkmk.com

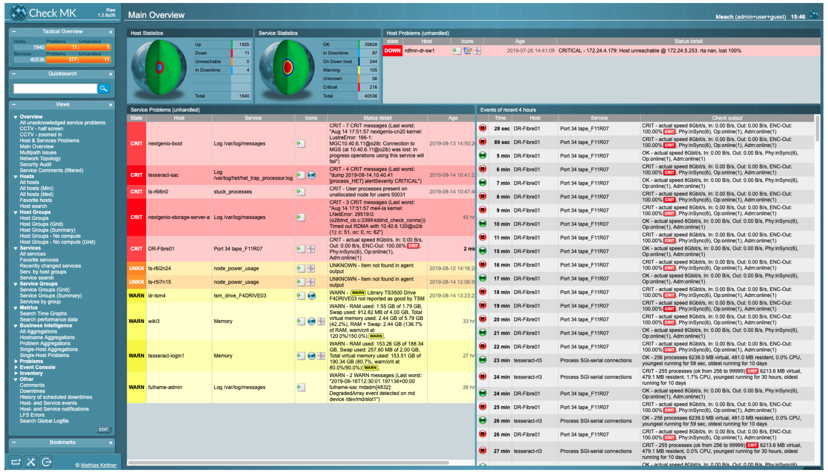
Panopticon system setup

- CheckMK server hosted on VM – named Panopticon.
- Each service on site has at least one monitoring and management VLAN on our site management network (current network design has little inter-VLAN routing).
- CheckMK VM is presented all monitoring and management VLANs and granted an IP on each.
- Panopticon presently has 18 virtual interfaces for monitoring various systems and services on site.

Panopticon networking



Panopticon user interface



Tactical Overview

Hosts	Problems	Unhandled
1940	11	1
Services	**Problems**	**Unhandled**
40536	377	11

Quicksearch

Views

- Overview
  - All unacknowledged service problems
  - CCTV - half screen
  - CCTV - zoomed in
  - Host & Services Problems
  - Main Overview
  - Multipath issues
  - Network Topology
  - Security Audit
  - Service Comments (filtered)
- Hosts
  - All hosts
  - All hosts (Mini)
  - All hosts (tiled)
  - Favorite hosts
  - Host search
- Host Groups
  - Host Groups
  - Host Groups (Grid)
  - Host Groups (Summary)
  - Host Groups - No compute
  - Host Groups - No compute (Grid)
- Services
  - All services
  - Favorite services
  - Recently changed services
  - Serv. by host groups
  - Service search
- Service Groups
  - Service Groups (Grid)
  - Service Groups (Summary)
  - Services by group
- Metrics
  - Search Time Graphs
  - Search performance data
- Business Intelligence
  - All Aggregations
  - Hostname Aggregations
  - Problem Aggregations
  - Single-Host Aggregations
  - Single-Host Problems
- Problems
- Event Console
- Inventory
- Other
  - Comments
  - Downtimes
  - History of scheduled downtimes
  - Host- and Service events
  - Host- and Service notifications
  - LFS Errors
  - Search Global Logfile

EDIT

Bookmarks

© Mathias Kettner

Host Statistics

Up	1925
Down	11
Unreachable	0
In Downtime	4
Total	1940

Service Statistics

OK	39828
In Downtime	87
On Down host	244
Warning	105
Unknown	56
Critical	216
Total	40536

Host Problems (unhandled)

state	Host	Icons	Age	Status detail
DOWN	rdfmn-dr-sw1		2019-07-26 14:41:08	CRITICAL - 172.24.4.179: Host unreachable @ 172.24.5.253. rta nan, lost 100%

Service Problems (unhandled)

State	Host	Service	Icons	Status detail	Age
CRIT	nextgenio-boot	Log /var/log/messages		CRIT - 7 CRIT messages (Last worst: "Aug 14 17:51:57 nextgenio-cn20 kernel: LustreError: 166-1: MGC10.40.6.11@o2ib: Connection to MGS (at 10.40.6.11@o2ib) was lost: in progress operations using this service will fail")	2019-08-13 14:50:2
CRIT	tesseract-sac	Log /var/log/het/het_trap_processor.log		CRIT - 4 CRIT messages (Last worst: "dump 2019-08-14.10.40.41 [process_HET] alertSeverity CRITICAL")	2019-08-14 10:41:2
CRIT	ts-r6i6n0	stuck_processes		CRIT - User processes present on unallocated node for users 50031	2019-08-14 10:47:4
CRIT	nextgenio-storage-server-a	Log /var/log/messages		CRIT - 3 CRIT messages (Last worst: "Aug 14 17:51:57 me4-la kernel: LNetError: 29519:0:(o2iblnd_cb.c:3399:kiblnd_check_conns()) Timed out RDMA with 10.40.6.120@o2ib (1): c: 51, oc: 0, rc: 62")	45 hr
CRIT	DR-Fibre01	Port 34 tape_F11R07		CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)	2 mi
UNKN	ts-r6i2n24	node_power_usage		UNKNOWN - Item not found in agent output	2019-08-12 14:16:2
UNKN	ts-r5i7n15	node_power_usage		UNKNOWN - Item not found in agent output	2019-08-14 12:06:5
WARN	dr-tsm4	tsm_drive_F4DRIVE03		WARN - WARN Library TS3500 Drive F4DRIVE03 not reported as good by TSM	2019-08-14 13:23:2
WARN	wiki3	Memory		WARN - RAM used: 1.55 GB of 1.79 GB, Swap used: 912.82 MB of 4.00 GB, Total virtual memory used: 2.44 GB of 5.79 GB (42.2%), RAM + Swap: 2.44 GB (136.7% of RAM, warn/crit at 120.0%/150.0%) WARN,	33 hr
WARN	tesseract-login1	Memory		WARN - RAM used: 153.26 GB of 188.34 GB, Swap used: 257.60 MB of 2.00 GB, Total virtual memory used: 153.51 GB of 190.34 GB (80.7%, warn/crit at 80.0%/90.0%) WARN,	27 hr
WARN	fulhame-admin	Log /var/log/messages		WARN - 2 WARN messages (Last worst: "2019-08-16T12:30:01.197138+00:00 fulhame-sac mdadm[4832]: DegradedArray event detected on md device /dev/md/slot1")	26 hr

Events of recent 4 hours

	Time	Host	Service	Check output
!!	28 sec	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	89 sec	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
OK	5 min	DR-Fibre01	Port 34 tape_F11R07	OK - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	6 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
OK	7 min	DR-Fibre01	Port 34 tape_F11R07	OK - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	8 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	9 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
OK	10 min	DR-Fibre01	Port 34 tape_F11R07	OK - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	11 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
OK	15 min	DR-Fibre01	Port 34 tape_F11R07	OK - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	16 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
OK	17 min	DR-Fibre01	Port 34 tape_F11R07	OK - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	18 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	19 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	20 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
OK	21 min	DR-Fibre01	Port 34 tape_F11R07	OK - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	22 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
OK	23 min	tesseract-rl3	Process SGI-serial connections	OK - 256 processes 6238.0 MB virtual, 481.0 MB resident, 0.0% CPU, youngest running for 58 sec, oldest running for 10 days
!!	24 min	tesseract-rl3	Process SGI-serial connections	CRIT - 255 processes (ok from 256 to 99999) CRIT 6213.6 MB virtual, 479.1 MB resident, 1.7% CPU, youngest running for 30 hours, oldest running for 10 days
OK	24 min	DR-Fibre01	Port 34 tape_F11R07	OK - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	25 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
OK	26 min	tesseract-rl3	Process SGI-serial connections	OK - 256 processes 6238.0 MB virtual, 481.0 MB resident, 0.0% CPU, youngest running for 59 sec, oldest running for 10 days
!!	26 min	DR-Fibre01	Port 34 tape_F11R07	CRIT - actual speed 8Gbit/s, In: 0.00 B/s, Out: 0.00 B/s, ENC-Out: 100.00% CRIT, Phy:inSync(6), Op:online(1), Adm:online(1)
!!	27 min	tesseract-rl3	Process SGI-serial connections	CRIT - 255 processes (ok from 256 to 99999) CRIT 6213.6 MB virtual, 479.1 MB resident, 0.0% CPU, youngest running for 30 hours, oldest running for 10 days

Day-to-day use

- Monitored during working hours by "on-shift" team member
  - Numerous issues caught this way including partial power failures, system, disk and component failures and networking issues.
- Monitorable over email out of hours
- Used as a "first port of call" for investigating issues reported by users or other team members
  - At-a-glance view of system status.

Panopticon user interface

- Delegable access: Partners and vendors can access limited views of only the items we select.
- Controllable email alerts: Email alerting can be managed and pruned on a per user basis.
  - Example: Cray staff supporting ARCHER have beepers which support email. A curated set of alerts will email the relevant address for immediate attention 24/7.
- Customisable views for different people/purposes.
  - Example: Each sysadmin office at EPCC has a screen dedicated to Panopticon using a customized view (large font, less detail, no unnecessary menus etc).

CheckMK on the Client

- CheckMK agent script runs on the node, conducts most checks when polled by the server but also runs more heavyweight checks in the background.
- Traditionally uses xinetd or systemd socket – can also configure the server to query via SSH or any custom command.
- Packaged install (rpm/deb) available.
- For both ARCHER (4920 node Cray XC30) and Tesseract (1476 node HPE SGI 8600) we have installed on all management and login nodes.
- Installed throughout the RDF (23PB DDN) storage system and within the attached Data Analytics Cluster.
- Pervasively installed on all support systems (authorisation, web, management servers etc.)

https://www.archer.ac.uk

"Out of the box" checks

- Checks available by default include:
  - CPU, Memory, disk utilisation and load.
  - IPMI checks (fans, temperatures, voltages).
  - Network interface status and statistics.
  - File system mounts.
  - Individual processes can be assigned for monitoring (e.g. pbs_mom or license servers).
  - Number of users logged in
    - We have no need to alert on this, but we installed the check anyway... and 18 months later we were asked to provide statistics for how many concurrent users we had on one of our systems
- Server can also directly query clients using e.g. SNMP
  - We host out-of-band interfaces for switches, tape libraries etc. on management vlans visible to panopticon. Not quite “one click” monitoring, but not far.
  - Can also check SSH servers, HTTP(s), DNS, LDAP etc.

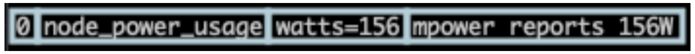
Imported checks

- Some checks we need have already been developed elsewhere
- We had a specific requirement for monitoring RAID controllers in a storage system.
- Investigation online showed that this had already been achieved and was available in a published repo.
- These checks are now in place and have been extremely useful.
- We have also used (and partially adapted) some lustre server performance checks

Custom in-house checks

- Developed in whatever language is appropriate.
  - Some bash, some python.
- Method used in checks presented here is to place correctly formatted output into /var/lib/checkmk/spool or to place the check in /var/lib/checkmk and have CheckMK run the check directly.
- Checks must output 1 service per line for monitoring.
- Service line details status, service name, metrics and status details (plaintext).

Developed checks



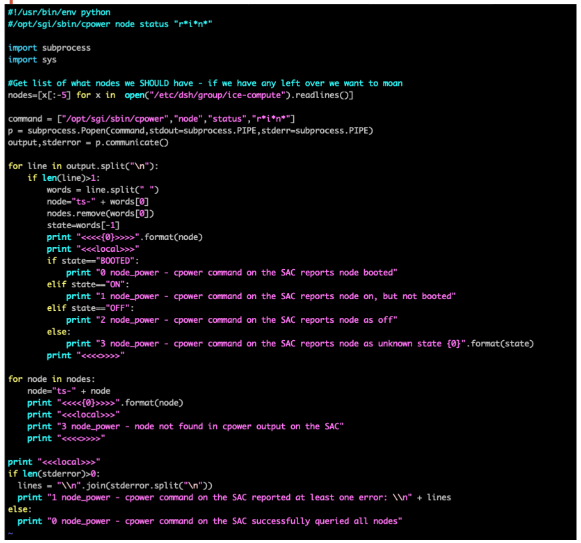
```
0 node_power_usage watts=156 mpower reports 156W
```

- **1**. Service status (0 OK, 1 WARN, 3 CRIT, 4 UNKOWN)
- **2**. Service name
- **3**. Metrics
- **4**. Status details – free text passed back with the check.

Developed checks – Tesseract node check

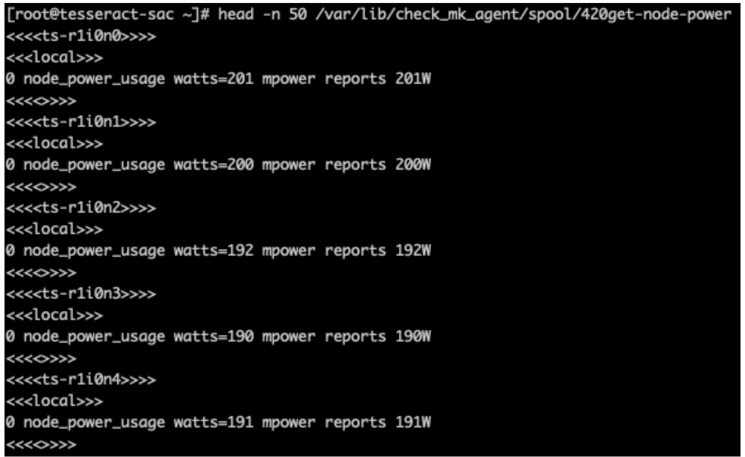
- When Tesseract arrived we had an immediate requirement for monitoring of compute node availability.
  - During handover training the “cpower” command showed how a basic health check could be put together.
  - “cpower node status r*i*n*” output was easily mangled into an appropriately formatted status report.
  - Appropriately formatted status report can then be dropped into an appropriate folder and picked up automatically.
  - Also makes use of CheckMK "piggybacking" - script run on one host but provides check output for other hosts

Developed checks – Tesseract node check



```
#!/usr/bin/env python
#/opt/sgi/sbin/cpower node status "r*i*n*"

import subprocess
import sys

#Get list of what nodes we SHOULD have - if we have any left over we want to moan
nodes=[x[:-5] for x in  open("/etc/dsh/group/ice-compute").readlines()]

command = ["/opt/sgi/sbin/cpower","node","status","r*i*n*"]
p = subprocess.Popen(command,stdout=subprocess.PIPE,stderr=subprocess.PIPE)
output,stderror = p.communicate()

for line in output.split("\n"):
    if len(line)>1:
        words = line.split(" ")
        node="ts-" + words[0]
        nodes.remove(words[0])
        state=words[-1]
        print "<<<<{0}>>>>".format(node)
        print "<<<local>>>"
        if state=="BOOTED":
            print "0 node_power - cpower command on the SAC reports node booted"
        elif state=="ON":
            print "1 node_power - cpower command on the SAC reports node on, but not booted"
        elif state=="OFF":
            print "2 node_power - cpower command on the SAC reports node as off"
        else:
            print "3 node_power - cpower command on the SAC reports node as unknown state {0}".format(state)
        print "<<<<>>>>"

for node in nodes:
    node="ts-" + node
    print "<<<<{0}>>>>".format(node)
    print "<<<local>>>"
    print "3 node_power - node not found in cpower output on the SAC"
    print "<<<<>>>>"

print "<<<local>>>"
if len(stderror)>0:
  lines = "\\n".join(stderror.split("\n"))
  print "1 node_power - cpower command on the SAC reported at least one error: \\n" + lines
else:
  print "0 node_power - cpower command on the SAC successfully queried all nodes"
~
```

Developed checks – Tesseract node check



```
[root@tesseract-sac ~]# head -n 50 /var/lib/check_mk_agent/spool/420get-node-power
<<<<ts-r1i0n0>>>>
<<<local>>>
0 node_power_usage watts=201 mpower reports 201W
<<<<>>>>
<<<<ts-r1i0n1>>>>
<<<local>>>
0 node_power_usage watts=200 mpower reports 200W
<<<<>>>>
<<<<ts-r1i0n2>>>>
<<<local>>>
0 node_power_usage watts=192 mpower reports 192W
<<<<>>>>
<<<<ts-r1i0n3>>>>
<<<local>>>
0 node_power_usage watts=190 mpower reports 190W
<<<<>>>>
<<<<ts-r1i0n4>>>>
<<<local>>>
0 node_power_usage watts=191 mpower reports 191W
<<<<>>>>
```

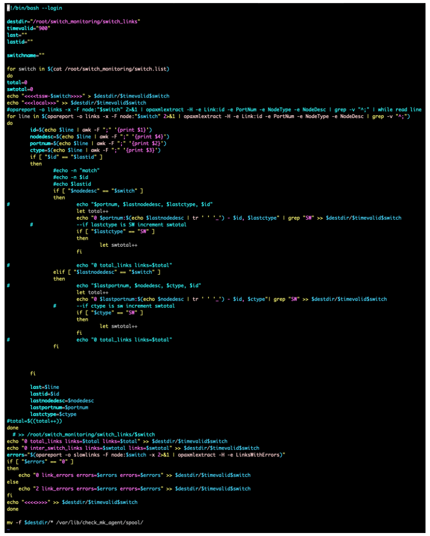
Developed checks – Tesseract Omnipatch checks

- Tesseract was our first system on site with an Omnipath network.
- Tesseract was not delivered with any particular setup for monitoring the fabric.
- Investigation showed that built-in tools (opareport) could provide a great deal of data but lacked context.

Developed checks – Tesseract Omnipath checks

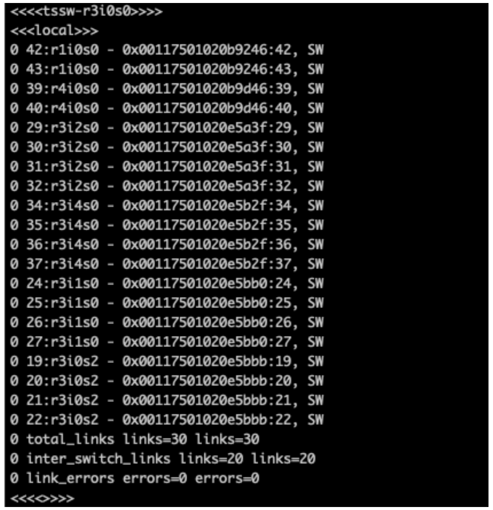
- Manually runnable checks were put together fairly quickly.
- The lack of a specification of how the fabric *should* look held things back initially.
  - Remonstrations with the vendor eventually got us this.
- Upon setting up and running a full verification of the fabric we were able to show the vendor that a number of links had been mis-cabled.
- Developed code for these checks was easily adapted into a CheckMK check.
- Bonus outcome: after implementing this we were able to show that OPA speed downgrades could occur on **neighbouring** nodes in the shared backplane whenever a blade was reconnected to the system

Developed checks – Tesseract Omnipath checks



```
!/bin/bash --login

destdir="/root/switch_monitoring/switch_links"
timevalid="900"
last=""
lastid=""

switchname=""

for switch in $(cat /root/switch_monitoring/switch.list)
do
total=0
swtotal=0
echo "<<<<tssw-$switch>>>>" > $destdir/$timevalid$switch
echo "<<<local>>>" >> $destdir/$timevalid$switch
#opareport -o links -x -F node:"$switch" 2>&1 | opaxmlextract -H -e Link:id -e PortNum -e NodeType -e NodeDesc | grep -v "^;" | while read line
for line in $(opareport -o links -x -F node:"$switch" 2>&1 | opaxmlextract -H -e Link:id -e PortNum -e NodeType -e NodeDesc | grep -v "^;")
do
        id=$(echo $line | awk -F ";" '{print $1}')
        nodedesc=$(echo $line | awk -F ";" '{print $4}')
        portnum=$(echo $line | awk -F ";" '{print $2}')
        ctype=$(echo $line | awk -F ";" '{print $3}')
        if [ "$id" == "$lastid" ]
        then
                #echo -n "match"
                #echo -n $id
                #echo $lastid
                if [ "$nodedesc" == "$switch" ]
                then
#                       echo "$portnum, $lastnodedesc, $lastctype, $id"
                        let total++
                        echo "0 $portnum:$(echo $lastnodedesc | tr ' ' '_') - $id, $lastctype" | grep "SW" >> $destdir/$timevalid$switch
        #               --if lastctype is SW increment swtotal
                        if [ "$lastctype" == "SW" ]
                        then
                                let swtotal++
                        fi

#                       echo "0 total_links links=$total"
                elif [ "$lastnodedesc" == "$switch" ]
                then
#                       echo "$lastportnum, $nodedesc, $ctype, $id"
                        let total++
                        echo "0 $lastportnum:$(echo $nodedesc | tr ' ' '_') - $id, $ctype"| grep "SW" >> $destdir/$timevalid$switch
                #       --if ctype is sw increment swtotal
                        if [ "$ctype" == "SW" ]
                        then
                                let swtotal++
                        fi
#                       echo "0 total_links links=$total"
                fi



        fi

        last=$line
        lastid=$id
        lastnodedesc=$nodedesc
        lastportnum=$portnum
        lastctype=$ctype
#total=$((total++))
done
  # >> /root/switch_monitoring/switch_links/$switch
echo "0 total_links links=$total links=$total" >> $destdir/$timevalid$switch
echo "0 inter_switch_links links=$swtotal links=$swtotal" >> $destdir/$timevalid$switch
errors="$(opareport -o slowlinks -F node:$switch -x 2>&1 | opaxmlextract -H -e LinksWithErrors)"
if [ "$errors" == "0" ]
then
    echo "0 link_errors errors=$errors errors=$errors" >> $destdir/$timevalid$switch
else
    echo "2 link_errors errors=$errors errors=$errors" >> $destdir/$timevalid$switch
fi
echo "<<<<>>>>" >> $destdir/$timevalid$switch
done

mv -f $destdir/* /var/lib/check_mk_agent/spool/
~
```

Developed checks – Tesseract Omnipath checks



```
<<<<tssw-r3i0s0>>>>
<<<local>>>
0 42:r1i0s0 - 0x00117501020b9246:42, SW
0 43:r1i0s0 - 0x00117501020b9246:43, SW
0 39:r4i0s0 - 0x00117501020b9d46:39, SW
0 40:r4i0s0 - 0x00117501020b9d46:40, SW
0 29:r3i2s0 - 0x00117501020e5a3f:29, SW
0 30:r3i2s0 - 0x00117501020e5a3f:30, SW
0 31:r3i2s0 - 0x00117501020e5a3f:31, SW
0 32:r3i2s0 - 0x00117501020e5a3f:32, SW
0 34:r3i4s0 - 0x00117501020e5b2f:34, SW
0 35:r3i4s0 - 0x00117501020e5b2f:35, SW
0 36:r3i4s0 - 0x00117501020e5b2f:36, SW
0 37:r3i4s0 - 0x00117501020e5b2f:37, SW
0 24:r3i1s0 - 0x00117501020e5bb0:24, SW
0 25:r3i1s0 - 0x00117501020e5bb0:25, SW
0 26:r3i1s0 - 0x00117501020e5bb0:26, SW
0 27:r3i1s0 - 0x00117501020e5bb0:27, SW
0 19:r3i0s2 - 0x00117501020e5bbb:19, SW
0 20:r3i0s2 - 0x00117501020e5bbb:20, SW
0 21:r3i0s2 - 0x00117501020e5bbb:21, SW
0 22:r3i0s2 - 0x00117501020e5bbb:22, SW
0 total_links links=30 links=30
0 inter_switch_links links=20 links=20
0 link_errors errors=0 errors=0
<<<<>>>>
```

Additional checks developed at EPCC

- DDN controller monitoring
- GPFS disk status
- SELinux/AppArmour status
- Vulnerability monitoring (Meltdown/Spectre etc.)
- Compute node status, power consumption, stuck processes
- Mainframe cabinet environmental status
- Unplaceable and orphan jobs
- Lustre servers read/write/metadata statistics, LNET statistics

Approach for new systems

- Initially, we concentrate on administration, service and front-end servers – these are easier to set up and we might get enough information about the compute nodes from them
- Add all applicable checks we know of – don't wait until we have a use-case (as long as they don't affect performance)
- If there's no available check for disk health, prioritise developing this (even if all it returns is a single good/bad status for the entire system)
- Any time we encounter issues, ask if it can be easily detected in the future, either by a custom check or especially using the built-in process or TCP checks

Facts & figures

- Directly query ~300 hosts
- 40,000 services
- Close to 2,000 hosts in CheckMK – the scripts mentioned previously provide "piggyback" information for compute nodes and OPA switches
- Hosted on dual-socket E7-4820 (2011, 16 cores, 2.00GHz) and close to capacity, especially if an issue results in 1,000+ events
  - Main issue is process forking - the paid version of CheckMK uses a different core which solves this problem
- Using about 4GB of RAM and 40GB of disk space; disk I/O is light

Next steps

- Our single monitoring server is reaching capacity
  - Multiple CheckMK instances can report to, and be managed by, a single central server.
- Our current version is old (1.2 vs 1.6)
  - Agent format has not changed – we should be able to upgrade the server without issues and roll out agent upgrades over time
- Custom SNMP checks for tape systems, OPA switches etc.
  - Not complicated but more work than simple local checks

Conclusions

- CheckMK centralised monitoring has been immensely useful.
- As experience has grown we have been able to rapidly deploy monitoring for new systems and to integrate site-developed monitoring for novel or boutique technologies.
- This has had demonstrable benefits for service success.
- Sharing of checks developed for HPC technologies would benefit the community.

Thanks to...

- UKRI, EPSRC, NERC, DiRAC
- HPC Systems Team at EPCC
- The HPE Support Team
- Neelofer Banglawala

Thanks for listening!