Cluster Monitoring: How can I monitor a CoolIT rack with Bright?

How can I monitor a CoolIT rack with Bright?

CoolIT is a vendor of water-cooled racks that have sensors to monitor the rack. CoolIT racks can be integrated with the Bright Cluster Manager monitoring framework as follows:

Assumptions:

The head node can access the CoolIT rack sensors directly. If the head node doesn't have direct access to the CoolIT rack sensors, then a static route should be set up through an intermediate host which has a direct connection to the sensors.

Monitoring Host:

The CoolIT rack sensors will be added as racksensor objects within Bright Cluster Manager's monitoring system. So, the health checks and metrics scripts running on the monitoring node -- the head node -- will take raw values from the rack sensors and store them in the monitoring database of the cluster manager.

Monitoring protocol:

The CoolIT rack sensors can be accessed via SNMP. The following MIB OID mappings, as verified by CoolIT support, are used to retrieve sensor data by Bright Cluster Manager:

"SNMPv2-SMI::enterprises.30518.16.2.1.2": used to retrieve sensor names

"SNMPv2-SMI::enterprises.30518.16.2.1.3": used to retrieve sensor units

"SNMPv2-SMI::enterprises.30518.16.2.1.4": used to retrieve the multipliers of the sensor values'

"SNMPv2-SMI::enterprises.30518.16.2.1.5": used to retrieve sensor real values

"SNMPv2-SMI::enterprises.30518.16.2.1.7": used to retrieve sensor minimum values

“SNMPv2-SMI::enterprises.30518.16.2.1.7”: used to retrieve sensor maximum values

Notes

* The max/min values are set to very high/very low. Simply remove the alarm from operation.

* Values for pressure, temperature and flow retrieved from the SNMP need to be divided by the multipliers (1000 for all cases).

Health checks:

The health checks have been implemented as generic scripts in python in the sense that they run on the head node given the IP address of their host, which is the IP address of the rack sensor (CMD_IP), and run the health check against this IP address.

Available health checks:

1. **CoolITMainChassisLeakDetector:** responsible for detecting any leakage in the main chassis of the rack. Assuming “0” is a “no leak” value, then the script will PASS the check if the value of the OID is zero, and otherwise the check will FAIL and an “Attention Needed” message is shown.

2. **CoolITReservoirLeakDetector:** responsible for detecting any leakage in the reservoir. Assuming “0” is a “no leak” value, the script will PASS the check if the value of the OID is zero and otherwise the check will FAIL and an “Attention Needed” message is shown.

Setup Health Checks:

1. Download the health checks from the following link:

```
# cd /cm/local/apps/cmd/scripts/

# wget -c http://support.brightcomputing.com/coolit/coolit.tar.gz
```

2. Unpack the tar.gz file under /cm/local/apps/cmd/scripts/

```
# cd /cm/local/apps/cmd/scripts/

# tar -xzvf coolit.tar.gz
```

3. Add the healthchecks to the monitoring:

```
# cmsh

% monitoring healthchecks

% add CoolITMainChassisLeakDetector

% set timeout 20

% set validfor racksensor

% set samplingmethod samplingonheadnode

% set command /cm/local/apps/cmd/scripts/coolit/healthchecks/mainLeak.py

% commit
```

4. Assign the health check for the head node or the racksensor

```
% monitoring setup healthconf racksensor

% add CoolITMainChassisLeakDetector

% commit
```

5. The previous steps should be repeated for the `CoolITReservoirLeakDetector` health check.

Metric Collection:

The metrics are collected through metric collection scripts implemented in python. The metric collection scripts are implemented as generic scripts in the sense that they run on the head node given the IP address of their host, which is the IP address of the rack sensor (CMD_IP), and run the health check against this IP address.

Cluster Monitoring: How can I monitor a CoolIT rack with Bright?

Available metrics under metric collections script:

`CoolITMetricCollection`: responsible for collecting all metrics available from SNMP output of the racksensor. The collected metrics are:

1. `CoolIT_24_V_Power`
2. `CoolIT_Ambient_Temperature`
3. `CoolIT_Dew_Point_Temp`
4. `CoolIT_Flow`
5. `CoolIT_Humidity`
6. `CoolIT_Pressure_Delta`
7. `CoolIT_Primary_Return_Temp`
8. `CoolIT_Primary_Supply_Temp`
9. `CoolIT_Proportional_Control`
10. `CoolIT_Pump1`
11. `CoolIT_Pump2`
12. `CoolIT_Reservoir_Pressure`
13. `CoolIT_Secondary_Pressure`
14. `CoolIT_Secondary_Return_Temp`
15. `CoolIT_Secondary_Supply_Temp`

Setup Metric Collection:

1. Download the metrics from the following link:

```
# wget -c http://support.brightcomputing.com/coolit/coolit.tar.gz
```

2. Unpack the tar.gz file under /cm/local/apps/cmd/scripts

```
# cd /cm/local/apps/cmd/scripts
```

```
# wget -c http://support.brightcomputing.com/coolit/coolit.tar.gz
```

3. Add the metrics to the monitoring:

```
# cmsh
```

```
% monitoring metrics
% add CoolITMetricCollection

% set timeout 20

% set validfor racksensor

% set samplingmethod samplingonheadnode

% set command /cm/local/apps/cmd/scripts/coolit/metrics/coolIT-metrics.py

% commit
```

4. Assign the health check for the head node or the racksensor

```
% monitoring setup metricconf racksensor

% add CoolITMetricCollection

% commit
```

Adding racksensors through cmsh:

```
# cmsh

% device add racksensor coolit001

% set ip 10.243.63.4

% set network internalnet

% commit
```

cmgui

A CMGUI client revision earlier than r6758 should not be used.