Redfish-Nagios: A Scalable Out-of-Band Data Center Monitoring Framework Based on Redfish Telemetry Model

**Authors:** Ghazanfar Ali⁺, Jon Hass‡, Alan Sill⁺, Elham Hojati⁺, Tommy Dang⁺, and Yong Chen⁺

⁺ Texas Tech University, Lubbock, TX, USA
‡ Dell Technologies, TX, USA
Data center infrastructure consists of **hardware** and **software** resources.

Monitoring service acquires metrics related to data center infrastructure, **often via in-band**, and store them in a database (typically time series DB).

Analytic services analyze the metrics and provide various useful insights about the applications and infrastructure resources.
In-band monitoring requires operating system to access the target service and perform monitoring functions:

- Causes the consumption of precious compute resources

- Uses ~1-2% CPU & ~17 MB memory in monitoring power metric (1-sec interval)

- Risks malfunctioning of the HW and SW components
Nagios is one of the widely used tools for data center monitoring [1]

Nagios have limitations due to its in-band-monitoring nature:
- Nagios monitoring requires monitoring-specific agents on each monitored node
- Nagios Remote Plugin Executor and the Nagios Service Check Acceptor are required on the Nagios server and each monitored node
- Manual effort is needed for the configuration of monitored nodes in the Nagios server

These shortcomings are inherently due to Nagios’ in-band implementation

To overcome these limitations, we introduce out-of-band and scalable Redfish-Nagios monitoring solution

Objectives

- To replace Nagios in-band protocols
  - Redfish-Nagios eliminates in-band protocols (i.e., NRPE, NSCA) by providing the monitoring functions through the baseboard management controller (BMC) via out-of-band (OOB) protocol

- To enable agent-less monitoring
  - Offloads monitoring processing from the on-node agent to the BMC
  - Simplifies monitoring (no requirement for development, installation, and maintenance of an agent on remotely monitored nodes)

- To automate configurations
  - Nagios requires manual effort to configure the monitored infrastructure
  - Automating the configuration process is an important capability for the monitoring of large-scale modern data centers
Out-of-band Monitoring Protocols

- **Intelligent Platform Management Interface (IPMI):**
  - Prominent initial OOB interfaces, widely adopted in HPC/cloud systems
  - Broadly used to perform remote control & acquisition of telemetry data

- **DMTF Redfish [2]:**
  - Due to notable disadvantages (i.e., security, scalability, bit-wise nature) of IPMI, Redfish standard was developed to address these concerns
  - Redfish is designed to manage and monitor data center in a secure and scalable manner
  - Redfish leverages common Internet and web service standards to expose monitoring information

- Monitoring Server initiates the acquisition of monitoring data from the monitored node.
- The in-band endpoint is responsible for implementing a particular monitoring function (in Nagios, NRPE acts as in-band endpoint and collects metrics)
Using OOB paradigm, monitoring server bypasses the node’s OS and directly communicates with the node’s BMC via OOB protocols (IPMI/Redfish).
Redfish-Nagios Architecture

Redfish-Nagios framework consists of:

- **Nagios core**, a key component of the Nagios framework, which performs check scheduling, check execution, event handling, alert, etc.

- **Redfish plugins** (abstraction, aggregation, etc.)

- **Redfish-enabled nodes** (and other devices, e.g., PDU (Power Distribution Unit), etc.)

- Nagios core communicates with Redfish plugins via an internal interface and plugins communicate with the monitored nodes via Redfish.
Integration of Redfish and Nagios

- Redfish plugins can be grouped into two types:
  - **Health monitoring** - check_host, check_CPU, check_memory and check_BMC
  - **Numeric data** - check_fans, check_temperature and check_power_usage

- Inter-working between Nagios and Redfish:
  - When the monitoring data denotes a health status of a resource, the Redfish state is directly mapped with the Nagios state
  - When the monitoring data is a numeric value, the value is translated to one of three Nagios states

---

### Redfish Plugins for Nagios [3]

<table>
<thead>
<tr>
<th>Plugin Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>check_BMC</td>
<td>Acquires BMC health</td>
</tr>
<tr>
<td>check_host</td>
<td>Acquires node health</td>
</tr>
<tr>
<td>check_CPU</td>
<td>Acquires CPU health</td>
</tr>
<tr>
<td>check_memory</td>
<td>Acquires memory health</td>
</tr>
<tr>
<td>check_fans</td>
<td>Acquires fan health &amp; speed</td>
</tr>
<tr>
<td>check_temperature</td>
<td>Acquires CPU temperature</td>
</tr>
<tr>
<td>check_power_usage</td>
<td>Acquires node power usage</td>
</tr>
</tbody>
</table>

### Inter-working Between Nagios and Redfish States

<table>
<thead>
<tr>
<th>Redfish Status</th>
<th>Nagios Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ok</td>
<td>OK</td>
<td>Working correctly</td>
</tr>
<tr>
<td>Warning</td>
<td>WARNING</td>
<td>Working, but needs attention</td>
</tr>
<tr>
<td>Critical</td>
<td>CRITICAL</td>
<td>Not working correctly or requires attention</td>
</tr>
<tr>
<td>Unknown</td>
<td>UNKNOWN</td>
<td>Plugin was unable to determine the status</td>
</tr>
</tbody>
</table>

We used the Quanah cluster with 467 Redfish-enabled nodes at High Performance Computing Center of Texas Tech as an infrastructure for implementation and deployment.

Each node is based on the Intel Xeon processor architecture and consists of 36 cores.

BMC uses the integrated Dell Remote Access Controller (iDRAC) 8, which implements the Redfish API.

The operating system of the compute nodes was Linux CentOS 7.6 (now CentOS 8.1).

The Redfish-Nagios monitoring server specs are described in the table below:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU:</td>
<td>2 x 4 cores Intel Xeon(R) E5540 @ 2.53GHz</td>
</tr>
<tr>
<td>RAM:</td>
<td>23 GB DDR3</td>
</tr>
<tr>
<td>STORAGE:</td>
<td>2TB HDD</td>
</tr>
<tr>
<td>NETWORK:</td>
<td>1Gbit/s, Broadcom NetXtreme II</td>
</tr>
</tbody>
</table>
Nagios-based monitoring requires configuration settings (e.g., IP addresses, node name) of the BMC of each monitored node. We automate this process as follows:

- First, the lists of node names and IP addresses of BMCs are acquired
- Second, a script was developed and used to generate configurations of nodes in hosts.cfg file
To make monitoring service **efficient and load balanced**, the monitoring workload is distributed among the available cores.

- **Concurrent Redfish requests** for monitoring power usage for 467 nodes in the cluster.

- E.g., seven cores handle 58 requests each and 8\textsuperscript{th} core handles remaining 61 requests.
Component Level Monitoring

- Component level monitoring enable checking node’s individual components
- Our implementation tracks seven components via Redfish and two from UGE:
  - bm_c_health, cpu_health, cpu_temperature, cpu_usage (UGE), fan_health, fan_speed, memory_health, memory_usage (UGE), system_power_usage
At node level:

- Redfish provides node status as Ok, Warning, or Critical

- Nagios shows node status as UP or DOWN

- Redfish Ok and Warning are translated as Nagios UP, and Redfish Critical is translated as Nagios DOWN
Cluster level visualization provides high-level view of the cluster in terms of nodes’ status.
The current Nagios monitoring tool is not efficient for modern data centers due to shortcomings originating from its in-band nature.

These inadequacies arise from Nagios protocols including:
- Requirement of monitoring specific in-band agents and plugins on the monitored nodes
- Consumption of computational resources of the monitored node
- Cumbersome manual configuration of the monitored nodes

We developed the Redfish-Nagios integration method, which:
- Enables Nagios to monitor HPC/cloud systems via BMC using out-of-band Redfish API
- Reduces the requirement of setting up any Nagios protocol, plugin, or agent
- Reduces compute nodes’ burden by shifting monitoring functions from the OS to the BMC
Ongoing Research and Development


https://github.com/nsfcac/MonSter
Ongoing Research and Development (cont.)
Ongoing Research and Development (cont.)

https://idatavisualizationlab.github.io/HPCC/
For more details, please check out our paper and repos:
https://github.com/nsfcac/Nagios-Redfish-API-Integration
https://github.com/nsfcac/MonSter
https://idatavisualizationlab.github.io/HPCC/

**Acknowledgment:** This research is supported in part by Dell Technologies and the National Science Foundation under grant CNS-1939140 (A U.S. National Science Foundation Industry-University Cooperative Research Center on Cloud and Autonomic Computing) and OAC-1835892. We are also very grateful to the High Performance Computing Center of Texas Tech University for providing HPC resources for this project.