

Technical Disclosure Commons

Defensive Publications Series

March 29, 2017

Using a Graph database to identify NFV domain monitors during run-time

Praneeth Nerimetla

Hewlett Packard Enterprise

Follow this and additional works at: http://www.tdcommons.org/dpubs_series

Recommended Citation

Nerimetla, Praneeth, "Using a Graph database to identify NFV domain monitors during run-time", Technical Disclosure Commons, (March 29, 2017)

http://www.tdcommons.org/dpubs_series/448



This work is licensed under a [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/).

This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.

Using a Graph database to identify NFV domain monitors during run-time

Abstract: This disclosure relates to the field of Infrastructure in virtualized performance monitoring of Virtual Network Functions' standard resources built on ETSI Guidelines. In real-time, the monitoring requirements are identified using a graph database with a faster, more efficient and monitoring identification made easier using graph data-modelling techniques.

A data modelling and identification technique is disclosed which looks into the concept of creating standard monitoring stacks for the resources such as virtual-CPU usage, virtual-memory usage, disk-usage, network-bandwidth usage or any application monitoring with minimal user inputs so as to create a good user experience. Using a graph database, with the data-modelling of each of the monitoring components which provide a method of identification of monitors enabled in the monitoring stacks in real-time.

The monitoring of the resources needs to start from a particular time or needs an initiation so as to continue the periodical monitoring. Current or previous solutions mostly deal with monitoring in a very generic note without the inclusion of NFV [Network Function Virtualization] based monitoring. Solutions existed as to fetch single or multiple values of KPIs [Key Performance Indicators] based on the monitored entity and not based on the Virtual Network Functions [VNF] or Network Services [NS]. The issue with single fetch or multiple fetch is the user must enter the inputs for identification of each of the monitored entities. In the current solution, the VNFs/NSs can be the input which the user need not specify the monitored entities' identities.

The solution design is as below –

1. The VNFs/NS can be checked for existence in the graph DB
2. If there are no VNFs/NS existing, the same message can be sent back in the response
3. In case of KPI status collection enabled for a VNF/NS apart from the user based access of a VNF.
 3. A. If the periodic KPI status collection is enabled, then it makes more sense to make a report and see a trend analysis w.r.t time or events or severity or cost and so on
 3. B. In case of a real-time view of the KPI metrics, the below process can be followed

Based on number of VNFs/NSes available for collection [which comes as part of the query], the an application engine can draw the same number of threads to -

- i) Check existence of VNFs/NSes and respond to the query accordingly. Identify the monitor-able entities in the VNF/NS from the graph DB and add their IDs and breach conditions to corresponding hash-maps/concurrent-Hashmap
- ii) Draw parallel threads [within or equal to a predefined configurable amount] to fetch KPI details for each of the monitor-able entities
- iii) Set/Update the hash-maps post null checks, datatype checks, round-off, etc.
- iv) Wait for all threads to complete
- v) Check for the breach conditions of each monitor and set any status, KPI user readable name, KPI units as required for the monitor-able object
- vi) Send data to the requested source [for that batch]

Sample Cypher Query in graph database: `START object=node(*) match object -[r*]->monitor where has(object.Id) and object.Id in ['VNF-ID','NS- ID','VNFC-ID'] and monitor.ObjectType='MONITOR' and monitor.Status='ENABLED' return distinct monitor`

Exceptions like finite breach conditions, embedded monitorable objects [e.g.: VNFC having more VNFCs].

Multiple breach conditions for the same condition type and values of null are not in the scope of this disclosure

Order of the results of the KPI fetch can be in decreasing order of the breach condition, with group by condition types or order by highest or latest state of error with value propagated in the VNF grouped by each VNF with/without user permissions

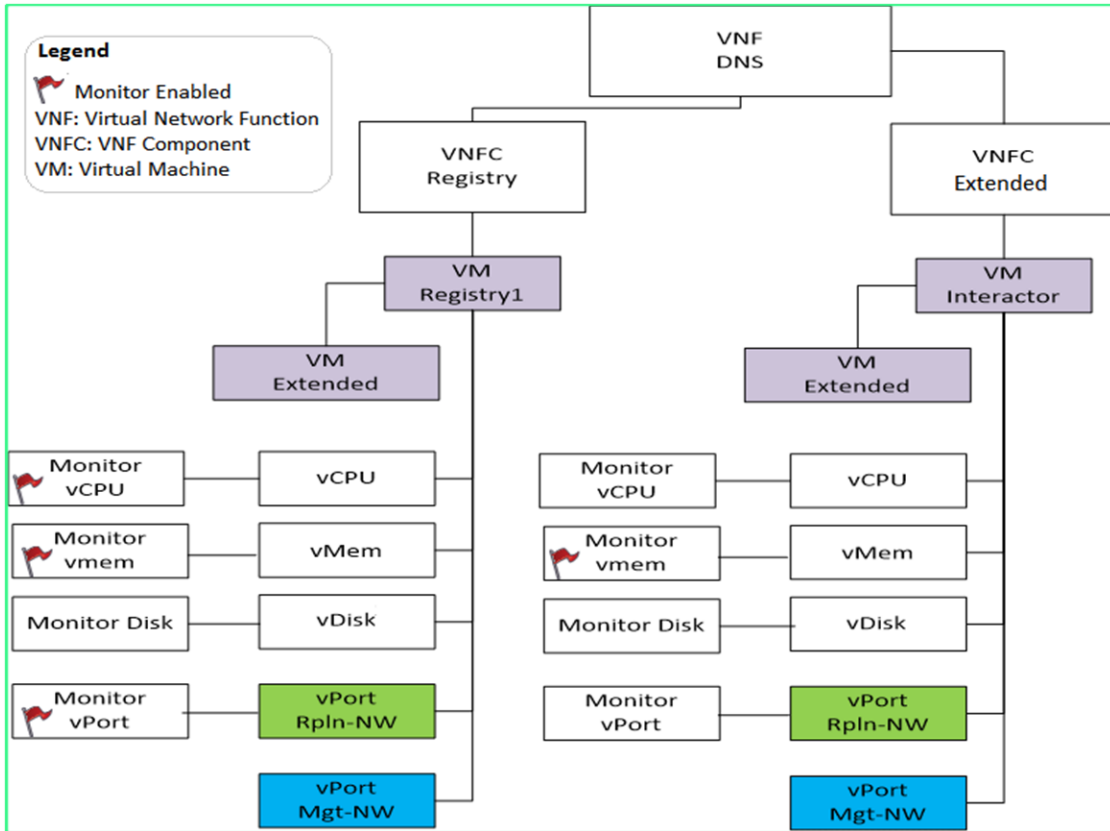
Further query options can be for -

1. for one VNF/NS
2. Multiple VNFs/NS
3. Combination of VNFs and NS
4. Combination of above with specific monitor type [e.g.: say CPU or memory types, etc.]
5. VNF/NS with priority which need quick manual remedy or a prevention plan
6. VNF/NS KPIs sorted by time [earliest on top]
7. VNF/NS KPIs breached with grouping by remediation actions and order by timestamps
8. VNF/NS KPI's history from and to dates for plotting of graphs [can be line/graph/pie, etc.]
9. Purge KPI details - real-time and scheduled


Below advantages can be harnessed once put to use –

Certain types [say just CPU usage or memory usages] or resource usage of the monitored entities -

1. KPI data obtained parallely which results in faster processing and lesser time
2. Conditions check for deriving the KPIs which provide status can also be included as part of the solution
3. In case of a large number of KPI details to be fetched, a default pagination feature to query the graph database can be used
4. User need not specify any identification of the monitored entity automatic identification of the monitored entities. Or providing the identification of the monitored entity becomes optional
5. Sorting the KPIs in a certain order opens up a new dimension in analytics for actions taken for the KPI breach and for real-time display reports,
6. Better inputs for infrastructure and virtualized resources capacity planning.
7. This can be further extended to per server/enclosure/rack based monitoring and better infrastructure management based on the KPI data per object over a period of time
8. Automatic scheduling of KPI status collection for the select VNFs/NSes can be reported
9. Control over specific VNFs for the specified users
10. Licensing based on usage is also possible using the framework



From the above block diagram -

1. A VNF/NS Can have multiple VNFCs as the logical grouping of Virtual machines and resource monitors for each of the VM's resources or on a selective basis[resource monitors imply: vCPU/vMem/vDisk/vPort]
2. Each monitor object shares a data-model of – ObjectType - MONITOR, Flag which is 'ENABLED' [represented using flag - 

http://www.tdcommons.org/dpubs_series/448

4. The above model/design can be extended to several/any of VNFs each with again multiple VNFCs/VMs and their resources.

The disclosure focusses on having a standard model for a monitor per resource and just having an enablement/disable flag [stored as data in the graph database which would be later discovered again based just on the VNF/NS/VNFC identifications during run-time as that can be modified] so as to have automatic monitoring once the VNF is activated.

Disclosed by Praneeth Nerimetla, Hewlett Packard Enterprise