Technical white paper

Using R-Functions to Integrate Custom Analytics into OpsA

Operations Analytics 2.10

May 2014

Table of Contents

Purpose

The purpose of this white paper is to document the steps that custom analytics developers can take to register custom analytics written using R and make use of them on data being collected by Operations Analytics (OpsA). Using OpsA 2.10, OpsA users can execute R functions on top of an Analytics Query Language (AQL) function that fetches entities and some measurements done for them based on data collected by OpsA.

Setting up the R Language Pack from Vertica

OpsA uses Vertica's R language runtime environment as the runtime environment for any R function you register with both OpsA and Vertica. It is mandatory that you have the Vertica R language pack set up on each node of the Vertica cluster used by your OpsA deployment. You must install the following packages on each node of the Vertica cluster in order to set up the Vertica R language pack:

compat-libgfortran-41-4.1.2-39.el6.x86_64.rpm

vertica-R-lang-6.1.1-0.x86_64.RHEL5.rpm

The former is a pre-requisite for the latter.

Complete the instructions shown in the *Use an Existing Vertica Installation and use the R Language Pack from Vertica* section of the *Operations Analytics Installation Guide*.

Creating the R Functions that Integrate with OpsA

OpsA expects all R functions to conform to the Vertica R UDX framework (R UDX). In order to have a valid R UDX, Vertica expects the following:

1. R functions must have a corresponding UDX factory function written in R. This function must capture input, output frame descriptions, and descriptions of optional input parameters to the core R function.
2. If an R function's output frame does not contain a fixed number of columns with fixed types, then the factory function needs to specify an output type callback R function that is written by the user. The output callback function describes the output frame structure to Vertica at runtime.
3. If an R function expects parameters for configuring the behavior of the computation done in the core R function in question, then those parameters need to be described using a parameter callback R function and the factory function must specify the same parameters.
4. It is expected that the core R function, the UDX factory R function, any optional output callback R function and any optional parameter callback R function must all be present in a single `.R` file that is used for registering the R function as a valid Vertica R UDX.

Identifying the Statistical Random Variables in an Input Frame for an R function

An R function's integration with OpsA currently assumes that the R function is written so that it identifies the statistical random variables in the OpsA domain from the input data frame that is fed to the R function at runtime. The following information helps you better understand the concept of OpsA random variables and how to write R code to identify these variables in the input frames.

As noted earlier, one can use OpsA dashboard panes to invoke R functions on an AQL function that results in OpsA timeseries data.

At runtime, an AQL function is translated to a Vertica SQL. In addition, OpsA's AQL wraps the Vertica SQL inside of another Vertica SQL involving the registered R UDX invocation.

The Vertica SQL translated from the AQL function represents the query that Vertica would run to supply the results as an input data frame to the R function.

AQL also supplies three internal parameters to an R function call in the outer SQL involving the invocation of the R function: `numentityparts`, `entitypart1columnindex`, and `timestampcolumnindex`. These parameters should permit R function writers to identify the entities, measurement names combinations, and their corresponding timeseries data. Once entity, measurements, and their corresponding timeseries data are identified, each entity, measurement combination could be considered a valid random variable backed by the timeseries data for itself being the observations for the random variable.

OpsA provides some example .R files containing core R functions, their UDX factory R functions, output callback functions, and parameter callback functions, in the following location: *`/opt/HP/opsa/inventory/lib/hp/r-udx-examples`*.

See the example named `MVCorr.R` that attempts to do statistical correlation between pairs of such random variables.
Note: You must provide a `parametertypecallback` R function in your .R file in order to achieve smooth passing of these parameters to the R function at runtime. The `parametertypecallback` R function is recognized by Vertica UDX framework as an optional function that allows specifying the description of parameters to UDX framework. For integration with OpsA, this callback is mandatory.

`The` following snippet from MVCorr.R example, demonstrates the boiler plate code that needs to be written for the `parametertypecallback` R function. This snippet also demonstrate how to specify the `parametertypecallback` function (in this example the function named *mvCorrParamType*) in the UDX factory function (in this example its function named *`mvCorrFactory`*).

```
mvCorrParamType<-function(){

        params <- data.frame(datatype=rep(NA, 3), length=rep(NA,3),
scale=rep(NA,3),name=rep(NA,3) )


        params[1,1] = "int"

        params[1,4] = "numentityparts"


        params[2,1] = "int"

        params[2,4] = "entitypart1columnindex"


        params[3,1] = "int"

        params[3,4] = "timestampcolumnindex"

        params

}


mvCorrFactory<-function()

{

        list(name=mvCorr,udxtype=c("transform"),intype=c("any"),
outtype=c("any"),
outtypecallback=mvCorrOutType,parametertypecallback=mvCorrParamType)

}
```

The various other helper methods in MVCorr.R also illustrates one could write code to identify the random variables in the input frame based on the parameters passed by AQL.

Take a look at the optional Vertica R UDX framework recognized `outtypecallback` R function that attempts to establish the contract with Vertica for the outgoing result or output frame columns. If you want the frame columns output by specific names or want to specify specialized types for some of these columns, then you must code the `outtypecallback` R function and register the same in UDX factory R function.

```
mvCorrOutType<-function(x,y){
    entityStartIndex<-y[['entitypart1columnindex']]
    numEntityParts<-y[['numentityparts']]


    ret <-
data.frame(datatype=rep(NA,(numEntityParts*2+2+1)),lenth=rep(NA,(numEntityParts*2+2+1)),scale=rep(NA,(n
umEntityParts*2+2+1)),name=rep(NA,(numEntityParts*2+2+1)))


    for ( i in 0: numEntityParts ){
        ret[i+1, 1]="varchar"
        ret[i+1, 4]=paste("rv1",x[entityStartIndex+i, 4],sep="_")
    }


    ret[numEntityParts+1, 1]="varchar"
    ret[numEntityParts+1, 4]="rv1_metric"


    for ( i in 0: numEntityParts ){
        ret[i + numEntityParts + 2, 1]="varchar"
        ret[i + numEntityParts + 2, 4]=paste("rv2",x[entityStartIndex+i,4],sep="_")
    }


    ret[numEntityParts*2+2, 1]="varchar"
    ret[numEntityParts*2+2, 4]="rv2_metric"


    ret[numEntityParts*2+2+1, 1]="float"
    ret[numEntityParts*2+2+1, 4]="correlation_coeff"
    ret
}
```

Note: Notice how the input parameters are used by the `mvCorrOutType` callback function to describe the output column names and types.

The names used above in the `outtypecallback` function are directly processed by AQL in its result processing and sent to the dashboard pane in the OpsA console.

Registering R function

You must register a newly created R Function with both Vertica and OpsA.

Registering your R function with Vertica

1. Prepare the .R file so that it contains the following:
    - The core R function implementing your custom analytics logic.
    - The Vertica R UDX factory function.
    - The parameter type callback R function.
    - Any optional output type callback R function.

    - Any other helper R functions used by the core R function.

2. Run the Vertica R UDX load commands to load the R function into Vertica. At this stage the R function becomes available as a valid UDX that can be invoked from Vertica SQL.
   Note: You must complete these steps as a valid Vertica database user who has the privileges to run SQL commands and who can create UDX functions in the Vertica database system.

   The following are example Vertica SQL commands that load the example R UDX provided by MVCorr.R:

   ```
   create library mvCorrLib as '/home/dbadmin/functions/MVCorr.R'
   language 'R';

   create transform function mvCorr as language 'R' name
   'mvCorrFactory' library mvCorrLib;
   ```

   You can also review Vertica documentation on how to load Vertica R UDX functions.

Registering your R function with OpsA

Once the R function is loaded and available in Vertica, you need to tell OpsA about it by registering an R function module into OPSA.

1. Create an R module specification file. The following example shows the contents of one such module definition file that defines the R module for the multi-variate correlation R UDX function example from the /`opt/HP/opsa/inventory/lib/hp/r-udx-examples/mvCorr.R` file.

   ```
   module MultiVariate;


   /* Does multivariate correlation */

   define mvCorr input(any, integer, integer) output(any);
   ```

   Save the content in a text file. For example, see /`opt/HP/opsa/inventory/lib/hp/r/multivariate.rpsec`.

2. Load the R module specification into OPSA by running the following command:

```
/opt/HP/opsa/bin/opsa-rspec-module-manager.sh -?
```

   You should see an output similar to the following:

```
OPSA_HOME is set to /opt/HP/opsa


-t <tenant name>                Name of Tenant (mandatory argument except when using –v
option)

-v <file>                       Validate File

-l modules                      List Summary of Loaded Modules

-l all                          List Contents of All Loaded Modules

-l <modulename>                 List Contents of Module

-i <file>                       Import File

-a <authorname>                 Specify Author for Import File

-d <modulename>                 Delete Module

-?                              This help message
```

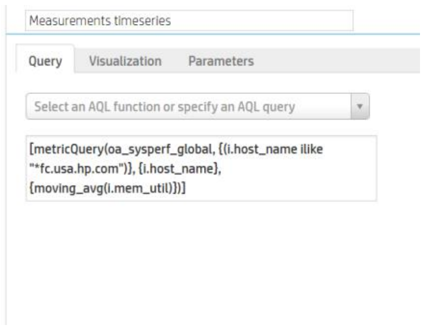
For example, you could load the R module named `MultiVariate` previously defined in the *`/opt/HP/opsa/inventory/lib/hp/r/multivariate.rpsec`* file by running the following command:

```
/opt/HP/opsa/bin/opsa-rspec-module-manager.sh – t opsa_default -i
/opt/HP/opsa/inventory/lib/hp/r/multivariate.rpsec
```

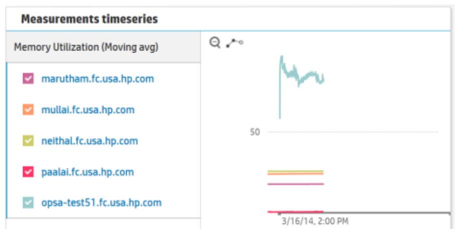
Using your R Function in an OpsA Dashboard

You can create your AQL function that returns time series data that you can visualize using an OpsA line chart, heatmap chart, or bar chart UI elements. For more information see *About Analytics Query Language (AQL) Functions* in the *Operations Analytics Help*.

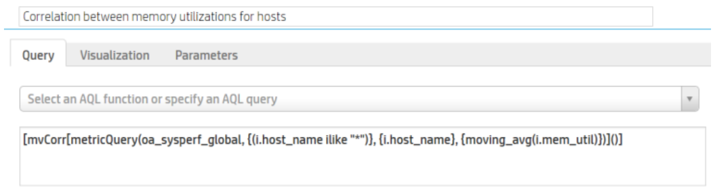
In a dashboard pane, you can visualize the results of the AQL function by using it in the query edit box for the pane as shown below:



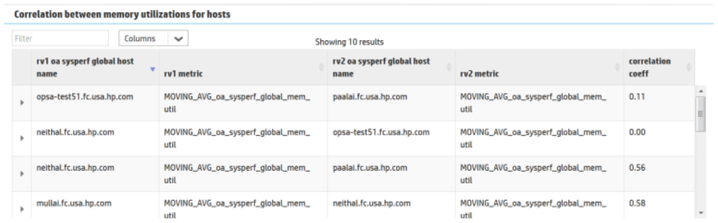
```
[metricQuery(oa_sysperf_global, {(i.host_name ilike
"*fc.usa.hp.com")}, {i.host_name},
{moving_avg(i.mem_util)})]
```



Now surround the AQL function call with a call to an R function as shown below to trigger the invocation of the registered R function:


Correlation between memory utilizations for hosts
Query
Visualization
Parameters
Select an AQL function or specify an AQL query

```
[mvCorr[metricQuery(oa_sysperf_global, {(i.host_name ilike "*")}, {i.host_name}, {moving_avg(i.mem_util)})]()]
```



rv1 oa sysperf global host name	rv1 metric	rv2 oa sysperf global host name	rv2 metric	correlation coeff
opsa-test51.fc.usa.hp.com	MOVING_AVG_oa_sysperf_global_mem_util	paalai.fc.usa.hp.com	MOVING_AVG_oa_sysperf_global_mem_util	0.11
neithal.fc.usa.hp.com	MOVING_AVG_oa_sysperf_global_mem_util	opsa-test51.fc.usa.hp.com	MOVING_AVG_oa_sysperf_global_mem_util	0.00
neithal.fc.usa.hp.com	MOVING_AVG_oa_sysperf_global_mem_util	paalai.fc.usa.hp.com	MOVING_AVG_oa_sysperf_global_mem_util	0.56
mullai.fc.usa.hp.com	MOVING_AVG_oa_sysperf_global_mem_util	neithal.fc.usa.hp.com	MOVING_AVG_oa_sysperf_global_mem_util	0.58

Limitations

Only a table visualization of the invoked R function is supported in OpsA 2.10.

You can only invoke R functions on data from a single collection. In other words, only a single AQL function could be passed as an argument to an R function invoked from a dashboard pane query editor. It is not possible to invoke R functions from OpsA dashboards on AQL expressions (such as concatenation of multiple AQL function invokes) representing queries of data from potentially multiple OpsA collections.

Support

Visit the HP Software Support web site at:

www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches and associated patch documentation
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp