Don't Hold My UDFs Hostage - Exporting UDFs For Debugging Purposes

Mark Raasveldt
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
m.raasveldt@cwi.nl

ABSTRACT

Data scientists rely on scripting languages, such as R and Python, to perform data analysis tasks. Combining these languages with a database has many advantages. The traditional method of using a database in conjunction with these scripting languages is to connect to a database using a client protocol. The data is then transferred from the database to the analytical tool. However, this is very inefficient when dealing with large amounts of data.

In-database analytics promises to solve this issue. By performing the analytics inside the database, the data transfer overhead is mitigated. The primary way of performing in-database analytics is through the use of UDFs. As shown in Table 1, most database vendors support UDFs in at least one scripting language frequently used for analysis.

Table 1: DBMSs Support for UDFs

DBMS	Python	R
Greenplum	X	X
DB2	-	-
MonetDB	X	X
MySQL	-	-
Oracle	-	X
PostgreSQL	X	X
Redshift	X	-
SQL Server	X	X
SQLite	X	-
Vertica	-	X

While UDFs can perform highly efficient data analysis, debugging them has been a long-standing problem. Because the UDFs are executed within the database server, developers cannot use sophisticated debugging techniques (e.g., Interactive Debugging). Instead, they have to resort to inefficient debugging strategies in an effort to make their code work.

With the increasing number of R and Python packages, these analytical programs are quickly growing in complexity and scope. Finding and correcting bugs in complex programs is a challenging and time consuming task. In some cases this can take up to 50% of development time. However, when using more primitive debugging techniques, such as print or log-file debugging, this can take even longer.

In this work, we propose a solution to this problem. From within a client connected to a database, we ask the user to provide us with a problematic SQL statement that they want to debug. We then analyze it, and extract any required input data from the database server. We transfer the input data and the source code of the UDF to the client. Finally, we execute the function in the client process exactly as it would have been executed in the database server. The developer can then easily use interactive debugging tools to aid him in creating and modifying the UDFs.

The main contributions of this work are:

- We introduce a novel strategy that allows developers to use interactive debugging techniques in conjunction with UDFs.
- We provide an Open-Source implementation of our work for the popular database MonetDB.
- We show our debugging framework being used for a concrete use-case, and compare it against the old ways of debugging UDFs.