SQL Server ML Services & Performance

A look at how SQL Server ML Services Performs

Agenda

- Push
- Streaming
- Parallel processing
- RevoScaleR / revoscalepy
- SQL Server Compute Context

Why In-database ML with SQL Server?

- Better Collaboration and Insights Sharing
- Streamlined Deployment of R/Python Scripts and Models
- Better Security and Compliance
- Faster Time to Insights

Faster Time to Insights

- Integration with SQL query execution
  - Parallel query pushing data to multiple external processes / threads
  - Use in-memory technology and Columnstore Indexes alongside your ML scripts
- Streaming mode execution
  - Stream data in batches to the R/Python process to scale beyond available memory
- Train and Predict using parallelism
  - Leverage RevoScaleR/revoscalepy and scale your R and Python scripts using multi-threading and parallel processing
- Native scoring for faster real-time predictions (New in 2017)

Push Data from SQL Server to External Runtime

- By using @input_data_1 you push data to external runtime
- Internally it uses a very efficient data transfer format: Binary eXchange Language (BXL).
- This is the only option when not using rx* functions.
- Requires enough memory to process and store data.

Push Data Code

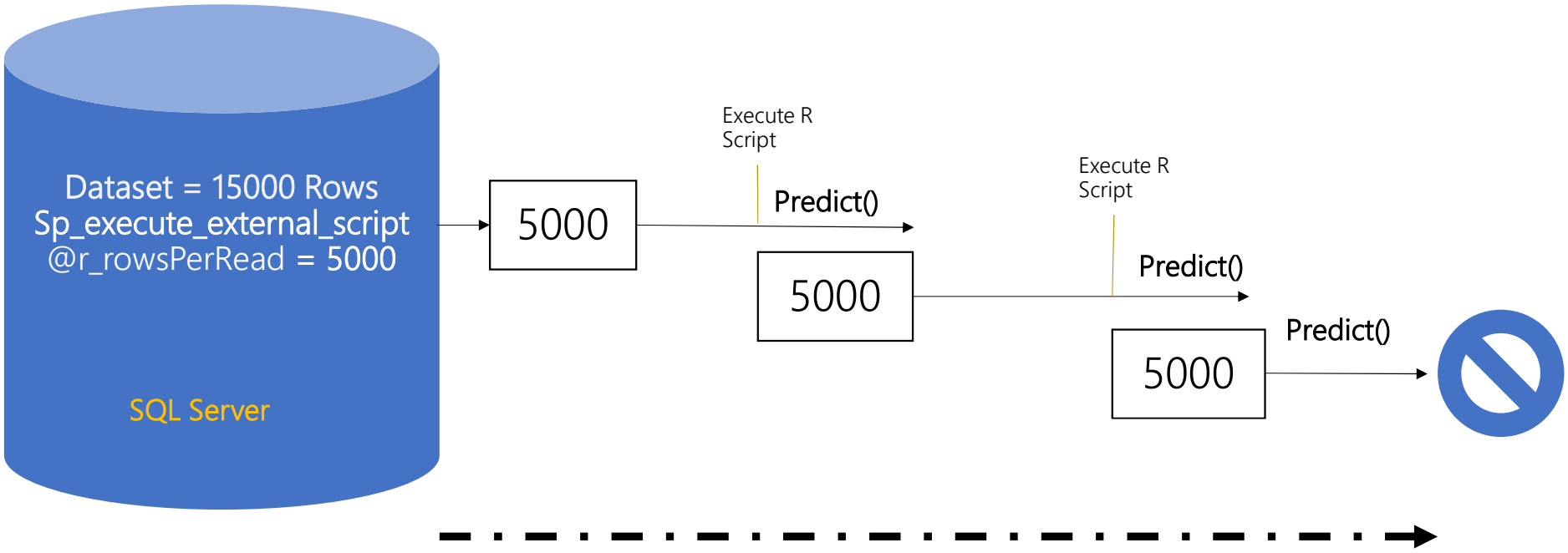
```
DECLARE @mod varbinary(max);
EXEC sp_execute_external_script
      @language = N'R'
    , @script = N'
          myModel <- glm(y ~ rand1 + rand2 + rand3 + rand4 + rand5,
                       data=InputDataSet)
           model <- serialize(myModel, NULL);'
   , @input_data_1 = N'
          SELECT  TOP(2500000) y, rand1, rand2, rand3, rand4, rand5
          FROM dbo.tb_Rand_5M TABLESAMPLE(75 PERCENT) REPEATABLE(98074)',
     @params = N'@model varbinary(max) OUT',
     @model = @mod OUT

INSERT INTO dbo.tb_Model2(ModelName, ModelBin)
VALUES ('GLM_75Pct', @mod);
GO
```

Streaming - I

- Streaming work with more data than fits in memory.
- It allows you to execute over chunks of data
- Can be used both in client (rx* functions) and server.
- However, it only works if no dependency between rows (like scoring).
- You define it with the `@r_rowsPerRead` parameter.

Streaming - II



http://nielsberglund.com

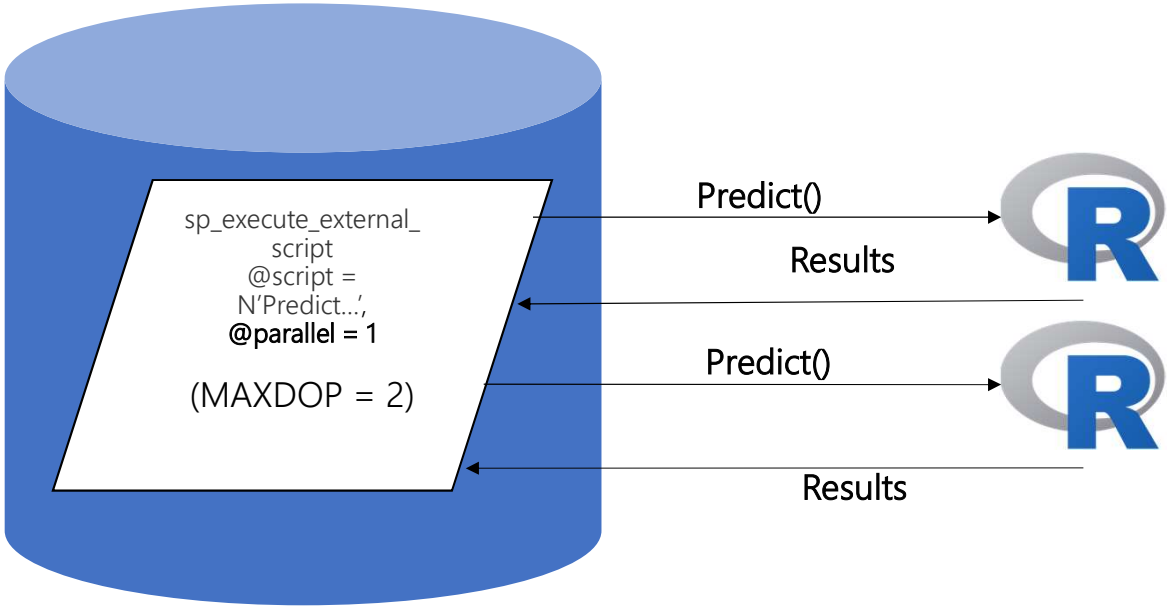
Streaming Code

```
DECLARE @model varbinary(max) = (SELECT TOP(1) ModelBin
                                 FROM dbo.tb_Model2
                                 WHERE ModelName = 'GLM_75Pct');
  EXEC sp_execute_external_script @language = N'R',
     @script = N'
       mod <- unserialize(model);
       OutputDataSet <- data.frame(predict(mod,
                                  newdata = InputDataSet,
                                  type = "response"))',
  @input_data_1 = N' SELECT TOP(10000000) y, rand1, rand2, rand3, rand4, rand5
              FROM dbo.tb_Rand_50M',
  @params = N'@model varbinary(max), @r_rowsPerRead int',
  @model = @model,
  @r_rowsPerRead = 100000
WITH RESULT SETS ((Y_predict float));
```

Parallel Execution

- SQL Server supports parallel execution.
  - can define number of processors through MAXDOP
- SQL decides whether to execute in parallel or not.
- In ML Services you need to indicate that you want parallel execution: `@parallel = 1`.
  - SQL still decides whether to execute in parallel
  - A parallel query plan is required.
- As with streaming, it only works if no dependency between rows (like scoring).

Parellelism



http://nielsberglund.com

Parallel Code

```
DECLARE @model varbinary(max) = (SELECT TOP(1) ModelBin FROM dbo.tb_Model2
                                 WHERE ModelName = 'RxLM_75Pct');
  EXEC sp_execute_external_script @language = N'R',
     @script = N'
       mod <- unserialize(model);
       OutputDataSet <- data.frame(rxPredict(modelObject = mod,
           data = InputDataSet,
           outData = NULL,
           type = "response",
           writeModelVars = FALSE, overwrite = TRUE));',
  @input_data_1 = N'SELECT TOP(2800000) y, rand1, rand2, rand3, rand4, rand5
              FROM dbo.tb_Rand_5M WHERE  rand5 >= 10
OPTION(querytraceon 8649, MAXDOP 4)',
  @parallel = 1,
  @params = N'@model varbinary(max)',
  @model = @model
WITH RESULT SETS ((Y_predict float));
```

http://nielsberglund.com

RevoScaleR / revoscalepy

- R / Python packages providing High Performance Computing and High Performance Analytics.
- Distribute execution across cores and nodes
- Highly memory optimized.
- Built in support for multithreading
- Supports Compute Contexts

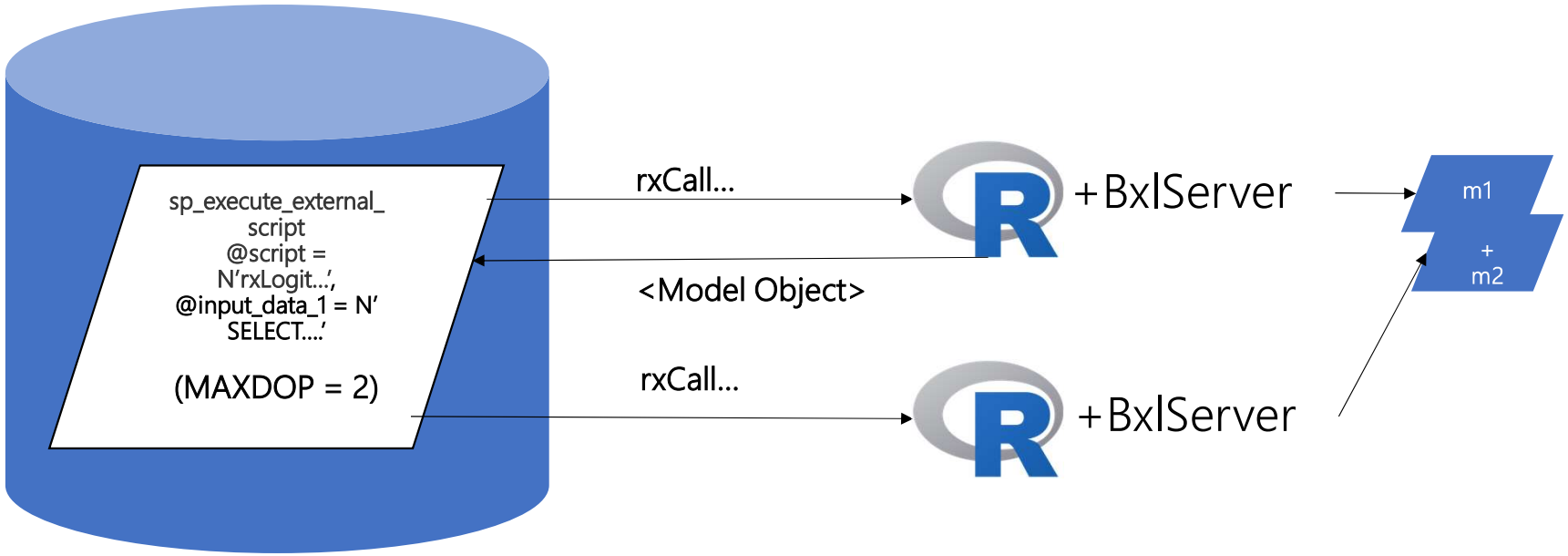
Compute Context

- Physical location of the computational engine handling a given workload.
- By setting a compute context you switch from local to remote, pushing execution of data-centric functions to a computational engine on another system.
- Compute Contexts
  - local
  - Spark
  - SQL Server

SQL Server Compute Context

- Can be used from both client and server.
- Leverages multiple CPU's
- Integrates with parallel query execution (without setting `@parallel`).
- Connection string to SQL Server to execute on required.
- Setting number of parallel tasks.
- RevoScaleR/revoscalepy methods automatically enlisted in compute context.
- Used for training as well as scoring in parallel.

Parallel Training / Scoring



Parallel Training

```
EXEC sp_execute_external_script
      @language = N'R'
    , @script = N'
    # set up the connection string
    sqlServerConnString <- "Driver=SQL Server;server=.\\sqlsat;
                            database=SqlSatPreCon; uid=sa;pwd=sapwd"

    # set up a compute context
    sqlCtx <- RxInSqlServer(connectionString = sqlServerConnString, numTasks = 4)
    # set the compute context to be the sql context
    rxSetComputeContext(sqlCtx)

    mydata <- RxSqlServerData(sqlQuery = "SELECT y, rand1, rand2, rand3, rand4, rand5
                                          FROM dbo.tb_Rand_5M",
                              connectionString = sqlServerConnString);

    myModel <- rxLinMod(y ~ rand1 + rand2 + rand3 + rand4 + rand5, data = mydata);'
```

Summary

- Push data to external script engine.
- Data can be streamed.
- Support for parallel processing
- Compute context pushes execution to remote.