An Array Library for Microsoft SQL Server with Astrophysical Applications

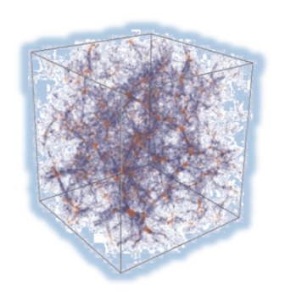
László Dobos[1], Alexander S. Szalay[2], José Blakeley[3], Bridget Falck[2], Tamás Budavári[2] and István Csabai[1]

[1] Eötvös Loránd University, Department of Physics od Complex Systems, Budapest, Hungary
[2] The Johns Hopkins University, Department of Physics & Astronomy, Baltimore, USA
[3] Microsoft Corporation, Redmond, USA

Today's scientific simulations produce output on the 10-100 TB scale. This unprecedented amount of data requires data handling techniques that are beyond ordinary files. Relational database systems have been successfully used to store and process scientific data but the new requirements constantly generate new challenges. Moving terabytes of data among servers on a timely basis is a tough problem, even with the newest high-throughput networks. Thus, moving the computations as close to the data as possible and minimizing the client--server overhead are absolutely necessary. At least data subsetting and preprocessing have to be done inside the server process. Out of the box commercial database systems perform very well in scientific applications from the prospective of data storage optimization, data retrieval and memory management, but lack basic functionality like handling scientific data structures or enabling advanced math inside the database server. The most important gap in Microsoft SQL Server is the lack of a native array data type. Fortunately, the technology exists to extend the database server with custom-written code that enables us to address these problems.

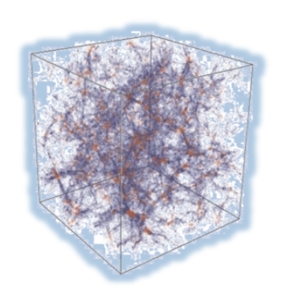
We present the prototype of a custom-built extension to Microsoft SQL Server that adds array handling functionality to the database system. With our Array Library, fix-sized arrays of all basic numeric data types can be created and manipulated efficiently. Also, the library is designed to be able to be seamlessly integrated with the most common math libraries, such as BLAS, LAPACK, FFTW etc. With the help of these libraries, complex operations such as matrix inversions or Fourier transformations can be done on-the-fly, from SQL code, inside the database server process.

We are currently testing the prototype with a bunch of different scientific data sets: The Indra cosmological simulation will use it to store particle and density data from N-body simulations, the Milky Way Laboratory project will use it to store galaxy simulation data.



The Indra Cosmological N-body DM Simulations Database

- 512 random instances, 1 Gpc h-1, WMAP 7 cosmology
  - 1024^3 particles, 64 snapshots per run
  - particle mass of ~10^{11} M_{Sun}
  - indentified dark matter halos and merging history
  - hundreds of TB worth of data
- Particle positions and velocities in SQL arrays
  - nearby particles grouped into chunks
  - chunks distributed along space filling curves
  - spatial search enabled
- Complex Fourier amplitudes of the density fields



Milky Way Laboratory

- Dark matter + gas dynamics at very high resolution
  - test dark matter properties on small scales
  - test assembly of various stellar populations
- Particle positions and velocities in arrays

Array manipulation

- Basic array manipulation functions
  - get and set items & subarrays
  - string conversion to and from
  - transpose and index permutation
  - basic matrix and vector algebra
- Aggregates over array items
  - min, max, average, stdev etc.
- Integration with relational data
  - convert arrays into SQL tables
  - aggregate values in SQL tables into arrays
  - slice arrays along arbitrary dimensions

Math library integration

- Call LAPACK, FFTW etc. directly from SQL!
- Optimized interface to native libraries
  - arrays stored in column major format
  - easy integration with FORTRAN code
  - pass pointers referencing server memory
  - no communication protocol overhead

Two storage models

- Optimized to match DB system storage model
- Small arrays (smaller than 8 kB)
  - for data points
  - stored in-page as varbinary(n)
  - up to 6 dimensions
  - int16 indices
  - optimized for speed, minimal overhead
- Max arrays (up to 2 GB)
  - for data grids
  - stored out of page as varbinary(max)
  - arbitrary number of dimensions
  - int32 indices
  - optimized for partial loading

Syntax examples

- Direct user-defined function calls
- Plans for pre-parser for array-enabled SQL

```
DECLARE @a VARBINARY(100) = FloatArray.Vector_5(1.0, 2.0, 3.0, 4.0, 5.0)  -- init vector
DECLARE @m VARBINARY(100) = FloatArray.Matrix_2(0.1, 0.2, 0.3, 0.4)       -- init matrix

SELECT FloatArray.Item_2(@m, 1, 0)                                         -- get item
SET @a = FloatArray.UpdateItem_2(@a, 0, 1, 4.5)                            -- set item

DECLARE @a VARBINARY(MAX)                                          -- storage for new array
DECLARE @l VARBINARY(100) = IntArray.Vector_2(100, 200)            -- size of new array
SELECT @a = FloatArrayMax.Concat(@l, ix, v) FROM table             -- aggregate from table

-- extract subarray from a large array (by offset and size)
SET @b =
   FloatArrayMax.Subarray(@a, IntArray.Vector_3(1, 4, 6), IntArray.Vector_3(5, 5, 5), 0)

-- convert array to a table
DECLARE @a varbinary(100) =  SqlArray.FloatArray.Parse('[[1,2,3],[4,5,6]]')
SELECT ix, v FROM SqlArray.FloatArray.ToTable(@a)
```

http://voservices.net/sqlarray

This work was supported by the following Hungarian grants: NKTH: Polányi, OTKA-80177 and KCKHA005.
The Project is supported by the European Union and co-financed by the European Social Fund (grant agreement no. TÁMOP 4.2.1./B-09/1/KMR-2010-0003).