Basic Concept

- Implement computational functionality in a separate process from the database server process
- Access through inter-process communication (IPC)
Why Do It This Way?

- Avoid DBMS permissions, threading restrictions, etc.
- Encapsulate native-code functionality (CUDA API calls)
- Exploit lower-level APIs to facilitate data movement between DBMS and CUDA implementation
SQL Callable Procedure

SQL code
SQL-callable procedure

IPC

CUDA functionality

Database server

Out-of-process server

- SQLCLR procedure in C#
  - Fills a shared-memory buffer with a SQL result set
  - Signals the out-of-process server and waits for a response
  - Returns data from the IPC buffer to the database server
Out-Of-Process Server

- Dynamically loads the CUDA implementation
- Invokes a method with a reference to the IPC buffer
- Signals the SQL-callable procedure that data is ready
Case Study 1: Spatial Statistics

- Correlation functions
  - Histogram of distances

- State of the art
  - Dual-tree traversal

- High resolution bins?
  - Just like brute force
Case Study 1: Spatial Statistics

- Correlation functions
  - Histogram of distances

- State of the art
  - Dual-tree traversal

- High resolution bins?
  - Just like brute force
User-Defined Functions

- Pair counts computed on GPU
  - Returns 2D histogram as a table (i, j, cts)
- Calculate the correlation fn in SQL

```sql
dbo.PairCounts(@maxmpc, @nbin, @qryD, @nD, null) dd
```
User-Defined Functions

- Pair counts computed on GPU
  - Returns 2D histogram as a table \((i, j, cts)\)
- Calculate the correlation fn in SQL

```sql
select dd.i, dd.j, dd.cts as dd, dr.cts as dr, rr.cts as rr
    (@Nrr*CONVERT(float,dd.cts)/@Ndd - 2*@Nrr*CONVERT(float,dr.cts)/@Ndr + CONVERT(float,rr.cts))
    / CONVERT(float,rr.cts) as xi
from dbo.PairCounts(@maxmpc, @nbin, @qryD, @nD, null) dd
    join dbo.PairCounts(@maxmpc, @nbin, @qryR, @nR, null) rr on dd.i = rr.i and dd.j = rr.j
    join dbo.PairCounts(@maxmpc, @nbin, @qryDR, @nD, @nR) dr on dd.i = dr.i and dd.j = dr.j
go
```
Async SQL Interface

```sql
declare @dd_req uniqueidentifier, @rr_req uniqueidentifier, @dr_req uniqueidentifier;

-- launch jobs on GPUs 0,1,2
set @dd_req = dbo.PairCountsAsyncBegin(0, @timeout, @maxmpc, @nbin, @qryD, @nD, null);
set @rr_req = dbo.PairCountsAsyncBegin(1, @timeout, @maxmpc, @nbin, @qryR, @nR, null);
set @dr_req = dbo.PairCountsAsyncBegin(2, @timeout, @maxmpc, @nbin, @qryDR, @nD, @nR);

-- wait for results
select dd.i, dd.j, dd.cts as dd, dr.cts as dr, rr.cts as rr, ...
from dbo.PairCountsAsyncEnd(@dd_req, @timeout) dd
    join dbo.PairCountsAsyncEnd(@rr_req, @timeout) rr on dd.i = rr.i and dd.j = rr.j
    join dbo.PairCountsAsyncEnd(@dr_req, @timeout) dr on dd.i = dr.i and dd.j = dr.j
go
```
590 Trillion Galaxy Pairs

800 × 800 bins
Summary

- New Moore’s law in GPUs – let’s use them!
- Novel extension to SQL Server
  - Provides direct access to multiple GPUs
  - Inter- and intra-task GPU parallelism
  - Also enables distributed parallel queries, MPI...
- Scientific applications
  - Corr fn, xmatch, photo-z and gene sequencing...