CATEGORY: BIG DATA ANALYTICS - BD03
POSTER P5233
CONTACT NAME Matthew England: mremtf@mail.missouri.edu
GPU TECHNOLOGY CONFERENCE

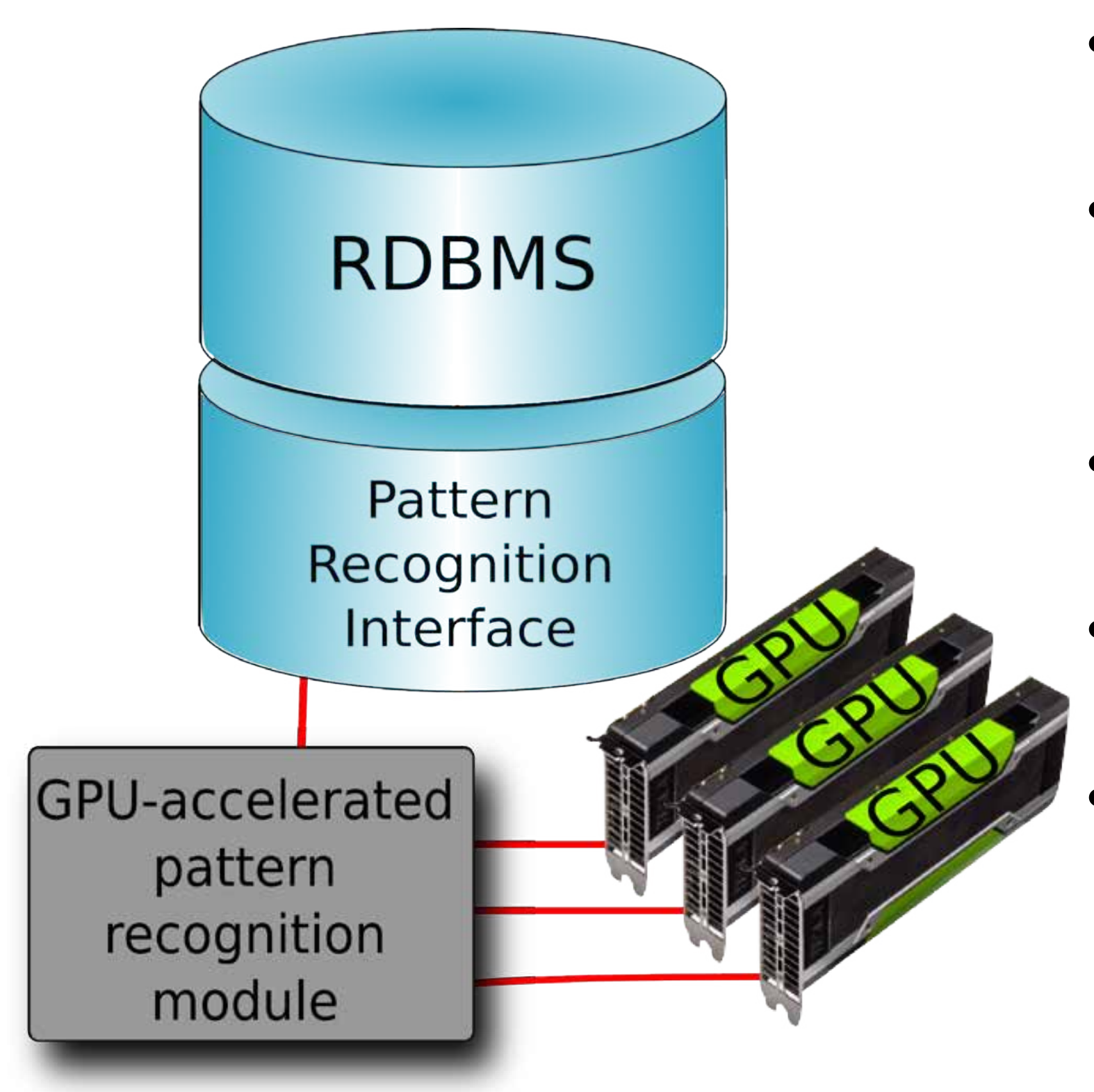
Large-Scale Pattern Recognition Using GPU-accelerated Relational Database

Matthew R. England and Dr. Grant Scott
Department of Computer Science
University of Missouri

Introduction

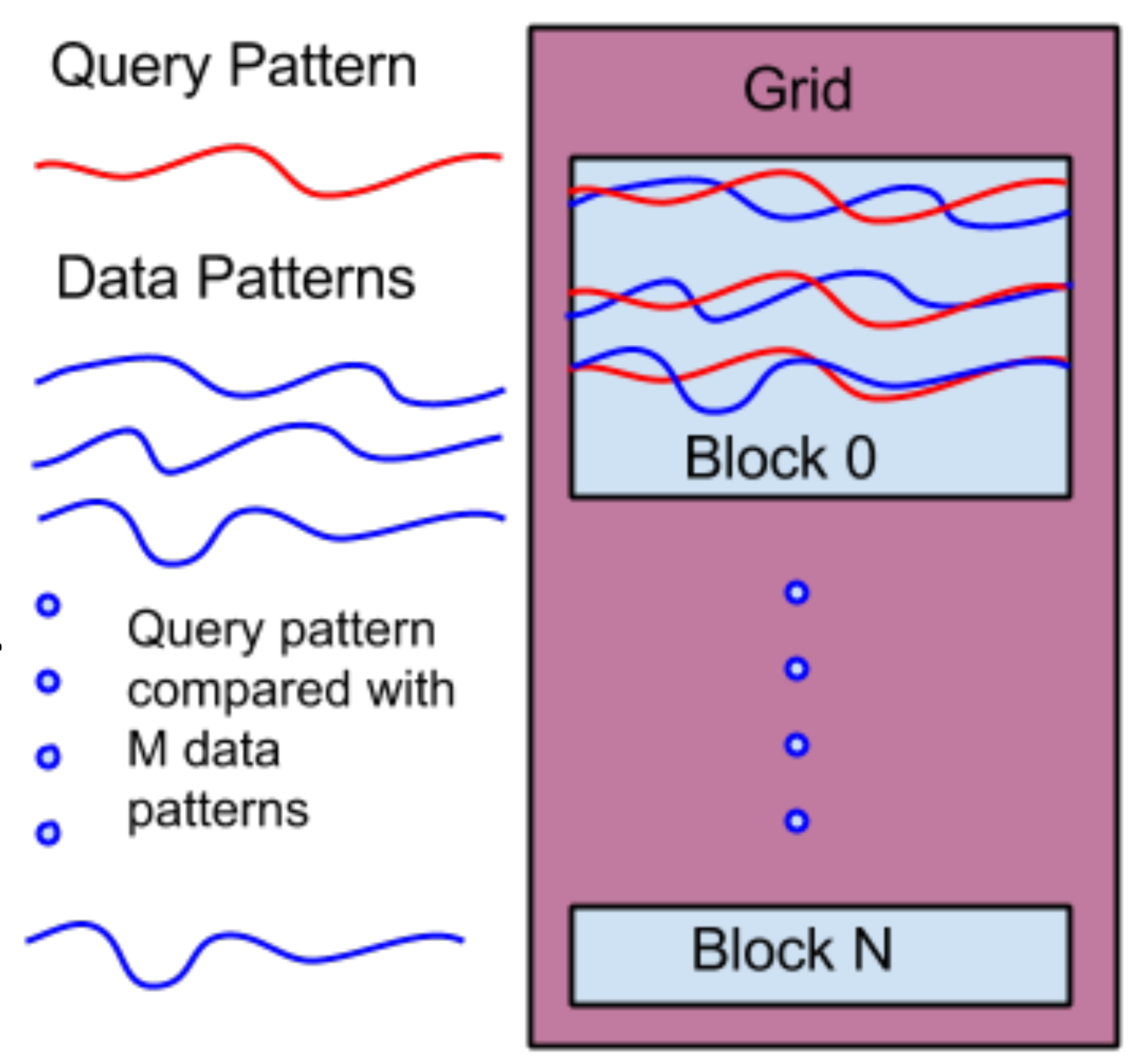
Numerous fields require large-scale pattern recognition to achieve a variety of computational goals. Scalability of systems designed for numerical pattern recognition often involve complex distributed architectures, custom data structures, and distributed programming. Traditional DBMS natively provide cost-effective scalable data access solutions. However, DBMS systems are not designed for high-performance numerical computation tasks. In contrast, GPU hardware is specifically designed for massively parallel computational tasks. To achieve a scalable, cost-effective architecture for large-scale pattern recognition we have developed a novel integration of GPU hardware into the PostgreSQL DBMS.

Building a Data Science Engine with a GPU-accelerated Database


RDBMS
Pattern Recognition Interface
GPU-accelerated pattern recognition module
GPU
GPU
GPU

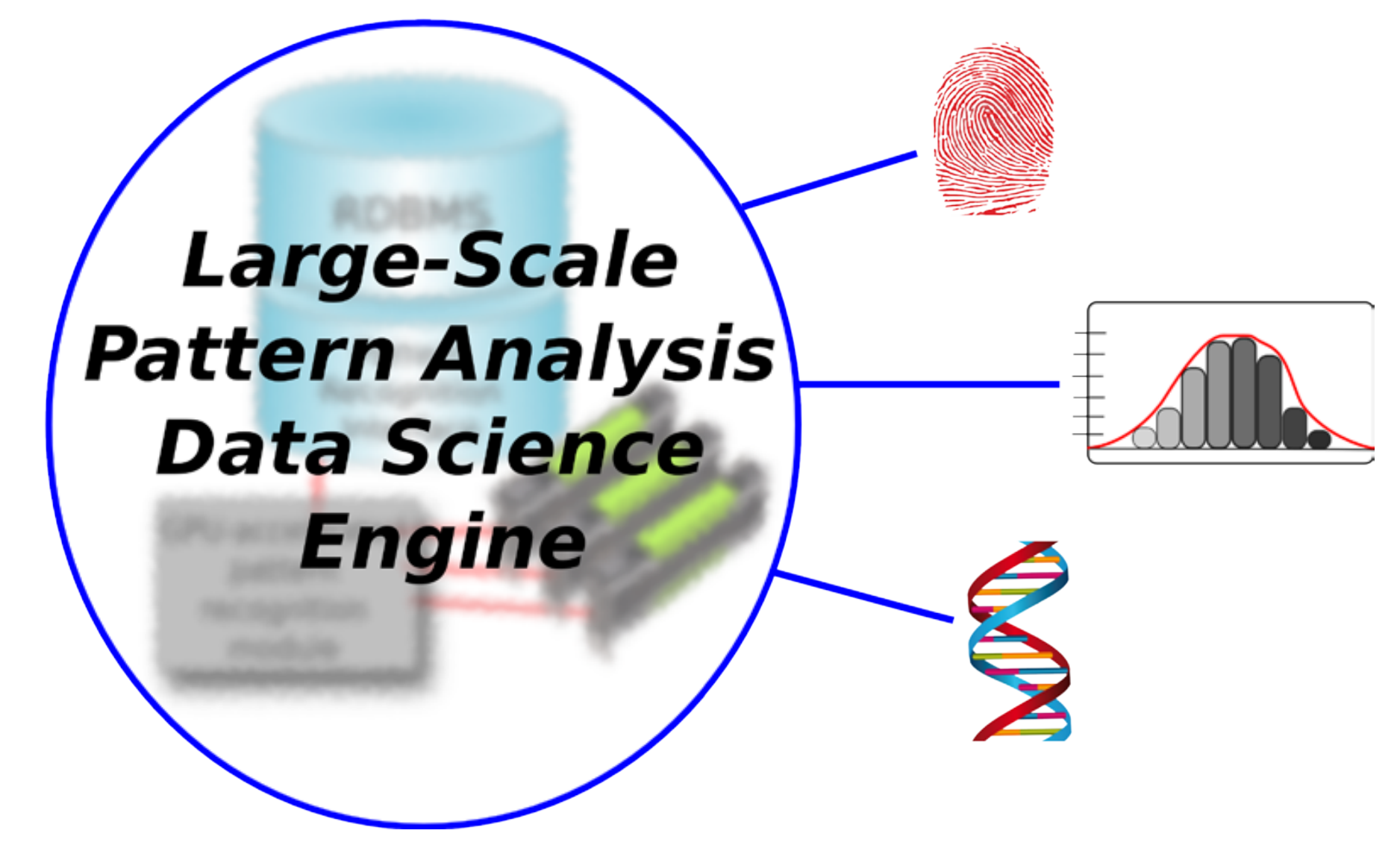
- Add computational pattern analysis
- GPU + *Server programming interface (SPI)*
- SPI extends database backend (C)
- GPU-based pattern analysis library (CUDA)
- Function interface provide output table expressions within the SQL environment

- Pattern data from an incoming table expression
- Partitioned into GPU grids
- Multiple grids can be used for large-scale pattern analysis
- GPU kernel invoked per-grid, matching query pattern against pattern data set
- Global results are accumulated across all grids via SPI library code



Benefits

- Cost effective solution for massively scaled *heterogeneous data analytics*
- Efficient data access and storage
  - Numerical Pattern Data
  - Spatial Data
  - Relational Data
  - … and more
- DBMS as a Data Science Engine
  - Leverage familiar SQL syntax
  - Numerical computation results as table expressions
  - Filtering via relational / spatial operators
    - Pre-filter pattern data
    - Post-filter pattern analysis results



Database Server Programming

- Database server programming interface (SPI)
- DB function calls, patternMatch(<table_expression>, <numerical_pattern>)

```
//Obtain the id associated with a pattern at the current row
Datum id = SPI_getbinval(SPI_tuptable->vals[i], SPI_tuptable->tupdesc, 1, &is_null);

// Obtain a pattern from the database at the current row
Datum raw_float4_array = SPI_getbinval(SPI_tuptable->vals[i],
SPI_tuptable->tupdesc, 2, &is_null);
ArrayType *pg_array = DatumGetArrayTypeP(raw_float4_array);
…
//Find current position in the bulkPattern array and copy pattern
const size_t bulkPatternOffset =bulkPatternRecord * patternSize;
float * bulkPatternLocation = bulkPattern + bulkPatternOffset
memcpy((void *) bulkPatternLocation, (void *)
ARR_DATA_PTR(pg_array), patternSize*sizeof(float));
…
//Call the module that calls the kl-divergence cuda function
bool cudaOk = klDivergenceCudaBulkMatch(0, bulkPatterns, pattern,
bulkPatternRecord, patternSize, bulkDiffResults);
…
```

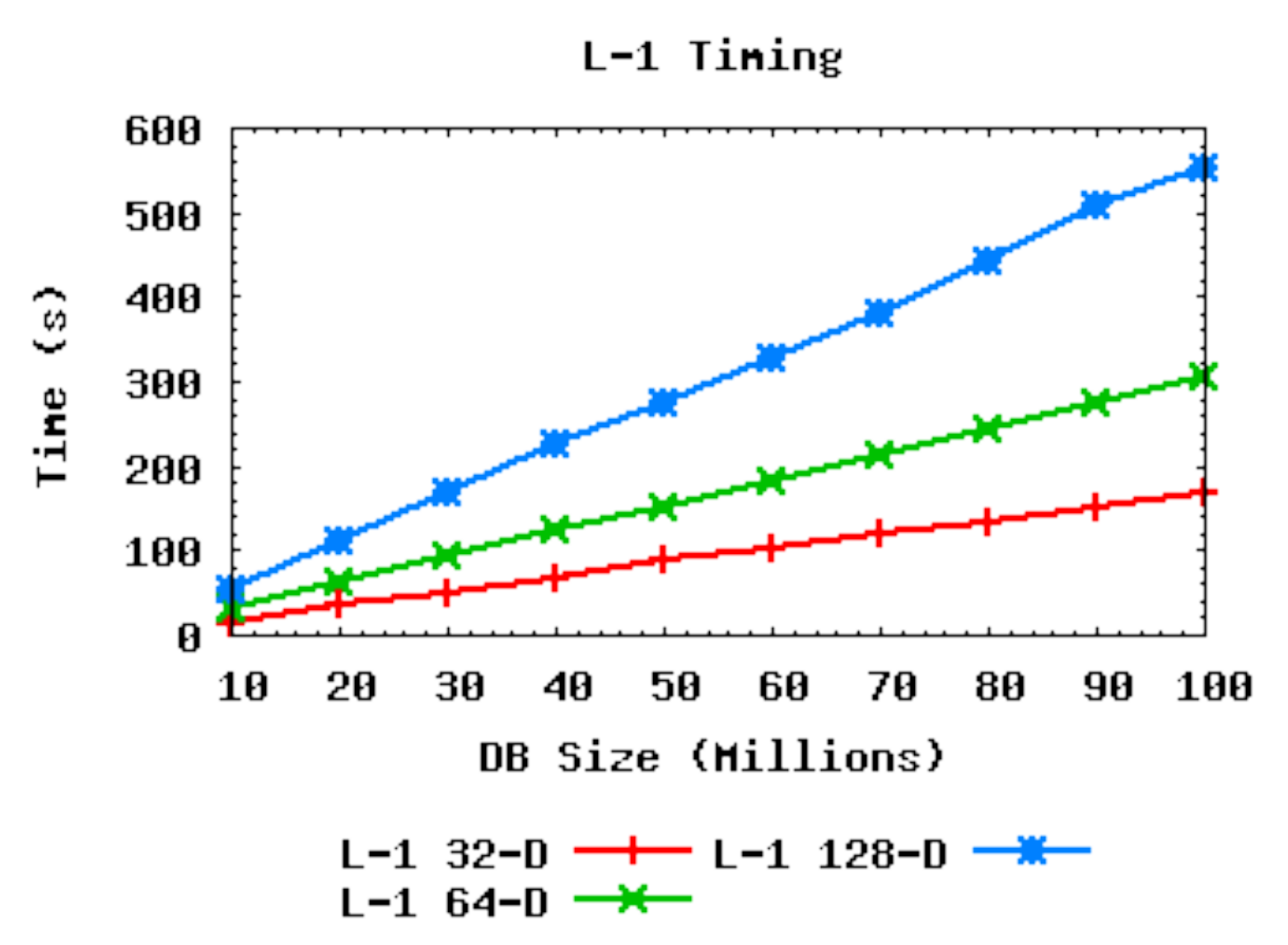
Pattern Analysis Kernels

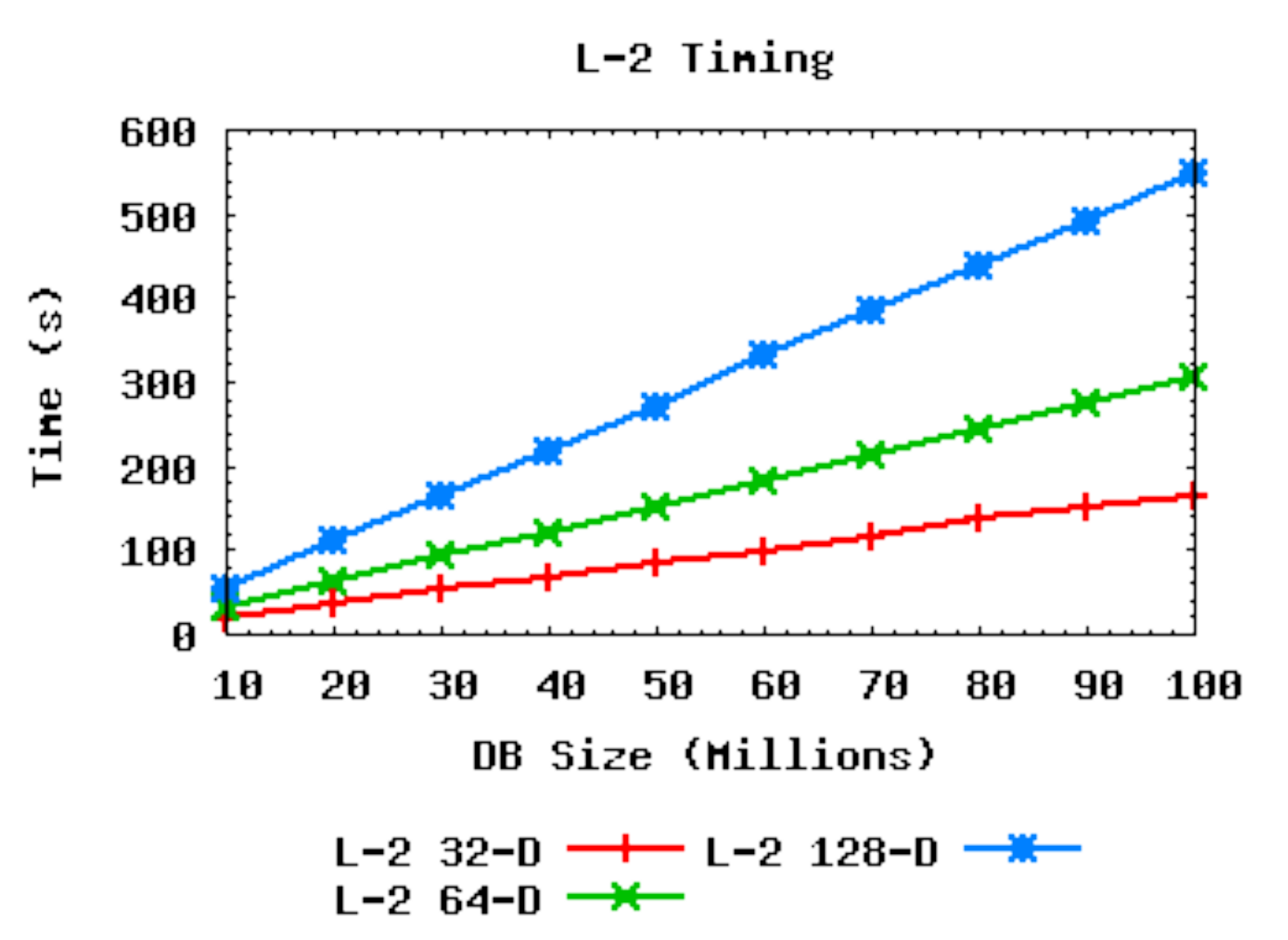
- L_1
- L_2
- KL-Divergence

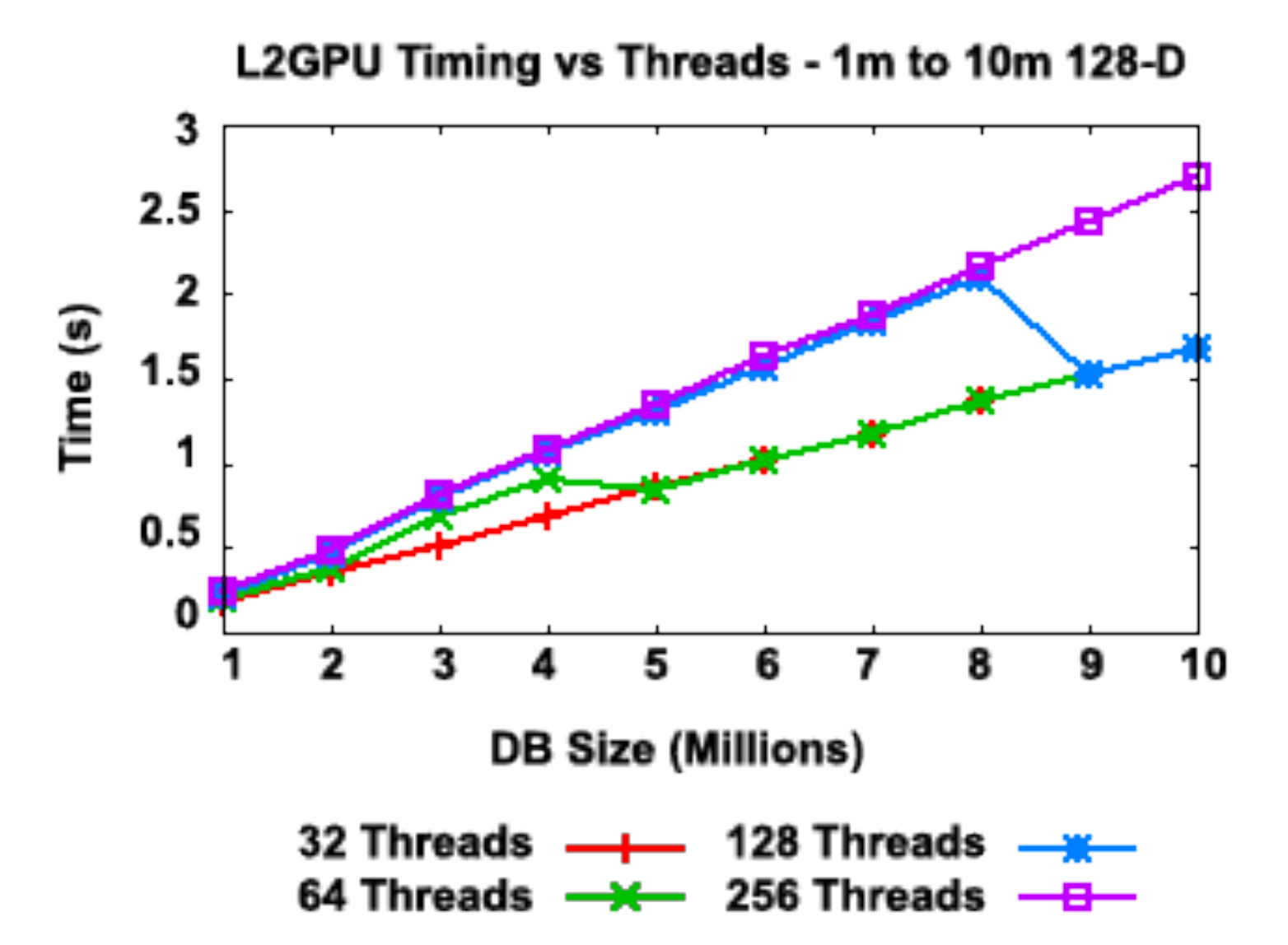
KL-Divergence measures the differences between probability distribution functions (PDF).

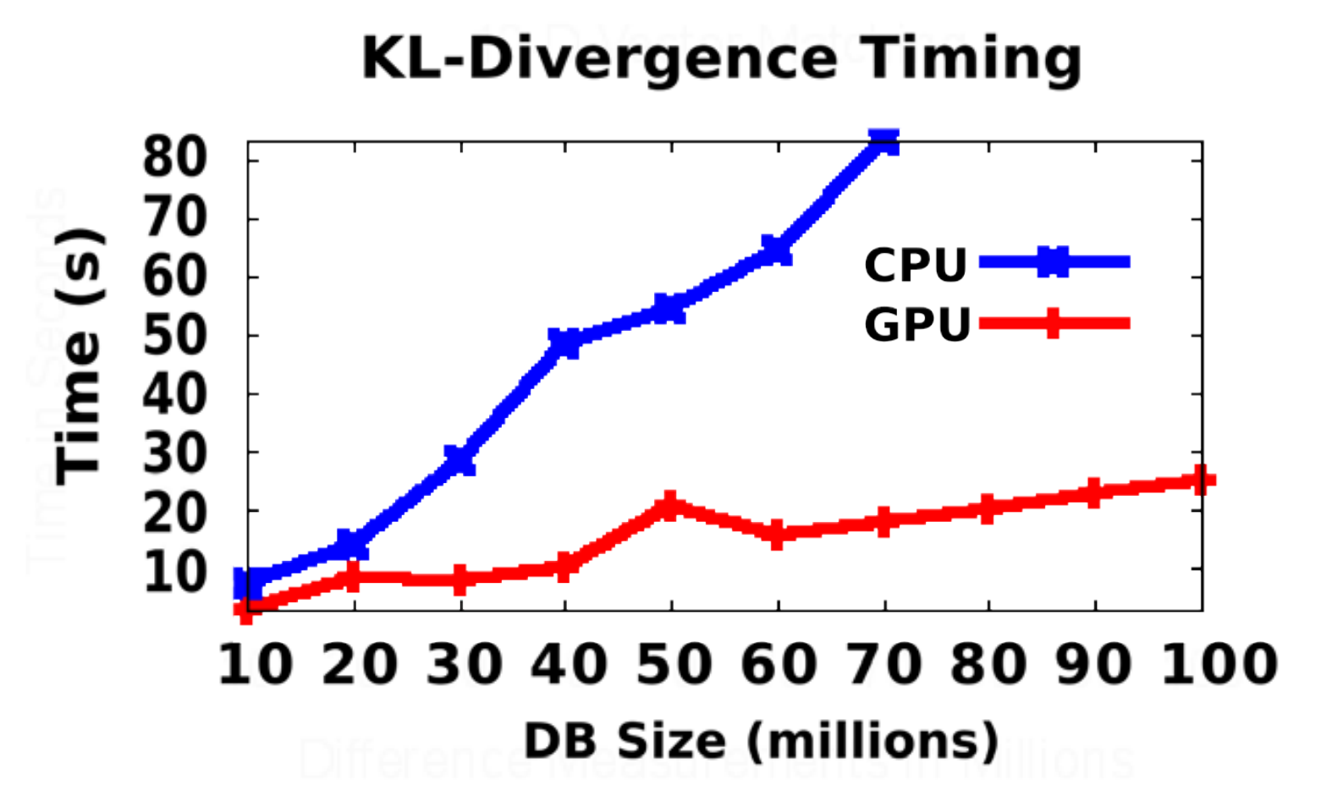
```
__global__ void kl_divergence (const float *bulkPDF, const float *
queryPDF, float *difference,const int bulkPDFSizeRows,const int
queryPDFSize)
{
 const int pos = blockIdx.x * blockDim.x + threadIdx.x;
  if (bulkPDFSizeRows <= pos) return;
  __shared__ float diff[1024]
  __shared__ int dataPosition[1024];
 dataPosition[threadIdx.x] = pos * queryPDFSize;
  int i = 0;
  for (i = 0; i < queryPDFSize; ++i) {
   if (queryPDF[i] > 0 && bulkPDF[dataPosition[threadIdx.x]] > 0)
     diff[threadIdx.x] +=  queryPDF[i] *
     log(queryPDF[i] bulkPDF[dataPosition[threadIdx.x]++]);
  difference[pos] = diff[threadIdx.x];
}
```

Timing Analysis









Future Work and Directions

We are focused on expanding the use of the DSE into various computationally intensive pattern analysis fields, such as biometrics, computer vision, and bioinformatics. We believe the use of the DSE's novel and scalable heterogeneous data analytical capabilities will find wide-spread applicability in a number of fields. The advantages of backend database integration using the GPGPU to handle data provides a efficient, cost-effective, and reliable data analysis platform. We are continuing to advance the scalability of the DSE through the exploration of distributed database system which will include our novel GPU integrations in the DBMS.