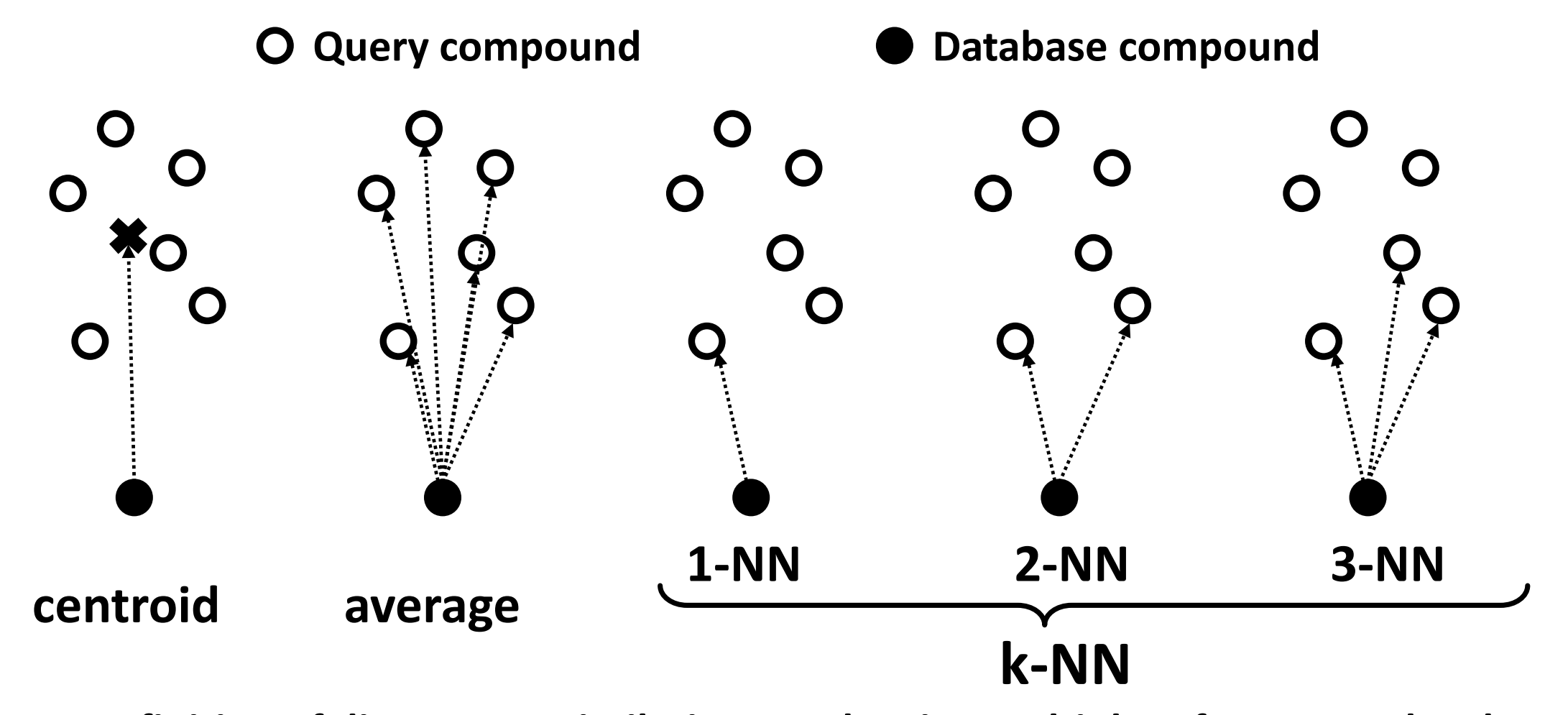
Efficient Similarity Search Using Multiple Reference Molecules on PG-Strom architecture

Kohei KaiGai[1] Atsushi Yoshimori[2]
<kaigai@ak.jp.nec.com> <yoshimori@itmol.com>

[1] NEC OSS Promotion Center, NEC Corporation, 1753, Shimonumabe, Nakahara-ku, Kawasaki, 211-8666, Japan
[2] Institute for Theoretical Medicine, Inc., 4259-3 Nagatsuta-cho, Midori-ku, Yokohama 226-8510, Japan

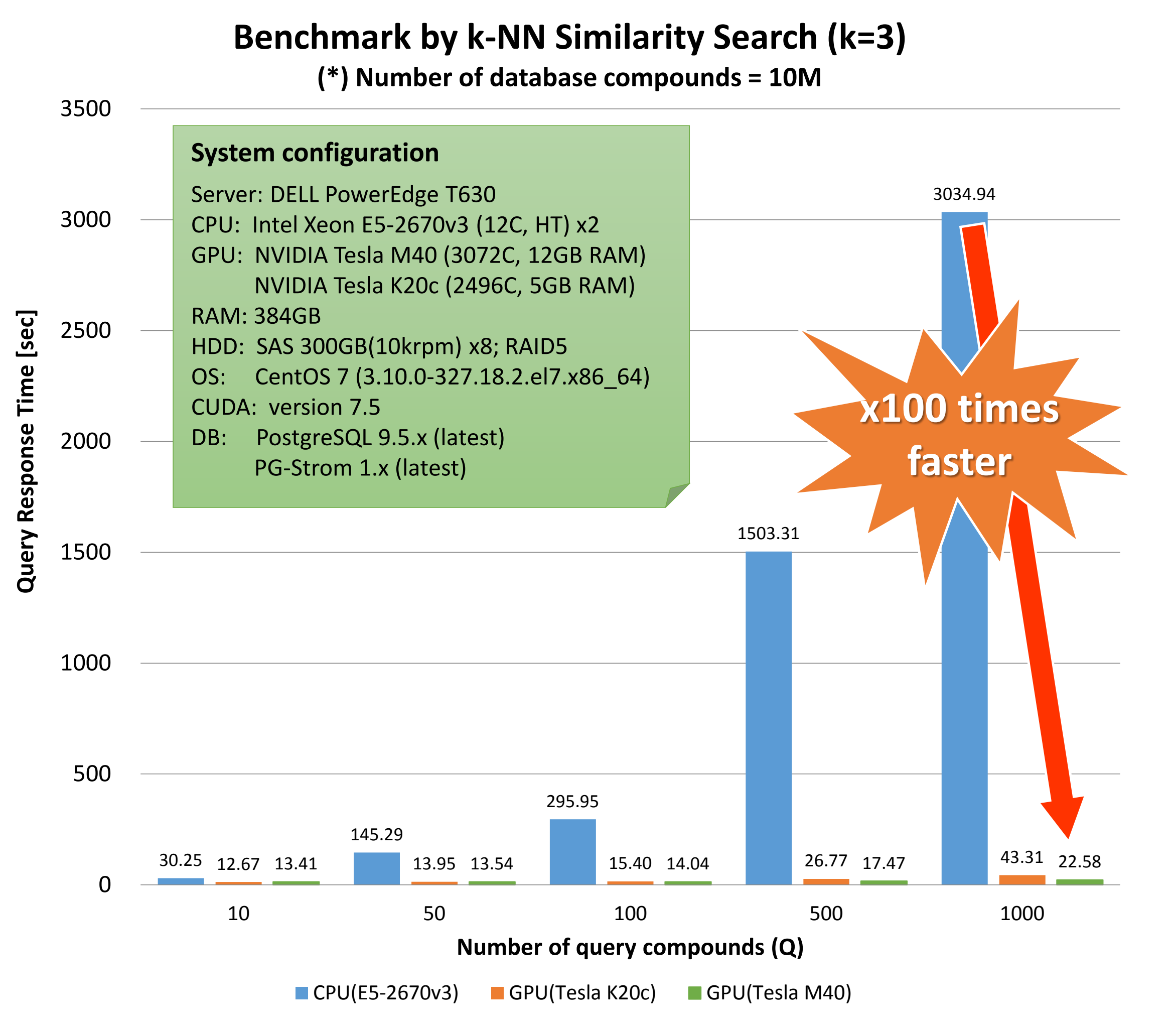
Introduction

In this study, we implemented k-nearest neighbors (k-NN) similarity search for multiple reference molecules as a SQL function which leverages more than thousands processor cores of GPU. Its performance was about as 20 times fast as the traditional CPU version.
By the acceleration of execution of complicated mathematical algorithm in-database, (1) we can utilize flexibility of SQL pre- and post-process to the main logic, and (2) we don't need to export entire data to process them by external tools.
These characteristics allows researchers to focus on their problems then eventually improves their productivity.
PG-Strom [1][2] is an extension of PostgreSQL [3]; designed to off-load some kinds of CPU intensive workloads to GPU devices automatically or manually. It fuses query processing capability with HPC grade computing power.
Similarity searches using multiple reference molecules are one of data fusion methods, which are widely used in virtual screening.
Various methods have been introduced to utilize multiple reference molecules, such as centroid fingerprints [3],
k-nearest neighbors (k-NN) [4], and SVM-based ranking [5]. Data fusion methods generally improve virtual screening performance.



Definition of distance on similarity search using multiple reference molecules

Performance Leap



System configuration
Server: DELL PowerEdge T630
CPU: Intel Xeon E5-2670v3 (12C, HT) x2
GPU: NVIDIA Tesla M40 (3072C, 12GB RAM)
NVIDIA Tesla K20c (2496C, 5GB RAM)
RAM: 384GB
HDD: SAS 300GB(10krpm) x8; RAID5
OS: CentOS 7 (3.10.0-327.18.2.el7.x86_64)
CUDA: version 7.5
DB: PostgreSQL 9.5.x (latest)
PG-Strom 1.x (latest)

```
-- construction of D-matrix (database chemical components)
SELECT matrix INTO finger_print_10m_matrix
  FROM (SELECT cbind(array_matrix(id),
                     array_matrix(bitmap))
          FROM finger_print_10m
      GROUP BY id % 4);

-- construction of Q-matrix (query chemical components)
SELECT id, name, bitmap
  FROM (SELECT * FROM finger_print
         ORDER BY random()
         LIMIT 100);

--
-- SQL Query for calculation of the chemical component
-- similarity based on k-NN method
--
PREPARE knn_sim_rand_10m_gpu_v2(int)    -- arg:@k-value
AS
SELECT float4_as_int4(key_id) key_id, similarity
  FROM matrix_unnest((SELECT rbind( knn_gpu_similarity($1,Q.matrix,
                                                         D.matrix))
                        FROM (SELECT cbind(array_matrix(id),
                                           array_matrix(bitmap)) matrix
                                FROM finger_print_query
                               LIMIT 99999) Q,
                             (SELECT matrix
                               FROM finger_print_10m_matrix) D))
    AS sim(key_id real, similarity real)
ORDER BY similarity DESC
LIMIT 1000;

EXECUTE knn_sim_rand_10m_cpu_v2(3);
```

Similarity Calculation
Load the Q-matrix
Load the D-matrix

Query for the k-NN similarity search using PL/CUDA function

We compared response time of SQL queries using two different version of SQL functions implemented with C and PL/CUDA(GPU). Problem size is $O(Q \times D)$. Both of the results almost follow the problem scale, on the other hands, GPU's results show much mild increase of the response time than CPU's one.
Another 'k' choice (1 and 5) didn't affect to the results significantly. To avoid the effect by I/O, we preload all the data onto the shared buffer of PostgreSQL. Entire database size was 1.5GB.

Advantages

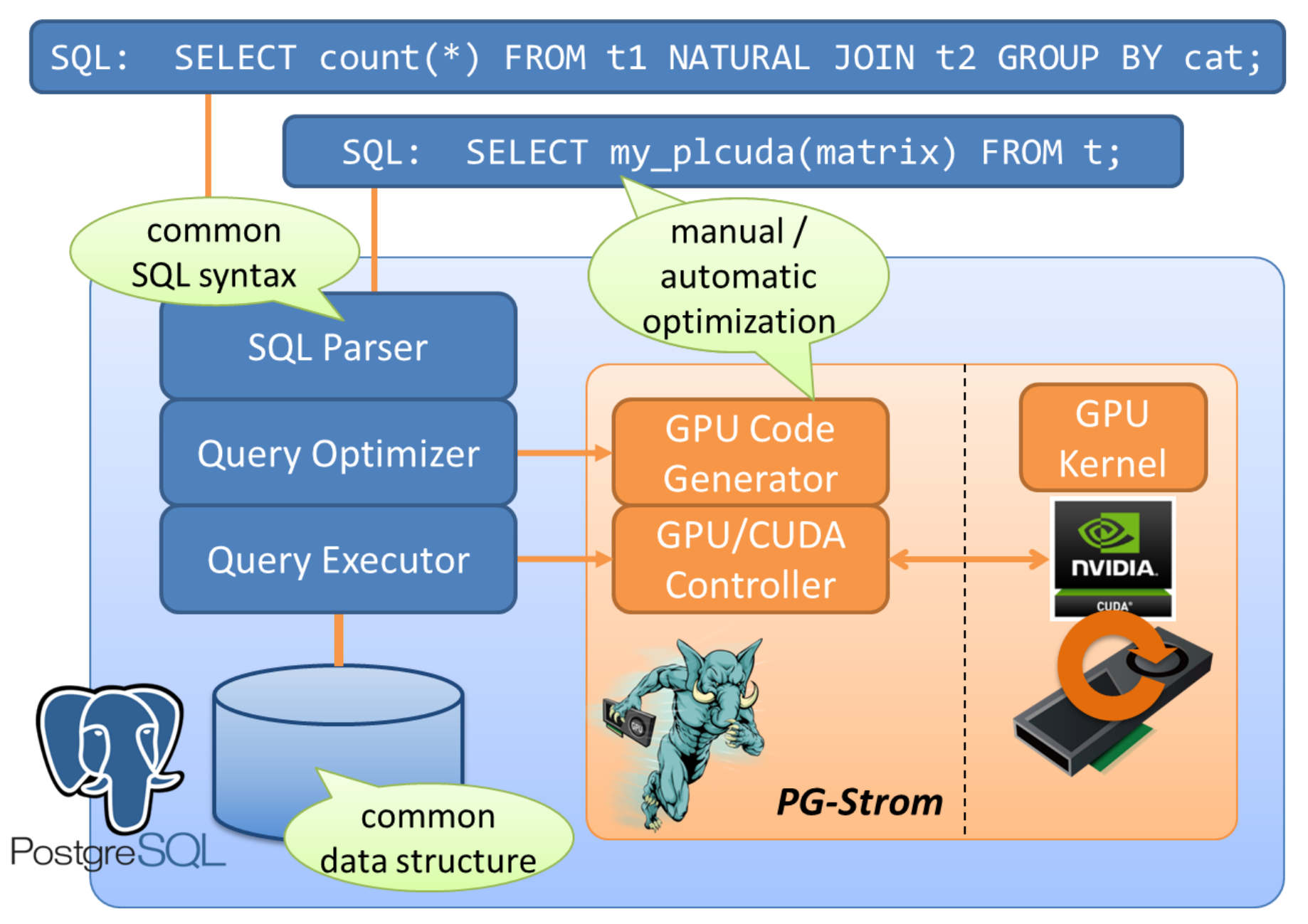
① No need to export the entire database

As a literal, DBMS is a software that is designed to manage database, not for a heavy calculations. Thus, people often export the database to process by external applications designed for specific calculation and analytics logics.
Once GPU's computing power got integrated with DBMS, it allows to run heavy calculations on the location closest to the data, and also allows to export the results only; already processed by PL/CUDA functions. It eliminates waste of network traffic and necessity of extra data management.

② SQL's flexibility for data manipulation on pre-/post-process of the analytics

SQL supports many features for data analytics – like window functions, aggregate calculations, sorting, tables join, etc...
Once complicated mathematical algorithms can be executed within database system, we can apply these SQL features for data analytics towards the results of PL/CUDA functions. It is much productive way to produce what we need to see. E.g, it is quite easy to pick up the top-100 compounds according to the similarity. All we need to do is adding "`ORDER BY similarity LIMIT 100`" on the tail of query.
If we would process the database by external applications, it needs to write back to the database system or re-invention of wheel.

Elements: PG-Strom + PL/CUDA



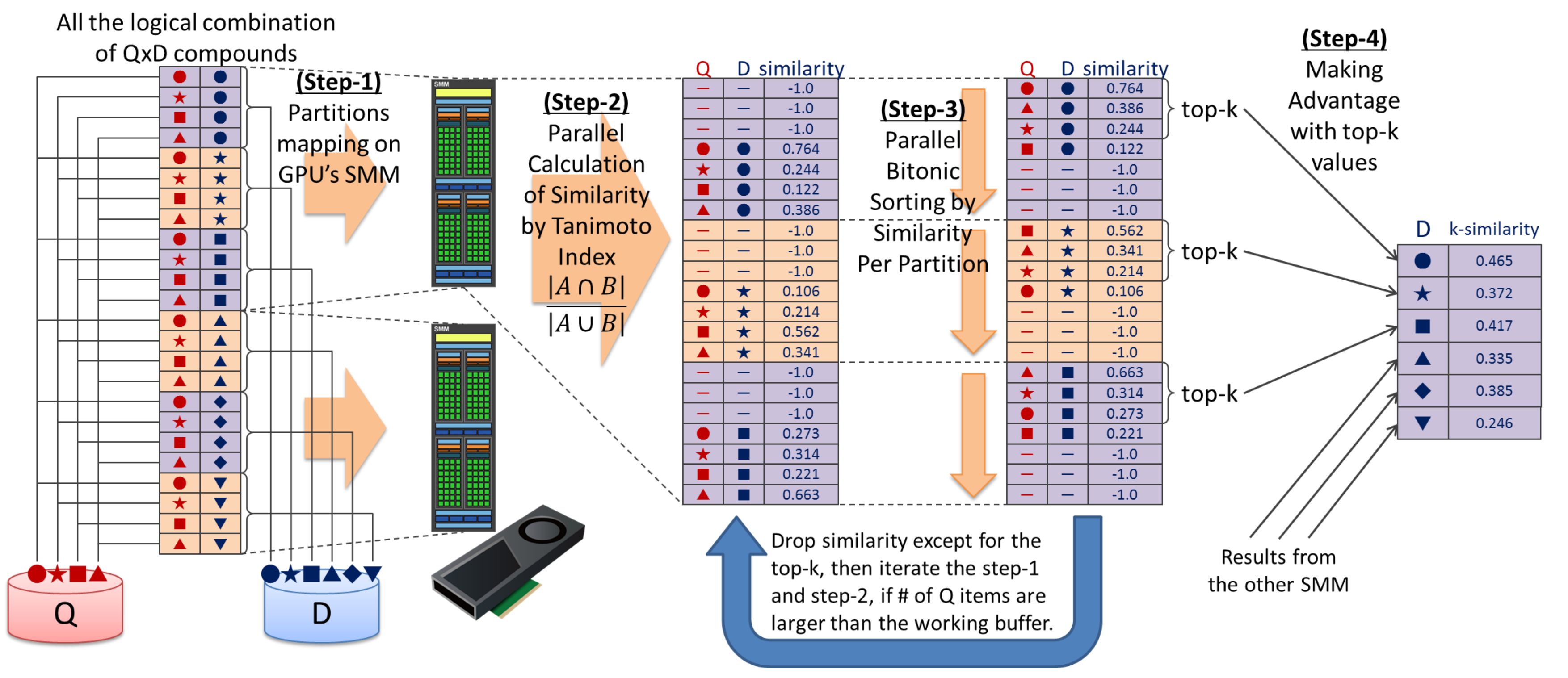
PG-Strom is an extension of PostgreSQL to off-load several class of SQL workloads using GPU automatically or manually.
It has two key features: (1) GPU native binary according to the supplied SQL workloads, and (2) asynchronous execution of the GPU code towards the regular tables.
Right now, it supports WHERE-clause, tables JOIN, GROUP BY and projection under the automatic optimization scenarios.

PL/CUDA is also a feature of PG-Strom for manual optimization scenarios.
The automatic GPU code often takes over inefficiency due to SQL compliance; like NULL checks for each variable reference.
Unlike these automatic GPU code, PL/CUDA allows to implement case-specific optimal GPU programs manually as SQL functions.
Once a PL/CUDA function appears in SQL statement, PG-Strom builds the user-written CUDA code block, sends its arguments, executes CUDA kernel, then writes-back the function results. The steps of invocation are processed by PG-Strom infrastructure, thus, users can focus on the data analytics logic, like k-NN similarity search in our study, to be processed within database systems.

```
CREATE OR REPLACE FUNCTION
knn_gpu_similarity(int,      -- number of the 'k' (k > 0)
                   int[],    -- ID+bitmap of query chemical components
                   int[])    -- ID+bitmap of database chemical components
RETURNS float4[]             -- result: ID(int32) + similarity(fp32)
AS $$
#plcuda_begin
cl_int      k = arg1.value;
MatrixType *Q = (MatrixType *) arg2.value;
MatrixType *D = (MatrixType *) arg3.value;
MatrixType *R = (MatrixType *) results;
cl_float   *values = SHARED_WORKMEM(cl_float);
    :
nloops = (ARRAY_MATRIX_HEIGHT(Q) + (part_sz - k - 1)) / (part_sz - k);
for (loop=0; loop < nloops; loop++)
{
    /* 1. calculation of the similarity */
    for (i = get_local_id();
         i < part_sz * part_nums;
         i += get_local_size())
    {
        j = i % part_sz;    /* index within partition */
        /* index to reference the database matrix (D) */
        dindex = part_nums * get_global_index() + (i / part_sz);
        /* index to reference the query matrix (Q) */
        qindex = loop * (part_sz - k) + (j - k);
        values[i] = knn_similarity_compute(D, dindex, Q, qindex);
    }
}
    :
#plcuda_end
$$ LANGUAGE 'plcuda';
```

The code block #plcuda_begin~#plcuda_end shall be applied to the GPU kernel by CUDA, as is. I/O and data transfer is a PG-Strom's job on behalf of the GPU kernel.

Elements: k-NN Similarity Search


All the logical combination of QxD compounds
(Step-1) Partitions mapping on GPU's SMM
(Step-2) Parallel Calculation of Similarity by Tanimoto Index $\frac{|A \cap B|}{|A \cup B|}$
(Step-3) Parallel Bitonic Sorting by Similarity Per Partition
(Step-4) Making Advantage with top-k values
Q D similarity
top-k
D k-similarity
Drop similarity except for the top-k, then iterate the step-1 and step-2, if # of Q items are larger than the working buffer.
Results from the other SMM
Q
D

Similarity searching based on molecular fingerprint is computationally efficient method for identification of a limited number of candidate compounds against a biological target protein.
k nearest neighbors (k-NN) can be used as a ranking method for similarity searching using multiple reference molecules. In this study, RDKit's Morgan fingerprint with radius 4 and 1024 bits is used as the molecular fingerprint[6] ,and Tanimoto index is used for the similarity determination [7].
Our PL/CUDA version of k-NN similarity search is consist of several steps. First, it splits all the logical combination of Q and D compounds to multiple partitions, then maps a partition on a certain GPU's SM (Streaming Multiprocessor). SM is a unit of execution on GPU, allows to run up to 1024 threads simultaneously. A GPU device usually has multiple SMs(2~56).
Divide-and-conquer is key of performance. Second, it calculates similarity values based on Tanimoto Index . Third, it sorts the combinations by its similarity. Both steps are processes in parallel, but no interactions to external entity of the partition. Last, it picks up the top-k items of the partition, then makes an average value and puts it on the result buffer.
Here is a special optimization for GPU; a choice of partition size. GPU has a special fast memory called "shared memory", not only DRAM. Its capacity is small (~64KB), but offers L1-cache grade latency. Our PL/CUDA function carefully choose the partition size to fit capacity of the shared memory of SM. In the results, it pulled out maximum performance of the device.

Conclusions

Our study shows the infrastructure to run fully-optimized GPU code inside of database system (PG-Strom + PL/CUDA) enables to run advanced data analytic algorithms much faster than CPU version of implementation, nearby the location of data. From the standpoint of data researcher, it eliminates the necessity to export the data-set from the database but allows to fetch the results of heavy calculation portion, and also allows to utilize the flexibility of SQL for data manipulation at pre- and post-process on core of the logic.
We tried to implement k-NN similarity search algorithm using PL/CUDA, towards the 10M of records; constructed from the ChEMBL database. It has x135 times shorter response time in the largest problem scale scenario. Usually, people cannot wait for 3000sec in front of the workstation, but 25sec may be reasonable for try&errors on research.
Visualization of the similarity search is a remained task. And, for further works, we will try to support larger dataset, and applying the technology to other problem area.

[1] KaiGai K, PG-Strom – GPGPU meets PostgreSQL -, PostgreSQL Conference 2015
[2] https://github.com/pg-strom/devel
[3] https://www.postgresql.org/
[4] Schuffenhauer A., Floersheim P., Acklin P., Jacoby E., Similarity metrics for ligands reflecting the similarity of the target protein, *Journal of Chemical Information and Computer Sciences*, 43:391–405, 2003.
[5] Geppert H., Horváth T., Gärtner T., Wrobel S., Bajorath J., Support-vector-machine-based ranking significantly improves the effectiveness of similarity searching using 2D fingerprints and multiple reference compounds, *Journal of Chemical Information and Modeling*, 48:742-746 2008.
[6] http://www.rdkit.org/
[7] Rogers, David J., Tanimoto, Taffee T. A Computer Program for Classifying Plants, *Science,* 132: 1115–1118, 1960.