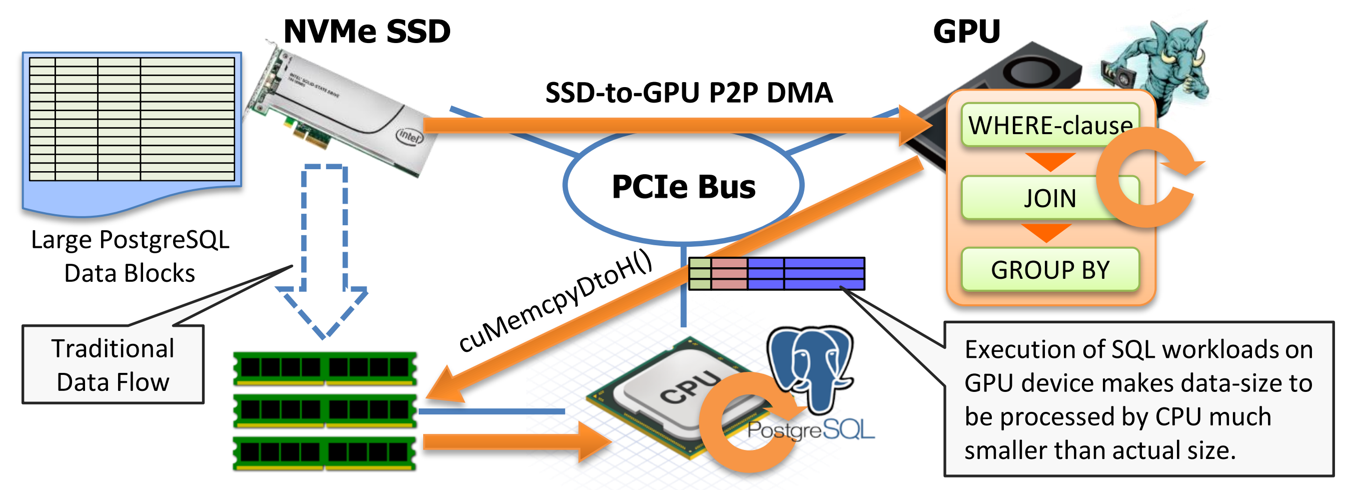
An intelligent storage for PostgreSQL database

KaiGai Kohei <kaigai@kaigai.gr.jp>
The PG-Strom Development Team

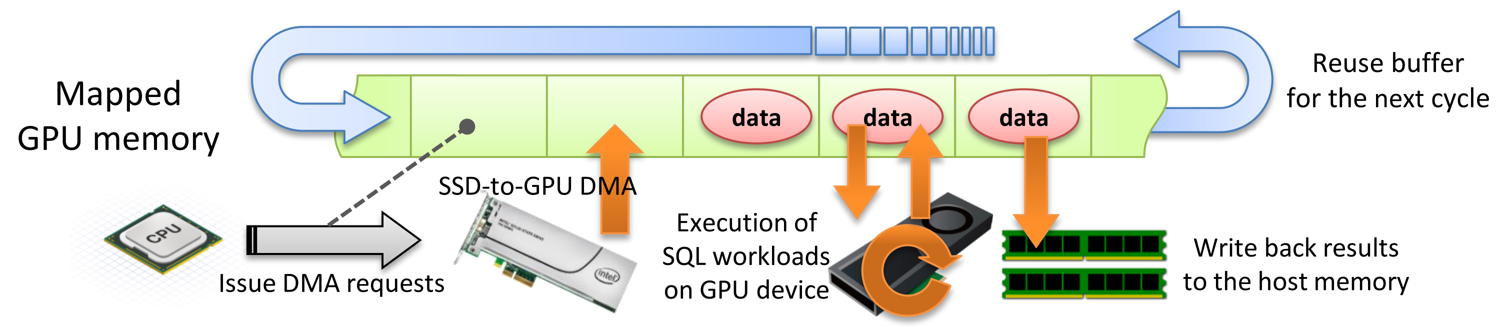
Abstract

GPUDirect RDMA allows peer-to-peer data load from PCIe devices to GPU RAM directly.
Our extension modules for Linux kernel and PostgreSQL collaboratively utilizes this infrastructure for very fast table scan, by P2P loading of database blocks on NVMe-SSD to GPU RAM, and SQL execution on GPU devices. Once data blocks get loaded on GPU RAM, kernel function works to reduce data size according to the supplied SQL (WHERE-clause, JOIN and GROUP BY). In the results, CPU/RAM will receive much smaller data size than actual table size, and it looks like storage performs intelligently with understanding of SQL.
According to SQL star-schema based benchmark, our feature enabled to scan 351GB flat table in 80sec; that is about 4.5GB/s query processing throughput and 2.5 times faster than usual filesystem base I/O implementation. This result shows GPU is also valuable for I/O intensive workloads, not only calculation intensive workloads.

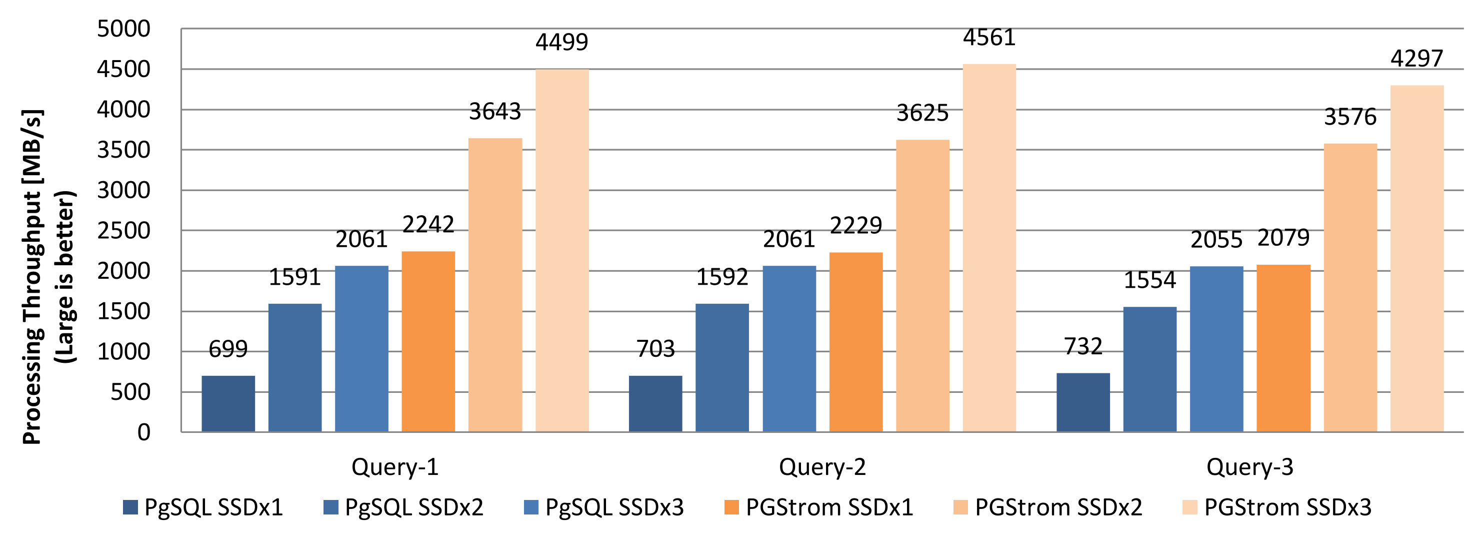
Brief Architecture


NVMe SSD
GPU
SSD-to-GPU P2P DMA
PCIe Bus
WHERE-clause
JOIN
GROUP BY
Large PostgreSQL Data Blocks
Traditional Data Flow
cuMemcpyDtoH()
CPU
PostgreSQL
Execution of SQL workloads on GPU device makes data-size to be processed by CPU much smaller than actual size.

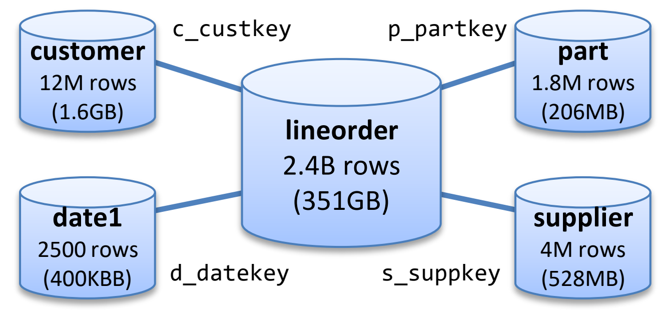
OLAP (online analytical processing) is one major workload in database area. Its characteristics is that query intends to generate (relatively) small summary from massive amount of data; usually bigger than physical RAM size.
Traditionally, data blocks of the database on storage device (like SSDs) are loaded to RAM of host system once, then CPU or GPU processes them. However, it is not efficient so much because most of rows in data blocks are filtered out or referenced only once during the reduction process for GROUP BY operations.
In case of SQL workloads, we can exactly know how data blocks shall be processed next to the loading. So, it is available to design GPU kernels to run SQL workloads on multiple blocks/rows in parallel. We implemented GPU version of WHERE-clause evaluation, Hash-Join and GROUP BY, as a functionality of PG-Strom[*1].
In addition, we developed a Linux kernel module that intermediates DMA transfer from SSD blocks to GPU RAM. PG-Strom is an extension of PostgreSQL [*2] database. It controls both of the Linux kernel module and GPU device to off-load SQL workloads.
This architecture allows a couple of SSDs and GPU co-work and send back pre-processed data stream to the host system according to the supplied SQL. From the standpoint of application, **it looks like the storage gets intelligence to run some of SQL workloads on device**.
Keys to handle large data with GPU RAM (= relatively rare) are circular buffer and asynchronous execution. We split mapped GPU memory into multiple chunks, and assigns job piece by piece. At a particular moment, CPU issues a SSD-to-GPU DMA request on a chunk, prior requests on another chunk is in-progress, GPU kernel is running to process another chunk, and usual GPU-to-RAM write back DMA is in-progress on another chunk.
It works very efficient to pull out hardware capability both of GPU and NVMe-SSDs.


Mapped GPU memory
data
data
data
Reuse buffer for the next cycle
SSD-to-GPU DMA
Execution of SQL workloads on GPU device
Write back results to the host memory
Issue DMA requests
CPU

Benchmarks



Data structure for star schema benchmarking (SF=401)



```
(Query-3)
SELECT c_city, s_city, d_year, sum(lo_revenue)
  FROM customer, lineorder, supplier, date1
 WHERE lo_custkey = c_custkey
   AND lo_suppkey = s_suppkey
   AND lo_orderdate = d_datekey
   AND (c_city='UNITED KI1' or c_city='UNITED KI5')
   AND (s_city='UNITED KI1' or s_city='UNITED KI5')
   AND d_yearmonth = 'Dec1997'
 GROUP BY c_city, s_city, d_year
 ORDER BY d_year asc, revenue desc;
```

We construct a database with above data structure on raw NVMe-SSD [*3], and md-raid0 volume which consists of 2 or 3 SSD devices. This workload is well known star schema benchmark that reflects typical DWH workloads, also similar to some of IoT class data structures.
We run some reporting queries like Query-3 above. The largest table size was 351GB but RAM size was 128GB. Therefore, this workload is typical I/O intensive workloads rather than calculations.
The blue bars are results of vanilla PostgreSQL for each RAID configurations, and orange bars are results of PG-Strom with SSD-to-GPU P2P DMA support. An inverse number of the query processing time (e.g, if query finished in 80sec, its data processing is 351GB / 80.0s = 4492MB/s).
It shows larger number of SSD devices makes data processing throughput higher. On the other hands, the results of PG-Strom shows SSD-to-GPU P2P DMA pulls out much higher capability of NVMe-SSDs than filesystem + CPU based data processing. Especially, "PG-Strom SSDx1" is a results on raw NVMe-SSD device. Its data processing throughput is about 2.1-2.2GB/s; which is almost equivalent to the catalog spec of the SSD device (2.2GB/s in SeqRead).
These results introduce us the recent semiconductor innovations (GPU, SSD, ...) allows us to catch up high-end DWH grade performance with much cheaper cost than before.

Conclusion & Future

Our challenge shows a pair of GPU and NVMe-SSD can perform as like an intelligent storage from the standpoint of application. GPU can process the data stream according to the knowledge of application side, but prior to the data arriving at CPU/RAM, with support of GPUDirect RDMA. It pulls up data processing throughput nearby the theoretical limitation of the hardware, but relatively small cost than existing high-end database solution.
Right now, our implementation still has unignorable overhead on md-raid0 configuration. It needs to be revised. Then, we will try 10GB/s data processing throughput per single computing node.

[*1] PG-Strom – Extension of PostgreSQL for GPU acceleration, http://strom.kaigai.gr.jp/
[*2] PostgreSQL – Well used open source RDBMS, https://www.postgresql.org/
[*3] Intel SSD 750 (400GB) – http://www.intel.com/content/www/us/en/solid-state-drives/solid-state-drives-750-series.html