Technology Updates of PG-Strom at Aug-2014

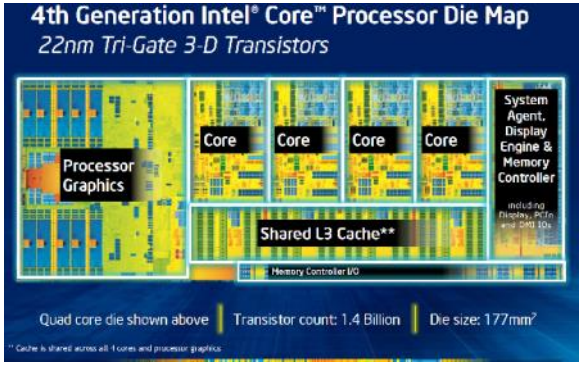
KaiGai Kohei <kaigai@kaigai.gr.jp>
(Tw: @kkaigai)

Background: Evolution of Semiconductors

PG-Strom intends to be a pioneer to introduce the semiconductor's new trend into the world of RDBMS

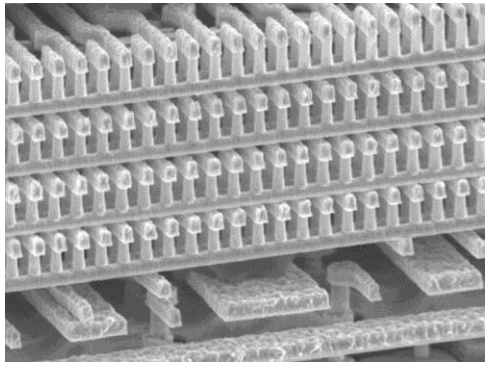
Processor

All major vendor drives to heterogeneous architecture because of power consumption and thermal problem.



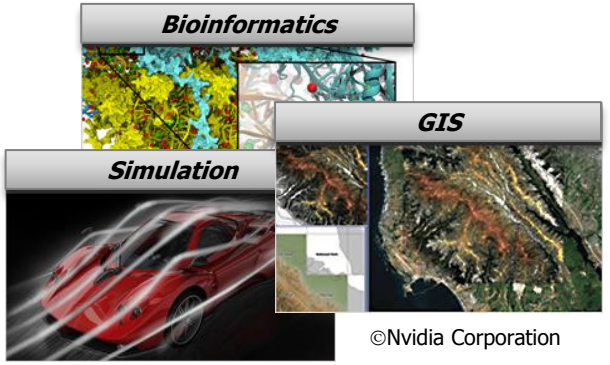
Memory

Stacked DRAM allows to keep the trend of memory capacity expansion on the next several years.



GPU

GPU, originated from graphics accelerator, expands supporting workloads. Nowadays, it becomes leading player in HPS region.



©Nvidia Corporation

How Software will evolve on the next generation hardware?

- ✓ Utilization of GPU computing capability is the key of performance.
- ✓ Data shall be deployed on semiconductor memory, rather than magnetic disk.
- ✓ GPU is well validated long-standing technology, ready for enterprise usage.

Overview of PG-Strom (1/2) – Basic idea

SELECT * FROM table WHERE sqrt((x-256)^2 + (y-100)^2) < 10;

Auto generation of GPU-code from user given SQL statement, then just-in-time compile

code for GPU/MIC

OpenCL runtime compiler

OpenCL

GPU/MIC device

vanilla PostgreSQL

CPU

Iteration of row-fetch and evaluation of WHERE clause

PostgreSQL + PG-Strom

CPU

GPU

Async DMA send & Async Kernel exec

Synchronization

Fetch multiple rows at once

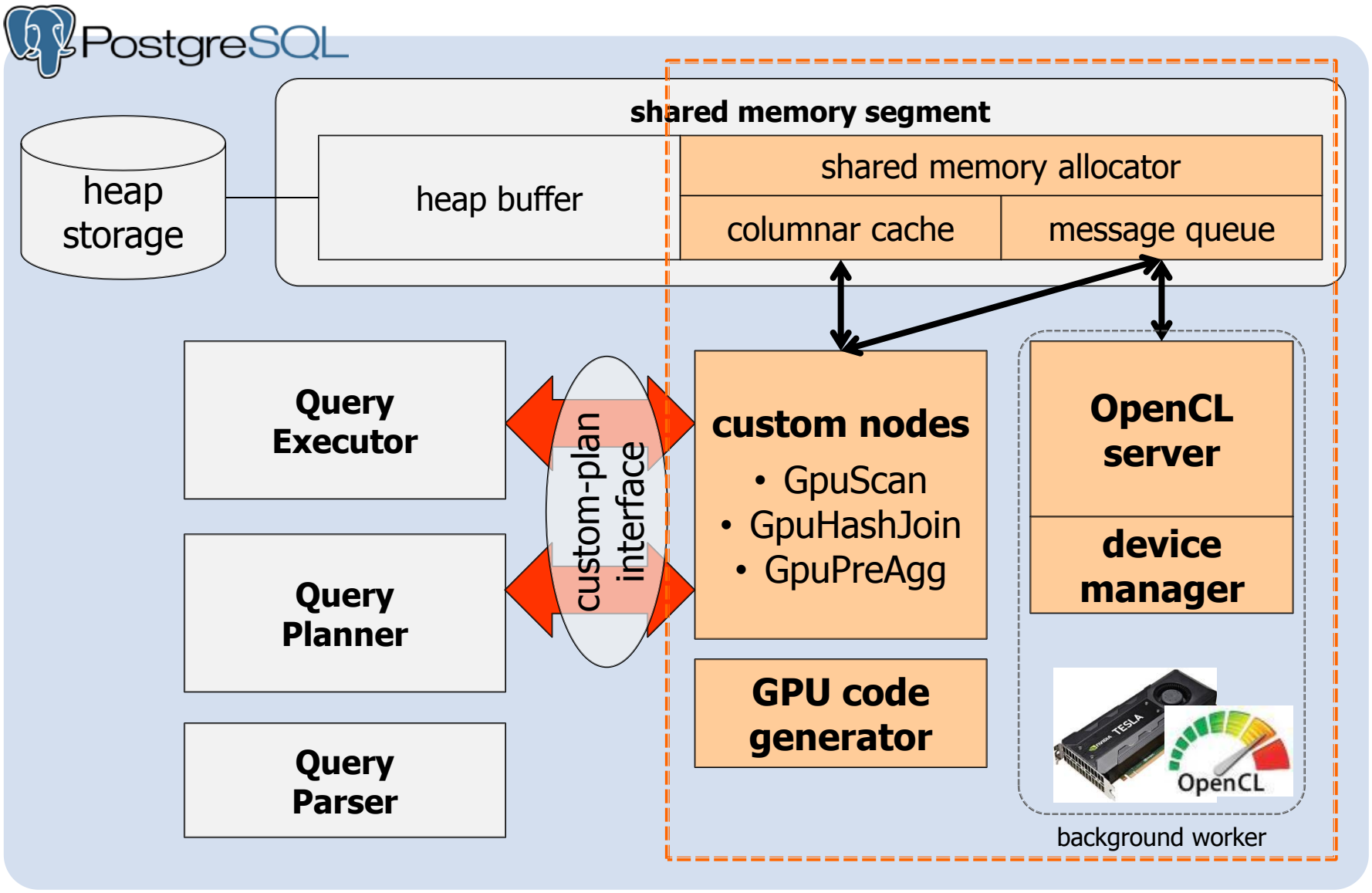
Shorter response time than CPU execution

: Fetch a row from shared buffer

: evaluation of WHERE clause

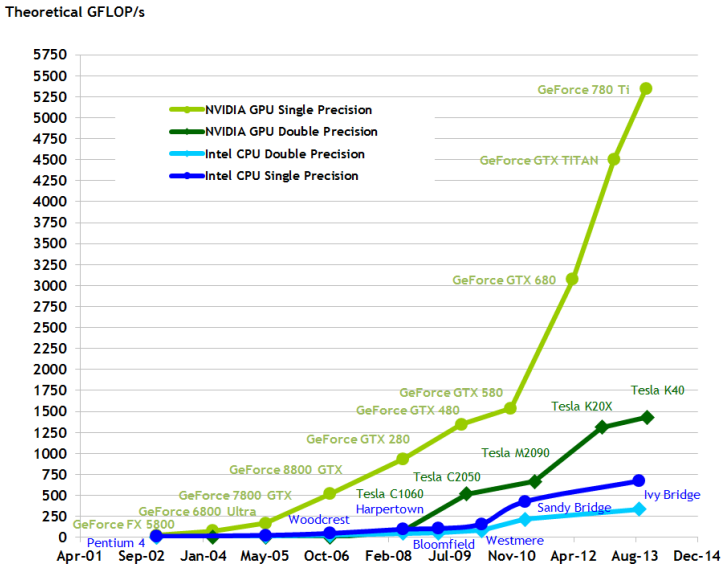
Due to parallel execution, time to evaluate gets shorten

Overview of PG-Strom (2/2) – SW Architecture



[FYR] Characteristics of GPU

Floating-Point Operations per Second for the CPU and GPU



The GPU Devotes More Transistors to Data Processing

CPU allocates more transistors for rich-functional ALUs and cache, GPU allocates more transistors for simple ALUs.

	CPU	GPU
model	Xeon E5-2690v2	Nvidia Tesla K20
generation	Q3-2013	Q4-2014
# of cores	10	2476
core clocks	3.0GHz	706MHz
code set	x86_64 (functional)	nv-kepler (simple)
peak flops (single/double)	480GFlops / 240GFlops	3.52TFlops / 1.17TFlops
memory size	up to 768GB	5GB
memory band	59.7GB/s	208GB/s
TDP	130W	225W
price	$2100	$2700

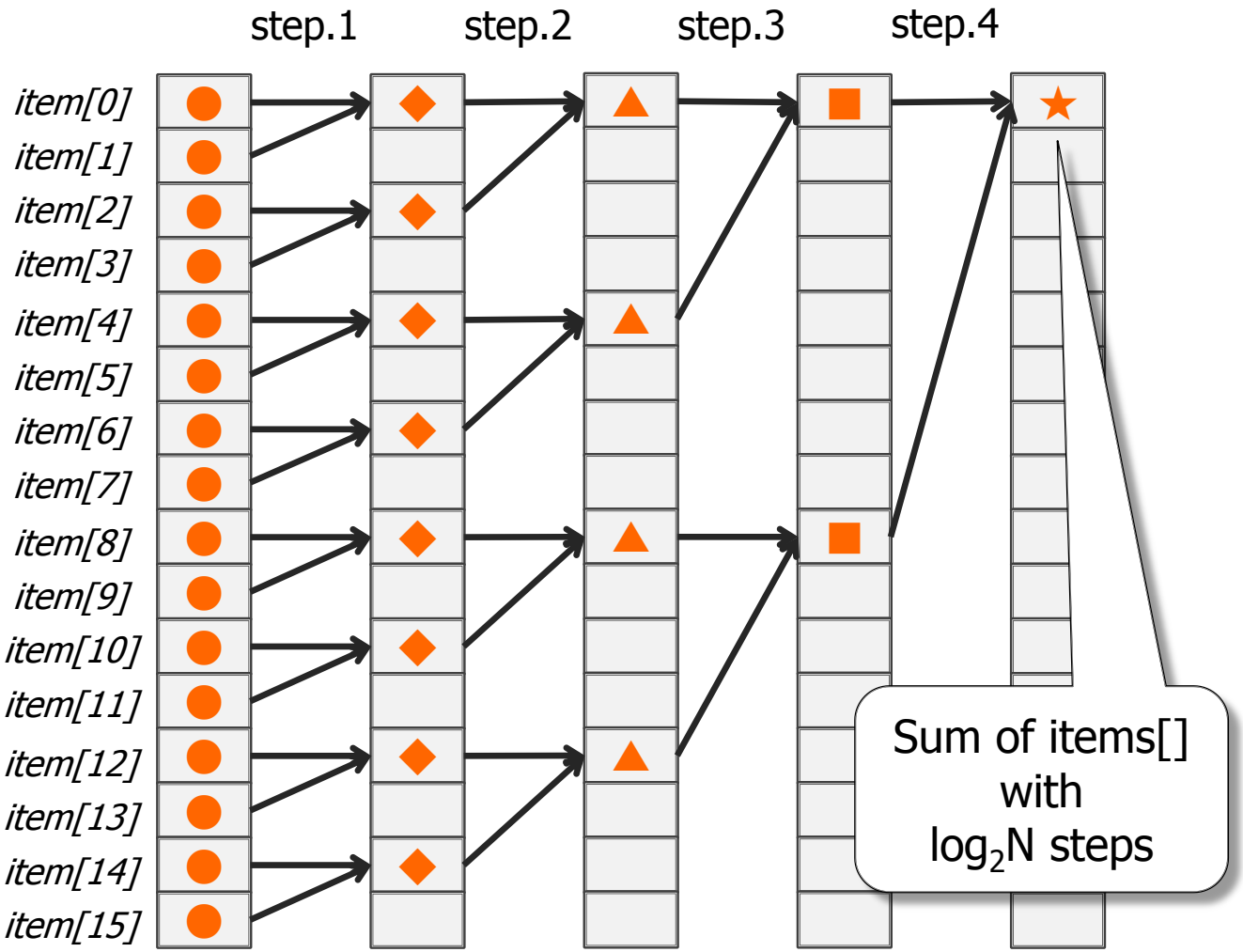
[FYR] How GPU processes the data

Reduction algorithm well fits aggregate functionality of RDBMS

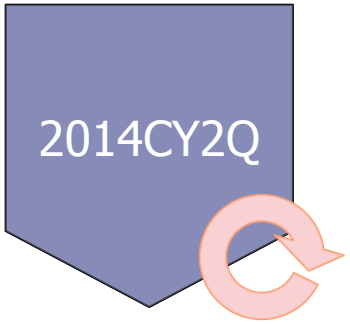
Calculation of

$\Sigma_{i=0...N-1}$item[i]

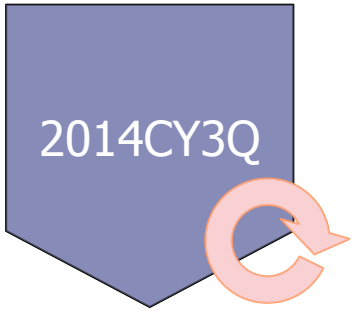
with N of GPU cores



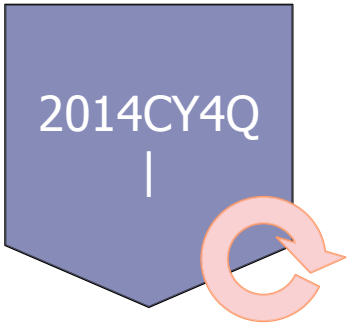
Status of the development (1/2)



- **Working example** – It demonstrates GPU acceleration works in RDBMS has a significant performance advantages.
- Full-scan, Hash-joining, Sorting were implemented
- x3~x23 times faster query response time than vanilla PostgreSQL
- **Things we learned** – reduction of rows to be processed by CPU is the key of higher performance



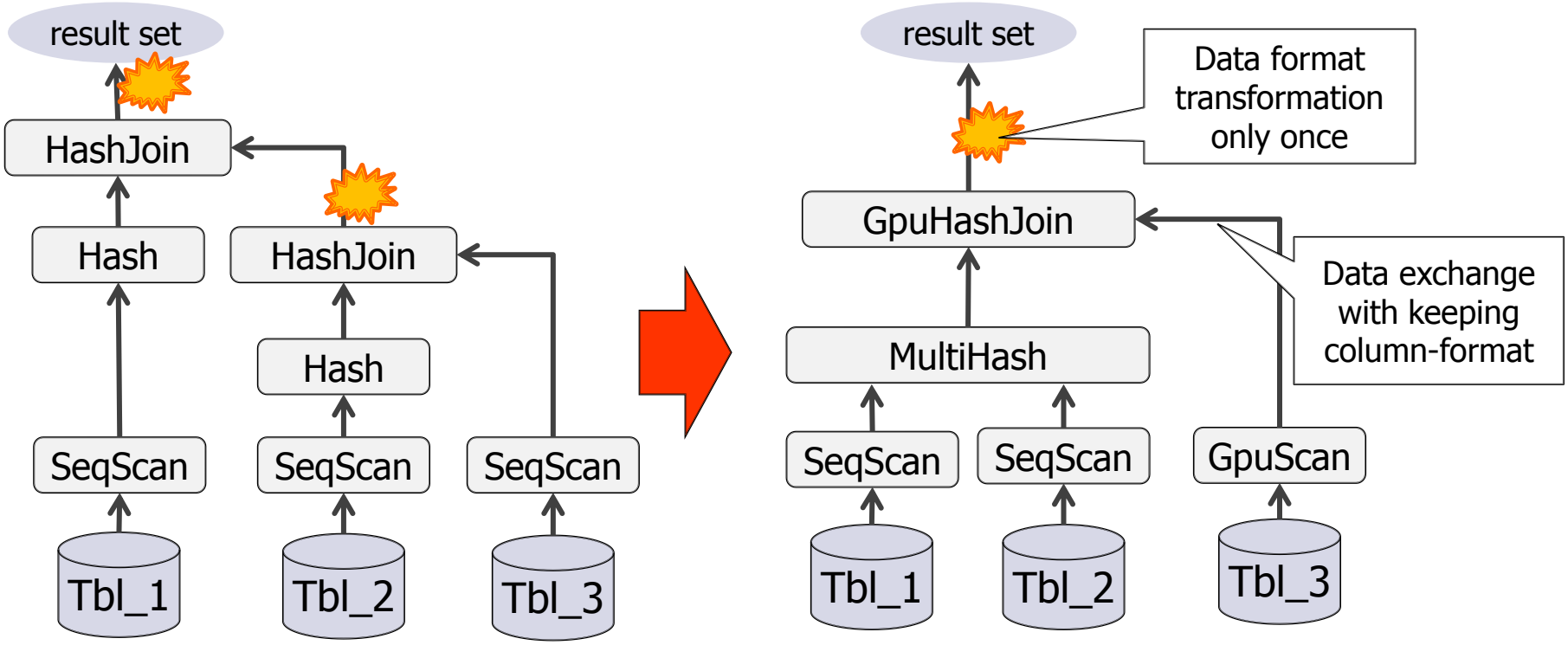
- **Beta development** – It develops minimum valuable product for evaluation purpose.
- Full-scan, Hash-join, Aggregation will be supported
- **Things we are learning** – materialization is still expensive, columnar-cache leads over-consumption of shared memory



- **Public beta** – It tries to gather potential use cases that can utilize PG-Strom to tackle people's problem.
- More functionalities may be implemented depending on the feedbacks.
- **Things we want to learn** – what kind of use-case may be expected, who can be prospective customers.

Works in progress (1/2) – Multi tables Hash-Join

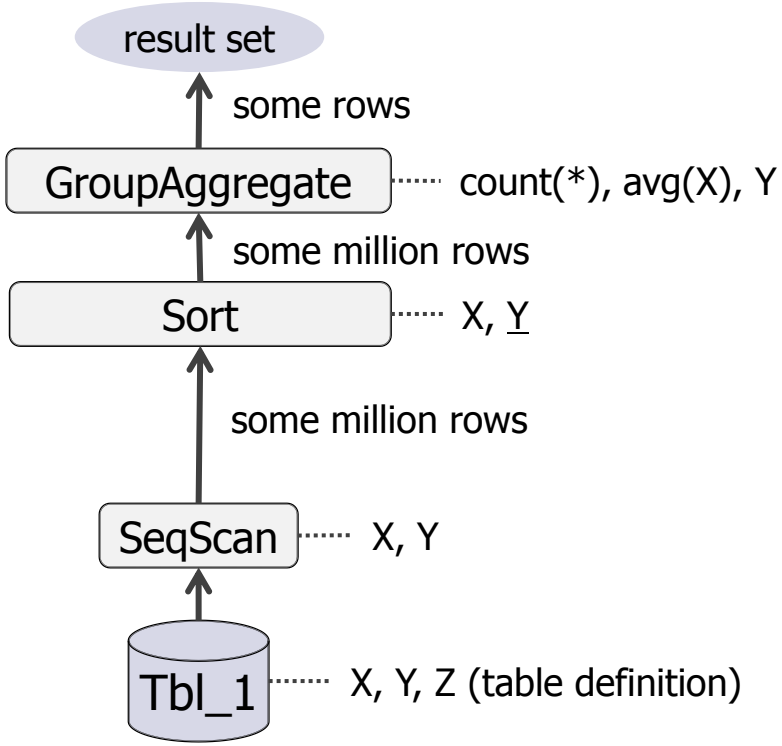
SELECT * FROM Tbl_1, Tbl_2, Tbl_3 WHERE .....;

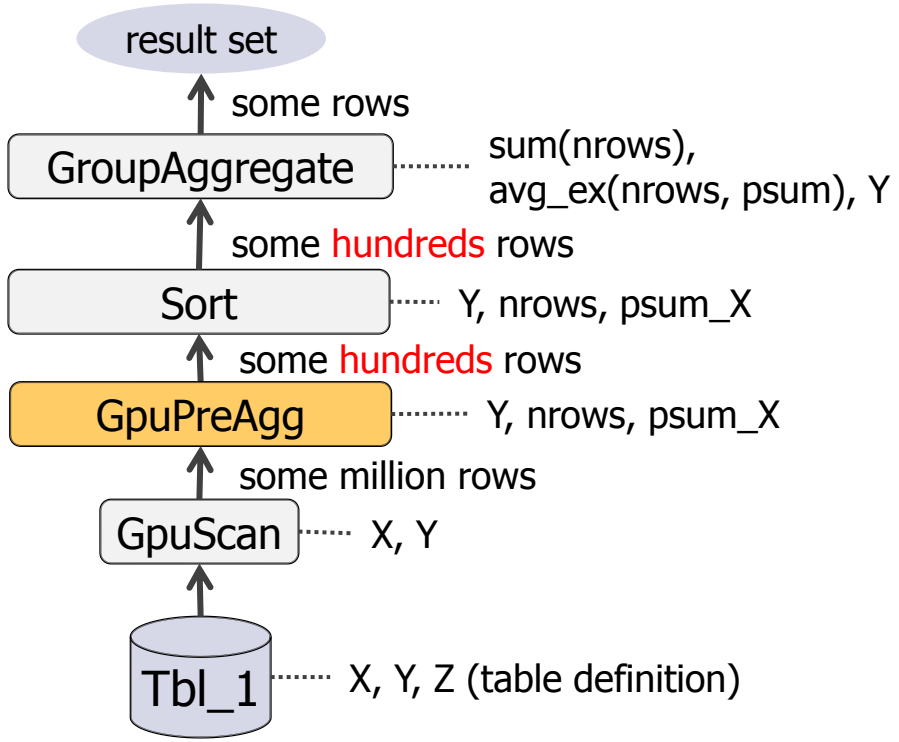


- Tuple materialization was a key factor of performance bottle-neck
  - ➔ minimizing number of tuple materialization makes sense
- GpuHashJoin loads multiple tables onto a hash-table then joins at once.
  - ➔ materialization will happen only once, even more than 3 tables are joined

Works in progress (2/2) – Aggregate Reduction

SELECT count(*), AVG(X), Y FROM Tbl_1 GROUP BY Y;

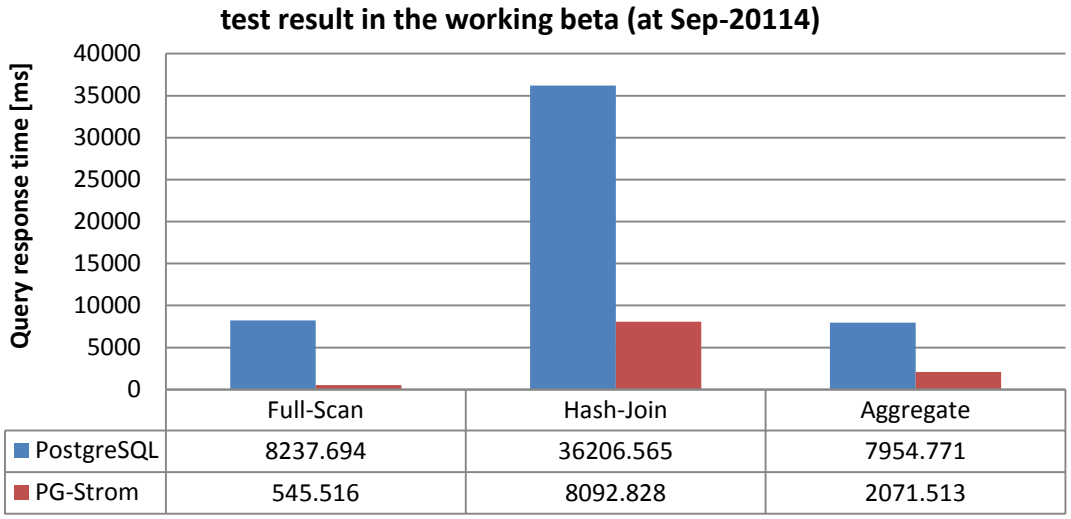




- GpuPreAgg intends to reduce number of rows to be processed by CPU
- GPU's DRAM has less capable than CPU's one, so not a good idea to run global sort, but local grouping and reduction is it's best fit.
- Due to soring algorithm, time to handle 1/N data size less than 1/N.

Status of the development (2/2)

- Jul-2014 – A test implementation
  - Technical validation of GPU acceleration on RDBMS
  - Full-scan, Hash-Join, Sorting
  - ➔ x3~x23 times faster than vanilla PostgreSQL
- Nov-2014 – A working beta
  - Target of evaluation on a series of PoC efforts
  - Full-scan, Hash-Join, Aggregate



	Full-Scan	Hash-Join	Aggregate
PostgreSQL	8237.694	36206.565	7954.771
PG-Strom	545.516	8092.828	2071.513

Test Environment
Server: NEC Express5800 HR120b-1
CPU: Xeon(R) CPU E5-2640 x2
RAM: 256GB
GPU: NVidia GTX750 Ti (640 cuda cores)

Each query run on a table with 20M records. Hash-join workload joins 3 other tables with 40K rows.
Query response time comes from "execution time" in EXPLAIN ANALYZE


ご清聴ありがとうございました