PG-Strom
~GPGPU meets PostgreSQL~

NEC Business Creation Division
The PG-Strom Project
KaiGai Kohei <kaigai@ak.jp.nec.com>

About me

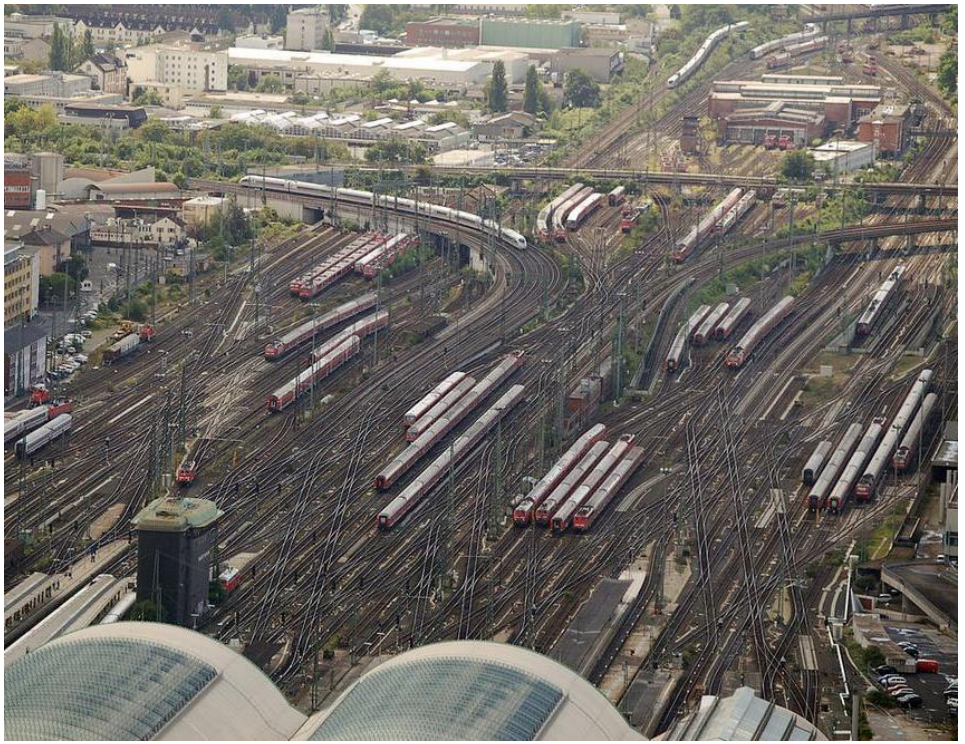
Who are you

- Name: KaiGai Kohei
- Works: NEC
- Roles:
  - development of software
  - development of business
- Past contributions:
  - SELinux integration (sepgsql) and various security stuff
  - Writable FDW & Remote Join Infrastructure
  - ...and so on

About PG-Strom project

- The 1st prototype was unveiled at Jan-2012, based on personal interest
- Now, it became NEC internal startup project.

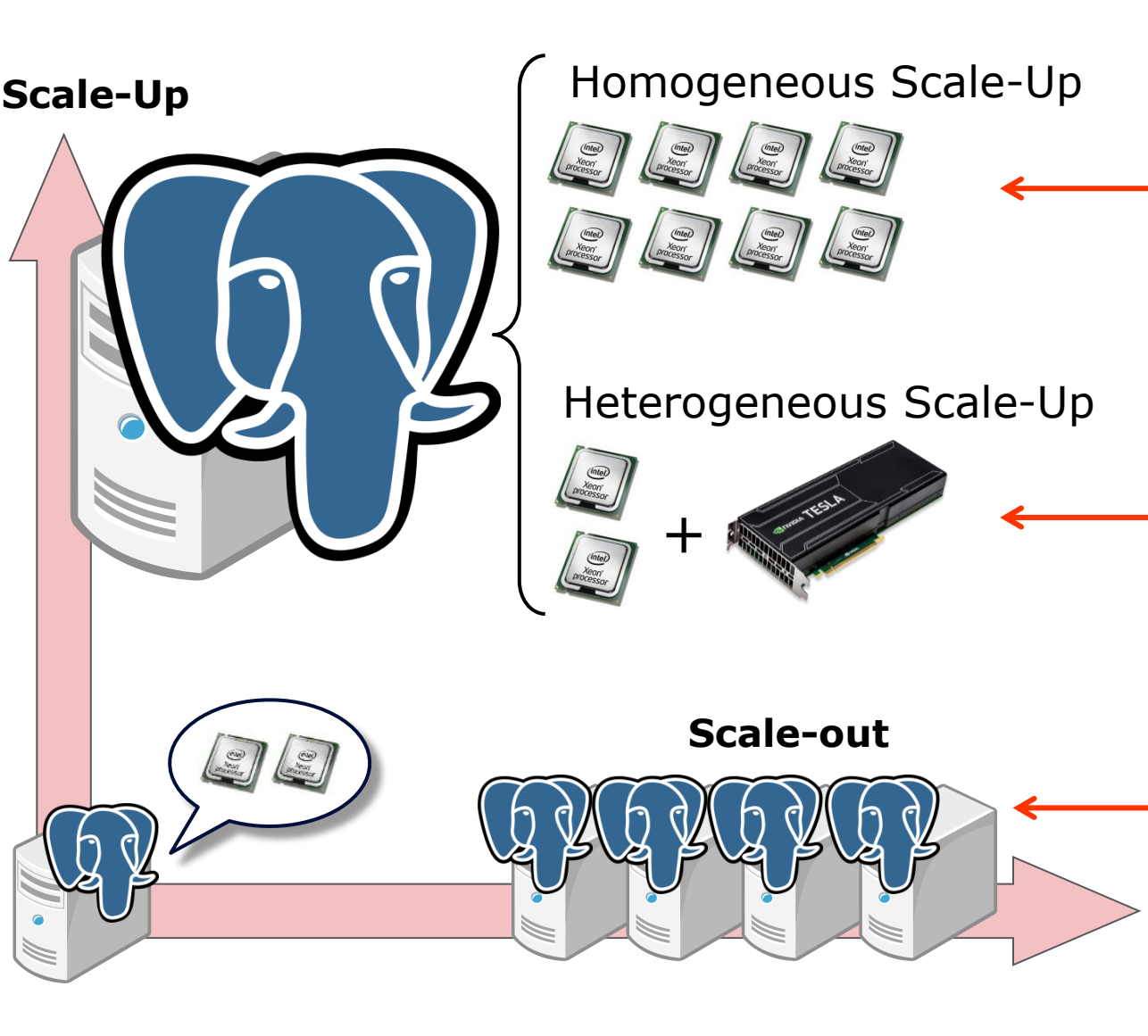
Parallel Database is fun!



- Growth of data size
- Analytics makes values hidden in data
- Price reduction of parallel processors

➔All the comprehensives requires database be parallel

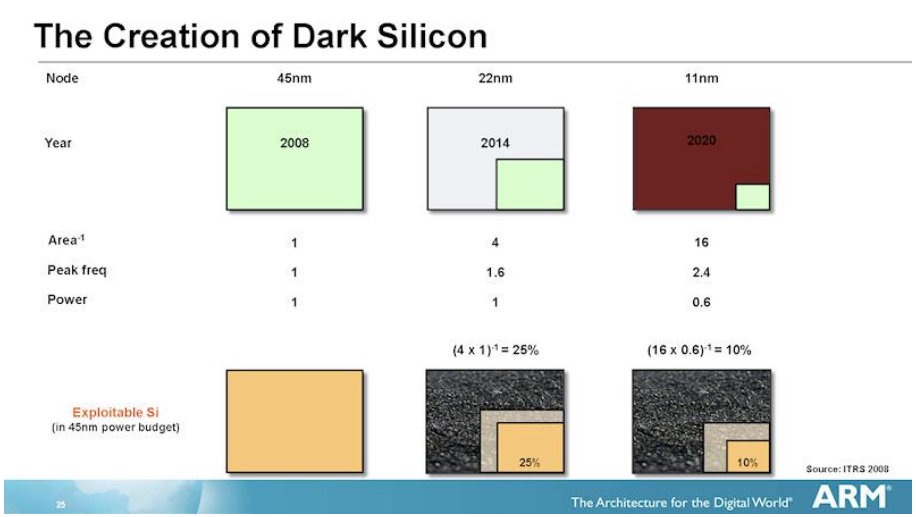
Approach to Parallel Database



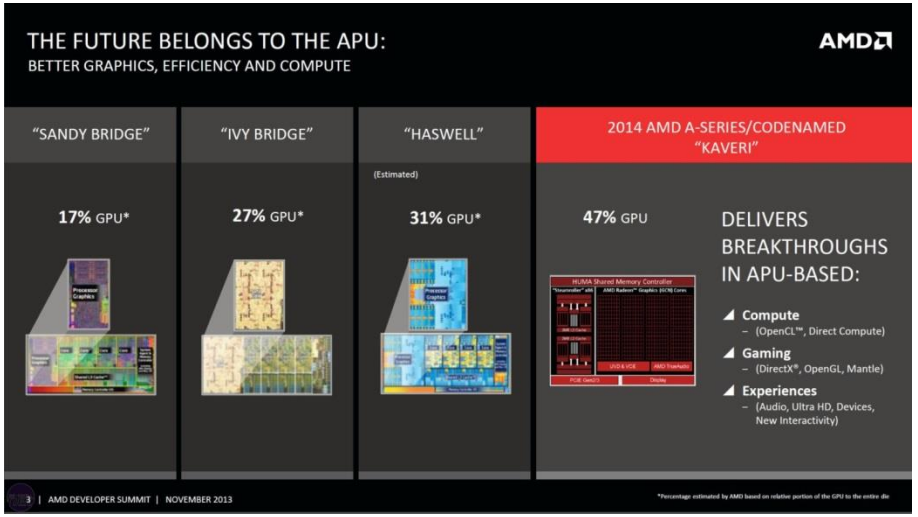
12:45
13:00 **Parallel Sequential Scan**
Unleashing a heard of elephants
Amit Kapila , **Robert Haas**
Track **Hacking**
13:15
13:30
13:45
14:00 **PG-Strom**
GPGPU meets PostgreSQL to accelerate analytic queries
KaiGai Kohei
Track **Performance**
14:15
14:30
14:45
15:00 **pg_shard: Shard and scale out PostgreSQL**
PostgreSQL extension to scale out real-time reads and writes
Ozgun Erdogan
Track **Scaling Out**
15:15
15:30

Why GPU?

No Free Lunch for Software, by Hardware



SOURCE: Compute Power with Energy-Efficiency, Jem Davies, at AMD Fusion Developer Summit 2011



SOURCE: THE HEART OF AMD INNOVATION, Lisa Su, at AMD Developer Summit 2013

- Power consumption & Dark silicon problem
- Heterogeneous architecture
- Software has to be designed to pull out full capability of the modern hardware

Features of GPU (Graphic Processor Unit)

SOURCE: CUDA C Programming Guide

- Massive parallel cores
- Much higher DRAM bandwidth
- Better price / performance ratio
- Advantage
  - Simple arithmetic operations
  - Agility in multi-threading
- Disadvantage
  - complex control logic
  - no operating system

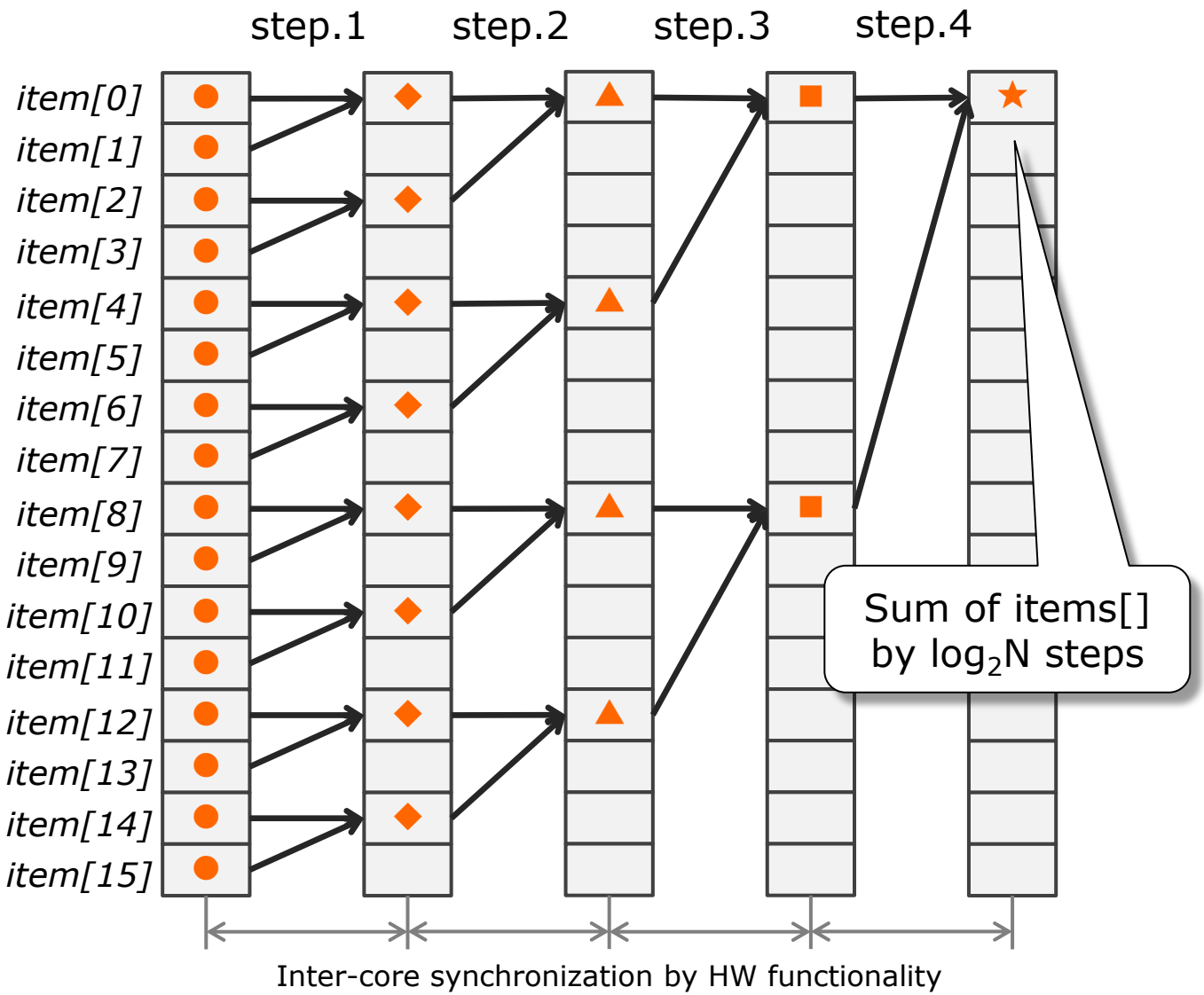
	GPU	CPU
Model	Nvidia GTX TITAN X	Intel Xeon E5-2690 v3
Architecture	Maxwell	Haswell
Launch	Mar-2015	Sep-2014
# of transistors	8.0billion	3.84billion
# of cores	**3072 (simple)**	12 (functional)
Core clock	1.0GHz	2.6GHz, up to 3.5GHz
Peak Flops (single precision)	6.6TFLOPS	998.4GFLOPS (with AVX2)
DRAM size	12GB, GDDR5 (384bits bus)	768GB/socket, DDR4
Memory band	**336.5GB/s**	68GB/s
Power consumption	250W	135W
Price	$999	$2,094

How GPU cores works

Calculation of

$$\sum_{i=0\ldots N-1} item[i]$$

with GPU cores



What is PG-Strom (1/2) – Core ideas

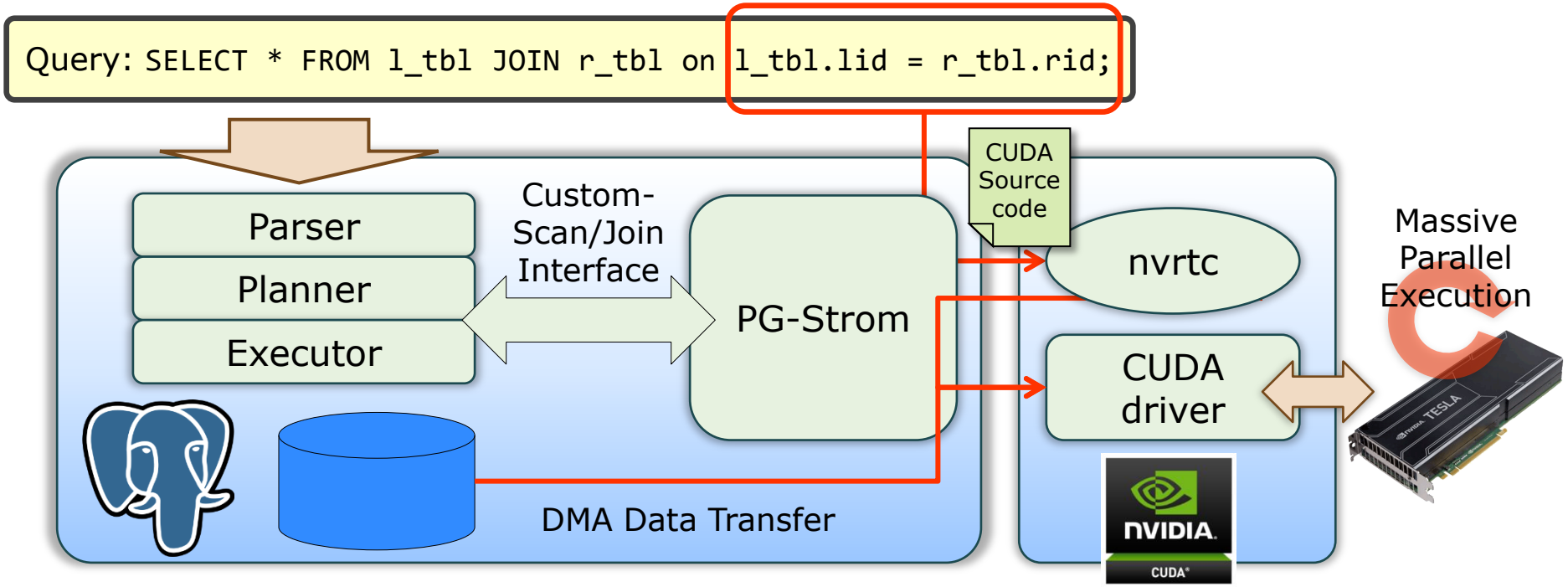
Core idea

① GPU native code generation on the fly

② Asynchronous execution and pipelining

Advantage

- Transparent acceleration with 100% query compatibility
- Heavy query involving relations join and/or aggregation



What is PG-Strom (2/2) – Beta functionality at Jun-2015

Logics

- GpuScan ... Simple loop extraction by GPU multithread
- GpuHashJoin ... GPU multithread based N-way hash-join
- GpuNestLoop ... GPU multithread based N-way nested-loop
- GpuPreAgg ... Row reduction prior to CPU aggregation
- GpuSort ... GPU bitonic + CPU merge, hybrid sorting

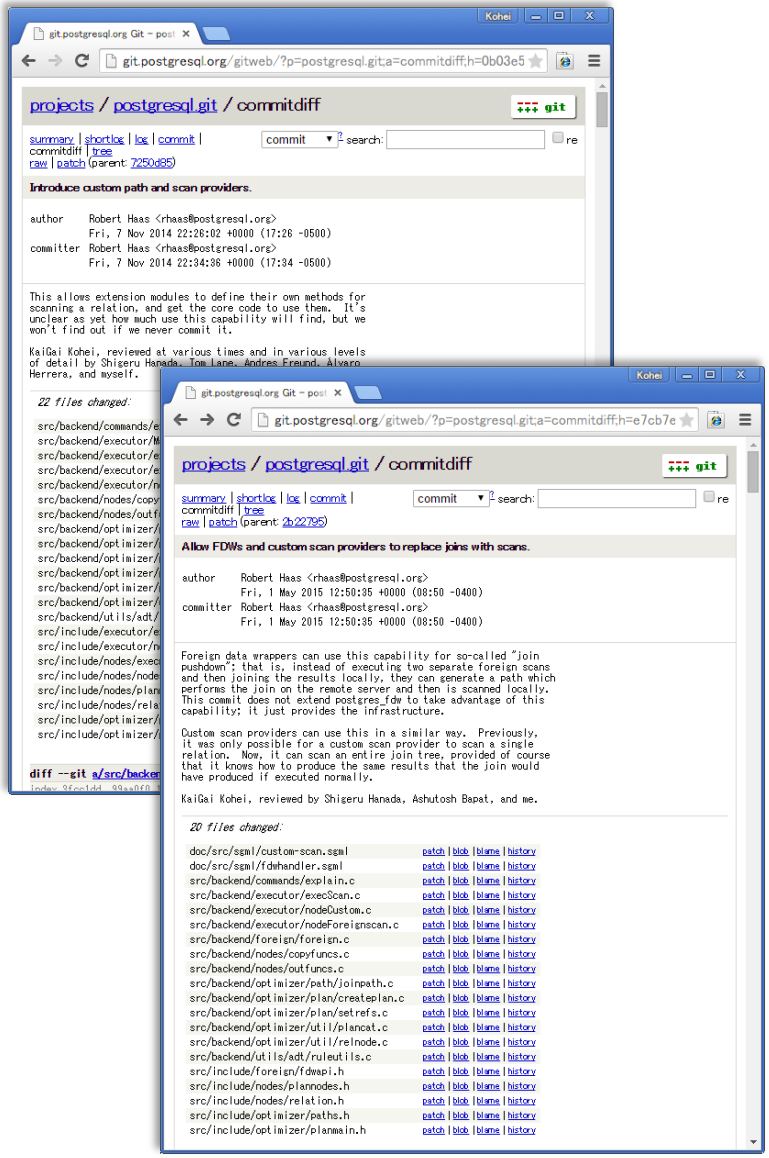
Data Types

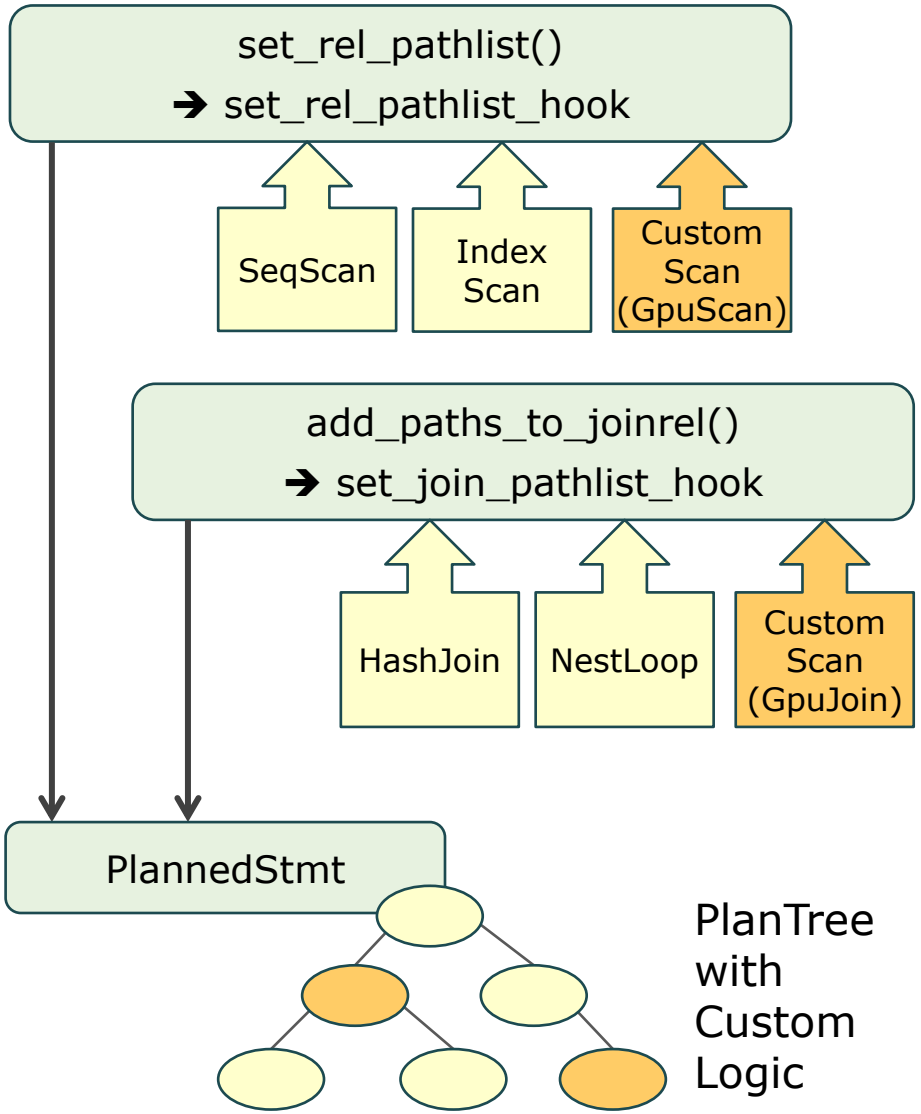
- Numeric ... int2/4/8, float4/8, numeric
- Date and Time ... date, time, timestamp, timestamptz
- Text ... Only uncompressed inline varlena

Functions

- Comparison operator ... <, <=, !=, =, >=, >
- Arithmetic operators ... +, -, *, /, %, ...
- Mathematical functions ... sqrt, log, exp, ...
- Aggregate functions ... min, max, sum, avg, stddev, ...

CustomScan Interface (v9.5 new feature)





GPU code generation and JIT compile

```
postgres=# SELECT cat, AVG(x) FROM t0
             WHERE sqrt((x-20)^2 + (y-10)^2) < 5
             GROUP BY cat;
```

CUDA runtime compiler
(nvrtc; CUDA7.0 or later)

`nvrtcCompileProgram(...)`

.ptx
GPU binary

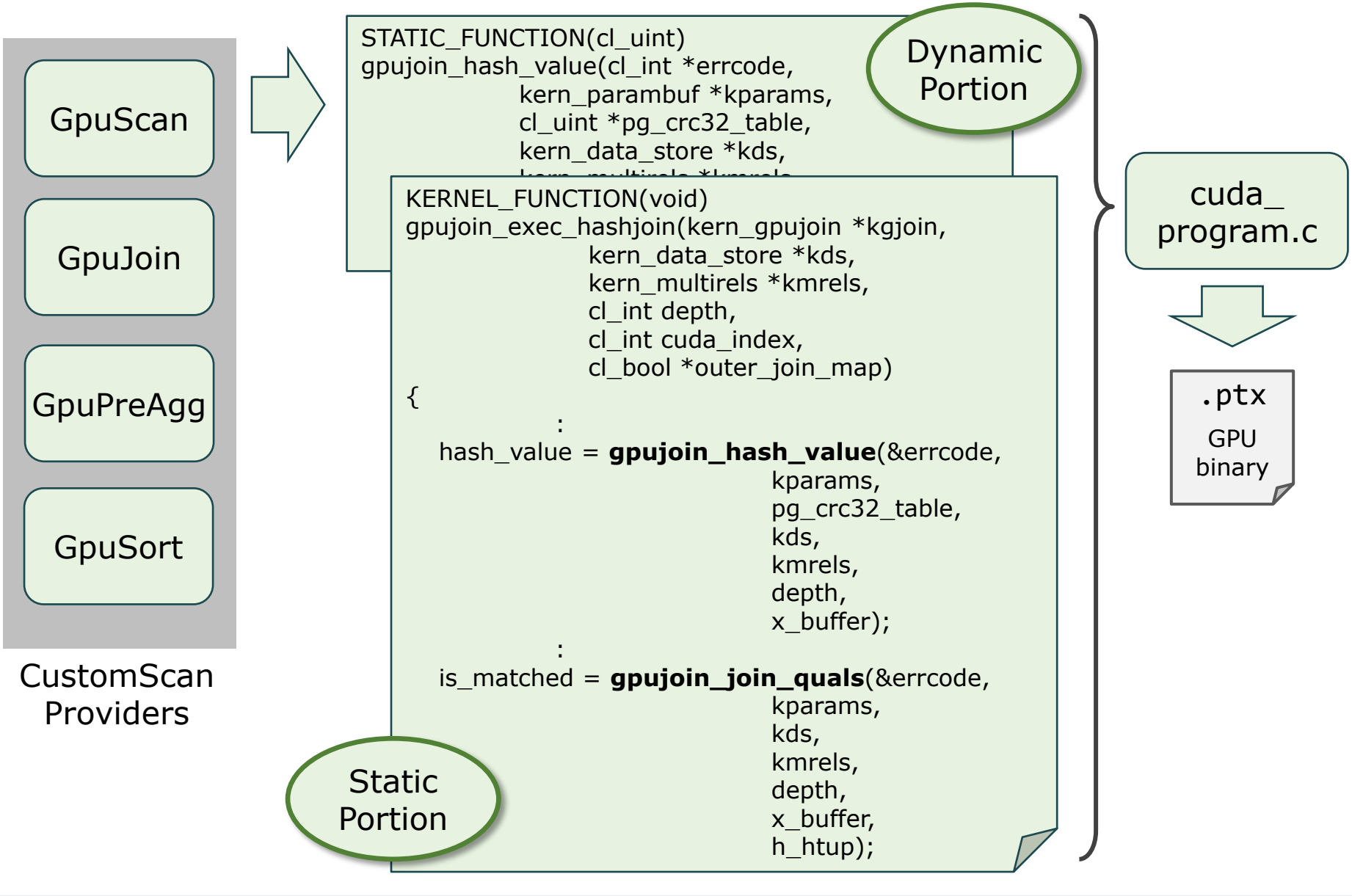
CUDA runtime

Massive Parallel Execution

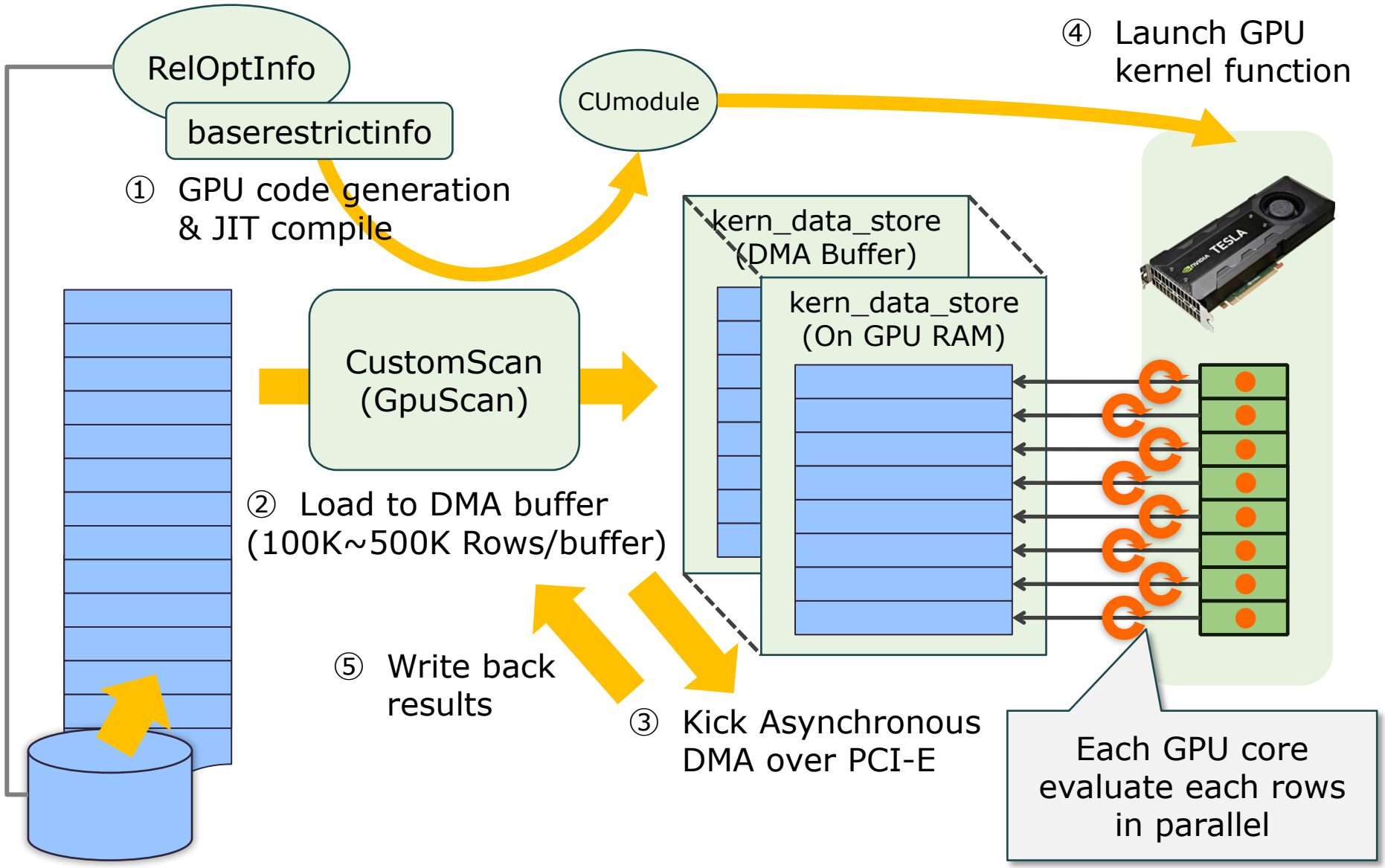
```
STATIC_FUNCTION(bool)
gpupreagg_qual_eval(cl_int *errcode,
                    kern_parambuf *kparams,
                    kern_data_store *kds,
                    kern_data_store *ktoast,
                    size_t kds_index)
{
  pg_float8_t KPARAM_1 = pg_float8_param(kparams,errcode,1);
  pg_float8_t KPARAM_2 = pg_float8_param(kparams,errcode,2);
  pg_float8_t KPARAM_3 = pg_float8_param(kparams,errcode,3);
  pg_float8_t KPARAM_4 = pg_float8_param(kparams,errcode,4);
  pg_float8_t KPARAM_5 = pg_float8_param(kparams,errcode,5);
  pg_float8_t KVAR_8 = pg_float8_vref(kds,errcode,7,kds_index);
  pg_float8_t KVAR_9 = pg_float8_vref(kds,errcode,8,kds_index);

  return EVAL(pgfn_float8lt(errcode,pgfn_dsqrt(errcode,
pgfn_float8pl(errcode, pgfn_dpow(errcode, pgfn_float8mi(errcode,
KVAR_8, KPARAM_1), KPARAM_2), pgfn_dpow(errcode,
pgfn_float8mi(errcode, KVAR_9, KPARAM_3), KPARAM_4))),
KPARAM_5));
}
```

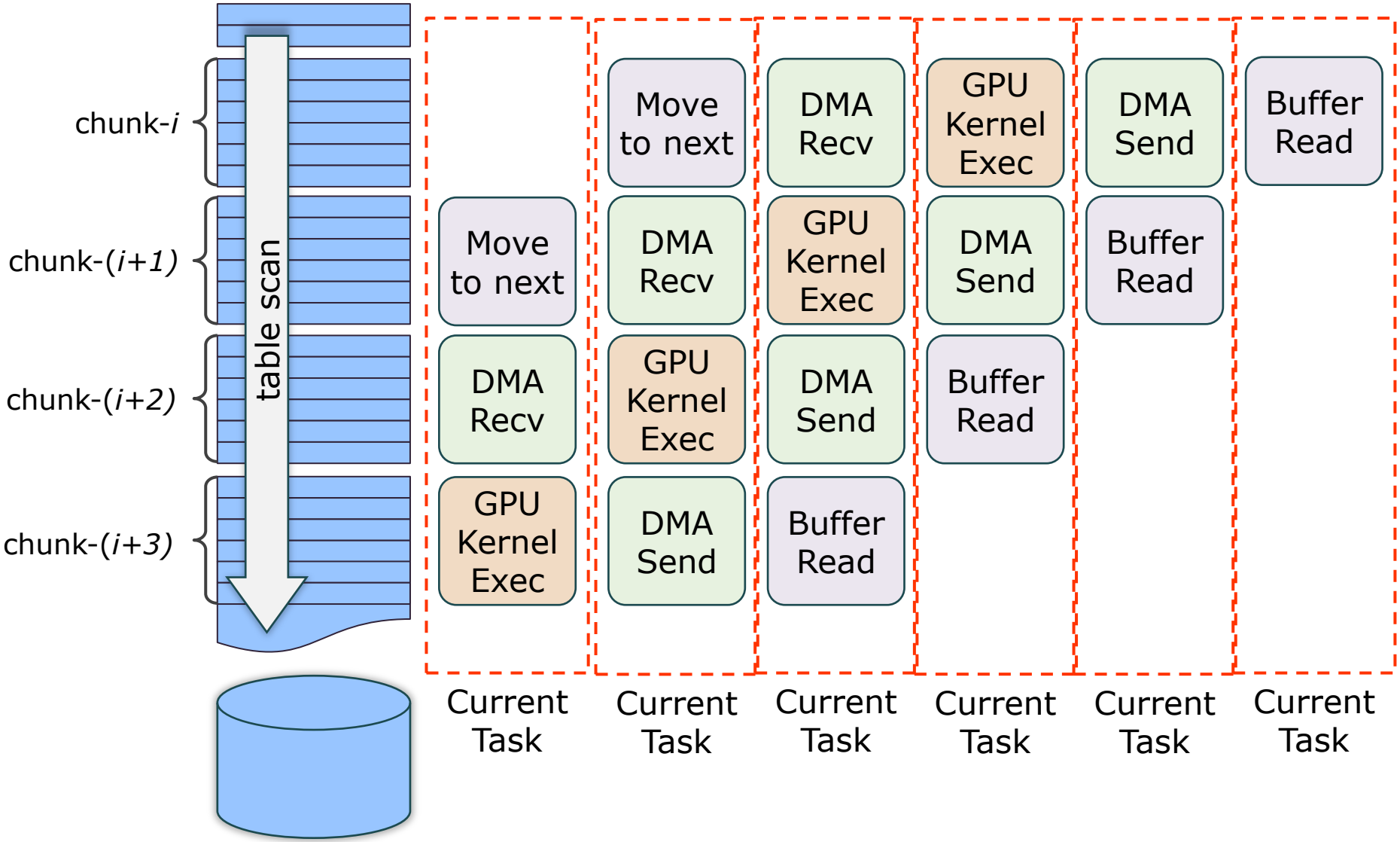
(OT) How to combine static and dynamic code



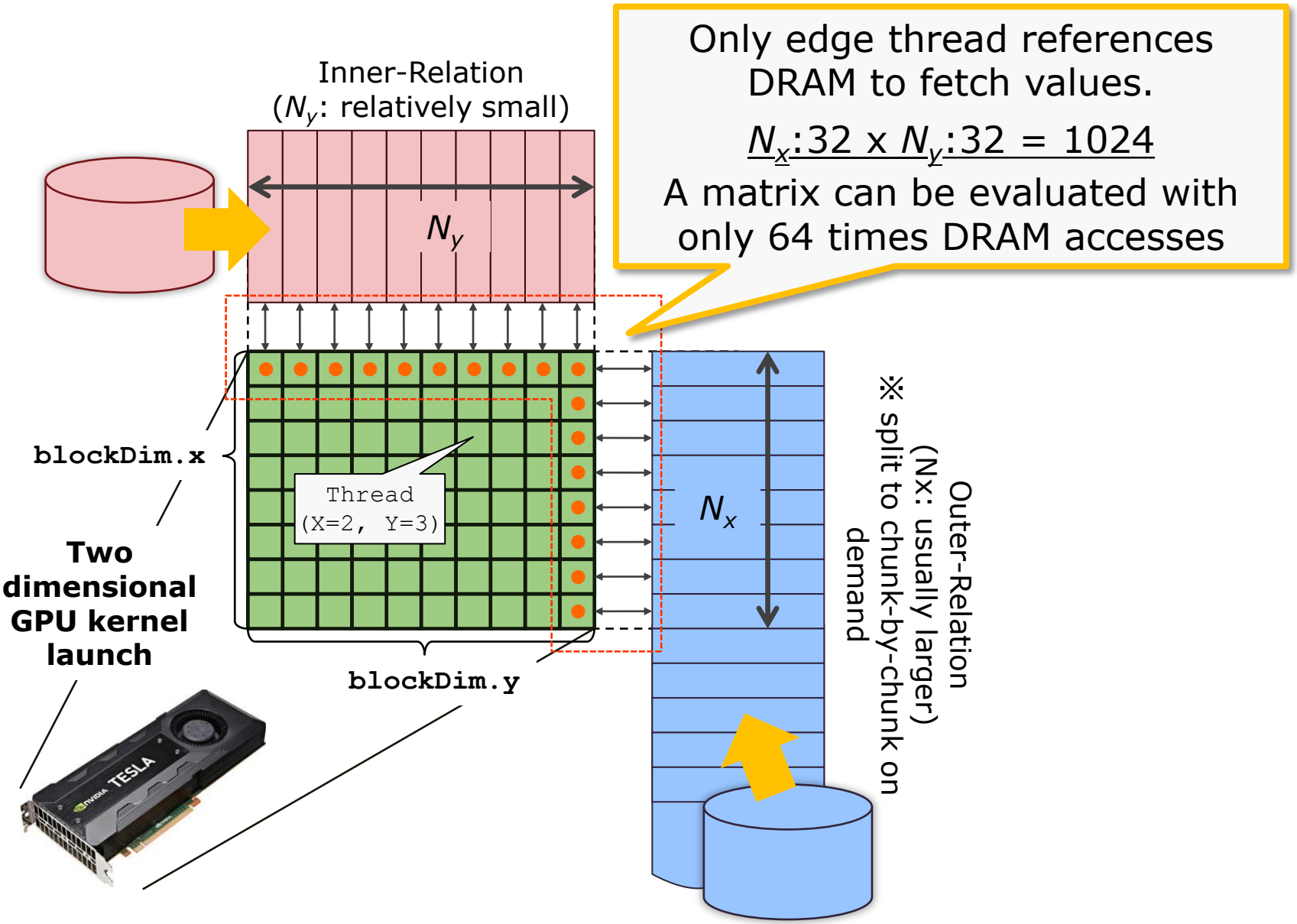
How GPU Logic works (1/2) – Case of GpuScan



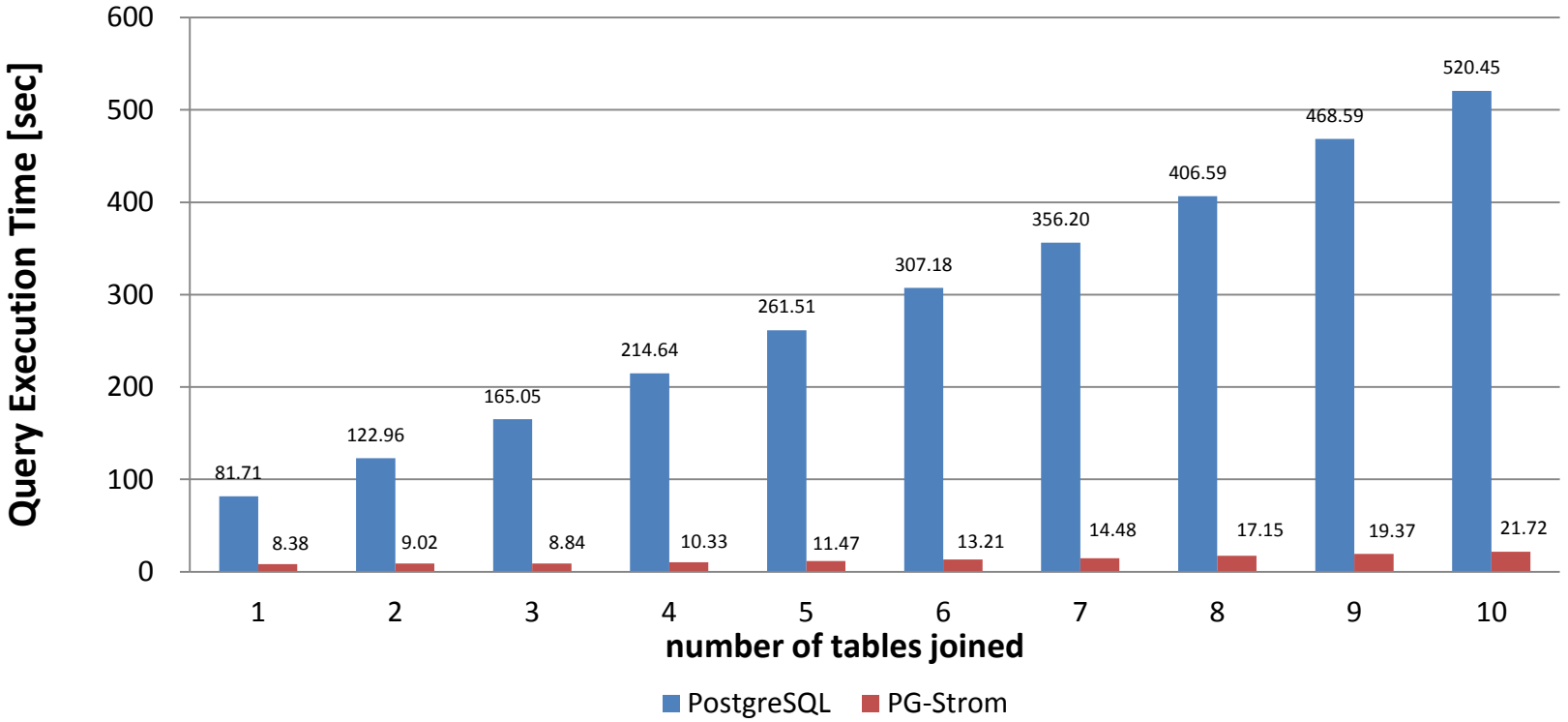
Asynchronous Execution and Pipelining



How GPU Logic works (2/2) – Case of GpuNestLoop



Benchmark Results (1/2) – Microbenchmark

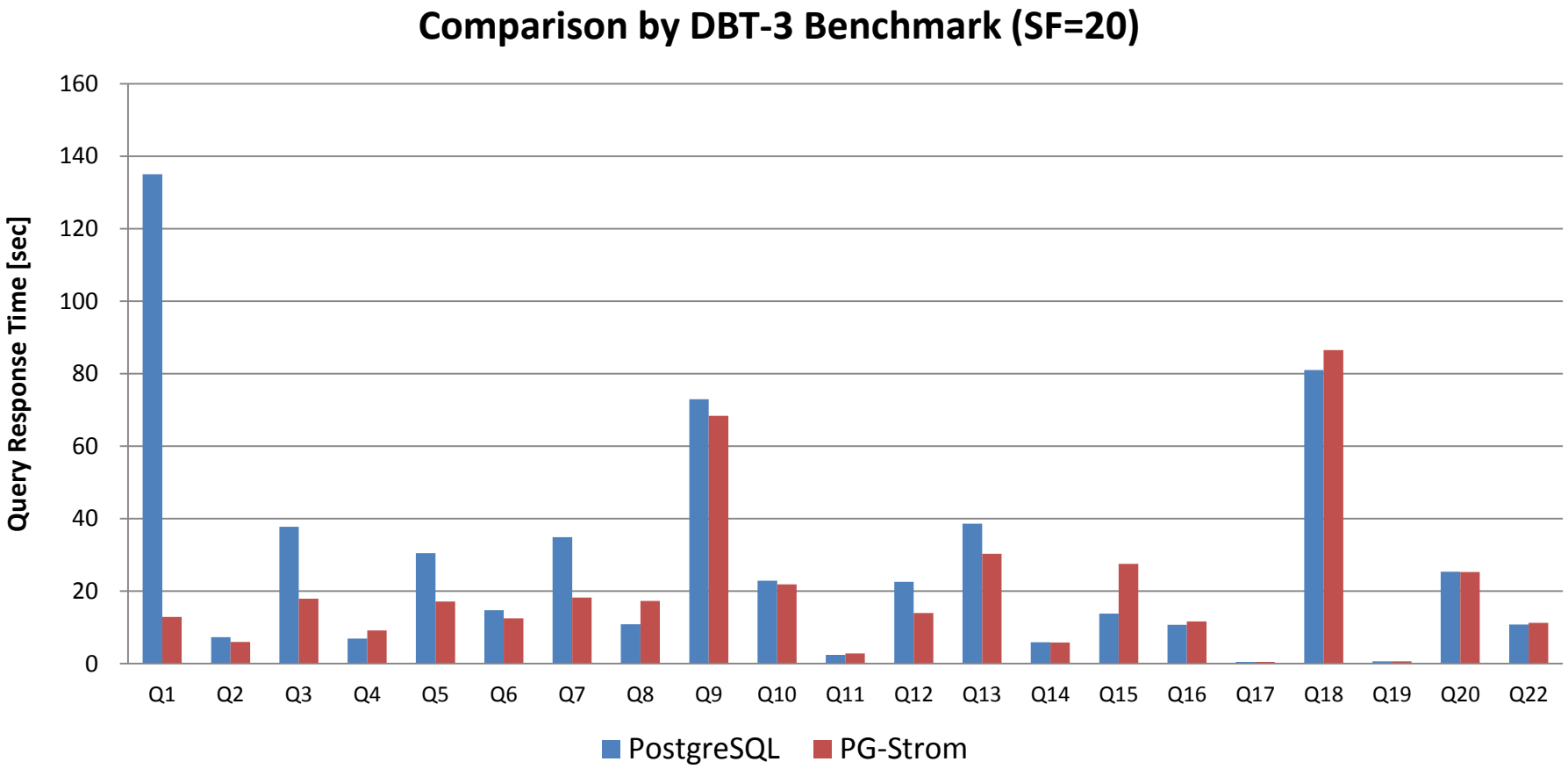


- `SELECT cat, AVG(x) FROM t0 NATURAL JOIN t1 [, ...] GROUP BY cat;`

➔measurement of query response time with increasing of inner relations

- `t0`: 100M rows, t1~t10: 100K rows for each, all the data was preloaded.
- PostgreSQL v9.5devel + PG-Strom (26-Mar), CUDA 7(x86_64)
- CPU: Xeon E5-2640, RAM: 256GB, GPU: NVIDIA GTX980

Benchmark Results (2/2) – DBT-3 with SF=20



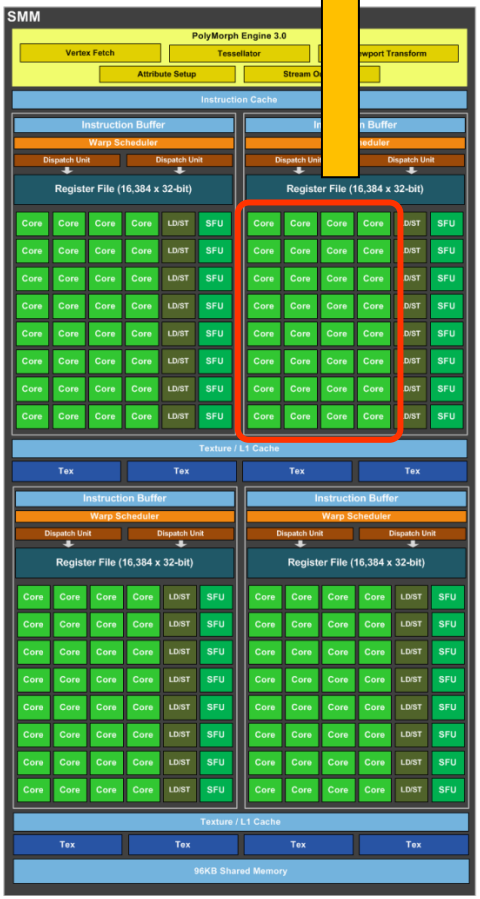
- PostgreSQL v9.5devel + PG-Strom (26-Mar), CUDA 7(x86_64)
- CPU: Xeon E5-2640, RAM: 256GB, GPU: NVIDIA GTX980

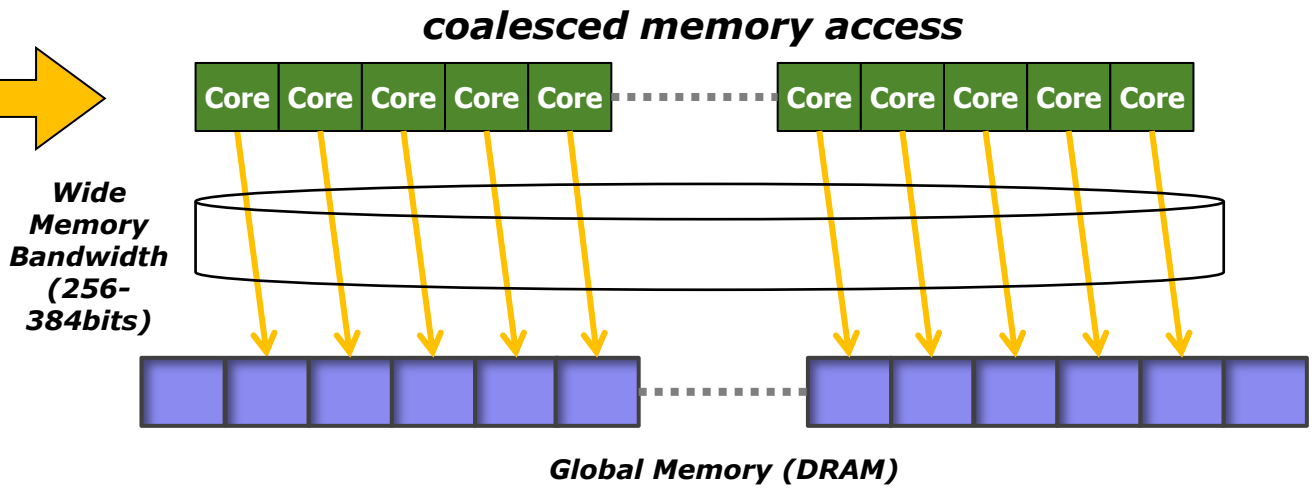
✓PG-Strom is almost faster than PostgreSQL, up to x10 times(!)

✓Q21 result is missing because of too large memory allocation by nodeHash.c

(OT) Why columnar-format is ideal for GPU

WARP:
Unit of GPU threads that share instruction pointer

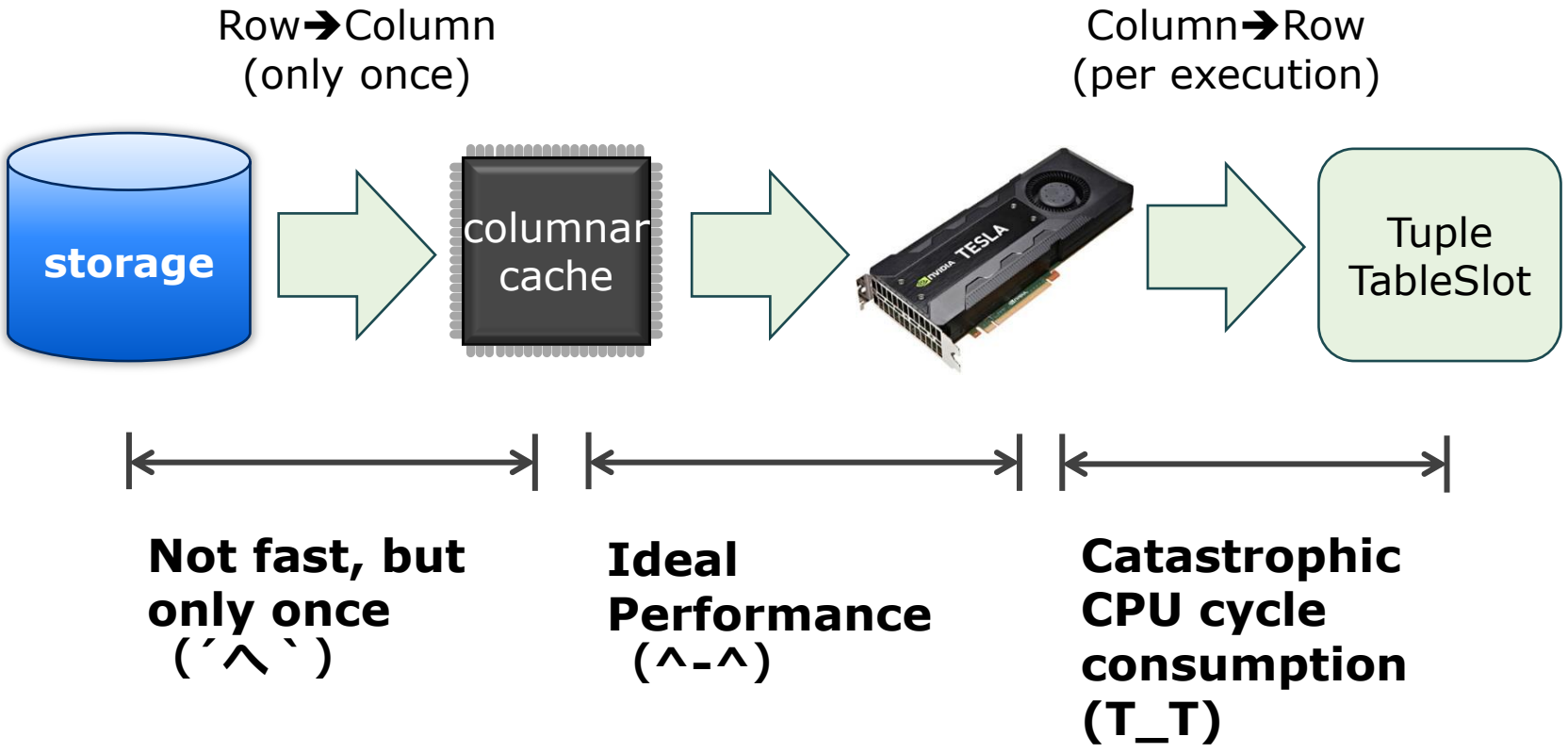




- Reduction of I/O workload
- Higher compression ratio
- Less amount of DMA transfer
- Suitable for SIMD operation
- Maximum performance on GPU kernel, by coalesced memory access

SOURCE: Maxwell: The Most Advanced CUDA GPU Ever Made

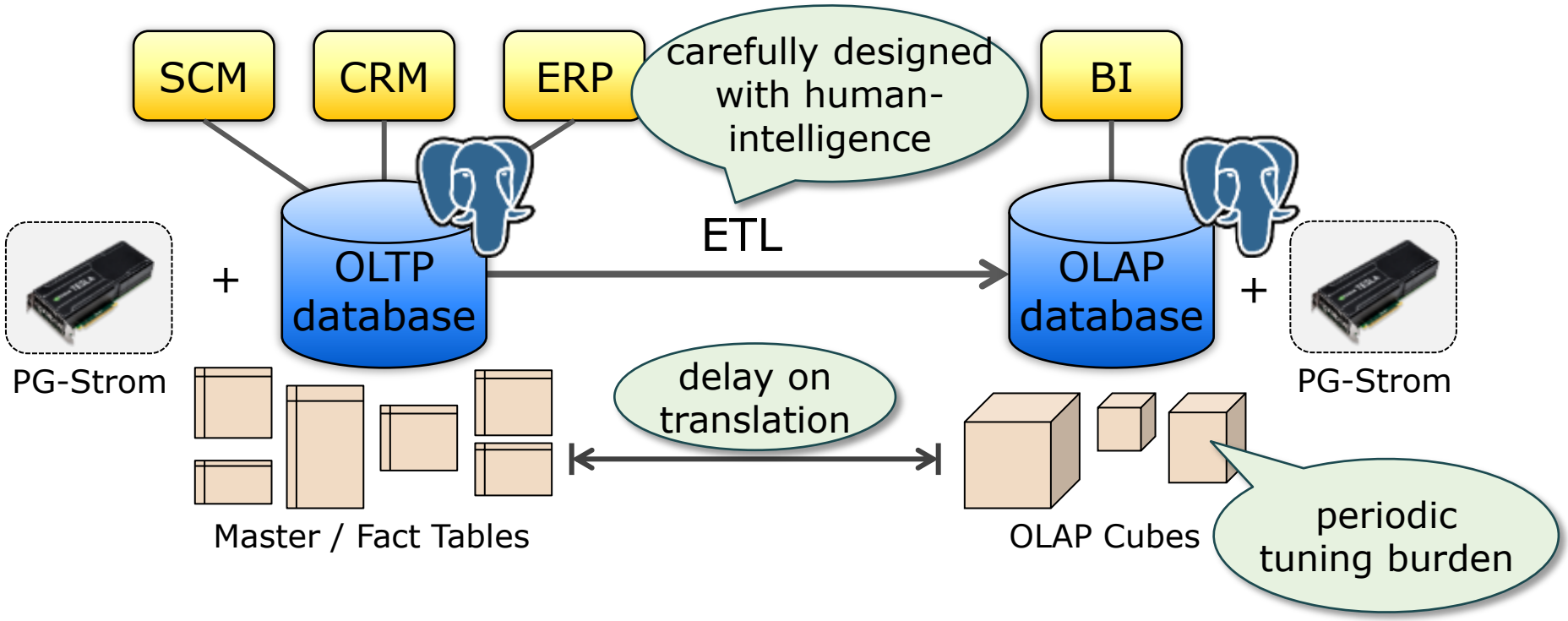
(OT) Why PG-Strom (at this moment) use row-format



Future direction

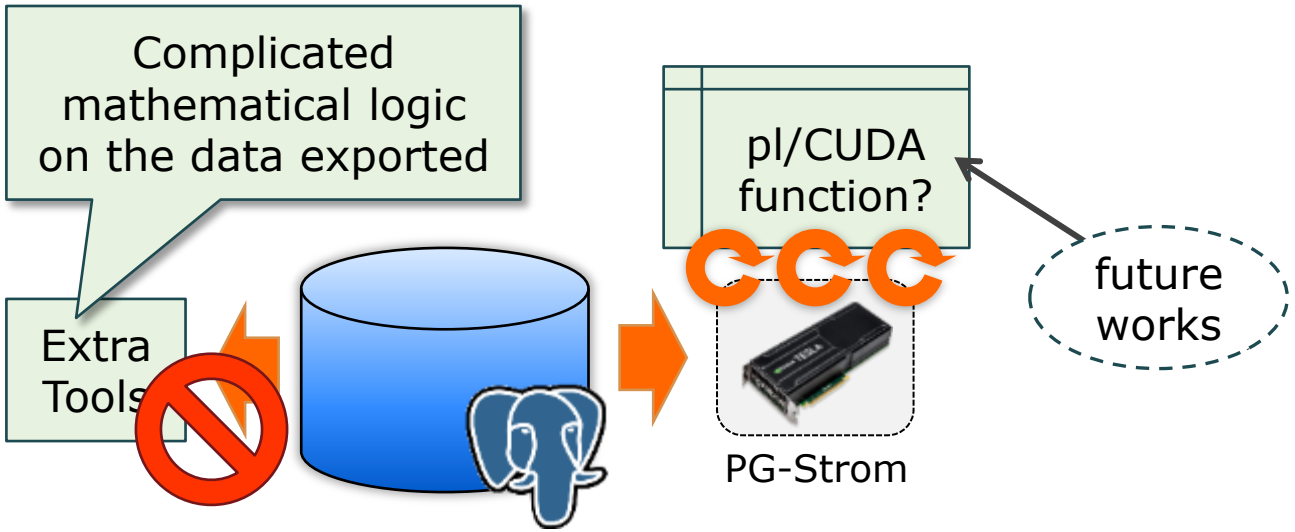
- Integration with native columnar storage
- Column ➔ Row translation in GPU space

Expected Scenario (1/2) – Backend of business intelligence



- Reduction of DBA work-loads/burden
- A new option for database tuning
- Analytics under the operation

Expected Scenario (2/2) – Computing In-Place

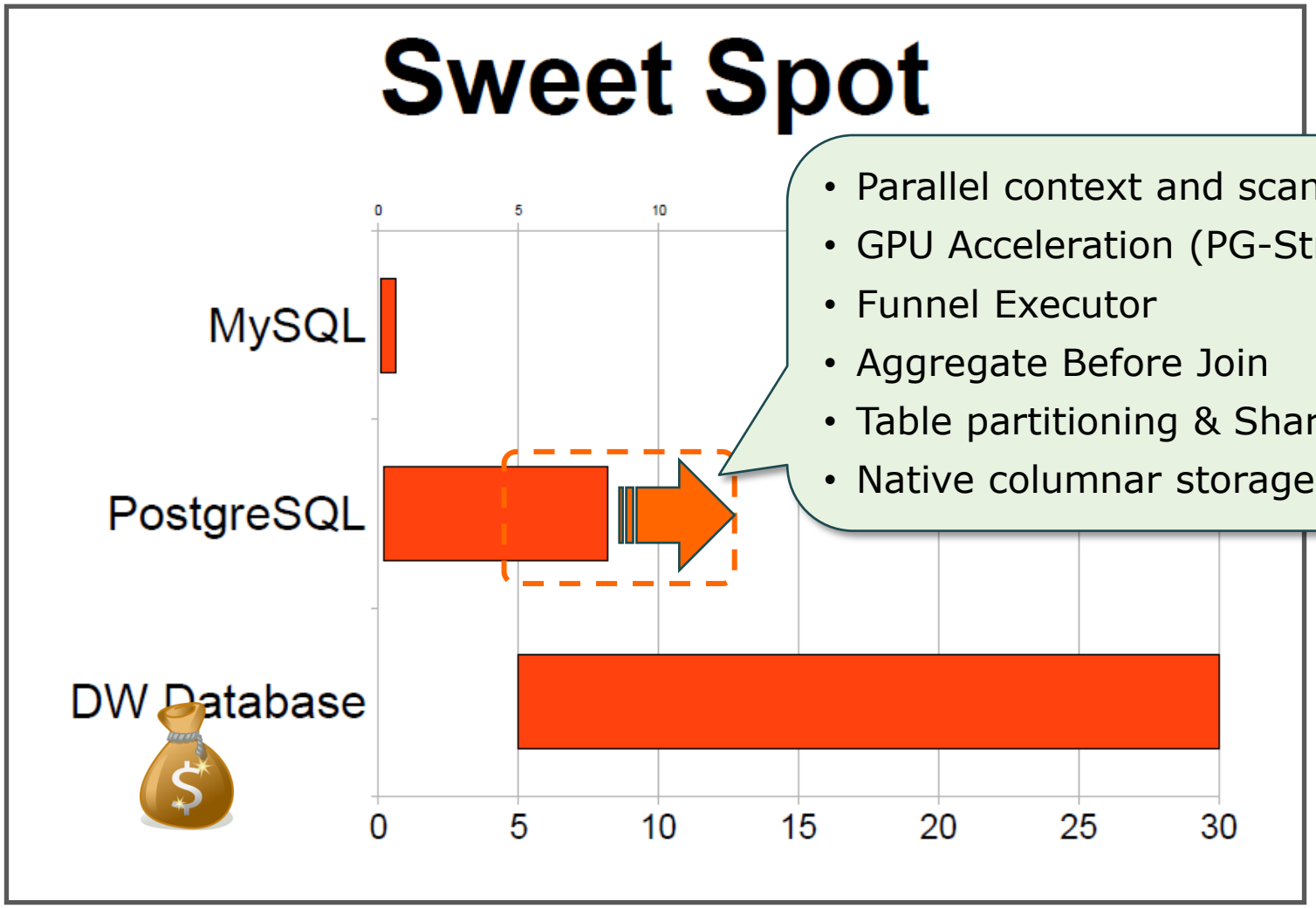


- Computing In-Place
  - Why people export data once, to run their algorithm?
  - ➔RDBMS is not designed as a tool compute stuff
  - If RDBMS can intermediate the world of data management and computing/calculation?
- All we need to fetch is data already processed
- System landscape gets simplified

Welcome your involvement

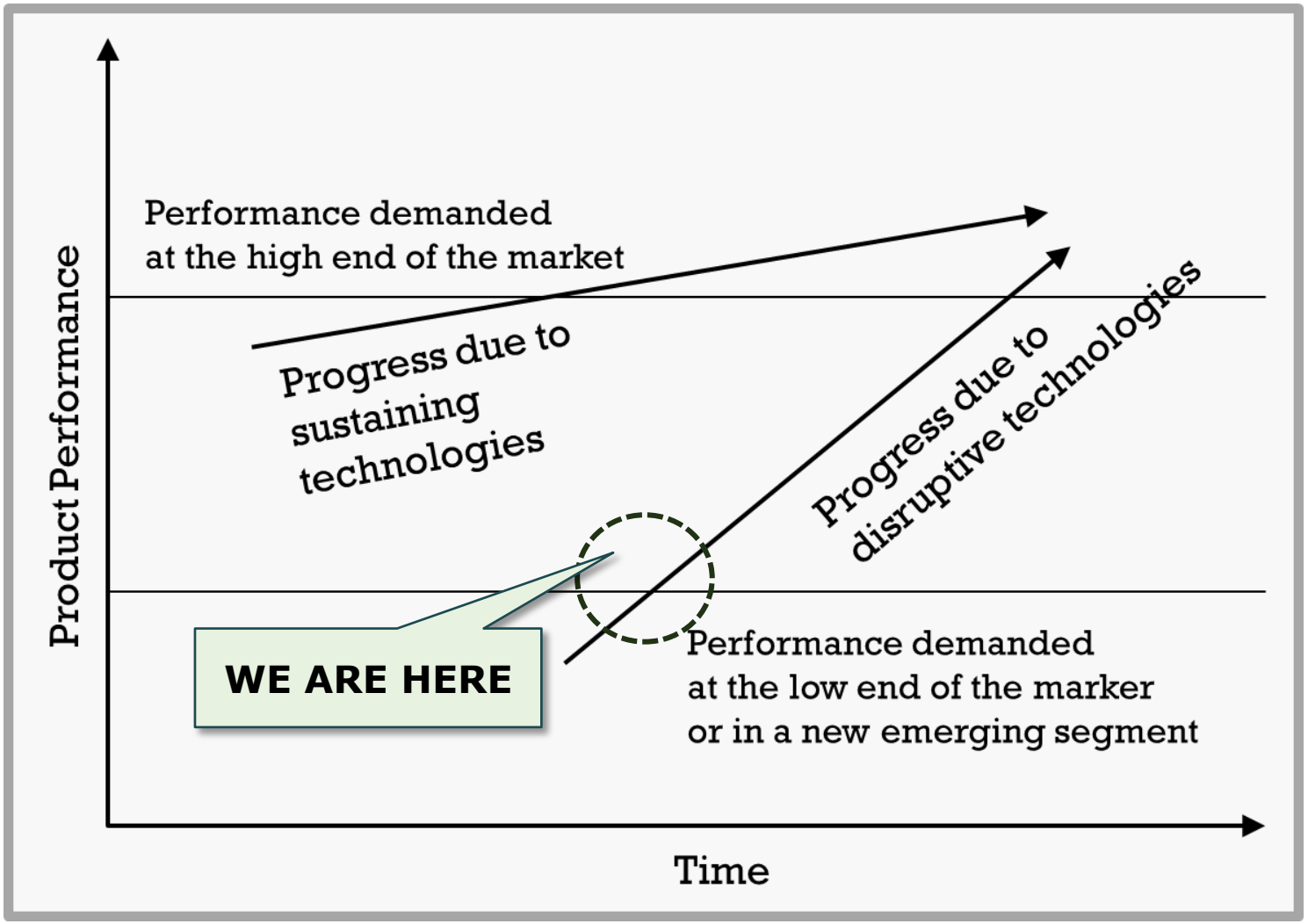
- Early adopters are big welcome
  - SaaS provider or ISV on top of PostgreSQL, notably
  - Folks who have real-life workloads and dataset
- Let’s have joint evaluation/development

Our sweet spot?



SOURCE: Really Big Elephants – Data Warehousing with PostgreSQL, Josh Berkus, MySQL User Conference 2011

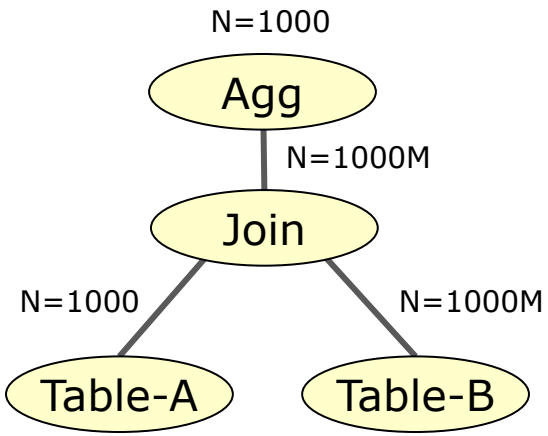
Our position



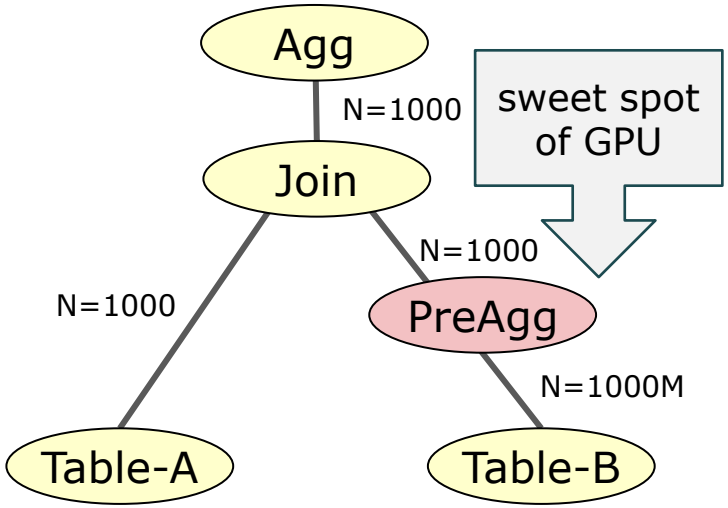
SOURCE: The Innovator's Dilemma,
Prof. Clayton Christensen , Harvard Business School

Towards v9.6 (1/2) – Aggregation before Join

Original Query



Aggregate before Join



Problem

- All the aggregations are done on the final stage of execution

Solution

- Make a partial aggregate first, then Join and final aggregate

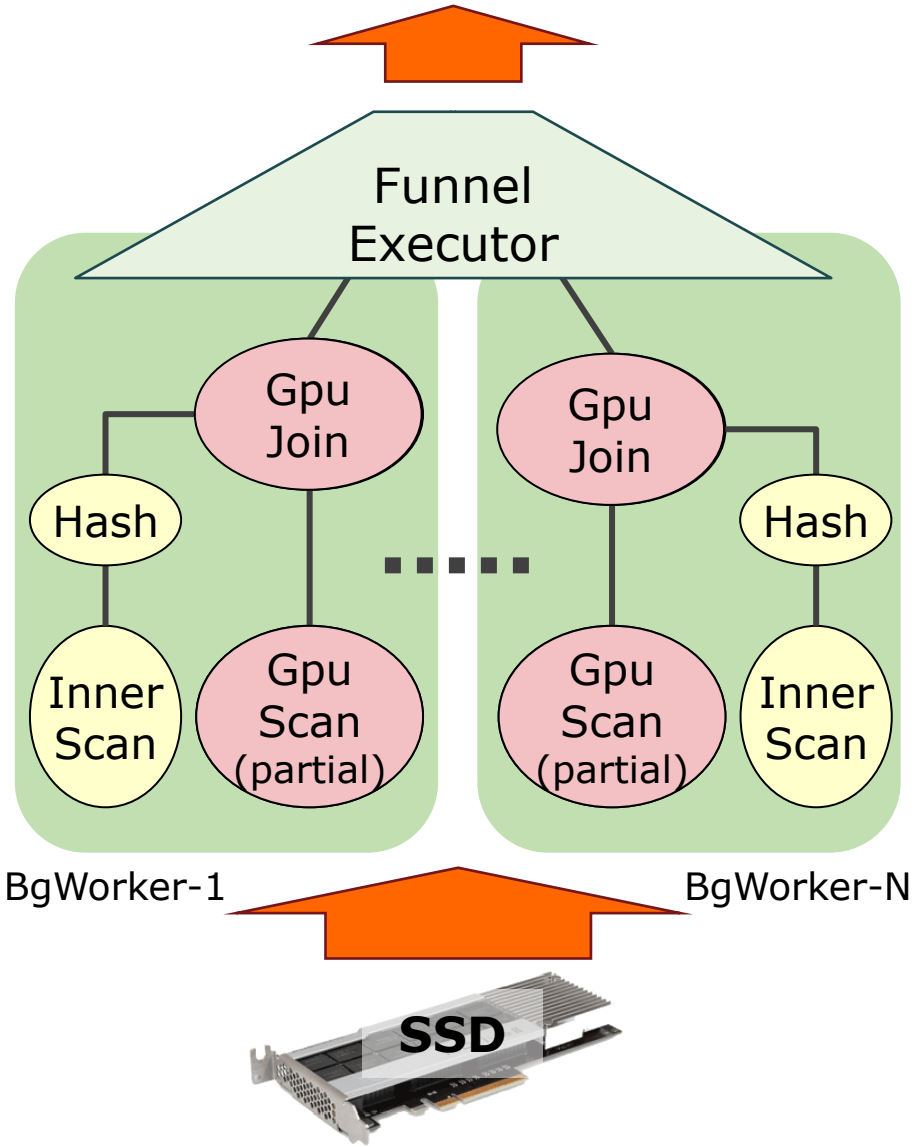
Benefit

- Reduction of Join workloads
- Partial aggregate is sweet spot of GPU acceleration.

Challenge

- Planner enhancement to deal with various path-nodes
- Aggregate Combined Function

Towards v9.6 (2/2) – CustomScan under Funnel Executor



Problem

- Low I/O density on Scan
- Throughput of input stream

Solution

- Split a large chunk into multiple chunks using BGW

Benefit

- Higher I/O density
- CPU+GPU hybrid parallel

Challenge

- Planner enhancement to deal with various path-nodes
- SSD optimization
- CustomScan nodes across multiple processes

Resources

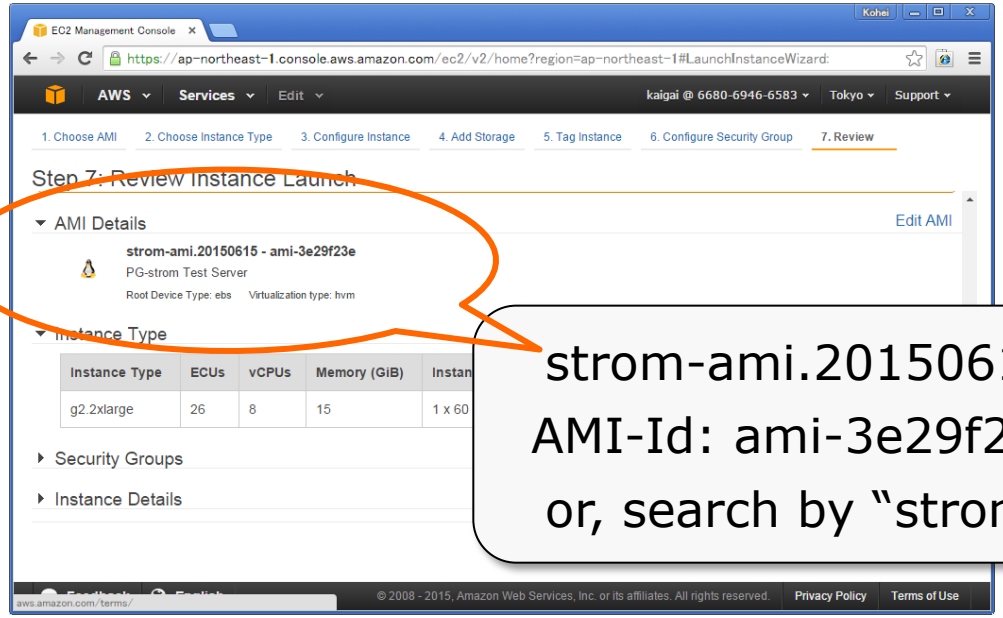
Source

- https://github.com/pg-strom/devel

Requirement

- PostgreSQL v9.5devel
- Hotfix patch (custom_join_children.v2.patch)
- CUDA 7.0 provided by NVIDIA

On cloud (AWS)



g2.2xlarge	
CPU	Xeon E5-2670 (8 xCPU)
RAM	15GB
GPU	NVIDIA GRID K2 (1536 core)
Storage	60GB of SSD
Price	$0.898/hour (*) Tokyo region, at Jun-2015

Questions?

\Orchestrating a brighter world

NEC

NEC brings together and integrates technology and expertise to create
the ICT-enabled society of tomorrow.
We collaborate closely with partners and customers around the world,
orchestrating each project to ensure all its parts are fine-turned to local needs.

Every day, our innovative solutions for society contribute to
greater safety, security, efficiency and equality,
and enable people to live brighter lives.