PG-Strom
Query Acceleration Engine of PostgreSQL
Powered by GPGPU

NEC OSS Promotion Center
The PG-Strom Project
KaiGai Kohei <kaigai@ak.jp.nec.com>
Self Introduction

Name: KaiGai Kohei
Company: NEC
Mission: Software architect & Intrepreneur

Background:
- Linux kernel development (2003~?)
- PostgreSQL development (2006~)
- SAP alliance (2011~2013)
- PG-Strom development & productization (2012~)

PG-Strom Project:
- In-company startup of NEC
- Also, an open source software project
What is PG-Strom

- An Extension of PostgreSQL
- Off-loads CPU intensive SQL workloads to GPU processors

**Major Features**
1. Automatic and just-in-time GPU code generation from SQL
2. Asynchronous and concurrent query executor

![Diagram of PG-Strom architecture](image)
### Concept

**No Pain**
- Looks like a traditional PostgreSQL database from standpoint of applications, thus, we can utilize existing tools, drivers, applications.

**No Tuning**
- Massive computing capability by GPGPU kills necessity of database tuning by human. It allows engineering folks to focus on the task only human can do.

**No Complexity**
- No need to export large data to external tools from RDBMS, because its computing performance is sufficient to run the workloads nearby data.
Technology Trend

Movement to CPU/GPU integrated architecture rather than multicore CPU
Free lunch for SW by HW evolution will finish soon

Unless software is not designed to utilize GPU capability, unable to pull-out the full hardware capability.

SOURCE: THE HEART OF AMD INNOVATION, Lisa Su, at AMD Developer Summit 2013
Background: How SQL is executed

Planner constructs query execution plan based on cost estimation

SQL never defines how to execute the query, but what shall be returned
Background: Custom-Plan Interface

```sql
SELECT cat, avg(x) FROM t1, t2
WHERE t1.id = t2.id AND y > 100
GROUP BY cat;
```

- **Aggregate**
  - `key: cat`
  - “GpuHashJoin”
  - `t1.id = t2.id`

- **t1**
  - IndexScan on t1
  - `y > 100`

- **t2**
  - “BulkLoad” on t1

**Join**
- Hash Join
- Merge Join
- Nested Loop
- **Custom Join**
- Seq Scan
- Index Scan
- Index-Only Scan
- Tid Scan
- **Custom Scan**
PG-Strom Features

Logics
- GpuScan ... Parallel evaluation of scan qualifiers
- GpuHashJoin ... Parallel multi-relational join
- GpuPreAgg ... Two phase aggregation
- GpuSort ... GPU + CPU Hybrid Sorting
- GpuNestedLoop (in develop)

Data Types
- Integer, Float, Date/Time, Numeric, Text

Function and Operators
- Equality and comparison operators
- Arithmetic operators and mathematical functions
- Aggregates: count, min/max, sum, avg, std, var, corr, regr
postgres=# SET pg_strom.show_device_kernel = on;
postgres=# EXPLAIN VERBOSE SELECT * FROM t0 WHERE sqrt(x+y) < 10;

QUERY PLAN

Custom Scan (GpuScan) on public.t0  (cost=500.00..357569.35 rows=6666683 width=77)
  Output: id, cat, aid, bid, cid, did, eid, x, y, z
  Device Filter: (sqrt((t0.x + t0.y)) < 10::double precision)
  Features: likely-tuple-slot
  Kernel Source: #include "opencl_common.h"
    :
    static pg_bool_t
gpuscan_qual_eval(__private cl_int *errcode,
    __global kern_parambuf *kparams,
    __global kern_data_store *kds,
    __global kern_data_store *ktoast,
    size_t kds_index)
    {
      pg_float8_t KPARAM_0 = pg_float8_param(kparams,errcode,0);
      pg_float8_t KVAR_8 = pg_float8_vref(kds,ktoast,errcode,7,kds_index);
      pg_float8_t KVAR_9 = pg_float8_vref(kds,ktoast,errcode,8,kds_index);

      return pgfn_float8lt(errcode,
                               pgfn_dsqrt(errcode, pgfn_float8pl(errcode, KVAR_8, KVAR_9)), KPARAM_0);
    }
Implementation (1/3) – GpuScan

Execution of auto-generated GPU code

Result Output Stream

DMA Send
DMA Recv

DMA Send
DMA Recv

Input Stream

Table

Chunk (16~64MB)
Software Architecture

SQL Query

- Query Parser
  - Breaks down the query to parse tree
- Query Optimizer
  - Makes query execution plan
- Query Executor
  - Run the query

Storage Manager

Storage

Shared Buffer

PostgreSQL

PG-Strom

Message Queue

- Custom-Plan APIs
  - GpuScan
  - GpuHashJoin
  - GpuPreAgg
  - GpuSort
- GPU Code Generator

PG-Strom OpenCL Server

GPU Program Manager

※ Current version based on OpenCL

Direct Memory Access (via PCI-E bus)
Implementation (2/3) – GpuHashJoin

vanilla Hash-Join

Next stage

Sequential Materialization by CPU

Hash Table Search by CPU

Inner relation

Outer relation

GpuHashJoin

Next stage

CPU just references materialized results

Parallel Materialization

Hash Table

Parallel Hash-Table Search

Inner relation

Outer relation
Benchmark result (1/2) – simple tables join

**Benchmark Query:**

```
SELECT * FROM t0 NATURAL JOIN t1 [NATURAL JOIN ....];
```

**Environment:**

- $t_0$ has 100million rows (13GB), $t_1$-$t_9$ has 40,000 rows for each, all-data pre-loaded
- CPU: Xeon E5-2670v3 (12C, 2.3GHz) x2, RAM: 384GB, GPU: Tesla K20c x1
Implementation (3/3) – GpuPreAgg

1st Stage Reduction

2nd Stage Reduction

Chunk (16~64MB)

Table
Benchmark result (2/2) – Star Schema Model

Typical Reporting Queries on Retail / Star-Schema Data

- 40 typical reporting queries
- 100GB of retail / start-schema data, all pre-loaded

Environment
- CPU: Xeon E5-2670v3(12C, 2.3GHz) x2, RAM: 384GB, GPU: Tesla K20c x1
Expected Scenario – Reduction of ETL

- ETL – Its design is human centric task
- Replication – much automatous task

**PG-Strom**

- Query Acceleration Engine of PostgreSQL Powered by GPGPU

**Replication**

- Replica of Master / Fact Tables
- Sufficient to **analytic** workloads also

**OLTP**

- Optimized to **transaction** workloads

**OLAP**

- Optimized to **analytic** workloads
Direction of PG-Strom
Development Plan

Current version: PG-Strom β + PostgreSQL v9.5devel

- Migration of OpenCL to CUDA
- Add support of GpuNestedLoop
- Add support multi-functional kernel
- Standardization of custom-join interface
- ...and more...?

Short term target: PostgreSQL v9.5 timeline (2015)

- Integration with funnel executor
- Investigation to SSD/NvRAM utilization
- Custom-sort/aggregate interface
- Add support for spatial data types (?)

Middle term target: PostgreSQL v9.6 timeline (2016)
Let’s try – Deployment on AWS

**AWS GPU Instance (g2.2xlarge)**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Xeon E5-2670 (8 xCPU)</td>
</tr>
<tr>
<td>RAM</td>
<td>15GB</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA GRID K2 (1536core)</td>
</tr>
<tr>
<td>Storage</td>
<td>60GB of SSD</td>
</tr>
<tr>
<td>Price</td>
<td>$0.898/hour, $646.56/mon</td>
</tr>
</tbody>
</table>

(*) Price for on-demand instance on Tokyo region at Nov-2014

Search by “strom”!
Welcome your involvement!

How to be involved?
- as a user
- as a developer
- as a business partner

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

Contact US
- e-mail: kaigai@ak.jp.nec.com
- twitter: @kkaigai

...or, catch me in the Convention Center
Orchestrating a brighter world

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-tuned 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.
Empowered by Innovation

NEC