GPU Accelerated Data Processing

Speed of Thought Analytics at Scale

S9373 - TPC-H Benchmark on DGX-2
A New Paradigm for OLAP and Decision Support
Key pain points

Flexibility

43%

Of analysts say their analytics is not flexible enough to meet their needs

Performance

32%

Of analysts say they have to deal with slow query speeds

The reason data insights is so challenging is analytics solutions today simply do not have the speed, flexibility, and ease of use to answer the data questions people are asking.
Where are analysts spending time?

- 64% of time is spent cleaning and organizing data
- 3 days per month is spent mining data for patterns or refining algorithms
- 37% of insight takes more than a week
- SQL is the most common technology used ahead of Hadoop, Python and R
The fastest, most advanced GPU database on the market

Our mission is to empower organisations through **Speed of Thought Analytics**.

- The world’s fastest database according to independent benchmarking.
- Four years in research and development.
- Only vendor to have patent pending IP for JOINs.
- Fourth generation GpuManagner bridges the gap between SQL and AI.

The true value of Brytlyt lies in how this extreme performance is package for the end user.
<table>
<thead>
<tr>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
<th>Query 4</th>
<th>Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>0.011</td>
<td>0.103</td>
<td>0.188</td>
<td><strong>BrytlytDB 2.1 &amp; 5-node IBM Minsky cluster</strong></td>
</tr>
<tr>
<td>0.009</td>
<td>0.027</td>
<td>0.287</td>
<td>0.428</td>
<td><strong>BrytlytDB 2.0 &amp; 2-node p2.16xlarge cluster</strong></td>
</tr>
<tr>
<td>0.021</td>
<td>0.053</td>
<td>0.165</td>
<td>0.51</td>
<td><strong>MapD &amp; 8 Nvidia Pascal Titan Xs</strong></td>
</tr>
<tr>
<td>0.027</td>
<td>0.083</td>
<td>0.163</td>
<td>0.891</td>
<td><strong>MapD &amp; 8 Nvidia Tesla K80s</strong></td>
</tr>
<tr>
<td>0.028</td>
<td>0.2</td>
<td>0.237</td>
<td>0.578</td>
<td><strong>MapD &amp; 4-node g2.8xlarge cluster</strong></td>
</tr>
<tr>
<td>0.034</td>
<td>0.061</td>
<td>0.178</td>
<td>0.498</td>
<td><strong>MapD &amp; 2-node p2.8xlarge cluster</strong></td>
</tr>
<tr>
<td>0.036</td>
<td>0.131</td>
<td>0.439</td>
<td>0.964</td>
<td><strong>MapD &amp; 4 Nvidia Titan Xs</strong></td>
</tr>
<tr>
<td>0.051</td>
<td>0.146</td>
<td>0.047</td>
<td>0.794</td>
<td><strong>kdb+/g &amp; 4 Intel Xeon Phi 7210 CPUs</strong></td>
</tr>
<tr>
<td>0.762</td>
<td>2.472</td>
<td>4.131</td>
<td>6.041</td>
<td><strong>BrytlytDB 1.0 &amp; 2-node p2.16xlarge cluster</strong></td>
</tr>
<tr>
<td>1.034</td>
<td>3.058</td>
<td>5.354</td>
<td>12.748</td>
<td><strong>ClickHouse, Intel Core i5 4670K</strong></td>
</tr>
<tr>
<td>1.56</td>
<td>1.25</td>
<td>2.25</td>
<td>2.97</td>
<td><strong>Redshift, 6-node ds2.8xlarge cluster</strong></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td><strong>BigQuery</strong></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>10</td>
<td>21</td>
<td><strong>Presto, 50-node n1-standard-4 cluster</strong></td>
</tr>
<tr>
<td>4.88</td>
<td>11</td>
<td>12</td>
<td>15</td>
<td><strong>Presto 0.188 &amp; 21-node m3.xlarge cluster</strong></td>
</tr>
<tr>
<td>6.41</td>
<td>6.19</td>
<td>6.09</td>
<td>6.63</td>
<td><strong>Amazon Athena</strong></td>
</tr>
<tr>
<td>8.1</td>
<td>18.18</td>
<td>n/a</td>
<td>n/a</td>
<td><strong>Elasticsearch (heavily tuned)</strong></td>
</tr>
<tr>
<td>10.19</td>
<td>8.134</td>
<td>19.624</td>
<td>85.942</td>
<td><strong>Spark 2.1, 11 x m3.xlarge cluster w/ HDFS</strong></td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>21</td>
<td>31</td>
<td><strong>Presto, 10-node n1-standard-4 cluster</strong></td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>16</td>
<td>22</td>
<td><strong>Presto 0.188 &amp; 38xlarge w/ HDFS</strong></td>
</tr>
<tr>
<td>14.389</td>
<td>32.148</td>
<td>33.448</td>
<td>67.312</td>
<td><strong>Vertica, Intel Core i5 4670K</strong></td>
</tr>
<tr>
<td>22</td>
<td>25</td>
<td>27</td>
<td>65</td>
<td><strong>Spark 2.3.0 &amp; 38xlarge w/ HDFS</strong></td>
</tr>
<tr>
<td>28</td>
<td>31</td>
<td>33</td>
<td>80</td>
<td><strong>Spark 2.2.1 &amp; 21-node m3.xlarge cluster</strong></td>
</tr>
</tbody>
</table>
Brytlyt is a PostgreSQL fork

- User
- Client
- PostgreSQL
  - Parser
  - Planner
- DB Engine
- Disk Storage
- Foreign Data Wrapper
- NVIDIA GPU Hardware
- Brytlyt GPU Manager
- 3rd Party Data Sources
- 3rd Party Data Sources
Brytlyt technology

SpotLyte Analytics Workbench
Canis Task Orchestration
BrytlytDB PostgreSQL on GPU
BrytMind Artificial Intelligence on GPU

Tools

Engines

GPU
TPC-H Benchmark

Why
• Measure of state of maturity of GPU database space.
• Performance comparisons of hardware and software.

What
• Examine large volumes of data, by executing queries with high degree of complexity, to give answers on real-world business decisions.

How
• Star schema, two large fact tables (88% of total row count) and six dimension tables
• Twenty two queries run as single user and concurrently.
• Based on typical retail use case.
• A data generator that goes up to and beyond 100TB
NVIDIA DGX-2

Why
• Step change in GPU footprint of a single server.
• Cluster of servers with network bottleneck less necessary.

What
• Sixteen NVIDIA V100 GPUs with 32GB VRAM.
• Total of 512 GB VRAM and 2 petaFLOPs.

How
• NVSwitch provides 2.4 TB/s of GPU data transfer between GPUs.
TPC-H Summary

Aggregations
• Occur in all TPC-H queries and group-by performance is important.

Complex expressions
• Raw expressions in aggregations, complex expressions in joins and also string matching.

Nested queries and sub-queries
• Used to handle intermediate results in the real world.

JOINs
• All but two of the queries contain joins.

Correlated queries
• Special case of nested query where the subquery uses values from the outer query.
TPC-H – Set up and comparisons

**Scale factor** 1,000 GB (6 billion rows in the lineitem table)

- **Brytlyt** Year: 2019, DGX-2, Version 3.1 Alpha
- **Exasol** Year: 2014, twenty machines, TCO $719k
- **Microsoft** Year: 2017, one machine, TCO $472k

*No results of full benchmark by other GPU vendors in public domain.*
Notes to benchmarking exercise

All queries run sub-second.

Redistributing lineitem table can be done sub-second (largest fact table, 70% of total data row count, 6 billion rows).
TPC-H Runtimes

Run time in seconds

Brytlyt  Exasol  Microsoft

0  2  4  6  8  10  12  14  16  18  20  12  14  16  17  18  20  21  22
SELECT l_returnflag, 
    l_linestatus, 
    sum(l_quantity) as sum_qty, 
    sum(l_extendedprice) as sum_base_price, 
    sum(l_extendedprice*(1-l_discount)) as sum_disc_price, 
    sum(l_extendedprice*(1-l_discount)*(1+l_tax)) as sum_charge, 
    avg(l_quantity) as avg_qty, 
    avg(l_extendedprice) as avg_price, 
    avg(l_discount) as avg_disc, 
    count(*) as count_order
FROM lineitem
WHERE l_shipdate <= date '1998-12-01' - interval '90 day'
GROUP BY l_returnflag, l_linestatus
ORDER BY l_returnflag, l_linestatus;
Runtime comparison – Q1

- brytlyt
- Exasol
- Microsoft
SELECT c_count,
    count(*) AS custdist
FROM ( SELECT c_custkey,
        count(o_orderkey)
    FROM customer
    LEFT OUTER JOIN orders ON
        c_custkey = o_custkey
        AND o_comment NOT LIKE '%a%b%'
    GROUP BY c_custkey
    ) AS c_orders (c_custkey, c_count)
GROUP BY c_count
ORDER BY custdist desc, c_count desc;
SELECT c_count,
count(*) AS custdist
FROM (SELECT c_custkey,
count(o_orderkey)
FROM customer
LEFT OUTER JOIN orders ON c_custkey = o_custkey
and o_comment NOT LIKE '%a%b%'
GROUP BY c_custkey
) AS c_orders (c_custkey, c_count)
GROUP BY c_count
ORDER BY custdist desc, c_count desc;
Runtime comparison – Q13
JOINs – Q5 uses six tables

```
SELECT n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
FROM customer,
    JOIN orders ON c_custkey = o_custkey
    JOIN lineitem ON l_orderkey = o_orderkey
    JOIN supplier ON l_suppkey = s_suppkey
    JOIN nation ON s_nationkey = n_nationkey
    JOIN region ON n_regionkey = r_regionkey
WHERE c_nationkey = s_nationkey
    and r_name = '[REGION]'
    and o_orderdate >= date '1995-01-01'
    and o_orderdate < date '1995-01-01' + interval '1' year
GROUP BY n_name
ORDER BY revenue desc;
```
Recursive Interaction Probability (RIP)

Why
• JOINs are the most costly and useful of SQL operations.
• Better performance and flexibility than hash- and index-based methods.

What
• Brytlyt’s patent pending intellectual property.
• Light weight pre-processing identifies tuples likely to fulfil JOIN predicate.
• Very efficient, Big O notation = O(n log n).

How
• Sorting JOIN columns.
• Recursively compare boundary elements of partitions of data.
Recursive Interaction Probability (RIP)

- Two number lines representing sorted JOIN columns.
- Using min and max values of sub-partition A.
- Comparing to min and max values of B and C.
- Determine there is zero probability of JOIN predicate being fulfilled within sub-partitions A and C.

- For sub-partitions like A and B that “interact”.
- Partition into smaller sub-partitions and repeat.

- Base case operation tests for JOIN.
- Incredibly efficient for “sparse” JOINs.
Runtime comparison – Q5

Run time in seconds

Brytlyt | Exasol | Microsoft

0.5 | 1 | 4
Correlated queries – Q11

SELECT ps_partkey, SUM(ps_supplycost * ps_availqty) as value
FROM partsupp JOIN supplier ON ps_suppkey = s_suppkey
JOIN nation ON s_nationkey = n_nationkey
WHERE n_name = 'ARGENTINA'
GROUP BY ps_partkey
HAVING SUM(ps_supplycost * ps_availqty) >
  (SELECT SUM(ps_supplycost * ps_availqty) * 0.015
   FROM partsupp JOIN supplier ON ps_suppkey = s_suppkey
   JOIN nation ON s_nationkey = n_nationkey
   WHERE n_name = 'PERU'
  )
ORDER BY value desc;
Correlated queries – Q11

SELECT ps_partkey, SUM(ps_supplycost * ps_availqty) as value
FROM partsupp JOIN supplier ON ps_suppkey = s_suppkey
JOIN nation ON s_nationkey = n_nationkey
WHERE n_name = 'ARGENTINA'
GROUP BY ps_partkey
HAVING SUM(ps_supplycost * ps_availqty) >
(SELECT SUM(ps_supplycost * ps_availqty) * 0.015
FROM partsupp JOIN supplier ON ps_suppkey = s_suppkey
JOIN nation ON s_nationkey = n_nationkey
WHERE n_name = 'PERU')
ORDER BY value desc;
Runtime comparison – Q11

- Brytlyt
- Exasol
- Microsoft

Run time in seconds

- 6
- 5
- 4
- 3
- 2
- 1
- 0
Brytlyt DB
GPU accelerated PostgreSQL
SpotLyt
Interactive analytics workbench for billion row datasets
BrytMind
SQL + AI + GPU
GPU Accelerated Data Processing

Speed of Thought Analytics at Scale

CEO  Richard Heyns
Email Richard.Heyns@Brytlyt.com
URL  www.brytlyt.com
Twitter @BrytlytDB