Are Databases Fit for Hybrid Workloads on GPUs? 
A Storage Engine’s Perspective

Marcus Pinnecke, David Broneske, Gabriel Campero Durand, Gunter Saake
Hybrid Transaction and Analytic Processing (HTAP)

HTAP database systems run both OLTP & OLAP
- HyPer, Peloton, HANA, ...

**benefit** is larger business value, through:
- less latency for analysis
- less synchronization effort

**related challenges**
- different data access pattern
- adapt record layout (NSM, DSM,...)
- interference between query types
- contradicting optimization goals
- different types of parallelism
- hot and cold data
Database Systems on Heterogenous Platforms

heterogenous systems use co-processors
- host (CPU), and device (e.g., GPU)
- CoGaDB, GPUTx, Ocelot, ...

benefit is exploiting compute capacities
- overcome limitations of power wall
- special jobs for specialized processors

related challenges
- data transfer costs for I/O
- different programming models
- device limitations (e.g., memory capacity)
- data and operator placement
Motivation
Hybridization of HTAP and Heterogenous Computing

First: Is there performance potential?

HTAP Database Systems

Heterogenous Database Systems

TPC-C Benchmark Dataset

"OLTP" query materialization

```
select *
from customers
where 150 customers
```

"HTAP" query aggregation of some

```
select sum(c_bought_item.price)
from customers \* ... \* item
where 150 items
```

"OLAP" query aggregation of all

```
select sum(price)
from item
where true
```
Hybridization of HTAP and Heterogenous Computing

First: Is there performance potential?

Setup
TPC-C benchmark customer record 96B (21 fields) / item record 20B + 8B (4 fields + price field), system configuration operator-at-a-time processing w/ late materialization, host: max. 8 threads blockwise partitioning, device: optimized parallel reduction kernel (≥ 1024 blocks w/ 512 threads), final reduction on 1 block w/ 1024 threads, effort for join processing not incl.
Hybridization of HTAP and Heterogenous Computing

First: Is there performance potential?

![Graph showing the comparison between different storage options (column-store, row-store) and different execution modes (host & single-threaded, host & multi-threaded) for the TPC-C benchmark.](image)

**Setup**
- **TPC-C** benchmark: customer record 96B (21 fields) / item record 20B + 8B (4 fields + price field)
- **System Configuration** operator-at-a-time processing, host: max. 8 threads blockwise partitioning, device: optimized parallel reduction kernel (>= 1024 blocks w/ 512 threads), final reduction on 1 block w/ 1024 threads, effort for join processing not incl.
Hybridization of HTAP and Heterogenous Computing

First: Is there performance potential?

<table>
<thead>
<tr>
<th>#records in item table</th>
<th>throughput [records/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5M</td>
<td>500M</td>
</tr>
<tr>
<td>15M</td>
<td>1000M</td>
</tr>
<tr>
<td>25M</td>
<td>1500M</td>
</tr>
<tr>
<td>35M</td>
<td>2000M</td>
</tr>
<tr>
<td>45M</td>
<td></td>
</tr>
<tr>
<td>55M</td>
<td></td>
</tr>
<tr>
<td>65M</td>
<td></td>
</tr>
</tbody>
</table>

"OLAP" query aggregation of all

higher values are better

- column-store / host & multi-threaded
- column-store / host & single-threaded
- row-store / host & multi-threaded
- row-store / host & single-threaded

Setup
TPC-C benchmark customer record 96B (21 fields) / item record 20B + 8B (4 fields + price field), system configuration operator-at-a-time processing w/ late materialization, host: max. 8 threads blockwise partitioning, device: optimized parallel reduction kernel (>= 1024 blocks w/ 512 threads), final reduction on 1 block w/ 1024 threads, effort for join processing not incl.
Hybridization of HTAP and Heterogenous Computing

First: Is there performance potential?

```
"OLAP" query aggregation of all

higher values are better

column-store / device *

column-store / host & multi-threaded

*Device transfer costs excluded

Setup
TPC-C benchmark customer record 96B (21 fields) / item record 20B + 8B (4 fields + price field), system configuration operator-at-a-time processing w/ late materialization, host: max. 8 threads blockwise partitioning, device: optimized parallel reduction kernel (>= 1024 blocks w/ 512 threads), final reduction on 1 block w/ 1024 threads, effort for join processing not incl.
```
Hybridization of HTAP and Heterogenous Computing

First: Is there performance potential?

„OLTP“ query materialization
150 customers
row-store / host & single-threaded

„HTAP“ query aggregation of some
150 items
column-store / host & single-threaded

„OLAP“ query aggregation of all
all items
column-store / device

column-store / host & multi-threaded

Storage Engine
Row-Store change Column-Store
Host switch

Column-Store
Device

Query Engine
Inter-query parallelism
Inter-query parallelism switch
Intra-query parallelism
To take advantage, we need to double hybridize to have the best of both worlds.
Contribution
Survey and Classification of SOTA Engines

Taxonomy

We defined a fine-grained set of concepts and relations between them in order to classify systems
- Unified terms to compare different systems
- Overview on design space
- Possibility to assess adaptability w.r.t. transitions

Core concepts
- Layout. Division of a relation in terms of fragments.
- Fragment. „Gapless“ region of data.
- Tuple. Portion of tuple that fell into a fragment.
# Survey and Classification of SOTA Engines

## Classification

<table>
<thead>
<tr>
<th>HTAP systems</th>
<th>Co-processor systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>- OLAP or OLTP</td>
<td>- OLAP or OLTP</td>
</tr>
<tr>
<td>- OLAP and OLTP</td>
<td>- OLAP and OLTP</td>
</tr>
<tr>
<td>- co-processor support</td>
<td>- co-processor support</td>
</tr>
<tr>
<td>- multiple layout support</td>
<td>- multiple layout-support</td>
</tr>
<tr>
<td>- advanced flexibility for layout</td>
<td>- advanced flexibility for layout</td>
</tr>
<tr>
<td>- adaptive layouts during runtime</td>
<td>- adaptive layouts during runtime</td>
</tr>
<tr>
<td>- mixed NSM/DSM support</td>
<td>- mixed NSM/DSM support</td>
</tr>
</tbody>
</table>
Survey and Classification of SOTA Engines

View on Results

Device HTAP

Heterogenous HTAP

compute device support

host/device

host

HTAP

workload type support

OLAP

restrictions to fragments

strong flexible

weak flexible

inflexible

layout adaptability

static

responsive (during runtime)

compute device support

workload type support

Heterogenous HTAP

Device HTAP

GPUTx

CoGaDB

PAX

Frac. Mirrors

HYRISE

ES²

L-Store

H₂O

HyPer

Peloton

year

2002

2016
Future Work
Self-managing multi-model HTAP database system on CPU/GPUs

Codename Vector Pipes
- Query Engine
- Micro-batch query execution
- UDFs as first-class citizen
- Multi query execution

Codename Grid Store
- Storage Engine
- Adaptive HTAP storage
- Heterogenous platforms
- Advanced-Tile based

Codename Alfred
- Optimizer
- Self-managing component
- Learning on event streams

MondrianDB

X100

PIPES MapReduce

Peloton L-Store CoGaDB
Grid Store: A Storage Engine for Heterogenous HTAP

Wrap Up and Outlook. Feedback is welcome

<table>
<thead>
<tr>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. different data access pattern</td>
</tr>
<tr>
<td>2. adapt record layout (NSM, DSM,...)</td>
</tr>
<tr>
<td>3. interference between query types</td>
</tr>
<tr>
<td>4. contradicting optimization goals</td>
</tr>
<tr>
<td>5. different types of parallelism</td>
</tr>
<tr>
<td>6. hot and cold data parts</td>
</tr>
</tbody>
</table>

HTAP

- data transfer costs for I/O
- different programming models
- device limitations (e.g., memory capacity)
- data and operator placement

Heter.
Grid Store: A Storage Engine for Heterogenous HTAP

Wrap Up and Outlook. Feedback is welcome

Example table layout in Grid Store
*(unconstrained)* strong flexible responsive layout w/ data placement

- cold data (compressed)
- hot data, stored on device (NSM)
- hot data, stored on device (DSM)
- hot data, stored on host (NSM)
- hot data, stored on host (DSM)
- reasonability currently under study

Built tables from *arbitrary tiles* (grids) which...
- ... may live on host or device (or both)
- ... are self-contained (NSM/DSM/Indexed/...)
- ... mutable w.r.t. to access pattern and shape

Table layouts respond online to forecasted workload changes
- shrink or enlarge grids, compress/uncompress
- move grids (temporary, permanent) to platform

Backed by *flexible query engine* (Vector Pipes) and *adaptive self-leaning optimizer* on real-time analytics of system event stream (Alfred)
**Grid Store: A Storage Engine for Heterogenous HTAP**

Wrap Up and Outlook. Feedback is welcome

**Challenges**

1. different data access pattern
2. adapt record layout (NSM, DSM, ...)
   - interference between query types
   - contradicting optimization goals
3. hot and cold data parts
4. data transfer costs for I/O
5. different programming models
6. device limitations (e.g., memory capacity)
7. data and operator placement

**HTAP**

- built tables by *arbitrary tiles (grids)* which...
  - ... may live on host or device (or both)
  - ... are self-contained (NSM/DSM/Indexed/...)
  - ... mutable w.r.t. to access pattern and shape

**Heter.**

- Table layouts respond online to forecasted workload changes
  - shrink or enlarge grids, compress/uncompress
  - move grids (temporary, permanent) to platform

Backed by *flexible query engine* (Vector Pipes) and *adaptive self-leaning optimizer on real-time analytics of system event stream* (Alfred)