Tensor Query Processing: How do we ride the AI investment wave for database analytics?

Carlo Curino
Microsoft Azure Data
Gray Systems Lab
Tensor Query Processing: How do we ride the AI investment wave for database analytics? Ok, Skynet is coming after the human race... but can we run queries on a T850?

Carlo Curino

Microsoft Azure Data
Gray Systems Lab

Image courtesy of pngimg.com
Tensor Query Processing: How do we ride the AI investment wave for database analytics? Ok, Skynet is coming after the human race... but can we run queries on a T850 A100?

Carlo Curino
Gray Systems Lab
Backdrop: **DB perf gains are hard to come by!**

- Slowing HW-driven perf improvements
- Decades of optimization saturated SW gains
Backdrop: **AI interest (and HW) is exploding!**

Big $$$ spent on Special HW for NN

VCs are pouring $2B/quarter

Market expected to exceed $200B/year by 2025.
Some examples of **AI HW**

<table>
<thead>
<tr>
<th>Type</th>
<th>Tech (nm)</th>
<th>Architecture</th>
<th># Trans.</th>
<th>Power</th>
<th>Cache</th>
<th>Mem / Storage</th>
<th>Mem BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN-Chips</td>
<td>7</td>
<td>Cerebras WSE-2</td>
<td>2.6 T</td>
<td>20 KW</td>
<td>40 GB</td>
<td>4 TB - 2.4 PB</td>
<td>20 PB/sec</td>
</tr>
<tr>
<td>GPU</td>
<td>7</td>
<td>NVIDIA A100</td>
<td>54 B</td>
<td>400 W</td>
<td>40 MB</td>
<td>40/80 GB</td>
<td>1.6 TB/sec (HBM)</td>
</tr>
</tbody>
</table>

* Data courtesy of Rathijit Sen
Tensor Runtime

Tensor as de-facto API
Very large/active communities
What about HW investments for Database?

- Existent but modest w.r.t AI
- Porting to each new HW is a costly N x M problem
Our Goal: **Save Humanity from Skynet!**

How? By keeping the AI HW busy with database analytics queries
Main Idea: **Tensor Query Processing**

Compile SQL, Classical ML, etc. to the popular tensor abstraction!
Pros and Cons of "Tensor Query Processing"

Pros

✓ Leverage the massive investments in special HW
✓ Scalable Approach (tensor runtimes are getting ported to each new HW)

Cons

? Is this even possible?
? What about performance? (as compared with state-of-the-art)
? How expensive is it going to be? (engineering wise)
System Design

SQL Query

```sql
SELECT
    MAX(p_supplycost) AS price,
    s_name AS supp
FROM supplier
JOIN partsupp
ON
    ps_suppkey=s_suppkey
GROUP BY
    supplier.s_name
ORDER BY
    price DESC;
```
SELECT MAX(p_supplycost) AS price, s_name AS supp
FROM supplier
JOIN partsupp ON ps_suppkey = s_suppkey
GROUP BY supplier.s_name
ORDER BY price DESC;
System Design

SELECT MAX(p_supplycost) AS price,
      s_name AS supp
FROM supplier
JOIN partsupp
ON ps_suppkey=s_suppkey
GROUP BY supplier.s_name
ORDER BY price DESC;

SQL Query

Parsing Layer

Planning Layer

IR Graph

Operator Plan

Physical Plan

Tensor program for Sort
Tensor program for Join
Tensor program for Filter
System Design

SQL Query

`SELECT MAX(p_supplycost) AS price, s_name AS supp
FROM supplier
JOIN partsupp
ON ps_suppkey=s_suppkey
GROUP BY supplier.s_name
ORDER BY price DESC;`

Parsing Layer

IR Graph

Planning Layer

Operator Plan

Tensor Runtime

CPU

GPU

TPU

Mobile

Browser

Physical Plan

Tensor program for Sort
Tensor program for Join
Tensor program for Filter
Example: Tensor Program for Filter

```
WHERE L_QUANTITY < 24

Opt 1:
1. mask = torch.lt(l_quantity, 24)
2. output = torch.masked_select(l_quantity, mask)

Opt 2:
1. mask = torch.lt(l_quantity, 24)
2. idx = torch.nonzero(mask)
3. output = torch.index_select(l_quantity, dim=0, idx)
```

Numeric as N x 1 tensors
Dates as N x 1 numeric
Strings as UTF-8 N x max_length
Implementing **SQL operators using tensor ops**

Tensor operators required for TPC-H

Tensor Operations (about 60 in total)
Pros and Cons of "Tensor Query Processing"

Pros
- Leverage the massive investments in special HW
- Scalable Approach (tensor runtimes are getting ported to each new HW)

Cons
- Is this even possible?
- What about performance? (as compared with state-of-the-art)
- How expensive is it going to be? (engineering wise)
Pros and Cons of "Tensor Query Processing"

**Pros**

- Leverage the massive investments in special HW
- Scalable Approach (tensor runtimes are getting ported to each new HW)
- Is this even possible? → YES we can easily cover TPC-H

**Cons**

- What about performance? (as compared with state-of-the-art)
- How expensive is it going to be? (engineering wise)
TPCH SF 50

SQL Server and DataBricks: Standard D64s v5 (64 vcpus, 256 GiB memory)
TQP: Standard NC24ads A100 v4 (24 vcpus, 220 GiB memory)
about 50% more expensive than the CPU HW
Buffer pools and TQP Scalability

Total Runtime (seconds) vs Scale Factor for A100 40GB and A100 80GB.

- Q21 spills
- Q1 spills
- Q21 spills
- Q1 spills
- Q21 OOM
- OOM
Is this a one-time gain?

More perf coming from HW improvements

Lots of headroom via SW optimization

HW Generation

P100 (2016)  V100 (2017-18)  A100 (2020)  H100 (2022)

HW perf gain (ratio)

HW Generation

TQP  TQP + fusion  TQP + fusion + encoding  TQP + fusion + encoding + Antares  Best HW can do

SW perf gain (ratio)
TPCH SF 50 drilldown

Can we do better?
Missing Optimizations

WHERE L_QUANTITY < 24

Opt 1:
1. mask = torch.lt(l_quantity, 24)
2. output = torch.masked_select(l_quantity, mask)

Opt 2:
1. mask = torch.lt(l_quantity, 24)
2. idx = torch.nonzero(mask)
3. output = torch.index_select(l_quantity, dim=0, idx)

Ongoing:
HW-customized operators
Operator Fusion
Representation / compression
Co-execution of CPU/GPU
IO-bottlenecks / Distributed exe

Future:
Tensor-aware Optimizer
TQP: Computing on RLE compressed data

QP directly on data encoded for compression
• Raw datasets can be larger than GPU memory
• Lower data transfer overhead
• Faster query processing

Encodings: dictionary, RLE, bit-packing, ...

<table>
<thead>
<tr>
<th>TQP/Vanilla</th>
<th>Dict</th>
<th>RLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rome, Rome, Rome, Rome, Rome, Tokyo, Tokyo, Tokyo, Tokyo, Los Angeles, Los Angeles, Los Angeles, Los Angeles, Rome</td>
<td>Rome:00, Tokyo:01, Los Angeles:10</td>
<td>00, 0, 01, 4, 10, 7, 00, 11</td>
</tr>
</tbody>
</table>
Optimizing TQP: Fusion + Portable custom operators

Antares is a Cross-compiling Engine

https://github.com/microsoft/antares
Interesting HW configuration: APU design where CPU and GPU share HBM (no PCI-e)

Predictable usage pattern: gamers mostly play during the evenings (AKA dark time)
GPU: NVIDIA P100  
Xbox: Series X  
CPU: Xeon E5-2690 (14 cores)

<table>
<thead>
<tr>
<th></th>
<th>Xeon E5-2690</th>
<th>P100</th>
<th>Xbox Series X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Bandwidth (GB/s)</td>
<td>154</td>
<td>732</td>
<td>560</td>
</tr>
<tr>
<td>Unidirectional PCIe3 (GB/s)</td>
<td>-</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>Theoretical TFLOps</td>
<td>1.4</td>
<td>9.5</td>
<td>12.0</td>
</tr>
</tbody>
</table>
Pros and Cons of "Tensor Query Processing"

**Pros**

- Leverage the massive investments in special HW
- Scalable Approach (tensor runtimes are getting ported to each new HW)
- Is this even possible?

**Cons**

- What about performance? (as compared with state-of-the-art)
- How expensive is it going to be? (engineering wise)
Pros and Cons of “Tensor Query Processing”

**Pros**

- Leverage the massive investments in special HW
- Scalable Approach (tensor runtimes are getting ported to each new HW)
- Is this even possible?
- What about performance? (as compared with state-of-the-art)

**Cons**

- How expensive is it going to be? (engineering wise)
Pros and Cons of "Tensor Query Processing"

**Pros**
- Leverage the massive investments in special HW
- Scalable Approach (tensor runtimes are getting ported to each new HW)
- Is this even possible?
- What about performance? (as compared with state-of-the-art)

**Cons**
- How expensive is it going to be? (engineering wise)
Pros and Cons of "Tensor Query Processing"

Pros

- Leverage the massive investments in special HW
- Scalable Approach (tensor runtimes are getting ported to each new HW)
- Is this even possible?
- What about performance? (as compared with state-of-the-art)
- How expensive is it going to be? (engineering wise)

Less than 20k LoC
Compile the TPC-H 5 query

```python
query = ""
  N_NAME,
  sum(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) as revenue
from
customer,
  orders,
  lineitem,
  supplier,
  nation,
  region
where
  C_CUSTKEY = O_CUSTKEY
and
  L_ORDERKEY = O_ORDERKEY
and
  L_SUPPKEY = S_SUPPKEY
and
  L_NATIONKEY = N_NATIONKEY
and
  L_REGIONKEY = R_REGIONKEY
and
  L_NAME = 'ASIA'
and
  O_ORDERDATE > date '1994-01-01'
and
  O_ORDERDATE < date '1994-01-01' + interval '1' year
group by
  N_NAME
order by
  revenue desc"
```

Register the table (Dataframe whether in Spark on Pandas format)
Future directions

1. Continue Perf Work (especially around I/O and multi-GPU)
2. Broader applicability (e.g., Classical ML with Hummingbird project*, pagerank)
3. Multimodal SQL (+ Differentiable SQL) for unstructured data input → single (optimizable) tensor program!

* GitHub - microsoft/hummingbird: Hummingbird compiles trained ML models into tensor computation
Teaser: Multi-modal query support

Broader implications of having a DBMS co-existing with an ML runtime

SELECT
    input AS images,
    image_text_similarity_model("KFC Receipt", input) AS score
FROM attachments
ORDER BY score DESC
LIMIT 1
Compile the TPC-H 5 query

```python
query = """select
N_NAME,
sum(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) as revenue
from
customer,
orders,
lineitem,
supplier,
nation,
region
where
C_CUSTKEY = o_custkey
and L_ORDERKEY = O_ORDERKEY
and L_SUPKEY = S_SUPKEY
and L_NATIONKEY = N_NATIONKEY
and N_REGIONKEY = r_regionkey
and n_name = 'ASIA'
and O_ORDERDATE <= date '1994-01-01'
and O_ORDERDATE > date '1994-01-01' + interval '1' year
group by
N_NAME
order by
revenue desc"""
```

Register the table (Dataframe whether in Spark on Pandas format)
Conclusion

Save humanity from Skynet!

How? Keep it busy running SQL (compiled to Tensors)

- Free-ride on AI investments
- Great perf/cost trade-offs
- Fun future directions
Join us... let’s invent the future together!

https://aka.ms/gsl
Gray Systems Lab (GSL)

GSL is an applied and embedded research group, comprised of Data-Scientists, Engineers and Researchers.