YourSQL: A High-Performing Database System Leveraging In-Storage Computing

I. Jo et al, VLDB 2016
Presented by Nishad Phadke

April 18, 2017
Motivation

- YourSQL is designed for data-intensive database queries
- Intuition: ↓ data transferred ⇒ ↑ query speed
- *Early filtering* is data-intensive but non-complex
- Number of I/O requests is largest contributor to cost of data-intensive operations
Motivation

- YourSQL is designed for data-intensive database queries
- Intuition: ↓ data transferred ⇒ ↑ query speed
- *Early filtering* is data-intensive but non-complex
- Number of I/O requests is largest contributor to cost of data-intensive operations
In-storage computing (ISC) in Solid State Drives (SSDs)

Figure: From Bae, et al.[1]
SSD photo from YourSQL

Figure 6: Dell R720 server with PM1725 SSD.
Database Architecture
Definitions

Query 1: A simple selection query.

```
SELECT p_partkey, p_mfgr
FROM part
WHERE p_size = 15 AND p_type LIKE '%BRASS';
```

- **Selectivity**: fraction of relevant rows from row set
  - 0 is highest (most selective)
  - 1 is lowest
  - ↑ selectivity ⇝ ↑ query speed

- **Filtering ratio**: fraction of relevant pages from page set
  - Filtering ratio is much better indicator of speed-up guarantees
Definitions

Query 1: A simple selection query.

```
SELECT p_partkey, p_mfgr FROM part
WHERE p_size = 15 AND p_type LIKE '%BRASS';
```

- **Selectivity**: fraction of relevant rows from row set
  - 0 is highest (most selective)
  - 1 is lowest
  - $\uparrow$ selectivity $\not\Rightarrow\uparrow$ query speed

- **Filtering ratio**: fraction of relevant pages from page set
  - Filtering ratio is much better indicator of speed-up guarantees
Query 1

Query 1: A simple selection query.

```
SELECT p_partkey, p_mfgr FROM part
WHERE p_size = 15 AND p_type LIKE '%BRASS';
```

What if the WHERE predicates are indices?
Index Condition Pushdown (ICP)¹

¹https://mariadb.com/kb/en/mariadb/index-condition-pushdown/
Query 1 (again)

**Query 1: A simple selection query.**

```sql
SELECT p_partkey, p_mfgr FROM part
WHERE p_size = 15 AND p_type LIKE '%BRASS';
```

What if the WHERE predicates are **not** indices?
Filtering Condition Pushdown (FCP)

Figure 4: ISC filters.
Hardware filtering

\[ \mathcal{X} = \text{lots of dates between 1990 and 2020} \]

\[ M_{2016,9} = \text{dates in 2016 September} \]

\[ M_{2016,10} = \text{dates in 2016 October} \]

\[ M_{2016,11} = \text{dates in 2016 November} \]

\[ Y_{2016} = \text{dates in 2016} \]

\[ \left[ \mathcal{X} \cap M_{2016,9} \right] \cup \left[ \mathcal{X} \cap M_{2016,10} \right] \cup \left[ \mathcal{X} \cap M_{2016,11} \right] \subseteq \left[ \mathcal{X} \cap Y_{2016} \right] \]
Hardware filtering

\[ \mathcal{X} = \text{lots of dates between 1990 and 2020} \]
\[ M_{2016,9} = \text{dates in 2016 September} \]
\[ M_{2016,10} = \text{dates in 2016 October} \]
\[ M_{2016,11} = \text{dates in 2016 November} \]
\[ Y_{2016} = \text{dates in 2016} \]

\[ [\mathcal{X} \cap M_{2016,9}] \cup [\mathcal{X} \cap M_{2016,10}] \cup [\mathcal{X} \cap M_{2016,11}] \subseteq [\mathcal{X} \cap Y_{2016}] \]
Hardware filtering

**Query 3: TPC-H Q14.**

```sql
SELECT 100.00 * SUM(CASE WHEN p_type LIKE 'PROMO%' THEN l_extendedprice * (1-l_discount) ELSE 0 END) / SUM(l_extendedprice * (1-l_discount)) AS promo_revenue
FROM lineitem, part
WHERE l_partkey = p_partkey
  AND l_shipdate >= date '1995-09-01'
  AND l_shipdate < date '1995-09-01' + INTERVAL '1' MONTH;
```

date >= '1995-09-01' AND date < '1995-10-01'
date >= 0x8F9721 AND date < 0x8F9741
date.startswith(0x8F97)
Hardware filtering


```
SELECT 100.00 * SUM(CASE WHEN p_type LIKE 'PROMO%' THEN l_extendedprice * (1-l_discount) ELSE 0 END) /
SUM(l_extendedprice * (1-l_discount)) AS promo_revenue
FROM lineitem, part
WHERE l_partkey = p_partkey
  AND l_shipdate >= date '1995-09-01'
  AND l_shipdate < date '1995-09-01' + INTERVAL '1' MONTH;
```

date >= '1995-09-01' AND date < '1995-10-01'
date >= 0x8F9721 AND date < 0x8F9741
date.startswith(0x8F97)
Hardware filtering


```sql
SELECT 100.00 * SUM(CASE WHEN p_type LIKE 'PROMO%' THEN l_extendedprice * (1 - l_discount) ELSE 0 END) /
SUM(l_extendedprice * (1 - l_discount)) AS promo_revenue
FROM lineitem, part
WHERE l_partkey = p_partkey
AND l_shipdate >= date '1995-09-01'
AND l_shipdate < date '1995-09-01' + INTERVAL '1' MONTH;
```

date >= '1995-09-01' AND date < '1995-10-01'
date >= 0x8F9721 AND date < 0x8F9741
date.startswith(0x8F97)
Hardware filtering inherent limitations

\[ \mathcal{X} = \text{lots of dates between 1990 and 2020} \]
\[ M_{2016,12} = \text{dates in 2016 December} \]
\[ M_{2017,1} = \text{dates in 2017 January} \]

\[ [\mathcal{X} \cap M_{2016,12}] \cup [\mathcal{X} \cap M_{2017,1}] \subseteq [\mathcal{X} \cap Y??] \]
Hardware filtering

\[
\begin{align*}
\mathcal{X} \cap M_{2016,9} & \cup \mathcal{X} \cap M_{2016,10} & \cup & \mathcal{X} \cap M_{2016,11} & \subseteq & \mathcal{X} \cap Y_{2016} \\
\mathcal{X} \cap M_{2016,12} & \cup & \mathcal{X} \cap M_{2017,1} & \subseteq & \mathcal{X} \cap Y_{??}
\end{align*}
\]

False positives? Possibly.
False negatives? No.
Filtering Condition Pushdown (FCP)

Figure 4: ISC filters.
Software filtering

- Only activated in some cases (depending on filter query)
- Removes (un-matches) pages that are 100% false positives

---

**Code 1: Pseudo code of the software filter.**

```plaintext
//IN: Match_Hint
//ARG: SW.KEY, OP.TYPE, BULK.MATCH.SIZE
//OUT: Match_Hint

DO Get Match_Hint from the hardware filter
FOR i = 0 TO i < BULK.MATCH.SIZE DO {
    IF Match_Hint[i] != 0 THEN {
        Do Match_Hint[i] to 0
        FOR j = 0 TO j < # of rows in Page i DO {
            FOR each filter column in row j DO {
                IF 'the column value OP.TYPE SW.KEY' is true THEN
                    Do Set Match_Hint[i] to 1
                    Do Break
                END IF
            }
            IF Match_Hint[i] == 1 THEN
                Do Break
            END IF
        }
    }
}
DO Put Match_Hint to output
```
YourSQL architecture

![Diagram of YourSQL architecture](image)

**Figure 1:** Two database system architectures. (a) Traditional system. (b) YourSQL.
ISC-enabled queries

![Diagram of ISC-enabled queries]

**Figure 2: Early filtering of YourSQL.**
Prefetcher

- Bulk read from match hints
- Choosing the size of `bulk_match_size` (number of match-hinted pages) is important
  - SSD can concurrently process requests up to a certain `bulk_match_size`
Complex queries

Query 1: A simple selection query.

```sql
SELECT p_partkey, p_mfgr
FROM part
WHERE p_size = 15 AND p_type LIKE '%%BRASS';
```

Query 2: TPC-H Q2.

```sql
SELECT s_acctbal, s_name, n_name, p_partkey, p_mfgr,
       s_address, s_phone, s_comment
FROM part, supplier, partsupp, nation, region
WHERE p_partkey = ps_partkey
  AND s_suppkey = ps_suppkey
  AND p_size = 15 AND p_type LIKE '%%BRASS'
  AND s_nationkey = n_nationkey
  AND n_regionkey = r_regionkey
  AND r_name = 'EUROPE'
  AND ps_supplycost = (SELECT MIN(ps_supplycost)
                        FROM partsupp, supplier, nation, region
                       WHERE p_partkey = ps_partkey
                         AND s_suppkey = ps_suppkey
                         AND s_nationkey = n_nationkey
                         AND n_regionkey = r_regionkey
                         AND r_name = 'EUROPE')
ORDER BY s_acctbal DESC, n_name, s_name, p_partkey LIMIT 100;
```
Goal

Put the smallest table first in JOIN order

- Nested-loop algorithm
- Why smallest first?

Query 2: TPC-H Q2.

```sql
SELECT s acctbal, s name, n name, p partkey, p mgr,
       s address, s phone, s comment
FROM part, supplier, partsupp, nation, region
WHERE p partkey = ps partkey
  AND s suppkey = ps suppkey
  AND p size = 15 AND p type LIKE 'BRASS'
  AND s nationkey = n nationkey
  AND n regionkey = r regionkey
  AND r name = 'EUROPE'
  AND ps supplycost = (SELECT MIN(ps supplycost)
                        FROM partsupp, supplier, nation, region
                        WHERE p partkey = ps partkey
                        AND s suppkey = ps suppkey
                        AND s nationkey = n nationkey
                        AND n regionkey = r regionkey
                        AND r name = 'EUROPE')
ORDER BY s acctbal DESC, n name, s name, p partkey LIMIT 100;
```
Estimating I/O reduction per table

Nominate by heuristic → accept/reject by estimate

- Limiting score (heuristic)
  - Threshold set presumably through experimentation

- Estimated filter ratio
  - Much more complex
  - Is quantitatively correlated with actual filtering ratio
  - Requires FCP
Estimating I/O reduction per table

Nominate by heuristic $\rightarrow$ accept/reject by estimate

- Limiting score (heuristic)
  - Threshold set presumably through experimentation

- Estimated filter ratio
  - Much more complex
  - is quantitatively correlated with actual filtering ratio
  - Requires FCP
Estimating I/O reduction per table

Nominate by heuristic → accept/reject by estimate

- Limiting score (heuristic)
  - Threshold set presumably through experimentation

- Estimated filter ratio
  - Much more complex
  - is quantitatively correlated with actual filtering ratio
  - Requires FCP
Best-guessing smallest table

- List all the tables with filter predicates
- Eliminate small tables
- Calculate the limiting score of each remaining table
- Eliminate the tables whose limiting score is below a given threshold
- Select the table with the highest limiting score as the candidate
- Estimate the filtering ratio of the candidate by the ISC sampler
- Determine the candidate as the target if the estimated filtering ratio is sufficiently high

Figure 3: Selection of the early filtering target table.
ISC Sampler

Figure 2: Early filtering of YourSQL.
Bandwidth comparisons

Figure: From Gu, et al.[2]
Speed of queries

<table>
<thead>
<tr>
<th>Table 4: Execution time for Query 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>MySQL</td>
</tr>
<tr>
<td>Execution time [sec]</td>
</tr>
</tbody>
</table>

- 7× speed up
- 6.7% of pages needed to answer query

<table>
<thead>
<tr>
<th>Query 1: A simple selection query.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT p_partkey, p.mfgr FROM part</td>
</tr>
<tr>
<td>WHERE p.size = 15 AND p.type LIKE 'BRASS';</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5: Execution time for TPC-H Q2.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>MySQL</td>
</tr>
<tr>
<td>Execution time [sec]</td>
</tr>
</tbody>
</table>

- 44× speed up

<table>
<thead>
<tr>
<th>Query 2: TPC-H Q2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT s_acctbal, s_name, n_name, p_partkey, p.mfgr, s_address, s_phone, s_comment</td>
</tr>
<tr>
<td>FROM part, supplier, partsupp, nation, region</td>
</tr>
<tr>
<td>WHERE p_partkey = ps_partkey</td>
</tr>
<tr>
<td>AND s_suppkey = ps_suppkey</td>
</tr>
<tr>
<td>AND p.size = 15 AND p.type LIKE 'BRASS'</td>
</tr>
<tr>
<td>AND s.nationkey = n.nationkey</td>
</tr>
<tr>
<td>AND n.regionkey = r.regionkey</td>
</tr>
<tr>
<td>AND r.name = 'EUROPE'</td>
</tr>
<tr>
<td>AND ps_supplycost = (SELECT MIN(ps_supplycost)</td>
</tr>
<tr>
<td>FROM partsupp, supplier, nation, region</td>
</tr>
<tr>
<td>WHERE p_partkey = ps_partkey</td>
</tr>
<tr>
<td>AND s_suppkey = ps_suppkey</td>
</tr>
<tr>
<td>AND s.nationkey = n.nationkey</td>
</tr>
<tr>
<td>AND n.regionkey = r.regionkey</td>
</tr>
<tr>
<td>AND r.name = 'EUROPE')</td>
</tr>
<tr>
<td>ORDER BY s_acctbal DESC, n_name, s_name, p_partkey LIMIT 100;</td>
</tr>
</tbody>
</table>
Power consumption

Table 5: Execution time for TPC-H Q2.

<table>
<thead>
<tr>
<th>Execution time [sec]</th>
<th>MySQL</th>
<th>YourSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,104</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

▶ 44× speed up

Table 7: Overall energy consumption.

<table>
<thead>
<tr>
<th></th>
<th>MySQL</th>
<th>YourSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total energy (kJ)</td>
<td>131.0</td>
<td>5.3</td>
</tr>
</tbody>
</table>

Figure 10: System power consumption during the execution of TPC-H Q2.
Effect of system memory size

Figure 7: Speed-up with different memory size.
TCP-H queries

Figure 8: TPC-H results.
Impact of optimization

![Graph showing speed-up of top five accelerated queries with different optimization schemes.]

**Figure 9:** Speed-ups of the top five accelerated queries with different optimization schemes.

**Table 6: Different levels of optimization.**

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opt-P</td>
<td>Hardware filter</td>
</tr>
<tr>
<td>Opt-PS</td>
<td>Hardware filter + Software filter</td>
</tr>
<tr>
<td>Opt-PSH</td>
<td>Hardware filter + Software filter + HABP</td>
</tr>
</tbody>
</table>

HABP == prefetching
Column store

*Lower (\textit{i.e.}, shorter bars) is better

Figure 11: Execution time for the column stores.