Hardware-Sensitive Scan Operator Variants for Compiled Selection Pipelines

David Broneske, Andreas Meister, Gunter Saake
University of Magdeburg
Introduction Query
Compilation

\[ \sum (A \times B) \]

\[ \sigma \text{lo}_\text{ord}date = \text{d}_\text{date}key \]

\[ \sigma \text{d}_\text{year} = 1993 \]

\[ \sigma \text{lo}_\text{discount} \_\ldots \_\text{lo}_\text{quantity} \]

Dates

Lineorder
Introduction Query Compilation

\[ \text{sum}(A \times B) \]

\[ \text{lo\_discount} \ldots, \text{lo\_quantity} \]

\[ \text{lo\_orderdate} = \text{d\_datekey} \]

\[ \text{d\_year} = 1993 \]

Dates \quad Lineorder
Introduction Query Compilation

Bandwidth-bound $\rightarrow$ compute-bound

Possibility for code optimizations
Motivating Examples
Motivating Examples

Branching

```
for(int i = 0; i < input_size; ++i){
  if(col[i] < pred)
    agg+=agg_col[i];
}
```
Motivating Examples

Branching

```plaintext
for(int i = 0; i < input_size; ++i){
    if(col[i] < pred)
        agg+=agg_col[i];
}
```

Predicated

```plaintext
for(int i = 0; i < input_size; ++i){
    agg+=agg_col[i]*(col[i] < pred);
}
```
Motivating Examples

**Branching**

```c
for(int i = 0; i < input_size; ++i){
    if(col[i] < pred)
        agg+=agg_col[i];
}
```

**Predicated**

```c
for(int i = 0; i < input_size; ++i){
    agg+=agg_col[i]*(col[i] < pred);
}
```

**SIMD [ZR02]**

```c
for(int i = 0; i < simd_size; ++i){
    mask= SIMD_COMP(simd_col[i],pred);
    if(mask){
        for (int j=0; j < SIMD_LENGTH;++j){
            if((mask >> j) & 1)
                agg+=agg_col[i];
        }
    }
}
```
Motivating Examples

Branching

```
for(int i = 0; i < input_size; ++i){
    if(col[i] < pred)
        agg+=agg_col[i];
}
```

Predicated

```
for(int i = 0; i < input_size; ++i){
    agg+=agg_col[i]*(col[i] < pred);
}
```

**SIMD** [ZR02]

```
for(int i = 0; i < simd_size; ++i){
    mask= SIMD_COMP(simd_col[i],pred);
    if(mask){
        for (int j=0; j < SIMD_LENGTH;++j){
            if((mask >> j) & 1)
                agg+=agg_col[i];
        }
    }
}
```
Motivating Examples

Graph showing response time in ms against selectivity for different scan methods:

- Branching Scan
- SIMD Scan
- Predicated Scan
- Predicated SIMD Scan

Legend:

- Branching Scan
- SIMD Scan
- Predicated Scan
- Predicated SIMD Scan
Motivating Examples

a) Single Predicate

b) Query Q1

c) Query Q6

Branching Scan — SIMD Scan — Predicated Scan — Predicated SIMD Scan
Motivating Examples

8 Aggregates
1 Filter Predicate

a) Single Predicate

b) Query Q1

c) Query Q6

Branching Scan
SIMD Scan
Predicated Scan
Predicated SIMD Scan
Motivating Examples

8 Aggregates
1 Filter Predicate

1 Aggregate
3 Filter Predicates

a) Single Predicate

b) Query Q1

c) Query Q6

Selectivity: 0.2 0.4 0.6 0.8 1

response time in ms:

Branching Scan
SIMD Scan
Predicated Scan
Predicated SIMD Scan
Motivating Examples

When to use which scan variant?

- Branching Scan
- SIMD Scan
- Predicated Scan
- Predicated SIMD Scan
Evaluation Setup

Evaluation Criteria

* Number of predicates
* Number of aggregates inside loop

Workload & Machine

* TPC-H LineItem table SF 10
* Intel Xeon E5-2630 v3 with SSE4.2

Variants:

* Branching vs. Predication
* Scalar vs. SIMD
Number of Predicates

Branching Scan

Selectivity P1

# of Predicates

Time in ms

0 0.5 1 5 10

0 200 400

6
Number of Predicates

- Branching Scan
- SIMD Scan
- Predicated Scan
- SIMD Predicated Scan

Selectivity P1

Time in ms

Number of Predicates

Selectivity P1

# of Predicates

Branching Scan

SIMD Scan

Predicated Scan

SIMD Predicated Scan
Number of Predicates

Results:
* For one predicate SIMD does not pay out
Number of Predicates

Results:
* For one predicate, SIMD does not pay out
* The more predicates, the better SIMD
Work Inside the Loop

Branching Scan

![3D Graph of Branching Scan]

SIMD Scan

![3D Graph of SIMD Scan]

Predicated Scan

![3D Graph of Predicated Scan]

SIMD Predicated Scan

![3D Graph of SIMD Predicated Scan]
Work Inside the Loop

Results:
* More aggregates, less impact of branch misprediction
Work Inside the Loop

Results:

- More aggregates, less impact of branch misprediction
- The more aggregates, the better branching scans for low selectivity
Decision Trees

Number of Predicates

- #predicates
  - < 4
    - Branching Scan
  - >= 4
    - SIMD Branching
  - < 2
    - SIMD Predicated
  - >= 2
    - Predicated Scan

Number of Aggregates

- #aggregates
  - < 6
    - SIMD Branching
  - >= 6
    - SIMD Predicated
  - < 0.05
    - selectivity
  - >= 0.05
    - selectivity
  - < 0.1
    - SIMD Predicated
  - >= 0.1
    - SIMD Branching
Conclusion

- Increasing number of aggregates slows down predicated variants
- SIMD outperforms scalar variants for several predicates
- Pipeline code for filter- & aggregate pipelines\(^1\)
- Decision trees as a result of our evaluation in the paper

Future Work

- Hash table put / probe (joins, groupings)
- Automatic calibration for query compilation

\(^1\)http://git.iti.cs.ovgu.de/dbronesk/BTW-Pipeline-Variants
References


Selectivity of Two Predicates

- **Conditional AND**
  - selectivity1
    - < 0.05
    - >= 0.05

- **Bitwise AND**
  - selectivity2
    - < 0.05
    - >= 0.05

- **SIMD Predicated**
Selectivity of Two Predicates

Conditional AND Scan

Bitwise AND Scan

SIMD Scan

Predicated Scan

SIMD Predicated Scan