A Cost Model for Data Stream Processing on Modern Hardware

Constantin Pohl, Philipp Götze, Kai-Uwe Sattler
Motivation and Introduction

• Main goals on Data Stream Processing Queries: High throughput & low latency

• Responsibility: Optimizer, based on cost model

• New challenges and opportunities in modern HW (like Manycore CPUs or NVRAM technology)

• Cost model with most important HW factors and Stream Processing Engine (SPE) characteristics
General idea (Optimizer)

• Hardware with generally valid properties, e.g.
  ➢ CPU: clock frequency, available cores/threads,…
  ➢ Memory: size, latency,…

• Hardware calibration tool for measuring relevant factors on different systems in advance

• Known measurements & memory accessing costs for SPE operators (joins, aggregations, etc.)

• Result: Cost model for parametrization avoiding worst-case performance
Query Costs

- Costs expressed as „necessary time to process a certain amount of tuples“
- Transformation into „avg. latency per processed tuple“
- Single thread for query

\[
c_q = \sum_{i=1}^{n} (c_{op(i)} \cdot t_{p_{in}})
\]

- \(1...n\): number of query operators
- \(t_{p_{in}}\): number of processed tuples
Inter-Operator Parallelism

Costs: \[ c_q = \max \left( \sum_{i=1}^{k} (c_{op(i)} \cdot tp_{in}), \sum_{i=k+1}^{n} (c_{op(i)} \cdot tp_{in}) \right) + c_{queue} \cdot tp_{in} \]
Intra-Operator Parallelism

Costs:

\[ c_q = \max \left( \sum_{i=1}^{k-1} \left( c_{op(i)} \cdot t_{p_{in}} \right) + c_p \cdot t_{p_{in}}, \left( c_{queue} + c_{op(k)} \right) \cdot t_{p_{in}}, \left( c_{queue} + c_m \right) \cdot t_{p_{in}} + \sum_{i=k+1}^{n} \left( c_{op(i)} \cdot t_{p_{in}} \right) \right) \]
Operator Costs

- Dependence on Algorithms/Implementation
- Different for used Stream Processing Engine, here: *PipeFabric*\(^1\)
- Example: **Projection** (reducing number of tuple attributes)

\[
c_{op}^{(proj)} = L_{latency} + mem_{latency} + \frac{c_{CPU}}{f_{clock}}
\]

1. Accessing tuple in L1 Cache
2. Writing Projection result in main memory
3. Clock frequency of CPU (Instructions per time unit)
4. CPU instructions (function calls, publish/subscribe tuple from preceding/following operators, ...)

\(^1\) https://github.com/dbis-ilm/pipefabric
Hardware Facts (Manycore CPU)

Intel Xeon Phi Knights Landing (KNL, 2016)

Some features:

- <72 cores à 4 threads, <1.5 GHz
- <384GB DDR4, 8-16GB MCDRAM (400+ GB/s)
Test Queries

- Tuples: *Integer, String, Double* attributes
- Query 1:
  - Selection (20% selectivity)
  - Projection (Integer, Double)
  - Aggregation (Double)
- Threads:

  ![Diagram](image)

Single-Threaded

Inter-Op. Parallelism

Intra-Op. Parallelism
Test Queries

• Tuples: *Integer*, *String*, *Double* attributes

• Query 2:
  ➢ Projection (Integer, Double)
  ➢ Grouping (Double)

• Threads:
Query Results

Q1, Q2

s = single thread

d = decoupled (Inter-Op.)

p = partitioned (Intra-Op.)
Example (Q1 d. with Inter-Op. Parallelism)

Inter-Op. Parallelism

Cost formula:

\[ c_{Q1} = \max \left( (c_{op(Sel)} \cdot tp_{in}), \left( c_{op(Proj)} \cdot tp_{in} + c_{op(Agg)} \cdot tp_{in} \right) \right) + c_{queue} \cdot tp_{in} \]

\[ c_{Q1} = c_{op(Proj)} \cdot tp_{in} + c_{op(Agg)} \cdot tp_{in} + c_{queue} \cdot tp_{in} \quad \text{// tp}_{in} = 20\% \text{ of 1mio tuples} \]

\[ c_{Q1} = (340 \text{ ns} + 552 \text{ ns} + 2650 \text{ ns}) \cdot 200.000 \]

\[ c_{Q1} = 708.400.000 \text{ ns} \]

\[ (= \text{Query execution time for processing 1mio tuples}) \]

\[ Latency_{Q1} = \frac{c_{Q1}}{1.000.000} \]

\[ = \sim 709\text{ ns / tuple} \]

- Accessing Tuple in L1 [\sim 3\text{ns}],
- Writing result in Main Memory [\sim 146\text{ns}],
- CPU time (function calls, subscribe/publish tuples,... \rightarrow \text{clock speed KNL!}) [\sim 191\text{ns}]
Conclusion

- Goal: Cost Model for Query Optimizer
  - Data Stream Processing
  - Modern Hardware (CPU, Memory)

- Cost Model focused first on multithreading aspect
  - Calibration approach for relevant HW factors
  - Parameters of operator implementations (SPE)

- Derived cost model, tested with Xeon Phi KNL
Summary

Cost Model:
➢ HW-Calibration approach, measuring relevant HW factors
➢ Knowledge about SPE: Implementation, Algorithms

(First) focus on multithreading aspect
➢ Inter-Operator & Intra-Operator parallelism

Testing:
➢ PipeFabric\(^1\) SPE
➢ Intel Xeon Phi KNL (7210) processor

\(^1\) https://github.com/dbis-ilm/pipefabric