Profiling Dataflow Systems on Multiple Abstraction Levels

Alexander Beischl, Timo Kersten, Maximilian Bandle
Jana Giceva, Thomas Neumann

Technische Universität München
### Compiling Dataflow Systems are Everywhere!

Dataflow systems in different areas

| Machine- and deep learning | Graph and stream-processing |
|----------------------------|-----------------------------|
| TensorFlow                | Flink                       |
| PyTorch                    | Ligra                       |

| Big-data processing        | Relational DBMS             |
|----------------------------|-----------------------------|
| Spark                      | Microsoft Hekaton           |
| Apache Naiad               | Microsoft HANA              |
|                            | UMBRA                       |
Profiling a Compiling Dataflow System

Trying to optimize the system

Query

df_sales.join(df_CPUs,
  col("df_sales.cpuID" ===
  col("df_CPUs.ID"), "inner")
Profiling a Compiling Dataflow System

Trying to optimize the system

```sql
df_sales.join(df_CPUs, 
  col("df_sales.cpuID" === 
  col("df_CPUs.ID"), "inner")
```

```plaintext
for tuple t in table T  
load int32 %40, i64 %13  
mov rax, [4 * rbx]
```
Profiling a Compiling Dataflow System

Trying to optimize the system

Query

df_sales.join(df_CPUs,
  col("df_sales.cpuID" ===
  col("df_CPUs.ID"), "inner")

Dataflow System

\[
\begin{align*}
\text{for tuple } t \text{ in table } T \\
\text{load int32 } & \%40, i64 \%13 \\
\text{mov rax, [4 * rbx]} \\
\text{cmp rax, 0} \\
\text{je } ...
\end{align*}
\]

Execution Time

5067ms
Profiling a Compiling Dataflow System

Trying to optimize the system

```
df_sales.join(df_CPUs, 
    col("df_sales.cpuID" === 
    col("df_CPUs.ID"), "inner")
```
Profiling a Compiling Dataflow System

Trying to optimize the system

Query
```
df_sales.join(df_CPUs, col("df_sales.cpuID" == col("df_CPUs.ID"), "inner")
```

Dataflow System
```
\[
\text{for tuple } t \text{ in table } T \\
\text{load int32 } %40, i64 %13 \\
\text{mov rax, [4 * rbx]}
\]
```

Execution Time
```
5067ms
```
Profiling a Compiling Dataflow System

Trying to optimize the system

Query
\[
\text{df}\_\text{sales}.\text{join(df}_\text{CPUs,}
\text{col("df}\_\text{sales.cpuID" ==}
\text{col("df}_\text{CPUs.ID"), "inner")}
\]

Dataflow System
\[
\begin{align*}
\text{for tuple } t \text{ in table } T \\
\text{load int32 } %40, \text{ i64 } %13 \\
\text{mov rax, [4 * rbx]}
\end{align*}
\]

Execution Time
5067ms

Query
\[
\text{df}\_\text{sales}.\text{join(df}_\text{CPUs,}
\text{col("df}\_\text{sales.cpuID" ==}
\text{col("df}_\text{CPUs.ID"), "inner")}
\]

Dataflow System
\[
\begin{align*}
\text{for tuple } t \text{ in table } T \\
\text{load int32 } %40, \text{ i64 } %13 \\
\text{mov rax, [4 * rbx]}
\end{align*}
\]

Execution

Perf report

Finding the slow path

Hashjoin (15%)

32%

3

Tailored Profiling

IPCl

567ms

567ms

Profiling Dataflow Systems on Multiple Abstraction Levels

beischl@in.tum.de
Profiling a Compiling Dataflow System

Trying to optimize the system

```
query1 = df_sales.join(df_CPUs, col("df_sales.cpuID" == col("df_CPUs.ID"), "inner")
```

```
query2 = df_sales.join(df_CPUs, col("df_sales.cpuID" == col("df_CPUs.ID"), "inner")
```

```
dataflow1 = 
    for tuple t in table T
    load int32 %40, i64 %13
    mov rax, [4 * rbx]
```

```
dataflow2 = 
    for tuple t in table T
    load int32 %40, i64 %13
    mov rax, [4 * rbx]
```

```
execution_time = 5067ms
```

```
perf_report = 
    loop_tuples:
    0%  %localTid = phi [%1, %loopBlocks %2, %contScan]
    0.1% %3 = getelemenptr int8 %state, i64 320
    0.1% %4 = getelemenptr int8 %3, i64 262144
    2.2% %5 = load int32 %4, %localTid
    2.3% %7 = crc32 i64 5961697176435608501, %5
    1.5% %8 = crc32 i64 2231409791114444147, %5
    1.2% %9 = rotr i64 %8, 32
    2.3% %10 = xor i64 %7, %9
    2.2% %11 = mul i64 %10, 268582165773638717
    1.2% %12 = shr %11, 16
    2.4% %13 = getelemenptr int8 %5, i64 %12
    32.1% %14 = load int32 %40, i64 %13
    0.2% %15 = isnnotnull ptr %12
    0.3% condbr %15 %loopHashChain %nextTuple
```
Profiling a Compiling Dataflow System

Trying to optimize the system

Query

```python
df_sales.join(df_CPUs, col("df_sales.cpuID" == col("df_CPUs.ID"), "inner")
```

Dataflow System

```
\[
\text{for tuple } t \text{ in table } T
\]
```

Execution

5067ms

Perf report

```
loopTuples:
0%  %localTid = phi [%1, %loopBlocks %2, %contScan]
0.1%  %3 = getelementptr int8 %state, i64 320
0.1%  %4 = getelementptr int8 %3, i64 262144
2.2%  %5 = load int32 %4, %localTid
2.3%  %7 = crc32 i64 5961697176435608501, %5
1.5%  %8 = crc32 i64 22314097911144444147, %5
1.2%  %9 = rotl i64 %8, 32
2.3%  %10 = xor i64 %7, %9
2.2%  %11 = mul i64 %10, 2685821657736338717
1.2%  %12 = shr %11, 16
2.4%  %13 = getelementptr int8 %5, i64 %12
32.1%  %14 = load int32 %40, i64 %13
0.2%  %15 = isnotnull ptr %12
0.3%  condbr %15 %loopHashChain %nextTuple
```

3
Why do we have this problem?

Identifying the gap

Query

Dataflow System

Dataflow Graph
Imperative Prog.
Machine IR
x86 Assembly
Why do we have this problem?
Identifying the gap

Query
↓
Dataflow System
↓
Dataflow Graph
Imperative Prog.
Machine IR
x86 Assembly

`df_sales.join(df_CPUs, col("df_sales.cpuID") ...)`
Why do we have this problem?

Identifying the gap

```
for tuple t in table T
    if t[1] > 5
```

```
df_sales.join(df_CPUs,
    col("df_sales.cpuID")
)
```

```
load int32 %40, i64 %13
isnotnull ptr %12
```
Why do we have this problem?

Identifying the gap

Query → Dataflow System

- Dataflow Graph
- Imperative Program
- Machine IR
- x86 Assembly

for tuple t in table T
if t[1] > 5
...

for tuple t in table T
if t[1] > 5
...
Why do we have this problem?

Identifying the gap

Query → Dataflow System

- Dataflow Graph
- Imperative Prog.
- Machine IR
- x86 Assembly

Dataflow System

```
for tuple t in table T
  if t[1] > 5
...
```

```
load int32 %40, i64 %13
isnotnull ptr %12
...
```
Why do we have this problem?

Identifying the gap

Query

Dataflow System

- Dataflow Graph
- Imperative Program
- Machine IR
- x86 Assembly

Example code snippet in x86 Assembly:
```
mov rax, [4 * rbx]
cmp rax, 0
je @ ...
```
Why do we have this problem?
Identifying the gap
Why do we have this problem?

Identifying the gap
Why do we have this problem?

Identifying the gap

Dataflow System

- Dataflow Graph
- Imperative Prog.
- Machine IR
- x86 Assembly

Query

Execution

Profiling

Result

Machine IR Results

Profiling Samples

CodeGen Dev

Machine IR

3UR*$ULQJ6DPSOHV

CodeGen Dev

W$LWHV

6DPSOH

Profiling Dataflow Systems on Multiple Abstraction Levels
Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap

Why do we have this problem?
Identifying the gap
Why do we have this problem?

Identifying the gap

Query

Dataflow System

Dataflow Graph

Imperative Prog.

Machine IR

x86 Assembly

Machine IR Results

Profiling Samples

Execution

Profiling

Result

Domain Expert

Optimizer Dev

CodeGen Dev

for tuple t in table T
if t[1] > 5
Why do we have this problem?

Identifying the gap

Query

Dataflow System

Dataflow Graph
Imperative Prog.
Machine IR
x86 Assembly

Connection lost

Machine IR Results
Profiling Samples

Execution
Result

Profiling

for tuple t in table T
if t[1] > 5
...
Tailored Profiling

Closing the gap

- Track connection between components of all abstraction levels down to generated code
- Map profiling samples back to higher abstraction levels

- Ingredients
  - Tagging Dictionary & Register Tagging
Tailored Profiling

Tagging Dictionary

① *Connection tracking* of abstraction components for each lowering step (top-down)
Tailored Profiling

Tagging Dictionary

② Store mapping in the Tagging Dictionary
Tailored Profiling

Tagging Dictionary

3. Map profiling results to each abstraction level’s components (bottom-up)
Tailored Profiling

Tagging Dictionary

④ Aggregate the data for profiling results
Tailored Profiling

Tagging Dictionary and Register Tagging

**Dataflow Graph**

```
...  
Filter_{price>500}  
Scan sales
```

**Generated Query Code**

```
for each tuple t in sales
    ...
    if t.price > 500
        ...
```
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

```
... 
Filter_{\text{price} > 500} 
Scan \text{sales}
```

Generated Query Code

```
for each tuple \( t \) in \text{sales}
...
if \( t.\text{price} > 500 \)
...
```

\{for each tuple \( t \) in \text{sales} \( s \) -> \text{Scan} \text{sales}\}

Tagging Dictionary
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

```
...  
Filter_{price>500}  
Scan sales  
```

Generated Query Code

```
for each tuple t in sales  
...
if t.price > 500  
...  
```

{if t.price > 500 -> Filter}

Tagging Dictionary
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

Generated Query Code

```
for each tuple t in sales
    call malloc(...)
    if t.price > 500
        call malloc(...)
```
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

... 
Filter\_price>500 
Scan sales

Generated Query Code

\begin{verbatim}
for each tuple t in sales
    call malloc(...) 
    if t.price > 500
        call malloc(...)
\end{verbatim}

Profiling Sample

Source Line

malloc(...) 

Tagging Dictionary

Scan, Filter
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

Generated Query Code

for each tuple t in sales
    call malloc(...) 
    if t.price > 500
        call malloc(...)

Profiling Sample

Source Line

malloc(...) 

Call-Stack Sample

Recorded Call-Stack

malloc(...) 
Scan: call malloc() 
...
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

- ... (omitted)
- Filter \( \text{price} > 500 \)
- Scan sales

Generated Query Code

- for each tuple \( t \) in sales
  - call malloc(…)
  - if \( t.\text{price} > 500 \)
  - call malloc(…)

Machine Register
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

```
...  
Filter \text{price} \geq 500  
Scan sales
```

Generated Query Code

```
for each tuple t in sales
    setTag(Scan)
    call malloc(...)  
    unsetTag()
    if \ t.\text{price} \ > \ 500
        setTag(Filter)
        call malloc(...)  
        unsetTag()
```

Machine Register

```
```
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

...  
Filter \text{price} > 500  
Scan \text{sales}

Generated Query Code

\begin{verbatim}
for each tuple \textit{t} in sales
  setTag(Scan)
  call malloc(...)  
  unsetTag()
  if \textit{t}.price > 500
    setTag(Filter)
    call malloc(...)  
    unsetTag()
\end{verbatim}

Machine Register

Scan
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

...  
Filter$_{price\geq 500}$  
Scan sales

Generated Query Code

```python
for each tuple t in sales
    setTag(Scan)
    call malloc(...)
    unsetTag()
    if t.price > 500
        setTag(Filter)
        call malloc(...)
        unsetTag()
```

Machine Register

Scan

Profiling Sample

| Source Line | Register Value |
|-------------|----------------|
| malloc(...) | Scan           |
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

```
... |
  |
  |
```

Generated Query Code

```
for each tuple t in sales
  setTag(Scan)
  call malloc(...)
  if t.price > 500
    setTag(Filter)
    call malloc(...)
  unsetTag()
```

Machine Register
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

```
... 
Filter_{price>500} 
Scan sales
```

Generated Query Code

```
for each tuple t in sales 
setTag(Scan)
call malloc(...) 
unsetTag() 
if t.price > 500 
setTag(Filter) 
call malloc(...) 
unsetTag()
```

Machine Register

```
Filter
```
Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

```
...                  
Filter_{\text{price}>500}  
Scan sales
```

Generated Query Code

```
for each tuple t in sales
  setTag(Scan)
  call malloc(...)  
  unsetTag()
if t.price > 500
  setTag(Filter)
  call malloc(...)  
  unsetTag()
```

Machine Register

```
Filter
```

Profiling Sample

```
| Source Line | Register Value |
|-------------|----------------|
| malloc(...)  | Filter         |
```
Insights with Tailored Profiling
Insights with Tailored Profiling
We begin the use cases with the domain expert and proceed with the optimizer developer and the operator developer. With a scale factor of 1 (dataset size 1 GB) for the use-cases, with use cases for di

Experimental Setup. For our experiments, we use the TPC-H benchmark [49] to measure performance and accuracy. However, the samples' timestamps provided by the Linux kernel during sample collection in PEBS are not ideal for profiling, as they may be affected by system events and other interferences.

To improve the accuracy of our timestamps, we use the processor's Timestamp Counter (TSC) [36] for recording sample times. The TSC has cycle-grained resolution and is available on most modern processors. By using the TSC, we ensure that our measurements are accurate and can be used for profiling.

In this section, we evaluate the advantages of Tailored Profiling, a tool that allows developers and operators to gain insights into the performance of their systems. We start with a use case involving the projection of a non-dominant plan, which is often used in database systems. The projection is slow because it involves a large hash table that is not optimized for speed.

To compare the performance of different plans, we use a cost function that takes into account the expected runtime and accuracy of each plan. The cost function helps us make informed decisions about which plan to use in different situations.

Finally, we show how Tailored Profiling can be used to optimize performance by providing insights into the behavior of different operators and their execution times. This information can be used to improve the performance of database systems and other data processing systems.
Insights with Tailored Profiling

Time per operator

Context-aware profiling over time

Memory access patterns
Impact of Tailored Profiling

Where can you apply it?

- Preserve connection information to close gap
- Profiling results on high abstraction levels

```plaintext
for tuple t in table T
  if t[1] > 5
    ...
```

Machine IR

CodeGen Dev  
Optimizer Dev  
CodeGen Dev

Profiling Samples

Execution

Profiling

Connection lost

32%
Impact of Tailored Profiling

Where can you apply it?

- Preserve connection information to close gap
- Profiling results on high abstraction levels
- Lightweight, high accuracy
- Easy to integrate
- Applicable to many systems
Impact of Tailored Profiling

Where can you apply it?

- Preserve connection information to close gap
- Profiling results on high abstraction levels
- Lightweight, high accuracy
- Easy to integrate
- Applicable to many systems

- *Already supported:* profiling code on CPUs (multi-socket and multicore)
- *Future work:* heterogenous compute resources, distributed systems
Thank you for watching!