Profiling Dataflow Systems on Multiple Abstraction Levels

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

Technische Universität München

beischl@in.tum.de

Compiling Dataflow Systems are Everywhere!

Dataflow systems in different areas

Machine- and deep learning

TensorFlow
PyTorch

Graph and stream-processing

Flink
Ligra

Big-data processing

Apache Spark
Microsoft Naiad

Relational DBMS

Microsoft Hekaton
SAP 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")
```

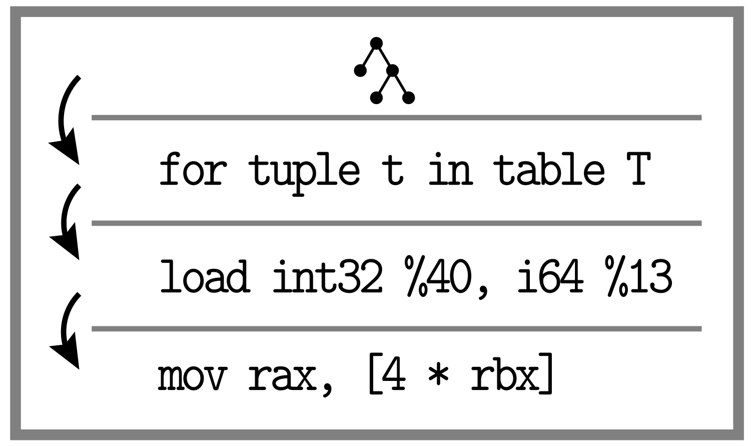
Profiling a Compiling Dataflow System

Trying to optimize the system

Query

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

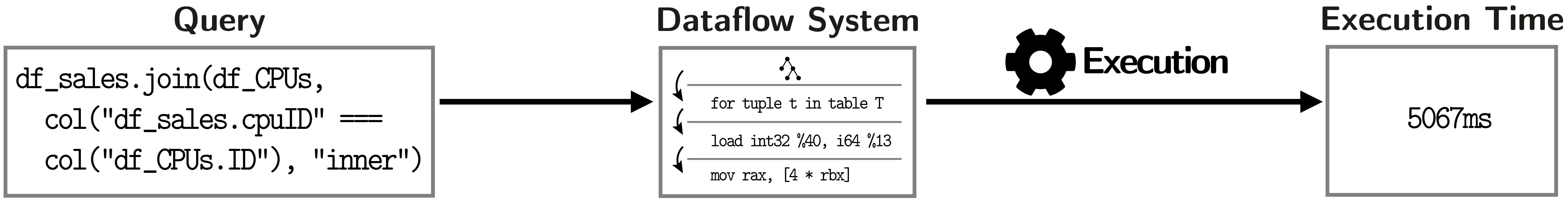
Dataflow System



beischl@in.tum.de

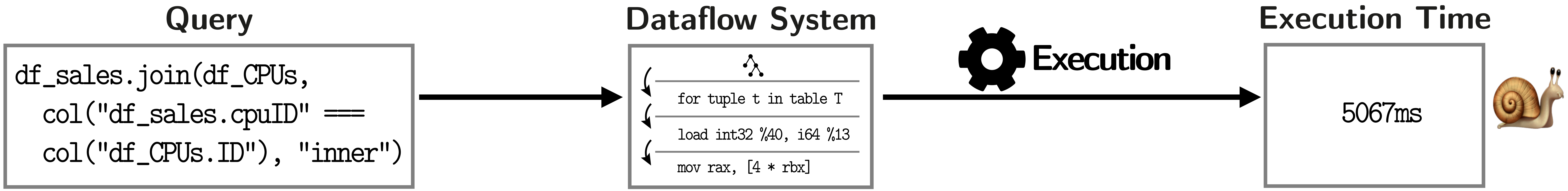
Profiling a Compiling Dataflow System

Trying to optimize the system



Profiling a Compiling Dataflow System

Trying to optimize the system



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

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

Execution

Execution Time

5067ms

Query

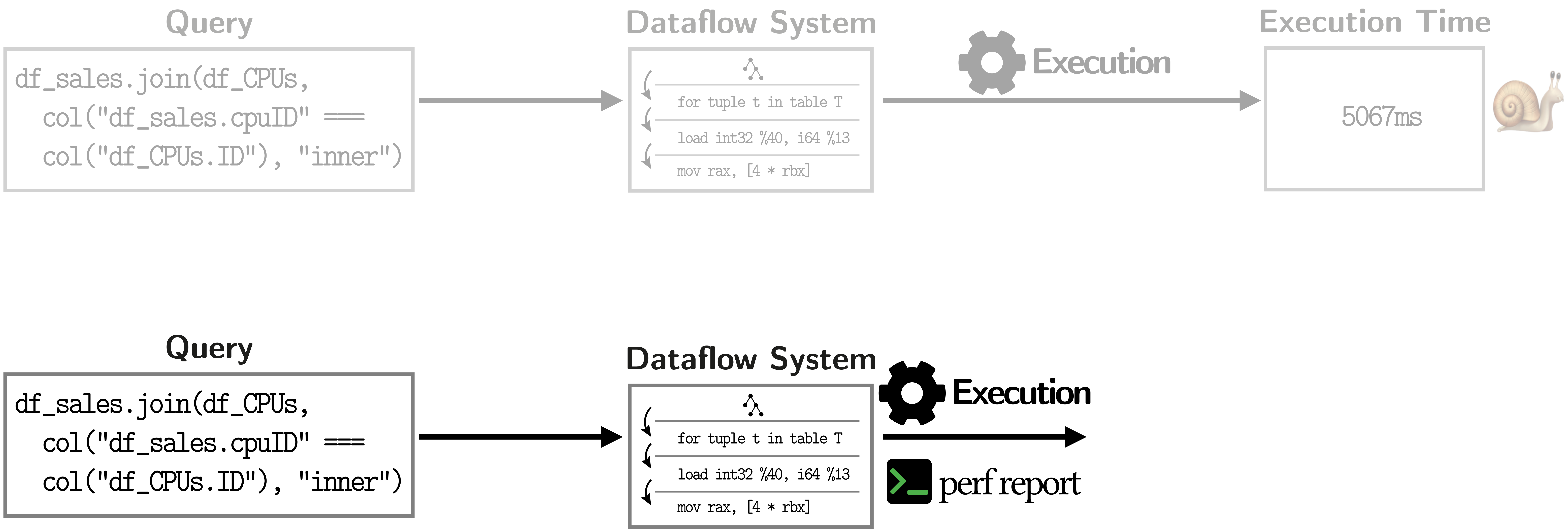
```
df_sales.join(df_CPUs,
  col("df_sales.cpuID" ===
  col("df_CPUs.ID"), "inner")
```

Dataflow System

```
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



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

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

Execution

Execution Time

5067ms

Query

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

Dataflow System

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

Execution

perf report

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 2231409791114444147, %5
 1.2%   %9 = rotr 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
```

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

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

Execution

Execution Time

5067ms

Query

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

Dataflow System

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

Execution

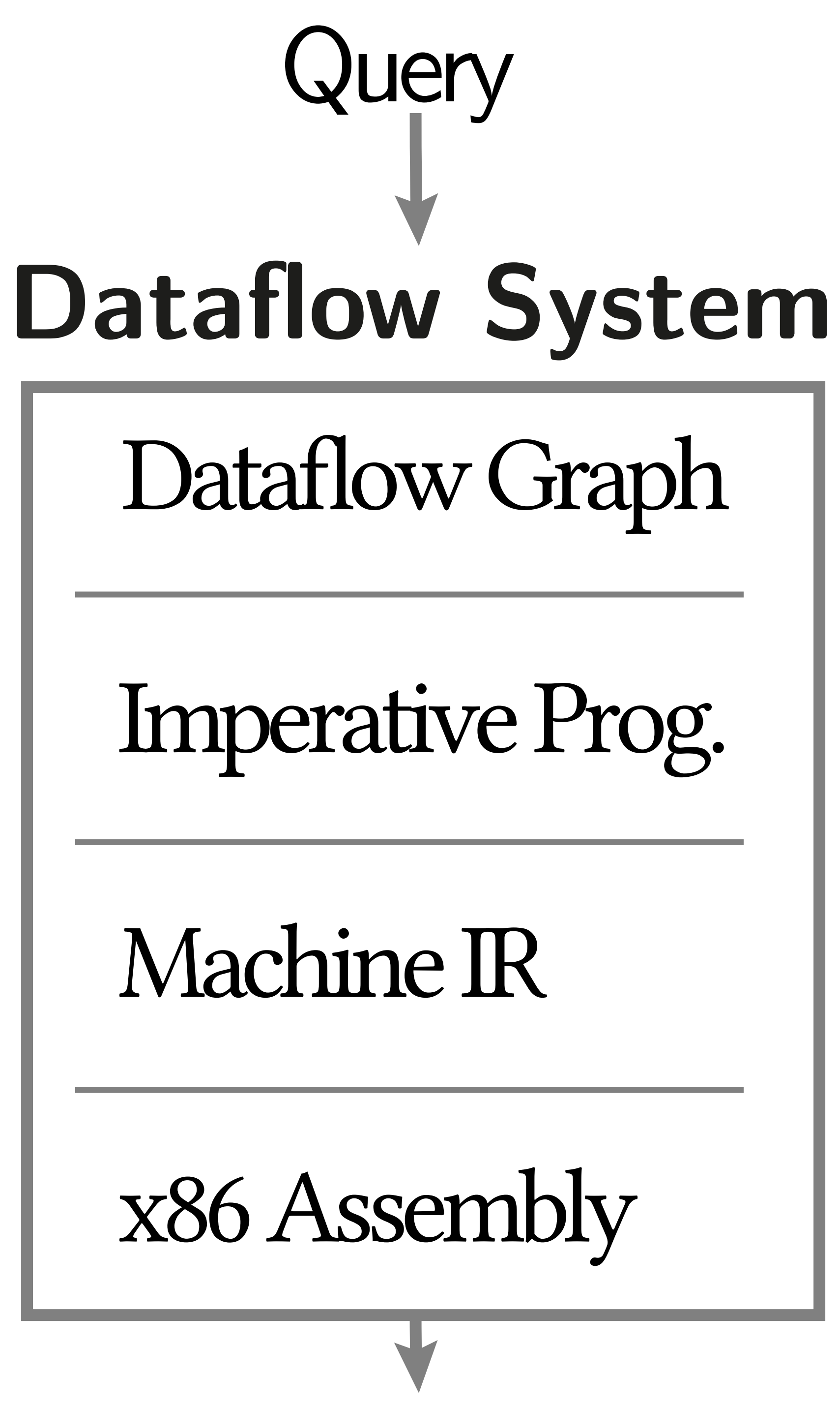
perf report

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 2231409791114444147, %5
 1.2%   %9 = rotr 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
```

Why do we have this problem?

Identifying the gap



Why do we have this problem?

Identifying the gap

Query

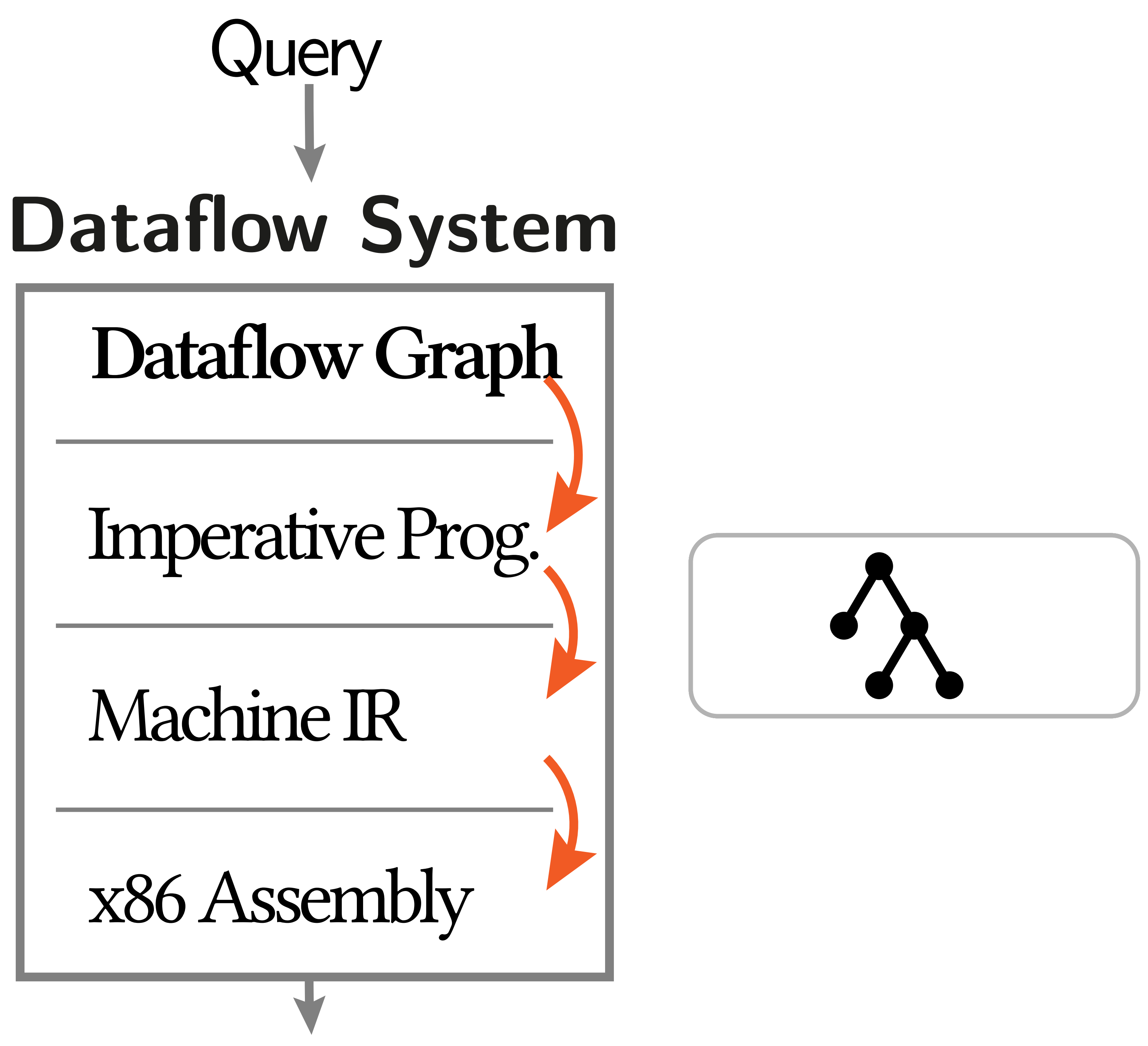
Dataflow System

Dataflow Graph
Imperative Prog.
Machine IR
x86 Assembly

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

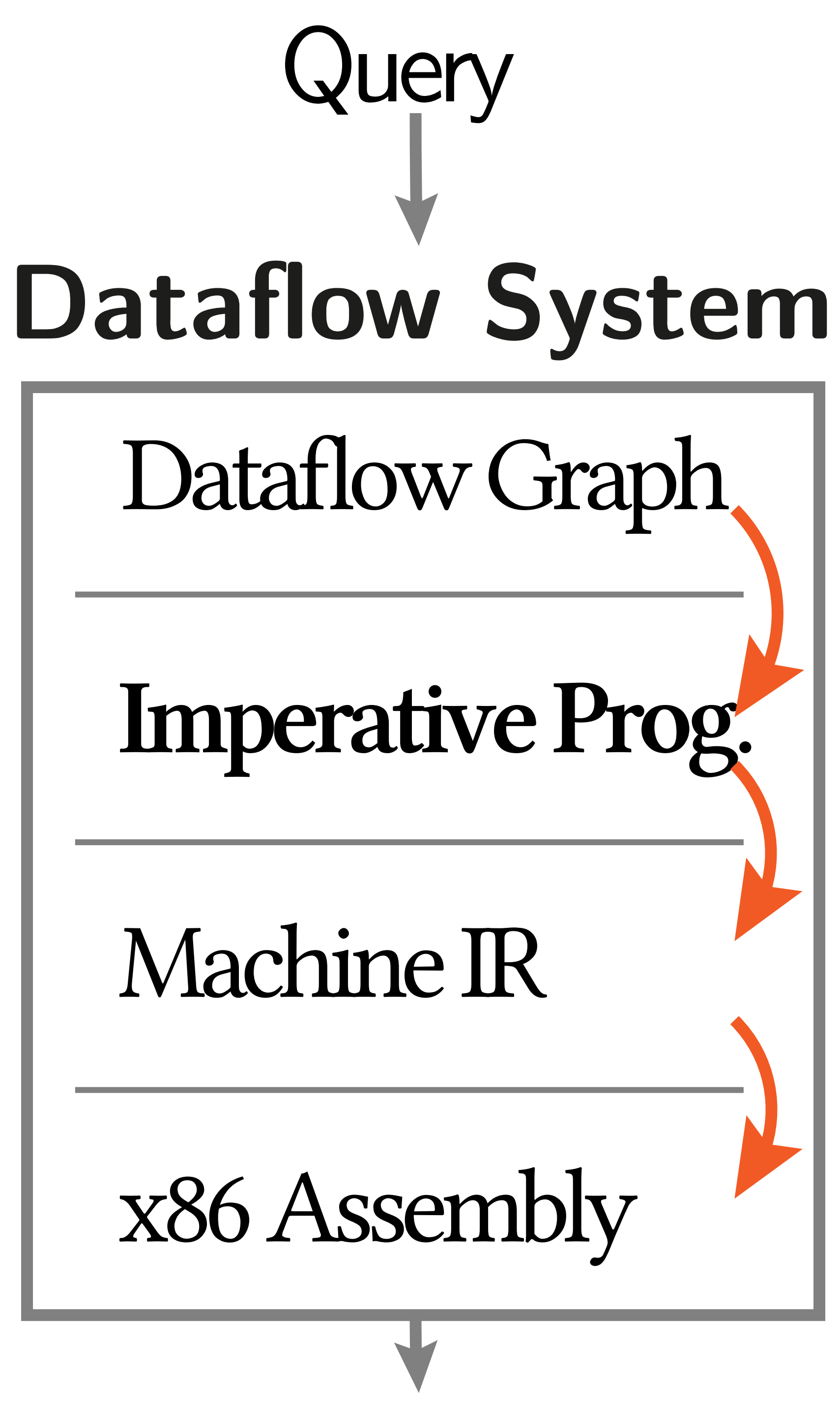
Why do we have this problem?

Identifying the gap



Why do we have this problem?

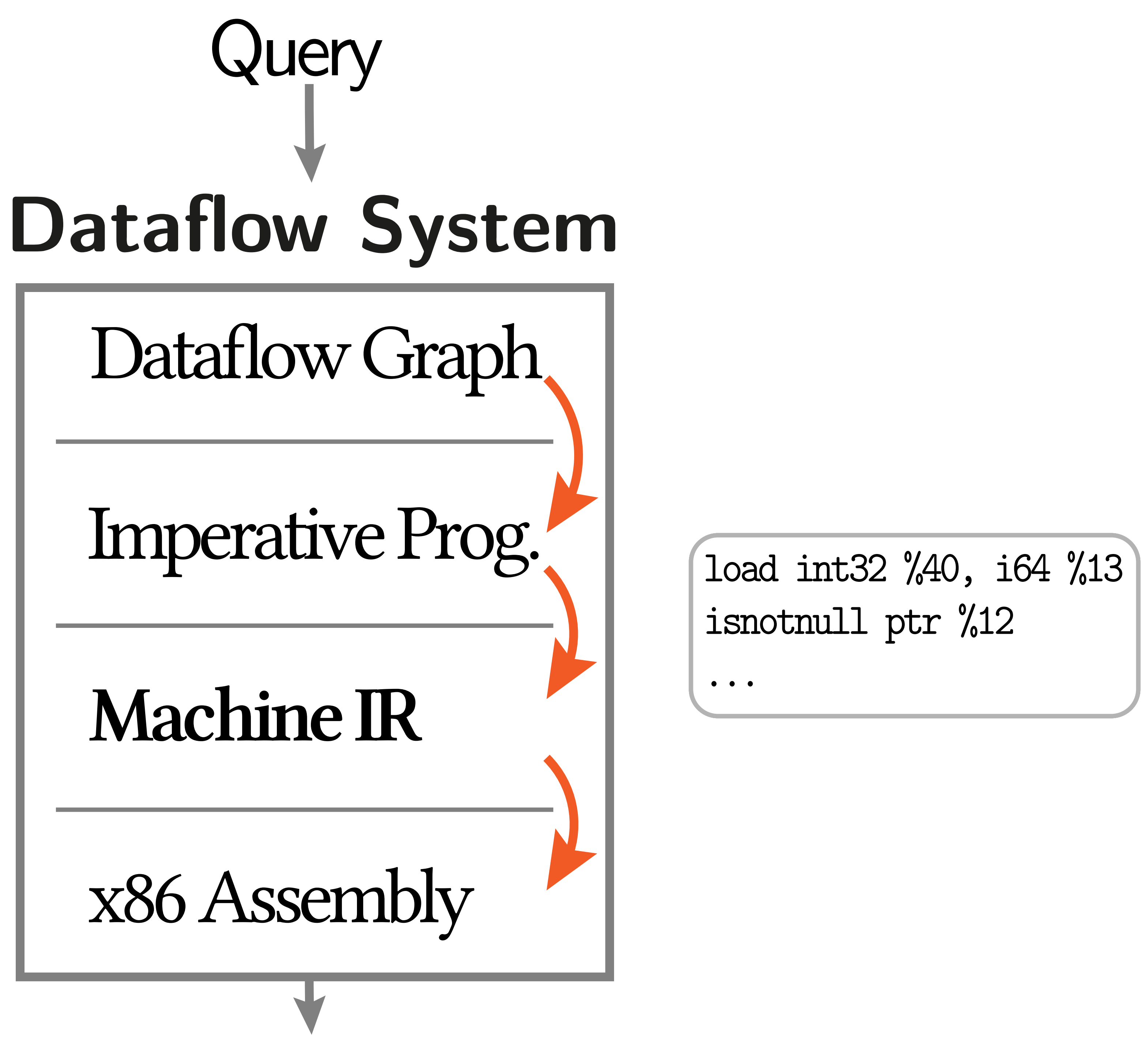
Identifying the gap




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

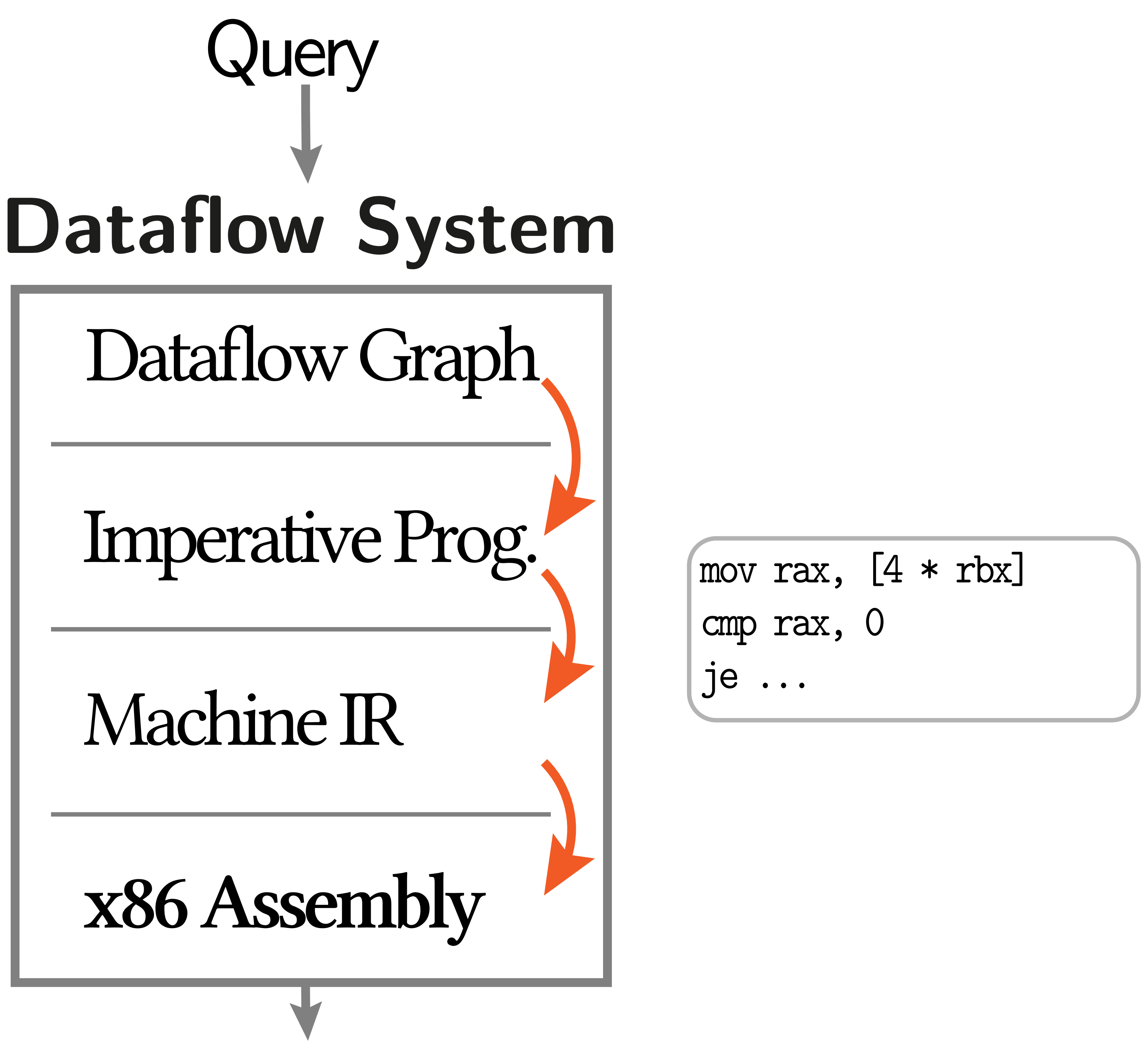
Why do we have this problem?

Identifying the gap



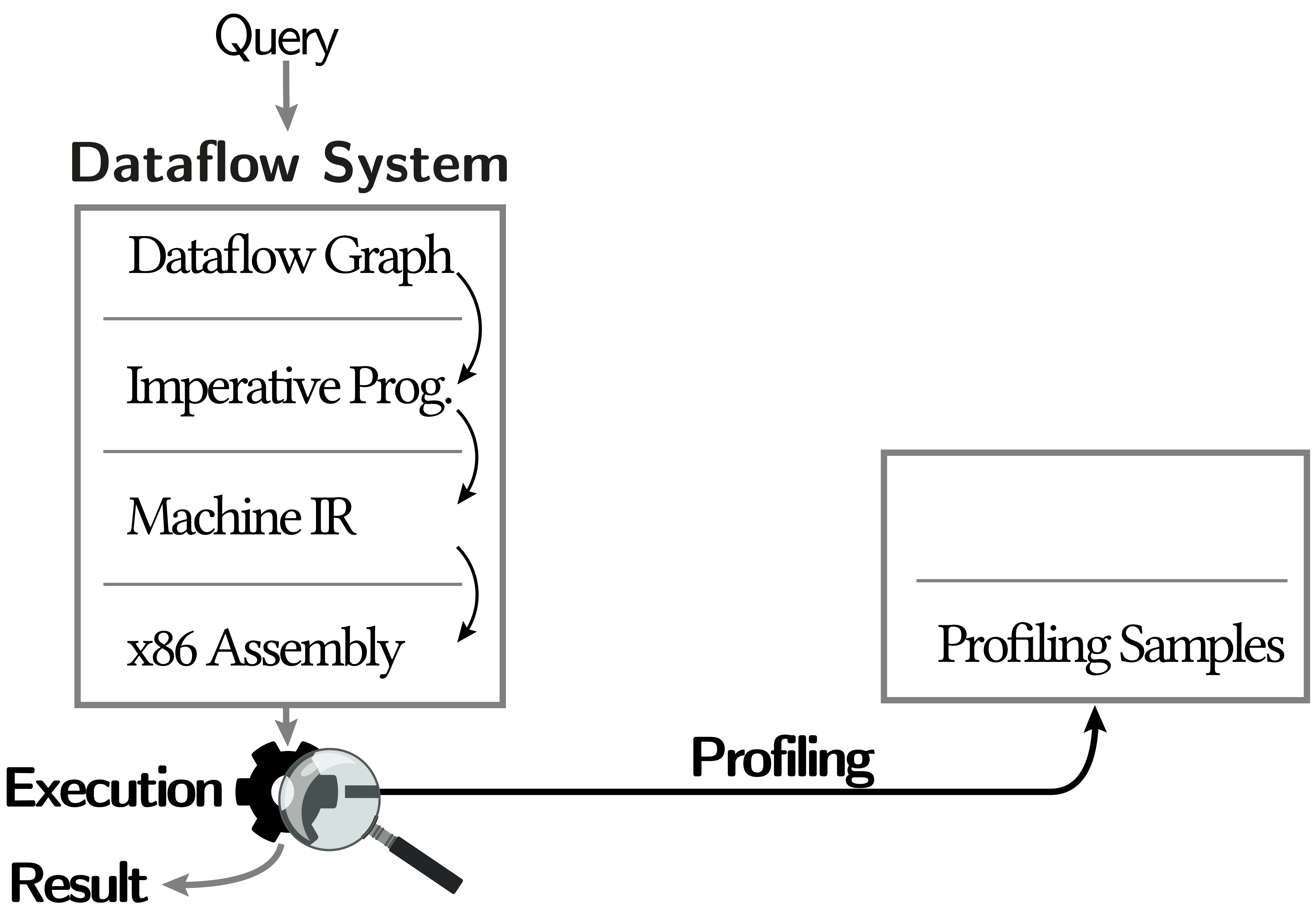
Why do we have this problem?

Identifying the gap



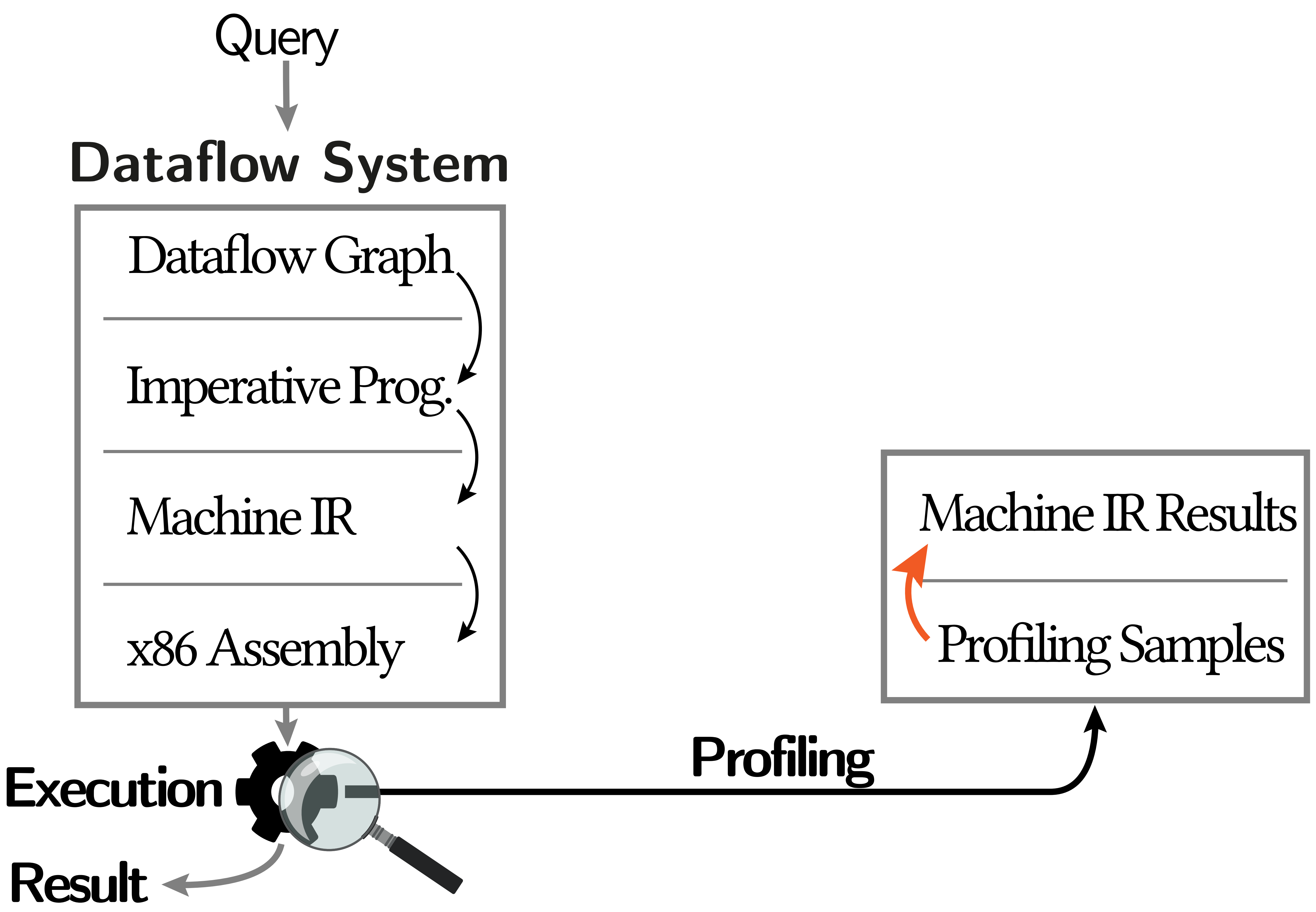
Why do we have this problem?

Identifying the gap



Why do we have this problem?

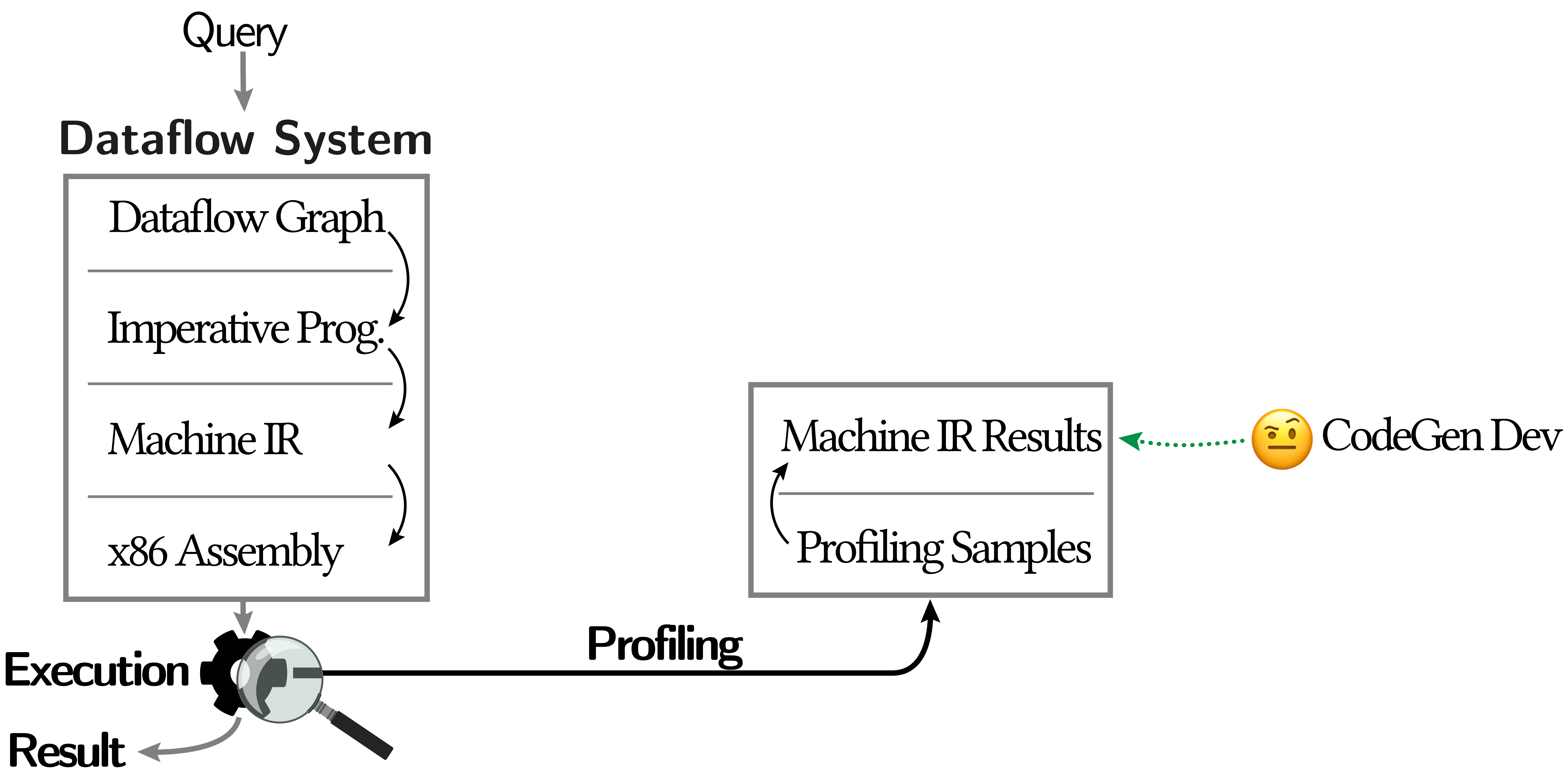
Identifying the gap



beischl@in.tum.de

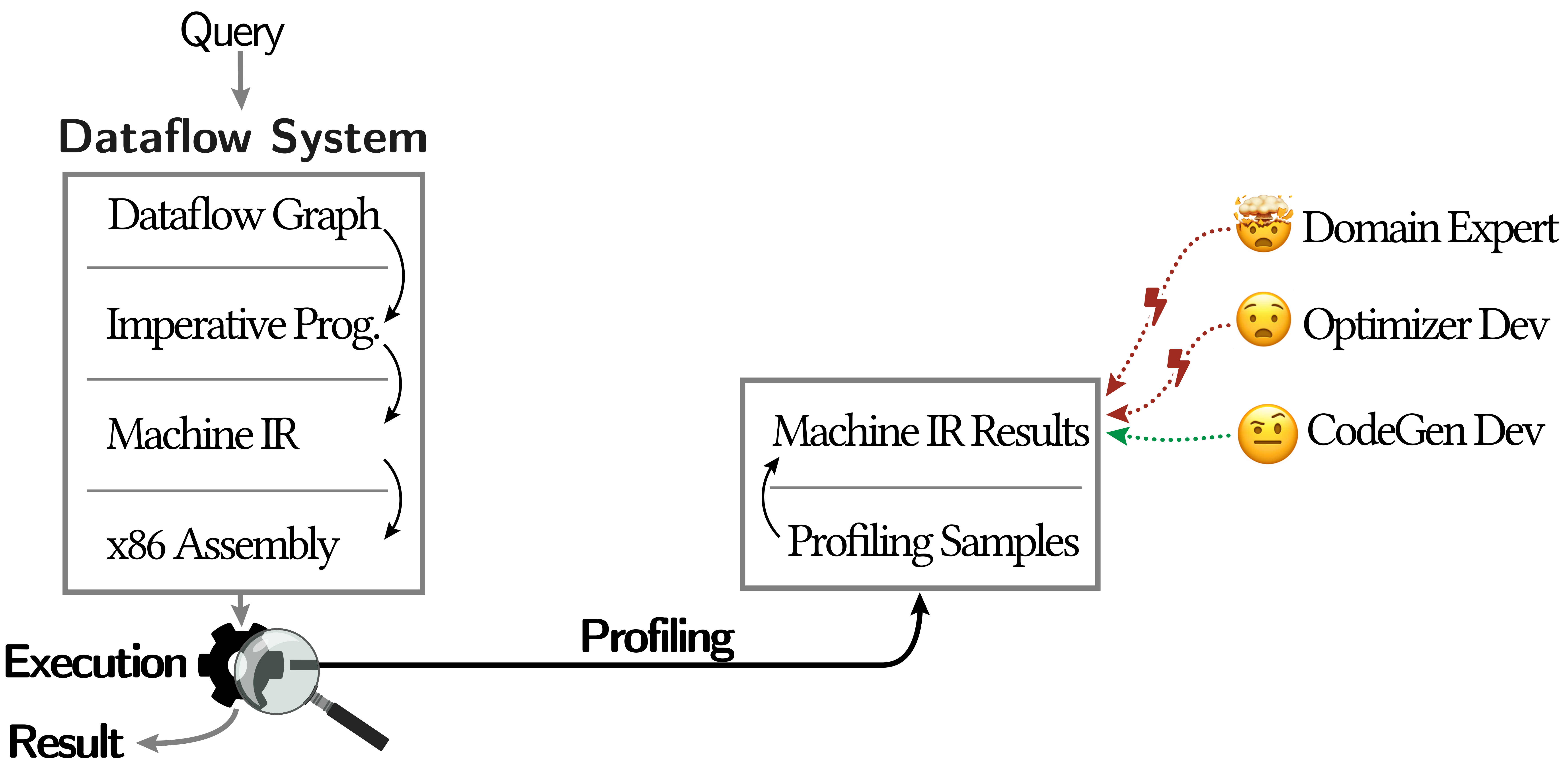
Why do we have this problem?

Identifying the gap



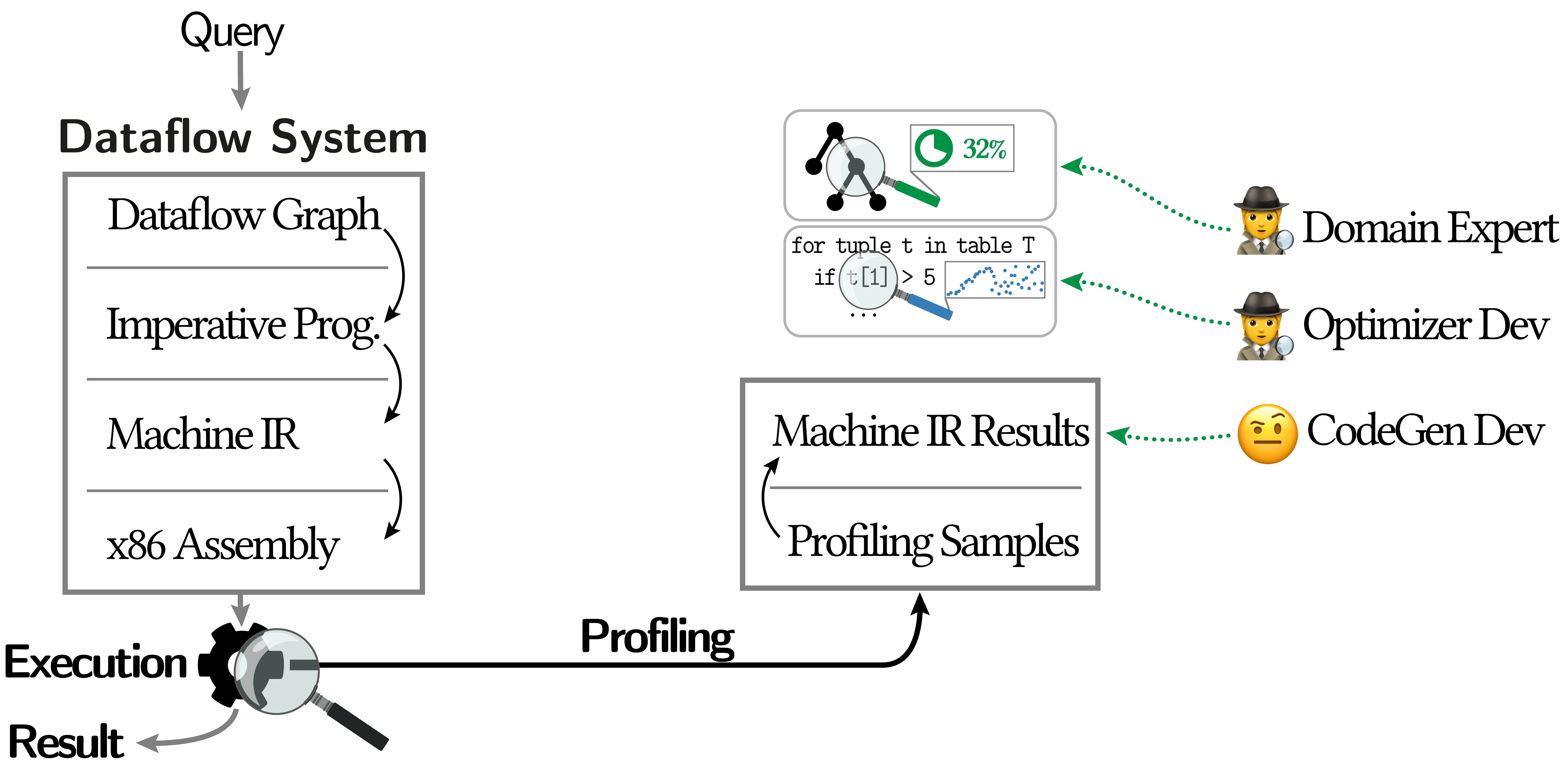
Why do we have this problem?

Identifying the gap



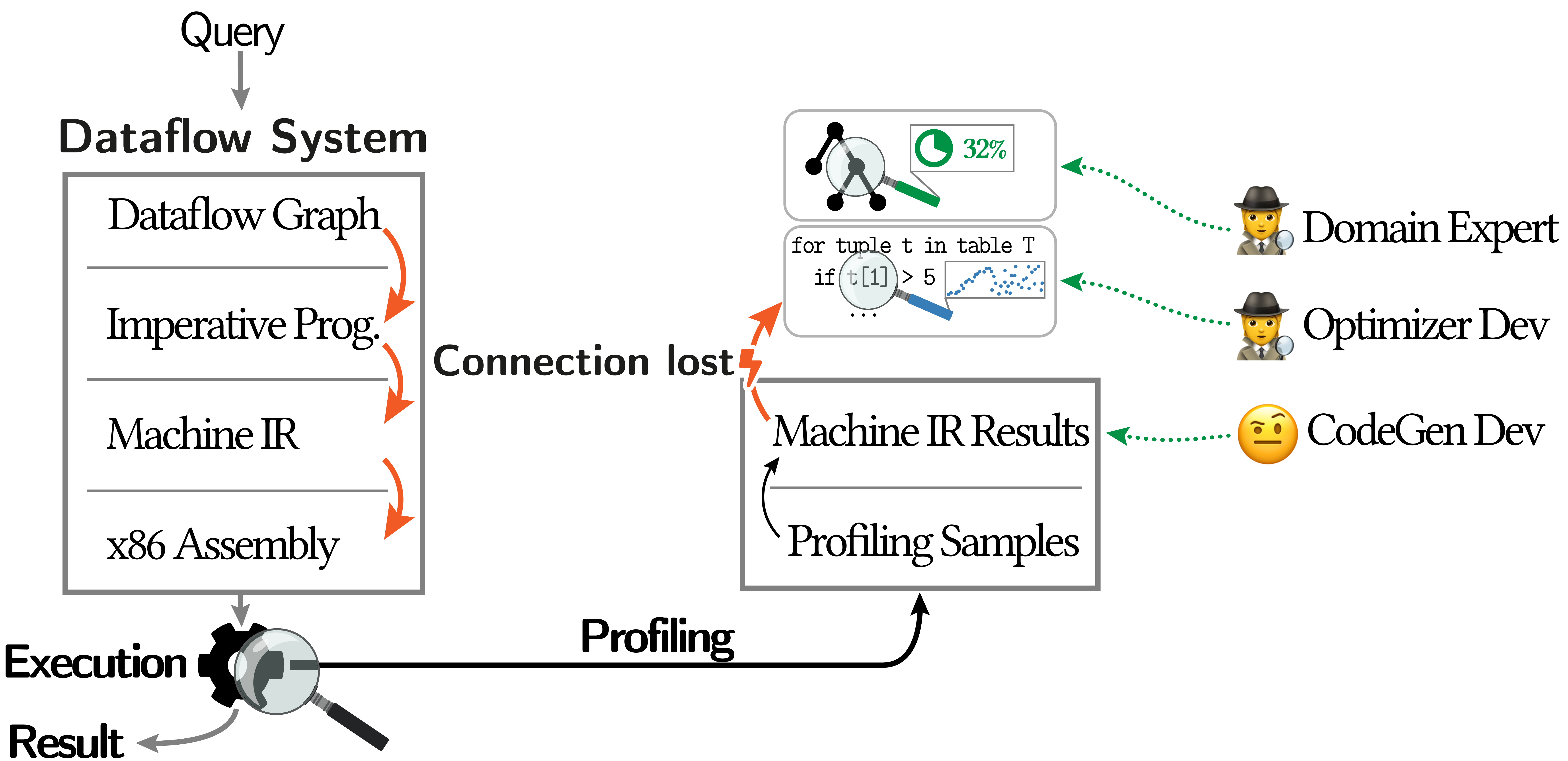
Why do we have this problem?

Identifying the gap



Why do we have this problem?

Identifying the gap

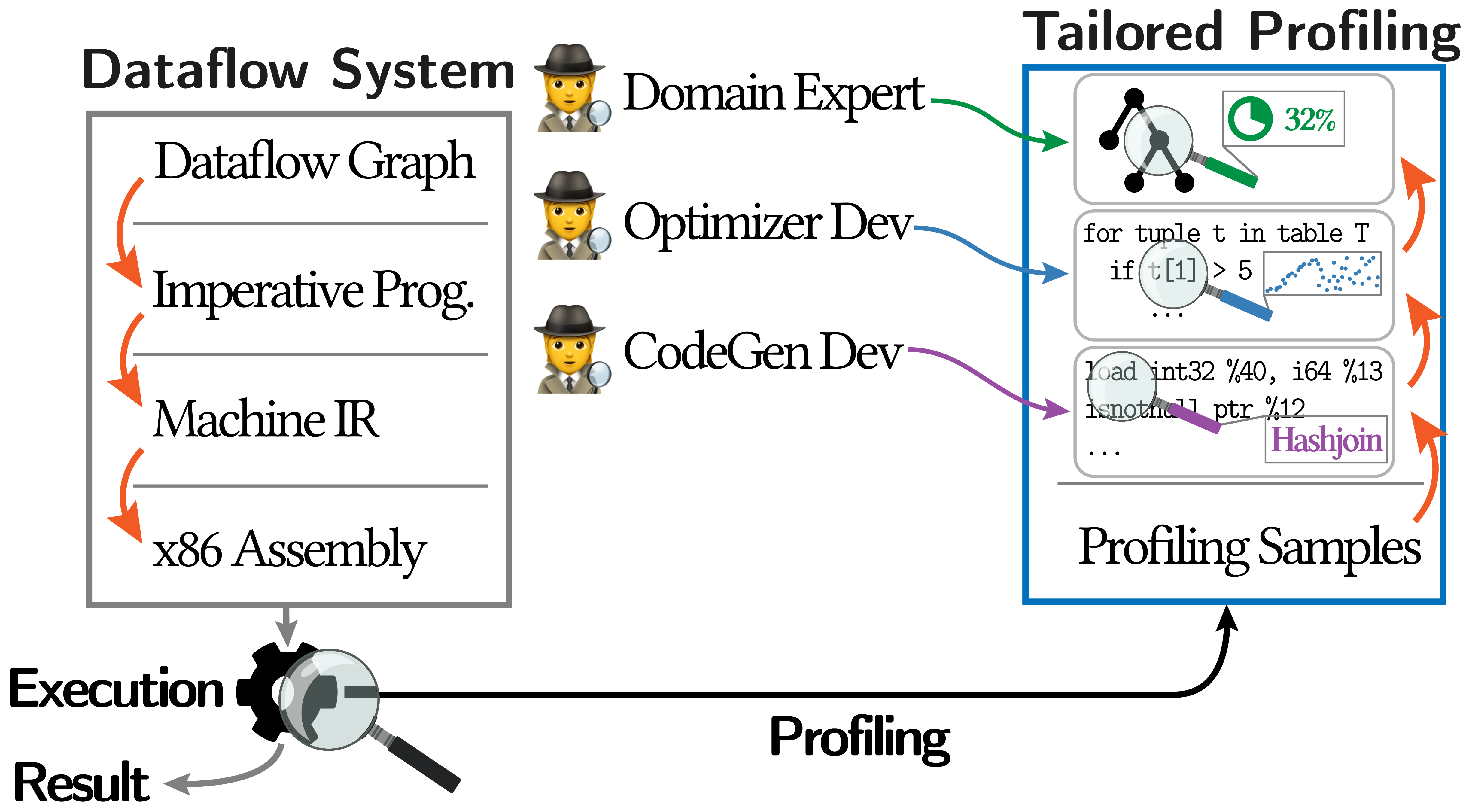


Tailored Profiling

Closing the gap

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

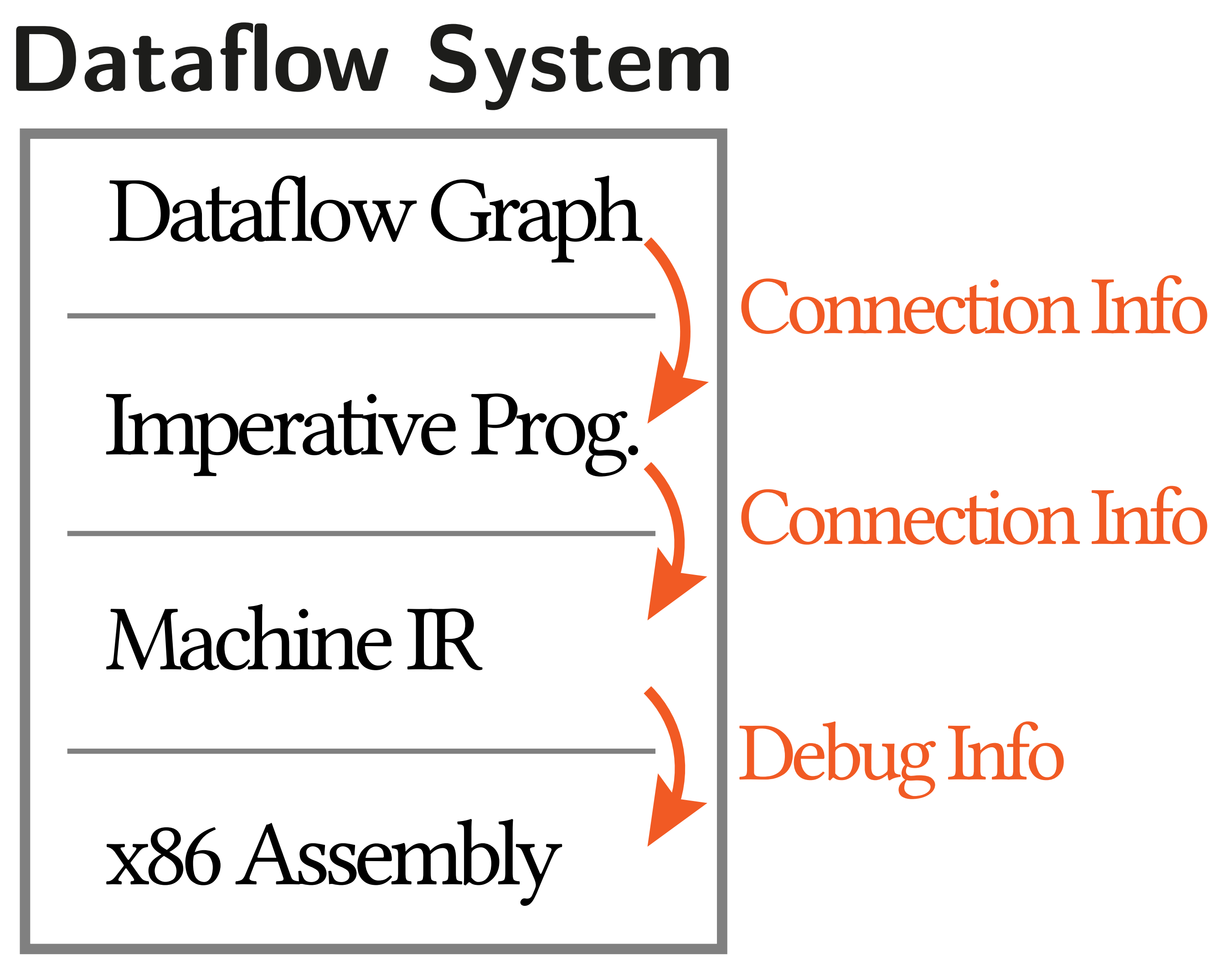
- Ingredients
  - *Tagging Dictionary & Register Tagging*



Tailored Profiling

Tagging Dictionary

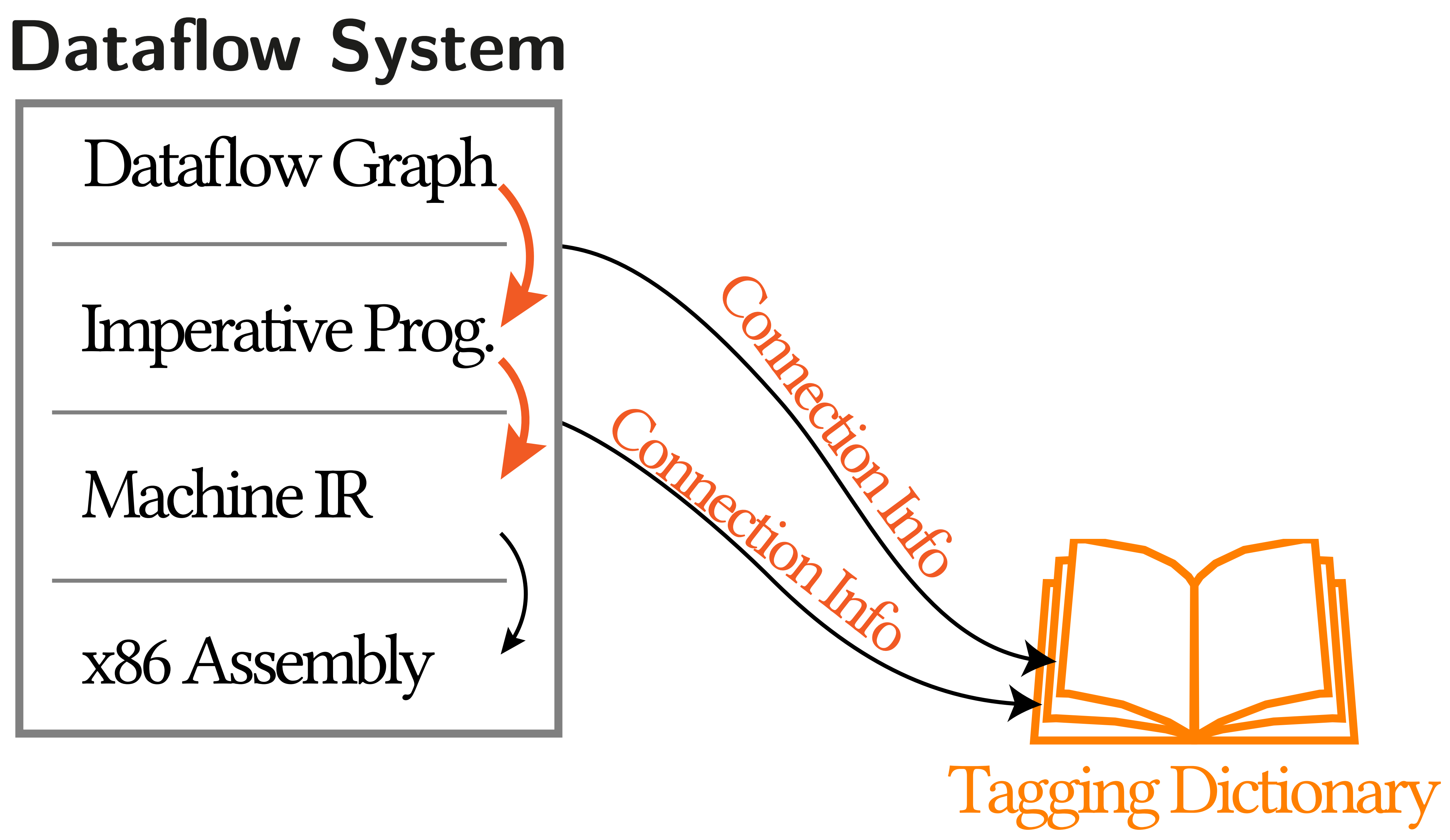
① *Connection tracking* of abstraction components for each lowering step (top-down)



Tailored Profiling

Tagging Dictionary

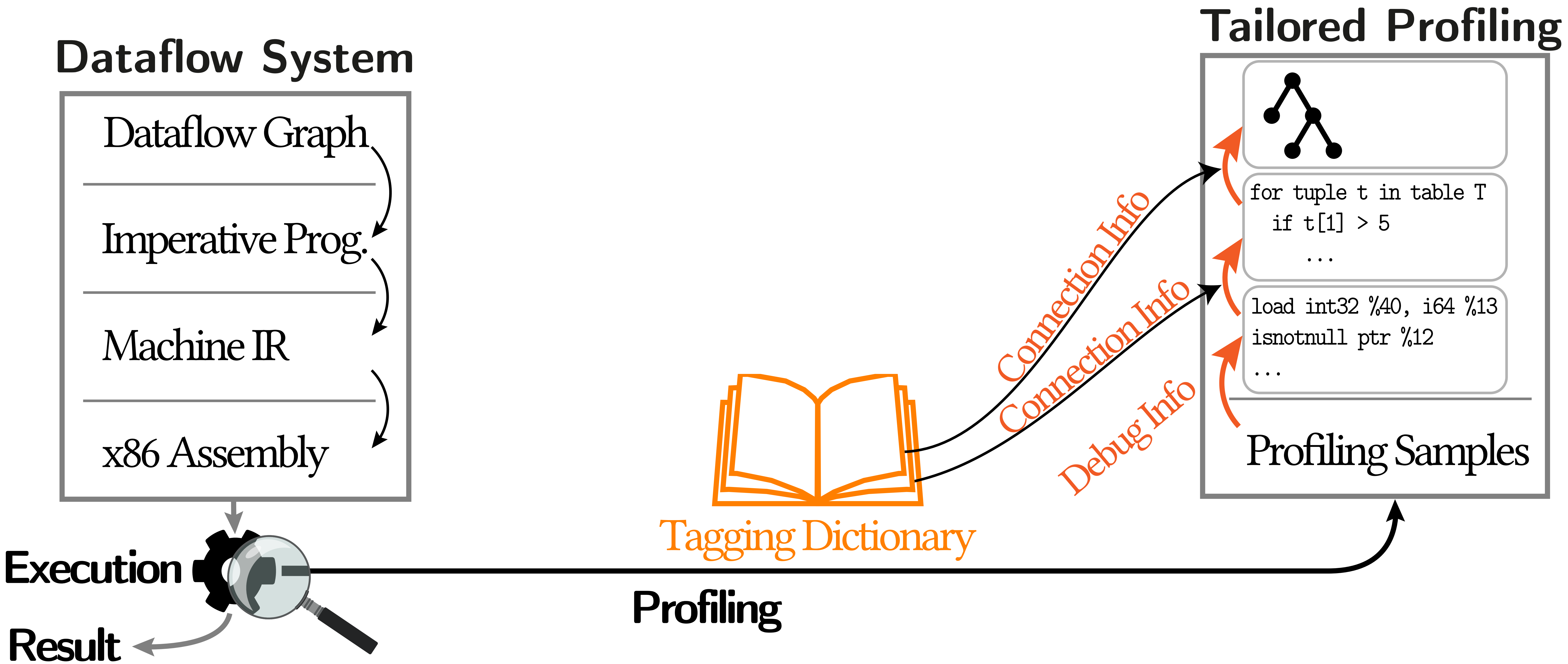
② Store mapping in the *Tagging Dictionary*



Tailored Profiling

Tagging Dictionary

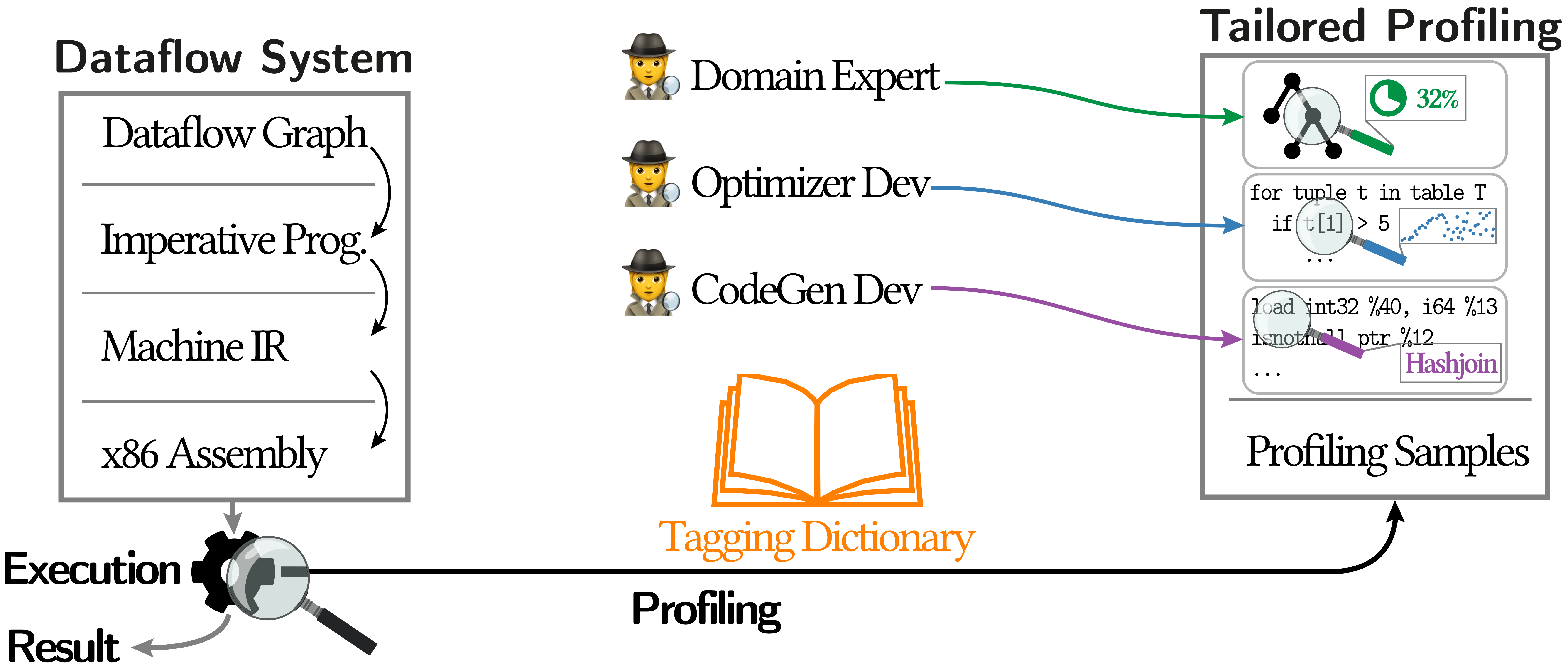
③ *Map* profiling results to each abstraction level's components (bottom-up)



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

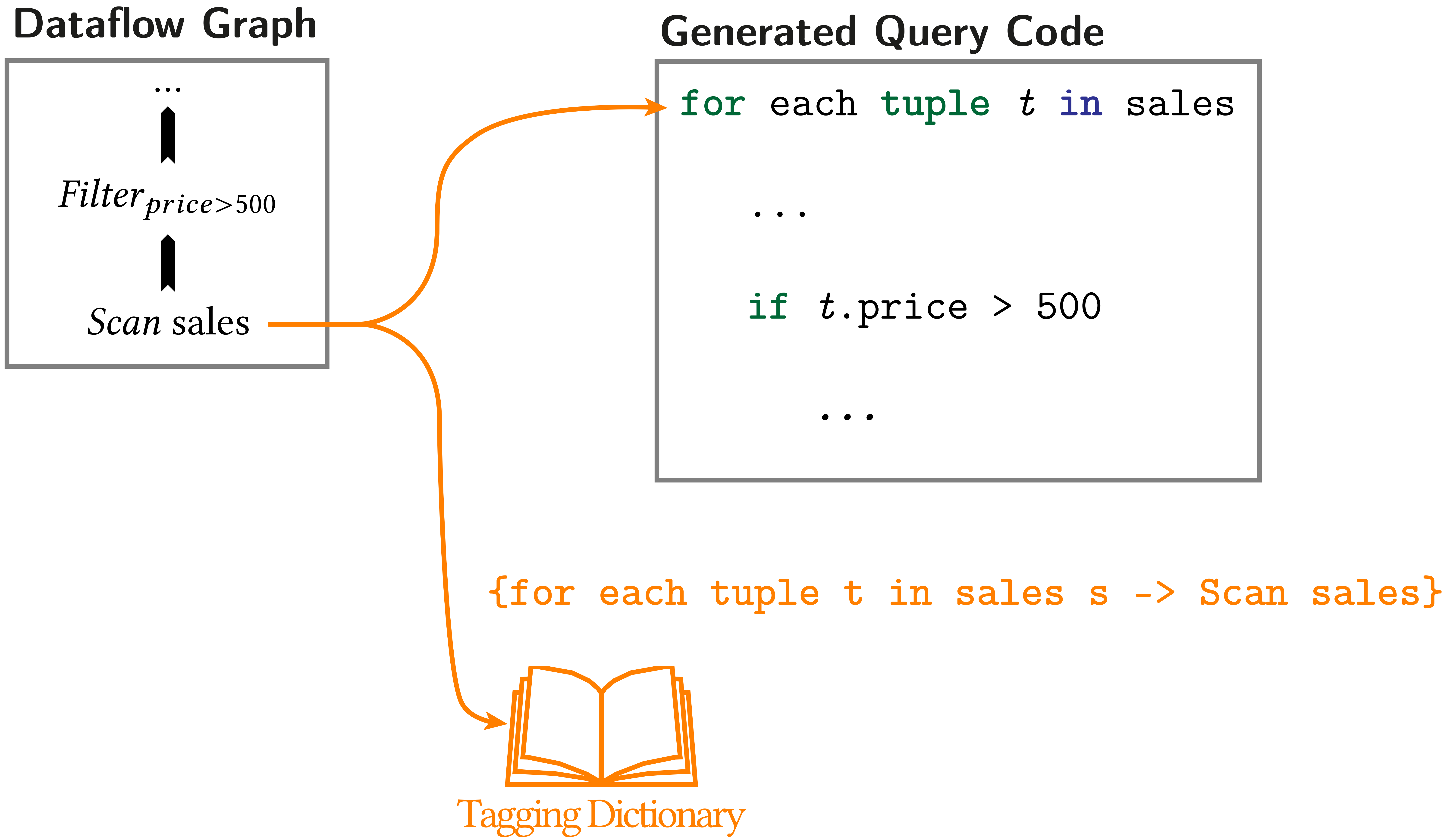
Generated Query Code

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

Tagging Dictionary

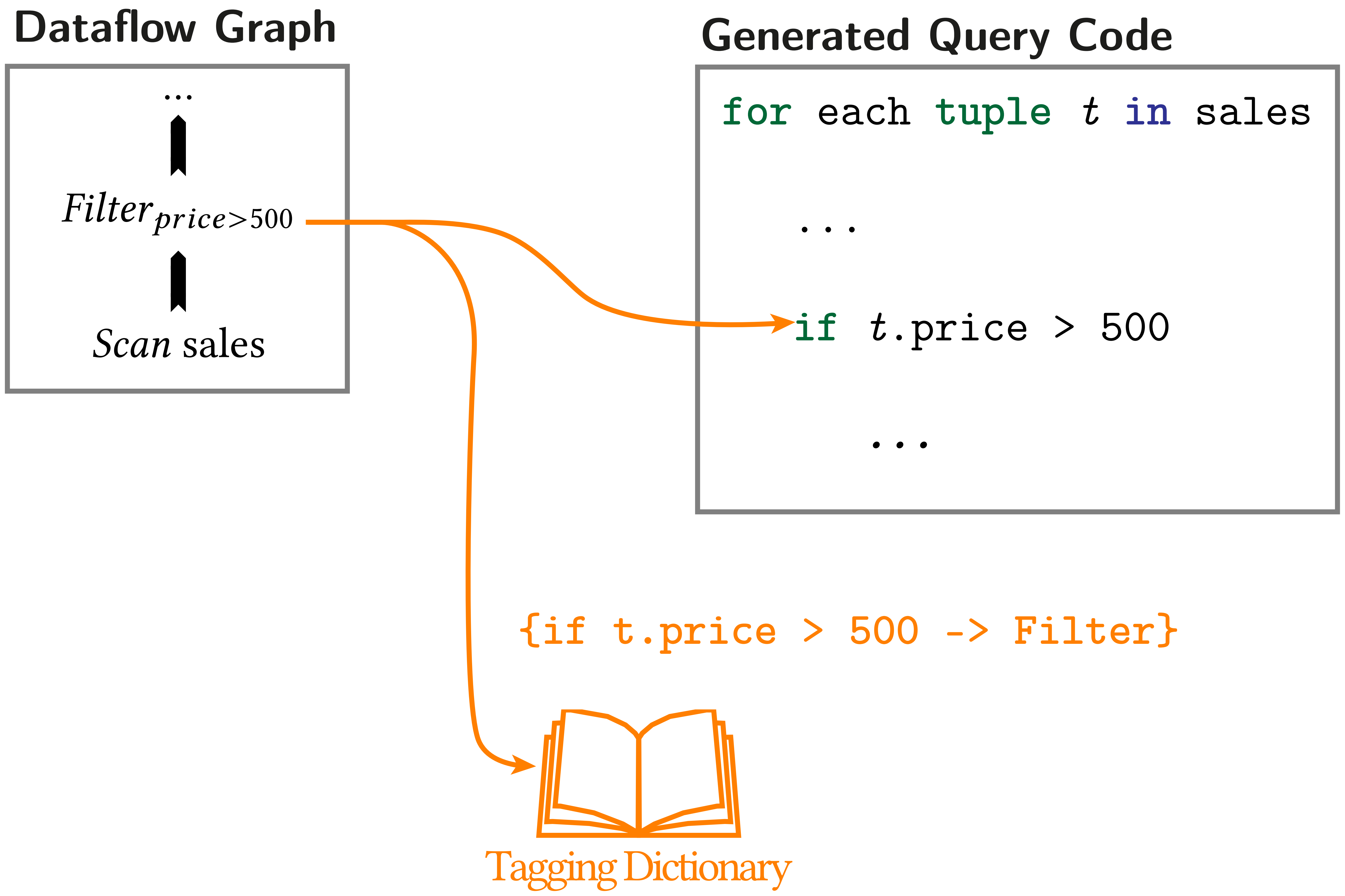
Tailored Profiling

Tagging Dictionary and Register Tagging



Tailored Profiling

Tagging Dictionary and Register Tagging



beischl@in.tum.de

Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

…

↑

$Filter_{price>500}$

↑

Scan sales

Generated Query Code

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

Tagging Dictionary

beischl@in.tum.de

Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

…

$\uparrow$

$Filter_{price>500}$

$\uparrow$

Scan sales

Generated Query Code

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

Profiling Sample

Source Line
malloc(...)

→

Tagging Dictionary

→ Scan, Filter ?

Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

...

$Filter_{price>500}$

Scan sales

Generated Query Code

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

Call-Stack Sample

Recorded Call-Stack
malloc(...)
Scan: call malloc()
...

Profiling Sample

Source Line
malloc(...)

→ Tagging Dictionary → Scan, Filter ?

Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

...

↑

$Filter_{price>500}$

↑

Scan sales

Generated Query Code

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

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

Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

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

Scan

beischl@in.tum.de

Tailored Profiling

Tagging Dictionary and Register Tagging

Dataflow Graph

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

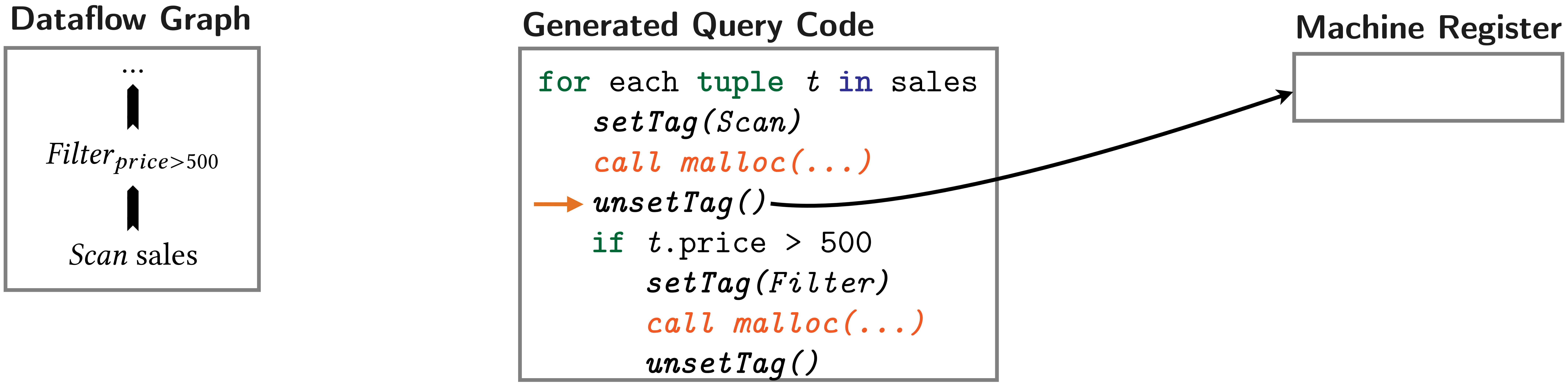
Scan

Profiling Sample

Source Line	Register Value
malloc(...)	Scan

Tailored Profiling

Tagging Dictionary and Register Tagging



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

$$\ldots$$
$$\uparrow$$
$$\mathit{Filter}_{price>500}$$
$$\uparrow$$
$$\mathit{Scan}\ \text{sales}$$

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

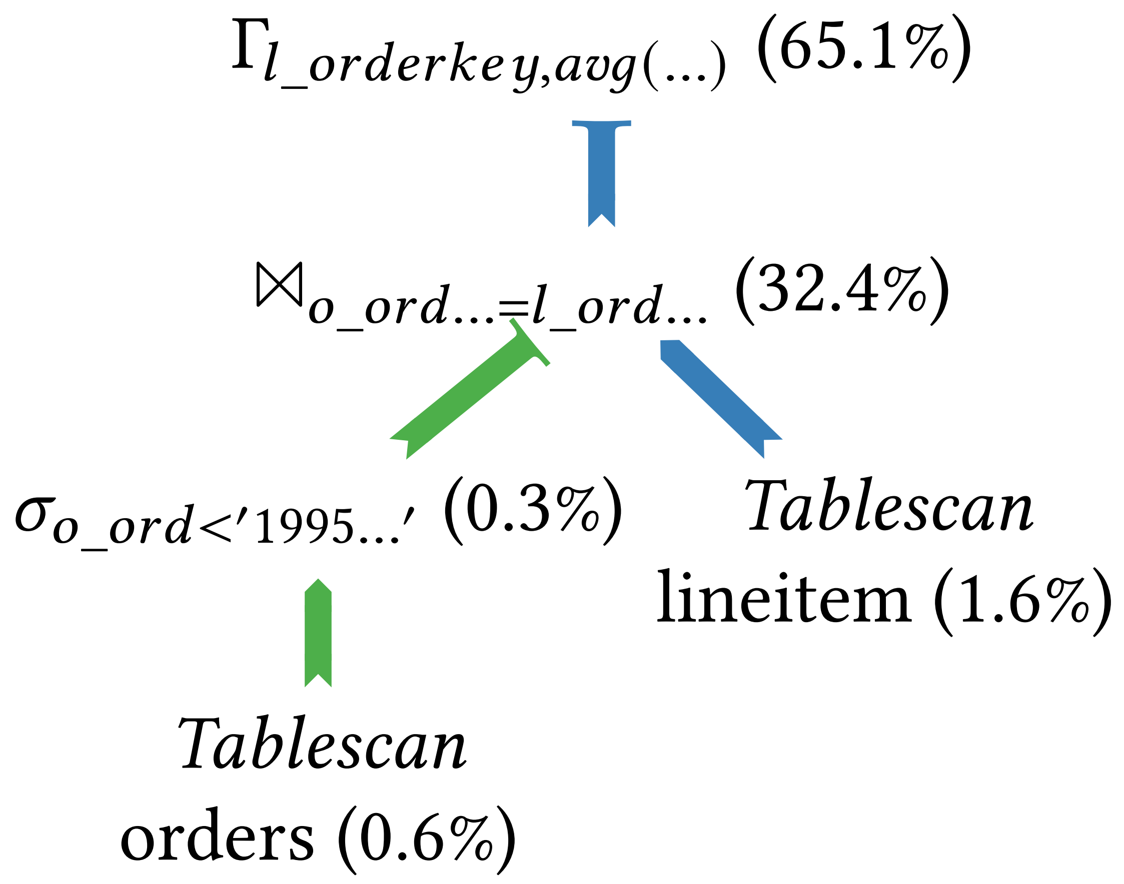
Source Line	Register Value
malloc(...)	Filter

Insights with Tailored Profiling

UMBRA

Insights with Tailored Profiling

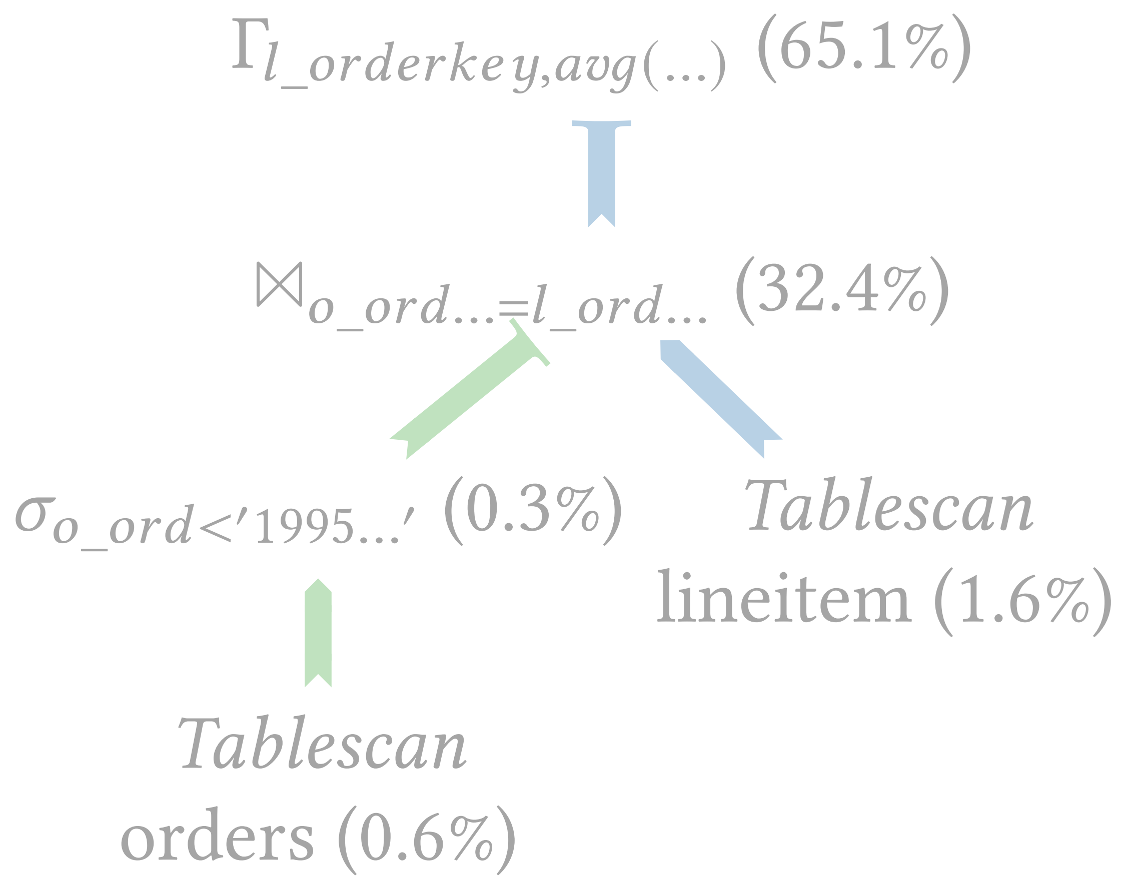
UMBRA

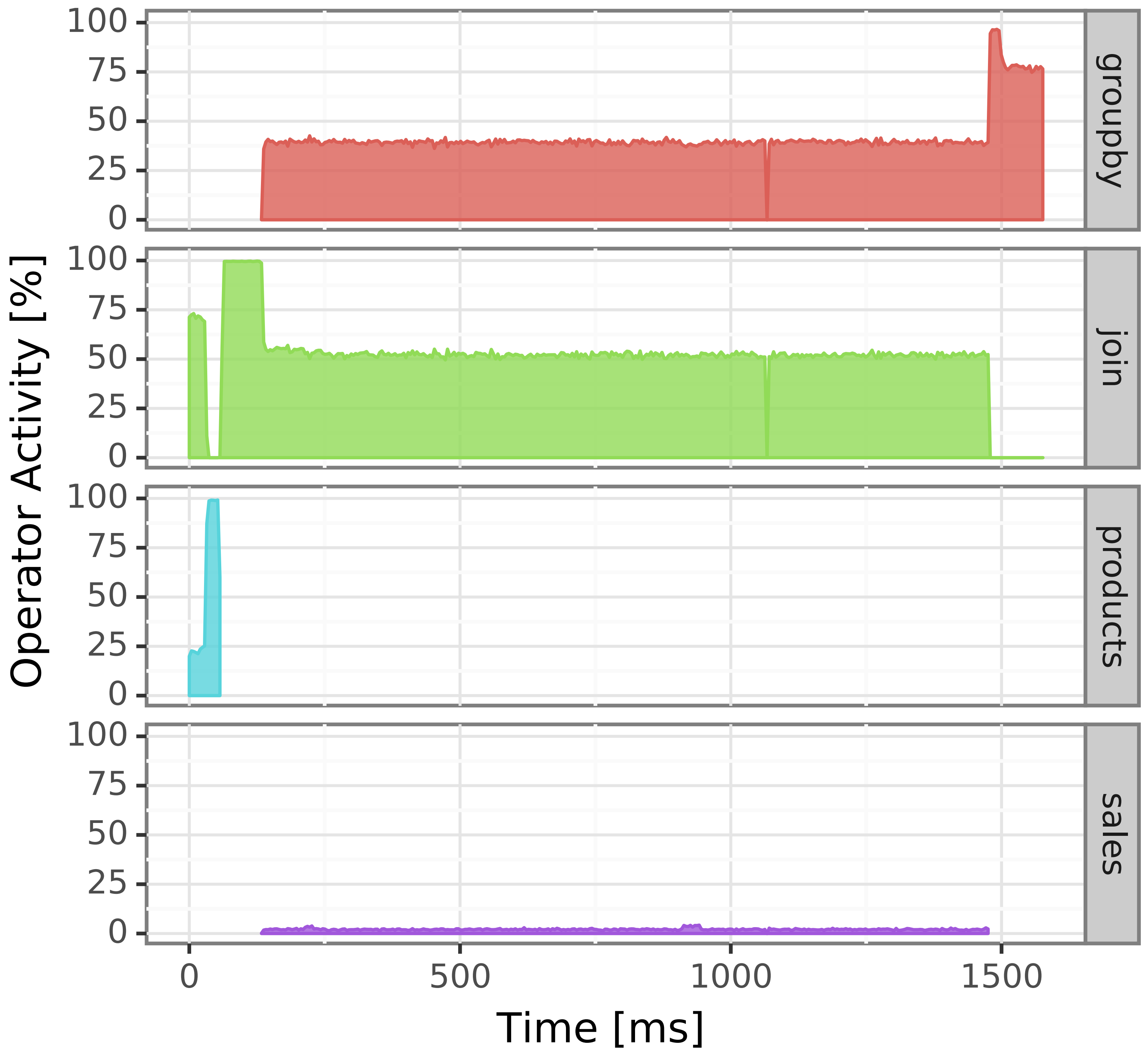


Time per operator

Insights with Tailored Profiling

UMBRA

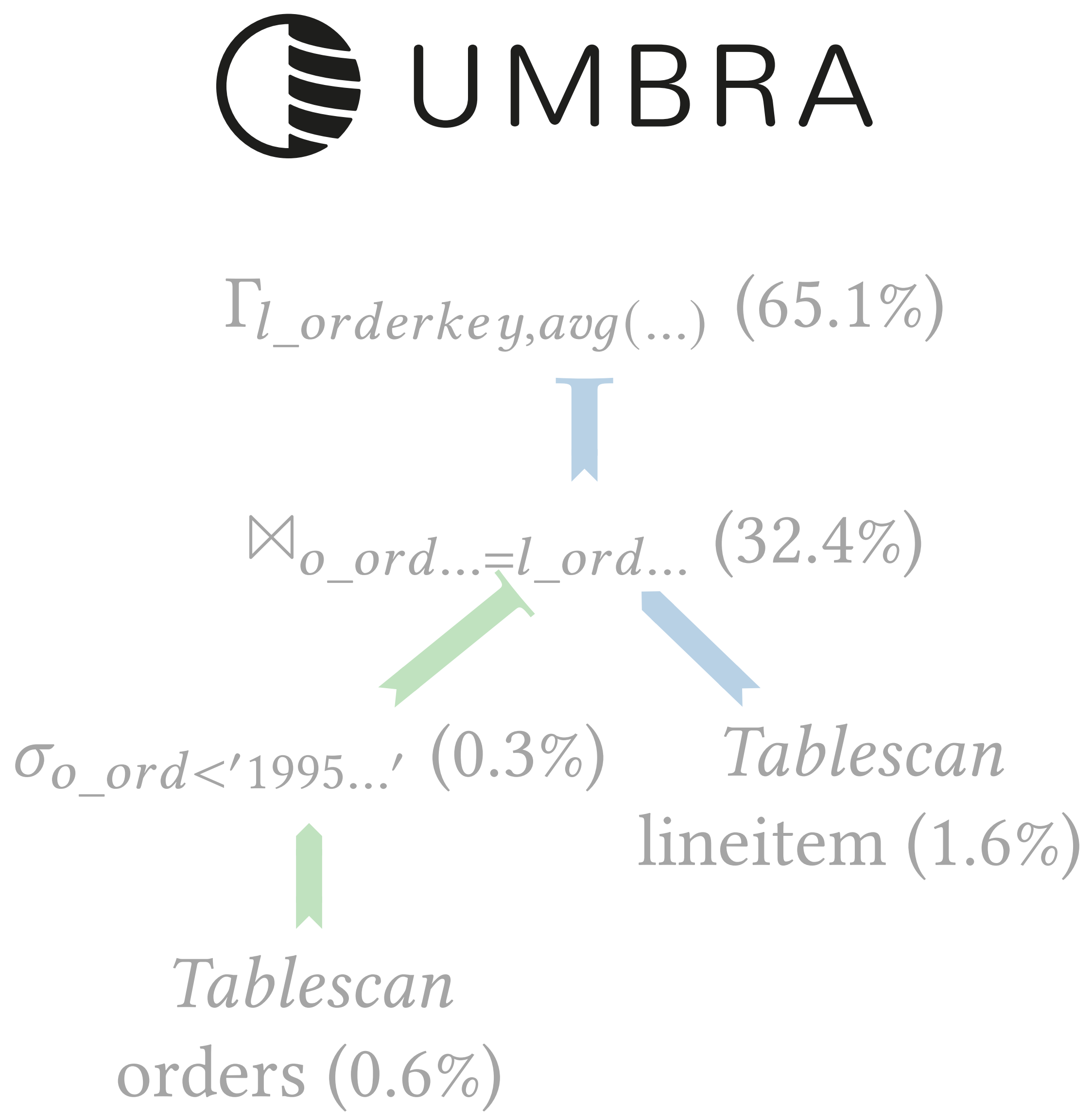


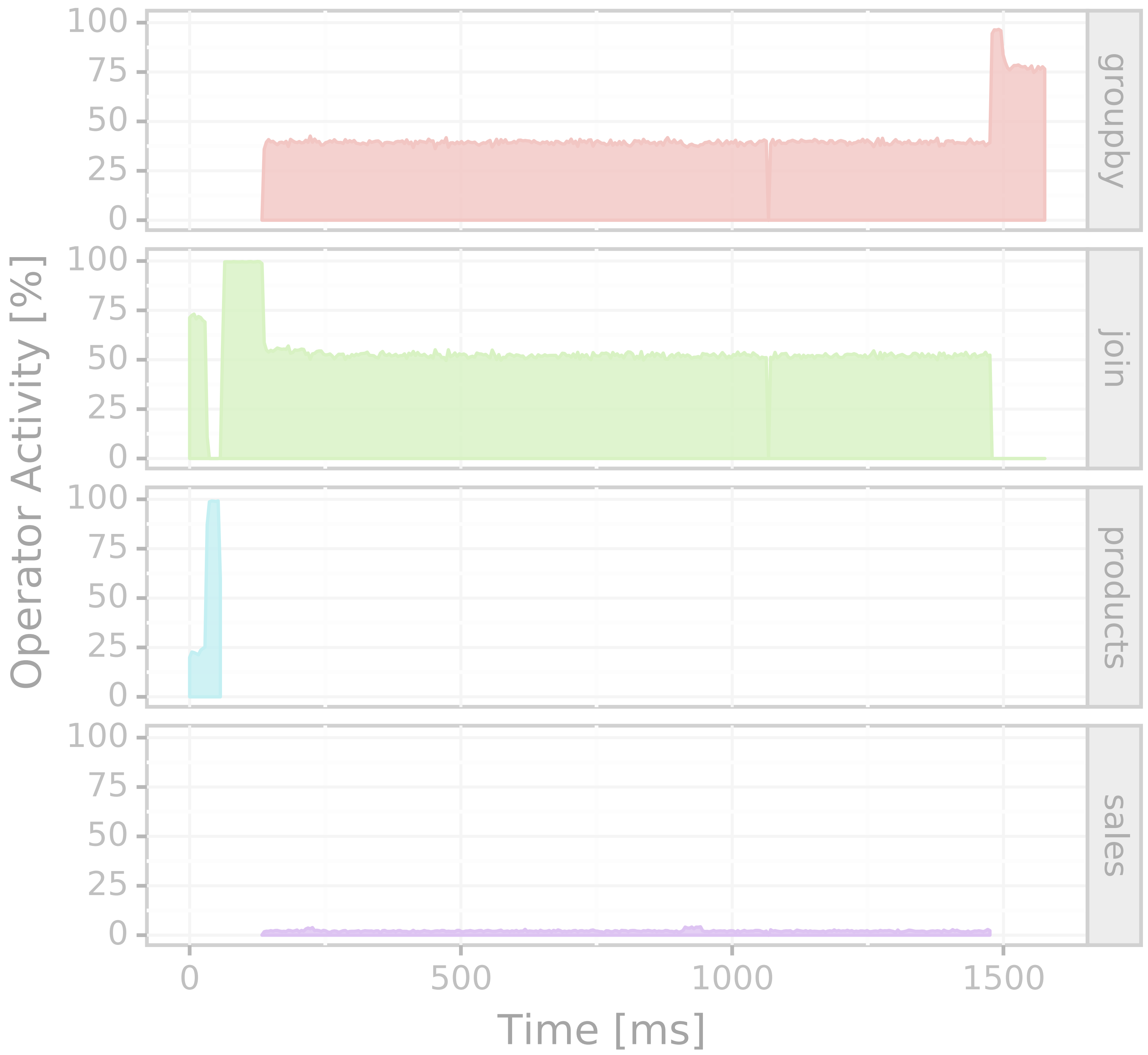


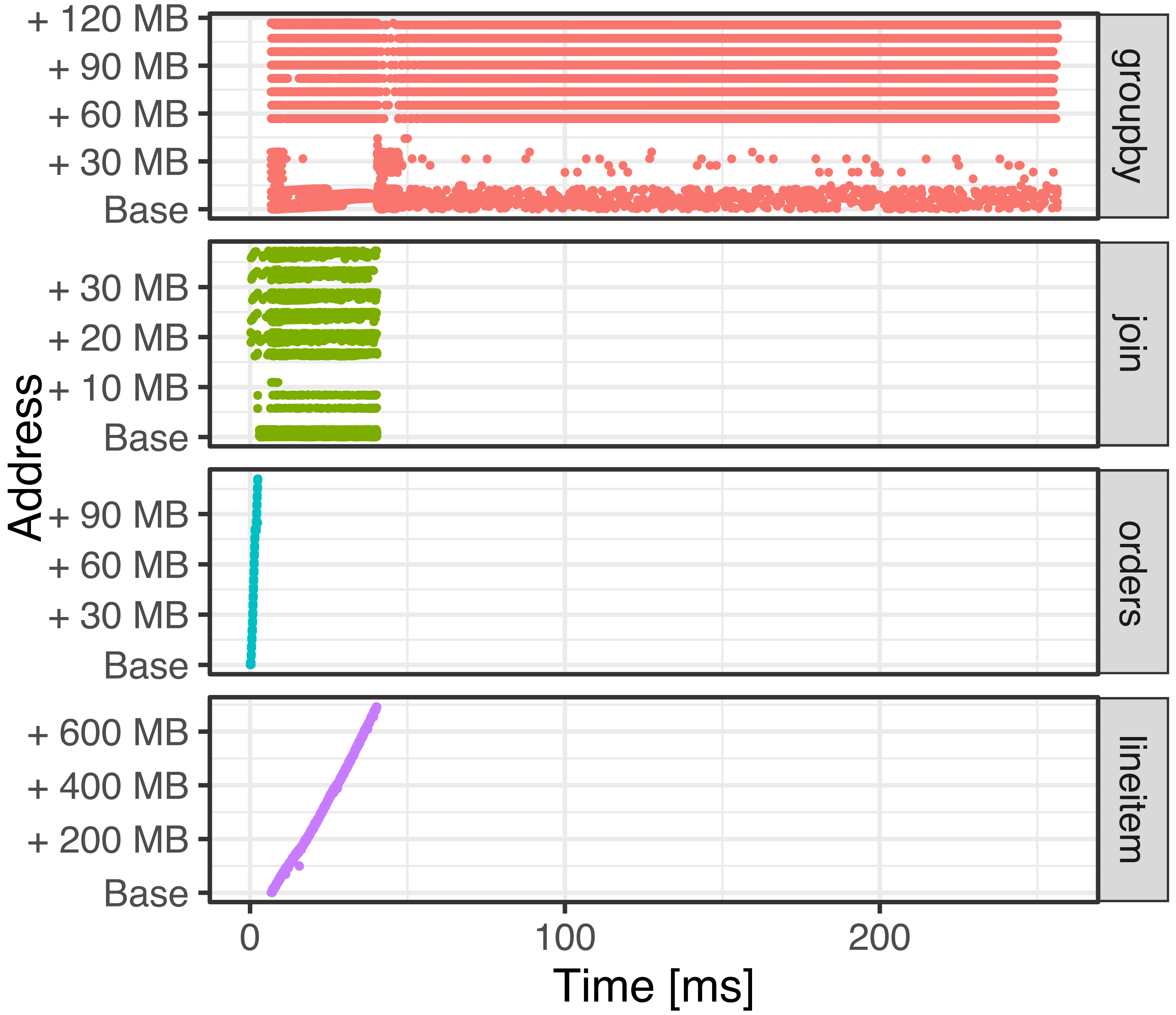
Time per operator

Context-aware profiling over time

Insights with Tailored Profiling







Time per operator

Context-aware profiling over time

Memory access patterns

Overhead and Integration Effort

Is it worth it?

Overhead and Integration Effort

Is it worth it?

- Little implementation effort

Component	Lines Added
Code Gen.	56
Sample Processing	1176
Visualizations	510

Overhead and Integration Effort

Is it worth it?

- Little implementation effort
- High accuracy

Component	Lines Added
Code Gen.	56
Sample Processing	1176
Visualizations	510

Attributed Samples	98 %

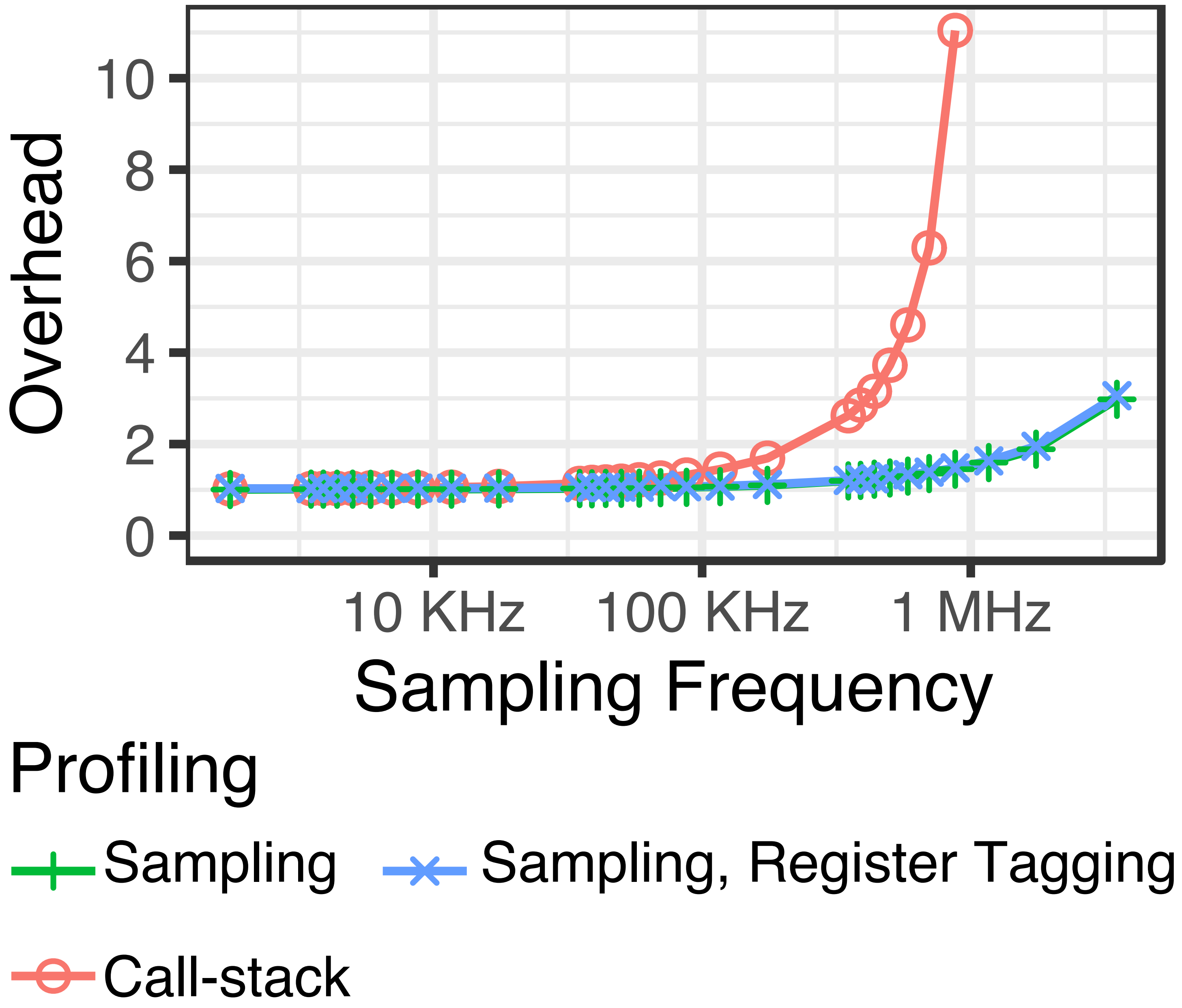
Overhead and Integration Effort

Is it worth it?

- Little implementation effort
- High accuracy
- Lightweight
  - Small query execution overhead
    - Tagging Dictionary: populated at compile time
    - Register Tagging: 3%
- Cheaper than call-stack sampling

Component	Lines Added
Code Gen.	56
Sample Processing	1176
Visualizations	510

Attributed Samples	98 %

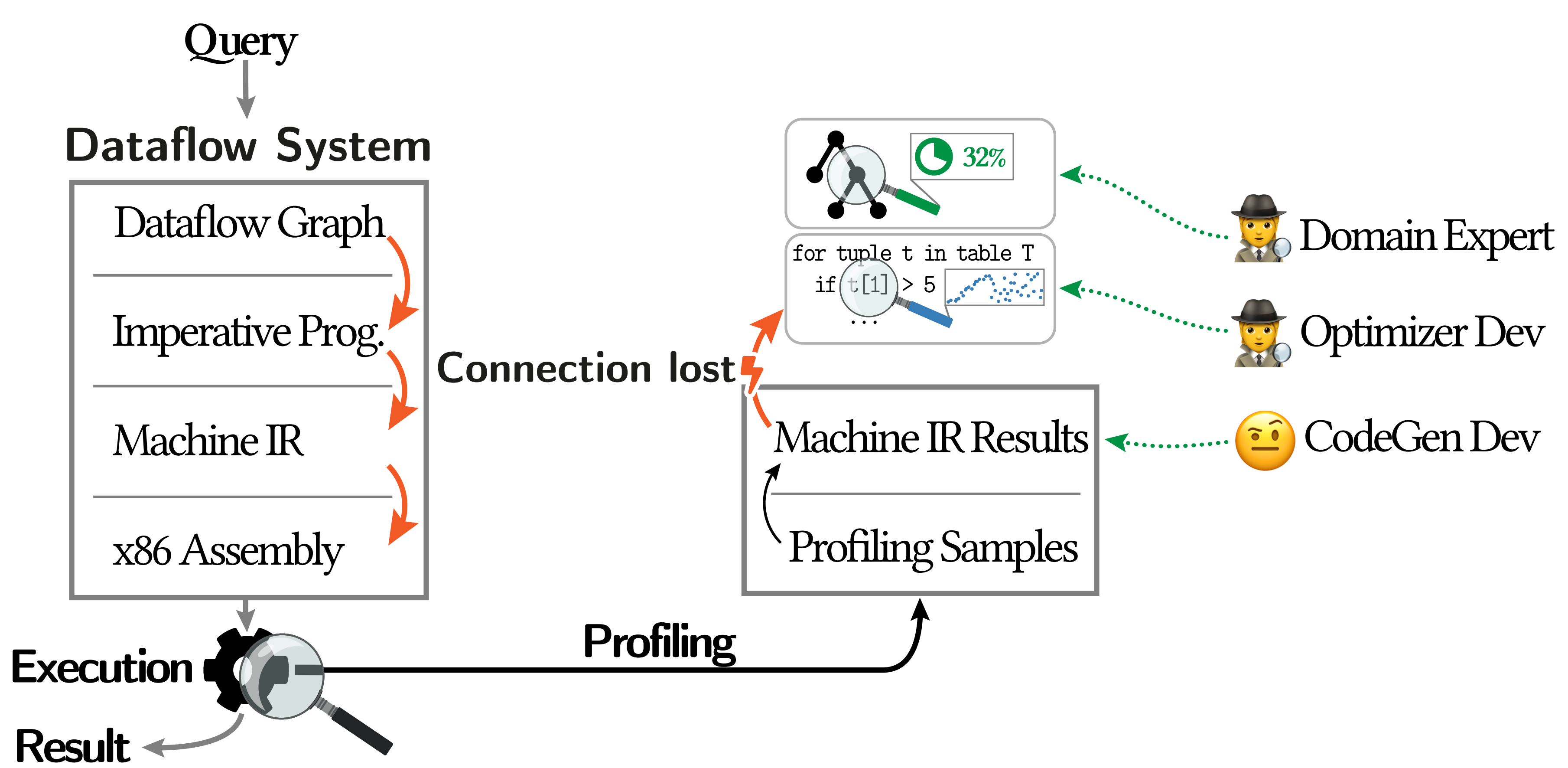


beischl@in.tum.de

Impact of Tailored Profiling

Where can you apply it?

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

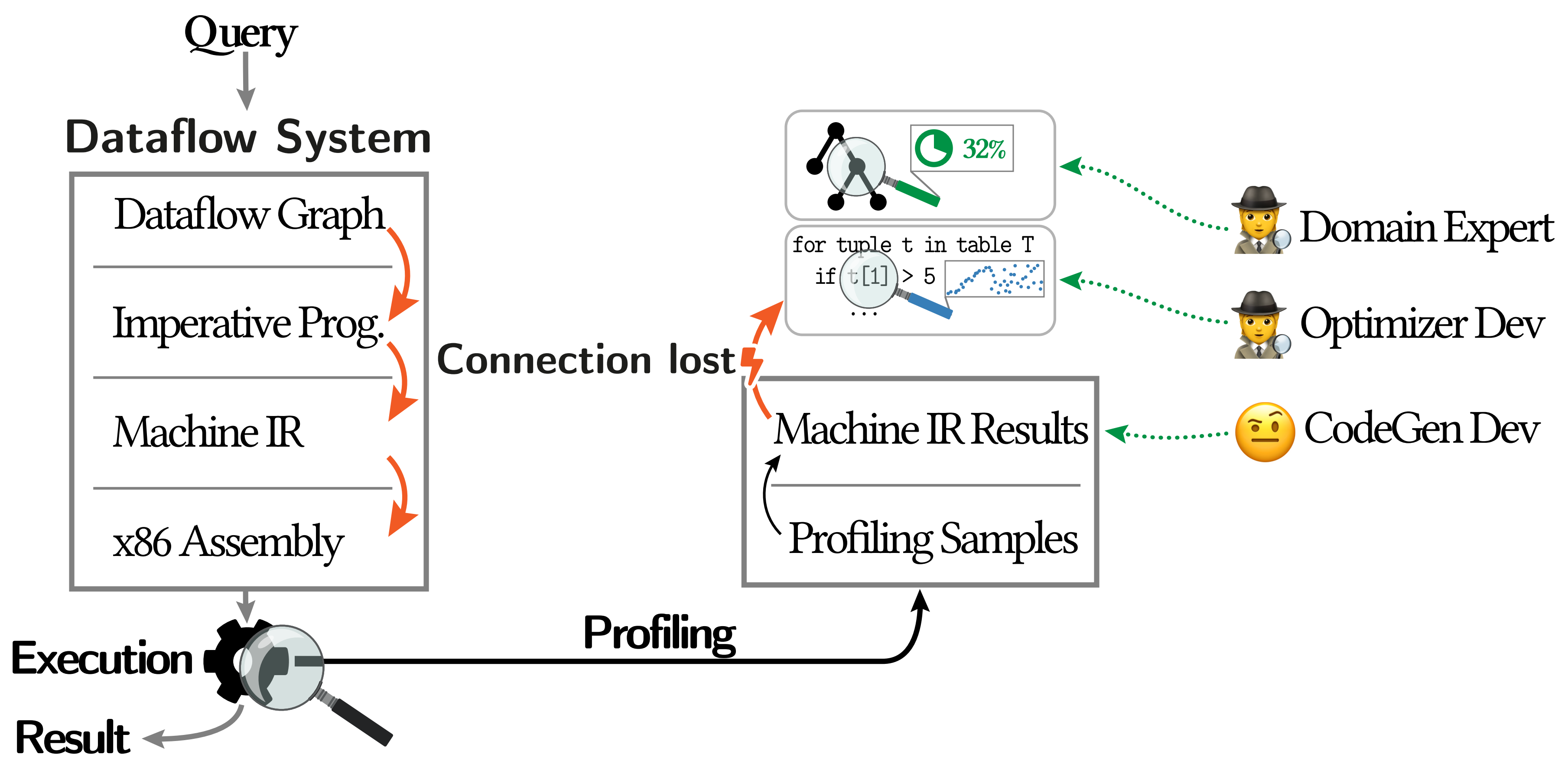


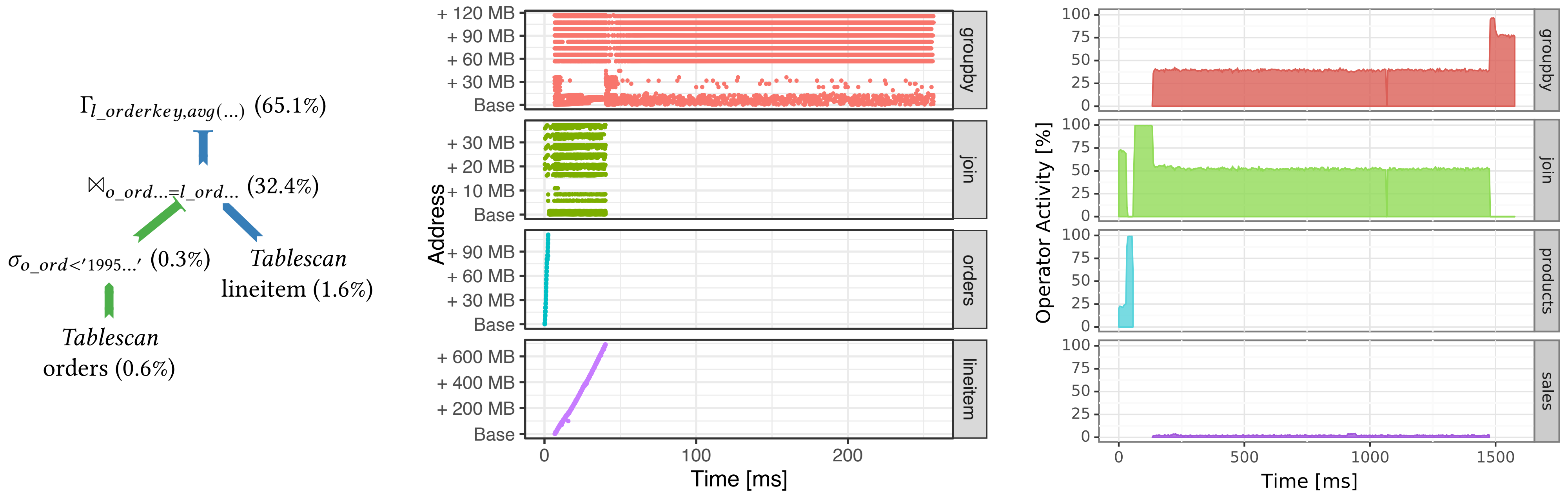
Impact of Tailored Profiling

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





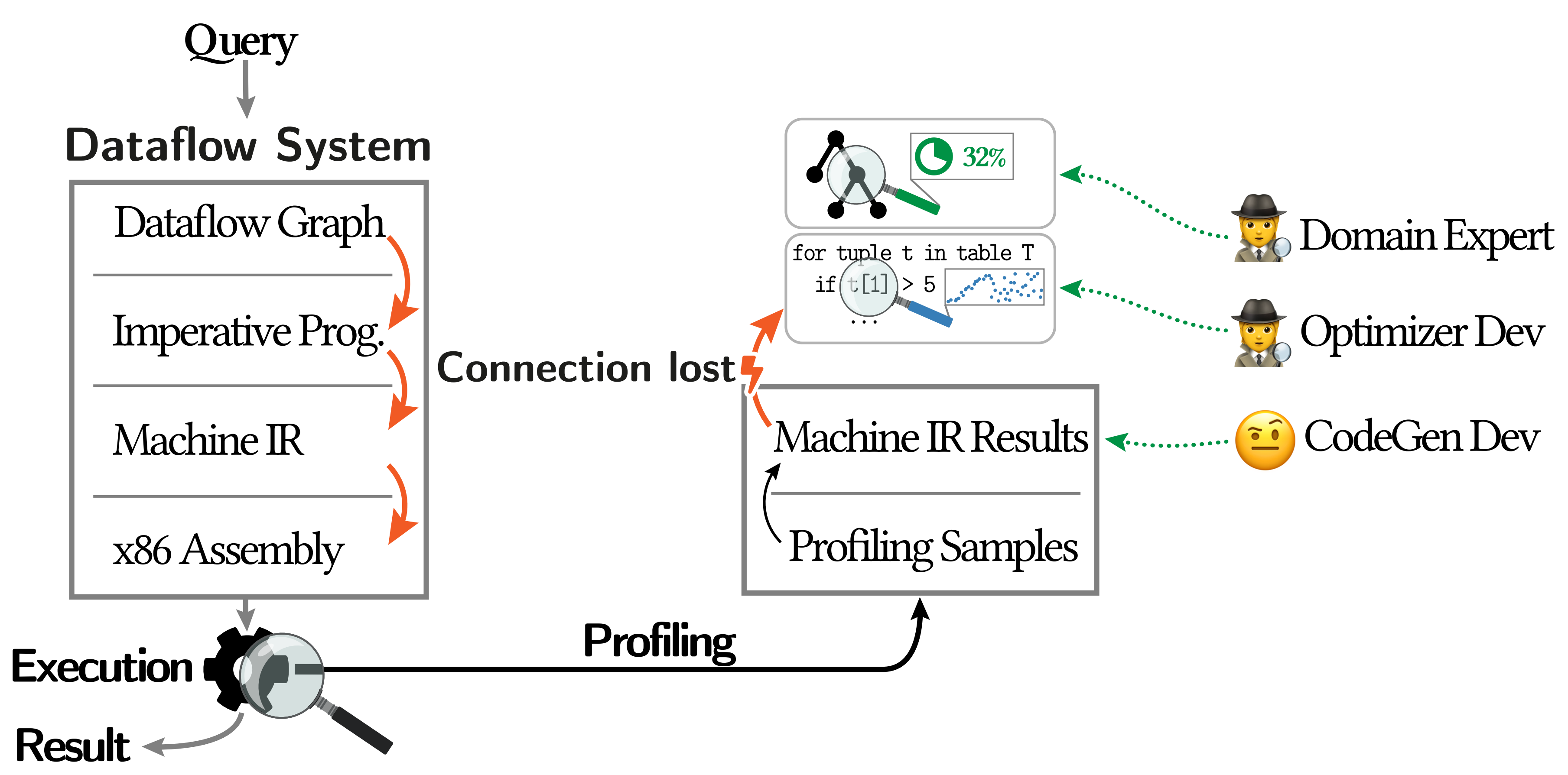
Impact of Tailored Profiling

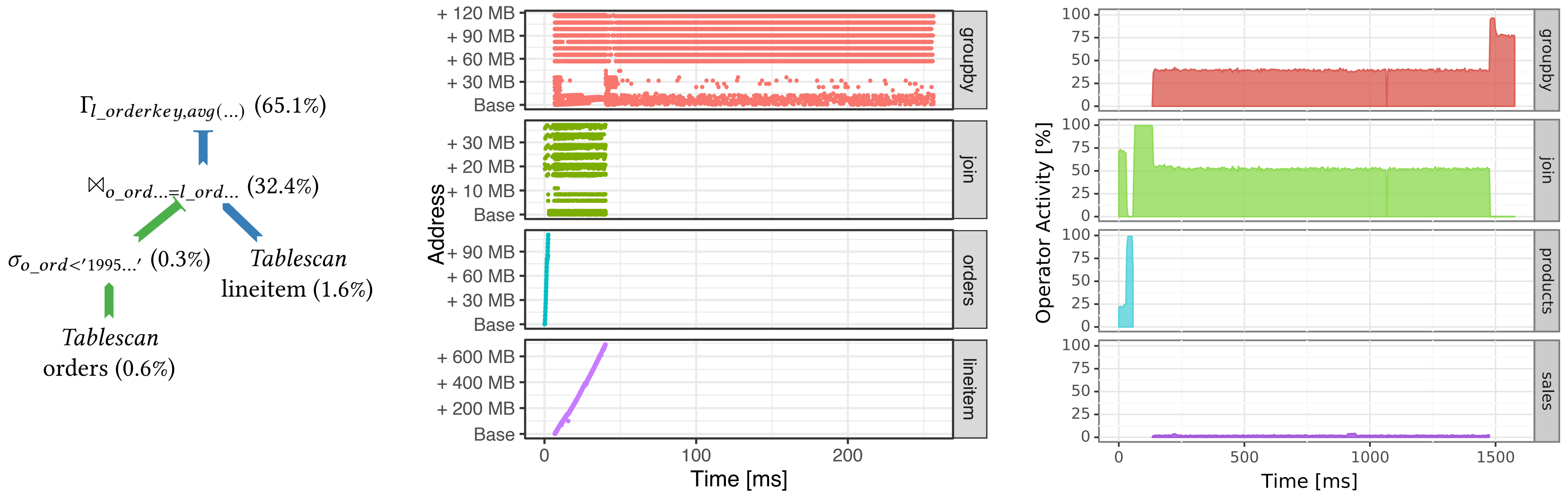
Where can you apply it?

- 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

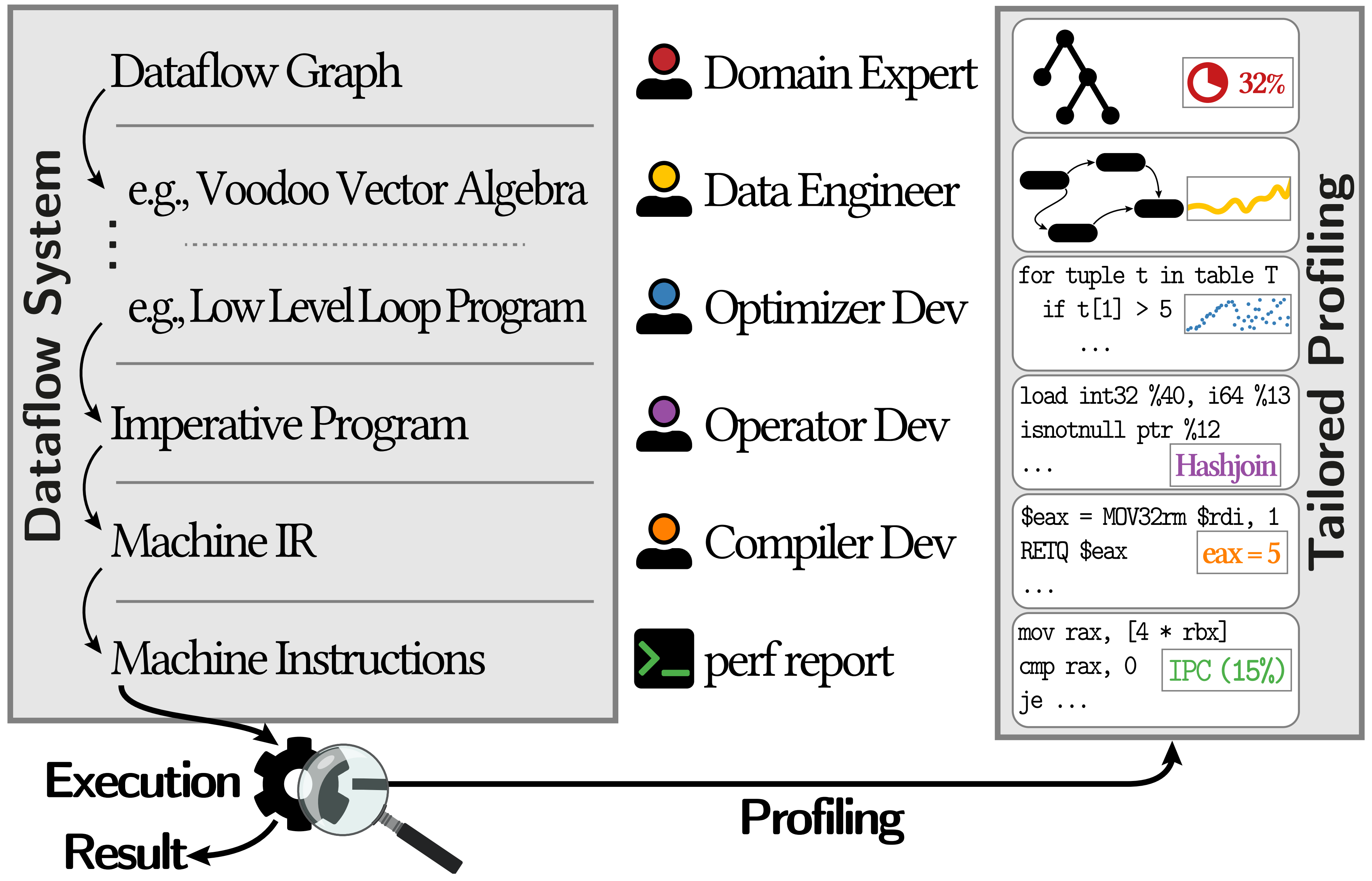




beischl@in.tum.de

What else can be found in the Paper

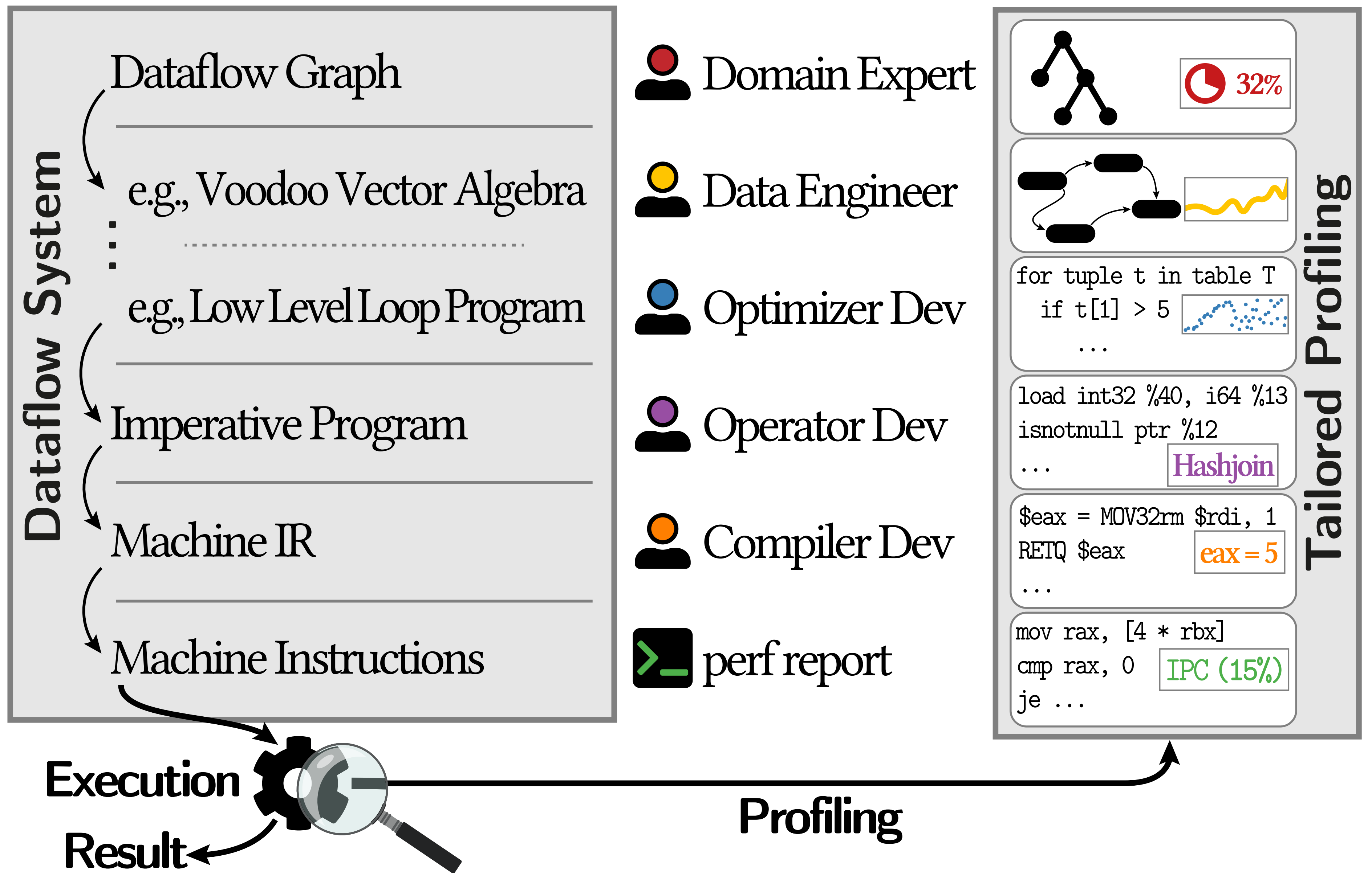
- Technical and implementation details
  - Tagging Dictionary for multiple abstraction levels
  - Register Tagging and call-stack sampling
  - Supported optimizations



What else can be found in the Paper

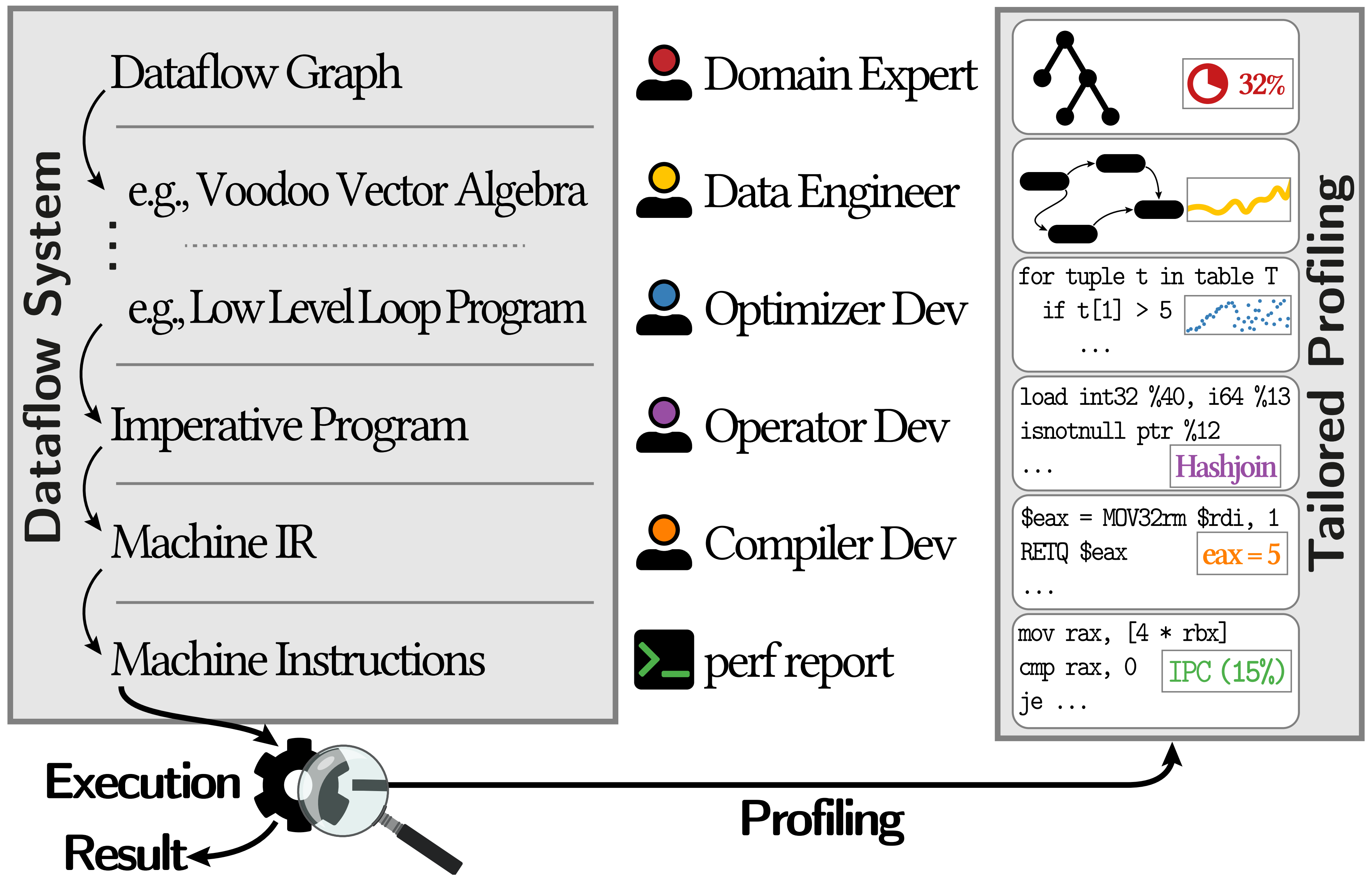
- Technical and implementation details
  - Tagging Dictionary for multiple abstraction levels
  - Register Tagging and call-stack sampling
  - Supported optimizations

- Integration details for our compiling DBMS Umbra



What else can be found in the Paper

- Technical and implementation details
  - Tagging Dictionary for multiple abstraction levels
  - Register Tagging and call-stack sampling
  - Supported optimizations
- Integration details for our compiling DBMS Umbra
- More detailed evaluation, runtime overhead and use cases



Thank you for watching!

beischl@in.tum.de