UNDERSTANDING DISTRIBUTED DATAFLOW SYSTEMS

John Liagouris

liagos@inf.ethz.ch

OUTPUT EXPLANATION AND PERFORMANCE ANALYSIS

Google

3 May 2017

PART I: Why is this record in the output of my distributed dataflow?

- Concise explanations of individual outputs
- On-demand output reproduction

PART II: Why is my distributed dataflow slow?

- Bottleneck detection
- Critical path analysis

COLLABORATORS

Desislava Dimitrova

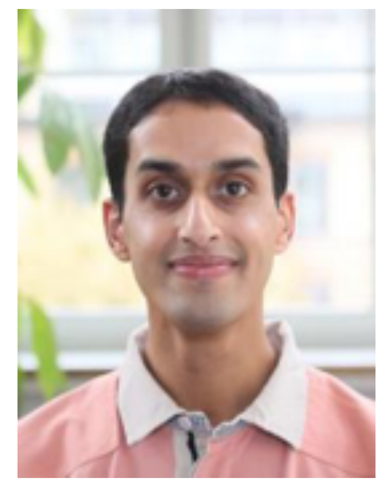
Vasiliki Kalavri

Ralf Sager

Andrea Lattuada

Frank McSherry

Moritz Hoffmann



Zaheer Chothia

Sebastian Wicki

Timothy Roscoe

THE BIG PICTURE: UNDERSTANDING THE DATACENTER

Strymon

Enterprise Datacenter

event logs

- The volume of datacenter logs is huge
- Keeping archives is not a viable solution
- We can process logs online

THE BIG PICTURE: UNDERSTANDING THE DATACENTER

Enterprise Datacenter

event logs

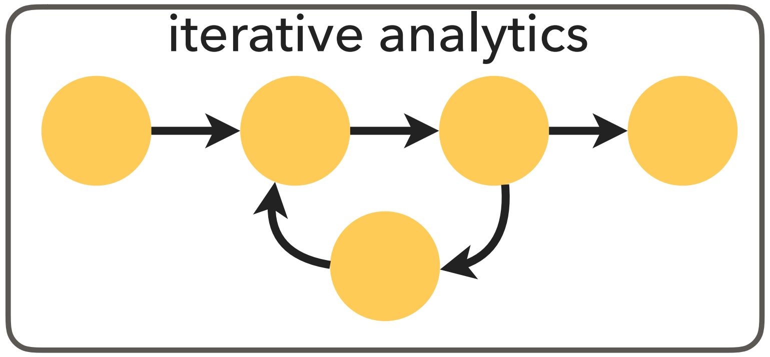
Strymon

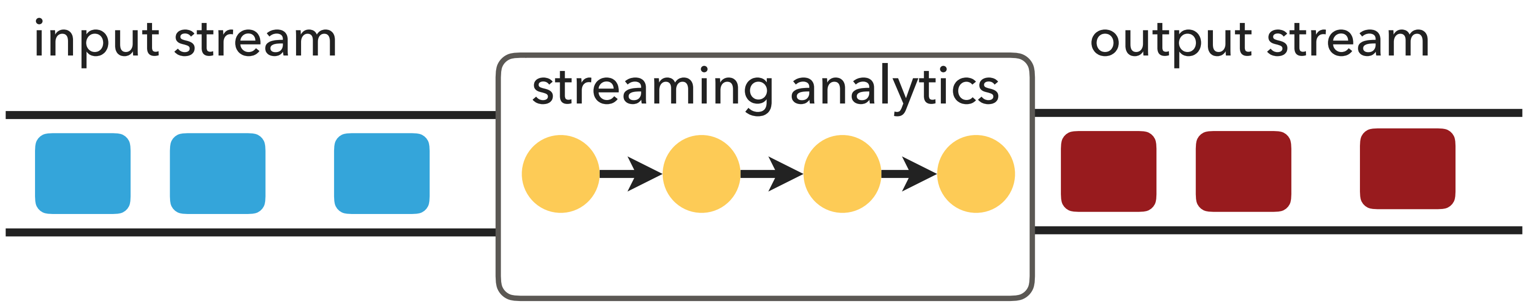
Strymon is a novel system able to:

- Perform deep analytics on thousands of distributed streams of event logs in parallel
- Explain its outputs interactively

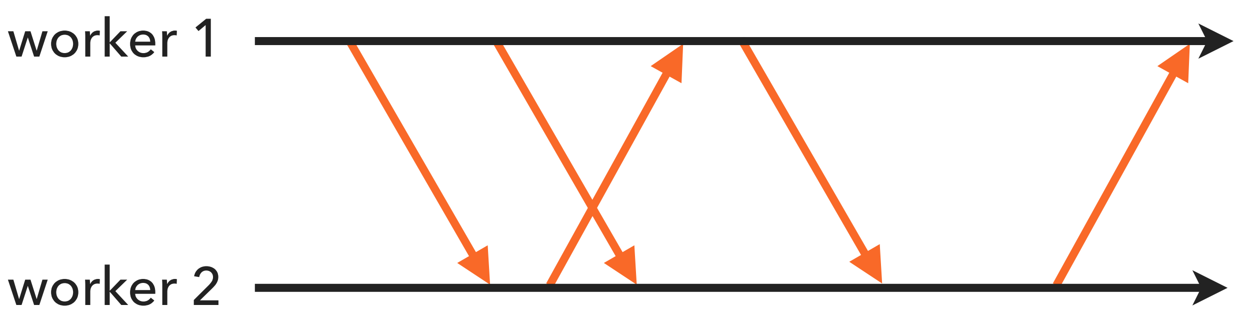
IDEAS IN STRYMON CAN BE GENERALIZED

for dataflow systems





and different execution models



synchronous vs asynchronous

shared-nothing vs shared-memory

TIMELY DATAFLOW

D. Murray, F. McSherry, M. Isard, R. Isaacs, P. Barham, M. Abadi. Naiad: A Timely Dataflow System. In SOSP, 2013.

- A steaming framework for data-parallel computations
  - Cyclic dataflows
  - Logical timestamps (epochs)
  - Asynchronous execution
  - Low latency

DIFFERENTIAL DATAFLOW

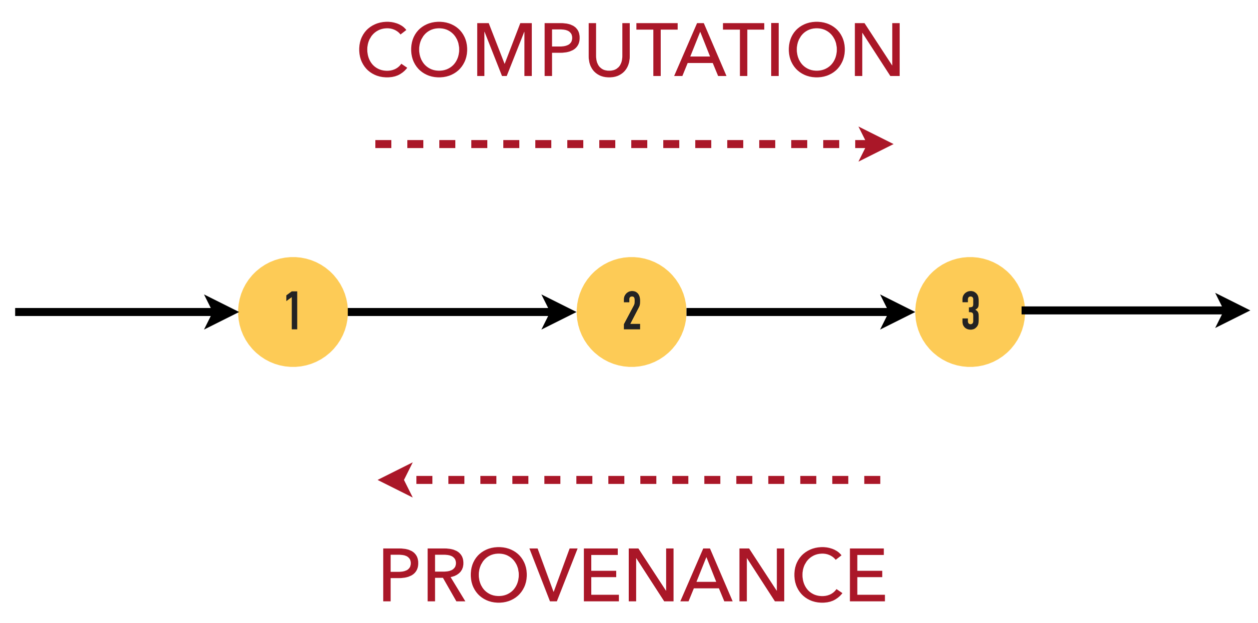
F. McSherry, D. Murray, R. Isaacs, M. Isard. *Differential Dataflow*. In CIDR, 2013.

- A high-level API on top of Timely Dataflow
  - Incremental computation

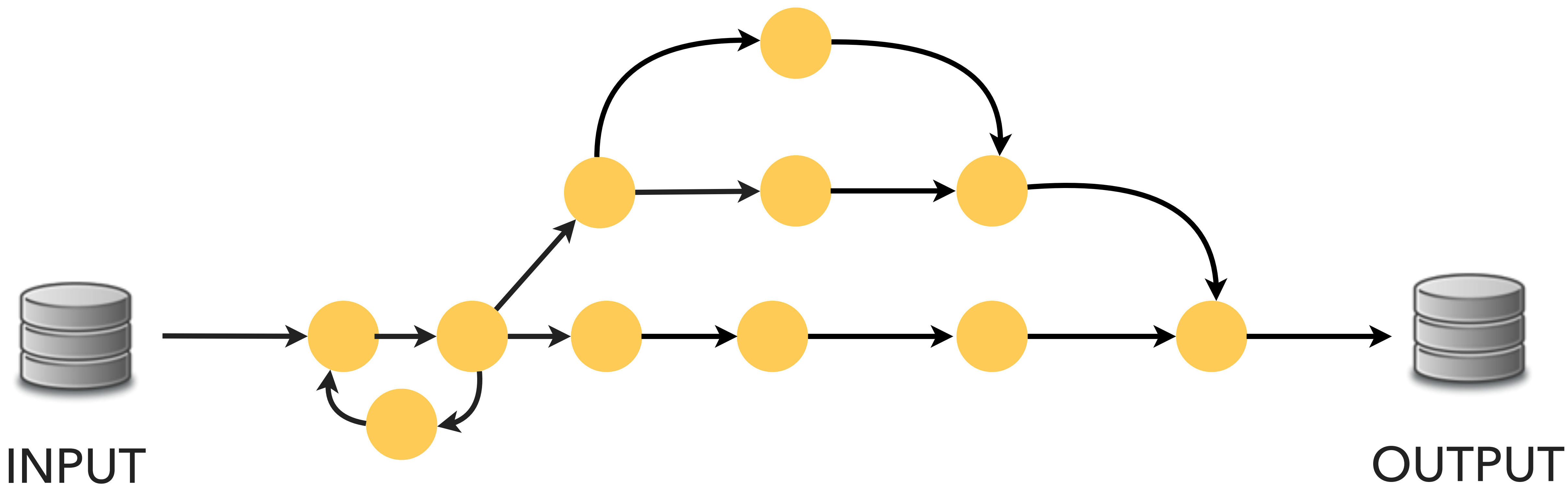
PART I

Why is this record in the output of my distributed dataflow?

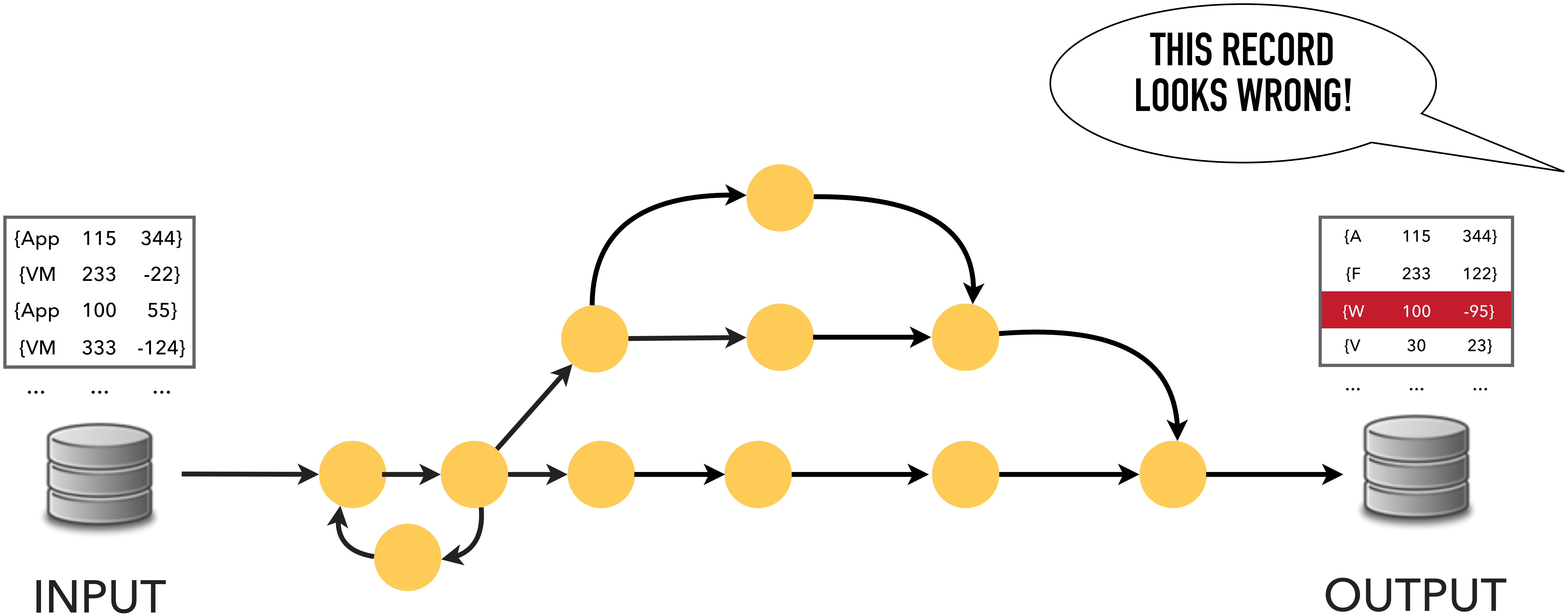
EXPLANATIONS IN DATABASES



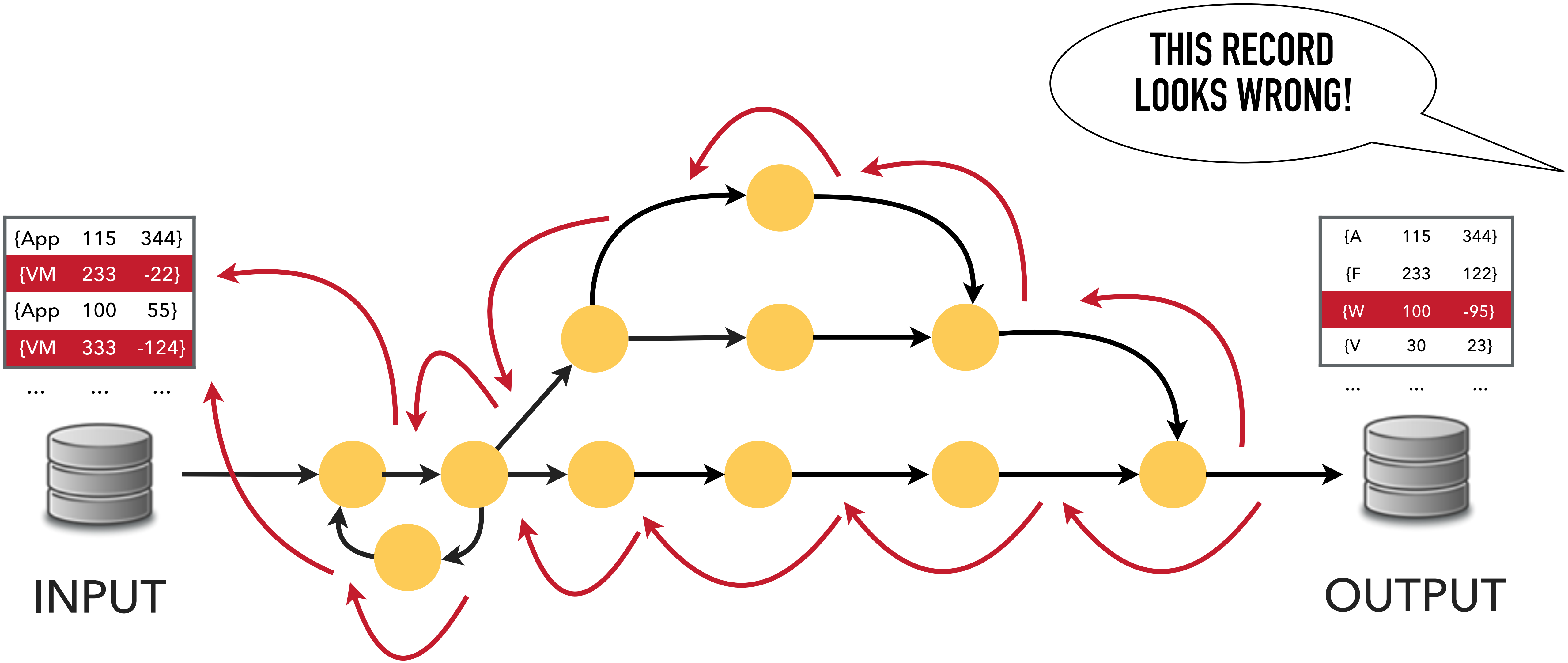
THE PROBLEM: OUTPUT EXPLANATION



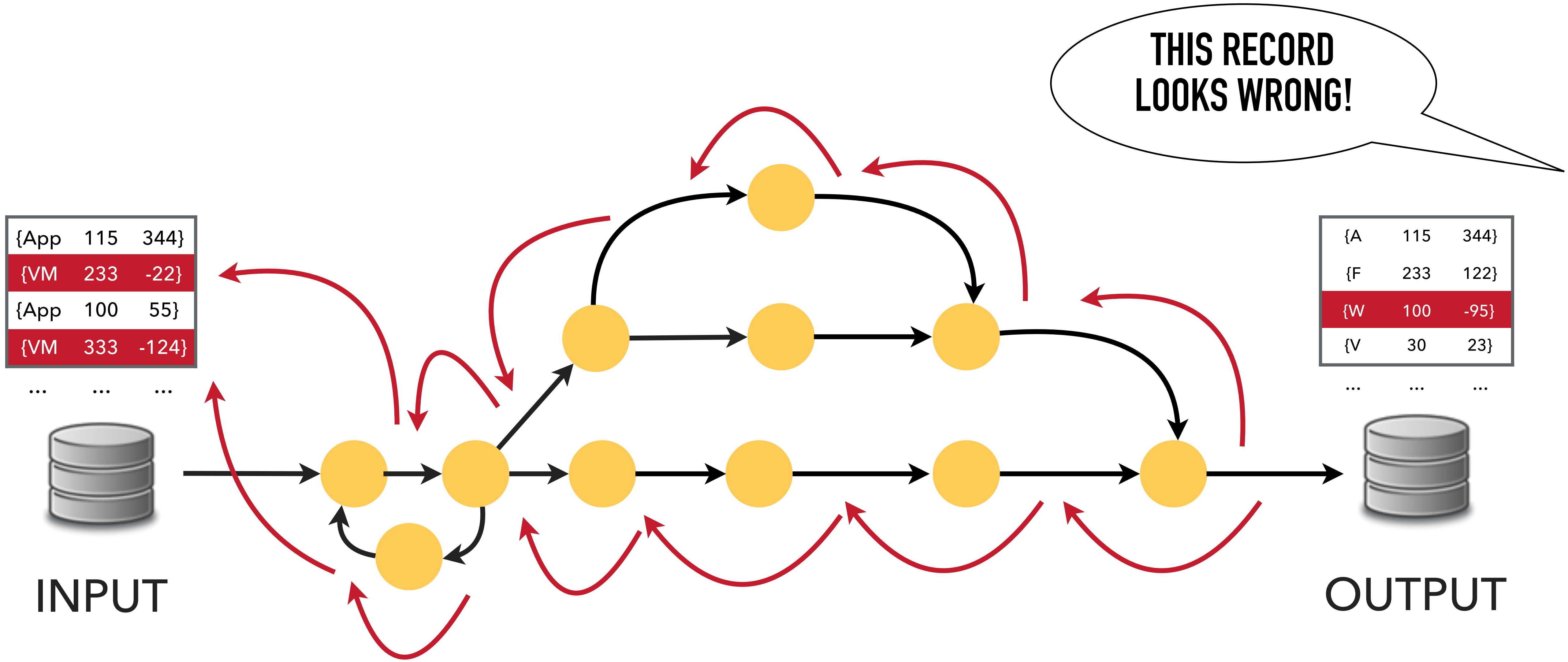
THE PROBLEM: OUTPUT EXPLANATION



THE PROBLEM: OUTPUT EXPLANATION

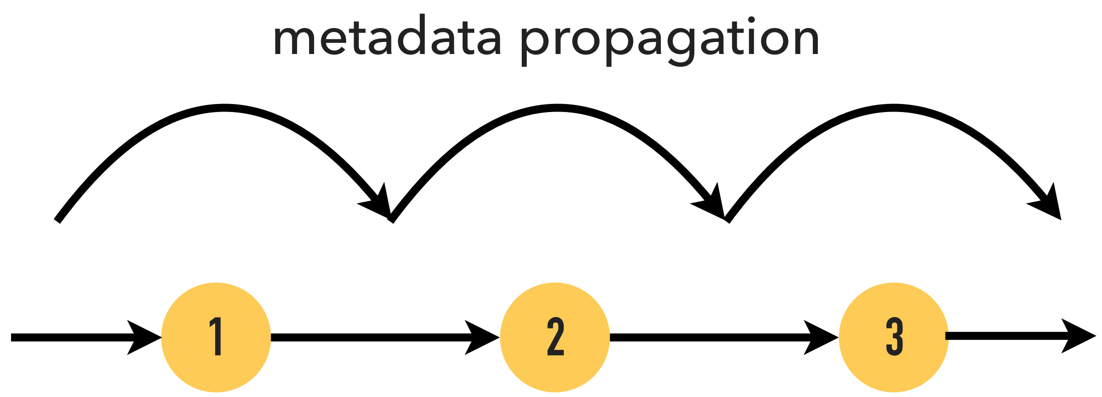


THE PROBLEM: OUTPUT EXPLANATION



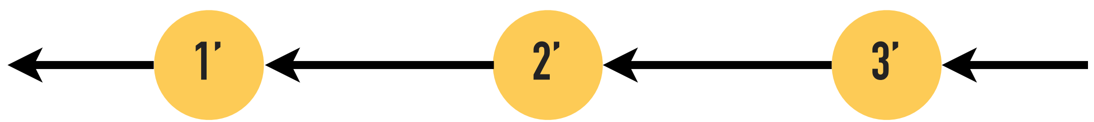
Output explanation: A subset of the input that is sufficient to reproduce the selected subset of the output

ANNOTATION-BASED TECHNIQUES



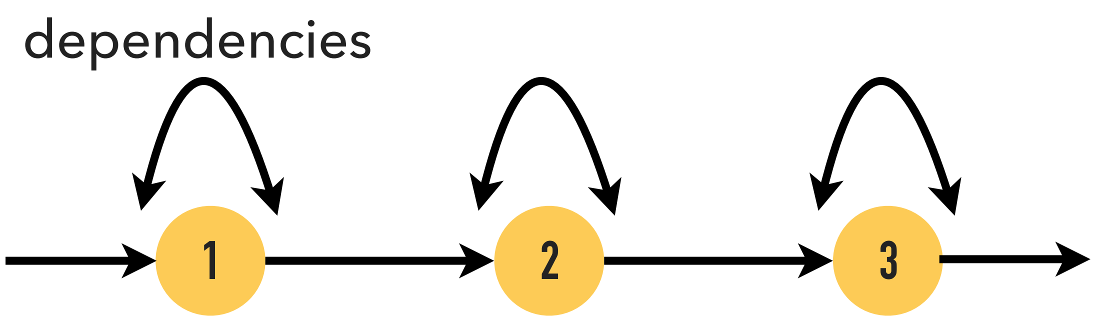
- Fast
- Explode in size

INVERSION-BASED TECHNIQUES



- Small memory footprint
- Not generally applicable

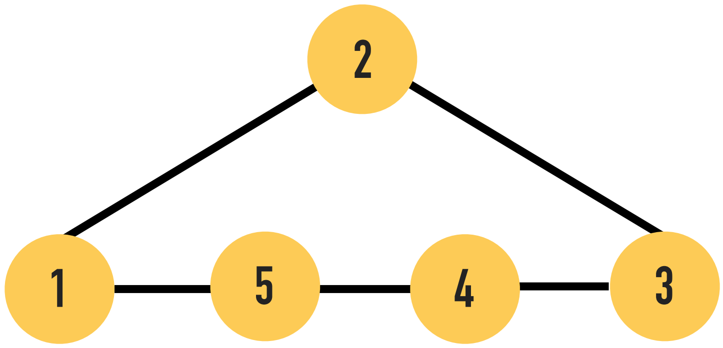
BACKWARD TRACING



- Small memory footprint
- Generally applicable
- Fast

PROBLEM 1: TOO MUCH INFORMATION

Use Case: Graph Rechability

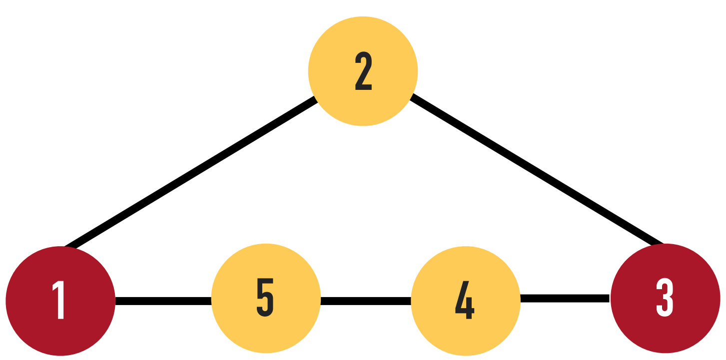


PROBLEM 1: TOO MUCH INFORMATION

Use Case: Graph Reachability

WHY IS (1,3) IN THE OUTPUT?

- Record (1,3) appears in the result

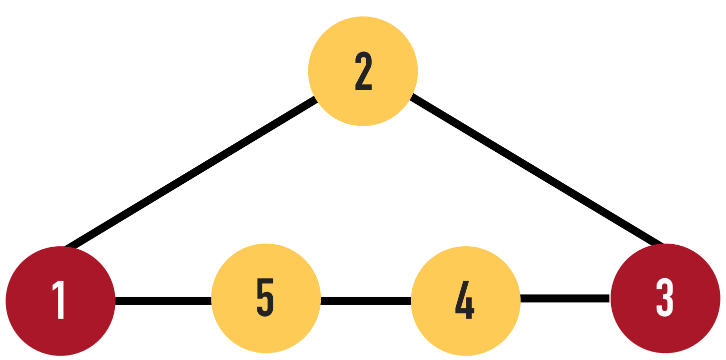


PROBLEM 1: TOO MUCH INFORMATION

Use Case: Graph Reachability

WHY IS (1,3) IN THE OUTPUT?

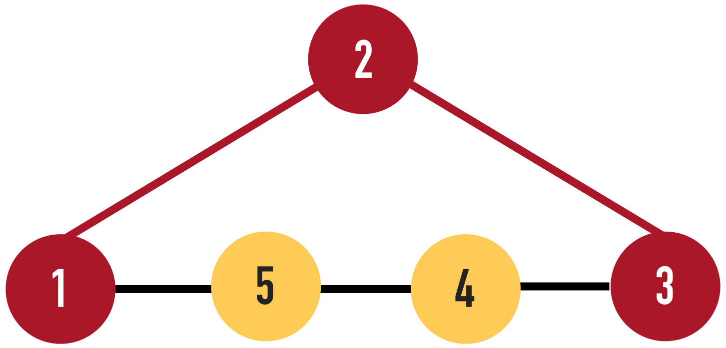
- Record (1,3) appears in the result
- Naive backward tracing returns as an explanation all edges of the graph



PROBLEM 1: TOO MUCH INFORMATION

Use Case: Graph Reachability

WHY IS (1,3) IN THE OUTPUT?



- Record (1,3) appears in the result
- Naive backward tracing returns as an explanation all edges of the graph
- A shortest path suffices

PROBLEM 2: NOT ENOUGH INFORMATION

Use Case: Word Set Difference

A

THE QUICK
BROWN FOX
...

B

THE LAZY DOG
...

PROBLEM 2: NOT ENOUGH INFORMATION

Use Case: Word Set Difference

WHY ONLY 3 WORDS ARE UNIQUE TO DOCUMENT A?

- Record (doc A, 3 unique words) appears in the result

A

THE QUICK BROWN FOX
...

(doc A, 3 unique words)

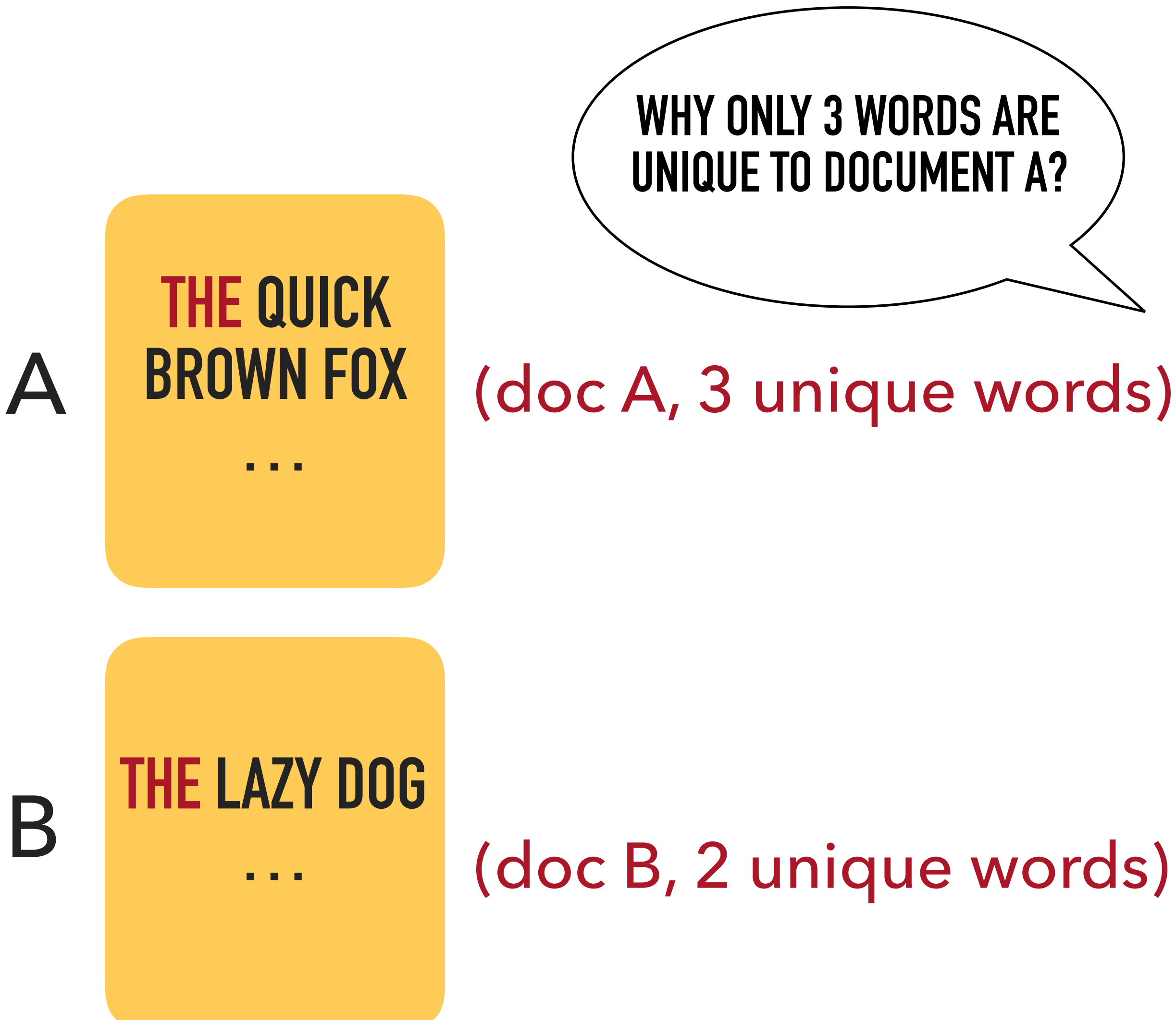
B

THE LAZY DOG
...

(doc B, 2 unique words)

PROBLEM 2: NOT ENOUGH INFORMATION

Use Case: Word Set Difference



- Record (doc A, 3 unique words) appears in the result
- Naive backward tracing returns as an explanation only the words of doc A

PROBLEM 2: NOT ENOUGH INFORMATION

Use Case: Word Set Difference

WHY ONLY 3 WORDS ARE UNIQUE TO DOCUMENT A?

A

THE QUICK BROWN FOX
...

(doc A, 3 unique words)

B

THE LAZY DOG
...

(doc B, 2 unique words)

- Record (doc A, 3 unique words) appears in the result
- Naive backward tracing returns as an explanation only the words of doc A
- We also need the words of doc B to reproduce the record (doc A, 3 unique words)

CAN WE SOLVE BOTH PROBLEMS?

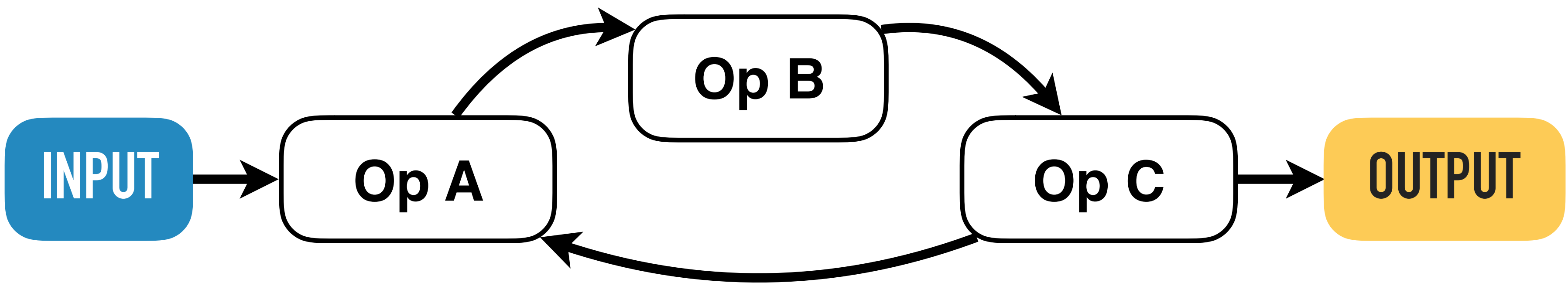
Yes! Given that the system is able to:

- Keep track of the exact point in the computation a data record was produced
- Detect divergent records when replaying the computation on a subset of the input

We exploit the main features of Differential Dataflow

EXPLANATIONS WITH DIFFERENTIAL DATAFLOW

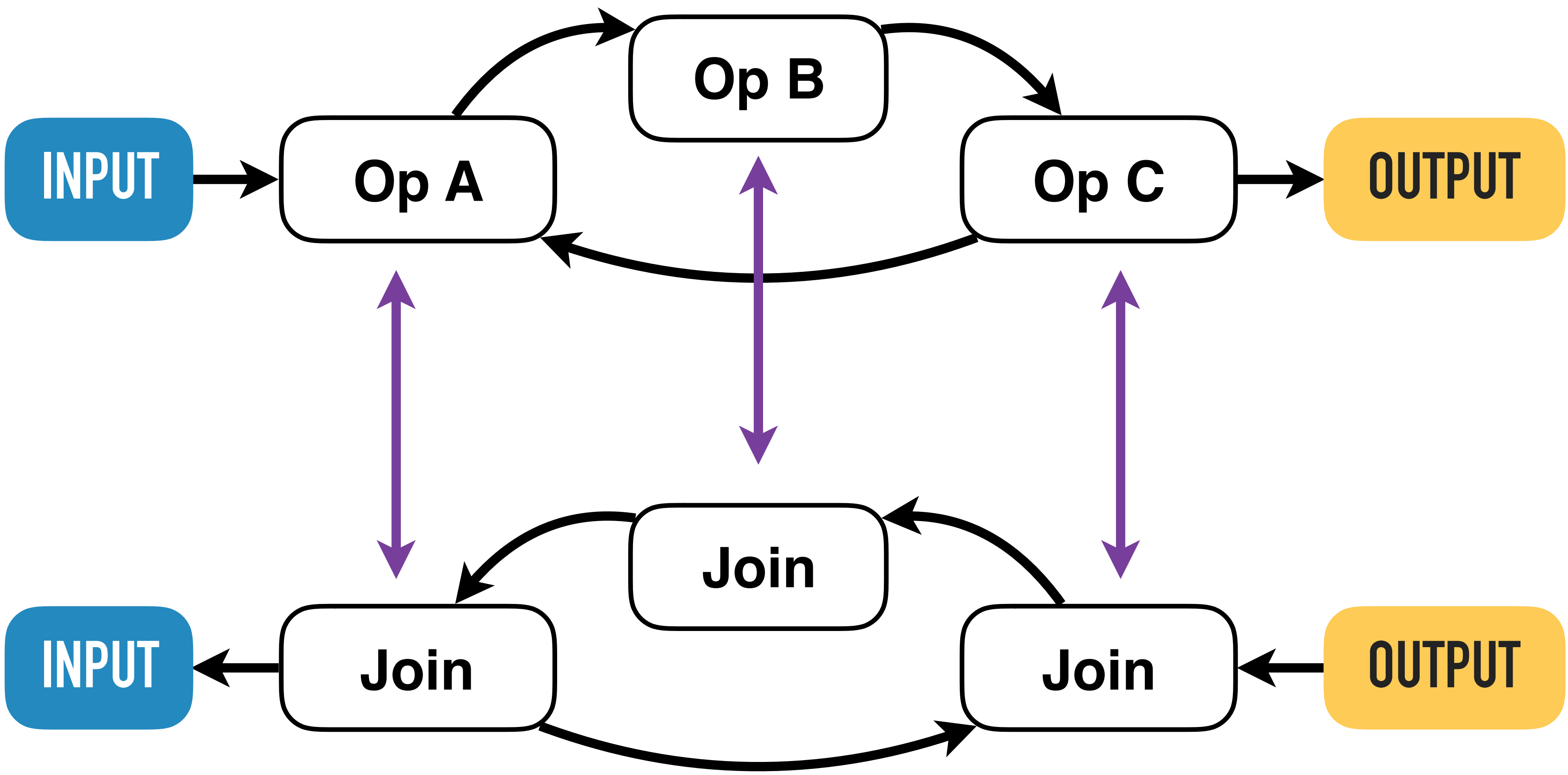
Original dataflow:



EXPLANATIONS WITH DIFFERENTIAL DATAFLOW

Original dataflow:

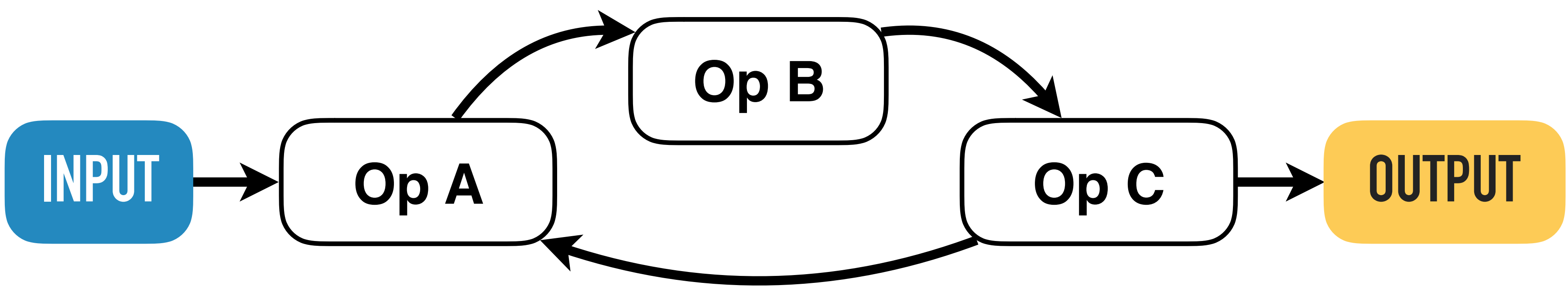
Explanation dataflow:



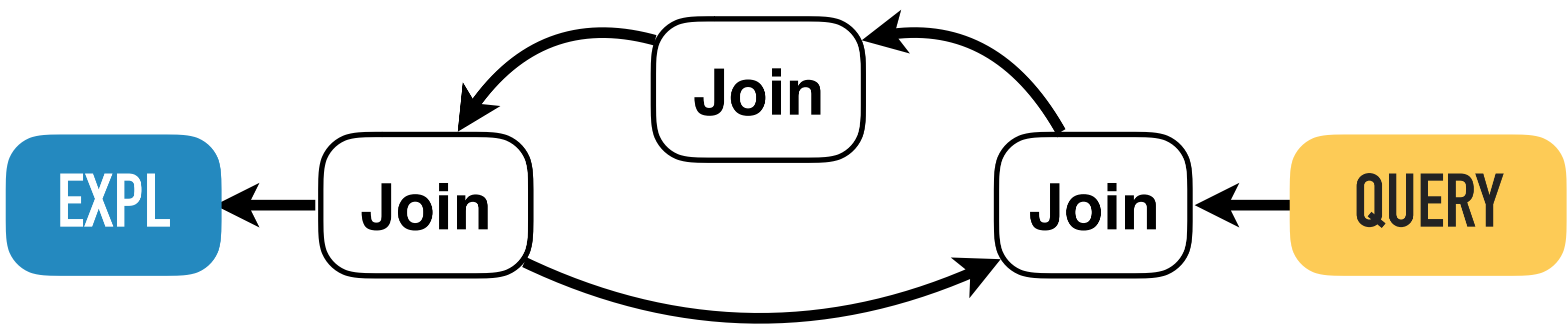
Augment the original dataflow with a shadow dataflow

ITERATIVE BACKWARD TRACING

Original dataflow:

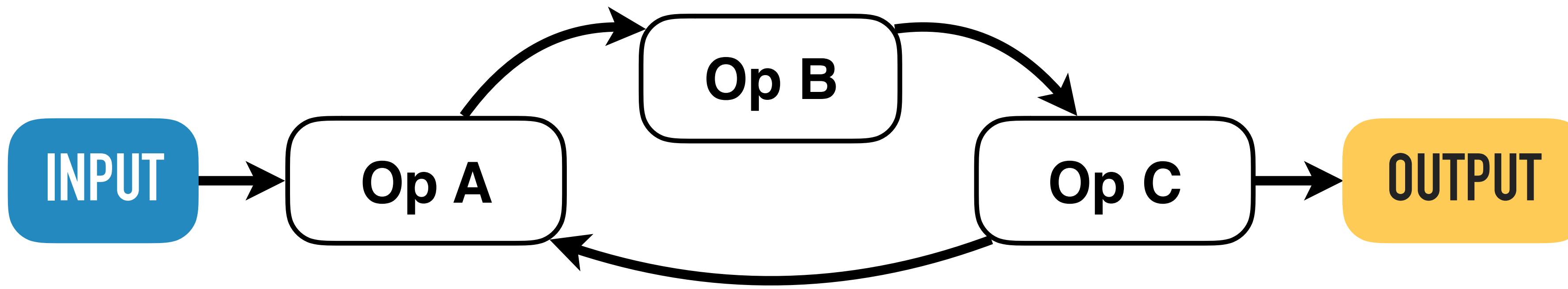


Explanation dataflow:

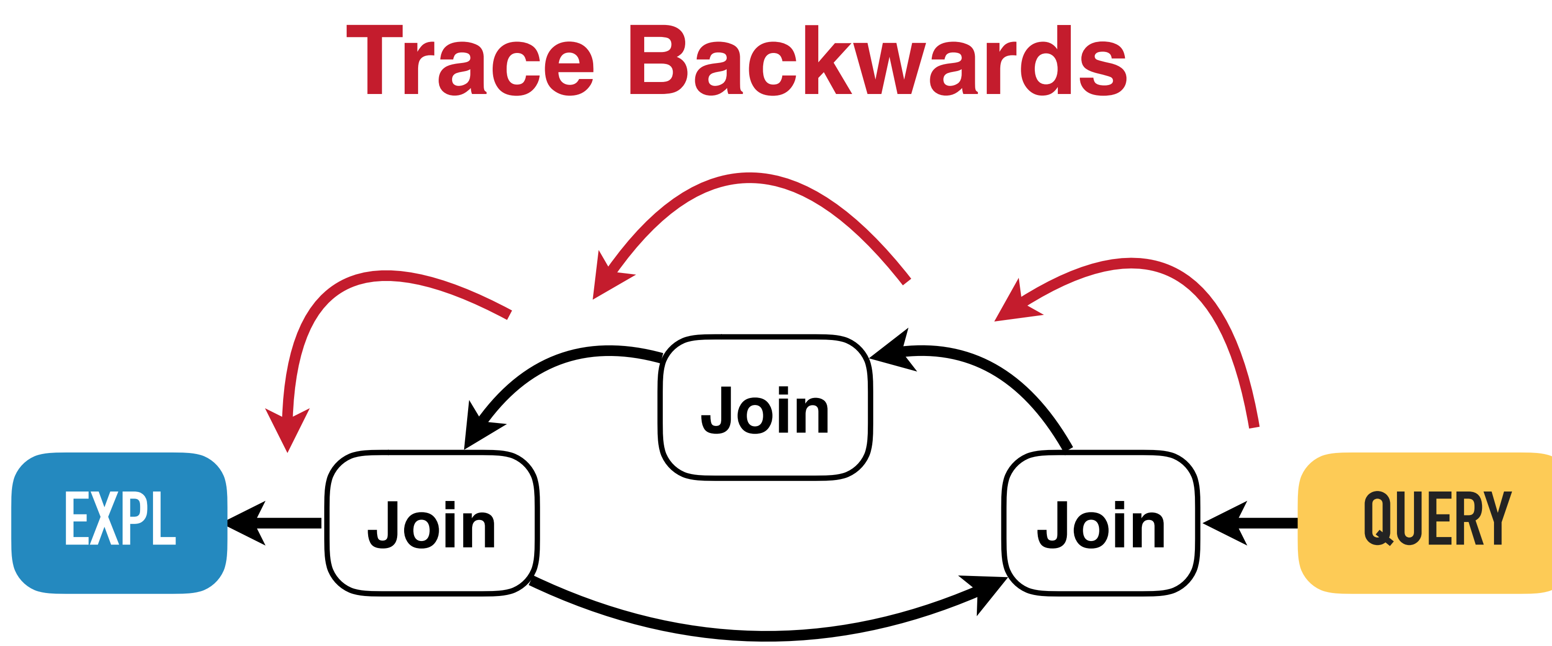


ITERATIVE BACKWARD TRACING

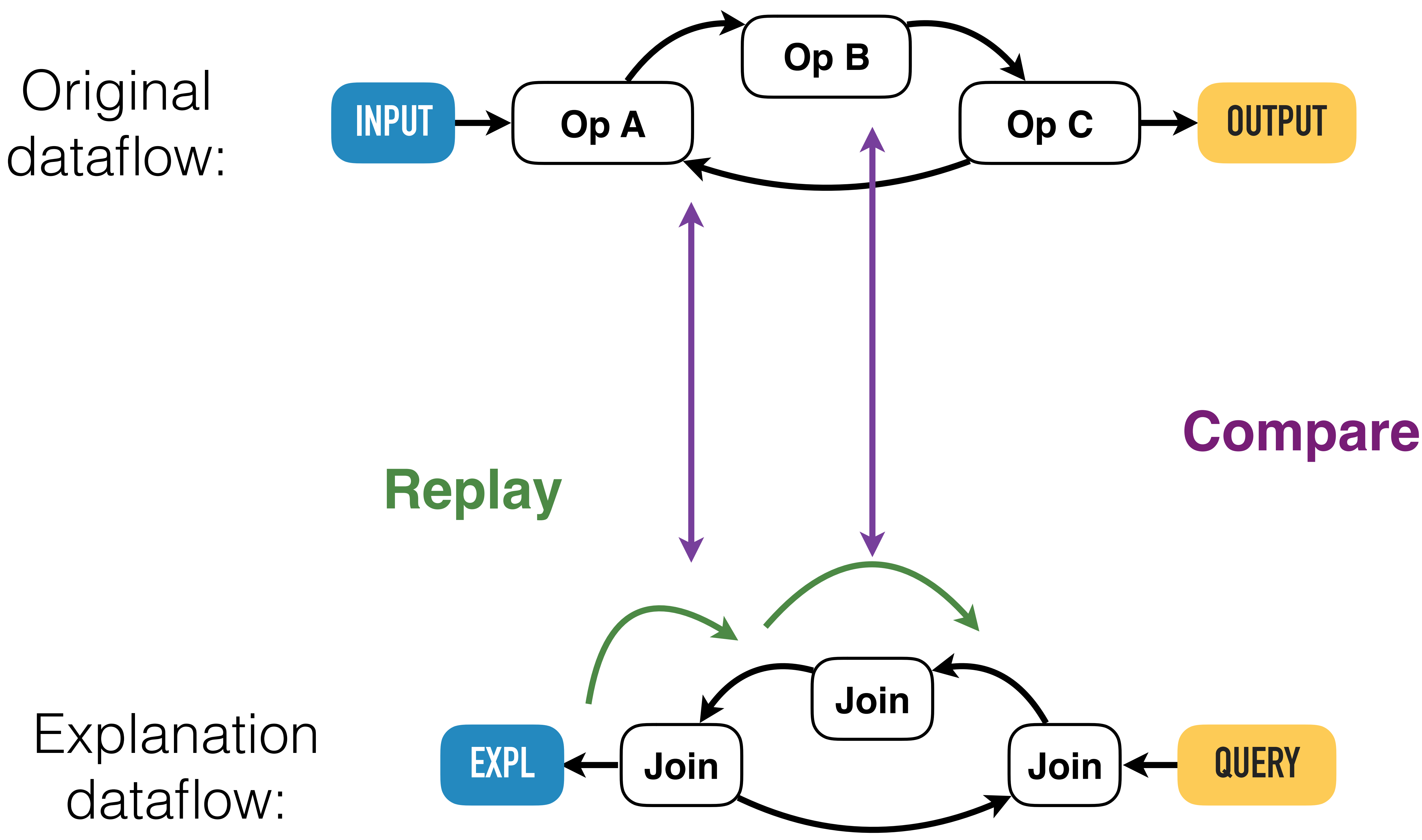
Original dataflow:



Explanation dataflow:



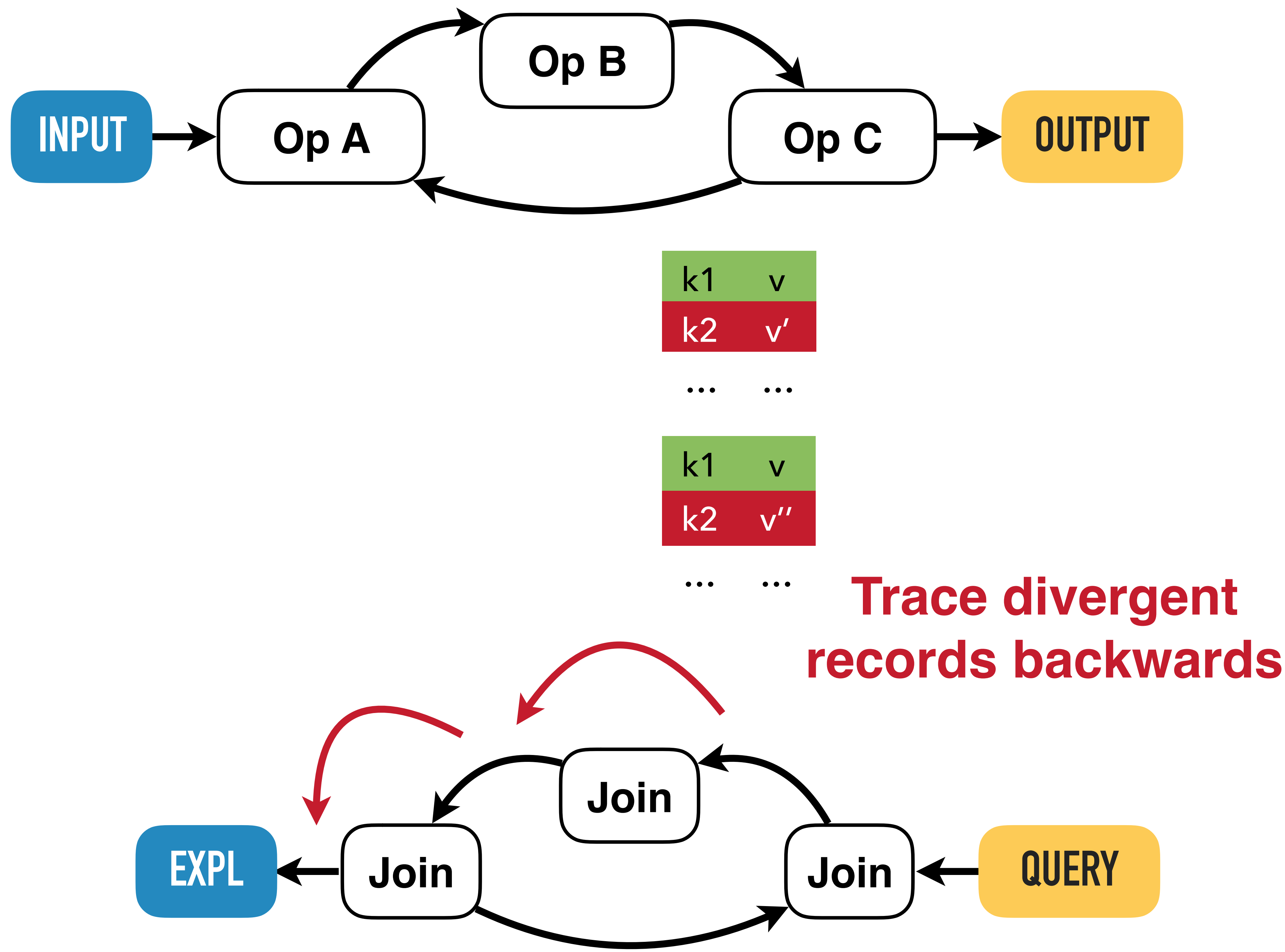
ITERATIVE BACKWARD TRACING



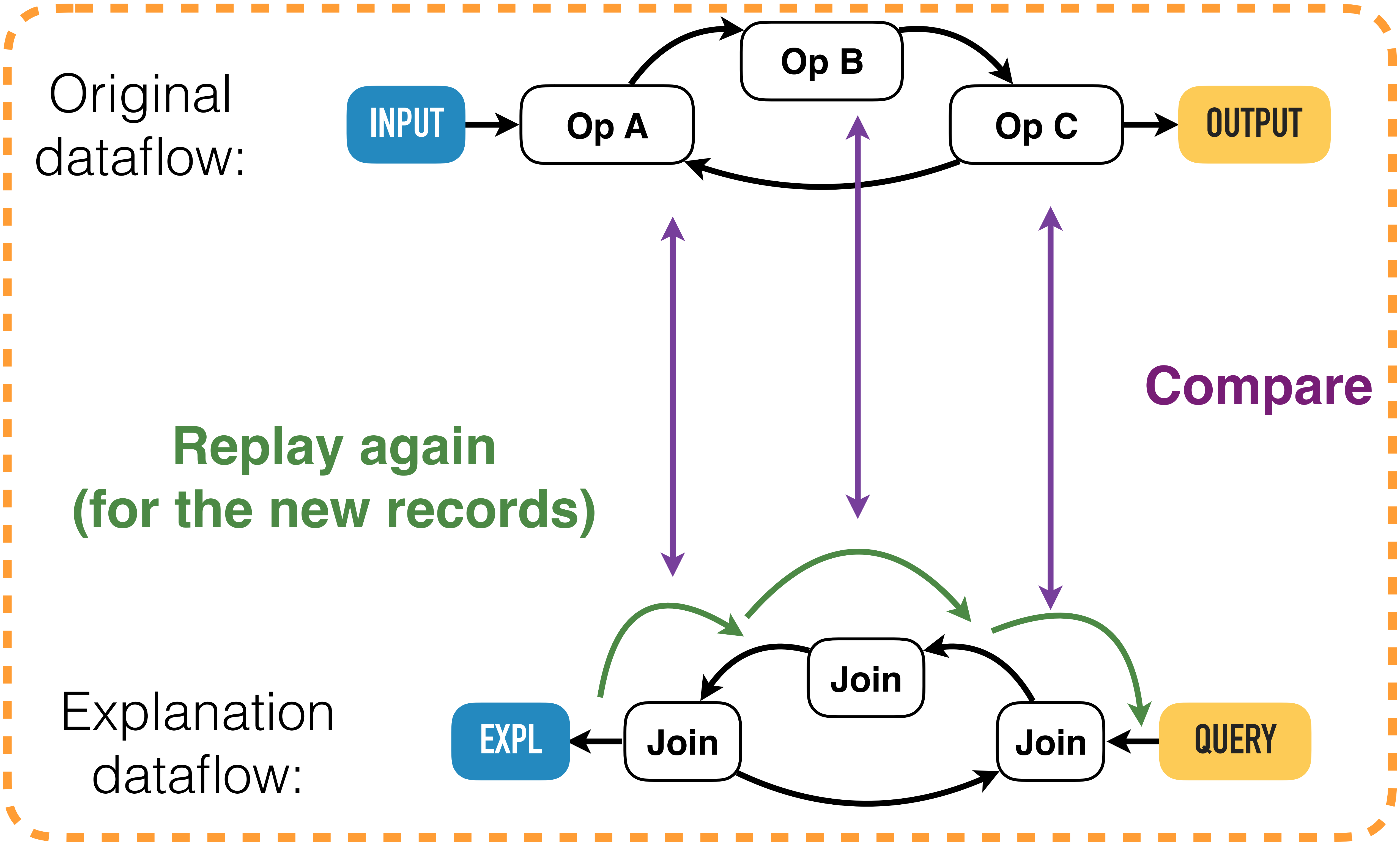
ITERATIVE BACKWARD TRACING

Original dataflow:

Explanation dataflow:



ITERATIVE BACKWARD TRACING



Repeat until a fix-point

EXAMPLE: EXPLAINING OUTPUTS OF WORD SET DIFFERENCE

A

THE QUICK BROWN FOX

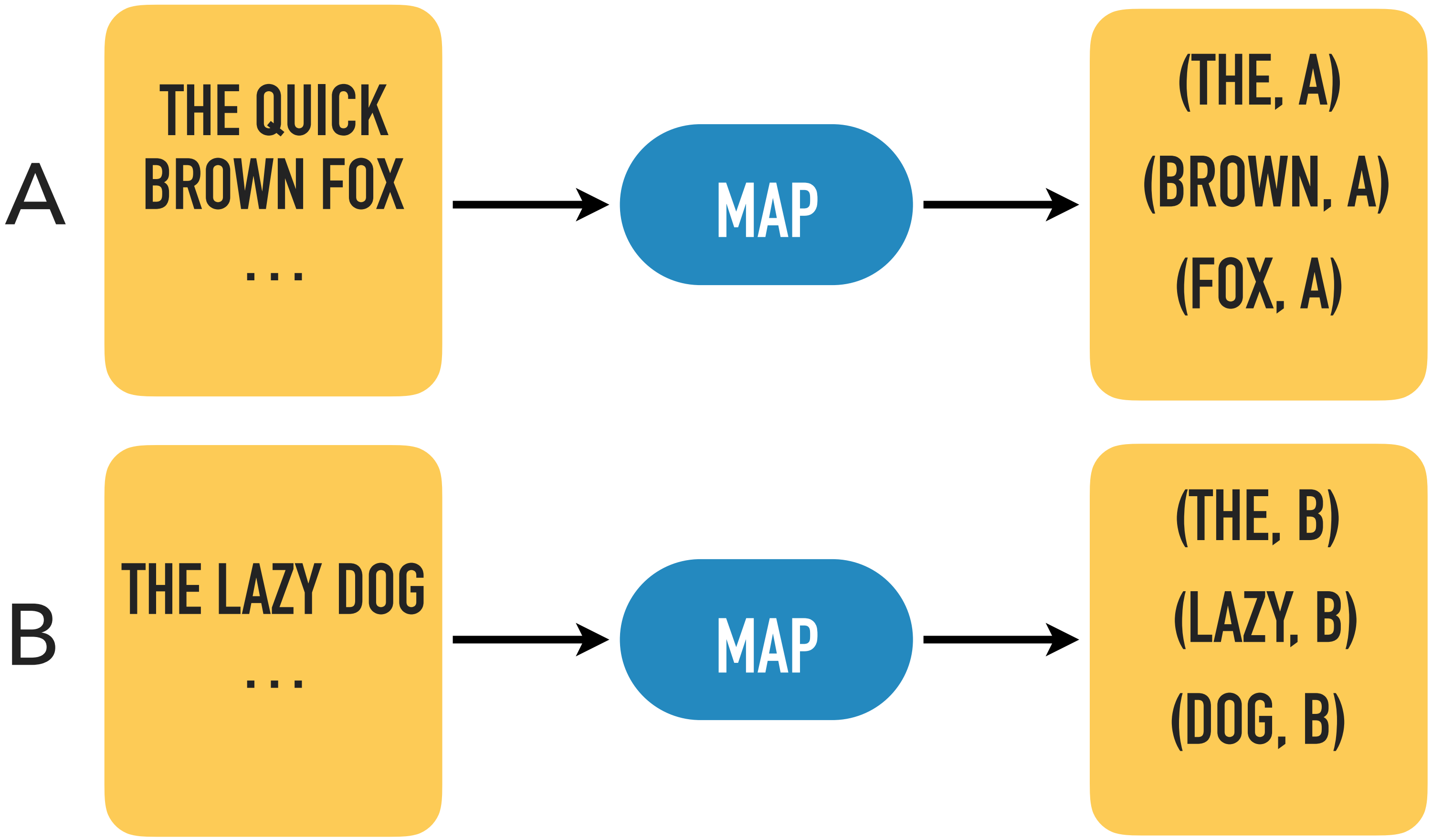
...

B

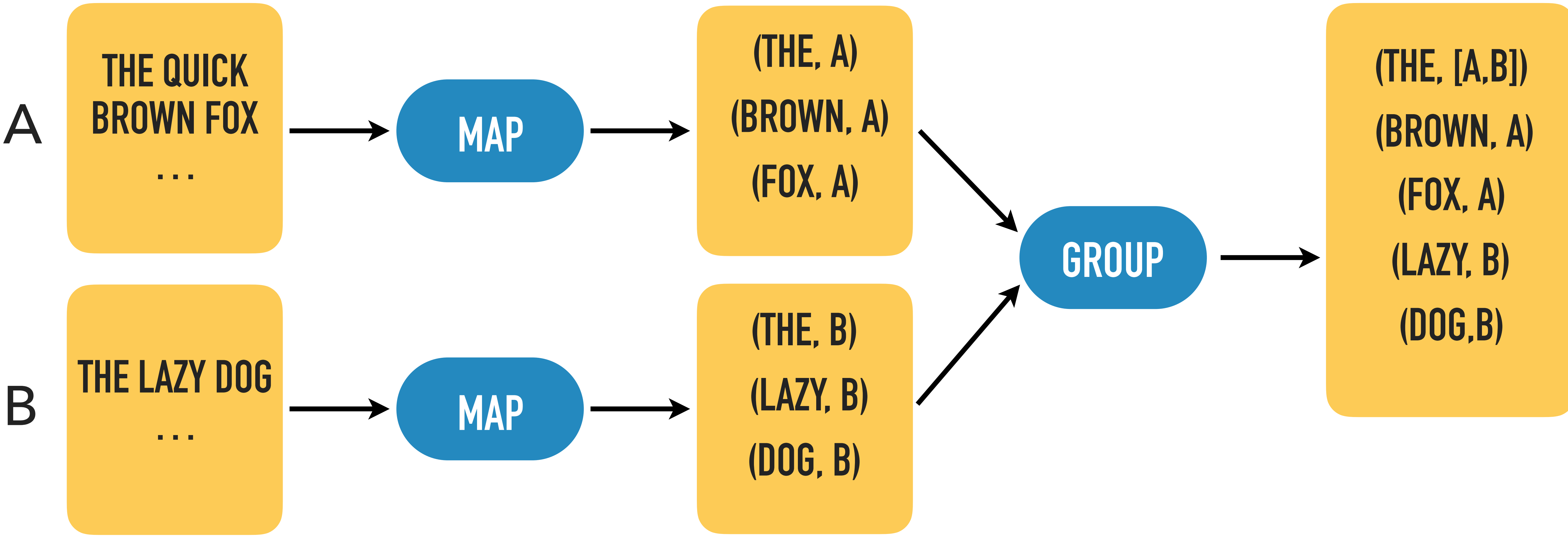
THE LAZY DOG

...

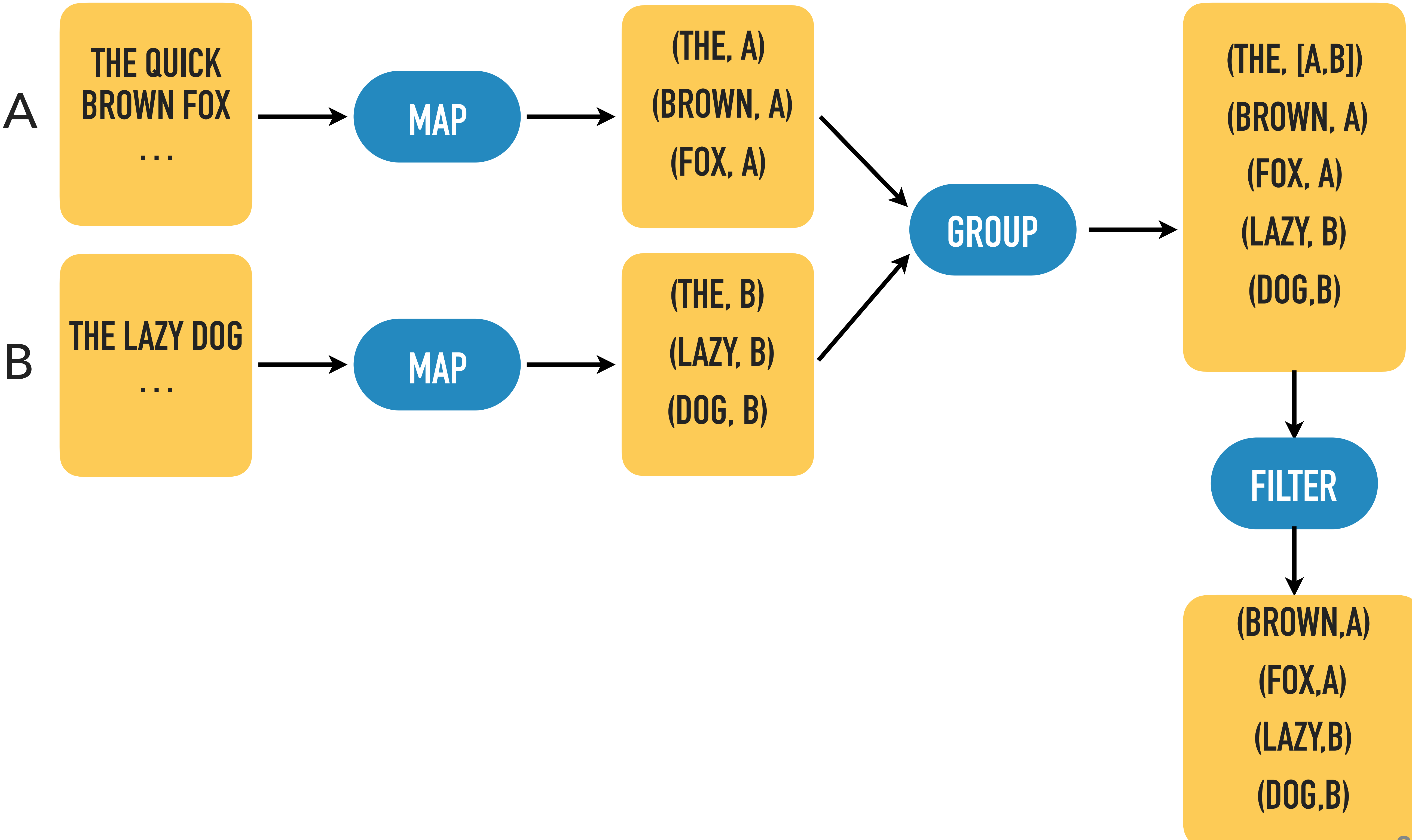
EXAMPLE: EXPLAINING OUTPUTS OF WORD SET DIFFERENCE



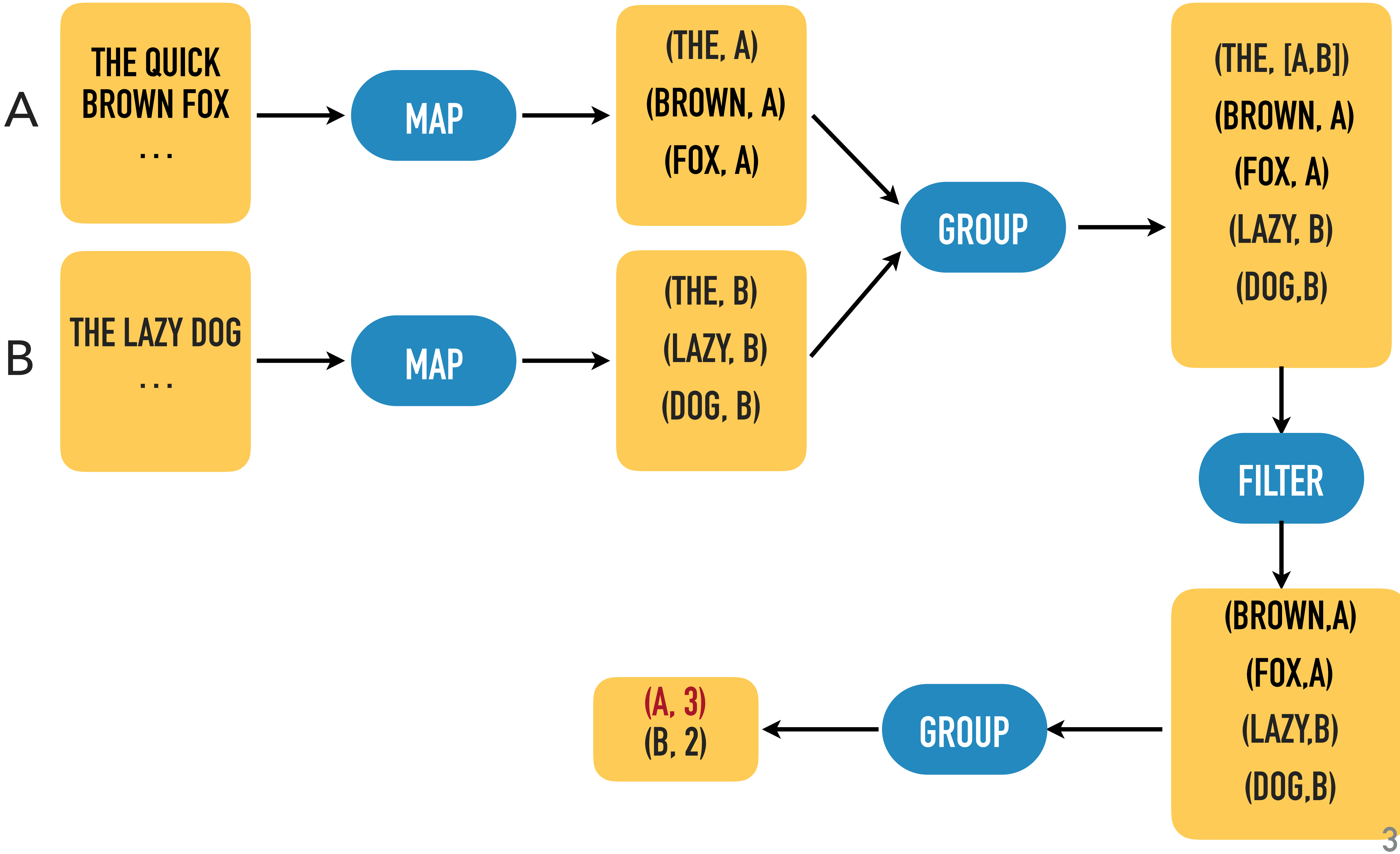
EXAMPLE: EXPLAINING OUTPUTS OF WORD SET DIFFERENCE



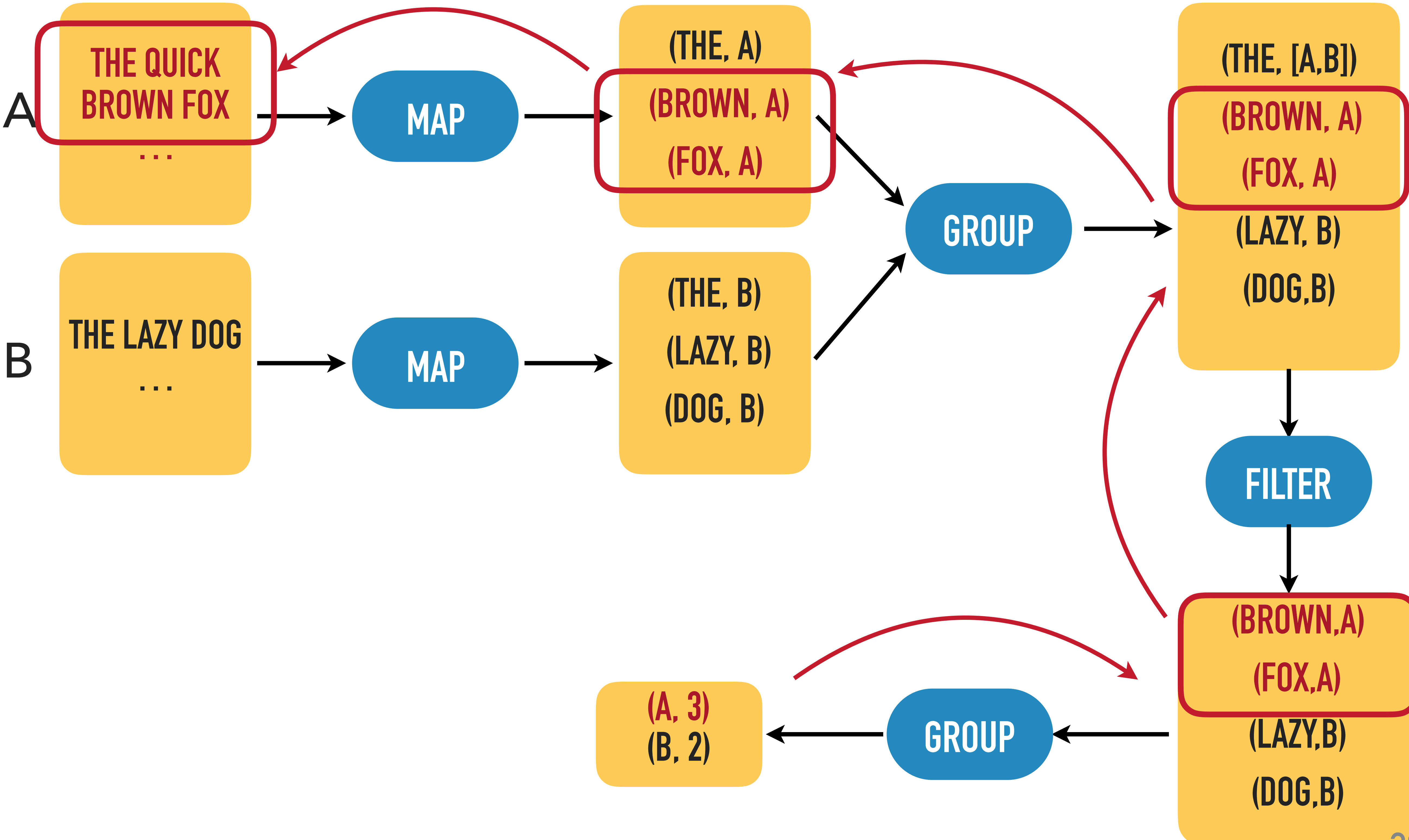
EXAMPLE: EXPLAINING OUTPUTS OF WORD SET DIFFERENCE



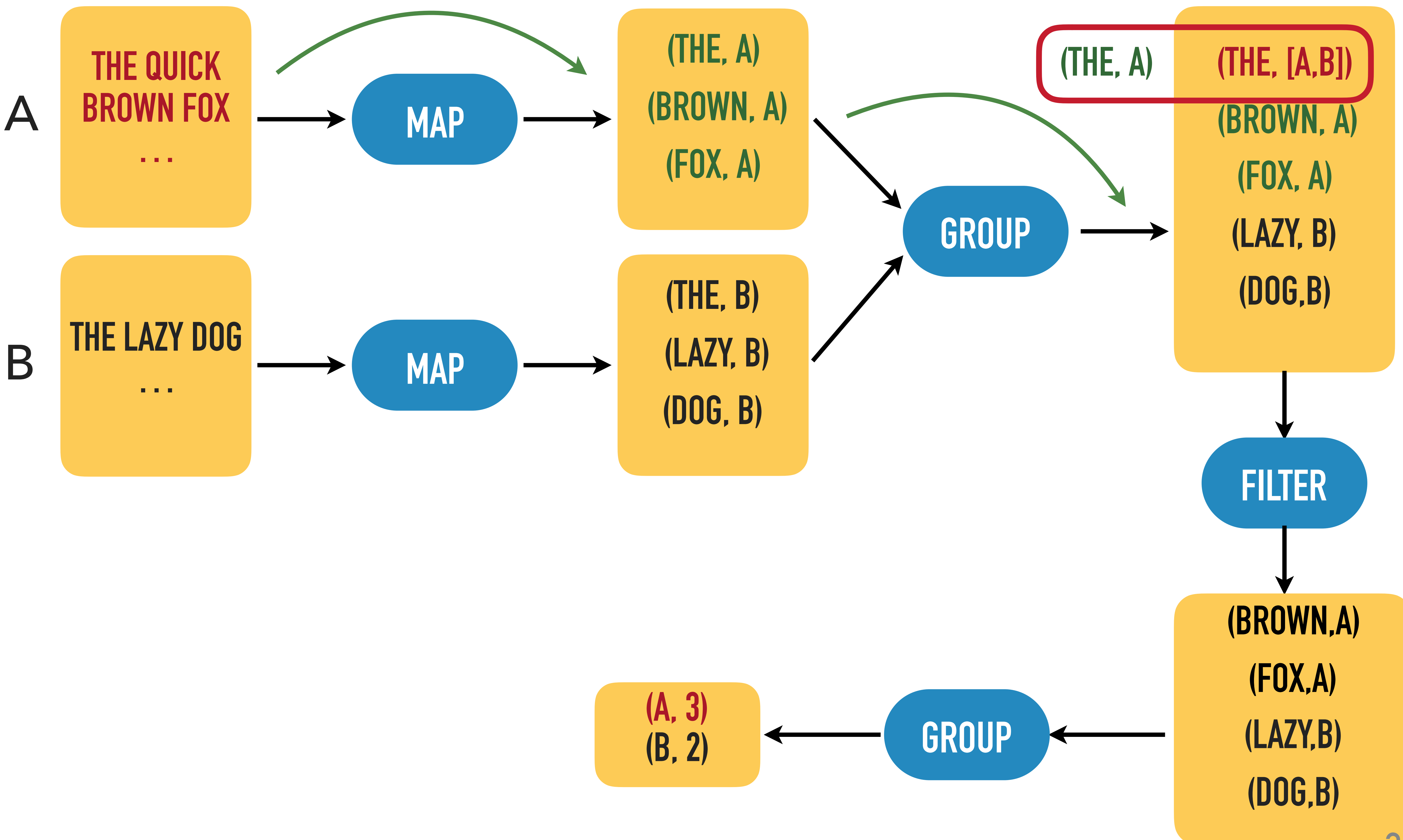
EXAMPLE: EXPLAINING OUTPUTS OF WORD SET DIFFERENCE



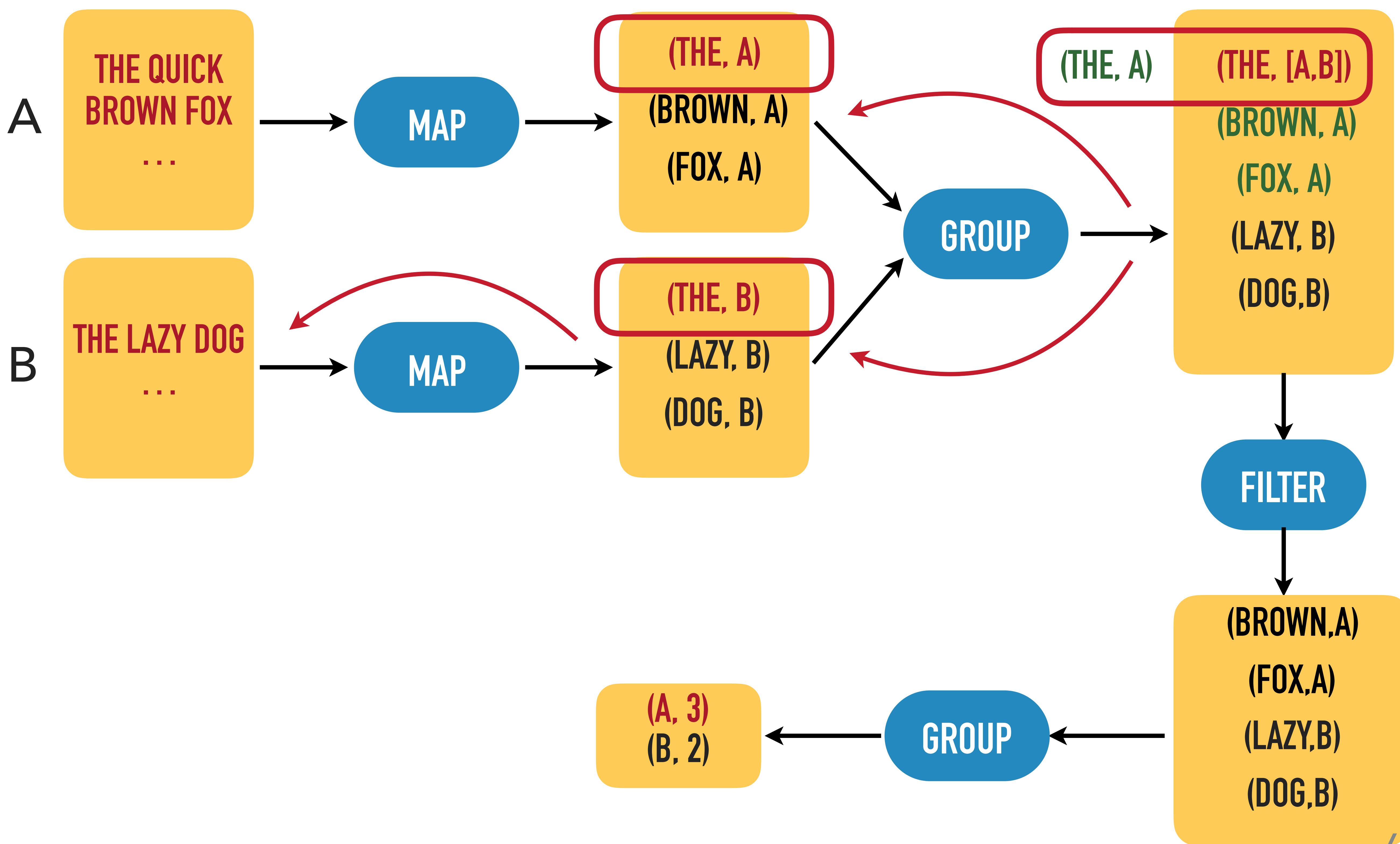
EXAMPLE: EXPLAINING OUTPUTS OF WORD SET DIFFERENCE



EXAMPLE: EXPLAINING OUTPUTS OF WORD SET DIFFERENCE



EXAMPLE: EXPLAINING OUTPUTS OF WORD SET DIFFERENCE



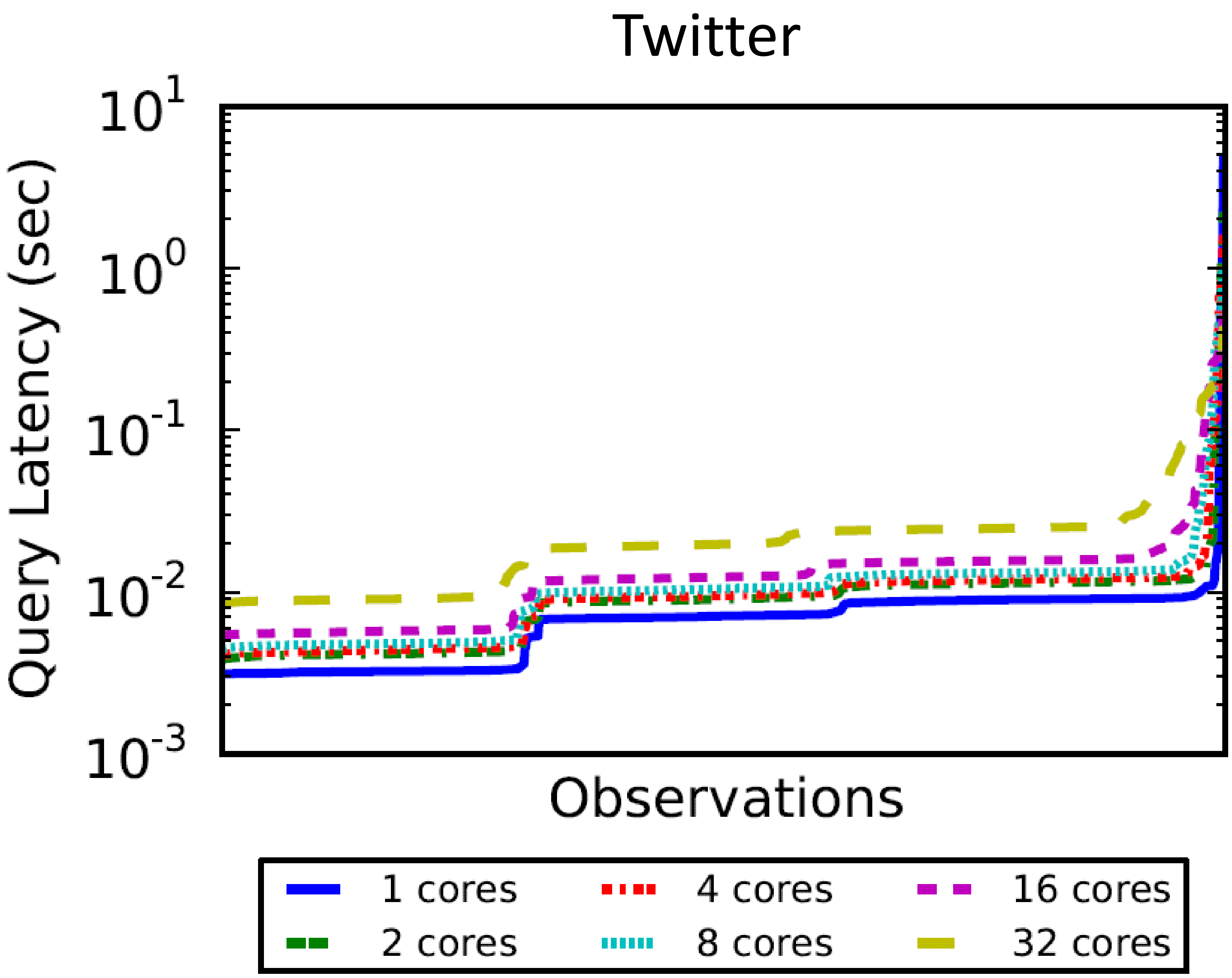
RESULTS: EXPLAINING CONNECTED COMPONENTS

- Dataset: A subset of the Twitter graph with 1B edges
- Algorithm: Label propagation
- Output: Records of the form (A,B) denoting that nodes A and B belong to the same connected component
- System used: Differential Dataflow
- Machine used: Intel Xeon E5-4640 at 2.4GHz with 32 cores and 500G RAM

More results:

Z. Chothia, J. Liagouris, F. McSherry, T. Roscoe *Explaining Outputs in Modern Data Analytics* PVDLB 9(12):1137-1148, 2016.

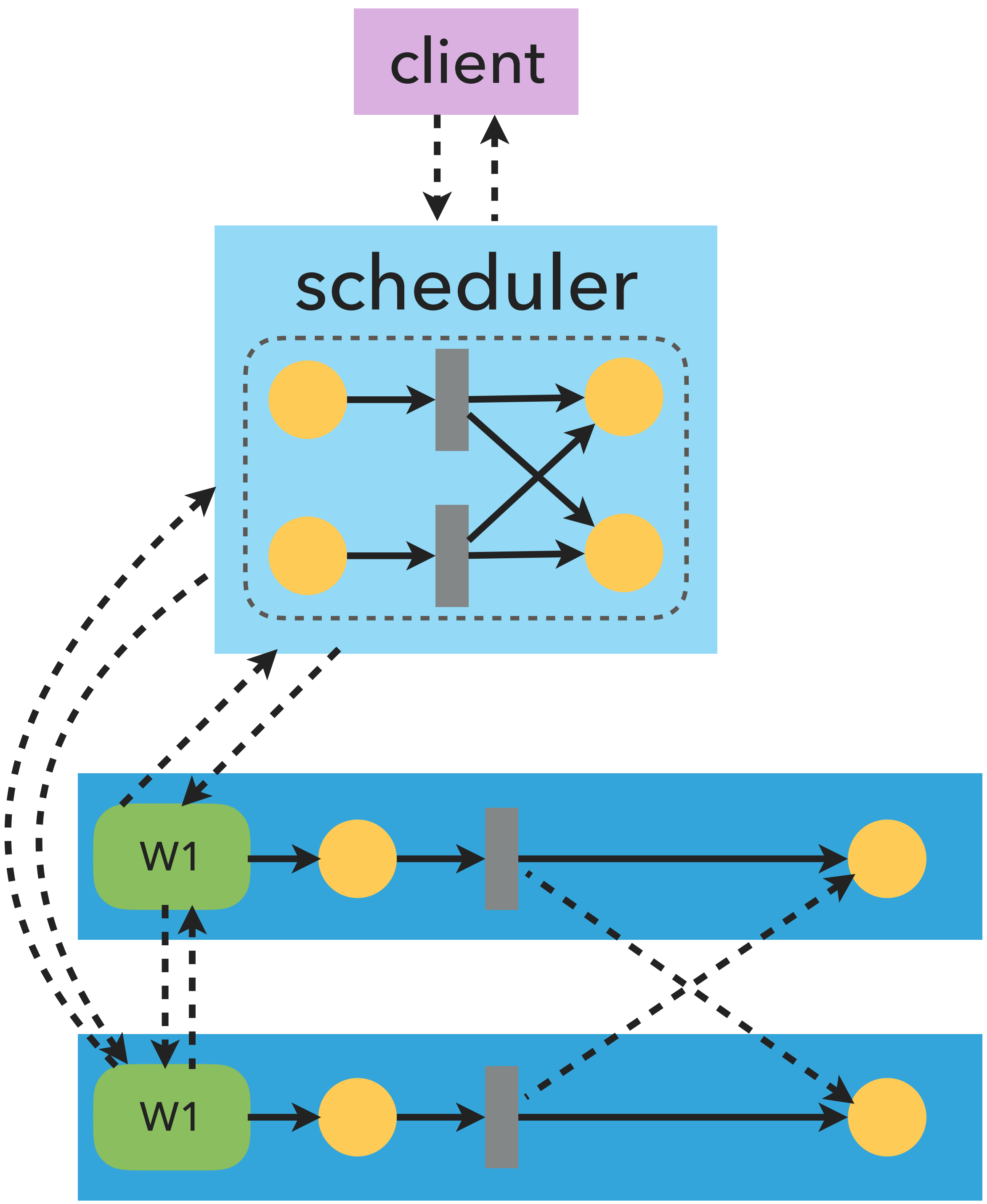
EXPLAINING CONNECTED COMPONENTS



PART II

Why is my distributed dataflow slow?

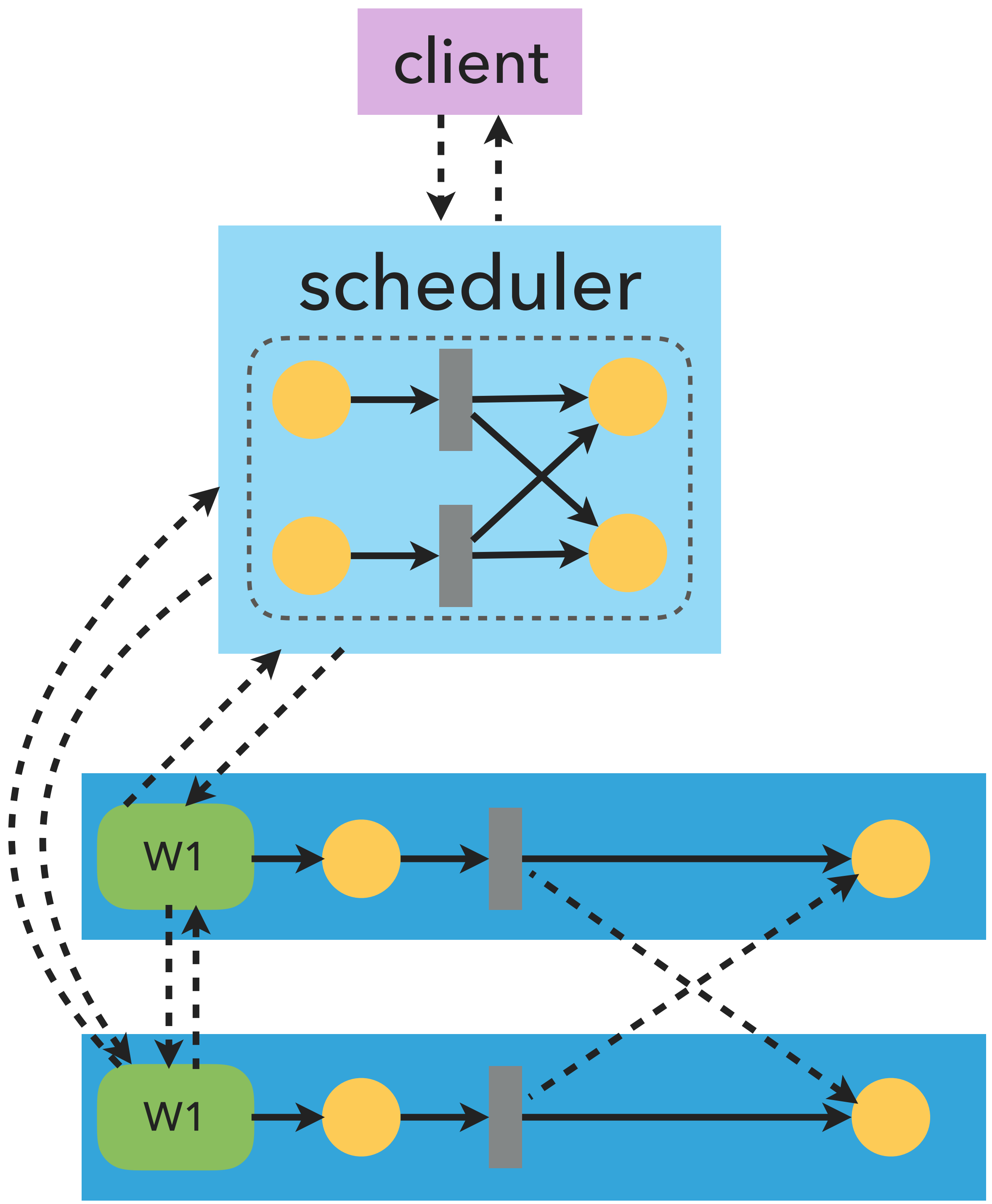
DISTRIBUTED DATAFLOWS



Apache Flink

Naiad

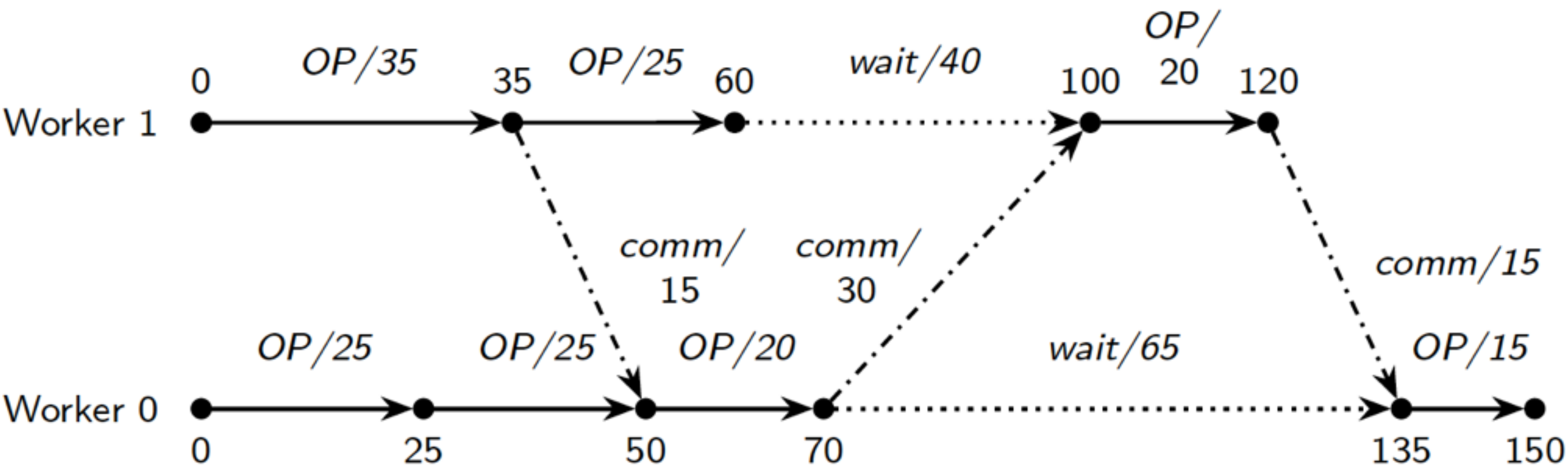
CHALLENGE: TROUBLESHOOTING IS HARD



- many processes and activities
- the cause is usually not isolated but spans multiple processes

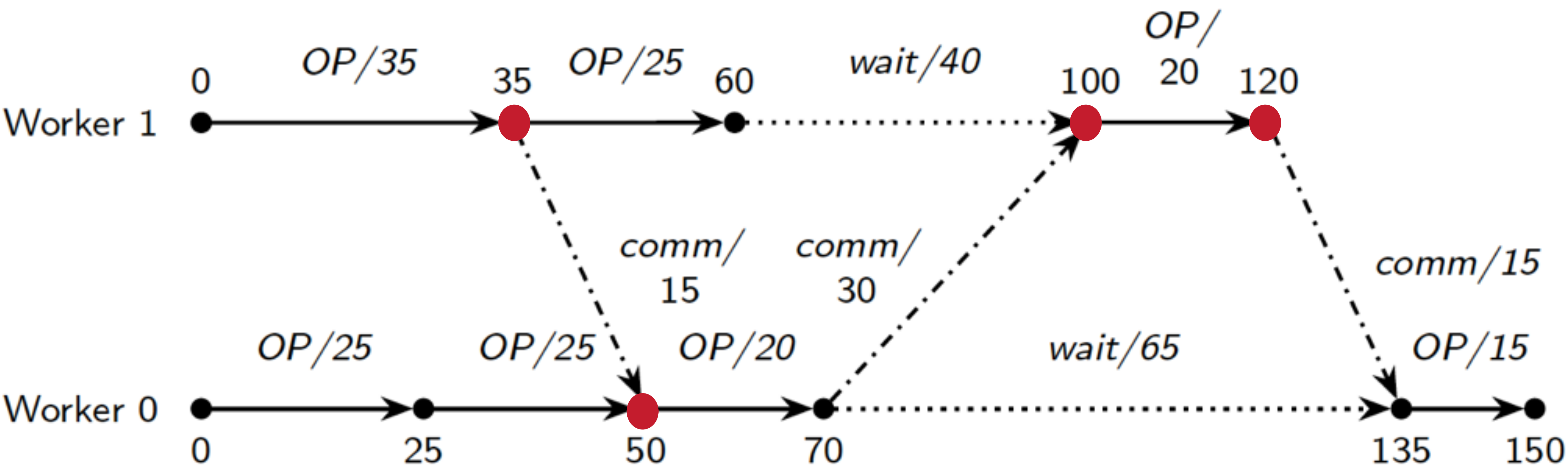
PROFILING: THE PROGRAM ACTIVITY GRAPH (PAG)

- Models Happened-Before relationships



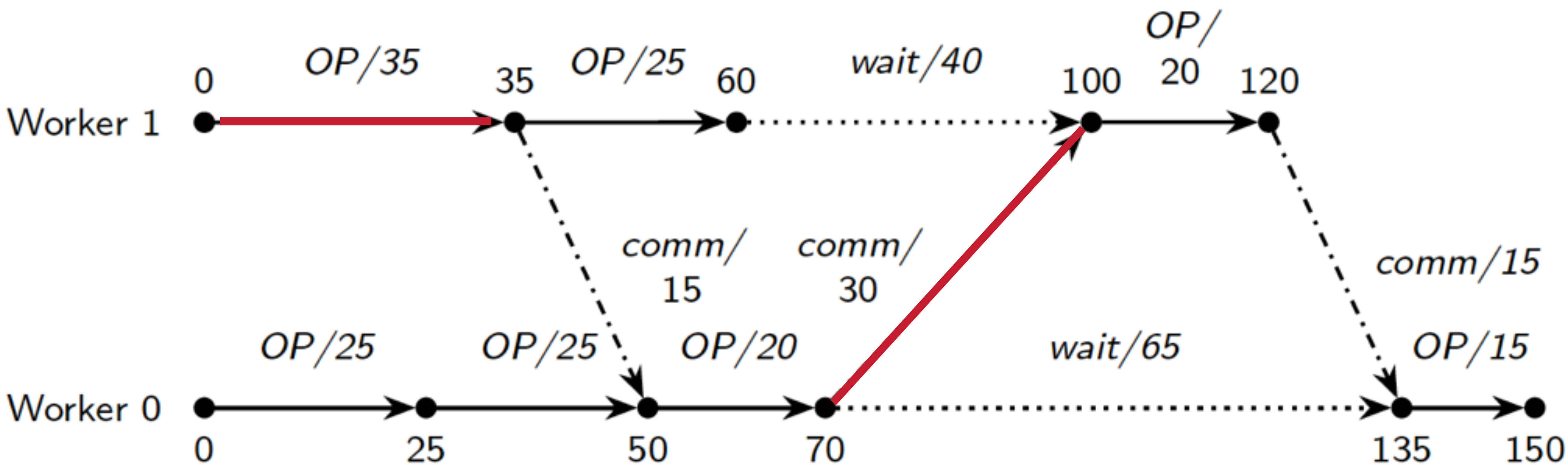
PROFILING: THE PROGRAM ACTIVITY GRAPH (PAG)

- Vertices: events with timestamps



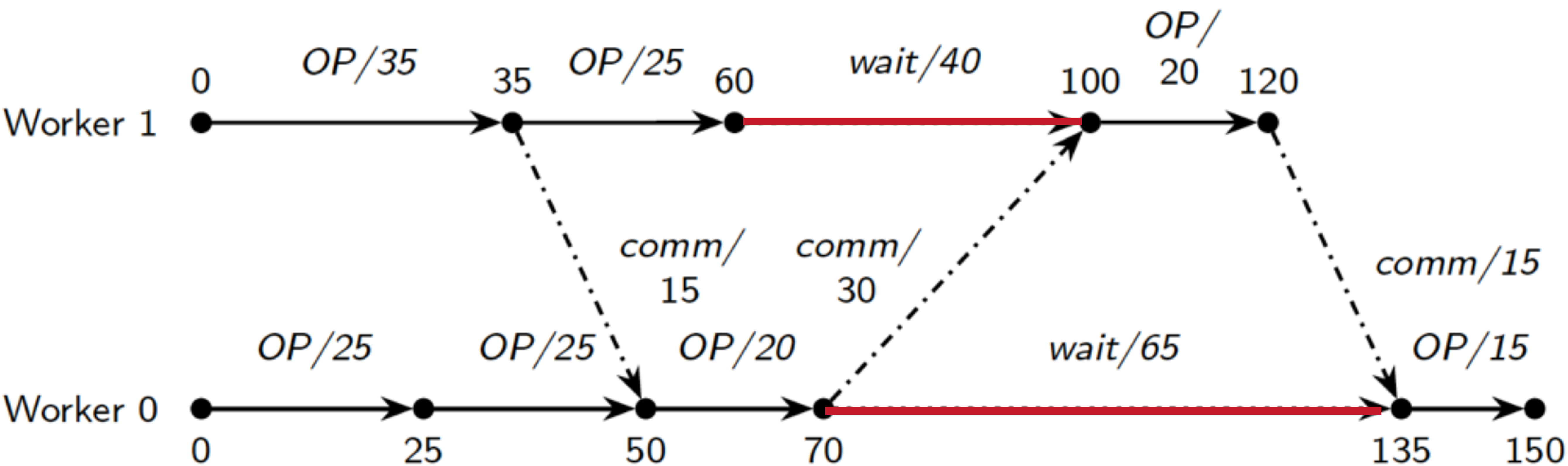
PROFILING: THE PROGRAM ACTIVITY GRAPH (PAG)

- Edges: duration of activities



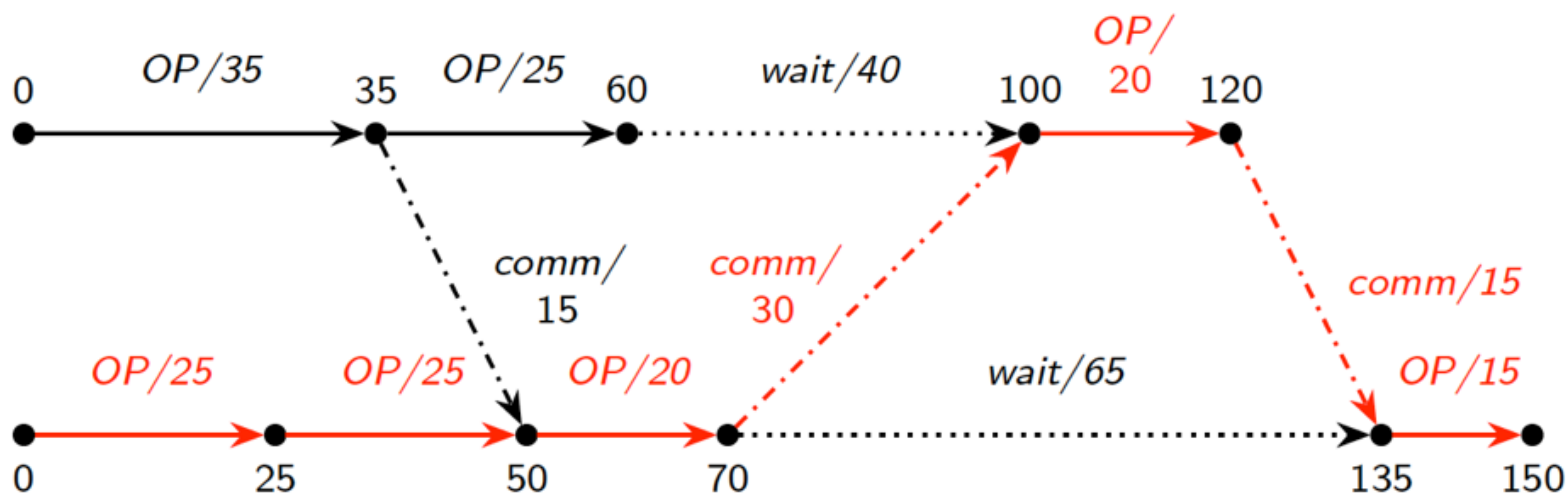
PROFILING: THE PROGRAM ACTIVITY GRAPH (PAG)

- Wait edges: time spent in waiting for a message



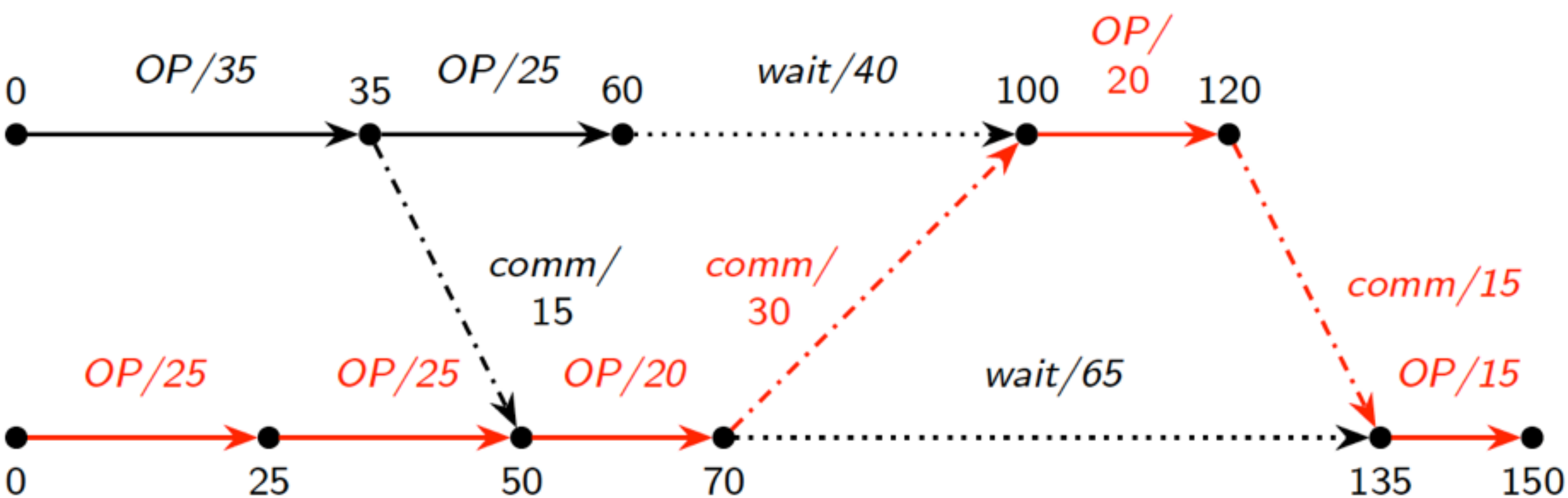
CRITICAL PATH ANALYSIS

The critical path is the path of non-waiting activities in the execution history of the program with the longest duration



CRITICAL PATH ANALYSIS

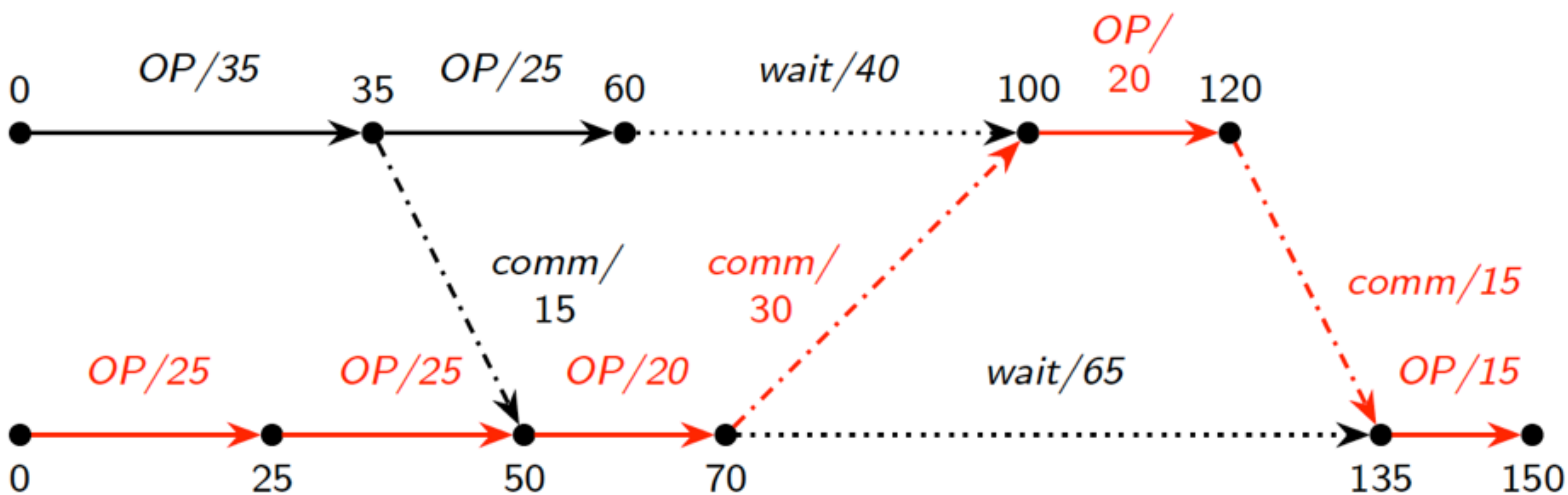
The program activity graph is a DAG so the critical path computation is tractable



CRITICAL PATH ANALYSIS

The critical path is constructed by starting from the last event and backtracking:

- Following the edges with the longest duration
- Avoiding waiting edges



How can we compute the critical path in long-running, dynamic distributed applications, with possibly unbounded input?

- There may be no "job end"
- The PAG is evolving while the job is running
- Stale profiling information is not useful

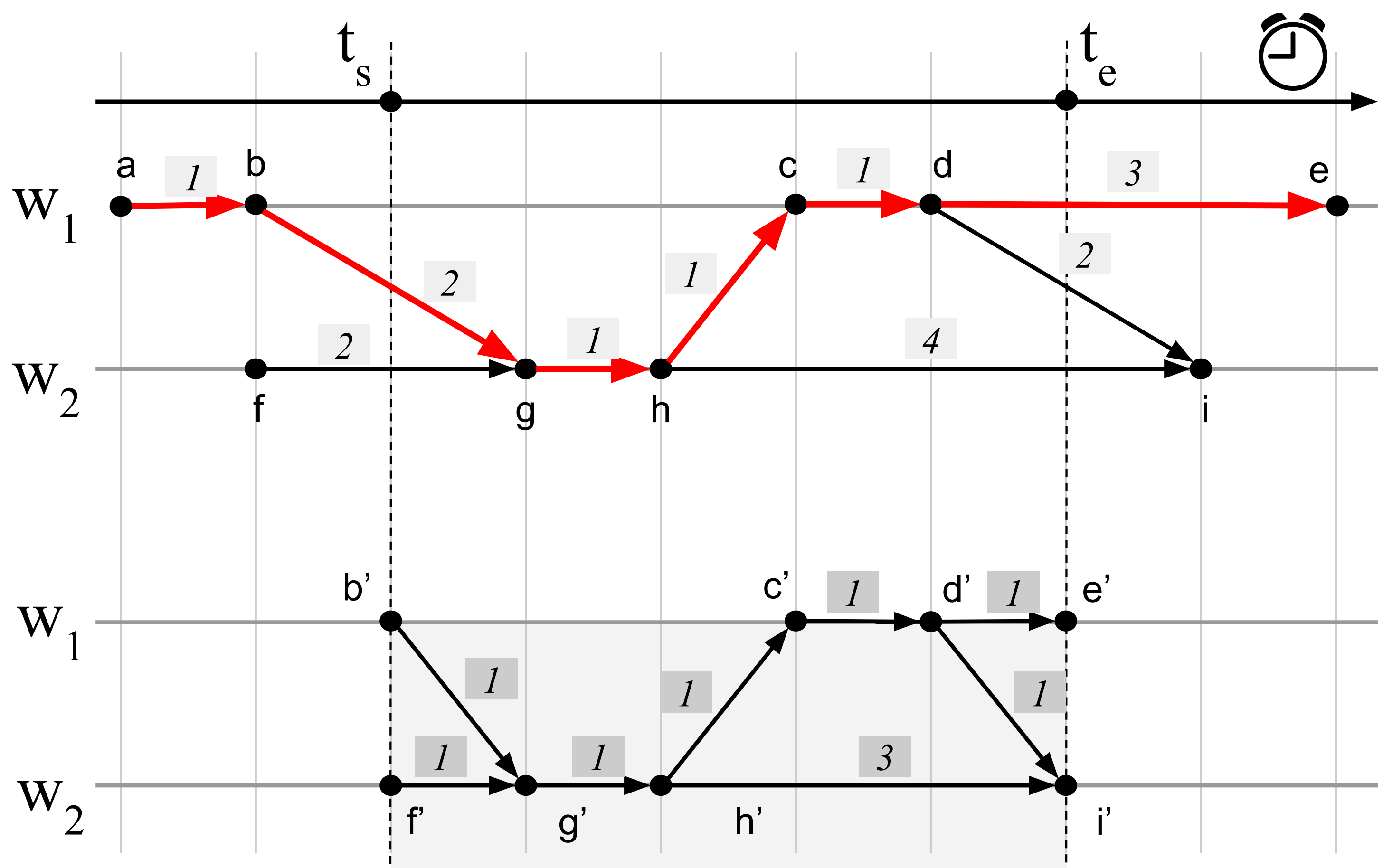
TRANSIENT CRITICAL PATHS (TCPs)

An adaptation of the standard critical path on trace *snapshots*

- tumbling, sliding or custom windows

Input Trace

Snapshot in [ts,te]

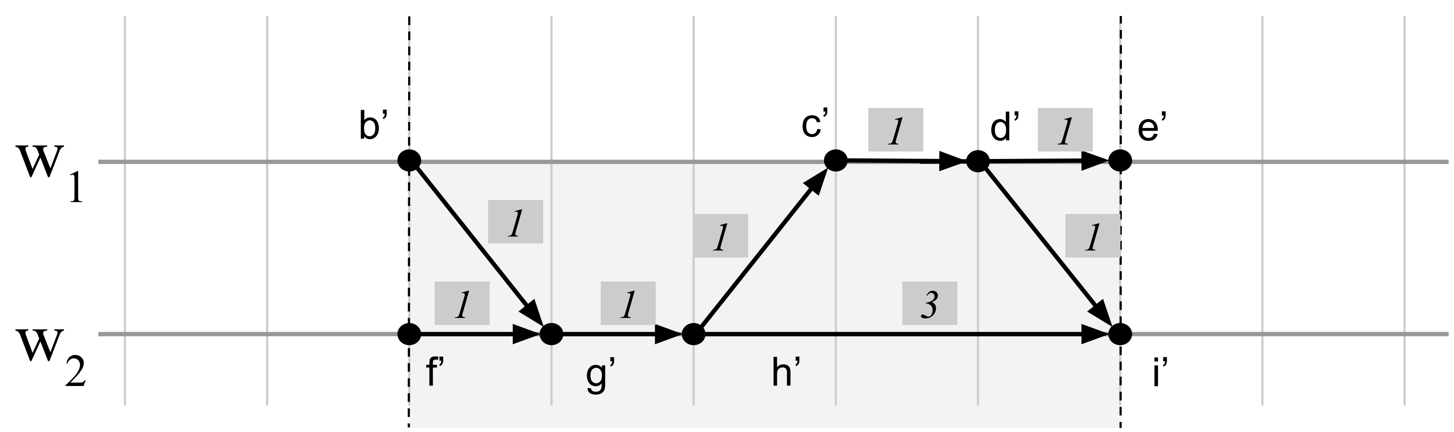


TRANSIENT CRITICAL PATHS (TCPs)

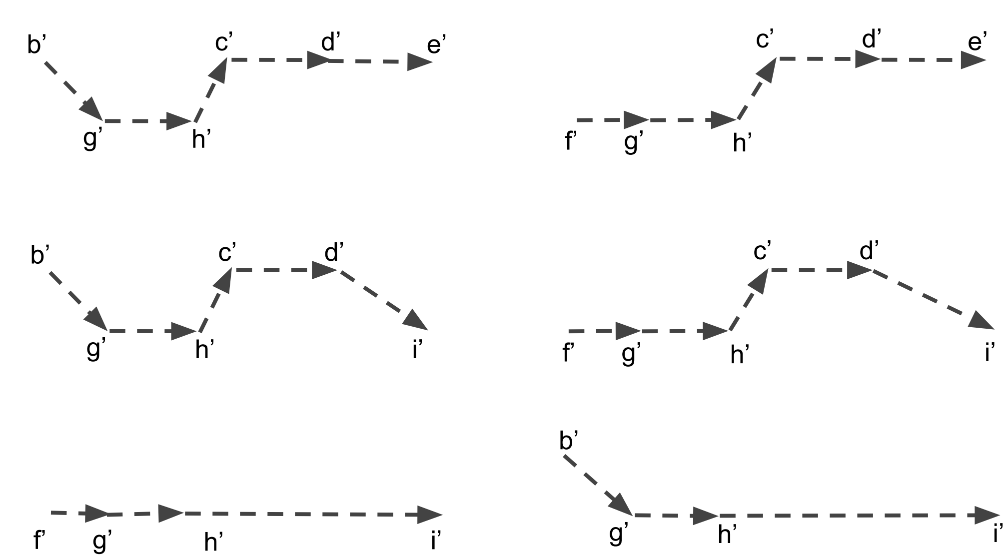
Multiple transient critical paths per snapshot

- All TCPs are possible parts of the unknown global critical path

Snapshot in [t_s,t_e]

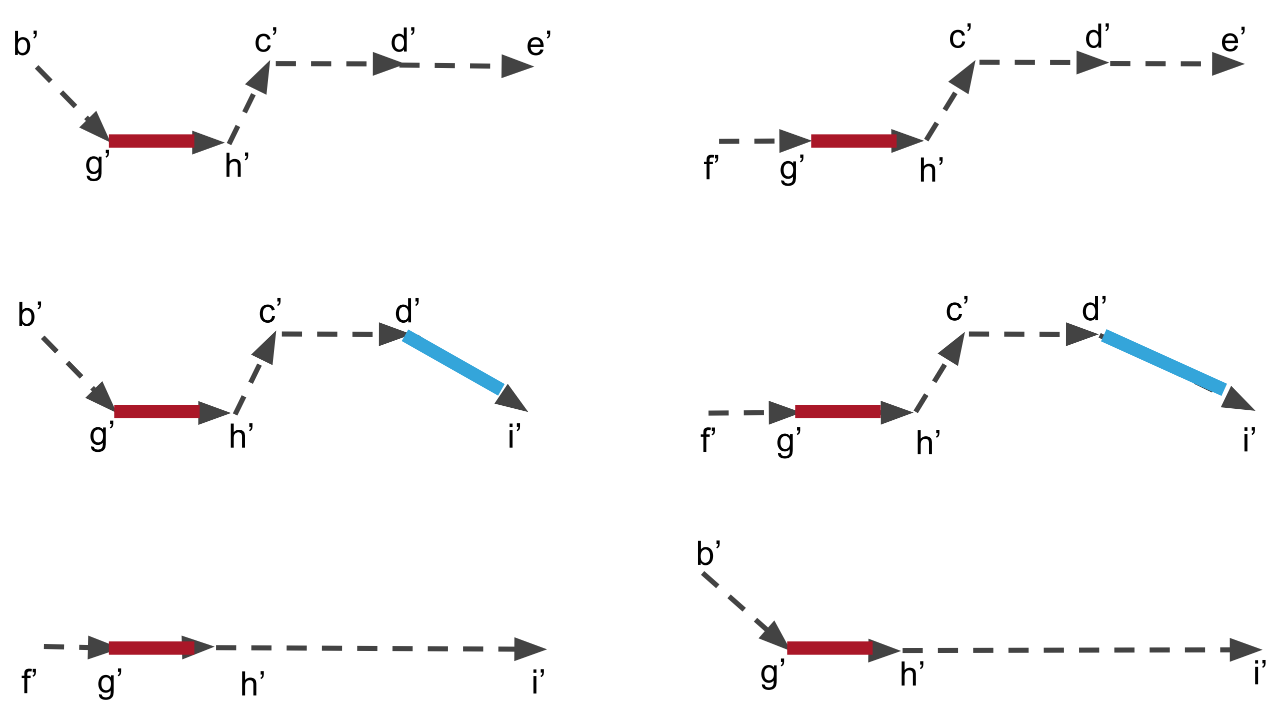


Transient Critical Paths in the snapshot [t_s,t_e]



TRANSIENT PATH CENTRALITY (TPC)

The number of transient critical paths an edge belongs to

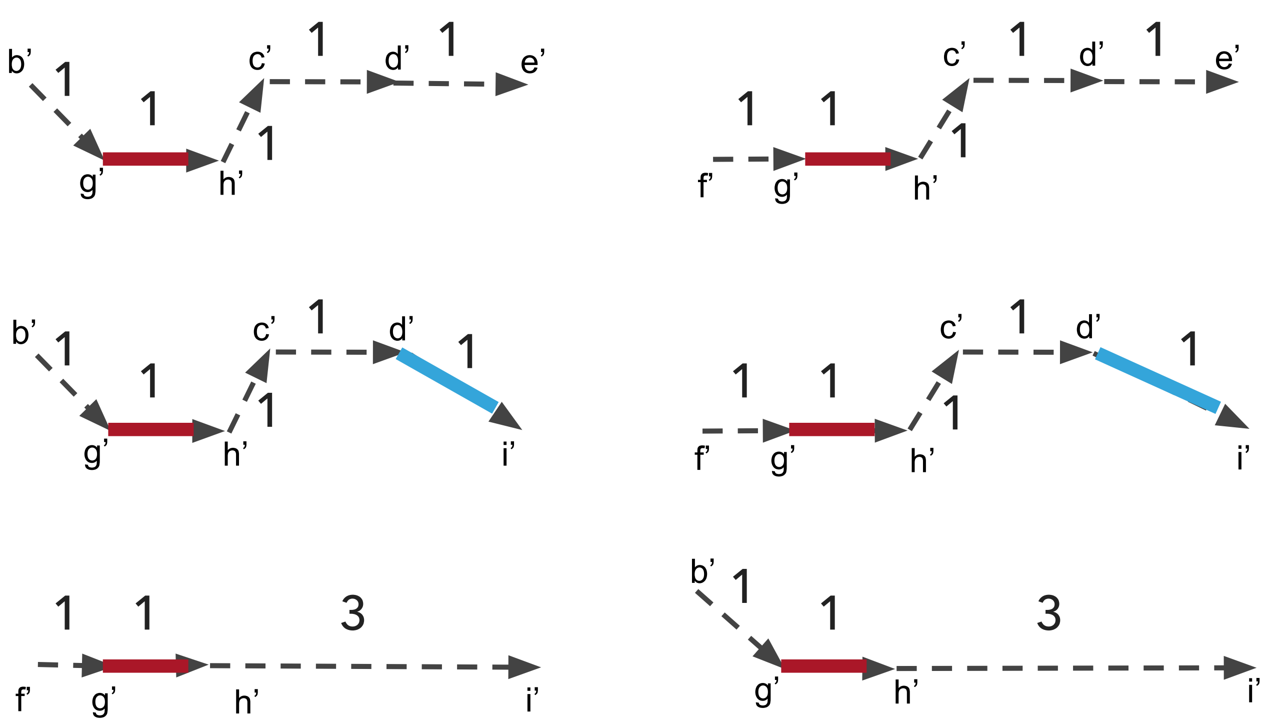


TPC(d’,i’) = 2

TPC(g’,h’) = 6

AVERAGE CRITICAL PARTICIPATION (CP)

An estimation of the activity's participation in the critical path



CP(g',h') = 6*1/6*5 = 0.2

CP(d',i') = 2*1/6*5 = 0.066

edge weight

$$CP_a = \frac{TPC(a) \cdot a_w}{N(t_e - t_s)}$$

number of transient critical paths

TRANSIENT CRITICAL PATHS ARE WIDELY APPLICABLE

"RDDs"	"DataStreams"	"Spouts and Bolts"	"Tensors"	Naiad

- data transformation
- data exchange
- control messages
- I/O
- data (de)-serialization
- buffer management
- scheduling

common set of
low-level primitives!

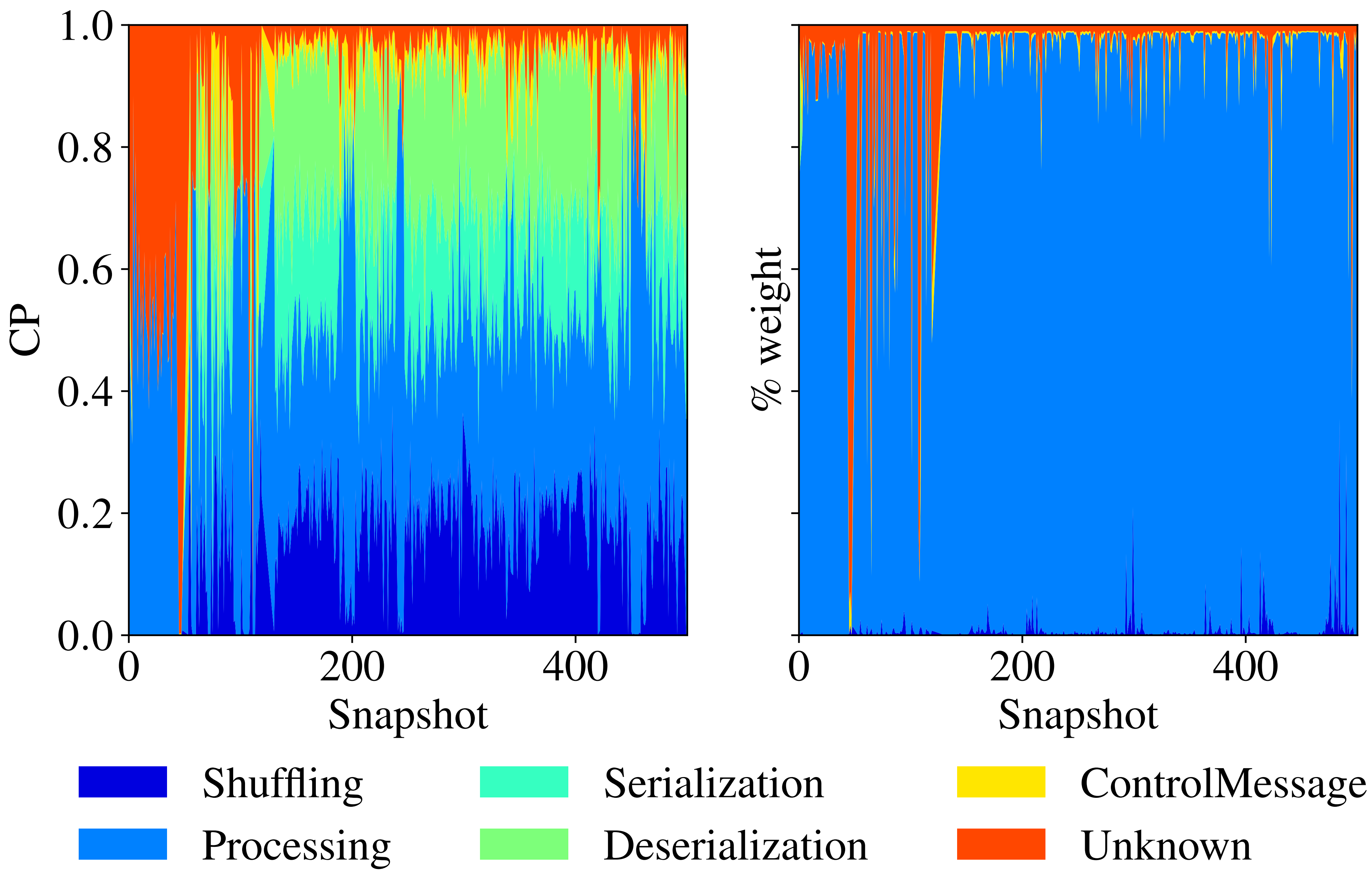
RESULTS: COMPARISON WITH CONVENTIONAL PROFILING

- Benchmark: TPC-DS [1]
- System under study: Spark (1.2.1)
- Setting: 20 machines with 8 workers each
- We actually used Spark logs from [2]
- Snapshot interval: 10 sec

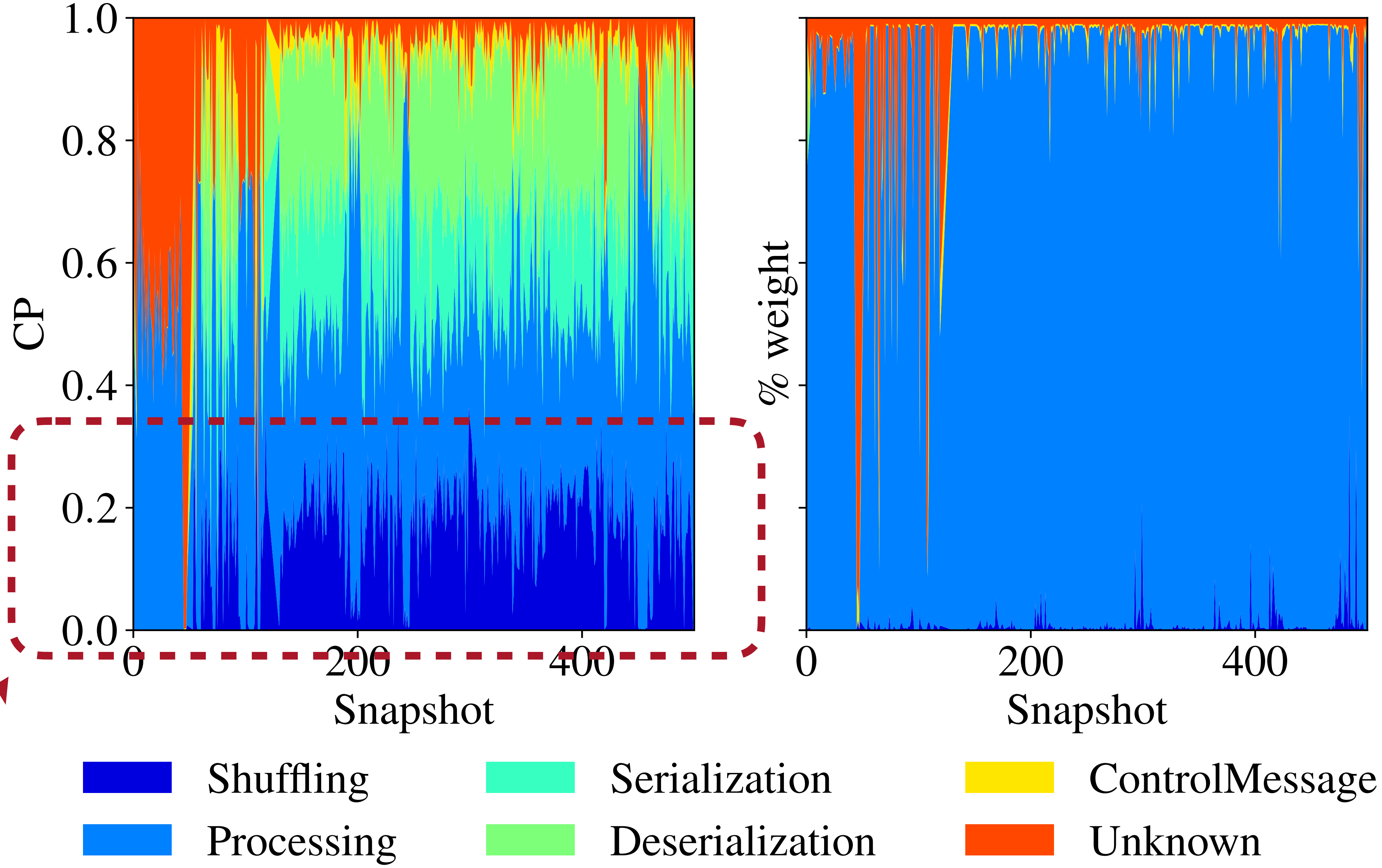
[1] TPC-DS. http://www.tpc.org/tpcds/

[2] Ousterhout, K. Spark performance analysis (accessed: April 2017) https://kayousterhout.github.io/trace- analysis/

COMPARISON WITH CONVENTIONAL PROFILING



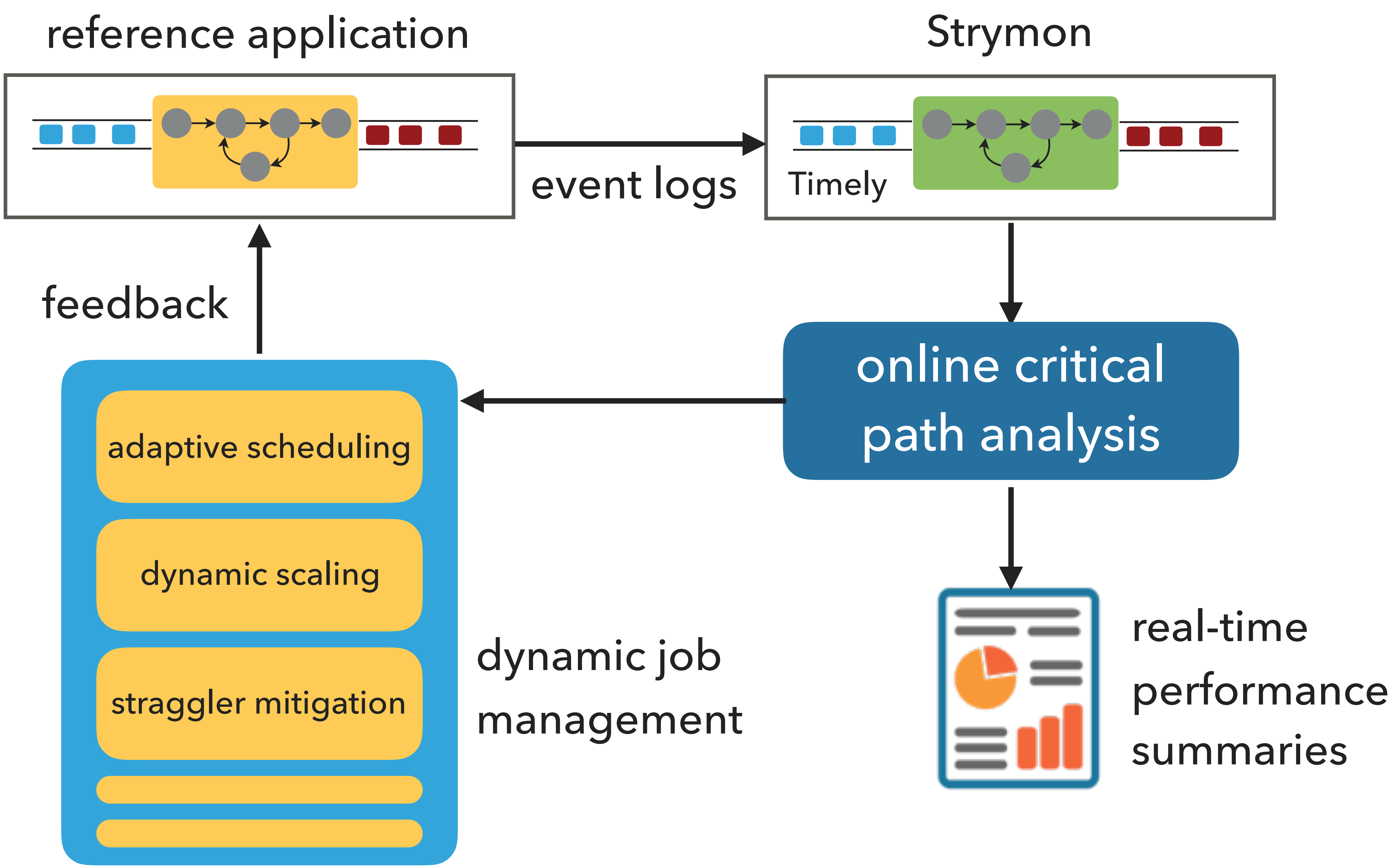
COMPARISON WITH CONVENTIONAL PROFILING



"Optimizing disk usage can improve performance by a median of at most 19%"

Ousterhout, K., Rasti, R., Ratnasamy, S., Shenker, S., and Chun, B.-G.
Making sense of performance in data analytics frameworks. In NSDI (2015).

ONGOING AND FUTURE WORK

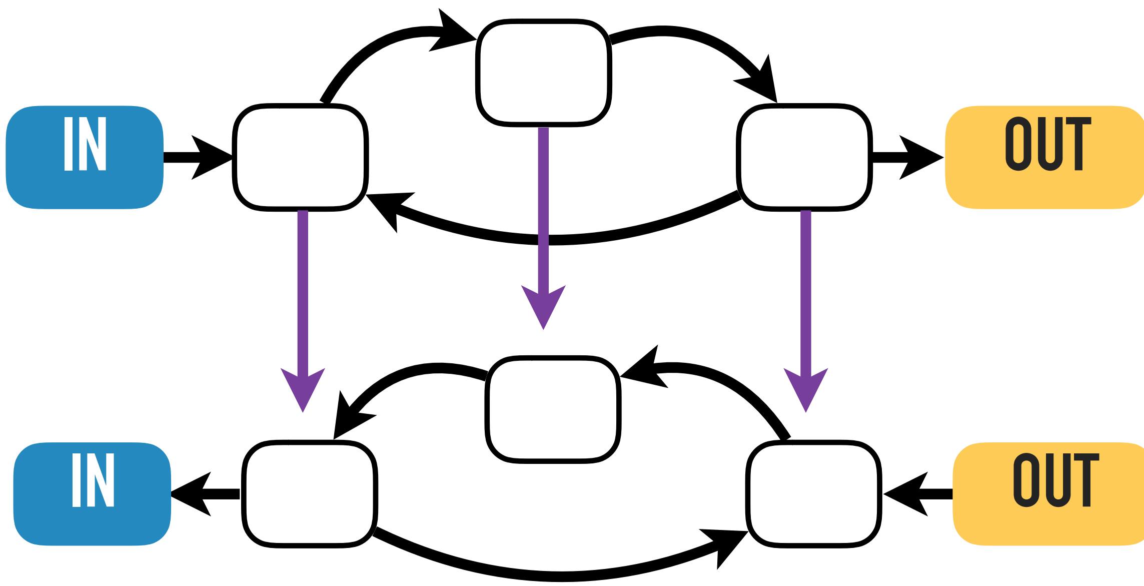


INTERESTING QUESTIONS

- What is the appropriate snapshot size for analyzing the performance of a dataflow execution?
- Can we use sampling to reduce the number of snapshots we examine without affecting the quality of the results?
- Can we use the Program Activity Graph to verify instrumentation?

SUMMARY

PART I: Iterative Backward Tracing

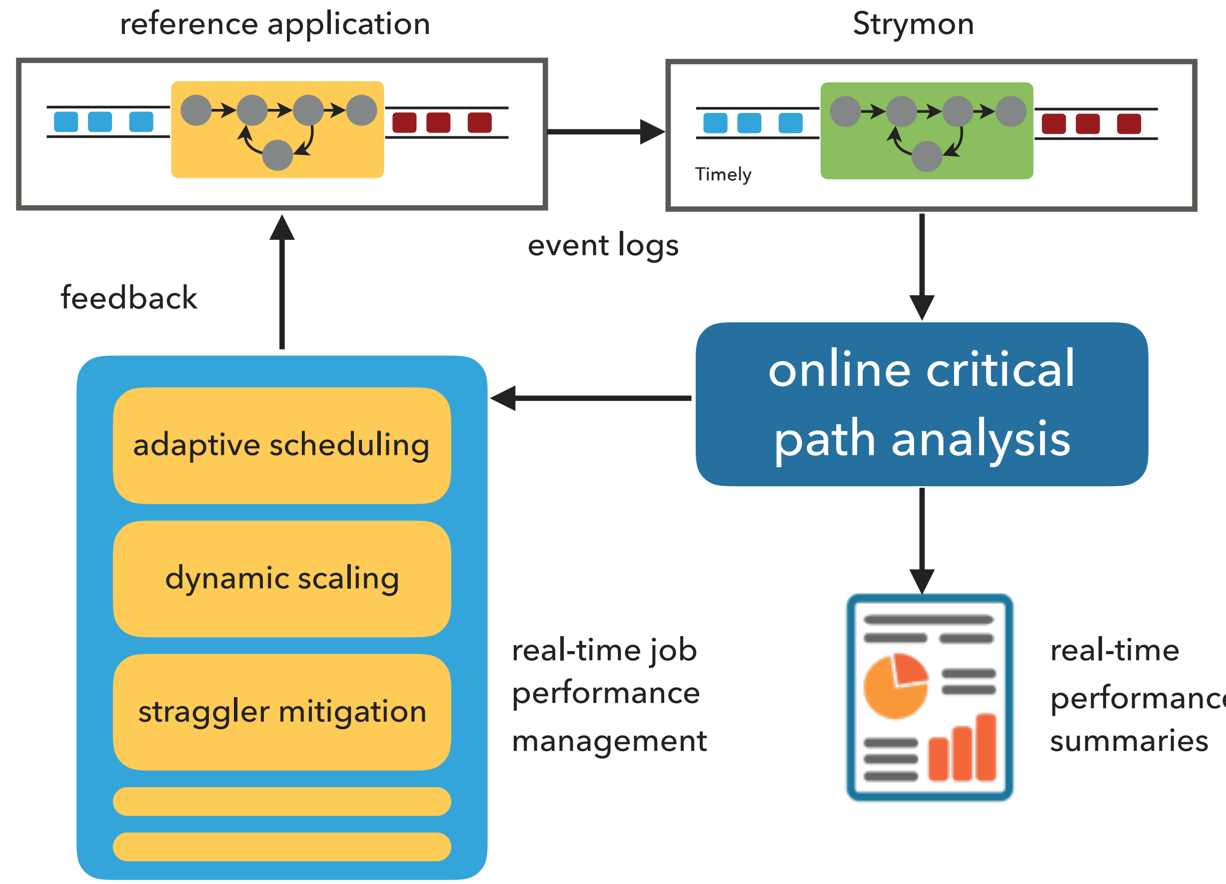


concise explanations

output reproduction guarantees

interactive times

Part II: Transient Critical Path Analysis



transient critical paths

real-time performance summaries

continuous computations

UNDERSTANDING DISTRIBUTED DATAFLOW SYSTEMS

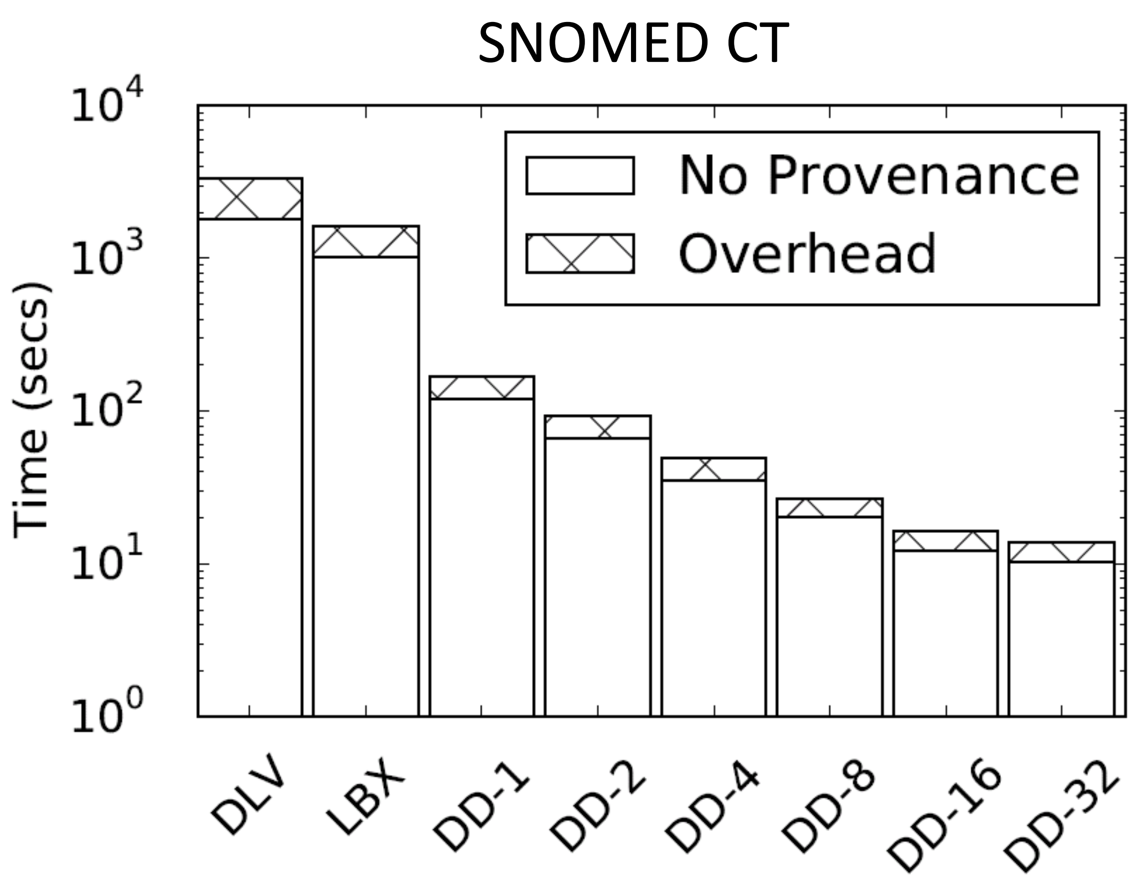
John Liagouris
liagos@inf.ethz.ch

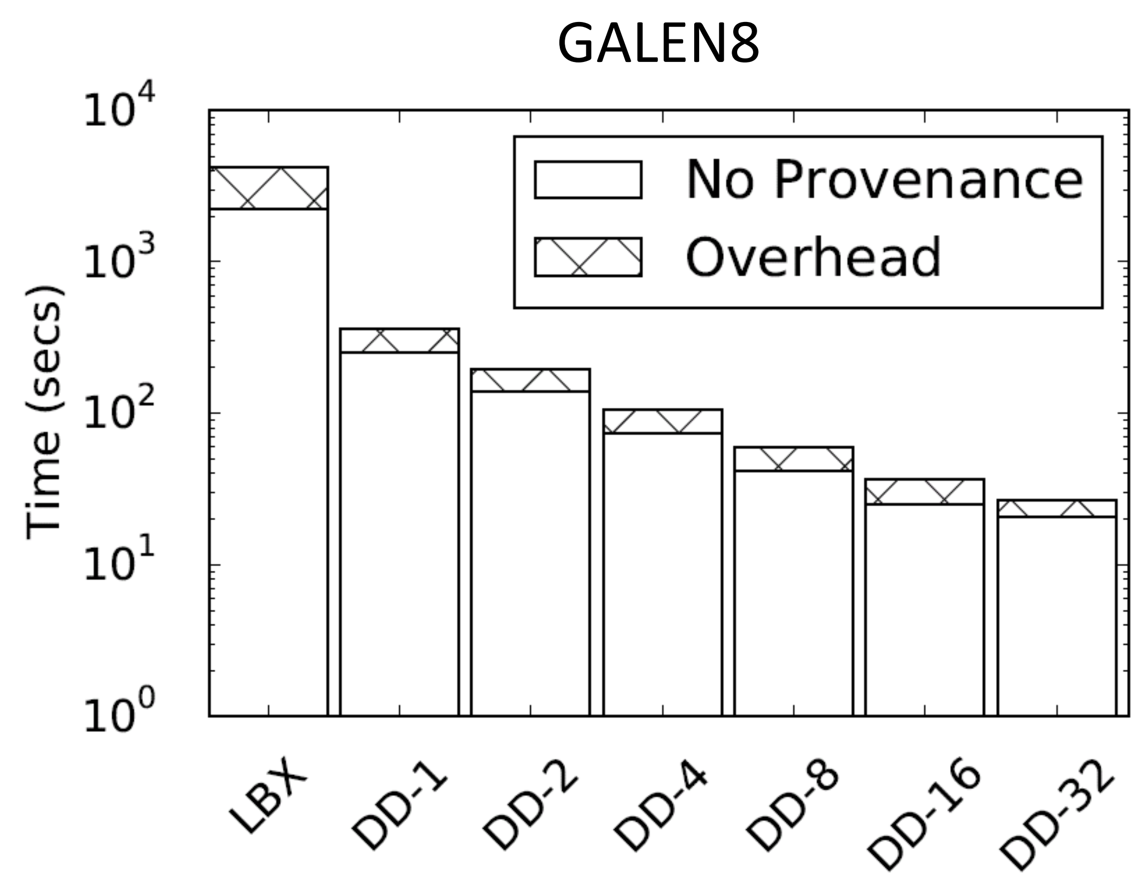
OUTPUT EXPLANATION AND PERFORMANCE ANALYSIS

Google

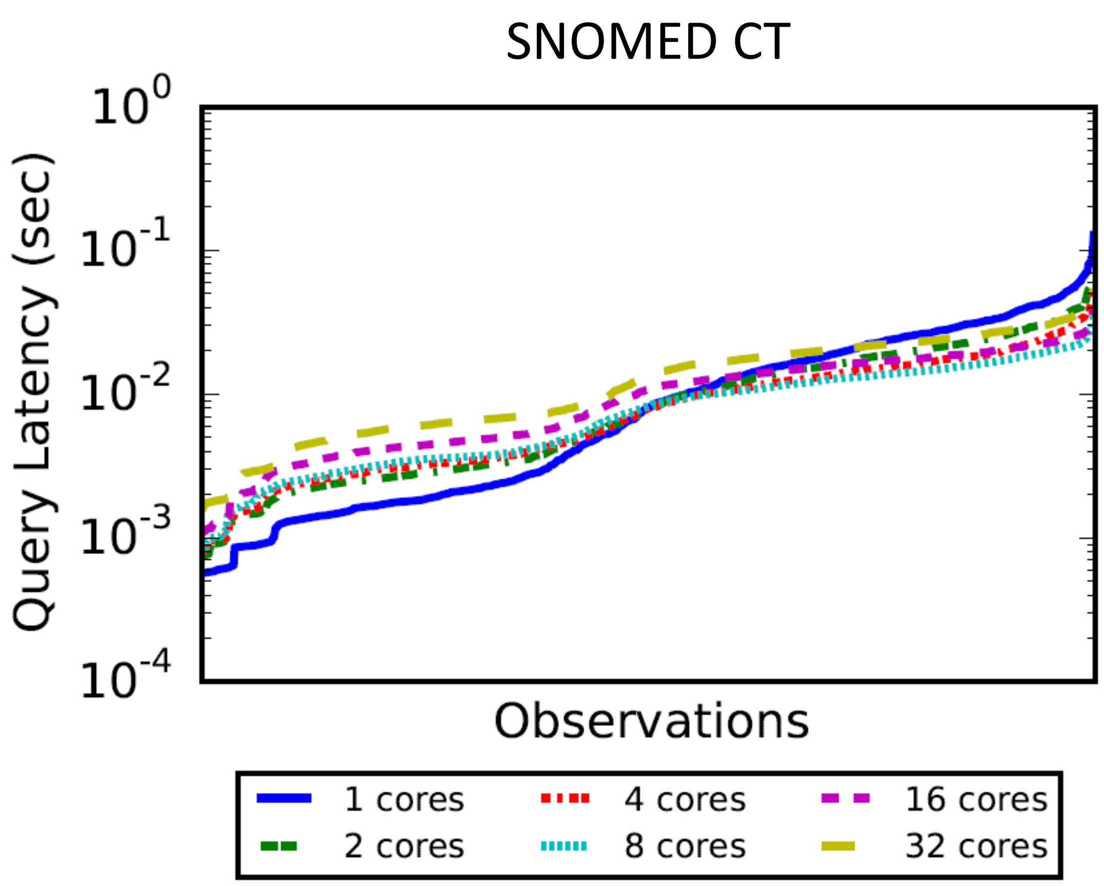
3 May 2017

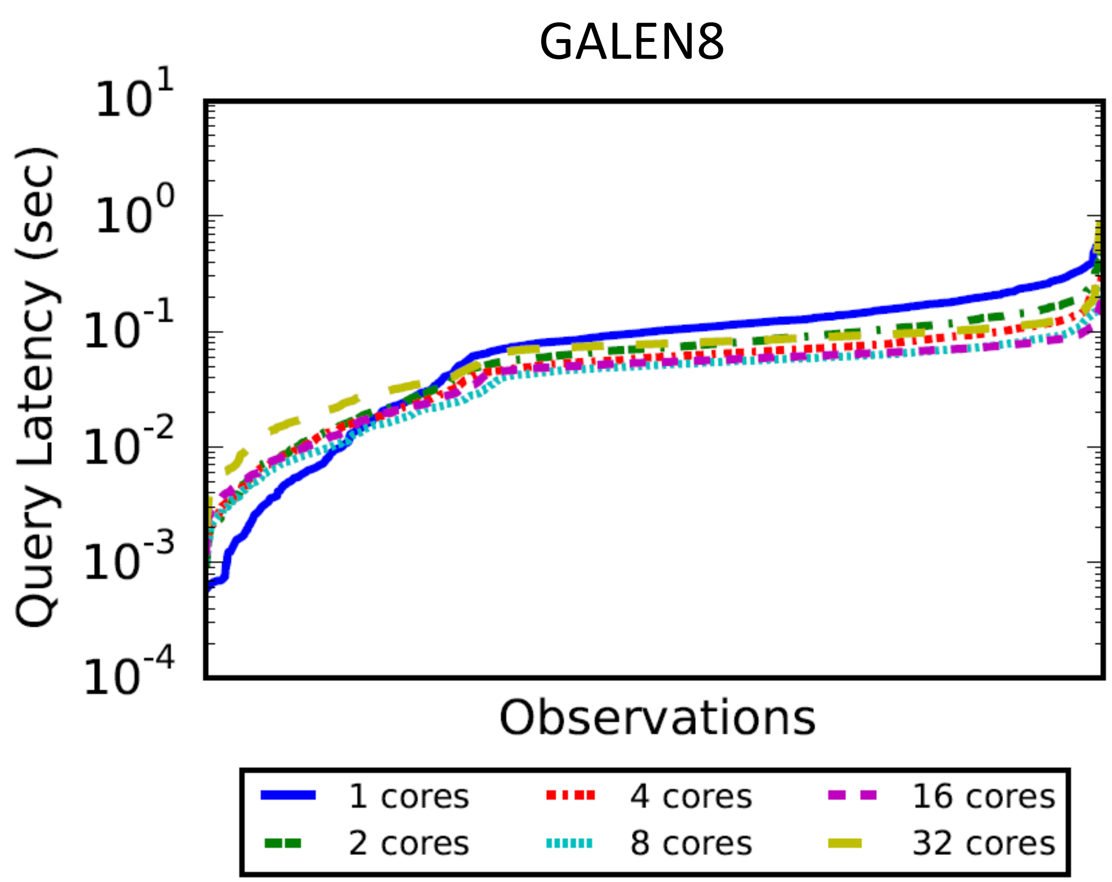
RESULTS: PROVENANCE OVERHEAD IN DATALOG



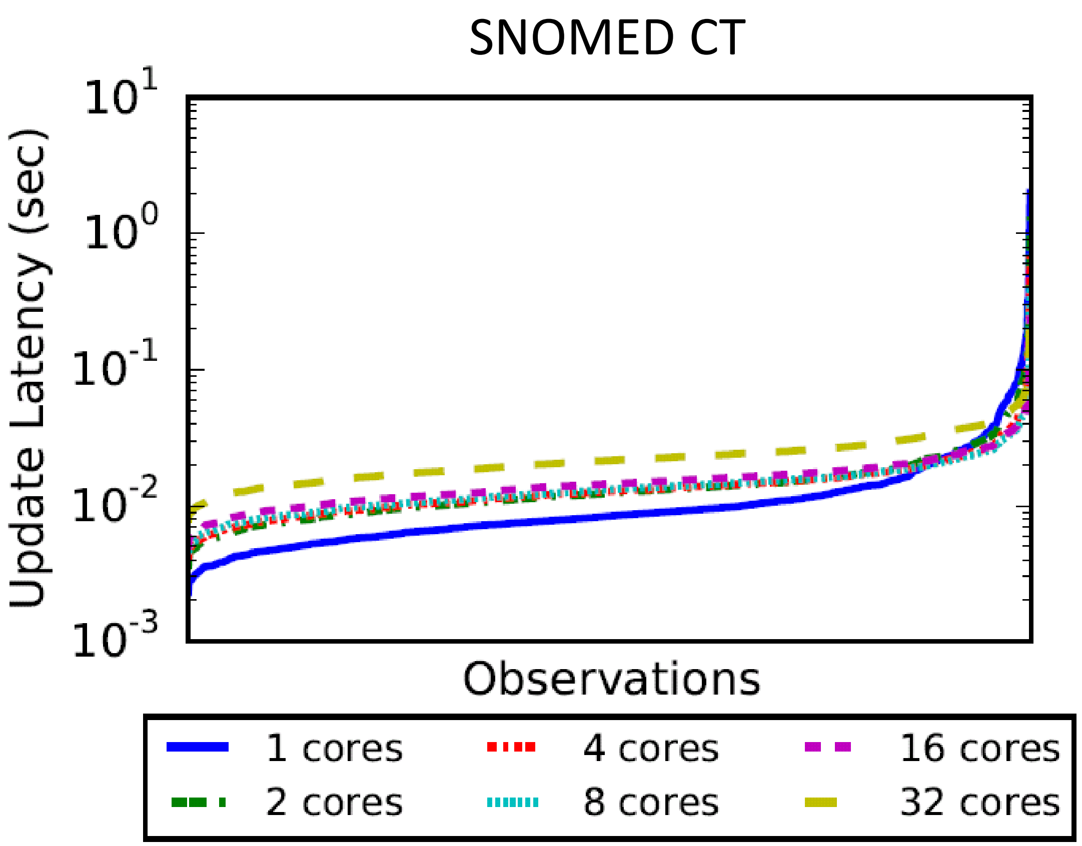


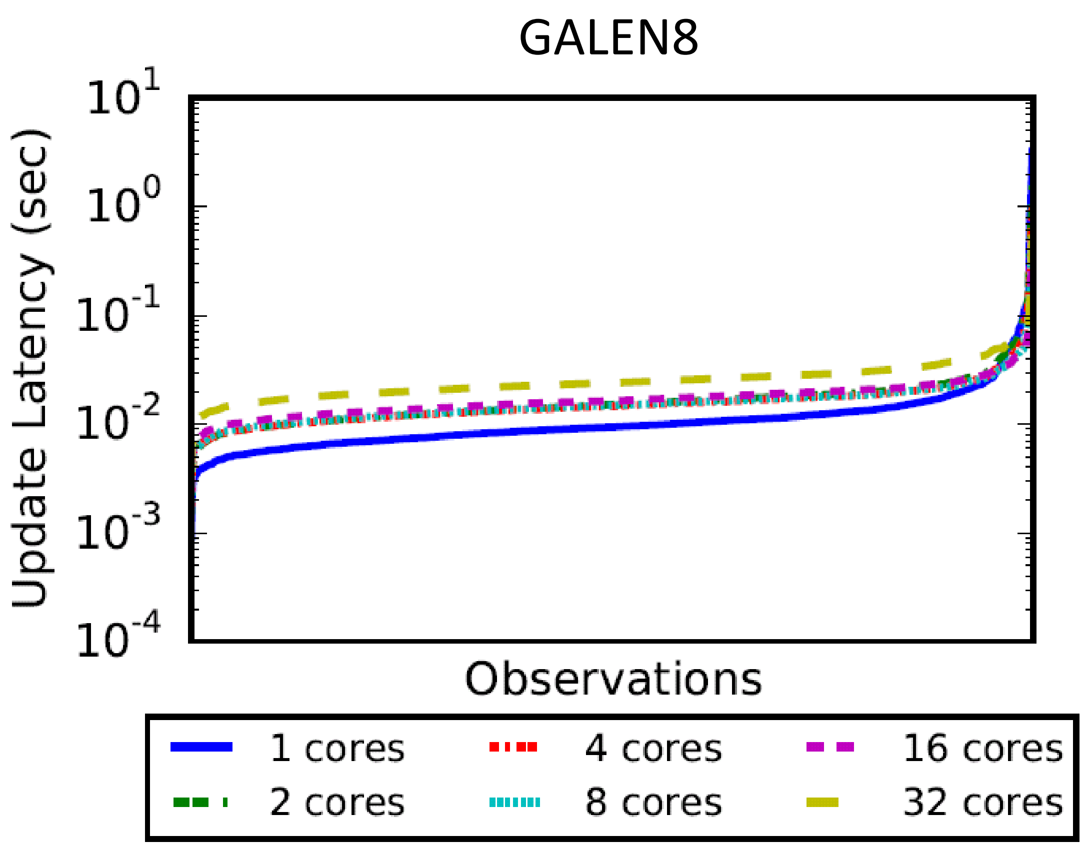
RESULTS: EXPLANATIONS IN DATALOG



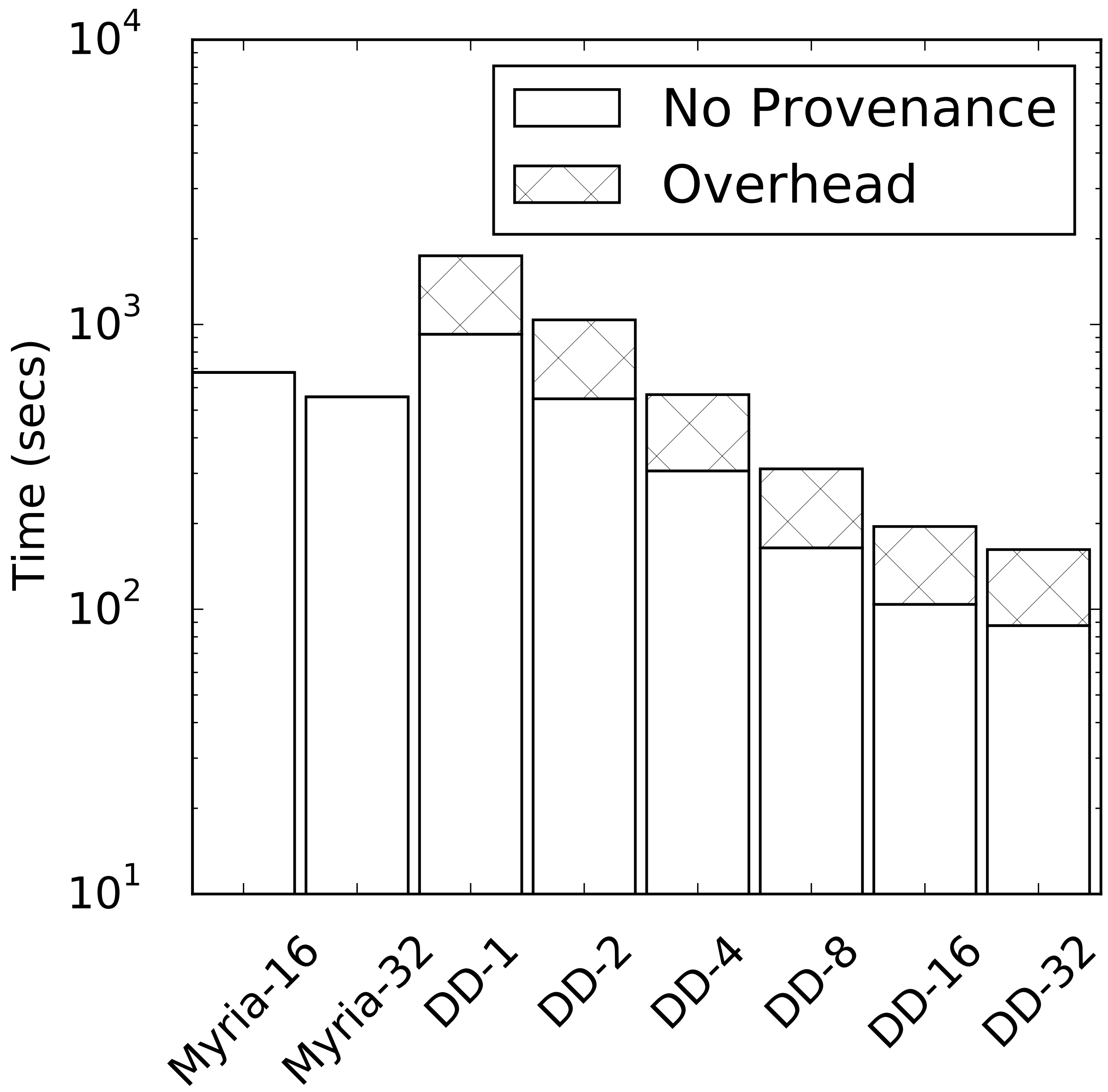


RESULTS: UPDATING PROVENANCE IN DATALOG

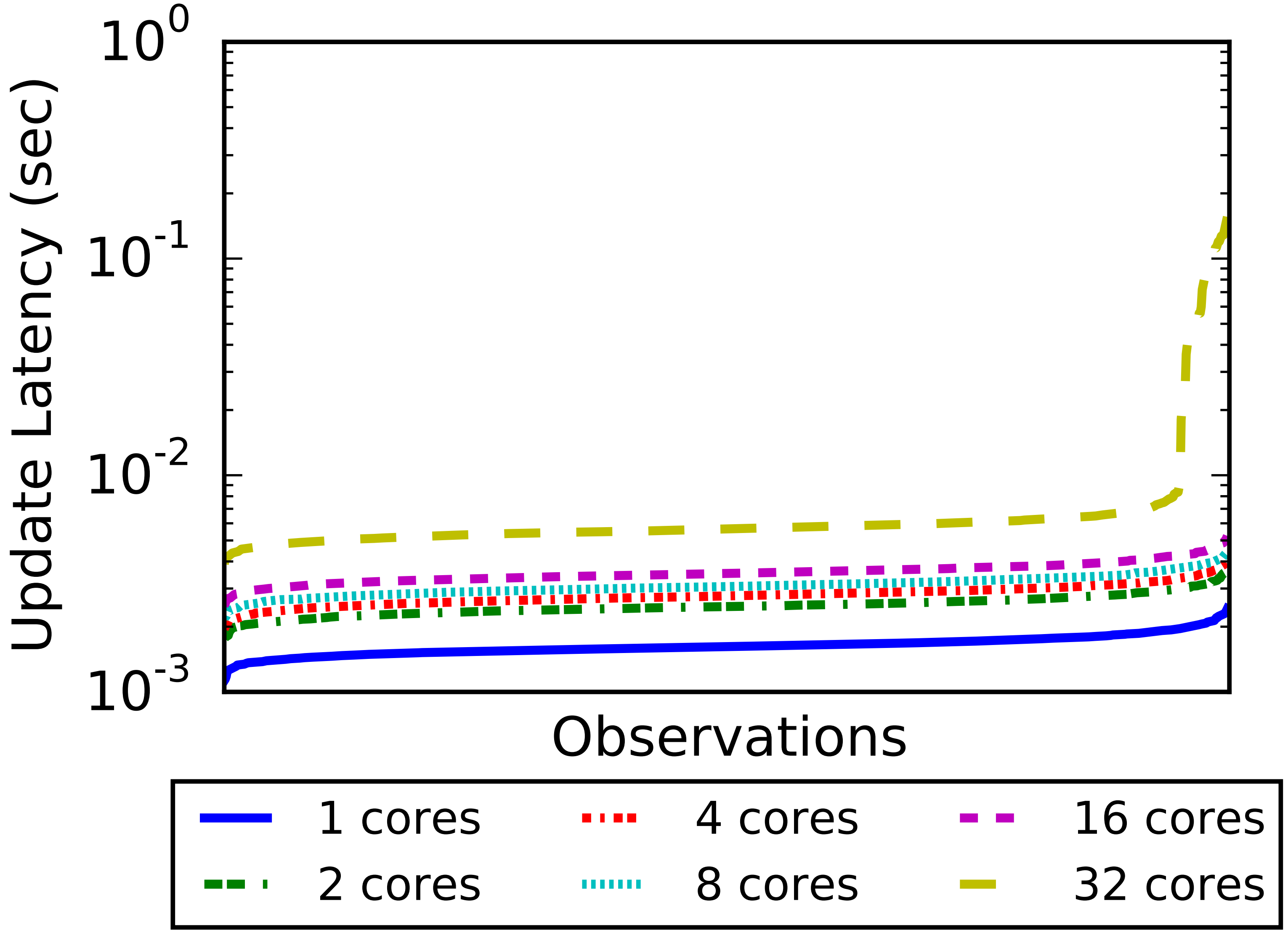




RESULTS: PROVENANCE OVERHEAD IN CONNECTED COMPONENTS



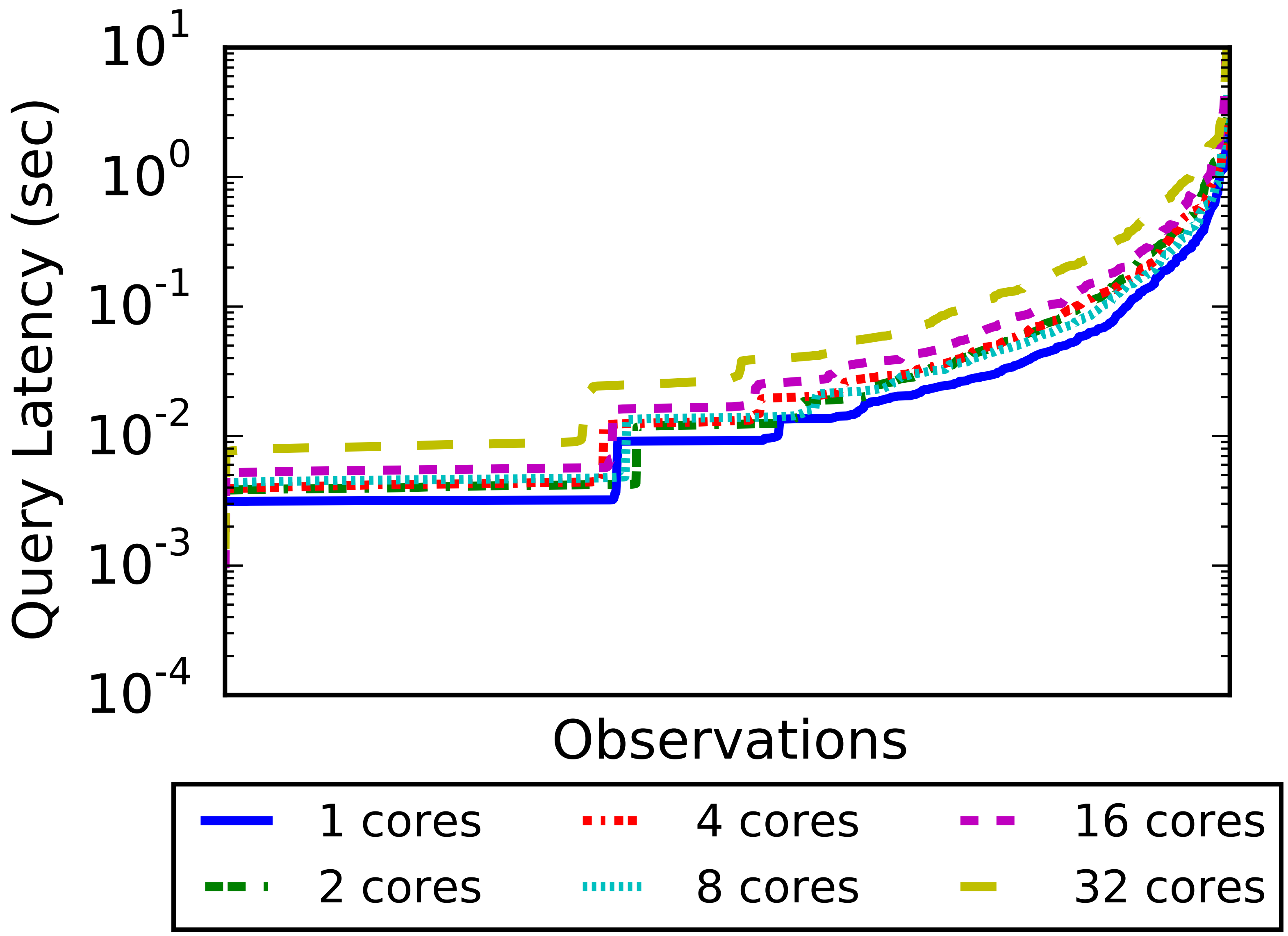
RESULTS: UPDATING EXPLANATIONS IN CONNECTED COMPONENTS



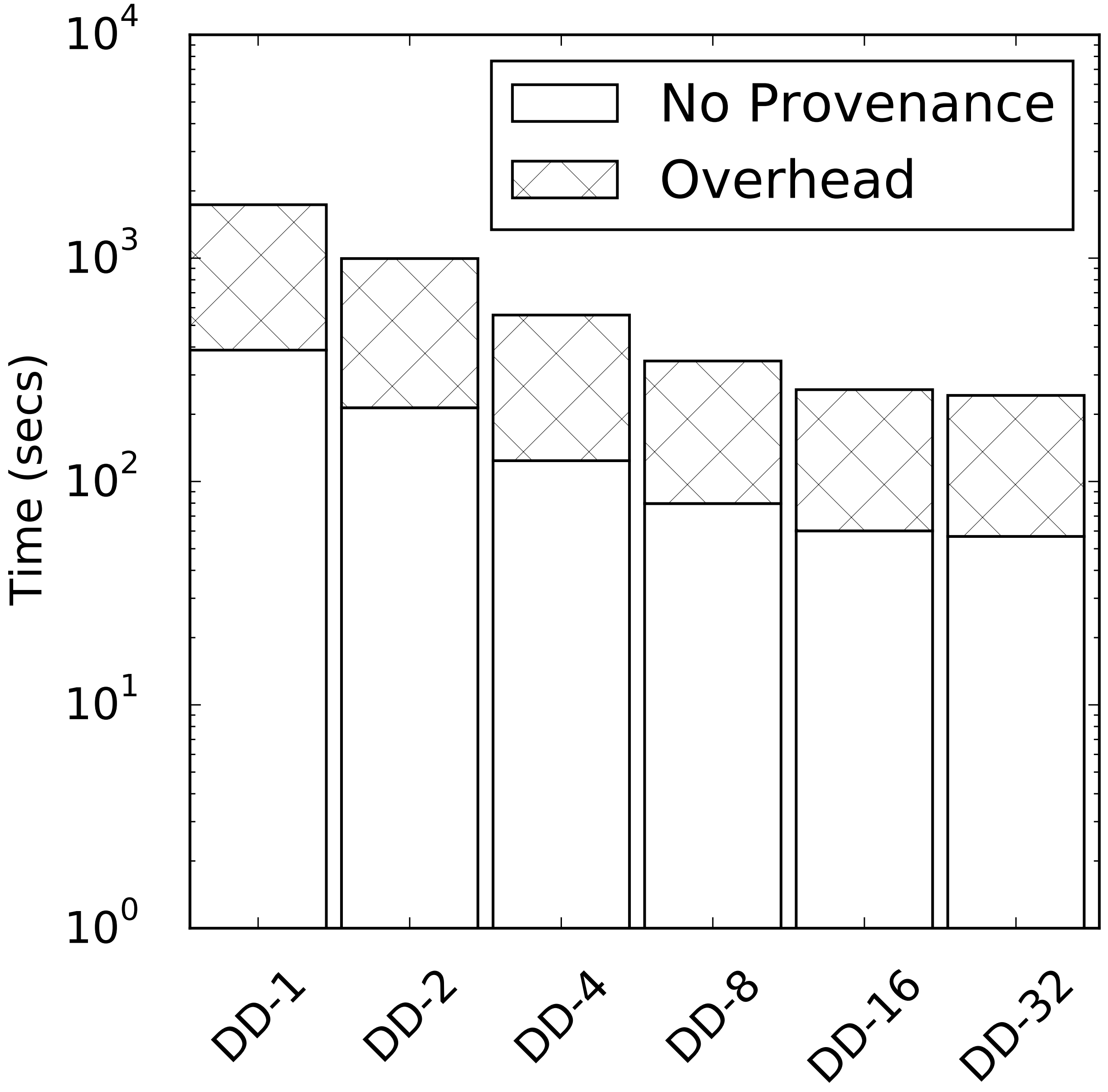
RESULTS: EXPLAINING STABLE MATCHING IN GRAPHS

- Dataset: A subset of the Twitter graph with 300M edges
- Algorithm: Stable Matching
- Output: Records of the form (A,B) denoting that nodes A and B matched
- System used: Differential Dataflow
- Machine used: Intel Xeon E5-4640 at 2.4GHz with 32 cores and 500G RAM

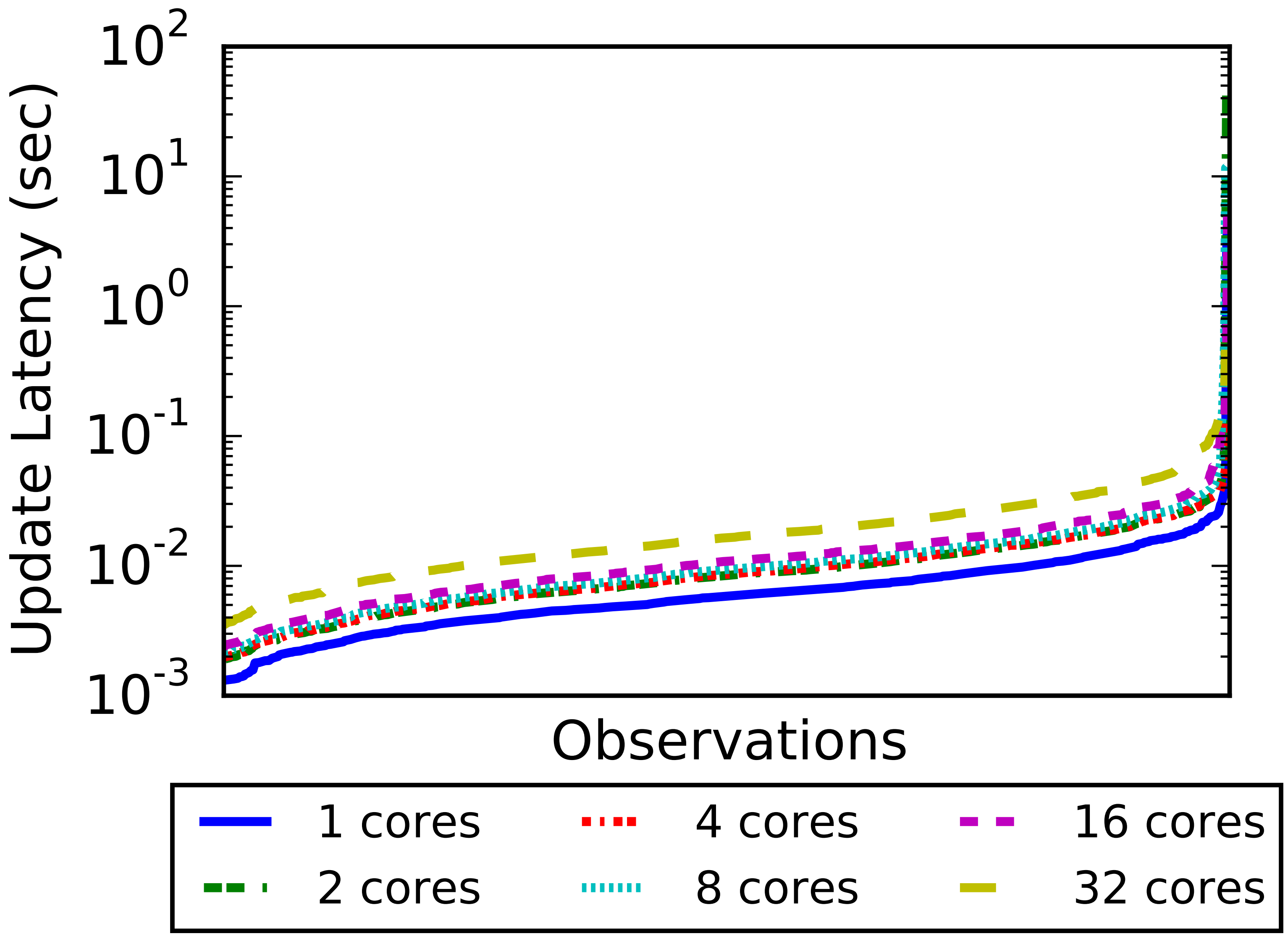
EXPLAINING STABLE MATCHING IN GRAPHS



RESULTS: PROVENANCE OVERHEAD IN STABLE MATCHING



RESULTS: UPDATING EXPLANATIONS IN STABLE MATCHING

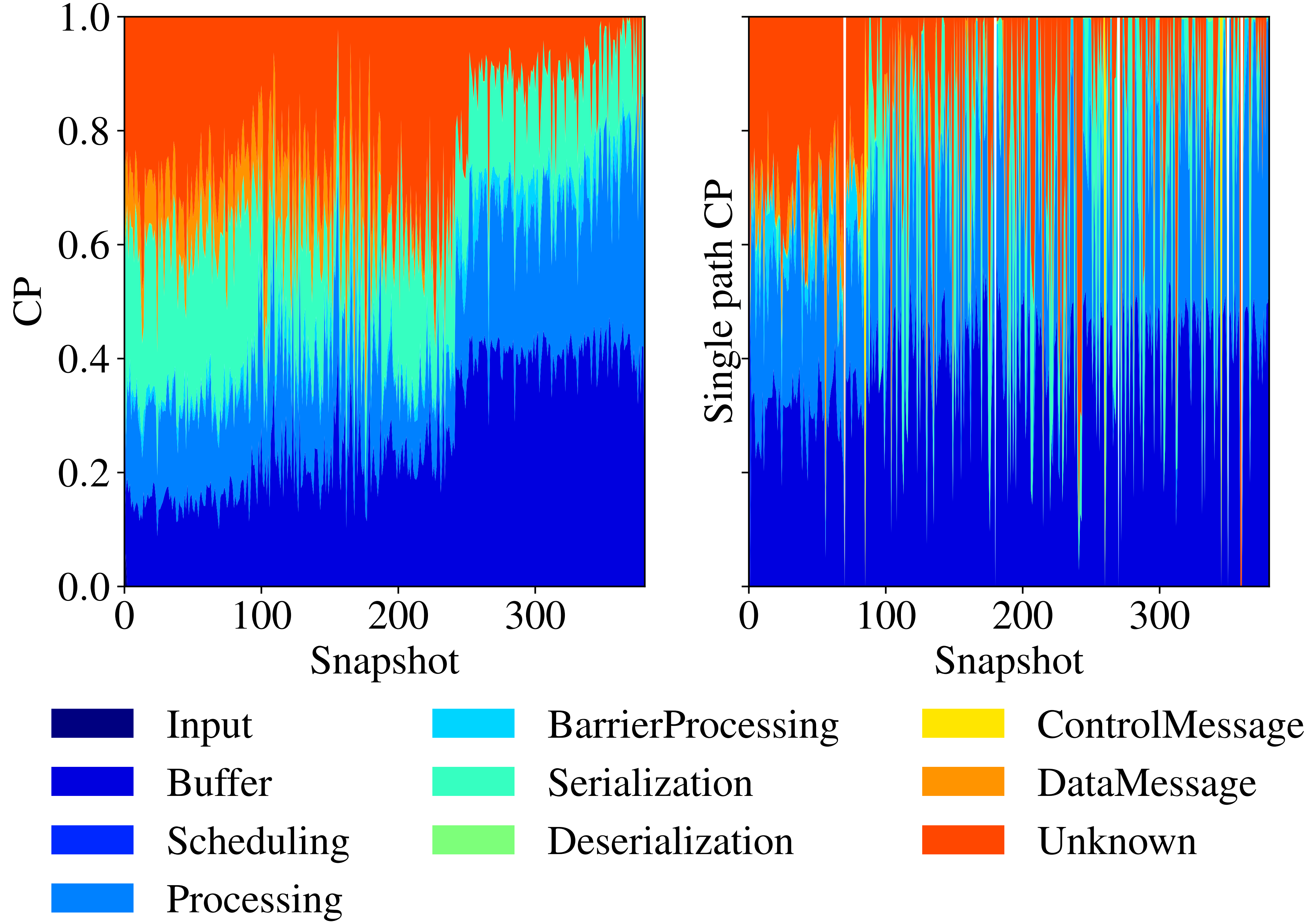


RESULTS: COMPARISON WITH SINGLE-PATH APPROACH

- Benchmark: Yahoo Streaming Benchmark (YSB) [1]
- System under study: Flink (1.2.0)
- Setting: 1 machine with 8 workers
- Snapshot interval: 1 sec

[1] Yahoo Streaming Benchmark.
https://github.com/yahoo/streaming-benchmarks

COMPARISON WITH SINGLE-PATH APPROACH



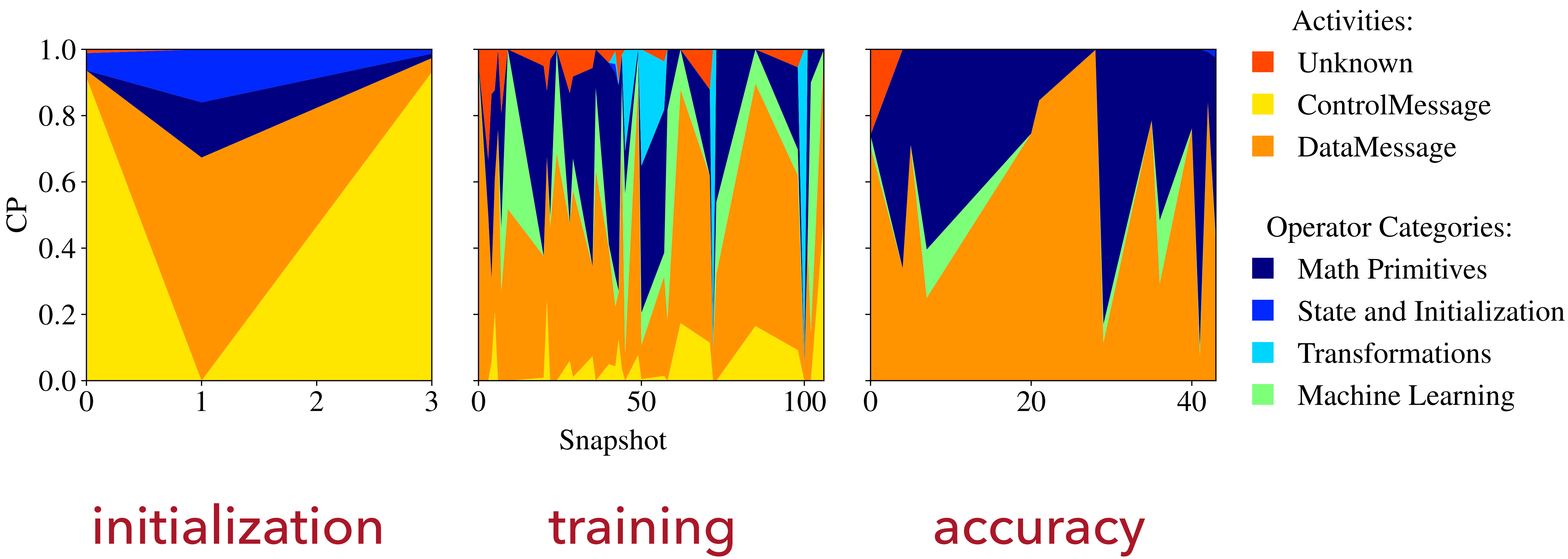
RESULTS: PROFILING DIFFERENT PHASES OF A ML JOB

- Benchmark: AlexNet program [1] on ImageNet [2]
- System under study: TensorFlow (1.0.1)
- Setting: 1 machine 16 workers (CPU threads)
- Snapshot interval: 1 sec

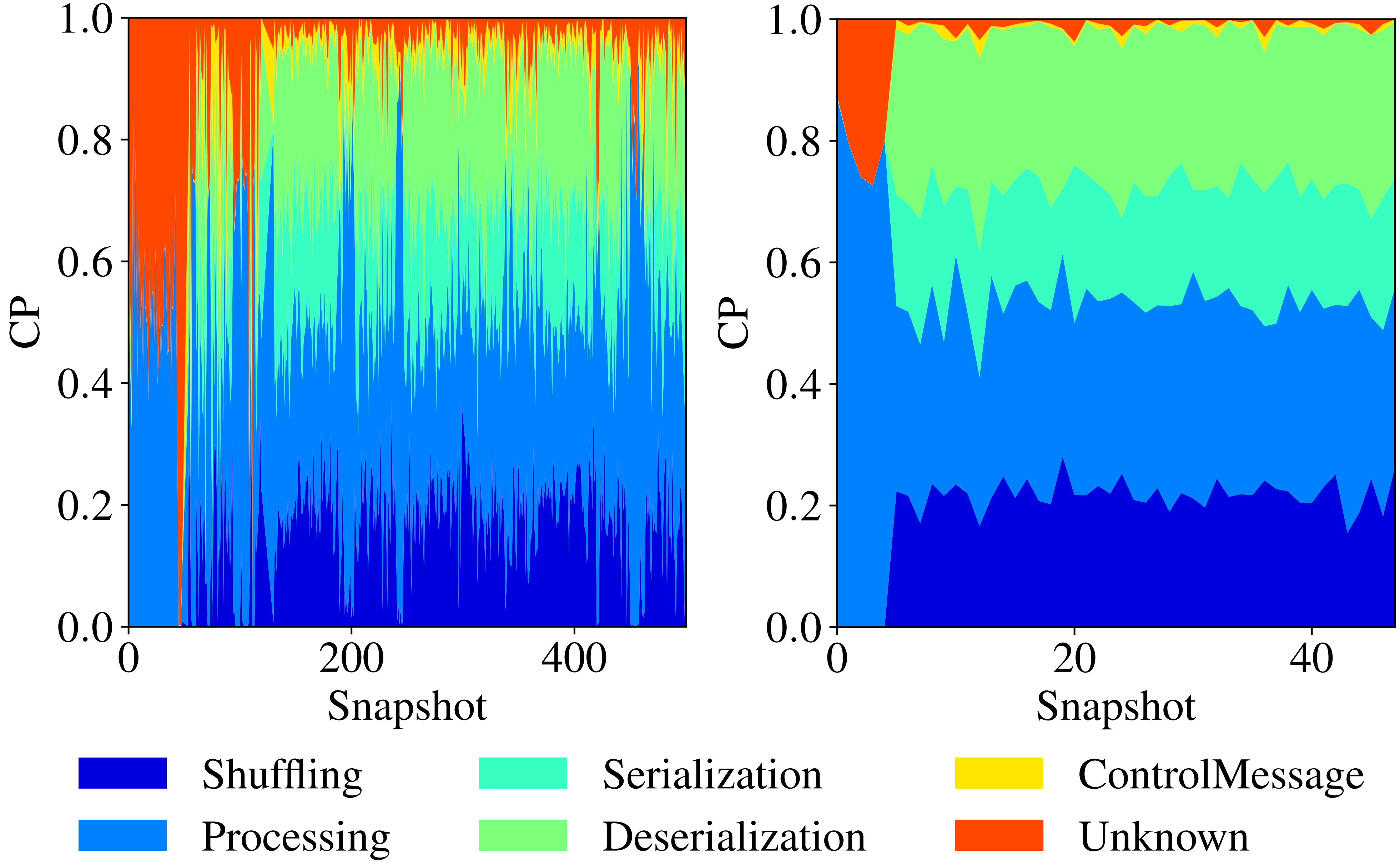
[1] Krizhevsky, A., Sutskever, I., and Hinton, G. E. *ImageNet classification with deep convolutional neural networks*. In Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[2] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. *ImageNet Large Scale Visual Recognition Challenge*. International Journal of Computer Vision (IJCV) 115, 3 (2015), 211–252.

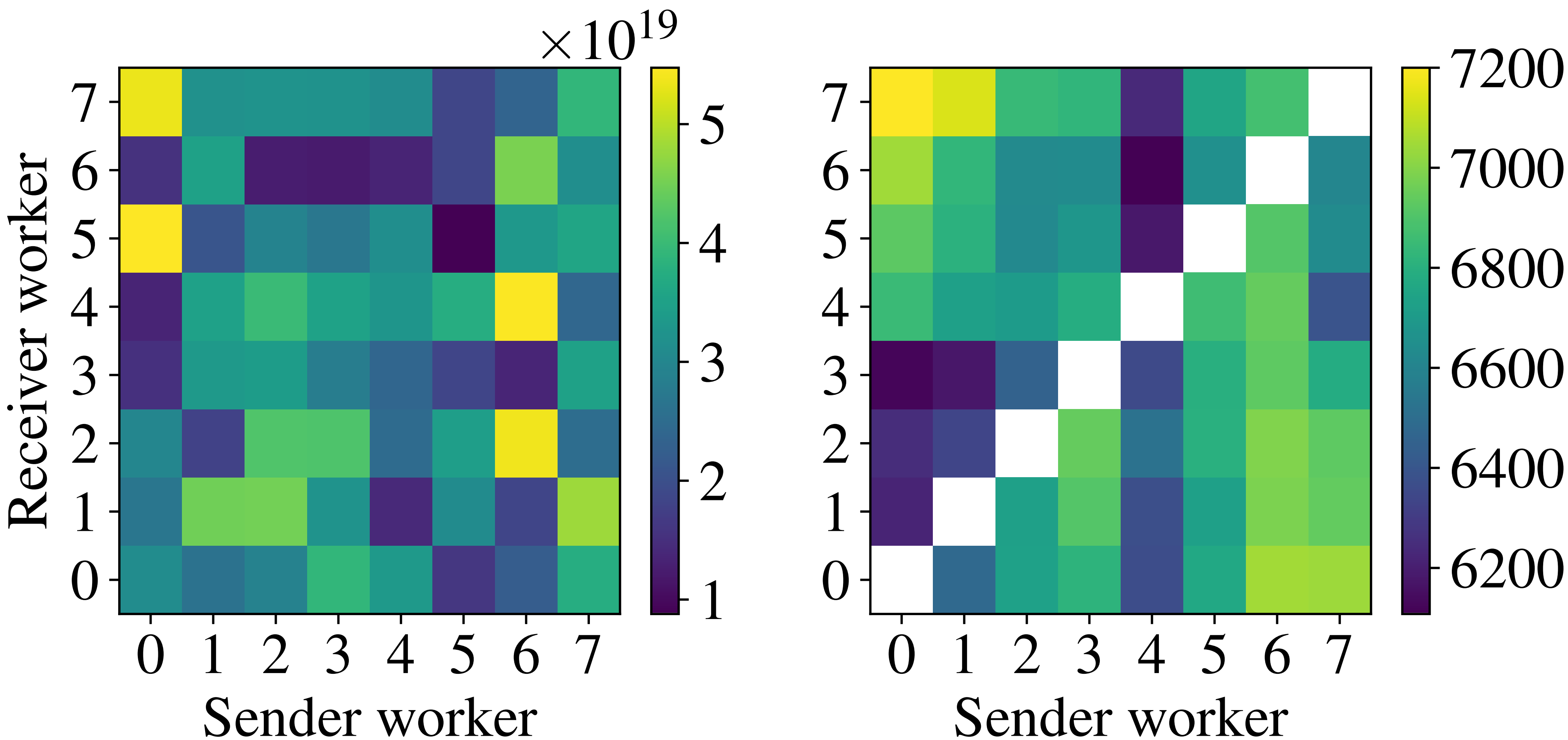
PROFILING DIFFERENT PHASES OF A MACHINE LEARNING JOB



STABILITY ACROSS DIFFERENT SNAPSHOT INTERVALS



COMMUNICATION SKEW IN TIMELY DATAFLOW



COMPUTATION SKEW IN FLINK