Online Reconstruction of Structural Information from Datacenter Logs

EuroSys’17, Belgrade – 25.04.2017

Zaheer Chothia

John Liagouris

Desislava Dimitrova

Timothy Roscoe

Systems@ETH Zürich

Overall ambition:

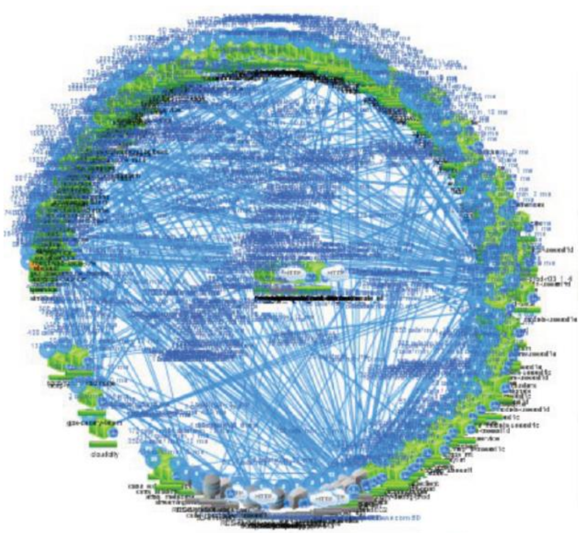
- Understand dynamics of real datacenter workloads
- Online, continuously and with modest resources

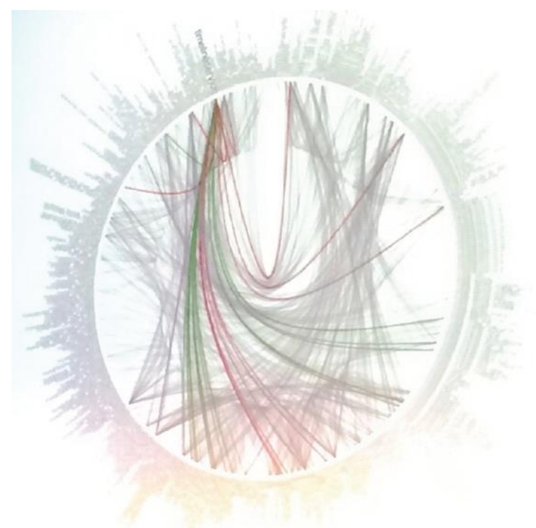
Glimpse:

- System to processes log streams at *gigabits* per second
- Reconstruct sessions comprising *millions of transactions*
- In real time while dealing with real-world phenomena that make such a task challenging

DC component interactions are complex and interwoven

- OS no longer has a global view of resources
- Within a node: thread pools, event loops, callbacks
- Across nodes: asynchrony, different vendors, deep stacks
- Timestamps alone do not give consistent ordering to events





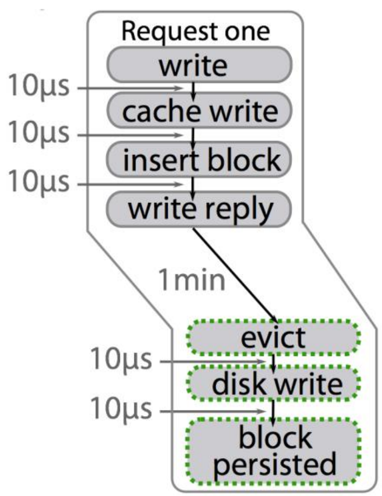
Microservice architecture (Netflix and Twitter)

http://gotocon.com/dl/goto-chicago-2014/slides/AdrianCockcroft_TuesdayMorningKeynoteSpeedAndScaleHowToGetThere.pdf
https://twitter.com/adrianco/status/441883572618948608

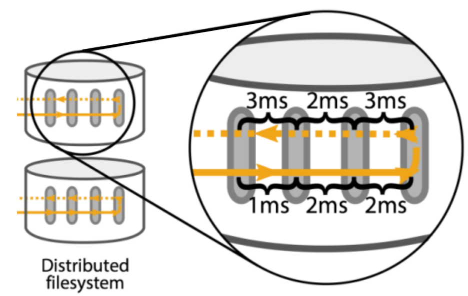
Motivation: resource accounting

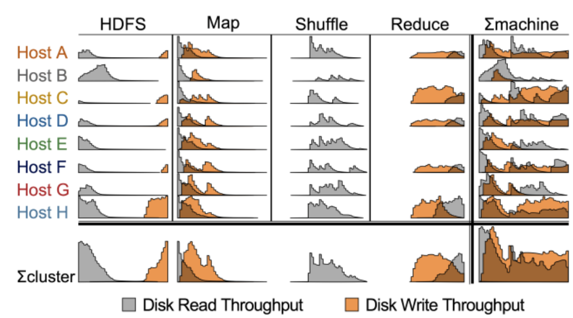
Task: relate all executed work back to the originating request or tenant

Distributed profiling
"Why is this request so slow?"



Billing
"Who should be charged?"

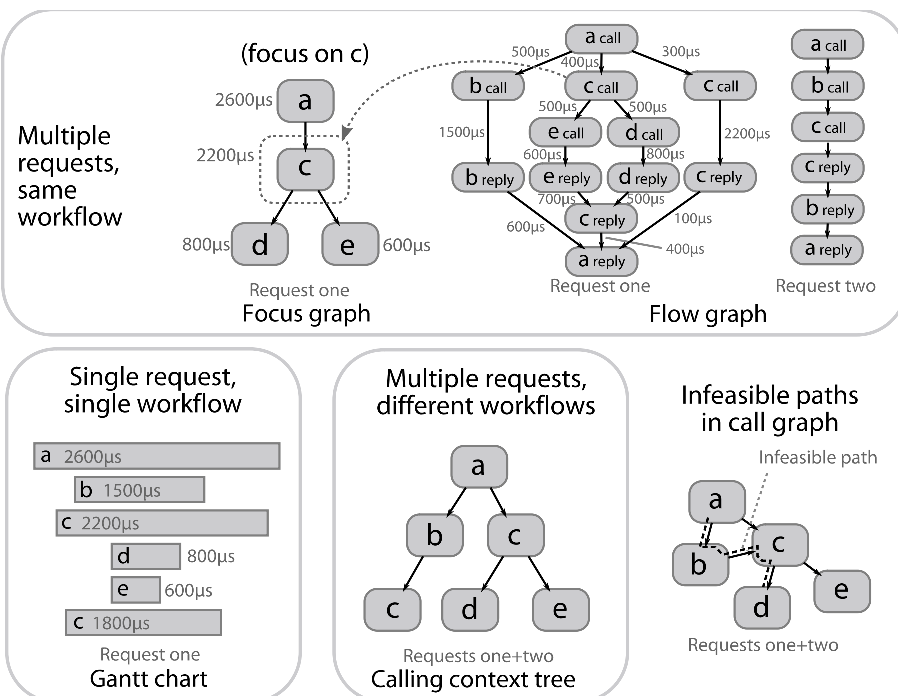




[Google's Dapper, Pivot Tracing]

Compact summaries shed insight

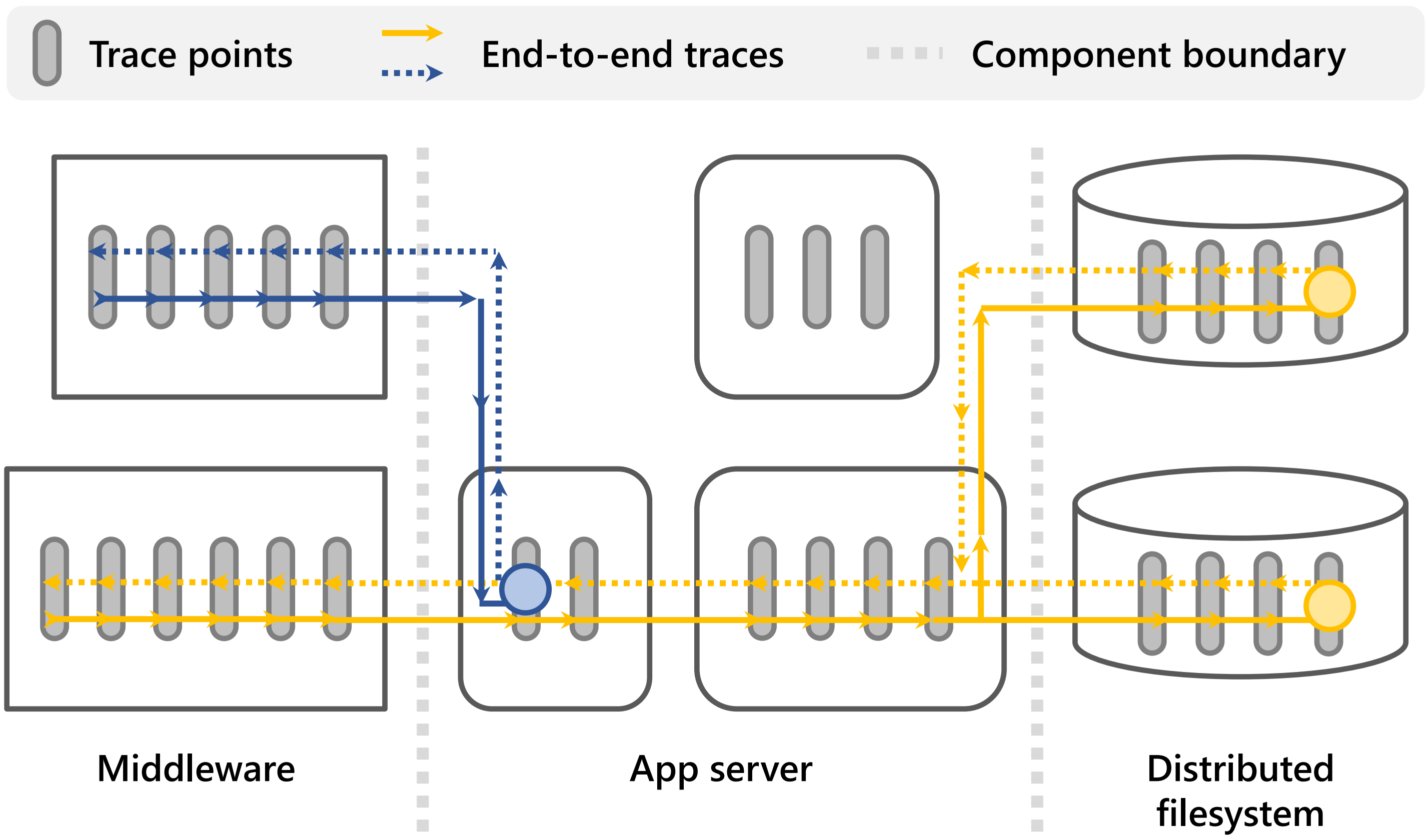
Foundation for diagnostic, profiling, and monitoring tasks essential to the operation of the datacenter



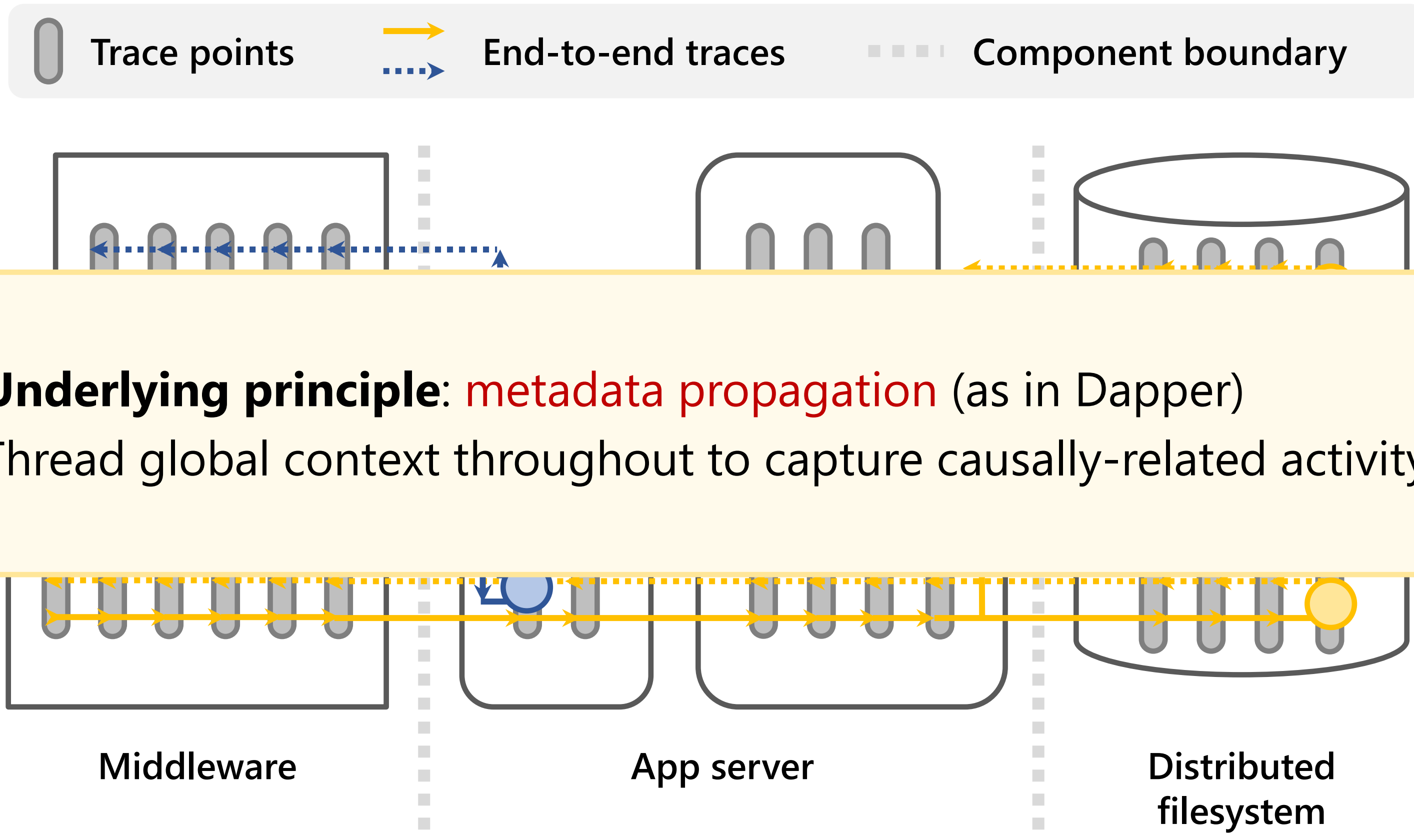
- User sessions
- Spans
- Call graphs
- Transaction trees
- Critical path
- Timing charts

[Sambasivan, Raja R., et al. *So, you want to trace your distributed system? Technical Report, CMU-PDL-14-102*]

DC stack is already heavily instrumented

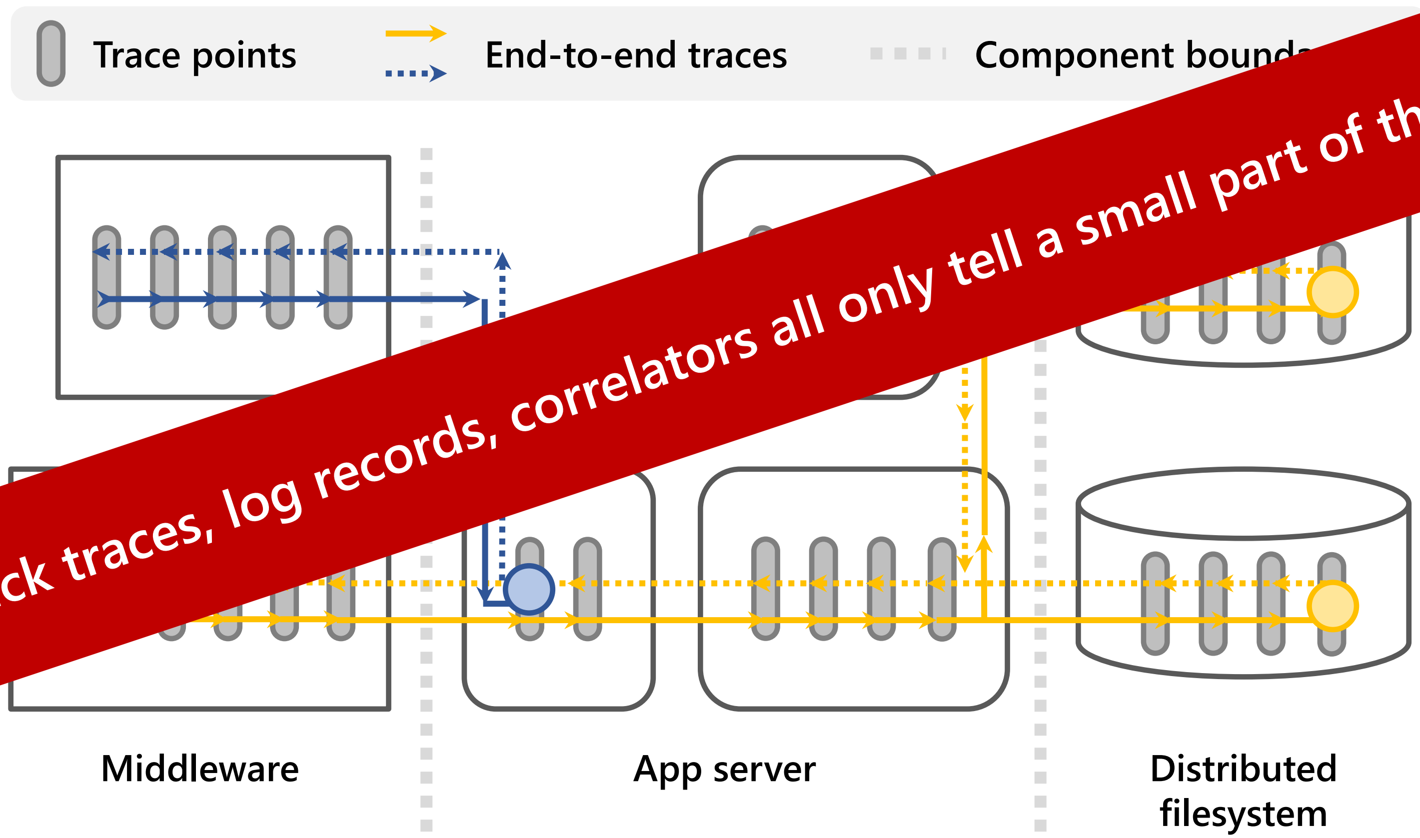


DC stack is already heavily instrumented



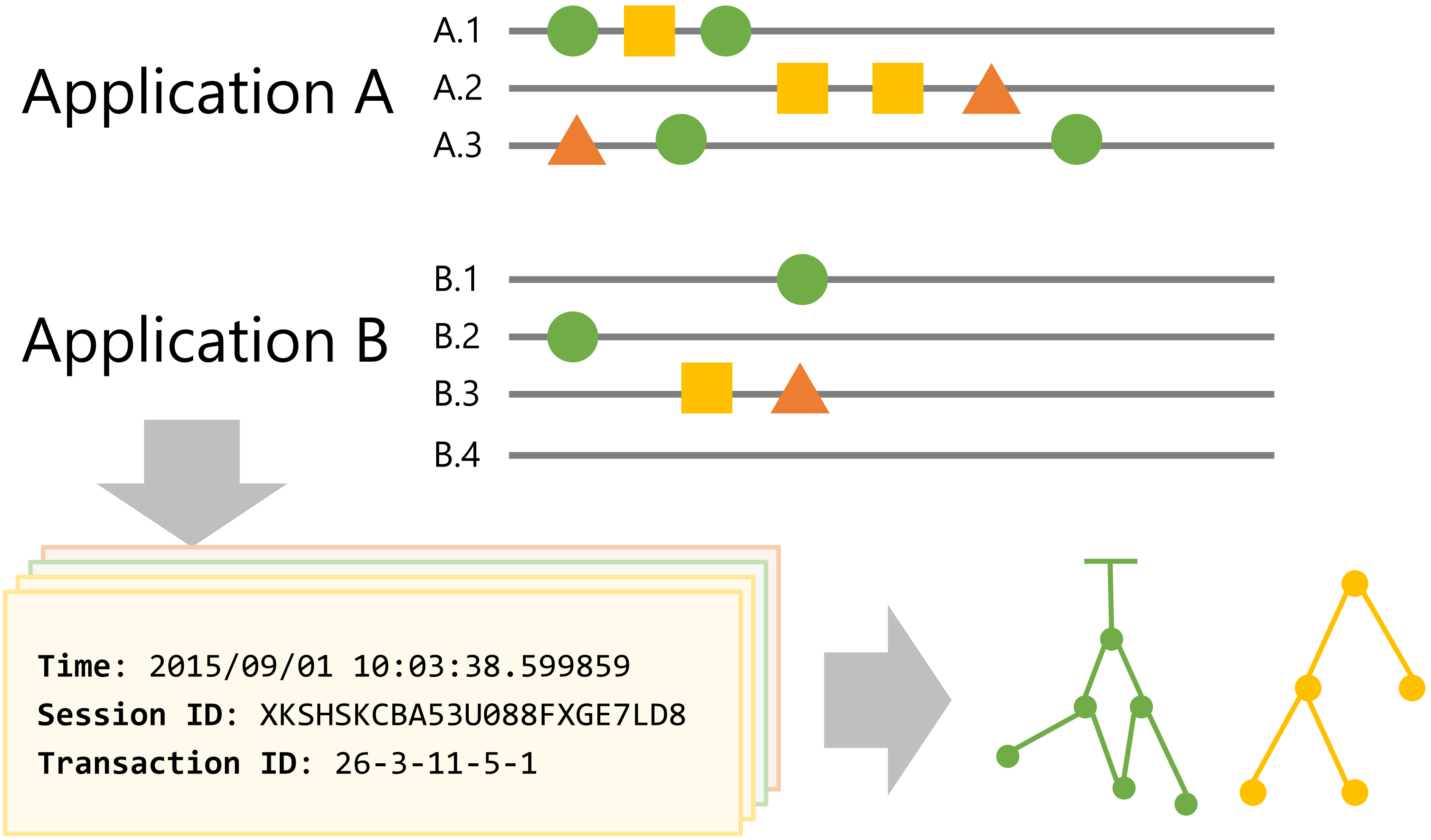
Underlying principle: metadata propagation (as in Dapper)
Thread global context throughout to capture causally-related activity.

DC stack is already heavily instrumented



⇒ Stack traces, log records, correlators all only tell a small part of the story

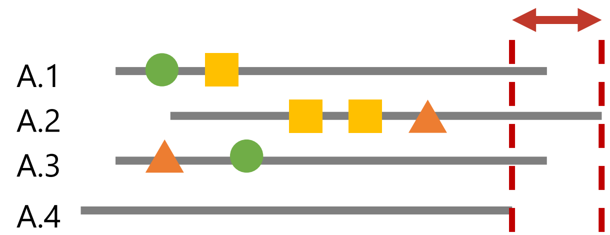
The reconstruction problem



Challenges of using real logs

Out-of-order arrivals

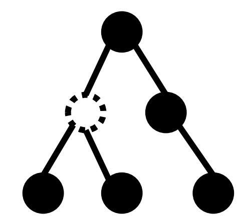
Records arrive in non-deterministic order but within limited time frame (max. observed: 10 seconds)



Missing logs

33.7% have no session ID or transaction number

Incomplete or fragmented trees prevent dependency inference



Reordered logs – Records typically buffered and flushed in batches

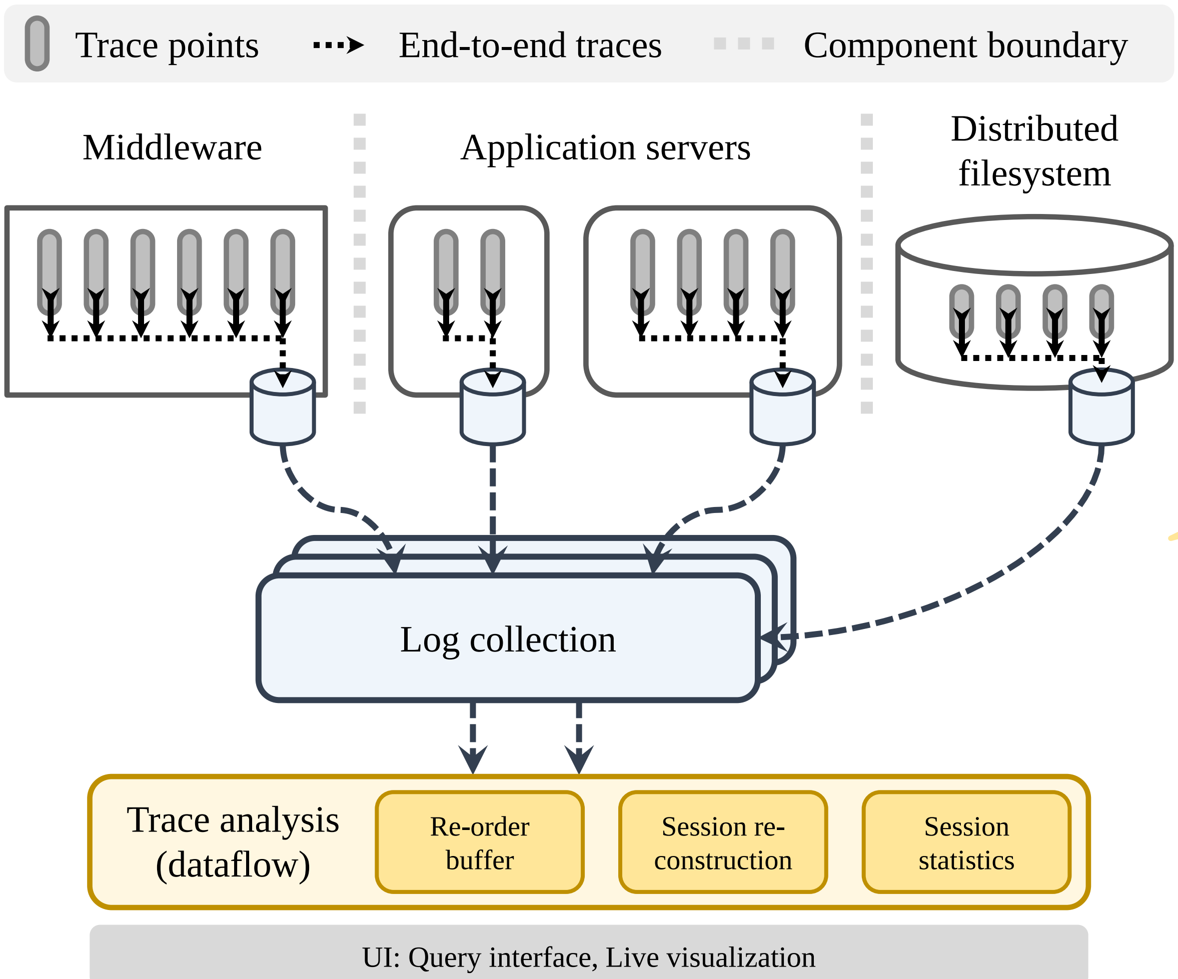
Very long sessions – Inherent skew, high memory requirements

Clock desynchronization

Misordering: message appears to be received *before* sent

Trigger inversion: parent transaction starts *after* child

System architecture and integration



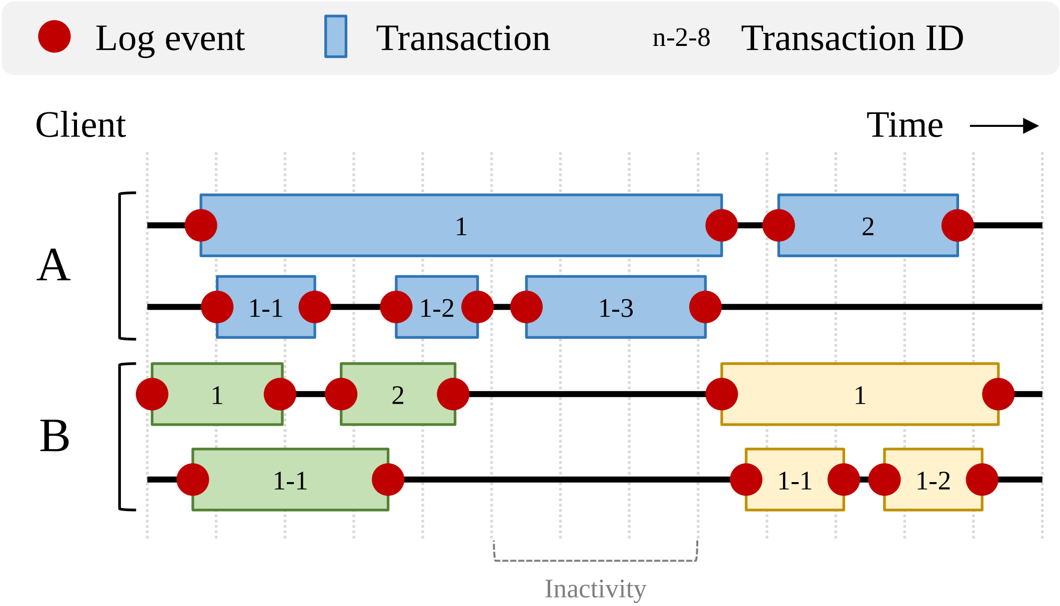
Logs spread across 1263 streams and 42 servers

Mean input rate: 1.3 million events/sec at 424.3 MB/sec

Discrete events → hierarchical trace tree

Terminology

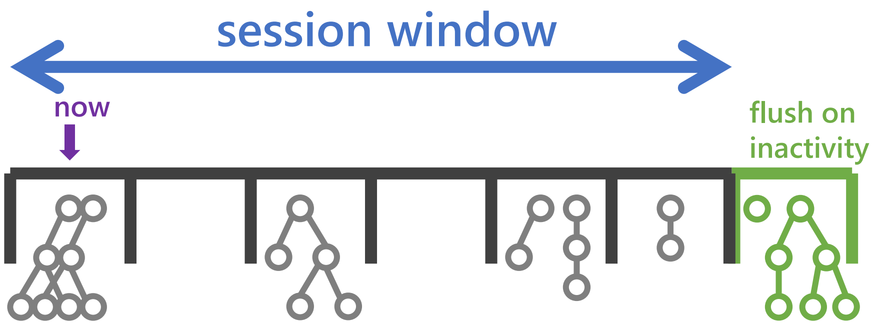
- Tree nodes are the basic unit of work (*spans*)
- Edges indicate casual relationship between a span and its *child spans*
- Timestamped records encode span's *start* and *end time* and application-specific *annotations*

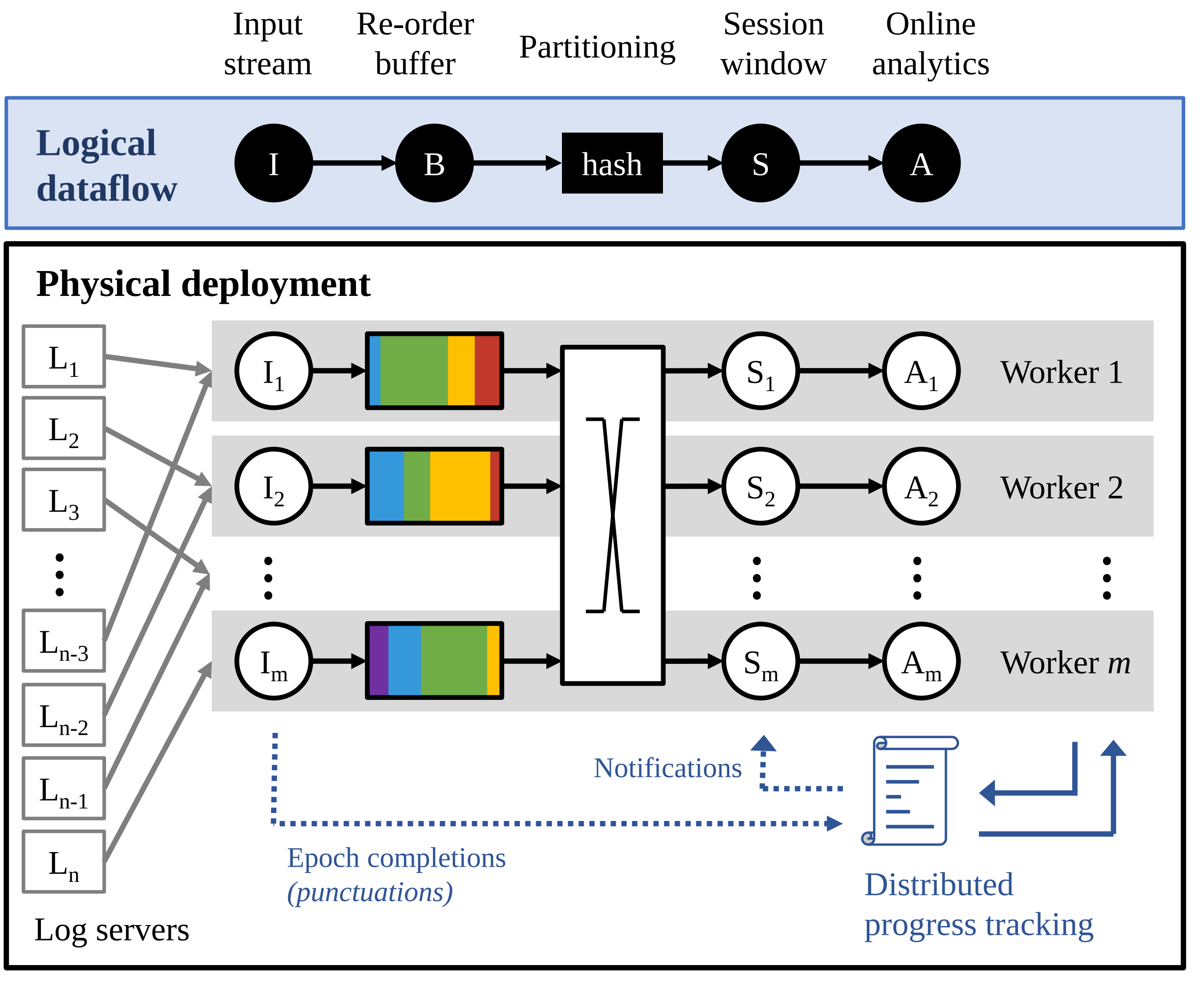


Data-parallel execution

Timely Dataflow – *a low-latency dataflow computational model*

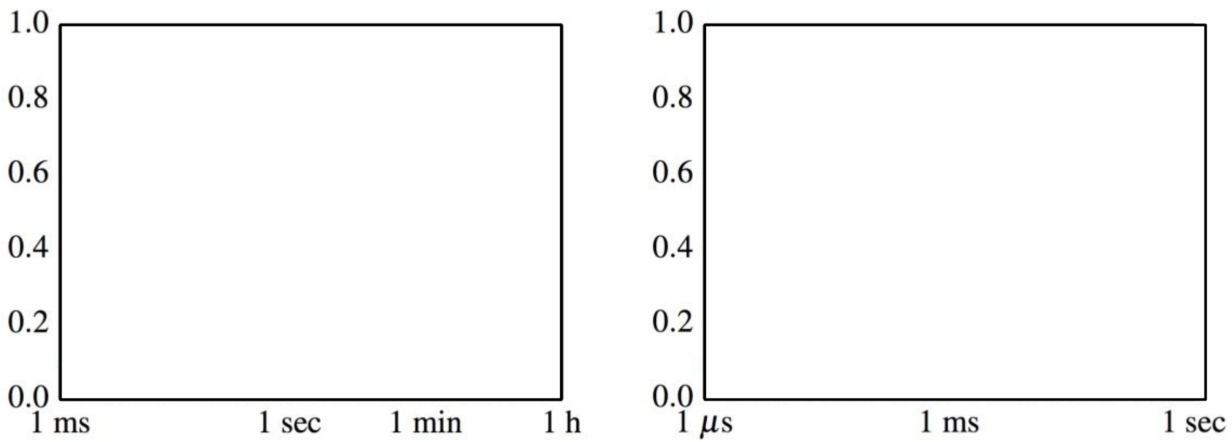
- Streaming arrivals: all workers participate in computation and receive input in parallel
- Ingestion: buffer-and-reorder
  Stash arriving records, wait fixed interval (*slack*) and sort
- Data exchange: re-partition by session ID; does not imply any logical barrier between shuffle and computation phases
- Time granularity (*epoch*) impacts execution efficiency and progress traffic





Input stream
Re-order buffer
Partitioning
Session window
Online analytics
Logical dataflow
I
B
hash
S
A
Physical deployment
L1
L2
L3
Ln-3
Ln-2
Ln-1
Ln
I1
I2
Im
S1
S2
Sm
A1
A2
Am
Worker 1
Worker 2
Worker m
Notifications
Epoch completions
(punctuations)
Distributed
progress tracking
Log servers

Characteristics of a production workload



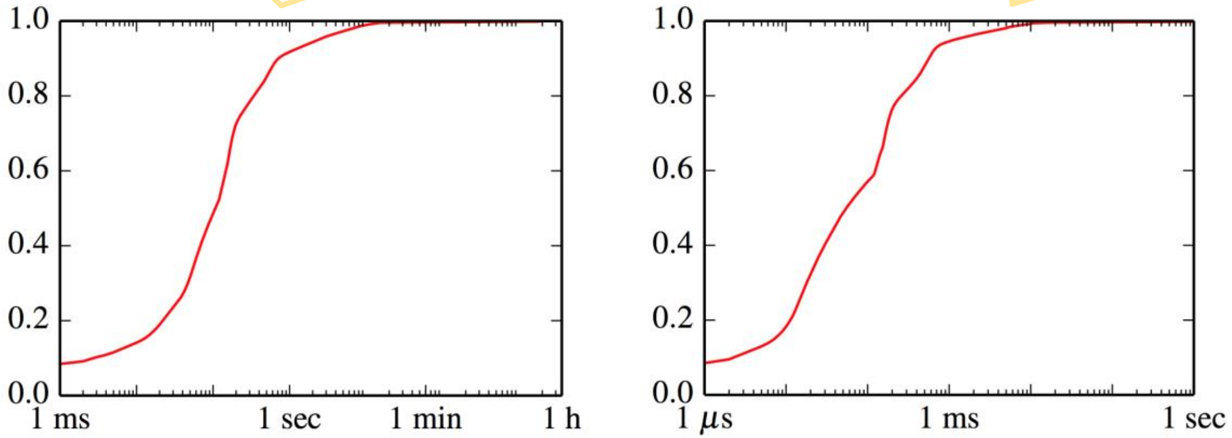
(a) Transaction Tree Duration (b) Maximum Inter-arrival Time

Figure 1. CDFs (Cumulative Distribution Functions) showing total duration of transaction trees and maximum interval between messages of a single session. Note: the x-axis is in logarithmic scale.

Characteristics of a production workload

~95% of root transactions are short-lived with total time span of ≤ 2 seconds

Only 0.24% of root transactions are dormant for more than a minute



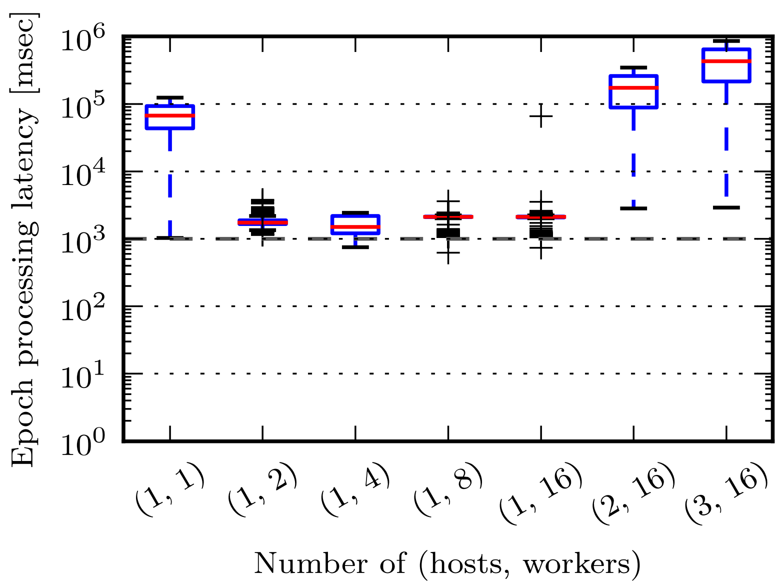
(a) Transaction Tree Duration (b) Maximum Inter-arrival Time

Figure 1. CDFs (Cumulative Distribution Functions) showing total duration of transaction trees and maximum interval between messages of a single session. Note: the x-axis is in logarithmic scale.

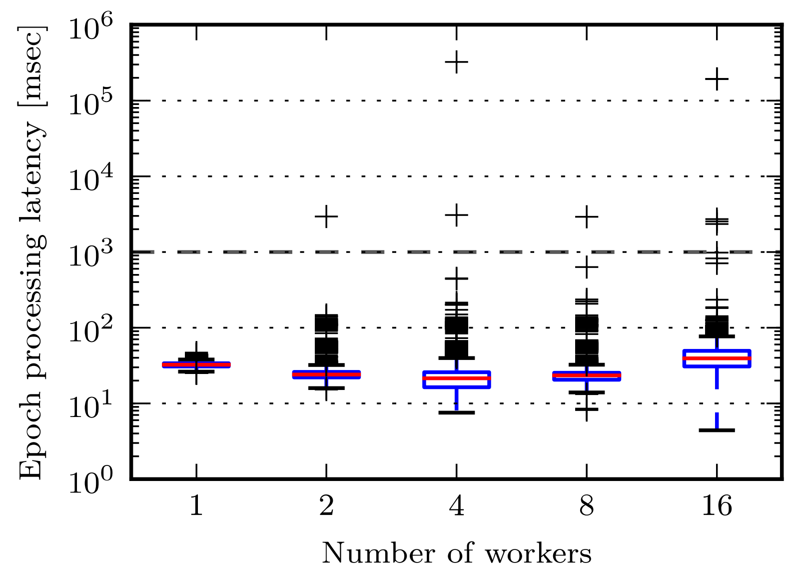
Real-time results with modest resource usage

Low latency: Flink spent on average 2.1 seconds (±1.1 s) for processing a single epoch of streaming logs whereas our system took only 26 milliseconds (±53 ms)

Peak resident set size remained stable and reached a peak of 203 MB while Flink's heap rose above 7.5 GB and required considerable tuning

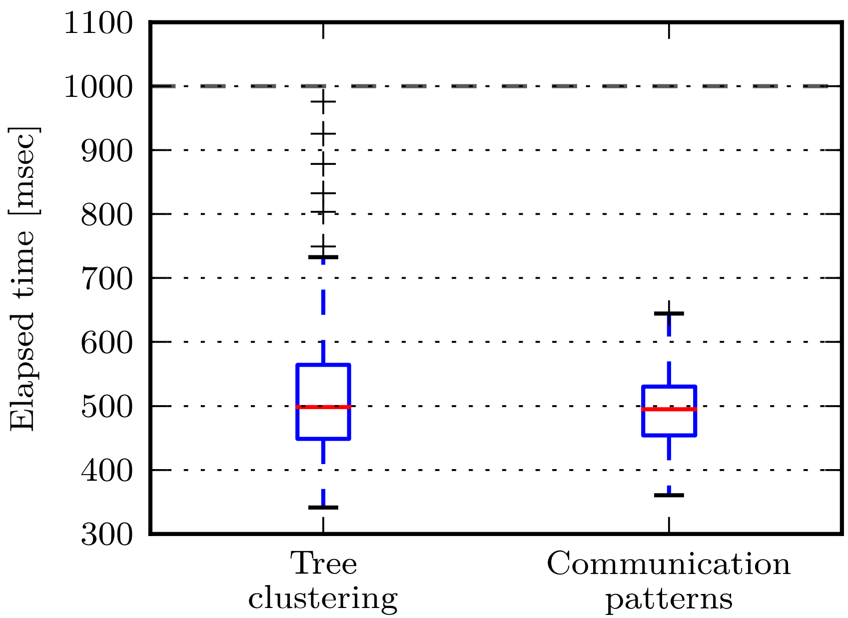


(a) Apache Flink



(b) Our system

Efficiency permits deeper analytics

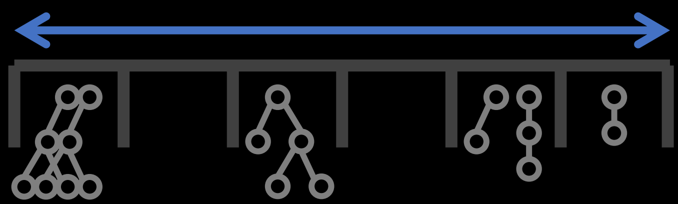


Exploiting a general framework permits a simple, concise implementation in 1770 lines of code while seamlessly integrating with management applications.

Composition of analytic tasks:

- Online trace tree clustering
- Service dependency extraction
- Inferring call-graph patterns

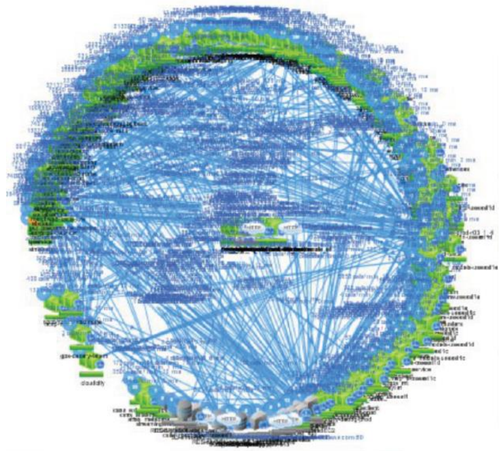
Summing up



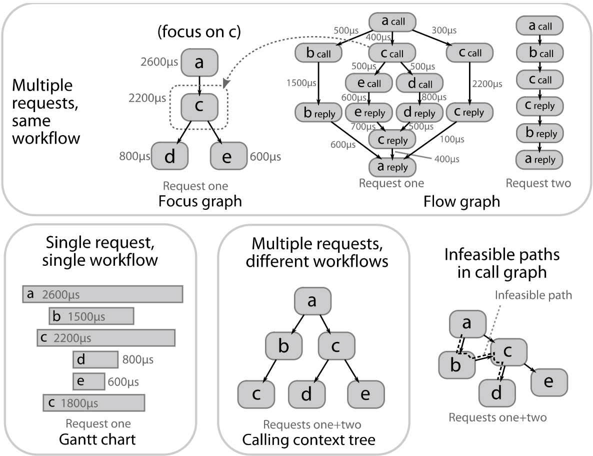
- Exploit comprehensive instrumentation already prevalent in data center applications
- Reconstruct user sessions, communication dependencies and trace tree clusters online
- Maintain and updates user sessions in real-time for an entire data center on a single commodity machine
- Processing latency in the range of tens of milliseconds

Questions? zchothia@inf.ethz.ch

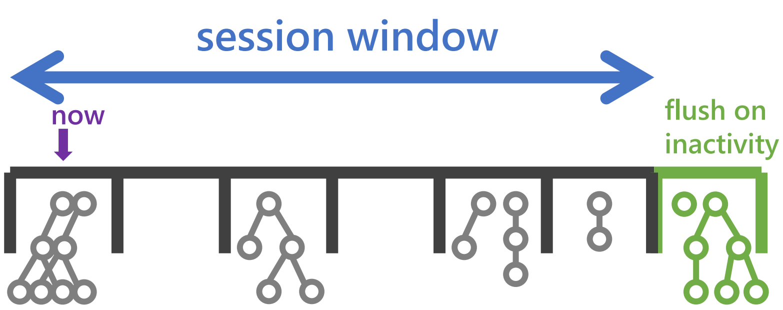
Online Reconstruction of Structural Information from Datacenter Logs



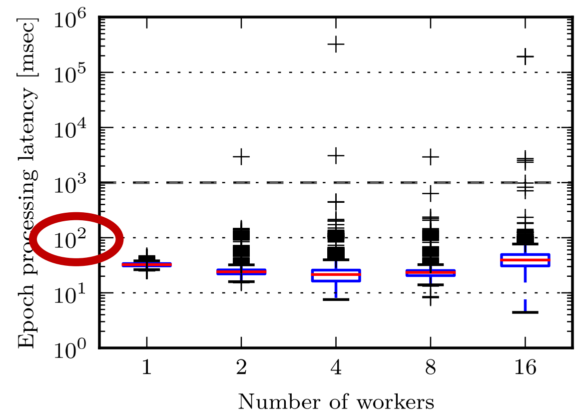
Problem: complex, interwoven interactions



User sessions, call graphs, transaction trees, timing charts shed insight



Approach: formulate sessionization as a Dataflow Operator



Timely Dataflow

Data parallel execution, gigabits per second, millions of transactions in real time