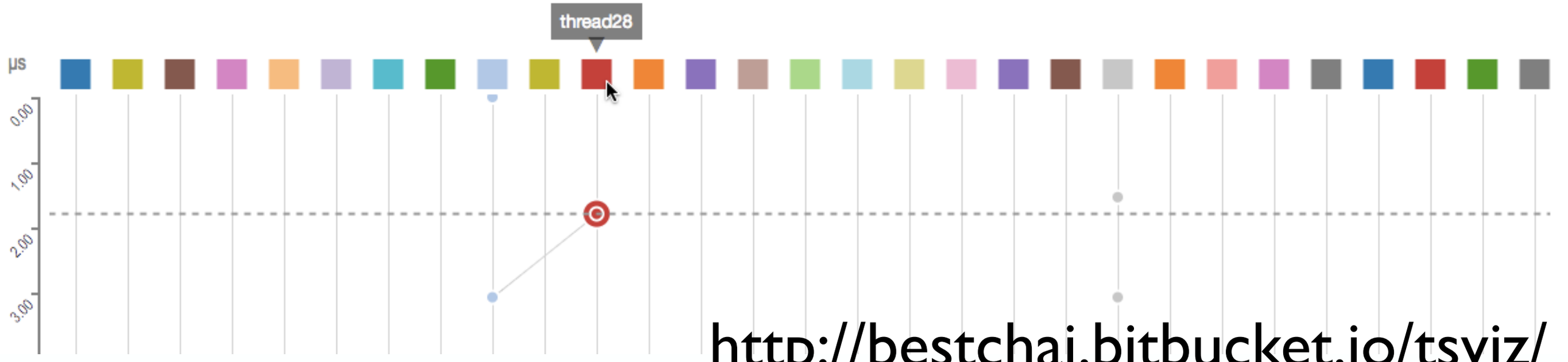
Studying multi-threaded behavior with TSViz



http://bestchai.bitbucket.io/tsviz/

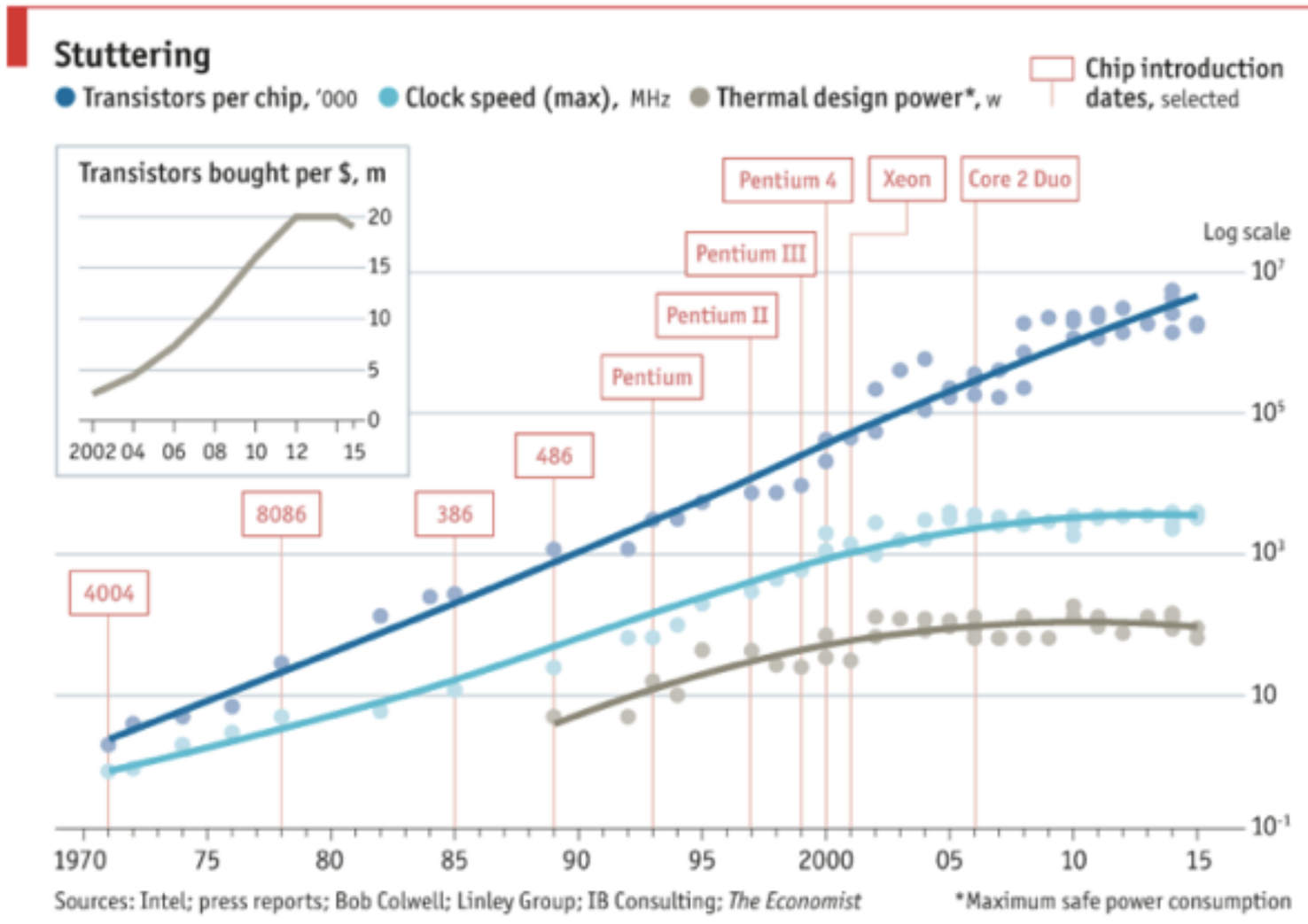
Matheus Nunes, Harjeet Lalh, Ashaya Sharma, Augustine Wong, Svetozar Miucin, Alexandra Fedorova, Ivan Beschastnikh

U. Federal de Minas Gerais
Brazil

U. of British Columbia
Canada

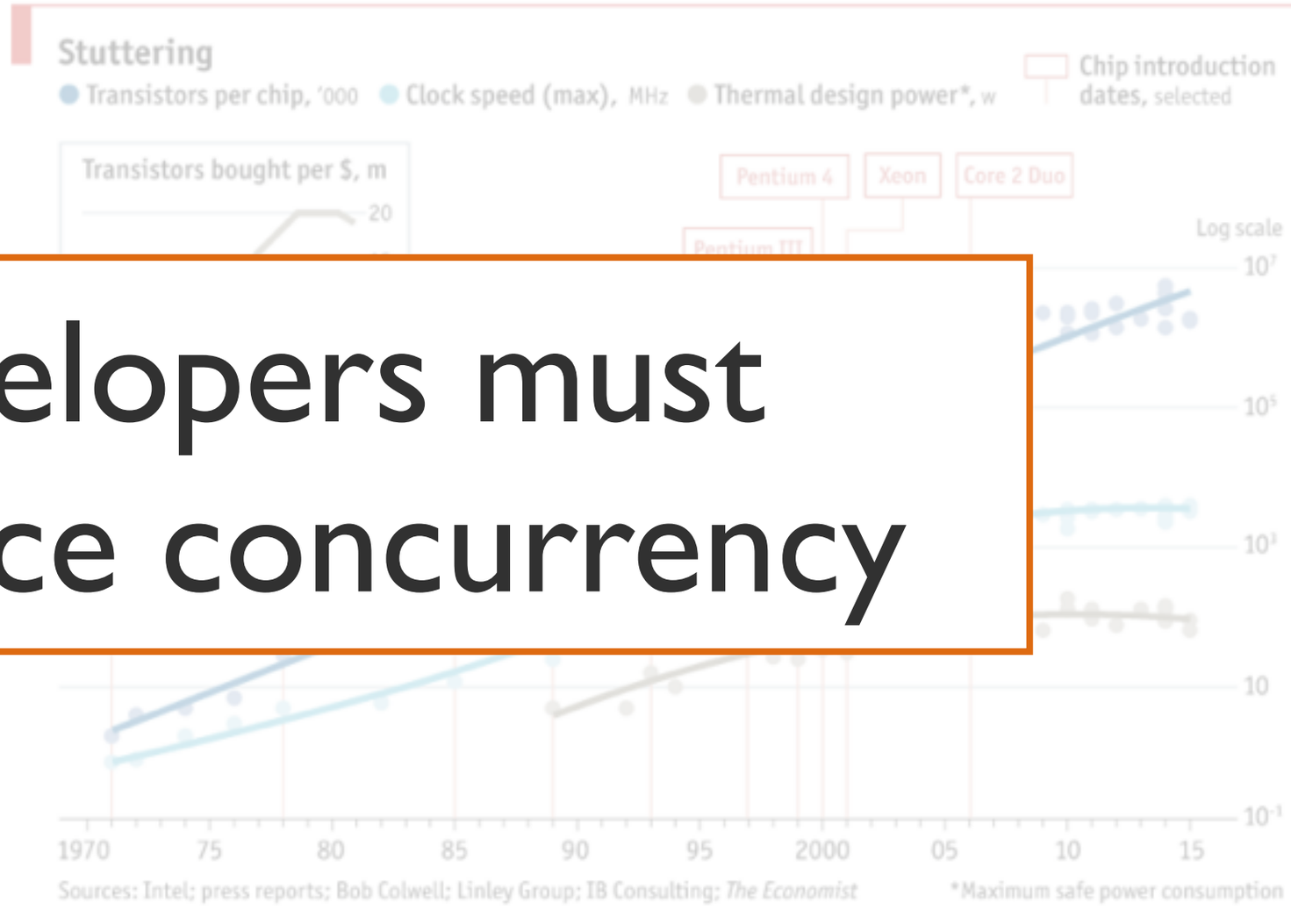
Concurrency is everywhere

- Performance:
  - Moore's law is dying out
  - iPhone 7 has 4 CPU cores and 6 GPU cores
- Coordination:
  - Today's apps depend on numerous services



Concurrency is everywhere

- Performance:
  - Moore's law is dying out
  - iPhone
    cores a
- Coordination:
  - Today's apps depend on numerous services



Developers must embrace concurrency

Concurrency complexities

- Multi-threading with shared state is the dominant model
  - Challenging to reason about order of events
  - Requires explicit concurrency control (e.g., locks)
- Few tools support concurrency comprehension
  - Commonly used tools: profilers/tracing tools
  - Many tools target distributed computing/systems
  - Most devs study logs, one per thread

[1] Debugging Distributed Systems: Challenges and options for validation and debugging, Beschastnikh et al. CACM 2016

[2] Combing the Communication Hairball: Visualizing Parallel Execution Traces using Logical Time. Isaacs et al. TVCG 2014

[3] Ordering Traces Logically to Identify Lateness in Message Passing Programs. Isaacs et al. TPDS 2016

Concurrency analysis in the small

Our view: understanding inter-thread interactions require understanding runtime behavior **in the small**

- A solution requires two pieces:

1. Instrumentation
   - e.g., LLVM-based tool called DINAMITE [1]
2. Interactive visualization of captured information
   - TSViz: a new tool based on ShiViz [2]

[1] End-to-end Memory Behavior Profiling with DINAMITE. Miucin et al. Tool demo at FSE 2016

[2] Debugging Distributed Systems: Challenges and options for validation and debugging, Beschastnikh et al. CACM 2016

Concurrency analysis in the small

Our view: understanding inter-thread interactions require understanding runtime behavior **in the small**

- A solution requires two pieces:

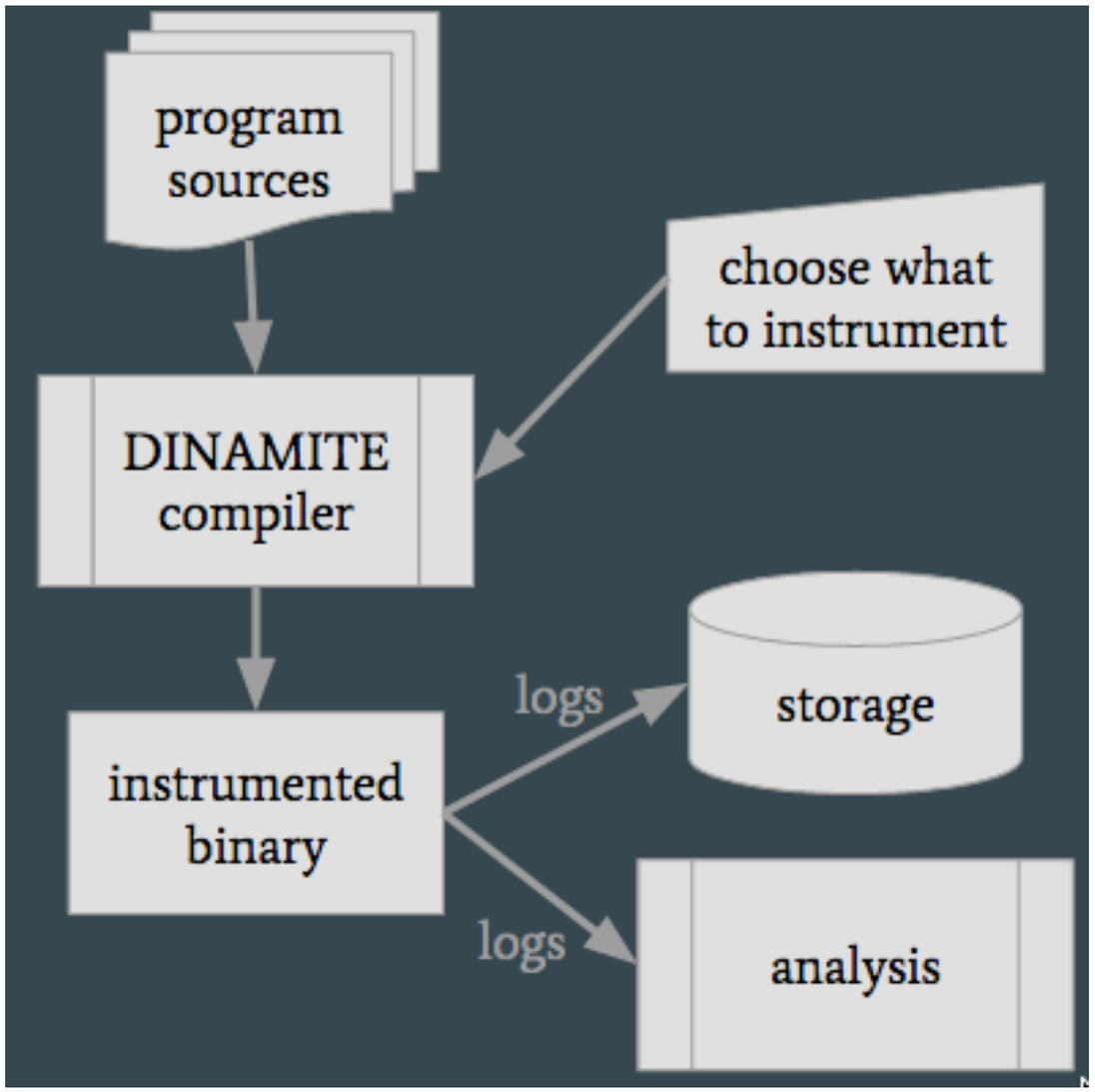
1. Instrumentation
   - e.g., LLVM-based tool called DINAMITE [1]
2. Interactive visualization of captured information
   - **TSViz**: a new tool based on ShiViz [2]

Focus of this talk

[1] End-to-end Memory Behavior Profiling with DINAMITE. Miucin et al. Tool demo at FSE 2016

[2] Debugging Distributed Systems: Challenges and options for validation and debugging, Beschastnikh et al. CACM 2016

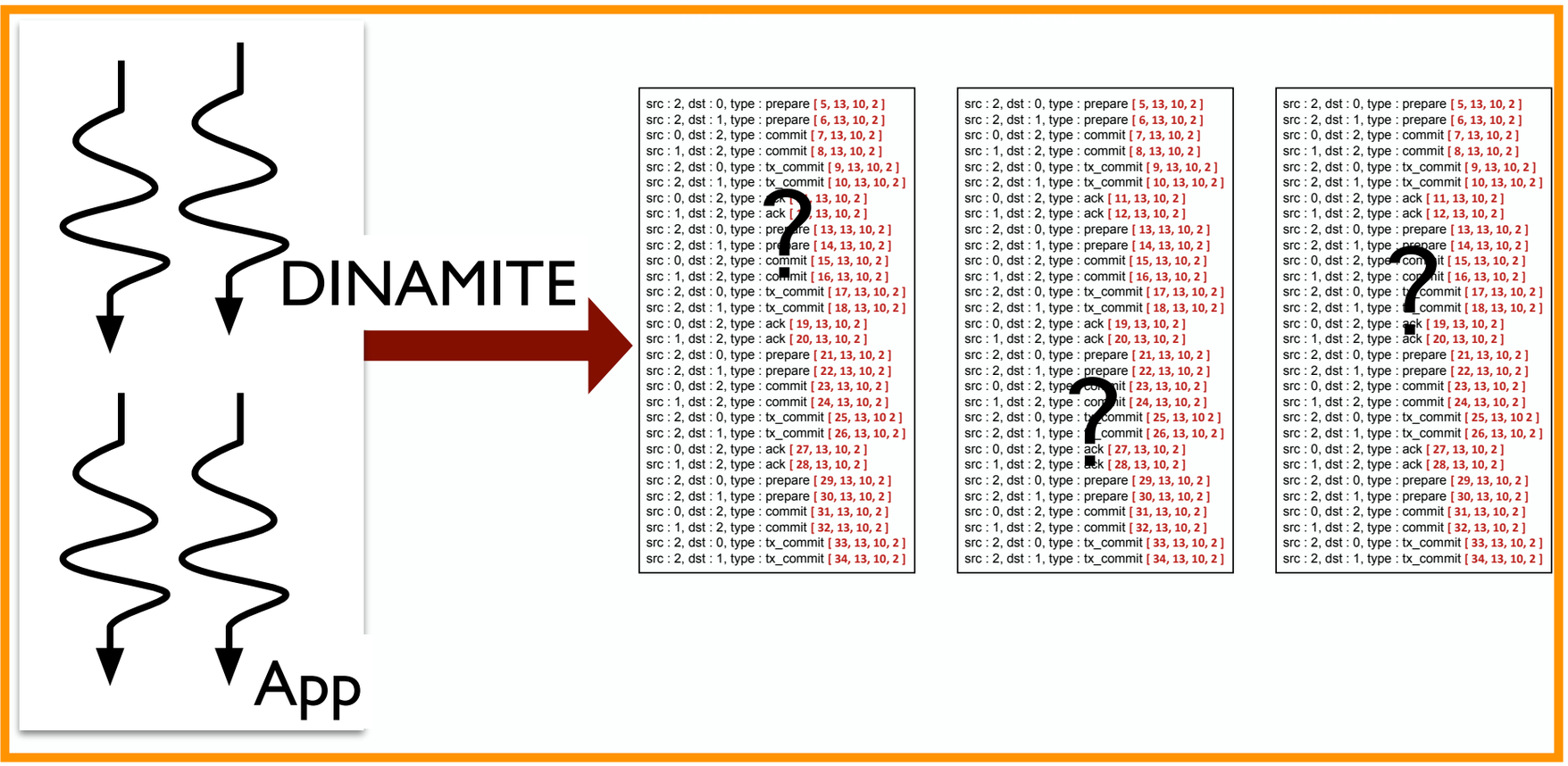
DINAMITE: LLVM-based tracer



- Can instrument functions, mutexes, memory allocations and accesses…
- Instrument what you care about (or use reasonable defaults)
- Possibility of analysis on the fly
- Controllable overhead between 10% and 30x
- Get all the debug information in logs! (even at -O3)
- People in our lab use it daily for performance debugging

https://dinamite-toolkit.github.io/

Runtime instrumentation



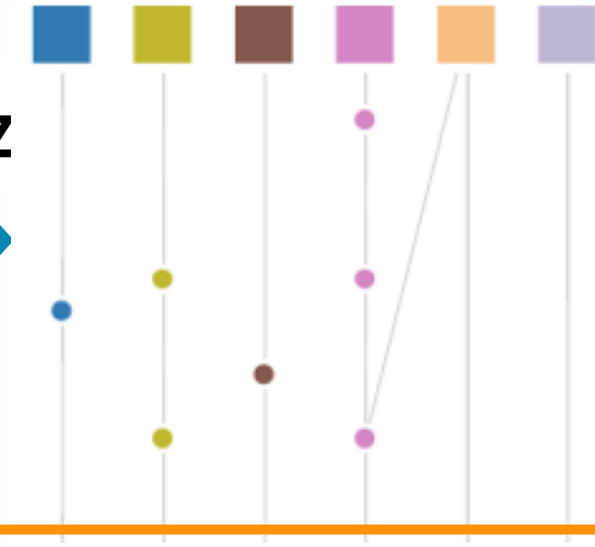
Runtime log visualization

src : 2, dst : 0, type : prepare [5, 13, 10, 2]
src : 2, dst : 1, type : prepare [6, 13, 10, 2]
src : 0, dst : 2, type : commit [7, 13, 10, 2]
src : 1, dst : 2, type : commit [8, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [9, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [10, 13, 10, 2]
src : 0, dst : 2, type : ack [11, 13, 10, 2]
src : 1, dst : 2, type : ack [12, 13, 10, 2]
src : 2, dst : 0, type : prepare [13, 13, 10, 2]
src : 2, dst : 1, type : prepare [14, 13, 10, 2]
src : 0, dst : 2, type : commit [15, 13, 10, 2]
src : 1, dst : 2, type : commit [16, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [17, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [18, 13, 10, 2]
src : 0, dst : 2, type : ack [19, 13, 10, 2]
src : 1, dst : 2, type : ack [20, 13, 10, 2]
src : 2, dst : 0, type : prepare [21, 13, 10, 2]
src : 2, dst : 1, type : prepare [22, 13, 10, 2]
src : 0, dst : 2, type : commit [23, 13, 10, 2]
src : 1, dst : 2, type : commit [24, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [25, 13, 10 2]
src : 2, dst : 1, type : tx_commit [26, 13, 10, 2]
src : 0, dst : 2, type : ack [27, 13, 10, 2]
src : 1, dst : 2, type : ack [28, 13, 10, 2]
src : 2, dst : 0, type : prepare [29, 13, 10, 2]
src : 2, dst : 1, type : prepare [30, 13, 10, 2]
src : 0, dst : 2, type : commit [31, 13, 10, 2]
src : 1, dst : 2, type : commit [32, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [33, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [34, 13, 10, 2]

src : 2, dst : 0, type : prepare [5, 13, 10, 2]
src : 2, dst : 1, type : prepare [6, 13, 10, 2]
src : 0, dst : 2, type : commit [7, 13, 10, 2]
src : 1, dst : 2, type : commit [8, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [9, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [10, 13, 10, 2]
src : 0, dst : 2, type : ack [11, 13, 10, 2]
src : 1, dst : 2, type : ack [12, 13, 10, 2]
src : 2, dst : 0, type : prepare [13, 13, 10, 2]
src : 2, dst : 1, type : prepare [14, 13, 10, 2]
src : 0, dst : 2, type : commit [15, 13, 10, 2]
src : 1, dst : 2, type : commit [16, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [17, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [18, 13, 10, 2]
src : 0, dst : 2, type : ack [19, 13, 10, 2]
src : 1, dst : 2, type : ack [20, 13, 10, 2]
src : 2, dst : 0, type : prepare [21, 13, 10, 2]
src : 2, dst : 1, type : prepare [22, 13, 10, 2]
src : 0, dst : 2, type : commit [23, 13, 10, 2]
src : 1, dst : 2, type : commit [24, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [25, 13, 10 2]
src : 2, dst : 1, type : tx_commit [26, 13, 10, 2]
src : 0, dst : 2, type : ack [27, 13, 10, 2]
src : 1, dst : 2, type : ack [28, 13, 10, 2]
src : 2, dst : 0, type : prepare [29, 13, 10, 2]
src : 2, dst : 1, type : prepare [30, 13, 10, 2]
src : 0, dst : 2, type : commit [31, 13, 10, 2]
src : 1, dst : 2, type : commit [32, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [33, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [34, 13, 10, 2]

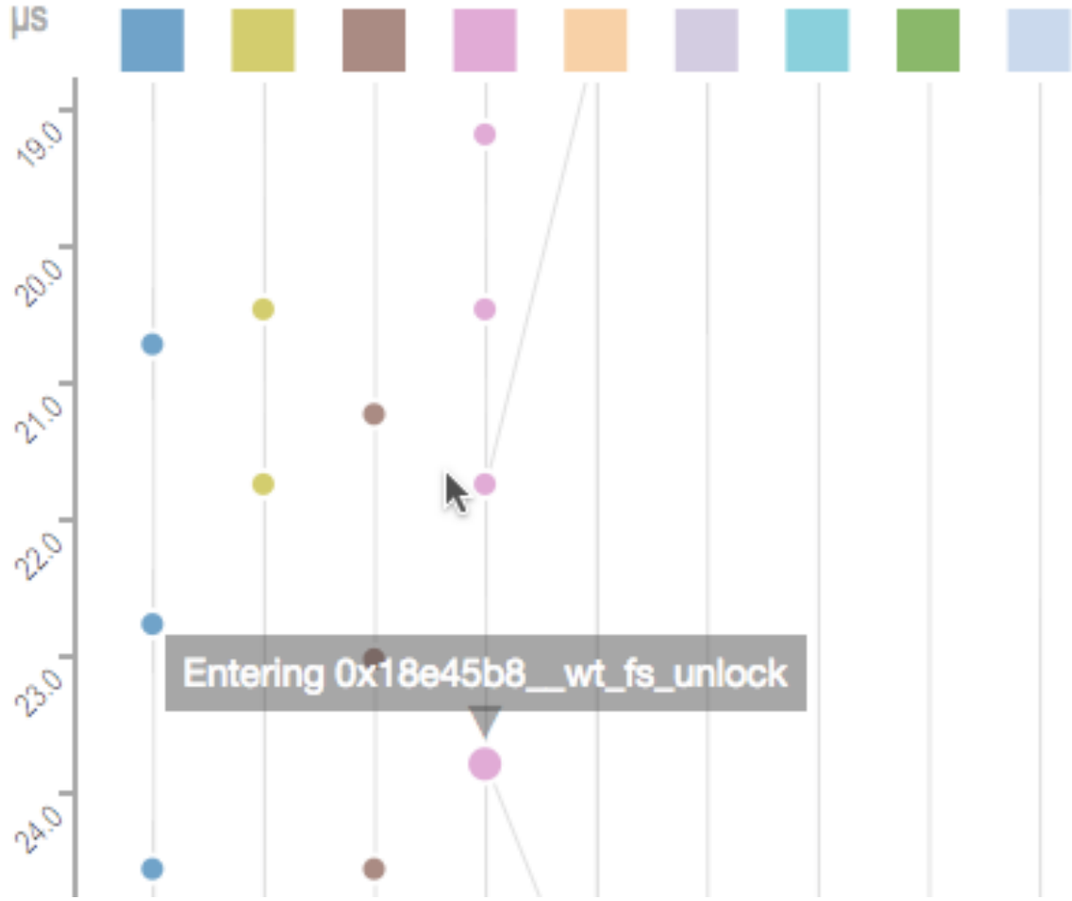
src : 2, dst : 0, type : prepare [5, 13, 10, 2]
src : 2, dst : 1, type : prepare [6, 13, 10, 2]
src : 0, dst : 2, type : commit [7, 13, 10, 2]
src : 1, dst : 2, type : commit [8, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [9, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [10, 13, 10, 2]
src : 0, dst : 2, type : ack [11, 13, 10, 2]
src : 1, dst : 2, type : ack [12, 13, 10, 2]
src : 2, dst : 0, type : prepare [13, 13, 10, 2]
src : 2, dst : 1, type : prepare [14, 13, 10, 2]
src : 0, dst : 2, type : commit [15, 13, 10, 2]
src : 1, dst : 2, type : commit [16, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [17, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [18, 13, 10, 2]
src : 0, dst : 2, type : ack [19, 13, 10, 2]
src : 1, dst : 2, type : ack [20, 13, 10, 2]
src : 2, dst : 0, type : prepare [21, 13, 10, 2]
src : 2, dst : 1, type : prepare [22, 13, 10, 2]
src : 0, dst : 2, type : commit [23, 13, 10, 2]
src : 1, dst : 2, type : commit [24, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [25, 13, 10 2]
src : 2, dst : 1, type : tx_commit [26, 13, 10, 2]
src : 0, dst : 2, type : ack [27, 13, 10, 2]
src : 1, dst : 2, type : ack [28, 13, 10, 2]
src : 2, dst : 0, type : prepare [29, 13, 10, 2]
src : 2, dst : 1, type : prepare [30, 13, 10, 2]
src : 0, dst : 2, type : commit [31, 13, 10, 2]
src : 1, dst : 2, type : commit [32, 13, 10, 2]
src : 2, dst : 0, type : tx_commit [33, 13, 10, 2]
src : 2, dst : 1, type : tx_commit [34, 13, 10, 2]

TSViz



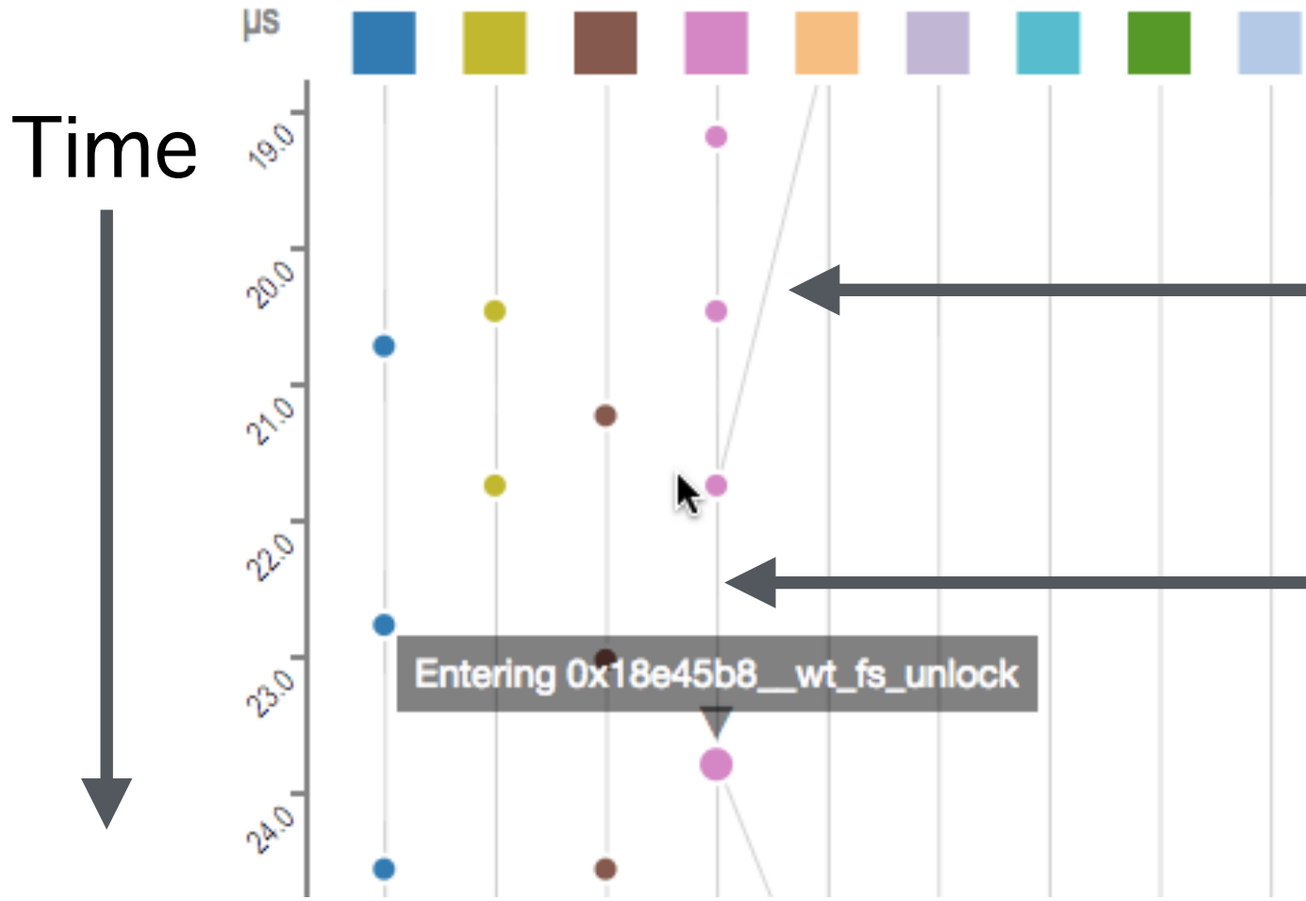
TSViz overview

- Takes a log with physical timestamps and logical timestamps as input, outputs a thread interaction graph
- Interactive tool to explore the physical and logical orderings
  - Show both orderings
  - To-scale visualization; zooming
  - Search queries
- Target population: concurrent system developers
- Client-side browser impl.



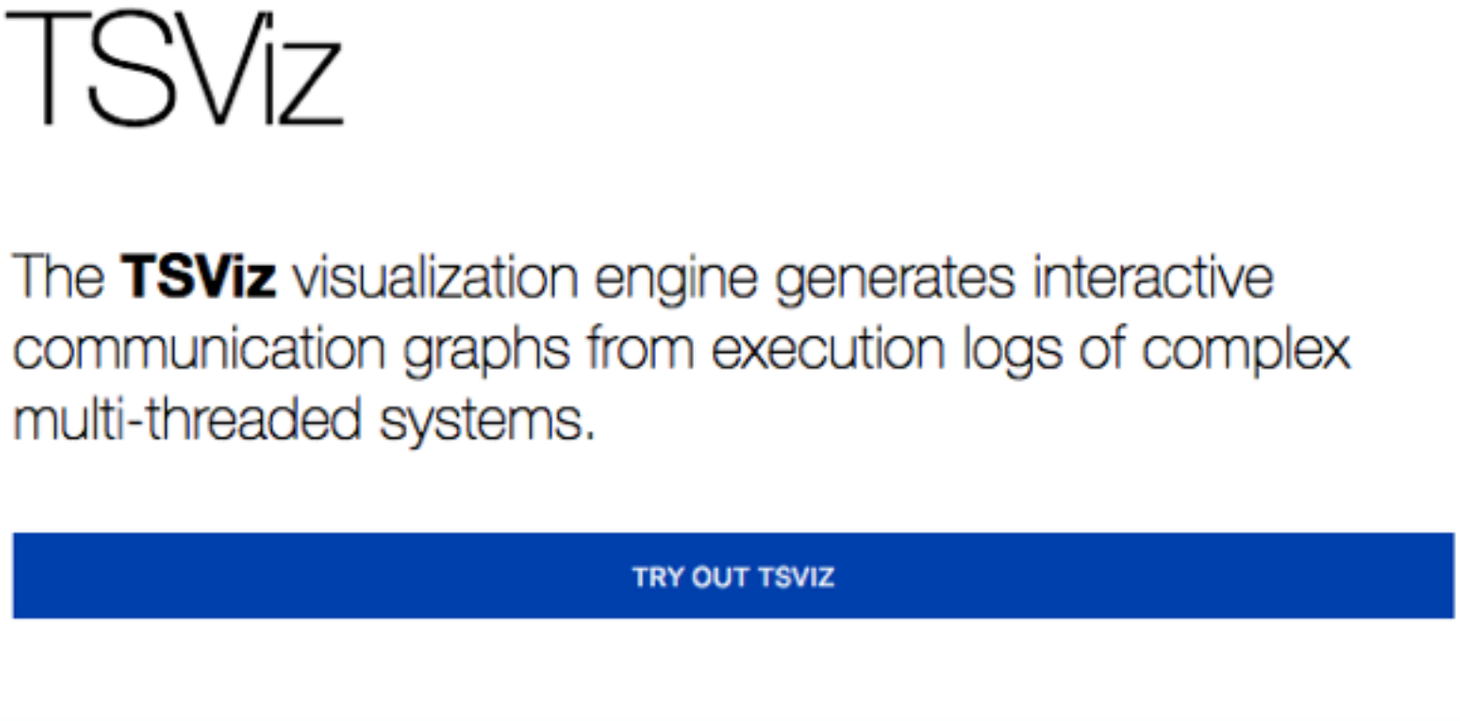
TSViz visual abstractions

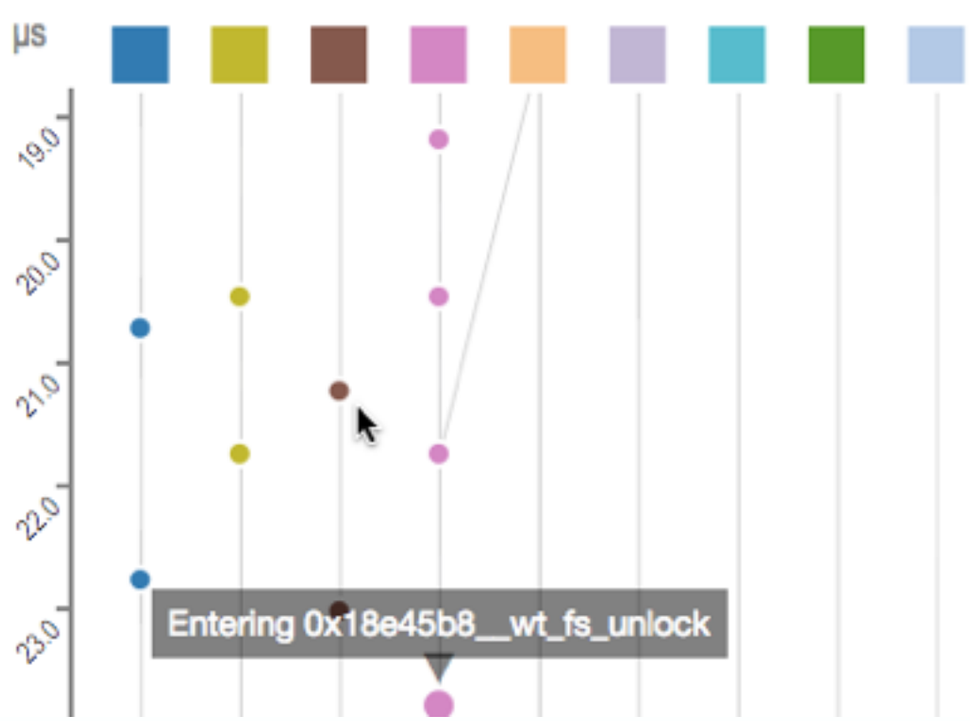
- Squares represent threads
- Circles represent thread events
- Lines between events represent orderings



Demo on WiredTiger k-v store [1]

Public deployment:
http://bestchai.bitbucket.io/tsviz/





[1] http://www.wiredtiger.com

Why does my concurrent system behave in a certain manner?

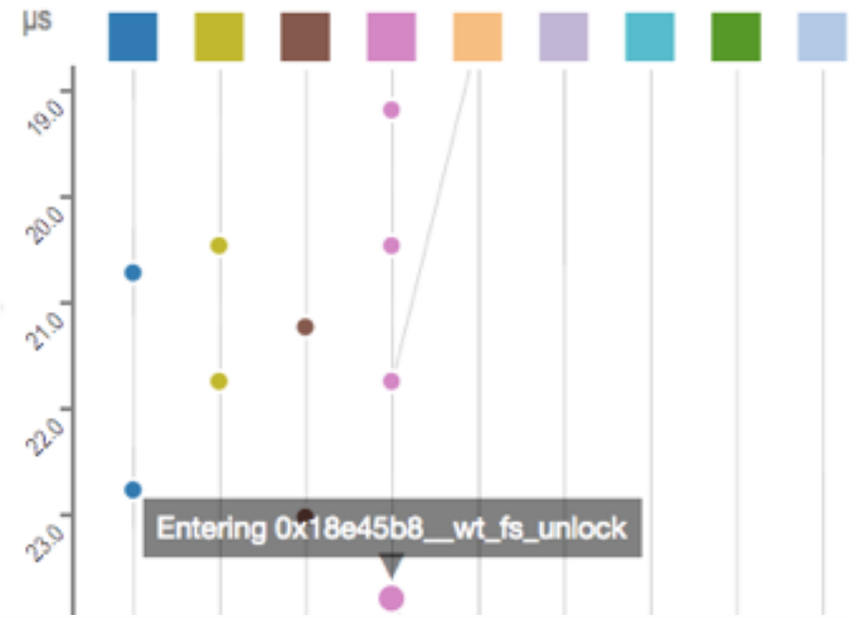
Dynamic analysis

Approach: instrument and analyze

Events State Visualization

★ TSViz : visualize concurrent executions

- Understand ordering of events
- Query for patterns; compare executions



http://bestchai.bitbucket.io/tsviz/