# File System Trace and Replay

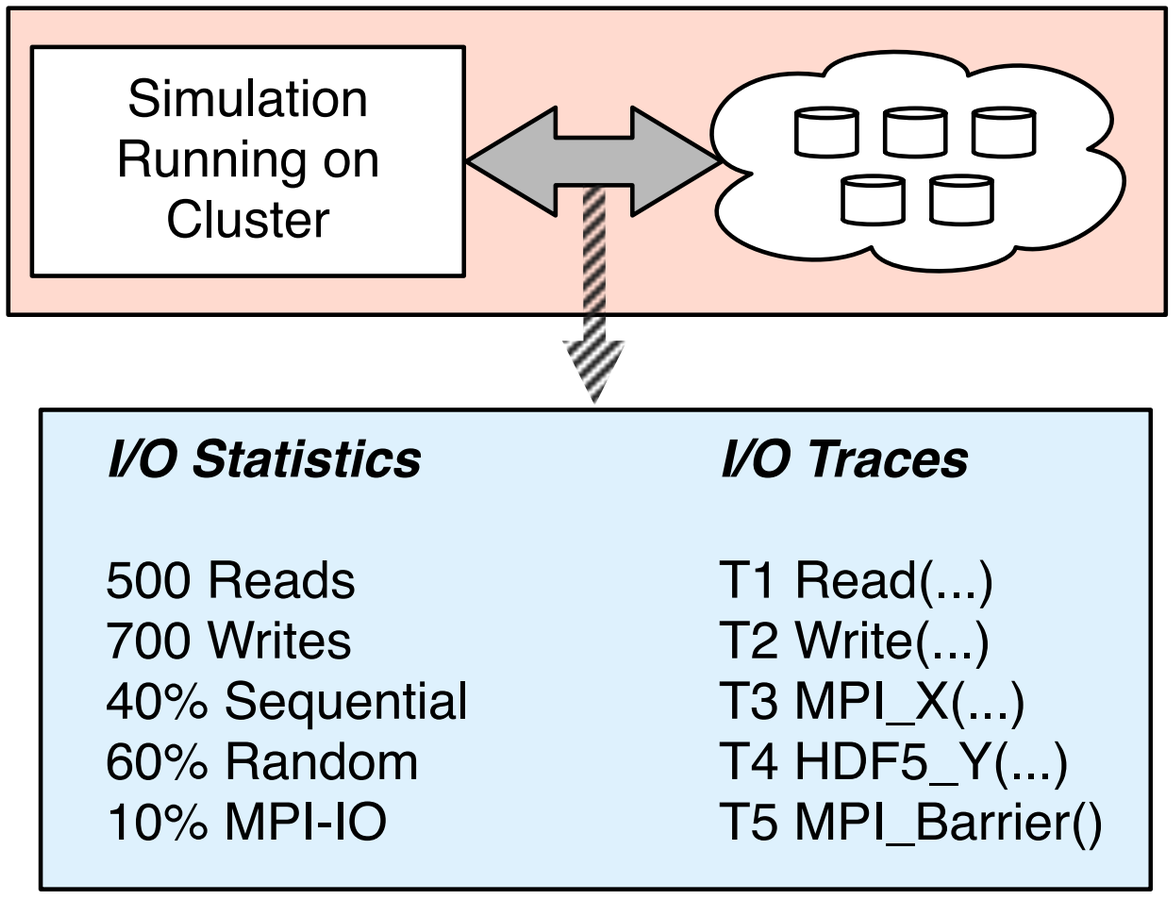
Noah Watkins / UC Santa Cruz

James Nunez, Meghan Wingate, John Bent / LANL

Advisors: Scott Brandt, Carlos Maltzahn

# What is tracing?

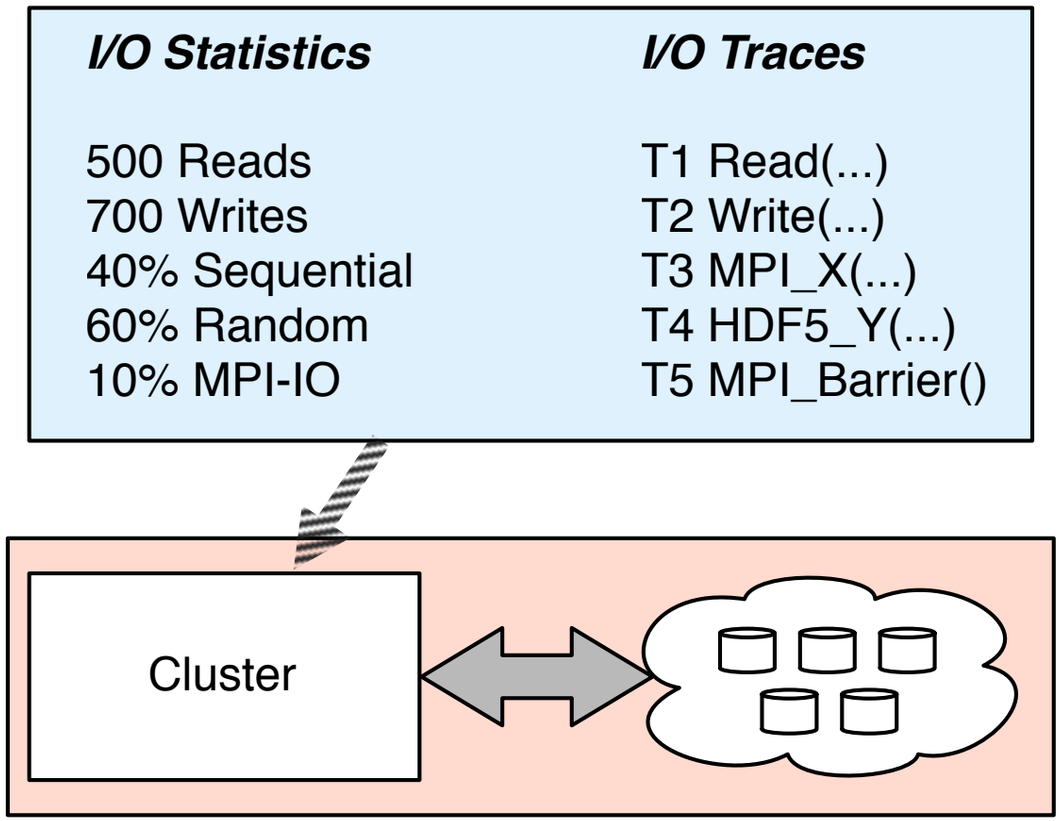
- Creation of a signature describing the execution of a system:
  - This talk is about file system traces
  - Ordered sequence of events
  - Statistical aggregation



# What is replay?

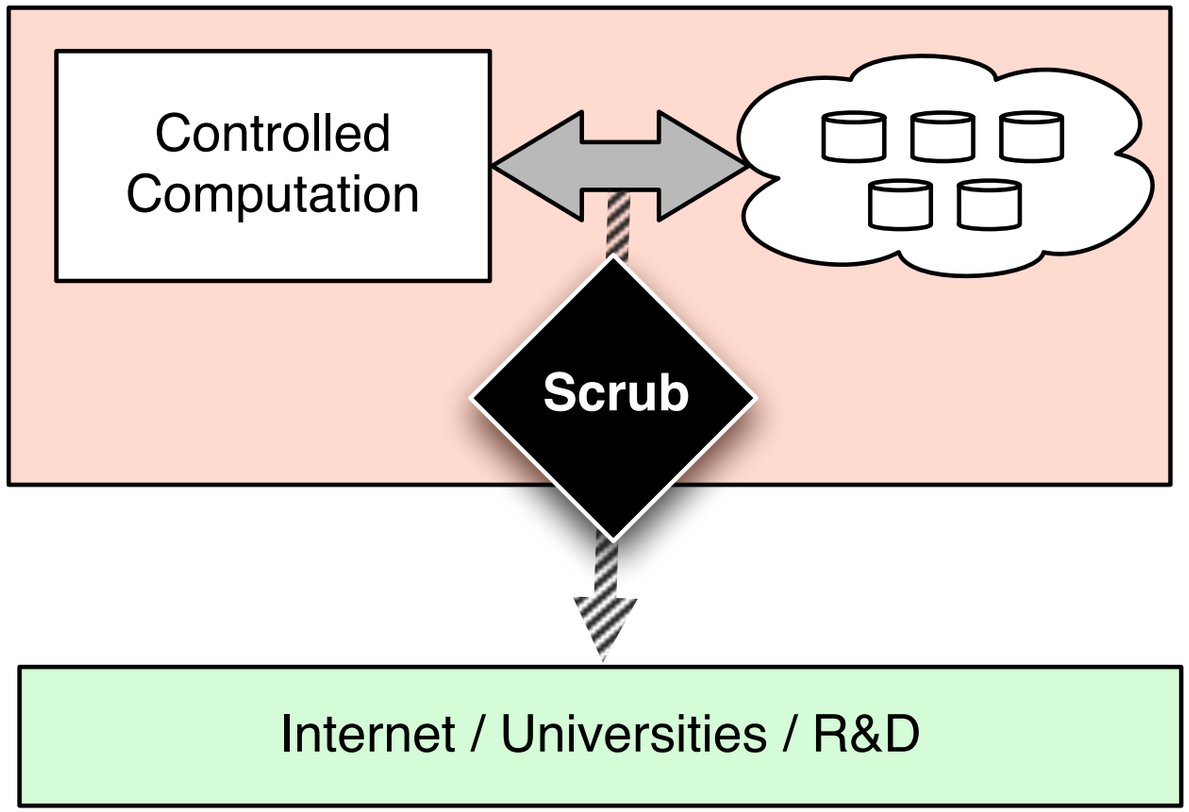
- Trace replay is the reproduction of a traced workload having only knowledge of a trace.

- Replay fidelity is how accurately the original workload is reproduced.

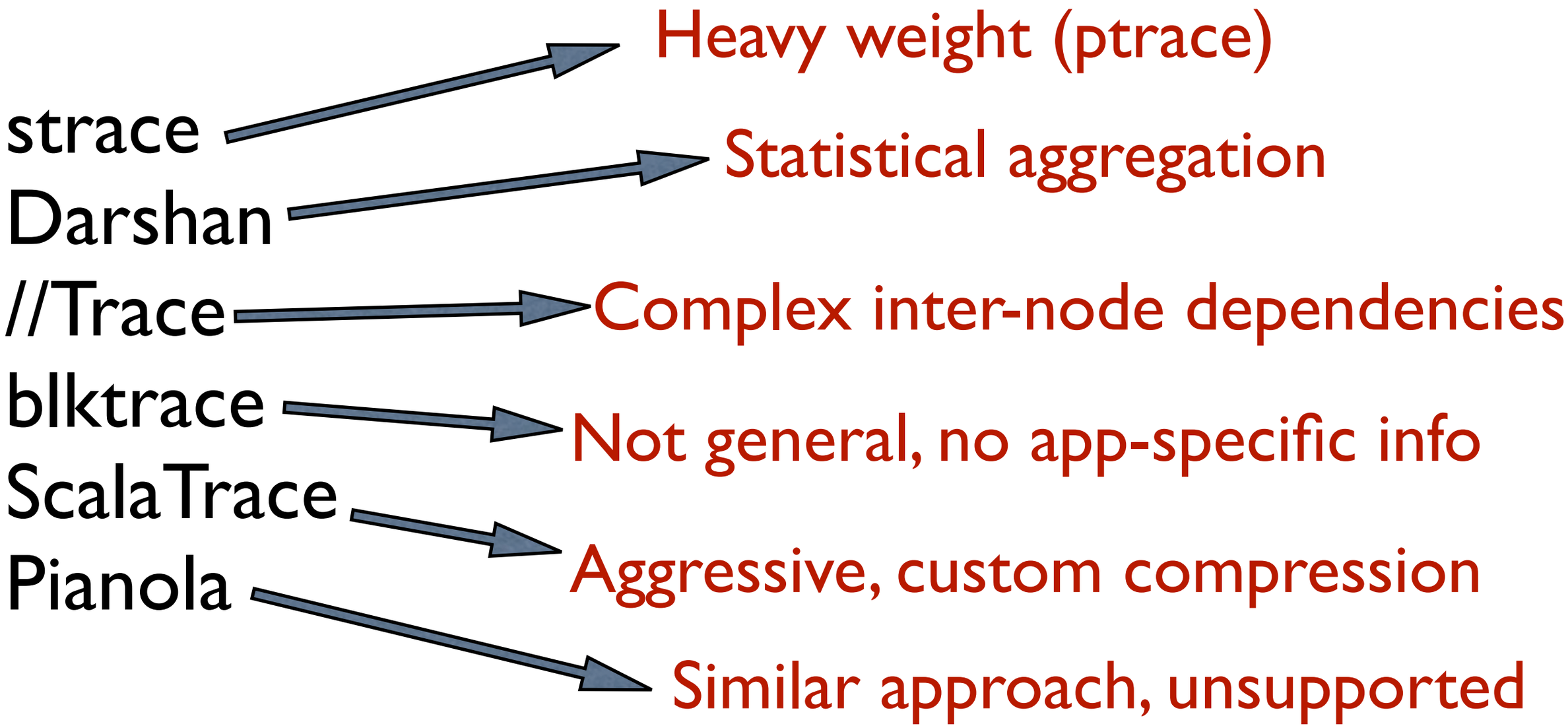


# Why trace and replay?

- HPC at LANL pushes the limits of storage systems
- R&D collaborators
- Classified and controlled applications cannot be distributed
- Distribute traces, not apps



# Previous work



# Tracing goals

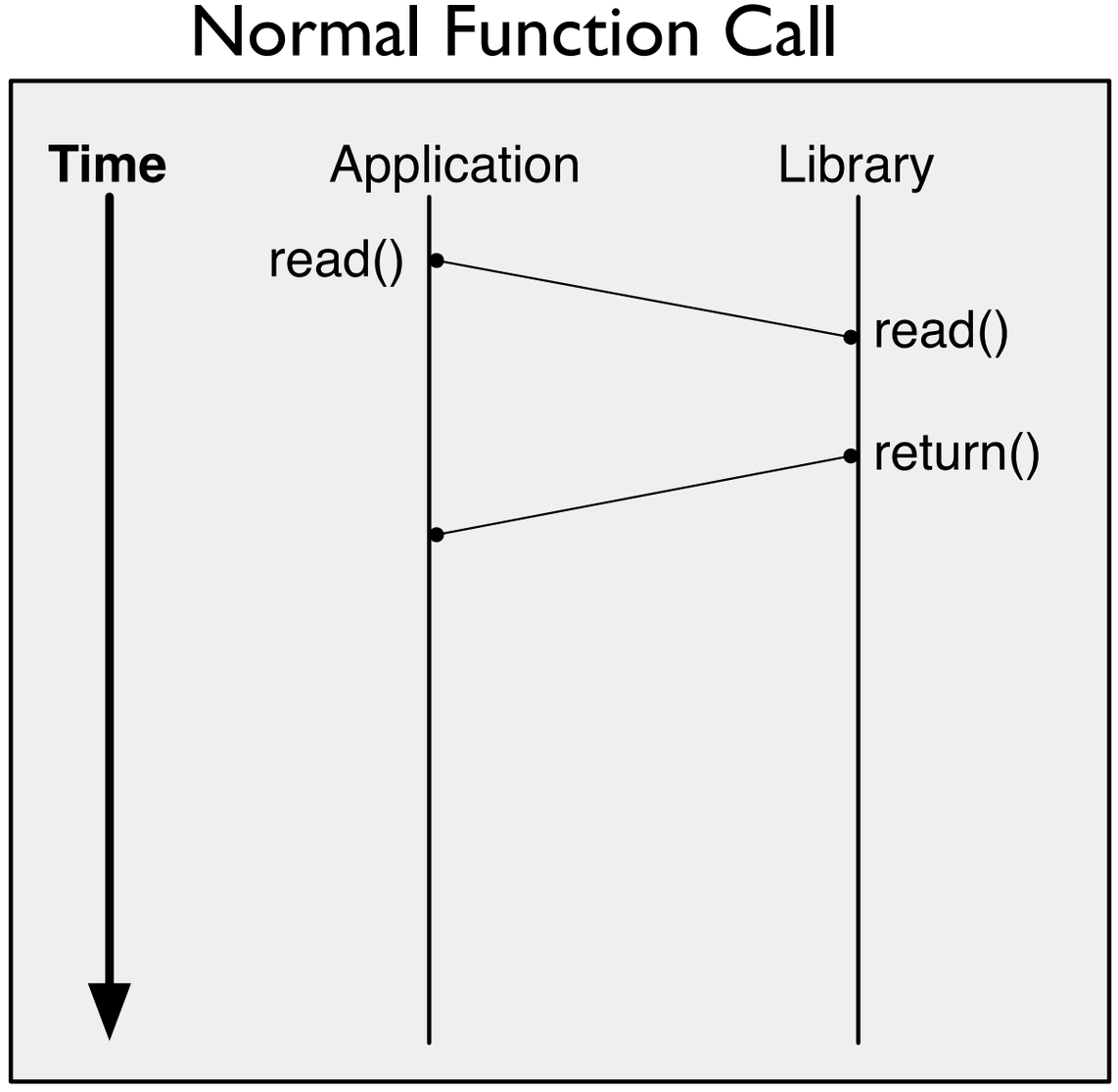
- No code changes
- Entire software stack
- Low-overhead logging
- Easily integrated with existing applications

# The observer effect

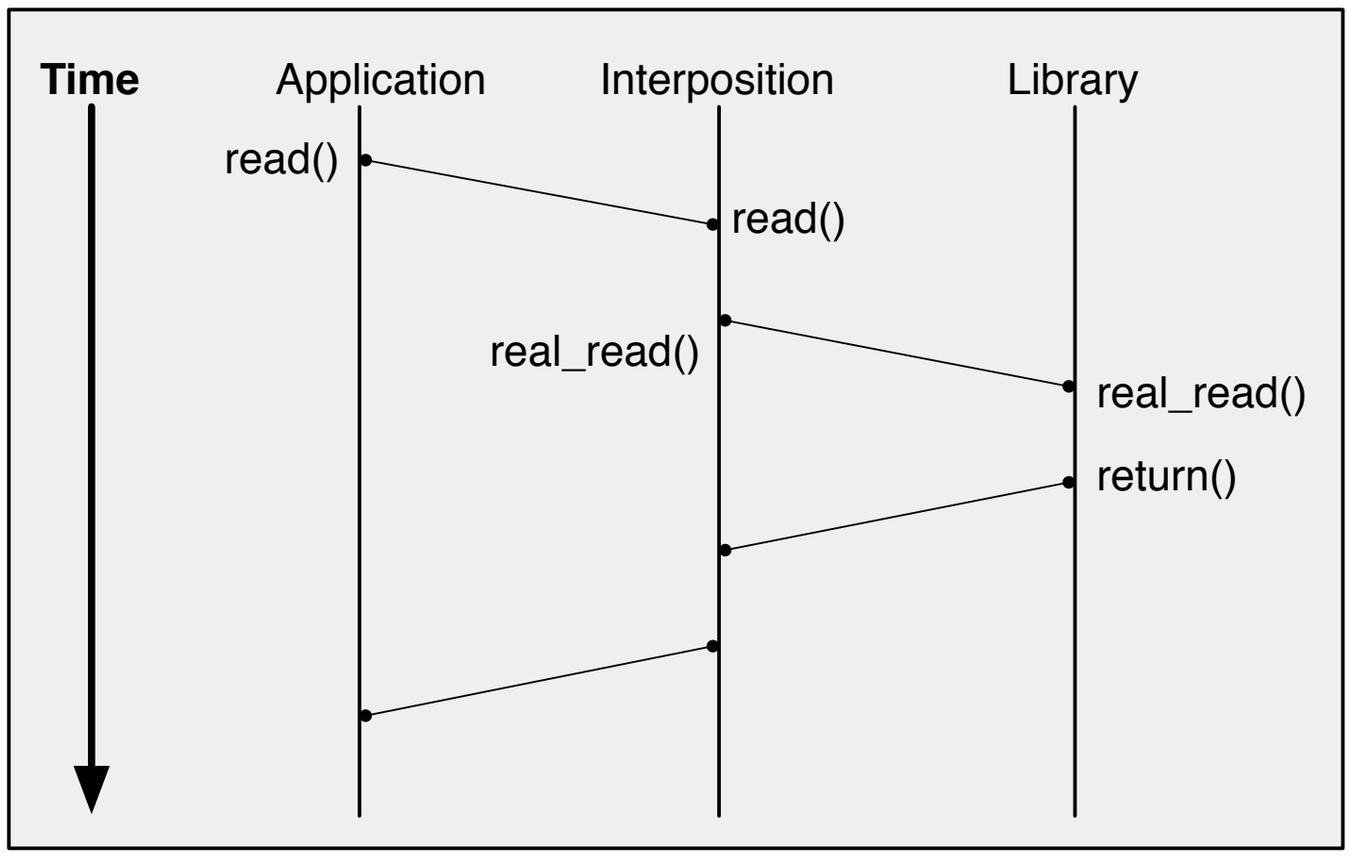
- A normal function call incurs virtually no unnecessary overhead

- In order to record function calls, they must be observed and saved

- Minimizing this overhead is important



# The observer effect




- *The act of observing an application's execution alters the application's behavior*
  - Record a timestamp
  - Encode data
  - Buffer/write data
  - Resource contention

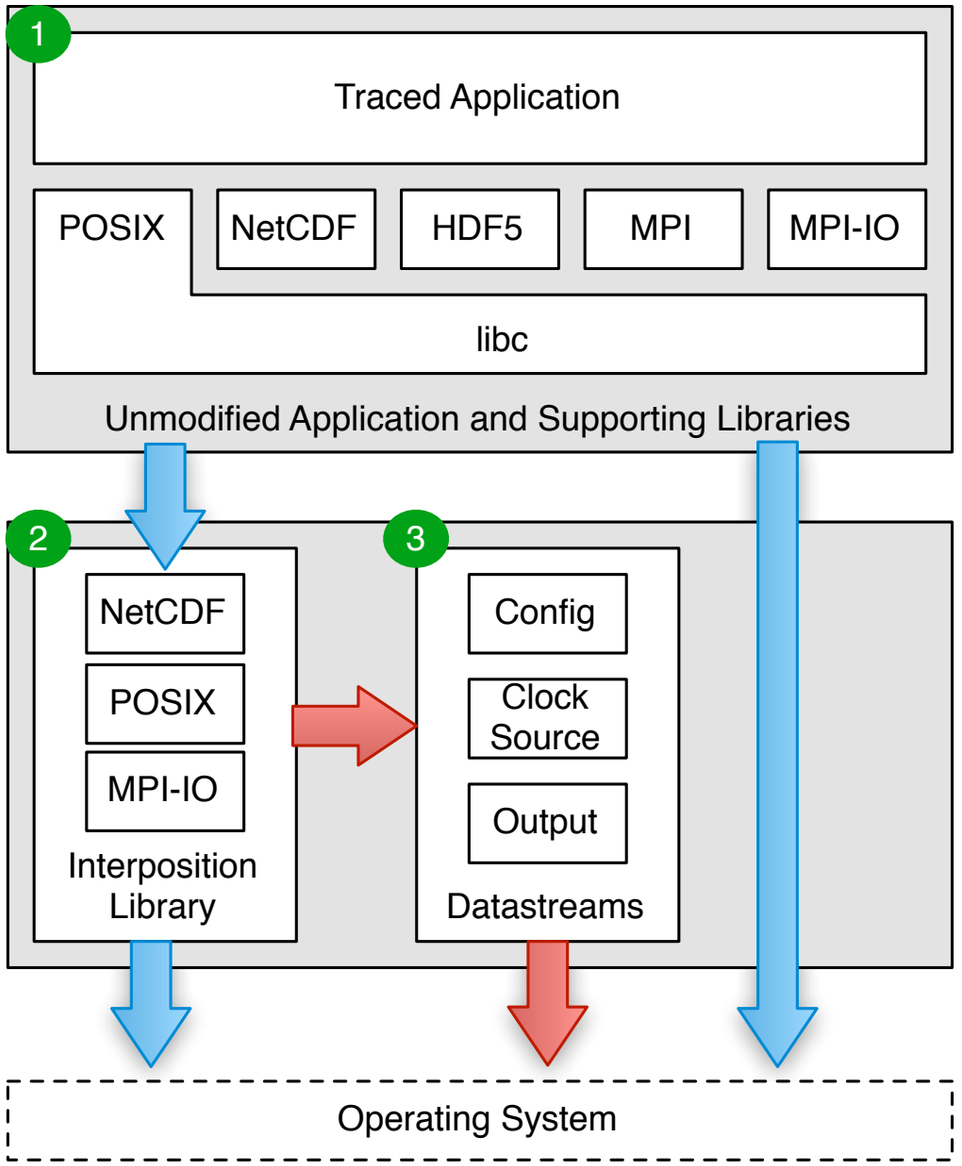
# Our approach to tracing

- Identify, measure, and minimize sources of overhead

1. Interposition
2. Timekeeping
3. Logging
4. Resource

# Tracing architecture

- Traced application execute as normal (1)
- Interposition library forwards events, and executes original function (2)
- Datastreams[1] library buffers and schedules trace events (3)




1. **http://www.ittc.ku.edu/kusp**

# Evaluation

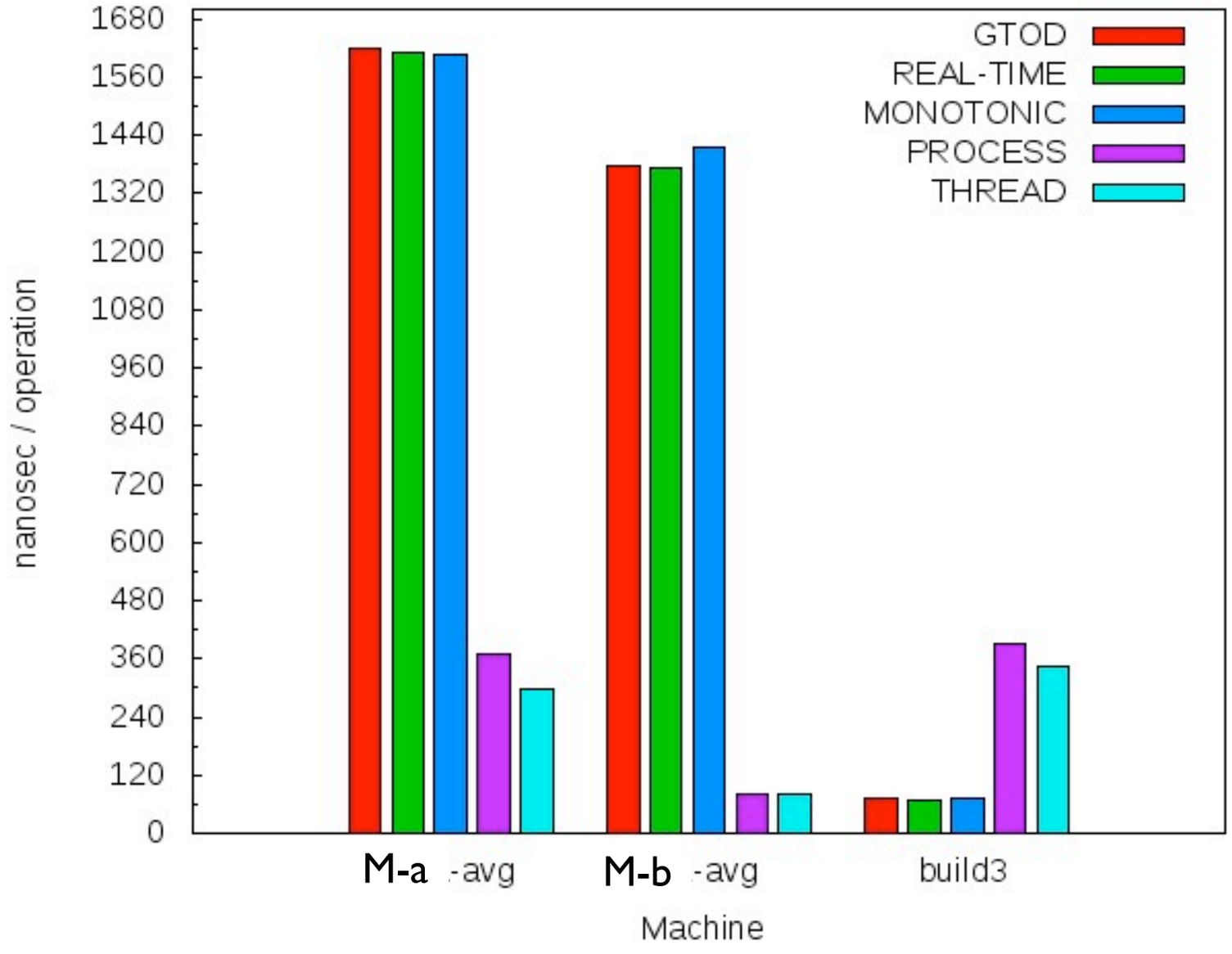
# Interposition cost and extensibility

- Link-time function wrapping provides *minimal interposition cost.*
  - *read(...)* --> *__wrap_read(...)*
  - __wrap_read(...)
    - logs event
    - calls *__real_read(...)*

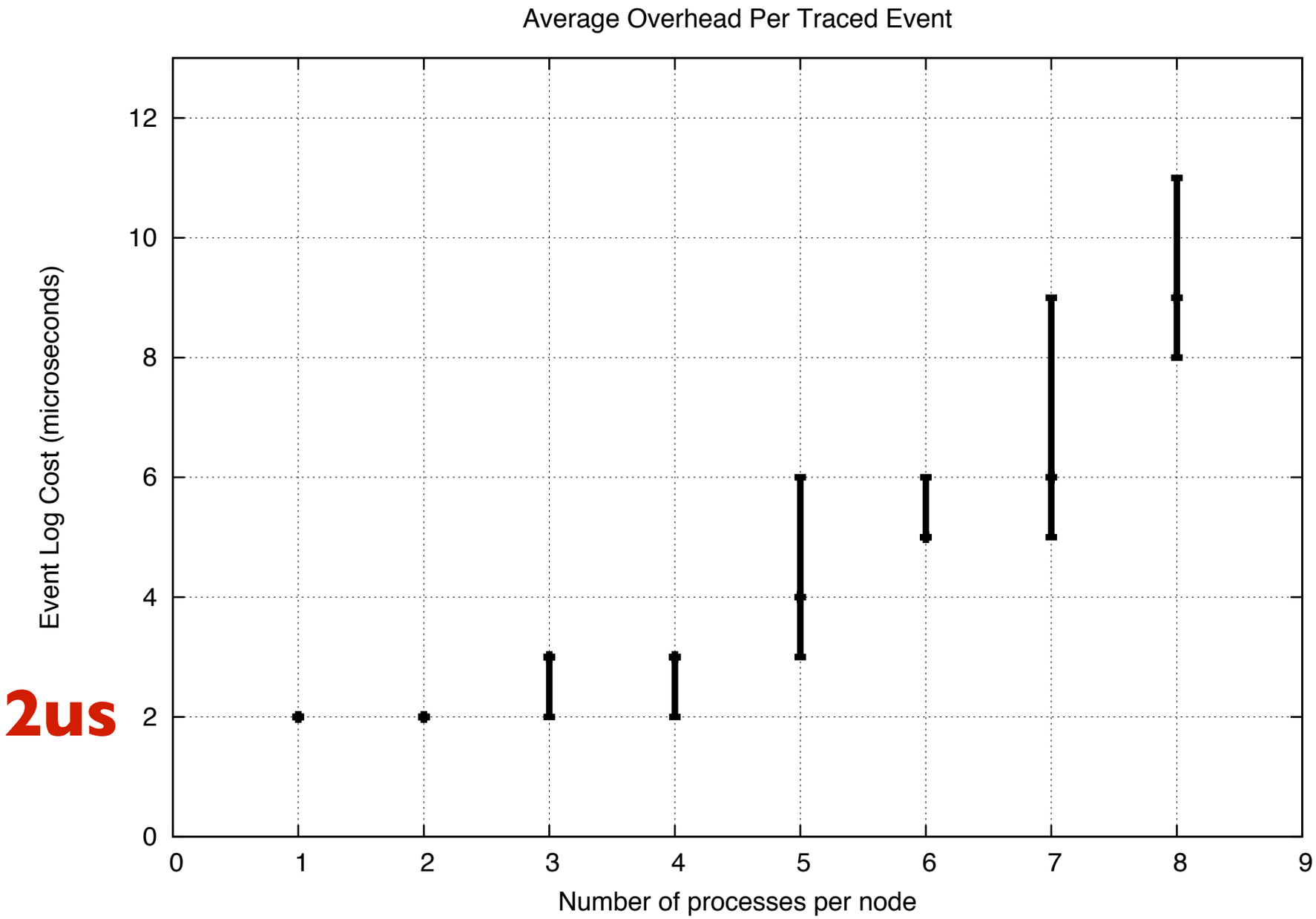
- Modular approach to interposition libraries:
  - MPI
  - MPI-IO
  - HDF
  - netCDF
  - POSIX

# Clocks are unpredictable

- Three x86 machine classes
- 5 clock sources
- Wildly different results
- TSC not shown
- Need pluggable clock architecture



# Logging cost per core

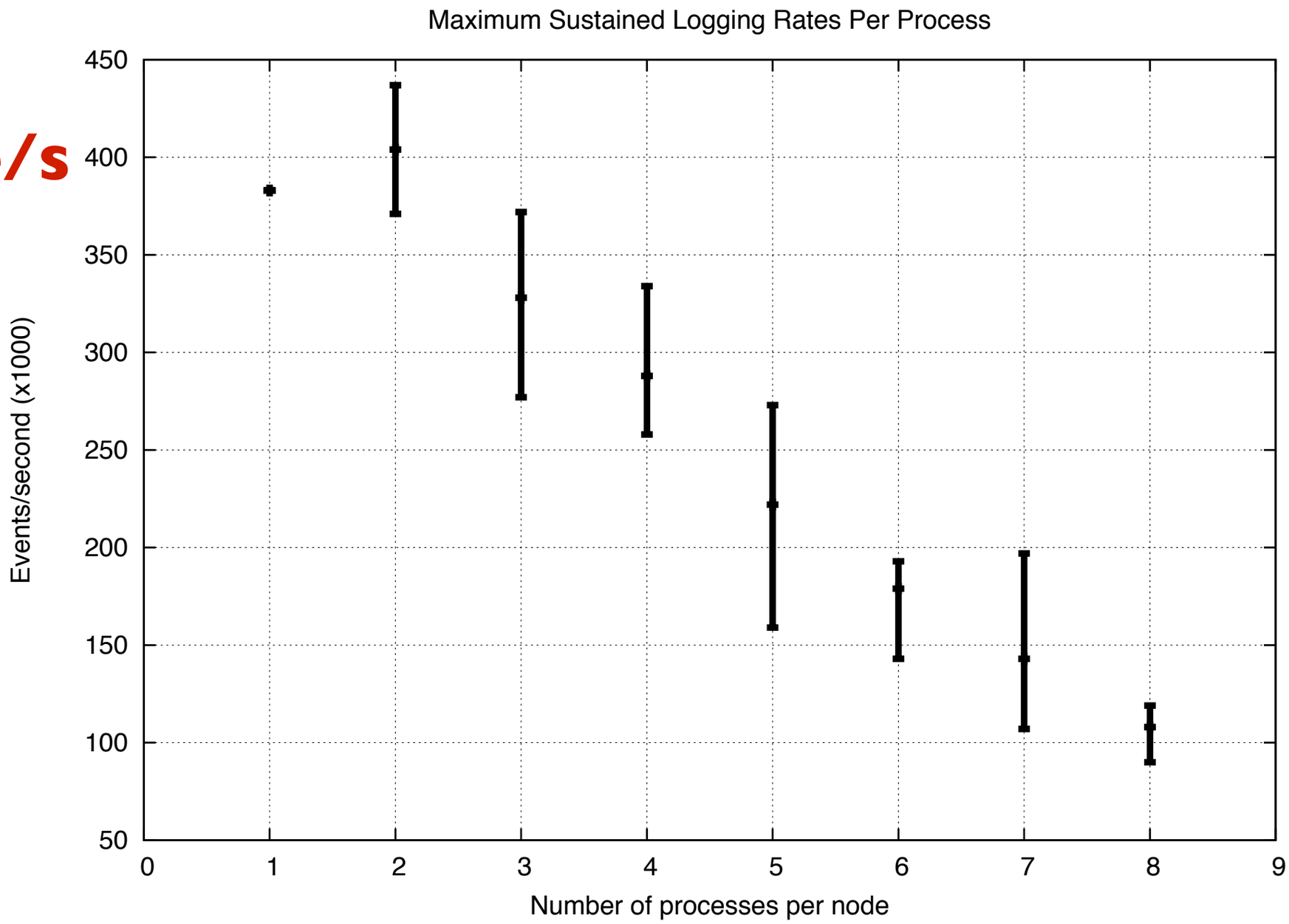



**1 1us**

**2us**

# Logging throughput per core

**400K e/s**




**100K e/s x 8 cores**

# Tracing status

- Integrated into real-world HPC applications
- Tracing infrastructure adapted to non-file system, parallel applications (VTK data models)
- Increasing usability and documentation

# Trace replay goals

- Standardized, portable trace format
  - XDR*, Google Buffers, Thrift
- Execution modes
  - Distributed POSIX traces v.s. MPI traces
- Fidelity assessment
  - E.g. total runtime, per-node measurements

# Trace replay

- Work in progress
- More research potential
- Inter-node synchronization important for fidelity[2]

**2. “Trace: Parallel Trace Replay With Approximate Causal Events”, Mesnier, Michael P., et. al., FAST 2007**


Questions?
jayhawk@cs.ucsc.edu