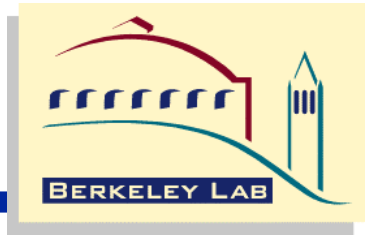
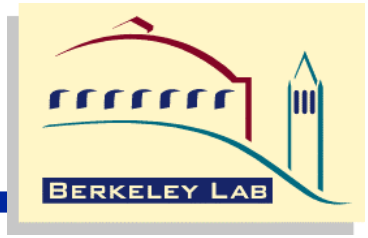


Using NetLogger for Distributed Systems Performance Analysis of the BaBar Data Analysis System

Brian L. Tierney
Dan Gunter

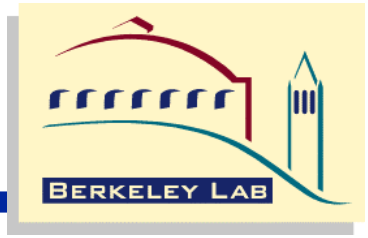
Data Intensive Distributed Computing Group
Lawrence Berkeley National Laboratory

Outline



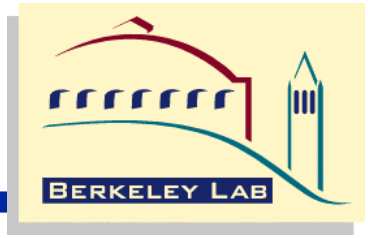
- NetLogger Overview
- NetLogger Components
- Results from BaBar analysis

Overview



- **The Problem**
  - **When building distributed systems, we often observe unexpectedly low performance**
    - **the reasons for which are usually not obvious**
  - **The bottlenecks can be in any of the following components:**
    - the applications
    - the operating systems
    - the disks or network adapters on either the sending or receiving host
    - the network switches and routers, and so on
- **The Solution:**
    - **Highly instrumented systems with precision timing information and analysis tools**

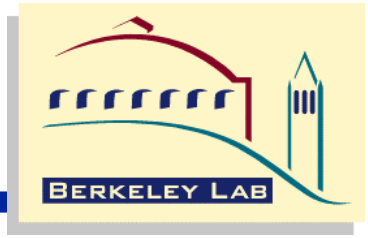
Bottleneck Analysis



- Distributed system users and developers often assume the problem is network congestion
  - This is often not true

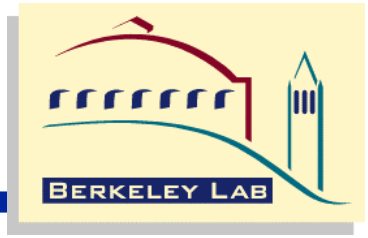
- In our experience tuning distributed applications, performance problems are due to:
  - network problems: 40%
  - host problems: 20%
  - application design problems/bugs: 40%
    - 50% client , 50% server
- Therefore it is equally important to instrument the applications

NetLogger Toolkit



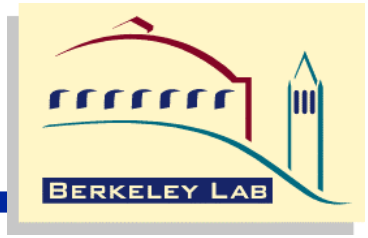
- We have developed the NetLogger Toolkit, which includes:
  - tools to make it easy for distributed applications to log interesting events at every critical point
  - tools for host and network monitoring
- The approach is novel in that it combines network, host, and application-level monitoring to provide a complete view of the entire system.
- This has proven invaluable for:
  - isolating and correcting performance bottlenecks
  - debugging distributed applications

Why “NetLogger”?



- The name “NetLogger” is somewhat misleading
  - Should really be called: “Distributed Application, Host, and Network Logger”
- “NetLogger” was a catchy name that stuck

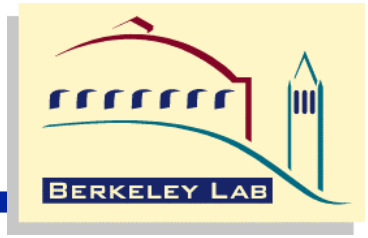
NetLogger Components



- **NetLogger Toolkit contains the following components:**
  - **NetLogger message format**
  - **NetLogger client library**
  - **NetLogger visualization tools**
  - **NetLogger host/network monitoring tools**

- **Additional critical component for distributed applications:**
  - **NTP (Network Time Protocol) or GPS host clock is required to synchronize the clocks of all systems**

NetLogger Message Format

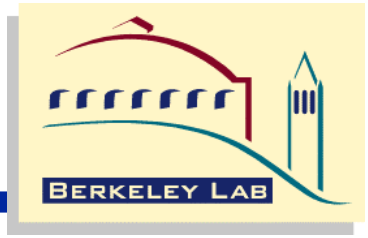


- We are using the IETF draft standard Universal Logger Message (ULM) format:
  - a list of “field=value” pairs
  - required fields: DATE, HOST, PROG; followed by optional user defined fields
  - http://www.ietf.org/internet-drafts/draft-abela-ulm-05.txt
- Sample ULM event

```
DATE=19980430133038.055784 HOST=foo.lbl.gov
  PROG=testprog LVL=Usage NL.EVNT=SEND_DATA
  SEND.SZ=49332
```

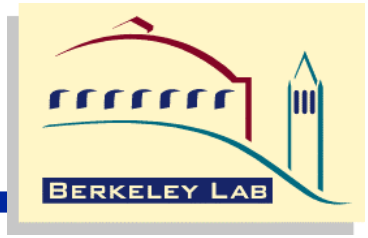
- We are currently adding XML support as well

NetLogger API



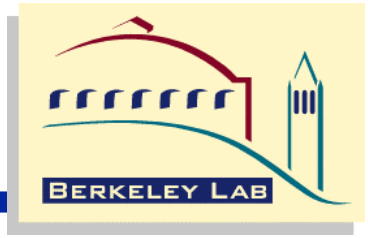
- **NetLogger Toolkit includes application libraries for generating NetLogger messages**
  - **Can send log messages to:**
    - **file**
    - **host/port (*netlogd*)**
    - **syslogd**
    - **memory, then one of the above**
- **C, C++, Java, Fortran, Perl, and Python APIs are currently supported**

NetLogger API



- **Only 6 simple calls:**
  - **NetLoggerOpen()**
    - **create NetLogger handle**
  - **NetLoggerWrite()**
    - **get timestamp, build NetLogger message, send to destination**
  - **NetLoggerGTWrite()**
    - **must pass in results of Unix gettimeofday() call**
  - **NetLoggerFlush()**
    - **flush any buffered message to destination**
  - **NetLoggerSetLevel()**
    - **set ULM severity level**
  - **NetLoggerClose()**
    - **destroy NetLogger handle**

Sample NetLogger Use



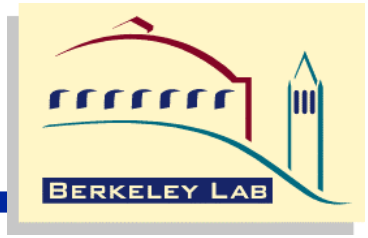

```
lp = NetLoggerOpen(method, progname, NULL,
                   hostname, NL_PORT);

while (!done)
{
        NetLoggerWrite(lp, "EVENT_START",
                           "TEST.SIZE=%d", size);

        /* perform the task to be monitored */
        done = do_something(data, size);

        NetLoggerWrite(lp, "EVENT_END");
}
NetLoggerClose(lp);
```

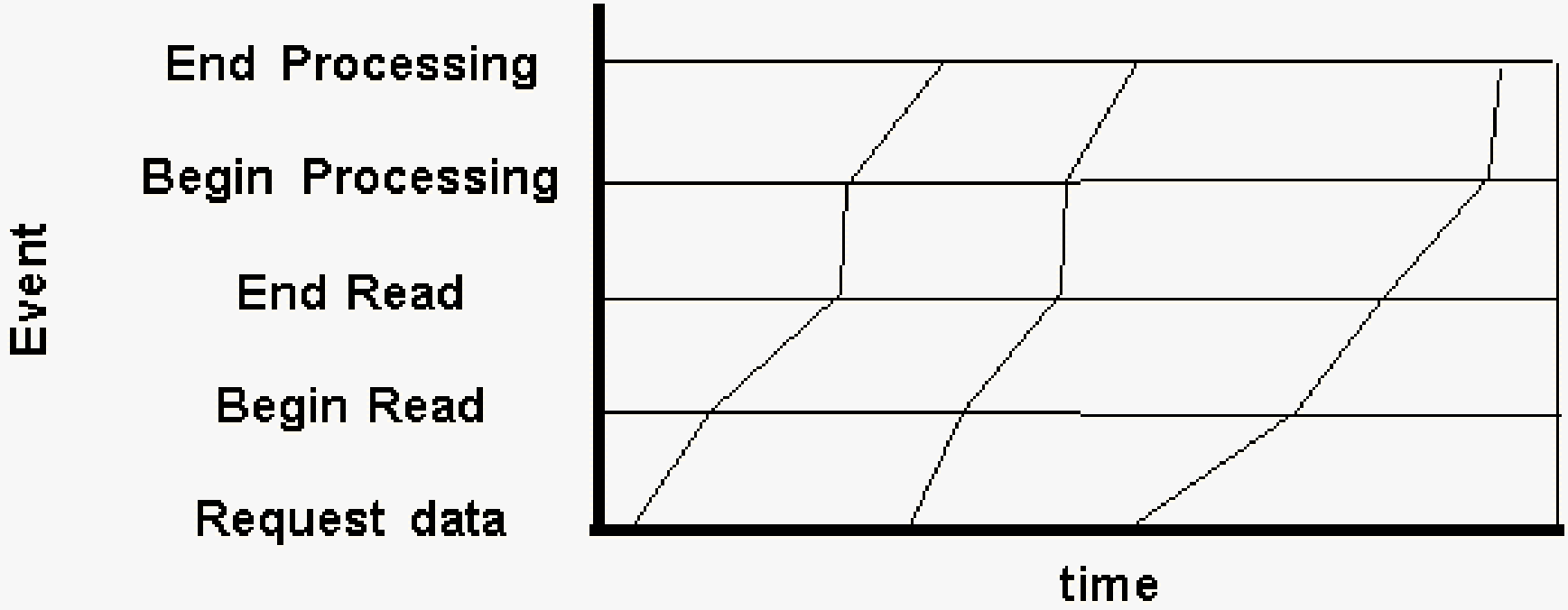
NetLogger Host/Network Tools



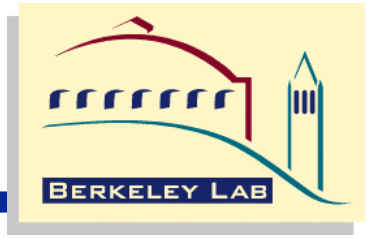
- **Wrapped UNIX network and OS monitoring tools to log "interesting" events using the same log format**
  - ***netstat* (TCP retransmissions, etc.)**
  - ***vmstat* (system load, available memory, etc.)**
  - ***iostat* (disk activity)**
  - ***ping***

- **These tools have been wrapped with Perl or Java programs which:**
  - **parse the output of the system utility**
  - **build NetLogger messages containing the results**

NetLogger Event “Life Lines”

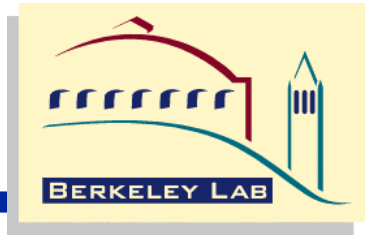


Event ID



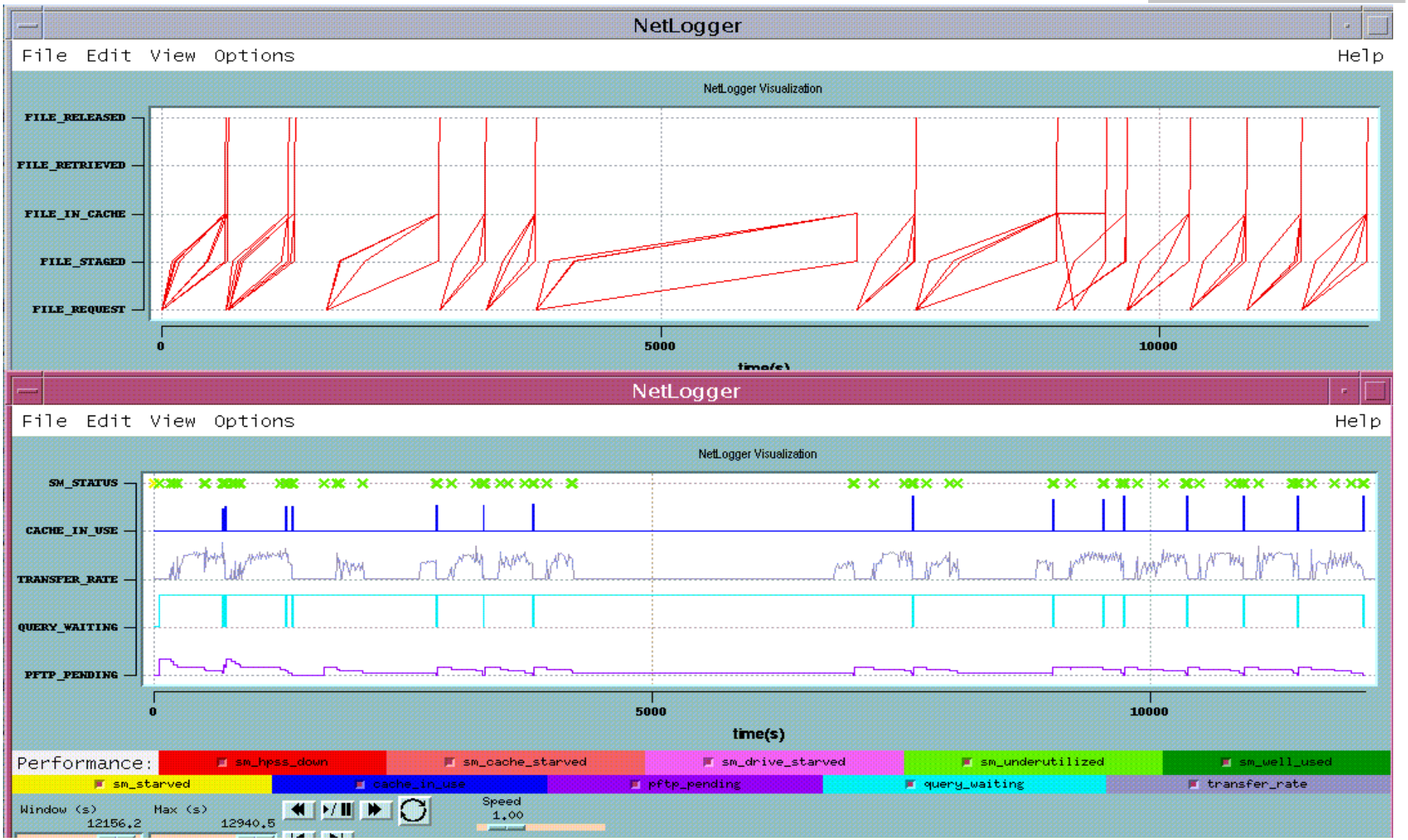
- In order to associate a group of events into a “lifeline”, you must assign an event ID to each NetLogger event
- Sample Event Ids
  - file name
  - block ID
  - frame ID
  - user name
  - host name
  - etc.

NetLogger Visualization Tools

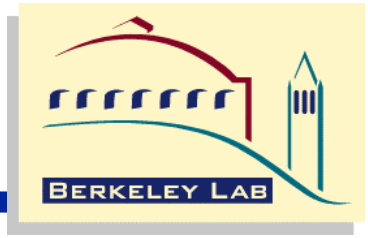


- **Exploratory, interactive analysis of the log data has proven to be the most important means of identifying problems**
  - **this is provided by *nlv* (NetLogger Visualization)**
- ***nlv* functionality:**
  - **can display several types of NetLogger events at once**
  - **user configurable: which events to plot, and the type of plot to draw (lifeline, load-line, or point)**
  - **play, pause, rewind, slow motion, zoom in/out, and so on**
  - ***nlv* can be run post-mortem or in real-time**
    - **real-time mode done by reading the output of *netlogd* as it is being written**

NLV Example

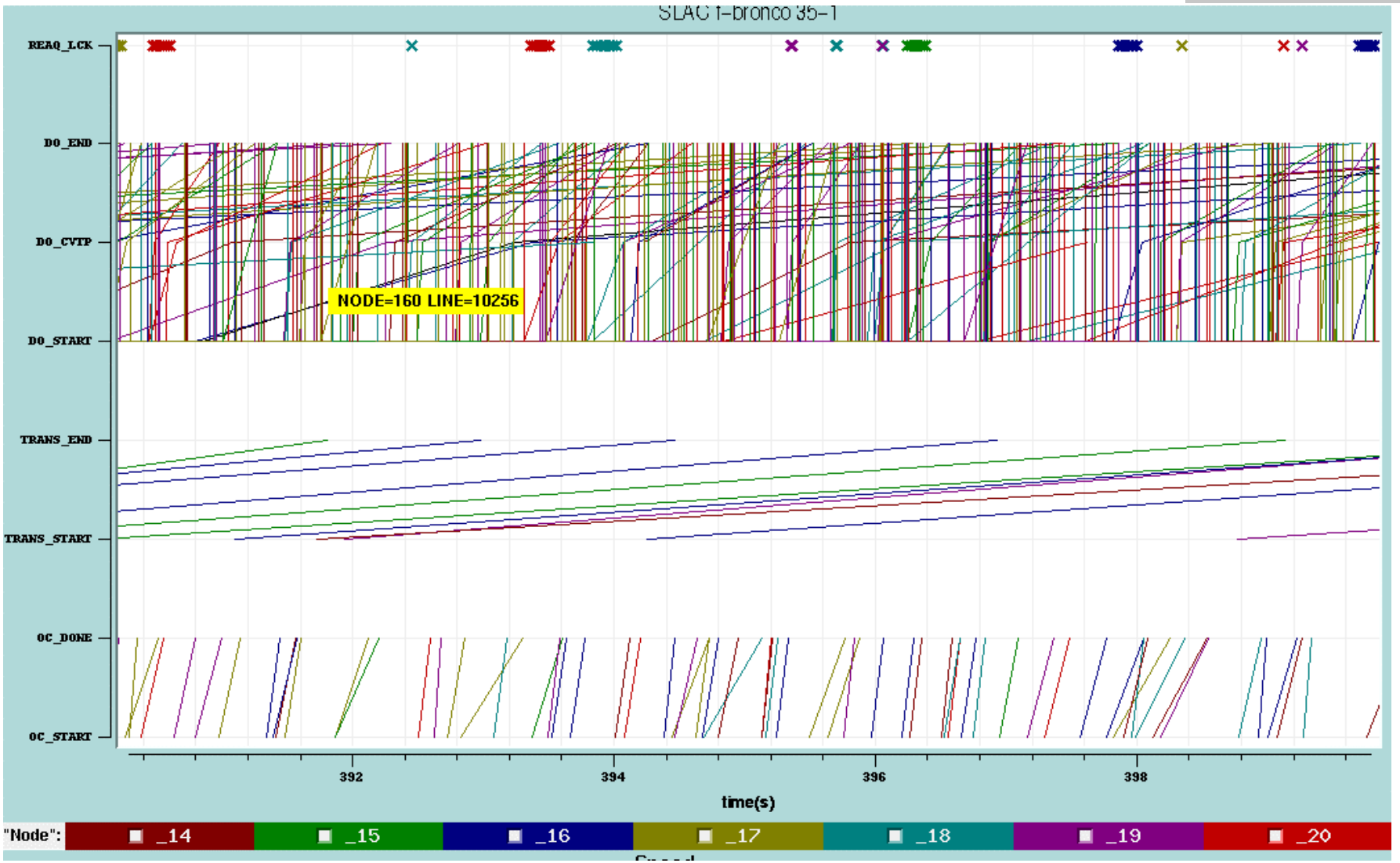

NetLogger
File Edit View Options
Help
NetLogger Visualization
FILE_RELEASED
FILE_RETRIEVED
FILE_IN_CACHE
FILE_STAGED
FILE_REQUEST
0
5000
10000
NetLogger
File Edit View Options
Help
NetLogger Visualization
SM_STATUS
CACHE_IN_USE
TRANSFER_RATE
QUERY_WAITING
PFTP_PENDING
0
5000
10000
time(s)
Performance:
sm_hpss_down
sm_cache_starved
sm_drive_starved
sm_underutilized
sm_well_used
sm_starved
cache_in_use
pftp_pending
query_waiting
transfer_rate
Window (s)
12156.2
Max (s)
12940.5
Speed
1.00

What to Instrument in Your Application

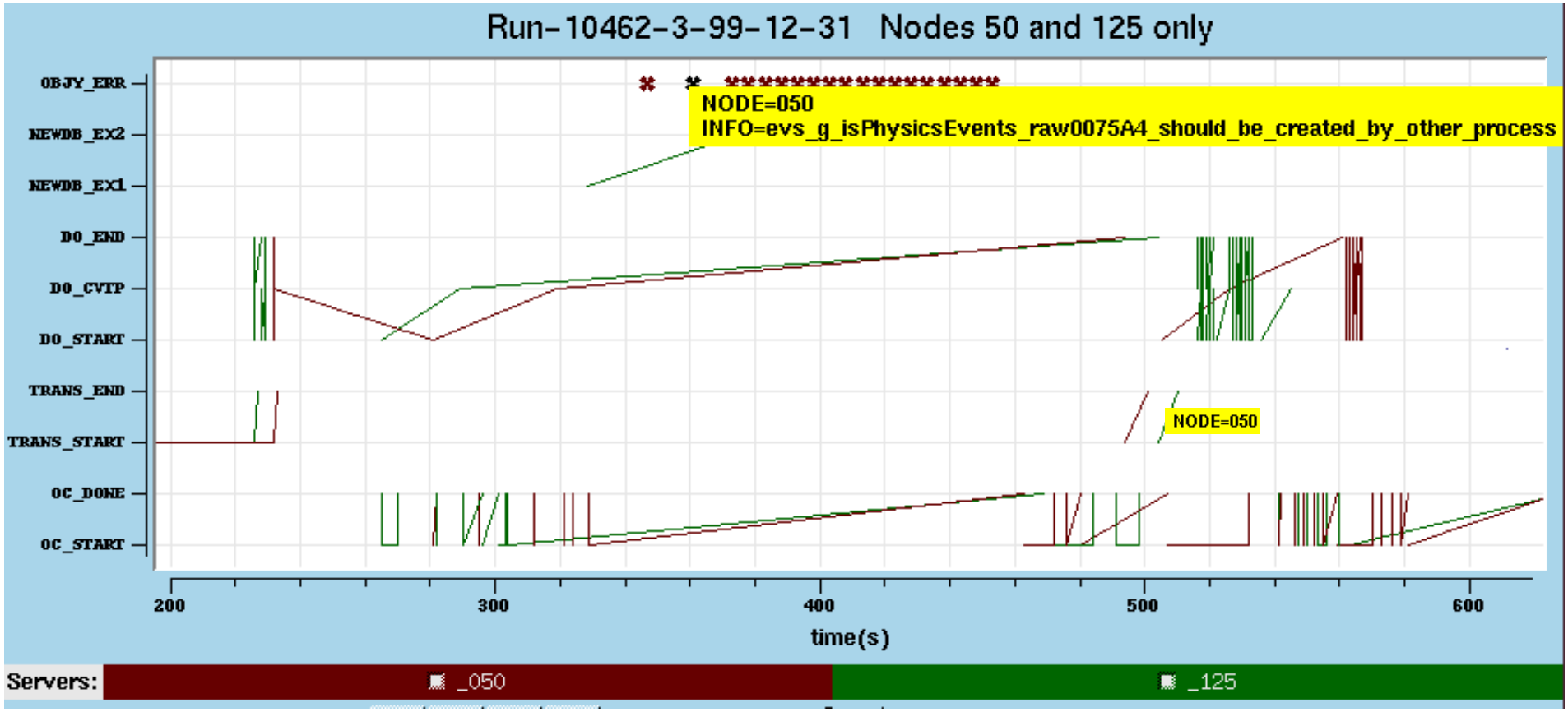


- You'll probably want to add a NetLogger event to the following places in your distributed application:
  - before and after all disk I/O
  - before and after all network I/O
  - entering and leaving each distributed component
  - before and after any significant computation
    - e.g.: an FFT operation
  - before and after any significant graphics call
    - e.g.: certain CPU intensive OpenGL calls
- This is usually an iterative process
  - add more NetLogger events as you zero in on the bottlenecks

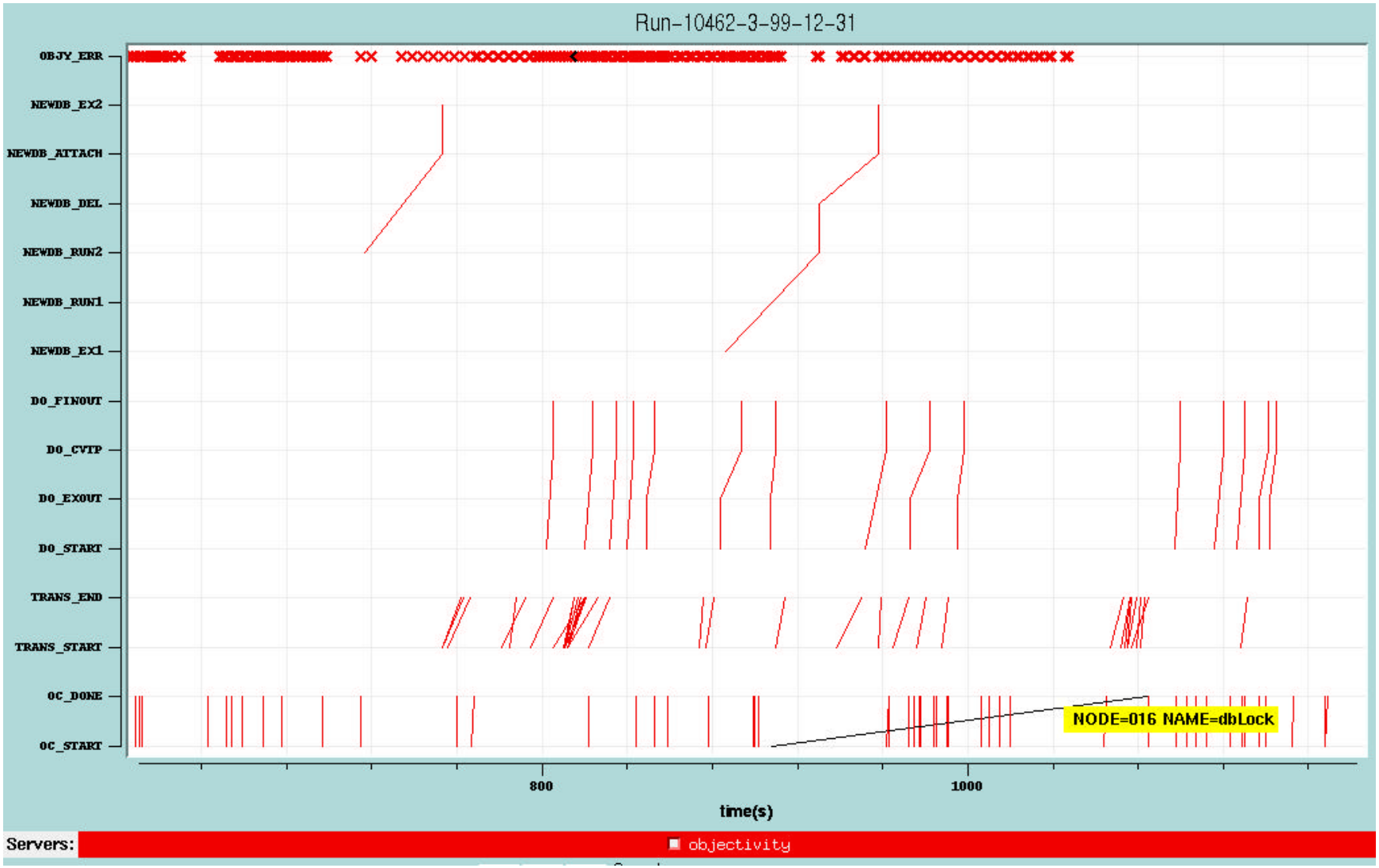
Results


SLAC 1-bronco 35-1
REAQ_LCK
DO_END
DO_CVTP
DO_START
TRANS_END
TRANS_START
OC_DONE
OC_START
NODE=160 LINE=10256
392
394
396
398
time(s)
"Node":
_14
_15
_16
_17
_18
_19
_20

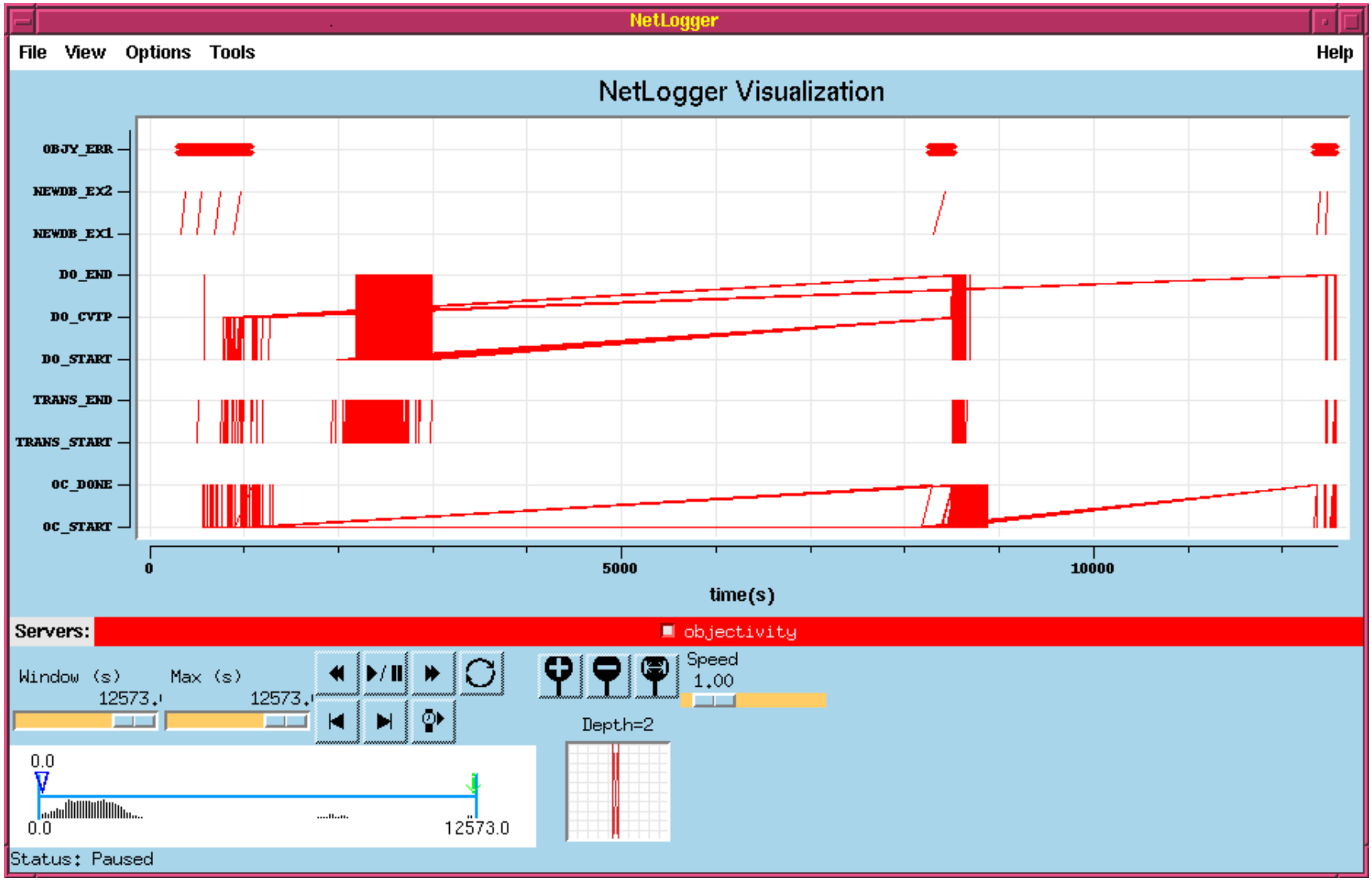
Results: 2 nodes with Objectivity Error



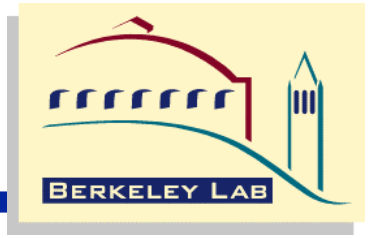
Results: dblock Example



Results: Possible Deadlock



Getting NetLogger



- **Source code and binaries are available at:**
  - **http://www-didc.lbl.gov/NetLogger**
- **Client libraries run on all Unix platforms**
- **Solaris, Linux, and Irix versions of *nlv* are currently supported**