NFS over TCP:

Excessive wakeups with BSD-based stacks

Ric Werme

Compaq Computer

Connectathon 2002

A single 64 KB NFS write

- 45 “classic” frames or
- 8 jumbo frames:

```
clnt.791 > svr.2049: P 829:9789(8960) ack 873 win 49152
clnt.791 > svr.2049: P 9789:18749(8960) ack 873 win 49152
clnt.791 > svr.2049: P 18749:27709(8960) ack 873 win 49152
clnt.791 > svr.2049: P 27709:36669(8960) ack 873 win 49152
clnt.791 > svr.2049: P 36669:45629(8960) ack 873 win 49152
clnt.791 > svr.2049: P 45629:54589(8960) ack 873 win 49152
clnt.791 > svr.2049: P 54589:63549(8960) ack 873 win 49152
clnt.791 > svr.2049: P 63549:66513(2964) ack 873 win 49152
svr.2049 > clnt.791: . ack 18749 win 40192
svr.2049 > clnt.791: . ack 36669 win 31232
svr.2049 > clnt.791: . ack 54589 win 22272
svr.2049 > clnt.791: . ack 66513 win 49152
svr.2049 > clnt.791: P 873:1037(164) ack 66513 win 49152
```

Data throttles

- TCP throttles:
  - window
  - congestion control
- Sockbuf throttle:
  - sb_hiwat/soreserve()
- Atomic sends
  - Can’t interleave RPC messages!

Data release

- TCP window updates
- TCP ACKs and sockbuf removals

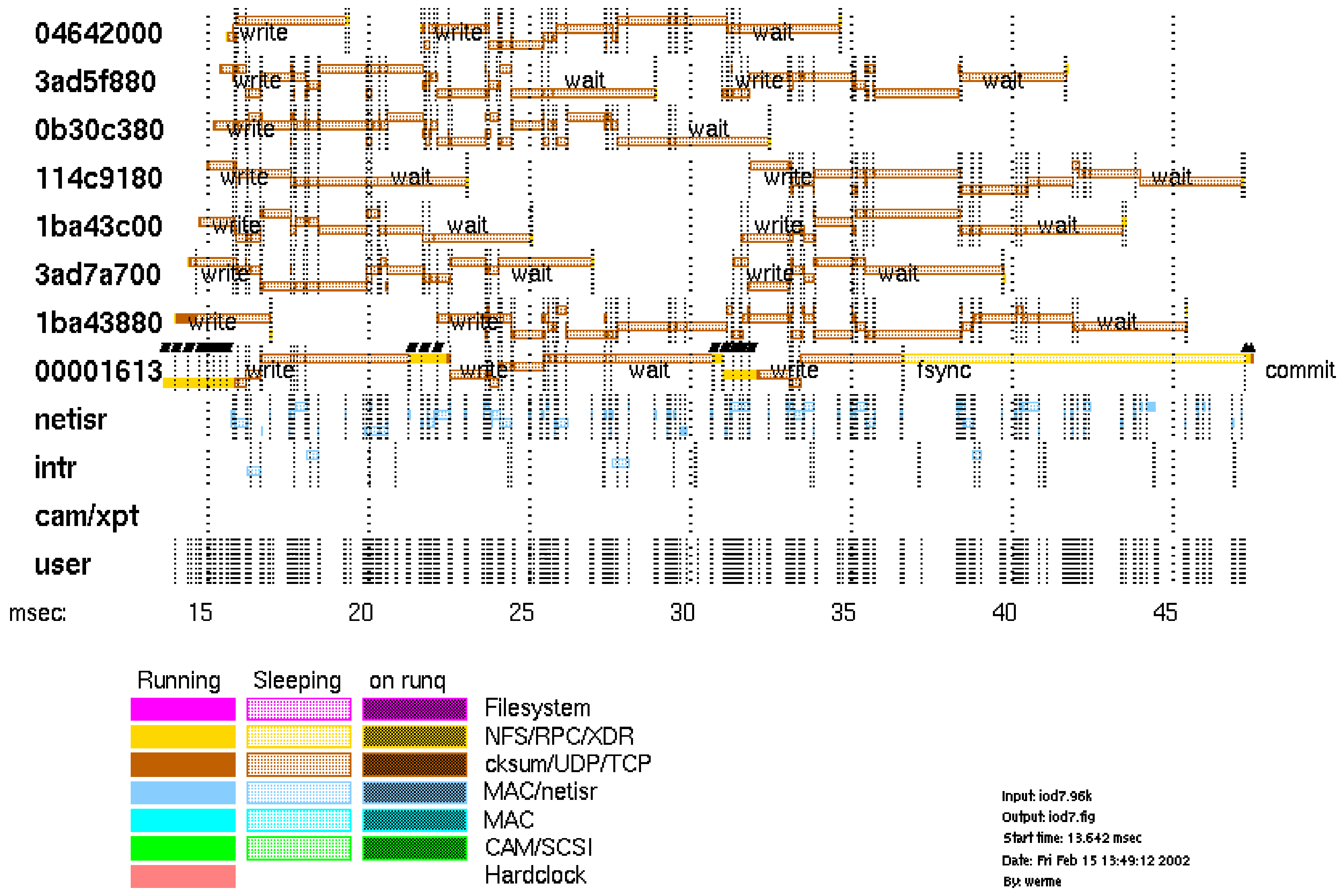
```
svr.2049 > clnt.791: . ack 54589 win 22272

svr.2049 > clnt.791: . ack 66513 win 49152
```

Multiple NFS writes

- Client: Sends several more 64 KB writes. Eventually, nfsiod threads block waiting for 64 KB in the sockbuf.
- As ACKs arrive, acknowledged data is removed and sowwakeup() called to let “the” sender buffer more data.
- All the client threads wakeup, typically all find nothing has changed and go back to sleep.
- Sometimes one thread gets lucky and buffers the next write.

16 64 KB writes, 96 KB window



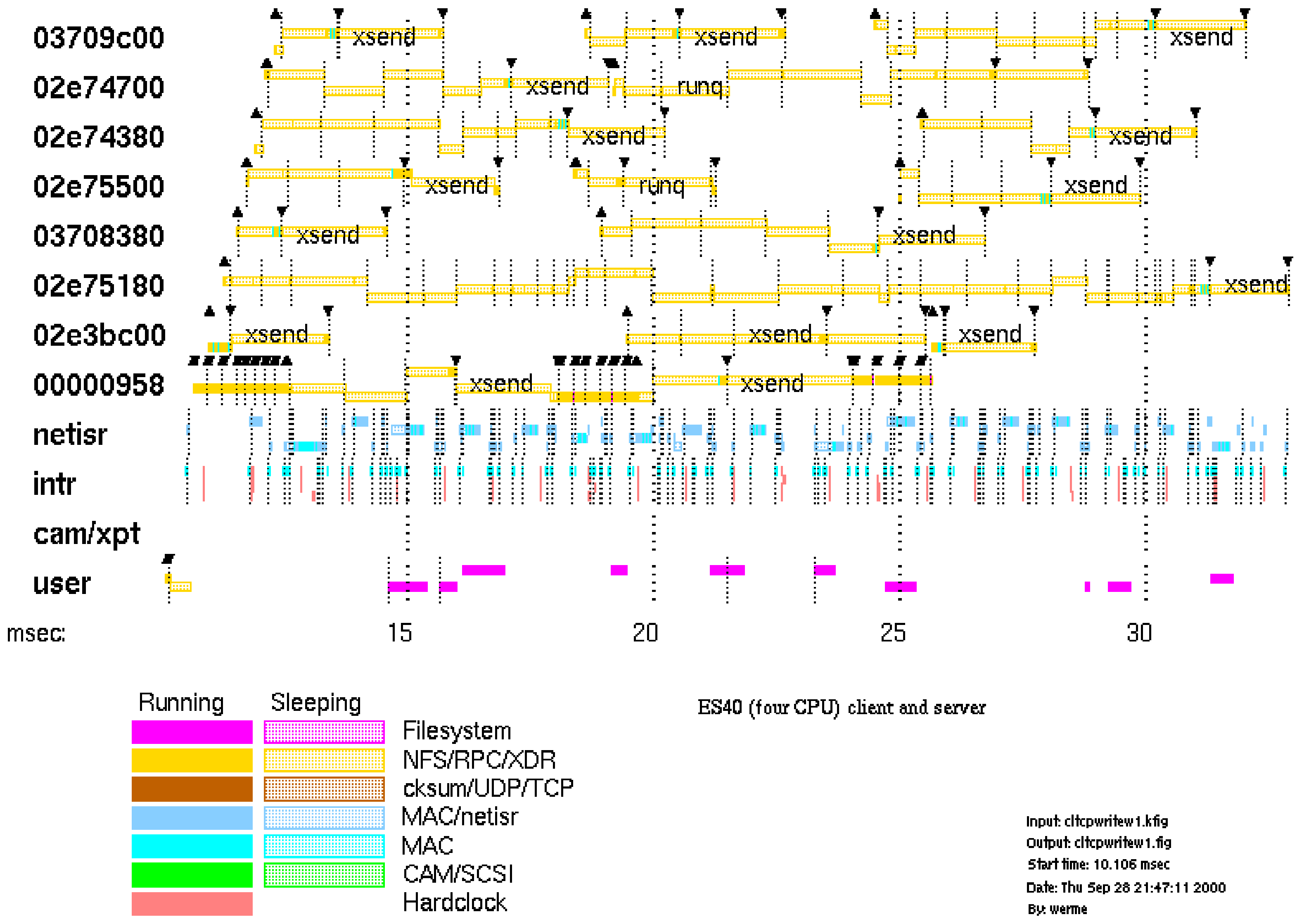
Input: iod7.96k
Output: iod7.fig
Start time: 13.642 msec
Date: Fri Feb 15 13:49:12 2002
By: werme

SB_WAKEONE?

- BSD can awaken a single thread in sowakeup(), currently not used in Tru64 Unix. (Other mechanisms are used in NFS and elsewhere.
- Does not solve the reordering problem as requests remain “SIRO” – sequential in, random out.

64 KB TCP client writes to 1152 KB file

98304 byte TCP window, 7 nfsiod threads, SB_WAKEONE



Other solutions

- Queue of threads waiting for sockbuf

  Extra locking, code, overhead.

  Could be done mostly in RPC code.

  Could use sb_wakeup() callback.

- Bigger sockbuf

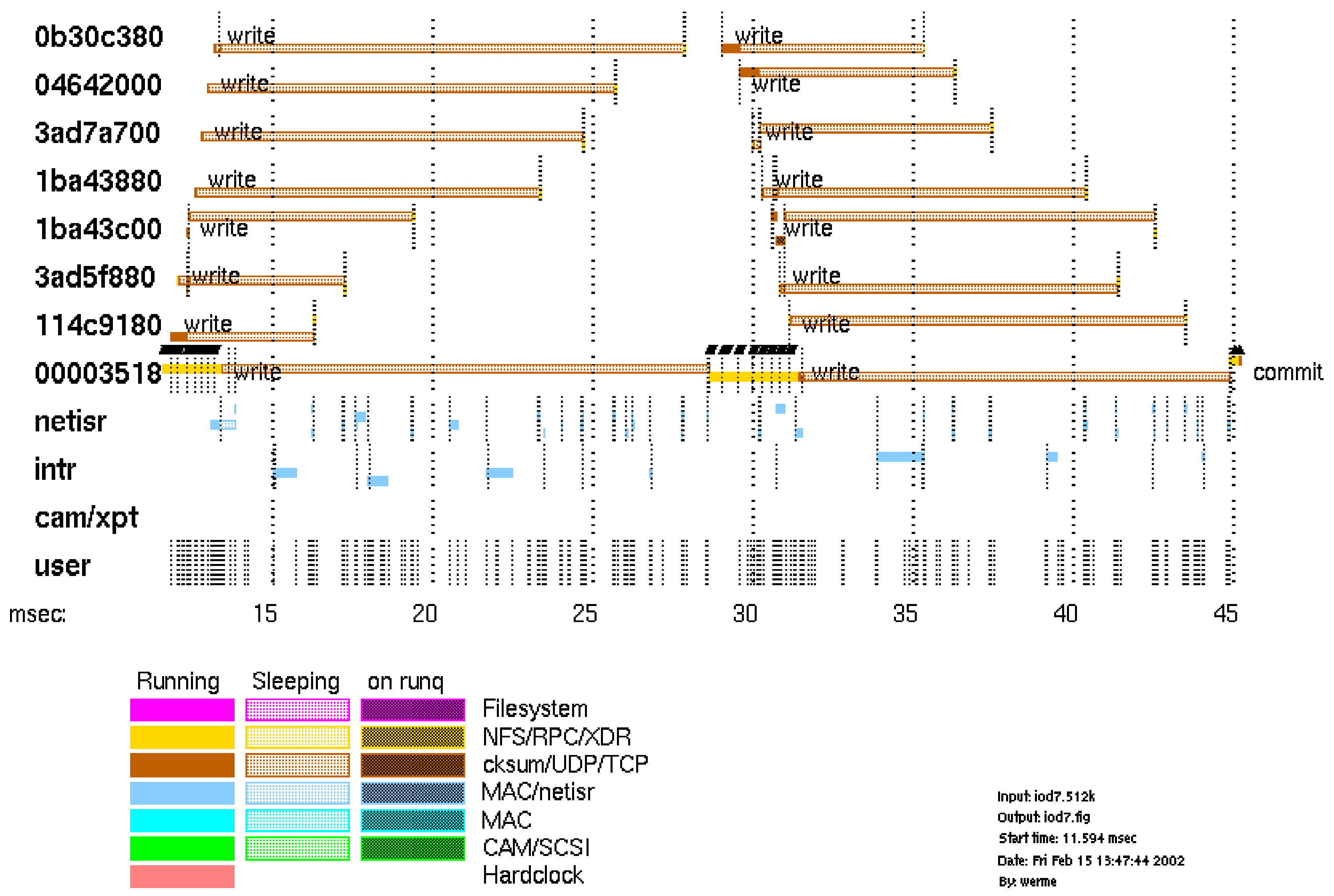
  Trivial change, at least for a first pass.

  Memory is cheap. Really cheap!

  Bigger TCP window on receive side,

  permits bigger bandwidth-delay product.

16 64 KB writes, 512 KB window



Input: iod7.512k
Output: iod7.fig
Start time: 11.594 msec
Date: Fri Feb 15 13:47:44 2002
By: werne

NFS reads

- Readahead done differently than write behind

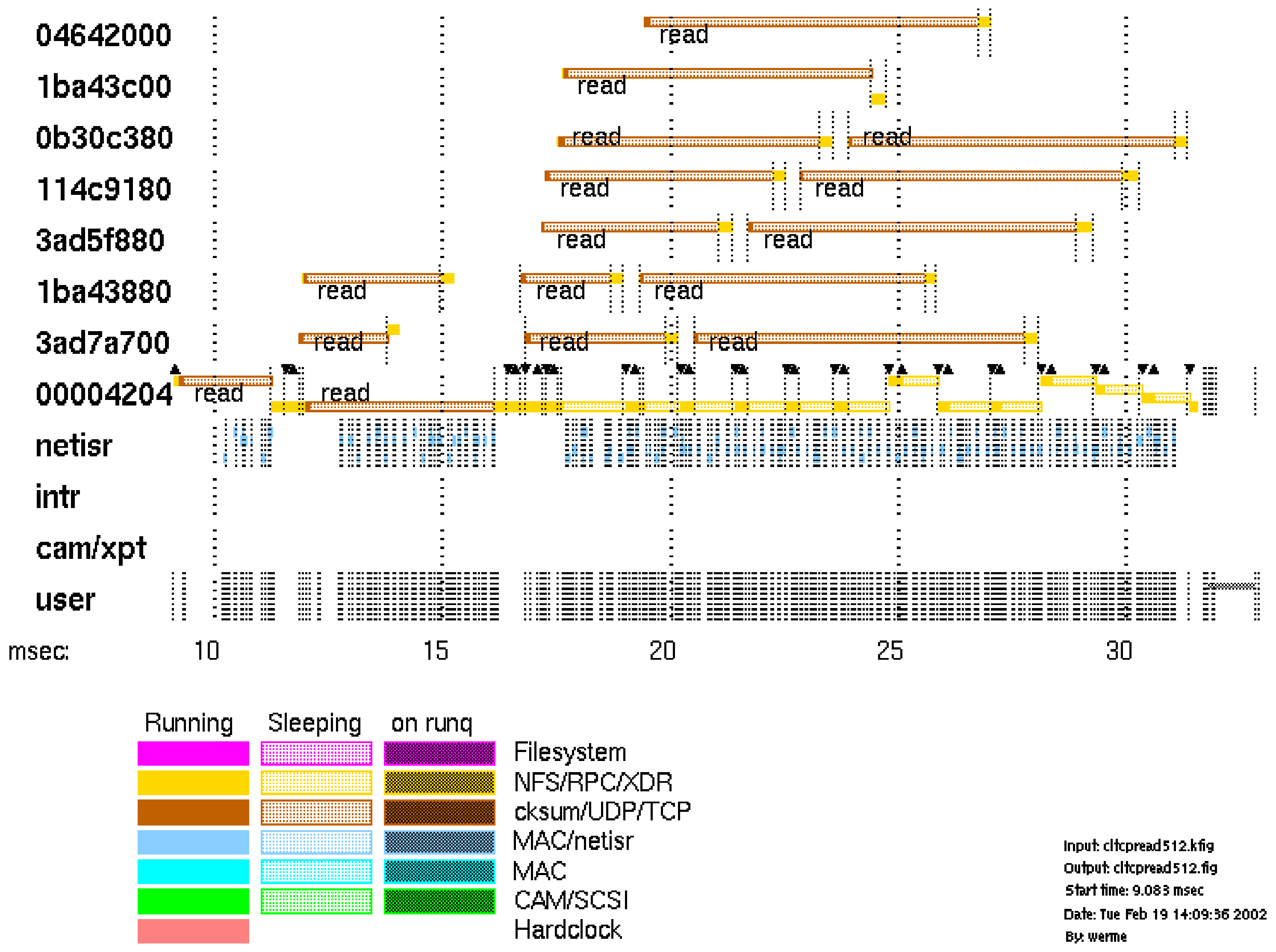
  Block 0: no read ahead

  Block 1: issue read aheads for 2 & 3, read 1

  Block 2: issue read aheads for 4 & 5, wait for 2

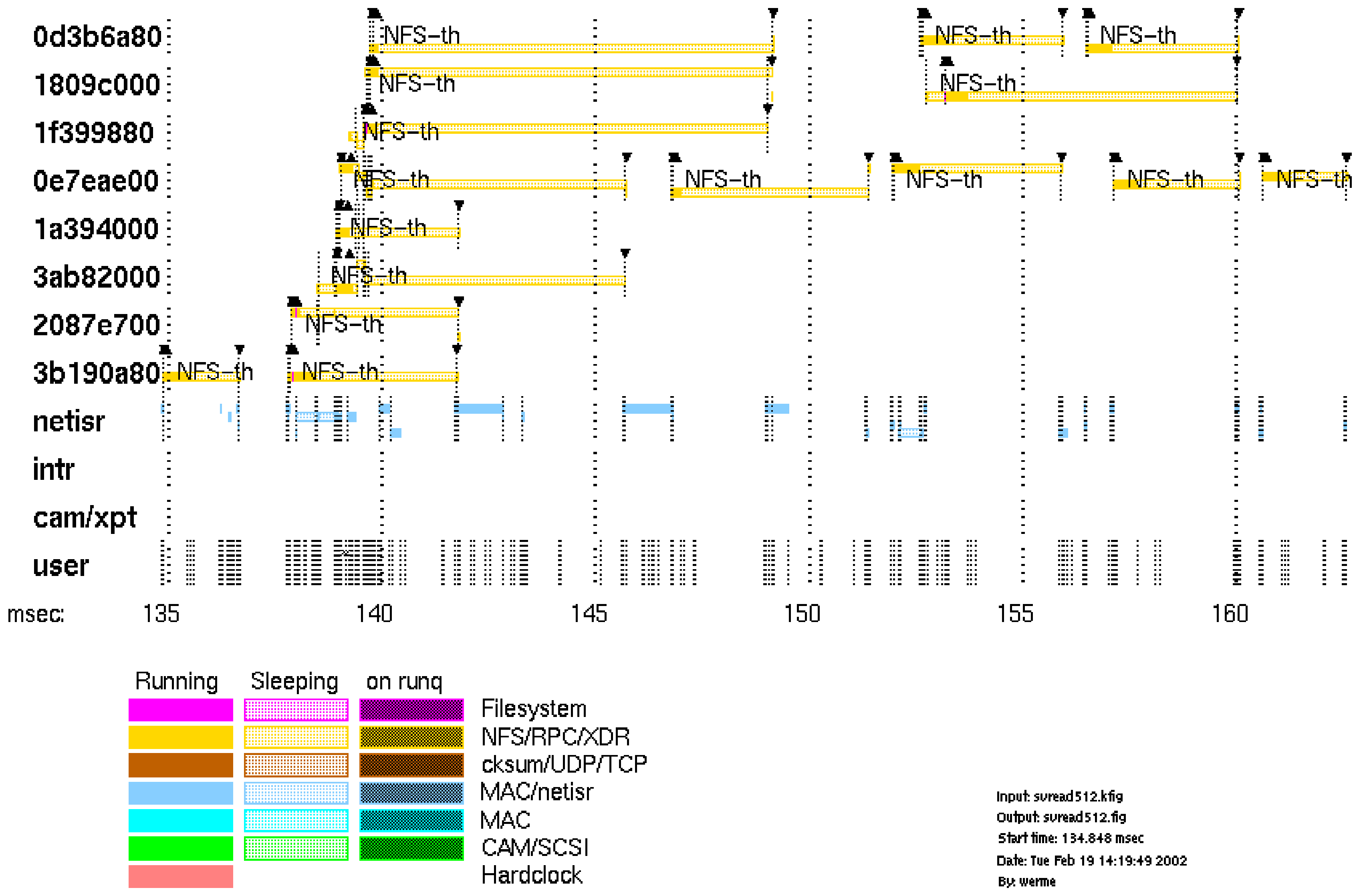
  Block n: issue read ahead for n+8, wait for n
- First, client trace (shows reads better)
- Second, server trace
- No big surprises

16 64 KB reads, 512 KB window, client view



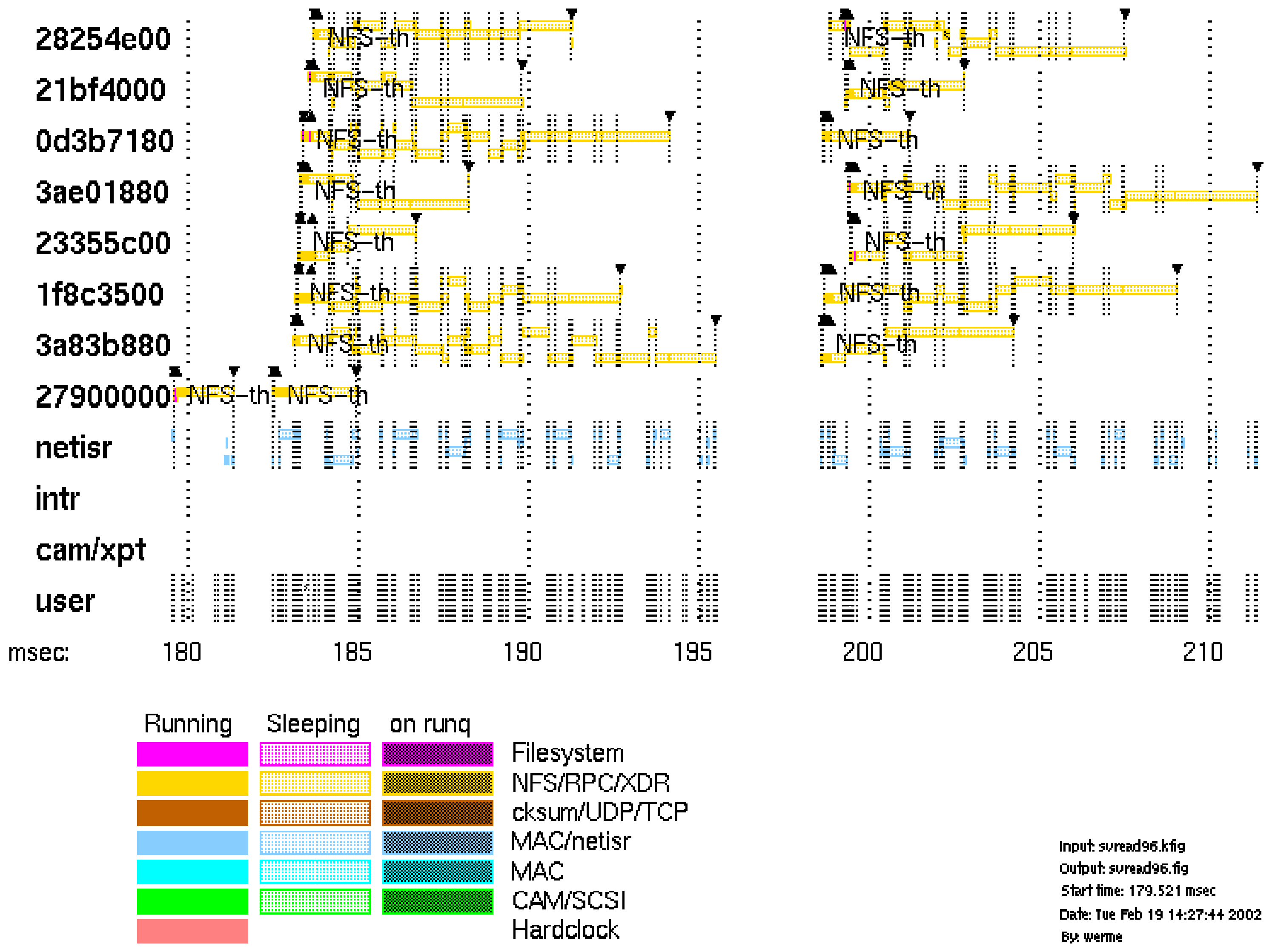
Input: cltcpread512.kfig
Output: cltcpread512.fig
Start time: 9.083 msec
Date: Tue Feb 19 14:09:36 2002
By: werme

16 64 KB reads, 512 KB window, server view



Input: suread512.kfig
Output: suread512.fig
Start time: 134.848 msec
Date: Tue Feb 19 14:19:49 2002
By: werme

16 64 KB reads, 96 KB window


28254e00
21bf4000
0d3b7180
3ae01880
23355c00
1f8c3500
3a83b880
netisr
intr
cam/xpt
user
NFS−th
msec:
Running
Sleeping
on runq
Filesystem
NFS/RPC/XDR
cksum/UDP/TCP
MAC/netisr
MAC
CAM/SCSI
Hardclock
Input: suread96.kfig
Output: suread96.fig
Start time: 179.521 msec
Date: Tue Feb 19 14:27:44 2002
By: werme