Case Study

Pat Kilfoyle
Hewlett Packard

HP WORLD 2003
Solutions and Technology Conference & Expo

Case Study

Problem -

- **RPC program 10.20-11.11 migration problem**
  - A multiprocess RPC application with many processes preforked and listening for new connections on a common TCP listen port. The connection-accept code path appears to hang and no further connections can be accepted.

Case Study

Initial application details and other relevant data -

- **Multiprocess RPC application**
  - Parent pre-forks children to handle incoming connections
  - All processes poll and attempt to accept on a common listen FD.
  - Problem is that the connection accept path hangs and no further connections can be accepted
  - 10.20 RPC library routines were BSD sockets based, while 11.X+ are XTI based by default.
  - At the customer written code level, the application is simply calling RPC library routines.

Case Study

Tools used -

- netstat
  - netstat –sp tcp
  - netstat –an
- Sample code provided by customer to duplicate the problem.
- tusc
  - Look for syscall activity leading up to the hang among all processes trying to accept new connections.

Case Study

netstat -

- netstat –sp tcp | grep 'requests dropped'

```
18 connect requests dropped due to full queue
67 connect requests dropped due to no listener
```

- netstat –an | grep EST | more

```
tcp    0    0  16.87.50.153.49350    16.87.50.153.49380    ESTABLISHED
tcp    0    0  16.87.50.153.49380    16.87.50.153.49350    ESTABLISHED
tcp    0    0  127.0.0.1.49383       127.0.0.1.49382       ESTABLISHED
tcp    0    0  16.87.50.153.49385    16.87.50.153.1013     ESTABLISHED
```

- TCP connections enter the ESTABLISHED state regardless of whether the application has completed the ‘accept’ call as long as the listen queue is not full.

Case Study

tusc -

- 11.X RPC application uses XTI library routines which are streams getmsg/putmsg syscall operations directly on /dev/tcp device vs. BSD sockets interface on 10.20
- Parent and all child processes needed to be traced.
- Sample code allowed for specifying the number of child processes from 0-20.
- Tusc trace taken of working case on 11.X (no child processes) and non-working case (multiple child processes).
- Tusc syntax:

```
tusc –flvtpE –T “” –ccc –o outputfile <pid pid pid…>
```

Case Study
-tusc output from working case

```
1054755397.720064 [4232]{12359} <0.000059> poll(0x4008ea20, 1, -1) .......... = 1
                poll[0].fd: 4
1054755397.722122 [4232]{12359} <0.000046> getmsg(4, 0x7f7f10d0, NULL, 0x7f7f1104) = 0
              ctlptr.maxlen: 1024
                ctlptr.len: 40
                ctlptr.buf: 0x40009184
  \0\0\016\0\0\010\0\0\018\0\0\0\0\0\0\0( \0\0\0\0\00203fa7f\0\001
  \0\0\0\0\0\0\0\0
                  *flagsp: 0
1054755397.722919 [4232]{12359} <0.000160> open("/dev/tcp", O_RDWR, 06050) ... = 5
1054755397.728021 [4232]{12359} <0.000052> getmsg(4, 0x7f7f11b8, NULL, 0x7f7f11c4) = 0
              ctlptr.maxlen: 1024
                ctlptr.len: 8
                ctlptr.buf: 0x40009184
  \0\0\01c\0\0\003
                  *flagsp: MSG_HIPRI
1054755397.730423 [4232]{12359} <0.000051> poll(0x4008ea20, 2, -1) .......... = 1
                poll[0].fd: 4
                poll[1].fd: 5
1054755397.731044 [4232]{12359} <0.000047> ioctl(5, I_XTI_RCV, 0x7f7f0df0) ... = 0
                  command: _IO('X', 90, 0)
1054755397.731619 [4232]{12359} <0.000089> ioctl(5, I_XTI_SND, 0x7f7f0fa8) ... = 0
                  command: _IO('X', 89, 0)
```

Case Study
-tusc output from failing MP case

```
1054755471.183712 [4237]{12385} <0.000082> poll(0x4008ea20, 1, -1) .......... = 1
                       poll[0].fd: 4
1054755471.184190 [4238]{12388} <0.000048> poll(0x4008ea20, 1, -1) .......... = 1
                       poll[0].fd: 4
1054755471.186324 [4237]{12385} <0.000050> getmsg(4, 0x7f7f10d0, NULL, 0x7f7f1104) = 0
                    ctlptr.maxlen: 1024
                       ctlptr.len: 40
                       ctlptr.buf: 0x40009184
   \0\0\016\0\0\010\0\0\018\0\0\0\0\0\0\0( \0\0\0\0\00202Y 7f\0\001
   \0\0\0\0\0\0\0\0
                          *flagsp: 0
1054755471.186736 [4238]{12388} <-0.000000> getmsg(4, 0x7f7f10d0, NULL, 0x7f7f1104) [entry]
1054755471.187350 [4237]{12385} <0.000214> open("/dev/tcp", O_RDWR, 06050) ... = 5
1054755471.191256 [4238]{12388} <0.000056> getmsg(4, 0x7f7f10d0, NULL, 0x7f7f1104) = 0      ← Race condition
                    ctlptr.maxlen: 1024
                       ctlptr.len: 8
                       ctlptr.buf: 0x40009184
   \0\0\01c\0\0\003
                          *flagsp: MSG_HIPRI
1054755471.191813 [4238]{12388} <-0.000000> poll(0x4008ea20, 1, -1) .......... [entry]
                       poll[0].fd: 4
                   poll[0].events: POLLIN|POLLPRI|POLLRDNORM|POLLRDBAND
1054755473.195888 [4237]{12385} <0.000000> getmsg(4, 0x7f7f11b8, NULL, 0x7f7f11c4) [sleeping]  ← hang
```

Case Study

Why did this work on 10.20 ?   -

- *On 10.20, the RPC library routines an application would link with were all BSD sockets based*

- *The BSD accept() syscall was an atomic operation*
    - *Kernel socket locks provided MP synchronization*

- *The 11.X RPC routines in librpcsvc are XTI based and are thread-safe, but not fork safe…per the man page.*

Case Study

Resolution -

- *A sockets based RPC library called librpcsoc is still provided with HP-UX 11.X and retains the BSD sockets based interface.*
    - *Relink the application using this library instead of librpcsvc*
- *Consider updating the application to use threads instead of multiple child processes.*

HP WORLD 2003

Solutions and Technology Conference & Expo

Interex, Encompass and HP bring you a powerful new HP World.

interex shared knowledge • shared power

encompass AN HP USER GROUP

hp invent