Case Study

Pat Kilfoyle
Hewlett Packard

HP WORLD 2003
Solutions and Technology Conference & Expo

Case Study

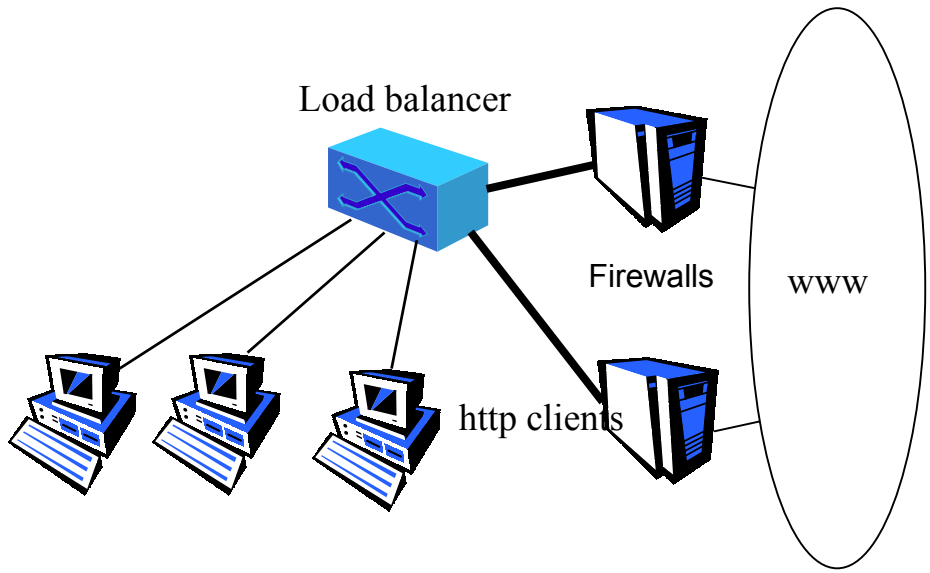
Problem -

- **Poor firewall performance – http traffic**
  - A multiprocess, multithreaded http daemon on a firewall was having slow connection handling stats according to an external network load balancer device.
  - Performance was compared with another HW vendor running the same revision of firewall product.

Case Study

Application details -

- http daemon had 10 processes with 8 kernel threads each.
- load balancer algorithm was ‘assign new connection to firewall with fewest active connections’
- New installation on 11.0



Case Study

Questions to be answered & tools to consider -

- *How do you find/ID an intermittent slow connection amongst 80 different threads spread among 10 processes?*
  - *http daemon logs that record time of transaction*
  - *A lot of network tracing and luck*
- *How do you measure/trace where a process threads spends its time?*
  - *glance process detail screens*
  - *nettl tracing at the IP layer to trace the network traffic*
  - *tusc syscall tracing …all threads traced at the same time.*
  - *kitrace syscall/kernel tool*
  - *kgmon tool to enable kernel profiling*
  - *Application logging with excruciating detail – wishful thinking.*

Case Study – tools

Tools used and the data they provided -

- http daemon logs showed which connections were delayed, but they seemed too few and infrequent to account for the overall slow performance.
  - Typically a failed DNS lookup was seen in the ***nettl*** IP layer traces.
  - The other vendors system would be subject to the same issue so this was ruled out as a root cause.
- ***tusc*** syscall trace, one tusc invocation for each process
  - Showed the thread interaction for each process
  - Searching for timestamp gaps in the syscall trace entries we were able to spot 'slow responses'.
  - recv() and ksleep() syscalls seemed to account for most of the thread delay time.
  - The tusc data showed an unexpected sequence of DNS lookups holding off other threads within the same process, calling kwakeup immediately after getting the DNS reply.
- sample threaded code was written to duplicate the DNS interaction outside of the http daemon…a simpler environment to debug.

Case Study
tusc – sample output


```
6.514689 [11972]{12562} <0.000046> socket(AF_INET, SOCK_DGRAM, 0) = 4

6.514860 [11972]{12562} <0.000033> connect(4, 0x400e0970, 16) = 0
                  sin_family: AF_INET
                    sin_port: 53
                sin_addr.s_addr: 201.155.160.51

6.515042 [11972]{12562} <0.000017> send(4, "\00201\0\001\0\0\0\0\0\0\ai p 2 ".., 25, 0) = 25

6.559376 [11972]{12562} <0.000020> select(5, 0x7f7918f0, NULL, NULL, 0x7f7918e8) = 1
                   readfds: 4
                  writefds: NULL
                  errorfds: NULL

6.559555 [11972]{12562} <0.000013> recv(4, "\0028183\001\0\0\001\0\0\ai p 2 "..,
1024, 0) = 100

6.559762 [11972]{12562} <0.000030> close(4) ..... = 0

6.563612 [11972]{12562} <0.000017> kwakeup(PTH_CONDVAR_OBJECT, 0x40001340,
WAKEUP_ONE, 0x7f790298) = 0              threads awakened: 1

6.563731 [11972]{12558} <0.000029> ksleep(PTH_CONDVAR_OBJECT, 0x40001340,
0x40001348, NULL) = 0
```

Case Study – resolution

Single threaded DNS code path found

- *The tusc output showed us an unexpected interaction among threads within the same process doing DNS queries.*
  - *It appeared to be a deliberately single threaded code path.*
  - *gdb debugger on the sample code showed us that the mutex lock was occurring in the DNS code within libnss_dns.1*
  - *Code review of the specific routines involved found old protection code in place from the days when the DNS resolver back end routines were not thread safe.*
  - *PHNE_27795 for 11.0 now contains the fix.*

Case Study

Key points - The tools and methodologies used are trying to answer the following:

- *Where is the thread/process spending it's time?*
  - *Kernel code active or sleeping?*
  - *User space active or sleeping?*
- *What is the process/thread doing?*
  - *What kernel code is it executing?*
  - *What user space code is it executing?*
- *Whatever it's doing, is it suppose to be doing it this way?*
  - *Between the application developers, the customer and HP, somebody had better know.*