dCache Network Tuning

Version 1.2
2006 Nov 02

M. Bowden, M. Crawford, W. Wu
{bowden,crawdad,wenji}@fnal.gov
Fermilab CD/CCF/Wide Area Systems

Introduction

Fermilab is a Tier-1 data center for CMS. A number of servers running "dCache" application software provide a disk caching function to reduce latency for file transfers to/from magnetic tape (figure 1).

As a test, network parameters on one of these machines were adjusted to values recommended by tuning guides [1] for long round trip and high bandwidth data links. This resulted in some instability at higher network loads. Possible causes for the instability are examined in this document.

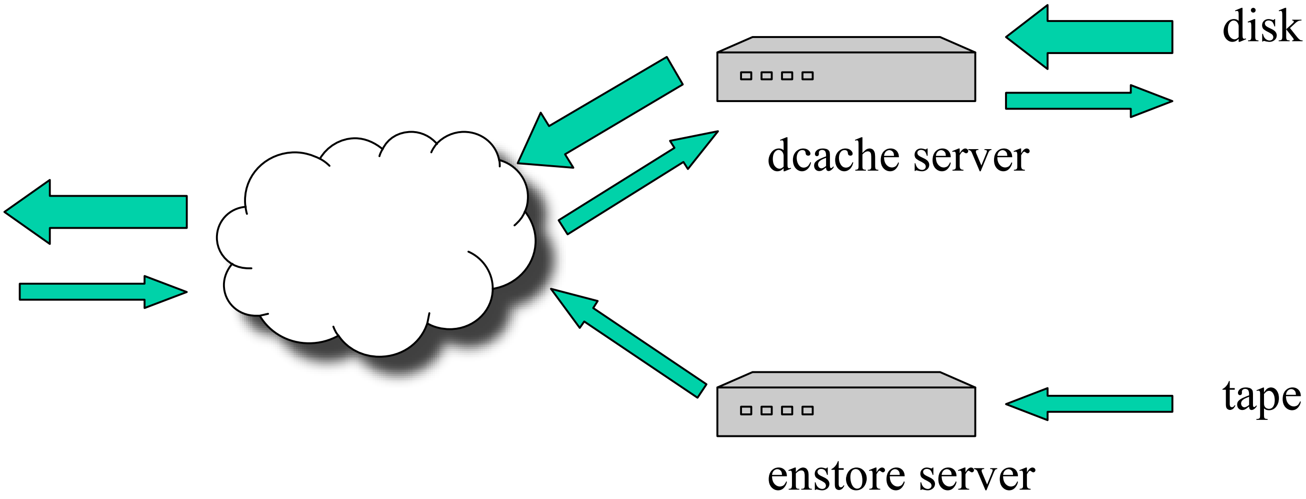

Figure 1. dCache server (read pool)

[1] http://dsd.lbl.gov/TCP-tuning/linux.html
http://www.psc.edu/networking/projects/tcptune/

System Description

The test system is a dual Xeon (3 GHz) machine with 8GBytes of memory, two 1 Gbps bonded network interface cards and a Fibre-channel disk controller. It is currently running Linux 2.6.13-2smp.

This is a 32-bit machine, which limits the available Linux "low memory" to 896MBytes (figure 2). Approximately 300MBytes of this low memory is used for the kernel and related functions (page tables, etc), leaving 600Mbytes for dynamic buffer allocation via kmalloc.

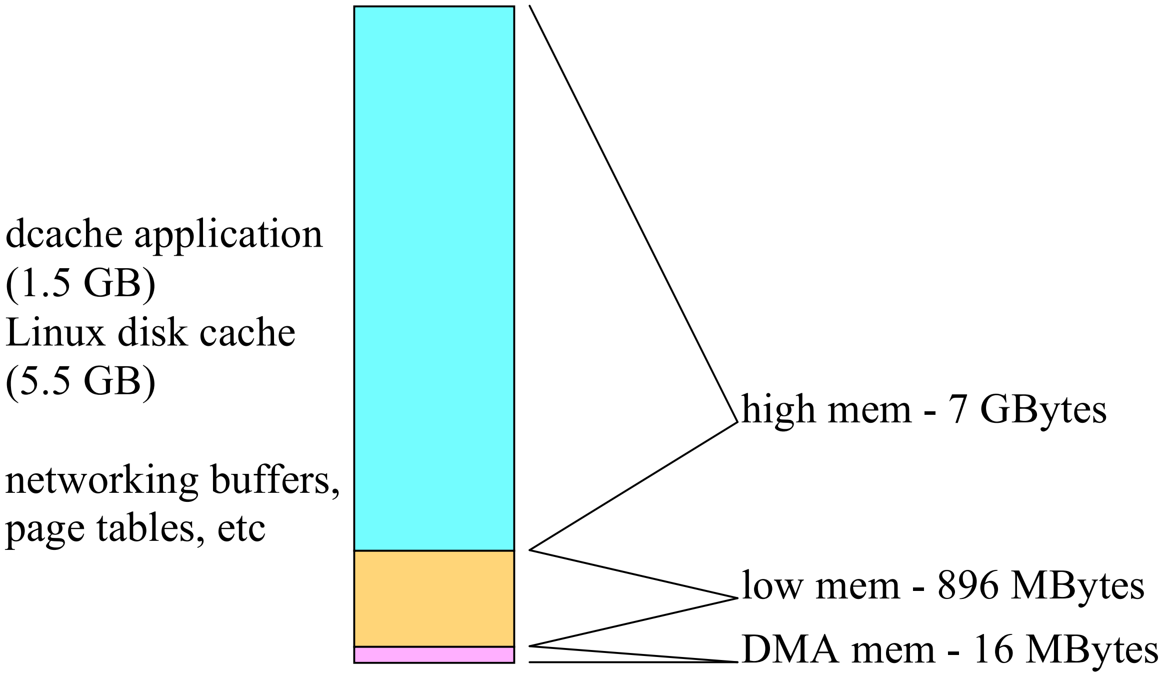

Figure 2. 32-bit memory map (8 GBytes)

Network Parameters

The default TCP buffer settings in Linux were determined at a time when network bandwidth was 10Mbps and available memory was limited. It is generally recognized that these settings are inadequate for higher bandwidth links.

The recommended settings for maximum TCP buffer size are based on the round trip time (RTT) and connection bandwidth (BW);

max_buffer_size = RTT * BW

For Fermilab-CERN traffic, the RTT is approximately 125msec. If the connection bandwidth is 1Gbps, this results in a recommended buffer size of 16MBytes. This recommendation is based on the assumption that the transfer involves only a single data stream..

Problem Description

When using GridFTP, high bandwidth transfers are typically split into 20 parallel data streams. Each stream then uses ~5% of the available bandwidth. If there are multiple GridFTP sessions using the same datapath, the bandwidth of each stream is reduced further. Applying the same RTT-BW calculation to individual GridFTP streams results in a maximum TCP buffer requirement of ~1MByte per stream.

With sufficient memory, setting the TCP buffer size to the 16MByte maximum per connection would cover all cases (single-stream and multi-stream transfers). Unfortunately, with only 600MBytes of low memory, a single GridFTP session with 20 parallel streams can theoretically exhaust all of the available kernel buffer space on a 32-bit machine. In practice, this is more likely to occur for outgoing traffic where the buffers are all filled to capacity. But under moderate load, the receive buffers will also fill quickly if the receiving application does not have enough time to offload the data. The worst case may be when there are outgoing connections using most of the kernel memory, and a few incoming connections then exhaust the remaining memory.

This situation is seen in the test system, where a maximum TCP buffer setting of 16MBytes quickly leads to page allocation failures as the network driver requests additional socket buffers;

```
Aug  4 17:42:41 kernel:  java: page allocation failure. order:0, mode:0x20
Aug  4 17:42:41 kernel:  [<c013e7ab>] __alloc_pages+0x35e/0x406
Aug  4 17:42:41 kernel:  [<c0140f89>] kmem_getpages+0x30/0x85
Aug  4 17:42:41 kernel:  [<c0141bcd>] cache_grow+0xb2/0x154
Aug  4 17:42:41 kernel:  [<c0141e4e>] cache_alloc_refill+0x1df/0x21c
Aug  4 17:42:41 kernel:  [<c014214a>] __kmalloc+0x7f/0x81
Aug  4 17:42:41 kernel:  [<c0293f23>] alloc_skb+0x35/0xc9
Aug  4 17:42:41 kernel:  [<f88f5dcd>] e1000_alloc_rx_buffers+0x64/0x38c [e1000]
Aug  4 17:42:41 kernel:  [<c0108f38>] convert_fxsr_to_user+0x122/0x180
Aug  4 17:42:41 kernel:  [<f88f55ca>] e1000_clean_rx_irq+0x210/0x508 [e1000]
Aug  4 17:42:41 kernel:  [<c02999a0>] net_rx_action+0x88/0x16e
Aug  4 17:42:41 kernel:  [<c011f589>] __do_softirq+0x69/0xd5
Aug  4 17:42:41 kernel:  [<c011f627>] do_softirq+0x32/0x34
Aug  4 17:42:41 kernel:  [<c0104c95>] do_IRQ+0x35/0x5a
Aug  4 17:42:41 kernel:  [<c01034c6>] common_interrupt+0x1a/0x20
Aug  4 17:42:41 kernel:  [<c0101ec2>] restore_sigcontext+0x3a/0x159
Aug  4 17:42:41 kernel:  [<c0102164>] sys_rt_sigreturn+0xad/0xed
Aug  4 17:42:41 kernel:  [<c0102a91>] syscall_call+0x7/0xb
```

As might be expected, these allocation errors are always seen on the receive side since the system has less control of incoming vs outgoing packets.

The TCP buffer size is a global parameter for all connections. Ideally there would be a way to set the maximum buffer size for individual connections, based on RTT and number of streams.

For the CMS-dCache application it is assumed that most of the traffic will flow form CERN to Fermilab and from Fermilab to US Tier-2 sites. This leads to a lower typical RTT for outgoing data connections.

Maximum TCP buffer sizes of 1048576 (receive) and 131072 (transmit) were chosen to reflect the differences in RTT, the "worst-case" number of concurrent connections, and the restrictions on low memory usage.

At startup the system calculates a limit for the total memory used by all TCP buffers (tcp_mem). For lowmem machines, this calculation appears to be too high by as much as a factor of 4, which could also lead to out-of-memory conditions with a large number of connections. The tcp_mem parameter is measured in pages (4kB on an IA32 machine), but one commonly sees advisory values that assume it's bytes or kB.

Other Observations

While examining the memory instability issue, the following observations were made related to Linux design philosophy as it affects high bandwidth networking applications.

Receive Priority

Linux places a lower priority on received data. The philosophy seems to be that packets can always be resent, so there is no risk. This is acceptable for LAN traffic, but dropping packets on a high RTT-bandwidth link is expensive.

The Intel e1000 driver also follows this philosophy, emphasizing transmit over receive. As a result the test machine can transmit at the full bandwidth of 2Gbps, but routinely drops received packets at rates above 200Mbps.

Linux Memory Management

Linux memory management has evolved of the last few years, becoming more and more complicated. Many of these features act against fast networking applications. For example;

1) swap

For a dedicated application like a dcache server, it could be argued that swap is unnecessary. There is no shortage of high memory where the application resides, and low memory is filled with non-swappable kernel allocated buffers. In fact, the "out-of-memory" condition in low memory is made worse by the processor spinning its wheels trying to find something to swap. The processing time might be better used by the dCache application to unload the TCP receive buffers.

2) disk caching

Linux views any unused memory as wasted, and confiscates all available high memory for disk buffer cache. Here again, it might be better to simply not use the memory than waste processing time on something that has little real value for the application. Removing half of the memory (4GBytes) in the test machine might actually improve its performance, by reducing the memory available for disk caching and reducing the size of the low memory page tables.

3) buddy zone allocation

Linux tries to concatenate small blocks of memory to form larger blocks, "just in case" an application requests a large block of consecutive memory. For networking applications, most memory requests are for single pages, and often all of the single pages have been absorbed into these higher order blocks. The memory allocator must then take a larger block and split it back into smaller blocks. This superfluous combining and splitting of memory blocks takes yet more processing time that might be better used by the networking application.

Frozen Connections

At one point during testing, we noticed a set of GridFTP connections stuck in the FIN_WAIT1 state. This condition lasted for over a week. If the TCP buffer size had been set to 16MBytes, then all of low memory could easily have been tied up in this transfer.

Recommendations

The following parameter settings are suggested for 32 bit machines;

/etc/sysctl.conf

```
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.core.rmem_max = 1048576
net.core.rmem_default = 87380
net.core.wmem_max = 131072
net.core.wmem_default = 32768
net.ipv4.tcp_rmem = 4096   87380  1048576
net.ipv4.tcp_wmem = 4096  32768  131072
net.ipv4.tcp_mem = 65536 87380 98304
vm.min_free_kbytes = 65536
vm.overcommit_memory = 2
```

/etc/rc.local

```
/sbin/ifconfig eth0 txqueuelen 1000
/sbin/ifconfig eth1 txqueuelen 1000
/sbin/ethtool -G eth0 rx 4096
/sbin/ethtool -G eth1 rx 4096
```

/etc/modprobe.conf

```
options e1000 RxDescriptors=4096,4096
```

(if using intel NIC)

64 bit machines

Sixty-four bit machines do not have the same low memory restrictions. It should be possible to increase values of tcp_mem, tcp_rmem and tcp_wmem by factors of 2-4 with no negative effects.

While 64 bit machines should be more stable in this application, first experiences are that they are not. This is currently being studied. So far, the instabilities do not appear to be related to network load. It may be simply that the development level of the 64 bit kernel lags the corresponding 32 bit kernel, or that the compilation of the dcache application under the 64 bit kernel is not as stable. A few occurrences of dcache server hangups with high cpu utilization and very high context switching rates have been noted.

Conclusions

1) With a 32-bit machine, either the TCP buffer size or the number of connections must be limited. A 64-bit machine (with 64-bit Linux) should relieve most of the memory problems.

2) Linux has become progressively less network-friendly. There is some evidence that network performance was better in Linux 2.4 and has become worse with successive releases of 2.6. Some of this may be due to increased network security and some may be due to increased complexity in the memory management.

3) Linux places higher priority on transmit vs receive network operations...exactly the opposite of what would be preferred in high RTT-BW applications.

Anecdotal Comments

...from various Google sources

.

1) “Bad things happen when you're out of lowmem.”

Linux documentation

2) “Oh, the answer is very simple: it's not going to happen. EVER. You need more than 32 bits of address space to handle that kind of memory. This is not something I'm going to discuss further...This is not negotiable.”

Linus Torvalds - 1999 (on running 32 bit Linux with more than 2GB of memory)

3) “...found through his studies that kernel 2.4...outperformed kernel 2.6 in every test and under every configuration in terms of [network] throughput and thus concluded that kernel 2.6 still has room for improvement.”

http://os.newsforge.com/article.pl?sid=05/07/22/1054216&tid=2&tid=18

4) “FreeBSD 4.9: Drops no packets at 900K pps
Linux 2.4.24: Starts dropping packets at 350K pps
Linux 2.6.12: Starts dropping packets at 100K pps”

http://www.gatago.com/linux/kernel/14693564.html

5) “... if the RX path gets very busy packets will end up being dropped ... by virtue of DMA rings being filled up ... and that is an acceptable compromise.”

http://www.spinics.net/lists/netdev/msg10601.html

6) “The issue is that RX is absolute, as you cannot "decide" to delay or selectively drop since you don't know what’s coming. Better to have some latency than dropped packets. But if you don't dedicate to RX, then you have an unknown amount of cpu resources doing "other stuff". The capacity issues always first manifest themselves as RX overruns, and they always happen a lot sooner on MP machines than UP machines. The LINUX camp made the mistake of not making RX important enough, and now their 2.6 kernels drop packets all over the ranch. But audio is nice and smooth...”

http://leaf.dragonflybsd.org/mailarchive/users/2005-12/msg00007.html

7) “...there's some latency involved... if the CPU is stuck in an interrupt handler refilling a huge network RX ring, then waking kswapd won't do anything and you will run out of memory.”

http://oss.sgi.com/archives/xfs/2004-12/msg00008.html

8) “By default, Linux follows an optimistic memory allocation strategy. This means that when malloc() returns non-NULL there is no guarantee that the memory really is available. This is a really bad bug.”

http://www.novell.com/coolsolutions/qna/11511.html