Apache Scaling

Scaling Apache 2.x beyond 20,000 concurrent downloads

`ftp.heanet.ie` is HEAnet's National Mirror Server for Ireland. Currently mirroring over 50,000 projects, it is a popular source of content on the internet. It serves mostly static content via HTTP, FTP and RSYNC, all available via IPv4 and IPv6. It regularly sustains over 20,000 concurrent connections on a single Apache instance and has served as many as 27,000 with about 3.5 Terabytes of content per day. The front-end system is a Dell 2650, with 2 2.4 Ghz Xeon processors, 12Gb of memory and the usual 2 system disks and 15k RPM SCSI disks, running Debian GNU/Linux and Apache 2.x.

Considerable system and application tuning enabled this system to achieve these performance rates. Apachebench was used for web server benchmarking, bonnie++ and iozone for file system benchmarking and an in-house script to measure buffering, virtual memory management and scheduling. The steps taken are highlighted below:

Apache

MPM Tuning

Apache 2.x has a choice of multi-processing modules. For this system, the prefork MPM was chosen, tuned to have 10 spare servers, the number of spare servers calculated such that there are enough child processes available to handle new requests when the rate of new connections exceeds the rate at which Apache can manage to create new processes.

Module Compilation

Apache modules can be compiled in directly to one binary, or as dynamically-loaded share dobjects which are then loaded by a smaller binary. For our load, a small performance gain (measurable as about 0.2%) was found by compiling the modules in directly.

htaccess

As the highperformance.conf sample provided with Apache suggests, turning off the use of .htaccess files, if appropriate can give significant performance improvements.

sendfile

Sendfile is a system call that enables programs to hand off the job of sending files out of network sockets to the kernel, improving performance and efficiency. It is enabled by default at compile-time if Apache detects that the system supports the call. However, the Linux implementation of sendfile corrupted IPv6 sessions so this was not implementable on ftp.heanet.ie for policy reasons.

Mmap

Mmap (memory map) support allows Apache to treat a file as if it were a contiguous region of memory, greatly speeding up the I/O by dispensing with unnecessary read operations. This allowed serving of files roughly 3 times quicker.

mod_cache

mod_disk_cache is an experimental feature in Apache 2.x that caches files in a defined area as they are being served for the first time. Repeated requests are served from this cache, avoiding the slower file systems. mod The default was further tuned to increase the CacheDirLevel4 to 5 to faciliate more files in the cache.

Configure Options

The following configure options were used:

```
CFLAGS="-D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE"; export CFLAGS
"./configure" \
"--with-mpm=prefork" \
"--prefix=/usr/local/web" \
"--enable-cgid" \
"--enable-rewrite" \
"--enable-expires" \
"--enable-cache" \
"--enable-disk-cache" \
"--without-sendfile"
```

As little as possible was compiled into the httpd binary to reduce the amount of memory used. The CFLAGS exported enabled serving of files over 2Gb in size.

File System

Chosing a fast and efficient filesystem is very important for a web server. Tests at the time showed XFS gave better performance than ext2 and ext3, up to a margin of 20%. As a caveat, as the number of used inodes in the filesystem grows, XFS becomes very slow at directory traversal. This resulted in a migration to ext3, despite the reduced perfomance.

noatime

One significant mount option, noatime, was set. This is the single easiest way to dramatically increase filesystem performance for read operations. Normally, when a file is read Unix-like systems update the inode for the file with this access time so that the time of last access is known. This operation means that read operations also involve writing to the filesystem - a severe performance bottleneck in most cases. If knowing this access time is not critical, and it certainly is not for a busy mirror server, it can be turned off by mounting the filesystem with the noatime option.

logbufs

For XFS, the logbufs mount option allows the administrator to specify how many in-memory log buffers are used.While it was not clear what these log buffers do, increasing this number to its maximum increased performance. This performance increase comes at the expense of memory which was acceptable for the overall design.

dir_index

For ext3, dir_index option is an option whereby ext3 uses hashed binary-trees to speed up lookup in directories. This has proved much faster for directory traversal.

Kernel

The system originally ran the SGI Linux 2.4 kernel which gave about 12,000 sessions as a maximum. However after simply upgrading to the 2.6 kernel the server hit the compiled-in 20,000 limit of Apache without any additional effort, so the scheduler in the 2.6 kernel appears to have markedly improved.

File Descriptors

One of the most important tunables for a large-scale webserver is the maximum number of file descriptors the system is allowed to have open at once. The default is not suffcient when serving thousands of clients. It is important to remember that regular files, sockets, pipes and the standard streams for every running process are all classed as filedescriptors and that it is easy to run out.

This figure was set to 5049800.

Virtual Memory Manager

In Linux, the VM manages the memory allocated to processes and the kernel and also manages the in-memory cache of files. By far the easiest way to “tune” the VM in this regard is to increase the amount of memory available to it. This is probably the most reliable and easy way of speeding up a webserver - add as much memory as you can afford.

The VM takes a similar approach to mod_disk_cache for freeing up space - it assigns programs memory as they request it and then periodically prunes back what can be made free. If a lot of files are being read very quickly, the rate of increase of memory usage will be very high. If this rate is so high that memory is exhausted before the VM had had a chance to free any there will be severe system instability. To correct for this 5 sysctl's were set:

```
vm/min_free_kbytes = 204800
vm/lower_zone_protection = 1024
vm/page-cluster = 20
vm/swappiness = 200
vm/vm_vfs_scan_ratio = 2
```

* The first sysctl sets the VM to aim for at least 200 Megabytes of memory to be free. * The second sysctl sets the amount of “lower zone” memory directly addressable by the CPU that should be kept free. * The third sysctl, “vm/page-cluster” tells Linux how many pages to free at a time when freeing pages. * The fourth sysctl, “swappiness,” is a very vague sysctl which seems to boil down to how much Linux “prefers” swap, or how “swappy” it should be. * The final sysctl, the “vm vfs scan ratio,” sets what proportion of the filesystem-data caches should be scanned when freeing memory. By setting this to 20 we mean that 1/20th of them should be scanned - this means that some cached data is kept longer than it otherwise would, leading to increased opportunity for re-use.

Network Stack

Six sysctl's were set relating to the network stack:

```
net/ipv4/tcp_rfc1337=1
net/ipv4/tcp_syncookies=1
net/ipv4/tcp_keepalive_time = 300
net/ipv4/tcp_max_orphans=1000
sys/net/core/rmem_default=262144
sys/net/core/rmem_max=262144
```

* TCP syncookies and the RFC1337 options were enabled for security reasons. * The default tcp keepalive time was set to 5 minutes to avoid the situation where httpd children handling connections which have not been responsive for 5 minutes are not needlessly waiting in the queue. This has the minor impact that if the client does try to continue with the TCP session at a later time it will disconnect. * The max orphans option ensures that even despite the 5 minute timeout there are never more than 1,000 processes held in such a state, and will instead start closing the sockets of the longest waiting processes. This prevents process starvation due to many broken connections. * The final two options increase the amount of memory generally available to the networking stack for queueing packets.

Hyperthreading

Hyperthreading is a technology which makes one processor show up as two with the aim of improving resource usage efficiency within the processor. The webserver was benchmarked with hyperthreading enabled and hyperthreading disabled. Hyperthreading enabled resulted in a 37% performance increase (from 721 requests per second to 989 requests per second, with the same test). It was therefore enabled.

References

Colm MacCárthaigh, *Scaling Apache 2.x beyond 20,000 concurrent downloads*, http://www.stdlib.net/~colmmacc/Apachecon-EU2005/scaling-apache-handout.pdf, ApacheCon 2005