SEE TEXT ONLY

svn UPDATE

by Steven Kreuzer

Life keeps getting in the way, so I have not been able to keep up with FreeBSD's fast-paced development. One day the sun and moon aligned and not only were both my daughters sound asleep, but I also found myself with enough energy to not only keep my eyes open, but also form cohesive thoughts. Not knowing what to do with all this free time I had on my hands, I did what any normal person would do and started to dig through the subversion logs. Imagine my surprise when I discovered that FreeBSD now supports pNFS, has a brand-new TCP congestion algorithm, and cron even gained new functionality!

Merge the pNFS server code from projects/pnfs-planb-server into head— https://svnweb.freebsd.org/changeset/base/335012

A pNFS service separates the Read/Write operations from all the other NFSv4.1 Metadata operations. It is hoped that this separation allows a pNFS service to be configured that exceeds the limits of a single NFS server for storage capacity and/or I/O bandwidth. It is possible to configure mirroring within the data servers (DSs) so that the data storage file for an MDS file will be mirrored on two or more of the DSs. When this is used, failure of a DS will not stop the pNFS service and a failed DS can be recovered, once repaired, while the pNFS service continues to operate. Although two-way mirroring would be the norm, it is possible to set a mirroring level of up to four, or the number of DSs, whichever is less. The Metadata server will always be a single point of failure, just as a single NFS server is.

Permit sysctl(8) to set an array of numeric values for a single node— https://svnweb.freebsd.org/changeset/base/330711

Most sysctl nodes only return a single value, but some nodes return an array of values (e.g., kern.cp_time). sysctl(8) understands how to display the values of a node that returns multiple values (it prints out each numeric value separated by spaces). However, until now, sysctl(8) has only been able to set sysctl nodes to a single value. This change allows sysctl to accept a new value for a numeric sysctl node that contains multiple values separated by either spaces or commas. sysctl(8) parses this list into an array of values and passes the array as the "new" value to sysctl(2).

Bring in the TCP high-precision timer system (tcp_hpts)— https://svnweb.freebsd.org/changeset/base/332770

It is the forerunner/foundational work of bringing in both Rack and BBR, which use hpts for pacing out packets. The feature is optional and requires the TCPHPTS option to be enabled before the feature will be active. TCP modules that use it must assure that the base components compile in the kernel in which they are loaded.

Bring in a new refactored TCP stack called Rack— https://svnweb.freebsd.org/changeset/base/334804

RACK ("Recent ACKnowledgment") is a time-based, fast-loss detection algorithm for TCP. RACK uses the notion of time instead of packet or sequence counts to detect losses for modern TCP implementations that can support per-packet timestamps and the selective acknowledgment (SACK) option. It is intended to replace the conventional DUPACK threshold approach and its variants, as well as other nonstandard approaches.

AF_UNIX: make unix socket locking finer grained— https://svnweb.freebsd.org/changeset/base/333744

This change moves to using a reference count across lock drop/reacquire to guarantee liveness. Currently sends on unix sockets contend heavily on read locking the list lock. unix1_processes in will-it-scale peaks at 6 processes and then declines. With this change, we see a substantial improvement in number of operations per second with 96 processes.

Fix spurious retransmit recovery on low-latency networks— https://svnweb.freebsd.org/changeset/base/333346

TCP's smoothed RTT (SRTT) can be much larger than an actual observed RTT. This can be due to hz restricting the calculable RTT to 10ms in VMs or

1ms using the default 1000hz or simply because SRTT recently incorporated a larger value. If an ACK arrives before the calculated badrxtwin (now + SRTT): tp->t_badrxtwin = ticks + (tp->t_srtt >> (TCP_RTT_SHIFT + 1)); we'll erroneously reset snd_una to snd_max. If multiple segments were dropped and this happens repeatedly, the transmit rate will be limited to 1MSS per RTO until we've retransmitted all drops.

Boost thread priority while changing CPU frequency— https://svnweb.freebsd.org/changeset/base/333325

Boost the priority of user-space threads when they set their affinity to a core to adjust its frequency. This avoids a situation where a CPU-bound kernel thread with the same affinity is running on a down-clocked core and will "block" powerd from up-clocking the core until the kernel thread yields. This can lead to poor performance and to things potentially getting stuck on Giant.

Import the netdump client code— https://svnweb.freebsd.org/changeset/base/333283

This is a component of a system which lets the kernel dump core to a remote host after a panic, rather than to a local storage device. The server component is available in the ports tree. netdump is particularly useful on diskless systems. The netdump(4) man page contains some details describing the protocol. To use netdump, the kernel must have been compiled with the NETDUMP option.

Improve VM page-queue scalability— https://svnweb.freebsd.org/changeset/base/332974

Currently both the page lock and a page-queue lock must be held in order to enqueue, dequeue or requeue a page in a given page queue. The queue locks are a scalability bottleneck in many workloads. This change reduces page-queue lock contention by batching queue operations. To detangle the page and page-queue locks, per-CPU batch queues are used to reference pages with pending queue operations. The requested operation is encoded in the page's aflags field with the page lock held, after which the page is enqueued for a deferred batch operation. Page-queue scans are similarly optimized to minimize the amount of work performed with a page-queue lock held.

Add new functionality and syntax to cron(1) to allow jobs to run at a given interval— https://svnweb.freebsd.org/changeset/base/334817

The practical goal here is to avoid overlap of previous job invocation to a new one, or to avoid too short interval(s) for jobs that last long and don't have any point of immediate launch soon after the previous run. Another useful effect of interval jobs can be noticed when a cluster of machines periodically communicates with a single node. Running the task time-based creates too much load on the node. Running interval-based spreads invocations across machines in cluster.

Load balance sockets with new SO_REUSEPORT_LB option— https://svnweb.freebsd.org/changeset/base/332894

This patch adds a new socket option, SO_REUSEPORT_LB, which allows multiple programs or threads to bind to the same port and incoming connections will be load balanced using a hash function.

Add the TCP Blackbox Recorder— https://svnweb.freebsd.org/changeset/base/331347

The TCP Blackbox Recorder allows you to capture events on a TCP connection in a ring buffer. It stores metadata with the event. It optionally stores the TCP header associated with an event (if the event is associated with a packet) and also optionally stores information on the sockets. It supports setting a log ID on a TCP connection and using this to correlate multiple connections that share a common log ID. You can log connections in different modes. If you are doing a coordinated test with a particular connection, you may tell the system to put it in mode 4 (continuous dump). Or, if you just want to monitor for errors, you can put it in mode 1 (ring buffer) and dump all the ring buffers associated with the connection ID when we receive an error signal for that connection ID. You can set a default mode that will be applied to a particular ratio of incoming connections. You can also manually set a mode using a socket option.

STEVEN KREUZER is a FreeBSD Developer and Unix Systems Administrator with an interest in retro-computing and air-cooled Volkswagens. He lives in Queens, New York, with his wife, daughter, and dog.