BSDCan 2009: The Technical BSD Conference

Ottawa, Canada
May 6–9, 2009

Summarized by Royce Williams (royce@tycho.org)

Slides for most of the presentations are available at http://bsdcan.org/2009/.

KEYNOTE ADDRESS

- ***Thinking about Thinking in Code***
  George V. Neville-Neil, Neville-Neil Consulting

In what he described as "a bit of a rant," George Neville-Neil challenged the BSD development community to think about their work in a different way.

Neville-Neil started by attacking the idea that software development is significantly more creative than, for example, automobile manufacturing. He pointed out that there has been little true innovation in graphical user interface design, showing similarities in GUIs ranging from the Xerox PARC user interface through Mac OS X. He discarded traditional explanations such as blaming marketing or that users demand front-end consistency. Even OS internals, he argued, have not substantially changed and do not fundamentally differ among the major families of operating systems. He stated that the languages that we work with truly dictate our work, that features of bad languages (sloppy, unsafe, confusing) lead to code that follows suit, and that making programming languages easier has effectively lowered the quality of code (by lowering the barrier to entry).

In a flurry of frank advice to programmers, Neville-Neil went on to encourage reading good code, working with good programmers (rather than poor ones, which he argues can actually cause your own code to suffer), and refraining from repeatedly reinventing the wheel by recreating low-level constructs (like lists, hashes, and other academic projects). Instead, he suggested reading research papers discriminatingly, exploring unfamiliar code and languages, avoiding too much specialization, and cultivating a willingness to "break things, look stupid, be wrong, learn from others." He warned against hubris, not starting projects, or never finishing them. He pointed out that we all have a finite amount of time to live, so finding people who will tell you when your idea is bad so that you can quickly move on to the next one is very important. In closing, he suggested seeking to reduce complexity, using visualization tools and new data organization methods, and working with safe yet powerful programming languages.

- ***Automating FreeBSD Installations: PXE Booting and install.cfg Demystified***
  Randi Harper, IronPort/Cisco

Randi Harper reviewed some of the common issues with customizing FreeBSD's sysinstall configuration (the install .cfg file) and installing FreeBSD over PXE. A basic walkthrough followed, noting common stumbling blocks along the way (e.g., that trailing whitespace in the install.cfg file can cause problems, and that some variables are case-sensitive). Since install.cfg is not well documented, reading the source was necessary.

Harper covered the entire sysinstall/PXE installation process. Steps included setting up the ISC dhcpd package; configuring supporting services (tftp via inetd, NFS); copying the contents of a FreeBSD installation CD to a staging area; and using mdconfig to mount the included mfsroot image in order to customize the install.cfg file within. Customization options included running in full automatic/unattended mode, specifying a single NIC to use, and optionally specifying packages to add post-install. The current system requires that the NIC type used for installation be known in advance, making it necessary to customize the install.cfg for different hardware families. Harper is working on adding support for a list of multiple NICs, tried in succession, to reduce the number of separate configuration profiles.

During the question period, the topic of how to avoid building the same system twice was raised. Matt Olander of ixSystems asked about the feasibility of knowing the MAC addresses of each system in advance. Harper replied that such asset management is usually already part of large-scale deployments. Olander noted that his environment consists of setting up large groups of systems and then shipping them to the end customer as quickly as possible, making such inventory work infeasible. Discussion followed about keeping a custom text file to incorporate into the dhcpd. conf file to track "state" (the MACs of successfully built systems).

- ***GEOM_SCHED: A Framework for Disk Scheduling within GEOM***
  Luigi Rizzo and Fabio Checconi, University of Pisa

Luigi Rizzo presented GEOM_SCHED, a disk-scheduling framework built on GEOM, the FreeBSD storage abstraction layer. FreeBSD now uses a primitive elevator/C-LOOK-based scheduler. While there has been previous work on disk scheduling in FreeBSD, it has not been committed to the base OS. Rizzo speculated that this might be due to the previous implementations being device-specific and that disk schedulers based on the GEOM framework could make development easier.

In turn, Rizzo examined the merits of each potential location to place a disk scheduler (the disk device, the device driver, and GEOM). He concluded that GEOM is a good option, because it does not require hardware awareness or driver modification, provides a single point of control, and provides for transparent insertion and removal, as well as runtime reconfiguration.

Another design goal was minimal kernel reconfiguration. Since GEOM_SCHED is not included in the base FreeBSD system, the existing implementation dynamically patches g_io_request() to repurpose some unused fields in the structure. It is otherwise implemented entirely outside of the GENERIC kernel as a userland object, a generic kernel module, and one or more kernel modules.

Rizzo went on to cover the GEOM_SCHED API, basic disk-scheduling concepts, and his measurement methodology, especially noting the hazards of measuring disk I/O performance when the caching and read-ahead policies used by drivers and firmware are sometimes not known.

Rizzo's example scheduler was a straightforward implementation of round-robin queues, with anticipation (in which seeks are delayed in case non-seek activity arrives soon after, and then grouped). He presented the results of his testing for various workloads. Even with the slight overhead caused by GEOM, disk performance for multiple greedy readers was significantly improved. He encouraged others to start from this basic framework, applying other algorithms for other workloads. His prototype is available at http:// info.iet.unipi.it/~luigi/FreeBSD/.

Robert Watson asked about the interaction between disk scheduling and process prioritization, referencing previous work that showed that particular I/O patterns (such as an fsck) can suffer in surprising ways when interacting with process scheduling. Rizzo encouraged further research in this area.

- ***Getting Started in Free and Open Source***
  Cat Allman and Leslie Hawthorn, Google

Cat Allman and Leslie Hawthorn took turns presenting ideas in a tag-team fashion. Since there were many in the audience who were decidedly not new to open source, they partially adjusted their talk to address how current members of open source projects can better understand and attract new contributors.

High points included coming as close as possible to "going back to being new" (by vicariously mentoring newcomers); recognizing that thorny problem areas (like bug wrangling) can be opportunities for ways to participate; understanding that FOSS projects are inherently reputation-based economies; designating a "newbie wrangler" (either some-one talented in this area or as a rotating responsibility) to protect people from burning out on hand-holding; creating a culture tolerant of failure and mistakes to aid growth; and not assuming that newcomers who make mistakes early will remain permanently clueless.

- ***Updates to the the FreeBSD Problem Reporting System***
  Mark Linimon, Lonesome Dove Computing Services

Mark Linimon, primary "bugmeister" for FreeBSD, proposed an initial conceptual prototype to start work toward a new system of problem reporting (PR) for FreeBSD, working under a grant from the FreeBSD Foundation. In broad terms, lessons learned from the current system will be applied to a temporary prototype to model this new workflow.

Linimon has discovered some specific areas for improvement. Some PR states (e.g., "patched" and "closed") are used consistently, while others ("feedback," "analyzed") are overloaded, which has been confusing enough to throttle PR throughput rates already hampered by resource limitations. Linimon reviewed the workflow categories used by similar frameworks (Bugzilla, Jira, and Trac) and, based on that review, has created distinct stages in the model for triage, submitter coordination, and development work, each of which can be worked by different people with different levels of skill.

As presented, there are other opportunities for improvement. Notifications are too broad, and none of the alternative systems allow developers to limit notifications to specific subsystems of interest or specialty. Current category names were chosen with developers in mind (and can be misunderstood by submitters). Linimon also identified a family of PRs that do not fit easily into the current system, including booting, installation, and performance issues, and proposed a "Usability" category to group them conceptually.

An emerging property of the recent system was that adding tagging support resulted in people using the relevant subsystem man pages as tags. Linimon has added a specific separate field in the prototype that is being populated using the man page names.

Linimon has chosen Jira as the prototype platform (but was careful to note that this does not mean that Jira will be selected). He will be applying these ideas to Jira and seeking feedback. People interested in helping were directed to the FreeBSD wiki's BugBusting page to coordinate.

- ***scrypt: A New Key Derivation Function***
  Colin Percival, Tarsnap

To provide some background, Percival started with an overview of encryption key derivation functions. KDFs are commonly used to hash passwords for secure storage and to generate cryptographic keys. Examples include the classic DES CRYPT, Poul-Henning Kamp's iterated MD5 CRYPT, PBKDF2, and bcrypt.

These (and most other) preceding KDFs have focused on raising the cost of "dollar-hours" by maximizing the amount of CPU time required to run. However, well-funded groups can afford farms of custom dedicated ASICs, each with thousands of cores optimized for specific cryptographic operations.

Percival's tagline for this presentation was "Doing our best to thwart TLAs with ASICs." Percival noted that the cost of an ASIC is roughly matched with its size and that large amounts of RAM can take up a significant amount of ASIC space. He reasoned that functions which both require very large amounts of RAM ("memory-hard" functions) and are not easily broken down to run in parallel ("sequential" functions) would increase the required size (and number) of dedicated ASICs, thereby significantly increasing the corresponding cost.

To enable such functions, Percival introduced a provably sequential memory-hard problem, ROMix, which fills a hash table with pseudo-random values and then accesses them in a pseudo-random order. In the accompanying paper Percival proves that any algorithm that correctly implements ROMix will be sequential memory-hard, but in the talk he left a review of the two-page proof as an exercise for interested listeners.

Percival then presented scrypt itself, which is a combination of PBKDF2, an algorithm that solves a given ROMix problem, HMAC-SHA256, and Daniel J. Bernstein's Salsa20 cipher to carry out the key derivation while also quickly requiring large amounts of RAM. Like other KDFs, scrypt takes parameters that can be used to adjust its costs to run, so the maximum amount of RAM and maximum time in seconds can be varied.

In order to illustrate the difference in strength, Percival estimated some real-world costs. Since entities in the business of brute-force attacks do not publish hardware costs, Percival also presented the assumptions he used for comparing algorithms. He noted that, even if off by orders of magnitude, these estimates nevertheless held constant among the functions and therefore are useful for relative comparisons.

Using the provided numbers and stating the parameters used (for the functions that take them), Percival compared DES CRYPT, MD5 (as a reference point, not useful for actual encryption purposes), MD5 CRYPT, PBKDF2, bcrypt, and scrypt. For example, an 8-character password with good entropy, encrypted with scrypt in .1 seconds (good enough for authentication speeds), will take roughly $4.8M to crack within one year, while bcrypt would cost only $130K. For longer times suitable for encrypting data (around 5 seconds), the 8-character costs jump to $4.3M for bcrypt (3.0s) and $19B for scrypt (3.8s).

While doing research for this work, Percival discovered that OpenSSL uses simple MD5 hashing as its key derivation function, and OpenSSH also uses simple MD5 for keyfile passphrases. This may be of some concern for people who carry their SSH keys on pocket USB devices.

During the question period, Brooks Davis asked about the feasibility of imposing a large per-transaction memory cost on systems used for high-volume authentication or which are subject to authentication floods. Percival replied that the function takes a number of parameters to adjust the CPU and memory cost of each calculation to fit the target platform and work load.

Percival's paper describing scrypt in more detail (including proofs), the full estimate comparison, and a cross-platform BSD-licensed implementation of scrypt are all available at tarsnap.com/scrypt/.

- ***Works in Progress Session (also known as the "lightning round")***
  Chaired by Robert Watson

Colin Percival announced that he is interested in reviving the project to build concurrency-awareness into the FreeBSD rc.d system. Now that multicore systems are the norm, startup times could be significantly improved. Interested contributors are encouraged to contact Colin.

Scott Ullrich of the pfSense Project outlined features of the BSD Installer, a proposed unified installer for all BSDs, and the installer used by Dragonfly BSD and the upcoming version of pfSense. Features include a clear separation between the front end and back end, enabling multiple possible front ends. Recent work is focusing on adding the remaining functionality included in FreeBSD's sysinstall and the PC-BSD installer, but missing from the BSD installer.

Philip Mullis briefly mentioned a new effort to create an independent VoIP peering exchange. More information will eventually be available at nopstn.net.

Zach Loafman of Isilon described kernel fault injection, a new set of APIs used to insert specific user-controlled errors at particular points in FreeBSD code ("failpoints"). The APIs include the ability to assign the errors' probabilities of occurring or a cap on how many times they occur. Isilon is using hundreds of these "failpoints" in production, and Loafman is working to get support for them committed to FreeBSD.

John Baldwin first gave a status report about the upcoming FreeBSD 8.0. New features include virtual network stacks, MIPS support, NFSv4, ECMP (which enables support for kernel awareness of multiple routing tables and default routes), virtual wireless access points, a reworked USB stack, support for 32-bit FreeBSD 8 as a Xen dom-U guest, and improved Linux binary compatibility. FreeBSD 8.0 is scheduled for release at the end of August.

Baldwin also talked briefly about extensions to device mmap() support, largely driven by the memory-mapping needs of modern GPUs. In the amd64 and i386 ports, this will be implemented via PAT (Page Attribute Table, required for good PCI-Express performance) and will pave the way for an Nvidia amd64 driver for FreeBSD.

John Birrell of Juniper Networks described jbuild, a new FreeBSD build system that eliminates multiple layers of redundant dependency calculation, significantly reducing build times. This is accomplished by front-loading the master jbuild process with all dependency information from the build directory.

Doug Rabson presented updates about FreeBSD on Xen. The included XEN and XENHVM kernel configs in FreeBSD-current are the best place to start experimenting with para-virtualization and hardware virtualization, respectively.

Rabson also gave a quick how-to about booting from ZFS and mapped out his planned future ZFS work, including teaching the FreeBSD installer about ZFS.

Warner Losh talked about recent progress with the various flavors of the MIPS port, working on the RMI XLR/XLS, RMI Alchemy, Cavium Octeon1, and Atheros AR71xx/91xx chips, using reference boards supplied by various sources.