BSDCan 2008 FreeBSD Developer Summit

Ottawa, Canada
May 14–15, 2008

Summarized by Bjoern A. Zeeb (bz@FreeBSD.org) and Marshall Kirk McKusick (McKusick@McKusick.com)

INTRODUCTION

A developer summit is a get-together of some FreeBSD developers to present recently finished work and to talk about ongoing work or future plans. Although this is not a place to make decisions concerning the entire project, there are usually enough of the key developers present that a lot of design issues get hashed out.

Developer summits are often aligned with a BSD conference, as many developers are going to be there anyway. The summit also provides an opportunity for people from companies using FreeBSD or other FreeBSD-derived projects to attend. This mix of developers and users presents a great opportunity for the developers to get direct feedback on their work and for the users to gain an understanding of likely future developments.

This report gives our thoughts on the developer summit events that we attended. For more complete details and often the slides used by the presenter, see the Developer Summit wiki page, http://wiki.FreeBSD.org/200805DevSummit.

The format of the summit was to have formal talks in the morning, with afternoons devoted to free-for-all discussions by smaller, self-selected groups on various specialized topics largely selected at the conclusion of the morning talks. Most of this report will focus on the morning sessions, as their content is more easily summarized.

SUMMIT DAY 1

The day opened with a welcome message from Robert Watson, the main organizer for the summit.

Poul-Henning Kamp started with a general discussion about GreenBSD. This project will not be a fork of FreeBSD, but it might become a new, large-scale project encompassing large parts of the system. The main thrust of GreenBSD is "green computing," that is, reducing power consumption whenever and wherever possible. Examples include turning down gigabit interfaces to 100 Mbit/s if there is no need for more speed, entirely shutting down unused NICs, or reducing clock speeds and voltages on otherwise idle CPUs. The key is to figure out the correct abstraction and support functionality so that it is not necessary to endlessly duplicate code in each device driver.

Next, Ivan Voras spoke about his Google Summer of Code 2007 project, finstall, a graphical installer that provides more automated decision making and provides interfaces to more parts of the systems such as geom, networking options, ZFS, and gjournal. It is split into a front end and a back end and is written in GTK and Python. The split allows finstall to implement a remote installation console. The back end can be used for non-base-system installs as well. At the moment it is still lacking a graphical partition editor.

During the question period Ivan was asked how much work would be required to replace the Python back end with a C implementation. Concerns were also raised about the front end using GTK because of it being LGPL. In particular, the discussion migrated to whether the front end could be replaced with a text-mode-only implementation in C and whether that would really just drop back to the current capabilities of the curses-based front end.

After a short break Alexey Tarasov spoke on his Google Summer of Code 2007 project, which involved kernel netbooting via HTTP. The work consists of a PXE API, a tiny TCP/IP stack, PXE sockets, etc. Unfortunately, there is no IPv6 support with PXE. There are basic filters to restrict IP/ports. The most important but also complicated task had been the user mode implementation of the tiny TCP/IP stack along with memory constraints (buffering issues). It uses httpfs provided by the boot loader including HTTP 1.1 support. DHCP, DNS client, ICMP echo, ARP, and boot console commands are already implemented. Work in progress includes a telnet client, socketfs, IPv6 support, and a possible parsing of HTTP server index pages.

Rafal Jaworowski was up next with an embedded-architecture status report: arm, MIPS, and PowerPC. The arm port is "almost" tier-1, as of the FreeBSD 7.0 release. More arm functions and features are in the pipeline. Unlike the other architectures, it is nearly impossible to have a single arm GENERIC kernel as there are so many incompatible variations of the arm architecture. Juniper Networks has been providing a lot of the MIPS support. FreeBSD 7.0 supports both the 32-bit and 64-bit MIPS architectures. As with the arm architecture, much additional MIPS functionality is in the pipeline.

The PowerPC is more of a work in progress, also being supported by Juniper. At the moment most of the PowerPC support is for the high-end chipsets and SMP support. Bridge mode is going toward 64-bit PowerPC support.

There are two Google Summer of Code 2008 embedded-systems projects. The first involves optimizing the build system for embedded systems. The second is to port to Efika, a cheap platform. Other embedded projects on the wish list are improving support for a flash-memory-based filesystem, an improved build system to support cross-building from Linux or Windows, and better system/kernel configuration for creating a smaller footprint.

Ed Schouten talked about reimplementing FreeBSD's tty layer. The tty system is the one part of the system that has not been rewritten to support SMP so still needs to run under the Giant lock. Ed started with a design overview, which involved the removal of the fragile clists buffer mechanism from tty. Among other things he is now destroying ptys when unused so that they do not clutter up /dev and consume kernel resources. He multi-threaded the transmit path buffering and eliminated the global buffer list. Best of all, he managed to keep all but sgetty ABI compatible. Still to be done is the (fairly mechanical) task of adding multi-threading support to all the serial devices.

This concluded the first day's formal presentations. After a lunch of pizza everyone broke up into smaller discussion groups. The ``Network Cabal'' started with Jeff Roberson, Julian Elischer, and Kip Macy on a redesign for mbufs. Jeff has a new concept that ref-counts mbufs so that they require less copying on forks. In addition he combines a clustered mbuf with a small mbuf header to increase efficiency when a copy of the mbuf needs to be retained, for example with TCP. The ongoing discussion was mostly with Sam Leffler and Robert Watson debating mbuf layout, mbuf tags, mbuf back traces, and techniques for tracking mbuf leaks.

Lawrence Steward led a discussion about TCP bug forensics. After introducing himself and a bit of TCP jargon and history he delved into congestion control algorithms. He explained which OS is using which algorithm and talked about what parameters to look at when comparing the various different algorithms for high-speed connections.

Next he turned his attention to tools. Dummynet has problems. SIFTR is their tool to generate CSV-like information for later analysis. He showed how to use the data of the tool in three interesting case studies of problems they had found.

The next slot was debugging and profiling tools. John Birrell led the discussion by opening with a demo on DTrace. One of the important things that he stressed was that DTrace is not a debugger. One key thing to understand about DTrace is that it can always be compiled into the kernel as it has almost no overhead when it is not being used. Indeed the (only) overhead is NULL pointer checks for the DTrace modules. If the modules are loaded, there will be an additional bitwise AND to do a mask check. So, one can expect to have DTrace available even on production systems. The other key point that came up is that DTrace does not replace other tools such as ktrace and gprof.

The last big section of the first day was a presentation on network stack virtualization by Marco Zec and Julian Elischer. They explained how they had implemented loadable network-stack support. In short, all the formerly global data structures and variables had been gathered together into a dynamically allocated structure. In this way, multiple copies of the network stack could run in isolation; for example, each jail could have its own network stack on which to operate. This functionality allows each jail to run its own packet filter, raw sockets, ICMP, ALTQ, etc.

SUMMIT DAY 2

Adrian Chadd started off the second day of presentations by talking about TCP content- and service-provider hijacking. He discussed both malicious and deliberate hijacking and described the different methods and technologies used. He explained the various problems that arise when deliberate hijacking is done. The issues involved TCP options, screw-ups in MTU discovery, failures in properly setting TCP options, and the effects of an older implementation such as TPROXY.

Next Doug Rabson talked about his work on replacing the error-prone userland NFS Lock Manager with a kernel-based one; this led to a lot of cheering. He started with a basic overview of the different NFS versions (2/3) and undocumented newer stuff. FreeBSD locking used to be done in the old userland rcp.lockd, which lacked proper client-side locking. The new kernel-based rcp.lockd supports everything but DOS shares. It has kernel-mode implementations for both client and server. Local locking now supports asynchronous operation and there is a graph-based deadlock detection. He also implemented fairness for contested locks so that locks are handed out first come, first served. He even added regression tests to ensure that it works and continues to work. The kernel-mode implementation is now the default, but there are options in GENERIC so that you can opt out of the kernel-mode lock manager and fall back to the “old rpc.lockd.”

Justin Gibbs provided a break in the onslaught of technical information by giving an update on the FreeBSD Foundation. He explained that the foundation was created to provide a way to channel money to fund development of unpopular parts of the system, as a way to build long-standing relationships with vendors and to provide an organization that can negotiate legal contracts. By providing a standalone organization rather than affiliating with an existing corporation they avoided any conflicts of interest. Thus, the FreeBSD Foundation is an independent corporation with management that is internally elected, and its activities are guided by its charter, which states its role as improving, nurturing, protecting, and evangelizing FreeBSD. In short, it is the “tie that binds” FreeBSD.

The FreeBSD Foundation is providing travel grants, event sponsorship, funds development, IP protection, legal and contract negotiation, and management of hardware donations. Challenges include knowing the user base, maintaining critical mass, finding funding for the FreeBSD platform, helping to define and set milestones, and growing the capabilities of FreeBSD.

Returning to the technical theme, Erwin Lansing talked about the task of the FreeBSD port manager. He gave some numbers and statistics and went into the details of the port monitoring software. As there are now over 18,000 actively maintained ports, many manual tasks have to be handled automatically. There is software for tracking problem reports, detecting maintainer-timeouts, ensuring that packages compile on all platforms, determining package dependencies, etc. This summer three Google Summer of Code students will work on the ports infrastructure.

Robert Watson stepped up the pace of discussion with his talk on TCP scalability in the presence of 16-core SMP systems. He started with the big picture: MPSAFEness, Giant free, and improving multi-threaded workloads. He went on to describe UDP problems: throttling by exclusive write locks, and excessive overhead from the socket buffer code. Once these were fixed, the bottleneck moved to the routing code, which has no parallelism. Streamlining the routing code led to the transmit queues, which need to be serialized to preserve ordering. The trick seems to be to serialize them only per connection and not for all UDP traffic, which is easier said than done.

The TCP stack has even more bottlenecks, including one lock for all incoming packets, serialized access to look up the connection for which a packet is destined, along with socket buffer send/receive, routing, and the transmit queues noted in UDP. He talked more about stack parallelism, direct dispatch versus input queuing, and maintaining per-connection ordering when running with multireceive and multisend queues.

The last formal talk was by Peter Wemm on Version Control. Whereas most agree that CVS has hit a wall, it was much less clear what should replace it. After much investigation Peter concluded that Subversion would be the best replacement. The command-line interface is nearly the same as that of CVS except that it uses URLs instead of paths in some places.

After the initial changeover, all changes into Subversion will be reflected into the CVS tree. That means cvsup will still work and there will be almost no visible changes for the world apart from minor things such as slightly different commit messages. That also means that cvsweb would still work and that there would be a backup plan in case Subversion does not work out. The project could just switch back to CVS.

The first after-lunch discussion group was on system-trust issues. Topics included the mandatory-access framework, auditing activities within jails, and increasing the granularity of privilege: getting away from all (root) or nothing (all other users).

Jeff Robinson led a discussion of an overhaul of the buffer cache. Much of the functionality formerly provided by the buffer cache (caching, identity, clustering, etc.) is now provided by the virtual-memory system. Jeff talked about which functionality remained in the buffer cache and how to push that functionality elsewhere so that the remains of the buffer-cache interface can be eliminated. Details are at http://wiki.FreeBSD.org/Buf0x.

The day ended with a long reprise of the network-stack virtualization discussion, getting down to the specifics of defining a set of steps toward its realization, ordering those steps, and setting a timeframe for each step.