MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

Building a PCI Compliant Infrastructure on AWS

AWS Infrastructure Security

Deep Dive Into Access Control Management

UNIX BLOG PRESENTATION

Hubert Feyrer

Interview with Kalin Staykov

Elastix on Bhyve

Password Cracking in UNIX

Raspberry Pi 3

VOL. 11 NO. 03
ISSUE 03/2017 (91)
ISSN 1898-9144

Table of Contents

UNIX BLOG PRESENTATION

hubertf's NetBSD Blog

Dr. Hubert Feyrer

His studies of technical computer science at the University of Applied Sciences (FH) Regensburg were followed by employment as a Unix/Solaris/NetBSD system administrator and as a computer science teacher, both at the University of Applied Sciences Regensburg as well as Stevens Institute of Technology in Hoboken, NJ, USA. After receiving a Ph.D. in Information Science from the University of Regensburg start as developer of hard- and software as well as network and security solutions, with next promotion to IT manager (CTO). As such, performing security consulting according to ISO 27001 and later changing into the automotive sector as a Chief Information Security Officer (CISO). Recent occupations include working as CISO for one of Germany`s largest process- and people service provider, currently responsible as CISO for a major German car manufacturer.

For more information, visit http://www.feyrer.de/

Some time ago, I tried to solve some real-world geocaching/math problem. In my brute-force approach, I allocated a number of jobs on multiple-CPU machines rented from Amazon AWS and were running NetBSD/Xen. After doing so, I discovered that the load distribution was not utilizing all CPUs. NetBSD's processor sets were a first attempt to a workaround, but I wanted a proper fix, as this was apparently not specific to the Xen port but affected the NetBSD scheduler on all platforms. NetBSD developer Michael van Elst hinted me at the real problem, and he also provided the first patch to the problem. I did setup a test environment and ran some tests, and documented my findings in what is one of my favorite blog posts:

Learning More About the NetBSD Scheduler (... Than I Wanted to Know)

I had another chat with Michael on the scheduler issue, and we agreed that someone should review his proposed patch. Some interesting things arose from the discussion:

I learned a bit more about the scheduler from Michael. With multiple CPUs, each CPU has a queue of processes that are either "on the CPU" (running) or waiting to be serviced (run) on that CPU. Those processes count as "migratable" in runqueue_t. Now and then, the system checks all its run queues to see if a CPU is idle, and can thus "steal" (migrate) processes from a busy CPU. This is done in sched_balance().
The "stealing" (migration) has a positive effect in that; the process doesn't have to wait to get serviced on the CPU it's currently on. On the other side, migrating the process affects both CPU's data and instruction caches. Therefore, switching CPUs shouldn't be taken too easy.

If migration happens, then this should be done from the CPU with the most processes that are waiting for CPU time. In this calculation, not only should the current number be counted in but also a bit of the CPU's history ishould be taken into account. So processes that started running on a CPU are not again taken away immediately. This is what is done with the help of the processes currently on migratable (r_mcount) and some "historic" average. This "historic" value is

taken from the previous round in r_avgcount. More or less weight can be given to this, and it seems that the current number of migratable processes had too little weight over all processes to be considered.

What happens in effect is that a process is not taken from its CPU, left there waiting, with another CPU spinning idle. Which is exactly what I saw in the first place.

Also, what I learned from Michael was that there are a number of sysctl variables that can be used to influence the scheduler. Those are available under the "kern.sched" sysctl-tree:

```
% sysctl -d kern.sched
kern.sched.cacheht_time: Cache hotness time (in ticks)
kern.sched.balance_period: Balance period (in ticks)
kern.sched.min_catch: Minimal count of threads for catching
kern.sched.timesoftints: Track CPU time for soft interrupts
kern.sched.kpreempt_pri: Minimum priority to trigger kernel preemption
kern.sched.upreempt_pri: Minimum priority to trigger user preemption
kern.sched.rtts: Round-robin time quantum (in milliseconds)
kern.sched.pri_min: Minimal POSIX real-time priority
kern.sched.pri_max: Maximal POSIX real-time priority
```

The above text shows that much more can be written about the scheduler and its whereabouts. However, this remains to be done by someone else (volunteers are welcome!).

Now, while digging into this, I also learned that I'm not the first to discover this issue; and there already exists another a PR on this. I have opened PR kern/51615, but there is also a kern/43561. Funny enough, the solution which has been proposed there is about the same, though with a slightly different implementation. Still, *2 and <<1 are the same as /2 and >>1, so there is no change. And renaming variables for fun doesn't count anyways. ;) Last but not least, it's worth noting that this whole issue is not Xen-specific.

Thus, with this in mind, I went to do a bit of testing. I had already tested running concurrent, long-running processes that did use up all the CPU they got, and the test was good.
To test a different load on the system, I started a "build.sh -j8" on a (VMware Fusion) VM with 4 CPUs on a MacBook Pro. That nearly brought the machine to a halt - Though, what I saw was lots of idle time on all CPUs. I aborted the exercise to get me back some CPU cycles. I blame the VM handling here, not the guest operating system.

I restarted the exercise with 2 CPUs in the same VM, and there I saw load distribution on both CPUs (not much wonder with -j8), but there were also quite some idle times in the 'make clean / install' phases, which I'm not sure if it is normal. During the actual build phases I wasn't able to see idle time, though the system spent quite some time in the kernel (system). Example top(1) output:

```
load averages:  9.01,  8.60,  7.15;               up 0+01:24:11      01:19:33
  67 processes: 7 runnable, 58 sleeping, 2 on CPU
  CPU0 states:  0.0% user, 55.4% nice, 44.6% system,  0.0% interrupt,  0.0% idle
  CPU1 states:  0.0% user, 69.3% nice, 30.7% system,  0.0% interrupt,  0.0% idle
  Memory: 311M Act, 99M Inact, 6736K Wired, 23M Exec, 322M File, 395M Free
  Swap: 1536M Total, 21M Used, 1516M Free

  PID USERNAME PRI NICE   SIZE   RES STATE      TIME   WCPU    CPU COMMAND
  27028 feyrer    20    5    62M   27M CPU/1      0:00  9.74%  0.93% cc1
    728 feyrer    85    0    78M 3808K select/1   1:03  0.73%  0.73% sshd
```

```
23274 feyrer    21    5    36M   14M RUN/0      0:00 10.00%  0.49% cc1
21634 feyrer    20    5    44M   20M RUN/0      0:00  7.00%  0.34% cc1
24697 feyrer    77    5  7988K 2480K select/1   0:00  0.31%  0.15% nbmake
24964 feyrer    74    5    11M 5496K select/1   0:00  0.44%  0.15% nbmake
18221 feyrer    21    5    49M   15M RUN/0      0:00  2.00%  0.10% cc1
14513 feyrer    20    5    43M   16M RUN/0      0:00  2.00%  0.10% cc1
  518 feyrer    43    0    15M 1764K CPU/0      0:02  0.00%  0.00% top
20842 feyrer    21    5  6992K  340K RUN/0      0:00  0.00%  0.00% x86_64--netb
16215 feyrer    21    5    28M  172K RUN/0      0:00  0.00%  0.00% cc1
 8922 feyrer    20    5    51M   14M RUN/0      0:00  0.00%  0.00% cc1
```

All in all, I'd say the patch is a good step forward from the current situation, which does not properly distribute pure CPU hogs, at all.

MEET Dr. Hubert Feyrer

Can you tell our readers about yourself and your blog?

Sure! I'm Hubert Feyrer and I encountered NetBSD while studying computer science. Back then, I came from the Amiga, and lots of people then ported Unix software to the Amiga. There was a "real" Unix available for the Amiga from Commodore, but it was an expensive commercial software. A group of enthusiasts tried to port Mach, but theirs was a little progress. Then one morning, Markus Wild turned up with a port of NetBSD to the Amiga, and that got the ball rolling - back in 1993. I stuck around at the University of Applied Sciences in Regensburg for quite some time, switching between studying, working at the CS department, working on my PhD and more work at the CS department. This also gave me lots of room to work on NetBSD. My blog at http://www.feyrer.de/NetBSD/blog.html is (as you may guess) about NetBSD. The first posts go back to 2004, and the link collection that it started from and that is still available go back to 1996.

How you first got involved in blogging?

When I was in academia and Linux started to grow big, attracting people long after NetBSD was popular in the (comparatively) tiny Amiga scene/ Then, dedicated Open Source / Linux events became en vogue. When I was invited to many of these events, I showed people what NetBSD was, and back then when the Internet came into fashion, I also put up some early NetBSD-related web pages - e.g. the postcard that I got in 1993 thanking me for NetBSD help. Things started out as a list of URLs of my own and other people, and this evolved into my NetBSD blog. See http://www.feyrer.de/NetBSD/ for that page which still serves as today's visual template for my blog.

At first, everything was home-written software. Later, I switched to Blosxom for its simplicity. I know there are a lot better and newer software, but Blosxom fits my purpose pretty well. One very nice feature of Blosxom is to add tags to postings, and I maintain a tag cloud of topics related to NetBSD - see the bottom of http://www.feyrer.de/NetBSD/blog.html. I had the idea to work this into a documentation system, but this never got beyond adding a few standard tags - see http://www.feyrer.de/NetBSD/bx/tags.html.

I use homegrown software for tracking progress on my blog and other selected pages, and I'm happy to see that it's still pretty highly frequented, even though my time for NetBSD has decreased somewhat after leaving academia. Today's access count is about 5.000 to 6.000 hits on my entire blog. My blog is also one of those aggregated at http://netbsd.fi,

and I'm always happy to add small blog posts anytime, knowing that it will merge into the bigger stream of NetBSD-related content there.

What's the best thing a blogger can give to his readers?

Insight and wisdom. Many of my blog posts are there to index external URLs with tags for easier retrieval via the tag cloud. But I really enjoy when I can gain new insights myself, pass on that wisdom and see people get back to me on that.

Everyone has a favorite/least favorite post. Name yours and why?

Least... besides many things, I'm the author of g4u, a hard disk cloning software (see http://www.feyrer.de/g4u). Some time ago, the core code was taken by someone and incorporated their Linux-based derivative under GPL, without giving any credits. I did an analysis of this and shared the findings in my blog post as a documentation of the case. This example of my BSD licensed code being relicensed without my consent was unpleasant, and it got me to join many discussions. If you want a URL to look into this mess, start here:
http://www.feyrer.de/NetBSD/blog.html/nb_20040917.html

Favorite... wow, so many! We're talking about 13 years. Maybe one from the recent past: while solving a geocaching/math problem, I stumbled across a problem in NetBSD's scheduler, and worked with people from the NetBSD community to understand the issue, get a proposed fix. Actually, get that one tested and into the tree, and then adding proper documentation. This was spread out over several blog posts (plus some NetBSD problem reports and mailing list postings). In the end, the posting with the (my!) core enlightenment is this one: ``Learning more about the NetBSD scheduler (... than I wanted to know)'', URL:

http://www.feyrer.de/NetBSD/blog.html/nb_20161113_0122.html

This story even made it to BSDtoday, a BSD-focused webcast. See
http://www.feyrer.de/NetBSD/blog.html/nb_20161124_2128.html :-)

What reason is there to choose Unix over another operating system from a programmer perspective?

That's a broad question. What context do we watch this in - Windows and its whole Sharepoint / Excel / Visual-Whatever ecosystem? I'm afraid few people trapped in that universe either have an idea about the beauty and simplicity of Unix, or (probably more likely) they don't have a choice on the operating system they prefer to use. Greetings from the corporate world! :-) If you start out without any prior constraints, there are again many modern systems that are completely agnostic to the underlying operating system, in the web page business. There is a huge ecosystem around PHP, Ruby and Javascript that work pretty well on any Unix system, but also (more or less) on Windows. Coming from the former, the preference of Unix for these platforms is obvious. As an exercise, try automating SSH from your favorite scripting language on a Unix platform, and then see if it still works on Windows. Good luck! Going even deeper, leaving out all those existing frameworks with their layers of complexity added on top of the operating system: an operating system is a software which gives an abstraction to hardware. It comes with an interface to talk to the Unix shell and the C API in our favorite case. Those are simple and elegant, and can be combined to do many jobs. Like all those frameworks that make so many wonderful things possible - all those Unix-based Web Pages, desktop environments like GNOME and KDE, databases, and lots of hardware where you don't even know, there is Unix inside.

Of course, there is quite some insight in the "Unix" camp today with major players like Linux and the ones in the BSD camp. And while they all have great features of their own, what made them great are the simple concepts that come with Unix. At this point, I'd like to recommend an excellent book for those interested in the original Unix concepts, ``The UNIX Programming Environment'' by Brian W. Kernighan and Rob Pike.

The NetBSD 7.1 was released a few days ago. Can you tell us what the best change is if we compare it to the previous releases?

It has all the greatness of a major NetBSD release (NetBSD 7) but without the bugs - as it's an update to the NetBSD 7.0 release. Seriously, 7.1 comes with many new and great features, while at the same time it uses the NetBSD 7.0 codebase from the stable release branch. This does not add all the latest features (and bugs) in development, but only those that have ripened to a level of maturity, where developers took the labor to port them from the development (-current) branch to the netbsd-7 release branch. As a result, you can not only see a solid number of great new features in NetBSD 7.1 but also a very long list of security improvements and bugs that are fixed, making 7.1 the ideal base for (continued) long term support.

Please, see the NetBSD 7.1 release notes for all the details: http://www.netbsd.org/releases/formal-7/NetBSD-7.1.html (geez, I didn't manage to blog about this. Shame on me!)

What do you think is the future of NetBSD?

The goal of the NetBSD project is to provide a free, secure and portable Unix-like Open-Source operating system. I think this sums it up pretty well - NetBSD comes from early BSD, has evolved a lot over time while being true to itself, and I think we will see what the greater "Unix" universe expands to, and remain compatible. At some points, new features are adapted either because they prove to be good and mature (think threads, SMP - nothing that was there either in 1969's Unix nor in 1993's BSD!) or are required from a User perspective (like all those Linux ABI calls to run proprietary Linux software). Another part of NetBSD's feature is our great system for cross-compiling the whole system from many source platforms, and build code for many target hardware platforms. With more and more embedded and "Internet of Things" systems, this will become more and more important - and we have all this from a single source tree today. Thirdly, with NetBSD's great base as an operating system, we can add a lot of freely available software on top of it - and for this, we have our great 3rd party software system, pkgsrc.

Please tell us more about the projects you are involved with?

Currently, there are no specific projects I am involved in. I currently have less time for NetBSD than I want. During that time, I follow the project and help out in various positions - e.g. assisting in projects like Google's Summer of Code, monitor internal and external news source and communication of those that I feel are relevant to the NetBSD community. For this, I plan to continue using my blog. But I also got access to post in the name of The NetBSD Project on Facebook, so we will see what happens. :-) As for projects in the past, pkgsrc is alive and kicking, working on the next quarterly stable release. g4u is sort of sleeping right now, lacking time - yet I still feel strong for it, and I should find time to revive it, getting it close to NetBSD-current and do regular releases again. Maybe I will make this my next project, again!

Thank you

Scheduler-Series From My NetBSD Blog

(URLs old to new)

[20161105] NetBSD 7.0/xen scheduling mystery, and how to fix it with processor sets — http://www.feyrer.de/NetBSD/blog.html/nb_20161105_1754.html

[20161109] Looking at the scheduler issue again (Updated) — http://www.feyrer.de/NetBSD/blog.html/nb_20161109_0059.html

[20161113] Learning more about the NetBSD scheduler (... than I wanted to know) — http://www.feyrer.de/NetBSD/blog.html/nb_20161113_0122.html

[20161222] Bringing the scheduler saga to the finishing line — http://www.feyrer.de/NetBSD/blog.html/nb_20161222_2113.html

[20170109] Documenting NetBSD's scheduler tweaks — http://www.feyrer.de/NetBSD/blog.html/nb_20170109_2108.html