EuroBSDCon 2023

BY OLIVIER CERTNER

Last September, I had the pleasure of traveling to the city of Coimbra, Portugal, where the annual European technical conference on BSD systems, EuroBSDCon, took place.

Although I had already been to Portugal a few times, in the Douro valley near Porto, in Porto itself, or in Lisboa, I had never had the opportunity to learn about Coimbra. The city was an ancient capital of the Kingdom of Portugal, almost a thousand years ago, and, today, is a major cultural center and a vibrant city renowned for its university, one of the oldest in the world. Its region is the third most populated of Portugal, and the city has one of the highest per capita incomes in the country. Hilly, downtown Coimbra, called "Baixa", is made of magnificent ancient buildings—especially the university's historic ones. The conference premises were only a few minutes away, giving even the attendees on tight schedules a glimpse of the architecture and atmosphere of this enchanting medieval city.

The opportunity to meet in-person with a lot of developers was very exciting, and I was fortunate to be very well served in this regard. The talks and presentations I attended were great. Summaries are provided below based on my personal notes, with the hope to pique your curiosity about these topics. You can also find slides for most of the talks on the conference website at https://2023.eurobsdcon.org/slides/, and video recordings on the EuroBSDCon YouTube channel.

FreeBSD's Developer Summit

Thanks to Joseph Mingrone from the FreeBSD Foundation, as a newly contracted general developer, I was invited to attend the developer summit on Thursday and Friday. At the introductory session, approximately 30 persons presented themselves and their connection to the project. I already knew almost half of them by name and had seen a few of them on videos or conference calls but hadn't met a single one in person. Surprisingly, I don't recall having seen or met any of them at EuroBSDCon 2017 in Paris, my only other BSD conference so far. Speaking of which, let me write a belated, but big thank you to Jean-Sébastien Pédron for taking time to introduce me to various people at that earlier conference.

A significant contingent from the FreeBSD Foundation was in attendance, as well as delegations from Netflix and Beckhoff, but also several freelance developers, faculties, and other professionals not working on FreeBSD by day, but managing to be quite active in the community, such as Guido van Rooij, one of the old-timers and a member of the EuroBSDCon Foundation, whom I would like to thank for his warm welcome.

The first session was a report from the FreeBSD Foundation and its various on-going projects, such as improving the installer (Pierre Pronchéry), Wi-Fi support (Bjoern Zeeb, En-Wei Wu), RISC V support (Mitchell Horne), bhyve (John Baldwin, Mark Johnston), fixing pressing security issues (John Baldwin), re-implementing libc's string functions with SIMD (Robert Clausecker), implementing SU+J snapshots (Kirk McKusick), improving CI and fixing ports (Moin Rahman), making FreeBSD a Cloud Init platform (Mina Galić), importing OpenSSL 3 in base (Pierre Pronchéry), fixing `rtprio(2)` and rationalizing scheduling priorities, both for applications and internally (by yours truly; still a work in progress as of this writing). I honestly couldn't keep track of all projects, so let me apologize for anything I missed and advise the interested reader to browse the Foundation's blog (at https://freebsdfoundation.org/blog) and the developer summit's wiki page (https://wiki.freebsd.org/DevSummit/202309) for more information.

> "What would you do if you weren't afraid?"
>
> —Justin Gibbs

This was followed by a keynote talk by Justin Gibbs on imagining the next 30 years (since, as you must know, FreeBSD turned 30 last year). His main message, through recollections of his own involvement with the project, both technically and in establishing the Foundation, was to foster a "trying in order to be successful" attitude and to ask everyone: "What would you do if you weren't afraid?" This prompted an exchange among participants, with a lot of ideas thrown on the table, including:

- Become the OS for smartphones, IoT, any mobile computing.
- Be the best operating system to run AI workloads.
- Become a large group that can help each other grow (mentoring, teaching).
- Have people under 20 with limited budget install FreeBSD on inexpensive hardware.
- A point-and-click jail deployment system to use instead of Docker.
- Spread the culture of upstreaming, convince involved businesses that it is in their best

interest, have them allow their experienced developers to engage more with the community.

- CI, automatic regression testing. In general, many more tests.

This synthetic list is my own, so I apologize if I've left out or misunderstood some points.

Sergio Carlavilla Delgado then presented the website for a new project he is working on and encouraged people to give feedback. A more mature front page was distributed around a month after the event through mailing lists, and it was quite exciting to see the progress made and the gap between it and our existing website.

Greg Wallace, the Foundation's Director of Partnerships & Research, organized a SWOT (Strengths, Weaknesses, Opportunities, Threats) Session, with again lots of ideas exchanged. Listing all of them here would be too long, and Greg has produced a document with all the output. It can be read at https://wiki.freebsd.org/DevSummit/202309 (scroll to find the "community SWOT" line, referencing a PDF attachment).

"AI is now 'the' market, just as networking and mobile were before."
— Jordan Hubbard

We were then greeted with a surprise talk by Jordan Hubbard who, if I recall correctly, was "by chance" spending a few days near Coimba (are we supposed to really believe that?). I remember it as a great depiction of project history, but also as a focused talk on what currently matters from his perspective (he's working with NVIDIA), and what he thinks FreeBSD would be wise to embrace. I only have a few notes that probably won't do it justice, but here's the gist. AI is now "the" market, just as networking and mobile were before. Anything of significance in AI cannot run on a single GPU, nor on a single node for that matter. Consequently, speed of communication is king. Direct communication between CPUs and GPUs, or CPUs and DPUs, must be supported by the kernel. UCX (see openucx.org) is the successor to Infiniband verbs. Additionally, inference in AI will happen at the edges.

I began the first evening by chatting with Ruslan Bukin, first about his initial RISC-V port for FreeBSD, then shifting to various non-technical subjects such as special, mostly low-fat, diets, or French engineering's high quality when it comes to bicycle parts. In the process, I found myself invited to a Netflix dinner with Jonathan Looney, Warner Losh, Gleb Smirnoff, John Baldwin (and Ruslan of course), where I spent most of the time listening to the conversations while enjoying delicious seafood. I don't know how Ruslan managed it, but thanks a lot to him for dragging me in.

The next day started with a talk by Bojan Novkovic on an experimental kernel benchmarking framework called "kbench". Its goal is to facilitate kernel testing in various ways by standardizing the flow for several use cases, helping check reproducibility and catch performance regressions. It could also be a good vehicle for research projects. Its Python scripts fetch, build, and run benchmarks from a particular, pre-assembled set. Bojan's short-term goals are to expand the benchmark pool and add tracking for more metrics (`dtrace(1)`, `libxo(3)`-based utilities, `pmc(3)`). A longer-term step would be to ease collected data's post-processing. The framework can be found at https://github.com/bnovkov/kbench.

Loosely connected to the previous talk was Ruslan Bukin's presentation of his Hardware Trace Framework (**hwt(9)**). It relies on dedicated hardware support built into CPUs, namely Processor Trace (PT) for Intel, and the CoreSight and Statistical Profiling Extensions (SPE) technologies from ARM. The framework's kernel side weights 3,5k lines, with scheduler and **mmap()** hooks, **ioctl**-based trace context management, code to support the **/dev/hwt** devices and multiple backends. **hwt(1)** can be used to choose the mode of operation, configure address range filtering and perform process management and symbol lookup. CoreSight is the oldest technology from ARM and it offers only a single stream per CPU. The backend for it can be found in-tree under **sys/arm64/coresight** and is fully functional. Some code snippets to support Intel PT are available. ARM Ltd. is currently working on the SPE backend.

Changing subject to ports and packages, Michael Reim presented "Ravenports", the latest take on a new ports system started by John Marino in 2017. Michael is using it as he is working for a small hosting company in Germany that started on FreeBSD but is mostly Linux today. Its core features are a high concurrency build system with a high level of automation as necessary for a small maintainer team, support for sub packages and variants (corresponding somewhat to FreeBSD's flavor it seems), multi-platform support (all BSDs, Linux, Solaris is currently on hold, macOS has been dropped), self-contained (including toolchain, GCC-based, currently 13.2), with binary bootstrap, and with very up-to-date software versions. Some drawbacks compared to FreeBSD's ports are that it isn't currently portable to niche or obsolete ISAs (which I suspect may have to do with the non-availability of a GNAT-based Ada compiler), is only lightly tested, and has a much lower port count.

In the evening, when I contacted Joe about his plans, he responded by inviting me to a follow-up dinner to a core team and Foundation meeting, where I could chat with Deb Goodkin, Greg Wallace, Li Wen-Hsu and Ed Maste from the Foundation, and core team's Mateusz Piotrowski, all of whom were very friendly and welcoming.

The Conference

Henning Brauer, EuroBSDCon Foudation's CTO (read: Chief Trolling Officer), opened the conference with humor, kindness, and practical information on how to get beverages and how to locate the different rooms and zones of interest. He introduced a keynote by Paula Alexandra Silva on "Facilitating Change: Embracing Gender Diversity in Computer Science".

As the first technical talk, I attended John Baldwin's NVMe over Fabrics (NVMe-oF) in the FreeBSD project. The aim of this technology is to permit the use of the NVMe interface (greater parallelism, lower latency) with devices that are not physically connected to the requesting computer, but are rather accessed over TCP, RDMA or Fibre Channel. For a reason I don't recall, I unfortunately missed the outline. Trying to make sense of my notes, I built the following short summary, to be taken with a grain of salt perhaps. FreeBSD's implementation is a 3-layer design, in the middle is the transport abstraction that handles capsules (data buffers). Thanks to that, you can just allocate a queue pair and never deal with transport specifics. The user space library, **libnvmf**, is designed for simplicity and debuggability (not thread-safe, blocking I/O on sockets). It provides a TCP implementation. There are also userspace implementations of part of the functionality of both a host (**nvmfdd**, doing I/O on a single namespace on a remote controller) and a controller (multiple namespaces supported, backed either by files, character devices or memory buffers). The kernel datapath mirrors this transport abstraction but uses asynchronous callbacks instead of blocking for

performance. **nvmf(4)** is the in-kernel Fabrics host. It creates the **/dev/nvmeX** devices (so **nvmecontrol(8)** works), and supports disk access through CAM (**/dev/ndaX**). Future work is to add an in-kernel Controller, implement support for RDMA and Fibre Channel, and TLS protection for TCP queue pairs. Current code can be seen at https://github.com/bsdjhb/freebsd.git, branch **nvmf2**. This work was sponsored by Chelsio.

I then went to Kristof Provost's talk on **if_ovpn(4)** and was pleased to discover that, even if Wireguard seems to be all the rage now, some people are working on improving OpenVPN's performance. **if_ovpn(4)** is a clean-room, in-kernel, OpenVPN client implementation which supports only a subset of functionalities: Only the AES-GCM and Cha-Cha ciphers are available (the others are old), it doesn't support layer 2 networking (to keep the kernel interface simple) and is UDP-only (much more work is required for TCP). User-space handles the control channel, while the kernel handles the data channel. They both share a single socket, whose file descriptor is passed to the kernel during connection set-up. Kernel passes up to userland unknown (i.e., control) packets. Userland drives the kernel through **ioctl(2)** and **nvlist(9)** for extensibility (where Linux uses netlink, which at time of this work had yet to be integrated into FreeBSD). Key rotation is a two-phase process: New keys are declared and later switched to, without traffic disruption. vnet helped a lot for testing (see **/usr/tests/sys/net/if_ovpn**). Performance of the Data Channel Offload (DCO) by **if_ovpn(4)** was tested on a Netgate 4100. DCO with QAT (Intel's QuickAssist Technology) offload reaches 1Gbit/s, DCO with AES-NI crypto ~750Mbit/s and DCO with software crypto ~210Mbit/s, to be compared with ~210Mbit/s with OpenVPN on **if_tun** but using AES-NI for crypto. Development was sponsored by Netgate.

I was pleased to discover that, even if Wireguard seems to be all the rage now, some people are working on improving OpenVPN's performance.

At the lunch break, we gathered outside, in a kind of loggia because of the weather, for a family photo, a great pretext for a lot of chatting and an unexpected attraction: A unique T-shirt parodying the sleeve of a famous album by AC/DC, with the text "UNIX, Highway to Shell", worn by EuroBSDCon Foundation's Katie McMillan. Chatting and technical discussions then continued at the university's restaurant.

The afternoon session began for me with Hiroki Sato's talk on USB DbC (Debug Capability). Serial consoles make debugging firmware or early boot possible, but there are no serial ports on modern hardware, only legacy interfaces such as BMC (Baseboard Management Controller) on server machines. USB replaces all these legacy interfaces. However, it doesn't allow a direct connection between two hosts, a tiered star topology is required. Thus, USB debugging works by changing one port of the host to debug (the "target host") into a USB device managed by the debugging host. Comparable debugging technologies include IEEE 1394 (FireWire; legacy), which supports point-to-point and physical access to memory (see **dcons(4)**), but also USB 2.0 (see EHCI specification) although it requires a special repeater hardware. On the debugging host, a normal USB 3 stack is enough. On the target host, the new **udbc(4)** experimental driver manages the port turned into a device for simple serial

communication. It doesn't need a full USB stack, as it only has to manage the TRB (Transfer Request Block) ring buffers of two USB pipes, and DbC is designed as a simple transport for more sophisticated debug protocols (such as JTAG and Intel DCI). Physically, an A-to-A USB 3.0 cross-cable is required, and, of all ports of the root hub, only one will convert to behave as a USB device to the debugging host. Code and an install image can be found at https://people.allbsd.org/~hrs/FreeBSD/udbc/20230915. After enough compatibility feedback, patches will be submitted. Nothing in this work is x86-specific, and porting to other BSDs should be easy. **`udbc(4)`** could be extended to mimic other types of USB devices (such as, possibly, a mass storage device).

Next, I attended Warner Losh's talk on booting FreeBSD with LinuxBoot, using **`loader.kboot`**. I'm not going into many details here since the presentation was dense and you can find them all in the public slides. LinuxBoot was started in 2017 by Google (as NERF) to provide a unified booting environment and to get rid of UEFI (mostly). With it, a low-level boot loader initializes a machine just enough to launch Linux, which then runs scripts to determine what to finally boot via **`kexec()`**. There is a very nice community around it. The low-level boot loader can be UEFI (Pre-EFI Interface), coreboot romstage, U-boot SPL, or the Slim bootloader. This talk only considers UEFI and its EDK2 implementation, which LinuxBoot strips down to only PEI (Pre-EFI Interface; initializes low-level details of the machine, such as memory, clocks, etc.) and the Runtime Services. By using LinuxBoot, you only have to write device drivers once—for Linux and not UEFI DXE—which brings several advantages: Faster time-to-market, often faster boot times, more security hardened (100k contributors versus only 100 to 200 for Grub/EDK2). FreeBSD needed to be ported to LinuxBoot for several reasons. On amd64 and aarch64, the kernel expects some metadata such as memory maps to boot, and some are provided by the system firmware (BIOS, UEFI) which the kernel can't necessarily access. These data are usually prepared by our **`loader(8)`**, so the first step was to port it as a Linux binary and add "stand" devices to access host resources from it. Also, Linux already sets the virtual address mapping, via UEFI's **`SetVirtualAddressMapping()`**, which can be called only once. This required changing FreeBSD's kernel to support a non-trivial virtual to physical address mapping (and there was also a problem with the GICv3 interrupt controller on aarch64). With this work, FreeBSD is the only non-Linux, non-GRUB-assisted fully booting UEFI OS under LinuxBoot on x86 and aarch64.

I finished the talks of the first day with Toshaan Bharvani's talk on Running FreeBSD on OpenPOWER, which is an exciting architecture that permits machines booting with completely open firmware. Porting efforts on POWER are shifting towards little endianness because a lot of desktop applications (such as Firefox, JS) can't run properly on a big-endian architecture. The open firmware includes OPAL (Open Power Abstraction Loader) implemented by skiboot, which launches Linux running the petitboot bootloader, but also some open BMC implementations such as LibreBMC or OpenBMC. Current developers are Alfredo Dal'Ava Junior, Leandro Lupori, Andre Fernando da Silva, among others. There is still a lot of work to do, so please volunteer! Current hardware is still expensive, with ~5k€ for an IBM AC922 Developer Machine or a Talos II entry-level developer system. "Soft" hardware, with FPGA-based Microwatt and LibreSOC, is also available. The OpenPOWER HUB initiative (https://openpowerfoundation.org/hub) provides access to several types of OpenPOWER hardware. The next generation hardware should come much better priced, first with entry-level PowerISA 3.1 Racks (start of 2024) to be followed by Workstations (expected price around $1000 to $1500), single-board computers such as PowerPi! Embedded Single Board

Computer in several versions and generations, an enablement platform for developers (maybe around $500), microcontroller systems, and FPGA-based devices. So, please help emPOWER BSDs!

The day concluded with a beautiful social event. It took place not far from the historic center, on the other side of the Mondego River, in a restaurant with a huge room to accommodate us all. I don't remember how it happened, but I soon found myself sitting at a table between John Baldwin and Mark Johnston. When these two giants engage in technical conversations, it is always a pleasure to listen to them. We enjoyed the evening, with lots of occasions for informal chats around a great buffet. The most spectacular moment was an unforgettable demonstration of "Fado de Coimbra" by a group of men coated with the traditional back academic suit. The emotional power their rendition conveyed is simply impossible to describe in text.

The day concluded with a beautiful social event.

The second and last day of the conference started for me with Eirik Øverby speaking about FreeBSD at Modirum, how they use it (and other open-source software), and what makes the community so good for them. The diverse war stories involving jails, MySQL on ZFS, `SO_REUSEPORT_LB`, epairs and `relayd` were captivating, and Eirik made this journey very lively. His slides are available on the conference website.

I was very interested in Christos Margiolis' presentation of his work on Arbitrary Instruction Tracing with DTrace. DTrace is a powerful live debugging tool I'm still not using nearly enough, and Christos has augmented it with a new provider called kinst ("kernel instructions"). The base FDT provider (Function Boundary Tracing) can only trace entry and return points of kernel functions that have not been inlined. kinst provides probes to trace all instructions in a function, a specific instruction, or the entry or return points of inlined functions. The selected probe information is passed from `dtrace(1)` to `libdtrace` and then kinst using `/dev/dtrace/kinst`. kinst then disassembles the function and creates the requested probes by overwriting target instructions with a breakpoint instruction. On execution, the breakpoint handler calls `dtrace_invop()` which calls `kinst_invop()`. The instructions replaced by breakpoints nonetheless have to be executed for the execution to stay correct. Since emulation is tedious and error prone, these instructions are copied into a trampoline where execution is transferred to manually. The main difficulty with this approach is that RIP/PC-relative instructions still either have to be re-encoded (amd64) or emulated (arm64, riscv). Most of the hard work for inline function tracing is done by `libtrace` (using ELF and DWARF information). If a probe designates a function that was not inlined, kinst defers it to FBT to avoid code duplication. Each kernel module has to be checked for an inline function, which is currently painfully slow.

As the afternoon session was beginning, I attended the `gunion(8)` talk by Kirk McKusick. Kirk has the knack of presenting his ideas in clear and didactic fashion, as you can experience when watching videos of his presentations on YouTube. After starting with a refresher on the GEOM framework, he presented the `gunion(8)` utility, which tracks changes to a read-only disk on a writeable disk. This is similar to what `unionfs(5)` does with files and di-

rectories, but at the lower level of blocks on a block device. The upper disk (the writeable one) must be at least the size of the lower disk (the read-only one). Union metadata exists only as long as the union exists (no persistency at the moment). `gunion(8)` offers `create` and `destroy` commands, as well as `revert` (discard the changes stored in the upper disk) and `commit` (write back the upper disk's changes in the lower disk). Uses of `gunion(8)` include trying to fix disks with broken filesystems without the risk of destroying them more and without the need for backing them up first: If repairing fails, just use `revert`, else you can `commit` and use the original disk. `gunion(8)` is also useful to implement poor-man's clones. `gnop(8)` was instrumental in testing error recovery, delay and out-of-order I/O handling during the implementation of `gunion(8)`.

For the last talk of my day, I stayed in the "Auditorio", as I had all day, to watch Michael Dexter's presentation about "the FreeBSD Appliance". He started by stating a definition of a software appliance inspired by Wikipedia's: Just enough OS ("JeOS") to run an application on commodity hardware. The guidelines of the presentation were then twofold. First, it shows how FreeBSD is well suited to the task (integrated OpenZFS, jails, bhyve), and with some advantages over illumos or Linux. "JeOS" may become a more practical reality with packaged base, but it's already possible to rule out almost everything through build options and produce a working OS that boots in seconds, "OccamBSD" (see github.com/michael-dexter/occambsd). Then, custom images based on "OccamBSD" and working on bare metal or under hypervisors can be produced. Second, it showcases some of the new features introduced in FreeBSD 14.0, a way to stay up to date with recent developments.

I had initially hoped to attend Mateusz Piotrowski's ZFS Directory Scaling talk but changed plans for a meetup with FreeBSD Foundation's Ed Maste to discuss my current projects and see how to push the latest commits.

That evening, since I was leaving Monday morning, I was planning to go out and contacted Mateusz Piotrowski about his plans. I was directed to a tapas bar, where I met with David Cottlehuber, and a bunch of other nice guys with whom we finally had dinner, including Christos Margiolis, Luca Pizzamiglio, Mohamed, and Peter. I finished the evening in an Irish bar with Peter and some others up to a time I have not been able to remember. Fortunately, my departure train was not too early in the morning...

These were lively and fulfilling four days spent meeting a lot of people and hearing about great technical projects. That I'm now looking forward to the next BSD conferences probably won't surprise you. If you've never attended and have the opportunity, well, come to get a taste of it and I don't think you will regret it!

OLIVIER CERTNER stumbled onto FreeBSD in 2004 as the result of a search for a more serious alternative to Linux, after a cataclysmic system crash destroyed his main ext3 HDD. He has been using it ever since everywhere he can and has maintained small changes throughout the system privately for around 15 years. He has recently engaged with the community and open-source development, and is currently working on a revamp of scheduling priorities (which gave birth to a paper at AsiaBSDCon 2024), has started analyzing weird OOM behaviors, and plans to work on some deep VFS problems in the near future, including a redesign of unionfs. He has been sponsored by the FreeBSD Foundation for most of his public work.