Showing and Debugging Haiku with QEMU

François Revol

Laboratoire d'Informatique de Grenoble (LIG), France;
Laboratoire de Conception et d'Intégration des Systèmes (LCIS), France
`Francois.Revol@imag.fr`

Abstract. Haiku is one of those many alternative OpenSource Operating Systems under development around, mostly written on leisure time but also used as a research test bed. Like many others, it requires lots of debugging sessions, as well as a showcase platform. We will detail the various uses of QEMU for these purposes with the Haiku operating system.

1 History and Goals

Haiku is a Free Software operating system written from scratch, following the design principles of the BeOS, with binary compatibility as a target for Release 1. It focuses on desktop usage, with its own vision of the KISS principle (Keep It Smart and Simple), allowing it to stay small. Development started in 2001, and a first alpha version was released september 14th 2009. Haiku has a coherent design [13], and has been used to demonstrate several research prototypes at the University of Auckland, including a new GUI layout model [12], and automatic GUI help generation [10]. Haiku uses its own filesystem, BFS [11], to provide specific features on the desktop including very fast live queries, but also support many other filesystems.

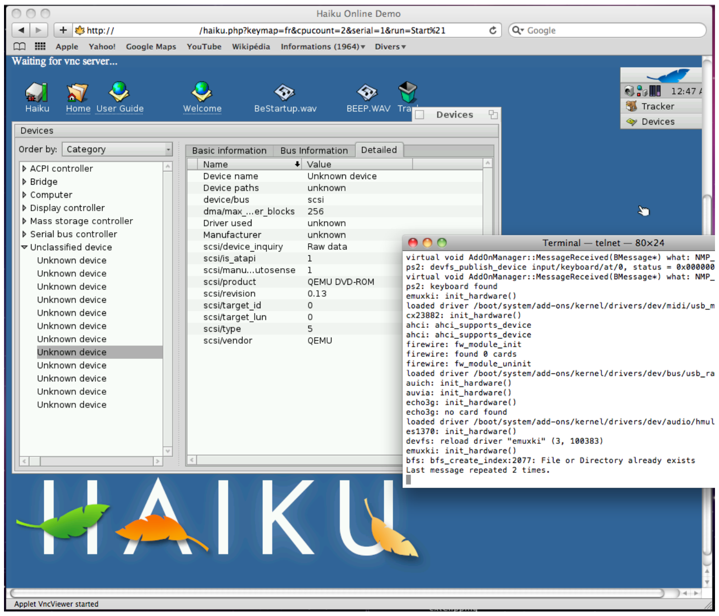
2 Showing off Haiku

2.1 Virtual Machine Images

In addition to the CD image, the Haiku website [4] proposes several disk images for the most common virtual machines (VMware, VirtualBox, and of course QEMU). QEMU's comprehensive format support allows it to use any of those.

2.2 Live on the Internet

The Free Live OS Zoo website [2] allows people to experiment with many Operating System directly in their browser with a VNC Java Applet. It is built on top of QEMU and its internal VNC server. A similar application has been built tailored to demonstrating Haiku only [3], which adds several features, like the serial port debug output available as a `telnet:` URI as shown on Fig. 1, processor count selection, and soon audio streaming. The need for an absolute pointing device for use with VNC led to fixing the Wacom tablet emulation code in QEMU.


Haiku Online Demo
http:// /haiku.php?keymap=fr&cpucount=2&serial=1&run=Start%21
Apple Yahoo! Google Maps YouTube Wikipédia Informations (1964) Divers
Waiting for vnc server...
Haiku Home User Guide Welcome BeStartup.wav BEEP.WAV
Devices
Tracker
Devices
Order by: Category
ACPI controller
Bridge
Computer
Display controller
Mass storage controller
Serial bus controller
Unclassified device
Unknown device
Basic information Bus Information Detailed
Terminal — telnet — 80×24
HAIKU
Applet VncViewer started

Name	Value
Device name	Unknown device
Device paths	unknown
device/bus	scsi
dma/max_...er_blocks	256
Driver used	unknown
Manufacturer	unknown
scsi/device_inquiry	Raw data
scsi/is_atapi	1
scsi/manu...utosense	1
scsi/product	QEMU DVD-ROM
scsi/revision	0.13
scsi/target_id	0
scsi/target_lun	0
scsi/type	5
scsi/vendor	QEMU

```
virtual void AddOnManager::MessageReceived(BMessage*) what: NMP_
ps2: devfs_publish_device input/keyboard/at/0, status = 0x000000
virtual void AddOnManager::MessageReceived(BMessage*) what: NMP_
ps2: keyboard found
emuxki: init_hardware()
loaded driver /boot/system/add-ons/kernel/drivers/dev/midi/usb_m
cx23882: init_hardware()
ahci: ahci_supports_device
ahci: ahci_supports_device
firewire: fw_module_init
firewire: found 0 cards
firewire: fw_module_uninit
loaded driver /boot/system/add-ons/kernel/drivers/dev/bus/usb_ra
auich: init_hardware()
auvia: init_hardware()
echo3g: init_hardware()
echo3g: no card found
loaded driver /boot/system/add-ons/kernel/drivers/dev/audio/hmul
es1370: init_hardware()
devfs: reload driver "emuxki" (3, 100383)
emuxki: init_hardware()
bfs: bfs_create_index:2077: File or Directory already exists
Last message repeated 2 times.
```

Fig. 1. The Haiku desktop running in QEMU shown in a Web browser by the VNC Applet, with serial debug output in a telnet client

3 Debugging the Kernel and Drivers

While using a traditional modular kernel, Haiku, like BeOS did a decade ago, already uses technologies being rediscovered on the desktop world, like preemptive and SMP kernel without giant lock, tickless timers, node monitoring... Development time greatly benefits from the flexibility provided by QEMU, starting from simple serial port redirection to inserting tests in the QEMU code itself to instrument the guest and assert its behaviour, the later being eased due to the Free Software nature of QEMU.

While Haiku runs very well on other virtual machines like VirtualBox or VMware, it strives at supporting as much real hardware as possible, and the diversity of devices emulated by QEMU allows better driver testing. While VirtualBox or VMware are more integrated with the host system, and their bias towards virtualization and guest additions might allow for better usage experience of the guest OS, QEMU is more suited to actual guest OS development from this perspective.

QEMU is also unique in the diversity of processor types and platforms it supports, allowing easy bootstrap of ports to PowerPC, ARM and other architectures. Haiku officially supports only the x86 architecture, but other ports are in various stages of completion, as shown in Table 1.

Haiku's Kernel Debugging Land provides many tools including a `gdb` stub, however the QEMU-provided stub has proven useful to debug the bootloader for

Table 1. Haiku ports and emulators used for development

Architecture	Platforms	Status	Emulators
x86	PC	100%	QEMU, VirtualBox, VMware
PowerPC	PowerMac, Pegasos...	25%	QEMU, PearPC [6]
ARM	Gumstix, BeagleBoard...	10%	QEMU
m68k	Atari Falcon, Amiga	10%	ARAnyM [1], QEMU-m68k [9]
MIPSel	Yeeloong[5]	2%	QEMU
X86_64	PC	2%	QEMU

the ARM port, with `gdb` reading the ELF version allowing to step through the C/C++ code. The Haiku build system setup doesn't build `gdb` along with the cross compiler, but it is a simple step to perform.

Interesting challenges facing Haiku include support for more time sources than just RDTSC, which causes problems in virtualized environments. Partial HPET support exists for timers which will require testing, first on QEMU before real hardware.

4 Debugging QEMU

OS diversity also means more hardware usage patterns, sometimes uncovering bugs in hardware chipsets, like the ATI IXP bridge not sending IRQs for their 8253/54 PIT (and APIC as well it seems) timer chip emulation when using mode 0, which BeOS and Haiku use to implement tickless timers.

Likewise, the diversity of guests systems which Haiku participates in allows for better emulation. For example a bug in the HD audio device caused an interrupt storm in the Haiku driver which wasn't triggered by other OSes drivers, because the Haiku driver handles codec status changes outside of the IRQ handler, unlike drivers in other OSes [8].

Also, when developing the `haiku.php` script, an absolute pointing device was required to keep both cursors in sync in the VNC applet, however the Wacom tablet emulation code had several issues, like wrong coordinate scaling and buttons reporting, as well as boggus asynchronous mode leading to high guest cpu usage. This simple fix [7] took several years to get through though.

5 Conclusion

Haiku provides various usage patterns to QEMU, allowing both to get fixes and thus better reliability. The availability of QEMU in most GNU/Linux distributions allows people to easily test and debug Haiku, attracting new developers. The many supported system emulations and the integrated GDB stub speed up bootstrapping ports to other architectures.

References

1. Atari running on any machine. `http://aranym.org/`.
2. The Free Live OS Zoo. `http://www.oszoo.org/`.
3. Haiku online demo php script. `http://dev.haiku-os.org/browser/haiku/trunk/3rdparty/mmu_man/onlinedemo/haiku.php`.
4. Haiku project. `http://www.haiku-os.org/`.
5. Lemote yeeloong notebook. `http://www.lemote.com/en/products/Notebook/2010/0310/112.html`.
6. Pearpc emulator. `http://pearpc.sourceforge.net/`.
7. QEMU fix: Fixed wacom emulation. `http://git.qemu.org/qemu.git/commit/?id=2ca2078e287174522e3a6229618947d3d285b8c0`.
8. QEMU fix: intel-hda: Honor WAKEEN bits. `http://git.qemu.org/qemu.git/commit/?id=af93485cde810f3c2f488533e0b60c99eae5d01d`.
9. Qemu m68k branch. `http://gitorious.org/qemu-m68k/qemu-m68k`.
10. Y. A. Chakravarthi, C. Lutteroth, and G. Weber. Aimhelp: generating help for gui applications automatically. In *CHINZ '09: Proceedings of the 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*, pages 21–28, New York, NY, USA, 2009. ACM.
11. D. Giampaolo. *Practical file system design with the BE file system.* Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 1999.
12. J. Kim and C. Lutteroth. Multi-platform document-oriented guis. In G. Weber and P. Calder, editors, *Tenth Australasian User Interface Conference (AUIC 2009)*, volume 93 of *CRPIT*, pages 31–38, Wellington, New Zealand, 2009. ACS.
13. F. Revol. The Haiku Operating System. `http://eurosys2010.sigops-france.fr/poster_demo/eurosys2010-final17.pdf`.