Plan 9 on modern, large AMD64 systems

Geoff Collyer

History

The late Jim McKie created the *9k* 64-bit-capable kernel from the 32-bit Plan 9 kernel. I took the March 2014 *9k* from Bell Labs, adapted it to work with larger memories (the original was limited to 600MB), and ported and updated drivers from Plan 9. The necessary pieces to make a terminal kernel are still absent, notably VGA support, which requires x86 Real mode and constant maintenance, other than VESA graphics. I have deliberately omitted support for obsolete or really annoying hardware.

Current Status

To date, there are two architectures supported, `k10` and `rv` (for 64-bit RISC-V). The `k10` port runs on AMD K10 and later AMD64 CPUs, and compatible Intel64 CPUs. It has been successfully run as a CPU or file server on

- ASUS P10S WS motherboards
- Supermicro X11SCL-F motherboards
- PC Engines APU2C4 and APU2E4 systems
- Intel NUC5i7s

This port is 64-bit capable. The kernel seems to now be 64-bit clean, and *fossil*, *venti*, *malloc*, and *memset* have been adapted to 64-bit memories, when appropriate. The rest of the C library has not yet been converted (mostly to use *uintptr* instead of *ulong*), but this is enough to enable use of more than 4GB of memory.

The memory management code can exploit up to 48 address bits (the AMD64 default) currently. It makes no use of 5-level page tables. One address bit is consumed by kernel memory mapping, so this permits the use of 128TB of memory, which should be enough for a little while. In principle, this port can use arbitrary amounts of memory, BIOS permitting. It needs 2GB of contiguous non-reserved memory at 0. Each 32GB of user memory is described by 1GB of kernel data structures (worst case, assuming only 4K pages), but those data structures can reside above 4 GB. Memories of 4, 16, 32 and 64 GB have all been exercised.

MONITOR and MWAIT instructions are now used where possible when idling or waiting for a lock to be released. This conserves power, reduces heat produced, and should reduce instructions executed on busy many-core systems. Hardware has these instructions, though they can be disabled in some BIOSes (don’t do it!), but VMs don’t always implement them, even on machines that do.

Supported Peripherals

MSI interrupts will be used when available, and MSI-X interrupts will not (there seems to be little or no benefit to MSI-X). When BIOS tables contain obviously bogus IRQ numbers (e.g., 0), *9k* will assign unique vectors instead of using the bogus IRQs, with or without MSI.

These are the supported peripherals:

- optional serial console on COM1
- serial ports on PCI(E) card
- Intel's 1 and 10 GB/s Ethernet controllers
- SATA and NVME disks and SSDs are standard; ATA and AOE disks can be easily made to work if needed.
- EHCI USB can be configured in; I don't use it, in large part because of the truly excessive interrupt load it often creates. The presence of xHCI controller(s) seems to interfere with EHCI.

Surprises

Since about 2015, at least some of Intel's [MIP]CH/chipset AHCI controllers have an incompatible layout of a few PCI configuration space registers needed to enable the controller and its ports. Intel needed to change something to accomodate more than 8 SATA ports, and they had painted themselves into a corner. Instead of maintaining compatibility (Intel's main virtue), they chose to silently break compatibility. The new layout can be detected if the controller supports more than 8 ports, but not all of the new (Series 100 and later) controllers do.

Limitations

It's useful to have a `/386/9pccpu` system around for running *gs*, *bridge*(3), and *aux/pm*.

Rebooting by direct loading of `386 9` and `amd64 9k` kernels now works, but Intel has a lot to answer for.

Page Table Hierarchy

size	use and level
4K	small page, pte level
2M	large page, pd level
1G	pdp level
512G	pml4 level
256T	pml4 total

Kernel Virtual Memory Map

From top of memory down:

address	name	size	use
-2M	PMAPADDR	2M?	?
-8M	PDMAP	6M?	?
-512M	VMAP	256M	
-2G	KSEG0	2G	kernel low memory to -512M above (VMAP), includes PDMAP in last kernel GB, may include Page structs.
˜-513G	KSEG1PML4		
-1T	KSEG1	512G	only embedded PML4 & page tables
-128T	KSEG2	-	all non-KSEG0 physical memory above. Page structs often go here.
-256T	PML4BASE	-	base of PML4, holds entire addr space
0	–	128T	user process