OpenBSD/sun4v: Porting OpenBSD to Sun’s UltraSPARC T1 and T2 processors

Mark Kettenis
kettenis@openbsd.org

OpenBSD

October 9, 2011
Outline

1. Introduction
2. History of 64-bit SPARC
3. Step 1: OpenBSD on “bare metal”
4. Step 2: OpenBSD in a guest domain
5. Step 3: OpenBSD in the primary domain
6. Future improvements
Introduction

Just like BSD, SPARC is very undead!
- Oracle just unveiled the a new SPARC CPU
- World’s Nr. 1 supercomputer is SPARC
Short History of SPARC

- **32-bit SPARC V7/V8**
  - Lots of implementations

- **64-bit SPARC V9**
  - Fully specified unprivileged mode
  - Partly specifies privileged mode

First implemented by HAL/Fujitsu: SPARC64
Sun followed with UltraSPARC (sun4u)
  - Privileged mode differs between SPARC64 and UltraSPARC (MMU)

Attempt to make SPARC64 and UltraSPARC more compatible: SPARC Joint Programming Specification (JPS1).
  - Fujitsu SPARC64-V
  - Sun UltraSPARC III

Sun UltraSPARC IV: First SPARC multicore

Sun UltraSPARC T1: Chip Multithreading, Hypervisor (sun4v)
OpenBSD on SPARC V9

- OpenBSD/sparc64 runs on
  - Sun UltraSPARC I, II, III, IV, T1 and T2
  - Fujitsu SPARC64-V, SPARC64-VI and SPARC64-VII

Almost all machines are supported (including laptops and E10000)

- Based on the NetBSD port by Eduardo Horvath
- Porting to OpenBSD started in 2001; mostly done by Jason Wright with help from Arthur Grabowski.
- Officially supported since OpenBSD 3.0
- OpenBSD 4.0 was the first release to run on UltraSPARC III
- OpenBSD 4.4 added support for Fujitsu SPARC64

OpenBSD is the only fully Open Source OS supporting Fujitsu SPARC64!
SPARC V9 “features”

- RISC
- Register windows
- Software TLB
- Fast traps
- Prioritized interrupts
Chip Multithreading

- Hyperthreading on steroids
  - 4 or 8 threads per core instead of just 2
- Modern CPUs spend a lot of time waiting for memory access
- Switch to another thread and continue to do useful work
- Multicore
  - Up to 64 virtual CPUs per chip.
  - Up to 4 chips per machine.
  - Up to 256 virtual CPUs per system (T5440).
Outline

1. Introduction
2. History of 64-bit SPARC
3. Step 1: OpenBSD on “bare metal”
4. Step 2: OpenBSD in a guest domain
5. Step 3: OpenBSD in the primary domain
6. Future improvements
Step 1: OpenBSD on “bare metal”

Initial Hypervisor release had no domaining capabilities
CPU support

- Unprivileged instruction set 100% compatible with SPARC V9
- Mostly compatible with older UltraSPARC processors
- MMU Translation Table Entries have different format
- `sun4u`: different sets (AG, IG, MG) of globals selected by trap type can be switched by modifying `%pstate`
- `sun4v`: different sets of globals selected by trap level; can be switched by modifying `%gl`
Bootloader

- 1st stage bootloader written in Forth; no changes necessary
- 2nd stage bootloader written in C; calls OpenBOOT for all hardware access no changes necessary either
- Kernel needs lots of changes

Goal

Single kernel for sun4u and sun4v
Code patching

#define NORMAL_GLOBALS() \ 
999: wrpr %g0, PSTATE_KERN, %pstate ;\ 
    .section .sun4v_patch, "ax" ;\ 
    .word 999b ;\ 
    wrpr %g0, 0, %gl ;\ 
    .previous

struct sun4v_patch {
    u_int32_t addr;
    u_int32_t insn;
}

Also used to patch away cache flushes;
UltraSPARC T1/2 no longer has virtual cache aliasing
Traps

- SPARC V9 trap handling can be deep:
  - Register windows
  - Software TLB
- sun4u: 4 levels of nested trap levels
- sun4v: 4 levels, but 2 reserved for Hyperprivileged mode
  - Hypervisor helps by doing some of the TLB handling
  - Still some trickery needed: invert order in which traps are handled
- Separate trap handlers for sun4u and sun4v
System support

CPU support is not enough
Also need to be able to talk to the system hardware to do I/O.

Device drivers:

- `vbus(4)` virtual device bus
- `vpci(4)` virtual PCIe host bridge
- `vrng(4)` virtual random number generator
- `vrtc(4)` virtual real time clock
PCI host bridge

Several generations of PCI host bridges on sun4u:

- Psycho  UltraSPARC I/II/IIi; psycho(4)
- Schizo  UltraSPARC III/IIIi/IV; schizo(4)
- Fire    UltraSPARC IIIi, PCIe; pyro(4)

Host bridge handles:

- PCI config space access
- PCI interrupt management
- IOMMU management

sun4v Hypervisor provides these services; vpci(4) makes Hypervisor calls instead of direct hardware access
Step 2: OpenBSD in a guest domain

Later Hypervisor added domaining capabilities
OpenBSD in a guest domain

Firmware upgrade for T1000/T2000 adds domaining capable Hypervisor. Allows creation of multiple domains. Domains get assigned resources for exclusive use:

- Virtual CPUs
- Memory
- Cryptographics resources
- IO devices

Control domain  Can configure the Hypervisor; has access to service processor

Service domain  Domain that provides virtual devices to other domains

IO domain  A domain with direct access to physical devices

Guest domain  A domain that uses virtual devices provided by a service domain
Device drivers implemented in this phase:

- `cbus(4)` channel device bus
- `vnet(4)` virtual network interface
- `vdsk(4)` virtual disk
Bootstrapping OpenBSD in a Guest domain

Bootstrapping was done on T1000 server:

- Create control domain and IO domain using Solaris
- Boot diskless kernel (NFS root) using network interface in PCIe slot
- Hack on vnet(4) code; recompile kernel; repeat until it works
- Boot diskless kernel (NFS root) using vnet(4) interface
- Hack on vdisk(4) code; recompile kernel; repeat until it works
- Boot kernel from vdisk(4)
Communication between domains

Logical domain channels (LDC)

- Hypervisor support:
  - send/receive 64-byte message (unreliable)
  - copy memory between domains
  - map another domain’s pages

- Standard protocols defined by Sun:
  - LDC Transport Layer:
    Reliable data streams using 64-byte messages
  - Virtual IO (VIO) protocols:
    - vDisk
    - vNet
    Built on top of the LDC Transport Layer
Virtual Network Interfaces

Implements vNet virtual IO protocol

- Memory containing Tx packets needs to be exposed to other domain.
- Can’t trust the other domain; don’t expose mbufs to it!
- Dedicated memory pool for each interface; copy mbufs into pool before Tx

cbus0 at vbus0
vnet0 at cbus0 chan 0x0: ivec 0x200, 0x201, address 00:14:4f:f8:38:e7
Virtual Disks

Implements client side of vDisk virtual IO protocol

- vdisk(4) emulates SCSI
  SCSI commands are converted into vDisk commands
- Expose buffers to other domain
  Domain providing storage has to be trusted anyway

cbus0 at vbus0
vdsk0 at cbus0 chan 0x2: ivec 0x204, 0x205
scsibus0 at vdisk0: 2 targets
sd0 at scsibus0 targ 0 lun 0: <SUN, Virtual Disk, 1.1> SCSI3 0/direct fixed
  sd0: 2048MB, 512 bytes/sec, 4194304 sec total
vdsk1 at cbus0 chan 0x3: ivec 0x206, 0x207
scsibus1 at vdisk1: 2 targets
sd1 at scsibus1 targ 0 lun 0: <SUN, Virtual Disk, 1.1> SCSI3 0/direct fixed
  sd1: 2048MB, 512 bytes/sec, 4194304 sec total

Mark Kettenis (OpenBSD)
OpenBSD/sun4v
October 9, 2011
22 / 37
Use case: pf firewall in the box

For example on a T1000 server:

- Put a decent NIC into the PCI slot
- Configure vNet between domains
Step 3: OpenBSD in the primary domain

Device drivers:

`vcc(4)` virtual console concentrator
`vcctty(4)` virtual console device
`vsw(4)` virtual switch
`vds(4)` virtual disk server
`vdsp(4)` virtual disk server port
Guest domain console access

vcc0 at cbus0
vcctty0 at vcc0 chan 0x19: ivec 0x232, 0x233 domain "svendsen"
vcc tty1 at vcc0 chan 0x1e: ivec 0x23c, 0x23d domain "alfven"
vcc tty2 at vcc0 chan 0x11: ivec 0x222, 0x223 domain "stenhammar"

# cu -l ttyV0
Connected

{0} ok
Virtual switch

Solaris has a virtual switch device driver

OpenBSD Philosophy
Avoid duplicating code!

bridge(4) already implements a layer 2 switch
Reuse by:
- Create a vnet(4) interface for each switch port
- Bridge them together using bridge(4)
Network configuration

vsw0 at cbus0
vnet0 at vsw0 chan 0x12: ivec 0x224, 0x225, address 00:00:00:00:00:00
vnet1 at vsw0 chan 0x1a: ivec 0x234, 0x235, address 00:00:00:00:00:00
vnet2 at vsw0 chan 0xb: ivec 0x216, 0x217, address 00:00:00:00:00:00

# ifconfig vnet0 -inet6 up
# ifconfig vnet1 -inet6 up
# ifconfig vnet2 -inet6 up
# ifconfig em1 up
# ifconfig bridge0 add vnet0 add vnet1 add vnet2 add em1 up
Virtual Disk Server

Implements server side of vDisk virtual IO protocol

- Exports disk images as virtual disks to other domains
- Much like vnd(4)
- All memory is exported by the client to the server
  No security issues!
Solaris as an OpenBSD guest

# cu -l ttyV2
Connected

{0} ok boot
Boot device: disk File and args:
SunOS Release 5.11 Version snv_151a 64-bit
Copyright (c) 1983, 2010, Oracle and/or its affiliates. All rights reserved.
Hostname: stenhammar

stenhammar console login: kettenis
Password:
Last login: Sat Jan 8 23:42:41 from nielsen.sibeliu
Oracle Corporation SunOS 5.11 snv_151a November 2010
kettenis@stenhammar:~$
Linux as an OpenBSD guest

- Only mainstream SPARC distro: Debian
- Doesn’t seem to support sun4v by default
- Installer boots, but no virtual hardware seems to be detected
- Poor support for installation over serial console
Outline

1. Introduction
2. History of 64-bit SPARC
3. Step 1: OpenBSD on “bare metal”
4. Step 2: OpenBSD in a guest domain
5. Step 3: OpenBSD in the primary domain
6. Future improvements
Domain Configuration

Currently only possible using Solaris:
- Reconfigure domains
- Start domains
- Stop domains

Needs to be possible from OpenBSD

Status:
- Start/Stop works; needs some cleanup.
- Reconfigure under investigation; lots of code still to be written

Meanwhile: Keep a Solaris disk around!
Hypervisor specification defines protocols to assist manageability:

- `domain-shutdown` Request graceful shutdown
- `domain-panic` Request panic
- `dr-cpu` Dynamic reconfiguration for virtual CPUs

OpenBSD needs to implement these protocols... ...but currently doesn't.
Support for Neptune

Neptune is Sun’s 10GigE network interface

- On-chip on UltraSPARC T2 (and SPARC T3?)
  - but 10GigE only (need XAUI card + XFP)
  - virtualizable
- Companion chip for UltraSPARC T2+
  - GigE or 10GigE (with XAUI card + XFP)

dlg@ needs to unslack!
Or if somebody could donate a XAUI card + XFP...
OpenBSD on Oracle SPARC T3?

SPARC T3 not radically different from UltraSPARC T2

OpenBSD should run, especially in a guest domain...
...but nobody tried this yet.

No chip-specific hardware documentation available
Hypervisor draft available
SPARC T4 has a new core
Better single-thread performance
OpenBSD might run, especially in a guest domain
No chip-specific hardware documentation available
Acknowledgements

Sun Microsystems for providing the hardware that made this work possible