開
放
的
열린
مفتوح
libre
मुक्त
ముక్త
livre
libero
ಮುಕ್ತ
开放的
açık
open
nyílt
פתוח
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை

OpenSolaris[TM] xVM

David Edmondson + friends
dme@sun.com + xen-discuss@opensolaris.org
Solaris Engineering

Introduction

- What is the OpenSolaris xVM?
- Why should I care?
- Using xVM
  - Control domain: booting, services, tools
  - Guest domains: creation, booting
  - Debugging
- Porting OpenSolaris
- Where are the pieces?
- Futures

What is xVM?

- An open source hypervisor
- A port of OpenSolaris to run on the hypervisor
- A set of control tools for the hypervisor
- A set of support tools for running other operating systems on the hypervisor under the direction of OpenSolaris

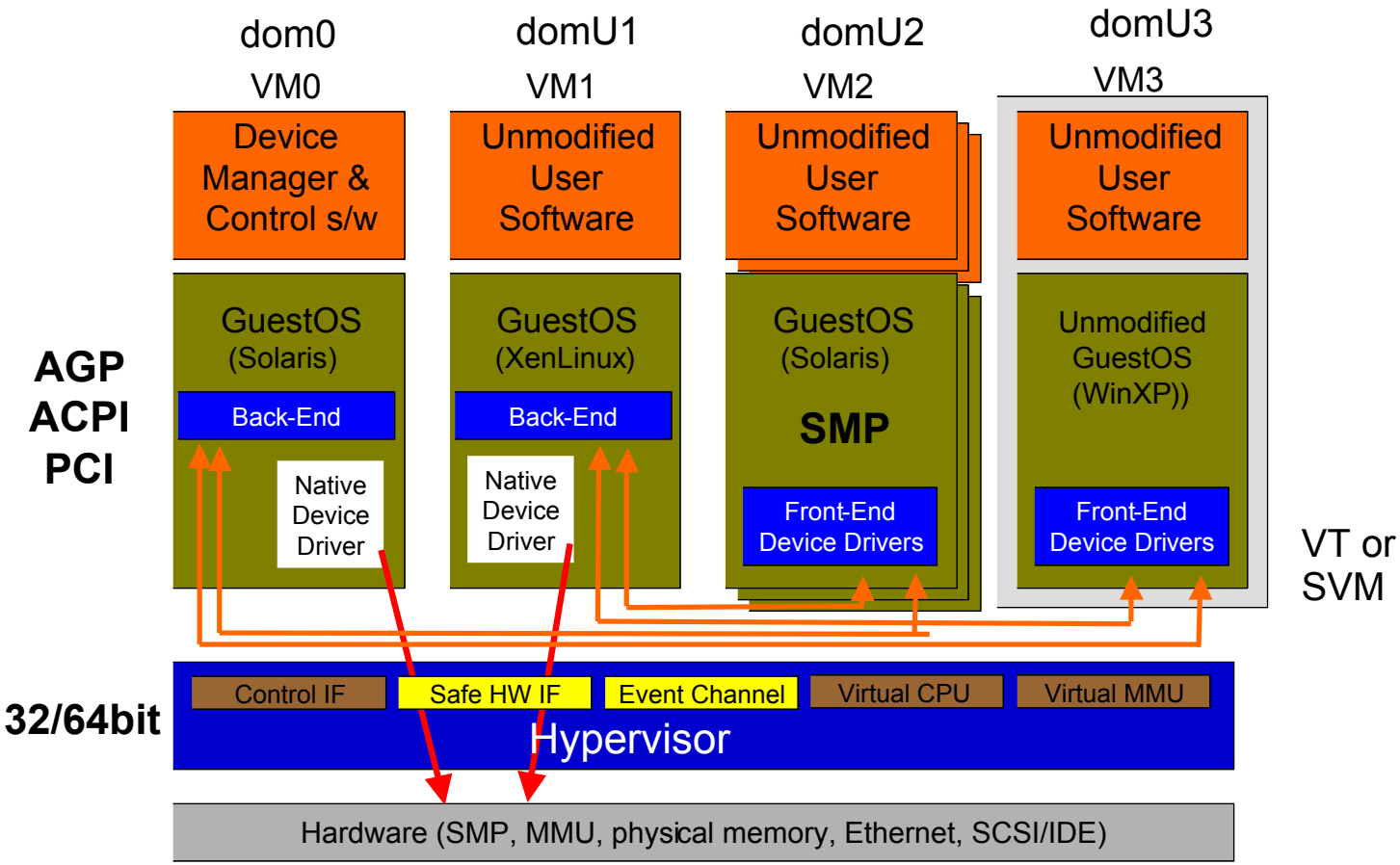
Open source hypervisor technology

- Originally developed at the University of Cambridge, England
  - Licensed under the GPLv2 and LGPL
  - XenSource: a start-up created by the original developers of the project to commercialise the results
- Significant contributions from Intel, AMD, IBM, HP, Fujitsu, ...
- Mostly x86, but also available on PPC and Itanium
- Now at version 3.1
  - OpenSolaris port is 3.0.4-1 based, moving to 3.1 soon

Hypervisor Design Principles and Goals

- Existing applications and binaries must run unmodified
- Support for multi-process, multi-application application environments
  - Permit complex server configurations to be virtualised within a single guest OS instance
- Paravirtualisation (PV) enables high performance and strong isolation between domains
  - Particularly on uncooperative architectures (x86)
- Support up to 100 active VM instances on modern servers
- Live migration of VM instances between servers

xVM architecture



Key Capabilities

- Checkpoint/restart and live migration
  - Managed provisioning
  - Grid operations: virtual platform
- Multiple OSes running simultaneously
  - Solaris, Linux, Windows
  - No longer a boot-time decision
- Special purpose kernels
  - JVM, drivers, filesystems, ...

When should I use it?

- Good for:
  - Develop and test:
    - Fast turn-around time (shutdown and reboot)
    - User-level code
    - Installation
    - General kernel components
  - Older Solaris, Microsoft, Linux, ...
  - “Network in a box”
  - Sharing canned system configurations

- Clone and snapshot of zvols
  - Quickly produce multiple identical guest domains
  - Quickly return to a known stable state

When should I avoid it?

- Not so good for:
  - Timing critical work
  - Heavy IO loads
- Poor for:
  - Hardware drivers
    - Will improve with driver domains, IOMMU

Using xVM: Booting the control domain

- Grub loads the hypervisor, kernel and boot archive:

```
title Solaris xVM
kernel$ /boot/$ISADIR/xen.gz console=com1 com1=9600,8n1
module$ /platform/i86xpv/kernel/$ISADIR/unix
  /platform/i86xpv/kernel/$ISADIR/unix
module$ /platform/i86pc/$ISADIR/boot_archive
```

- Hypervisor:
  - Initialises, probes hardware, etc.
  - Creates dom0 environment around the kernel and boot archive
  - Jumps to dom0 kernel
- Note:
  - Extended Grub syntax to allow expansion of environment specific tokens (`kernel$`, `module$`, `$ISADIR`)
  - Boot archive is separated into 32 bit and 64 bit

Using xVM: dom0 services

- `svc:/system/xvm/store:default`
  - File-based database used to store configuration of known domains
- `svc:/system/xvm/xend:default`
  - Long running daemon used by administrative tools to communicate with the hypervisor
  - Performs much of the work of creating guest domains, migration, etc.
- `svc:/system/xvm/console:default`
  - Mediates access to guest domain consoles (badly)
- `svc:/system/xvm/domains:default`
  - Automatically creates and destroys guest domains at service start/stop time (typically system boot/shutdown)

Using xVM: dom0 tools (1)

- xm
  - Low-level xVM specific command to query the state of the hypervisor, create domains, manipulate configuration, etc.

```
shocks# xm start x1
shocks# xm list
Name                ID   Mem VCPUs      State   Time(s)
Domain-0             0   984     2      r-----    810.3
x1                   2  1023     1      r-----      9.1
shocks# xm console x1
...
x1 console login: root
Password:
Last login: Sat Sep  8 02:02:28 on console
Sep  8 18:00:13 x1 login: ROOT LOGIN /dev/console
Sun Microsystems Inc.   SunOS 5.11      matrix-build-2007-08-21 October 2007
#
```

Using xVM: dom0 tools (2)

- `virsh`
  - Hypervisor agnostic command to query the state of the hypervisor, create domains, manipulate configuration, etc.
    - Only xVM support for now, but Logical Domains, Zones and others coming
  - Built on `libvirt`

```
: shocks#; virsh dominfo x1
Id:             2
Name:           x1
UUID:           b0bece06-8bee-085b-b657-dd642da0daa0
OS Type:        linux
State:          blocked
CPU(s):         1
CPU time:       98.7s
Max memory:     1048576 kB
Used memory:    1047540 kB
: shocks#;
```

Using xVM: dom0 tools (3)

- `virt-install`
  - Facilitate the installation of para-virtual and HVM guests
  - Interactive or command line arguments
  - Install off media (DVD), from an ISO, or over NFS
  - Built on `libvirt`

Solaris PV Guest

```
virt-install -n solarisPV --paravirt -r 1024 \
  --nographics -f /export/solarisPV/root.img -s 16 \
  -l /ws/matrix-gate/public/isos/72-0910/solarisdvd.iso
```

Solaris HVM Guest

```
virt-install -n solarisHVM --hvm -r 1024 --vnc \
  -f /export/solarisHVM/root.img -s 16 \
  -c /ws/matrix-gate/public/isos/72-0910/solarisdvd.iso
```

Using xVM: dom0 tools (3)

- `virt-install` continued

WinXP HVM Guest

```
# virt-install -n winxp --hvm -r 1024 --vnc \
  -f /export/winxp/root.img -s 16 -c
  /windows/media.iso
```

- Set the VNC password property in xend's SMF configuration before starting a HVM domain which uses VNC

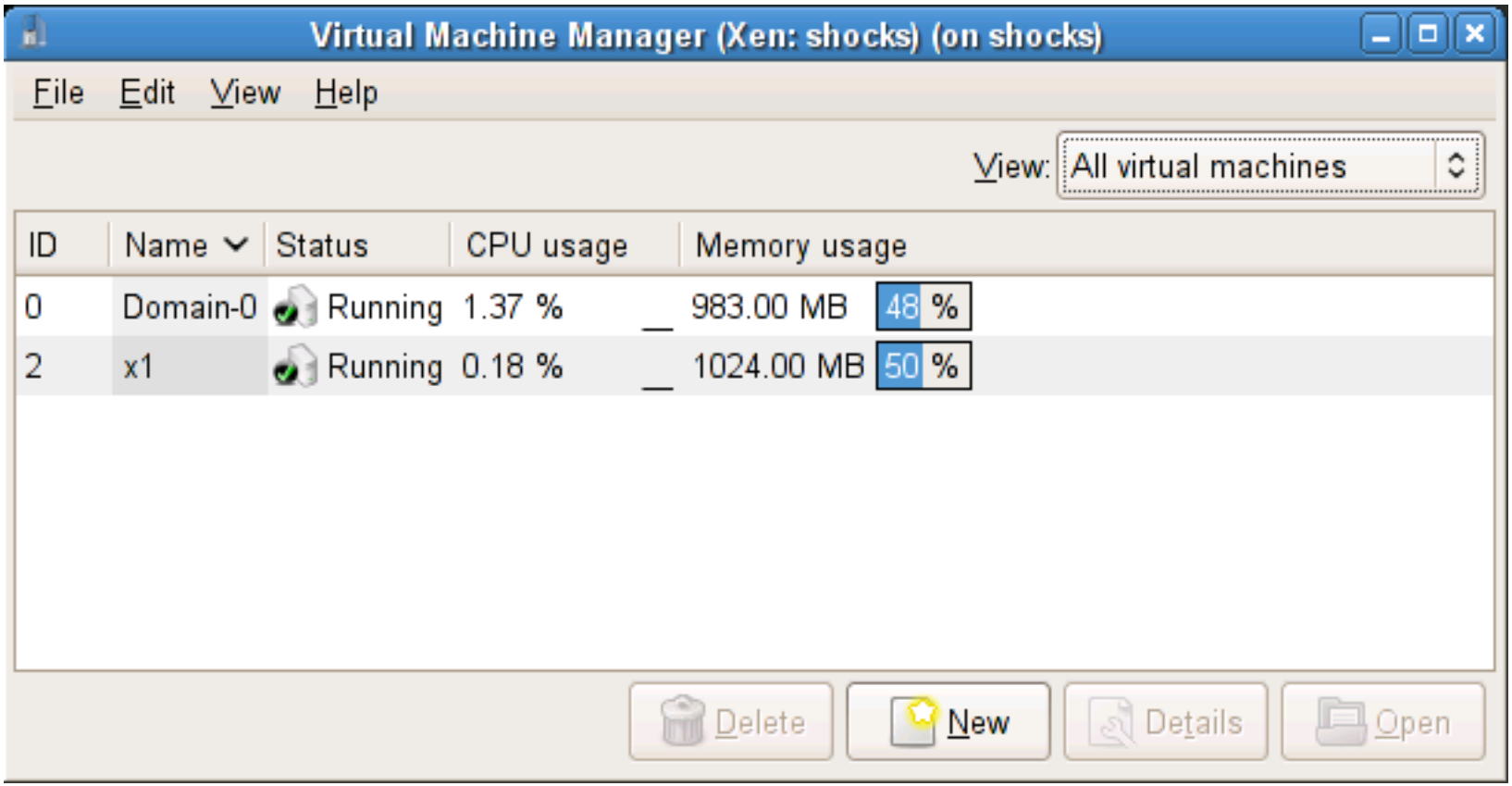
```
# svccfg -s xvm/xend setprop \
        config/vncpasswd = astring: \"somepwd\"
# svcadm refresh xvm/xend; svcadm restart xvm/xend
```

- If remotely displaying the VNC session remotely, you must also set the `vnc-listen` property

```
# svccfg -s xvm/xend setprop \
        config/vnc-listen = astring: \"0.0.0.0\"
# svcadm refresh xvm/xend; svcadm restart xvm/xend
```

Using xVM: dom0 tools (4)

- `virt-manager` (build 76 or later)
  - Gnome desktop application for managing virtual machines
  - Single physical system focus
  - Built on `libvirt`



Using xVM: Guest domain creation

- Create new guest domains using `virt-install`
  - Normal Solaris install for the guest domain, including jumpstart, etc.
  - Linux and HVM (e.g. Windows) install still something of a work in progress
- Acquire guest domain disk images and configuration from others
  - Save the need for everyone to run through the installation
  - Guest domains have relatively small configuration matrix
  - Clone and snapshot of ZFS volumes a powerful management tool

Using xVM: Booting guest domains

- A request to start a guest domain is passed to `xend` by tools (`xm`, `virsh`, ...)
- Guest domain image is created “in core” by `xend`
- Kernel image, boot archive, etc. are located and inserted into the domain image
  - From local files, extracted from guest domain filesystem (`pygrub`), ...
- Backend devices necessary to support the domain are checked and, if necessary, created
  - `lofi` for file based disk images
  - Create virtual NICs
- Domain image and details passed to the hypervisor
- Hypervisor completes domain creation, jumps to the kernel

Using xVM: Debugging the hypervisor

- `printf()` is your friend (or not)
- If the hypervisor panics, Solaris can usually take a dump
  - Includes the hypervisor image, which looks like a kernel module in the dump

Using xVM: Debugging dom0 and domU

- Typical OpenSolaris tools work well
  - `mdb`, `kmdb`, `dtrace`
- The hypervisor console can be used to send a 'break' signal to domains
  - Type '`^A^A^A`' at the hypervisor console to start
  - Particularly useful for dom0
- Dom0 tools can be used to:
  - Send a 'break' signal to guest domains:
    - `xm sysrq b <domain>`
  - Dump the image of a guest domain, for use with `mdb`:
    - `xm dump-core <domain> <dump-file>`
    - `mdb <dump-file>`

Full virtualisation (HVM)

- Some operating systems have not been paravirtualised
  - Microsoft, older Solaris, older Linux, OS/2 (!), ...
- New processor features to enable full virtualisation
  - Intel VT and AMD SVM
    - Needs to be enabled by the BIOS, so having the right CPU may not be enough
  - Trap to the hypervisor for “unsafe” instructions, memory access, etc.
    - Hypervisor emulates some effects, uses device emulation for others
- More features coming to provide more assist
  - Nested page tables, improved VT/SVM, ...

HVM: IO device emulation

- A subset of QEMU (`qemu-dm`) is used to provide IO device emulation
  - VGA (Cirrus Logic)
  - IDE controller
  - NIC (AMD PCnet and RTL8139)
- Hardware emulation runs in user-space in dom0:
  - Trap on emulated hardware access by HVM domain
  - Hypervisor passes details to `qemu-dm`
  - `qemu-dm` emulates, signals hypervisor on completion
  - Hypervisor re-starts HVM domain
- Performance is not great

HVM Console access (1)

Need a way to display the emulated framebuffer:

- VNC
  - `qemu-dm` exports the virtual framebuffer as a VNC server
  - Reusable sessions
  - Standard VNC protocol, compatible with most viewers
  - Solaris has a bundled client
    - `java -jar /usr/share/gnome/vino/vnic-client.jar`
- `libSDL`
  - Simple X11 window shows virtual framebuffer
  - Grabs keyboard and mouse for guest
    - `ctrl-alt` breaks grab

HVM Console Access (2)

- Remote Desktop Protocol (RDP)
  - Windows has built-in RDP server
    - Solaris does not, yet
  - Best option for controlling Windows, even forwards audio
  - Enable via Windows Control Panel
    - “System” -> “Remote”
  - `rdesktop` is open source RDP client, targeting future Solaris integration
  - Available for preview at `/ws/matrix-gate/public/bin/rdesktop`

Porting OpenSolaris

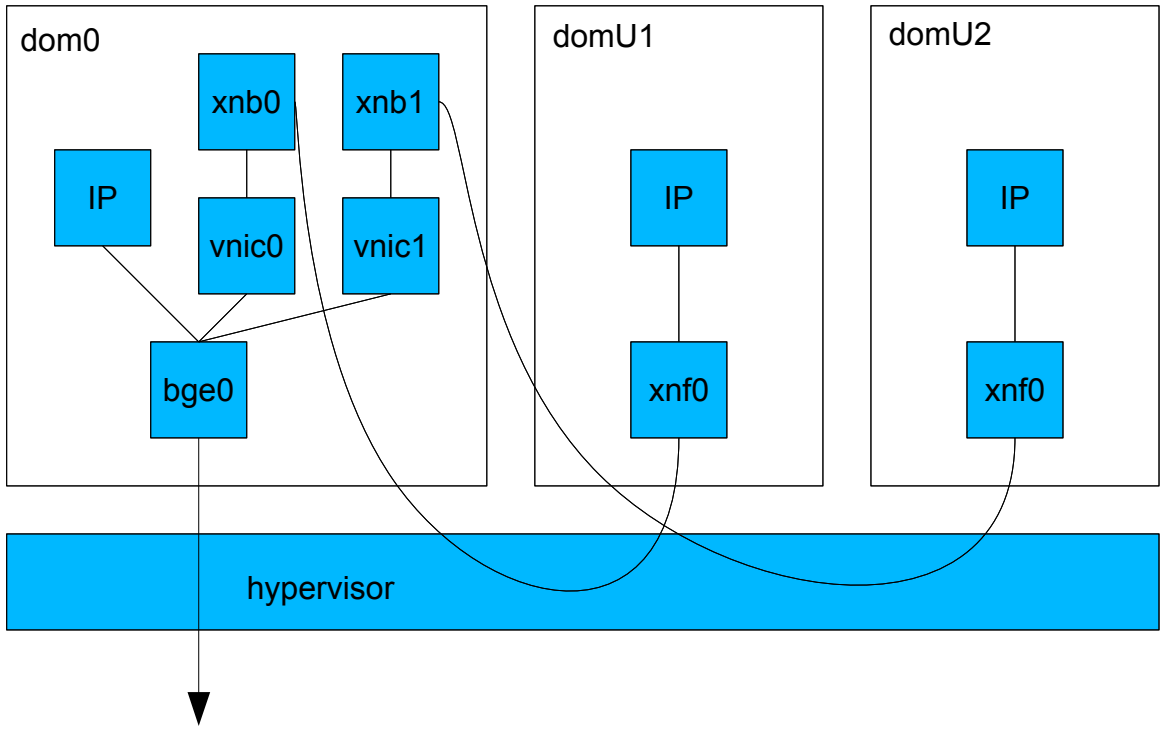
- A new platform, `i86xpv`
  - As much as possible shared with `i86pc`
- Platform support module replaces direct hardware access with hypervisor calls
  - Page table manipulation, interrupt management, clock, ...
- Implement inter-domain protocols for PV console, disk and network IO:
  - Frontend drivers fit in to standard frameworks (e.g. GLD) as providers
  - Backend drivers provide access to dom0 resources
- Implement inter-domain protocols for access to configuration database

Inter-domain protocols

- Hypervisor provided facilities are used to implement communication paths:
  - Shared memory
  - Transfer of ownership of memory
  - Send and receive of event notifications
- The hypervisor reference OS implementation defines a set of communications protocols using these facilities:
  - Console IO: simple character IO
  - Network IO: “point to point” ethernet segment
  - Disk IO: a simple block device

Network Backend

- Provide access to shared physical device
- Early cut of Crossbow virtual NIC (VNIC) implementation



Disk Backend

- Open devices using layered operations
- Various options for storage:
  - Existing physical device
  - or partition
  - ZFS volume
  - SVM volume
  - Plain file (lofi)
- Embedded labels makes access from dom0 “difficult”

Where are the pieces?

- OpenSolaris kernel parts are in the Nevada gate
  - managed using TeamWare, for now
- The hypervisor, tools, utilities and associated bits are in the xVM gate
  - `/ws/xvm-gate` internally
    - Externally on `opensolaris.org` soon
  - Five mercurial repositories:
    - `xen.hg`: a child of the open source project gate, using `mq` to manage a queue of patches
    - `sunos.hg`: build infrastructure, OpenSolaris specific scripts, etc.
    - `libvirt.hg`, `urlgrabber.hg`, `virtinst.hg`: imported tools and patches for local use, some using `mq` to manage patches

Danger, Will Robinson!

- There are a small number of header files in the Nevada and xVM gates that must be kept in sync:
  - `usr/src/uts/common/xen/public/`
  - `xen.hg/xen/include/public/`

When things go wrong

- New bug categories:
  - `solaris/xvm/hypervisor`
  - `solaris/xvm/kernel`
  - `solaris/xvm/utility`
- Log files in `/var/log/xen`:
  - `xend.log` – logging and backtraces from the long running daemon
  - `xpvd-event.log` – logs from backend device creation, removal, etc.

PV drivers for Solaris 10

- No PV version of Solaris 10
  - IO performance using emulated hardware (IDE and RTL8139) is poor
- Provide PV disk and network drivers for older Solaris releases
- Bundled in a future Solaris 10 update
- Performance of PV drivers in HVM domain looks similar to that of a fully PV guest domain

Futures

- Move to a newer version of the hypervisor (3.1)
- Examine performance closely, and then fix it
- Driver domain support
- Higher performance networking IO
  - Inter-domain protocol extensions
  - Hybrid IO
  - Crossbow
- Support for more disk image formats
  - `blktap` style approach
- PCI IOV
- Fault management
- ...

Finding out more

- OpenSolaris community
  - xen-discuss@opensolaris.org
  - http://opensolaris.org/os/community/xen
  - irc://irc.oftc.net/solaris-xen

With help from...

- Chris, Russ, Joe, Gina, Kevin, Todd, David, Penny, John, Claudia, Nicolas, David, Roger, Bill, Mark, Rob, Susan, Bill, John, Angela, Tariq, Gavin, Tim, Stu, Jerri-Ann, Allan, Nils, Gary, Ed, Joost, Shalon, Michael, Ryan, Jan, Ann, Frank, Kirk, John, Steve, Max...
- ...and everyone who knows me.

開
放
的
열린
مفتوح
libre
मुक्त
ముక్త
livre
libero
ಮುಕ್ತ
开放的
açık
open
nyílt
פתוח
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை

Thank you!

David Edmondson
Solaris Engineering
dme@sun.com
http://dme.org

“open” artwork and icons by chandan:
http://blogs.sun.com/chandan

Glossary

- dom0: control domain
- domU: guest domain
- HVM: Hardware (assisted) Virtual Machine
- PV: paravirtualised

HVM capable hardware

- M2 variants of Sun AMD machines
- Intel Core processors
  - But check the BIOS

Porting History

- Guest domain support on version 2.0.7
  - Internal only, based on nv21
- Guest and control domain support on version 3.0
  - Public releases based on nv44 and nv66
  - Integrated into nv75
    - http://opensolaris.org/os/community/on/flag-days/pages/2007091801/