

**virtualiron**



# “Hostless” Xen Deployment

**Xen Summit**

**Fall 2007**

David Lively

[dlively@virtualiron.com](mailto:dlively@virtualiron.com)

[dave.lively@gmail.com](mailto:dave.lively@gmail.com)

# “Hostless” Xen Deployment

- What “Hostless” Means
- Motivation
- System Architecture
- Challenges and Solutions
- Questions?

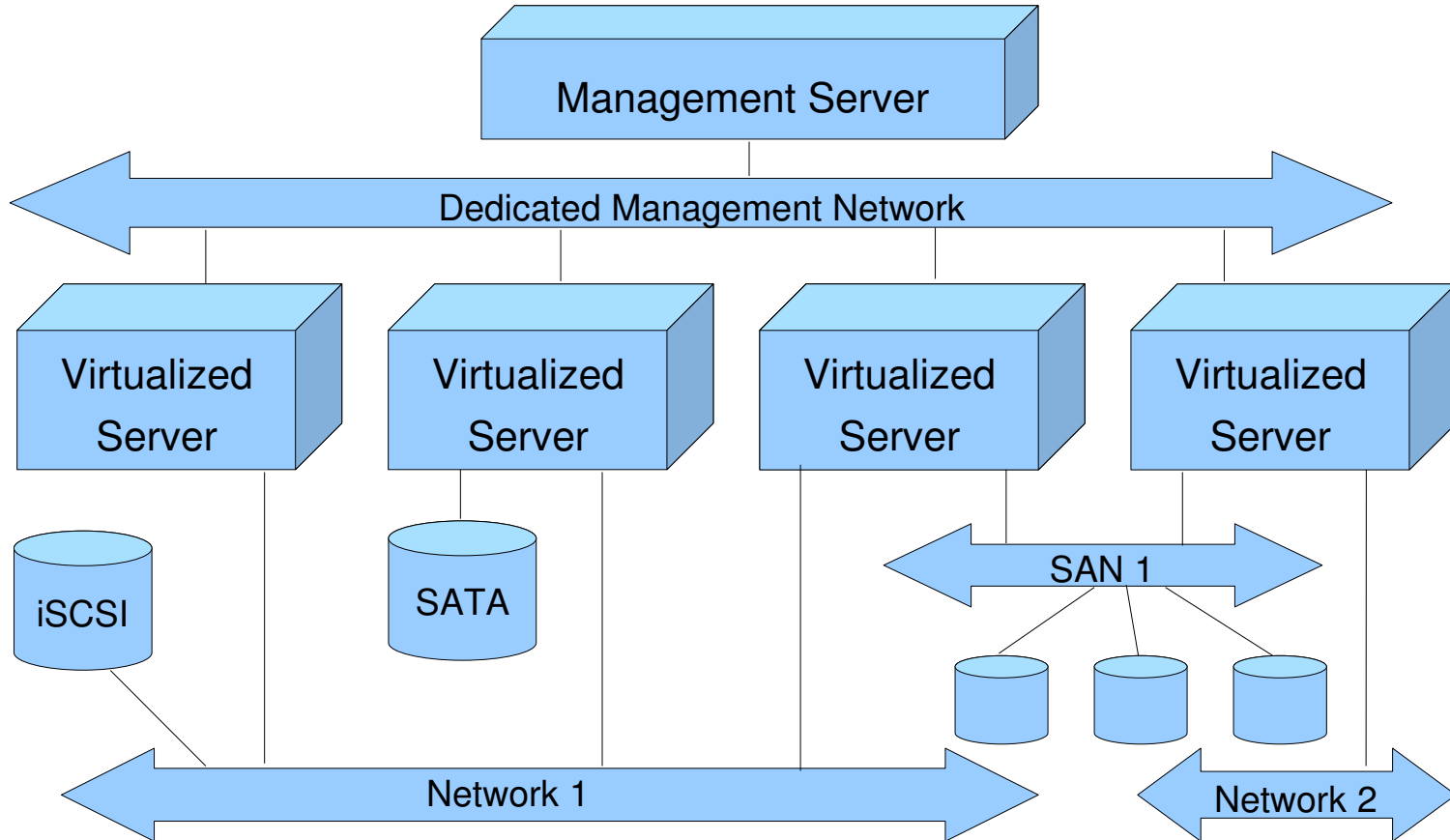
# “Hostless” means:

- No installation of hypervisor & dom0
- Servers booted via network (or flash)
- No need for local storage on servers

# Motivation

- Security & Robustness
  - No user access to dom0
  - Dom0 for virtualization *only*
    - No other arbitrary apps running
- Manageability
  - Scales easily
  - Updates easily propagated

# System Architecture

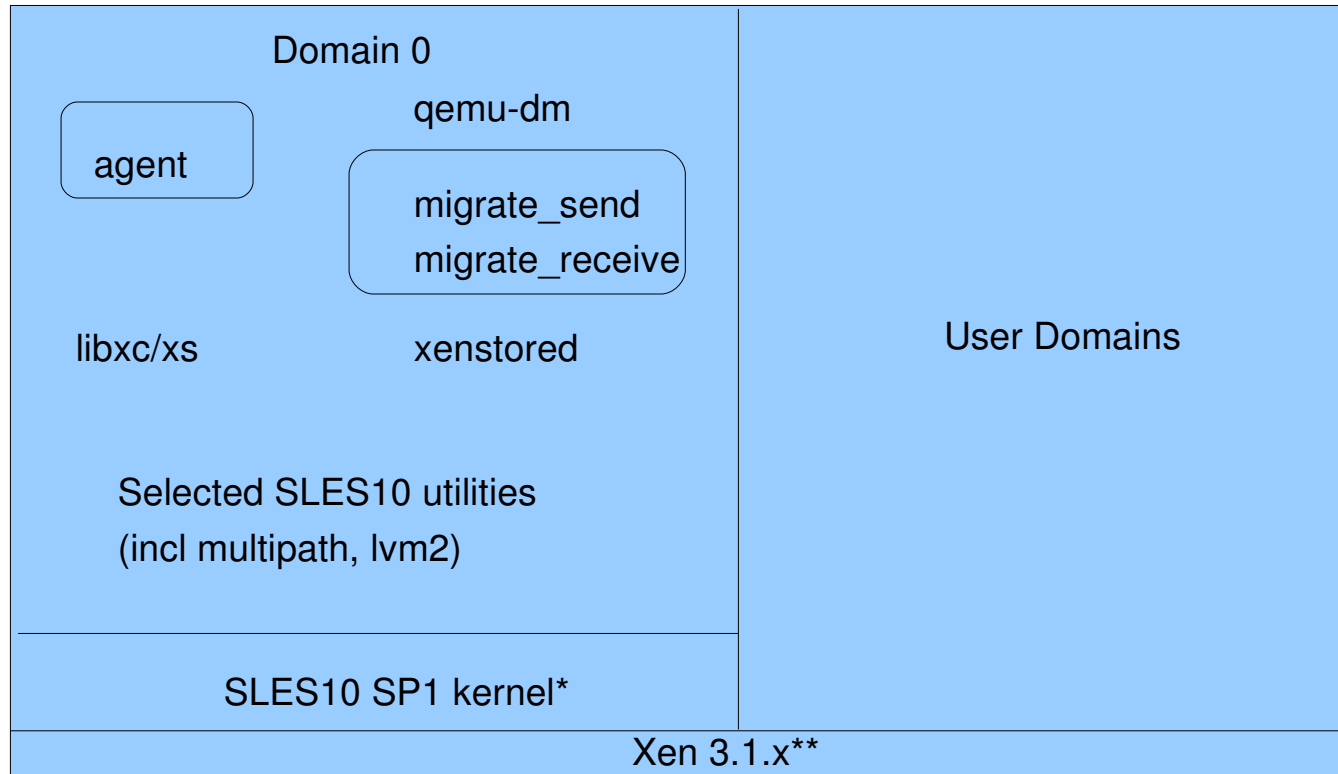


# System Architecture

- Management Server
  - Has all persistent state
  - Discovers managed servers and their connectivity to shared resources
  - Generally orchestrates servers
    - Boot, reboot, power control
    - Launches guests and implements migration policies
  - User doms keep running w/o Mgmt Server

# System Architecture

## Virtualized Server



# System Architecture

- Current Server Control Stack
  - Server “agent” talks to Mgmt Server
    - Re-purposed from our previous product (our own cluster hypervisor)
  - Supports HVM guests only (for now)
  - Custom migration control code
  - Provides Mgmt Server access to dom0:
    - Virtualization services
    - Administration services

# Challenges & Solutions

- Dealing with Multiple Servers
- Distributed Resource Discovery
- Dom0 Administration Services
- Limiting Dom0 Memory Footprint
- Buffer-Cache Issues
- Working Downstream

# Challenges & Solutions

- Dealing with Multiple Servers
  - Responsibility of Mgmt Server
  - Constructs connectivity matrix of servers to:
    - Networks
    - Storage (Local, FC, or iSCSI)
  - Arbitrates access to these shared resources
  - Tracks CPU features to disallow incompatible migrations (e.g., between Intel & AMD)

# Challenges & Solutions

- Dom0 Administrative Services
  - Physical hardware monitoring
  - Logical Volume Management
  - Network configuration and control
  - Multipath I/O
    - Fiber channel
    - Network
    - iSCSI
    - Typically hardware-specific :-)

# Challenges & Solutions

- Distributed Resource Discovery
  - What does each server see for:
    - Networks
    - Storage
      - Remote (recommended) or local
      - Logical Volumes
      - File-backed
        - Requires mounting underlying fs
        - Mgmt Server coordinates parallel scan

# Challenges & Solutions

- Limiting dom0 memory footprint
  - Keep ramdisk image small
    - No need to go crazy, though
  - Keep kernel page array small
    - Most memory is for user domains
    - So hide it from dom0 via boot arg
  - Keep write portion of buffer cache small
    - No need to go crazy, though

# Challenges & Solutions

- Buffer-cache issues
  - Memory usage
  - Consistency w/block drivers
    - First tried using O\_DIRECT in QEMU
      - Very invasive due to alignment issues
    - Now using fsync / fadvise(DONT\_NEED)
      - Not nearly as invasive
      - Faster
      - Batched

# Challenges & Solutions

- Working Downstream
  - Branching is necessary
    - Producing a stable release demands it
    - Downstream contributors often have different focus
      - This is a Good Thing
  - Branching is normal
    - Look at kernel.org and SuSE/Redhat
  - It's the *amount* of divergence that matters

# Challenges & Solutions

- Working Downstream
  - Too much divergence bad in both directions
    - Painful for downstream to take new features and fixes
    - Many downstream features / fixes won't apply upstream
  - Amount of divergence a function of upstream maturity *in the areas used by each particular downstream vendor*
    - e.g. VI focus on 64-bit hypervisor & HVM guests only

# Challenges & Solutions

- Working Downstream
  - Example: Our HVM shadow/migration code
    - Very necessary at the time
    - Proposed work covered by shadow2
      - But shadow2 has broader scope
        - So it was harder, and took longer
  - Result: too much VI-specific code
    - Works fine, but *hard to maintain*

# Challenges & Solutions

- Working Downstream

- Over time, we're converging with upstream
  - (Mostly) automated process maintains our changes as separate patches
    - Easier to keep up and contribute back
  - Using xenbus / xenstore since our 4.0 rls
  - Using shadow2 code since our 4.0 rls
  - Control stack migrating to XenAPI

**VirtualIron**



“Hostless” Xen Deployment

**Xen Summit**

**Fall 2007**

David Lively  
dlively@virtualiron.com  
dave.lively@gmail.com

# “Hostless” Xen Deployment

- What “Hostless” Means
- Motivation
- System Architecture
- Challenges and Solutions
- Questions?

This talk was originally called “diskless Xen deployment”, but “diskless” isn't quite right, since our guests have access to disks. “Embedded” is a little more accurate, but to me embedded suggests small(ish) devices. So I've used the term “hostless” (which I hated when I first heard it from our marketing folks). I'll explain what I mean by this in the next slide.

## “Hostless” means:

- No installation of hypervisor & dom0
- Servers booted via network (or flash)
- No need for local storage on servers

i.e., hypervisor/dom0 not installed to persistent storage (disk) – no installation to manage, is really the point

Note local storage **can be used by guests – just not required.**

# Motivation

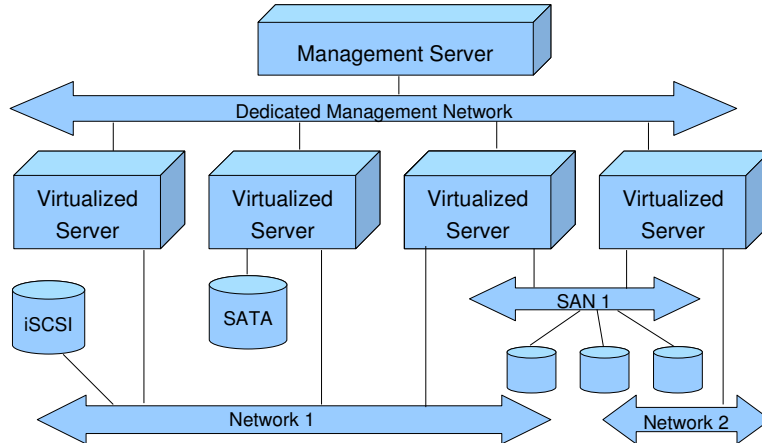
- Security & Robustness
  - No user access to dom0
  - Dom0 for virtualization *only*
    - No other arbitrary apps running
- Manageability
  - Scales easily
  - Updates easily propagated

Intro: our product is aimed at data center virtualization – many servers in a complex pre-existing environment. We put a high value on ease of deployment and ease of use in this environment. In particular, we do not want our customers to have to manage a Linux system (or even worse, a bunch of Linux systems).

Scales easily --- in terms of deployment, that is.

Also note that disallowing use of dom0 by users makes VM performance more predictable.

# System Architecture



This picture shows the system architecture in the network boot case. Note the dedicated management network.

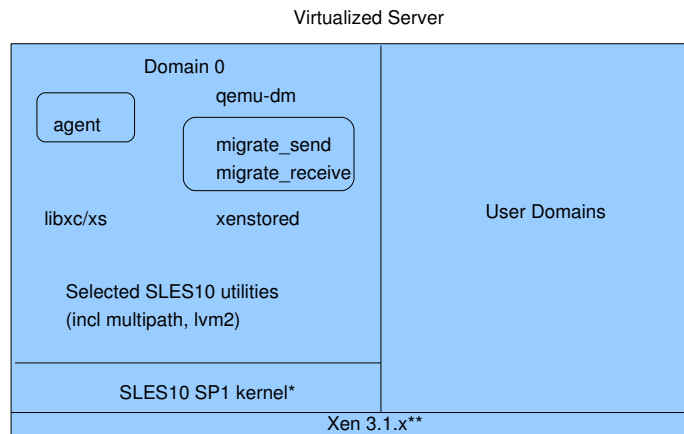
While this scheme is great for some deployments, the reliance on a dedicated mgmt network is problematic in some cases. So we're working on relaxing this requirement to support other (e.g., flash-based) deployments.

Note that managed servers may have different connectivity to shared resources (networks, storage).

# System Architecture

- Management Server
  - Has all persistent state
  - Discovers managed servers and their connectivity to shared resources
  - Generally orchestrates servers
    - Boot, reboot, power control
    - Launches guests and implements migration policies
  - User doms keep running w/o Mgmt Server

# System Architecture



7

VirtualIron

Let's dive down into the architecture of a virtualized server.

Non-standard components are in the rounded boxes. Note these are all GPLd.

Note we (currently) aren't using xend – agent implements much of the same functionality --- dead kittens everywhere!! A little later I'll explain how that came to be, and what we're doing about it.

SP1\* kernel – plus minimal modifications to make it run with Xen 3.1.x (originally based on 3.0.4):

- \* update Xen headers
- \* backport key backend driver changes

Xen 3.1.x\*: This is our patched version of 3.1.x.

# System Architecture

- Current Server Control Stack
  - Server “agent” talks to Mgmt Server
    - Re-purposed from our previous product (our own cluster hypervisor)
  - Supports HVM guests only (for now)
  - Custom migration control code
  - Provides Mgmt Server access to dom0:
    - Virtualization services
    - Administration services

# Challenges & Solutions

- Dealing with Multiple Servers
- Distributed Resource Discovery
- Dom0 Administration Services
- Limiting Dom0 Memory Footprint
- Buffer-Cache Issues
- Working Downstream

# Challenges & Solutions

- Dealing with Multiple Servers
  - Responsibility of Mgmt Server
  - Constructs connectivity matrix of servers to:
    - Networks
    - Storage (Local, FC, or iSCSI)
  - Arbitrates access to these shared resources
  - Tracks CPU features to disallow incompatible migrations (e.g., between Intel & AMD)

Arbitrates access:

e.g. Don't allow two guests to access the same disk partition.

\* We also disallow migration of guests started on servers with the NX/XD capability to servers without that capability (or with it disabled in BIOS).

# Challenges & Solutions

- Dom0 Administrative Services
  - Physical hardware monitoring
  - Logical Volume Management
  - Network configuration and control
  - Multipath I/O
    - Fiber channel
    - Network
    - iSCSI
    - Typically hardware-specific :-)

Mostly a matter of plumbing agent through to various “standard” (SLES10) utilities, sometimes with the help of intermediary scripts.

The netwk config/control includes bonding, VLAN config.

Multipath – not rocket science, but challenging nonetheless, mostly due to the immaturity of generic Linux multipath and lack of interoperability among hw-vendor-supplied Linux multipath implementations.

# Challenges & Solutions

- Distributed Resource Discovery
  - What does each server see for:
    - Networks
    - Storage
      - Remote (recommended) or local
      - Logical Volumes
      - File-backed
        - Requires mounting underlying fs
        - Mgmt Server coordinates parallel scan

Note we control Logical Volume / Volume Group creation & modification--- user does not need to understand or use LVM. Mgmt server is responsible for propagating updated LVM info to all servers.

Discovering file-backed VM storage is the trickiest part of discovery. Only one server can mount a filesystem at once. The mgmt server coordinates this. The first (native) approach accomplishes this by discovering one server at a time. Now this is being changed to serialize on a per-filesystem basis, allowing discovery to scale much better.

# Challenges & Solutions

- Limiting dom0 memory footprint
  - Keep ramdisk image small
    - No need to go crazy, though
  - Keep kernel page array small
    - Most memory is for user domains
    - So hide it from dom0 via boot arg
  - Keep write portion of buffer cache small
    - No need to go crazy, though

# Challenges & Solutions

- Buffer-cache issues
  - Memory usage
  - Consistency w/block drivers
    - First tried using O\_DIRECT in QEMU
      - Very invasive due to alignment issues
    - Now using fsync / fadvise(DONT\_NEED)
      - Not nearly as invasive
      - Faster
      - Batched

Our use of fsync in QEMU keeps the writeback portion of the buffer cache small and (more importantly) makes modifications to QEMU-emulated devices make it to disk. Otherwise subsequent reads of the same disk via the PV driver will get stale data.

The fadvise(DONT\_NEED) keeps potentially-stale read data out of the buffer cache. This can cause problems in the following situation:

- \* QEMU reads block from backing disk (stays in buffer cache)
- \* PV driver takes over disk, writes new data to this block (but blkback bypasses the buffer cache, so this doesn't invalidate the old data QEMU read)
- \* Guest reboots and accesses emulated disk, getting old stale data.

# Challenges & Solutions

- Working Downstream
  - Branching is necessary
    - Producing a stable release demands it
    - Downstream contributors often have different focus
      - This is a Good Thing
  - Branching is normal
    - Look at kernel.org and SuSE/Redhat
  - It's the *amount* of divergence that matters

## Stable rls:

... unless you're doing the CentOS thing, you're doing some amount of integration, which often requires branching (if only to fix bugs found going down paths that aren't normally exercised). This is really the only course that's responsible to your users.

# Challenges & Solutions

- Working Downstream
  - Too much divergence bad in both directions
    - Painful for downstream to take new features and fixes
    - Many downstream features / fixes won't apply upstream
  - Amount of divergence a function of upstream maturity *in the areas used by each particular downstream vendor*
    - e.g. VI focus on 64-bit hypervisor & HVM guests only

# Challenges & Solutions

- Working Downstream
  - Example: Our HVM shadow/migration code
    - Very necessary at the time
    - Proposed work covered by shadow2
      - But shadow2 has broader scope
        - So it was harder, and took longer
  - Result: too much VI-specific code
    - Works fine, but *hard to maintain*

The pain of divergence serves as a powerful corrective force here!

# Challenges & Solutions

- Working Downstream
  - Over time, we're converging with upstream
    - (Mostly) automated process maintains our changes as separate patches
      - Easier to keep up and contribute back
    - Using xenbus / xenstore since our 4.0 rls
    - Using shadow2 code since our 4.0 rls
    - Control stack migrating to XenAPI

Note that we now have a continuous ongoing effort to get / keep our patches ported to unstable and submitted when ready.