CLOUD COMPUTING

- Xen
  - Arhitektura
  - Instalacija
  - Korišćenje

Architecture

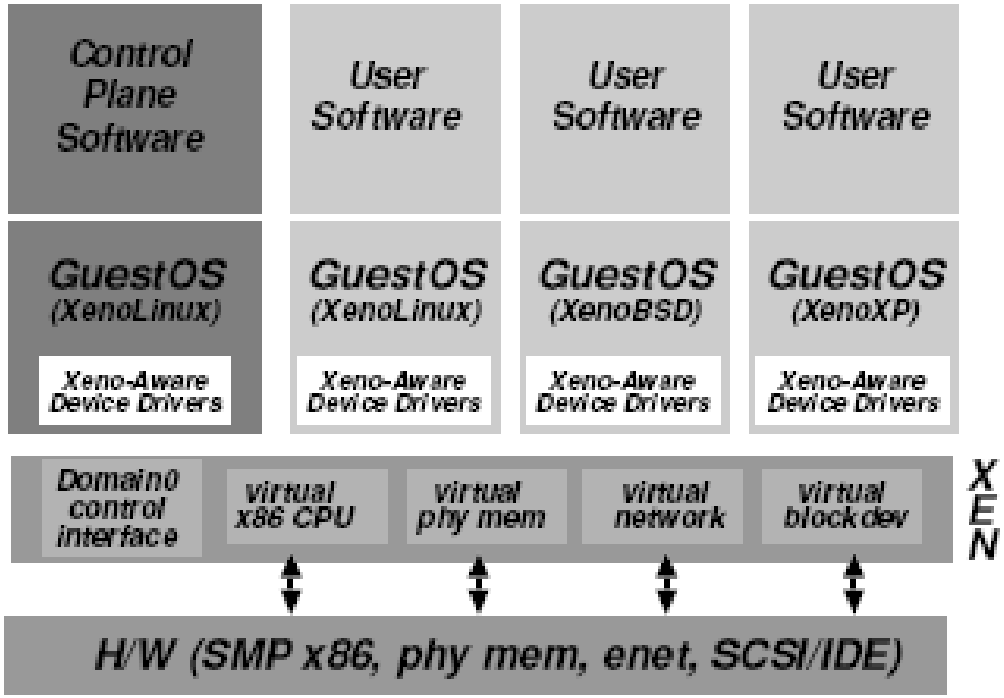


Figure 1: The structure of a machine running the Xen hypervisor, hosting a number of different guest operating systems, including *Domain0* running control software in a XenoLinux environment.

Xen

2.3 Control and Management

- Throughout the design and implementation of Xen, a goal has been to separate policy from mechanism wherever possible. Although the hypervisor must be involved in data-path aspects (for example, scheduling the CPU between domains, ltering network packets before transmission, or enforcing access control when reading data blocks), there is no need for it to be involved in, or even aware of, higher level issues such as how the CPU is to be shared, or which kinds of packet each domain may transmit.

- **The resulting architecture** is one in which the hypervisor itself provides only basic control operations.
- These are exported through an interface accessible from authorized domains; potentially complex policy decisions, such as admission control, are best performed by management software running over a guest OS rather than in privileged hypervisor code.

2.3 Control and Management

- The overall system structure is illustrated in Figure 1.
- Note that a domain is created at boot time which is permitted to use the *control interface*.
- This initial domain, termed ***Domain0***, is **responsible** for hosting the application-level management software.
- The control interface provides
  - the **ability to create** and **terminate** other domains
  - and
  - to **control** their associated
    - ✔ scheduling parameters
    - ✔ physical memory allocations
    - ✔ and the access they are given to the machine's physical disks and network devices

2.3 Control and Management

- In addition to processor and memory resources
  - the control interface supports
  - the creation and deletion of
    - ✔ **virtual network interfaces (VIFs)**
    - ✔ and
    - ✔ **block devices (VBDs)**
- These virtual I/O devices have associated access-control information
  - which determines which domains can access them,
  - and with what restrictions (for example, a read-only VBD may be created, or a VIF may lter IP packets to prevent source-address spoong).

- This **control interface**, together with proling statistics on the current state of the system, is exported to a suite of application level management software running in *Domain0*
- This **complement of administrative tools** allows convenient management of the entire server: current tools can create and destroy domains, set network filters and routing rules, monitor per-domain netwok activity at packet and ow granularity, and create and delete virtual network interfaces and virtual block devices. We anticipate the development of higher-level tools to further automate the application of administrative policy

3. DETAILED DESIGN

- In this section we introduce the design of the major subsystems that make up a Xen-based server.
- In each case **we present both Xen and guest OS functionality for clarity of exposition.**
- The current discussion of guest OSes focuses on XenoLinux as this is the most mature; nonetheless our ongoing porting of Windows XP and NetBSD gives us condence that Xen is guest OS agnostic.

3.1 Control Transfer: Hypercalls and Events

- **Two mechanisms** exist for control interactions between Xen and an overlying domain:
  - **synchronous calls** from a domain to Xen may be made using a ***hypercall***
  - while notications are delivered to domains from Xen using an **asynchronous event mechanism**

- The hypercall interface allows domains to perform a synchronous software trap into the hypervisor to perform a privileged operation, analogous to the use of system calls in conventional operating systems.
- An example use of a hypercall is to request a set of page table updates, in which Xen validates and applies a list of updates, returning control to the calling domain when this is completed.

3.1 Control Transfer: Hypercalls and Events

- Communication from Xen to a domain is provided through an asynchronous event mechanism, which replaces the usual delivery mechanisms for device interrupts and allows lightweight notication of important events such as domain-termination requests.
- Like to traditional Unix signals, there are only a small number of events, each acting toflag a particular type of occurrence. For instance, events are used to indicate that new data has been received over the network, or that a virtual disk request has completed.
- Pending events are stored in a per-domain bitmask which is updated by Xen before invoking an event-callback handler specified by the guest OS.
- The callback handler is responsible for resetting the set of pending events, and responding to the notications in an appropriate manner.
- A domain may explicitly defer event handling by setting a Xen-readable software ag: this is analogous to disabling interrupts on a real processor

3.2 Data Transfer: I/O Rings

- The presence of a hypervisor means there is an **additional protection** domain between guest OSes and I/O devices, so it is crucial that a data transfer mechanism be provided that allows data to move vertically through the system with as little overhead as possible.

- Two main factors have shaped the design of our I/O-transfer mechanism:
- **resource management**
- **and**
- **event notication**

- For resource accountability, we attempt to minimize the work required to demultiplex data to a specic domain when an interrupt is received from a device . the overhead of managing buffers is carried out later where computation may be accounted to the appropriate domain.
- Similarly, memory committed to device I/O is provided by the relevant domains wherever possible to prevent the crosstalk inherent in shared buffer pools; I/O buffers are protected during data transfer by pinning the underlying page frames within Xen.

3.2 Data Transfer: I/O Rings

- Figure 2 shows the structure of our I/O descriptor rings.
- A ring is a **circular queue** of descriptors allocated by a domain but accessible from within Xen.
- Descriptors do not directly contain I/O data; instead, I/O data buffers are allocated out-of-band by the guest OS and indirectly referenced by I/O descriptors.
- Access to each ring is based around two pairs of producer-consumer pointers:
  - domains place **requests** on a ring,
  - advancing a request producer pointer,
  - and Xen removes these requests for handling, advancing an associated request consumer pointer.

- **Responses** are placed back on the ring similarly, save with Xen as the producer and the guest OS as the consumer. There is no requirement that requests be processed in order: the guest OS associates a unique identier with each request which is reproduced in the associated response.
- This allows Xen to unambiguously reorder I/O operations due to scheduling or priority considerations

3.2 Data Transfer: I/O Rings

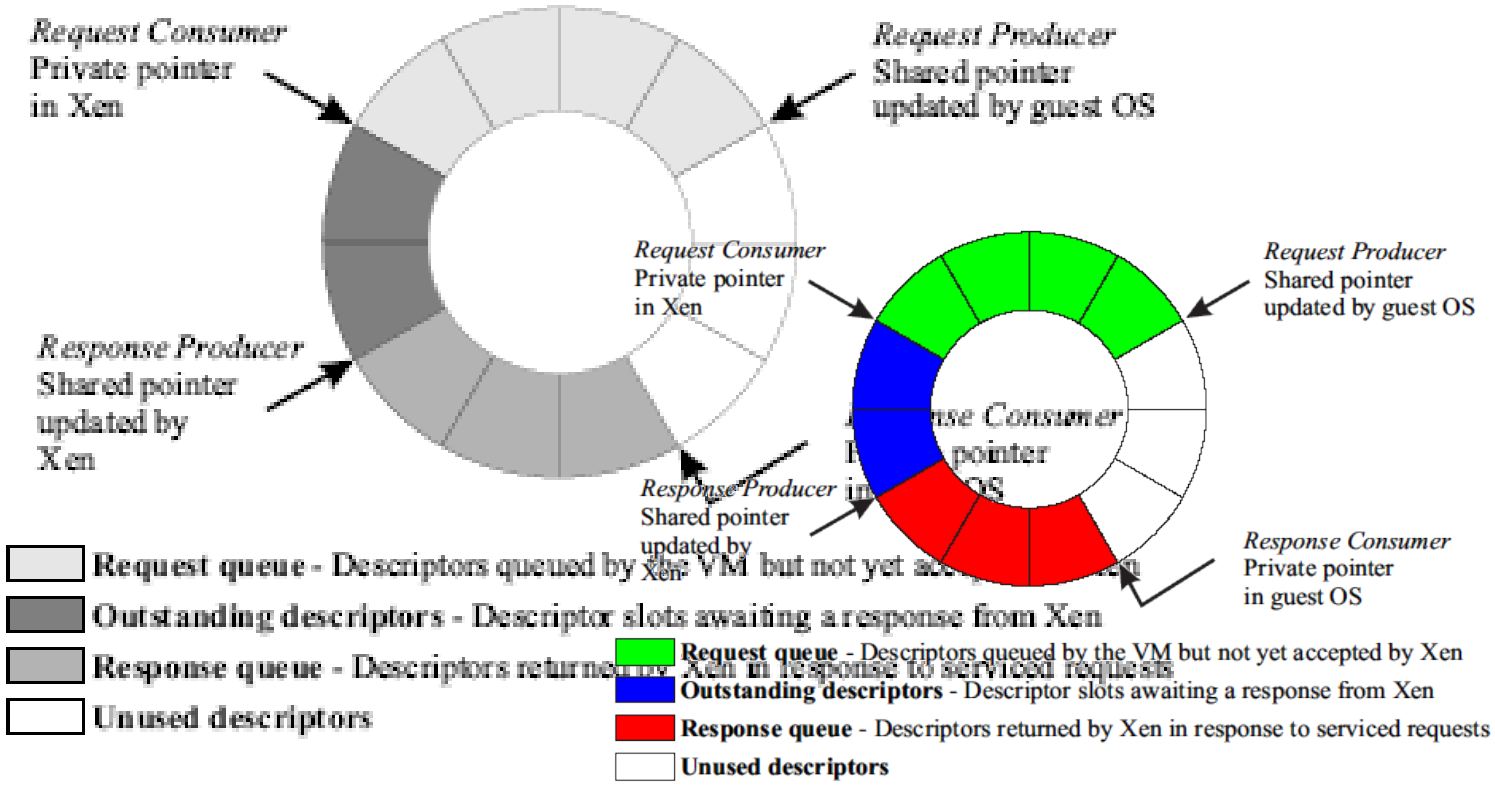


Figure 2: The structure of asynchronous I/O rings, which are used for data transfer between Xen and guest OSes.

3.2 Data Transfer: I/O Rings

- This structure is sufciently generic to support a number of different device paradigms.

- For example, a set of `requests' can provide buffers for network packet reception; subsequent `responses‘ then signal the arrival of packets into these buffers.

- Reordering is useful when dealing with disk requests as it allows them to be scheduled within Xen for efciency, and the use of descriptors with out-of-band buffers makes implementing zero-copy transfer easy.

3.2 Data Transfer: I/O Rings

- We decouple the production of requests or responses from the notication of the other party:
- in the case of requests, a domain may enqueue multiple entries before invoking a hypercall to alert Xen;
- in the case of responses, a domain can defer delivery of a notication event by specifying a threshold number of responses.

3.3 Subsystem Virtualization

- The control and data transfer mechanisms described are used in
- our virtualization of the various subsystems.

- In the following, we
- discuss how this virtualization is achieved for
  - CPU,
  - timers,
  - memory,
  - network
  - and
  - disk.

3.3.1 CPU scheduling

- Xen currently schedules domains according to the Borrowed Virtual Time (BVT) scheduling algorithm
- We chose this particular algorithms since it is both work-conserving and has a special mechanism for low-latency wake-up (or *dispatch*) of a domain when it receives an event.

- Fast dispatch is particularly important to minimize the effect of virtualization on OS subsystems that are designed to run in a timely fashion; for example, TCP relies on the timely delivery of acknowledgments to correctly estimate network round-trip times.

- BVT provides low-latency dispatch by using virtual-time warping, a mechanism which temporarily violates `ideal' fair sharing to favor recently-woken domains. However, other scheduling algorithms could be trivially implemented over our generic scheduler abstraction.
- Per-domain scheduling parameters can be adjusted by management software running in *Domain0*.

3.3.2 Time and timers

- Xen provides guest OSes with notions of
  - **real time**
  - **virtual time**
  - **wall-clock time**

- Real time is expressed in nanoseconds passed since machine boot and is maintained to the accuracy of the processor's cycle counter and can be frequency-locked to an external time source (for example, via NTP).

- A domain's virtual time only advances while it is executing: this is typically used by the guest OS scheduler to ensure correct sharing of its timeslice between application processes.

- Finally, wall-clock time is specied as an offset to be added to the current real time. This allows the wall-clock time to be adjusted without affecting the forward progress of real time.

- Each guest OS can program a **pair of alarm timers**, **one for real time** and the other **for virtual time**. Guest OSes are expected to maintain internal timer queues and use the Xen-provided alarm timers to trigger the earliest timeout. Timeouts are delivered using Xen's event mechanism.

3.3.3 Virtual address translation

- As with other subsystems, Xen attempts to virtualize memory access with as little overhead as possible.
- As discussed in Section 2.1.1, this goal is made somewhat more dificult by the x86 architecture's use of hardware page tables.
- The approach taken by VMware is to provide each guest OS with a virtual page table, not visible to the memory-management unit (MMU).
- The hypervisor is then responsible for trapping accesses to the virtual page table, validating updates, and propagating changes back and forth between it and the MMU-visible `shadow' page table.
- This greatly increases the cost of certain guest OS operations, such as creating new virtual address spaces, and requires explicit propagation of hardware updates to `accessed' and `dirty' bits.

3.3.3 Virtual address translation

- Although full virtualization forces the use of shadow page tables, to give the illusion of contiguous physical memory, Xen is not so constrained.
- Indeed, Xen need only be involved in page table *updates*, to prevent guest OSes from making unacceptable changes.
- Thus we avoid the overhead and additional complexity associated with the use of shadow page tables . the approach in Xen is to register guest OS page tables directly with the MMU, and restrict guest OSes to read-only access.
- Page table updates are passed to Xen via a hypercall; to ensure safety, requests are *validated* before being applied.

3.3.3 Virtual address translation

- To aid validation, we associate a type and reference count with each machine **page frame**. A **frame** may have any one of the following mutually-exclusive types at any point in time: page directory (PD), page table (PT), local descriptor table (LDT), global descriptor table (GDT), or writable (RW). Note that a guest OS may always create readable mappings to its own page frames, regardless of their current types. A frame may only safely be retasked when its reference count is zero. This mechanism is used to maintain the invariants required for safety; for example, a domain cannot have a writable mapping to any part of a page table as this would require the frame concerned to simultaneously be of types PT and RW.

- The type system is also used to track which frames have already been validated for use in page tables. To this end, guest OSes indicate when a frame is allocated for page-table use . this requires a one-off validation of every entry in the frame by Xen, after which its type is pinned to PD or PT as appropriate, until a subsequent unpin request from the guest OS. This is particularly useful when changing the page table base pointer, as it obviates the need to validate the new page table on every context switch. Note that a frame cannot be retasked until it is both unpinned and its reference count has reduced to zero . this prevents guest OSes from using unpin requests to circumvent the reference-counting mechanism

3.3.3 Virtual address translation

- To minimize the number of hypercalls required, guest OSes can locally queue updates before applying an entire batch with a single hypercall . this is particularly benecial when creating new address spaces. However we must ensure that updates are committed early enough to guarantee correctness.

- Fortunately, a guest OS will typically execute a TLB flush before the first use of a new mapping: this ensures that any cached translation is invalidated. Hence, committing pending updates immediately before a TLB ush usually sufces for correctness. However, some guest OSes elide the ush when it is certain that no stale entry exists in the TLB. In this case it is possible that the rst attempted use of the new mapping will cause a page-not-present fault. Hence the guest OS fault handler must check for outstanding updates; if any are found then they are ushed and the faulting instruction is retried.

3.3.4 Physical memory

- The initial memory allocation, or *reservation*, for each domain is specied at the time of its creation; memory is thus statically partitioned between domains, providing strong isolation.
- A maximum allowable reservation may also be specied: if memory pressure within a domain increases, it may then attempt to claim additional memory pages from Xen, up to this reservation limit.
- Conversely, if a domain wishes to save resources, perhaps to avoid incurring unnecessary costs, it can reduce its memory reservation by releasing memory pages back to Xen.

- **XenoLinux implements a *balloon driver*** , which adjusts a domain's memory usage by passing memory pages back and forth **between Xen and XenoLinux's page allocator.**
- Although we could modify Linux's memory-management routines directly, the balloon driver makes adjustments by using existing OS functions, thus simplifying the Linux porting effort.
- However, paravirtualization can be used to extend the capabilities of the balloon driver; for example, the out-of-memory handling mechanism in the guest OS can be modied to automatically alleviate memory pressure by requesting more memory from Xen.

3.3.4 Physical memory

- Most operating systems assume that memory comprises at most a few large contiguous extents. Because Xen does not guarantee to allocate contiguous regions of memory, guest OSes will typically create for themselves the illusion of contiguous *physical memory*, even though their underlying allocation of *hardware memory* is sparse. Mapping from physical to hardware addresses is entirely the responsibility of the guest OS, which can simply maintain an array indexed by physical page frame number.

- **Xen supports efficient hardware-to-physical mapping** by providing a shared translation array that is directly readable by all domains - updates to this array are validated by Xen to ensure that the OS concerned owns the relevant hardware page frames.

- Note that even if a guest OS chooses to ignore hardware addresses in most cases, it must use the translation tables when accessing its page tables (which necessarily use hardware addresses). Hardware addresses may also be exposed to limited parts of the OS's memory-management system to optimize memory access.

- For example, a guest OS might allocate particular hardware pages so as to optimize placement within a physically indexed cache, or map naturally aligned contiguous portions of hardware memory using superpages.

3.3.5 Network

- Xen provides the abstraction of a virtual rewall-router (VFR),
  - where each domain has one or more **network interfaces (VIFs)**
  - logically attached to the VFR
- A VIF looks somewhat like a modern network interface card:
  - there are two I/O rings of buffer descriptors, one for transmit and one for receive.
  - Each direction also has a list of associated rules of the form
  - (<*pattern*>, <*action*>)
  - If the *pattern* matches then the associated *action* is applied.
- *Domain0* is responsible for inserting and removing rules.
  - In typical cases, rules will be installed
  - to prevent IP source address spoofing,
  - and to ensure correct demultiplexing based on destination IP address and port.
  - Rules may also be associated with hardware interfaces on the VFR.
  - In particular, we may install rules to perform traditional rewalling functions such as preventing incoming connection attempts on insecure ports.

3.3.5 Network

- To transmit a packet,
  - the guest OS simply enqueues a buffer descriptor onto the transmit ring.
  - Xen copies the descriptor and, to ensure safety, then copies the packet header and executes any matching lter rules.
  - The packet payload is not copied since we use scatter-gather DMA; however note that the relevant page frames must be pinned until transmission is complete.
  - To ensure fairness, Xen implements a simple round-robin packet scheduler

- To efciently implement packet reception, we require the guest OS to exchange an unused page frame for each packet it receives; this avoids the need to copy the packet between Xen and the guest OS, although it requires that page-aligned receive buffers be queued at the network interface.
- When a packet is received, Xen immediately checks the set of receive rules to determine the destination VIF, and exchanges the packet buffer for a page frame on the relevant receive ring. If no frame is available, the packet is dropped

3.3.6 Disk

- Only *Domain0* has direct unchecked access to physical (IDE and SCSI) disks.
- All other domains access persistent storage through the abstraction of virtual block devices (VBDs), which are created and congured by management software running within *Domain0*

- Allowing *Domain0* to manage the VBDs keeps the mechanisms within Xen very simple and avoids more intricate solutions such as the UDFs used by the Exokernel.

- A VBD comprises a list of extents with associated ownership and access control information, and is accessed via the I/O ring mechanism.

- A typical guest OS disk scheduling algorithm will reorder requests prior to enqueuing them on the ring in an attempt to reduce response time, and to apply differentiated service (for example, it may choose to aggressively schedule synchronous metadata requests at the expense of speculative readahead requests).
- However, because Xen has more complete knowledge of the actual disk layout, we also support reordering within Xen, and so responses may be returned out of order.
- A VBD thus appears to the guest OS somewhat like a SCSI disk

3.3.6 Disk

- A translation table is maintained within the hypervisor for each VBD; the entries within this table are installed and managed by *Domain0* via a privileged control interface.

- On receiving a disk request, Xen inspects the VBD identier and offset and produces the corresponding sector address and physical device. Permission checks also take place at this time. Zero-copy data transfer takes place using DMA between the disk and pinned memory pages in the requesting domain.

- Xen services *batches* of requests from competing domains in a simple round-robin fashion; these are then passed to a standard elevator scheduler before reaching the disk hardware. Domains may explicitly pass down *reorder barriers* to prevent reordering when this is necessary to maintain higher level semantics (e.g. when using a write-ahead log).

- The low-level scheduling gives us good throughput, while the batching of requests provides reasonably fair access. Future work will investigate providing more predictable isolation and differentiated service, perhaps using existing techniques and schedulers.

3.4 Building a New Domain

- The **task of building the initial guest OS structures**
  - for a new domain is mostly delegated to *Domain0*
  - which uses its privileged control interfaces
  - to access the new domain's memory
  - and inform Xen of initial register state

- This approach has a number of advantages
- compared with building a domain entirely within Xen,
- including reduced hypervisor complexity
- and improved robustness (accesses to the privileged interface are sanity checked which allowed us to catch many bugs during initial development)

3.4 Building a New Domain

- Most important, however, is the ease with which the building process can be extended and specialized to cope with new guest OSes.
- For example, the boot-time address space assumed by the Linux kernel is considerably simpler than that expected by Windows XP.

- It would be possible to specify a xed initial memory layout
  - for all guest OSes,
  - but this would require additional bootstrap code
  - within every guest OS to lay things out as required by the rest of the OS.
- Unfortunately this type of code is tricky to implement correctly;
  - for simplicity and robustness it is therefore better to implement it within *Domain0*
  - which can provide much richer diagnostics and debugging support than a bootstrap environment

Performances

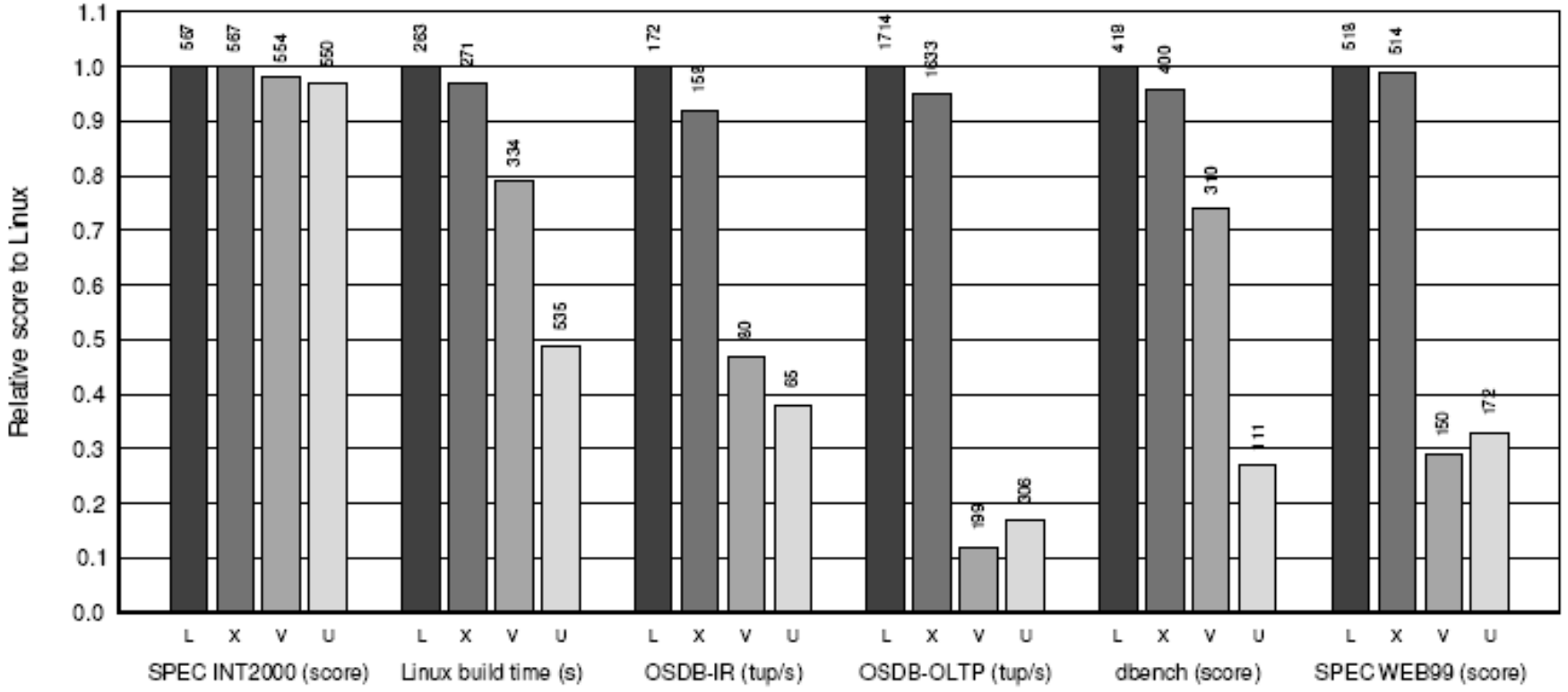


Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

Xen

Hypervisor

- U računarstvu hypervisor, ili monitor virtuelnih mašina, je jedan od načina konkretne realizacije virtuelne mašina, za koji se najčešće vezuje pojam virtualizacije.

- Iako ne možemo reći da svaka virtuelizacija ima hypervisor (na primer Sun JAVA), većina kategorija virtuelizacije može se realizovati isključivo preko nekog vida hypervisora.
- Hypervisor je softver, koji simulira virtuelne mašine gostujućim operativnim sistemima.

- Termin hypervisor nastaje iz toga što se operativni sistemi obraćaju hardveru preko hiper poziva (hypervisor call, tj. hypercall), odnosno komunicira inderektno sa hardverom preko nekog softvera koji ga imitira, a koji posle prosleđuje dalje prema hardveru.

Hypervisor

- Hypervisori se trenutno svrstavaju u dva tipa:

- **Tip 1, odnosno prirodni**, native, bare-metal, je hypervisor koji radi direktno na hardveru, ispod operativnog/operativnih sistema, kao kontorlni program operativnog sitema. Gostujući operativni sitem se onda pokreće na drugom nivou iznad hardvera. To je bila i prva realizacija virtelne mašine u opšte, od strane IBM-a 1960. godine. Drugi primeri te realizacije su Xen, Citrix XenServer, OrecleVM, VMware-ov ESX Server, L4 microkernels , Microsoft-ov Hyper-V, Sun's Logical Domains Hypervisor. Takođe tu su i varijacije. Na primer KVM koji pretvara ceo linuxov kernel u hypervisor.

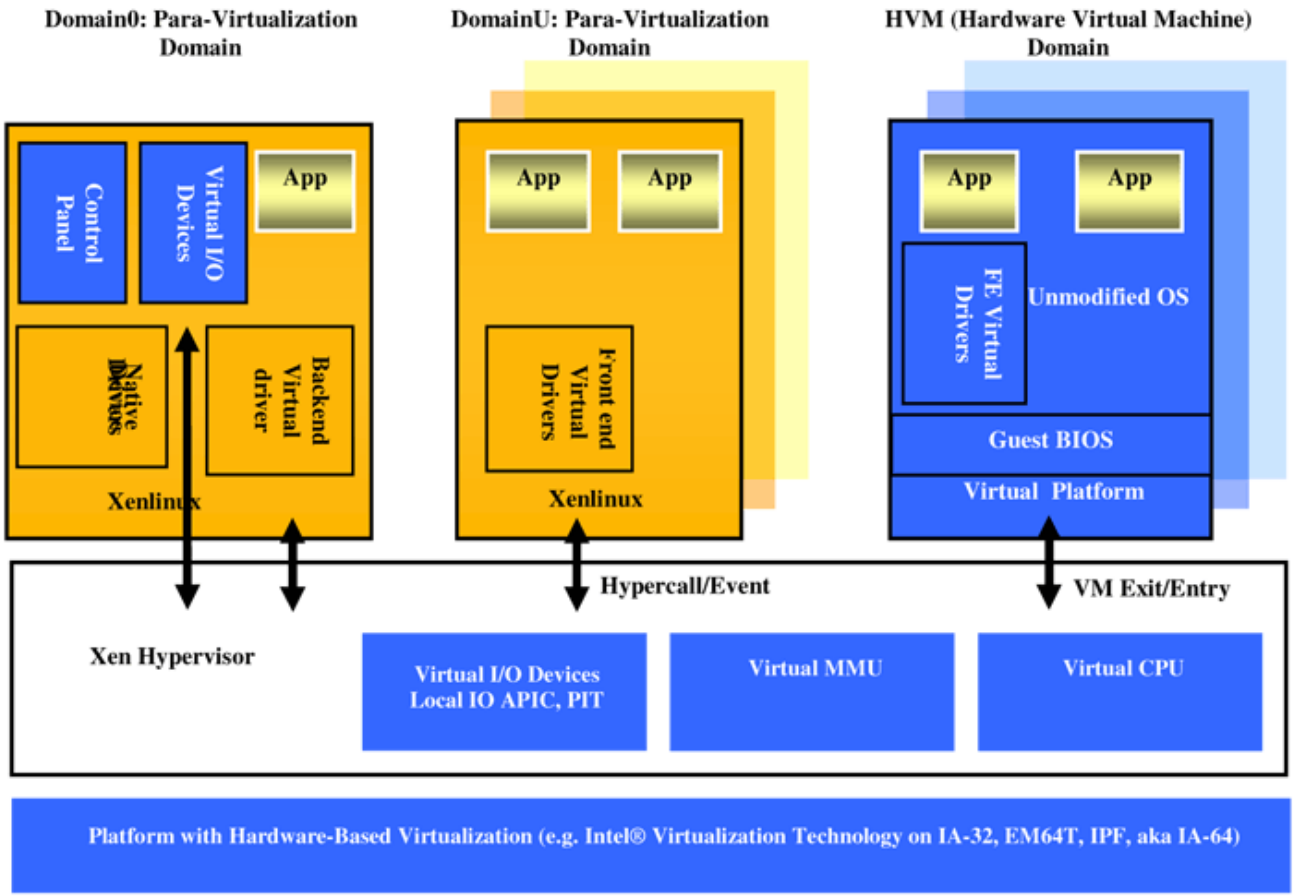
- **Tip 2, ili gostujući**, hosted, hypervisor je softver koji se pokreće unutar operativnog sistema, otprilike kao i svaka druga aplikacija. Gostujući operativni sitemi se onda instaliraju na tek trećem nivou iznad harvera, posle operativnog istema i samog hypervisora. Neki od hypervisora drugog tipa su VMware Server (nekada GSX), VMware Workstation, VMware Fusion, QEMU, Microsoft's Virtual PC, Microsoft Virtual Server products, VirtualBox.

2. Xen

- **Xen je hypervisor i tehnologija slobodnog softvera, razvijena u saradnji sa Xen zajednicom i inženjerima na preko 20 najmoćnijih informatičkih kuća**, među kojima su AMD, Cisco, Dell, HP, IBM, Intel, Mellanox, Network Appliance, Novell, Red Hat, SGI, Sun, Unisys, Veritas, Voltaire i Citrix. Iako je razvijan sa idejom rada i virtuelizacije i386 athitekture, koja nikada nije bila ugodna za virtuelizaciju, sada radi na nekoliko arihtektura x86-64, IA-64 i PowerPC 970. Takođe to je i dovelo do neke hardverske implementacije tehnologija virtuelizacije od strane ključnih proizvođača procesora, Intela i AMD-a. Veoma je zanimljivo da ceo kod hypervisora projekta nije veći od 50 000 linija.
- Xen-ovo poreklo vodi od Kembridž univerziteta, i razvoj je vođen od strane Ian Pratt-a, starijeg predavača na kembridžu i osnivača XenSource Inc. Ova kompanija sada podržava razvoj open source dela softvera i takođe prodaje enterprise verziju. Xen je još uvek relativno mlad, a prvo izdanje potiče iz 2003 godine. XenSource Inc je preuzela kompanija Cytrix oktobra 2007 godine. Cytrix sada izdaje više komercijalno licenciranih verzija xena, ali se u nastavaku text usresređuje na ne komercijalnu granu, mada skoro sve važi i za komercijalnu.
- Kada bi trebalo svrstati virtuelizaciju xena u neku od kategorija virtuelizacije, u užem smislu, to bi bila **paravirtualizacija.** Međutim stvari nisu tako jednostavne. Projekat je takođe u sebe objedinio i još jedan projekat slobodnog softvera vezanog za **virtuelizaciju QEMU**, i tako zalazi i u kategoriju hardverski zasnovane virtuelizacije objedinjujući sve u isti paket. **Tako da imamo konacno dve grane Xen-a, jedna je paravirutelizacija, a druga je hardverski zasnovana**.

Arhitektura

- Xen bi se mogao prestaviti kroz **par ključnih komponenata**, a to je pre svega **hypervisor sa gostujućim operativnim sistemima**. Specifičan je **prvi operativni sitem koji je više deo samog hypervisora nego što je u pravom smislu reči gostujući operativni sistem**. Kod preostalih "pravih" gostujućih sitema, pravi se podela na dva tipa, jedan je **paravirtuelni gostujući operativni sistem, a drugi je hardveski podržan gostujući operativni sistem**



2.2 Jezgro i prvi gostujući operativni sitem - domain 0

- **Jezgro**
- Xen (od verzije 3.0) ima malo jezgo (kernel, već spomenuto ima manje od 50 000 linija koda) hypervisora. Xen-ov hypervisor je u osnovi samo apstrakcija sloja softvera koja se nalazi na hardveru ispod bilo kog operativnog sistema.
- Znači ne imitra hardver, već imitira softverski deo samog operativnog sistema koji je u direktnoj vezi sa hardverom.
- Hypervisor je odgovoran za raspoređivanje procesorskog vremena i particionisanje memorije za virtuelne mašine koje na njemu rade.
- On je zadužen za apstrakciju hardvera za sve virtuelne mašine i kontroliše njihov pristup zajedničkom procesoru, ali on nije svestan mreže, eksterne memorije, grafičkoga adaptera, ili bilo kog drugog ulazno izlaznog uređaja. Ta uloga je prepuštena prvom gostujućem domenu.

2.2 Jezgro i prvi gostujući operativni sitem - domain 0

- Prvi gostujući sistem, se u Xen-ovoj terminologiji naziva "**domain 0" (dom0)** i poseduje velike specifičnosti u odnosu na ostale gostujuće sisteme koje se u Xen terminologiji nazivaju "**domain U" (domU).**

- **Dom 0 je više integralni deo Xen-a nego što je u pravom smislu reči gostujući operativni sistem** i on je takođe **relizovan jednim vidom paravirtuelizacije.**
- **Jednim vidom jer se takođe ponaša i kao samostalan operativni sistem i kao deo hypervisora**.
- **Dom 0 se boot-uje automatski, odmah pošto se boot-uje Xen hypervisor**, a bez njega nisu moguće nikakve druge radnje na drugim gostujućim operativnim sistemima.
- **Takođe dom 0 poseduje specijalne privilegije**, koje mu omogućavaju direktan kontakt sa hardverom, kao što je **pristup svim ulazno-izlaznim uređajima.**
- On pristup **hardveru obezbeđuje preko pravih, nativnih, drajvera operativnog** sitema na kome je zasnovan, i preko pozadniskih virtuelnih drajvera (backend Paravirtual driver, backend virtual driver).
- Pozadinski drajveri komuniciraju direktno sa hardverom, i uključuju sve zahteve koji dolaze sa gostujućih domain-a U

2.2 Jezgro i prvi gostujući operativni sitem - domain 0

- *Xen 3.0 uključuje kontrolni interfejs* koji konfiguriše deljenje procesora, memorije, mreže i drugih uređaja, a pristup tom interfejsu je jedino moguć preko operativnog sistema instaliranog kao dom 0.

- Sistem administrator se, posle bootovanja, dalje uloguje u dom 0, da bi mogao da instalira bilo koji drugi gostujući operativni sistem dom U.

- Takođe je zadužen i za administrativne zadatke, kao što su pokretanje, gašenje, suspendovanje i migracija gostujućih domena na druge virtuelne mašine.

- Modifikovane verzije jezgra GNU/Linuxa, BSD-ija i Solarisa mogu biti korišćena kao dom 0.

2.3 Višesturki korisnički domeni

- Višestruki korisnički domeni (Multiple User domains), odakle potiče U u nazivu domain U, mogu biti kreirani kao gostujući operativni sistemi.
- Tu razlikujemo dva tipa korisničkih domena, a to su:
  - Paravirtuelizovan - Dom U PV (paravirtualized)
  - Hardverski zasnovan -DomU HVM (hardware-based virtual machines)

2.4 Domen U PV

- Domeni realizovani preko paravirtuelizacije su srž Xen projekta. U računarstvu paravirtualizacija je tehnika virtuelizacije koja predstavlja softveru interfejs ka viruelnoj mašini koji je voma sličan ali ne i identičan hardveru na kome rade. **Paravirtuelizacija zato omogućava monitoru virtuelnih mašina da postigne performanse približne** ne virtuelnoj masini, jer je moguće napraviti hypervisor koji nije previše komplikovan i težak. Taj pristup virtuelizaciji je dao Xen-u minimalan, brz i pouzdan hypervisor, koji je u stvari sloj idealizovanog hardvera, na kome i sam radi. Xen-ovo jezgro hypervisor-a u sebi ne sadrži ni drajvere za rad sa hardverom na kome je. Drajveri su sadržani u dom 0, i koriste se ne modifikovani drajveri samog operativnog sistema na kome radi dom 0, za pristup hardveru. Drajveri su izvan i ispod jezgra hypervisora. Druga rešenja virtuelnih mašina, su često sama mikro kernel, koji pak u sebi poseduje sopstvene drajvere za pristup hardveru, što predstavlja dodati problem i komplikuje razvoj.

- To što Xen predstavlja idealizaciju hardvera, znači i da ga ne oslikava potpunosti. To dovodi do toga da je **neophodno izmeniti i prilagoditi jezgro operativnog sistema** koji želimo da instaliramo kao dom U PV. Glavna prednost tog pristupa je upravo brzina rada oeprativnog sistema gostujućih virtuelnih mašina. Naime Xen nudi performanse za virtuelne sisteme koje su jako blizu performansi ne virtuelnih. Overhead virtuelizacije, se uglavnom kreće od 0.1 do 3.5 %, na standardnim industriskim benčmarcima, retko kad se približavajući 5%. U odnosu na standardni pristup virtuelnim mašinama koje imaju overhead i više od 35%.

- **Dom U PV gostujući operativni sistemi su svesni da nemaju direktan pristup hardveru koji se nalazi ispod njih, i prepoznaju da druge virtuelne mašine rade pored njih, na istoj fizičkoj mašini**. Oni poseduju **virtuelne drajvere** (virtual driver, frontend virtual driver) za pristup fizičkim uređajima poput mrežne karte ili diska, koji se obraćaju pozadinskom (backend delu) virtuelnih drajvera na domain 0- mašini.

2.4 Domen U PV

- Iako je kernel promenjen, veoma je bitno da, **sav drugi softver, poput korsničkih aplikacija, biblioteka i slično, je moguće koristiti u potpuno ne promenjenom binarnom obliku.**
- **Glavni problem ipak ostaje potreba da se promeni jezgro operativnog sistema** kako bi mogao da se instalira kao paravirtuelni gost. Za sada je kao Domain U PV moguće koristiti sledeće operativne sisteme: **Linux, BSD, Solaris, GNU/Hurd/Mach, Minix, Plan 9, NetWare i OZONE.**

- Pored izvesnog truda koji treba uložiti u samu modifikaciju potrebno je imati pristup i dozvolu za modifikaciju samog koda kernela operativnog sitema koji bi želeli da preradimo, što često nije moguće. Zato su uglavnom portovana jezgra pod slobodnim licencama, ili drugim sličnim tipom open source licenci. Glavni problem su komercijalni operativni sistemi. Kod njih, uglavnom, nije moguć ni uvid u sam kod, a i da jeste licenca ne dozvoljava promenu. Sami proizvođači komercijalnih operativnih sistema, iako bi mogli sami, često ne žele, ili ne umeju da modifikuju jezgro za primenu pod Xen-ovom paravirtuelizacijom. **Dobar primer je Microsoft, čiji proizvod Windows XP, još za vreme razvoja XEN 1.x, a u saradnji Kembridž univerziteta sa Microsoft Research odeljkom, uspešno portovan na Xen platformu. To je bilo moguće zahvaljujući Microsoft's Academic Licensing Program**. Uslovi licence, na žalost, nisu dozvoljavali da se portovani Windows XP objavi. Postoje neke naznake da bi nova serija Windows operativnog sistema (Vista i Winodws 2008) mogla biti prilagođena radu. Dosta je izvesno da bi neka od verzija Windows servera 2008 mogla biti implementirana kao Xen hypervisor, dok rad kao gostujući operativni sistem, u trenutku pisanja nije zvanično najavljen, najverovatnije zbog straha da ne favorizuju konkurentske platforme. Iz **sličnog razloga Microsoft je najavio sopstvenu paravirtuelizaciju, koja dosta podseća na Xen**.

2.5 Domen U HVM

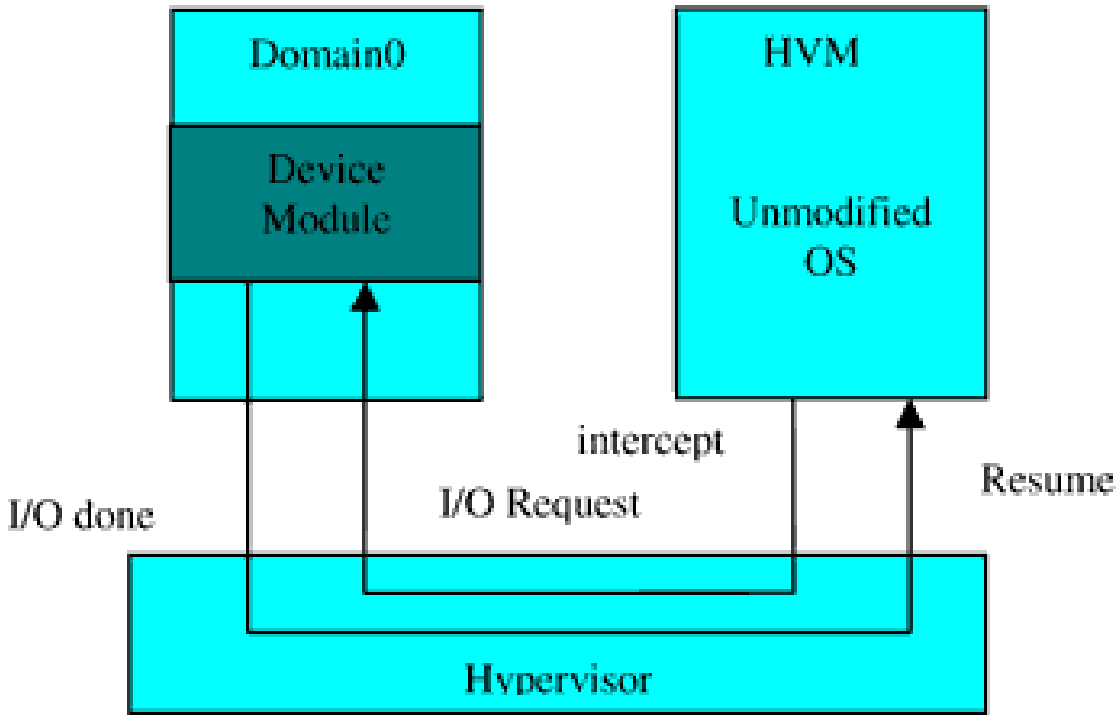
- Zbog ograničenja paravirtuelizacije, u pogledu raznovrsnosti operatvnih sistema koji mogu da se instaliraju, a da bi Xen bio sveobuhvatno rešenje, morao se naći način da se svi, pa i ne modifikovani operativni sistemi mogu izvršavati na virtuelnoj mašini, zajedno i sa paravirtuelnim. Da ne bi pisali jos jedno virtuelizaciono rešenje, kreatori xena su iskoristili mogućnosti slobodne licence pod kojom rade, i u Xen integrisali drugi open source projekat, zarad postizanja tog cilja. Reč je o projektu QEMU.
- **QEMU je emulator centralnog procesora računara, kome su pridodate emulacije hardverskih uređaja, zahvaljujući čemu je moguće pokretati ne modifikaovene operativne sisteme**. Glavna mana mu je, za sada, **veliki overhead**, zbog čega se razvijaju razni načini akceleracije. Uglavnom se može posmatrati kao gostujući monitor virtuelnih mašina. To je veliki projekat sa mnogo grana, sa različitim primenama u raznim drugim projektima. QEMU, integrisan u Xen ima naziv Xen-HVM (hardware-basede virtual machine, odnosno, hardverski-zasnovana virtuelna mašina, ali se često skraćuje kao hardverska virtuelizacija).
- Xen-HVM virtuelne mašine bi spadale u red hardverski zasnovanih virtuelnih mašina, jer su za **njihovo korišćenje potrebne hardverske virtuelne ekstenzije kod centranog procesora. To su, za sada, kod Intela VT-x, a kod AMD-a AMD-V ekstenzije**. Ove ekstenzije se, iako se veoma razlikuju u svojoj implementaciji i setu instrukcija, kontorolišu pomoću uobičajenog apstraktnog sloja interfejsa. Hardverski potpomognuta (zasnovana) virtuelizacija nudi novi set procesorskih instrukcija koji podržava direktno obraćanje (hiperpozive), drajvera gostujićih oprativnih sistema hardveru, što poboljšava performanse. **QEMU se u Xen-ovoj verziji pokreće kao daemon (pozadinski proces) u domain 0 operativnom sistemu**.

2.5 Domen U HVM

- **Xen-HVM gostujući operativni sistemi nisu svesni da dele resurse sa drugim virtuelnim mašinama, pa im se zato mora predstaviti celokupan lažni hardver**. To uključuje sve, od **lažnog BIOS-a** (realizovanog preko ekstenzija procesora), do virtuelizacije svakog hardvera koji želimo, ili koji je neophodan (**emulacija grafičke karte** je neophodna za neke operativne sisteme, jer je i prisustvo fizičke grafičke karte neophodno u računaru prilikom instalcije istih na ne virtuelnu mašinu).
- **Xen-HVM uključuje virtuelizaciju sledećeg hardvera**: PIIX3 IDE (sa nekim osnovnim PIIX4 mogućnostima), Cirrus Logic, ili vanilla VGA emulated video, RTL8139 or NE2000 mrežnu emulaciju, PAE i veoma ograničenu ACPI i APIC podršku, ali SCSI podrške. Hardver je emuliran preko patched QEMU menadžera uređaja. Tu leži jedan od najvećih problema Xen HVM-a, a to je ne mogućnost korišćenja hardverske akceleracije kod gostujućih operativnih sistema, što se posebno odražava na multimediju, jer bi se, na primer, puštanje zvuka kompletno softverski emuliralo na fizički hardver, što se negativno odražava na performanse i rad procesora. Domain U HVM nema nikakve virtuelne drajvere, on koristi svoje nativne drajvere za obraćanje virtuelnom hardveru koji kontroliše hypervisor.

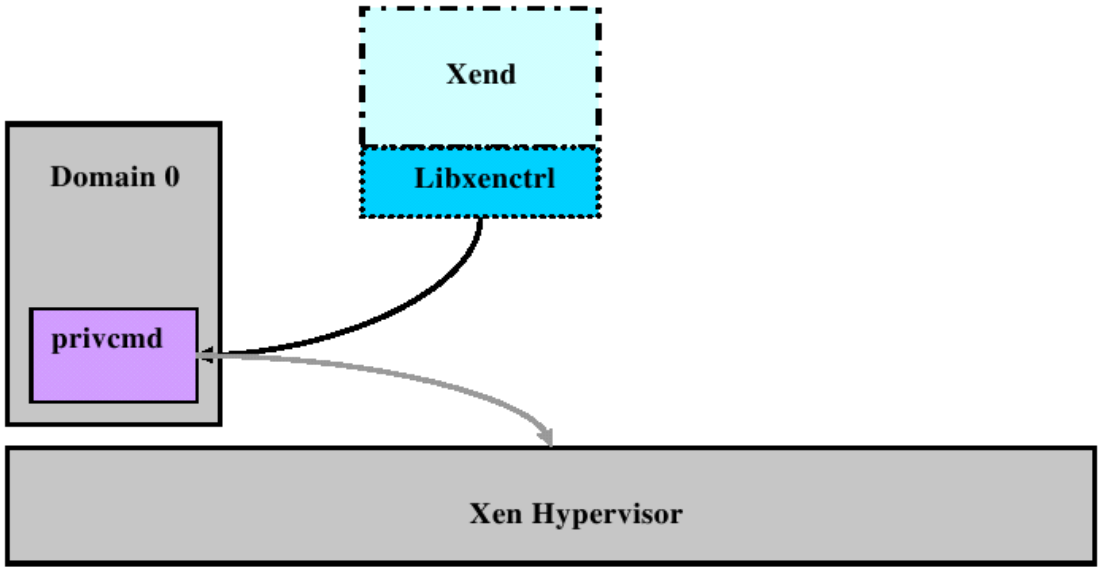
- Pored svih paravirtuelnih operatvnih sistema, koji mogu biti pokrenuti i preko Xen HVM-a, sada pruža zvaničnu podršku i **za razne verzije Microsoft-ovih operativnih sistema.**

2.5 Domen u HVM



2.6 Kontrola i menadžment domena

- Način upravljanja domena se ne razlikuje znatno u zavisnosti od operatvnog sitema Domain 0, pa je u nastavku opis za Domain 0 na GNU/Linux operativnom sistemu. U **GNU/Linuxu su implementirani preko serije daemon-a i pokreću se, naravno, unutar Domain 0.**
- Napomena: dameon-i su prokazani van Domain 0 radi jasnoće



2.6 Kontrola i menadžment domena

- **Xend**
- Xend je daemon, koji je napisan kao aplikacija na jeziku python i smatra se menadžerom sistema za Xen okruženje. On opslužuje libxenctrl biblioteku i prosleđuje zahteve Xen hypervisor-u. Svi zahtevi koji opslužuje Xend, se prenose preko XML RPC interfejsa pomoću Xm alata.

- **Xm**
- Je alat komandne linije koji uzima korisnički ulaz i prosleđuje do XenD-a preko XML RPC-a

- **Xenstored Virtual Machine Manager**
- Xenstored je daemon koji održava registar o informacijama, kao što su memoriski kanali i kanali događaja, između Domain 0 i svih Domain U gostiju.

- **Libxenctrl**
- Libxenctrl je C biblioteka koja **omogućava Xend komunikaciju sa Xen hypervisor-em, sa Domain-a 0**. Specijalni drajver u Domain-u 0, koji se zove **pivcmd** šalje zahteve hypervisoru.

2.6 Kontrola i menadžment domena

- **Qemu-dm**
- Svaki HVM Gost koji se pokrene u Xen okruženju mora da ima sopstveni QEMU daemon. Takođe se razvija Stub-dm koji će u budućnosti poboljšati korišćenje QEMU-a i ukloniti potrebu da svaki HVM gost mora da pokrene zaseban daemon.

- **Xen upravljačke konzole**
- Xm je kao osnovni konzolni alat dosta sirov, i ne baš prevše jednostavan ni za učenje ni za svakodnevnu primenu. Iz tog razloga postoje mnogi alati, koji su razvijani uglavnom od samih korisnika, kao što su linux distribucije i firme. Te alatke su najčešće integrisane sa Linux distribucijom, i daju mogućnost olakšane administracije Xen Domena, kao što su instalacija, konfiguracija, menadžment resurasa, pokretanje i zaustavljanje virtuelnih mašina. Tu bi pre svega pomenuli sledeće: Python zasnovan Enomalism dashboard (LGPL), Xen Tools, Googl-ov Ganeti, Perl zasnovan MLN, web bazirani HyperVM i FluidVM, i grafičke aplikacije poput GUI ConVirt (nekada poznati kao XenMan) i Red Hat-ov Virtual Machine Manager, virt-manager.

3 UPOTREBA

- **Xen, kao i druge virtuelne mašine imaju široku primenu u IBM-ovim, HP-ovim i drugim ponuđačima mainframe i velikih servera.** U ogromnoj upotrebi je u Internet Hosting kompanijama koje pružaju usluge virtuelnih Dedicated servera. Glavna korist od Xen, kao i od drugih virtuelizacija čine usaglašavanje, povećanje upotrebe i iskorišćenosti hardvera, mogućnost pojednostavljene administracije i boljeg reagovanja na ne predviđene situacije.
- **Xen omogućava prenošenje virtuelne mašine na drugi hardver, bez praktičnog downtime-a**, što je često od izuzetne važnosti. Xen ima mogućnost migracije u živo (live migration) dva virtuelna operativna sistema između dve fizičke mašine koje su međusobno povezane LAN mrežom bez gubitka raspoloživosti. Tokom ove procedure, memorija virtuelne mašine se iterativno kopira na destinaciju skoro bez prekida u izvršenju. Skoro jer je potrebno nekih 60-300 milisekundi da bi se izvršila konačna sinhronizacija pre nego što virtuelna mašina počne izvršavanje na njenom konačnom odredištu migracije. Slična tehnologija se koristi i za live backup, trenutnog rada mašine na Hard disk, i mogućnost pokretanje prvobitne virtuelne mašine kasnije na nekoj drugoj virtuelnoj mašini.
- Tu je i mogućnost da se dodaju mnogi operativni sistemi koji bi paralelno radili na istom računaru. Gubi se veći deo potrebe za dual boot-operativnim sistemom, jer je sada moguće, npr. Windows pokreniuti unutar prozora u Linuxu.
- **Tu je naravno i ogromna primena u razvoju i testiranju softvera**.

- Primer instalacije u grafičkom režimu
- Xen server
- Xen center

Citrix Systems Xen

- Xen je potpuno besplatni VMM (engl. *Virtual Machine Monitor*) Dostupan je veći broj alata za upravljanje Xen sistemom preko korisničkog interfejsa (engl. *User Interface*), a među njima su:
  - *XenExpress* - Najjednostavnija varijanta, koja podržava pokretanje do četiri VM (virtuelne mašine) i potpuno je besplatna.
  - *XenServer* - Nešto naprednija verzija. Na njoj je moguće pokrenuti do osam VM.
  - *XenEnterprise,* robustan softver namenjen najzahtevnijim korisnicima. Uz godišnju pretlatu za dva snažnija rešenja, *XenSource* obezbeđuje i tehničku podršku za rešavanje svih zahteva korisnika Xen virtuelizacije. *XenEnterprise* omogućava kreiranje neograničenog broja virtuelnih mašina. Jedino realno ograničenje su hardverski kapaciteti računara na kojem se ovaj sistem pokreće.
  - *XenTools* - Perl alati za Debian GNU/Linux,
  - *Ganeti* - Orijentisani na upravljanje grozdovima računara i paralelizaciju,
  - ***HyperVM*** web orijentisani, polu-komericjalni (engl. *proprietary*) alat namenjen Linux sistemima.

XEN

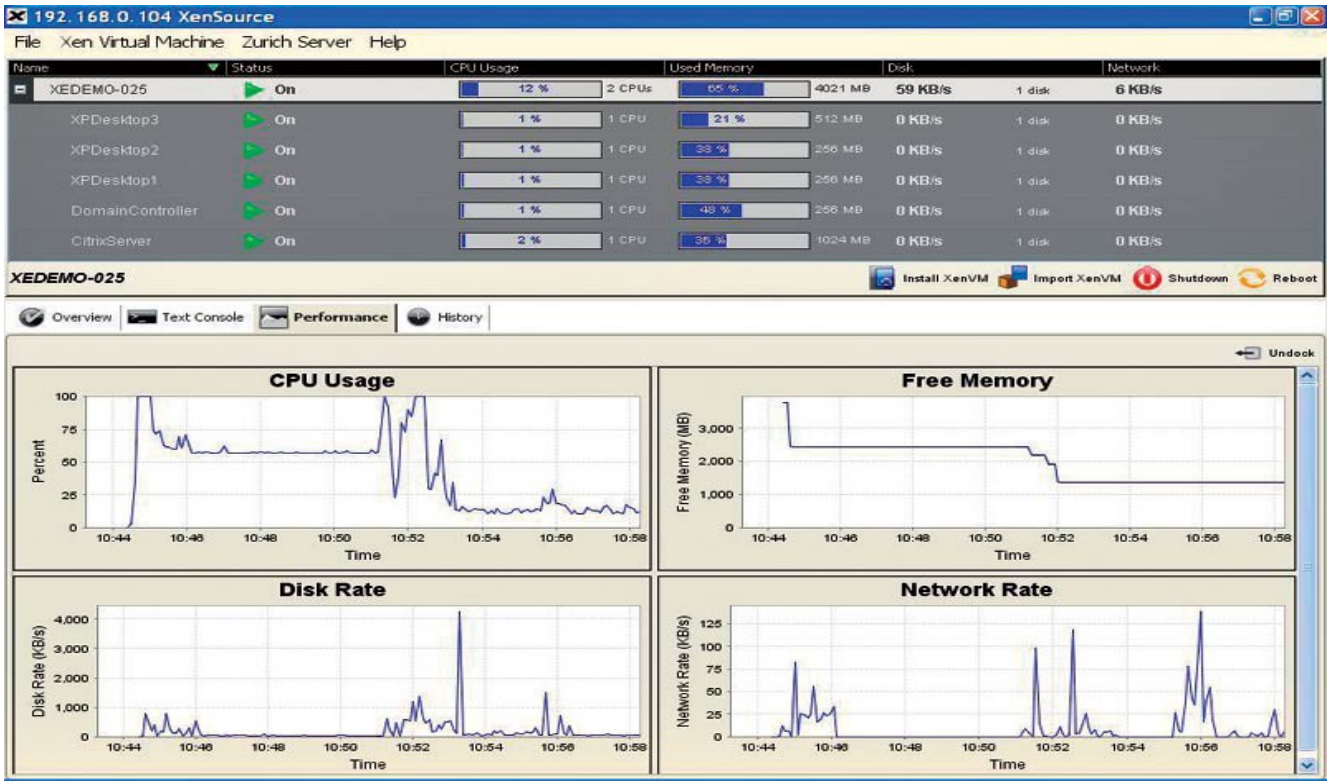
- XEN je tehnologija virtuelizacije nastala kao istraživački projekat na Cambridge univerzitetu 2003. godine, koji je vodio Jan Prat (Ian Pratt). U međuvremenu, on je osnovao kompaniju XenSource koju je kupila kompanija Citrix, tako da se sad trenutno Xen održava i razvija od strane više kompanija udruženim pod imenom Xen AB (Advisory Board) a čine ga kompanije Citrix, IBM, Intel, Novell, Red Hat, Oracle, kao i developeri širom sveta.

- 2010. godine je Xen publikovan kao open-source arhitektura pod GNU GPL2 licencom (General Public license).

- Citrix je u proleće 2010. godine objavio 4 proizvoda baziranim na Xen-u:
  - **XenServer (free edition)**
  - **XenServer (enterprise, advanced i platinum)**

- Kao što i ime govori, prva verzija je bazirana na open-source projektu i besplatna je, dok su ostale 3 komercijalne

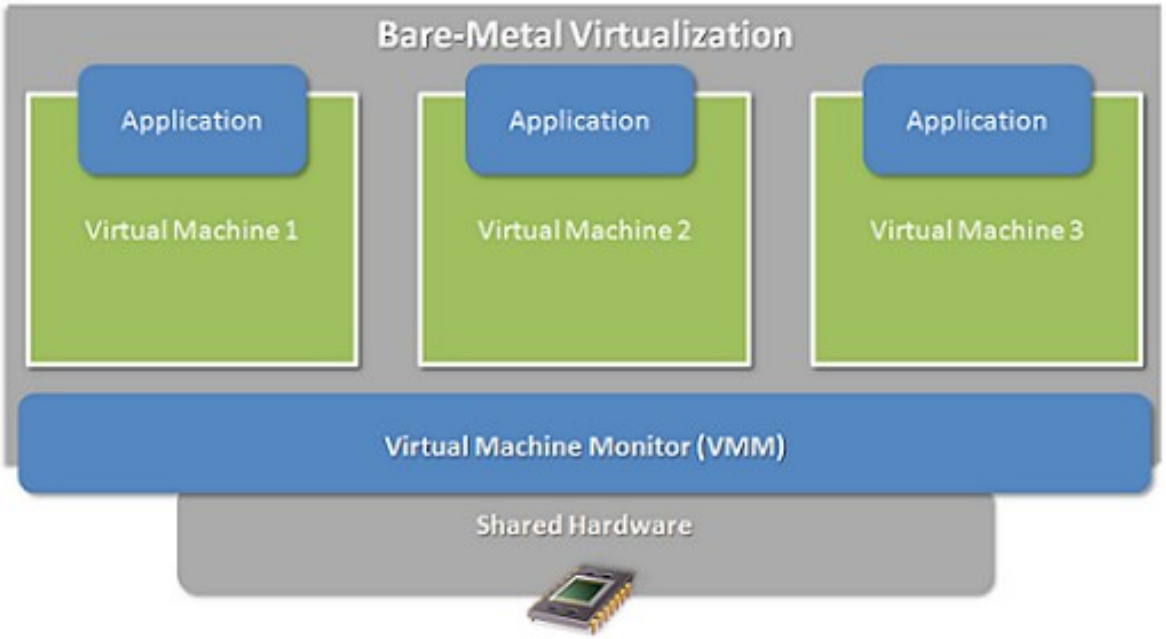
7.2.2. Konsolidovanje infrastrukture



- *Slika 7.3. Iskorišćenost kapaciteta hardvera i performanse virtuelnih mašina*
- Xen rešenja se veoma brzo razvijaju i unapređuju. O tome svedoči i činjenica da u početku nije bilo moguće pokrenuti Windows virtuelne mašine, a danas to funkcioniše. To je garancija da će nove tehnologije koje hardver ponudi vrlo brzo dobijati podršku u Xen rešenjima.

7.2.3. XenServer

- ***XenServer*** je komercijalna implementacija Xen-ovog open source hipervizora, američke softverske korporacije **Citrix Systems** i s obzirom da je open source, može se preuzeti i koristiti besplatno. XenServer je *bare-metal* hipervizor, odnosno ne zahteva matični operativni sistem i instalira se direktno na hardver. Preciznije, kada se **XenServer instalira na čist računar**, on postaje matični računar gde će se skladištiti i upravljati virtuelnim mašinama.

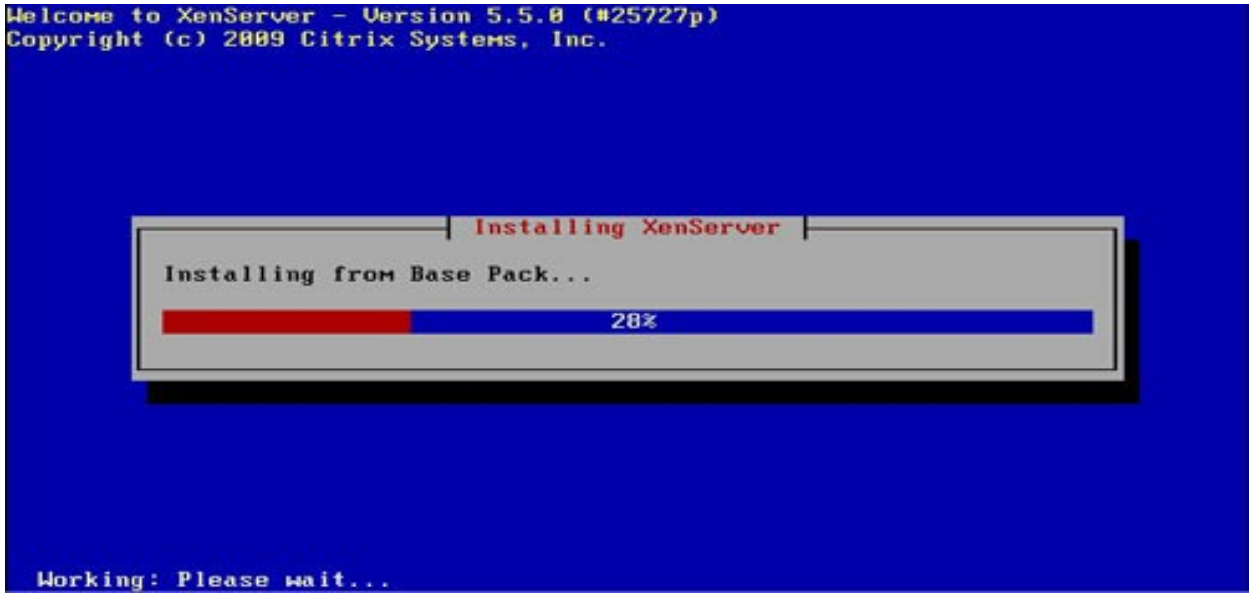


- *Slika 7.4. Bare-metal hipervizor*

Xen

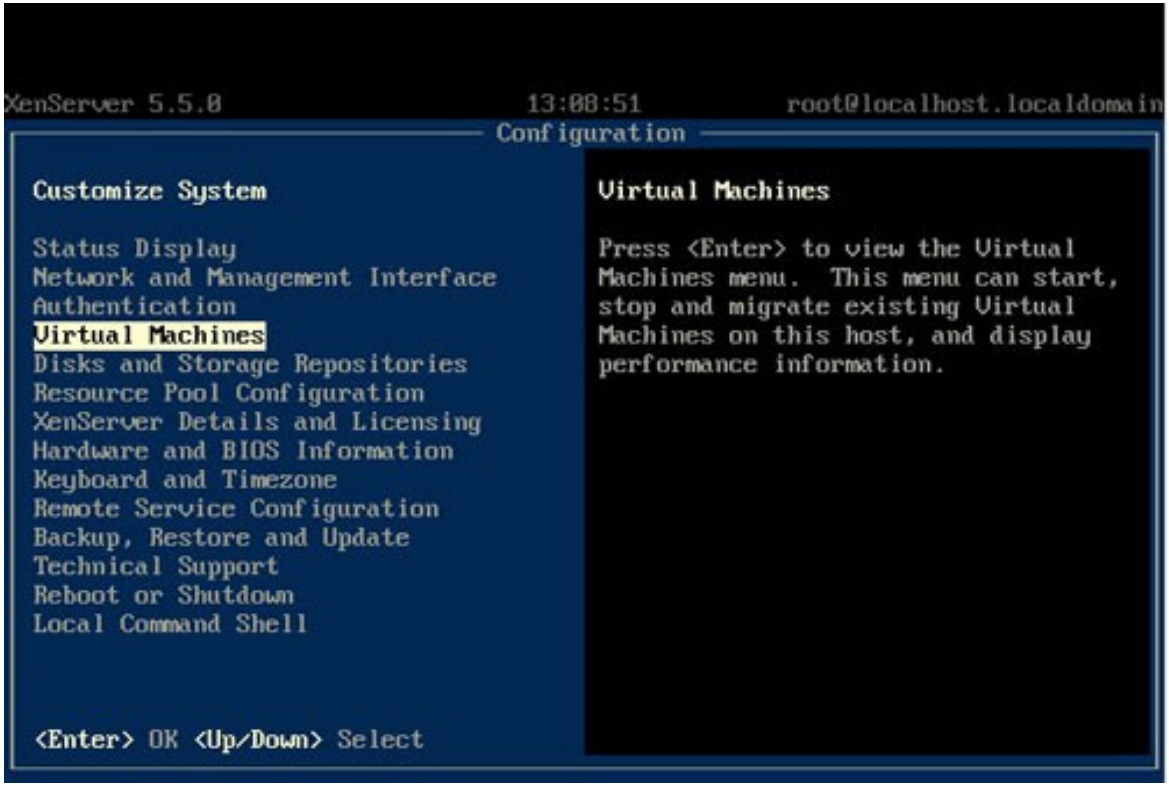
7.2.3. XenServer

- Instalacija je relativno jednostavna, tj. posle butovanja ***XenServer* će prvo od kosrisnika tražiti da izabere način instalacije:**
- **1. kao matični server** ili
- 2. da postojeći operativni sistem pretvori u virtuelnu mašinu.
- Zatim će se pojaviti dijalozi za postavljanje administratorske lozinke, setovanje mrežnog interfejsa (mrežne adrese servera), naziv servera, podešavanje vremena i vremenske zone servera. Posle ovoga, instalacija počinje.



- *Slika 7.5. Proces instalacije XenServer-a*

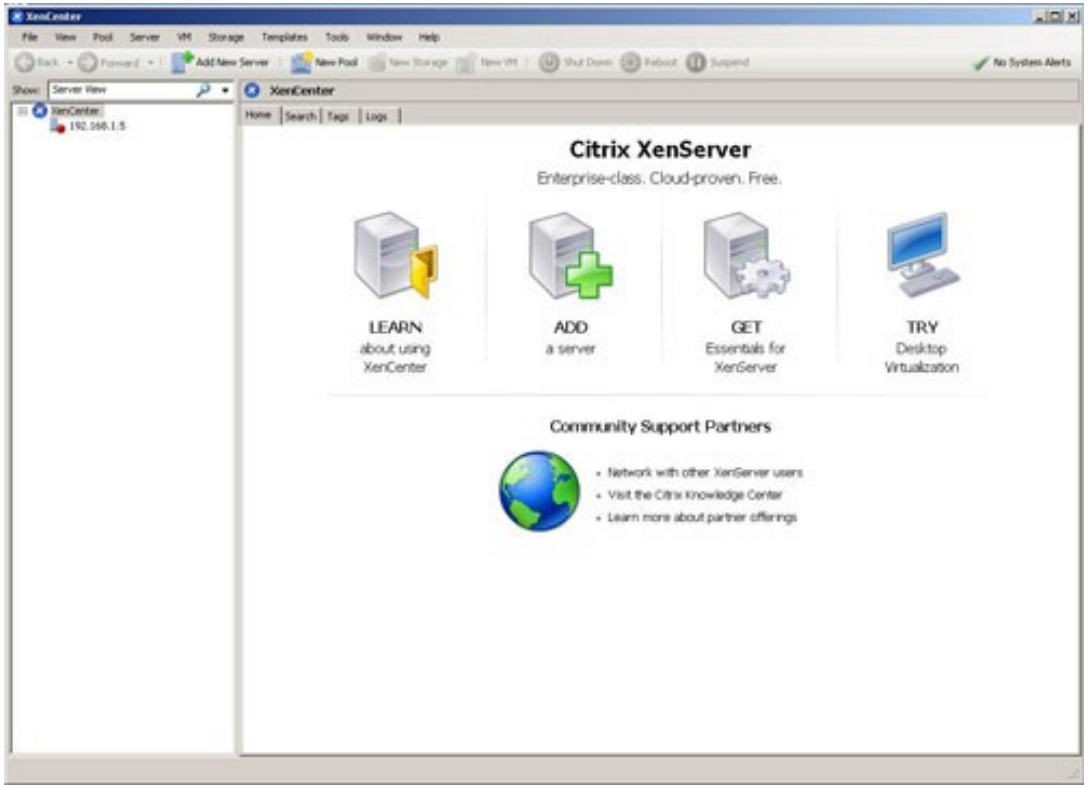
7.2.3. XenServer

- Sam proces instalacije ne traje dugo i ubrzo će se računar restartovati i butovati *XenServer*.
- 

- *Slika 7.6. XenServer konfigurisanje*

7.2.3. XenServer

- U okviru samog ***XenServera*****,** moguće je **administrirati sam server** (tu spada mrežna konfiguracija, *backup*, *restore*, *update* podataka na serveru i sl.), kreirati hard diskove, tzv. "pool-ove" resursa računara, pratiti stanje virtuelnih mašina koje su na serveru, ali nije moguće i samo kreiranje virtuelnih mašina. Za to se koristi druga fizička mašina sa već postojećim operativnim sistemom (Windows, Linux...) na koju će se instalirati *XenCenter* (nalazi se na istom disku, gde se nalazi i sama instalacija *XenServera*).



- *Slika 7.7. XenCenter*

7.2.3. XenServer

- Posle instalacije *XenCenter*-a, da bi se ostvarila komunikacija sa *XenServer*-om, neophodno je poslati informaciju *XenCenter*-u na kojoj adresi se *XenServer* nalazi, kako bi mogao da ugosti virtuelne mašine. To se postiže unosom ranije definisane IP adrese *XenServera*.

- Nakon dodavanja servera, može se preći na kreiranje virtuelnih mašina, koje je, kao i svugde do sad, relativno jednostavno i sastoji se od svega nekoliko (već poznatih) koraka.

- Preciznije, sastoji se od: odabira operativnog sistema koji se instalira, imena virtuelne mašine, odabira instalacionog medija, određivanja koliko će RAM memorije i procesora/jezgara koristiti, kreiranja/dodavanja hard diska i mrežnih uređaja. Posle ovih izbora, instalacija gostujućeg operativnog sistema se automatski nastavlja. Kada se se instalacija završi, neophodno je instalirati *XenTools* radi poboljšanja peformansi virtuelnih mašina i instalacije alata za migraciju virtuelnih mašina.

Primer instalacije Xen-a

Postupak instalacije:

- 1. Instalacija potrebnih paketa;
- 2. Instalacija Xen-a 3.0.4 ;
- 3. Rekompjaliranje kernela Xen-a ;
- 4. Konfigurisanje boot loader-a ;

```
title Xen 3.0 / XenLinux 2.6
        kernel /boot/xen-3.0.gz console=vga
        module /boot/vmlinuz-2.6-xen root=/dev/hda2 ro console=tty0
        module /boot/initrd-2.6-xen.img
```

Virtuelizacija

- **Virtualne mašine su definisane** i njhove slike zapamćene na **VM Serveru**.
- Definicije su zapamćene u konfiguracionom fajlu **/etc/xen/vm/vm_name**
- Konfiguracioni fajl definiše virtualne resurse uključujući **CPU**, memoriju, mrežnu kartu i blok uređaje. **OS** ga vidi kad je on istaliran i boot-ovan na virtualnu mašinu.

Instalacija Xen-a

5. Restartovanje računara, pokretanje XenLinux-a. Pri pokretanju XenLinux-a pokrene se privilegovani domen *Domain0*

```
student@student-desktop:~$ sudo su -
Password:
root@student-desktop:~# xm list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0   746     1     r-----    658.0
```

Kreiranje virtualnih mašina

6. Kreiranje config. file-ova VM-a

```
kernel = "/boot/vmlinuz-2.6.16.33-xen"
ramdisk = "/boot/initrd-2.6-xen.img"
memory = 128
name = "master"
vif = ['mac=00:0C:76:BC:17:78','bridge=xenbr0']
disk = ['file:/opt/xen/cray/master/centos.4-4.2GB.img,hda2,w', 'file:/opt/xen/cray/master/centos.swap,hda1,w']
root = "/dev/hda2 ro"
```

```
kernel = "/boot/vmlinuz-2.6.16.33-xen"
ramdisk = "/boot/initrd-2.6-xen.img"
memory = 64
name = "slave1"
vcpus = 2
vif = ['']                                                                                                    ,w']
disk = ['file:/opt/xen/cray/slave1/centos.4-4.img,hda2,w', 'file:/opt/xen/cray/slave1/centos.swap,hda1,w']
root = "/dev/hda2 ro"
```

```
kernel = "/boot/vmlinuz-2.6.16.33-xen"
ramdisk = "/boot/initrd-2.6-xen.img"
memory = 64
name = "slave2"
vcpus = 2
vif = ['']                                                                                                     w']
disk = ['file:/opt/xen/cray/slave2/centos.4-4.img,hda2,w', 'file:/opt/xen/cray/slave2/centos.swap,hda1,w']
root = "/dev/hda2 ro"
```

Kreiranje virtualnih mašina

7. VM-a pokrenute (pomoću skripte start-cluster.sh)

```
student@student-desktop:~$ sudo su -
Password:
root@student-desktop:~# xm list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0   746     1     r-----    658.0
master                                       8   128     1     -b----     58.8
slave1                                       9    64     2     -b----     53.9
slave2                                      10    64     2     -b----     46.2
root@student-desktop:~#
```

Logovanje na VM-e

8. Logovanje na virtuelne mašine (master, slave1, slave2).

```
CentOS release 4.4 (Final)
Kernel 2.6.16.33-xen on an i686

master login: castudent
Password:
[castudent@master ~]$
```

```
CentOS release 4.4 (Final)
Kernel 2.6.16.33-xen on an i686

slave1 login: castudent
Password:
[castudent@slave1 ~]$
```

```
CentOS release 4.4 (Final)
Kernel 2.6.16.33-xen on an i686

slave2 login: castudent
Password:
[castudent@slave2 ~]$
```

Xen

4. INSTALACIJA XENA

- **Svaka Xen distribucija uključuje tri osnovne komponente**: Xen, Linux portove i NetBSD da se izvršava na Xenu, kao i alate za korisnike neophodne za vođenje sistema baziranog na Xen-u.
- **Xen 3.0** se može instalirati iz source distribucije, a sa druge strane, moguće je naći i prethodno kompajlirane pakete kao deo distribucije korisnikovog operativnog sistema.

- Postoje određeni zahtevi za Xend kontrolne aplikacije i rad više od jedne virtuelne mašine odjednom, kao i oni koji se koriste samo ako se radi build iz source-a.
- One koji su određeni za Xend kontrolne aplikacije obeleženi su sa "*", a one koji se koriste za "build" od "source" sa znakom "#".
  - *Radna Linux distribucija koja koristi GRUB bootloader i radi na P6 ili najnovijem tipu CPU
  - * Iproute2 paket
  - * Linux bridge-utils
  - * Linux hotplug sistem i odgovarajuće skripte. Na najnovijim distribucijama ovo je uključeno u Linux udev sistem
  - # Build alati (gcc v 3.2 .x ili v 3.3.x, binutils, GNU make)
  - # Razvojna instalacija zlib-a
  - # Razvojna instalacija Python-a v 2.2 ili novijeg
  - # LATEX i transfig su potrebni za kreiranje dokumentacije
- Kad su svi zahtevi ispunjeni, može se dalje instalirati binarna ili source distribucija Xena.

4. INSTALACIJA XENA

- **Instalacija binarne tar arhive**
- Izbildovane tar arhive se mogu skinuti sa XenSource downloads stranice i zatim se otpakuj i instalira na sledeći način:
  - *# tar zxvf xen-3.0-install.tgz*
  - *# cd xen-3.0-install*
  - *# sh ./install.sh*
- Kad se instalira, potrebno je konfigurisati sistem, što će kasnije biti opisano.

- **Instaliranje preko RPM-a**
- Već izbildovani RPM-ovi su isto tako dostupni na XenSource downloads stranici i kada se jednom paket skine, uobičajeno se instalira preko RPM komandi
- *# rpm -iv rpmname*
- Ujedno se tu mogu videti i beleške o trenutno izabranoj verziji.

4. INSTALACIJA XENA

- **Instalacija iz source-a**
- U sledećem segmentu je opisano kako pronaći, izbildovati i instalirati Xen preko source-a.
- Prvenstveno je potrebno da skinemo source kao kompresovan tarball source ili kao klon glavnog razvojnog "skladišta". Bilo koja od verzija se može skinuti sa download stranica sajta.
- Kod izgradnje XenSource-a pokrećemo **Makefile** koji gradi **Xen**, kontrolne alate i **Xend**. On takođe obavlja i funkciju otpakivanja i po potrebi skidanja Linux-a 2.6 source koda i može da ga "zakrpi" za upotrebu uz Xen. Vrši se i izgradnja Linux kernela za upotrebu od strane domena 0 i manjeg, neprivilegovanog kernela za neprivilegovane virtuelne mašine. Kada se bildovanje završi, morao bi da postoji direktorijum dist/ koji je na najvišem nivou i u kome će se kasnije naći svi ciljevi.

4. INSTALACIJA XENA

- **Instalacija iz source-a**
- Posebno su važna dva XenLinux kernel image:
  - jedan sa -xen0 ekstenzijom
  - drugi sa -xenU ekstenzijom
- Kod prvog XenLinux kernel image-a su hardver device drajveri i drajveri za Xen virtuelne uređaje, a kod drugog su samo virtuelni. Oni se nalaze u dist/install/boot/ zajedno sa image-om samog Xen-a i konfiguracionim fajlovima korišćenim za vreme bildovanja.

- Da bi prilagodili skup kernela, potrebno je editovati Makefile tako što ćemo promeniti sledeću liniju:
- **KERNELS ?= linux-2.6 -xen0 linux-2.6-xenU**
- Ova linija se može editovati da uključi bilo koji skup kernela operativnih sistema čija je konfiguracija u buildconfigs/ direktorijumu na najvišem nivou.

4. INSTALACIJA XENA

- Ako želimo da izgradimo prilagođeni XenLinux kernel (npr. da podrži dodatne uređaje ili da omogućimo opcije koje zavise od pojedinačne distribucije) mogu se koristiti standardne Linux konfiguracioni mehanizmi, pri čemu se mora naglasiti da se arhitektura gradi za Xen, i to bi glasilo ovako:
- ***# cd linux-2.6.12-xen0***
- ***# make ARCH=xen xconfig***
- ***# cd ..***
- ***# make***
- Takođe se može kopirati postojeća Linux konfiguracija (.config fajl) u *linux-2.6.12-xen0* i izvršiti
- ***#make -ARCH=xen oldconfig***
- Može da se desi da se ponude opcije specifične za Xen, ali najbolje bi bilo prihvatiti default za ove opcije. Treba primetiti da je u pogledu građe oba ponuđena tipa Linux kernela razlika samo u njihovom konfiguracionom fajlu, gde verzije sa U sufiksom (neprivilegovane) nemaju device drajvere za fizički hardver što dovodi do redukovanja u veličini od 30% i zbog toga može predstavljati dobar izbor za domene bez privilegija. Sufiks 0 imaju privilegovane verzije, koje se mogu koristiti za boot sistema, kao i za drajver domene i domene bez privilegija.

4. INSTALACIJA XENA

- Ako želimo da izgradimo prilagođeni XenLinux kernel (npr. da podrži dodatne uređaje ili
- Fajlovi stvoreni u procesu bildovanja se čuvaju u *dist/install/* direktorijumu, i da bi se instalirali na njihovoj default lokaciji treba uraditi sledeće:
- *# make install*
- Takođe, korisnici koji imaju specijalne instalacione zahteve mogu izvršiti instalaciju tako što će ručno kopirati fajlove u odgovarajuće direktorijume.
- Sam direktorijum *dist/install/boot* takođe sadrži konfiguracione fajlove korišćene tokom izgradnje XenLinux kernela, a i verzija Xena i XenLinux kernela koje sadrže debug simbole poput: *xen-syms-3.0.0* i *vmlinux-syms-2.6.12.6-xen0*, veoma važnih za interpretaciju poruka o greškama. Treba sačuvati ove fajlove jer ih developeri mogu iskoristiti za pronalaženje grešaka.

4.1 KONFIGURACIJA XENA

- Sada kada smo izgradili i instalirali Xen distribuciju možemo jednostavno pripremiti mašinu **za start** i rad Xen-a. Dakle, dolazimo do dela o konfiguraciji koja je neophodna nakon instaliranja binarnih arhiva. Tako, u GRUB konfiguraciji treba ubaciti kod koji dozvoljava Xen/XenLinux-u da startuje, i on se najčešće nalazi ispod */boot* ili */boot/grub* direktorijuma. Ovaj fajl se ponekad naziva menu.1st u zavisnosti od distribucije i njegovi redovi bi trebalo da izgledaju poput navedenih:
  - ***title Xen 3.0 / XenLinux 2.6kernel /boot/xen-3.0.gz dom0_mem=262144 module /boot/vmlinuz-2.6-xen0 root=/dev/sda4 ro console=tty0***
  - Ove linije kernela daju GRUB-u instrukcije za nalaženje Xen-a, kao i informacije o tome koje mu boot parametre treba predati (i to je u ovom slučaju alokacija memorije nultog domena u kilobajtima i postavka serijskog porta). Linija modula konfiguracije opisuje lokaciju XenLinux kernela koju Xen treba da startuje i onih parametara koji treba da mu se predaju.Ti parametri su standardni linux parametri, prepoznaju root device, i nalažu da se isti inicijalno mountuje read-only. Ovi parametri takođe daju instrukcije koje čine da se izlaz sa konzole šalje na monitor. Neke distribucije kao SuSe ne zahtevaju *ro* parametar. Ako želimo da koristimo *initrd,* dodaje se još jedna modul liniju u konfiguraciju, poput: *module /boot/my_initrd.gz.* Kada se instalira novi karnel preporučuje se da se ne briše postojeća opcija menija sa menu.1st ukoliko želimo da startujemo stari Linux kernel u budućnosti i to naročito ukoliko bude bilo problema.

4.1 KONFIGURACIJA XENA

- **Serijska konzola**

- **Preko serijske konzole nam je omogućeno organizovanje, praćenje i interakcija sa našim sistemom.**
- Takođe, na taj način je omogućen pristup sa drugog sistema koji je u blizini i to preko nultog modem kabla ili daljinskog serijskog koncentratora.
- Naš sistem BIOS, bootloader (GRUB), Xen, Linux i login pristupi moraju biti individualno konfigurisani za pristup preko serijske konzole. Pokazalo se da nije neophodna funkcionalnost svake komponente, ali bi moglo biti korisno

4.1 KONFIGURACIJA XENA

- **Konfiguracija bios serijske konzole**

- Kada je omogućen izlaz sistemske serijske konzole, čime nisu niti ukinute niti osposobljene serijske mogućnosti u GRUB-u, Xen-u ili Linux-u, možemo da osposobimo daljinsko upravljanje sistemom i to lakše uz prikaz POST-a i drugih boot poruka preko serijalnog porta i tako što ćemo dozvoliti daljinsku BIOS konfiguraciju.

- Naravno da bi sve ovo bilo moguće, prethodno treba obratiti pažnju na dokumentaciju snabdevača hardvera za mogućnosti i procedure u vezi osposobljavanja BIOS serijske redirekcije.

4.1 KONFIGURACIJA XENA

- **GRUB konfiguracija serijske konzole**
- Kada se omogući "izlaz" GRUB serijske konzole, niti se osposobljavaju niti ukidaju serijske mogucnosti u Xen-u ili Linux-u, ali se može osposobiti daljinsko upravljanje sistemom na lakši način tako što će se prikazati GRUB promptovi, meniji i akcije preko serijskog porta i tako što će se dozvoliti daljinsko GRUB upravljanje.

- U zavisnosti od distribucije ,GRUB-u ćemo omogućiti izlaz i dodavanjem sledeće dve linije njegovom konfiguracionom fajlu:
- ***serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1 terminal --timeout=10 serial console***

- Treba imati u vidu da kada su i serijski port i lokalni monitor i tastatura osposobljeni, tekst "Press any key to continue" će se pojaviti na oba mesta. Pritisak na dugme jednog od ta dva uređaja će da dovede do toga da će GRUB prikazati taj uređaj.
- Drugi uređaj neće videti nikakav "output". Ako nijedno dugme nije pritisnuto u datom roku, sistem će startovati uređaj koji je default u GRUB-u.

4.1 KONFIGURACIJA XENA

- **Konfiguracija xen serijske konzole**

- Osposobljavanje Xen serijske konzole u pogledu njegovog izlaza niti osposobljava niti ukida izlaz Linux kernela ili logovanja u Linux preko serijskog porta.Međutim, to ipak omogućava logovanje i praćenje Xen boot procesa preko serijske konzole što bi se moglo pokazati korisnim pri debagovanju. Da bi se mogao konfigurisati izlaz Xen-a na serijskoj konzoli, neophodno je dodati boot opciju na našem GRUB konfiguracionom fajlu, tj. zameniti prethodno navedeni primer linije koda sa:

- *kernel /boot/xen.gz dom0_mem=131072 com1=115200,8n1*

- Ovo konfiguriše Xen za izlaz na COM1 na 115,200 bauda, 8 data bita, 1 stop bit i bez parnosti. Parametre treba modifikovati prema potrebama okruženja. XenLinux može biti konfigurisan tako da deli serijsku konzolu i da to omogućimo, dodajemo: *"console=ttyS0"'* na našu modul liniju.

4.1 KONFIGURACIJA XENA

- **Konfiguracija linux serijske konzole**
  - Osposobljavanje Linux serijske konzole nam dozvoljava praćenje i logovanje Linux boot procesa preko serijske konzole, što može biti korisno pri debagovanju. Dakle, u samom startu osposobljavanje nam niti osposobljava niti ukida logovanje u Linux preko serijskog porta. Da bi se osposobio Linux izlaz za vreme starta (boot-a) treba dodati sledeći parametar u našoj liniji koda kernela u GRUB-u: *module /vmlinuz-2.6-xen0 ro root=/dev/VolGroup00/LogVol00 \ console=ttyS0, 115200*čime se omogućava izlaz preko 115,200 bauda. Konfiguracija logovanja serijske konzole
- Logovanje u Linux preko serijske konzole pod Xen-om ili drugačije, zahteva postojanje prompta za logovanje pri startu na serijskom portu. Da bi dozvolili root logovanje preko serijske konzole, serijski port mora biti dodan u /etc/securetty. Da bi se automatski startovao login prompt preko serijskog porta, dodajemo sledeću liniju:
- *c:2345:respawn:/sbin/mingetty ttyS0* u /etc/inttab.
- Startujemo
- ***init q***
- da bi naterali ponovan strart inttab-a i start getty-a. Da bi omogućili root login, dodajemo ttyS0 u /etc/securetty ako već nije tamo.Treba naglasiti da naša distribucija može da koristi drugačiji getty, uključujući tu i getty, mgetty i agetty. Informišemo se u pogledu toga na osnovu naše dokumentacije vezano za distibuciju.

4.1 KONFIGURACIJA XENA

- **TLS biblioteke**
- Korisnici XenLinux 2.6 kernela bi **trebalo da onemoguće TLS** pre pokušaja starta XenLinux kernela, i to npr. komandom

- */lib/tls /lib/tls.disabled*

- i naravno uvek ga mogu ponovo osposobiti ukoliko im je nepohodan i to komandom:

- */lib/tls.disabled /lib/tls*

- Razlozi za ukidanje TLs-a su u tome što on koristi određene segmente koji ne pogoduju tj. nisu dopušteni pod Xen-om. Ako TLS nije onemogućen, onda se Xen startuje u emulacionom modu, koji drastično smanjuje performanse. Da bi se osigurale pune performanse, potrebno je instalirati "Xen-prijateljsku" verziju biblioteke.

STARTOVANJE XENA

- Posle ovih podešavanja, trebalo bi da smo u mogućnosti da restartujemo sistem i počnemo da koristimo Xen. Posle restartovanja biramo novi Xen kada se pojavi GRUB na ekranu. Prvo sledi nešto čiji izgled asocira na regularni Linux boot, gde prvi deo izlaza dolazi direktno iz Xen-a i dostavlja nam informacije sa najnižeg stepena koje opisuju sam Xen i hardver na kojem je instaliran, dok poslednji deo dolazi iz XenLinux-a.

- Može se desiti da se prikažu neke greške za vreme XenLinux starta, i zbog njih uglavnom ne bi trebalo brinuti, jer one u većini slučajeva mogu biti prouzrukovane razlikama između kernel konfiguracije XenLinux kernela i onih koje obično koristimo. Kada je startovanje privedeno kraju, trebalo bi da smo u mogućnosti da se ulogujemo na naš sistem na uobičajen način. Međutim, ukoliko to nije moguće, možemo da restartujemo sistem i ponovo izaberemo naš regularni Linux kada se pojavi GRUB opcija-odmah posle restarta

- **Ukoliko smo uspešno obavili startovanje sistema u Xen-u, ono će nas dovesti u privilegovan rukovodeći domen - domen 0**.

- Nakon toga, u stanju smo da pravimo gostujuće domene i startujemo ih sa *xm create* komandom

STARTOVANJE XENA

- **Startovanje nultog domena (domen 0)**
- Posle instalacije i konfiguracije, sistem se restartuje i vrši se izbor opcije za novi Xen, dakle nakon pojavljivanja GRUB opcije na ekranu, kao što je već pomenuto. Zatim sledi nešto što izgleda kao regularni Linux boot sa izlazom direktno iz Xen-a i drugim iz XenLinux-a. Onda sledi startovanje sistema i logovanje na uobičajen način.

- Prvi korak u pravljenju novog domena jeste setovanje root fajl sistema koji će domen da startuje, i on je obično sačuvan na normalnoj particiji, LVM ili nekoj drugoj particiji koja je volume menadžer, disk fajlu ili na NFS serveru. Jednostavan način da se ovo uradi je startovanje sa standardnog OS instalacionog CD-a i instalacija distribucije na drugoj particiji hard drajva. Da startujemo xend kontrolni daemon, kucamo:
- *#xend start*
- tj. može i automatski da se startuje, o čemu će biti govora kasnije. Ukoliko je daemon proradio, omogućeno nam je korišćenje xm aplikacije za praćenje i održavanje domena.

STARTOVANJE XENA

- **Startovanje gostujućih domena i pravljenje konfiguracionog fajla za domen**
- Pre nego što startujemo dodatni domen, kreiramo konfiguracioni fajl, npr:
- */etc/xen/xmexample1*
- je prost templejt konfiguracioni fajl za opis jednostavne VM-e

- */etc/xen/xmexample2*
- je opis templejta koji služi za iskorišćavanje na više virtuelnih mašina i gde određivanje vrednosti vmid varijable na xm komandnoj liniji ispunjava delove ovog templejta.

STARTOVANJE XENA

- Postoje mnogi drugi primeri koji se mogu koristiti, takoda samo treba prekopirati neki od tih fajlova i editovati ga kako nam odgovara. Neke od vrednosti koje se uglavnom menjaju su (dati su i primeri):
  - kernel
  - *"/boot/vmlinuz-2.6-xenU"* odrediti da ovo bude putanja do kernela koji kompajlirate za rad sa Xen-om
  - memory
  - *memory = 64* odrediti da ovo bude veličina memorije domena u megabajtima
  - disk
  - *disk = ['phy:your_hard_drive%d,sda1,w' % (base_partition_number + vmid), 'phy:your_usr_partition,sda6,r'*
  - Odrediti prvi unos u ovoj listi da izračuna graničnu particiju root domena, zasnovano na domenovom ID-u. Odrediti drugi unos da bude lokacija od /usr ako je delimo između domena.
  - dhcp
  - *dhcp="dhcp"* izbacite znake navoda oko dhcp varijable, tako da domen primi svoju IP adresu oko dhcp servera.

STARTOVANJE XENA

- Takođe, možemo da promentimo i vif varijablu da bi mogli samostalno da odaberemo MAC adresu virtuelnog ethernet interfejsa.
- Na primer:
- *vif = ['mac=00:16:3E:F6:BB:B3']*

- Ukoliko ne odredimo ovu varijablu, xend će automatski da generiše MAC adresu iz opsega 00:16:3E:xx:xx:xx koja je dodeljena XenSource-u od IEEE-a kao organizaciono jedinstveni idenitifikator (OUI). XenSource Inc. dozvoljava svakome da koristi slučajne adrese dodeljene iz datog opsega za korišćenje Xen domena.

STARTOVANJE XENA

- **Startovanje gostojućeg domena**
- Xm alatka obezbeđuje širok spektar komandi za upravljanje domenima, dakle create komandu ćemo kristiti da startujemo nove domene.

- Ukoliko smo već kreirali konfiguracioni fajl *myvmconf* koji se zasniva na */etc/xen/xmexample2* i želimo da startujemo domen sa virtuelnom mašinom ID 1, kucamo sledeće:
- *# xm create -c myvmconf vmid=1*

STARTOVANJE XENA

- Automatsko startovanje/zaustavljanje domena
- Moguće je automatsko startovanje nekih domena za vreme boot-a i da domen 0 čeka da se svi domeni koji rade prvo ugase, pre nego što dođe do gašenja celog sistema. Ubacivanjem konfiguracionog fajla određenog domena (ili linka ka fajlu) u /etc/xen/auto direktorijum, oderđujemo koji će domen da startuje pri boot-u.
- Sys-V tip init skripte za Red Hat i LSB-slične sisteme je uključen i automatski prekopiran u /etc/init.d/ za vreme instalacije, tako da se lako kasnije može osposobiti na način koji je u skladu sa distribucijom koju posedujemo. Na primer, na Red Hat-u otkucamo
- *# chkconfig --add xendomains*
- gde će po default-u ovo startovati domene u run-nivouu 3, 4 i 5. Takođe se može iskoristiti i service komanda ukoliko hoćemo da manualno pokrenemo skriptu koja glasi:
- *# service xendomains start*
- S tim smo startovali sve domene ciji su konfiguracioni fajlovi pod /etc/xen/auto, a komandom
- *# service xendomains stop*
- zaustavljamo sve domene koji su aktivni tog momenta

4.3 KONFIGURACIJA I UPRAVLJANJE

- **Alati za upravljanje domenima**
- Xen Daemon (**xend**) obavlja funkcije upravljanja sistemima koji su u vezi sa virtuelnim mašinama, i to tako što formira centar kontrole nad mašinom i ujedno se istovremeno može kontrolisati preko protokola zasnovanom na HTTP-u.
- Da bi virtuelne mašine startovale i da bi se njima i upravljalo, bitno je aktivirati Xend, i on mora da radi kao root radi neophodnosti pristupa privilegovanim funkcijama upravljanja sistemom.

- Na xend komandnoj liniji se može izdati omanja grupa komandi:
  - **# xend start** — **Startvovanje Xend-a ukoliko još nije pokrenut**
  - **# xend stop** — **Zaustavljanje Xend-a ukoliko je pokrenut**
  - **# xend restart** — **Restart Xend-a ako je pokrenut, start ako nije**
  - **# xend status** — **Prikazuje stanje Xend-a na osnovu vraćenog koda**

4.3 KONFIGURACIJA I UPRAVLJANJE

- **Alati za upravljanje domenima**
- SysV init skripta zvana Xend postoji da bi se Xend startovao pri bootovanju
- ***make install*** instalira skriptu u */etc/init.d.* Da bi to osposobili, pravimo simbolički link u odgovarajućim runlevel direktorijumima ili ćemo iskoristiti chkconfig opciju ako postoji.

- Kada **xend** počne da radi, administracija se obavlja preko xm aplikacije.

- Dok xend radi, rezultati i događaji se loguju u
- ***/var/log/xend.log***
- i
- ***/var/log/xend-debug.log***.

- Ovi, uz standardne syslog fajlove, su korisni pri rešavanju problema.

4.3 KONFIGURACIJA I UPRAVLJANJE

- **xm**
- Xm aplikacija je primarna aplikacija za upravljanjem Xen-om sa konzole, i format Xm komandne linije obično izgleda ovako:

- *# xm command [switches] [arguments] [variables]*

- Postojeći skretnice i argumenti zavise od izabrane komande. Varijable se mogu odrediti korišćenjem deklaracije forme *variable=value* i deklaracije komandne linije koja će pregaziti bilo koju od vrednosti u konfiguracionom fajlu koji smo koristili, uključujući i već opisane standardne varijable i bilo koje uobičajene varijable (na primer, *xmdefconfig* fajl koristi *vmid* varijable).
- Za on-line pomoć u vezi postojećih komandi, kucamo *# xm help,*čime ćemo biti u mogućnosti da izlistamo komande koje se najčešće koriste. Ukoliko želimo da vidimo potpunu listu otkucaćemo *xm help-long*, a ukoliko otkucamo *xm help <command>* dobijamo više informacija za tu komandu.

- Osnovne komande za upravljanje
- Jedna od korisnih komandi je
- *# xm*
- koja po redovima lista sve aktivne domene i to u formatu :
- *name domid memory vcpus state cputime*

XEN

- Pored ovih paketa, standardni alat za listanje, kreiranje i brisanje virtualnih mašina koji dolazi inicijalno uz instalaciju Xen-a na Linux serveru je “xm”, koji takođe omogućava listanje virtualnih mašina sa još dodatnih informacija:
- **server:~# xm list**

Name	ID	Mem(MiB)	VCPUs	State	Time(s)
Domain-0	0	675	2	r-----	53930.3
first-xen-vserver	6	64	1	-b----	251.8
second-xen-vserver	8	128	1	-b----	387.5
third-xen-vserver	9	128	1	-b----	199.0

- Ono što ovim listanjem možemo da vidimo je ime virtualne mašine (Domain-0, first-xen-vserver, second-xen-vserver), njen ID broj (0,6,8,9), koliko ram memorije zauzima (675,64,128,128), koliko virtualnih cpu jezgara koristi (2,1,1,1), u kom je statusu (**r je running** i znači da se trenutno izvršavaju neke operacije, b je blocked i označava da je trenutno blokirano izvršavanje operacija ali je virtualna mašina i dalje aktivna), i vreme (time) u sekundama koje označava koliko dugo virtualna mašina radi

4.3 KONFIGURACIJA I UPRAVLJANJE

- **xm**
- Značenja ovih polja su:
  - **name** opisno ime virtuelne mašine
  - **domid** broj domen ID-a na kome radi virtuelna mašina
  - **memory** veličina memorije u megabajtima
  - **mcpus** broj virtuelni CPU-a koje domen ima
  - **state**
    - položaj domena im pet polja :
    - **r** (running) - aktivan
    - **b** (blocked) - blokiran
    - **p** (paused) - pauziran
    - **s** (shutdown)- ugašen
    - **c** (crashed)- oboren
  - **cputime** koliko CPU vremena je domen iskoristio do sada (u sekundama)

- xm list komanda takođe podržava duži format prikaza kada se -l switch koristi, što prikazuje sve detalje aktivnih domena u Xend sxp konfiguracionom formatu.
- Takođe se može dobiti priastup konzoli pojedinačnog domena ukoliko koristimo komandu
- ***# xm console myVM***

4.3 KONFIGURACIJA I UPRAVLJANJE

- **Konfiguracija domena i konfiguracioni fajlovi**
- Xen konfiguracioni fajlovi sadrže sledeće standardne varijable, i ukoliko nije drugačije naznačeno, konfiguracione jedinice bi trebalo da budu zatvorene navodnicima (pogledati konfiguracione skripte u /etc/xen za primere).
  - **kernel**
  - putanja do slike kernela
  - **ramdisk**
  - putanja do slike ram diska (opcionalna)
  - **memory**
  - veličina memorije u megabajtima
  - **vcpus**
  - broj virtuelnih CPU-a
  - **nics**
  - broj virtuelnih mrežnih interfejsa
  - **vif**
  - lista MAC adresa (proizvoljne adrese su dodeljene ako nisu date) i premošćavanja za korišćenje domenovih mrežnih interfejsa, na primer:
  - vif = ['mac=00:16:3E:00:00:11, bridge=xen-br0','bridge=xen-br1']da bi se dodelila MAC adresa i most do prvog interfejsa kao i drugi most do drugog interfejsa, ostavljajući Xend-u da izabere MAC adresu.

4.3 KONFIGURACIJA I UPRAVLJANJE

- **Konfiguracija domena i konfiguracioni fajlovi**
  - **disk** lista block device-a koji eksportuju domen (npr. *disk = ['phy:hda1,sda1,r']*) eksportuje fizički devices */dev/hda1* u domen */dev/sda1* sa read-only pristupom. Eksportovanje read-write-a na disku kao u slučaju kad je disk upravo namešten je opasno. Ukoliko smo potpuno sigurni da to želimo da uradimo, možemo odrediti w! kao mod.
  - **dhcp** podešavamo kao '*dhcp'*, ako želimo da koristimo dhcp pri konfigurisanju mreže
  - **netmask** manualno konfigurisanje IP netmask-a
  - **gateway** manualno konfigurisanje IP gateway-a
  - **hostname** određivanje hostname-a za virtuelnu mašinu
  - **root** određivanje root device parametra na komandnoj liniji kernela
  - **nfs_server** IP adresa za NFS server(ako postoji)
  - **nfs_root** putanja root fajl sistema na BFS serveru
  - **extra** dodatni niz koji se dodaje komndnoj liniji kernela(ako postoji).Ostala polja su dokumentovana u primeru konfiguracionog fajla (npr. konfigurisanje virtuelne TPM funkcionalnosti). Za dodatnu fleksibilnost, moguće je uključiti Python komande za skriptovanje unutar konfiguracionih fajlova.
  - Primer za ovo je *xmexample2* fajl, koji koristi Python kod za rad sa *vmid* varijablom.

4.4 KONFIGURACIJA MREŽE

- Default instalacija bi za većinu korisnika trebalo da radi onako kako je i osmišljena, sem u slučaju komplikovanije postavljenih mreža, koje traže specijalnu konfiguraciju- na primer, više ethernet interfejsa i/ili postojeće mreže sa bridževima.

- **Topologija Xen virtuelne mreže**
- Svaki mrežni interfejs domena je povezan sa virtuelnim mrežnim interfejsom u dom0 preko *point to point* veze (što je kao "virtuelni crossover kabl").

- Ovi uređaji su nazvani ***vif<domid>.<vifid>***
- (npr. *vif1.0* za prvi interfejs u domenu 1, *vif3.1* za drugi interfejs u domenu 3)

- Protok kroz ove virtuelne interfejse je regulisan u domenu 0 i koristi se standardnim Linux mehanizmima za premošćavanje, usmeravanje, ograničavanje protoka, itd. Xend poziva dve shell skripte da odrade inicijalnu konfiguraciju mreže i konfiguraciju novih virtuelnih interfejsa, a po default-u ove skripte konfiguriše i pojedinačni most za sve virtuelne interfejse. Proizvoljno rutiranje/bridžovanje se može konfigurisati podešavanjem skripta.

4.4 KONFIGURACIJA MREŽE

- **Xen mrežne skripte**

- Xen virtuelna mreža je konfigurisana kroz **dve shell skripte** (po default-u ***network-bridge*** i ***vif-bridge***). One se pozivaju automatski u slučaju određenog događaja, sa argumentima za skriptu koji obezbeđuju dalje kontekstualne informacije. Po default-u, ove skripte su */etc/xen/scripts*, a nazivi i lokacije skripti se mogu konfigurisati u */etc/xen/xend-config.sxp*

- **network-bridge(mrežni most):**Ova skripta je pozvana kada se startuje ili stopira Xend, da bi se inicijalizovala ili ugasila Xen virtuelna mreža. U početnoj konfiguraciji, inicijalizacija stvara most "xen-br0" i pomeri eth0 na taj most, menjajući rutiranje u isto vreme. Kada se Xend zatvori, Xen most se briše, eth0 uklanja i normalna IP adresa sa odgovarajućom ruting konfiguracijom se ponovo uspostavlja. vif-bridge: Ova skripta se poziva za svaki virtuelni interfejs domena i može da konfiguriše pravila firewall-a, kao i da doda vif odgovarajućem mostu. Na Xen mostu, koji je početno predodređen, to dodaje i uklanja Vif-ove. Postoje i druge skripte kao primeri (network-route i vif-route, network-nat i vif-nat). Za komplikovanije postavke mreža, tamo gde je na primer potrebno rutiranje ili koje su integrisane sa postojećim mostovima, ove skripte se mogu zameniti podešenim varijantama koje više odgovaraju konfiguraciji našeg sajta.

4.5 SKLADIŠTENJE I RUKOVANJE FAJL SISTEMOM

- U slučaju virtuelnih mašina, skladištenje se može obezbediti na više načina.
- Kao najjednostavniji metod imamo eksportovanje fizičkog blok uređaja (hard drajva ili particije) sa dom0 direktno na gostujući domen, kao virtuelni blok uređaj.
- Skladištenje može takođe da se eksportuje od fajl sistem image-a ili od image particioniranog fajl sistema kao fajl-podržan VBD.
- Na kraju, standardni skladišni protokoli NBD, iSCSI, NFS, itd se mogu koristiti kao skladište virtuelnim mašinama.

4.6 EKSPORTOVANJE FIZIČKOG UREĐAJA KAO VBD

- Kao jedna od najprostijih konfiguracija, javlja se direktno eksportovanje individualnih particija sa domena 0 na druge domene, i da bi se to izvelo, koristimo *phy :* specifikator u našem konfiguracionom fajlu domena. Na primer, linija poput
- *disk = ['phy:hda3,sda1,w']*
- specificira da particija /dev/hda3 u domenu 0 treba da se eksportuje read-write u novi domen kao /dev/sda1, a to isto može da se eksportuje i kao /dev/hda ili /dev/sdb5, po želji. Pored lokalnih diskova i particija, moguće je eksportovati bilo koji uređaj a da ga Linux smatra "diskom" i to na isti način. Na primer, ako imamo iSCSI disk ili GNBD volume ubačen u domen 0, onda smo u mogućnosti da eksportujemo te iste u neke druge domene koristeći phy:disk syntax (npr. *disk = ['phy:vg/lvm1,sda2,w']*).
- Blok uređaji bi trebalo da se dele između domena samo kao read-only varijanta, inače u suprotnom, fajl sistemi Linux-ovog kernela bi mogli postati "zbunjeni" jer se struktura fajl sistema može promeniti pod njima (ako se ext3 particija mauntuje kao rw dva puta, sigurno će izazvati nepopravljivu štetu!).
- Xend će pokušati da nas odvrati od toga tako što će proveriti da uređaj nije mauntovan read-write u domenu 0 i da nije već eksportovan read-write u drugi domen.
- Ukoliko želimo read-write deo, eksportujemo direktorijum u drugi domen preko NFS-a sa domenom 0 (ili koristimo grupni fajl sistem kao što je GFS ili ocfs2)

4.7 RAD SA FAJL-PODRŽANIM VBD-OVIMA

- Kao primarno skladištenje za virtuelnu mašinu, moguće je koristiti fajl u domenu 0, što ne samo da je prikladno nego čak i daje prednost. Ta prednost se ogleda u tome što će virtuelni blok uređaj biti "razrađen" tj. prostor će biti dodeljen samo kao deo korišćenog fajla. To znači da ako virtuelna mašina koristi samo polovinu svog disk prostora onda fajl stvarno zauzima samo pola dodeljenog prostora.

- Na primer, kreiranje 2GB razrađenog fajl-podržanog virtuelnog blok uređaja (koji u suštini zauzima 1KB diska):
- *# dd if=/dev/zero of=vm1disk bs=1k seek=2048k count=1*
- Pravimo fajl sistem u disk fajlu *# mkfs -t ext3 vm1disk* kada alat zatraži potvrdu, izaberemo "y". Popunjavamo fajl sistem, na primer, kopiramo iz sadašnjeg root-a:
- *# mount -o loop vm1disk /mnt*
- *# cp -ax /{root,dev,var,etc,usr,bin,sbin,lib} /mnt*
- *# mkdir /mnt/{proc,sys,home,tmp}*
- Podešavamo fajl sistem tako što menjamo /etc/fstab, /etc/hostname, itd. Važno je da promenimo i fajlove na mauntovanom fajl sistemu, umesto u našem domen 0 fajl sistemu. Na primer, promenimo /mnt/etc/fstab umesto /etc/fstab. U ovom primeru stavljamo /dev/sda1 u root od fstab-a.

- Sada unmauntujemo (i to je vrlo važno) :
- *# unmount / mnt*

4.7 RAD SA FAJL-PODRŽANIM VBD-OVIMA

- U konfiguracionom fajlu određujemo:
- *disk =['file:/full/path/to/vm1disk,sda1,w']*
- Dok virtuelna mašina piše po svom disku, razrađeni fajl će biti popunjen i zauzeće više mesta od orginalnih 2GB.

- Treba imati u vidu da fajl-podržani VBD-ovi mogu biti nezgodni za podršku domena sa intezivnim I/O. Fajl-podržani VBD su poznati po osobini čestog usporavanja pri jačim I/O opterećenjima, iz razloga što loopback blok uređaj koji upravlja I/O -om takođe podržava fajl-podržane VBD u dom0. Bolju I/O performansu možemo dobiti tako što ćemo koristiti ili LVM-podržane VBD ili fizičke uređaje kao VBD.

- Linux podržava najviše 8 fajl-podržanih VBD-ova na svim domenima.
- Ovo ograničenje se može statično povećati tako što se koristi max_loop modularni parametar ako je CONFIG_BLK_DEV_LOOP kompajliran kao modul u dom0 kernelu, ili korišćenjem max_loop=n boot opcije ako je CONFIG_BLK_DEV_LOOP kompajliran direktno u dom0 kernelu

4.8 RAD SA LVM-PODRŽANIM VBD-OVIMA

- Posebno korisna solucija jeste korišćenje LVM sadržaja kao podrške za domen fajl-sisteme, jer ta opcija dozvoljava dinamičan rast odnosno smanjenje celine kao i snapshot-ova i drugih opcija.
- Da bi smo inicijalizovali particiju koja će podržati LVM sadržaj? , kucamo:
- # pvcreate /dev/ sda10
- i sledeći korak jeste da napravimo volume grupu nazvanu "vg" na fizičkoj particiji sa komandom
- # vgcreate vg /dev/ sda10
- Zatim pravimo logičku celinu od 4GB nazvan 'myvmdisk1':
- # lvcreate -L4096M -n myvmdisk1 vg
- U tom momentu bi trebalo da vidimo da imamo /dev/vg/myvmdisk1. Napravićemo fajl sistem, mauntovati ga i popuniti sa :
- # mkfs -t ext3 /dev/vg/myvmdisk1
- # mount /dev/vg/myvmdisk1 /mnt
- # cp -ax / /mnt
- # umount /mnt
- Sada konfigurišemo VM sa sledećom disk konfiguracijom:
- disk = ['phy:vg/myvmdisk1,sda1,w']

4.8 RAD SA LVM-PODRŽANIM VBD-OVIMA

- LVM omogućava rast logičke celine, ali će prethodno morati da se promeni veličina odgovarajućeg fajl sistema ukoliko želimo da koristimo novonastali prostor. Neki fajl sistemi (npr. Ext3) sada podržavaju online promenu veličine. LVM može takođe da se koristi za kreiranje copy-on-write (CoW) klonova LVM sadržaja (poznatio i kao writable persistent snapshots u LVM terminologiji). Ova opcija je nova u Linux-u 2.6.8 pa stoga nije stabilna kao što bi se moglo očekivati. Ustvari, korišćenje više CoW LVM diskova odutima dosta dom0 memorije, a samim tim se i situacije u pogledu hendlovanja grešaka kao što je na primer ostajanje bez disk prostora, ne prihvataju dobro. Postoji nada da će ovo da se popravi u budućnosti.

- Ukoliko želimo da napravimo dva CoW klona gore navedenog fajl sistema, koristimo komande:
- # lvcreate -s -L1024M -n myclonedisk1 /dev/vg/myvmdisk1
- # lvcreate -s -L1024M -n myclonedisk2 /dev/vg/myvmdisk1
- Svaki od njih može da naraste čak do 1GB razlike od glavne celine. Korišćenjem
- # lvextend +100M /dev/vg/myclonedisk
- komande, moguće je povećati količinu prostora za skladištenje razlika. Ne sme se dozvoliti da se razlika celina ikada napuni do kraja, jer može doći do "zbunjivanja" LVM-a. Proces rasta je moguće automatizovati korišćenjem dmsetup wait koji bi primetio kada se celina uveliko ispuni i koji bi onda izdao lvextend komandu. U principu, moguć je nastavak upisa u celinu koja je klonirana (jer promene klonovi ne vide) ali nije preporučljivo imati klon kao "čist" fajl sistem koji nije mauntovan direktno na bilo koju virtuelnu mašinu.

4.9 RAD SA NFS ROOT-OM

- Prvo se popuni root fajl sistem u direktorijumu na server mašini, i to može da bude na posebnoj fizičkoj mašini ili unutar virtuelne mašine na istom nodu. Zatim konfigurišemo NFS server da eksportuje fajl sistem preko mreže dodavanjem linije u etc/exports, na primer:
- */export/vm1root 1.2.3.4/24 (rw,sync,no_root_squash)*

- Domen konfigurišemo da koristi NFS root, i pored normalnih varijabli treba da osiguramo da odredimo i sledeće vrednosti u konfiguracionom fajlu sistema:
- *root = '/dev/nfs'*
- *nfs_server = '2.3.4.5' # substitute IP address of server*
- *nfs_root = '/path/to/root' # path to root FS on the server*
- Domen zahteva pristup mreži za vreme starta, pa stoga treba i statički konfigurisati IP adresu tako što ćemo koristiti config varijable *ip, netmask, gateway, hostname* ili osposobiti DHCP.
- Treba imati u vidu da implementacija Linux NFS root-a je poznata po svojoj nestabilnosti pod velikim opterećenjem (ovo nije problem koji je vezan za Xen), tako da ova konfiguracija nije preporučljiva za servere koji imaju neku kritičnu ulogu.

4.11 PREBACIVANJE DOMENA

- Čuvanje domena i rekonstrukcija

- Administrator Xen sistema ima mogućnost da pauzira trenutno stanje virtuelne mašine na disk fajl u domenu 0, čime se dobija mogućnost nastavka kasnijeg delovanja.
- Na primer, može da se pauzira domen "VM1" na disku uz korišćenje komande:
- # xm save VM1 VM1.chk
- Ovo će zaustaviti domen "VM1" i sačuvaće trenutno stanje u fajlu pod imenom VM1.chk. Da bi nastavili sa "izvršavanjem" ovog domena, upotrebićemo xm restore komandu:
- # xm restore VM1.chk
- Ovo će rekonstruisati stanje domena i nastaviti izvršenje, tako da domen nastavlja da radi ko i pre, a konzola se može ponovo povezati preko xm console komande, kao što je već rečeno.

4.11 PREBACIVANJE DOMENA

- **Prebacivanje i prebacivanje tokom rada**
- Prebacivanje se koristi za transfer domena između fizičkih hostova. Razlikuju se dve verijante prebacivanja: **regularno prebacivanje i prebacivanje tokom rada.**
- **Regularno prebacivanje pomera virtuelnu mašinu sa jednog hosta do drugog bez pravljenja pauze i to tako što se čitav sadržaj memorije kopira i nastavlja sa radom na drugoj (ciljnoj) mašini.**
- **Prebacivanje tokom rada obavlja istu logičku funkcionalnost ali bez potrebe za pauziranjem domena za to vreme**. U principu, kada se izvršava prebacivanje tokom rada, domen nastavlja sa svojim uobičajenim aktivnostima i sa tačke gledišta korisnika to prebacivanje ne bi trebalo ni da bude primetno. Za obavljanje prebacivanja tokom rada, oba hosta moraju da imaju na sebi aktivan Xen i ciljna mašina mora da ima dovoljno resursa (npr. kapacitit memorije) da podrži domen kada je prebacivanje završeno, a i obe mašine bi morale biti na istom L2 subnetu. Trenutno, ne postoji podrška za obezbeđivanje daljinskog pristupa fajl sistemima sačuvanim na lokalnom disku na koji je prebačen domen. Administratori bi trebali da pažljivo izaberu odgovarajuću soluciju za skladištenje (npr. SAN, NAS, itd.) da bi bili sigurni da će fajl sistemi domena postojati na ciljnoj mašini (nodu). GNBD je dobar metod za eksportovanje celina sa jedne mašine na drugu. iSCSI može da odradi sličan posao, ali je komplikovaniji za postavku.

4.11 PREBACIVANJE DOMENA

- **Prebacivanje i prebacivanje tokom rada**
- Kada se domen prebaci, njegove MAC i IP adrese se isto prebacuju uz njega, tako da je VM moguće prebaciti jedino unutar iste layer-2 mreže i IP subnet-a. Ukoliko je ciljni node na različitom subnet-u neophodno je manuelno konfigurisanje od strane administratora odgovarajućeg etherip ili IP tunela u domenu 0 ili u daljinskom nodu.

- **Domen se može prebaciti sa xm migrate komandom**. Komanda za prebacivanje domena u toku rada na drugi domen je:

- # xm migrate --live mydomain destination.ournetwork.com

- Bez -live flega, xend samo zaustavi domene i kopira image iz memorije na novi node i restartuje ga. S obzirom da domeni imaju velike prostore to može potrajati, čak i na Gigabit mreži. Bez -live flega, Xend pokušava da ostavi domen da radi dok je prebacivanje u toku, što dovodi do neaktivnosti od samo 60-300ms. Za sada je neophodno povezati konzolu domena sa novom mašinom, preko xm console komande, i ukoliko taj prebačeni domen ima neke otvorene veze na mreži, one će biti sačuvane, pa SSH veze nemaju ovo ograničenje.

4.12 OBEZBEDJIVANJE XEN-A

- Postoji više mogućih scenarija i odgovarajućih saveta ukoliko je bezbednost Xen sistema narušena.
- Šta imati u vidu u vezi bezbednosti Xena
- Pri postavci Xen sistema, upravljački domen (domen 0) se mora obezbediti koliko je god to moguće, jer ako je sigurnost tog domena dovedena u pitanje time automatski i svi ostali postaju ranjivi.
- **Ovo su "najbolji" saveti za domen 0:**
- 1. Aktiviranje što je moguće manjeg broja potrebnih servisa, jer što je manje stvari u upravljačkoj particiji, to bolje. Servis koji je aktiviran kao root u upravljačkom domenu ima pun pristup bilo kom domenu na sistemu.
- 2. Koristimo firewall da bismo smanjili protok prema upravljačkom domenu , jer on je postavljen da odbija pa će automatski biti i od pomoći kod napada na menadžment domen.
- 3. Korisnicima se ne sme dozvoliti pristup za domen 0. Linux kernel se može ugroziti preko lokalnog korisnika root-a. Ako bi se dozvolio pristup običnim korisnicima (čak i neprivilegovanim korisnicima) mogli bi prouzrokovati povećani rizik ugrožavanja kernela, a to ujedno ugrožava i sve domene.
-

4.12 OBEZBEDJIVANJE XEN-A

- Šeme obezbeđenja

- Odvojena upravljačka mreža
- U ovoj mreži svaki nod ima dve mrežne kartice u grupi, i jedna je povezana sa spoljnim svetom a druga je u fizički odvojenoj upravljačkoj mreži, napravljenoj specijalno za Xen. Sve dok su upravljačke particije pođednako pouzdane, ovo je najsigurnija šema, i nije potrebna nikakva dodatna konfiguracija osim naterivanja Xenda da se poveže na upravljački interfejs za pomeranje.

- Subnet iza firewall-a
- Kod ove šeme, svaki node ima samo jednu mrežnu karticu, ali cela grupa je iza firewall-a koji bi trebao da bude u stanju da:
  - 1. spreči IP spoofing koji dolazi izvan subneta
  - 2. spreči pristup prebačenom portu bilo kog noda u grupi sem ukoliko je on iz unutrašnjosti grupe

4.12 OBEZBEDJIVANJE XEN-A

- Sledeća ip-tables pravila mogu biti korišćena na svakom nodu za sprečavanje spoljnih prebacivanja na njega, ukoliko glavni firewall to već nije preuzeo na sebe.
-
- # this command disables all access to the Xen relocation port:
- iptables -A INPUT -p tcp --destination-port 8002 -j REJECT
- # this command enables Xen relocations only from the specific subnet:
- iptables -I INPUT -p tcp -{}-source 192.168.1.1/8 \
- --destination-port 8002 -j ACCEPT

- **Nodovi na nepoverljivom subnetu**

- Na nepoverljiv subnet prebacivanje nije sigurno po pitanju najnovijih verzija Xena, već je to moguće izvesti kroz osigurani tunel preko VPN ili SSH. Ako ne postoje sigurni tuneli, ostaje nam opcija ukidanja prebacivanja u potpunosti , što je najlakše izvesti preko Iptables:

- # this command disables all access to the Xen relocation port
- iptables -A INPUT -p tcp -{}-destination-port 8002 -j REJECT

4.13 BACKUP XENA

- Iako opcija migracije naizgled pruža skoro upravo ono što nam treba vezano za bekapovanje Xena, to ipak nije tako.

- Osim relativno jednostavnog načina bekapovanja kada pauziramo server, moguće je uraditi i bekap aktivnog servera.
- Jedan od jednostavnih načina jeste da se sve digne **na LVM i uradi snapshot**.

- Taj snapshot se mauntuje na dom0 i onda bekapuje standardnim načinima.
- Međutim ovaj način nije savršen, jer kad se napravi nekoliko screenshot-ova, blok uređaji počinju sve više da se zaključavaju, pa ukoliko se ne očiste, potrebno je čak i restartovati dom0.

Opcije uz bildovanje (pravljenje) i start (boot)

- **Opcije uz bildovanje (pravljenje) i start (boot)**
- Ovde će biti opisane opcije koje se mogu koristiti pri bildovanju i opcije za vreme starta koje se mogu koristiti za nameštanje Xen sistema.

- **Konfiguracione opcije najvišeg nivoa**
- Konfiguracija najvišeg nivoa se izvodi izmenama jednog ili dva fajla: Config.mk i Makefile. Prvi omogućava određivanje ciljne arhitekture i uglavnom se ne menja osim u slučaju ako radimo kros-kompajliranje ili ako želimo da stvorimo Xen sistem za PAE.
- Ostale konfiguracione opcije su dokumentovane u Config.mk. fajlu. Makefile najvišeg nivoa se uglavnom koristi za prilagođavanje kernela koji se grade. Potražimo liniju:
- KERNELS ?= linux-2.6-xen0 linux-2.6-xenU
- Dozvoljene opcije su bilo koji kerneli koji imaj odogovarajući konfiguracioni fajl u buildconfigs/ direktorijumu.

Xen opcije pri pravljenju

- Postoji veliki broj opcija koje Xen koristi za vreme pravljenja, i one se postavljaju kao središnje varijable ili dodate na komandnu liniju make
- **verbose=y**
- Omogućava debugging poruke ako Xen otkrije neočekivano stanje, i isto tako omogućava izlaz konzole za sve domene.
- **debug=y**
- Omogućava debug zaštitu. Uglavnom se koristi kao verbose=y i to najviše za praćenje bagova u Xen-u.
- **debugger=y**
- Omogućava debugger unutar Xen-a, dakle sa njim se može debug Xen, gostujući OS i aplikacije.
- **perfc=y**
- Omogućava merače performanse za značajne događaje unutar Xen-a, i to merenje se može anulirati ili prikazati na Xen konzoli preko kontrolnih dugmadi konzole.

Xen opcije na startu

- Koriste se za konfigurisanje Xen-ovog ponašanja na samom početku. Dodaju se na Xen komandnu liniju, manualno ili menjanjem grub.conf.
  - **noreboot**
  - Komanda koja služi za ukidanje automatskog restarta pri greškama i korisna je da se sačuva prikaz debug-a ukoliko nismo u stanju da vidimo konzoline poruke preko serijske linije.
  - **nosmp**
  - Ova komanda ukida SMP podršku i aktivirana je sa "ignorebiostables".
  - watchdog
  - Omogućava NMI "čuvara" koji je u stanju da prijavi određene greške.
  - **noirqbalance**
  - Ukida softver IRQ balansiranje i sličnosti. Ovo može da se koristi na sistemima poput Dell 1850/2850 koji u hardveru imaju zaobilazna rešenja za rešavanje IRQ-routing problema.

Xen opcije na startu

- **badpage=<page number>,<page number>, ...**
- Određuje listu stranica koje ne trebaj da se alociraju jer poseduju loše bajtove. Na primer, ako tester memorije prijavi da je bajt 0x12345678 loš, onda stavljamo 'badpage=0x12345' na Xen-ovu komandnu liniju.

- **com1=<baud>,DPS,<io_base>,<irq>**
  com2=<baud>,DPS,<io_base>,<irq>
- Xen podržava do 16550 kompatibilnih serijskih portova. Na primer, 'com1=9600, 8n1, 0x408, 5' mapira Com1 na 9600 port, 8 data bita, bez sličnosti, 1 stop bit, I/O port base 0x408, IRQ 5. Ako su neke konfiguracione opcije standardne onda treba naznačiti samo prefix od celog konfiguracionog niza . Ako je baud rate pre-konfigurisana, onda se sme naznačiti "auto" umesto broja koji predstavlja baud rate.

- **console=<specifier list>**
- Komanda koja određuje cilj za I/O Xen-ove konzole i ova lista je odvojena zarezima.

Xen opcije na startu

- **vga**
- Komanda za korišćenje VGA konzole i dozvoljava unos preko tastature.
- **coml**
- Služi za korišćenje serijskog porta com1.
- **com2H**
- Služi za korišćenje serijskog porta com2, gde će prebačeni simboli imati MSB set, a i primljeni simboli moraju imati MSB set.
- **com2L**
- Služi za korišćenje serijskog porta com2, ali ovde simboli koji su prebačeni neće imati MSB kao ni primljeni simboli. Poslednja dva primera omogućuju da se pojedinačni port podeli između dva podsistema (na primer konzola i debugger) i ta podela je kontrolisana preko MSB-a svakog poslatog ili pak primljenog simbola (početna opcija je "com1,vga").
- **sync_console**
- Ova opcija nateruje sinhronizovani izlaz konzole, što je korisno ukoliko sistem iznendano "pukne" pre nego je poslao postojeći izlaz prema konzoli. U većini slučajeva, Xen automatski prelazi u sinhronizovani modus rada ako se desi takav slučaj, ali ova opcija omogućava manualnu protekciju.

Xen opcije na startu

- **conswitch=<switch-char><auto-switch-char>**
- Ova opcija određuje kako prebaciti unos serijske konzole između Xen-a i Dom0. Odgovarajuća sekvenca je CTRL pritisnut tri puta. Naznačivanje backtrick simbola ukida ovo prebacivanje. <auto switch char> naznačava da li će Xen automatski da se prebaci na Dom0 kada startuje- ako je "x" izabrano, onda je automatsko prebacivanje ukinuto. Bilo koja druga vrednost ili u slučaju ako vrednost nije uopšte naznačena, će prihvatiti automatsko prebacivanje.

- **nmi=xxx**
  - Posebno naznačava šta raditi sa NMI paritetom ili I/O greškom.
  - 'nmi=fatal': Xen kopira dijagnozu i onda ističe
  - 'nmi=dom0': Informiše Dom0 o NMI-u
  - 'nmi=ignore': Ignoriše NMI

Xen opcije na startu

- **mem=xxx**
- Određuje granicu fizičke RAM adrese i bilo koji Ram koji se pojavljuje izvan nje u mapi memorije će biti ignorisan. Ovaj parametar se može odrediti sa B, K, M ili G sufiksima koji predstavljaju bajte, kilobajte, megabajte i gigabajte. U slučaju kad sufiks nije naznačen, početne jedinice su kilobajti.
- **dom0_mem=xxx**
- Određuje količinu memorije za domen 0 U Xen 3.x. , parametar se može odrediti sa B, K, M ili G sufiksima koji predstavljaju bajte, kilobajte, megabajte i gigabajte. Ako sufiks nijhe naznačen, početna jedinica su kilobajti. U ranijim verzijama Xen-a, sufiksi nisu bili podržani tako da je vrednost uvek bila u kilobajtima.
- **tbuf_size=xxx**
- Određuje veličinu per-cpu trace buffer-a na stranicama. Treba imati u vidu da su trace buffers aktivirani samo u debug builds, tako da većina korisnika može jednostavno da preskoči ovu opciju.
- **sched=xxx**
- Ova opcija odabira koji CPU raspored Xen treba da koristi. Postoje mogućnosti: "sedf" (default) i "bvt".
- **apic_verbosity=debug,verbose**
- Izbacuje detaljnije informacije o lokalnim APIC i IOAPIC konfiguracijama.

Xen opcije na startu

- **lapic**
- Forsira korišćenje lokalnog APIC čak i ako je ta opcija ukinuta preko BIOS-a sa jednim procesorom.
- **nolapic**
- Ignoriše lokalni APIC u sistemu sa jednim procesorom, čak i ako je opcija omogućena preko BIOS-a.
- **apic=bigsmp,default,es7000,summit**
- Određuje NUMA platformu, i to se uglavnom može automatski testirati (proveriti).
- Takođe, i sledeće opcije se mogu koristiti na Xen komandnoj liniji. Pošto domen 0 snosi deo odgovornosti pri startovanju platforme, Xen će automatski da otkuca ove opcije na svojoj komandnoj liniji, i te opcije su uzete iz sintakse Linux komandnih linija, bez promene u značenju reči.
- **acpi=off,force,strict,ht,noirq,...**
- Naglašava kako Xen (i domen 0) rasčlanjavaju BIOS ACPI tabele
- **acpi_skip_timer_override**
- ': vezivanje konzole za /dev/ttyS0

Xen opcije na startu

- **acpi_skip_timer_override**
- Daje instrukcije Xen-u (i domenu 0) da ne obraća pažnju na vremenske prekide prelaznih instrukcija koje su naznačene preko BIOS ACPI tabele
- **noapic**
- Daje instrukcije Xen-u (i domenu 0) da ne obrate pažnju na bilo koji IOAPIC u sistemu, i da umesto njih nastave da koriste nasledni? PIC.

- **XenLinux opcije pri startu**
- Pored standardnih Linux kernel opcija, podržavaju se:
- **xencons=xxx**
- Naznačuje device nod na koji je drajver Xen virtuelne konzole prikačen. Sledeće opcije su podržane:

- 'xencons=off': onemogućavanje virtuelne konzole
- 'xencons=tty': vezivanje konzole za /dev/tty1 (tty0 pri boot-ovanju)
- 'xencons=ttyS': vezivanje konzole za /dev/ttyS0