Chapter 16: Virtual Machines

هذا التشابتر يتكلم عن الآلات الافتراضية (Virtual Machines - VMs)، وهي تقنية تسمح بتشغيل عدة أنظمة تشغيل مختلفة على نفس الجهاز الحقيقي وكأن كل نظام يمتلك جهازه الخاص.

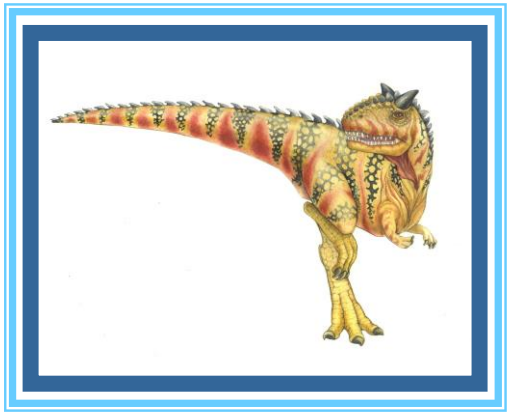
بدل أن يكون لديك جهاز لكل نظام تشغيل، يمكن لجهاز واحد أن ينشئ عدة "أجهزة وهمية" (Virtual Machines)، وكل جهاز افتراضي يعمل بشكل مستقل ويحتوي على:

نظام تشغيل خاص به.

ذاكرة خاصة به.

معالج افتراضي.

ملفات وبرامج خاصة به.



Silberschatz, Galvin and Gagne ©2013



Overview

الفكرة الأساسية في Virtual Machines هي أننا نحول جهاز كمبيوتر واحد حقيقي إلى عدة بيئات تشغيل مستقلة.
يعني نفس الجهاز يمكنه أن يعمل كأنه عدة أجهزة منفصلة.
المكونات:
Host: الجهاز الحقيقي نفسه، أي الهاردوير الأساسي.
VMM / Hypervisor: البرنامج الذي ينشئ ويشغّل الـ Virtual Machines، ويعطيها واجهة تشبه الجهاز الحقيقي.
Guest: النظام أو البرنامج الذي يعمل داخل الـ Virtual Machine، وغالبًا يكون نظام تشغيل.

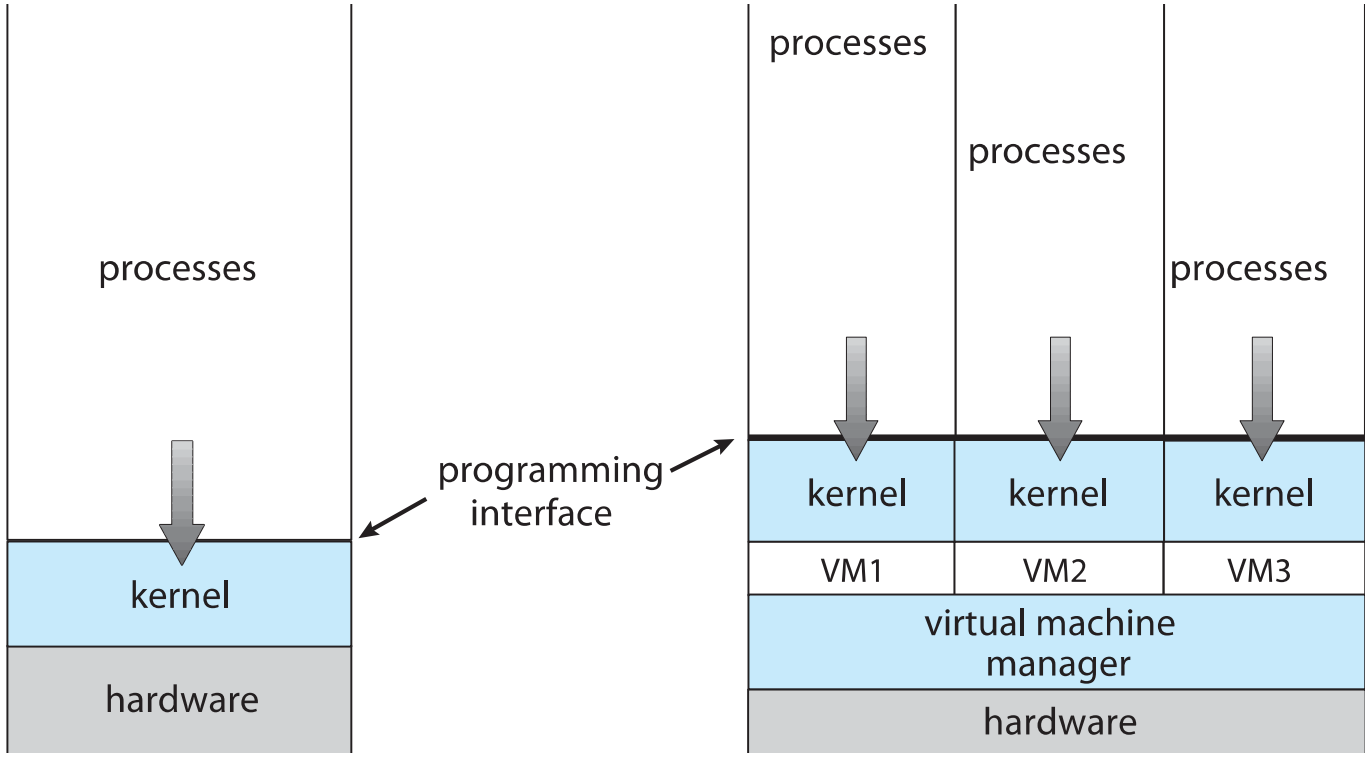
- Fundamental idea – abstract hardware of a single computer into several different execution environments
  - Similar to layered approach
  - But layer creates virtual system (**virtual machine**, or **VM**) on which operation systems or applications can run
- Several components
  - **Host** – underlying hardware system
  - **Virtual machine manager** (**VMM**) or **hypervisor** – creates and runs virtual machines by providing interface that is ***identical*** to the host
    - (Except in the case of paravirtualization)
  - **Guest** – process provided with virtual copy of the host
    - Usually an operating system
- Single physical machine can run multiple operating systems concurrently, each in its own virtual machine





System Models

شكل توضيحي للآلة الإفتراضية والغير إفتراضية



Non-virtual machine

Virtual machine





History

- First appeared in IBM mainframes in 1972
- Allowed multiple users to share a batch-oriented system
- Formal definition of virtualization helped move it beyond IBM
  1. A VMM provides an environment for programs that is essentially identical to the original machine
  2. Programs running within that environment show only minor performance decreases
  3. The VMM is in complete control of system resources
- In late 1990s Intel CPUs fast enough for researchers to try virtualizing on general purpose PCs
  - **Xen** and **VMware** created technologies, still used today
  - Virtualization has expanded to many OSes, CPUs, VMMs





Benefits and Features

- Host system protected from VMs, VMs protected from each other
  - I.e. A virus less likely to spread
  - Sharing is provided though via shared file system volume, network communication
- Freeze, suspend, running VM
  - Then can move or copy somewhere else and resume
  - Snapshot of a given state, able to restore back to that state
    - Some VMMs allow multiple snapshots per VM
  - Clone by creating copy and running both original and copy
- Great for OS research, better system development efficiency
- Run multiple, different OSes on a single machine
  - Consolidation, app dev, …

1) الحماية والعزل
كل VM تكون معزولة عن الثانية، والجهاز الأساسي أيضًا يكون محميًا منها.
هذا يعني أن الفيروسات أو المشاكل داخل VM أقل احتمالًا أنها تنتقل لباقي الـ VMs أو للـ Host.

2) المشاركة ممكنة
رغم العزل، ما زالت الـ VMs تستطيع المشاركة في أشياء مثل:
Shared folders / file system
Network communication

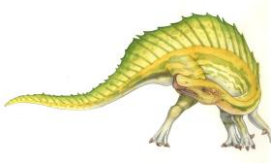
3) الإيقاف والحفظ والاسترجاع
يمكنك:
Freeze: تجميد الـ VM
Suspend: إيقافها مؤقتًا
ثم نقلها أو نسخها إلى مكان آخر وإكمال العمل منها لاحقًا.

4) Snapshot
الـ Snapshot يعني أخذ صورة للحالة الحالية للـ VM.
إذا صار خطأ، نقدر نرجع لنفس الحالة القديمة بسهولة.
وبعض الأنظمة تسمح بأكثر من Snapshot لنفس الـ VM.

5) Clone
يعني إنشاء نسخة كاملة من الـ VM، ثم تشغيل الأصل والنسخة معًا.

6) مفيدة جدًا في التطوير والبحث
الـ Virtual Machines ممتازة في:
بحث أنظمة التشغيل
تطوير البرامج
تجربة التغييرات بدون خطر على النظام الحقيقي

7) تشغيل أكثر من نظام تشغيل على نفس الجهاز
يمكنك تشغيل أنظمة مختلفة على نفس الكمبيوتر في نفس الوقت، وهذا يفيد في:
Consolidation
App development





Benefits and Features (cont.)

- **Templating** – create an OS + application VM, provide it to customers, use it to create multiple instances of that combination
- **Live migration** – move a running VM from one host to another!
  - No interruption of user access
- All those features taken together -> **cloud computing**
  - Using APIs, programs tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, virtual desktops

8) Templating
يمكن إنشاء **VM** جاهزة تحتوي:
نظام تشغيل
برامج معينة
ثم يتم نسخها وتقديمها للعملاء أو إنشاء عدة نسخ منها بسرعة.

9) Live Migration
يعني نقل **VM** وهي شغالة من جهاز إلى جهاز آخر بدون انقطاع للمستخدم.

10) علاقتها بالحوسبة السحابية
هذه الميزات كلها تعتبر أساس مهم في **Cloud Computing**، حيث يمكن للبرامج أن تطلب من السحابة إنشاء:
VMs
servers
storage
virtual desktops

الخلاصة
الـ **Virtual Machines** تعطيك عزل، أمان، مرونة، سهولة نسخ ونقل، وتشغيل عدة أنظمة على نفس الجهاز.
وهذا هو السبب الأساسي في أهميتها الكبيرة في التطوير والسحابة.





Building Blocks

- Generally difficult to provide an ***exact*** duplicate of underlying machine
  - Especially if only dual-mode operation available on CPU
  - But getting easier over time as CPU features and support for VMM improves
  - Most VMMs implement virtual CPU (VCPU) to represent state of CPU per guest as guest believes it to be
    - When guest context switched onto CPU by VMM, information from VCPU loaded and stored
  - Several techniques, as described in next slides

مشكلة أساسية
ليس من السهل دائمًا إنشاء نسخة مطابقة 100٪ من الجهاز الحقيقي داخل الـ VM.
والسبب أن بعض المعالجات (CPUs) لا توفر كل الميزات اللازمة للمحاكاة الكاملة، خصوصًا إذا كانت تعتمد فقط على:
User Mode
Kernel Mode
(ما يسمى Dual-Mode Operation).
مع تطور المعالجات
أصبحت المهمة أسهل لأن المعالجات الحديثة أضافت دعمًا خاصًا للـ Virtualization.
لذلك:
أصبح إنشاء الـ VMs أسهل.
الأداء أصبح أفضل.
عمل الـ Hypervisor أصبح أكثر كفاءة.
Virtual CPU (VCPU)
معظم الـ VMMs تنشئ شيئًا يسمى:
VCPU = Virtual CPU
وهو معالج افتراضي لكل Guest.
الـ Guest يعتقد أنه يمتلك CPU خاصًا به، لكن في الحقيقة هو يستخدم الـ CPU الحقيقي بشكل غير مباشر.
عند تشغيل VM:
يقوم الـ VMM بـ:
تحميل حالة الـ VCPU إلى المعالج الحقيقي.
تشغيل الـ Guest.
عند التبديل إلى VM أخرى، يحفظ الحالة الحالية.
يحمل حالة الـ VCPU الخاصة بالـ VM التالية.
هذه العملية تشبه Context Switching بين العمليات.





Types of Virtual Machines and Implementations

- Many variations as well as HW details
  - Assume VMMs take advantage of HW features
    - HW features can simplify implementation, improve performance
- Whatever the type, a VM has a lifecycle
  - Created by VMM
  - Resources assigned to it (number of cores, amount of memory, networking details, storage details)
  - In type 0 hypervisor, resources usually dedicated
  - Other types dedicate or share resources, or a mix
  - When no longer needed, VM can be deleted, freeing resouces
- Steps simpler, faster than with a physical machine install
  - Can lead to virtual machine sprawl with lots of VMs, history and state difficult to track

توجد أنواع كثيرة من الـ VMs
دورة حياة الـ VM (VM Lifecycle)
1. Creation
يقوم الـ VMM بإنشاء الـ VM.
2. Resource Assignment
يتم تخصيص الموارد لها مثل:
عدد الأنوية (CPU Cores)
حجم الذاكرة (RAM)
إعدادات الشبكة
مساحة التخزين
3. Running
تبدأ الـ VM بالعمل واستخدام الموارد.
4. Deletion
عندما لا تحتاجها:
يتم حذف الـ VM.
يتم تحرير الموارد وإعادتها للنظام.
الموارد قد تكون:
Dedicated
مخصصة بالكامل لـ VM واحدة.
Shared
تشاركها عدة VMs.
Mix
مزيج بين الاثنين.
ميزة مهمة
إنشاء VM أسرع بكثير من إنشاء جهاز حقيقي جديد أو تثبيت نظام تشغيل من الصفر.

انتشار عدد كبير من الـ VMs مما يجعل إدارتها صعبة





Virtualization and Operating-System Components

- Now look at operating system aspects of virtualization
  - CPU scheduling, memory management, I/O, storage, and unique VM migration feature
    - How do VMMs schedule CPU use when guests believe they have dedicated CPUs?
    - How can memory management work when many guests require large amounts of memory?

عندما نشغل عدة VMs على جهاز واحد، تظهر عدة أسئلة:
كيف يتم توزيع المعالج؟
كل Guest يعتقد أن لديه CPU خاص به، لكن في الواقع هناك عدد محدود من المعالجات الحقيقية.
كيف تتم إدارة الذاكرة؟
قد تحتاج عدة VMs إلى كميات كبيرة من الـ RAM في نفس الوقت.
ماذا عن:
I/O (الإدخال والإخراج)
Storage (التخزين)
VM Migration (نقل الـ VM)
كل هذه الأمور يجب أن يديرها الـ VMM (Hypervisor).





OS Component – CPU Scheduling

- Even single-CPU systems act like multiprocessor ones when virtualized
  - One or more virtual CPUs per guest
- Generally VMM has one or more physical CPUs and number of threads to run on them
  - Guests configured with certain number of VCPUs
    - Can be adjusted throughout life of VM
  - When enough CPUs for all guests -> VMM can allocate dedicated CPUs, each guest much like native operating system managing its CPUs
  - Usually not enough CPUs -> CPU **overcommitment**
    - VMM can use standard scheduling algorithms to put threads on CPUs
    - Some add fairness aspect

كل Guest يمتلك VCPU
حتى لو كان الجهاز يحتوي على CPU واحد فقط، فإن كل VM تشعر وكأن لديها معالجها الخاص.
إذا كانت المعالجات كافية
يقوم الـ VMM بتخصيص CPU لكل Guest.
فتعمل الـ VMs تقريباً مثل الأجهزة الحقيقية.

إذا لم تكن المعالجات كافية
يحدث:
CPU Overcommitment
أي:
عدد الـ VCPUs > عدد الـ CPUs الحقيقية
وهنا يبدأ الـ VMM بجدولة (Scheduling) وقت المعالج بين الـ Guests





OS Component – CPU Scheduling (cont.)

- Cycle stealing by VMM and oversubscription of CPUs means guests don't get CPU cycles they expect
  - Consider timesharing scheduler in a guest trying to schedule 100ms time slices -> each may take 100ms, 1 second, or longer
    - Poor response times for users of guest
    - Time-of-day clocks incorrect
  - Some VMMs provide application to run in each guest to fix time-of-day and provide other integration features

ما معنى Cycle Stealing؟
عندما يقرر الـ VMM إعطاء الـ CPU لـ VM أخرى، فإنه "يسرق" بعض الوقت الذي كانت VM الحالية تتوقع الحصول عليه.
لذلك الـ Guest يعتقد أنه يملك CPU بشكل مستمر، لكن الواقع مختلف.
Oversubscription
يعني أن عدد الـ VMs أو الـ VCPUs أكبر من قدرة المعالج الحقيقية.
فتبدأ الـ VMs بالتنافس على وقت الـ CPU.
مثال مهم
افترض أن داخل الـ Guest يوجد Scheduler عادي.
وهو يعطي كل Process:
Time Slice = 100 ms
الـ Guest يتوقع أن العملية ستأخذ:
100 ms
لكن بسبب تدخل الـ VMM قد يحدث التالي:
100 ms
أو 1 ثانية
أو أكثر
قبل أن تحصل العملية على وقتها الكامل.
النتائج
1. بطء الاستجابة
2. مشاكل الوقت
كيف يتم حل المشكلة؟
بعض أنظمة الـ Virtualization توفر برنامجًا خاصًا داخل الـ Guest يسمى غالبًا:
Guest Tools / Integration Tools
وظيفته:
تصحيح الوقت داخل الـ VM.
مزامنة الساعة مع الـ Host.
تحسين الأداء.
تسهيل التواصل بين الـ Guest والـ Host.





OS Component – Memory Management

أي أن مجموع الذاكرة التي تحتاجها الـ VMs أكبر من الـ RAM الحقيقية المتوفرة.

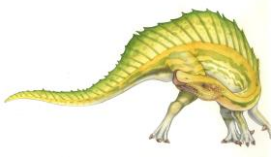
- Also suffers from oversubscription -> requires extra management efficiency from VMM
- For example, VMware ESX guests have a configured amount of physical memory, then ESX uses 3 methods of memory management
  1. Double-paging, in which the guest page table indicates a page is in a physical frame but the VMM moves some of those pages to backing store
  2. Install a pseudo-device driver in each guest (it looks like a device driver to the guest kernel but really just adds kernel-mode code to the guest)
     - Balloon memory manager communicates with VMM and is told to allocate or deallocate memory to decrease or increase physical memory use of guest, causing guest OS to free or have more memory available
  3. Deduplication by VMM determining if same page loaded more than once, memory mapping the same page into multiple guests

طرق VMware ESX لإدارة الذاكرة

1) Double Paging
الـ Guest يعتقد أن الصفحة موجودة في RAM.
لكن الـ VMM قد ينقل بعض الصفحات إلى:
Backing Store (Disk)
فتصبح الصفحة موجودة على القرص بدل الذاكرة دون أن يعرف الـ Guest.

2) Balloon Memory Manager
يتم تثبيت Driver خاص داخل الـ Guest.
يسمى:
Balloon Driver
وظيفته التواصل مع الـ VMM.
إذا احتاج الـ VMM إلى توفير ذاكرة:
يطلب من الـ Balloon Driver حجز جزء من ذاكرة الـ Guest.
فيظن الـ Guest أن ذاكرته امتلأت.
فيبدأ بتحرير صفحات أو استخدام Paging بنفسه.
إذا توفرت ذاكرة أكثر:
يفرغ الـ Balloon Driver الذاكرة المحجوزة.
فتعود للـ Guest.

3) Deduplication
إذا اكتشف الـ VMM أن عدة VMs تحمل نفس الصفحة بالضبط (مثلاً نفس ملفات النظام):
بدلاً من تخزينها عدة مرات:
يخزن نسخة واحدة فقط.
يجعل جميع الـ Guests تشير إليها.
النتيجة:
توفير كبير في الذاكرة.





OS Component – I/O

- Easier for VMMs to integrate with guests because I/O has lots of variation
  - Already somewhat segregated / flexible via device drivers
  - VMM can provide new devices and device drivers
- But overall I/O is complicated for VMMs
  - Many short paths for I/O in standard OSes for improved performance
  - Less hypervisor needs to do for I/O for guests, the better
  - Possibilities include direct device access, DMA pass-through, direct interrupt delivery
    - Again, HW support needed for these
- Networking also complex as VMM and guests all need network access
  - VMM can bridge guest to network (allowing direct access)
  - And / or provide network address translation (NAT)
    - NAT address local to machine on which guest is running, VMM provides address translation to guest to hide its address

لماذا I/O أسهل قليلاً؟
لأن أنظمة التشغيل أصلاً تعتمد على:
Device Drivers
لذلك يستطيع الـ VMM إنشاء أجهزة افتراضية جديدة مع Drivers خاصة بها.
لكن ما زالت العملية معقدة لأن عمليات الإدخال والإخراج تحتاج سرعة عالية.
كلما تدخل الـ Hypervisor أقل: كان الأداء أفضل.

طرق تحسين الأداء
Direct Device Access
إعطاء الـ Guest وصولًا مباشرًا للجهاز.
DMA Pass-Through
Direct Interrupt Delivery
إرسال Interrupts مباشرة إلى الـ Guest.
غالبًا تحتاج دعمًا من الهاردوير.

Networking
كل من:
Host
Hypervisor
Guests
يحتاجون اتصالًا بالشبكة.
Bridging
الـ Guest يظهر على الشبكة كجهاز مستقل.
يحصل على عنوان IP خاص به.
NAT
الـ Guest يحصل على عنوان داخلي.
والـ VMM يقوم بترجمة العناوين.
ميزة NAT:
إخفاء عنوان الـ Guest الحقيقي.
زيادة الأمان.

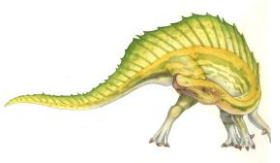




OS Component – Storage Management

- Both boot disk and general data access need be provided by VMM
- Need to support potentially dozens of guests per VMM (so standard disk partitioning not sufficient)
- Type 1 – storage guest root disks and config information within file system provided by VMM as a disk image
- Type 2 – store as files in file system provided by host OS
- Duplicate file -> create new guest
- Move file to another system -> move guest
- Physical-to-virtual (P-to-V) convert native disk blocks into VMM format
- Virtual-to-physical (V-to-P) convert from virtual format to native or disk format
- VMM also needs to provide access to network attached storage (just networking) and other disk images, disk partitions, disks, etc

ما الذي يحتاجه الـ VMM؟
يجب أن يوفر:
Boot Disk
Data Storage
لكل Guest.
Type 1 Hypervisor
يخزن بيانات الـ Guest على شكل:
Disk Image
داخل نظام التخزين الخاص بالـ VMM.
Type 2 Hypervisor
يخزن الـ Guest كملفات داخل نظام ملفات الـ Host.
مثال:
Windows10.vmdk
Ubuntu.vdi
ميزات Disk Images
نسخ الملف
= إنشاء Guest جديدة.
نقل الملف
= نقل الـ Guest بالكامل إلى جهاز آخر.
تحويل الأنظمة
P-to-V
Physical to Virtual
تحويل جهاز حقيقي إلى VM.
V-to-P
Virtual to Physical
تحويل VM إلى جهاز حقيقي.

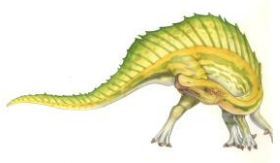




OS Component – Live Migration

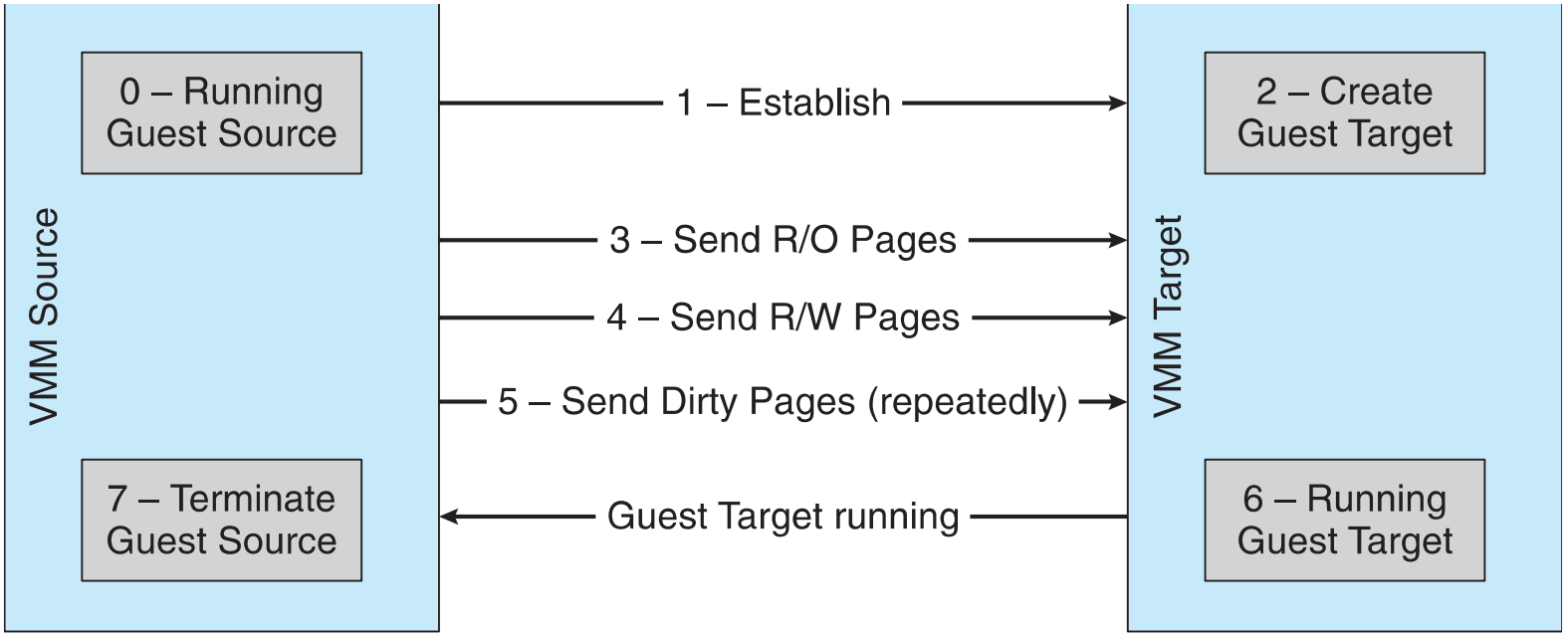
- Taking advantage of VMM features leads to new functionality not found on general operating systems such as live migration
- Running guest can be moved between systems, without interrupting user access to the guest or its apps
- Very useful for resource management, maintenance downtime windows, etc
  1. The source VMM establishes a connection with the target VMM
  2. The target creates a new guest by creating a new VCPU, etc
  3. The source sends all read-only guest memory pages to the target
  4. The source sends all read-write pages to the target, marking them as clean
  5. The source repeats step 4, as during that step some pages were probably modified by the guest and are now dirty
  6. When cycle of steps 4 and 5 becomes very short, source VMM freezes guest, sends VCPU's final state, sends other state details, sends final dirty pages, and tells target to start running the guest
     - Once target acknowledges that guest running, source terminates guest

ميزة تسمح بنقل VM وهي تعمل من جهاز إلى جهاز آخر:
بدون أن يشعر المستخدم بأي انقطاع.
الفوائد
الصيانة.
توزيع الأحمال.
إدارة الموارد.
تجنب إيقاف الخدمات.
خطوات Live Migration
1
ينشئ Source VMM اتصالاً مع Target VMM.
2
ينشئ Target VM جديدة.
(ينشئ VCPU والموارد الأساسية)
3
إرسال صفحات الذاكرة Read-Only.
4
إرسال صفحات Read-Write.
ويتم اعتبارها Clean.
5
أثناء النقل قد تتغير بعض الصفحات.
فتصبح:
Dirty Pages
لذلك يعاد إرسالها مرة أخرى.
6
يتكرر الإرسال حتى يصبح عدد الصفحات المتغيرة قليلاً جداً.
عندها:
يتم إيقاف الـ Guest مؤقتاً (Freeze).
إرسال حالة الـ VCPU النهائية.
إرسال آخر Dirty Pages.
تشغيل الـ Guest على الجهاز الهدف.
بعد التأكد من نجاح التشغيل:
يتم حذف النسخة الموجودة على الجهاز المصدر.





Live Migration of Guest Between Servers







Examples - VMware

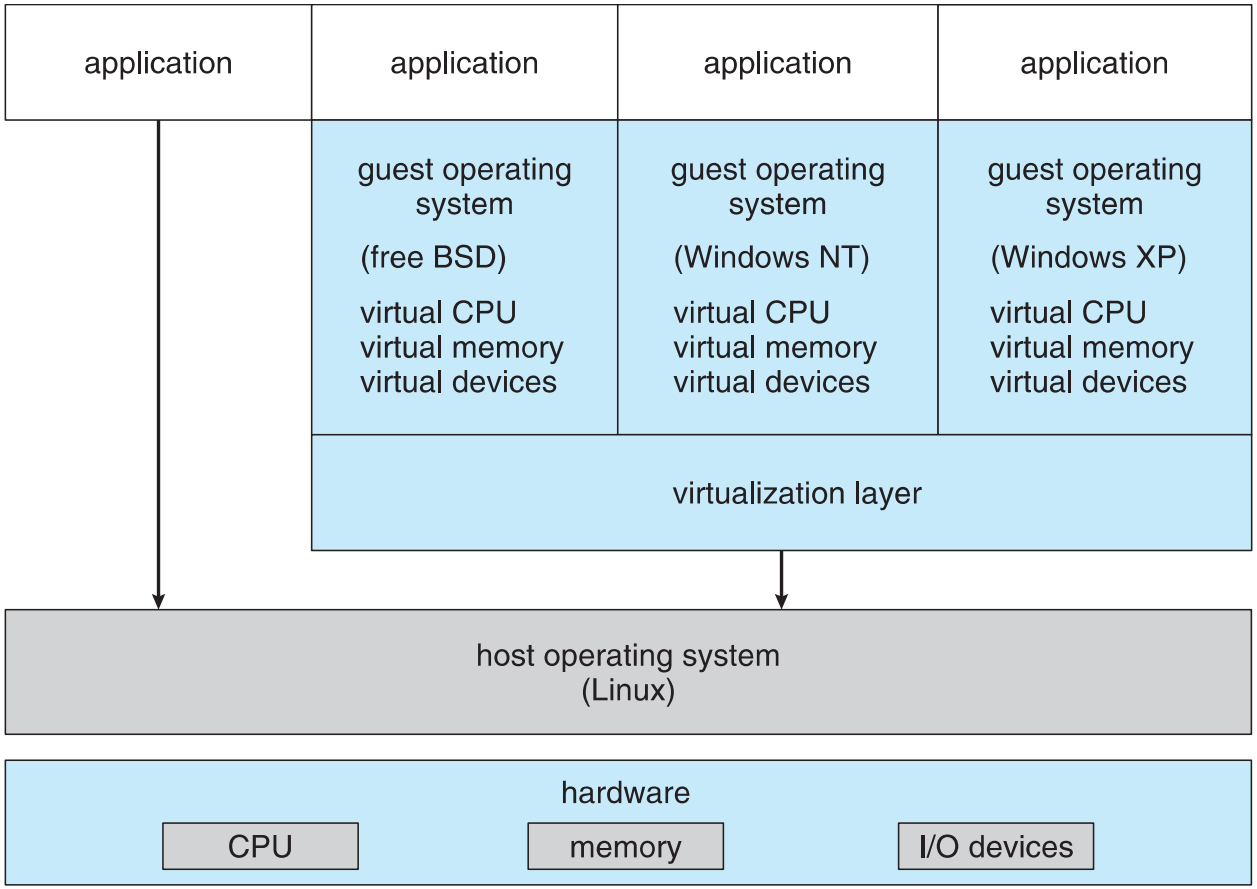
- VMware Workstation runs on x86, provides VMM for guests
- Runs as application on other native, installed host operating system -> Type 2
- Lots of guests possible, including Windows, Linux, etc all runnable concurrently (as resources allow)
- Virtualization layer abstracts underlying HW, providing guest with is own virtual CPUs, memory, disk drives, network interfaces, etc
- Physical disks can be provided to guests, or virtual physical disks (just files within host file system)

VMware
Type 2 Hypervisor.
يعمل فوق Host OS.
يشغل عدة Guest OS.
يوفر Virtual CPU و Memory و Disk.





VMware Workstation Architecture







Examples – Java Virtual Machine

- Example of programming-environment virtualization
- Very popular language / application environment invented by Sun Microsystems in 1995
- Write once, run anywhere
- Includes language specification (Java), API library, Java virtual machine (JVM)
- Java objects specified by class construct, Java program is one or more objects
- Each Java object compiled into architecture-neutral **bytecode** output (`.class`) which JVM **class loader** loads
- JVM compiled per architecture, reads bytecode and executes
- Includes **garbage collection** to reclaim memory no longer in use
- Made faster by **just-in-time** (**JIT**) compiler that turns bytecodes into native code and caches them

JVM
Programming Virtual Machine.
تدعم مبدأ Write Once, Run Anywhere.
تشغل ملفات .class.
تستخدم Bytecode.
تحتوي على Garbage Collection.
تستخدم JIT Compiler لتحسين الأداء.





The Java Virtual Machine

