2 Virtualización

2.1 Introducción

La Real Academia de la Lengua define virtual (en una de sus acepciones): "Que tiene existencia aparente y no real". Con base en esta definición, podríamos decir que en los ambientes creados por software todo es virtual y, en particular, que el ambiente creado por un sistema operativo para la ejecución de aplicaciones es también virtual. Un sistema operativo permite virtualizar los recursos de un computador: el procesador, el disco y la memoria RAM; así podemos compartir recursos y darles un uso más eficiente.

La virtualización permitió no solo la construcción de una máquina virtual sobre una máquina física (lo que hace el sistema operativo), además permitió a la comunidad diseñar e implementar el concepto de máquina virtual: muchas máquinas virtuales corriendo sobre una máquina física. El concepto de máquina virtual fue propuesto en 1974, pero solo recientemente el desarrollo tecnológico permitió llevarlo a la práctica de forma eficiente haciendo posible la construcción flexible de ambientes de software diversos.

Este capítulo aborda los conceptos fundamentales de virtualización y está organizado así:

- La sección 2.2 presenta una introducción general a los conceptos virtuales que el sistema operativo crea para manejar procesos, espacio en disco y memoria RAM, creando para los usuarios una apariencia de uso exclusivo de estos recursos.
- Las secciones 2.3 y 2.4 presentan en detalle el manejo de la memoria virtual y el sistema de archivos respectivamente. Estos componentes del sistema operativo son fundamentales para crear el ambiente apropiado para un usuario.
- La última sección presenta una introducción al concepto de máquinas virtuales y contenedores.

El contenido de este documento está basado en dos fuentes principales: Modern Operating Systems de Andrew Tanenbaum y Conceptos de Sistemas Operativos de Abraham Silberschatz, Peter Galvin y Greg Gagne. La sección de máquinas virtuales en el artículo "Formal requirements for virtualizable third generation architectures" de Gerald Popek, Gerald y Robert Goldberg.

2.2 El sistema operativo como manejador de un ambiente virtual

Aunque no es común encontrar esta definición de sistema operativo como manejador de un ambiente virtual, el sistema operativo se encarga de administrar el hardware de una máquina física: procesador, dispositivos de Entrada/Salida y memoria RAM, para presentar a los usuarios una representación virtual de una máquina física. Adicionalmente, en un sistema multiusuario como los servidores hoy día, el sistema operativo administra de forma eficiente la asignación de los recursos a múltiples usuarios creando para cada usuario la apariencia de una máquina que funciona para ese usuario de forma exclusiva.

Virtualización del procesador. El sistema operativo administra de forma eficiente la asignación del procesador, asigna turnos a los procesos que están en ejecución para que avancen de forma concurrente dando a los usuarios la apariencia de una máquina que solamente ejecuta sus propios procesos. En algunas ocasiones el tiempo de respuesta aumenta y solo entonces somos conscientes de estar trabajando en una máquina compartida.

Para aparentar la disponibilidad de un procesador exclusivo para cada proceso, el sistema operativo combina el manejo de estados de un proceso, la asignación de turnos y el manejo de eventos; temas que estudiamos en el módulo de concurrencia.

Recuerde que el manejo de estados, turnos y eventos para un proceso es posible gracias al bloque de control de proceso (PCB – *Process Control Block*), una estructura de datos que el sistema operativo usa para guardar el estado de un proceso y así poder recuperarlo más tarde. Entre los valores almacenados en el PCB están los registros del procesador, el registro de instrucción y el contador de programa. A partir de estos valores, un proceso, que no está en el estado Ejecutando,32puede continuar en la instrucción siguiente a la última instrucción ejecutada antes de perder el procesador.

La virtualización se puede dar a diferentes niveles; mientras en los párrafos anteriores el sistema operativo virtualiza el procesador para sus propios procesos y los procesos de los usuarios, en este párrafo hablaremos de *hyperthreading* de Intel® una técnica que virtualiza un core para el sistema operativo. Un core es una unidad de procesamiento: ALU, unidad de control y registros. Un procesador, es decir una unidad física (o chip en hardware) típicamente tenía un core, sin embargo, hoy día puede tener varios cores y por eso vemos procesadores dual-core (2 cores) y quad-core (4 cores). La tecnología Hyperthreading de Intel ® adiciona un juego de registros al que usualmente tiene el core, creando la apariencia de contar con dos cores lógicos, aunque físicamente solo se cuenta con uno[5]. La Figura 2-1 ilustra la configuración.

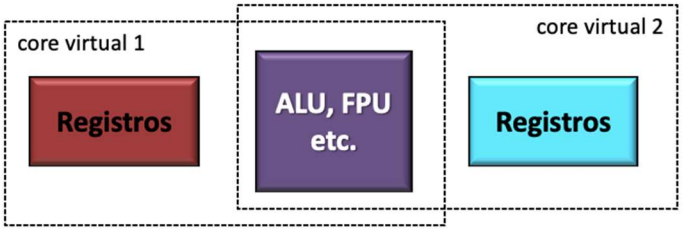

Figura 2-1. Hyperthreading de Intel adiciona un segundo juego de registros para presentar dos cores virtuales con base en un solo core físico.

Recuerde la diferencia de tiempos que existe entre que la ALU acceda a un dato en registro o que lo acceda en memoria principal. Durante la ejecución de un proceso, cuando los datos a los que acceden las instrucciones no se encuentran en los registros (o al menos en las cachés), la ALU tiene que esperar mucho tiempo a que le lleguen dichos datos desde la RAM. Durante ese tiempo de espera puede aprovechar para ejecutar otro thread siempre y cuando este segundo thread ya esté "listo", para eso le sirve el segundo conjunto de registros. La asignación de recursos al segundo thread es muy rápida porque la información de estado está disponible en el segundo juego de registros en el mismo procesador, no es necesario copiar información del PCB desde la memoria RAM. No todos los sistemas operativos han sido desarrollados para usar un segundo core; las aplicaciones pueden tomar ventaja de esta característica si son multithread y el sistema operativo puede manejar este tipo de configuración.

[5] Intel Pentium 4 3.06GHz CPU with Hyper-Threading Technology: Killing Two Birds with a Stone. http://www.xbitlabs.com/articles/cpu/display/pentium4-3066.html

Virtualización de la memoria RAM. Como en el caso del procesador, el sistema operativo administra de forma eficiente el espacio disponible en memoria física o memoria RAM, dando la apariencia de uso exclusivo para un proceso, aunque sea un recurso compartido. Incluso con un solo usuario, la memoria debe almacenar información de los procesos en ejecución del sistema operativo y los procesos del usuario. Solo cuando el tiempo de ejecución aumenta en exceso, recordamos que estamos trabajando en una máquina compartida.

El objetivo de virtualizar la memoria es independizar la memoria utilizada por los procesos de la memoria física de la máquina. Esto permite partir la memoria en fragmentos y asignar a cada proceso en ejecución uno o varios de esos fragmentos, cargar procesos en cualquier fragmento disponible de la memoria y cargar para cada proceso en ejecución solo aquellas partes que son estrictamente necesarias para su avance. La memoria virtual también permite que dos procesos compartan un mismo fragmento de memoria física cuando así lo requieran. La implementación se basa en la combinación de memoria física, cachés y almacenamiento en disco, junto con rutinas de administración eficiente de estos recursos.

Virtualización del disco. Otro esquema de virtualización se da en el caso del espacio en disco duro donde el sistema operativo crea para los usuarios la apariencia de un disco de uso fácil y exclusivo, cuando en realidad el acceso directo requeriría el manejo de identificadores únicos de sector en un disco y sincronizar el acceso de múltiples usuarios que almacenan información en el mismo disco. El disco se virtualiza por medio de dos mecanismos: por un lado, los usuarios no tienen acceso directo al recurso, deben invocar el API del sistema operativo para tener acceso al disco, y por otro lado los usuarios no conocen el nombre real del recurso de interés, conocen el nombre virtual que debe ser traducido.

Las funciones en el API del sistema operativo se encargan de resolver operaciones como `open` y `close` sobre un archivo de datos. Estas funciones generan llamadas al sistema (*system calls*) de tal forma que el sistema operativo se encarga de ejecutar la función solicitada, siempre y cuando se cumplan las condiciones necesarias, y retornar los valores necesarios al proceso que hace la invocación.

Adicionalmente, recordemos que los discos duros tradicionales están organizados en platos, pistas y sectores. Si un usuario quisiera tener acceso directo a la información almacenada necesitaría saber el plato, la pista y el sector que almacena su información. En vez de esto, el sistema operativo, por medio del sistema de archivos, ofrece a los usuarios el manejo de nombres significativos que representan los datos de interés. Nombres típicos como c:\usuarios\miusuario\contenido.txt o /usuarios/miusuario/contenido.txt son fáciles de recordar y son traducidos por el sistema al recurso real, es decir, el plato, la pista y el sector que almacena la información correspondiente.

En los capítulos siguientes desarrollaremos las ideas asociadas con la virtualización de la memoria y del disco.

2.3 Memoria Virtual

El sistema operativo debe resolver múltiples retos para manejar la memoria física como un recurso compartido y presentar al usuario la apariencia de un recurso de uso exclusivo. Hay varios problemas asociados que dieron origen al concepto de memoria virtual:

- Sin memoria virtual los programadores deben usar direcciones reales de memoria. Es un esquema difícil de manejar y poco flexible.
- Los programas que manejan direcciones reales de memoria solo pueden correr si las direcciones de memoria requeridas están disponibles. Si dos o más programas requieren una misma dirección de

memoria física, no importa si todas las demás posiciones están disponibles, solo uno de ellos podrá correr.
- Aunque hoy día los computadores cuentan con capacidad extensa en memoria física (GB), en las décadas anteriores la capacidad era mucho menor y algunos programas no cabían completamente en memoria. Pero un programa debe estar en memoria física para poder correr, ¿o no?

2.3.1 Espacio de direcciones

Para resolver los problemas planteados, los ingenieros desarrollaron el concepto de espacio de direcciones: el conjunto de direcciones que un proceso puede usar durante su ejecución. Cada proceso tiene un espacio de direcciones propio, a menos que quiera compartirlo con otro proceso.

Una de las primeras soluciones para crear un espacio de direcciones para cada proceso adicionó un registro base y un registro límite a cada proceso. En el hardware, se adicionaron dos registros (base y límite) a la CPU. Cuando un proceso se carga en memoria física, el valor base corresponde a la dirección de memoria física donde empieza la zona de memoria asignada al proceso y el valor límite corresponde a la longitud del programa. Cuando el proceso está en ejecución (tiene procesador asignado), el hardware de la CPU toma cada dirección de memoria que genere y le suma el valor del registro base. Además, compara el valor generado con el registro límite para verificar que la dirección no sale del rango permitido.

La Figura 2-2 muestra un ejemplo con dos procesos:
- Cuando el proceso 1 genera la dirección 16, la CPU automáticamente adiciona el valor en el registro base: 16 + 0 = 16, generando así la dirección correcta a la que el proceso debe saltar.
- Cuando el proceso 2 genera la dirección 32, la CPU automáticamente adicionar el valor en el registro base: 32 + 1024 = 1056, generando así la dirección correcta a la que el proceso debe saltar.

Aunque conveniente, el uso de registros base y límite para generar espacios propios de direcciones no es suficiente para manejar todos los procesos que un sistema de cómputo requiere; por ejemplo, no permite cargar al tiempo todos los procesos que el sistema requiere. Como consecuencia, se desarrolló un concepto más sofisticado: la memoria virtual.

2.3.2 Diseño de la Memoria Virtual

Uno de los objetivos de la memoria virtual, al igual que los registros base y límite, es ofrecer a cada proceso su propio espacio de direcciones. En un sistema que maneja memoria virtual, todas las direcciones generadas por un proceso son consideradas direcciones virtuales y el sistema debe traducirlas a direcciones físicas reales. El espacio virtual puede ser, y usualmente es, más grande que el espacio de direcciones físicas. Adicionalmente, la memoria virtual permite que un proceso pueda correr sin estar completamente cargado en memoria, generando así la posibilidad de correr más procesos de forma concurrente.

Como todo lo virtual, la memoria virtual es una representación, implementada en este caso con base en la memoria física, almacenamiento en disco y dos tablas conocidas como la tabla de páginas y la tabla auxiliar que permiten administrar la información que está almacenada en la memoria física y en el almacenamiento en disco. La Figura 2-3 muestra esta composición.

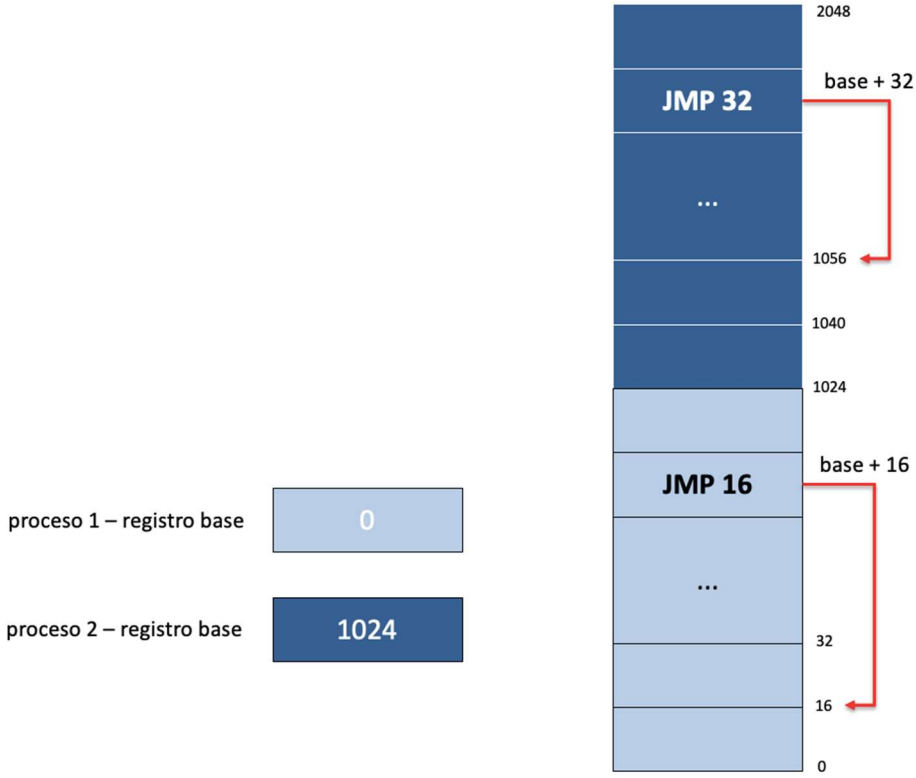

Figura 2-2. Ejemplo de uso dl registro base para generar espacios de direcciones por proceso.

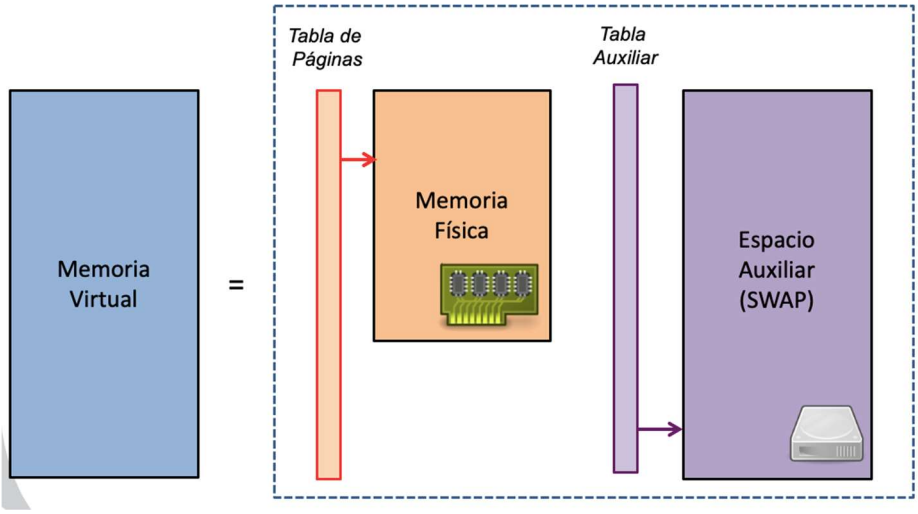

Figura 2-3. Componentes de la memoria virtual.

Para su funcionamiento, la memoria virtual divide el espacio de direcciones de un proceso en partes, llamadas páginas, y permite que un proceso corra sin tener que cargar en memoria física todas sus páginas, solo aquellas que son efectivamente necesarias para avanzar en la ejecución, las demás páginas estarán en el área de swap en

el disco duro. Si un proceso genera una dirección que no está cargada en memoria física, entonces el sistema operativo debe buscar y cargar en memoria física la página necesaria y volver a ejecutar la instrucción que falló.

A alto nivel, la traducción de una dirección virtual a real se da así: primero, un proceso en ejecución genera una dirección virtual, el sistema determina la página virtual que contiene esa dirección y consulta la tabla de páginas para traducir esa dirección virtual a una dirección en la memoria física. Si la página que contiene la dirección está en la memoria física el proceso de traducción entrega la nueva dirección y termina, si no está, entonces el sistema debe consultar la tabla auxiliar para determinar la ubicación en el área de swap y procede a copiar esa página del área de swap a la memoria física.

A continuación, describimos detalladamente el proceso de traducción de una dirección virtual a una dirección real (en memoria física). Primero revisaremos los recursos necesarios y a continuación un ejemplo.

- Páginas. El sistema divide la información asociada a un proceso (código, datos, heap, stack) en unidades de igual tamaño conocidos como páginas. Así mismo, las direcciones de memoria física también se agrupan en unidades del mismo tamaño conocidos como marcos de página. Usualmente las páginas y los marcos de página tienen el mismo tamaño.
- Tabla de páginas. La tabla de páginas indica la correspondencia entre las páginas de memoria virtual de un proceso y los marcos de página que almacenan dichas páginas. Algunas entradas en la tabla de página no tendrán valor definido dado que es posible que algunas páginas del proceso no estén cargadas en memoria física. La Figura 2-4 ilustra la estructura de la tabla de páginas: suponga que la página 0 del espacio de direcciones virtual se encuentra en el marco de página 7, entonces en la entrada 0 de la tabla de páginas aparecerá el valor 7. Como la página virtual 2 no está cargada en memoria física, la entrada correspondiente almacena un valor indefinido.
- Tabla auxiliar. La tabla auxiliar establece la correspondencia entre las páginas de memoria virtual de un proceso y el área de swap que almacena dichas páginas. Recuerde que, si una página de un proceso no se encuentra en memoria física, entonces debe estar en el área de swap.

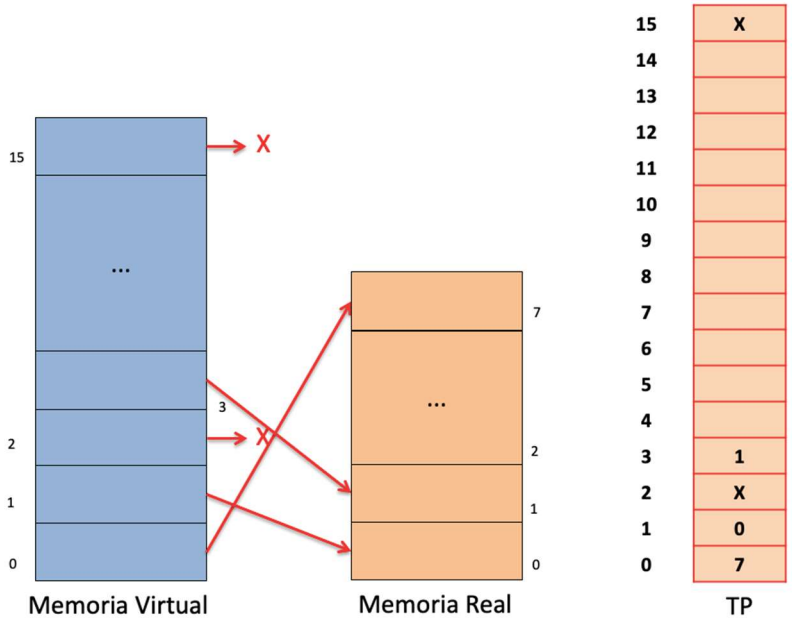

Figura 2-4. Estructura de la Tabla de Páginas

Ejemplo. La Figura 2-5 ilustra un sistema que maneja memoria virtual, páginas y marcos de página que agrupan 4 posiciones/direcciones de memoria. Nota: Por claridad usamos páginas que agrupan 4 posiciones de memoria, pero el tamaño de página usado por sistemas reales es mayor: 2 a 8 KiB.

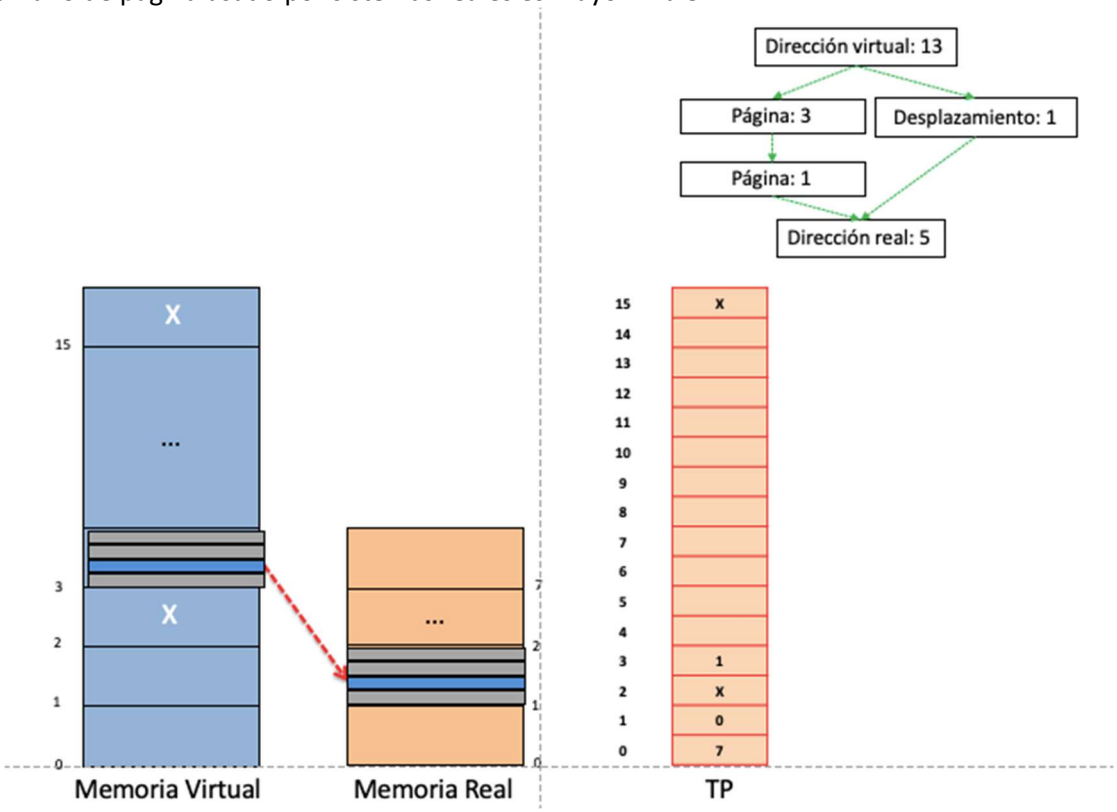

Figura 2-5. Esquema usado para traducir una dirección virtual a dirección física.

La tabla de páginas (TP) indica qué páginas virtuales están cargadas en marcos de página en la memoria física. Por ejemplo, la página virtual 0 está cargada en el marco de página 7, la página 1 en el marco de página 0, la página 3 en el marco de página 1, etc. Las páginas virtuales 2 y 15 no están cargadas en la memoria física, usamos el carácter X para representar esta situación.

La página virtual 3 permite ver las cuatro direcciones de memoria agrupadas por una página. Las direcciones agrupadas en la página 3 corresponden a las direcciones virtuales 12, 13, 14 y 15. El sistema también puede ver estas direcciones como un identificador de página y un desplazamiento: página 3, desplazamientos 0, 1, 2 y 3 respectivamente. Cuando el sistema copia la página virtual 3 en el marco de página 1, el sistema copia las direcciones de memoria agrupadas en el mismo orden. Por ejemplo, el valor almacenado en la página virtual 3 desplazamiento 1, estará en el marco de página físico 1 pero seguirá en el desplazamiento 1. La flecha roja punteada representa este ejemplo. Lo mismo pasa con todas las direcciones agrupadas en una página virtual al ser copiadas a memoria física, el desplazamiento se conserva.

Suponga que en un sistema con las características ilustradas por la figura es necesario traducir la dirección virtual 13:

- El espacio virtual del sistema tiene 16 páginas virtuales y 4 direcciones por página, así que tiene un espacio total de direcciones virtuales de 64. Como consecuencia, este sistema necesita al menos 6 bits para direccionamiento.

- Recordemos que el hardware maneja representación binaria; 13 en base 2 = 001101_2.
- Dado que tenemos páginas que agrupan 4 direcciones de memoria, sabemos que el sistema necesita 2 bits para representar los desplazamientos posibles en una página (0, 1, 2, 3). Entonces, los últimos dos bits de la dirección corresponden al desplazamiento en la página: 01_2.
- El resto de los bits corresponden a la identificación de la página virtual: 0011_2. Observe que este valor corresponde al identificador de página: 3.
- La tabla de páginas establece que la página virtual 3 (0011_2) se encuentra en el marco de página 1 (0001_2).
- El sistema divide la dirección en dos partes: identificador de página (0011_2) y desplazamiento (01_2), usa la TP reemplazar el identificador de página por el marco de página físico ($0011_2 \rightarrow 0001_2$) y conserva el desplazamiento (01_2). La dirección física es entonces: marco de página 0001_2 y desplazamiento $01_2 \rightarrow 000101_2$, valor que corresponde a 5.

Soporte en Hardware. Para mejorar el tiempo de respuesta asociado con la traducción de direcciones virtuales, los sistemas cuentan con dos componentes adicionales en hardware:

- La unidad de manejo de memoria (*Memory Managment Unit* – MMU)
- La zona de memoria caché (*Translation Lookaside Buffer* – TLB)

Las direcciones virtuales que un programa genera no van directo al bus de direcciones. Son enviadas primero a la MMU, este componente se encarga de la traducción, es decir, la separación en identificador de página y desplazamiento, traducción de página a marco de página y construcción de la dirección física que es efectivamente enviada por el bus de direcciones a la memoria física.

La TLB puede definirse con la zona de caché para la traducción de direcciones virtuales. Es una zona de memoria de acceso rápido que permite almacenar las traducciones recientes. Como cualquier caché si el valor buscado se encuentra almacenado, el tiempo de respuesta para una búsqueda se reduce considerablemente. En este caso, el TLB ofrece un tiempo de respuesta menor al tiempo de consulta en la tabla de páginas que se encuentra en memoria RAM.

Actividad 2-1: Suponga un sistema que maneja direcciones de memoria virtual de 32 bits, páginas y marcos de página que agrupan 2^{10} direcciones, y la tabla de páginas en la figura. Calcule la dirección real correspondiente a la dirección virtual 4105_{10} (use el mismo procedimiento que sigue la MMU).
Actividad 2-2: ¿Qué haría el sistema si después de generar la dirección virtual 4105_{10} el mismo proceso generara la dirección virtual 4125_{10}?
Actividad 2-3: Cómo cambia la traducción de la dirección 4105_{10} si el sistema usa páginas y marcos de página que agrupan 2^{11} direcciones

...	...
11	
10	20
9	X
8	X
7	X
6	22
5	...
4	5
3	1
2	6
1	0
0	7

Actividad 2-4: En un sistema con direcciones de memoria virtual de 32 bits, páginas y marcos de página de 4K:
(a) ¿Cuál es el tamaño máximo de la memoria virtual?
(b) ¿cuál es el tamaño de la tabla de páginas si cada entrada en la tabla ocupa 32 bits?

Actividad 2-5: Suponga que en un sistema el tiempo para leer de la memoria RAM es 5 nanosegundos. El sistema cuenta con una TLB con 32 entradas y tiempo de respuesta de 1 nanosegundo. ¿Qué tasa de éxito en las búsquedas en la TLB se requiere para tener un tiempo promedio de traducción de direcciones virtuales a direcciones reales de 2 nanosegundos?

Fallo de Página. Como vimos previamente, solamente una parte de las páginas virtuales de un proceso se cargan en memoria física. ¿Qué pasa entonces cuando un proceso genera una dirección virtual asociada a una página que no está en memoria física? Se produce un evento que llamaremos fallo de página, el sistema debe recuperar de la memoria swap la página necesaria para que el proceso que la necesita pueda continuar su ejecución.

Sin embargo, en algunos casos no hay espacio libre disponible en la memoria física. Entonces, es necesario escoger un marco de página y copiarlo al área de swap para dejar espacio libre. A continuación, se copia la página virtual requerida desde el área de swap a la memoria física y se actualizan las tablas de páginas y auxiliar. Al terminar de copiar la página requerida, el sistema operativo puede continuar la ejecución del proceso.

Dado que el tiempo de servicio de un fallo de página implica uno o dos accesos al área de swap en el disco duro, el número de fallas de página que un programa genera afecta considerablemente y de forma adversa el tiempo de respuesta. Afortunadamente, la mayoría de los programas presentan, de forma natural, un comportamiento que genera relativamente pocos fallos de página. El número promedio de veces que un programa genera y no genera fallos de página da lugar al concepto de tiempo de acceso efectivo.

Actividad 2-6: Calcule el tiempo de acceso efectivo para un sistema que tiene una probabilidad de fallo de página de 0,3, tiempo de acceso a memoria de 100 nanosegundos y tiempo promedio de servicio por falla de página de 10 milisegundos.

Sobrepaginación. En algunos casos, por el número de procesos que el sistema corre de forma concurrente para todos los usuarios, el sistema puede llegar a un estado que se conoce como sobrepaginación. En este estado, los procesos generan un número muy alto de fallos de página y como consecuencia el tiempo invertido en resolución de fallos de página es superior al tiempo que el sistema efectivamente invierte en la ejecución de las instrucciones de los procesos. Usualmente está relacionado con el grado de multiprogramación, o número de procesos que se ejecutan de forma concurrente; a mayor número de procesos en ejecución, cada proceso recibe un número menor de marcos de página y se incrementa el número de fallos de página.

Tabla de páginas multinivel. En algunos sistemas el espacio de direcciones virtuales es muy extenso dando lugar a tablas de páginas muy grandes. Al igual que con los procesos, no es necesario mantener toda la tabla de páginas en memoria física. Es deseable, pero dado que la memoria física es un recurso escaso, es posible paginar la tabla de páginas y mantener en memoria RAM solo aquellas partes que son efectivamente necesarias. La Figura 2-6 ilustra la estructura de una tabla de páginas de dos niveles.

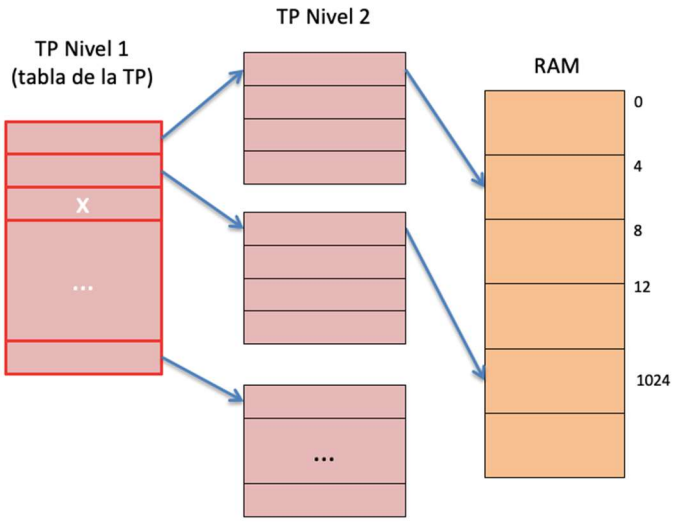

Figura 2-6. Estructura de la tabla de páginas multinivel.

Con esta configuración, el sistema administra una dirección virtual como un elemento con tres partes:

- La primera parte se usa como índice en la tabla de nivel 1 (TP1) para producir la dirección del marco de página asociado con la página en la tabla de páginas de segundo nivel (TP2).
- La segunda parte se usa como índice en la tabla de nivel 2 (TP2) para producir la dirección del marco de página asociado con la dirección de interés.
- El desplazamiento se conserva.

Como consecuencia de esta configuración, el tiempo de acceso a memoria puede aumentar porque ahora hay un acceso adicional si un proceso genera una dirección virtual que no se encuentre en la TLB: un acceso a la tabla de nivel 1 y un acceso a la tabla de nivel 2. Adicionalmente, puede haber dos fallos de página: uno por la tabla de nivel 2 y uno por la página con los datos necesarios. La tabla de nivel 1 siempre estará completa en memoria física.

El esquema de memoria virtual que acabamos de estudiar resuelve los problemas inicialmente planteados relacionados con reubicación:

- Los procesos pueden correr sin estar atados a zonas fijas de memoria; pueden ser cargados en cualquier marco de página disponible.
- El proceso está completamente cargado en memoria virtual, pero solo parcialmente en memoria física y esto no impide avanzar en su ejecución.

Sin embargo, no hemos discutido cómo resolver los problemas relacionados con protección:

- Permitir que dos procesos compartan zonas de memoria cuando lo requieren.
- Garantizar que un proceso no puede afectar la información en los marcos de página de otro proceso.

Para resolver estos dos problemas el sistema operativo maneja una tabla de páginas por proceso; solo el sistema operativo puede actualizar la información en las tablas de páginas. La Figura 2-7Figura 2-7 muestra un ejemplo: hay dos procesos y el sistema asocia una tabla de páginas independiente a cada uno. Por otro lado, dos procesos pueden compartir una zona de memoria física pidiendo al sistema que adicione el mismo marco de página en sus respectivas tablas de páginas.

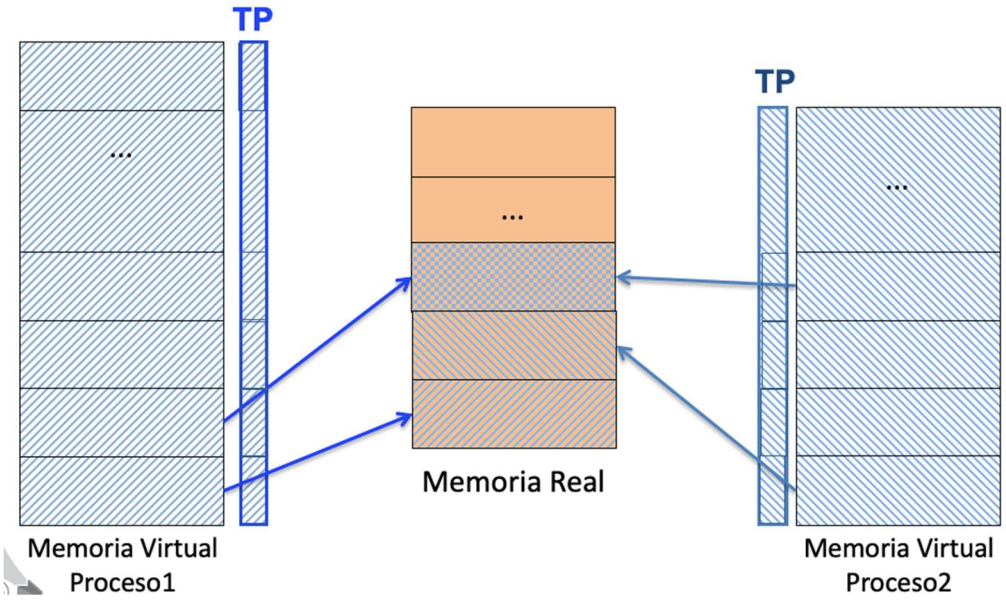

Figura 2-7. El sistema ofrece a los dos procesos la posibilidad de compartir el mismo marco de página en memoria real.

Adicionalmente, el sistema mantiene un indicador de solo lectura (L) o lectura/escritura(E) para las páginas en las tablas de páginas. Un proceso solo puede escribir una variable en una dirección de memoria si esa dirección hace parte de uno de sus marcos de página y el indicador asociado es E.

Eficiencia. Previamente se mencionó que los programas generan, de forma natural, relativamente pocos fallos de página. Este comportamiento está asociado con el principio de localidad: si una página fue referenciada recientemente, es muy probable que sea nuevamente referenciada en un futuro próximo. Los ciclos ilustran de forma intuitiva el principio: las referencias que genera un proceso que está ejecutando un ciclo corresponden a un subconjunto reducido de páginas que almacenan las instrucciones y datos del ciclo. Se genera un fallo de página la primera vez que el proceso genera la referencia a las instrucciones y datos del ciclo, la siguiente vez (o siguientes veces) que se genere la referencia, no se genera fallo de página porque las instrucciones y datos ya han sido cargados en memoria física.

Algoritmos de remplazo. Si se genera un fallo de página y no hay espacio libre en memoria física, el sistema operativo debe seleccionar una de las páginas ya cargadas en memoria física para enviarla al área de swap y así dejar espacio disponible para copiar la página que se necesita. La página ideal para mover de memoria física al área de swap es aquella que no se vuelve a necesitar o que se necesita de forma más lejana en el futuro. Como el sistema no puede predecir este comportamiento, hay diferentes algoritmos para seleccionar la página que será enviada al área de swap, FIFO (*First in First out*), LRU (*Least recently used*) y bit de referencia y cambio, entre otros. Aunque estos algoritmos usan heurísticas diferentes, todos intentan predecir con base en información conocida cuál es la página ideal para reemplazar.

Actividad 2-7: Compare el número de fallos de página que generan los algoritmos FIFO y LRU para un proceso que genera la siguiente secuencia de referencias a páginas virtuales y cuenta con 4 marcos de página (suponga que inicialmente no hay ninguna página en memoria real y cuente la primera referencia).
1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

Espacio de trabajo. Para todo proceso existe un número x, conocido como espacio de trabajo, tal que si a ese proceso se le asignan menos marcos de página (< x), el número de fallas de página crece considerablemente. Si el sistema no puede asignar un número mínimo de marcos de página a un proceso, entonces es mejor no admitir procesos adicionales para su ejecución, hacerlo contribuye a llevar al sistema a un estado de sobrepaginación.

Calcular el tamaño ideal para el espacio de trabajo de cada proceso requeriría correr el proceso y monitorearlo para determinar cuántos marcos de página requiere, lo cual no es práctico. En vez de esto, se usa una aproximación que monitorea el comportamiento de un proceso durante una ventana de observación. Esta aproximación usa un parámetro delta que define el tamaño de la ventana de observación: se observa el conjunto de referencias generadas durante esa ventana. Para ser más precisos: se observa el conjunto de referencias generadas por un proceso, en las delta referencias más recientes. Observe que este conjunto representa una aproximación de la localidad del programa. Por ejemplo, si un proceso genera las referencias 2,6,1,5,7,7,7,7,5,1,6,2,3,4 con un delta de 10 tendríamos espacios de trabajo de tamaños: 5, 4, 5, 6, 7 (el número de referencias diferentes cambia a medida que la ventana avanza). Con base en delta 10, el tamaño óptimo del espacio de trabajo (el tamaño que garantiza menos fallas de página) sería 7.

Actividad 2-8: Suponga que se tiene un programa que genera referencias a las páginas indicadas, en el siguiente orden: 1,2,3,4,5,3,4,1,6,7,8,7,8,9,7,8,9,5,4,2. ¿Cuál es el tamaño de su espacio de trabajo (con deltas=4 y 6)? Justifique su respuesta.

2.4 Sistema de Archivos

Esta sección presenta una introducción al sistema de archivos, el componente del sistema operativo que se encarga de almacenar y recuperar información para usuarios y procesos a largo plazo, es decir, sin importar que un usuario termine su sesión, que un proceso termine su ejecución, o que la máquina se apague. Esta sección se basa en dos fuentes principales: Sistemas Operativos Modernos de Andrew Tanenbaum y Conceptos de Sistemas Operativos de Abraham Silberschatz, Peter Galvin and Greg Gagne.

La sección está organizada así: la primera parte presenta la introducción, las partes 2, 3 y 4 presentan los tres problemas principales que el sistema de archivos debe resolver para almacenar y recuperar información a largo

plazo y presentarla de forma adecuada: presentación de información a los usuarios, administración de espacio asignado y administración de espacio libre. La última sección presenta los conceptos estudiados a lo largo de la sección en el contexto del sistema operativo Linux.

2.4.1 Introducción

Antes de estudiar cada uno de los problemas mencionados revisaremos cómo se almacena la información del sistema de archivos en el disco. El primer sector del disco, el sector 0, almacena el MBR (*Master Boot Record*) registro que almacena información para el arranque de un computador. A continuación, se encuentra la tabla de particiones, una estructura que guarda información sobre las particiones que existen en el disco y que además puede guardar cuál es la partición activa por defecto.

Cada partición en el disco almacena un bloque de arranque, un sistema operativo y el sistema de archivos correspondiente. El bloque de arranque tiene instrucciones para cargar el sistema operativo asociado, a continuación, está el código del sistema operativo y después el sistema de archivos que almacena información sobre los archivos existentes en el sistema, espacio asignado y espacio libre. La Figura 2-8 presenta el esquema de la organización del disco.

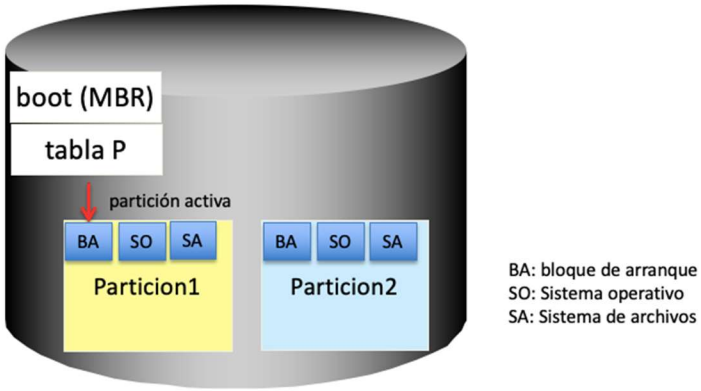

Figura 2-8. Organización del disco para almacenar información del sistema de archivos.

2.4.2 Presentación a los Usuarios

El sistema de archivos es posiblemente el componente del sistema operativo más conocido para todos los usuarios, con y sin formación en computación. Todos entienden intuitivamente el concepto de archivo y directorio. Un archivo es la unidad de almacenamiento de información, lo usan para almacenar y más tarde recuperar información, mientras los directorios permiten almacenar la información de forma jerárquica agrupando diferentes archivos bajo un concepto lógico.

La Figura 2-9 ilustra la organización de la información en un esquema jerárquico. Hay un directorio raíz y a partir de este directorio raíz es posible crear un árbol que permite almacenar la información. En cada directorio hay subdirectorios o carpetas que ayudan a organizar la información.

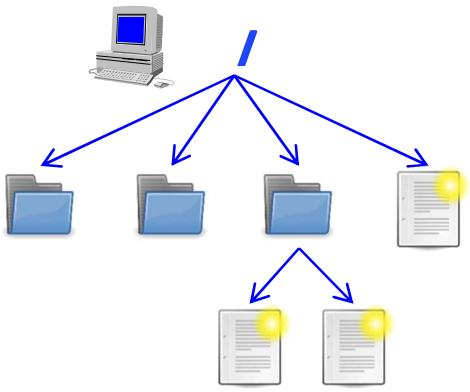

Figura 2-9. Estructura jerárquica para almacenar un archivo.

Un directorio puede incluir archivos almacenados remotamente, en otras máquinas con las que existe conexión por medio de una red. El sistema debe implementar un servicio para soportar este esquema, por ejemplo, Unix – NFS (Network File System) y Windows – SMB (Server Message Block), también conocido como CIFS (Common Internet File System). Hoy día, además de los servicios tradicionales como los mencionados, hay servicios soportados con tecnología cloud como onedrive y dropbox.

El manejo de archivos remotos requiere la presencia de un cliente y un servidor que implementen un protocolo de comunicación para intercambiar información sobre el archivo: montaje del archivo, actualización y consulta. La Figura 2-10 ilustra dicho protocolo de comunicación: en la máquina de la izquierda el archivo /b/e no es local, es una referencia a un archivo almacenado de forma remota. El cliente local maneja las operaciones sobre este archivo y las envía al servidor, la máquina de la derecha, que efectivamente almacena el archivo e.

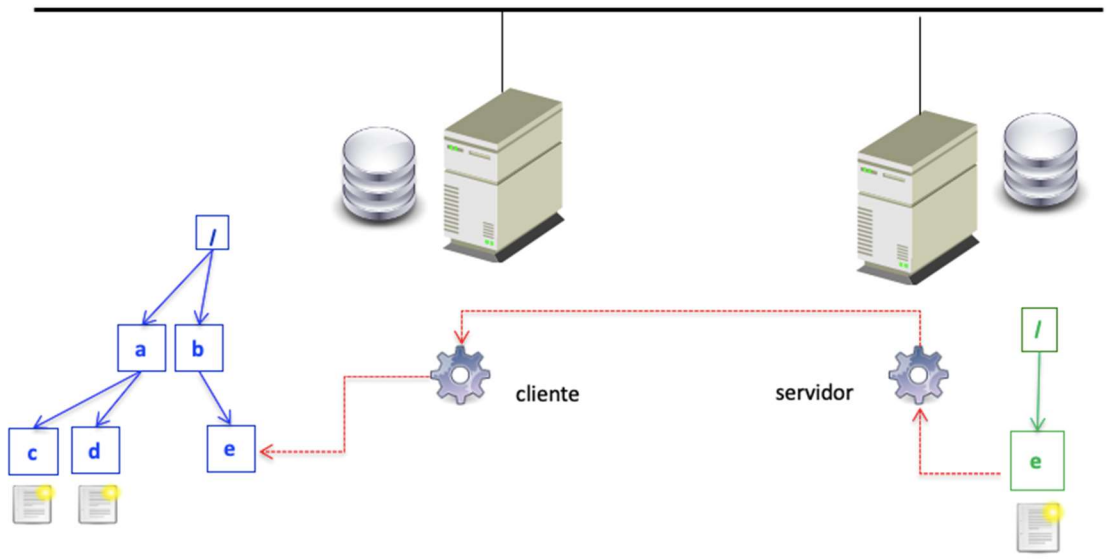

Figura 2-10. Esquema del protocolo de comunicación entre un cliente y un servidor para manejo de un archivo remoto.

Actividad 2-9: Busque información sobre los servicios NFS y SMB para manejo remoto de archivos.

2.4.3 Administración de Espacio Asignado

Para los usuarios la presentación jerárquica de archivos y directorios satisface sus necesidades; la representación subyacente es transparente. Sin embargo, para nosotros es importante entender cómo administra el sistema operativo el espacio asignado a los archivos. Esta sección y la siguiente presentan los conceptos básicos asociados con esta tarea de administración.

Representación de un Archivo. El sistema de archivos no solo maneja el contenido de los archivos, además maneja metadatos para facilitar la búsqueda del contenido y la ejecución de operaciones sobre un archivo. Los metadatos de un archivo se almacenan en una estructura de datos conocida como un descriptor. Cuando un proceso o un usuario abre un archivo, el sistema operativo debe buscar el descriptor del archivo involucrado y copiarlo en memoria RAM. Las operaciones sobre el archivo tendrán un mejor tiempo de respuesta al tener el descriptor en memoria dado que allí está toda la información necesaria para ejecutar operaciones sobre el archivo. La Figura 2-11 muestra la estructura de un descriptor.

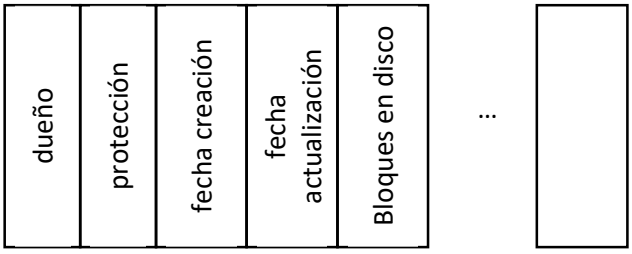

Figura 2-11. Estructura de un descriptor de archivo.

Un descriptor incluye datos como dueño del archivo, permisos de acceso, fecha de creación y la lista de bloques en disco que almacenan el contenido del archivo.

Tamaño de Bloque. Para administrar el espacio asignado a los archivos, el sistema operativo organiza el espacio disponible en disco en bloques. El sistema podría manejar bloques de tamaño variable que agrupen sectores contiguos en disco o bloques de tamaño fijo que no necesariamente están contiguos. La principal condición que los diseñadores de sistemas de archivos tienen en cuenta para tomar una decisión al respecto es que los archivos tienen tamaños variables; algunos archivos pueden ser muy pequeños, mientras otros pueden ser muy grandes. Además, los archivos cambian de tamaño con frecuencia. En este contexto la segunda opción es más flexible, con un solo bloque de sectores contiguos la disminución en tamaño del archivo dejaría fragmentos libres que podrían ser muy pequeños para usar y el aumento en tamaño obligaría al movimiento del archivo completo a otra zona del disco. De otra parte, manejar varios bloques de tamaño fijo permite asignar más bloques si el archivo crece en tamaño y liberar bloques si el archivo disminuye, facilitando el manejo de espacio y reduciendo el número de operaciones requeridas para la asignación de espacio.

Los diseñadores deben tomar una decisión adicional: el tamaño del bloque. Este tamaño debe ser múltiplo del tamaño de sector del disco subyacente y debe ser estratégico. No puede ser muy grande porque los archivos pequeños desperdiciarían mucho espacio y tampoco puede ser muy pequeño porque buscar el contenido de un

archivo implicaría múltiples lecturas, una por cada bloque, aumentando el tiempo de respuesta y reduciendo el desempeño.

Registro de Bloques Asignados. Aparece entonces otro problema que debe ser resuelto. ¿Cómo representar el espacio asignado a un archivo cuando hay varios bloques? Hay diferentes opciones entre las cuales queremos mencionar lista enlazada y nodos-i.

- En una representación con lista enlazada, el descriptor de archivo guarda un apuntador al primer bloque del archivo y cada uno de los bloques siguientes guarda un apuntador al siguiente bloque. Esta estructura no es conveniente para implementar acceso aleatorio a cualquier parte del archivo porque siempre será necesario recorrer los primeros bloques para llegar al bloque buscado. La Figura 2-12Figura 2-12 ilustra la estructura: el identificador en la primera línea corresponde al bloque físico, el número de bloque dentro de cada cuadrado corresponde a la posición del bloque como parte del archivo A.

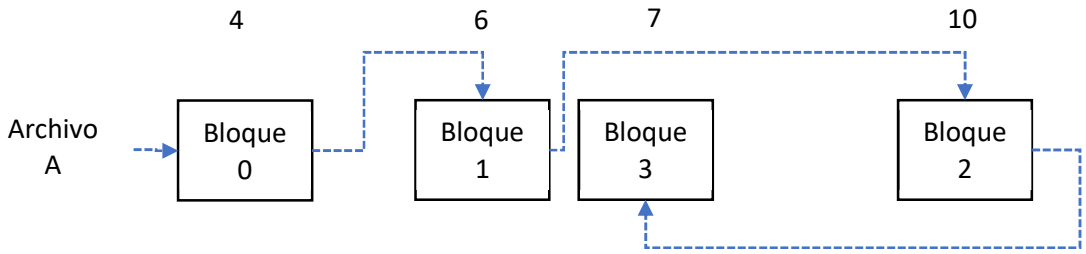

Figura 2-12. Representación de los bloques asignados a un archivo por medio de lista enlazada.

- En una representación con nodo-i (nodo índice) el descriptor guarda una tabla. La primera posición de la tabla apunta al primer bloque del archivo, la segunda posición apunta al segundo bloque y así sucesivamente. La última posición en la tabla apunta a un bloque que no se usa para almacenar contenido, se usa para almacenar apuntadores a otros bloques que sí almacenan contenido. Cada una de las posiciones, en particular la última, se usan o se dejan libres dependiendo del tamaño del archivo. Esta estructura responde mejor que la estructura de lista enlazada al acceso aleatorio a cualquier parte del archivo. La Figura 2-13ilustra la estructura.

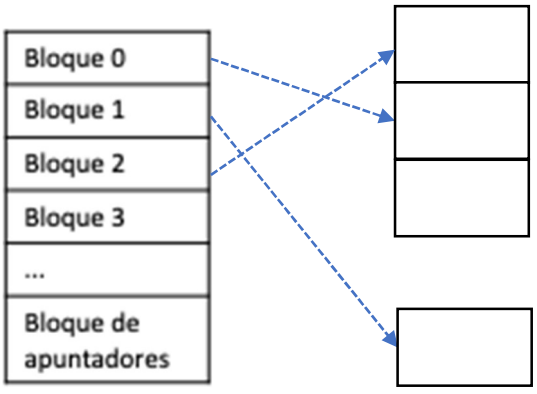

Figura 2-13. Representación de los bloques asignados a un archivo por medio de nodos-i.

Actividad 2-10: El espacio libre en disco se puede contabilizar mediante el uso de una lista de bloques libres o un mapa de bits. Suponga que las direcciones de disco requieren D bits. Para un disco con B bloques, F de los cuales son libres, indique la condición bajo la cual la lista de bloques libres utiliza menos espacio que el mapa de bits. Si D es 16 bits, exprese su respuesta como un porcentaje del espacio en el disco que debe estar libre. [Tomado de Tanenbaum]

Actividad 2-11: Una manera de utilizar la asignación contigua del disco y no sufrir de huecos es compactar el disco cada vez que se elimina un archivo. Como todos los archivos son contiguos, para copiar un archivo se requiere una búsqueda y un retraso rotacional para leerlo, seguidos de la transferencia. Para escribir el archivo se requiere el mismo trabajo. Suponiendo un tiempo de búsqueda de 5 ms., un retraso rotacional de 4 ms., una velocidad de transferencia de 8 MB/s y un tamaño promedio de archivo de 8 KB. [Tomado de Tanenbaum]
(a) ¿Cuánto tiempo se requiere para leer un archivo en la memoria principal y luego escribirlo en el disco en una nueva ubicación?
(b) Utilizando estos números ¿Cuánto tiempo se requeriría para compactar un disco de 16 GB?

Optimización. Por otro lado, la ubicación de los bloques asignados a un archivo en un disco duro tradicional puede afectar el tiempo de búsqueda del contenido. En un disco duro tradicional el movimiento de la cabeza de lectura domina el tiempo de búsqueda del sector y el tiempo de transferencia, como consecuencia, si los bloques están dispersos en muchas pistas diferentes el desempeño es pobre. La Figura 2-14 muestra la diferencia entre una asignación aleatoria y una asignación óptima.

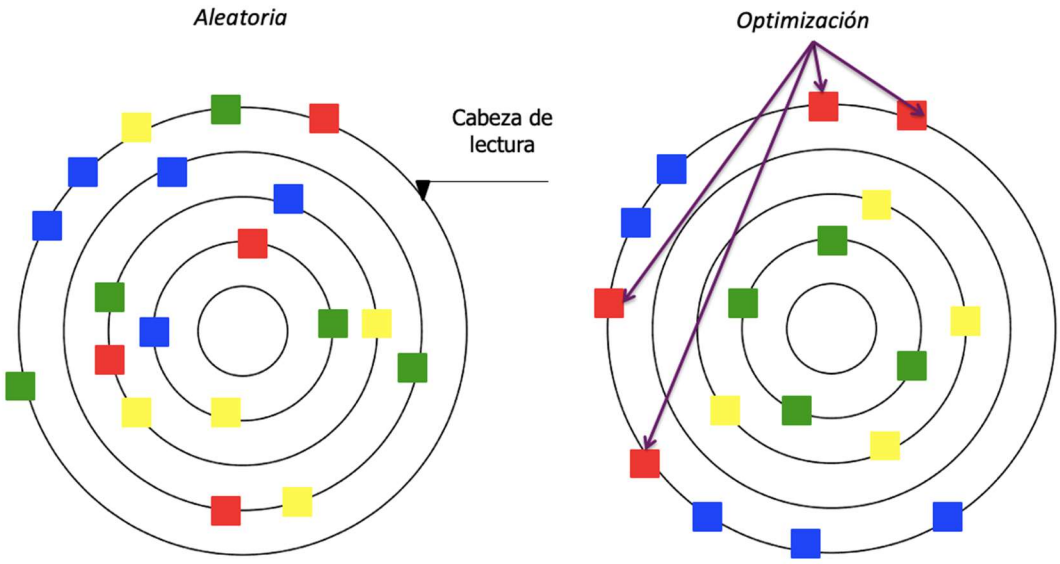

Figura 2-14. Asignación de bloques de un archivo.

Los discos de estado sólido tienen una estructura diferente y un comportamiento diferente, y como consecuencia no presentan este problema.

Para reducir el aumento en el tiempo de respuesta por datos dispersos en el disco, podemos adelantar la desfragmentación del mismo. Esta tarea busca reubicar el contenido de los archivos en bloques contiguos o al

menos en bloques sobre las mismas pistas. La desfragmentación toma tiempo y espacio dado que requiere la copia de bloques a diferentes partes del disco.

Otro aspecto relacionado con la asignación de bloques en disco es el servicio para atender pedidos de un disco. Es posible optimizar la respuesta usando políticas diferentes a un esquema FIFO *(First In*, *First served*) para reducir el movimiento de la cabeza de lectura. La primera posibilidad puede ser atender primero aquellos pedidos que corresponden a los bloques que están más cerca de la cabeza, pero esta política conocida como *shortest seek first* puede conducir a inanición. Una alternativa es la política del elevador, la cabeza se mueve en la misma dirección hasta llegar a la última pista en esa dirección y luego cambia de dirección. Hoy día los controladores de disco implementan optimizaciones adicionales porque cuentan con zonas de caché que les permiten almacenar información de bloques que pueden no haber sido solicitados.

Representación de Directorios. Así como el sistema de archivos debe manejar metadatos para los archivos, también debe manejar metadatos para los directorios. Un directorio debe registrar de alguna manera la lista de archivos y subdirectorios que agrupa en una jerarquía. Esta lista puede ser almacenada en tablas o en archivos, sin embargo, las tablas tienen tamaño fijo y la información de los directorios es variable, dado que pueden almacenar uno o muchos archivos y subdirectorios. Como consecuencia, los archivos son manejados como archivos, pero son archivos 'especiales'.

Un directorio guarda la información de los archivos y subdirectorios que agrupa por medio de sus descriptores. Sin embargo, el esquema de almacenamiento de los descriptores puede variar. La Figura 2-15 presenta dos esquemas diferentes: el esquema de la izquierda guarda directamente los descriptores, mientras el esquema de la derecha guarda apuntadores a los descriptores, este último es más eficiente en uso de espacio.

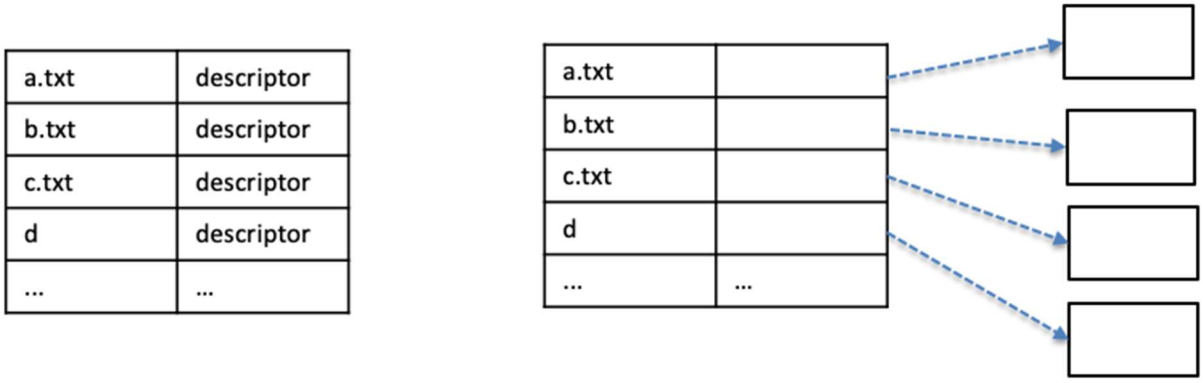

Figura 2-15. Estructura usada por un directorio para almacenar información sobre sus archivos y subdirectorios.

2.4.4 Administración de Espacio Libre

El sistema operativo debe mantener una lista de bloques libres. Cuando un usuario o proceso solicita la creación de un archivo, el sistema puede tomar bloques de dicha lista y asignarlos, también puede adicionar bloques cuando un archivo es eliminado.

Como en los múltiples casos de decisiones de diseño presentadas con anterioridad, es posible manejar diferentes esquemas. La Figura 2-16Figura 2-16 muestra dos esquemas diferentes de manejo: lista enlazada de bloques (a la izquierda) y mapa de bits (a la derecha). En la lista enlazada se registran los identificadores de los bloques libres y a medida que se van asignando el apuntador se actualiza. En el mapa de bits se tiene un registro de

ocupado o libre por cada uno de los bloques en el disco. Por ejemplo, al observar la primera fila del mapa podemos ver que el bloque 0 y el 1 están ocupados (están marcados con el bit 1), mientras el bloque 2 está libre (está marcado con el bit 0).

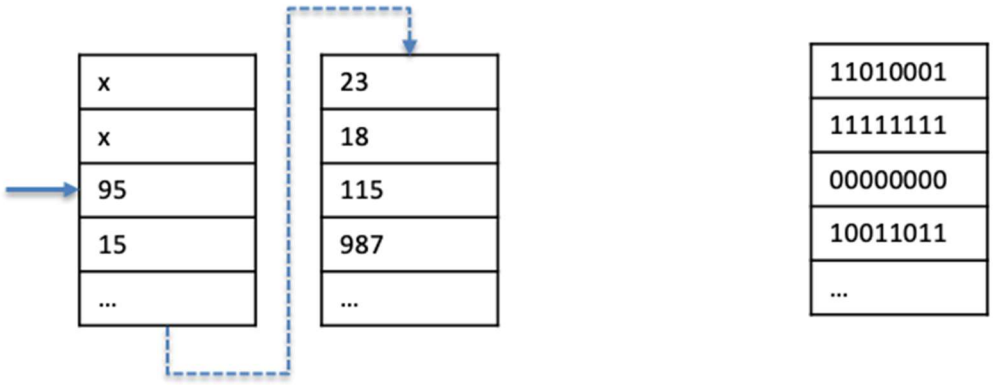

Figura 2-16. Registro de bloques libres en el sistema de archivos.

Protección. El descriptor de un archivo guarda el propietario del archivo y guarda la lista de control de acceso. Este esquema responde parcialmente a requerimientos de seguridad de la información garantizando que solamente aquellos con autorización pueden leer, modificar o ejecutar un archivo. Este aspecto del sistema de archivos será abordado cuando estudiemos el tema de seguridad.

2.4.5 Ejemplo - Linux

En los sistemas Linux el tamaño de bloque por defecto es 1024 bytes (1KiB)[6], sin embargo, este valor puede cambiar; algunos sistemas actuales usan 4KiB para adaptarse mejor al aumento actual en tamaño de archivo y en tamaño de disco.

Para el registro de bloques asignados a un archivo, Linux y todos los sistemas operativos basados en Unix usan una estructura basada en nodos-i. La Figura 2-17Figura 2-17 presenta la estructura. La tabla tiene 13 posiciones, las primeras 10 apuntan a bloques que almacenan contenido del archivo. La posición 11 no apunta a un bloque con contenido, en vez de esto, apunta a un bloque que se usa para almacenar apuntadores a bloques con contenido, en la figura este bloque aparece como bloque indirecto simple. La posición 12 almacena un bloque indirecto doble y la posición 13 almacena un bloque indirecto triple.

[6] https://www.gnu.org/software/coreutils/manual/html_node/Block-size.html

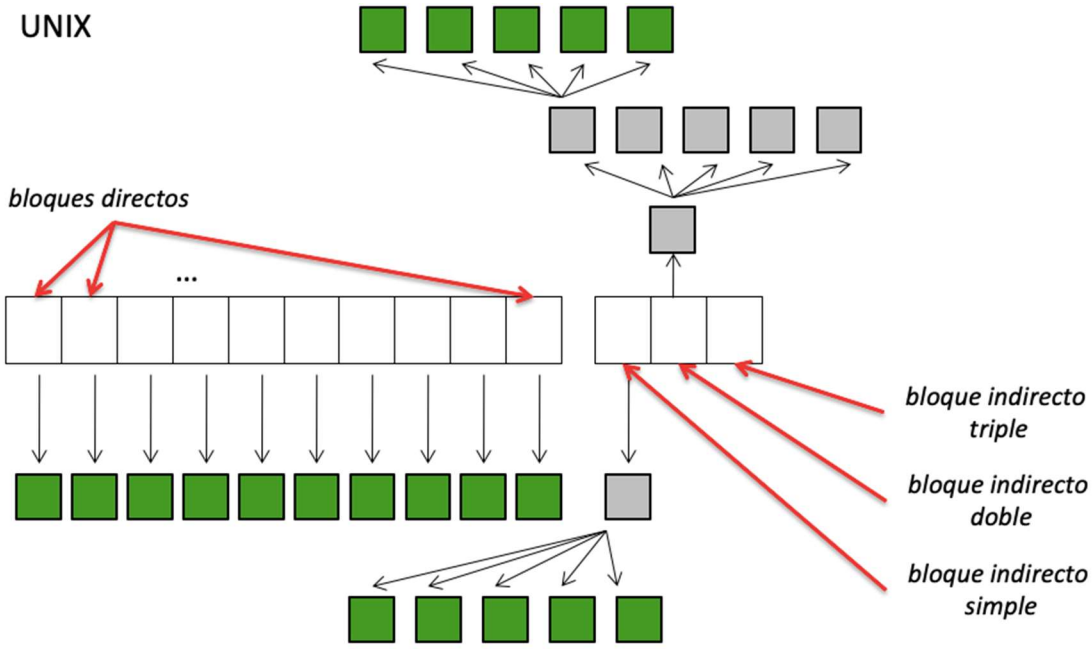

Figura 2-17. Representación Unix y Linux de los bloques asignados a un archivo.

Actividad 2-12:
(a) Cuál es el tamaño más grande de archivo que se puede manejar en un sistema de archivos que maneja representación con nodos-i con las siguientes características: Tabla de 8 entradas, la última entrada es un bloque de apuntadores, Bloques de 1KiB, Identificadores de bloque (direcciones) de 32 bits.
(b) Cuál es el tamaño más grande de archivo que se puede manejar en un sistema de archivos que maneja representación con nodos-i que contienen 10 direcciones directas de 4 bytes cada una y todos los bloques de disco son de 1024 KiB. [Tomado de Tanenbaum]
(c) Cuál es el tamaño del archivo más grande en Unix si los bloques son de tamaño 4KiB y los bloques del disco se direccionan con 4 bytes.

Representación de Archivos y Directorios. El sistema de archivos Linux usa los nodos-i para manejar los descriptores de los archivos y directorios. Todos los nodos se encuentran en una tabla, la tabla de nodos-i. Un directorio es un archivo especial que almacena las referencias a los nodos-i de sus archivos y subdirectorios.

La Figura 2-18 muestra un ejemplo: la tabla de la izquierda representa la tabla con todos los descriptores del sistema, cada índice es un entero que representa la posición en la tabla-i. El archivo de la derecha muestra la representación del directorio es un archivo con la lista de sus archivos: *a* y *d*, y por cada uno guarda el índice del descriptor. La flecha muestra explícitamente la relación entre el índice que se guarda en el directorio y la posición del descriptor en la tabla de nodos-i.

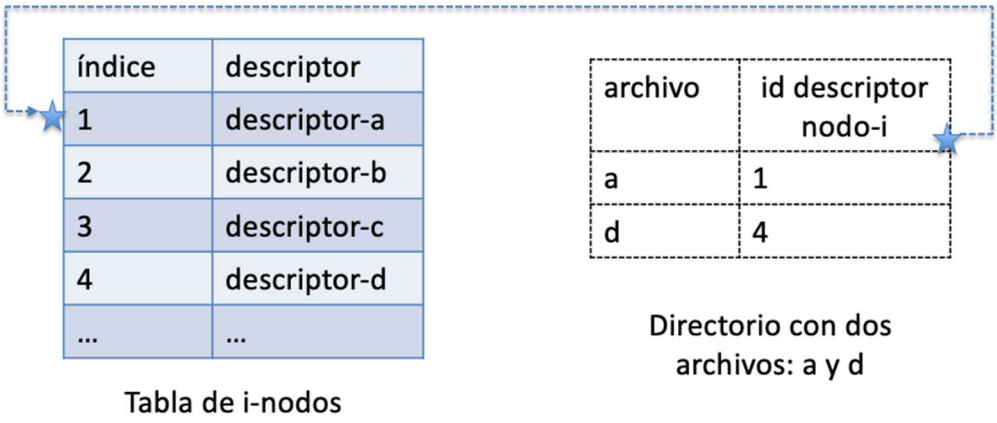

Figura 2-18. Tabla Linux de nodos-i.

Observe que para abrir el archivo *d* a partir del directorio en el que se encuentra es necesario: leer el archivo especial que almacena el contenido del directorio en forma secuencial, buscar la entrada correspondiente al archivo *d* para obtener el identificador del nodo-i, ir a la tabla general y buscar la posición asociada con el identificador del nodo-i y obtener el descriptor del archivo. Con el descriptor del archivo es posible pedir al controlador del disco los bloques asignados al archivo.

Actividad 2-13: Describa paso a paso el procedimiento que sigue un sistema Unix para abrir un archivo.

2.5 Máquinas Virtuales

En un enfoque tradicional sobre un servidor o en un computador físico, se instala y se ejecuta un sistema operativo sobre el hardware para ejecutar aplicaciones encima de este.

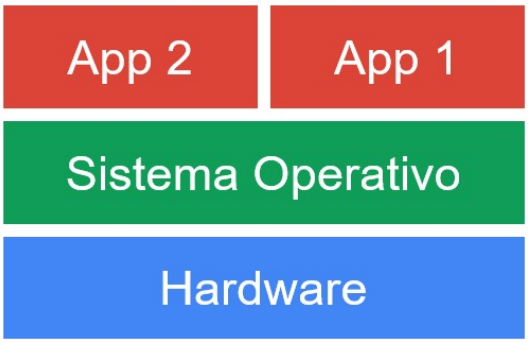

Figura 2-19. Computo tradicional

La virtualización es una tecnología que crea recursos lógicos y los asigna a recursos físicos. Este proceso se puede realizar utilizando una funcionalidad de hardware específica o a través de una capa de software llamada manejador, monitor o hipervisor. El objetivo de la virtualización es desacoplar el hardware del software para dividir y compartir recursos entre varias cargas de trabajo; máquinas virtuales y/o contenedores que ejecutan sistemas operativos y aplicaciones sobre ellas.

La idea de usar virtualización como una herramienta que permite ofrecer múltiples máquinas virtuales ejecutándose sobre una única máquina física ha sido usada por varias décadas [Goldberg & Popek, 1974]. El uso de máquinas virtuales tiene varias ventajas, entre ellas podemos mencionar:

- Una organización puede correr múltiples servicios en máquinas virtuales independientes, sin incurrir en los costos asociados con múltiples máquinas físicas. Por ejemplo, una organización puede correr servidor web, servidor de correo, LMS (*Learning Management System*), etc., en una sola máquina física y sin que se generen incompatibilidades entre los diferentes servicios.

- Contar con máquinas independientes permite aislar los problemas y continuar parcialmente con la operación. Si un servidor virtual falla, los otros servidores que corren en máquinas virtuales independientes, no se verán afectados.

- Es posible crear ambientes personalizados de manera rápida y flexible. Un usuario o administrador puede crear fácilmente un ambiente con plataformas y librerías específicas de acuerdo con sus necesidades. La organización no debe comprar un servidor adicional para hacer pruebas o correr un ambiente personalizado.

- No todas las aplicaciones aprovechan la escalabilidad vertical de una máquina: agregar CPU, memoria u otros recursos a una máquina puede terminar en recursos que son subutilizados. La virtualización permite segregar estos recursos y que una aplicación reciba solo aquellos que puede utilizar eficientemente.

Nota: Existen escenarios en los que no es posible utilizar las tecnologías de virtualización. Por ejemplo, aplicaciones que se ejecutan en un sistema operativo que no está en la lista de sistemas operativos soportados por el proveedor de software de virtualización.

2.5.1 Máquina Virtual

Este ambiente es posible gracias a un componente conocido como manejador (o monitor) de máquina virtual (MMV). La Figura 2-20 muestra la organización de alto nivel del ambiente. Las máquinas virtuales requieren la mediación del MMV para poder usar los recursos y el MMV controla el acceso. El MMV es el programa encargado de construir y administrar el ambiente para las máquinas virtuales, cada máquina debe ejecutar las aplicaciones como si estas corrieran en la máquina física, con un incremento mínimo en el tiempo de respuesta. Para ofrecer este ambiente, el MMV debe tener control total de los recursos, de manera similar al sistema operativo en un ambiente tradicional.

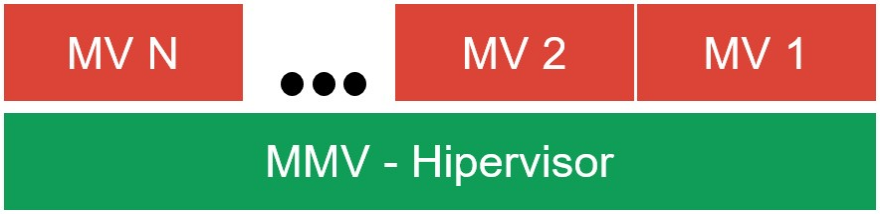

Figura 2-20. Componentes de alto nivel de un ambiente de máquinas virtuales.

El MMV es responsable de asegurar las diferentes tareas de gestión de la virtualización, como proporcionar hardware virtual, gestión del ciclo de vida de la máquina virtual, migrar máquinas virtuales, asignar recursos en tiempo real, definir políticas para la gestión de la máquina virtual, etc. También es responsable de controlar de manera eficiente los recursos de la plataforma física, como la traducción de memoria y el mapeo de Entrada/Salida.

Cada ambiente de ejecución es conocido como máquina virtual huésped (guest), mientras el servidor sobre el que corren las máquinas virtuales se conoce como anfitrión (host). El MMV corre en el host.

El MMV es responsable de asignar los recursos solicitados por las máquinas virtuales huéspedes. El hardware del sistema como el procesador, la memoria, etc., debe asignarse a estas máquinas virtuales de acuerdo con su configuración, y el MMV se encarga de esta tarea.

Entonces, si tiene un servidor con 8 núcleos en CPU y 16 GiB en RAM que ejecuta un manejador (hipervisor), puede crear fácilmente una o varias máquinas virtuales con 2 núcleos y 2 GiB en RAM cada una e iniciarlas. Los límites con respecto a la cantidad de máquinas virtuales que puede iniciar es algo que se basa en los límites físicos del hardware real y en las restricciones definidas por el manejador (típicamente asociadas a buenas prácticas y licenciamiento), pero no necesariamente es una correspondencia uno a uno.

Los MMV se clasifican principalmente como manejadores o hipervisores de tipo 1 o de tipo 2, según el lugar en el que residen en el sistema o, en otros términos, si el sistema operativo subyacente está presente en el sistema o no.

Tipo 1 o bare metal hypervisor. El MMV está a cargo de todos los recursos y corre directamente sobre la máquina (de allí el nombre de hipervisor tipo metal desnudo). Este tipo de MMV interactúa directamente con el hardware del sistema; no necesita ningún sistema operativo host. VMware ESXi/vSphere y Red Hat Enterprise Virtualization Hypervisor (RHEV-H) son ejemplos de este tipo de MMV.

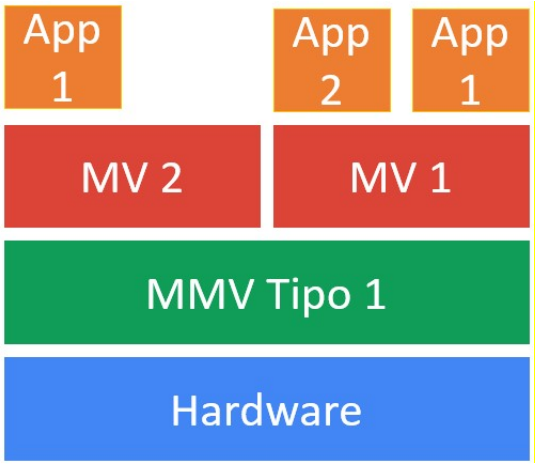

Figura 2-21 Tipo 1 o Bare Metal Hypervisor

Los MMV tipo 1 son fáciles de instalar/configurar y de tamaño pequeño para entregar la mayor cantidad de los recursos a las máquinas virtuales huéspedes. Consumen pocos recursos de máquina (CPU, Memoria y Disco), ya que solo incluyen las aplicaciones necesarias para ejecutar máquinas virtuales.

Tipo 2 o hosted. Un MMV de tipo 2 se ejecuta sobre un sistema operativo como Windows o Linux, lo que le permite realizar numerosas personalizaciones. Como consecuencia, depende del sistema operativo para ejecutar las operaciones. Los MMV de tipo 2 también se conocen como hipervisores alojados ya que dependen del sistema operativo del host para sus operaciones.

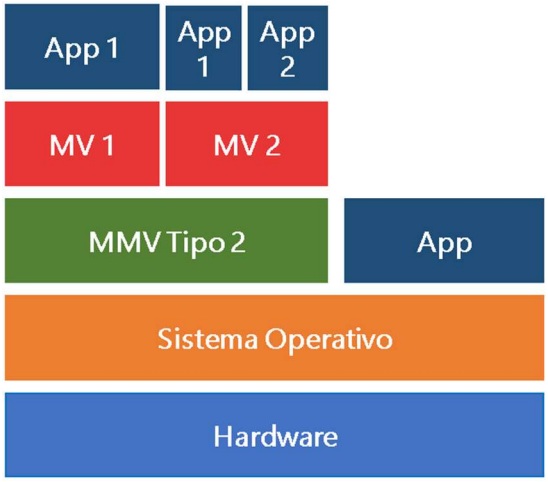

Figura 2-22 Tipo 2 o Hosted

La principal ventaja de los hipervisores de tipo 2 es la amplia gama de soporte de hardware, porque el sistema operativo del host subyacente controla el acceso al hardware. Depende principalmente de si ya tenemos un SO ejecutándose en un servidor donde queremos implementar máquinas virtuales. Los MMV de tipo 2 más conocidos incluyen VMware Player, Workstation, Fusion y Oracle VirtualBox.

Aunque el propósito y los objetivos de los MMV de tipo 1 y tipo 2 son idénticos, la presencia de un sistema operativo en el tipo 2 introduce una latencia; todas las actividades del MMV y el trabajo de cada máquina virtual deben pasar por el sistema operativo de la máquina real. Es importante resaltar que una falla de seguridad o vulnerabilidad en el sistema operativo de la máquina real podría comprometer potencialmente todas las máquinas virtuales que se ejecutan por encima de él.

Seleccionar el tipo de MMV depende estrictamente de los requerimientos de negocio definidos en un caso específico. Sin embargo, es importante resaltar que los MMV tipo 2 son comúnmente utilizados en ambientes de aprendizaje en tecnologías de virtualización, ambientes de desarrollo y pruebas. Los MMV de tipo 2 generalmente no se usan en un centro de datos y están reservados para sistemas de clientes o usuarios finales, donde el rendimiento y la seguridad no son críticas.

Los MMV tipo 1 son comunes en ambientes de producción donde se requiere una capa de virtualización robusta y los costos asociados a la implementación y el esquema de licenciamiento no son una gran restricción. El esquema o modelo de licenciamiento de estos MMV debe analizarse con detalle al seleccionar a un fabricante y un producto, dado que estos modelos incluyen licenciamiento por servidor, por CPU y por núcleos.

Los MMV tipo 1 al ejecutarse directamente en hardware físico también son muy seguros. La virtualización mitiga el riesgo de ataques que apuntan a fallas de seguridad y vulnerabilidades en los sistemas operativos. Esto asegura que un ataque a una máquina virtual esté lógicamente aislado y no se pueda propagar a otras que se ejecuten en el mismo hardware.

Otra característica importante de los MMV de tipo 1 es la facilidad para escalar las cargas de trabajo en varios terabytes de RAM y cientos de núcleos de CPU. Además, brindan soporte para almacenamiento y redes definidos por software, lo que crea seguridad y portabilidad adicionales para cargas de trabajo virtualizadas, pero también incrementan los costos de licenciamiento e implementación.

Para estandarizar la selección entre un MMV tipo 1 y un tipo 2, usted debe considerar el tipo y tamaño de sus cargas de trabajo. Si su carga de trabajo corresponde a la de una organización grande y deben implementar cientos de máquinas virtuales, un MMV de tipo 1 se adaptará a sus requerimientos.

Teniendo en cuenta lo mencionado previamente, tanto los MMV tipo 1 como el tipo 2 deben garantizar los siguientes requerimientos de operación:

- **Seguridad**, el MMV debe tener el control total de los recursos virtualizados.
- **Fidelidad**, el comportamiento de una aplicación en una máquina virtual debe ser idéntico al ejecutarlo sobre el hardware real.
- **Eficiencia**, de preferencia el código de la máquina virtual debe ejecutarse sin la intervención del MMV.

2.5.2 Implementación

Para tener control total, el MMV debe mediar cualquier intento de acceso a los recursos; esto se logra usando un esquema similar al usado por los sistemas operativos tradicionales.

La Figura 2-23 muestra el esquema de control que permite que un sistema operativo tradicional tenga control total sobre los recursos. Un computador corre en uno de los siguientes modos: modo supervisor o modo usuario. Si la máquina está en modo supervisor, es posible ejecutar cualquier instrucción existente en el conjunto de instrucciones de la arquitectura. Por el contrario, si la máquina está en modo usuario, hay un subconjunto de instrucciones llamadas privilegiadas que no pueden ser ejecutadas directamente; si un programa intenta ejecutar una instrucción privilegiada se produce una “trampa” y el control pasa al sistema operativo que se encarga de procesar la solicitud correspondiente.

Esto significa que las instrucciones que permiten el acceso a los recursos protegidos deben ser identificadas y marcadas explícitamente por los arquitectos del conjunto de instrucciones del procesador; así el sistema puede generar la trampa automáticamente.

Actividad 2-14: ¿Qué pasaría si algunas instrucciones que permiten el acceso a recursos protegidos no son identificadas y marcadas como privilegiadas?

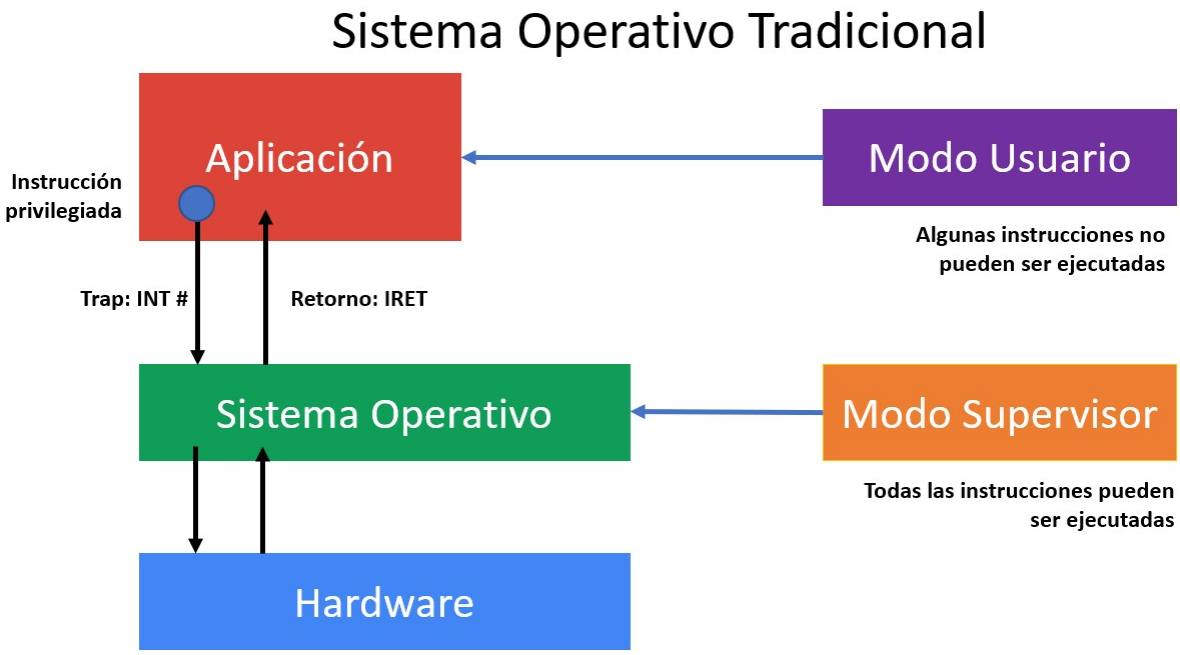

Figura 2-23. Control de ejecución de instrucciones privilegiadas.

La diferencia entre instrucciones que deberían ser marcadas como privilegiadas y las instrucciones que efectivamente son marcadas como privilegiadas da lugar a dos conceptos diferentes, instrucciones sensibles e instrucciones privilegiadas:

- Una instrucción es sensible si permite el acceso a recursos protegidos.

- Una instrucción es privilegiada si es sensible y los arquitectos del procesador la identificaron y marcaron como privilegiada para que el hardware la pueda identificar automáticamente.

Todas las instrucciones sensibles deberían ser marcadas como privilegiadas, pero en el pasado algunas arquitecturas del procesador han presentado errores. Afortunadamente, estos errores han sido corregidos en las arquitecturas actuales.

Entonces, en un ambiente con máquinas virtuales las instrucciones privilegiadas deben ser manejadas por el MMV y las trampas ya no deben pasar el control al sistema operativo, ahora deben pasar el control al MMV. Entonces, ¿qué pasa con el sistema operativo? O más bien ¿qué pasa con los sistemas operativos?, recordemos que cada máquina virtual corre un sistema operativo independiente. Cuando se produce una trampa, el control pasa al MMV y este debe decidir quién maneja el evento, él mismo o el sistema operativo que corre en la máquina virtual.

Además de identificar cuidadosamente las instrucciones privilegiadas, recientemente los fabricantes de hardware han extendido sus arquitecturas con soportes a la virtualización para mejorar los tiempos de respuesta. La virtualización indudablemente aumenta el tiempo de respuesta, pero el objetivo es minimizar este aumento. Tanto Intel-VT® como AMD-V ® adicionan hardware a los chips que contienen el procesador para mejorar el desempeño.

Los habilitadores que requieren los MMV modernos con soporte completo de virtualización asistida por hardware son:

- **Las tecnologías de aceleración de hardware** están ampliamente disponibles para las tareas de virtualización. Entre ellas se encuentran la virtualización Intel para procesadores Intel y extensiones de virtualización AMD para procesadores AMD. La aceleración de hardware mejora el rendimiento de la virtualización y la cantidad práctica de máquinas virtuales que una computadora puede alojar por encima de lo que puede hacer el hipervisor por sí solo. Los MMV de tipo 1 y el tipo 2 utilizan estas tecnologías, pero en diferente nivel.

- **Second-Level Address Translation, Rapid Virtualization Indexing, Extended Page Tables (SLAT/RVI/EPT)**, tecnología de CPU que utiliza un MMV para tener un mapa de direcciones de memoria virtual a física. Las máquinas virtuales operan en un espacio de memoria virtual que se puede dispersar por toda la memoria física, por lo que al usar un mapa adicional como SLAT/EPT reduce la latencia de la memoria acceso. Sin esta tecnología la máquina virtual tendría acceso de memoria física a las direcciones físicas de la memoria, que es inseguro y con alta latencia.

- **Intel VT o AMD-V**: si una CPU Intel tiene VT (o una CPU AMD tiene AMD-V), eso significa que admite extensiones de virtualización de hardware y virtualización completa.

- **Long mode support**, CPUs con soporte de 64 bits. Sin una arquitectura de 64 bits, la virtualización sería básicamente inútil porque solo sería posible utilizar 4 GiB de memoria para distribuir entre las máquinas virtuales.

- **Input/Output Memory Management Unit (IOMMU)**, acceso directo de las máquinas virtuales al hardware periférico (tarjetas de video, controladores de almacenamiento, dispositivos de red, etc.).

- **Single Root Input Output Virtualization (SR/IOV)**, habilita reenviar directamente un dispositivo PCI Express a múltiples máquinas virtuales. Facilita compartir un dispositivo físico con varias máquinas virtuales a través de las llamadas Funciones virtuales (VF).

- **PCI passthrough**, lo que significa que podemos tomar una tarjeta conectada PCI Express (por ejemplo, una tarjeta de video) conectada a la tarjeta madre y presentarla a una máquina virtual como si esa tarjeta estuviera conectada directamente a la máquina virtual a través de las funciones físicas (FP).

- **Trusted Platform Module (TPM)**, habilita el soporte criptográfico (es decir, para crear, guardar y asegurar el uso de llaves criptográficas).

2.5.3 Paravirtualización

Los hipervisores tipo 1 y tipo 2 corren sistemas operativos guest sin modificaciones, tal como uno los descarga e instala en su propia máquina. Aunque correr el sistema operativo sin modificaciones suena atractivo, tiene impacto en el desempeño; el tiempo de respuesta se ve afectado porque cada instrucción sensible debe ser atrapada por el hardware, manejada por el MMV y en algunos casos delegada al sistema operativo guest.

La paravirtualización resuelve el problema proponiendo que los sistemas operativos guest sean modificados reemplazando el intento de ejecución de instrucciones sensibles por llamadas a un API del hipervisor que se encarga de la ejecución. La Figura 2-24 compara el esquema de virtualización tradicional, también conocido como virtualización completa (*full virtualization*) con un sistema que usa paravirtualización.

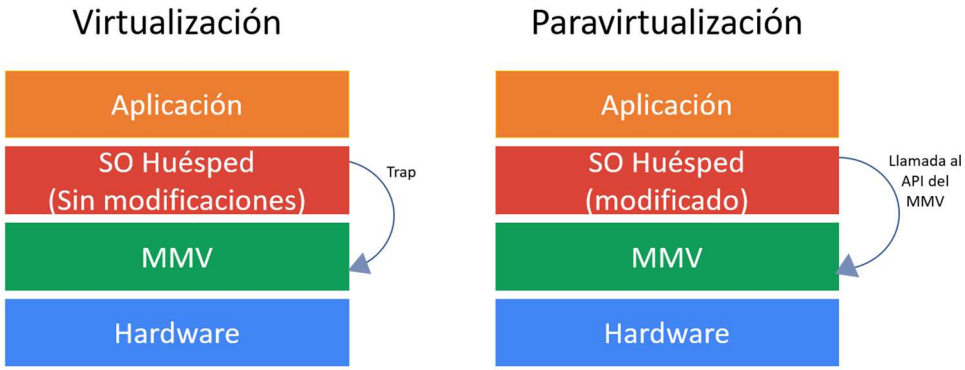

Figura 2-24. Manejo de instrucciones privilegiadas en Virtualización total vs. Paravirtualización

La virtualización ciertamente ofrece ventajas. Como ya mencionamos, reduce los costos de máquinas físicas, ofrece ambientes más flexibles y personalizables, mejora el soporte para la administración de la infraestructura de una organización y ofrece ambientes más escalables, por mencionar algunas. Pero, por otro lado, la virtualización también genera retos que los administradores deben tener en cuenta, y el principal es la administración de las máquinas virtuales. Las organizaciones son responsables de definir políticas de administración que atiendan la reglamentación vigente y las necesidades de sus empleados y usuarios.

2.6 Contenedores

Además de las ventajas mencionadas en la sección anterior, las máquinas virtuales ofrecen una forma de "empaquetar" una plataforma o aplicación con todos las dependencias y componentes necesarios para facilitar su despliegue. Sin embargo, este esquema puede ser más pesado de lo necesario; una aplicación puede ser liviana y tener pocas dependencias y aun así requerirá una máquina virtual completa generando el desperdicio de recursos.

Un contenedor responde a la misma necesidad de empaquetar una aplicación, pero crea una estructura contenedora más liviana. Esta estructura solo contiene la aplicación, sus dependencias y posiblemente algunos servicios del sistema operativo, en vez de un sistema operativo completo. Se puede decir que los contenedores crean un nivel de virtualización adicional (comparados con las máquinas virtuales) y virtualizan el sistema operativo (en lugar de la máquina completa).

Es posible ejecutar un contenedor en cualquier ambiente que ejecute el mismo kernel, por lo tanto, es posible mover un contenedor entre máquinas con el mismo kernel sin cambiar o reiniciar nada.

Los contenedores comparten el mismo kernel del sistema operativo porque son solo procesos aislados. Simplemente agregaremos un sistema de archivos con plantilla y recursos (CPU, memoria, de disco, red, etc.) a un proceso.

Los contenedores son procesos que se ejecutan en un espacio aislado en el interior y solo utilizarán su entorno definido. Como resultado, los contenedores son livianos y se inician y se detienen tan rápido como sus procesos principales. Los contenedores son tan livianos como los procesos que ejecutan, ya que no tenemos nada más ejecutándose dentro de un contenedor. Todos los recursos que consume un contenedor están relacionados con el proceso.

Actividad 2-15: Estudie el cuadro comparativo publicado por Microsoft sobre máquinas virtuales y contenedores en el enlace: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm. Piense una situación en la que es mejor usar máquinas virtuales y una en la que es mejor usar contenedores.

2.6.1 Contenedores: Docker

La diferencia entre un contenedor y máquina virtual es la ubicación de la capa de virtualización y la forma en que se utilizan los recursos del sistema operativo. Un contenedor y una máquina virtual son diferentes tecnologías para aprovisionar recursos de cómputo (CPU, memoria, etc.) que ya están presentes en una computadora física. Aunque el objetivo es el mismo, el enfoque es notablemente diferente y cada enfoque ofrece características y compensaciones únicas para las cargas de trabajo.

Figura 2-25 Virtualización vs Contenedores

Al igual que encontramos diferentes fabricantes y licencias cuando hablamos de MMV de tipo 1 y de tipo 2, en la oferta de contenedores encontramos un mercado similar con algunas diferencias técnicas que vale la pena comprender:

- **Docker**: Docker es la plataforma de contenedores más conocida y utilizada del mercado. Docker es una tecnología que le permite agregar y almacenar una aplicación y sus dependencias en un paquete llamado imagen. Esta imagen se puede usar para generar una instancia de su aplicación, el contenedor.

- **rkt**: distribuido por CoreOS, como una alternativa a Docker centrada en la seguridad.

- **LXC**: distribuido por Canonical Ltd., la empresa detrás de Ubuntu, con el objetivo de ofrecer contenedores de sistema completo. LXD es un MMV de contenedores más centrado en el sistema operativo que en la aplicación.

- **VServidor Linux**: Proporciona capacidades de contenerización a nivel de sistema operativo a través de un kernel de Linux modificado.

- **Contenedores de Windows**: Microsoft también ha introducido la función de contenedores de Windows desde Windows Server 2016. Actualmente, existen dos tipos de contenedores de Windows:

    - **Contenedores de Windows**: similares a Docker, los contenedores de Windows usan espacios de nombres y límites de recursos para aislar los procesos entre sí. Estos contenedores comparten un kernel común, a diferencia de una máquina virtual que tiene su propio kernel.

    - **Contenedores de Hyper-V**: los contenedores de Hyper-V son máquinas virtuales totalmente aisladas, altamente optimizadas que contienen una copia del kernel de Windows. A diferencia de los contenedores Docker que aíslan procesos y comparten el mismo kernel, los contenedores Hyper-V tienen cada uno su propio kernel.

En esta sección nos vamos a concentrar en Docker y su arquitectura de referencia para entender los contenedores como un habilitador diferente a la virtualización tradicional. Docker es una plataforma que le permite construir, enviar/mover y ejecutar cualquier aplicación, en cualquier lugar. Los contenedores han recorrido un largo camino en un tiempo increíblemente corto y ahora se considera una forma estándar de resolver uno de los aspectos más costosos del software: el **despliegue**.

Docker y en general las tecnologías de contenedores mueven las aplicaciones entre los entornos de desarrollo, prueba y producción con agilidad, pero también permiten adoptar la nube minimizando el esfuerzo de adopción.

Para entender Docker, realicemos un símil con un operario que trasladaba productos comerciales dentro y fuera de los barcos de carga cuando atracaban en los puertos. En el barco de carga hay cajas y artículos de diferentes tamaños y formas; los operarios experimentados fueron apreciados por su capacidad de acomodar bienes en los barcos a mano de manera rentable. Contratar personas para mover los productos no era barato.

Los sistemas modernos de carga transportan los productos comerciales dentro del barco en contenedores y fuera de los barcos no es importante el producto sino poder mover el contenedor. Para esto se requiere un solo operario de grúa que mueve los contenedores del barco al puerto, sin importarle su contenido y la estandarización facilita optimizar los tiempos de traslado. Los contenedores permiten empaquetar todo tipo de contenido en algo estándar que cualquier barco/puerto puede manipular.

Cuando hablamos de contenedores, debemos comprender los componentes que habilitan esta tecnología:

- **Runtime**: el runtime es la aplicación y las características del sistema operativo que hacen posible la ejecución y el aislamiento del proceso. En el caso de Docker, este proporciona un runtime de contenedores llamado Docker.

- **Imágenes**: las imágenes son plantillas para crear contenedores. Las imágenes contienen todo lo requerido por nuestro proceso para operar correctamente (binarios, bibliotecas, archivos de configuración, etc.).

Una característica importante de las imágenes es que son inmutables, es decir, no cambian entre ejecuciones. Con cada imagen se obtienen los mismos resultados. Las imágenes de Docker se crean a partir de una serie de capas, y todas estas capas empaquetadas contienen todo lo necesario para ejecutar un proceso. Todas las capas son de solo lectura y los cambios se almacenan en la siguiente capa superior durante la creación de la imagen y cada capa solo tiene un conjunto de diferencias con respecto a la capa anterior.

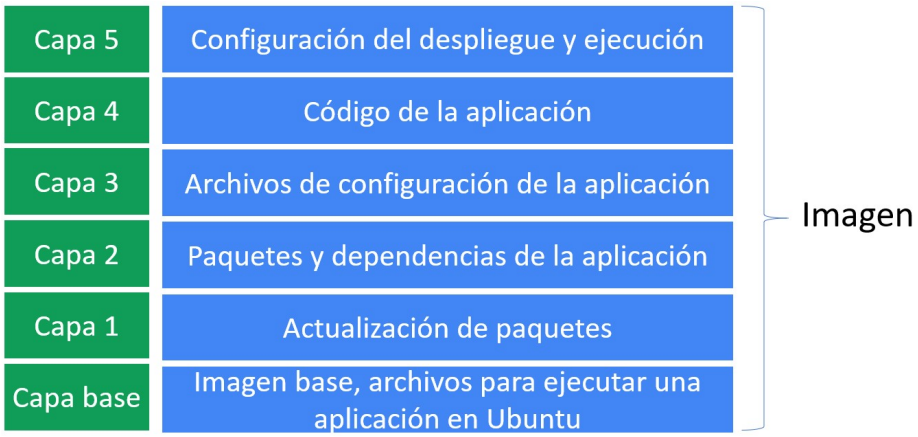

Figura 2-26. Capas que conforman una imagen

Las capas están empaquetadas para facilitar el transporte entre diferentes entornos e incluyen información sobre la arquitectura requerida para ejecutarse. Las imágenes incluyen información sobre cómo se debe ejecutar el proceso, dónde persiste la información, qué puertos expondrá el proceso para comunicarse, etc.

En el caso de Docker, las imágenes se pueden construir con métodos reproducibles usando Dockerfiles o almacenar cambios realizados en contenedores en ejecución para obtener una nueva imagen.

- **El contenedor**: un contenedor es un proceso como lo mencionamos previamente. Los contenedores se crean utilizando imágenes como plantillas. De hecho, un contenedor agrega una nueva capa de lectura y escritura en la parte superior de las capas de imagen para almacenar las diferencias del sistema de archivos de estas capas.

  Todas las capas de imágenes son capas de solo lectura, los cambios se almacenan en la capa de lectura y escritura del contenedor. Estos cambios se perderán cuando se elimina un contenedor, pero la imagen permanecerá inmutable hasta que sea borrada.

Figura 2-27. Capas que conforman una imagen

Esta característica permite ejecutar muchos contenedores usando la misma imagen subyacente, y cada uno almacenará los cambios en su propia capa de lectura y escritura. Es importante resaltar que los contenedores no son efímeros para un host; cuando se ejecuta un contenedor en un host, permanecerá allí hasta que alguien lo elimine. Se puede iniciar un contenedor detenido en el mismo host si aún no se ha eliminado conservando lo que está dentro de este, pero no es un buen lugar para almacenar el estado del proceso porque solo es local para ese host.

- **Aislamiento de procesos**: Un kernel proporciona espacios de nombres para el aislamiento de procesos. Cada contenedor se ejecuta con sus propios espacios de nombres de kernel para procesos, red, IPC, puntos de montaje, etc.
- **Registro**: El único requisito para aprovisionar un contenedor en un nuevo nodo es la imagen para crear ese contenedor y recursos de cómputo. La imagen se puede compartir entre nodos, para garantizar la distribución de imágenes se utilizan registros de imágenes, que son puntos de almacenamiento para este tipo de objetos, similares a un repositorio, similar a los repositorios de código como Github ofreciendo control de versiones de la imagen.

Cuando hablamos de una tecnología específica, como lo es Docker, siempre nos preguntamos: ¿Cuándo utilizar Docker? Surgen algunas preguntas prácticas cruciales: ¿por qué usaría Docker y para qué? La respuesta simple al "por qué" es por un mínimo esfuerzo. Analicemos el cuándo y el para qué.

Docker u otro de sus equivalentes se puede usar para reemplazar máquinas virtuales en muchas situaciones. Si solo importa la aplicación, no el sistema operativo, el contenedor puede reemplazar la VM y dejar de preocuparse por el sistema operativo.

Antes de los contenedores Docker, el despliegue de aplicación en diferentes entornos requería un esfuerzo considerable. Incluso si no estuviera ejecutando scripts para aprovisionar software en diferentes máquinas (y mucha gente todavía hace exactamente eso), se debe asumir el costo de gestionar las herramientas de administración de configuración.

Figura 1. Antes de los Contenedores Docker - Triple esfuerzo

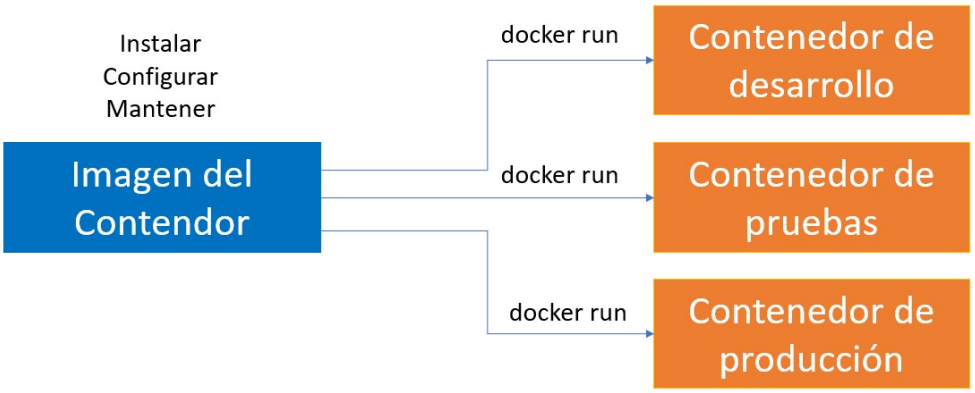

Figura 2-28. Esfuerzo con Contenedores Docker

Un contenedor Docker es un paquete de software que comprende todo lo necesario para ejecutar una aplicación de forma independiente. Puede haber múltiples contenedores Docker en una sola máquina y los contenedores son procesos independientes en el sistema operativo del host. En otras palabras, un contenedor Docker incluye un componente de software junto con todas sus dependencias (binarios, bibliotecas, archivos de configuración, scripts, etc.). El contenedor Docker tiene su propio espacio de proceso e interfaz de red.

Anteriormente ilustramos el concepto de Imagen y su mecanismo de capas. En Docker, las capas de una imagen son esencialmente archivos generados al ejecutar algún comando (instalación, configuración, copiado o borrado, etc.). Las capas están ordenadas porque pueden ser reutilizadas por varias imágenes, lo que ahorra espacio en el disco y reduce el tiempo de creación de imágenes manteniendo su integridad. En el ejemplo que presentamos a continuación podemos ver 3 imágenes que parten de la misma base para desplegar diferentes aplicaciones.

Figura 2-29. Imagen de un servidor web Linux – Apache – PHP

Figura 2-30. Imagen de un servidor de base de datos MySQL

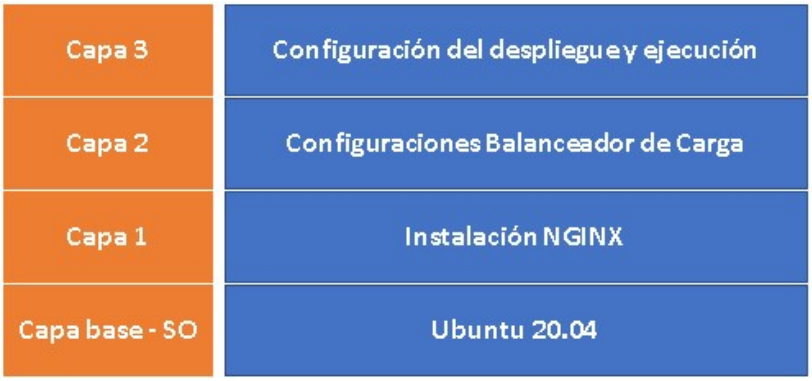

Figura 2-31. Imagen de un balanceador de carga con NGINX

Los contenedores Docker son bloques de construcción para aplicaciones. Cada contenedor es una imagen con una capa que se puede leer/escribir encima de un montón de capas de solo lectura. Estas capas (llamadas

imágenes intermedias) se generan a partir de un archivo de configuración que se utiliza para compilación de la imagen.

Al instanciar estas imágenes en contenedores Docker se puede desplegar una aplicación completa, por ejemplo:

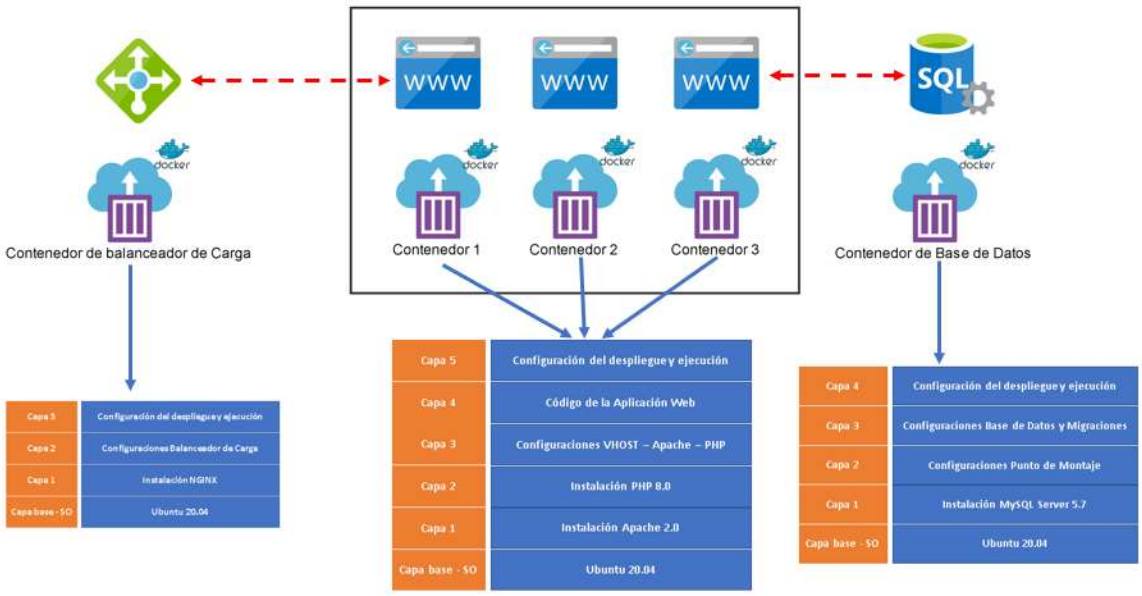

Figura 2-32. Arquitectura de ejemplo: Balanceador de carga + grupo de contenedores capa web + base de datos

Recordemos: Un proceso es un mecanismo de interacción con un sistema operativo. Un programa es un conjunto de instrucciones para ejecutar en el sistema; un proceso será ese código en ejecución. Un proceso utiliza recursos de máquina (CPU, memoria, etc.) y aunque se ejecutará en su propio entorno, puede compartir información con otro proceso que se ejecute en la misma máquina. Los contenedores, como se mencionó previamente son procesos aislados en el sistema, pero se pueden comunicar por red.

Cada vez que Docker crea y ejecuta un contenedor desde una imagen, agrega una capa de escritura, conocida como capa contenedora, que almacena todos los cambios en el contenedor durante su tiempo de ejecución.

Como esta capa es la única diferencia entre un contenedor en ejecución y la imagen, cualquier número de contenedores similares puede potencialmente compartir el acceso a la misma imagen subyacente mientras mantiene su propio estado individual.

Nota: La primera capa de una imagen de Docker se conoce como imagen principal. Es la base sobre la que se construyen todas las demás capas y proporciona los bloques de construcción básicos para sus entornos de contenedores. Una imagen principal típica puede ser una distribución de Linux simplificada.

Estas características hacen de Docker una herramienta ideal para implementar microservicios (un habilitador para Cloud Native), ya que facilita la descomposición de un sistema complejo en una serie de piezas. Esto puede permitirle reestructurar su software para que sus partes sean más manejables e integrables sin afectar el conjunto.

Docker como un habilitador para continuous delivery. La entrega continua (CD) es un paradigma para la entrega de software basado en un pipeline que reconstruye el sistema en cada cambio y luego entrega a producción a través de un proceso automatizado (o parcialmente automatizado).

En pocas palabras, Docker nos permite ensamblar rápidamente aplicaciones empresariales y críticas para el negocio. Para hacer esto, podemos usar componentes de software diferentes y distribuidos; Docker también nos permite probar el código y luego implementarlo en producción lo más rápido posible.

2.7 Conclusión: Máquinas virtuales vs Contenedores

La selección de una tecnología como las máquinas virtuales o los contenedores se base en objetivos de negocio y la arquitectura de su aplicación. Por ejemplo, las aplicaciones monolíticas son comunes en ambientes empresariales, generalmente funcionan bien con máquinas virtuales y los mecanismos tradicionales para garantizar disponibilidad.

Los contenedores son ideales en entornos donde se requieran características como despliegue rápido, alta escalabilidad y tolerancia a fallos. Los contenedores son un habilitador para el desarrollo de aplicaciones con arquitecturas de nueva generación, como las arquitecturas de microservicios y las arquitecturas Cloud Native.

En última instancia, la elección de contenedores frente a máquinas virtuales no es mutuamente excluyente. Los contenedores y las máquinas virtuales pueden coexistir fácilmente en el mismo ambiente, incluso en el mismo servidor, por lo que las dos tecnologías se consideran complementarias.

Actividad 2-16: Ingrese a la URL https://labs.play-with-docker.com/ y cree una cuenta de usuario.
Play with Docker es un servicio en la nube que permite a los usuarios ejecutar comandos de Docker y desplegar contenedores en cuestión de segundos sin instalar ni aprovisionar ningún recurso local.