Virtualización eficiente de la memoria

Sistemas de Virtualización y Seguridad

Rodrigo Peña

Index

Introducción

La virtualización es una herramienta que nos da beneficios como la gestión de recursos, seguridad y tolerancia a fallos. Además, también nos proporciona acceso al hardware a pedido en los entornos clouds.

Motivación

Los benificios de la virtualización vienen con overheads en el procesamiento, la memoria y la entrada/salida y los avances hardware han reducido sus overhead. Sin embargo los overheads debidos a la virtualización de memoria no siempre son bajos.

Motivación

"We will show that the increase in translation lookaside buffer (TLB) miss handling costs due to the hardware-assisted memory management unit (MMU) is the largest contributor to the performance gap between native and virtual servers."
–Buell et al. VMware Technical Journal, Summer 2013

Motivación

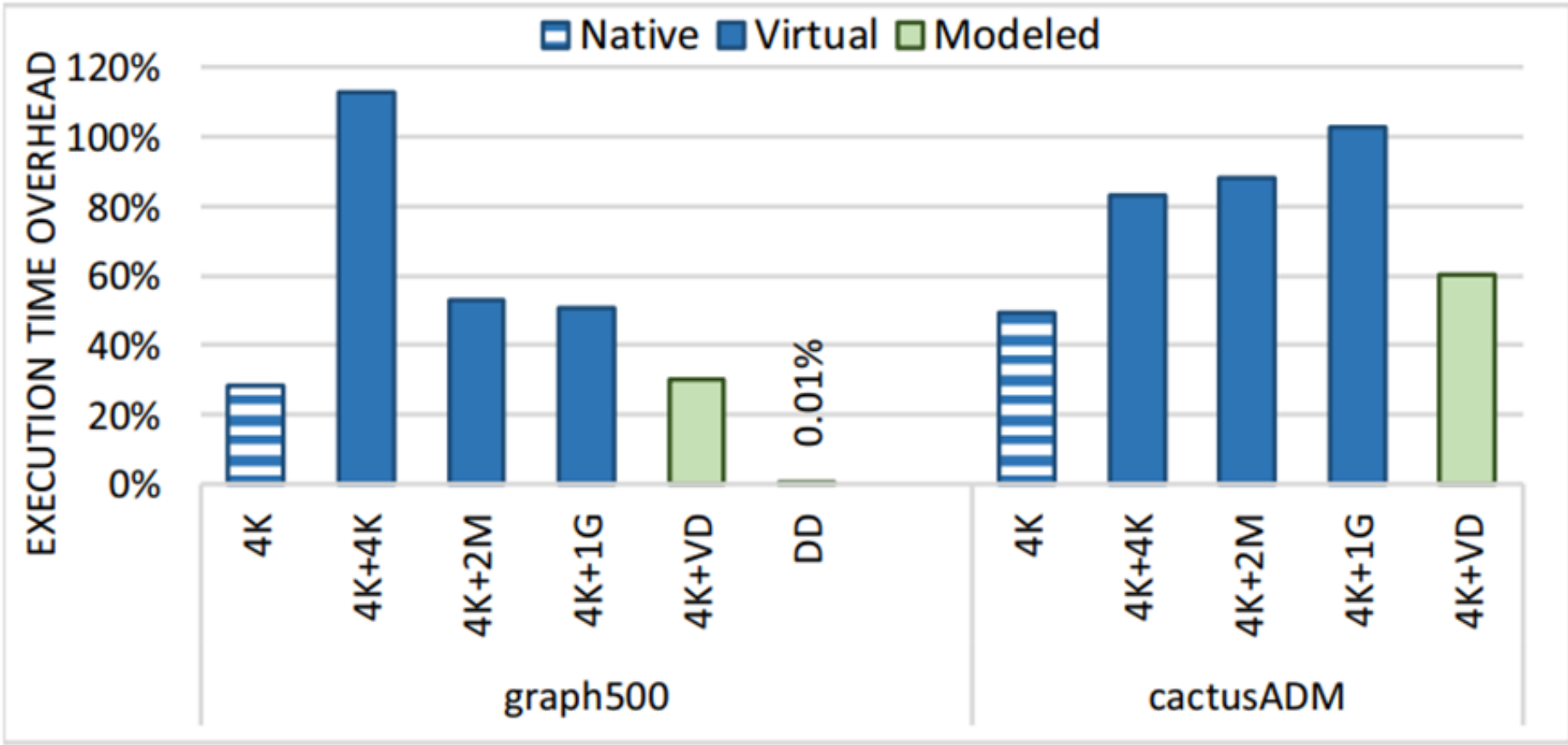


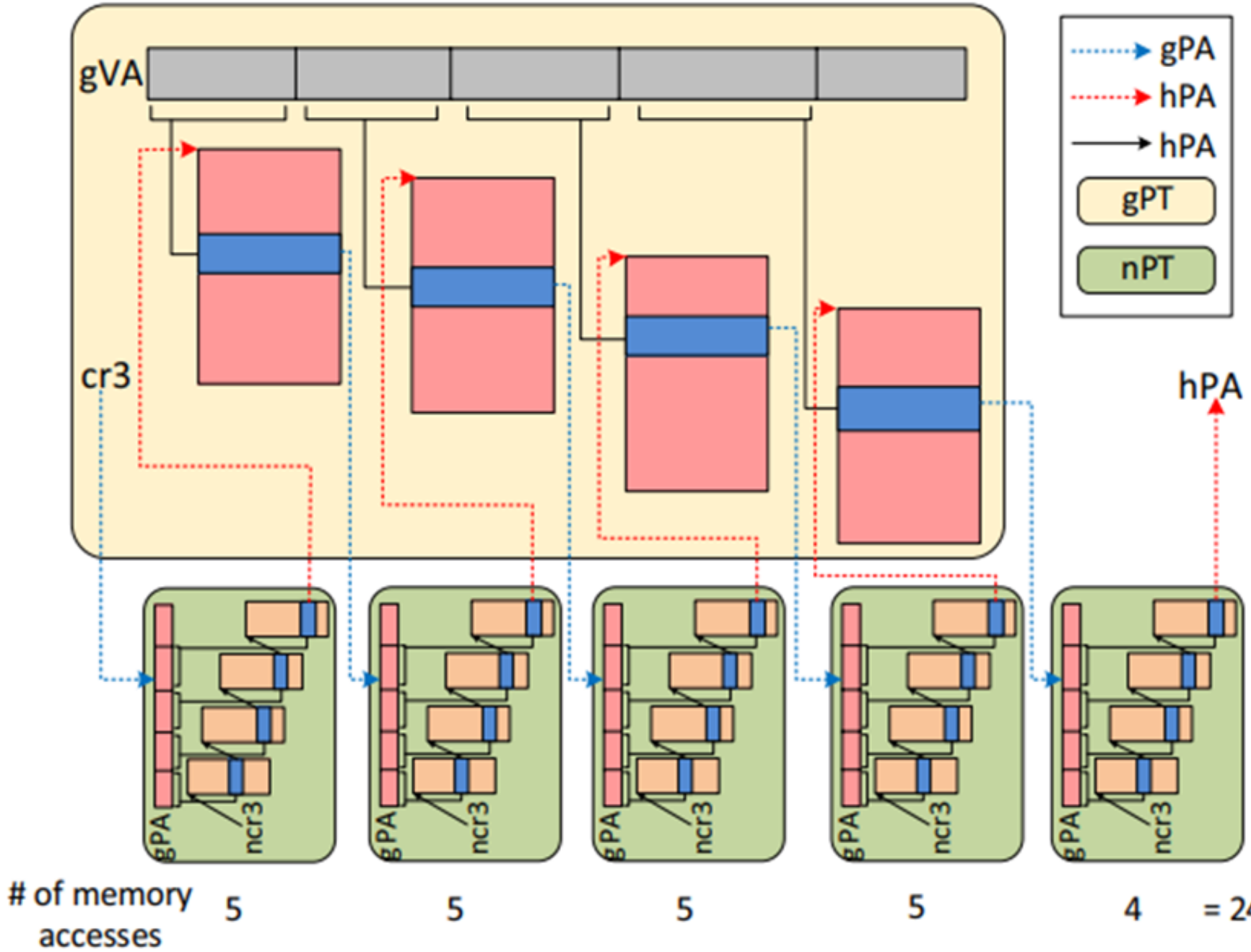
Figure 1 Overheads associated with virtual memory for selected workloads and configurations

Objetivo

Conseguir una virtualización eficiente para cualquier carga de trabajo.

Equipo de trabajo

- Se utilizará el hardware para soporte de virtualización x86-64.
- En el mejor caso el TLB traduce la dirección virtual del guest a la dirección física del host sin overhead, y en el peor caso esto produce un TLB miss con un 2D page walk.


gVA
cr3
gPA
hPA
hPA
gPT
nPT
gPA
ncr3
# of memory accesses
5
5
5
5
4
= 24

Propuesta

- Nuevo hardware que soporta 3 nuevos modos de virtualización y extiende los segmentos directos.
- self-ballooning
- escape filter

Propuesta

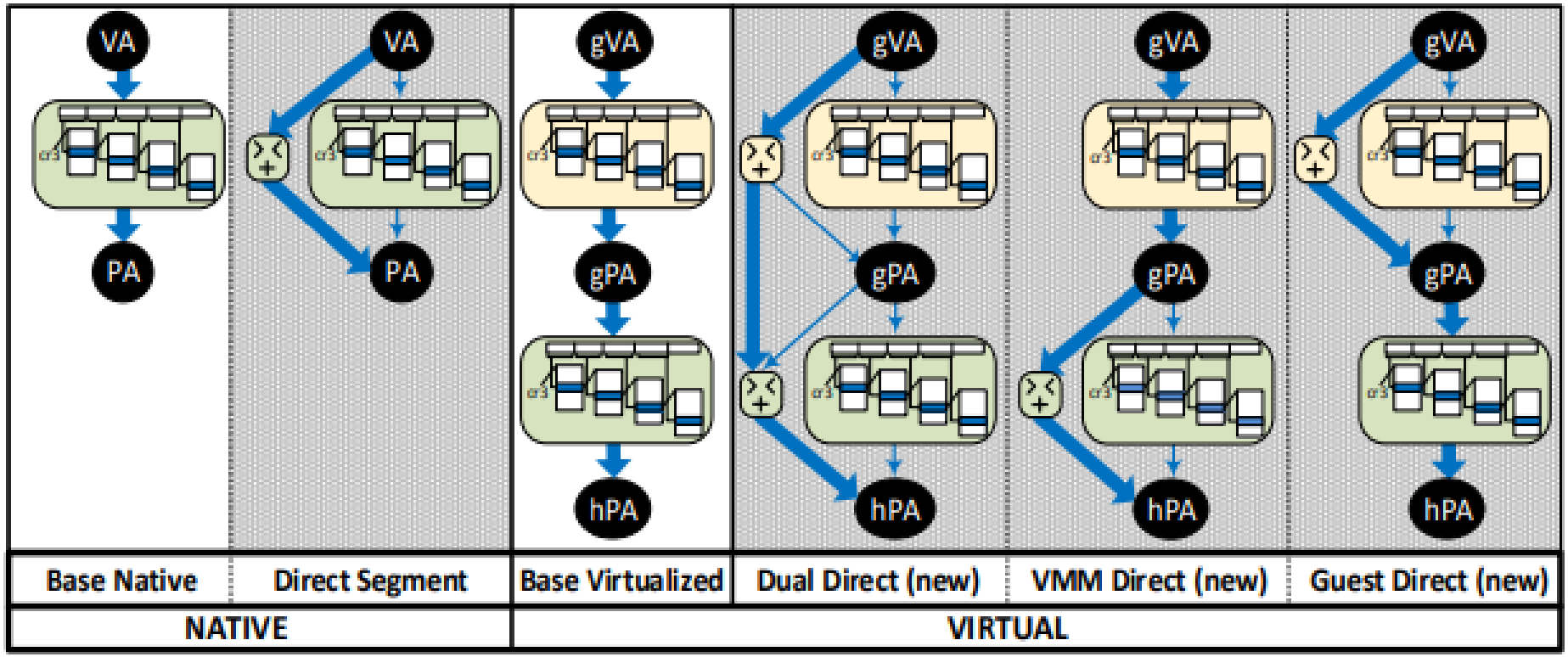


Figure 3 Native and virtualized address translation modes supported by proposed hardware

Direct Segment

- 3 registros: BASE, LIMIT y OFFSET
- Usan Primary Region
- Convive con la paginación

Direct Segment

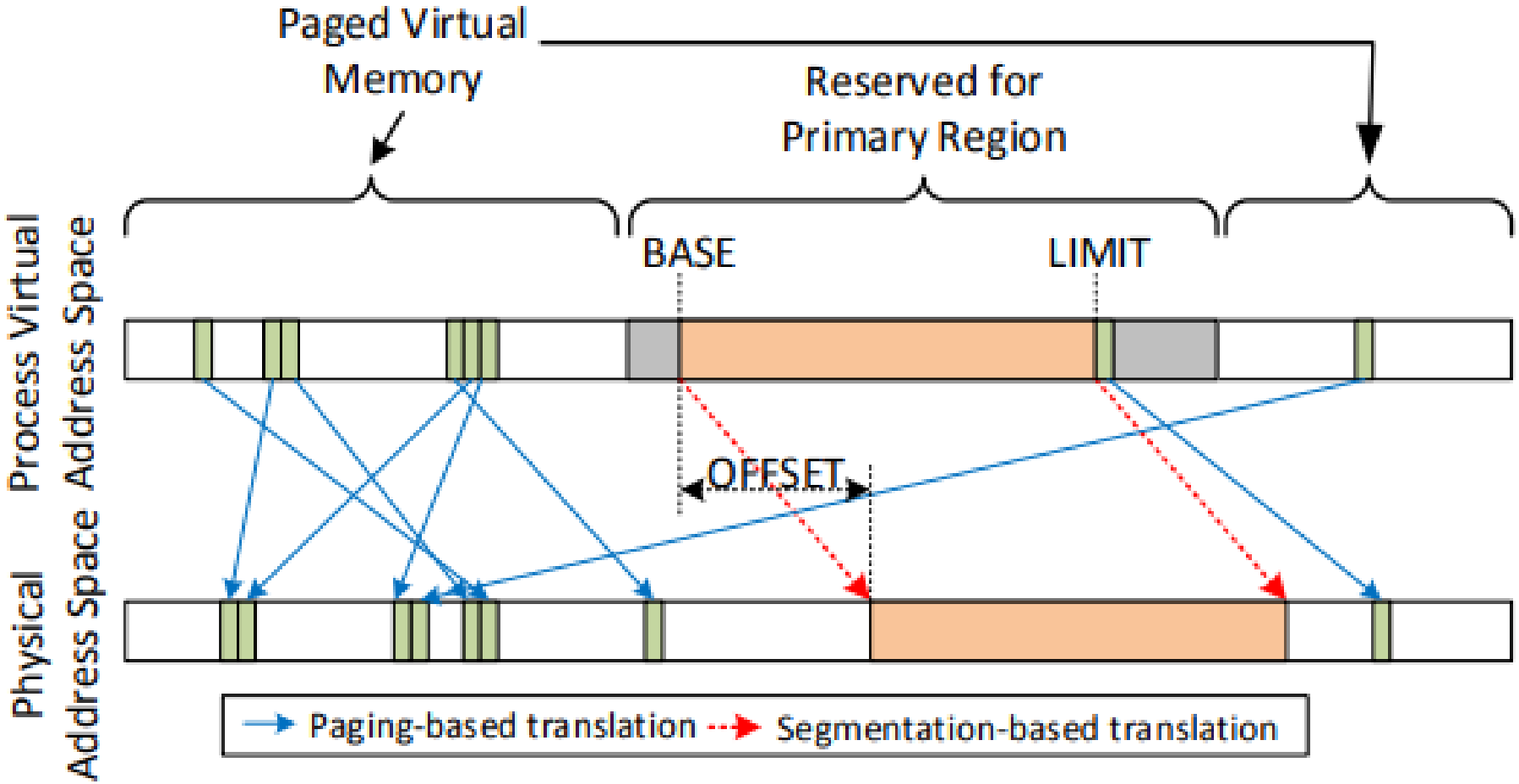


Figure 4 Address space layout using direct segment

Index

Hardware

- Soporta dos niveles de registros:
  $BASE_g, LIMIT_g, OFFSET_g$
  $BASE_v, LIMIT_v, OFFSET_v$
- Puede utilizar independientemente uno, dos o ningún nivel de registros.

Hardware

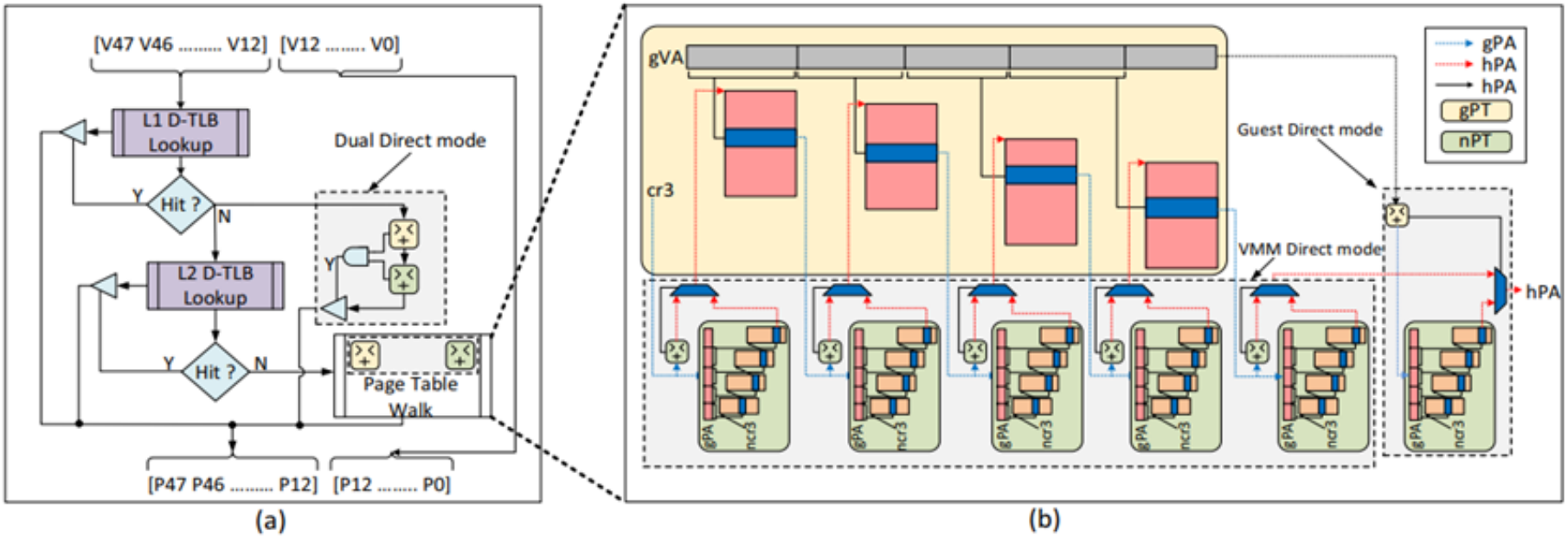


Figure 5 (a) Address translation flow chart (b) Steps of page walk state machine for various supported modes

Dual Direct Mode

- Busca utilizar un 0D walk page
- Usa 2 capas de segmento directo, una en el guest y la otra en el host.
- Primer nivel traduce gVA $\longrightarrow$ gFA
- Segundo nivel hVA $\longrightarrow$ hFA
- Aplicaciones de gran memoria
- Requiere de cambios en el guest OS y el VMM
- 0 accesos a memoria en la mayoría de casos

Dual Direct Mode

Cambios en el guest OS:

Los valores de los registros de segmento se asignan a cada proceso y deben ser actualizados en los cambios de contexto.

Poco útiles para cargas de trabajo de mucho cómputo.

Cambios en el VMM

Necesita pedir un bloque de memoria contiguo.

Necesita guardar y escribir los registros para cada máquina virtual.

Dual Direct Mode

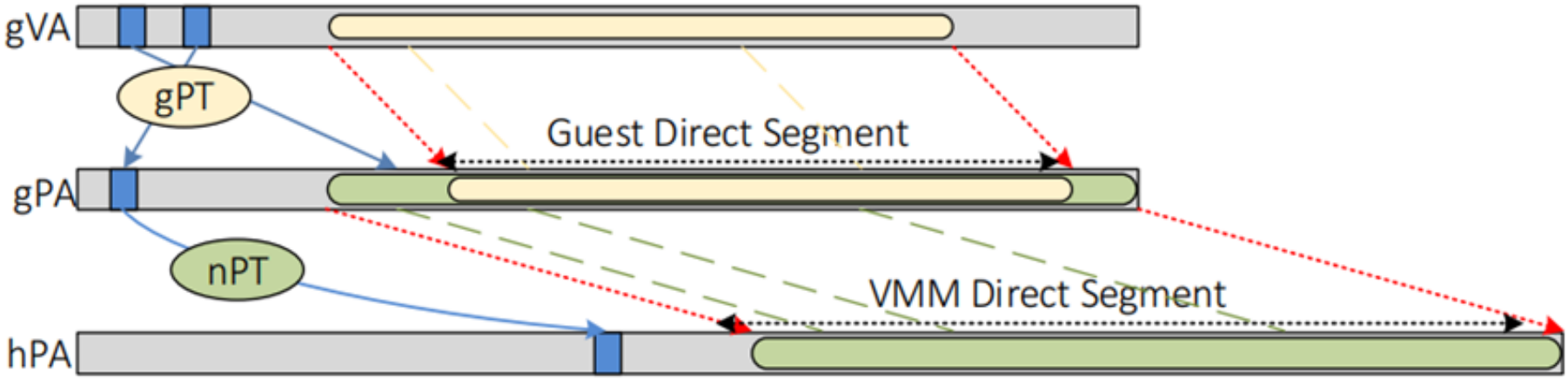


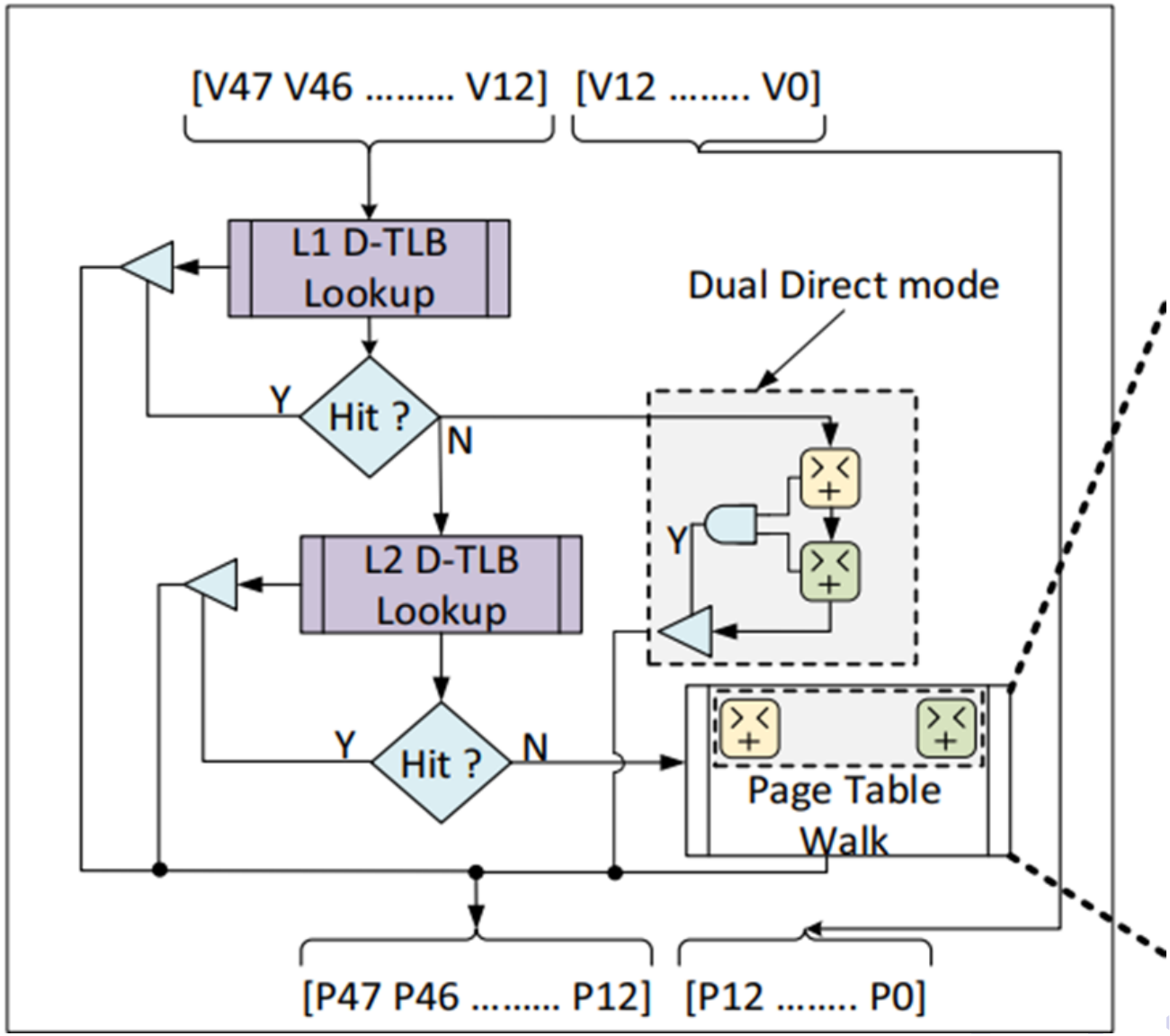
Figure 6 Memory layout for Dual Direct mode.

Funcionamiento Dual Direct

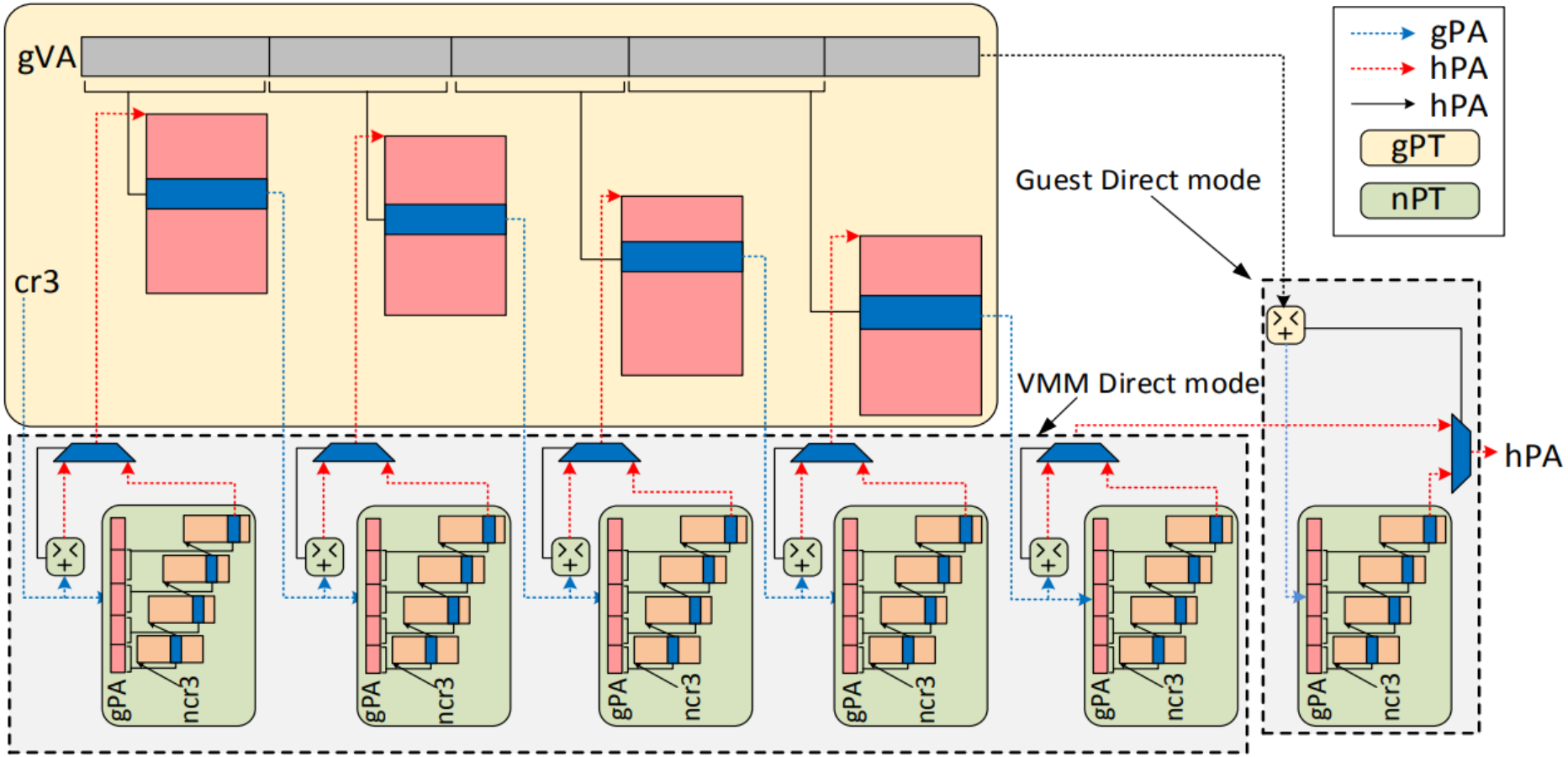
TABLE I STEPS IN ADDRESS TRANSLATION OF A GUEST VIRTUAL ADDRESS IN DUAL DIRECT MODE.

Steps of Translation	Guest Virtual Address in Guest Segment or VMM Segment?			
	Both	**VMM segment only**	**Guest segment only**	**Neither**
L1 TLB Hit	Translation complete	Translation complete	Translation complete	Translation complete
L1 TLB Miss	$hPA=gVA+OFFSET_G+OFFSET_V$ Insert L1 TLB entry	L2 TLB lookup	L2 TLB lookup	L2 TLB lookup
L2 TLB Hit	—	Insert L1 TLB entry	Insert L1 TLB entry	Insert L1 TLB entry
L2 TLB Miss	—	Invoke PTW	Invoke PTW	Invoke PTW
PTW: gVA→gPA	—	Walk guest OS page table	$gPA=gVA+OFFSET_V$	Walk guest page table
PTW: gPA→hPA	—	For each gPA, $hPA=gPA+OFFSET_V$ or walk nested page table	Walk nested page table	For each gPA, walk nested page table
PTW: End	—	Insert L1 TLB entry	Insert L1 TLB entry	Insert L1 TLB entry

Funcionamiento Dual Direct



Funcionamiento Dual Direct



VMM direct

- Busca utilizar un 1D walk page
- Usa paginación en el guest y la segmentación en el host
- Se ponen los registros $BASE_g$ y $LIMIT_g$ a un mismo valor anulando el Dual Direct Mode
- Aplicaciones de cualquier tamaño
- Requiere de cambios en el VMM
- Con cambios en el guest OS aumenta más el rendimiento
- 4 accesos a memoria y 5 sumas en la mayoría de los casos

Funcionamiento VMM direct

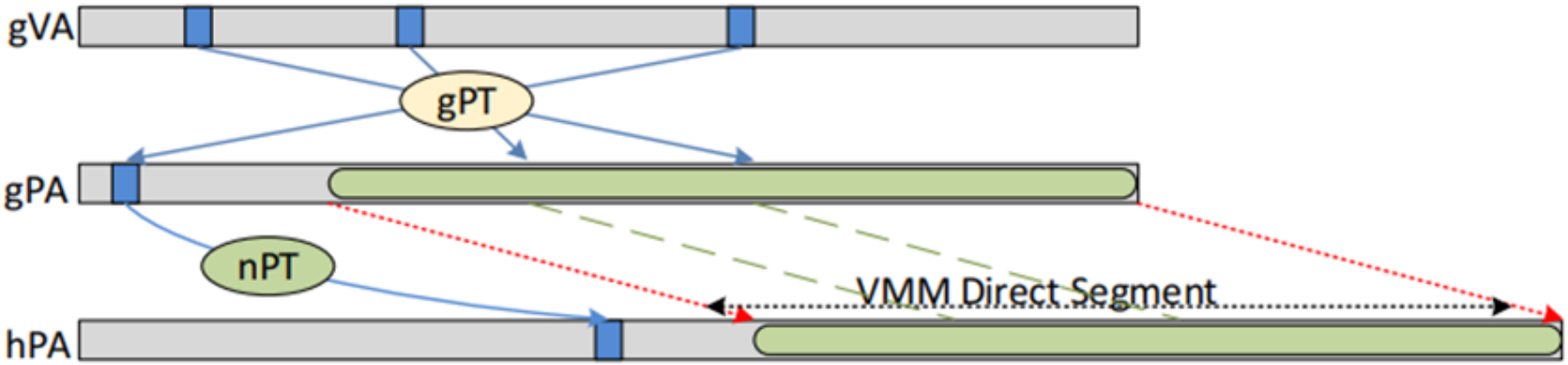


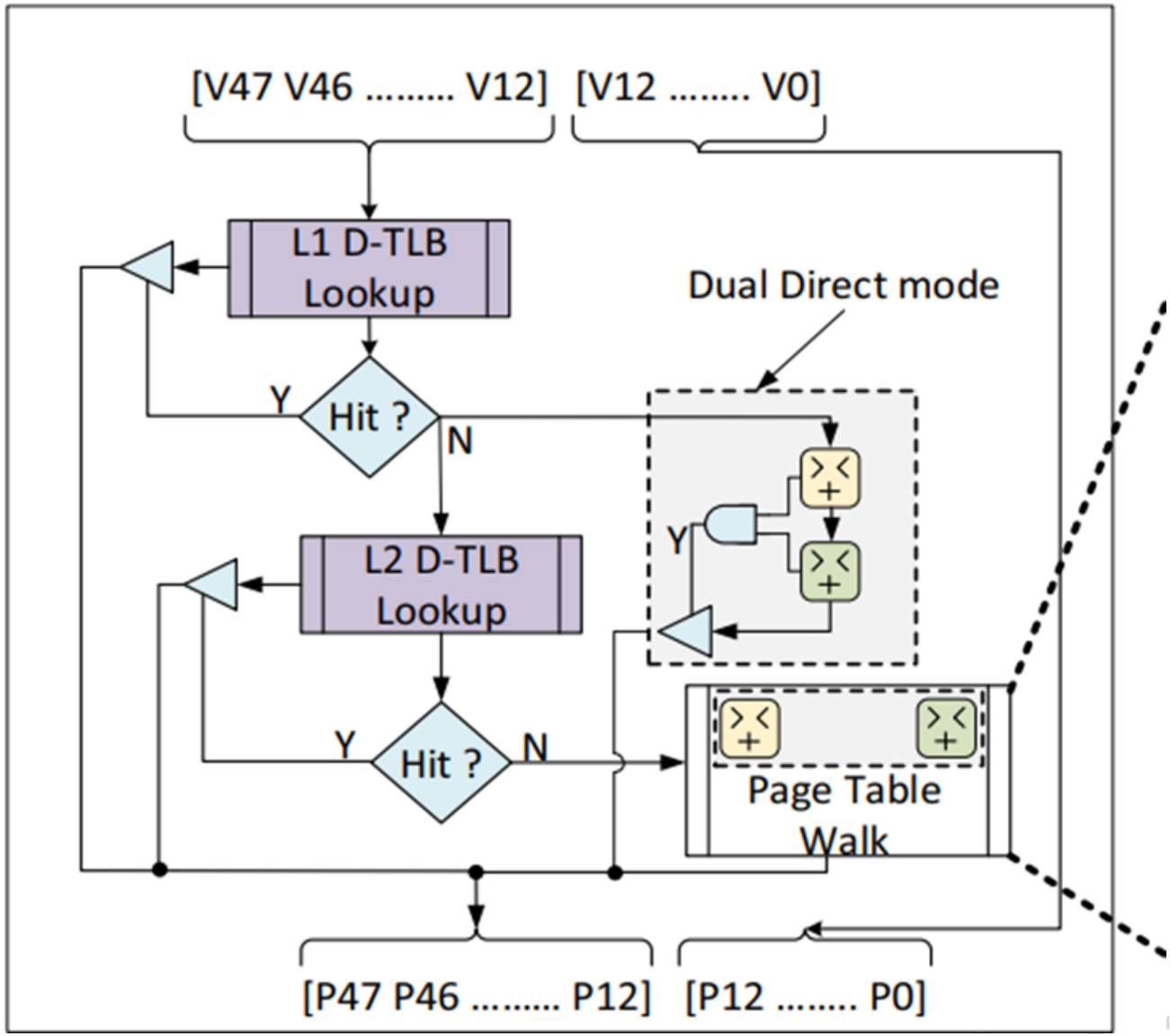
Figure 7 Memory layout for VMM Direct mode

Funcionamiento VMM direct

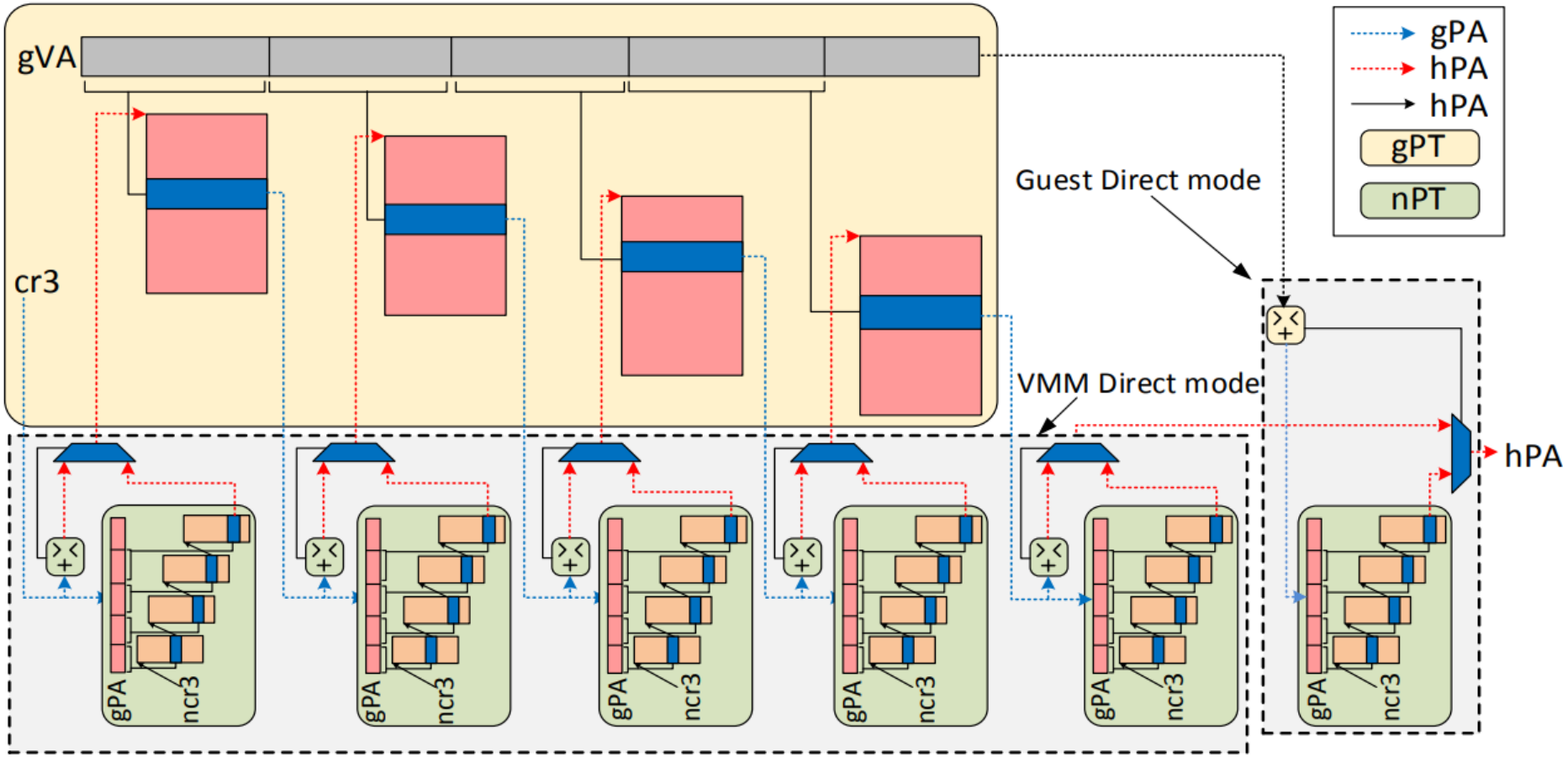
TABLE I STEPS IN ADDRESS TRANSLATION OF A GUEST VIRTUAL ADDRESS IN DUAL DIRECT MODE.

Steps of Translation	Guest Virtual Address in Guest Segment or VMM Segment?			
	Both	**VMM segment only**	**Guest segment only**	**Neither**
L1 TLB Hit	Translation complete	Translation complete	Translation complete	Translation complete
L1 TLB Miss	$hPA=gVA+OFFSET_G+OFFSET_V$ Insert L1 TLB entry	L2 TLB lookup	L2 TLB lookup	L2 TLB lookup
L2 TLB Hit	—	Insert L1 TLB entry	Insert L1 TLB entry	Insert L1 TLB entry
L2 TLB Miss	—	Invoke PTW	Invoke PTW	Invoke PTW
PTW: gVA→gPA	—	Walk guest OS page table	$gPA=gVA+OFFSET_V$	Walk guest page table
PTW: gPA→hPA	—	For each gPA, $hPA=gPA+OFFSET_V$ or walk nested page table	Walk nested page table	For each gPA, walk nested page table
PTW: End	—	Insert L1 TLB entry	Insert L1 TLB entry	Insert L1 TLB entry

Funcionamiento VMM direct



Funcionamiento VMM direct



Guest direct

- Busca realizar 1D page walk.
- Usa segmentación en el guest y paginación en el VMM
- Soporta page sharing y live migration.
- Aplicaciones de gran memoria.
- Se ponen los registros $BASE_v$ y $LIMIT_v$ a un mismo valor anulando el Dual Direct Mode
- 4 accesos de memoria y 1 suma en la mayoría de los casos

Funcionamiento guest direct

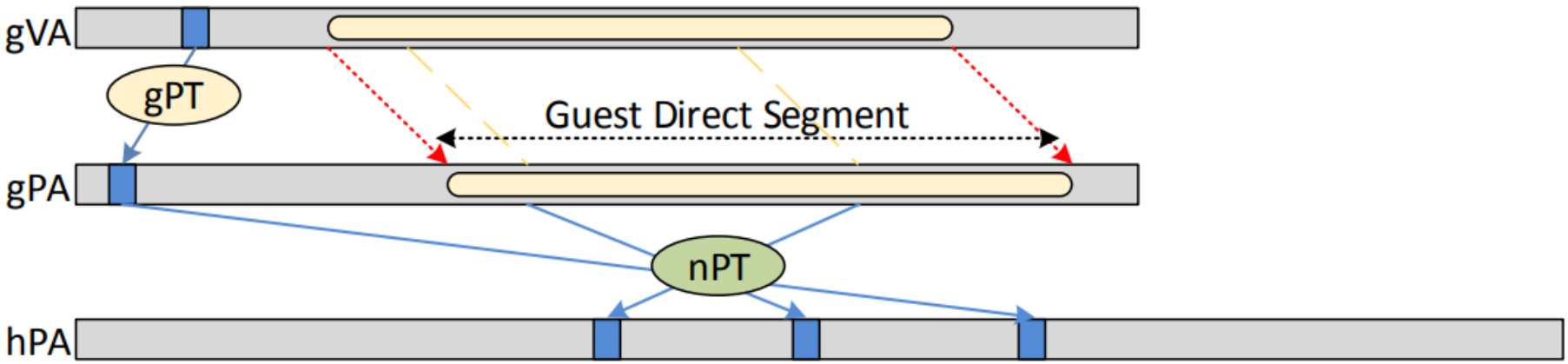


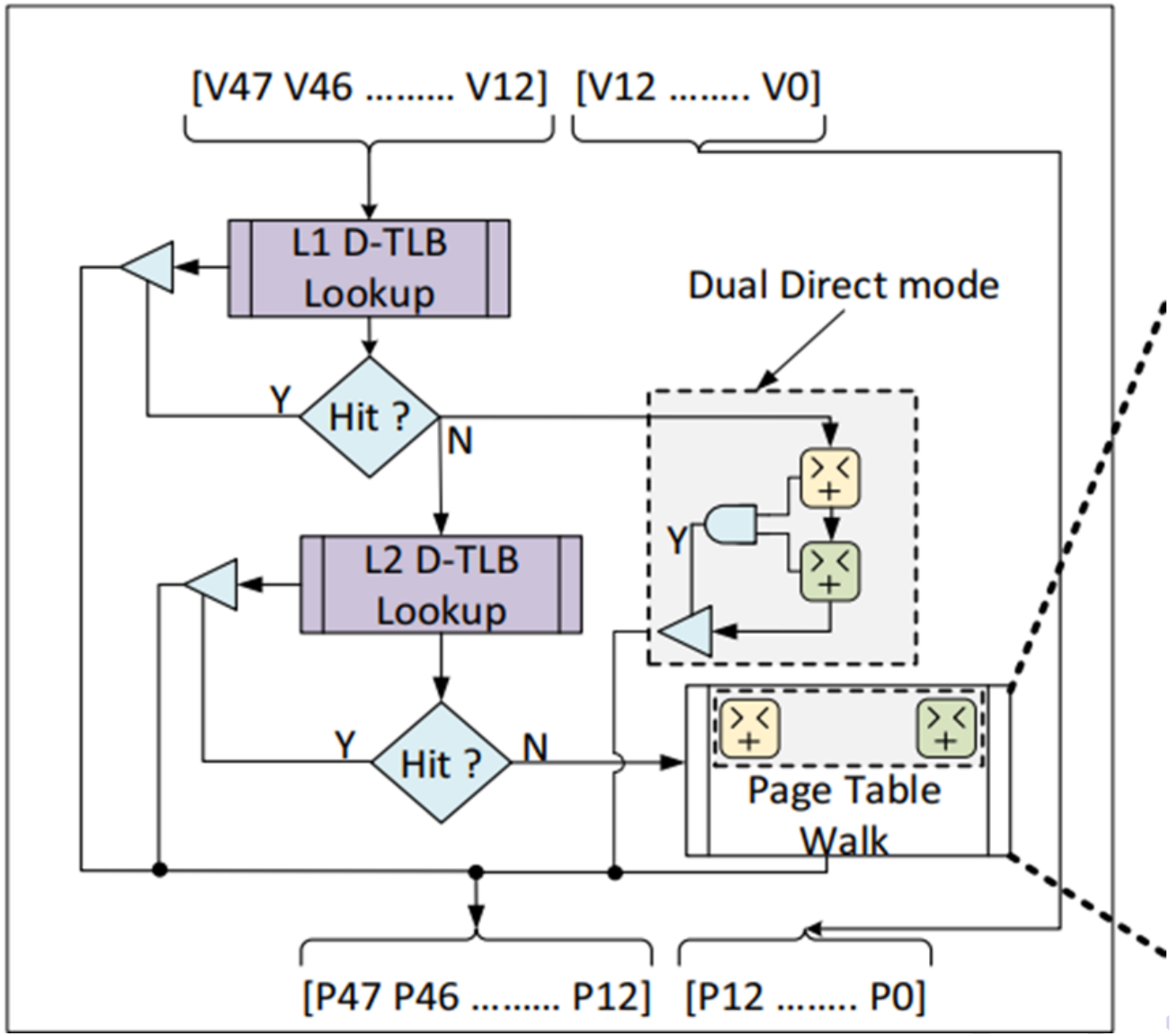
Figure 8 Memory layout for Guest Direct mode

Funcionamiento guest direct

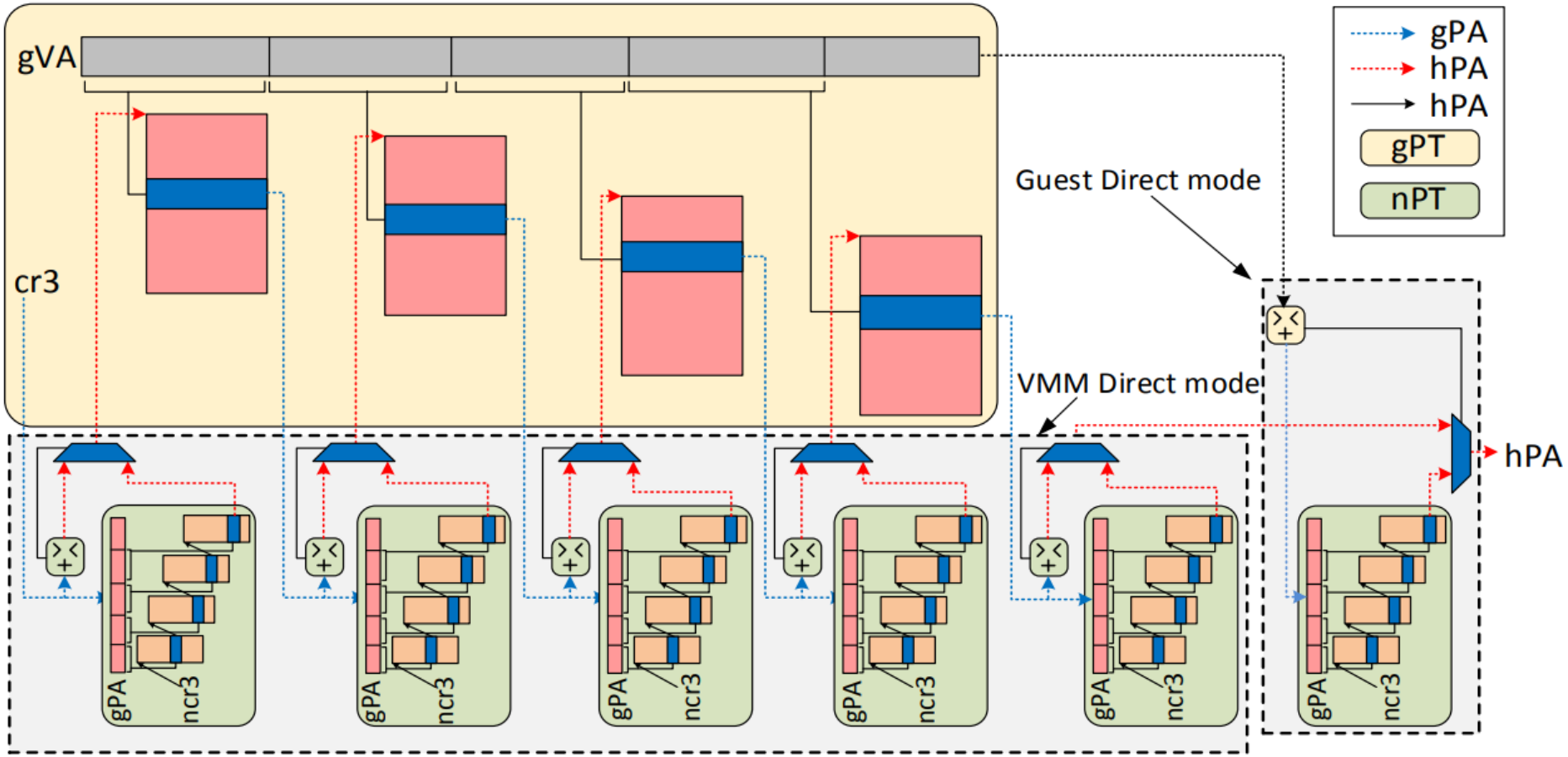
TABLE I STEPS IN ADDRESS TRANSLATION OF A GUEST VIRTUAL ADDRESS IN DUAL DIRECT MODE.

Steps of Translation	Guest Virtual Address in Guest Segment or VMM Segment?			
	Both	**VMM segment only**	**Guest segment only**	**Neither**
L1 TLB Hit	Translation complete	Translation complete	Translation complete	Translation complete
L1 TLB Miss	$hPA=gVA+OFFSET_G+OFFSET_V$ Insert L1 TLB entry	L2 TLB lookup	L2 TLB lookup	L2 TLB lookup
L2 TLB Hit	—	Insert L1 TLB entry	Insert L1 TLB entry	Insert L1 TLB entry
L2 TLB Miss	—	Invoke PTW	Invoke PTW	Invoke PTW
PTW: gVA→gPA	—	Walk guest OS page table	$gPA=gVA+OFFSET_V$	Walk guest page table
PTW: gPA→hPA	—	For each gPA, $hPA=gPA+OFFSET_V$ or walk nested page table	Walk nested page table	For each gPA, walk nested page table
PTW: End	—	Insert L1 TLB entry	Insert L1 TLB entry	Insert L1 TLB entry

Funcionamiento guest direct



Funcionamiento guest direct



Index

Self-ballooning

Para evitar la fragmentación en la memoría física del guest y facilitar una región contigua de memoria se propone una nueva técnica self-ballonig que consta de 2 pasos.

- Primer paso, el balloon driver marca las páginas de memoría y las reserva para que no puedan ser utilizadas.
- Segundo paso, el driver cede esta memoria al VMM que usa hotplug memmory para añadir la misma cantidad de memoria a la máquina virtual.

Diseñado para el modo guest direct.

Self-ballooning

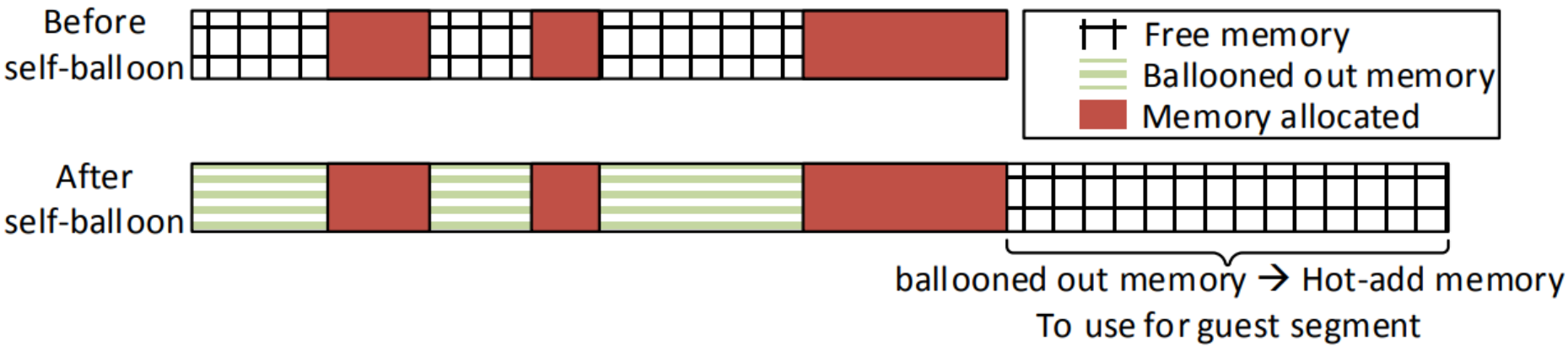


Figure 9 Illustration of guest physical memory with self-ballooning.

I/O gap

Para evitar fragmentación debido al I/O gap que hay en la arquitectura x86-64 se utiliza hot-unplug y se extiende la misma cantidad de memoria.

Compactación de memoria

Para desfragmentar la memoria física del host se utiliza compactación en el sistema operativo. Mientras se está compactando la memoría el modo Dual Direct funciona como Guest Direct y el modo VMM direct como Base Virtualized.
Cuando el sistema operativo nos provee un suficiente memoría contigua se crea un segmento en el VMM consiguiendo un mejor rendimiento en estos modos.

Index

Páginas defectuosas

Para evitar el uso de las páginas defectuosas algunos sistemas operativos utilizan una lista con páginas incorrectas para evitar su uso.

Scape filter

Con el scape filter se forman agujeros en el segmento en los que usa paginación para traducir la dirección de memoria.
Se ha implementado con un Bloom filter y se comprueba en paralelo con los registros del segmento del VMM.

Index

Implementación del prototipo

El prototipo tiene 3 piezas:

- Asignación contigua de memoria física: En las aplicaciones de gran uso de memoría se puede conocer a priori la memoría que necesitan y reservarla.
- Emular los segmentos:
  - Los segmentos se emulan mapeando páginas de 4KB.
  - Se modifica el manejador de fallos de TLB para comprobar si el fallo de página está dentro de un segmento directo. En caso de que esté añade la nueva dirección al TLB.
- Prototipo de self-balloning: Modificando QEMU-KVM y virtio balloon driver. KVM no soporta hot-adding por lo que se se amplía la segunda ranura de memoría al máximo 96GB.

Index

Evaluación de la propuesta

Para evaluar el rendimiento de la propuesta se han utilizado los siguientes modelos

TABLE IV LINEAR MODEL FOR CYCLES SPENT ON PAGE WALK.

Design	Model
Direct Segment	$C_n*(1-F_{DS}) * M_n$
Dual Direct	$[(C_n+\Delta_{VD})*F_{VD} + (C_n+\Delta_{GD})*F_{GD} + C_v*(1-F_{GD}- F_{VD}-F_{DD})]*M_n$
VMM Direct	$[(C_n+\Delta_{VD})*F_{VD} + C_v*(1-F_{VD})]*M_n$
Guest Direct	$[(C_n+\Delta_{GD})*F_{GD} + C_v*(1-F_{GD})]*M_n$

- C_n: Ciclos de un page walk nativos
- C_v: Ciclos de un page walk virtual
- F: porcentaje de TLB misses que caen en los segmentos directos
- M_n: misses nativos

Workloads

TABLE V WORKLOAD DESCRIPTION.

Workload	Description
Graph500	Generation, compression and breadth-first search (BFS) of very large graphs, as often used in social networking analytics and HPC computing.
Memcached	In-memory key-value cache widely used by large web-sites, for low-latency data retrieval.
NPB:CG	NASA's high performance parallel benchmark suite. CG workload from the suite.
GUPS	Random access benchmark defined by the High Performance Computing Challenge.
SPEC® 2006	Compute single-threaded workloads: cactusADM, GemsFDTD, mcf, omnetpp (ref inputs)
PARSEC3.0	Compute multi-threaded workloads: canneal, streamcluster (native input set)

Sistema

TABLE VI DETAILS OF THE NATIVE AND VIRTUALIZED SYSTEMS.

Native System	
Processor	Dual-socket Intel Xeon E5-2430 (SandyBridge) 6 cores/socket, 2 threads/core, 2.2 GHz
Physical Memory	96 GB DDR3 1066MHz
Operating System	Fedora-20 (Linux LTS Kernel v3.12.13)
L1 Data TLB	4KB: 64 entries 4-way associative 2MB: 32 entries 4-way associative 1GB: 4-entry fully associative
L2 TLB	4KB: 512 entries 4-way associative
Fully-Virtualized System with KVM	
VMM	QEMU (with KVM) v1.4.1, 24vCPUs
Physical Memory	85GB
Operating System	Fedora-20 (Linux LTS Kernel v3.12.13)
EPT TLB/NTLB	Shares the TLB (no separate structure)

Resultados Big-Memmory workloads

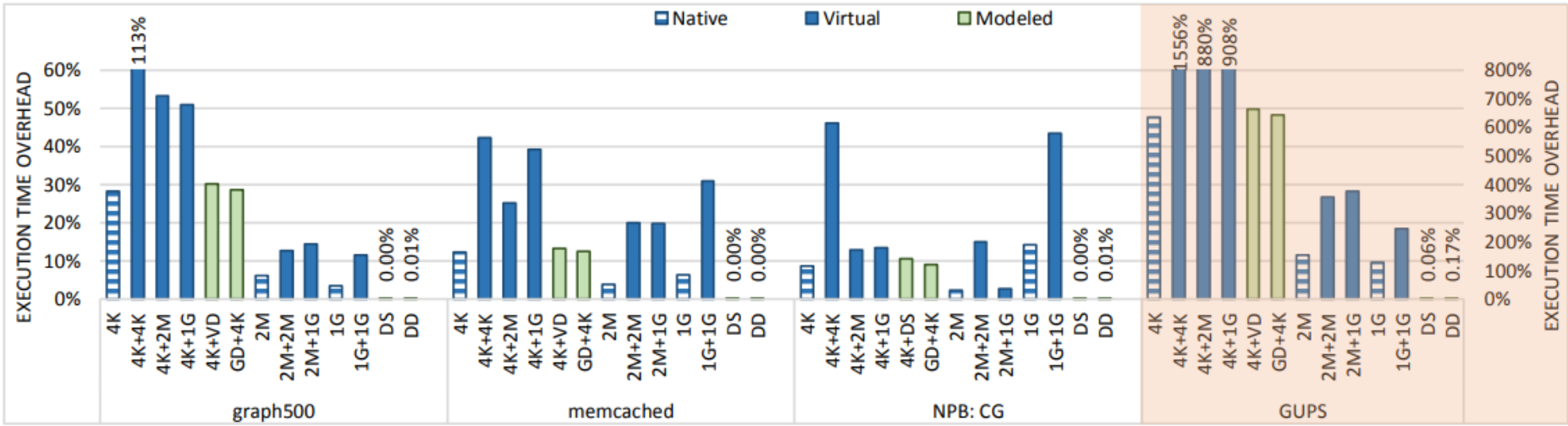


Figure 11 Virtual memory overhead for each configuration per big-memory workload.

Resultados compute workloads

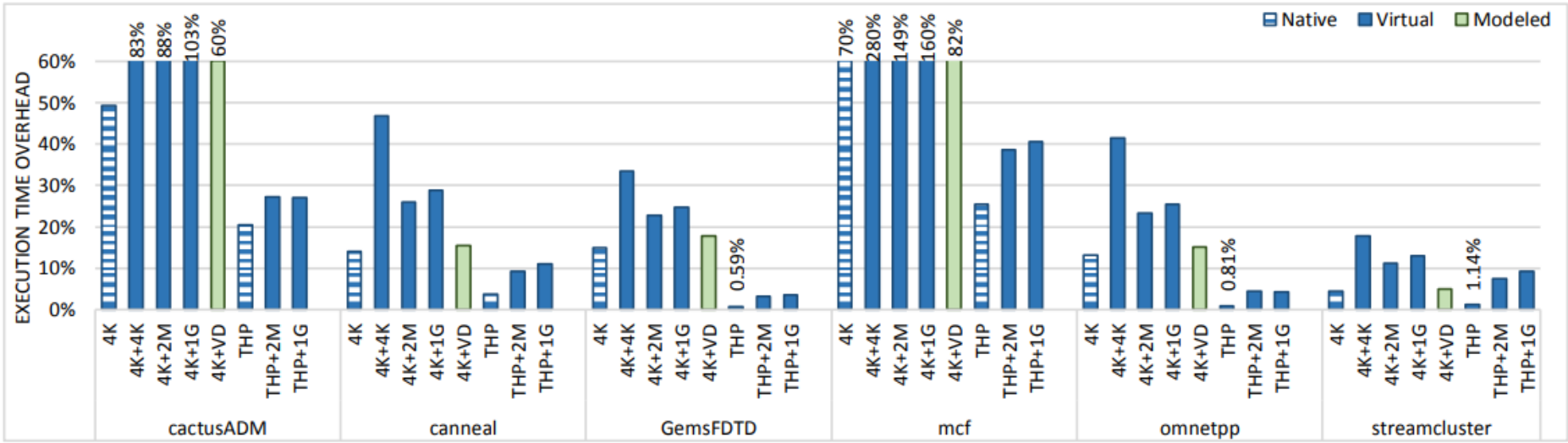


Figure 12 Virtual memory overhead for each configuration per compute workload.

Resultados y obeservaciones

- VMM direct y Guest Direct consigues unos overheads casi nativos en big-memmory workloads
- El rendimiento de VMM direct se vería mejorado con el uso de páginas de 2MB, 1 GB o THP.
- Dual Direct consigue un rendimiendo similar al Direct Segment sin virtualizar
- El numero de ciclos por TLB miss media aumenta un 13 % con VMM direct y un 3 % con guest direct, menor que entre 2.4x o 1.5x dependiendo del tamaño de páginas.
- el uso de scape filter solo supone un overhead de un 0.5 % en el peor caso para Dual Direct.

Index

Evaluación del paper

Comentarios positivos

- Explicación clara y sencilla de los 3 nuevos modos de funcionamientos y self-ballooning.
- Gran cantidad de imágenes explicativas

Comentarios negativos

- Bastante información repetida o redundande en el paper.
- No queda claro si se puede cambiar de modo con la máquina en funcionamiento.
- Falta de test de rendimiento con otras configuraciones como 2MB pages con big memmory workloads
- Faltan test para comprobar si hay una mejora significativa en los workloads de cómputo con el uso de THP.