

# LUF (Lazy Unmap Flush)

Skip TLB Shutdown on Unmapping

SK hynix

Byungchul Park

[max.byungchul.park@sk.com](mailto:max.byungchul.park@sk.com)

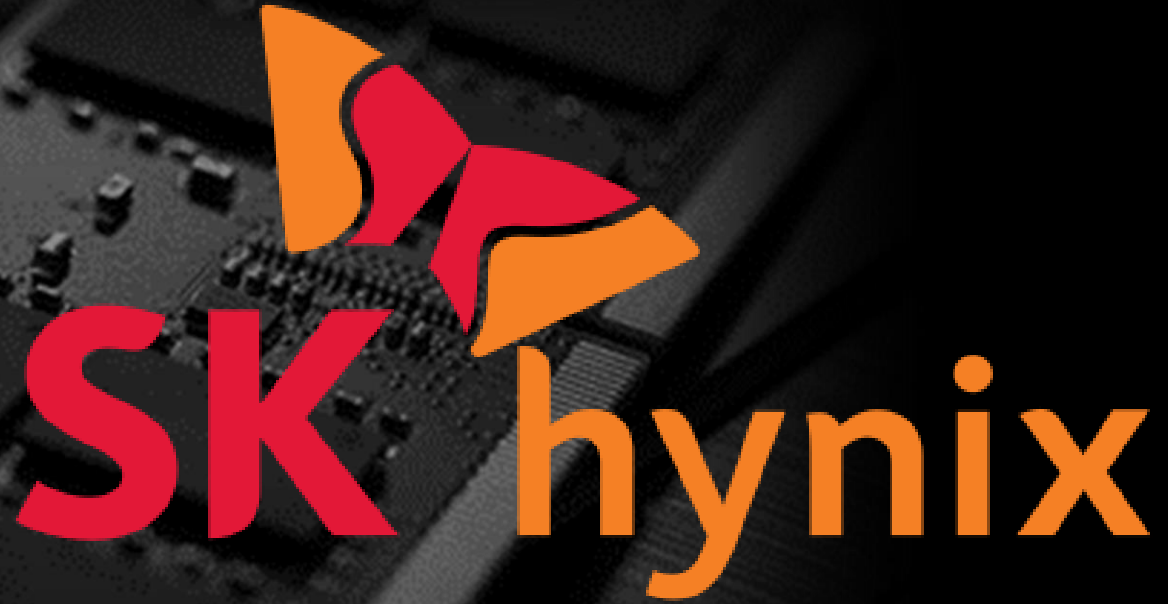
[max.byungchul.park@gmail.com](mailto:max.byungchul.park@gmail.com)

DRAM

HBM

CXL Expander

CXL Pooled Memory

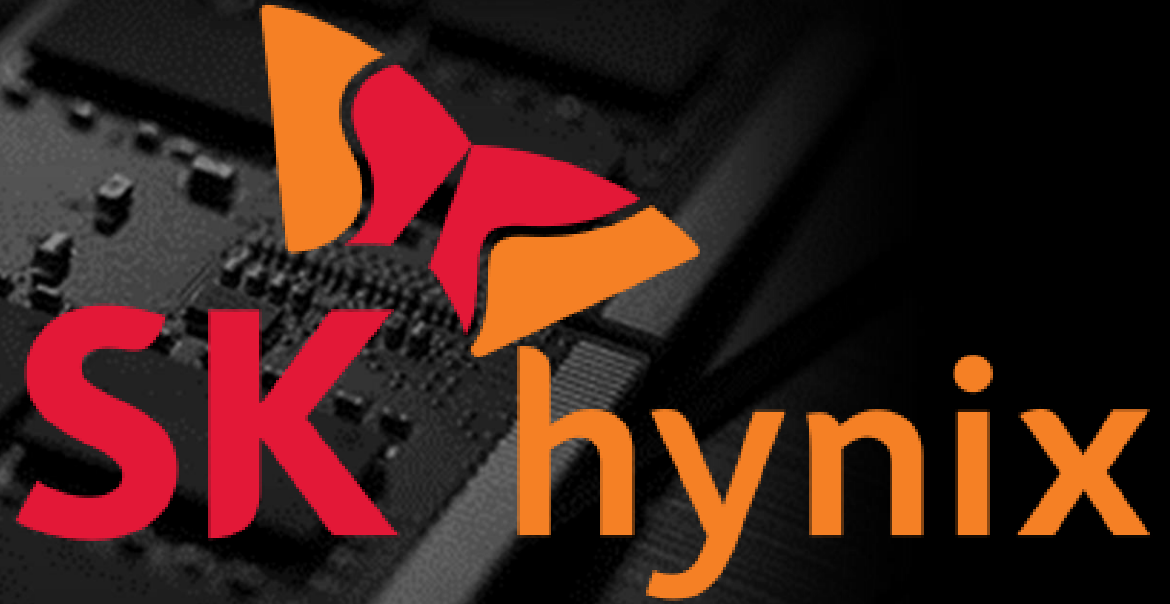


**DRAM**

**HBM**

**CXL Expander**

**CXL Pooled Memory**



## MACHINE

Intel(R) Xeon(R) Gold 6430

1 socket 64 cores hyper threaded on (x2)

DRAM 42GB + CXL Expander 96GB

swap off, numa balancing tiering on, demotion enabled

## MEASUREMENT (LLM inference)

```
llama.cpp/main -m $(70G_model1) -p "who are you?" -s 1 -t 15 -n 20 &
```

```
llama.cpp/main -m $(70G_model2) -p "who are you?" -s 1 -t 15 -n 20 &
```

```
llama.cpp/main -m $(70G_model3) -p "who are you?" -s 1 -t 15 -n 20 &
```



## MACHINE

Intel(R) Xeon(R) Gold 6430

1 socket 64 cores hyper threaded on (x2)

DRAM 42GB + CXL Expander 96GB

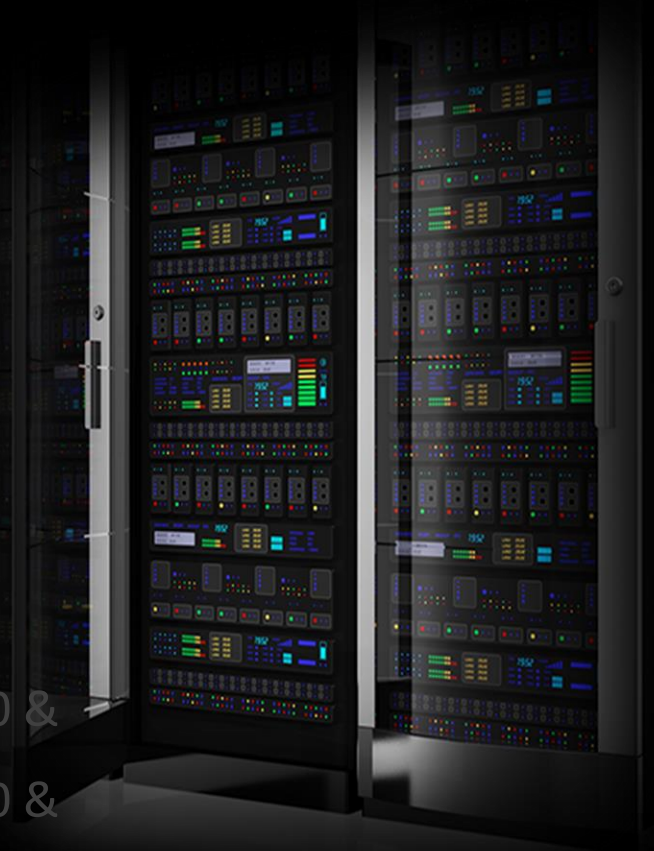
swap off, numa balancing tiering on, demotion enabled

## MEASUREMENT (LLM inference)

```
llama.cpp/main -m $(70G_model1) -p "who are you?" -s 1 -t 15 -n 20 &
```

```
llama.cpp/main -m $(70G_model2) -p "who are you?" -s 1 -t 15 -n 20 &
```

```
llama.cpp/main -m $(70G_model3) -p "who are you?" -s 1 -t 15 -n 20 &
```



## MACHINE

Intel(R) Xeon(R) Gold 6430

1 socket 64 cores hyper threaded on (x2)

DRAM 42GB + CXL Expander 96GB

swap off, running on a single node in a multi-tenant environment

# Heavy page reclaim

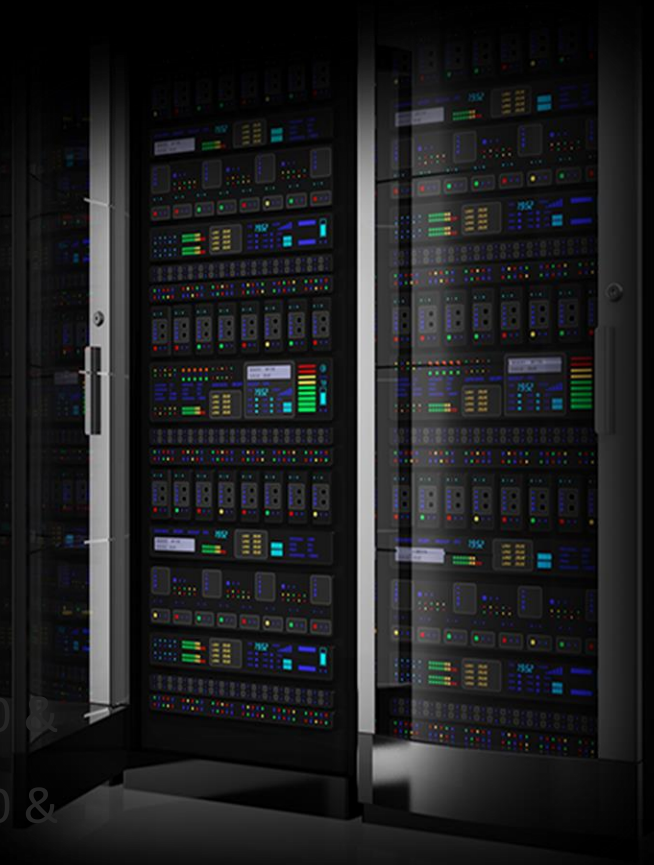
## MEASUREMENT (MLM (Free))

# Heavy page migration

```
llama.cpp/main -m $(70G_model1) -p "who are you?" -s 1 -t 15 -n 20 &
```

```
llama.cpp/main -m $(70G_model2) -p "who are you?" -s 1 -t 15 -n 20 &
```

```
llama.cpp/main -m $(70G_model3) -p "who are you?" -s 1 -t 15 -n 20 &
```



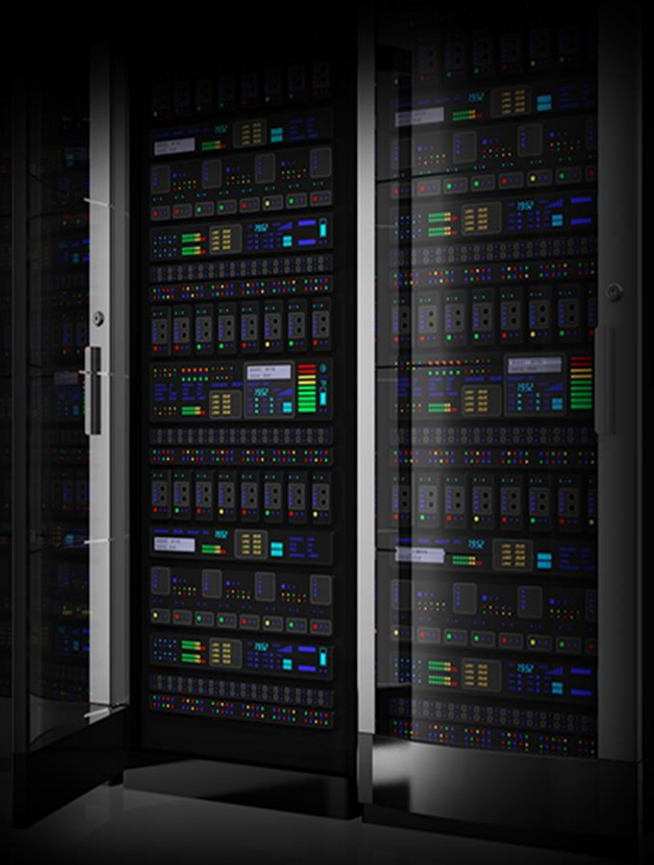


While the workload of llama.cpp runs for 4000 secs...

```
$ grep TLB /proc/interrupts
```

TLB:

```
80911532 93691786 100296251 111062810 109769109 109862429
108968588 119175230 115779676 118377498 119325266 120300143
124514185 116697222 121068466 118031913 122660681 117494403
121819907 116960596 120936335 117217061 118630217 122322724
119595577 111693298 119232201 120030377 115334687 113179982
118808254 116353592 140987367 137095516 131724276 139742240
136501150 130428761 127585535 132483981 133430250 133756207
131786710 126365824 129812539 133850040 131742690 125142213
128572830 132234350 131945922 128417707 133355434 129972846
126331823 134050849 133991626 121129038 124637283 132830916
126875507 122322440 125776487 124340278 TLB shutdowns
```



# TLB (Translation Lookaside Buffer)

TLB is a CPU cache for mapping.

What if **mappings have changed**?

CPUs with the mappings should **invalidate their TLBs**,

So that the CPUs can run with the updated ones.



# TLB Shutdown

A method to invalidate TLBs involved.

Request TLB flush to CPUs.

Pay the cost of IPI(Inter Processor **Interrupt**).

Pay the cost of **TLB miss**.

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

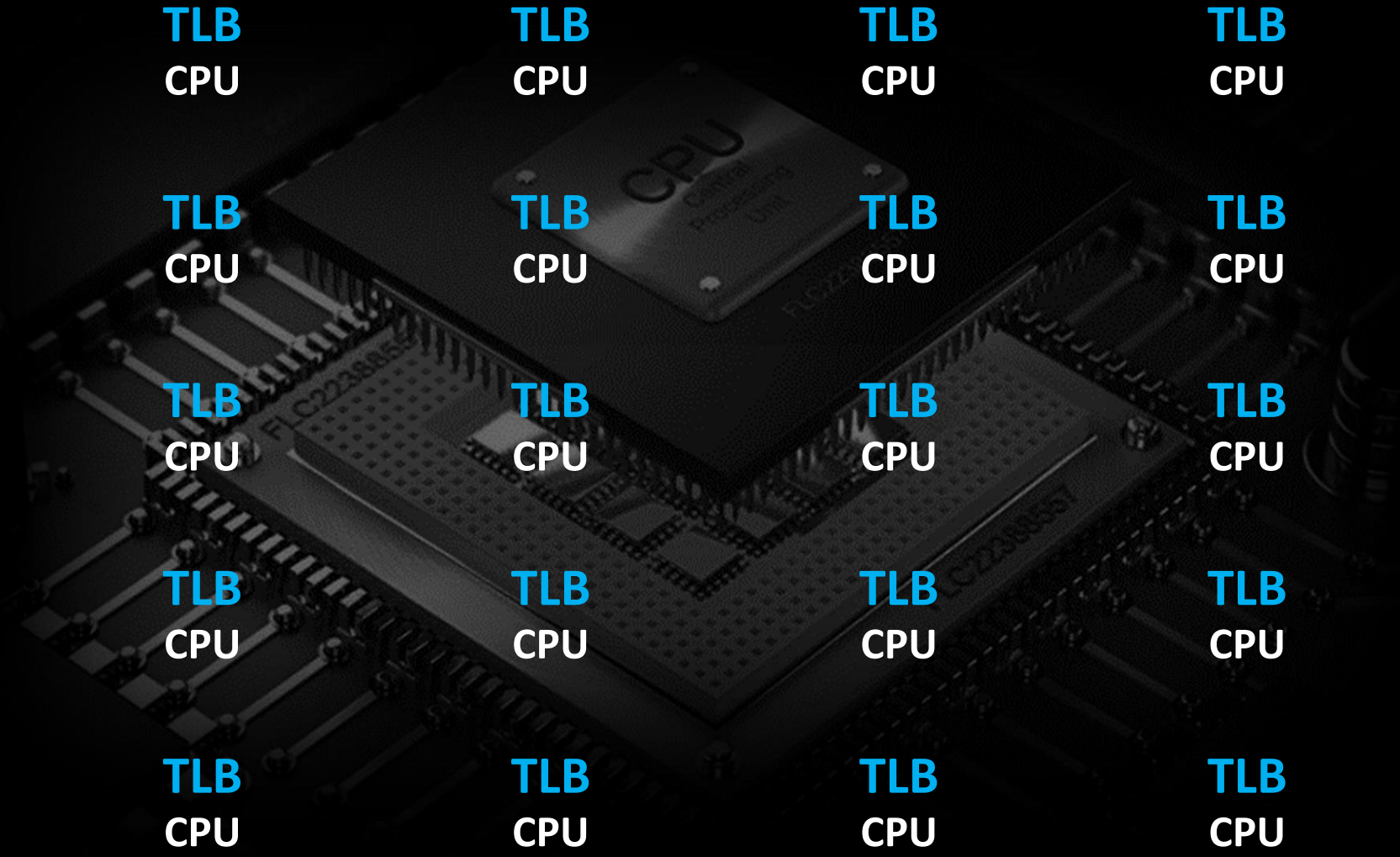
**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU



TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

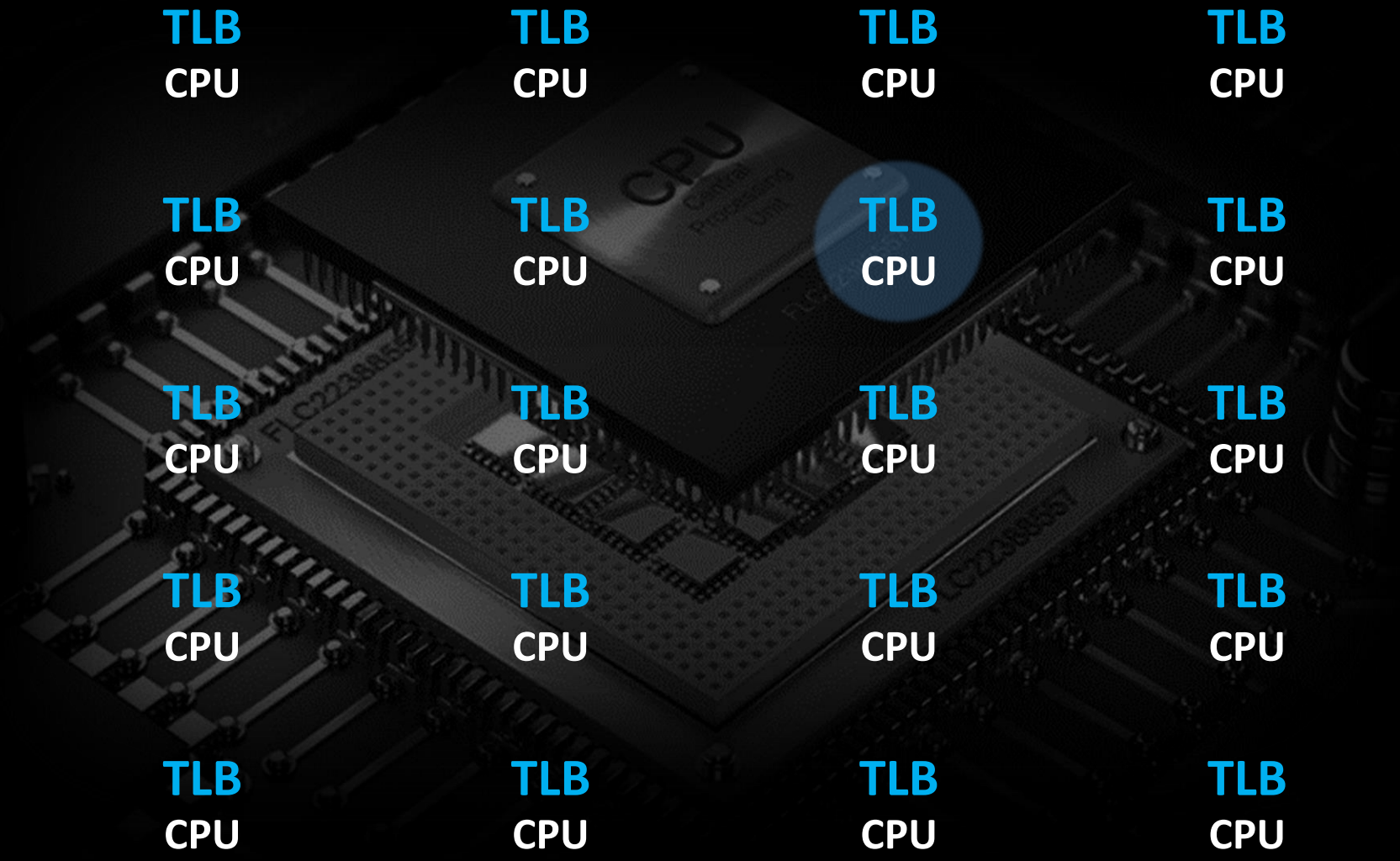
TLB  
CPU

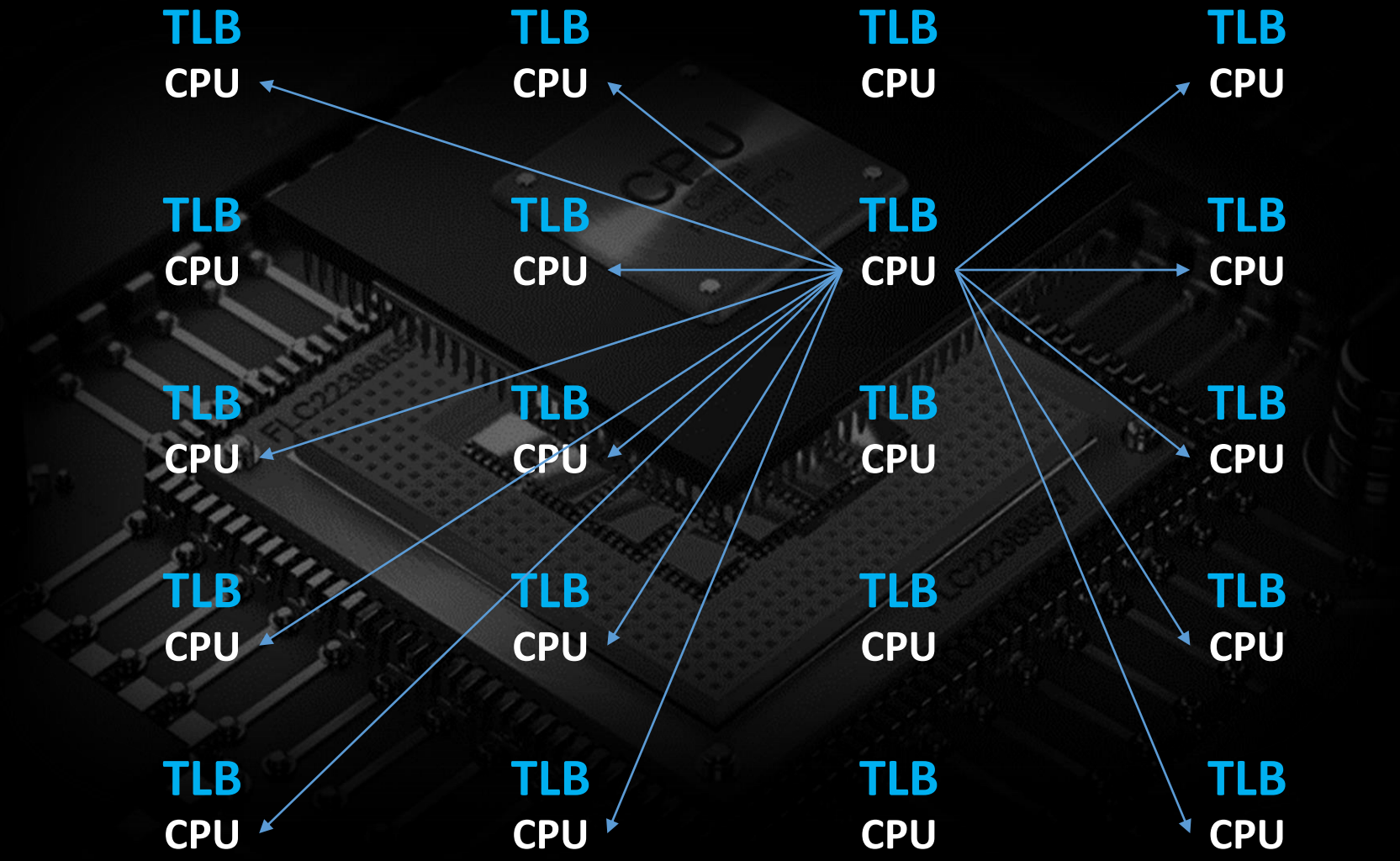
TLB  
CPU

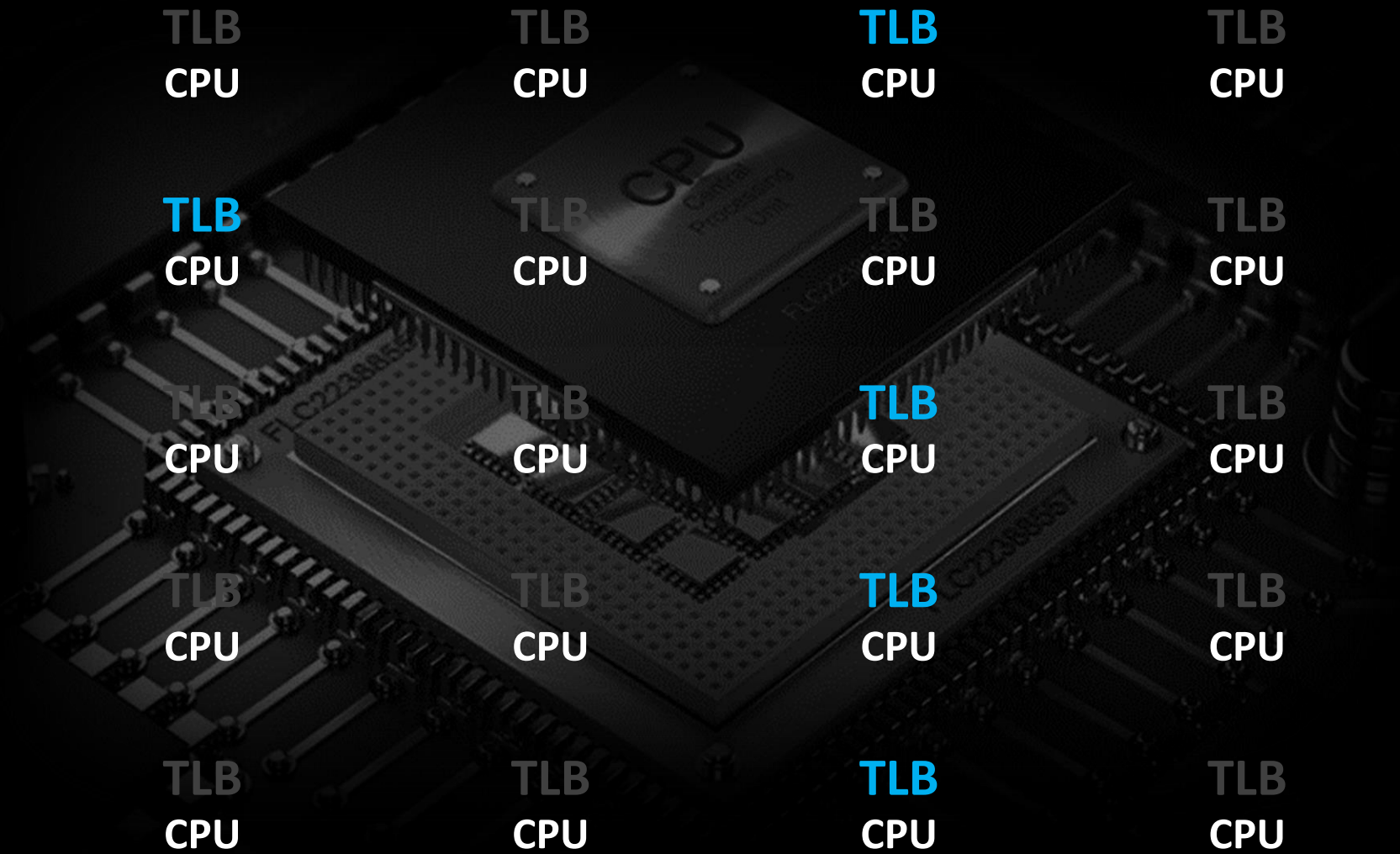
TLB  
CPU

TLB  
CPU

TLB  
CPU







TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

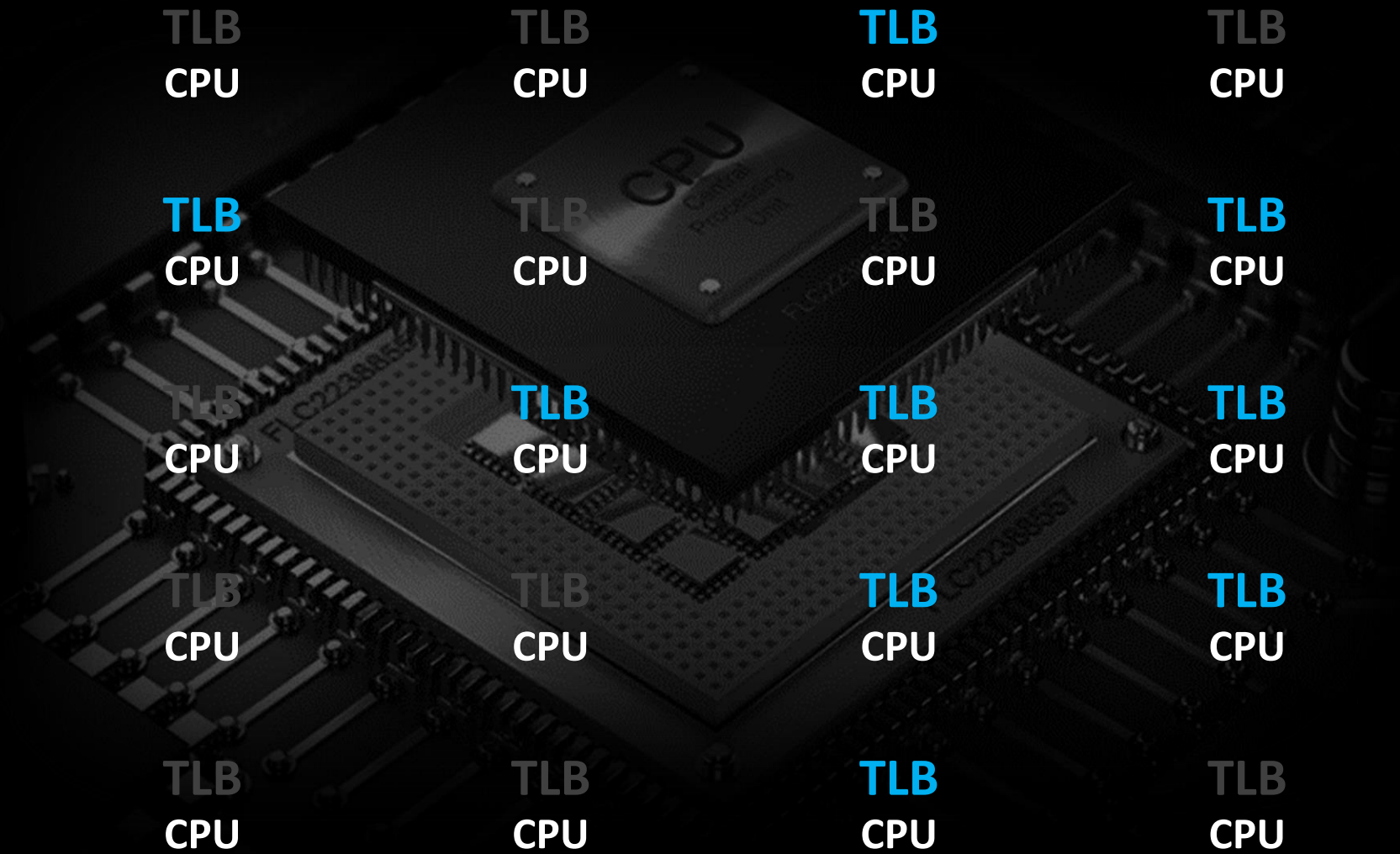
TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU





TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

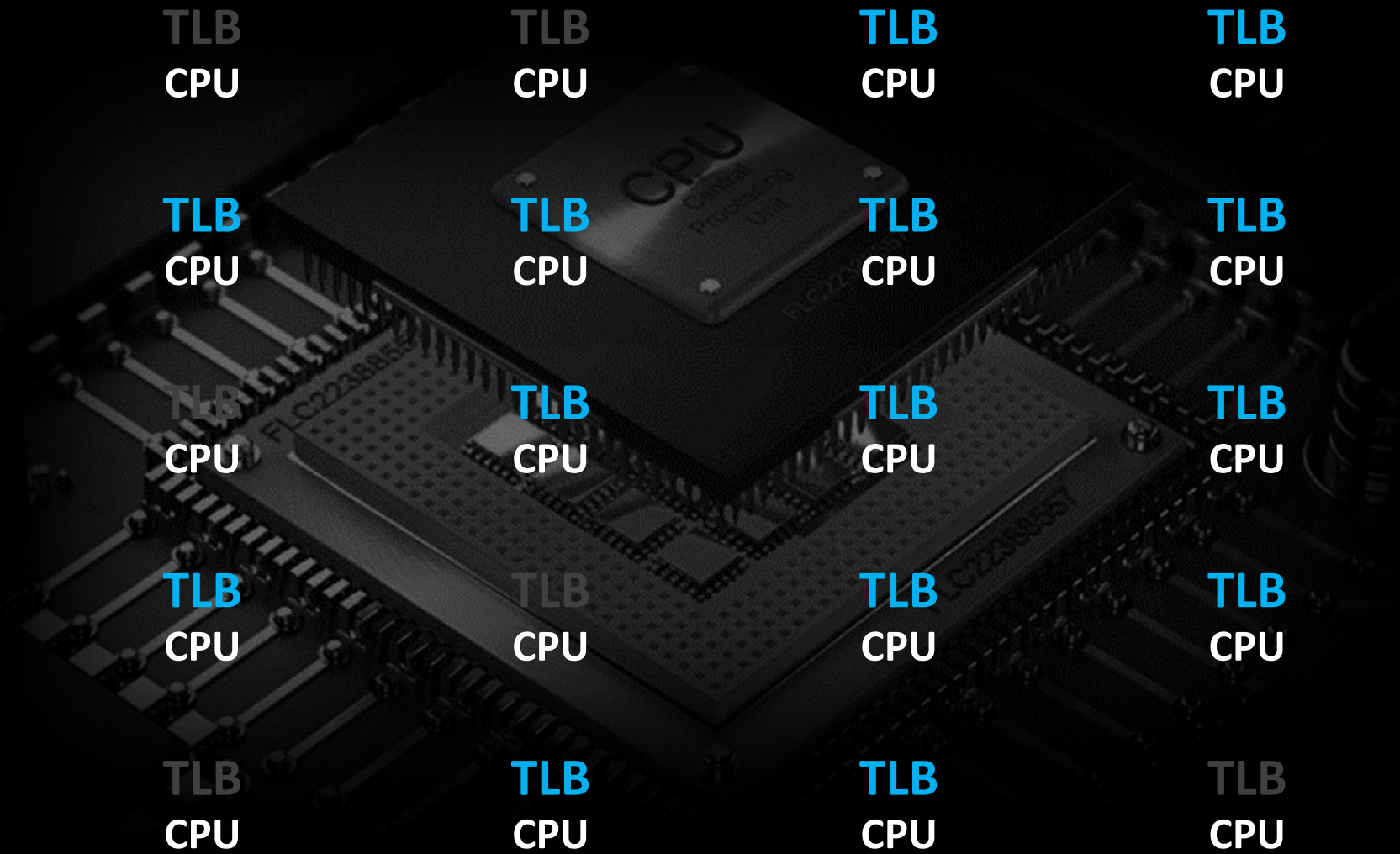
TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU



TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU

TLB  
CPU



**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

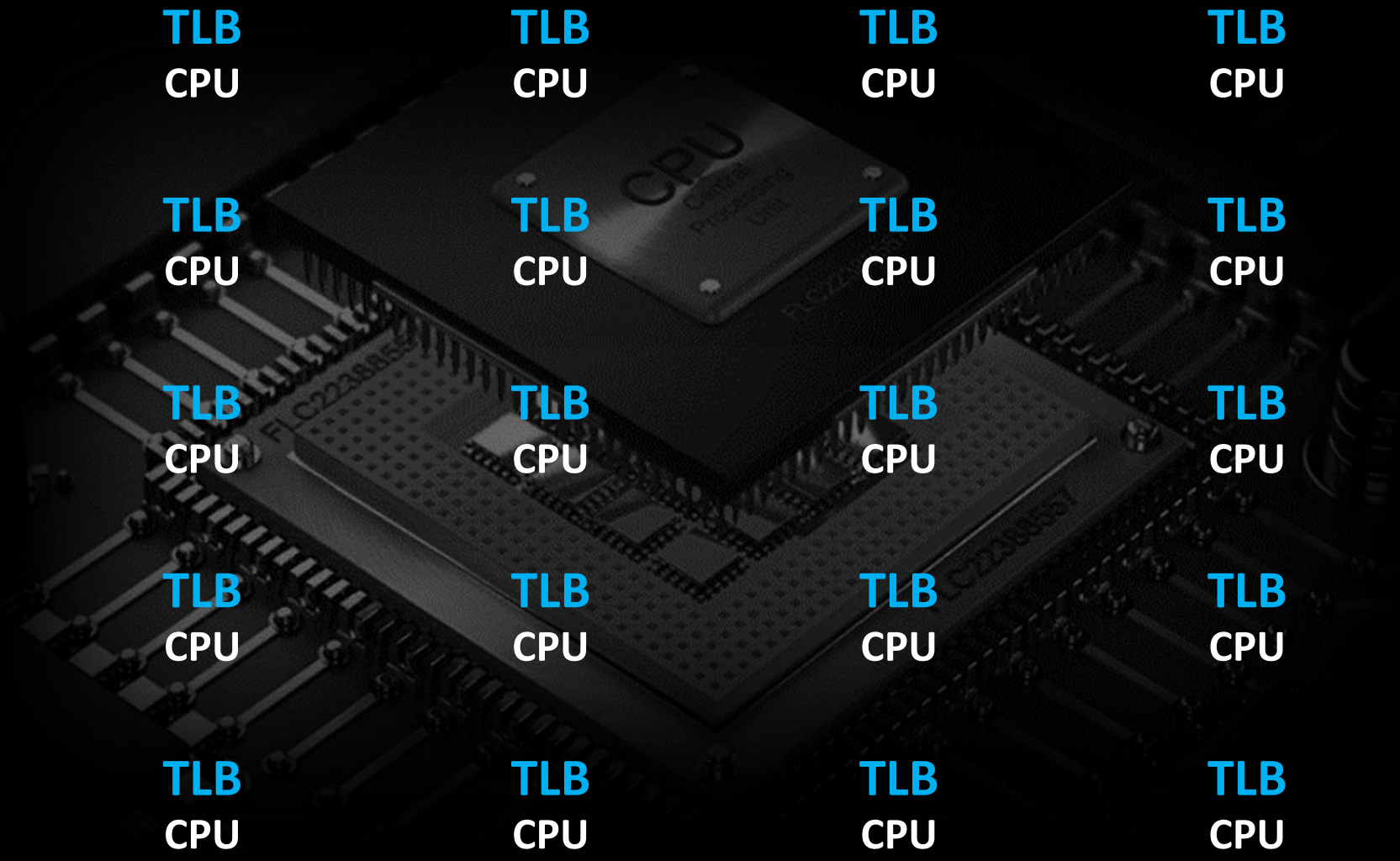
**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU

**TLB**  
CPU



ME

Is it required to perform TLB shutdown when unmapping virtual space?

CHAT GPT

Yes, performing a TLB shutdown is typically required when unmapping virtual space to ensure that all stale TLB entries are invalidated. This prevents the CPU from using old, invalid mappings and maintains consistency across multiple processors. The operating system's role is to manage this process, ensuring correctness and synchronization across the system.

While the workload of llama.cpp runs for 4000 secs...

```
$ grep TLB /proc/interrupts
```

TLB:

```
80911532 93691786 100296251 111062810 109769109 109862429
108968588 119175230 115779676 118377498 119325266 120300143
124514185 116697222 121068466 118031913 122660681 117494403
121819907 116960596 120936335 117217061 118630217 122322724
119595577 111693298 119232201 120030377 115334687 113179982
118808254 116353592 140987367 137095516 131724276 139742240
136501150 130428761 127585535 132483981 133430250 133756207
131786710 126365824 129812539 133850040 131742690 125142213
128572830 132234350 131945922 128417707 133355434 129972846
126331823 134050849 133991626 121129038 124637283 132830916
126875507 122322440 125776487 124340278 TLB shutdowns
```



# LUF's Goal

# TLB Shutdown

A method to invalidate TLBs involved.

Request TLB flush to CPUs.

Pay the cost of IPI(Inter Processor **Interrupt**).

Pay the cost of **TLB miss**.

# TLB Shutdown

A method to invalidate TLBs involved.

Request TLB flush to CPUs.

~~Pay the cost of~~ IPI(Inter Processor **Interrupt**).

~~Pay the cost of~~ **TLB** miss.

# TLB Shutdown

A method to invalidate TLBs involved.

Request TLB flush to CPUs.

~~Pay the cost of IPI(Inter Processor Interrupt).~~ **Skip**

~~Pay the cost of TLB hits.~~ **Keep**



# LUF's Interest

# Unmap Cases

A user explicitly unmaps using e.g. `munmap()`.

Page cache cleans folios e.g. `folio_mkclean()`.

Page reclaim unmaps to free pages.

Page migration unmaps to move pages.

And so on...

# Unmap Cases

~~A user explicitly unmaps using e.g. `munmap()`.~~

~~Page cache cleans folios e.g. `folio_mkclean()`.~~

Page reclaim unmaps to free pages.

Page migration unmaps to move pages.

~~And so on...~~

# Unmap Cases

~~A user explicitly unmaps using e.g. `munmap()`.~~

~~Page cache cleans folios e.g. `folio_mkclean()`.~~

Page reclaim unmaps to free pages.

Page migration unmaps to move pages.

~~And so on...~~

# Page Migration Steps

Unmap mappings of the source page.

TLB shutdown to invalidate TLBs involved.

Allocate a new page.

Copy the source page to the new page.

Free the source page.

# Page Migration Steps

Unmap mappings of the source page.

TLB shutdown to invalidate TLBs involved.

Allocate a new page.

Copy the source page to the new page.

Free the source page.

# Page Migration Steps

Unmap mappings of the source page.

~~TLB shutdown~~ **Skip** ~~invalidate TLBs involved.~~

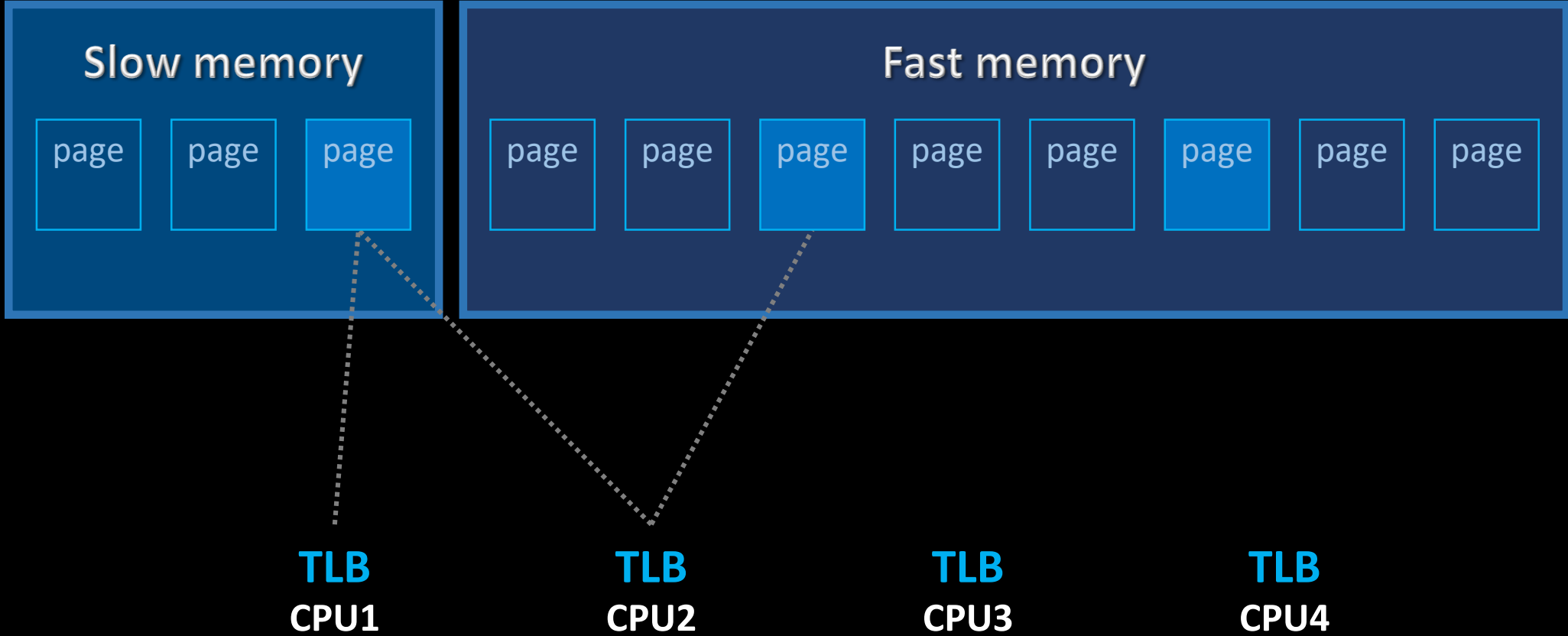
Allocate a new page.

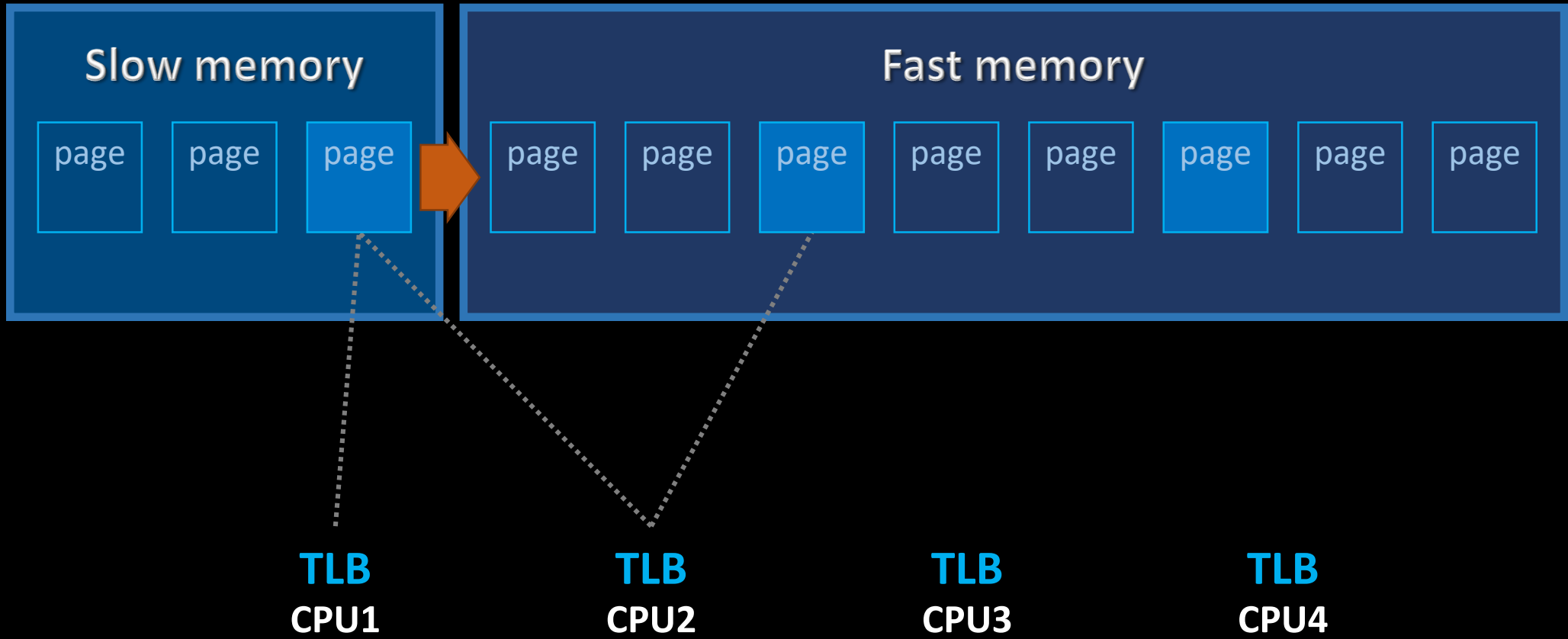
Copy the source page to the new page.

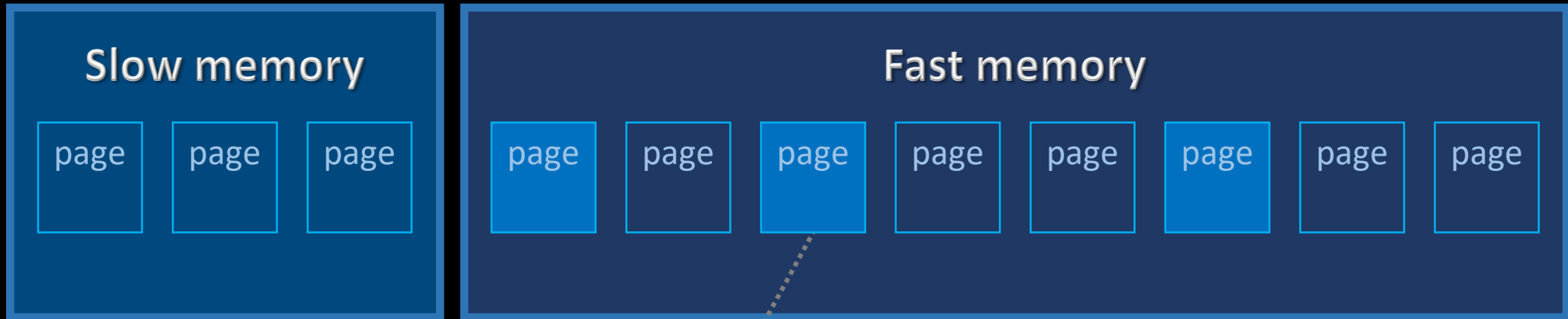
Free the source page.



**How?**







**TLB shutdown**

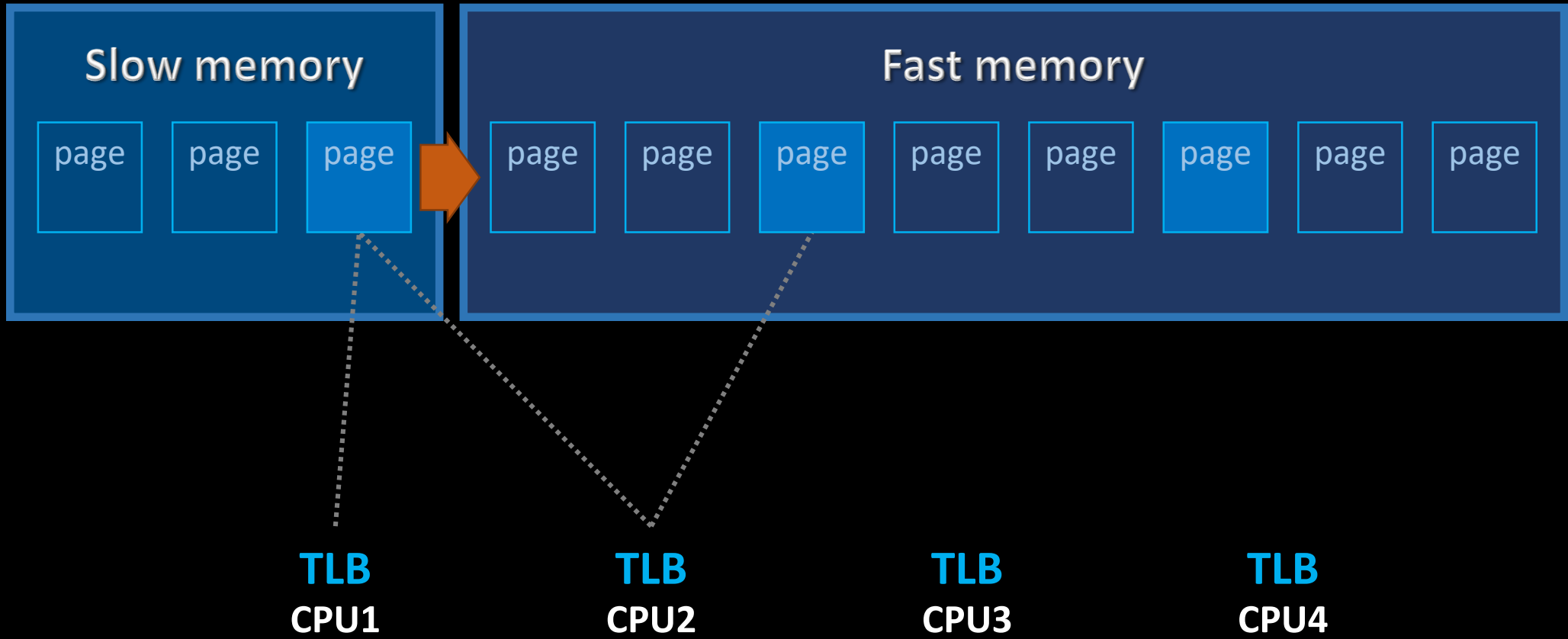
TLB  
CPU1

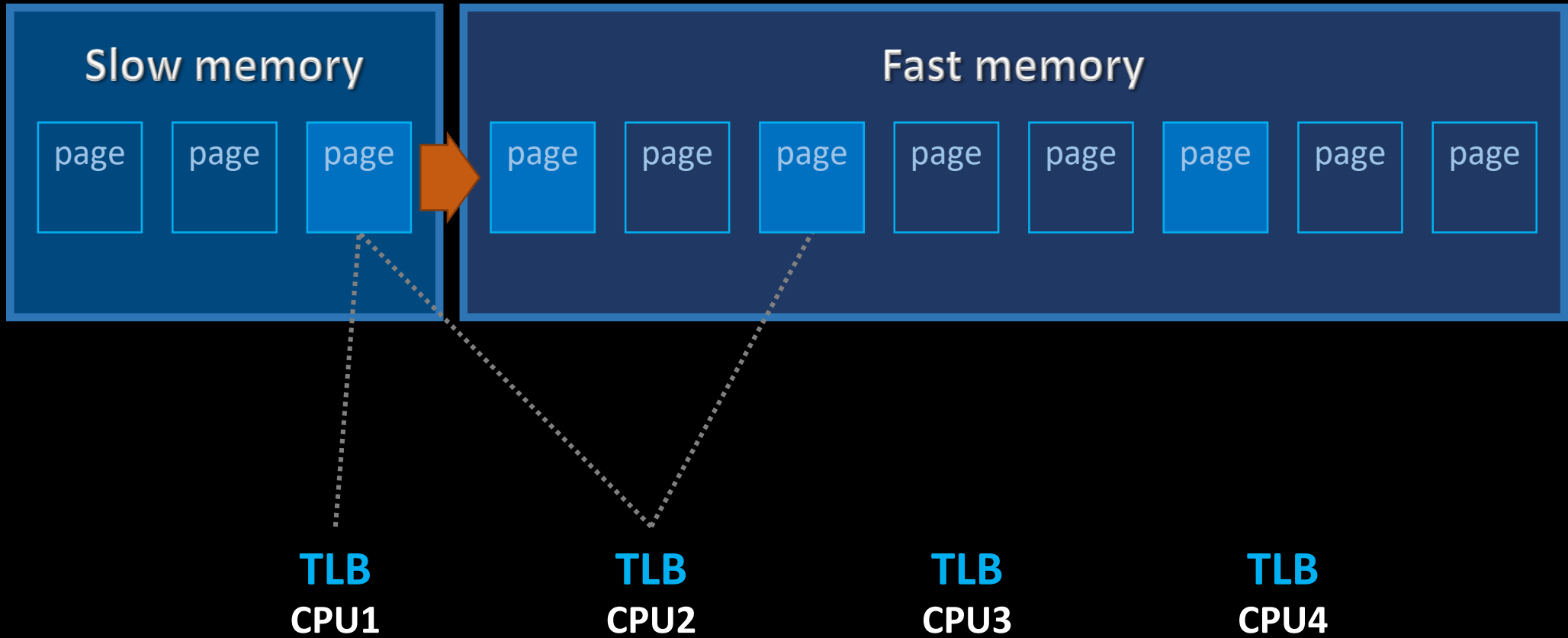
TLB  
CPU2

TLB  
CPU3

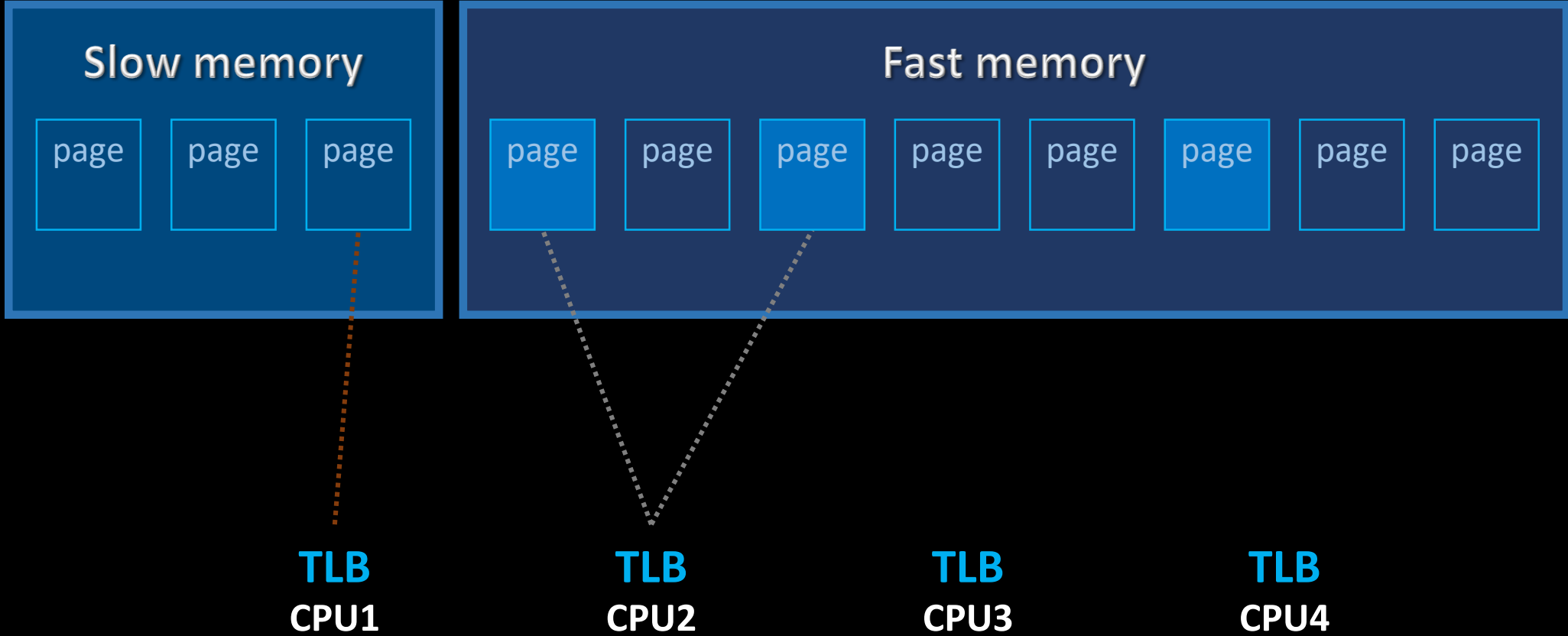
TLB  
CPU4

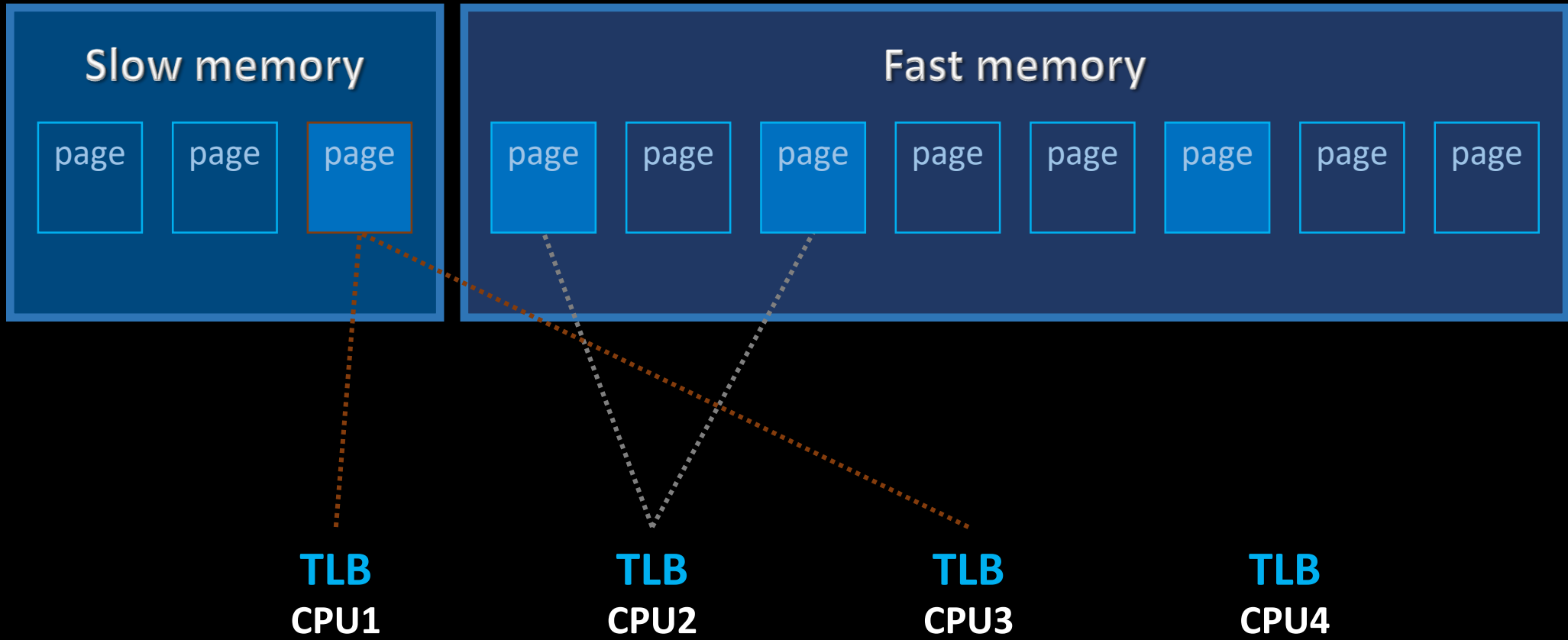
**What if w/o TLB shutdown?**

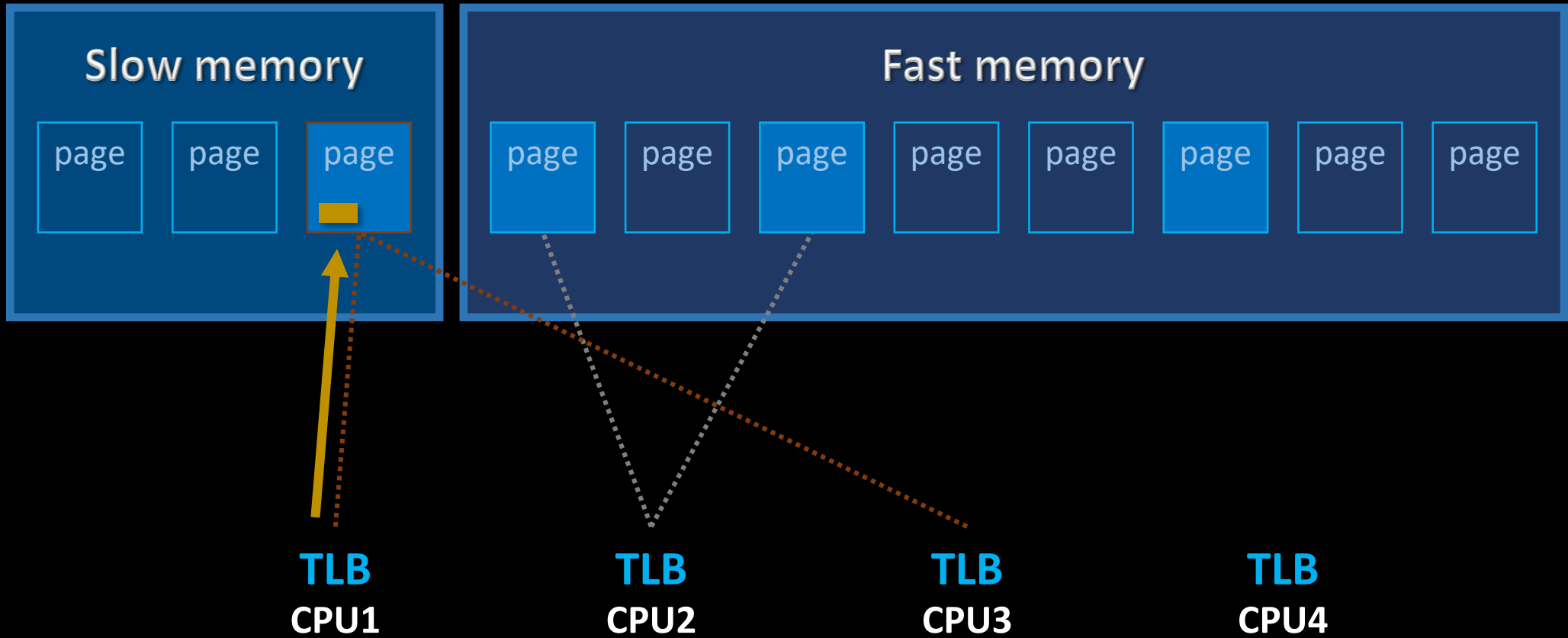


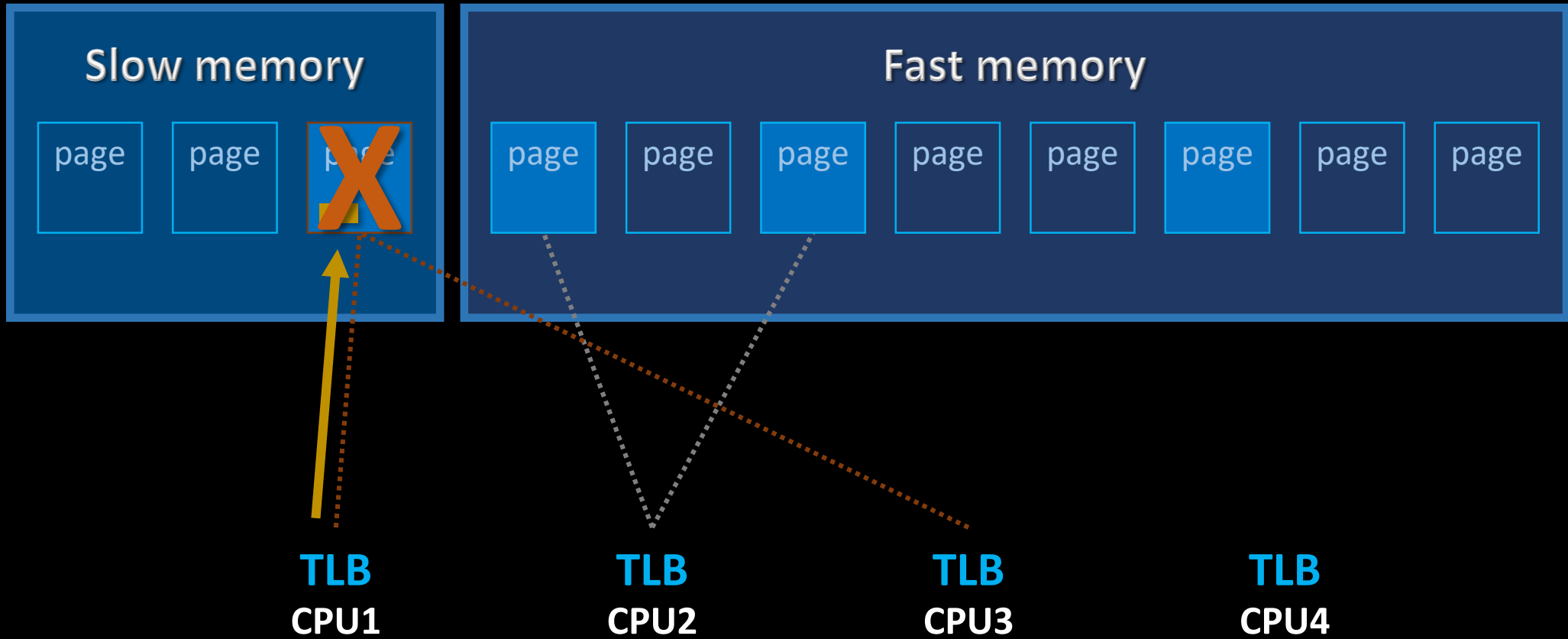




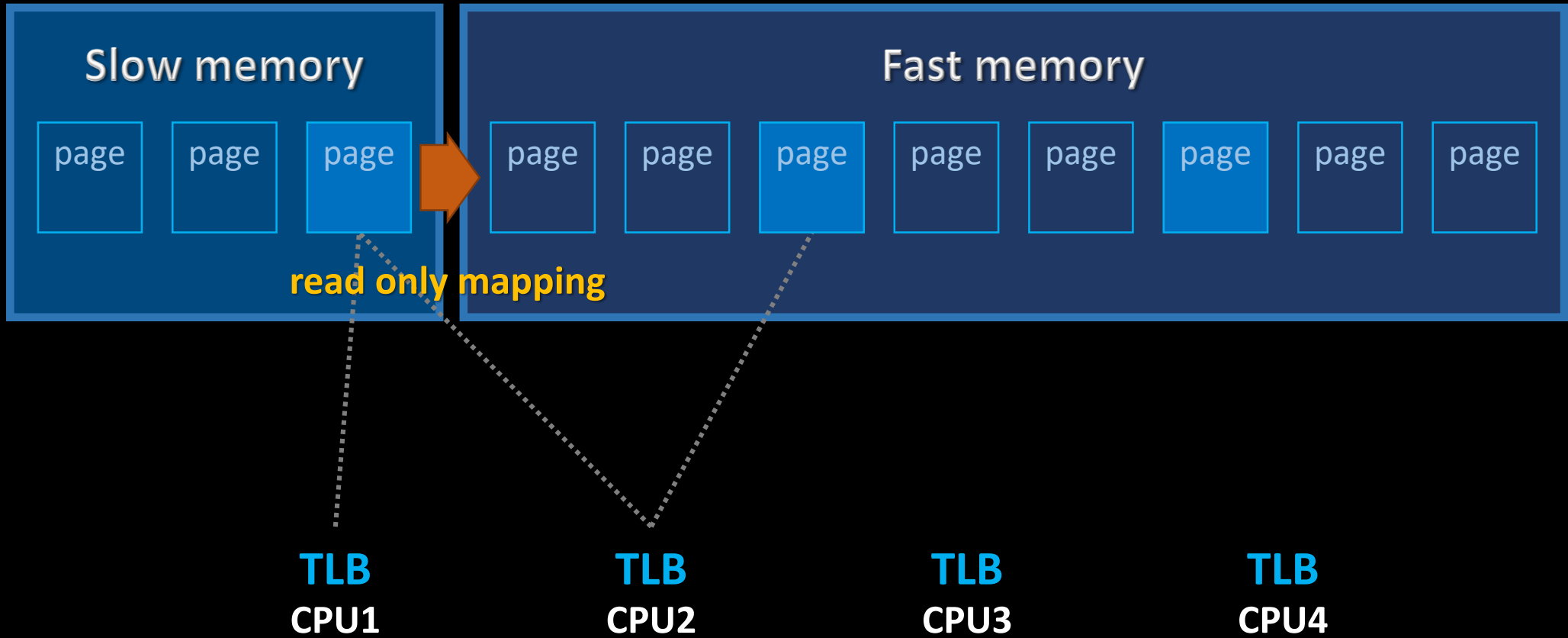


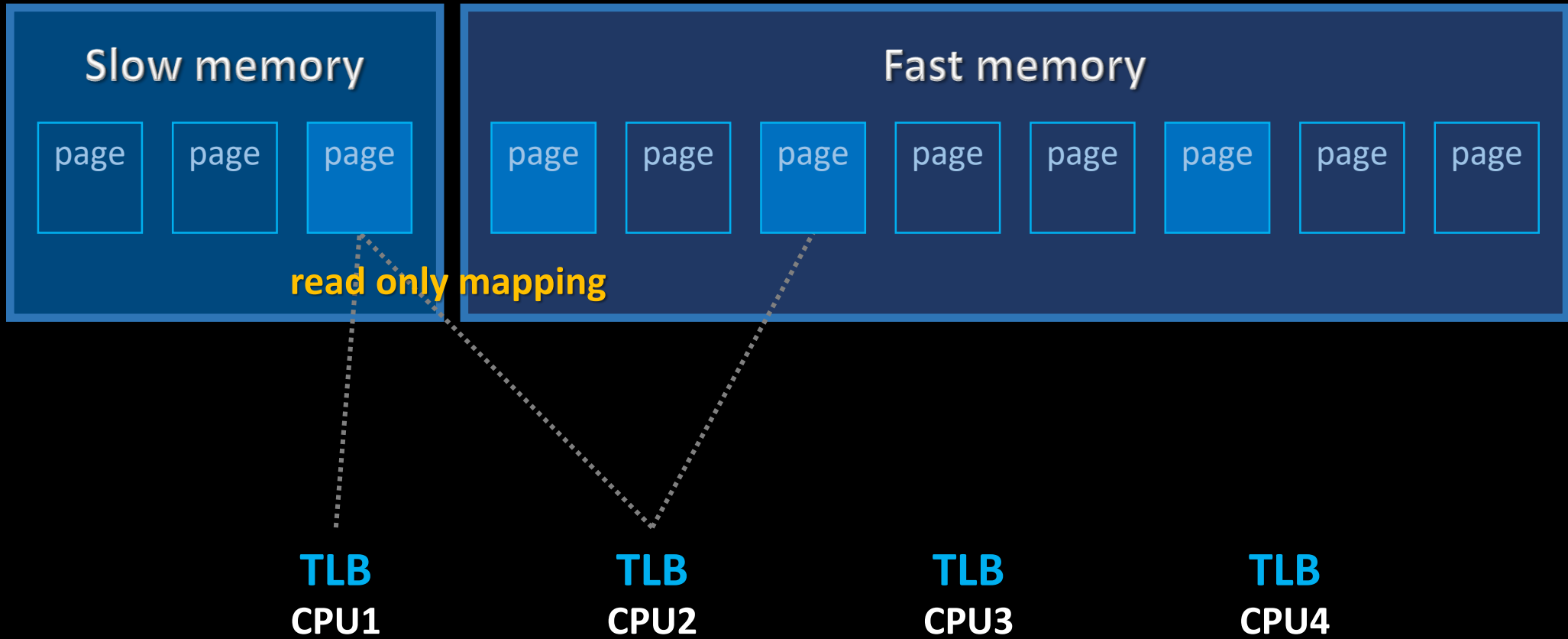


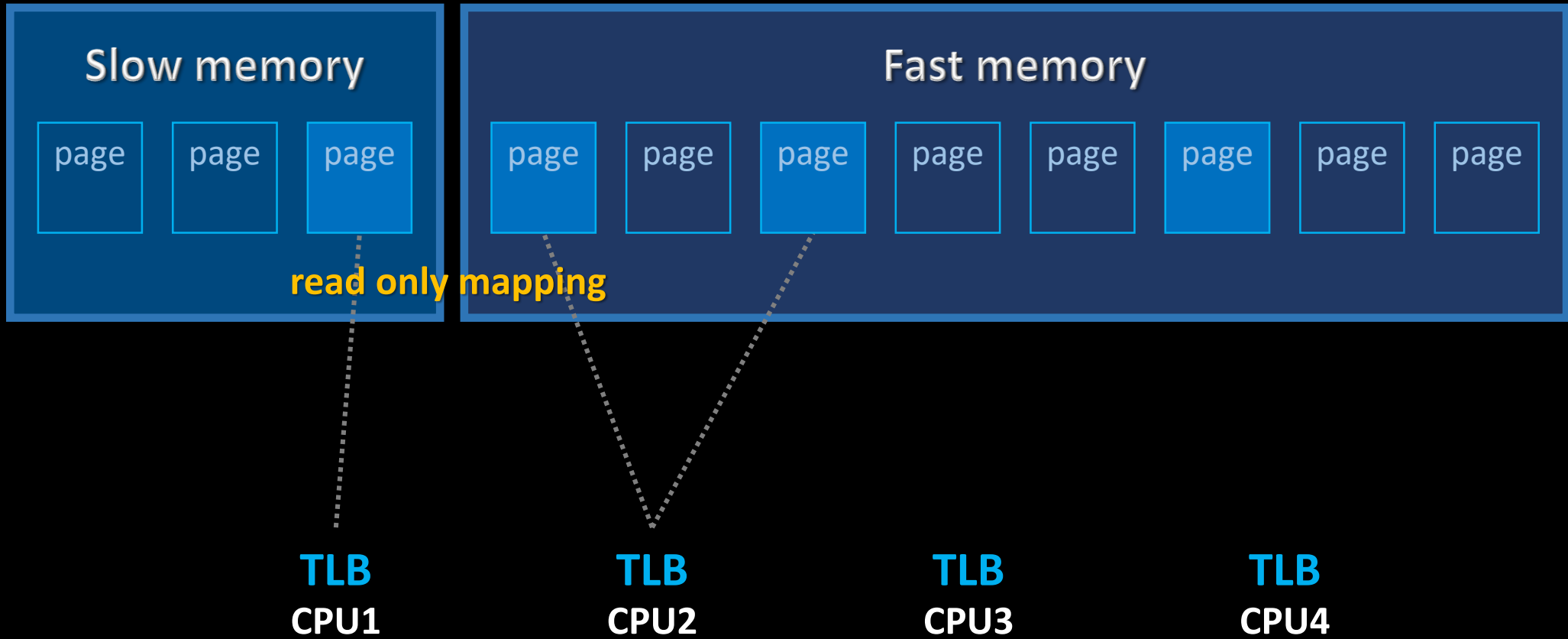




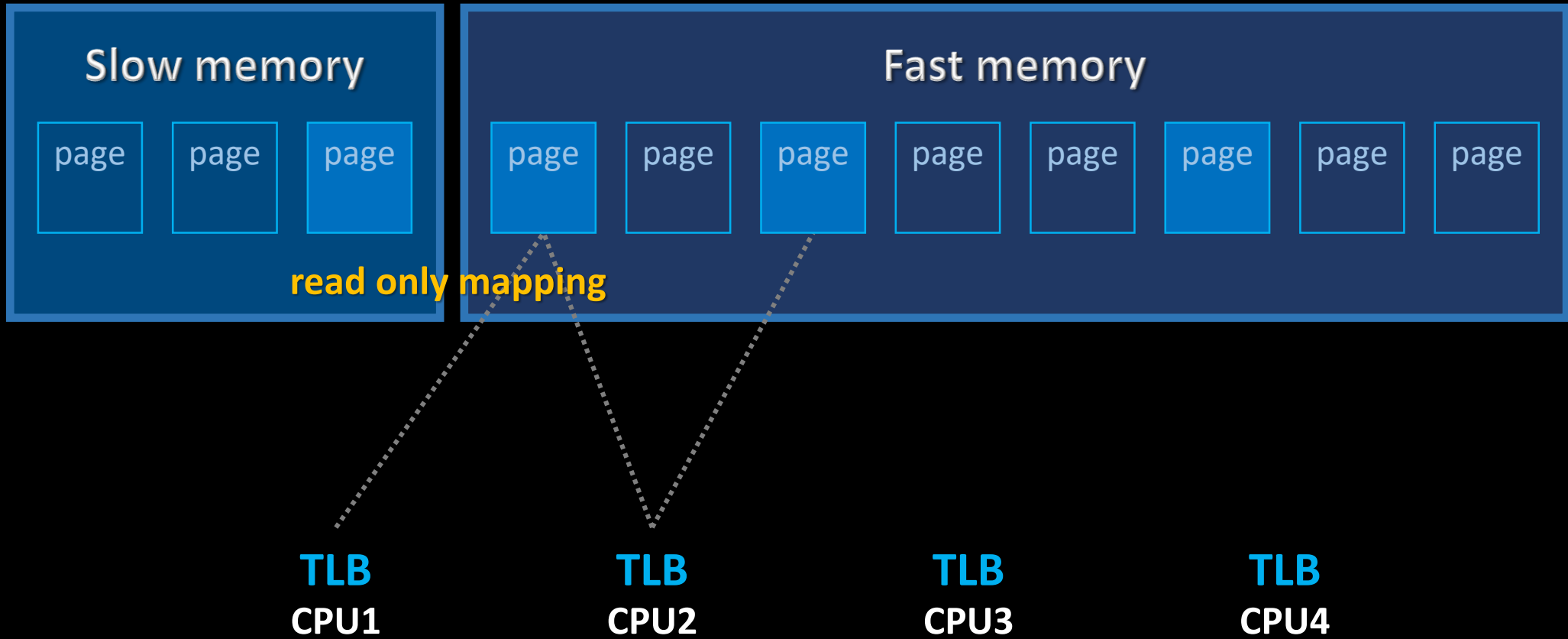
**What if it's read only mapping?**

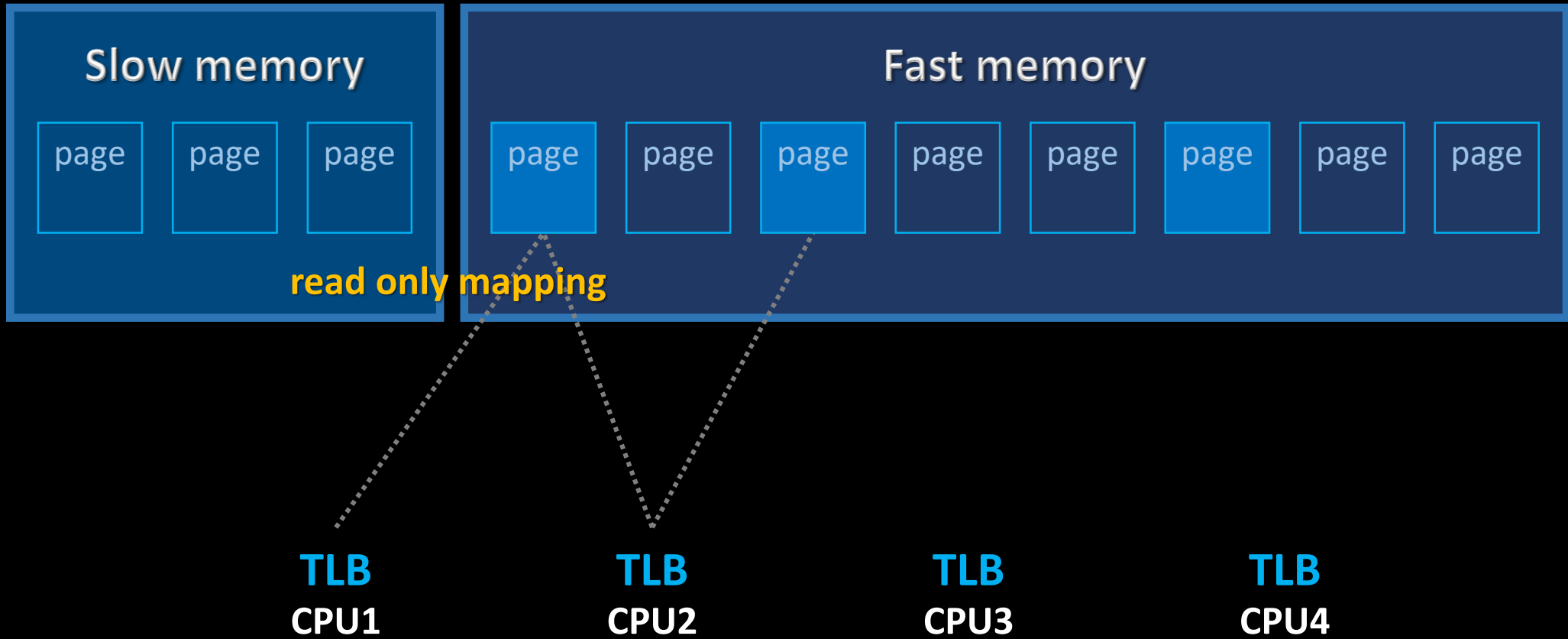












**Safe, even w/o TLB shutdown!**

So...

For **read only** mappings,

Keep **read copies** even after unmapping,

So that some CPUs can run with **stale TLB entries**.

Hopefully skip whole TLB shutdown.

Or partially **TLB shutdown** when it's **inevitable**.

For read only mappings,

Keep read copies even after unmapping,

## **How LUF works**

So that some CPUs can run with stale TLB entries.

Hopefully skip whole TLB shutdown.

Or partially TLB shutdown when it's inevitable.

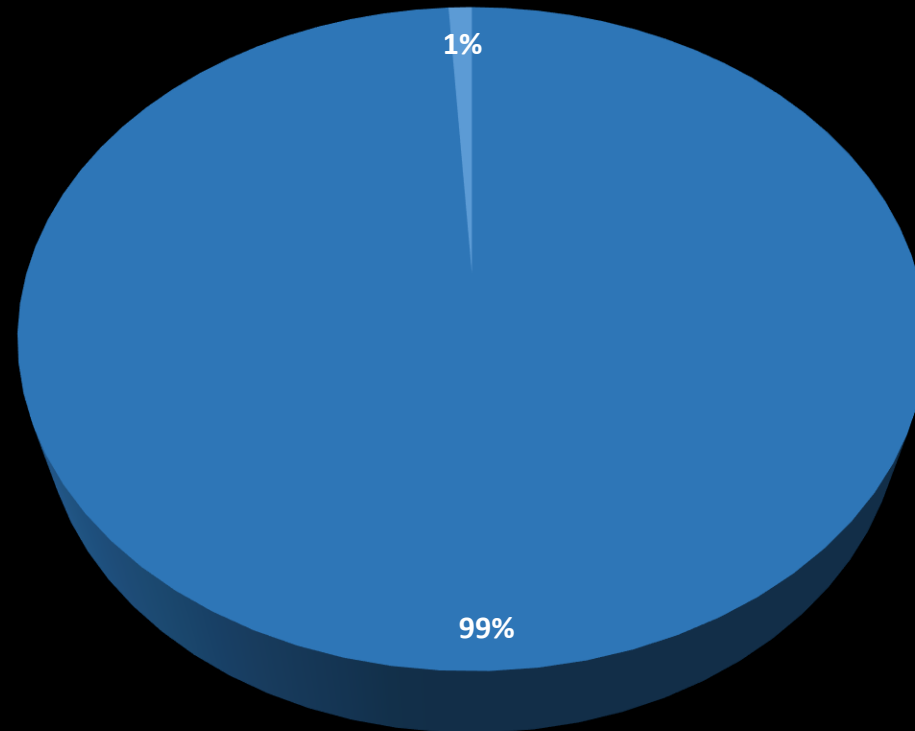
**But...**

**Worthy only with RO mappings**



# RO : RW Ratio

## Collected on Unmapping



■ RO ■ RW

# Dive Into LUF

Approaches

Considerations

Measurement

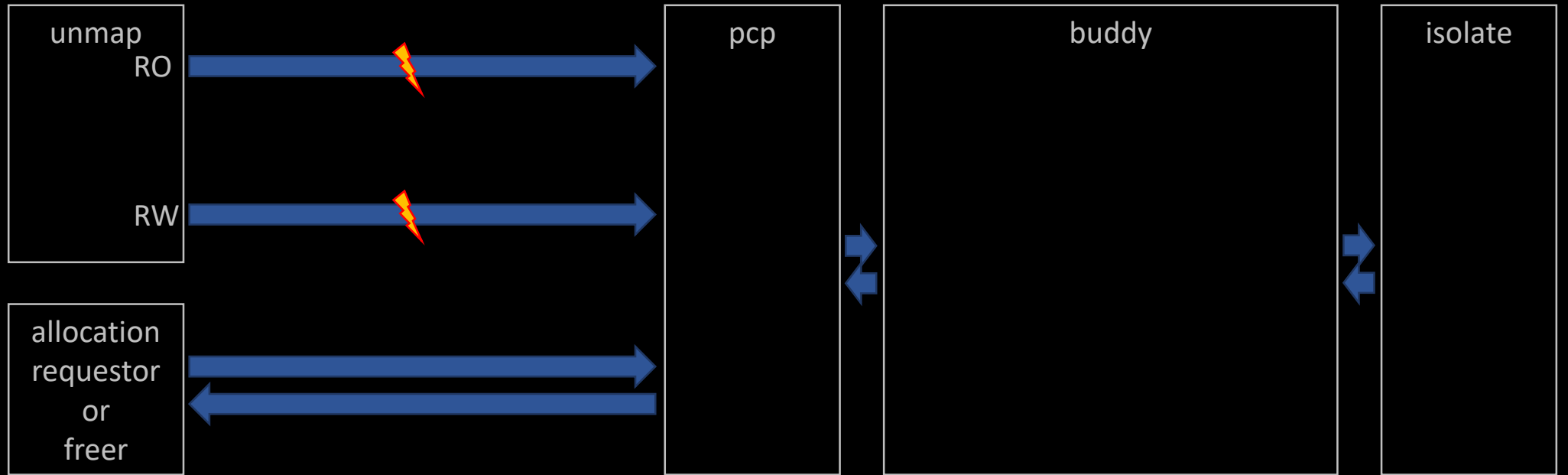
- 1 Keep read copies **before freeing**
- 2 Keep read copies **residing allocator**


# Keep Read Copies **Before Freeing**

Approaches

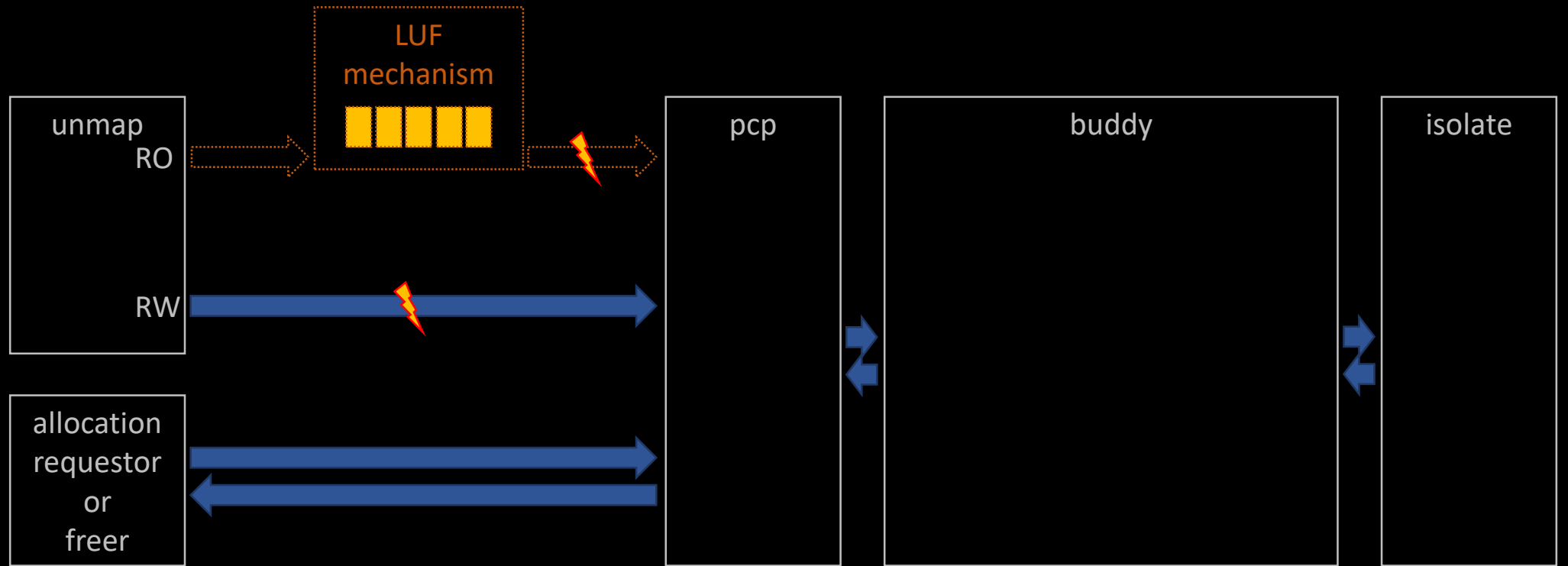
Considerations


Measurement



 TLB shutdown

# Keep Read Copies Before Freeing



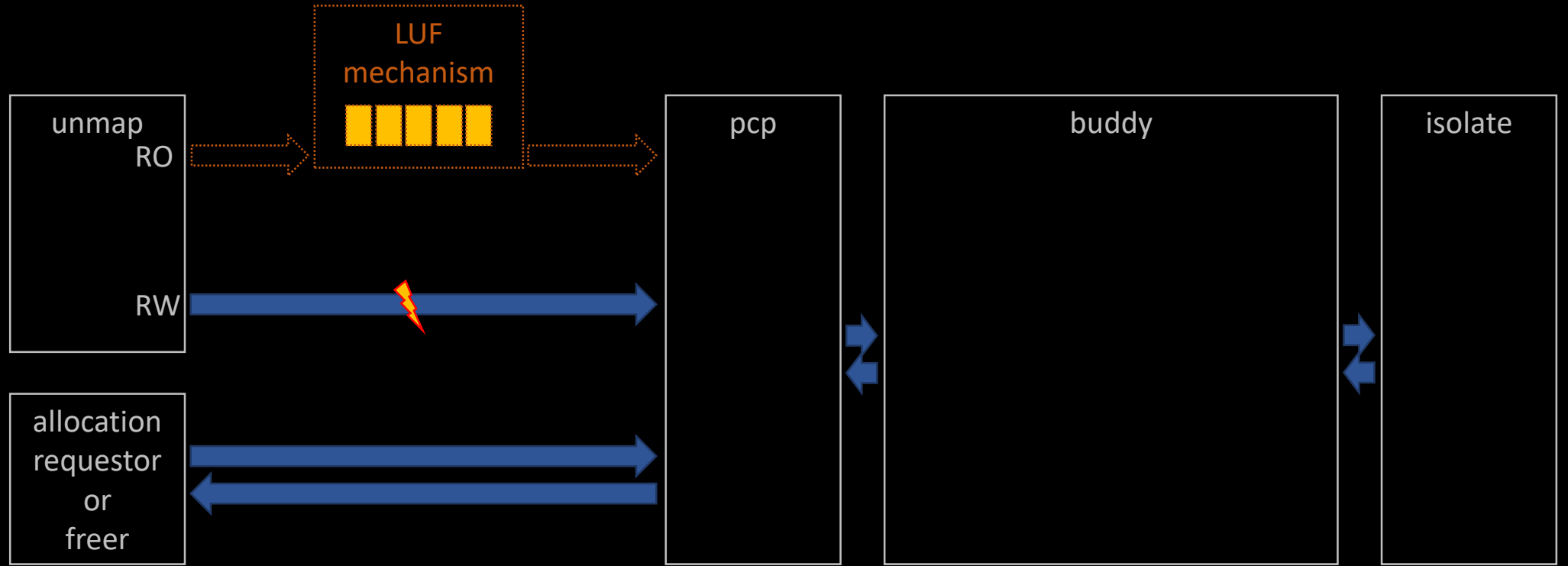
 TLB shutdown


Approaches

Considerations

Measurement

# Keep Read Copies Before Freeing



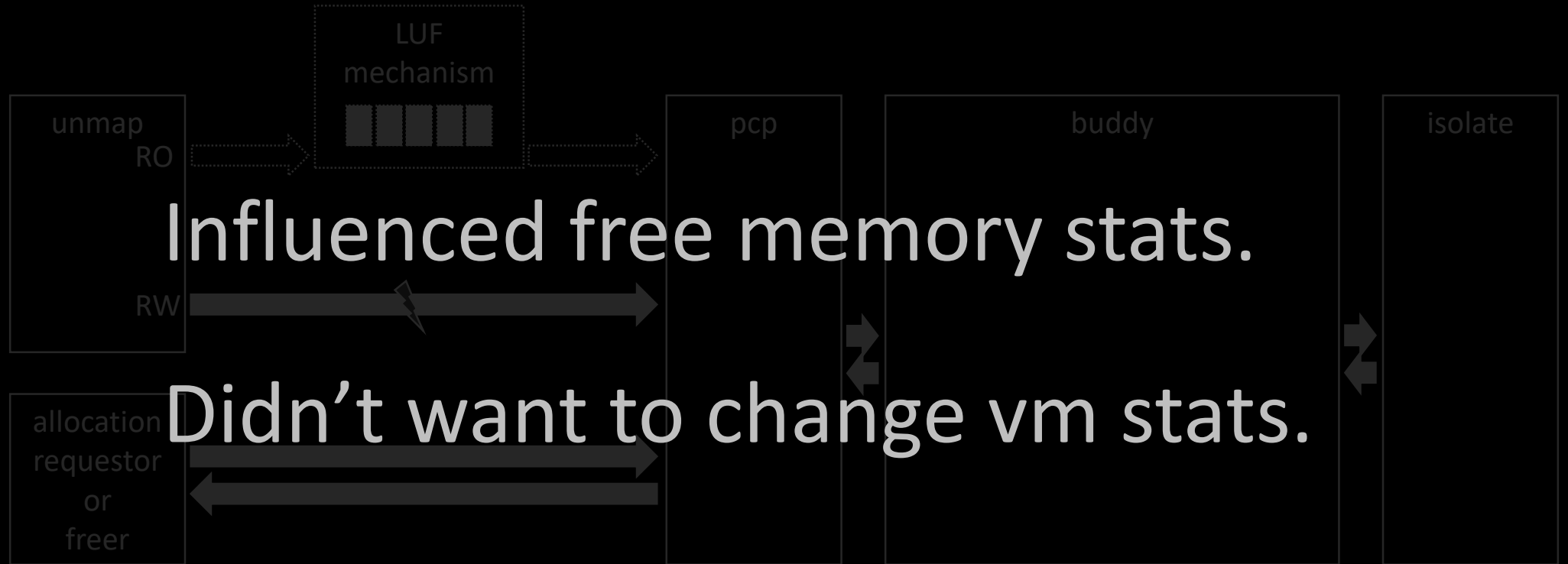
 TLB shutdown

Approaches

Considerations

Measurement

# Keep Read Copies Before Freeing



⚡ TLB shutdown

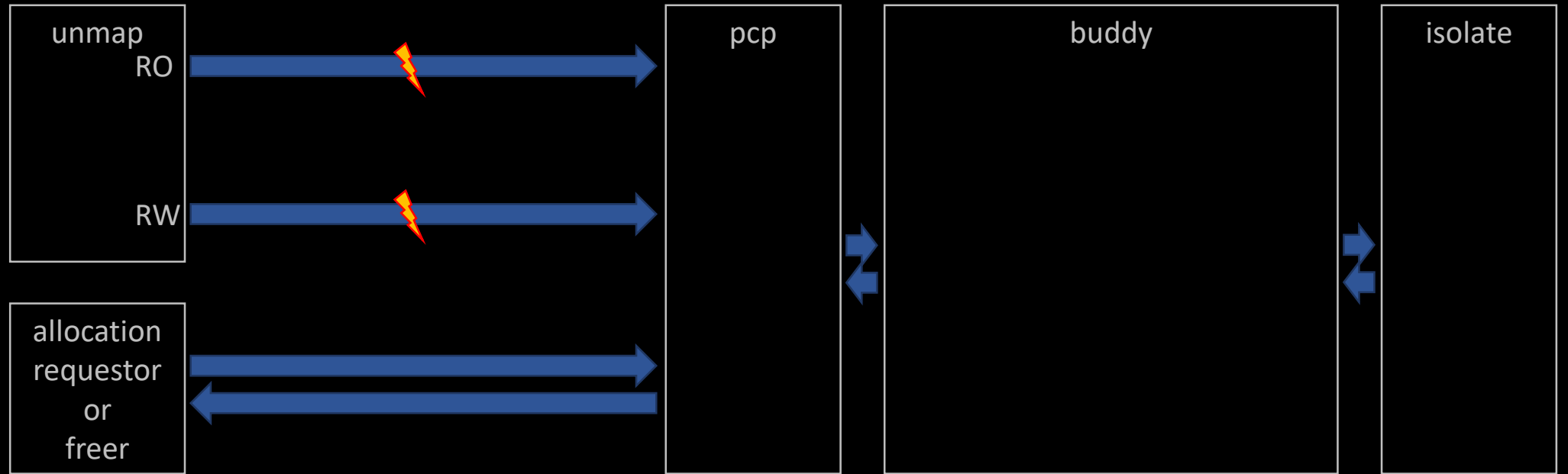
- Approaches
- Considerations
- Measurement


# Keep Read Copies Residing Allocator

Approaches

Considerations

Measurement



 TLB shutdown

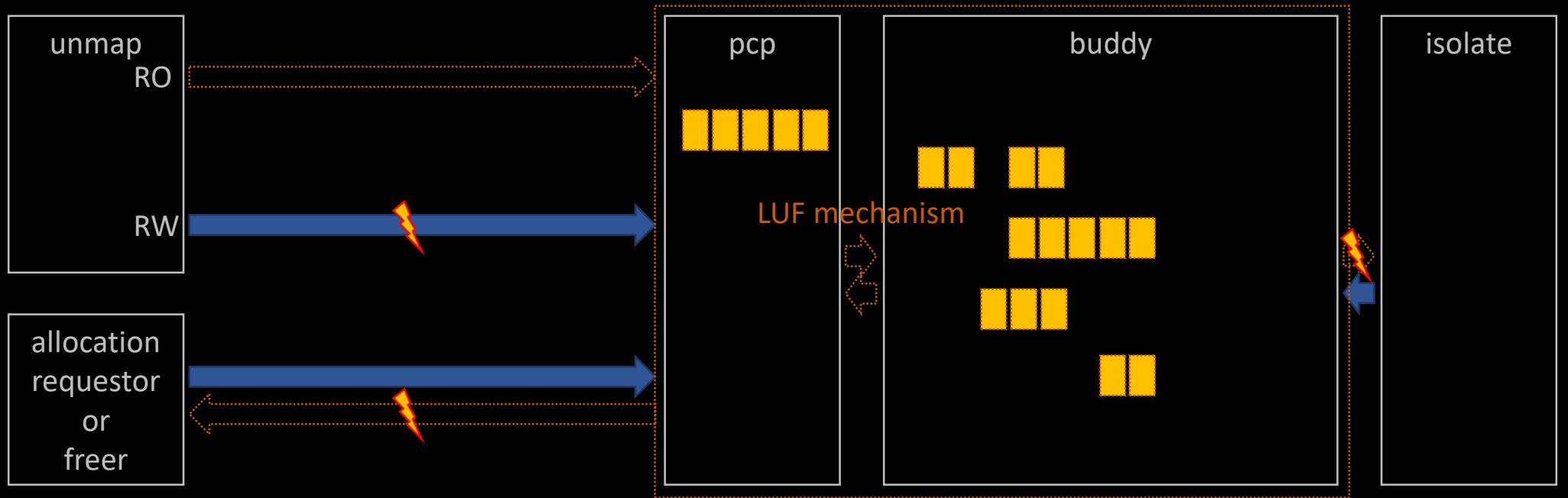



# Keep Read Copies Residing Allocator

Approaches

Considerations

Measurement



 TLB shutdown

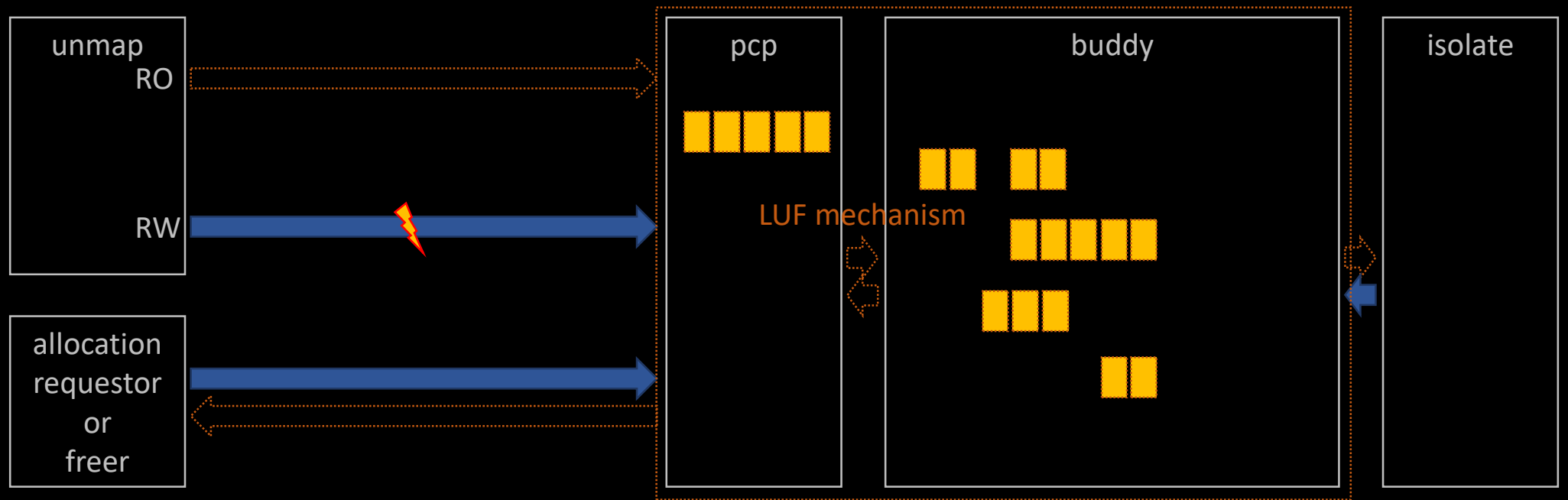
- ※ `want_init_on_free()` should be false for LUF to work.
- ※ `should_skip_kasan_poison()` should be true for LUF to work.

# Keep Read Copies Residing Allocator

Approaches

Considerations

Measurement



※ want\_init\_on\_free() should be false for LUF to work.

※ should\_skip\_kasan\_poison() should be true for LUF to work.

Approaches

Considerations

Measurement

Limit to anon pages + file pages

struct **page** : manages each page

struct **mm\_struct** : manages virtual space per task

struct **address\_space** : manages page cache per file

Approaches

Considerations

Measurement

- 1 Track CPUs to TLB shutdown per `page`
- 2 Track CPUs to TLB shutdown per `mm_struct`
- 3 Track CPUs to TLB shutdown per `address_space`

Approaches

Considerations

Measurement

- 4 When to TLB shutdown per `page`
- 5 When to TLB shutdown per `mm_struct`
- 6 When to TLB shutdown per `address_space`

- 1 Track CPUs to TLB shutdown per page
- 2 Track CPUs to TLB shutdown per mm\_struct
- 3 Track CPUs to TLB shutdown per address\_space

Approaches

Considerations

Measurement

- 4 When to TLB shutdown per **page**
- 5 When to TLB shutdown per **mm\_struct**
- 6 When to TLB shutdown per **address\_space**

- 1 Track CPUs to TLB shutdown per page
- 2 Track CPUs to TLB shutdown per mm\_struct
- 3 Track CPUs to TLB shutdown per address\_space

Approaches

Considerations

Measurement

- 4 When to TLB shutdown per page
- 5 When to TLB shutdown per mm\_struct
- 6 When to TLB shutdown per address\_space

# Page Allocation Steps under Pressure



Approaches

Considerations

Measurement

- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do TLB shutdown during reclaim.
- ⑥ Return pages on successful reclaim.
- ⑦ LUF does TLB shutdowns if needed on return.



# Page Allocation Steps under Pressure



- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do TLB shutdown during reclaim.
- ⑥ Return pages on successful reclaim.
- ⑦ LUF does TLB shutdowns if needed on return.

Approaches

Considerations

Measurement

# Page Allocation Steps under Pressure



Approaches

Considerations

Measurement

- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do TLB shutdown during reclaim.
- ⑥ Return pages on successful reclaim.
- ⑦ LUF does TLB shutdowns if needed on return.

# Page Allocation Steps under Pressure



Approaches

Considerations

Measurement

- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do TLB shutdown during reclaim.
- ⑥ Return pages on successful reclaim.
- ⑦ LUF does TLB shutdowns if needed on return.

# Page Allocation Steps under Pressure



- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do **TLB shutdown** during reclaim.
- ⑥ Return pages on successful reclaim.
- ⑦ LUF does TLB shutdowns if needed on return.

Approaches

Considerations

Measurement

# Page Allocation Steps under Pressure



- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do **TLB shutdown** during reclaim + **LUF**.
- ⑥ Return pages on successful reclaim.
- ⑦ LUF does TLB shutdowns if needed on return.

Approaches

Considerations

Measurement

# Page Allocation Steps under Pressure



- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do **TLB shutdown** during reclaim + **LUF**.
- ⑥ Return pages on successful reclaim.
- ⑦ `LUF` does TLB shutdowns if needed on return.

Approaches

Considerations

Measurement

# Page Allocation Steps under Pressure



- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do **TLB shutdown** during reclaim + **LUF**.
- ⑥ Return pages on successful reclaim.
- ⑦ **LUF** does **TLB shutdowns** if needed on return.

Approaches

Considerations

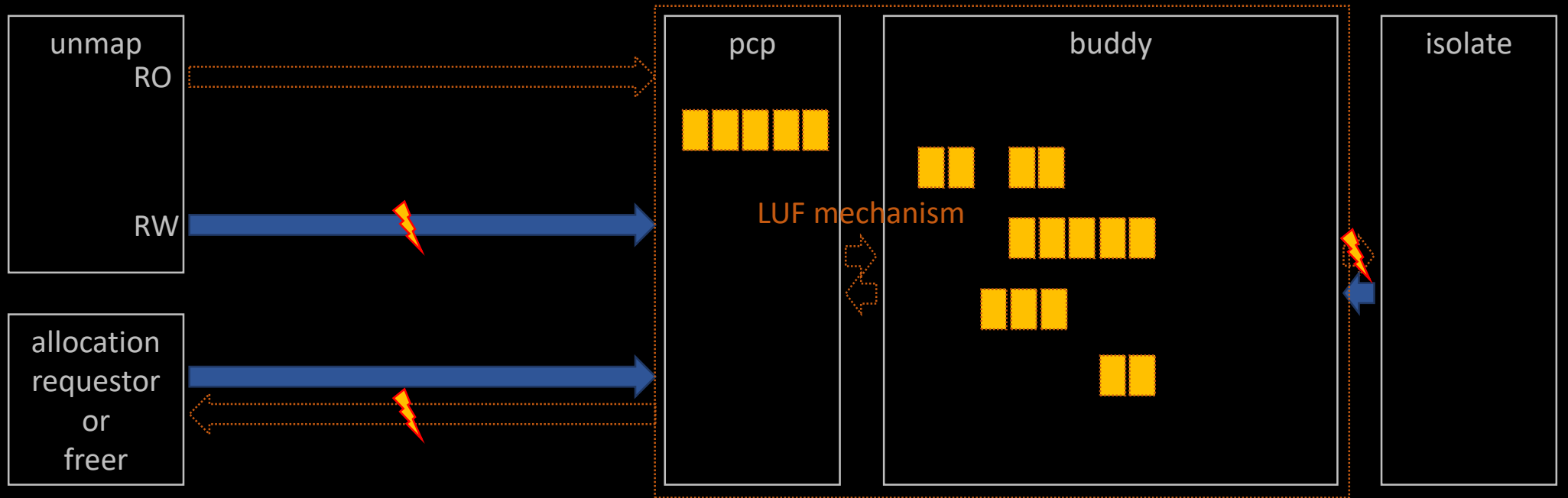
Measurement


# Keep Read Copies Residing Allocator

Approaches

Considerations

Measurement



 TLB shutdown

- ※ `want_init_on_free()` should be false for LUF to work.
- ※ `should_skip_kasan_poison()` should be true for LUF to work.

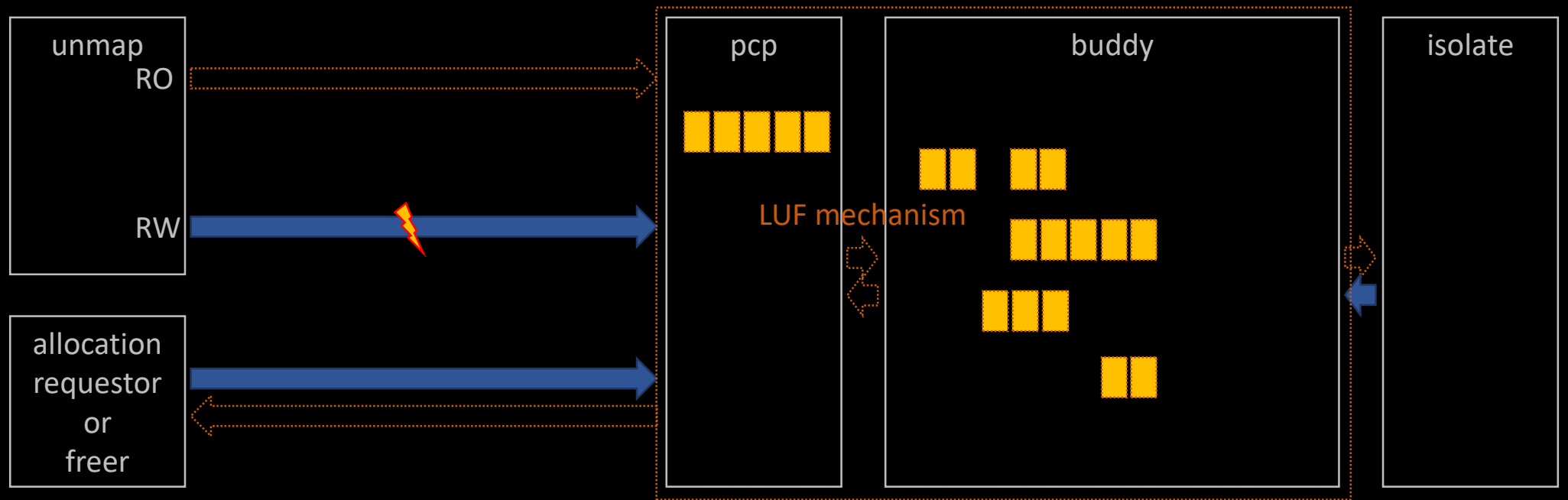



# Keep Read Copies Residing Allocator

Approaches

Considerations

Measurement



 TLB shutdown

- ※ `want_init_on_free()` should be false for LUF to work.
- ※ `should_skip_kasan_poison()` should be true for LUF to work.

# Page Allocation Steps under Pressure



- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do **TLB shutdown** during reclaim + **LUF**.
- ⑥ Return pages on successful reclaim.
- ⑦ **LUF** does **TLB shutdowns** if needed on return.

Approaches

Considerations

Measurement

# Page Allocation Steps under Pressure



- ① Request page allocation.
- ② Check if there are enough free pages.
- ③ If hitting `WMARK_LOW`, let `kswapd` reclaim.
- ④ If hitting `WMARK_MIN`, do direct reclaim.
- ⑤ Do **TLB shutdown** during reclaim + **LUF**.
- ⑥ Return pages on successful reclaim.
- ~~⑦ LUF does TLB shutdowns if needed on return.~~

Approaches

Considerations

Measurement

# When to TLB Shootdowns Vanilla

Approaches

Considerations

Measurement



● TLB shutdown

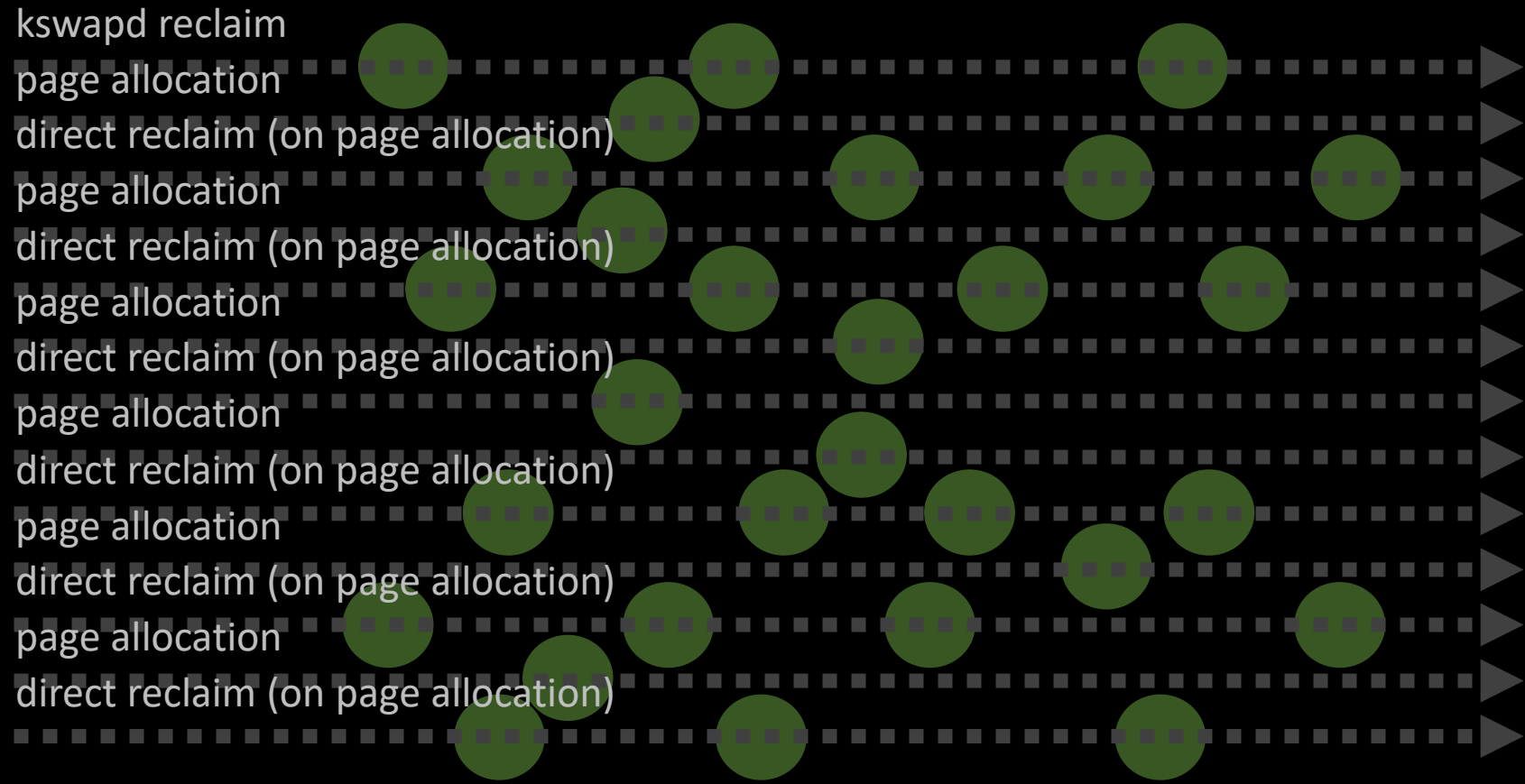
# When to TLB Shootdowns

## Vanilla + LUF

Approaches

Considerations

Measurement



● TLB shutdown

# When to TLB Shootdowns

## Vanilla + LUF

Approaches

Considerations

Measurement

kswapd reclaim

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

● TLB shutdown

- 1 Track CPUs to TLB shutdown per page
- 2 Track CPUs to TLB shutdown per mm\_struct
- 3 Track CPUs to TLB shutdown per address\_space

Approaches

Considerations

Measurement

- 4 When to TLB shutdown per page
- 5 When to TLB shutdown per mm\_struct
- 6 When to TLB shutdown per address\_space

Approaches

Considerations

Measurement

# When manipulating mappings



# Changing Mappings Driven by Users

```
int madvise_cold_or_pageout_pte_range(...)
```

```
or int madvise_free_pte_range(...)
```

```
or unsigned long zap_pte_range(...)
```

```
or long change_pte_range(...)
```

```
or int move_ptes(...)
```

```
{
```

```
    flush_tlb_batched_pending(mm);
```

```
    ...
```

```
}
```

Approaches

Considerations

Measurement

Approaches

Considerations

Measurement

# When creating RW mappings

# Creating Write Mappings

Approaches

Considerations

Measurement

```
vm_fault_t handle_mm_fault(...)  
{  
    ...  
    if (vma->vm_flags & VM_WRITE)  
        luf_flush_mm(mm);  
}
```

Approaches

Considerations

Measurement

# When updating page cache

# Updating Page Cache via Buffered IO

Approaches

Considerations

Measurement

```
ret = mapping->a_ops->write_begin(...);  
If (!ret) {  
    luf_flush_mapping(mapping);  
    ...  
    mapping->a_ops->write_end(...);  
}
```

Approaches

Considerations

Measurement

# When directly updating files

# Updating Files via Direct IO

```
void truncate_inode_pages_range(...)
or int invalidate_inode_pages2_range(...)
or int truncate_inode_folio(...)
{
    ...
    luf_flush_mapping(mapping);
}
```

Approaches

Considerations

Measurement

Approaches

Considerations

Measurement

# What if I miss things?



# Design LUF Debug Feature

Approaches

Use additional space in struct page.

Considerations

Strongly synchronize on objects for checking.

Measurement

Check the sanity on every `kmap()` and its family.

Check the sanity on every `{pte,pmd,pud}_mkwrite()`.

1 TLB shutdown number

2 TLB miss number

3 Runtime of the workload

Approaches

Considerations

Measurement

# TLB Shutdown Number

Vanilla

While the workload of llama.cpp runs for 4000 secs...

```
$ grep TLB /proc/interrupts
```

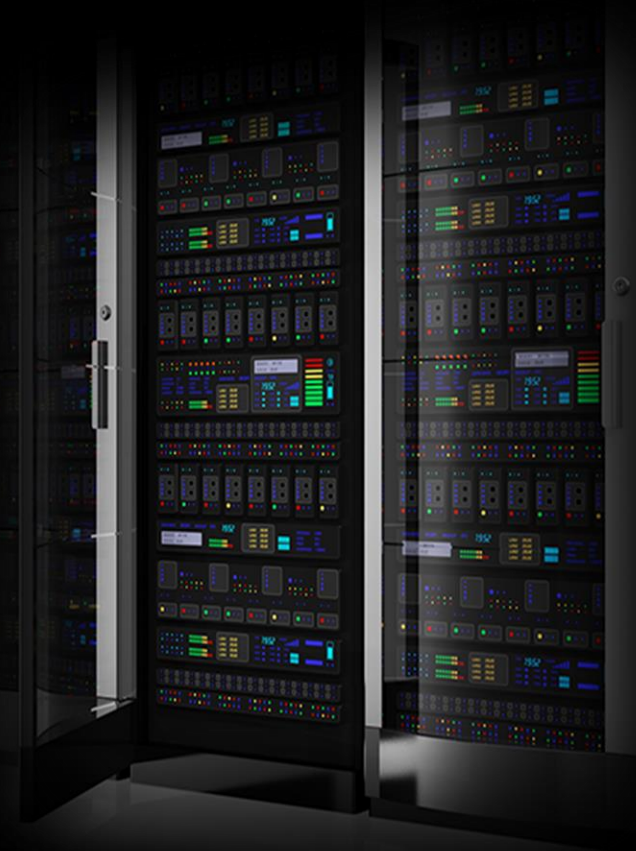
TLB:

```
80911532 93691786 100296251 111062810 109769109 109862429
108968588 119175230 115779676 118377498 119325266 120300143
124514185 116697222 121068466 118031913 122660681 117494403
121819907 116960596 120936335 117217061 118630217 122322724
119595577 111693298 119232201 120030377 115334687 113179982
118808254 116353592 140987367 137095516 131724276 139742240
136501150 130428761 127585535 132483981 133430250 133756207
131786710 126365824 129812539 133850040 131742690 125142213
128572830 132234350 131945922 128417707 133355434 129972846
126331823 134050849 133991626 121129038 124637283 132830916
126875507 122322440 125776487 124340278 TLB shutdowns
```

Approaches

Considerations

Measurement



# TLB Shutdown Number

# Vanilla + LUF

While the workload of llama.cpp runs for 4000 secs...

```
$ grep TLB /proc/interrupts
```

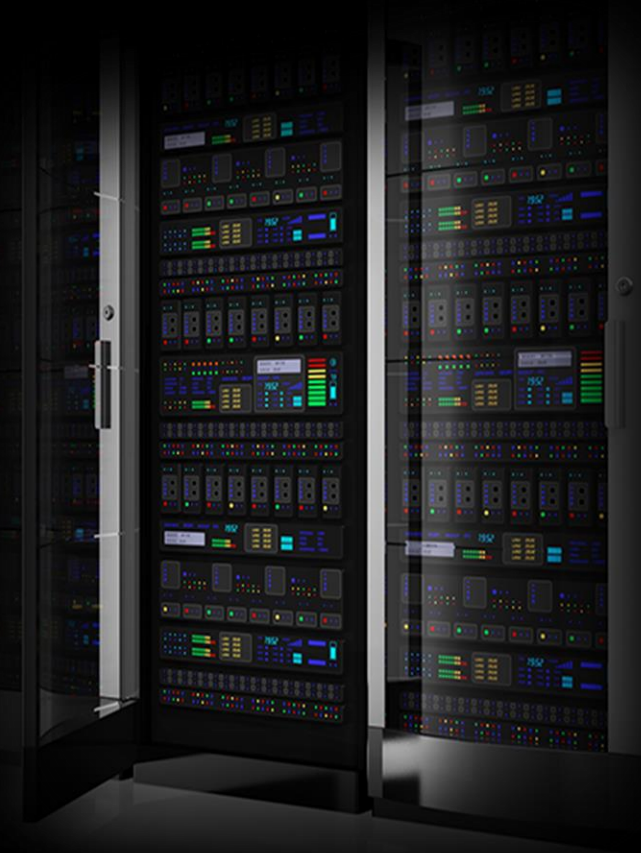
TLB:

2121206	2615108	2983494	2911950	3055086	3092672
3204894	3346082	3286744	3307310	3357296	3315940
3428034	3112596	3143325	3185551	3186493	3322314
3330523	3339663	3156064	3272070	3296309	3198962
3332662	3315870	3234467	3353240	3281234	3300666
3345452	3173097	4009196	3932215	3898735	3726531
3717982	3671726	3728788	3724613	3799147	3691764
3620630	3684655	3666688	3393974	3448651	3487593
3446357	3618418	3671920	3712949	3575264	3715385
3641513	3630897	3691047	3630690	3504933	3662647
3629926	3443044	3832970	3548813	TLB shutdowns	

Approaches

Considerations

Measurement



# TLB Shutdown Number

# Vanilla + LUF

While the workload of llama.cpp runs for 4000 secs...

```
$ grep TLB /proc/interrupts
```

TLB:

2121206	2615108	2983494	3111150	3655081	3091372
3204894	3346082	3286744	3071100	3729011	3115940
3428034	3112596	3143325	3187151	3111493	3311131
3330523	3339663	3156064	3272070	3296309	3198962
3332662	3315870	3234467	3353240	3281234	3300666
3345452	3173097	4009196	3932215	3898735	3726531
3717982	3671726	3728788	3724613	3799147	3691764
3620630	3684655	3666688	3393974	3448651	3487593
3446357	3618418	3671920	3712949	3575264	3715385
3641513	3630897	3691047	3630690	3504933	3662647
3629926	3443044	3832970	3548813	TLB shutdowns	

**-95%**

Approaches

Considerations

Measurement

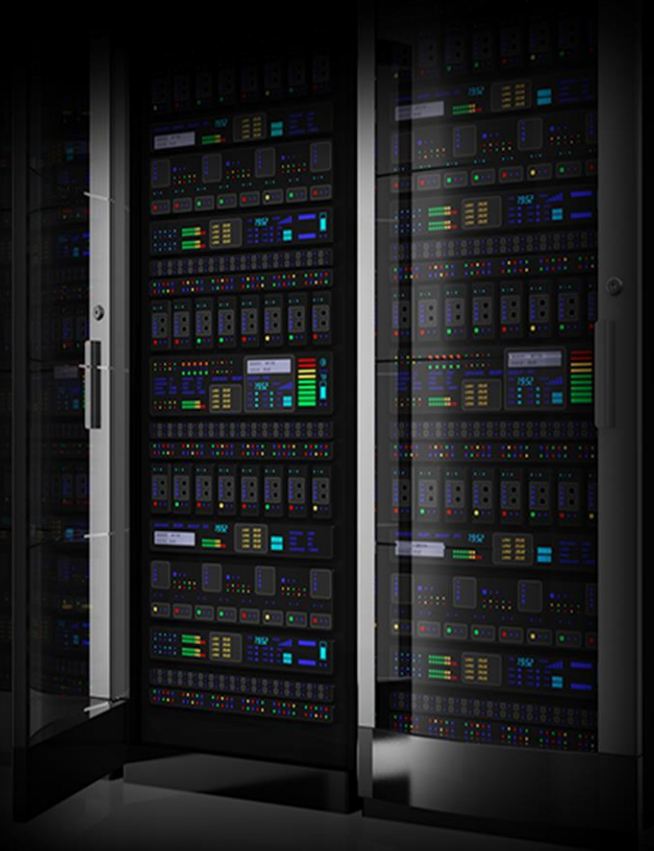


# TLB Miss Number Vanilla

While the workload of llama.cpp runs for 4000 secs...

```
$ perf stat -a \  
> -e dTLB-load-misses \  
> -e dTLB-store-misses \  
> -e iTLB-load-misses \  
> $(llama.cpp.test.sh)
```

25296288810	dTLB-load-misses
21592110362	dTLB-store-misses
3985725200	iTLB-load-misses



Approaches

Considerations

Measurement

# TLB Miss Number Vanilla + LUF

While the workload of llama.cpp runs for 4000 secs...

```
$ perf stat -a \  
> -e dTLB-load-misses \  
> -e dTLB-store-misses \  
> -e iTLB-load-misses \  
> $(llama.cpp.test.sh)
```

12892995056	dTLB-load-misses
5357137957	dTLB-store-misses
669019529	iTLB-load-misses



Approaches

Considerations

Measurement

# TLB Miss Number Vanilla + LUF

While the workload of llama.cpp runs for 4000 secs...

```
$ perf stat -a \  
> -e dTLB-load-misses \  
> -e dTLB-store-misses \  
> -e iTLB-load-misses \  
> $(llama.cpp.test.sh)
```

**-50%**

**-70%**

**-80%**

```
12892995056 dTLB-load-misses  
5357137957 dTLB-store-misses  
669019529 iTLB-load-misses
```

Approaches

Considerations

Measurement

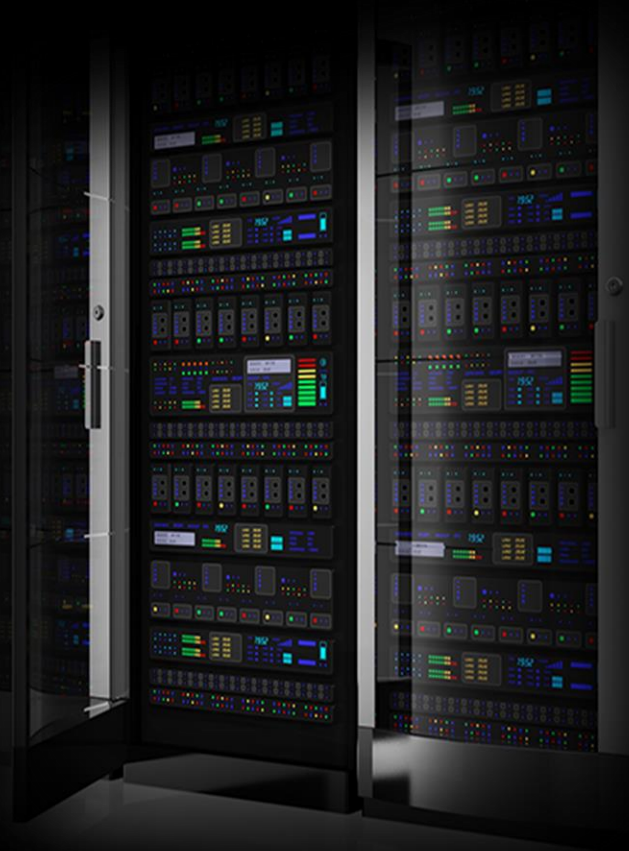




# Runtime of the Workload

## Vanilla

llama_print_timings:	total time = 883450.54 ms / 24 tokens
llama_print_timings:	total time = 861665.91 ms / 24 tokens
llama_print_timings:	total time = 898079.02 ms / 24 tokens
llama_print_timings:	total time = 879897.69 ms / 24 tokens
llama_print_timings:	total time = 892360.75 ms / 24 tokens
llama_print_timings:	total time = 884587.85 ms / 24 tokens
llama_print_timings:	total time = 861023.19 ms / 24 tokens
llama_print_timings:	total time = 900022.18 ms / 24 tokens
llama_print_timings:	total time = 878771.88 ms / 24 tokens
llama_print_timings:	total time = 889027.98 ms / 24 tokens
llama_print_timings:	total time = 880783.90 ms / 24 tokens
llama_print_timings:	total time = 856475.29 ms / 24 tokens
llama_print_timings:	total time = 896842.21 ms / 24 tokens
llama_print_timings:	total time = 878883.53 ms / 24 tokens
llama_print_timings:	total time = 890122.10 ms / 24 tokens



Approaches

Considerations

Measurement

# Runtime of the Workload

Vanilla + LUF

llama\_print\_timings: total time = 871060.86 ms / 24 tokens  
llama\_print\_timings: total time = 825609.53 ms / 24 tokens  
llama\_print\_timings: total time = 836854.81 ms / 24 tokens  
llama\_print\_timings: total time = 843147.99 ms / 24 tokens  
llama\_print\_timings: total time = 831426.65 ms / 24 tokens  
llama\_print\_timings: total time = 873939.23 ms / 24 tokens  
llama\_print\_timings: total time = 826127.69 ms / 24 tokens  
llama\_print\_timings: total time = 835489.26 ms / 24 tokens  
llama\_print\_timings: total time = 842589.62 ms / 24 tokens  
llama\_print\_timings: total time = 833700.66 ms / 24 tokens  
llama\_print\_timings: total time = 875996.19 ms / 24 tokens  
llama\_print\_timings: total time = 826401.73 ms / 24 tokens  
llama\_print\_timings: total time = 839341.28 ms / 24 tokens  
llama\_print\_timings: total time = 841075.10 ms / 24 tokens  
llama\_print\_timings: total time = 835136.41 ms / 24 tokens

Approaches

Considerations

Measurement



# Runtime of the Workload

# Vanilla + LUF

llama\_print\_timings: total time = 871060.86 ms / 24 tokens  
llama\_print\_timings: total time = 825609.53 ms / 24 tokens  
llama\_print\_timings: total time = 836854.81 ms / 24 tokens  
llama\_print\_timings: total time = 843147.99 ms / 24 tokens  
llama\_print\_timings: total time = 834116.0 ms / 24 tokens  
llama\_print\_timings: total time = 827319.25 ms / 24 tokens  
llama\_print\_timings: total time = 826117.69 ms / 24 tokens  
llama\_print\_timings: total time = 835489.26 ms / 24 tokens  
llama\_print\_timings: total time = 842589.62 ms / 24 tokens  
llama\_print\_timings: total time = 833700.66 ms / 24 tokens  
llama\_print\_timings: total time = 875996.19 ms / 24 tokens  
llama\_print\_timings: total time = 826401.73 ms / 24 tokens  
llama\_print\_timings: total time = 839341.28 ms / 24 tokens  
llama\_print\_timings: total time = 841075.10 ms / 24 tokens  
llama\_print\_timings: total time = 835136.41 ms / 24 tokens

**-4%**



Approaches

Considerations

Measurement

**But...**

# When to TLB Shootdowns Vanilla

Approaches

Considerations

Measurement



● TLB shutdown

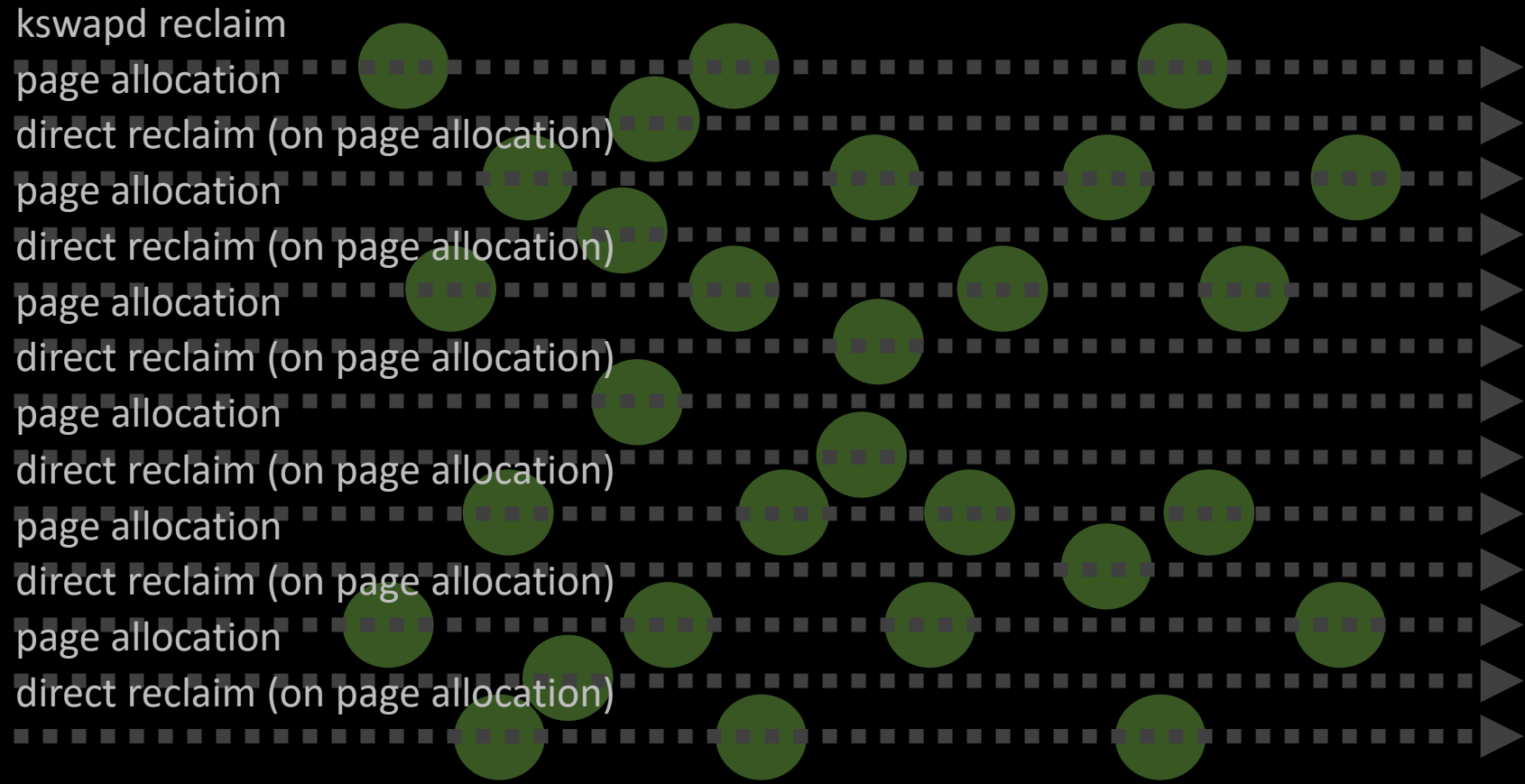
# When to TLB Shootdowns

## Vanilla + LUF

Approaches

Considerations

Measurement



● TLB shutdown

# When to TLB Shootdowns

## Vanilla + LUF

Approaches

Considerations

Measurement

kswapd reclaim

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

page allocation

direct reclaim (on page allocation)

● TLB shutdown

Still thinking how to synchronize...

Seen a better runtime

Seen a even worse runtime



**Question?**