Funded by the European Union

Elastic Translations: Fast virtual memory with multiple translation sizes

Stratos Psomadakis; Chloe Alverti‡; Vasileios Karakostas†; Christos Katsakioris; Dimitrios Siakavaras; Konstantinos Nikas; Georgios Goumas; Nectarios Koziris

National Technical University of Athens

‡ University of Illinois Urbana-Champaign

† University of Athens

In a nutshell

***Address Translation Wall**:*
TLBs under pressure as memory footprints swell

Large pages, ***2MiB*** and ***1GiB,*** extend TLB reach
2MiB → *overfit / underfit,* 1GiB → *too large*

ARMv8 supports ***coalesced 64KiB*** and ***32MiB*** translations
OS support is limited

Our Proposal: ***Elastic Translations***

→ ***Transparent*** OS support for coalesced translations

→ ***CoalaPaging*** for practical contiguity

→ ***Leshy*** for informed size selection
via lightweight HW-assisted sampling

Outline

→ OS-assisted TLB coalescing

→ Elastic Translations

i. CoalaPaging for practical contiguity

ii. Transparent Contig Bit Management

iii. Asynchronous Promotions

iv. Leshy for translation size selection

→ Evaluation

→ Conclusion

Outline

→ OS-assisted TLB coalescing

→ Elastic Translations

i. CoalaPaging for practical contiguity

ii. Transparent Contig Bit Management

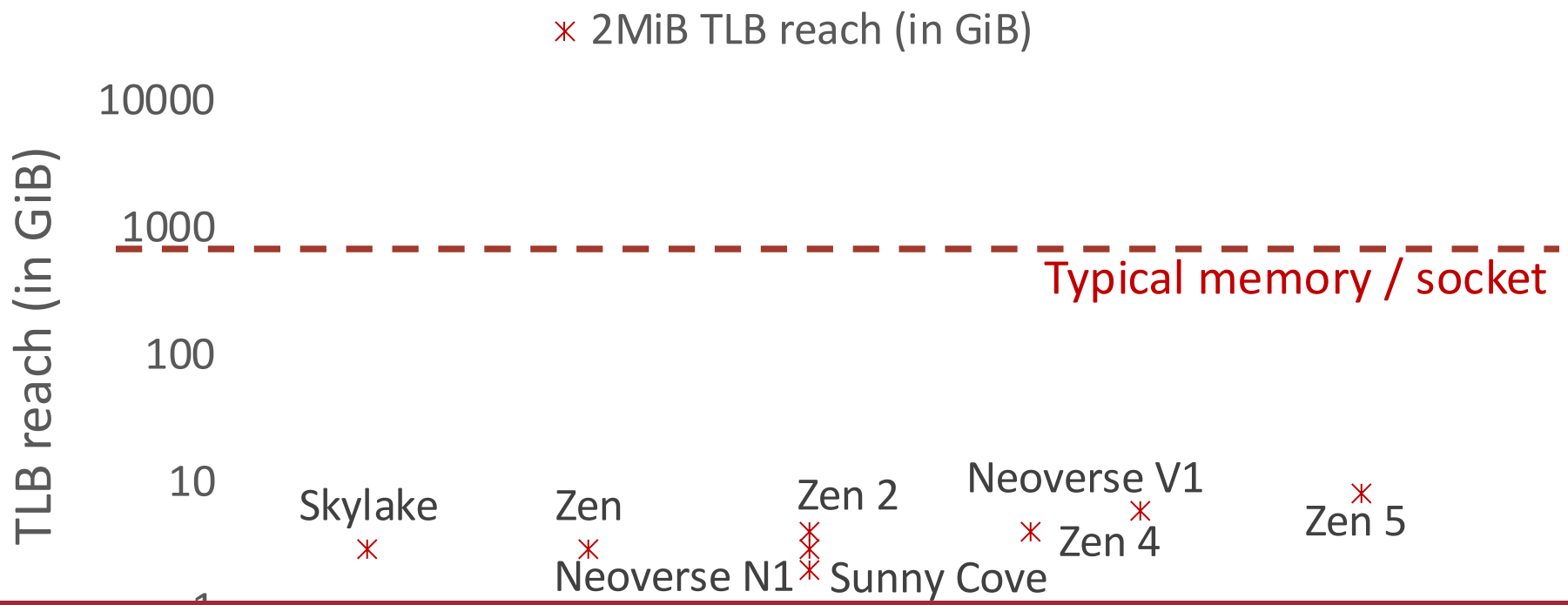
iii. Asynchronous Promotions

iv. Leshy for translation size selection

→ Evaluation

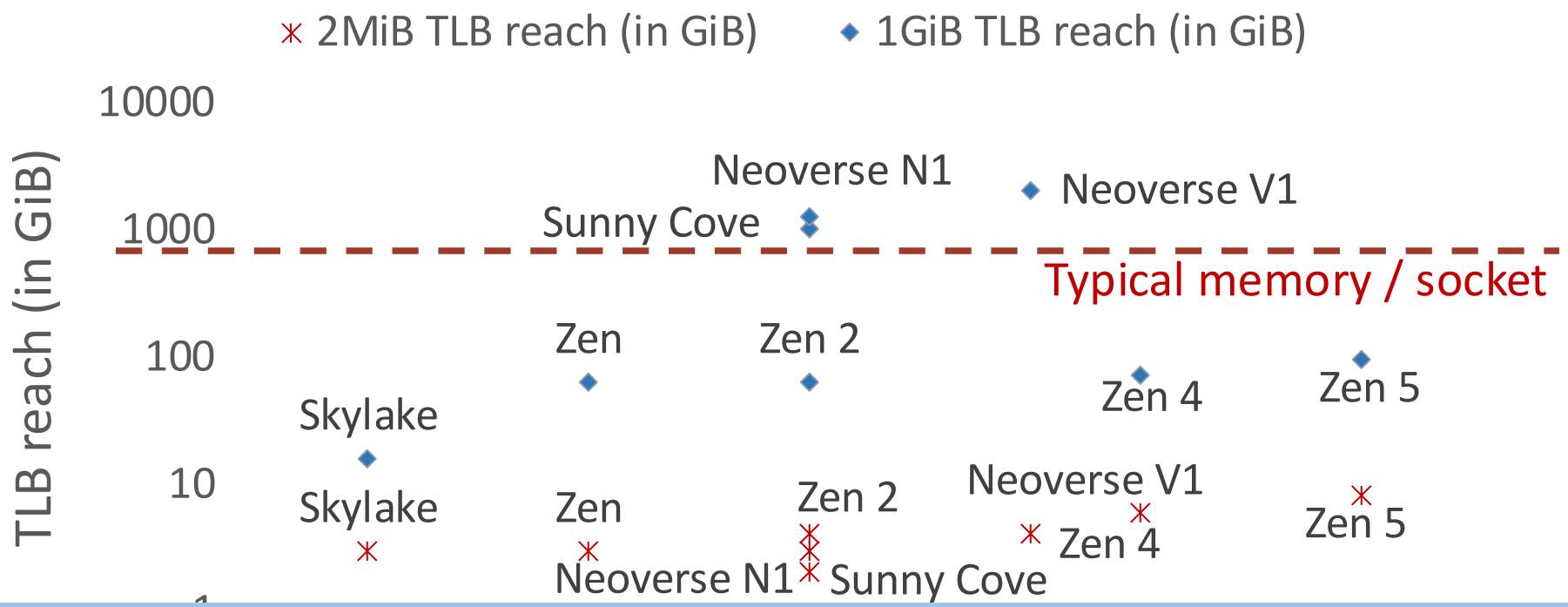
→ Conclusion

Address Translation Hits the Wall



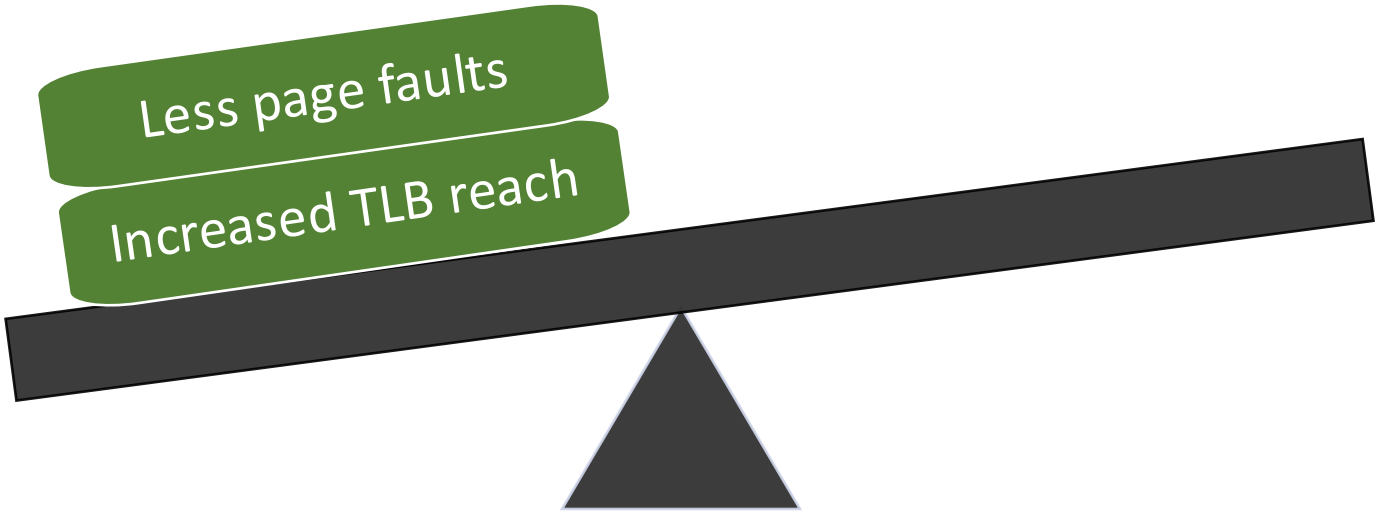
2MiB → Falling Behind

Address Translation Hits the Wall

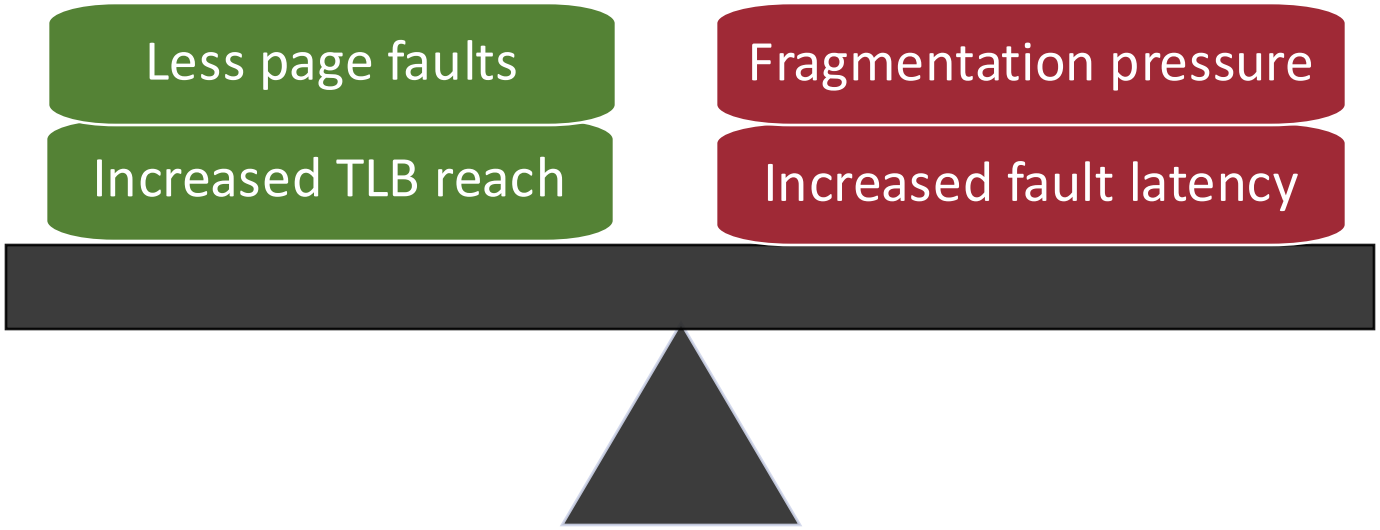


1GiB → Enough potential coverage but….

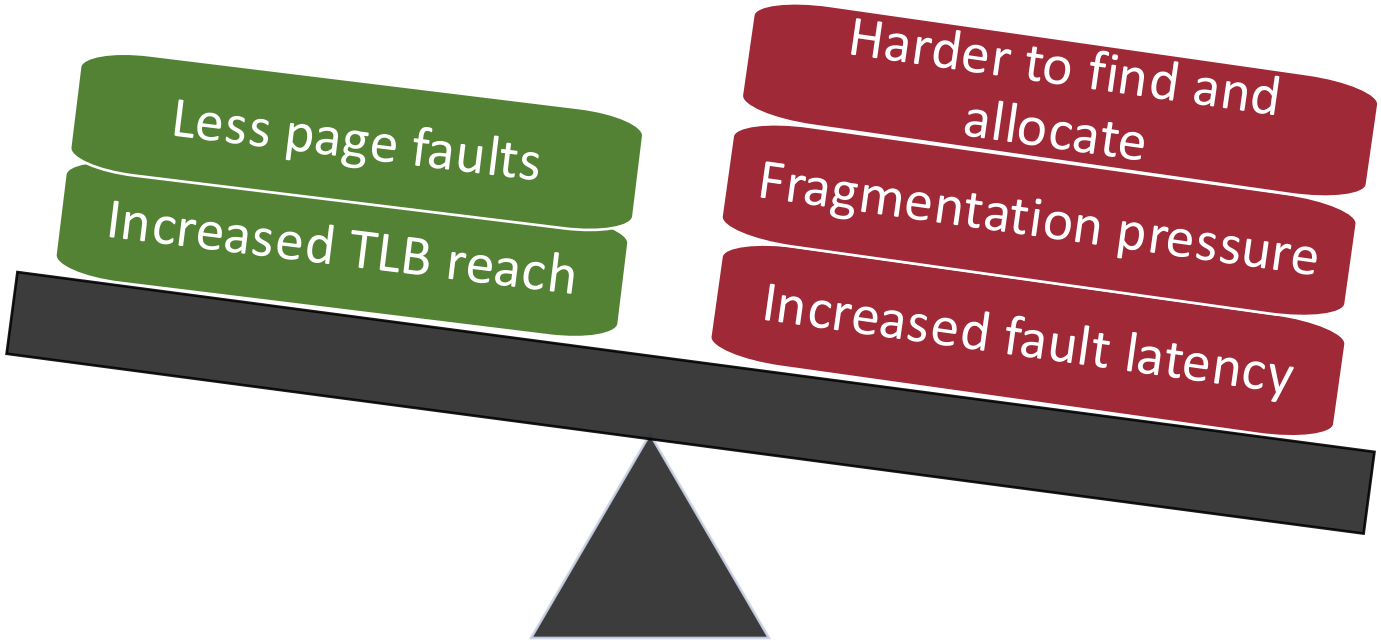
Increasing the Large Page Size



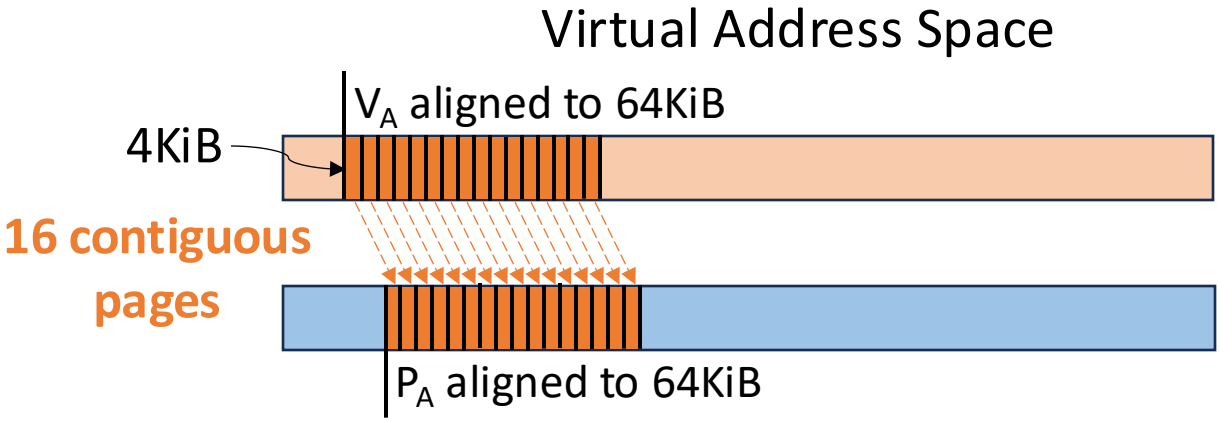
Increasing the Large Page Size

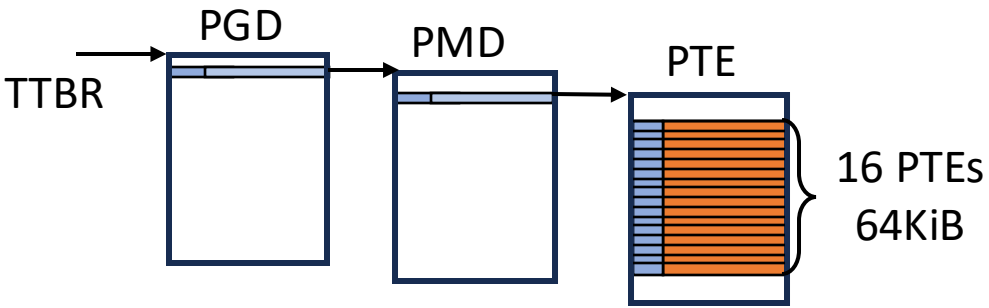


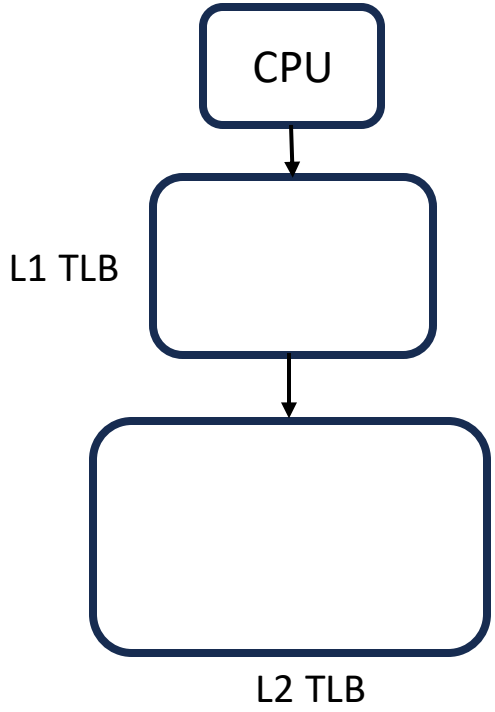
Increasing the Large Page Size



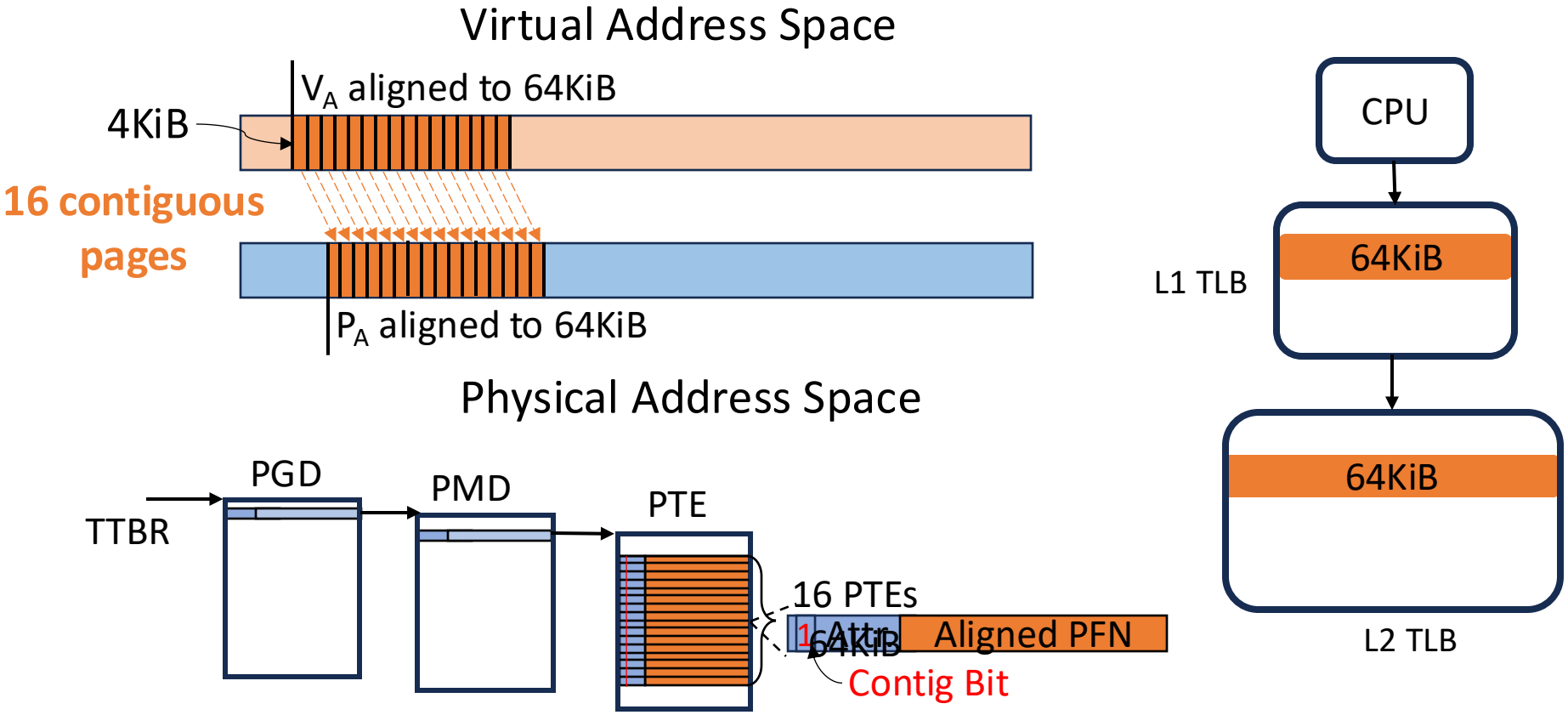
ARMv8 OS-assisted TLB coalescing



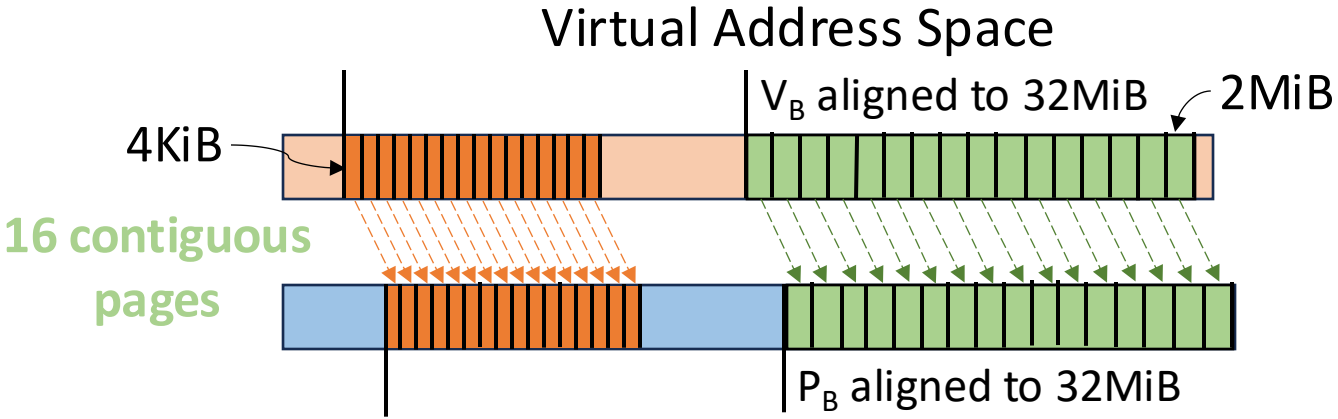


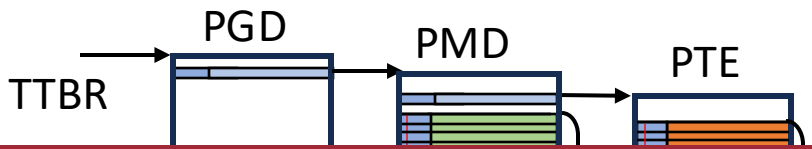


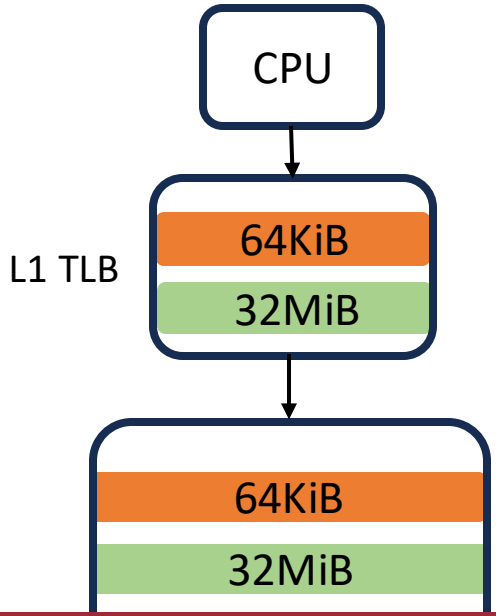
ARMv8 OS-assisted TLB coalescing



ARMv8 OS-assisted TLB coalescing







While commodity CPUs support TLB coalescing, OS support is lagging behind

OS Interfaces for coalesced translations

Interface
Linux HugeTLB
Linux (m)THP

OS Interfaces for coalesced translations

Interface	64KiB	32MiB	Transparent	Allocation Policy	Size Selection	Virtualized Execution	Asynchronous Promotions
Linux HugeTLB	✓	✓	✗	Pre-allocate	User-defined	✗	✗
Linux (m)THP	✓	✗	✓	First-touch	2MiB→ 64KiB→ 4KiB	✗	✗

Limited transparent support

Greedy allocation and promotion policies

OS Interfaces for coalesced translations

Interface	64KiB	32MiB	Transparent	Allocation Policy	Size Selection	Virtualized Execution	Asynchronous Promotions
Linux HugeTLB	✓	✓	✗	Pre-allocate	User-defined	✗	✗
Linux (m)THP	✓	✗	✓	First-touch	2MiB→ 64KiB→ 4KiB	✗	✗
Our Proposal							
Elastic Translations	✓	✓	✓	Opportunistic	Guided	✓	✓

Elastic Translations

Opportunistic

Opportunistically create ***64KiB and 32MiB*** contiguity with ***CoalaPaging*** and ***CoalaKhugepaged***

Transparent

Transparently manage the contig bit at fault time

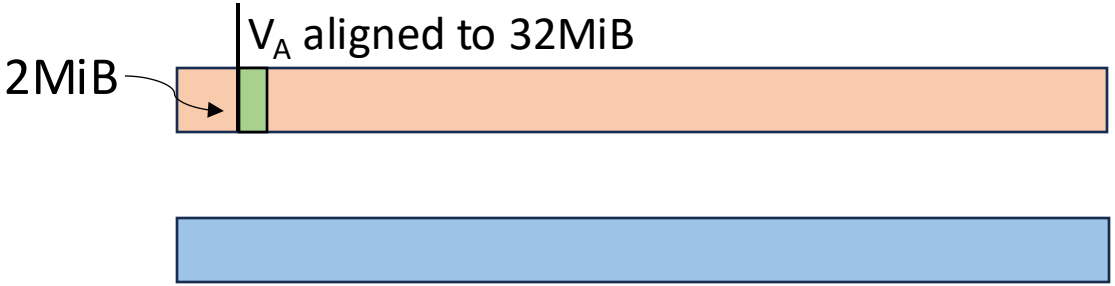
Guided

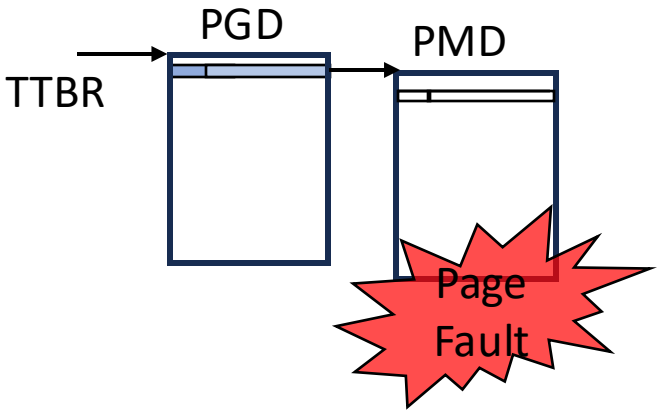
Informed translation size selection with the **Leshy** HW-assisted profiler

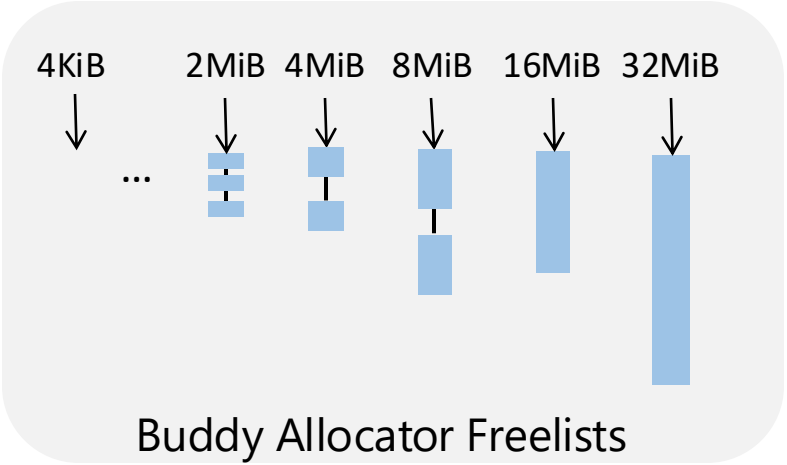
Outline

- → OS-assisted TLB coalescing
- → Elastic Translations
  - i. CoalaPaging for practical contiguity
  - ii. Transparent Contig Bit Management
  - iii. Asynchronous Promotions
  - iv. Leshy for translation size selection
- → Evaluation
- → Conclusion

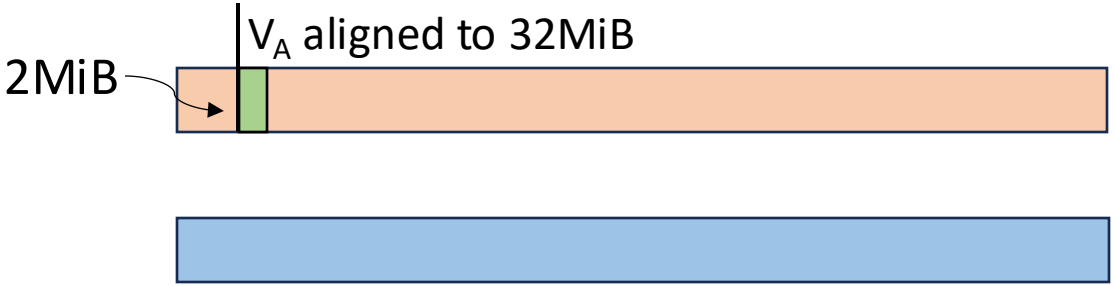
CoalaPaging: Coalescing-aware Paging

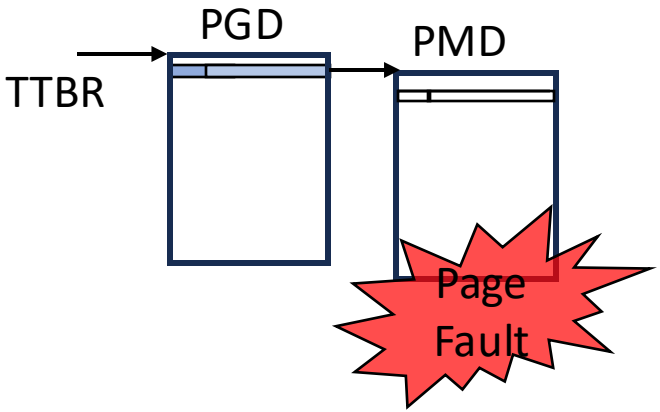


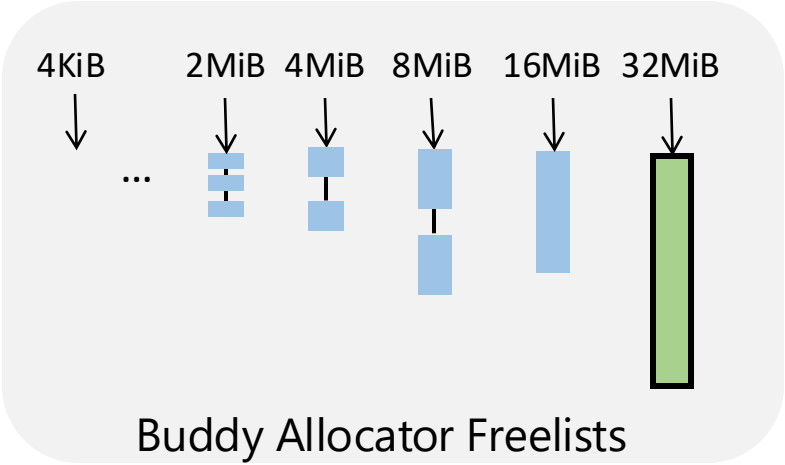




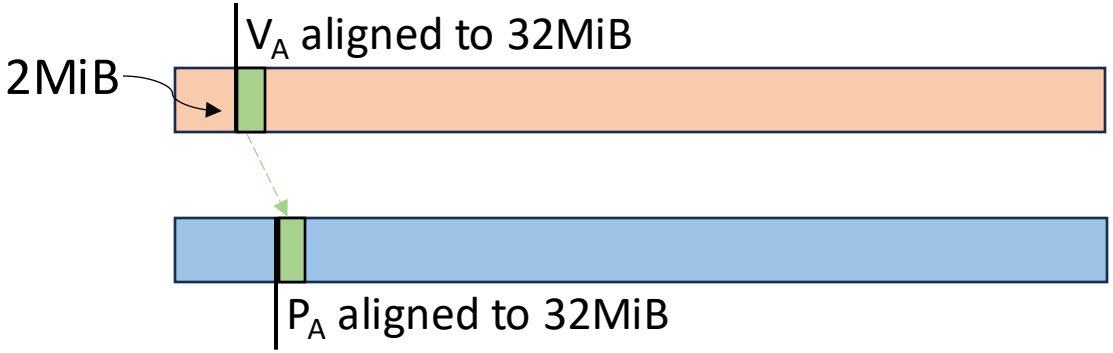
CoalaPaging: Coalescing-aware Paging

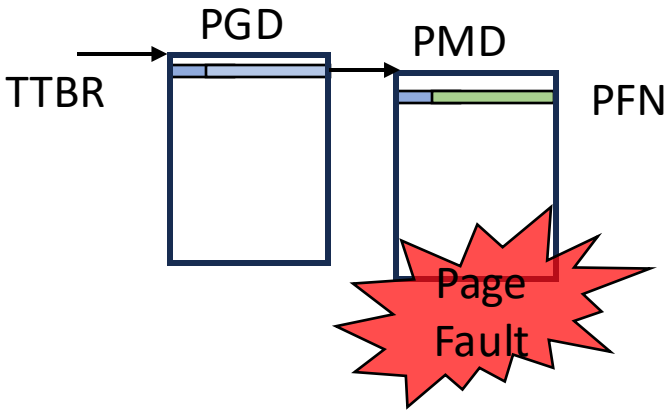


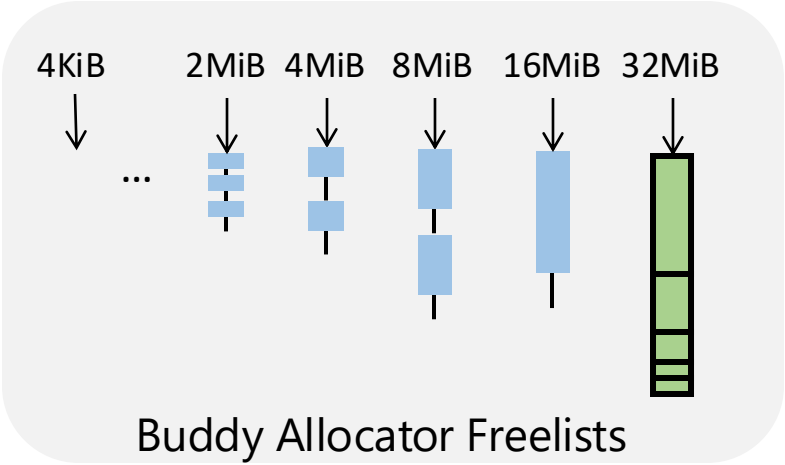




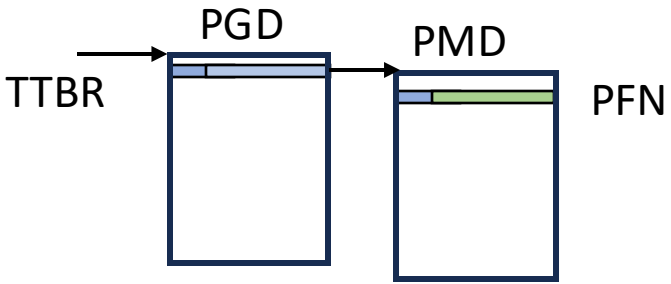
CoalaPaging: Coalescing-aware Paging

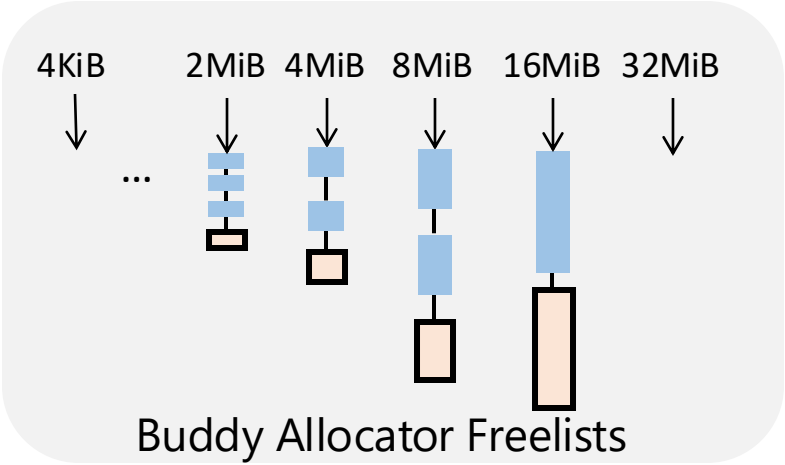




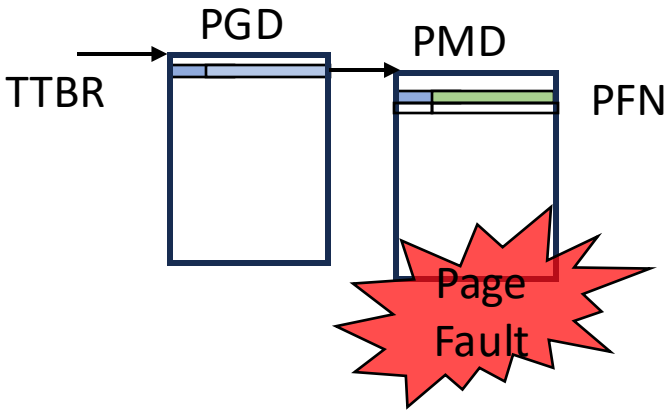


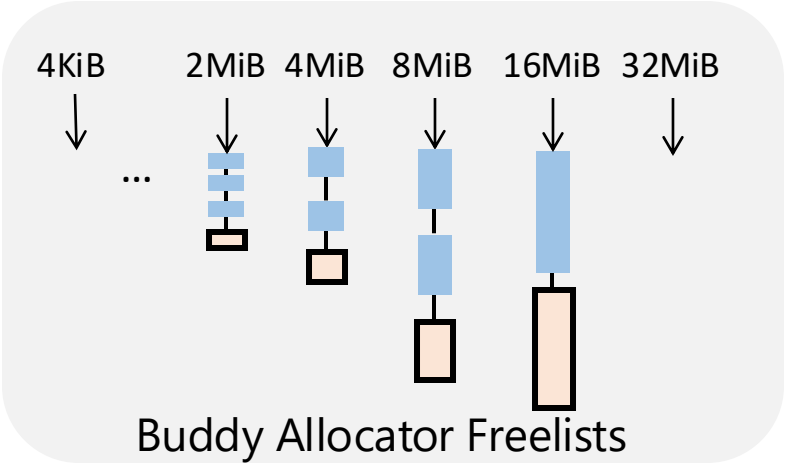
CoalaPaging: Coalescing-aware Paging



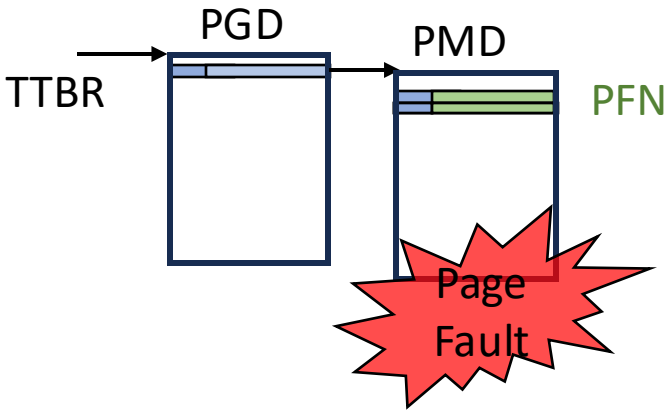


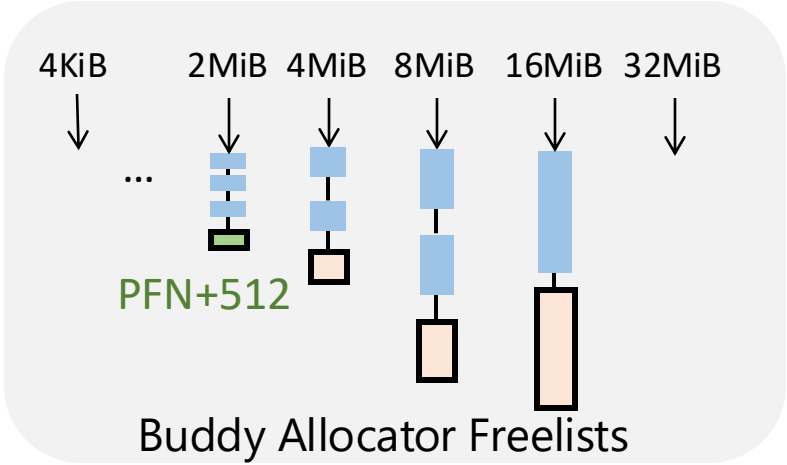
CoalaPaging: Coalescing-aware Paging



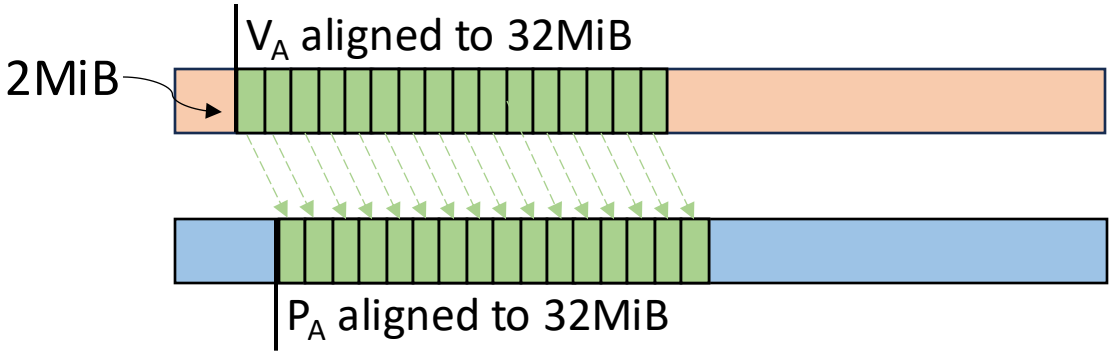


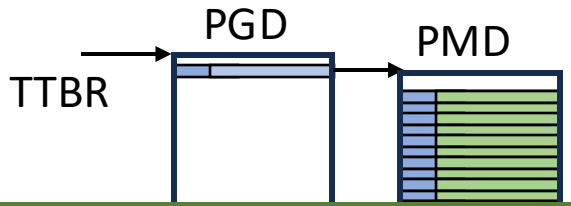
CoalaPaging: Coalescing-aware Paging

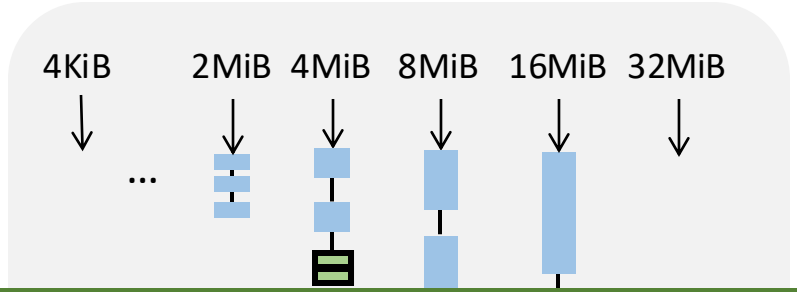




CoalaPaging: Coalescing-aware Paging







Same across 4KiB faults → 64KiB contiguity

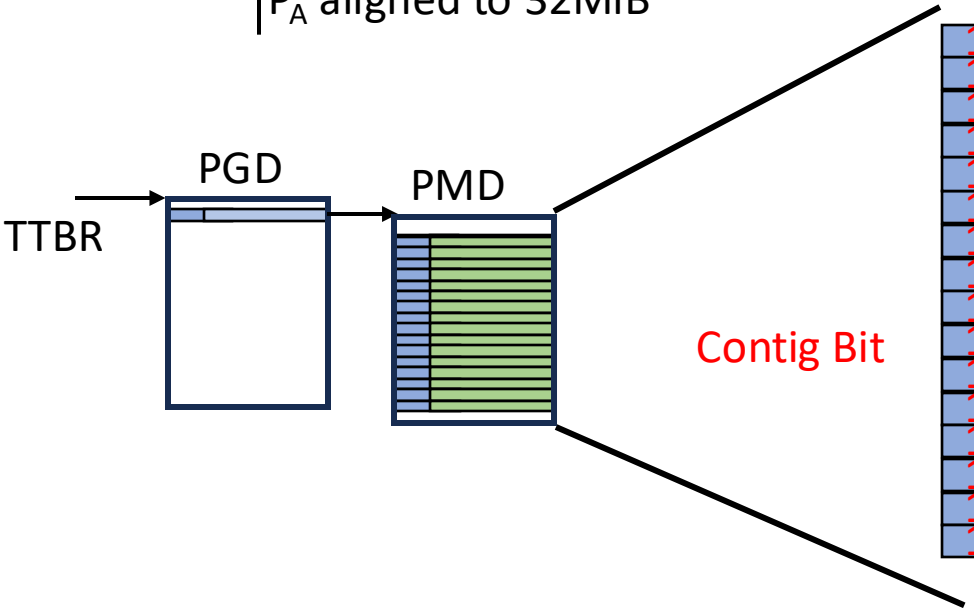
Outline

- → OS-assisted TLB coalescing
- → Elastic Translations
  - i. CoalaPaging for practical contiguity
  - ii. Transparent Contig Bit Management
  - iii. Asynchronous Promotions
  - iv. Leshy for translation size selection
- → Evaluation
- → Conclusion

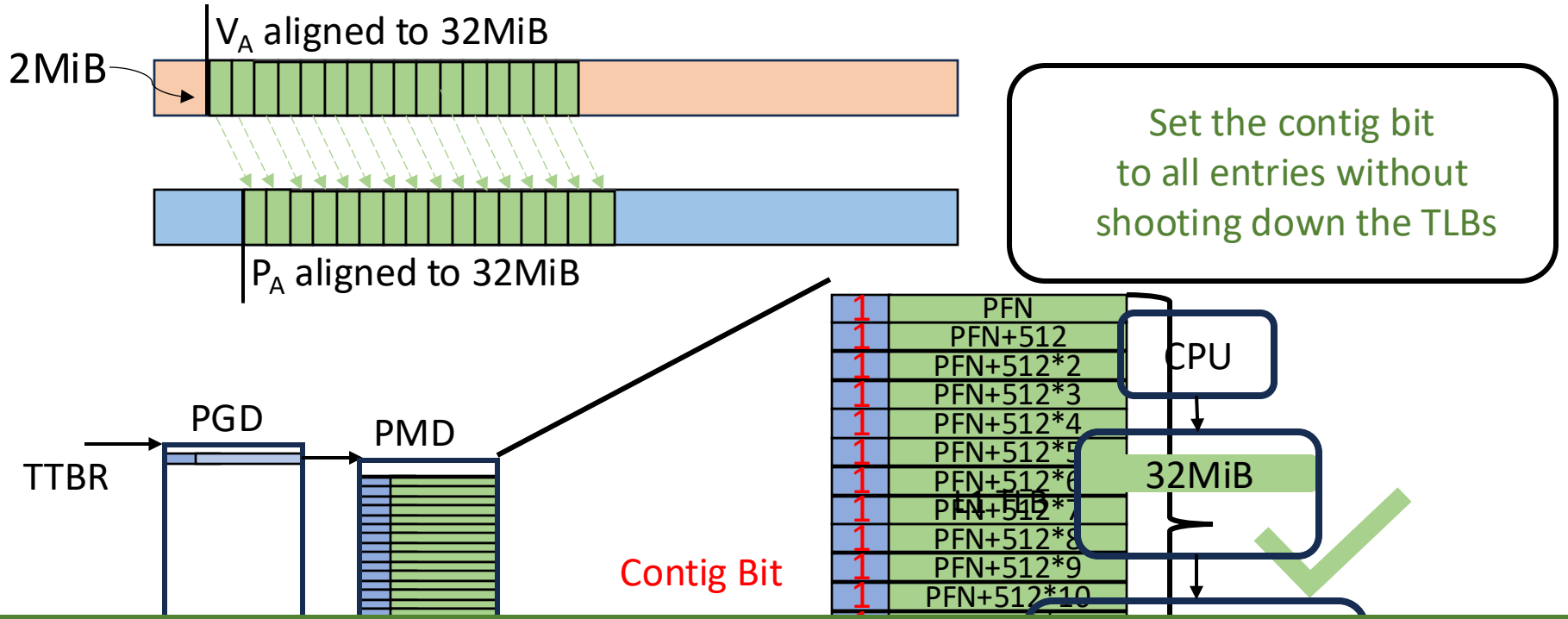
Transparent Contig Bit Management

Check if properly aligned
16 neighbors are present
and contiguous

Set the contig bit
to all bases without
shooting down TLBs



Transparent Contig Bit Management

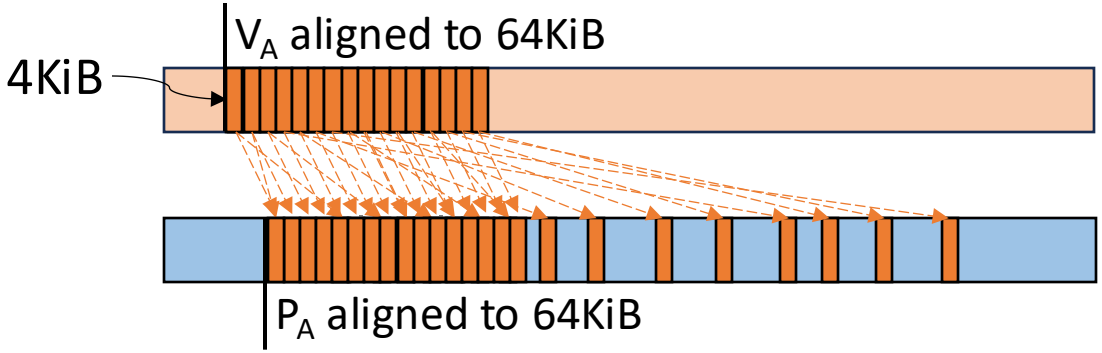


Opportunistic and transparent control of the contig bit during faults enables transparent TLB coalescing

Outline

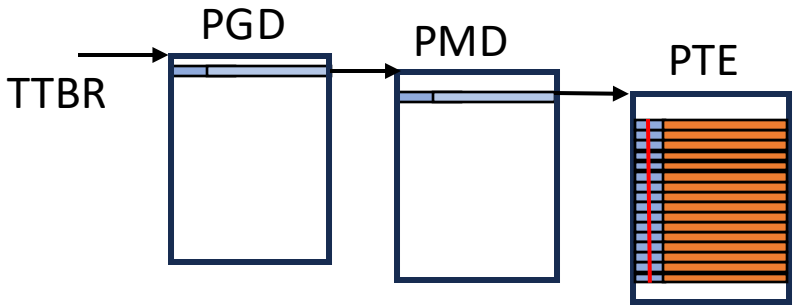
- → OS-assisted TLB coalescing
- → Elastic Translations
  - i. CoalaPaging for practical contiguity
  - ii. Transparent Contig Bit Management
  - iii. Asynchronous Promotions
  - iv. Leshy for translation size selection
- → Evaluation
- → Conclusion

CoalaKhugepaged: Asynchronous Promotions



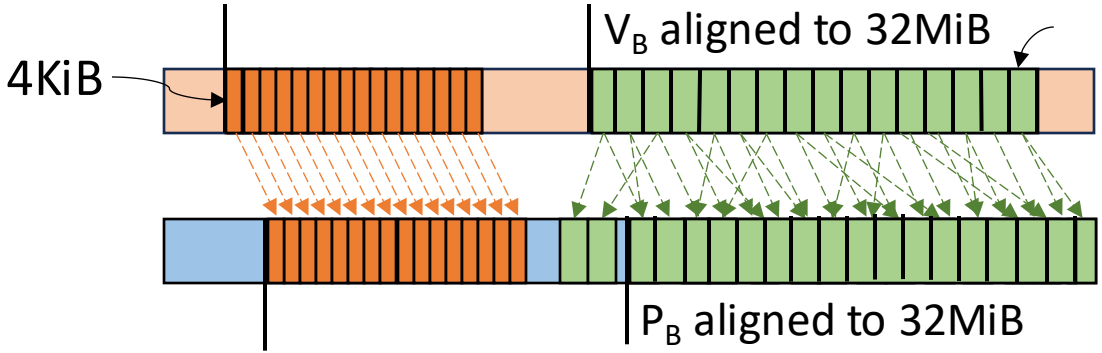
Virtual Address Space

Physical Address Space



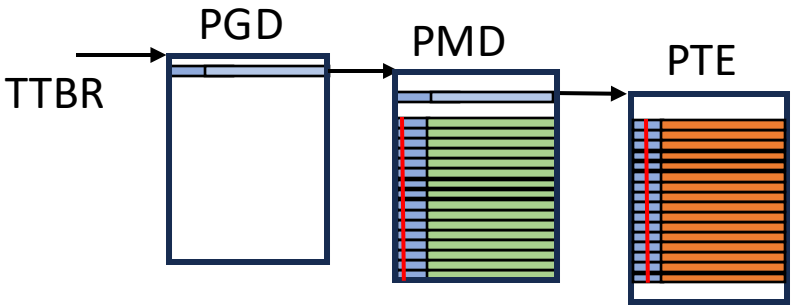
Contig Bit

CoalaKhugepaged: Asynchronous Promotions



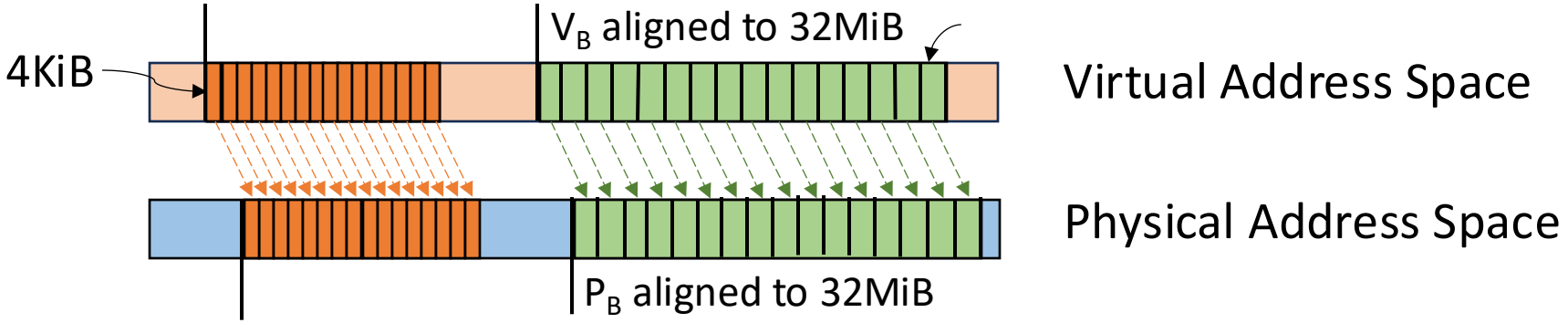
Virtual Address Space

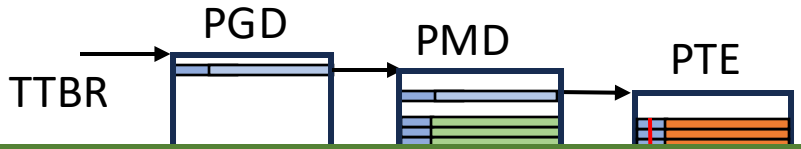
Physical Address Space



Contig Bit

CoalaKhugepaged: Asynchronous Promotions



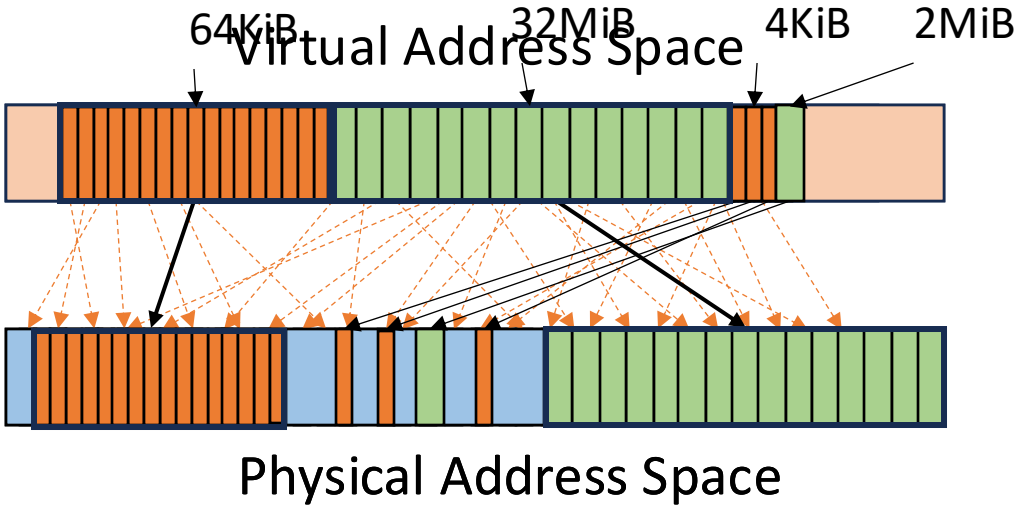


Synergy with CoalaPaging to minimize migrations
(more in the paper…)

Outline

- → OS-assisted TLB coalescing
- → Elastic Translations
  - i. CoalaPaging for practical contiguity
  - ii. Transparent Contig Bit Management
  - iii. Asynchronous Promotions
  - iv. Leshy for translation size selection
- → Evaluation
- → Conclusion

How to pick translation size?

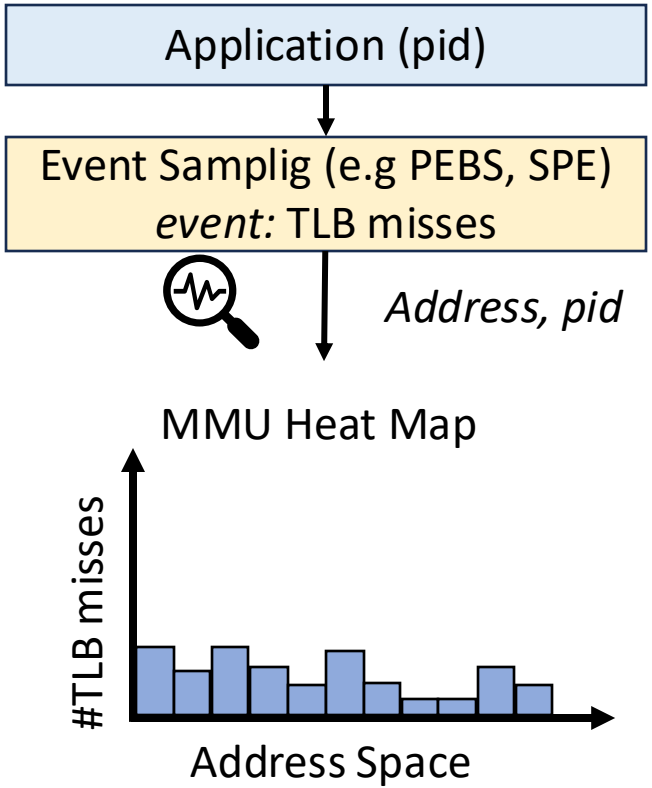


Elastic Translations: Size Selection

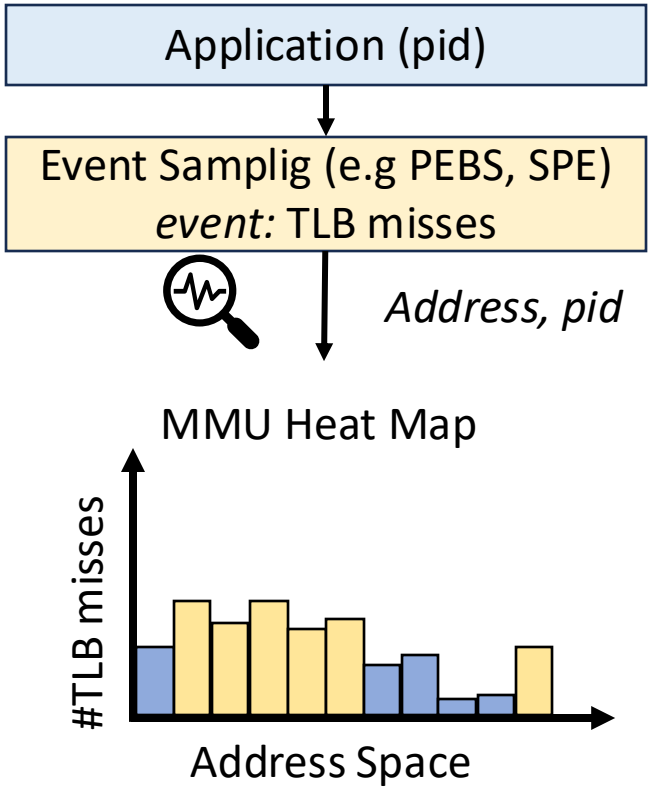
Fault-time allocations → Mapping size threshold

Asynchronous Promotions → Profiling-assisted size selection

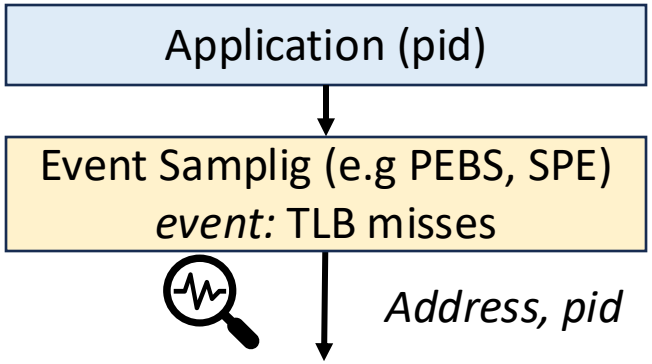
Leshy: Profiling-assisted size selection

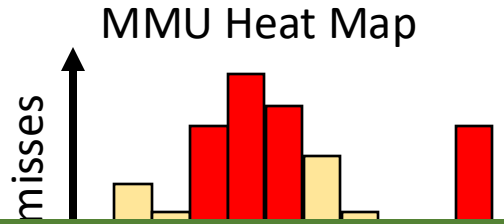


Leshy: Profiling-assisted size selection

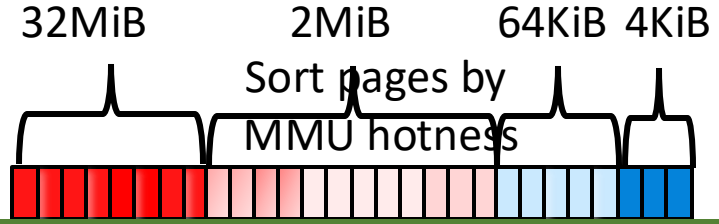


Leshy: Profiling-assisted size selection





Run placement algorithm
to map regions
with translation sizes
to minimize TLB misses



Guided promotions based on real MMU activity

Outline

- → OS-assisted TLB coalescing
- → Elastic Translations
  - i. CoalaPaging for practical contiguity
  - ii. Transparent Contig Bit Manegement
  - iii. Asynchronous Promotions
  - iv. Leshy for translation size selection
- → Evaluation
- → Conclusion

Methodology

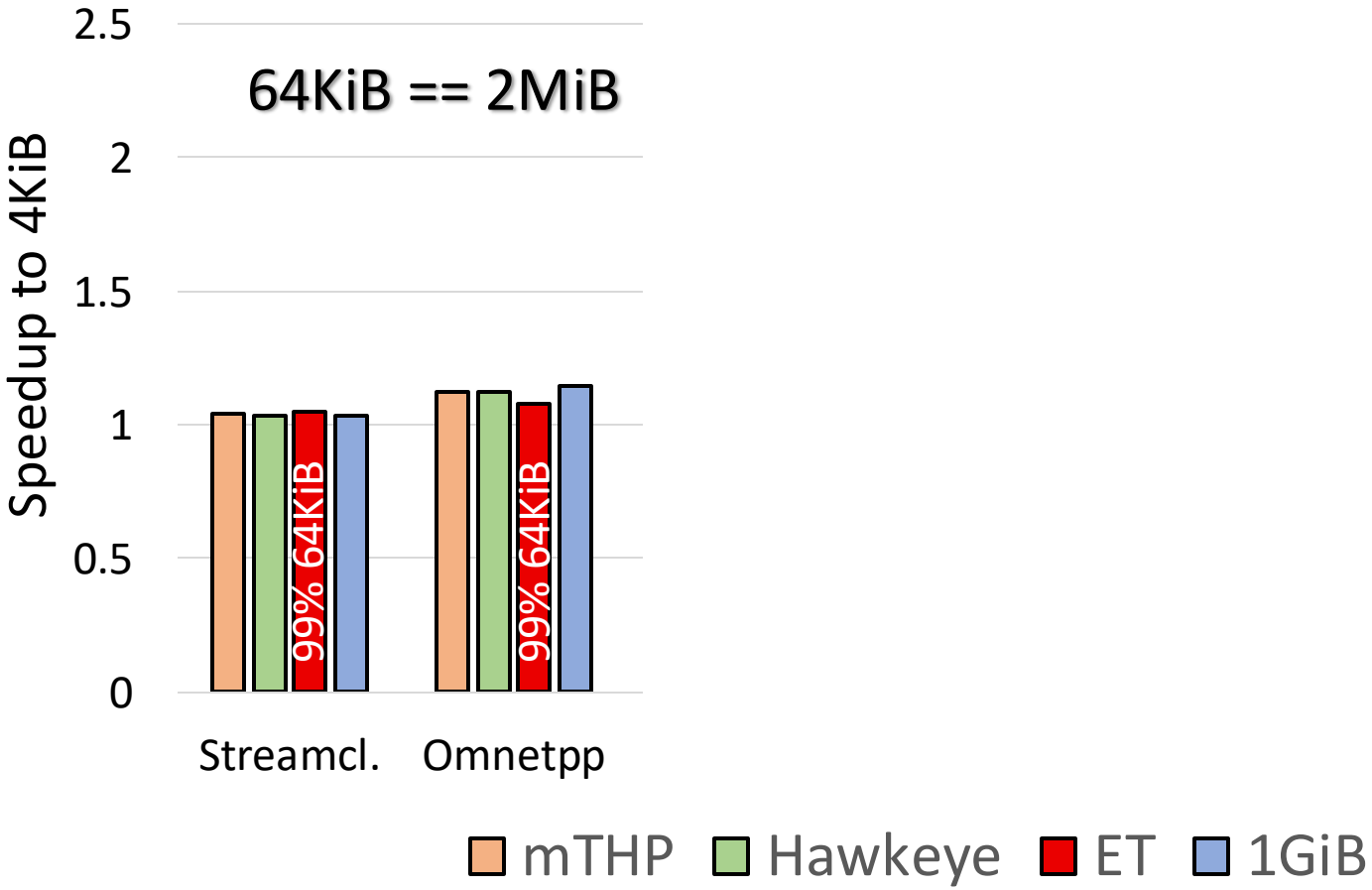
- → Real System Implementation and Evaluation
  - → **ET** on **Linux v5.18**
  - → Armv8 Ampere Altra server (N1), NVIDIA GraceHopper (V2)
- → Comparison with
  - → **mTHP**, SOP for transparent huge pages
  - → **Hawkeye**, SOTA for transparent huge pages
  - → **HugeTLB** for 1GiB
- → Presented Results for

In the paper

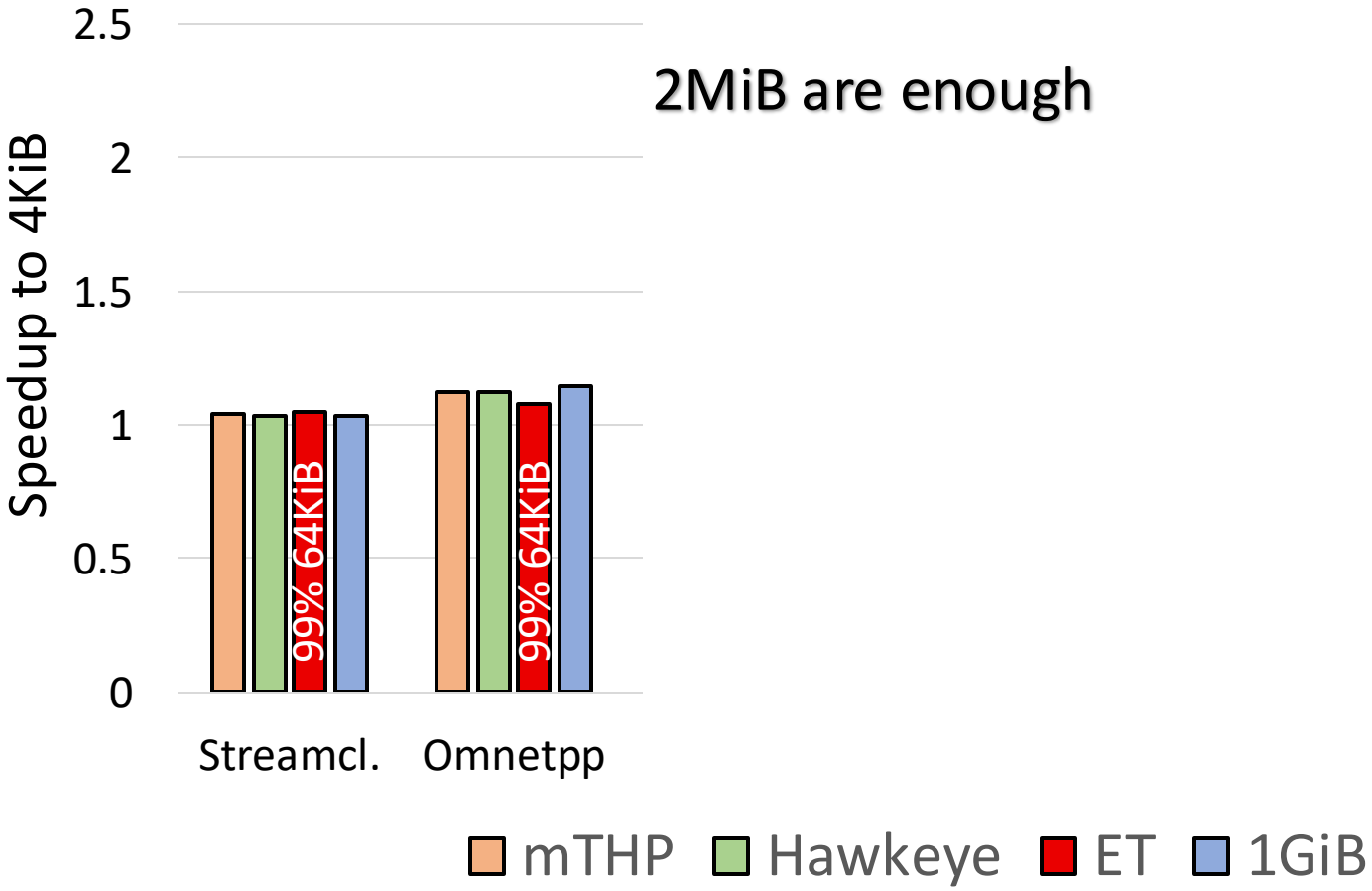
- → Distribution of translation sizes

Code available on Github:
https://github.com/cslab-ntua/elastic-translations-MICRO2024

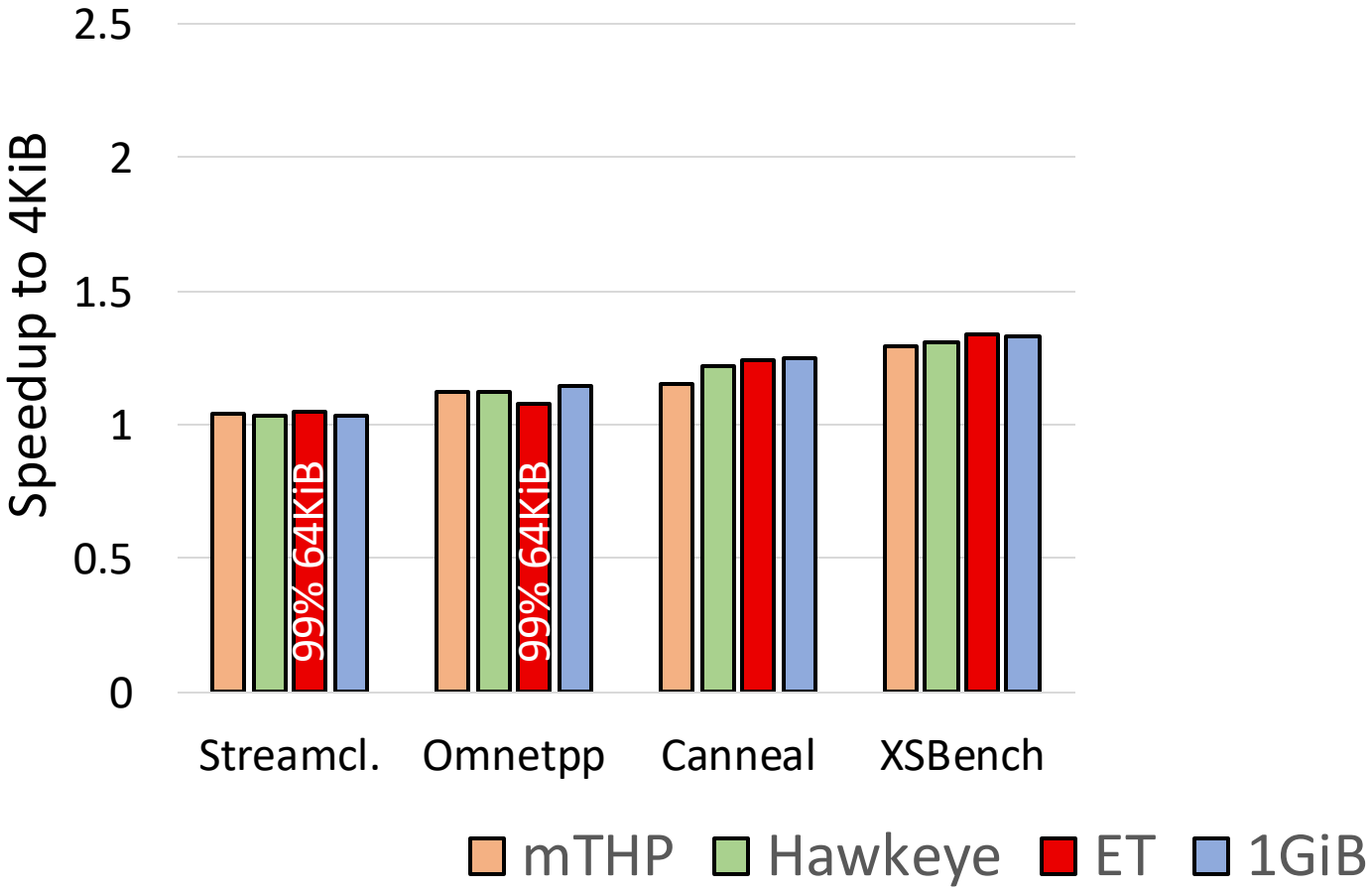
Native, No Fragmentation

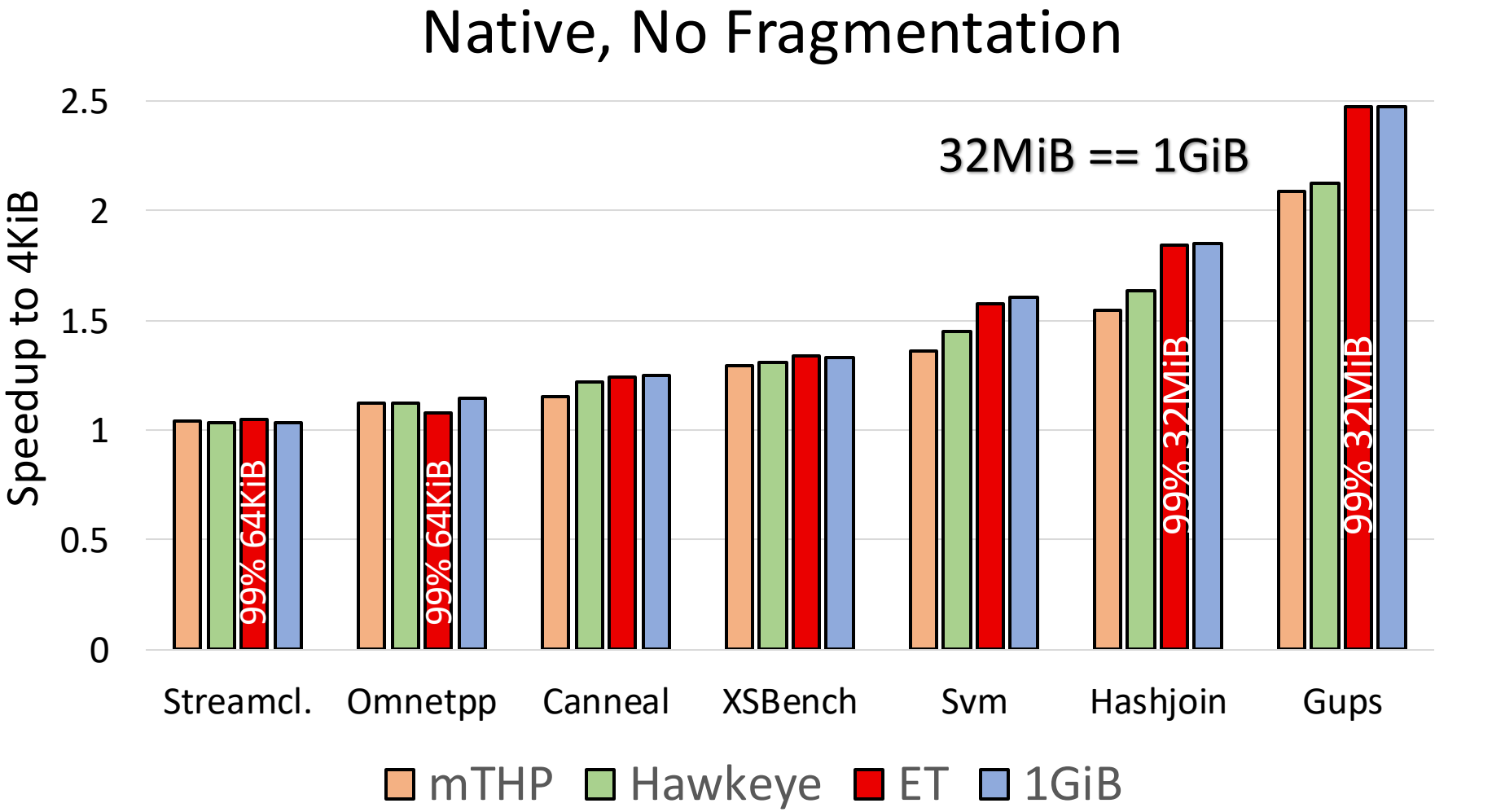


Native, No Fragmentation

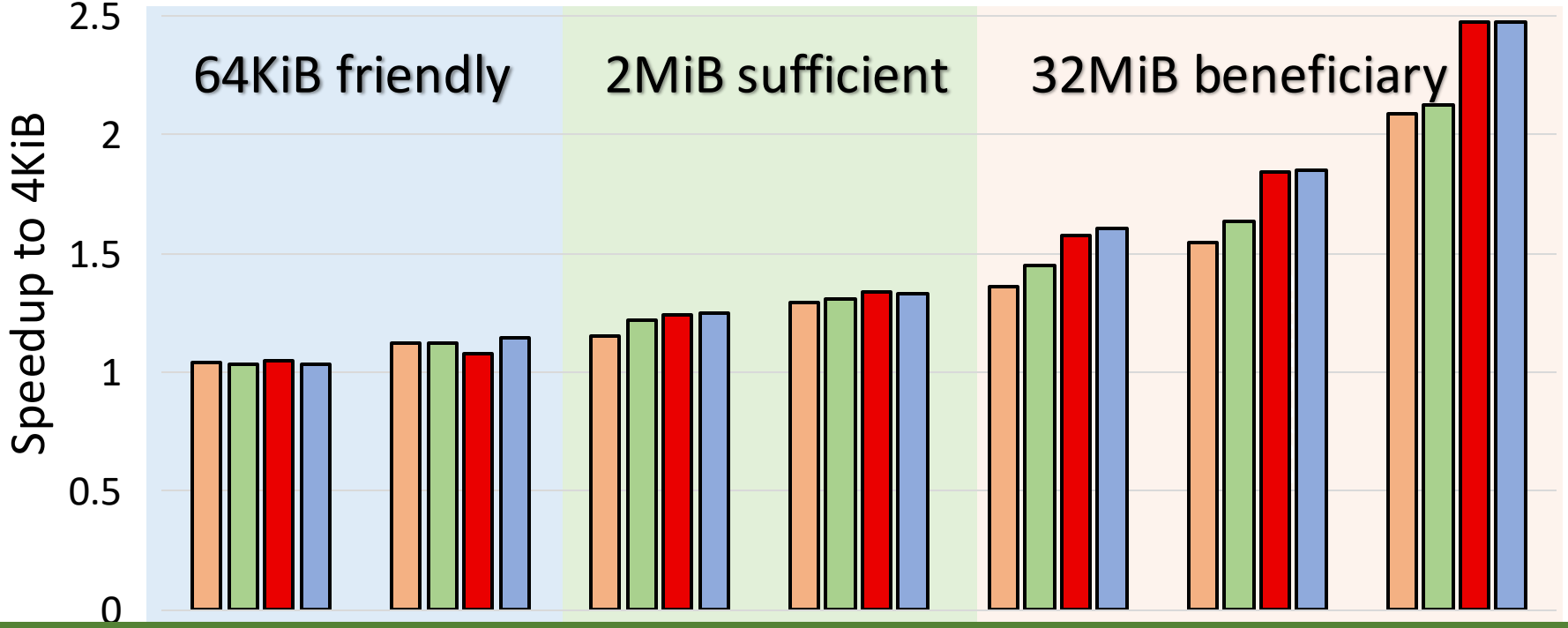


Native, No Fragmentation




Native, No Fragmentation
Speedup to 4KiB
2.5
2
1.5
1
0.5
0
32MiB == 1GiB
99% 64KiB
99% 64KiB
99% 32MiB
99% 32MiB
Streamcl.
Omnetpp
Canneal
XSBench
Svm
Hashjoin
Gups
mTHP
Hawkeye
ET
1GiB

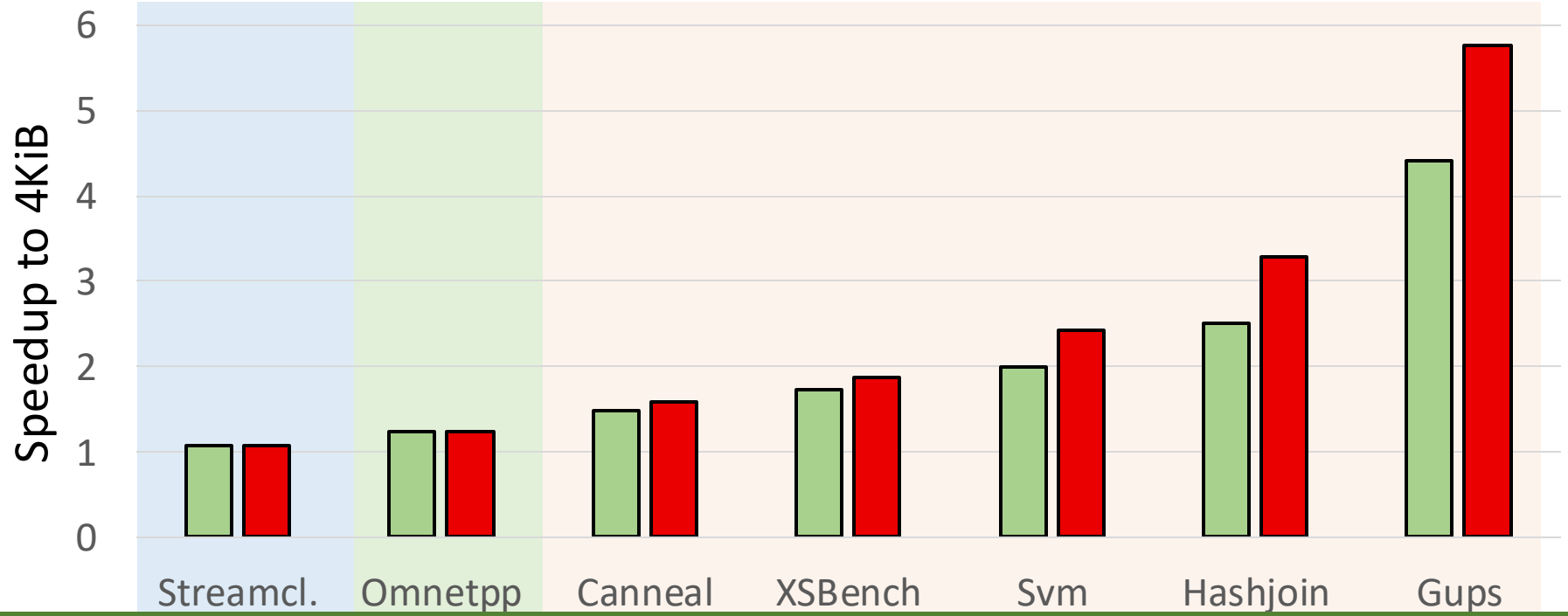
Native, No Fragmentation



One size does not fit all!

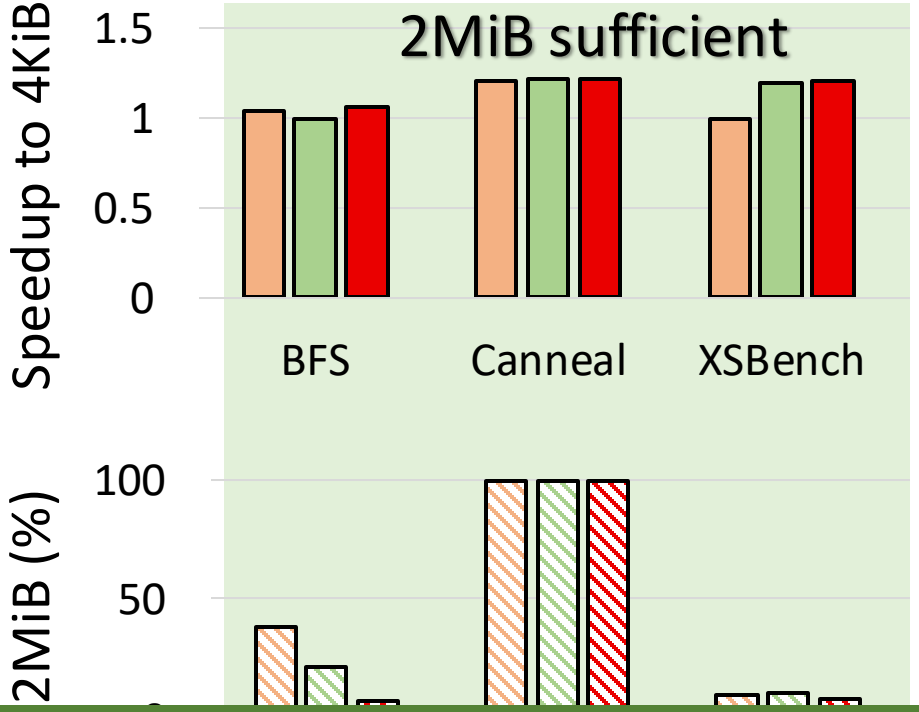
ET 64KiB and 32MiB translations can be exploited to address limitations of the 2MiB / 1GiB model

Virtualized, No Fragmentation



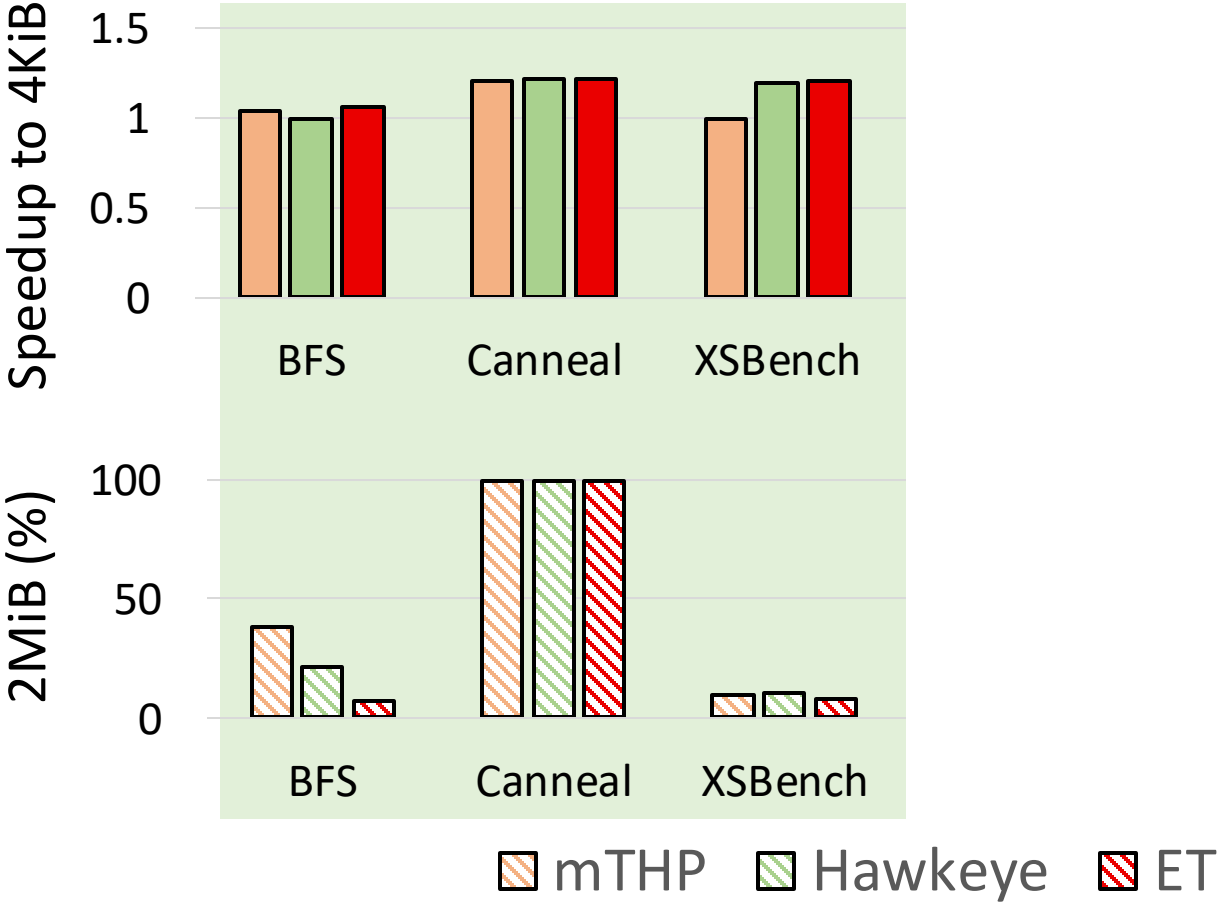
Nested paging makes TLB misses costlier, pronouncing the benefits of 32MiB translations and boosting performance up to 30% over Hawkeye

Native, 99% Fragmentation

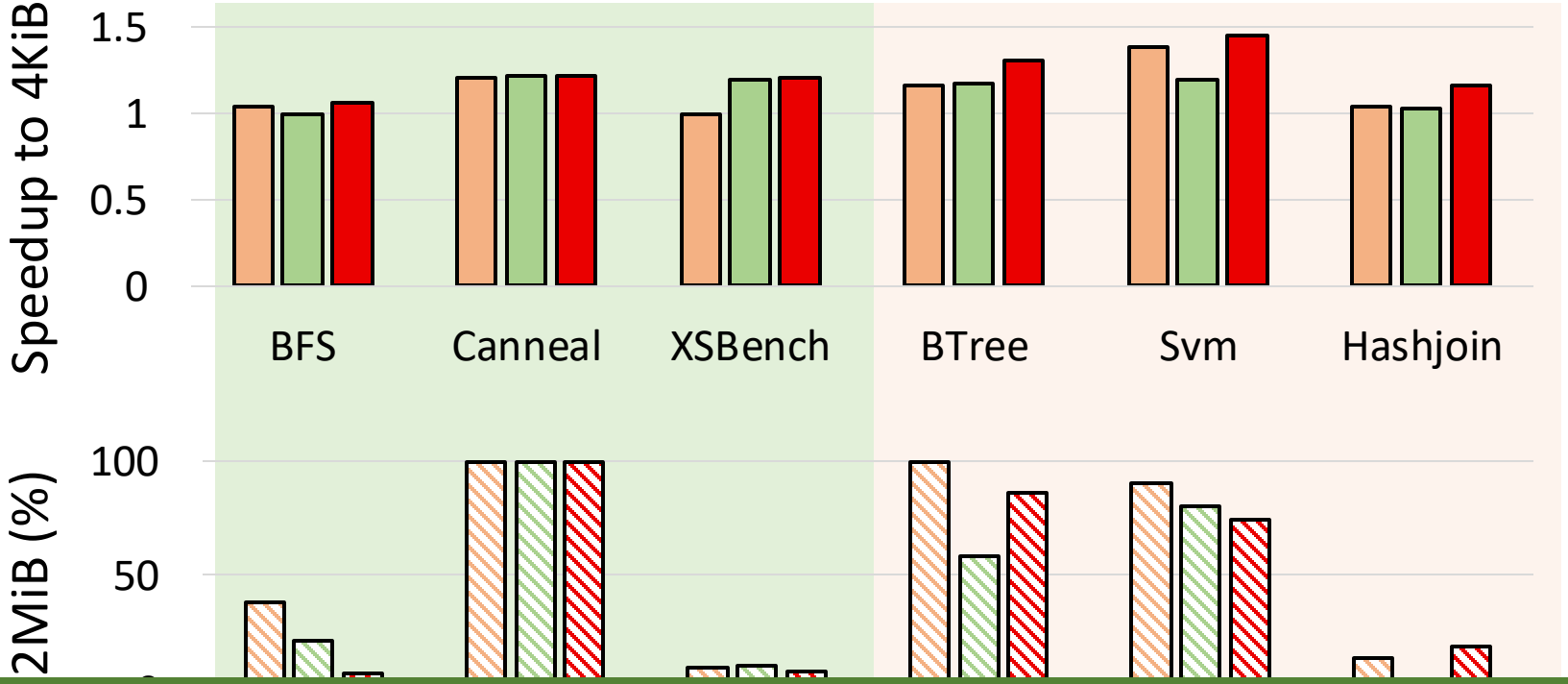


Leshy accurately identifies MMU hotspots, sustaining performance while reducing 2MiB usage by up to 15% over Hawkeye

Native, 99% Fragmentation



Native, 99% Fragmentation



Leshy guidance allows ET to optimally use all translation sizes to boost performance by 10% on average while reducing 2MiB usage

Outline

→ OS-assisted TLB coalescing

→ Elastic Translations

i. CoalaPaging for practical contiguity

ii. Transparent Contig Bit Management

iii. Leshy for translation size selection

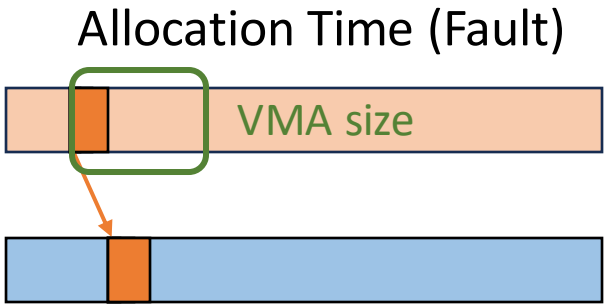
→ Evaluation

→ Conclusion

Conclusion

Artifact available on Github

How to pick translation size?



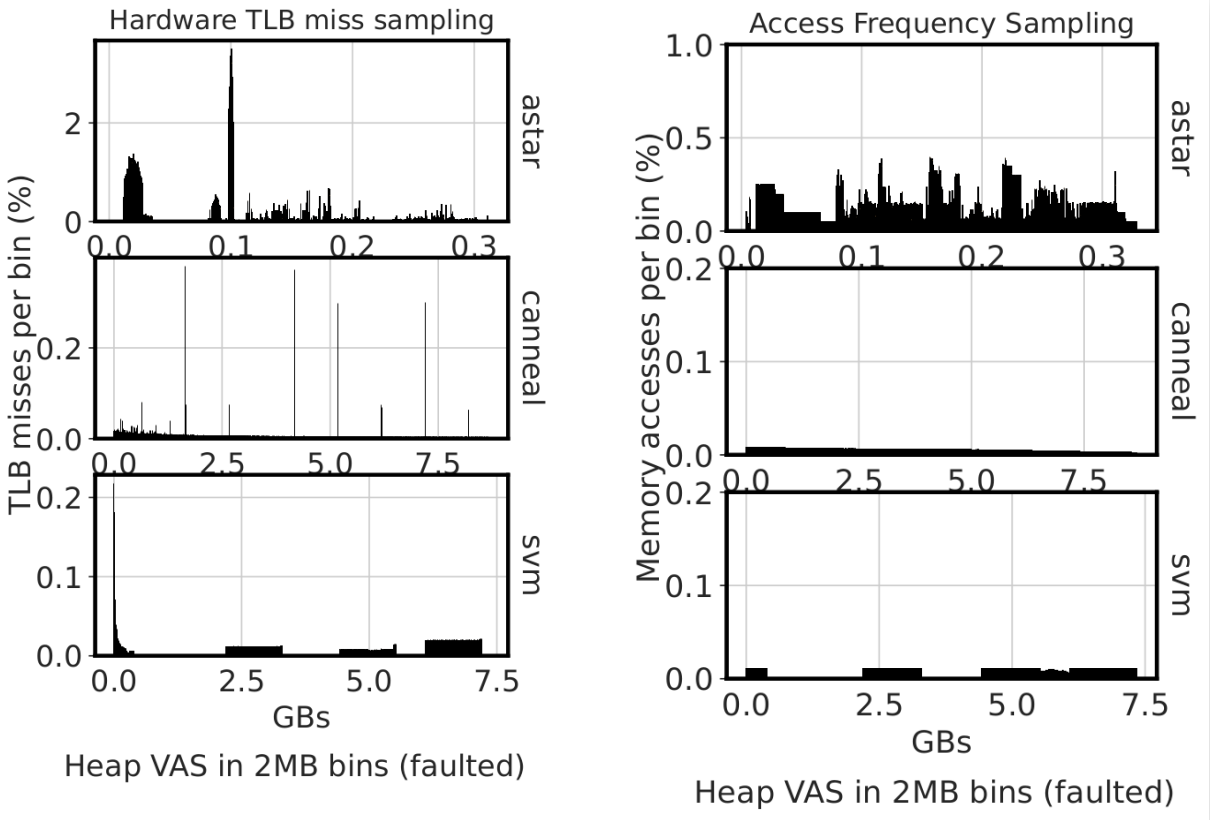
Aggressive and fallback

m(THP): 2MiB → 64 KiB → 4KiB
Trident: 1GiB → 2MiB → 4KiB
Hawkeye: 2MiB → 4KiB

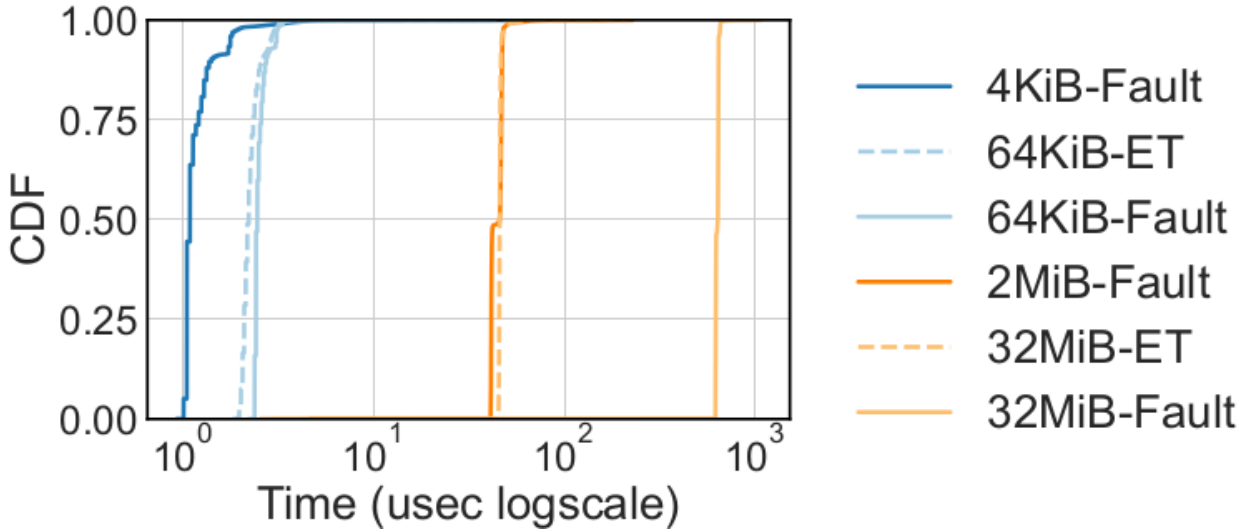
Potential waste of large pages

Elastic Translations: Use mapping size as proxy

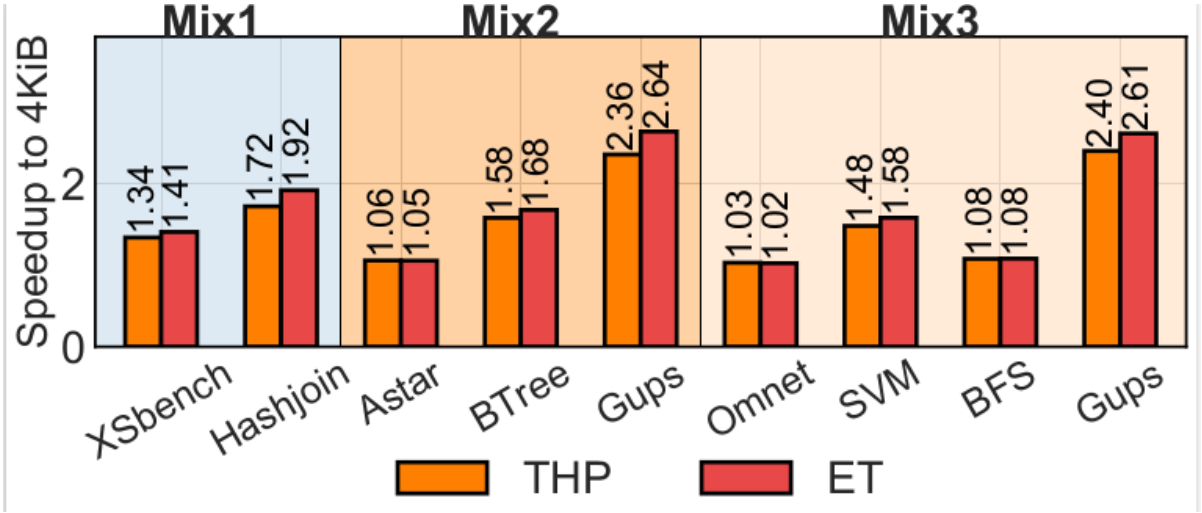
TLB miss sampling vs access bit tracking



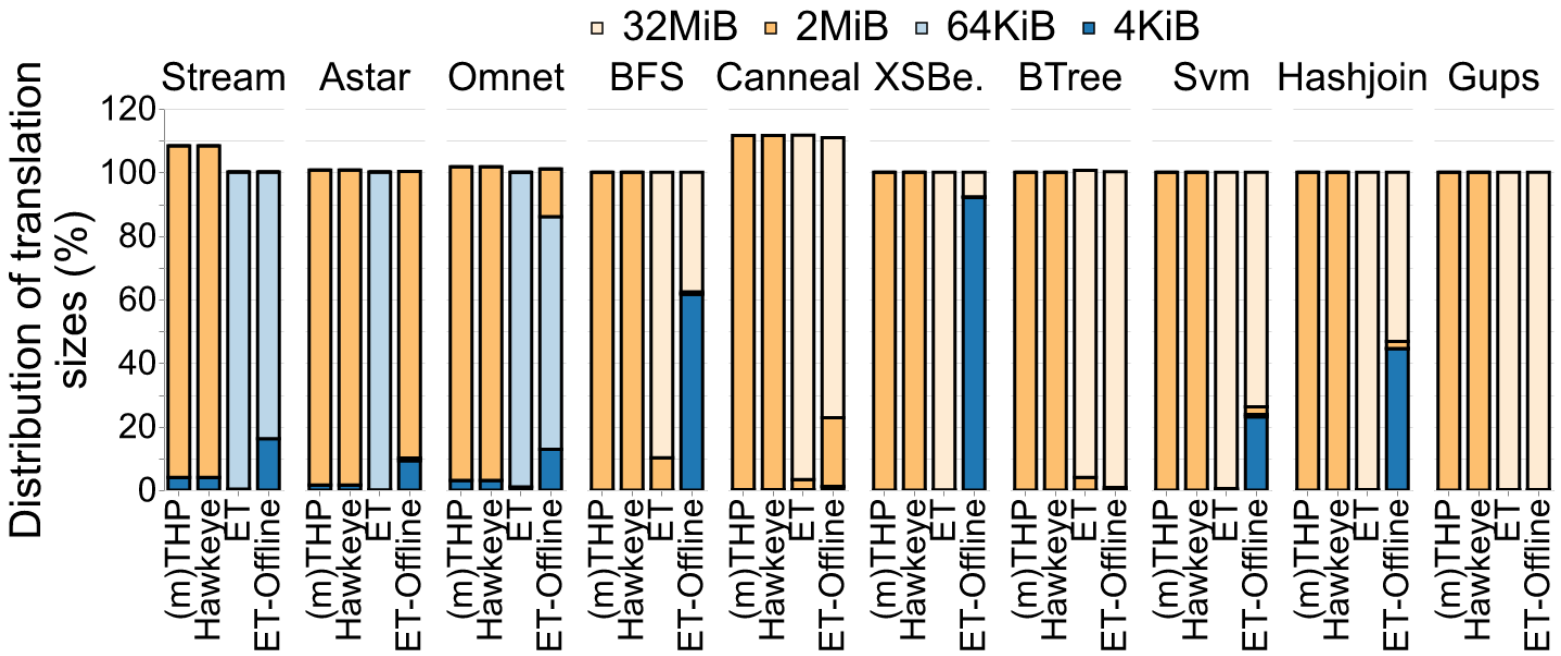
Page Fault Latency



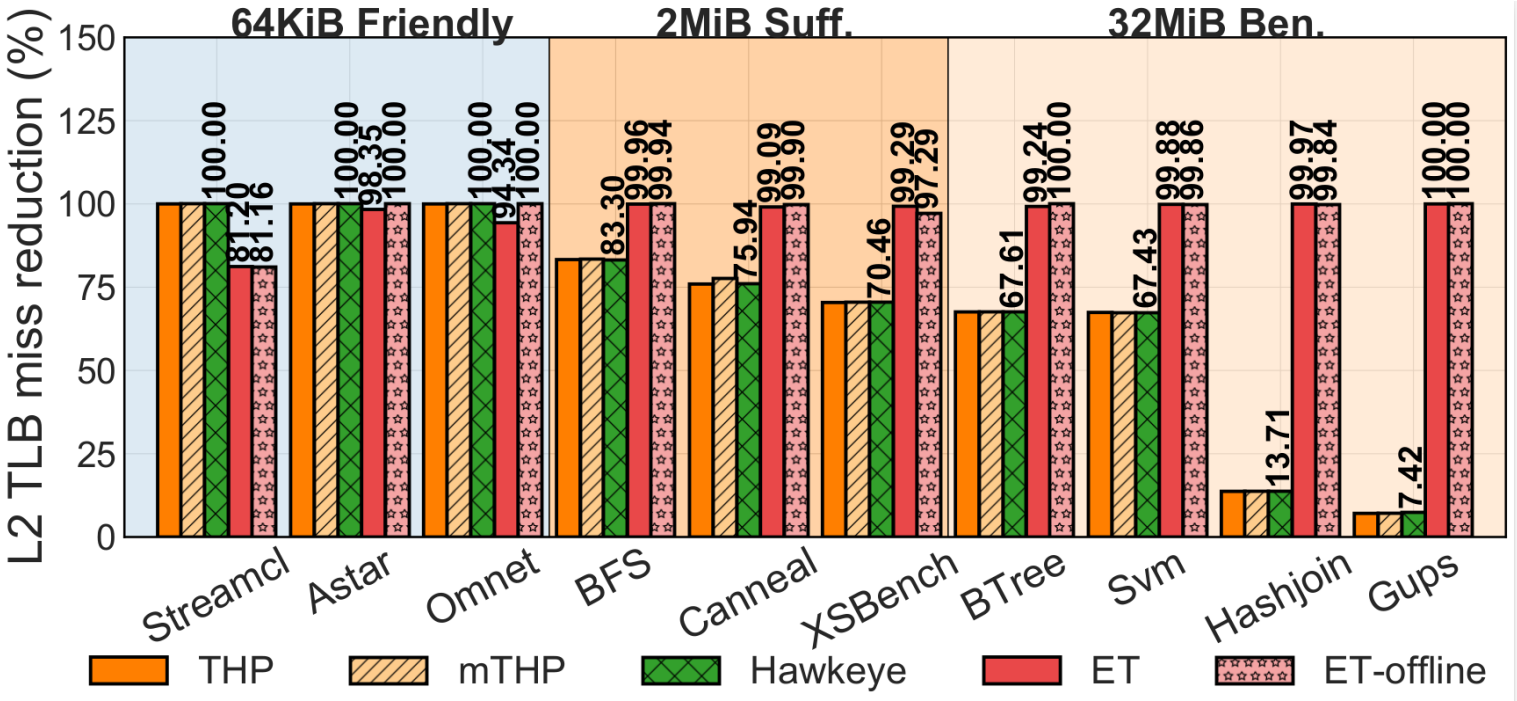
Multi-program



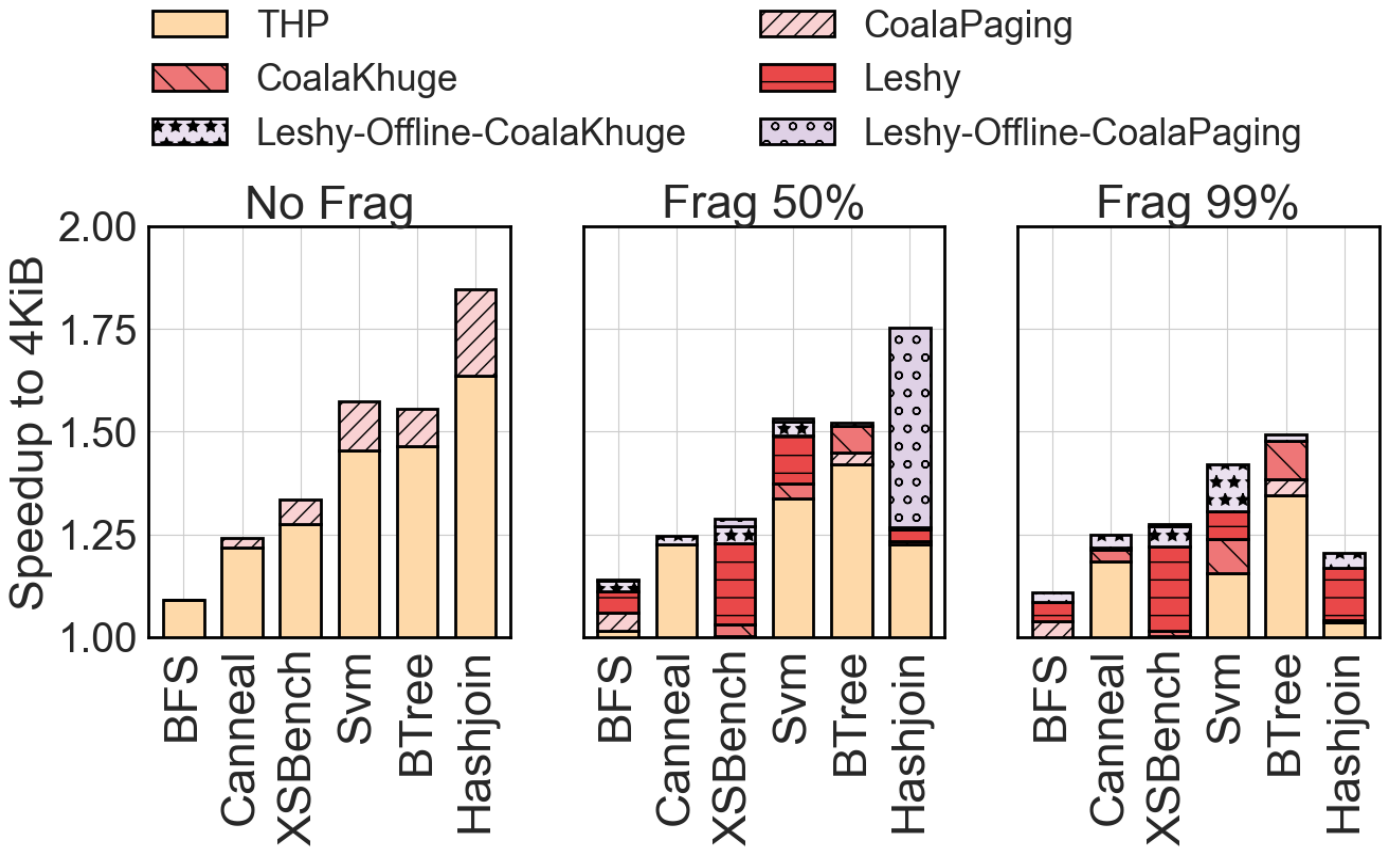
Page Distribution



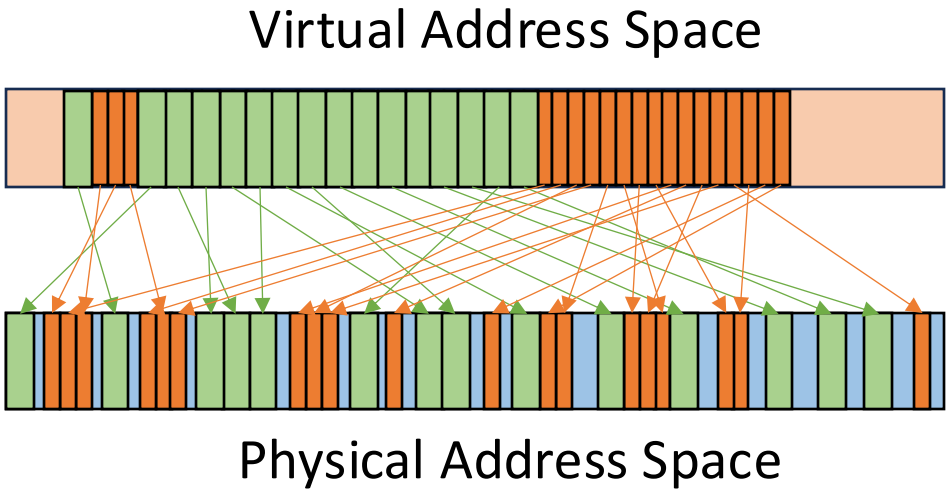
TLB misses


64KiB Friendly
2MiB Suff.
32MiB Ben.
L2 TLB miss reduction (%)
150
125
100
75
50
25
0
100.00
81.20
81.16
100.00
98.35
100.00
100.00
94.34
100.00
83.30
99.96
99.94
75.94
99.09
99.90
70.46
99.29
97.29
67.61
99.24
100.00
67.43
99.88
99.86
13.71
99.97
99.84
7.42
100.00
100.00
Streamcl
Astar
Omnet
BFS
Canneal
XSBench
BTree
Svm
Hashjoin
Gups
THP
mTHP
Hawkeye
ET
ET-offline

Breakdown


THP
CoalaKhuge
Leshy-Offline-CoalaKhuge
CoalaPaging
Leshy
Leshy-Offline-CoalaPaging
No Frag
Frag 50%
Frag 99%
Speedup to 4KiB
2.00
1.75
1.50
1.25
1.00
BFS
Canneal
XSBench
Svm
BTree
Hashjoin

How to pick translation size?



- m(THP): promote blindly everything to 2MiB
- Trident: promote blindly everything to 1GiB

Greedy

- Hawkeye: promote frequently accessed to 2MiB

→ Bad proxy
Binary decision

Elastic Translations: Size Selection

Fault-time allocations

- Employ CoalaPaging to opportunistically and greedily create the maximum achievable contiguity.
- CoalaPaging switches between 64KiB and 32MiB fault-time contiguity generation based on a configurable threshold for VMA size

Asynchronous Promotions

- Use Leshy for online MMU overhead profiling and generate translation size hints for the application footprint.
- CoalaKhugepaged uses these translation size hints to optimally select between the 4 available translation sizes (4KiB / 64KiB / 2MiB / 32MiB) and guide asynchronous promotions