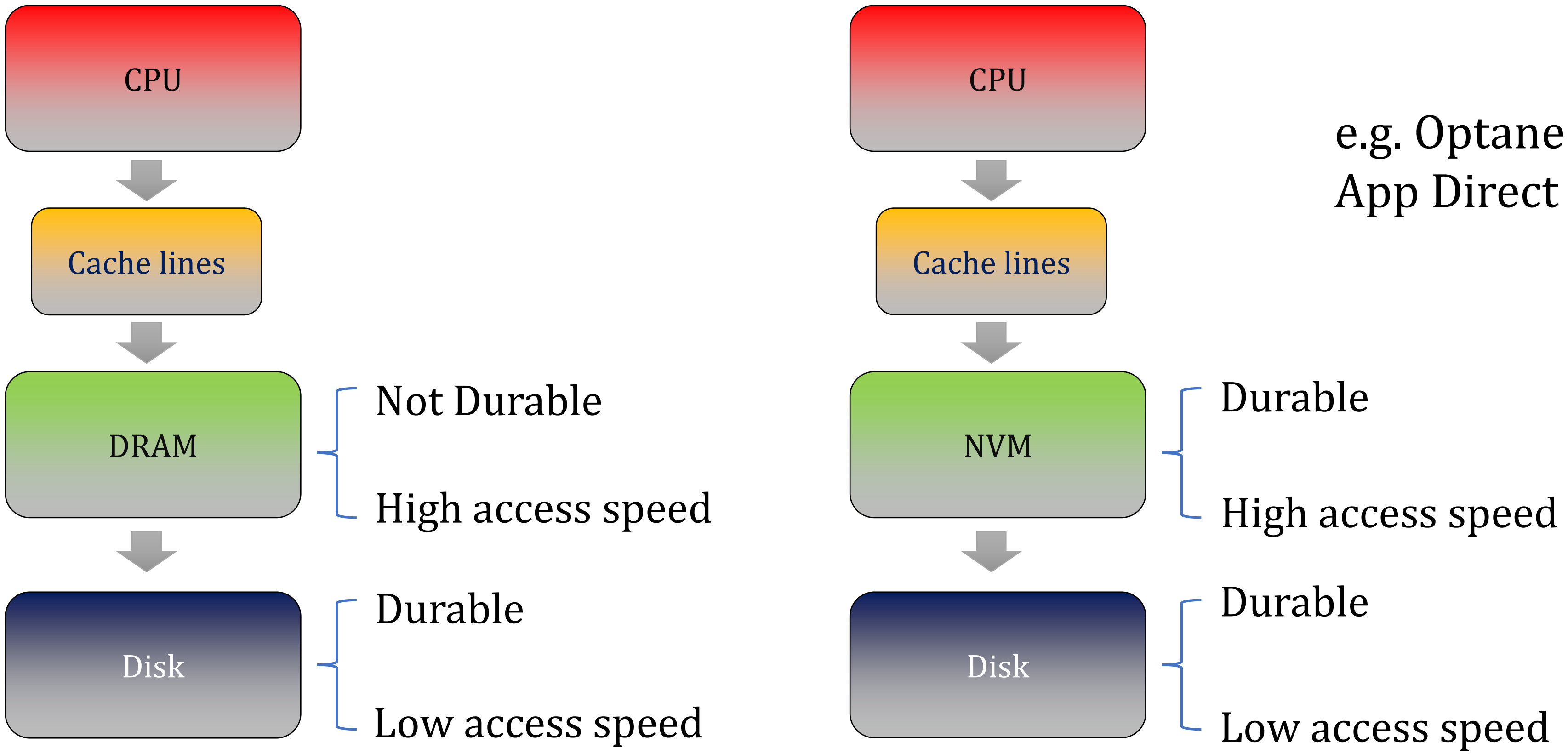
Boosting the Search Performance of B+-tree with Sentinels for Non-volatile Memory

Chongnan Ye, Chundong Wang

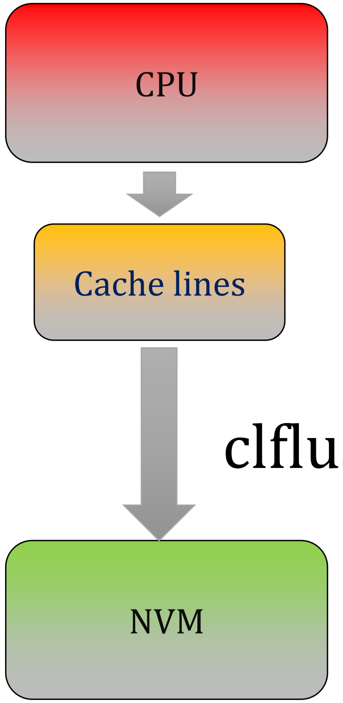
ShanghaiTech University, Shanghai, China

Embedded Architecture: Non-volatile memory



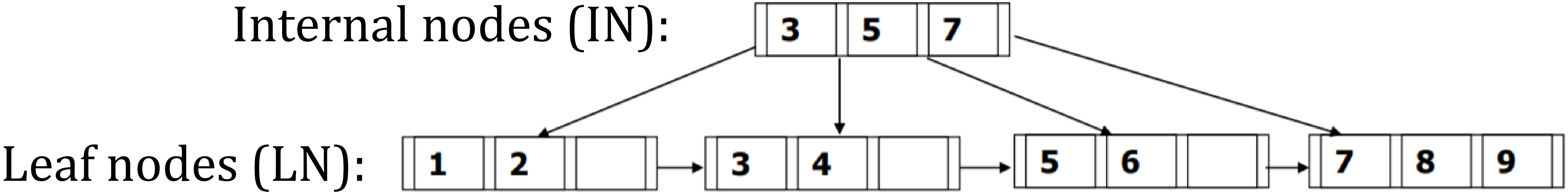
e.g. Optane DC
App Direct mode

Embedded Architecture: Non-volatile memory



clflush, clwb, etc. Large overhead.

Widely used in KV store: B+-tree



Variety of B+-tree on NVM:

FAST-FAIR (FAST’ 18)

Circle-Tree (TC’ 21)

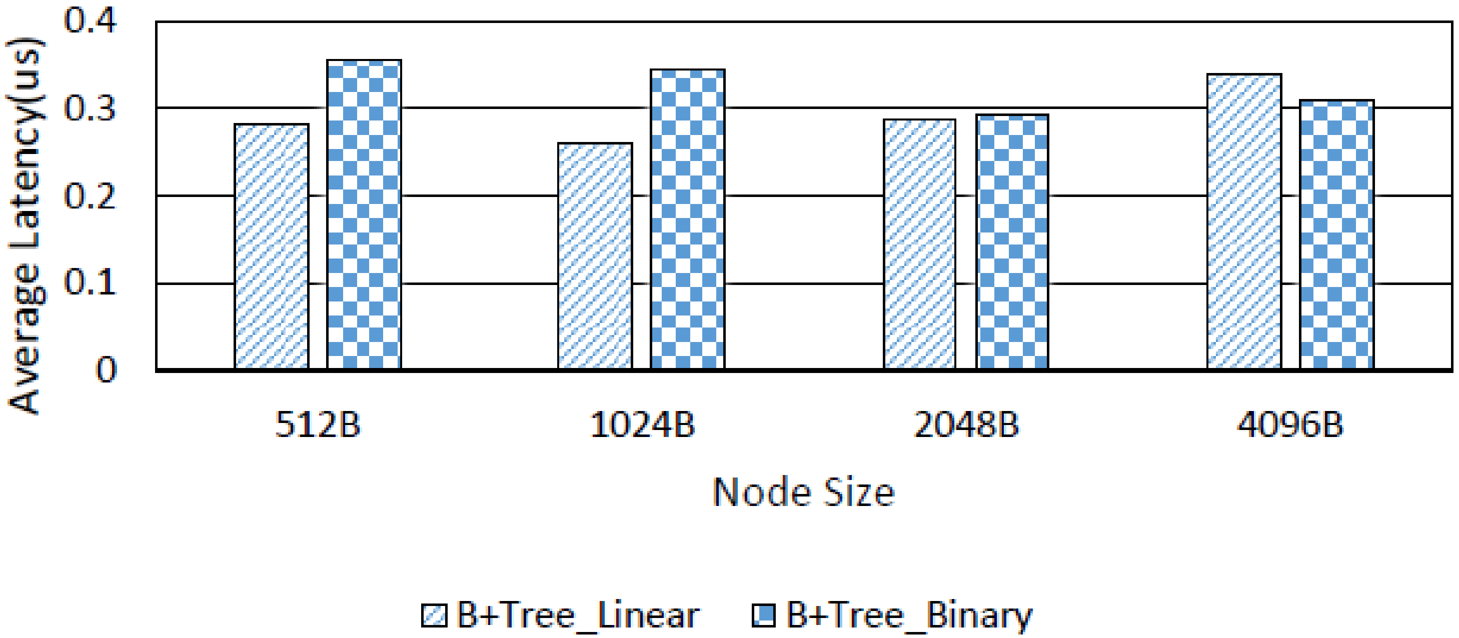
Motivation

Read performance optimization of B+-tree?

Motivation

Inspired by the cache organization from the previous researches:

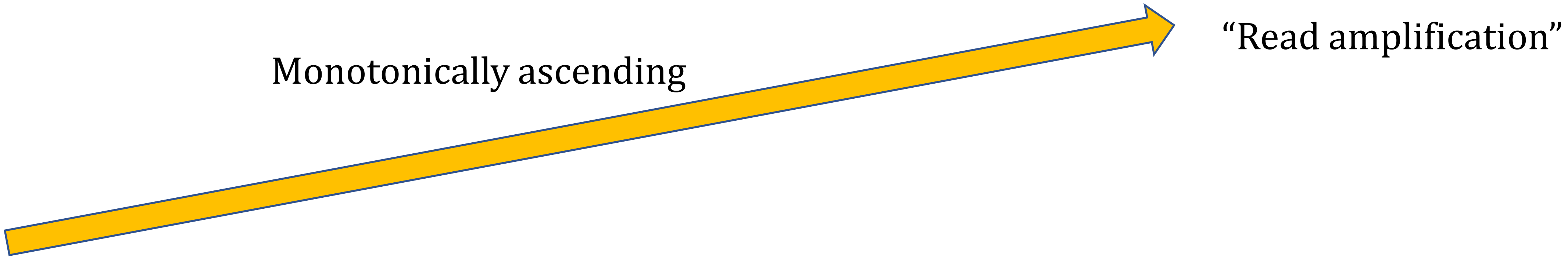
I. Linear search outperforms binary search in a sorted B+-tree node.



Motivation

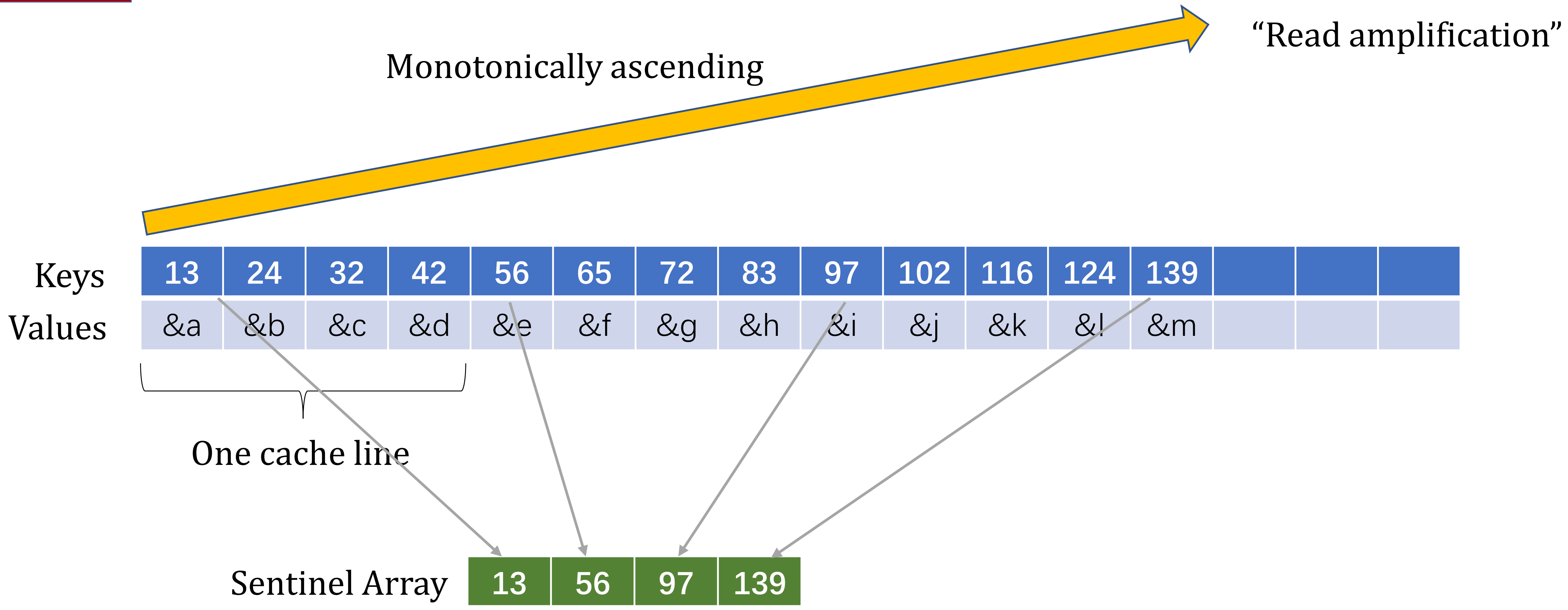
Inspired by the cache organization from the previous researches:

II. Sorted B+-tree nodes contain monotonically ascending keys.



Keys	13	24	32	42	56	65	72	83	97	102	116	124	139			
Values	&a	&b	&c	&d	&e	&f	&g	&h	&i	&j	&k	&l	&m			

Design



Monotonically ascending

“Read amplification”

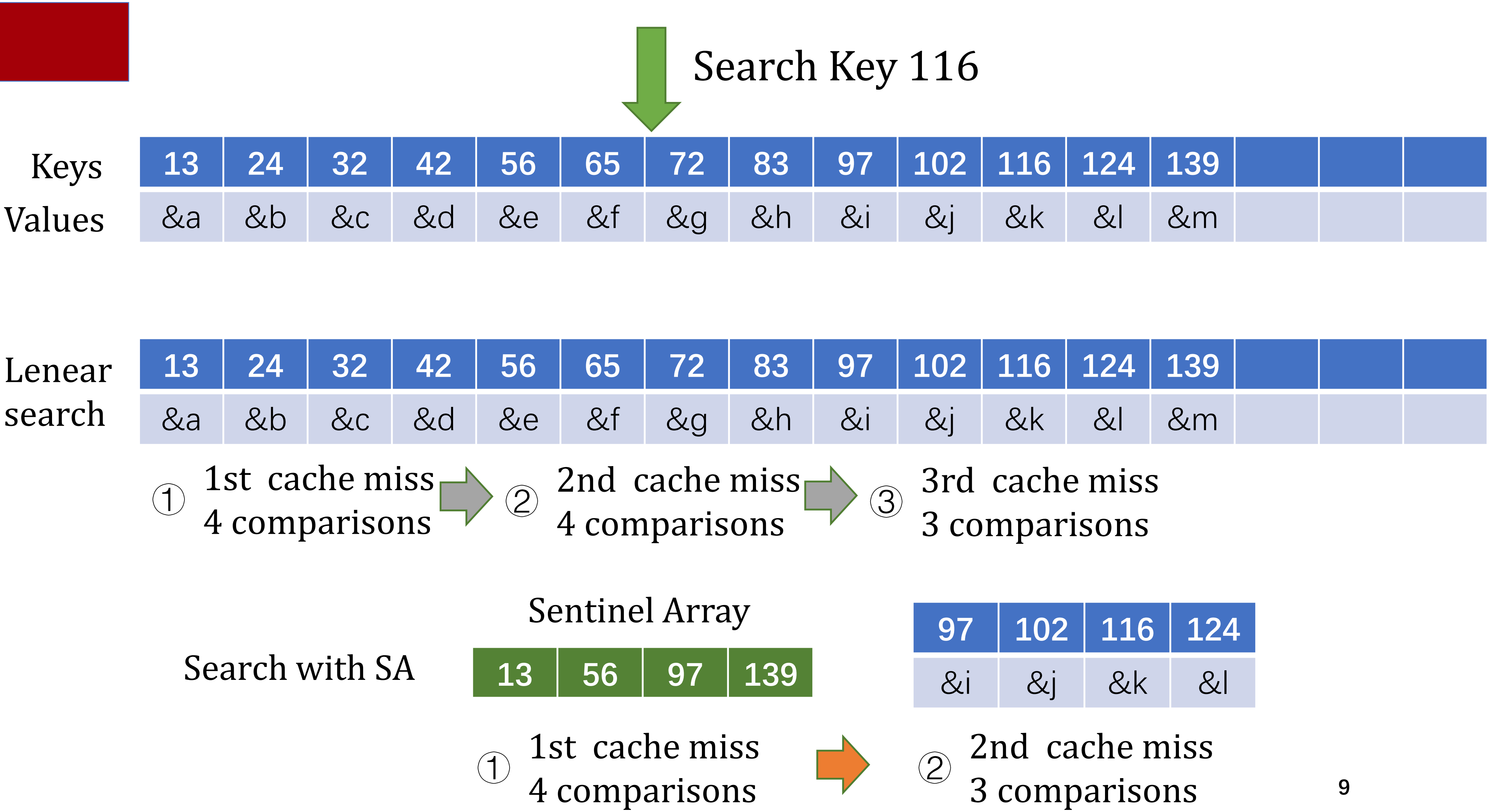
Keys	13	24	32	42	56	65	72	83	97	102	116	124	139			
Values	&a	&b	&c	&d	&e	&f	&g	&h	&i	&j	&k	&l	&m			

One cache line

Sentinel Array	13	56	97	139

Each sentinels is the smallest key in the corresponding cache line

Design



Search Key 116

Keys	13	24	32	42	56	65	72	83	97	102	116	124	139			
Values	&a	&b	&c	&d	&e	&f	&g	&h	&i	&j	&k	&l	&m			

Lenear search	13	24	32	42	56	65	72	83	97	102	116	124	139			
	&a	&b	&c	&d	&e	&f	&g	&h	&i	&j	&k	&l	&m			

① 1st cache miss
4 comparisons

② 2nd cache miss
4 comparisons

③ 3rd cache miss
3 comparisons

Search with SA

Sentinel Array

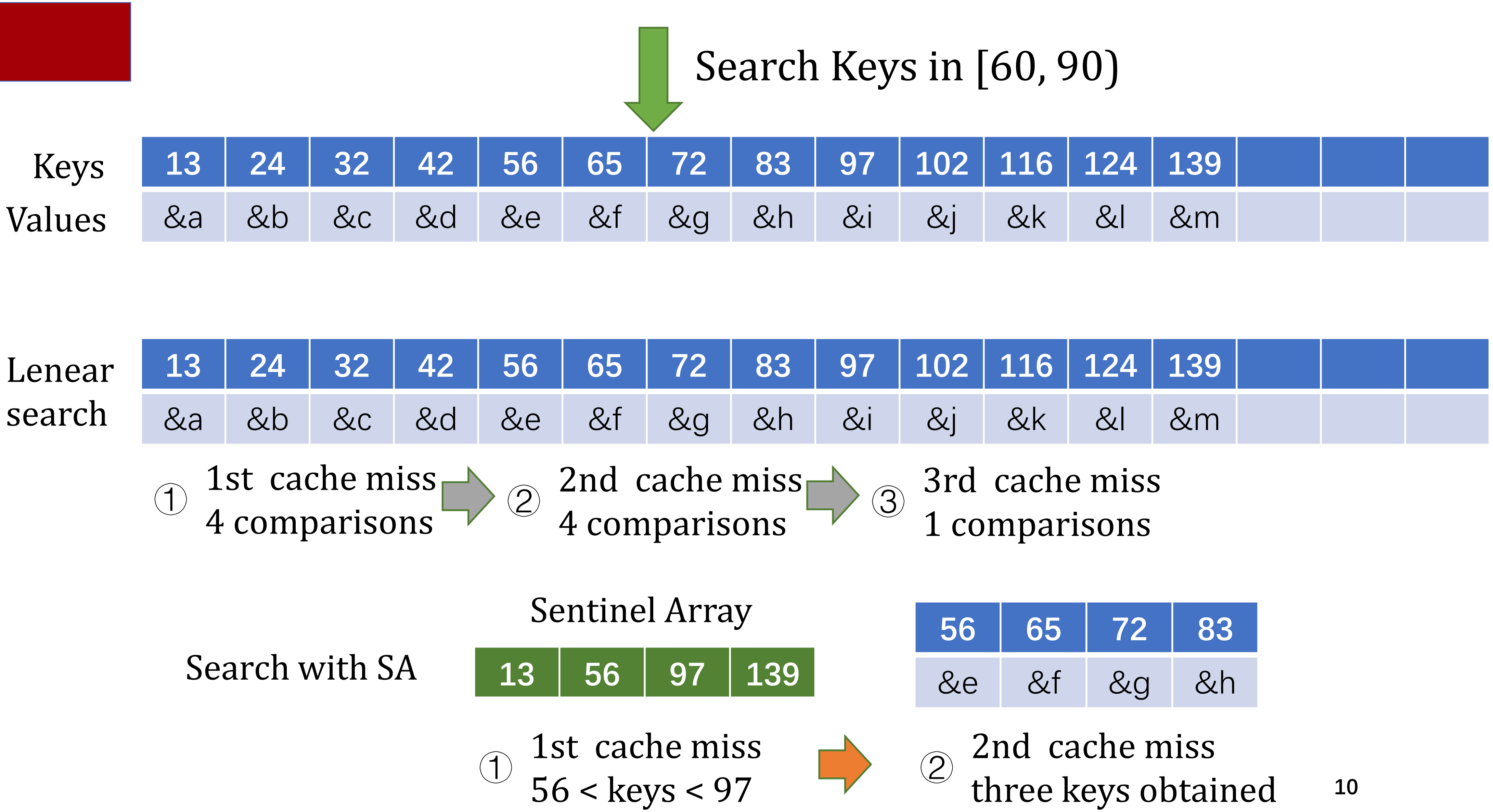
13	56	97	139

97	102	116	124
&i	&j	&k	&l

① 1st cache miss
4 comparisons

② 2nd cache miss
3 comparisons

Design



Search Keys in [60, 90)

Keys	13	24	32	42	56	65	72	83	97	102	116	124	139			
Values	&a	&b	&c	&d	&e	&f	&g	&h	&i	&j	&k	&l	&m			

Lenear search	13	24	32	42	56	65	72	83	97	102	116	124	139			
	&a	&b	&c	&d	&e	&f	&g	&h	&i	&j	&k	&l	&m			

① 1st cache miss
4 comparisons

② 2nd cache miss
4 comparisons

③ 3rd cache miss
1 comparisons

Search with SA

Sentinel Array

13	56	97	139

56	65	72	83
&e	&f	&g	&h

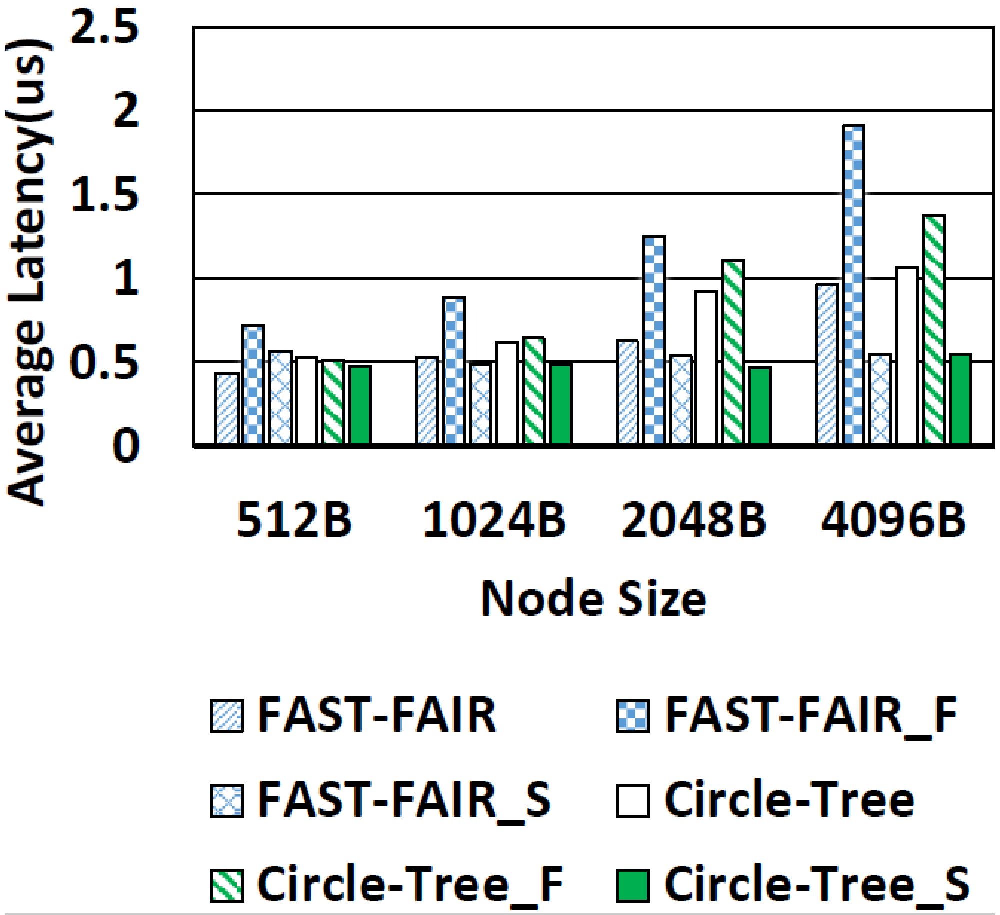
① 1st cache miss
56 < keys < 97

② 2nd cache miss
three keys obtained

Evaluation

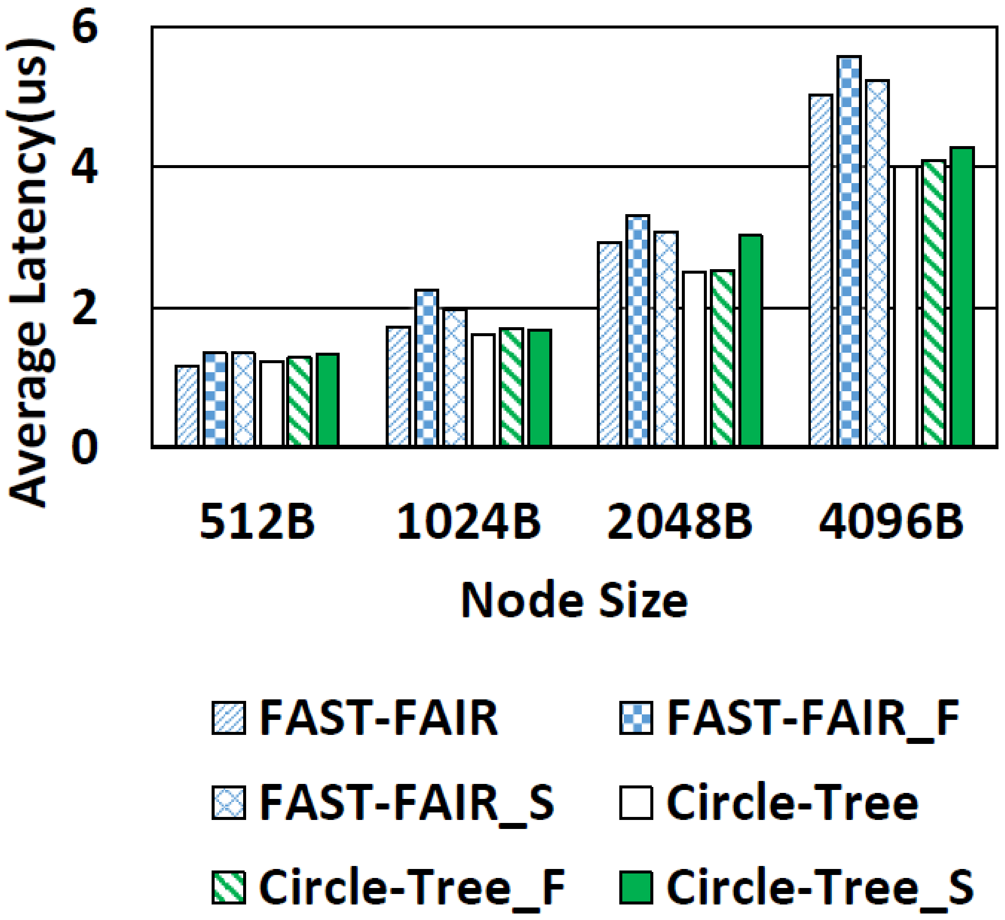
System	Linux Server
CPU	Intel Xeon E5-2620v4 2.10GHz
Caches	512KB/2MB/20MB L1/L2/L3
DRAM(add 300ns write latency to emulate NVM)	8GB

Evaluation



Search performance:
In 4096B size node
FAST-FAIR_S improves 42.6%
Circle-Tree_S improves 48.4%

Evaluation

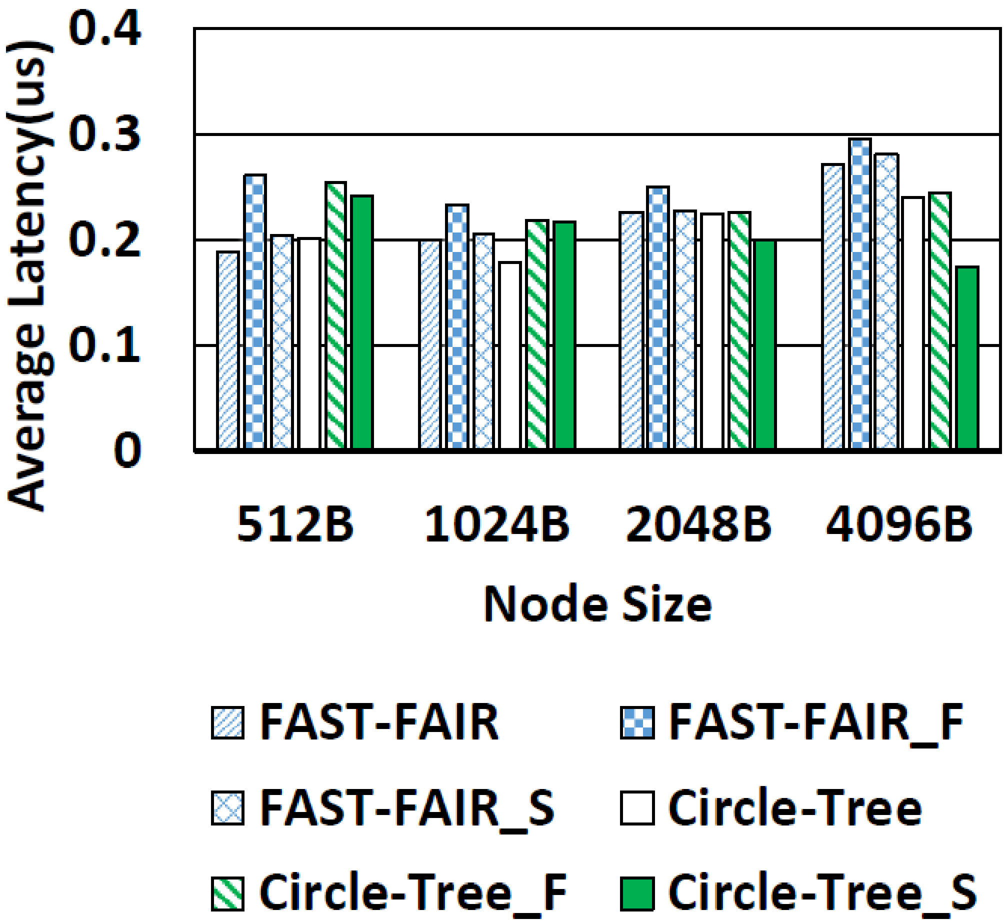


Insertion performance in 4096B size node:

FAST-FAIR_S overhead 4.0%

Circle-Tree_S overhead 6.5%

Evaluation



YCSB Search tail latency
in 4096B size node:
FAST-FAIR_S gets 16.9% improvement
Circle-Tree_S gets 13.0% improvement

Conclusion

- We proposed a sentinel array to reduce the read amplification of searching the ascending B+-tree sorted node.
- The results show that our design reduces the cache misses and obtain the performance improvement for in-NVM B+-tree.

Thank you for your listening!