Space-Efficient, High-Performance Rank & Select Structures

Dong Zhou (CMU), David G. Andersen (CMU), Michael Kaminsky (Intel Labs)

Background – Rank & Select

A fundamental building blocks for ***succinct data structures***

- monotone sequences of integers
- binary of n-ary trees

Given a bit array *B* of length *n* (zero indexed)

- Rank(*x*): the number of 1s up to position *x*
- Select(*y*): the position of *y*-th 1

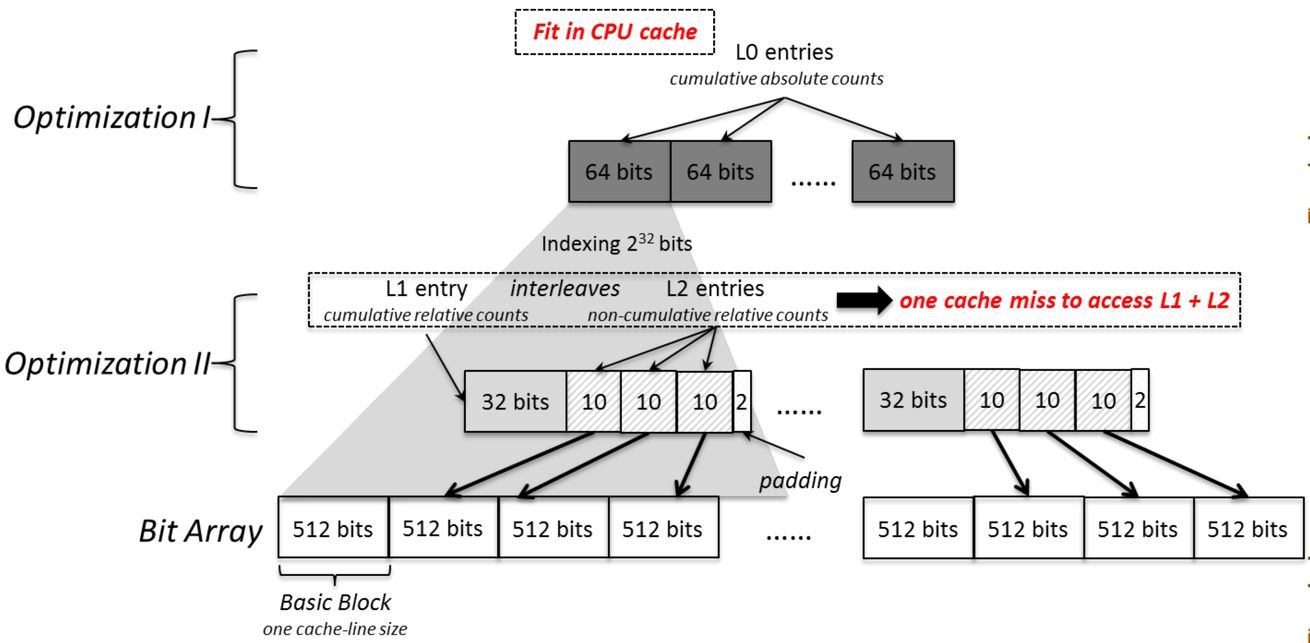
index	0	1	2	3	4
B	0	1	0	1	0

- Rank(2) = 1, Select(2) = 3

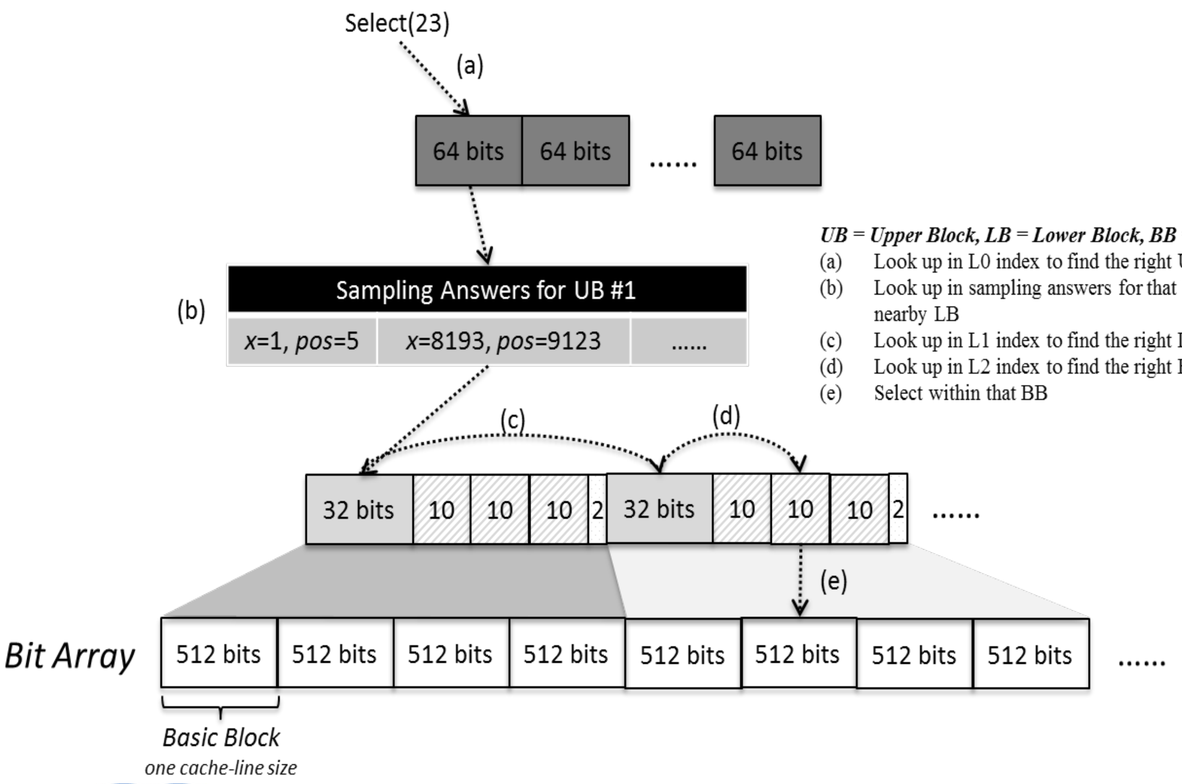
Architectural Insights

- Performance strongly determined by cache misses
- Parallel operations are cheap
- Optimize for cache misses, then branches, then arithmetical/logical operations

Rank Structure



Select Structure



UB = Upper Block, LB = Lower Block, BB = Basic Block

(a) Look up in L0 index to find the right UB
(b) Look up in sampling answers for that UB to find a nearby LB
(c) Look up in L1 index to find the right LB
(d) Look up in L2 index to find the right BB
(e) Select within that BB

Results Overview

Rank

Approach	Space Overhead	Max Supported Size
Ours (*poppy*)	3.125%	2^{64}
rank9	25%	2^{64}
combined sampling	3.125%	2^{32}

Select

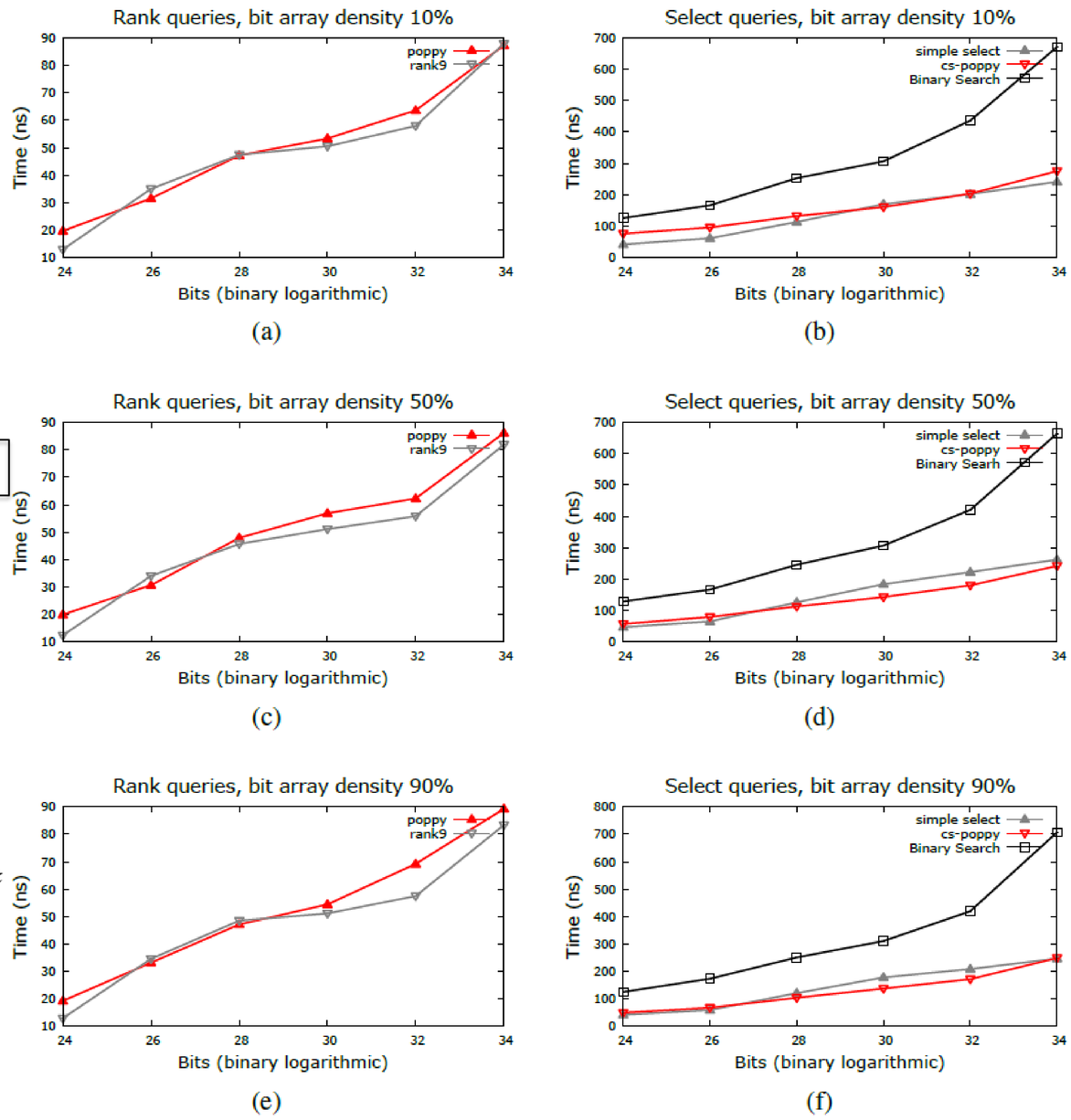
Approach	Space Overhead	Max Supported Size
Ours (*cs-poppy*)	~0.39%	2^{64}
select9	9.01%-45.94%	2^{64}
combined sampling	~0.39%	2^{32}

Observation

The key problem is to ***efficiently*** increase the number of bits that can be processed with no auxilliary information

- The `popcnt` instruction is the fastest
- The bit array should be operated in the granularity of cache-line size

Comparable or Better Performance



Conclusion

Our Rank & Select structures

- Support bit arrays with up to 2^{64} bits
- Add no more than 3.125% / 0.39% extra space
- Offer performance competitive to the state-of-the-art

Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley