Functional Hash Maps in a Data Parallel Language

William Henrich Due [1] Martin Elsman [1] Troels Henriksen [1]

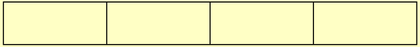
[1]Department of Computer Science, University of Copenhagen

August 22nd, 2025

Contact: `widu@di.ku.dk`

Open Adressing Example

- Keys $k_0, k_1 \in K$.
- Hash function $h : K \to \{0, 1, 2, 3\}$.
- $h(k_0) = h(k_1) = 1$.



Open Adressing Example

- Keys $k_0, k_1 \in K$.
- Hash function $h : K \to \{0, 1, 2, 3\}$.
- $h(k_0) = h(k_1) = 1$.

	k_0		

Open Adressing Example

- Keys $k_0, k_1 \in K$.
- Hash function $h : K \to \{0, 1, 2, 3\}$.
- $h(k_0) = h(k_1) = 1$.

	k_0, k_1		

Open Adressing Example

- Keys $k_0, k_1 \in K$.
- Hash function $h : K \to \{0, 1, 2, 3\}$.
- $h(k_0) = h(k_1) = 1$.

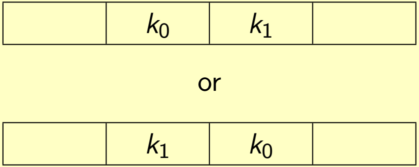
	k_0	k_1	

Core Ideas

- Concurrency.

Core Ideas

- Concurrency.
- Collision resolution.



k_0 k_1

or

k_1 k_0

Core Ideas

- Concurrency.
- Collision resolution.
- Functional Array Languages.

	k_0	k_1	

or

	k_1	k_0	

$$\texttt{map} : (\alpha \to \beta) \to [n]\alpha \to [n]\beta$$

Perfect Hashing with FKS

- Find a collision-free hash function.
- $\{k_0, k_1, k_2\} \subseteq K$.
- Pick some $h \in H$ where H is a universal hash family.

$h(k_0)$	$h(k_1)$	$h(k_2)$
0	0	1

Perfect Hashing with FKS

- Find a collision-free hash function.
- $\{k_0, k_1, k_2\} \subseteq K$.
- Pick some $h \in H$ where H is a universal hash family.
- Pick perfect hash functions h_0, h_1, h_2 from universal hash families.

$h(k_0)$	$h(k_1)$	$h(k_2)$
0	0	1

Bin Size	2	1	0
Subhash Map Size	2^2	1^2	0^2
Offset (o_i)	0	4	5
Hash Function	h_0	h_1	h_2

Perfect Hashing with FKS

- Find a collision-free hash function.
- $\{k_0, k_1, k_2\} \subseteq K$.
- Pick some $h \in H$ where H is a universal hash family.
- Pick perfect hash functions h_0, h_1, h_2 from universal hash families.

$h(k_0)$	$h(k_1)$	$h(k_2)$
0	0	1

Bin Size	2	1	0
Subhash Map Size	2^2	1^2	0^2
Offset (o_i)	0	4	5
Hash Function	h_0	h_1	h_2

	$o_0 + h_0(k_0)$		$o_0 + h_0(k_1)$	$o_1 + h_1(k_2)$
	k_0		k_1	k_2

Finding collision-free hash functions

- Pick hash functions h_i for every bin.
- Compute $o_i + h_i(k)$ for every k.
- Compute a histogram to count the number of collisions.
- Using a segmented scan, check if any subhash map has a collision.
- Partition subhash maps by if they had collision.
- Continue on subhash maps with collisions.

Comparison

	FKS	Open Addressing
Hashing	Universal Hash Family	Any[1]
Lookup	$O(1)$	Expected $O(1)$
Construction	Expected $O(n)$	$O(n)$
Dynamic	Yes[2]	Yes

[1] Technically not true.

[2] Seems impractical.

Benchmarks

	64-bit integer keys ($n = 10^7$)		
	Construction	*Lookup*	*Membership*
Futhark (hash maps)	18.3	3.3	1.6
Futhark (binary search)	40.9	6.2	5.8
Futhark (Eytzinger)	42.3	4.3	2.4
cuCollections	2.7	1.1	0.9

All times in milliseconds.

Benchmarks

	String keys ($n = 10^7$)		
	Construction	*Lookup*	*Membership*
Futhark (hash maps)	33.2	4.3	2.8
Futhark (binary search)	83.0	5.7	5.8
Futhark (Eytzinger)	85.3	5.3	5.3
cuCollections	2.7	1.3	1.2

All times in milliseconds.

The End

Towards Efficient Hash Maps in Functional Array Languages

`https://arxiv.org/abs/2508.11443`

Code

`https://github.com/diku-dk/containers`