Functional Hash Maps in a Data Parallel Language

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

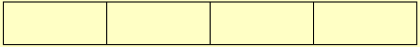
[1]Department of Computer Science, University of Copenhagen

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.

	k_0	k_1	

or

	k_1	k_0	

Core Ideas

- Concurrency.
- Difficult.

	k_0	k_1	

or

	k_1	k_0	

	k_1	k_1	

Hash Maps in Functional Data Parallel Languages

- Avoid collisions.
- Bulk operations.

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

$$\texttt{from\_array} : [\mathit{n}](\alpha, \beta) \to \textbf{hashmap}\ \alpha\ \beta$$

Hash Maps in Functional Data Parallel Languages

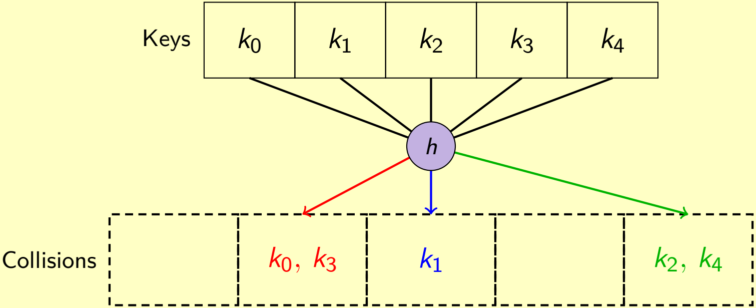
- Avoid collisions.
- Bulk operations.
- Fredman-Komlós-Szemerédi (FKS) construction.

$$\begin{aligned} \texttt{map} &: (\alpha \to \beta) \to [n]\alpha \to [n]\beta \\ \texttt{from\_array} &: [n](\alpha, \beta) \to \textbf{hashmap}\ \alpha\ \beta \end{aligned}$$

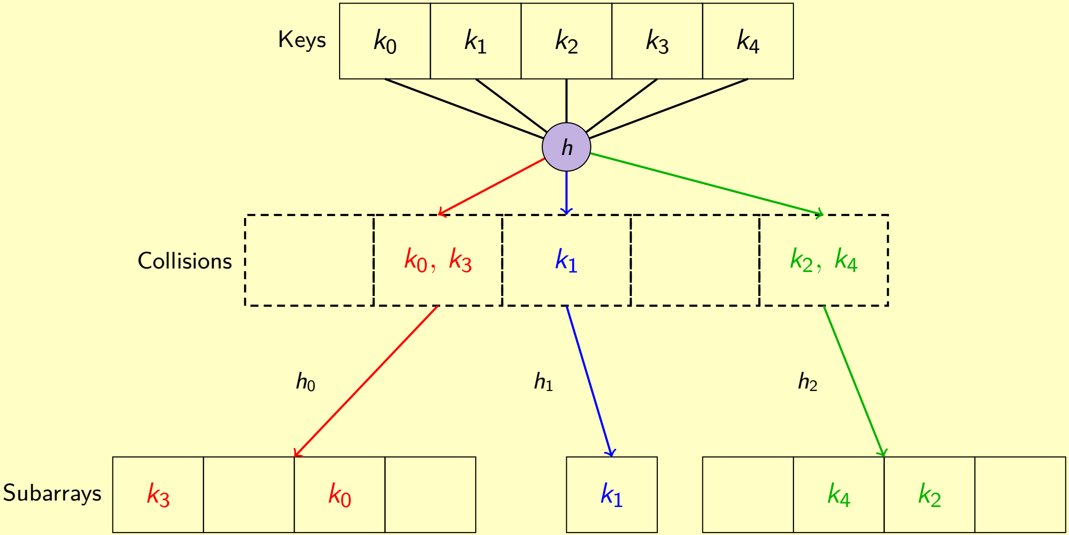
Perfect Hashing with FKS

Keys	k_0	k_1	k_2	k_3	k_4

Perfect Hashing with FKS



Perfect Hashing with FKS



Keys k_0 k_1 k_2 k_3 k_4

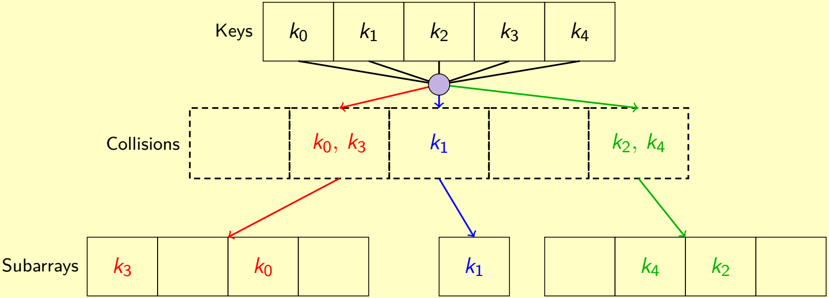
h

Collisions | k_0, k_3 | k_1 | | k_2, k_4

h_0 h_1 h_2

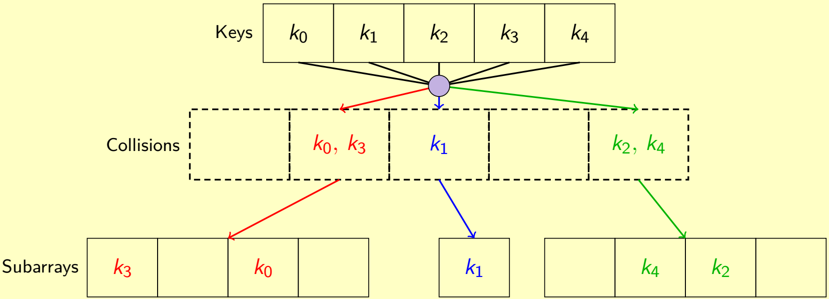
Subarrays k_3 | | k_0 | k_1 | k_4 | k_2 |

Flattening The Finding of Collision-free Hash Functions



```
map $\lambda subarray \rightarrow$
  while $h_i$ leads to collisions do
    Pick a random hash function $h_i$
```

Flattening The Finding of Collision-free Hash Functions



Keys k_0 k_1 k_2 k_3 k_4

Collisions $k_0,\ k_3$ k_1 $k_2,\ k_4$

Subarrays k_3 k_0 k_1 k_4 k_2

```
map $\lambda subarray \rightarrow$
  while $h_i$ leads to collisions do
    Pick a random hash function $h_i$
```

$\mapsto$

```
while any collisions in subarrays do
  map $\lambda keys \rightarrow$ pick new hash functions
```

Benchmarks

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

All times in milliseconds measured on an A100 GPU.

The End

Towards Efficient Hash Maps in Functional Array Languages

https://arxiv.org/abs/2508.11443

Code

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

https://github.com/diku-dk/futhark-hashmap-experiments