

# Functional Hash Maps in a Data Parallel Language

**William Henrich Due**<sup>1</sup>   Martin Elsman<sup>1</sup>   Troels Henriksen<sup>1</sup>

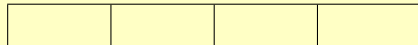
<sup>1</sup>Department of Computer Science, University of Copenhagen

August 22nd, 2025

Contact: [widu@di.ku.dk](mailto:widu@di.ku.dk)

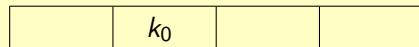
# Open Addressing Example

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



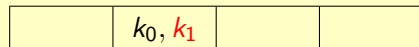
# Open Addressing Example

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



# Open Addressing Example

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



# Open Addressing Example

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

	$k_0$	$k_1$	
--	-------	-------	--

# Core Ideas

- Concurrency.
- Collision resolution.



or



# Hash Maps in Functional Array Language

- Avoid collisions.
- Bulk operations.

$$\text{map} : (\alpha \rightarrow \beta) \rightarrow [n]\alpha \rightarrow [n]\beta$$
$$\text{from\_array} : [n](\alpha, \beta) \rightarrow \mathbf{map} \ \alpha \ \beta$$

# Hash Maps in Functional Array Language

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

$$\text{map} : (\alpha \rightarrow \beta) \rightarrow [n]\alpha \rightarrow [n]\beta$$
$$\text{from\_array} : [n](\alpha, \beta) \rightarrow \mathbf{map} \ \alpha \ \beta$$

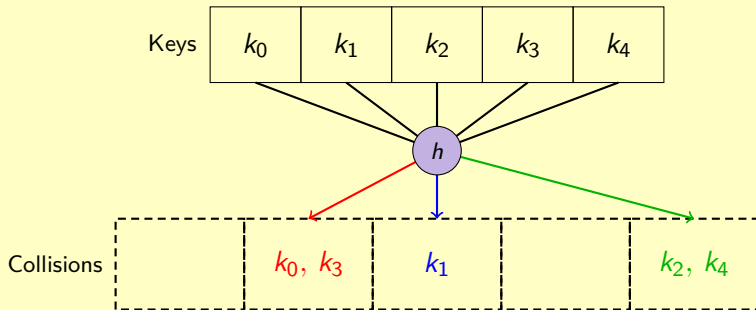


# Perfect Hashing with FKS

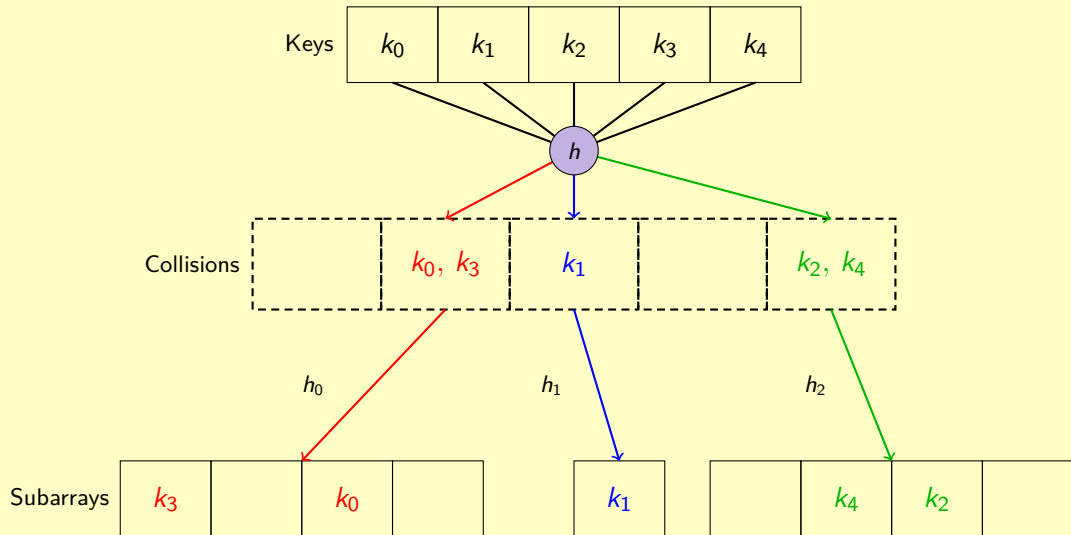
Keys

$k_0$	$k_1$	$k_2$	$k_3$	$k_4$
-------	-------	-------	-------	-------

# Perfect Hashing with FKS



# Perfect Hashing with FKS



# Mapping FKS to GPU

- Irregular parallelism.

# Mapping FKS to GPU

- Irregular parallelism.
- Map flattening:

```
map (map  $f$ ) [[1, 2], [3, 4, 5]]
```

# Mapping FKS to GPU

- Irregular parallelism.
- Map flattening:

$$\begin{aligned} & \text{map } (\text{map } f) \ [[1, 2], [3, 4, 5]] \\ & \equiv [\text{map } f \ [1, 2], \text{map } f \ [3, 4, 5]] \end{aligned}$$

# Mapping FKS to GPU

- Irregular parallelism.
- Map flattening:

$$\begin{aligned} & \text{map } (\text{map } f) \ [[1, 2], [3, 4, 5]] \\ \equiv & \ [\text{map } f \ [1, 2], \text{map } f \ [3, 4, 5]] \\ \equiv & \ [[f \ 1, f \ 2], [f \ 3, f \ 4, f \ 5]] \end{aligned}$$

# Mapping FKS to GPU

- Irregular parallelism.
- Map flattening:

$$\begin{aligned} & \text{map } (\text{map } f) \ [[1, 2], [3, 4, 5]] \\ & \equiv [\text{map } f \ [1, 2], \text{map } f \ [3, 4, 5]] \\ & \equiv [[f \ 1, f \ 2], [f \ 3, f \ 4, f \ 5]] \end{aligned}$$

- Flattening the finding of collision-free hash functions.

$\text{map } \lambda \text{subkeys} \rightarrow$

while  $h$  leads to collisions do

Pick a random hash function  $h$



# Mapping FKS to GPU

- Irregular parallelism.
- Map flattening:

$$\begin{aligned} & \text{map } (\text{map } f) \ [[1, 2], [3, 4, 5]] \\ & \equiv [\text{map } f \ [1, 2], \text{map } f \ [3, 4, 5]] \\ & \equiv [[f \ 1, f \ 2], [f \ 3, f \ 4, f \ 5]] \end{aligned}$$

- Flattening the finding of collision-free hash functions.

$\text{map } \lambda \text{subkeys} \rightarrow$

$\text{while } h \text{ leads to collisions do}$

Pick a random hash function  $h$

$\mapsto$

$\text{while any collision in subarrays do}$

$\text{map } \lambda \text{keys} \rightarrow$  pick new hash functions

# Benchmarks

	<b>64-bit integer keys (<math>n = 10^7</math>)</b>		
	<i>Construction</i>	<i>Lookup</i>	<i>Membership</i>
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.

## **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`