Functional Hash Maps in a Data Parallel Language

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

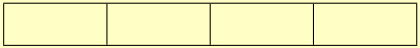
[1]Department of Computer Science, University of Copenhagen

August 22nd, 2025

Contact: widu@di.ku.dk

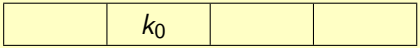
Open Adressing Example

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



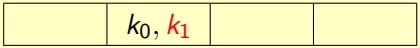
Open Adressing Example

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



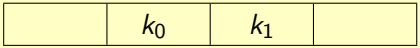
Open Adressing Example

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



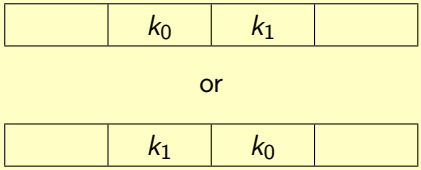
Open Adressing Example

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



Core Ideas

- Concurrency.
- Collision resolution.



Hash Maps in Functional Array Language

- Avoid collisions.
- Bulk operations.

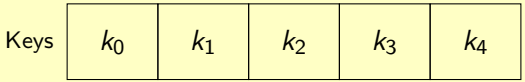
$$\texttt{map} : (\alpha \to \beta) \to [n]\alpha \to [n]\beta$$
$$\texttt{from\_array} : [n](\alpha, \beta) \to \textbf{map}\ \alpha\ \beta$$

Hash Maps in Functional Array Language

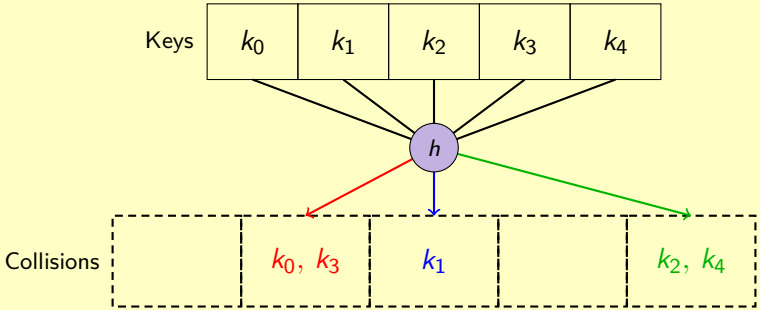
- Avoid collisions.
- Bulk operations.
- Fredman-Komlós-Szemerédi (FKS) construction.

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

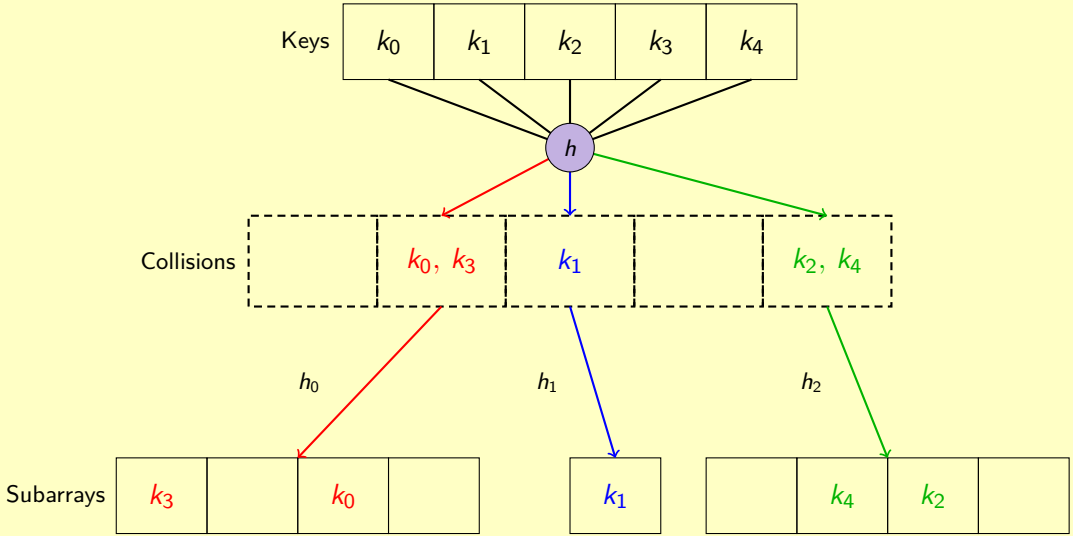
Perfect Hashing with FKS



Perfect Hashing with FKS



Perfect Hashing with FKS



Mapping FKS to GPU

- Irregular parallism.

Mapping FKS to GPU

- Irregular parallism.
- Map flattening:

$$\texttt{map}\ (\texttt{map}\ f)\ [[1, 2], [3, 4, 5]]$$

Mapping FKS to GPU

- Irregular parallism.
- Map flattening:

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

Mapping FKS to GPU

- Irregular parallism.
- Map flattening:

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

Mapping FKS to GPU

- Irregular parallism.
- Map flattening:

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

- Flattening the finding of collision-free hash functions.

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

  `while` h leads to collisions `do`

    Pick a random hash function h

Mapping FKS to GPU

- Irregular parallism.
- Map flattening:

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

- Flattening the finding of collision-free hash functions.

$\texttt{map}\ \lambda \mathit{subkeys} \rightarrow$
  $\texttt{while}\ h$ leads to collisions do
    Pick a random hash function h

$\mapsto$

$\texttt{while}$ any collisions in subarrays do
  $\texttt{map}\ \lambda \mathit{keys} \rightarrow$ pick new hash functions

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.

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