Introduction to a Wait-Free Hash Table

Steven Feldman (UCF), Pierre LaBorde (UCF), Damian Dechev (Sandia, UCF)

Hash Table

- Key-Value pair stores and retrieves data
- Three Core Functions:
  - Put, Get, Delete
  - Constant time: O(1)

Lock-Free/Non-Blocking: ensure that no single thread can prevent another from accessing information.

Wait-Free: Guarantees that any process can complete any operation in a finite number of steps, regardless of the execution speeds of the other processes.

Design Goals:

- Lock-Free and Wait-Free Nature
  - All threads make progress
- Concurrent/Efficient Resize
- Thread-Death Safety
- No CAS loops

Features:

- Probing and non-probing versions
- No global resize
  - Concurrent expansion
- Predictable memory usage
  - Lookahead
- Each key has a unique final position

Brief Algorithm Description

Two types of Nodes:

- Spine Node: State, Memory Array
- Data Node: State, Key, Value

  Key treated a bit string

  Divided into blocks

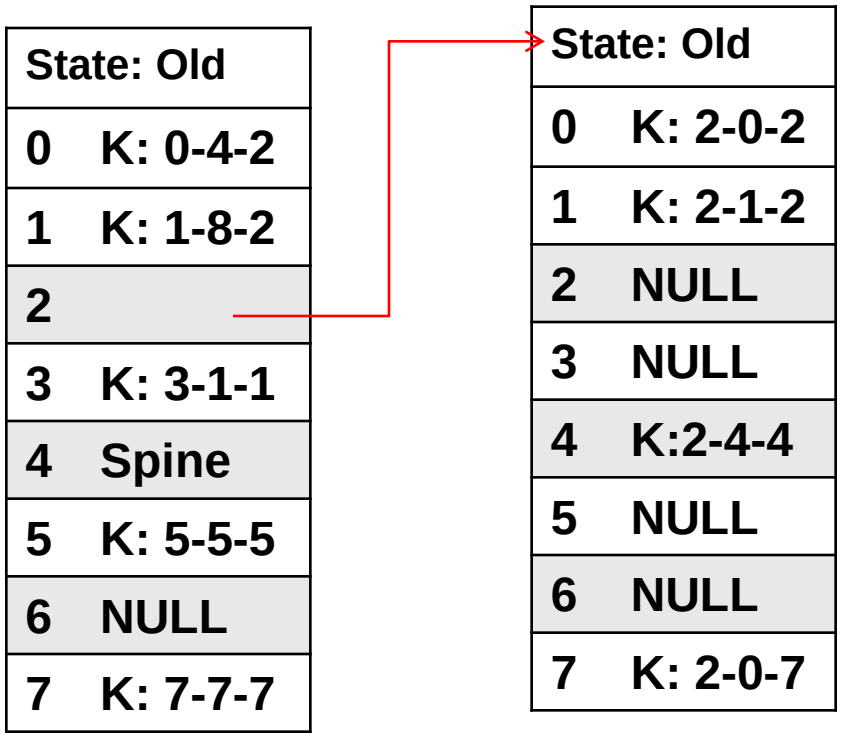
  Each block determines the position the key belongs in at each level.

Traversal Procedure:

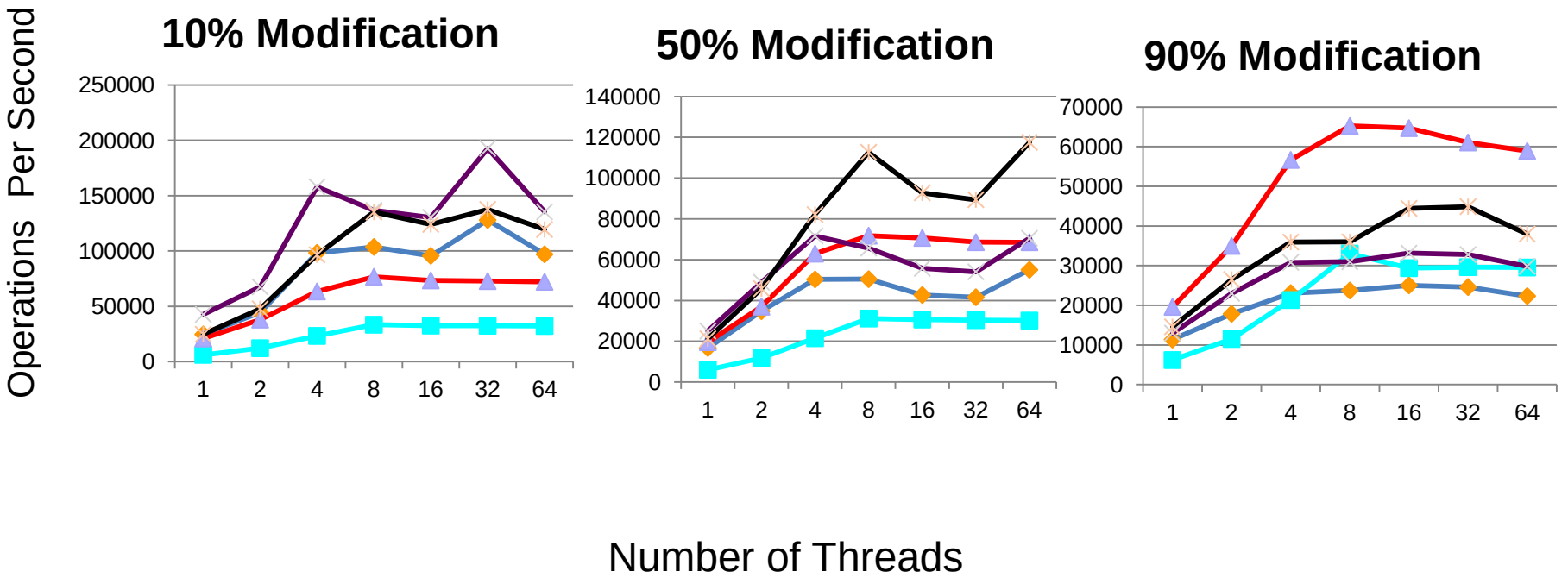
1) Bits in key are rearranged
2) First X bits determine the location
3) Examine the node
   A. If Spine Node:
      Using the next X bits, examine that spine's memory array
   B. If Data Node:
      Complete the thread's operation

Composed of Spine Nodes

- A position points to a spine or data node
- Max depth is X, where 2^X=Spine Length



Performance Comparison



Performance Analysis:

- **Wait-Free performs better at Modification > 75%**
- **Algorithm still not optimized**
- **Probing version not worth it.**

Legend:
Wait-Free Hash Table (Non-Probing)
Wait-Free Hash Table (Probing)
TBB Concurrent Hashmap
CDS Split List Hash Map (Lock-Free)
Michaels Lock-Free Hash Map

Conclusion

Future Work:

- Additional Functions
  - Put-if-absent
  - Put-if-equals
  - Delete-if-equals
- Memory pool
- Optimization of code
- Integration in real-world applications

Acknowledgements:

Funding provided by UCF, NSF, and UCF's YES program

Source Code: github.com/MrStevenFeldman/

Example

- Finding Key: 0-4-2
- 2-1-2 Hashes to 2-1-2
- Examines position 2
- Examines position1
- Returns associated value

- Inserting Key: 1-2-3
- 1-2-3 Hashes to 5-6-1
- Position 5 is occupied
  - Adds and expands the table