PMap: A Non-volatile Lock-free Hash Map with Open Addressing

Kenneth Lamar (UCF)

Christina Peterson (UCF)

Damian Dechev (UCF)

Roger Pearce (LLNL)

Keita Iwabuchi (LLNL)

Peter Pirkelbauer (LLNL)

Setting

- Intel Optane DC
  - Newly available
  - High capacity
  - Persistence
- Graph analytics
  - Billions of vertices
- Concurrent data structures
  - High performance access to data in shared memory
- **Hash maps**
  - Fundamental data structure
  - Commonly used in graph analytics
  - Few high-performance NVM options

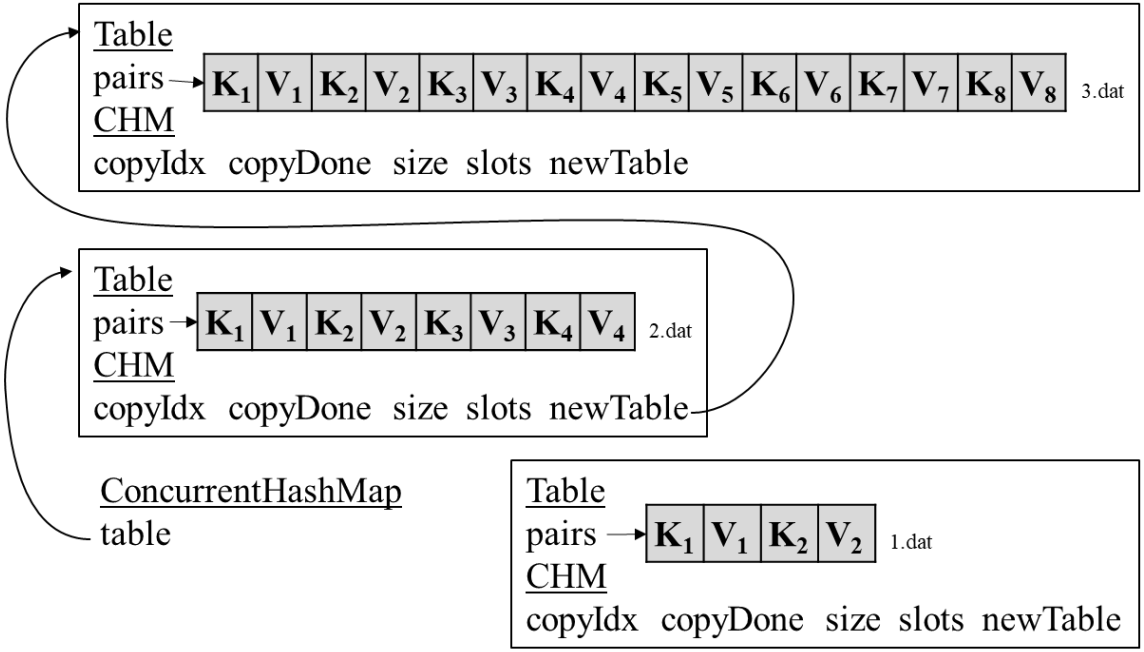
Design Goals

- Read optimized
  - Persistence need no flush or fence after first read
- Runtime over recovery
  - Persist as little as possible
- Compact representation and few cache misses
  - Arrays
  - Open addressing
- Low memory management overhead
  - Allocate large table chunks

Solution

- PMap (**P**ersistent concurrent hash **Map**)
  - Non-volatile
  - Lock-free
    - Guaranteed system-wide progress
    - Scales up with multiple threads
  - Open addressing
    - In-place keys and values
  - Resizable
    - Shrink or expand

Data Layout



Legend:

Class names

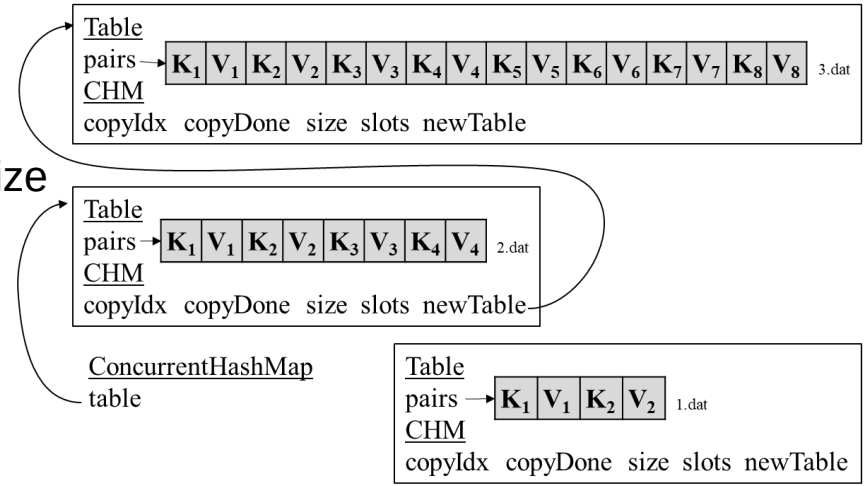
Persistent data

Operations

- `insert()`
  - Find a free slot
  - Insert if the key does not yet exist
- `replace()`
  - Find the key
  - Replace the value
- `remove()`
  - Find the key
  - Replace the value with a tombstone
  - Note: Key claims this slot permanently
- `update()`
  - Find the key
  - Perform CAS on value
    - Replacement value dependent on old value
    - Ex. Counter increments by 1 atomically

Resizing

- Adapted from Cliff Click's hash map
- Lock-free resizing is challenging
  - Keys and values are separate atomics
    - Partial operations are possible
- Allocate a table twice (or half) the current table size
- Key-value pairs are individually migrated
  - Concurrent
  - Parallel
  - Incremental
- Reserves a resize bit for each key and value
  - Indicates migration in progress
    - Thread must help migrate before returning value
  - Resize bit cuts into usable bits
    - Limits keys and values to 63 bits each
- Once migrated, the old slot is replaced with a migration sentinel
  - Slot cannot be reused
  - Migration is complete when all slots have migration sentinels

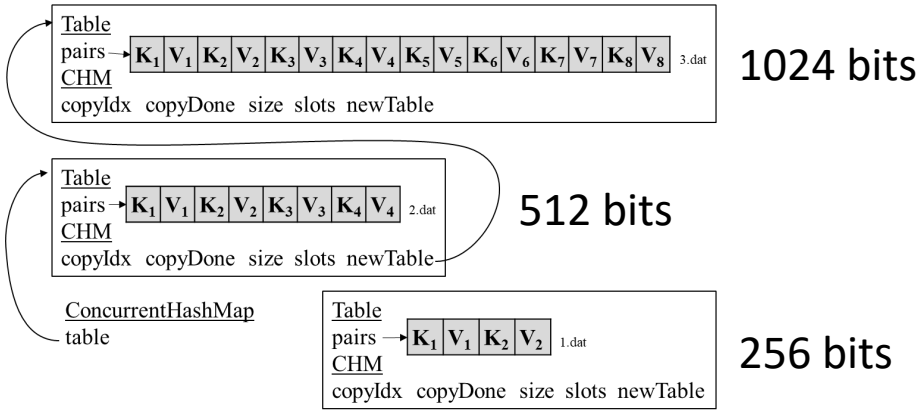


Persistence

- Adaptation of link-and-persist
- Adds memory consistency after crash
- Minimizes overhead for read-heavy workloads
- Minimal changes required
- Cost: 1 bit per variable
  - Effective limit: 62-bit variables
  - Reasonable tradeoff for our use cases

Recovery

- Only persist keys and values
- Filesystem data (name, size) infers contents
- Link-and-persist ensures data is consistent
  - Orphans are possible but discarded during recovery



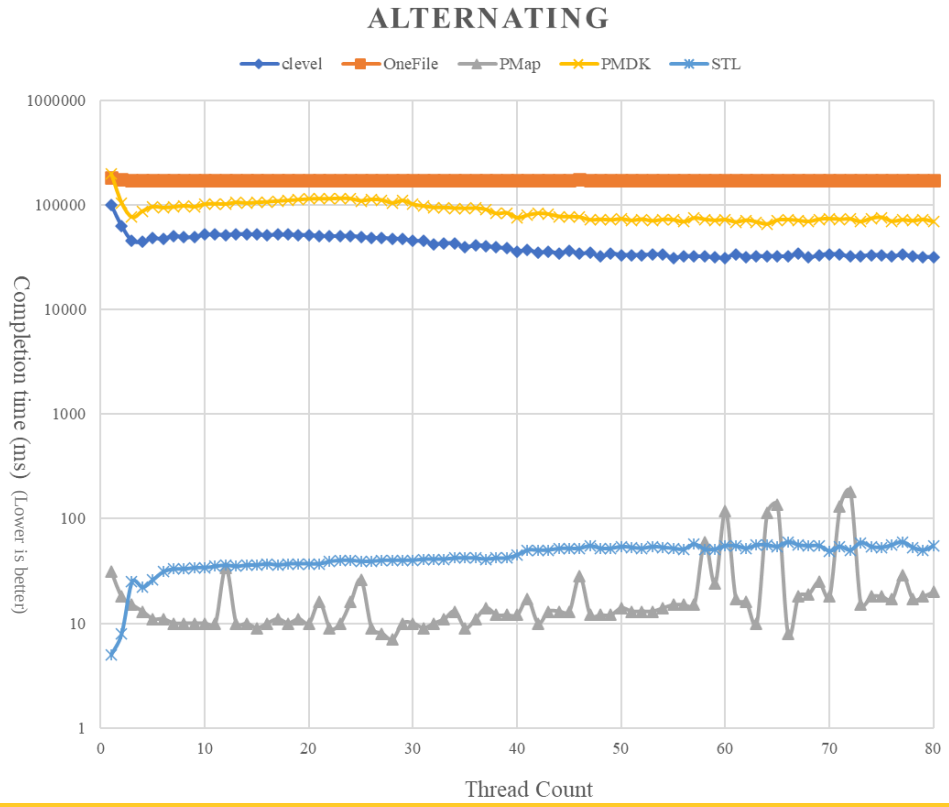
Related Works Compared

- Concurrent level hashing (clevel)
  - Lock-free
  - Open addressing (of pointers)
  - Resize (but only expansion)
- OneFile hash map (OneFile)
  - Wait-free
  - Transactional
  - Node-based
- Standard Template Library hash map (STL)
  - Volatile
  - std::map with global lock
- Persistent Memory Development Kit concurrent_hash_map (PMDK)
  - Based on Intel TBB
  - Reader-writer locks

Testing Environment

- System:
  - 2x 20 core / 40 thread Intel Xeon Gold 6230
  - 134GB DRAM, 248GB Optane DC
- Optane Configuration:
  - App Direct mode
  - Filesystem-DAX (fsdax) mode
- Code Configuration:
  - C++
  - GCC (-O3 -march=native -flto)
- Test Configuration:
  - 62-bit keys and values
  - Tables initially 2^{14}
  - No garbage collection

Performance Comparisons



Limitations and Future Work

- Limited to 62-bit keys and 62-bit values
- Linear probing
  - Simple
  - High load factor means poor performance
  - Alternatives:
    - Cuckoo hashing
    - Hopscotch hashing
    - Multiple hash functions

Conclusions

- PMap
- Outperforms state-of-the-art alternatives
- Useful guarantees
  - Non-volatile
  - Scalable
  - Lock-free
  - Resizable
  - Open addressing

Speaker's
Webcam view

Thank You

Contact:
kenneth@knights.ucf.edu

Code:
https://github.com/ucf-cs/PMap

UCF