Split-Ordered Lists: Lock-Free Extensible Hash Tables

Gunnar Johansson
Paper by Ori Shalev, Nir Shavit
(Tel-Aviv University)

Presentation outline

- Motivation
- The lock-free idea (basics)
- Hash tables: Brief review
- Solution
- Results
- Conclusions

Motivation

- Previous work shows that lock-free approaches are optimal for high concurrency
- Many applications use hash tables
- Requirements
  - Support for high concurrency – many parallel threads
  - Extensibility – the table should be able to grow

Lock-free parallelism

- For high concurrency, locks become large bottlenecks!
- Example: shared counter

```
shared int counter;
shared Mutex lock;
…
acquire_lock(lock);
counter++;
release_lock(lock);
…
```

Lock-free parallelism

- Basic problem: cannot perform counter++ atomically
- Use compare-and-swap (CAS), supported in hardware!

```
bool CAS(int * p, int old, int new) {
  atomic {
    if (*p == old) {
      *p = new;
      return true;
    }
    return false;
  }
}
```

[Datormagazin, no 7/2006]

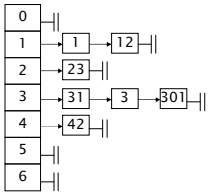
Lock-free parallelism

- Now, counter++ can be implemented using a fetch-and-add (FAA) operation

```
void FAA(int * p, int value) {
  int old;
  do {
    old = *p;
  } while (!CAS(p, old, old+value));
}
```

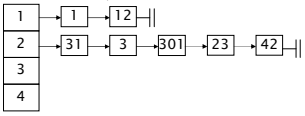
Hash tables: brief review

- Insert, delete, find in O(1)
  - Assumed a good, balanced hash-function



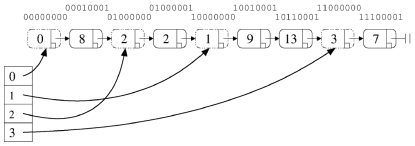
Hash tables: brief review

- Rehashing, extensibility
  - When load increases, O(1) cannot be maintained
  - Solution
    - Increase array size
    - Rehash items (redistribute items among buckets)



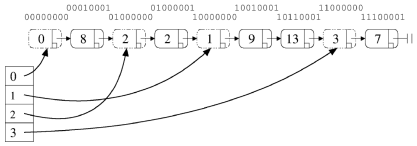
Solution

- Rehashing expensive, breaks concurrency
- Instead of “moving items among buckets”, let’s “move buckets among items”



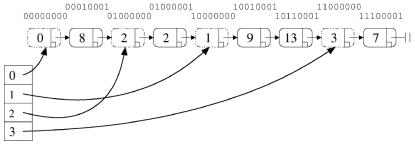
Solution

- Keep single list, let buckets provide shortcuts into the list
- Extending bucket array should not require changing list
- Sort order: recursive split-order !



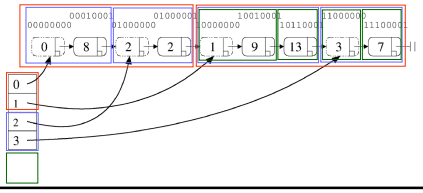
Solution

- Hash function: modulo 2^i
- Array size: power of two (doubles at each extension)



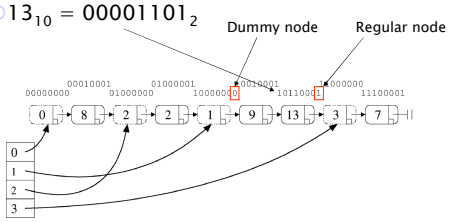
Solution

- Hash function 2^2
  - $13_{10} = 00001101_2$
  - $9_{10} = 00001001_2$



Solution

- Sorting: recursive split-order?
  - Binary reversal
  - $13_{10} = 00001101_2$



Solution

- Lock-free
  - Based on previously known CAS lock-free list
- Benefits
  - No reordering of the list items
  - Items can be reached at all "hashing recursion levels" (parallel threads could operate on different levels)

Results

- Compared with lock-based resizable hash table [Lea, 2003]

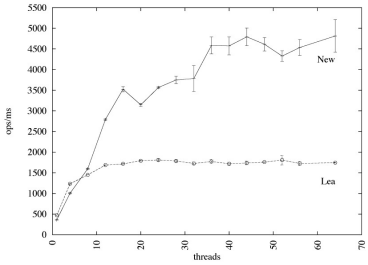
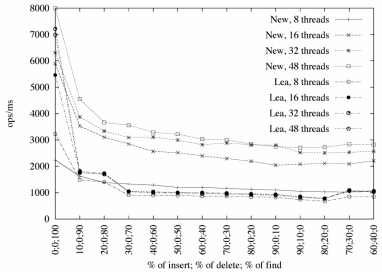


FIG. 9. Throughput of both algorithms. Standard deviation is denoted by vertical bars.

FIG. 10. Varying operation distribution.



Conclusions

- Hash table that requires no redistribution of items when extended
- Offers significantly better performance than lock-based alternatives when concurrency is high
- Split-ordering can possibly also improve sequential implementations