FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

Parallel Computing

k-LSM: A Relaxed Lock-Free Priority Queue

Masterstudium:
Software Engineering & Internet Computing

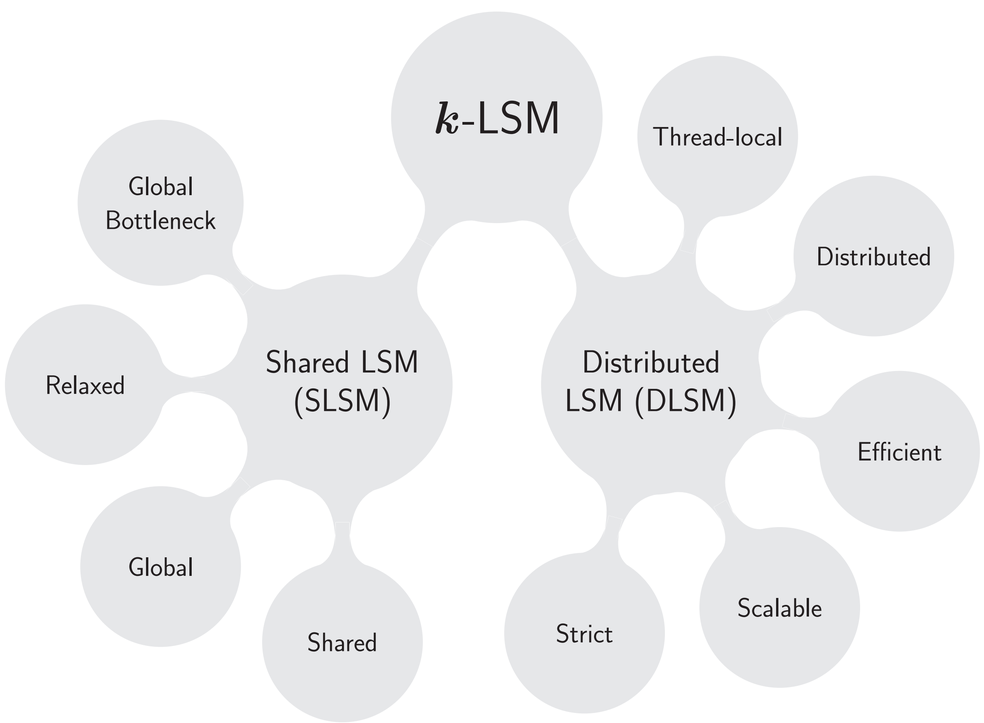
Jakob Gruber

Technische Universität Wien
Institut für Informationssysteme
Arbeitsbereich: Parallel Computing
Betreuer: Prof. Dr. Scient. Jesper Larsson Träff

Concurrent Priority Queues

- **Priority queues**: Abstract data structures for storing key/value pairs; efficient access to the item with the minimal (maximal) key.
- **Concurrent priority queues**: From early locking heaps, through lock-free SkipList-based designs, to current research into relaxed priority queues.
- **Lock-freedom**: Strong progress condition that can be efficient in practice.
- **Linearizability**: Each operation appears to take effect at some instant in time within the operation invocation.
- **Relaxed**: Weaken semantic guarantees in return for higher efficiency and/or scalability; useful for concurrent priority queues to alleviate the global bottleneck at the minimal (maximal) element.

The k-LSM Concurrent Priority Queue



The k-LSM (Log-structured Merge Tree) is a **relaxed**, **linearizable** and **lock-free** priority queue design by Wimmer, composed of two complementary priority queues; the SLSM, a centralized queue with strong guarantees but limited performance; and the DLSM, a distributed design without global guarantees but very high throughput.

- **Idea**: An LSM maintains items in a logarithmic number of ordered arrays to achieve amortized logarithmic times for deletions and insertions.
- **Relaxation**: Deletions may return one of the kP minimal items, where k is a configuration parameter and P is the number of threads.
- **Cache-efficient**: Usage of arrays and the standard merge algorithm result in good cache locality properties and high performance.

Publications

- Martin Wimmer, Jakob Gruber, Jesper Larsson Träff, and Philippas Tsigas. "The lock-free k-LSM relaxed priority queue". In: *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM. 2015, pp. 277–278.

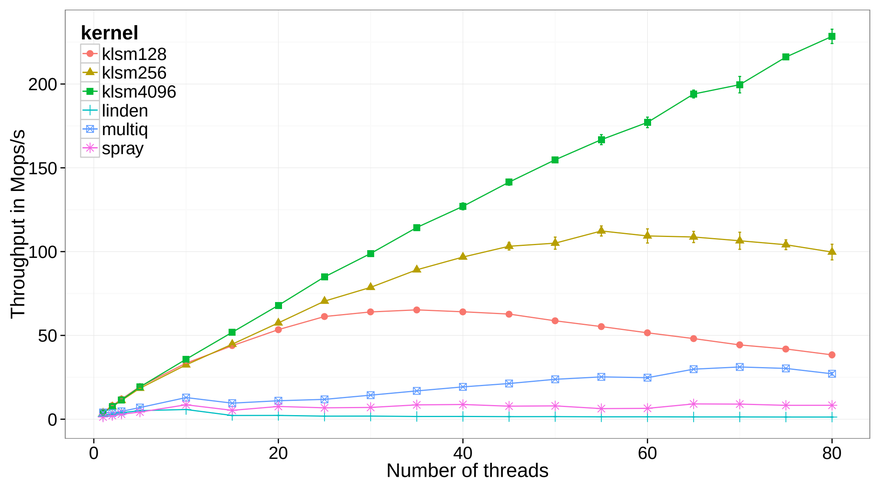
Our Implementation

We improve on the k-LSM by providing a new, efficient, standalone implementation in addition to the previously available version as part of the task-scheduling framework *Pheet*. Our implementation is:

- **Scalable**: Under ideal circumstances and with sufficient relaxation, the k-LSM scales until the maximal thread-count on all tested machines.
- **Performant**: Outperforms the best other concurrent priority queues by up to a factor of ten.
- **Maintainable**: Uses classical software engineering concepts such as composability, readability, and separation of concerns.
- **Portable**: Implemented using C++11 atomics without reliance on external libraries.

Evaluation & Results

- **Algorithms**: Evaluated against other state of the art designs, including the SprayList (relaxed, lock-free, probabilistic guarantees), the Lindén queue (strict, lock-free, linearizable), and Multiqueues (relaxed, lock-based, no guarantees).
- **Benchmarks**: Evaluation under a variety of scenarios, including balanced and unbalanced workloads combined with different key generation algorithms. We demonstrate that these parameters have a high impact on the performance of some data structures.
- **Throughput benchmarks** measure performance in millions of operations per second. Results are varied: in evenly distributed settings, the k-LSM outperforms all other data structures by a factor of 10, while in others, performance suffers since more stress is placed on the centralized SLSM.



- **Quality benchmarks** measure ranks of returned items, e.g., for each returned item, how many other items within the priority queue were skipped when determining it as the minimal item. The k-LSM more than fulfills its semantic guarantees, usually delivering results within the best fifth percentile.

Rank Error	20 threads		40 threads		80 threads	
(in # items)	Mean	StdDev	Mean	StdDev	Mean	StdDev
`globallock`	1.9	1.0	2.9	1.7	4.8	2.9
`klsm128`	33	31	55	46	430	294
`klsm256`	42	42	71	61	750	828
`klsm4096`	297	496	625	1014	10353	12667
`multiq`	984	2899	2252	7433	3787	12549

Kontakt: gruber@par.tuwien.ac.at