Karlsruhe Institute of Technology

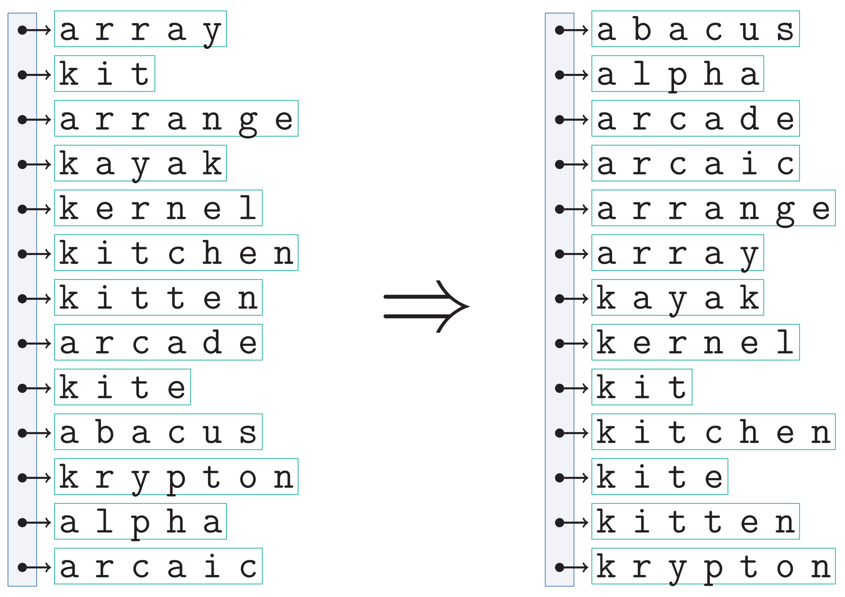
Institute of Theoretical Informatics
Algorithmics

Parallel Super Scalar String Sample Sort

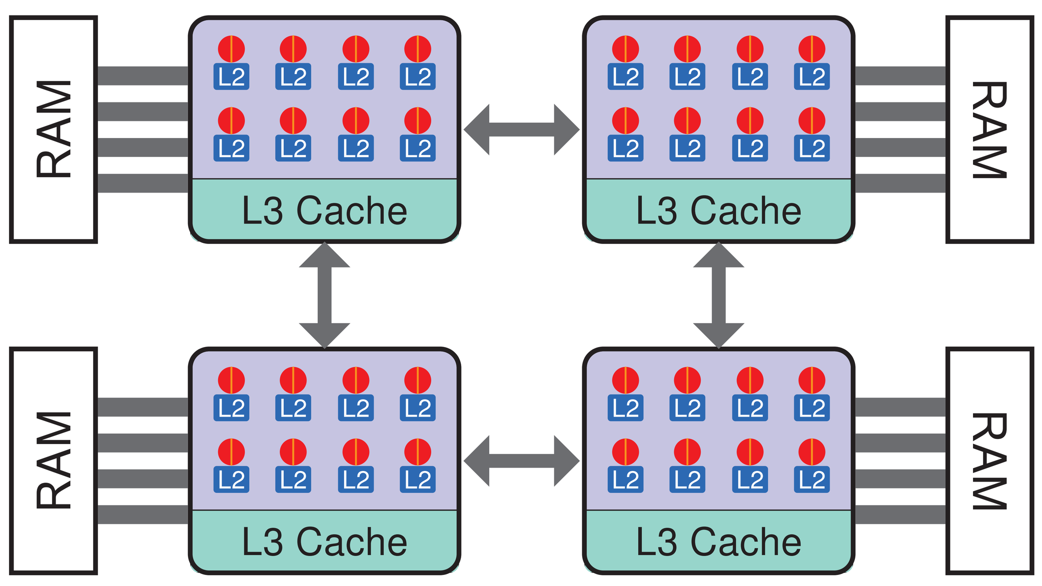
Timo Bingmann and Peter Sanders

1. Sorting Strings

Sorting strings is one of the basic operations needed for text indexing, MapReduce, in databases, and many other applications. It requires to order a set of strings lexicographically like in a dictionary.



Remarkably, no publication about practical parallel string sorting existed. We focus on sorting large sets of strings with modern multi-core architectures and how to best utilize **shared-memory parallelism** on these machines.

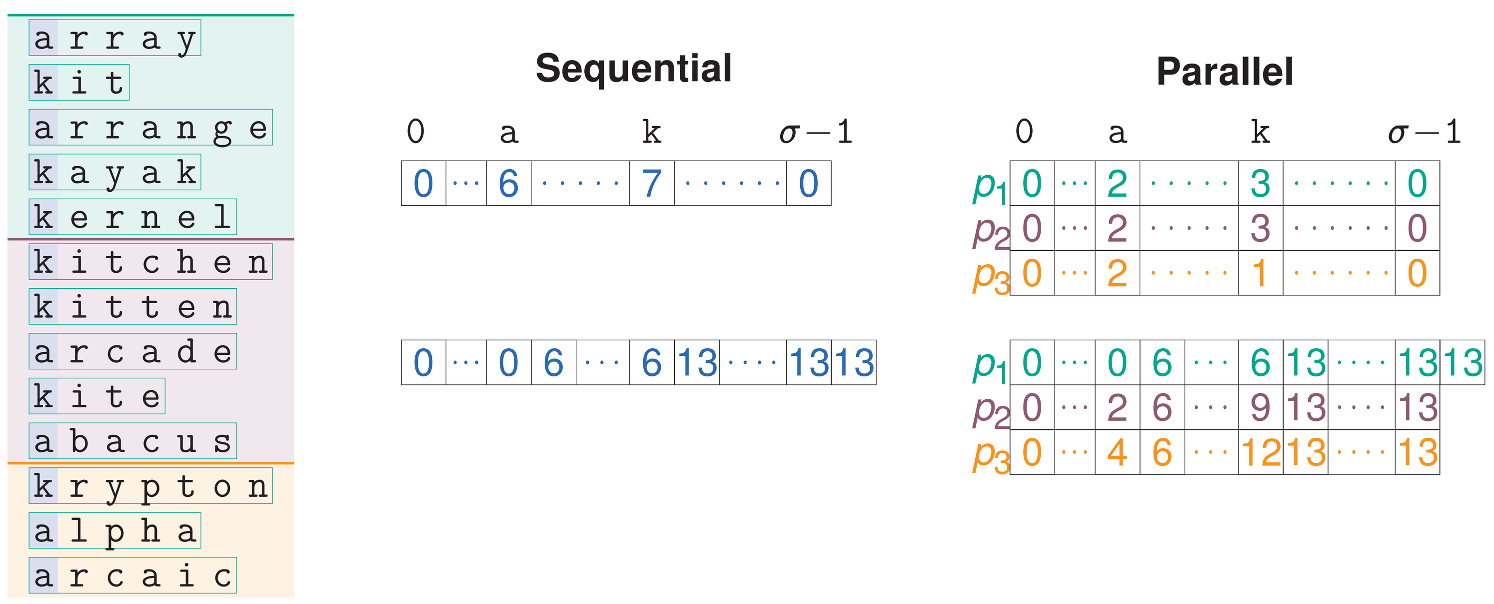


2. Parallelization of Radix Sort and Multikey Quicksort

Of existing sequential string sorting algorithms, we parallelized two promising candidates: radix sort [5, 4] and multikey quicksort [1].

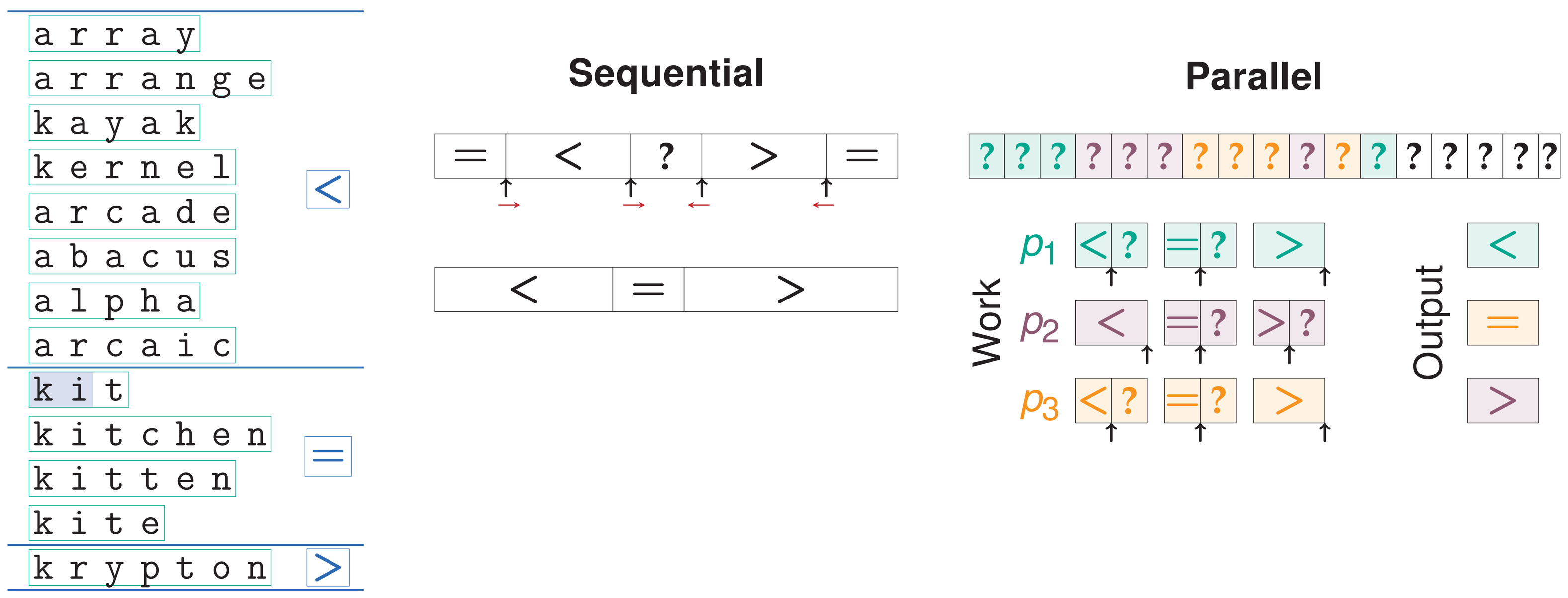
2.1 Radix Sort

Radix sort [5, 4] considers only a single character at a time, counts the number of occurrences and calculates a prefix sum, which yields the boundaries for rearranging strings for deeper sorting. The algorithm is easy to parallelize, but uses only one byte of the cache line retrieved per random access to characters.



2.2 Multikey Quicksort

Multikey quicksort [1] partitions the string set into three parts according to the next w characters of a selected pivot string. A variant of this algorithm by Tommi Rantala [6] using $w = 8$ and caching of characters is generally the **fastest sequential algorithm**. To parallelize it, we extended from a well-known blocking scheme used for parallel quicksort.

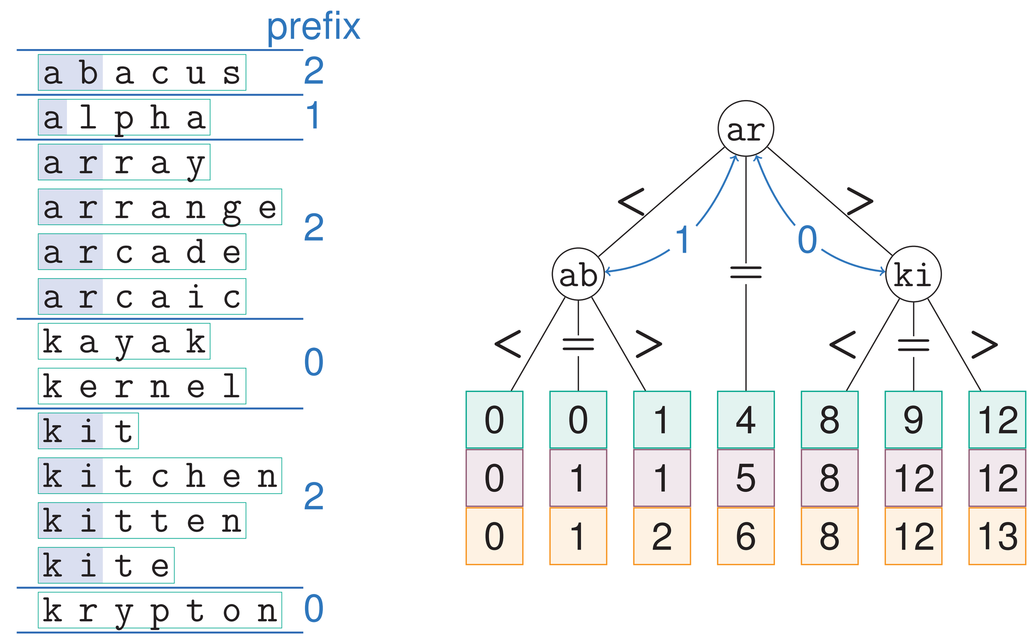


2.3 Parallelization Toolkit

Beyond parallelizing the basic sequential algorithms' cores we also developed a load balancing framework to efficiently process recursive sorting steps. It uses a **voluntary work sharing** method that avoids many costly atomic operations and synchronizations.

3. Super Scalar String Sample Sort

With Super Scalar String Sample Sort (S^5) [2, 3] we generalize both from multikey quicksort and from (integer) Super Scalar Sample Sort [7] by using **multiple pivots**. The $v = 2^d - 1$ pivots are organized into a perfect binary search tree, which is used to classify all strings into $2v + 1$ buckets using ternary comparisons. Buckets contain strings with equal prefixes, either w or the longest common prefix of consecutive pivots, and are recursively sorted deeper.

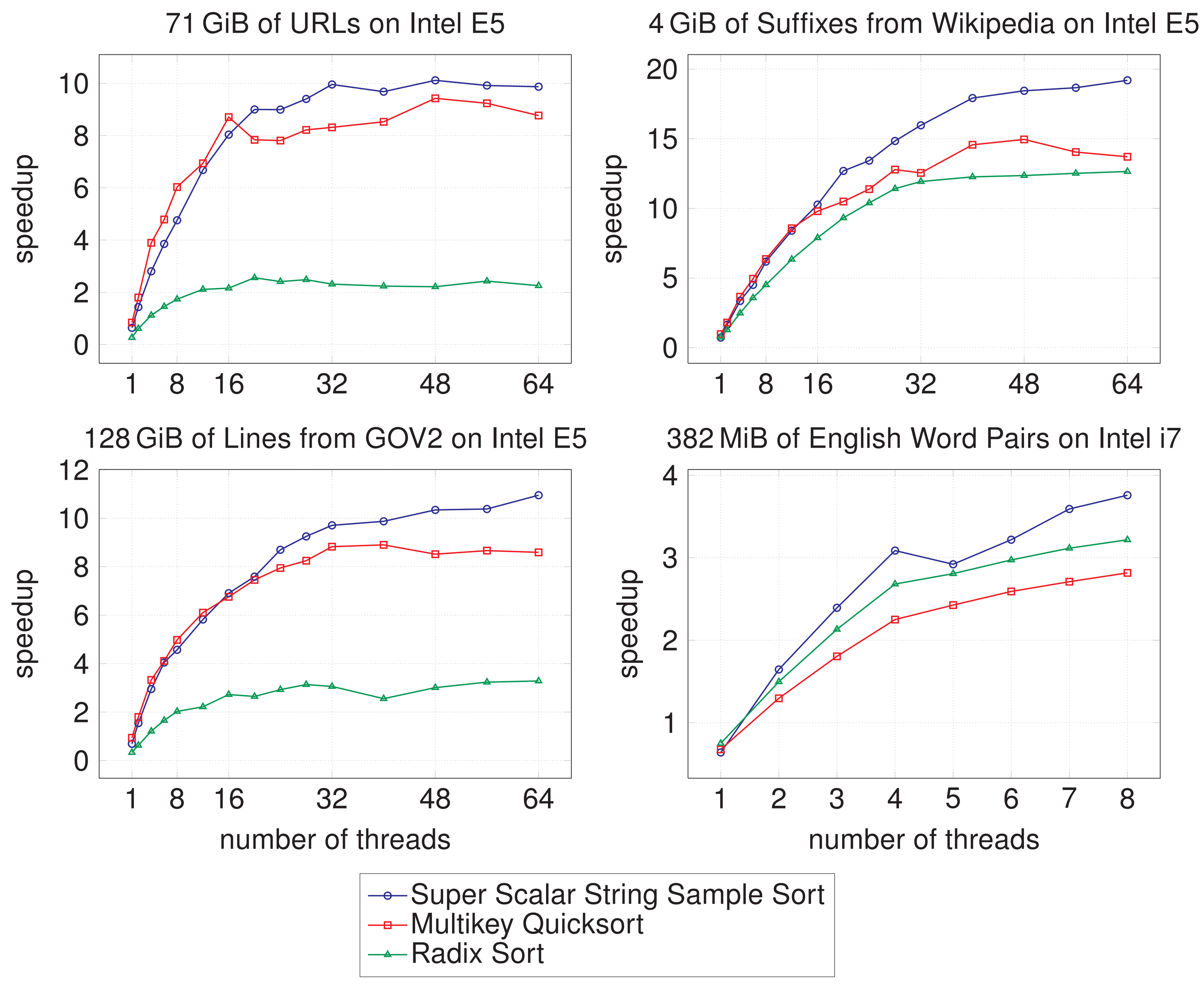


3.1 Parallelization and Engineering Aspects

- Due to the independent classification of strings using the search tree, parallelization of S^5 is straightforward. Depending on the size of the string set, different sub-algorithms are used.
- We use $w = 8$ and adapt the search tree size to fit into L2 cache.
- The binary search tree is represented implicitly in an array and we use **predicated instructions** to traverse it, thus avoiding branch mispredictions.
- Instead of equal comparisons at each node, we compare only after a **full descent** of the tree. This enables us to **interleave** the classification of multiple strings, allowing the processor to use several super scalar processing units in parallel.

4. Experimental Results

We implemented all three algorithms in C++ and show experimental results from only two platforms here (see [3] for more): a 32-core, 32-HTcore **Intel E5**-4640 machine with 2.4 GHz and 512 GiB RAM, and a 4-core, 4-HTcore **Intel i7**-920 with 2.67 GHz and 12 GiB RAM.



On all inputs, Parallel Super Scalar String Sample Sort achieves the highest speedups and is, overall, currently the best parallel string sorting implementation on these platforms.

References

[1] Jon L. Bentley and Robert Sedgewick. "Fast algorithms for sorting and searching strings". In: *SODA*. 1997.
[2] Timo Bingmann and Peter Sanders. "Parallel String Sample Sort". In: *ESA*. LNCS 8125. 2013.
[3] Timo Bingmann and Peter Sanders. *Parallel String Sample Sort*. see ArXiv e-print arXiv:1305.1157. May 2013.
[4] Juha Kärkkäinen and Tommi Rantala. "Engineering Radix Sort for Strings". In: *SPIRE*. LNCS 5280. 2009.
[5] P. M. McIlroy, K. Bostic, and M. D. McIlroy. "Engineering radix sort". In: *Computing Systems* 6.1 (1993).
[6] Tommi Rantala. *Library of String Sorting Algorithms in C++*. http://github.com/rantala/string-sorting. Git repository accessed November 2012. 2007.
[7] Peter Sanders and Sebastian Winkel. "Super Scalar Sample Sort". In: *ESA*. LNCS 3221. 2004.