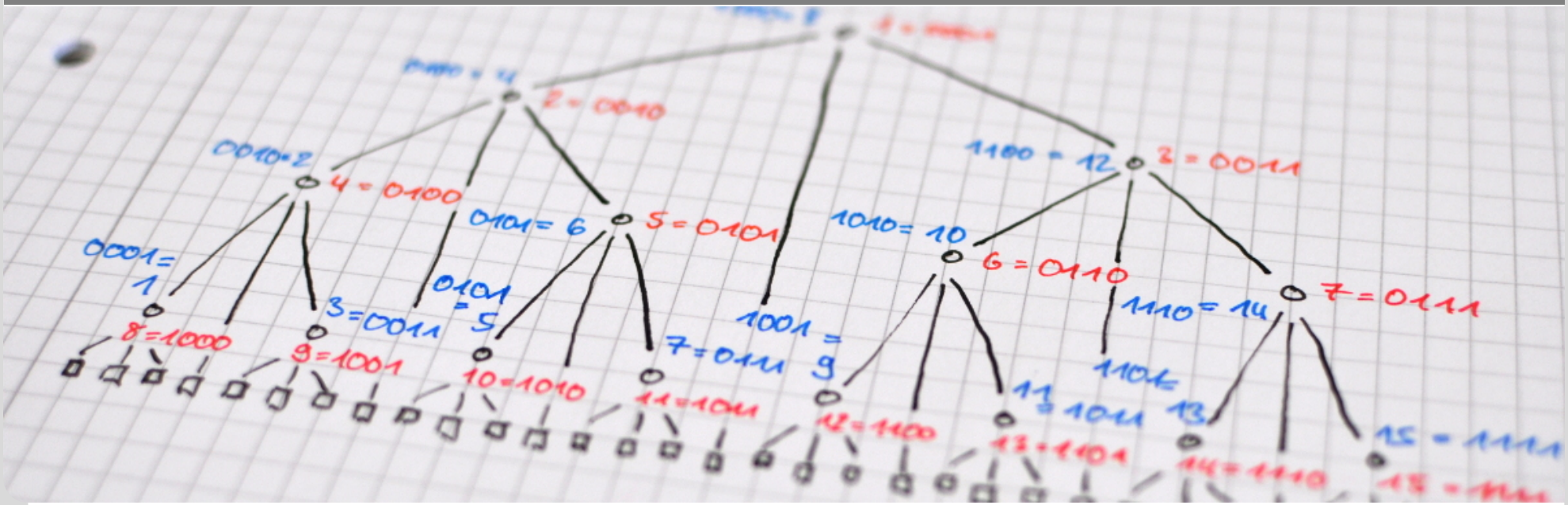
Karlsruhe Institute of Technology

Parallel String Sorting with Super Scalar String Sample Sort

Timo Bingmann and Peter Sanders | September 4th, 2013 @ ESA'13

INSTITUTE OF THEORETICAL INFORMATICS – ALGORITHMICS



Abstract

We present the currently fastest parallel string sorting algorithm for modern multi-core shared memory architectures.

First, we describe the challenges posed by these new architectures, and discuss key points to achieving high performance gains. Then we give an overview of existing sequential and parallel string sorting algorithms and implementations. Thereafter, we continue by developing super scalar string sample sort (S^5), which is easily parallelizable and yields higher parallel speedups than all previously known algorithms.

Overview

String Sorting Algorithms

Theoretical Parallel Algorithms

- “Optimal Parallel String Algorithms: . . .” [Hagerup ’94]

Existing Basic Sequential Algorithms

- Radix Sort [McIlroy et al. ’95]
- Multikey Quicksort [Bentley, Sedgewick ’97]
- Burstsort [Sinha, Zobel ’04]
- LCP-Mergesort [Ng, Kakehi ’08]

Existing Algorithm Library

- by Tommi Rantala (for Radix Sort [Kärkkäinen, Rantala ’09])
  `http://github.com/rantala/string-sorting`

String Sorting Algorithms

Theoretical Parallel Algorithms

- "Optimal Parallel String Algorithms: . . ." [Hagerup '94]

Existing Basic Sequential Algorithms

→ Radix Sort [McIlroy et al. '95]

→ Multikey Quicksort [Bentley, Sedgewick '97]

- Burstsort [Sinha, Zobel '04]
- LCP-Mergesort [Ng, Kakehi '08]

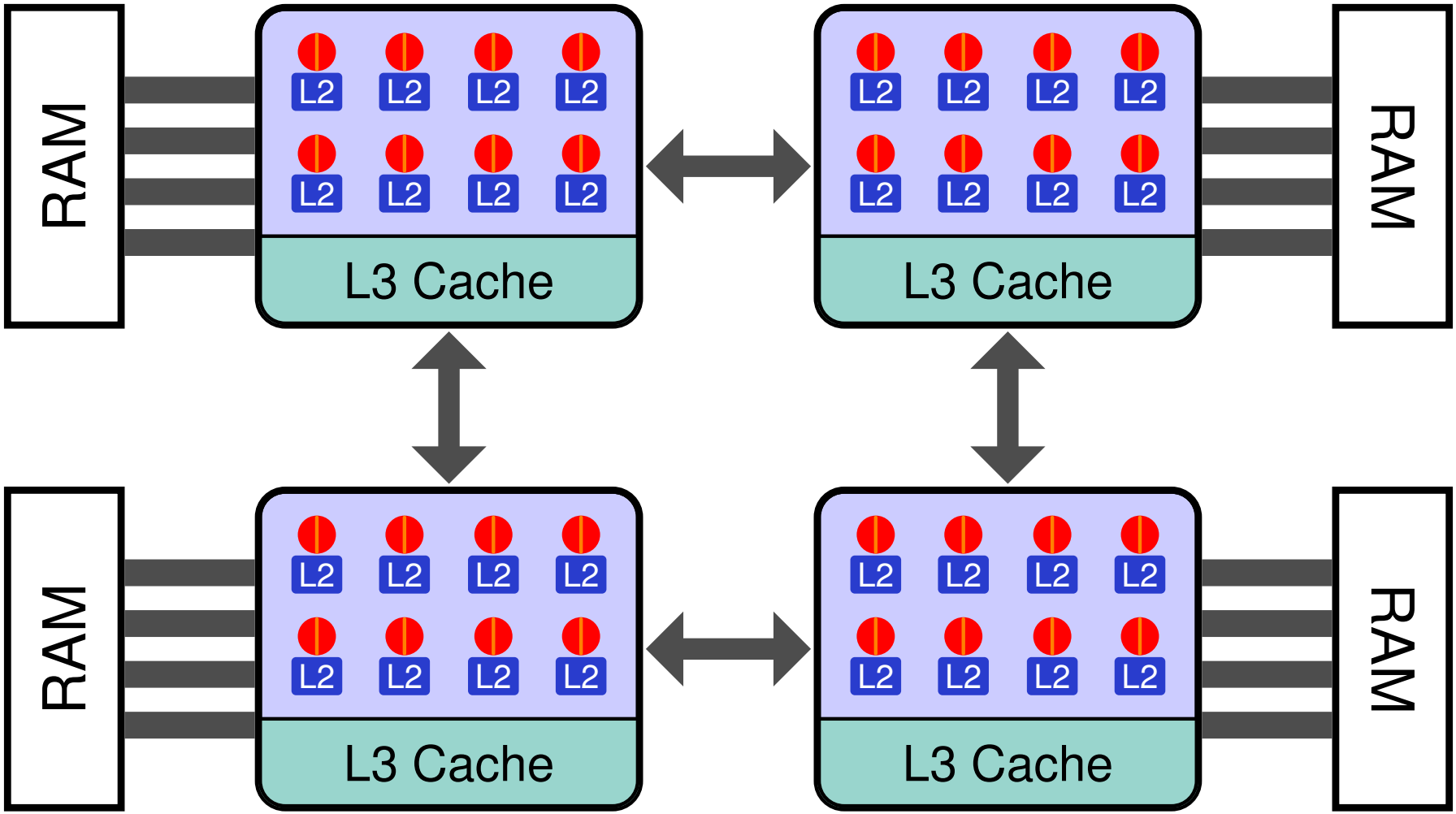
Existing Algorithm Library

- by Tommi Rantala (for Radix Sort [Kärkkäinen, Rantala '09])
  `http://github.com/rantala/string-sorting`

Our Contribution: Practical Parallel Algorithms

→ Parallel Super Scalar String Sample Sort (pS^5) [This Work]

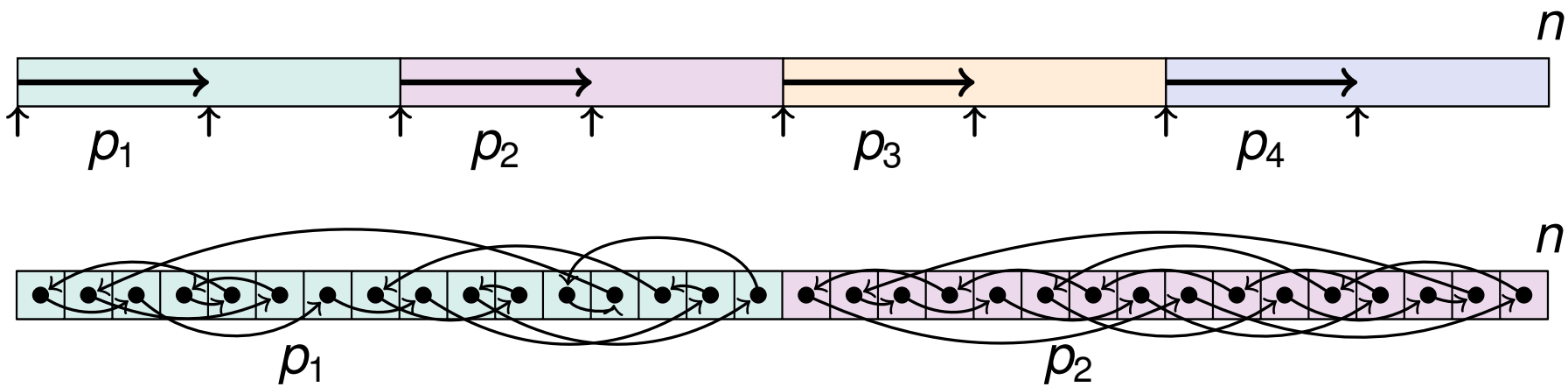
Modern Multi-Core Architecture



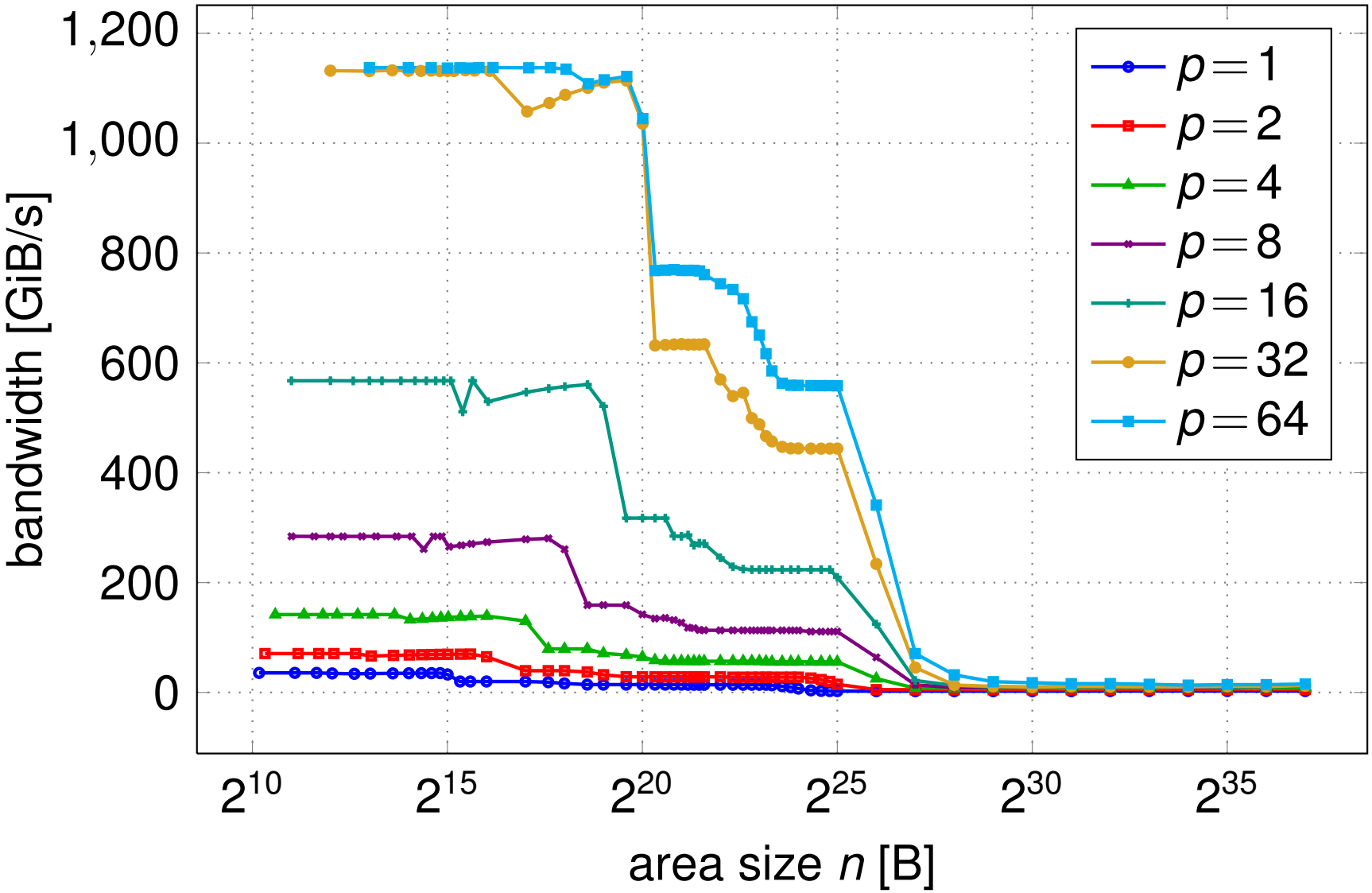
Parallel Memory Bandwidth Test

```
// ScanRead64IndexUnrollLoop
for (size_t i=0; i<n; i+=16) {
    uint64_t x0 = array[i+0];
    // ... 14 times
    uint64_t x15 = array[i+15];
}
```

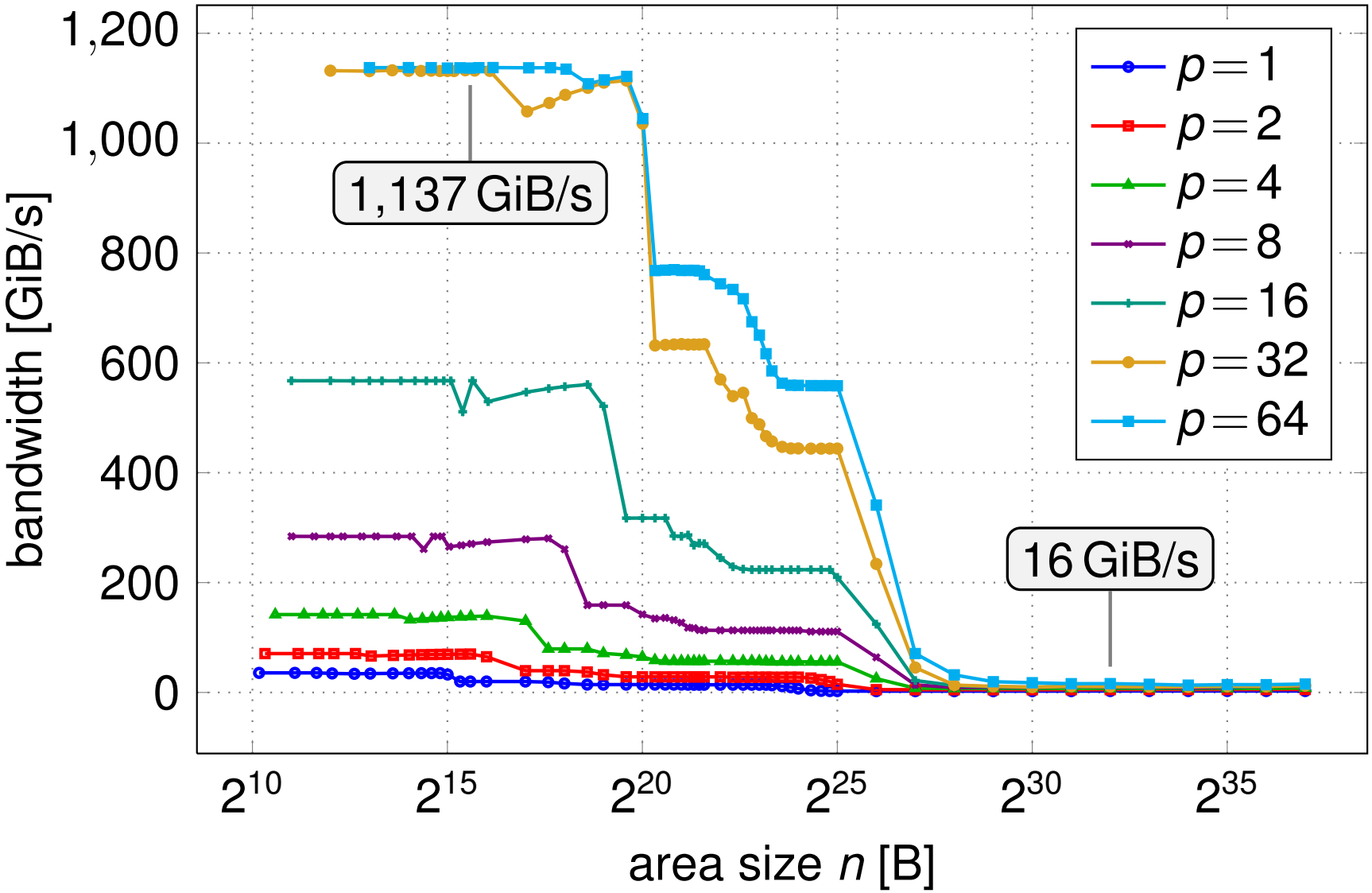
```
// PermRead64SimpleLoop
uint64_t p = *array;
while((uint64_t*)p != array)
    p = *(uint64_t*)p;
```



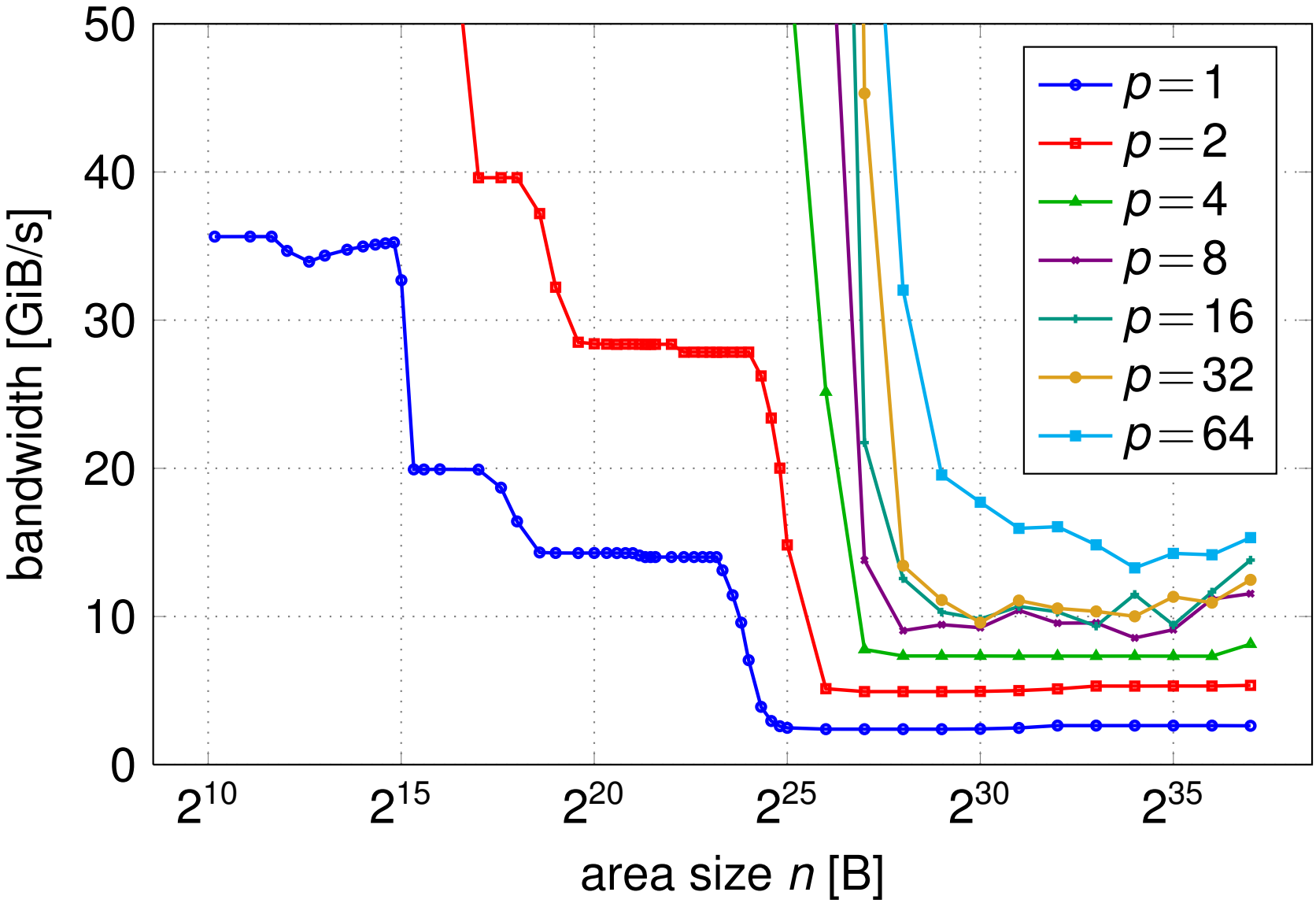
ScanRead64IndexUnrollLoop



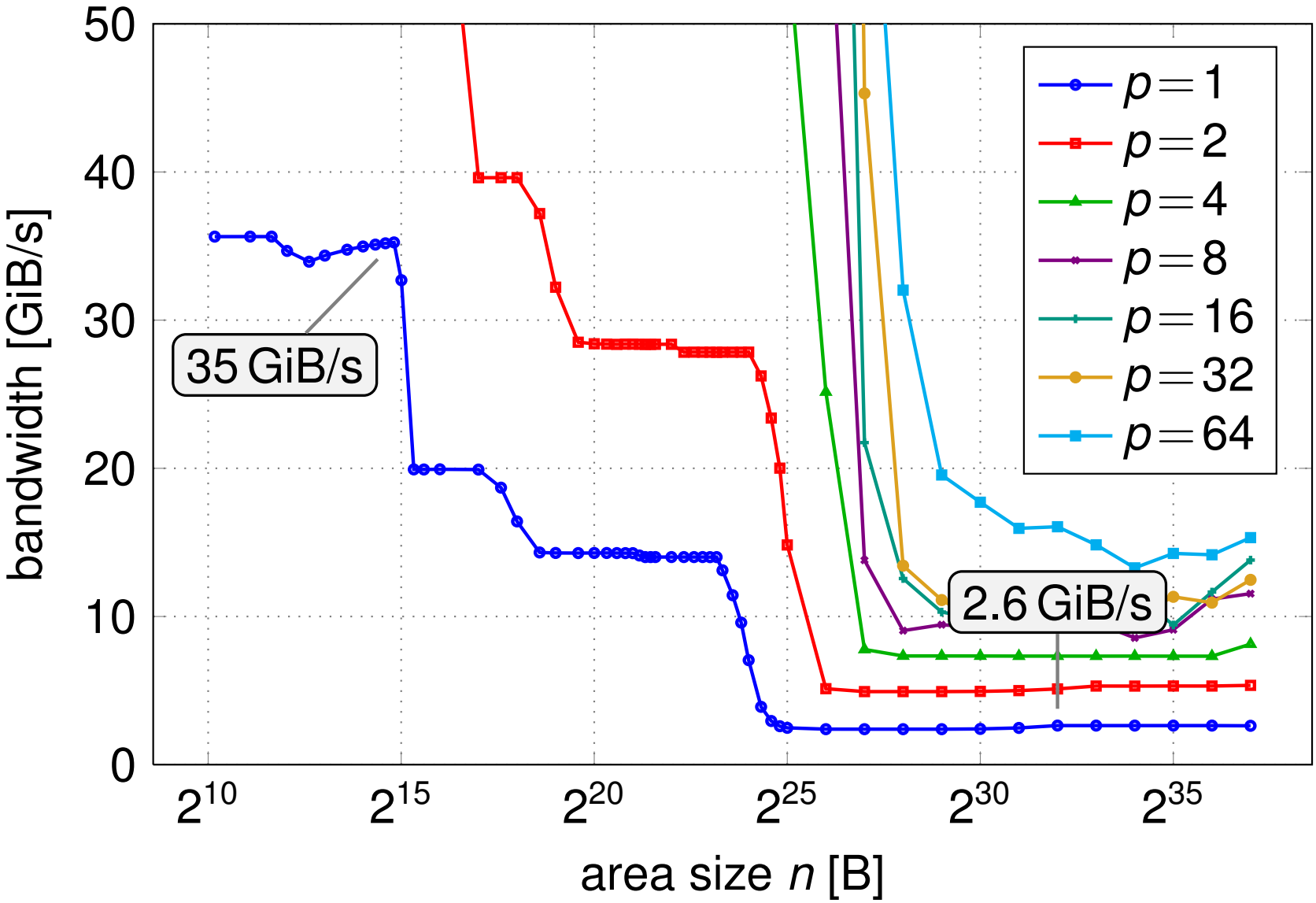
ScanRead64IndexUnrollLoop



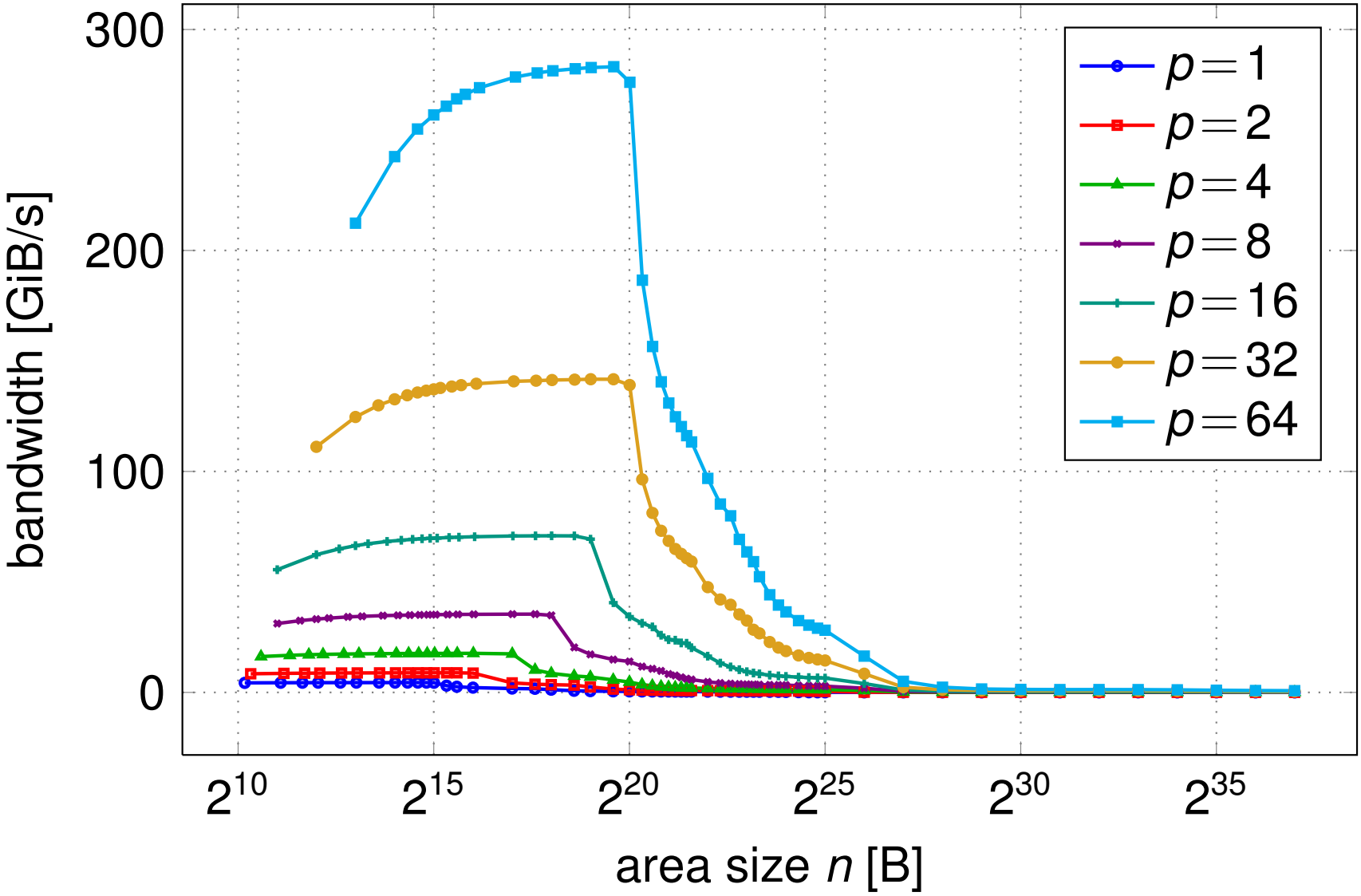
ScanRead64IndexUnrollLoop



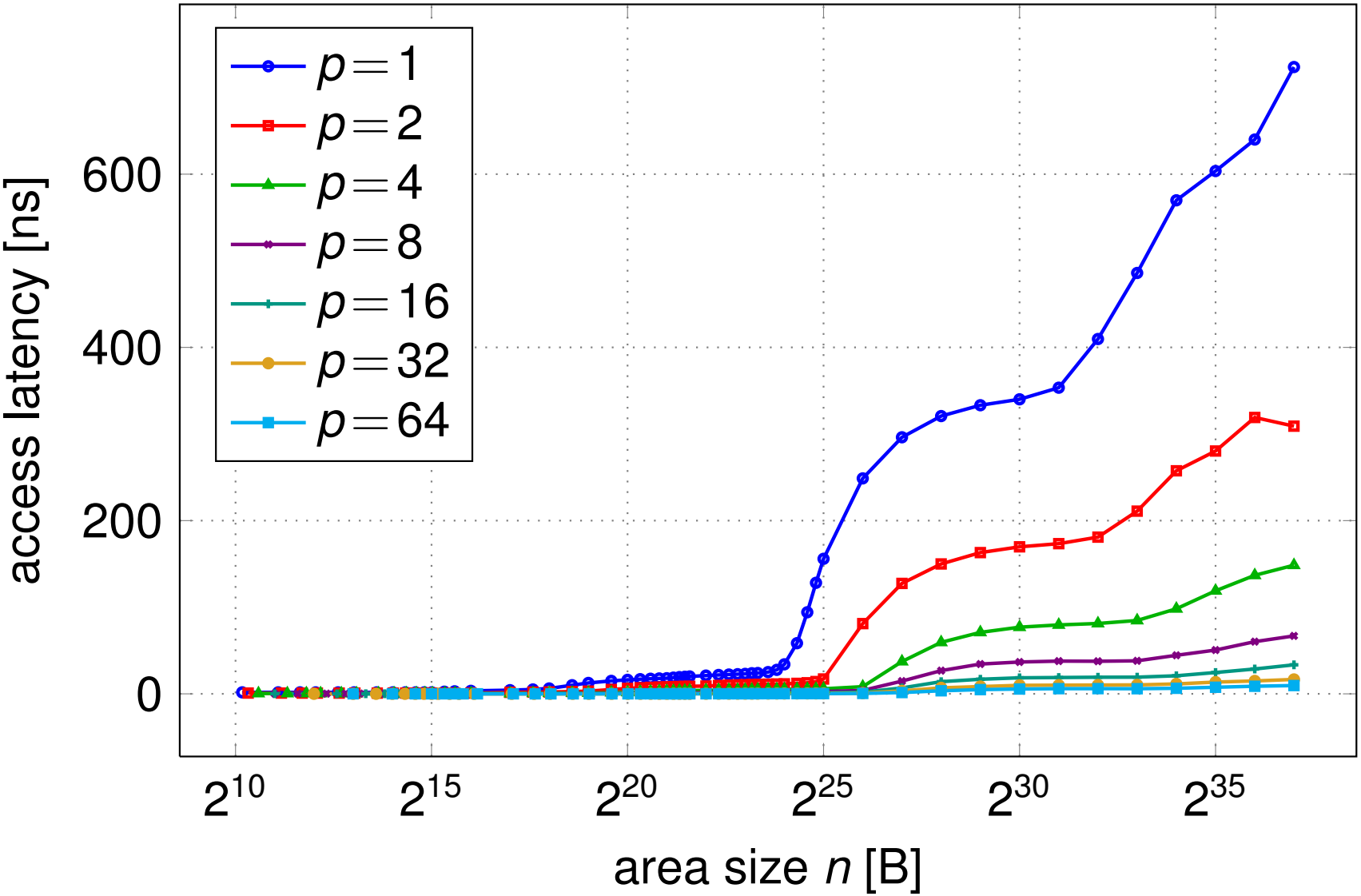
ScanRead64IndexUnrollLoop



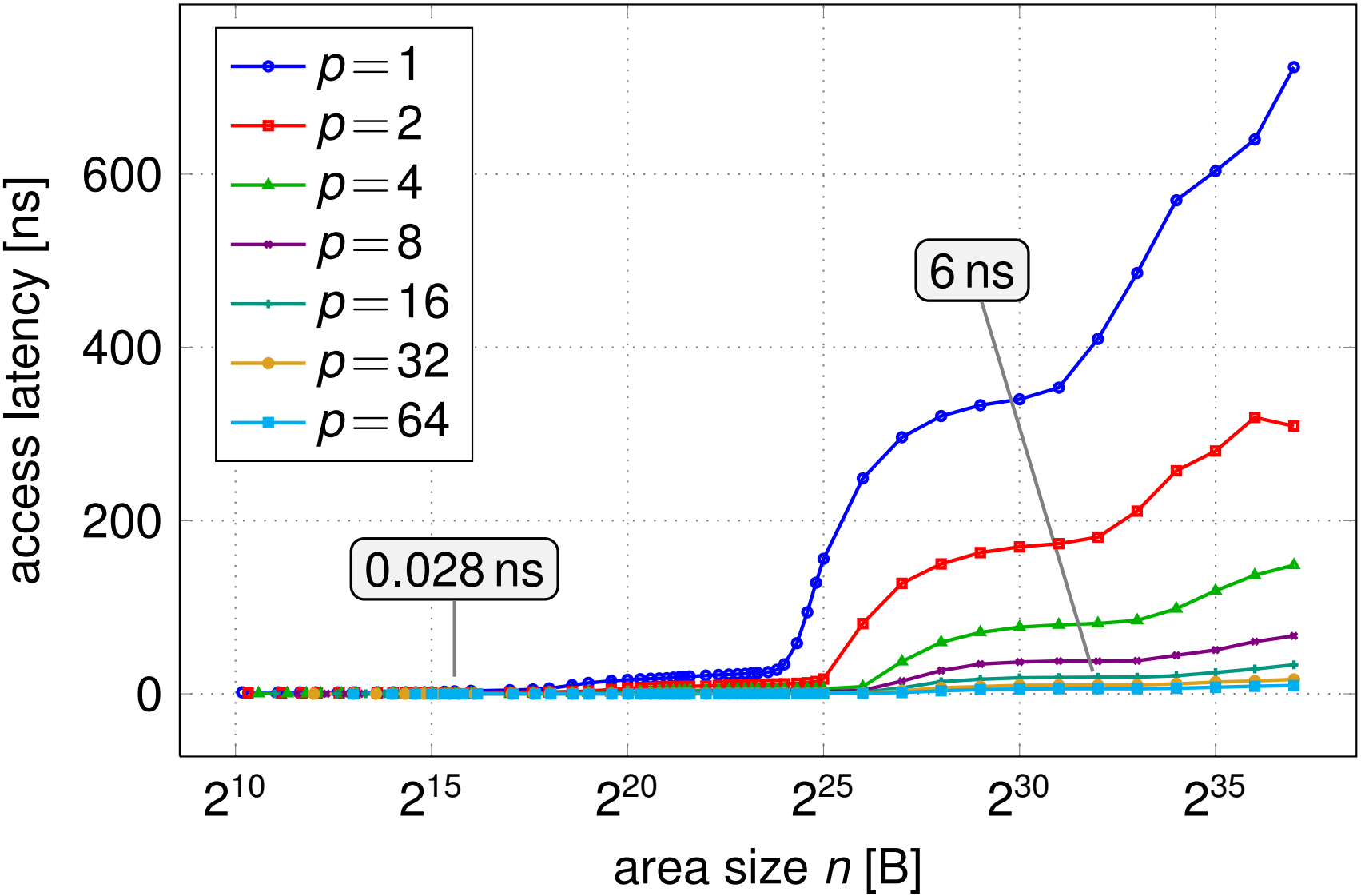
PermRead64SimpleLoop



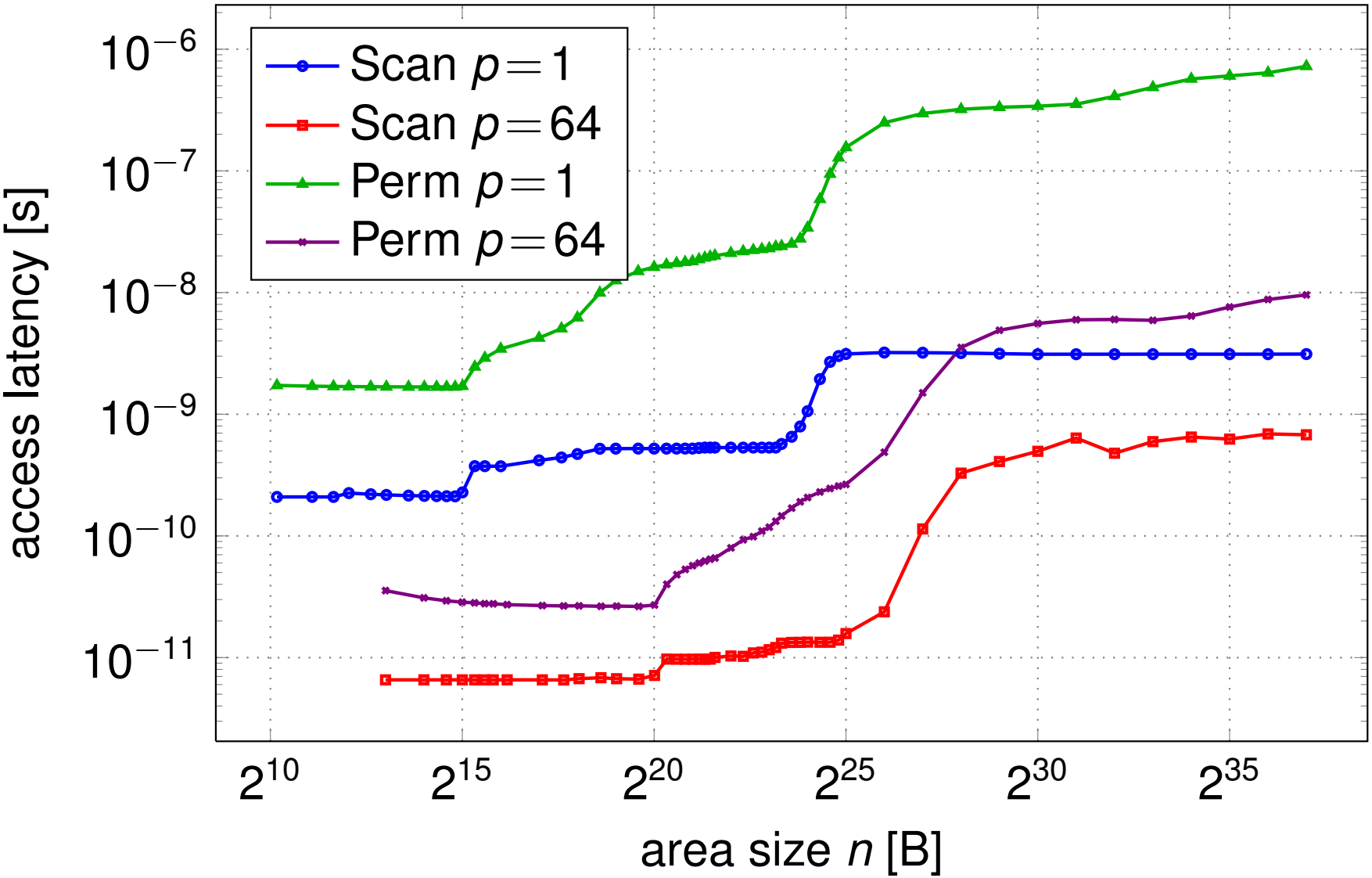
PermRead64SimpleLoop



PermRead64SimpleLoop

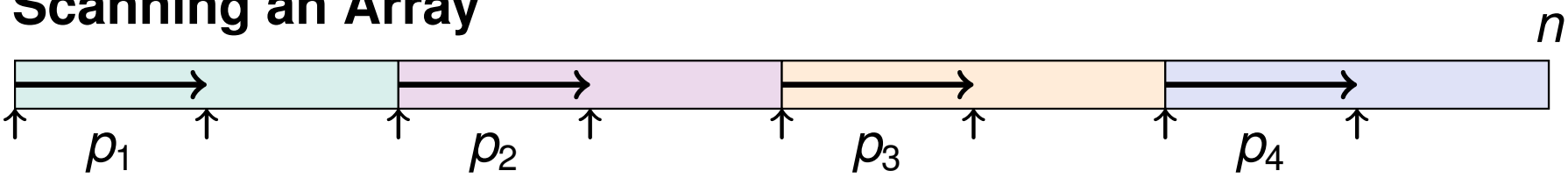


Log-Log Access Latency

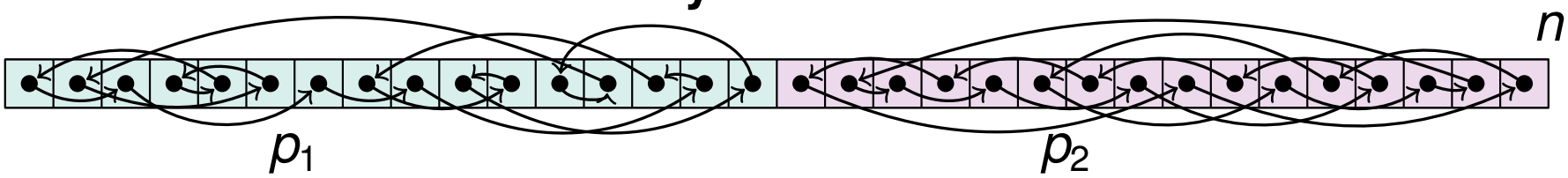


Parallel Memory Bandwidth Test

Scanning an Array



Random Access in an Array



Cache ($n = 16$ KiB)

p	Scan	Random
1	35 GiB/s	4.4 GiB/s
16	567 GiB/s	69 GiB/s
32	1131 GiB/s	137 GiB/s

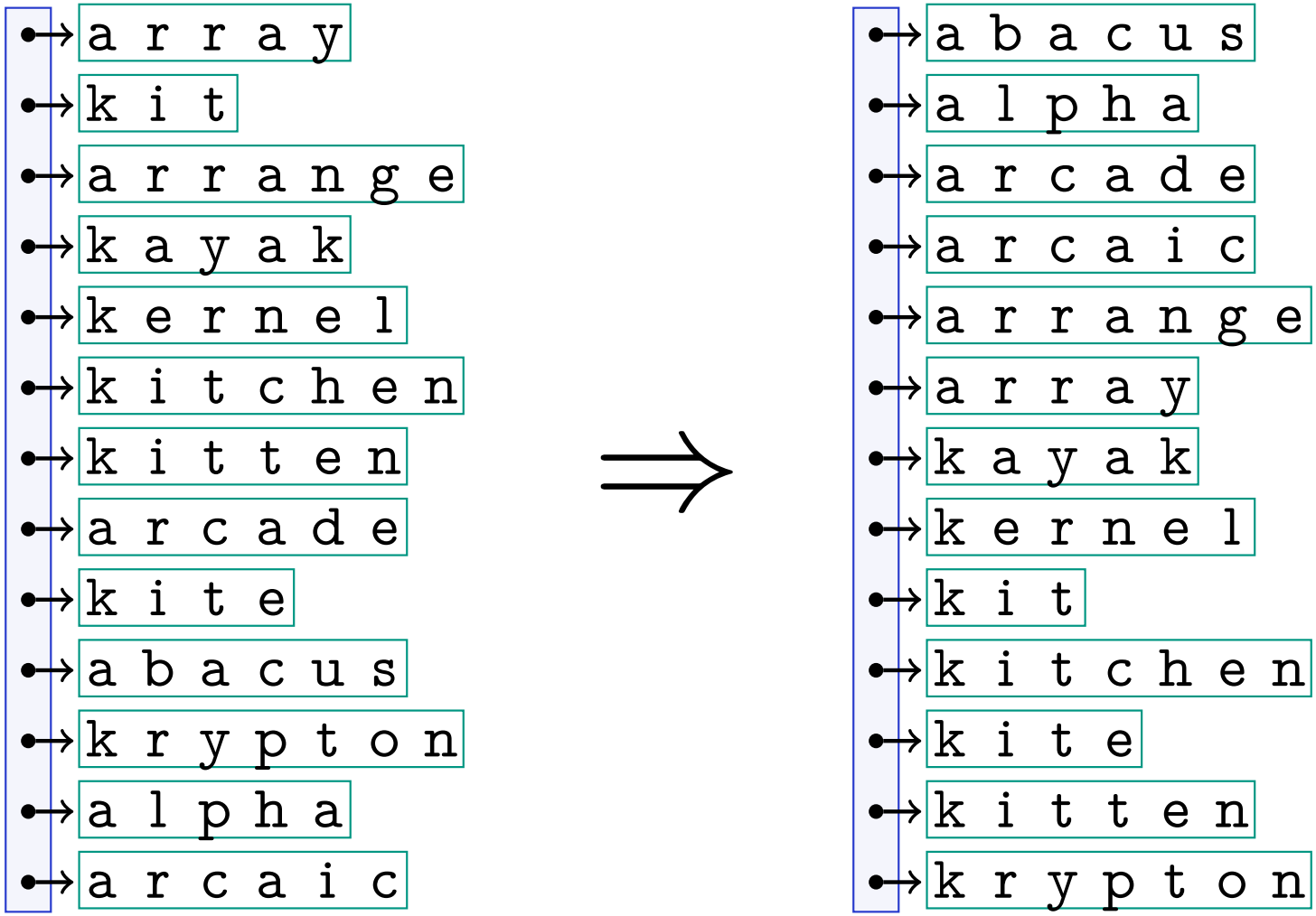
RAM ($n = 4$ GiB)

p	Scan	Random
1	2.6 GiB/s	19 MiB/s
16	10.3 GiB/s	403 MiB/s
32	10.6 GiB/s	763 MiB/s

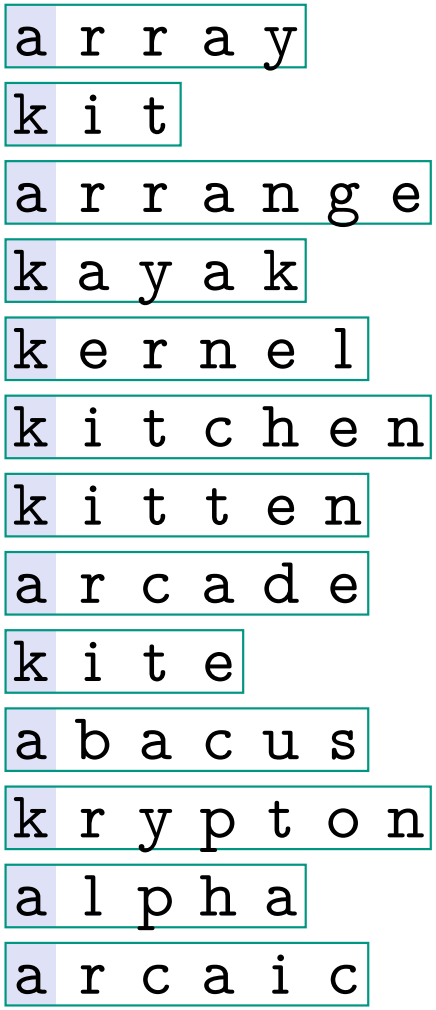
Sorting Strings

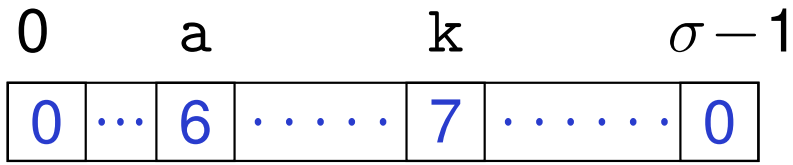
a r r a y		a b a c u s
k i t		a l p h a
a r r a n g e		a r c a d e
k a y a k		a r c a i c
k e r n e l		a r r a n g e
k i t c h e n		a r r a y
k i t t e n	⇒	k a y a k
a r c a d e		k e r n e l
k i t e		k i t
a b a c u s		k i t c h e n
k r y p t o n		k i t e
a l p h a		k i t t e n
a r c a i c		k r y p t o n

Sorting Strings

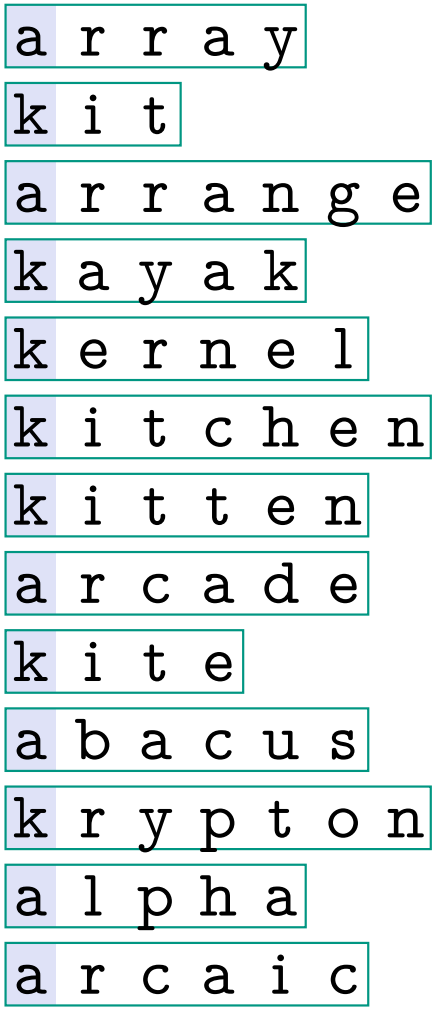


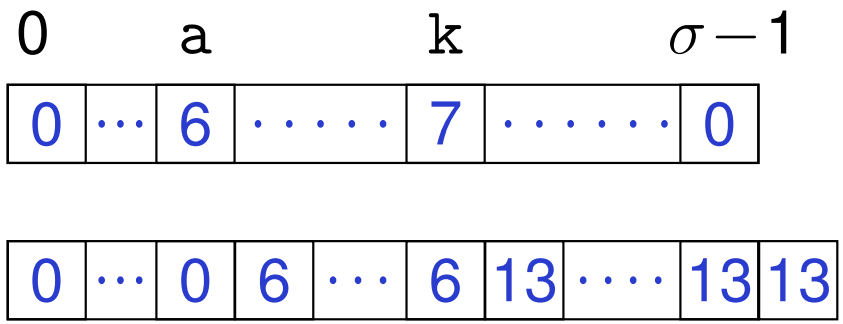
Sorting Strings: Radix Sort





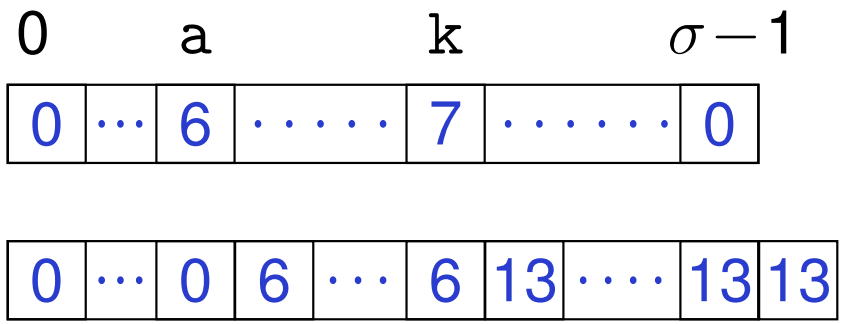
Sorting Strings: Radix Sort



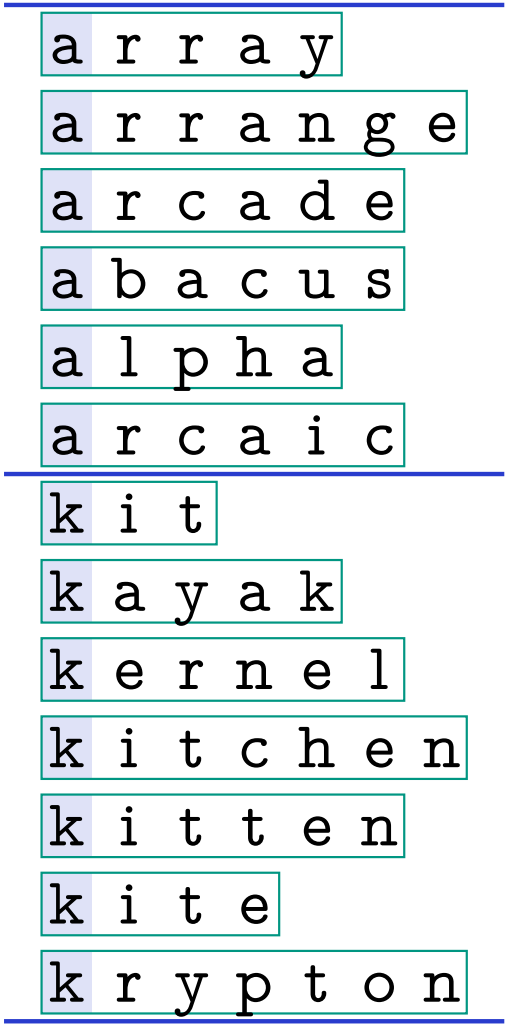


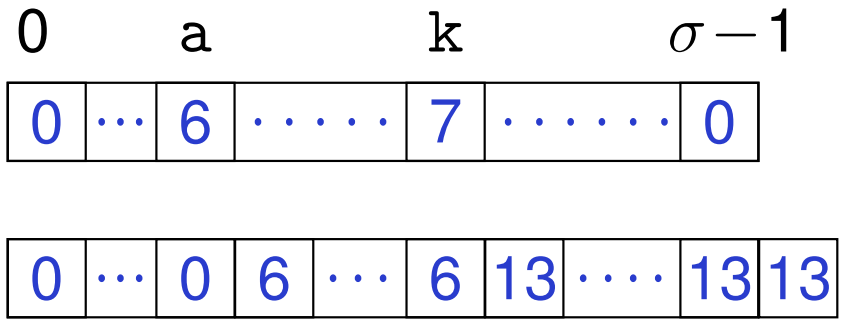
Sorting Strings: Radix Sort

a r r a y
k i t
a r r a n g e
k a y a k
k e r n e l
k i t c h e n
k i t t e n
a r c a d e
k i t e
a b a c u s
k r y p t o n
a l p h a
a r c a i c

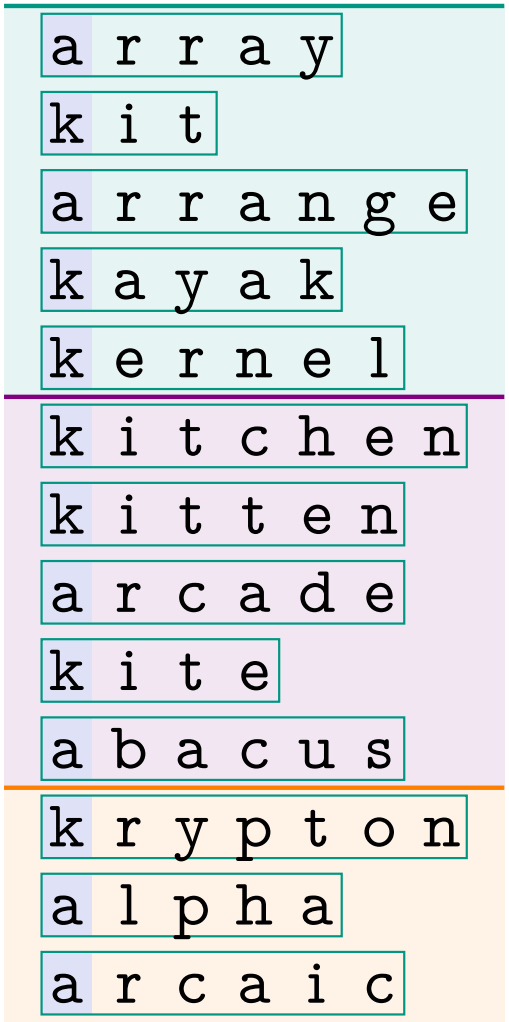


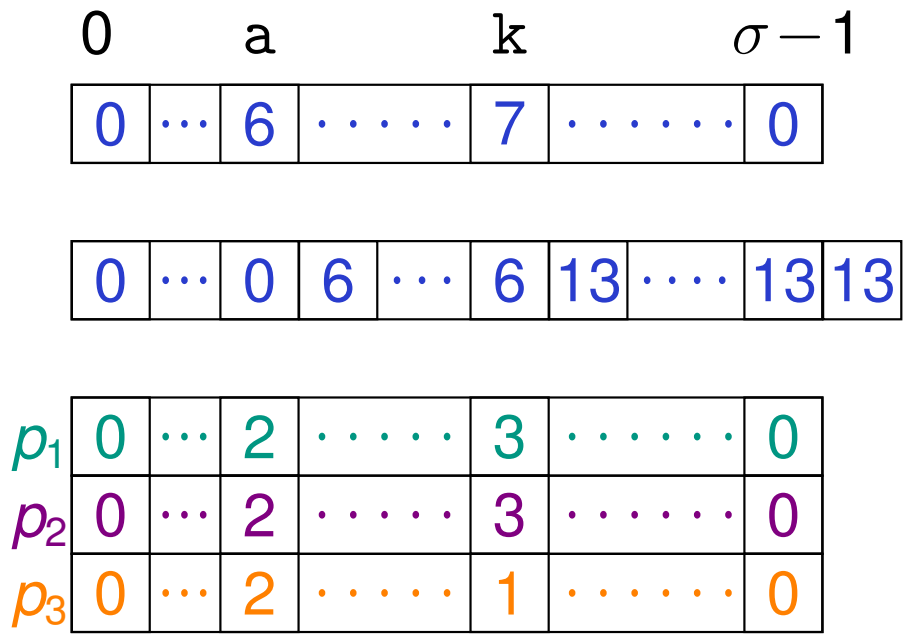
Sorting Strings: Radix Sort



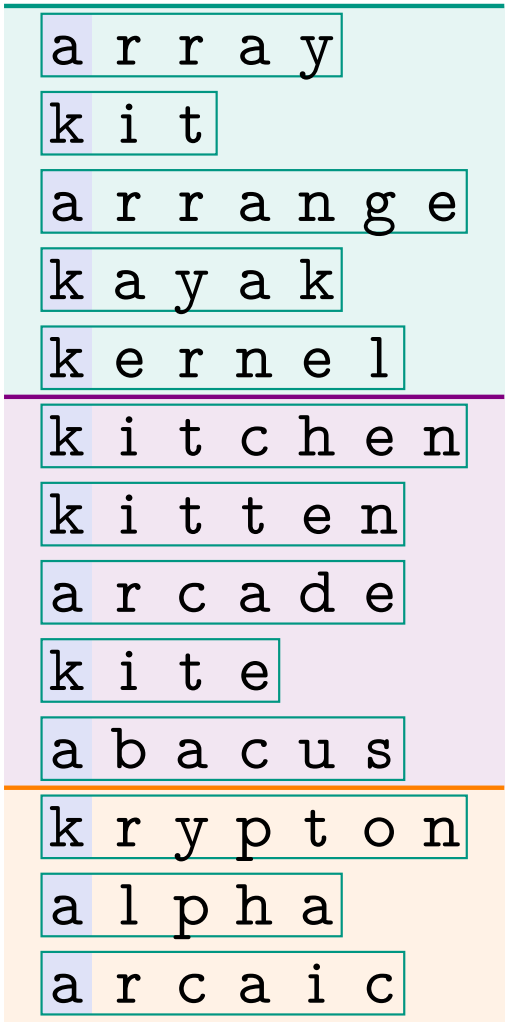


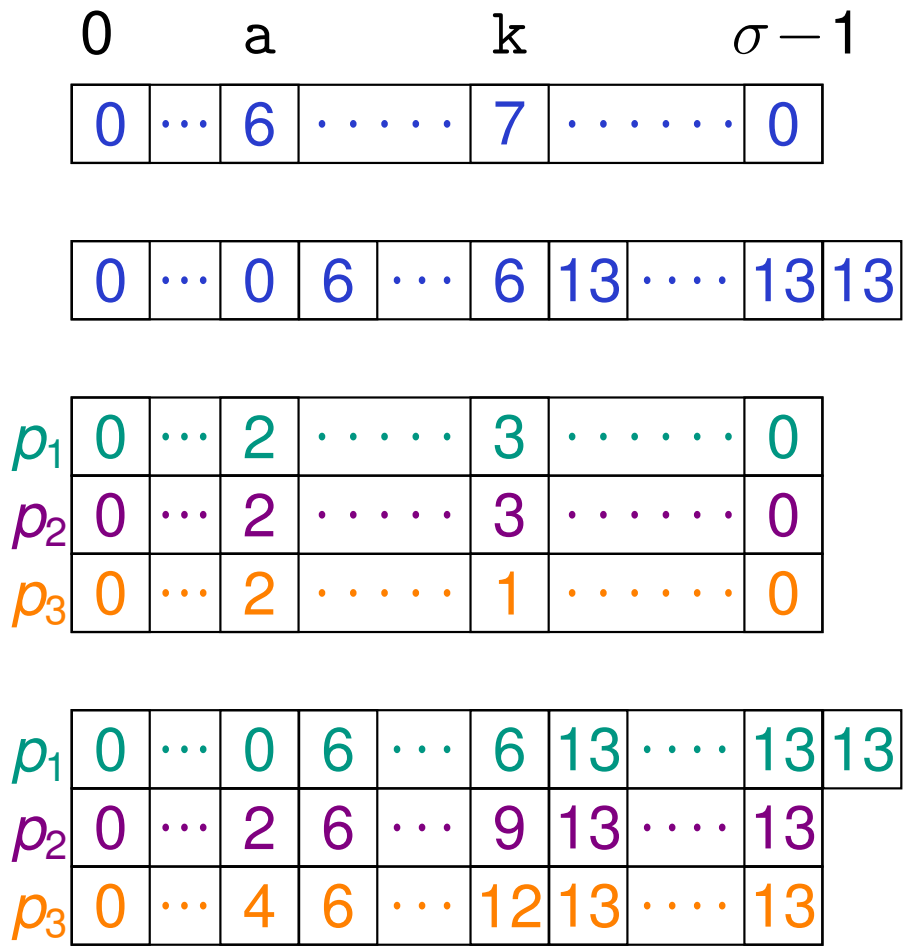
Sorting Strings: Radix Sort





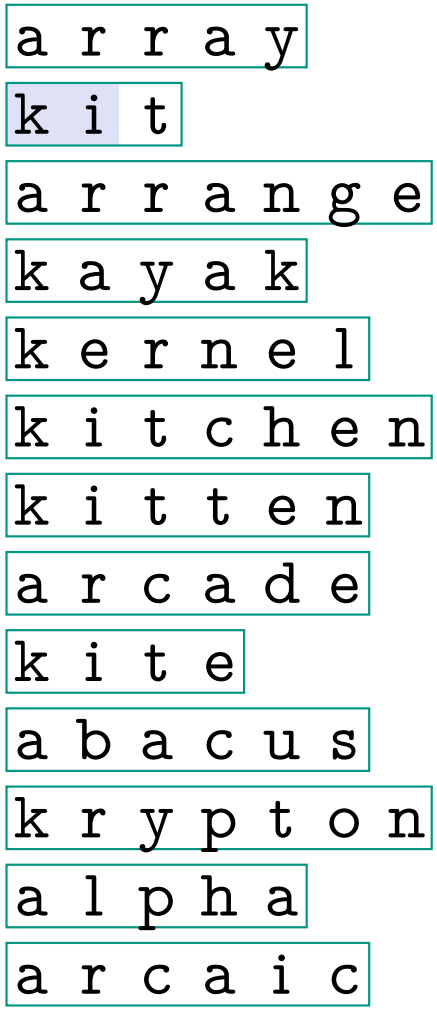
Sorting Strings: Radix Sort





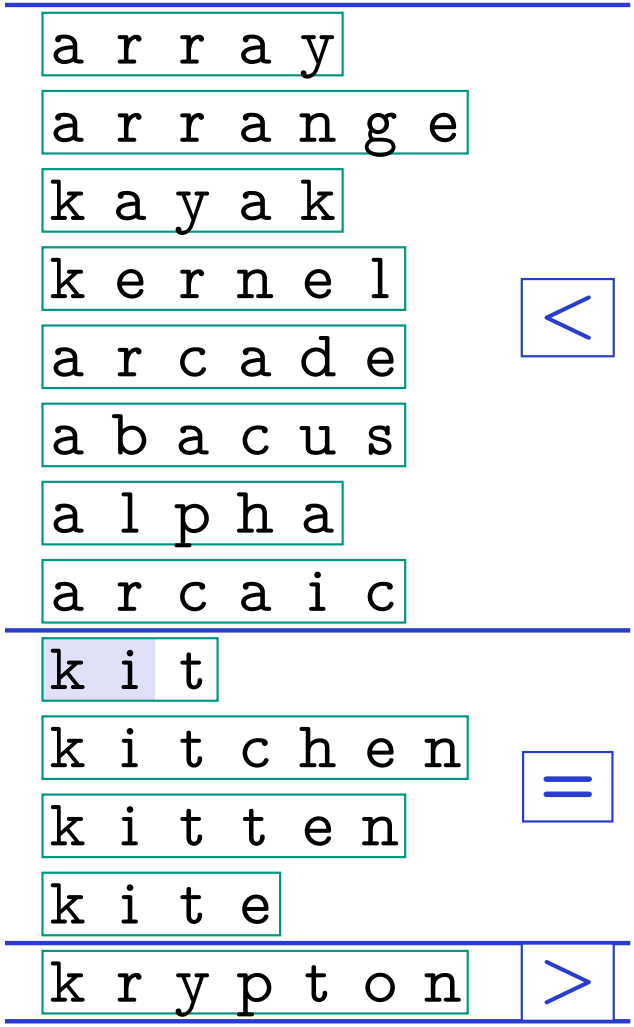
Sorting Strings: Multikey Quicksort

[Bentley, Sedgewick '97]



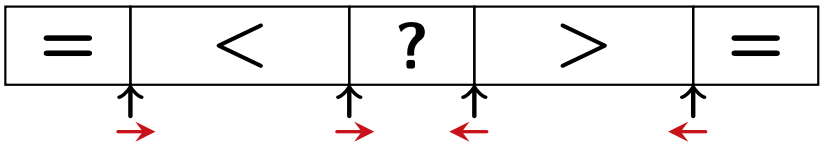
Sorting Strings: Multikey Quicksort

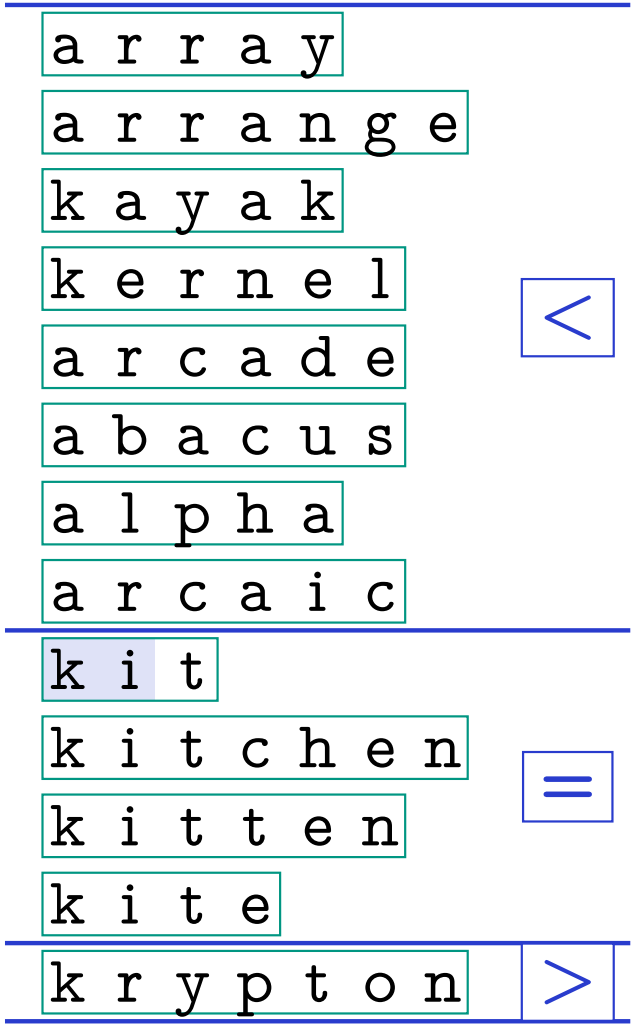
[Bentley, Sedgewick '97]



Sorting Strings: Multikey Quicksort

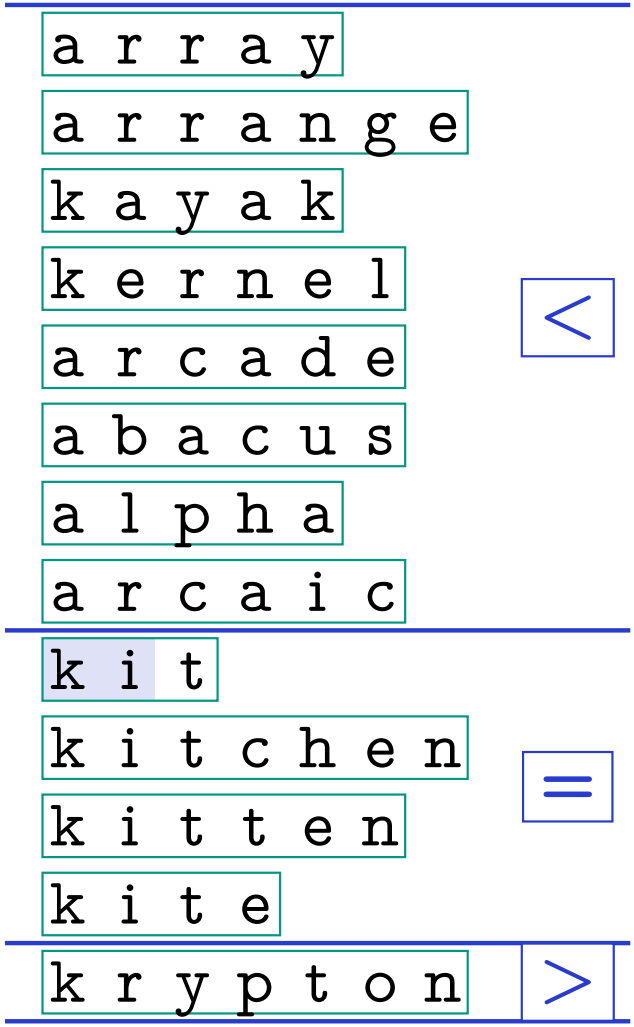
[Bentley, Sedgewick '97]

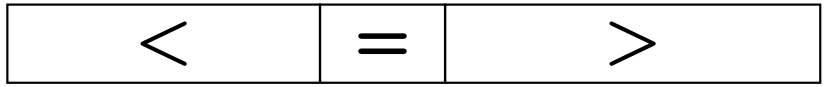




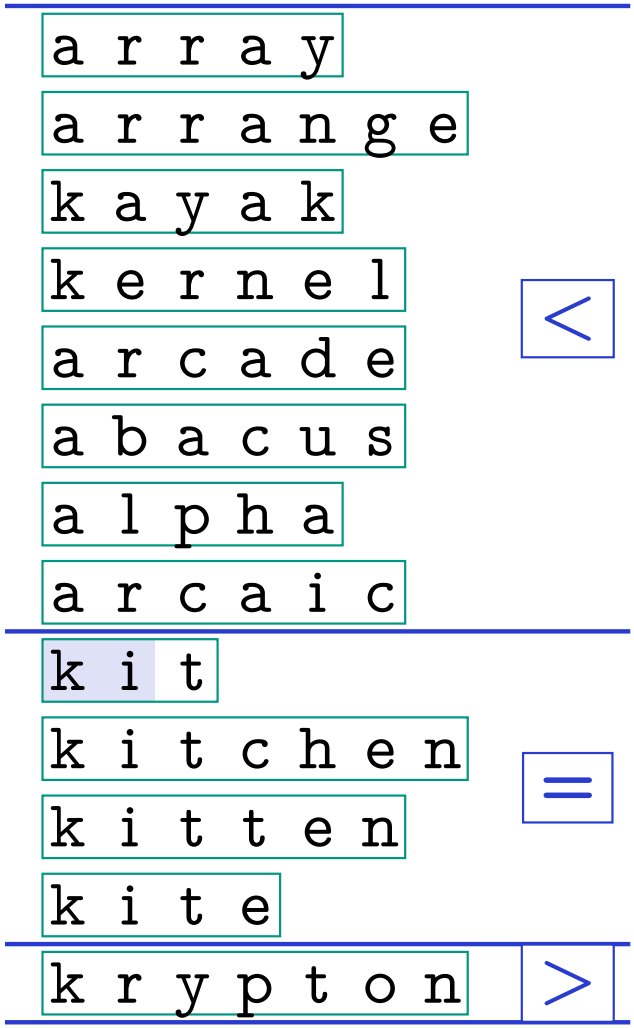
Sorting Strings: Multikey Quicksort

[Bentley, Sedgewick '97]

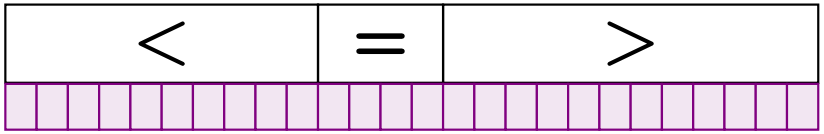




Sorting Strings: Multikey Quicksort



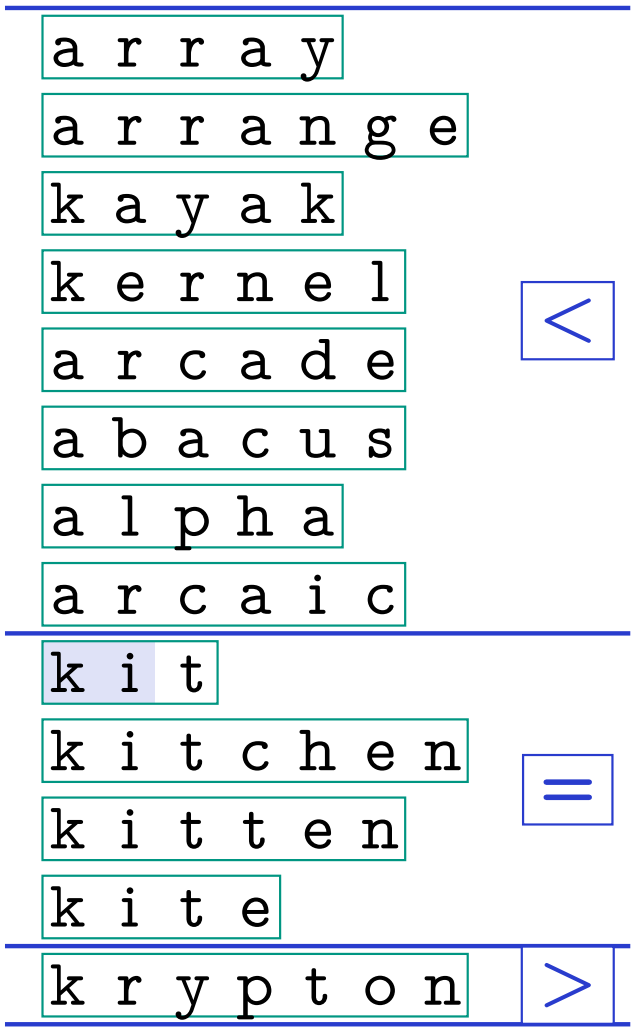
[Bentley, Sedgewick '97]



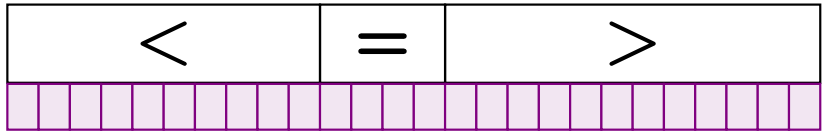
[Rantala '??]

- partition by $w = 8$ characters
- cache characters
  $\Rightarrow$ fewer random accesses
- fastest sequential algorithm

Sorting Strings: Multikey Quicksort



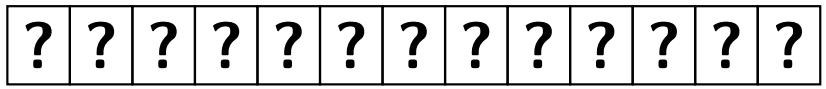
[Bentley, Sedgewick '97]



[Rantala '??]

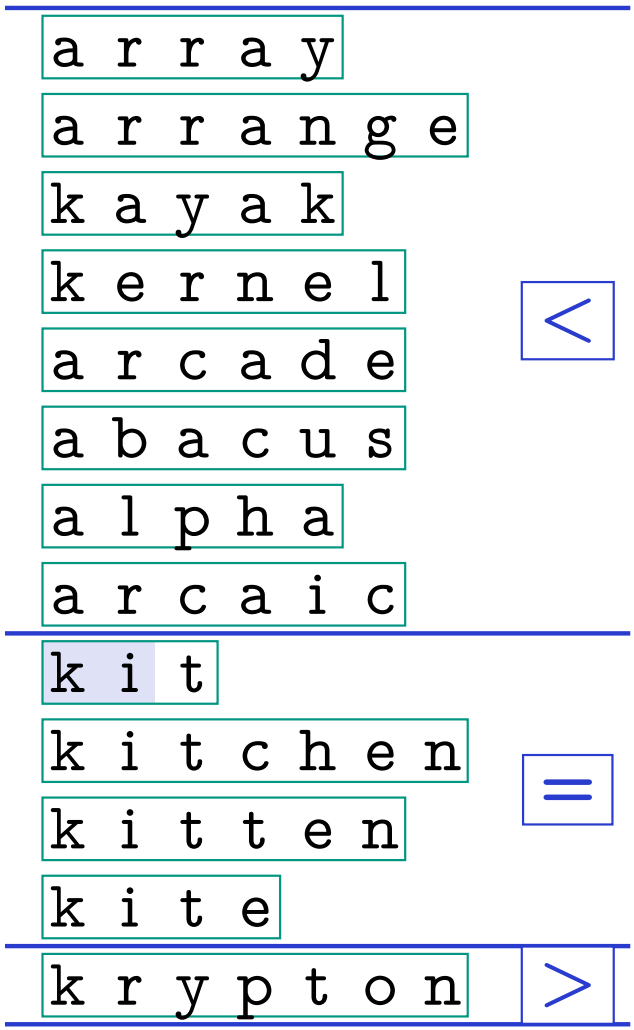
- partition by $w = 8$ characters
- cache characters
  $\Rightarrow$ fewer random accesses
- fastest sequential algorithm

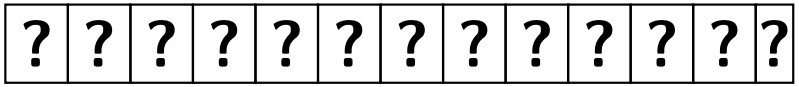
[This Work]



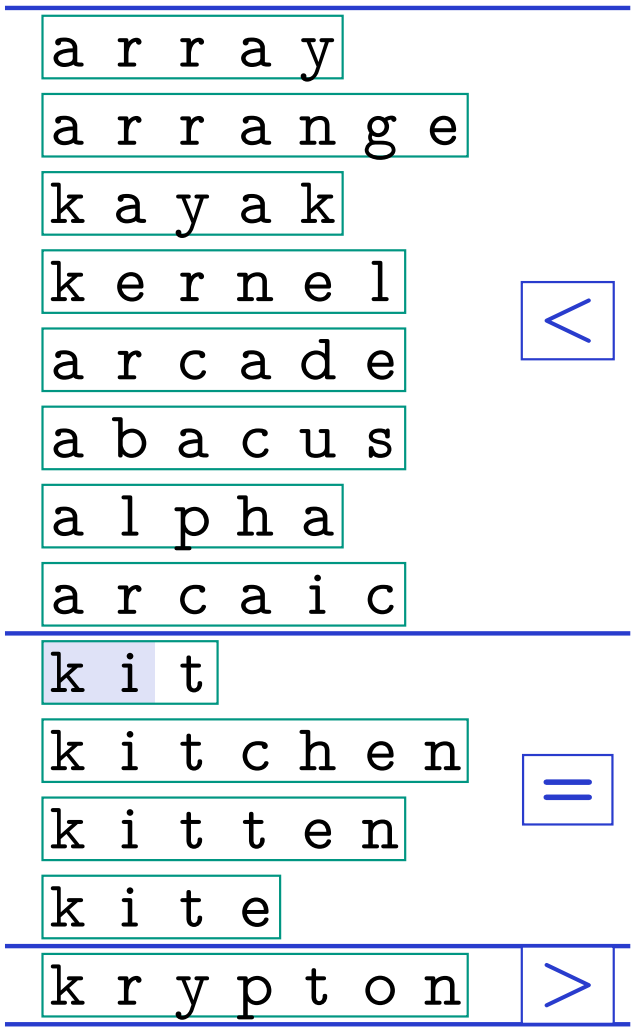
- parallelize using blocks

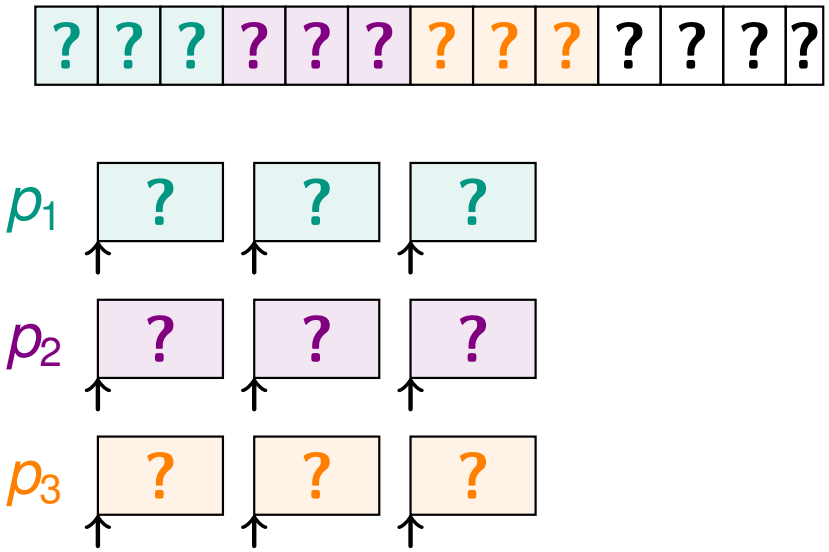
Sorting Strings: Multikey Quicksort



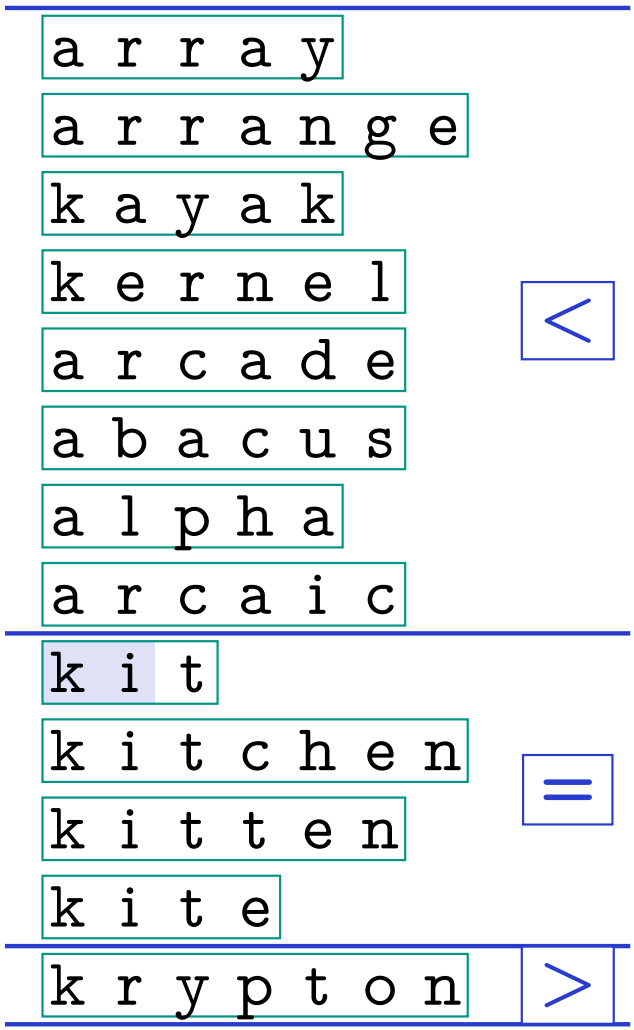


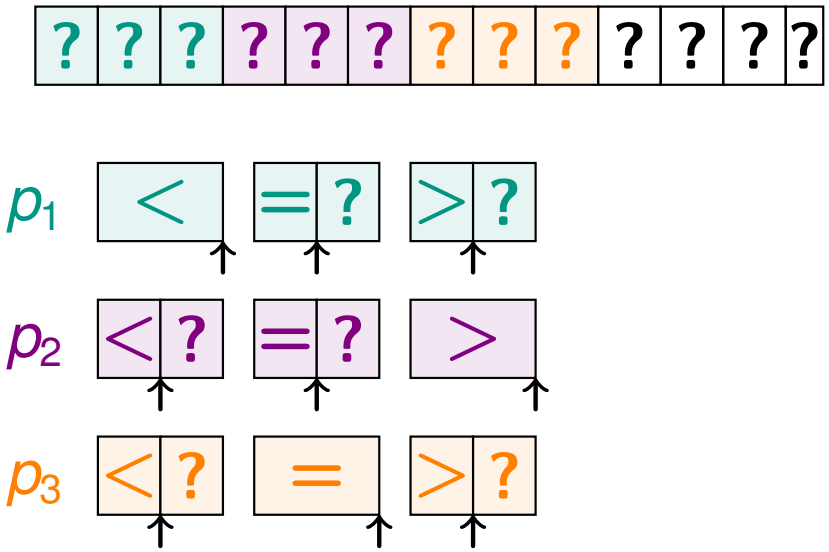
Sorting Strings: Multikey Quicksort



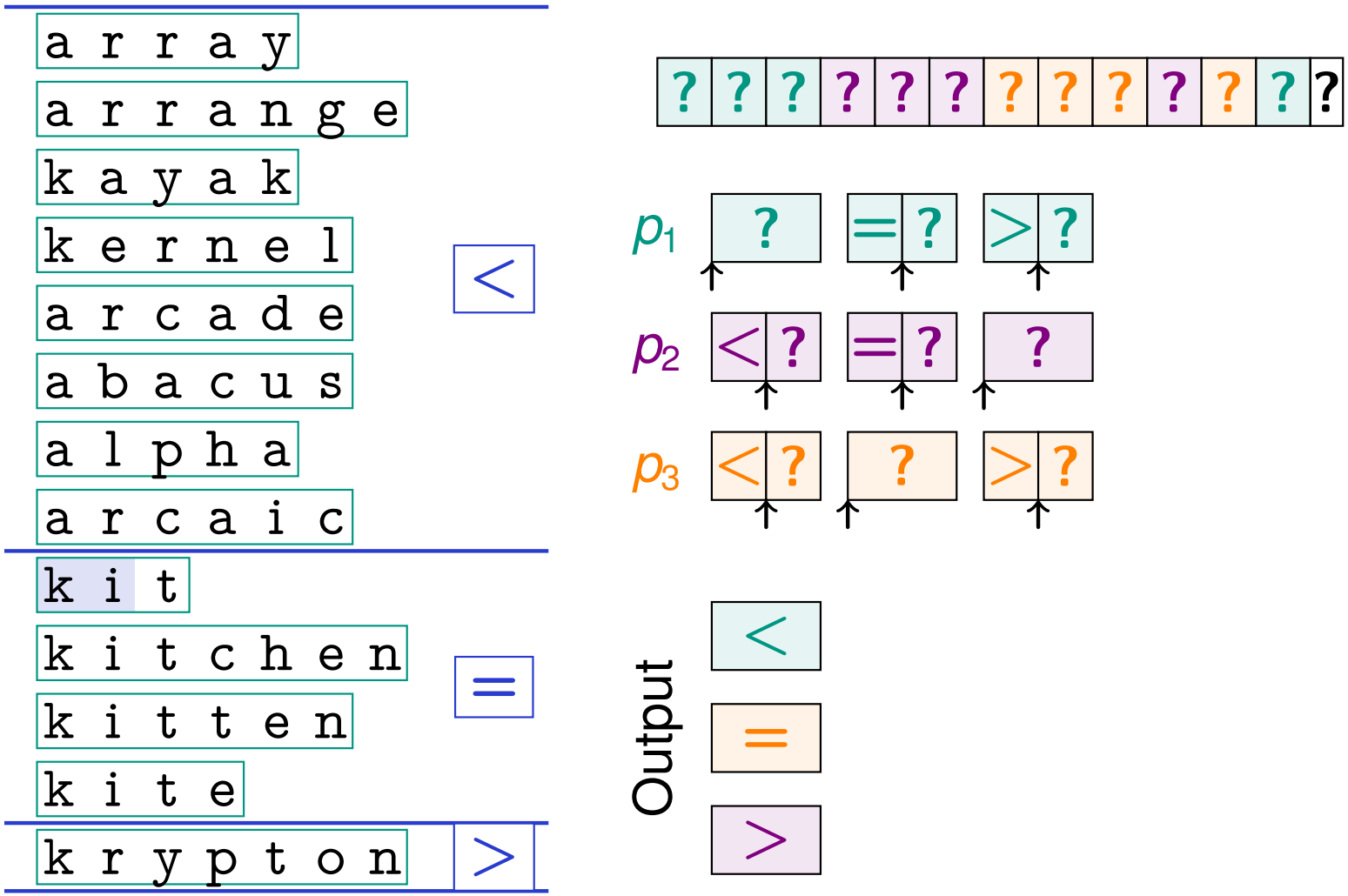


Sorting Strings: Multikey Quicksort

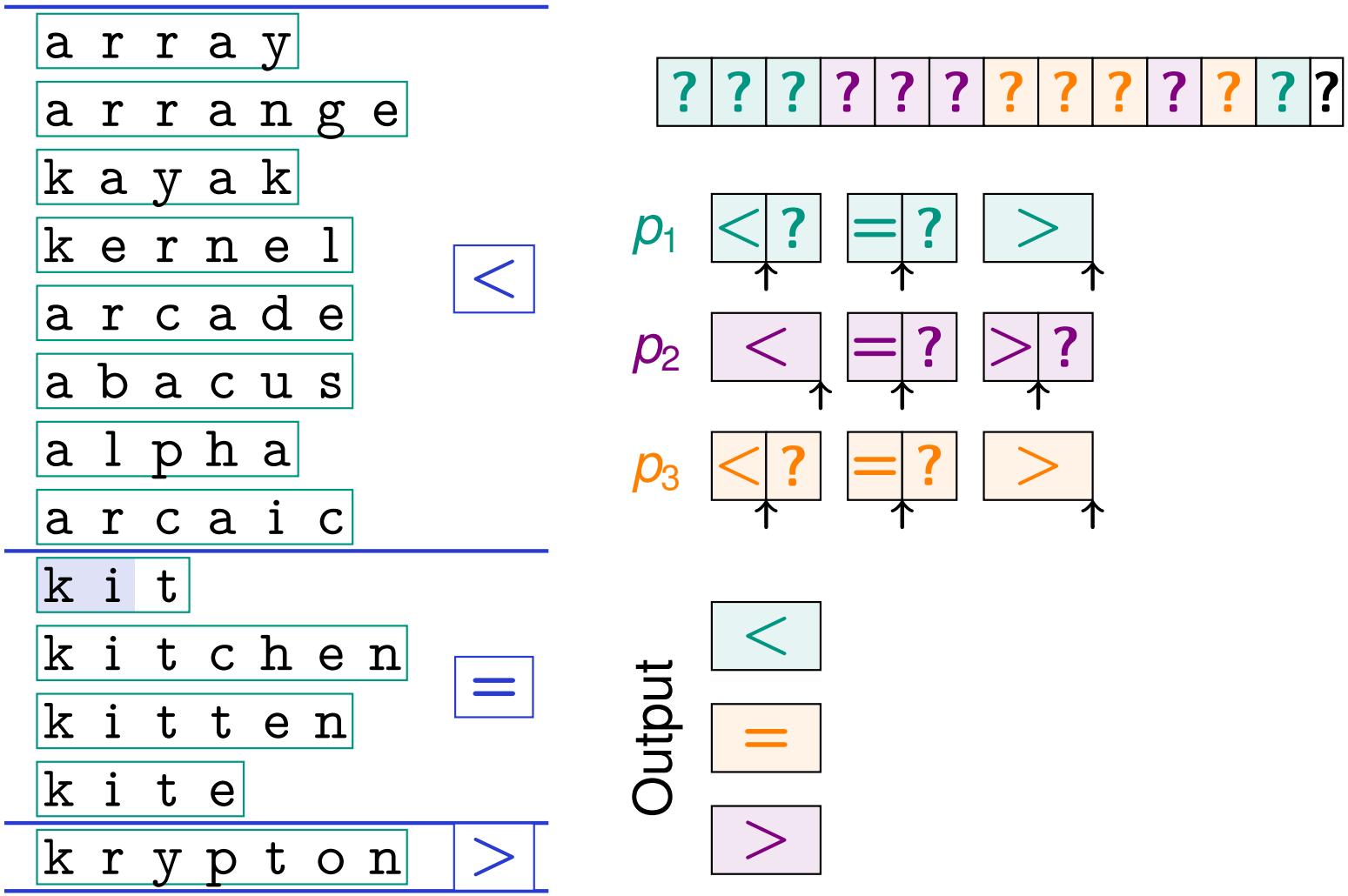




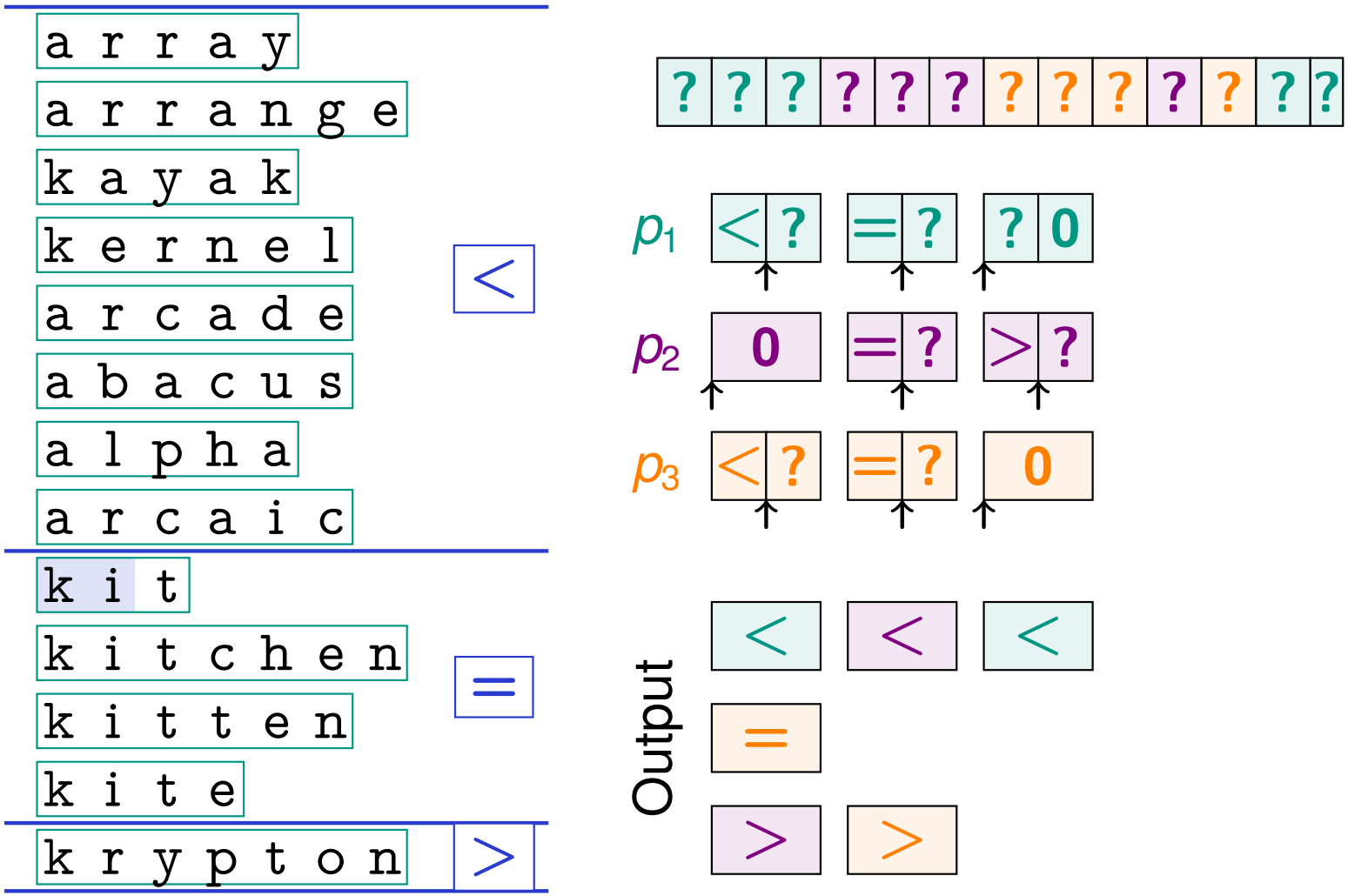
Sorting Strings: Multikey Quicksort



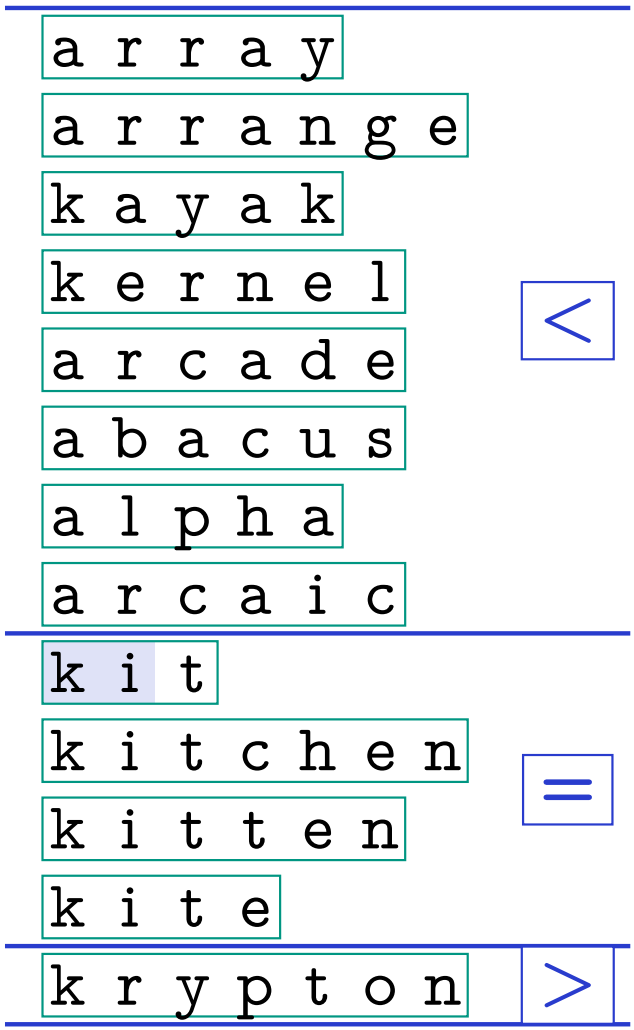
Sorting Strings: Multikey Quicksort

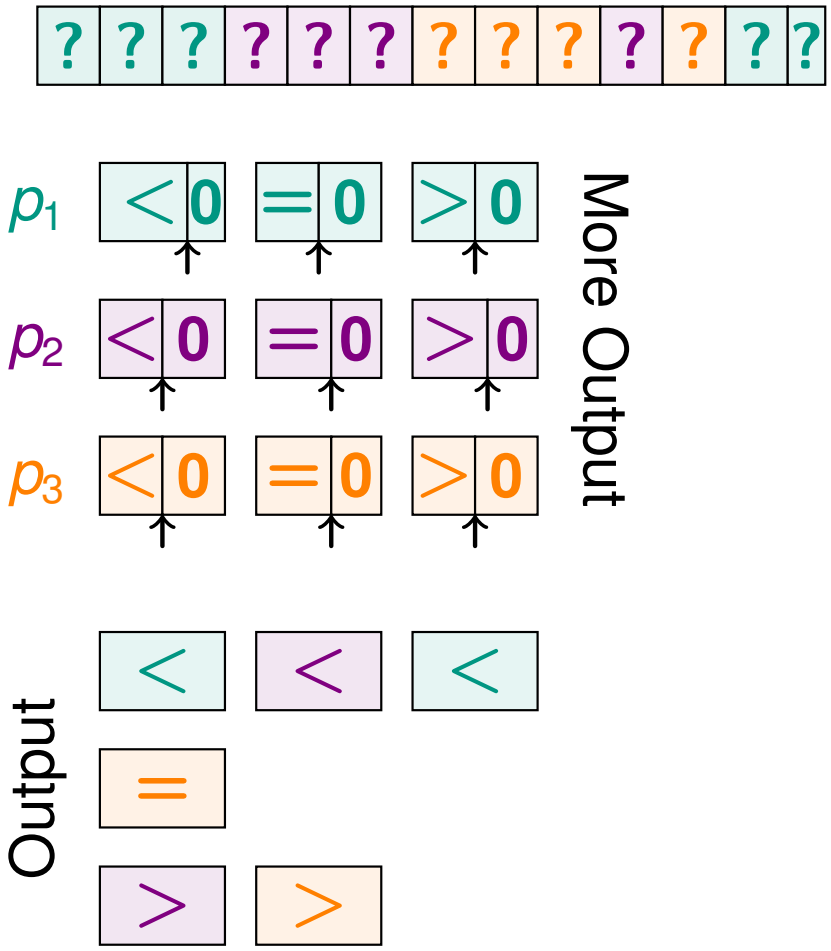


Sorting Strings: Multikey Quicksort



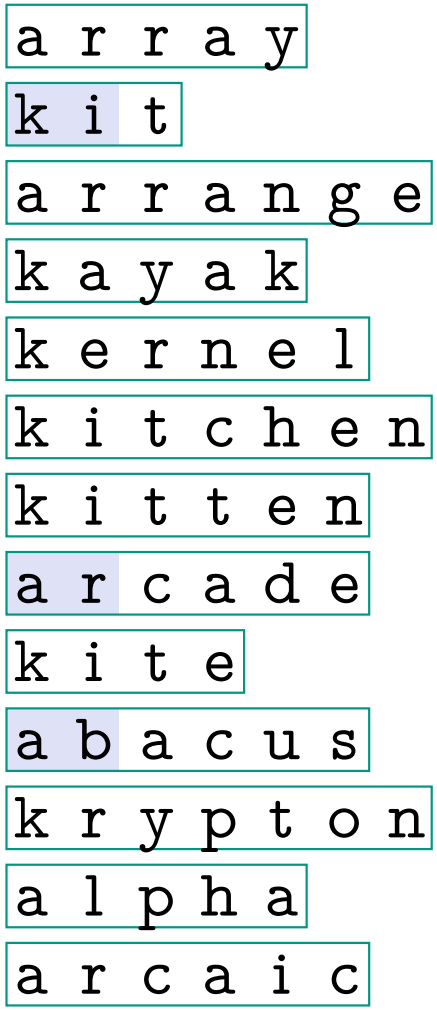
Sorting Strings: Multikey Quicksort





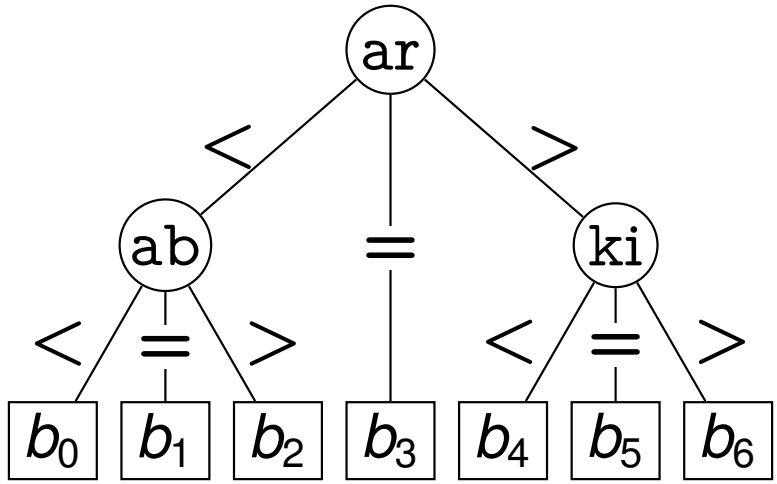
Super Scalar String Sample Sort (S^5)

Karlsruhe Institute of Technology

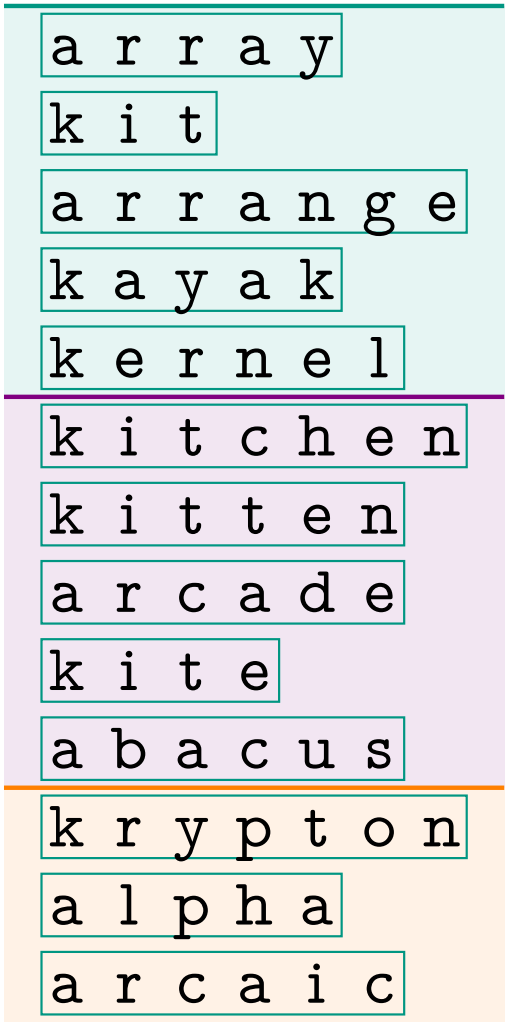


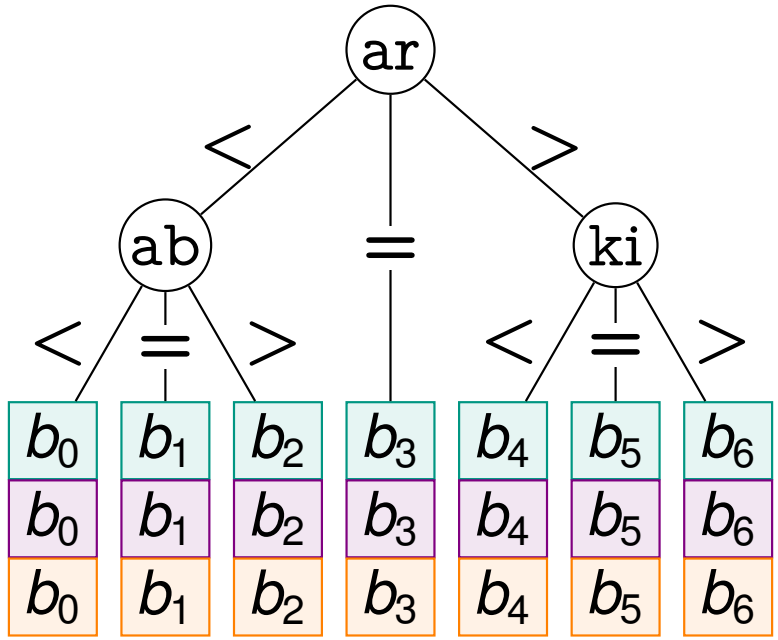
Super Scalar String Sample Sort (S^5)

a r r a y
k i t
a r r a n g e
k a y a k
k e r n e l
k i t c h e n
k i t t e n
a r c a d e
k i t e
a b a c u s
k r y p t o n
a l p h a
a r c a i c

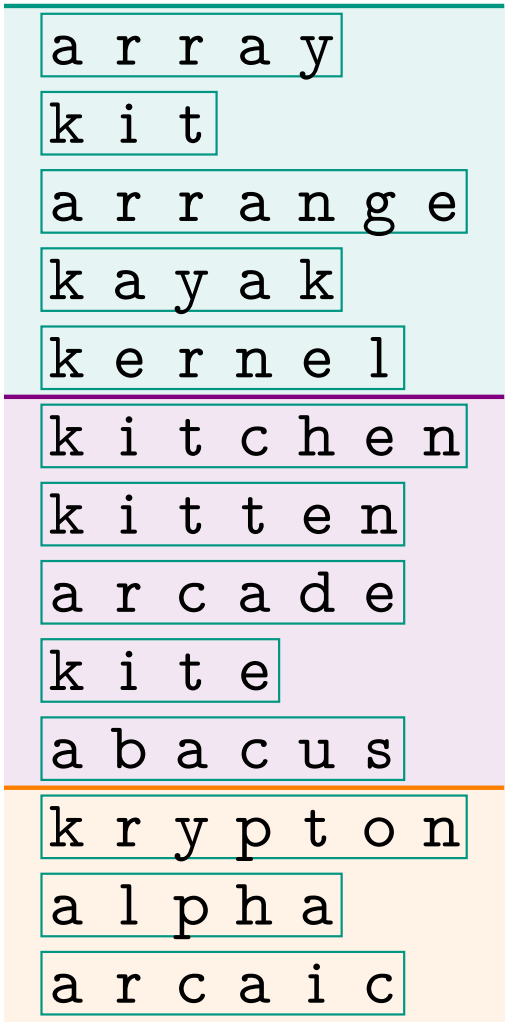


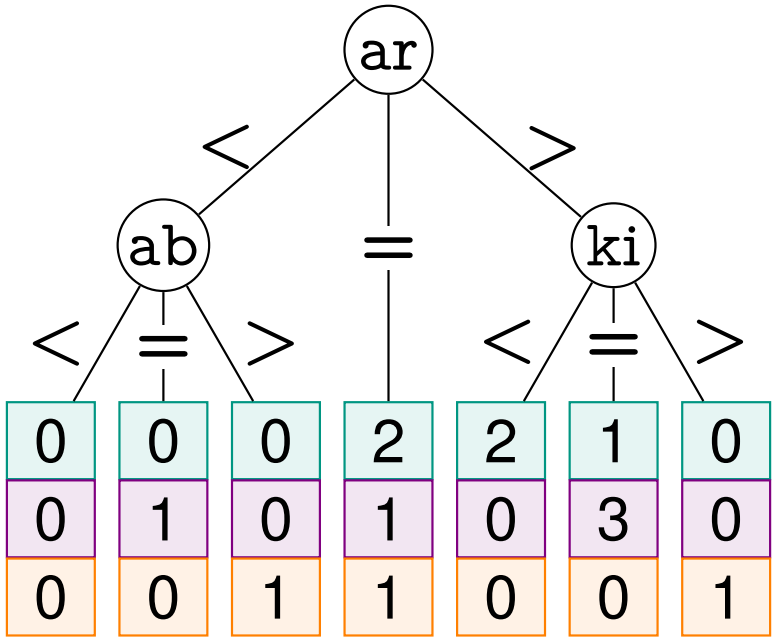
Super Scalar String Sample Sort (S^5)



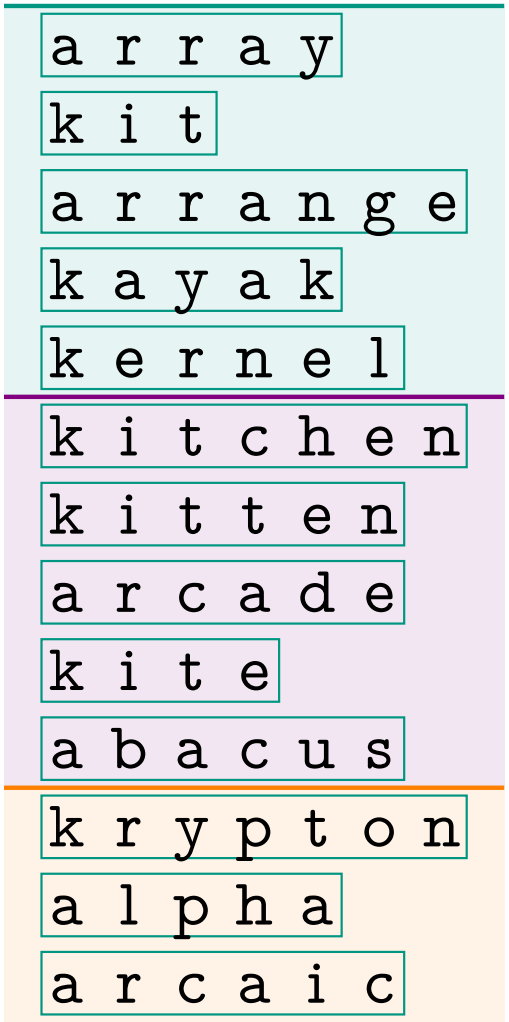


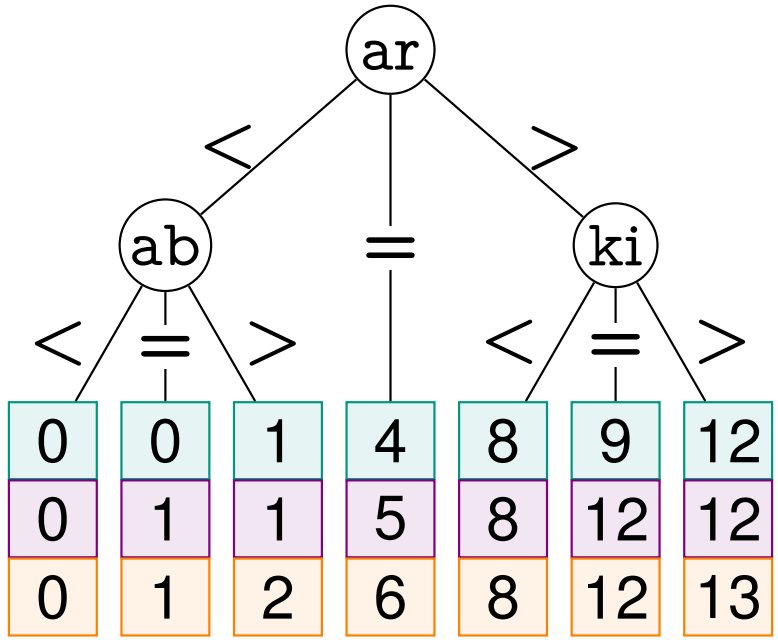
Super Scalar String Sample Sort (S^5)





Super Scalar String Sample Sort (S^5)





Super Scalar String Sample Sort (S^5)

a b a c u s

a l p h a

a r r a y

a r r a n g e

a r c a d e

a r c a i c

k a y a k

k e r n e l

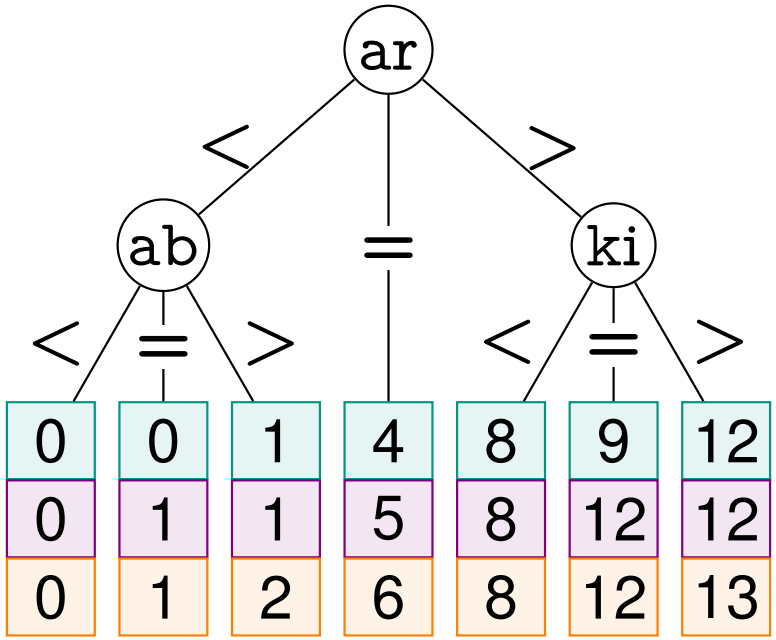
k i t

k i t c h e n

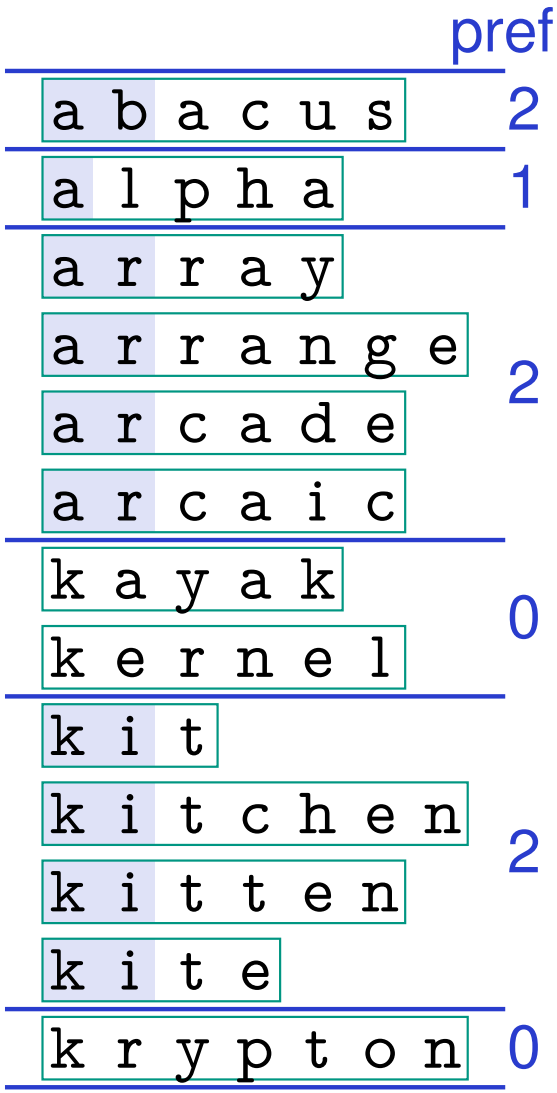
k i t t e n

k i t e

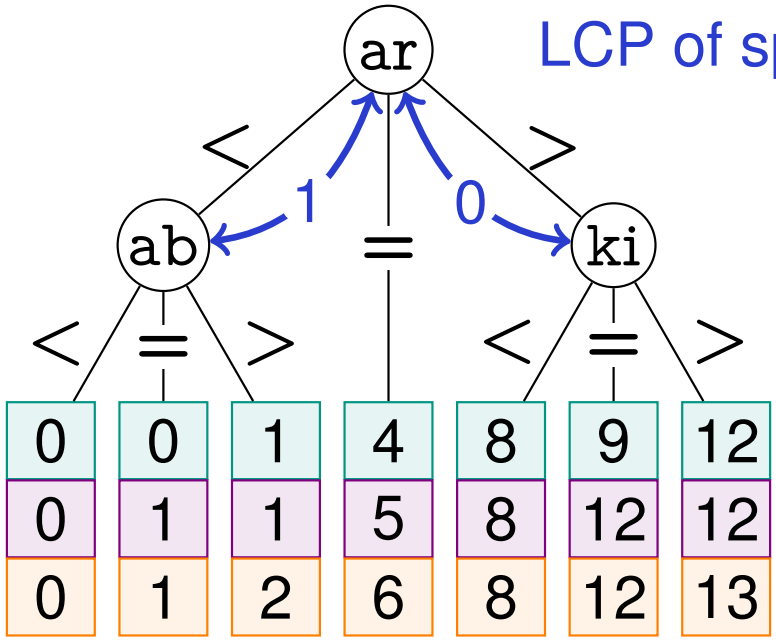
k r y p t o n



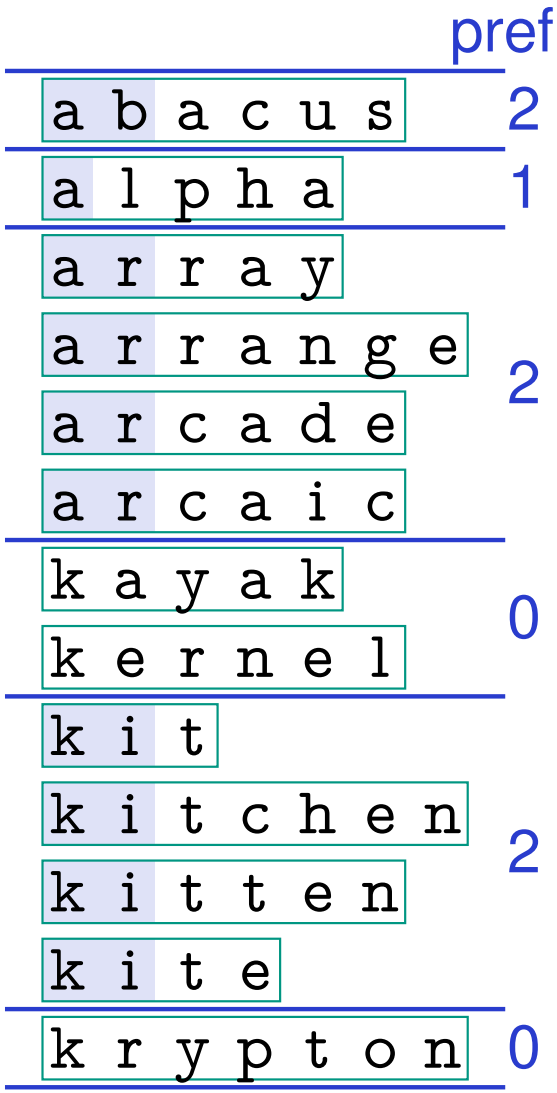
Super Scalar String Sample Sort (S^5)



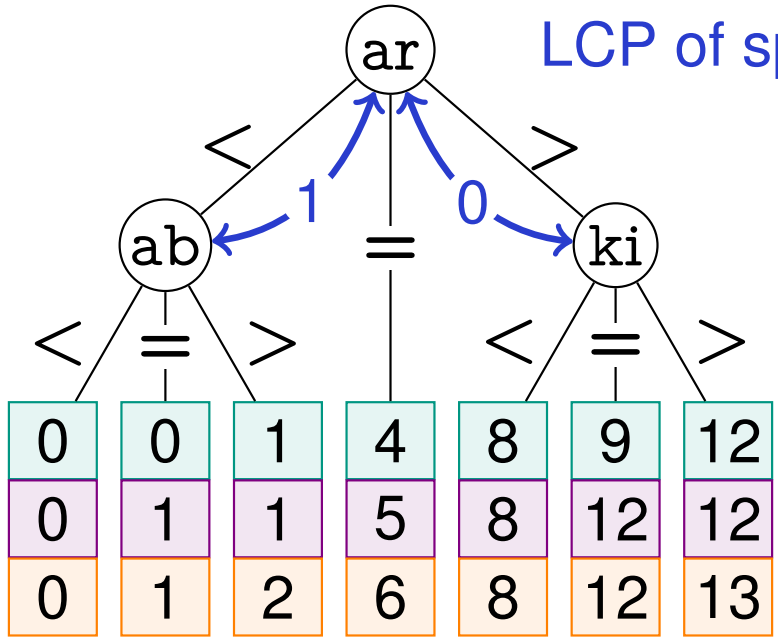
increase prefix by LCP of splitters or key size.



Super Scalar String Sample Sort (S^5)

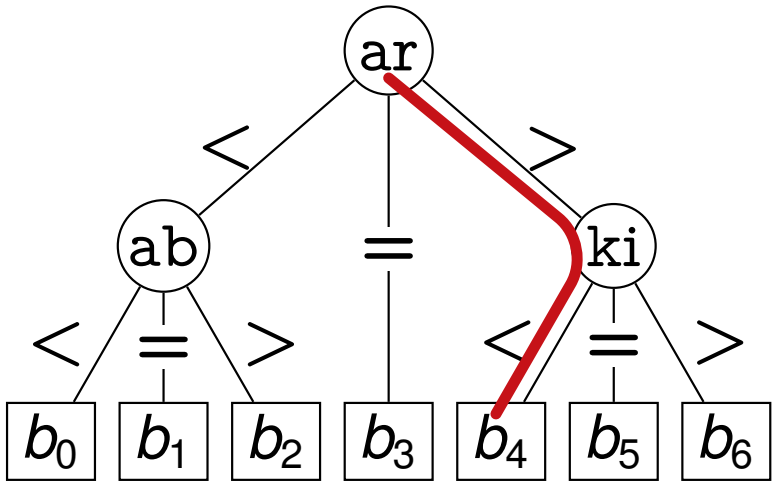


increase prefix by LCP of splitters or key size.



- partitions by $w = 8$ chars
- easy parallelization

Super Scalar String Sample Sort (S^5)



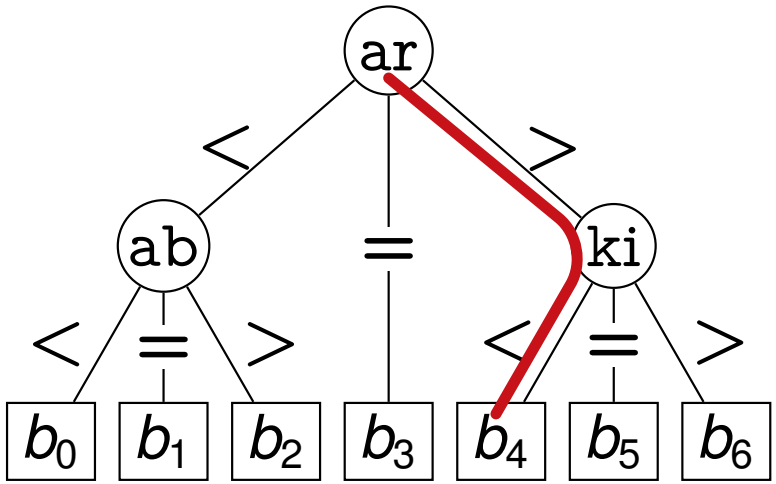
- predicated instructions

1	2	3
ar	ab	ki

$i := 2i + 0/1$

- partitions by $w = 8$ chars
- easy parallelization
- 256 KiB L2 cache: 13 levels

Super Scalar String Sample Sort (S^5)



- partitions by $w = 8$ chars
- easy parallelization
- 256 KiB L2 cache: 13 levels

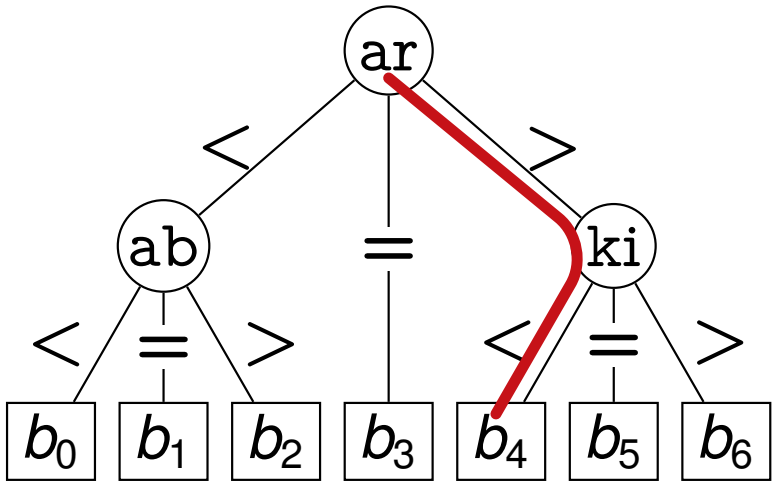
- predicated instructions

1	2	3
ar	ab	ki

$$i := 2i + 0/1$$

- equality checking:
  1. at each node
  2. after full descent

Super Scalar String Sample Sort (S^5)



- partitions by $w = 8$ chars
- easy parallelization
- 256 KiB L2 cache: 13 levels

- predicated instructions

1	2	3
ar	ab	ki

$$i := 2i + 0/1$$

- equality checking:
  1. at each node
  2. after full descent

- interleave tree descents: classify 4 strings at once
  $\Rightarrow$ super scalar parallelism

Parallel S^5 – Sub-Algorithms

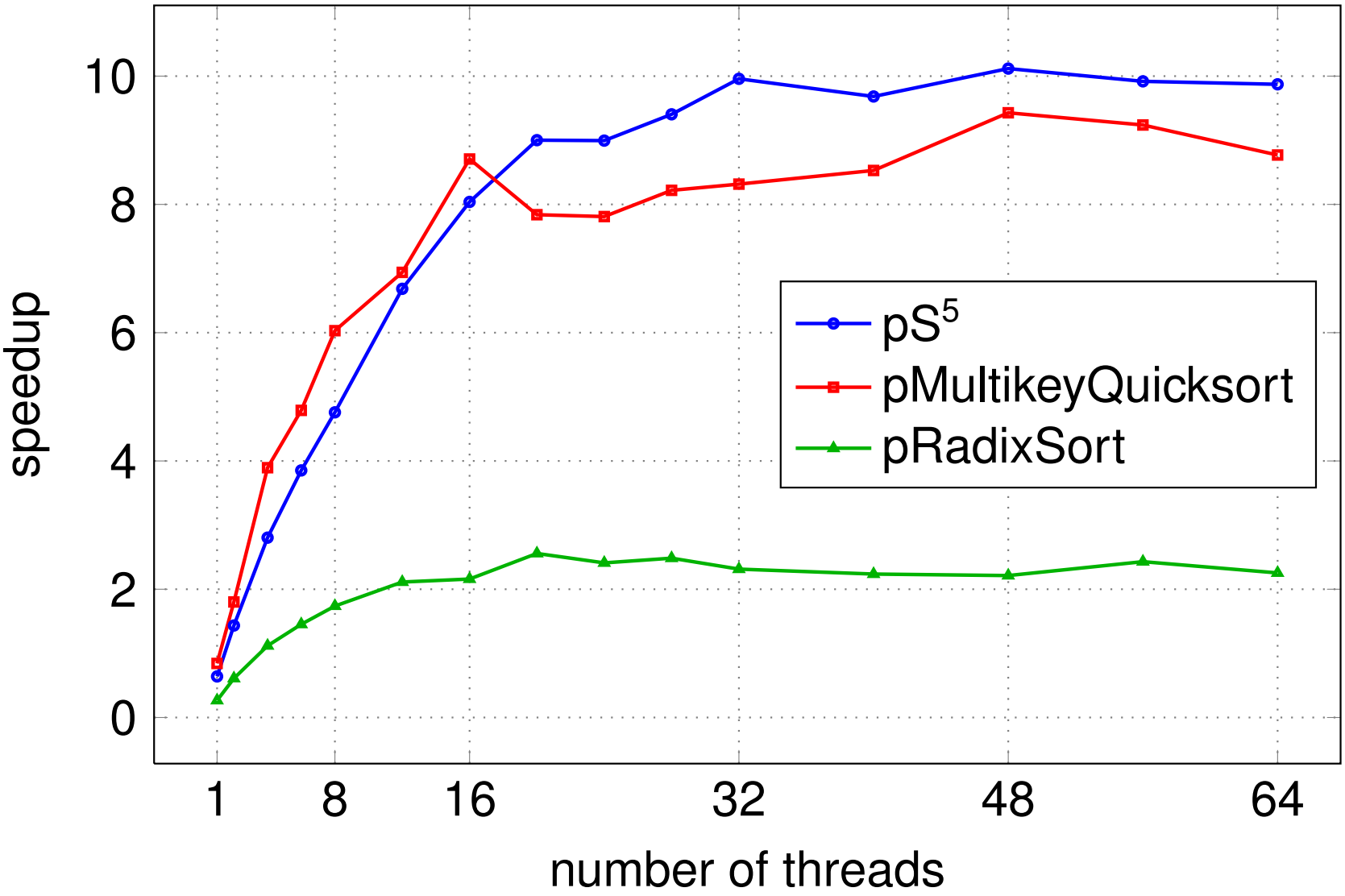
- $|\mathcal{S}| \geq \frac{n}{p}$ fully parallel S^5
- $\frac{n}{p} > |\mathcal{S}| \geq 2^{16}$ sequential S^5
- $2^{16} > |\mathcal{S}| \geq 64$ caching multikey quicksort
- $64 > |\mathcal{S}|$ insertion sort

Important: dynamic load balancing with voluntary work freeing

URLs – 1.1 G Lines, 70.7 GiB

http://algo2.iti.kit.edu/index.php
http://algo2.iti.kit.edu/english/index.php
http://algo2.iti.kit.edu/1483.php
http://algo2.iti.kit.edu/1484.php
http://www.kit.edu/
http://algo2.iti.kit.edu/286.php
http://algo2.iti.kit.edu/1294.php
http://algo2.iti.kit.edu/research.php
http://algo2.iti.kit.edu/publications.php
http://algo2.iti.kit.edu/members.php
http://algo2.iti.kit.edu/lehre.php
http://algo2.iti.kit.edu/1844.php
http://algo2.iti.kit.edu/294.php
http://algo2.iti.kit.edu/basic-toolbox-page.php
http://map.project-osrm.org?dest=49.0137004,8.419307&destname=%22Am%20Fa...
http://www.uni-karlsruhe.de/fs/Uni/info/campusplan/index.php?id=50.34
http://www.informatik.kit.edu/1158.php
http://algo2.iti.kit.edu/emailform.php?id=eea49c752e4c1329710cba2efae10511
http://algo2.iti.kit.edu/routeplanning.php
http://algo2.iti.kit.edu/sanders.php
http://www.mwk.baden-wuerttemberg.de/forschung/forschungsfoerderung/land...
http://algo2.iti.kit.edu/1996.php

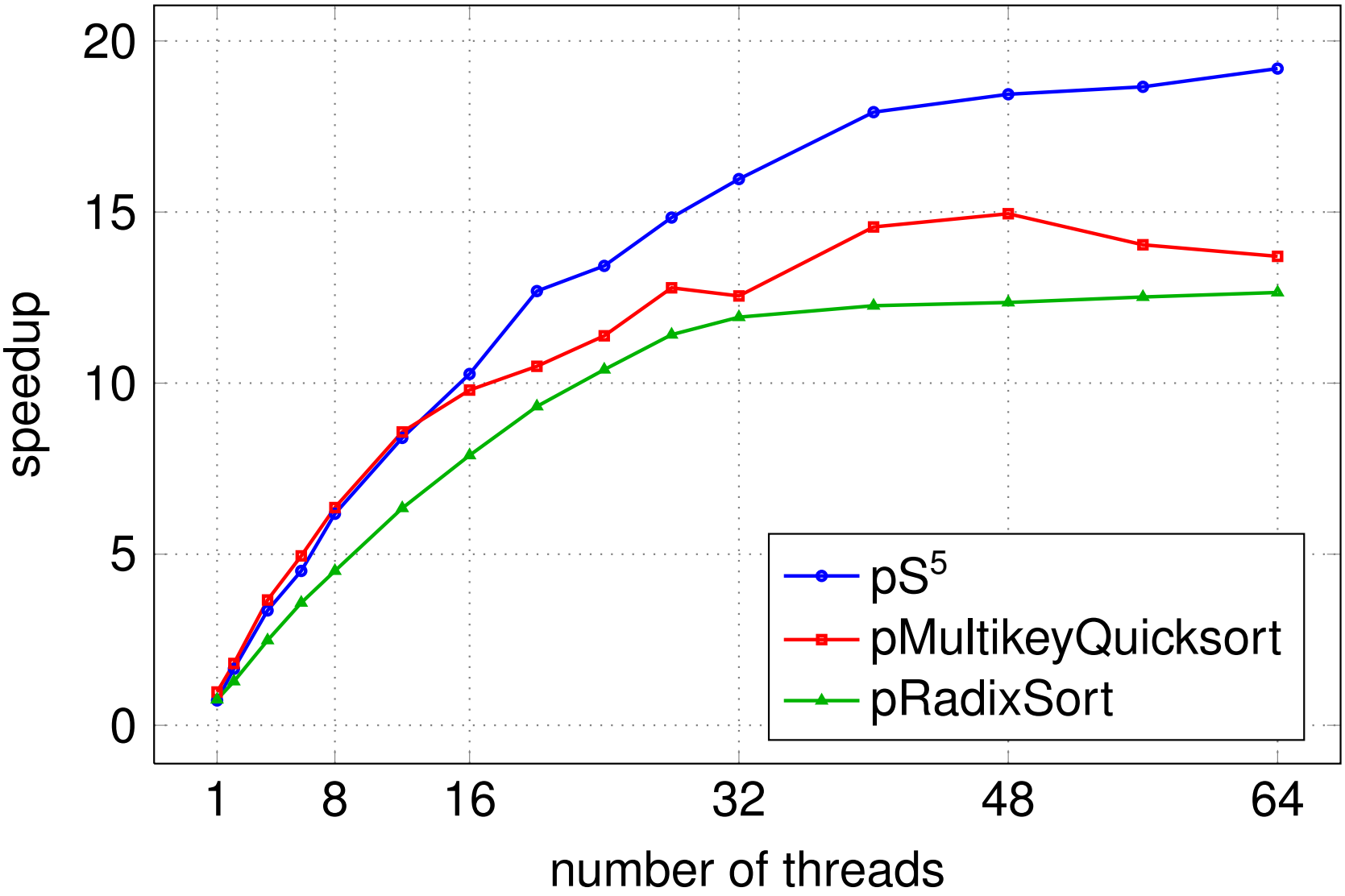
URLs – Speedup on 32-core Intel E5



Wikipedia – Suffixes of 4 GiB


```
<page>
  <title>Karlsruhe Institute of Technology</title>
  <ns>0</ns>
  <id>5977314</id>
  <revision>
    <id>491222814</id>
    <timestamp>2013-04-24T03:17:12Z</timestamp>
    <text xml:space="preserve">{{Infobox university
|name            = Karlsruhe Institute of Technology
|image = [[Image:KIT logo.png|270px|]]
|tagline         = ''{{lang|de|Forschungsuniversität}}'' (Research University)
|established      = Fridericiana: 1825 as polytechnical school, 1865 as university;&lt;br...
|president        = Horst Hippler, Eberhard Umbach
|students         = 23,905 (October 2012)
|staff           = 3.423<ref name="kit.edu"/>}}
The '''Karlsruhe Institute of Technology''' ('''KIT''') is one of the largest research and
educations institution in Germany, resulting from a merger of the university (''Universität
Karlsruhe (TH)'') and the research center (''Forschungszentrum Karlsruhe'')&lt;ref&gt;[[
Federal Ministry of Education and Research (Germany)]]: http://www.bmbf.de/pub/eck
punktepapier_kit.pdf&lt;/ref&gt; of the city of [[Karlsruhe]]. The university, also known
as '''''Fridericiana''''', was founded in 1825. In 2009, it merged with the former national
nuclear research center founded in 1956 as the ''Kernforschungszentrum Karlsruhe (KfK)''.
```

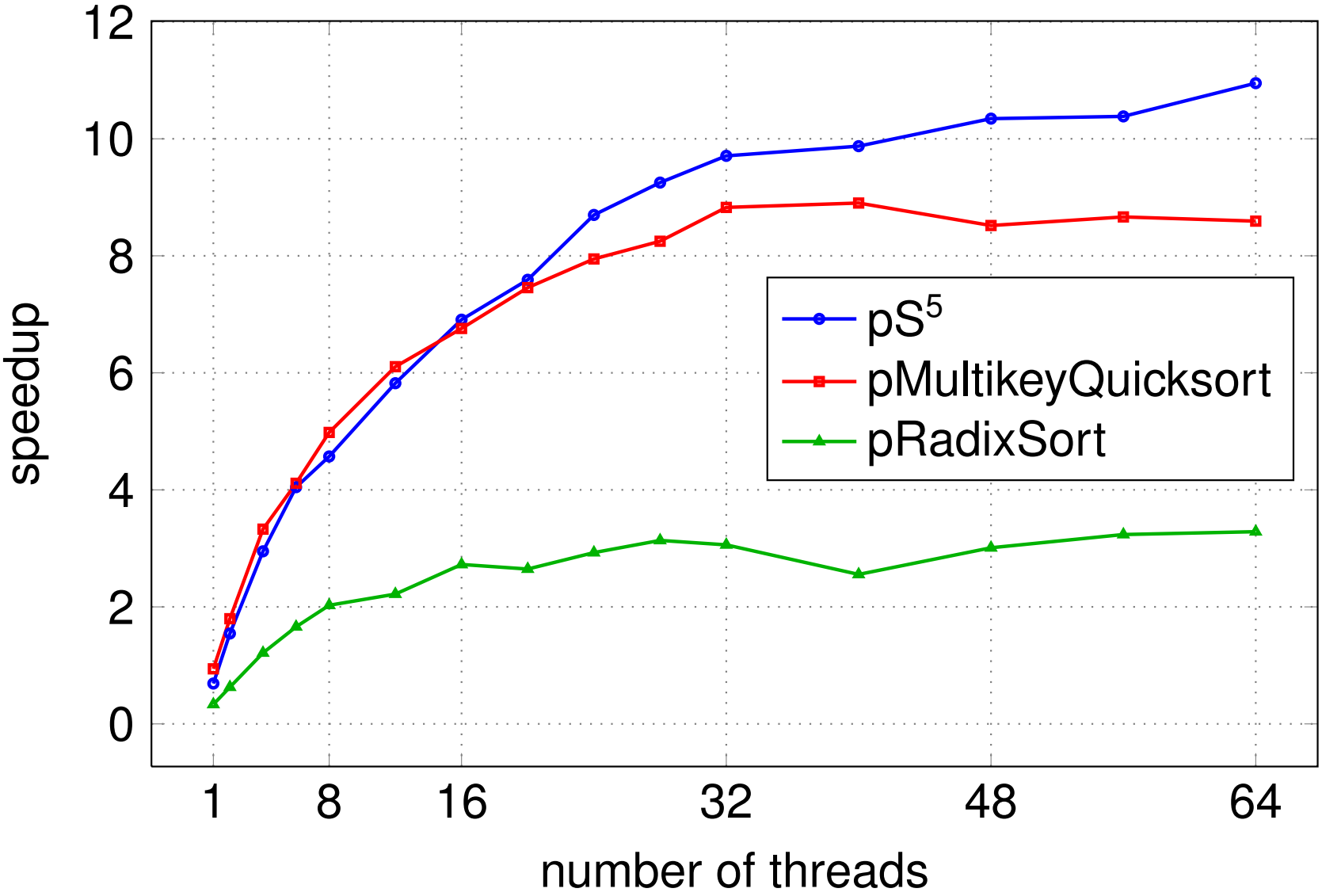
Suffixes – Speedup on 32-core Intel E5



GOV2 – 3.1 G Lines in 128 GiB

```
  <p align="left"><font face="Verdana, Arial, Helvetica, sans-serif"><b>BACKGROUND
    INFORMATION. </b>The Forest Ecosystem Dynamics (FED) Project is concerned
    with modeling and monitoring ecosystem processes and patterns in response
    to natural and anthropogenic effects. The project uses coupled ecosystem
    models and remote sensing models and measurements to predict and observe
    ecosystem change. The overall objective of the FED project is to link
    and use models of forest dynamics, soil processes, and canopy energetics
    to understand how ecosystem response to change affects patterns and processes
    in northern and boreal forests and to assess the implications for global
    change. See <a href="http://fedwww.gsfc.nasa.gov/html/conceptdiagram.html">
    Conceptual Diagram</a> for model schematic. </font></p>
  <p align="left"><font face="Verdana, Arial, Helvetica, sans-serif">The Forest
    Ecosystem Dynamics World-Wide-Web server has been online since July 1994.
    The FED server was created for the dissemination of project information,
    to archive numerous spatial and scientific data sets, and demonstrate
    the linking of ecosystem and remote sensing models. </font></p>
</td>
<td valign="top" width="28%" align="left" height="565">
  <p><img src="vertrule.gif" alt="vertical rule" width="2" height="580" hspace="5" ...
  <p><font size="2" face="Verdana, Arial, Helvetica, sans-serif"><a href="http://fedwww...
    Abstract</a> </font></p>
  <p><font face="Verdana, Arial, Helvetica, sans-serif" size="2"><a href="http://fedwww...
```

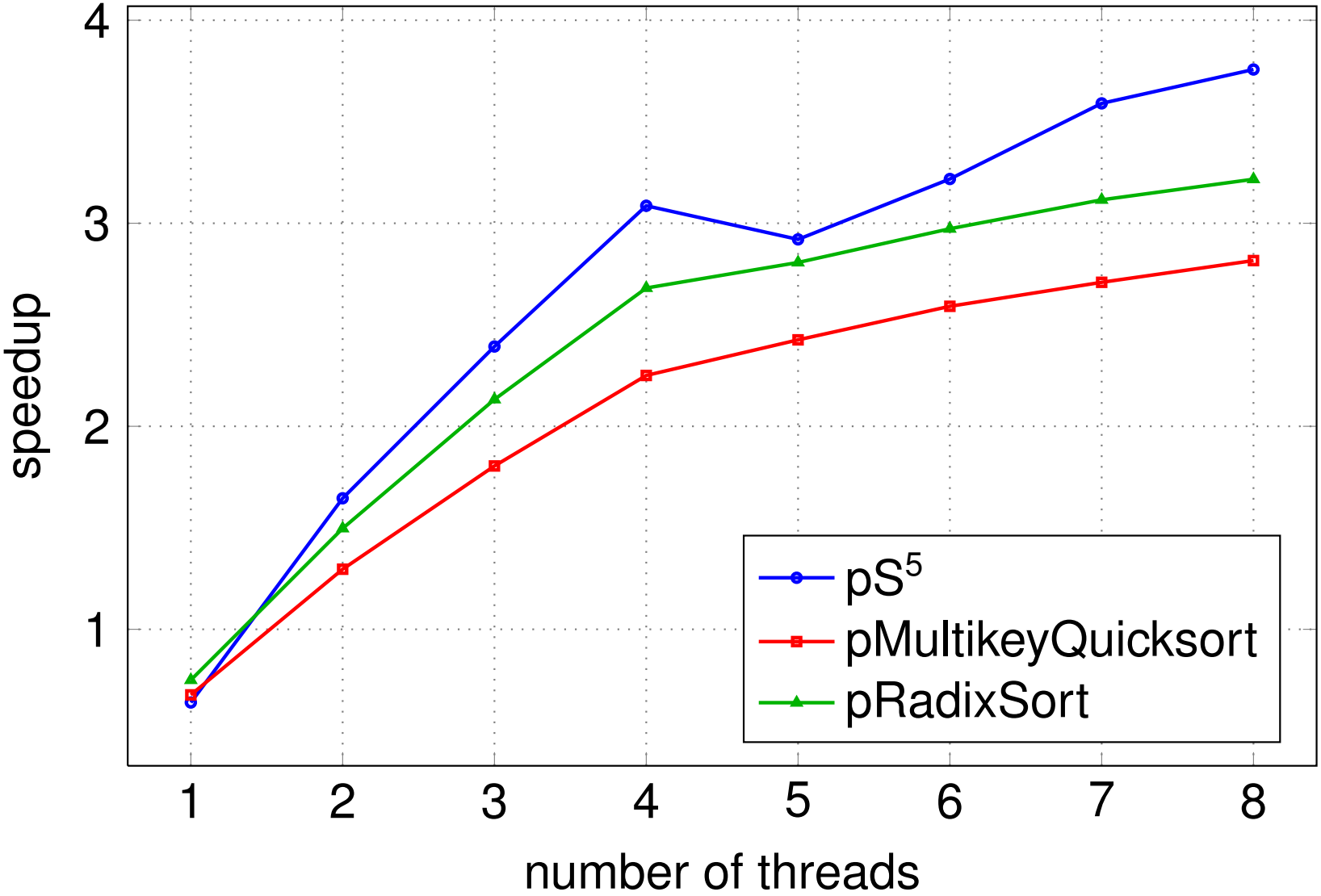
GOV2 – Speedup on 32-core Intel E5



Words – 31.6 M Lines, 382 MiB

stereosto	doesPGP	15Discussions
tocellular	PGPencrypt	aboutall
cellularand	encrypta	alltypes
andportable	amessage	typesof
wellas	messageWhat	computergraphics
asdiscussion	apublic	graphicsand
ofcomputer	publickey	languageconference
computersecurity	keyencrypted	forthe
securityissues	messagesthemselves	theexchange
issuesin	themselvesare	exchangeof
ingeneral	offtopic	ofideas
generalWhile	topicHost	ideasin
Whilewe	CamelotMessage	Learnmore
wewelcome	MessageFile	moreabout
welcomeall	FileBBS	theSpanish
allmanner	BBSCulver	Spanishculture
mannerof	CulverCity	culturewhile
ofquestion	CityCA	communicatingin
questionabout	ListGraphics	thelanguage
aboutencrypted	GraphicsGeneral	languageHost
encryptedmessages	GeneralGraphics	ListFantsySciF
messagesHow	Graphics15	FantsySciFFantasy

Words – Speedup on 4-core Intel i7



Future Work

- Non-uniform memory architecture (NUMA) effects
- distributed string sorting algorithms
- distributed text index construction and query
  - high-performance middleware?

Thank you for your attention!

Questions?