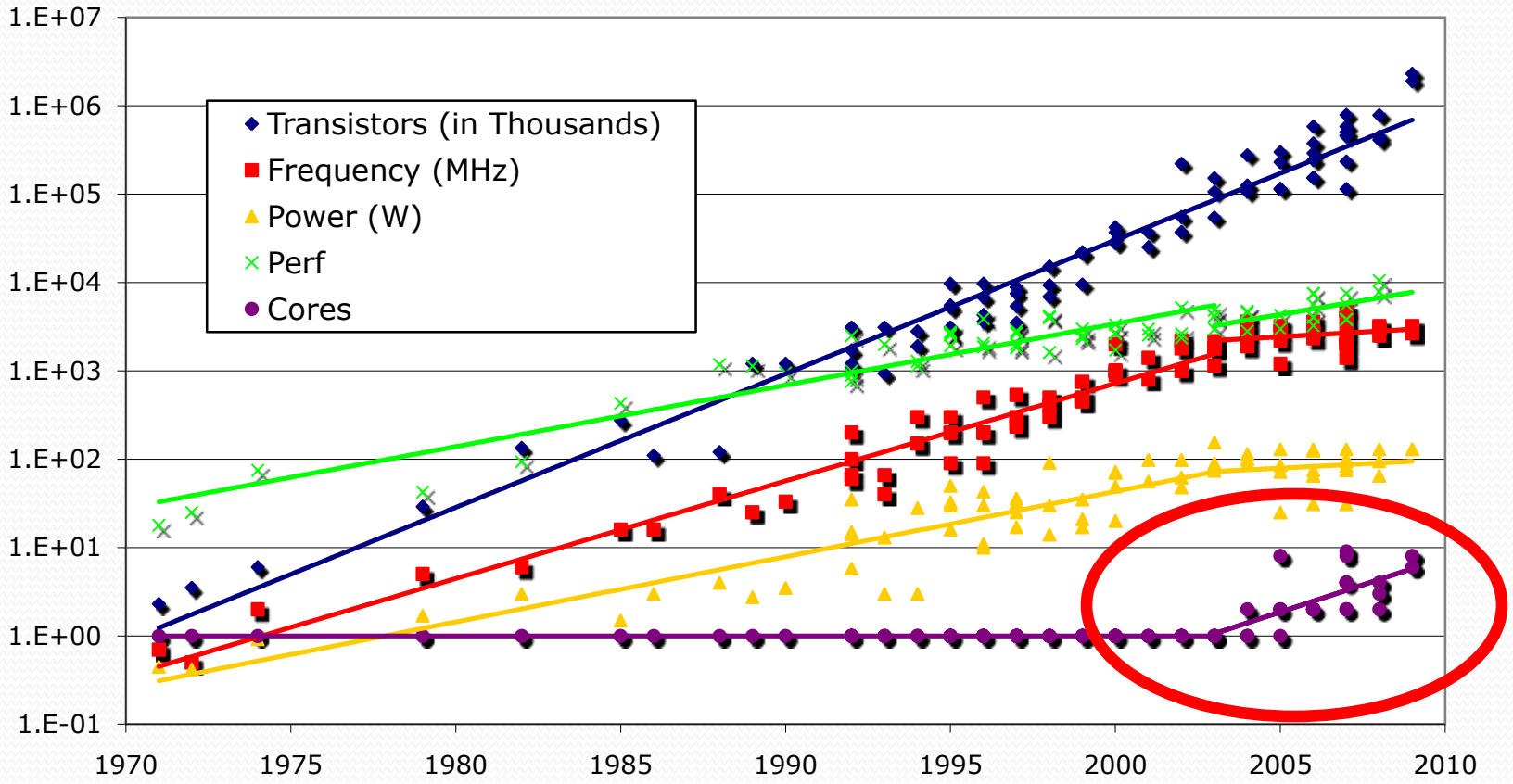
Cache-Conscious Concurrent Data Structures

Michael Spiegel

University of Virginia
Department of Computer Science

April 18, 2011

Motivation



Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanoviç. Courtesy of Kathy Yelick (ISCA '09 – Ten Ways to Waste a Parallel Computer).

Thesis

Cache-conscious concurrent data structures for many-core systems will show significant performance improvements over the state of the art in concurrent data structure designs for those applications that must contend with the memory wall.

Data Structure Properties

Data Structure	Lock-free	Cache-conscious	Randomized	Sorted
j.u.c. skip list	Y	N	Y	Y
Treap, Randomized search tree	N	N	Y	Y
B^{link}-tree	N	Y	N	Y
Cache-oblivious B-tree	N	N	Y[a]	Y
Lock-free skip tree	Y	Y	Y	Y

[a] randomized cache-oblivious B-tree algorithm does not support deletions

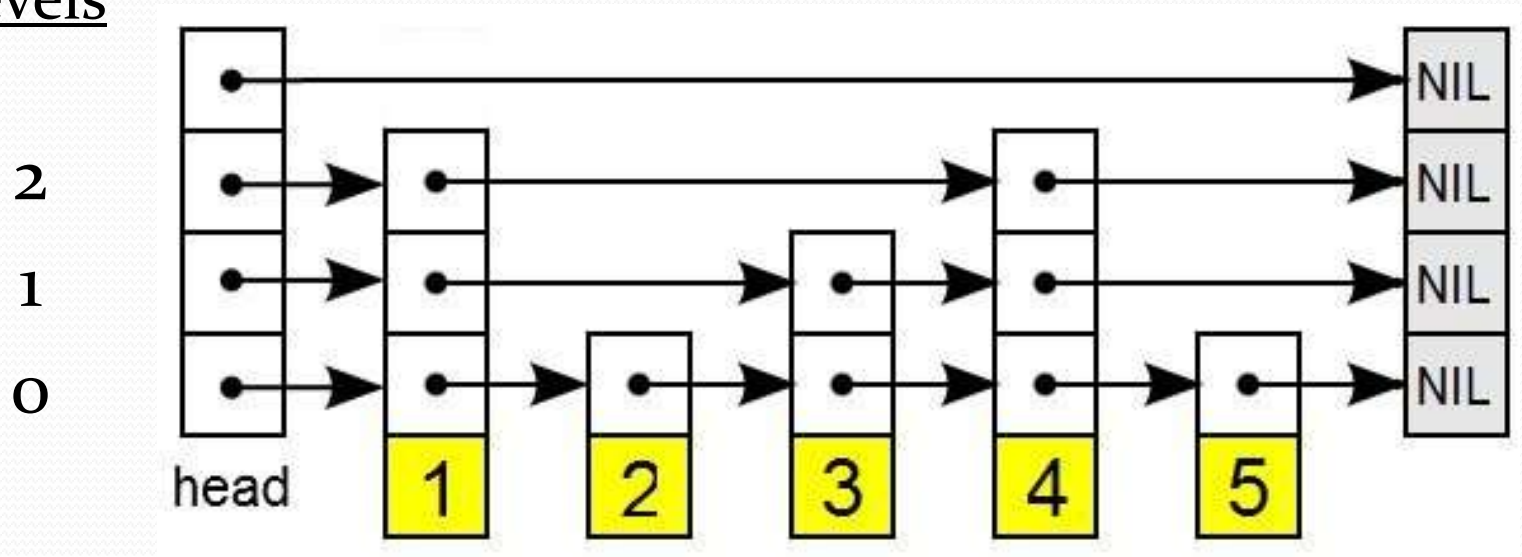
Agenda

- Motivation
- Thesis
- Summary of results & contributions
- Background & related work
- Dense skip tree definition & proofs
- Parallel branch & bound applications
- Synthetic branch & bound application
- Shared-memory supercomputer benchmarks
- Future work

Contributions

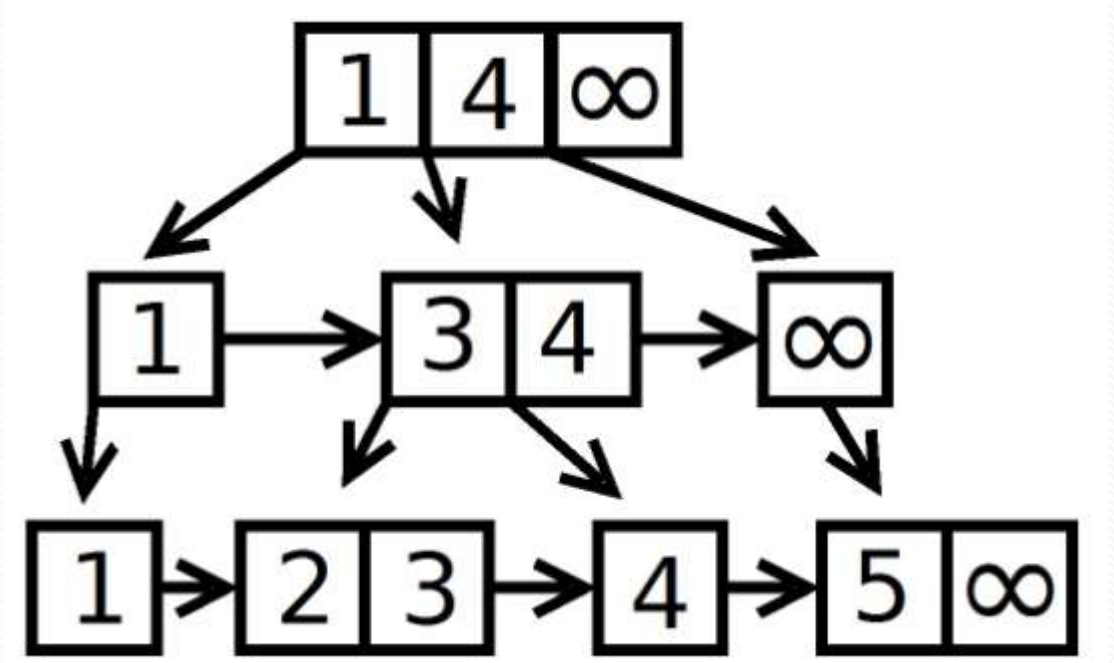
- Novel cache-conscious lock-free algorithms that implement an ordered set abstract data type.
- Prove cache-conscious structure of lock-free skip tree.
- Lock-free skip tree up to x2.3 faster in some workloads compared to lock-free skip list with only a 13% maximum penalty across all workloads.
- Set of guidelines for selecting the skip tree as a priority queue in a parallel branch-and-bound application.
- Submitting lock-free skip tree to JSR-166 for Java 8.

Skip List



Pugh, William (June 1990). "Skip lists: a probabilistic alternative to balanced trees." *Communications of the ACM*.

Dense Skip Tree



Contributions to Skip Tree

- Introduce “link” pointers to allow nodes to split independently of their parent nodes.
- Relax the requirement that non-leaf nodes behave as partitions.
- Optimal paths through the tree are temporarily violated by deletion operations, and eventually restored using online node compaction.
- Use empty nodes only for deletion operations.

Deriving Node Size Distribution

Given:

$\Pr(H = h) = q^h p$ where $p + q = 1$ and $Q = 1/q$

H is the random variable for the height of a key in the tree.

Show:

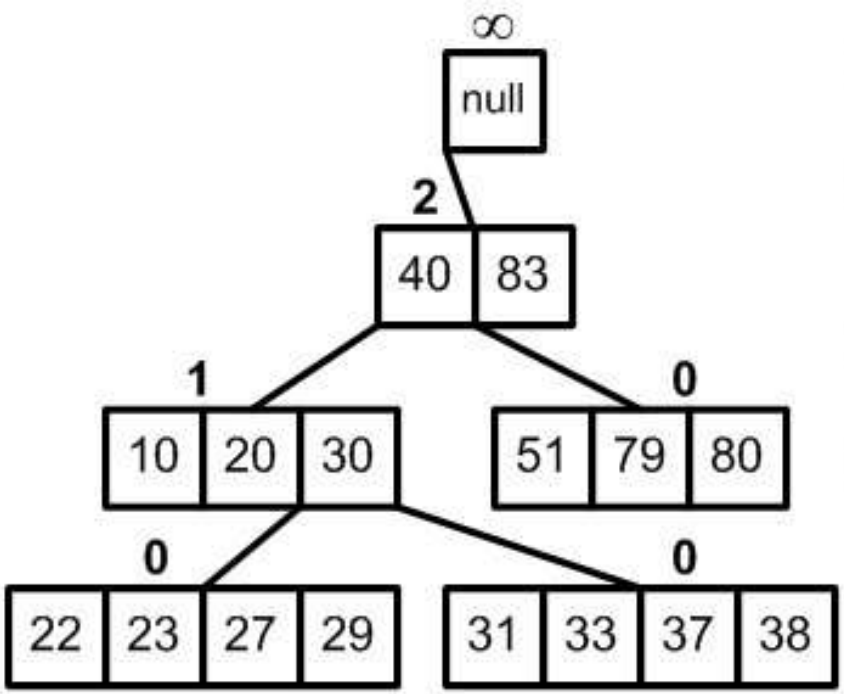
$\Pr(S = s) = p^{s-1} q$

S is the random variable for the number of keys in a node (the size of a node).

Proof Outline

- Begin with $\Pr(H = h) = q^h p$.
- Reduce $\Pr(S = s)$ to a one-player game:
  - Player starts with score of zero.
  - Let random variable X represent the final score.
  - At each turn, three outcomes:
    - continue the game with +1 score
    - continue the game without addition to score
    - terminate the game
- Identify the probability generating function of X.
- Identify the probability distribution of X.
- Use the distribution of X to determine S.

Key and Height Vectors



K_i	10	20	22	23	27	29	30	31
H_i	1	1	0	0	0	0	1	0
K_i	33	37	38	40	51	79	80	83
H_i	0	0	0	2	0	0	0	2

Agenda

- Motivation
- Thesis
- Summary of results & contributions
- Background & related work
- Dense skip tree definition & proofs
- Parallel branch & bound applications
- Synthetic branch & bound application
- Shared-memory supercomputer benchmarks
- Future work

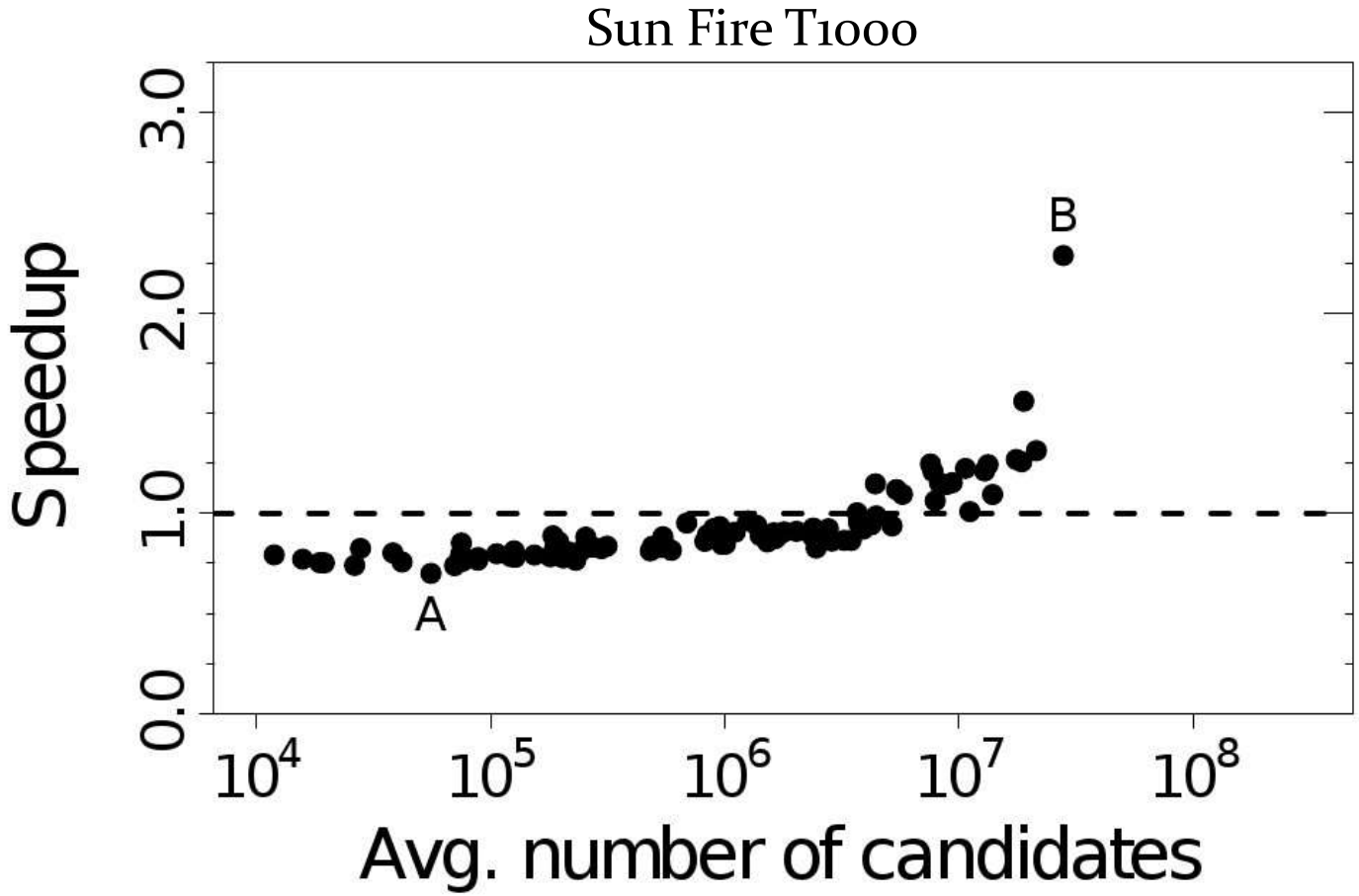
Evaluation Methodology

- Synthetic benchmarks
  - Execution consists solely of data structure operations
  - Application-neutral comparison
- Application benchmarks
  - Measure performance in a productive context
  - Difficult to compare across applications
- Measures of success
  - Throughput
  - Scalability
  - Memory footprint

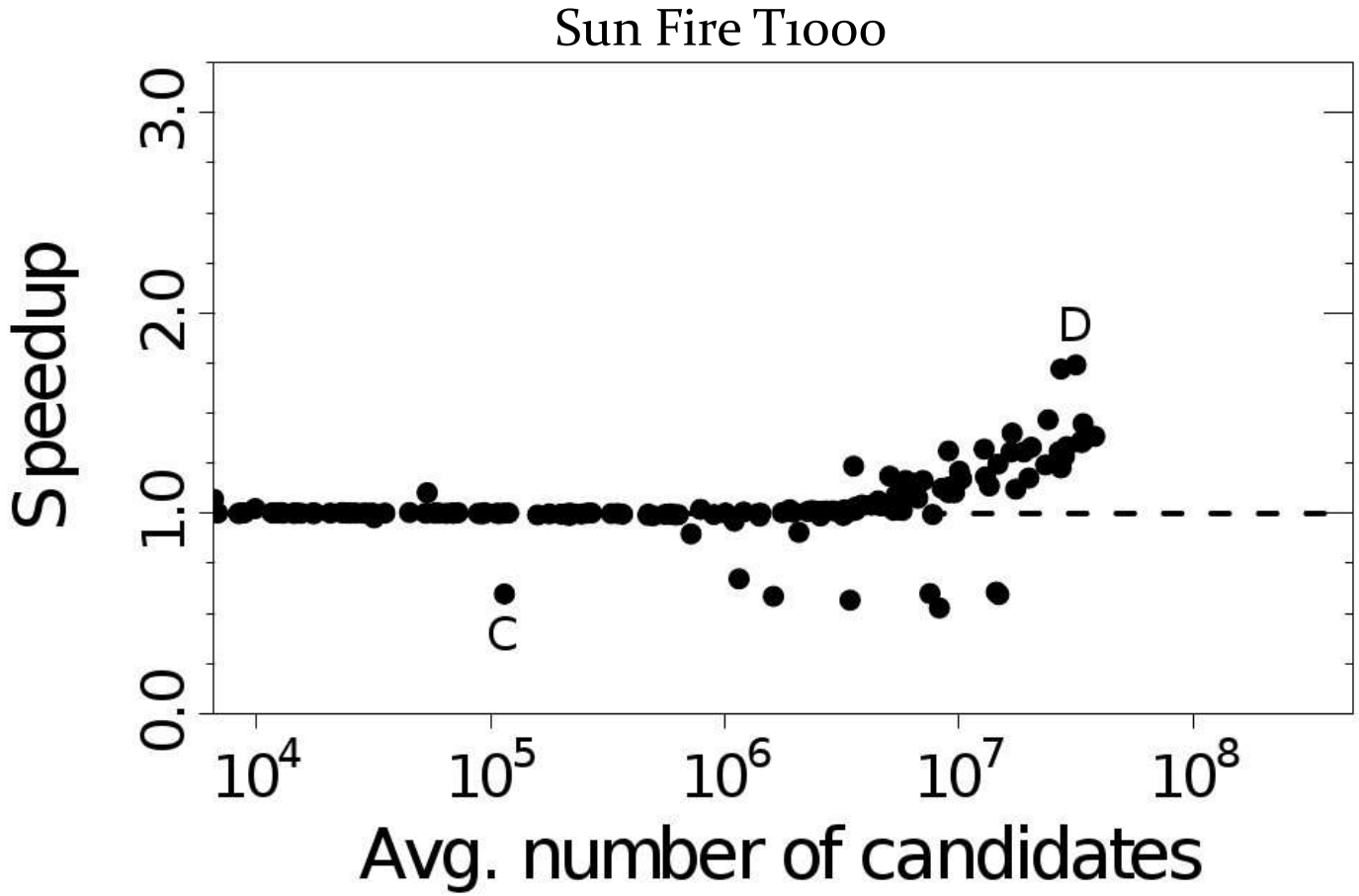
Parallel Branch & Bound (B&B)

- Given a solution space X and an objective function $f(X) \rightarrow \mathbb{R}$, find x* such that $f(x^*)=min\{f(x) \mid x \in X\}$
- $branch(S)=\{S_1, S_2, \ldots, S_n\}$
- if $lower(A)>upper(B)$, then A may be discarded from the search space.
- Four Applications:
  - 15 puzzle
  - Graph coloring
  - Asymmetric traveling salesperson problem
  - 0-1 Knapsack

Skip Tree Performance: 15 Puzzle



Skip Tree Performance: Graph Coloring



Application Phases

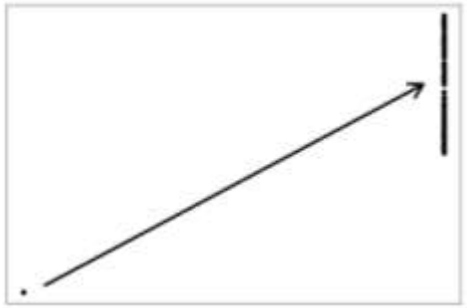
Application (input)	Extract min	Insert queue	Computation	Number of candidates
		— skip list —		
		— skip tree —		
15 puzzle (A)	4.8 ± 0.34	$44. \pm 2.4$	$51. \pm 1.5$	$5.9 \cdot 10^4 \pm 1.7 \cdot 10^3$
	4.3 ± 0.27	$76. \pm 6.2$	$52. \pm 3.6$	$5.7 \cdot 10^4 \pm 4.3 \cdot 10^3$
15 puzzle (B)	1.1 ± 0.23	$89. \pm 4.8$	9.7 ± 0.36	$2.8 \cdot 10^7 \pm 2.2 \cdot 10^3$
	0.49 ± 0.075	$38. \pm 8.3$	7.2 ± 0.18	$2.8 \cdot 10^7 \pm 1.8 \cdot 10^3$
Graph color (C)	2.7 ± 0.65	8.5 ± 1.6	$89. \pm 11.$	$1.2 \cdot 10^5 \pm 2.2 \cdot 10^4$
	5.8 ± 3.3	$75. \pm 21.$	$83. \pm 9.4$	$1.1 \cdot 10^5 \pm 2.4 \cdot 10^4$
Graph color (D)	0.95 ± 0.10	$62. \pm 2.4$	$37. \pm 0.37$	$3.2 \cdot 10^7 \pm 6.4 \cdot 10^2$
	0.54 ± 0.15	$28. \pm 2.4$	$42. \pm 0.085$	$3.2 \cdot 10^7 \pm 5.6 \cdot 10^2$
Asymmetric TSP	$0.0087 \pm 8.4 \cdot 10^{-5}$	0.11 ± 0.002	$99. \pm 0.11$	$1.0 \cdot 10^6 \pm 0$
	$0.0081 \pm 2.9 \cdot 10^{-4}$	0.16 ± 0.004	$99. \pm 0.06$	$1.0 \cdot 10^6 \pm 0$
0-1 Knapsack	$17. \pm 2.4$	$65. \pm 9.5$	$17. \pm 3.7$	$2.3 \cdot 10^7 \pm 0$
	$26. \pm 3.7$	$49. \pm 4.6$	$20. \pm 1.9$	$2.3 \cdot 10^7 \pm 0$

Table 1: Relative performance on Sun Fire T1000 (as percentage)

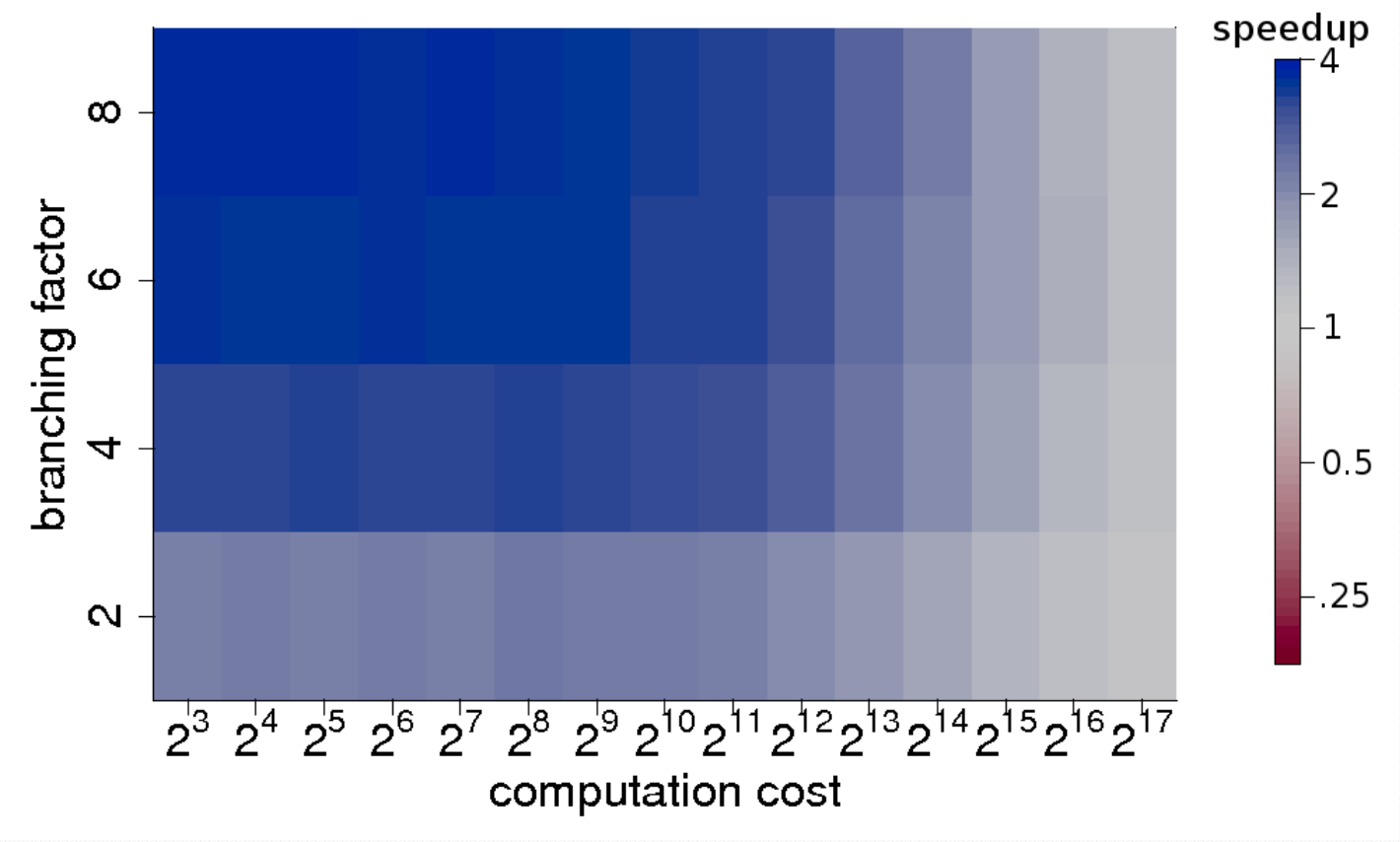
Synthetic Application

- Simplify a real branch & bound application to its essentials.
- Ask the following questions:
  - Does the distribution of lower bounds affect performance?
  - Does the computation time of the lower bounds affect performance?
  - Does the branching factor of the application affect performance?

Synthetic Application I

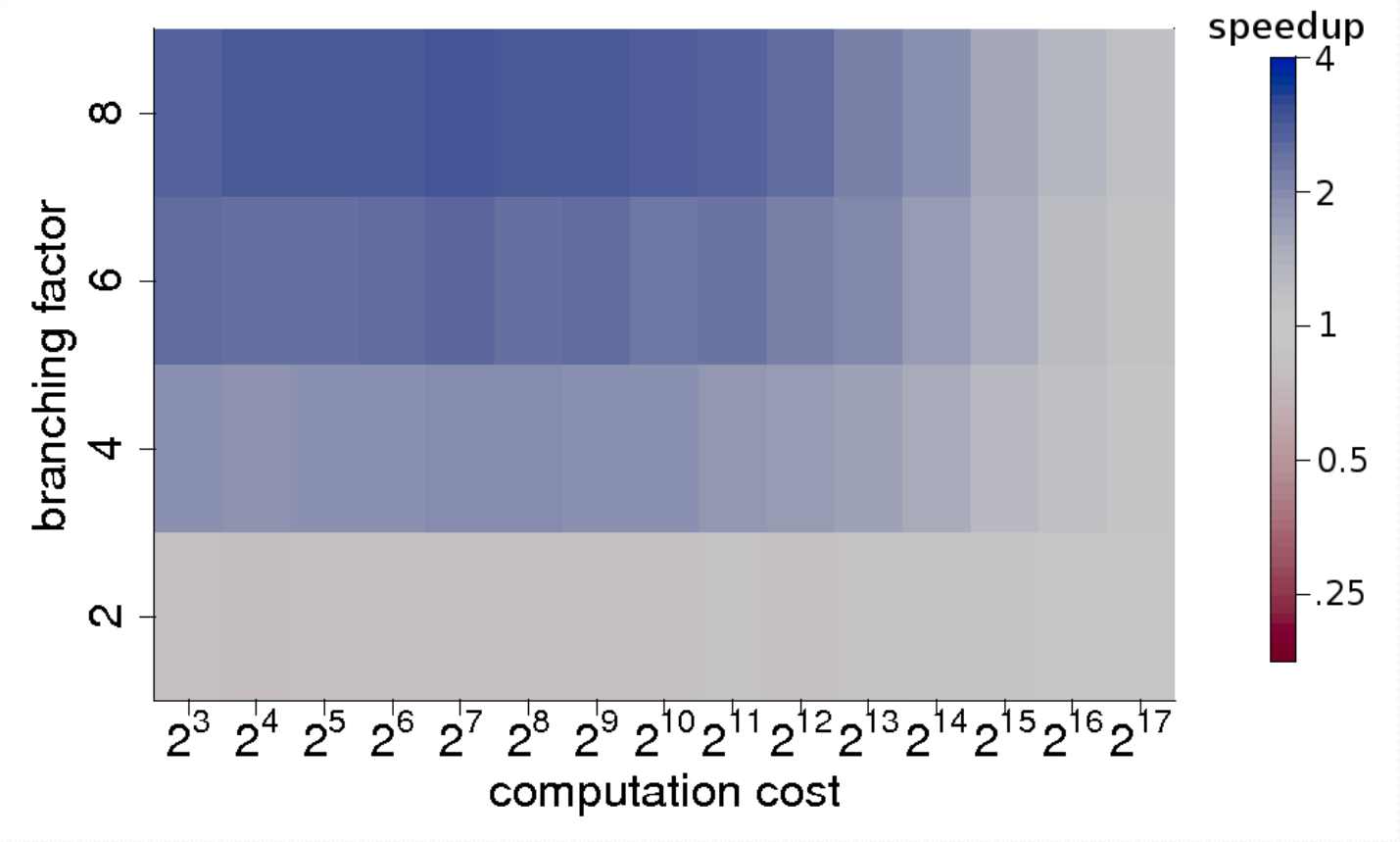


Monotonic Distribution



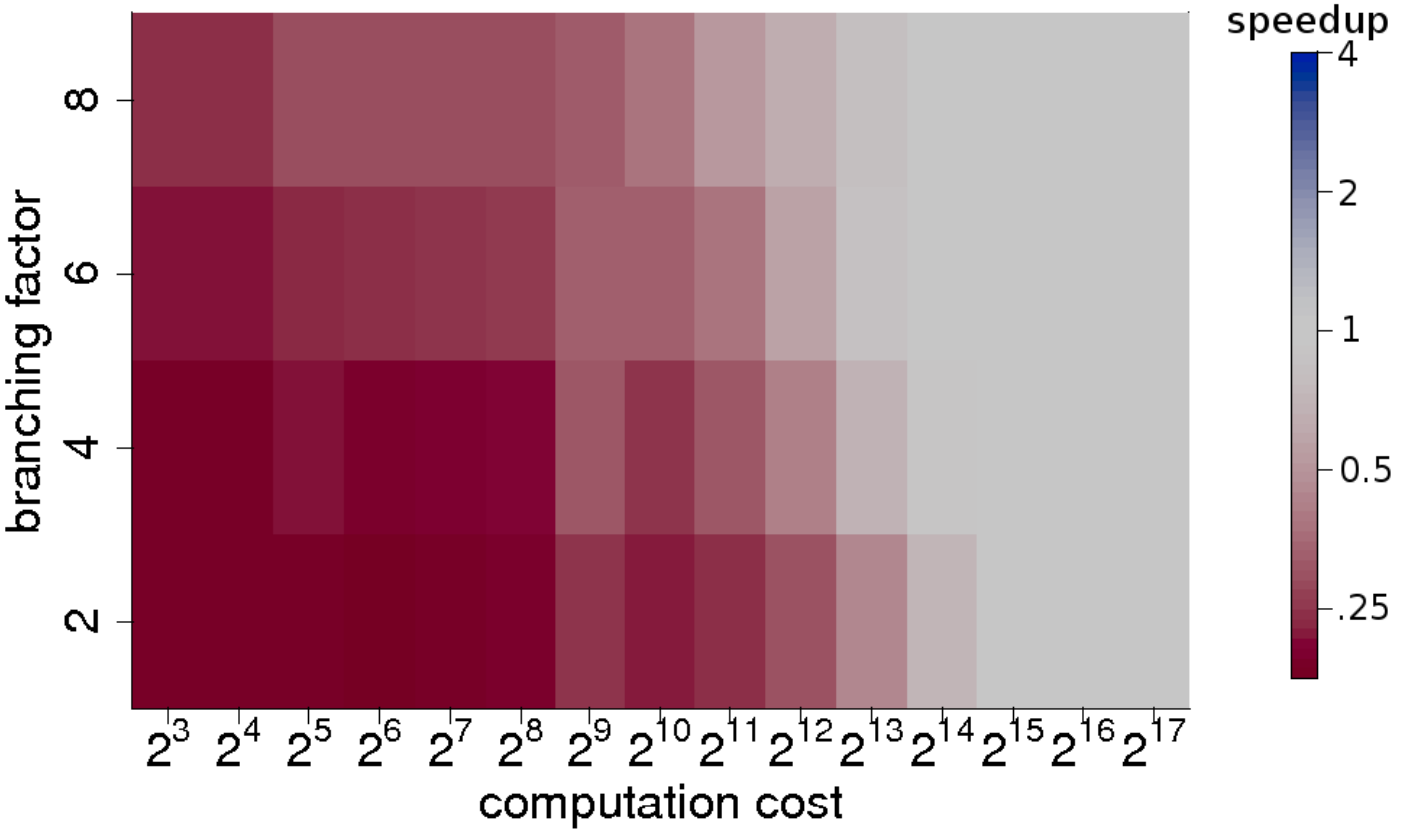
Synthetic Application II

Uniform Distribution



Synthetic Application III

Restricted Distribution

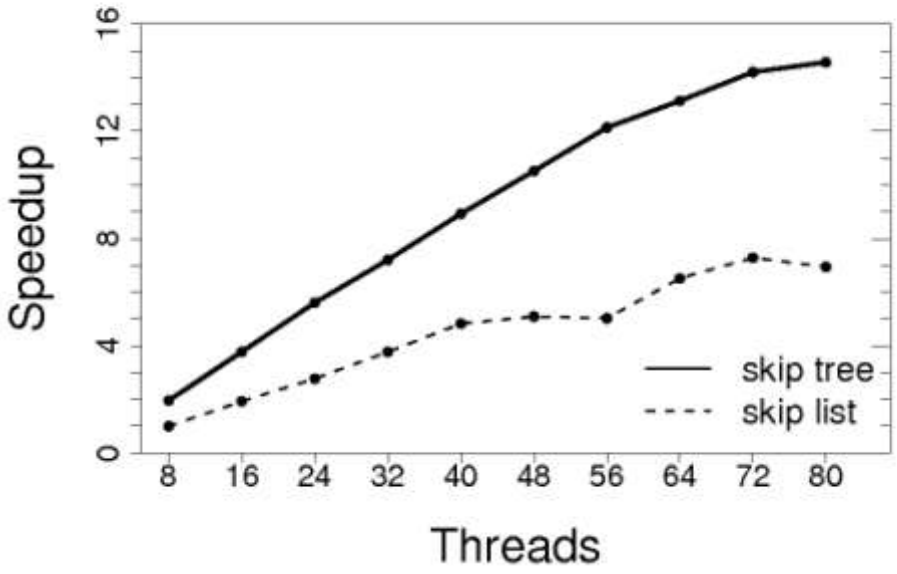


Azul Systems Supercomputer

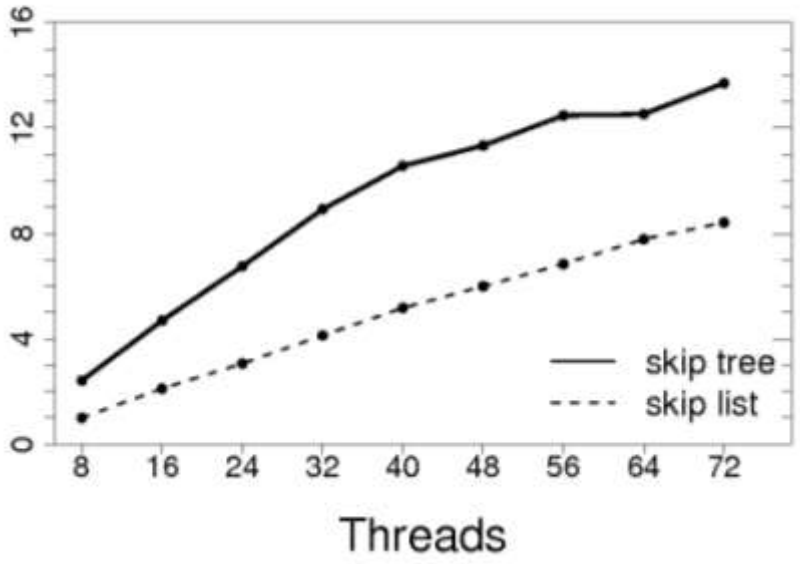
- RISC instruction set
- 54 core processor
- Up to 16 processors in appliance (864 cores)
- Instructions added for Java object allocation & garbage collection
- Custom Java VM based on OpenJDK
- Uniform memory access
- Pauseless GC, low latency, needs more heap

Azul Systems B&B Benchmarks

Graph Coloring



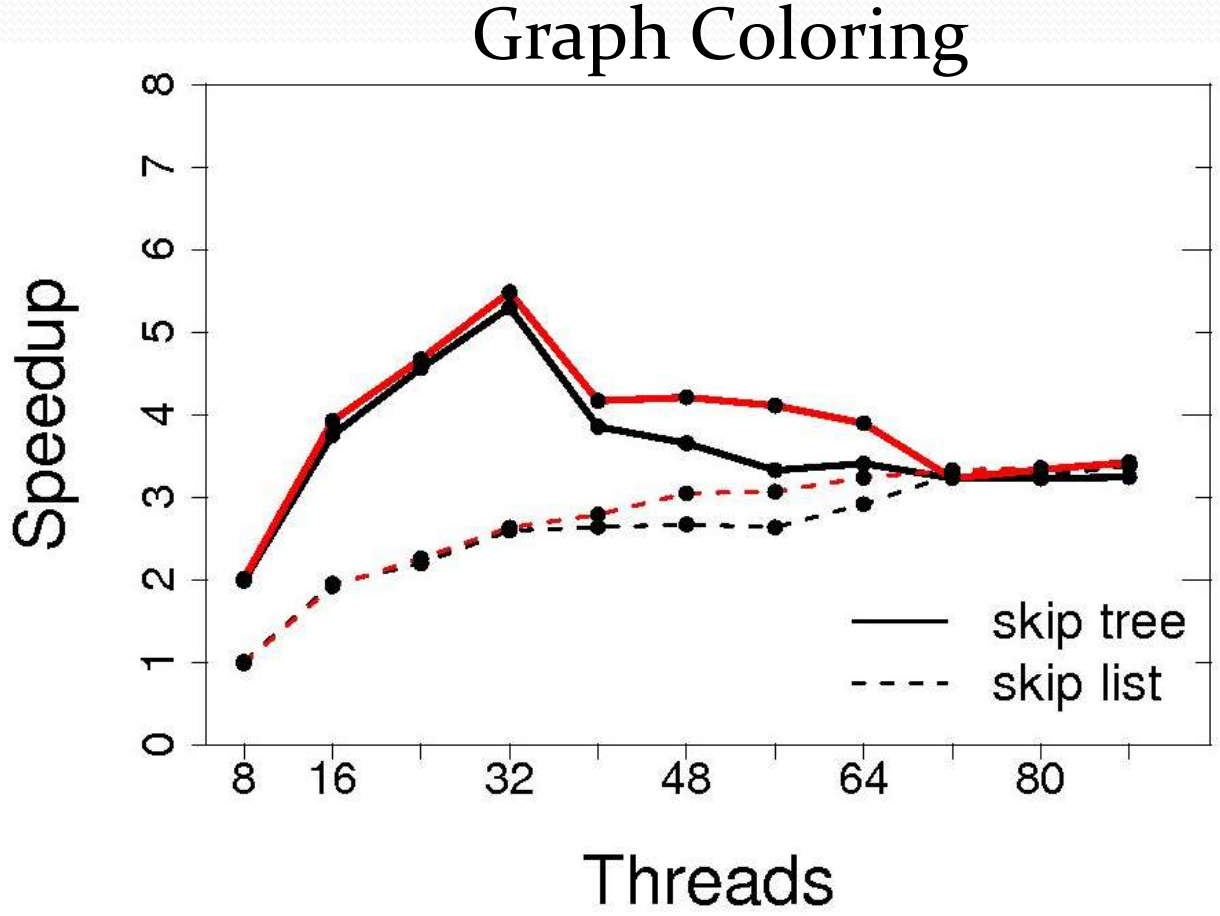
15 Puzzle



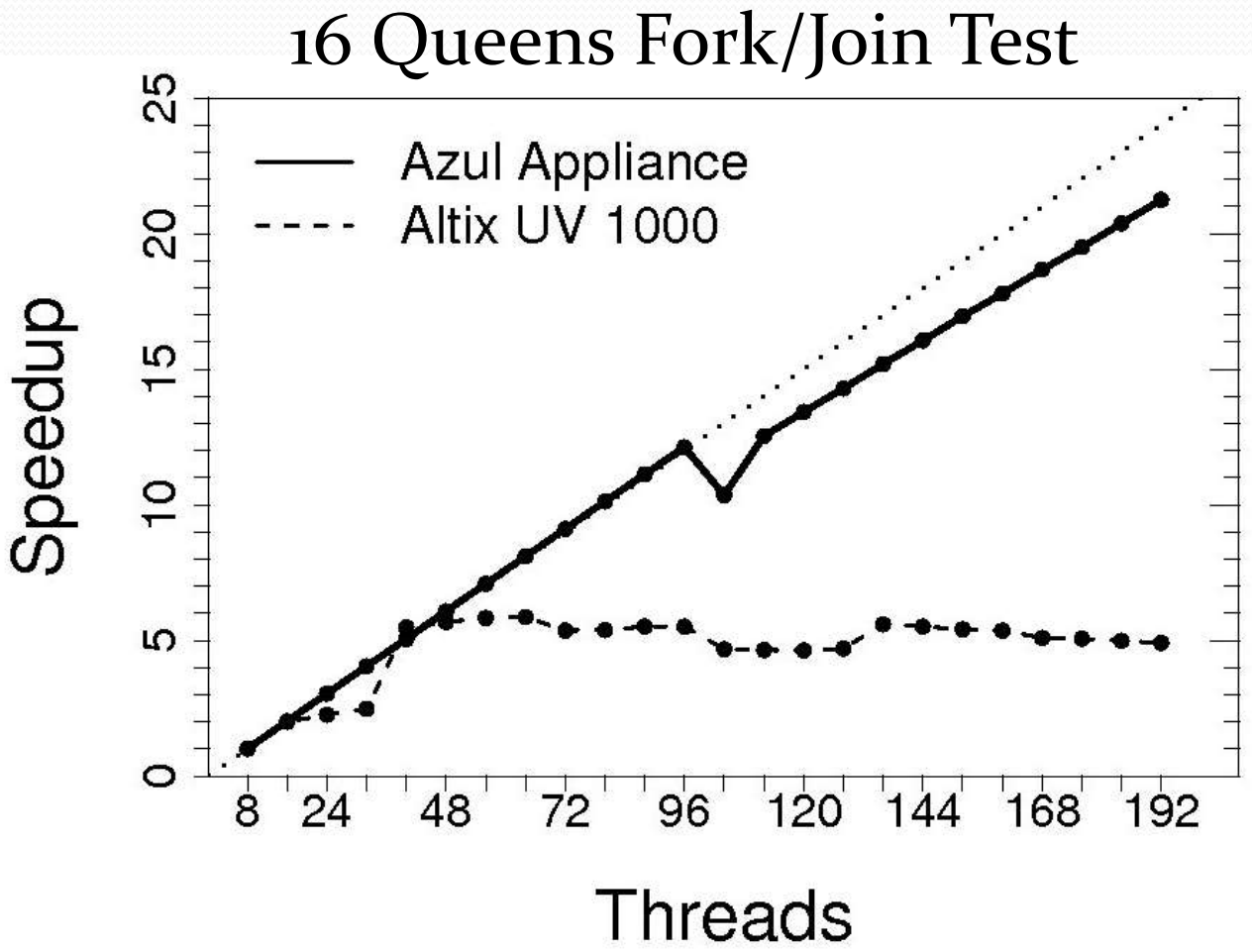
SGI Altix UV 1000

- Each blade is 2 Intel Xeon X7560 (Nehalem) 8-core processors.
- 256 blades (4096 processors)
- 16 cores on each blade share 128 GB of local memory.
- Up to 16 TB of shared memory.
- NUMAlink interconnect. Paired node 2D torus.
- SUSE Linux Enterprise Server 11.
- PBS batch scheduling. Linux cpusets interface to manage resources.

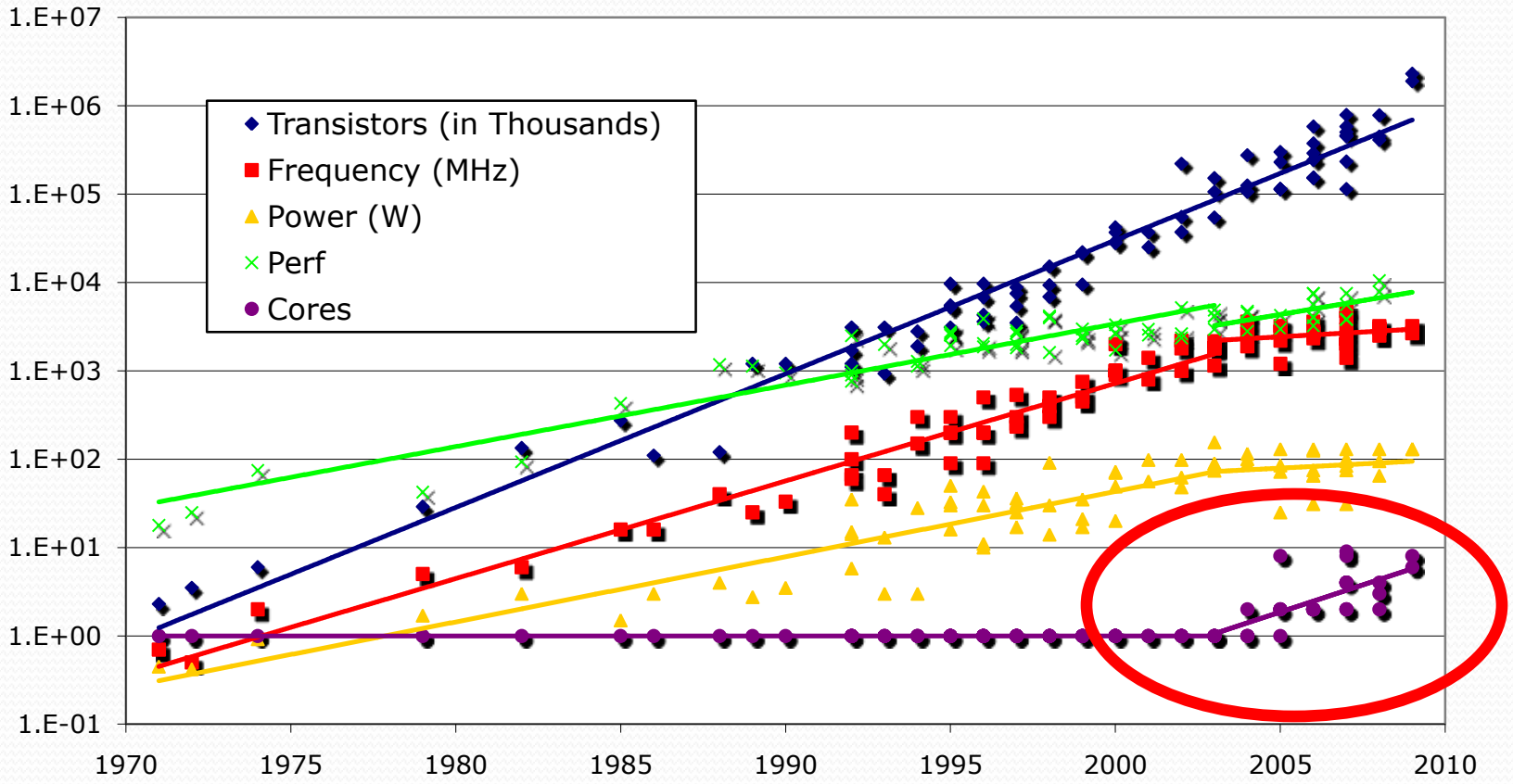
SGI Altix B&B Benchmarks



SGI Altix JVM Benchmarks

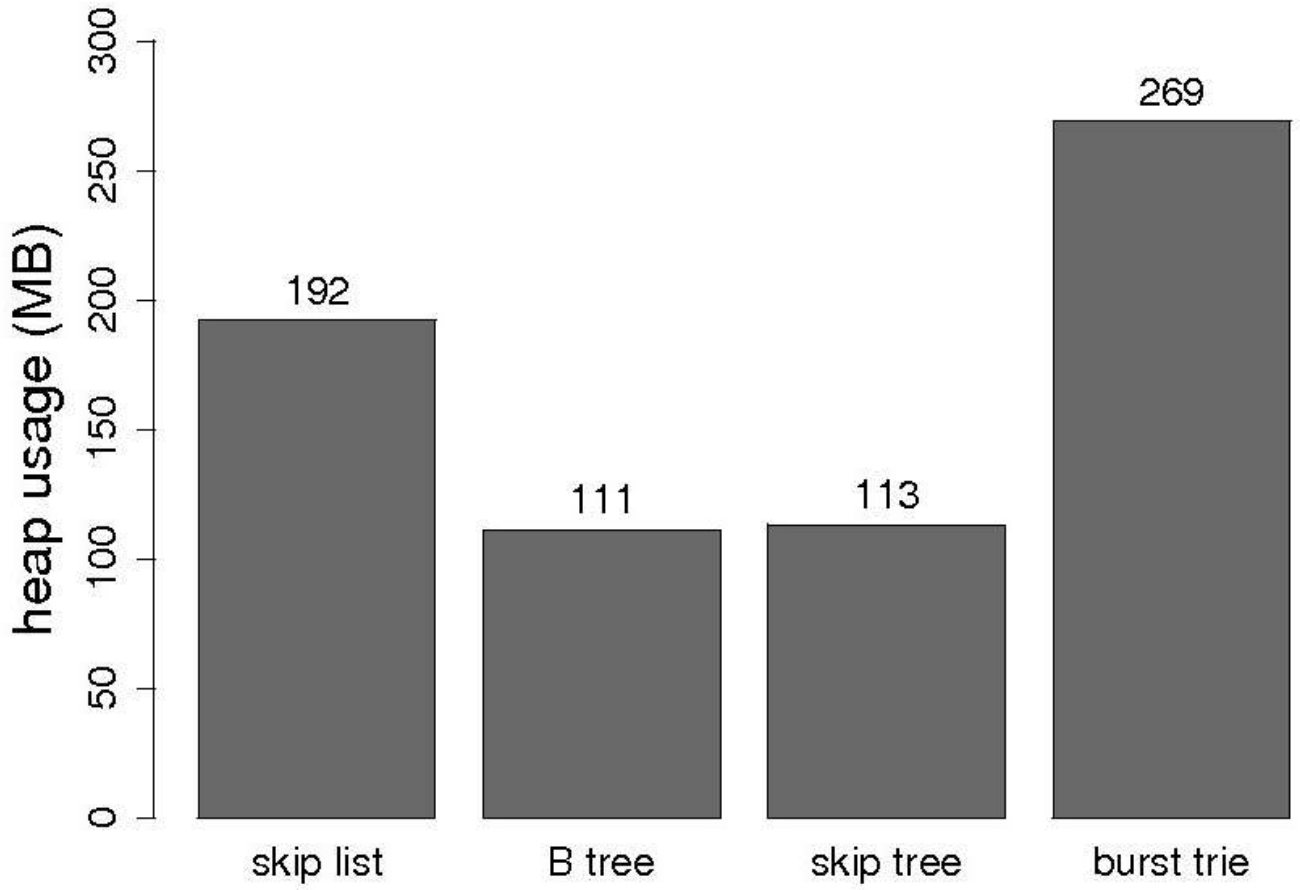


Scope



Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanoviç. Courtesy of Kathy Yelick (ISCA '09 – Ten Ways to Waste a Parallel Computer).

Heap Usage


heap usage (MB)
0
50
100
150
200
250
300
192
111
113
269
skip list
B tree
skip tree
burst trie

Contributions

- Novel cache-conscious lock-free algorithms that implement an ordered set abstract data type.
- Prove cache-conscious structure of lock-free skip tree.
- Lock-free skip tree up to x2.3 faster in some workloads compared to lock-free skip list with only a 13% maximum penalty across all workloads.
- Set of guidelines for selecting the skip tree as a priority queue in a parallel branch-and-bound application versus the skip list.
- Submitting lock-free skip tree to JSR-166 for Java 8.

Future Work

Near-term:

- Extend cache-conscious randomization techniques to other irregular computation patterns.
- One domain: graph processing algorithms.
- Post-doctoral assignment at Renaissance Computing Institute (www.renci.org).

Long-term:

- Three strategies for cache aware applications — in the algorithm, in the application library, or in the runtime.
- Runtime optimization is cheaper when cores are available.
- (Most) runtime IR specify *how* to compute, instead of *what* to compute.

Questions

[[[BACKUP SLIDES]]]

Consistency Models (I)

- Sequential consistency
  - Enforces program order within each individual processor.
  - Allows all processors to assume they are observing the same order of events.
- Quiescent consistency
  - Operations of any processors separated by a period of quiescence should appear to take effect in their real-time order.

Consistency Models (II)

- Linearizable consistency
  - All function calls have a linearization point at some instant between their invocation and their response.
  - All functions appear to occur instantly at their linearization point.
  - Weakest consistency model that preserves program order among individual processes and satisfies compositionality.

- Serializable consistency
  - An extension of sequential consistency. The result of any execution is the same as if all transactions were executed in some sequential order.

Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Source: Jeff Dean LADIS 2009 keynote

Mean Node Size

- (I) Increment State:
  $(H_i=h)$ and $(H_0,H_1,\ldots H_{i-1} \le h)$
- (N) Neutral State:
  $(H_i \le h)$ and $(H_0,H_1,\ldots H_{i-1} \le h)$
- (T) Terminating State:
  At least one of $H_0,H_1,\ldots H_i$ is > h.

$$A = \begin{array}{c} \\ I \\ N \\ T \end{array} \begin{array}{c} \begin{array}{ccc} I & N & T \end{array} \\ \begin{bmatrix} \Pr(H=h) & \Pr(H<h) & \Pr(H>h) \\ \Pr(H=h) & \Pr(H<h) & \Pr(H>h) \\ 0 & 0 & 1 \end{bmatrix} \end{array} = \begin{bmatrix} rp & 1-r & rq \\ rp & 1-r & rq \\ 0 & 0 & 1 \end{bmatrix} \quad \left(r = q^h\right)$$

Node Size Probability Distribution

- (I) The game continues with the addition of one point.

$$a = \Pr(\tau=\mathrm{I}) = \Pr(\mathrm{H}=\mathrm{h}) = q^h p$$

- (N) The game continues without addition to the score.

$$b = \Pr(\tau=\mathrm{N}) = \Pr(\mathrm{H}<\mathrm{h}) = 1-q^h$$

- (T) The game terminates.

$$c = \Pr(\tau=\mathrm{T}) = \Pr(\mathrm{H}>\mathrm{h}) = q^{h+1}$$

$$E(X) = a \cdot E(X \mid I) + b \cdot E(X \mid N) + c \cdot E(X \mid T)$$

$$g_X(z) = E(z^X)$$

$$g_X(z) = a \cdot E(z^X \mid I) + b \cdot E(z^X \mid N) + c \cdot E(z^X \mid T)$$

Probability Generating Function

$$E(z^X \mid I) = E(z^{X+1}) = zE(z^X)$$

$$E(z^X \mid N) = E(z^X)$$

$$E(z^X \mid T) = \sum_{x=0}^{\infty} z^x \Pr(X = x \mid T) = (z^0)(1) + \sum_{x=1}^{\infty} (z^x)(0) = 1$$

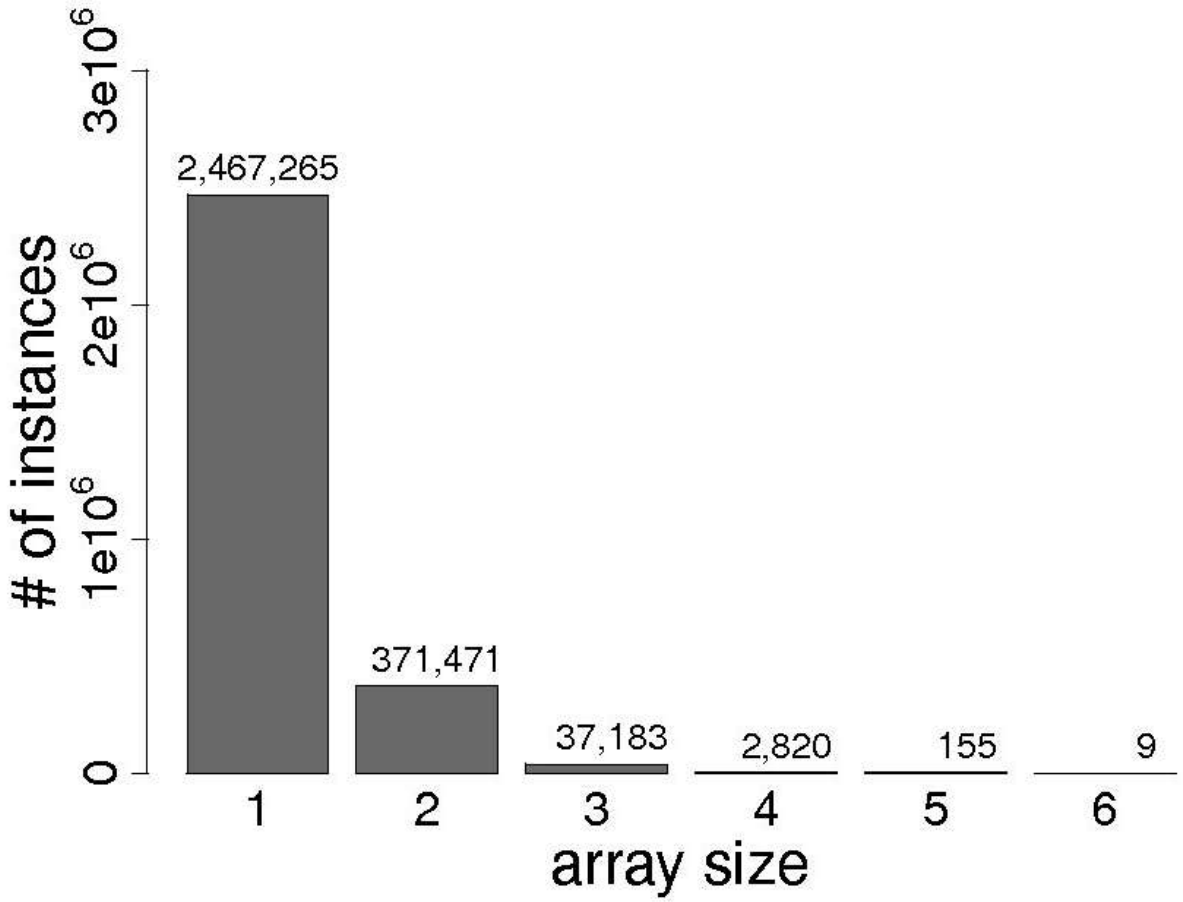
$$g_X(z) = a \cdot z g_X(z) + b g_X(z) + c$$

$$= \frac{c}{1-b}\left(1 - \frac{za}{1-b}\right)^{-1} = \frac{c}{1-b}\sum_{k=0}^{\infty}\left(\frac{za}{1-b}\right)^k$$

$$\Pr(X = x) = \frac{c}{1-b}\left(\frac{a}{1-b}\right)^x = p^x q$$

$$\Pr(S = s) = \Pr(X = s-1) = p^{s-1} q$$

Trie Node Distribution



Fixing the Runtime

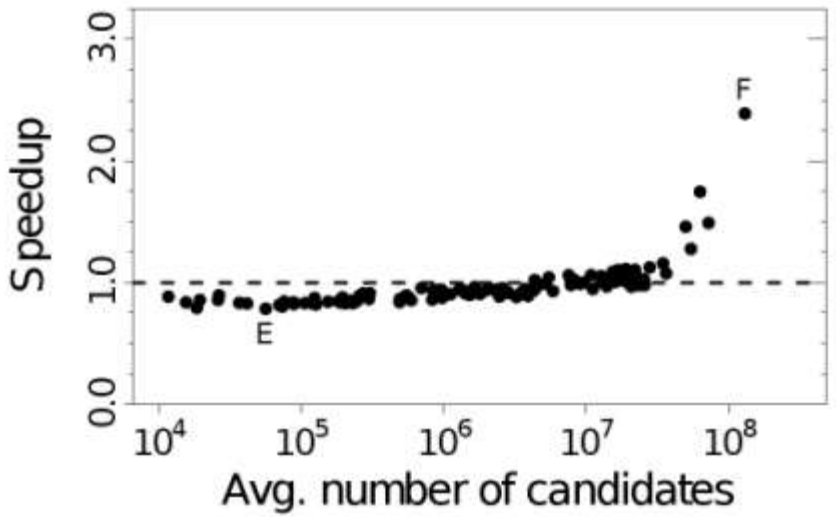
- HotSpot JVM on Linux is not cpuset-aware.
  - 5 line patch submitted to the OpenJDK project.
- HotSpot NUMA garbage collection on Linux is not cpuset-aware. Requires moving to libnuma 1.2 API from older libnuma 1.1 API.
  - 200 line patch to the OpenJDK project.
- libnuma bug in cpuset interface.
  - 3 line patch accepted in libnuma 2.0.6-rc4.

Related Data Structures

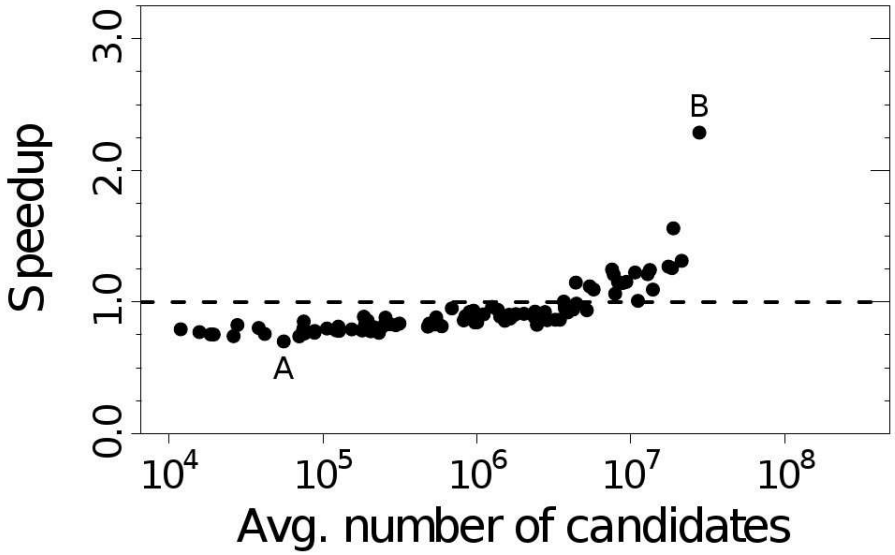
- Lock-free skip list in the java.util.concurrent library. Written by Doug Lea and the JCP JSR-166 Expert Group.
- Optimistic relaxed-balance AVL tree [Bronson et. al., 2010].
- B^{link}-tree [Lehman and Yao, 1981; Sagiv, 1985].

Skip Tree Performance: 15 Puzzle

Intel Quad-core Xeon

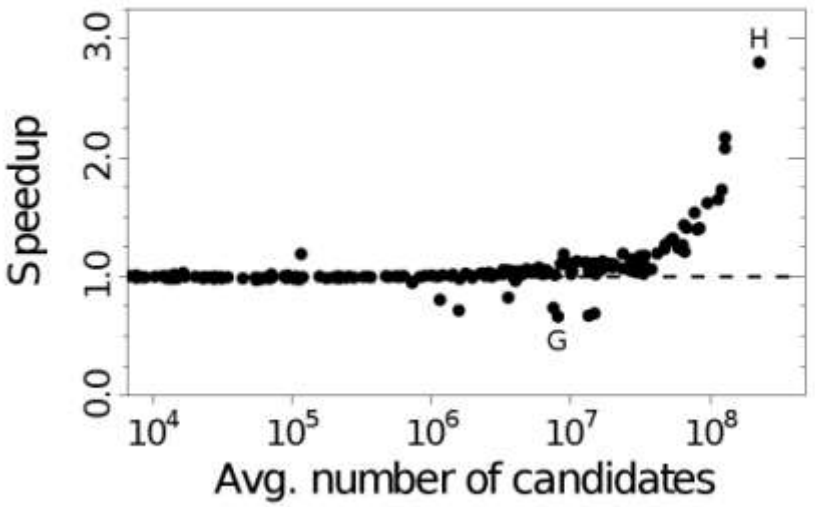


Sun Fire T1000

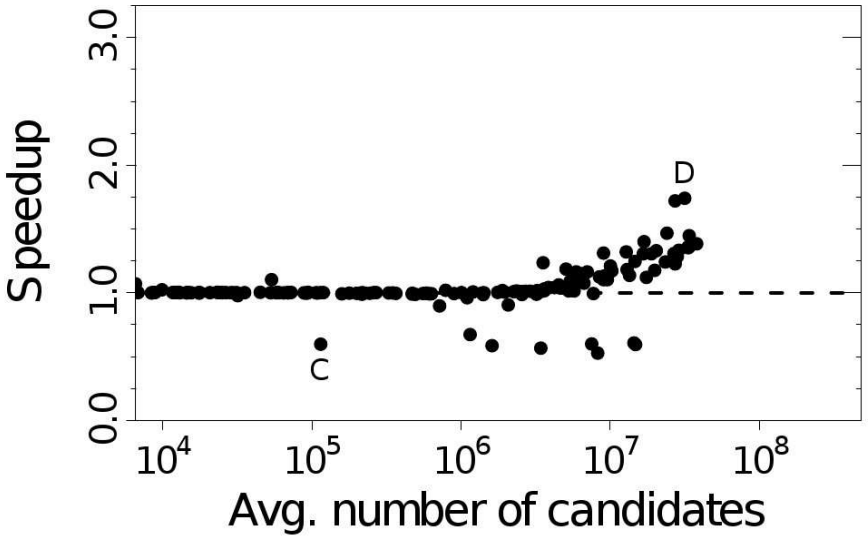


Skip Tree Performance: Graph Coloring

Intel Quad-core Xeon

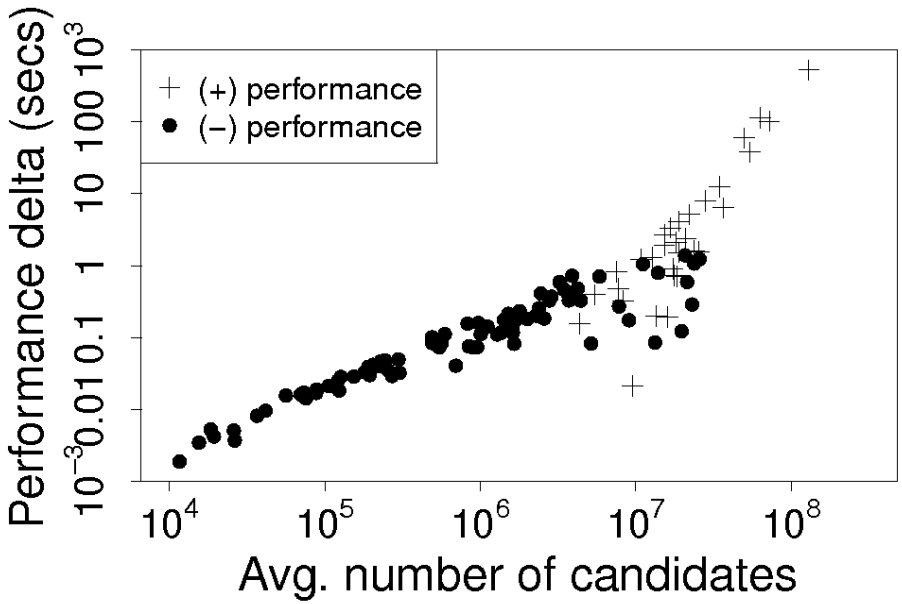


Sun Fire T1000

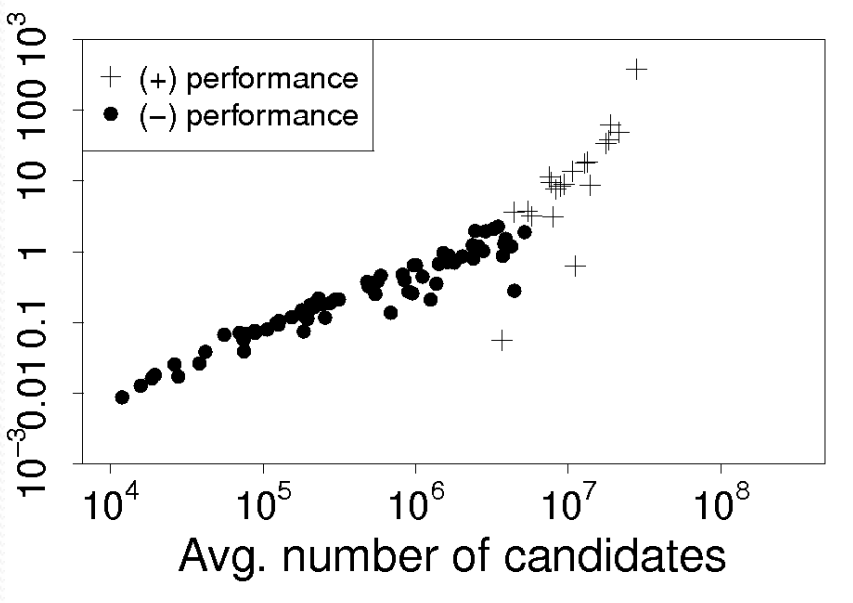


Skip Tree Absolute Speedup: 15 Puzzle

Intel Quad-core Xeon

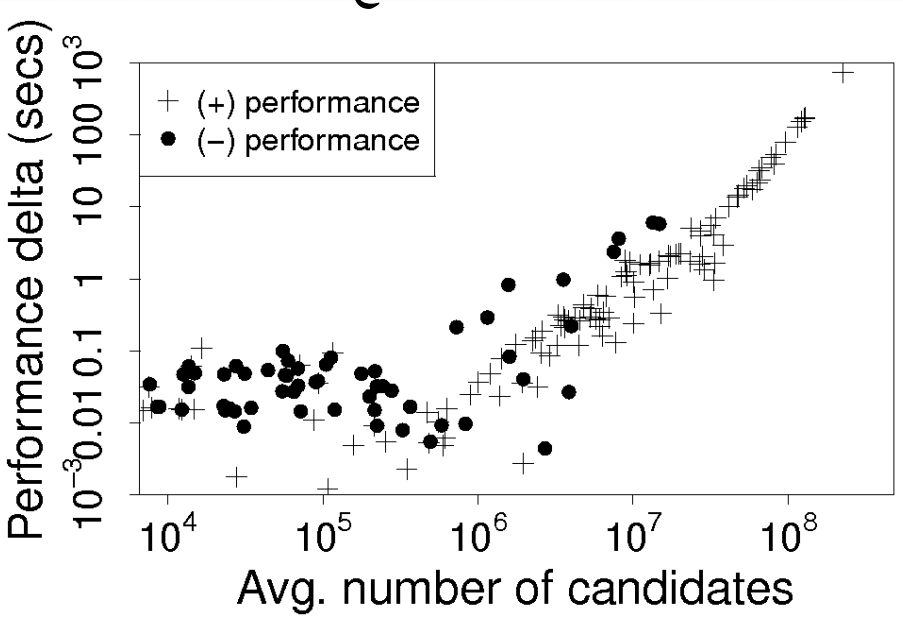


Sun Fire T1000

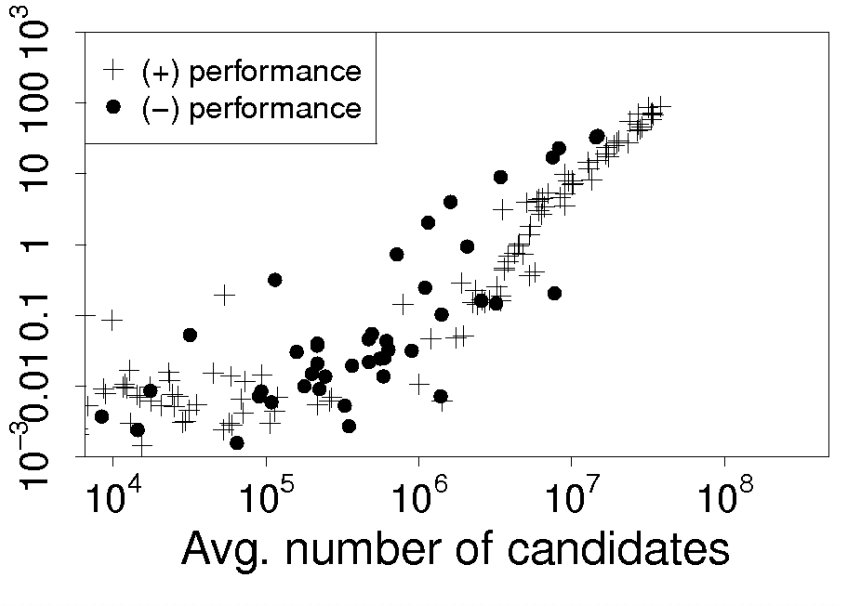


Skip Tree Absolute Speedup: Graph Coloring

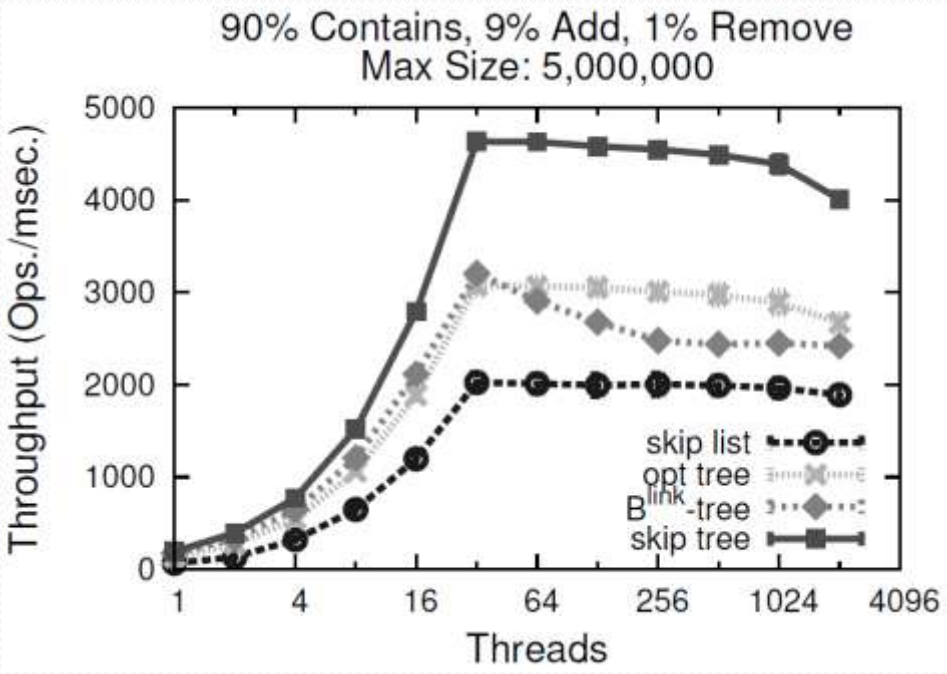
Intel Quad-core Xeon

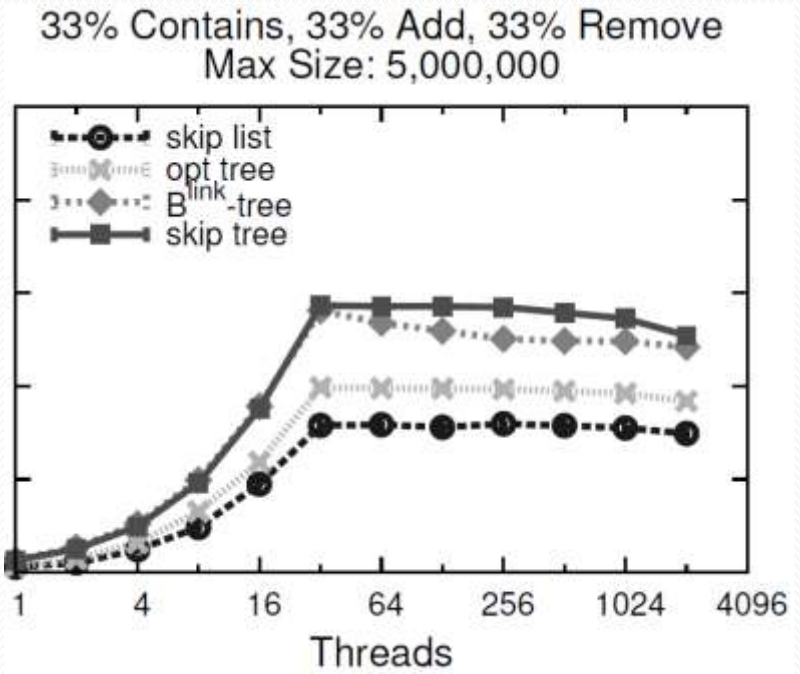


Sun Fire T1000

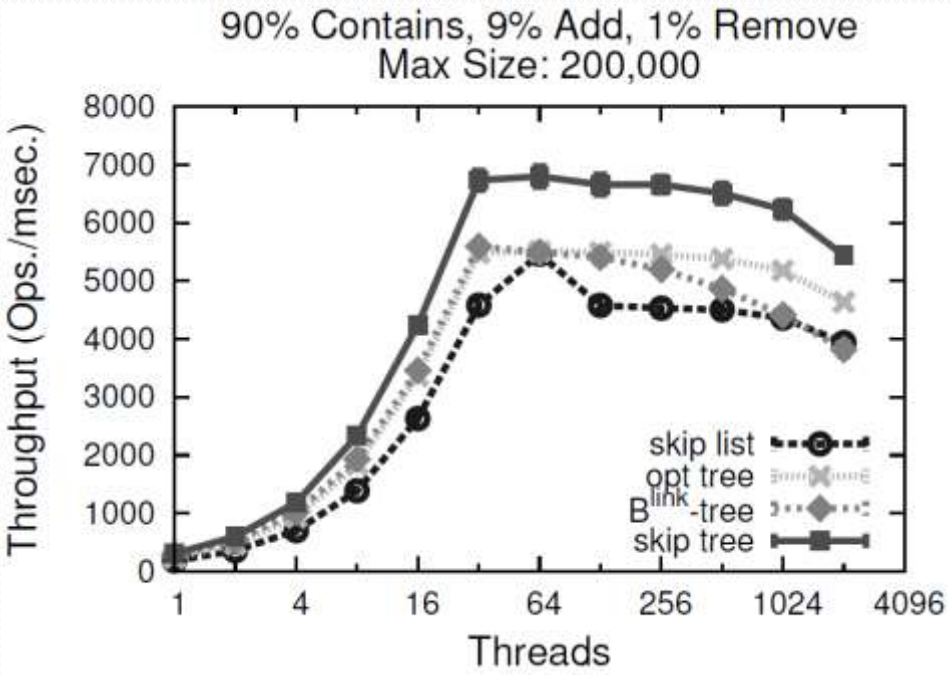


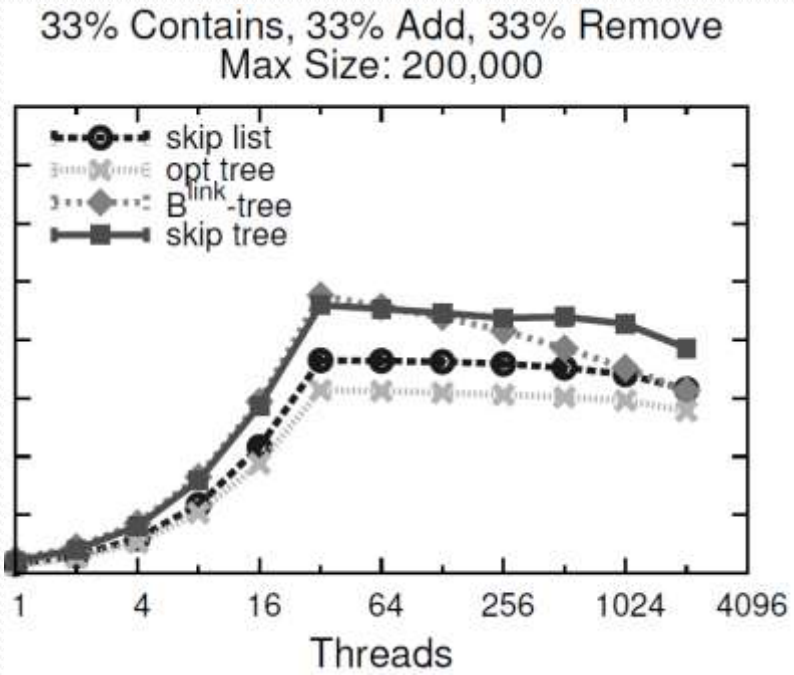
Synthetic Benchmarks - I



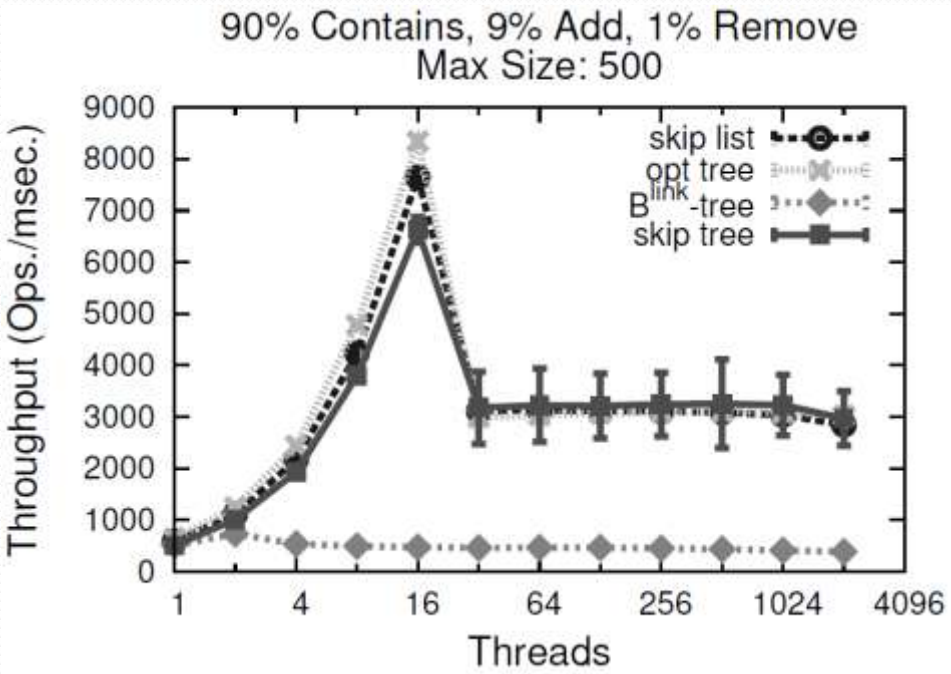


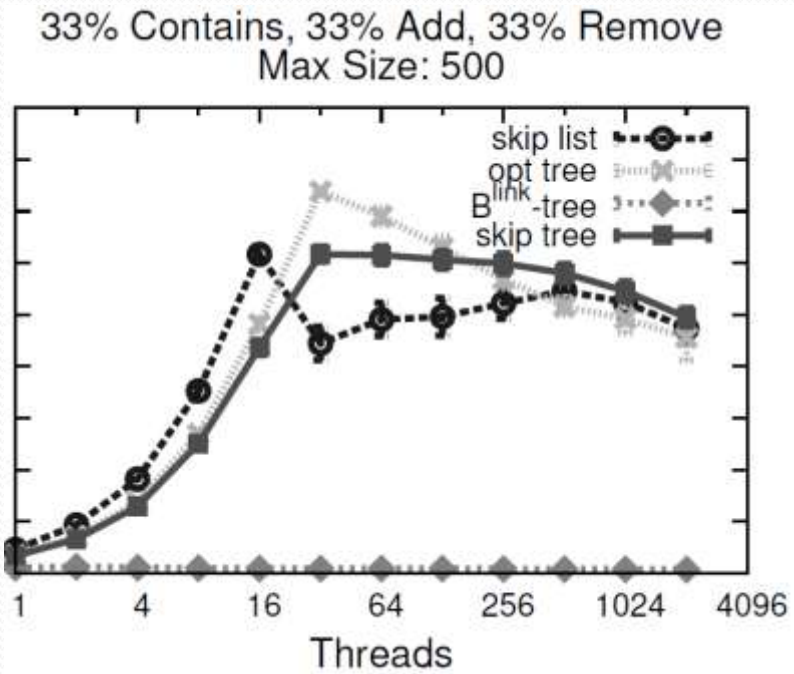
Synthetic Benchmarks - II



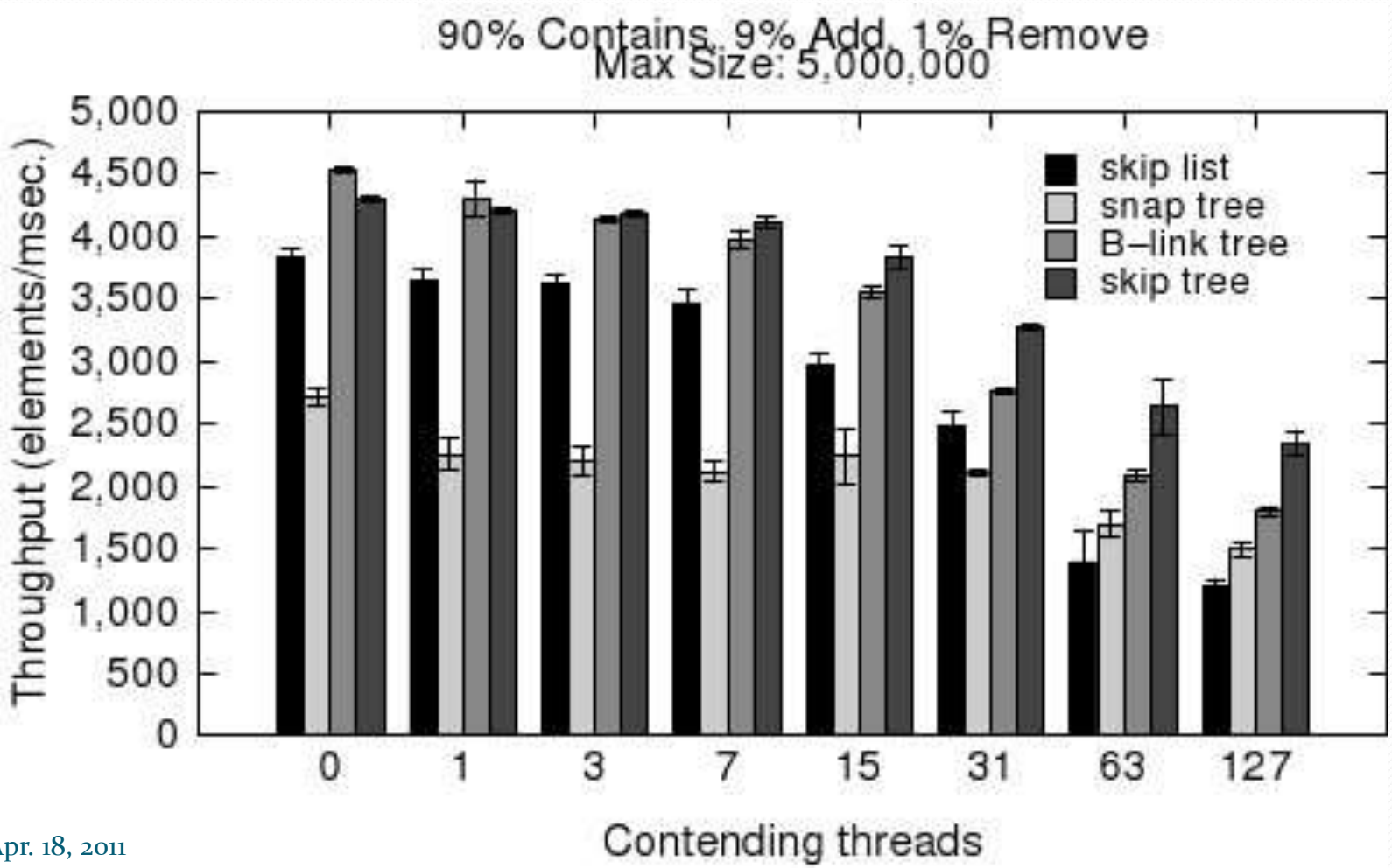


Synthetic Benchmarks - III





Iterators Under Contention



Additional Desirable Properties

Data Structure	Concurrent	Cache-conscious	Randomized	Sorted
Lea skip list	Y	N	Y	Y
Treap, Randomized search tree	N	N	Y	Y
B^{link}-tree	Y	Y	N	Y
Cache-oblivious B-tree	Y	N	Y[a]	Y
Lea hash table	Y	closed address (N)	N	N
Purcell & Harris hash table	Y	open address (N)	N	N
Hopscotch hash	Y	open address (Y)	N	N
HAT-trie	N	Y	N	Y
Lock-free skip tree	Y	Y	Y	Y

Lock-Free Skip Tree - I

A. Each level contains a sequence of elements that are in sorted order. Each level ends with $+\infty$.

B. The leaf level does not contain duplicate elements.

C. Given some value, v, and some level, there exists exactly one pair of adjacent elements A and B such that $A < v \leq B$.

Lock-Free Skip Tree - II

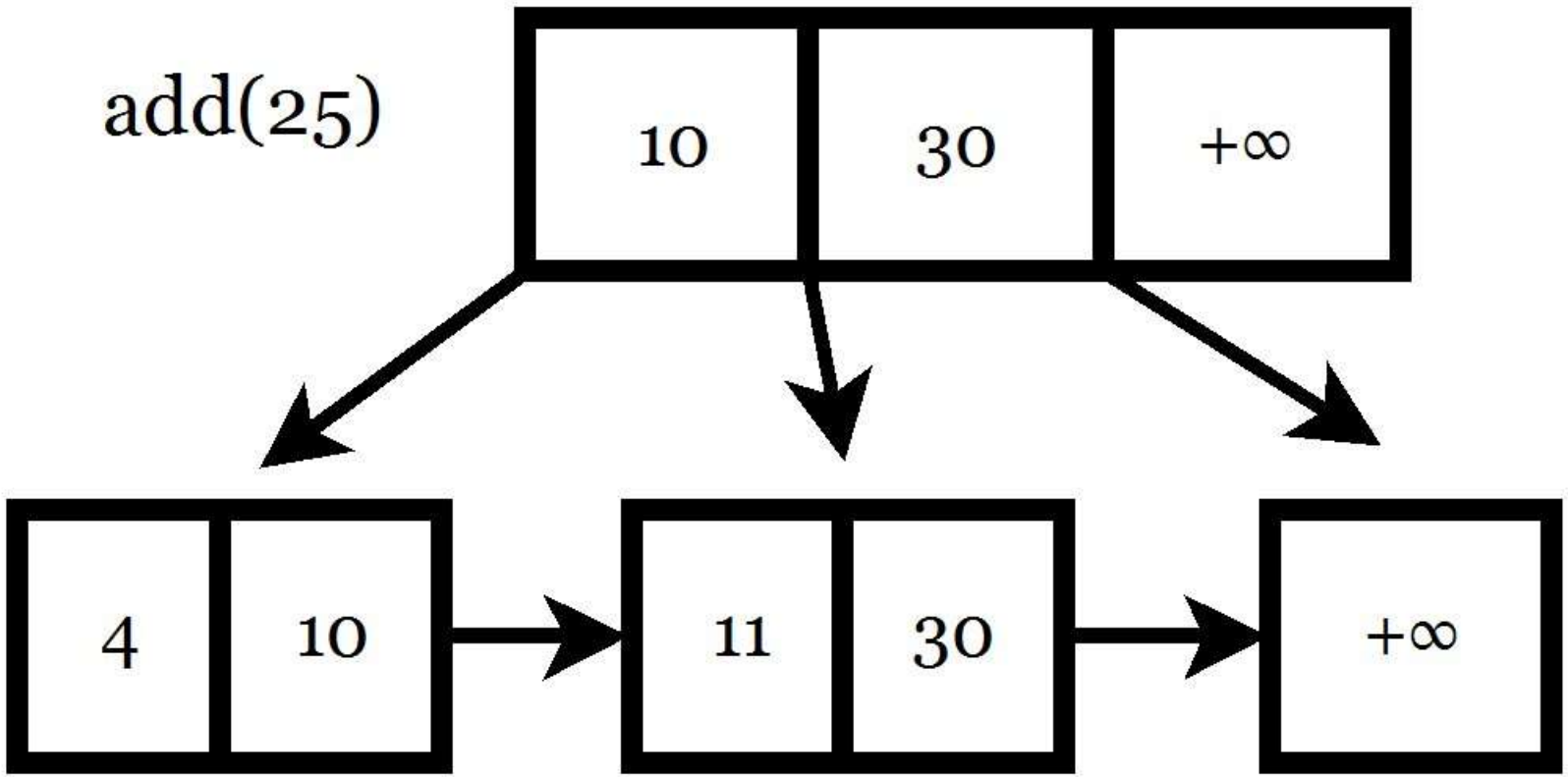
tail(N) - The *tail set* of some node *N* consists of the union of *N* and the *tail set* of the right sibling of *N*.

D. Given some value *v*, let *{W, X}* and *{Y , Z}* be the elements at adjacent levels that satisfy property C. Define *source* as the child node of *W* and *X*, and *target* as the node of Z.

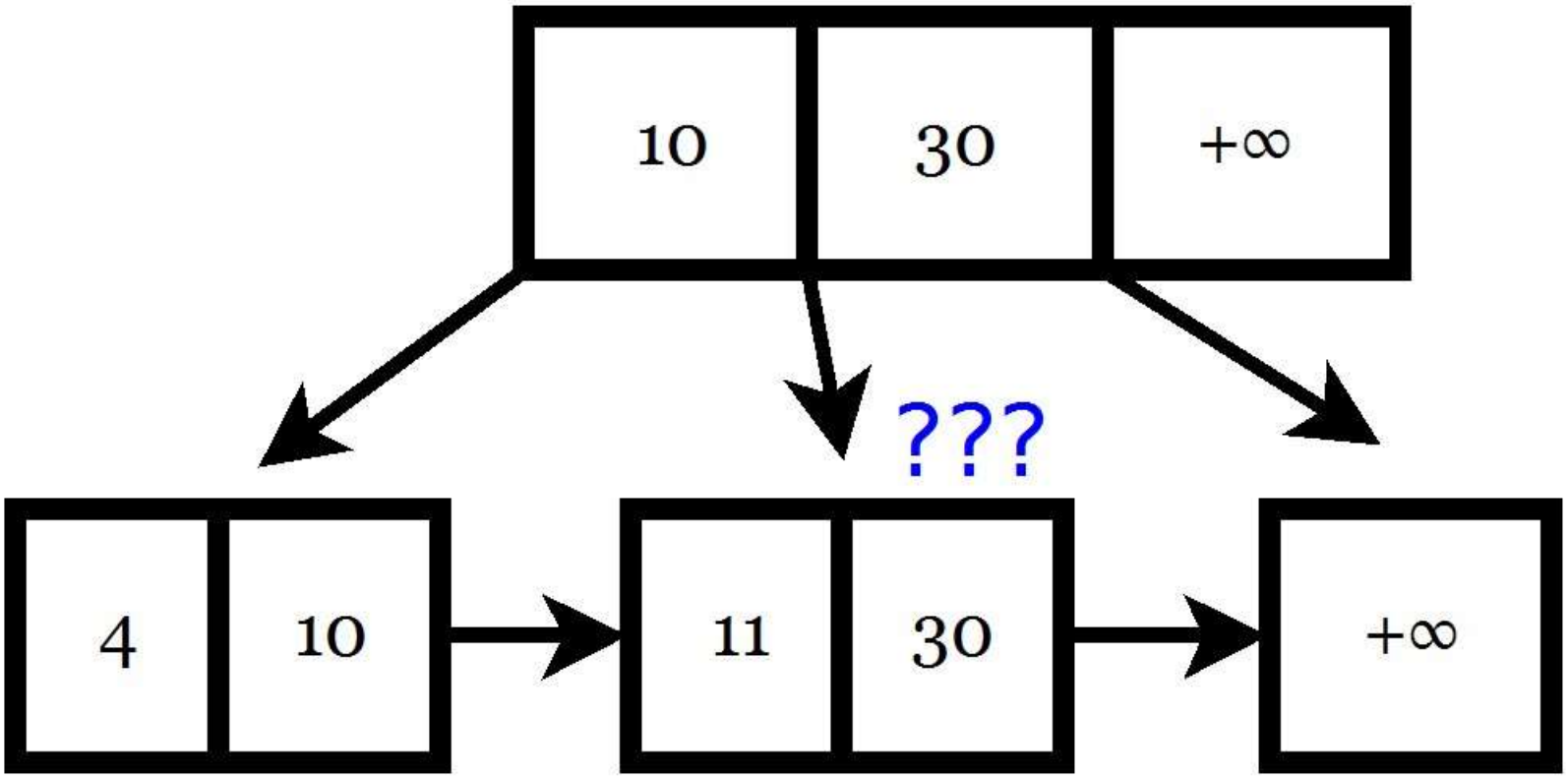
Require that $target \in tail(source)$.

E. Given some value, *v*, and some node *N*. If *v* is ≥ all the elements of *N*, then *v* will always be ≥ all the elements of *N*.

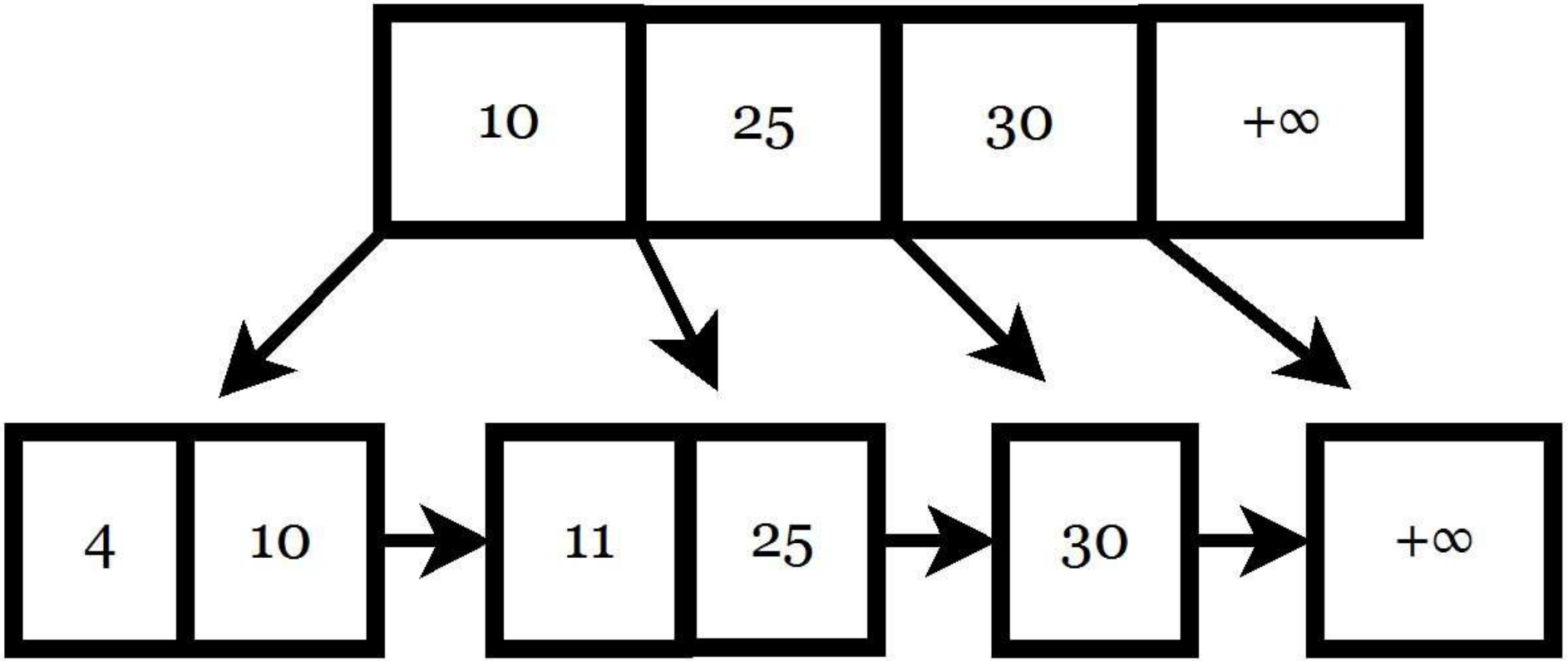
Insertion



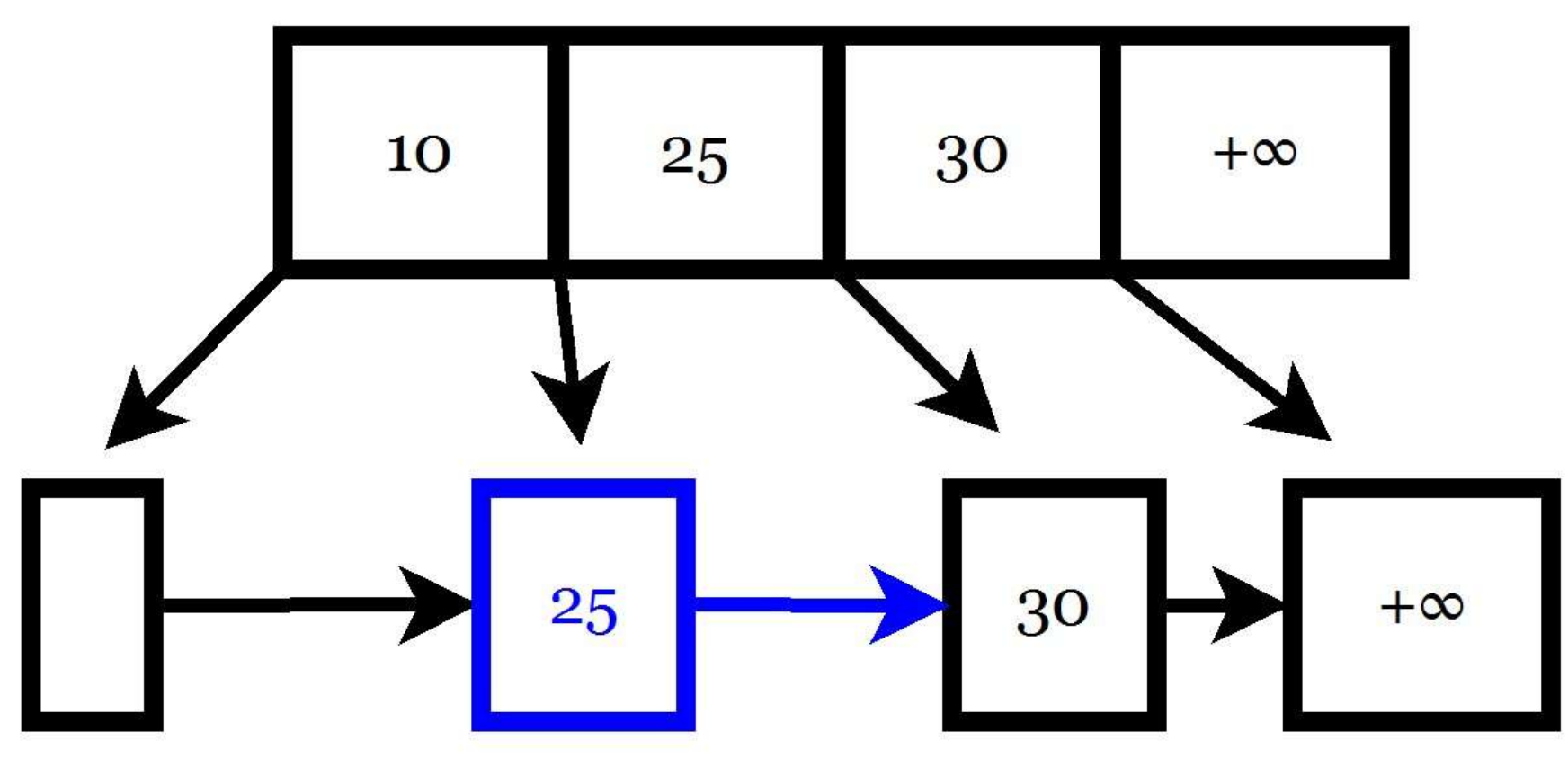
Insertion



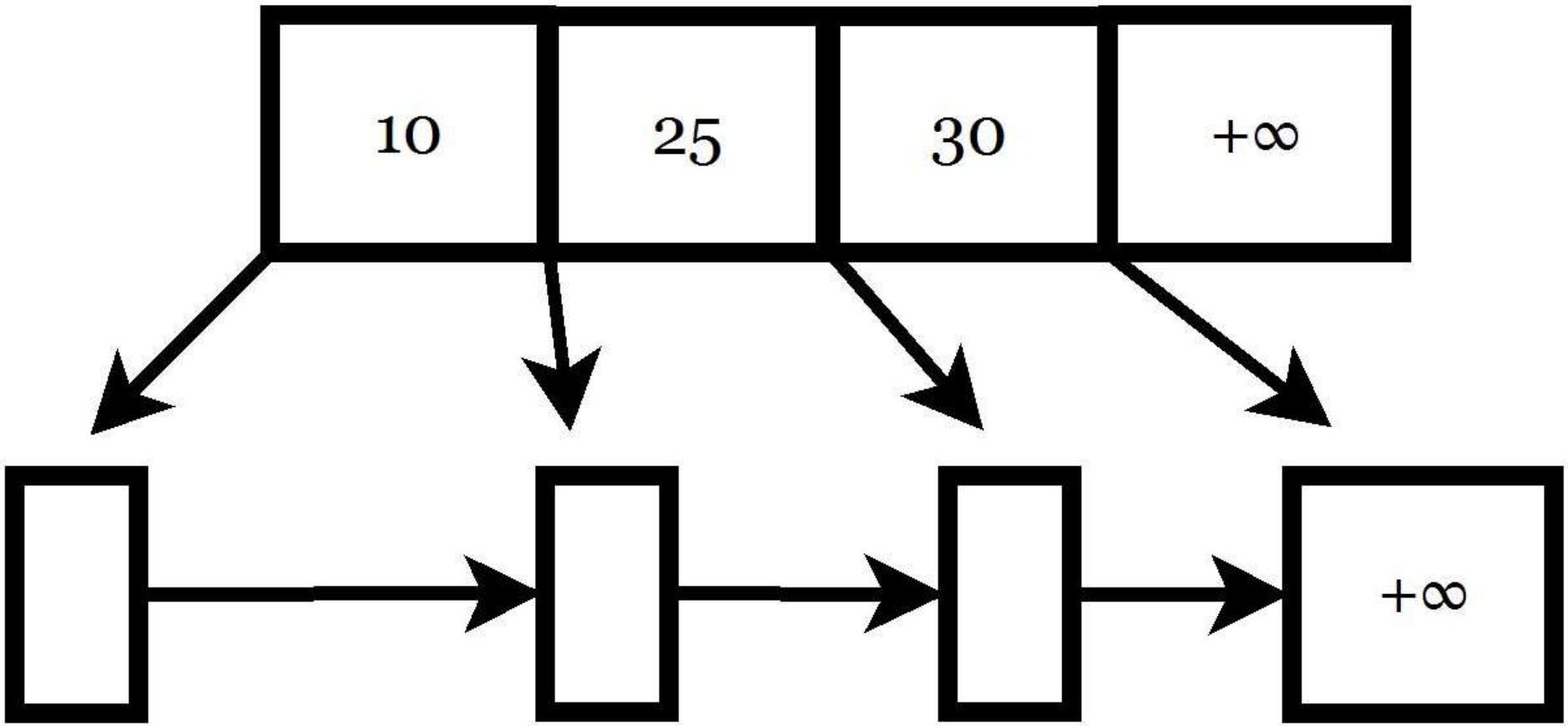
Deletion



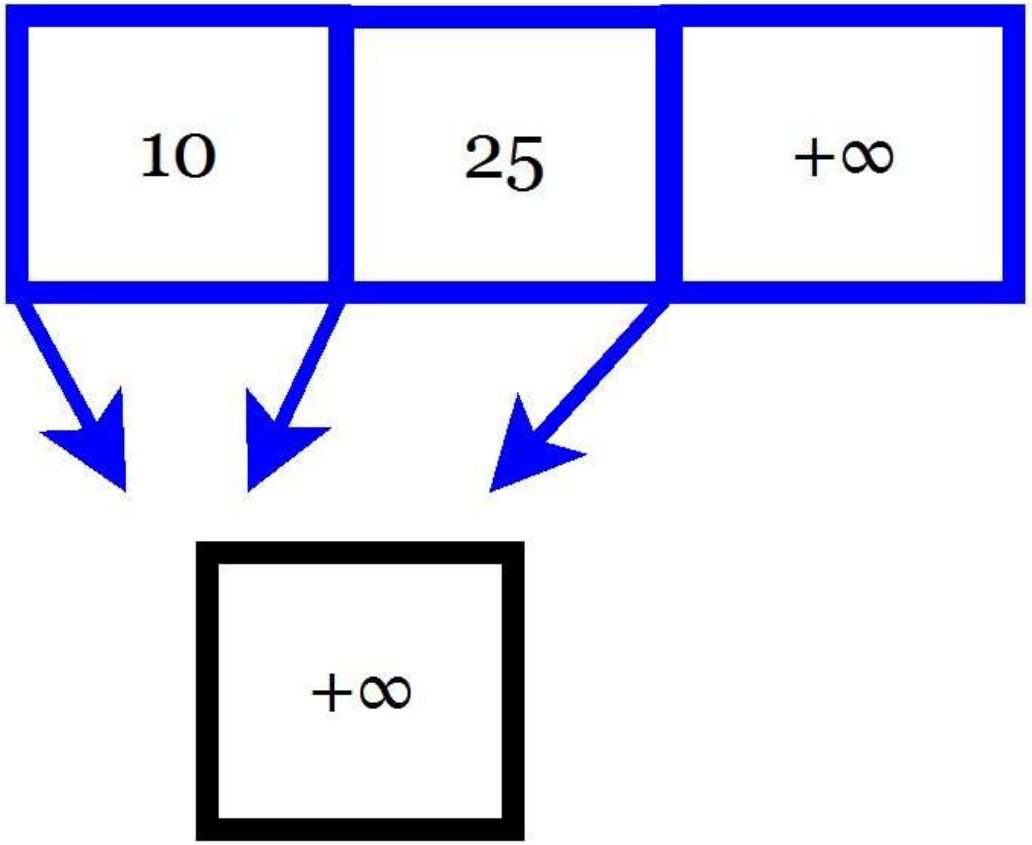
Deletion



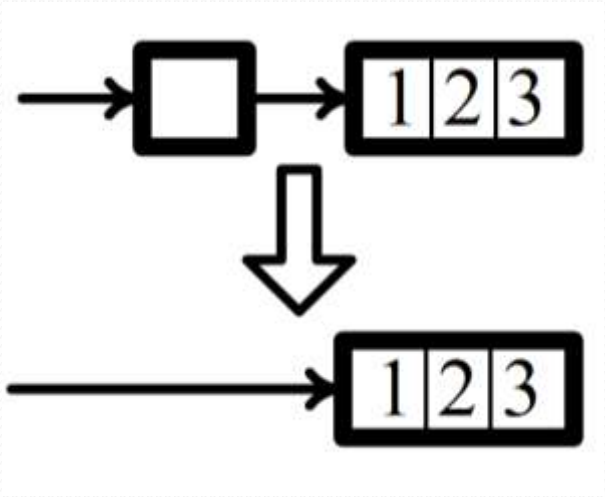
Node Compaction

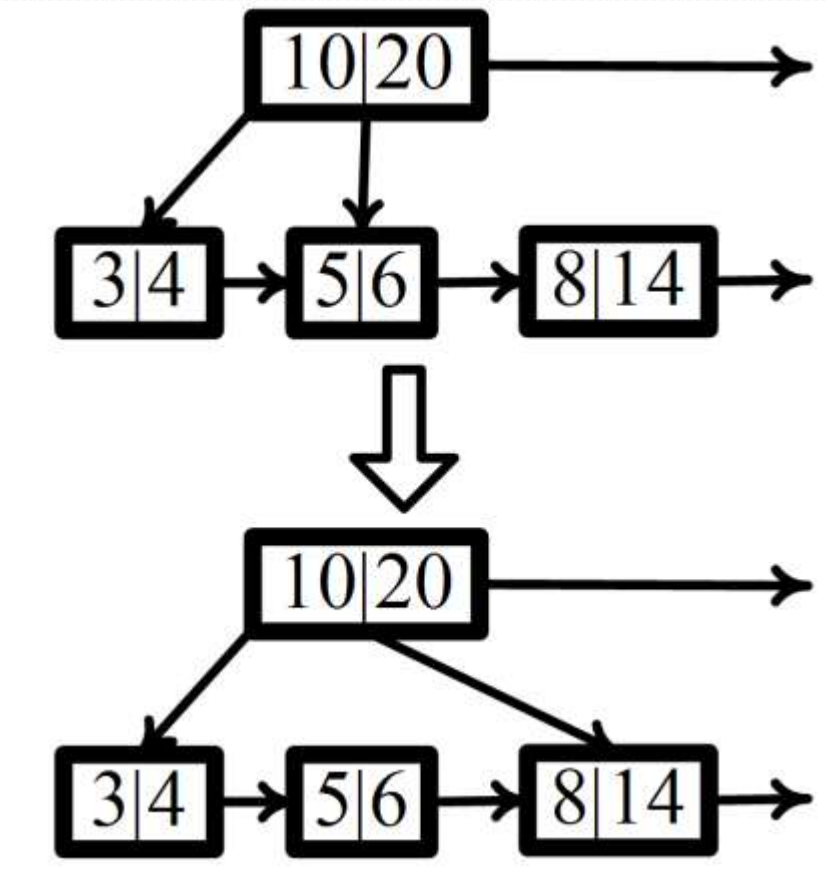


Node Compaction

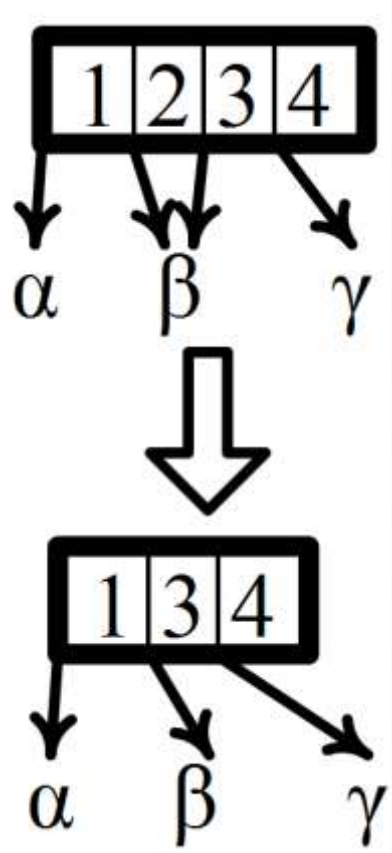


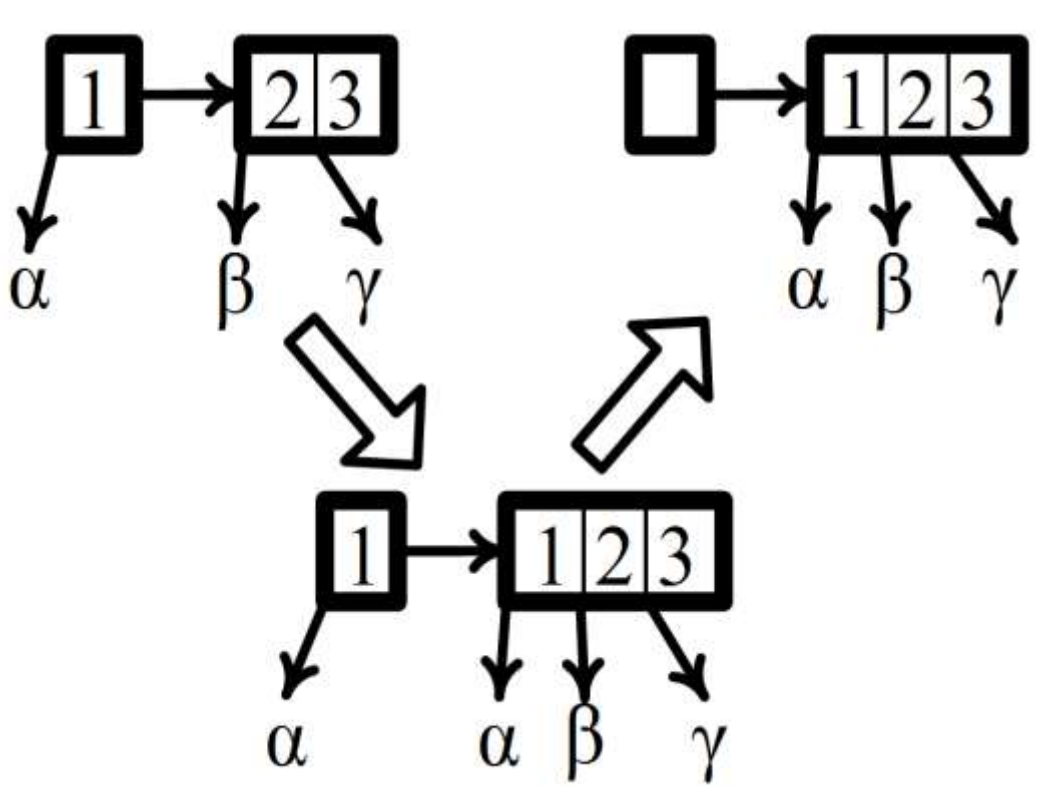
Node compaction – I & II



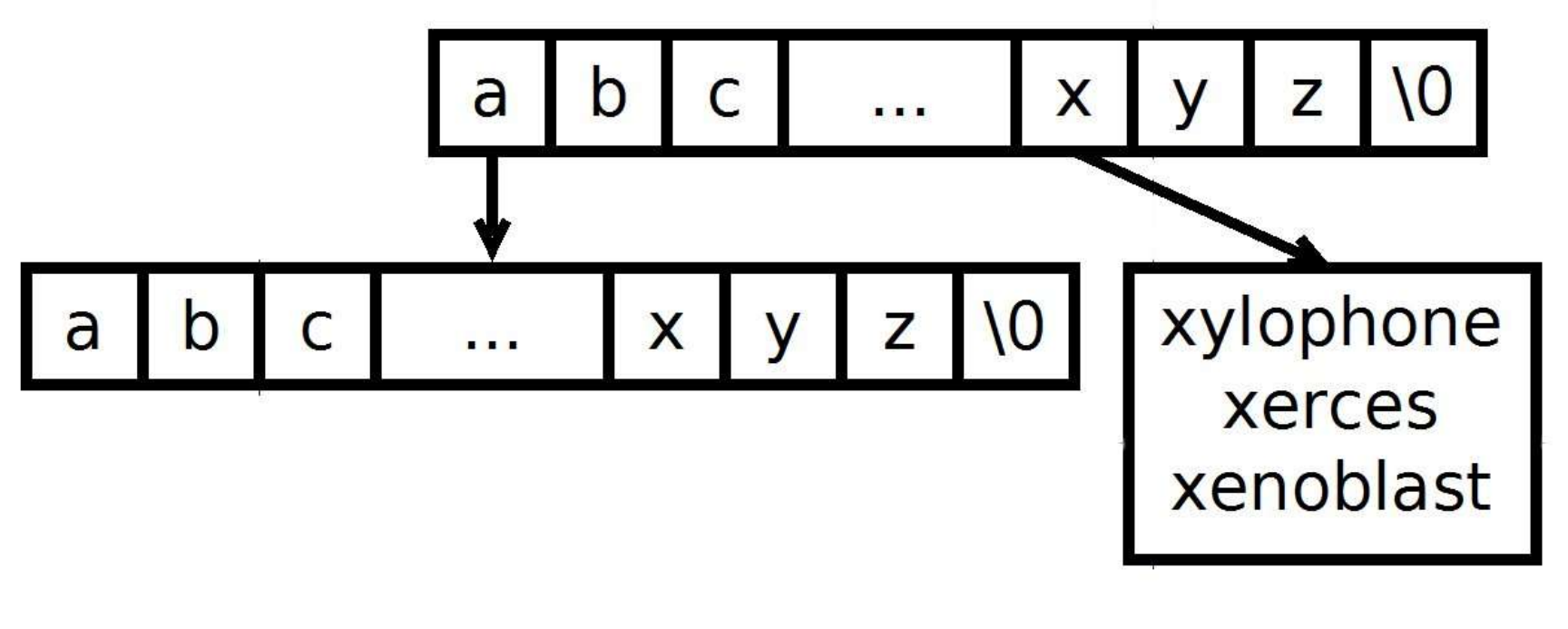


Node compaction – III & IV





Lock-free Burst Trie



Burst Trie States