Faster Cover Trees

Mike Izbicki and Christian R. Shelton

UC Riverside

Outline

- Why care about faster cover trees?
- Making cover trees faster.

Methods for fast nearest neighbor queries:

	provable speedup	arbitrary metric	high dimensions
quadtree	yes	no	no
*k*d-tree	yes	no	somewhat
hashing	yes	no	yes
ball tree	no	yes	somewhat
cover tree	yes	yes	yes

(Beygelzimer, Kakade, and Langford, 2006)

Other uses of cover trees

Any learning algorithm that cares about distance can be made faster using cover trees.

Examples:

- k-nearest neighbor
- Support vector machines (Segata and Blanzieri, 2010)
- Dimensionality reduction (Lisitsyn *et. al.*, 2010)
- Reinforcement learning (Tziortziotis *et. al.*, 2014)

Outline

- Why care about faster cover trees?
- Making cover trees faster.
  - Experimental setup
  - Simpler definition reduces the number of nodes
  - The nearest ancestor invariant
  - Better cache performance
  - Constructing and querying the tree in parallel

Experimental setup

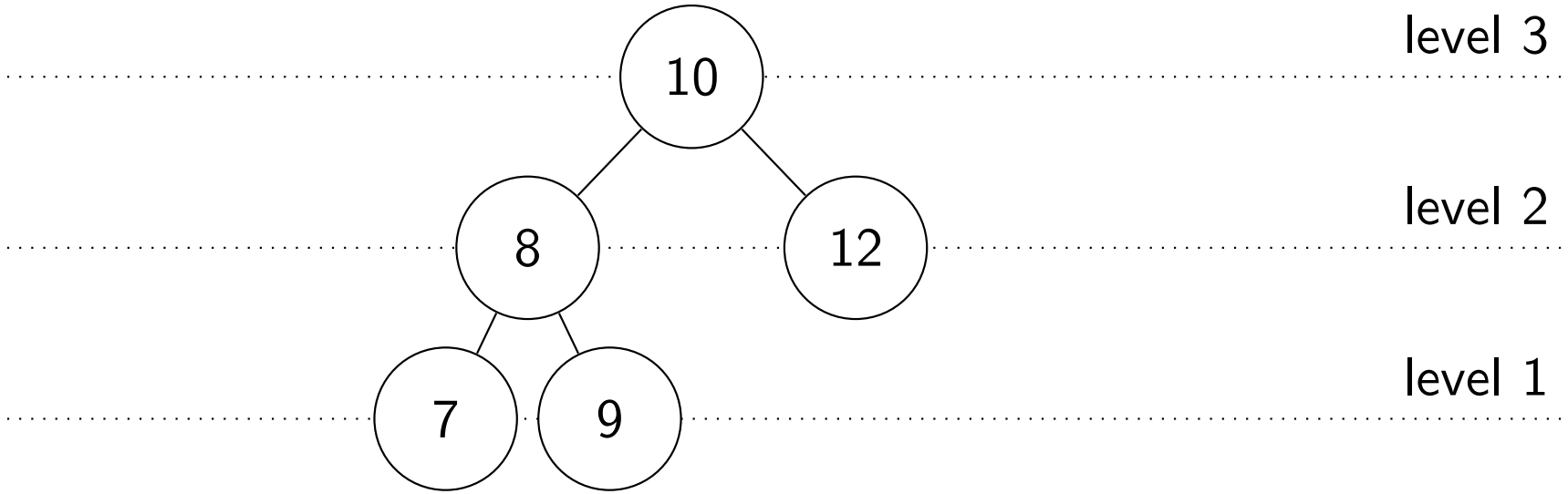
Three data sources:

- MLPack benchmarks with Euclidean distance
- Protein dataset with the random walk graph distance
- Yahoo! 1.5 million creative common images with the earth movers distance

Benchmarking procedure:

- Construct a cover tree on the dataset
- For each data point in the dataset, find the nearest neighbor

The simplified cover tree



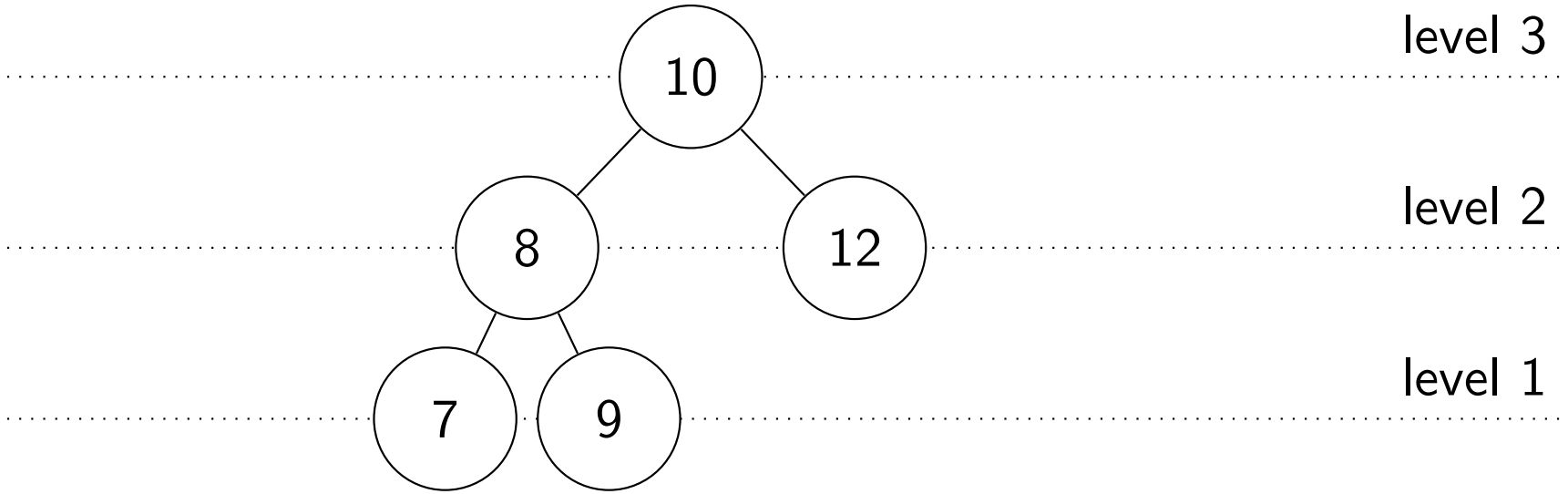
The covering invariant. For every node p, define the function $\texttt{covdist}(p) = 2^{\texttt{level}(p)}$. For each child q of p

$$d(p, q) \leq \texttt{covdist}(p)$$

The separating invariant. For every node p, define the function $\texttt{sepdist}(p) = 2^{\texttt{level}(p)-1}$. For all distinct children q_1 and q_2 of p

$$d(q_1, q_2) \geq \texttt{sepdist}(p)$$

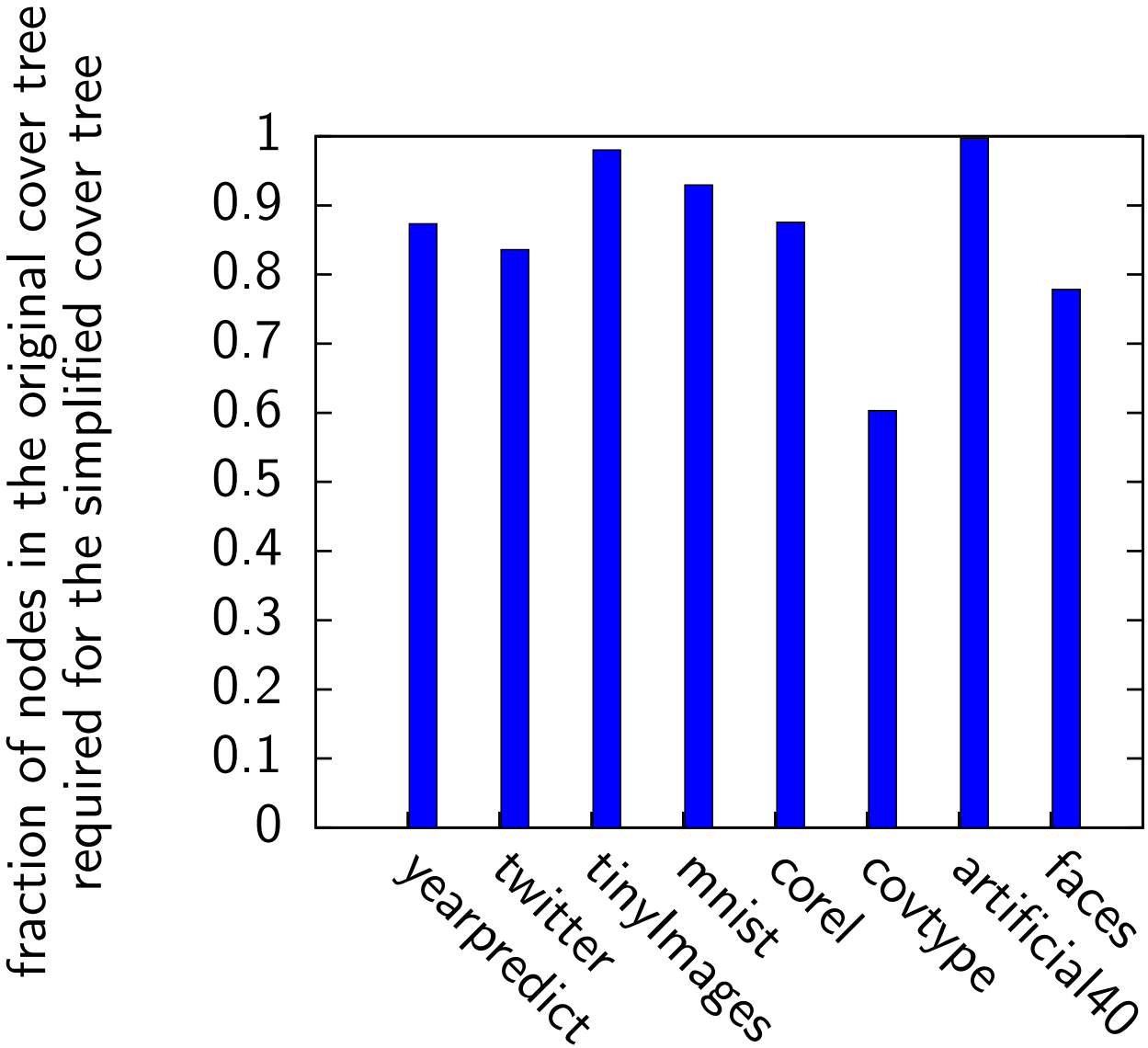
The simplified cover tree



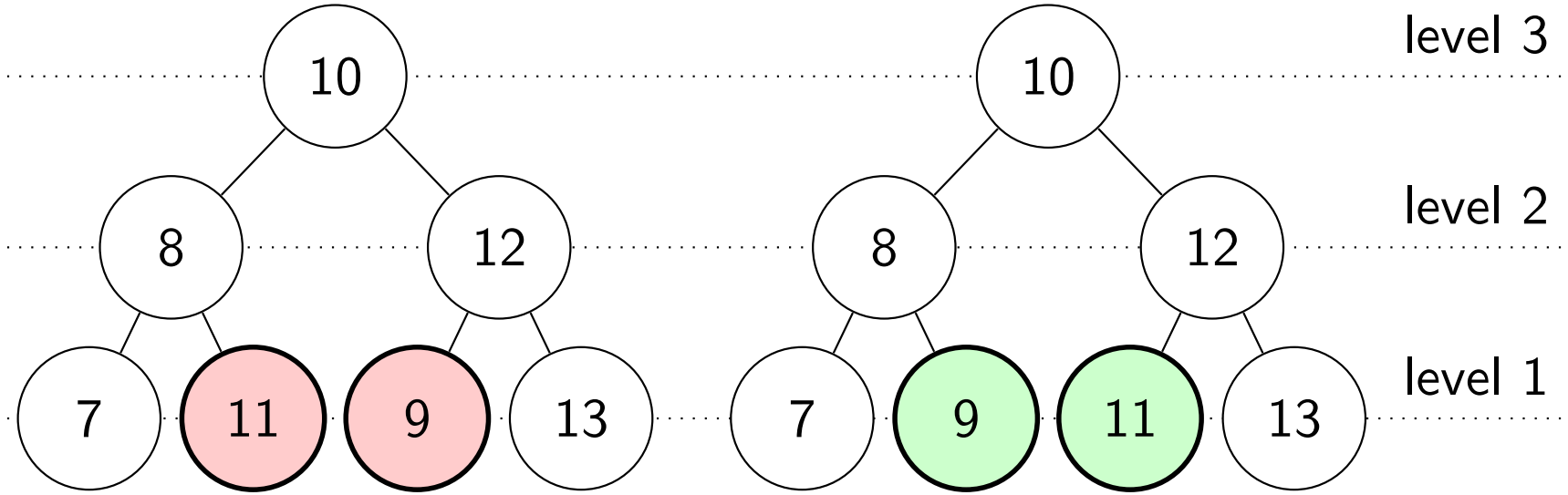
Advantages of the simplified cover tree:

- Maintains all runtime guarantees of the original cover tree.
- Significantly easier to understand and implement.
  The original cover tree was described in terms of an infinitely large tree, only a subset of which actually gets implemented.
- Requires exactly n nodes instead of $O(n)$ nodes.
  Fewer nodes means a faster constant factor for all algorithms.

The simplified cover tree



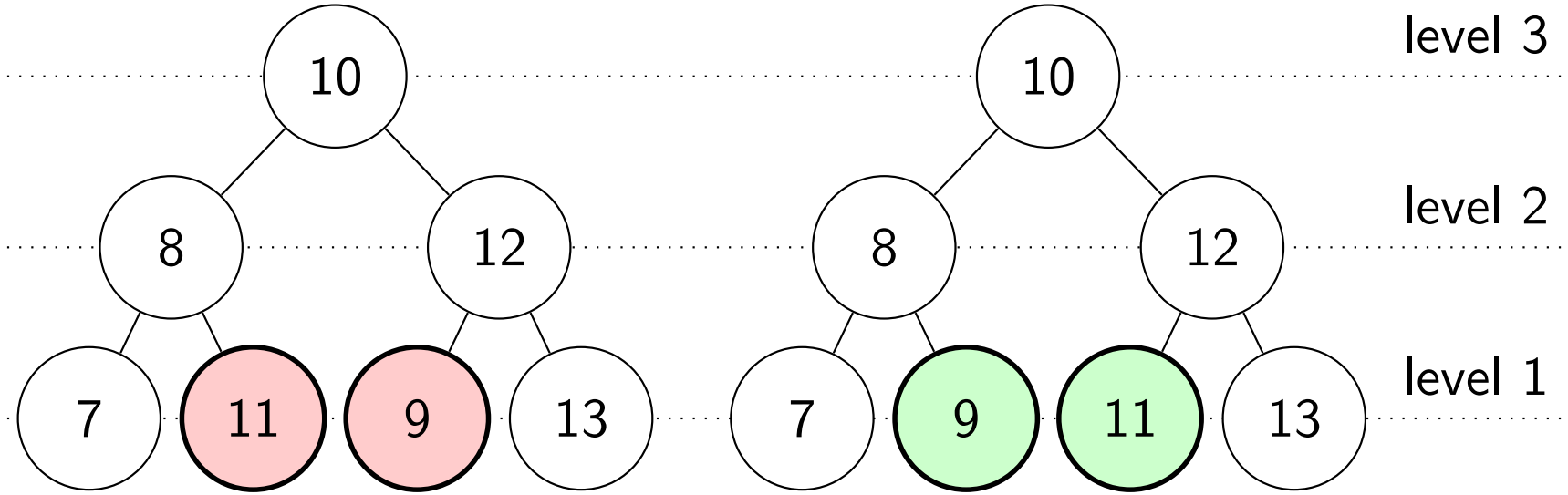
The nearest ancestor cover tree



A **nearest ancestor cover tree** is a simplified cover tree where every point p satisfies the additional invariant that if q_1 is an ancestor of p and q_2 is a sibling of q_1, then

$$d(p, q_1) \leq d(p, q_2)$$

The nearest ancestor cover tree

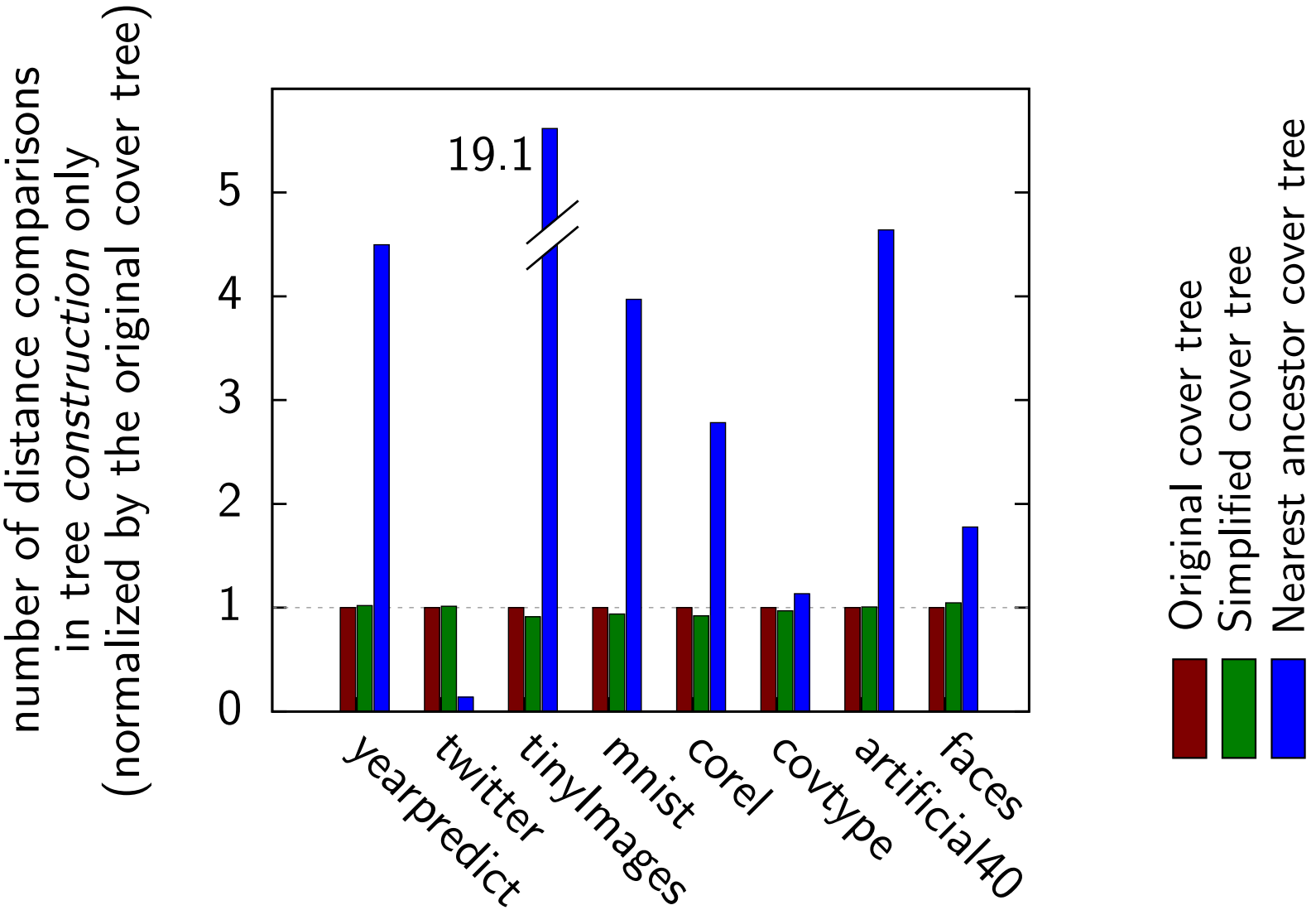


Insertions require rebalancing.

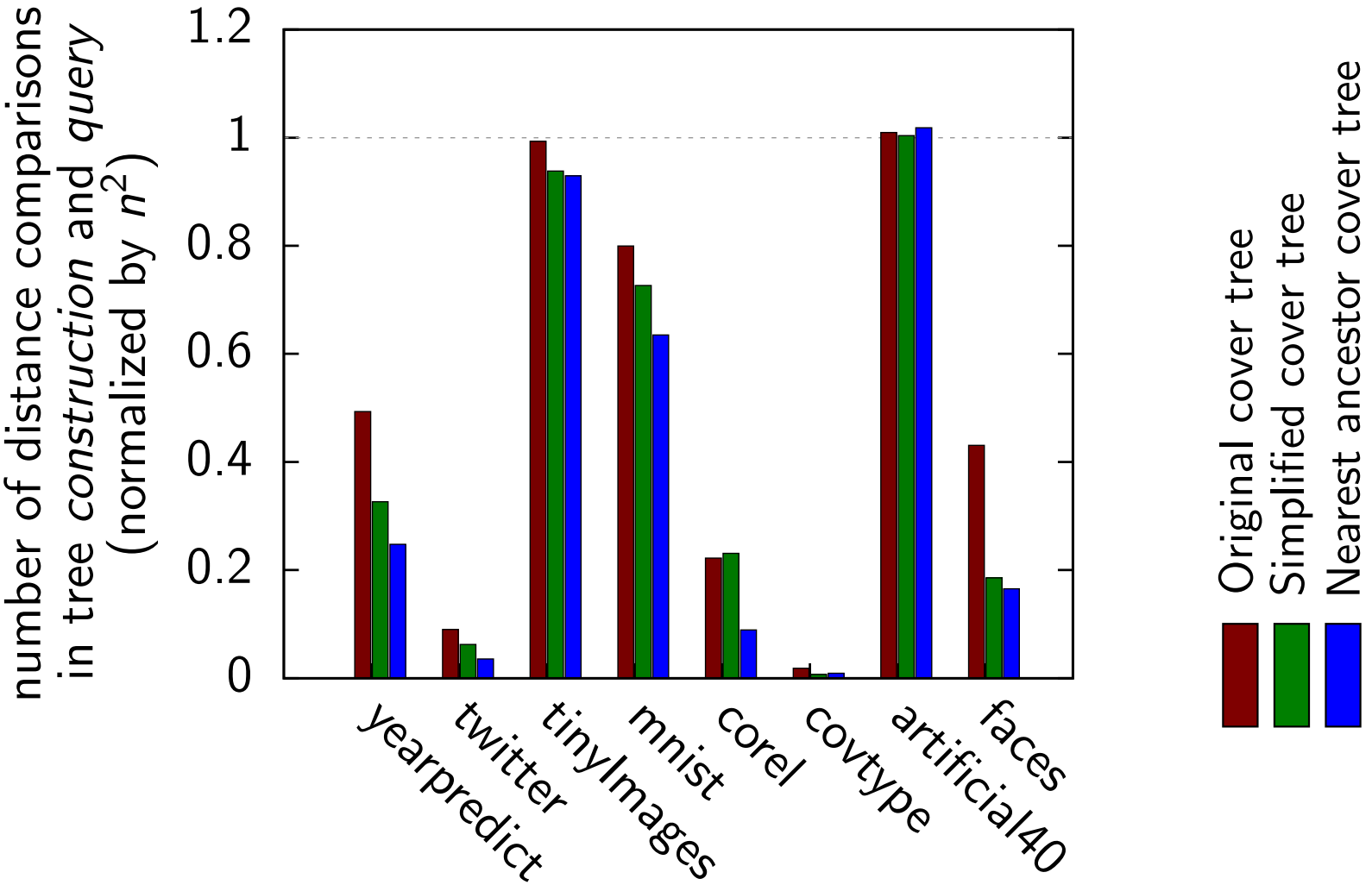
No runtime guarantees on the rebalance step.

In practice, queries are much faster and construction is only slightly slower.

Comparing cover trees on *construction* time

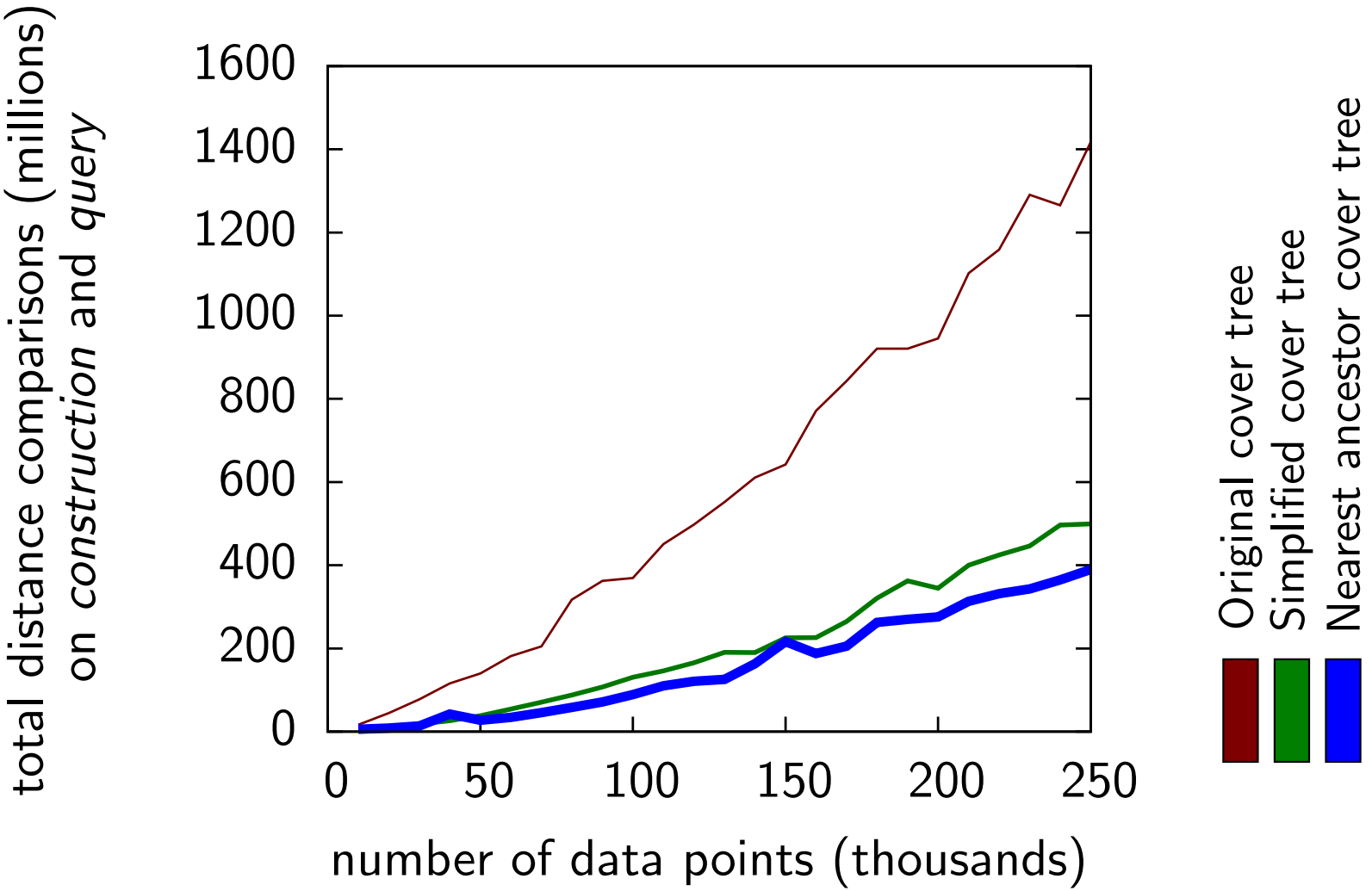


Comparing cover trees on *construction and query* time



All of the cover trees scale similarly

This experiment uses the protein data and the random walk graph kernel.



Cache oblivious cover tree

Need to consider cache accesses for fast, modern data structures

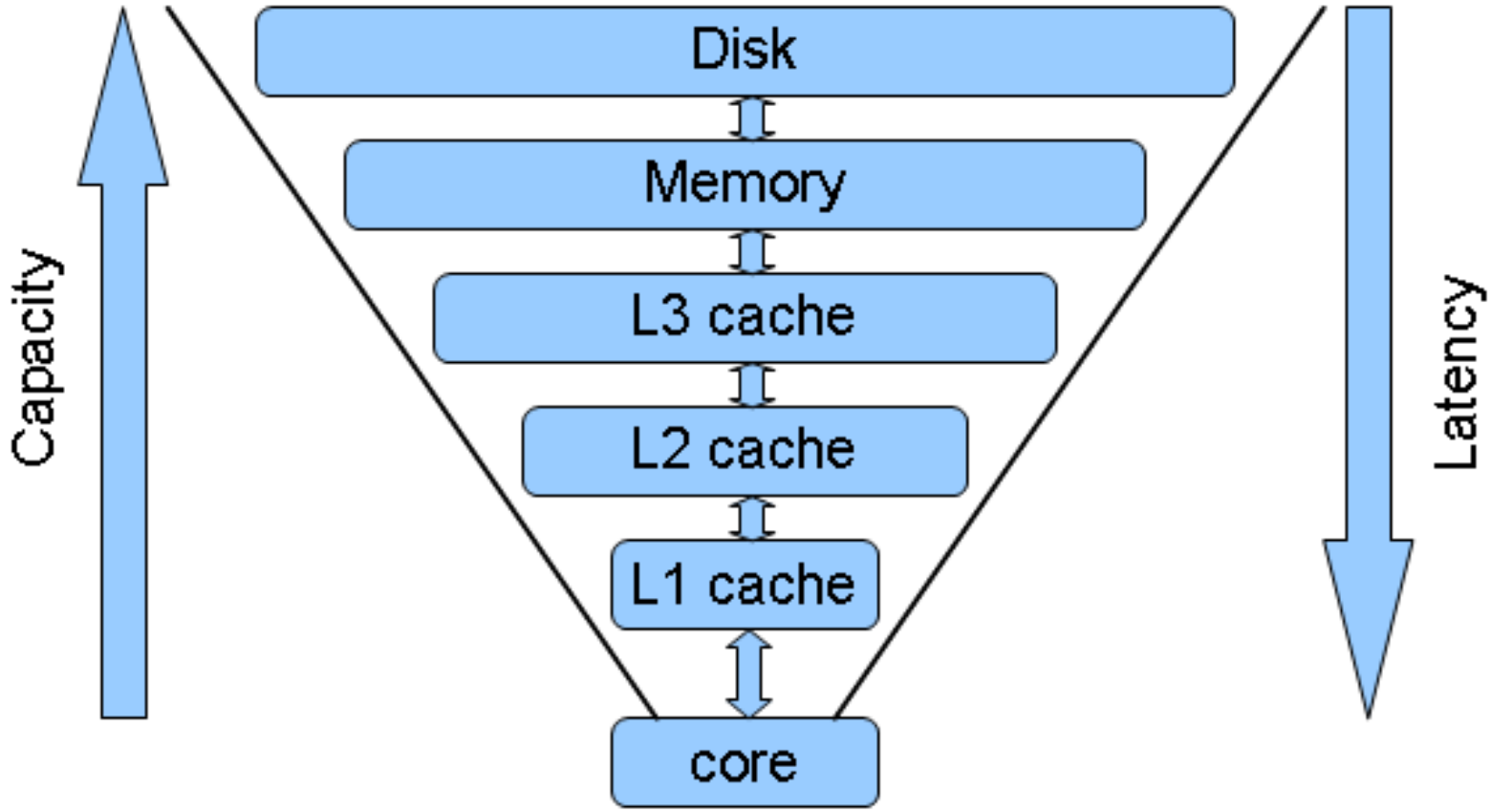


image from: `http://1024cores.net`

Cache oblivious cover tree

Arrange nodes in memory according to a preorder traversal of the tree (van Emde Boas *et al.*, 1966; Demaine, 2002)

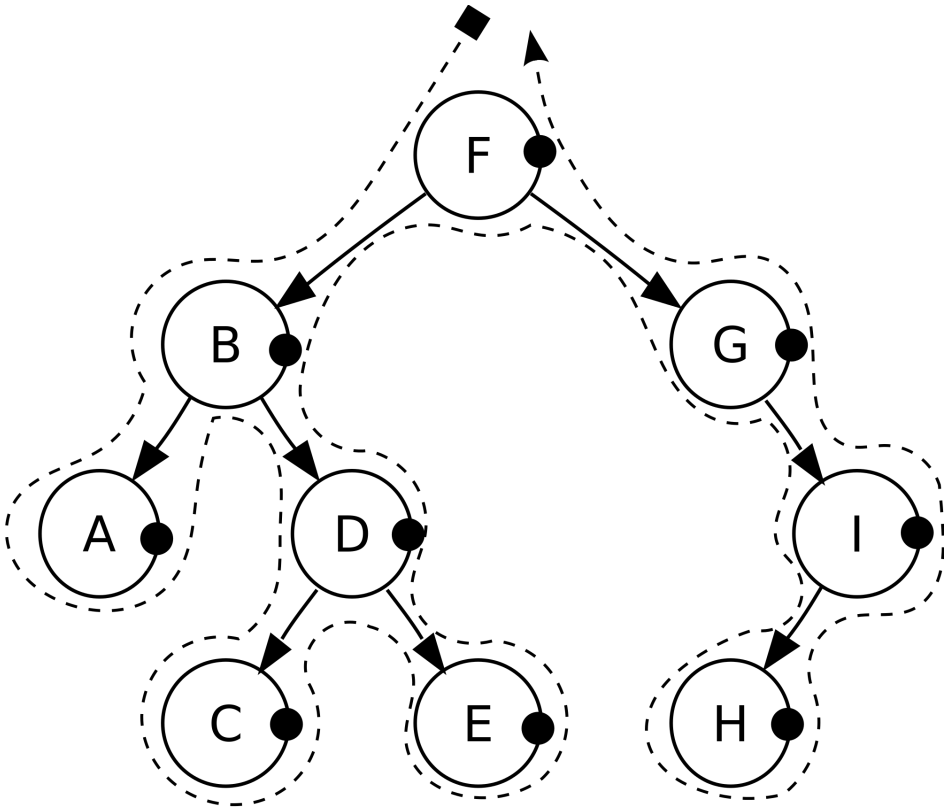
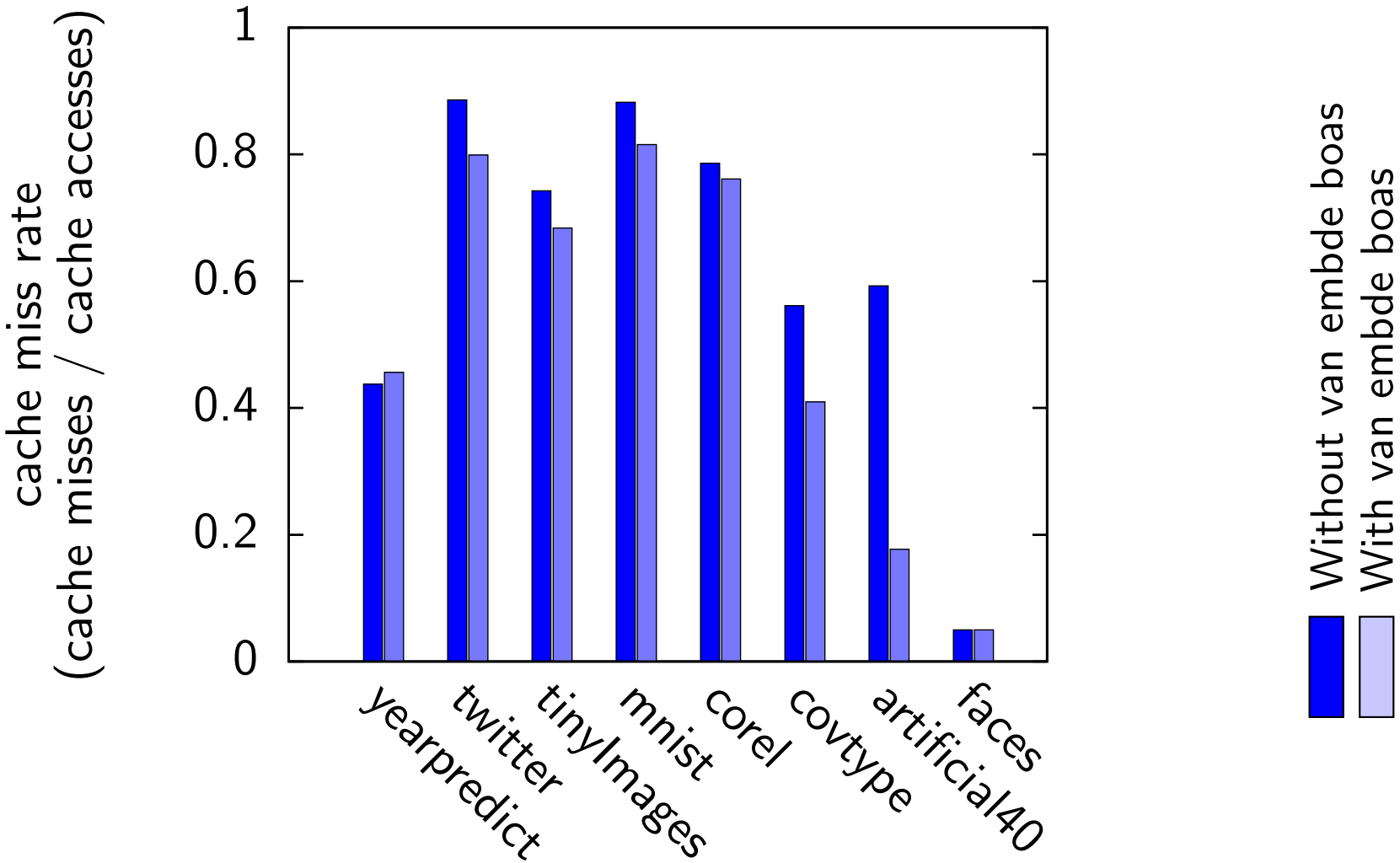


image from: Wikipedia

The cache efficiency of three cover tree implementations

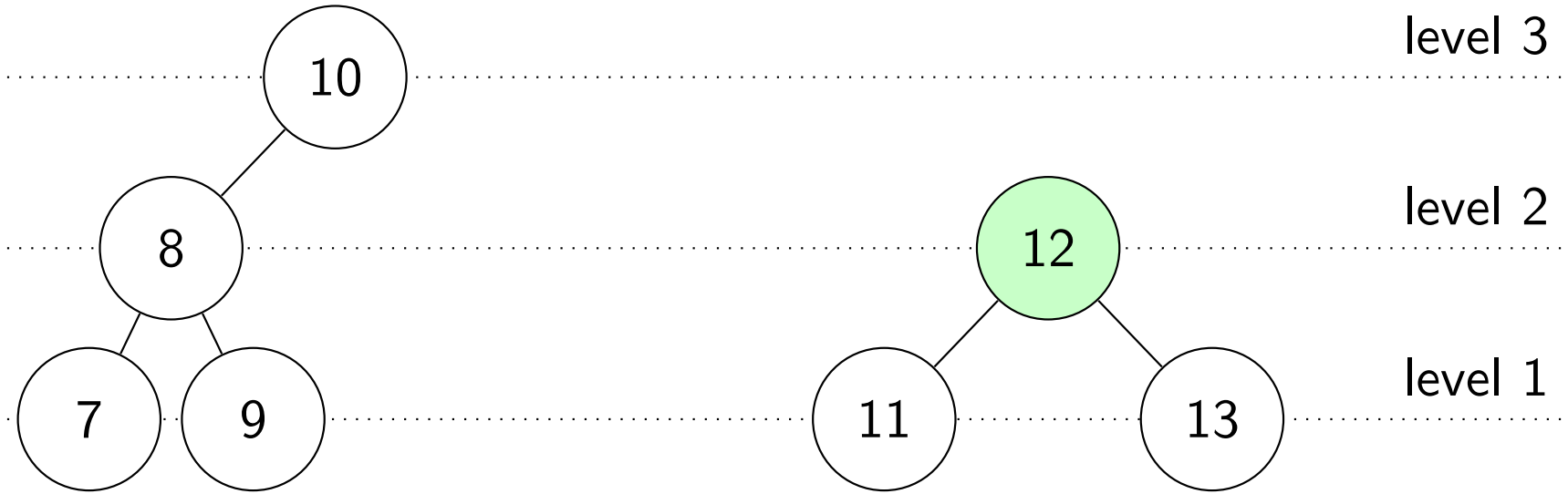


Measured using Linux's perf stat utility on an Amazon AWS instance

Merging cover trees

Merging cover trees gives us a parallel tree construction algorithm

Sometimes, merging cover trees is **easy**:

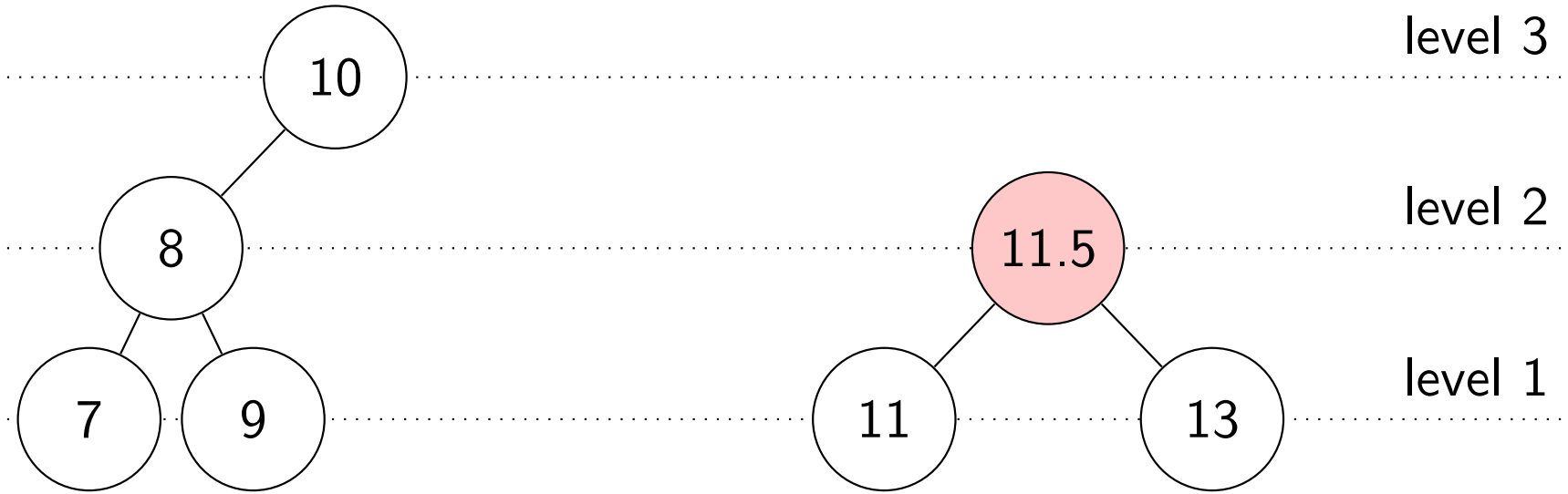


No runtime bound on the merge operation, but it is fast in practice

Merging cover trees

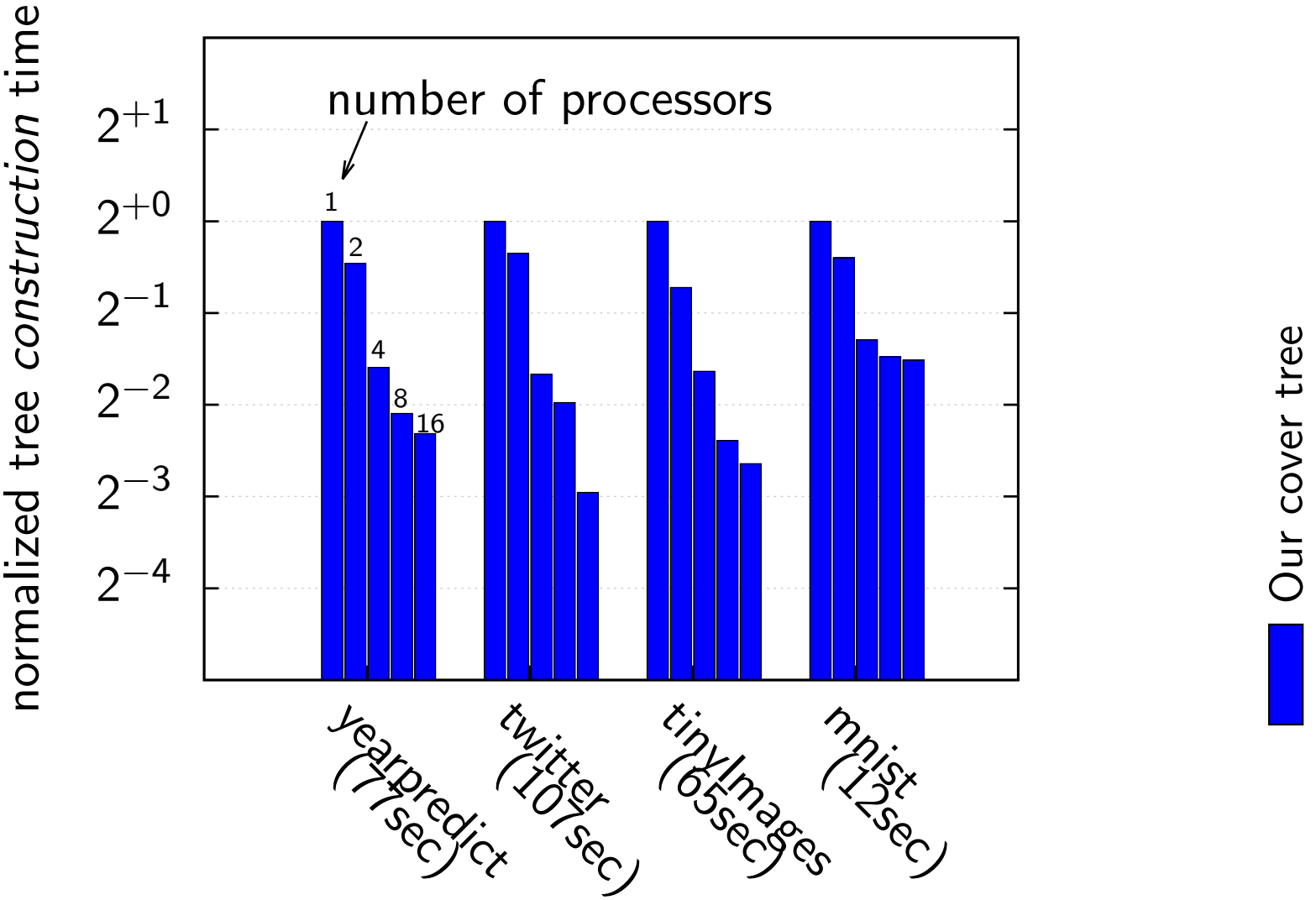
Merging cover trees gives us a parallel tree construction algorithm

Sometimes, merging cover trees is **hard**:



No runtime bound on the merge operation, but it is fast in practice

The effect of parallel tree *construction* on small datasets



Experiments run on an Amazon AWS instance with 16 true cores

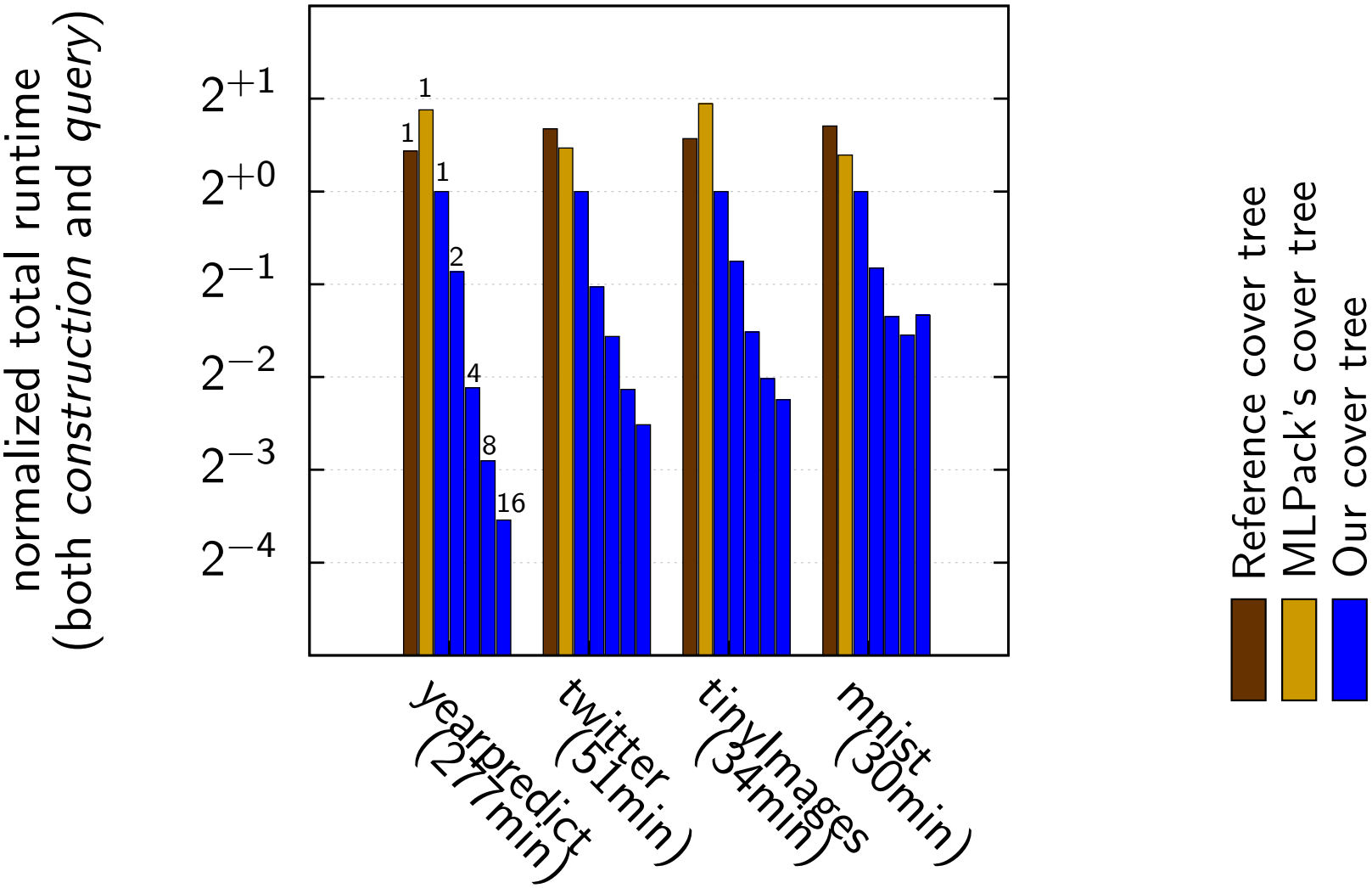
Parallel tree construction really matters on larger data sets

On large datasets with an expensive metric, parallelism is more useful

Yahoo! Flickr dataset with 1.5 million images and earth mover distance

num cores	simplified tree		nearest ancestor tree	
	time	speedup	time	speedup
1	70.7 min	1.0	210.9 min	1.0
2	36.6 min	1.9	94.2 min	2.2
4	18.5 min	3.8	48.5 min	4.3
8	10.2 min	6.9	25.3 min	8.3
16	6.7 min	10.5	12.0 min	17.6

The effect of parallel tree *construction and query*



Experiments run on an Amazon AWS instance with 16 true cores

Summary

You should use cover trees.

We made them easier to implement and faster.

All the code is licensed under the BSD3 and available at:

`http://github.com/mikeizbicki/hlearn`