DeltaTree
A Locality-aware Concurrent Search Tree
Ibrahim Umar, Otto J. Anshus, Phuong H. Ha
ibrahim.umar@uit.no, otto@cs.uit.no, phuong@cs.uit.no
Arctic Green Computing group, Department of Computer Science,
University of Tromsø - The Arctic University of Norway
(http://site.uit.no/arcticgreen)

/ INTRODUCTION: Problems with locality-aware trees
Locality-aware trees:
• Cache-conscious trees (e.g., Intel Fast, Intel Palm)
  • Usually platform dependent
• Cache-oblivious (CO) trees using conventional van Emde Boas (vEB) layout
  • Poorly support concurrent update operations
• Inserting (or deleting) a node in the contiguous block of memory may trigger
  a restructuring of a large part of the tree
• Need to allocate a new contiguous block of memory for the whole tree if the
  previously allocated block of memory is full

Cache-oblivious tree layout (or van Emde Boas (vEB) layout) compared to BFS layout.
• Numbers inside each node are the physical memory location of the nodes
• Block transfer size is 3 ($B = 3$)
• vEB layout has $\log B$ less memory transfers than BFS layout.

/ OBJECTIVE
Create a platform-independent locality-aware concurrent search tree by
making vEB layout suitable for concurrent update operations.

/ PROPOSED SOLUTIONS:
Relaxed Cache Oblivious and dynamic vEB layout
• Relaxed cache-oblivious algorithms:
  • Cache-oblivious (CO) algorithms with a restriction that upper bound $UB$ on the
    unknown memory block size $B$ is known in advance
• Novel concurrency-aware dynamic vEB layout:
  • Supports dynamic node allocation via pointers
  • Optimal search cost of $O(\log_B N)$ memory transfers without knowing $B$

DeltaTree is a k-ary leaf-oriented tree of DeltaNodes in which each DeltaNode
is a fixed-size tree-container with the vEB layout

/ IMPLEMENTATION: DeltaTree
• Based on the new concurrency-aware vEB layout, we developed a new locality-
  aware concurrent search tree called DeltaTree (or $\Delta$Tree)

/ EXPERIMENTAL RESULTS
We experimentally evaluated DeltaTree ($\Delta$Tree) against CBtree (a fast
concurrent B-tree by Lehman and Yao (1981)) on multiple platforms.

/ CONCLUSIONS
• We present DeltaTree, a platform-independent locality-aware concurrent
  search tree
• DeltaTree minimizes data transfer from memory to CPU and supports high
  concurrency
• DeltaTree is up to 50%, 65%, 170%, and 4x faster than highly concurrent B-
  trees on a commodity Intel high performance computing (HPC) platform,
  an ARM embedded platform, an accelerator platform, and a specialized
  computing platform, respectively

This work has received funding from the European Union Seventh Framework Programme (grant 611183) and from the Research Council of Norway (grant 231746/F20). The authors would like to thank the Norwegian Metacenter for Computational Science for giving us access to HPC clusters.