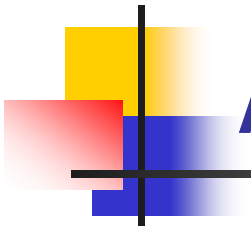
Accommodating *k*-d Trees in the SDDS Model

Panayiotis Bozanis

pbozanis@inf.uth.gr

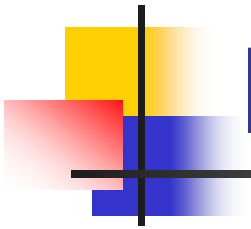
Computer & Communications Engineering Dept.,

School of Engineering, University of Thessaly, Greece



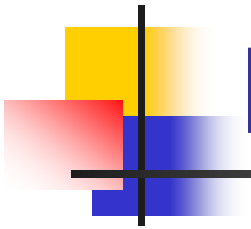
Abstract

We propose a new scheme for accommodating pseudo *k*-d trees, random versions of the well known multidimensional search structures, in the SDDS model. Our scheme exhibits logarithmic update time, either constant or logarithmic search time for single key queries and output sensitive query time for range search query.



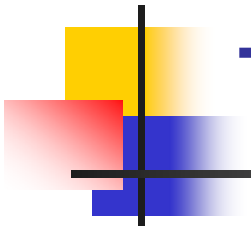
Key Terms

- **Network computing:** fast networks interconnect many powerful and low-priced workstations, creating a pool of perhaps terabytes of RAM and even more of disc space.
- **Server:** Every site that manages data. It provides a storage space of *b* data elements, termed *bucket,* to accommodate a part of the file under maintenance.



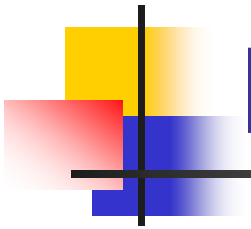
Key Terms (Cont.)

- **Client:** Every site that requests access to data. Each client ignores the presence of other clients.
- Sites communicate by sending and receiving *point-to-point* messages.
- The underlying network is assumed *error-free*.
- The complexity of operations is measured in number of exchanged messages.



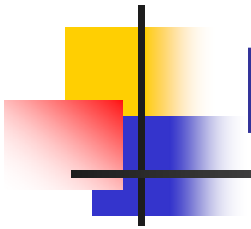
The SDDS Model [LNS93]

- Calls out for the design and implementation of distributed algorithms and data structures that
  - should expand to new servers gracefully, and only when servers already used are efficiently loaded; and
  - their access and maintenance operations never require atomic updates to multiple clients, while there is no centralized “access” site.



Previous Work

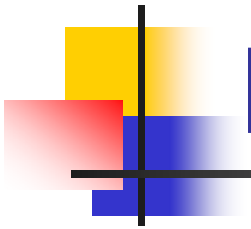
- Hash Schemes
  - [LNS93] W., Litwin, M. A., Neitmat, D. A., Schneider. LH*–Linear hashing for distributed files, *ACM TODS'96*
  - [D94] R., Devine. Design and implementation of DDH: distributed dynamic hashing, FODO'93
  - [VBW94] R., Vingralek, Y., Breitbart, G., Weikum. Distributed file organization with scalable cost/performance, ACM SIGMOD'94
  - [LS00] W., Litwin, T., Schwarz. LH^*_{RS}– A high availability scalable distributed data structure using red solomon codes, ACM SIGMOD'00
  - [LR02] W., Litwin, T., Risch. LH^*_G– A high availability scalable distributed data structure by record grouping, *TKDE'02*



Previous Work (Cont.)

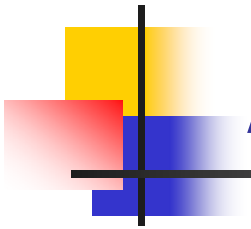
- Unbalanced-Balanced Binary Trees
  - [KW94] B., Kröll, P., Widmayer. Distributing a search tree among a growing number of processors, ACM SIGMOD'94
  - [dPN99] A., di Pasquale, E., Nardelli. Balanced and distributed search trees, WDAS'99
  - [BM00] P., Bozanis, Y., Manolopoulos. Accommodating Skip Lists in the SDDS Model, WDAS'00
  - [dPN01a] A., di Pasquale, E., Nardelli. ADST: An Order Preserving SDDS with Constant Update Cost, SOFSEM'01

    [dPN01b] A very Efficient Order Preserving SDDS, DEXA'01
  - [BM02] P., Bozanis, Y., Manolopoulos. LDT: A Logarithmic Distributed Search Tree, WDAS'02
- B-Trees
  - [LMS94] W., Litwin, M. A., Neitmat, D. A., Schneider. RP* –A family of ordered-preserving scalable distributed data structures, VLDB'94



Previous Work (Cont.)

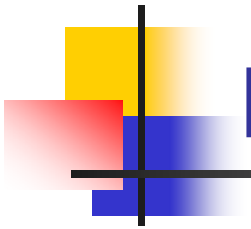
- Multi-Dimensional Search Trees
  - [LN96] W., Litwin, M.-A., Neitman. *k*-RP*$_s$: a Scalable Distributed Data Structure for High Performance Multi-Attribute Access, IEEE PDIS'96
  - [Na96] E., Nardelli. Distributed *k*-d Trees, SCCC'96
  - [NBP98] E., Nardelli, F., Barillari, M., Pepe. Distributed Searching of Multidimensional Data: a Performance Evaluation Study, *JPDC '*98
  - [dPN00] A., di Pasquale, E., Nardelli. Distributed Searching of *k*-dimensional Data, WDAS'00
  - [dPN00] A., di Pasquale, E., Nardelli. Distributed Searching of *k*-dimensional Data with almost Constant Costs, ADBIS'00



k- d Trees

- Introduced by Bentley [*Comm. ACM* '75]
  - As an improvement over quad-Trees
  - Are defined as recursively built binary search trees, based on data decomposition:

    A point $p \in S$ is chosen to be associated with the root ρ of tree. The first coordinate p_{x1} of p divides S into two disjoint subsets S_1, S_2
    - the first subset includes all points $p' \in S$ such that $p'_{x1} \leq p_{x1}$
    - the second one consists of every point $p'' \in S$ with $p''_{x1} > p_{x1}$.
  - S_1 is related with the left child of ρ and S_2 with the right one.
  - Both subsets are split with respect to the second coordinate x_2 and so on.
  - After splitting with respect to the *d*-th coordinate, we continue with the first coordinate.



Pseudo *k*-d Trees

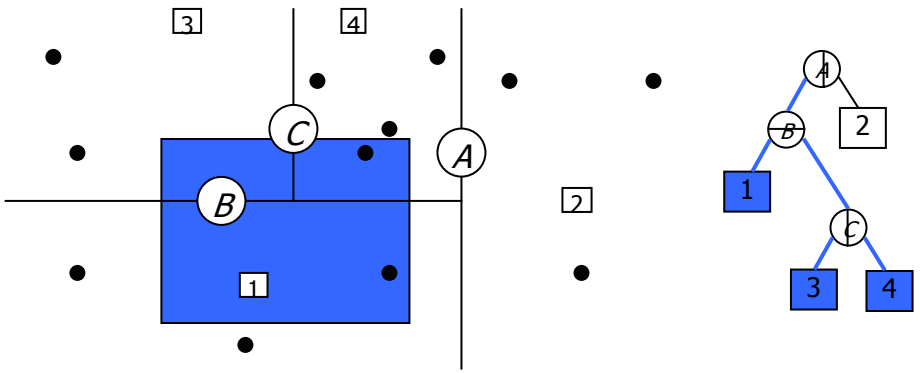
- Insertion

  After searching for the appropriate leaf l, corresponding to the finest (smallest) space subdivision that contains the point to be inserted, we make l internal node with two leaf-children. Using the appropriate coordinate for the splitting, one leaf accommodates the new point and the other the 'occupant' of l.

Bucketed version of *k*-d Trees

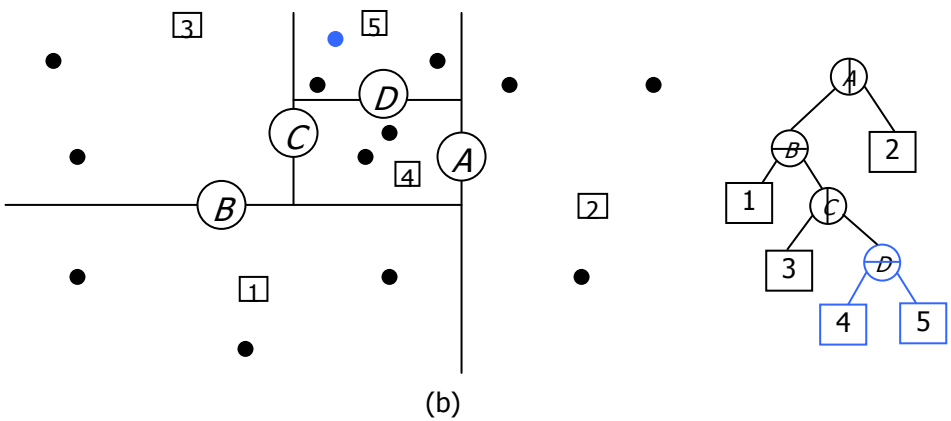
- In order to use the scheme in the SDDS context we will expand the leaves into buckets so that they can accommodate instead of just one point, at most b [LM96, Na96]:
  - if the involved leaf l stores less than b points, then it accommodates the new point without any further alterations;
  - else, we split the bucket, distributing evenly the 'old' points and the new one between l and a newly allocated bucket.

Instance of (bucketed) *k*-d Trees



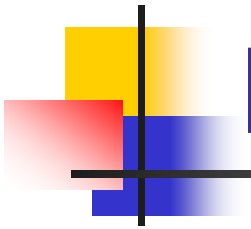
(a)

(a) Range query



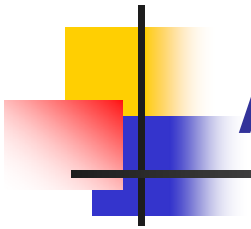
(b)

(b) Insertion of new point



Distribution

- Servers manage
  - One data bucket
  - Auxiliary indexing information
- Clients manage
  - Auxiliary indexing information



A fact

- The resulting random ‘routing’ tree performance depends on data distribution
- This calls out for auxiliary tree structure on top of random ‘routing’ tree in order to create ‘path’ nodes, extending rooted Trees of G.N. Frederickson [*J. Algorithms* ’97]

Definitions

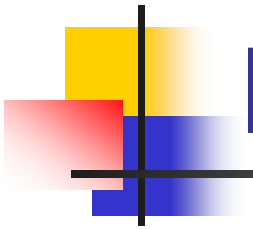
- Let r be a rectangle split w.r.t. x_i-axis into two (disjoint) rectangles r_1,r_2; i.e., $r=r_1 \cup r_2$. We say that x_i is the *definition axis* of r_1,r_2
- Let r_1,r_2 be two multi-dimensional rectangles of the same definition axis x_i. We consider that

  $$r_1 \preceq_{xi} r_2$$

  if their projections to x_i-axis satisfy the relation $\pi_{xi}(r_1) \leq_{xi} \pi_{xi}(r_2)$. That is,
  - either $\pi_{xi}(r_1)$ is contained into $\pi_{xi}(r_2)$, or
  - the upper bound $\pi_{xi}(r_1)_u$ of the first projected interval is less than or equal to the lower bound $\pi_{xi}(r_2)_l$ of the second one.
- $\preceq_{xi}$ defines a partial order over the universe of multi-dimensional rectangles.
- Let r_1,r_2 be two multi-dimensional rectangles. We consider that

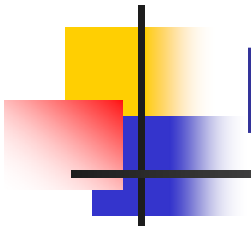
  $$r_1 \preceq r_2$$

  if $\exists$ min axis x_j : $\pi_{xi}(r_1) \leq_{xj} \pi_{xi}(r_2)$.



Description

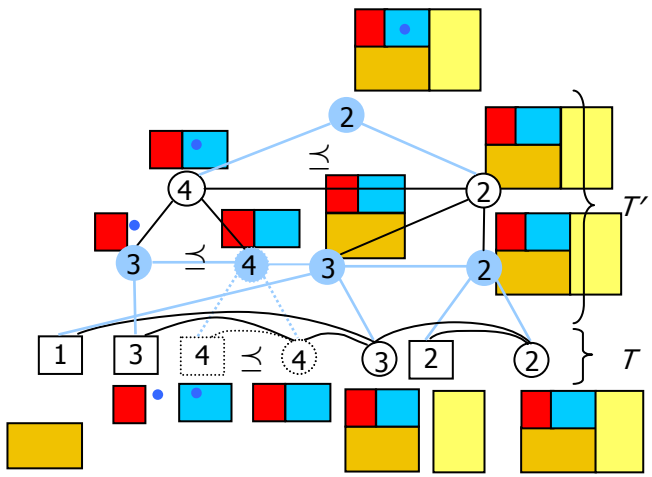
- On the topological order defined by $\preceq$ over the nodes of the original tree T, we can merge adjacent nodes, whose extents respect the $\preceq$ relation, level by level, until we come up with just one node.
- Whenever we want to insert a new leaf f' due to overflow of an old one f:
  - firstly, we add it,
  - and then we follow the upward path from f in T', merging nodes so that the per level topological node sorting is maintained.
- This scheme consists a multi-dimensional extension of Bozanis, Manolopoulos [WDAS'02].



Description (Cont.)

- Search operation is reduced to following upward/downward/sibling pointers until finding the pertinent server
- Clients:
  - Consult their own view, refreshing it, for example, like [KW95,Na96,LN97]
  - Termination of search can be accomplished e.g. by logical volumes of [NBP98]

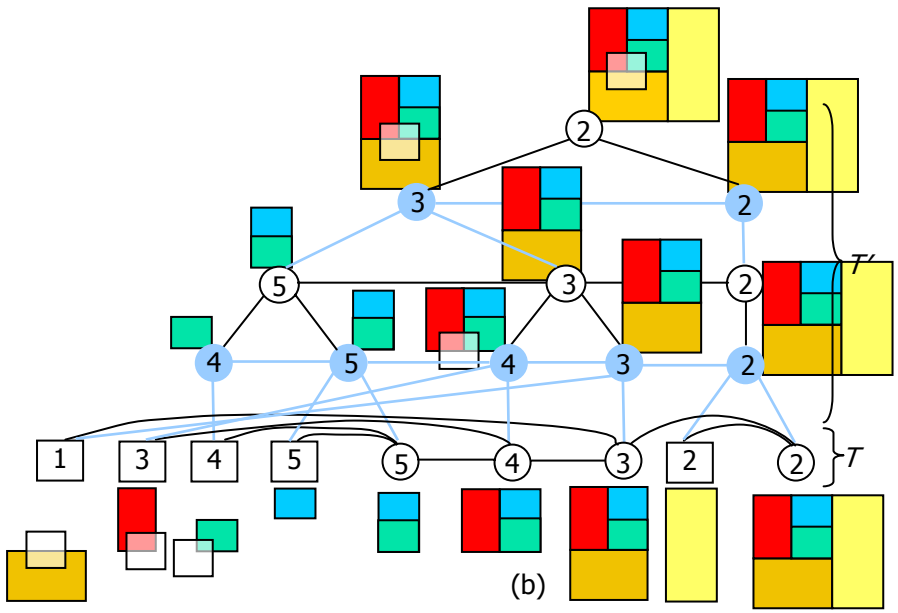
Description (Cont.)

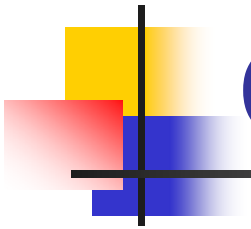


(a)

- Linear Space Complexity (all newly generated nodes are stored in the new server s' allocated to leaf f')
- The out-degree is logarithmic, in the worst case

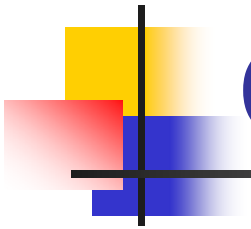
Description (Cont.)





Complexity Bounds

- With linear Auxiliary Space
  - Insertion time: O(log n)
  - Search time:
    - Single element query O(log n)
    - Range query O(log n+k), k =# servers covering query area
- With O(n log n) Auxiliary Space [BM02]
  - Insertion time: O(log n)
  - Search time:
    - Single query O(1)
    - Range query O(k), k =# servers covering query area



Conclusions

- Distributed *k*-d Trees
  - is an alternative proposal to both [Na96] and [LN96]
  - employs simple operations
  - trade-offs between time and space
- Future investigation
  - Experimentation
  - Extension of the method to quad-Trees