Interactive Poster: Exploring Block Access Patterns of Native XML Storage

Halldór Janetzko, Daniel A. Keim, Marc Kramis, Florian Mansmann, and Marcel Waldvogel

University of Konstanz, Germany
Email: {janetzko, keim, kramis, mansmann, waldvogel}@inf.uni-konstanz.de

ABSTRACT

Recent block-based native XML storage systems such as IDEFIX touch blocks according to the XQuery engine's execution plan. The resulting access patterns are virtually unknown and potentially cause many expensive disk seeks. Visualization comes to the rescue when extensive log files must be analyzed – a tedious and difficult task. The dynamic time-based block-touch animation as well as the static block-type information of VISUAL IDEFIX foster the insight into the performance-critical internals of the XML storage and help to optimize both the block layout and the XQuery engine to speed up queries.

Keywords: native XML storage, block layout, access patterns, visualization

Index Terms: E.1 [Data]: Data Structures—; I.3.8 [Computing Methodologies]: Computer Graphics—Applications

1 INTRODUCTION

Efficient storage and retrieval of XML data is a challenging research area. IDEFIX [1] competes in this field by storing native XML data in blocks of a random-access device. The XML data is organized in three main data structures as shown in Fig. 1. The node list stores the tree structure of the XML data. Each element, attribute, or text node appears in the node list in `pre` order which corresponds to the `pre` order (i. e. first-depth) traversal of the XML tree [2]. A prefix tree locates each entry in the node list by its `pre` position. Tag and attribute names as well as attribute and text values are stored in the name and value map respectively. Both names and values are accessible through so-called tries. All tries are stored in metadata blocks while the node list, name, and value map entries are stored in data blocks.

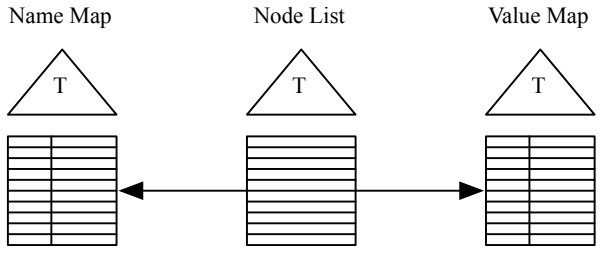

Figure 1: Three main data structures of IDEFIX consisting of the node list (tightly packs all XML nodes) as well as the name and value maps which source strings out of the node list. `T` denotes a trie.

During the evaluation of the XMark benchmark [4] we run different queries against IDEFIX. The limiting performance factor, random disk I/O, clearly asked for an optimized layout of the blocks to efficiently support different access patterns. The analysis of the block layout as well as the access patterns is effectively supported by the visualization tool described in the next section.

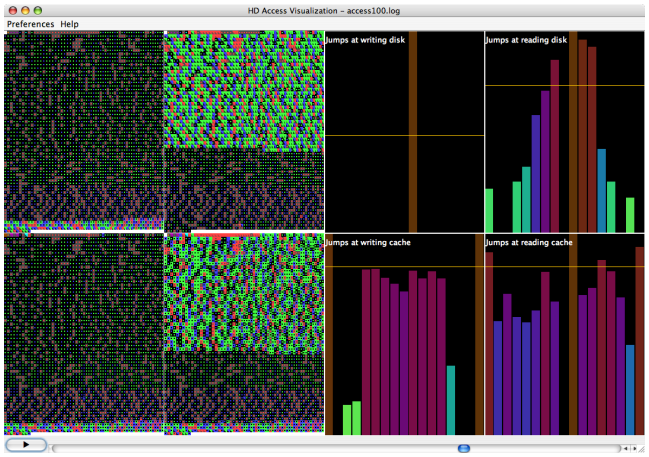

Figure 2: VISUAL IDEFIX is a tool to explore block access patterns of the native XML storage IDEFIX. The tool consists of four block access diagrams (read disk, write disk, read cache, write cache) and four jump distance histograms. The detailed view of the disk read operations shows touches of both metadata and data blocks (gray vs. black background) by highlighting them over time. The jump histogram (top right) reveals that the storage is already fairly optimized for the shown query as the distribution of disk read operations is focused around the center, which indicates that data is read mostly sequentially, avoiding expensive seek operations.

2 VISUAL IDEFIX

VISUAL IDEFIX is a visualization tool that supports the analysis of block layout and access patterns of the native XML storage IDEFIX (see Fig. 2). The basic idea is to see where data and metadata are stored on the disk to evaluate both the locality of block storage as well as the correctness of block access of XML queries. Furthermore, our tool animates the execution of queries by visualizing block access operations on disk and cache to efficiently support the system engineer in his task to verify hypotheses about the system's behaviour as well as to formulate novel hypotheses and search for their causes.

The visualization is designed in a way that each storage block is represented through a small rectangular icon on the screen. Color depicts the type of block as illustrated in Fig. 3. We distinguish between three states of a block (*untouched*, *recently touched*, and *touched*), which are represented through distinct icons. Note that there are variations of the icon for *recently touched*; the more a block is touched, the more intense is the color of the bottom left corner of the icon. Over time, icons of *recently touched* blocks continually fade out until they reach the state of *touched*. The strong

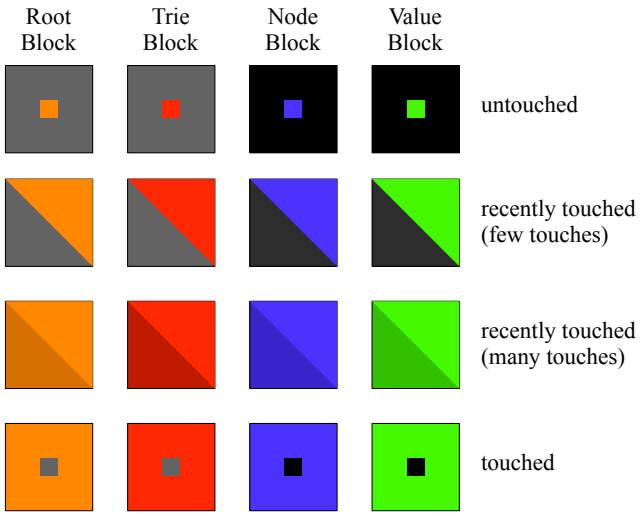

Figure 3: Icons for metadata (gray) and data blocks (black). Variations of the base icon for each block type are utilized to show different usage states.

contrast between "untouched icons" and "touched icons" enables us to easily distinguish between used and unused areas of disk and cache during query execution. Conceptually, we classify the blocks into two groups: data (i.e., XML nodes and their values) and metadata (i.e., root and trie) blocks. Black (gray) is employed as background for the data (metadata) icons.

Current random-access devices with block-oriented interfaces provide a logical block order from the `0`th to the `n`th block. We opted for a line-by-line arrangement of the blocks in our block access diagrams, alternating in forward and backward direction to better preserve visual clusters of subsequent blocks. It is possible to retrieve details (i.e., block number and number of touches) about each block by clicking on it within the diagram. The actual arrangement of the icons is based on a recursive pattern [3] implementation. We abandoned more complicated parameter settings of the recursive pattern due to the cognitive overhead required for proper interpretation of the patterns.

The arrangement of data and metadata blocks already gives a feeling for the effectiveness of the used allocation scheme. In Fig. 4, the upper part is dominated by value blocks (green) with relatively few trie blocks (red), which store metadata of the index structure, whereas the middle part shows proportionally more trie and node blocks (blue). Whether this partitioning is good or bad depends largely on the characteristics of queries for which the storage should be optimized.

In addition to the block access diagrams, VISUAL IDEFIX offers linearly and logarithmically scaled histograms that are used to display counts of block jump distances (see Fig. 2). The value ranges of the histogram bins increase from the middle bin to the outer bins. We implemented a normal mode to investigate jumps for short time spans as well as a cumulative mode enabling analysis of larger time spans. The horizontal bar denotes the average bin size for each histogram.

3 Conclusion

The visualization of block layout and access patterns of the native XML storage IDEFIX not only endows us with an excellent tool to analyze different queries, but also gives many valuable hints how to organize blocks more efficiently for various workloads. One benefit of the visualization has been the identification of a query eagerly touching unnecessary blocks.

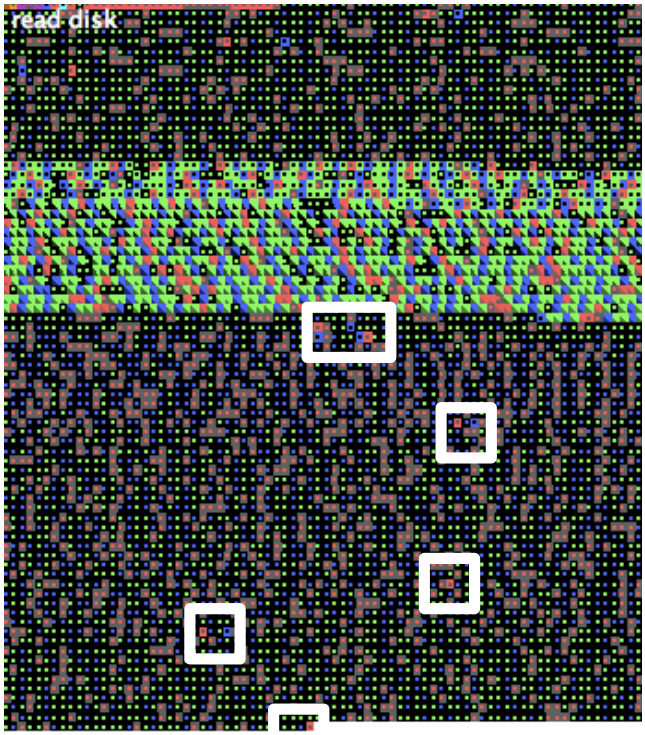

Figure 4: Outlier blocks touched in the course of a query are directly visible and reveal valuable insight to the system engineer for further optimization.

VISUAL IDEFIX proofs to be very instructive for students to understand a block-based native XML storage system. Future work will include visualization of the internal block fragmentation due to updates, a faster visualization that allows for online observation of IDEFIX as well as interaction possibilities to enlarge regions of interest.

Acknowledgement

This work was partially funded by the German Research Foundation (DFG) under grant GK-1042, Explorative Analysis and Visualization of Large Information Spaces, University of Konstanz.

References

[1] C. Grün, A. Holupirek, M. Kramis, M. H. Scholl, and M. Waldvogel. Pushing XPath Accelerator to its Limits. In *Proceedings of EXPDB 2006*, Chicago, IL, USA, 2006.

[2] T. Grust. Accelerating XPath Location Steps. In *Proc. of ACM SIGMOD/PODS Int'l Conference on Management of Data/Principles of Database Systems*, pages 109–120, Madison, Wisconsin, USA, June 2002.

[3] D. A. Keim, M. Ankerst, and H.-P. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings of Sixth IEEE Visualization 1995 (VIS'95)*, 1995.

[4] A. R. Schmidt, F. Waas, et al. XMark: A Benchmark for XML Data Management. In *Proc. of Int'l Conference on Very Large Data Bases (VLDB)*, pages 974–985, Hong Kong, China, Aug. 2002.