Query-Driven Visualization of Large Data Sets

Arnika kaithwas, Krishna Kumar Bais, Pradeep Sahu

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur (IITK)

email: {}@cse.iitk.ac.in

Abstract

- The aim of this paper is to make the analysis and visualization process more efficient and manageable, especially as the size of datasets continues to grow.
- This abstract introduces a new approach to analyzing large and complex visual datasets, particularly in scientific research, by focusing on "interesting" subsets of data.
- Previous research mainly focused on range queries within scalar fields. And these methods have limitations when dealing with more complex, multidimensional, and multivariate queries.
- The authors propose a new tool called "DEX" (short for Dextrous Data Explorer) to address these limitations which uses a technique called “Bitmap Indexing”.
- In addition to benchmark performance and analysis, the authors applied DEX to a real-world problem in combustion simulation data analysis.

Introduction

- **Motivation and Problem Statement:** Managing and analyzing large scientific datasets is a major challenge in research, with the sheer volume of data making it difficult to focus on the most relevant subsets for analysis and visualization.
- The authors proposes a methodology known as “Query-Driven Visualization” and to implement this they combine the state of arts scientific data management technology with visualization tools.
- The existing methods like isosurface acceleration are limited to single-valued search criteria and cannot handle complex, multidimensional queries, restricting their applicability in more intricate data analysis scenarios.
- The experimental results show that DEX outperforms the well-known accelerated isosurfacing algorithm by factors ranging from 137% to 392% depending upon the source data.

Related Work

Large and Complex Data Visualization

- Scalable Visualization technique: As data resolution increases researchers have developed techniques to increase the capacity of the visualization pipeline through parallelism.
- Two examples of scalable applications are VisIt and ParaView.
- It increases the processing load on human observer and do not necessarily aid in the understanding of large and complex datasets.

- Data Simplification and Reduction: focuses on reducing the amount of data that needs to be processed and visualized.
- There are also automatic feature detection and data-mining techniques that are useful to confirm the presence and absence of known phenomena but might miss unexpected features.

Query-Driven Visualization

- Highlights the data that a user defines “interesting”.

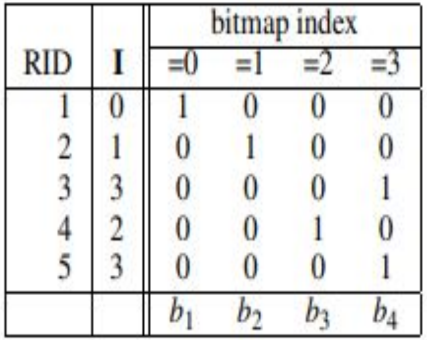
- **VisDB System:** helps to visualize data by filtering and ranking data based on relevance to a user-defined query.
- The result is O(n) memory and processing complexity for each and every query.
- Relevance Factor: VisDB uses statistical heuristics to rank and cluster similar data together. And particularly well suited for use with qualitative and fuzzy queries.

- **Scout Software System**: Provides the ability to perform expression based queries using a simple programming language along with visualization on GPU.
- For 2D data, it renders a single quadrilateral, and for 3D data, it uses view-aligned slices to perform direct volume rendering.

- Both the systems have O(n) complexity.
- “Visual Analytics” has been coined to describe a set of activities, it’s goal simplify complex and multidimensional data by showing only information that is relevant to the user’s current inquiry.

• Bitmap Indices

- An efficient index data structures for accelerating multi-dimensional range queries for read-only or read-mostly datasets.
- Bitwise logical operations are used to handle multidimensional and multivariate queries.
- A potential downside of bitmap indices is that they can require a lot of storage but can be compressed and the methods are-
  a. Byte aligned Bitmap Code(BBC)
  b. Word-Aligned Hybrid(WAH)
- It has binning strategies to speed up multidimensional queries for high cardinality attributes.



RID	I	bitmap index			
		=0	=1	=2	=3
1	0	1	0	0	0
2	1	0	1	0	0
3	3	0	0	0	1
4	2	0	0	1	0
5	3	0	0	0	1
		b_1	b_2	b_3	b_4

Figure 1: A sample bitmap index where RID is the record ID and **I** is the integer attribute with values in the range of 0 to 3.

- **Isosurfaces**

  - The canonical isosurface algorithm consist of two broad processing step:
    1. Find the cells that contain isosurfaces
    2. Generate the geometry for the isosurface:
  - Marching Cubes algorithm is the earliest methods for isosurface extraction having O(*n*) complexity in searching.
  - To improve the search process there are several techniques:
    1. Octree
    2. Span-space Searches
    3. 2D Regular lattice
    4. Out-of-Core approach
  - None of these algorithms would be effective to handle multidimensional, multivariate queries or dynamic data.

Dex Architecture

The implementation consists of four broad phases:

- ❑ Index Contruction
- ❑ Index searching
- ❑ Region Growing
- ❑ Geometry Contruction

Index Construction

There are two primary factors of concern with respect to constructing the index structures used to accelerate data queries:

1. **Computational Complexity**
   - In case of **Bitmap Indices** each data value is examined and a corresponding bitmap code is added to the index. The computational complexity of this operation is O(*n*).
   - Tree-based methods, such as B-trees or quad-trees require either a complete sort or sorted insertion.

2. **Storage Requirements**
   - in the worst-case scenario, the size of the bitmap index can reach 2n words**.**
   - The size of tree-based structures has an upper bound of O(nlogk n) and tend to require about 4n words of storage.

Index Searching

- The primary strengths of bitmap indices for searching is :
    1) Low computational and storage complexity
    2) Efficient for complex, multidimensional queries
    3) The search is done using FastBit software, which executes queries with a worst-case time complexity of O(*k*).
- Conversion of compressed bitmap into a list of blocks in space. For 3D space data, there are three types of block
    1) Line Segments: A series of connected cells along a line.
    2) Groups of Connected Lines: Multiple connected lines of cells.
    3) Groups of Connected Planes: Entire connected planes of cells
- The process of converting a consecutive group of 1s to a block takes a constant number of machine instructions.
- Converting WAH compressed bitmap to a list of blocks scales linearly with the number of blocks.

Region Growing

- **Region Growing** is the process of taking the individual blocks and grouping them into larger connected regions.
- The goal is to identify which blocks are part of the same spatial region using an algorithm that has an order of complexity that is sublinear with the number of blocks.
- Blocks are considered connected if they touch at a face, edge, or corner (26-connected neighbors).
- Algorithm well regardless of the resulting concavity or convexity of the resulting grown region.

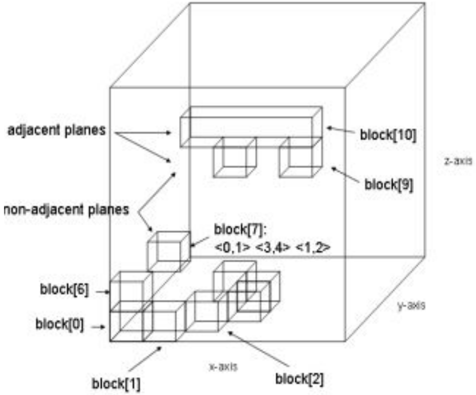
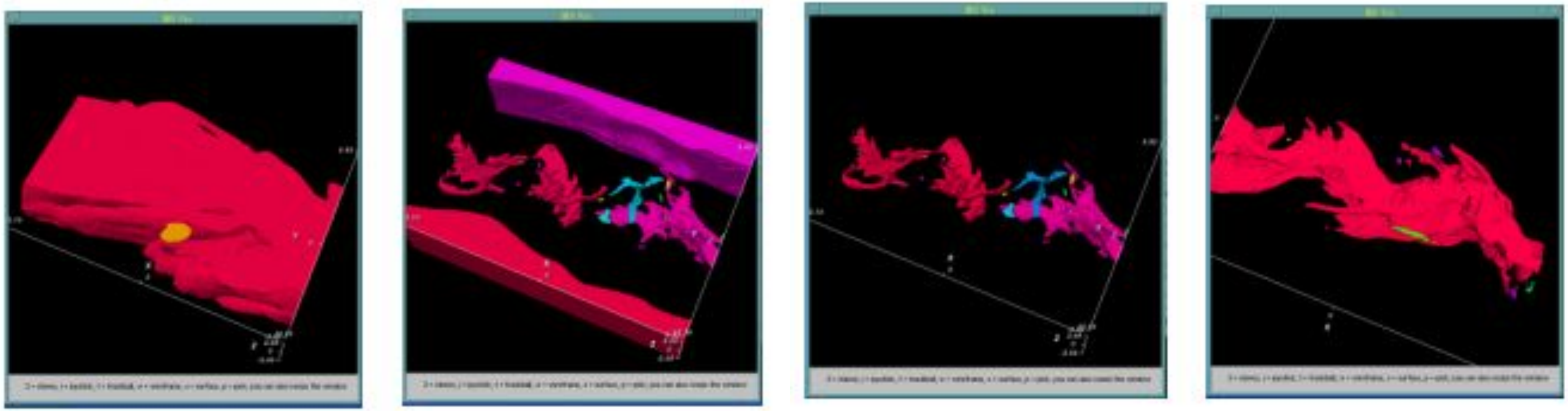


Figure 2: Blocks in space with three spatial dimensions.

Geometry Construction

- **Bounding Boxes:** FastBit uses 3D bounding boxes to represent regions matching a query.
- **Disjoint Issue:** These boxes are topologically disjoint, unsuitable for direct visualization.
- **Mesh Conversion:** The boxes are converted into an unstructured mesh representing the selection’s outer hull.
- **Data Mapping:** This mesh allows accurate mapping of data for visualization.
- **Visualization Use:** The mesh enables further visualization techniques like volume rendering on the selected data.



(a) $CH_4 > 0.3$

(b) $temp < 3$

(c) $CH_4 > 0.3$ AND $temp < 3$

(d) $CH_4 > 0.3$ AND $temp < 4$

Figure 3: A visualization of a combustion analysis dataset displays the cells selected by various user queries. The selected cells are colorized by their region label that is assigned by a 3D region growing algorithm. The image is an example taken from the combustion studies where the goal is to track the ignition kernel of a flame.

Multiresolution Support

- if the query result exceed the available processing resources , FastBit supports returning data at different levels of resolution.
- It encodes indices for hierarchy of resolutions, each provide a downsampled version of the data.
- Bitmap indices estimate the number of cells a query will return, helping choose an appropriate resolution level.
- The current method selects one cell from each 2x2x2 cluster, potentially losing topological detail.
- **Future Improvement:** Plans include multiresolution sampling to preserve topological correctness.

4 EXPERIMENTAL RESULTS

Two key questions addressed:

- What is the cost of creating **bitmap indices**?
- How does the **data search performance** of DEX compare to an isosurface algorithm using a **span-space technique**?

Source data:

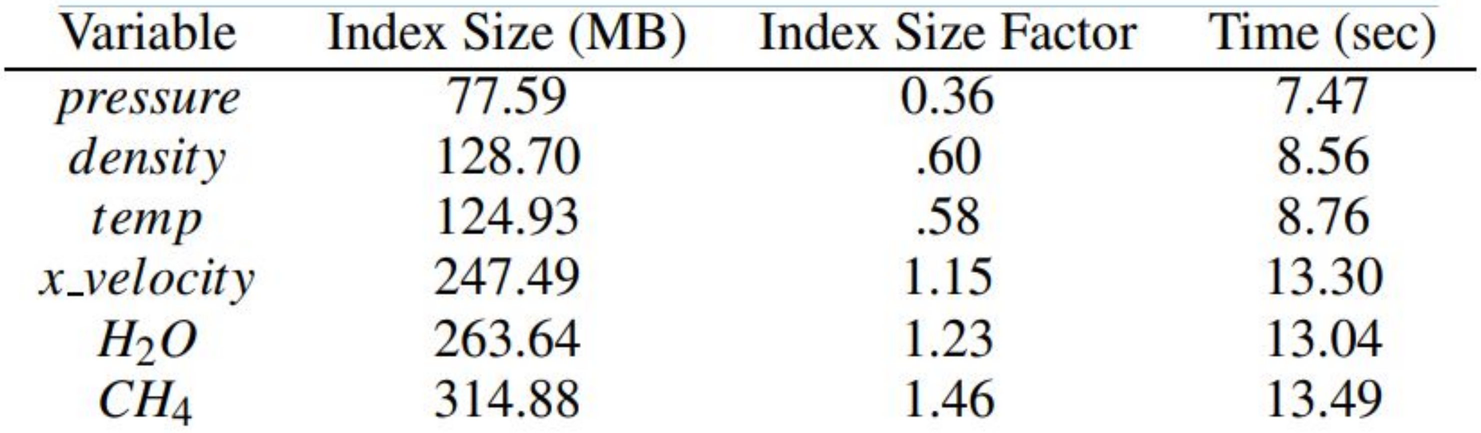
- Multivalued **combustion simulation dataset** with ~56 million data points.
- Grid resolution of **383x383x383** with **38 variables per grid cell**.

Dataset choice:

- Chose this resolution so both DEX and VTK’s Accelerated **Marching Cubes** algorithm could run **entirely in memory** (avoiding memory swapping issues).

Hardware setup:

- **2.8GHz Pentium 4** processor.
- **2GB RAM**.
- **SCSI RAID** with **60MB/s** I/O bandwidth.



Variable	Index Size (MB)	Index Size Factor	Time (sec)
pressure	77.59	0.36	7.47
density	128.70	.60	8.56
temp	124.93	.58	8.76
x_velocity	247.49	1.15	13.30
H_2O	263.64	1.23	13.04
CH_4	314.88	1.46	13.49

Table 1 Overview:

- The table shows the **time and storage requirements** for creating bitmap indices from the original dataset during the index construction phase.

Data Size:

- Each input variable has a size of **214.31 MB** (uncompressed data from a **383x383x383** grid, with 4-byte floating point values).

Bitmap Index:

- The **bitmap index** for each variable is created by compressing 100 range-encoded bins using a technique called **WAH compression**.

Index Size Variation:

- The size of each bitmap index varies depending on the original data. For example:
  - The **pressure index** is **36%** of the original data size.
  - The **CH4 index** is **146%** of the original data size (indicating it's larger than the original data).

Time Taken:

- The table shows the **time in seconds** to create each index from the original data. For example:
  - It took **7.47 seconds** to construct the index for **pressure**.
  - It took **13.49 seconds** to create the index for **CH4**.

Comparison to VTK’s Isosurface Algorithm:

- They were unable to measure the size of the **span-space tree** used in VTK’s isosurface acceleration algorithm, but tree-based structures generally follow a storage complexity of **O(n log k n)**, which means they grow faster in size than the bitmap indices.

4.2 Data Search Performance Analysis

The benchmark comparison between **DEX** and **VTK's Accelerated Marching Cubes** for data search and geometry generation:The benchmark measures two phases of performance:

- **Data Search Phase**:
  - **VTK** uses a **span-space algorithm** to find cells that intersect the isosurface (points where a surface passes through the dataset).
  - **DEX** uses a **bitmap index** to find cells based on a <= operation, locating all data points inside the isosurface, not just those on the surface.
- **Geometry Generation Phase**:
  - **VTK** generates **triangles** (about 2.5 per cell) representing the isosurface.
  - **DEX** generates **12 triangles per cell** (a finite-element hexahedron), which means it generates more geometry overall.

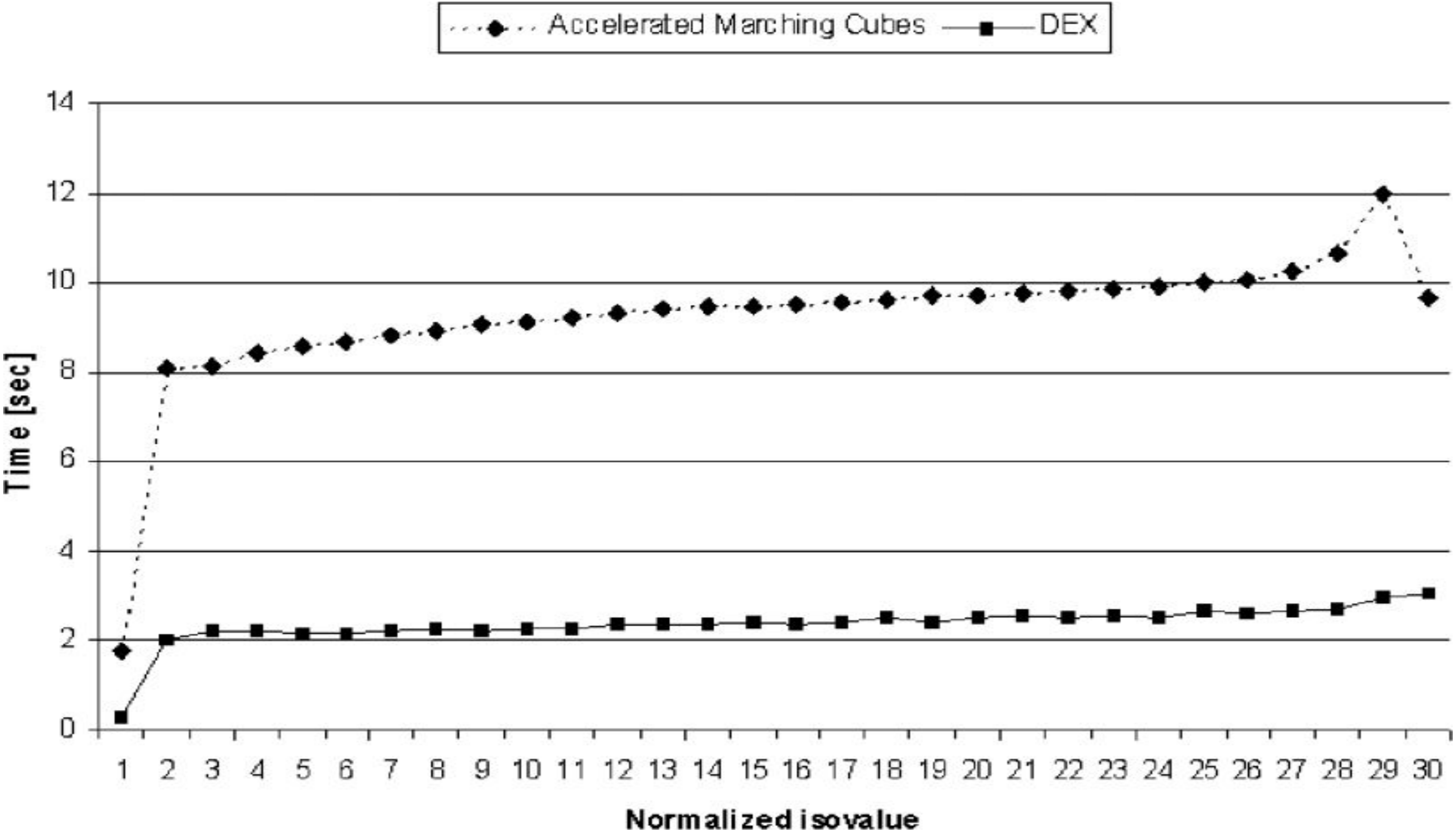
DEX performs better than VTK, producing results faster in both the search and geometry generation phases.

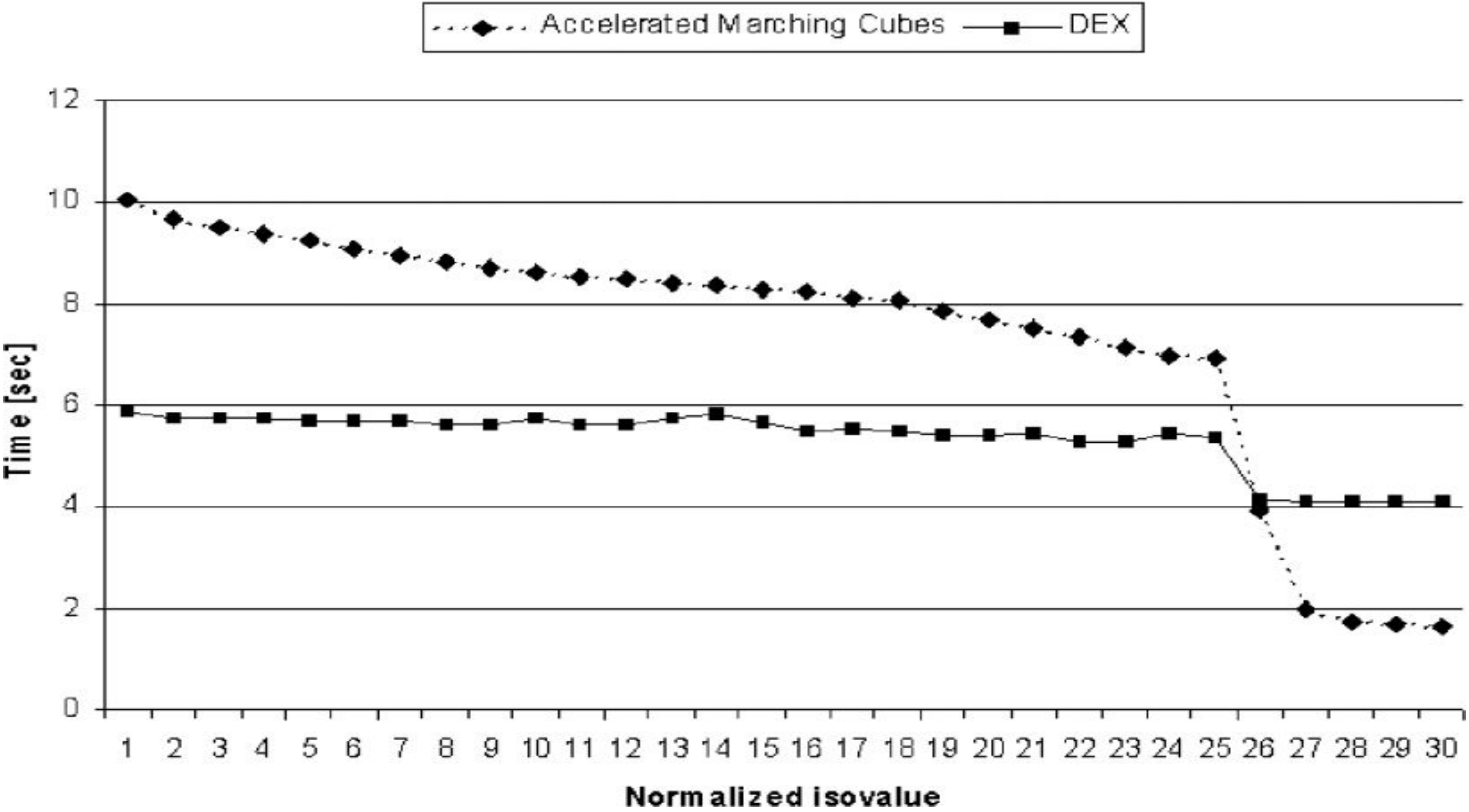
DEX's speedup (improvement over VTK):

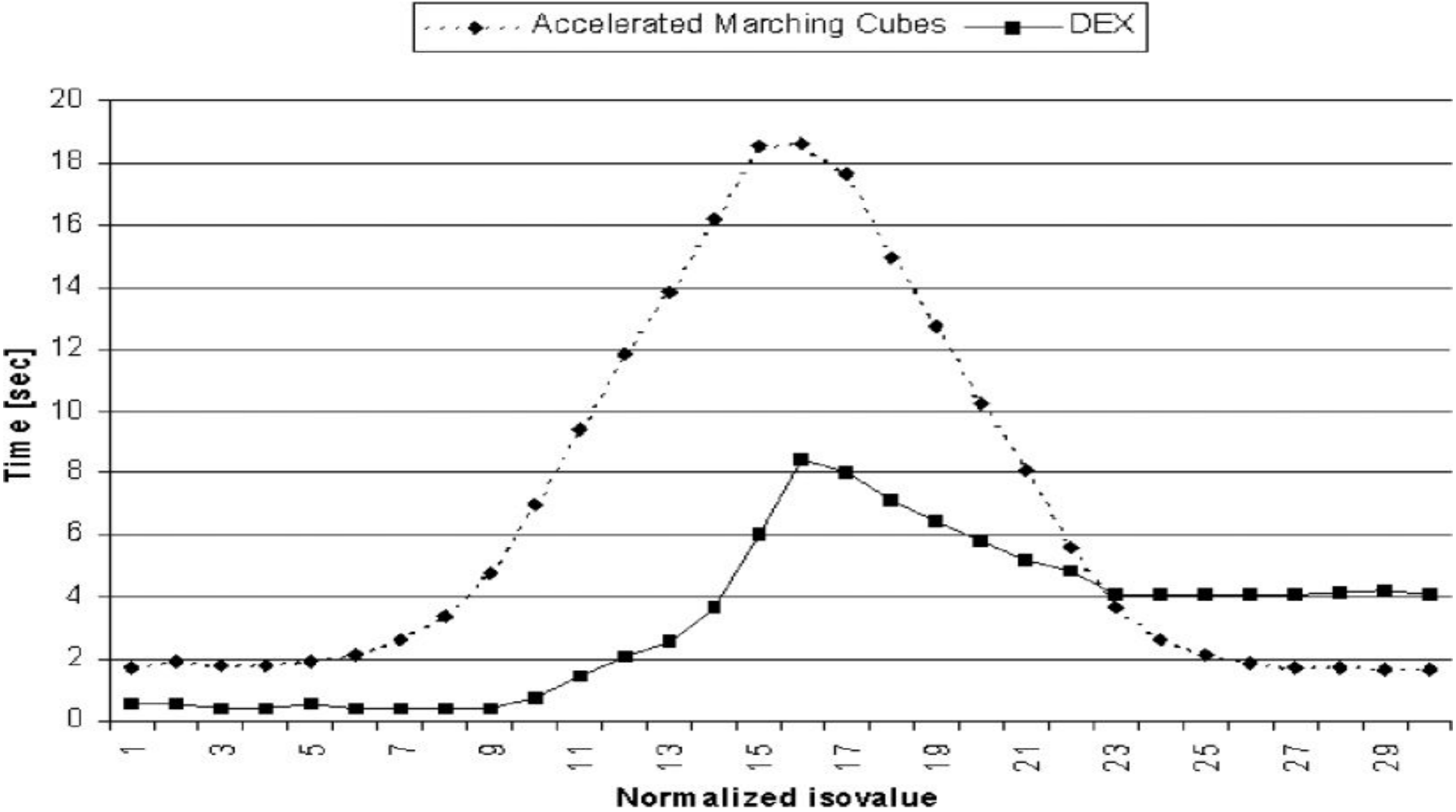
- For the **density** field: **392% faster**
- For the **H2O** field: **137% faster**
- For the **x velocity** field: **204% faster**

Processing Stages in DEX:

- **Index Searching**: Locates the cells that meet the search criteria.
- **Region Labeling**: Labels connected regions within the data that satisfy the criteria.
- **Geometry Construction**: Builds the finite-element geometry for each cell.


Accelerated Marching Cubes
DEX
Time [sec]
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
Normalized isovalue


Accelerated Marching Cubes
DEX
12
10
8
6
4
2
0
Time [sec]
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
Normalized isovalue


Accelerated Marching Cubes
DEX
Time [sec]
20
18
16
14
12
10
8
6
4
2
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
Normalized isovalue

DEX performs better than VTK, producing results faster in both the search and geometry generation phases.

DEX's speedup (improvement over VTK):

- For the **density** field: **392% faster**
- For the **H2O** field: **137% faster**
- For the **x velocity** field: **204% faster**

Processing Stages in DEX:

- **Index Searching**: Locates the cells that meet the search criteria.
- **Region Labeling**: Labels connected regions within the data that satisfy the criteria.
- **Geometry Construction**: Builds the finite-element geometry for each cell

For attributes like **density**, all three stages take roughly equal time. For attributes like **H2O** and **x velocity**, most time is spent in the geometry construction phase. If the cells that meet the criteria are densely packed, finding and labeling them takes less time.

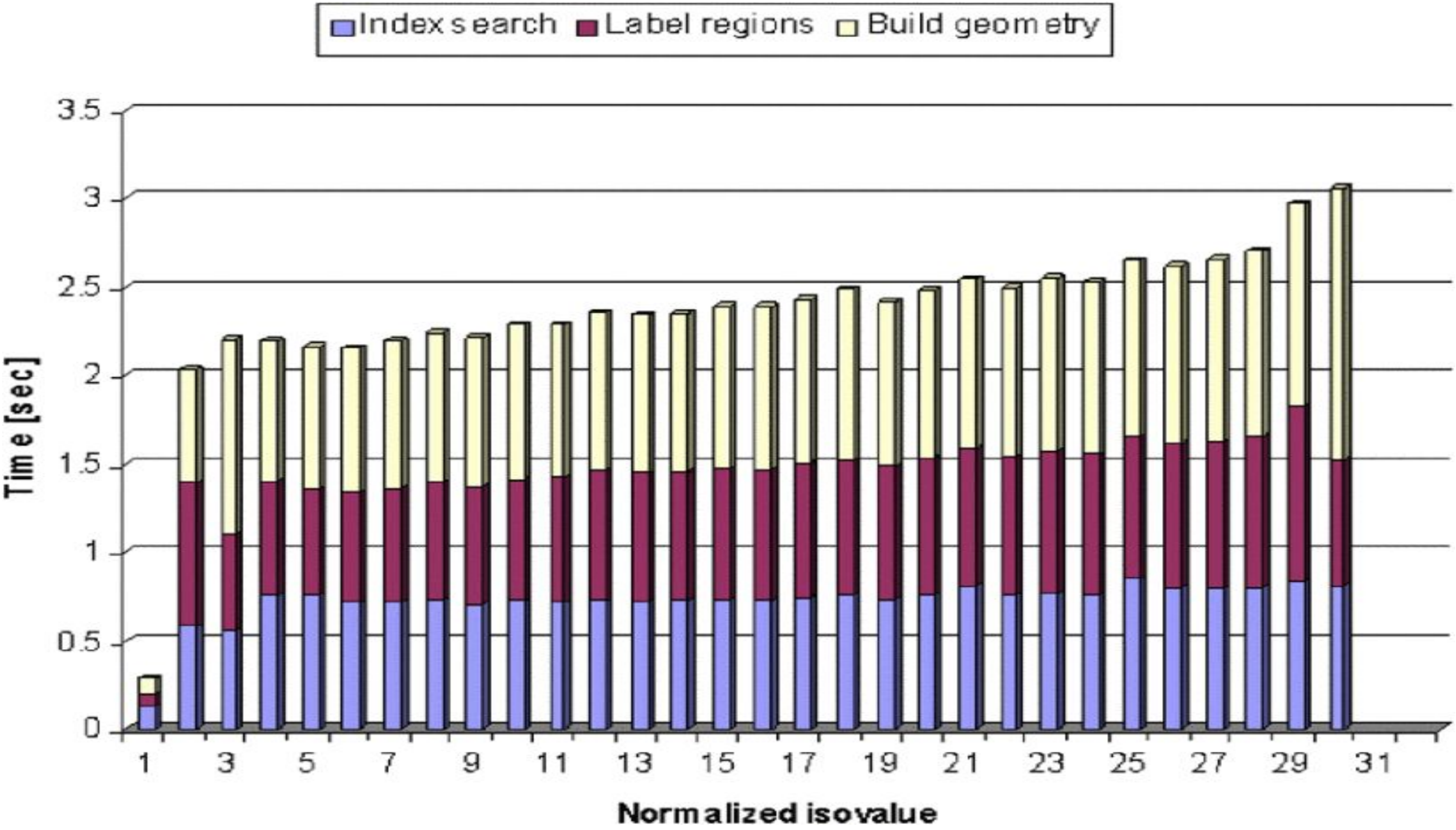
Mainstream Visualization Tools:

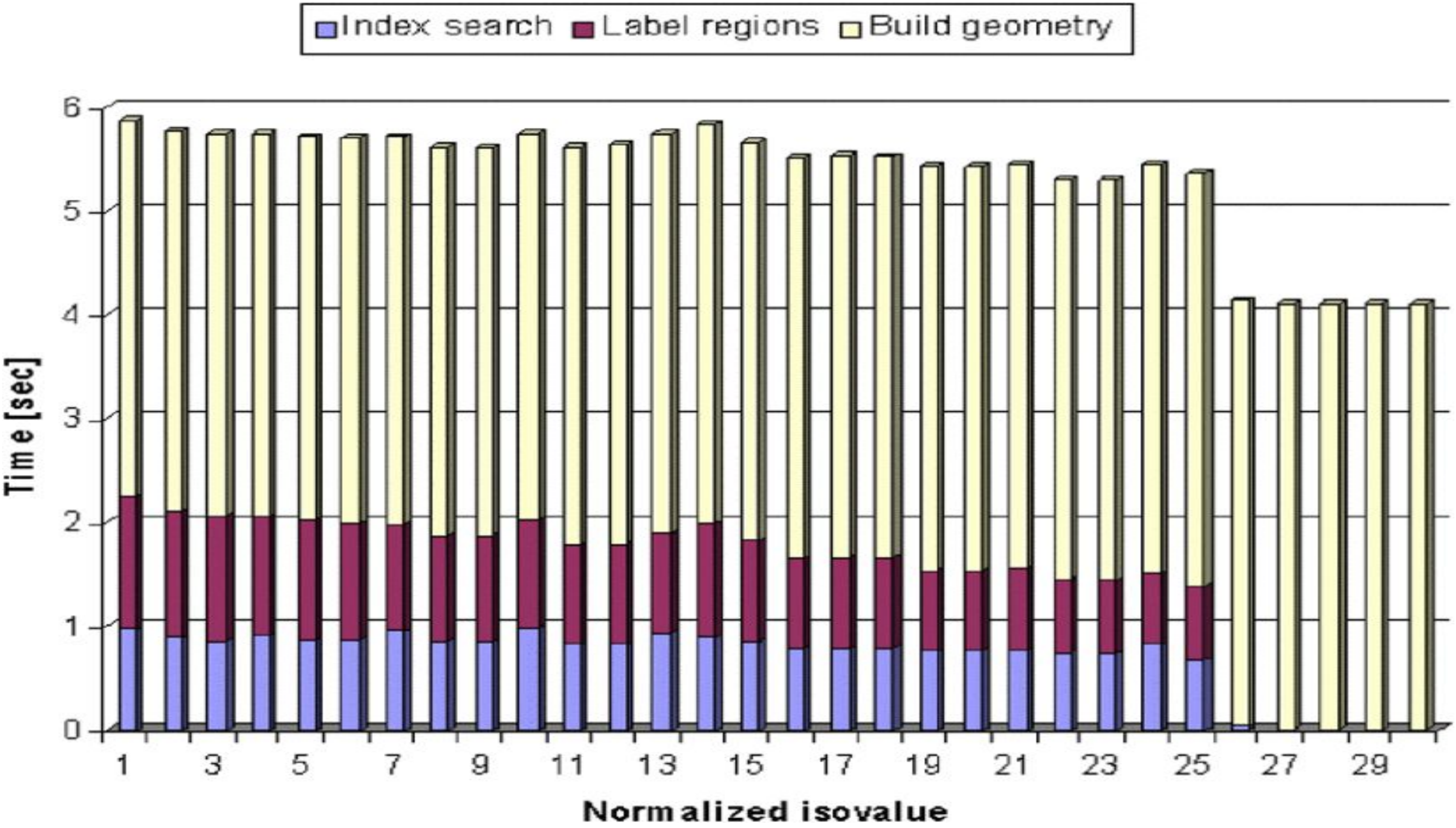
- **Current State**: Fast bitmap indexing technology is currently used in a research tool called DEX.
- **Future Goal**: Integrate this technology into popular visualization tools used by scientists, making it widely available for complex data analysis and visualization.

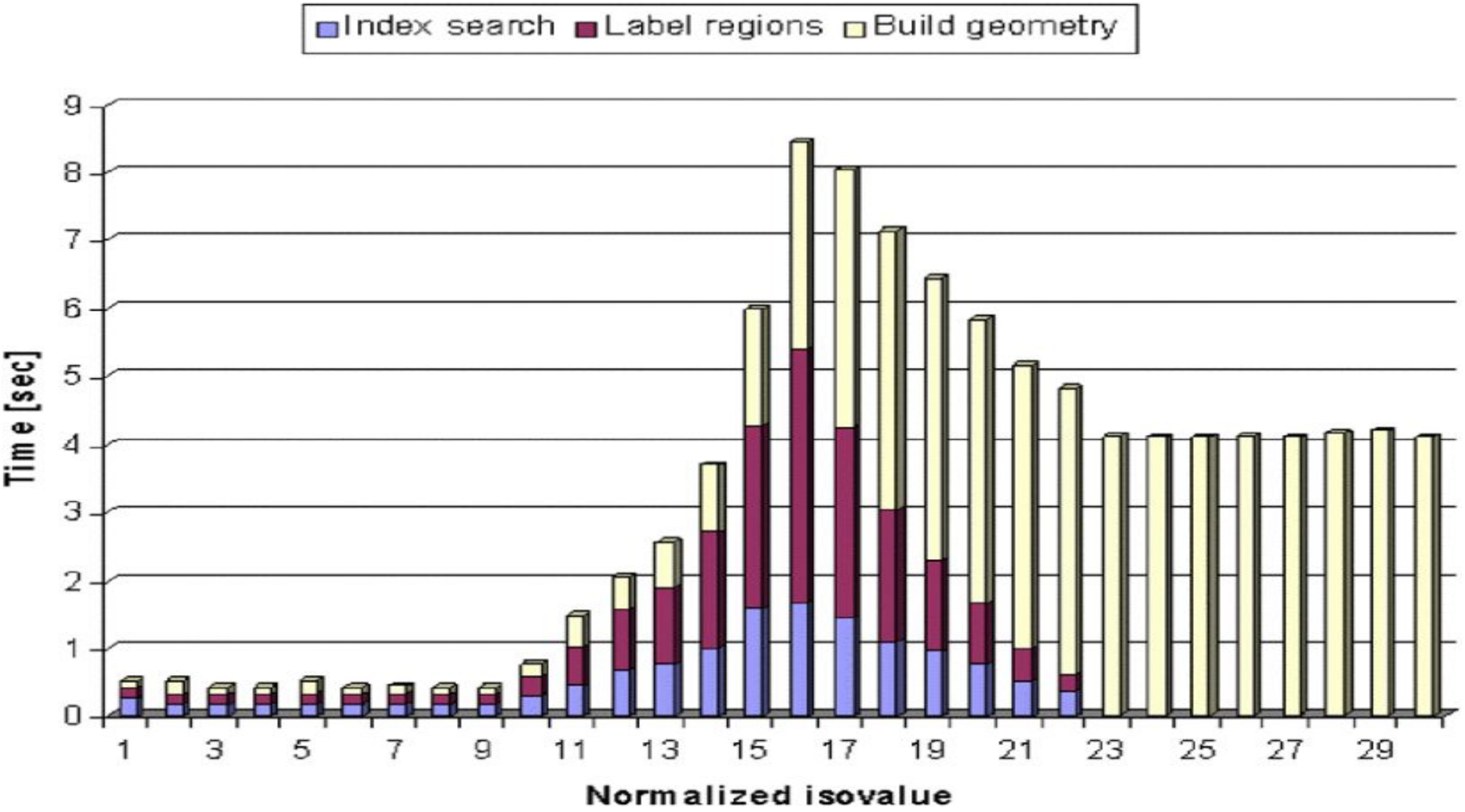
Search Estimation and Multiresolution Queries:

- **Challenge**: Poorly designed queries can end up returning too much data.
- **Solution**: Use FastBit's ability to quickly estimate how many data cells meet a query, allowing for reduced-resolution searches to avoid overwhelming the system.
- **Topology-Preserving Multiresolution Queries**:
    a. **Current Approach**: Early methods create multiresolution data by simply downsampling the data.
    b. **Improved Approach**: Enhance methods to ensure that even if a coarser level of data meets the query requirements, the finer details are preserved. This ensures that important features remain visible at all levels of detail.
- **View Dependent Subsetting**:
    a. **Current Limitation**: Query times can be affected by the complexity of the query.
    b. **Improved Approach**: Encode the location of each grid point in the indices. This allows queries to focus on regions of interest within the current view, improving efficiency by only processing the visible cells.

These future directions aim to make advanced data indexing and searching technologies more practical and effective in real-world applications.


Index search
Label regions
Build geometry
Time [sec]
3.5
3
2.5
2
1.5
1
0.5
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
Normalized isovalue


Index search
Label regions
Build geometry
Time [sec]
0
1
2
3
4
5
6
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
Normalized isovalue


Index search
Label regions
Build geometry
Time [sec]
0
1
2
3
4
5
6
7
8
9
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
Normalized isovalue

Conclusion:

Introduction to DEX:

- DEX is a tool for query-driven visualization that uses advanced data indexing and searching technology.

Benefits Over Previous Methods:

- **Efficiency**: DEX reduces computational and storage complexity compared to older visualization and isosurface algorithms.
- **Index Construction**:
  - **DEX**: Requires linear time and storage, O(n).
  - **Previous Methods**: Required more complex resources, with storage and computational complexity of O(n log n).

Search Performance:

- **DEX**: FastBit, the search component, performs searches with linear complexity, O(k), where k is the number of results.
- **Compared to VTK's Accelerated Marching Cubes**: DEX performs 137% to 392% faster.

Advanced Capabilities:

- **Complex Queries**: DEX can handle complex, multivariate, and multidimensional queries efficiently, which is not possible with traditional tree-based methods.
- **Applications**: This capability has been useful in complex data analysis, including in fields like High Energy Physics.

Modular Architecture:

- **Structure**: DEX is designed with separate stages for index construction, searching, visualization, and rendering.
- **Flexibility**: This modular approach makes it adaptable to various data analysis and visualization tasks.

Thank You