Query-Driven Visualization

Kurt Stockinger, John Wu, John Shalf, and Wes Bethel

Computational Research Division

Lawrence Berkeley National Laboratory

October 2005

VIS 05

MINNEAPOLIS, MN USA

Motivation and Problem Statement

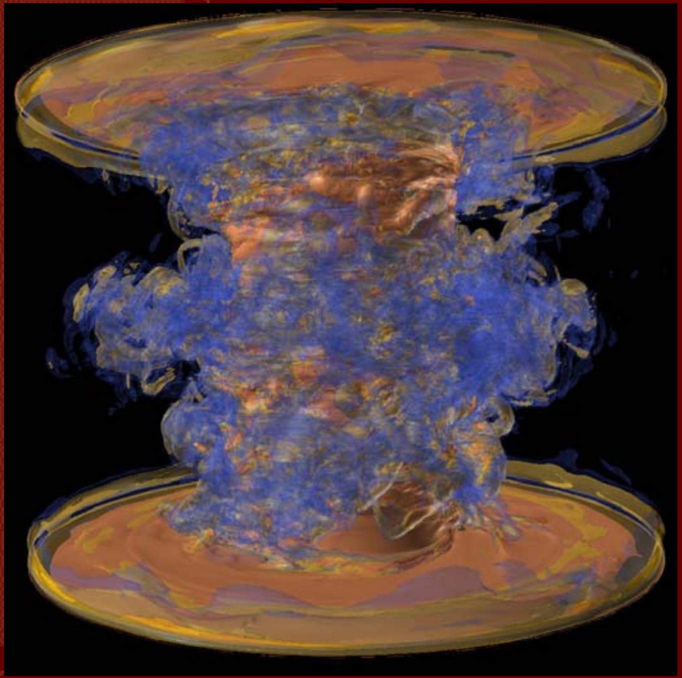
- Too much data.
- Visualization “meat grinders” not especially responsive to needs of scientific research community.
- What scientific users want:
  - Quantitative results
  - Feature detection, tracking, characterization
  - (lots of bullets here omitted)
- See:

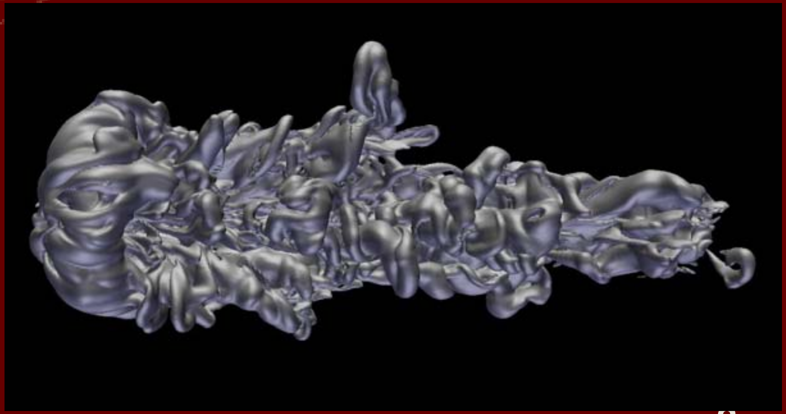
  http://vis.lbl.gov/Publications/2002/VisGreenFindings-LBNL-51699.pdf

  http://www-user.slac.stanford.edu/rmount/dm-workshop-04/Final-report.pdf

Scalable Visualization Isn't Always the Answer

- Premise: rely on humans to interpret more data.
- Decades of work on scalable vis and rendering algorithms.
- Problems:
  - You can't really "see" a Terabyte.
    - Gestalt != Quantitative results
  - Fundamental cognitive science problem: 1+1=3
  - Adding more information to the display may produce a net *loss* in understanding.
  - Throwing more data at the user doesn't solve the "overwhelmed by firehose of data" problem.





Another Approach: Selective Save and Vis

- Premise: only save “interesting” data, throw away the rest.
- Appropriate when focusing vis/analysis to confirm expected
- Opportunity cost: no discovery possible in stuff thrown away

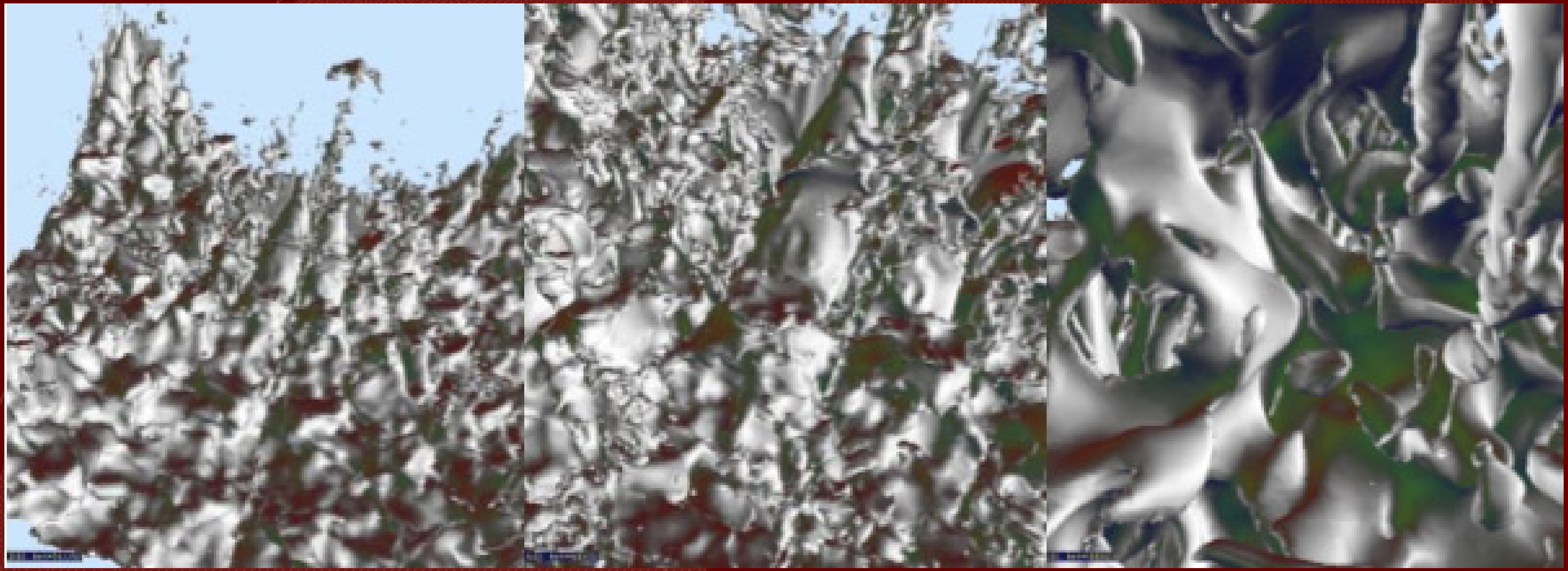


Image source: ASCI TSB Project

What is Query-Driven Visualization?

- Focus visualization processing on subsets of data deemed to be “interesting.”
  - “Interesting” is something the user needs to define.

- Challenges
  - How to define “interesting.”
    - Formulation of definition (domain-specific).
    - Expression of definition (semantic).
  - Find interesting data quickly (SDM).
  - Effective visual presentation of “interesting data” (Vis).
  - Architectures/deployment that complements existing visualization algorithms and applications (CS).

Query-Driven Visualization

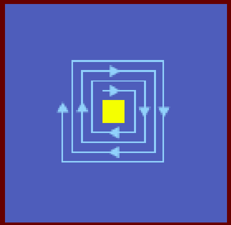
- Our paper's contribution:
  - Find interesting data quickly.
    - Leverage technology from SDM community for visualization.
    - Performance analysis.
  - Architecture: a general approach broadly applicable to most data and visualization applications (plays nicely with others).

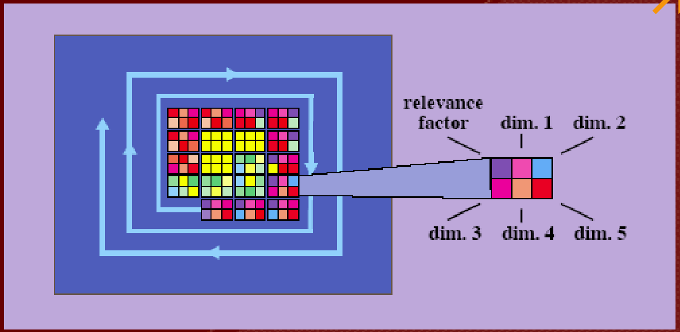
- Topics for another day:
  - Assisted query posing.
  - Effective visualization techniques for query results.

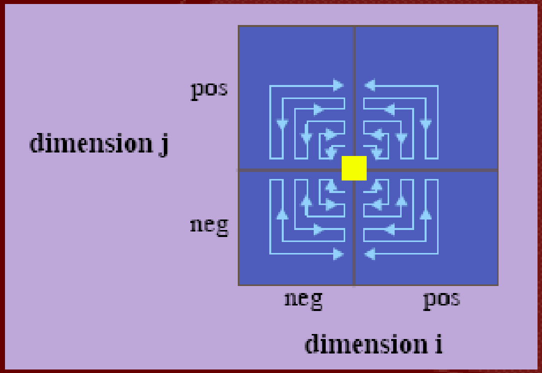
Related Work

- Query-Driven Visualization
  - VisDB – Keim & Kriegel, 1994.
    http://www.dbs.informatik.uni-muenchen.de/dbs/projekt/visdb/visdb.html
  - Demand Driven Visualization. Moran & Henze, 1999.
  - Scout – McCormick et. al., 2004.
- Finding Data Quickly
  - Traditional SDM: relational database systems; tree-based structures, bitmap indices.
  - Visualization: isocontouring algorithms:
    - Marching cubes
    - Octrees – Wilhelms & Gelder, 1992.
    - Span-space methods:
      - NOISE – Livnat, et. al., 1996.
      - ISSUE – Shen, et. al., 1996.
      - Interval Tree – Cignoni et. al., 1996.

VisDB – Keim and Kriegel, 1994



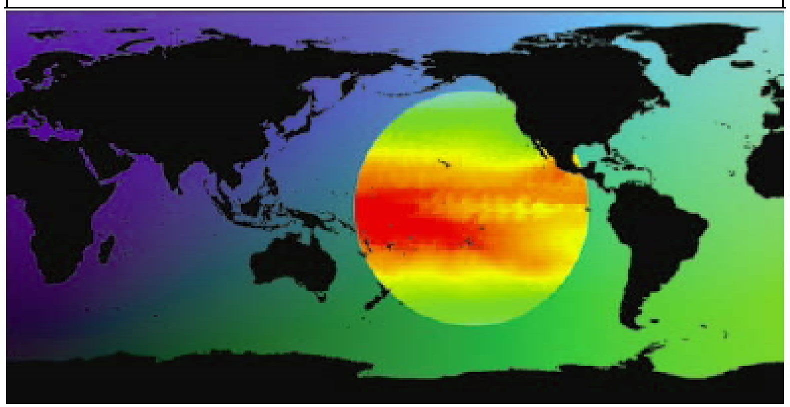




- Motivation: assist in specification of query formulation.
- Approach: rank-ordered query results.
- How:
  - For each data point [i], compute a "relevance factor" indicating how closely data point [i] matches the query (distance).
  - Compute statistical moments.
  - Sort all relevance factors, display in sorted relevance order or by colorizing relevance ranking.
- For n data values, O*(n)* complexity.

Scout – McCormick et. al., 2004

```
// Compute the distance from our location (i,j) to the center
// of the circle clip region at (2400, 1000).
float radius = sqrt(pow(abs(2400-i),2) + pow(abs(1000-j),2));
where (land == 1)
  image = 0; // Render land as black.
else where (radius < 600) // Color by pt within the circle.
  image = colormap[positionsof(colormap) * norm(pt)];
else
  // Color by spatial location. dimof() returns the dimension
  // of pt along the given axis index (0: x axis, 1: y axis).
  image = rgba(0, i/dimof(pt, 0), j/dimof(pt, 1), 1);
```



- Motivation: interactive, expression-based queries.
- How: data-parallel language that executes on the GPU.
- For *n* data points, O(*n*) complexity.
- *N* will be small, though: limited GPU memory.
- Other: floating point resolution on the GPU.

Query-Driven Visualization: Summary

- VisDB:
  - O(*n*) processing time for each query.
  - Data presented in relevance order, reduced in part by quartile culling.
  - Helpful for guiding queries.
- Demand-Driven Visualization:
  - Shown effective for subset selection based upon spatial characteristics rather than data characteristics.
- Scout:
  - High performance (GPU-based) subsetting, expressive data-parallel language.
  - Limited memory, floating-point resolution.
  - Output is imagery rather than data suitable for external use.

Finding Data Quickly

- Isosurface algorithms:
  - Nice summary in: Sutton et. al., A Case Study of Isosurface Extraction Algorithm Performance *2nd Joint Eurographics-IEEE TCCG Symposium on Visualization*, May. 2000
  - For *n* data values and *k* cells intersecting the surface:
    - Marching Cubes: O(*n*)
    - Octtree methods: O(*k* + *k* log (*n*/*k*))
      - – Acceleration: pruning; sensitive to noisy data.
    - Span-space methods:
      - – NOISE: O(sqrt(*n*) + *k*)
      - – ISSUE: O(log (*n*/*L*) + sqrt(*n*)/*L* + *k*)
        - » *L* is a tunable parameter
      - – Interval Tree: O(log *n* + *k*)

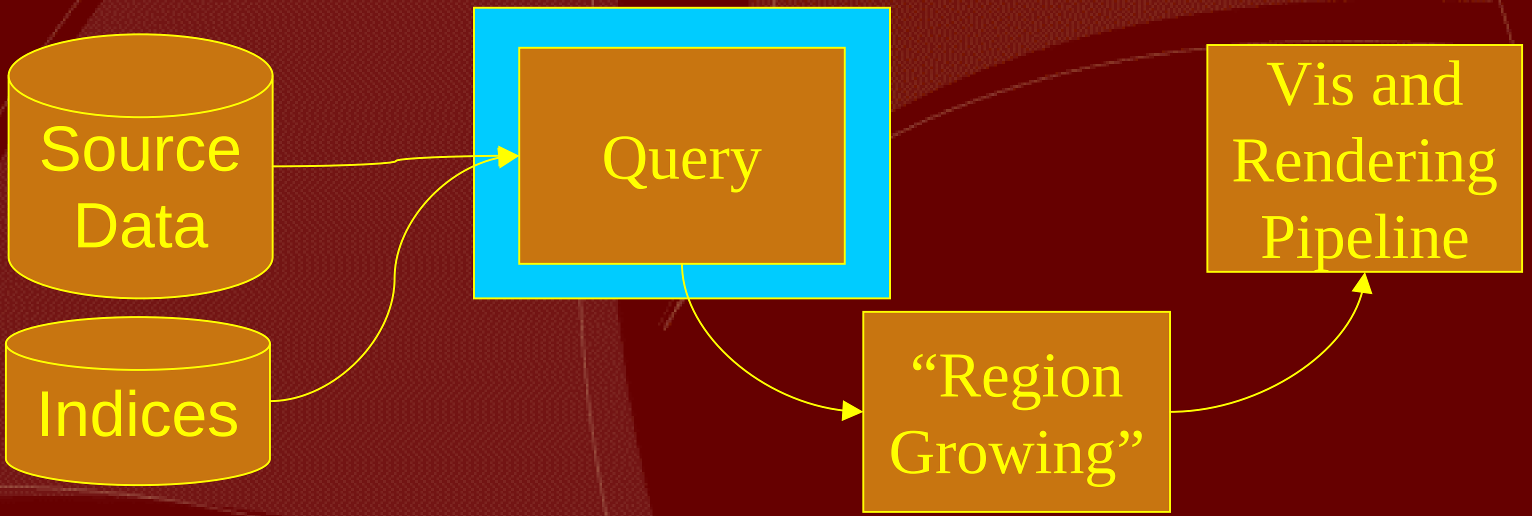
Finding Data Quickly

These approaches work well for isocontouring, but users want more than isosurfaces.:

- These queries are for a single variable.
  - Want multi-valued queries. Current simulations produce 10s-100s of variables per cell.
- These queries only find cells that contain the isovalue.
  - May want interior cells for quantitative analysis.
- What about combinatorial tree-based methods?
  - Curse of dimensionality: adding more dimensions results in an exponential growth in storage and processing complexity.
- Want to have general purpose implementation to feed data to multitude of processing pipelines, not just isosurfacing.

Overview of Our Implementation and Results

- Bitmap indices: the indexing structure and query engine.
  - See http://sdm.lbl.gov/fastbit
  - State-of-the-art from the scientific data management community.
- Preprocessing query output.
- Provide to visualization engine.
- Experimental performance results.



What is a Bitmap Index?

Data values	b_0	b_1	b_2	b_3	b_4	b_5
0	*1*	*0*	*0*	*0*	*0*	*0*
1	*0*	*1*	*0*	*0*	*0*	*0*
5	*0*	*0*	*0*	*0*	*0*	*1*
3	*0*	*0*	*0*	*1*	*0*	*0*
1	*0*	*1*	*0*	*0*	*0*	*0*
2	*0*	*0*	*1*	*0*	*0*	*0*
0	*1*	*0*	*0*	*0*	*0*	*0*
4	*0*	*0*	*0*	*0*	*1*	*0*
1	*0*	*1*	*0*	*0*	*0*	*0*

- Compact: one bit per distinct value per object.
- Easy and fast to build: O(*n*) vs. O(*n* log *n*) for trees.
- Efficient to query: use bitwise logical operations.
  - (0.0 < H_2O < 0.1) AND (1000 < temp < 2000)
- Efficient for multidimensional queries.
  - No “curse of dimensionality”
- What about floating-point data?
  - Binning strategies.

Bitmap Index Query Complexity and Space Requirements

- How Fast are Queries Answered?
  - Let N denote the number of objects and H denote the number of hits of a condition.
  - Using uncompressed bitmap indices, search time is O(N)
  - With a good compression scheme, the search time is O(H) – the theoretical optimum.

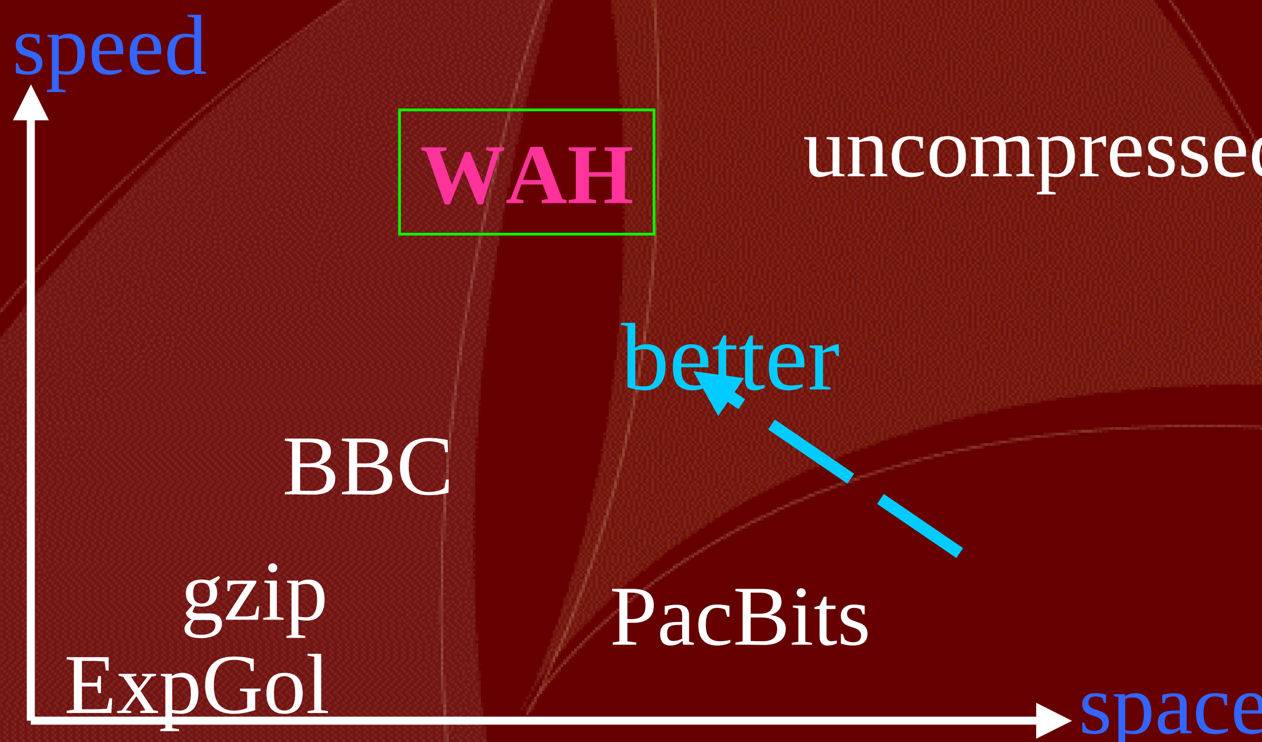
- How Big are the Indices?
  - In the worst case (completely random data), the bitmap index requires about 2x in data size.
  - On the average, we've seen a cost of 1/10th the size of the original data.

Index Sizes for our Performance Study

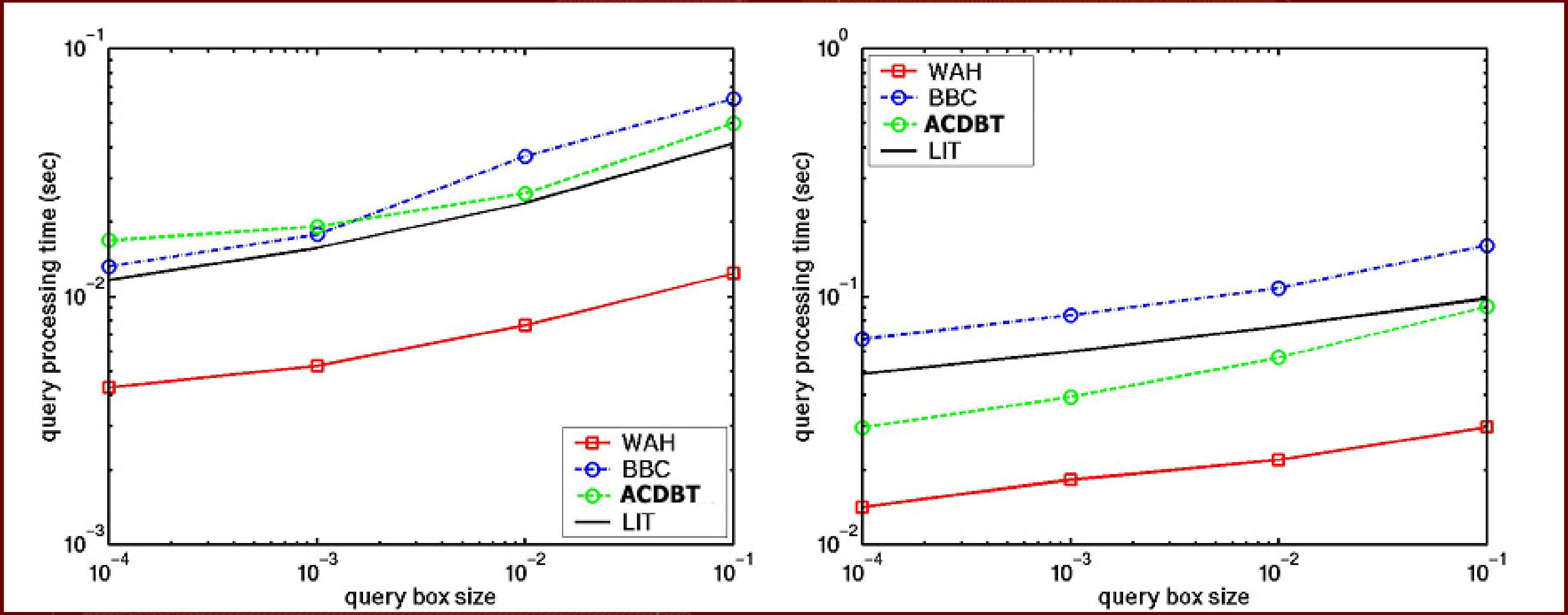
- Original data: 383^3 grid of 4-byte floats: ~215MB

Variable	Index Size (MB)	Size Factor	Time (sec)
Pressure	77.59	0.36	7.47
Density	128.70	0.60	8.56
Temperature	124.93	0.58	8.76
Velocityx	247.49	1.15	13.30
H2O	263.64	1.23	13.04
CH4	314.88	1.46	13.49

Bitmap Index Compression

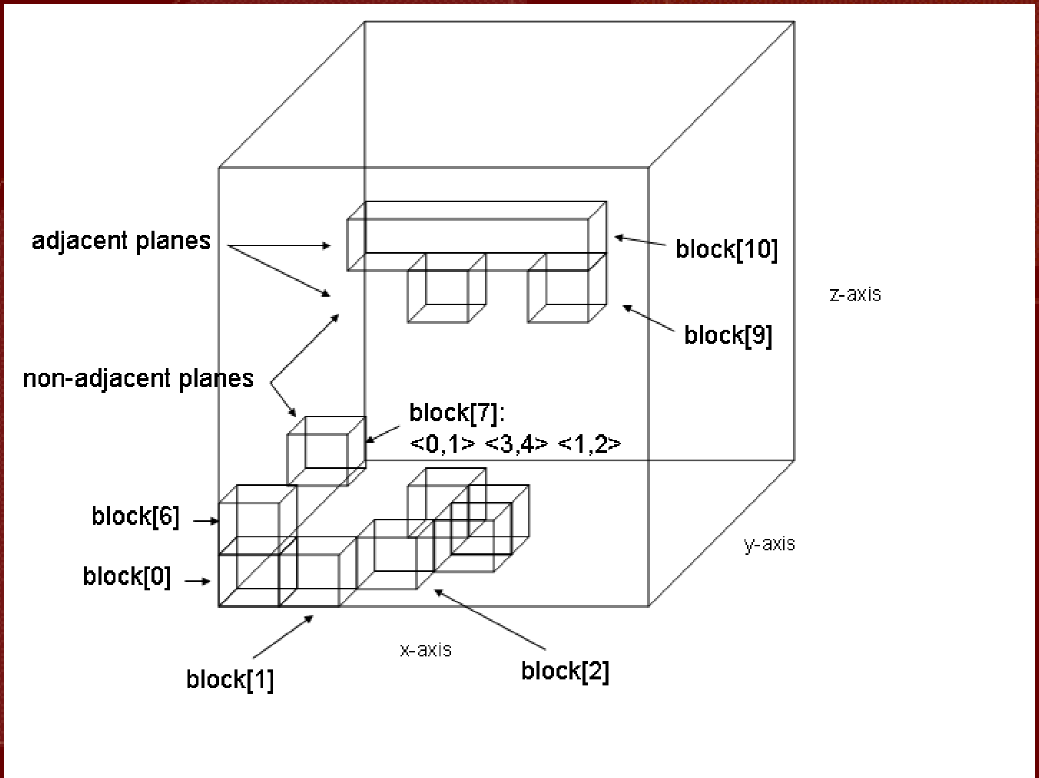


Bitmap Index Query Performance

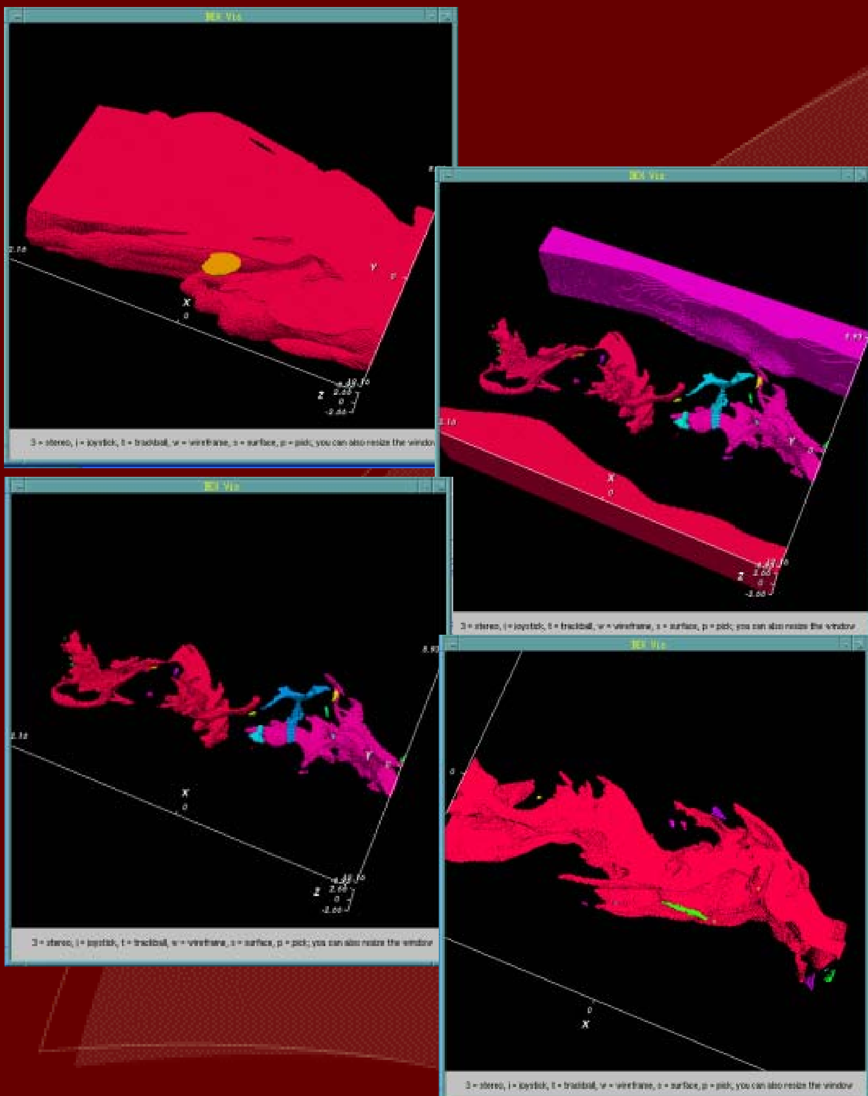


Consolidating Query Results: Region Growing

- Find and label cells that share an edge, face or vertex.
- Not strictly necessary for “meat grinder” visualization.
- Imperative for meaningful analysis operations.



Visual Results



- $CH_4 > 0.3$
- $Temp < T_1$
- $CH_4 > 0.3$ AND $temp < T_1$
- $CH_4 > 0.3$ AND $temp < T_2$
  - $T_1 < T_2$

Performance Analysis Experiment

- The performance experiment:
  - Compare speed of answering queries: fastbit vs. an “industry standard implementation” of span-space isosurfacing.

- Experimental methodology.
  - Isosurface: find cells, construct geometry.
  - DEX: find cells, construct geometry.
  - For each implementation:
    - Load dataset, disregard time required for one-time initialization.
    - For several different isovalues, measure time required to find cells and generate geometry.

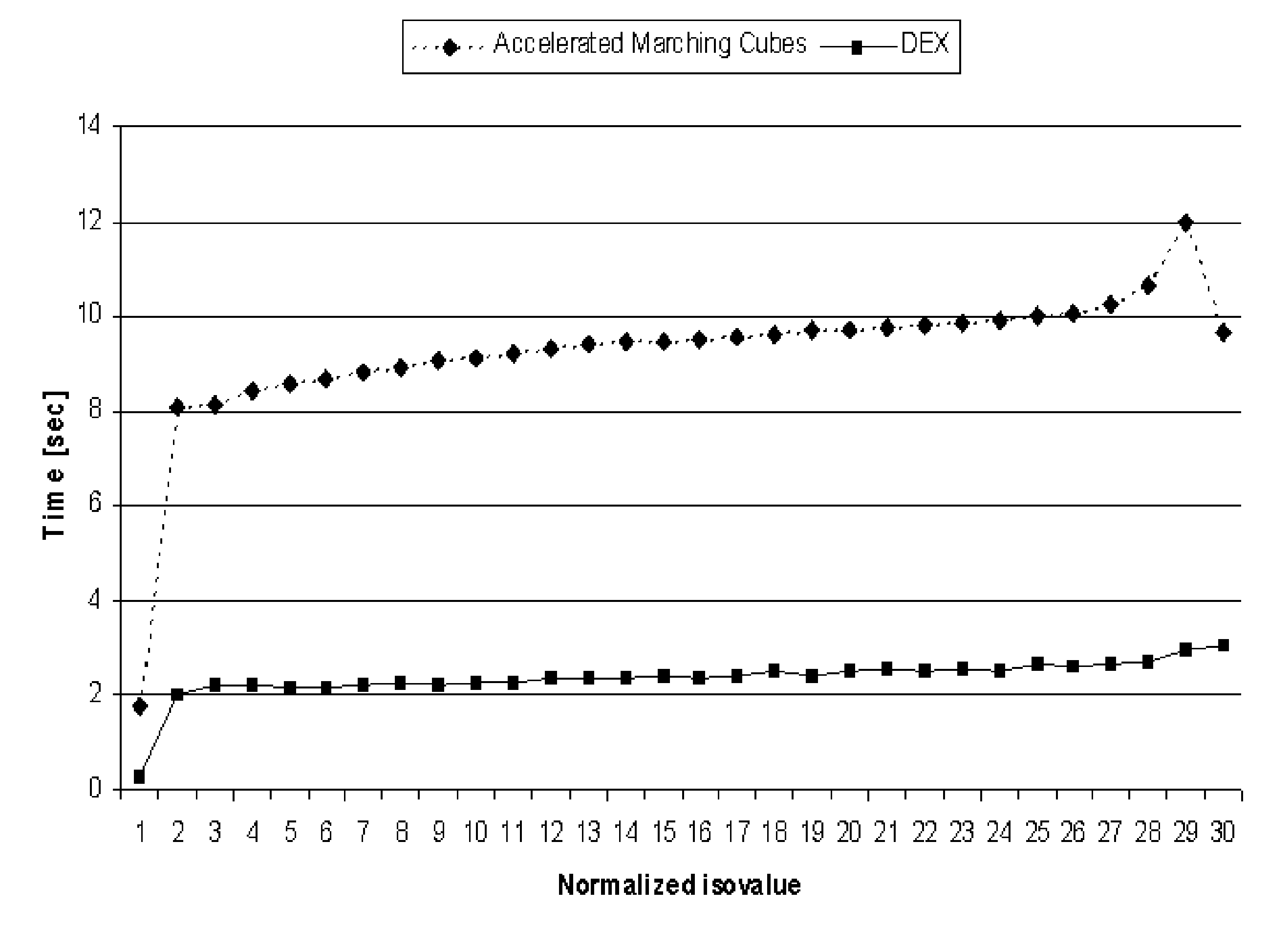
Experimental Methodology

- Ideally, we want to measure and compare only the time required for finding cells (exclude geometry construction).
  - Not possible due to implementation details.
- Second best: want to measure and separately report time required for search and geometry construction.
  - Again, not possible due to implementation details.
- Is including geometry construction time valid?
  - Yes. See Sutton et. al., A Case Study of Isosurface Extraction Algorithm Performance *2nd Joint Eurographics-IEEE TCCG Symposium on Visualization*, May 2000.

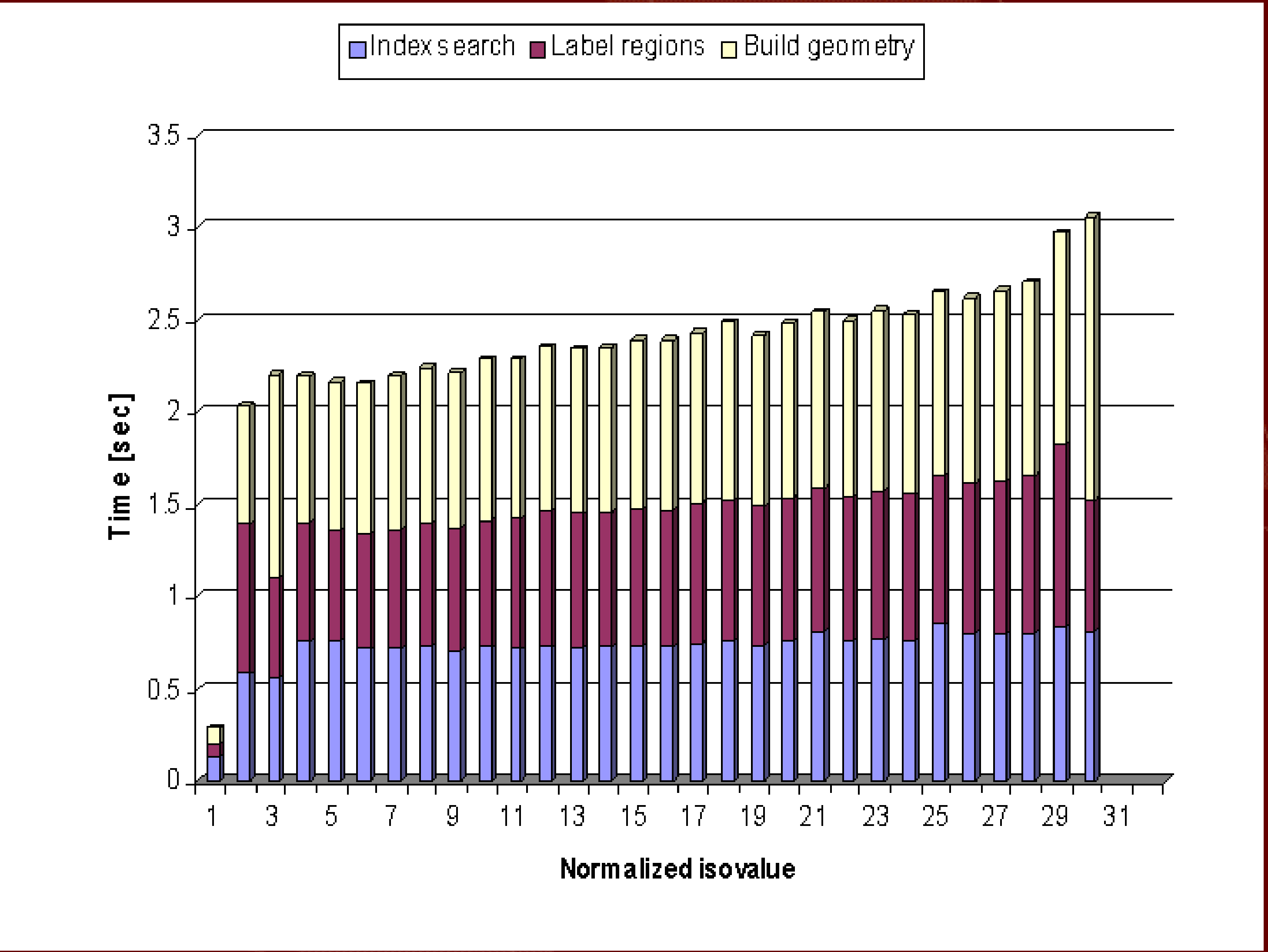
Experimental Methodology, ctd.

- How does geometry construction phase differ between isocontouring and DEX?
  - Isosurfacing:
    - Each cell containing the surface generates between 1-*n* triangles, where *n* varies between 4-10 depending upon the implementation.
    - Experimental results show an average of about 2.5 triangles/cell.
    - Some math required to produce triangles.
  - DEX:
    - Each cell satisfying search criteria is visually represented as a cube composed 12 triangles.
    - No math required to produce triangles.
    - In our experiments, DEX is returning the *interior* cells as well.
    - We include time for region growing in our overall time.
- Net result:
  - DEX is doing a lot more work in the performance study.

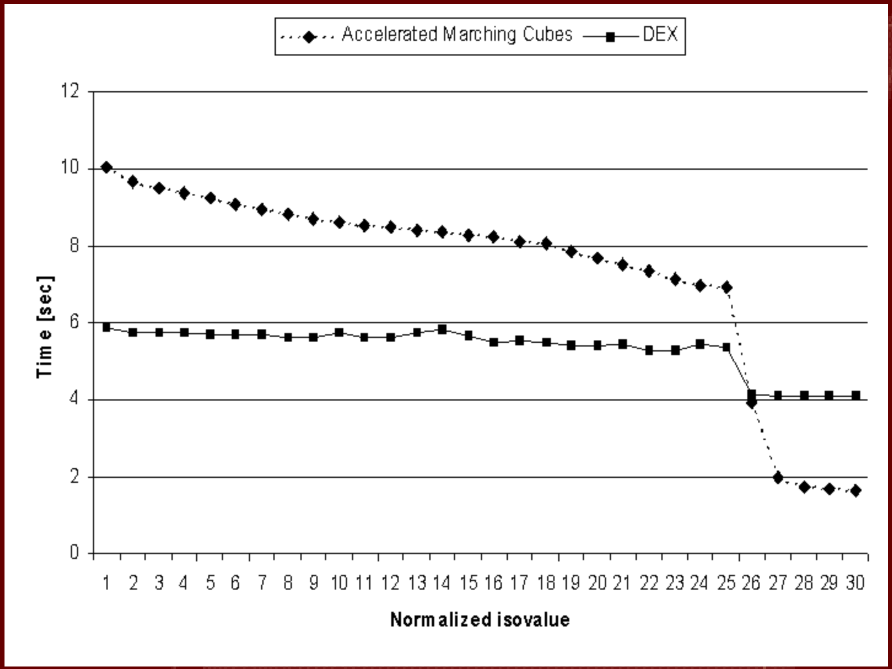
Results: Density

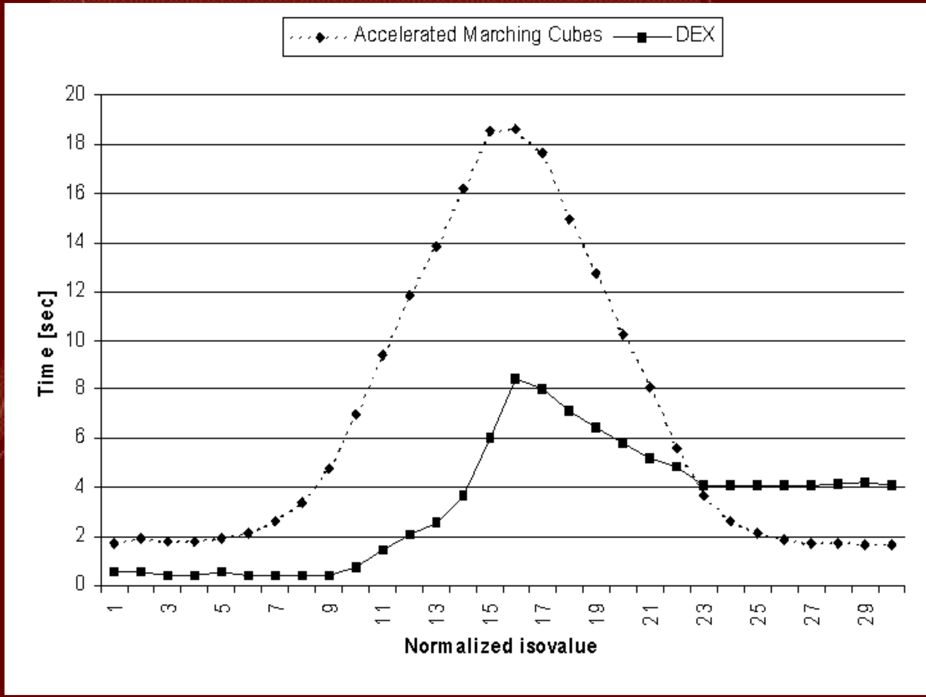


Results: Density



Results: Other Variables





Future Directions

- Include in mainstream visualization tools.
  - Existing use in ROOT package from CERN.
  - AVS/Express module under development.
- Parallel implementation.
  - SC05 HPC Analytics Challenge – Network Connection Data Analysis.
    - Parallel queries reduce search time from ~2200 seconds using existing tools (grep) to ~22 seconds using FastBit.
- Demonstrate and deploy integrated query-analysis-visualization.
- Better visualization of query results.
- Help users pose queries, iterative queries over derived variables.
- Multiresolution queries, topology-preserving multires queries (AMR).
- Constraints relaxation based upon proximity (space, data values, time).

Conclusion

- DEX faster than industry standard implementation by 137% to 392%.
  - DEX doing more work: more triangles/cell, more cells per query, and a region growing step to label connected cells.
- DEX architecture amenable to use in a general way for visualization, analysis, …
- This approach offers new traction on the task of helping meet the needs of the scientific research community.
  - Focus vis processing and human interpretation on relevant data.
  - Fast: multidimensional queries suitable for use with multi TB data.

Acknowledgement

- This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

The End

The End