Efficient Indexing Technology for Data Mining of Scientific Data

Arie Shoshani

(and colleagues)

Lawrence Berkeley National Laboratory

ICDM Conference

Nov 28-30, 2005

Scientific Data Mining

- **What's special about it**
  - **Observing spatial behavior over time**
  - **Identifying known patterns**
  - **Selecting subsets from billions of objects based on attribute properties**
  - **Looking for rare objects based on attribute properties**
  - **Dynamic data exploration – real time response**
  - **Visualization / summary statistics**
- **Implication**
  - **Need very efficient indexing over multidimensional space of attributes**
  - **Bitmap indexing is particularly suitable**
  - **A compute-efficient method - FastBit**

Observing spatial behavior over time

Combustion: Flame Front Tracking

Need to perform:

- **Cell identification**

  Identify all cells that satisfy user specified conditions, such as, "$600 < \text{Temperature} < 700$ AND HO_2 concentration $> 10^{-7}$"

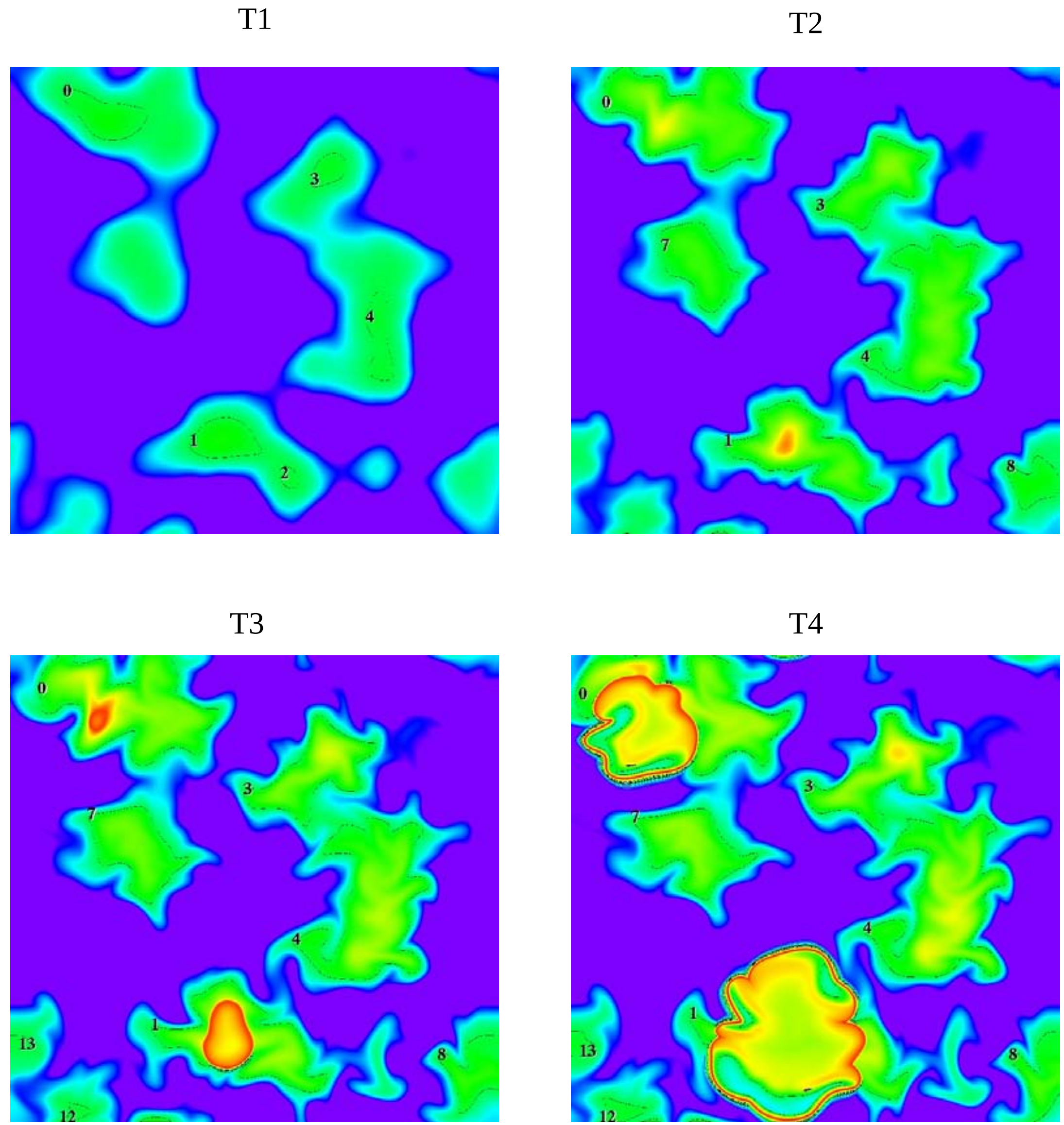
- **Region growing**

  Connect neighboring cells into regions

- **Region tracking**

  Track the evolution of the regions (i.e., features) through time

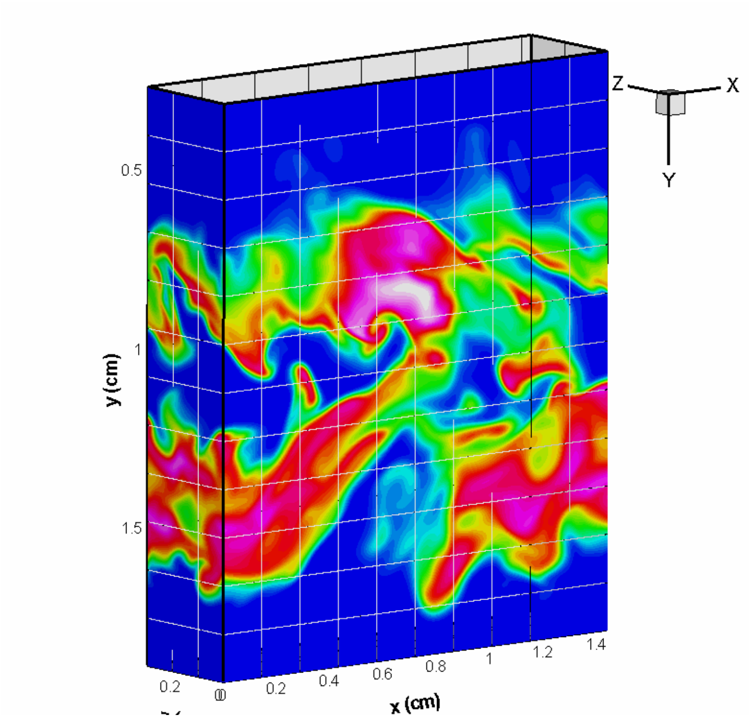
All steps perform with Bitmap structures



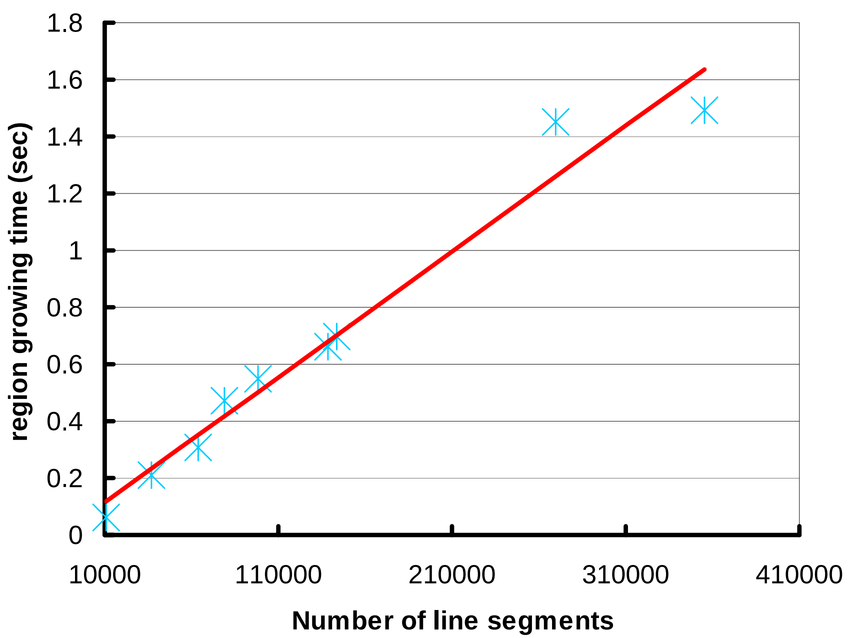
Indexing Requirements

- **Search over large spatio-temporal data**
  - **Combustion simulation: 1000x1000x1000 mesh with 100s of chemical species over 1000s of time steps**
  - **Supernova simulation: 1000x1000x1000 mesh with 10s of variables per cell over 1000s of time steps**
- **Common searches are partial range queries**
  - **Temperature > 1000 AND (pressure > 10^6 OR ($10^{-6} < HO_2 < 10^{-7}$))**
- **Features**
  - **Search time proportional to number of hits**
  - **Index generation linear with data values (require read-once only)**

FastBit-Based Multi-Attribute Region Finding is Theoretically Optimal



Flame Front discovery
(range conditions for multiple measures)
in a combustion simulation (Sandia)



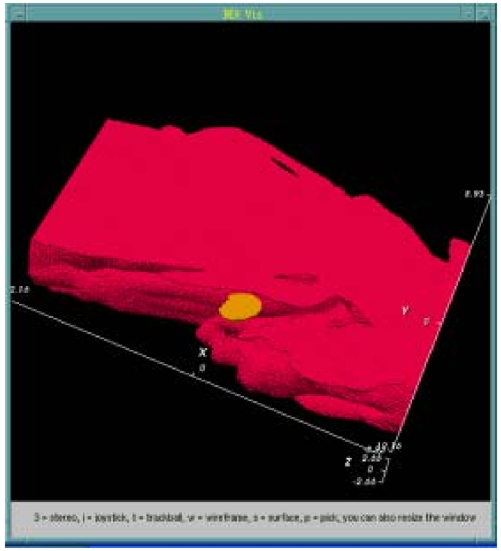
Time required to identify regions

On 3D data with over 110 million points,

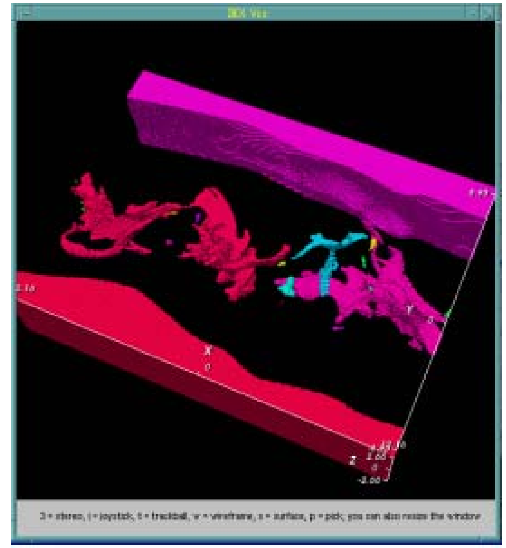
region finding takes less than 2 seconds

Multi-Variable Visualization of Combustion Data Set

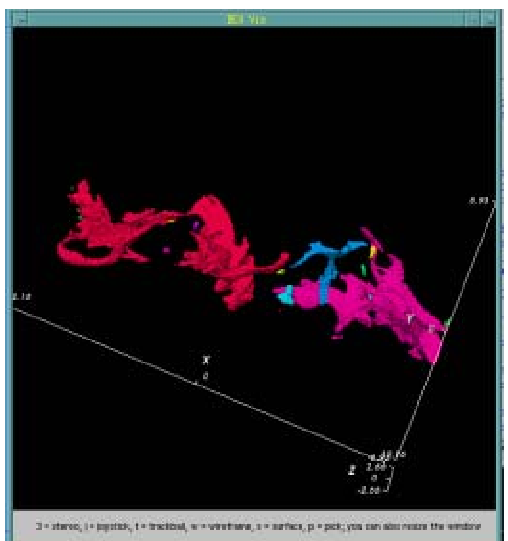
a) Query: CH4 > 0.3



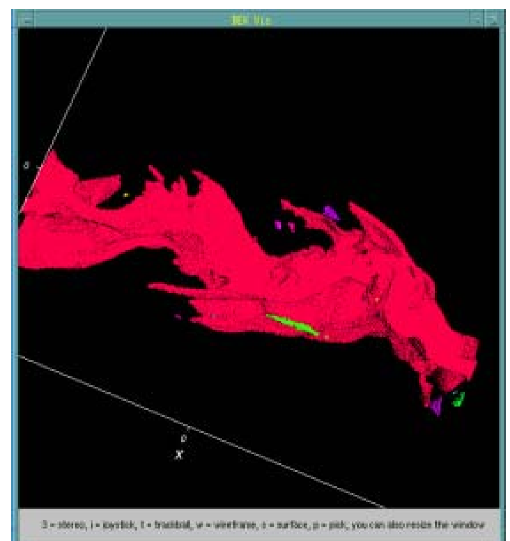
b) Q: temp < 3



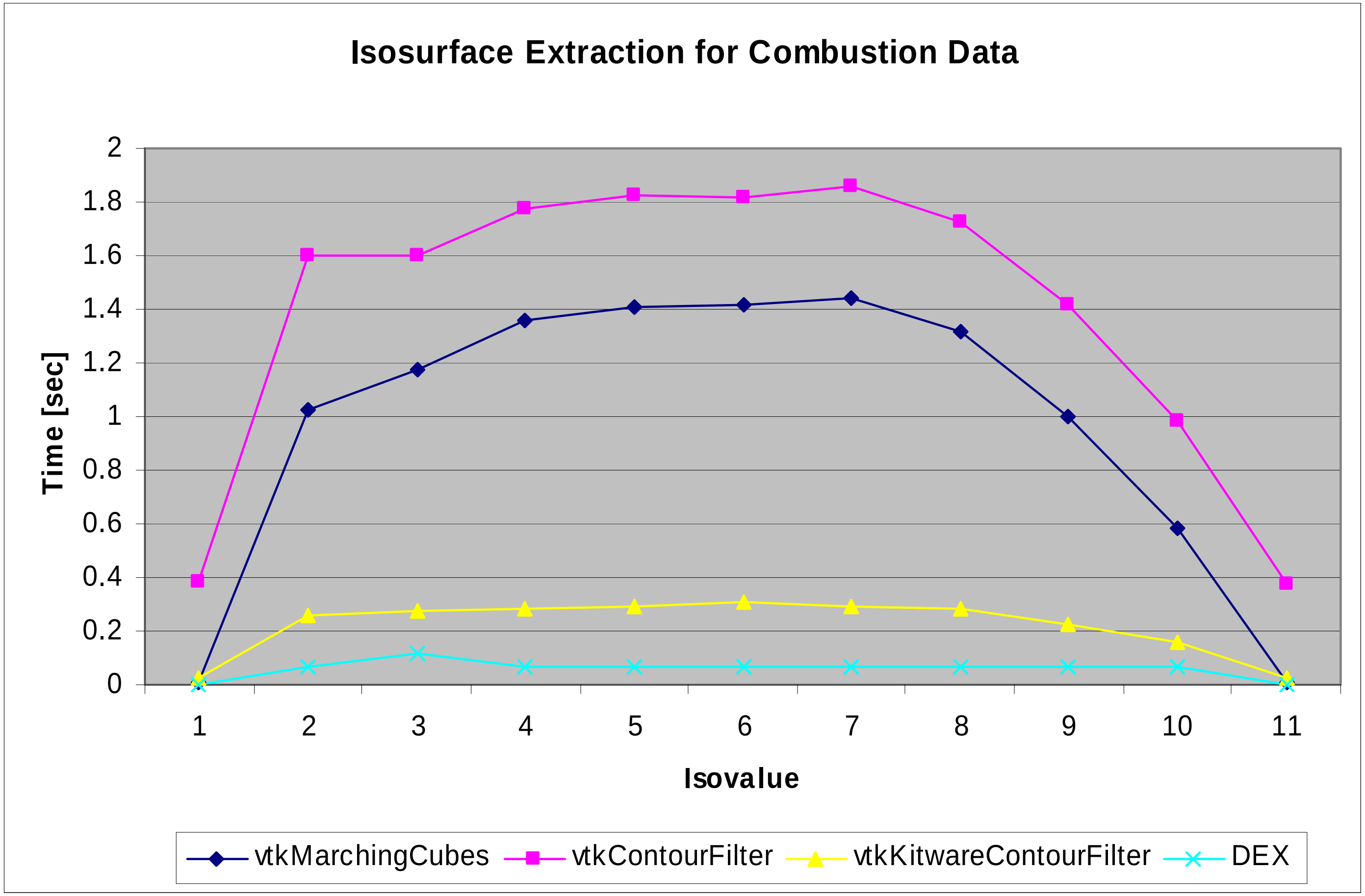
c) Q: CH4 > 0.3 AND temp < 3



d) Q: CH4 > 0.3 AND temp < 4



Performance Results of FastBit-VTK



VTK rendering time: 0.2 – 2 seconds.

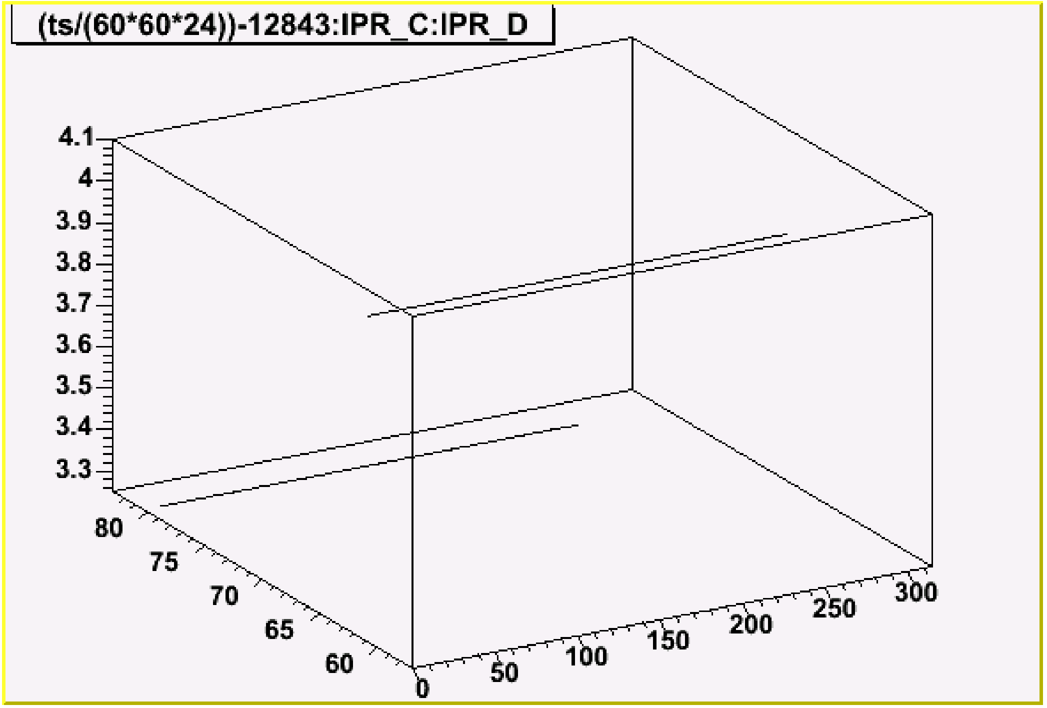
DEX is on average a factor of three to four faster than the best isosurface algorithm of VTK.

Identifying Known Patterns

Network Traffic Flow Analysis

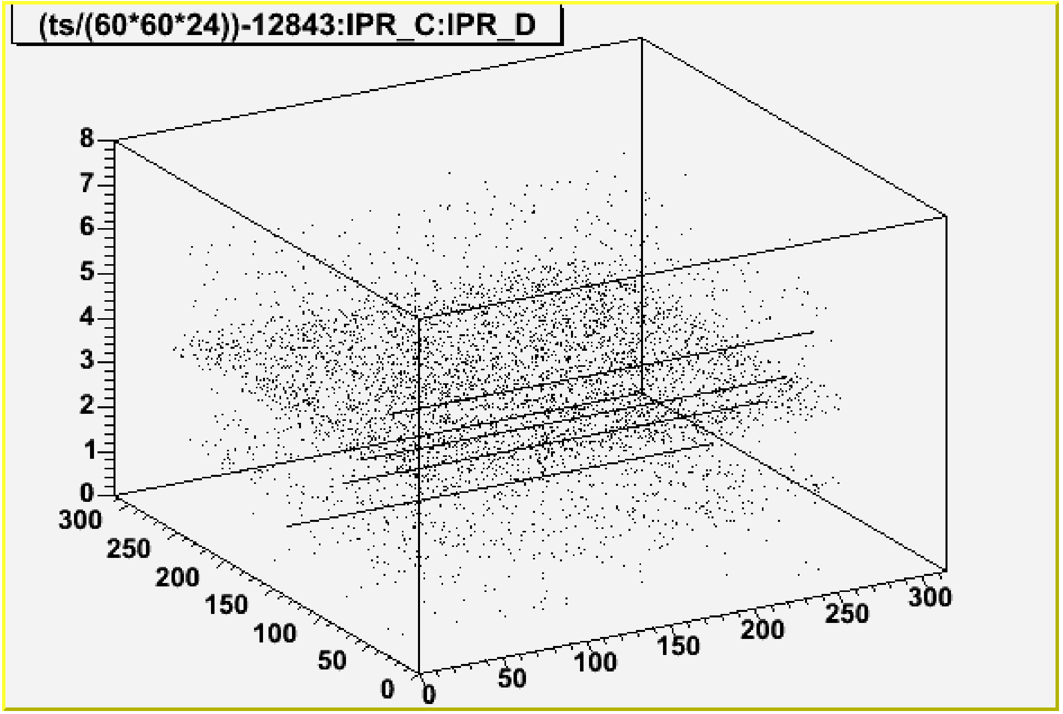
- Each record is a complete network communication session
  - Source IP, Destination IP, Start time, Duration, Protocol, Data volume, State, Flag, Transfer Rate, …
- Goals
  - Parallel visual data analysis framework
  - High-speed forensics
  - Large scale profiling
- Intrusion Detection System (BRO) log shows
  - Jul 28 17:19:56 AddressScan 221.207.14.164 has scanned 19 hosts
  - Jul 28 19:19:56 AddressScan 221.207.14.88 has scanned 19 hosts
  - FastBit integrated with data analysis environment (for visualization)
- Using FastBit/Vis can be used to explore what else might be going on

Scans from the Two Hosts



- Query: select ts/(60*60*24)-12843, IPR_C, IPR_D where IPS_A=221 and IPS_B=207 and IPS_C=14 and IPS_D in (88, 164)
- Picture: scatter plot (dots) of the three selected variables
- Two lines indicating two sets of scans
- Note: a lot more than 19 hosts scanned

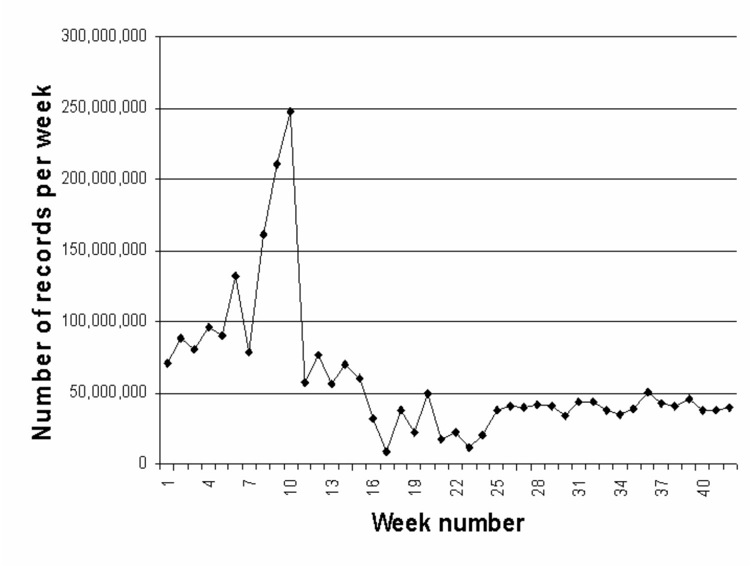
Are There More Scans?



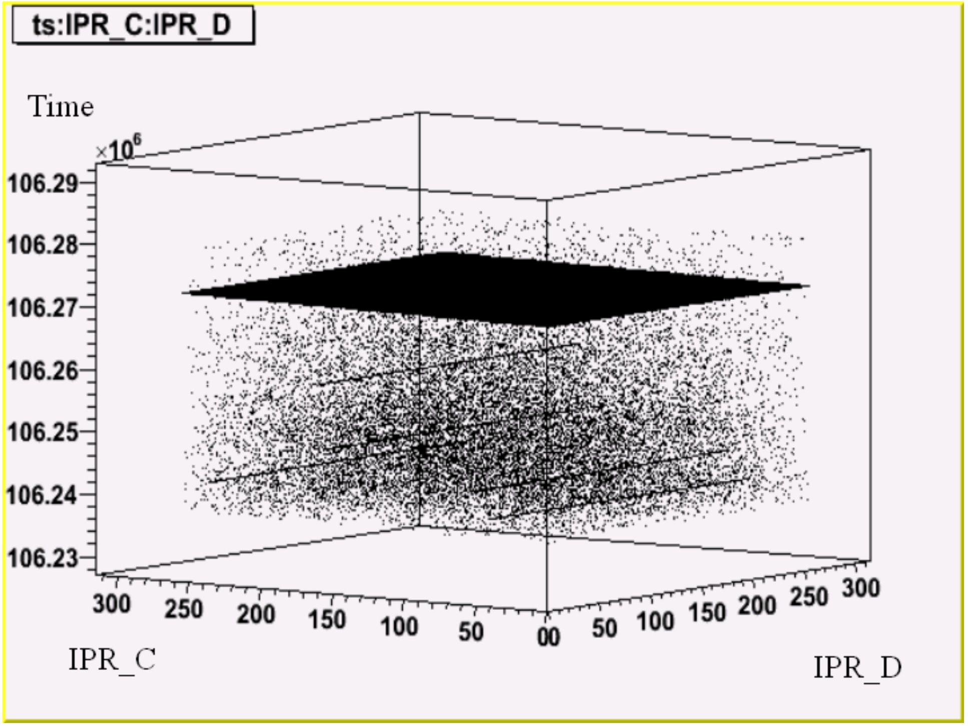
- **Query: select ts/(60*60*24)-12843, IPR_C, IPR_D where IPS_A=221 and IPS_B=207**
- **More scans from the same subnet**

Selecting subsets from billions of objects based on attribute properties

Need: increase number of scans to be searched

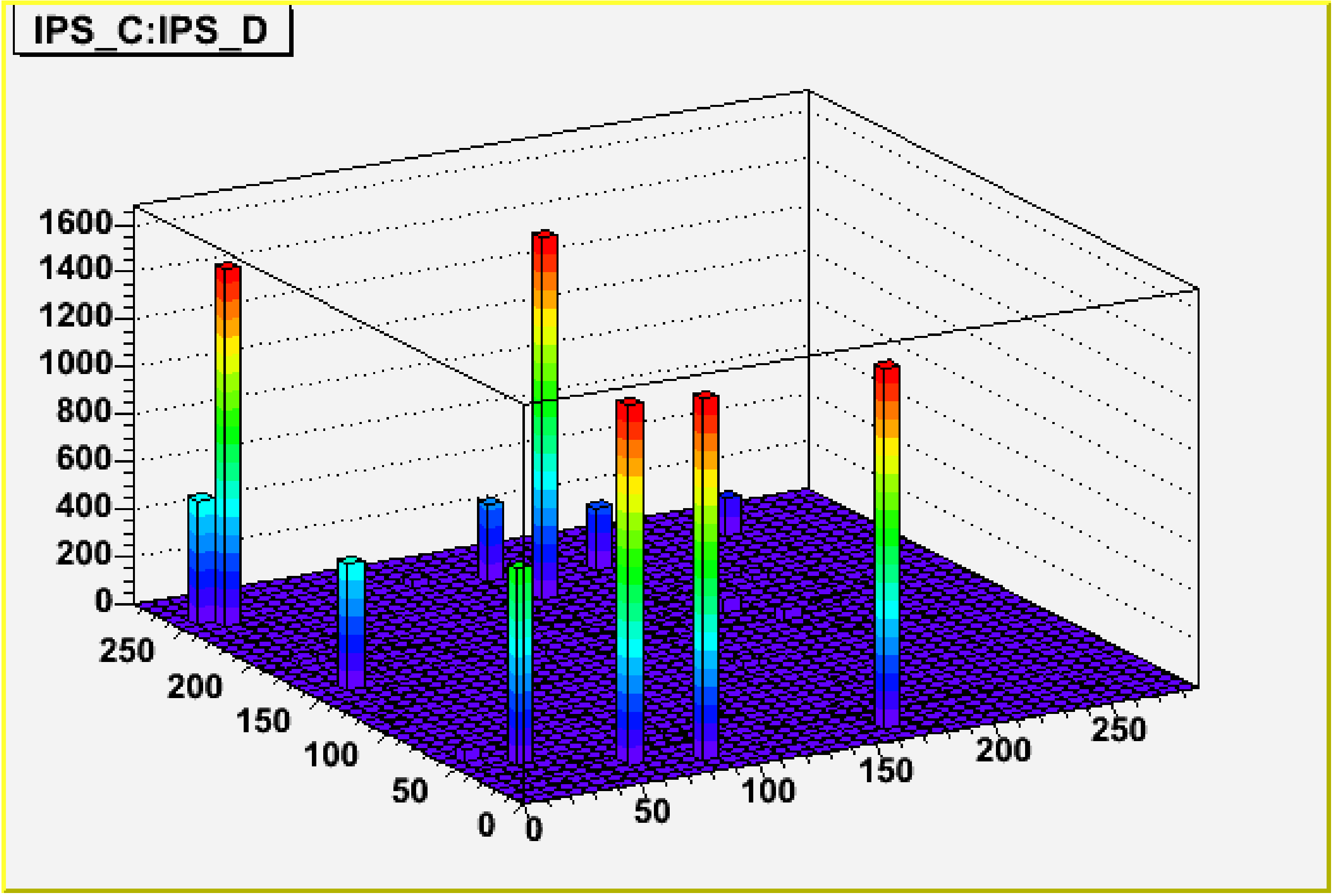


Search over 2.5 billion objects (42 weeks)



Can this be done in real time?

Who Is Doing It? - real time response

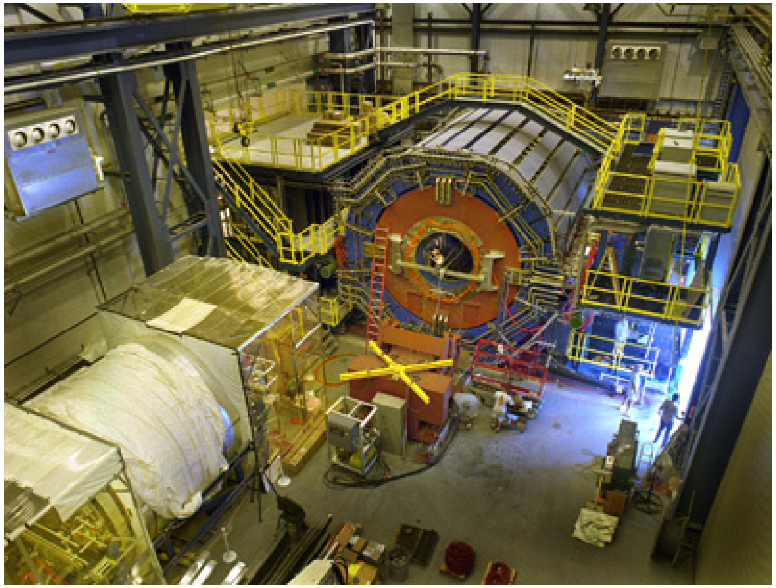


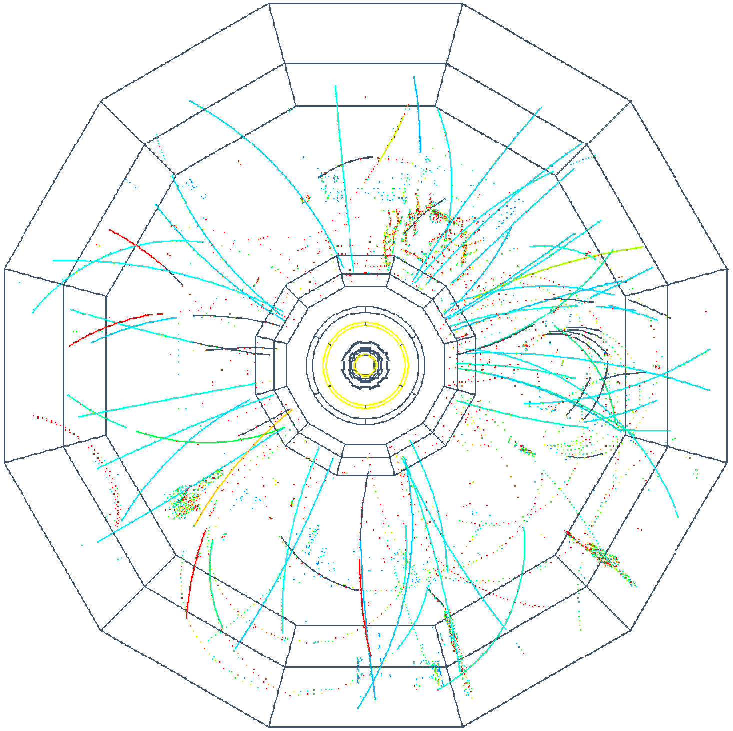
- **Query: select IPS_C, IPS_D where IPS_A==221 and IPS_B==207**
- **Generate: the histogram of the IPS_C and IPS_D**
- **Five IP addresses started most of the scans!**

Looking for rare objects based on attribute properties

The STAR Experiment at Brookhaven National Laboratory

- **STAR: Solenoidal Tracker At RHIC; RHIC: Relativistic Heavy Ion Collider**
- **600 participants / 50 institutions / 12 countries / in production since 2000**
- **~100 million collision events a year, ~5 MB raw data per event, several levels of summary data**
- **Generated 3 petabytes and 5 million files**





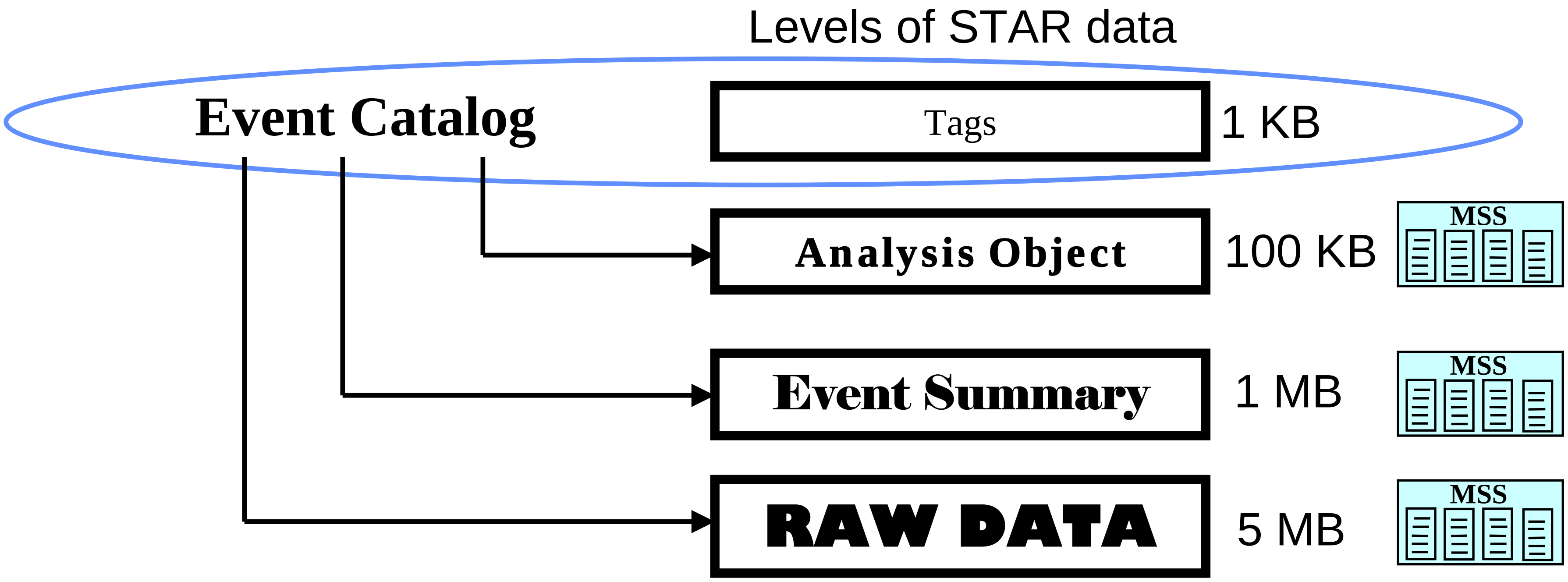
Read-only data, write-once read-many (WORM) data

Typical Scientific Exploration Process

- **Generate large amounts of raw data**
  - **large simulations**
  - **collect from experiments**
- **Post-processing of data**
  - **analyze data (find particles produced, tracks)**
  - **generate summary data**
    - **e.g. momentum, no. of pions, transverse energy**
    - **Number of properties is large (50-100)**
- **Analyze data**
  - **use summary data as guide**
  - **extract subsets from the large dataset**
    - **Need to access events based on partial properties specification (range queries)**
    - **e.g. (0.1 < AVpT < 0.2) ^ (numberOfPrimaryTracks > 1000)**
  - **apply analysis code**

STAR Event Catalog with FastBit

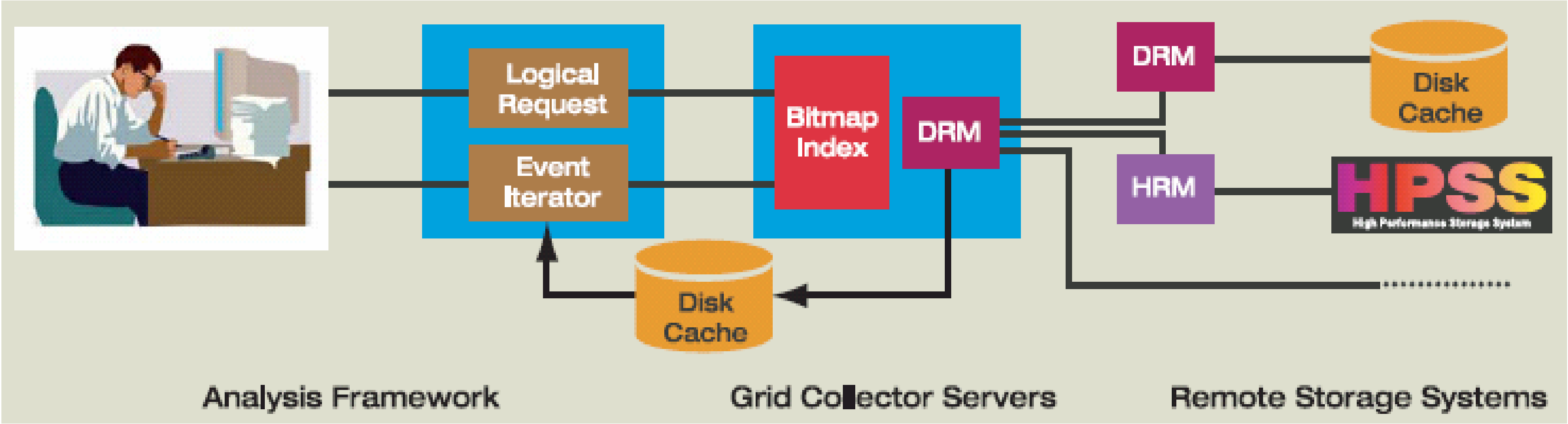
- STAR data is organized into several levels
- The Event Catalog indexes all tags but only maintains references to other levels



An Example of Using the Grid Collector: Searching Problem in STAR

- One of the primary goals of STAR is to search for Quark Gluon Plasma (QGP)
- A small number (~hundreds) of collision events may contain the clearest evidence of QGP
- Using high-level summary data, researchers found 80 special events
    - Have track distributions that may indicate presence of QGP
- Further analysis needs to access more detailed data
    - Detailed data are large (terabytes) and reside on HPSS (MSS)
    - May take many weeks to manually migrate to disk

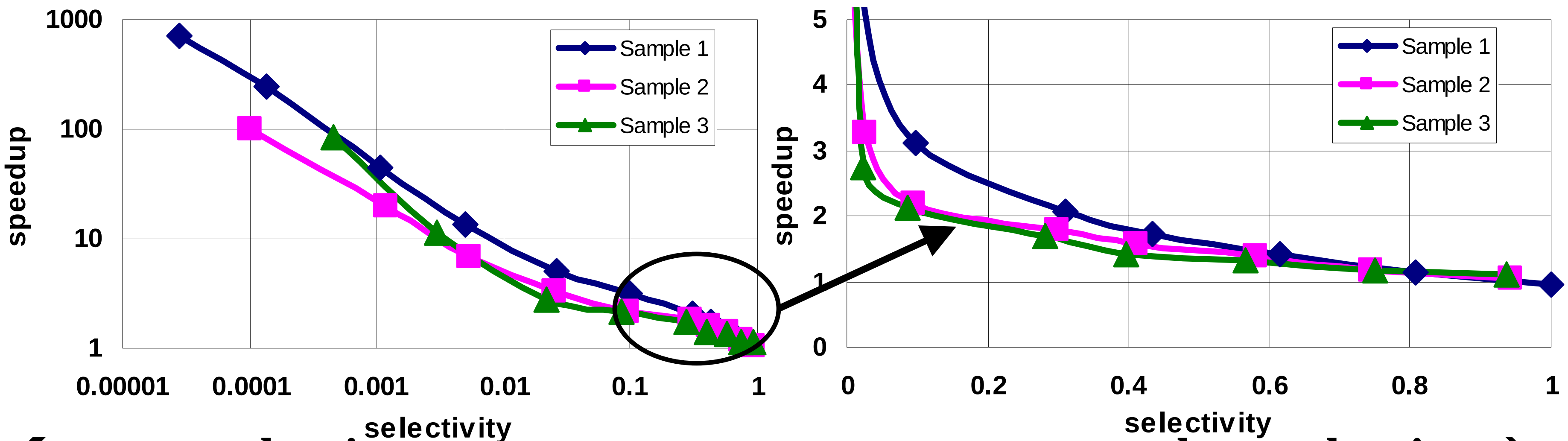
Grid Collector Features



Key features of the Grid Collector:

- Providing transparent object access
- Selecting objects based on their attribute values
- Improving analysis system's throughput
- Enabling interactive distributed data analysis

Grid Collector Speeds up Analyses



← **more selective** **less selective** →

- **Legend**
  - **Selectivity: fraction of events needed by the analysis**
  - **Speedup = ratio of time to read events without GC and with GC**
  - **Speedup = 1: speed of the existing system (without GC)**
- **Results**
  - **When searching for rare events, say, selecting one event out of 1000 (selectivity = 0.001), using GC is 20 to 50 times faster**
  - **Even using GC to read 1/2 of events, speedup > 1.5**

Grid Collector Facilitates Difficult Analyses

- Searching for anti-^{3}He
- Lee Barnby, Birmingham, UK
- Previous studies identified collision events that possibly contain anti-^{3}He, need further analysis

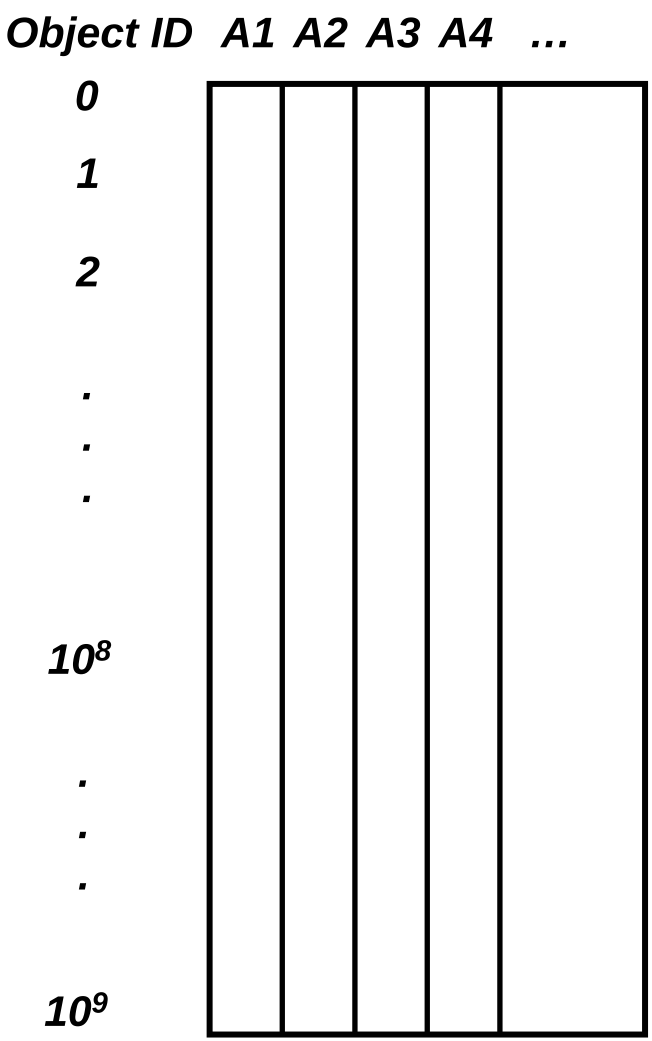
- Searching for strangelet
- Aihong Tang, BNL
- Previous studies identified events that behave close to strangelets, need further investigation

- Without Grid Collector, one has to retrieve many files from mass storage systems and scan them for the wanted events – may take weeks or months, no one wants to actually do it
- With Grid Collector, both jobs completed within a day

FastBit:
Compute-Efficient Bitmap Indexing

Efficient Indexing

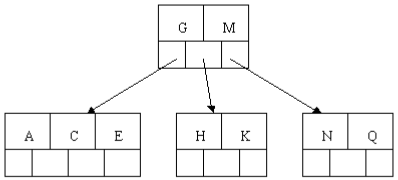
- Efficient search is necessary to facilitate real-time data mining over a very large number of objects
  - Millions-Billions of objects
  - Each having multiple (hundreds) attributes
  - Attributes may be categorical or numeric
  - {A1, A2, …, An} form a multidimensional space where objects are points in that space



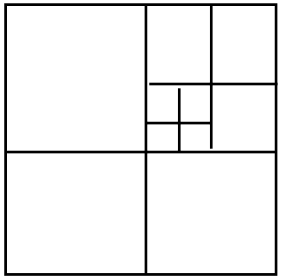
Why use bitmap indexing?

- <u>Goal</u>: efficient search of *multi-dimensional* data
- Indexes for data that needs to be updated
  - e.g. family of B-Trees
  - Sacrifice search efficiency to permit dynamic update
- Space-partitioning multi-dimensional indexes
  - e.g. R-tree, Quad-trees, KD-trees, …
  - Don't scale for large number of dimensions
  - Are inefficient for partial searches (subset of attributes)
- Bitmap indexes are good for:
  - Non-updatable data
  - Partial range queries
  - Multi-attribute queries (multi-dimensional)
  - Use compression methods
- Bitmap indexes can use compression methods
  - To reduce size of index
  - Logical operation on compressed data

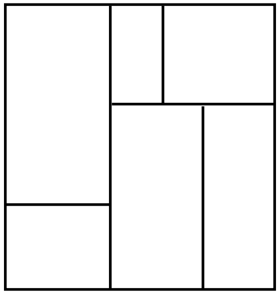
B-tree



QUAD-tree

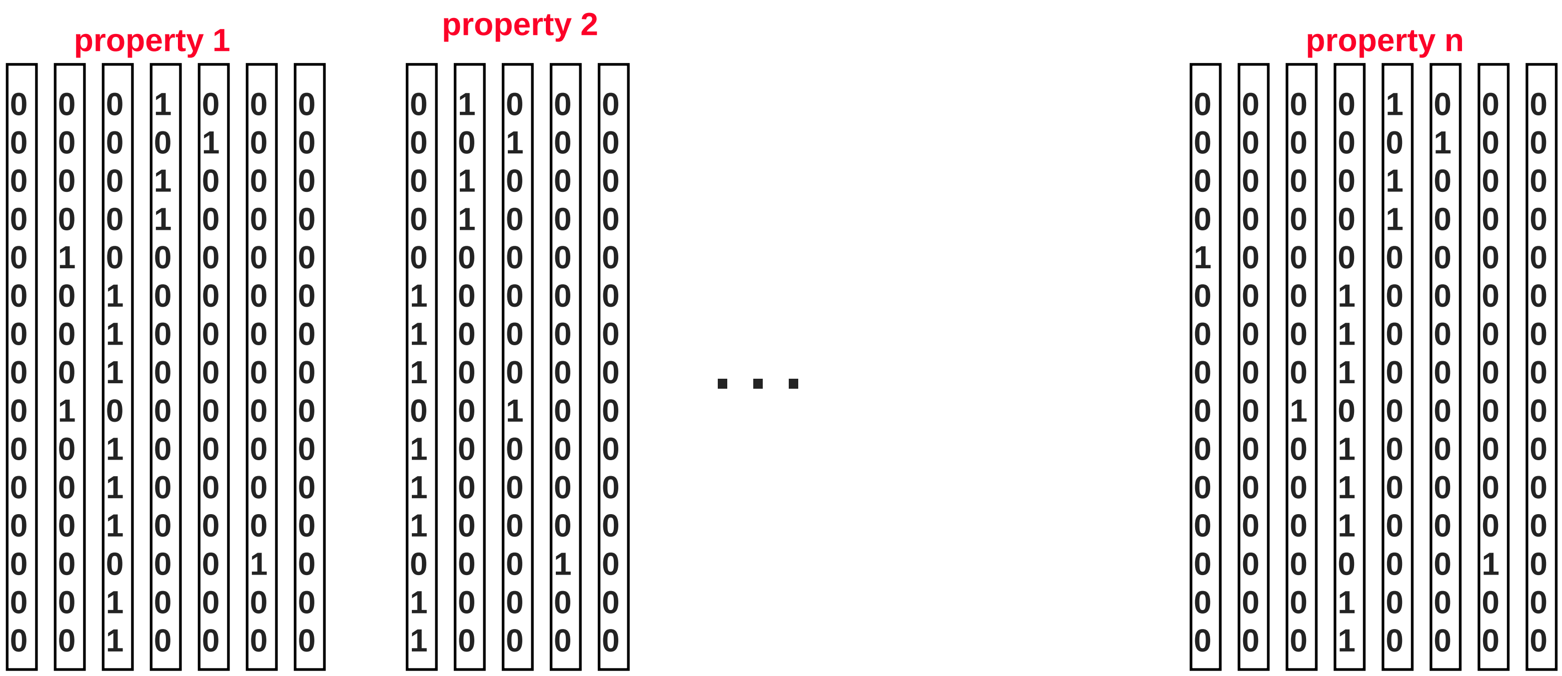


KD-tree



Bit-Sliced Index

- Take advantage that index need to be is append only
- partition each property into bins
  - (e.g. for 0<Np<300, have 300 equal size bins)
- for each bin generate a bit vector
- compress each bit vector (some version of run length encoding)



Basic 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
	=0	=1	=2	=3	=4	=5

A < 2   2 < A < 5

- **First commercial version**
  - **Model 204, P. O'Neil, 1987**
- **<u>Easy to build</u>: faster than building B-trees**
- **<u>Efficient to query</u>: only bitwise logical operations**
  - **$A < 2 \rightarrow b_0$ OR b_1**
  - **$2<A<5 \rightarrow b_3$ OR b_4**
- **Efficient for multi-dimensional queries**
  - **Use bitwise operations to combine the partial results**
- **Size: one bit per distinct value per object**
  - **Definition: Cardinality == number of distinct values**
  - **Compact for low cardinality attributes only, say, < 100**
  - **Need to control size for high cardinality attributes**

Run Length Encoding

Uncompressed:
0000000000001111000000000 ......0000001000000001111100000000 000000

Compressed:
12, 4, 1000,1,8, 5,492

Practical considerations:
- **Store very short sequences as-is (literal words)**
- **Count bytes/words rather than bits (for long sequences)**
- **Use first bit for type of word: literal or count**
- **Use second bit of count to indicate 0 or 1 sequence**

[literal]	[31 0-words]	[literal]	[31 0-words]
[00 0F 00 00]	[80 00 00 1F]	[02 01 F0 00]	[80 00 00 0F]

Other ideas
- **repeated byte patterns, with counts**
- **Well-known method use in Oracle: Byte-aligned Bitmap Code (BBC)**

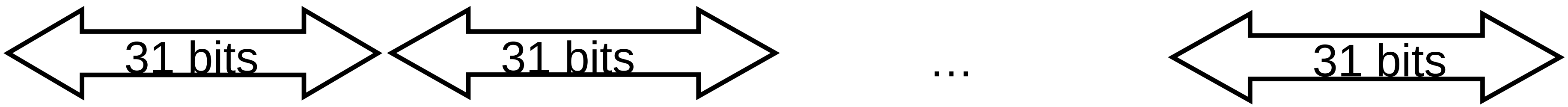
Advantage:
Can perform logical operations such as: AND, OR, NOT, XOR, …
And COUNT operations directly on compressed data

The Special Compression Method in FastBit Is Compute-Efficient

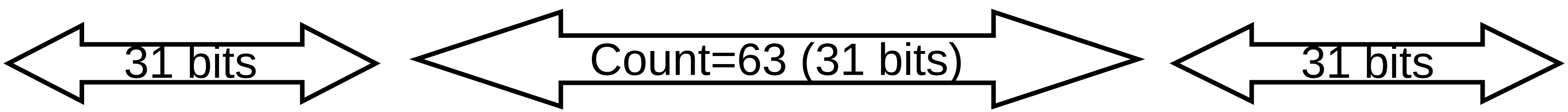
Example: 2015 bits

100000000000000000000111000000000000000000000000000000……………….00000000000000000000000000000001111111111111111111111111

<u>Main Idea</u>: Use run-length-encoding, but..
group bits into 31-bit groups



Merge neighboring 0 or 1 groups



Encode each group using one word

- Name: Word-Aligned Hybrid (WAH) code
- Key features: WAH is compute-efficient because it
  - Uses the run-length encoding (simple)
  - Allows operations directly on compressed bitmaps
  - Never breaks any words into smaller pieces during operations

Bitmap Indices Encoding

a) list of attributes **b) equality encoding** **c) range encoding**

	$\pi_A(R)$	E^9	E^8	E^7	E^6	E^5	E^4	E^3	E^2	E^1	E^0	R^8	R^7	R^6	R^5	R^4	R^3	R^2	R^1	R^0
1	3	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0
2	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
3	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0
4	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
5	8	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
6	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	0	0
7	9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
9	7	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
10	5	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0
11	6	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
12	4	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0	0	0	0

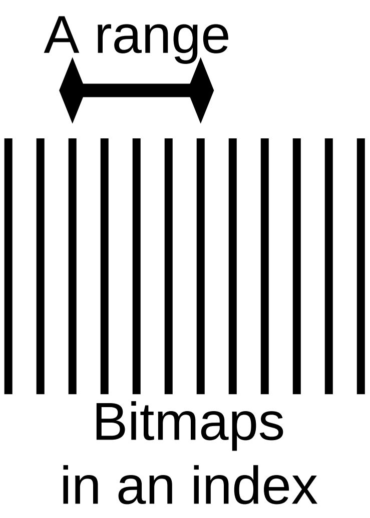
(a) (b) (c)

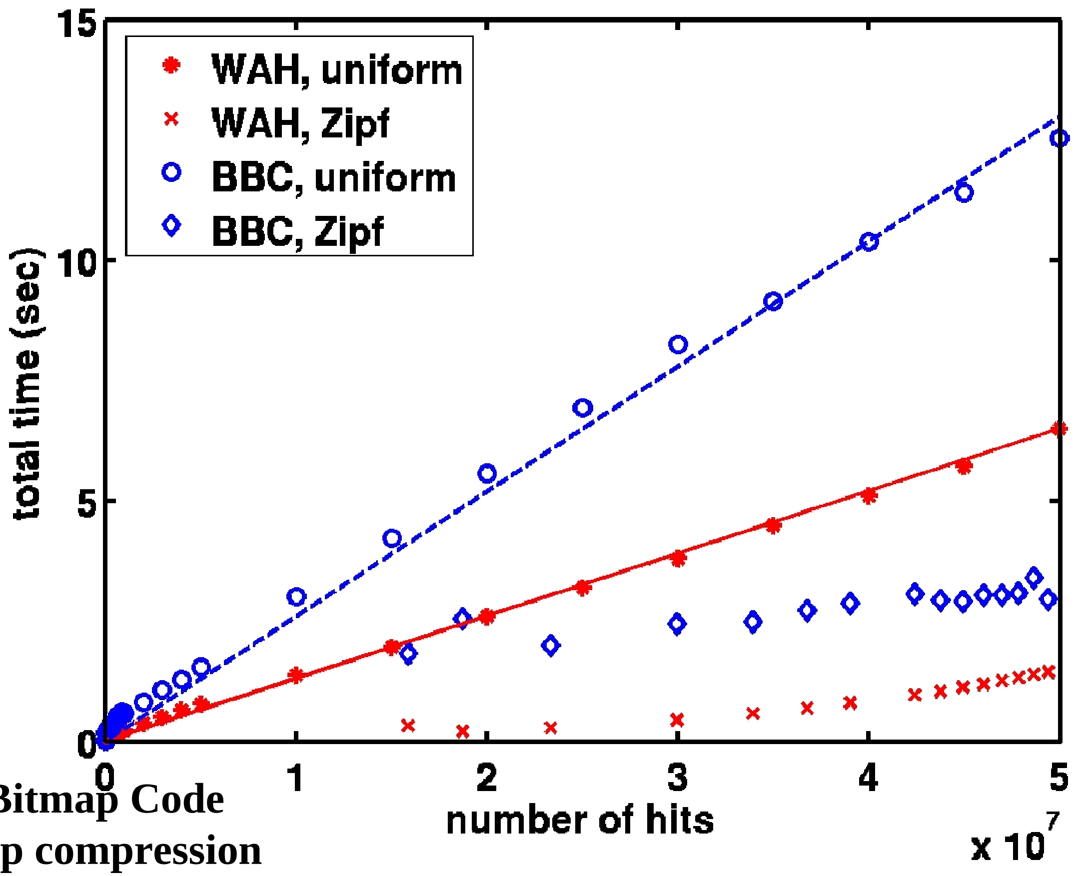
Equality encoding compresses very well

Range encoding optimized for one-sided range queries, e.g. temp < 3

Time to Evaluate a Single-Attribute Range Condition in FastBit is Optimal

- **Evaluating a single attribute range condition may require <u>OR'ing multiple bitmaps</u>**
- **Both analysis and timing measurement confirm that the query processing time is at worst proportional to the number of hits**



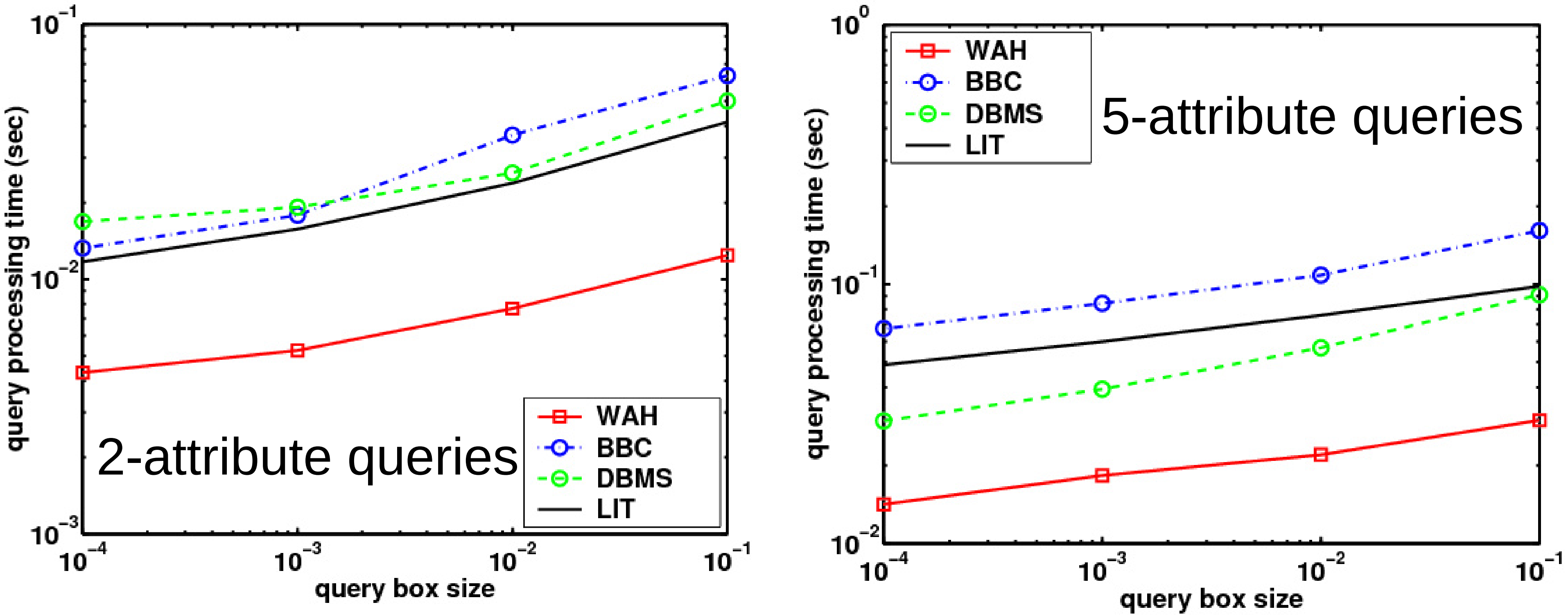


Worst case: Uniform Random Data

Realistic case: Zipf Data

BBC: Byte-aligned Bitmap Code
The best known bitmap compression

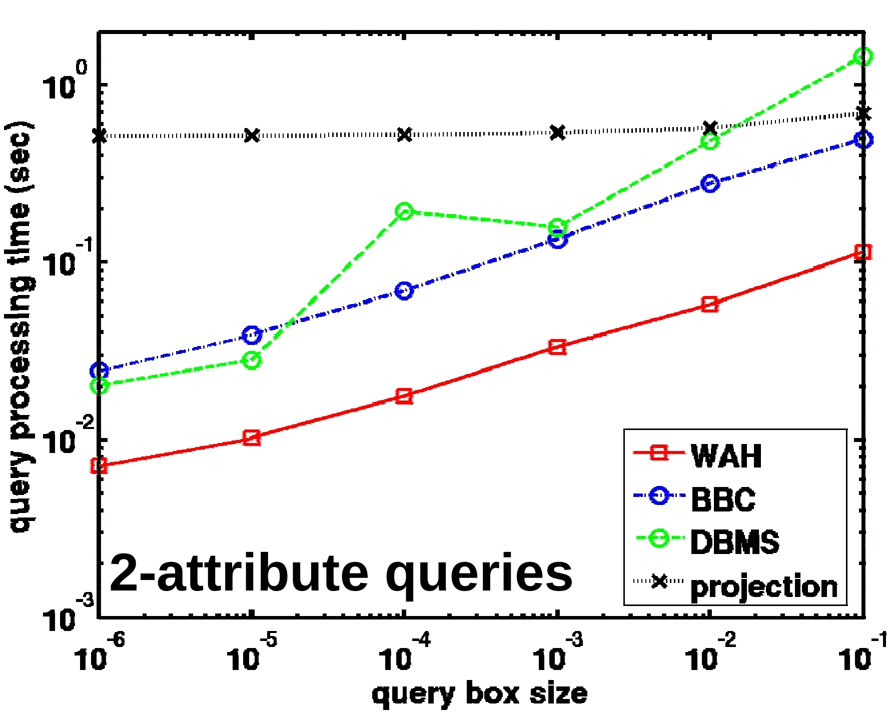
Multi-attribute Range Queries
Low Cardinality Attributes

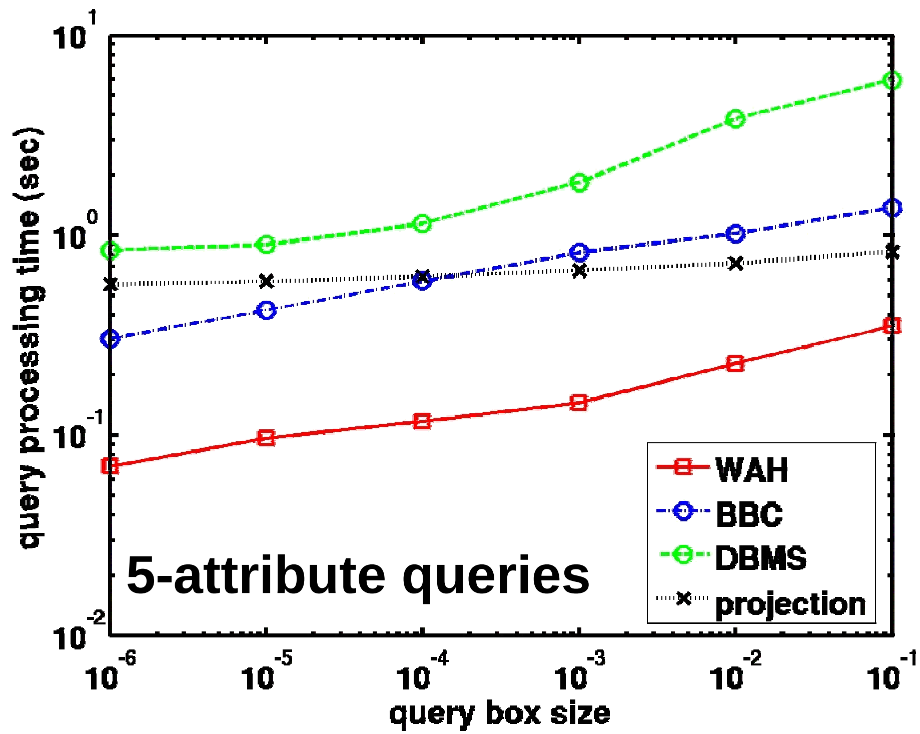


- Bitmap indices are known to work well for low cardinality attributes
- **WAH compressed index is faster than uncompressed index (3X) and BBC compressed index (3X)**

Legend: Query box is the relative volume of the box formed by the range conditions

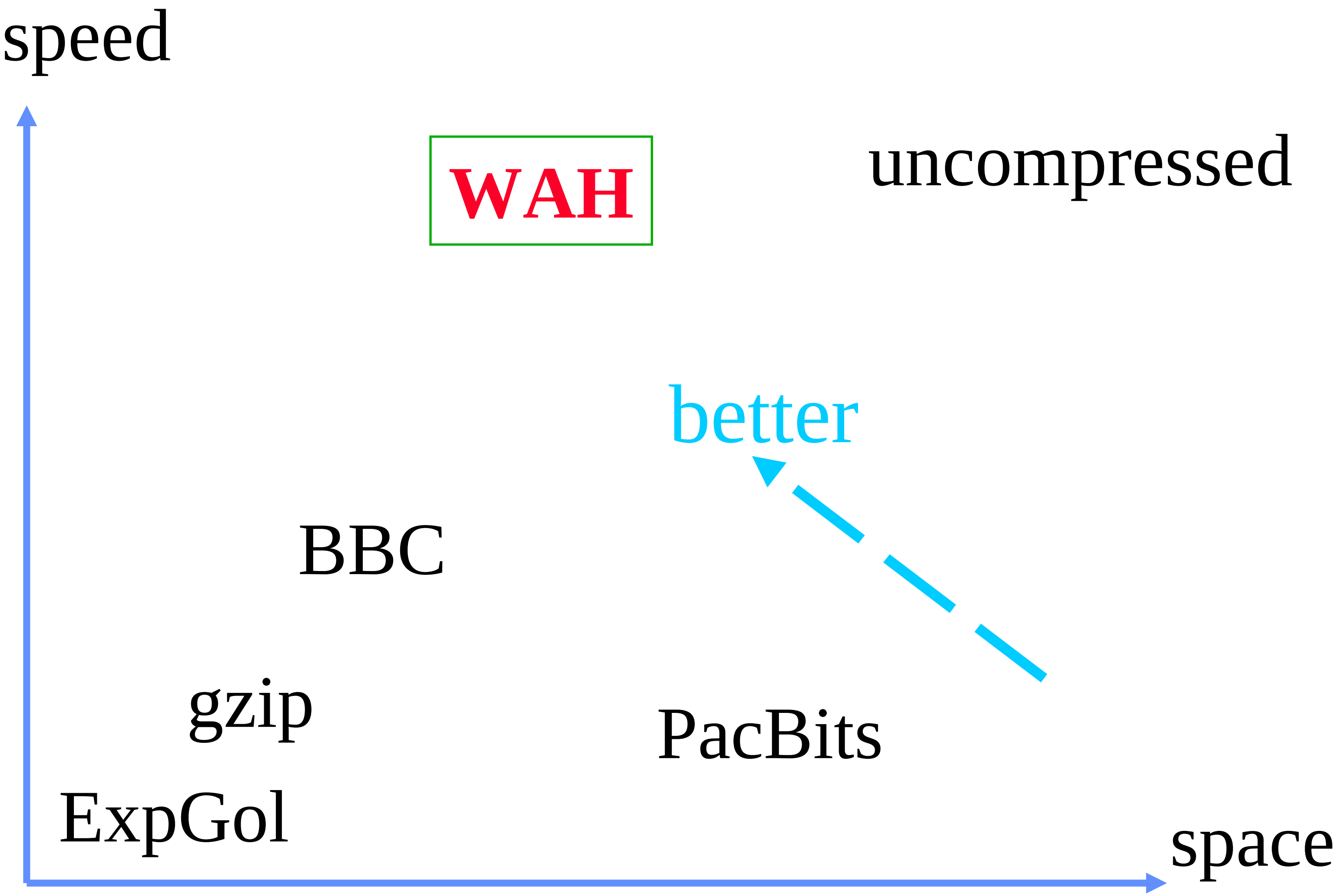
Multi-Attribute Range Queries
High Cardinality Attributes



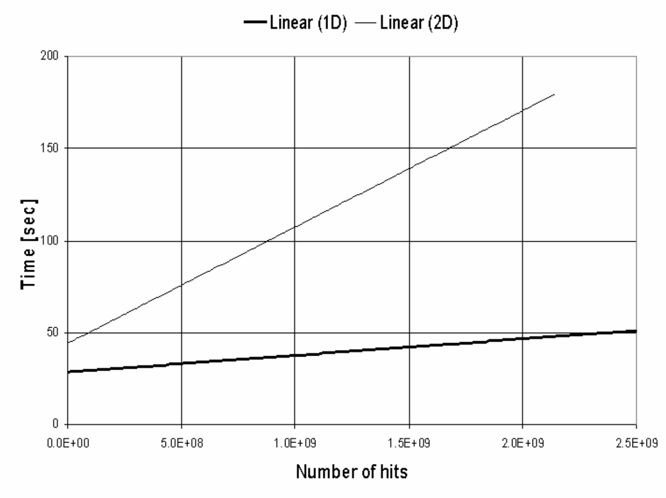


- **WAH works efficiently on HIGH CARDINALITY as well!**
- **WAH compressed indexes are 10X faster than DBMS, 5X faster than our own version of BBC**
  - Based on 12 most queried attributes from STAR, average attribute cardinality 222,000

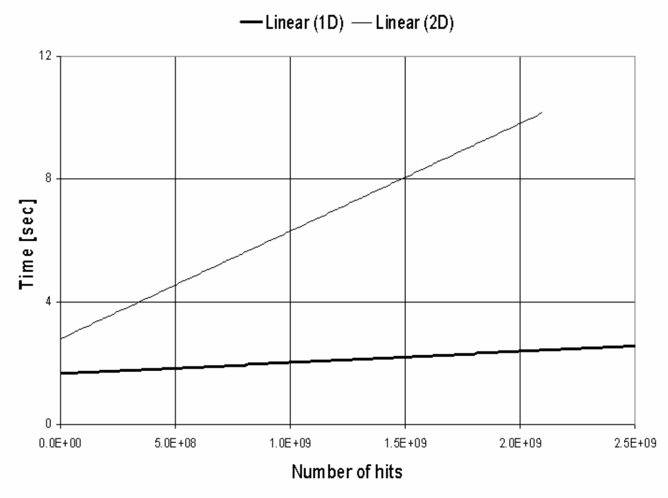
Trade-off of Compression Schemes



FastBit provides real time search of up to a billion elements



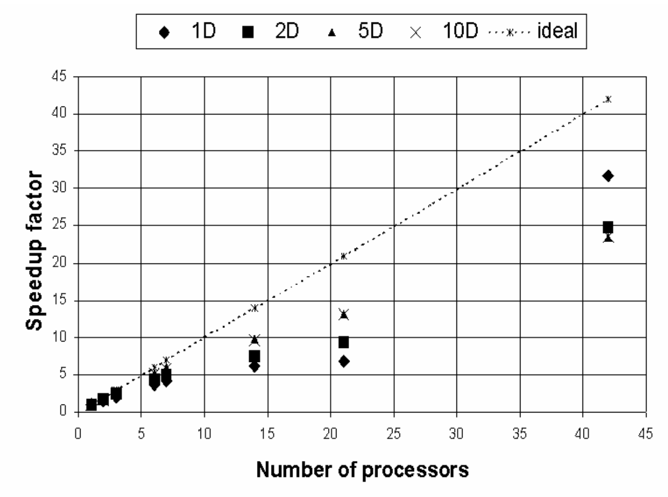
Search over 2.5 billion objects
on a single processor



Search over 2.5 billion objects
on a 42 processor

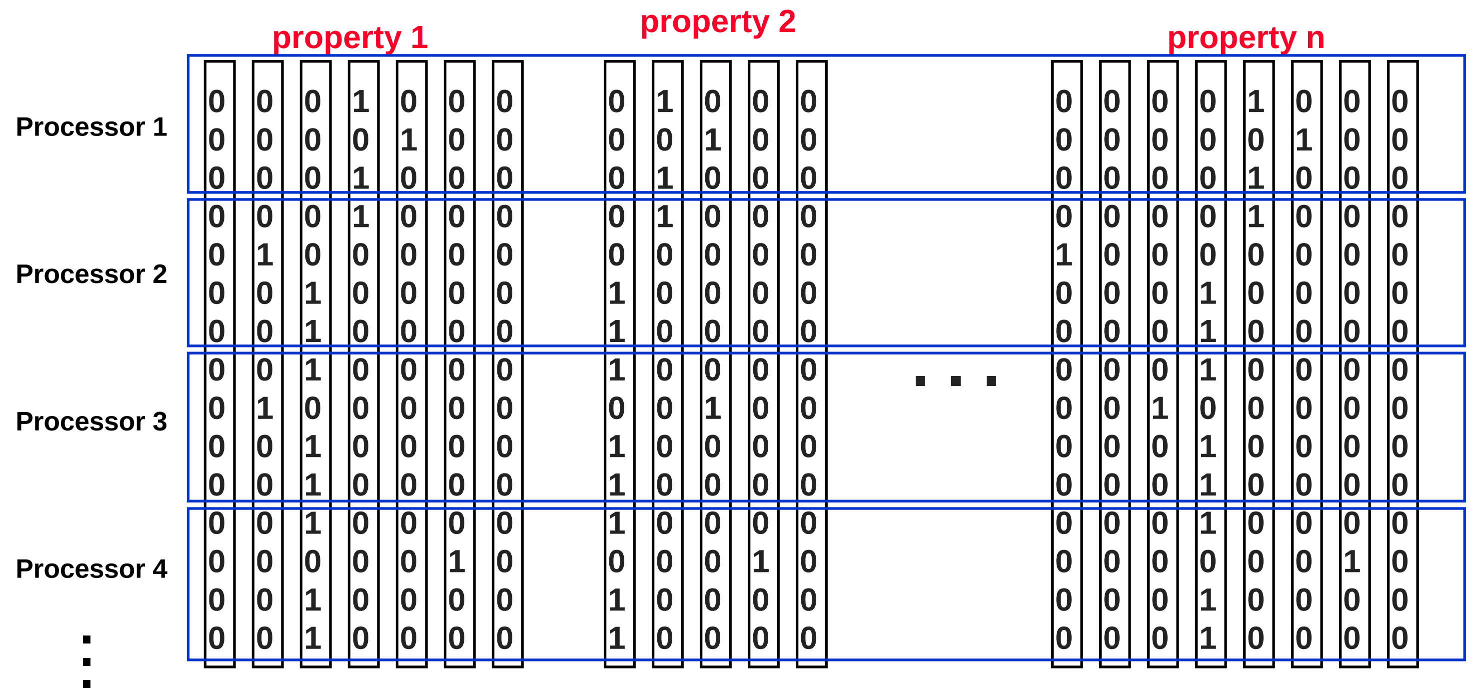
Rate on a single processor:
- about 15 sec per billion on a 1D query
- about 40 sec per billion on a 2D query

Speedup on parallel processors:
50% - 90% of ideal depending on balancing factor

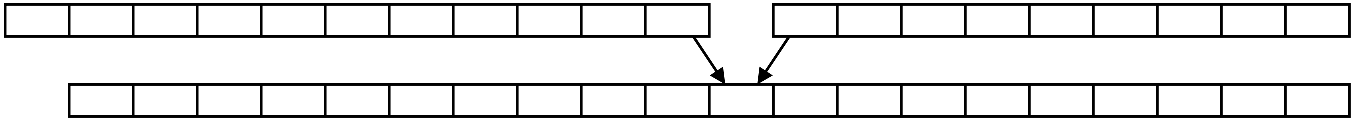


Can be Easily Parallelized for Very Large Indexes

- **Partition bitmaps horizontally**



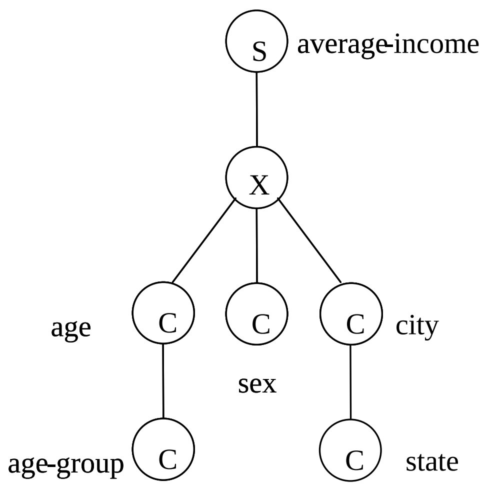
- **Concatenate partial results**
- **Open last and first word only – possible because of 31 bit words**

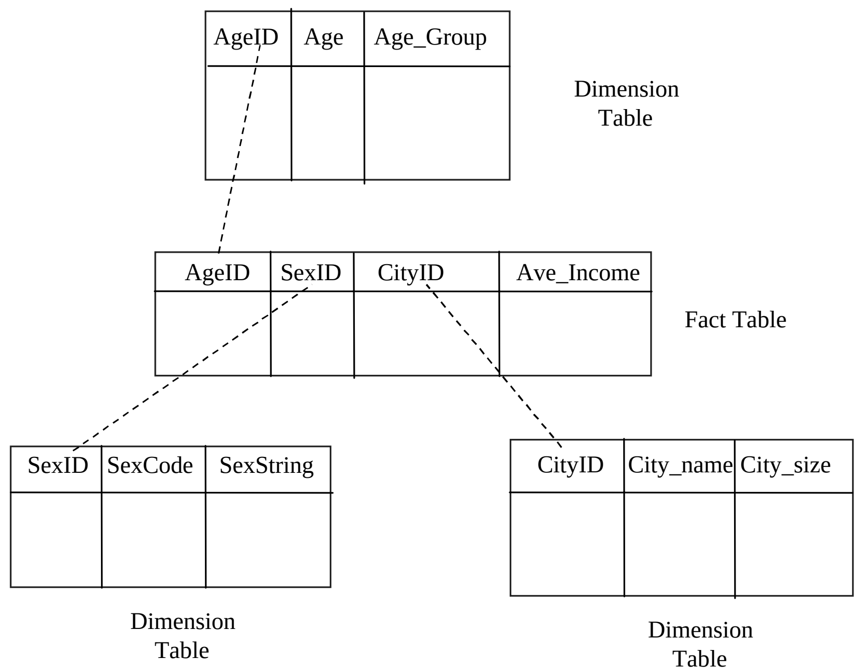


Other Application Domains

On-Line Analytical Processing (OLAP)

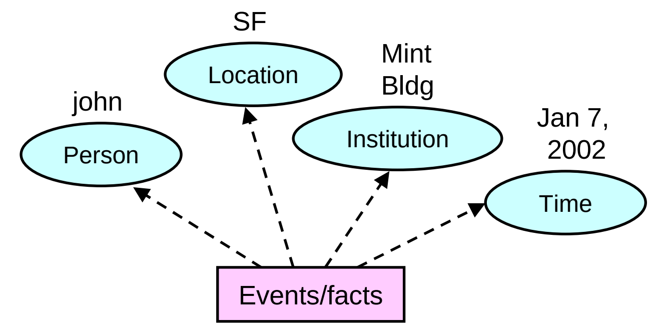
- Consists of multi-dimensional (cross-product) object space
  - e.g. city X sex X age
- Each Dimension can be based on a Category Hierarchy
  - e.g. state → city
- Each multi-dimensional object has one (or more) summary element associated with it
  - e.g. average_income, population, …





Applying FastBit for Information Searching

- **FastBit can be used to represent:**
  - **Events, relationships as high-dimensional structures**
  - **Ontology as highly compressible multi-dimensional structures**
- **Potential to provide**
  - **On-line search capability over very large high-dimensional objects**
  - **Dynamic incremental index building**

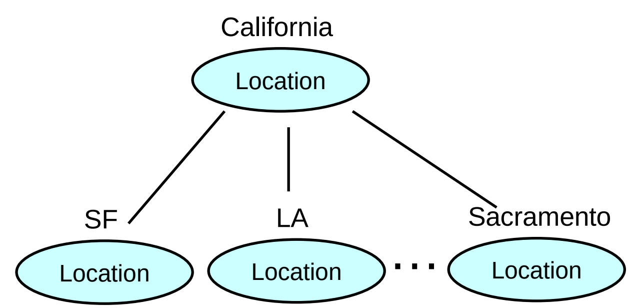


Visited

Event_ID	Action	Person	Location	Institution	Time
12375	Visit	John	SF	Mint Bldg	02.01.07
12388	Visit	Mary	SF	Mint Bldg	02.01.10
…	…	…	…	…	…

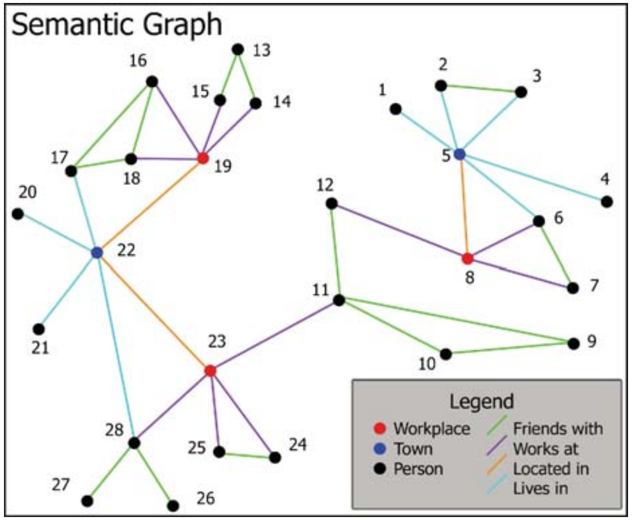
State							City				
0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0

State	City	address
California	SF	Flower St
California	SF	Folk St
…	…	…
California	LA	Alvarado St
…	…	…



Ontology

Representing Semantic Graphs as Bitmaps



Edge Connectivity (very large)

Edge_ID	edge-type	from-node	from-type	to-node	to-type
12375	works-at	3365	person	377	workplace
680257	lives-in	320045	person	4005	town
…	…	…	…	…	…

Country

0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	1	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0

restrict

restrict

Workplace node attributes (small-medium)

Workplace_ID	Name	type-of-business	Address	Country
3365	SF Mint	print money	13 Market St. San Francisco, CA	USA
3365	J&J	Hardware store	1 James Rd. Columbus, OH	USA
…	…	…	…	…

…

Person node attributes (small-medium)

Person_ID	Name	Birthplace	Age	Sex	Address	Country
3365	John	Germany	32	M	13 main St. Los Angeles	USA
320045	Maria	US	24	F	7 Ortega St. Mexico City	Mexico
…	…	…	…	…	…	…

Summary: FastBit Technology

- Main ideas
  - Take advantage of append-only data
  - Vertical partitioning – touch only attributes in query
  - Binning – touch only bins in ranges specified
  - Use compression that permits efficient logical operation on uncompressed bitmaps
  - Straight-forward parallelism – horizontal partitioning of bitmaps, and concatenation of result bitmaps
- Results
  - Real-time compute-optimized search for multi-dimensional data
  - Particularly effective for partial range queries
  - Small index size – on average about 50% of data size
  - Search only part of index – time proportional to size of result
  - Dynamic building of index – 1 sec per million elements
  - Scales to billions of objects and hundreds of attributes per object
  - Found effective and useful for on-line scientific data exploration

FastBit Technology Details

Contacts:

John Wu, Arie Shoshani, Ekow Otoo,

Kurt Stockinger

http://sdm.lbl.gov/fastbit