The STAR Grid Collector
and
TBitmapIndex

John Wu

Kurt Stockinger, Rene Brun, Philippe Canal – TBitmapIndex

Junmin Gu, Jerome Lauret, Arthur M. Poskanzer, Arie Shoshani, Alexander Sim,
Wei-Ming Zhang – Grid Collector

Outline

- TBitmapIndex preview
  - — A preliminary integration of FastBit and ROOT
- Grid Collector for STAR
  - — Using FastBit as an efficient event filter
- FastBit searching technology
  - — A set of efficient compressed bitmap indices

TBitmapIndex: An attempt to introduce FastBit to ROOT

Kurt Stockinger[1], Kesheng Wu[1], Rene Brun[2], Philippe Canal[3]

(1) Berkeley Lab, Berkeley, USA

(2) CERN, Geneva, Switzerland

(3) Fermi Lab, Batavia, USA

Current Status

- FastBit:
  - Bitmap Index software developed at Berkeley Lab
  - Includes very efficient bitmap compression algorithm
- Integrated bitmap indices to support:
  - TTree::Draw
  - TTree::Chain
- Each Index is currently stored in a binary file

Example - Build Index

```
// open ROOT-file
TFile f("data/root/data.root");
TTree *tree = (TTree*) f.Get("tree");

TBitmapIndex bitmapIndex;
char indexLocation[1024] = “/data/index/";

// build indices for all leaves of a tree
bitmapIndex.BuildIndex (tree, indexLocation);

// build index for two attributes “a1”, “a2” of a tree
bitmapIndex.BuildIndex(tree, "a1", indexLocation);
bitmapIndex.BuildIndex(tree, "a2", indexLocation);
```

Example - Tree::Draw with Index

```
// open ROOT-file
TFile f("data/root/data.root");
TTree *tree = (TTree*) f.Get("tree");

TBitmapIndex bitmapIndex;
bitmapIndex.Draw(tree, "a1:a2", "a1 < 200 && a2 > 700");
```

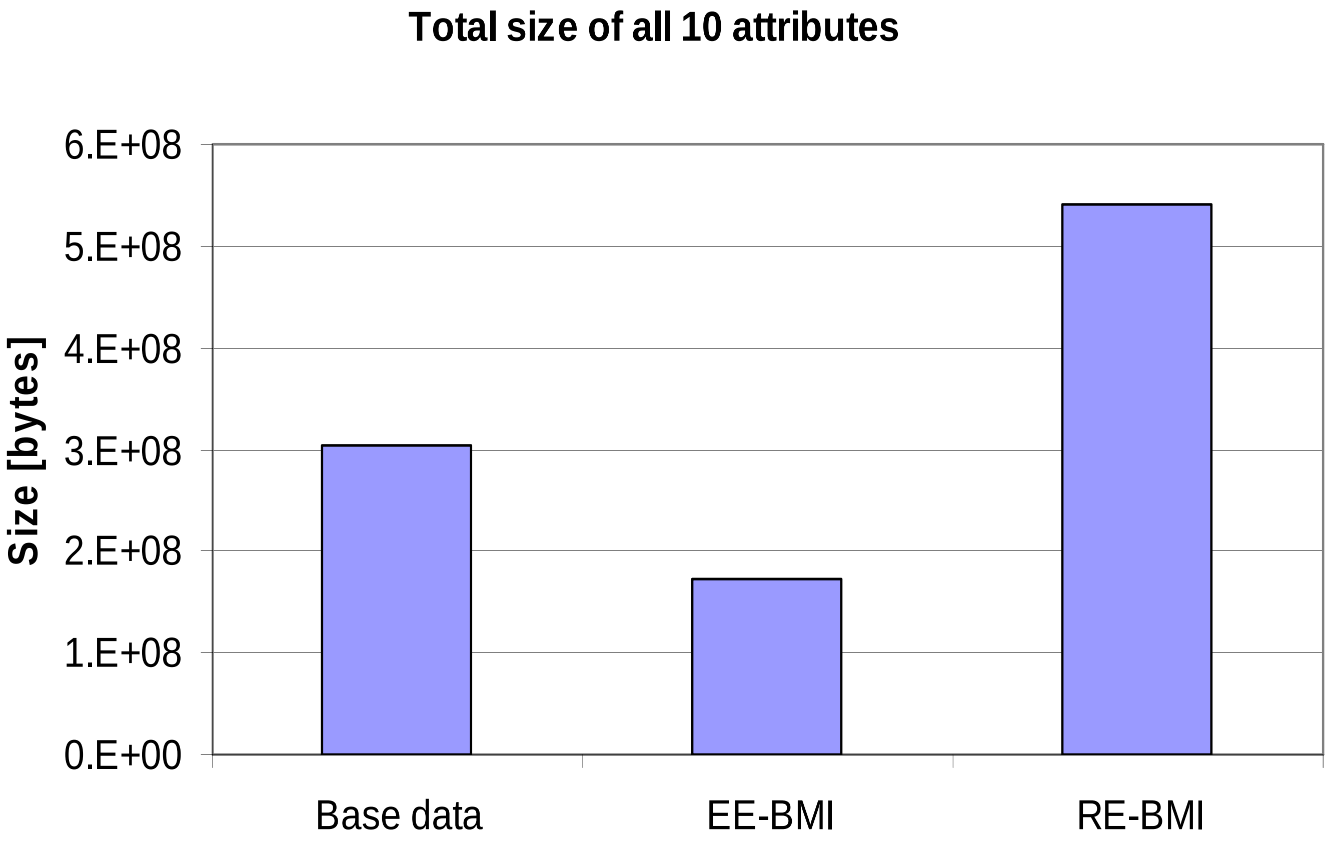
Performance Measurements

- Compare performance of TTreeFormula with TBitmapIndex::EvaluateQuery
- Do not include time for drawing histograms
- Run multi-dimensional queries (cuts with multiple predicates)

Experiments With BaBar Data

- **Software/Hardware:**
  - **Bitmap Index Software is implemented in C++**
  - **Tests carried out on:**
    - **Linux CentOS**
    - **2.8 GHz Intel Pentium 4 with 1 GB RAM**
    - **Hardware RAID with SCSI disk**
- **Data:**
  - **7.6 million records with ~100 attributes each**
  - **Babar data set:**
- **Bitmap Indices (FastBit):**
  - **10 out of ~100 attributes**
  - **1000 equality-encoded bins**
  - **100 range-encoded bins**

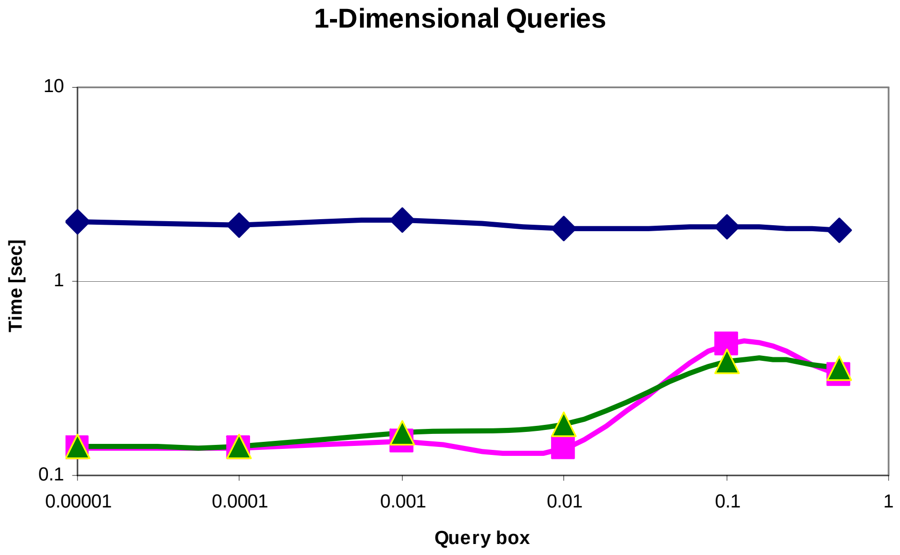
Size of Compressed Bitmap Indices

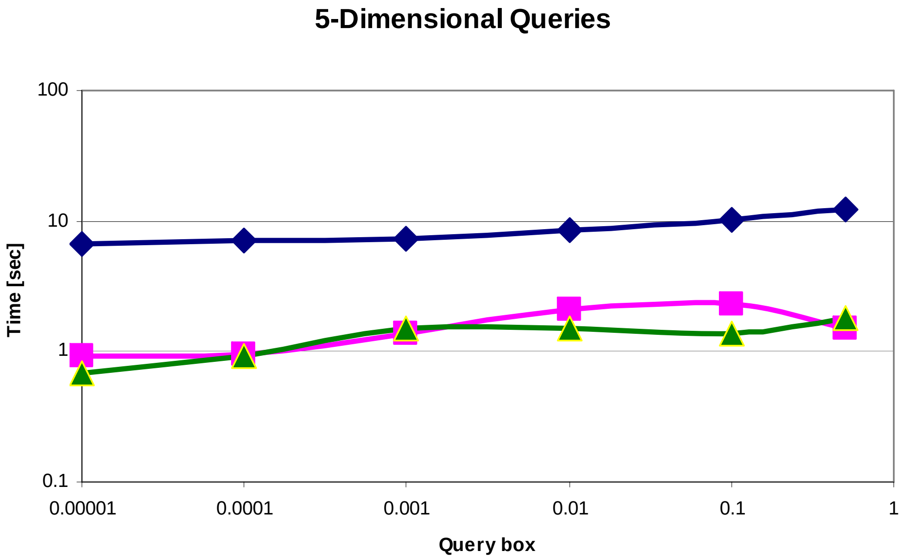


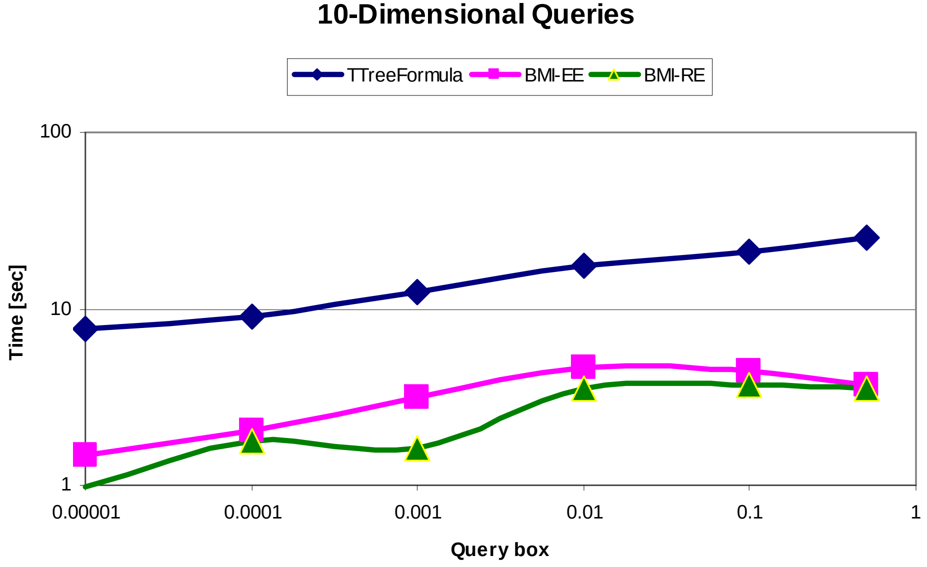
EE-BMI: equality-encoded bitmap index

RE-BMI: range-encoded bitmap index

Query Performance - TTreeFormula vs. Bitmap Indices







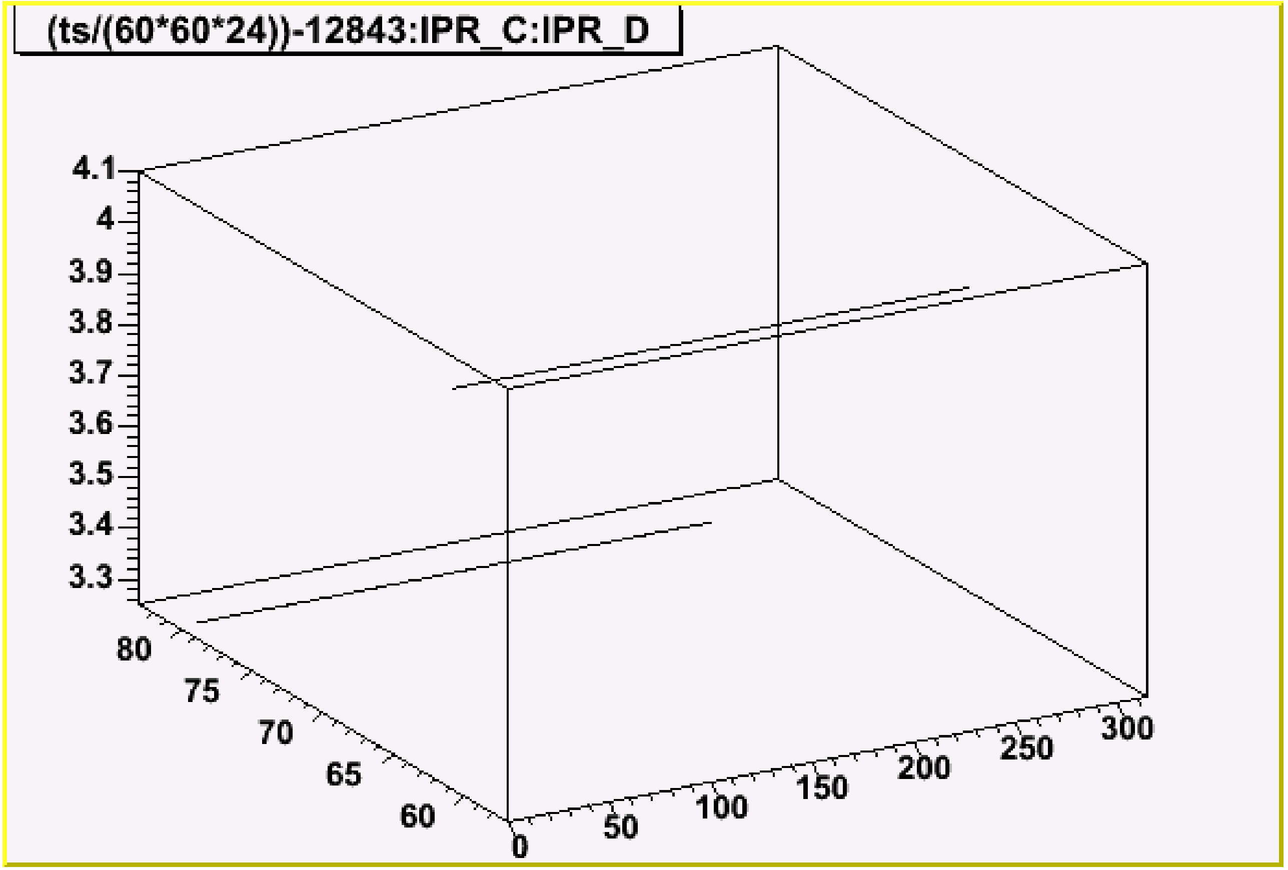
Bitmap indices 10X faster than TTreeFormula

Network Flow Analysis: Another Example

- IDS log shows
  - Jul 28 17:19:56 AddressScan 221.207.14.164 has scanned 19 hosts (62320/tcp)
  - Jul 28 19:19:56 AddressScan 221.207.14.88 has scanned 19 hosts (62320/tcp)
- Using FastBit/ROOT to explore what else might be going on

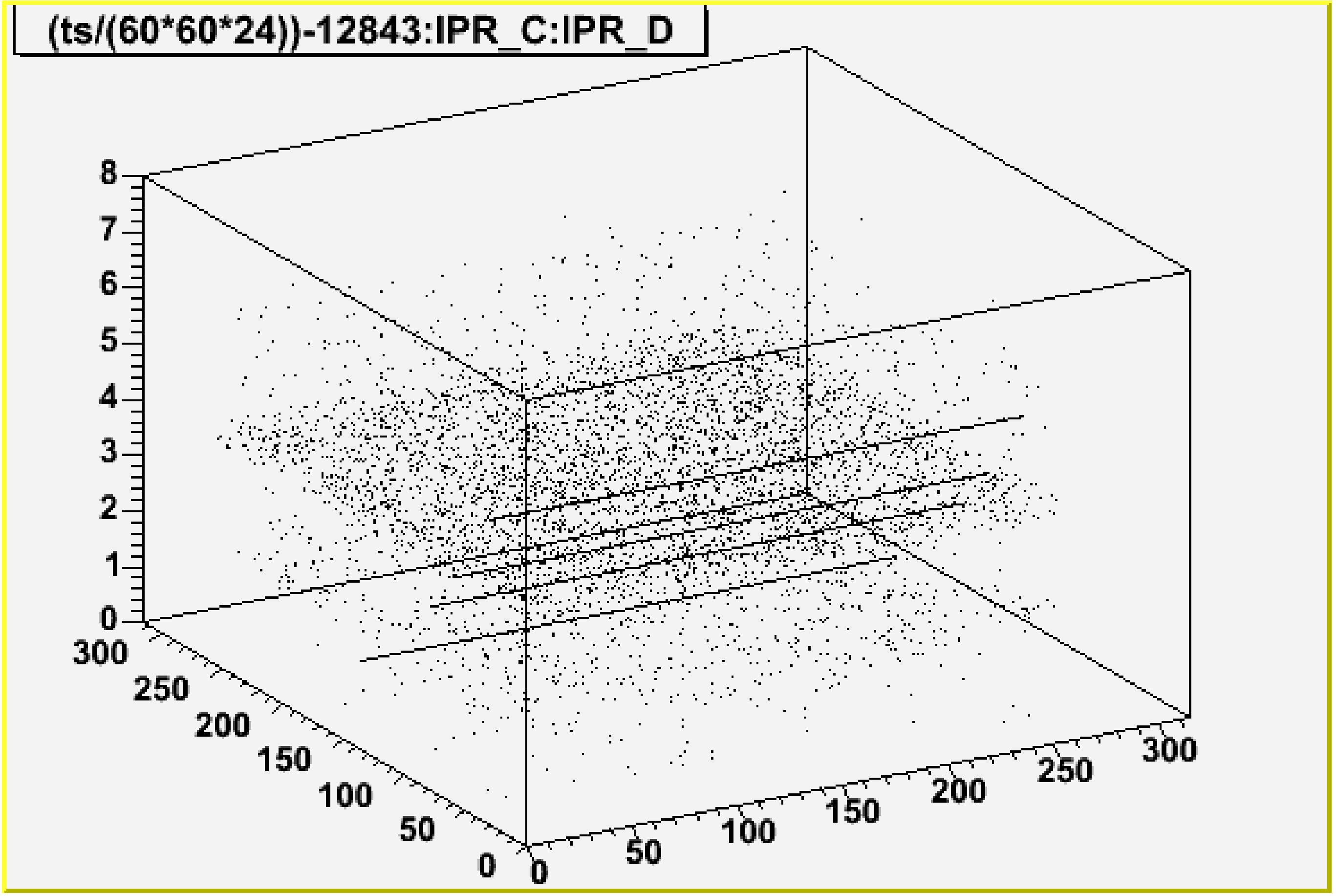
- Queries prepared by Scott Campbell. More details at http://www.nersc.gov/~scottc/papers/ROOT/rootuse.prod.html

See the Scans from the Two Hosts



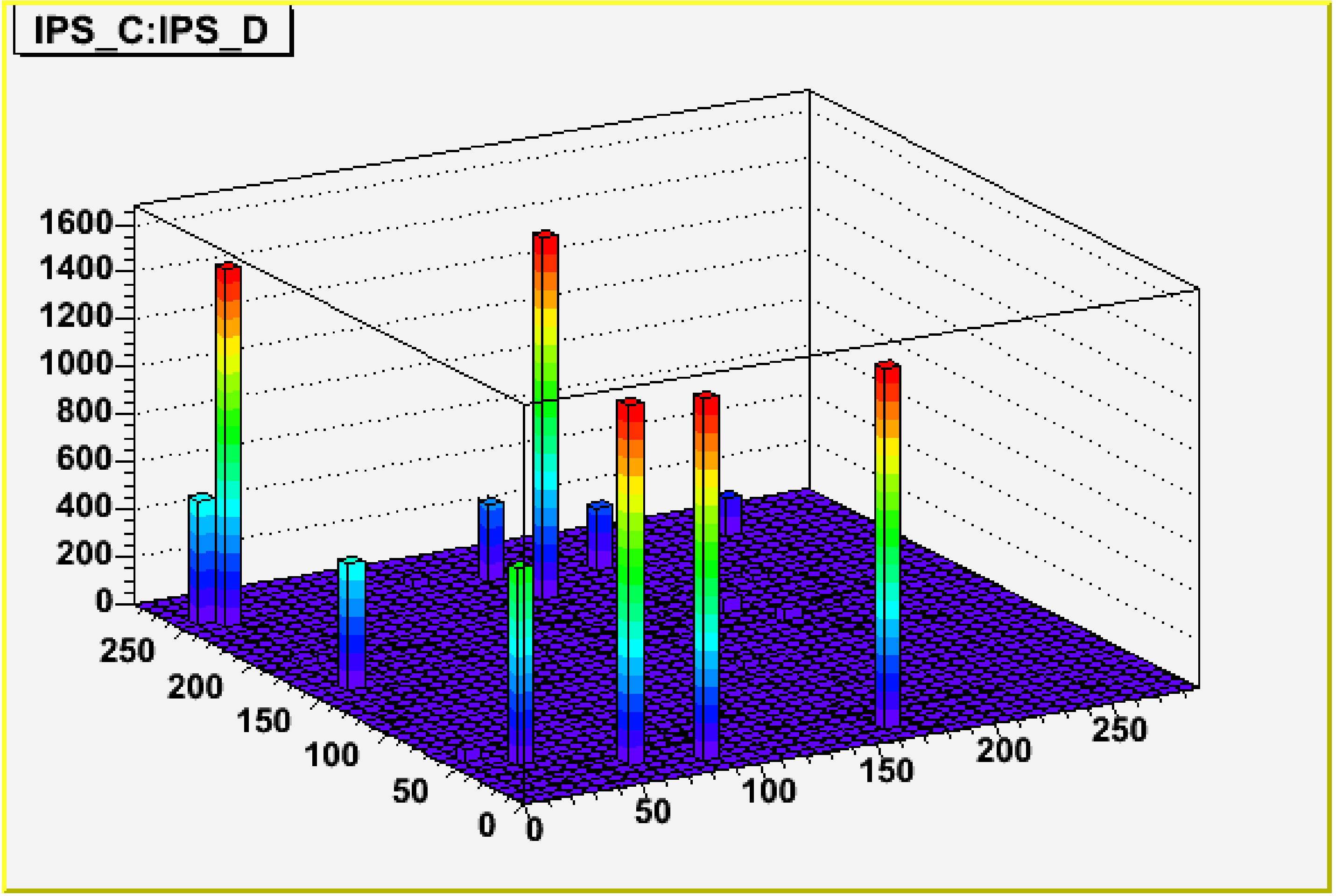
- Query: select ts/(60*60*24)-12843, IPR_C, IPR_D where IPS_A=211 and IPS_B=207 and IPS_C=14 and IPS_D in (88, 164)
- Picture: scatter plot (dots) of the three selected variables
- Two lines indicating two sets of slow scans

Are There More Scans?



- Query: select ts/(60*60*24)-12843, IPR_C, IPR_D where IPS_A=211 and IPS_B=207
- More scans from the same subnet

Who Is Doing It?



- **Query: select IPS_C, IPS_D where IPS_A==211 and IPS_B==207**
- **Picture: the histogram of the IPS_C and IPS_D**
- **Five IP addresses started most of the scans!**

Grid Collector

Put FastBit and SRM together to improve the efficiency of STAR analysis jobs

http://www.star.bnl.gov/

Design Goals of Grid Collector

Goals

Make scientific analysis more productive by

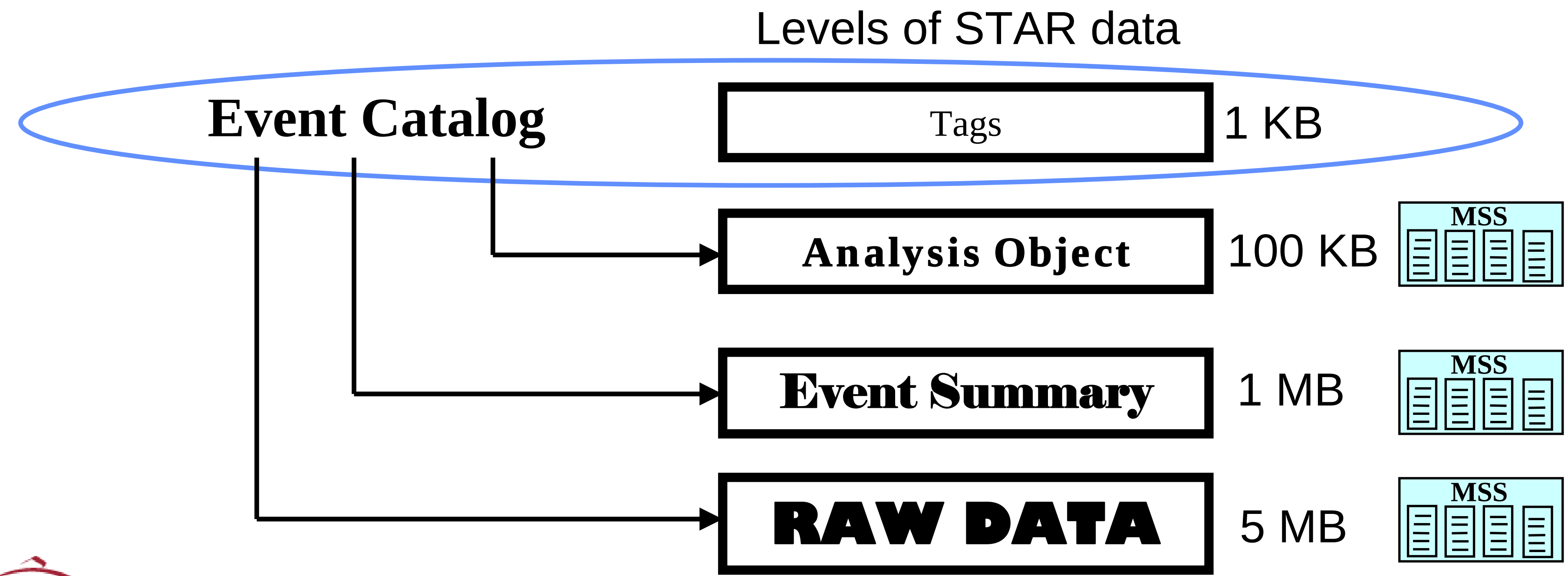
- Specifying events of interest using meaningful physical quantities
  - — numberOfPrimaryTracks > 1000 AND SumOfPt > 20
- Reading only events selected
- Automating the management of distributed files and disks

Practical considerations

- Working in the existing analysis framework
- Overhead should be insignificant
- Efficient for finding a few events (e.g., rare events) as well as a large number of events (e.g., for statistical analysis)

Using FastBit to Build STAR Event Catalog

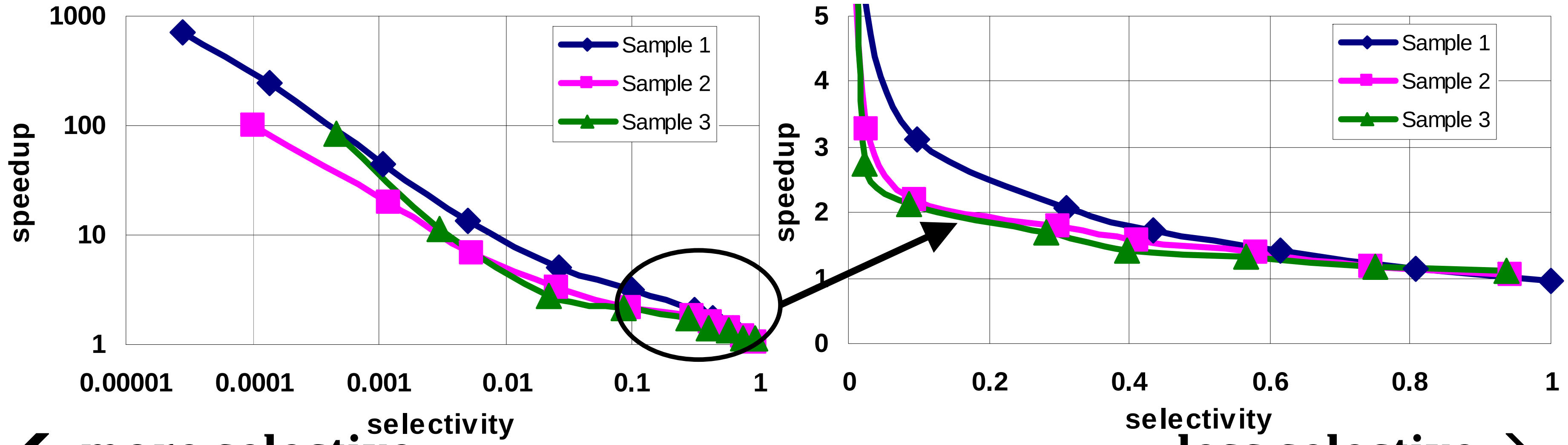
- STAR data is organized into several levels
- The Event Catalog indexes all tags but only maintains references to other levels



Key Steps of Analysis Process

1. Locate the files containing the events of interest
   - FastBit Event Catalog to associate events with files
   - File & replica catalogs for locations of files
2. Prepare disk space and transfer
   - Storage Resource Managers (SRMs):
     - Prepare disk space for the files
     - Transfer the files to the disks from HPSS
     - Recover from HPSS and network transfer failures
3. Read the events of interest from files
   - Event Iterator with fast forward capability using information from the Event Catalog
4. Remove the files
   - SRMs perform garbage collection

Grid Collector Speeds up Analyses



← more selective

less selective →

- Legend
  - Selectivity: fraction of events selected for an 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 Rare-Event 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

A compressed bitmap indexing technology for efficient searching of read-only data

http://sdm.lbl.gov/fastbit

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

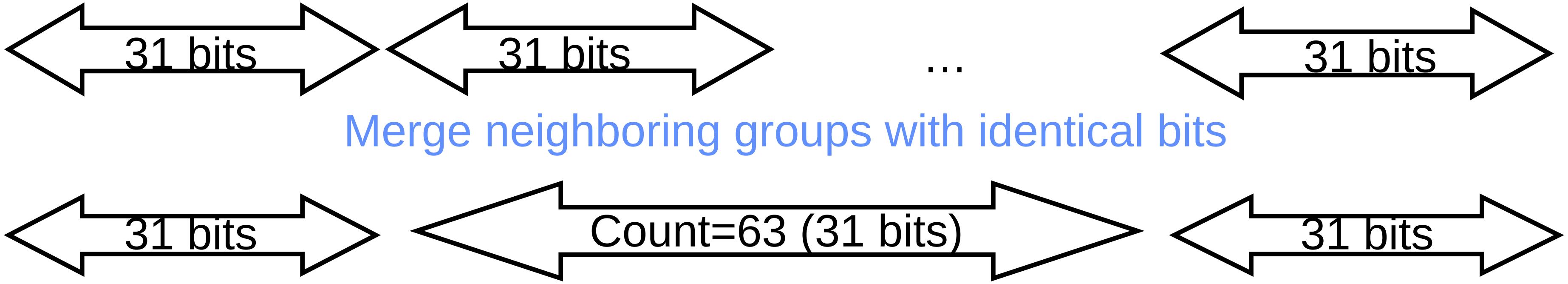
- First commercial version
  - Model 204, P. O’Neil, 1987
- Easy to build: faster than building B-trees
- Efficient to query: only bitwise logical operations
  - A < 2 → b_0 OR b_1
  - 2<A<5 → b_3 OR b_4
- Efficient for multi-dimensional queries
  - Use bitwise operations to combine the partial results
- 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

The Special Compression Method in FastBit Is Compute-Efficient

Example: 2015 bits

100000000000000000000111000000000000000000000000000000000……………….000000000000000000000000000000011111111111111111111111111

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

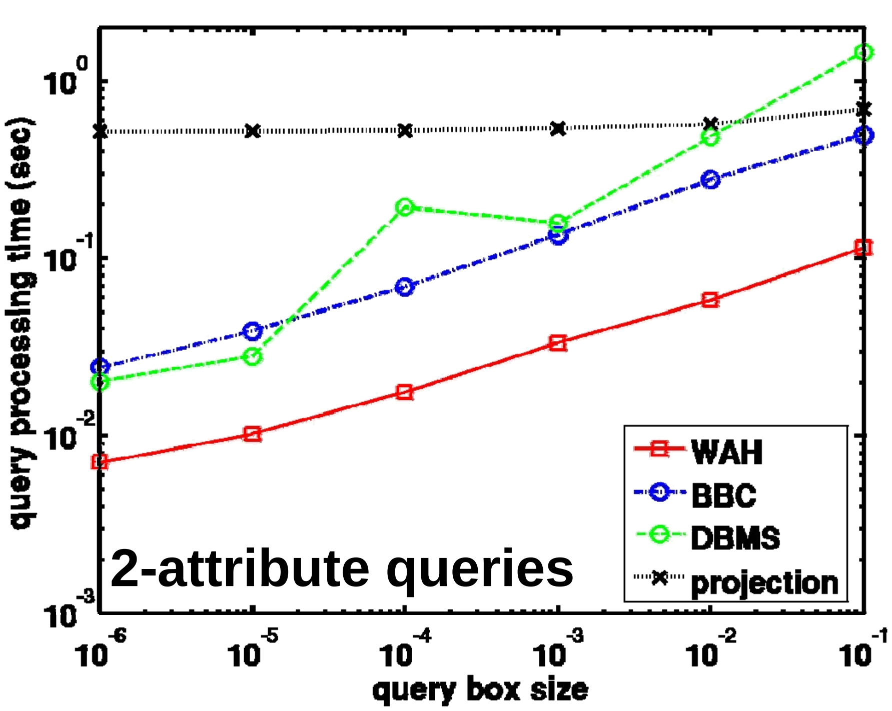


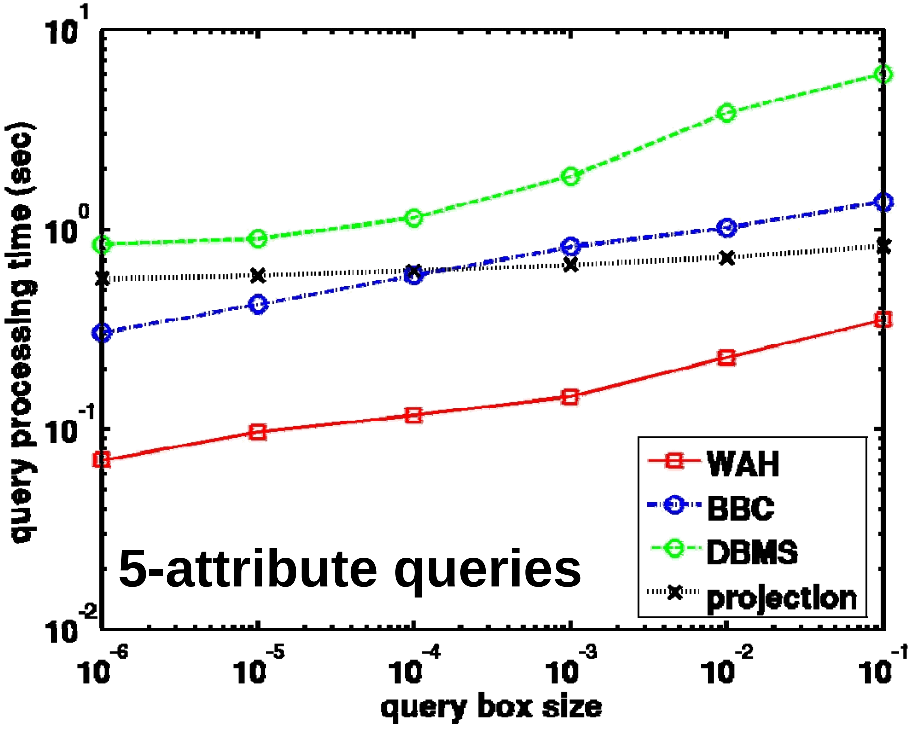
Merge neighboring groups with identical bits

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

Performance on Multi-Attribute Range Queries





- **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

Summary / Future Work

- We integrated bitmap indices into ROOT to support:
  - TTree::Draw
  - TChain::Draw
- Using bitmap index speeds data selection by up to 10X
  - With approximate answers of 0.1-1% error the performance improvement is up to a factor of 30
- Bitmap indices are also used successfully in STAR as a form of Event Index to speed event access
- Future work:
  - Tighter integration with ROOT to provide more functionality
  - Store bitmap indices in ROOT files
  - Integrate with PROOF to support parallel evaluation