Compressing Data Cube in Parallel OLAP Systems

Bo-Yong Liang

School of Computer Science, Carleton University
Ottawa Canada

Agenda

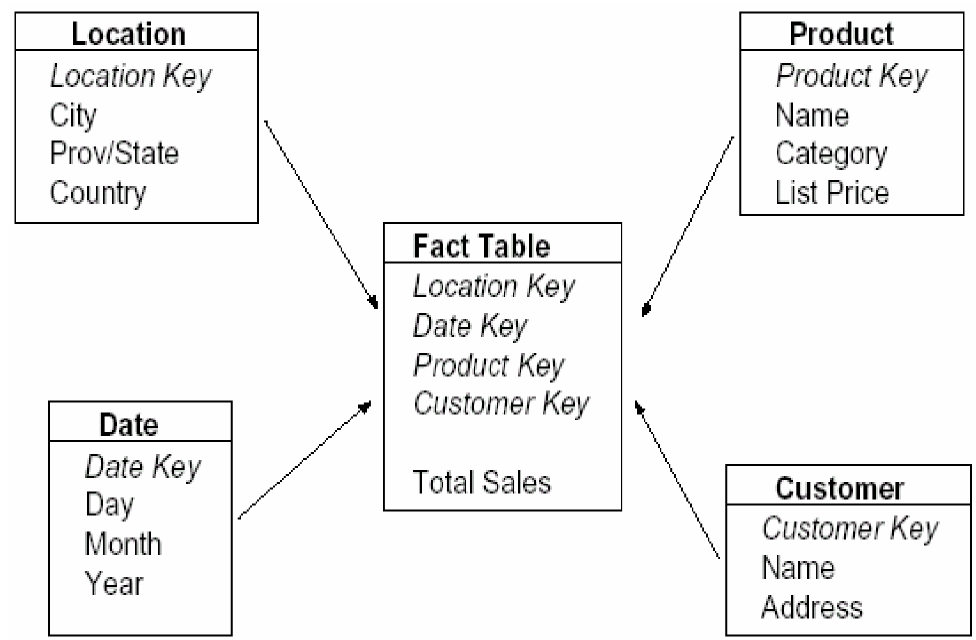
- Purpose of the Project
- Background and Related work
- Data Cube Compression Algorithm
- Evaluation and Conclusion
- Future Work
- Reference

Purpose

- Using data compression techniques in high performance of OLAP computation
  - Focus on compressing data cube to
    - Reduce data storage space
    - Reduce I/O access bandwidth
  - Working on an efficient Parallel OLAP system - PANDA[1]

Data Warehouse

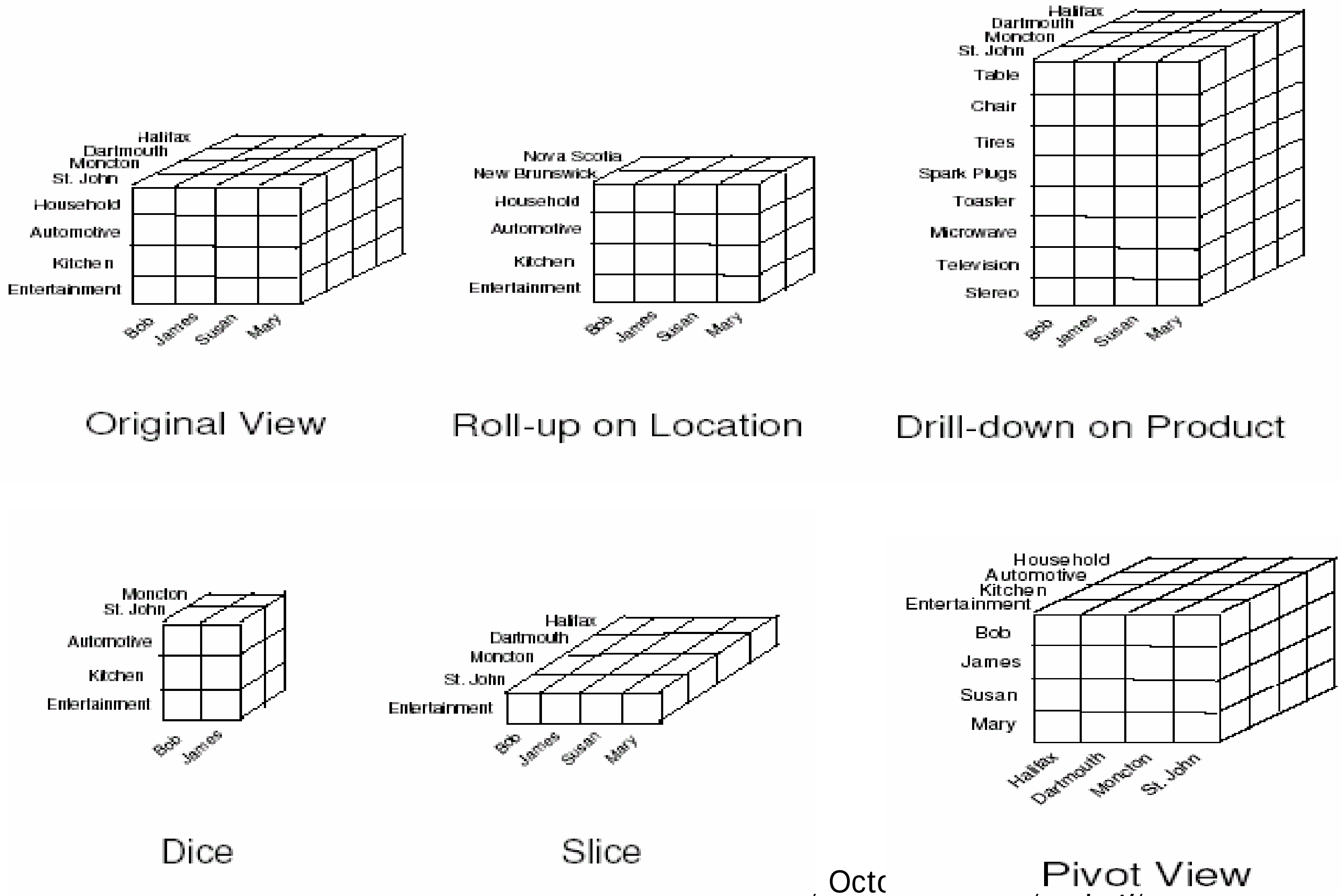
- Multi-dimensional model
  - Alternative Entity-Relationship (E/R) modeling
  - Dimension tables
    - surrogate keys
  - Fact table
    - combining key
    - summary fields



Data Warehouse – Cube & OLAP

- Data cube
  - Cells - measure values
  - Edges – dimensions
- On-Line Analytical Processing (OLAP)
  - Drill-down, Roll-up, Slice, Dice, Pivot
- Data cube properties
  - Massive data: 2^d views
  - Pre-computed views
  - Dynamically views

OLAP Operations - Examples



Data Compression

- Categories - lossless
  - Statistical data modeling
    - Huffman, Arithmetic
  - Dictionary algorithms
    - Lempel-Ziv (WinZip, GZIP), Block-sorting (BZIP)
  - Others: Run-length-coding(RLC)
- Properties
  - Serialization (FIFO)
  - Consistency (Data model)

Database Compression

- Issues of database compression
  - Keep relation structures for random query
  - Avoid decompress a large portion of data
  - Use relation knowledge for high compression ratio
- Compressing relations with numeric attribute domains
  - BIT
  - Goldstein's (Block-BIT) [5]
  - TDC [2]

Database Compression - BIT

- BIT
  - Represents numerical attributes in bits, instead of bytes.
  - Advantages
    - Fast query - Keep the structure of relation very well
    - Fast de/compression – no complicate data model
- Goldstein's Algorithm[5] – Block-BIT
  - Compress relation by block – physical IO unit
  - Use the smallest values of each attributes as reference
  - For each attribute, only store difference of the reference

Database Compression - TDC

- Tuple Differential Coding – TDC[2]
  - Tuples are converted into ordinal numbers in ascending mixed-radix order.

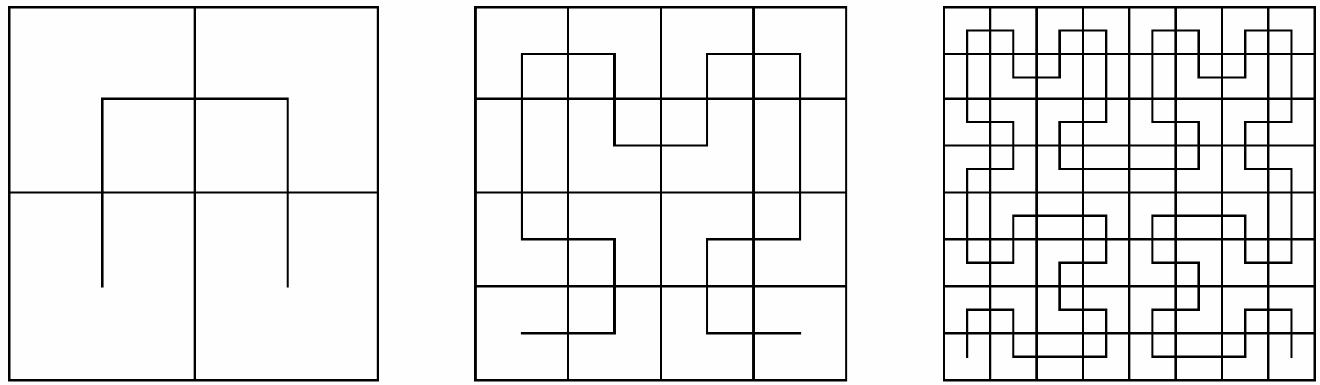
$$\varphi(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} \left(a_i \prod_{j=i+1}^{n} |A_j| \right)$$

  - A compressed block only stores
    - a value of the first tuple as reference.
    - Each succeeding tuple is replaced by its difference with respect to its preceding tuple

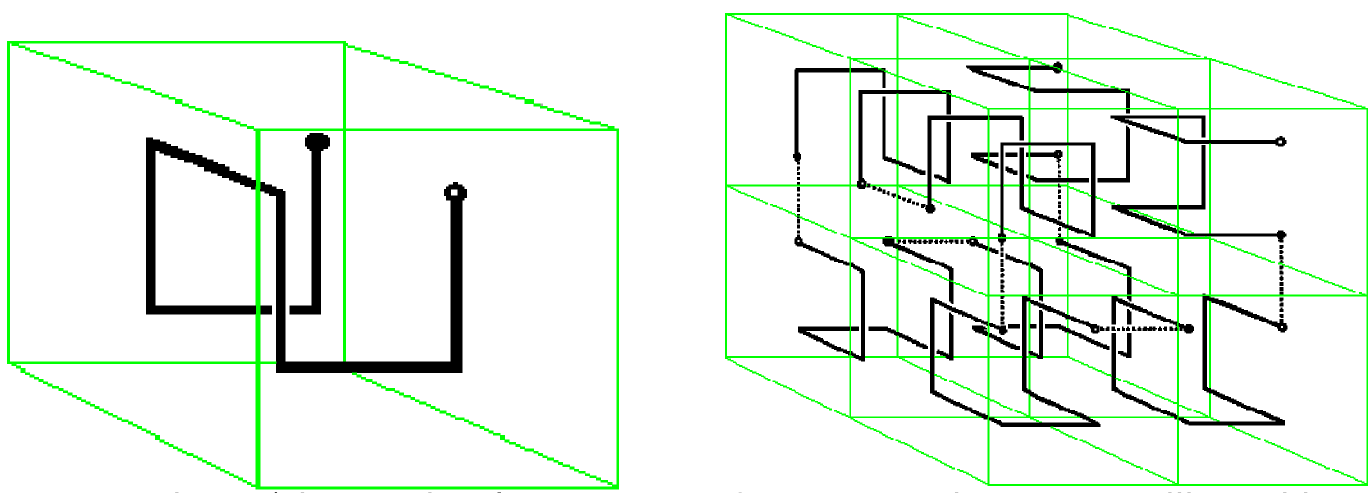
Hilbert Curve

- Hilbert Space Filling
  - A continuous one dimensional curve that passes through every point of a multidimensional space
  - Property: points near one another in the original space are closed in the linearly ordered space
  - Examples:
    - 2-dimensional Hilbert Curve
    - 3-dimensional Hilbert Curve

Hilbert Curve - Examples



(a) First step (b) Second step (c) Third step



Data Cube Compression

- Some characteristics of Fact Table in Data cube
  - Seldom updated (unlike transaction DBMS)
  - Surrogate keys: integers, consecutive
  - The tuples are sorted
  - Measure data may be integer, float, double …
  - Meta data are known during ETL stage
    - Number of dimensions
    - Cardinality of each dimension
    - …

Data Cube Compression
XTDC Algorithm …

- XTDC Algorithm
  - Compressing dimensional data of views in block level
  - Using tuple differential coding
    - Introduce tuple operations: Tuple_Minus, Tuple_Add
  - Expressing tuple differences in <u>bit</u> in block wise
    - Using compact data structure to remove byte-alignment gaps
  - Counter mechanism
    - Count the number of consecutive tuple with difference equals 1
  - Dynamic block determining
    - Dynamically determine the number of tuples in one block

XTDC Algorithm …

- Algorithm (XTDC)
  - Step 1: Compute difference of conjunctive tuples
    - Dynamic determine number of tuples in the block
    - Count consecutive 1 differences
    - Determine number of bits for each tuple
  - Step 2: Compact the differences into bits
  - Step 3: Copy measure data to 2nd part of block
  - Step 4: Create block header

XTDC - Data Structure

- Data Structure
  - Block Header
    - First tuple, #tuple, #bit for each difference, counter
  - Dimension segment – compressed data (differences in bits)
  - Summary fields segment – summary fields
- Advantages
  - Keep the relation structure – fast query
  - Remove Byte-Alignment gap – high compression ratio
  - Opportunity to compress Summary fields later

XTDC – Data Structure…

- Data structure

Block header	Length of the header # of tuple # of bits of difference # of bytes of measure data Counter First tuple (original form)
Dimensional data	Difference between 2nd tuple and 1st one ……
Measure data	Measure of 2nd tuple ……

XTDC

Example

- Example
  - Dimensions: 4
  - Cardinalities: 10
  - Block size: 40B
  - Header: 32B
- Process
  - Tuple values (b)
  - Differences (c)
  - Block (d)
    - Two segments
    - Dynamically
    - counter

	View (a)	Tuple values (b)	Differences (c)
1	<1,1,1,1,3>	1111	1111
2	<1,1,2,1,9>	1121	10
3	<1,1,2,2,15>	1122	1
4	<1,1,2,5,2>	1125	3
5	<1,1,2,7,8>	1127	2
6	<1,1,2,8,3>	1128	1
7	<1,1,2,9,7>	1129	1129
8	<1,1,3,1,2>	1131	2
9	<1,1,3,4,14>	1134	3
10	<1,1,3,5,8>	1135	1
11	<1,1,3,6,7>	1136	1
12	<1,1,3,7,2>	1137	1
13	<1,1,3,9,8>	1139	2
14	<1,1,4,1,25>	1141	1141
15	<1,1,4,2,1>	1142	1
16	<1,1,4,3,4>	1143	1
17	<1,1,4,4,2>	1144	1
18	<1,1,4,5,9>	1145	1
19	<1,1,4,6,8>	1146	1
20	<1,1,4,7,12>	1147	1
21	<1,1,7,8,7>	1178	31

(d)

Block 1	Offset
#tuples = 5, #bits = 4, counter=0	0
<1,1,1,1,3>	
1010,0001	31
0011,0010	
0001,----	
9	34
15	
2	
8	
3	

Block 2	Offset
#tuples=6, #bits=2, counter=0	0
<1,1,2,9,7>	
10,11,01,01	31
01, 10,--,--	
2	33
14	
8	
7	
2	
8	

Block 3	Offset
#tuples=7, #bits=5, counter=6	0
<1,1,4,1,25>	
11111,---	31
1	32
4	
2	
9	
8	
12	
7	

XTDC - Operations

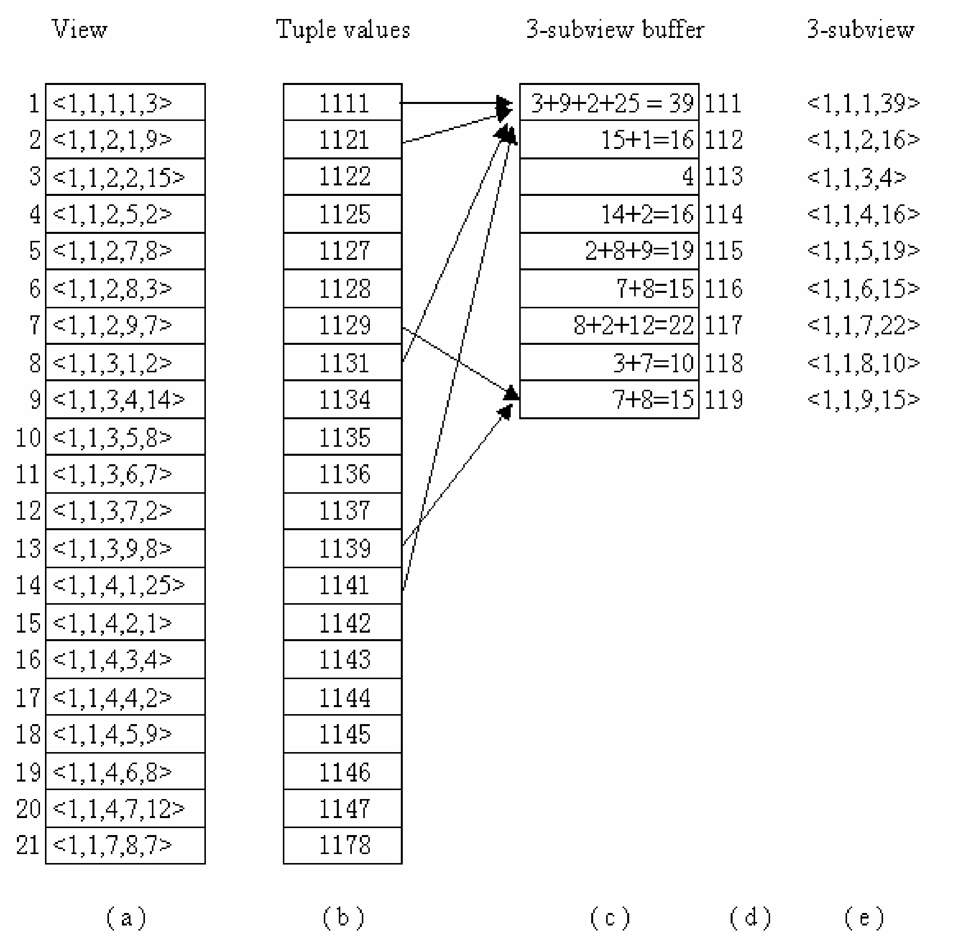
- Indexing
  - First tuple is in Block-header
  - B-tree
- Query
  - Locate the block
  - Compute the difference (t) to first tuple
  - Go through the different segment to accumulate the difference, until reach the difference (t), if exists.
  - Get measure data
- Update
  - Need some works
  - Not often in Date Warehouse application

XTDC - Operation …

- Subview generation
  - Compute tuple value from parent view

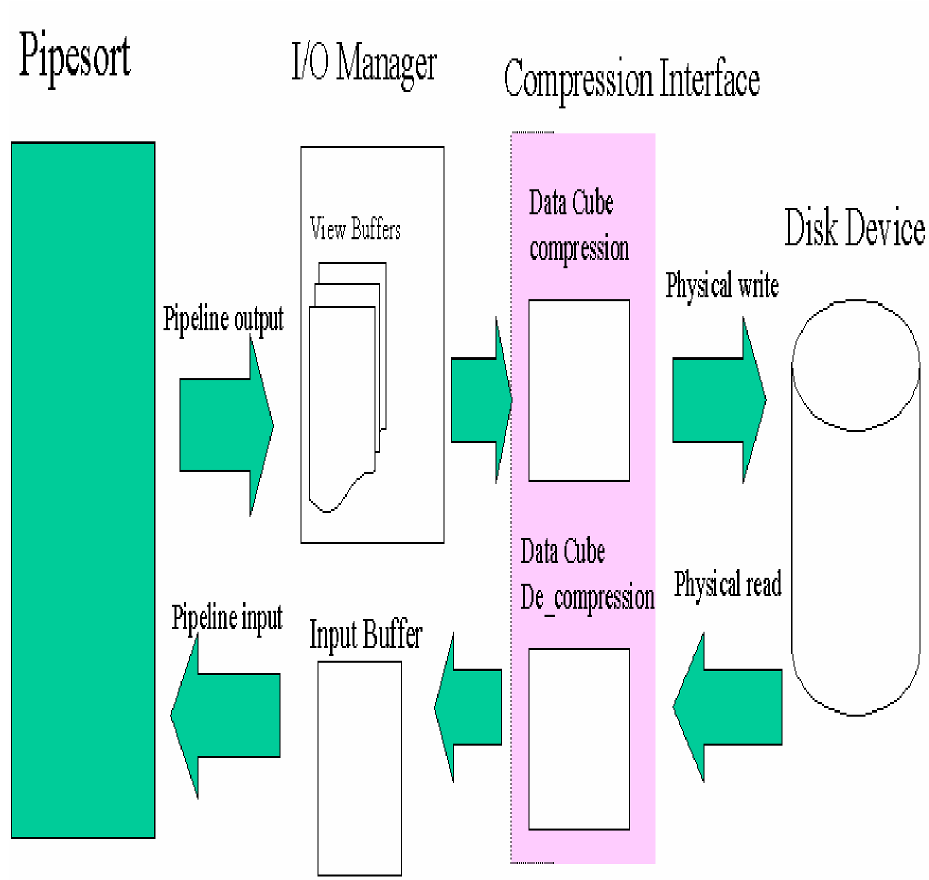
$$\varphi' = \left(\varphi \ div \prod_{l=k}^{n} |A_l|\right) \times \prod_{l=k+1}^{n} |A_l| + \varphi \ mod \prod_{j=k+1}^{n} |A_j|$$

  - Example:
    - 3-subview
  - Processes
    - Create a buffer
    - Go thru the parent, add the measure by index to construct subview
    - Create blocks



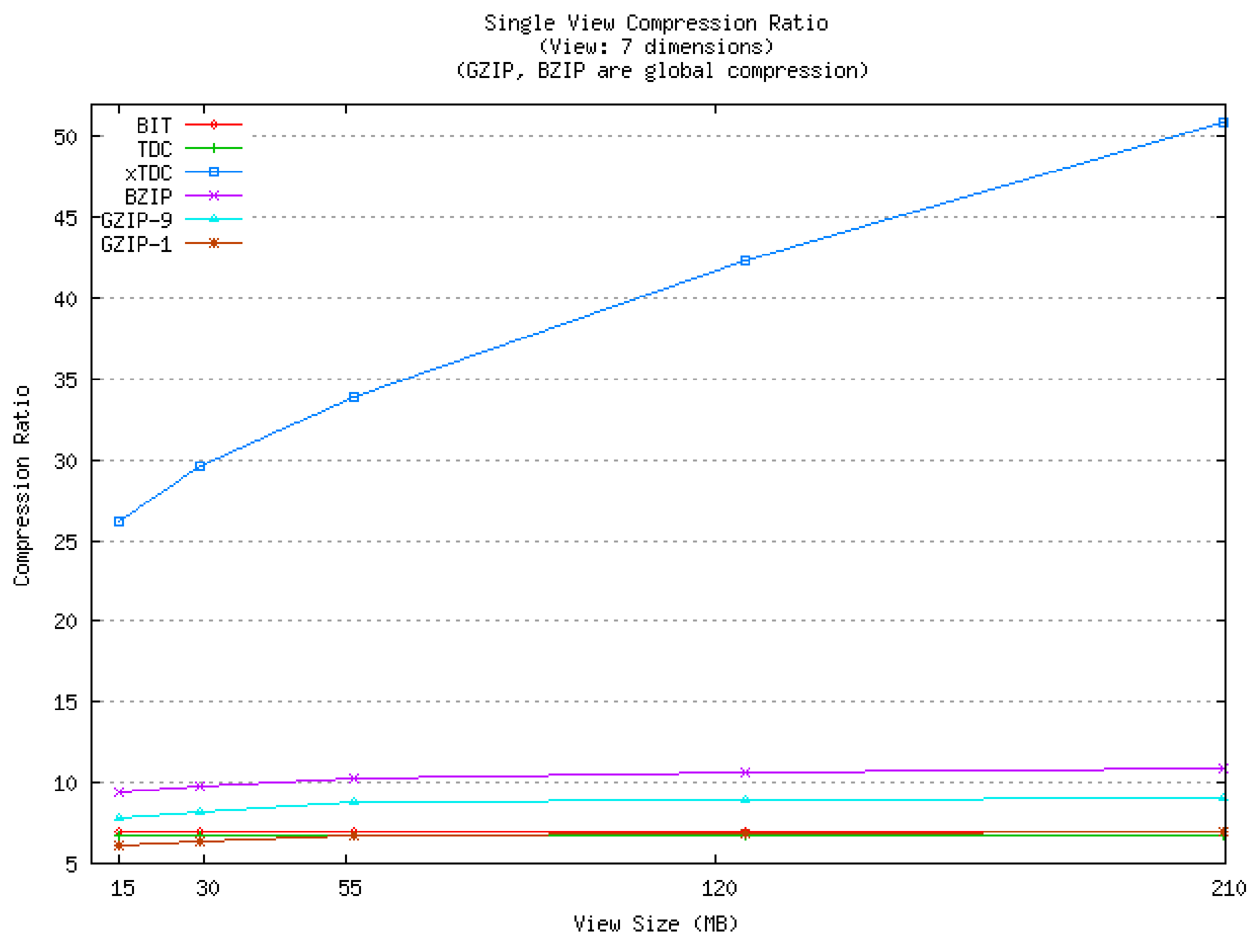
Data Cube Compression - Integration

- Environment
  - HPCVL Linux Cluster
  - MPI
- PANDA I/O Manager
  - Write – compress
  - Read – decompress



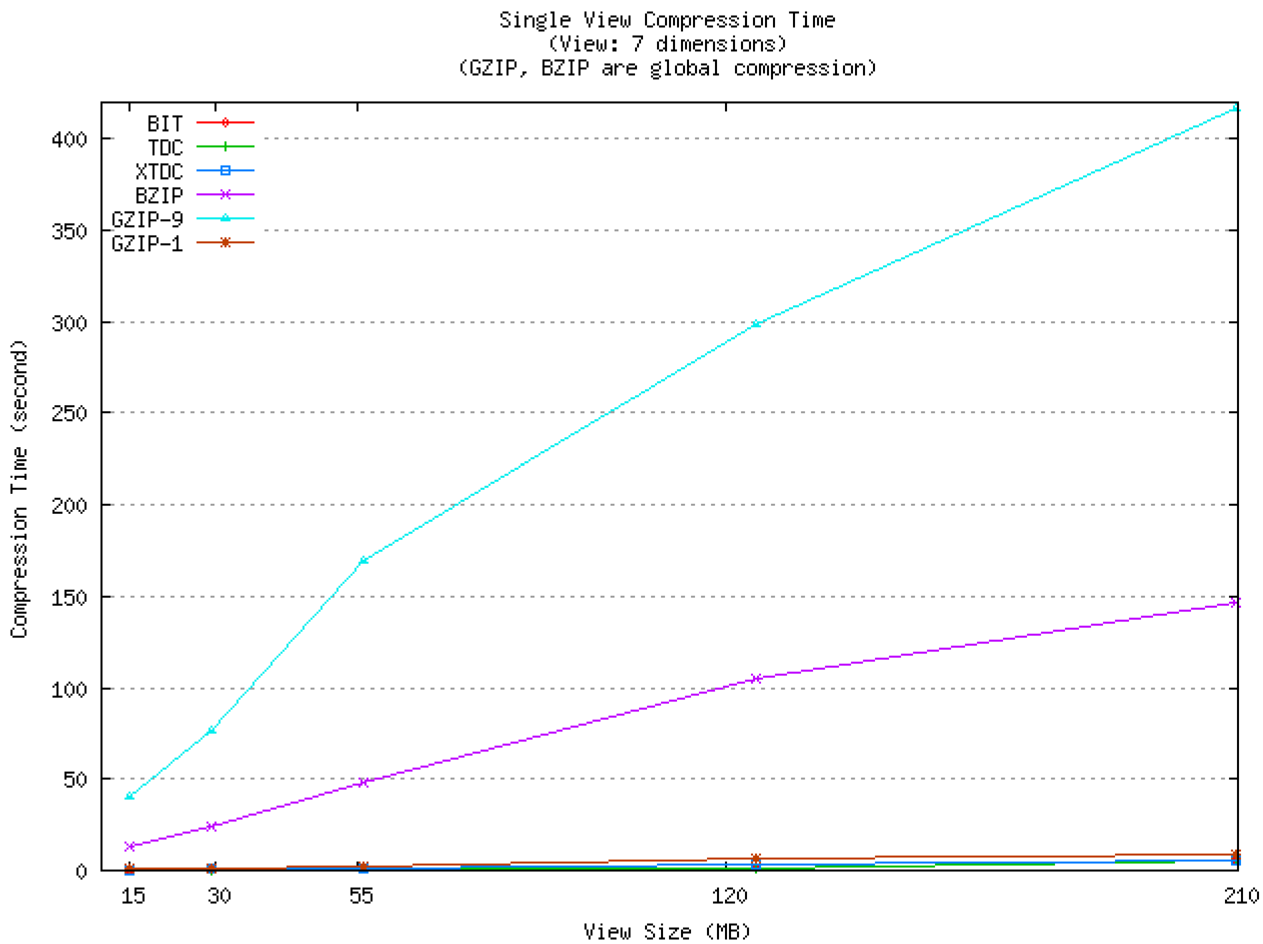
Evaluation – Single View

- Single view – compression ratio



Evaluation – Single View …

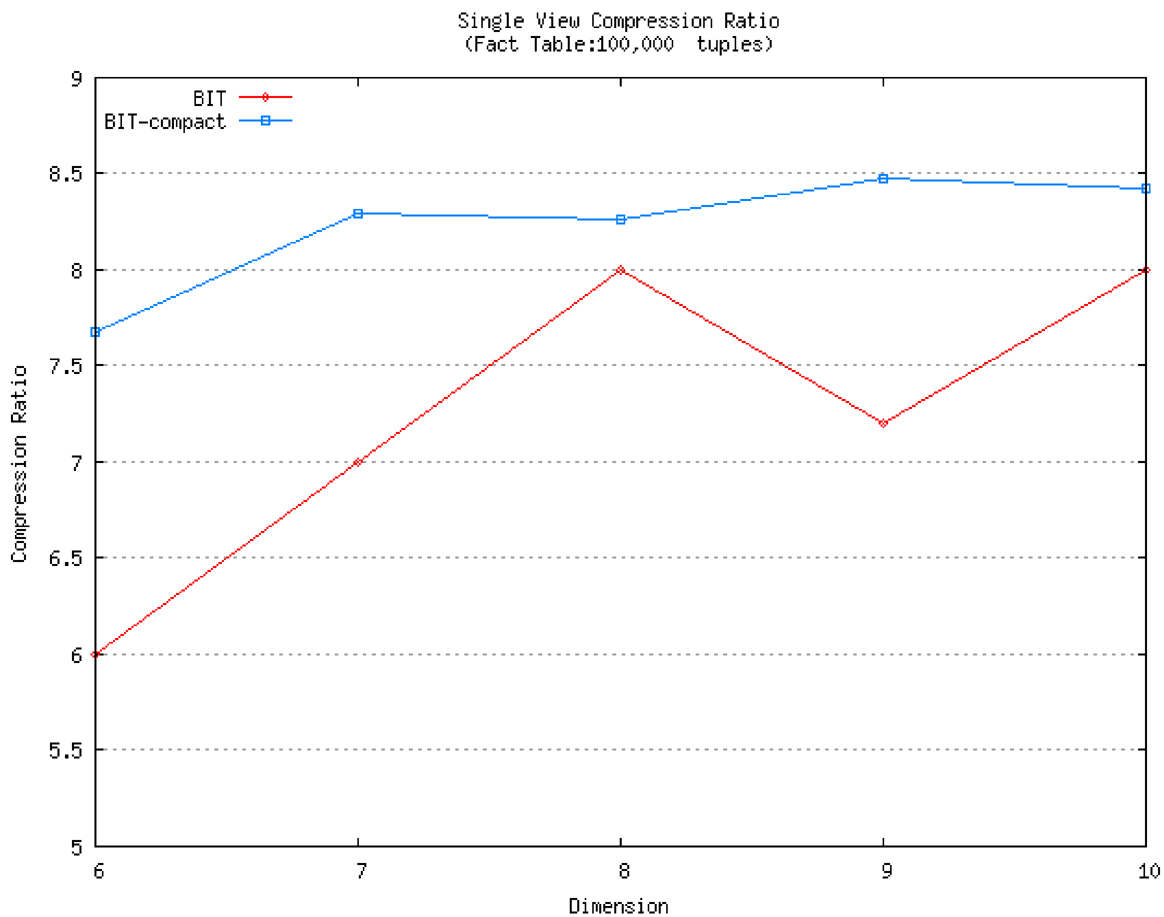
- Single view – compression time



Evaluation – Single View…

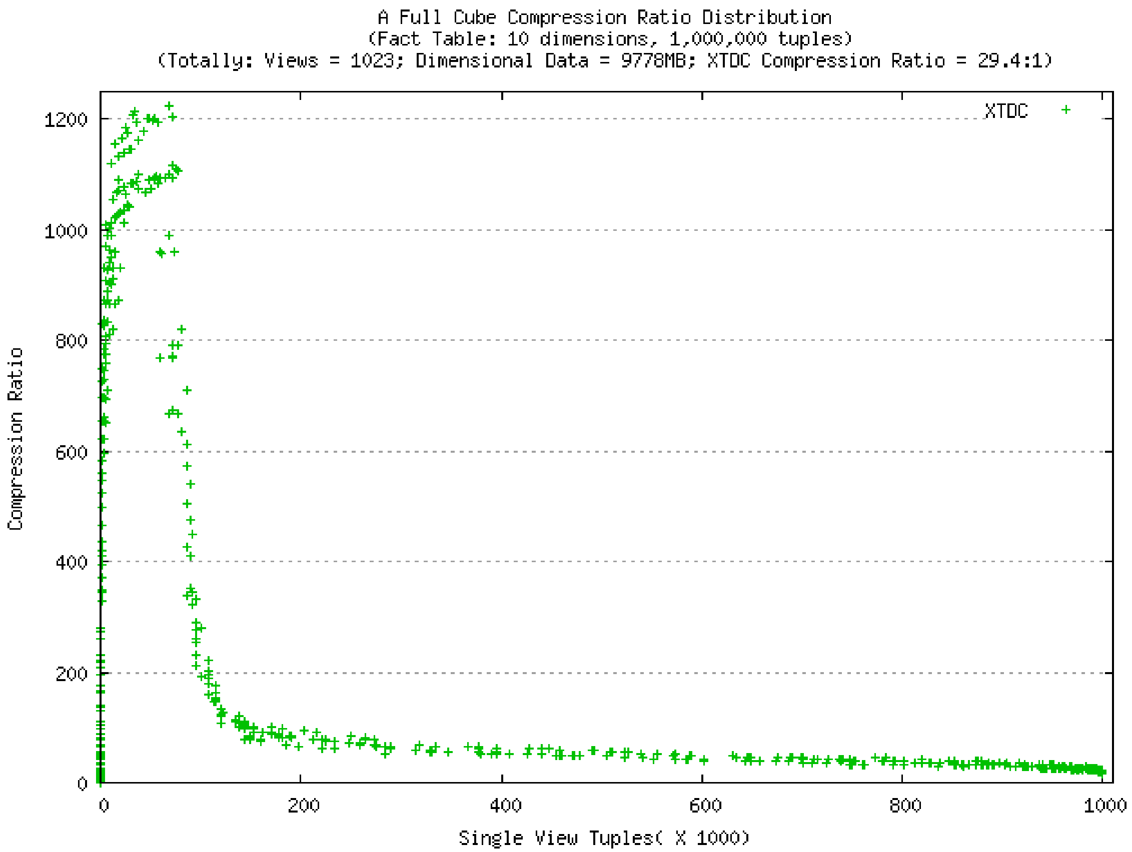
- Single view compression with Bit-compact

d	bits	gap
6	25	7
7	27	5
8	31	1
9	34	6
10	39	1



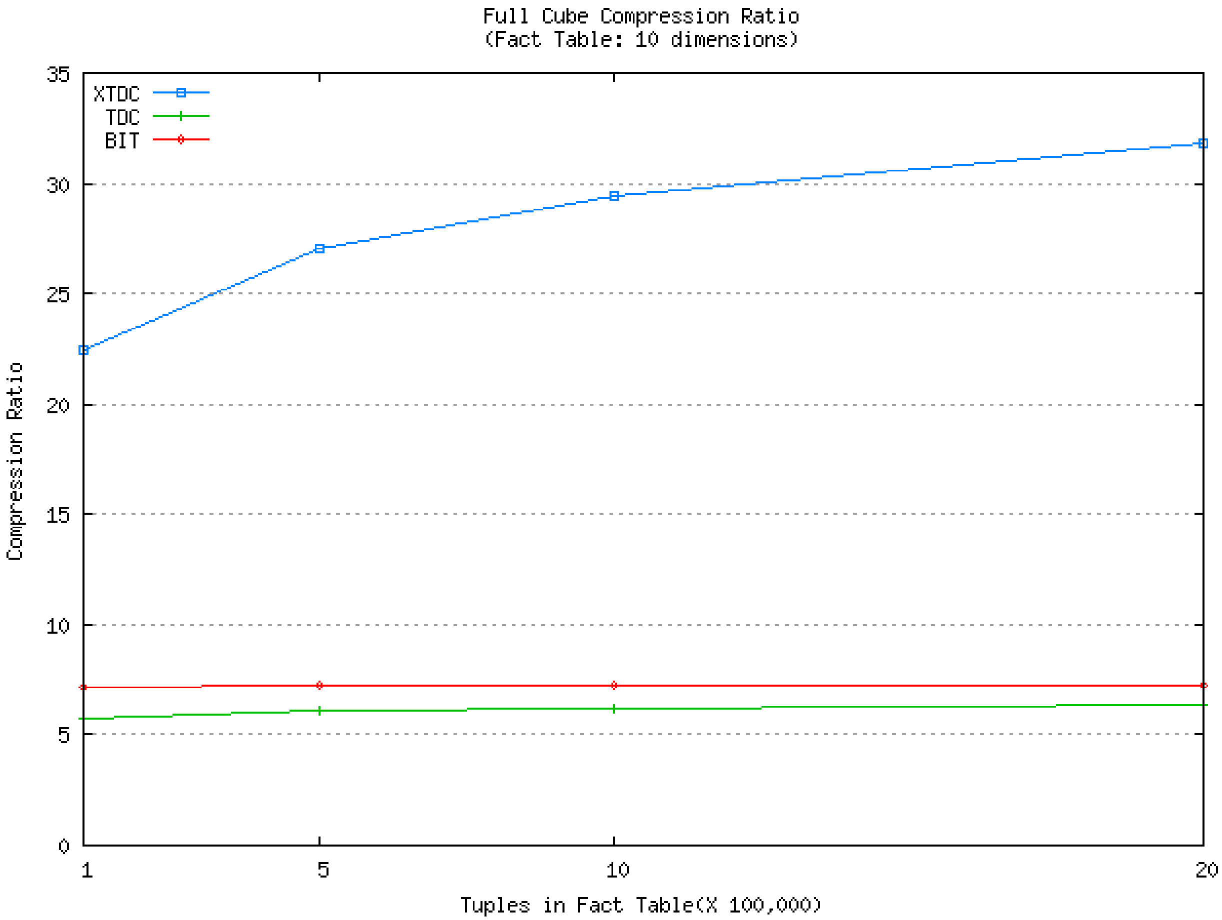
Evaluation – Full Cube

- Full Cube Compression - Distribution
  - 10-dimension
  - 1023 views
  - 1M tuples
  - 9778MB
  - 29.4:1



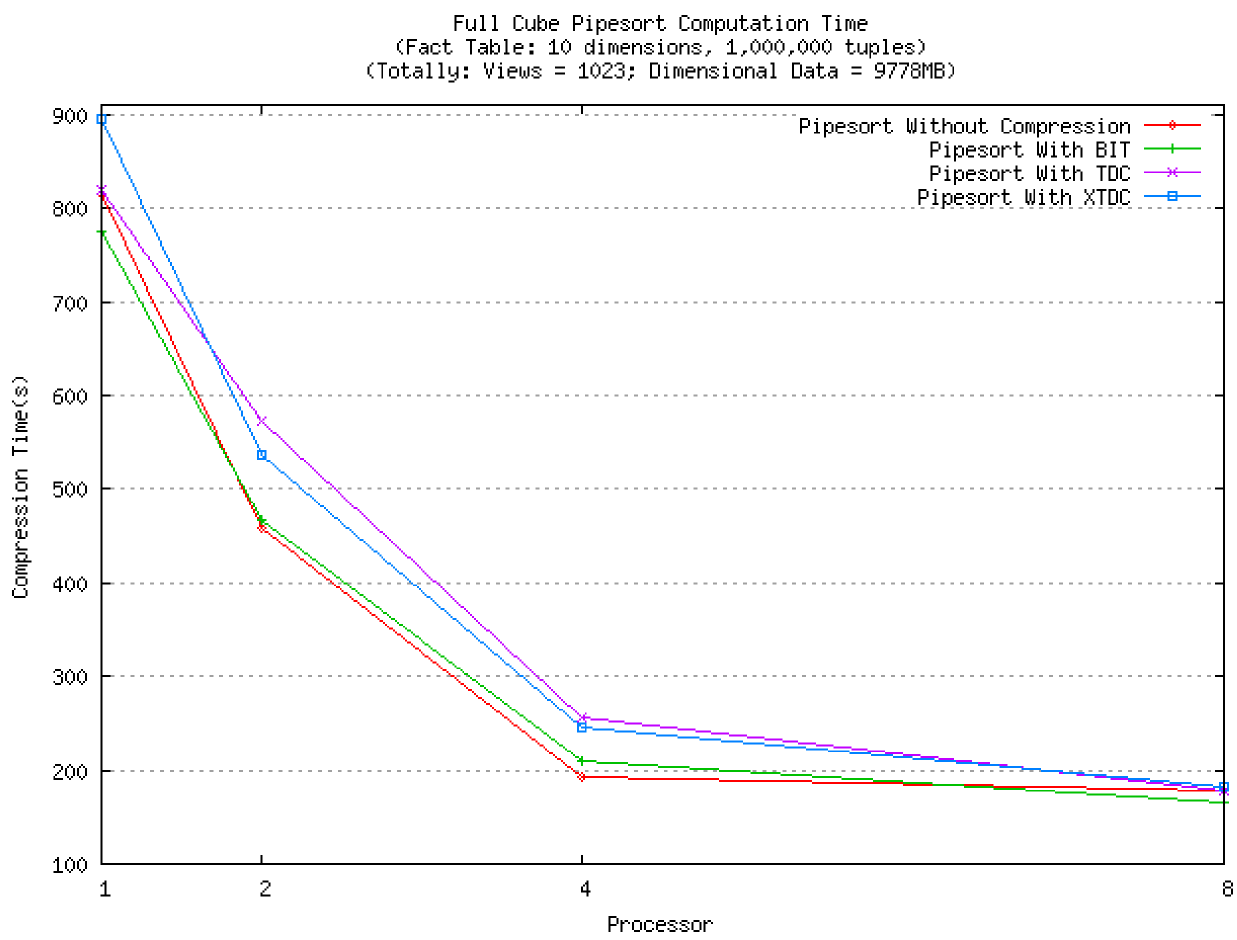
Evaluation – Full Cube …

- Full Cube Compression - Comparison



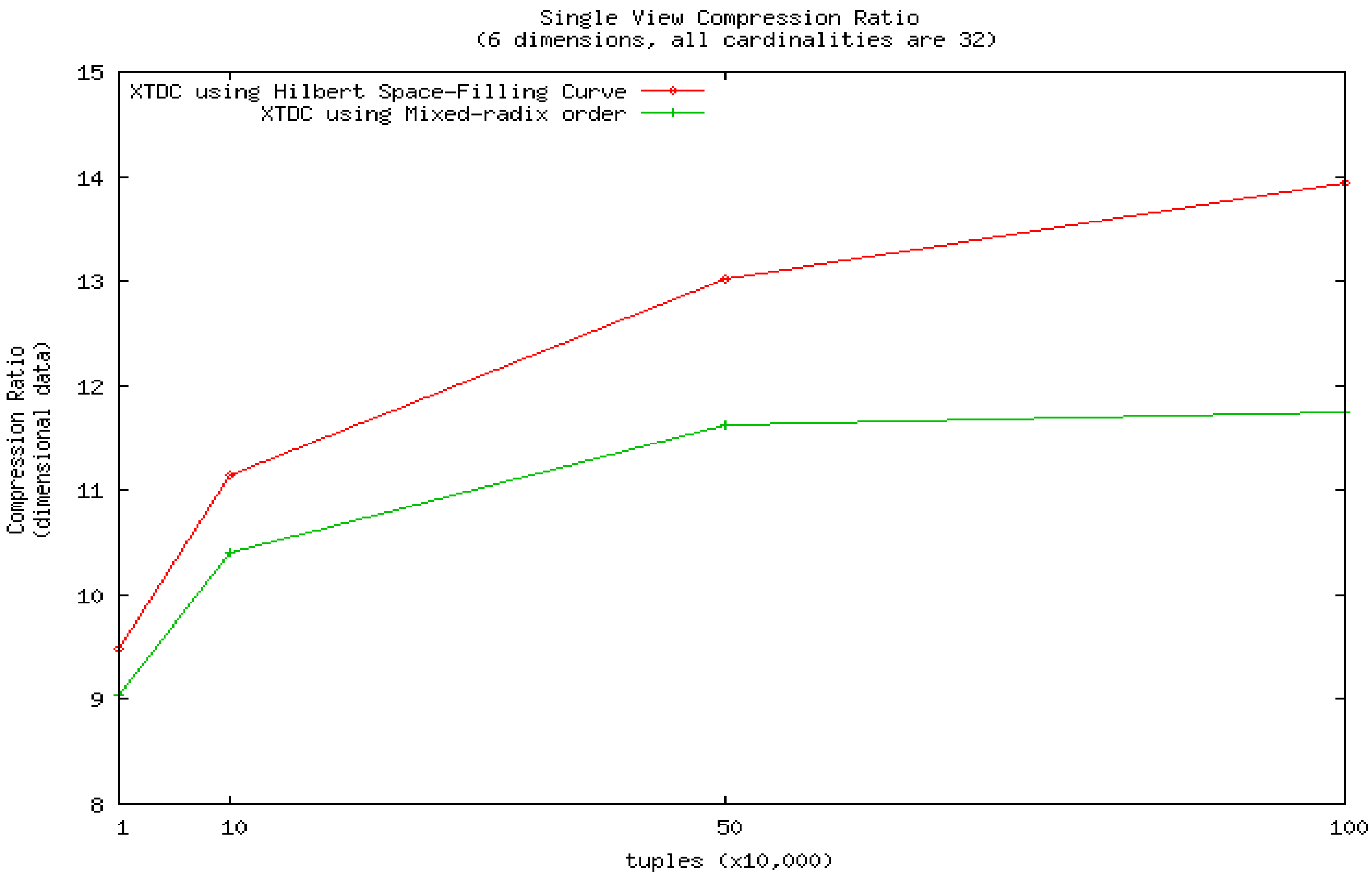
Evaluation – Full Cube …

- Full Cube Compression - speedup



Evaluation – Hilbert Order

- Single views compression



Conclusion

- Dynamic Block-oriented
- Tuple differential coding
- Bit-wise compression
- Related algorithms
  - Tuple minus, Tuple add, Point query, Subview generation
- High compression ratio
  - For Full Cube: 29.4:1 (9778MB to 333MB 96.6%)
  - Single View: 29.5:1
- Speed
  - ‘For free’
  - Well suited in parallel OLAP computing system
- Hilbert ordering is well suited to XTDC

Future Work

- Computation on compressed data
  - Conduct sub-views from compressed view
  - Reduce de/compression
- Using Hilbert Space Filling Curve to XTDC

Reference

- [1] Todd Eavis Parallel OLAP computing, 2004, Doctor Thesis, Dalhousie University
- [2] W.Ng, C.V.Ravishankar Block-Oriented Compression Techniques for Large Statistical Database, 1997
- [3] Ziv J., Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.
- [4] G. Ray, J. Haritsa, and S. Seshadri. Database compression: A performance enhancement tool. In Proc. COMAD, Pune, India, December 1995.
- [5] J. Goldstein, R. Ramakrishnan, and U. Shaft. Compressing relations and indexes. In Proc. IEEE Conf. on Data Engineering, Orlando, FL, USA, 1998.
- [6] Ralph Kimball, et al. The data warehouse lifecycle toolkit, John Wiley & Sons, Inc, ISBN 0-471-25547-5, 1998
- [7] Doug Moore http://www.caam.rice.edu/dougm/twiddle/hilbert
- [8] PANDA http://www.cs.dal.ca/panda
- [9] Julian Seward http://www.redhat.com/bzip2
- [10] Bo-Yong Liang, Compressing Data Cube in Parallel OLAP System, 2005, Master Thesis, Carleton University

Thank you