Deep Dive on BLU Acceleration in DB2 10.5, Super Analytics, Super Easy

DB2 TechTalk

May 30, 2013

Presented by:
Sam Lightstone, IBM Distinguished Engineer and DB2 Expert

Disclaimer

The information contained in this presentation is provided for informational purposes only.

While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided “as is”, without warranty of any kind, express or implied.

In addition, this information is based on IBM’s current product plans and strategy, which are subject to change by IBM without notice.

IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other documentation.

Nothing contained in this presentation is intended to, or shall have the effect of:

- Creating any warranty or representation from IBM (or its affiliates or its or their suppliers and/or licensors); or
- Altering the terms and conditions of the applicable license agreement governing the use of IBM software.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Agenda

1. Introduction to BLU Acceleration in DB2 10.5 and it's business value
2. Usage and getting started
3. Load and compression
4. Access plans and workload management

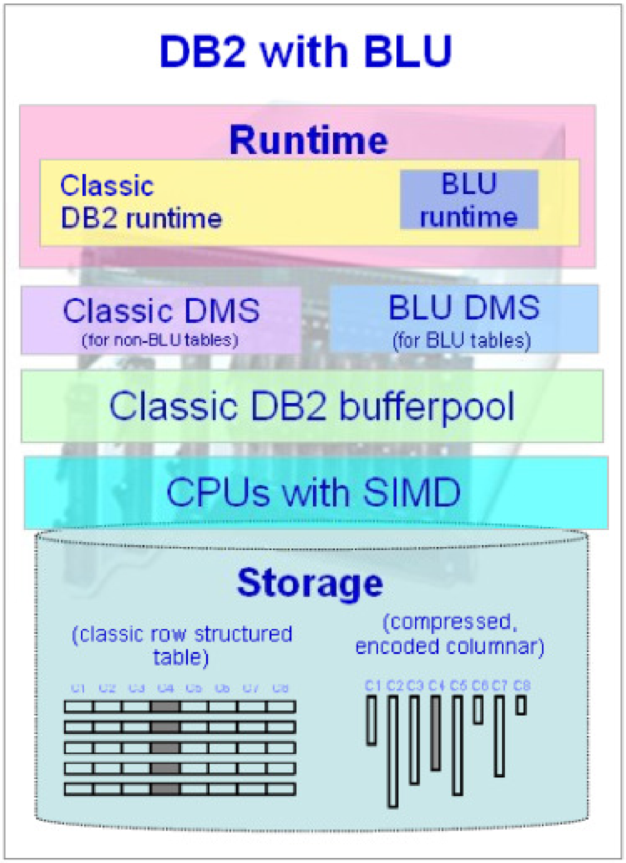
Part 1: The Value of BLU Acceleration

What is DB2 with BLU Acceleration?

- **New innovative technology for analytic queries**
  - Columnar storage
  - New run-time engine with vector (aka SIMD) processing, deep multi-core optimizations and cache-aware memory management
  - "Active compression" - unique encoding for further storage reduction beyond DB2 10 levels, and run-time processing without decompression

- **Value : Order-of-magnitude benefits in …**
  - **Performance**
  - **Storage savings**
  - **Simplicity !**

- **"Revolution by Evolution"**
  - Built directly into the DB2 kernel
  - BLU tables can coexists with traditional row tables, in same schema, tablespaces, bufferpools
  - Query any combination of BLU or row data
  - Memory-optimized (not "in-memory")



DB2 TechTalk
Need webcast troubleshooting help? Click attachments

IDUG Leading the DB2 User Community since 1988

Super fast, Super Easy – Create, Load, and Go

Database Design and Tuning

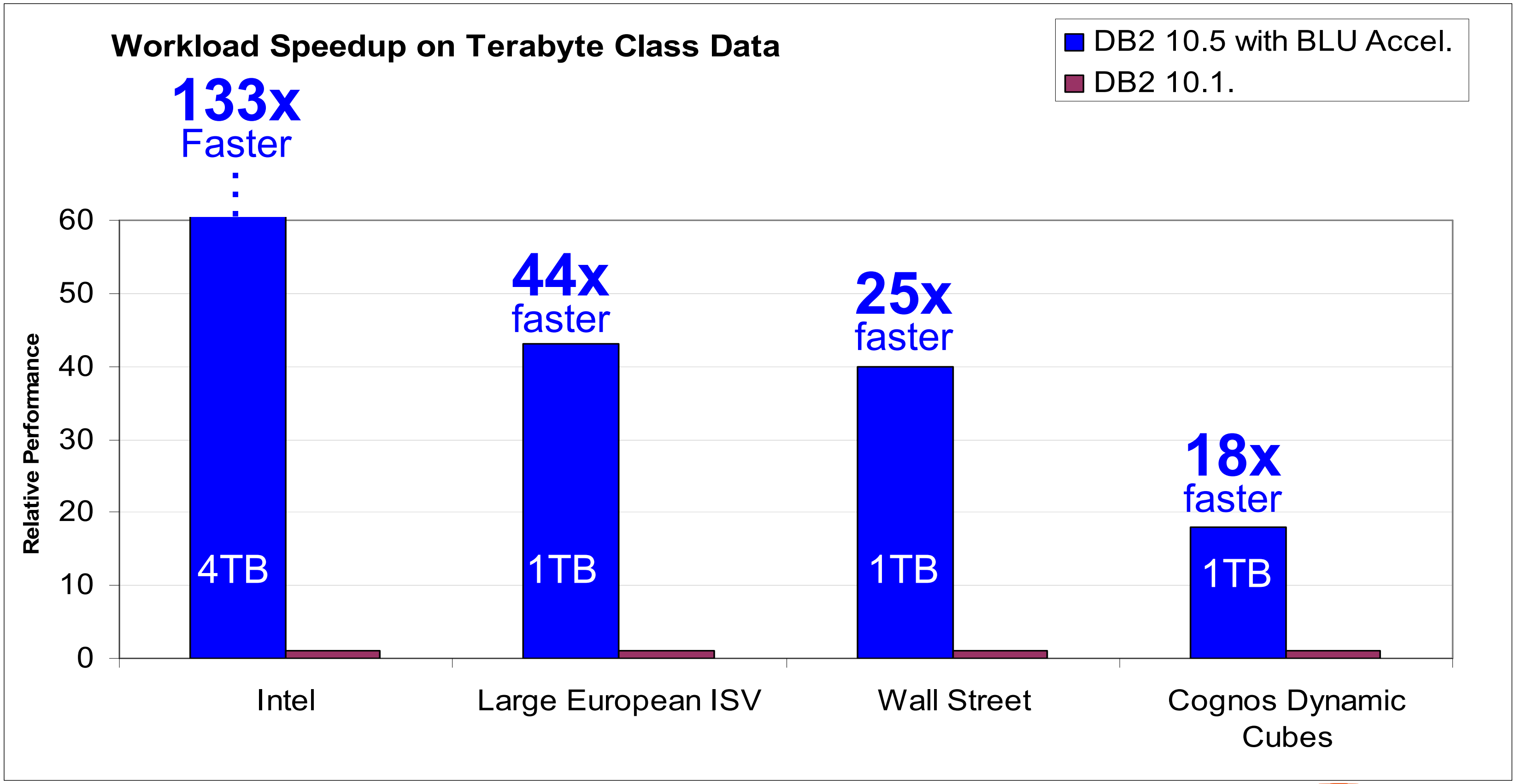
1. Decide on partition strategies
2. Select Compression Strategy
3. Create Table
4. Load data
5. Create Auxiliary Performance Structures
   - Materialized views
   - Create indexes
     - B+ indexes
     - Bitmap indexes
6. Tune memory
7. Tune I/O
8. Add Optimizer hints
9. Statistics collection

Repeat

DB2 with BLU Acceleration

1. Create Table
2. Load data

Terabyte Class Results, March-April 2013



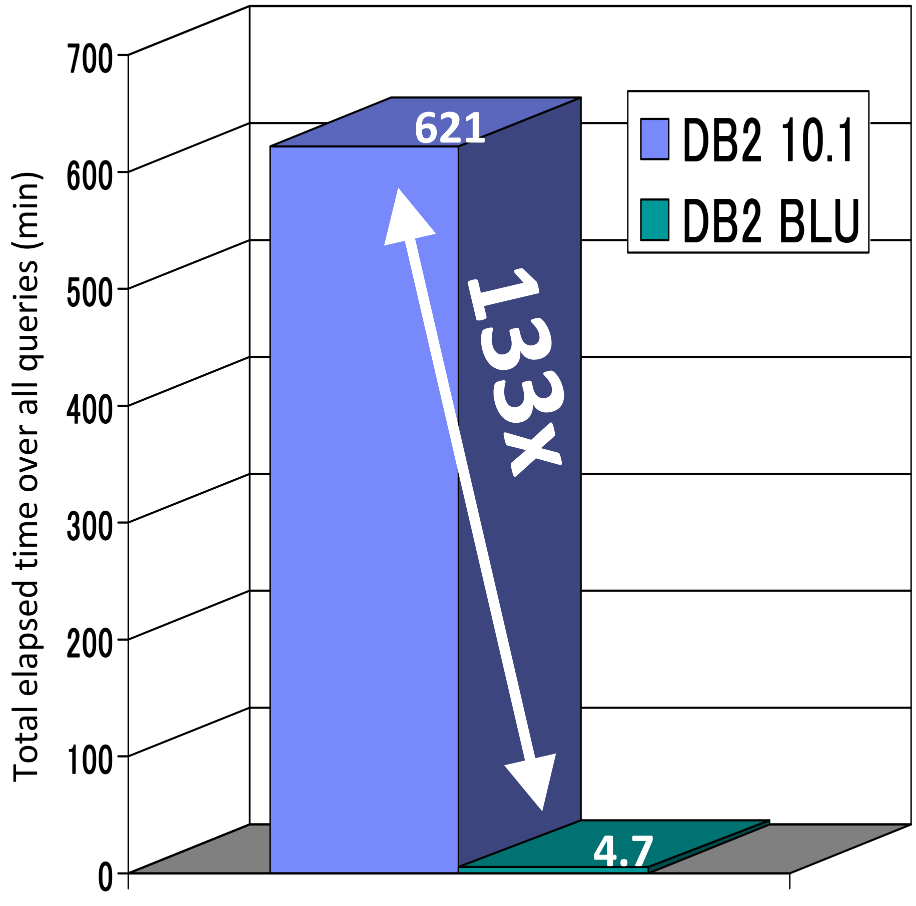
BNSF RAILWAY

"It was amazing to see the faster query times compared to the performance results with our row-organized tables. ***The performance of four of our queries improved by over 100-fold! The best outcome was a query that finished 137x faster by using BLU Acceleration."***

- Kent Collins, Database Solutions Architect, BNSF Railway

Recent Internal Test

- **POPS (Proof of Performance and Scalability)**
  - Derived from Redbrick performance test
  - Classic sales analytics
  - 5.5years of data (2000 days) for 63 stores
    - ~4TB of raw data
    - 2 fact tables
    - 5 dimension tables
  - Broad range of queries with varying selectivity / aggregation
- **Substantial Storage Savings with BLU Acceleration**
  - 2.5x less space than DB2 10.1
- **Massive Performance Gains**
  - **133x** speedup over DB2 10.1
  - Maximum query speed up over **900x**



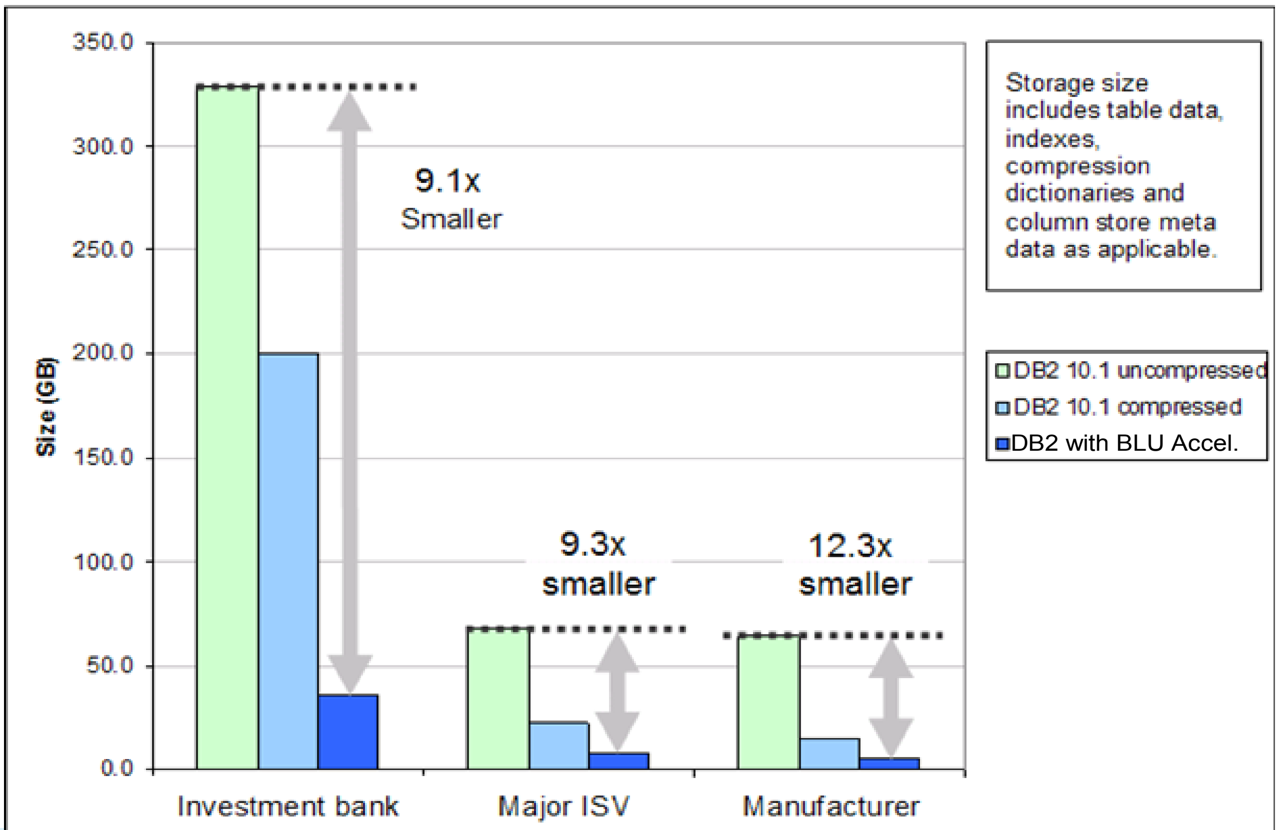
Intel® Xeon® Processor E5-4650
32 cores total (4 CPUs)
384 GB
DS5300 (2x16 disks)

Lab tests - YMMV

Need webcast troubleshooting help? Click attachments

Significant Storage Savings

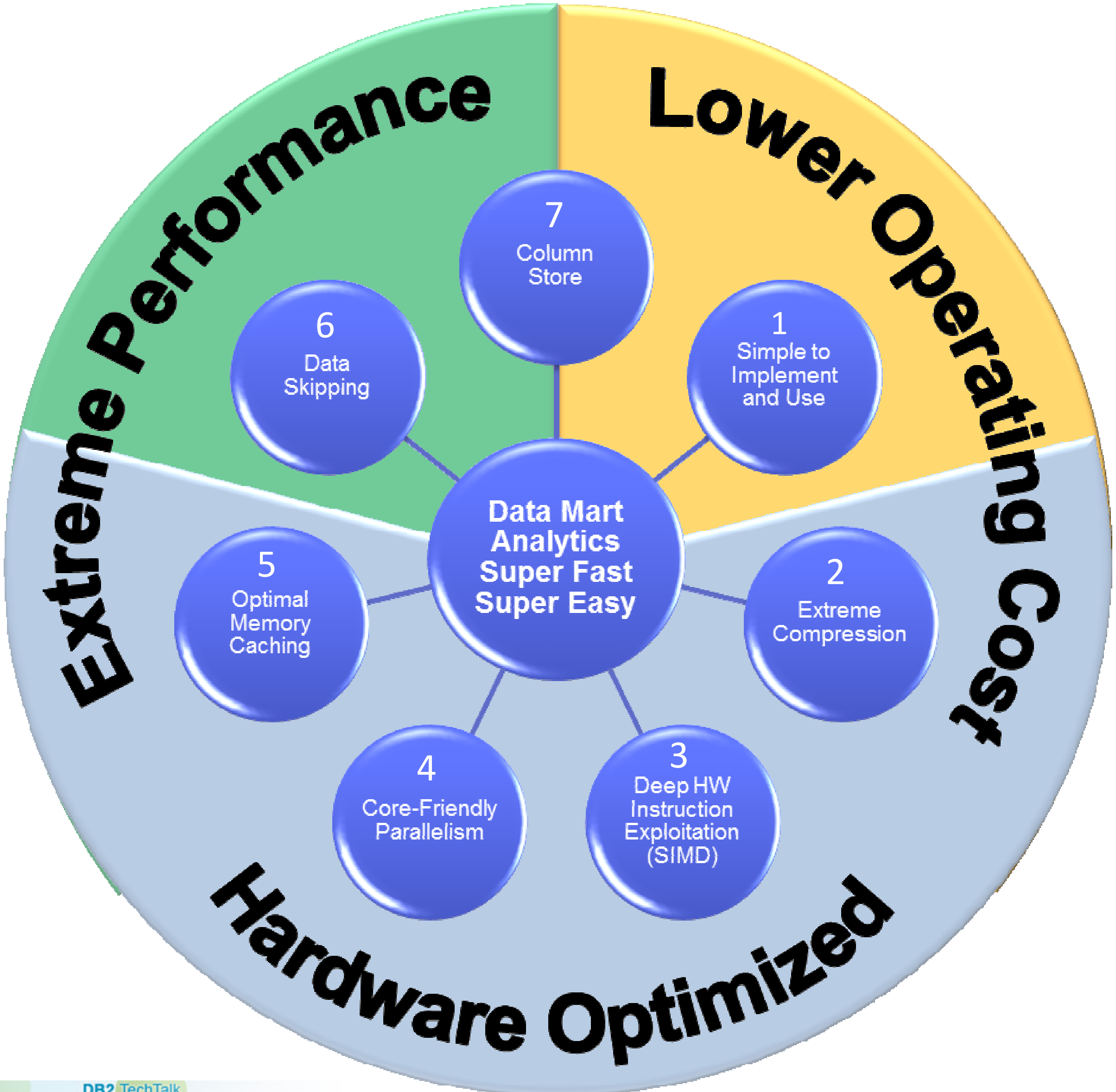
- ~2x-3x storage reduction vs DB2 10.1 adaptive compression (comparing all objects - tables, indexes, etc)
  - New advanced compression techniques
  - Fewer storage objects required



Lab tests - YMMV

Need webcast troubleshooting help? Click attachments

DB2 with BLU Acceleration: The 7 Big Ideas



Need webcast troubleshooting help? Click attachments

7 Big Ideas: 1 *Simple to Implement and Use*

- LOAD and then… run queries
  - – Significantly reduced or no need for …
    - Indexes
    - `REORG` (it's automated)
    - `RUNSTATS` (it's automated)
    - MDC or MQTs
    - Statistical views
    - Optimizer profiles/guidelines

- It is just DB2!
  - – Same SQL, language interfaces, administration
  - – Same DB2 process model, storage, bufferpools

Coca-Cola Bottling Co. CONSOLIDATED

"The BLU Acceleration technology has some obvious benefits: … But it's when I think about **all the things I don't have to do with BLU**, it made me appreciate the technology even more: **no tuning, no partitioning, no indexes, no aggregates**."

-Andrew Juarez, Lead SAP Basis and DBA

7 Big Ideas: 1 *Simple to Implement and Use*

- One setting optimizes the system for BLU Acceleration
  - Set `DB2_WORKLOAD=ANALYTICS`
  - Informs DB2 that the database will be used for analytic workloads
- Automatically configures DB2 for optimal analytics performance
  - Makes column-organized tables the default table type
  - Sets up default page (32KB) and extent size (4) appropriate for analytics
  - Enables automatic workload concurrency management
  - Enables automatic space reclaim
  - Memory for caching, sorting and hashing (*bufferpool, sortheap*), utilities (*utility heap*) are automatically initialized based on the server size and available RAM
- Simple Table Creation
  - If `DB2_WORKLOAD=ANALYTICS`, tables will be created column organized automatically
  - Data is always automatically compressed - no options
  - For mixed table types can define tables as `ORGANIZE BY COLUMN` or `ROW`
- Utility to convert tables from row-organized to column-organized
  - `db2convert` utility

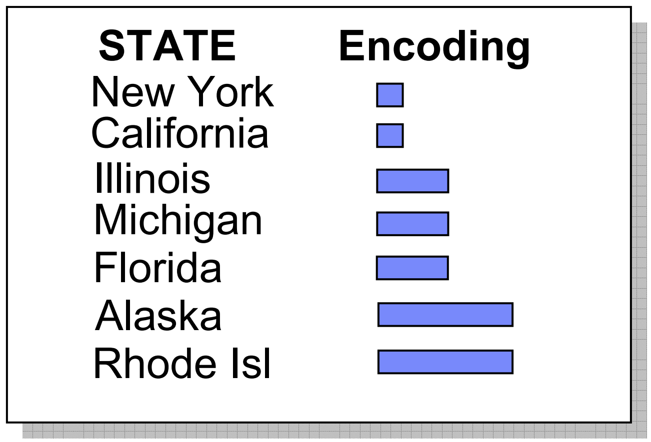
Need webcast troubleshooting help? Click attachments

© 2013 IBM Corporation

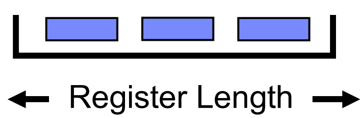
7 Big Ideas: 2 *Compute Friendly Encoding and Compression*

- Massive compression with approximate Huffman encoding
  - – The more frequent the value, the fewer bits it is encoded with
  - – E.g., there will typically be more sales records from states with higher populations
    - New York and California, may be encoded with only 1 or 2 bits
    - Alaska and Rhode Island may be encoded in 12 bits

Conceptual Compression Dictionary



- Register-friendly encoding optimizes CPU & memory efficiency
  - – Encoded values packed together to match the register width of the CPU
  - – Fewer I/Os, better memory utilization, fewer CPU cycles to process



7 Big Ideas: 2 *Data Remains Compressed During Evaluation*

- **Encoded values do not need to be decompressed during evaluation**
  – predicates (=, <>,<, >, >=, <=, between, etc), joins, aggregations, ….
  work directly on encoded values

```
SELECT COUNT(*) FROM T1 WHERE STATE = 'California'
```

STATE	Encoding
Michigan	
California	
New York	
California	
New York	
Illinois	
California	
Alaska	
Rhode Is	
California	

7 Big Ideas: 2 *Data Remains Compressed During Evaluation*

- **Encoded values do not need to be decompressed during evaluation**
  – predicates (=, <>,<, >, >=, <=, between, etc), joins, aggregations, ….
  work directly on encoded values

```
SELECT COUNT(*) FROM T1 WHERE STATE =
```

STATE **Encoding**

Michigan
California
New York
California
New York
Illinois
California
Alaska
Rhode Is
California

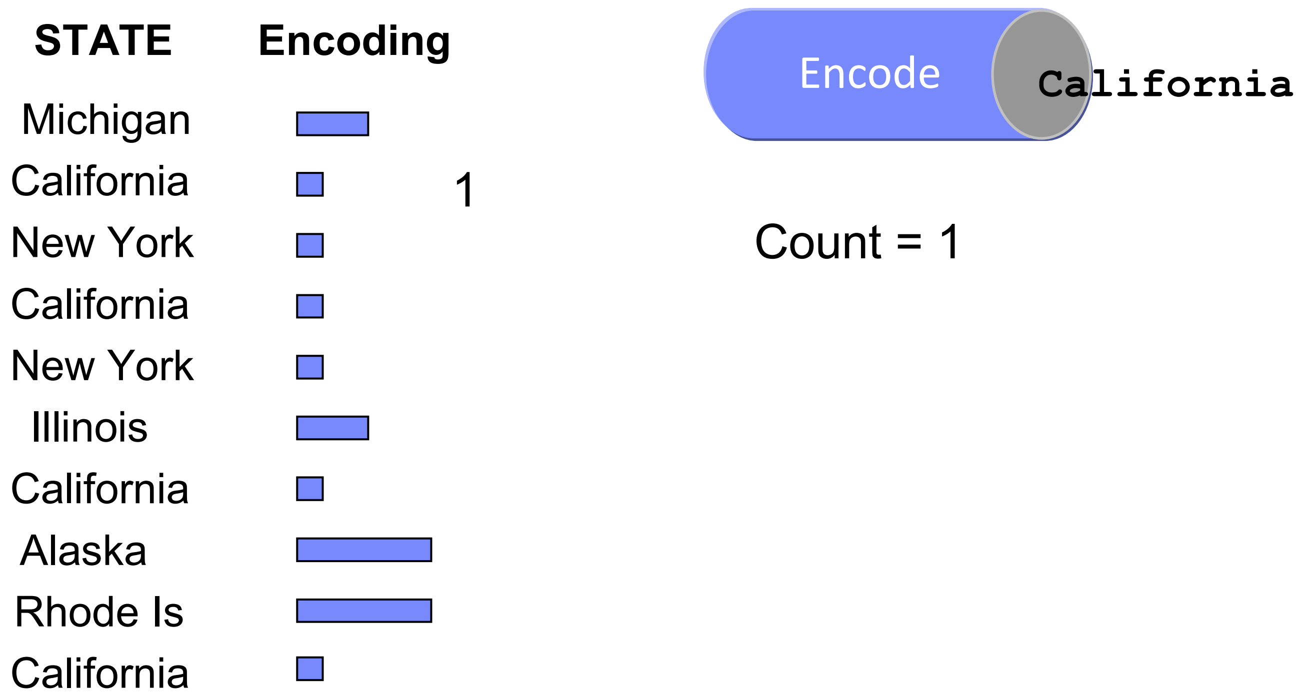
Encode California

Need webcast troubleshooting help? Click attachments

7 Big Ideas: 2 *Data Remains Compressed During Evaluation*

- **Encoded values do not need to be decompressed during evaluation**
  – predicates (=, <>,<, >, >=, <=, between, etc), joins, aggregations, …. work directly on encoded values

```
SELECT COUNT(*) FROM T1 WHERE STATE =
```



7 Big Ideas: 2 *Data Remains Compressed During Evaluation*

- **Encoded values do not need to be decompressed during evaluation**
  - predicates (=, <>,<, >, >=, <=, between, etc), joins, aggregations, …. work directly on encoded values

```
SELECT COUNT(*) FROM T1 WHERE STATE =
```

STATE	Encoding	
Michigan		
California		1
New York		
California		2
New York		
Illinois		
California		3
Alaska		
Rhode Is		
California		4

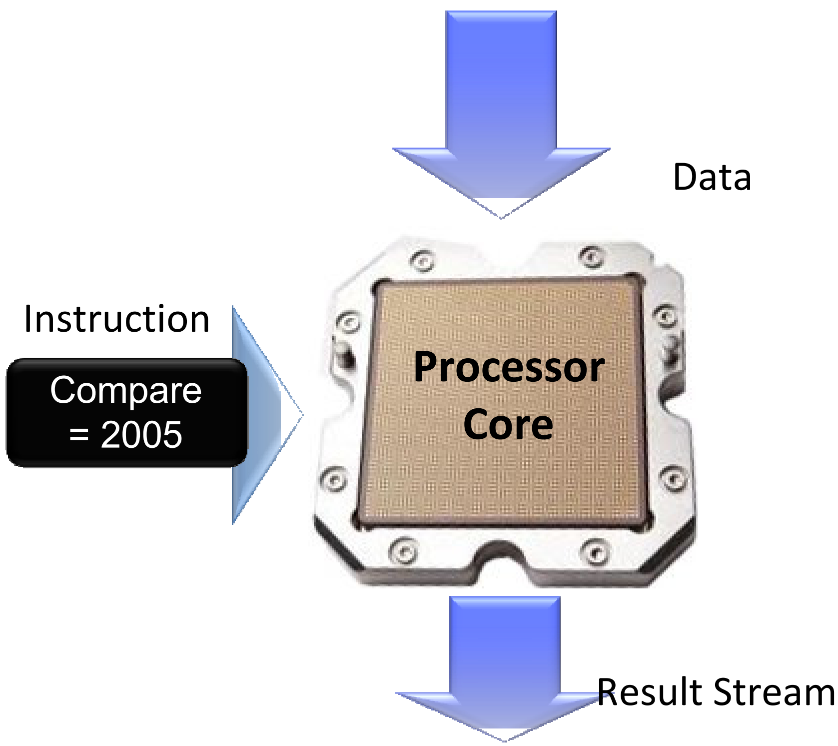
Encode California

Count = 2 .. 3 .. 4

Final Count = 4

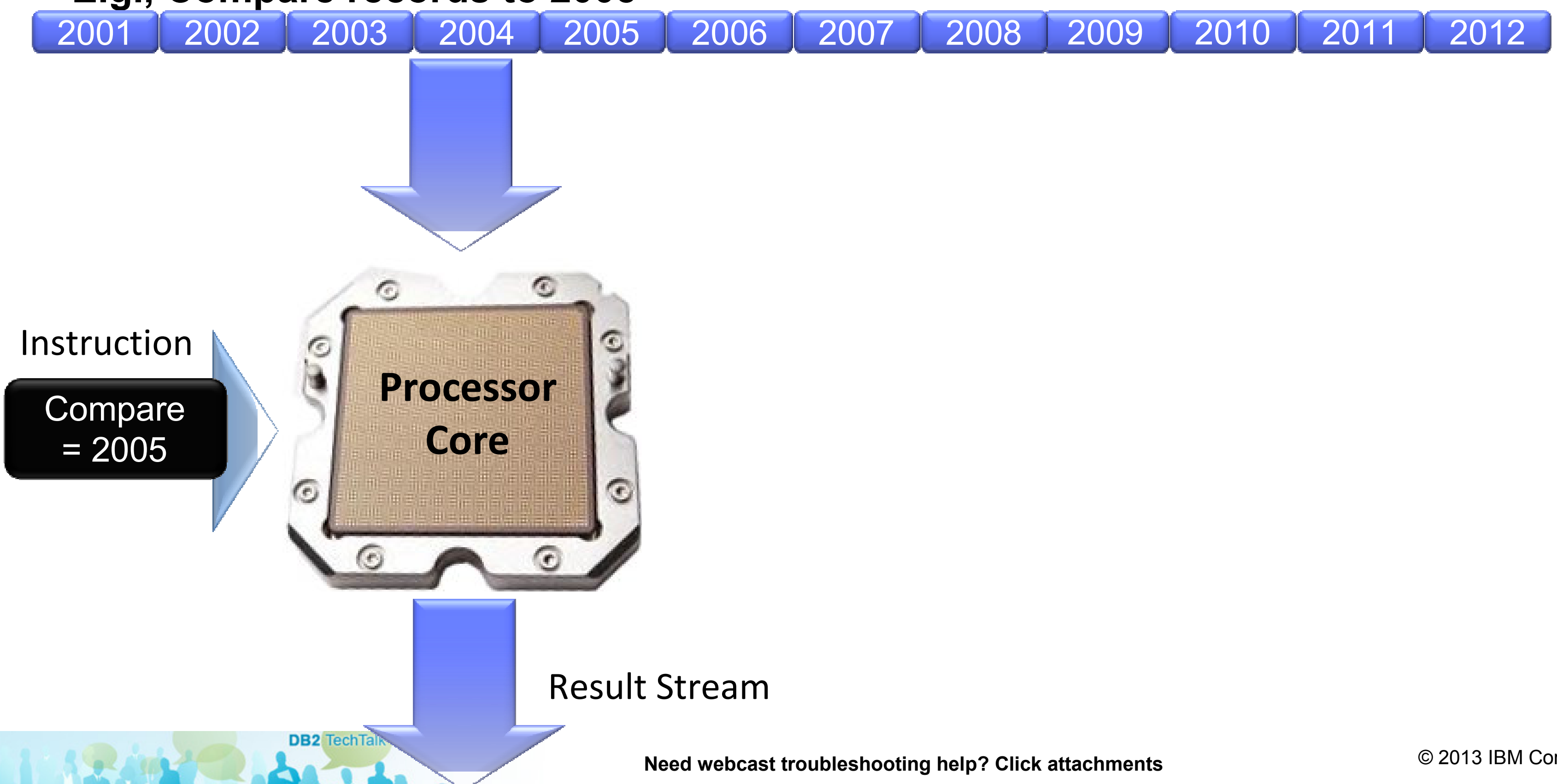
7 Big Ideas: 3 *Multiply the Power of the CPU*

- **Without `SIMD` processing the CPU will apply each instruction to each**
- Performance increase with Single Instruction Multiple Data (`SIMD`)
- Using hardware instructions, DB2 with BLU Acceleration can apply a single instruction to many data elements simultaneously
  - Predicate evaluation, joins, grouping, arithmetic



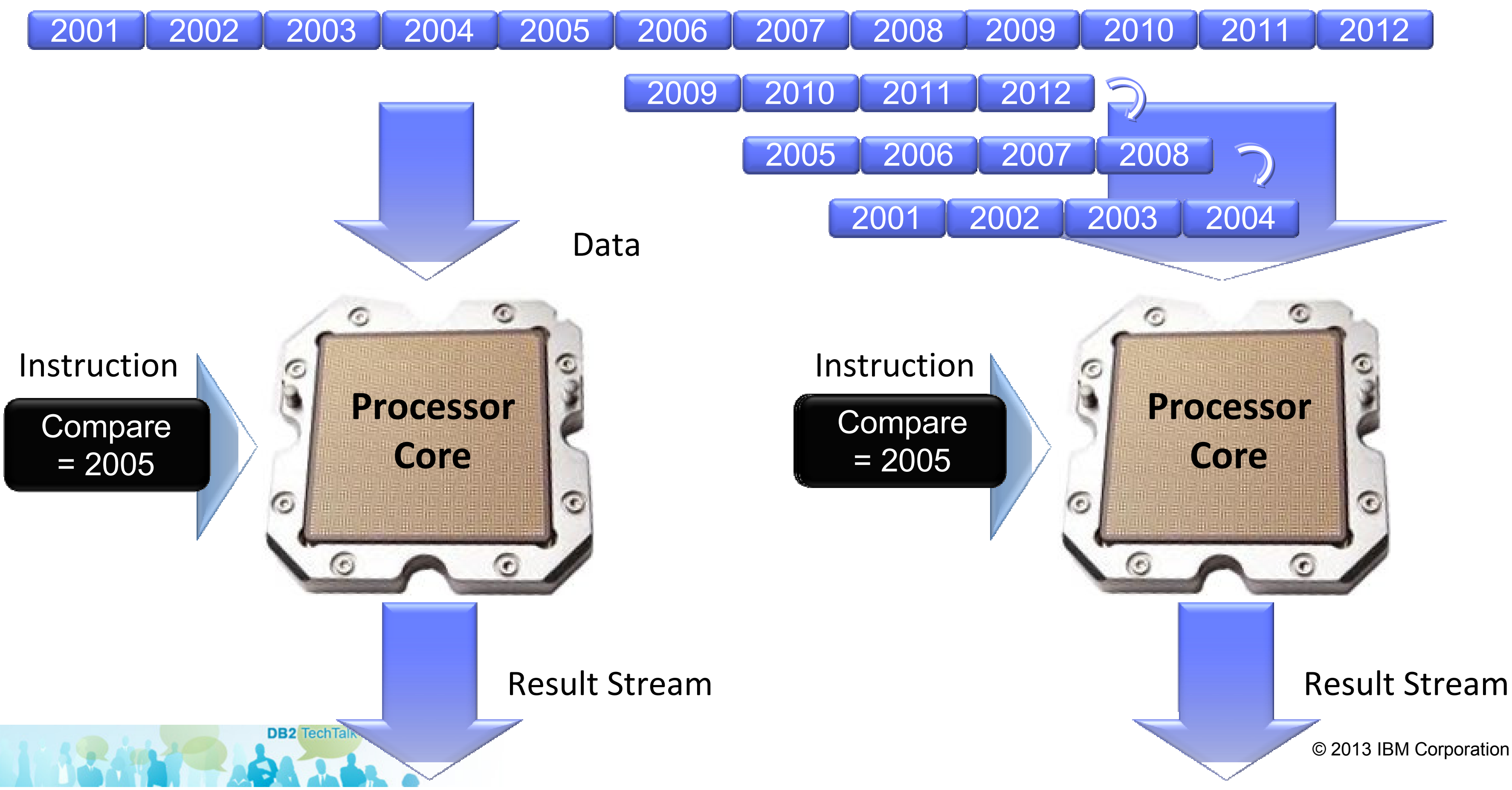
7 Big Ideas: 3 *Multiply the Power of the CPU*

- Performance increase with Single Instruction Multiple Data (`SIMD`)
- Using hardware instructions, DB2 with BLU Acceleration can apply a single instruction to many data elements simultaneously
  - Predicate evaluation, joins, grouping, arithmetic
  - **E.g., Compare records to 2005**



7 Big Ideas: 3 *Multiply the Power of the CPU*

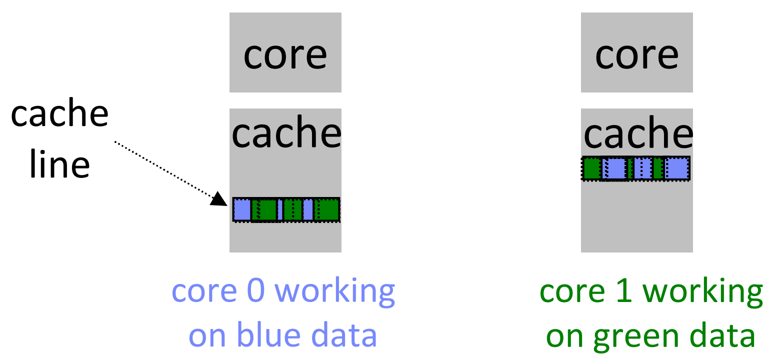
- Performance increase with Single Instruction Multiple Data (`SIMD`)
- Using hardware instructions, DB2 with BLU Acceleration can apply a single instruction to many data elements simultaneously
  - Predicate evaluation, joins, grouping, arithmetic



7 Big Ideas: 4 *Core-Friendly Parallelism*

- **BLU queries automatically parallelized across cores, and, achieve excellent multi-core scalability via …**
  - careful data placement and alignment
  - careful attention to physical attributes of the server
  - and other factors, designed to …

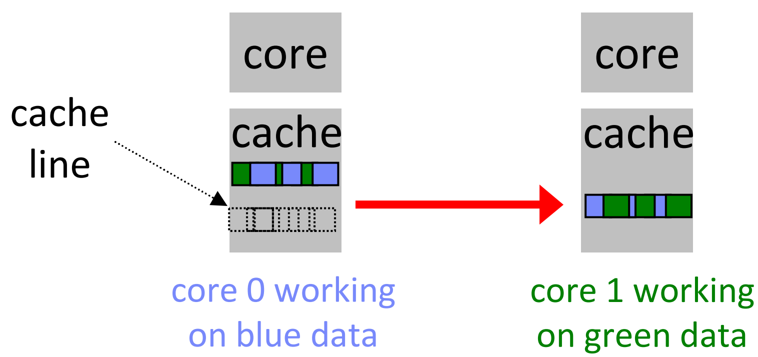
… maximize CPU cache hit rate & cacheline efficiency



7 Big Ideas: 4 *Core-Friendly Parallelism*

- **BLU queries automatically parallelized across cores, and, achieve excellent multi-core scalability via …**
  – careful data placement and alignment
  – careful attention to physical attributes of the server
  – and other factors, designed to …

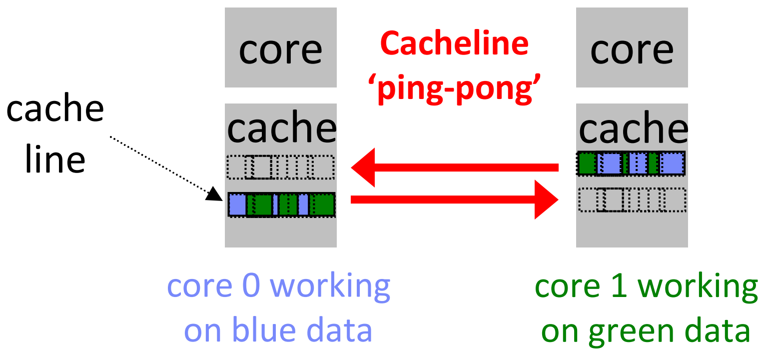
… maximize CPU cache hit rate & cacheline efficiency



7 Big Ideas: 4 *Core-Friendly Parallelism*

- **BLU queries automatically parallelized across cores, and, achieve excellent multi-core scalability via …**
  – careful data placement and alignment
  – careful attention to physical attributes of the server
  – and other factors, designed to …

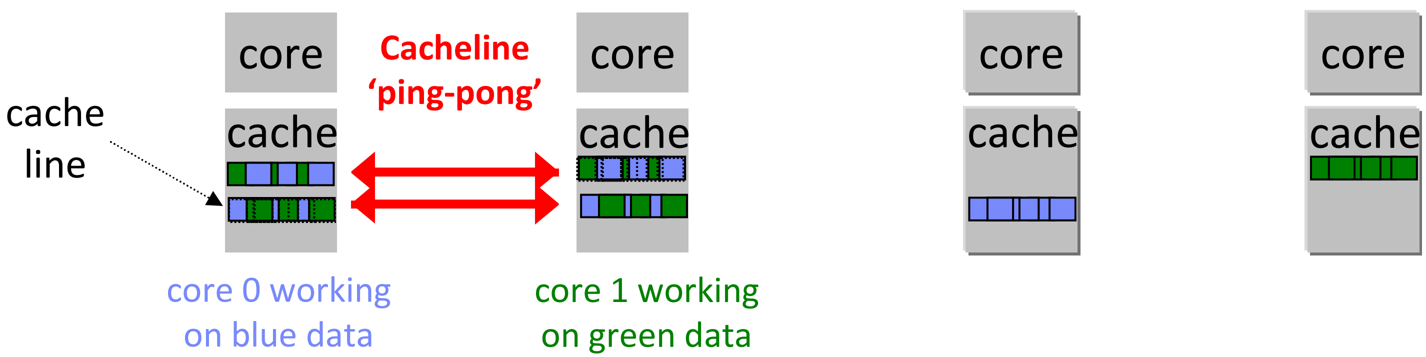
… maximize CPU cache hit rate & cacheline efficiency



7 Big Ideas: 4 *Core-Friendly Parallelism*

- **BLU queries automatically parallelized across cores, and, achieve excellent multi-core scalability via …**
  - careful data placement and alignment
  - careful attention to physical attributes of the server
  - and other factors, designed to …

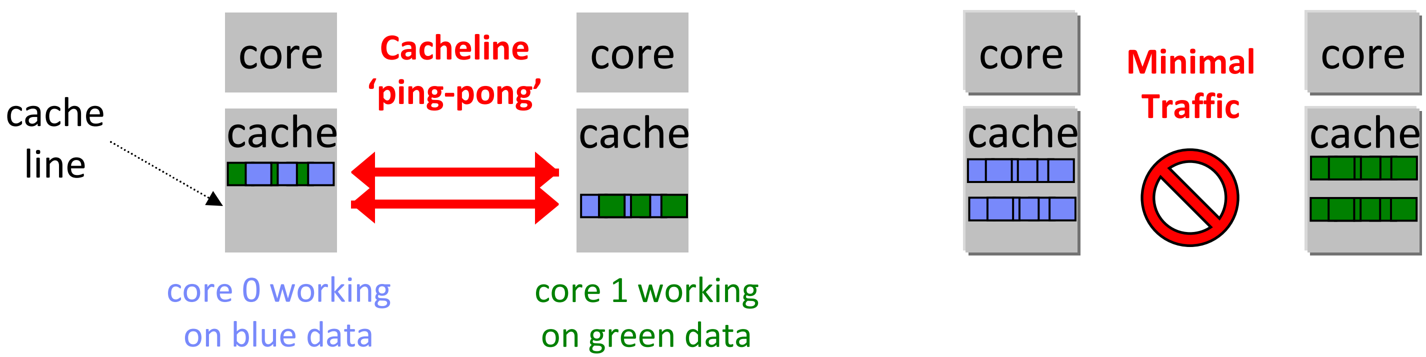
… maximize CPU cache hit rate & cacheline efficiency



7 Big Ideas: 4 *Core-Friendly Parallelism*

- **BLU queries automatically parallelized across cores, and, achieve excellent multi-core scalability via …**
  – careful data placement and alignment
  – careful attention to physical attributes of the server
  – and other factors, designed to …

… maximize CPU cache hit rate & cacheline efficiency



7 Big Ideas: 5 *Column Oriented Storage*

- Massive improvements in I/O efficiency
  - Only perform I/O on the columns involved in the query
  - No need to consume bandwidth for other columns
  - Deeper compression possible due to commonality within column values

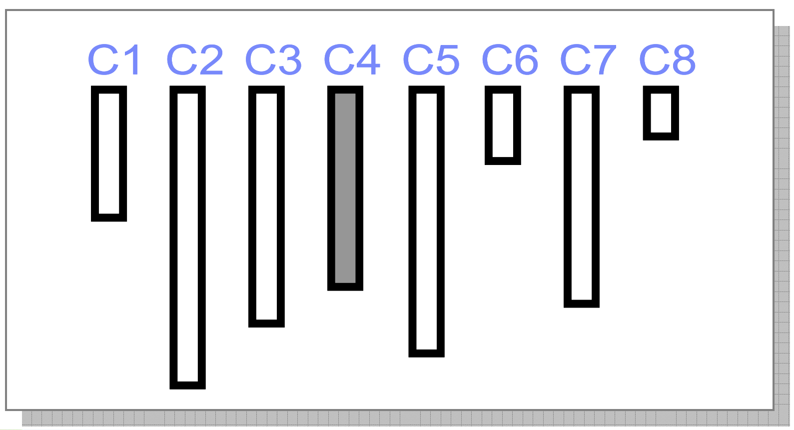
- Massive improvements in memory and cache efficiency
  - Columnar data kept compressed in memory
  - Data packed into cache friendly structures
  - Late materialization
    - Predicates, joins, scans, etc. all operate on columns packed in memory
  - Rows are not materialized until absolutely necessary to build result set
  - No need to consume memory/cache space & bandwidth for unneeded columns

7 Big Ideas: 5 *Column Oriented Storage*

- Massive improvements in I/O efficiency
  - Only perform I/O on the columns involved in the query
  - No need to consume bandwidth for other columns
  - Deeper compression possible due to commonality within column values

- Massive improvements in memory and cache efficiency
  - Columnar data kept compressed in memory
  - Data packed into cache friendly structures
  - Late materialization
    - Predicates, joins, scans, etc. all operate on columns packed in memory
  - Rows are not materialized until absolutely necessary to build result set
  - No need to consume memory/cache space & bandwidth for unneeded columns

Columns stored separately and packed in different buffers in memory



SELECT C4 ... WHERE C4=X

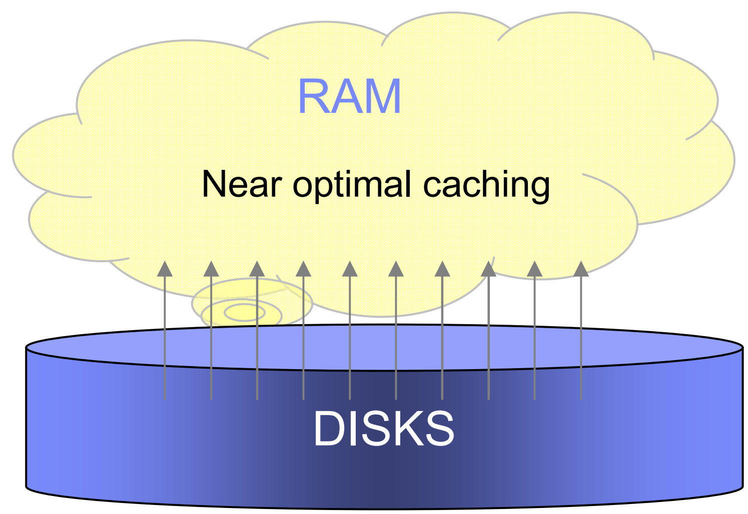
Consumes I/O bandwidth memory buffers and memory bandwidth only for C4

Need webcast troubleshooting help? Click attachments

7 Big Ideas: 6 *Scan-Friendly Memory Caching*

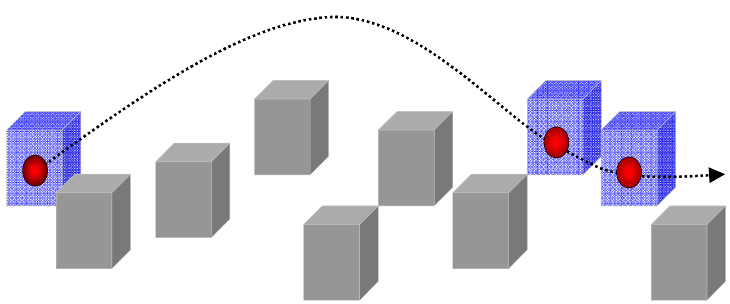
- Memory-optimized (not "In-Memory")
  - No need to ensure all data fits in memory

- BLU includes new scan-friendly victim selection to keep a near optimal % of pages buffered in memory
  - Traditional RDMSes use 'most recently used' victim selection for large scans
    - "There's no hope of caching everything, so just victimize the last page read"
  - A key BLU design point is to run well when all data fits in memory, and when it doesn't !
    - Even with large scans, BLU prefers selected pages in the bufferpool, using an algorithm that adaptively computes a target hit ratio for the current scan, based on the size of the bufferpool, the frequency of pages being re-accessed in the same scan, and other factors
  - Benefit: less I/O !



7 Big Ideas: 7 *Data skipping*

- Automatic detection of large sections of data that do not qualify for a query and can be ignored
- Order of magnitude savings in all of I/O, RAM, and CPU
- No DBA action to define or use – "Synopsis" automatically created and maintained as data is LOADed or INSERTed
  - Persistent storage of min and max values for sections of data values



Coca-Cola Bottling Co. CONSOLIDATED

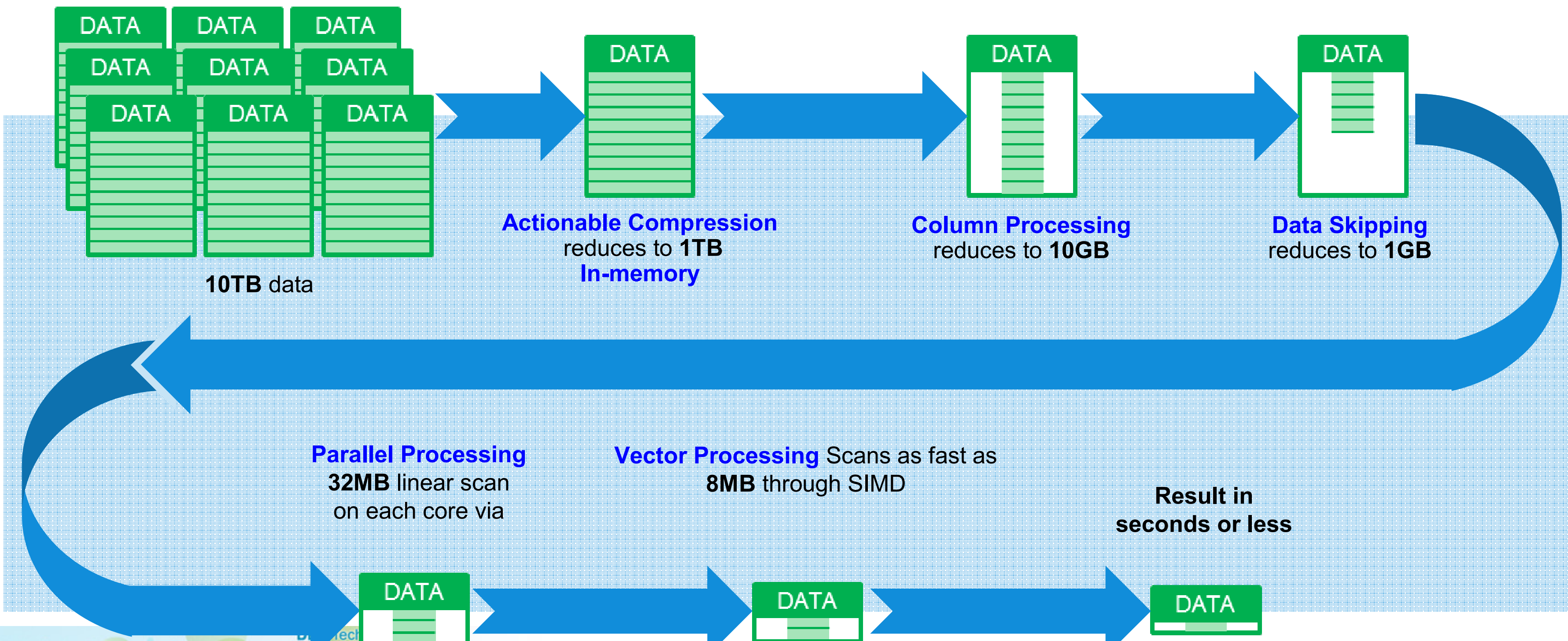
> *"One thing evident to me is that there is a lot of technology behind BLU Acceleration. It's* ***beyond a simple in-memory column store. It includes leveraging the latest CPU technologies, parallelism techniques, and so much more."***
>
> *-Andrew Juarez, Lead SAP Basis and DBA*

BLU Acceleration Illustration

10TB query in seconds or less

Register encoded vector processing

- **The System**: 32 cores, 1TB memory, 10TB table with 100 columns and 10 years of data
- **The Query**: How many "sales" did we have in 2010?
  - `SELECT COUNT(*) from MYTABLE where YEAR = '2010'`
- **The Result**: In seconds or less as each CPU core examines the equivalent of just 8MB of data



Need webcast troubleshooting help? Click attachments

Part 2: Getting Started with BLU Acceleration

Getting Started with BLU: Platforms and Hardware

- Supported platforms
  - Linux 64-bit on Intel/AMD hardware
    - RHEL 6 or higher, SLES 10 SP4, SLES 11 SP2
  - AIX on Power hardware
    - AIX 6.1 TL7 SP6, AIX 7.1 TL1 SP6
  - No support yet for Windows, Solaris, HP-UX, zLinux, etc.

- For best results, use:
  - Intel Nehalem or better
  - Power 7

Capacity Recommendations (subject to verification)

- Small: ~1TB of raw CSV data
  - 16 cores with 128 to 256GB of RAM
- Medium: ~5TB of raw CSV data
  - 16 or 32 cores with 384 to 512 GB of RAM
- Large: ~10TB of raw CSV data
  - 32 or 64 cores with 1 to 2 TB of RAM

Will your workload benefit from BLU?

Probably:

- Analytical workloads, data marts, etc.
- Grouping, aggregation, range scans, joins
- Queries touch only a subset of the columns in a table
- Star Schema

Probably not:

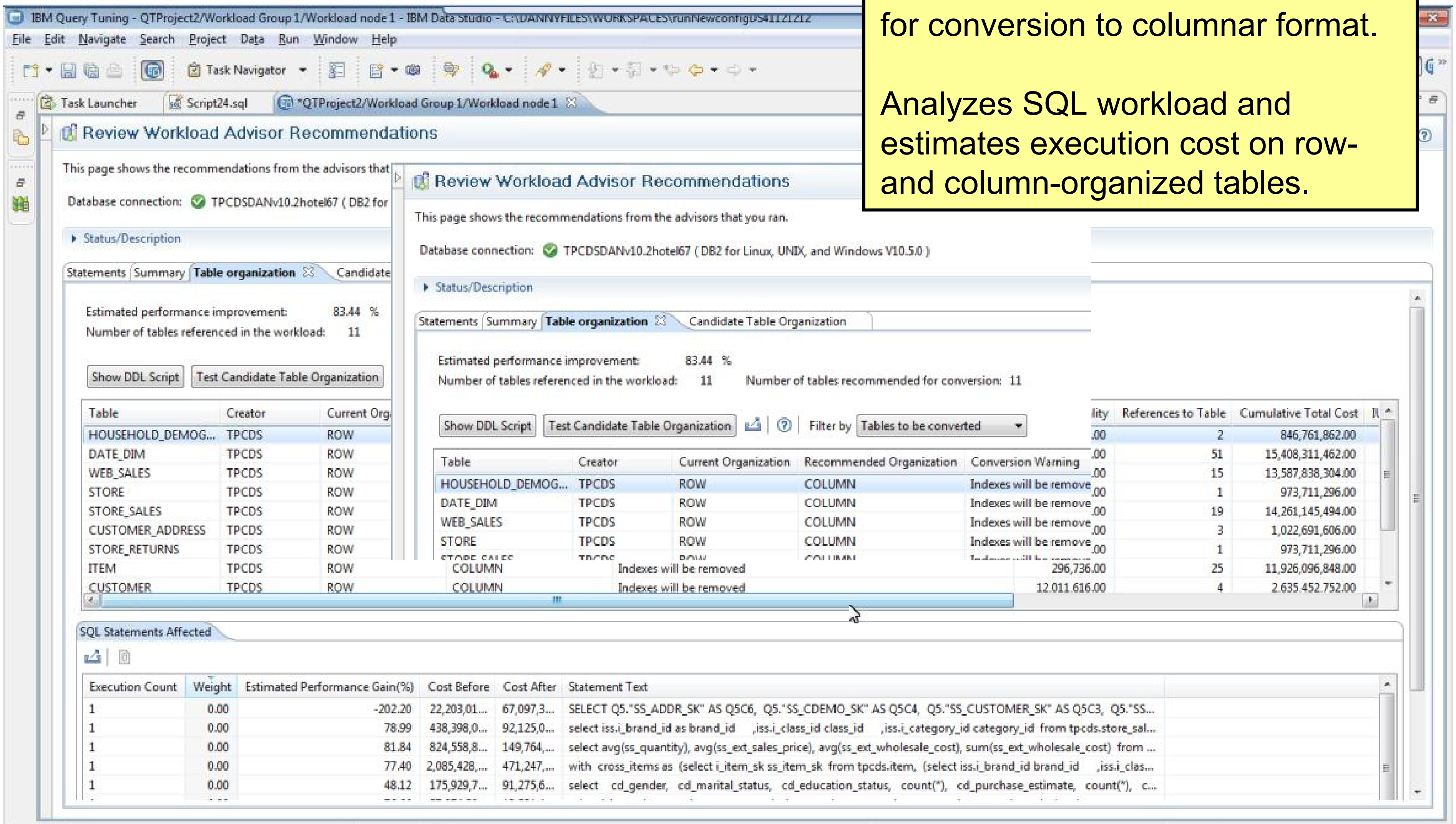
- OLTP
- Point access to 1 or few rows
- Insert, Update, Delete of few rows per transaction
- Queries touch many or all columns in a table
- Heavy use of XML, Temporal, LOBs, etc.

Need webcast troubleshooting help? Click attachments

IBM Optim Query Tuner

Advisor identifies candidate tables for conversion to columnar format.

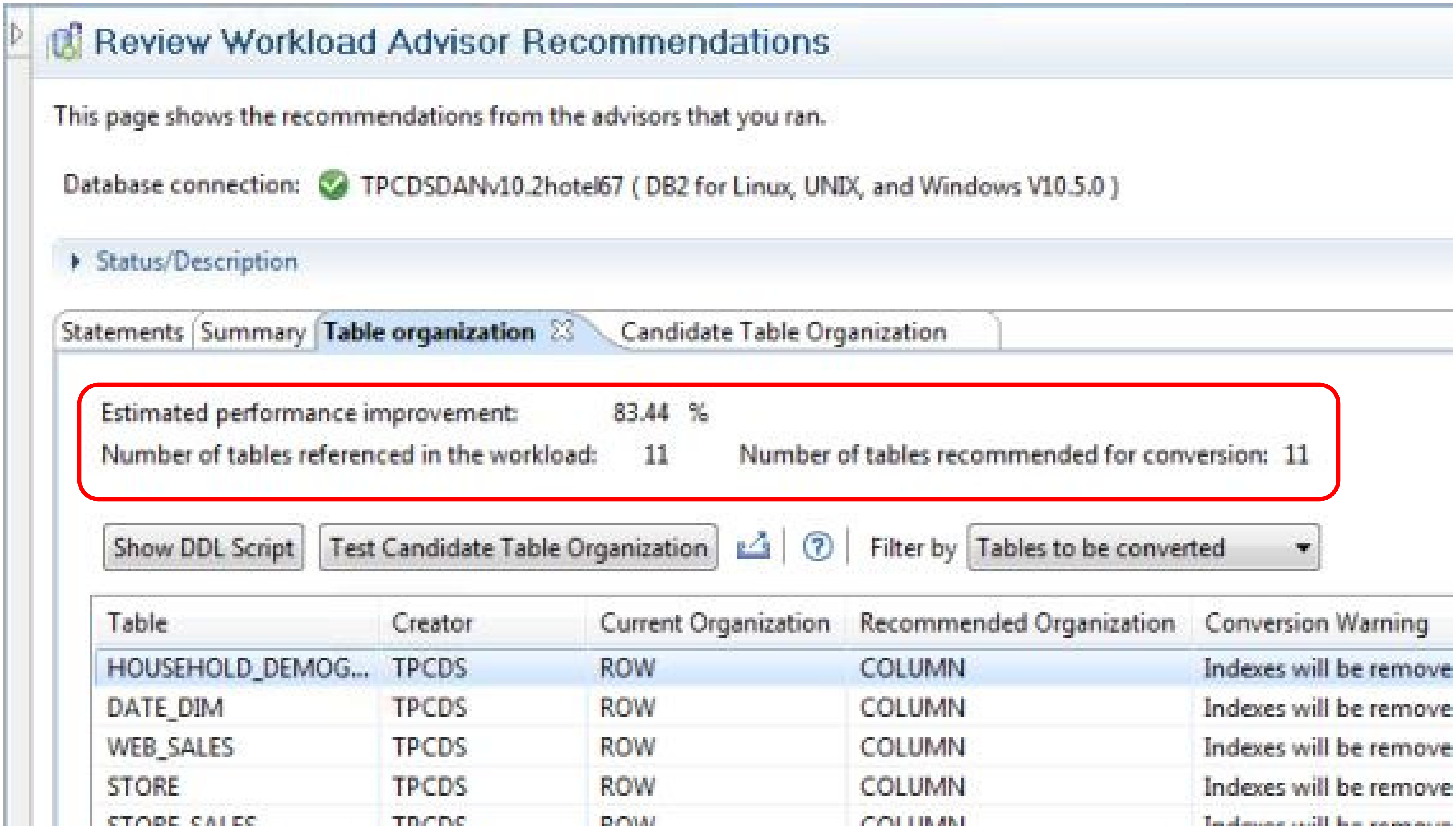
Analyzes SQL workload and estimates execution cost on row- and column-organized tables.



DB2 TechTalk

Need webcast troubleshooting help? Click attachments

IBM Optim Query Tuner



db2set DB2_WORKLOAD=ANALYTICS

- Set **DB2_WORKLOAD=ANALYTICS** **before** creating your database
- Don't disable AUTOCONFIGURE

- For an existing database:
  - set DB2_WORKLOAD=ANALYTICS
  - then run AUTOCONFIGURE

    to get some (but not all) of the recommended settings

- Ideally, you won't need to set anything else!
- Verify that sort heap, utility heap, and BPs are large

Need webcast troubleshooting help? Click attachments

DB2_WORKLOAD=ANALYTICS – What does it do?

- dft_table_org **= COLUMN**
- default page size **for a new database is 32KB**
- dft_extent_sz **= 4**
- dft_degree **= ANY**
- Intra query parallelism **is enabled for any workload (including SYSDEFAULTUSERWORKLOAD) that specifies MAXIMUM DEGREE DEFAULT, even if intra_parallel is disabled.**
- catalogcache_sz **- higher value than default**
- sortheap **and** sheapthres_shr **- higher value than default.**
- util_heap_sz **– higher value than default**
- WLM **controls concurrency on SYSDEFAULTMANAGEDSUBCLASS.**
- Automatic table maintenance **and** auto_reorg **= ON,**
  performs space reclamation for column-organized tables by default.

Creating a column-organized table

- Example:

```
CREATE TABLE sales_col (
   c1 INTEGER NOT NULL,
   c2 INTEGER,
    ...
   PRIMARY KEY (c1) ) ORGANIZE BY COLUMN;
```

Columnar tables are always compressed by default.

- If dft_table_org = COLUMN (or DB2_WORKLOAD= ANALYTICS):
  - `ORGANIZE BY COLUMN` is the default and can be omitted
  - Use `ORGANIZE BY ROW` to create row-organized tables

- Do not specify compression, MDC, or partitioning for BLU tables.
- Do not create indexes or MQTs.

Non-enforced PK / FK constraints

- Only non-enforced foreign keys are supported.
- Primary keys and unique constraints can be enforced or not enforced:

```
CREATE TABLE sales_col (
   c1 INTEGER NOT NULL,
   c2 INTEGER,
    ...
   PRIMARY KEY (c1) NOT ENFORCED) ORGANIZE BY COLUMN;
```

Columnar storage in DB2 (conceptual)

- **Separate set of extents and pages for each column**

TSN = Tuple Sequence Number

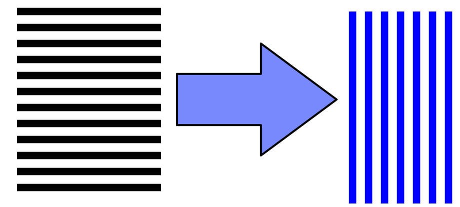
TSN						
0	John Piconne	47	18 Main Street	Springfield	MA	01111
1	Susan Nakagawa	32	455 N. 1st St.	San Jose	CA	95113
2	Sam Gerstner	55	911 Elm St.	Toledo	OH	43601
3	Chou Zhang	22	300 Grand Ave	Los Angeles	CA	90047
4	Mike Hernandez	43	404 Escuela St.	Los Angeles	CA	90033
5	Pamela Funk	29	166 Elk Road #47	Beaverton	OR	97075
6	Rick Washington	78	5661 Bloom St.	Raleigh	NC	27605
7	Ernesto Fry	35	8883 Longhorn Dr.	Tucson	AZ	85701
8	Whitney Samuels	80	14 California Blvd.	Pasadena	CA	91117
9	Carol Whitehead	61	1114 Apple Lane	Cupertino	CA	95014
10						
11						
…						

page

page

- **Typically, column-organized tables use less space than row-organized tables**
- **Column-organized tables with many columns and few rows can be larger than row-organized tables! (many extents, possibly largely empty)**
- **TSNs indicate which column values belong together as a logical "row"**

Converting existing tables: db2convert



- Converts a row-organized table into a column-organized table
- Calls ADMIN_MOVE_TABLE
- Has the same options and restrictions as ADMIN_MOVE_TABLE

```
db2convert
       -d <database-name>     (this is the only mandatory parameter)
       -stopBeforeSwap
       -continue              (resumes a previously stopped conversion)
       -z <schema-name>
       -t <table-name>
       -ts <tablespace for new table>
       -opt <ADMIN_MOVE_TABLE options>   (e.g. COPY_USE_LOAD)
       ...
```

What you see in the DB2 catalog: TABLEORG

- Which tables are column-organized?
  - New column in syscat.tables: TABLEORG

```
SELECT tabname, tableorg, compression
FROM   syscat.tables
WHERE  tabname like 'SALES%';

TABNAME                         TABLEORG COMPRESSION
------------------------------- -------- -----------
SALES_COL                       C
SALES_ROW                       R        N

  2 record(s) selected.
```

For column-organized tables, COMPRESSION is always blank because you cannot enable/disable compression.

What you see in the DB2 catalog: Synopsis Tables

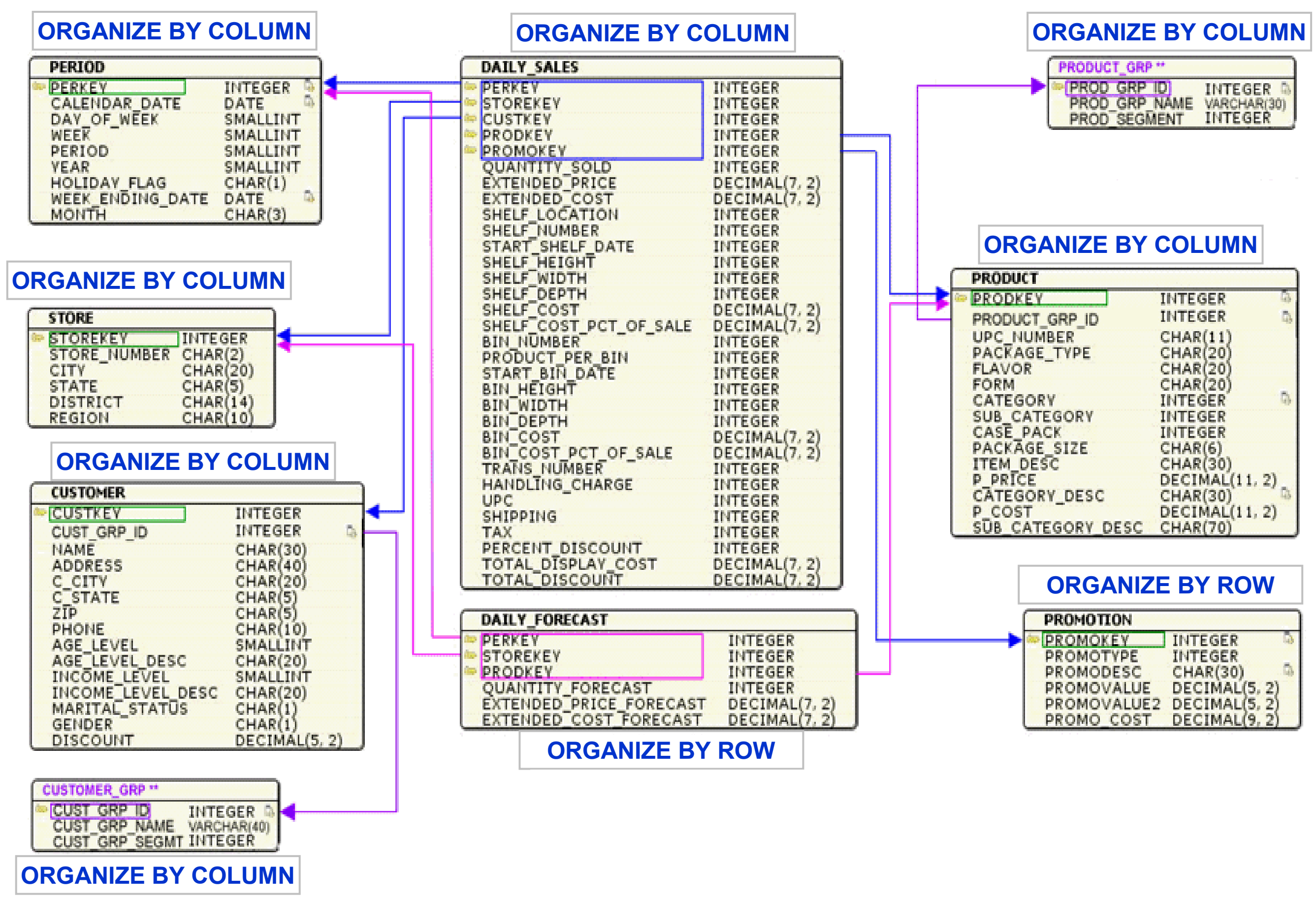
- For each columnar table there is a corresponding *synopsis table*, automatically created and maintained.

```
SELECT tabschema, tabname, tableorg
FROM syscat.tables
WHERE tableorg = 'C';

TABSCHEMA       TABNAME                              TABLEORG
--------------- ------------------------------------ --------
MNICOLA         SALES_COL                            C
SYSIBM          SYN130330165216275152_SALES_COL      C

  2 record(s) selected.
```

  – Size of the synopsis table: ~0.1% of the user table
  – 1 row for every 1024 rows in the user table

Mixing Row and Columnar Tables

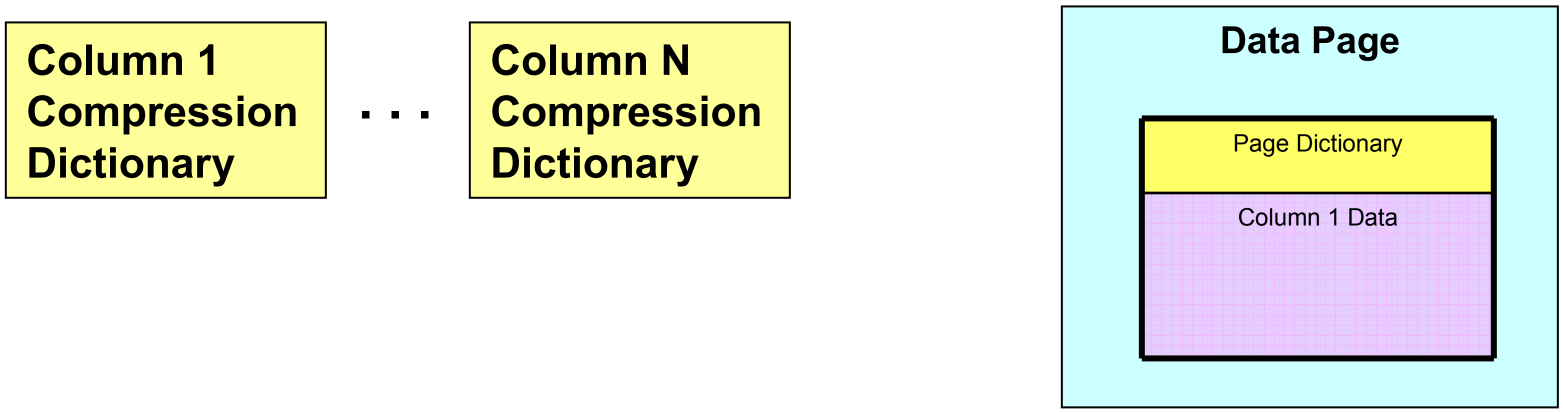
- DB2 10.5 supports mixing row and columnar tables seamlessly
  - In the same tablespace and bufferpools
  - In the same query
- Best query performance for analytic queries usually occurs with all tables columnar
- Mixing row and columnar can be necessary
  - Point queries (highly selective access) favor row-organized tables with index access
  - Small, frequent, write operations favor row-organized tables



© 2013 IBM Corporation

Part 3: LOAD and Compression

Compression Dictionaries for Column-Organized Tables



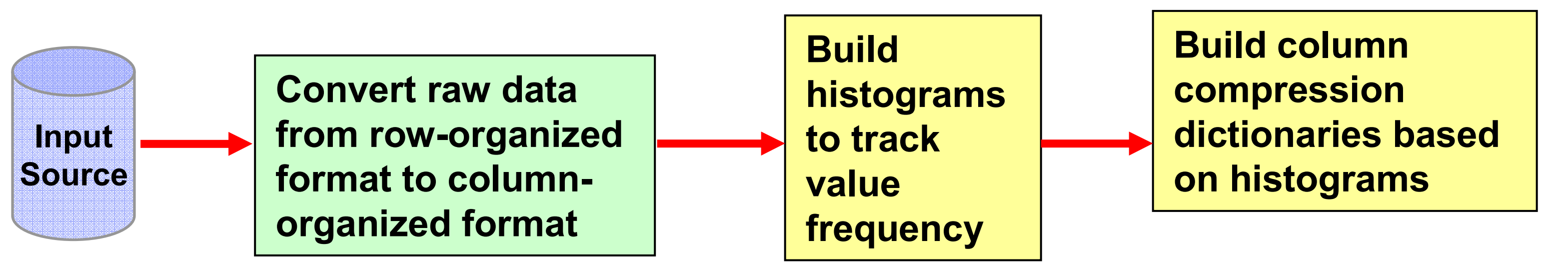
- **Column-level dictionaries:** **Always one per column**
  - Dictionary populated during load replace, load insert into an empty table, or Automatic Dictionary Creation during Insert
- **Page-level dictionaries:** **May also be created**
  - Exploit local data clustering at page level to further compress data
  - Space savings must outweigh cost of storing page-level dictionaries

Load Example

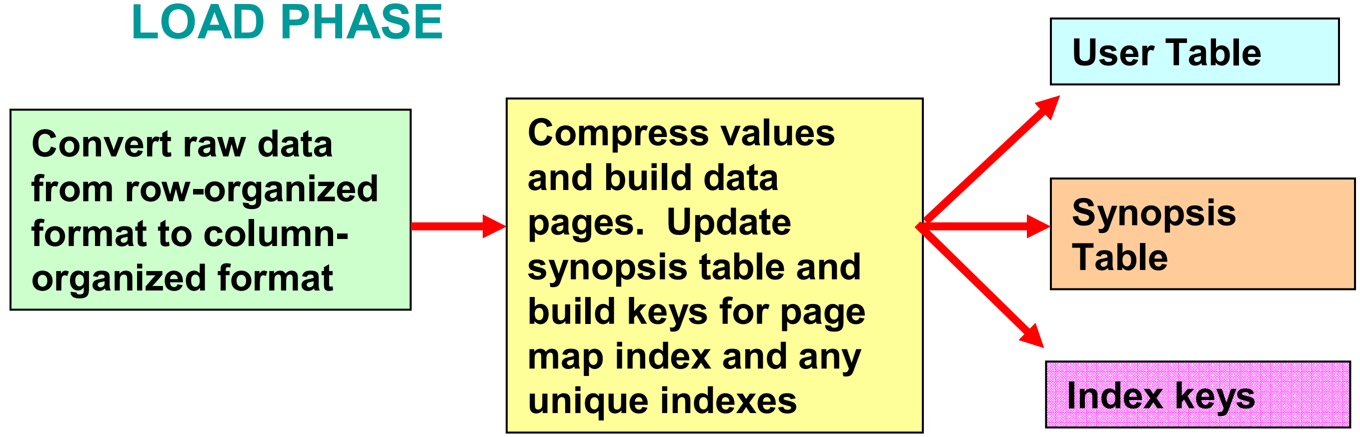
```
LOAD FROM /db1/svtdbm1/data.del OF DEL INSERT INTO colTable1;

SQL3109N The utility is beginning to load data from file "/db1/svtdbm1/data.del".
SQL3500W The utility is beginning the "ANALYZE" phase at time "04/15/2013 14:56:02.272825".
SQL3519W Begin Load Consistency Point. Input record count = "0".
SQL3520W Load Consistency Point was successful.
SQL3515W The utility has finished the "ANALYZE" phase at time "04/15/2013 14:56:03.327893".

SQL3500W The utility is beginning the "LOAD" phase at time "04/15/2013 14:56:03.332048".
SQL3110N The utility has completed processing. "300000“ rows were read from the input file.
SQL3519W Begin Load Consistency Point. Input record count = "300000".
SQL3520W Load Consistency Point was successful.
SQL3515W The utility has finished the "LOAD" phase at time "04/15/2013 14:56:04.639261".

SQL3500W  The utility is beginning the "BUILD" phase at time "04/15/2013 14:57:06.848727".
SQL3213I  The indexing mode is "REBUILD".
SQL3515W  The utility has finished the "BUILD" phase at time "04/15/2013 14:59:07.487172".

Number of rows read = 300000
Number of rows skipped = 0
Number of rows loaded = 300000
Number of rows rejected = 0
Number of rows deleted = 0
Number of rows committed = 300000
```

Load for Column-Organized Tables

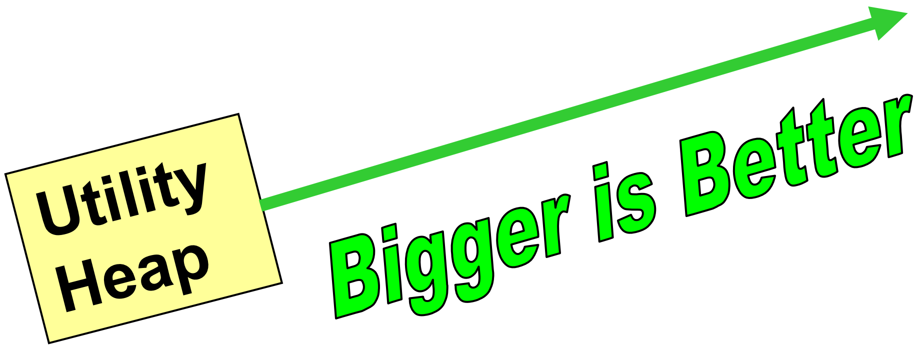
ANALYZE PHASE only if dictionaries need to be built



LOAD PHASE



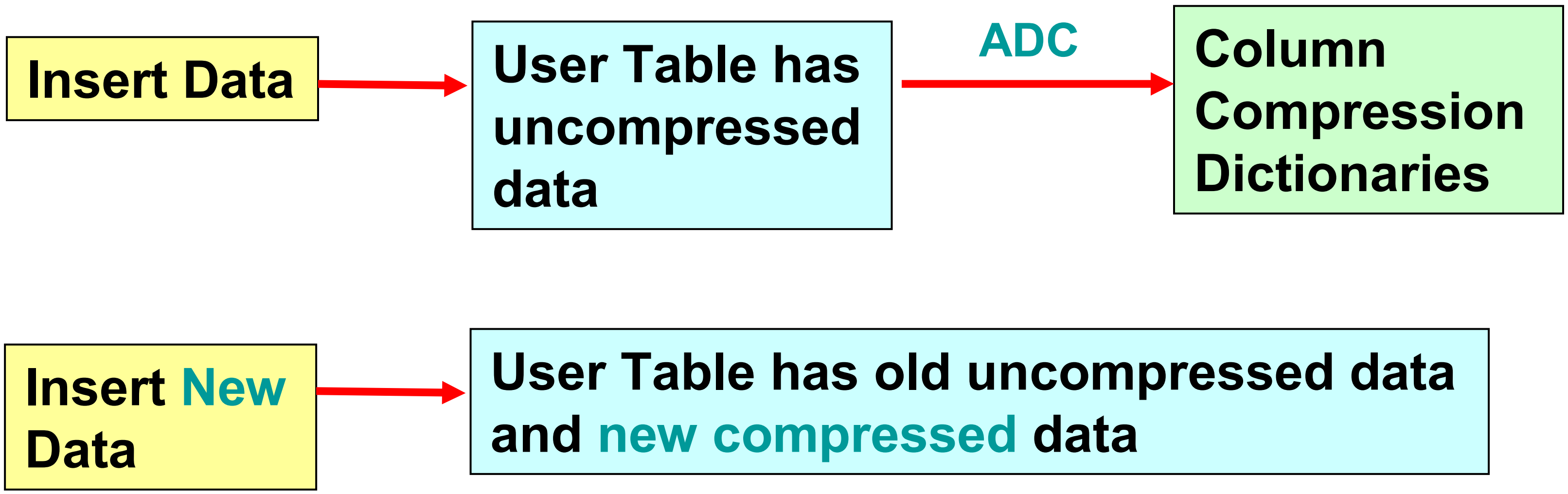
Memory Considerations for Load



- •Faster Load Performance
- •Better Compressed Tables
- •Faster Query Performance

- Load allocates memory from utility heap
- **util_heap_sz** recommendations:
  - At least **1,000,000** pages
  - **4,000,000** pages if database server has >= 128 GB of memory
  - If concurrent utilities need to be run, util_heap_sz should be increased to accommodate higher memory requirements
  - Consider reducing util_heap_sz after load completes to have more SORTHEAP memory for query usage

Inserting into Column-Organized Tables



- Initial data inserted before Automatic Dictionary Creation is uncompressed
- When threshold number of values inserted, ADC builds column compression dictionaries
  - Need enough input values to build effective dictionaries
- New values inserted after dictionaries are built are compressed

Recommendations to get Good Compression

Input a LARGE amount of representative data → **FIRST LOAD** → **Highly compressed table**

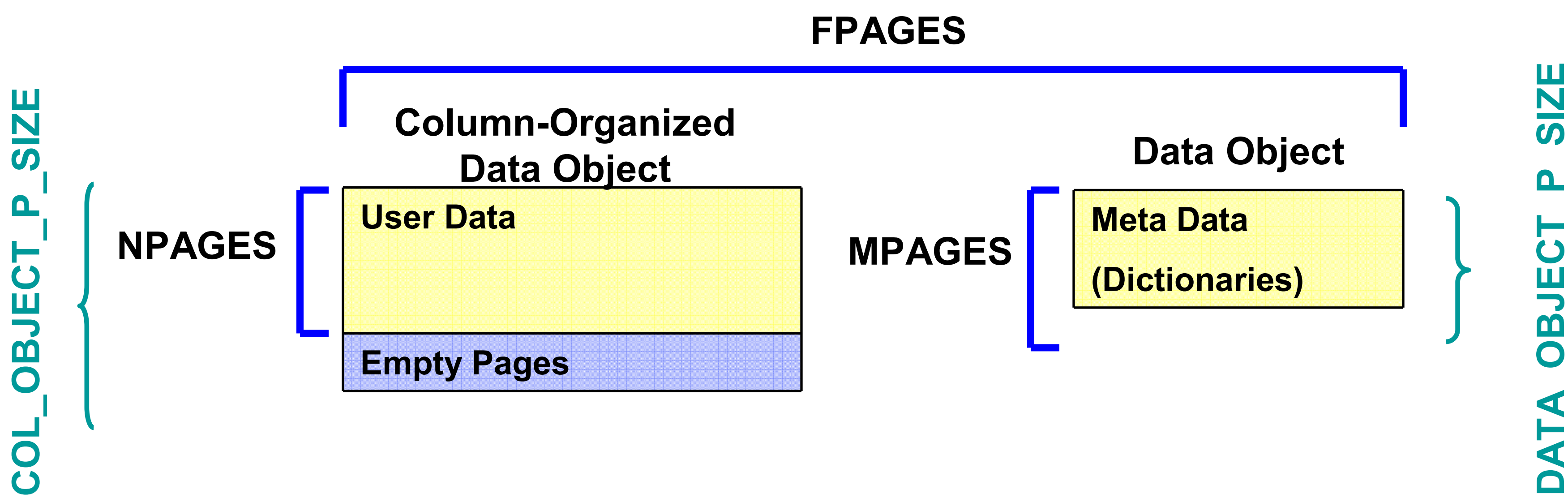
- ☑ Load instead of Insert for initial dictionary creation
  - Load utility can analyze more initial data than ADC during Insert and build better column compression dictionaries
  - Values inserted before ADC won't be compressed at the column level
- ☑ Use sufficiently large amount of representative data in 1st Load that builds dictionaries
- ☑ Set util_heap_sz >= 1,000,000 pages
- ☒ Don't load a small initial subset of data for 1st Load

Table Compression Statistics in SYSCAT.TABLES

Row-Organized Table Statistics	Column-Organized Table Statistics
PCTPAGESSAVED	PCTPAGESSAVED
AVGCOMPRESSEDROWSIZE	
AVGROWCOMPRESSIONRATIO	
AVGROWSIZE	
PCTROWCOMPRESSED	

- Only PCTPAGESSAVED applies to column-organized tables too
  - Approximate percentage of pages saved in the table
  - Runstats collects PCTPAGESSAVED by estimating the number of data pages needed to store table in uncompressed row orientation

ADMIN_GET_TAB_INFO for Column-Organized Tables



- ADMIN_GET_TAB_INFO table function reports
  - **COL_OBJECT_P_SIZE:** Physical size of column-organized data object containing user data
  - **DATA_OBJECT_P_SIZE:** Physical size of data object containing meta data

Need webcast troubleshooting help? Click attachments

IDUG Leading the DB2 User Community since 1988

IBM

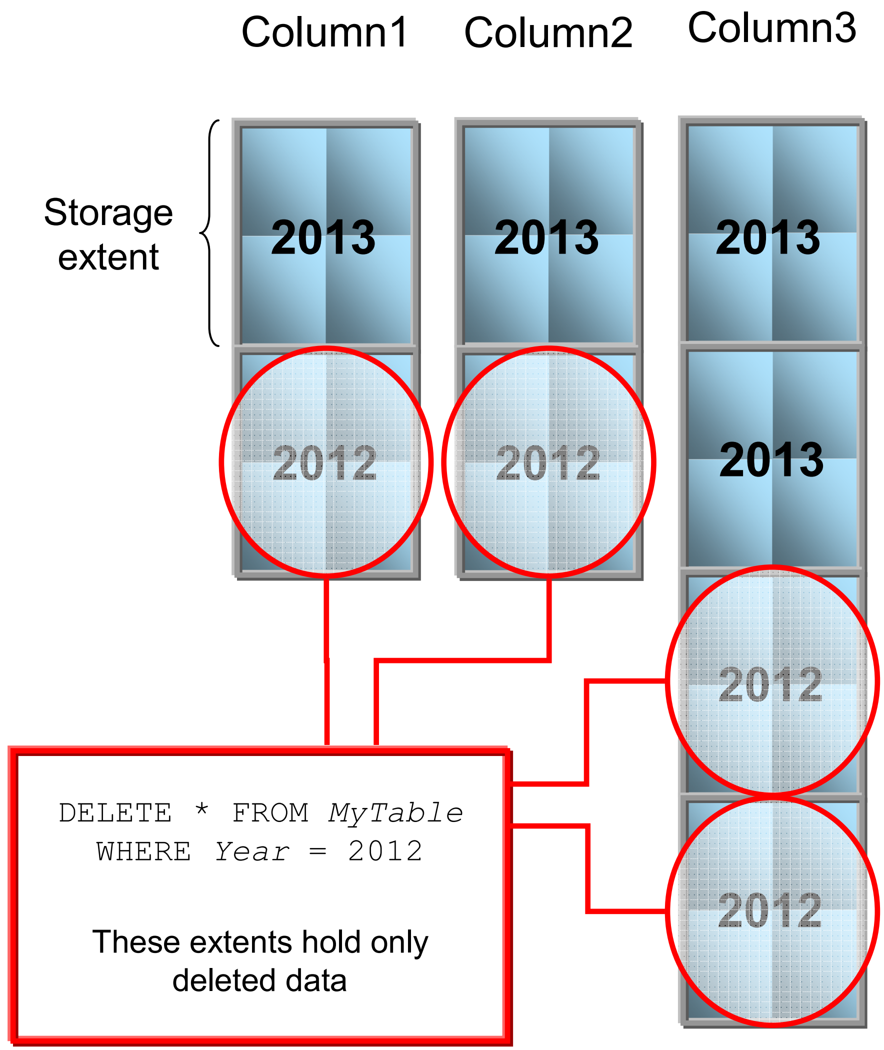
Calculating Column-Organized Storage Sizes

User Table	COL_OBJECT_P_SIZE
User Table + Meta Data + Page Map/Unique Indexes	COL_OBJECT_P_SIZE + DATA_OBJECT_P_SIZE + INDEX_OBJECT_P_SIZE

- NPAGES is approximate, but doesn't take meta data or empty pages into account
- Use the table function ADMIN_GET_TAB_INFO or admin view ADMINTABINFO to retrieve
  - COL_OBJECT_P_SIZE
  - DATA_OBJECT_P_SIZE
  - INDEX_OBJECT_P_SIZE

Automatic Space Reclaim

- Automatic space reclamation
  - Frees extents with no active values
  - The storage can be subsequently reused by any table in the table space
- No need for costly DBA space management and `REORG` utility
- Enabled out-of-the box for column-organized tables when `DB2_WORKLOAD=ANALYTICS`
- Space is freed online while work continues
- Regular space management can result in increased performance of `RUNSTATS` and some queries



Reclaiming Space in the Table

- Objective: **Find empty storage extents and return pages to table space for re-use**
- Option 1: **If DB2_WORKLOAD=ANALYTICS, automatic space reclamation is active for all column-organized tables**
- Option 2: **Enable Automatic Table Maintenance (ATM)**

```
update db cfg using auto_maint ON auto_tbl_maint ON auto_reorg ON;
```

- Option 3: **Use REORG TABLE explicitly**
  - Can use RECLAIMABLE_SPACE from ADMINTABINFO/ADMIN_GET_TAB_INFO to determine when to REORG

```
                                              .-ALLOW WRITE ACCESS--.
>>-REORG-TABLE--table-name--RECLAIM EXTENTS--+---------------------+------><
                                              +--ALLOW READ ACCESS--+
                                              '--ALLOW NO ACCESS----'
```

Part 4: Query Execution and Workload Management

Sample Query

```
SELECT c.trading_name
FROM f, c, dt
WHERE f.client_dim_key = c.client_dim_key
  AND f.trade_dt = dt.dt_dim_key
  AND f.is_cancelled = 0
GROUP BY c.trading_name, dt.year
ORDER BY c.trading_name
```

Let's review the execution plan of this query….

Need webcast troubleshooting help? Click attachments

Sample Execution Plan

```
                        <snip>
                         SORT
                         (   4)
                           |
                          CTQ
                         (   5)
                           |
                         GRPBY
                         (   6)
                           |
                        UNIQUE
                         (   7)
                           |
                        ^HSJOIN
                         (   8)
             /-----------+-----------\
         ^HSJOIN                      TBSCAN
         (   9)                       (  12)
    /---------+---------\               |
 TBSCAN               TBSCAN      CO-TABLE:  dt
 (  10)               (  11)
   |                    |
CO-TABLE:  f       CO-TABLE:  c
```

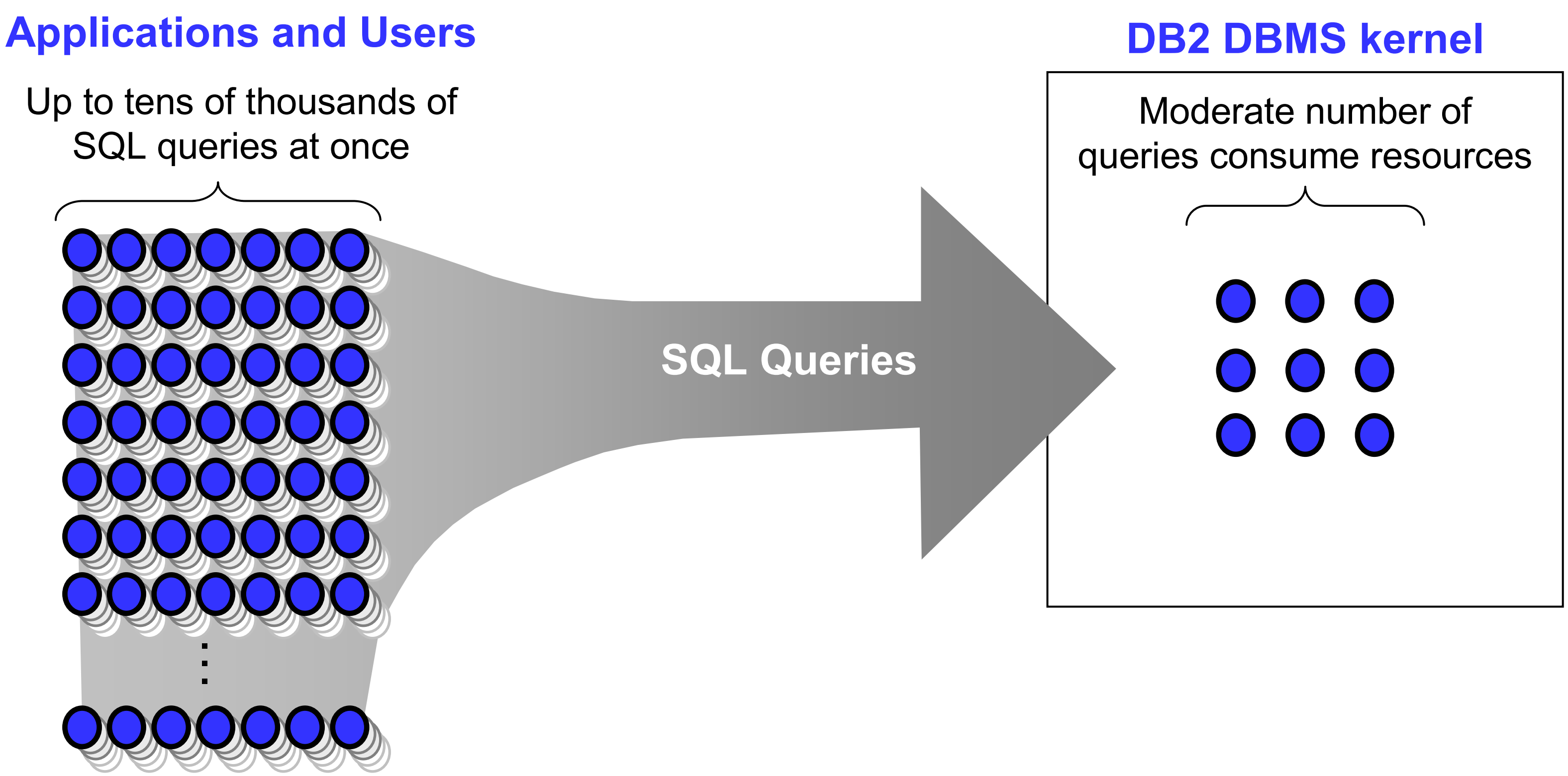
Operators above CTQ use DB2's regular row-based processing

Operators below CTQ are optimized for column-organized tables

Here: All table scans, hash joins, and grouping are performed in columnar query runtime. (Good.)

Automatic Workload Management

- **Built-in and automated query resource consumption control**
- **Enabled automatically when `DB2_WORKLOAD=ANALYTICS`**
- **Many queries can be submitted, but limited number get executed concurrently**



DB2 10.5 with BLU Acceleration

- BLU Acceleration provides three key benefits:
  - Fast
    - Unprecedented performance for analytical workloads, often 8x to 25x faster.
    - Examples of workloads > 100x
    - Examples of individual queries > 1000x
  - Small
    - Stronger compression and less space required for auxiliary data structures.
    - 10x savings is versus uncompressed row-tables is common.
  - Simple
    - Much less tuning needed, more predictable and reliable performance
    - Tuning, statistics collection, space reclaim, workload management all tuned and automated right out of the box
    - Adapts automatically to your server's memory and CPUs

*"Intel is excited to see **greater than 30x improvement** in query processing performance using DB2 10.5 with BLU acceleration over DB2 10.1. To achieve these amazing gains, IBM has taken advantage of the Advanced Vector Extensions (AVX) instruction set on Intel® Xeon® processor E5-based systems. Customers running this hardware can now immediately **realize dramatically greater performance boost at lower cost per query**."*
-Pauline Nist, Intel General Manager, Enterprise Software Alliances, Datacenter and Connected Systems Group

DB2 Tech Talk: Technical Tour of DB2 10.5 with BLU Acceleration

Next Steps Roadmap

Step One

Listen to the short video overviews

- YouTube BLU Acceleration technical video: bit.ly/147fWzo
- Hear from the developers: bit.ly/133DBDh

Step Two

Read the technical information

- Free eBook, DB2 with BLU Acceleration: ibm.co/ZBWysX
- IBM Data Mag: Super Analytics, Super Easy: bit.ly/15tauNy

Step Three

Solidify your foundation in warehousing as needed

•Real-World Warehousing for Tech Pros Tech Talk: bit.ly/tt2013mar

Step Four

Download the Technology Preview

- Software download: ibm.co/10Icf2j
- Technology preview forum: ibm.co/16czx7n

Step Five

Listen to customer and partner feedback

•IBM Champion Tony Winch: bit.ly/13dmvDv

•IBM Champions Jean-Marc Blaise and Iqbal Goralwalla: bit.ly/10z83AI

Reference

Call IBM to schedule a demo or learn more

- 1 800 966-9875 (U.S)
- 1-888-746-7426 (Canada)
- 1800-425-3333 (India)
- Or visit http://www.ibm.com/planetwide/ for contact information worldwide

IBM DB2 10.5 product page
Ibm.com/db2

IBM DB2 10.5 Product features
ibm.co/12c1PJz

Tech forum on developerWorks
bit.ly/db2forumluw

Rick Swagerman SQL Tips Blog:
www.sqltips4db2.com

BACKUP

Using Monitor Elements

- How is my table organized?
  - TAB_ORGANIZATION from MON_GET_TABLE()
    - C: column-organized
    - R: row-organized
- Is my table suitably organized?
  - num_columns_referenced: columns referenced in queries
  - section_exec_with_col_references: queries referencing columns using scan
  - Compute avg num columns accessed by query: (num_columns_referenced/ section_exec_with_col_references)
  - Favor column-organization if this avg is much less than number of table columns

Using Monitor Elements

- How is query performance?
  - New! From MON_GET_DATABASE(), MON_GET_SERVICE_SUBCLASS(), etc.
  - TOTAL_COL_TIME: elapsed time
  - TOTAL_COL_PROC_TIME: excludes lock wait, I/O, etc.
  - TOTAL_COL_EXECUTIONS: num column-organized table accesses
- How is bufferpool performance?
  Monitor column-organized versions of existing elements
  Example: COL_HIT_RATIO_PERCENT
- How is prefetch performance?
  Monitor column-organized versions of existing elements
  Example: POOL_QUEUED_ASYNC_COL_PAGES