

DB2 with BLU Acceleration: So Much More than Just a Column Store

April 30, 2014

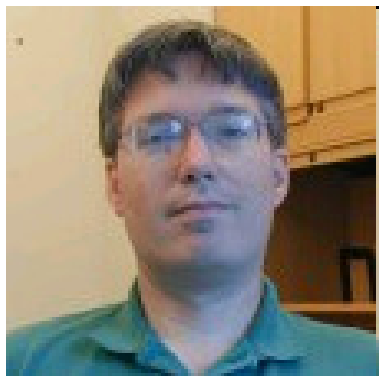
Guy Lohman

Research Manager

Disruptive Information Management Architectures

IBM Research -- Almaden

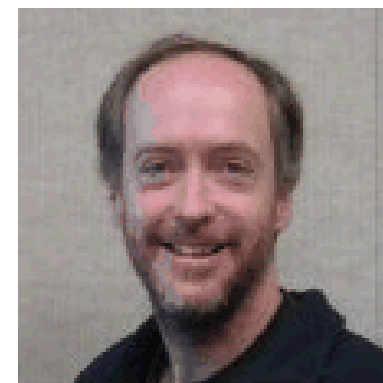
BLU Research Team



Ron Barber



Vijayshankar Raman



Richard Sidle



Ippokratis Pandis



Guy Lohman



Rene Mueller

DB2 with BLU Acceleration: Agenda

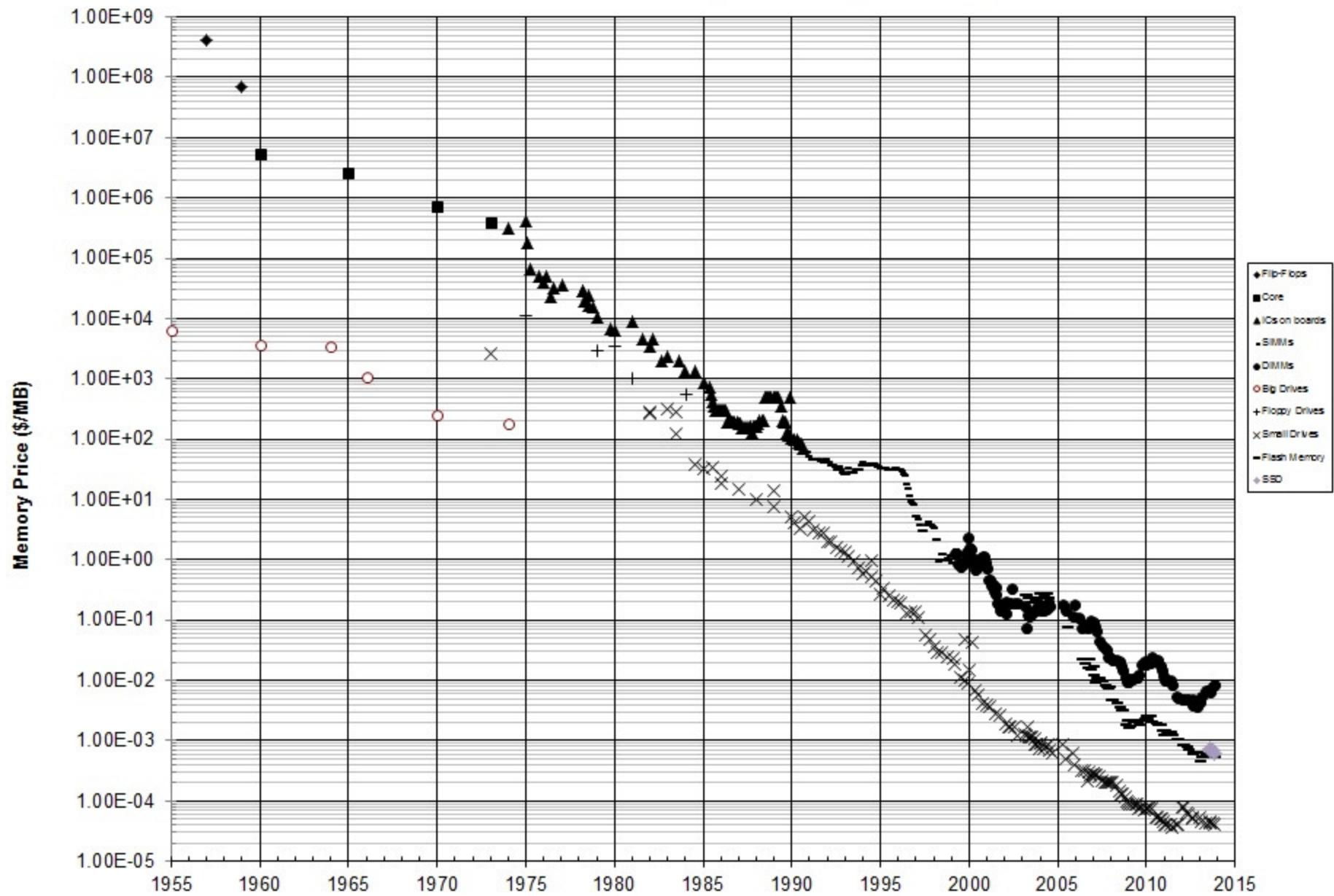


- 1. Industry Context & Motivation**
 - A. Memory-optimized databases**
 - B. Column stores**
 - C. Transactional vs. Analytic Workloads**
- 2. Introduction to BLU Acceleration**
 - A. What is BLU Acceleration?**
 - B. Business Value of BLU**
- 3. BLU's "Secret Sauce"**
- 4. Behind the Curtain -- the Technology**
- 5. Conclusions**

Part 1:

Industry Context and Motivation

Historical Cost of Computer Memory and Storage



Source: <http://www.jcmit.com/mem2013.htm>

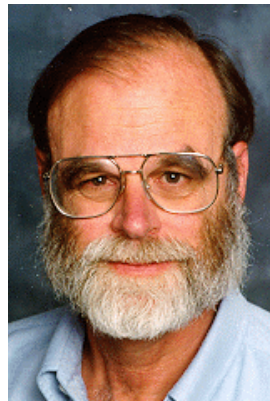
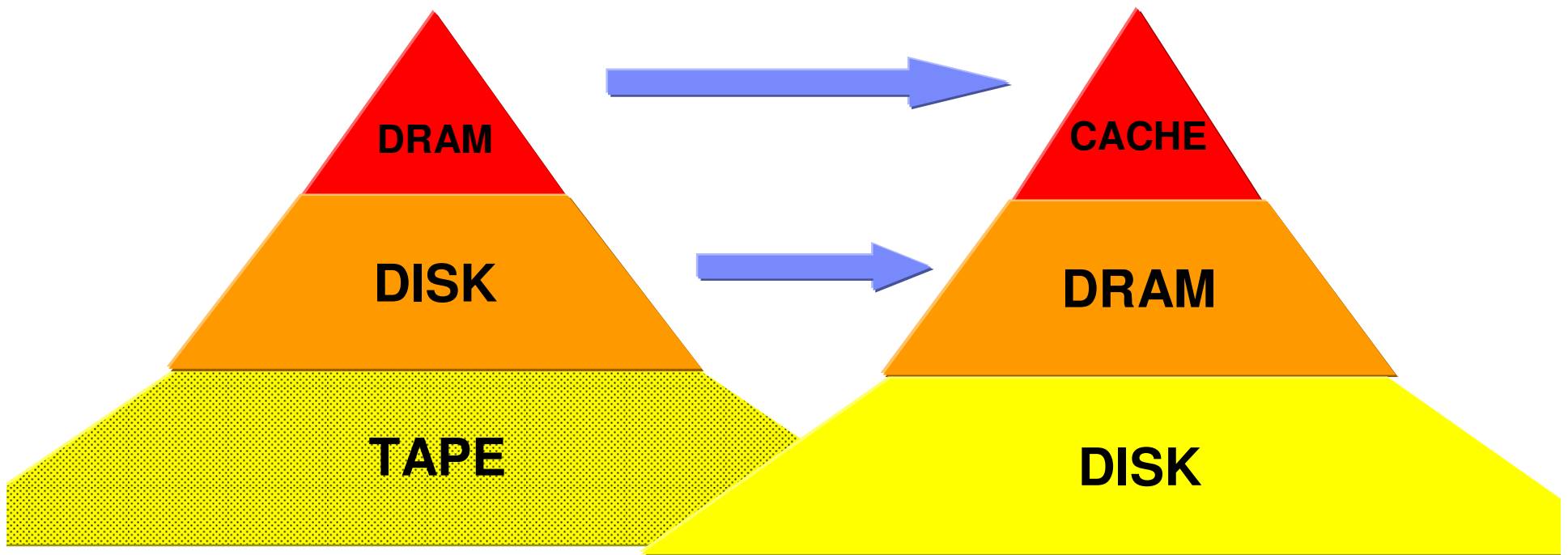
Year

Main-Memory \neq In-Memory



- **Economics + Greed → There will always be a “memory hierarchy”**
 - Yes, DRAM is getting cheaper
 - Moore’s Law has not (yet) been repealed!
 - BUT our appetites for data is growing even faster
 - The death of update-in-place
 - “Big Data” Analytics
 - Some data *will* spill to disk (just as some data spills to archive)
- **There are some differences!**
 - DRAM is byte-addressable
 - Page structures less important
 - NUMA effects between sockets
- **We’ve just moved up one level in our focus...**

Focus of Memory Hierarchy Has Shifted Up 1 Level **IBM**



**“Disk is the new tape;
Memory is the new disk.”**

What IS a Column Store?

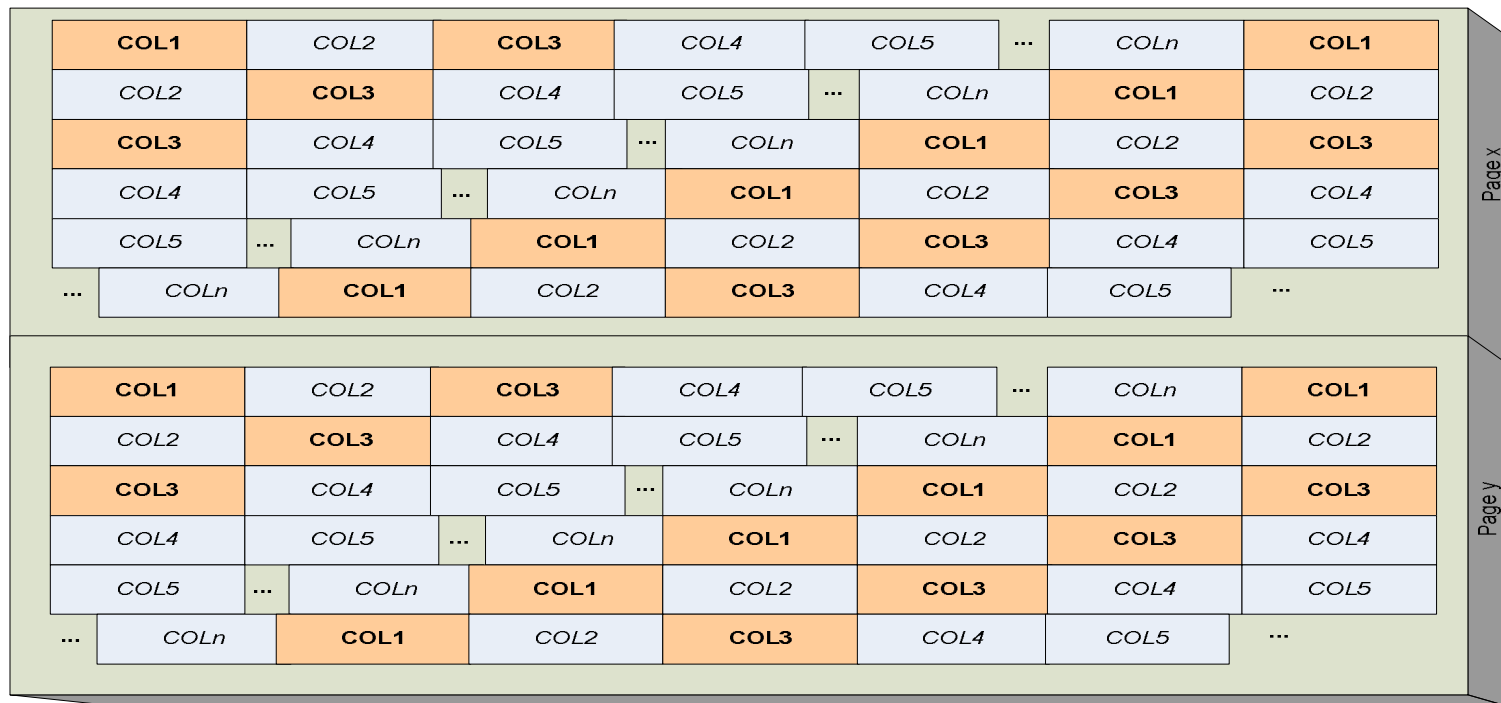


- A column store is just like a row store

| | | | | | | |
|-----------------|----|----------------------------|-------------|----|-------|----------|
| John Piconne | 47 | 18 Main Street | Springfield | MA | 01111 | \$343.12 |
| Susan Nakagawa | 32 | 455 N. 1 st St. | San Jose | CA | 95113 | \$89.00 |
| Sam Gerstner | 55 | 911 Elm St. | Toledo | OH | 43601 | \$934.12 |
| Chou Zhang | 22 | 300 Grand Ave | Los Angeles | CA | 90047 | \$722.49 |
| Mike Hernandez | 43 | 404 Escuela St. | Los Angeles | CA | 90033 | \$41.50 |
| Pamela Funk | 29 | 166 Elk Road #47 | Beaverton | OR | 97075 | \$199.43 |
| Rick Washington | 78 | 5661 Bloom St. | Raleigh | NC | 27605 | \$850.00 |
| Ernesto Fry | 35 | 8883 Longhorn Dr. | Tucson | AZ | 85701 | \$66.78 |
| Whitney Samuels | 80 | 14 California Blvd. | Pasadena | CA | 91117 | \$732,14 |
| Carol Whitehead | 61 | 1114 Apple Lane | Cupertino | CA | 95014 | \$999.11 |

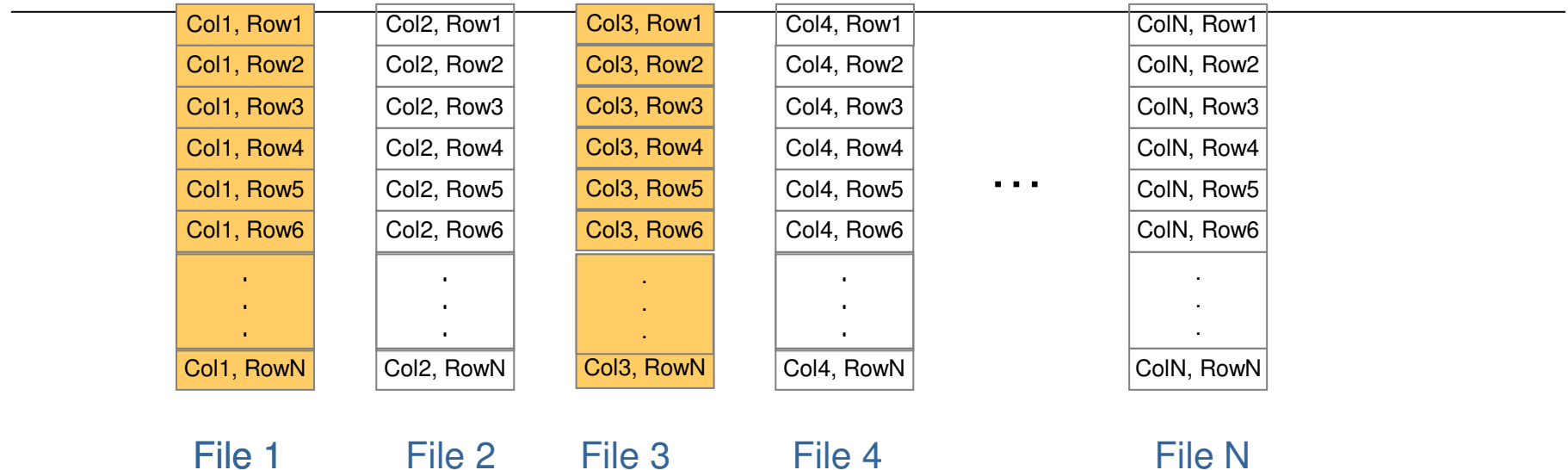
...if you tilt your head on its side!

Row Stores Group Many Complete Rows on a Page



- + Single I/O to access (all columns of) one row
- + No overhead to re-assemble row
- All columns are read, whether or not they're needed!
- Waste space in the buffer pool!

Column-Stores Group All Rows for each Column in a Separate File



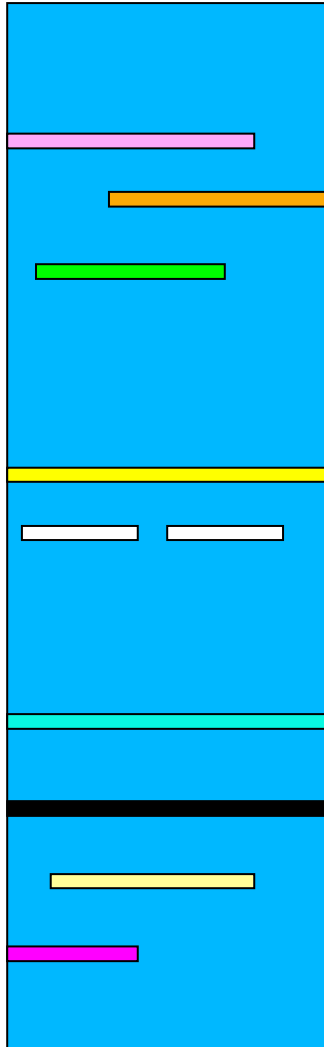
Advantages:

- + Scan only the columns required
→ Large reduction in I/O
- + High compression rates
- + Excellent CPU cache locality while processing column data
- + Obviates limits on # of columns and row length (to fit on a page) for stored tables.

Problems:

- CPU expense combining columns of qualifying rows into answer set (**projection**)
- Random I/Os performing projection
 - Particularly on the measures of the fact table
 - Eliminates much of the benefit from the I/O savings!
- Additional working storage required to merge via a key (if explicit)
- Multiple I/Os per record for all write operations
(INSERT, UPDATE, DELETE).

Access Pattern of Transaction Processing (OLTP) Workload

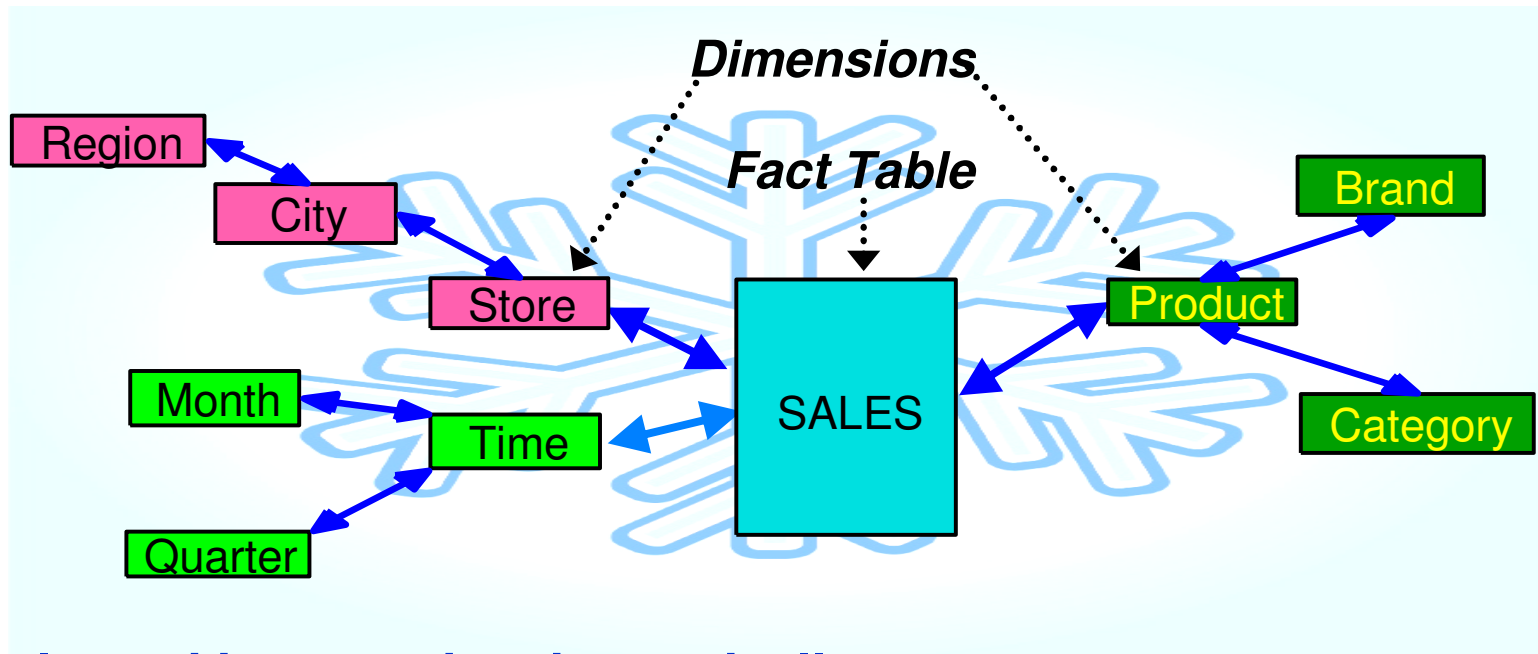


- Past 3 decades has focused on transactions (OLTP)
- Table typically has 10s to (at most) 100s of columns
- Usually access many (in reads) or most (in writes) columns of the row
- For example, you'd usually access Street, City, State, and Zip together
- Access one or a few rows per query
- Almost always accessed via index(es)
- ***Write-dominant* workload**
- Well-suited to traditional **row stores** like DB2

Target Market: Business Intelligence (BI)



- Characterized by:
 - “Star” or “snowflake” schema:

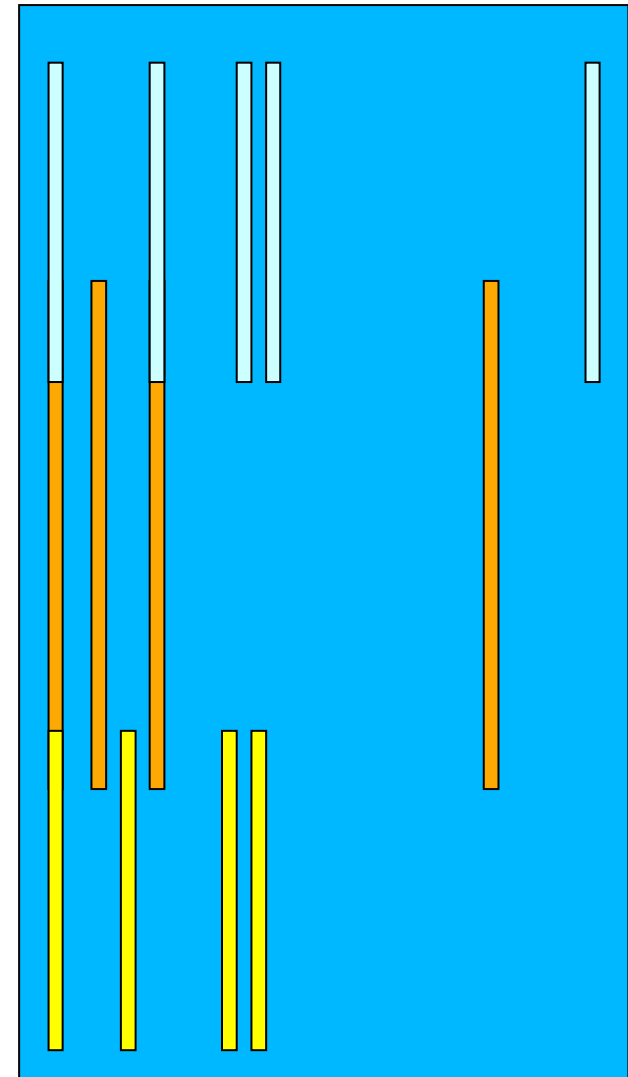


- Complex, ad hoc queries that typically
 - Look for trends, exceptions to make actionable business decisions
 - Touch large subset of the database (unlike OLTP)
 - Involve aggregation functions (e.g., COUNT, SUM, AVG,...)
 - **The “Sweet Spot” for BLU Acceleration!**

Access pattern of Business Intelligence (BI) Workload

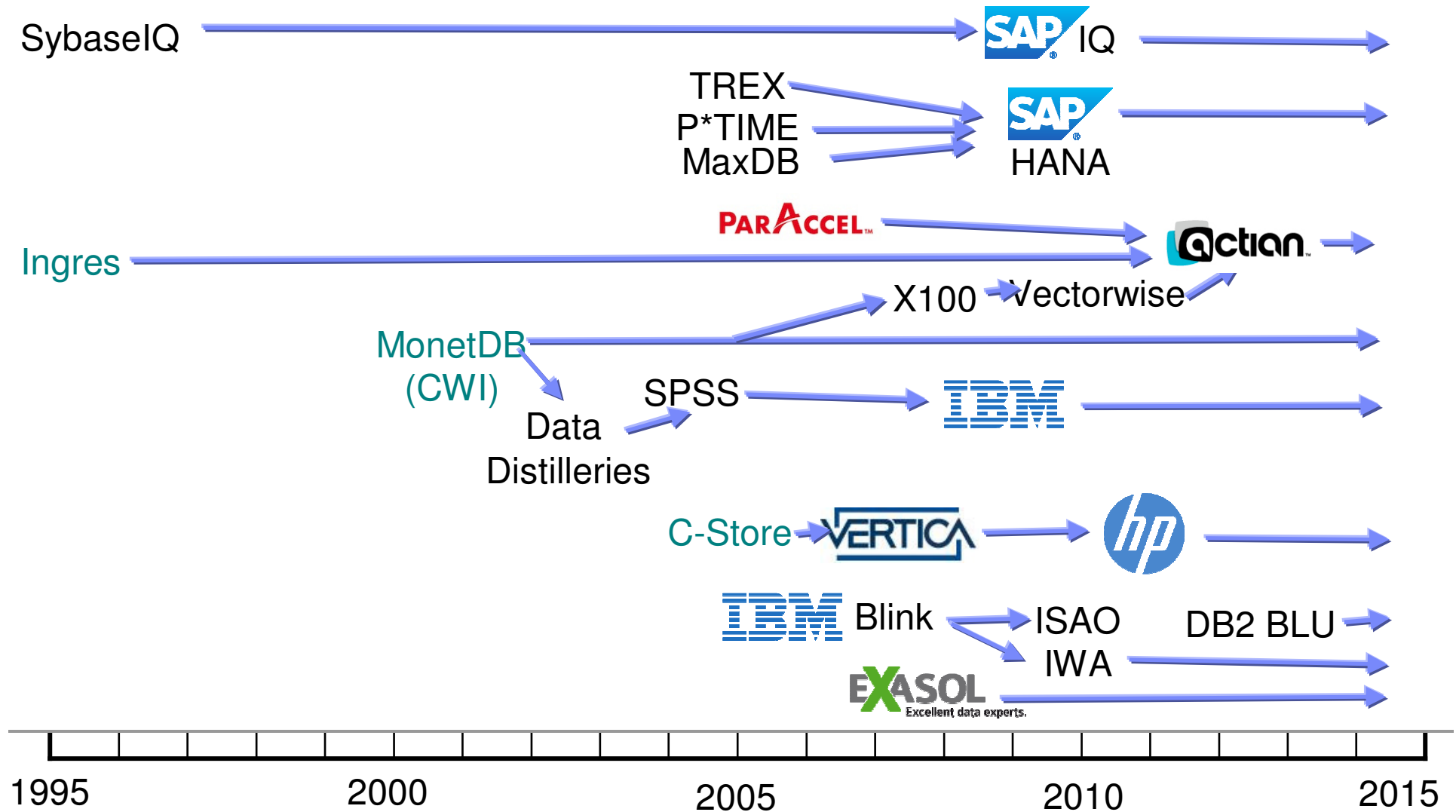


- Table typically has 100s to 1000s of columns
- Usually access small percentage of all columns (10s)
- Access many or even most rows, per query
- For example, you'd typically access all transactions for a particular State, Month, and Product Category
- Indexes less useful (more scans)
- ***Read-dominant workload***
- **Better suited to column stores**



“Related Work”

IBM



Part 2: Introduction to BLU Acceleration



BLU is Memory-Optimized Analytics
to Accelerate Your Applications...
...and Improve Your Productivity!

What is BLU Acceleration?



• New technology for accelerating BI queries

- 2nd generation of Almaden's Blink Research technology
- Columnar database within DB2
- In-memory run-time that **operates on compressed data**
- Optimized for modern multi-core hardware

• Order-of-magnitude benefits

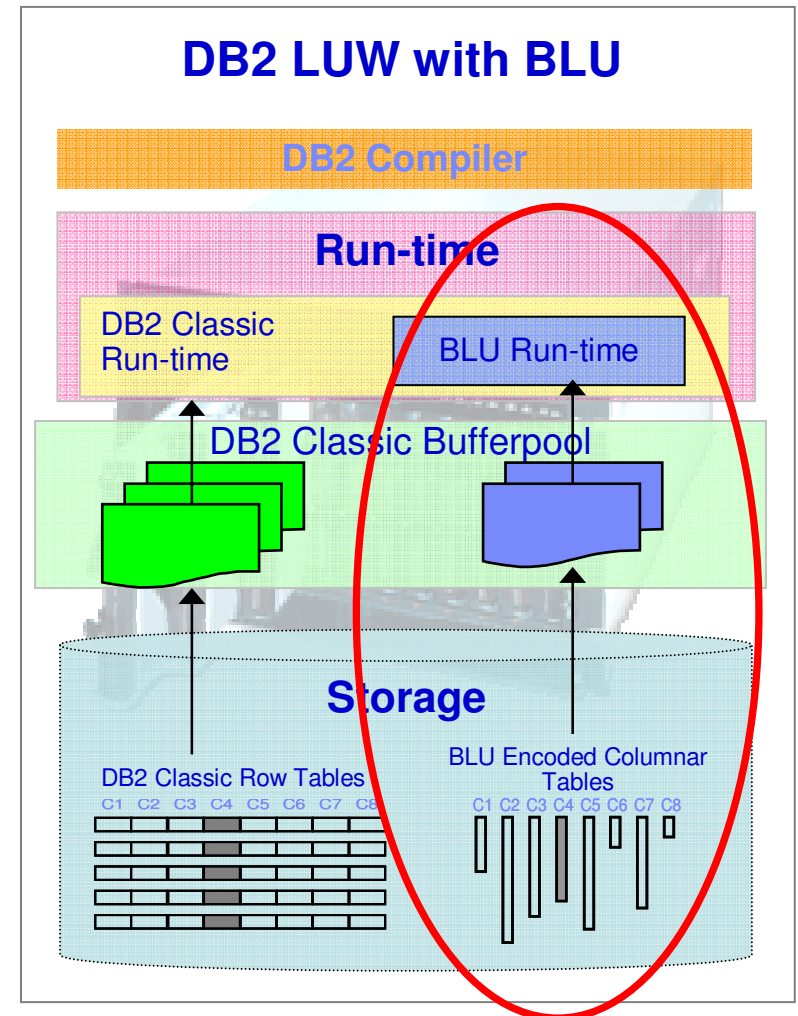
1. **Performance**
2. **Storage savings**
3. **Simplicity and Time to Value!**

• Deeply integrated within DB2 10.5

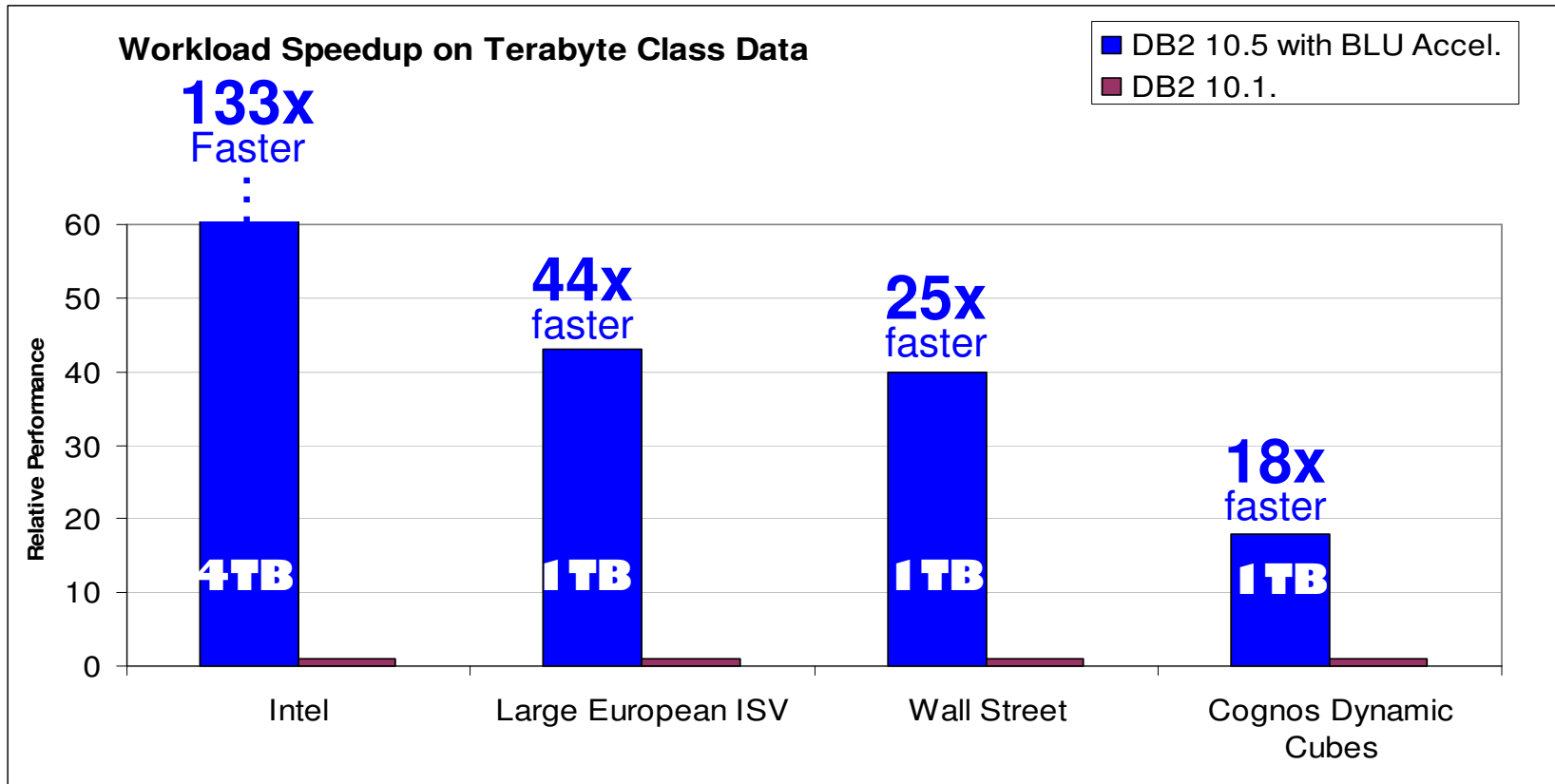
- New columnar page format & run-time
- Memory optimized (not limited to "in-memory")
- Exploits DB2 full functionality, utilities, & tools

• "Revolution via Evolution"

- Easy conversion of row tables to BLU (columnar) tables
- BLU tables can co-exist with traditional row tables
 - In same schema, storage, & memory
- Query any combination of BLU or row data
- No need to change applications or SQL queries
- DB2 run-time compensates for any missing functionality in BLU



Business Value 1: PERFORMANCE!



*"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!***

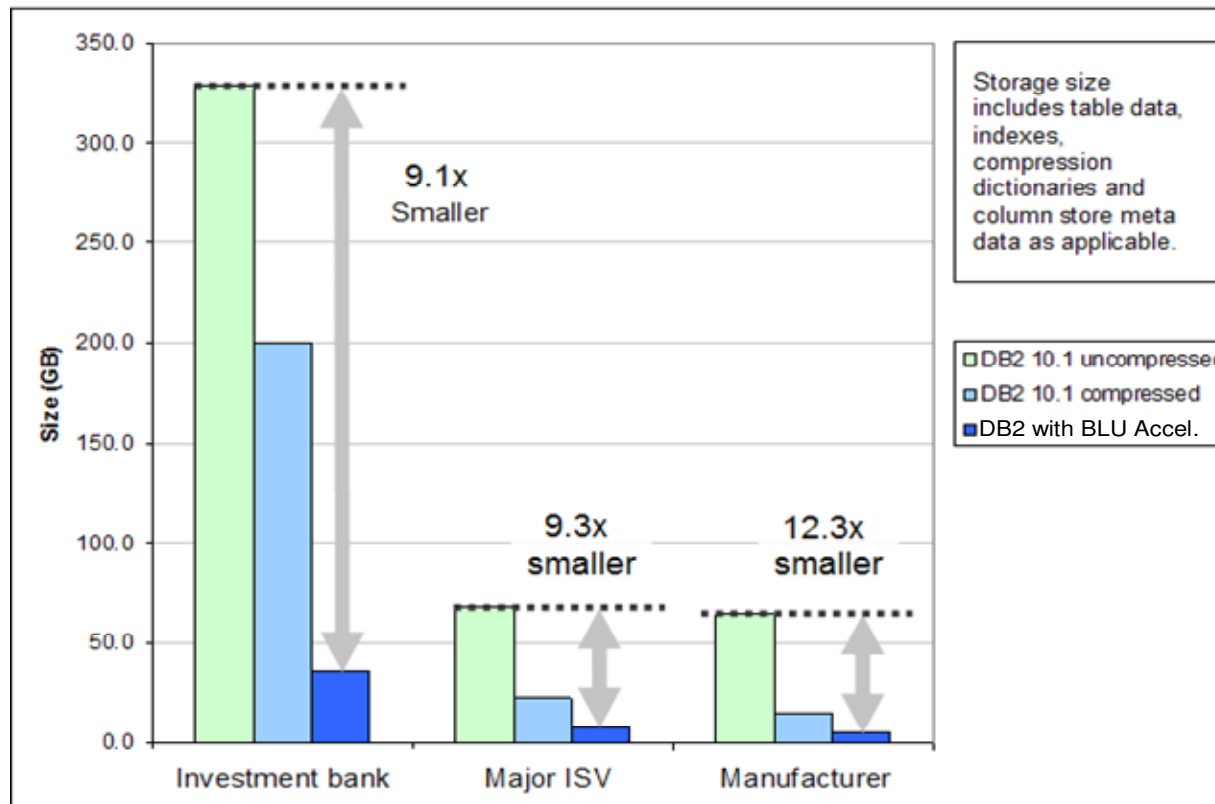
The best outcome was a query that finished 137x faster by using BLU Acceleration."

- Kent Collins, Database Solutions Architect, BNSF Railway

Business Value 2: Storage Savings!



- ~2x-3x storage reduction vs. DB2 10.1 (comparing all objects – tables, indexes, etc.)
 - Patented columnar compression techniques
 - Fewer storage objects (indexes, MQTs) required



Lab tests – YMMV

© 2013 IBM Corporation

Business Value 3: SIMPLICITY and Time to Value!



“Super Fast, Super Easy” – Just 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. Statistics collection
9. Add Optimizer hints

Repeat



DB2 with BLU Acceleration

1. Create Table
2. Load data

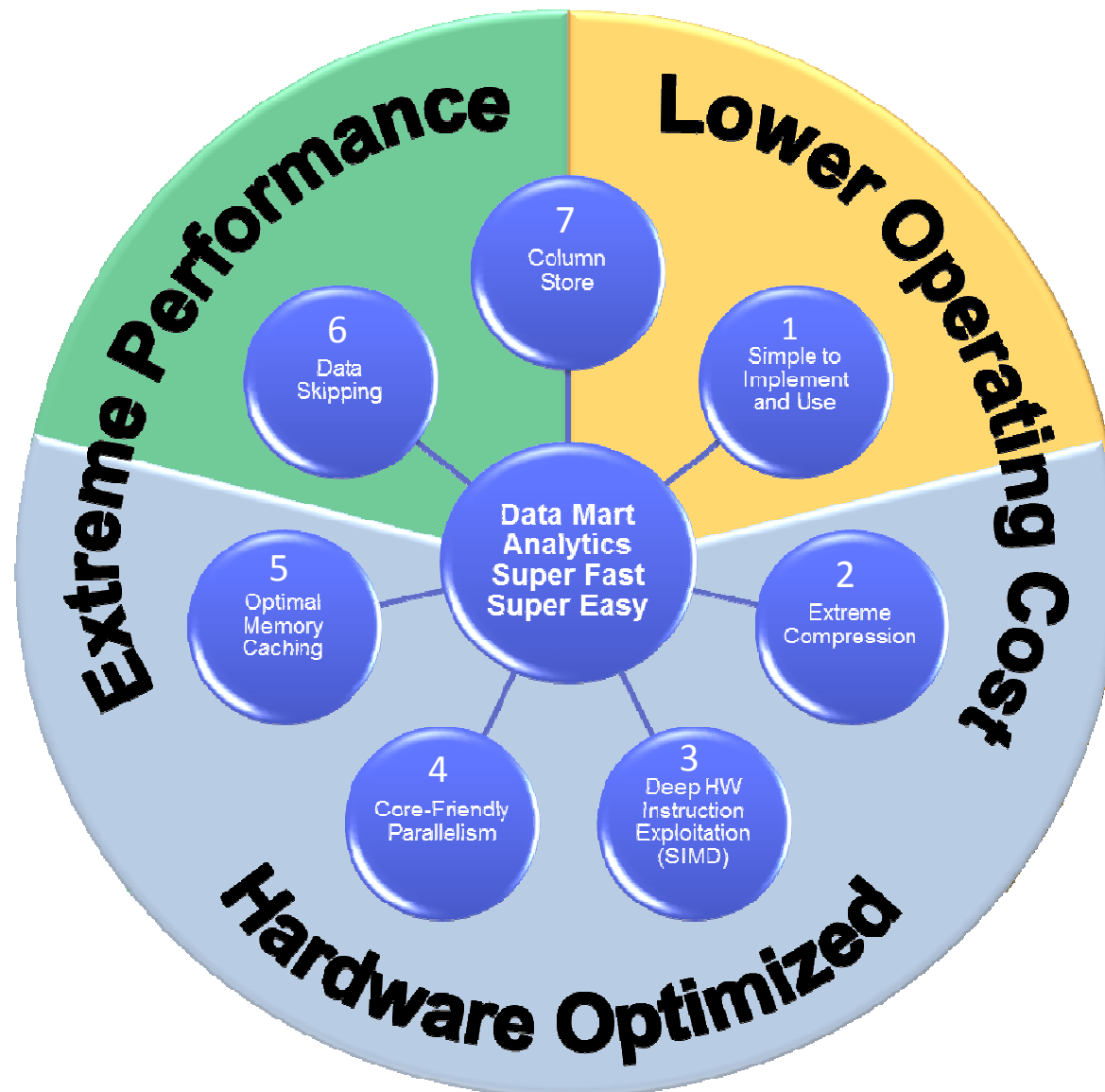


Part 3:

BLU's "Secret Sauce":

7 Big Ideas

DB2 with BLU Acceleration: The 7 Big Ideas



7 Big Ideas: 1 Simple to Implement and Use



- **Always access data with simple scans, done well**
- **LOAD and then... run queries**
 - Significantly reduced or no need for ...
 - Indexes
 - REORG (it's automated)
 - RUNSTATS (it's automated)
 - MDC or Materialized Query Tables (MQTs)
 - Statistical views
 - Optimizer profiles / guidelines
- **No external changes**
 - Same SQL, language interfaces, administration
 - Same DB2 process model, storage, bufferpools
 - No need to change the SQL in your apps!



“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: 2 Operate on Compressed Values

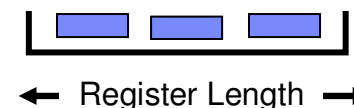


- **Frequency compression (approximate Huffman encoding) exploits skew**
 - The more frequent the value, the fewer bits it is encoded with
 - For example, typically a few populous states may dominate the number of sales
 - New York and California may be encoded with only 1 or 2 bits
 - Alaska and Rhode Island may be encoded in 6 bits

Conceptual
Compression
Dictionary

| STATE | Encoding |
|------------|--|
| New York |  |
| California |  |
| Illinois |  |
| Michigan |  |
| Florida |  |
| Alaska |  |
| Rhode Isl |  |

- **Perform SQL Operations on the Encoded Data!**
 - Apply predicates ($=$, $<$, $>$, \geq , \leq , $<>$, BETWEEN, IN, etc.)
 - Perform joins & grouping

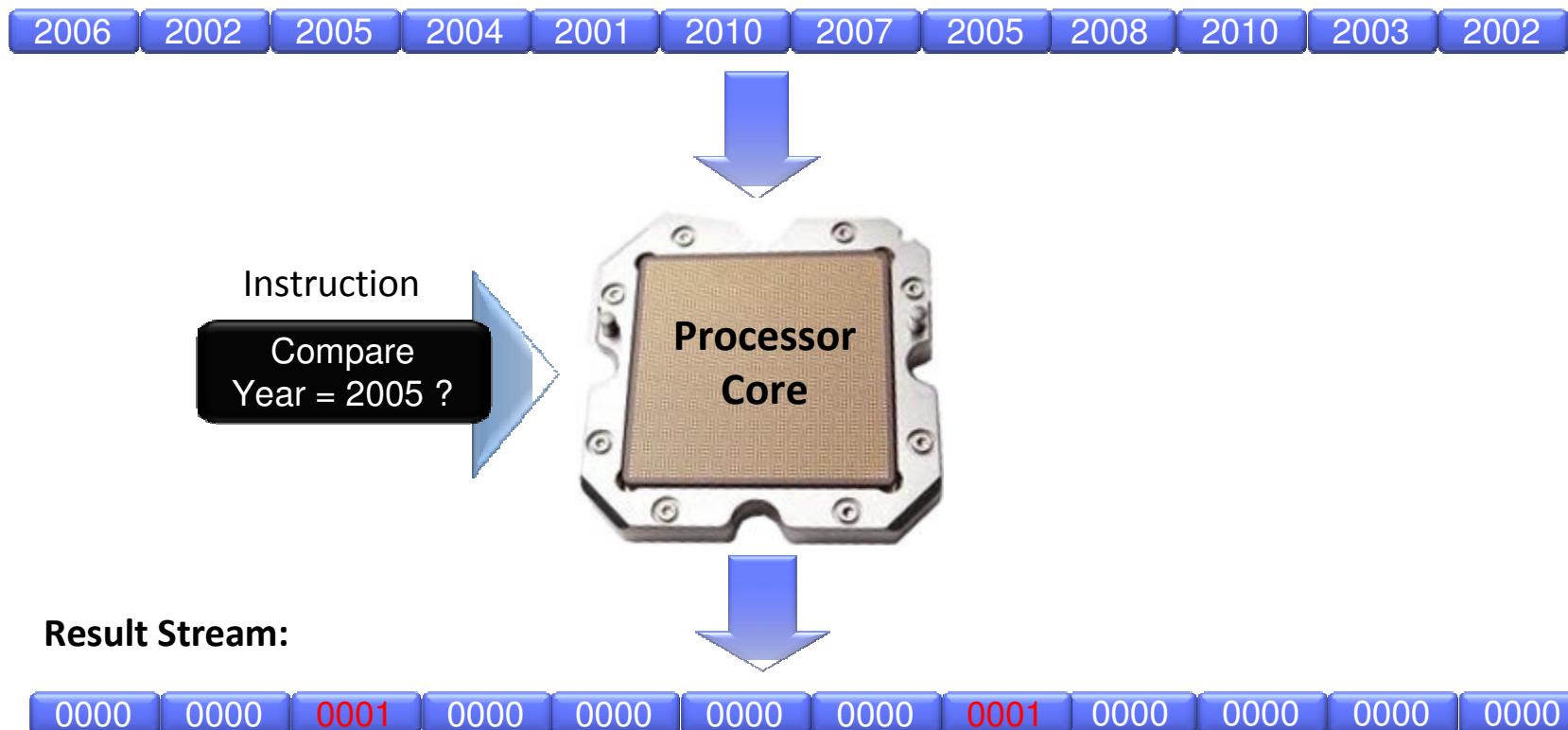


- **Encoded data is smaller, uses less machine resources**
 - Encoded values packed together densely in register-width chunks
 - Fewer I/Os, better memory & cache utilization, fewer CPU cycles to process

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

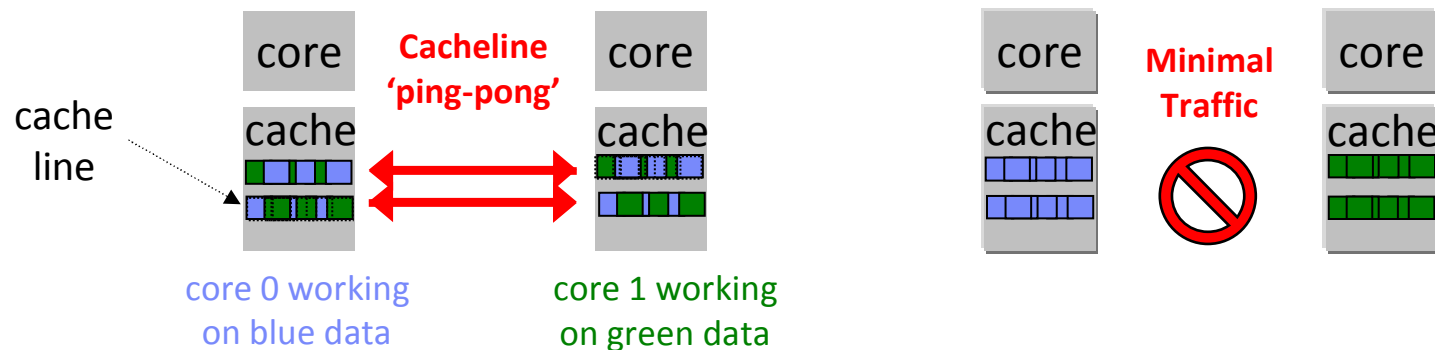


- Apply a single vector instruction to many data elements simultaneously
- Called Single Instruction Multiple Data (SIMD)
- Can be used in predicate evaluation, joins, grouping, arithmetic
- **Example: Compare Year values to 2005**



7 Big Ideas: 4 Core- & Cache-Friendly Parallelism

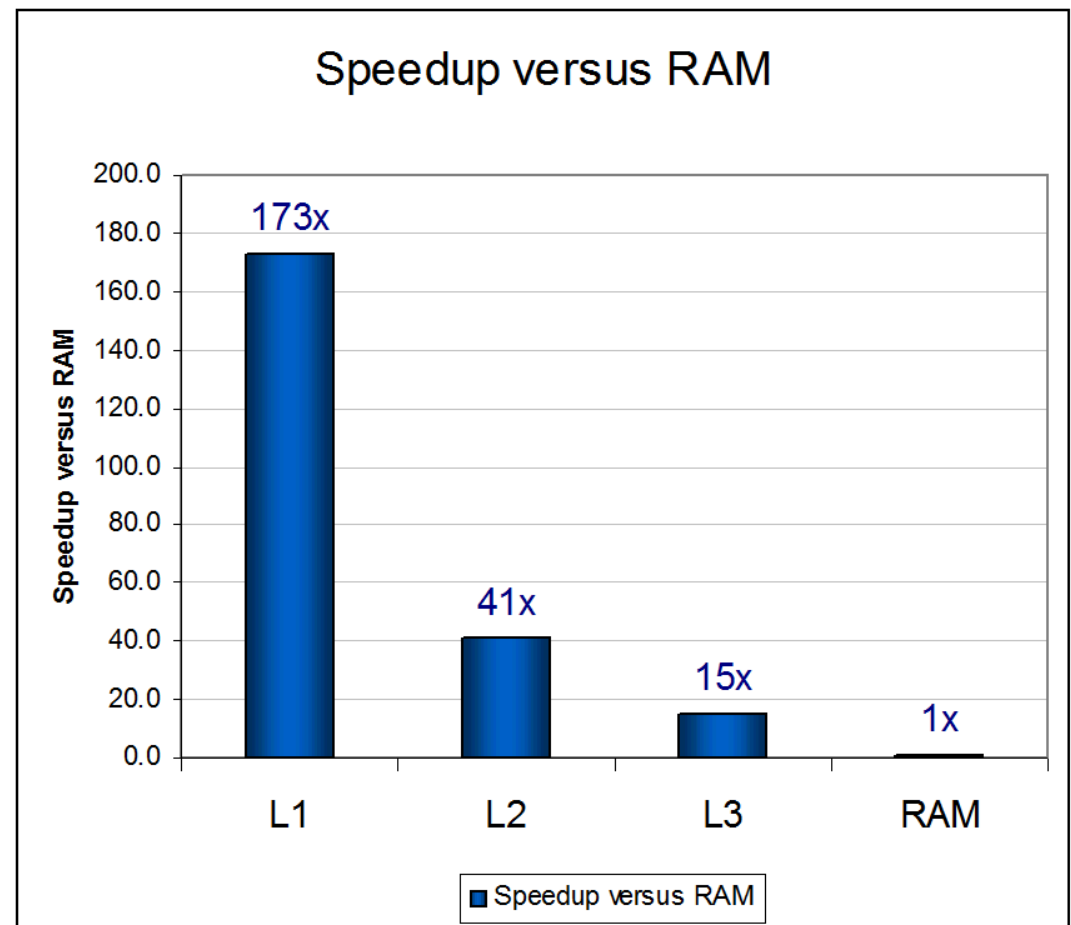
- **BLU's legacy: main-memory DBMS**
- **BLU's run-time was built from the ground up to automatically:**
 - Exploit multi-core parallelism within queries
 - Minimize sharing of common data structures, to minimize latching
 - Pay careful attention to physical attributes of the server, e.g. cache sizes
- **Maximize CPU cache hit rate & cache-line efficiency**



Why The Focus on Cache... It's Super-Fast!

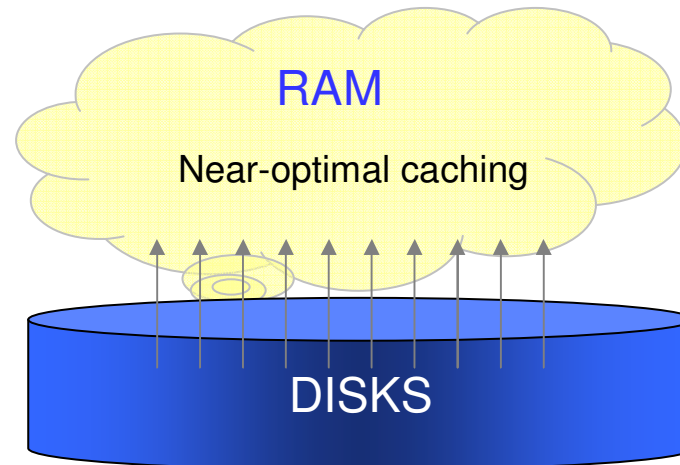


- CPU cache is **many times faster** than DRAM
- BLU's run-time carefully designed to:
 - Auto-detect HW cache sizes
 - Adapt algorithms to them
 - Partition data into cache-sized blocks
 - Exploit L2 & L3 caches
 - Minimize cache-line misses (to DRAM)



7 Big Ideas: 5 *Scan-Friendly Memory Caching* IBM

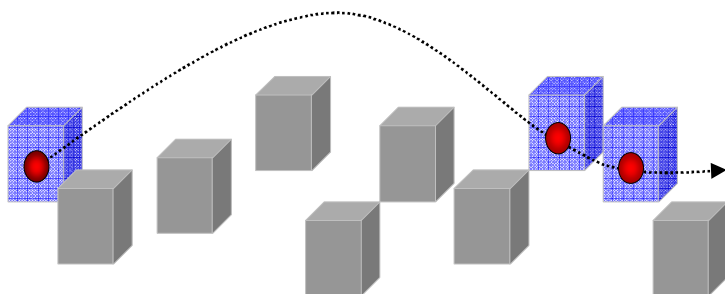
- **BLU run-time optimized for scans that scream!**
- **Memory-optimized (not “In-Memory”)**
 - No need to ensure all data fits in memory
- **New scan-friendly victim selection keeps near-optimal % of pages buffered in memory**
 - Traditional DBMSs use ‘Most Recently Used’ victim selection for large scans
 - “There’s no hope of caching everything, so just victimize the last page read.”
 - BLU design point is to run well when all data fits in memory, and when it doesn’t!
 - New algorithm that adaptively computes target hit ratio for the current scan
 - Based on bufferpool size, frequency of pages being re-accessed in same scan, etc.
 - **Benefit: less disk I/O !**



7 Big Ideas: 6 Data skipping



- Automatic detection of large portions of data that
 - Do not qualify for a query, and therefore...
 - 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
 - Skip scanning data any partition not having literal value in its dictionary



*“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

7 Big Ideas: 7 *Column-Oriented Storage*

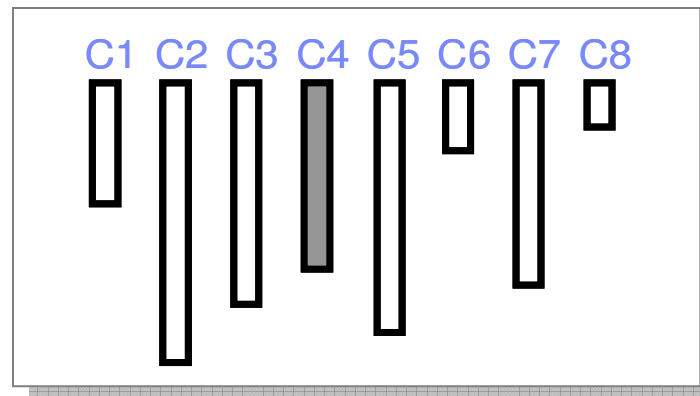


Each column stored and accessed separately, achieving...

- Massive improvements in I/O efficiency
 - Only perform I/O on the columns involved in the query
 - No need to consume bandwidth or buffer pool for unreferenced columns
 - Late materialization of columns possible, after predicates & joins applied
 - Deeper compression possible due to commonality within column values
- Massive improvements in memory and cache efficiency
 - Data pre-packed into cache-friendly structures (no reformatting needed!)
 - Columnar data kept compressed in memory & cache
 - Predicates, joins, scans, etc. all operate on compressed columns
 - Rows are not materialized until absolutely necessary to build result set
 - No need to consume memory, cache space, & bandwidth for unneeded columns

EXAMPLE:

SELECT C4 ... WHERE C4=X

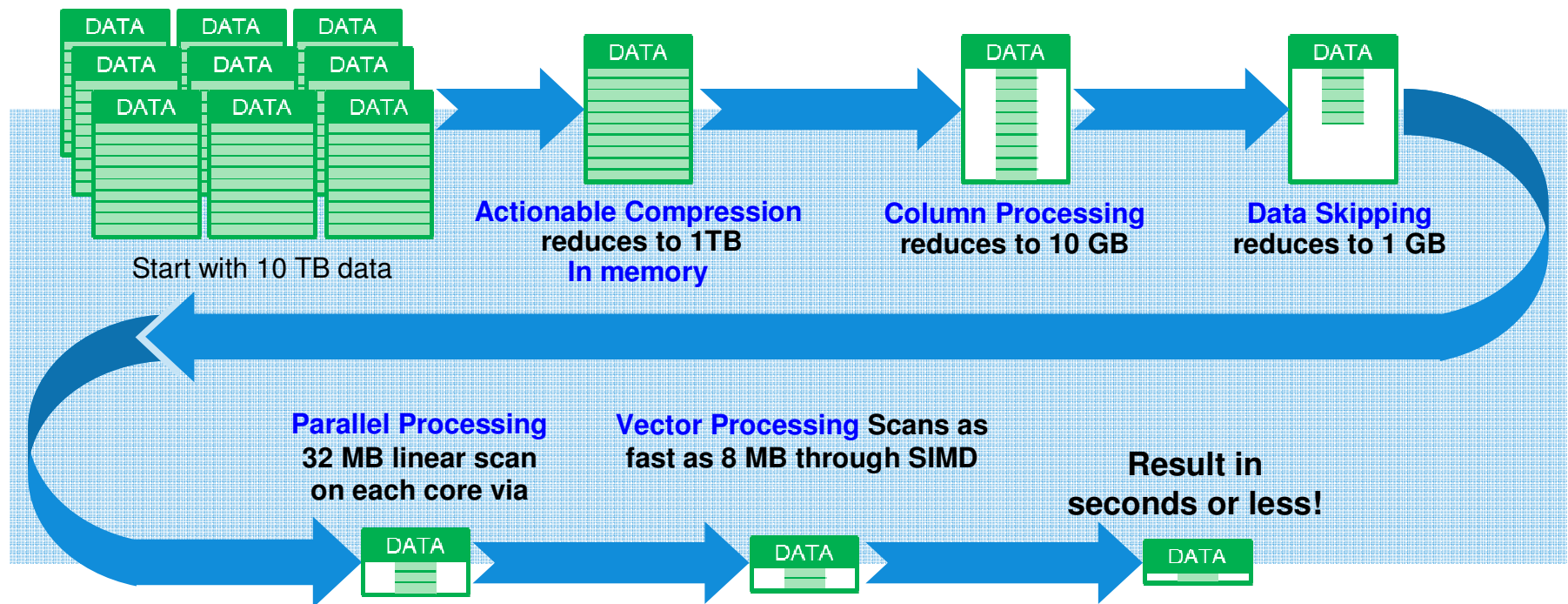


Consumes I/O bandwidth,
memory buffers, and
memory bandwidth
only for C4

How the 7 Big Ideas are Synergistic

10TB query in seconds or less

- **The System:** 32 cores, 1 TB memory, 10 TB 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: each CPU core examines the equivalent of just 8 MB of data!



Part 4:

Behind the Curtain

Columnar storage in DB2 with BLU Acceleration (conceptual)

TSN = Tuple Sequence Number

| | | | | | | | |
|-----|-----------------|----|----------------------------|-------------|----|-------|--------|
| 0 | John Piconne | 47 | 18 Main Street | Springfield | MA | 01111 | |
| 1 | Susan Nakagawa | 32 | 455 N. 1 st St. | San Jose | CA | 95113 | |
| 2 | Sam Gerstner | 55 | 911 Elm St. | Toledo | OH | 43601 | ← page |
| 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 | ← page |
| 10 | | | | | | | |
| 11 | | | | | | | |
| ... | | | | | | | |

- Separate set of extents and pages for each column, stored as “vectors”
- TSNs indicate which column values belong together as a logical “row”

1. Dictionary Coding

- Direct mapping from values to compressed codes via dictionary
- Best for columns with few distinct values

2. Minus Coding

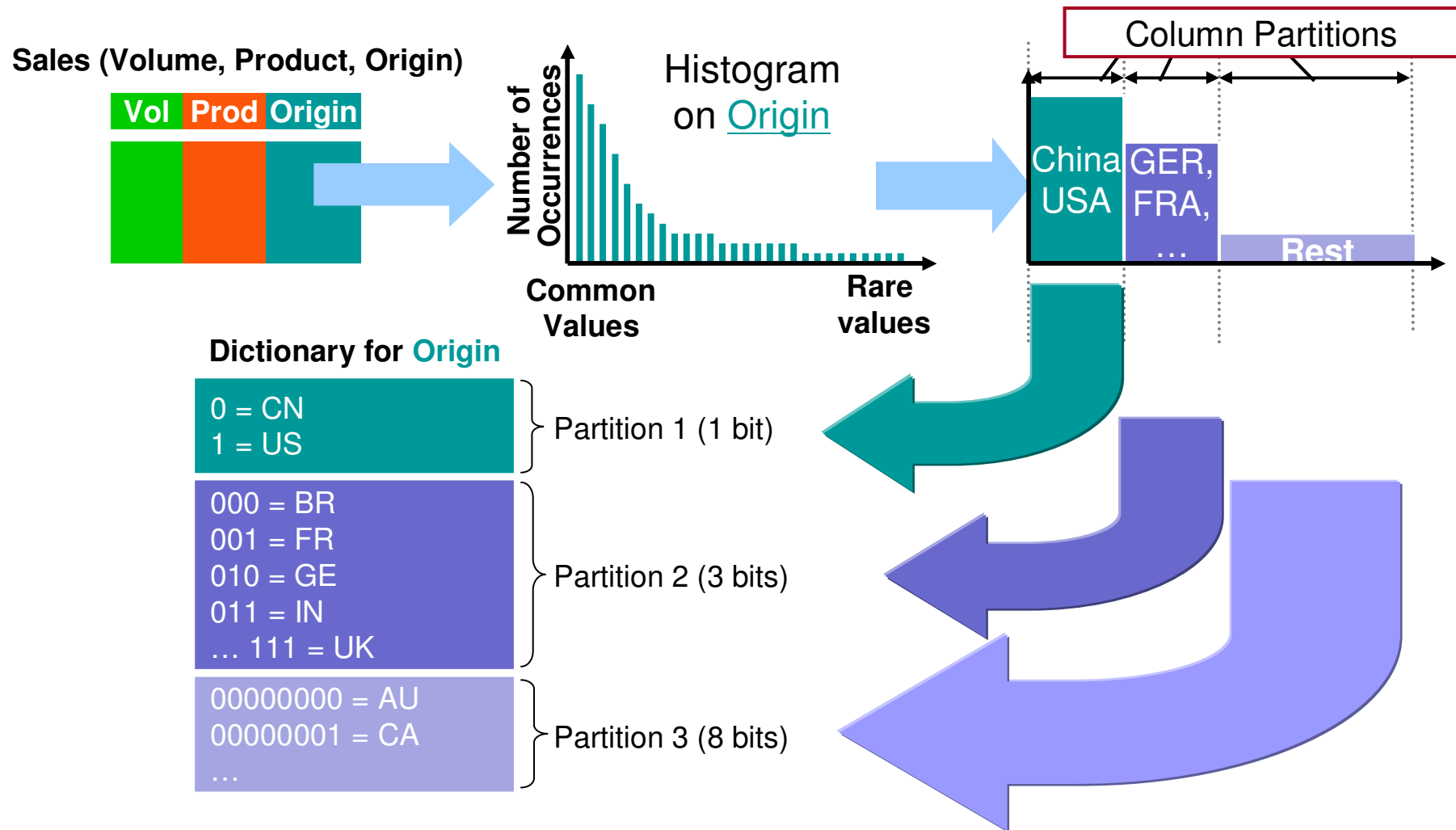
- Encoded as the delta from a single base value
- Applied to types stored internally as INTEGER types
 - Including DECIMAL, DATE, TIME, and TIMESTAMP

3. Prefix Coding

- Most common bit-sized prefixes of the data values
- Compressed codes are the index of the dictionary entry + the remainder (suffix) of the data value.
- Applied to all data types, including short strings
 - Including CHAR, VARCHAR, FLOAT, and DOUBLE

Best method is decided automatically by BLU

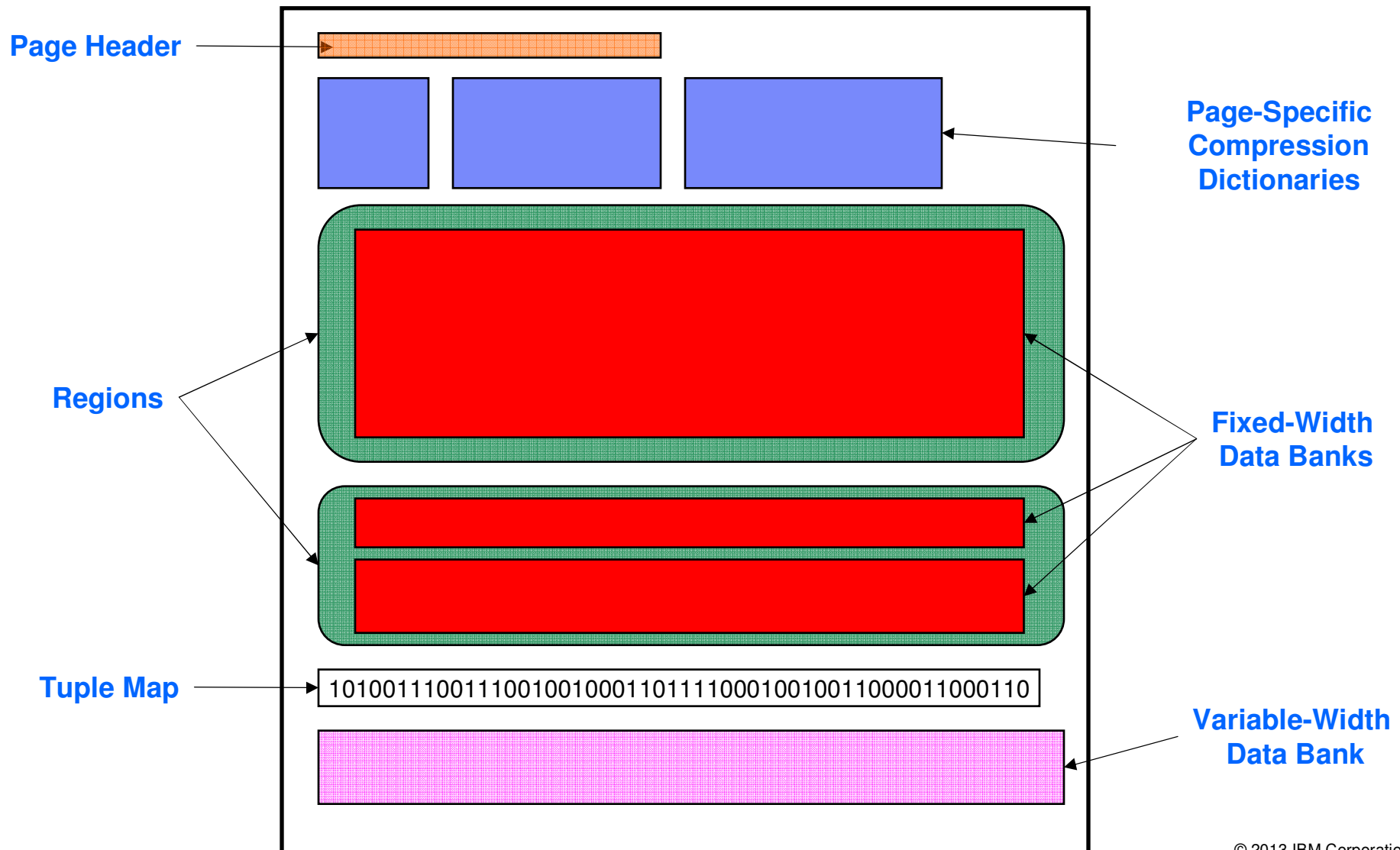
Frequency (Dictionary) Compression



NOTE: Within each partition, dictionary codes are:

- Fixed in length!
- Order-preserving!

Page Format



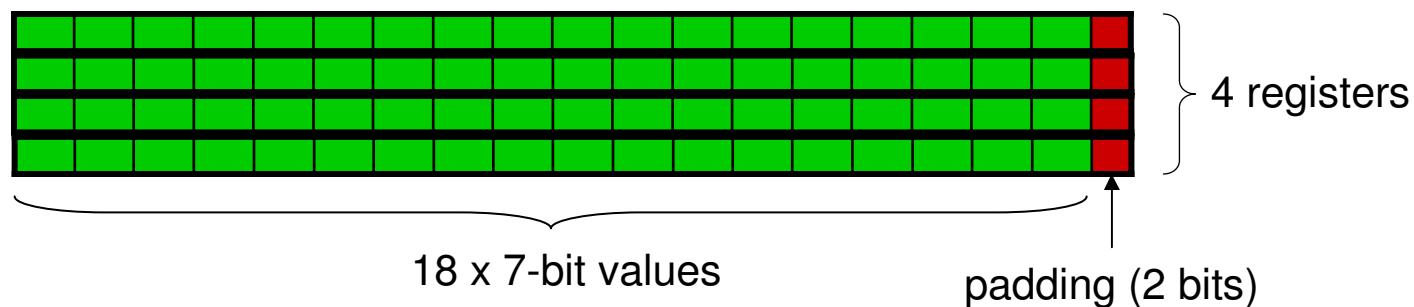
Multi-Value Layout for Compressed Data



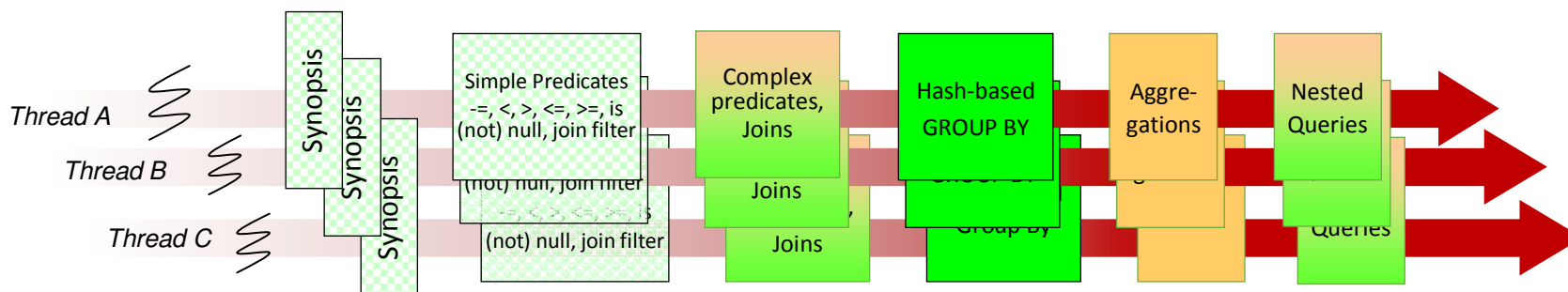
- Data is stored on disk as it will be in memory (No copy! No re-formatting!)
- Column values packed on bit (not byte!) boundaries,
 - As many as fit in machine register
 - Register is 128 or 256 bits
- Padded so that no value spans a register
- Very fast, parallel predicate evaluation applies to multiple values simultaneously

Example: 7-bit column values

- Can fit 18 of these in a 128-bit register
- Requires 2 bits of padding ($128 - 18 \times 7 = 128 - 126 = 2$)



Query Processing



- All queries done as a series of **scans**, one table at a time

- Multi-core and multi-socket optimized

- Built for parallel execution from the ground up
- Each thread operates on separate subset of data
- Separate data structures for minimal synchronization


- Columnar, compressed, & vectorized all the way through


- Most predicates (=, <>, range, NULL tests, IN-lists, etc.)
 - “Fast path” exploits SIMD evaluation
- Grouping and Join done via hash tables
- Only expressions & aggregation require decompressing

- Sophisticated, cache-conscious, scalable operators

- Memory- & cache-optimized

- Compaction of intermediate values
- Partitioning to fit in caches
- Novel compacted hash tables
- NUMA-aware, aggressive prefetching to reduce memory stalls

 = “fast path”

 = operate on encoded values

 = must decode values (math exprs.)

- Processor-optimized

POWER7:

- Apply predicates in 12 instructions
- Employ specialized bit permutation instructions, etc..
- Exploit high memory bandwidth

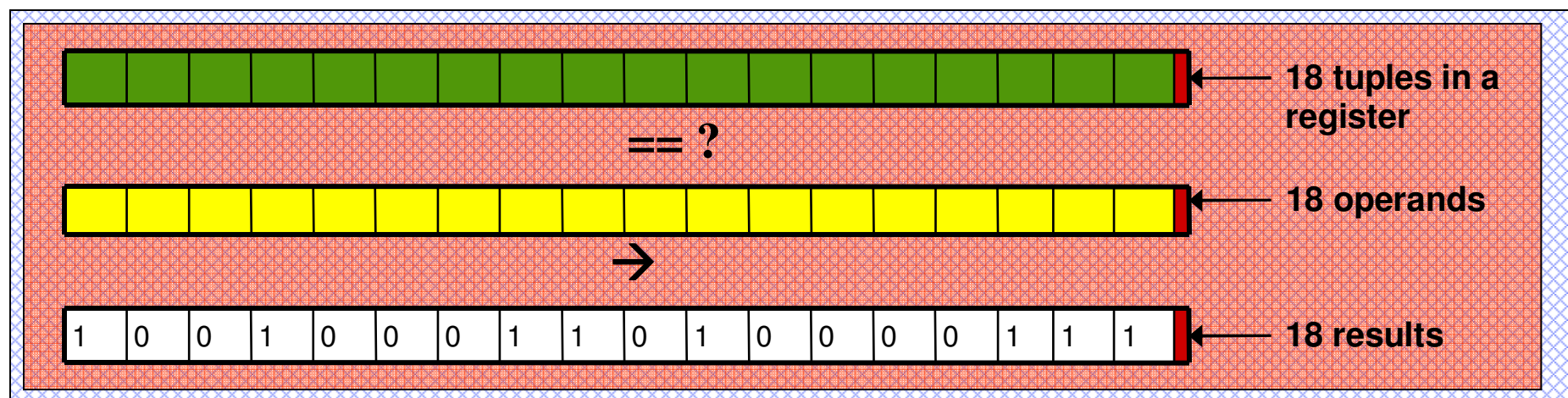
Intel x86_64:

- SSE4.2, AVX

Scan Operators



- **LeafPredicate:** Apply predicate(range of rows) → bit vector of results
 - Applies local (non-join) predicates while scanning within a page
 - For example: (A>5 OR (A IS NULL AND B<13) OR (C+D > 4 AND D == 3))
 - All range predicates converted to LowerValue <= Column <= UpperValue *on codes*
 - **Software SIMD:** simultaneously apply predicate on all tuples that fit in a register
 - **Hardware SIMD** on top of software SIMD: (work on two 64-bit words in parallel)
- **LoadCol:** load additional columns, for tuples passing previous predicate
- **Example:**



Synopsis Tables (used for “Data Skipping”)

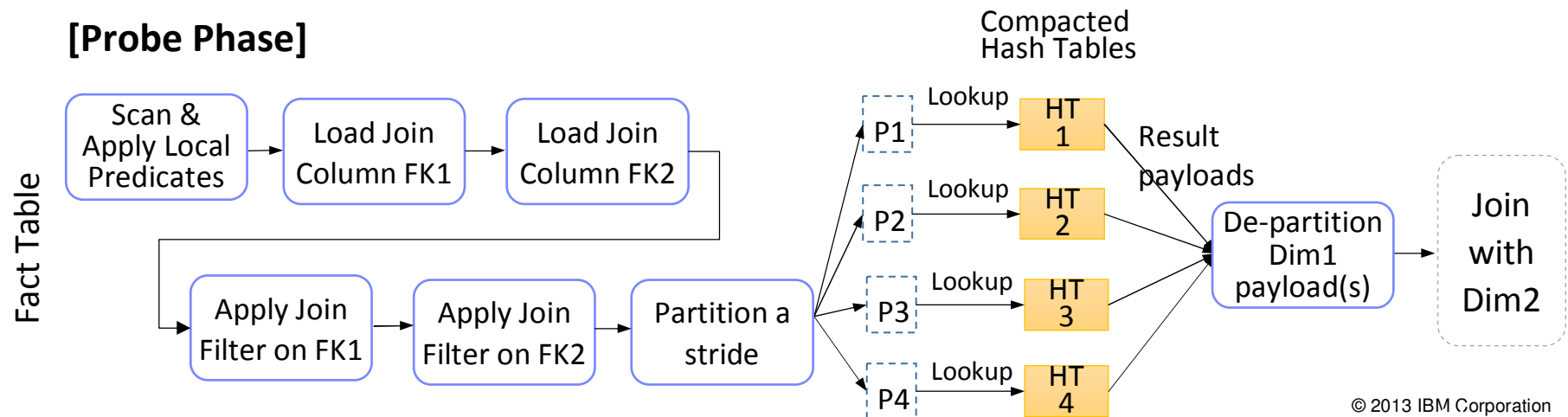
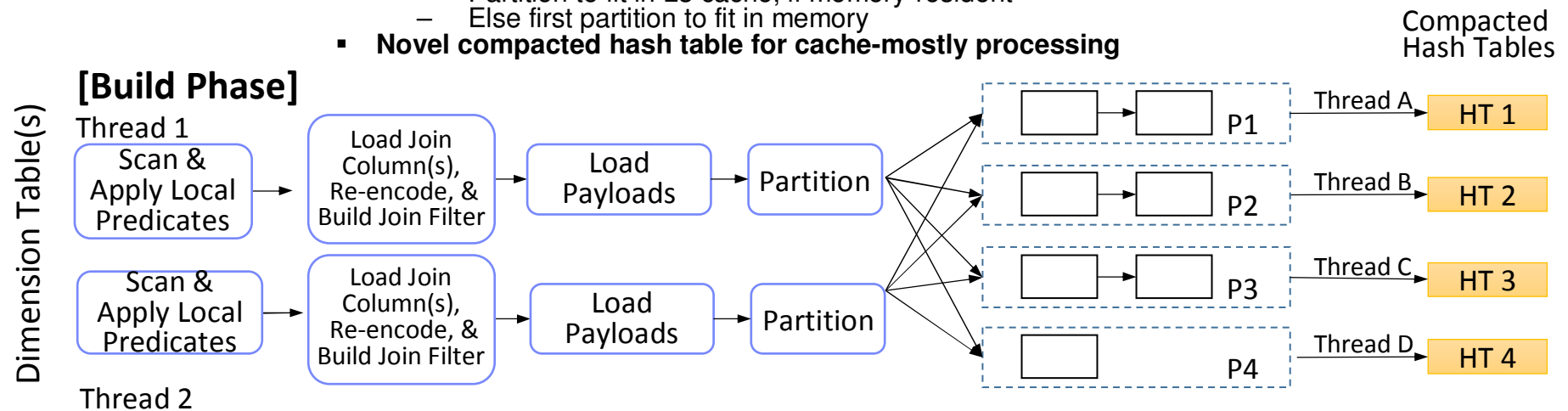


- Created and populated automatically for each BLU table during LOAD
- Contain the min/max values for each column of its source table for fixed # of rows
 - **Example:** One synopsis row for every 1000 rows of the source table.
- Usually cached in RAM (automatically)
- **Purpose:** avoid I/O and scans for pages that cannot satisfy predicates of a query
- **Contains two extra columns:**
 1. **minTSN** – The starting TSN of the tuple interval of the source table
 2. **maxTSN** – The ending TSN of the tuple interval of the source table
- Use the same compression dictionary as the source table
- Query processing:
 1. **Synopsis table is scanned first**
 - Applying predicates of the original query
 - Results are ranges of qualifying rows (TSNs)
 2. **Source table is scanned second,**
 - Only those pages that contain ranges of qualifying TSNs from the synopsis scan.
 3. **Page map** is used to locate the needed pages.

Joins



- **BLU supports all**
 - SQL join types (inner-, LEFT OUTER, RIGHT OUTER, ANTI-, ...)
 - Data types (VARCHARs, trailing blanks,...)
- **No assumption that anything fits in memory, including inners**
 - Partition to fit in L3 cache, if memory-resident
 - Else first partition to fit in memory
- **Novel compacted hash table for cache-mostly processing**

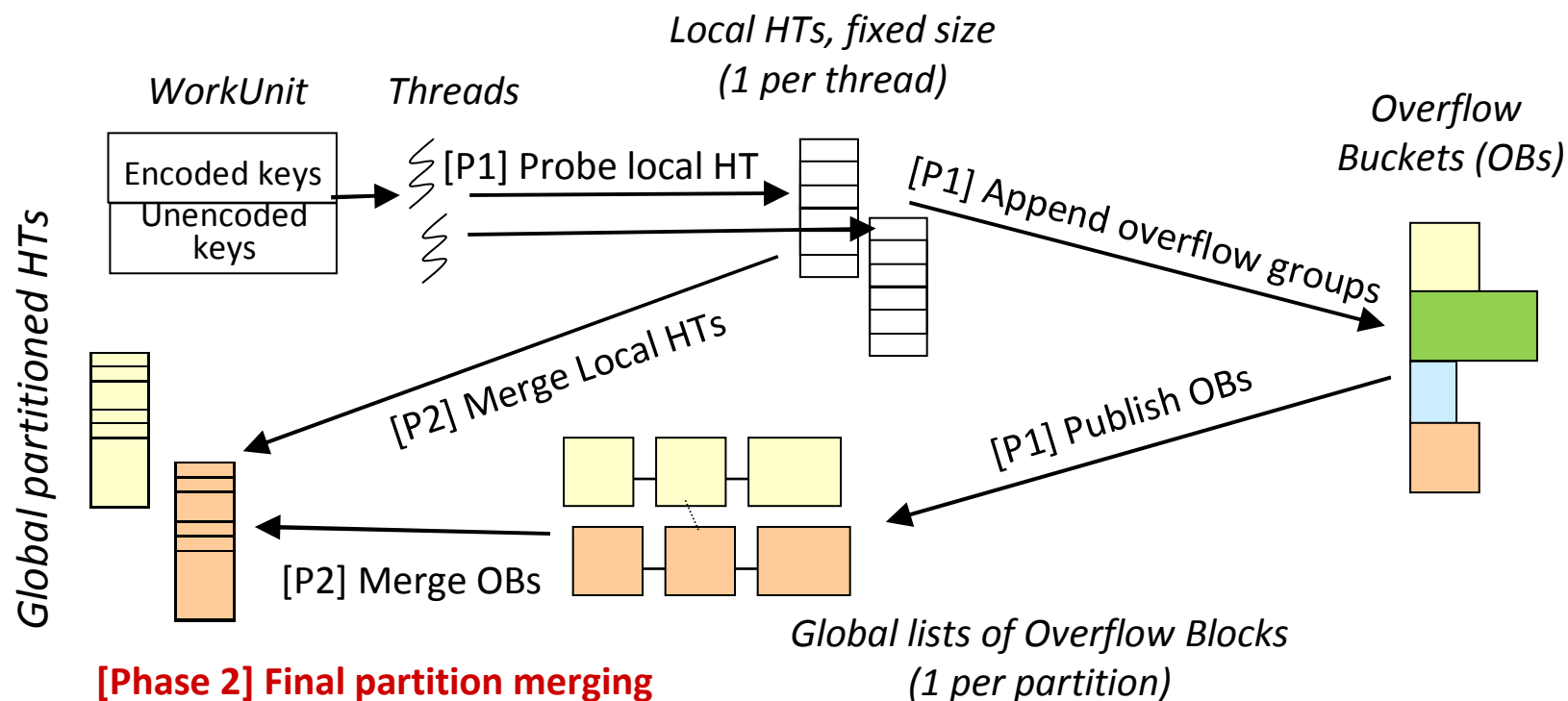


Group By / Aggregation



- Need to perform well on queries that output from few tens to billions of groups
- Cache- and NUMA*-aware (* Non-Uniform Memory Architecture)

[Phase 1] Local Hash Table (HT) probes and appends to Overflow Buckets (OBs)



What About Updates?



- **BLU tables may be updated with UPDATE, DELETE, and INSERT commands**
- **Changes made directly to BLU (column-organized) tables**
 - No row-organized staging tables, unlike SAP HANA and SQL Server
- **Multi-versioning – no in-place updates!**
- **Maintains DB2's usual Isolation, Concurrency Control, and Durability**
 - Fully logged, so recoverable
 - Supported:
 - Isolation Levels: CS + CC, UR
 - Searched UPDATE & DELETE, INSERT from VALUES, INSERT from sub-select
 - Not supported:
 - Positioned update & delete (by cursor), MERGE, select-from-UDI, update & delete of UNION views
- **Insert speed on par with row-organized tables**
 - Sometimes faster, because much fewer (or no) indexes
 - Best performance for large transactions, to amortize logging overheads
 - INSERTing or UPDATEing 100s or 1000s of rows, or more
 - DELETEing, if the clustering of pages matches that of the DELETE (e.g., time)
- **New values compressed with page-level dictionaries, if beneficial**
 - In addition to (on top of) column-level dictionary

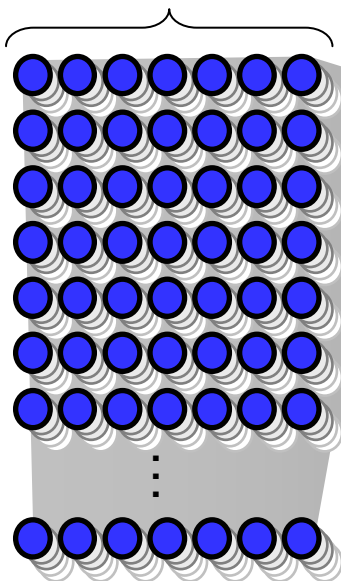
Automatic Workload Management



- Even with a single query, BLU tends to utilize most machine resources
- 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
- Ensures maximum exploitation of parallelism while minimizing thrashing

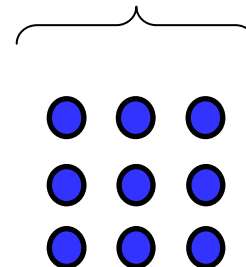
Applications and Users

Up to tens of thousands of SQL queries at once



DB2 DBMS kernel

Moderate number of queries consume resources



Conclusion



BLU Acceleration provides 3 key benefits:

- **Fast**
 - Unprecedented performance for analytic workloads, often 8x to 25x faster
 - Examples of workloads > 100x
 - Examples of individual queries > 1000x
- **Small**
 - Better column-oriented compression
 - Less space required for auxiliary data structures
 - 10x savings vs. uncompressed row tables is common.
- **Simple**
 - Much less tuning needed, due to
 - No secondary indexes or MQTs needed
 - Automation of statistics collection, workload management, space reclamation, etc.
 - More predictable and reliable performance
 - 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

धन्यवाद

Hindi

Ευχαριστώ!

Greek

ขอบคุณ

Thai

Спасибо

Russian

多謝

Traditional Chinese

Gracias

Spanish

شكراً

Arabic

Thank You

English

Obrigado

Brazilian Portuguese

Danke

German

Grazie

Italian

多谢

Simplified Chinese

Merci

French

நன்றி

Tamil

감사합니다

Korean

ありがとうございました

Japanese

For more information on DB2 10.5 with BLU Acceleration

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/10lcf2j
- 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/10z83Al

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

Handelsbanken

“We were very impressed with the performance and simplicity of BLU. We found that some queries achieved an almost 100x speed up with literally no tuning!”

- Lennart Henäng, IT Architect, Handelsbanken



*“One table that's used in our production environment went from about **7 GB** to just over **1/2 GB** after it was converted into a column-organized table. It's kind of scary to think that with these compression levels, I could almost place some of the same workloads that run our business into memory, on my laptop!”*

-Andrew Juarez, Lead SAP Basis and DBA



*“My largest row-organized, adaptive compressed table gave me 3.2x storage savings. However, **converting this row-organized uncompressed table to a column-organized table in DB2 10.5 delivered a massive 15.4x savings!**”*

- Iqbal Goralwalla, Head of DB2 Managed Services, Triton

Part 4: Getting Started with BLU Acceleration

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.

db2set DB2_WORKLOAD=ANALYTICS

- For a new database:
 - 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 buffer pools are large



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.

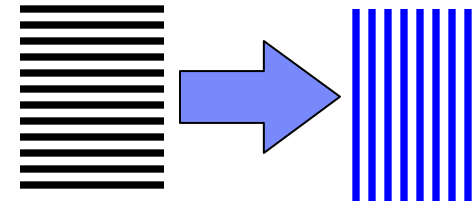
- If dft_table_org = COLUMN (or DB2_WORKLOAD= ANALYTICS):
 - ORGANIZE BY COLUMN is the default and can be omitted
 - Use ORGANIZE BY ROW to create row-organized tables
- **Do NOT**
 - Specify compression, MDC, or partitioning for BLU tables.
 - Create indexes or MQTs.

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;
```

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)
...
```