Modern Databases: BLINK

Themis Palpanas
University of Paris

Data Intensive and Knowledge Oriented Systems

1

- thanks for slides to
  - Guy M. Lohman

2

This is joint work with:

Ronald Barber, Peter Bendel, Marco Czech, Oliver Draese, Frederick Ho, Namik Hrle, Stratos Idreos, Min-Soo Kim, Oliver Koeth, Jae Gil Lee, Tianchao Tim Lee, Guy Lohman, Konstantinos Morfonios, Keshava Murthy, Lin Qiao, Vijayshankar Raman, Richard Sidle, Knut Stolze, … ***and many more****!*

IBM Almaden Research Center

Blink – Agenda

- **Why and What is Blink**
- **Blink Market – Business Intelligence**
- Blink Architecture
- It's All About Performance!
- What's the Big Deal?
- Behind the Curtain – The Query Engine Technology
- References and Related Work
- Next Steps
- Conclusions

IBM

5

Motivation

- **Today, performance of Business Intelligence (BI) queries is too unpredictable**
  - –When an analyst submits a query, s/he doesn't know whether to:
    - •Wait for the response
    - •Go out for coffee
    - •Go out for dinner
    - •Go home for the night!
  - –Response time depends upon "performance layer" of indexes & materializations
  - –Depends critically on predicting the workload
  - –But BI is inherently ***ad hoc!***

- **Goal of Blink:**
  Predictably Fast (i.e., Interactive) Ad Hoc Querying
  - –**Any** query should run in about the same time
  - –Permit an Analyst to interact with the data

IBM

6

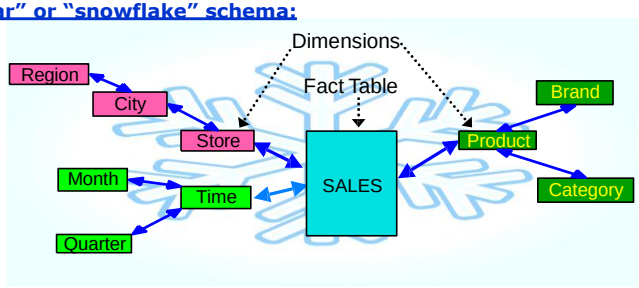
What Is Blink?

- Accelerator technology developed by IBM Almaden Research since 2007
- Contains a compressed copy of a (portion of a) data warehouse
- Exploits:
  - Large main memories
  - Commodity multi-core processors
  - Proprietary compression
- Speeds up typical Business Intelligence SQL queries by **10x to 100x**

- Without requiring tuning of indexes, materialized views, etc.
- **Products offered by IBM based upon Blink:**
  - **IBM Smart Analytics Optimizer for DB2 for z/OS V1.1** – GA'd Nov. 2010
    - Appliance: Runs on zEnterprise Blade eXtension (zBX), network-attached to zEnterprise
  - **Informix Warehouse Accelerator** – GA'd March 2011
    - Virtual Appliance: Runs in same machine as Informix IDS

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 Blink!**

What Blink is Designed For

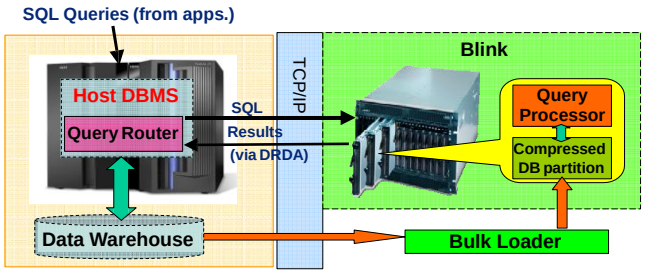
- **OLAP-style SQL queries:**
  - Relational star schema (large **fact table** joined to multiple **dimensions**)
  - Large subset of data warehouse accessed, reduced significantly by...
  - **Aggregations** (SUM, AVG, COUNT) and optional **grouping** (GROUP BY)
  - **Looking for trends or exceptions**
- **EXAMPLE SQL**:

```
SELECT P.Manufacturer, S.Type, SUM(Revenue)
FROM Fact_Sales F
      INNER JOIN Dim_Product P ON F.FKP = P.PK
      INNER JOIN Dim_Store S ON F.FKS = S.PK
      LEFT OUTER JOIN Dim_Time T ON F.FKT = T.PK
WHERE P.Type = ,JEANS' AND S.Size > 50000 AND
  T.Year = 2007
GROUP BY P.Manufacturer, S.Type
```


Blink – Agenda

- Why and What is Blink?
- Blink Market – Business Intelligence
- **Blink Architecture**
- It's All About Performance!
- What's the Big Deal?
- Behind the Curtain – The Query Engine Technology
- References and Related Work
- Next Steps
- Conclusions

Blink Configuration



Host DBMS (DB2 or IDS):

- Routes SQL queries to accelerator
- User need not change SQL or apps.
- No externalized interfaces!
- Can always run query in Host DBMS, e.g., if
  - –too complex SQL, or
  - –too short an est. execution time

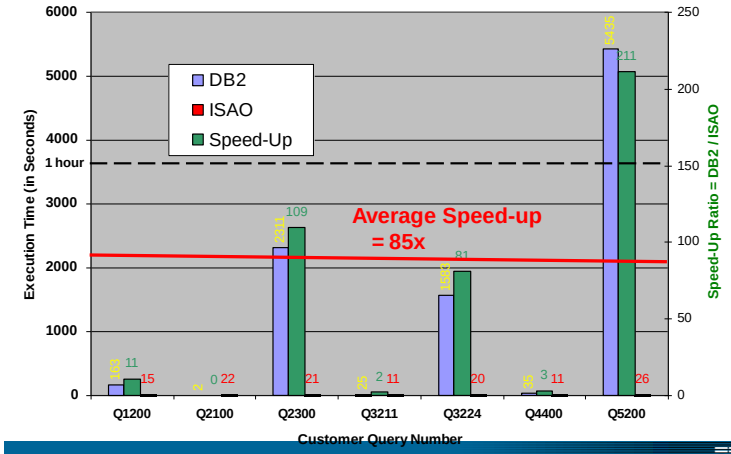
Blink:

- Commodity blades
- Connects to Host DBMS via TCP/IP & DRDA
- Analyzes, compresses, and loads
  - Copy of (portion of) warehouse
  - Partitioned among nodes
- Processes routed SQL query and returns answer to Host DBMS

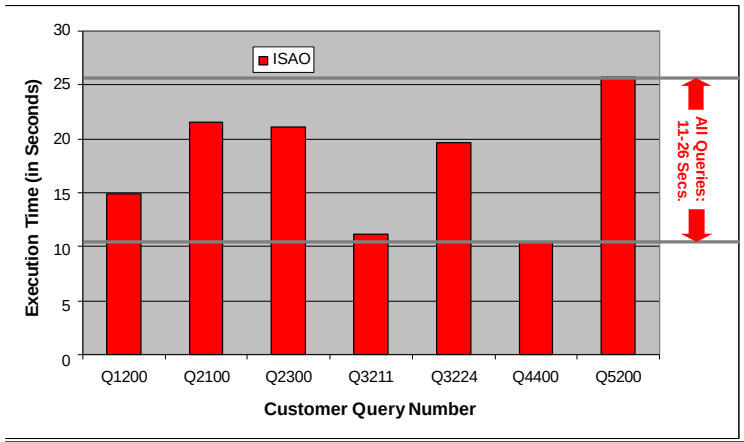
Blink – Agenda

- Why and What is Blink?
- Blink Market – Business Intelligence
- Blink Architecture
- **It's All About Performance!**
- What's the Big Deal?
- Behind the Curtain – The Query Engine Technology
- Related Work
- Conclusions

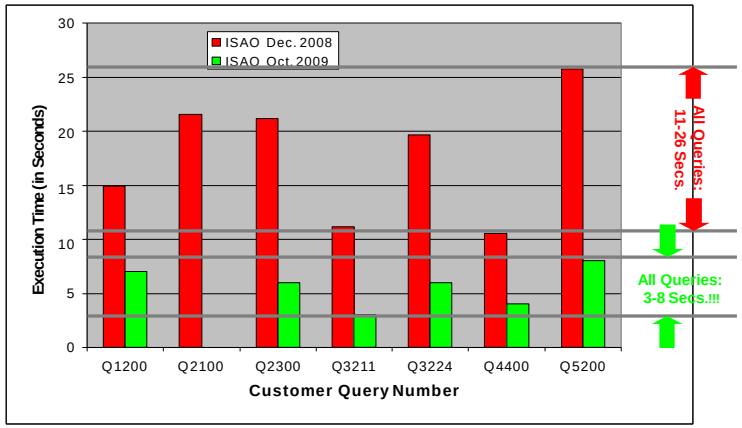
Blink Accelerates Most the Longest-Running Queries



Blink Query Execution times (magnified)

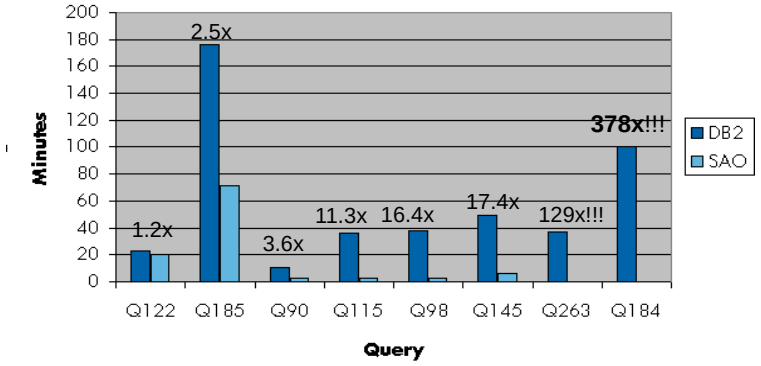


Blink Query Execution times (magnified)



15

Beta Test – Blink Elapsed Time & Speedup



16

Blink – Agenda

- Why and What is Blink?
- Blink Market – Business Intelligence
- Blink Architecture
- It's All About Performance!
- **What's the Big Deal?**
- The Query Engine Technology
- Behind the Curtain – The Query Engine Technology
- References and Related Work
- Next Steps
- Conclusions

IBM

17

What's the Big Deal? What's so Disruptive?

- **Blink rides the wave of hardware technology trends:**
  - –Multi-core processors
  - –Large main memories
  - –Fast interconnects
  - –Increasing latency gap between DRAM and disk

- **Blink disrupts at least 4 major tenets that have been held sacrosanct for over 4 decades!**

IBM

18

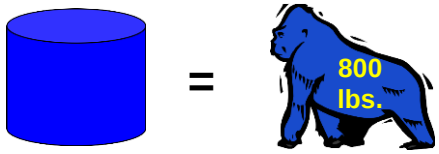
Disruption 1 of 4

- **Tenet #1**: General-purpose DBMSs are most cost-effective
- **Consequence of Tenet #1**: BI pays for OLTP overheads
  - Locking
  - Logging

- **Disruption #1**: Specialized DBMSs for BI now commonplace in market!
- **Consequences of Disruption #1**:
  - BI uses snapshot semantics (typically roll-in or roll-out in batches of rows)
  - Can simplify and eliminate OLTP overheads
  - Can still embed specialty engines in general-purpose DBMS!
    - "Workload-optimized systems"

IBM

Disruption 2 of 4



- **Tenet #2:** Data warehouses are too big for memory
- **Consequence of Tenet #2:** Disk I/O concerns dominate DBMS…
  - Costs
  - Performance
  - Administration efforts
- **Disruption #2:** Huge, cheap main memories (RAM) and flash memories
- **Consequences of Disruption #2:**
  - Portions of warehouse can fit, if partitioned among multiple machines
  - Compression helps!
  - New bottleneck is memory bandwidth (RAM vs L2 cache) and CPU
  - No preferred access path

IBM

Disruption 3 of 4

- **Tenet #3:** Need many Indexes & MQTs for scalable OLAP performance
- **Consequences of Tenet #3:**
  - Need an optimizer to choose among access paths
  - Need a very good wizard to design "performance layer" (expensive!)
  - Must anticipate queries
  - Large time to update performance layer when new data added
- **Disruption #3:** Massive parallelism achieves DB scan in seconds!
  - Arbitrarily partition database among nodes (32–64 GB RAM / node)
  - Exploit multi-core architectures within nodes (1 user or DB cell / core)
- **Consequences of Disruption #3:**
  - Only need to define 1 view in DB2 to satisfy many queries on the accelerator
  - Always scan tables!!
  - Accelerator automatically does equivalent of partition elimination
    - If literal is not in dictionary of that partition
  - Accelerator itself doesn't need
    - Performance layer (indexes or materialized views)!
    - Optimizer!
  - **Simpler! (no need for 4-star wizard)**
  - **Lower TCO!**
  - Consistent response times

IBM

Disruption 4 of 4

- **Tenet #4:** Main-memory DBMSs are the same as a big buffer pool
- **Consequence of Tenet #4:** Don't need a special main-memory DBMS
- **Disruption #4:** Clever engineering can save lots more!
- **Examples of Disruption #4:**
  - Specialized order-preserving & fixed-length compression within partitions permits:
    - Faster access
    - Performing most operations **on encoded values**
      - **Saves CPU for most processing and decoding**
      - **More efficient use of cache and memory bandwidth**
    - **Simultaneous application of predicate conjuncts (1 compare!)**
  - Cache-conscious algorithms make max. use of L2 cache and large registers
  - Exploit multi-core processors
  - Hash-based grouping avoids sorting

IBM

Blink's "Secret Sauce"

1

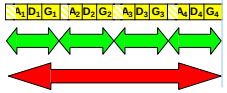
Operate on encoded data

- Dictionary compression with approximate Huffman encoding (fixed length within each part.)
- **Most SQL operations on compressed data!**
- Enables SIMD operations on multiple values in a register
- Dramatically improves efficiency in utilization of RAM, cache, and memory bandwidth

2

Register Store

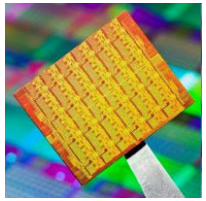
- Pack several column values into a register
- Access only columns referenced in query
- Favors scan-based processing
- L2 / L3 efficiency



3

Parallelism

- KIWI: Kill It With Iron!
- Multiple nodes (blades)
- Designed and built for multi-core, from the ground up



IBM

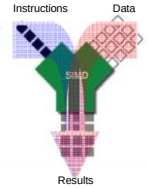
23

Blink's "Secret Sauce"

4

Single Instruction, Multiple Data (SIMD)

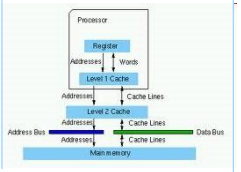
- Enabled by encoded data and register store
- CPU vector processing
- Large gains in CPU efficiency
- 3rd level of parallelism!



5

Architecture-conscious

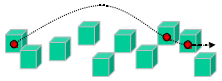
- Cache-conscious query evaluation
- Operate on groups of rows
- Scan-friendly



6

Selection via Synopses

- Skip entire blocks based upon meta-data
- No DBA action to define or use – truly invisible.
- Similar to Netezza's "zonal maps"



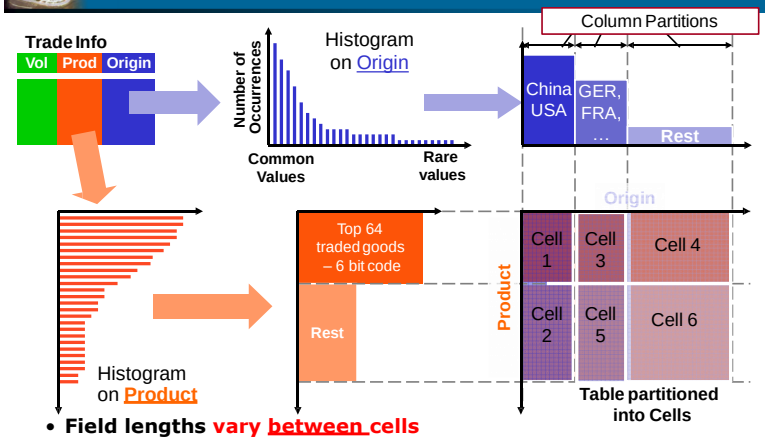
IBM

24

Blink – Agenda

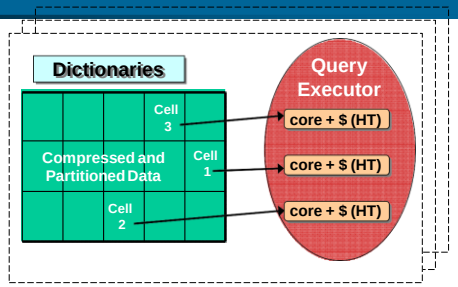
- Why and What is Blink?
- Blink Market – Business Intelligence
- Blink Architecture
- It's All About Performance!
- What's the Big Deal?
- **Behind the Curtain – The Query Engine Technology**
- References and Related Work
- Next Steps
- Conclusions

Compression: Frequency Partitioning



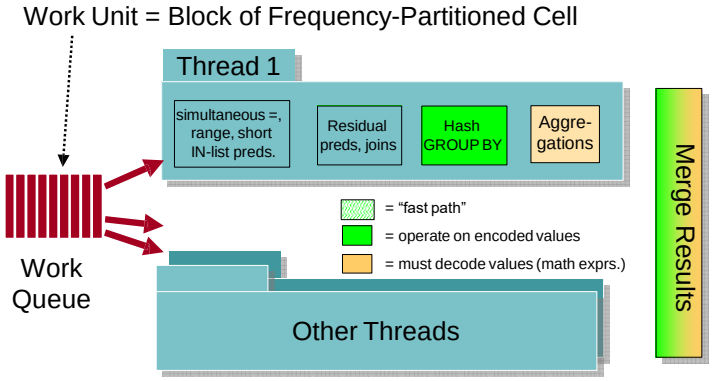
- **Field lengths vary between cells**
  - –Higher Frequencies -> Shorter Codes (Approximate Huffman)
- **Field lengths fixed within cells**

Query Processing

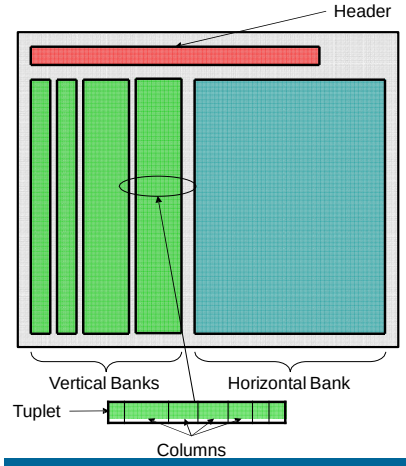


- **Cell is also the unit of processing, each cell…**
  - Assigned to one core
  - Has its own hash table in cache (so no shared object that needs latching!)
- **Main operator: SCAN over compressed, main-memory table**
  - Do selections, GROUP BY, and aggregation as part of this SCAN
  - Only need de-compress for arithmetic operations
- **Response time ∝ (database size) / (# cores x # nodes)**
  - Embarrassing Parallelism – little data exchange across nodes

Fine-Grained Data Parallelism



Blink PAX Data Storage Format – Overview

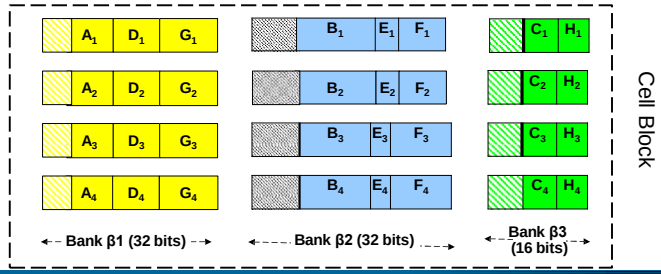


- **Frequency Partitioning into Cells**
  - –Column-wise compression
  - –Each col. dictionary partitioned by value frequency
  - – Cross-product of col. partitions ⓪Cells
  - –Encoded columns are fixed-length (bits) in a cell
- **Cell data are stored in fixed-sized (1MB) Blocks**
- **Rows are partitioned vertically within blocks, into Banks**
  - –Encoded columns are bin-packed into word-sized (8,16,32,64 bit) banks
  - –**Vertical Banks**: contain columns typically used in predicates and grouping
  - –**Horizontal Bank**: contains measure columns
  - – **Access Pattern**:
    - •Scan V-banks to apply predicates.
    - •RID access to V-banks for residual predicates, grouping columns.
    - • RID access to H-bank for aggregation.

29

Banks and Tuplets in Blink

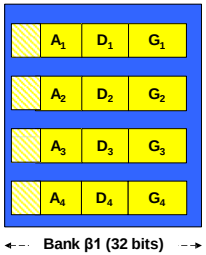
- **A *bank* is a vertical partition of a table, containing a subset of its columns**
  - – Assignment of columns to banks is cell-specific, since column's length
    - Varies from cell to cell, but
    - Is fixed within a cell
  - – Banks contain
    - Concatenations of the fixed-length column codes
    - Padded to the nearest fraction of a word length (8 / 16 / 32 / 64 bits).
    - We call these word-sized units ***tuplets***.
- **Blink's bank-major layouts are a hybrid of row-major and column-major**



30

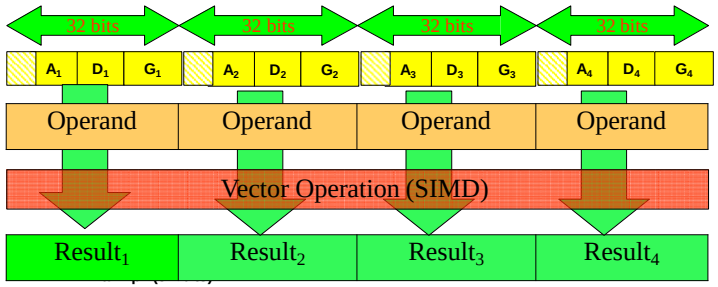
Register Stores Facilitate SIMD Parallelism

- **Access only the banks referenced in the query (like a column store):**
  - **SELECT SUM (T.G)**
  - **FROM T**
  - **WHERE T.A > 5**
  - **GROUP BY T.D**



Register Stores Facilitate SIMD Parallelism

- **Access only the banks referenced in the query (like a column store):**
  - **SELECT SUM (T.G)**
  - **FROM T**
  - **WHERE T.A > 5**
  - **GROUP BY T.D**
- **Pack multiple rows from the same bank into the 128-bit register**
- **Enables yet another layer of parallelism: SIMD** (Single-Instruction, Multiple-Data)!



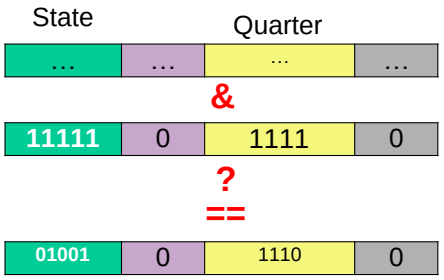
Simultaneous Evaluation of Equality Predicates

- CPU operates on 128-bit units
  - Lots of fields fit in 128 bits
- These fields are at fixed offsets
- Apply predicates to all columns simultaneously!
- Also works for range queries!

State=='CA' && Quarter == '2011Q4'

Translate Value Query to Coded Query

State==01001 && Quarter==1110



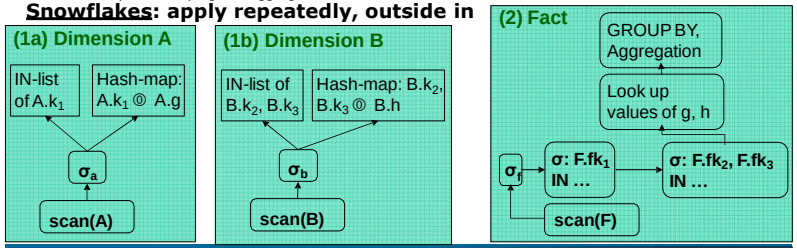
Joins

No shuffle needed:
- Fact table partitioned among nodes and cores
- Dimension tables replicated

Basic idea: Re-write Join as multiple scans:

1. Over each **dimension**, to form:
   - A list of qualifying ***primary keys (PKs),*** decoded
   - A ***hash-map*** of primary key @ ***auxiliary columns*** (those used later in query for GROUP BY, etc.)
2. Over **fact** table:
   - First convert PKs to ***foreign keys (FKs)*** in fact table column
   - Apply as (very big) IN-list predicates (a semi-join), one per dimension
   - Look up into hash-maps to pick up other columns
   - Complete Grouping and Aggregation

Snowflakes: apply repeatedly, outside in



What About Updates?

- Blink uses **snapshot semantics** (batch updates), common in BI
- System maintains a **currentEpoch** number (monotone increasing)
  - Think of it as a batch or version number
  - Prevents seeing incomplete updates, without needing locking
  - Bumped (N++) atomically after each batch of inserts & deletes completes
- Tables have two new columns
  - **startEpoch** – epoch in which that row inserted
  - **endEpoch** – epoch in which that row deleted (initially Infinity)
- Queries are automatically appended with two predicates:
  - startEpoch < currentEpoch AND
  - endEpoch > currentEpoch
- Encoding of updated values
  - If value is in dictionary, use that encoding
  - Otherwise, store unencoded in a special cell, called the "catch-all" cell

Blink – Agenda

- Why and What is Blink?
- Blink Market – Business Intelligence
- Blink Architecture
- It's All About Performance!
- What's the Big Deal?
- Behind the Curtain – The Query Engine Technology
- **References and Related Work**
- **Next Steps**
- Conclusions

Blink Refereed Publications

- **VLDB 2008:** *"Main-Memory Scan Sharing for Multi-core CPUs",* Lin Qiao, Vijayshankar Raman, Frederick Reiss, Peter Haas, Guy Lohman
- **VLDB 2008:** *"Row-Wise Parallel Predicate Evaluation",* Ryan Johnson, Vijayshankar Raman, Richard Sidle, Garret Swart
- **ICDE 2008**: *"Constant-time Query Processing",* Vijayshankar Raman, Garret Swart, Lin Qiao, Frederick Reiss, Vijay Dialani, Donald Kossmann, Inderpal Narang, Richard Sidle
- **SIGMOD 2007**: *"How to barter bits for chronons: compression and bandwidth trade offs for database scans",* Allison L. Holloway, Vijayshankar Raman, Garret Swart, David J. DeWitt
- **VLDB 2006**: *"How to wring a table Dry: Entropy Compression of Relations and Querying Compressed Relations",* Vijayshankar Raman, Garret Swart

Related Work

- **SAP HANA / HYRISE**
  - claim OLTP and BI workloads
  - single copy
  - single node (HYRISE)
- **VectorWise**
  - pure column store, disk-based
  - single copy
  - single node
- **Vertica**
  - pure column store, disk-based
  - projections
  - many (specialized) kinds of compression
- **ParAccel, Exasol** – ??

Next Steps: BLink Ultra (BLU)

- What, you can't afford to put 100 TB in RAM?
  - Relax main-memory-only to disk-based
- You say your dimension table has 2000 columns?
  - Allow and exploit pure column store
- You've got HOW MANY fact tables?
- Yikes, your dimension table is HOW BIG?
  - Allow multiple partitioned tables
  - Need traditional MPP optimization for join ordering
- Yeah, synchronizing multiple copies is a pain.
  - Have Blink store the only copy
- What, you have point queries, too?
  - May need some indexes
- Um, we haven't implemented that yet…
  - Tighter coupling with traditional DBMS

IBM

Summary – Not Your Father's Database!

- Radical changes are happening in hardware
  - Large, cheap memories
  - Multi-core processors promise cheap, massive CPU parallelism
- Blink exploits these trends:
  - Special-purpose accelerator (BI only, snapshot semantics, no transactions)
  - Main-memory DBMS
  - Massive parallelism of commodity multi-core hardware (blade center format)
  - Query processing on compressed values!
  - Cache-conscious algorithms
- Blink speeds up your problem queries the most!
- Blink is an appliance product that is transparent to the user
  - Minimal set-up
  - Applications need not change
  - Tuning not needed!
  - Lower TCO

Questions?

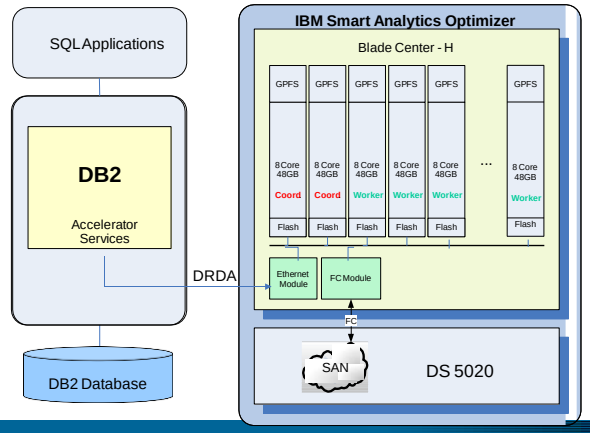
IBM

41

BACKUP SLIDES

42

IBM Smart Analytics Optimizer Architecture

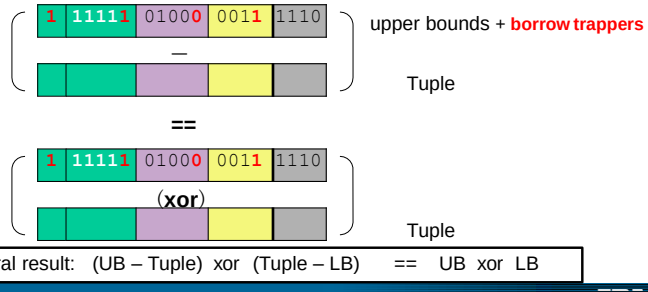


Evaluation of Range Predicates on Encoded Values

B<='CA' and C < 17 and D <= 'Q4'

Translate value query to encoded query (exploits ***order-preserving code***)

A <= 11111 and B<=01000 and C <=0011 and D<=1110



Blink vs. a Column Store

Aspect	Column Store	Blink
Compression	Every column padded to word boundary ◎ more padding/column ◎ worse compression	Multiple columns / word ◎ less padding overhead
Query Processing	▪ Like having an index on every column ▪ To answer query: ▪ Determine list(s) of matching records ▪ Intersect these lists on RID	▪ Can skip blocks based upon predicates ▪ To answer query: ▪ Do table scan
Updating	▪ Insert requires: ▪ Separate updates to every column ◎ Multiple random I/Os, 1/column	▪ Insert requires: ▪ Single update to each bank, 1 / bank ◎ One I/O to one cell block
Evaluation Matches Hardware?	**Evaluation doesn't match w/ Hardware:** ▪ Index navigation involves random accesses ▪ Index navigation involves branches ▪ Predicate evaluation has to be done serially	**Evaluation matches with Hardware** ▪ Scan does sequential memory access ▪ Almost no branches ▪ Simultaneous predicate evaluation ▪ SIMD predicate evaluation