Why do I need columnar storage?

Andrew Hutchings
ColumnStore Engineering Manager
MariaDB Corporation

Who Am I?

- Andrew Hutchings, aka “LinuxJedi”
- Engineering Manager for MariaDB’s ColumnStore
- Previous worked for:
  - NGINX - Senior Developer Advocate / Technical Product Manager
  - HP - Principal Software Engineer (HP Cloud / ATG)
  - SkySQL - Senior Sustaining Engineer
  - Rackspace - Senior Software Engineer
  - Sun/Oracle - MySQL Senior Support Engineer
- Co-author of MySQL 5.1 Plugin Development
- IRC/Twitter: LinuxJedi
- EMail: linuxjedi@mariadb.com

MariaDB

What is Columnar Storage?

MariaDB

Row-oriented vs. Column-oriented format

SELECT Fname FROM Table 1 WHERE State = 'NY'

ID	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F

ID	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F

- **Row oriented**
  - Rows stored sequentially in a file
  - Scans through every record row by row
- **Column oriented**
  - Each column is stored in a separate file
  - Scans the only relevant column

MariaDB

Single-Row Operations - Insert

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F
6	Marvin	Martian	CA	91602	(818) 761-9964	26	M

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F
6	Marvin	Martian	CA	91602	(818) 761-9964	26	M

Row oriented:
new rows appended to the end.

Column oriented:
new value added to each file.

MariaDB

Single-Row Operations - Update

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F

Row oriented:
Update 100% of rows means change 100% of blocks on disk.

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F

Column oriented:
Just update the blocks needed to be updated

Single-Row Operations - Delete

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F
6	Marvin	Martian	CA	91602	(818) 761-9964	26	M

Row oriented:
Entire rows deleted

Key	Fname	Lname	State	Zip	Phone	Age	Sex
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F
6	Marvin	Martian	CA	91602	(818) 761-9964	26	M

Column oriented:
Value deleted from each file

MariaDB

Changing the table structure

Key	Fname	Lname	State	Zip	Phone	Age	Sex	Active
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M	Y
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M	N
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M	N
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M	Y
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F	N

Key	Fname	Lname	State	Zip	Phone	Age	Sex	Active
1	Bugs	Bunny	NY	11217	(718) 938-3235	34	M	Y
2	Yosemite	Sam	CA	95389	(209) 375-6572	52	M	N
3	Daffy	Duck	NY	10013	(212) 227-1810	35	M	N
4	Elmer	Fudd	ME	04578	(207) 882-7323	43	M	Y
5	Witch	Hazel	MA	01970	(978) 744-0991	57	F	N

Row oriented:
Requires rebuilding of the whole table

Column oriented:
Create new file for the new column

MariaDB

So why use it?

Technical Use Case

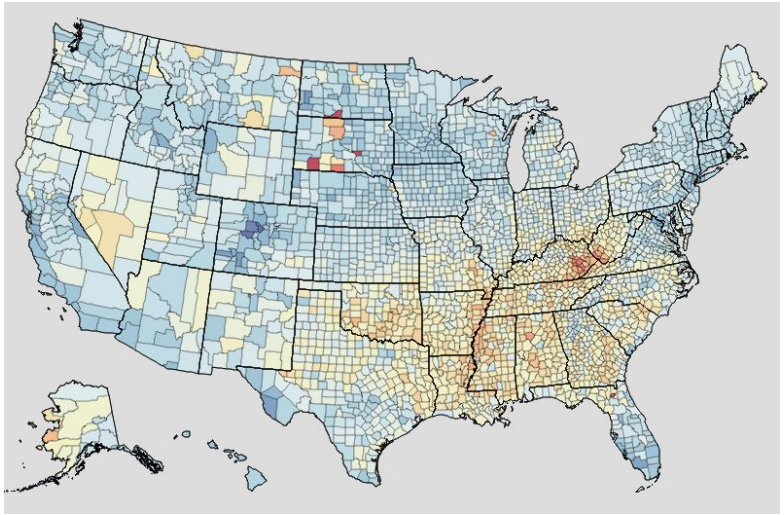
MariaDB ColumnStore

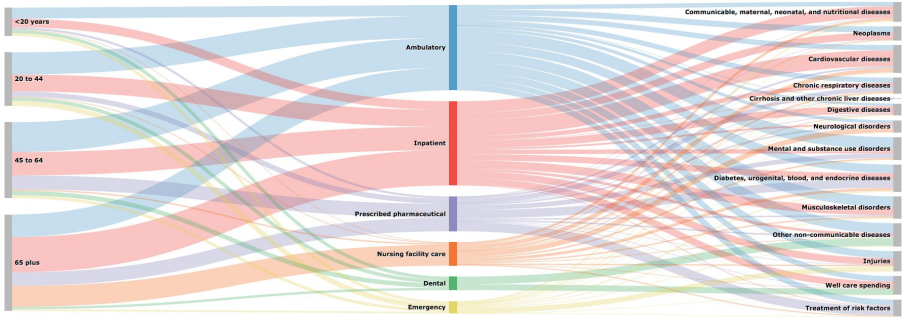
- Very large data sets
  - Many columns
  - Many millions of rows
- Aggregates over large data
- Rapid bulk data insertion
  - The larger the batch the better

Traditional OLTP Engines

- Smaller data sets
- Basic queries
- Lots of DML queries
- Complex data types

Use case: IHME Visualization





IHME Visualizations library: http://www.healthdata.org/results/data-visualizations

MariaDB

About MariaDB ColumnStore

History

- March 2010 - Calpont launches InfiniDB
- September 2014 - Calpont (now itself called InfiniDB) closes down
  - MariaDB (then SkySQL) supports InfiniDB customers
- April 2016 - MariaDB announces development of MariaDB ColumnStore
- August 2016 - I joined MariaDB and jumped straight into ColumnStore
- December 2016 - MariaDB ColumnStore 1.0 GA
  - InfiniDB + MariaDB 10.1 + Many fixes and improvements
- November 2017 - MariaDB ColumnStore 1.1 GA
  - MariaDB 10.2 + APIs + Even more improvements
- December 2018 - MariaDB ColumnStore 1.2 GA
  - MariaDB 10.3 + TIME, microseconds, UDAFs + Lots more
- January 2020 - MariaDB Enterprise 10.4 include ColumnStore 1.4 GA
  - Convergence, more data types, S3 storage, performance improvements

ColumnStore in MariaDB

- Version 1.2 - uses a fork of MariaDB 10.3
- Version 1.4 - included in MariaDB Enterprise 10.4
  - Can be built using MariaDB Community 10.4
- Version 1.5 - to be included in MariaDB Community & Enterprise 10.5

Extent Elimination

Storage Architecture reduces I/O

- Only touch column files that are in filter, projection, group by, and join conditions
- Eliminate disk block touches to partitions outside filter and join conditions

SELECT **Item**, sum(**Quantity**) FROM Orders
WHERE **ShipDate** between '2016-01-01' and '2016-01-31'
GROUP BY **Item**

Extent 1:
ShipDate: 2016-01-12 - 2016-03-05

Extent 2:
ShipDate: 2016-03-05 - 2016-09-23

Extent 3:
ShipDate: 2016-09-24 - 2017-01-06

Horizontal Partition: 8 Million Rows Extent 1

Horizontal Partition: 8 Million Rows Extent 2

Horizontal Partition: 8 Million Rows Extent 3

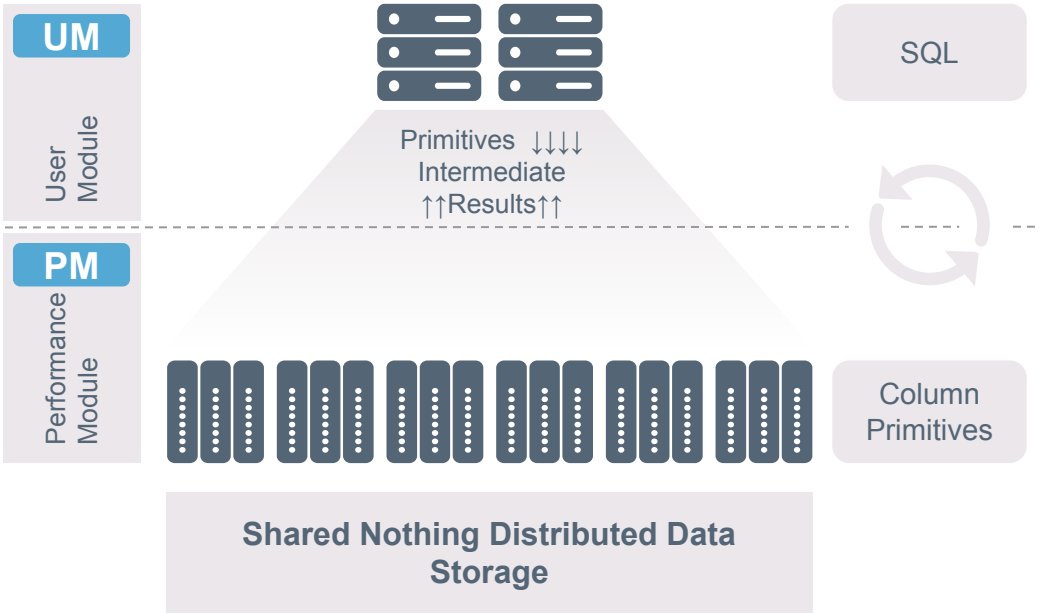
Id	OrderId	Line	Item	Quantity	Price	Supplier	ShipDate	ShipMode
1	1	1	Laptop	5	1000	Dell	2016-01-12	G
2	1	2	Monitor	5	200	LG	2016-01-13	G
3	2	1	Mouse	1	20	Logitech	2016-02-05	M
4	3	1	Laptop	3	1600	Apple	2016-01-31	P
...	...	...	...	...	...	...	...	...
8M							2016-03-05	
8M+1							2016-03-05	
...	...	...	...	...	...	...	...	...
16M							2016-09-23	
16M+1							2016-09-24	
...	...	...	...	...	...	...	...	...
24M							2017-01-06	

ELIMINATED PARTITION

ELIMINATED PARTITION

MariaDB

Massively Parallel, Shared Nothing Architecture



- Query received and parsed by MariaDB Front End on UM
- Storage Engine Plugin breaks down query in primitive operations and distributes across PM
- Primitives processed on PM
- One thread working on a range of rows
- Execute column restrictions and projections
- Execute group by/aggregation against local data
- Each PM works on primitives in parallel and fully distributed
- Each primitive executes in a fraction of a second
- Return intermediate results to UM

MariaDB

Column Types

1-byte Field	2-byte Field	4-byte Field	8-byte Field
with 8192 values per 8k block	with 4096 values per 8k block	with 2048 values per 8k block	with 1024 values per 8k block

Dictionary structure made up of 2 files/extents with:

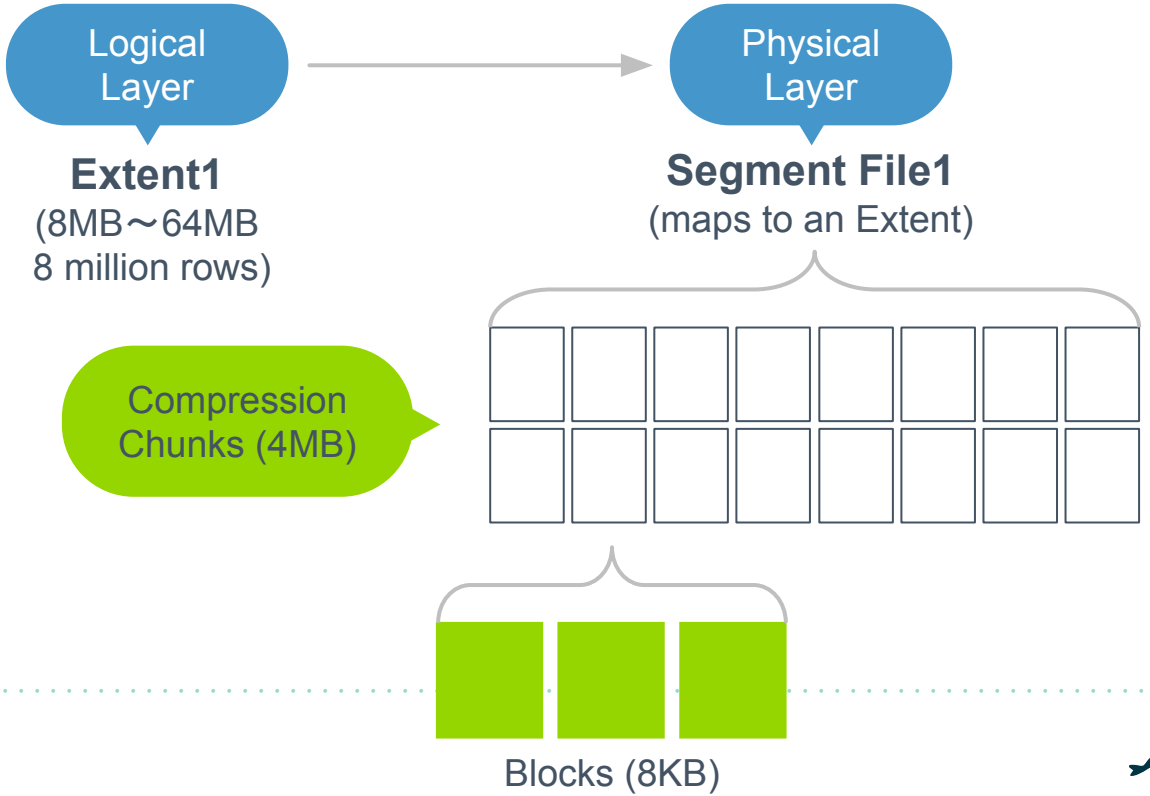
- 8-byte fixed length token (pointer).
- A variable length value stored at the location identified by the pointer.

MariaDB

Extent Map

Metadata	Definition
Object ID	The ID for the column (or dictionary)
Object Type	Column or Dictionary
LBID Start / End	Start / End Logical Block Pointer
Minimum Value	Lowest value in the extent
Maximum Value	Highest value in the extent
Width	Column Width
DBRoot	DBRoot (disk partition) number
Partition ID / Segment ID / Block Offset	The extent number
High Water Mark	Atomic last block pointer

Disk Storage

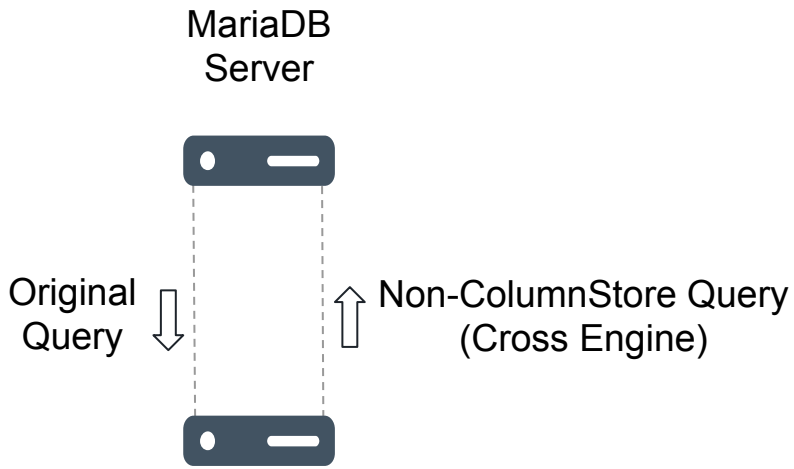


Performance Characteristics

- Uses HASH joins
- No indexes (extent map elimination instead)
- Multi-threaded joins, multi-threaded aggregates
- Bulk writes (especially using the cpimport tool) very fast
- DML writes are very slow
- Every query will try to use every core

Cross Engine Joins

- Allows non-ColumnStore tables to join with ColumnStore
- The whole query is processed by ColumnStore
- Cross Engine makes new MariaDB connections to retrieve data from non-ColumnStore tables



MariaDB


OPENWORKS
2020
MAY 4-6
CONRAD NEW YORK
EARLY BIRD REGISTRATION OPEN:
MARIADB.COM/OPENWORKS
MariaDB

Thank you