Few Words About Percona

Your Partner in MySQL and MongoDB Success

100% Open Source Software

We work with MySQL, MariaDB, MongoDB, Amazon RDS and Aurora

“No Lock in Required” Solutions and Services
About the Talk

A bit of the History

Approaches to Data Compression

What some of the popular systems implement
Let's Define The Term

Compression - Any Technique to make data size smaller
A bit of History

Early Computers were too slow to compress data in Software

Hardware Encryption (ie Tape)

Compression first appears for non performance critical data
We did not need it much for space...
Welcome to the modern age

- Data Growth outpaces HDD improvements
- Powerful CPUs
- Flash
- Data we store now
- Cloud
Exponential Data Size Growth

EMC’s The Digital Universe

- 44 ZB in 7 years!
- 4.4 ZB last year!
- 22x Growth!
Powerful CPUs

- High Performance
- Multiple Cores
Can Compress and Decompress Fast!

Snappy, LZ4

- Up to 1GB/sec compression
- Up to 2GB/sec decompression
Flash

- Disk space is more costly than for HDDs
- Write Endurance is expensive
- Want to write less data
- Decent at handling fragmentation
Cloud

- Pay for Space
- Pay for IOPS
- Network Performance may be limited
- More limited Storage Performance
Data we store in Databases

Modern Data Compresses Well!

- Text
- JSON
- XML
Introduction into a ways of making your data smaller

COMPRESSSION BASICS
Lossy and Lossless

Database generally use Lossless Compression

Lossy compression done on the application level
Some ways of getting data smaller

- Layout Optimizations
- “Encoding”
- Block Compression
- Dictionary Compression
Layout Optimizations

- Column Store versus Row Store
- Hybrid Formats
- Variable Block Sizes
<table>
<thead>
<tr>
<th>Encoding</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depends on Data Type and Domain</td>
<td></td>
</tr>
<tr>
<td>Delta Encoding, Run Length Encoding (RLE)</td>
<td></td>
</tr>
<tr>
<td>Can be faster than read of uncompressed data</td>
<td></td>
</tr>
<tr>
<td>UTF8 (strings) and VLQ (Integers)</td>
<td></td>
</tr>
<tr>
<td>Index Prefix Compression</td>
<td></td>
</tr>
</tbody>
</table>
Dictionary Compression

- Replacing frequent values with Dictionary Pointers
- Kind of like STL String
- ENUM type in MySQL
Block Compression

Compress “block” of data so it is smaller for storage

Finding Patterns in Data and Efficiently encoding them

Many Algorithms Exist: Snappy, Zlib, LZ4, LZMA
Block Compression Details

Compression rate highly depends on data

Compression rate depends on block size

Speed depends on block size and data
Block Size Dependence (by Leif Walsh)
There is no one size fits all

Typically Compression Algorithm can be selected

Often with additional settings
Where do we compress data and how do we do that

WHERE AND HOW
Where to Compress Data

- In Memory?
- In the Database Data Store?
- As Part of File System?
- Storage Hardware?
- Application?
Compression in Memory

- Reduce amount of memory needed for same working set
- Reduce IO for Fixed amount of Memory
- Encoding/Dictionary Compression are good fit
- Typically in-Memory Performance Hit
Database Data Store

- Reduce Database Size on Disk
- Works with all file systems and storage
- Dealing with fragmentation is common issue
- With OS cache can be used as In-Memory compression variant
Compression on File System Level

- Works with all Databases/Storage Engines
- Performance Impact can be significant
- ZFS
- Logical Space on disk is not reduced
Compression on Storage Hardware

- Hardware Dependent
- Does not reduce space on disk
- Can become a choke point
- Can result in Performance Gains rather than free space (SSD)
By Application

- No Database Support needed
- Reduce Database Load and Network Traffic
- Application may know more about data
- More Complexity
- Give up many DBMS features (search, index)
What makes database system to do well with compression

DESIGN CONSIDERATIONS
The Goal

Minimize Negative Impact for User Operations (Reads and Writes)
Design Principles

- Fast Decompression
- Compression in Background
- Reduce need of Re-Compression on Update
- Parallel Compression/Decompression
Choosing Block Size

**Large Blocks**
- Most efficient for compression

**Small Blocks**
- Fastest to Decompress
- Best for point lookups
What Database systems Really do with Compression

IMPLEMENTATION EXAMPLES
MySQL “Packed” MyISAM

Compress table “offline” with myisampack

Table Becomes Read Only

Variety of compression methods are used

Only data is compressed, not indexes

Note MyISAM support index prefix compression for all indexes
MySQL Archive Storage Engine

- Does not support indexes
- Essentially file of rows with sequential access
- Uses zlib compression
Innodb Table Compression

Available Since MySQL 5.1

Pages compressed using zlib

Compressed page target (1K, 4K, 8K) has to be set

Both Compressed and Uncompressed pages can be cached in Buffer Pool

Per Page “log” of changes to avoid recompression

Extenrally Stored BLOBs are compressed as single entity
InnoDB Transparent Page Compression

- Available in MySQL 5.7
- Zlib and LZ4 Compression
- Compresses pages as they are written to disk
- Free space on the page is given back using "hole punching"
- Originally designed to work with FusionIO NVMFS
- Can cause problems for current filesystem due to very high hole number
Disk usage (Linkbench data set by Sunny Bains)
Performance on Fast SSD (FusionIO NVMFS)

![Bar chart showing LinkBench Requests Per Second](chart.png)

- **Normal**: 76,261 requests per second
- **Old Compression**: 16,523 requests per second
- **New Compression**: 46,629 requests per second
Results on Slower SSD (Intel 730*2, EXT4)
Fractal Trees Compression

- Available as Storage Engine for MySQL and MongoDB
- Can use many compression libraries
- Tunable Compression Block Size
- Reduce Re-Compression due to message buffering
Can get a lot of compression
MongoDB WiredTiger Storage Engine

- Engine Has many compression settings
- Indexes are using Index Prefix Compression
- Data Pages can be compressed using zlib or Snappy
Compression Size (results by Asya Kamsky)

MongoDB 3.0 – With and Without Compression

Size (MB)

Flights

- MMAPv1
- WiredTiger (no compression)
- WiredTiger (Snappy)
- WiredTiger (zlib)
Compression in RocksDB

RocksDB – LSM Based Storage Engine for MongoDB and MySQL

LSM works very well with compression

Supports, zlib, lz4, bzip2 compression

Can use different compression methods for different Levels in LSM
Compression results from Mike Kania

![Storage Efficiency Chart]

- **MMAPv1**: 3,245GB
- **WiredTiger Snappy compression**: 318GB
- **RocksDB Snappy compression**: 283GB
PostgreSQL

Uses compression by default with TOAST

2KB (default) or longer Strings, BLOBs

Unlike InnoDB, External Storage is not required for Compression

Recommended to use File system compression i.e. ZFS if Compression is Desired
Compression is Important in Modern Age

Consider it for your system

Many different techniques are used to make data smaller by databases

Compression support is rapidly changing and improving
Want More?

I’m talking about MySQL Replication Options

Free (as in Beer) Moscow MySQL Users Group meetup November 6th,

Hosted by Mail.ru

http://www.meetup.com/moscowmysql/
Percona Live 2016 call for paper is Open

Call for Papers Open until November 29, 2016

MySQL, MongoDB, NoSQL, Data in The Cloud

Anything to make Data Happy!

Thank You!
Peter Zaitsev
pz@percona.com
https://www.linkedin.com/in/peterzaitsev