# Additional Benchmark Experiments

By Xiufeng Liu

# Some Directions

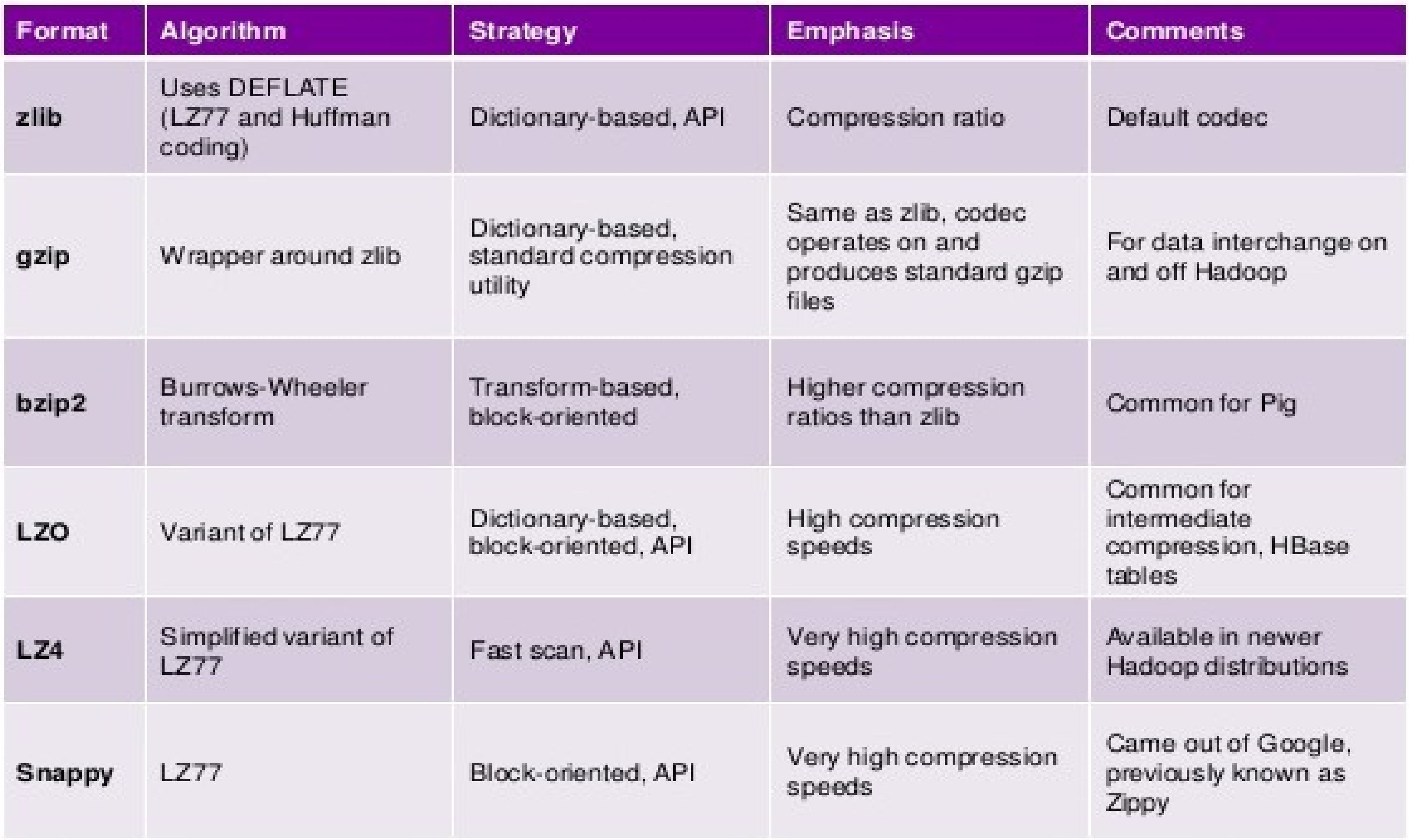
- Data File Format
- Data Compression
- Data Serialization  Methods
- Index
- File System
- Shark (on Spark) vs. Spark

# Data File Format

- Data file formats:
  - Text file format
  - SequenceFile format ( http://blog.cloudera.com/blog/2011/09/snappy-and-hadoop/ )
    - Binary storage format can improve Hadoop scan performance by 3X [1] than text format
  - Parquet: columnar data format [2]
  - RCFile: record columnar file format [3]
  - Skip list column format [1]

- [1] Floratou, A., Patel, J. M., Shekita, E. J., & Tata, S. (2011). Column-oriented storage techniques for MapReduce. Proceedings of the VLDB Endowment, 4(7), 419-429.
- [2] Parquet. www.parquet.io
- [3] He, Yongqiang, et al. "RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems." Data Engineering (ICDE), 2011 IEEE 27th International Conference on. IEEE, 2011.

# Data Compression



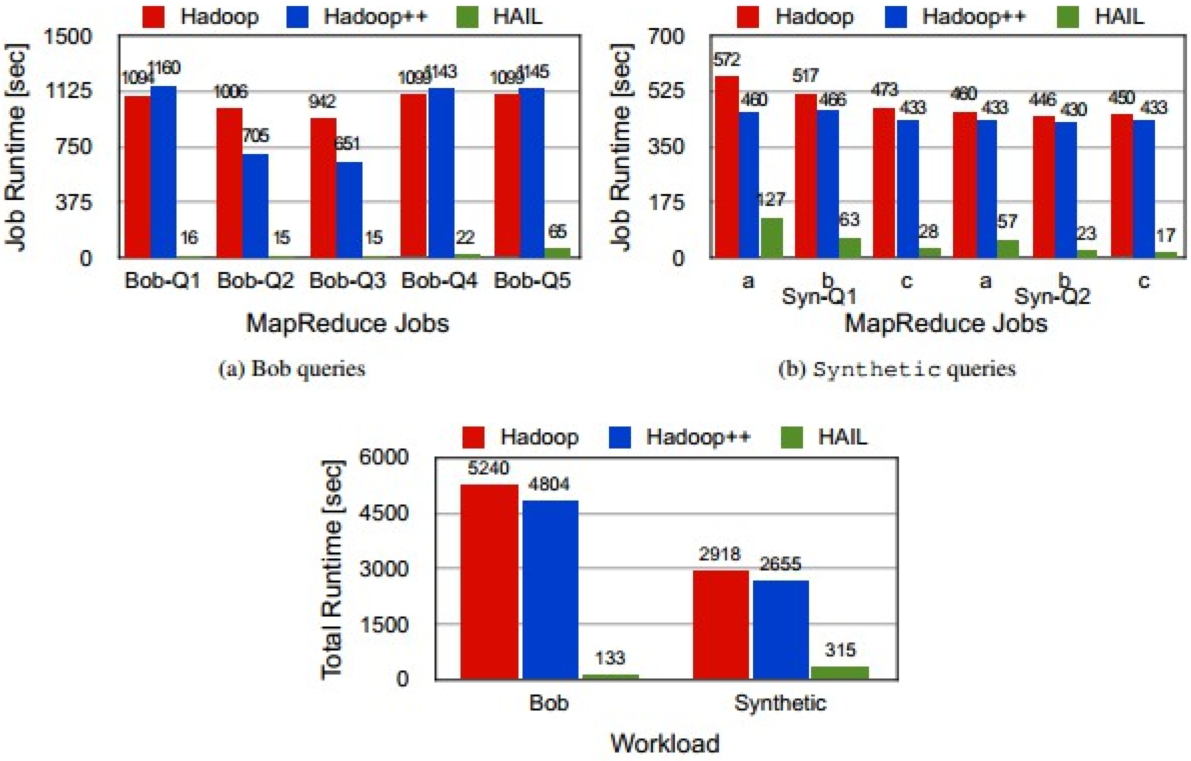
| Format | Algorithm | Strategy | Emphasis | Comments |
|---|---|---|---|---|
| **zlib** | Uses DEFLATE (LZ77 and Huffman coding) | Dictionary-based, API | Compression ratio | Default codec |
| **gzip** | Wrapper around zlib | Dictionary-based, standard compression utility | Same as zlib, codec operates on and produces standard gzip files | For data interchange on and off Hadoop |
| **bzip2** | Burrows-Wheeler transform | Transform-based, block-oriented | Higher compression ratios than zlib | Common for Pig |
| **LZO** | Variant of LZ77 | Dictionary-based, block-oriented, API | High compression speeds | Common for intermediate compression, HBase tables |
| **LZ4** | Simplified variant of LZ77 | Fast scan, API | Very high compression speeds | Available in newer Hadoop distributions |
| **Snappy** | LZ77 | Block-oriented, API | Very high compression speeds | Came out of Google, previously known as Zippy |

# Data (De-)Serialization

- Readable/Writable (built-in support by Hadoop)
- Avro (http://avro.apache.org/)
- Thrift (https://thrift.apache.org/)
- Protobufs (https://code.google.com/p/protobuf/)

# Indexing on HDFS

- Hadoop++
- HAIL (Lazy Indexing and Adaptivity in Hadoop)



(a) Bob queries

(b) Synthetic queries

(c) Total workload

***Reference***:

- Dittrich, Jens, et al. "Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing)." Proceedings of the VLDB Endowment 3.1-2 (2010): 515-529.
- Dittrich, Jens, et al. “Efficient OR Hadoop: Why not both?” see http://sites.computer.org/debull/A13mar/jens.pdf

# File System

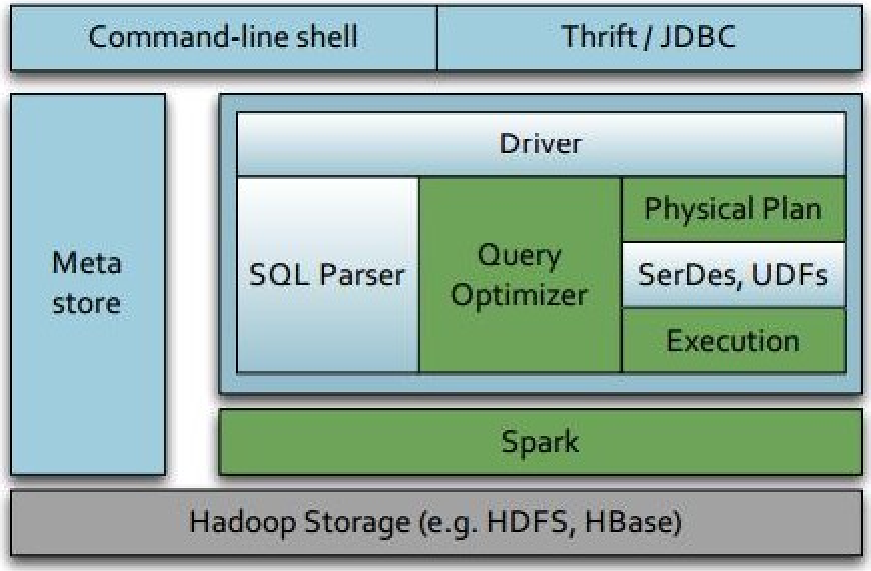
- Hadoop Distributed File System (HDFS)
  - Data co-location vs. no co-location (see CoHadoop)

- Network File System

- Access file on disk/HDFS VS. cached files (e,g., ramfs, tmpfs)

# Shark on Spark

- Shark builds on Hive that translate SQL-like queries to execution plans running on Spark
- Our Purpose: Present our experience of using Shark of facilitating processing time-series data vs. merely using Spark
- Benefits and comprises?



Shark Architecture