Yandex

Text formats

CSV, TSV, JSON, XML

CSV & TSV (comma- & tab-separated values)

›› Space efficiency

BAD

›› Speed

GOOD

›› Data types

ONLY STRINGS

›› Splittable

SPLITTABLE
W/O HEADER

›› Extensibility

BAD

```
Ticker,Date,Open,High,Low,Close,Adj  Close,Volume
^IXIC,2014-01-02,4160.029785,4160.959961,4131.790039,4143.069824,4143.069824,1738820000
^IXIC,2014-01-03,4148.560059,4152.959961,4124.959961,4131.910156,4131.910156,1667480000
^IXIC,2014-01-06,4137.029785,4139.779785,4103.750000,4113.680176,4113.680176,2292840000
^IXIC,2014-01-07,4128.569824,4158.180176,4126.479980,4153.180176,4153.180176,2278220000
^IXIC,2014-01-08,4154.279785,4171.750000,4145.000000,4165.609863,4165.609863,2345220000
^IXIC,2014-01-09,4179.040039,4182.740234,4142.700195,4156.189941,4156.189941,2214770000
^IXIC,2014-01-10,4168.939941,4174.680176,4142.209961,4174.669922,4174.669922,2143070000
^IXIC,2014-01-13,4167.410156,4179.470215,4097.990234,4113.299805,4113.299805,2322240000
^IXIC,2014-01-14,4129.600098,4183.839844,4125.810059,4183.020020,4183.020020,2034180000
^IXIC,2014-01-15,4196.529785,4218.790039,4195.979980,4214.879883,4214.879883,2101870000
^IXIC,2014-01-16,4209.589844,4219.279785,4204.160156,4218.689941,4218.689941,2005850000
^IXIC,2014-01-17,4207.819824,4217.240234,4187.310059,4197.580078,4197.580078,2150370000
^IXIC,2014-01-21,4222.979980,4227.930176,4193.169922,4225.759766,4225.759766,2034030000
^IXIC,2014-01-22,4234.580078,4246.549805,4225.520020,4243.000000,4243.000000,2026910000
^IXIC,2014-01-23,4224.359863,4224.439941,4192.279785,4218.879883,4218.879883,2191980000
^IXIC,2014-01-24,4194.970215,4197.930176,4128.169922,4128.169922,4128.169922,2489470000
^IXIC,2014-01-27,4132.220215,4136.459961,4052.629883,4083.610107,4083.610107,2398280000
^IXIC,2014-01-28,4067.860107,4099.810059,4067.689941,4097.959961,4097.959961,2091180000
^IXIC,2014-01-29,4060.610107,4091.270020,4044.760010,4051.429932,4051.429932,2231850000
^IXIC,2014-01-30,4098.810059,4135.839844,4094.169922,4123.129883,4123.129883,2168410000
^IXIC,2014-01-31,4068.629883,4124.919922,4067.610107,4103.879883,4103.879883,2300570000
```

JSON (JavaScript Object Notation)

›› Space efficiency

BAD (WORSE THAN CSV)

›› Speed

GOOD ENOUGH

›› Data types

STRINGS, NUMBERS, BOOLEANS, MAPS, LISTS

›› Splittable

SPLITTABLE IF 1 DOCUMENT PER LINE

›› Extensibility

YES


```
[
{"Ticker": "^IXIC", "Date": "2014-01-02", "Adj Close": 4143.069824, "Volume": 1738820000},
{"Ticker": "^IXIC", "Date": "2014-01-03", "Adj Close": 4131.910156, "Volume": 1667480000},
{"Ticker": "^IXIC", "Date": "2014-01-06", "Adj Close": 4113.680176, "Volume": 2292840000},
{"Ticker": "^IXIC", "Date": "2014-01-07", "Adj Close": 4153.180176, "Volume": 2278220000},
{"Ticker": "^IXIC", "Date": "2014-01-08", "Adj Close": 4165.609863, "Volume": 2345220000},
{"Ticker": "^IXIC", "Date": "2014-01-09", "Adj Close": 4156.189941, "Volume": 2214770000},
{"Ticker": "^IXIC", "Date": "2014-01-10", "Adj Close": 4174.669922, "Volume": 2143070000},
{"Ticker": "^IXIC", "Date": "2014-01-13", "Adj Close": 4113.299805, "Volume": 2322240000},
{"Ticker": "^IXIC", "Date": "2014-01-14", "Adj Close": 4183.020020, "Volume": 2034180000},
{"Ticker": "^IXIC", "Date": "2014-01-15", "Adj Close": 4214.879883, "Volume": 2101870000},
{"Ticker": "^IXIC", "Date": "2014-01-16", "Adj Close": 4218.689941, "Volume": 2005850000},
{"Ticker": "^IXIC", "Date": "2014-01-17", "Adj Close": 4197.580078, "Volume": 2150370000},
{"Ticker": "^IXIC", "Date": "2014-01-21", "Adj Close": 4225.759766, "Volume": 2034030000},
{"Ticker": "^IXIC", "Date": "2014-01-22", "Adj Close": 4243.000000, "Volume": 2026910000},
{"Ticker": "^IXIC", "Date": "2014-01-23", "Adj Close": 4218.879883, "Volume": 2191980000},
{"Ticker": "^IXIC", "Date": "2014-01-24", "Adj Close": 4128.169922, "Volume": 2489470000},
{"Ticker": "^IXIC", "Date": "2014-01-27", "Adj Close": 4083.610107, "Volume": 2398280000},
{"Ticker": "^IXIC", "Date": "2014-01-28", "Adj Close": 4097.959961, "Volume": 2091180000},
{"Ticker": "^IXIC", "Date": "2014-01-29", "Adj Close": 4051.429932, "Volume": 2231850000},
{"Ticker": "^IXIC", "Date": "2014-01-30", "Adj Close": 4123.129883, "Volume": 2168410000},
{"Ticker": "^IXIC", "Date": "2014-01-31", "Adj Close": 4103.879883, "Volume": 2300570000}
]
```

Example: XML

```
<?xml version="1.0" encoding="utf-8"?>
<Item>
  <Ticker>^IXIC</Ticker>
  <Date>2014-01-02</Date>
  <Open>4160.029785</Open>
  <High>4160.959961</High>
  <Low>4131.790039</Low>
  <Close>4143.069824</Close>
  <Adj Close>4143.069824</Adj Close>
  <Volume>1738820000</Volume>
</Item>
<Item>
  <Ticker>^IXIC</Ticker>
  <Date>2014-01-03</Date>
  <Open>4148.560059</Open>
  <High>4152.959961</High>
  <Low>4124.959961</Low>
  <Close>4131.910156</Close>
  <Adj Close>4131.910156</Adj Close>
  <Volume>1667480000</Volume>
</Item>
<Item>
  <Ticker>^IXIC</Ticker>
  <Date>2014-01-06</Date>
  <Open>4137.029785</Open>
  <High>4139.779785</High>
  <Low>4103.75</Low>
  <Close>4113.680176</Close>
  <Adj Close>4113.680176</Adj Close>
  <Volume>2292840000</Volume>
</Item>
```

Conclusion

›› Text formats
  ›› are popular, human-readable, easy to generate, easy to parse (with libraries)
  ››occupy a lot of disk space because of readability and redundancy

데이터를 처리하기 위한 PARSING 작업이 필요

›› CSV, TSV, JSON, XML are examples of text formats

JSON
- 데이터를 저장 또는 전송하기에 용이한 포맷
- 몇 가지 데이터 타입 지원
- 여전히 공간을 많이 차지함

Binary formats 1

SequenceFile, Avro

Record(Row)-Oriented Format

Inefficiencies of text formats

›› To parse "100500"
  ››iterate over characters: '1', '0', '0', '5', '0', '0'
  ››convert them to digits: 1, 0, 0, 5, 0, 0
  ››fold into the result: 1*100000 + 0*10000 + 0*1000 + 5*100 + 0*10 + 0*1

›› Not as fast as simple copying

Encoding : 데이터를 유저가 원하는 형태(format/type/object)로 convert하는 것

Serialization : 데이터를 저장 또는 전송을 위한 형태로 convert하는 것 (byte 직렬화)

SequenceFile

MapReduce 과정 중 중간 데이터를 저장하기 위한 용도로 개발됨

저장 format이 memory format과 유사

›› First binary format implemented in Hadoop

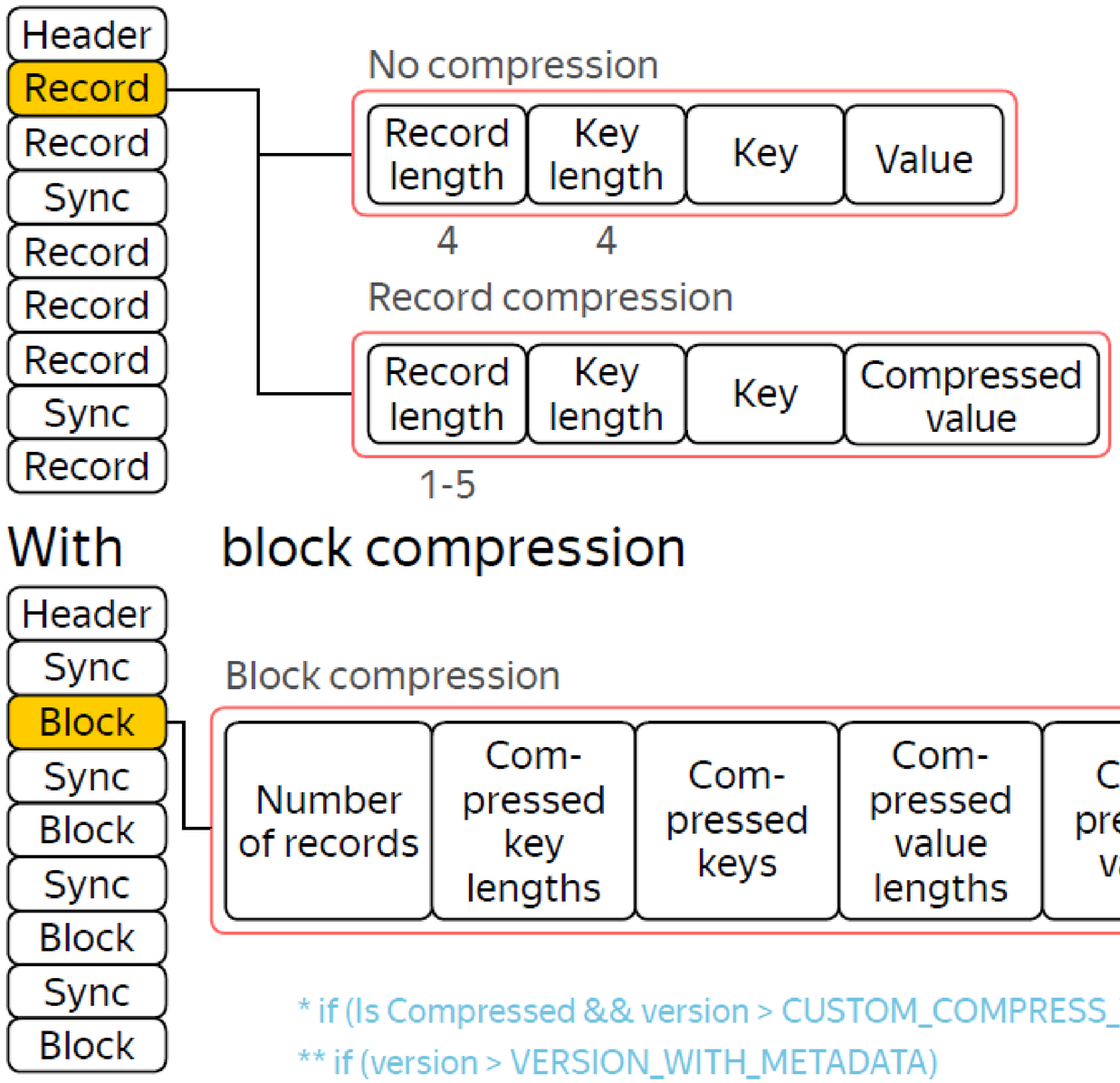
›› Stores sequence of key-value pairs

›› Java-specific serialization/deserialization

SequenceFile

Header
4 bytes - 'SEQ' + VERSION
Text - Key Class Name
Text - Value Class Name
Boolean - Is Compressed
Boolean - Is Block Compressed
Text - Compress Codec Class Name*
Metadata**
16 bytes - Sync Marker

Without block compression



With block compression

* if (Is Compressed && version > CUSTOM_COMPRESS_VERSION)

** if (version > VERSION_WITH_METADATA)

Sync Marker – '₩n'처럼 데이터를 splitting하는 용도 /

SequenceFile

› Space efficiency

MODERATE TO GOOD

› Speed

GOOD

› Data types

ANY W/ SER./DESER. CODE

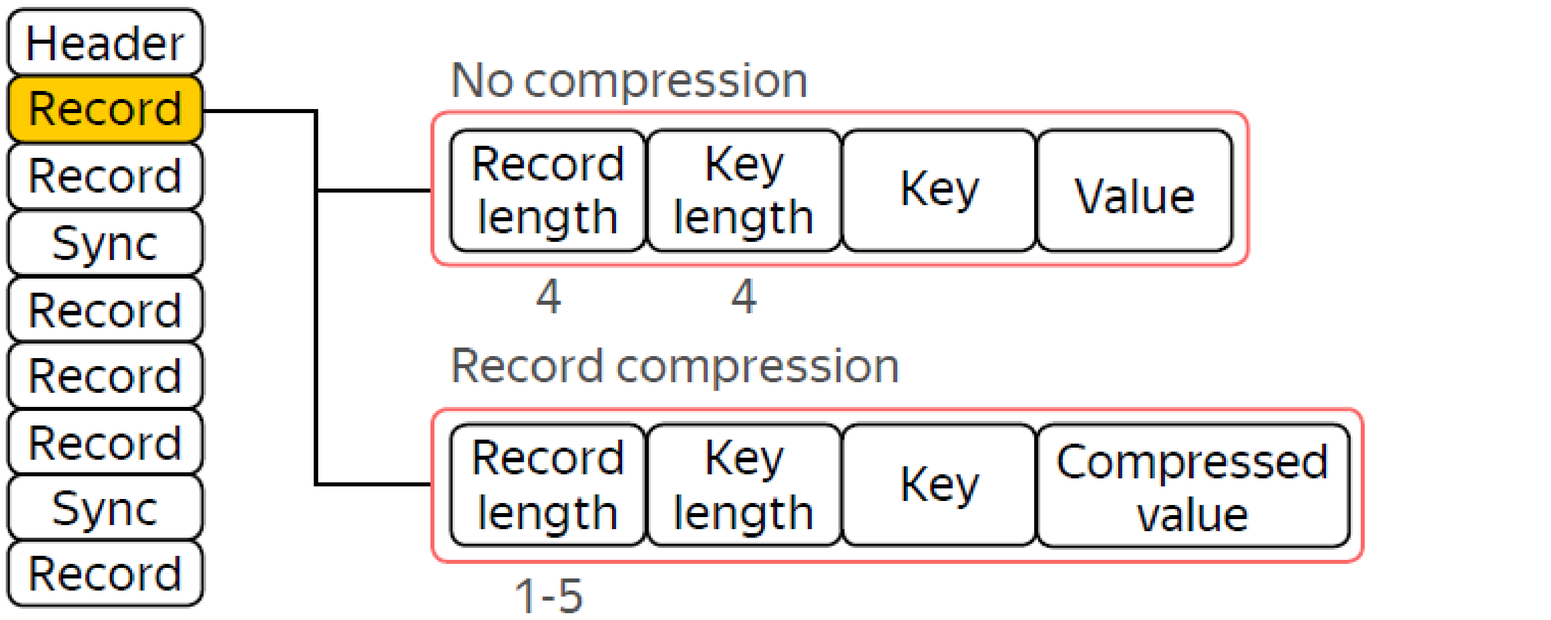
› Splittable

SPLITTABLE

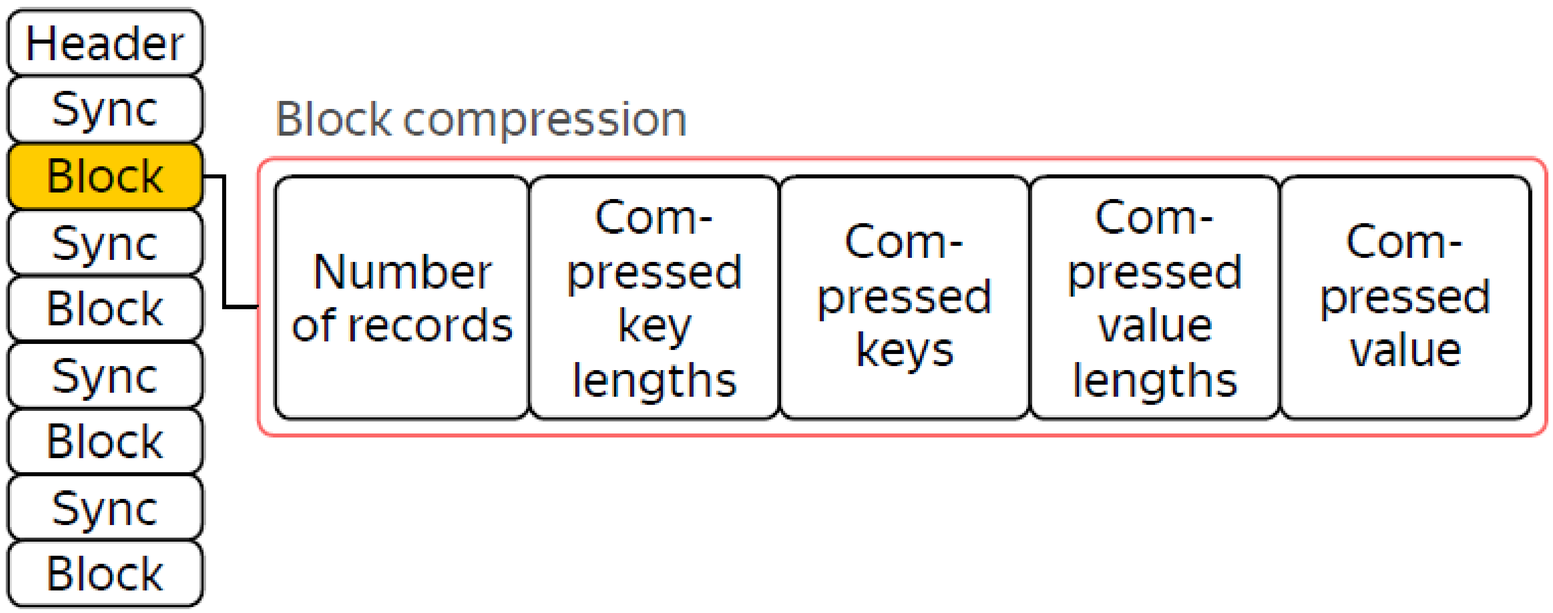
› Extensibility

NO

Without block compression



Without block compression



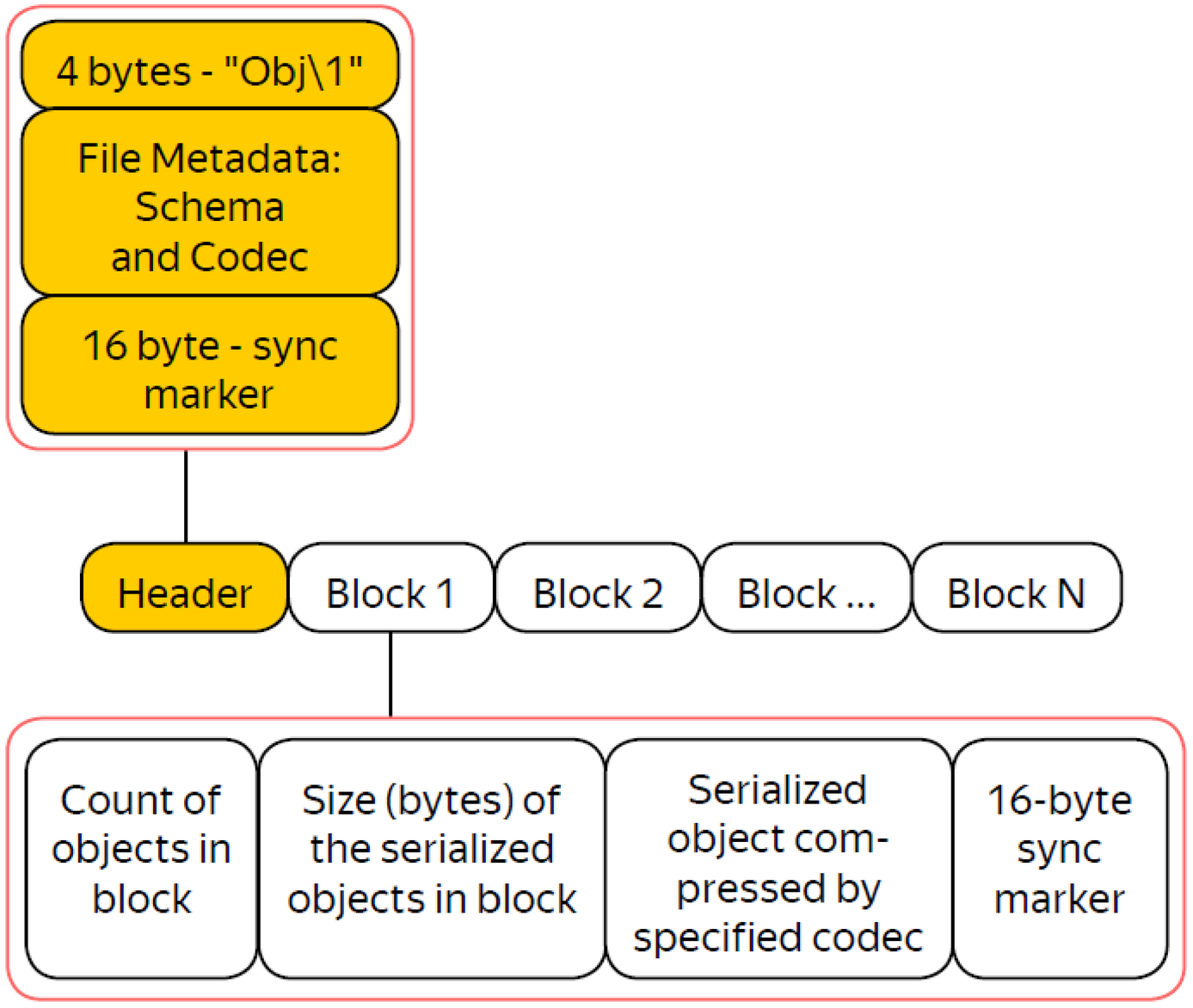
Avro

›› Both format & support library
›› Stores objects defined by the schema Schema 정의가 필요
  ››specifies field names, types, aliases
  ››defines serialization/deserialization code
  ››allows some schema updates
›› Interoperability with many languages 다양한 언어 지원

- Compression 지원
- 다른 언어와 호환대는 대신 데이터 타입이 SequenceFile보다 제한적

Avro

- Space efficiency
  MODERATE TO GOOD
- Speed
  GOOD WITH CODEGEN
- Data types
  JSON-LIKE
- Splittable
  SPLITTABLE
- Extensibility
  YES



Binary formats 2

RCFile, Parquet

Columnar data Format

분석 용도 데이터에 적합

Row-based & column-based formats

Column-based의 장점 :

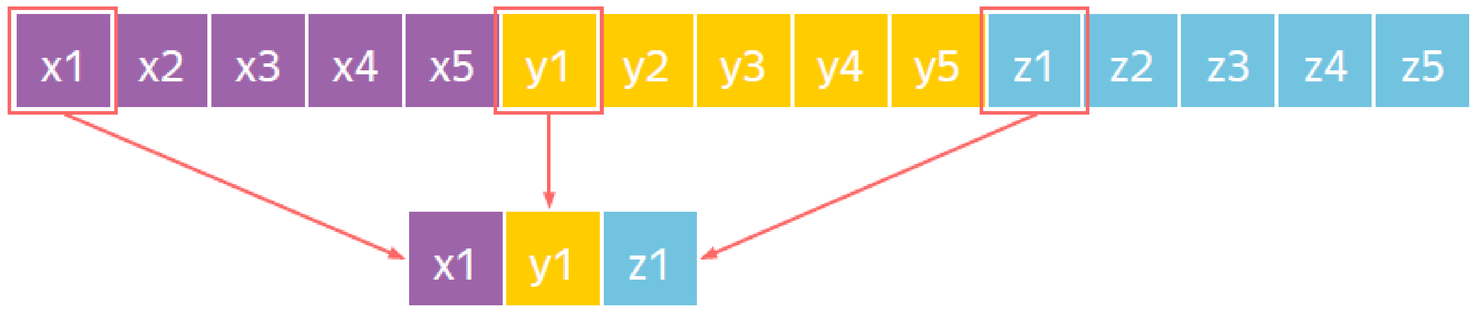
- 필요한 데이터만 scan이 가능 (I/O bound 최소화)
- 압축성능이 좋음 (비슷한 타입끼리 묶이기 때문)

Table representation

X	Y	Z
x1	y1	z1
x2	y2	z2
x3	y3	z3
x4	y4	z4
x5	y5	z5

Row format

Columnar format



Row 데이터를 보기 위해서는 Row assembly 작업이 필요하다.

RCFile

Record Columnar

Hive와 호환이 뛰어나다.

›› First columnar format* in Hadoop()

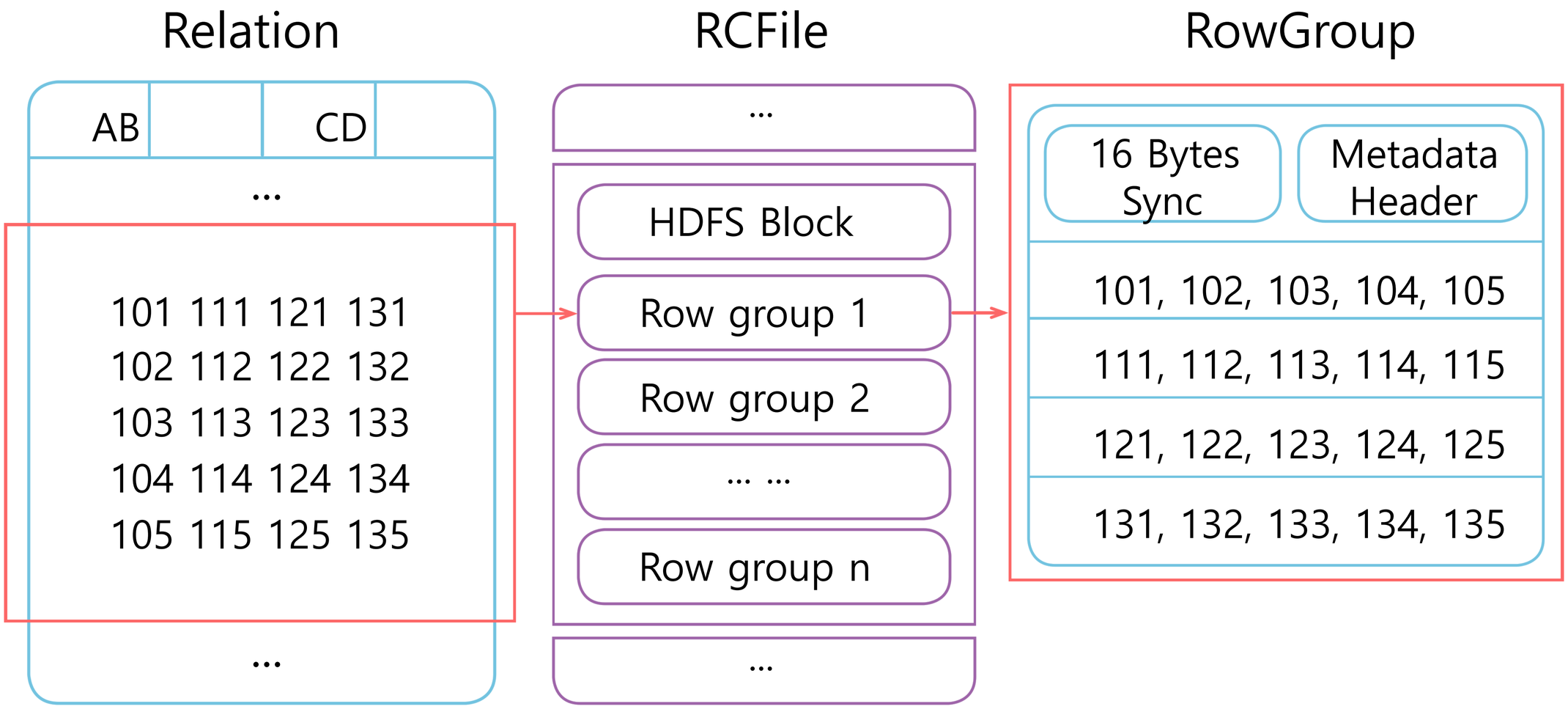
›› Horizontal/vertical partitioning

›› split rows into row groups

›› transpose values within a row group

* " RCFile: A Fast and Space-efficient Data Placement Structure in MapReduce-based Warehouse Systems", by He et. al; Facebook, Ohio State University; Chinese Academy of Sciences

RCFile



데이터를 ROW GROUP으로 나누고 그 안에서 컬럼끼리 분리 / 모든 데이터는 binary 형태

Row Assembly는 local에서 실행되도록 data를 저장한다. (네트워크 I/O 불필요)

RCFile

Metadata는 run-length encoding??

Column 데이터는 일반적인 압축(zip)을 사용하여 그다지 빠르지는 않다.

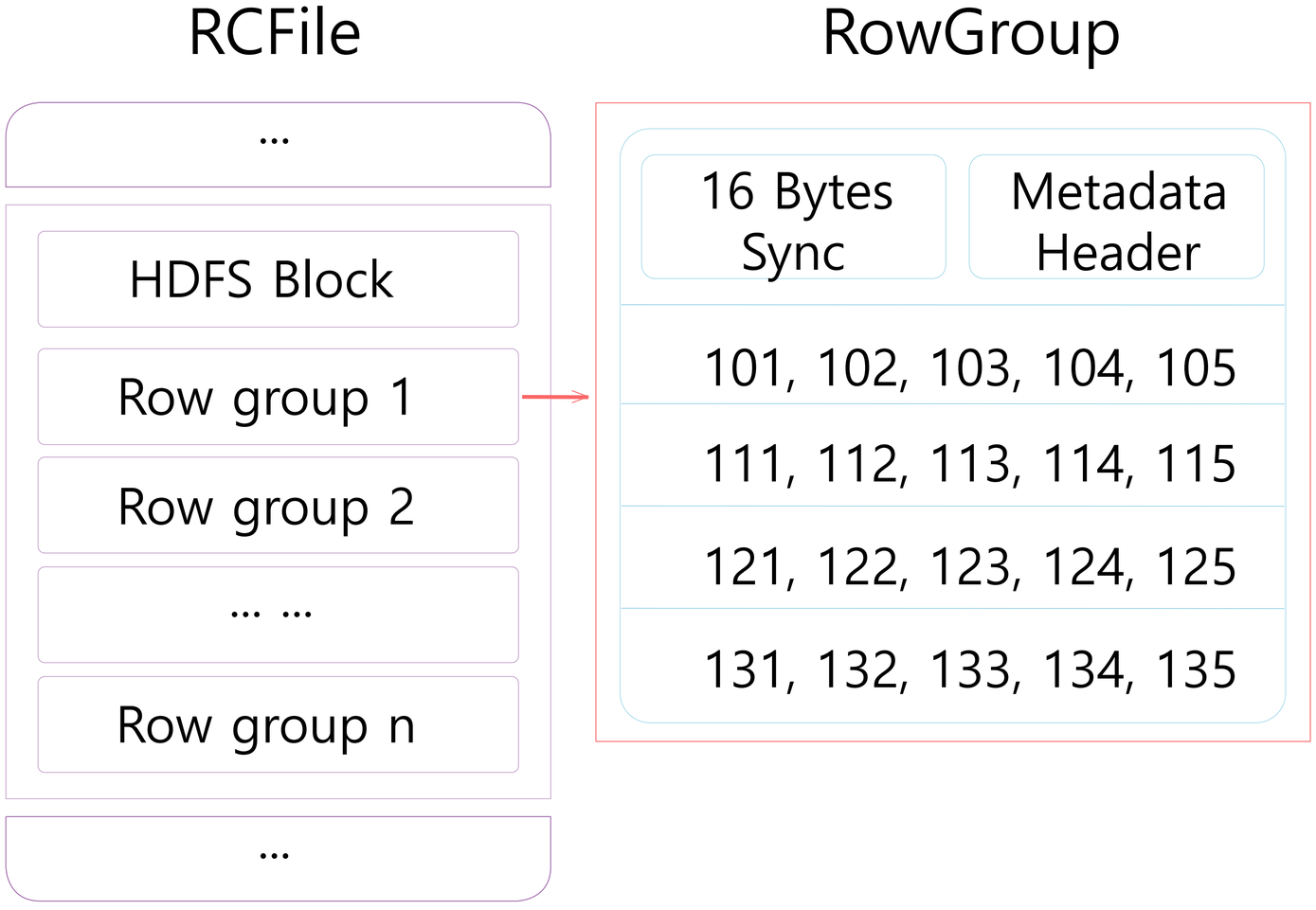
›› Space efficiency
GOOD

›› Speed
MODERATE TO GOOD, LESS I/O

›› Data types
BYTE STRINGS

›› Splittable
SPLITTABLE

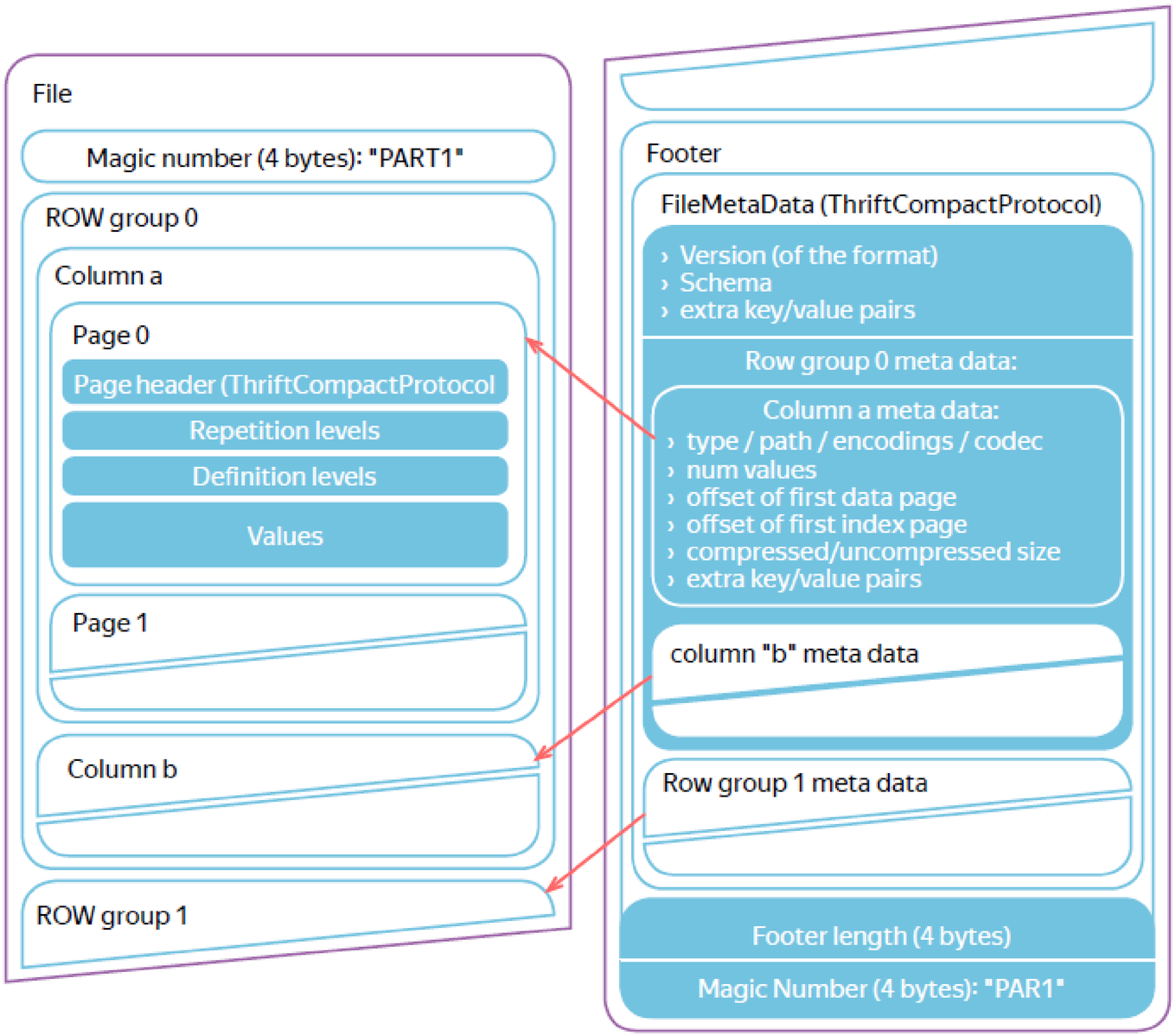
›› Extensibility
NO



Parquet

RC File은 몇 가지 최적화가 덜 된 것들이 존재하여 대안으로 사용되는 format

- The most sophisticated columnar format in Hadoop
- Collaborative effort by Twitter & Cloudera
- Supports nested and repeated data
- Exploits many columnar optimizations (such as predicate pruning, per column codecs)
- Optimizes write path



http://parquet.apache.org

Conclusion

- Binary formats are efficient in coding data
  - SequenceFile is a reasonable choice for Java users
  - Avro is a good alternative for many use cases
  - RCFile/ORC/Parquet are best for “wide” tables and analytical workloads

Compression

Kinds of compression

2가지 방식이 혼용된다.

SyncMarker & metadata가 모든 데이터가 아닌 필요한 데이터만

Decompressing하도록 돕는 역할을 한다. (효율적)

›› Block-level compression
  ››used in SequenceFiles, RCFiles, Parquet
  ››applied within a block of data

›› File-level compression
  ››applied to the file as a whole
  ››hinders an ability to navigate through file

Codecs

›› Gzip
  ›› compression speed ~16-90 MiB/s
  ››d ecompression speed ~250-320 MiB/s
  ››ratio ~2.77 .. 3.43

›› Bzip2 Slowest but efficient
  ›› compression speed ~12-14 MB/s
  ››d ecompression speed ~38-42 MiB/s
  ››ratio ~4.02 .. 4.80

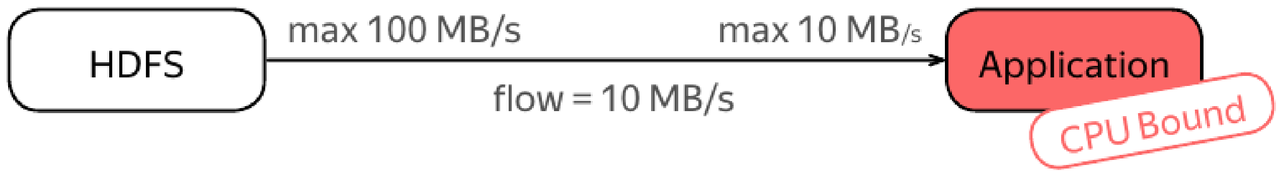
›› LZO
  ››c ompression speed ~77-150 MiB/s
  ››d ecompression speed ~290-314 MiB/s
  ››ratio ~2.10 .. 2.48

›› Snappy Fastest but efficient
  ››compression speed ~200 MiB/s
  ››decompression speed ~475 MiB/s
  ››ratio ~2.05

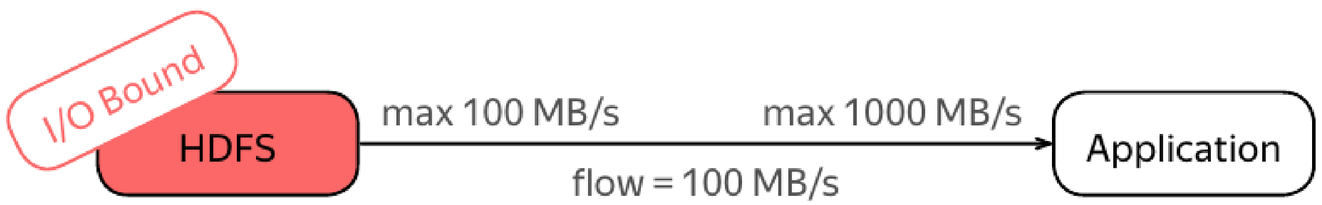
Numbers are from https://quixdb.github.io/squash-benchmark/ (webster/peltast)

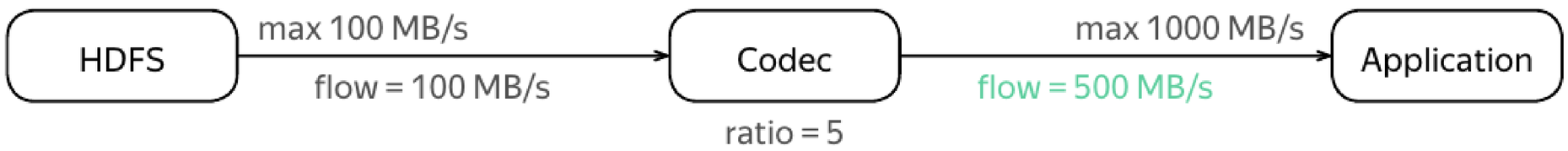
When to use compression?

Application이 감당가능한 데이터를 보내야 한다.



No benefit in using compression





Five times more throughput when using compression

Conclusion

›› Raise awareness about application bottlenecks
  ››CPU-bound → cannot benefit from the compression
  ››I/O-bound → can benefit from the compression

›› Codec performance vary depending on data, many options available

Conclusion (lesson)

›› Many applications assume relational data model

›› File format defines encoding of your data
  ››text formats are readable, allow quick prototyping, but inefficient
  ››binary formats are efficient, but more complicated to use

›› File formats vary in terms of space efficiency, encoding & decoding speed, support for data types, extensibility

›› When I/O bound, can benefit from compression

›› When CPU bound, compression may increase completion time

BigDATAteam