Map output Compression Properties

- mapreduce.map.output.compress - boolean | false
- mapreduce.map.output.compress.codec - class | DefaultCodec

(3) Serialization

- turnup objects into byte streams
- Used for transmission over n/w, writing to persistent Stora
- Deserialization - reverse process
- Used in 2 distinct area - IPC (RPC), Persistant storage
- RPC used Serialization to render msg into binary Stre
  - ↳ compact | fast | extensible | Interoperable
- RPC lifespan less than second | Persistent data - years
- RPC 4 properties - Persistent also consider
- Hadoop uses its own Serialization formats.
- Writable - compact, fast | not easy to extend or use from other language than Java

Writable interface

- 2 methods (write(), readfields()) readfields()
  - ↳ 1) writing its state to DataOutput binary stream
  - 2) Reading its state from DataInput binary stream
- IntWritable - wrapper for Java int.
- IntWritable w = new IntWritable;
  w.set(160);
  or IntWritable w = new IntWritable(160);
- Need help to serialized IntWritable

Serialization

```
ByteArrayOutputStream out = new ByteArrayOutputStream();
DataOutputStream do = new DataOutputStream(out);
Writable.write(do);
do.close()
```

```
ByteArrayInputStream in = new ByteArrayInputStream(bytes)
DataInputStream datain = new DataInputStream(in)
writable.readFields(dataIn)
    dataIn.close()
```

- IntWritable implements WritableComparable interface
  ↓ subinterface of (Writable, Comparable)
- It permit implementers to compare record w/o deserializing from object - avoid overhead of object creation

Writable classes

Java Primitive	WritableImpl	Serialize Size (byte)
boolean	BooleanWritable	1
byte	ByteWritable	1
short	ShortWritable	2
int	IntWritable	4
float	FloatWritable	4
long	LongWritable	8
double	DoubleWritable	8

Text - Writable for UTF-8 seq
- writable equivalent to String

⇒ Sequence file
- Log file - each log record is a new line of text
- For Log binary type, Plain text is not suitable format
- SequenceFile.Writer - providing persistent data structure for binary key-value pair.

- Key - Timestamp (LongWritable), Value (Writable) data
- Also containers for smaller files
- Packing files into SequenceFile makes storing and processing smaller file - efficient
- Work naturally with Hadoop Data type
- Includes metadata which identifies the data types of key value
- Actually 3 file types in one
  - ↳ UnCompressed | Record compressed | Block Compressed
- Used in Map Reduce
  - ↳ SequenceFileInputFormat
  - ↳ SequenceFileOutputFormat

Problems

- Useful but Potential problems
- Only typical accessible via Java API
- If def of key-value object changes, file becomes unreadable.
- Alternate is AVRO
- Append only

Advantage

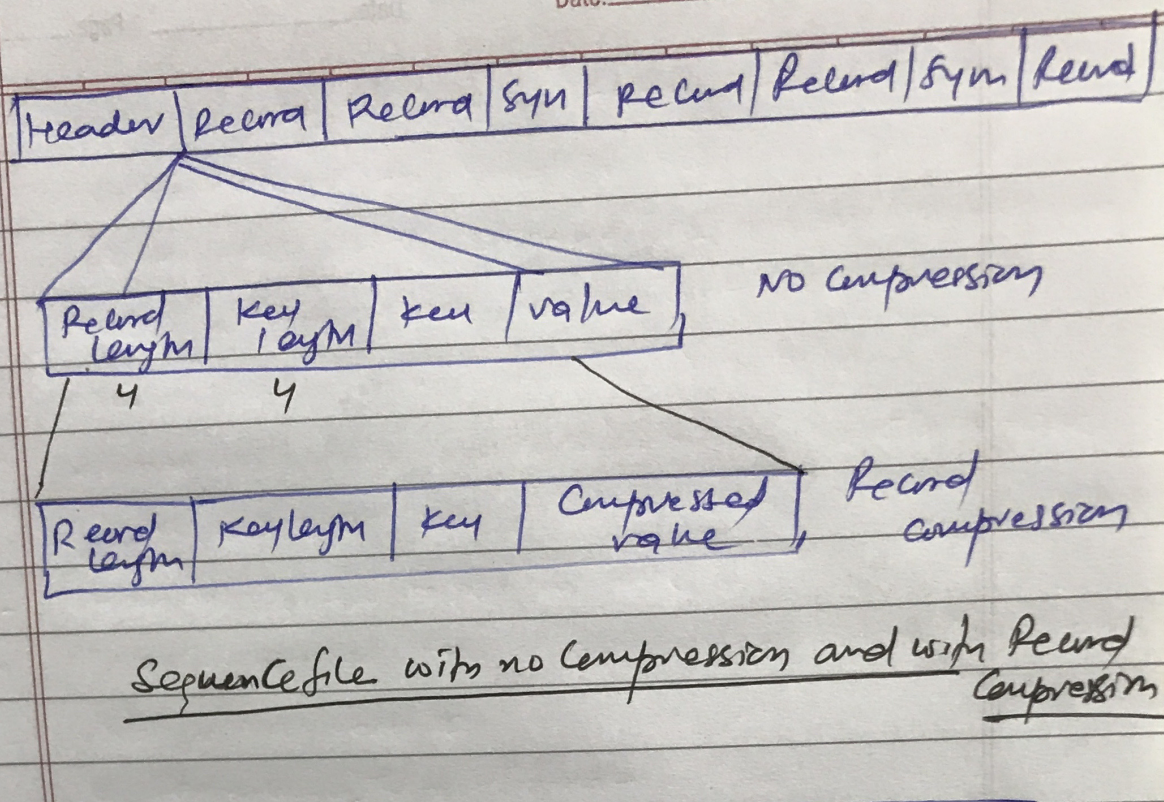
1- More compact than text files
2- Provide support for compression at diff level - Block | Record
3- Can be split and processed in parallel
4- Solve large number of small file Problem.
5- Temp output of mapper can be stored in Seq file

Writing a SequenceFile

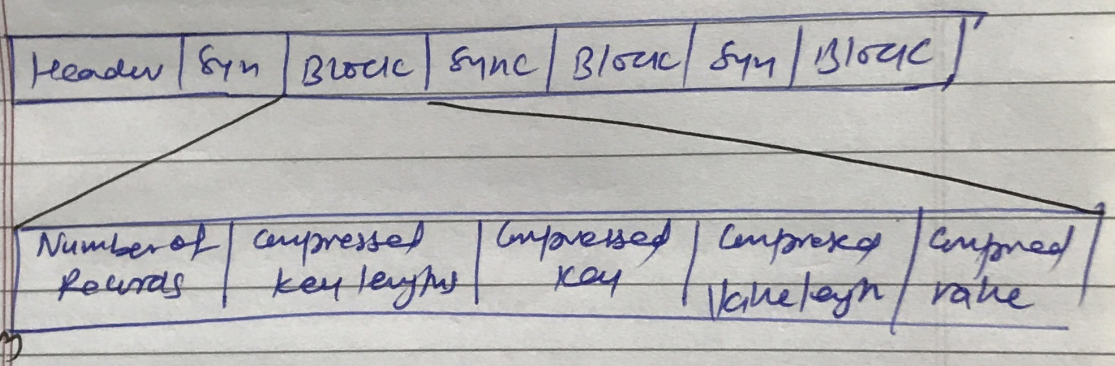
```
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(URI.create(uri), conf);
Path path = new Path(uri);
IntWritable key = new IntWritable();
Text value = new Text();
SequenceFile.Writer writer = null;
try {
    writer = SequenceFile.createWriter(fs, conf, path,
        key.getClass(), value.getClass());
```

Reading a SequenceFile

```
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(URI.create(uri), conf);
Path path = new Path(uri);
SequenceFile.Reader reader = null;
try {
    reader = new SequenceFile.Reader(fs, path, conf);
    Writable key = (Writable) ReflectionUtils.newInstance
        (reader.getKeyClass(), conf);
    Writable value = (Writable) ReflectionUtils.newInstance
        (reader.getValueClass(), conf);
```



SequenceFile with no Compression and with Record Compression



Seqfile with block Compression

Sequence file Programs