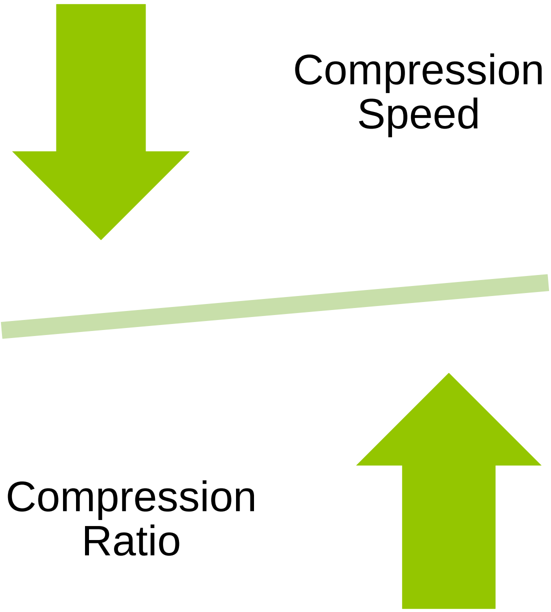
Data Compression Tradeoffs

+

- Reduces Storage Need
- Less Disk I/O
- Speeds up Network Transfer

−

- Consumes CPU



Compression Speed

Compression Ratio

Compression Algorithms

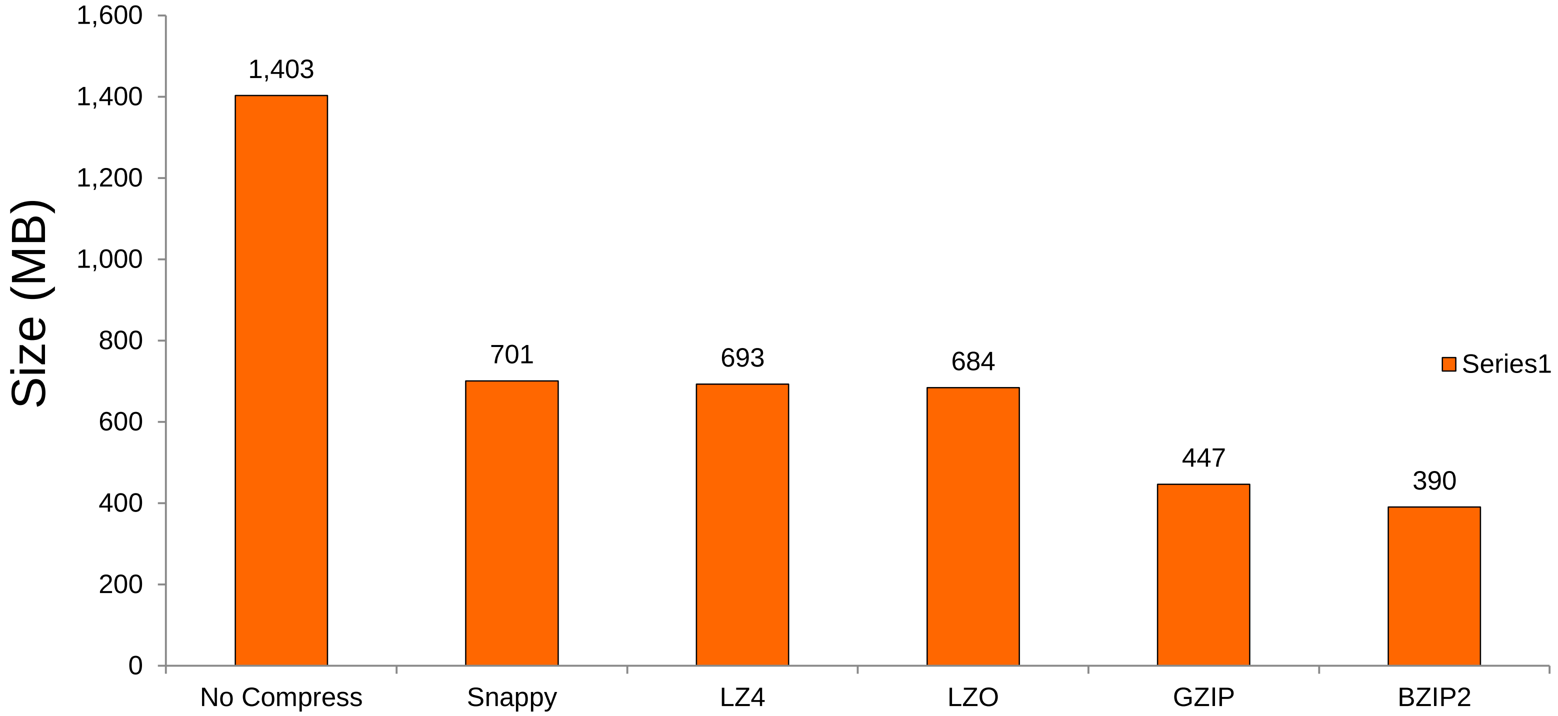
Format	Algorithm	File Extension	Splittable	Java / Native
GZIP	Deflate	.gz	N	Both
BZIP2	Bzip2	.bz2	Y	Both
LZO	LZO	.lzo	Y (Indexed)	Native
Snappy	Snappy	.snappy	N	Native
LZ4	Kind of LZ77	.lz4	N	Native

- Splittability: Every compressed split of the file can be uncompressed and processed independently. Parallel processing is possible.
- Native implementations are preferable due to higher performance rates.

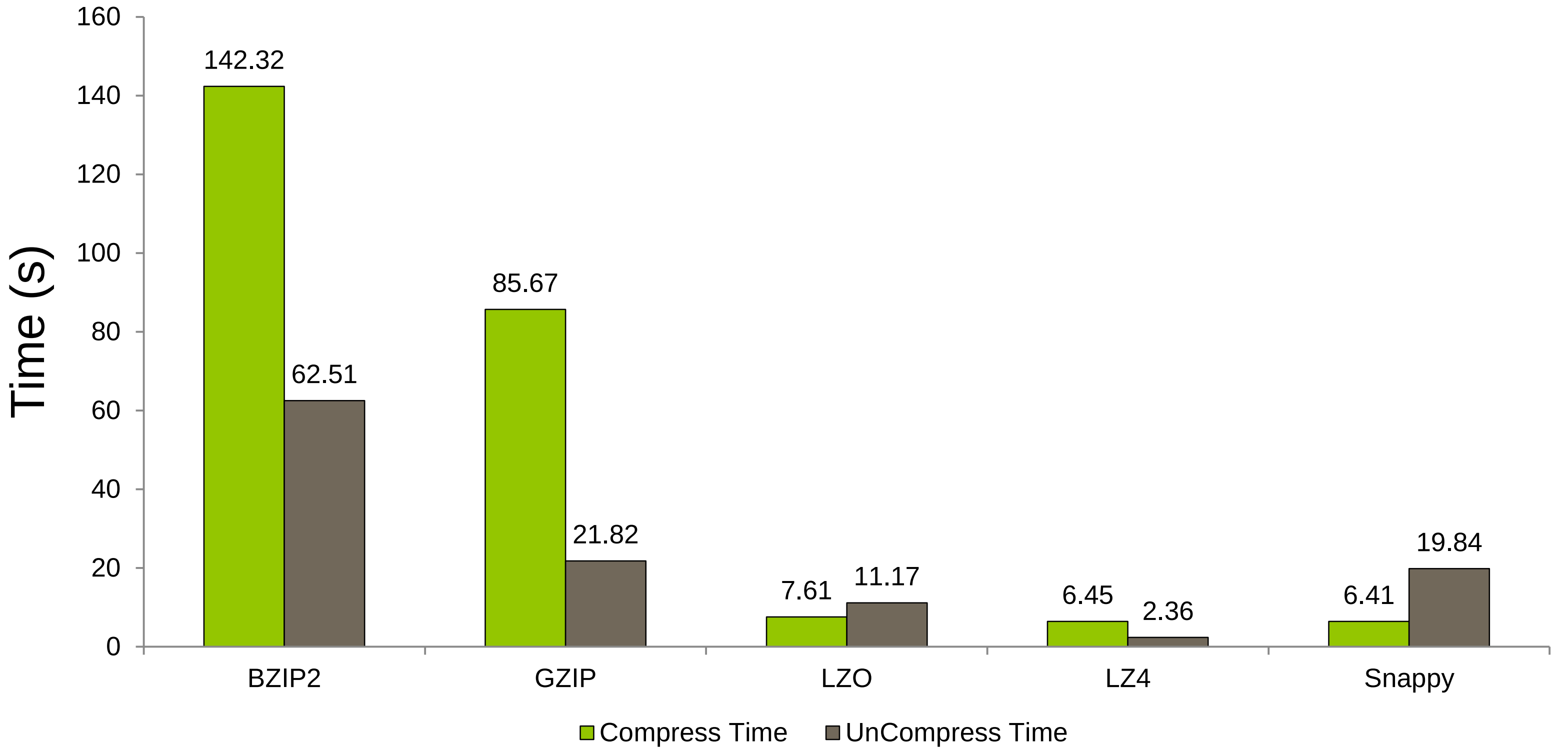
Test Environment

- 8 core i7 CPU
- 8 GB memory
- 64 bit CentOS operating system
- 1.4 GB Wikipedia Corpus 2-gram text input

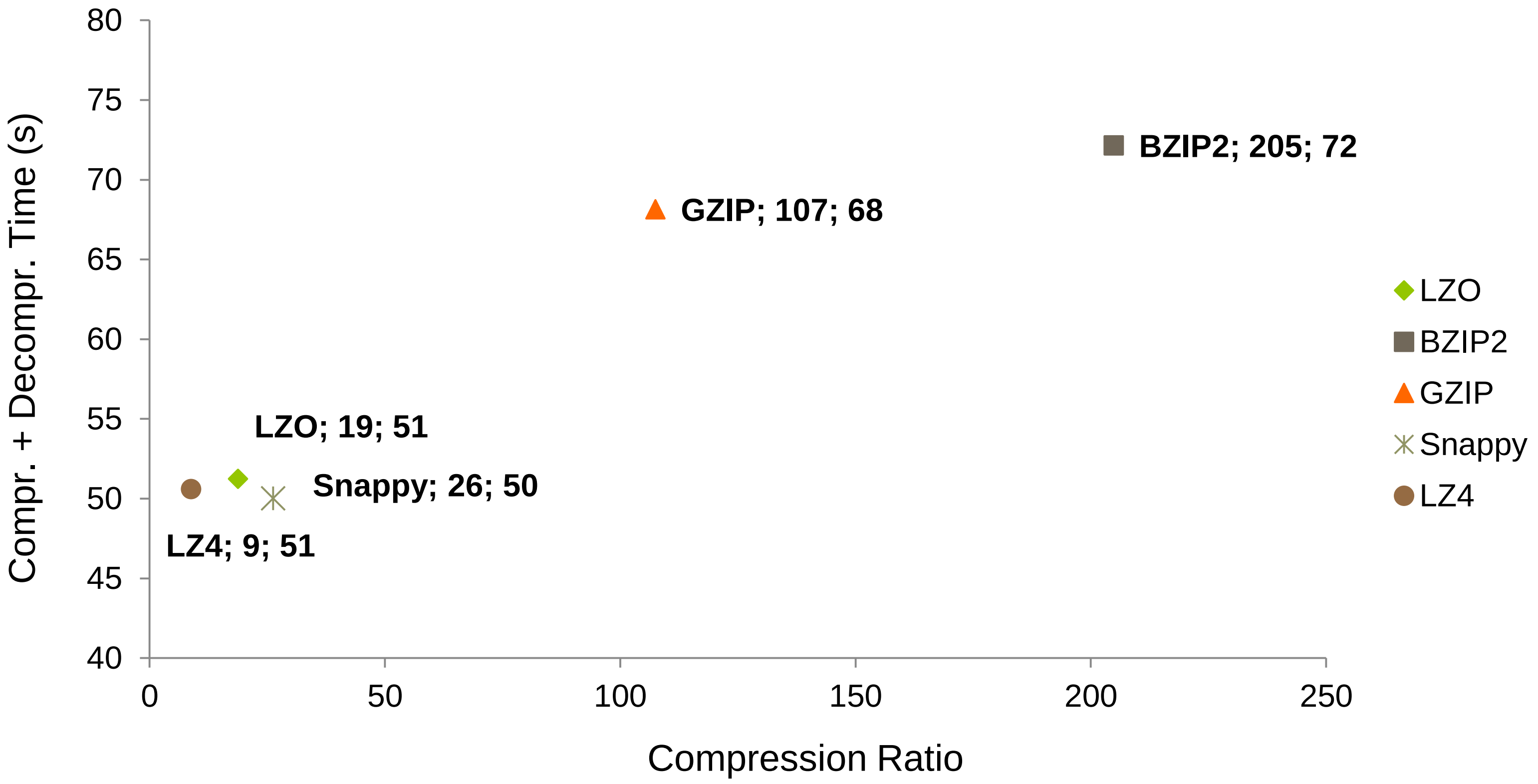
Compression Ratio



Compression Speed



Comp. Ratio / Speed Tradeoff



- Compression Ratio: 1- (Compressed Size / UnCompressed Size) * 100
- 1.4 GB Sized Wikipedia Corpus data is used for performance comparisons

Test Results

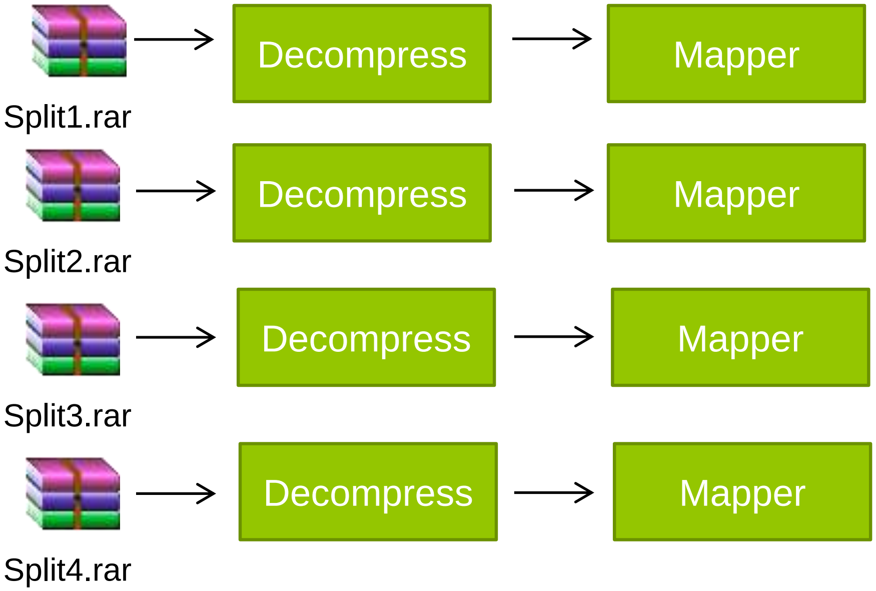
Format	Strengths	Weaknesses
GZIP	➢ Relatively high compression ratio ➢ Reasonable speed	➢ Relatively slower than lzo, snappy and lz4 ➢ Non splittable
BZIP2	➢ Best compression ratio ➢ Splittable	➢ 2x slower than gzip
LZO	➢ Rapid compression ➢ Balanced comp/decomp times	➢ Non splittable
Snappy	➢ Quickest compression method	➢ Relatively slow in decompression ➢ Non splittable
LZ4	➢ Very quick compression method ➢ Best results in decompression speed	➢ Non splittable

Data Compression in Hadoop

- Hadoop jobs are usually I/O bound
- Compression reduces the size of data transferred accross network
- Overall job performance may be increased by simply enabling compression
- Splittability must be taken into account!

What is Splittable Compression?

- If a compression method is splittable, every compressed input split can be extracted and processed independently.
- Otherwise, in order to decompress the input file every compressed split should be transferred to a single mapper node.



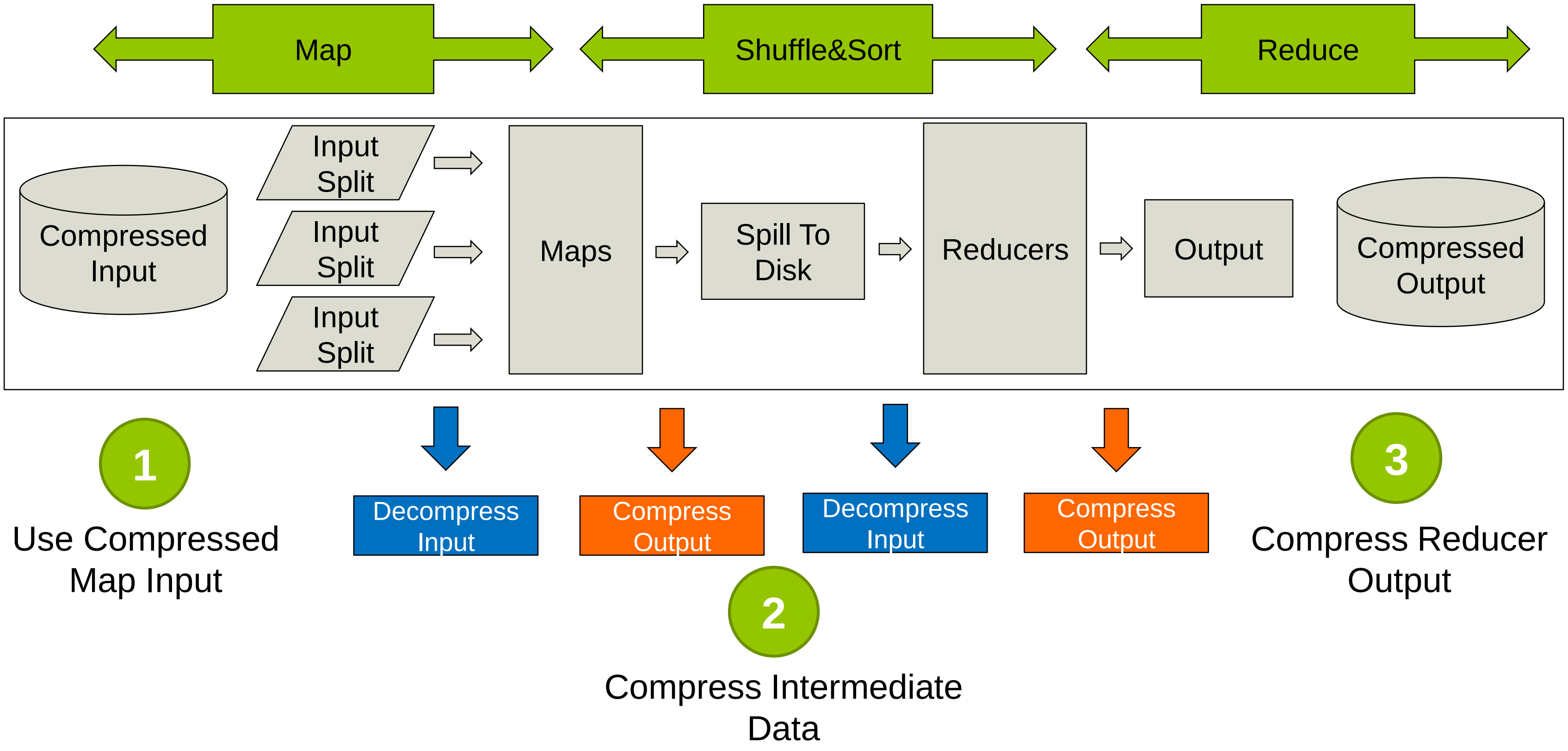
Split1.rar → Decompress → Mapper

Split2.rar → Decompress → Mapper

Split3.rar → Decompress → Mapper

Split4.rar → Decompress → Mapper

Compression in MapReduce Pipeline



Map

Shuffle&Sort

Reduce

Compressed Input

Input Split

Input Split

Input Split

Maps

Spill To Disk

Reducers

Output

Compressed Output

1

Use Compressed Map Input

Decompress Input

Compress Output

Decompress Input

Compress Output

2

Compress Intermediate Data

3

Compress Reducer Output

When to Compress?

1

Use Compressed Map Input

- Mapreduce jobs read input from HDFS
- Compress if input data is large. This will reduce disk read cost.
- Compress with splittable algorithms like Bzip2
- Or use compression with splittable file structures such as Sequence Files, RC Files etc.

2

Compress Intermediate Data

- Map output is written to disk (spill) and transferred accross the network
- Always use compression to reduce both disk write, and network transfer load
- Beneficial in performace point of view even if input and output is uncompressed
- Use faster codecs such as Snappy, LZO

3

Compress Reducer Output

- Mapreduce output used for both archiving or chaining mapreduce jobs
- Use compression to reduce disk space for archiving
- Compression is also beneficial for chaining jobs especially with limited disk throughput resource.
- Use compression methods with higher compress ratio to save more disk space

Supported Codecs in Hadoop

- Zlib → `org.apache.hadoop.io.compress.DefaultCodec`
- Gzip → `org.apache.hadoop.io.compress.GzipCodec`
- Bzip2 → `org.apache.hadoop.io.compress.BZip2Codec`
- Lzo → `com.hadoop.compression.lzo.LzoCodec`
- Lz4 → `org.apache.hadoop.io.compress.Lz4Codec`
- Snappy → `org.apache.hadoop.io.compress.SnappyCodec`

Compression in MapReduce

Compressed Input Usage	File format is auto recognized with extension. Codec must be defined in core-site.xml.

Compress Intermediate Data (Map Output)	mapreduce.map.output.compress = *True;*
	mapreduce.map.output.compress.codec = *CodecName;*

Compress Job Output (Reducer Output)	mapreduce.output.fileoutputformat.compress = *True;*
	mapreduce.output.fileoutputformat.compress.codec = *CodecName;*

Compression in Pig

Compressed Input Usage	File format is auto recognized with extension. Codec must be defined in core-site.xml.

Compress Intermediate Data (Map Output)	`pig.tmpfilecompression = *True*;`
	`pig.tmpfilecompression.codec = *CodecName*;`
	
	Use faster codecs such as Snappy, Lzo, LZ4
	Useful for chained mapreduce jobs with lots of intermediate data such as joins.

Compress Job Output (Reducer Output)	(Same as MapReduce)
	
	`mapreduce.output.fileoutputformat.compress=*True*;`
	`mapreduce.output.fileoutputformat.`
	`compress.codec = *CodecName*;`

Compression in Hive

Compressed Input Usage	Can be defined in table definition
	`STORED AS INPUTFORMAT`
	`\"com.hadoop.mapred.DeprecatedLzoTextInputFormat\"`

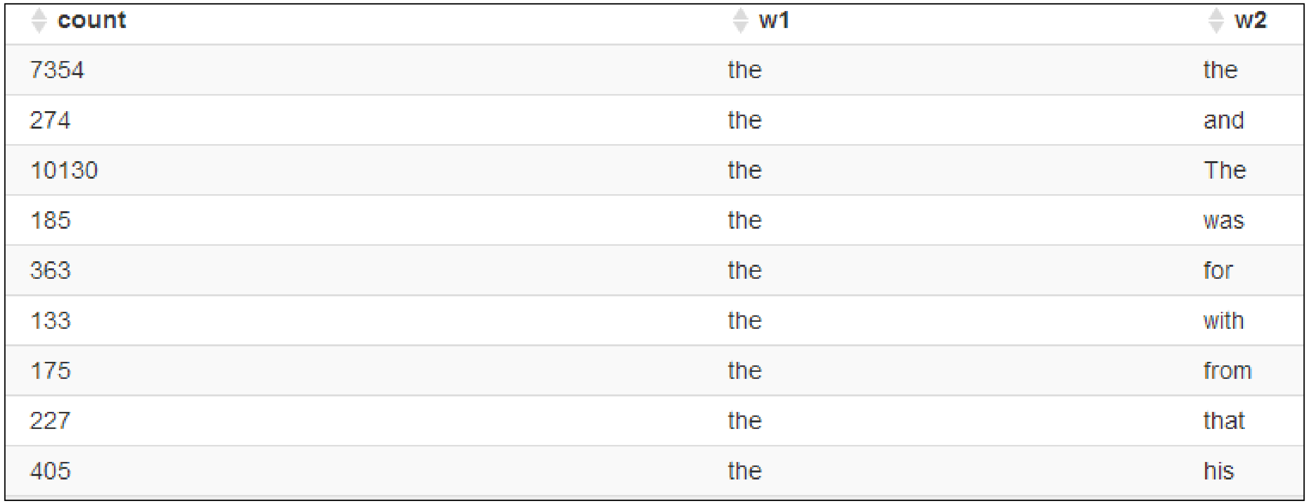
Compress Intermediate Data (Map Output)	`SET hive.exec.compress.intermediate = True;`
	`SET mapred.map.output.compression.codec = CodecName;`
	`SET mapred.map.output.compression.type = BLOCK / RECORD;`
	Use faster codecs such as Snappy, Lzo, LZ4
	Useful for chained mapreduce jobs with lots of intermediate data such as joins.

Compress Job Output (Reducer Output)	`SET hive.exec.compress.output = True;`
	`SET mapred.output.compression.codec = CodecName;`
	`SET mapred.output.compression.type = BLOCK / RECORD;`

Performance Test For Hive

We are going to test the performance effect of compression in Hive

Input File: Wikipedia Corpus 2-gram text data



count	w1	w2
7354	the	the
274	the	and
10130	the	The
185	the	was
363	the	for
133	the	with
175	the	from
227	the	that
405	the	his

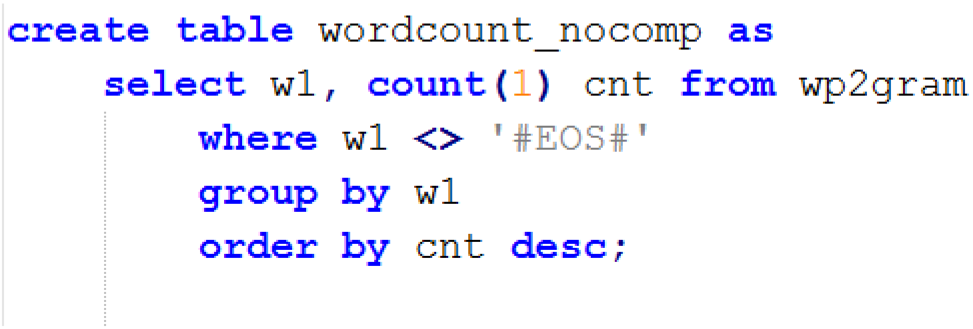
Performance Test For Hive

Case1:

- Input data is uncompressed text file
- No intermediate compression
- No output compression

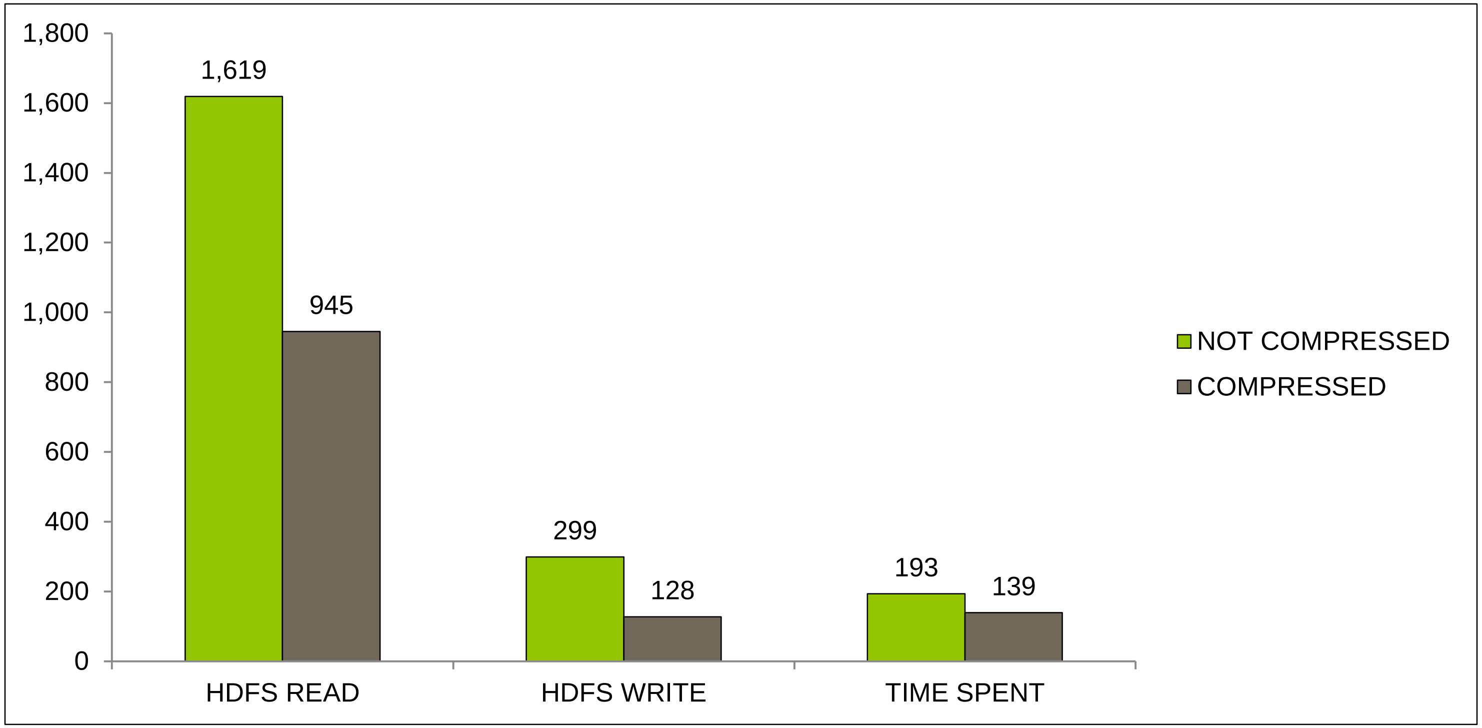
Case2:

- Input data is sequence file compressed with Snappy format
- Intermediate data is compressed with Snappy
- Output data is compressed with Snappy



```
create table wordcount_nocomp as
    select w1, count(1) cnt from wp2gram
        where w1 <> '#EOS#'
        group by w1
        order by cnt desc;
```

Performance Test For Hive



Questions