2017·09

Hadoop离线大数据分析

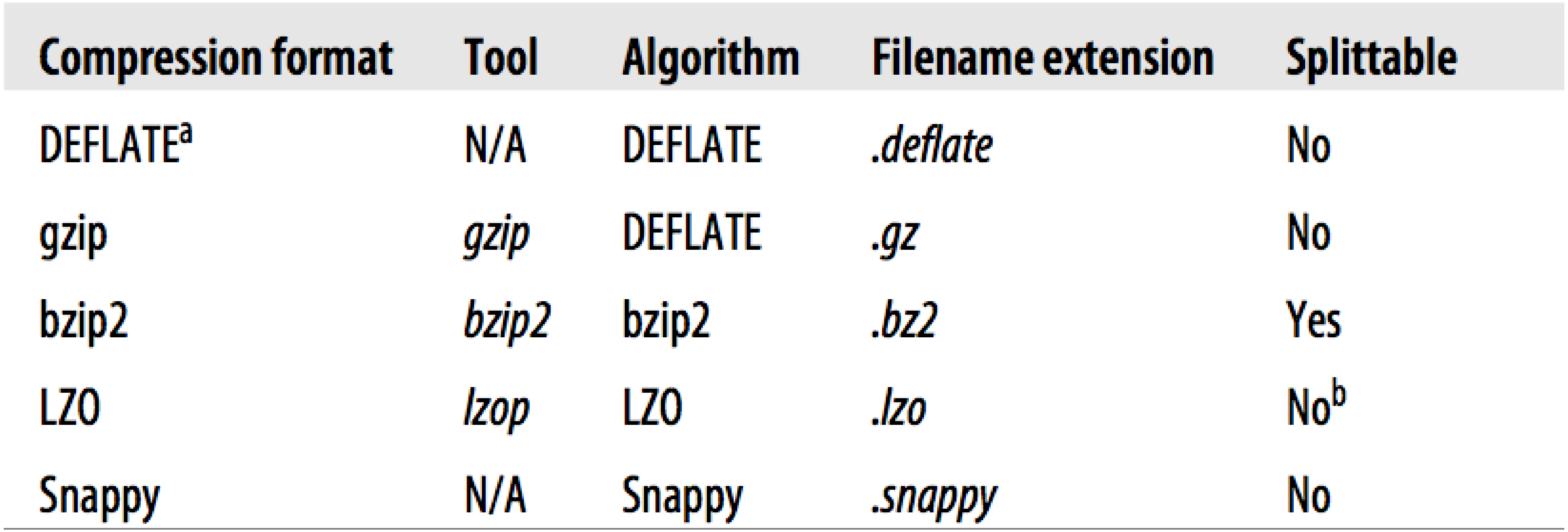
Hadoop IO

贝毅君 beiyj@zju.edu.cn

Hadoop 数据完整性

- 写入数据校验和-DataNode负责
- 读取数据校验和-client负责
- 后台校验和-DataNode后台自动运行
- 数据副本机制-Hadoop平台提供

Hadoop 压缩格式



Compression format	Tool	Algorithm	Filename extension	Splittable
DEFLATE[a]	N/A	DEFLATE	*.deflate*	No
gzip	*gzip*	DEFLATE	*.gz*	No
bzip2	*bzip2*	bzip2	*.bz2*	Yes
LZO	*lzop*	LZO	*.lzo*	No[b]
Snappy	N/A	Snappy	*.snappy*	No

7 Hadoop 压缩

mapreduce 中的输出结果压缩

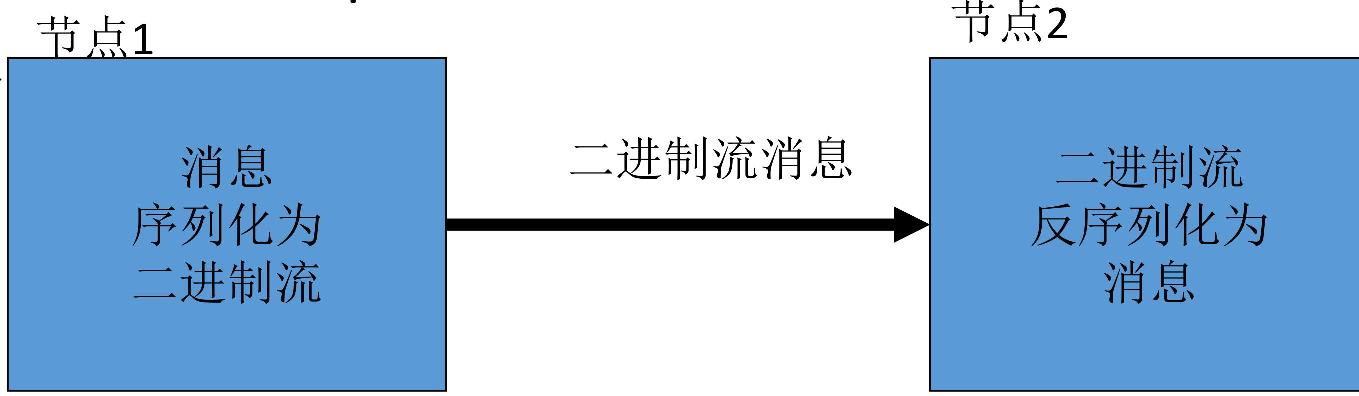
```
Job job = new Job();
job.setJarByClass(MaxTemperature.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

FileOutputFormat.setCompressOutput(job, true);
FileOutputFormat.setOutputCompressorClass(job, GzipCodec.class);

job.setMapperClass(MaxTemperatureMapper.class);
job.setCombinerClass(MaxTemperatureReducer.class);
job.setReducerClass(MaxTemperatureReducer.class);

System.exit(job.waitForCompletion(true) ? 0 : 1);
```

Hadoop序列化

- 序列化（Serialization）是指把结构化对象转化为字节流。
- 反序列化（Deserialization）是序列化的逆过程。即把字节流转回结构化对象。
- Java序列化（java.io.Serializable）

- Hadoop序列化格式特点：
- 紧凑：高效使用存储空间。
- 快速：读写数据的额外开销小
- 可扩展：可透明地读取老格式的数据
- 互操作：支持多语言的交互

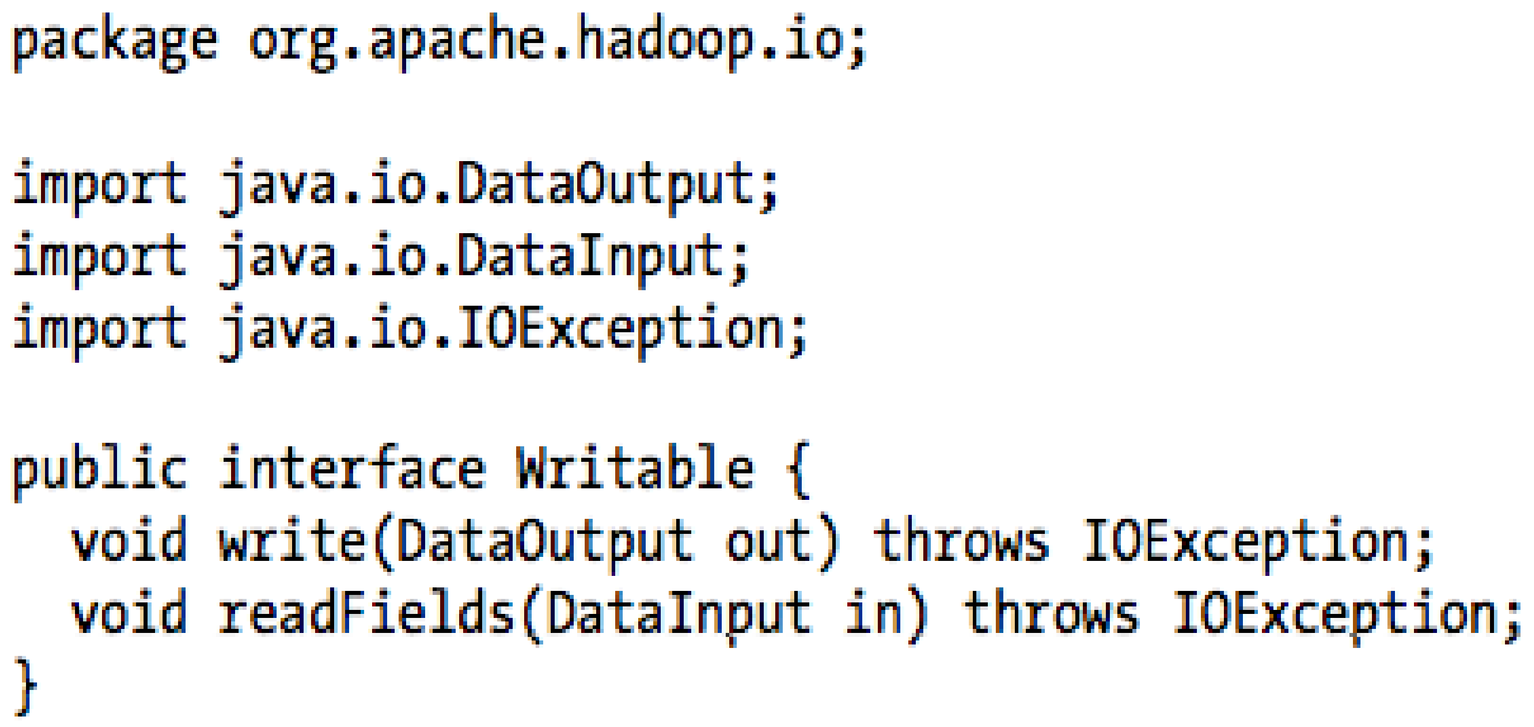
- 序列化在分布式环境的两大作用：
- 进程间通信，永久存储。
- Hadoop节点间通信。



3 Writable接口

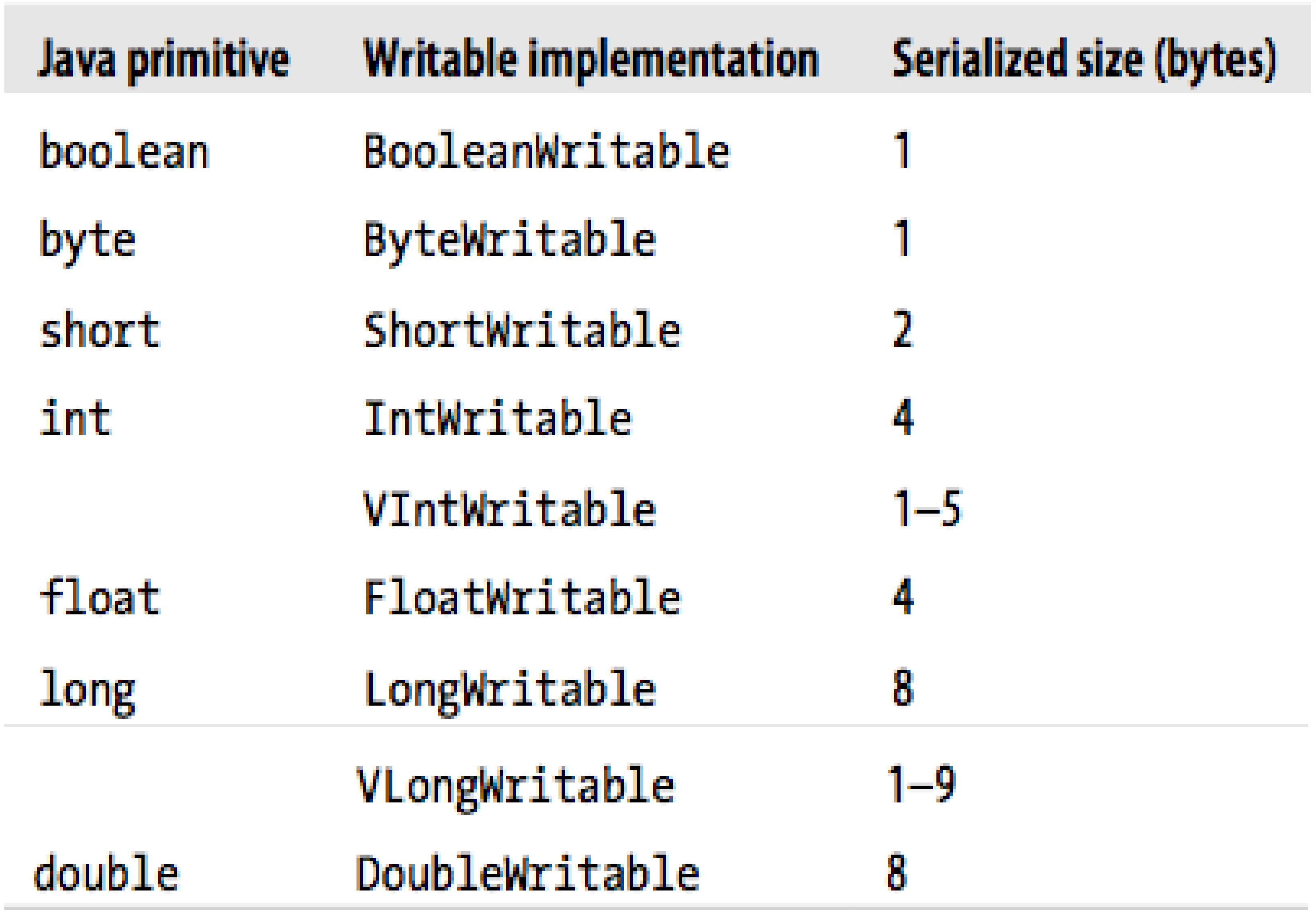
Writable接口, 是根据 DataInput 和 DataOutput 实现的简单、有效的序列化对象.

MR的任意Key和Value必须实现Writable接口.



```
package org.apache.hadoop.io;

import java.io.DataOutput;
import java.io.DataInput;
import java.io.IOException;

public interface Writable {
  void write(DataOutput out) throws IOException;
  void readFields(DataInput in) throws IOException;
}
```

3 Writable接口

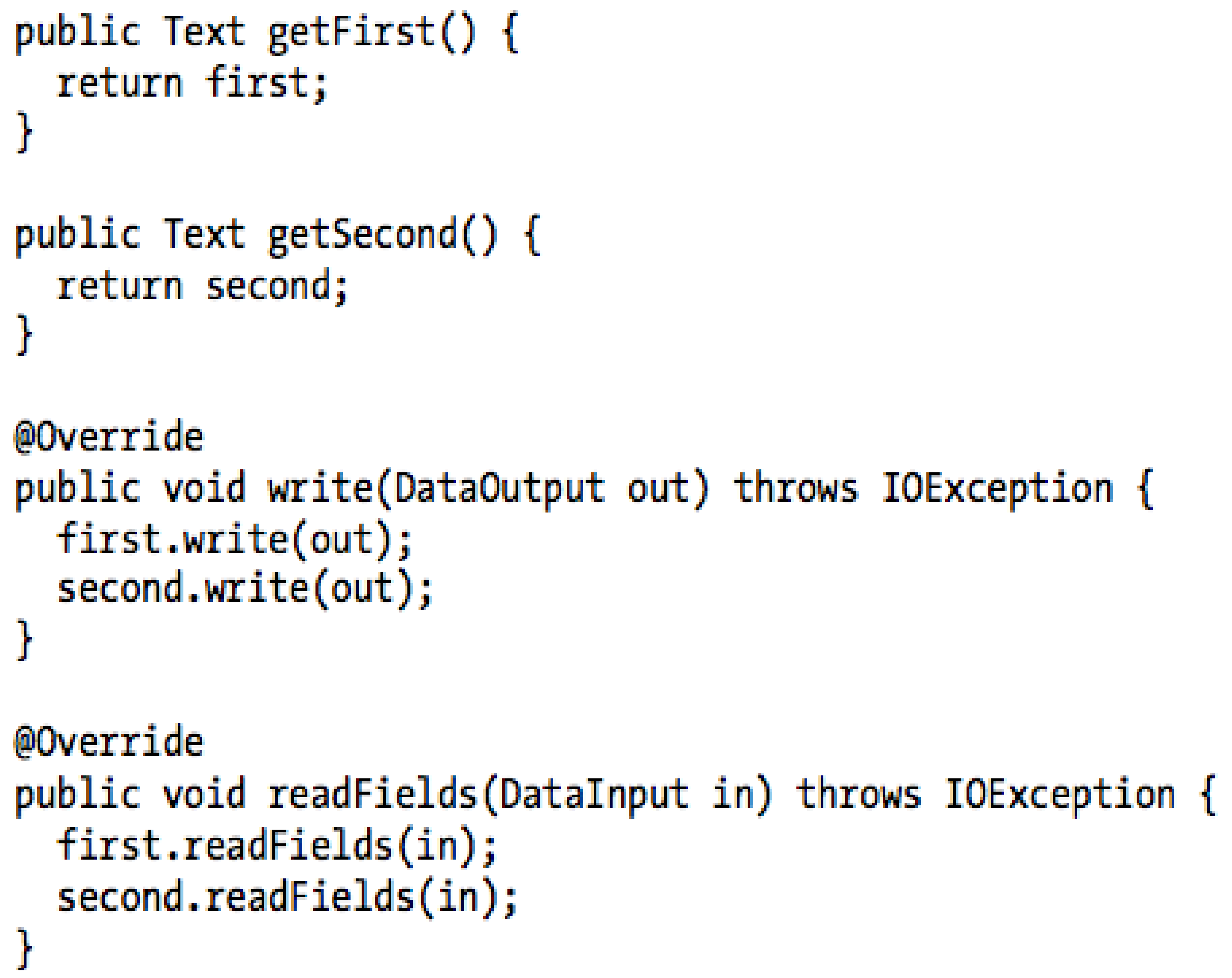


Java primitive	Writable implementation	Serialized size (bytes)
boolean	BooleanWritable	1
byte	ByteWritable	1
short	ShortWritable	2
int	IntWritable	4
	VIntWritable	1–5
float	FloatWritable	4
long	LongWritable	8
	VLongWritable	1–9
double	DoubleWritable	8

自定义Writable

```
public class TextPair implements WritableComparable<TextPair> {

  private Text first;
  private Text second;

  public TextPair() {
    set(new Text(), new Text());
  }

  public TextPair(String first, String second) {
    set(new Text(first), new Text(second));
  }

  public TextPair(Text first, Text second) {
    set(first, second);
  }

  public void set(Text first, Text second) {
    this.first = first;
    this.second = second;
  }
```

自定义Writable



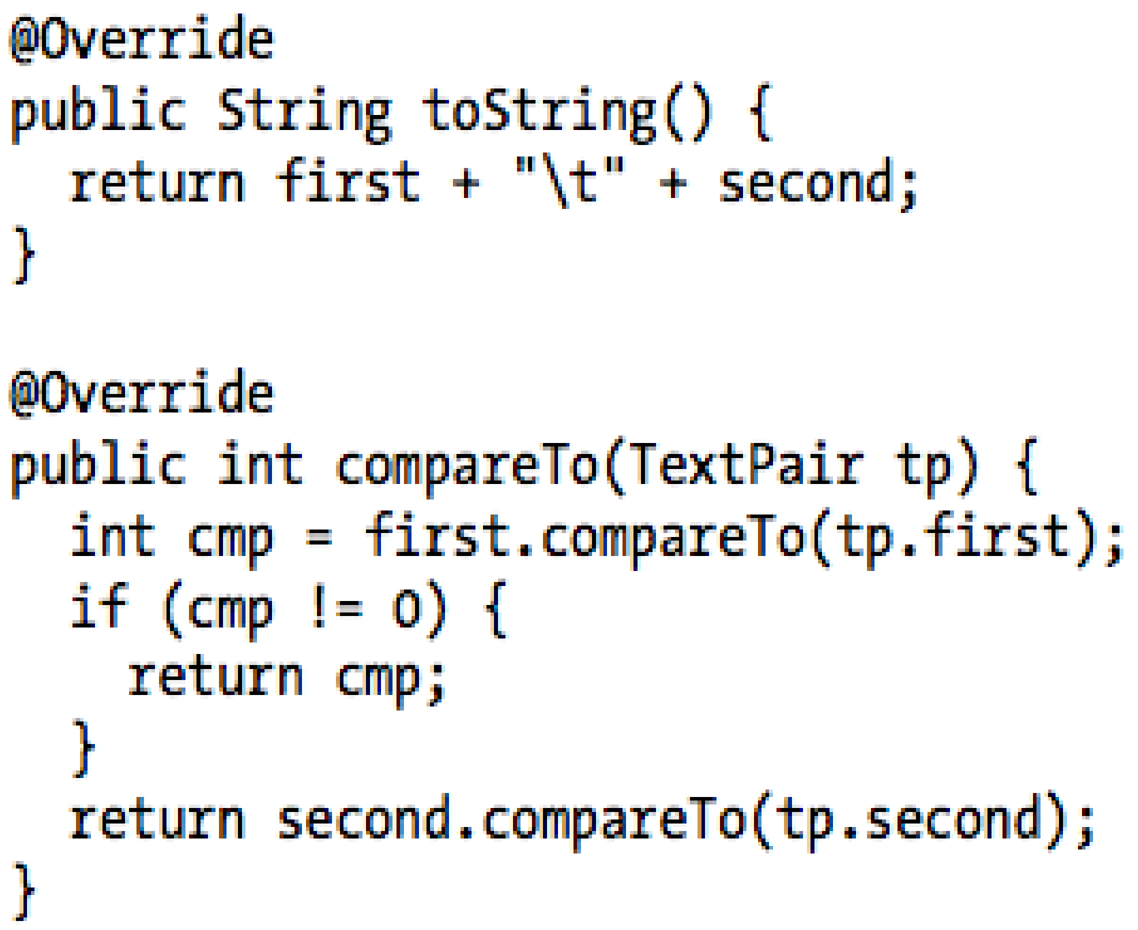
```
public Text getFirst() {
  return first;
}

public Text getSecond() {
  return second;
}

@Override
public void write(DataOutput out) throws IOException {
  first.write(out);
  second.write(out);
}

@Override
public void readFields(DataInput in) throws IOException {
  first.readFields(in);
  second.readFields(in);
}
```

7 Writable接口

自定义Writable

```
@Override
public int hashCode() {
  return first.hashCode() * 163 + second.hashCode();
}

@Override
public boolean equals(Object o) {
  if (o instanceof TextPair) {
    TextPair tp = (TextPair) o;
    return first.equals(tp.first) && second.equals(tp.second);
  }
  return false;
}
```

自定义Writable



```
@Override
public String toString() {
  return first + "\t" + second;
}

@Override
public int compareTo(TextPair tp) {
  int cmp = first.compareTo(tp.first);
  if (cmp != 0) {
    return cmp;
  }
  return second.compareTo(tp.second);
}
```

SequenceFile

- SequenceFile 使用键值对持久化保存数据
- SequenceFile 非常适合日志文件
- SequenceFile 也可以作为小文件的容器

7 基于文件数据结构

SequenceFile 写

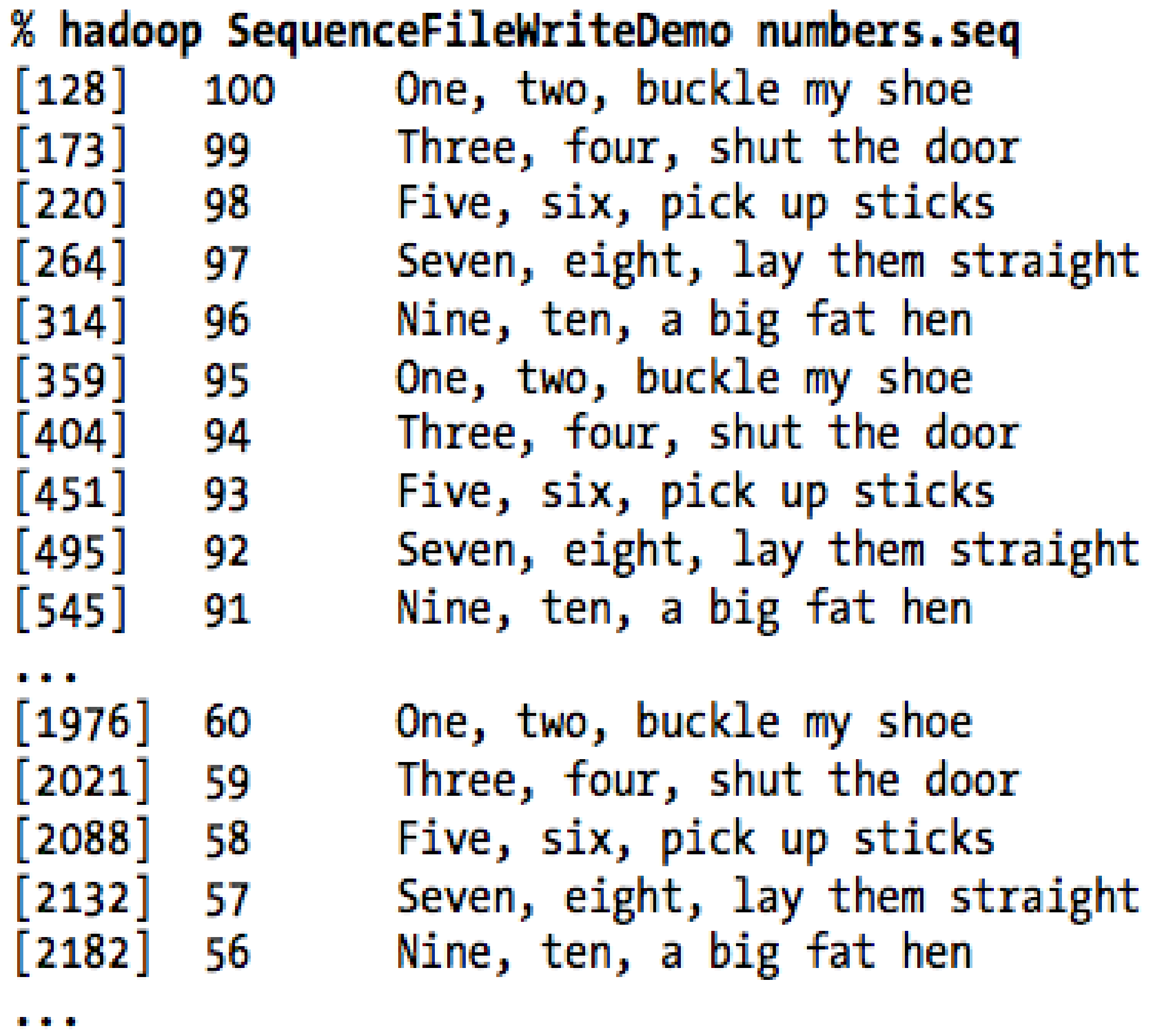
```
public class SequenceFileWriteDemo {

  private static final String[] DATA = {
    "One, two, buckle my shoe",
    "Three, four, shut the door",
    "Five, six, pick up sticks",
    "Seven, eight, lay them straight",
    "Nine, ten, a big fat hen"
  };

  public static void main(String[] args) throws IOException {
    String uri = args[0];
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(uri), conf);
    Path path = new Path(uri);

    IntWritable key = new IntWritable();
```

7 基于文件数据结构

SequenceFile 写

```
Text value = new Text();
SequenceFile.Writer writer = null;
try {
  writer = SequenceFile.createWriter(fs, conf, path,
      key.getClass(), value.getClass());

  for (int i = 0; i < 100; i++) {
    key.set(100 - i);
    value.set(DATA[i % DATA.length]);
    System.out.printf("[%s]\t%s\t%s\n", writer.getLength(), key, value);
    writer.append(key, value);
  }
} finally {
  IOUtils.closeStream(writer);
}
```

SequenceFile 文件信息

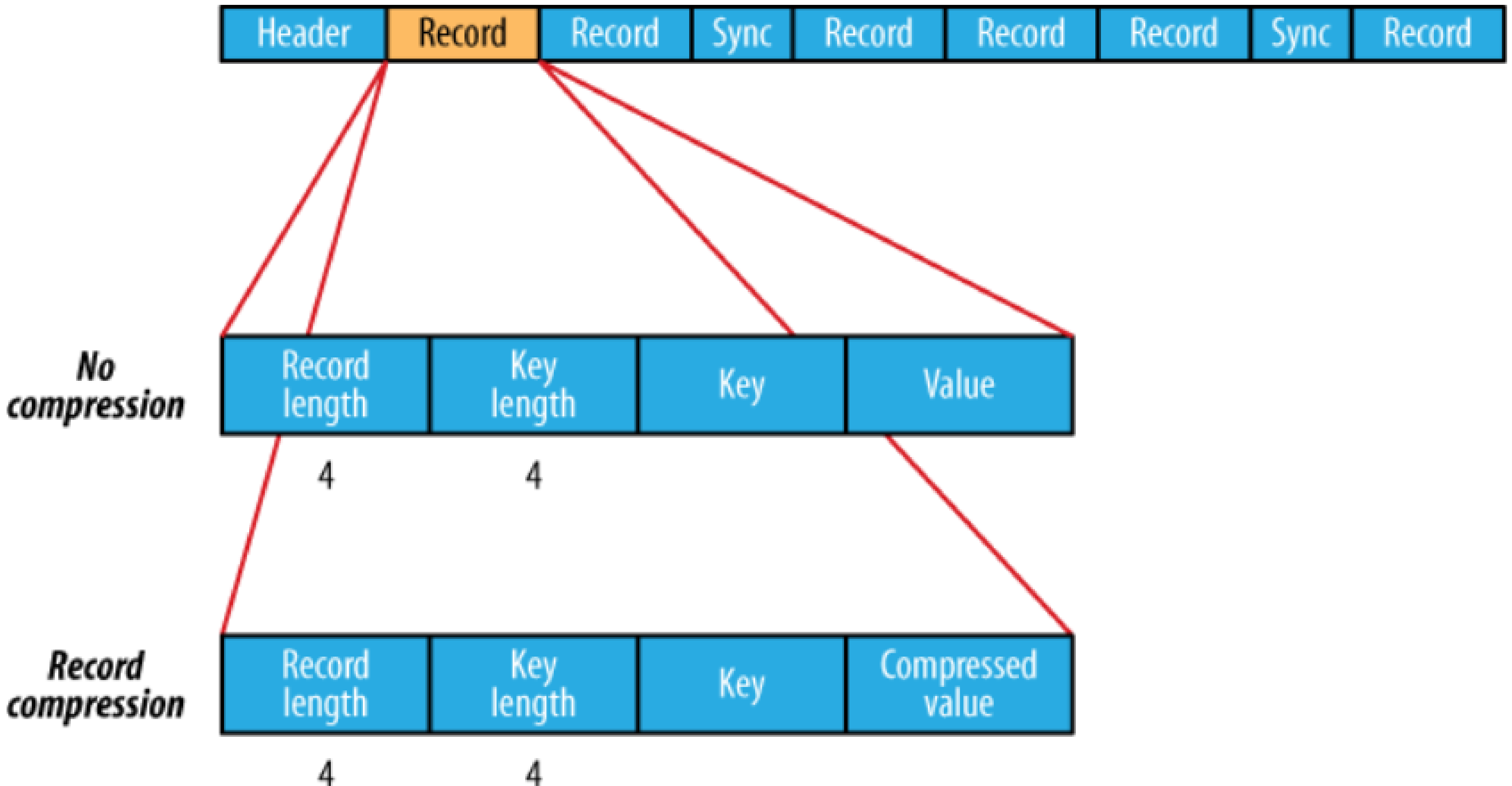


```
% hadoop SequenceFileWriteDemo numbers.seq
[128]   100     One, two, buckle my shoe
[173]   99      Three, four, shut the door
[220]   98      Five, six, pick up sticks
[264]   97      Seven, eight, lay them straight
[314]   96      Nine, ten, a big fat hen
[359]   95      One, two, buckle my shoe
[404]   94      Three, four, shut the door
[451]   93      Five, six, pick up sticks
[495]   92      Seven, eight, lay them straight
[545]   91      Nine, ten, a big fat hen
...
[1976]  60      One, two, buckle my shoe
[2021]  59      Three, four, shut the door
[2088]  58      Five, six, pick up sticks
[2132]  57      Seven, eight, lay them straight
[2182]  56      Nine, ten, a big fat hen
...
```

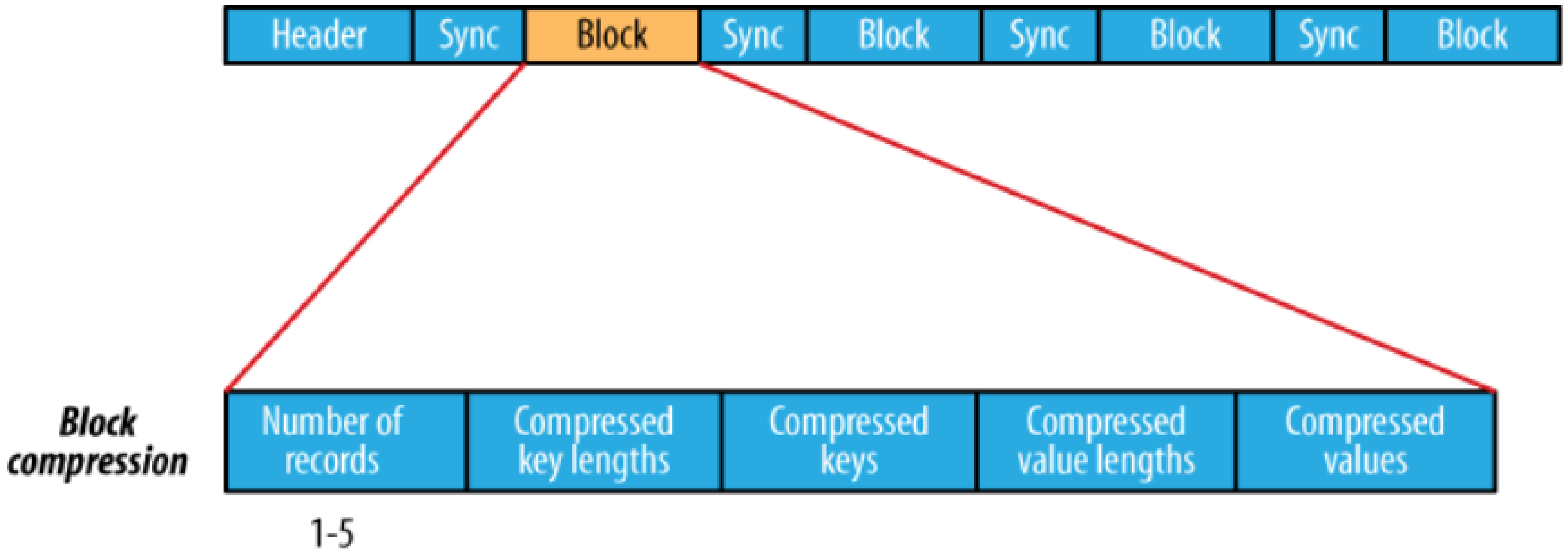
SequenceFile 读

```
public class SequenceFileReadDemo {

  public static void main(String[] args) throws IOException {
    String uri = args[0];
    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);
      long position = reader.getPosition();
      while (reader.next(key, value)) {
        String syncSeen = reader.syncSeen() ? "*" : "";
        System.out.printf("[%s%s]\t%s\t%s\n", position, syncSeen, key, value);
        position = reader.getPosition(); // beginning of next record
      }
    } finally {
      IOUtils.closeStream(reader);
```

SequenceFile 格式（普通压缩）



SequenceFile 文件格式（块压缩）



- 块压缩是指一次性压缩多条记录，利用记录间的相似性进行压缩，效率更高

MapFile

- MapFile 是经过排序的SequenceFile
- MapFile 有索引
- MapFile 可以视为是Java.util.Map的持久化形势

2017•08

THE
END