HADOOP-1722 and typed bytes

Klaas Bosteels

HADOOP-1722

- Hadoop Streaming maps/reduces by
  - running a given executable as a separate process,
  - writing the input to the process' *stdin* , and
  - reading the output from the process' *stdout.*
- HADOOP-1722
  - abstracts this reading/writing, which allows it to
  - add two alternative communication formats:
    - raw bytes: *<4 byte length> <bytes>*
    - typed bytes: *<1 byte typecode> <bytes>*
- Usage:
  - cmdline option "-io *<identifier>* ", or
  - set config properties manually for finer control.

InputWriter and *OutputReader*

```
class InputWriter<K,V>
{
  void initialize(...);

  void writeKey(K key);

  void writeValue(V value);
}
```

Disclaimer: omitted lots of details (like "public" and "abstract" keywords and throws declarations) in order to improve clarity.

```
class OutputReader<K,V>
{
  void initialize(...);

  boolean readKeyValue();

  K getCurrentKey();

  V getCurrentValue();

  String getLastOutput();
}
```

IdentifierResolver

```
class IdentifierResolver {

  void resolve(String id)
  getInputWriterClass()
  getOutputReaderClass()

  getOutputKeyClass()
  getOutputValueClass()

  // also protected setters
}
```

- External code can add new identifiers by extending this class and overriding *resolve()*
- Default identifiers:
  - "text"
    - *TextInputWriter*
    - *TextOutputReader*
    - *Text* for output classes
  - "rawbytes"
    - *RawBytesInputWriter*
    - *RawBytesOutputReader*
    - *BytesWritable*
  - "typedbytes"
    - *TypedBytesInputWriter*
    - *TypedBytesOutputReader*
    - *TypedBytesWritable*

Typed bytes

- Typed bytes = typed communication format that is
  - easy to implement, and
  - fast to (de)serialize.
- Allows Hadoop Streaming programs to:
  - easily consume sequence files, since
    - all common *Writable* objects are converted,
    - including *Record IO* objects.
  - read different kinds of inputs simultaneously;
    - you can even consume both text and seq files
    - via *AutoInputFormat* (also in HADOOP-1722).
- Can also be used as a binary dumping format:
  - hadoop jar <*Streaming JAR*> dumptb <*DFS path*> > dump.tb
  - hadoop jar <*Streaming JAR*> loadtb <*DFS path*> < dump.tb

TypedBytesInputWriter

```
TypedBytesWritbleOutput tbwOut;
TypedBytesOutput tbOut;

writeKey(Object key) {
  writeTypedBytes(key);
}

writeValue(Object val) {
  writeTypedBytes(val);
}

writeTypedBytes(Object obj) {
  if (obj instanceof Writable) {
    tbwOut.write((Writable) obj);
  } else {
    tbOut.write(obj);
  }
}
```

<u>TypedBytesWritableOutput:</u>

```
write(Writable w) {
  if (w instanceof TypedBytesWritable) {
    writeTypedBytes(...);
  } else if (w instanceof BytesWritable) {
    writeBytes(...);
  } else if (w instanceof ByteWritable) {
    writeByte(...);
  } else if (w instanceof BooleanWritable) {
    ...
```

<u>TypedBytesOutput:</u>

```
write(Object obj) {
  if (obj instanceof Buffer) {
    writeBytes(((Buffer) obj).get());
  } else if (obj instanceof Byte) {
    writeByte((Byte) obj);
  } else if (obj instanceof Boolean) {
    ...
```

Main motivation: Dumbo

- Dumbo = Python module that makes Streaming easy
- Speed not crucial, but becoming more important:
  - originally only intended for one-off jobs, but
  - also used for jobs that run regularly now.
- Communication format:
  - <u>before</u>: *repr/eval* + conversion classes in Java
  - <u>now</u>: typed bytes
- Timings for *IP count* program on 300gigs of weblogs:
  - <u>Java</u>: *8 minutes*
  - <u>Dumbo with typed bytes</u>: *10 minutes*
  - <u>Hive</u>: *13 minutes*
  - <u>Dumbo without typed bytes</u>: *16 minutes*

<u>Disclaimer:</u> The Java program used the slow *split()* method and might have combined less.