Home · eishay/jvm-serializers Wiki · GitHub

github.com/eishay/jvm-serializers/wiki

eishay

Clone this wiki locally

Test Platform

OS:Windows 8.1
JVM:Oracle Corporation 1.7.0_75
CPU:Intel64 Family 6 Model 60 Stepping 3, GenuineIntel os-arch:AMD64
Cores (incl HT):4

Disclaimer

This test focusses on en/decoding of a cyclefree data structure, but the featureset of the libraries compared differs a lot:

- some serializers support cycle detection/object sharing others just write non-cyclic tree structures
- some include full metadata in serialized output, some don’t
- some are cross platform, some are language specific
- some are text based, some are binary,
- some support versioning forward/backward, both, some don’t

(See ToolBehavior)
Other test data will yield different results (e.g. adding a non ascii char to every string :-)). However the results give a raw estimation of library performance.

Serializers (no shared refs)

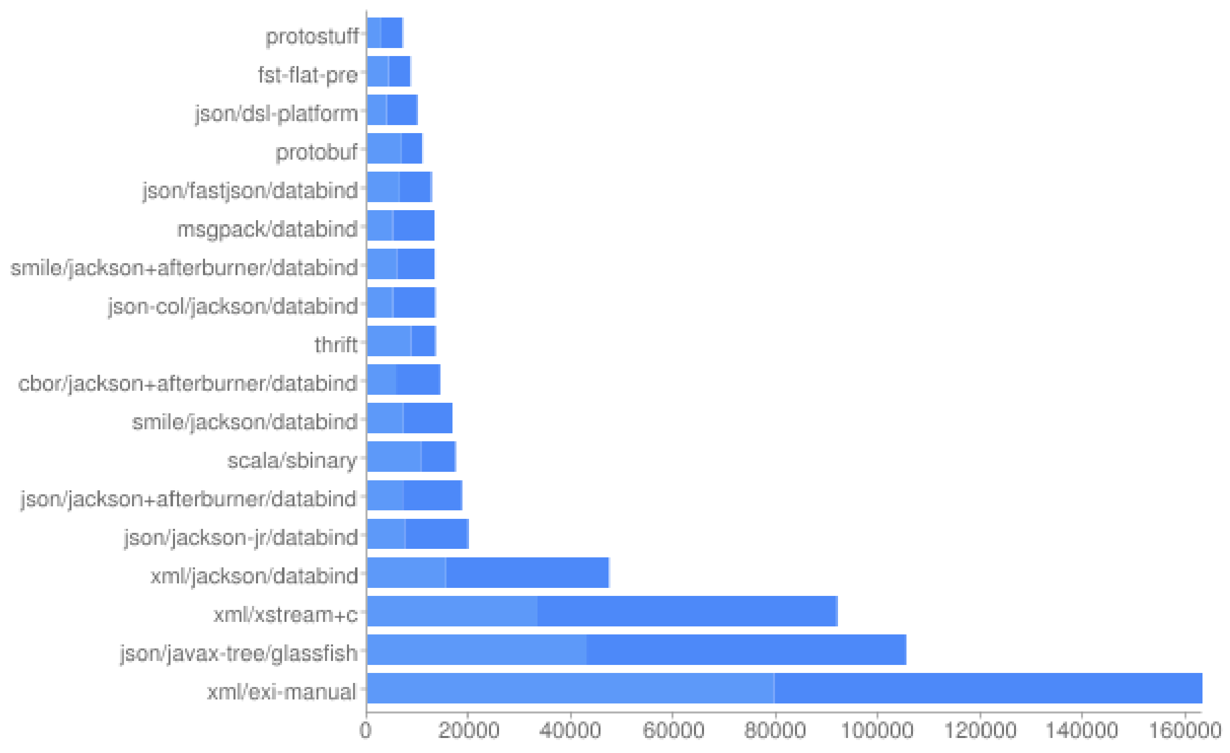
Benchmarks serializers

- Only cycle free tree structures. An object referenced twice will be serialized twice.

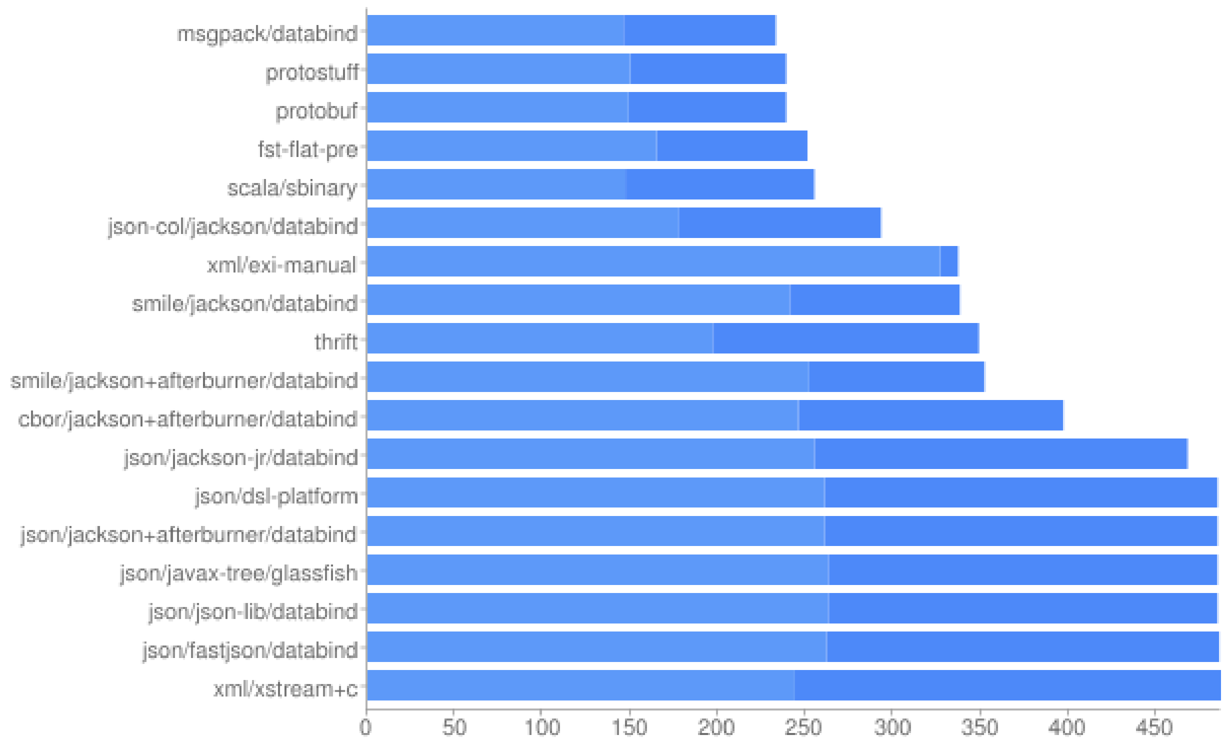
- no manual optimizations.

- schema is known in advance (pre registration or even class generation). (Not all might make use of that)

Ser Time+Deser Time (ns)



Size, Compressed size [light] in bytes



```
 create ser deser total size dfl

protostuff 438 2735 4154 6889 239 150

fst-flat-pre 332 4199 4234 8433 251 165

json/dsl-platform 290 3806 6013 9818 485 261

protobuf 690 6679 4100 10779 239 149

json/fastjson/databind 326 6295 6248 12543 486 262

msgpack/databind 331 4907 8190 13097 233 146

smile/jacksonafterburner/databind 318 5674 7443 13117 352 252

json-col/jackson/databind 325 5035 8213 13248 293 178

thrift 701 8492 4818 13311 349 197

cbor/jackson+afterburner/databind 321 5560 8662 14223 397 246

smile/jackson/databind 332 7047 9593 16641 338 241

scala/sbinary 2645 10418 6856 17274 255 147

json/jackson+afterburner/databind 333 6802 11738 18540 485
261

json/jackson-jr/databind 312 7257 12478 19735 468 255

xml/jackson/databind 304 15381 31919 47300 683 286

xml/xstream+c 339 33291 58579 91871 487 244

json/javax-tree/glassfish 6684 42814 62570 105384 485 263

xml/exi-manual 327 79439 83859 163298 337 327

java-built-in 354 30048 180401 210449 889 514

json/protobuf 700 38627 285788 324415 488 253

json/json-lib/databind 310 107977 786497 894474 485 263
```

Full Object Graph Serializers

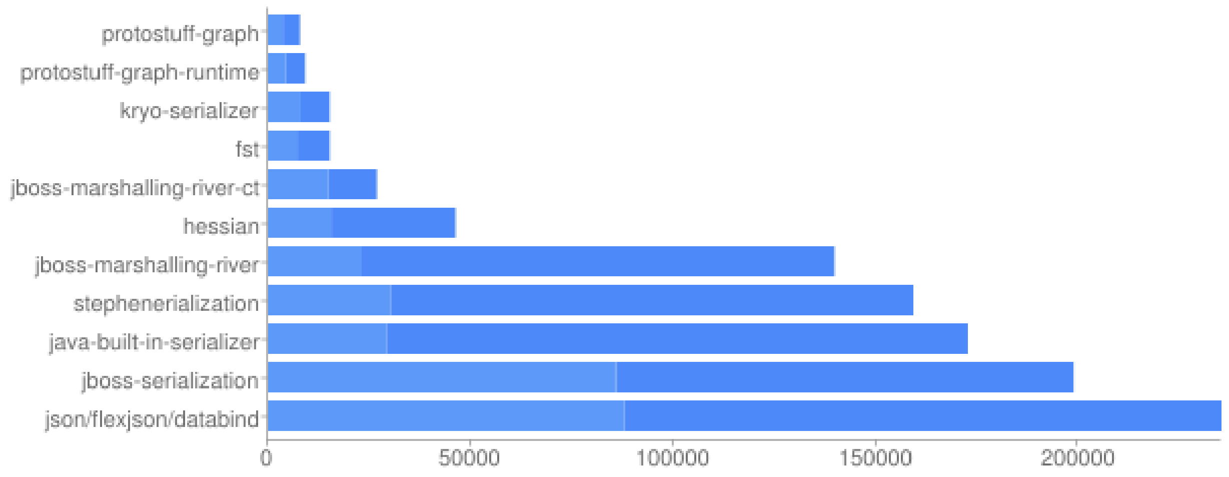
Contains serializer(-configurations)

- supporting full object graph write/read. Object graph may contain cycles. If an Object is referenced twice, it will be so after deserialization.

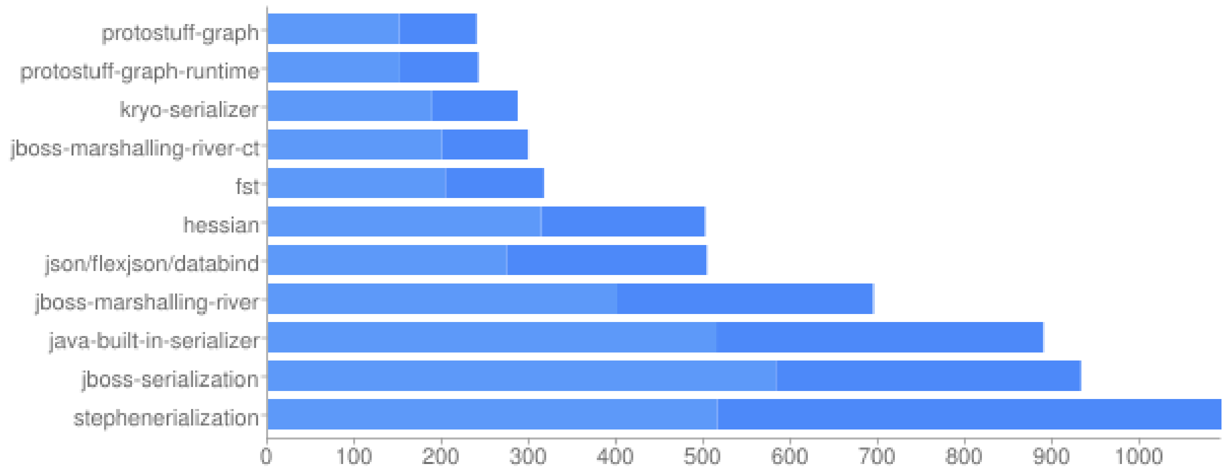
- nothing is known in advance, no class generation, no preregistering of classes. Everything is captured at runtime using e.g. reflection.

- note this usually cannot be used cross language, however JSON/XML formats may enable cross language deserialization.

Ser Time+Deser Time (ns)



Size, Compressed size [light] in bytes



```
 create ser deser total size +dfl

protostuff-graph 458 3567 4283 7851 239 150

protostuff-graph-runtime 373 4398 4745 9143 241 151

kryo-serializer 319 7996 7174 15170 286 188

fst 328 7062 8116 15179 316 203

jboss-marshalling-river-ct 375 14910 11937 26848 298
199

hessian 324 15564 30728 46292 501 313

jboss-marshalling-river 327 23039 116844 139884 694 400

stephenerialization 321 30420 128815 159234 1093 515

java-built-in-serializer 313 29427 143326 172752 889
514

jboss-serialization 320 85858 112995 198853 932 582

json/flexjson/databind 334 87989 147366 235355 503 273
```

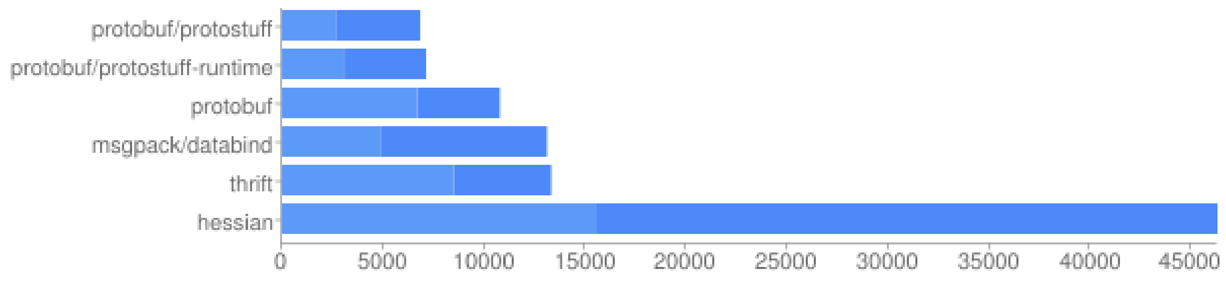
Cross Lang Binary Serializers

Contains serializer(-configurations)

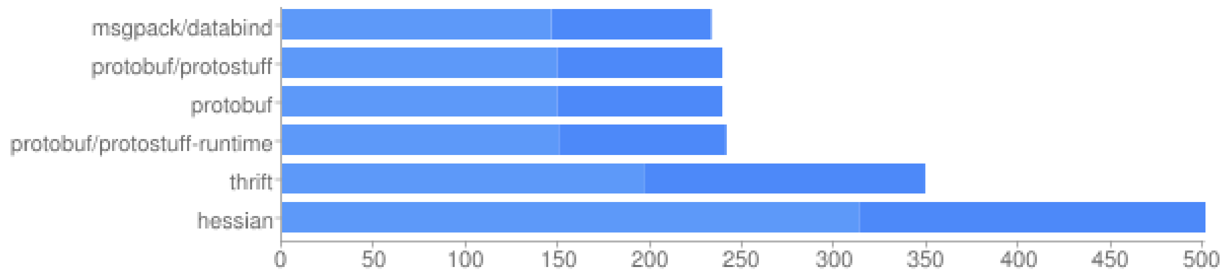
- Only cycle free tree structures. An object referenced twice will be serialized twice.

- schema is known in advance (pre registration, intermediate message description languages, class generation).

Ser Time+Deser Time (ns)



Size, Compressed size [light] in bytes

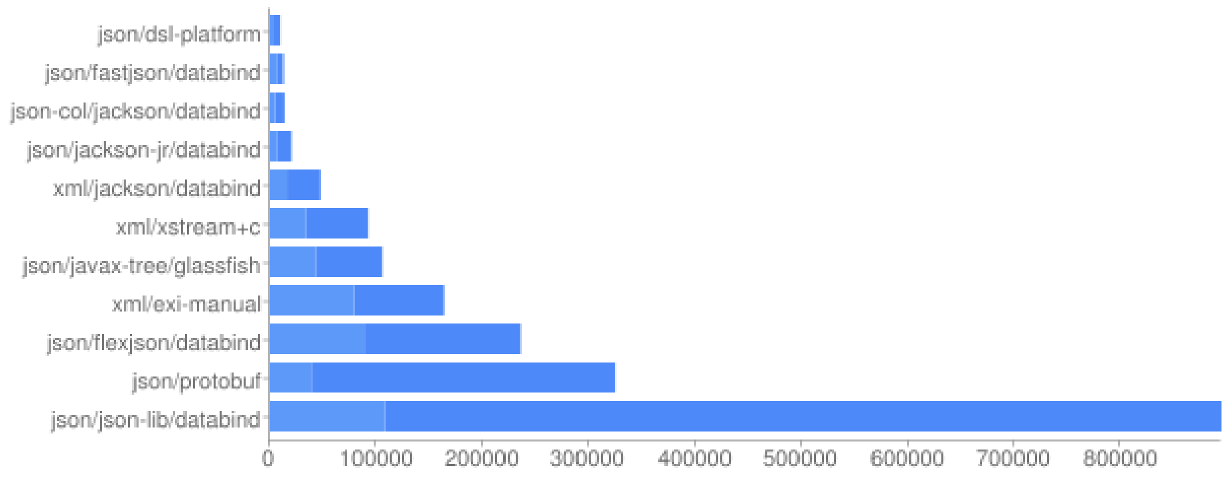


```
 create ser deser total size +dfl

protobuf/protostuff 435 2636 4197 6833 239 149

protobuf/protostuff-runtime 314 3029 4099 7128 241
150

protobuf 690 6679 4100 10779 239 149

msgpack/databind 331 4907 8190 13097 233 146

thrift 701 8492 4818 13311 349 197

hessian 324 15564 30728 46292 501 313
```

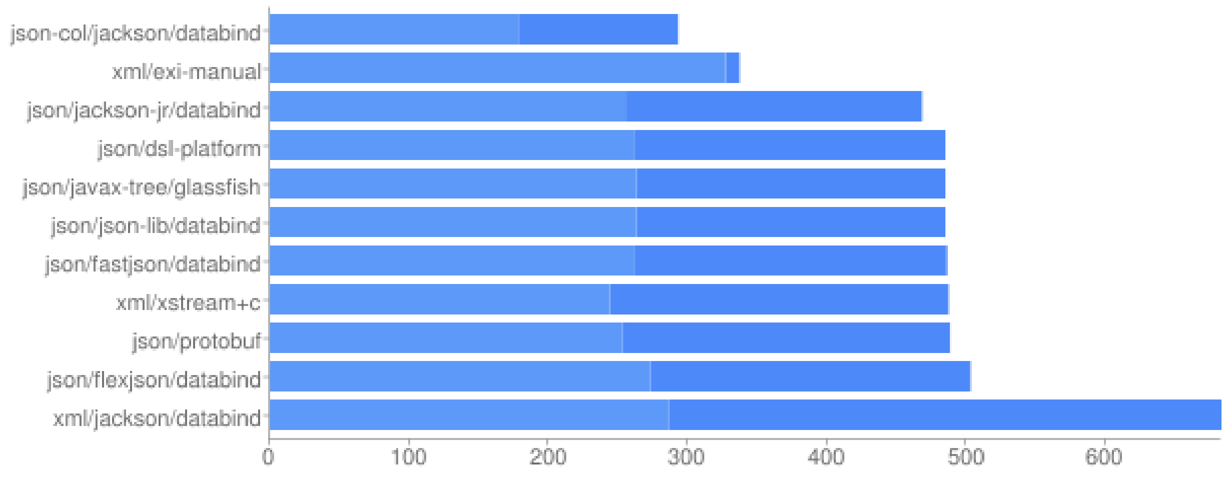
XML/JSon Serializers

- text format based. Usually can be read by anybody. Frequently inline schema inside data.
- Mixed regarding required preparation, object graph awareness (references).

Ser Time+Deser Time (ns)



Size, Compressed size [light] in bytes



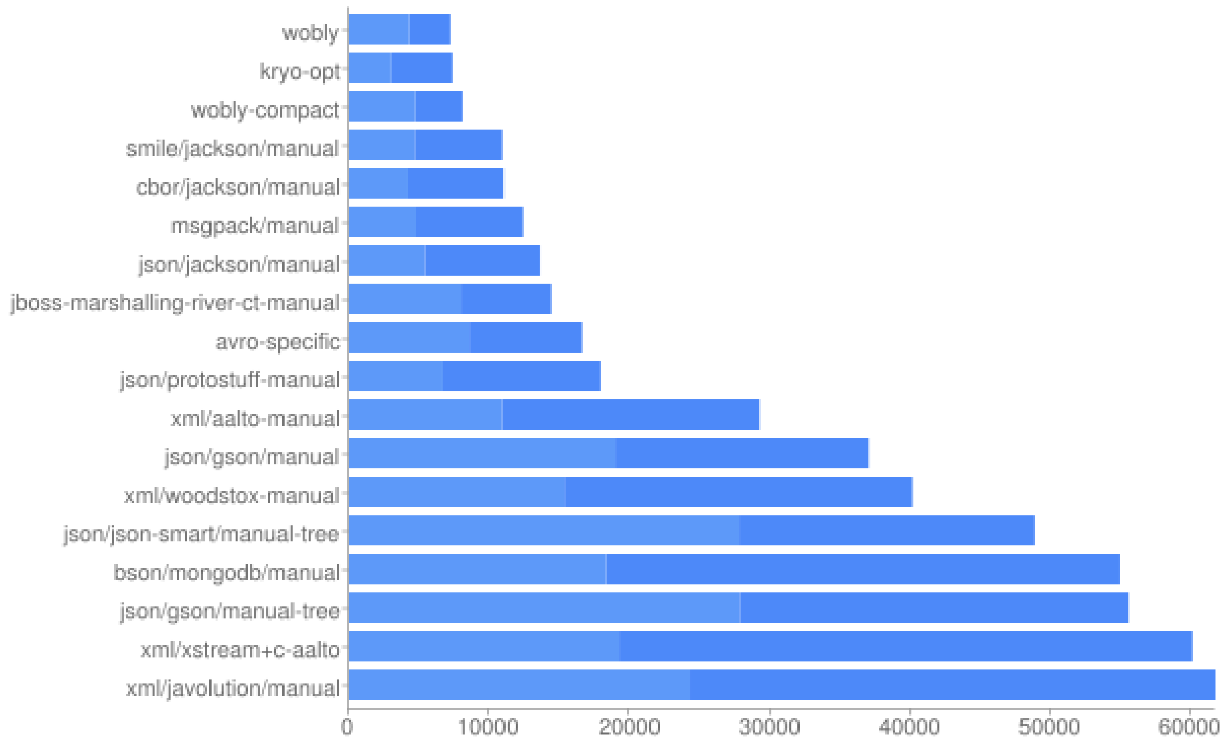
```
 create ser deser total size dfl

json/dsl-platform 290 3806 6013 9818 485 261

json/fastjson/databind 326 6295 6248 12543 486 262

json-col/jackson/databind 325 5035 8213 13248 293 178

json/jackson-jr/databind 312 7257 12478 19735 468 255

xml/jackson/databind 304 15381 31919 47300 683 286

xml/xstreamc 339 33291 58579 91871 487 244

json/javax-tree/glassfish 6684 42814 62570 105384 485
263

xml/exi-manual 327 79439 83859 163298 337 327

json/flexjson/databind 334 87989 147366 235355 503 273

json/protobuf 700 38627 285788 324415 488 253

json/json-lib/databind 310 107977 786497 894474 485 263
```

Manually optimized Serializers

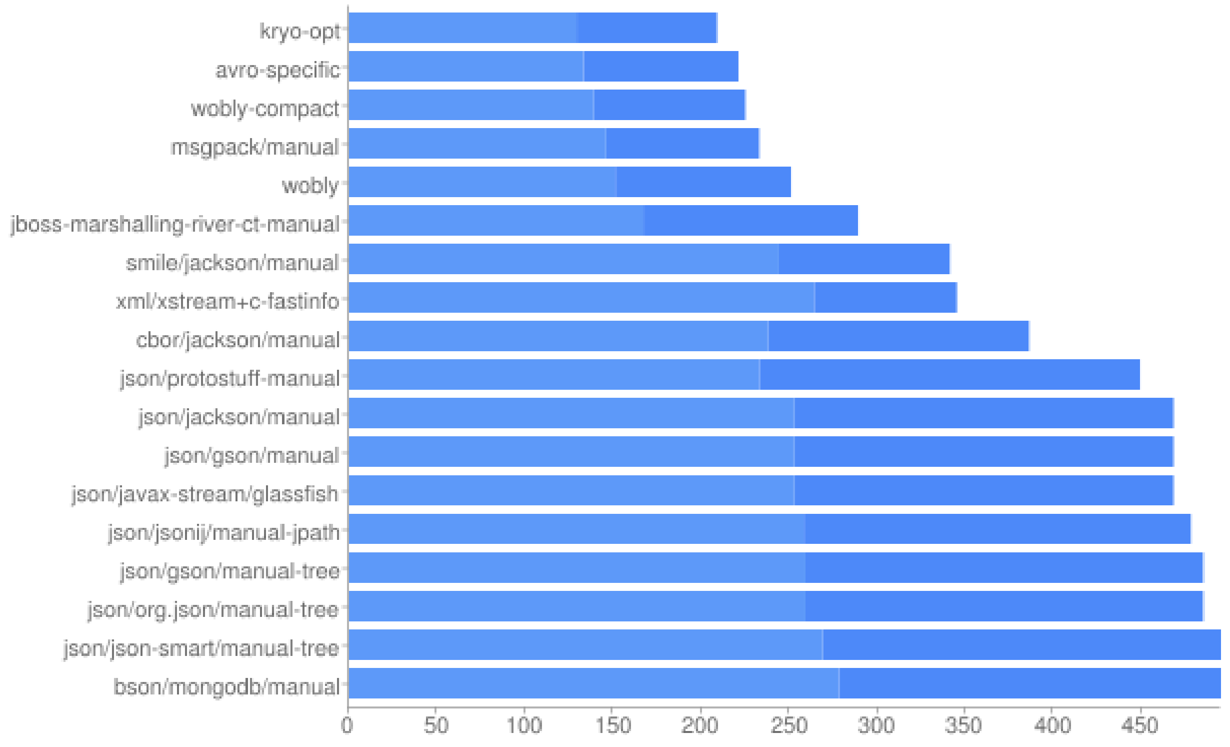
all flavours of manually optimized serializers. Handcoded and hardwired to exactly the benchmark's message structures.

- illustrates what's possible, at what level generic approaches can be optimized in case

Ser Time+Deser Time (ns)



Size, Compressed size [light] in bytes



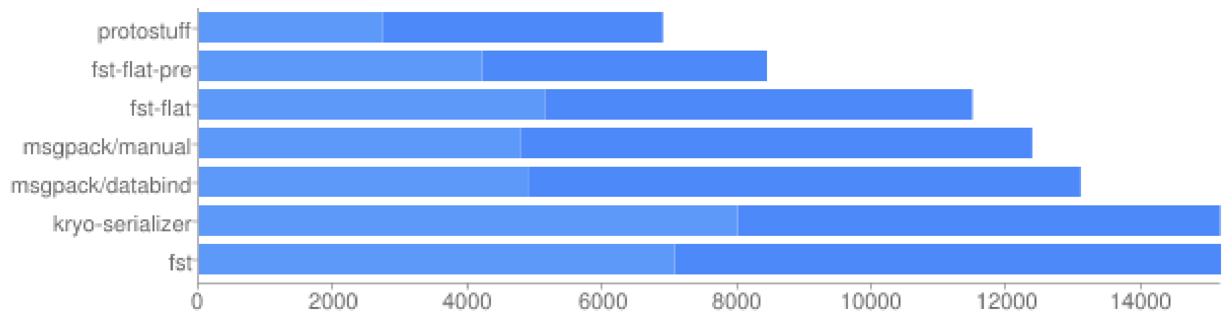
```
 create ser deser total size dfl

wobly 256 4291 2911 7202 251 151

kryo-opt 377 3002 4347 7349 209 129

wobly-compact 243 4725 3351 8077 225 139

smile/jackson/manual 319 4668 6268 10936 341 244

cbor/jackson/manual 324 4189 6812 11001 386 238

msgpack/manual 314 4768 7607 12375 233 146

json/jackson/manual 357 5407 8182 13589 468 253

jboss-marshalling-river-ct-manual 354 7956 6451 14407 289 
167

avro-specific 453 8543 8016 16560 221 133

json/protostuff-manual 366 6643 11256 17899 449 233

xml/aalto-manual 323 10918 18308 29226 653 304

json/gson/manual 338 18940 18087 37027 468 253

xml/woodstox-manual 329 15340 24776 40116 653 304

json/json-smart/manual-tree 337 27752 21073 48825 495 269

bson/mongodb/manual 366 18296 36611 54907 495 278

json/gson/manual-tree 358 27796 27732 55528 485 259

xml/xstreamc-aalto 325 19228 40851 60079 525 273

xml/javolution/manual 308 24278 37427 61704 504 263

xml/xstream+c-fastinfo 351 34776 40057 74833 345 264

xml/xstream+c-woodstox 319 23171 55257 78427 525 273

json/org.json/manual-tree 297 33397 46017 79415 485 259

json/javax-stream/glassfish 361 30221 52407 82627 468 253

json/jsonij/manual-jpath 356 182909 72334 255243 478 258
```

Cost of features

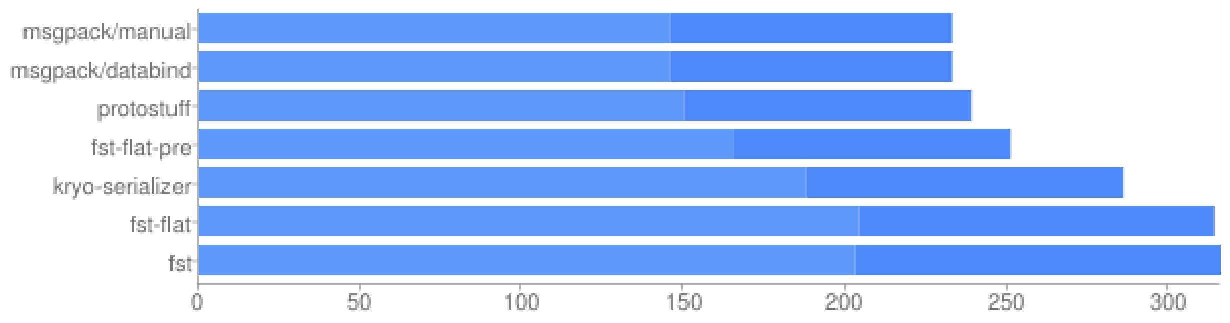
shows performance vs convenience of manually-selected libs.

- cycle free, schema known at compile time, manual optimization: kryo-manual, msgpack/manual
- cycle free, schema known at compile time: protostuff, fst-flat-pre, kryo-flat-pre. (note: protostuff uses class generation while the other two just require a list of classes to be written)
- cycle free, schema UNKNOWN at compile time: fst-flat, kryo-flat, protostuff-runtime, msgpack/databind
- full object graph awareness, schema UNKNOWN at compile time: fst, kryo.

Ser Time+Deser Time (ns)



Size, Compressed size [light] in bytes



```
 create ser deser total size +dfl

protostuff 438 2735 4154 6889 239 150

fst-flat-pre 332 4199 4234 8433 251 165

fst-flat 320 5130 6360 11490 314 204

msgpack/manual 314 4768 7607 12375 233 146

msgpack/databind 331 4907 8190 13097 233
146

kryo-serializer 319 7996 7174 15170 286 188

fst 328 7062 8116 15179 316 203
```

Full data

```
 create ser deser total size +dfl
protobuf/protostuff 435 2636 4197 6833 239 149
protostuff 438 2735 4154 6889 239 150
protobuf/protostuff-runtime 314 3029 4099 7128 241 150
wobly 256 4291 2911 7202 251 151
kryo-opt 377 3002 4347 7349 209 129
protostuff-graph 458 3567 4283 7851 239 150
wobly-compact 243 4725 3351 8077 225 139
fst-flat-pre 332 4199 4234 8433 251 165
protostuff-graph-runtime 373 4398 4745 9143 241 151
json/dsl-platform 290 3806 6013 9818 485 261
protobuf 690 6679 4100 10779 239 149
smile/jackson/manual 319 4668 6268 10936 341 244
cbor/jackson/manual 324 4189 6812 11001 386 238
fst-flat 320 5130 6360 11490 314 204
msgpack/manual 314 4768 7607 12375 233 146
json/fastjson/databind 326 6295 6248 12543 486 262
msgpack/databind 331 4907 8190 13097 233 146
smile/jackson+afterburner/databind 318 5674 7443 13117 352
252
json-col/jackson/databind 325 5035 8213 13248 293 178
thrift 701 8492 4818 13311 349 197
json/jackson/manual 357 5407 8182 13589 468 253
cbor/jackson+afterburner/databind 321 5560 8662 14223 397 246
jboss-marshalling-river-ct-manual 354 7956 6451 14407 289 167
kryo-serializer 319 7996 7174 15170 286 188
fst 328 7062 8116 15179 316 203
avro-specific 453 8543 8016 16560 221 133
smile/jackson/databind 332 7047 9593 16641 338 241
scala/sbinary 2645 10418 6856 17274 255 147
json/protostuff-manual 366 6643 11256 17899 449 233
```

```
json/jackson+afterburner/databind 333 6802 11738 18540 485
261
json/jackson-jr/databind 312 7257 12478 19735 468 255
jboss-marshalling-river-ct 375 14910 11937 26848 298 199
xml/aalto-manual 323 10918 18308 29226 653 304
json/gson/manual 338 18940 18087 37027 468 253
xml/woodstox-manual 329 15340 24776 40116 653 304
hessian 324 15564 30728 46292 501 313
xml/jackson/databind 304 15381 31919 47300 683 286
json/json-smart/manual-tree 337 27752 21073 48825 495 269
bson/mongodb/manual 366 18296 36611 54907 495 278
json/gson/manual-tree 358 27796 27732 55528 485 259
xml/xstream+c-aalto 325 19228 40851 60079 525 273
xml/javolution/manual 308 24278 37427 61704 504 263
xml/xstream+c-fastinfo 351 34776 40057 74833 345 264
xml/xstream+c-woodstox 319 23171 55257 78427 525 273
json/org.json/manual-tree 297 33397 46017 79415 485 259
json/javax-stream/glassfish 361 30221 52407 82627 468 253
xml/xstream+c 339 33291 58579 91871 487 244
json/javax-tree/glassfish 6684 42814 62570 105384 485 263
jboss-marshalling-river 327 23039 116844 139884 694 400
stephenerialization 321 30420 128815 159234 1093 515
xml/exi-manual 327 79439 83859 163298 337 327
java-built-in-serializer 313 29427 143326 172752 889 514
jboss-serialization 320 85858 112995 198853 932 582
java-built-in 354 30048 180401 210449 889 514
json/flexjson/databind 334 87989 147366 235355 503 273
json/jsonij/manual-jpath 356 182909 72334 255243 478 258
json/protobuf 700 38627 285788 324415 488 253
json/json-lib/databind 310 107977 786497 894474 485 263


 Effort Format Structure Misc
protobuf/protostuff CLASSES_KNOWN BIN_CROSSLANG FLAT_TREE [] protobuf + generated code
protostuff CLASSES_KNOWN BINARY FLAT_TREE [] generated code
protobuf/protostuff-runtime ZERO_KNOWLEDGE BIN_CROSSLANG FLAT_TREE [] protobuf +
reflection
wobly MANUAL_OPT BINARY FLAT_TREE []
kryo-opt MANUAL_OPT BINARY FLAT_TREE [] manually optimized
protostuff-graph CLASSES_KNOWN BINARY FULL_GRAPH [] graph + generated code
wobly-compact MANUAL_OPT BINARY FLAT_TREE []
fst-flat-pre CLASSES_KNOWN BINARY FLAT_TREE [] fst in unshared mode with
preregistered classes
protostuff-graph-runtime ZERO_KNOWLEDGE BINARY FULL_GRAPH [] graph + reflection
json/dsl-platform CLASSES_KNOWN JSON FLAT_TREE [] Serializes all properties.
protobuf CLASSES_KNOWN BIN_CROSSLANG FLAT_TREE []
smile/jackson/manual MANUAL_OPT BINARY FLAT_TREE []
cbor/jackson/manual MANUAL_OPT BIN_CROSSLANG FLAT_TREE []
fst-flat ZERO_KNOWLEDGE BINARY FLAT_TREE [] fst default, but unshared mode
msgpack/manual MANUAL_OPT BIN_CROSSLANG FLAT_TREE [] uses positional (column) layout
(instead of Maps std impl uses) to eliminate use of names
json/fastjson/databind ZERO_KNOWLEDGE JSON FLAT_TREE []
msgpack/databind CLASSES_KNOWN BIN_CROSSLANG FLAT_TREE [] uses positional (column)
layout (instead of Maps std impl uses) to eliminate use of names
smile/jackson+afterburner/databind ZERO_KNOWLEDGE BINARY FLAT_TREE [] uses bytecode
generation to reduce overhead
json-col/jackson/databind ZERO_KNOWLEDGE JSON FLAT_TREE [] uses positional (column)
```

```
layout to eliminate use of names
thrift CLASSES_KNOWN BIN_CROSSLANG FLAT_TREE []
json/jackson/manual MANUAL_OPT JSON FLAT_TREE []
cbor/jackson+afterburner/databind ZERO_KNOWLEDGE BINARY FLAT_TREE [] uses bytecode
generation to reduce overhead
jboss-marshalling-river-ct-manual MANUAL_OPT BINARY FULL_GRAPH [] full graph
preregistered classes, manual optimization
kryo-serializer ZERO_KNOWLEDGE BINARY FULL_GRAPH [] default
fst ZERO_KNOWLEDGE BINARY FULL_GRAPH [] default: JDK serialization drop-in-
replacement mode
avro-specific MANUAL_OPT BIN_CROSSLANG UNKNOWN []
smile/jackson/databind ZERO_KNOWLEDGE BINARY FLAT_TREE []
scala/sbinary MISC MISC UNKNOWN [] null
json/protostuff-manual MANUAL_OPT JSON FLAT_TREE [] json + manual
json/jackson+afterburner/databind ZERO_KNOWLEDGE BINARY FLAT_TREE [] uses bytecode
generation to reduce overhead
json/jackson-jr/databind ZERO_KNOWLEDGE JSON FLAT_TREE []
jboss-marshalling-river-ct CLASSES_KNOWN BINARY FULL_GRAPH [] full graph with
preregistered classes
xml/aalto-manual MANUAL_OPT XML UNKNOWN []
json/gson/manual MANUAL_OPT JSON FLAT_TREE []
xml/woodstox-manual MANUAL_OPT XML UNKNOWN []
hessian ZERO_KNOWLEDGE BIN_CROSSLANG FULL_GRAPH []
xml/jackson/databind ZERO_KNOWLEDGE XML FLAT_TREE []
json/json-smart/manual-tree MANUAL_OPT JSON FLAT_TREE []
bson/mongodb/manual MANUAL_OPT BIN_CROSSLANG FLAT_TREE []
json/gson/manual-tree MANUAL_OPT JSON FLAT_TREE []
xml/xstream+c-aalto MANUAL_OPT XML FLAT_TREE []
xml/javolution/manual MANUAL_OPT XML FLAT_TREE []
xml/xstream+c-fastinfo MANUAL_OPT XML FLAT_TREE []
xml/xstream+c-woodstox MANUAL_OPT XML FLAT_TREE []
json/org.json/manual-tree MANUAL_OPT JSON FLAT_TREE []
json/javax-stream/glassfish MANUAL_OPT JSON FLAT_TREE []
xml/xstream+c ZERO_KNOWLEDGE XML FLAT_TREE []
json/javax-tree/glassfish ZERO_KNOWLEDGE JSON FLAT_TREE []
jboss-marshalling-river ZERO_KNOWLEDGE BINARY FULL_GRAPH [] full graph zero knowledge
stephenerialization ZERO_KNOWLEDGE BINARY FULL_GRAPH [] null
xml/exi-manual ZERO_KNOWLEDGE XML UNKNOWN []
java-built-in-serializer ZERO_KNOWLEDGE BINARY FULL_GRAPH []
jboss-serialization ZERO_KNOWLEDGE BINARY FULL_GRAPH []
java-built-in ZERO_KNOWLEDGE BINARY FLAT_TREE []
json/flexjson/databind ZERO_KNOWLEDGE JSON FULL_GRAPH []
json/jsonij/manual-jpath MANUAL_OPT JSON FLAT_TREE []
json/protobuf CLASSES_KNOWN JSON FLAT_TREE []
json/json-lib/databind ZERO_KNOWLEDGE JSON FLAT_TREE []
```

Something went wrong with that request. Please try again.