ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ БИБЛИОТЕК СЕРИАЛИЗАЦИИ JAVA, JAVASCRIPT И PHP ДЛЯ XML, JSON

Ёрш А.О., Байданов А.М.
Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь
Дик С.К. – канд. физ.-мат. наук, доцент

С развитием информационных технологий растет потребность в обмене информацией. Простые веб-приложения требуют только минимальных ресурсов на стороне клиента, которым часто является веб-браузер. Мобильные приложения являются еще одним примером. В последние годы наблюдается их беспрецедентный рост. По сути, тонкие приложения просто отображают данные и позволяют манипулировать этими данными через графический интерфейс пользователя (GUI). Затем выполняются операции с данными, и данные сохраняются на сервере.

Целью данной статьи является сравнение форматов и библиотек, используемых для сериализации и десериализации данных, обычно с RESTful веб-сервисам, с точки зрения времени обработки и размера выходных данных. Протестированные форматы включают XML, JSON, MessagePack, Avro, Protocol Buffers и собственную сериализацию каждого из протестированных языков программирования. Сериализация и десериализация протестирована на PHP, Java и JavaScript с использованием 49 различных официальных и сторонних библиотек. Среда тестирования спроектирована таким образом, чтобы быть изолированной от остальной части операционной системы с помощью контейнеров Docker с нулевым снижением производительности в отличие от виртуализации. Результаты показывают огромные различия во времени обработки среди библиотек. Учитывая размер выходных данных, бинарные форматы с предопределенной схемой, такие как Avro и Protocol Buffers, обеспечивают наилучшую эффективность.

Измерение времени сериализации и десериализации проводилось в миллисекундах. Это легко понять: чем меньше число, тем быстрее библиотека (формат). Размер сериализованных данных измерялся в килобайтах (кБ); чем меньше размер, тем лучше библиотека (и формат). Один проход (де) сериализации занимает всего несколько микросекунд для достаточно больших входных данных. Измерение таких коротких интервалов может привести к большой ошибке и отклонению. По этой причине выполняется несколько измерений. Одни и те же данные многократно (де) сериализуются и измеряется общее время. Весь процесс также повторяется, чтобы получить несколько образцов. Оба цикла реализованы, например, в Java согласно коду.

Значения, представленные в результатах, представляют собой среднее время t нескольких прогонов всего внутреннего цикла в соответствии с формулой:

$$\bar{t} = \frac{\sum_{j=1}^{N} t_j}{N}, \quad (1)$$

где N - количество внешних повторений, а t_j - время внутреннего цикла.

Созданный эталонный тест написан на трех языках программирования: Java, PHP и JavaScript. Отдельные приложения разрабатываются в виде консольных программ (без графического интерфейса). Они могут работать практически на любой операционной системе, такой как GNU / Linux, Windows или MacOS. Все тесты были выполнены на процессоре Intel Core I7-2600k 3,4 ГГц с 8 ГБ оперативной памяти DDR3 и Debian Linux 8. Были использованы версии PHP 7.1, JDK 8 и NodeJS 7.7. Для того, чтобы выполнить весь тест просто без установки правильных версий PHP, Java и NodeJS (включая правильную конфигурацию), все приложения готовы к работе в контейнерах Docker. Благодаря Docker можно выполнять тестирование в изолированной и четко определенной среде без снижения производительности (что является обычным явлением в классической виртуализации).

На рисунке 1 показан график размера сериализованных данных в килобайтах для всех протестированных библиотек. Результаты показывают, что форматы Avro и Protobuf являются лучшими, за ними следуют MessagePack, JSON и XML.

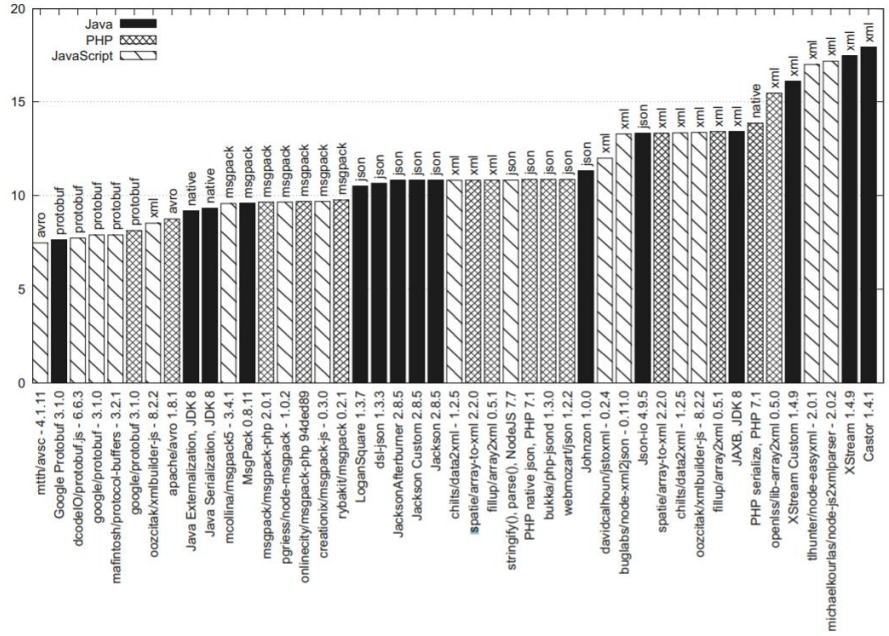

Рисунок 1 – Размер сериализованных данных

На рисунке 2 показан график времени сериализации. Чем меньше полученное значение, тем лучше. Поскольку разница между лучшим временем и худшим временем огромна, ось Y имеет логарифмическую шкалу для большей ясности.

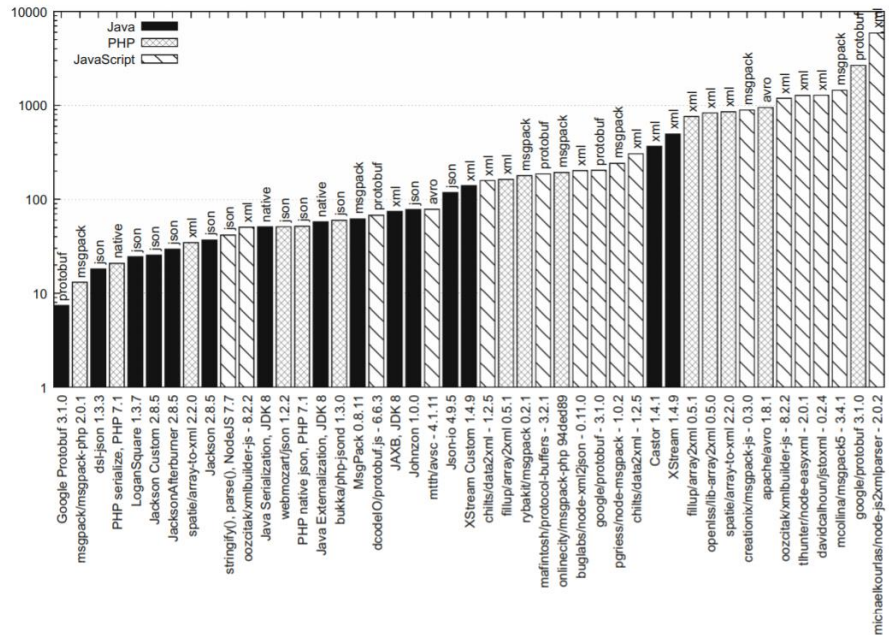

Рисунок 2 – Производительность сериализации (логарифмическая шкала)

Библиотека Message Pack для PHP имеет очень хороший результат - 13,1 мс. Библиотеки для JavaScript (NodeJS) сработали намного хуже, особенно библиотека pgriess / node-msgpack, имеющая 242 мс. У Java хорошая общая производительность. Особенно библиотеки JSON для Java, которые преодолевают JSON.stringify (), встроенную функцию JavaScript. Результаты также подтвердили, что JSON обычно быстрее XML.

Список использованных источников:

1. Wang, G.: Improving data transmission in web applications via the translation between XML and JSON. In: 2011 Third International Conference on Communications and Mobile Computing, pp. 182–185, April 2011.

2. Maeda, K.: Performance evaluation of object serialization libraries in XML, JSON and binary formats. In: 2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), pp. 177– 182, May 2012.