[Json框架选型]Android开发中应该使用哪一种主流json框架？

前言

前段时间@寒江不钓同学针对国内Top500和Google Play Top200 Android应用做了全面的分析（具体分析报告见文末的参考资料），其中有涉及到对主流应用使用json框架Gson、FastJson、Jackson的统计，具体情况如下：

三种json框架使用情况			
	Gson	FastJson	Jackson
国内前500	227	87	26
国际前200	57	0	15

可见无论是在国内还是国际上，有很多应用会用到json框架，其中谷歌提供的gson框架是被使用得最多的，老牌json框架Jackson属于小众，阿里出品的FastJson在国内的使用情况仅次于Gson，但在国际上却基本没有APP在使用。

测试方案

被使用得最多并不代表就一定是最优方案，在使用json框架的过程中，我们最关注的应该是效率和易用性的问题，三种框架的具体实现各不相同，肯定会存在效率和易用性上的差异，针对这两方面的分析网络上的文章并没有比较权威、能够说服大家的观点(FastJson引用的测试结果可以参考，但不足以说服所有人)，本文结合工作中的实际情况，使用三种框架分别对bean转String、String转bean、List转String、String转List、Map转String、String转Map这几种情况进行实际测试，得出在耗时、CPU占用、内存变化、易用性上的测试结果。

为了让测试结果显得更为可信，有必要说明一下测试方案：

- 使用相同的测试环境：入参相同（数据来源、测试次数）、运行的环境相同；
- 每个框架都使用最新的版本，如果有Android定制版优先使用Android定制版（Gson:2.7、FastJson:1.1.52.android、jackson:jackson-all-1.9.9）；
- 为了避免单次测试可能存在的不确定性因素，每个框架的每次测试都是重复测试100次，然后算平均值；
- 为了观察不同量级的数据量对效率的影响，数据量可以人为设置，测试时分别从10的0次方到10的4次方，每次测试以10为倍数的增长方式（即1、10、100、1000、10000，根据实际情况，测试10万次就显得不是很必要了，如果数据量太大，可以考虑分多次获取和转换）观察不同框架在不同量级数据的表现情况；
- 考虑到生成数据也会导致耗时的问题，对每个框架的每次测试，100次的测试中只生成一次数据；
- 考虑到测试的可操作性，对每个框架的每次测试中，bean转String、String转bean、List转String、String转List、Map转String、String转Map都是一并进行的，所以对内存变化和CPU变化的测试结果是这六项测试的综合结果。

考虑到可读性和代码量的问题，直接提供链接会比贴代码好一点：

- ParseJson4Android

测试结果

详细的测试结果如下：

Gson、FastJson、Jackson三种框架性能测试结果															
	1倍数据量			10倍数据量			100倍数据量			1000倍数据量			10000倍数据量		
	Gson	FastJson	Jackson	Gson	FastJson	Jackson	Gson	FastJson	Jackson	Gson	FastJson	Jackson	Gson	FastJson	Jackson
bean→String（ms）	2	0	12	2	0	12	2	0	12	2	0	12	2	0	18
String→bean（ms）	2	0	22	2	0	22	2	0	22	2	0	22	2	0	27
List→String（ms）	4	0	38	5	0	39	14	6	44	118	61	88	1226	737	705
String→List（ms）	4	0	27	5	1	27	14	8	36	108	90	111	1152	1037	1056
Map→String（ms）	4	0	33	5	0	34	16	6	39	119	63	86	1282	769	710
String→Map（ms）	3	0	23	4	1	24	13	10	33	114	95	129	1305	1117	1440
内存变化（M）	1.62	2.09	1	3.62	1.54	0.48	1.72	1.89	1.92	3.8	2.69	4.55	22.48	24.39	35.23
CPU变化（百分比）	50%	20%	60%	65%	30%	55%	60%	65%	65%	65%	65%	65%	65%	65%	65%

结合本人实际操作和测试结果，可以得出如下结论：

Gson、FastJson、Jackson综合对比								
	原理	是否仍在维护	使用版本	包大小	效率	内存占用	CPU占用	易用性	说明
Gson	1、如果是通过GsonBuilder创建的Gson对象，那么就用自定义的TyperAdapter来完成json的解析； 2、如果是通过new Gson()创建的Gson对象，那么就用Java反射机制来完成json的解析。	是，最近一次提交是2个月前	2.7	227KB	在1000倍数据量级内，效率中等，在10000倍数据量级及以上，效率最差。	在1000倍数据量级内，内存占用低，峰值为3.62M，在10000倍数据量级及以上，内存占用率超过22M，并且存在明显的内存抖动。	高	简单	
FastJson	1、bean to JSON ：利用反射找到对象类的所有Get方法，然后把"get"去掉，小写化，作为JSON的每个key值，如getA对应的key值为a，而与真实的类成员名无关。 2、JSON to bean ：先同样通过反射找到对象类所有的Set方法，然后使用无参数构造函数（所以一定要有无参数的构造函数）新建一个类对象，从JSON字符串中取出一个key如a，先大写化为A，那么从所有Set方法中找到SetA()，然后进行赋值。 如果找不到setA（seta也不行），那么该值被忽略，也不报错。	是，最近一次提交是4天前	1.1.52.android	196KB	在1000倍数据量级内，效率最高，在10000倍数据量级及以上，效率和Jackson相差无几。	在1000倍数据量级内，内存占用低，峰值为2.69M，在10000倍数据量级及以上，内存占用率超过24M，并且存在明显的内存抖动。	高	简单	bean需要有默认的构造函数
Jackson	原理和FastJson一致，但是在 JSON to Java pojo的步骤中，做了更加科学的check，因而能识别seta这样的小写。但是如果getA 和geta都找不到，则会抛出异常（除非把a设置为忽略）。	是，最近一次提交是9天前	1.9.9	1100KB	在1000倍数据量级内，效率最低，在10000倍数据量级及以上，效率和FastJson相差无几。	在1000倍数据量级内，内存占用低，峰值为4.55M，在10000倍数据量级及以上，内存占用率超过35M，并且存在明显的内存抖动。	高	中等	

- 三种框架在实现上都使用了反射机制；
- 三种框架目前仍然在维护，所以从这一点来讲，使用任何一个框架是是靠谱的，因为有人在维护，反馈问题可以改，遇到问题也可以问；
- 三种框架FastJson的包最小（为了方便对比大小，均是对jar包做的统计），只有196K，其次是Gson的227K，Jackson会大很多，超过了1M；
- 三种框架在1000倍数据量级以内效率都很高、占用内存也低，效率最高的是FastJson，内存占用都差不多；1000倍数据量级以上内存占用会很大，并且平均耗时会超过1S，FastJson表现中规中矩；
- json解析、转换是一件很耗CPU的工作；
- 三种框架在bean转换为json时都有很好的易用性，但在json解析的实现上，FastJson使用起来最简单，其次是Gson;
- FastJson对bean有要求，必须要有默认的构造函数。

综合上述各方面的测试结果，并结合实际工作情况来看（如果数据量很大，可以考虑分页，多次获取），在项目中应该优先使用FastJson框架，它在一定数据量范围内，内存占用、效率等方面会表现得更为优秀；其次是Gson（结合实际工作经验，频繁的使用json解析和转换更应该使用FastJson）。

特别说明：

由于测试方案、测试环境以及具体用法并不一定权威，所以测试结果仅供参考，在实际开发过程中可以借鉴本文的测试结论，但建议在任何开源项目的选型前，还是要亲自动手，对性能、效率、易用性、功耗、大小、是否有人在维护、稳定性等各方面进行测试，根据综合分析后的结果选择最适合自己项目的框架。

参考资料

- 测试Demo
- gson
- FastJson Android版本

- Jackson
- FastJson 使用详解
- gson使用详解
- Google Play Top200 应用分析报告
- 国内Top500Android应用分析报告