# 测试场景

为了更好的说明JD\_Thrift跟原生Thrift的比较，我们按下面的场景进行性能测试：

* 服务器提供下面这个服务：

list<list<UserProfile>> retrieve(1: i32 uid, 2: required string test) throws(1: Xception err1)；

该方法返回10个list，每个list里包含10个UserProfile实例。

* client端调用远程retrieve方法，每10000次打印一条消耗时间；

# 测试结果(单位：毫秒)

* **场景1：原生Thrift服务 + 原生Client**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 第1个10000 | 第2个10000 | 第3个10000 | 第4个10000 | 第5个10000 | 第6个10000 | 第7个10000 | 第8个10000 | 第9个10000 | 第10个10000 | 平均值 |
| 6109 | 5985 | 5968 | 5922 | 6047 | 6093 | 5828 | 5984 | 5985 | 5750 | 5926.55 |
| 6031 | 5812 | 5938 | 5750 | 5656 | 5719 | 5735 | 6031 | 6141 | 6437 | 5939.05 |
| 6234 | 6625 | 6204 | 6562 | 6328 | 6438 | 6203 | 6329 | 5812 | 5875 | 6214.05 |

* **场景2：自动Thrift服务 + 原生Client**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 第1个10000 | 第2个10000 | 第3个10000 | 第4个10000 | 第5个10000 | 第6个10000 | 第7个10000 | 第8个10000 | 第9个10000 | 第10个10000 | 平均值 |
| 6297 | 6578 | 6422 | 6313 | 6110 | 6594 | 6546 | 6312 | 6547 | 6078 | 6375.0 |
| 6594 | 5921 | 5953 | 6547 | 6125 | 6000 | 6313 | 6469 | 6078 | 6328 | 6158.6 |
| 6281 | 6562 | 6031 | 6016 | 6000 | 6437 | 6531 | 6532 | 6484 | 6578 | 6335.15 |

* **场景3：原生Thrift服务 + 自动Client**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 第1个10000 | 第2个10000 | 第3个10000 | 第4个10000 | 第5个10000 | 第6个10000 | 第7个10000 | 第8个10000 | 第9个10000 | 第10个10000 | 平均值 |
| 7156 | 7094 | 7234 | 7219 | 7063 | 7015 | 7219 | 7031 | 6922 | 6938 | 7078.15 |
| 7110 | 6922 | 6953 | 7015 | 6829 | 7297 | 7172 | 6984 | 6860 | 6875 | 7003.15 |
| 7250 | 6922 | 6906 | 7031 | 7032 | 6938 | 7078 | 7094 | 7125 | 6906 | 7020.3 |

结论：

* 表1和表2的数据对比，作为**服务提供者**JD\_Thrift相对原生Thrift要损失**5%**的处理性能。
* 表1和表3的数据对比，作为**client端**JD\_Thrift相对原生Thrift要损失**17%**的处理性能。

这个性能损耗的原因是因为原生Thrift是直接调用生成的静态代码中的方法和属性，而JD\_Thrift是通过反射、hashmap/arrayList查找遍历等操作而导致性能的损失。