火龙果•整理
uml.org.cn

金融界
JRJ.COM

Mongodb 2.0 性能测试

王 乐 2012.3

火龙果•整理
uml.org.cn

提　纲

1. 单节点基本操作(CRUD)
2. Rest服务的压力测试
3. Replica Set性能及可靠性
4. Mongodb2.0 VS Mysql5.5

测试环境

- 服务器：
  - 硬件配置：Xeon E5310 1.60GHz、8GB RAM
  - 操作系统：Red Hat ES 5.5
  - 数据库：Mongodb 2.0.2、Mysql 5.5.19
- 客户机：
  - 位于相邻网段（千兆以太网 2跳 3ms）
  - 硬件配置基本同上
  - 安装测试脚本的运行环境及相关测试工具

测试用例

- 测试数据：
  - {
  - "_id" : ObjectId("4f4f42479792b60fe1e4bd09"),
  - "age" : 63（0~100的随机整数），
  - "test1" : "asdf123456撒旦法！#￥%QWE7890...长度256byte",
  - "test2" : "asdf123456撒旦法！#￥%QWE7890...",
  - "test3" : "asdf123456撒旦法！#￥%QWE7890...",
  - "test4" : "asdf123456撒旦法！#￥%QWE7890..."
  - }
  - 一个Document的大小约为1KB
- 测试步骤：
  - 使用以上测试数据分别进行各个场景下的操作，并记录测试结果

测试方法

- 基本操作：
  - 数据量采用10万、100万、1千万、1亿条四个数量级
  - 并发分别为单线程、10线程、50线程、100线程
  - 使用java、php、JavaScript等Driver编写测试脚本
- Rest服务：
  - 启用mongodb自带的rest服务
  - 第三方rest API（mongoose、nodejs rest等）
  - 使用ab、webbench、jmeter等工具进行压力测试
- Replica Set：
  - 以三台服务器为例组成Set，进行CRUD测试
  - 模拟节点故障，测试稳定性、可靠性
- Mysql Benchmark：
  - 在相同的服务器上搭建mongodb2.0和mysql5.5
  - 进行适当的优化使其发挥更好的性能
  - 将之前的case再跑一遍，分析对比测试结果

火龙果•整理
uml.org.cn

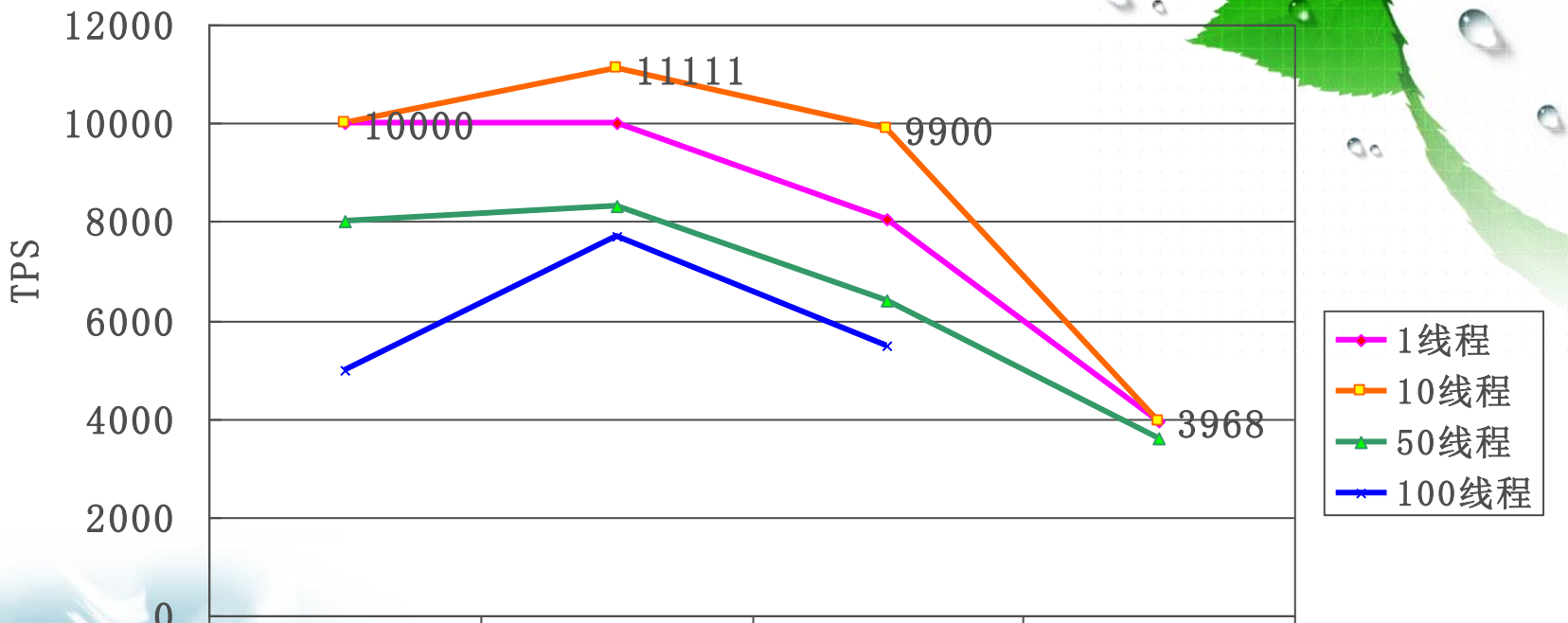
提　纲

1. 单节点基本操作(CRUD)
2. Rest服务的压力测试
3. Replica Set性能及可靠性
4. Mongodb2.0 VS Mysql5.5

基本操作（CRUD）

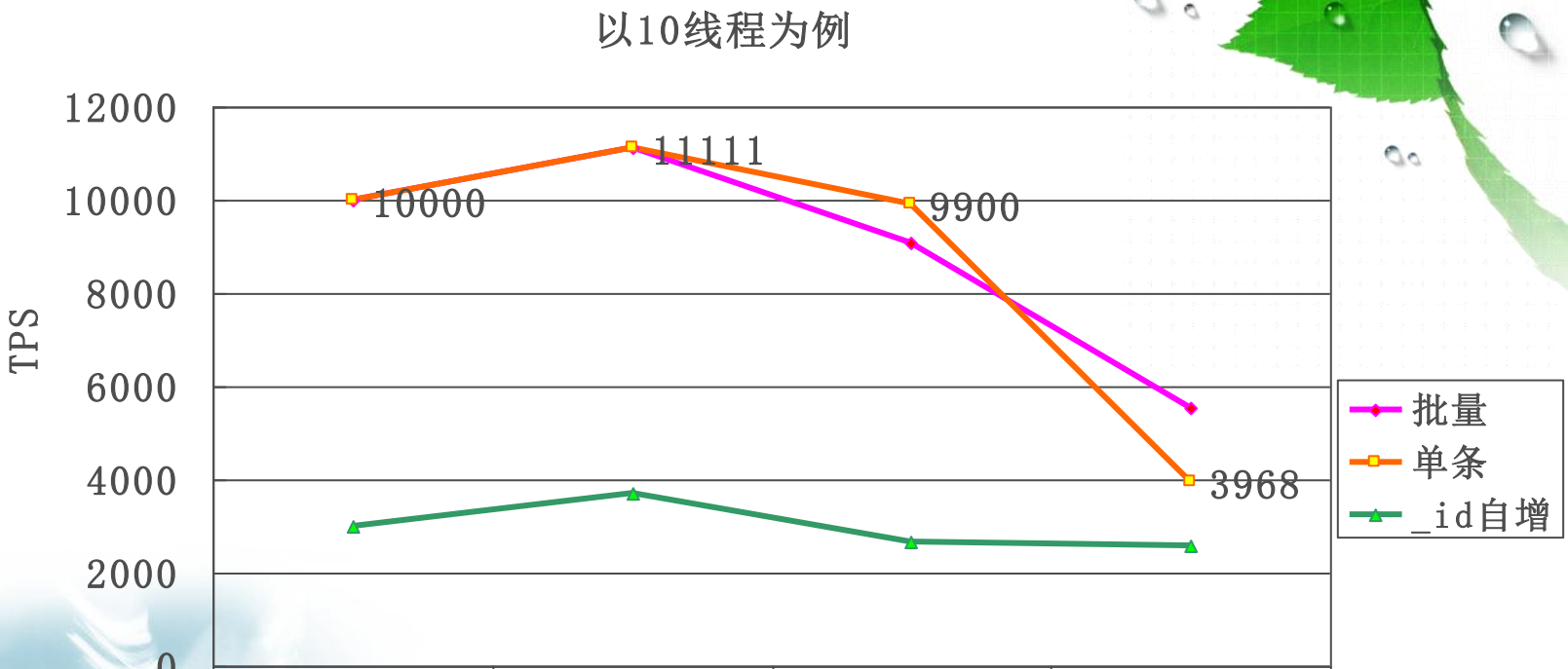
- 插入：
  - 针对不同数据量、不同并发进行单条插入
  - 针对不同数据量、不同并发进行批量插入
  - 默认主键生成方式与自增主键对比
- 更新：
  - 各种场景下的主键更新
  - 各种场景下的批量更新
- 查询：
  - 各种场景下的主键查询
  - 各种场景下的索引查询
  - 各种场景下的模糊查询
  - 各种场景下的查询并排序
- 删除：
  - 各种场景下的主键删除
  - 各种场景下的批量删除

数据插入（单条）



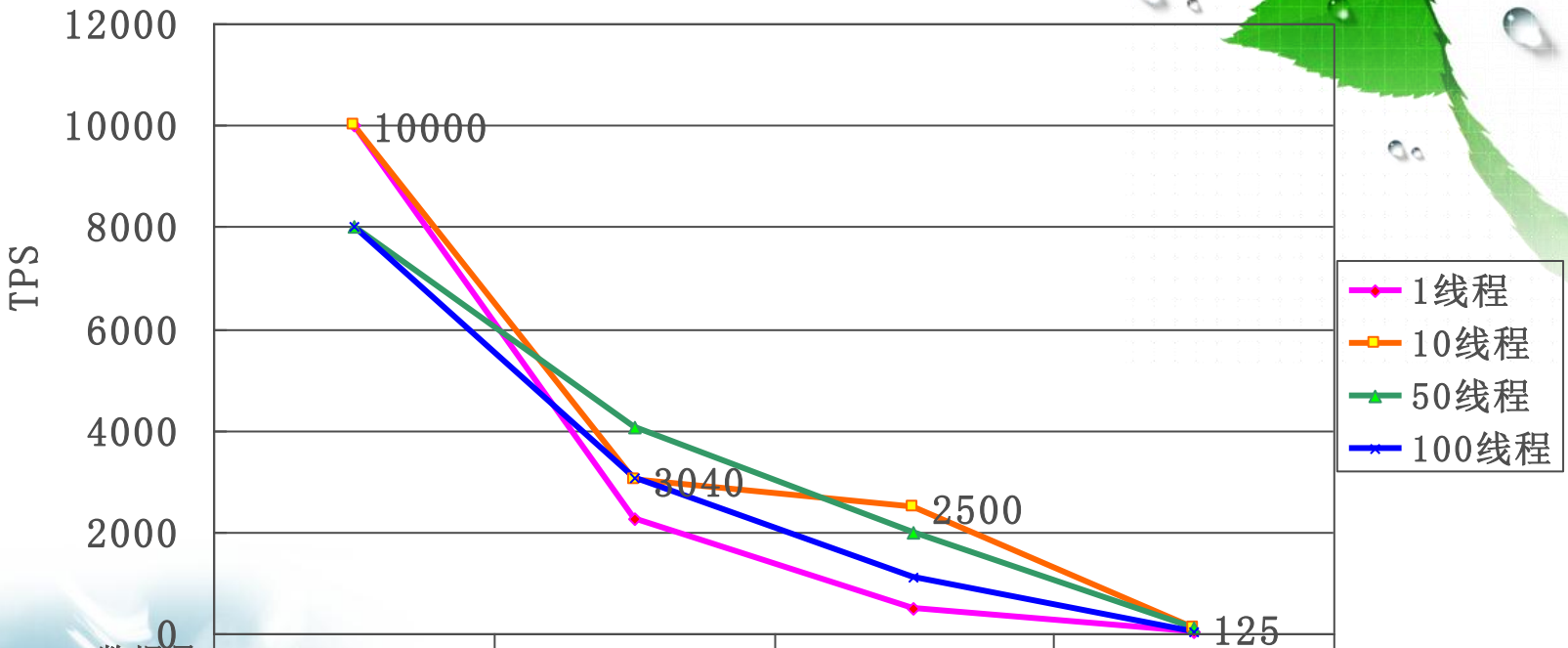
数据量	1万	10万	100万	1000万
1线程	10000	10000	8064	3968
10线程	10000	11111	9900	3968
50线程	8000	8333	6410	3606
100线程	5000	7692	5500	

数据插入（批量、自增）



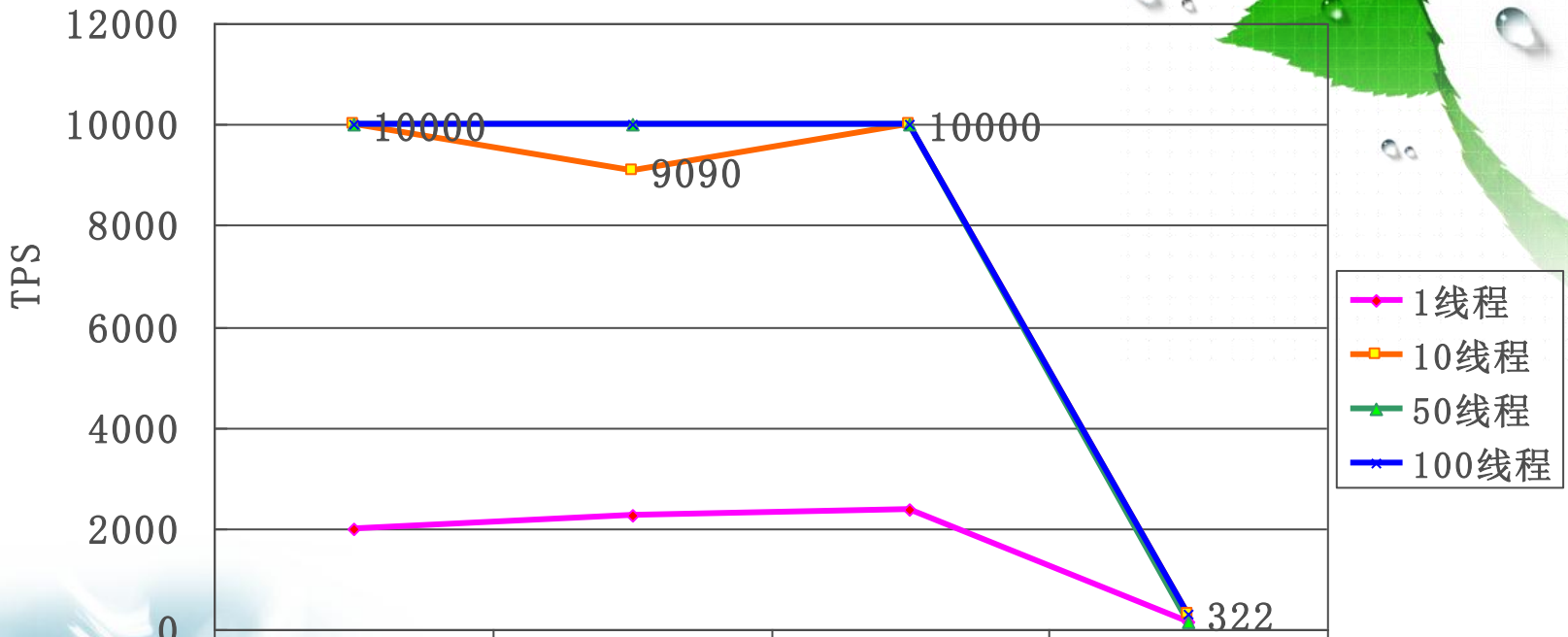
数据量	1万	10万	100万	1000万
批量	10000	11111	9090	5546
单条	10000	11111	9900	3968
_id自增	3000	3703	2659	2564

数据更新（主键）



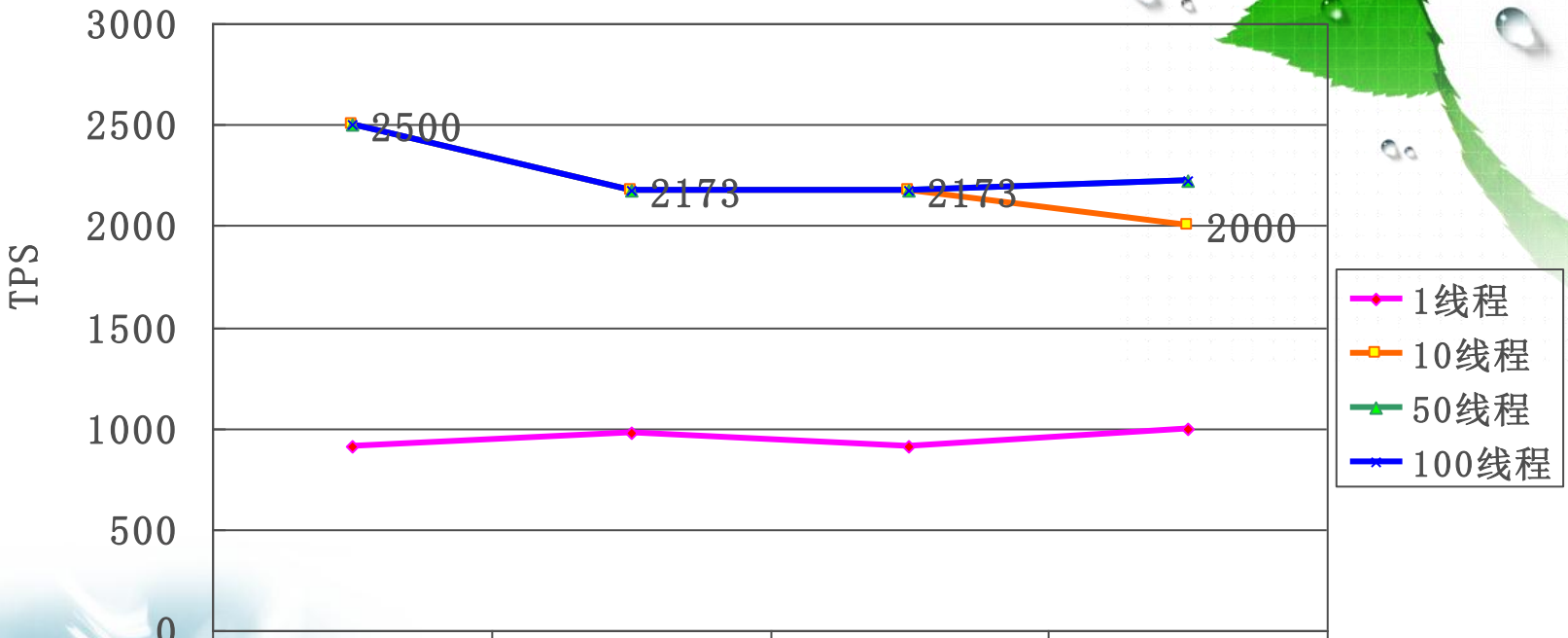
数据量	1万	10万	100万	1000万
1线程	10000	2272	500	57
10线程	10000	3040	2500	125
50线程	8000	4055	2000	104
100线程	8000	3060	1111	33

数据查询（主键）



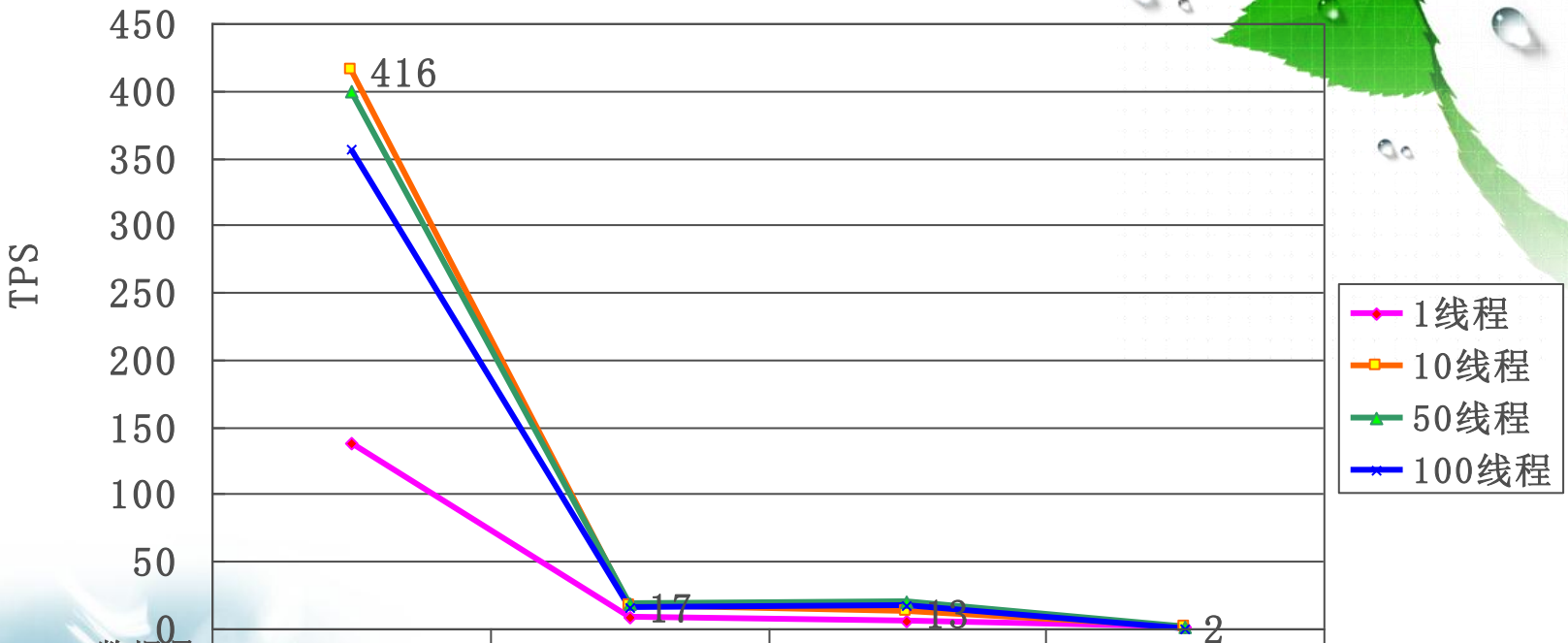
数据量	1万	10万	100万	1000万
1线程	2000	2272	2380	166
10线程	10000	9090	10000	322
50线程	10000	10000	10000	163
100线程	10000	10000	10000	312

数据查询（索引）



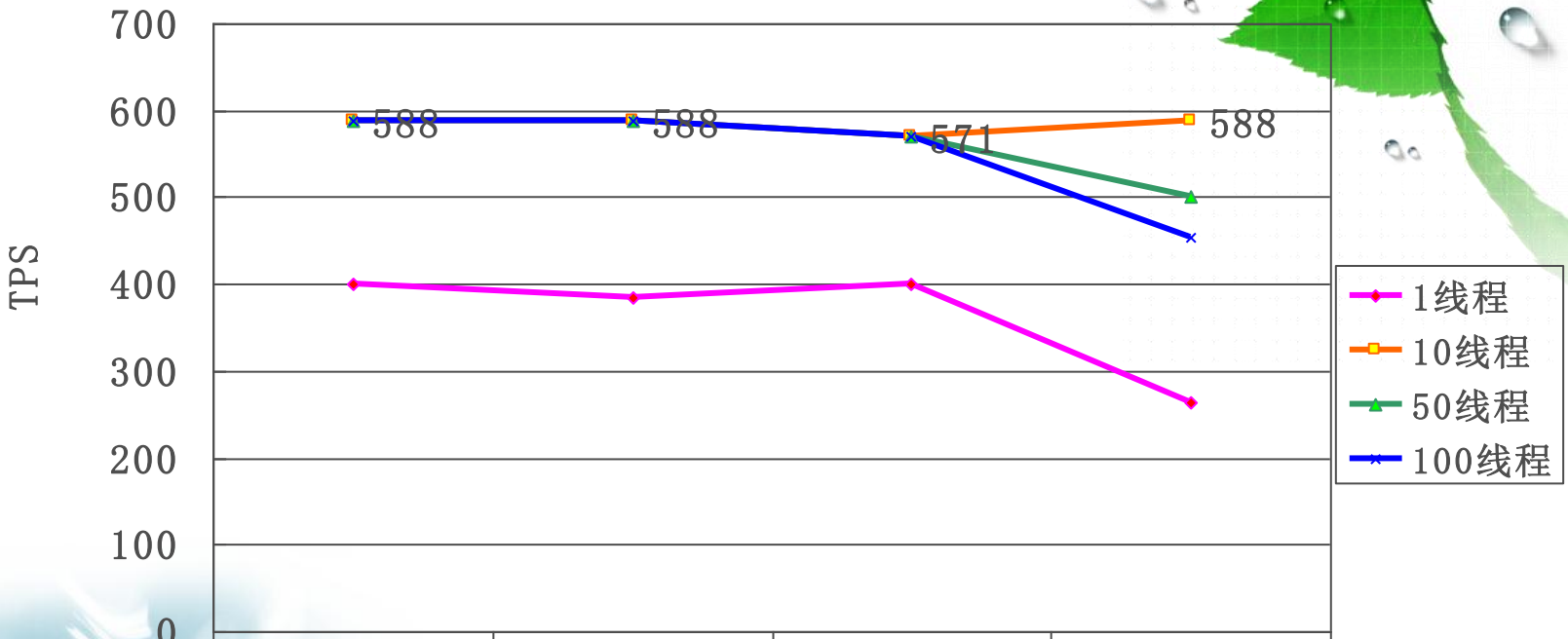
数据量	1万	10万	100万	1000万
1线程	909	980	909	1000
10线程	2500	2173	2173	2000
50线程	2500	2173	2173	2222
100线程	2500	2173	2173	2222

数据查询（模糊）



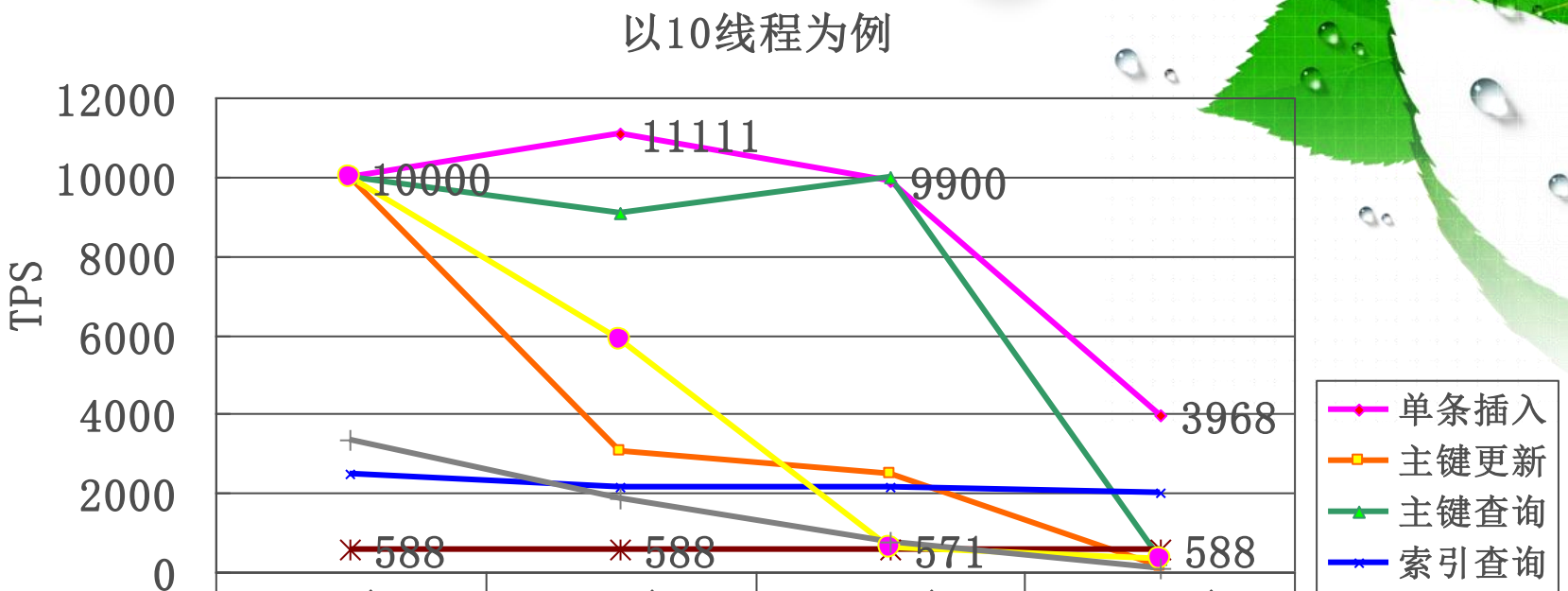
数据量	1万	10万	100万	1000万
1线程	138	9	6	1
10线程	416	17	13	2
50线程	400	19	20	1
100线程	357	16	17	0

数据查询（排序）



数据量	1万	10万	100万	1000万
1线程	400	384	400	263
10线程	588	588	571	588
50线程	588	588	571	500
100线程	588	588	571	454

CRUD对比



	1万	10万	100万	1000万
单条插入	10000	11111	9900	3968
主键更新	10000	3040	2500	125
主键查询	10000	9090	10000	322
索引查询	2500	2173	2173	2000
查询排序	588	588	571	588
主键删除	10000	5882	641	333
随机读写	3333	1886	781	83

资源占用情况

- CPU：
  - 随着数据量和并发数的提高，cpu的idle逐步减小
  - 多线程会充分利用多核的CPU
  - iowait间歇性较高，此时正在将数据同步至硬盘
- 内存：
  - 当数据文件小于内存时全部装载，大于内存时装载热数据及索引
  - 最好限制swap的使用，sysctl -w vm.swappiness=0
- 磁盘：
  - 硬盘空间占用约为实际数据集大小的两倍，索引约为数据集的10%
  - Journal与写入操作有关，约为数据集的20%
- 负载：
  - load值基本保持稳定，未出现超负荷情况
- 带宽：
  - 基本相当于接收请求和返回数据的字节数，额外开销不大
  - Replica Set模式下节点间带宽I/O情况见下文

性能优化

- 适当建立索引，查询时要使索引生效(explain)
- 尽量避免无索引的模糊查询
- 对于排序时用到多个field的情况，应建立组合索引，且先后顺序一致
- 如果存在组合索引，则无需建立第一个字段的单独索引，性能几乎一样
- 最佳并发与硬件配置有关，选择合适的连接池PoolSize可提高性能
- 索引field的CUD操作会触发索引更新，在数据量较大时会影响性能

注意事项

- update $set 有$set则更新属性，无$set则更新整个对象
- update操作的效率与命中数量有关，抖动比较明显
- 网上有说authenticate会导致性能低下，经测试如果使用连接池的方式，问题不大
- nojournal可以提高性能、同时降低可靠性
- 选择合适的WriteConcern级别（NORMAL、SAFE、FSYNC_SAFE、REPLICAS_SAFE等）
- 将Mongo和DB设置为static变量，以免产生大量connection
- 使用Replica Set以实现负载均衡和自动故障转移

火龙果•整理
uml.org.cn

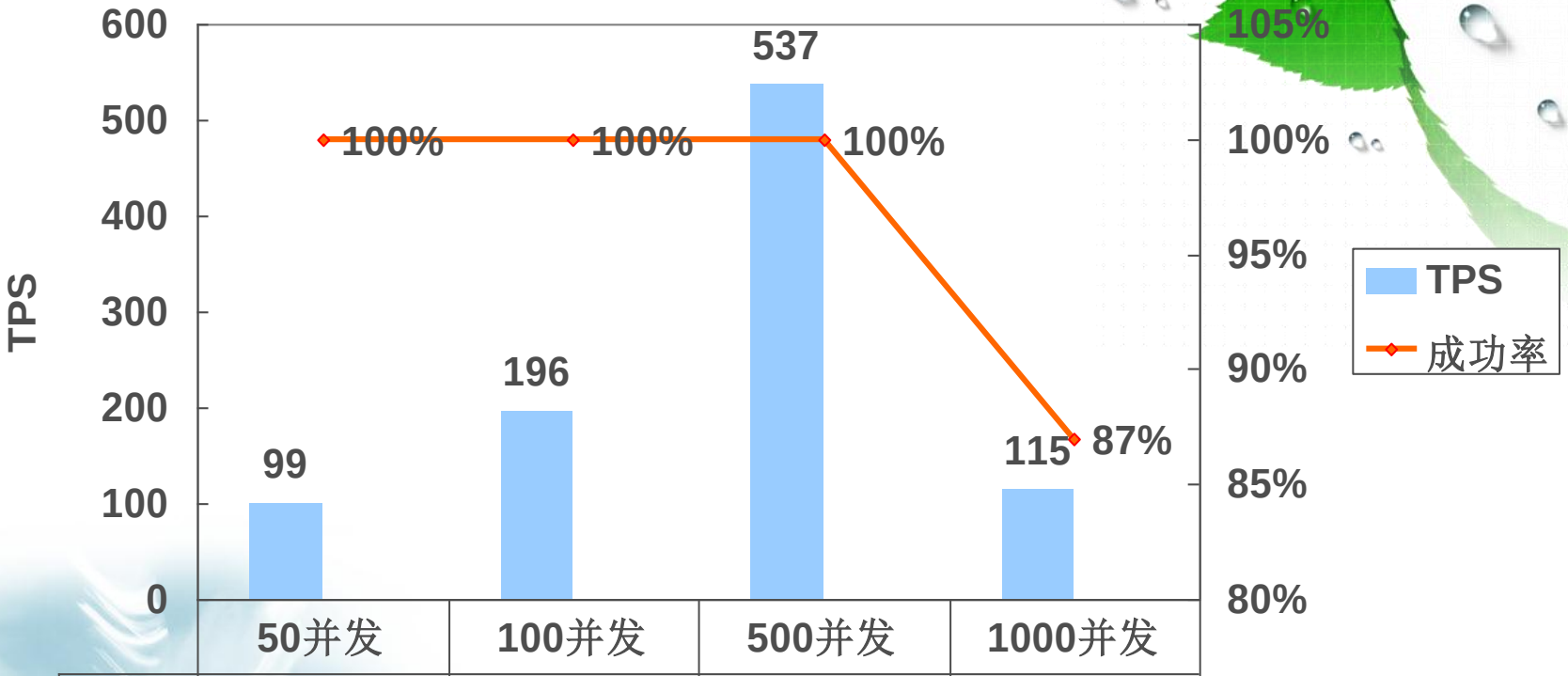
提　纲

1. 单节点基本操作(CRUD)
2. **Rest服务的压力测试**
3. Replica Set性能及可靠性
4. Mongodb2.0 VS Mysql5.5

测试方法

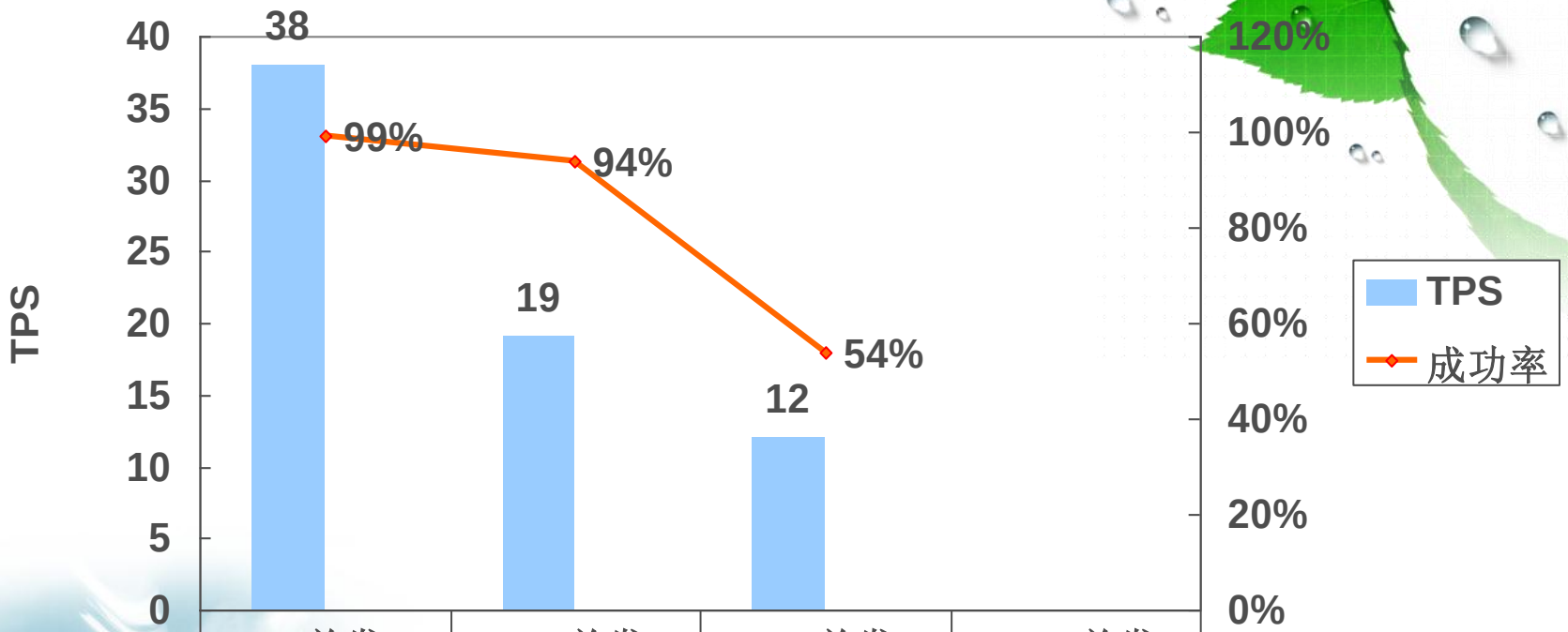
- Mongodb自带的simple Rest：
  - ./mongod --dbpath ../db --logpath ../log/mongo.log --rest &
  - http://host:28017/test/test/?filter_age=0~99&skip=10&limit=20
  - siege -c500 -t1m -furl.txt
  - 记录测试结果
- Python Rest（Sleepy Mongoose）：
  - 关闭自带rest，启动http
  - 测试方法同上
  - 插入及更新操作
- Java Rest（Tomcat+SpringData）：
  - 测试方法同上
- Javascript Rest（Node.js）：
  - 测试方法同上

mongodb内置的rest



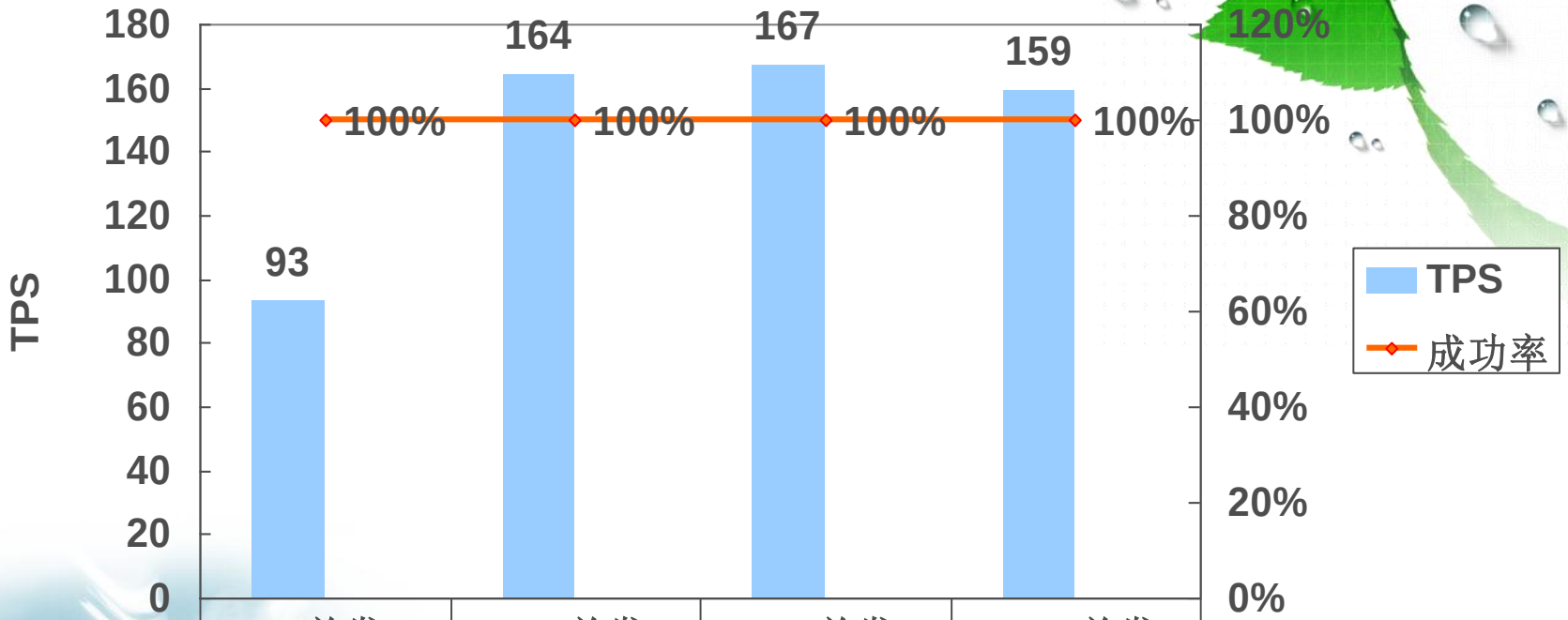
	50并发	100并发	500并发	1000并发
TPS	99	196	537	115
吞吐量	2MB/s	4MB/s	11MB/s	2MB/s
成功率	100%	100%	100%	87%

Sleepy Mongoose



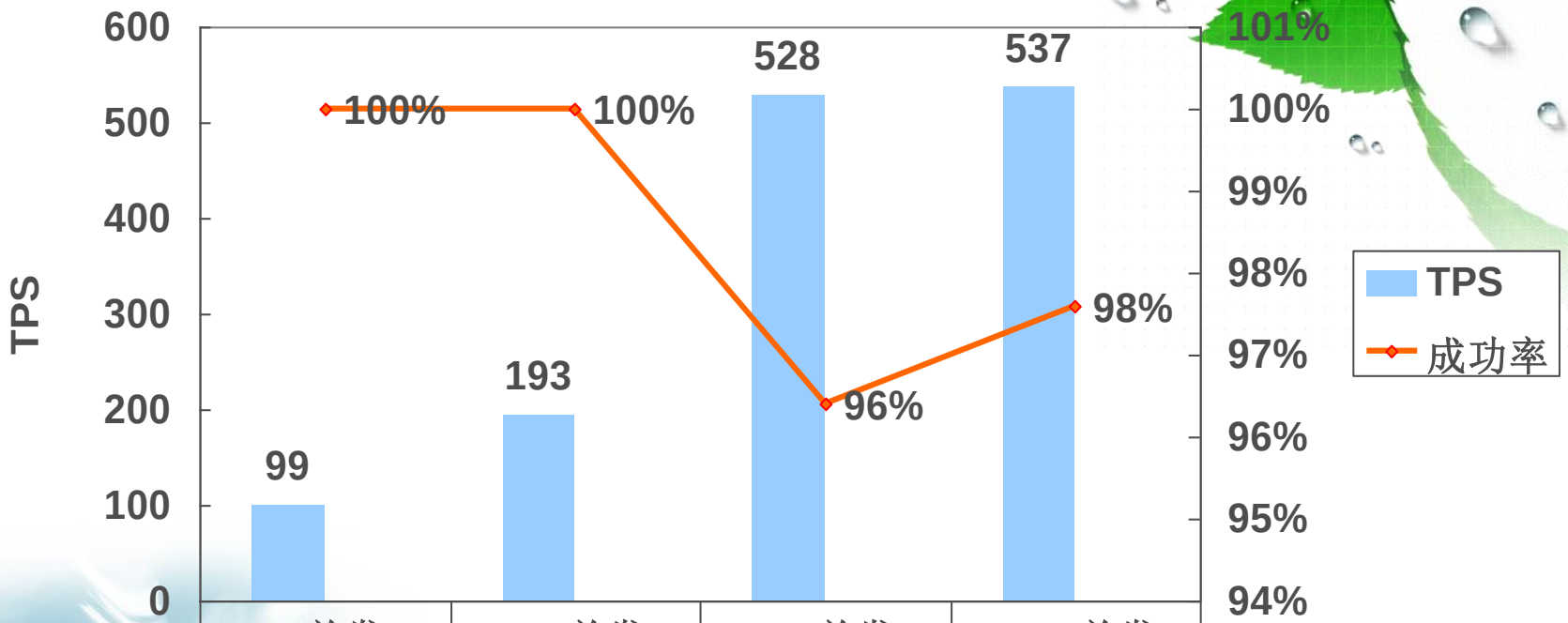
	50并发	100并发	500并发	1000并发
TPS	38	19	12	
吞吐量	0.7MB/s	0.35MB/s	0.22MB/s	
成功率	99%	94%	54%	

Node.js Rest



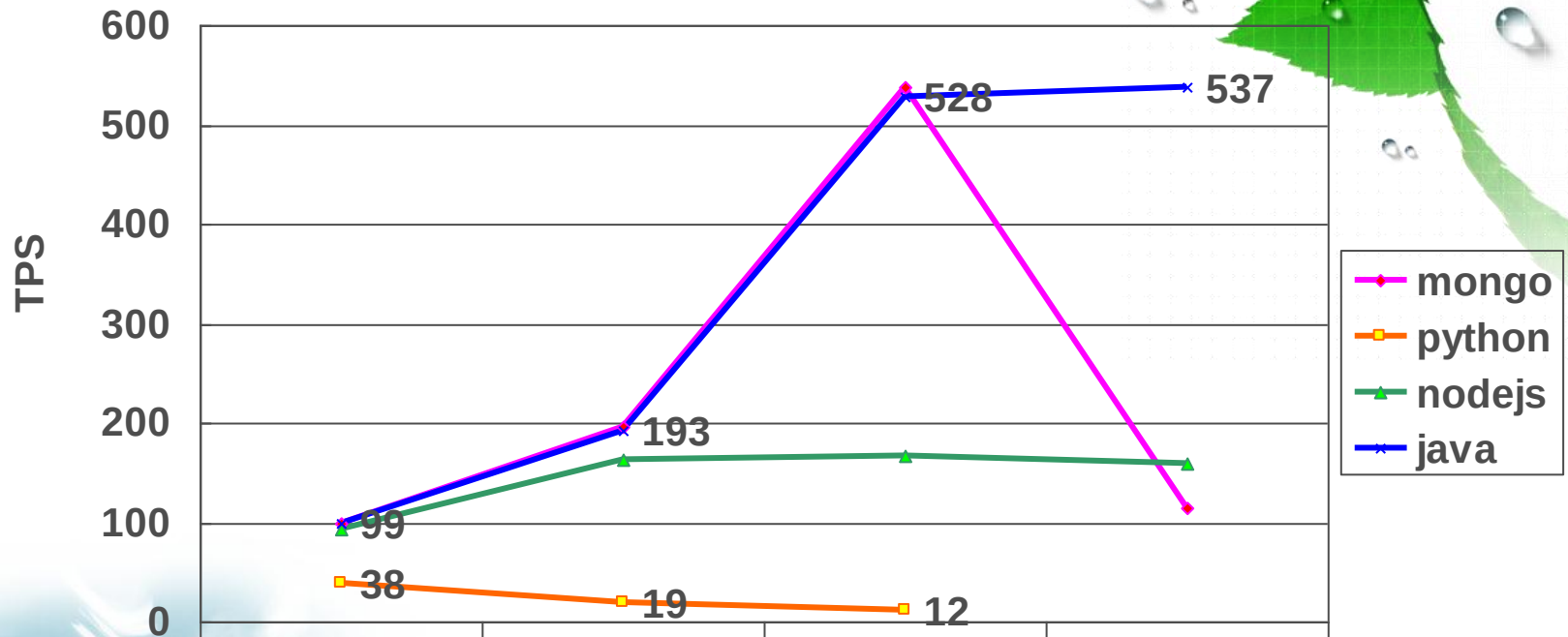
	50并发	100并发	500并发	1000并发
TPS	93	164	167	159
吞吐量	1.83MB/s	3.23MB/s	3.29MB/s	3.13MB/s
成功率	100%	100%	100%	100%

Java Rest



	50并发	100并发	500并发	1000并发
TPS	99	193	528	537
吞吐量	2MB/s	3.9MB/s	10.9MB/s	11MB/s
成功率	100%	100%	96%	98%

Rest性能对比



	50并发	100并发	500并发	1000并发
mongo	99	196	537	115
python	38	19	12	
nodejs	93	164	167	159
java	99	193	528	537

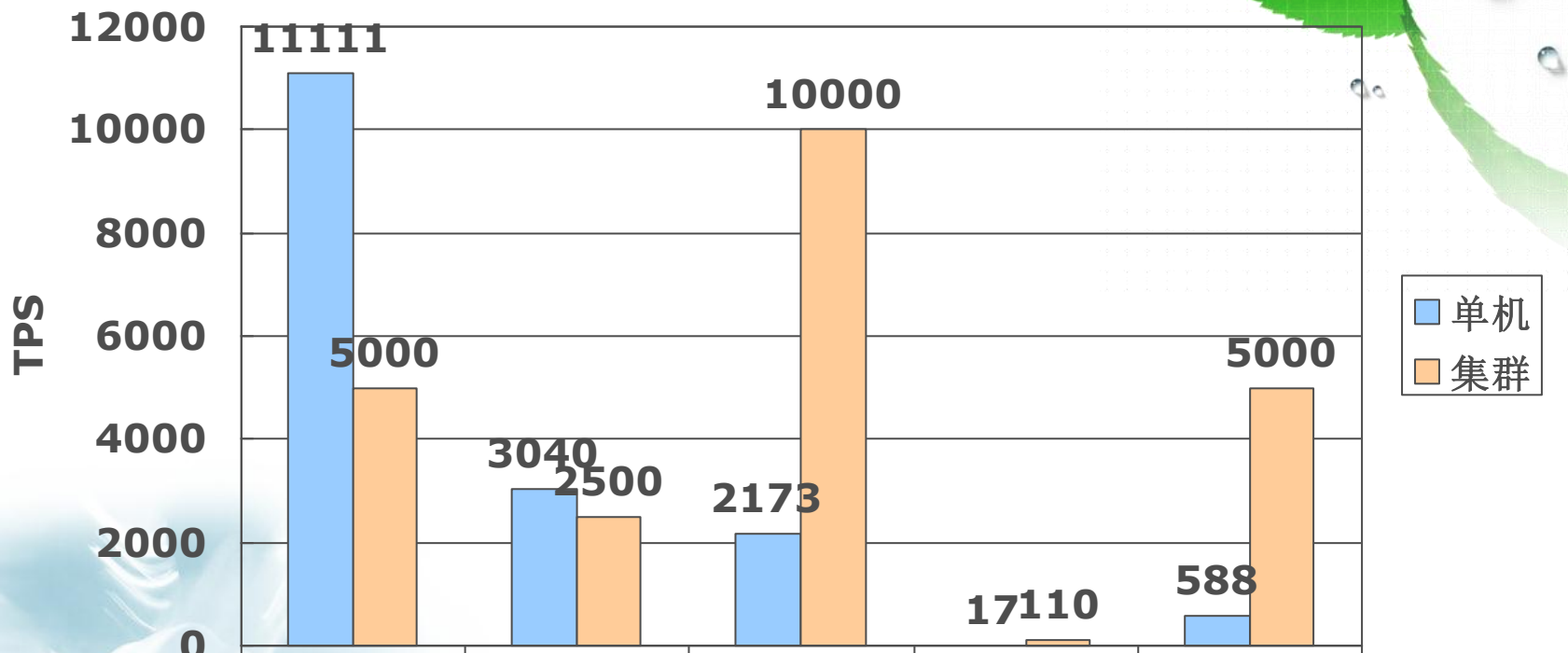
火龙果•整理
uml.org.cn

提　纲

1. 单节点基本操作(CRUD)
2. Rest服务的压力测试
3. **Replica Set性能及可靠性**
4. Mongodb2.0 VS Mysql5.5

Replica Set 性能

10线程10万数据、一主两从



	单条插入	主键更新	索引查询	模糊查询	索引排序
单机	11111	3040	2173	17	588
集群	5000	2500	10000	110	5000

备注：从机的硬件性能高于主机约30%

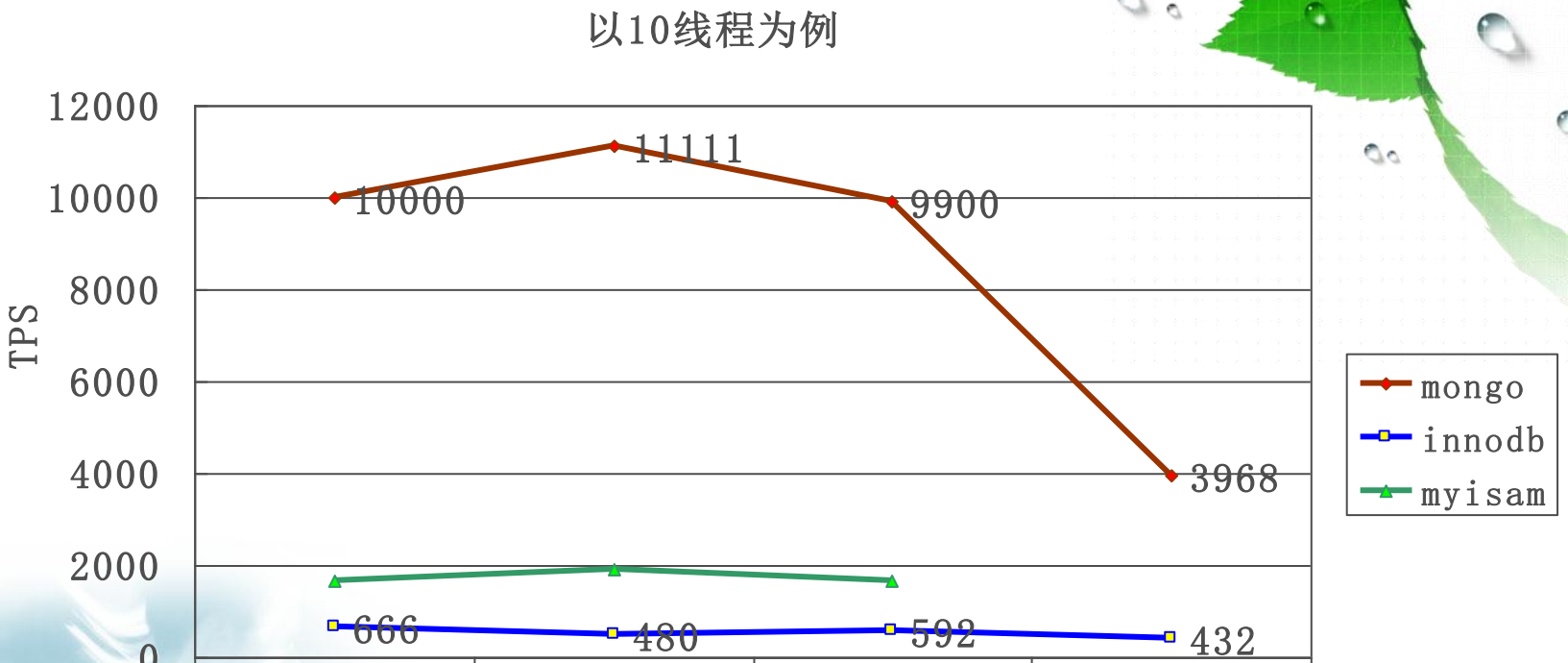
Replica Set 可靠性

- 测试方法：
  - 根据不同的WriteConcern级别循环插入10万条数据，期间依次kill掉一个secondary和primary
  - 查看自动failover的情况，统计成功、失败、及丢失的数量
  - 观察各服务器资源占用情况
- 测试结果：
  - WriteConcern.*NORMAL*：成功96873条、失败669条、丢失2458条、耗时9秒
  - WriteConcern.*SAFE*：成功99395条、失败604条、丢失1条、耗时25秒
  - WriteConcern.*REPLICAS_SAFE*：成功99402条、失败598条、丢失0条、耗时344秒
  - 可根据应用的具体场景权衡安全与效率
  - primary的CPU比secondary要高（与secondary的数量有关），secondary带宽in的字节数基本相当于oplog的增量，即未压缩的document，100M数据写入时secondary大概有20秒左右的同步滞后

提　纲

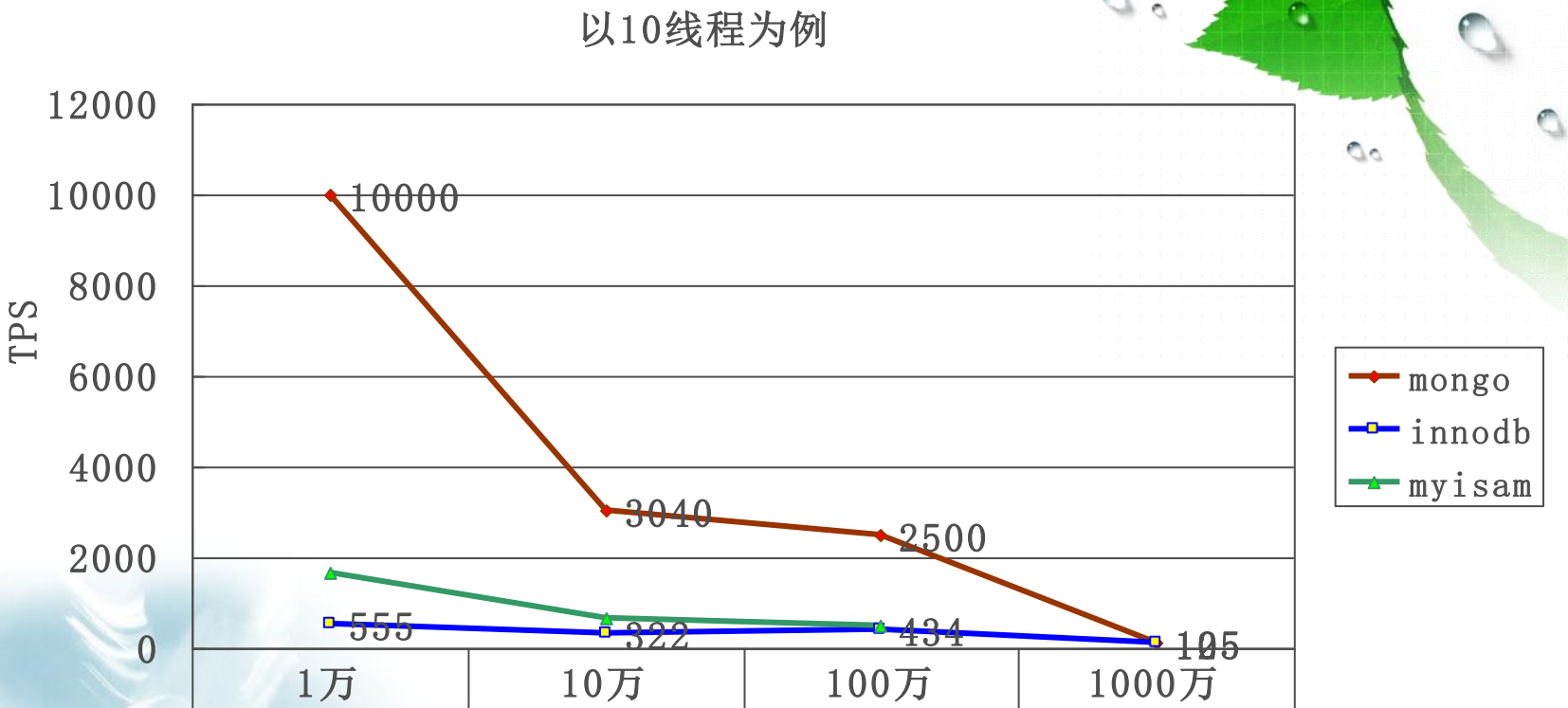
1. 单节点基本操作(CRUD)
2. Rest服务的压力测试
3. Replica Set性能及可靠性
4. **Mongodb2.0 VS Mysql5.5**

数据插入（单条）



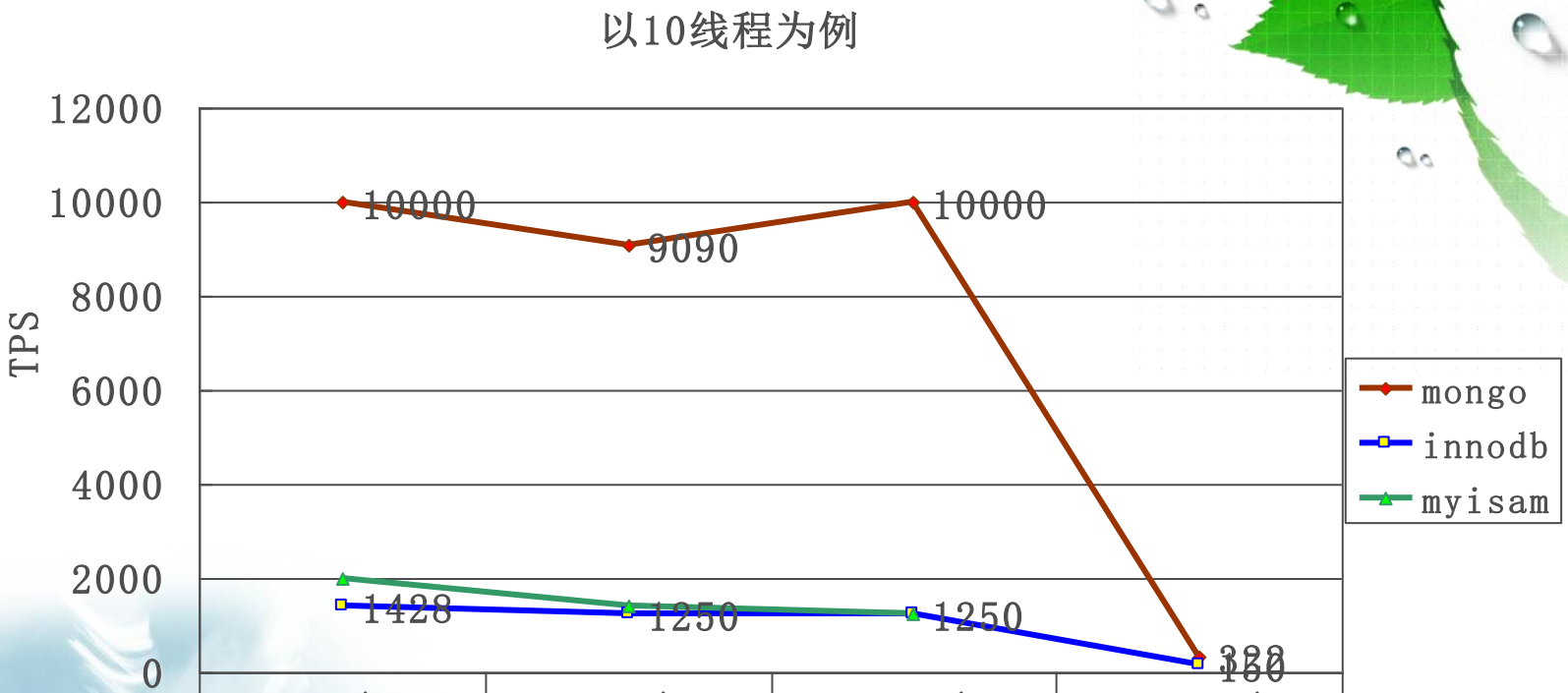
	1万	10万	100万	1000万
mongo	10000	11111	9900	3968
innodb	666	480	592	432
myisam	1666	1923	1651	

数据更新（主键）



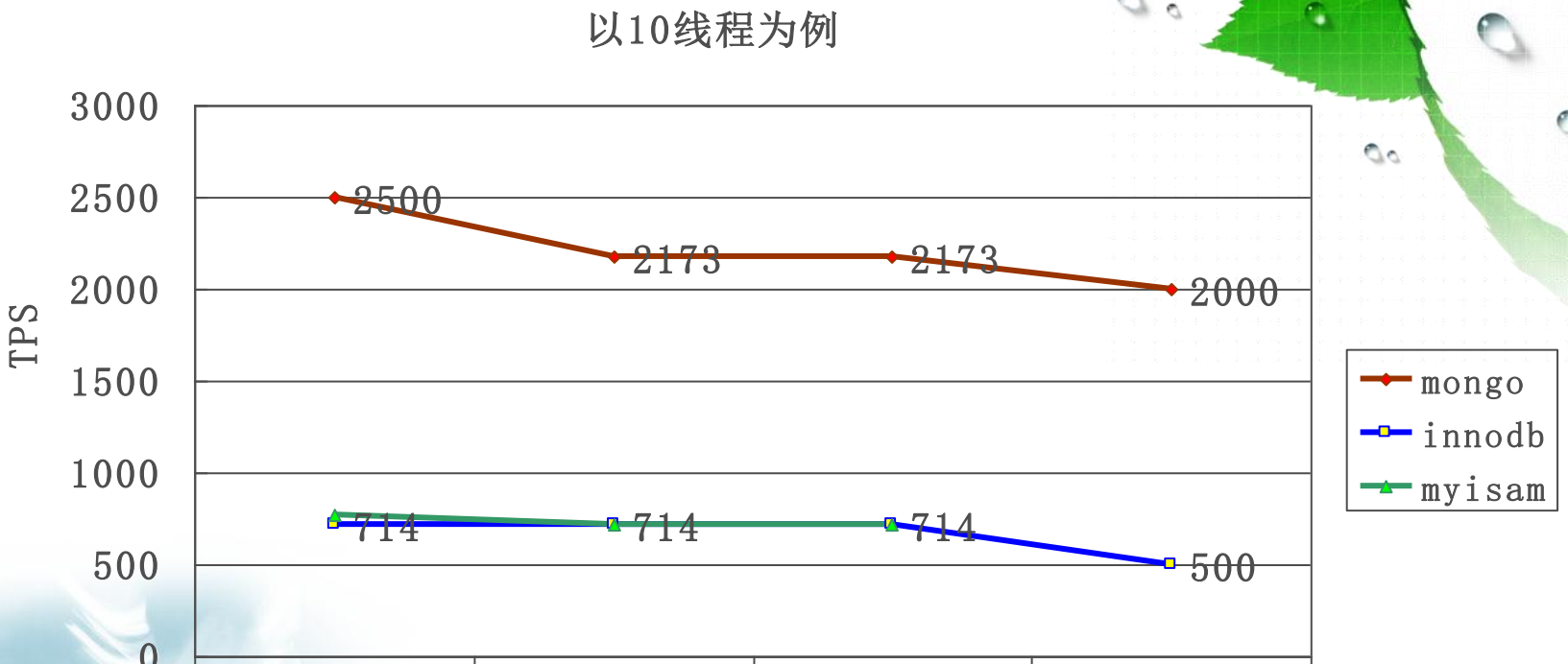
	1万	10万	100万	1000万
mongo	10000	3040	2500	125
innodb	555	322	434	105
myisam	1666	666	500	

数据查询（主键）



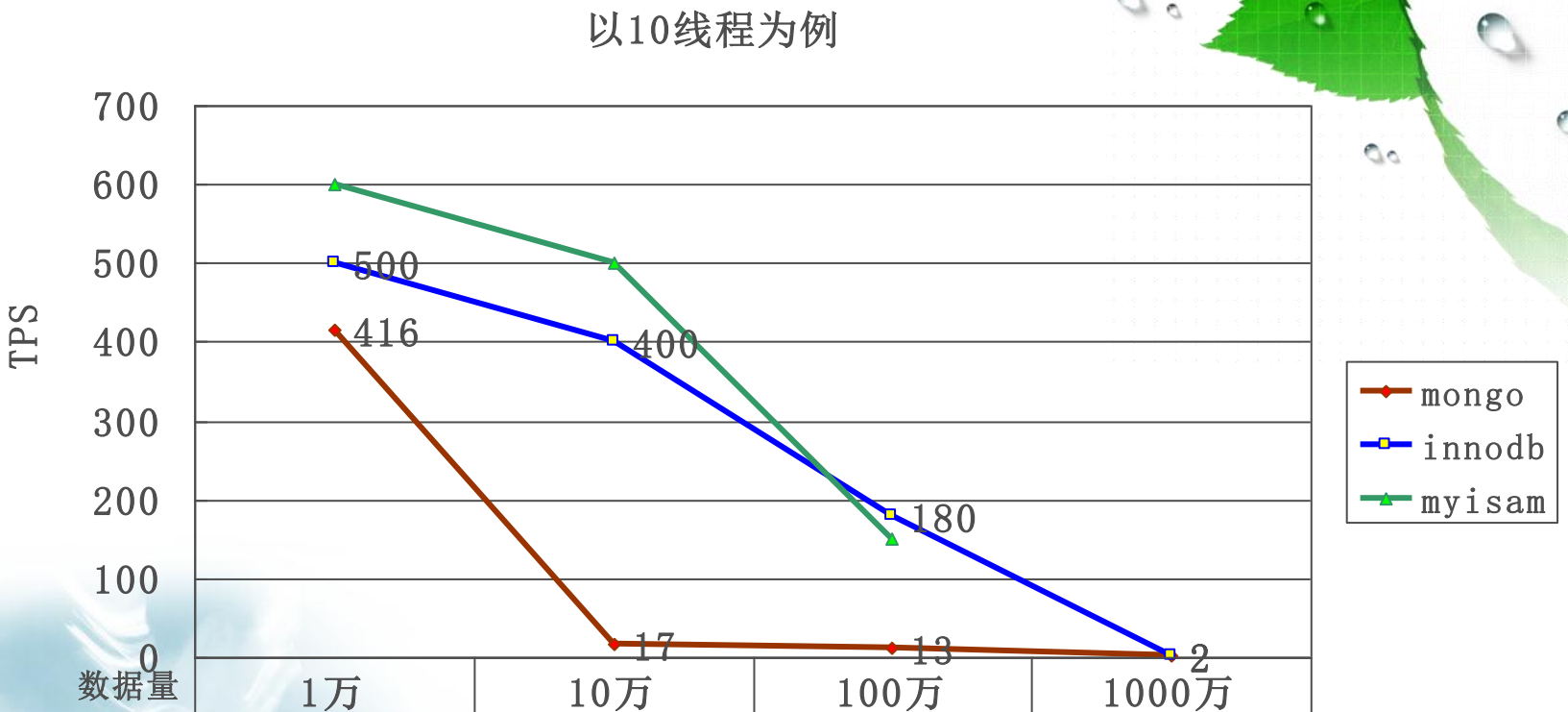
	1万	10万	100万	1000万
mongo	10000	9090	10000	322
innodb	1428	1250	1250	150
myisam	2000	1428	1250	

数据查询（索引）



	1万	10万	100万	1000万
mongo	2500	2173	2173	2000
innodb	714	714	714	500
myisam	769	714	714	

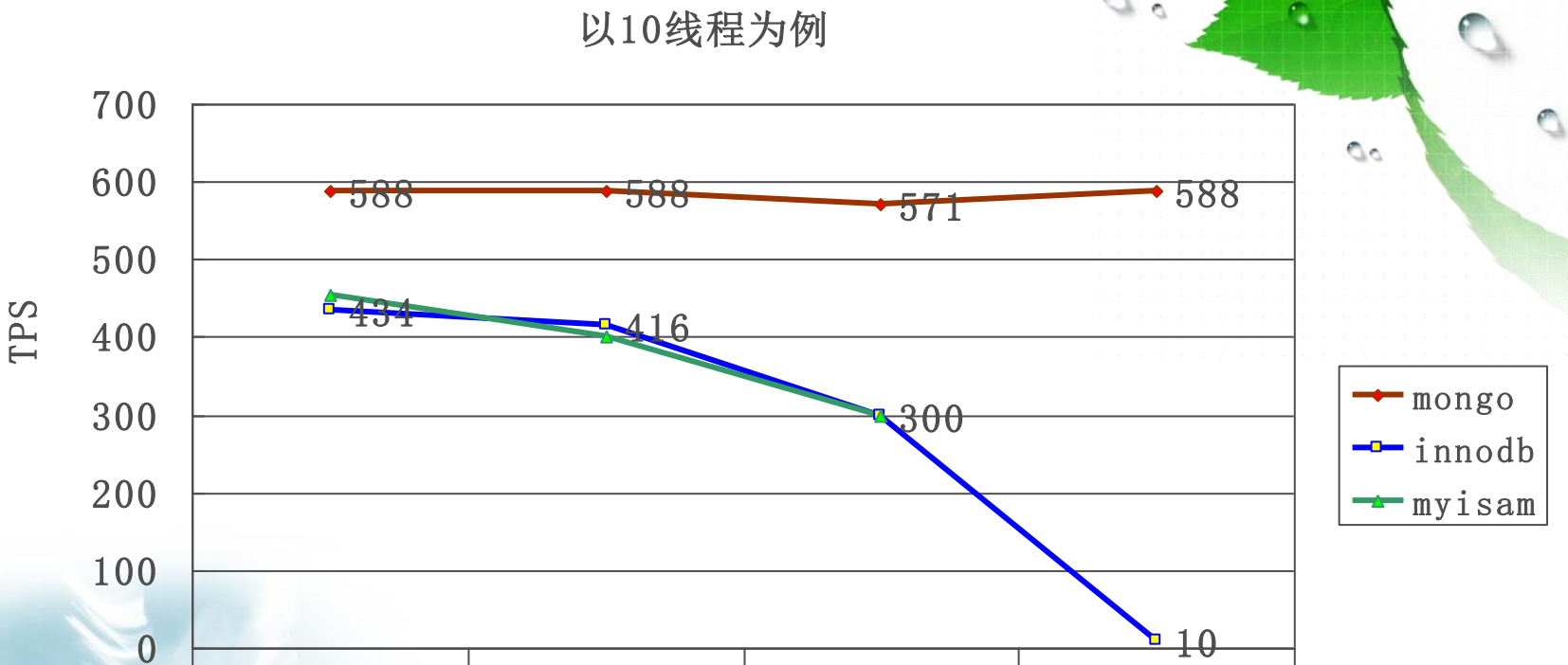
数据查询（模糊）



数据量	1万	10万	100万	1000万
mongo	416	17	13	2
innodb	500	400	180	2
myisam	600	500	150	

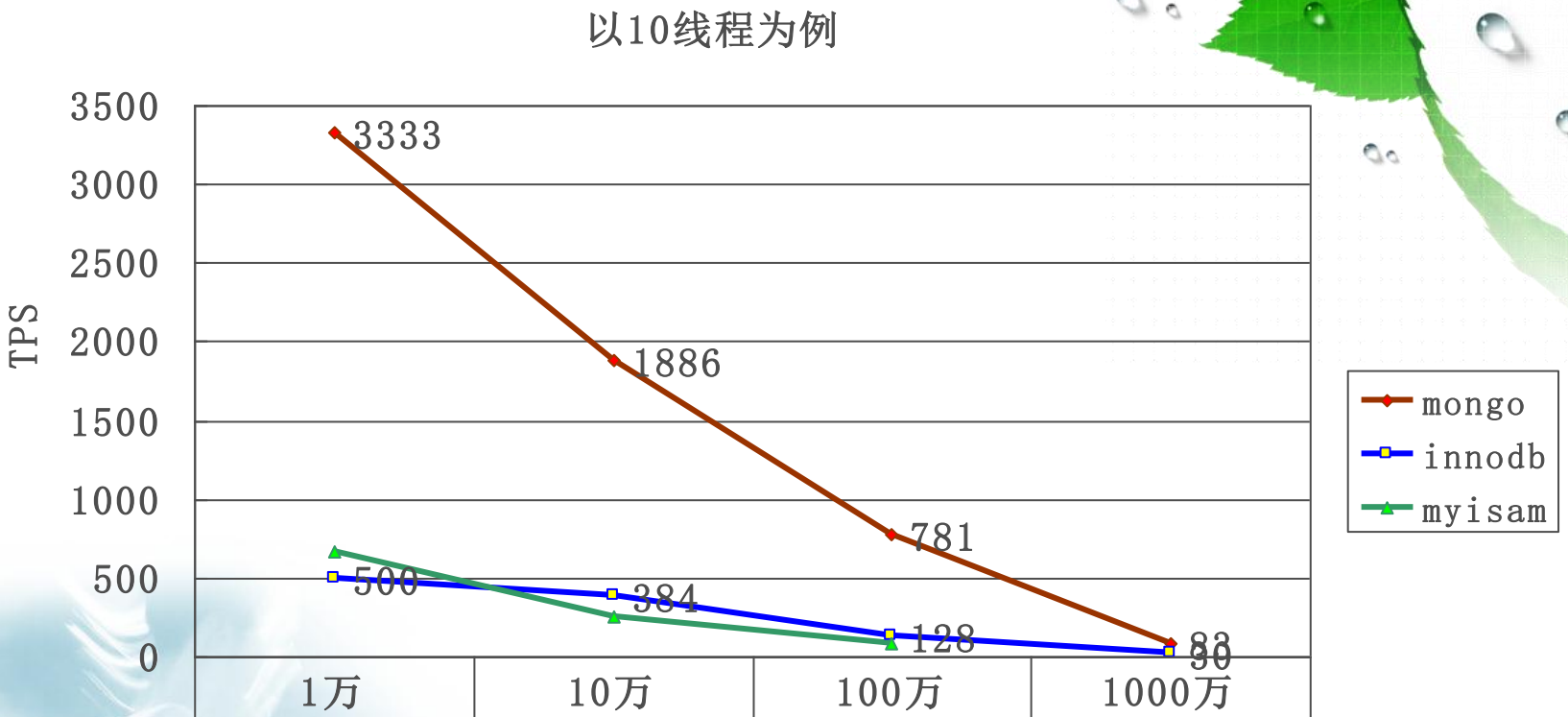
火龙果•整理
uml.org.cn

数据查询（排序）



	1万	10万	100万	1000万
mongo	588	588	571	588
innodb	434	416	300	10
myisam	454	400	300	

随机读写



	1万	10万	100万	1000万
mongo	3333	1886	781	83
innodb	500	384	128	30
myisam	666	250	80	

小结

- mongodb的数据插入和索引查询明显优于mysql，模糊查询明显逊于mysql
- 以最接近实际应用场景的查询排序和随机读写来看，mongodb的性能高于mysql
- 当数据量小于内存容量时，mongodb发挥的很出色，随着数据量的增长，磁盘IO将成为系统的最终瓶颈
- 目前mongodb自带的rest不支持post，Replica Set模式下不支持secondary的rest服务，如果Mongodb在将来的版本完善其rest API，将会有非常广泛的应用前景
- 从并发的角度来看，应当在Web层或App层设置缓存，否则容易导致DB层的Connection reset by peer
- 本次评测重点在于性能，在进行技术选型时应综合考虑系统的功能、性能、稳定性、可靠性、可维护性、安全性、开发效率、运营成本等方面

未测试到的场景

- C# php ruby 等 Driver
- 基于Sharding的benchmark
- 产品纵向对比（1.8 vs 2.0）
- Mongodb Sharding VS Mysql Cluster

参考资料

- http://hi.baidu.com/farmerluo/blog/item/0c47ad6e5c9318d681cb4a91.html
- http://www.cnblogs.com/lovecindywang/archive/2011/03/02/1969324.html
- http://blog.csdn.net/crazyjixiang/article/details/6630624
- http://www.michaelckennedy.net/blog/2010/04/29/MongoDBVsSQLServer2008PerformanceShowdown.aspx
- http://blog.nosqlfan.com/html/3304.html
- http://blog.nosqlfan.com/html/1329.html
- http://rdc.taobao.com/blog/cs/?p=198
- http://www.tbdata.org/archives/1285
- http://mysql.com/why-mysql/white-papers/mysql-cluster-7.2-ga.html

火龙果•整理
uml.org.cn

金融界
JRJ.COM

Thank You !