0、JStorm和Storm的区别

由于storm的内核是clojure编写的，目前阿里巴巴公司已经有开源的Copy版本Jstorn，以下本ID为你带来其中相关区别。

关于流处理框架，在先前的文章汇总已经介绍过[Strom](http://www.biaodianfu.com/storm.html)， 今天学习的是来自阿里的的流处理框架JStorm。简单的概述Storm就是：JStorm 比Storm更稳定，更强大，更快，Storm上跑的程序，一行代码不变可以运行在JStorm上。直白的将JStorm是阿里巴巴的团队基于Storm 的二次开发产物，相当于他们的Tengine是基于Ngix开发的一样。以下为阿里巴巴团队放弃直接使用Storm选择自行开发JStorm的原因：



阿里拥有自己的实时计算引擎

1. 类似于hadoop 中的MR
2. 开源storm响应太慢
3. 开源社区的速度完全跟不上Ali的需求
4. 降低未来运维成本
5. 提供更多技术支持，加快内部业务响应速度

现有Storm无法满足一些需求

1. 现有storm调度太简单粗暴，无法定制化
2. Storm 任务分配不平衡
3. RPC OOM一直没有解决
4. 监控太简单
5. 对ZK 访问频繁

**JStorm相比Storm更稳定**

1. Nimbus 实现HA：当一台nimbus挂了，自动热切到备份nimbus
2. 原生Storm RPC：Zeromq 使用堆外内存，导致OS 内存不够，Netty 导致OOM；JStorm底层RPC 采用netty + disruptor保证发送速度和接受速度是匹配的
3. 新上线的任务不会冲击老的任务：新调度从cpu，memory，disk，net 四个角度对任务进行分配，已经分配好的新任务，无需去抢占老任务的cpu，memory，disk和net
4. Supervisor主线
5. Spout/Bolt 的open/prepar
6. 所有IO, 序列化，反序列化
7. 减少对ZK的访问量：去掉大量无用的watch；task的心跳时间延长一倍；Task心跳检测无需全ZK扫描。

**JStorm相比Storm调度更强大**

1. 彻底解决了storm 任务分配不均衡问题
2. 从4个维度进行任务分配：CPU、Memory、Disk、Net
3. 默认一个task，一个cpu slot。当task消耗更多的cpu时，可以申请更多cpu slot
4. 默认一个task，一个memory slot。当task需要更多内存时，可以申请更多内存slot
5. 默认task，不申请disk slot。当task 磁盘IO较重时，可以申请disk slot
6. 可以强制某个component的task 运行在不同的节点上
7. 可以强制topology运行在单独一个节点上
8. 可以自定义任务分配，提前预约任务分配到哪台机器上，哪个端口，多少个cpu slot，多少内存，是否申请磁盘
9. 可以预约上一次成功运行时的任务分配，上次task分配了什么资源，这次还是使用这些资源

**JStorm相比Storm性能更好**

JStorm 0.9.0 性能非常的好，使用netty时单worker 发送最大速度为11万QPS，使用zeromq时，最大速度为12万QPS。

* JStorm 0.9.0 在使用Netty的情况下，比Storm 0.9.0 使用netty情况下，快10%， 并且JStorm netty是稳定的而Storm 的Netty是不稳定的
* 在使用ZeroMQ的情况下， JStorm 0.9.0 比Storm 0.9.0 快30%

性能提升的原因：

1. Zeromq 减少一次内存拷贝
2. 增加反序列化线程
3. 重写采样代码，大幅减少采样影响
4. 优化ack代码
5. 优化缓冲map性能
6. Java 比clojure更底层

**JStorm的其他优化点**

1. 资源隔离。不同部门，使用不同的组名，每个组有自己的Quato；不同组的资源隔离；采用cgroups 硬隔离
2. Classloader。解决应用的类和Jstorm的类发生冲突，应用的类在自己的类空间中
3. Task 内部异步化。Worker 内部全流水线模式，Spout nextTuple和ack/fail运行在不同线程