**Jstorm0.9.4.1 Performance**

# 1 测试环境

jstorm-0.9.4.1 使用 netty rpc框架

Storm-0.9.2-netty-p297(current release, STORM-297)

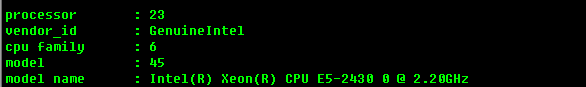
测试拓扑实例：[Jstorm\_test](https://github.com/hustfxj/jstorm_test)

[Storm\_test](https://github.com/hustfxj/storm_test/tree/master)

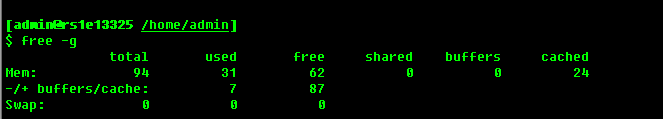
os: redhat 5



cpu: 24核



内存: 96G



# 2 测试进程并发度情况

测试说明：测试在保持executor(thread)并发度不变的情况下，改变worker(process)的并发度对性能的影响。

## 2.1 配置参数

Spout=18 bolt=18 acker=18 messageSize=10byte

Max.spout.pending =10000 level=1

**备注：**

Level代表拓扑层数

Polltuples表示每10s，spout发射的tuple数量

## 2.2 测试结果

图1 发射速度与worker并发度的关系

备注：横坐标表示workers并发数，纵轴表示拓扑spout每10s发射的平均tuples

图2 cpu利用率与worker并发度的关系

## 2.3 结果分析

* Jstorm-0.9.4.1经过优化后，传输性能显著高于storm（p297）；在参数配置适当情况下，jstorm的性能最高达到storm的两倍；
* Jstorm在各个worker并发度情况下，cpu利用率都比storm（p297）低；
* Jstorm和storm在并发度w=12,吞吐率达到峰值，再增加worker数，吞吐率下降，cpu利用率下降不明显，这是因为：

一方面，每新增加一个worker进程，都会将一些原本线程间的内存通信变为进程间的网络通信，这些进程间的网络通信还需要进行序列化与反序列化操作，这些降低了吞吐率；

另一方面，每新增加一个worker进程，都会额外地增加至少多个线程，这些线程切换消耗了不少CPU，sys cpu消耗占比增加，在CPU总利率率受限的情况下，use cpu消耗降低，这也影响了吞吐率的提升；

同时task保持不变时，当woker并发度减少，由于在计算过程中不同task之间可能需要切换，worker比较少的情况下，加大了进程切换的频繁度，这也一定消耗时间，降低了吞吐率。

# 3 测试消息大小情况

测试说明：测试不同消息大小对性能的影响。

## 3.1 配置参数

Spout=18 bolt=18 acker=18 workers=12

Max.spout.pending =10000 level=1

## 3.2 测试结果

图3 吞吐率与消息大小的关系

图5 cpu利用率与消息大小的关系

## 3.3 结果分析

* Storm（p297）和jstorm随着message-size增大，吞吐率都会上升，但是增长的幅度逐渐变缓；但总体传输速度都在增加，说明网络带宽利用均没有达到极限；
* 一般情况下jstorm在吞吐率和cpu利用率方面都表现比storm上，说明jstorm在cpu和带宽都允许的前提上，还有性能优化的潜力；

# 4 测试拓扑层数情况

测试说明：测试拓扑层数对性能的影响。

## 4.1 配置参数

Spout=16 bolt=6 acker=6 workers=12

Max.spout.pending =10000 message-size=10byte

## 4.2 测试结果

图5 发射速度与拓扑层数的关系

图6 cpu利用率与拓扑层数的关系

## 4.3 结果分析

* 随着拓扑层数的增加，系统吞吐率下降，同时CPU利用率一直在上升，网络带宽未达到极限；
* 拓扑的层数与上层业务的复杂度直接相关。业务越复杂，则拓扑层数越多，相应地吞吐率越低。（一般bolt处理速度比较慢，bolt层数越多会一定程度抑制spout的发射速率）

# 5 注意事项

* 不管是利用storm和jstorm，一般情况下都要设置max.spout.pending，否则拓扑中spout发射过快，jstorm和storm都会出现数据波动或者延迟时间延长的现象；
* **以上测试未反映**：storm和jstorm中当每一个worker并发的task数量过大，都因为线程的频繁切换，造成性能的下降。只是jstorm中worker中线程数量往往大于storm线程数，所以**jstorm中worker最好保证的task数量少于12个，而storm中worker的task数量可以保证在18个以下**；一旦jstorm和storm超过这个合理值范围时，性能都会下降，延迟时间加长，failed现象出现；

# 6 总结

* Jstorm传输性能显著storm;
* 进程并发数与task数量的比例在适当时，能发挥出最大吞吐率（权衡的结果）；
* 在cpu和带宽都未到达极限时，最大的吞吐率是随着消息大小增大而增大的；
* 业务使用的拓扑层次越小时，越能提高吞吐率；
* 配置拓扑合理的参数，可以尽可能增大storm和jstorm的传输性能；但若拓扑中task很多时，为了防止**注意事项2**的出现，应该加大一定数量的worker数。