Spark mllib kmeans 算法结果分析

一.算法可扩展性:

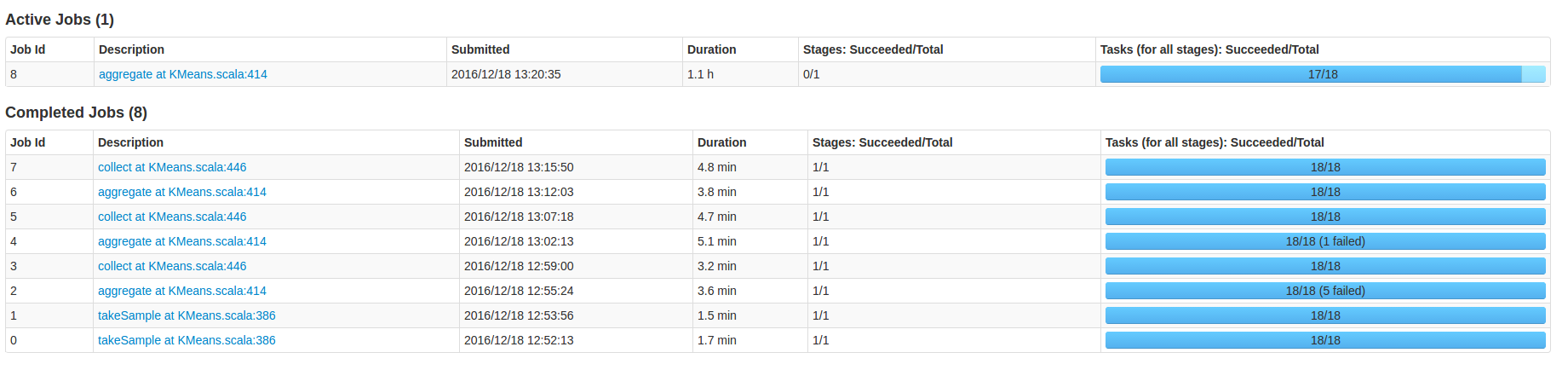
本实验的注意事项,刚开始一定要把所有的Executors 都利用起来,只是单纯的增加数据量.(程序中,通过固定partition个数为 5 来达到这个目的)

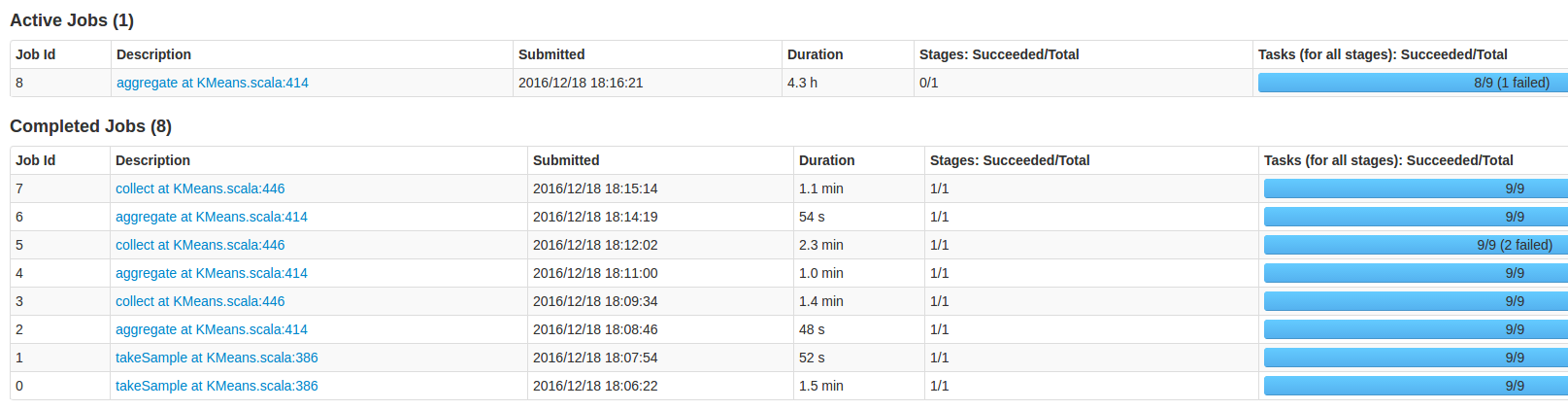
DataSet:2 dims/point

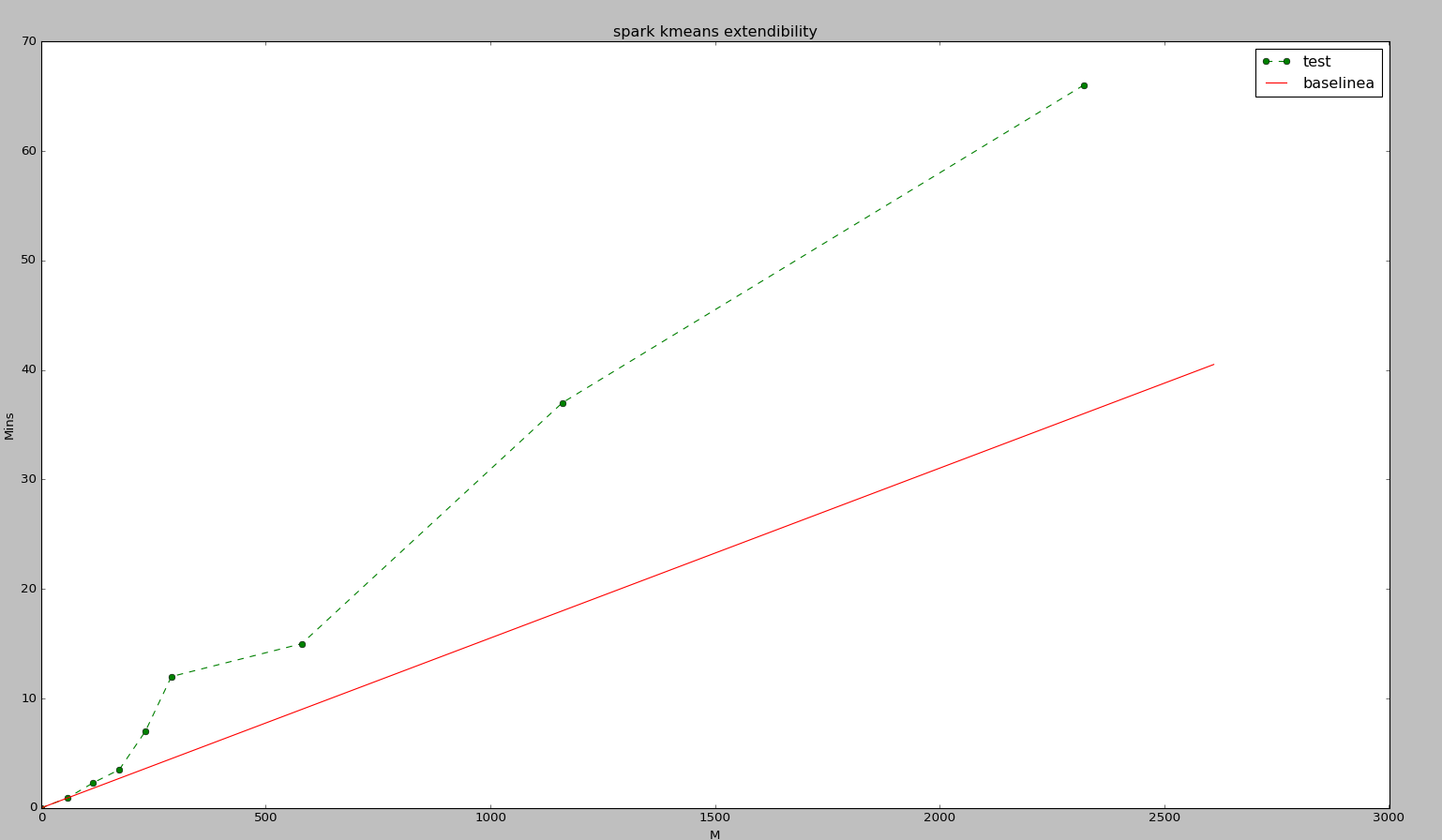
Configuration 5Executor（10Cores）4G/Executor

|  |  |  |
| --- | --- | --- |
| **DataSet Size** | **Run time** | **Error** |
| 58M | 52s | No |
| 116M | 2.3Mins | No |
| 174M | 3.5Mins | No |
| 232M | 7Mins | No |
| 290M | 12Mins | No |
| 580M | 15Mins | No |
| 1.16G | 37Mins | No |
| 2.32G | Failed | 进行aggregate 操作卡住了 |
| 2.32G | 1.1H | Yes |

Aggregate 卡死了







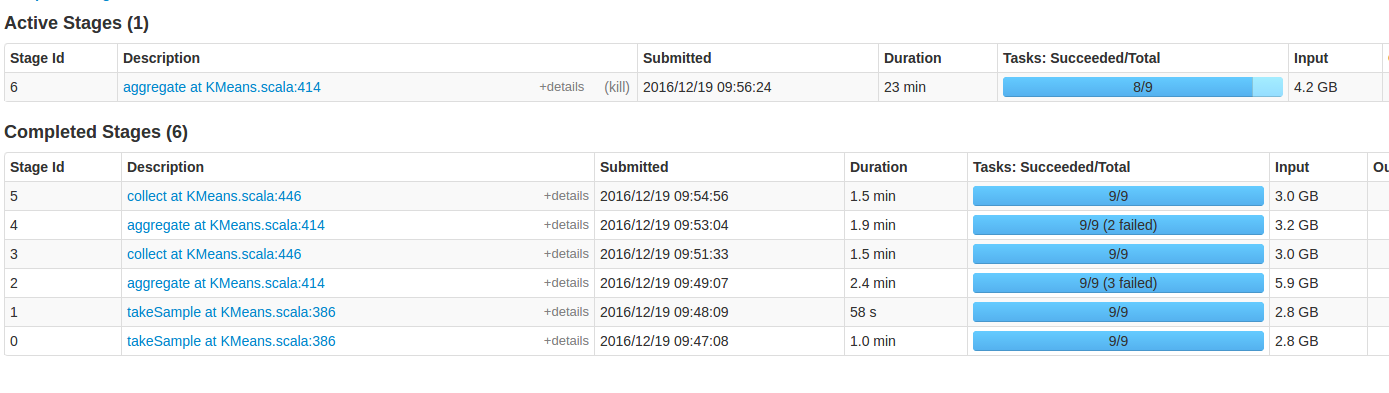
结论:算法整体具有线性可扩展性.

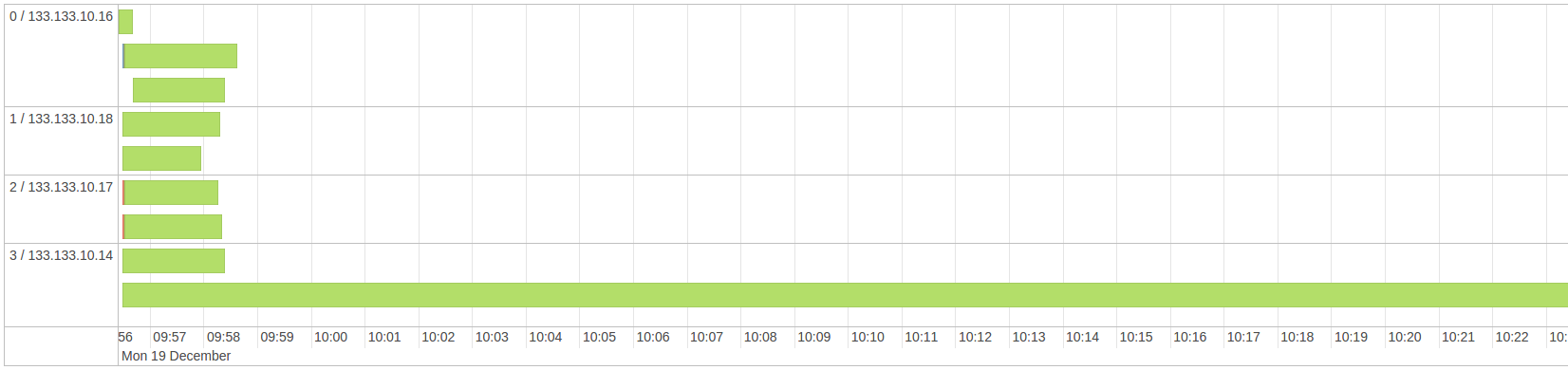
1. 算法加速比:

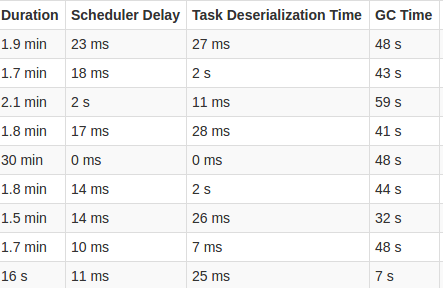
算法加速比这一项,很难测试,实验发现,对于固定的测试集,比如这里使用了1.16G,当我们设置的Executor过多,比如36,作业失败,当我们设置的过少吧比如4 作业失败,合理的设置比较少,这个测试目标很难达到.

Executor: 4G,2Cores

|  |  |  |
| --- | --- | --- |
| Number of executor | Time | Error |
| 36 | Failed | Aggregate 进行了太长的时间 |
| 18,9 | 19Min | No |
| 4 | Failed | GC 时间太长,以及个别Task 运行时间无限长,原因未知. |





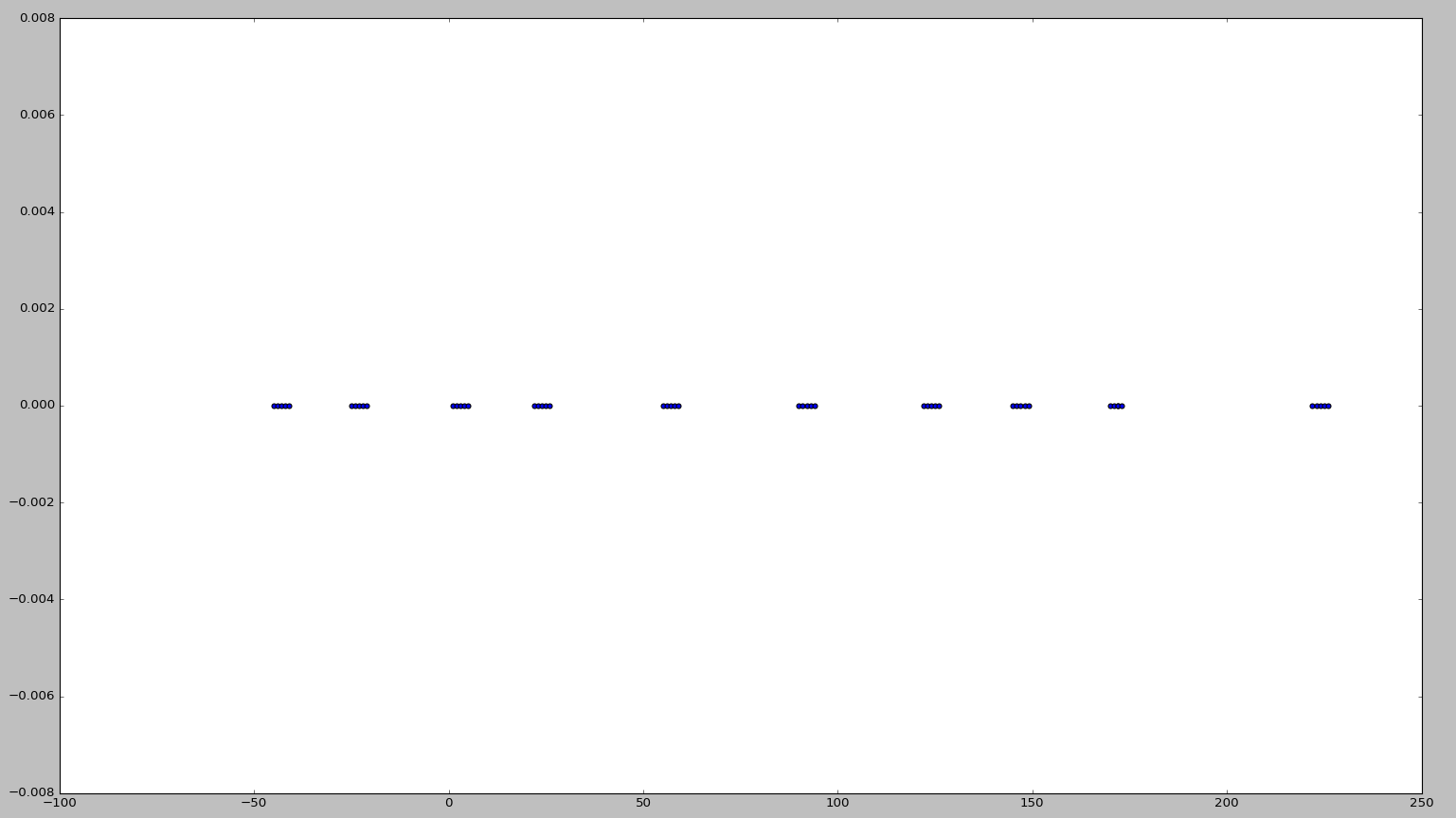


三.算法稳定性:

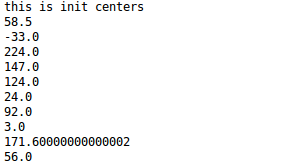
Spark目前算法稳定性欠缺:

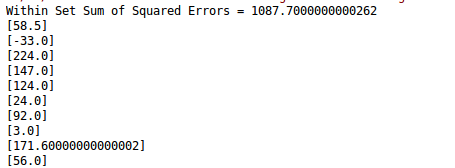
Scikit-learn重复跑Kmeans 100,00 次,没有出现下面的错误.

比如对于如下例子:

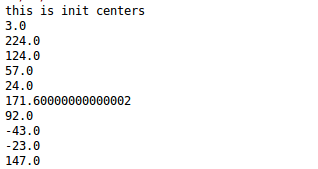


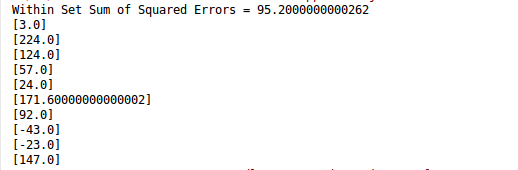
利用 Kmeans || 确定中心点却出现了如下结果:





正确的结果应该如下:





原因分析:

Spark 使用了 Kmeans||(Kmeans++ 的改进版)初始化中心点

Scikit-learn 使用了Kmeans++ 初始化中心点

具体原因待分析.