UITableView——reloadData与reloadSection性能比较

周五上午，测试，有bug：每次reset模拟器后，第一次进入界面，闪退，第二次进入界面，结果正常。

以下是这个bug的错误日志：

*** Terminating app due to uncaught exception 'NSInternalInconsistencyException', reason: 'Invalid update: invalid number of rows in section 1. The number of rows contained in an existing section after the update (1) must be equal to the number of rows contained in that section before the update (2), plus or minus the number of rows inserted or deleted from that section (0 inserted, 0 deleted) and plus or minus the number of rows moved into or out of that section (0 moved in, 0 moved out).’

发现刷新列表时，闪退。问题出在：[self.tableViewreloadSections:[NSIndexSetindexSetWithIndex:2]withRowAnimation:UITableViewRowAnimationNone];

这个方法我没用过，从方法名看出，这个方法是局部刷新列表的某个section。但一般我都是用reloadData，刷新整个列表。于是，我把reloadSection改回reloadData，发现这个两个bug都不再出现了。

按道理，使用reloadSection应该没有问题啊，但为什么这里不能使用reloadSection呢？

其实，仔细看错误日志就明白，问题出在该section的row是动态变化的。

那么，问题来了，为什么要把reloadData改成reloadSection呢？

应该是因为reloadSection效率更高，速度更快？

但真的是这样吗？我想验证一下。

好吧，我承认我是有多无聊？闲的蛋疼……

新建一个test工程，创建一个tableview，3个section，每个section有30行。这个tableview总共有90行数据。

代码如下：

```
    self.tableview.delegate = self;

    self.tableview.dataSource = self;

//    for(int i=0;i<1000000;i++){

//        [self.tableview reloadSections:[NSIndexSet indexSetWithIndex:2]
withRowAnimation:UITableViewRowAnimationNone];

    for(int i=0;i<1000000;i++){

        [self.tableview reloadData];

- (void)didReceiveMemoryWarning {

    [super didReceiveMemoryWarning];

    // Dispose of any resources that can be recreated.
```

```
-(NSInteger)numberOfSectionsInTableView:(UITableView *)tableView{

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:
(NSInteger)section{

-(UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath{

    UITableViewCell *cell = (UITableViewCell *)[tableView
dequeueReusableCellWithIdentifier:@"cell"];

    cell.textLabel.text = [NSString
stringWithFormat:@"section:%li,row:%li",(long)indexPath.section,
(long)indexPath.row];

    for(int i=0;i<1000;i++){

        int y = indexPath.row+indexPath.section;

    UIImageView *image = [[UIImageView alloc]
initWithFrame:CGRectMake(200, 5, 90, 75)];

    image.image = [UIImage imageNamed:@"254"];

    [cell addSubview:image];
```

在iPhone 6 plus模拟器下做了三组对照试验，试验结果如下：

在每个cell只有一个label的情况下，分别用reloadSection，reloadData方法刷新列表，10次，100次，1000次，10000次，100000次，1000000次时，时间、CPU、内存的比较：

label | label+image | label+image+for | 图表

10	start	end	time（s）	CPU（%）	memory（M）
reloadSection	0.775	0.777	0.002	0	42
reloadData	0.949	0.950	0.001	0	42
100	start	end	time	CPU	memory
reloadSection	0.084	0.085	0.001	0	42.1
reloadData	0.291	0.292	0.001	0	42.3
1000	start	end	time	CPU	memory
reloadSection	0.698	0.706	0.008	0	42.1
reloadData	0.631	0.640	0.009	0	42.2
10000	start	end	time	CPU	memory
reloadSection	0.170	0.220	0.050	0	42.7
reloadData	0.610	0.771	0.161		43.5
100000	start	end	time	CPU	memory
reloadSection	0.064	0.602	0.538	0	48.1
reloadData	0.814	1.704	0.89	0	52.9
1000000	start	end	time	CPU	memory
reloadSection	3.133	7.617	4.484	99	114
reloadData	16.883	25.889	9.006	99	184

每个cell有一个label和一张图片，比较

label | label+image | label+image+for | 图表

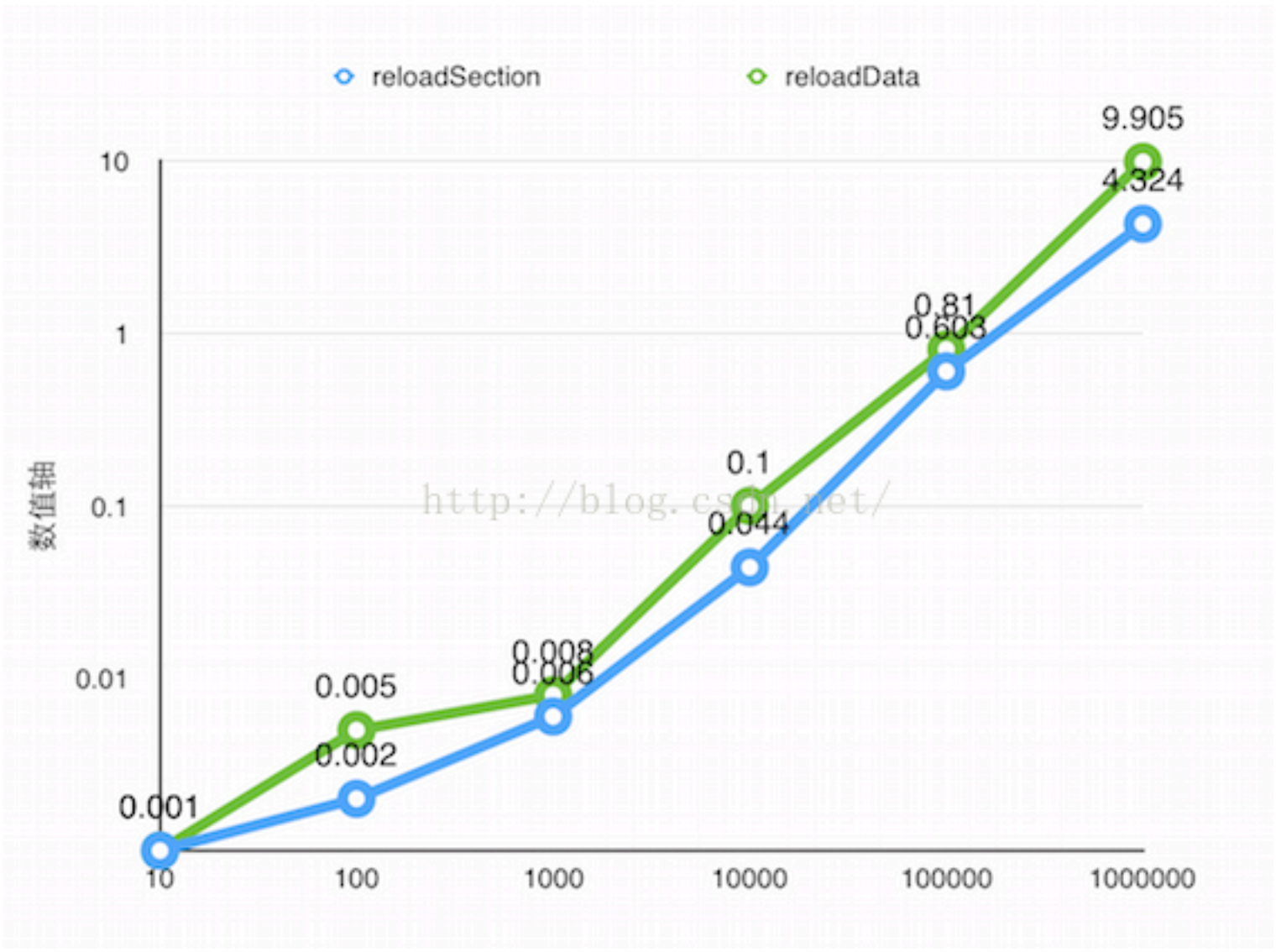
10	start	end	time（s）	CPU（%）	memory（M）
reloadSection	0.834	0.835	0.001	0	42.1
reloadData	0.353	0.354	0.001	0	42.1
100	start	end	time	CPU	memory
reloadSection	0.671	0.672	0.001	0	42.1
reloadData	0.166	0.169	0.003	0	42
1000	start	end	time	CPU	memory
reloadSection	0.169	0.177	0.008	0	42.1
reloadData	0.047	0.055	0.008	0	42.1
10000	start	end	time	CPU	memory
reloadSection	0.352	0.405	0.053	0	42.6
reloadData	0.244	0.317	0.073	0	43.5
100000	start	end	time	CPU	memory
reloadSection	0.672	1.131	0.459	0	48.2
reloadData	0.716	1.579	0.863	0	52.7
1000000	start	end	time	CPU	memory
reloadSection	5.339	9.867	4.528	99	112.8
reloadData	6.237	20.909	14.672	99	203

每个cell有一个label、一张图片和执行1000次的for循环，比较

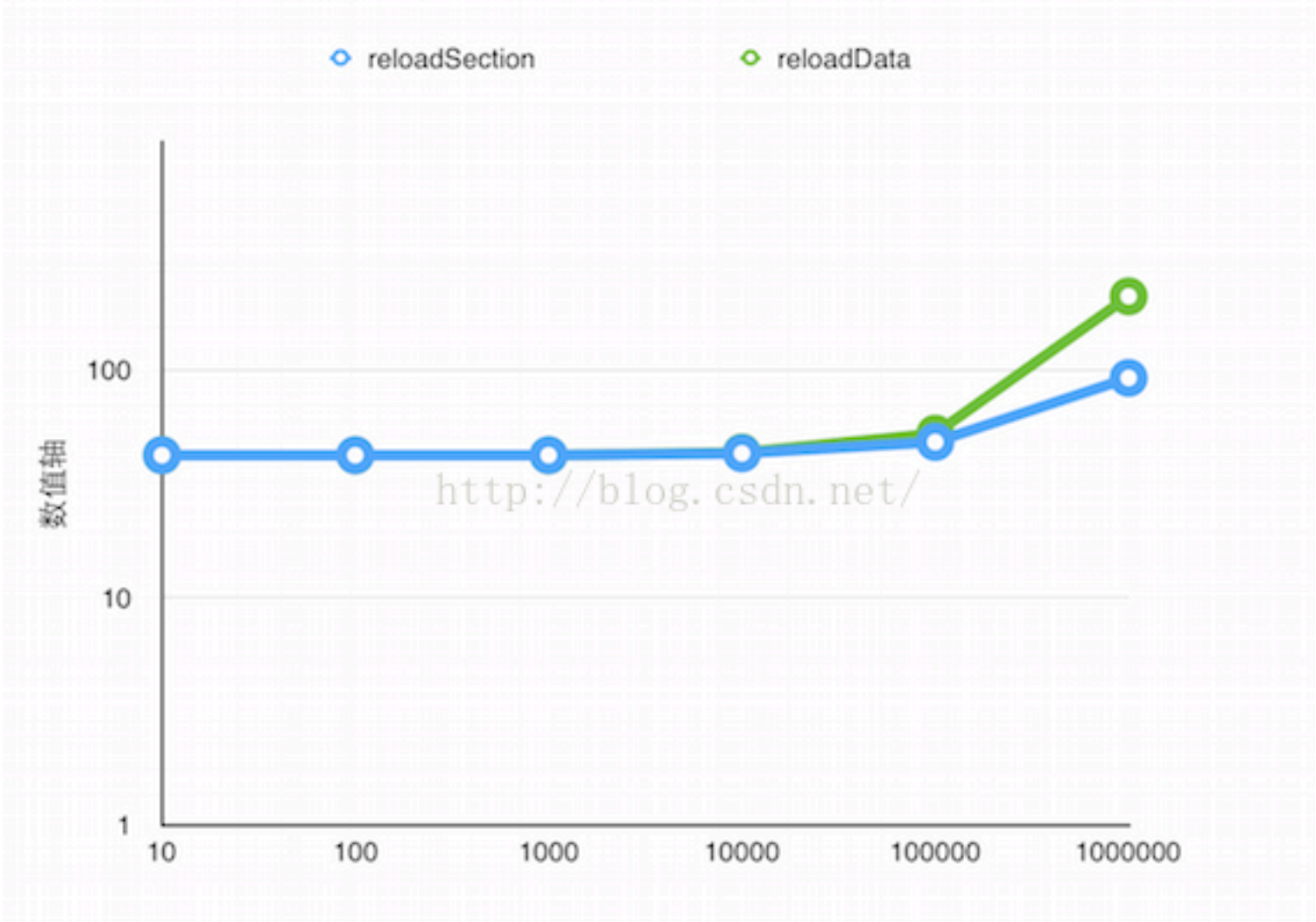
label | label+image | label+image+for | 图表

10	start	end	time（s）	CPU（%）	memory（M）
reloadSection	0.918	0.919	0.001	0	42.1
reloadData	0.402	0.403	0.001	0	42.1
100	start	end	time	CPU	memory
reloadSection	0.265	0.267	0.002	0	42.1
reloadData	0.096	0.101	0.005	0	42.1
1000	start	end	time	CPU	memory
reloadSection	0.950	0.956	0.006	0	42.1
reloadData	0.714	0.722	0.008	0	42.1
10000	start	end	time	CPU	memory
reloadSection	0.782	0.826	0.044	0	42.9
reloadData	0.663	0.763	0.1	0	43.3
100000	start	end	time	CPU	memory
reloadSection	2.653	3.256	0.603	0	48.1
reloadData	1.576	2.386	0.81	0	52.4
1000000	start	end	time	CPU	memory
reloadSection	1.828	6.152	4.324	99	92
reloadData	41.589	51.494	9.905	99	210

每个cell有一个label、一张图片和执行1000次的for循环，
reloadSection,reloadData,在执行10次~1000000次花费时间对数表（前两种情况就不列图表了，和最后情况类似）



reloadSection,reloadData,在执行10次~1000000次花费内存对数表



从这些实验数据发现：在10000次以内，reloadSection和reloadData两者在时间、CPU、内存相差并不大，甚至在某些情况，reloadData性能要优于reloadSection。大于10000次以后，reloadSection的性能高于reloadData。

而我们实际的项目中几乎不会刷新某个列表超过100次，两者性能差不多，但reloadSection不能用于row，section动态变化的情况下，所以还是更加推荐使用reloadData方法。

总结：平时编码过程中，通常会根据自己的经验判断，采用某种性价比更高的方式。但事实真的是这样的吗？我们很少去想这个问题，也几乎不会去验证，因为我们觉得理所当然。套用知乎的一句名言：凡事先问是不是，再问为什么，警戒自己，不要被自己所谓的“经验”误导。

————————————————————————————我是分割线
————————————————————————————

之后有同事提到：

关于reloadSections vs reloadData

测试例子考虑的是数据源不变的情况下cell的重绘制

但复杂场景，之所以考虑部分刷新reloadSections，是因为刷新每个section,cell不单单是cell的绘制，也都有包含I/O和运算操作

而这个时候 reloadData会触发其他不必要的运算和I/O

tableview的主要作用是展示数据，而I/O操作其实不适合放在cell中。

包括图片的下载其实不宜放在cell中，这会导致界面卡顿。

更好的方法是：将获取数据源的代码放到次要线程中，这样主线程才能更好的加载视图。

我们常用的MVC，MVVM架构也就是为了将业务逻辑与视图尽量剥离，让UI更好的展示数据。

至于运算操作，在第三种情况中，我加了一个执行一千次的for循环来模拟复杂的运算。

reloadData相比reloadSection，前者执行运算操作的次数是后者的三倍。

在这种情况下，两者的性能在100000次以内，依然相差无几。

而在数据源动态变化的情况下，例如在某些页面中，每个Section中的row是动态变化的，单独使用reloadSection会导致文中的bug。而使用reloadData不会出现这个bug。

而reloadSection相比reloadData多一点的是，在刷新的时候有动画。

更多情况下，是搭配beginUpdates和endUpdates来实现deleteSections:withRowAnimation:的功能。