## 背景介绍

随着商业网络流量的快速增长，大规模数据的处理越来越成为网络研究的一大挑战。为了能够有效解决这个问题，Google公司随即推出了MapReduce框架，该框架的主要思想是将数据流量分配到许多个运算节点上，再由这些运算节点并行处理这些数据。Hadoop是基于MapReduce一个开源的框架，它可以通过将流量分配至数千个运算节点来处理PB量级的数据，因为它的高性能和扩展性，使得它获得了很高的关注度，许多大型公司或团队，都将Hadoop应用于数据密集型地应用。

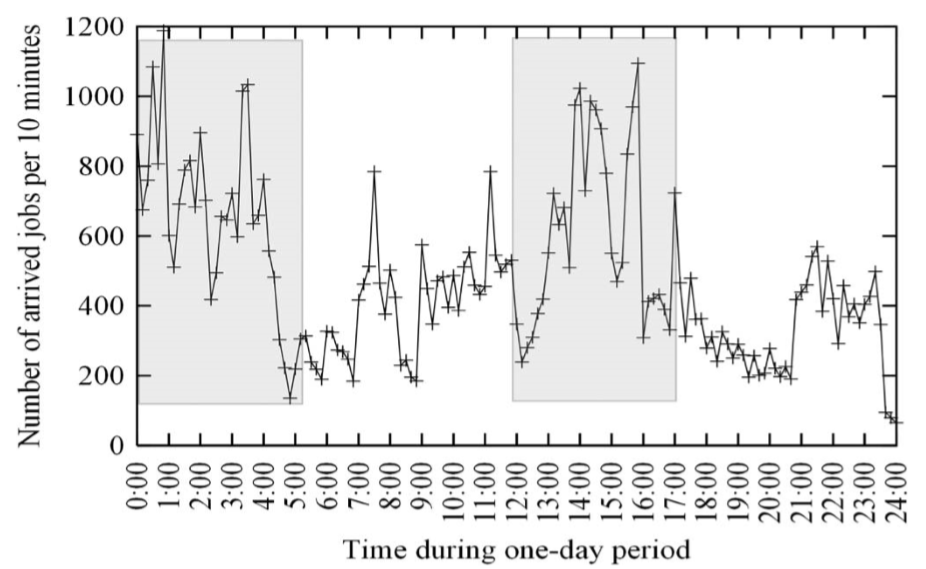
通过调研，我们获得了淘宝网2011年12月4日到2011年12月20日的业务数据，这些数据是来自名为云梯平台的Hadoop Cluster框架，里面包含了2000个节点的超过百万的job数量。通过对这些数据特征的分析可以帮助找到系统的性能瓶颈，从而为改善Hadoop框架的性能提供思路。

## 数据特征分析

### 2.1对于Job特征的分析

#### 2.1.1 job到达速率特征

图1展示了在一天内每十分钟的到达job数目变化图，由图可以看出，job的到达速率在1:00 am到4:00 am内达到当日第一个峰值，并且在1:00 pm至4:00 pm达到当日第二个峰值，峰值时十分钟间隔内的job到达数量最高可以超过1100。

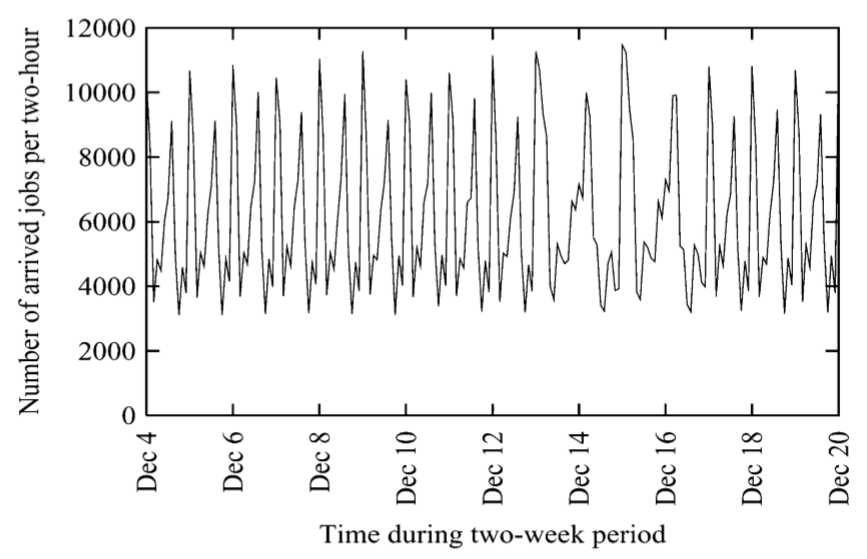


Figure

分析可知，第一个峰值主要时由定期性的job组成的，它们是由应用开发者预先定义好的，并且在每天重复发生。这些job被安排在每天凌晨执行，大多数的重复性job都是相同的，并且定期生成报告。第二个峰值主要由临时的job组成，它们临时被定义，并由应用开发者手动提交上去。

通过曲线拟合分析，可以发现这些job的到达时间序列服从两个泊松过程，假设单位时间为1分钟，则这两个泊松过程的值分别为76和42。

图2描绘了在两周内job每2小时的到达数量变化曲线。由图可知，除了在12月13日和12月14日出现了一些突发job外，每天的job到达速率模型是相似的，突发job情况的出现主要是由于12月12日的大型活动所致，这恰巧可以用来评估该活动的影响。

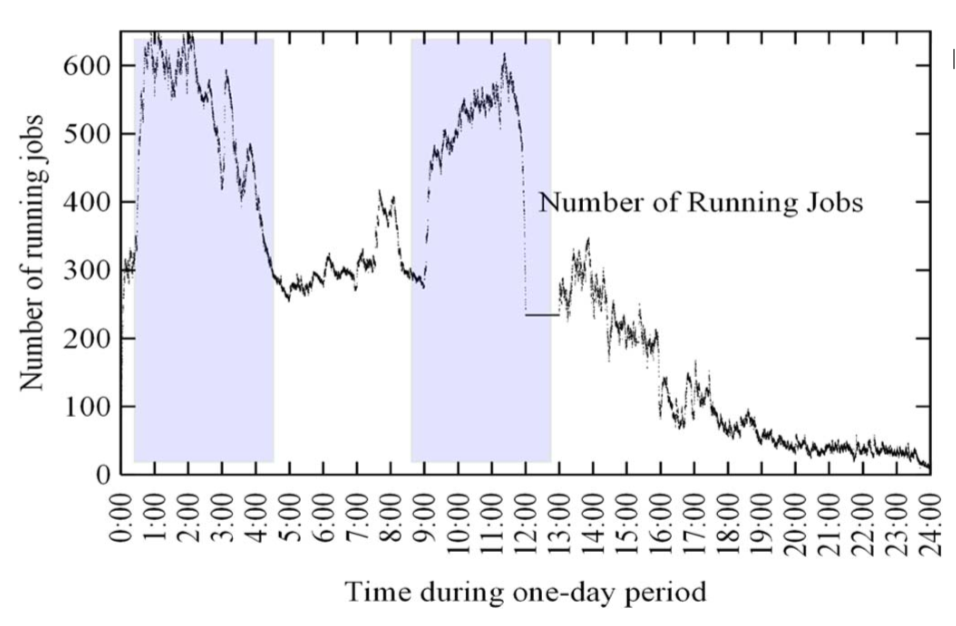


Figure

通过以上数据可知，每日job到达速率基本服从同一个模型，job的到达时间服从泊松分布。这些结论暗示了可以根据job的到达速率模型来制定更加节能的资源调度方案。

#### 2.1.2 同时运行的job数量特征

图3展示了云梯平台上同时在运行的job的数量在一天内的变化特征，其中取样间隔为5秒。由图线可知，同时在运行的job数量首先在1:00 am到4:00 am内达到第一个峰值，其最大值超过了600，然后在5:00 am时下降至大约300，之后在9:00 am时达到第二个峰值，这时正好是其公司员工开始上班的时间。



Figure

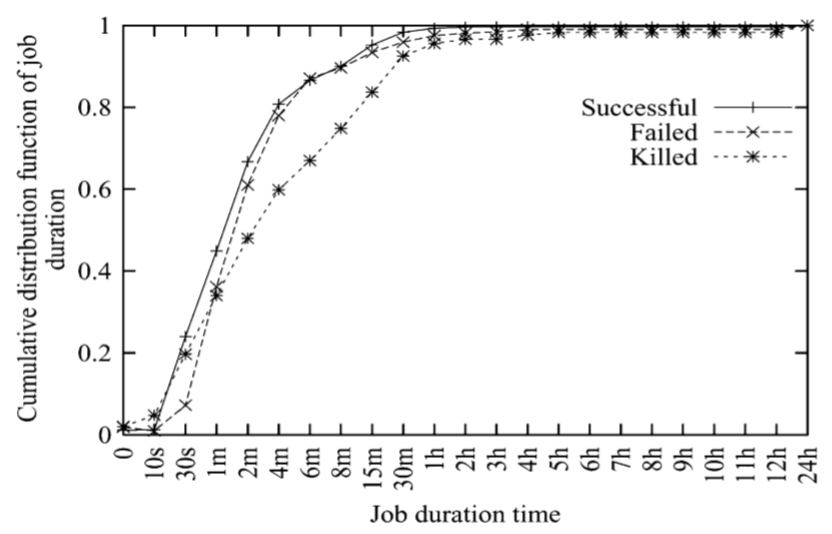
这里的两个峰值的情形和job到达速率的那两个峰值的情形有几分相似，然而，对比图1和图3后可以发现，它们的第二个峰值的到达时间有所不同。这个区别可以归结为job持续时间的影响，更详细的调查可以发现，上午到达的job相比于下午而言中间包含了更多持续时间较长的job，因此导致了上午的同时运行的job数量可以达到峰值。

实际上同时运行的job数量上限被设置为了300，但是因为实际运行时却超出了这个上限，这是因为一些优先级较高的job到达后会致使调度者结束掉一些task并且当没有足够的时隙时会为其抢占时隙，使得一些正在运行的job不得不先转换为等待状态。

通过以上数据分析可知，抢占时隙的情况时有发生，因此优化时隙方案是关键。

#### 2.1.3 job的执行时间长度特征

图4展示的是两周内测得的执行成功、执行失败和被强行结束的job的执行时间CDF曲线，其中执行失败的job就是那些因为task错误而被取消的job，强行结束的job就是那些因为时隙抢占或其他原因被Hadoop执行者强行终止的job。整个job的执行时间是指从job的第一个task开始到最后一个task结束的时间间隔长度。由图可知，job的执行时间具有长尾分布的特点，超过80%的job执行时间不超过4分钟，执行时间最长的job可以达到16小时24分钟。执行失败的job比被强行结束的job消耗更短的时间。90%的执行失败的job在8分钟内就被取消了，同时90%的强行结束的job都在25分钟内被终止。通过Levenberg-Marquardt算法，可以将job的执行时间拟合为Weibull分布、指数分布或者对数正态分布。再通过KS校验可以发现，对数正态分布是执行成功、执行失败和强行结束的job的执行时间拟合最好的分布模型。

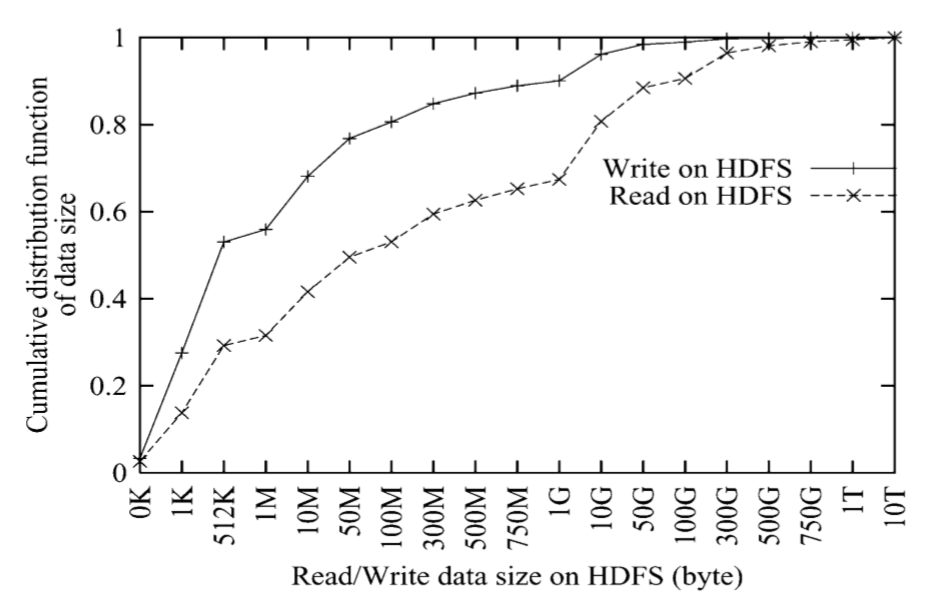


Figure

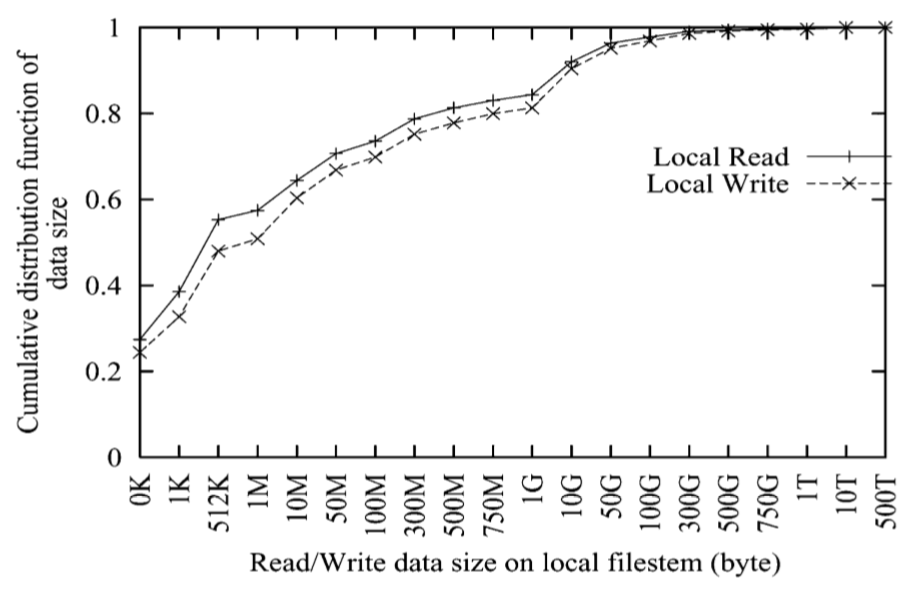
#### 2.1.4 job数据大小特征

Job数据字节的读和写包含了两个部分，一个是HDFS的字节读写还有一个是本地磁盘的字节读写。HDFS的字节读写包含了从HDFS读取的输入数据和HDFS写入的输出数据，本地磁盘的字节读写包括了在MapReduce平台中间产生的数据，这些中间数据被直接存储在了本地磁盘中。

图5和图6分别展示了HDFS和本地磁盘上每个job的读写的字节数CDF曲线。通过图线可以发现80%的写入到HDFS的job都小于64MB，而50%从HDFS读取的job都小于64MB，这和期望读取的数据要远远大于写入数据的规律保持一致，因为大多数的job输入数据要远大于输出数据。80%的读取和写入本地磁盘的job都小于1GB，其中，写入的job的数据大小要略大于读取的job。这个结果依然符合期望，因为中间数据往往是job执行过程中产生的存储到本地磁盘的数据。



Figure



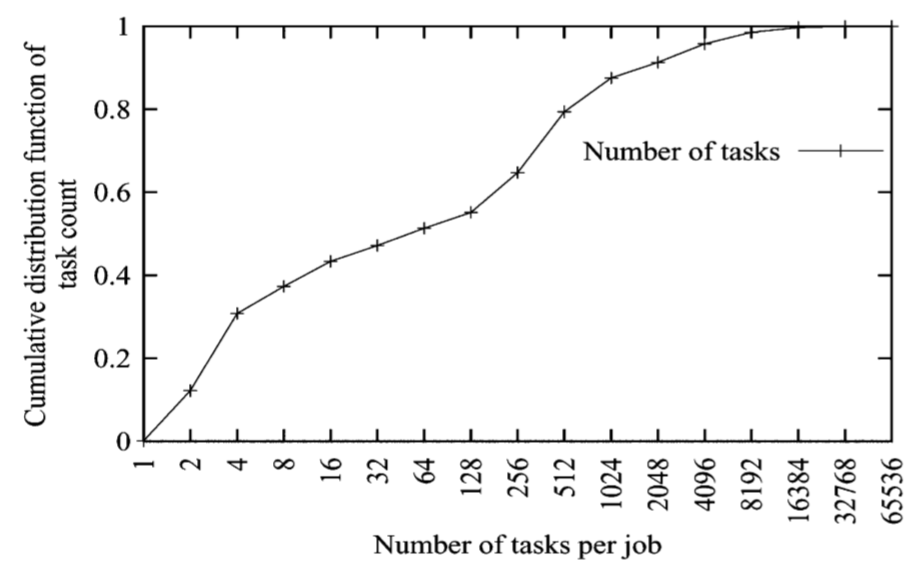
Figure

由以上的数据分析可知，80%的job写入到HDFS的文件都小于64MB，之前就有研究报告说大量的小文件会降低NameNode的效率，因此，HDFS上小文件的存储效率是非常重要且亟待解决的问题。

### 2.2对于Task特征的分析

#### 2.2.1 task数量特征

图7描绘了每个job的task数量CDF曲线，超过40%的job包含的task都不足10，还有50%的job包含的task数量在10到2000之间，最大的job包含了91798个task，而最小的job只包含一个task。表1也列出了job的大小分布情况。



Figure

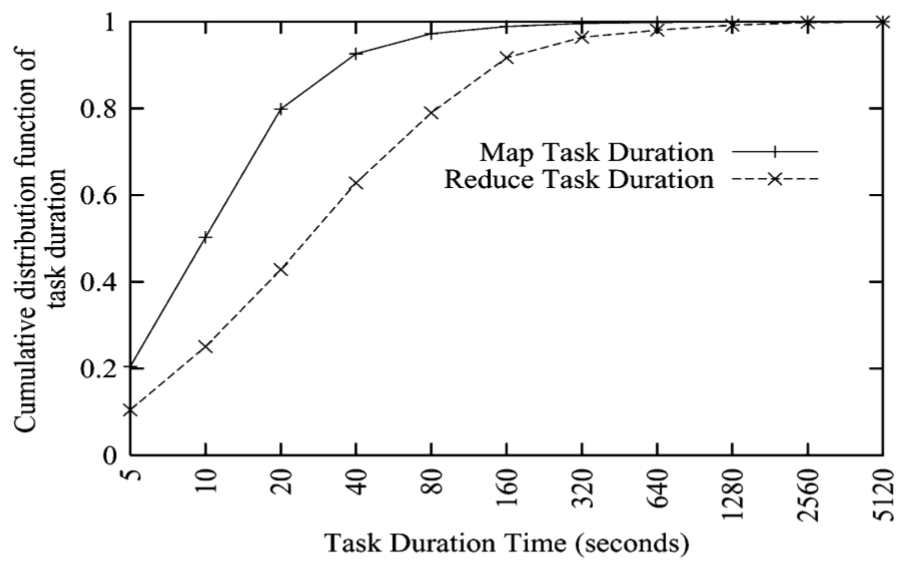
|  |  |  |
| --- | --- | --- |
| Task count within a job | Number of jobs | Percentage |
| <10 | 370,062 | 40.57% |
| 10-500 | 358,751 | 39.33% |
| 500-2,000 | 109.733 | 12.03% |
| >2,000 | 73,611 | 8.07% |

Table

由以上数据分析可知，在云梯平台上，job中包含的task的数量多样，小尺寸job占了大部分，中等或大尺寸的job都各占剩下一部分，因此，hadoop调度器需要提升小尺寸的job的调度方案。

#### 2.2.2 task持续时间特征

图8展示了map和reduce的task持续时间的CDF曲线，其中超过50%的task的执行都不超过10秒，最长的task持续时间达到了20分钟48秒。



Figure

为了更深入的理解task的持续时间分布，可以将task根据它们的持续时间分为三类：second-task、minute-task和hour-task。second-task执行时间不超过一个固定值，如120秒，minute-task包含了执行时间在2分钟至1小时之间的task，hour-task指的是持续时间超过1小时的task。表2列出了各类task的总数。

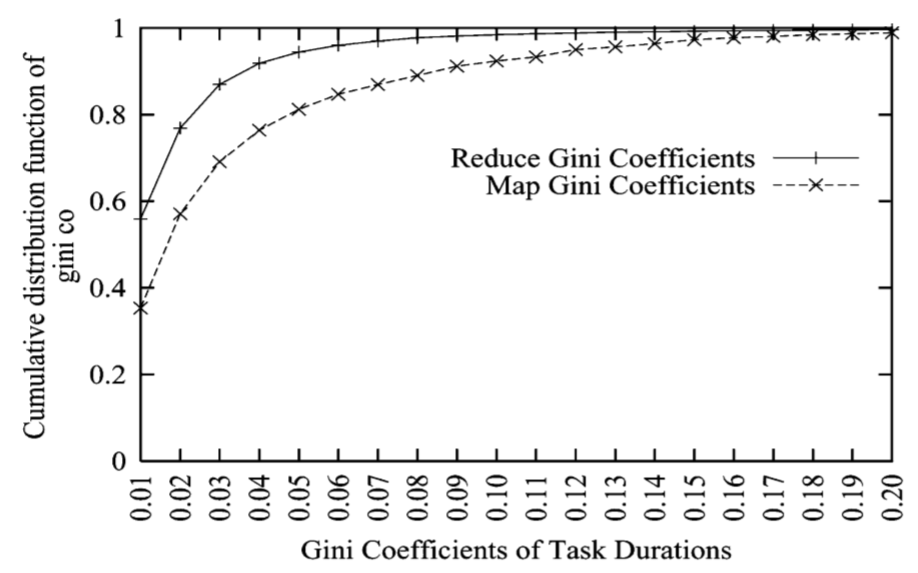
|  |  |  |
| --- | --- | --- |
| Task type | Quantity | Percentage |
| Map second-tasks(<120s) | 12954135 | 98% |
| Map minute-tasks(2m-1h) | 206234 | 1.5% |
| Map hour-task(>1h) | 1290 | 0.5% |
| Reduce second-tasks(<120s) | 1352878 | 87% |
| Reduce minute-tasks(2m-1h) | 184939 | 12% |
| Reduce hour-tasks(>1h) | 596 | 1% |

Table

由以上数据分析可知，map task和reduce task的持续时间都服从对数正态分布。

#### 2.2.3 task的一致性

Task一致性描述的是task持续时间之间的差异，理想状态下，同一个job的task的执行时间是相似的，task一致性不仅指明了task的执行效率也指明了并行程度。这里使用基尼系数来描述task持续时间的一致性。图9展示了job的map task和reduce task的基尼系数CDF曲线，由曲线可知，超过95%的job的map task持续时间基尼系数都小于0.15，超过95的job的reduce task持续时间基尼系数要小于0.05。



Figure

由以上的数据分析可知，task不一致的问题在map task中要比在reduce task中严重的多，这和我们传统的看法不一致。经过深入分析得知，这是由像table JOIN这样歪曲的map操作造成的，起到广播作用的map task比其他的要执行更多的I/O操作。

#### 2.2.4 数据的位置特征

为了减少数据的迁移，Hadoop中比较合理的调度方案是使用延迟调度机制，最优化数据task的位置，将task放置在包含它输入数据的节点上，job调度器选择数据离TaskTracker最近的task，尝试将task放置在同一个节点上，否则就放置在同一个rack上，再不行就放置再其他rack上。表3列出了所有task的数据位置信息，可以看出，92.7%的task都在数据所在的本地节点上。

|  |  |
| --- | --- |
| Locality level | Percentage |
| Node-local | 92.7% |
| Rack-local | 4.4% |
| Remote-rack | 2.9% |

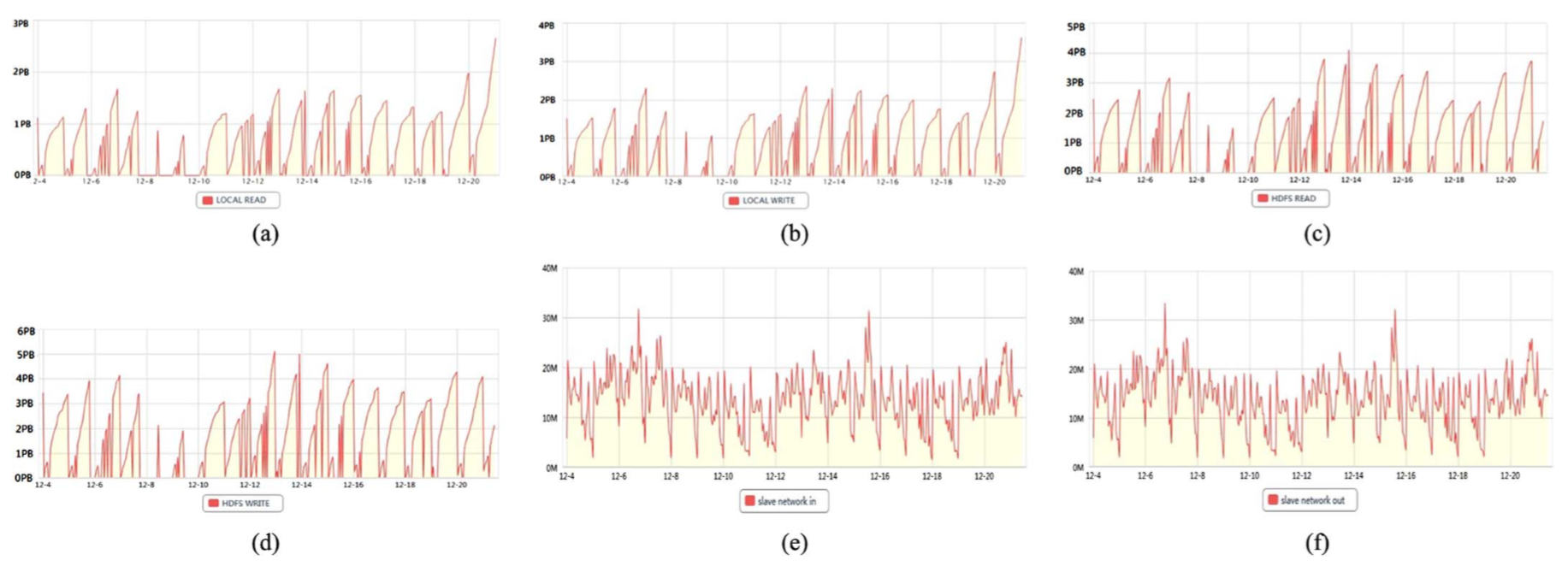
Table

由以上数据分析得，基本上多数调度器都可以实现task和数据的位置保持一致，因此，继续去减少数据迁移不是特别必要。

### 2.3资源利用情况分析

#### 2.3.1 I/O数据特征

图10(a)和(b)分别展示了本地磁盘在两周内每天的I/O数据变化，本地磁盘的读取和写入主要是由map平台中间数据的产生导致的。由图可知，本地磁盘的读取和写入字节数分别达到了约1.5PB和2PB每天，因为hadoop cluster由2000个节点组成，因此每个DataNode的数据读取和写入速度平均为9MB/s和12MB/s。图10(c)和(d)分别展示了HDFS在两周内每天的I/O数据变化，HDFS的读取主要是来自HDFS的输入数据，HDFS的写入主要是写入到HDFS的数据，可以发现数据在HDFS上的读取和写入分别达到了3PB每天和4PB每天，因此每个DataNode数据平均读取和写入速度分别为18MB/s和25MB/s。



Figure

根据上述分析可知，HDFS和本地磁盘上读取和写入的数据量是很大的，采用MapReduce会很有益处，采用分布式的内存系统会是一个减少读取和写入数据流的较好的解决方案。

#### 2.3.2 网络传输特征

对于每一个job的执行，云梯平台的节点间都会发生着数据的传输或交换，一个节点上的网络流量可以分为流入数据和流出数据，前者是指从其他节点流入到该节点的数据，后者是指从该节点发送到其他节点的数据。

图10(e)和(f)分别展示了两周内网络中每秒的流入数据和流出数据图，Y坐标代表网络每个节点I/O速率，这些I/O数据由读取的输入数据、shuffle和写入的输出数据组成，在大多数时间，流入数据和流出数据流量大小在10MB/s和20MB/s之间。

由上述数据分析可知，网络的负载比较繁忙，随着数据量的不断增加和集群服务器的规模不断变大，网络传输可能会成为系统潜在的性能瓶颈，所以需要高效的数据部署和调度策略来减小网络负载。