Resilient Distributed Datasets

A Fault---Tolerant Abstraction for In-Memory Cluster Computing

Matei Zaharia uc berkely nsdi 12

弹性分布式数据集阅读报告：
邓志会 2015210926
元东 2015210938

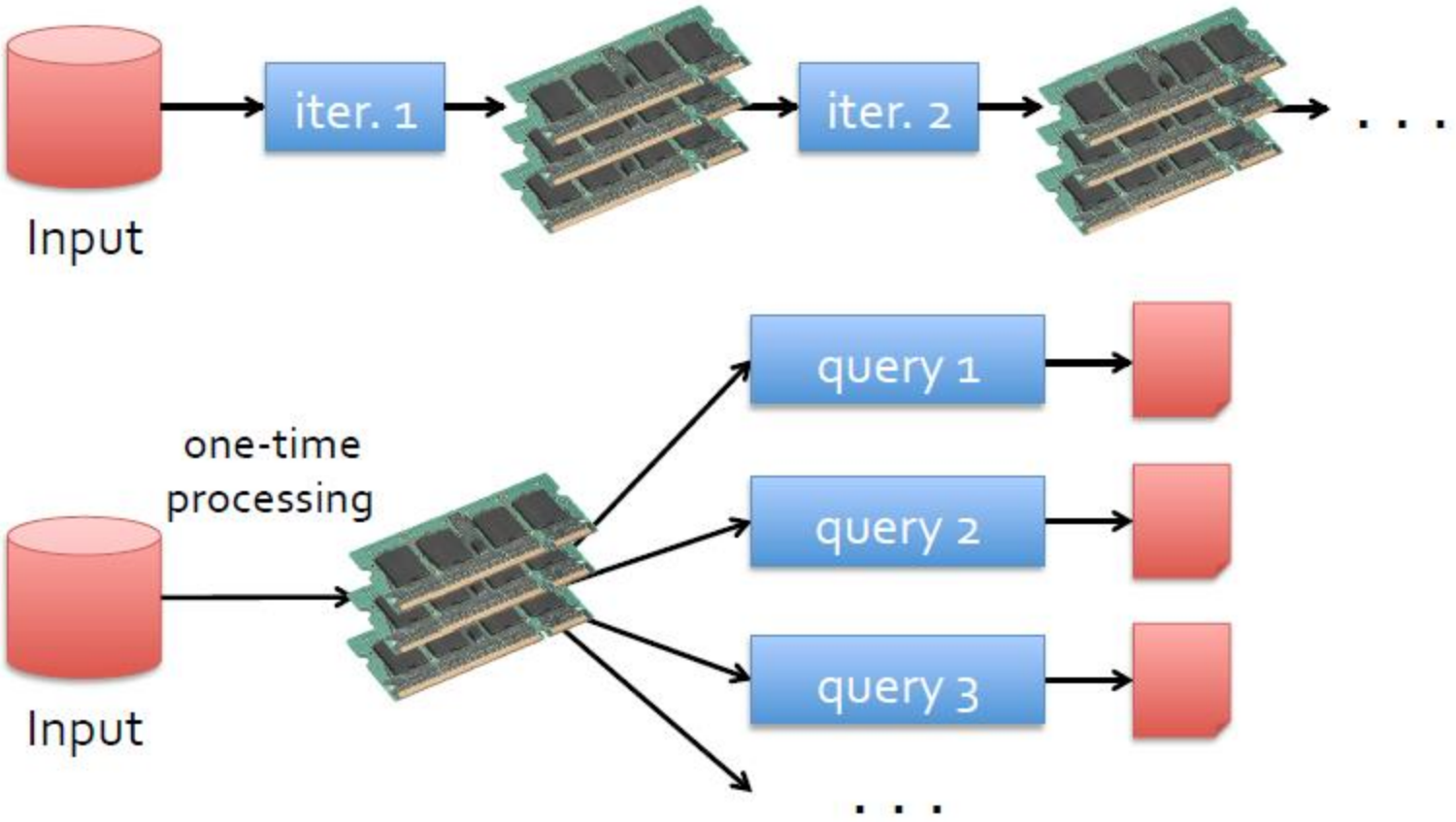
背景动机

- MapReduce 使得在很大但是不可靠的集群上进行大数据分析得到很大的简化
  但是MapReduce的性能还不够

用户希望它能支持更加复杂，多个阶段的应用程序（迭代式的机器学习和图运算），交互式自适应查询

mapReduce的瓶颈就是高效数据共享单元
mapReduce只能通过持久存储进行共享数据

设计目标：内存数据共享



内存共享比网络磁盘快10×100倍

挑战

设计容错并且高效的分布式内存抽象
已有的基于细粒度更新存储抽象接口：RAMCloud，databases,distributed mem,Piccolo

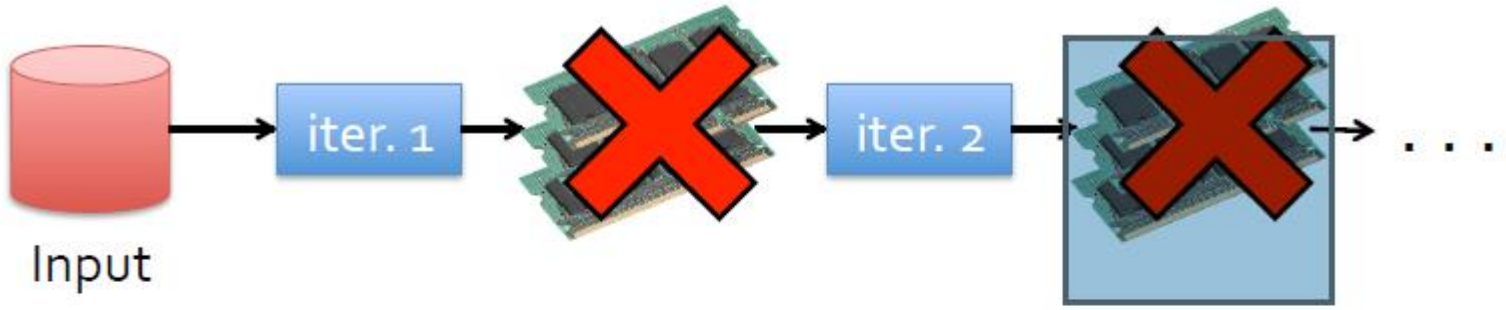
需要在节点之间复制数据或日志来做容错
对于数据密集应用开销大
比内存的写慢很多

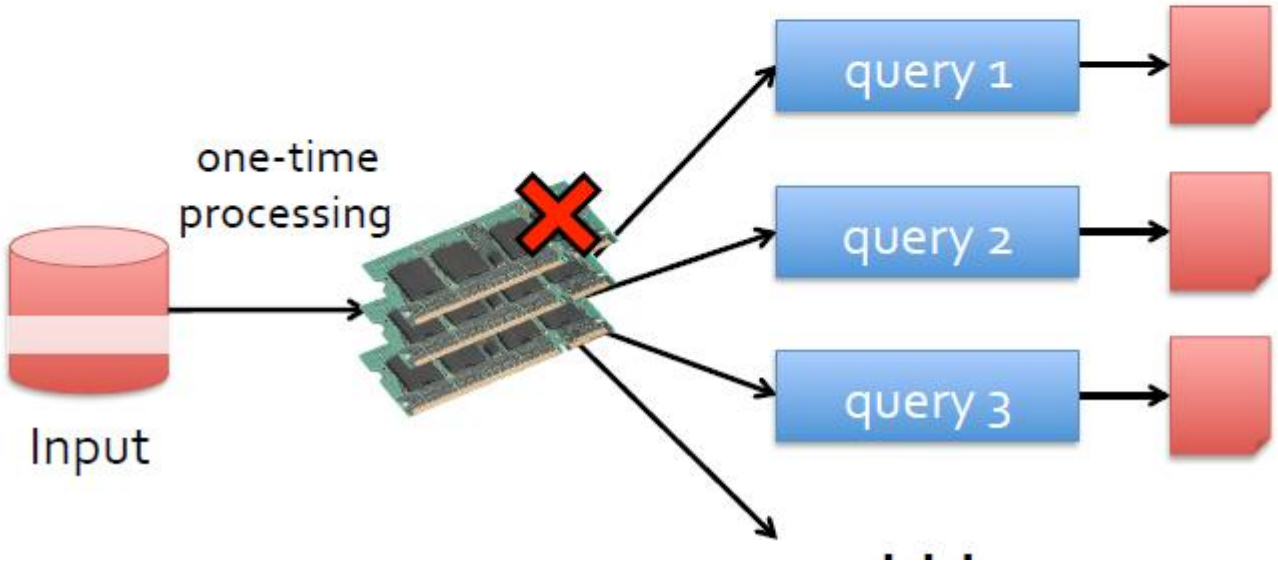
解决方案：弹性分布式数据集(RDDs)

分布式共享内存受限形式
不变，分隔的记录集合
只可以通过粗粒度的确定转换来构建(map,filter,join)

通过lineage实现有效的错误恢复
记录一个操作应用许多元素
重新计算失败的丢失的分区
如果没有错误则没有开销

RDD 恢复


iter. 1
iter. 2
Input

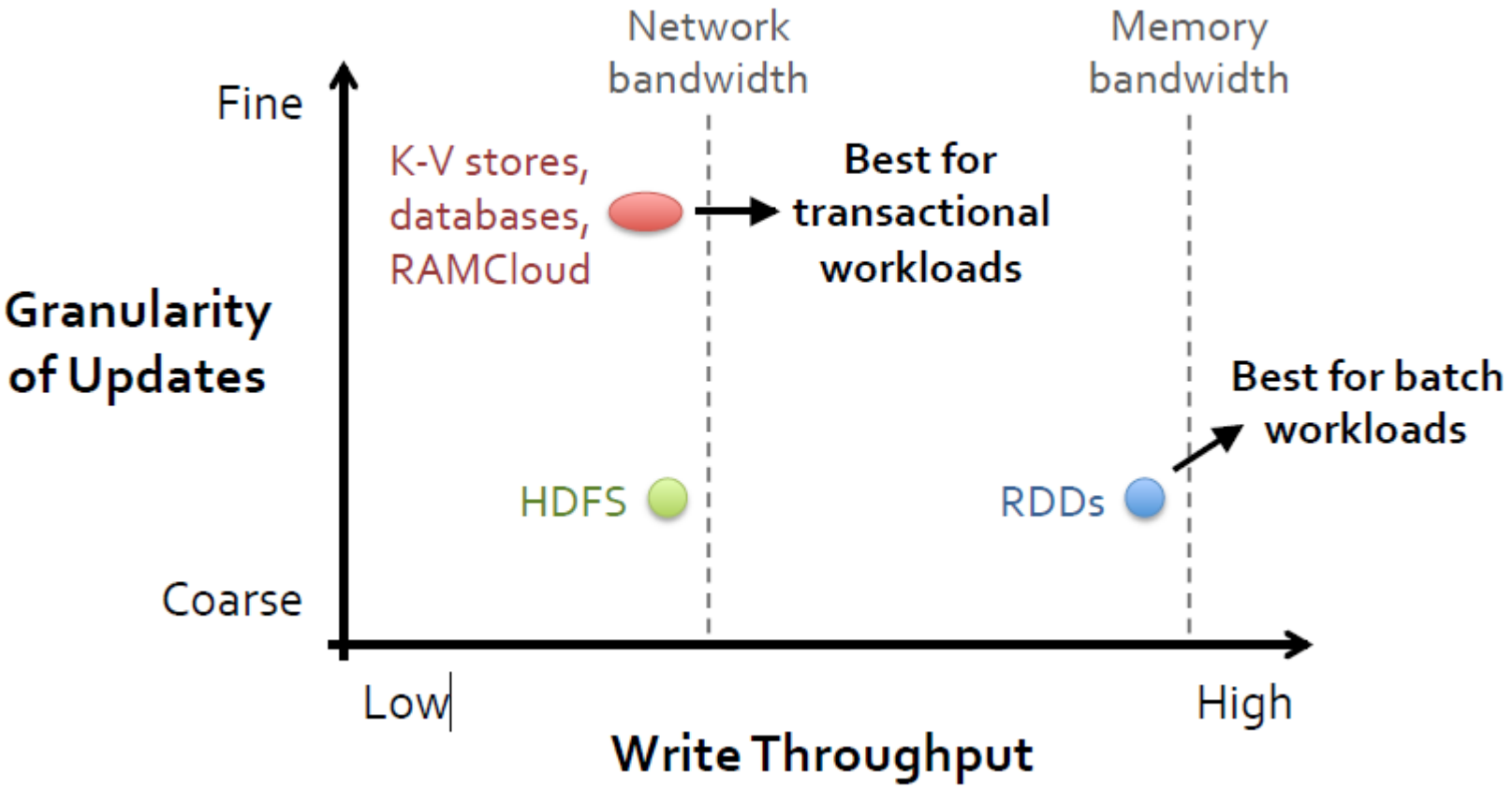

one-time
processing
query 1
query 2
query 3
Input

RDDs 概述

尽管有诸多限制，RDDs可以很好表达许多并行算法

统一多流编程模型
数据流模型：MapReduce，Dryad,SQL
专门迭代应用模型:BSP,迭代式MapReduce,bulk incremental

空间做权衡


Network bandwidth
Memory bandwidth
Fine
K-V stores, databases, RAMCloud
Best for transactional workloads
Granularity of Updates
Best for batch workloads
HDFS
RDDs
Coarse
Low
High
Write Throughput

概述

Spark编程接口

使用scalar语言类DryadLINQ API
从Scala解释器 交互可用
提供弹性分布式数据集，RDDs上面的操作,变换(构建新的RDds)，作用(计算和输出结果)；控制每个RDDs分隔和持久性

举例：日志挖掘
从日志到内存加载错误信息，然后交互式搜索不同的模式

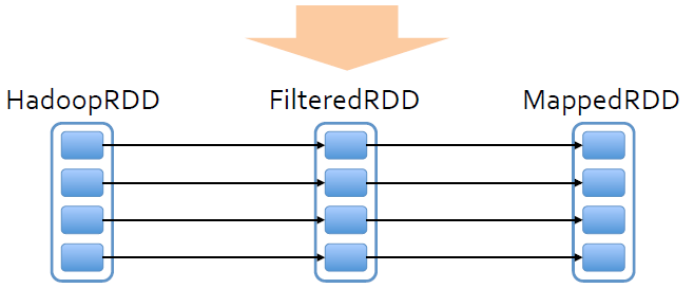
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
messages.persist()
messages.filter(_.contains(“foo”)).count
messages.filter(_.contains(“bar”)).count
```

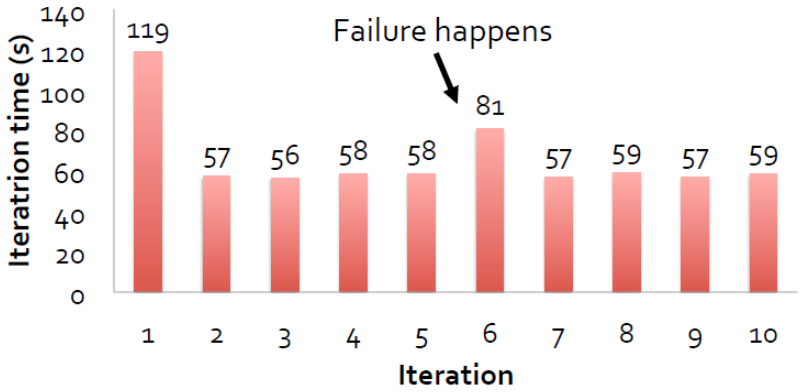
5-7s内扩展1TB数据

错误恢复

RDDs跟踪变换图，重建丢失数据

E.g.:
messages = textFile(...).filter(_.contains(“error”))
.map(_.split(‘\t’)(2))





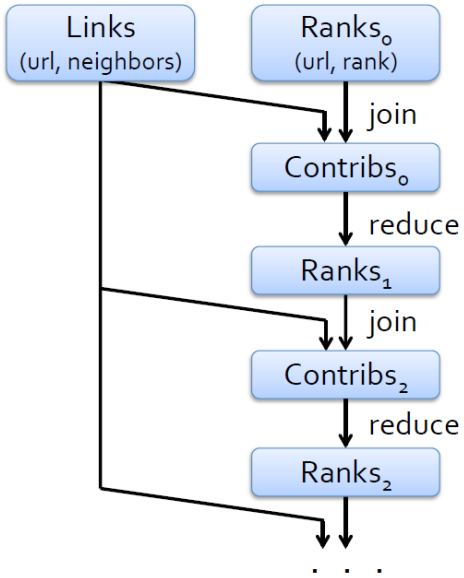
PageRank举例

1.初始化每页面等级为1
2.每次迭代更新每个页面
级别为

$$\Sigma_{i \in neighbors} rank_i / |neighbors_i|$$

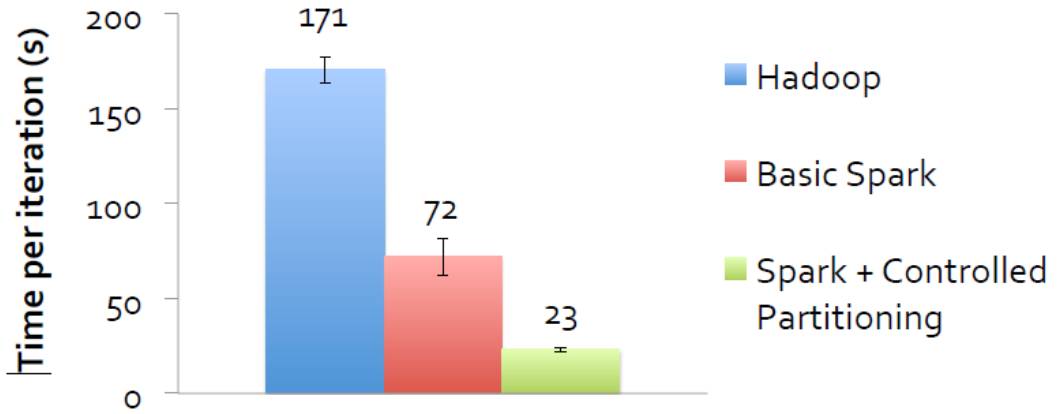
```
links = // RDD of (url, neighbors) pairs
ranks = // RDD of (url, rank) pairs

for (i <- 1 to ITERATIONS) {
  ranks = links.join(ranks).flatMap {
    (url, (links, rank)) =>
      links.map(dest => (dest, rank/links.size))
  }.reduceByKey(_ + _)
}
```

置换优化



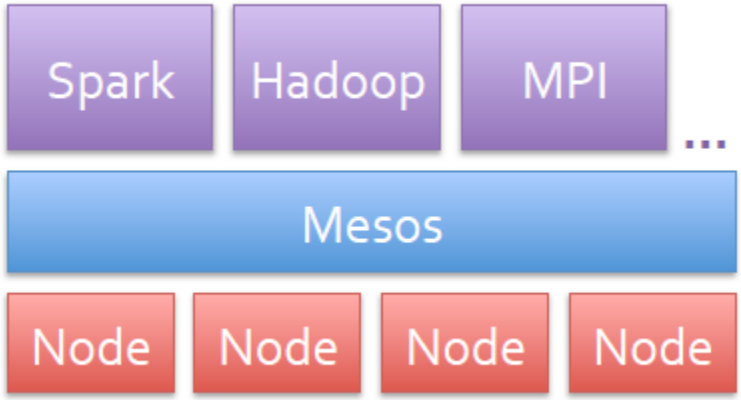
链接和排名交替joined
可以共用分区来避免Shuffles
也可以使用应用语言，比如:在域名上hash
links = links.partitionBy(new URLPartitioner())

pageRank性能



实现

运行在Mesos共享集群 w/ Hadoop
可以从任何Hadoop输入源读取
对于Scala语言或者编译器没有改变
反射和字节码分析跳过代码



编程模型在Spark上面实现
RDDs可以解释许多已有的并行模型
mapReduce,DryadLINQ
Pregel 图计算 迭代式MapReduce
SQL
使得应用可以高效混合这些模型

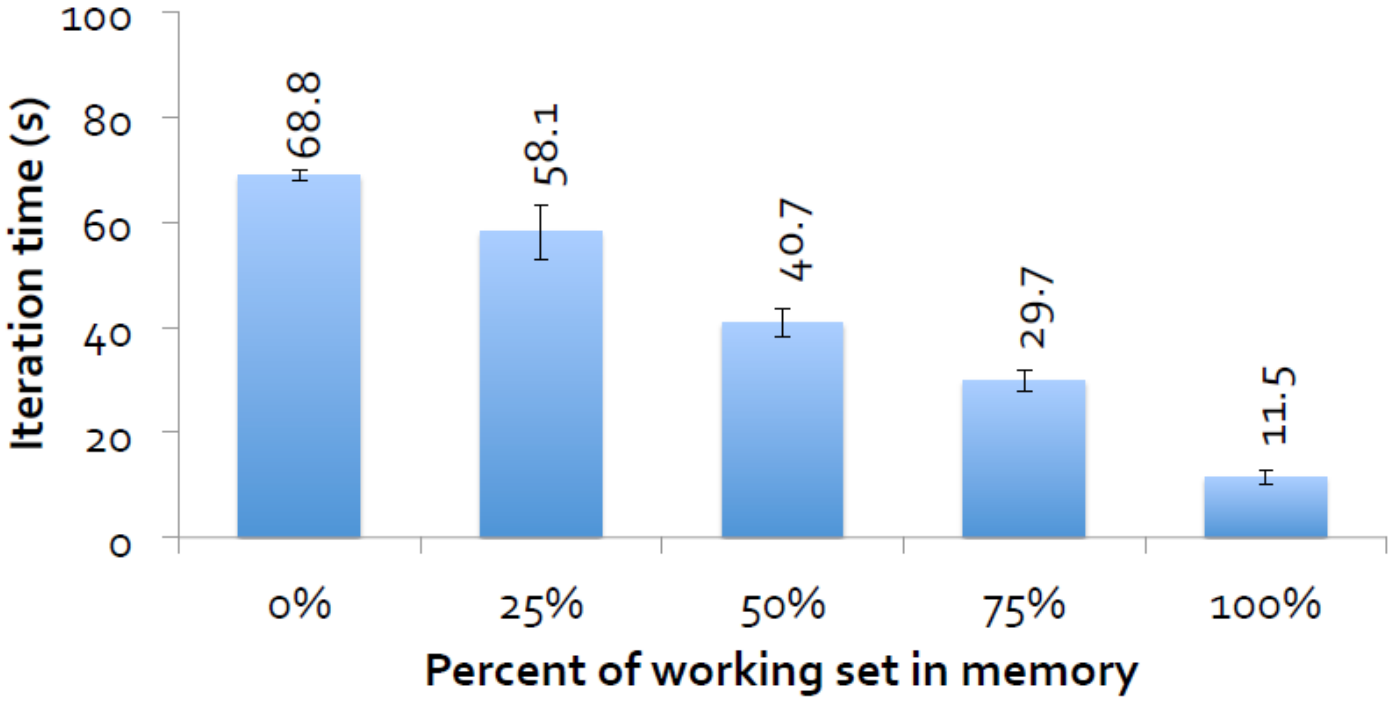
总结

RDDs给广泛应用提供一个简单高效的编程模型

对于低开销恢复，利用许多并行算法粗粒度本质

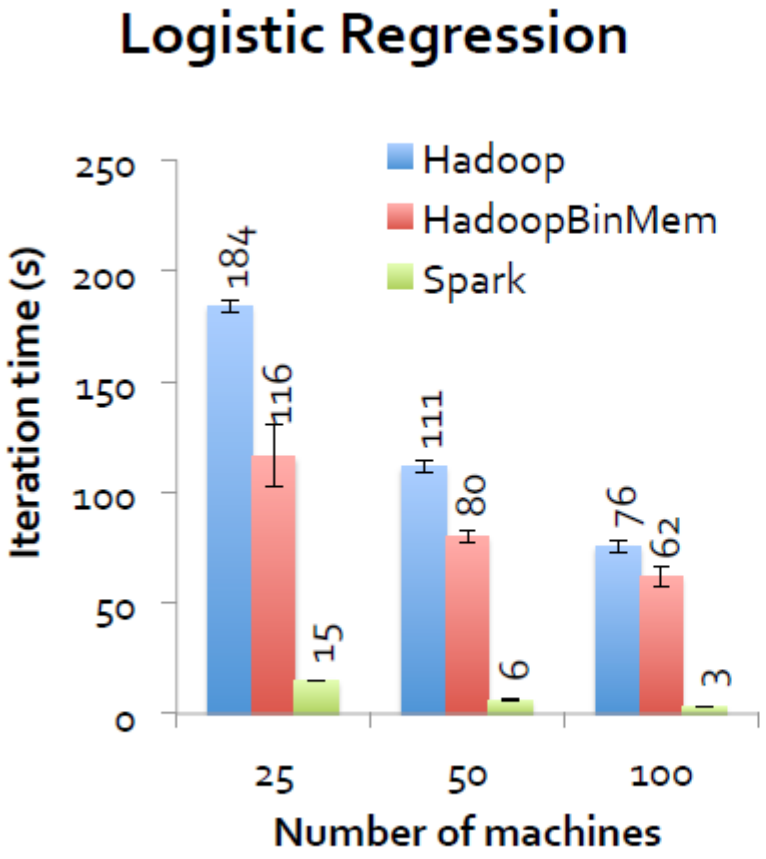
性能评估

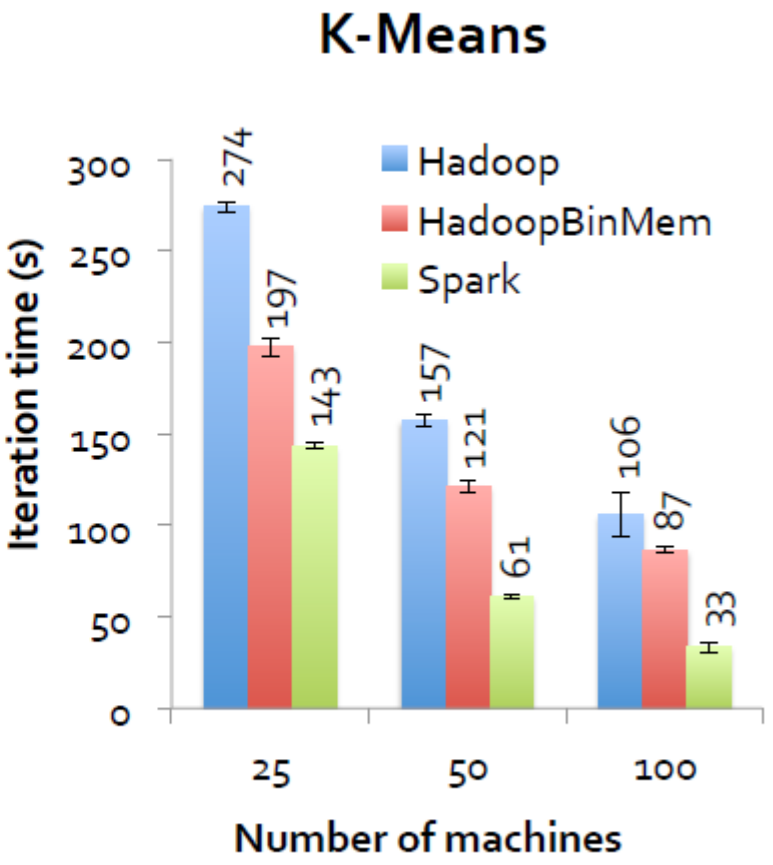
不足够内存


100
80
60
40
20
0
Iteration time (s)
68.8
58.1
40.7
29.7
11.5
0%
25%
50%
75%
100%
Percent of working set in memory

性能评估

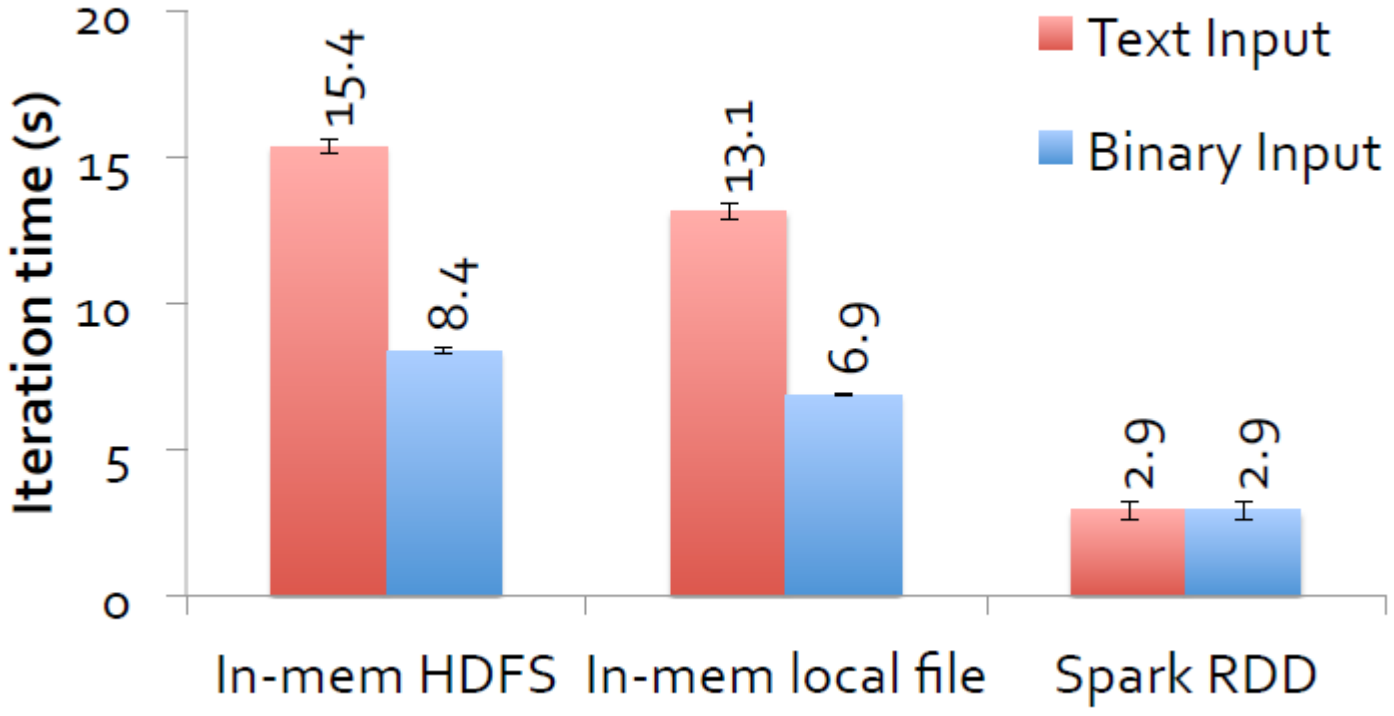
可扩展性


Logistic Regression
Hadoop
HadoopBinMem
Spark
Iteration time (s)
250
200
150
100
50
0
184
116
15
111
80
6
76
62
3
25
50
100
Number of machines


K-Means
Hadoop
HadoopBinMem
Spark
Iteration time (s)
300
250
200
150
100
50
0
274
197
143
157
121
61
106
87
33
25
50
100
Number of machines

性能评估

打破加速


Iteration time (s)
20
15
10
5
0
15.4
8.4
13.1
6.9
2.9
2.9
Text Input
Binary Input
In-mem HDFS
In-mem local file
Spark RDD

Spark 操作

Transformations (define a new RDD)	map filter sample groupByKey reduceByKey sortByKey	flatMap union join cogroup cross mapValues
Actions (return a result to driver program)	collect reduce count save lookupKey	

谢谢