TRANSWARP
星 环 科 技

# 分布式计算框架

星环科技
www.transwarp.io
2017年7月5日

TRANSWARP
星 环 科 技

# MapReduce

# 大纲



- 概述
- 基本概念
- Mapper-Reducer运行机制
- MR任务监控与诊断

# 概述



- 起源
  - Google MapReduce论文
  - 初衷是解决搜索引擎中大规模网页数据的并行化处理
- MR是什么
  - 分布式计算框架
  - 分而治之的思想
- MR特点
  - 自动化并行和分布式计算
  - 计算能力随着节点数增加近似线性递增
  - 出错容忍
  - I/O调度（计算跟着数据走）
  - 状态监控

# 引例--WordCount



- 词频统计
  - 有非常多的文本文件，包含非常多的单词，GB-TB级
  - 统计每个单词出现的次数
- 解决方法
  - 写个小程序，按顺序统计所有单词词频
  - 写个多线程程序，并发遍历所有文件
  - 把方法1的程序部署到N台机器上，把文件分成N份，每台机器统计一份文件，最后再统计
  - 使用MapReduce框架统计
    - 通过map函数和reduce函数统计

# 大纲



- 概念
- 基本概念
- Mapper-Reducer运行机制
- MR任务监控与诊断

# 基本概念--job和task



- 作业job是客户端要求执行的一个工作单元
  - 输入数据、MapReduce程序、配置信息
- 任务task是MapReduce将作业拆成的小单元
  - map任务和reduce任务
- Job Tracker节点（master）
  - 调度task在Task Tracker上运行，协调所有作业运行
  - 如果一个task失败，Job Tracker指定一个Task Tracker重新开始
- Task Tracker节点（worker）
  - 执行任务，发送进度报告

# 基本概念--分片split



- 分片的定义
  - MapReduce把输入的数据划分成等长的小数据块，称为输入分片input split，简称分片
- 分片大小
  - 分片越小，负载越平衡
    - 异构时根据计算机性能分配任务个数
    - 失败重启更加平衡
  - 分片越小，框架开销越大
    - 每个分片一个map任务
    - 管理分配的总时间和构建map任务时间变大
  - 默认HDFS块大小，128MB
- 计算数据本地化
  - 在本地存有HDFS数据的节点上运行map任务

# 基本概念--Map&Reduce



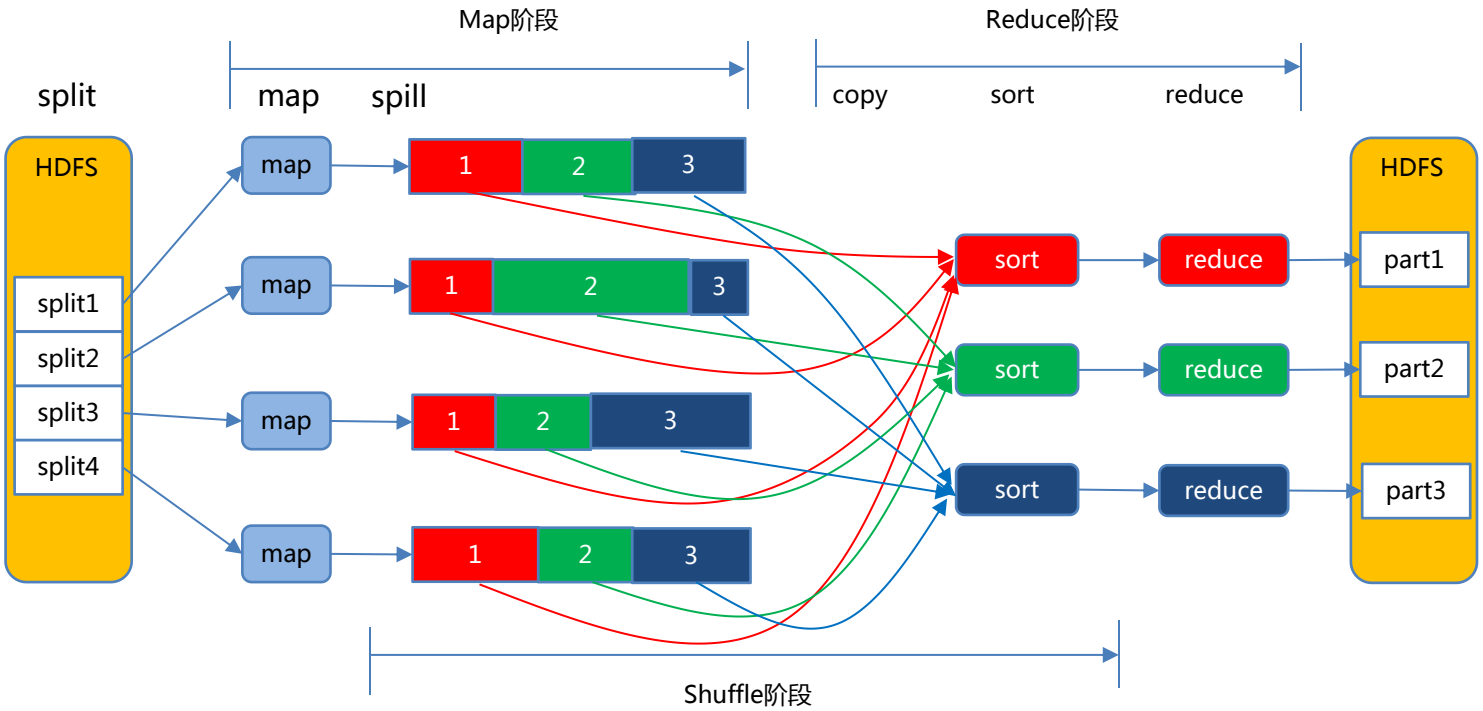
- Map阶段
  - 将split数据以键值对形式输出
- Reduce阶段
  - 按照逻辑对每个map相应partition的键值做合并处理
- Shuffle阶段
  - 将数据从Map传到Reduce的过程

# 大纲



- 概念
- 基本概念
- Mapper-Reducer运行机制
- MR任务监控与诊断

# MapReduce过程



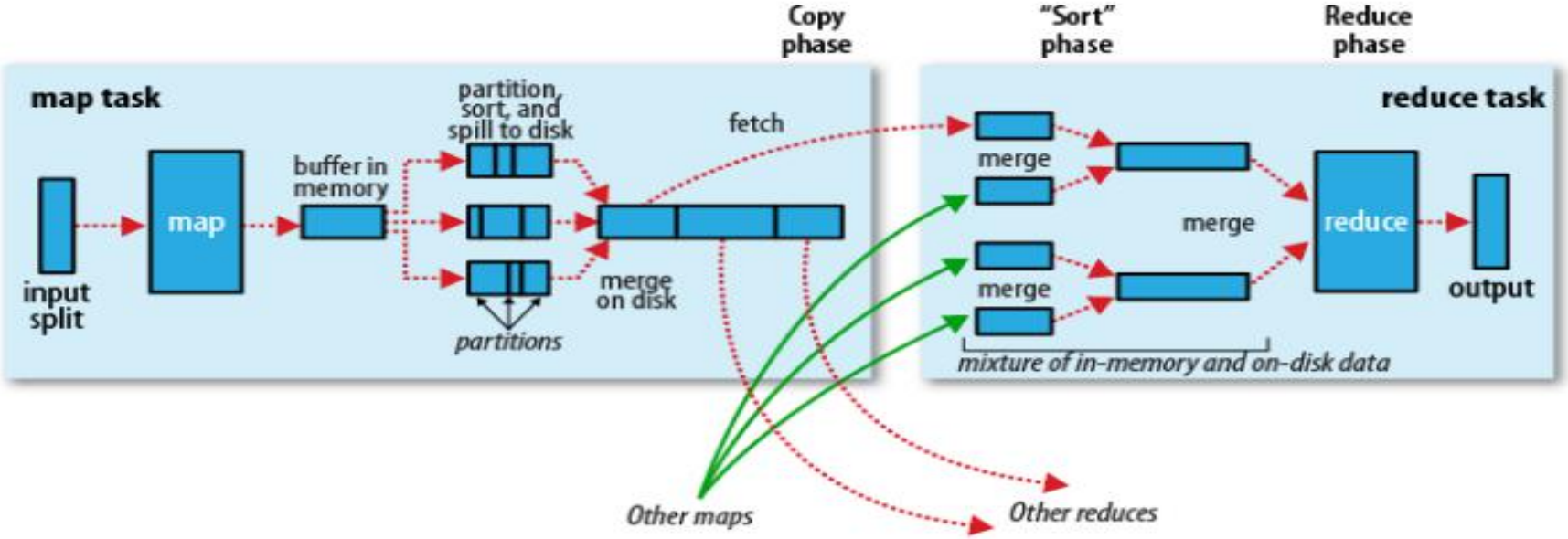
# MapReduce过程



- 一个split（分片）起一个map任务
  - 通常为HDFS的一个block大小
  - 计算跟着数据走
- map输出的记录由一个个键值对构成，分区排序写入本地磁盘
- map数据到reduce过程（shuffle）中，同一个key的所有value会发到同一个reduce
- reduce接受到的每个key包含一组value，将value合并然后写到hdfs上

# MapReduce详细过程





# MapReduce详细过程



- 一个split（切片）起一个map任务
- map输出时会先将输出中间结果写入到buffer中
- 在buffer中对数据进行partition（分区，partition数为reduce数）和基于key的sort（排序），达到阈值后spill到本地磁盘
- 在map任务结束之前，会对输出的多个文件进行merge（合并），合并成一个文件
- 每个reduce任务会从多个map输出中拷贝自己的partition
- reduce也会将数据先放到buffer中，达到阈值会写到磁盘
- 当数据该reduce的map输出全部拷贝完成，合并多个文件成一个文件，并保持基于key的有序
- 最后，执行reduce阶段，运行我们实现的reduce中化简逻辑，最终将结果直接输出到HDFS中

# 大纲



- 概念
- 基本概念
- Mapper-Reducer运行机制
- MR任务监控与诊断

# MR任务监控和诊断



- 任务提交

```
hadoop  jar  {jarFile}  [mainClass]  args

jarFIle: MapReduce运行程序的jar包
mainClass: jar包中main函数的类名
args: 程序调用需要的参数，比如输入输出路径
```

- 任务查看

```
sudo –u yarn application -list
```

- 任务终止

```
sudo –u yarn application –kill {application_id}
```

# MR任务监控和诊断

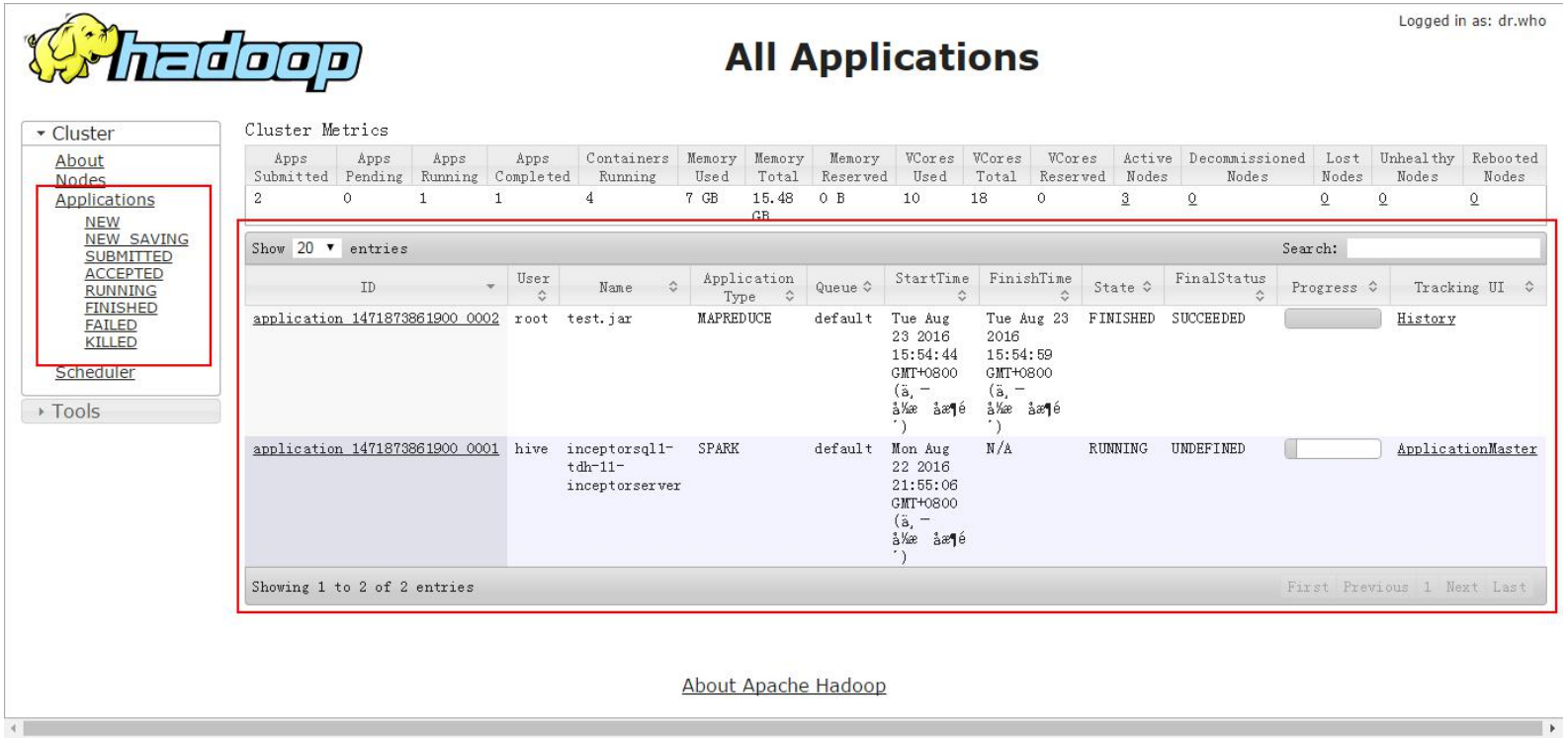


```
t3126poc4:~ # hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar pi 10 10
Number of Maps  = 10
Samples per Map = 10
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Wrote input for Map #5
Wrote input for Map #6
Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
Starting Job
2016-05-10 14:09:58,250 INFO client.RMProxy: Connecting to ResourceManager at t3126poc5/172.16.2.85:8032
2016-05-10 14:09:58,834 INFO input.FileInputFormat: Total input paths to process : 10
2016-05-10 14:09:58,915 INFO mapreduce.JobSubmitter: number of splits:10
2016-05-10 14:09:59,188 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1462786145119_0002
2016-05-10 14:09:59,453 INFO impl.YarnClientImpl: Submitted application application_1462786145119_0002
2016-05-10 14:09:59,498 INFO mapreduce.Job: The url to track the job: http://t3126poc5:8088/proxy/application_1462786145119_0002/
2016-05-10 14:09:59,499 INFO mapreduce.Job: Running job: job_1462786145119_0002
2016-05-10 14:10:05,641 INFO mapreduce.Job: Job job_1462786145119_0002 running in uber mode : false
2016-05-10 14:10:05,644 INFO mapreduce.Job:  map 0% reduce 0%
```

- application id: application_1462786145119_0002
- web监控地址: http://{AM_IP}:8088/proxy/{applicationId}/

# MR任务监控和诊断

- web监控地址: http://{AM_IP}:8088/proxy/{applicationId}/



# MR任务监控和诊断



参数yarn.nodemanager.log-dirs.
配置了保存MapReduce运行日志信息的目录
默认值为/mnt/disk*/hadoop/yarn/
根据运行出错信息，可以到指定节点下分析日志

```
t3126poc5:~ # ls /mnt/disk2/hadoop/yarn/logs/application_1462783245088_0002/container_1462783245088_0002_01_000002/
stderr  stdout  syslog
```

TRANSWARP
星 环 科 技

# Spark

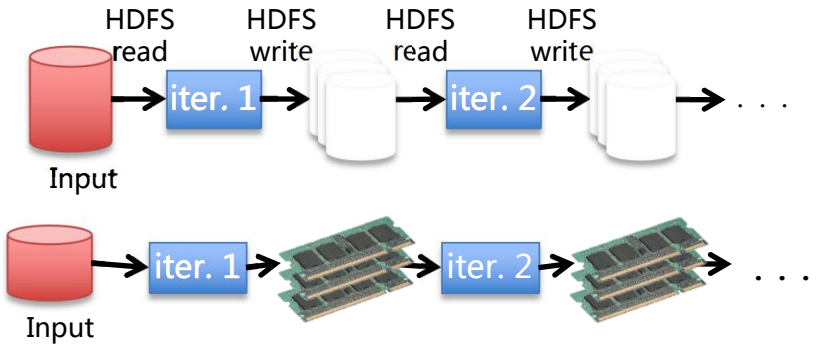
# 大纲



- 概述
- Spark on YARN
- 基本概念
- Spark任务监控

# 概述

- 由Berkeley实验室开发
- 一种新的基于MapReduce模型的分布式计算引擎
- 解决MapReduce实现的一些弱点
  - 难以支持复杂应用场景，如机器学习、流式计算、图计算等
  - 迭代式计算的效率低下



# 优势



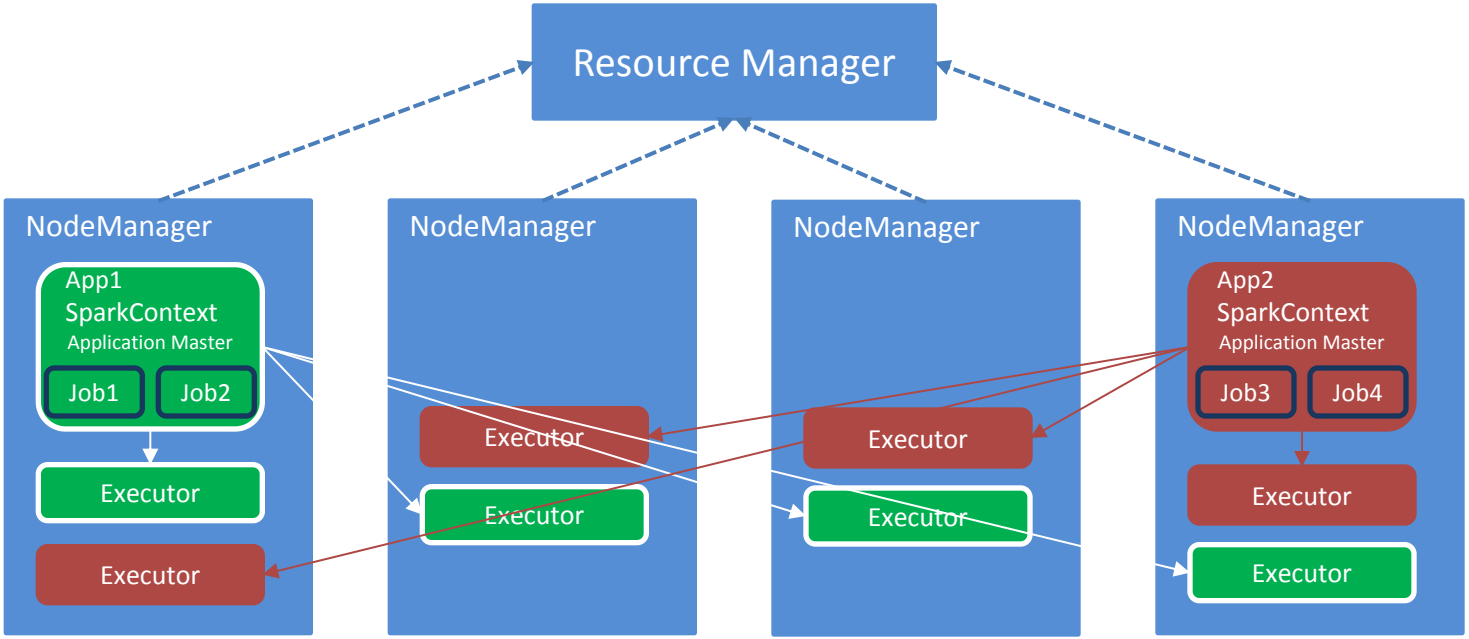
- Spark优势
  - 基于内存计算（RDD）
  - 基于DAG优化任务流程（延迟计算）
  - 易于部署，更低的框架开销
  - 丰富的API支持，如Scala、Java、Python
- Spark适用场景
  - 机器学习和图应用中常用的迭代算法（每一步对数据执行相似的函数）
  - 交互式数据挖掘工具（用户反复查询一个数据子集）

# 大纲



- 概述
- Spark on YARN
- 基本概念
- Spark任务监控

# Spark on YARN



# Spark on YARN



- SparkContext
  - Spark应用的主入口
  - 在YARN中为Application Master角色
  - 负者该应用的任务调度
- Executor
  - 应用程序的任务容器，任务在Executor内部运行

# 大纲



- 概述
- Spark on YARN
- 基本概念
- Spark任务监控

# 基本概念



- RDD：弹性分布式数据集
- DAG：有向无环图

# RDD



- RDD：弹性分布式数据集
  - 支持工作集的应用，允许缓存工作集，进行重用
  - 对应用透明，应用只需要通过RDD的操作实现业务
  - 只能基于在稳定物理存储中的数据集和其他已有的RDD上执行确定性操作，这些操作称为转换transformation
    - map、filter、groupby、join等
  - 不需要物化，含有从其他RDD衍生出本RDD的相关信息（lineage）
  - 发生错误时可通过血缘关系自动重构

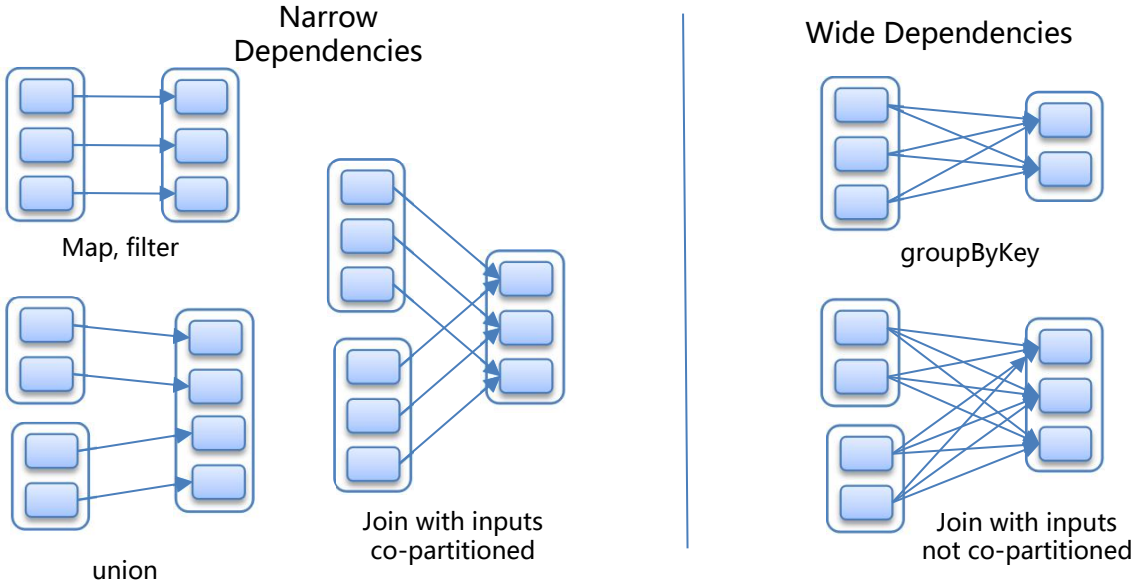
# RDD操作



- Transformation转换
  - 转换时，只创建依赖关系Lineage，不实际执行
  - 延迟计算，只有第一次action时RDD才会计算，可以在构建RDD时，通过管道方式传输多个转换
  - map、filter、groupBy、join等
- Action动作
  - 真正触发RDD计算
  - 向应用结果返回结果或将结果持久化到底层存储
  - count、collect、save、cache等

# RDD依赖

- 窄依赖narrow dependencies
  - 子RDD的每个分区依赖常数个父分区
- 宽依赖wide dependencies
  - 子RDD的每个分区依赖所有父RDD



# RDD依赖



- 窄依赖允许在一个集群节点上以流水线的方式计算所有父分区
- 宽依赖需要先计算好所有父分区数据
- 窄依赖能够更有效的进行失效节点恢复，只需要重新计算丢失RDD分区的父分区，不同节点之间并行计算
- 宽依赖中，单个节点失效可能导致这个RDD的所有祖先丢失部分分区，需要整体重新计算
- 当数据已经按规则分片了，就不会再进行shuffle操作了

# RDD容错



- Lineage恢复故障
  - Lineage链过长，恢复很耗时
  - 宽依赖，需要重新计算所有父分区
- 检查点
  - Lineage链过长和宽依赖的分区采用检查点机制
  - 将RDD物化到磁盘

# RDD示例

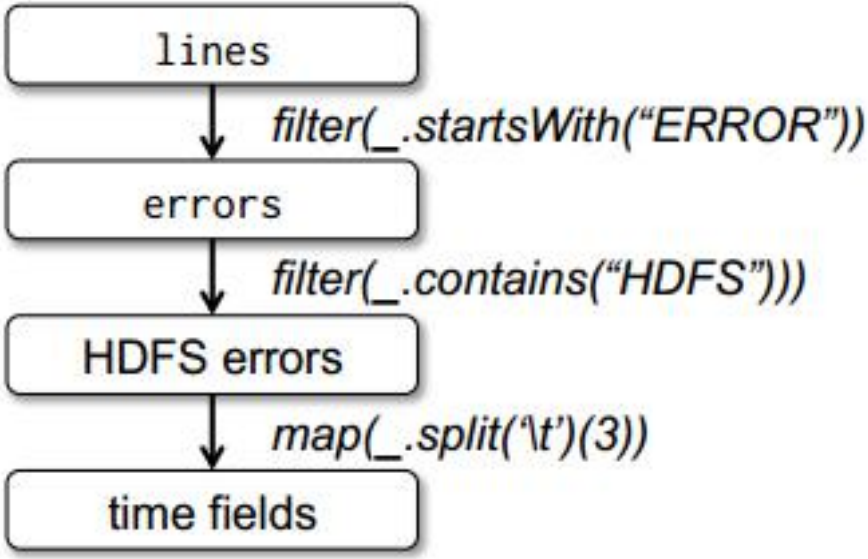


```
val   lines   =
spark.textFile(“hdfs://...”)

val   errors   =
lines.filter(_.startswith(“ERROR”))

val   hdfsErrors   =
errors.filter(_.contains(“HDFS”))

val   time   = hdfsErrors
hdfsErrors.map(_.split(“\t”)(3))
```

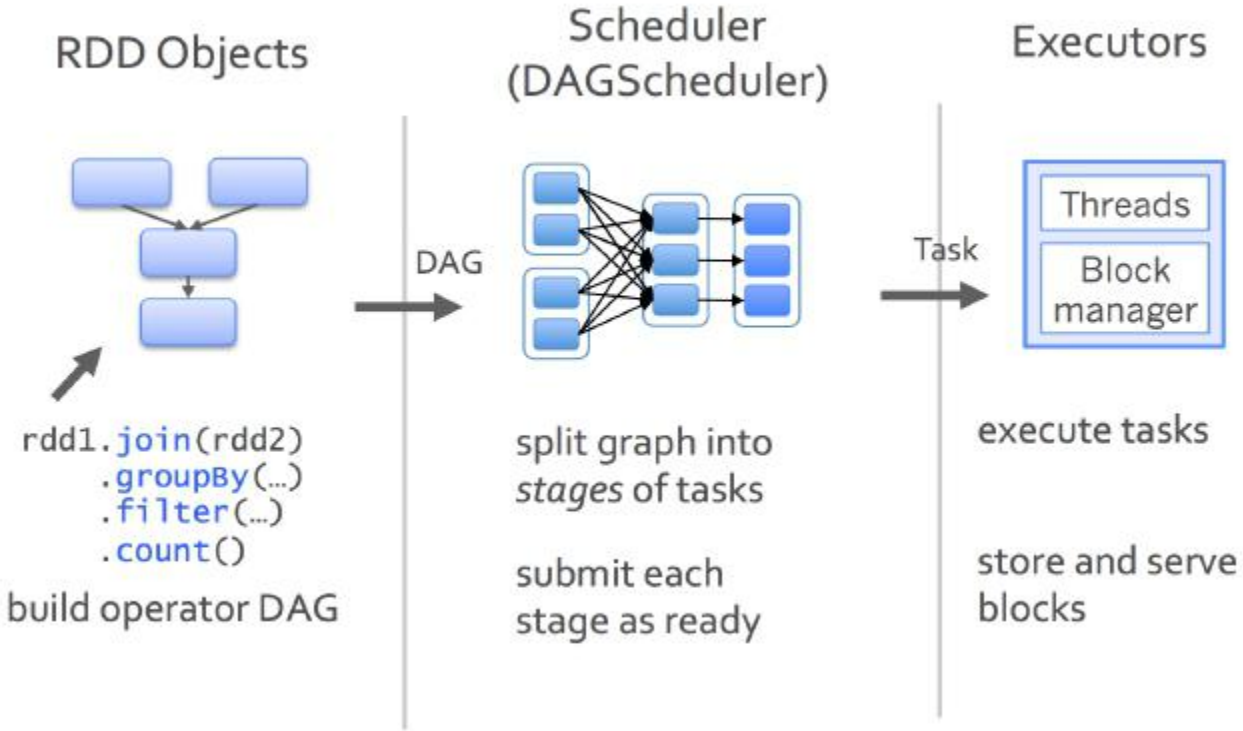


# DAG



- 有向无环图
  - 一个有向图无法从任意顶点出发经过若干条边回到该点，则这个图是一个有向无环图
  - 受制于某些任务必须比另一些任务较早执行的限制，必须排序为一个队列的任务集合，可以由一个DAG图呈现
    - 每个顶点表示一个任务
    - 每条边表示一种限制约束
  - Spark任务可以用DAG表示

# DAG Scheduler

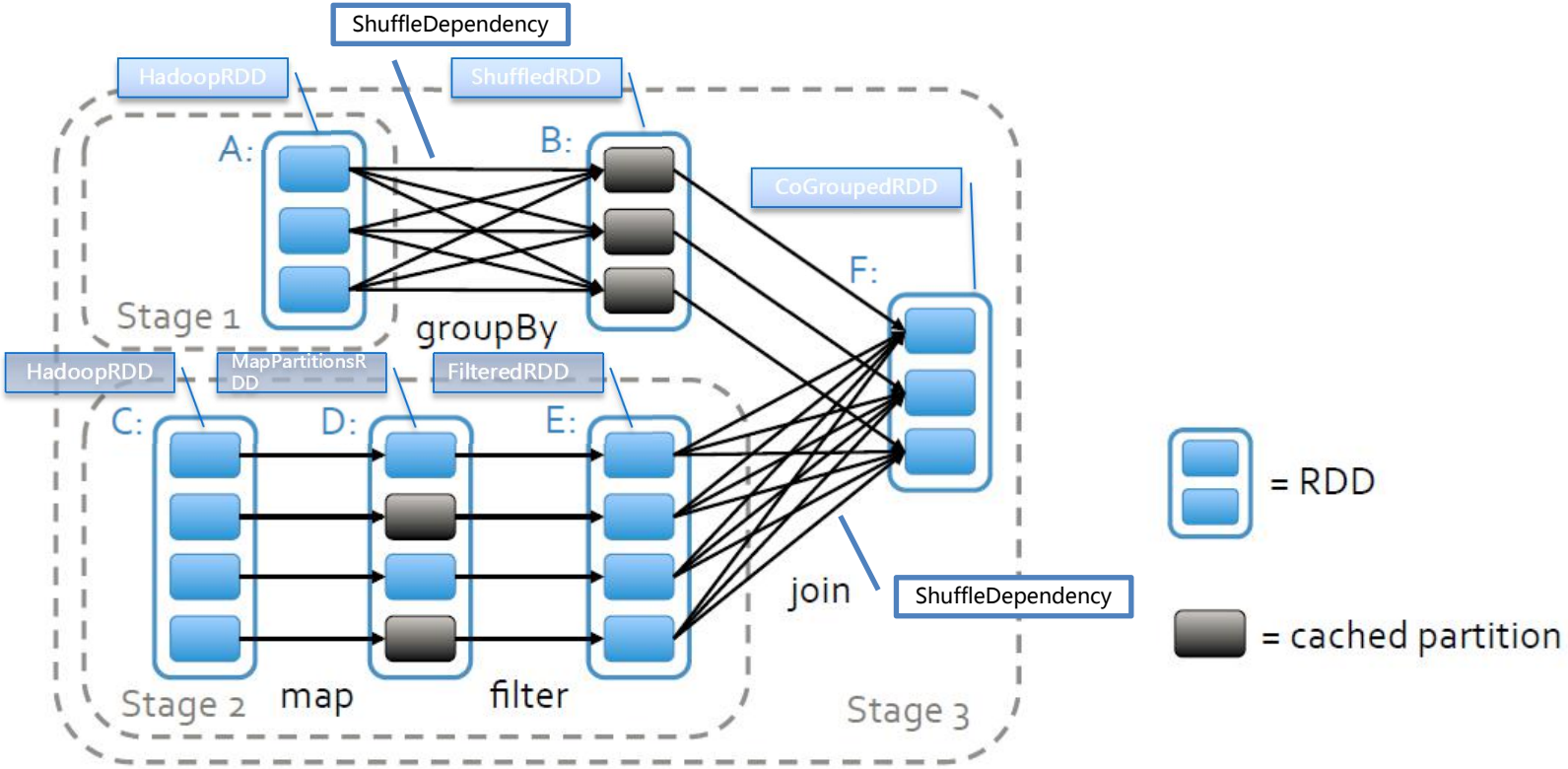


# DAG Scheduler



- 输入：RDD操作依赖图
- 输出：RDD分片的操作依赖，并划分操作Stage
- 主要操作
  - 将任务组织成操作Stage，有Shuffle操作就会生成一个Stage
  - 将Stage中包含的Task交给底层TaskScheduler来调度
  - 如果一个Stage的输出丢失或失败，可以重新调度这个Stage来运行

# Spark任务调度器




From：Matei Zaharia撰写的论文An Architecture for Fast and General Data Processing on Large Clusters

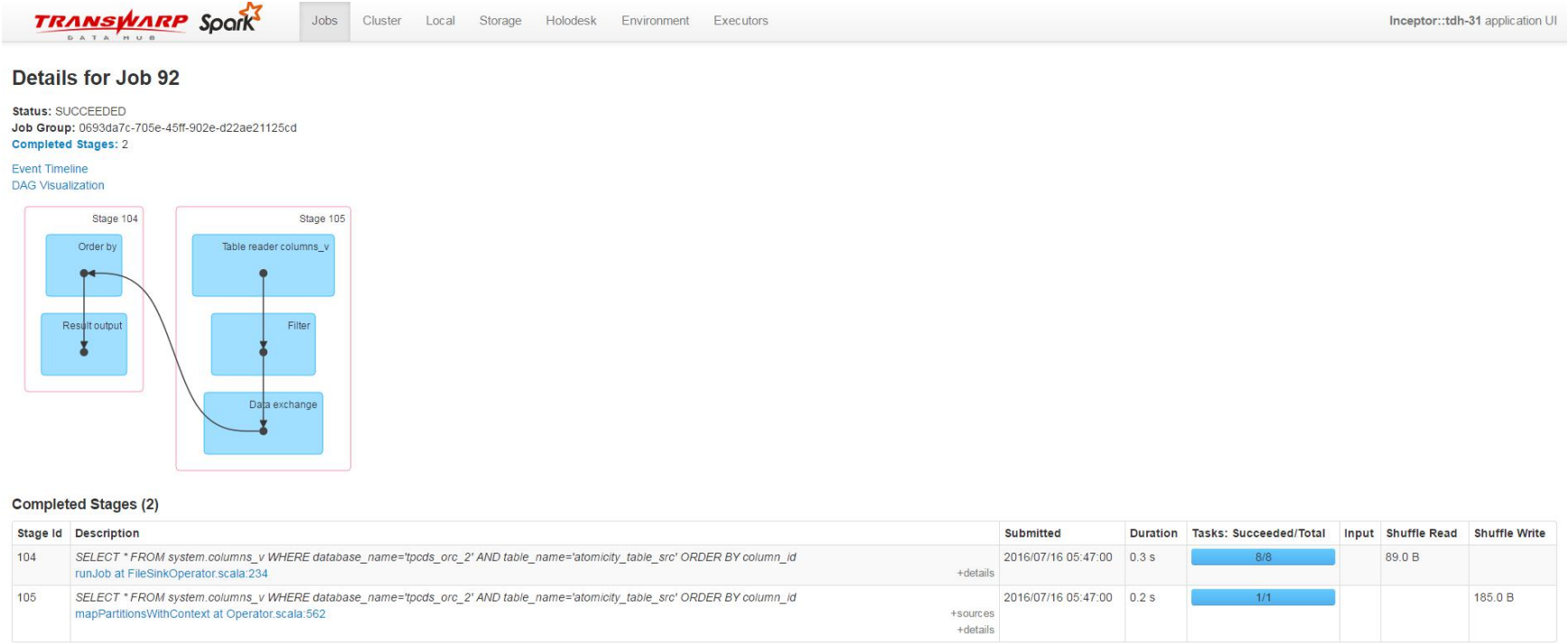
# 大纲



- 概述
- Spark on YARN
- 基本概念
- Spark任务监控

# Spark任务监控

- web监控地址: http://{INCEPTOR_SERVER_IP}:4040



TRANSWARP
星 环 科 技

# Q&A