MapReduce

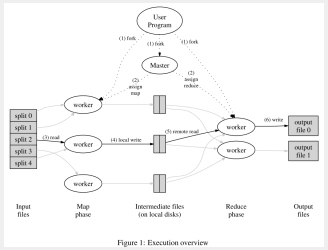
- 编程模型（受Lisp启发）
  - Map散列
  - Reduce规约
  - key联系
  - 应用例子
    - Word Count
    - Distributed Grep
    - Count of URL Access Frequency
    - Reverse Web-Link Graph
    - Term-Vector per Host
    - Inverted Index
    - Distributed Sort
- 执行流程
  - 

    Figure 1: Execution overview
  - MapReduce执行整体架构
- 容错
  - Worker Failure
    - 检查：ping，心跳
    - 恢复：重做map任务，然后通知所有未读取第一次map结果的reduce任务，读取map结果
  - Master Failure
    - 检查点
    - 重启
  - Semantics in the Presence of Failures(幂等操作)
    - map：master接受第一个消息，忽略重复消息
    - reduce：原子重命名临时文件到最终文件，保证最终文件只是来源一个reduce task
- 本地化（数据分布）
  - map task 工作在数据位置所在节点，减少网络传输
- 任务粒度（负载均衡）
  - 理想：M、R>workers
  - master
    - O(M+R) 调度决策
    - O(M*R)状态
  - 实际生产：M大（每个分片16-64MB），R小（R为worker的几倍）
- 长尾任务
  - 机器节点，受限硬件环境，软件bug，执行map，reduce的task缓慢
  - 缓解方法：备份任务，当存在自己的task完成后，备份正在执行的进行中的任务，然后无论主任务还是备份任务执行完毕，都标记该任务做完。
- 数据倾斜
  - Combiner，提前在map task完成后做部分合并，减少网络传输
  - 与reduce区别：输出结果不写入最终文件而是中间文件发往某个reduce
- 目标
  - 计算并行化
  - 容错
  - 数据分布
  - 负载均衡
- 环境
  - 大量入门商业PC（上千）
  - 低速网络百兆千兆（04年）
  - 机器failures常发生
  - 廉价的IDE磁盘
  - 批任务
- 优势
  - Easy to program
  - Scales well
  - Easily expressible
- 缺点
  - Not the most efficient or flexible
  - too many write disk