Hadoop YARN
基本架构和发展趋势

董西成
email ： dongxicheng@yahoo.com
微信号：hadoop-123 [口号：做微信上最好的hadoop技术传播者]

主要内容

1. Hadoop YARN产生背景
2. Hadoop YARN基本架构
3. 运行在YARN上的计算框架
4. YARN发展趋势

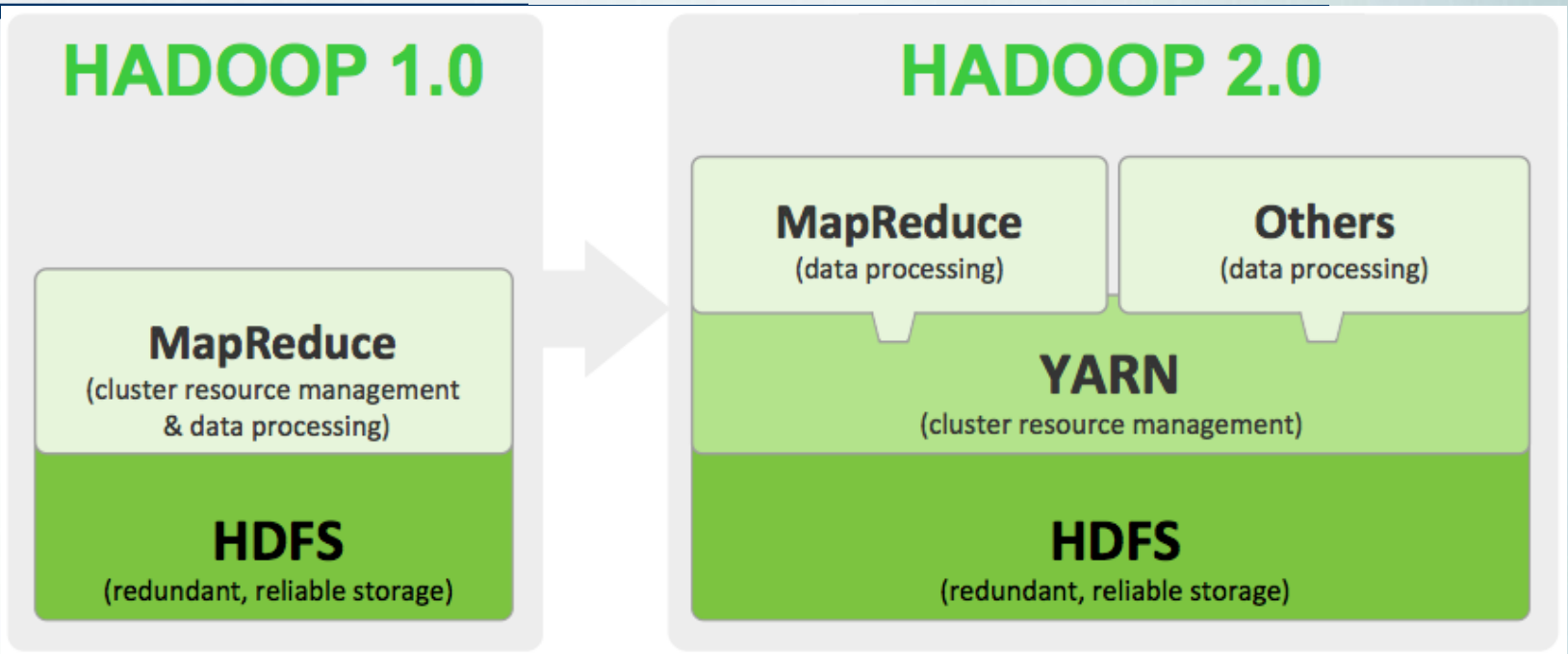
主要内容

1. Hadoop YARN产生背景
2. Hadoop YARN基本架构
3. 运行在YARN上的计算框架
4. YARN发展趋势

Hadoop YARN产生背景

- 直接源于MRv1在几个方面的缺陷
  - ✓ 扩展性受限
  - ✓ 单点故障
  - ✓ 难以支持MR之外的计算
- 多计算框架各自为战，数据共享困难
  - ✓ MR：离线计算框架
  - ✓ Storm：实时计算框架
  - ✓ Spark：内存计算框架

Hadoop 1.0和2.0

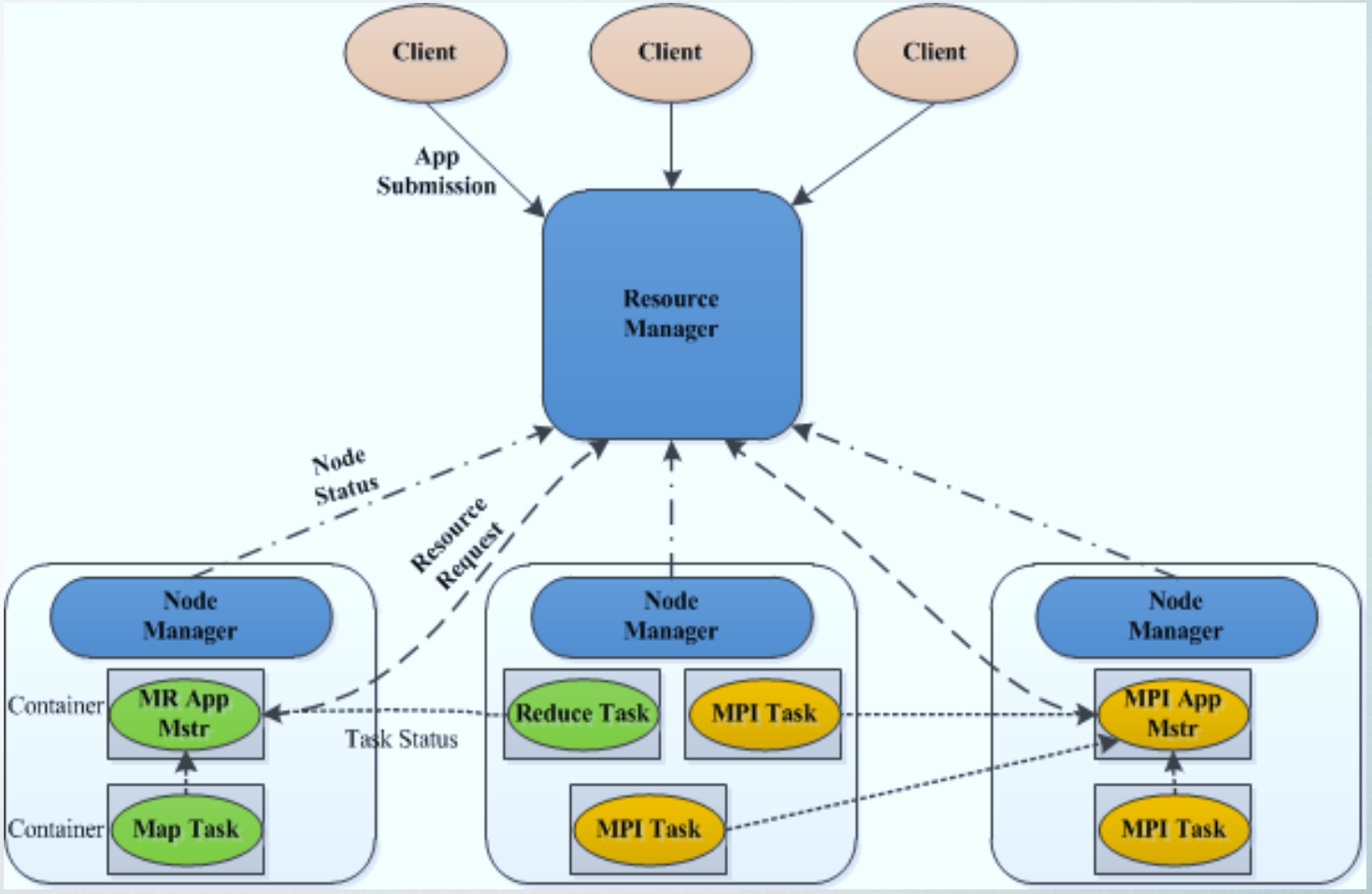


- ✓ Hadoop 2.0由HDFS、MapReduce和YARN三个分支构成；
- ✓ HDFS：NN Federation、HA；
- ✓ MapReduce：运行在YARN上的MR；
- ✓ YARN：资源管理系统

主要内容

1. Hadoop YARN产生背景
2. **Hadoop YARN基本架构**
3. 运行在YARN上的计算框架
4. YARN发展趋势

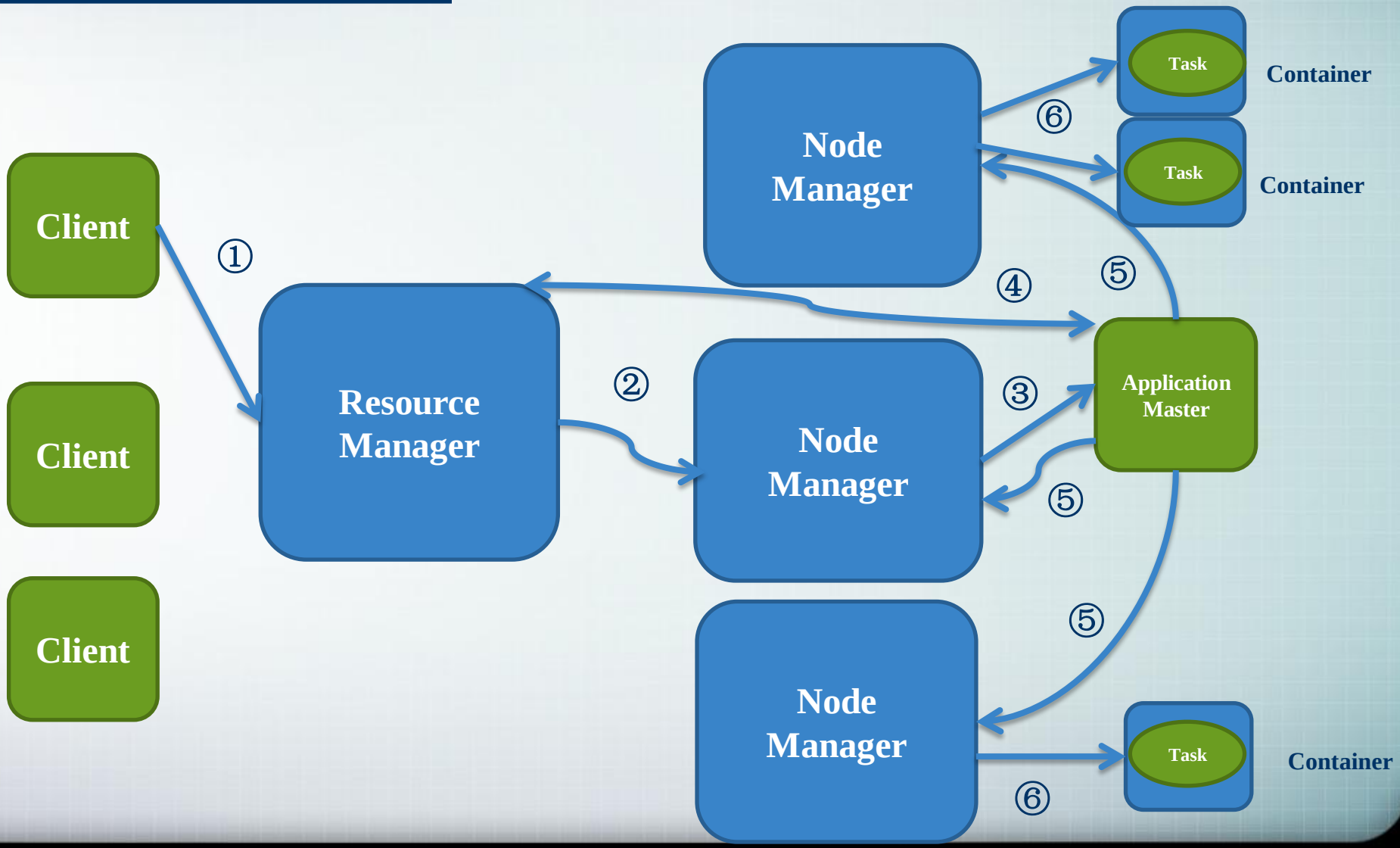
Hadoop YARN基本架构



Hadoop YARN各模块组成

- ResourceManager
  - ✓ 处理客户端请求
  - ✓ 启动/监控ApplicationMaster
  - ✓ 监控NodeManager
  - ✓ 资源分配与调度
- NodeManager
  - ✓ 单个节点上的资源管理
  - ✓ 处理来自ResourceManager的命令
  - ✓ 处理来自ApplicationMaster的命令
- ApplicationMaster
  - ✓ 数据切分
  - ✓ 为应用程序申请资源，并分配给内部任务
  - ✓ 任务监控与容错

Hadoop YARN运行流程分析

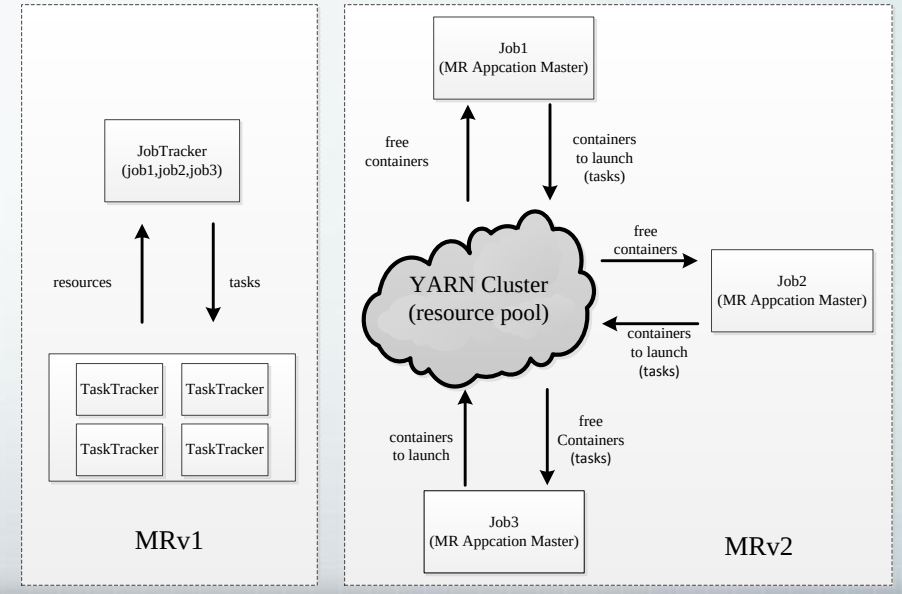


Hadoop YARN容错

- ResourceManager
  - 存在单点故障；
  - 正在基于ZooKeeper实现HA。
- NodeManager
  - 失败后，RM将失败任务告诉对应的AM；
  - AM决定如何处理失败的任务。
- ApplicationMaster
  - 失败后，由RM负责重启；
  - AM需处理内部任务的容错问题；
  - RMAppMaster会保存已经运行完成的Task，重启后无需重新运行。

Hadoop YARN调度框架

- 双层调度框架
  - ✓ RM将资源分配给AM
  - ✓ AM将资源进一步分配给各个Task
- 基于资源预留的调度策略
  - ✓ 资源不够时，会为Task预留，直到资源充足
  - ✓ 与“all or nothing”策略不同（Apache Mesos）



Hadoop YARN资源调度器

- 多类型资源调度
  - 采用**DRF**算法（论文：“Dominant Resource Fairness: Fair Allocation of Multiple Resource Types”）
  - 目前支持**CPU**和内存两种资源
- 提供多种资源调度器
  - **FIFO**
  - **Fair Scheduler**
  - **Capacity Scheduler**
- 多租户资源调度器
  - 支持资源按比例分配
  - 支持层级队列划分方式
  - 支持资源抢占

【实例】 假设系统中共有 9 CPUs 和 18 GB RAM，有两个用户（或者框架）分别运行了两种任务，需要的资源量分别为<1 CPU，4 GB> 和 <3 CPUs，1 GB>。对于用户 A，每个任务要消耗总 CPU 的 1/9（份额）和总内存的 2/9，因而 A 的主资源为内存；对于用户 B，每个任务要消耗总 CPU 的 1/3 和总内存的 1/18，因而 B 的主资源为 CPU。DRF 将最大化所有用户的主资源，具体分配过程如表 11-2 所示，最终，A 获取的资源量为<3 CPUs，12 GB>，可运行 3 个任务；而 B 获取的资源量为<6 CPUs，2GB>，可运行 2 个任务。

调度序列	User A		User B		CPU使用量	RAM使用量
	资源份额	主资源份额	资源份额	主资源份额		
User B	<0,0>	0	<3/9,1/18>	1/3	3/9	1/18
User A	<1/9,4/18>	2/9	<3/9,1/18>	1/3	4/9	5/18
User A	<2/9,8/18>	4/9	<3/9,1/18>	1/3	5/9	9/18
User B	<2/9,8/18>	4/9	<6/9,2/18>	2/3	8/9	10/18
User A	<3/9,12/18>	2/3	<6/9,2/18>	2/3	1	14/18

Hadoop YARN资源隔离方案

- 支持内存和CPU两种资源隔离
  - ✓ 内存是一种“决定生死”的资源
  - ✓ CPU是一种“影响快慢”的资源
- 内存隔离
  - ✓ 基于线程监控的方案
  - ✓ 基于Cgroups的方案
- CPU隔离
  - ✓ 默认不对CPU资源进行隔离
  - ✓ 基于Cgroups的方案

Hadoop YARN资源调度语义

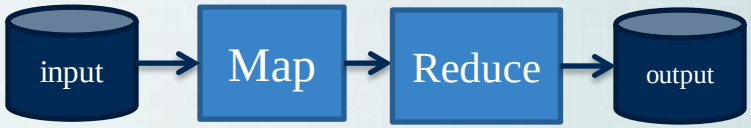
- 支持的语义
  - ✓ 请求某个特定节点/机架上的特定资源量
  - ✓ 将某些节点加入（或移除）黑名单，不再为自己分配这些节点上的资源
  - ✓ 请求归还某些资源
- 不支持的语义
  - ✓ 请求任意节点/机架上的特定资源量
  - ✓ 请求一组或几组符合某种特质的资源
  - ✓ 超细粒度资源
  - ✓ 动态调整Container资源

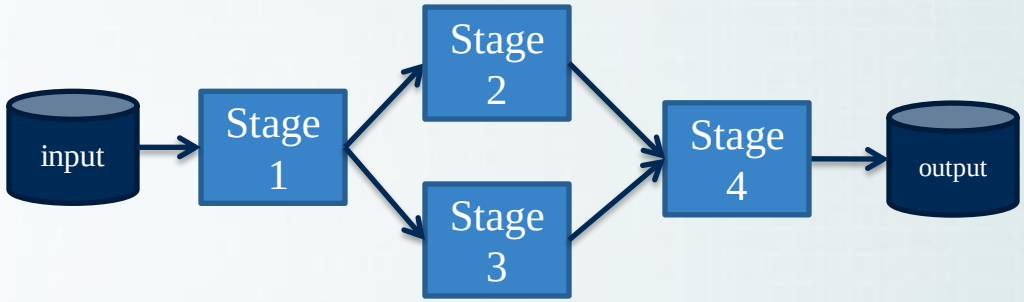
主要内容

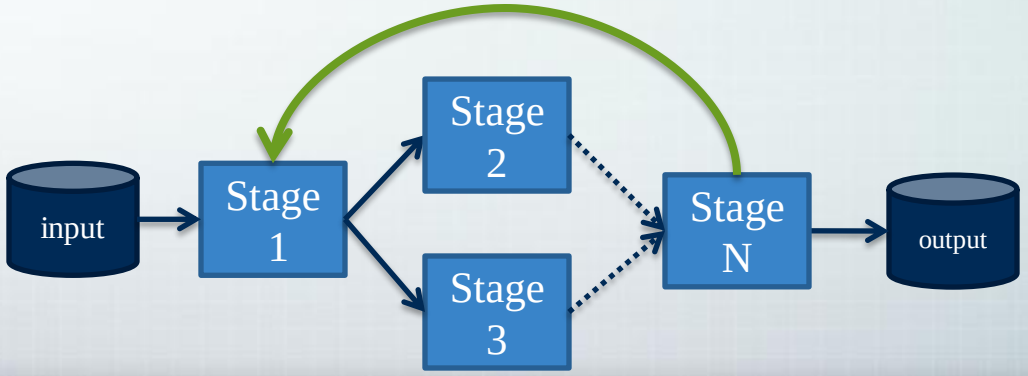
1. Hadoop YARN产生背景
2. Hadoop YARN基本架构
3. 运行在YARN上的计算框架
4. YARN发展趋势

应用程序的运行模型


input
Map
output


input
Map
Reduce
output

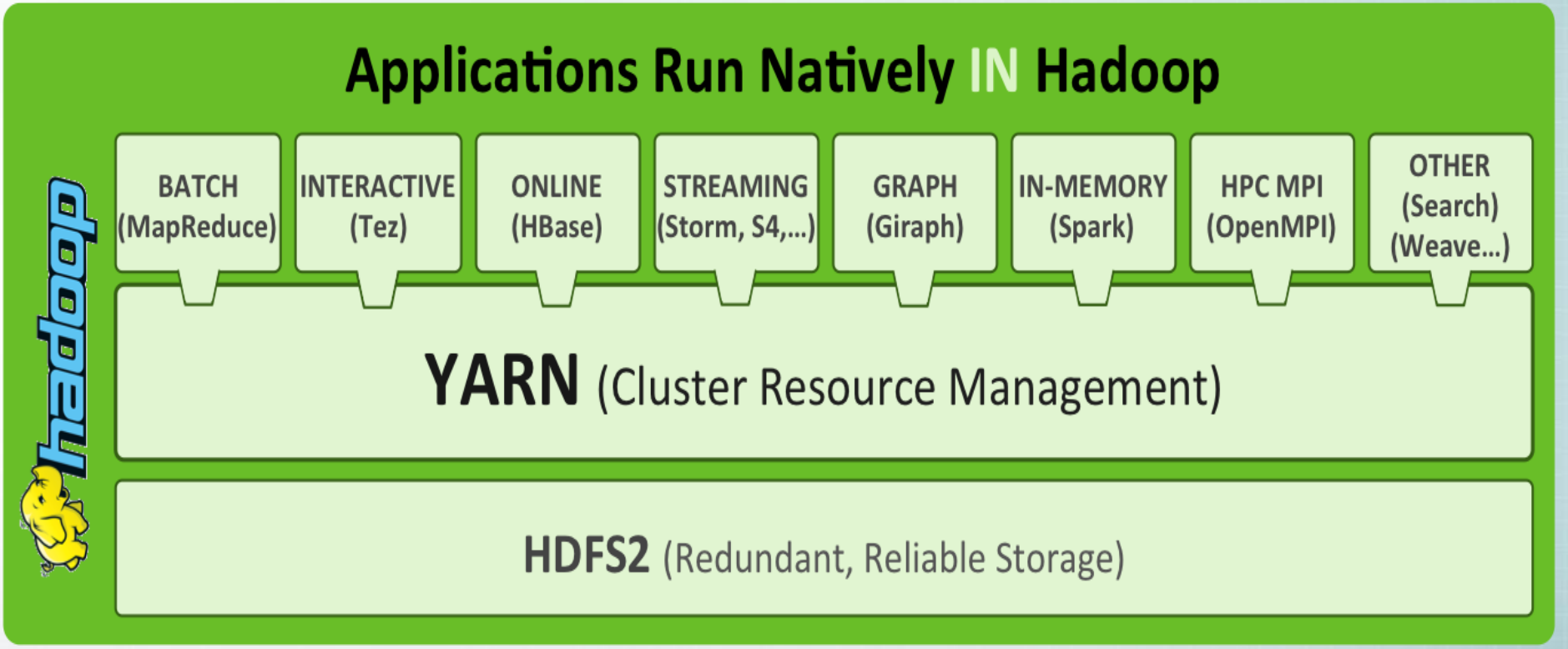

input
Stage 1
Stage 2
Stage 3
Stage 4
output


input
Stage 1
Stage 2
Stage 3
Stage N
output

YARN应用程序类型

- 长应用程序和短应用程序
- 长应用程序
  - ✓Service、HTTP Server等
- 短应用程序
  - ✓MR job、Spark Job等

以YARN为核心的生态系统



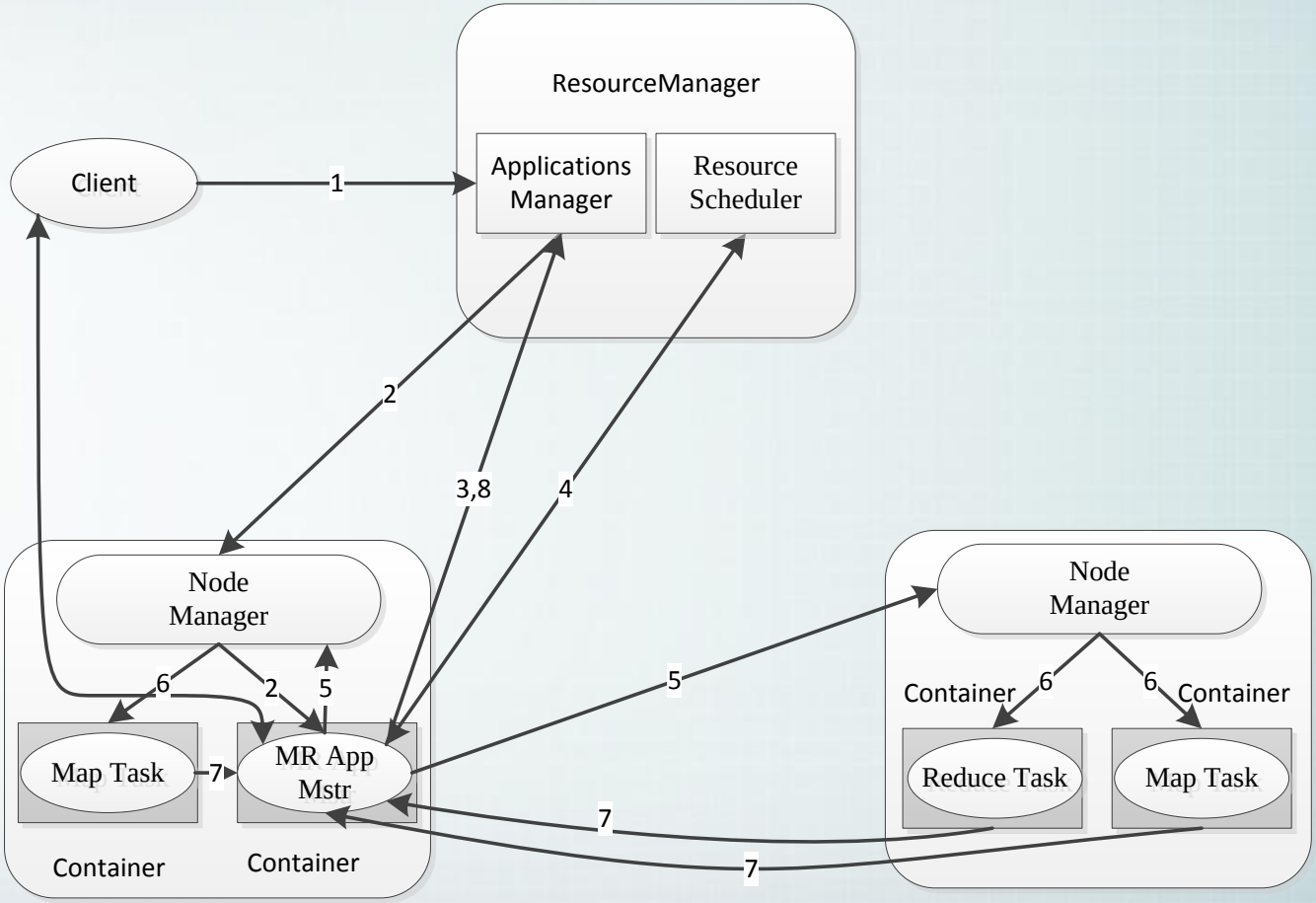
运行在YARN上的计算框架

- 离线计算框架：**MapReduce**
- **DAG计算框架：Tez**
- 流式计算框架：**Storm**
- 内存计算框架：**Spark**
- 图计算框架：**Giraph、GraphLib**

离线计算框架MapReduce

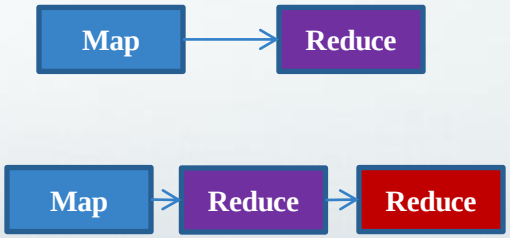
- **将计算过程分为两个阶段，Map和Reduce**
  - Map 阶段并行处理输入数据
  - Reduce阶段对Map结果进行汇总
- **Shuffle连接Map和Reduce两个阶段**
  - Map Task将数据写到本地磁盘
  - Reduce Task从每个Map Task上读取一份数据
- **仅适合离线批处理**
  - 具有很好的容错性和扩展性
  - 适合简单的批处理任务
- **缺点明显**
  - 启动开销大、过多使用磁盘导致效率低下等

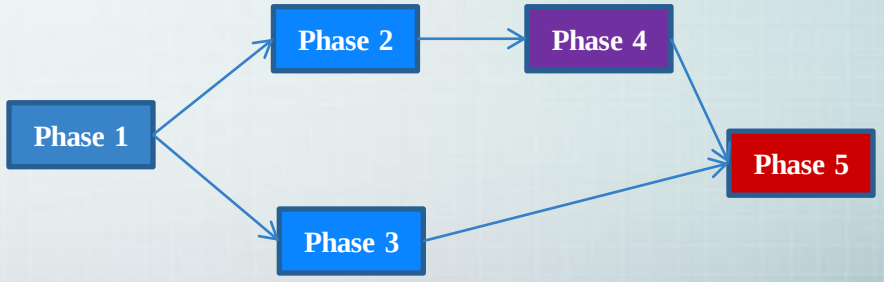
MapReduce On YARN



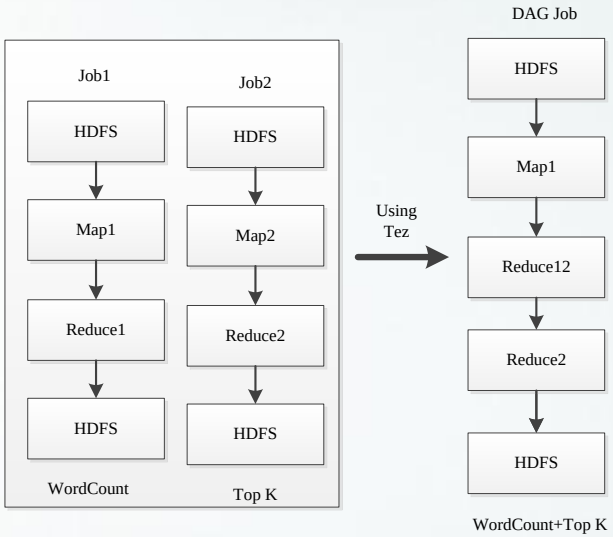
DAG计算框架Tez

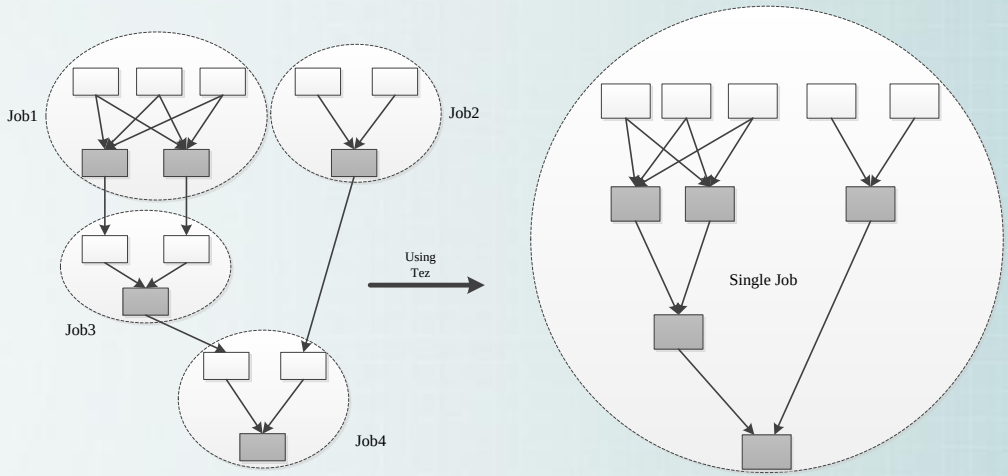
- 多个作业之间存在数据依赖关系，并形成一个依赖关系有向图（Directed Acyclic Graph），该图的计算称为“DAG计算”
- Apache Tez：基于YARN的DAG计算框架
  - ✓ 运行在YARN之上，充分利用YARN的资源管理和容错等功能；
  - ✓ 提供了丰富的数据流（dataflow）API；
  - ✓ 扩展性良好的“Input-Processor-Output”运行时模型；
  - ✓ 动态生成物理数据流关系。





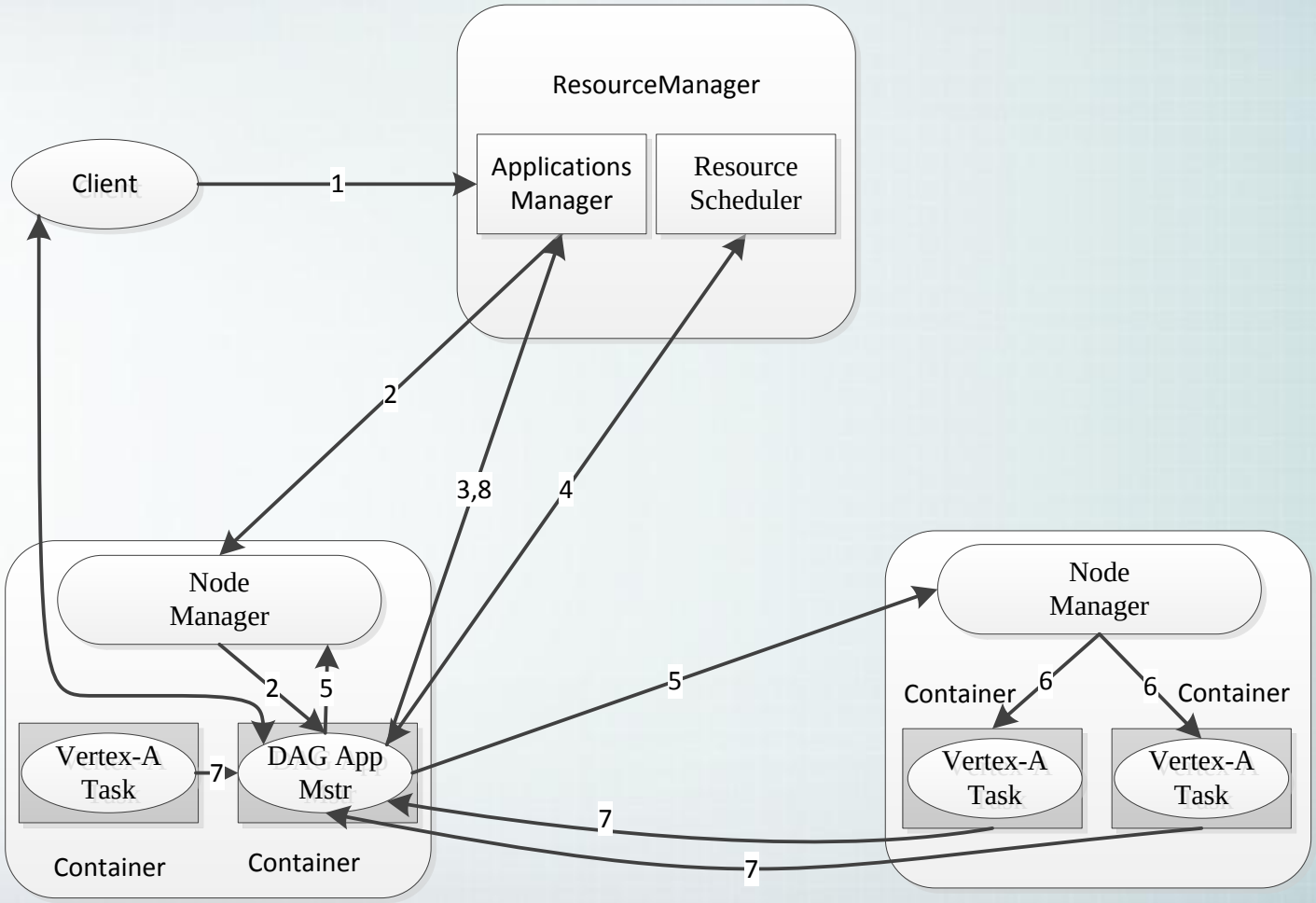
DAG计算框架Tez






```
SELECT a.state, COUNT(*), AVERAGE(c.price)
  FROM a
  JOIN b ON(a.id = b.id)
  JOIN c ON(a.itemId = c.itemId)
  GROUP BY a.state
```

Tez On YARN


ResourceManager
Applications Manager
Resource Scheduler
Client
1
2
3,8
4
Node Manager
2
5
Vertex-A Task
7
DAG App Mstr
Container
Container
5
Node Manager
Container
6
6
Container
Vertex-A Task
Vertex-A Task
7
7

Tez优化技术

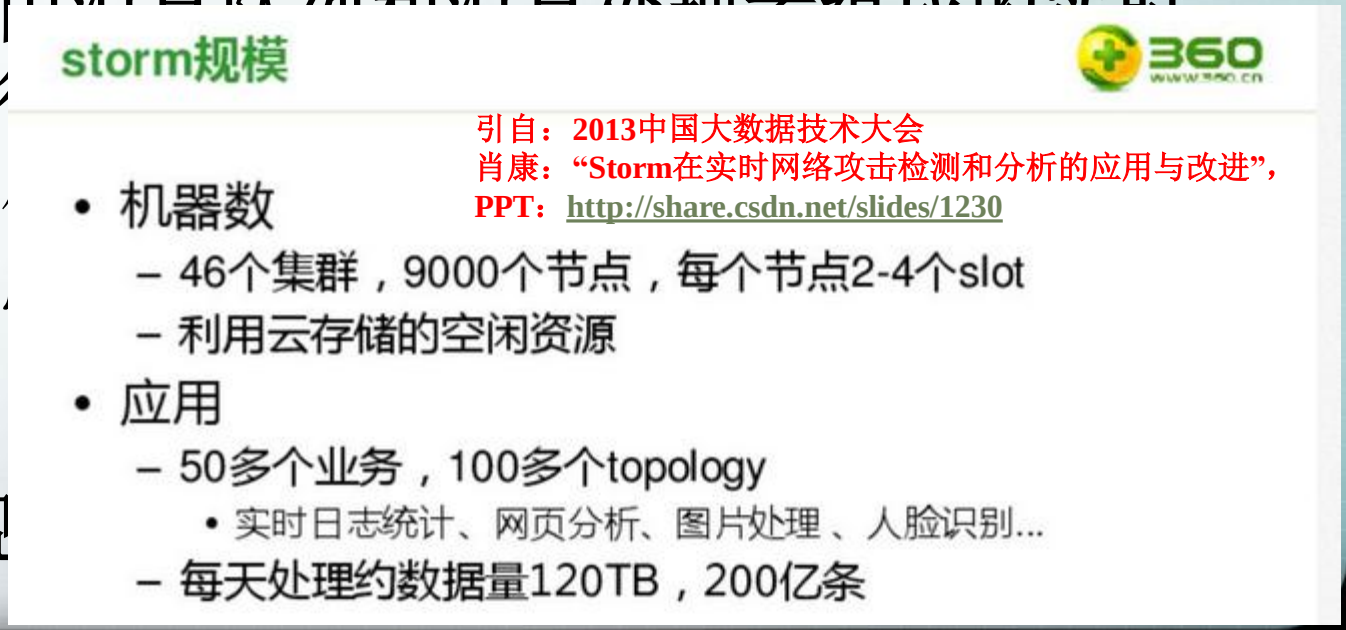
- **ApplicationMaster缓冲池**
  - ✓ 作业提交到AMPoolServer服务上
  - ✓ 预启动若干个ApplicationMaster，形成一个ApplicationMaster缓冲池
- **预先启动Container**
  - ✓ ApplicationMaster启动时可以预先启动若干个Container
- **Container重用**
  - ✓ 任务运行完成后，ApplicationMaster不会马上注销它使用的Container，而是将它重新分配给其他未运行的任务

Tez应用场景

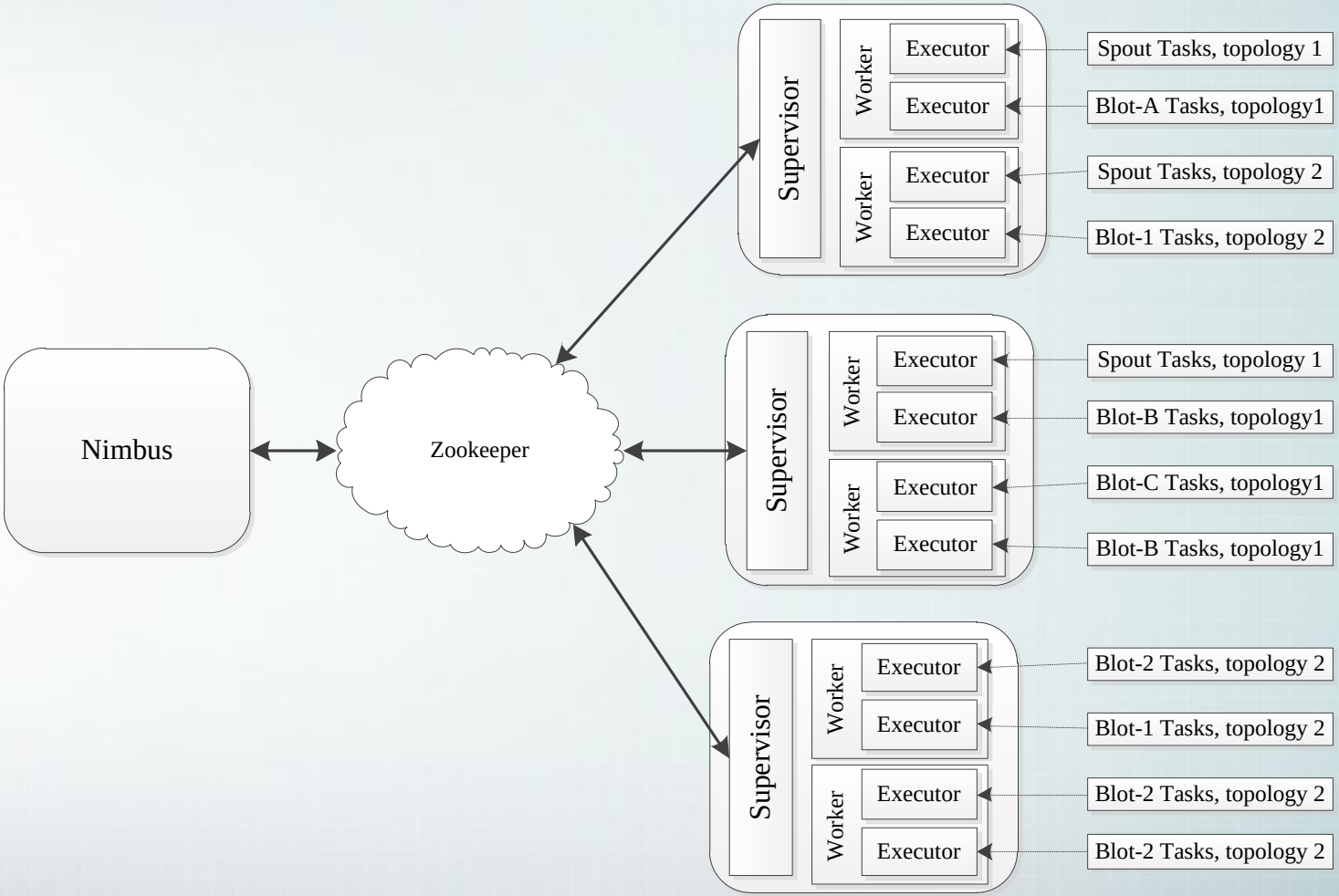
- **直接编写应用程序**
  - ✓ Tez提供了一套通用编程接口
  - ✓ 适合编写有依赖关系的作业
- **优化Pig、Hive等引擎**
  - ✓ 下一代Hive：Stinger
  - ✓ 好处1：避免查询语句转换成过多的MapReduce作业后产生大量不必要的网络和磁盘IO
  - ✓ 好处2：更加智能的任务处理引擎

流式计算Storm

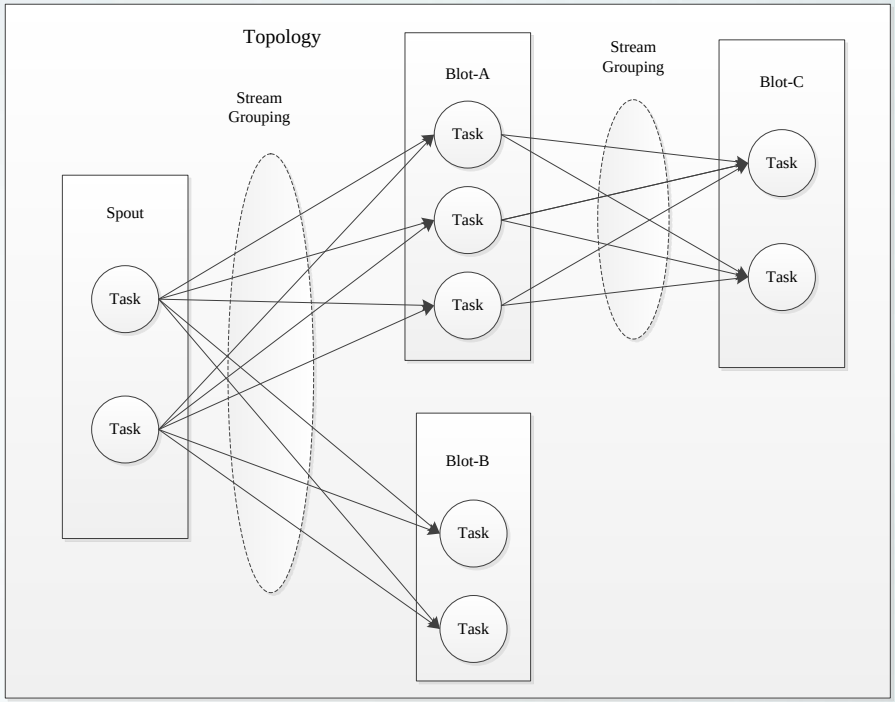
- 流式（**Streaming**）计算，是指被处理的数据像流水一样不断流入系统，而系统需要针对每条数据进行实时处理和计算，并永不停止（直到用户显式杀死进程）；
- 传统做法：中消息队列和消息处理者组成的实时处理网络进行
  - ✓缺乏自动
  - ✓缺乏健壮
  - ✓伸缩性差
- **Storm出现**



流式计算框架Storm

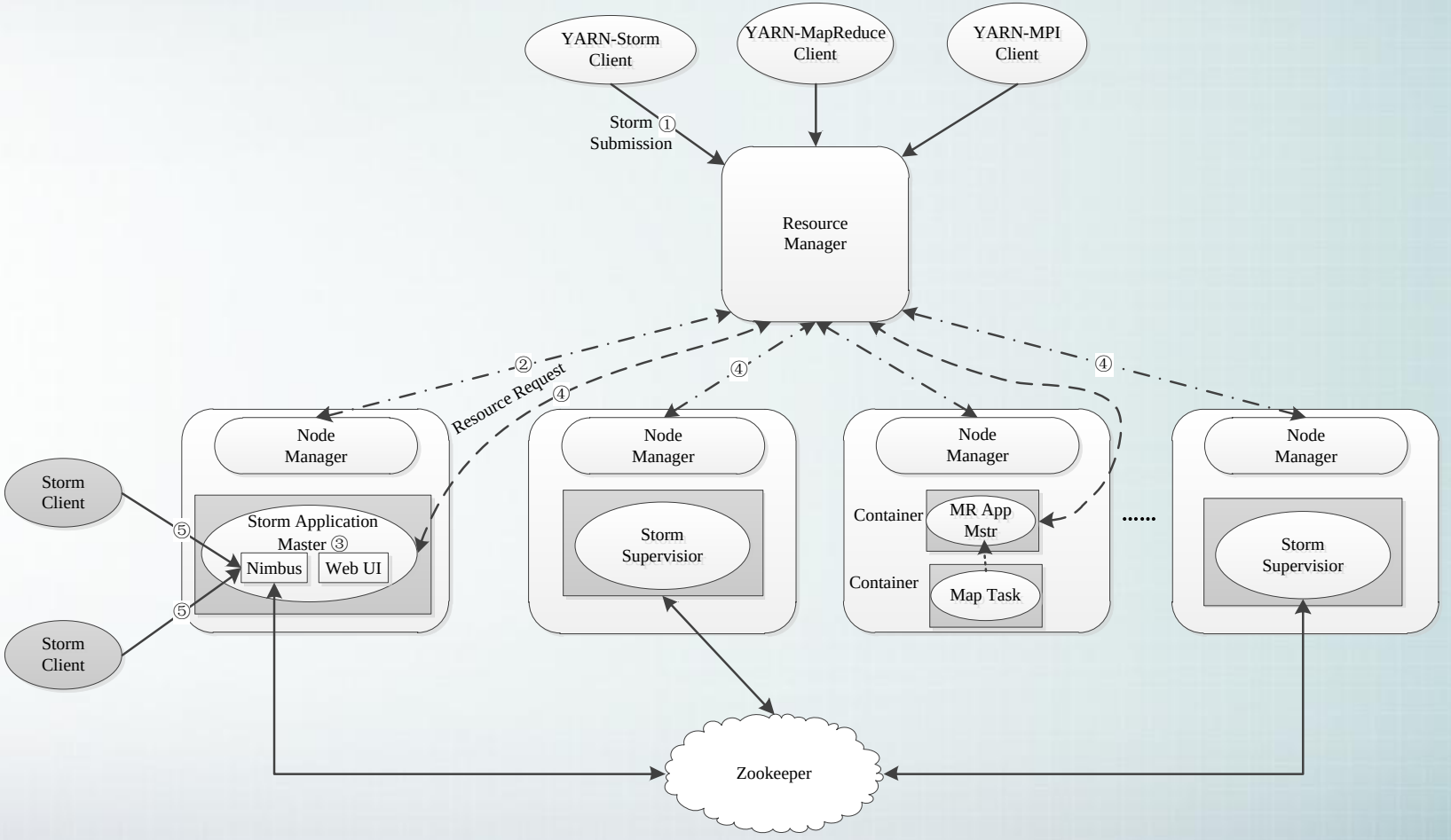


流式计算框架Storm



	Hadoop MapReduce（MRv1）	Storm
系统服务	JobTracker	Nimbus
	TaskTracker	Supervisor
	Child	Worker
应用程序名称	Job	Topology
编程模型	Map/Reduce	Spout/Blot
	Shuffle	Stream Grouping

Storm On YARN

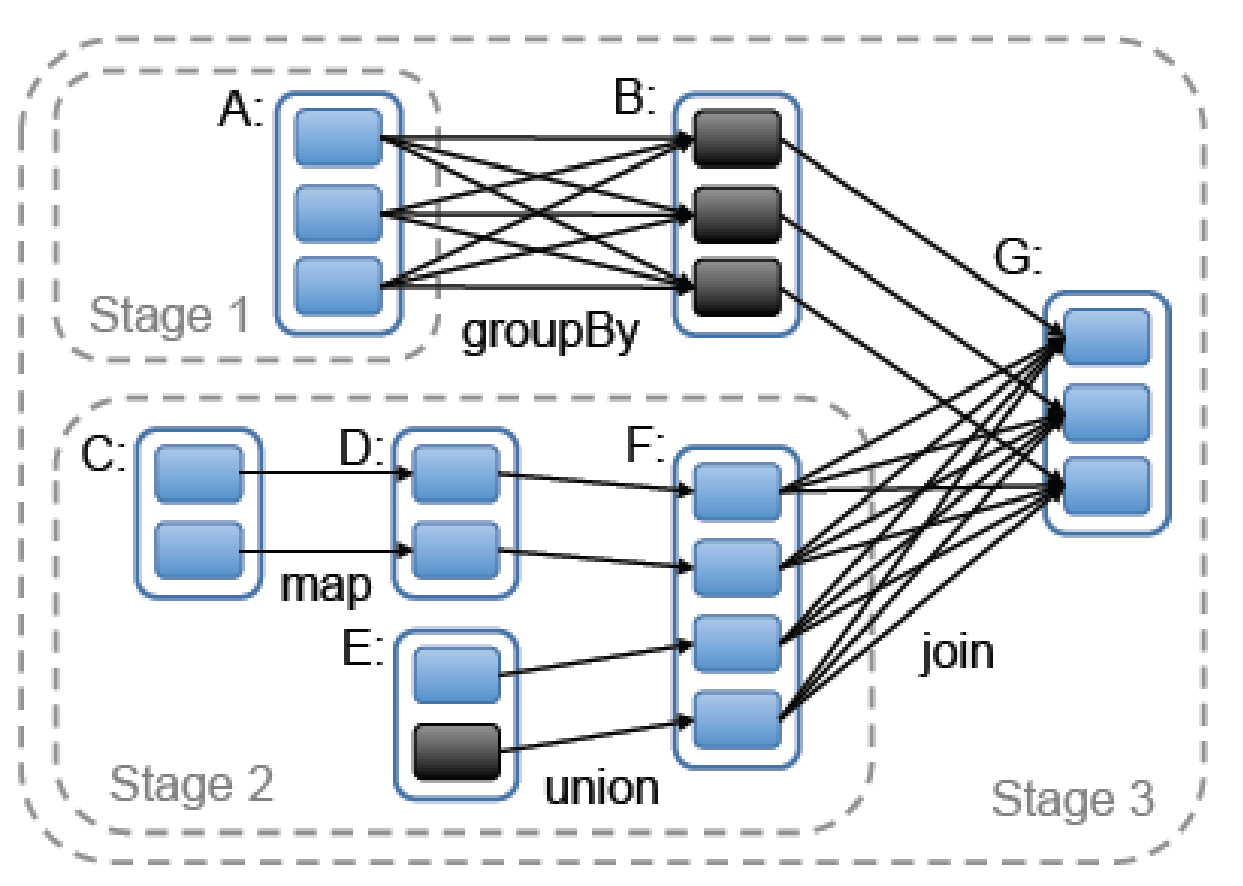


内存计算框架Spark

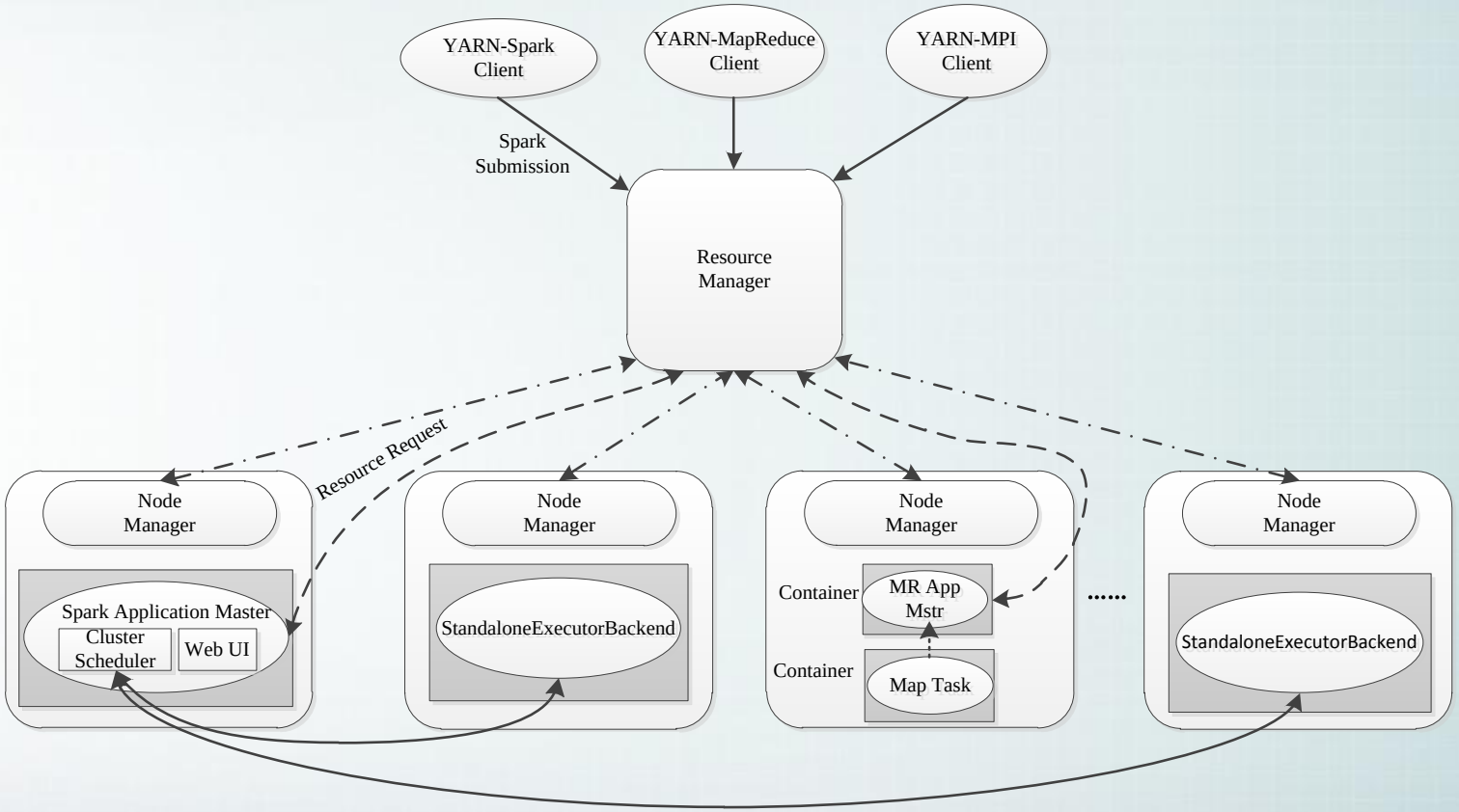
- 克服**MapReduce**在迭代式计算和交互式计算方面的不足；
- 引入**RDD（Resilient Distributed Datasets）**数据表示模型；
- **RDD**是[illegible]行操作的数据集合，[illegible]

引自：“基于Spark on Yarn的淘宝数据挖掘平台”，
PPT：http://vdisk.weibo.com/s/dn9q7A_XuVrf

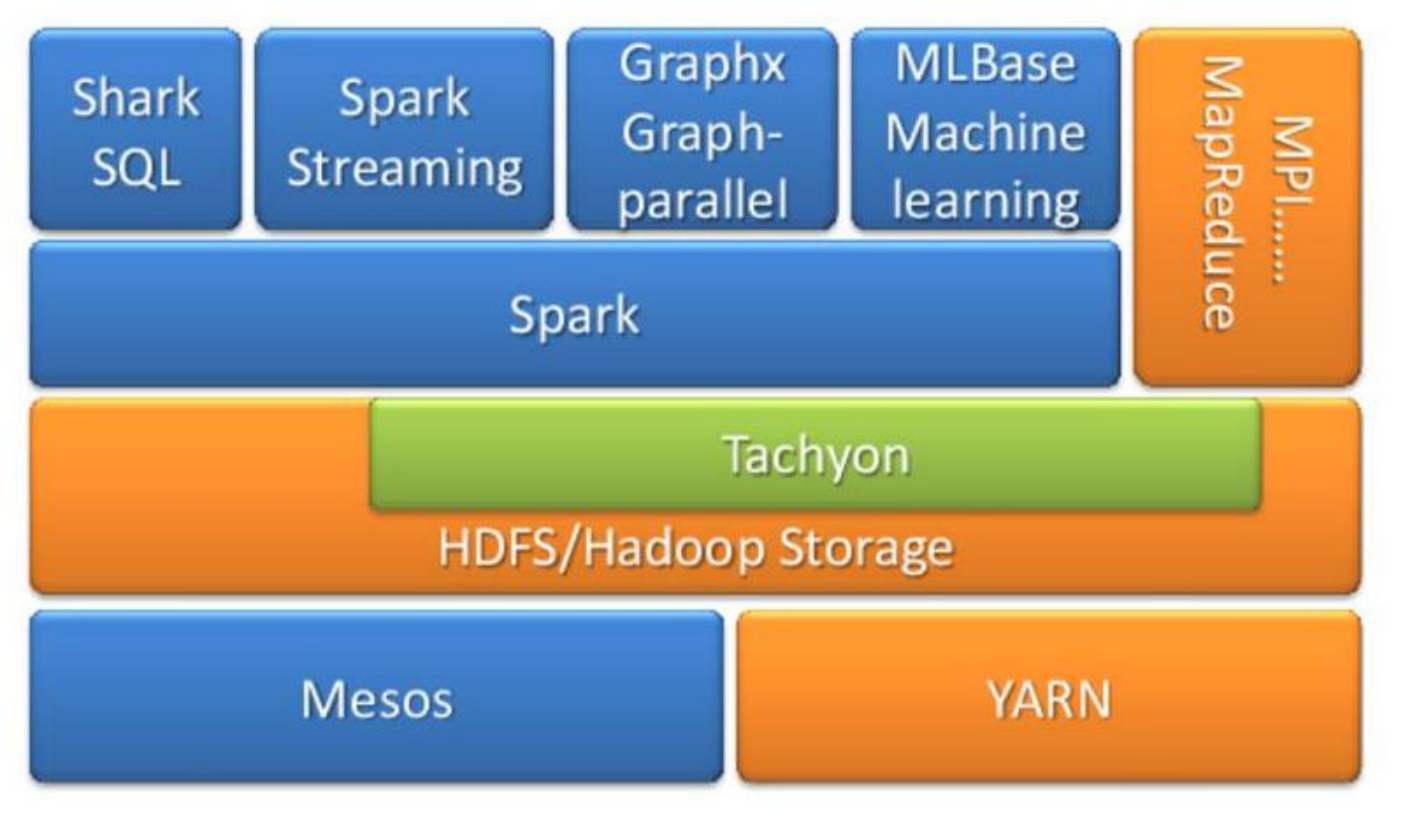
内存计算框架Spark



Spark On YARN



Spark生态系统


Shark SQL
Spark Streaming
Graphx Graph-parallel
MLBase Machine learning
MPI..... MapReduce
Spark
Tachyon
HDFS/Hadoop Storage
Mesos
YARN

其他Framework On YARN

- **Hoya：HBase on YARN ；**

https://github.com/hortonworks/hoya/

- **LLAMA：Impala On YARN**

http://cloudera.github.io/llama/

- **Kafka On YARN**

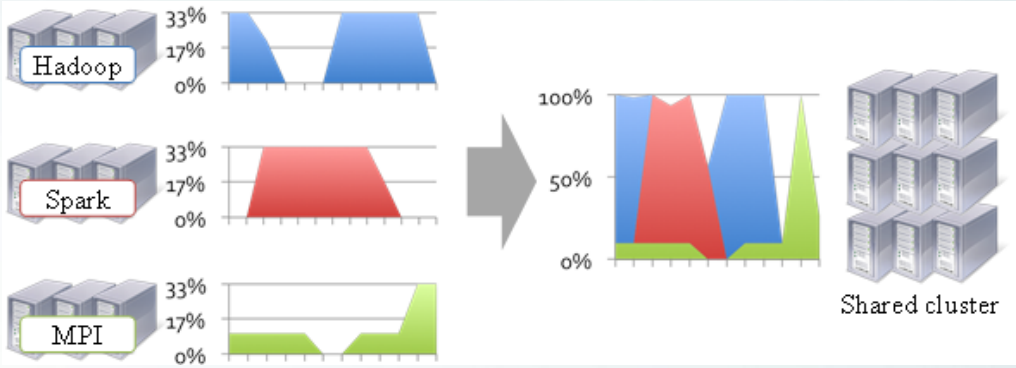
https://github.com/kkasravi/kafka-yarn

主要内容

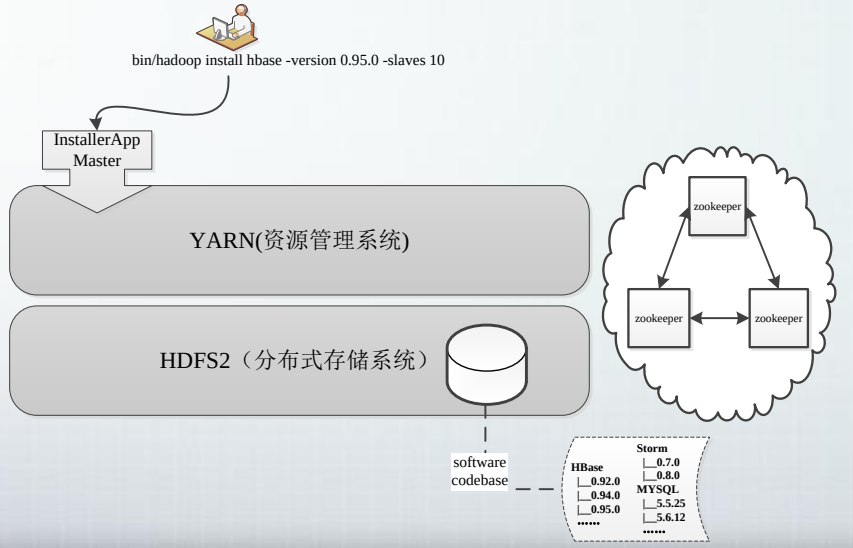
1. Hadoop YARN产生背景
2. Hadoop YARN基本架构
3. 运行在YARN上的计算框架
4. YARN发展趋势

资源管理系统带来的好处

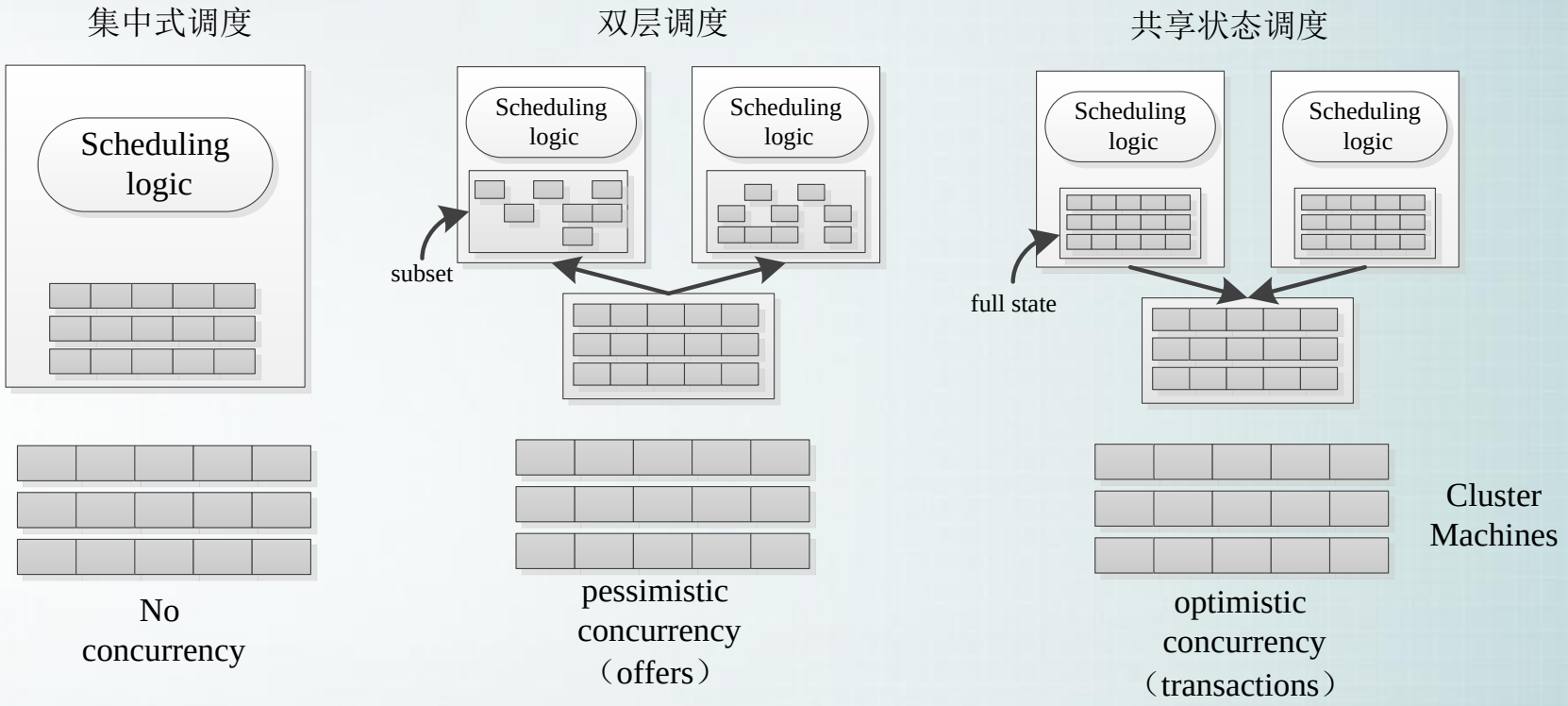
- 提高集群资源利用率



- 服务自动化部署



资源管理系统发展趋势



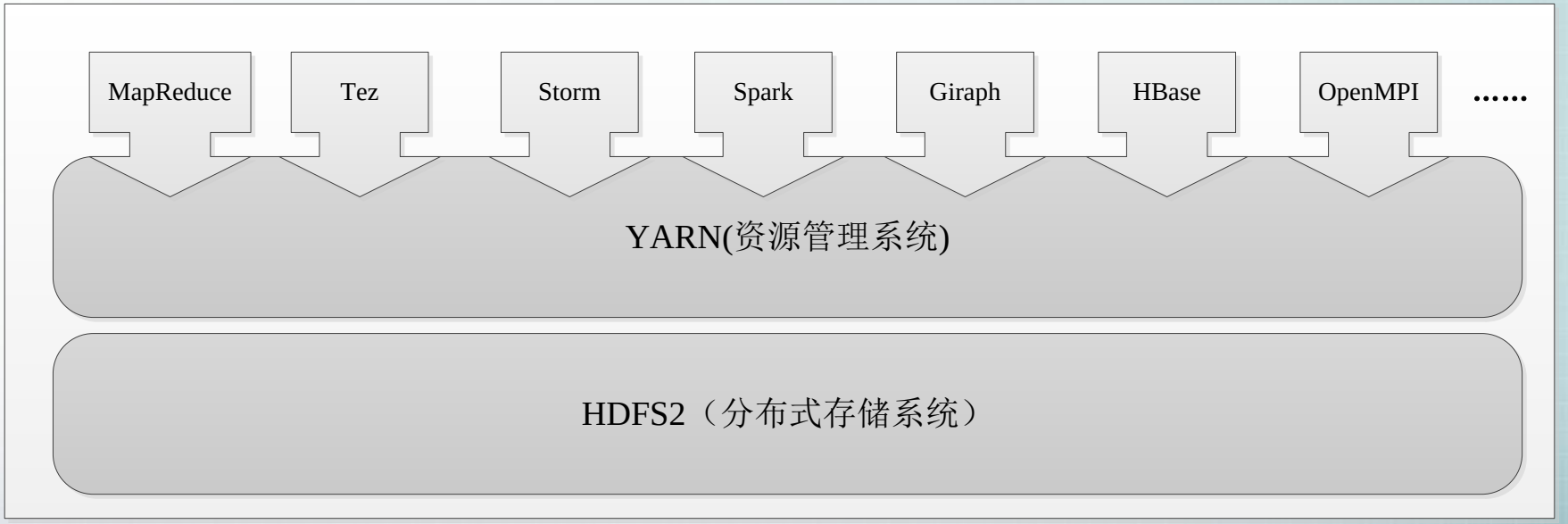
引自：Google论文“Omega: flexible, scalable schedulers for large compute clusters”

YARN自身的完善

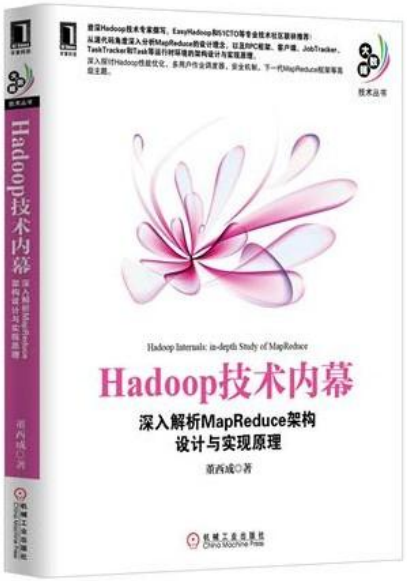
- 调度框架的完善:YARN-896
  - ✓支持更多的资源类型（网络、磁盘等）
  - ✓支持更多的调度语义
- 长作业的在线升级
  - ✓Storm在线升级等
  - ✓Container资源动态调整：YARN-1197
- 容错机制
  - ✓ResourceManager自身容错
  - ✓NodeManager宕掉，任务不受影响
  - ✓ApplicationMaster个性化容错

以YARN为核心的生态系统

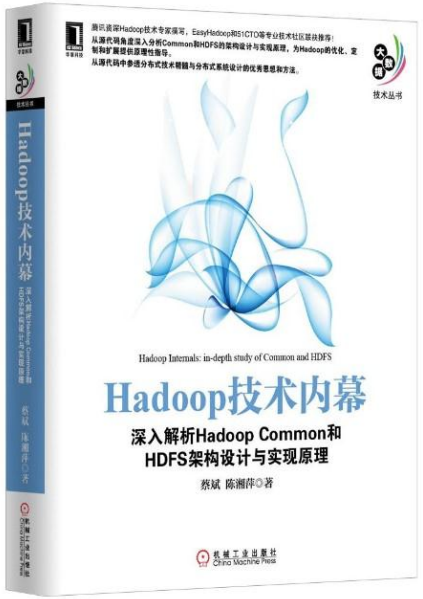
- **YARN使得Hadoop生态系统更加强大**
  - ✓吸纳其他开源计算框架，比Storm、Spark等
  - ✓变为集群管理者



Hadoop技术内幕


Hadoop技术内幕
Hadoop Internals: in-depth Study of MapReduce
Hadoop技术内幕
深入解析MapReduce架构
设计与实现原理
董西成◎著
技术丛书


Hadoop技术内幕
Hadoop Internals: in-depth study of YARN
Hadoop技术内幕
深入解析YARN架构设计与实现原理
董西成◎著
技术丛书


Hadoop技术内幕
Hadoop Internals: in-depth study of Common and HDFS
Hadoop技术内幕
深入解析Hadoop Common和
HDFS架构设计与实现原理
蔡斌 陈湘萍◎著
技术丛书

引用链接

- **Hadoop；**

http://hadoop.apache.org/

- **Tez**

http://tez.incubator.apache.org/

- **Storm**

http://storm-project.net/

- **Storm On YARN**

https://github.com/yahoo/storm-yarn

- **Spark**

http://spark.incubator.apache.org/

- **Spark On YARN**

https://github.com/apache/incubator-spark/tree/branch-0.8/yarn

Q&A?

Thank You !

【联系我】

邮　箱：dongxicheng@yahoo.com
微信号：hadoop-123（每天一篇hadoop文章）
微博ID：西成懂
博　客：dongxicheng.org
书　籍：hadoop123.com