

# Spark企业级大数据项目实战 第11课

DATAGURU专业数据分析社区

**【声明】** 本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

<http://edu.dataguru.cn>

- Spark优化与正确使用广播变量
- ElasticSearch与Hbase整合
- Spark读写关系型数据库（集群与DB网络不通，Driver调优）
- Azkaban调度

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 <http://edu.dataguru.cn>

## 广播变量使用的场景

### 1. 减轻内存和网络传输开销，降低GC频率

对于某个变量，如果不是用广播变量，如果一个Executor中有N个task，那么这个Executor中就有N个此变量的副本。

如果使用广播变量，在Executor中，只有一个此变量的副本。

### 2. 避免Shuffle

在Map端join，没有shuffle

## 广播变量:

//商品信息(商品id,商品名称)

```
val goods_info = sc.parallelize(Array(("001","phone"),("002","tv"),("003","notebook"))).collectAsMap()
```

//订单详细信息(商品id, 订单地址, 订单id)

```
val orders = sc.parallelize(Array(("001","hefei","1000023"),  
    ("002","beijing","1000024"),  
    ("003","nanjing","1000025"),  
    ("002","hangzhou","1000026")))
```

// 广播小的数据集

```
val goodinfoBroad = sc.broadcast(goods_info)
```

# 1 Spark优化与广播变量-案例

## 使用变量:

方式一:

```
val res2 = orders.mapPartitions(iter =>{
    val goodMap = goodinfoBroad.value
    val arrayBuffer = ArrayBuffer[(String,String,String)]()
    iter.foreach{case (goodsId,address,orderId) =>{
        if(goodMap.contains(goodsId)){
            arrayBuffer.+= ((orderId, goodMap.getOrElse(goodsId,""),address))
        }
    }}
    arrayBuffer.iterator
})
```

方式二:

```
val res1 = orders.mapPartitions(iter => {
    val goodMap = goodinfoBroad.value
    for{
        (goodId, address, orderid) <- iter
        if(goodMap.contains(goodId))
    } yield (goodId, goodMap.getOrElse(goodId,""),address)
})
```

# 1 Spark优化与广播变量-正确使用广播变量

1. 如果不使用如下语句进行广播， 直接使用goods\_info进行关联？

```
val goodinfoBroad = sc.broadcast(goods_info)
```

2. 将mapPartition改成map？

```
val res = orders.map(iter =>{  
  val goodMap = goodinfoBroad.value  
  val arrayBuffer = ArrayBuffer[(String,String,String)]()  
  if(goodMap.contains(iter._1)){  
    arrayBuffer += ((iter._1, goodMap.getOrElse(iter._1,""),iter._2))  
  }  
  arrayBuffer  
})
```

每个Task都会复制  
goods\_info的部分，  
浪费资源

频繁创建goodMap  
=goodinfoBroad.value



# 1 Spark优化与广播变量-Spark SQL广播表

1. 将DataFrame注册成表

```
df.registerTempTable("smalltable")
sql("CACHE TABLE smalTable")
```

2. 设置spark.sql.autoBroadcastJoinThreshold

spark.sql.autoBroadcastJoinThreshold默认为 $10 * 1024 * 1024$ Bytes

只有小于这个值的表才会被广播

**错误的方式: `sc.broadcast(df)`**

**不能将RDD或者DataFrame直接广播出去**

### Hbase二级索引

Hbase 非rowkey字段查询只能全表扫描， 不支持模糊查询

建立索引：

将hbase 需要索引的字段和rowkey存入Elasticsearch， 对于文档关键字有检索需求可以使用分词。

根据二级索引字段查询：

1. 根据查询的字段条件， 从Elasticsearch查询数据， 并返回查询结果的rowkey。
2. 根据第一步返回的rowkey， 去hbase检索数据， 并返回结果集。


## 3 Spark读写关系型数据库

### □ 读MySQL

```
var jdbcDf=sqlContext.read.format("jdbc").options(Map("url">"jdbc:mysql://localhost:3306/myspark",  
"driver" -> "com.mysql.jdbc.Driver",  
"dbtable"->"mytopic",  
"user"->"root","password"->"123456")).load()
```

### □ 写MySQL

```
// 将DataFrame数据写入mysql  
val prop=new Properties()  
prop.setProperty("user","root")  
prop.setProperty("password","123456")  
val url = "jdbc:mysql://localhost:3306/myspark"  
df.write.mode(SaveMode.Append).jdbc(url,"my_kafka",prop)
```

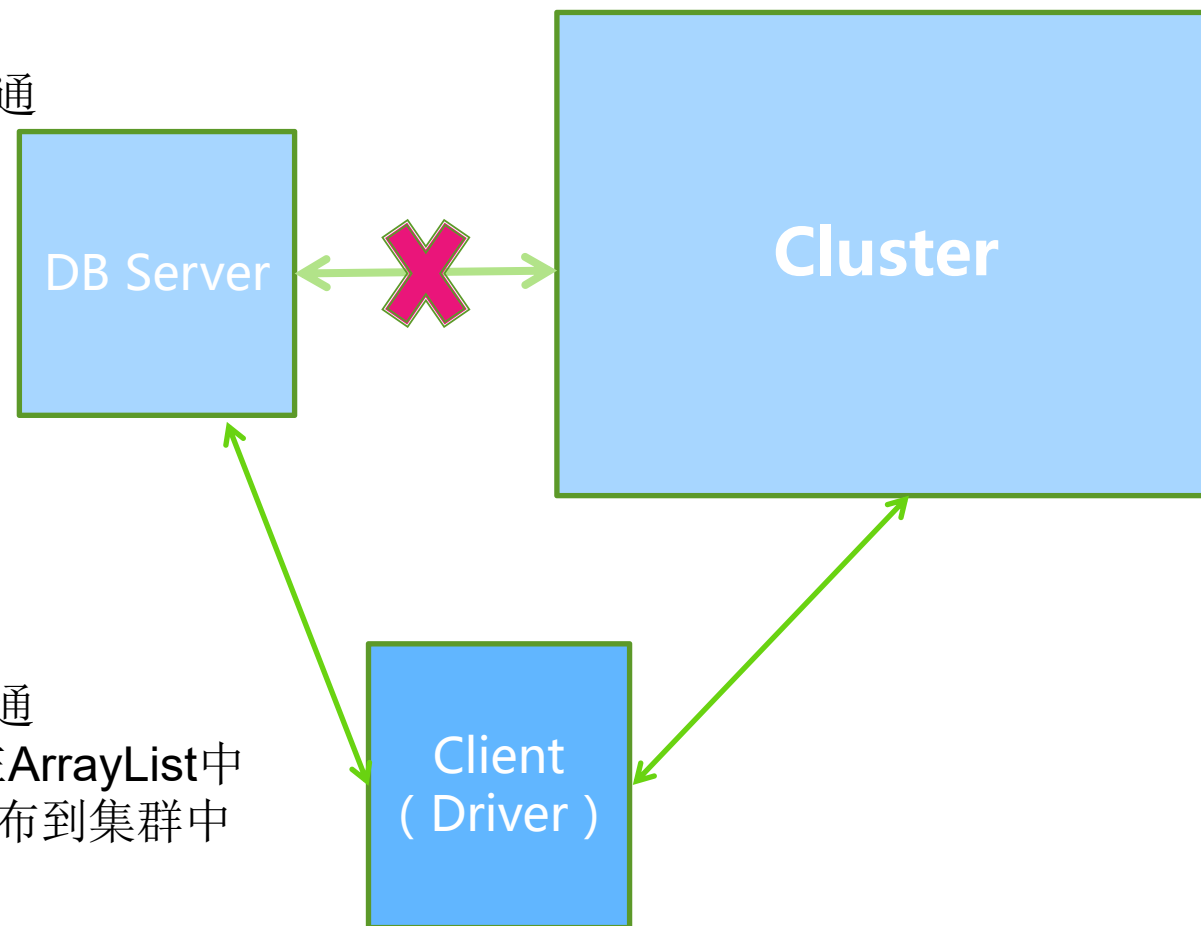


**集群节点与DB  
服务器网络不  
通？**

## 3 Spark读写关系型数据库

### □ 写MySQL数据

1. Spark 程序提交使用yarn-client模式
2. Spark客户端与MySQL服务器和Hadoop集群网络连通
3. 从集群里面读取数据， rdd或者DataFrame， 使用collect算子， 将数据拉取到driver端
4. 在driver端（hadoop客户端）将数据写入到MySQL



### □ 读MySQL

1. Spark 程序提交使用yarn-client模式
2. Spark客户端与MySQL服务器和Hadoop集群网络连通
3. 使用单机模式， 从MySQL读取数据， 将数据保存在ArrayList中
4. 使用MakeRDD或者parallalize 将数据转换成RDD分布到集群中

### 分发本地集合形成RDD

```
def parallelize[T: ClassTag](  
    seq: Seq[T],  
    numSlices: Int = defaultParallelism): RDD[T] = withScope { }
```

// 与parallelize功能相同

```
def makeRDD[T: ClassTag](  
    seq: Seq[T],  
    numSlices: Int = defaultParallelism): RDD[T] = withScope { }
```

// 分配本地Scala集合以形成RDD，并为每个对象提供一个或多个位置首选项（Spark节点的主机名）。为每个集合item创建一个新的分区。


```
def makeRDD[T: ClassTag](seq: Seq[(T, Seq[String])]): RDD[T] = withScope { }
```

### Driver内存

**collect** 算子将数据拉取到**driver**端， **driver**内存必须足够大。

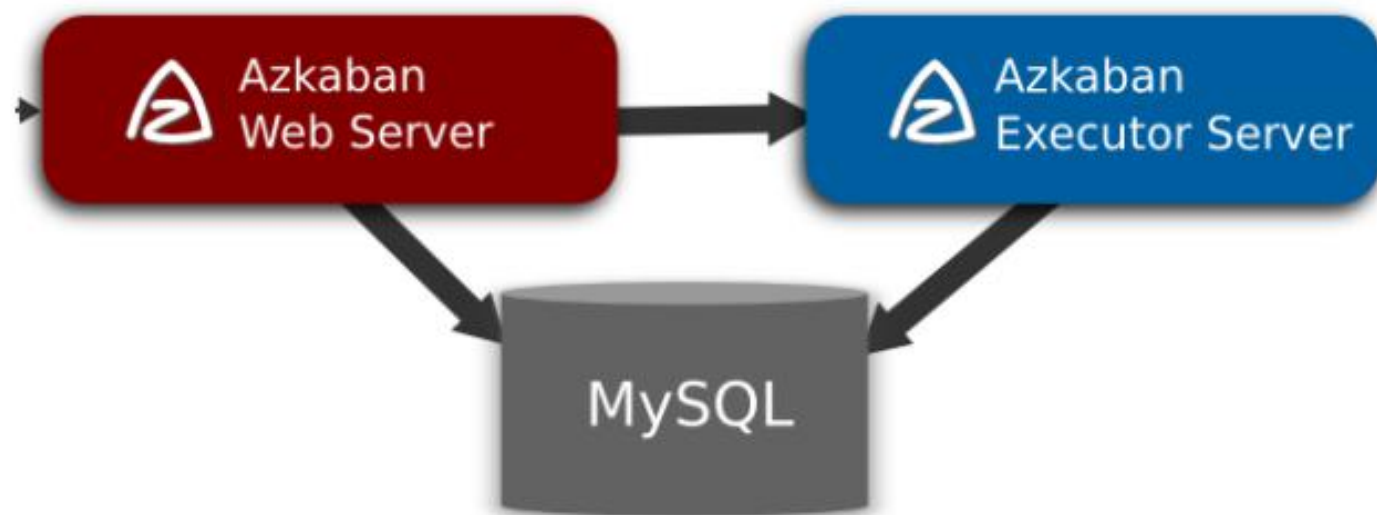
```
$ ./bin/spark-submit --class org.apache.spark.examples.SparkPi \  
  --master yarn \  
  --deploy-mode cluster \  
  --driver-memory 4g \  
  --executor-memory 2g \  
  --executor-cores 1 \  
  --queue thequeue \  
  examples/jars/spark-examples*.jar
```

### Spark读写ElasticSearch ?



**集群节点与ES  
服务器网络不  
通？**

## 4 Azkaban调度





## 4 Azkaban 编译

### 1. 下载azkaban源码

github地址: <https://github.com/azkaban/azkaban>

### 2. 在azkaban的源码目录编译

# Build without running tests

./gradlew build -x test

注: 有关详细信息, 请使用 `-Xlint:deprecation` 重新编译。

> Task :az-reportal:compileJava

注: /hadoop/azkaban/azkaban-master/az-reportal/src/main/java/azkaban/reportal

注: 有关详细信息, 请使用 `-Xlint:deprecation` 重新编译。

> Task :azkaban-web-server:npmSetup

/hadoop/azkaban/azkaban-master/azkaban-web-server/.gradle/npm/npm-v5.6.0/bin/n  
m-v5.6.0/lib/node\_modules/npm/bin/npm-cli.js

/hadoop/azkaban/azkaban-master/azkaban-web-server/.gradle/npm/npm-v5.6.0/bin/n  
m-v5.6.0/lib/node\_modules/npm/bin/npm-cli.js

+ npm@5.6.0

added 476 packages in 18.45s

> Task :azkaban-web-server:npm\_install

added 39 packages in 9.794s

**BUILD SUCCESSFUL in 6m 27s**

54 actionable tasks: 52 executed, 2 from cache

[hadoop@spark123 azkaban-master]\$

编译完成后截图

## 4 Azkaban三种模式

### ❑ solo server mode

内置H2数据库， web server和executor server运行在同一个进程里。 仅用于测试场景。

### ❑ two server mode

适用于比较严格的生产环境。 DB为应当为master-slave架构的MySQL数据库。 web server和executor server应当运行在不同的进程。

### ❑ multiple executor mode

适用于最严格的生产环境。 DB为应当为master-slave架构的MySQL数据库。 web server和executor理想情况下应运行在不同的主机上， 以便升级和维护部影响用户。 这种多主机设置使Azkaban具备强大且可扩展的功能。

## 4 Azkaban安装

- 数据库安装设置
- Azkaban Web Server安装配置
- Azkaban Executor配置

## 4 Azkaban使用

### □ 服务启动

启动MySQL服务

启动WebServer

启动Executor

### □ Azkaban 流程创建

创建jobs

创建流程

嵌入式流程

### □ Job配置

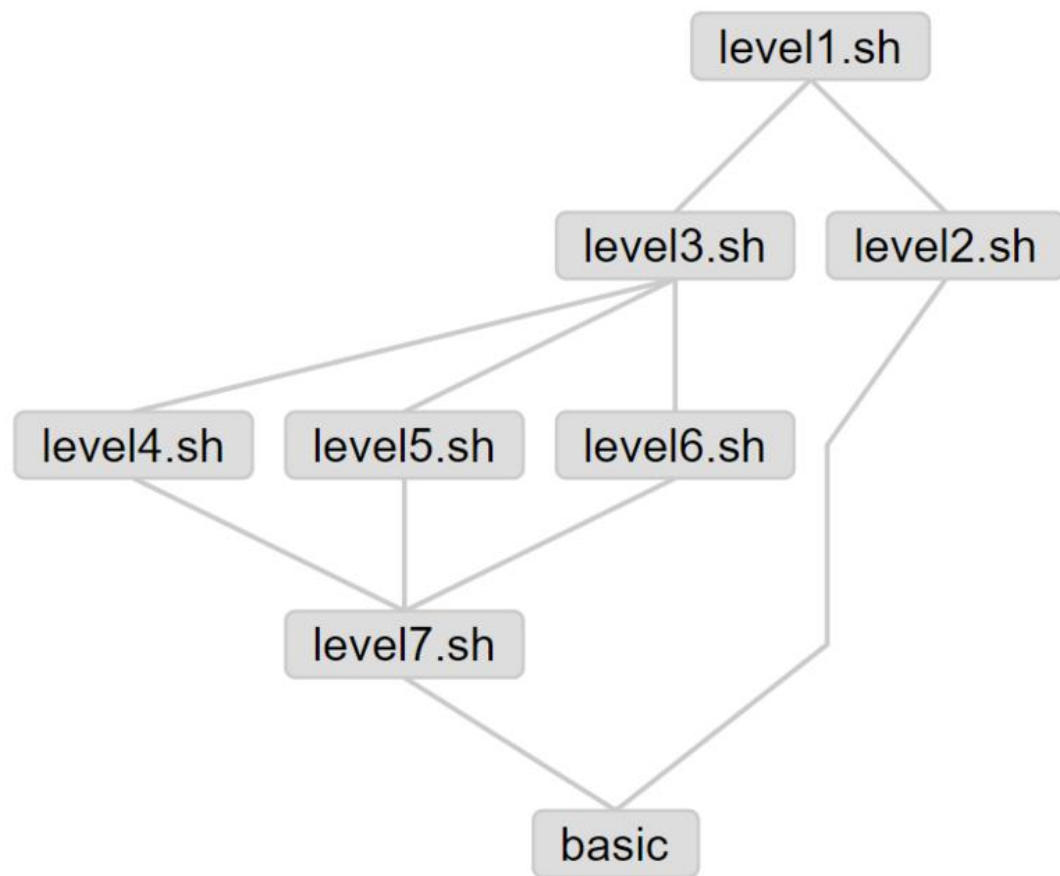
通用参数

运行时属性

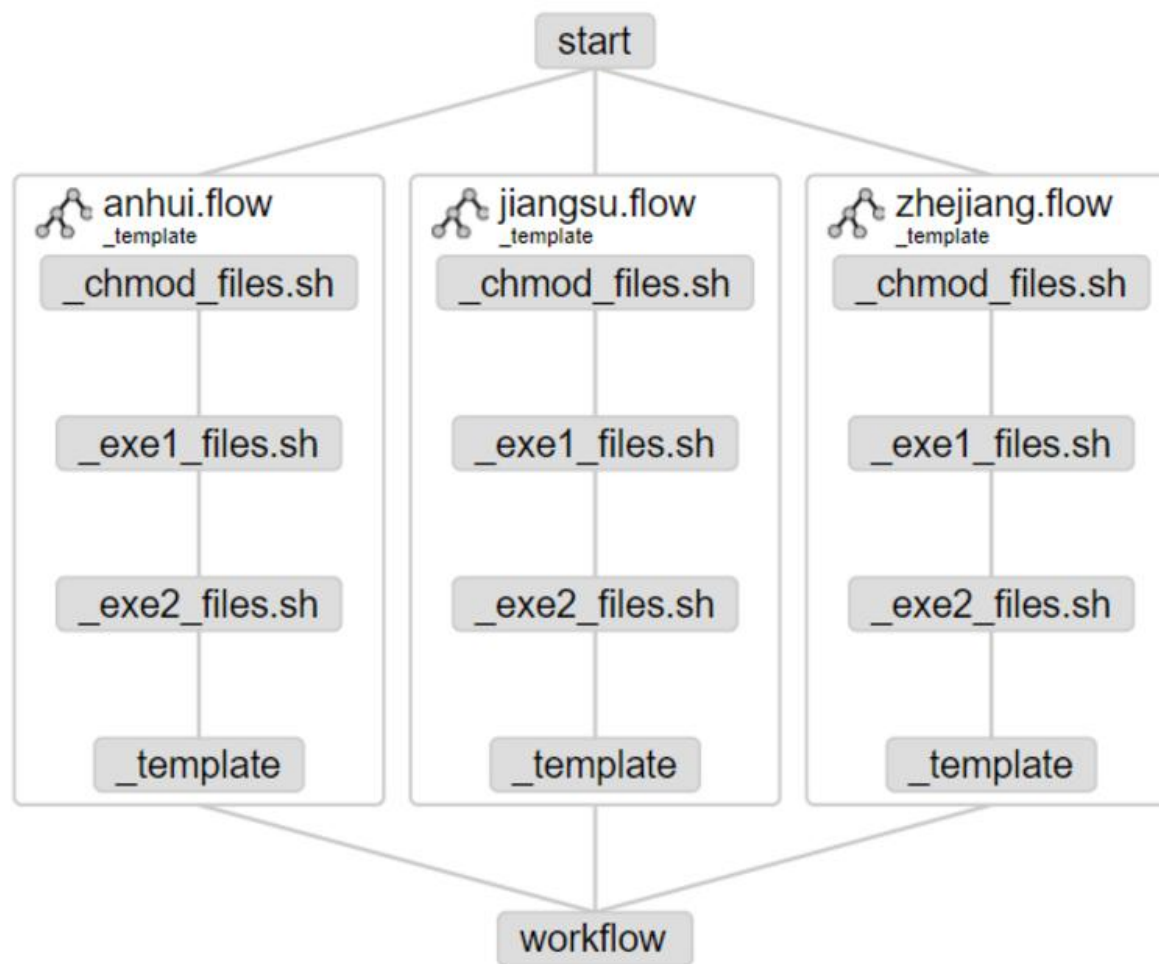
继承的参数

参数变量替换

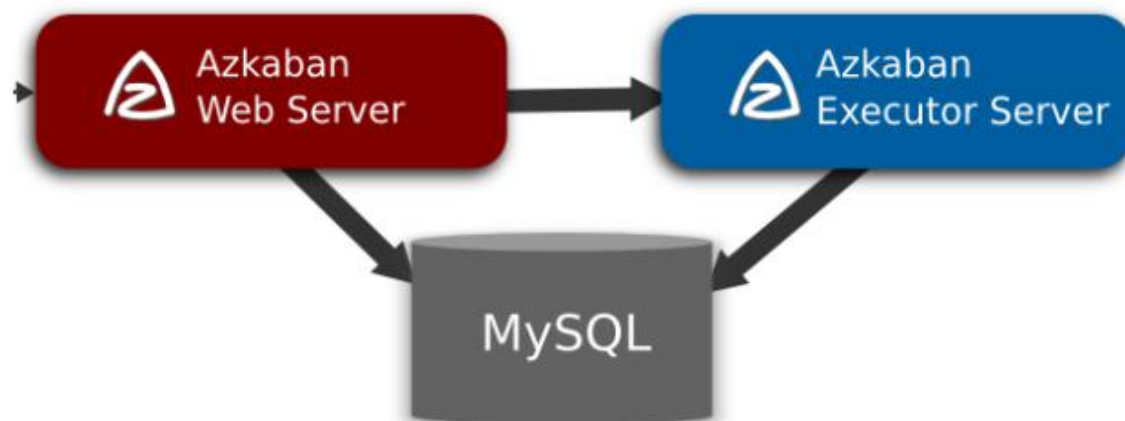
## 4 Azkaban使用案例1-基本流程



## 4 Azkaban使用案例2-模板流程



## 4 Azkaban高可用



Azkaban Web Server: 两台主机，使用Keepalive, 确保只有一台Web Server工作。

MySQL: 主从复制、HA

Azkaban Executor: Multiple Executor模式

# Thanks

**FAQ时间**