北风网 IBEIFENG.COM

大数据Hadoop高薪直通车课程

Spark初识入门

讲师：轩宇（北风网版权所有)

课程大纲

1. Spark 概述、生态系统
2. Spark 编译、测试
3. Scala 基本知识
4. Spark 安装部署
5. Spark 应用提交

课程大纲

1. Spark 概述、生态系统
2. Spark 编译、测试
3. Scala 基本知识
4. Spark 安装部署
5. Spark 应用提交

What is Spark?

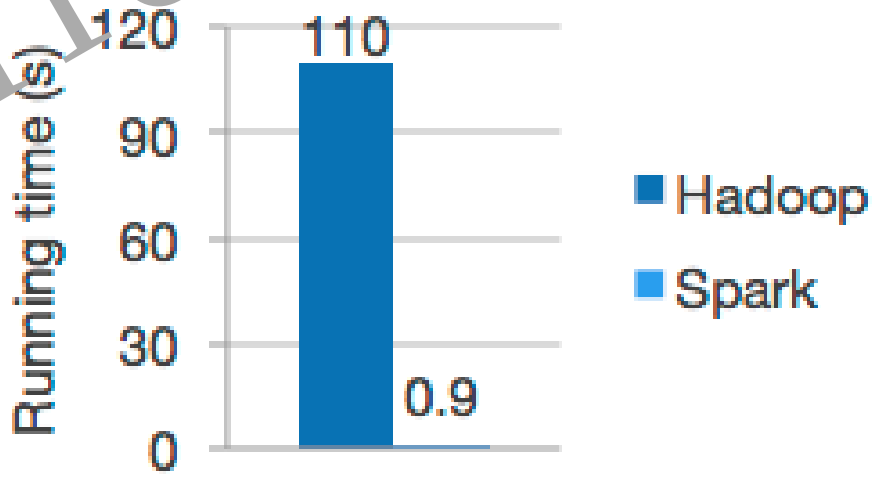
Apache Spark™ is a fast and general engine for large-scale data processing.

What is Spark?

Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.



Logistic regression in Hadoop and Spark

What is Spark?

Ease of Use

Write applications quickly in Java, Scala or Python.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala and Python shells.

```
file = spark.textFile("hdfs://...")

file.flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a+b)
```

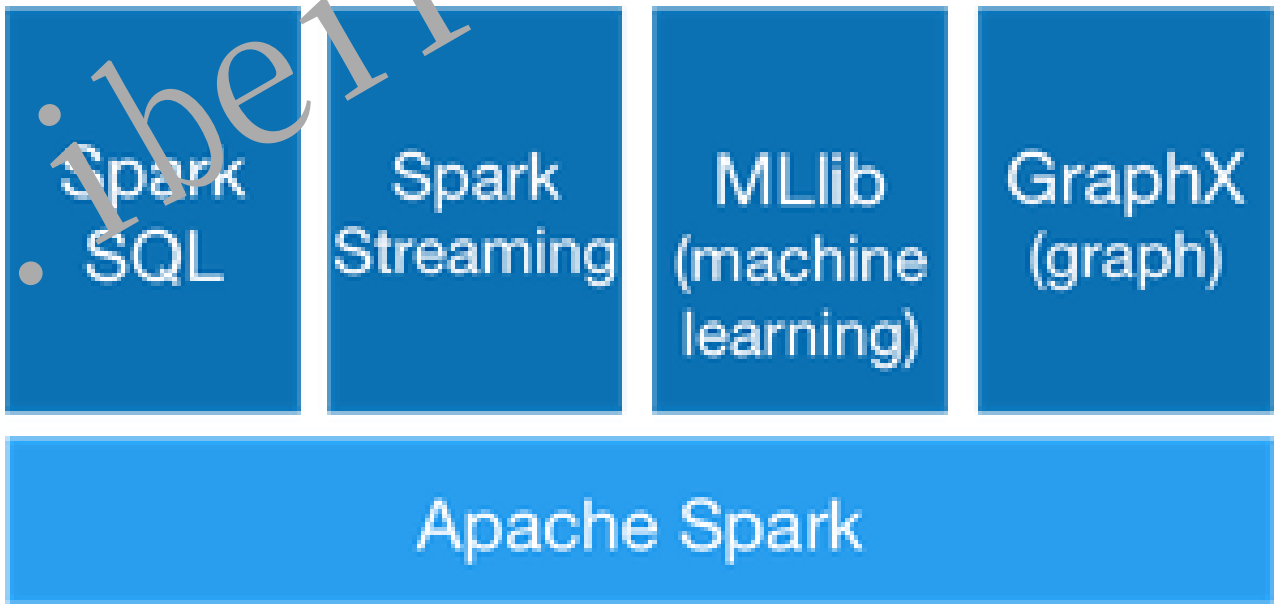
Word count in Spark's Python API

What is Spark?

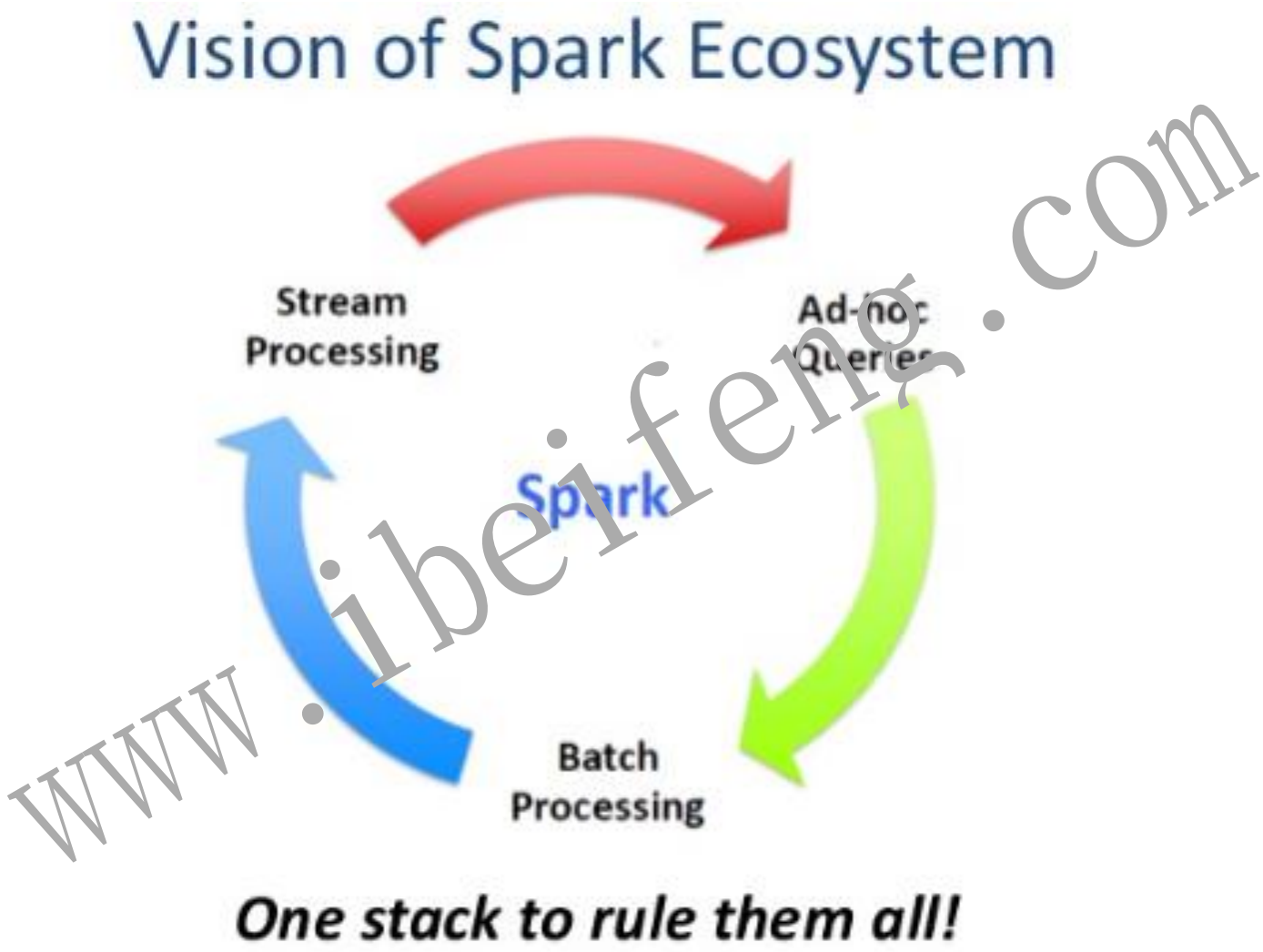
Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of high-level tools including Spark SQL, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.



One Stack to Rule them All



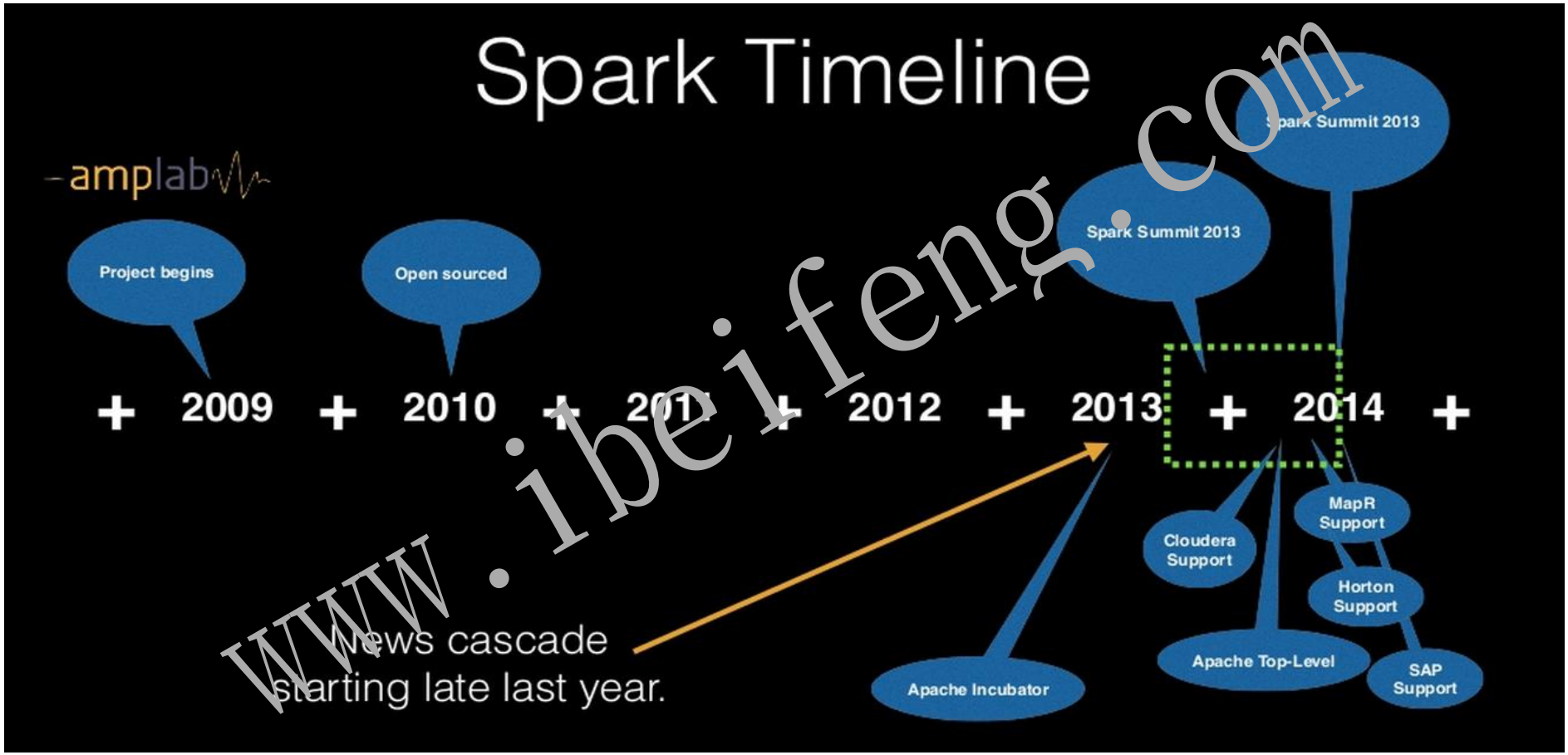
What is Spark?

Runs Everywhere

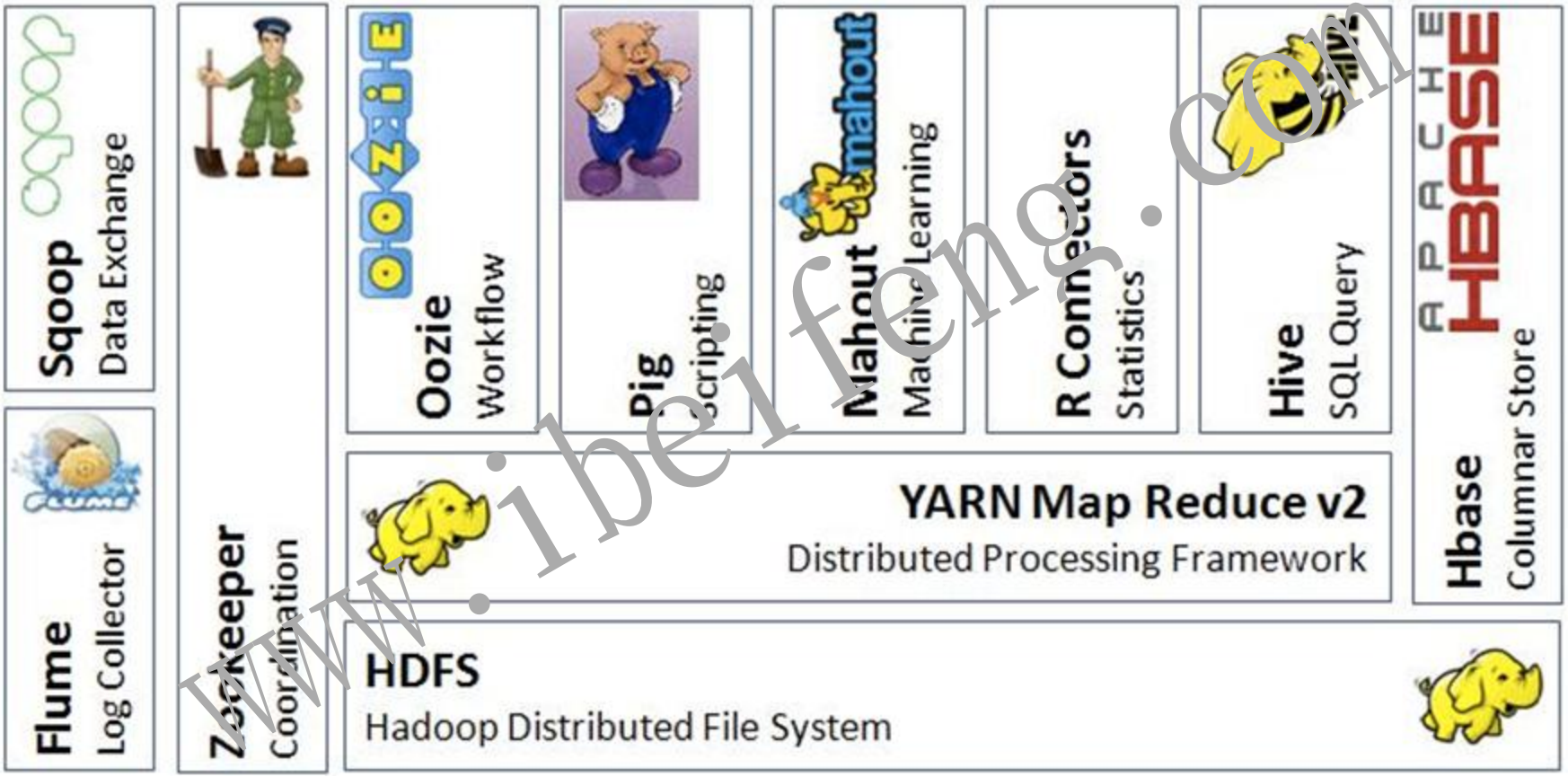
Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, S3.

You can run Spark readily using its standalone cluster mode, on EC2, or run it on Hadoop YARN or Apache Mesos. It can read from HDFS, HBase, Cassandra, and any Hadoop data source.

Spark Timeline

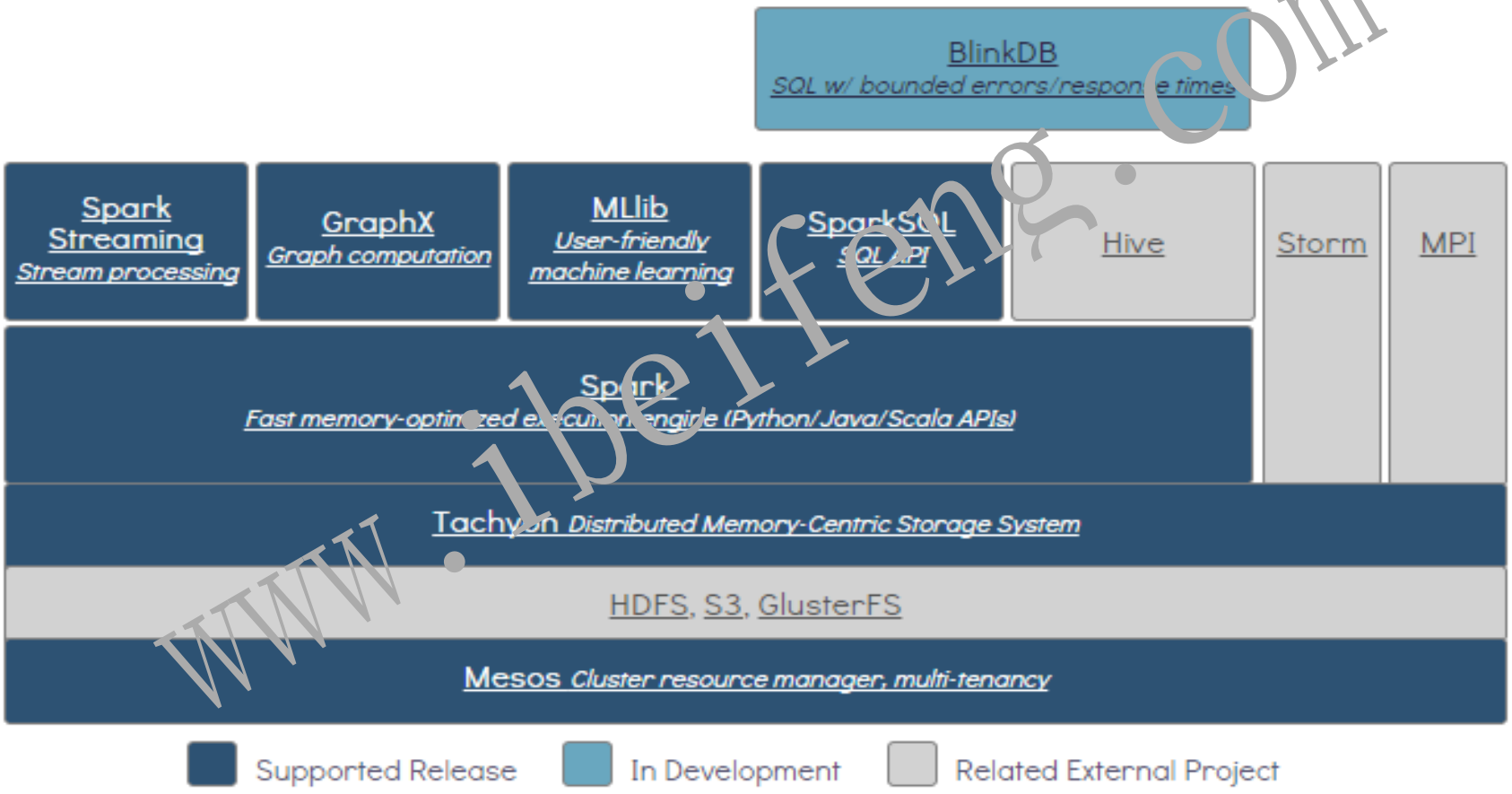


Hadoop Ecosystem



BDAS

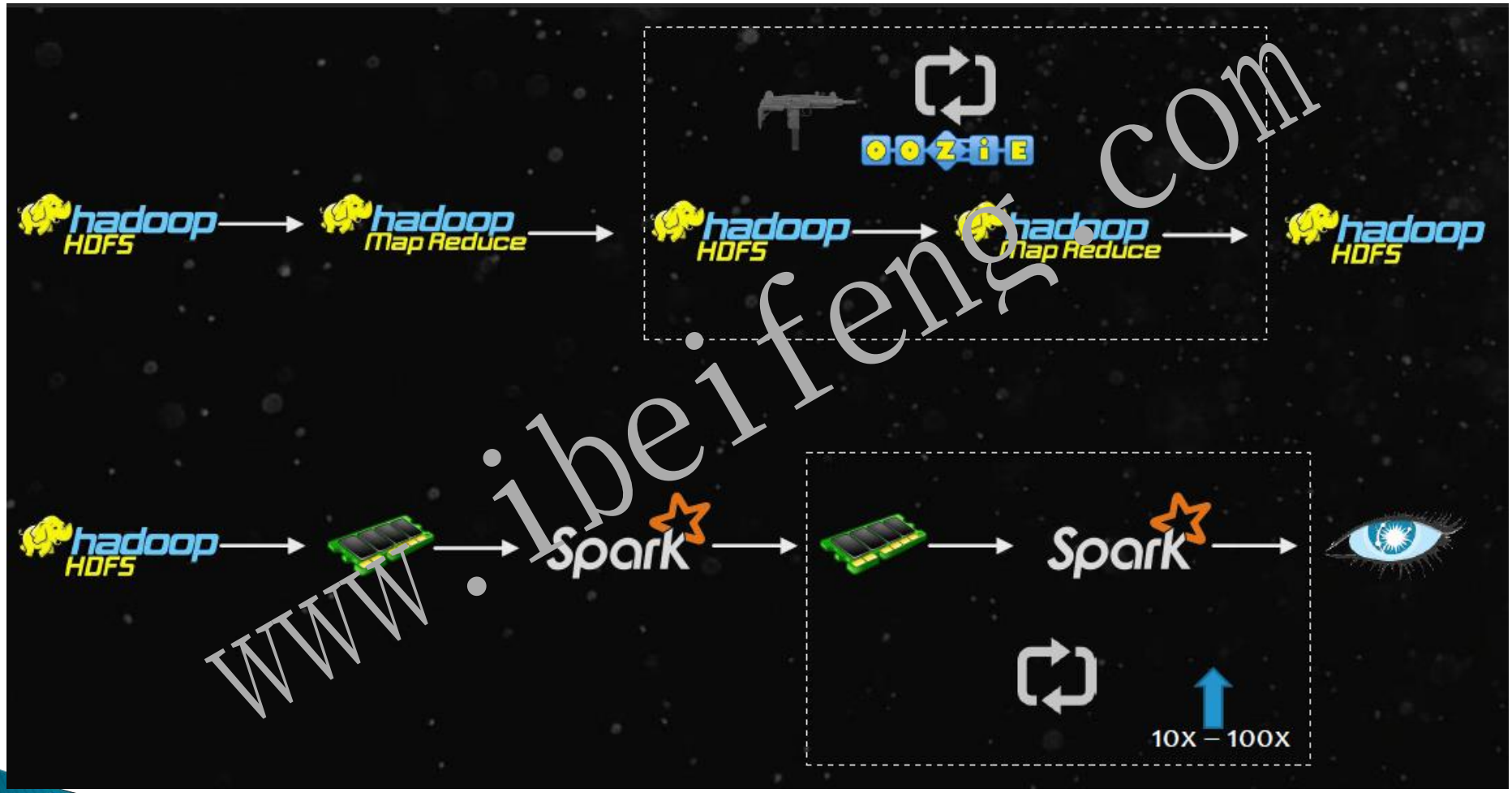
BDAS: Berkeley Data Analytics Stack



Spark vs MapReduce

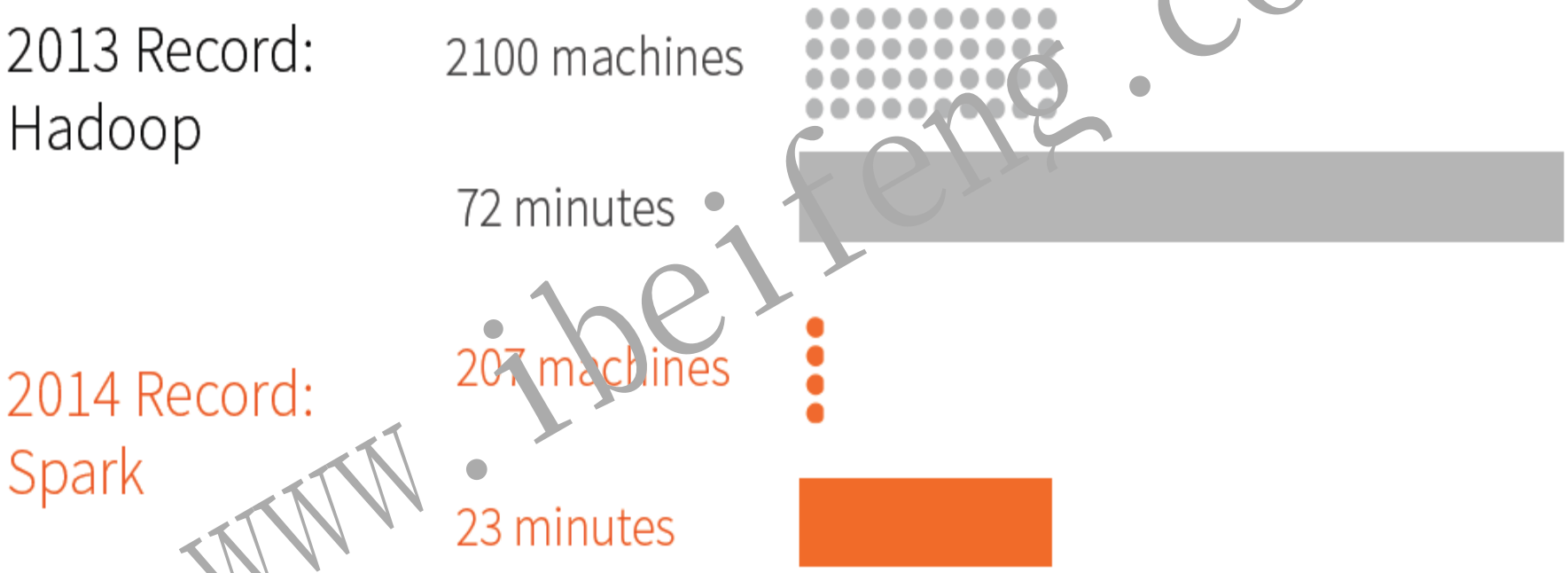
MapReduce	Spark
数据存储结构：磁盘hdfs文件系统的split	使用内存构建弹性分布式数据集RDD，对数据进行运算和cache
编程范式：Map + Reduce	DAG(有向无环图)：Transformation + action
计算中间数据落磁盘，io及序列化、反序列化代价大	计算中间数据在内存中维护，存取速度是磁盘的多个数量级
Task以进程的方式维护，任务启动就有数秒	Task以线程的方式维护，对小数据集的读取能达到亚秒级的延迟

MapReduce vs Spark



MapReduce vs Spark

On-Disk Sort Record: Time to sort 100TB



http://www.csdn.net/article/2014-10-11/2822041-spark-breaks-previous-large-scale-sort-record

课程大纲

1. Spark 概述、生态系统
2. Spark 编译、测试
3. Scala 基本知识
4. Spark 安装部署
5. Spark 应用提交

Download Spark

http://spark.apache.org/downloads.html

The latest release of Spark is Spark 1.3.1, released on April 17, 2015 (release notes) (git tag)

1. Chose a Spark release: 1.3.1 (Apr 17 2015)
2. Chose a package type: Source Code [can build several Hadoop versions]
3. Chose a download type: Select Apache Mirror
4. Download Spark: spark-1.3.1.tgz
5. Verify this release using the 1.3.1 signatures and checksums.

Spark 1.x编译方式

- SBT 编译
- Maven 编译
- 打包编译make-distribution.sh

Maven安装部署

- 下载地址

  http://maven.apache.org/download.cgi

- 解压

  tar -zvxf apache-maven-3.0.5

- 配置环境变量

  export MAVEN_HOME=/opt/modules/maven-3.0.5

  export PATH=$PATH:$MAVEN_HOME/bin

- 验证

  mvn -version

Spark 1.x源码编译

- 解压

```
tar -zvxf spark-1.3.1
```

- mvn编译

```
mvn clean package \
-DskipTests -Phadoop-2.4 \
-Dhadoop.version=2.5.0 -Pyarn \
-Phive-0.13.1 -Phive-thriftserver
```

- make-distribution编译

```
./make-distribution.sh --tgz \
-Phadoop-2.4 -Dhadoop.version=2.5.0-cdh5.3.6 \
-Pyarn \
-Phive-0.13.1 -Phive-thriftserver
```

Spark 1.x源码编译

SCALA_VERSION=2.10

SPARK_HADOOP_VERSION=2.6.0-cdh5.4.2

SPARK_HIVE=1

SCALA_VERSION配置上你的scala的版本， 可能是2.10 或者2.11

SPARK_HADOOP_VERSION配置上你的hadoop版本

SPARK_HIVE 1表示需要将hive的打包进去， 非1数字表示不打包hive

```
#SPARK_HADOOP_VERSION=$("$MVN" help:evaluate -Dexpression=hadoop.version $@ 2>/dev/null\
#    | grep -v "INFO"\
#    | tail -n 1)
#SPARK_HIVE=$("$MVN" help:evaluate -Dexpression=project.activeProfiles -pl sql/hive $@ 2>/dev/null\
#    | grep -v "INFO"\
#    | fgrep --count "<id>hive</id>";\
#    # Reset exit status to 0, otherwise the script stops here if the last grep finds nothing\
#    # because we use "set -o pipefail"
#    echo -n)
SPARK_HADOOP_VERSION=2.6.0-cdh5.4.0
SPARK_HIVE=1
SCALA_VERSION=2.10.4
```

Spark 1.x源码编译

◆ 配置镜像

```
<mirror>
        <id>nexus-osc</id>
        <mirrorOf>*</mirrorOf>
        <name>Nexus osc</name>
        <url>http://maven.oschina.net/content/groups/public/</url>
</mirror>
```

◆ 配置域名解析服务区

vi /etc/resolv.conf

内容：

nameserver 8.8.8.8

nameserver 8.8.4.4

WAYS TO RUN SPARK

- ✓Local
- ✓Standalone
- ✓YARN
- ✓Mesos

Spark Demo

```
./bin/spark-shell
```

Spark's primary abstraction is a distributed collection of items called a Resilient Distributed Dataset (RDD). RDDs can be created from Hadoop InputFormats (such as HDFS files) or by transforming other RDDs. Let's make a new RDD from the text of the README file in the Spark source directory:

```
scala> val textFile = sc.textFile("README.md")
textFile: spark.RDD[String] = spark.MappedRDD@2ee9b6e3
```

RDDs have *actions*, which return values, and *transformations*, which return pointers to new RDDs. Let's start with a few actions:

```
scala> textFile.count() // Number of items in this RDD
res0: Long = 126

scala> textFile.first() // First item in this RDD
res1: String = # Apache Spark
```

Spark Demo

Now let's use a transformation. We will use the `filter` transformation to return a new RDD with a subset of the items in the file.

```
scala> val linesWithSpark = textFile.filter(line => line.contains("Spark"))
linesWithSpark: spark.RDD[String] = spark.FilteredRDD@7cd4af09
```

We can chain together transformations and actions

```
scala> textFile.filter(line => line.contains("Spark")).count() // How many lines cont
ain "Spark"?
res3: Long = 15
```

课程大纲

1. Spark 概述、生态系统
2. Spark 编译、测试
3. Scala 基本知识
4. Spark 安装部署
5. Spark 应用提交

Scala介绍

- JVM的高层次语言
  - ✓面向对象 + 面向过程（函数式编程）
- 静态类型
  - ✓性能与Java差不多
  - ✓通常不需要显式写出类型（类型推断机制）

Scala 对比 Java

- 定义变量：

```
var x: Int = 6
var x = 6 // 类型推断
val y = “scala” //只读的
```

- 函数：

```
def square(x: Int): Int = x*x
def square(x: Int) = {x*x}
def announce(text: String) {
  println(text)
}
```

- Java等价代码

```
int x = 6;
final String y = “scala”;
int square(int x) {
  return x*x;
}
void announce(String text) {
  System.out.println(text);
}
```

Scala 对比 Java

- 泛型：

```
var arr = new Array[Int](8)

val lst = List(1, 2, 3)
```

- 索引：

```
arr(5) = 7

println(lst(5))
```

- Java等价代码

```
int [] arr =  new int[8];

List<Integer> lst =

 new ArrayList<Interger>();

lst.add(…)

arr[5] = 7;

system.out.println(lst.get(5));
```

Scala 集合操作

Method on Seq[T]	Explanation
`map(f: T => U): Seq[U]`	Pass each element through f
`flatMap(f: T => Seq[U]): Seq[U]`	One-to-many map
`filter(f: T => Boolean): Seq[T]`	Keep elements passing f
`exists(f: T => Boolean): Boolean`	True if one element passes
`forall(f: T => Boolean): Boolean`	True if all elements pass
`reduce(f: (T, T) => T): T`	Merge elements using f
`groupBy(f: T => K): Map[K,List[T]]`	Group elements by f(element)
`sortBy(f: T => K): Seq[T]`	Sort elements by f(element)
`. . .`	

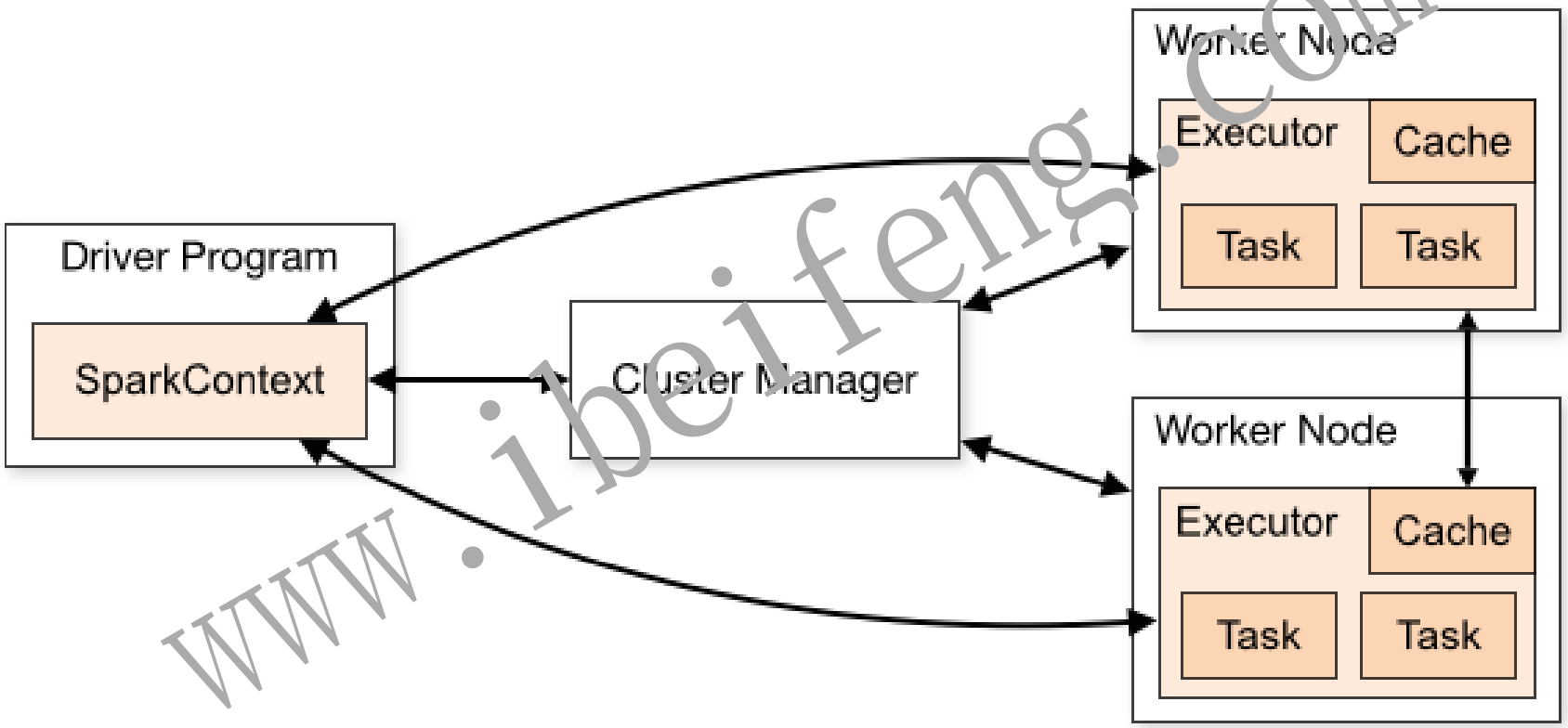
课程大纲

1. Spark 概述、生态系统
2. Spark 编译、测试
3. Scala 基本知识
4. Spark 安装部署
5. Spark 应用提交

WAYS TO RUN SPARK

- ✓Local
- ✓Standalone
- ✓YARN
- ✓Mesos

Cluster Mode



Spark 1.x环境搭建步骤

- 安装JDK（建议JDK 7以上）
- 安装Scala（2.10.4）
- 安装Hadoop 2.x（至少HDFS）
- 安装Spark Standalone

Hadoop 2.x安装部署

◆下载

http://apache.dataguru.cn/hadoop/common/

◆解压

$ tar -zxvf hadoop-2.5.0.tar.gz

◆替换本地库

$ rm -rf ./$HADOOP_HOME/lib/native/

$ cp -r $HADOOP_SRC_HOME/hadoop-dist/target/hadoop-2.5.0/lib/native/* $HADOOP_HOME/lib/native/

◆修改配置文件（$HADOOP_HOME/etc/hadoop/目录下）

hadoop-env.sh、core-site.xml、hdfs-site.xml、yarn-site.xml、mapred-site.xml

◆ 注意点：**native下面的链接文件**

Spark 1.x 搭建部署

- 解压

  tar -zvxf spark-1.3.0-bin-2.5.0

- 配置环境变量

  export SPARK_HOME=/opt/modules/spark-1.3.0-bin-2.5.0

- 配置文件

  spark-env.sh spark-default.conf

- 启动

  start-all.sh

- 验证
  - jps
  - Web UI

Spark Core

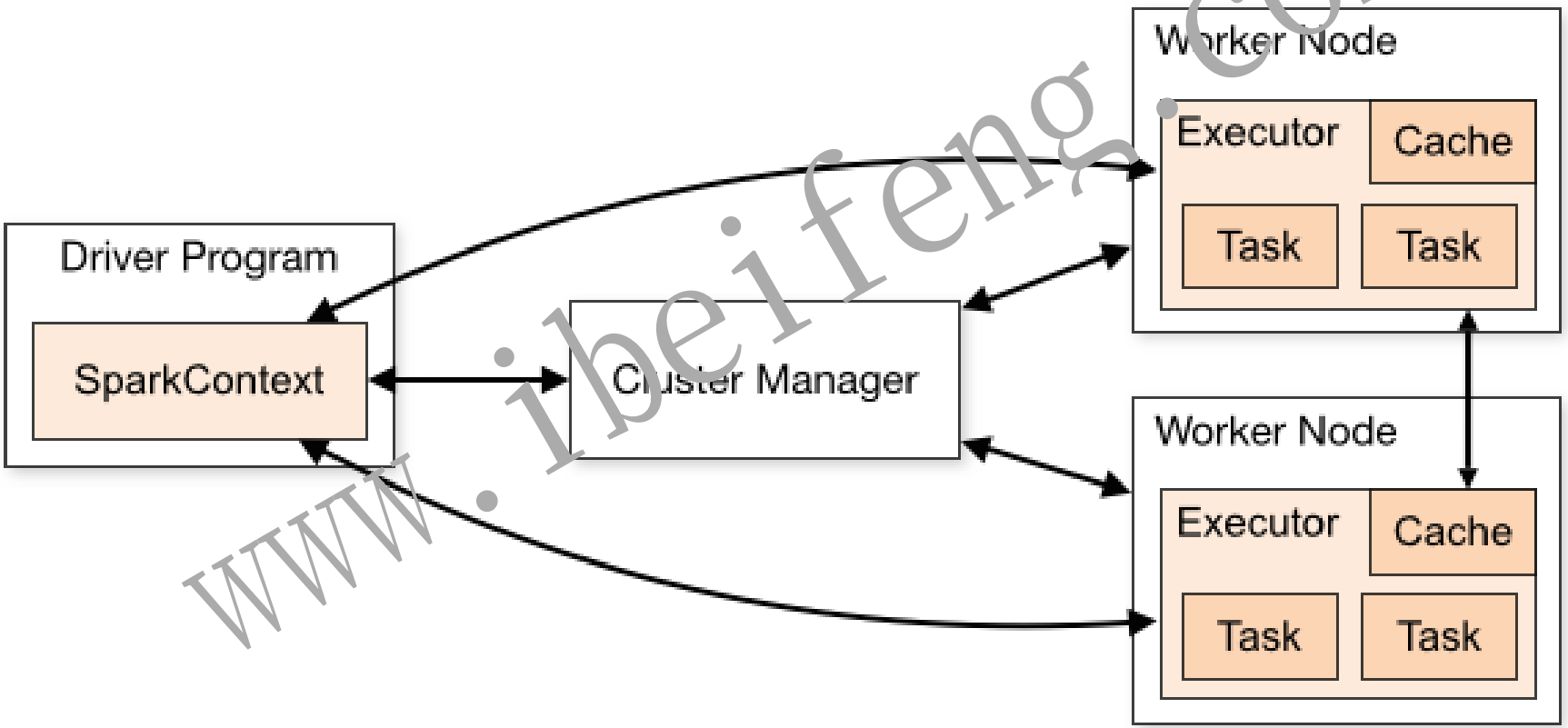
WordCount

```
sc.textFile("data/README.md")
        .map(line => line.split("\t"))
        .map(_,1)
        .reduceByKey(_+_, 3)
        .collect()
```

课程大纲

1. Spark 概述、生态系统
2. Spark 编译、测试
3. Scala 基本知识
4. Spark 安装部署
5. Spark 应用提交

Spark Running Architecture



Cluster Concepts

Term	Meaning
Application	User program built on Spark. Consists of a *driver program* and *executors* on the cluster.
Application jar	A jar containing the user's Spark application. In some cases users will want to create an "uber jar" containing their application along with its dependencies. The user's jar should never include Hadoop or Spark libraries, however, these will be added at runtime.
Driver program	The process running the main() function of the application and creating the SparkContext
Cluster manager	An external service for acquiring resources on the cluster (e.g. standalone manager, Mesos, YARN)
Deploy mode	Distinguishes where the driver process runs. In "cluster" mode, the framework launches the driver inside of the cluster. In "client" mode, the submitter launches the driver outside of the cluster.

Cluster Concepts

Worker node	Any node that can run application code in the cluster
Executor	A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them. Each application has its own executors.
Task	A unit of work that will be sent to one executor
Job	A parallel computation consisting of multiple tasks that gets spawned in response to a Spark action (e.g. `save`, `collect`); you'll see this term used in the driver's logs.
Stage	Each job gets divided into smaller sets of tasks called *stages* that depend on each other (similar to the map and reduce stages in MapReduce); you'll see this term used in the driver's logs.

Spark Running Architecture

1、构建Spark Application运行环境；

在Driver Program中新建SparkContext（包含sparkcontext的程序称为Driver Program）；

Spark Application运行的表现方式为：在集群上运行着一组独立的executor进程，这些进程由sparkcontext来协调；

2、SparkContext向资源管理器申请运行Executor资源，并启动StandaloneExecutorBackend，executor向sparkcontext申请task；

集群通过SparkContext连接到不同的cluster manager（standalone、yarn、mesos），cluster manager为运行应用的Executor分配资源；一旦连接建立之后，Spark每个Application就会获得各个节点上的Executor（进程）；每个Application都有自己独立的executor进程；Executor才是真正运行在WorkNode上的工作进程，它们为应用来计算或者存储数据；

3、SparkContext获取到executor之后，Application的应用代码将会被发送到各个executor；

4、SparkContext构建RDD DAG图，将RDD DAG图分解成Stage DAG图，将Stage提交给TaskScheduler，最后由TaskScheduler将Task发送给Executor运行；

5、Task在Executor上运行，运行完毕后释放所有资源；

Launching Applications with spark-submit

```
./bin/spark-submit \
  --class <main-class>
  --master <master-url> \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

Some of the commonly used options are:

- `--class`: The entry point for your application (e.g. `org.apache.spark.examples.SparkPi`)
- `--master`: The master URL for the cluster (e.g. `spark://23.195.26.187:7077`)
- `--deploy-mode`: Whether to deploy your driver on the worker nodes (`cluster`) or locally as an external client (`client`) (default: `client`) †
- `--conf`: Arbitrary Spark configuration property in key=value format. For values that contain spaces wrap "key=value" in quotes (as shown).
- `application-jar`: Path to a bundled jar including your application and all dependencies. The URL must be globally visible inside of your cluster, for instance, an `hdfs://` path or a `file://` path that is present on all nodes.
- `application-arguments`: Arguments passed to the main method of your main class, if any

Master URLs

Master URL	Meaning
local	Run Spark locally with one worker thread (i.e. no parallelism at all).
local[K]	Run Spark locally with K worker threads (ideally, set this to the number of cores on your machine).
local[*]	Run Spark locally with as many worker threads as logical cores on your machine.
spark://HOST:PORT	Connect to the given Spark standalone cluster master. The port must be whichever one your master is configured to use, which is 7077 by default.
mesos://HOST:PORT	Connect to the given Mesos cluster. The port must be whichever one your is configured to use, which is 5050 by default. Or, for a Mesos cluster using ZooKeeper, use `mesos://zk://....`
yarn-client	Connect to a YARN cluster in client mode. The cluster location will be found based on the HADOOP_CONF_DIR variable.
yarn-cluster	Connect to a YARN cluster in cluster mode. The cluster location will be found based on HADOOP_CONF_DIR.

本课程版权归北风网所有

欢迎访问我们的官方网站

www.ibeifeng.com