一、选择题

1. 下列下列说法错误的是（ D ）。

A. 从语义保障来说，Storm 可以做到 at least once

B. 从语义保障来说，Spark Streaming 可以做到 exactly once

C. 从语义保障来说，Flink 可以做到 exactly once/at least once

D. 从语义保障来说，Spark Streaming 可以做到 exactly once/at least once

2. 下列关于 Zookeeper 命令行操作说法错误的是（ D ）。

A. Zookeeper集群的启动,需要在每个Zookeeper的节点上执行zkServer.sh start。

B. Zookeeper集群的停止,需要在每个Zookeeper的节点上执行zkServer.sh stop。

C. Zookeeper 集群的状态查看，需要在每个 Zookeeper 的节点上执行 zkServer.sh status。

D. Zookeeper 集群的进程，是不能通过 jps 指令查看到。

3. 下列 Kafka 集群说法错误的是（ D ）。

A. Zookeeper 启动的进程名为 QuorumPeerMain

B. Kafka 启动的进程名为 Kafka

C. Kafka 集群的启动，需要在每个 Kafka 节点上执行 bin/kafka-server-start.sh -daemon config/server.properties

D. Kafka 集群的停止，需要在每个 Kafka 节点上执行 bin/kafka-server-stop.sh

4. 下列关于 Kafka 的架构说法错误的是（ D ）。

A、Kafka 的主题可以有多个分区，每个分区可以分布在不同的机器。

B、Kafka 的分区可以有多个副本因子，每个副本可以分布在不同的机器。

C、Kafka 的多个分区划分为一个 leader 和多个 follower。

D、Kafka 的架构是点对点的模式。

5.下列关于 Storm 与 Hadoop 的区别说法错误的是（ A ）。

A、Storm 用于离线计算，Hadoop 用于实时计算。

B、Storm 的数据通过网络传输进来；Hadoop 的数据保存在磁盘中。

C、Storm 与 Hadoop 的编程模型相似。

D、Storm 处理的数据保存在内存中，源源不断；Hadoop 处理的数据保存在文件系统 中，一批一批处理。

6.下列关于 Storm Bolt 的程序编写说法错误的是（ B ）。

A、编写的 Bolt 类通常继承于 BaseRichBolt

B、OutputFieldsDeclarer 声明发射的属性只能是一个。

C、OutputCollector 一次发送的字段值可以是一个或多个，需要与 OutputFieldsDeclarer 声明发射的属性个数相对应。

D、prepare 方法通常完成初始化功能。

7.在 Storm 中，下面哪个是传输数据的基本单位（ A ）。

A. Tuple　　B. Stream

C. Component　　D. Task

8.Spark 是一个（ D ）计算引擎。

A. MapReduce　　B. In-Memory

C. Distributed　　D. All of the above

9.在 Spark Streaming 中，数据被切分成(A)时间窗口进行处理？

A. 微批次 B. 实时流 C. 批处理 D. 分布式计算

10.Flink 是一种(C)类型的分布式计算系统？

A. Hadoop　　B. Spark
C. Stream Processing　　D. Batch Processing

11.下列计算框架延迟最高的是(D)
A. Storm　　B. Spark Streaming
C. Flink　　D. Hadoop

12. 下列不是Flume Agent 的主要组成部分的是(D)
A. Source　　B. Channel
C. Sink　　D. Cloud

13. 关于bin/kafka-topics.sh --zookeeper master:2181,slave1:2181 --create --replication-factor 3 --partitions 1 --topic first 下列说法错误的是(C)
A. 该指令表示要创建要在kafka中创建名称为first的主题
B. 该指定表示使用Zookeeper连接Kafka
C. 该指令表示要创建的分区数量为3
D. 该指令表示使用Zookeeper连接Kafka，先使用 master:2181的Zookeeper连接，　　当连接不成功时，再使用slave1:2181的Zookeeper连接。

14. 下列哪项不是Storm的应用场景(D)
A. 每年双十一时，淘宝和京东实时订单销售额和产品数量大屏展示。
B. 通过工业数据分析设备运行状况和稳健程度，展示工件完成情况，运行情况。
C. 集群监控：一般的大型集群和平台，都需要对其进行监控，监控的需求。
D. 批量分析一个月的销售额数据。

15. 下列关于Storm特点说法错误的是(B)
A. Storm可以适用实时处理消息、更新数据库、持续计算等场景
B. Storm无法保证所有的数据都被处理
C. 在消息处理过程中出现异常，Storm会进行重试
D. Storm集群非常容易管理，轮流重启节点不影响应用

16. 下列哪项是Storm Spout向下游发射数据的方法(C)
A. open()　　B. declareOutputFields()
C. nextTuple()　　D. execute()

17. Spark是一个（ D ）计算引擎。
A. MapReduce　　B. In-Memory
C. Distributed　　D. All of the above

18. 在Spark Streaming中，（ A ）算子将DStream中的元素映射到新的DStream中?
A. map()　　B. filter()
C. reduceByKey()　　D. join()

19. Spark Structured Streaming支持（ B ）型数据源。
A. 半结构化　　B. 结构化
C. 非结构化　　D. 半非结构化

20. 下列哪个不是DataStream API中的时间概念（ D ）?
A. event time　　B. processing time
C. ingestion time　　D. execution time

21.下列说法错误的是（ D ）。
A. Flume是Cloudera 提供的一个高可用的，高可靠的，分布式系统

B. Flume 基于流式架构，灵活简单
C. 海量日志采集、聚合和传输的系统
D. 业务数据库采集分析和可视化的系统

22.下列说法错误的是（ D ）。
A. 从延迟来说 Storm 可以达到毫秒级
B. 从延迟来说 Spark Streaming 可以达到秒级
C. 从延迟来说 Flink 可以达到毫秒级
D. 从延迟来说 Spark Streaming 可以达到秒级和毫秒级

23.关于批处理下列说法错误的是（ B ）。
A. 属于静态数据
B. 批大小无界
C. 访问全部数据
D. 高效、易于管理

24.关于实时处理下列说法错误的是（ D ）。
A. 属于动态数据
B. 数据以流的形式存在，是无界的
C. 数据在时间、窗口或者微批级别上处理
D. 实时分析，但是系统相对于批处理较为健壮

25.在过去的几年中，下列哪一项不是大数据应用程序的发展过程（ C ）。
A. 仅批处理应用程序（早期的 Hadoop 实现）
B. 仅流处理（早期的 Storm 实现）
C. 仅流处理（Spark 实现）
D. Lambda 架构

26.下列关于 Zookeeper 命令行操作说法错误的是（ D ）。
A. Zookeeper 集群的启动,需要在每个 Zookeeper 的节点上执行 zkServer.sh start。
B. Zookeeper 集群的停止,需要在每个 Zookeeper 的节点上执行 zkServer.sh stop。
C. Zookeeper 集群的状态查看，需要在每个 Zookeeper 的节点上执行 zkServer.sh status。
D. Zookeeper 集群的进程，是不能通过 jps 指令查看到。

27. 下列关于 Kafka 的架构说法错误的是（ D ）。
A. Kafka 的主题可以有多个分区，每个分区可以分布在不同的机器。
B. Kafka 的分区可以有多个副本因子，每个副本可以分布在不同的机器。
C. Kafka 的多个分区划分为一个 leader 和多个 follower。
D. Kafka 的架构是点对点的模式。

28. 下列关于 Bolt 的程序编写说法错误的是（ B ）。
A. 编写的 Bolt 类通常继承于 BaseRichBolt
B. OutputFieldsDeclarer 声明发射的属性只能是一个。
C. OutputCollector 一次发送的字段值可以是一个或多个，需要与 OutputFieldsDeclarer 声明发射的属性个数相对应。
D、prepare 方法通常完成初始化功能。

29. Spark 中的 RDD 指的是（B ）。
A. Random Data Distribution　B. Resilient Distributed Dataset
C. Recent Data Distribution　D. None of the above

30. Spark Streaming 中，数据被切分成哪个时间窗口进行处理（ A ）？

A. 微批次　　B. 实时流
C. 批处理　　D. 分布式计算

31. 下列关于 Flume 说法错误的是（　D ）。
A. Flume 是 Cloudera 提供的一个高可用的，高可靠的，分布式系统
B. Flume 基于流式架构，灵活简单
C. 海量日志采集、聚合和传输的系统
D. 业务数据库采集分析和可视化的系统

32. 下列关于实时计算框架说法错误的是（ D ）。
A. 从延迟来说 Storm 可以达到毫秒级
B. 从延迟来说 Spark Streaming 可以达到秒级
C. 从延迟来说 Flink 可以达到毫秒级
D. 从延迟来说 Spark Streaming 可以达到秒级和毫秒级

33. 下列关于语义保障说法错误的是（ D ）。
A. 从语义保障来说，Storm 可以做到 at least once
B. 从语义保障来说，Spark Streaming 可以做到 exactly once
C. 从语义保障来说，Flink 可以做到 exactly once/at least once
D. 从语义保障来说，Spark Streaming 可以做到 exactly once/at least once

34. 下列说法错误的是（ C ）。
A. 每月广告的活跃用户相关度分析属于批处理，时间大于 1h
B. 广告印象数标签趋势属于在线处理，时间大于 1s
C. 确定性的工作流程如查找推文属于批处理，时间小于 500ms
D. 金融交易属于近实时处理，时效小于 1ms

35. 关于批处理下列说法错误的是（B　）。
A. 属于静态数据
B. 批大小无界
C. 访问全部数据
D. 高效、易于管理

36. 关于实时处理下列说法错误的是（ D ）。
A. 属于动态数据
B. 数据以流的形式存在，是无界的
C. 数据在时间、窗口或者微批级别上处理
D. 实时分析系统相对于批处理较为健壮

37. 从近实时解决方案在较高层面看，不是基本分析过程的是（ A ）。
A. 流数据的生成
B. 分布式流数据的高性能计算
C. 以可查询消耗层的形式探索和可视化
D. 仪表板的形式探索和可视化

38. 下列哪一项不是以流的形式处理数据（ B ）。
A. 计算　　B. 存储
C. 转换　　D. 处理和增强

39. 在过去的几年中，下列哪一项不是大数据应用程序的发展过程（ C ）。
A. 仅批处理应用程序（早期的 Hadoop 实现）
B. 仅流处理（早期的 Storm 实现）
C. 仅流处理（Spark 实现）

D. Lambda 架构

40. 关于流处理和批处理下列说法错误的是（ D ）。

A. Hadoop 可以构建几乎处理任何数据量的应用程序

B. Storm 可扩展性强，且具有轻量级快速处理能力

C. Hadoop 满足了对数据量的需求，Storm 在速度方面非常出色

D. Storm 在整个数据集的短时窗口上变现的很好，并且在以后的某个时间有修正数

据的机制。

41.下列计算框架延迟最高的是(D)

A. Storm　　B. Spark Streaming

C. Flink　　D. Hadoop

42.下列不是 Flume Agent 的主要组成部分的是(D)

A. Source　　B. Channel

C. Sink　　D. Cloud

43.下列下列说法错误的是（ D ）。

A. 从语义保障来说，Storm 可以做到 at least once

B. 从语义保障来说，Spark Streaming 可以做到 exactly once

C. 从语义保障来说，Flink 可以做到 exactly once/at least once

D. 从语义保障来说，Spark Streaming 可以做到 exactly once/at least once

44.下列说法错误的是（ D ）。

A. 从延迟来说 Storm 可以达到毫秒级

B. 从延迟来说 Spark Streaming 可以达到秒级

C. 从延迟来说 Flink 可以达到毫秒级

D. 从延迟来说 Spark Streaming 可以达到秒级和毫秒级

45.下列关于 Kafka 的架构说法错误的是（ D ）。

A. Kafka 的主题可以有多个分区，每个分区可以分布在不同的机器。

B. Kafka 的分区可以有多个副本因子，每个副本可以分布在不同的机器。

C. Kafka 的多个分区划分为一个 leader 和多个 follower。

D. Kafka 的架构是点对点的模式。

46. Apache Spark 支持哪种语言（ D ）。

A. Java　　B. Python

C. Scala　　D. All of the above

47. Spark 中的 RDD 指的是（ B ）。

A. Random Data Distribution　B. Resilient Distributed Dataset

C. Recent Data Distribution　　D. None of the above

48. Spark Streaming 支持哪些类型的数据源（ D ）。

A. Hadoop Distributed File System（HDFS）　　B. Kafka

C. Flume　　D. 所有以上的数据源

49. Spark Streaming 中，数据被切分成哪个时间窗口进行处理（ A ）。

A. 微批次　　B. 实时流

C. 批处理　　D. 分布式计算

50. 下列哪个不是 Flink 流处理框架的核心特性（ D ）？

A. 支持高吞吐低延迟　　B. 支持有状态流处理

C. 支持 Exactly-Once 语义　　D. 支持广播流

二、填空题

1. Kafka 是一个分布式的基于发布/订阅模式的消息队列（Message Queue），主要应用于大数据 实时处理 领域。

2. Kafka 中消息是以 topic 进行分类的，生产者生产消息，消费者消费消息，都是面向 topic 的。

3. Storm框架中， Spout 产生数据源的组件，从外部获取数据，输出原始Tuple。

4. Storm 框架中， Bolt 接收 Spout/Bolt 输出的 Tuple，处理，输出新 Tuple。

5. 在 Storm 中， 可靠的消息处理机制是从 Spout 开始的。

6. 在 Shuffle Grouping 中，Tuple 将 随机 均匀地分发到下游 Bolt 的 所有任务中。

7. Spark 中，数据被存储在一种名为 RDD 的分布式数据集中。

8. RDD 具备 弹性容错性 优化能力，可避免数据丢失和降低计算成本。

9. Spark Streaming 中， DStream 组件用于表示连续的数据流。

10. Flink 可以通过 flink-conf.yaml 文件、环境变量和命令行参数的方式进行参数配置。

11. 流处理框架的演进过程大致可以分为四个阶段，分别是实时阶段、Lambda 架构阶段、 微批处理阶段 、全能阶段。

12. 消息队列的两种模式分别是 点对点模式 、发布/订阅模式 。

13. Storm 架构中 Nimbus 负责资源分配和任务调度。

14. . Storm 架构中 Supervisor 负责接受 nimbus 分配的任务，启动和停止属于自己管理的 worker 进程。

15. Storm 定义了七种内置数据流分组的方式， Fields grouping 方式会根据指定字段的值进行分组。所有具有相同字段值的 tuple 会路由到同一个 bolt 的 task 中。

16. Storm 中， Spout 负责从数据源读取数据，并将其发送到下一个组件。

17. Spark 的集群管理器有三种类型，分别是 Standalone 、YARN 和 Mesos。

18. 在 Spark 中，使用 spark-shell 命令可以启动交互式 Shell。

19. 在 Spark Streaming 中， DStream 组件用于表示连续的数据流。

20. Flink 的启动脚本是 start-cluster.sh 。

21. Kafka 中消息是以 topic 进行分类的，生产者生产消息，消费者消费消息，都是面向 topic 的。

22. Storm 架构中 Worker 负责运行具体处理组件逻辑的进程。

23. Spout 产生数据源的组件，从外部获取数据，输出原始 Tuple。

24. Bolt 接收 Spout/Bolt 输出的 Tuple，处理，输出新 Tuple。

25. Storm 定义了七种内置数据流分组的方式， Globle grouping 方式这种分组方式将所有的 tuples 路由到唯一一个 task 上。Storm 按照最小的 task ID 来选取接收数据的 task。

26. RDD 具备 弹性容错性 优化能力，可避免数据丢失和降低计算成本。

27. 在 Spark Streaming 中， filter 算子用于对 DStream 中的元素进行过滤操作。

28. Spark Structured Streaming 中， groupBy 算子用于对 Dataset 进行分组操作。

29. Flink 的默认 Web UI 端口是 8080 。

30. Flink standalone 模式的停止脚本是 stop-cluster.sh 。

31.Kafka 是一个分布式的基于 发布/订阅 模式的消息队列（Message Queue），主要应用于大数据实时处理领域。

32.Kafka 中消息是以 topic 进行分类的，生产者生产消息，消费者消费消息，都是面向 topic 的。

33.Storm 架构中 Nimbus 负责资源分配和任务调度。

34.Storm 架构中 Supervisor 负责接受 nimbus 分配的任务，启动和停止属于自己管理的

worker 进程。

35.Storm 架构中 Worker 负责运行具体处理组件逻辑的进程。

36. Tuple 是消息传递的基本单元，由多个 Field 组成。

37.Spark 中，数据被存储在一种名为 RDD 的分布式数据集中。

38.Spark 集群管理器有三种类型，分别是 Standalone、YARN 和 Mesos。

39.Spark Streaming 中， DStream 组件用于表示连续的数据流。

40.Spark Structured Streaming 支持 结构化 型数据源。

41.KAFKA topic 是逻辑上的概念，而 partition 是物理上的概念，每个 partition 对应于一个 log 文件。

42.Kafka 生产者 API 需要用到的类，每条数据都要封装成一个 ProducerRecord 对象。

43.Spark 中，最基本的数据抽象是 RDD 。

44.Spark 使用的默认集群管理器是 Standalone 。

45.Spark 中，用于执行机器学习任务的组件是 Spark MLlib 。

46.Spark Streaming 中， DStream 组件用于表示连续的数据流。

47.Spark Streaming 中， countByWindow 算子用于返回滑动时间窗口内的元素数量。

48.Spark Structured Streaming 支持 结构化 型数据源。

49.Flink 流处理框架中，数据处理的基本单位是 DataStream 。

50.Flink 流处理框架中的时间概念包括 Event Time、Processing Time、Ingestion Time。

三、判断题

1. Flume Header 用来存放该 event 的一些属性，为 K-V 结构。(√)

2. Flume Body 用来存放该条数据，形式为字节数组。(√)

3. 从微信每次发送一行文字，可以理解为实时处理，每次发送一段文字可以理解为批处理。(√)

4. Storm 是第一代实时计算框架、Flink 是目前最新的实时计算框架。(√)

5. Lambda 架构是将实时计算和批处理计算模式集成在一起的一种计算。(√)

6. 批数据处理是处理的静态数据，数据批大小是无界的。(×)

7. 实时处理是动态数据，数据以流的形式存在，是有界的。(×)

8. Spark Streaming 是以微批处理方式处理数据，延迟性相对 storm 较高。(√)

9. Flink 是目前相对比较完善和易用的实时计算框架。(√)

10.Kafka 的主题可以理解为一个消息队列，生产者生产消息放入消息队列，消费者从消息队列中消费。(√)

11.通用的数据处理流程是从数据源采集和传输（flume、kafka)、再进行数据处理（批处理、流处理)、数据存储（Mysql、redis)、最后数据应用（数据大屏、实时看板)。（ √ ）

12.Flume 传输的基本单元是 Event，有 Header 和 Body 量部分组成。（ √ ）

13.Kafka 消息队列是发布订阅模式，生产者生产消息可以被多个消费者订阅。（ √ ）

14.Kafka 的同一个消费者组内的每个消费者消费同一个主题的不同分区。（ √ ）

15.Nimbus 负责资源分配和任务调度及任务的运行。（× ）

16.负责接受 nimbus 分配的任务，启动和停止属于自己管理的 worker 的进程是 Executor。

（ × ）

17.Storm 的全局分组是将所有的 tuple 复制后分发到所有的 bolt task。每个订阅数据流的 task 都会接收到 tuple 的拷贝。（ × ）

18.使用 distinct()函数可以对 RDD 中的元素进行去重。（ √ ）

19.Spark 中，用于将流式数据进行处理和转换的组件是 Spark Streaming。(√)

20.Flink 流处理框架中，数据处理的基本单位是 DataStream。（ √ ）

21.Flume Header 用来存放该 event 的一些属性，为 K-V 结构。（ √ ）

22.从微信每次发送一行文字，可以理解为实时处理，每次发送一段文字可以理解为批处理。（ √ ）

23.Lambda 架构是将实时计算和批处理计算模式集成在一起的一种计算。（ √ ）

24.批数据处理是处理的静态数据，数据批大小是无界的。（ × ）

25.实时处理是动态数据，数据以流的形式存在，是有界的。（ × ）

26.Spark Streaming 是以微批处理方式处理数据，延迟性相对 storm 较高。（ √ ）

27.Flink 是目前相对比较完善和易用的实时计算框架。（ √ ）

28.Storm 是第一代实时计算框架、Flink 是目前最新的实时计算框架。（ √ ）

29.kafka 将每个 partition 分为多个 segment。每个 segment 对应两个文件——".log"文件和".data"文件。（ × ）

30.Flink 是一种 Stream Processing 类型的分布式计算系统。（ √ ）

31.通用的数据处理流程是从数据源采集和传输（flume、kafka)、再进行数据处理（批处理、流处理)、数据存储（Mysql、redis)、最后数据应用（数据大屏、实时看板)。（√ ）

32.Flume 传输的基本单元是 Event，有 Header 和 Body 量部分组成。（√ ）

33.从微信每次发送一行文字，可以理解为实时处理，每次发送一段文字可以理解为批处理。（ √ ）

34.Storm 是第一代实时计算框架、Flink 是目前最新的实时计算框架。（ √ ）

35.Lambda 架构是将实时计算和批处理计算模式集成在一起的一种计算。（√ ）

36.Storm 近实时的解决方案是：通过发送/提取收集代理(如 Flume、Logstash)、可以从不同数据源收集实时流数据。然后数据被写入 kafka 分区，Storm 拓扑从 Kafka 中提取/读取流数据并在拓扑中处理数据。（√ ）

37.批数据处理是处理的静态数据，数据批大小是无界的。（× ）

38.实时处理是动态数据，数据以流的形式存在，是有界的。（× ）

39.Spark Streaming 是以微批处理方式处理数据，延迟性相对 storm 较高。（ √ ）

40.Flink 是目前相对比较完善和易用的实时计算框架。（√ ）

41.Storm Nimbus 负责资源分配和任务调度及任务的运行。（× ）

42. 负责接受 nimbus 分配的任务，启动和停止属于自己管理的 worker 的进程是

Executor。(×)

43. 产生数据源的组件,从外部获取数据,storm输出原始Tuple的组件是Bolt。(×)

44. 接收 Spout/Bolt 输出的 Tuple，处理，输出新 Tuple 的组件是 Spout。(×)

45. 消息传递的基本单元，一个 Tuple 由多个 Field 组成，该组件是 stream。(×)

46. 批数据处理是处理的静态数据，数据批大小是无界的。(×)

47.实时处理是动态数据，数据以流的形式存在，是有界的。(√)

48.Spark Streaming 是以微批处理方式处理数据，延迟性相对 storm 较高。(√)

49.Flink 是目前相对比较完善和易用的实时计算框架。(√)

50.Lambda 架构是将实时计算和批处理计算模式集成在一起的一种计算。(√)

四、简答题

1.请简述消息传递语义?

1) 最多一次：消息立即传输。如果传输陈宫，消息将不再发送过去。但是，许多失败情况都会导致消息丢失。

2) 至少一次：每条消息至少传输一次。在失败的情况下，消息可能会被传输两次。

3) 只有一次：每条消息只传输一次。

2.请简述 Kafka 文件存储基本结构?

1) 在 Kafka 文件存储中，同一个 topic 下有多个不同 partition，每个 partition 为一个目录，partiton 命名规则为 topic 名称+有序序号，第一个 partiton 序号从 0 开始，序号最大值为 partitions 数量减 1。

2) 每个 partion(目录)相当于一个巨型文件被平均分配到多个大小相等 segment(段)数据文件中。但每个段 segment file 消息数量不一定相等，这种特性方便 old segment file 快速被删除。默认保留 7 天的数据。

3) 每个 partiton 只需要支持顺序读写就行了，segment 文件生命周期由服务端配置参数决定。

3.请简述 Storm Spout 的可靠性?

1) 在 Storm 中， 可靠的消息处理机制是从 spout 开始的。 一个提供了可靠的处理机制的 spout 需要记录它发射出去的 tuple ，当下游 bolt 处理 tuple 或者子 tuple 失败时 spout 能够重新发射。

2) 在有保障数据的处理过程中， bolt 每收到一个 tuple， 都需要向上游确认应答（ack）或者报错。

3) 对主干 tuple 中的一个 tuple,如果 tuple 树上的每个 bolt 进行了确认应答，spout 会调用 ack 方法来标明这条消息已经完全处理了。如果树中任何一个 bolt 处理 tuple 报错， 或者处理超时，spout 会调用 fail 方法。

4.请简述 Spark 的特点？

1） 速度快：由于 Apache Spark 支持内存计算，并且通过 DAG（有向无环图）执行引擎支持无环数据流。

2） 易于使用：Spark 支持了包括 Java、Scala、Python 、R 和 SQL 语言在内的多种语言。

3） 通用性强：在 Spark 的基础上，Spark 还提供了包括 Spark SQL、Spark Streaming、MLib 及 GraphX 在内的多个工具库。

4） 多种运行方式：Spark 支持多种运行方式，包括在 Hadoop 和 Mesos 上，也支持 Standalone 的独立运行模式，同时也可以运行在云 Kubernetes 上。

5.Spark 的主要组件有哪些，并具体阐述？

1） Spark SQL：Spark 用来操作结构化数据的程序包。通过 Spark SQL，我们可以使用 SQL 操作数据。

2） Spark Streaming：Spark 提供的对实时数据进行流式计算的组件。提供了用来操作数据流的 API。

3） Spark MLlib：提供常见的机器学习(ML)功能的程序库。包括分类、回归、聚类、协同过滤等，还提供了模型评估、数据导入等额外的支持功能。

4） Spark GraphX：Spark 中用于图计算的 API，性能良好，拥有丰富的功能和运算符，能在海量数据上自如地运行复杂的图算法。

6.简述 Flume Agent 包含的三个重要组成部分？

1） Source 是负责接收数据到 Flume Agent 的组件。Source 组件可以处理各种类型、各种 格式的日志数据；

2） Sink 不断地轮询 Channel 中的事件且批量地移除它们，并将这些事件批量写入到存储 或索引系统、或者被发送到另一个 Flume Agent；

3） Channel 是位于 Source 和 Sink 之间的缓冲区。因此，Channel 允许 Source 和 Sink 运行在不同的速率上。Channel 是线程安全的，可以同时处理几个 Source 的写入操作和几个 Sink 的读取操作。

7.阐述消息队列的好处？

1） 解耦：允许你独立的扩展或修改两边的处理过程，只要确保它们遵守同样的接口约束。

2） 可恢复性：系统的一部分组件失效时，不会影响到整个系统。

3) 缓冲：有助于控制和优化数据流经过系统的速度，解决生产消息和消费消息的处理速度不一致的情况。

4) 灵活性 & 峰值处理能力：在访问量剧增的情况下，应用仍然需要继续发挥作用。

5) 异步通信：消息队列提供了异步处理机制，允许用户把一个消息放入队列，但并不立即处理它。在需要的时候再去处理它们。

8.简述 Storm 特点？

1) 适用场景广泛：Storm 可以适用实时处理消息、更新数据库、持续计算等场景。

2) 可伸缩性高：Storm 的可伸缩性可以让 Storm 每秒处理的消息量达到很高。扩展一个实时计算任务，你所需要做的就是加机器并且提高这个计算任务的并行度。Storm 使用 Zookeeper 来协调机器内的各种配置使得 Storm 的集群可以很容易的扩展。

3) 保证无数据丢失：Storm 保证所有的数据都被处理。

4) 异常健壮：Storm 集群非常容易管理，轮流重启节点不影响应用。

5) 容错性好：在消息处理过程中出现异常，Storm 会进行重试。

9.简述 Storm 与 Hadoop 的区别？

1) Storm 用于实时计算，Hadoop 用于离线计算。

2) Storm 处理的数据保存在内存中，源源不断；Hadoop 处理的数据保存在文件系统中，一批一批处理。

3) Storm 的数据通过网络传输进来；Hadoop 的数据保存在磁盘中。

4) Storm 与 Hadoop 的编程模型相似。

10.简述 Spark 的主要组件？

整个 Spark 框架模块包含：Spark Coke、Spark SQL、Spark Streaming、Spark GraphX、Spark MLlib。

1) Spark Core：实现了 Spark 的基本功能。

2) Spark SQL：Spark 用来操作结构化数据的程序包。

3) Spark Streaming：Spark 提供的对实时数据进行流式计算的组件。

4) Spark MLlib：提供常见的机器学习(ML)功能的程序库。

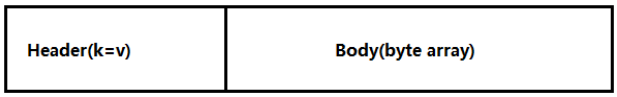
5) Spark GraphX：Spark 中用于图计算的 API，性能良好，拥有丰富的功能和运算符，能在海量数据上自如地运行复杂的图算法。

11.请简述 Flume 的 Event 结构？

1) Flume 数据传输的基本单元，以 Event 的形式将数据从源头送至目的地；

2) Event 由 Header 和 Body 两部分组成，Header 用来存放该 event 的一些属性，为 K-V 结构， Body 用来存放该条数据，形式为字节数组

3) 结构图如下：



12.请简述 Kafka Producer 消息发送的应答机制？

1) 设置发送数据是否需要服务端的反馈,有三个值 0,1,-1

2) 0: producer 不会等待 broker 发送 ack

3) 1: 当 leader 接收到消息之后发送 ack

4) -1: 当所有的 follower 都同步消息成功后发送 ack

13.请请简述 bolt 要实现可靠的消息处理机制包含的两个步骤？

1) 当发射衍生的 tuple 时， 需要锚定读入的 tuple

collector emit(tuple ， new Values(word));

非锚定： collector emit(new Values(word));

2) 当处理消息成功或者失败时分别确认应答或者报错。

this.collector.ack(tuple);

this.collector.fail(tuple);

14.请简述 Storm 中的 Grouping 方式有哪些？

1) Storm 中的 Grouping 方式包括 Shuffle Grouping、Fields Grouping、All Grouping

2) Global Grouping、Partial Key Grouping、Direct Grouping

3) 不同的 Grouping 方式适合不同的数据处理场景，可以根据具体需求进行选择

15.请简述 Spark Streaming？

1) Spark Streaming 是 Spark 生态系统中的组件之一，支持实时流数据处理，将实时数据流划分为一段段的微批次，具有高吞吐量、低延迟等特点。

2) 其核心概念包括 DStream、Receiver、Transformation、Output，其中 DStream 表示数据流，Receiver 用于数据输入，Transformation 用于数据处理，Output 用于数据输出。

3) Spark Streaming 支持多种数据源类型和数据输出类型，可以灵活地处理各种

实时数据场景。

五、编程题

1.使用 Flink 批处理程序完成从文件中读取文本的每一行内容，进行分词，统计每个单词出现的次数。

```
public class BatchWordCount {
    public static void main(String[] args) throws Exception {
        // 1. 创建执行环境
        ExecutionEnvironment env =
            ExecutionEnvironment.getExecutionEnvironment();
        // 2. 从文件读取数据 按行读取(存储的元素就是每行的文本)
        DataSource<String> lineDS = env.readTextFile("input/words.txt");
        // 3. 转换数据格式
        FlatMapOperator<String, Tuple2<String, Long>> wordAndOne =
        lineDS.flatMap(new FlatMapFunction<String, Tuple2<String, Long>>() {
            @Override
            public void flatMap(String line, Collector<Tuple2<String, Long>>
                out) throws Exception {
                //将字符串按空格切分成单词，放入数组 words
                String[] words=line.split(" ");
                for(String word:words) {
                    //将每个单词转换成（单词,1）格式
                    out.collect(Tuple2.of(word,1L));
                }
            }
        });
        // 4. 按照 word 进行分组
        UnsortedGrouping <Tuple2 < String,Long>>
                wordAndOneUG=wordAndOne.groupBy(0);
        // 5. 分组内聚合统计
        AggregateOperator<Tuple2<String, Long>> sum =
                wordAndOneUG.sum(1);
        // 6. 打印结果
        sum.print();
    }
}
```

2.使用 Flink 流处理程序完成从文件中读取文本的每一行内容，进行分词，统计每个单词出现的次数。

```
public class BoundedStreamWordCount {
        public static void main(String[] args)throws Exception {
            StreamExecutionEnvironment env=
                    StreamExecutionEnvironment.getExecutionEnvironment();
```

```
        // 2. 读取文件
         DataStreamSource<String> lineDSS =
                env.readTextFile("input/words.txt");
        // 3. 转换数据格式
        SingleOutputStreamOperator<Tuple2<String, Long>>  wordAndOne =
            lineDSS.flatMap(new FlatMapFunction<String, Tuple2<String,
            Long>>() {
                 @Override
                    public void flatMap(String line,
                        Collector<Tuple2<String,
                        Long>> out) throws Exception {
                             String[] words=line.split(" ");
                             for(String word:words) {
                                 out.collect(Tuple2.of(word,1L));
                             }
                    }
            });
         // 4. 分 组
           KeyedStream<Tuple2<String, Long>, String> wordAndOneKS =
                    wordAndOne.keyBy(t->t.f0);
         // 5. 求 和
           SingleOutputStreamOperator<Tuple2<String, Long>> result =
                        wordAndOneKS.sum(1);
         // 6. 打 印
          result.print();
         // 7. 执 行
           env.execute()
    }

}
```

3.编写 Spark Streaming 实时计算程序：接收 node01 主机 9999 发送过来的数据，并统计每个单词出现的次数。客户端会向 node01 主机 9999 端口不断的发送一个个句子。

```
object StreamingWordCount {
   def main(args: Array[String]): Unit = {
   //创建初始化配置对象
    val sparkConf = new SparkConf().setAppName("streaming word count")
                                .setMaster("local[2]")
   // StreamingContext 其实就是 Spark Streaming 的入口
   val ssc = new StreamingContext(sparkConf, Seconds(5))
   //设置日志输出级别
   ssc.sparkContext.setLogLevel("WARN")
   //接收 node01 主机 9999 端口发送过来的数据
   val lines: ReceiverInputDStream[String] = ssc.socketTextStream(
```

```
                                        hostname = "node01",
                                        port = 9999,
                storageLevel = StorageLevel.MEMORY_AND_DISK_SER
              )
        // 1.把句子拆为单词，放入变量 words
        val words: DStream[String] = lines.flatMap(_.split(" "))
        // 2.转换单词,放入变量 tuples
        val tuples: DStream[(String, Int)] = words.map((_, 1))

        // 3.词频 reduce,放入变量 counts
        val counts: DStream[(String, Int)] = tuples.reduceByKey(_ + _)

        // 4.展示和启动,变量 counts 打印
        counts.print()
        ssc.start()
        // main 方法执行完毕后整个程序就会退出，所以需要阻塞主线程
        ssc.awaitTermination()
    }
}
```

4. 编写 Spark Streaming 实时计算程序：接收 node01 主机 9999 发送过来的数据，每 3 秒钟划分一个批次，统计窗口的大小为 12 秒，每 6 秒统计一次窗口内单词出现的次数。客户端会向 node01 主机 9999 端口不断的发送一个个句子。

```
object WordCount_reduce_Window {
  def main(args: Array[String]): Unit = {
      //1.创建 SparkConf 对象，设置 Appname 属性和集群模式，当前设置为本地模式
      val conf = new SparkConf().setAppName("WordCount_Window")
            .setMaster("local[3]")
      //2.构造 SparkStreaming 程序主入口，对象名为 ssc 并设置批处理间隔为 3 秒
      val ssc = new StreamingContext(conf, Seconds(3))
      //3.ssc 对象的 SparkContexts 设置日志级别为 OFF,关闭日志
      ssc.sparkContext.setLogLevel("OFF")
      //4.设置检查点目录，用于保存状态
      ssc.checkpoint("checkpoint")
      //5.创建 socket 连接，Socket 服务端为主机名 localhost，端口号：9999
      val lines = ssc.socketTextStream("node01", 9999)
      //6.获取数据存储变量 pairs，每 3 秒流数据 FlatMap()处理后，转为二元组(word,1)
      val pairs = lines.flatMap(x => x.split("\\s+")).map((_, 1))
      //7.定义变量 wordsWindows 接收窗口的大小为 12 秒，每次滑动的距离为 6 秒
      val wordsWindow = pairs.window(Seconds(12), Seconds(6))
      //8.创建窗口 wordsWindows 执行 reduceByKey 完成单词累加，存储变量 stateDstream
      val stateDstream = wordsWindow.reduceByKey((x: Int, y: Int) => x + y)
      //9.打印当前批处理数据的结果
```

```
    stateDstream.print()
    //10.启动流的执行：
    ssc.start()
    //11.等待执行停止。执行过程中发生的任何异常将在这个线程抛出。
    ssc.awaitTermination()
    }
}
```

5.使用 Flink 流处理程序完成从文件中读取文本的每一行内容，进行分词，统计每个单词出现的次数。

```
public class BoundedStreamWordCount {
    public static void main(String[] args)throws Exception {
        StreamExecutionEnvironment env=
            StreamExecutionEnvironment.getExecutionEnvironment();
        // 2. 读取文件
        DataStreamSource<String> lineDSS =
            env.readTextFile("input/words.txt");
        // 3. 转换数据格式
        SingleOutputStreamOperator<Tuple2<String, Long>>  wordAndOne =
        lineDSS.flatMap(new FlatMapFunction<String, Tuple2<String,
        Long>>() {
         @Override
         public void flatMap(String line, Collector<Tuple2<String,
          Long>> out) throws Exception {
            String[] words=line.split(" ");
            for(String word:words) {
                out.collect(Tuple2.of(word,1L));
            }
          }
        });
         // 4. 分 组
          KeyedStream<Tuple2<String, Long>, String> wordAndOneKS =
                wordAndOne.keyBy(t->t.f0);
         // 5. 求 和
          SingleOutputStreamOperator<Tuple2<String, Long>> result =
                    wordAndOneKS.sum(1);
         // 6. 打 印
          result.print();
         // 7. 执 行
          env.execute()
    }
}
```

6.编写 Spark Streaming 程序。完成从主机 node01 端口 9999 接收数据，统计数据流中每个单词出现的次数。

```
object WordCount_UpdateStateByKey {
   def main(args: Array[String]): Unit = {
      //1.创建 SparkConf 对象，设置 Appname 属性和集群模式
      val conf = new SparkConf()
              .setAppName("WordCount_UpdateStateByKey")
              .setMaster("local[3]")
      //2.SparkStreaming 程序主入口 ssc，并设置批处理间隔为 5 秒
      val ssc = new StreamingContext(conf,Seconds(5))
      //3.ssc 对象的 SparkContexts 设置日志级别为 OFF,关闭日志
      ssc.sparkContext.setLogLevel("OFF")
     //4.设置检查点目录，用于保存状态
     ssc.checkpoint("checkpoint")
     //5.定义更新状态方法，参数 values 为当前批次单词频度，
      //state 为以往批次单词频度
      val updateFunc = (newValue: Seq[Int],
                  runningValue: Option[Int]) => {
      // newValue 之所以是一个 Seq, 是因为它是某一个
      //Batch 的某个 Key 的全部 Value
      val currentBatchSum = newValue.sum
      val state = runningValue.getOrElse(0)
      // 返回的这个 Some(count) 会再次进入 Checkpoint 中当作状态存储
      Some(currentBatchSum + state)
      }
      //6.调用 socketTextStream()函数创建 socket 连接，存储变量 lines
      //Socket 服务端为主机名 node01，端口号：9999
      val lines = ssc.socketTextStream("node01",9999)
      //7.获取 Scoket 的数据存储变量 pairs，将每 5 秒的流数据
      //进行 FlatMap()处理后，转为二元组形式(word,1)
      val pairs = lines.flatMap(x=>x.split("\\s+")).map(( ,1))
      //8.使用 updateStateByKey 来更新状态，存储变量 stateDstream
      //统计从运行开始以来单词总的次数
      val stateDstream = pairs.updateStateByKey[Int](updateFunc)
      //9.打印当前批处理数据的结果
      stateDstream.print()
      //10.启动流的执行：开启 Job，并启动 JobGenerator(Job 生成器)
      //和 ReceiverTracker(接收器)
      ssc.start()
      //11.等待执行停止。执行过程中发生的任何异常将在这个线程抛出。
      ssc.awaitTermination()
     }
}
```