Spark의 설계와 운행 원리

Slide 3: 2.3.2 Spark 운행 기본 프로세스 - 도입

현재 우리는 본 절의 핵심 내용인 **2.3.2 Spark 운행 기본 프로세스**로 들어갑니다.

이는 Spark 작동 메커니즘을 이해하는 핵심 부분입니다.

Slide 4: Spark 운행 기본 프로세스 다이어그램

이제 Spark 운행의 기본 프로세스 다이어그램을 살펴보겠습니다. 이 그림은 Spark 애플리케이션이 시작부터 완료까지의 전체 과정을 보여줍니다.
먼저 **SparkContext**가 있는데, 이는 RDD Objects와 구축된 DAG 그래프를 포함합니다. 다음으로 DAG 그래프는 **DAG Scheduler**로 전달되어 DAG 그래프를 Stage로 분해하는 역할을 담당합니다.
그런 다음 Stage는 Taskset을 생성하고, 마지막으로 **Task Scheduler**가 Task를 제출하고 모니터링합니다.
오른쪽에서는 리소스 매니저가 리소스 할당과 사용 상황 모니터링을 담당하는 것을 볼 수 있습니다. 전체 과정은 네 가지 주요 단계를 포함합니다:

1. 등록 및 리소스 신청
2. 리소스 할당
3. 등록 및 Task 신청
4. 등록 해제

마지막으로 Task는 Executor 프로세스로 분배되어 실행되고, 리소스 사용 상황을 보고합니다.

Slide 3: 2.3.2 Spark运行基本流程 - 导入页

现在我们进入本节的核心内容—Spark运行基本流程。

这是理解Spark工作机制的关键部分。

Slide 4: Spark运行基本流程图

现在让我们来看Spark运行的基本流程图。
这个图展示了Spark应用程序从启动到完成的完整过程。

首先，我们有SparkContext，它包含RDD Objects和构建的DAG图。
接下来，DAG图会被送到DAG Scheduler，负责将DAG图分解成Stage。
然后Stage会生成Taskset，最后由Task Scheduler来提交和监控Task。

在右侧，我们看到资源管理器负责分配和监控资源使用情况。整个过程包括四个主要步骤：

1. 注册(zhùcè)并申请资源
2. 分配资源
3. 注册并申请Task
4. 注销(**zhùxiāo)**

最后，Task会被分发到Executor进程中执行，并汇报资源使用状况。

Slide 5: 단순화된 운행 프로세스

여기서는 더 단순화된 프로세스 다이어그램을 볼 수 있습니다. Driver가 SparkContext를 포함하고 있으며, 작업을 여러 단계로 분해합니다.
각 단계가 완료되면 결과가 사용자에게 반환됩니다.
이 과정은 Spark의 분산 컴퓨팅 특성을 잘 보여줍니다.

Slide 6: 클러스터와 애플리케이션의 관계

이 그림에서는 클러스터가 애플리케이션을 통해 Driver의 SparkContext와 어떻게 연결을 구축하는지 볼 수 있습니다.

- **클러스터**: 우리의 컴퓨팅 리소스
- **애플리케이션**: 실행할 구체적인 작업
- **SparkContext**: 둘을 연결하는 다리 역할

Slide 7: 리소스 신청과 작업 분배

여기서는 SparkContext가 리소스 매니저에게 리소스를 신청하고 모니터링을 수행하는 방법을 보여줍니다.

작업이 서로 다른 단계로 분해되고, 다시 구체적인 작업으로 세분화되어 최종적으로 다른 워커 노드에 분배되어 실행되는 것을 볼 수 있습니다.

Slide 5: 简化的运行流程

这里我们看到一个更简化的流程图。
Driver包含SparkContext，它将作业分解成多个阶段，
每个阶段处理完成后，
结果会返回给用户。

这个过程体现了Spark的分布式计算特性。

Slide 6: 集群与应用程序的关系

在这张图中，我们看到集群如何通过应用程序与Driver中的SparkContext建立连接。

- 集群代表了我们的计算资源，
- 而应用程序是我们要执行的具体任务，
- SparkContext则是连接两者的桥梁。

Slide 7: 资源申请和任务分发

这里展示了SparkContext如何向资源管理器申请资源，以及如何进行监控。

我们可以看到作业被分解成不同的阶段，
然后进一步分解成具体的任务，
最终分发到不同的工作节点上执行。

Slide 8: 리소스 매니저의 역할

SparkContext는 리소스 요청을 통해 리소스 매니저에게 컴퓨팅 리소스를 신청합니다.

리소스 매니저는 **YARN**이나 **Mesos**가 될 수 있으며, 이들은 리소스를 할당하고 Executor 프로세스를 시작하는 책임을 집니다.

여기서 CPU 아이콘을 볼 수 있는데, 이는 실제 컴퓨팅 리소스를 나타냅니다.

Slide 9: 분산 실행 환경

이 그림에서는 여러 개의 Worker Node를 볼 수 있으며, 각 노드에서 Executor 프로세스가 실행되고 있습니다.

이는 Spark의 진정한 분산 특성을 보여줍니다 - 컴퓨팅 작업이 클러스터의 여러 머신에 분산되어 병렬로 실행됩니다.

Slide 10: RDD와 DAG의 관계

여기서는 SparkContext가 코드의 RDD 연산을 바탕으로 RDD 의존 관계를 구축하고 최종적으로 DAG 그래프를 형성하는 방법을 설명합니다.

이는 Spark 컴퓨팅 모델의 핵심입니다 - **모든 연산이 방향성 비순환 그래프(DAG)로 구축됩니다.**

Slide 8: 资源管理器的作用

SparkContext通过资源请求向资源管理器申请(**shēnqǐng**)计算资源。

资源管理器可以是YARN或Mesos，它们负责分配资源并启动Executor进程。

这里我们可以看到CPU的图标，代表了实际的计算资源。

Slide 9: 分布式执行环境

在这张图中，我们看到多个Worker Node，每个节点上都运行着Executor进程。

这展示了Spark真正的分布式特性——计算任务被分散到集群中的多台机器上并行执行。

Slide 10: RDD和DAG的关系

这里说明了SparkContext如何根据代码中的RDD操作构建RDD依赖关系，最终形成DAG图。

这是Spark计算模型的核心——所有的操作都会被构建成一个有向无环图。

Slide 11: DAG Scheduler의 역할

DAG 그래프가 구축 완료되면 **DAG Scheduler**로 전달됩니다.
DAG Scheduler의 주요 임무는 복잡한 DAG 그래프를 여러 **Stage(단계)**로 분해하여 각 Stage 내의 연산이 병렬로 실행될 수 있도록 하는 것입니다.

Slide 12: Stage와 Task의 관계

이 그림은 Stage의 개념을 보여줍니다.
각 Stage는 실제로 작업의 집합이며, 하나의 Stage는 병렬로 실행할 수 있는 여러 Task를 포함합니다.

이러한 계층적 설계는 Spark가 대규모 컴퓨팅 작업을 효율적으로 관리하고 스케줄링할 수 있게 합니다.

Slide 13: Task Scheduler의 책임

Task Scheduler는 작업 분배를 담당합니다.
Worker Node에서 실행 리소스를 신청한 후, 구체적인 Task를 각 실행 노드에 분배합니다.
이는 Spark 작업 스케줄링의 마지막 단계입니다.

Slide 11: DAG Scheduler的作用

DAG图构建完成后，会被送到DAG Scheduler。
DAG Scheduler的主要任务是将复杂的DAG图分解成多个Stage（阶段），
使得每个Stage内的操作可以并行执行。

Slide 12: Stage和Task的关系

这张图展示(**zhǎnshì**)了Stage的概念。
每个Stage实际上是任务的集合，一个Stage包含多个可以并行执行的Task。

这种分层(**fēncéng**)的设计使得Spark能够高效地管理和调度大规模的计算任务。

Slide 13: Task Scheduler的职责

Task Scheduler负责(**fùzé**)分发任务。

它从Worker Node申请运行资源，
然后将具体的Task分发到各个执行节点上。

这是Spark任务调度的最后一个环节。

Slide 14: 작업 반환 메커니즘

작업 실행이 완료되면 Task Scheduler가 작업을 Worker Node로 반환하여 전체 컴퓨팅 프로세스를 완료합니다.

이러한 양방향 통신 메커니즘은 컴퓨팅 결과가 올바르게 수집되고 집계될 수 있도록 보장합니다.

Slide 15-17: 계산이 데이터로 이동하는 원리

이 몇 장의 그림은 Spark의 "컴퓨터가 데이터 있는데로 이동한다"는 중요한 원리를 생생하게 보여줍니다.
처음에는 머신 A의 계산 프로그램과 머신 B의 데이터가 있습니다. 전통적인 방법은 데이터를 계산 프로그램이 있는 머신으로 전송하는 것일 수 있지만, 이는 대량의 네트워크 전송 오버헤드를 발생시킵니다.

Spark는 반대 전략을 채택합니다 - **계산 프로그램을 데이터가 있는 머신으로 보내는 것**입니다.
이렇게 하는 장점:

- 네트워크 전송 오버헤드 감소
- 계산 효율성 향상
- 클러스터 리소스의 더 나은 활용

Slide 14: 任务返回机制

任务执行完成后，Task Scheduler会将任务返回给Worker Node，完成整个计算流程。

这种双向通信机制确保了计算结果能够正确地收集和汇总。

Slide 15:-17: 计算向数据靠拢的原理
(여기서는 애니메이션 보면서)
这张图生动地展示了Spark"计算向数据靠拢(**kàolǒng**)"的重要原理。

最初，我们有机器A上的计算程序和机器B上的数据。传统的做法可能是将数据传输到计算程序所在的机器，但这样会产生大量的网络传输开销。

Spark采用了相反的策略——将计算程序发送到数据所在的机器。

这样做的好处是：

- 减少网络传输开销
- 提高计算效率
- 更好地利用集群资源

Slide 18-19: Task Scheduler의 지능형 스케줄링

Task Scheduler는 지능적으로 계산 프로그램을 데이터가 있는 머신으로 분배합니다.

- 머신 A에 데이터가 있을 때는 계산 프로그램이 머신 A로 전송
- 데이터가 머신 B에 있을 때는 계산 프로그램이 머신 B로 전송

이러한 전략은 Spark의 실행 효율성을 크게 향상시킵니다.

Slide 20: 결과 수집과 출력

계산이 완료되면 결과가 각 Executor에서 Task Scheduler로 반환되고, 다시 DAG Scheduler로 전달되어 최종적으로 SparkContext를 통해 사용자에게 반환되거나 HDFS 등의 저장 시스템에 기록됩니다.

이로써 **전체 Spark 애플리케이션의 라이프사이클이 완료**됩니다.

Slide 18-19: Task Scheduler的智能调度

Task Scheduler会智能地将计算程序分发到数据所在的机器上。

- 当机器A有数据时，计算程序就会被发送到机器A；
- 当数据在机器B时，计算程序就会被发送到机器B。

这种策略大大提高了Spark的执行效率。

Slide 20: 结果收集和输出

计算完成后，结果会从各个Executor返回到Task Scheduler，

再传递给DAG Scheduler，

最终通过SparkContext返回给用户，

或者写入HDFS等存储系统。

这完成了整个Spark应用程序的生命周期。

Slide 21-23: SparkContext의 핵심 역할

마지막 몇 장의 그림은 **SparkContext의 중요성**을 강조합니다.

SparkContext는 애플리케이션이 클러스터에 연결하는 채널을 나타내며, 애플리케이션과 클러스터 사이의 다리 역할을 합니다.

비유:

- **애플리케이션**: 우리가 전송하려는 정보
- **클러스터**: 목적지
- **SparkContext**: 둘을 연결하는 채널이나 파이프라인

이 채널이 없다면 애플리케이션은 클러스터의 컴퓨팅 리소스를 활용할 수 없습니다.

Slide 21-23: SparkContext的核心作用

最后这几张图强调了SparkContext的重要性。

SparkContext代表了应用程序连接集群的通道，它是应用程序和集群之间的桥梁(**qiáoliáng)**。

想象一下，

- 应用程序就像是我们要发送的信息，
- 集群就像是目的地，
- 而SparkContext就是连接两者的通道或管道。

没有这个通道，应用程序就无法利用集群的计算资源。

Slide 24: 문제 1 B

Slide 25: 문제 2 C

Slide 26: 문제 3 C

Slide 27: 문제 4 C

Slide 28: 문제 5 C

Slide 29: 문제 6 B

Slide 30: 문제 7 C

Slide 31: 문제 8 B

Slide 32: 문제 9 C

Slide 33: 문제 10 B

Slide 24: 정리

이것이 Spark 운행의 기본 프로세스입니다.
우리는 SparkContext 생성부터 시작해서 DAG 구축, Stage 분할, Task 스케줄링, 마지막으로 결과 수집까지의 완전한 과정을 학습했습니다.

이 프로세스를 이해하는 것은 Spark를 깊이 있게 학습하는 데 매우 중요합니다.

Slide 24: 문제 1 B

Slide 25: 문제 2 C

Slide 26: 문제 3 C

Slide 27: 문제 4 C

Slide 28: 문제 5 C

Slide 29: 문제 6 B

Slide 30: 문제 7 C

Slide 31: 문제 8 B

Slide 32: 문제 9 C

Slide 33: 문제 10 B

Slide 24: 总结

这就是 Spark 运行的基本流程。

我们学习了从 SparkContext 创建开始，到 DAG 构建、Stage 划分、Task 调度，最后到结果收集的完整过程。

理解这个流程对于我们深入学习 Spark 非常重要

핵심 내용 요약

Spark 운행 프로세스의 주요 단계:

1. **SparkContext 생성**: 애플리케이션과 클러스터를 연결하는 진입점
2. **RDD 및 DAG 구축**: 코드의 연산을 방향성 비순환 그래프로 변환
3. **DAG Scheduler**: DAG를 Stage로 분할
4. **Task Scheduler**: Stage를 Task로 분할하고 실행 노드에 분배
5. **Executor 실행**: 실제 계산 수행
6. **결과 수집**: 계산 결과를 수집하고 사용자에게 반환

핵심 원리:

- **계산이 데이터로 이동**: 네트워크 오버헤드를 최소화하기 위해 데이터가 있는 곳으로 계산을 이동
- **분산 병렬 처리**: 여러 노드에서 동시에 작업을 수행하여 효율성 극대화
- **계층적 스케줄링**: DAG → Stage → Task의 계층 구조로 효율적인 작업 관리