**Slide 1: Slide 2: Slide 3:**

**Slide 4: Slide 5:**

**Slide 6: Hadoop의 기원**

여러분, 오늘 우리가 배울 빅데이터의 대표적인 기술은 바로 **Hadoop**입니다!

🗓️ Hadoop은 2005년에 처음 탄생했고, 2015년에는 완전한 생태계를 형성했습니다.
Hadoop의 등장은 데이터 저장과 계산 방식을 **완전히 바꾸어놓았습니다**
이렇게 이해하시면 됩니다: "마을에 슈퍼 창고 관리자 + 일을 엄청 빨리 하는 공사팀이 왔다!"

**Slide 7: 단일 소프트웨어가 아니다!**

많은 학생들이 처음 Hadoop을 들으면 어떤 "고급 프로그래밍 소프트웨어"나 "Excel 같은 것"이라고 생각합니다.
사실 그렇지 않습니다!

📌 **Hadoop은 단일 소프트웨어가 아니라**, 여러 모듈과 서브시스템을 포함하는 **완전한 플랫폼**입니다.

고속철도가 단순히 객차(车厢 - chēxiāng)만이 아니라, 선로(轨道 - guǐdào), 스케줄링 시스템(调度系统 - diàodù xìtǒng), 전력 시스템을 포함하는 것과 같습니다!

**Slide 1: Slide 2: Slide 3:**

**Slide 4: 4개의 대표적 제품Slide 5:**

**Slide** 6:Hadoop 的起源

同学们，我们今天要讲的大数据代表性(**dài biǎo xìng)**技术是——Hadoop！

🗓️ Hadoop 最早诞生于 2005 年，
到了 2015 年已经形成完整的生态系统。

它的出现，
彻底改变(chèdǐ)了数据存储和计算方式。
你可以把它理解为"村里来了个超级仓库管理员 + 干活(gànhuó)特别快的工程队"！

**Slide** 7:不是一个软件！

很多同学第一次听到 Hadoop，
会以为它是某个"高级编程软件"或者"像 Excel 的东西"。
其实不是！

📌 Hadoop 不是单个软件，
而是一个完整平台，包含很多模块和子系统。

就像一辆高铁不仅是车厢chēxiāng，
还包括轨道guǐdào、调度系统diàodù xìtǒng(스케줄링시스템)、电力系统一样！

## Slide 8: Hadoop은 생태계이다

을 더 정확하게 표현하면,
**생태계(ecosystem)**입니다!

포함된 것들:

- **HDFS**: "저장" 담당
- **MapReduce**: "계산" 담당
- **YARN**: "스케줄링(调度 - diàodù)과 조정(协调 - xiétiáo)" 담당
- **Hive, Pig, Mahout** 등: 다양한 도구 체인(链条 - liàntiáo), 각기 다른 일을 담당!

📌 축구팀으로 생각하면 됩니다.
공격수(前锋 - qiánfēng)도 있고,
수비수(后卫 - hòuwèi)도 있고,
코치(教练 - jiàoliàn)도 있으며,
**각자 자신의 역할을 다합니다(各司其职 - gè sī qí zhí)**!

## Slide 8:Hadoop 是个生态系统

Hadoop 更准确地说，
是一个生态系统（ecosystem）！

里面包括：

- HDFS：负责"存东西"
- MapReduce：负责"算东西"
- YARN：负责"调度(diàodù스케줄링하다)和协调(xiétiáo협력적 조정)"
- Hive、Pig、Mahout 等：各种工具链条 (liàntiáo): 체인，干不同的活！

📌 可以把它想成一支球队，有前锋 (qiánfēng)、有后卫、有教练，各司其职！
(yǒu qiánfēng, yǒu hòuwèi, yǒu jiàoliàn):
→ 各司其职 (gè sī qí zhí):

## Slide 9: Hadoop 생태계 (다이어그램)

🎤 이제 이 그림을 보면, Hadoop 생태계의 **"가족사진(全家福 - quánjiāfú)"** 입니다 👪.

중심에는 핵심 요소들:

- **HDFS** (분산 파일 시스템)
- **MapReduce** (계산 모델)
- **YARN** (리소스 스케줄링)

주변(旁边 - pángbiān)을 둘러싸고(围绕 - wéirào) 있는 것은 기능 계층:

- 저장: **HBase**
- 분석: **Hive, Pig**
- 마이닝: **Mahout**
- 모니터링 관리: **Zookeeper, Ambari**

📢 "빅데이터 운영체제 + 앱 마켓"으로 이해하시면 됩니다!

## Slide 10: 두 가지 핵심 기술

Hadoop의 핵심 구성을 살펴보겠습니다:

1️⃣ **분산 파일 저장 (HDFS)**

2️⃣ **분산 계산 프레임워크 (MapReduce)**

이 두 가지는 "좌우호법"으로, 전체 Hadoop 세계를 지탱합니다.
사람이 달리려면 "튼튼한 다리"와 "충분한 폐활량"이 필요한 것처럼 - 이것이 HDFS와 MapReduce의 관계입니다.

## Slide 9:Hadoop 生态系统（图示）

现在我们看这张图，
就是 Hadoop
生态系统的"全家福(quánjiāfú)"👪。

中间是核心：

- HDFS（分布式文件系统）
- MapReduce（计算模型）
- YARN（资源调度）

旁边 (pángbiān)围绕(wéirào)的是功能层：

- 存储：HBase
- 分析：Hive、Pig
- 挖掘：Mahout
- 监控管理：Zookeeper、Ambari

📢 你可以理解为"大数据操作系统+应用市场"！

## Slide 10:两大核心技术

我们来看 Hadoop 的核心组成：

1️⃣ 分布式文件存储（HDFS）

2️⃣ 分布式计算框架（MapReduce）

这两个是"左右护法(zuǒ yòu hù fǎ)"，撑(chēng )起整个 Hadoop 世界。
就像一个人要跑步，得有(děi yǒu）"强壮(qiángzhuàng )的腿( tuǐ)"和"充沛(chōngpèi)的肺(fèi)"——这就是 HDFS 和 MapReduce 的关系。

**Slide 11: HDFS - 대용량 분산 파일 저장**

HDFS의 정식 명칭은: **Hadoop Distributed File System**입니다.

HDFS의 임무는 초대형 파일을 조각으로 나누어 여러 노드에 분산 저장하고, 각 조각을 여러 번 백업하는 것입니다!

📦 **장점:**

- 내결함성이 강함
- 대용량 파일에 적합
- 확장성이 강함

📌 이렇게 상상해보세요: 논문을 10페이지로 나누어 인쇄하고, 각각 10명의 친구 USB에 저장하면서 모두 백업본을 만들어두면, 어떤 USB가 고장나도 걱정 없습니다!

**Slide 12: YARN - 리소스 스케줄링 및 관리**

YARN은 **Yet Another Resource Negotiator** 입니다.

역할:
누가 계산할까?
어디서 계산할까?
어떻게 계산할까?
회사의 HR 시스템 같은 것입니다:
작업이 들어오면 누가 할지,
어느 자리에서 할지,
끝나면 어디에 제출할지 결정합니다.

📌 YARN 덕분에 여러 프레임워크가 동일한 클러스터에서 충돌 없이 함께 작동할 수 있습니다!

**Slide** 11:HDFS：海量分布式文件存储

HDFS 的全称是：Hadoop Distributed File System。

它的任务是把超大的文件切片分发到多个节点去存储，每个片还会备份好几份！

📦 优势：

- 容错性强
- 适合大文件
- 扩展性强

📌 想象你把论文分成10页打印，分别存在10个朋友U盘里，还都备份一份，任何一个U盘坏了都不怕！

**Slide** 12:YARN：资源调度与管理

YARN 是 Yet Another Resource Negotiator（又一个资源协调者）。

它的职责(zhí zé)是：
谁来算？
在哪算？
怎么算？
就像公司里的HR系统：
任务来了以后，
它决定谁来干、
在哪个工位干、干完去哪提交。

📌 有了 YARN，多种框架就能共存在同一个集群下工作，不打架！

HR: **人力资源管理系统**

**Slide 13: Hive 数据仓库**

Hadoop에서 복잡한 MapReduce 프로그래밍 없이 SQL만으로 대용량 데이터를 분석할 수 있게 합니다.
"SQL on Hadoop" 의 대표 주자입니다.

**Slide 14: 데이터베이스 vs 데이터 웨어하우스 (다이어그램: 10 개)**

이 그림은 좀 귀엽습니다: 재고가 원래 10개였습니다.
데이터베이스는 이 "현재 상태"를 기록할 수 있습니다:

📦 예를 들어 데이터베이스를 열면 "재고: 10개"라고 표시됩니다

📌 하지만! 내일 9개로 바뀌면, 이전의 "10개"라는 기록은 사라집니다...

따라서 데이터베이스는 "현재 스냅샷"만 기록할 수 있고, 역사적 추세를 반영할 수 없습니다.

**Slide** 13: Hive 数据仓库

Hive 是建立在 Hadoop 上的数据仓库基础工具，通过 类SQL语言 对大规模数据进行处理和分析 .
被称为 “SQL on Hadoop” 的典型代表

**Slide** 14:数据库 vs 数据仓库（图示：10件）

我们来看看数据库和数据仓库的区别.

这张图有点可爱：库存原本是 10 件。

数据库能记录这个“当下的状态”：

📦 比如你打开数据库，它显示“库存：10件”

📌 但是！如果明天变成 9 件，旧的那个“10件”的记录就没了……

所以数据库只能记录“当前快照”，无法反映历史趋势。

**Slide 15: 데이터베이스 상태 변화 (다이어그램: 9개로 변경)**

이어서 보면 - 재고가 10개에서 9개로 변했습니다.

데이터베이스는 현재 9개라고 알려줄 수 있지만, 다음을 알려줄 수 없습니다:

- 어제 1개를 판 것인가?
- 누가 샀는가?
- 판매 속도가 빠른가?

📌 그래서 우리는 "매번의 변화를 기록"하는 시스템이 필요합니다. 이것이 바로 "데이터 웨어하우스"입니다!

**Slide 16: 데이터 웨어하우스란 무엇인가?**

방금 데이터베이스는 "현재 상태"만 기록한다고 했는데, 그럼 어떻게 해야 할까요?

📦 우리는 또 다른 개념을 도입합니다:

**데이터 웨어하우스(Data Warehouse)**

이것은 "시간축 위의 창고"와 같습니다 - 현재 재고가 얼마인지만 알려주는 게 아니라, 과거의 모든 변화를 기록합니다.

📌 비유하자면:

- 데이터베이스 = 폴라로이드 사진 📸
- 데이터 웨어하우스 = 감시 카메라 녹화 📹 (재생 가능)

**Slide 17:**

**Slide** 15: 数据库状态变化（图示：变成 9 件）

接着看—库存从 10 变成了 9。

数据库能告诉你现在是9件，但没办法告诉你：

- 昨天是不是卖了1件？
- 是谁买的？
- 卖得快不快？

📌 所以我们需要"记录每一次变化"的系统，这就叫做"数据仓库"！

**Slide** 16:数据仓库是什么？

刚刚我们提到数据库只能记录"现在的状态"，那怎么办呢？

📦 我们引入另一个概念：

数据仓库（Data Warehouse）

它就像一个"时间轴(zhóu)上的仓库"——不仅告诉你现在有多少货，还记录过去的每次变化。

📌 类比一下：

- 数据库 = 拍立得(Pāilìdé)照片 📸
- 数据仓库 = 监控(jiānkòng)录像(lùxiàng) 📹（能回放）

**Slide** 17:

**Slide 18: 데이터 웨어하우스는 역사 데이터를 지속적으로 저장**

데이터 웨어하우스의 핵심은:
✅ 시간 차원으로 연속적인 변화 데이터를 저장합니다

예를 들어 재고가 10 → 9 → 8로 변하면, 매번의 변화가 기록됩니다.

📌 이렇게 하면 분석할 수 있습니다: "지난주에 어떻게 팔렸나?", "어느 시간대의 판매량이 가장 높았나?"
반면 데이터베이스는 "현재 8개"라고만 알려줄 뿐, 과정은 사라집니다!

**Slide** 19:

**Slide 20: 데이터베이스 vs 데이터 웨어하우스 (요약)**
차이점을 정리하면:

| **항목** | **데이터베이스** | **데이터 웨어하우스** |
|---|---|---|
| 데이터 유형 | 실시간 상태 데이터 | 시간 차원 연속 데이터 |
| 사용 목적 | 트랜잭션 처리 (예: 주문) | 의사결정 분석 (예: 판매 분석) |
| 쿼리 방식 | 정확한 위치 지정 | 집계 추세 분석 |

📌 창고 관리자라면, 데이터베이스는 손에 있는 "라벨 건"이고, 데이터 웨어하우스는 "월간 보고서 분석표"입니다.

**Slide** 18:数据仓库持续存储历史数据

数据仓库的关键就是：
✅ 它按时间维度，保存连续(lián xù)变化的数据

比如库存从10 → 9 → 8，每一次变化都会被记录。

📌 这样我们就可以分析："上周是怎么卖出去的？"、"哪个时间段销量最高？"
而数据库只能告诉你"现在是8件"，
过程没了！

**Slide** 19:슬라이드에 있는 내용 말하기

**Slide** 20: 数据库 vs 数据仓库（总结）

我们总结一下区别：

| 项目 | 数据库 | 数据仓库 |
|---|---|---|
| 数据类型 | 实时状态数据 | 时间维度连续数据 |
| 使用目的 | 事务处理（如下单） | 决策分析（如销量分析） |
| 查询方式 | 精确定位 | 聚合趋势分析 |

📌 如果你是仓库管理员，数据库像是你手里的"标签枪(**biāoqiān qiāng**)"，数据仓库就像"月度报表分析表"。

**Slide 21: 데이터 웨어하우스가 할 수 있는 일**

여러분, 데이터 웨어하우스가 강력하다는 것을 알았는데, 구체적으로 무엇을 할 수 있을까요?

✅ 가장 흔한 용도는: **의사결정 분석 지원 (Decision Making)**

예를 들어:

- 어떤 상품이 가장 잘 팔리나?
- 어느 시간대에 사용자가 가장 활발한가?
- 어느 지역의 소비력이 강한가?

📌 데이터로 말하면, 상사가 정확한 판단을 내릴 수 있습니다!

**Slide 22: 의사결정 분석 지원 시스템**

그림에 표시된 것은 "의사결정 분석" 체인입니다:

📊 데이터 → 데이터 웨어하우스 → 분석 도구 → 의사결정 지원 시스템(DSS)

이러한 시스템은 실시간으로 "관리 대시보드"를 제공할 수 있습니다. 비행기 조종석처럼 언제든지 지표에 "빨간불"이 켜졌는지 알려줍니다.

**Slide** 21:数据仓库能干什么？

同学们，知道数据仓库厉害，

那它具体能干啥？

✅ 它最常见的用途就是：支持决策分析（Decision Making）

比如：

- 哪个商品卖得最好？
- 什么时间点用户活跃huó yuè最高？
- 哪个区域消费力强？

📌 用数据说话，老板才能做出精准判断(jīng zhǔn pàn duàn)！

**Slide** 22: 决策分析支持系统

图中展示的就是"决策分析"链路：

📊 数据 → 数据仓库 → 分析工具 → 决策支持系统（DSS）

这种系统可以实时提供"管理仪表盘(yí biǎo pán)"，就像飞机驾驶舱(jiàshǐcāng)，随时告诉你指标(zhǐbiāo)有没有"亮红灯(liàng hóngdēng)"。

**Slide 23: OLAP 분석이 핵심!**
새로운 용어가 나옵니다: **OLAP (Online Analytical Processing)**

이것은 다차원 데이터 분석 방법으로, 다양한 각도에서 데이터를 분석할 수 있습니다:

- 시간 차원
- 지역 차원
- 사용자 차원
- 제품 범주 차원

📌 데이터계의 "루빅스 큐브"로 생각하세요: 어느 면을 돌리든 다양한 조합 뷰를 볼 수 있습니다!

**Slide 24: 상품 판매 분석 (사례)**

구체적인 응용을 살펴봅시다:
🛒 예를 들어 특정 상품의 판매량을 분석하려면:

- 언제 잘 팔렸나?
- 어느 지역의 판매량이 높나?
- 명절에 더 인기가 있나?

데이터 웨어하우스와 OLAP을 결합하면 다음을 할 수 있습니다:
📌 가로로는 시간, 세로로는 지역, 제품 분류 차원까지 결합 → 다차원 교차 분석!

**학생 A**: 선생님, 이게 타오바오의 "데이터 나침반"과 비슷한가요?
**선생님 (웃음)**: 맞아요! 타오바오, 징동, 더우인 전자상거래 모두 이런 분석 시스템으로 먹고 살아요!

**Slide** 23: OLAP分析是重点！
我们进入一个新词：OLAP（Online Analytical Processing）

它是一种多维数据分析方法，可以从不同角度分析数据，比如：

- 时间维度
- 地区维度
- 用户维度
- 产品种类维度

📌 你可以把它想成数据界的"魔方(mó fang)"：你从哪一面转，都能看到不同的组合视图！

**Slide** 24: 商品销量分析（案例）

我们来看具体应用：
🛒 比如你要分析某个商品的销量：

- 什么时候卖得好？
- 哪个地区销量高？
- 是不是节假日更火？

数据仓库结合 OLAP 可以做到：

📌 横向看时间，纵向看地域，还能结合商品分类维度 → 多维交叉分析！

学生A：老师，那这个是不是和淘宝"数据罗盘"很像？
老师（笑）：对啊，淘宝、京东、抖音电商全靠这套分析系统吃饭呢！

**Slide 25: 전통적인 데이터 웨어하우스의 난제**

여러분도 아시다시피, 전통적인 데이터 웨어하우스와 관계형 데이터베이스 시스템은 빅데이터 처리에서 도전에 직면합니다:

📌 왜 그럴까요?

- 데이터량이 너무 많아서 단일 장비로는 버틸 수 없습니다!
- 쿼리가 느리고 확장이 어렵습니다
- 유지보수 비용이 높습니다

따라서 우리는 Hadoop 생태계의 도구를 사용하여 처리를 보조해야 합니다. 예: Hive, Pig, Mahout!

**Slide 26:**

**Slide 27: Hive의 SQL 쿼리 기능**

어떤 학생이 말합니다: "선생님, 저는 SQL만 할 줄 아는데 어떡하죠?"
걱정 마세요! Hive가 있으면 SQL만 작성하면 되고, 뒤에서 자동으로 MapReduce로 변환되어 실행됩니다!

🎯 예를 들어:

```
SELECT 상품명, COUNT(*)
FROM 판매기록
GROUP BY 상품명;
```

시스템이 자동으로 이것을 분산 프로그램으로 번역해서 실행합니다.
편리하지 않나요? 😆

**Slide** 25: 传统数据仓库的难题

大家知道，传统的数据仓库和关系数据库系统在处理大数据时面临挑战(tiǎo zhàn)：

📌 为什么？

- 数据量太大，单机撑不住 **(chēng bú zhù)** ！
- 查询慢，扩展难
- 维护成本高

所以我们需要 Hadoop 生态系统中的工具来辅助( fǔ zhù)处理，比如 Hive、Pig、Mahout！

**Slide 26: 앞의 내용과 연결해서**

**Slide** 27: Hive 的 SQL 查询能力

有的同学说："老师，那我只会写 SQL，怎么办？"
别担心！有了 Hive，你只需要写 SQL，背后就会自动转成 MapReduce 来运行！

🎯 你写一句：

```
SELECT 商品名, COUNT(*) FROM 销售记录
GROUP BY 商品名;
```

系统会自动把它翻译成分布式程序来执行。
是不是很贴心(tiē xīn)？😆

## Slide 28: Hive의 핵심 장점

📌 Hive의 장점:

- 친숙한 인터페이스, 학습 비용 낮음
- 구조화된 데이터 처리 가능
- 자동 실행 계획 최적화
- 유연한 출력 형식

"말만 하면" 시스템이 "알아서 처리해줍니다"!

## Slide 29: Pig Latin - 프로그래머를 위한 또 다른 길

Hive의 SQL이 충분히 유연하지 않다고 생각되면, 또 다른 도구가 있습니다:

**Pig Latin**이라는 스크립트 언어를 사용하며, 프로그래머가 다음 작업에 적합합니다:

- 데이터 정제
- 형식 변환
- 배치 처리 변환 로직

📌 이렇게 이해하세요: Hive는 분석가용, Pig는 엔지니어용!

## Slide 28: Hive 的核心优势

📌 Hive 的好处是：

- 接口熟悉，学习成本低
- 可以处理结构化数据
- 自动优化执行计划
- 输出格式灵活

就像你只需要"开口说话"，系统就"安排上了"！

## Slide 29: Pig Latin——程序员的另一条路

如果你觉得 Hive 的 SQL 不够灵活，还有一款(**yī kuǎn**)工具：Pig

它使用一种叫做 Pig Latin 的脚本(**jiǎo běn**)语言，适合程序员做：

- 数据清洗
- 格式转换
- 批处理转换逻辑

📌 你可以理解为：Hive 给分析师用，Pig 给工程师用！

Analysts：What
Engineer: How

## Slide 30: Hadoop 생태계 다이어그램 보충

Hadoop 생태계를 다시 한번 복습합니다.

📌 주목하세요:

- **HDFS**는 "데이터 웨어하우스"
- **MapReduce**는 "데이터 가공 공장"
- **Hive/Pig**는 "조작 인터페이스"
- **Mahout**는 "지능 분석 어시스턴트"

이들이 함께 빅데이터 공장의 "생산라인"을 구성합니다!

## Slide 31: Mahout - 데이터 마이닝 알고리즘 구현

데이터 분석을 언급하면 Mahout를 빼놓을 수 없습니다!

Hadoop 위의 데이터 마이닝 라이브러리로, 다음 일반적인 알고리즘을 지원합니다:

- 분류 (Classification)
- 군집화 (Clustering)
- 회귀 (Regression)

📌 "데이터 과학 도구 상자"처럼, 바로 가져다 쓸 수 있습니다!

## Slide 30: Hadoop 生态图补充

我们再回顾一次 Hadoop 的生态系统。

📌 请注意：

- HDFS 是"数据仓库"
- MapReduce 是"数据加工厂"
- Hive/Pig 是"操作界面"
- Mahout 是"智能分析助手"

它们共同构成了一个大数据工厂的"流水线"！

## Slide 31: Mahout：实现数据挖掘算法

讲到数据分析，不能不提 Mahout！

它是 Hadoop 上的数据挖掘库，支持以下常见算法：

- 分类（Classification）
- 聚类（Clustering）
- 回归（Regression）

📌 就像一个"数据科学工具箱"，能直接拿来用！

## Slide 32: Hadoop에서 Mahout 사용의 도전

하지만 주의하세요 ⚠️:
전통적인 데이터 마이닝 알고리즘을 그냥 가져다가 분산 환경에서 사용하면 안 됩니다!

반드시 개조해야 하며, "병렬 실행", "내결함성", "데이터 분할"을 할 수 있어야 합니다...
📢 그래서 Mahout 개발자들이 많은 리팩토링 작업을 해서 Hadoop에서 실행될 수 있도록 했습니다.

## Slide 33: 고전적인 예시 - 의사결정 트리 구현은 정말 힘들다!

의사결정 트리 모델 구현이 간단하다고 생각하시나요?
분산 환경에서는 고려해야 할 사항:

- 데이터를 어떻게 분할할까?
- 각 노드에서 하위 트리를 어떻게 계산할까?
- 트리 구조를 어떻게 병합할까?

🫠 한마디로: 구현하기 정말 힘듭니다...

그래서 Mahout가 캡슐화된 알고리즘 라이브러리를 제공하여 부담을 덜어줍니다!

## Slide 32:在 Hadoop 上用 Mahout 的挑战

不过你要注意⚠️：
传统的数据挖掘算法，直接拿来放到分布式环境用，是不行的！

必须进行改造，能"并行执行"、"容错"、"数据切分"……

📢 所以 Mahout 的开发者做了很多重构，才让它能跑在 Hadoop 上。

## Slide 33: 经典例子：决策树实现很累！

你以为实现一个决策树模型很简单？

在分布式环境下，要考虑：

- 数据如何切片？
- 每个节点如何算子树？
- 怎么合并树结构？

🫠 一句话：实现起来，真的很累……

所以 Mahout 提供了封装好的算法库，减轻(**jiǎn qīng**)大家的负担(**fù dān**)！

## Slide 34: Mahout의 장점 요약

Mahout 팀이 기존에 쓰이던 데이터 마이닝 알고리즘을 전부 MapReduce로 돌아가게改造해서，그냥 갖다 쓸 수 있는 알고리즘 라이브러리로 만들어提供了。

## Slide 35:

Mahout가 제공하는 두 가지 큰 장점:

📦 인터페이스 호출만으로 사용 가능, 복잡한 MapReduce 코드를 직접 작성할 필요 없음

🎯 파라미터만 전달하면, 나머지 계산은 시스템이 분산 병렬로 처리!

배달 음식 주문처럼: 메뉴만 선택하고 주소만 입력하면, 나머지는 플랫폼에 맡기면 됩니다!

## Slide 36: Ambari

Ambari는 Hadoop 운영자와 관리자를 위한 필수 도구로, 개발자나 데이터 분석가보다는 인프라를 관리하는 사람이 주로 사용합니다.

비유하자면, "Hadoop 클러스터의 공용 리모컨"이라고 볼수 있습니다.

Ambari라는 통합 웹 리모컨에서 버튼 클릭만으로 모든 것을 관리할 수 있습니다.

## Slide 34: Mahout 的优势总结

Mahout 团队把以前常用的那些数据挖掘算法，都给改造成了能用 MapReduce 跑的算法库，直接提供给大家用了

## Slide 35:

Mahout 给大家带来两个大优势：

📦 调用接口即可使用，不用手写复杂 MapReduce 代码

🎯 只要传参数，后面计算系统帮你分布式并行搞定(**gǎo ding**)！

就像点外卖：你只需要选菜 + 输入地址，剩下的全交给平台！

## Slide 36: Ambari

Ambari是面向Hadoop运维人员和管理员的必备工具，主要使用者是管理基础设施的人员，而非开发人员或数据分析师。

打个比方，可以把它看作是"Hadoop集群的公用遥控器"。

通过Ambari这个统一的网页遥控器，只需点击按钮就能管理所有操作

**Slide 37: Spark가 MapReduce를 대체하기 시작?**

주의 ⚠️

🗓️ 2015년부터 Spark가 점차 MapReduce를 대체하여 주류가 되기 시작했습니다.

왜일까요?

- 빠른 속도: 인메모리 계산 지원
- 풍부한 인터페이스: SQL, 머신러닝, 그래프 계산 지원
- 프로그래밍이 간단하고 효율적
-

🧠 전기차가 내연기관 차를 대체하는 것처럼, MapReduce는 더 이상 유일한 선택이 아닙니다!

**Slide 38: Mahout 팀도 Spark로 전환**

Mahout 개발팀도 추세를 느꼈습니다:

- 원래 MapReduce 기반 알고리즘이 점점 느려짐
- 개발 비용이 높고 유연하지 않음
- 사용자들이 Spark로 이동하기 시작

📢 최종적으로 Mahout 공식 팀은 선언했습니다: MapReduce 기반 버전은 더 이상 업데이트하지 않고, 전면적으로 Spark 플랫폼으로 전환!

여러분, 이것이 무엇을 말해주나요?

**기술은 시대를 따라가야 합니다!(技术要跟上时代)**

**倒戈 (Dàogē)**: "편을 바꾸다", "변절하다" (정치, 스포츠, 기업 등 다양한 맥락에서 사용)

**Slide** 37: : Spark 开始替代 MapReduce？

注意 ⚠️

🗓️ 从 2015 年起，Spark 开始逐渐替代( **zhújiàn tìdài)** MapReduce 成为主流。

为什么？

- 速度快：支持内存计算
- 接口丰富：支持 SQL、机器学习、图计算
- 编程简单，效率高
-

🧠 就像电动车取代燃油车(rán yóu chē)一样，MapReduce 已经不再是唯一选择！

**Slide** 38: Mahout 团队转向 Spark

Mahout 的开发团队也感受到了趋势：

- 原来基于 MapReduce 的算法越来越慢
- 开发成本高，不灵活
- 用户开始流向 Spark

📢 最终 Mahout 官方宣布：不再更新基于 MapReduce 的版本，全面转向 Spark 平台！

同学们，这说明什么？——

技术要跟上时代！

Dàogē "편을 바꾸다", "변절하다" (정치, 스포츠, 기업 등 다양한 맥락에서 사용).

## Slide 39: Spark가 점차 MapReduce를 대체

2015년부터 사람들은 발견했습니다:
🌀 MapReduce의 처리 방식은 안정적이지만 속도가 느리고 개발이 복잡합니다.
그래서 차세대 기술인 **Spark**가 등장했습니다!

지원하는 것들:

- 인메모리 계산, 빠른 속도 🚀
- 배치 처리 + 스트리밍 처리 지원
- 머신러닝, 그래프 계산 지원

📢 그래서 Spark가 점차 MapReduce를 대체하여 주류가 되고 있습니다!

## Slide 40: Mahout 팀도 Spark로

Mahout 프로젝트는 원래 MapReduce 위에서 실행되었습니다.

하지만 Spark가 떠오르면서 개발자들은 깨달았습니다: 👨‍💻 "계속 MR로 개발하기 너무 힘들다, Spark로 전환하자!"

그래서 Mahout 팀은 MapReduce 버전의 유지보수를 중단하고 전면적으로 Spark 플랫폼을 지원한다고 발표했습니다.

📌 이것은 말해줍니다: 기술 발전은 기다려주지 않으며, 따라가지 못하면 도태됩니다!

## Slide 39: Spark 逐渐取代 MapReduce

从 2015 年开始，大家发现：
🌀 MapReduce 的处理方式虽然稳，但速度慢、开发复杂。

于是新一代技术——Spark 上场(shàng chǎng)了！

它支持：

- 内存计算，速度快 🚀
- 支持批处理 + 流处理
- 支持机器学习、图计算

📢 所以，Spark 正在逐步(zhúbù)取代 MapReduce 成为主流！

## Slide 40: Mahout 团队也投向 Spark

Mahout 这个项目原本是跑在 MapReduce 上的。

但随着 Spark 兴起，开发者意识到：
👨‍💻 “继续写 MR 太费劲了，我们转 Spark 吧！”

于是 Mahout 团队宣布不再维护 MapReduce 版本，全面支持 Spark 平台。

📌 这说明了：技术发展不等人，跟不上就淘汰(táo tài)！

## Slide 41: Hive

이 페이지의 그림은 매우 중요합니다!

📌 Hadoop의 아키텍처는 전형적인 **"마스터-슬레이브"(Master-Slave)** 구조입니다:

- **NameNode**: 파일 디렉토리와 분산 정보 관리 담당 (지휘관처럼)
- **DataNode**: 실제 데이터 저장 담당 (병사처럼)

📦 대용량 파일은 분할되어 여러 DataNode에 저장되도록 할당됩니다.
NameNode는 그들의 위치를 기록합니다.
택배 창고 시스템처럼: 본부는 각 패키지가 어느 분산 창고에 있는지 알고 있습니다!

## Slide 42: Zookeeper

ZooKeeper는 분산 시스템을 구성하는 여러 노드들이 공통으로 참조해야 하는 설정 정보를 중앙에서 집중적으로 저장하고 관리하는 역할을 수행합니다.

대표적으로 데이터베이스 연결 정보나 시스템 전반의 환경 변수 등이 해당되며, 이러한 설정이 변경될 경우 각 노드에 일관되게 전파될 수 있도록 합니다.

## Slide 41: Hadoop 的一主多从架构

这一页图非常重要！

📌 Hadoop 的架构是典型的“一主多从”（Master-Slave）结构：

- NameNode 负责管理文件目录和分布信息（像指挥官 zhǐ huī guān）
- DataNode 负责实际存数据（像士兵 shìbīng）

📦 大文件会被切分，分配给多个 DataNode 存储。
NameNode 记录它们的位置。
就像一个快递仓库系统：总部(**zǒng bù**)知道每个包裹(**bāo guǒ**)在哪个分仓！

## Slide 42: 系统结构图细化

ZooKeeper负责集中存储和管理分布式系统中多个节点需要共同引用的配置信息。

例如数据库连接信息或系统全局环境变量等，当这些配置发生变更时，ZooKeeper能确保其一致地传播到各个节点。

## Slide 43: 로그 수집 및 분석 시나리오

실제로 많은 회사들이 Hadoop을 사용하여 로그 분석을 합니다:

📄 예를 들어:

- 사용자 방문 로그 (클릭, 브라우징)
- 오류 로그 (404, 타임아웃)
- 행동 경로 (이벤트 추적 데이터)

이런 데이터는 매일 대량으로 생성되며, 전통적인 도구로는 감당할 수 없습니다.

📌 Hadoop은 Flume + Hive를 결합하여 로그 자동 수집 → 저장 → 분석을 구현할 수 있습니다

## Slide 44: Hadoop 모듈 간의 협업

이 페이지는 매우 중요한 점을 강조합니다:

📌 Hadoop의 각 모듈은 고립되어 있지 않고, **고도로 협업합니다**!

예를 들어:

- Hive 쿼리 데이터는 HDFS에 저장됨
- Mahout 분석 모델은 Hive가 생성한 중간 테이블을 사용
- Pig는 Sqoop을 통해 외부 데이터베이스 데이터를 가져올 수 있음

📌 그들 사이는 패스를 주고받는 "농구팀"처럼, 패스가 부드럽고 협력이 완벽합니다!

## Slide 43: 日志采集与分析场景

现实中，有很多公司使用 Hadoop 来做日志分析：

📄 比如：

- 用户访问日志（点击、浏览）
- 错误日志（404、超时）
- 行为路径（埋点数据）

这类数据每天产生海量，靠传统工具根本应付不了**yìngfù bú liǎo**。

📌 Hadoop 可以结合 Flume + Hive 实现日志自动采集 → 存储 → 分析

## Slide 44: Hadoop 模块之间的协作

这页强调了一个很重要的点：

📌 Hadoop 各模块之间并不是孤立的，而是高度协作的！

例如：

- Hive 查询的数据存放在 HDFS
- Mahout 分析模型用的是 Hive 生成的中间表
- Pig 可以接入 Sqoop 导入外部数据库数据

📌 它们之间像打配合的"篮球队"，传球顺畅(**shùnchàng**)，配合默契(**pèihé mòqì**)！

## Slide 45: MapReduce의 두 가지 핵심 함수

MapReduce의 핵심은 두 가지 함수입니다:

- **Map**: "분할 + 초기 처리" 담당
- **Reduce**: "집계 + 최종 병합" 담당

📌 간단한 예: 반 학생들이 각자 몇 개의 주먹밥을 먹었는지 통계를 내려면:

- **Map 단계**: 각자 자기가 몇 개 먹었는지 센다
- **Reduce 단계**: 반장이 수거해서 총합과 평균을 계산한다

이것이 "분업-집계"의 논리입니다!

## Slide 45: MapReduce 的两个核心函数

MapReduce 的核心是两个函数：

- Map：负责"分片 + 初步处理"
- Reduce：负责"汇总 + 最终合并"

📌 举个简单例子：

你要统计一个班级谁吃了多少饭团( fàn tuán)：

- Map 阶段：每个人数自己吃了几个
- Reduce 阶段：班长收上来，统计总数和平均值

就是这么一套"分工–聚合"的逻辑！

## Slide 46: MapReduce 프로그래밍의 단순화 장점

많은 학생들이 MapReduce 프로그래밍이 어려울까 걱정하는데, 사실 걱정할 필요 없습니다!

✅ 가장 큰 장점은: **복잡한 세부 사항을 숨김**

여러분은 구현만 하면 됩니다:

- **Map()**: 처리 로직 작성
- **Reduce()**: 병합 로직 작성

다음은 어떻게 하나요:

- 데이터를 어떻게 분할할까?
- 노드를 어떻게 조정할까?
- 작업을 어떻게 실행할까?

👉 모두 Hadoop이 자동으로 처리해줍니다!

## Slide 46: MapReduce 编程的简化优势

很多同学担心 MapReduce 编程会不会很难，其实不用怕！

✅ 它最大的优点就是：屏蔽(**píng bì**)复杂细节

你只要负责实现：

- Map()：写处理逻辑
- Reduce()：写合并逻辑

至于：

- 数据怎么切？
- 节点怎么调？
- 任务怎么跑？

👉 全都 Hadoop 自动帮你搞定！

## Slide 47: "분할 정복"의 사고방식

MapReduce 뒤에 있는 철학은: **분할 정복 (Divide and Conquer)**

📦 큰 작업을 여러 작은 작업으로 나누어 여러 노드가 동시에 처리하게 하고, 마지막에 결과를 병합합니다.

논문을 쓰는데 밤새고 싶지 않다면 어떻게 할까요?

- 네 명의 팀원에게 각자 한 장씩 쓰게 하고, 마지막에 합쳐서 제출하면 됩니다! 😂

🎤 "분할 정복" 전략의 핵심 사상

MapReduce의 가장 핵심적인 계산 사고는: **분할 정복**

무슨 뜻일까요?
📦 큰 작업을 작은 작업으로 나누어 다른 기계에 배분 (Map) 그런 다음 결과를 통합 집계 (Reduce)

학교 청소를 예로 들면, 한 사람이 하면 너무 느리니까, 1학년은 운동장 청소, 2학년은 복도 청소, 마지막에 모두 합쳐서 "깨끗한 캠퍼스"를 만들면 됩니다 😆

## Slide 47: “分而治之”的思维

MapReduce 背后的哲学叫做：
分而治之（Divide and Conquer）

📦 就是把一个大任务拆成(**chāi chéng**)若干小任务，分给多个节点同时处理，最后再合并结果。

就像你写论文不想通宵怎么办？

- 请四个队友一人写一章，最后拼起来交稿！😂

🎤 第46页：“分而治之”策略(**cè lüè**)的核心思想
MapReduce 最核心的计算思维叫：分而治之

什么意思呢？
📦 把大任务拆成小任务，分给不同机器（Map）
然后再统一归总结果（Reduce）

就像打扫学校，一个人太慢了，一年级扫操场，二年级扫楼道，最后大家一起合成“干净校园”😆

## Slide 48: MapReduce의 완전한 흐름도

이 다이어그램은 특히 중요하니 주의 깊게 보세요!
흐름은 다음과 같습니다:

1. 데이터가 여러 "조각"으로 분할됨
2. 각 조각이 Map() 작업에 들어가 처리됨
3. 중간 결과가 key에 따라 집계됨 (Shuffle)
4. 동일한 key가 같은 Reduce()에 할당되어 최종 집계
5. 여러 결과 파일 출력

📌 이것이 MapReduce의 실행 엔진 메커니즘입니다!

## Slide 48: MapReduce 的完整流程图

这张图特别关键，请大家认真看！

流程如下：

1. 数据被切成多个“分片”
2. 每个分片进入一个 Map() 任务处理
3. 中间结果会根据 key 进行聚合（Shuffle）
4. 相同 key 分配到同一个 Reduce() 中做最终汇总
5. 输出多个结果文件

📌 这就是 MapReduce 的执行引擎机制！

## Slide 49: YARN은 "하나의 클러스터, 여러 프레임워크" 지원

예전 Hadoop 클러스터는 MapReduce만 실행할 수 있어서 매우 낭비였습니다!

📌 YARN의 등장으로 완전히 바뀌었습니다!
슬로건은: **One cluster, multiple engines**
즉, 동일한 클러스터에서 동시에 실행 가능:

- MapReduce
- Spark
- Storm
- Impala 등등!

리소스도 동적으로 할당 가능! 매우 유연합니다!

## Slide 49: YARN 支持"一集群多框架"

以前的 Hadoop 集群只能跑 MapReduce，很浪费！

📌 YARN 的出现，彻底改变了这一点！

它的口号是：One cluster, multiple engines
也就是说，同一个集群，可以同时跑：

- MapReduce
- Spark
- Storm
- Impala 等等！

资源也能动态调配！非常灵活！

## Slide 50: 기업의 다양한 비즈니스 요구

기업에는 다양한 비즈니스 요구가 있습니다:

- 배치 작업 실행 필요 (MapReduce)
- 실시간 분석 필요 (Storm)
- 대화형 쿼리 필요 (Impala)
- 머신러닝 실행 필요 (Spark)

📌 한 클러스터가 하나의 프로그램만 실행할 수 있다면 너무 낭비입니다! YARN이 이런 "다중 프레임워크 공존"의 스케줄링을 처리해줍니다!

## Slide 50: 企业的多业务需求

企业中有各种各样的业务需求：

- 需要跑批任务（MapReduce）
- 需要做实时分析（Storm）
- 需要做交互式查询（Impala）
- 还想跑机器学习（Spark）

📌 如果一个集群只能运行一个程序，那未免 **(wèimiǎn)** 太浪费了！
YARN 就帮我们搞定这种"多框架并存"的调度！

## Slide 51: 각 프레임워크의 전형적인 용도

이 페이지는 Hadoop 생태계의 4대 프레임워크 비교입니다:

| 프레임워크 | 주요 용도 |
| --- | --- |
| MapReduce | 오프라인 배치 처리 |
| Impala | 실시간 쿼리 분석 |
| Storm | 스트리밍 실시간 처리 |
| Spark | 반복 계산 / 인메모리 처리 |

📌 각 프레임워크는 "전문 선수"와 같습니다: 어떤 것은 장거리 달리기(MapReduce), 어떤 것은 단거리 달리기(Impala), 어떤 것은 춤(Storm)에 능합니다 😆

📢 각 프레임워크가 각자의 역할을 하는 것은 서로 다른 전공의 선생님들이 수업하는 것과 같습니다!

## Slide 51: 各种框架的典型用途

这页列出的是 Hadoop 生态中四大框架的对比：

| 框架 | 主要用途 |
| --- | --- |
| MapReduce | 离线批处理 |
| Impala | 实时查询分析 |
| Storm | 流式实时处理 |
| Spark | 迭代计算 / 内存处理 |

📌 每个框架像一位"专业选手(xuǎnshǒu)"：有的擅长(shàncháng)长跑（MapReduce），有的擅长短跑（Impala），有的跳舞 tiàowǔ（Storm）😆

📢 每种框架各司其职，就像不同专业的老师授课一样！

## Slide 52: 여러 프레임워크가 서로 다른 팀에 의해 유지보수됨

하지만 문제도 있습니다:
이 프레임워크들은 종종 다른 팀이 개발하고, 다른 언어로 작성됩니다:

- MapReduce는 Java 사용
- Spark는 Scala, Python 지원
- Impala는 C++로 작성
- Storm은 고유한 스타일

📌 이로 인해: 환경이 복잡하고, 리소스 충돌, 배포가 번거로워집니다!

## Slide 53: 존재하는 문제 - 비용 및 리소스 낭비

각 프레임워크가 자체적으로 "환경 구축, 리소스 점유"를 하면 무슨 일이 벌어질까요?

🤯 문제들:

- 높은 비용 (각 팀이 서버 구축)
- 리소스 유휴 (일부 클러스터가 공회전)
- 관리 어려움 (버전 충돌)

따라서 통합 스케줄링을 하는 "큰 관리자"가 필요합니다 - YARN!

## Slide 52: 多个框架由不同团队维护

但问题也来了：

这些框架常常是不同团队开发、不同语言写的：

- MapReduce 用 Java
- Spark 支持 Scala、Python
- Impala 是 C++ 写的
- Storm 有自己的风格

📌 这就导致：环境复杂、资源冲突、部署麻烦！

## Slide 53: 存在的问题：成本与资源浪费

如果每个框架都自己"搭环境、占资源 **zhàn zīyuán**"，会发生什么？

🤯 问题包括：

- 成本高（每个团队都搭服务器）
- 资源闲置**xián zhì**（有的集群空转）
- 系统难管理（版本冲突**chōng tū**）

所以必须有一个统一调度的"大管家"——YARN！

## Slide 54: YARN 등장, 통합 스케줄링

이 다이어그램은 이상적인 상태를 보여줍니다:
📌 Spark, MapReduce, Impala, Storm이 모두 동일한 YARN 플랫폼에서 실행됩니다.

- 중복 배포 없음
- 리소스 공유 사용 가능
- 어떤 작업이 바쁘면 우선적으로 리소스 제공!

"서로 다른 임차인"이 하나의 데이터 센터를 공유하는 것으로 이해할 수 있습니다!

## Slide 55: YARN이 이러한 프레임워크를 어떻게 스케줄링하나?

YARN의 핵심 기능:

- 리소스 스케줄링 (Resource Scheduling)
- 작업 생명주기 관리

다중 애플리케이션 동시 지원
배치 처리, 대화형 쿼리, 그래프 계산, 스트리밍 처리 모두 동일한 플랫폼에서 실행할 수 있습니다!

📌 큰 관리자처럼, 온 가족의 냉장고, 전기밥솥, 전기 계량기를 관리하며, 누가 배고프면 먼저 사용하게 합니다! 😂

## Slide 54: YARN 登场统一调度

这张图展示了一个理想状态：

📌 Spark、MapReduce、Impala、Storm 都跑在同一个 YARN 平台上。

- 不再重复部署
- 资源可以共享使用
- 哪个任务忙，优先给资源！

你可以理解为：不同"租户(**zù hù**)"共享一座数据中心！

## Slide 55: YARN 如何调度这些框架？

YARN 的核心功能是：
资源调度（Resource Scheduling）

- 任务生命周期管理
- 多应用并发支持

无论你是批处理、交互查询、图计算、流处理，都可以跑在同一个平台上！

📌 它就像一个大管家，管理全家人的冰箱、电饭锅、电表，谁饿了先用！😂

## Slide 56: YARN 위의 계산 프레임워크 전체 그림

마지막 두 페이지는 Hadoop 2.0의 가장 중요한 다이어그램을 보여줍니다:
📍 YARN 플랫폼에서 다양한 유형의 애플리케이션을 배포할 수 있습니다:

| 유형 | 프레임워크 |
|---|---|
| BATCH | MapReduce |
| STREAMING | Storm, S4 |
| IN-MEMORY | Spark |
| INTERACTIVE | Tez |
| GRAPH | Giraph |
| ONLINE | HBase |
| HPC | OpenMPI |
| SEARCH | Weave 등 |

📌 통합 리소스 플랫폼, 통합 저장 시스템(HDFS2), 통합 스케줄링 정책!
이것이 현대 데이터 플랫폼의 "국가 중앙 플랫폼" 전략입니다!

---

## Slide 57: (Slide 56과 동일한 내용 반복)

## Slide 56: YARN 上的计算框架全景图

最后这两页展示了 Hadoop2.0 最重要的一张图：
📍 在 YARN 平台上，我们可以部署各种类型的应用：

| 类型 | 框架 |
|---|---|
| BATCH | MapReduce |
| STREAMING | Storm、S4 |
| IN-MEMORY | Spark |
| INTERACTIVE | Tez |
| GRAPH | Giraph |
| ONLINE | Hbase |
| HPC | OpenMPI |
| SEARCH | Weave 等 |

📌 统一资源平台、统一存储系统（HDFS2）、统一调度策略！
这就是现代数据ロ平台的"国家中台"战略！

### Slide 58-67: 퀴즈 정답

- Slide 58: ✅ 정답: C
- Slide 59: ✅ 정답: C
- Slide 60: ✅ 정답: D
- Slide 61: ✅ 정답: B
- Slide 62: ✅ 정답: C
- Slide 63: ✅ 정답: D
- Slide 64: ✅ 정답: B
- Slide 65: ✅ 정답: C
- Slide 66: ✅ 정답: B
- Slide 67: ✅ 정답: C

---

### Slide 68: 강의 마무리

좋습니다, 오늘 수업은 여기까지입니다!
오늘 우리는 **"빅데이터 시대"**의 전체 과정을 알아보았습니다. 정보화 물결, 세 가지 핵심 지원 기술부터 데이터가 어떻게 생성되고 어떻게 진화하는지까지 살펴보았습니다.
다음 수업에서는 계속해서 **"빅데이터의 핵심 기술"**을 탐구하겠습니다.
마지막 질문: 오늘 여러분도 매일 실제로 "데이터를 생산"하고 있다는 것을 발견하셨나요? 😉
감사합니다, 여러분! 수업 마치겠습니다!

### Slide 58-67: 퀴즈 정답

- Slide 58: ✅ 정답: C
- Slide 59: ✅ 정답: C
- Slide 60: ✅ 정답: D
- Slide 61: ✅ 정답: B
- Slide 62: ✅ 정답: C
- Slide 63: ✅ 정답: D
- Slide 64: ✅ 정답: B
- Slide 65: ✅ 정답: C
- Slide 66: ✅ 정답: B
- Slide 67: ✅ 정답: C

### Slide 68:

好了，我们本节课到这里就结束啦！

今天我们认识了"大数据时代"的来龙去脉，从信息化浪潮、三大支撑技术，到数据如何产生、如何演变。

下节课我们会继续探讨"大数据的关键技术"。

最后提问一下：你今天有没有发现，自己每天其实也在"生产数据"呢？😉

谢谢大家，下课！👏👏👏

**부록: 주요 중국어 용어 및 발음**

| 중국어 | 병음 | 한국어 의미 |
|---|---|---|
| 彻底改变 | chèdǐ gǎibiàn | 완전히 바꾸다 |
| 干活 | gànhuó | 일하다 |
| 车厢 | chēxiāng | 객차 |
| 轨道 | guǐdào | 선로 |
| 调度系统 | diàodù xìtǒng | 스케줄링 시스템 |
| 调度 | diàodù | 스케줄링하다 |
| 协调 | xiétiáo | 협력적 조정 |
| 链条 | liàntiáo | 체인 |
| 各司其职 | gè sī qí zhí | 각자 자신의 역할을 다하다 |
| 全家福 | quánjiāfú | 가족사진 |
| 旁边 | pángbiān | 주변 |
| 围绕 | wéirào | 둘러싸다 |
| 飞驰 | fēichí | 쏜살같이 달리다 |
| 卡车 | kǎchē | 트럭 |
| 倒戈 | dàogē | 편을 바꾸다, 변절하다 |