科研训练（第二讲）

科研文献阅读

从按需学习到发现研究问题

晏潇 博士 | 武汉大学 · 武汉数学与智能研究院 | yanxiaosunny@whu.edu.cn

为什么读了很多论文，仍然找不到研究问题？

“我都看懂了，但不知道下一步做什么。”

没有目的

每篇都从头读，阅读深度与当前任务无关。

没有结构

只记方法名称，没有重建问题、假设与证据链。

没有比较

逐篇摘要，没有在共同维度上寻找差异与权衡。

本讲目标：把“读过”推进到“能比较、能质疑、能定义问题”。

本讲路线：找 → 筛 → 读 → 懂 → 比 → 问

1. **明确目的**
   阅读目标卡
2. **检索筛选**
   Search Log
3. **三步阅读**
   Reading Note
4. **核心掌握**
   实现 / 讲解
5. **归纳比较**
   比较表
6. **定义问题**
   Problem Statement

贯穿原则：目标决定检索范围，检索范围决定阅读深度，比较结果决定问题定义。

本讲结束时，你应该能交出什么？

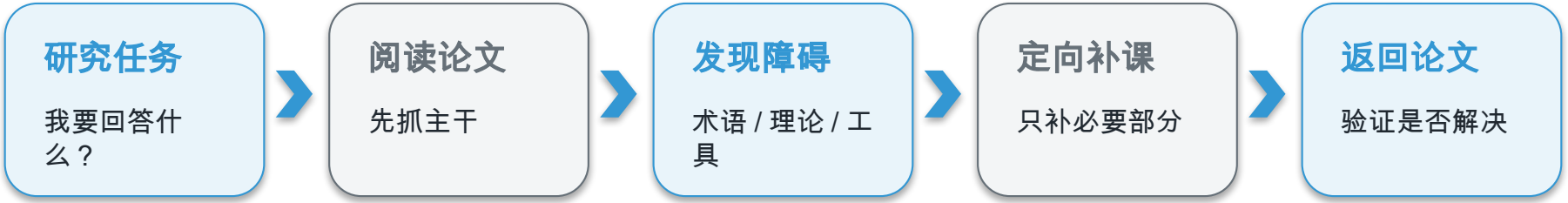
01	**检索记录**	检索式、来源、筛选理由
02	**论文阅读卡**	问题、洞察、方法、证据
03	**核心掌握记录**	实现、复现或 Teach-back
04	**文献比较表**	共同维度、差异与权衡
05	**候选问题陈述**	场景、限制、机理与目标

科研文献阅读，不是教材学习的缩小版

维度	教材学习	科研文献阅读
目标	建立系统知识体系	回答当前研究任务中的问题
知识状态	相对成熟、结构稳定	前沿、可能争议或错误
阅读顺序	通常按章节递进	允许跳读、回读和补背景
可信方式	依赖教材体系与教学筛选	主动检查假设、方法和证据
完成标准	理解并掌握知识点	形成判断、比较与下一步行动

核心转换：从“学完再用”转向“为了完成任务而按需学习”。

按需学习：遇到障碍时再补背景



判断标准：补背景之后，是否更接近当前阅读产出？

如果没有，就先记录问题，避免掉进“无限补课”的支线。

同一篇论文，因为目的不同，读法也不同

了解领域

回答：这个方向在研究什么？
深度：第一遍为主

解决具体问题

回答：别人如何处理相似困难？
深度：第二遍 + 定位细节

复现或采用方法

回答：方法如何运行、条件是什么？
深度：第三遍 + 实现

寻找研究问题

回答：共识、冲突、边界和权衡是什么？
深度：多篇比较

阅读深度不是论文决定的，而是“论文与当前任务的关系”决定的。

开始检索前，先写一张阅读目标卡

目的	我需要为选题、复现、方案还是写作服务？
核心问题	阅读后必须能回答哪一个具体问题？
时间预算	10 分钟、 1 小时，还是半天？
完成标准	三句话、阅读卡、实现记录，还是比较表？

示例

我要判断：为什么 MapReduce 不适合迭代工作负载，并找到后续系统采用的改进机制。

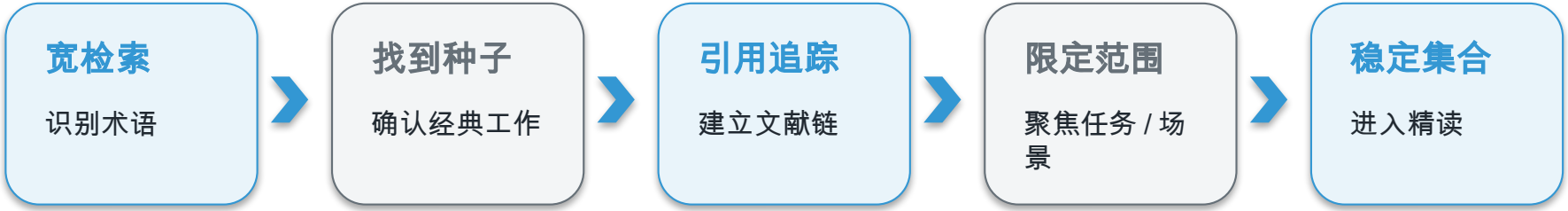
把研究兴趣拆成可检索的关键词

研究对象	任务	方法	场景	指标
cluster computing data processing	iterative analytics interactive query	in-memory dataflow	large scale fault tolerant	latency I/O overhead

组合方式 **对象 + 任务 + 方法 / 场景 + 指标**

示例： "iterative data processing" AND "in-memory" AND fault tolerance

检索不是一次命中，而是逐轮收敛



每轮只问两件事：还缺哪类论文？哪些关键词需要修改？

种子论文 + 双向引用追踪

向后追：References

找到概念来源、基础方法和被继承的假设。

种子论文

与问题高度相关、论证结构清晰、引用网络丰富。

向前追： Cited by

找到后续改进、反例、扩展场景与新问题。

领域地图不是搜索引擎一次给出的，而是沿引用关系逐步长出来的。

检索来源与 Search Log

Google Scholar 跨领域入口与被引追踪

DBLP 计算机会议 / 期刊书目信息

ACM / IEEE 正式出版版本与元数据

arXiv 最新预印本，需额外核验

Search Log 最少记录 日期 | 检索式 | 数据库 | 结果数 | 保留论文 | 排除原因 | 下一轮关键词

参考： S. Keshav, How to Read a Paper, 2007.

阅读价值由“质量”与“当前相关性”共同决定

课堂任务：根据材料信息卡判断阅读优先级

本次目标　理解“传统批处理为何不适合迭代分析，以及后续系统怎样改进”。

A　MapReduce 研究论文
定义传统批处理模型 | 正式会议论文 | 直接相关

B　RDD 研究论文
聚焦迭代与交互分析 | 正式会议论文 | 直接相关

C　Hadoop 用户手册
说明软件如何配置和使用 | 工程文档 | 间接相关

D　Spark 介绍文章
快速解释主要功能 | 二手材料 | 需回到原文核验

课堂任务
2 分钟分类

提交物　分类结果 + 一句判断依据

参考答案：优先阅读与目标直接相关的原始证据

优先精读： A + B

两篇都是直接回答目标问题的原始研究论文，可构成“已有设计—后续改进”的证据链。

按需查阅： C

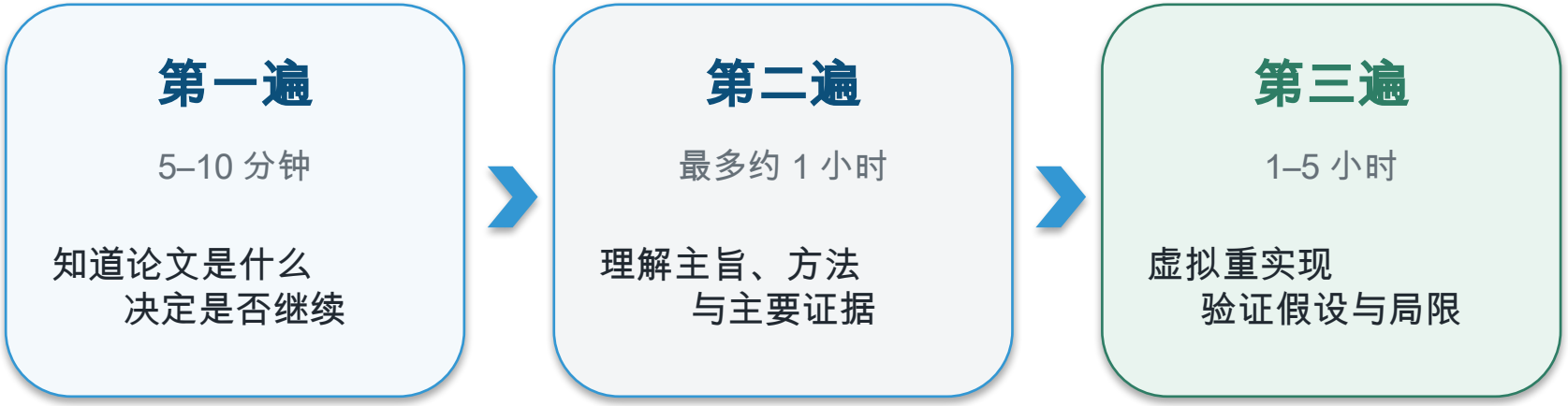
当任务转为实现 Hadoop 作业或核对配置细节时，用户手册价值很高。

暂作线索： D

适合快速建立直觉，但关键主张、数据和引用必须回到原论文核验。

这项练习只训练“价值判断”，不要求现场读懂任何一篇论文。

三步阅读法：筛选 → 理解 → 验证



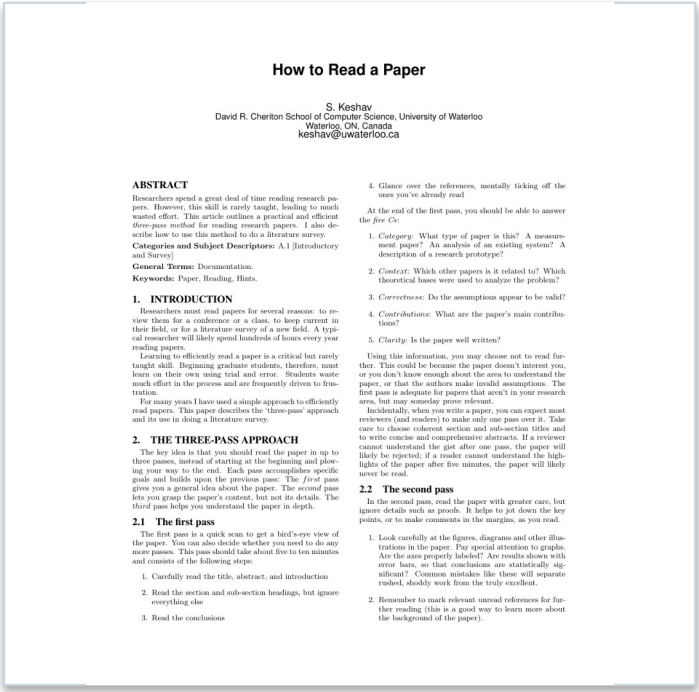
原则：阅读成本随“论文对当前任务的重要性”递增。

参考： S. Keshav, How to Read a Paper, 2007.

第一遍：先获得论文的鸟瞰图

1. 标题、摘要和引言
2. 章节与小节标题
3. 结论
4. 浏览参考文献并标记熟悉项

完成标准 能回答五个 C ，并决定：停止、补背景，还是进入第二遍。



How to Read a Paper

S. Keshav
David R. Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, Canada
keshav@uwaterloo.ca

ABSTRACT

Researchers spend a great deal of time reading research papers. However, this skill is rarely taught, leading to much wasted effort. This article outlines a practical and efficient *three-pass method* for reading research papers. I also describe how to use this method to do a literature survey.

Categories and Subject Descriptors: A.1 [Introductory and Survey]

General Terms: Documentation.

Keywords: Paper, Reading, Hints.

1. INTRODUCTION

Researchers must read papers for several reasons: to review them for a conference or a class, to keep current in their field, or for a literature survey of a new field. A typical researcher will likely spend hundreds of hours every year reading papers.

Learning to efficiently read a paper is a critical but rarely taught skill. Beginning graduate students, therefore, must learn on their own using trial and error. Students waste much effort in the process and are frequently driven to frustration.

For many years I have used a simple approach to efficiently read papers. This paper describes the 'three-pass' approach and its use in doing a literature survey.

2. THE THREE-PASS APPROACH

The key idea is that you should read the paper in up to three passes, instead of starting at the beginning and plowing your way to the end. Each pass accomplishes specific goals and builds upon the previous pass: The *first* pass gives you a general idea about the paper. The *second* pass lets you grasp the paper's content, but not its details. The *third* pass helps you understand the paper in depth.

2.1 The first pass

The first pass is a quick scan to get a bird's-eye view of the paper. You can also decide whether you need to do any more passes. This pass should take about five to ten minutes and consists of the following steps:

1. Carefully read the title, abstract, and introduction
2. Read the section and sub-section headings, but ignore everything else
3. Read the conclusions
4. Glance over the references, mentally ticking off the ones you've already read

At the end of the first pass, you should be able to answer the *five Cs*:

1. *Category*: What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?
2. *Context*: Which other papers is it related to? Which theoretical bases were used to analyze the problem?
3. *Correctness*: Do the assumptions appear to be valid?
4. *Contributions*: What are the paper's main contributions?
5. *Clarity*: Is the paper well written?

Using this information, you may choose not to read further. This could be because the paper doesn't interest you, or you don't know enough about the area to understand the paper, or that the authors make invalid assumptions. The first pass is adequate for papers that aren't in your research area, but may someday prove relevant.

Incidentally, when you write a paper, you can expect most reviewers (and readers) to make only one pass over it. Take care to choose coherent section and sub-section titles and to write concise and comprehensive abstracts. If a reviewer cannot understand the gist after one pass, the paper will likely be rejected; if a reader cannot understand the highlights of the paper after five minutes, the paper will likely never be read.

2.2 The second pass

In the second pass, read the paper with greater care, but ignore details such as proofs. It helps to jot down the key points, or to make comments in the margins, as you read.

1. Look carefully at the figures, diagrams and other illustrations in the paper. Pay special attention to graphs. Are the axes properly labeled? Are results shown with error bars, so that conclusions are statistically significant? Common mistakes like these will separate rushed, shoddy work from the truly excellent.
2. Remember to mark relevant unread references for further reading (this is a good way to learn more about the background of the paper).

参考： S. Keshav, How to Read a Paper, 2007.

第一遍的“五个 C”

Category	**论文类型**	测量、分析、原型或理论？
Context	**研究语境**	与哪些工作和理论相关？
Correctness	**初步正确性**	关键假设看起来成立吗？
Contributions	**主要贡献**	提出了什么新的东西？
Clarity	**表达清晰度**	论证结构是否容易理解？

参考： S. Keshav, How to Read a Paper, 2007.

课堂任务：从裁剪材料中识别论文结构

A 处理 TB 级数据时，计算逻辑并不复杂；困难在于数据分片、并行调度、通信和机器故障。

B 用户只定义 Map 和 Reduce 函数，框架自动完成并行化、数据传输、调度和失败任务重执行。

C 论文展示了 1 TB 排序实验，并在排序过程中主动杀死 200 个 worker 测试故障恢复。

任务 把 A/B/C 分别标为 Problem 、 Insight / Method 、 Evidence ；再用一句话概括论文。

课堂任务
4 分钟归类

提交物 三项归类 + 一句话概括

材料改写自： Dean & Ghemawat, MapReduce, OSDI 2004.

参考拆解：从信息卡重建 MapReduce 论证

A → Problem	大规模批处理的核心困难，是分布式执行细节掩盖了简单的计算逻辑。
B → Insight / Method	通过受限的 Map/Reduce 接口，让框架自动负责并行化、通信、调度与容错。
C → Evidence	TB 级实验和 worker 故障实验用于支持可扩展性与容错主张。
一句话概括	MapReduce 用简单的函数式接口封装大规模集群执行细节，并通过 TB 级任务和故障实验验证其可扩展性与容错能力。

论文： Dean & Ghemawat, MapReduce, OSDI 2004.

第二遍：理解方法与主要证据

读什么

正文主干；暂时跳过证明、复杂推导和非关键实现细节。

怎么看

仔细检查图、表、坐标、单位、误差、样本和实验条件。

记什么

关键观点、疑问、未读引用，以及 Claim 与 Evidence 的对应关系。

完成标准

能够向别人说明论文主旨、方法和最重要的支持证据。

不要问“实验多不多”，而要问“这些实验支持了哪些主张”。

参考： S. Keshav, How to Read a Paper, 2007.

用 Claim—Evidence 表检查论文论证

核心主张	论文中的主要证据	证据边界
可扩展处理 TB 级数据	1 TB 排序约 891 秒；分布式 grep 约 150 秒	特定 Google 集群和工作负载
机器故障可透明恢复	排序中杀死 200 个 worker ，总时间约增加 5%	未透明恢复 master 故障
备份任务缓解拖尾	禁用备份后 891 秒增至 1283 秒	资源较空闲的实验环境
简化应用开发	索引阶段约 3800 行降到约 700 行	单个内部应用案例

数据来自： Dean & Ghemawat, MapReduce, OSDI 2004.

第三遍：进行“虚拟重新实现”

1. **重建**
   不看原文画出问题、方法与实验
2. **挑战**
   找出显式和隐式假设
3. **比较**
   自己的方案与作者方案有何差异
4. **验证**
   检查实验或证明能否支持主张
5. **延伸**
   记录改进、反例和未来工作

第三遍的完成标准：能凭记忆重构论文，并指出强项、弱项和隐含假设。

参考： S. Keshav, How to Read a Paper, 2007.

阅读不是无限深入：给未知项分级

现在必须理解

直接影响论文核心问题、方法、主张或证据。

稍后补充

影响细节理解，但不妨碍当前阅读产出。

当前可以忽略

与当前目标无关的证明、实现或背景支线。

高效阅读不是“什么都懂”，而是知道什么必须现在懂。

什么样的论文值得成为“核心文献”？

直接相关

决定研究问题、方法或评估

反复出现

多篇工作共同引用或比较

代表性强

提出关键概念、基线或范式

可操作

能够用于实现、复现或方案设计

核心文献数量要少：宁可真正掌握 3 篇，也不要只记住 30 篇标题。

核心文献的四级掌握阶梯

1 复述

能说清论文做了什么

2 重建

能画出问题—方法—证据

3 实现 / 复现

能运行关键流程或结果

4 讲解

能让同伴理解并回答追问

用七段式提纲重建论文论证

Problem

问题与重要性

Prior assumption

已有工作的关键假设

Insight

打破旧假设的新洞察

Technical overview

方法、系统或证明

Proof

评估或证明

Impact

影响与后续变化

Title / Citation

论文基本信息与可追溯来源

评分关注： Accuracy · Completeness · Clarity

参考： Stanford CS197C, Assignment 1: Reading a Paper, 2025.

不同论文，“实现”具有不同含义

算法 / 系统	最小实现、流程追踪、复杂度与故障路径
机器学习	运行基线、复现关键指标或消融
理论	重建定义、定理条件、证明骨架和反例
HCI/ 实证	重建研究设计、变量、样本与分析流程

目标不是复刻全部结果，而是验证你真正依赖的核心机制或主张。

课堂任务：补全 WordCount 的后半流程

输入

to be or not to be

Map（已给出）

(to,1) (be,1) (or,1)
(not,1) (to,1) (be,1)

Shuffle

请按 key 分组

Reduce

请写最终计数

故障恢复选择题

保存中间结果的 Map worker 故障后： A. 重做丢失的 Map 任务 B. 忽略丢失数据
C. 重启全部集群

课堂任务

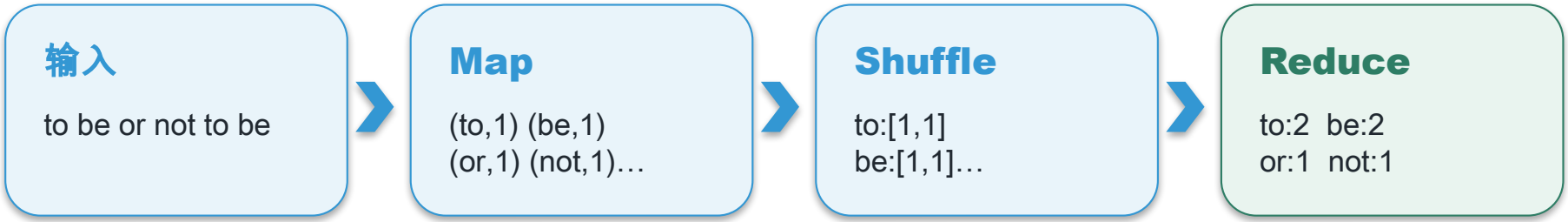
3 分钟补全

提交物

Shuffle 结果 + Reduce 结果 + 选项

案例基于： Dean & Ghemawat, MapReduce, OSDI 2004.

参考答案： WordCount 与故障恢复



Map worker 故障　已完成的 Map 任务也要重做：中间结果只在该 worker 本地磁盘；重新执行后由 master 通知 Reduce 新的位置。

隐含前提： Map/Reduce 函数应确定，重执行才能保持等价语义。

案例基于： Dean & Ghemawat, MapReduce, OSDI 2004.

核心结果复现：先写“复现契约”

字段	需要写清楚
待验证 Claim	哪一句主张需要证据？
环境与输入	硬件、软件、数据、规模和随机种子
对照与指标	与谁比较？测量什么？
目标结果	预期复现哪张图、哪个表或哪个数量级？
偏差与失败	与论文不同在哪里？可能原因是什么？
结论边界	复现结果能够支持什么，不能支持什么？

做表之前，先确定“比较目的”

找基线

比较任务、数据、指标和公开实现

找瓶颈

比较假设、资源和失败场景

选方法

比较效果、效率、成本和部署条件

找问题

比较共识、冲突、边界和权衡

表头不是固定模板，而是你要回答的问题的操作化。

用于发现问题的文献比较表

论文	场景 / 问题	关键假设	核心洞察	证据	效果	效率 / 开销	适应性 / 局限
A	...	...	...	...	...	...	...
B	...	...	...	...	...	...	...
C	...	...	...	...	...	...	...

读表时重点找　　反复出现的假设 | 无法同时满足的目标 | 特定场景下的失效 | 证据覆盖不到的范围

课堂任务：把事实卡放进比较表

- **A** 面向大规模批处理
- **B** 阶段间写入稳定存储
- **C** 失败任务重新执行
- **D** 面向迭代与交互分析
- **E** 工作集可在内存中持久化
- **F** 沿 lineage 重建丢失分区

比较维度	MapReduce	RDD / Spark
工作负载	____	____
数据共享	____	____
容错方式	____	____

课堂任务
4 分钟归类

提交物
将 A—F 填入对应单元格

事实改写自： MapReduce, OSDI 2004 ； RDD, NSDI 2012.

参考答案： MapReduce 到 RDD 的问题演进

维度	MapReduce	RDD / Spark
目标	大规模批处理	迭代算法与交互分析
数据共享	阶段 / 作业间经稳定存储	工作集可在内存中持久化
容错	任务重执行；依赖输入 / 中间数据	用 lineage 重建丢失分区
优势	简单、可扩展、容错成熟	减少重复 I/O ；迭代最高约 20×
边界	迭代与交互产生高 I/O 开销	不适合细粒度异步共享状态

问题演进：不是“ MapReduce 不好”，而是“工作负载改变后，原设计假设产生了新瓶颈”。

论文： MapReduce, OSDI 2004 ； RDD, NSDI 2012.

从四个维度扫描潜在研究问题

效果 / 质量

准确率、正确性、覆盖率、用户体验或结论可信度

效率

延迟、吞吐、收敛速度、扩展性或开发效率

开销

计算、内存、通信、能耗、标注、金钱与维护成本

适应性

数据、规模、硬件、任务、环境改变后是否仍成立

真正有价值的问题，往往出现在四者之间无法同时满足的权衡中。

从“观察到不足”走到“定义研究问题”

目标场景

在哪种数据、任务、规模或约束下？

已有方法

当前主流方法依赖什么关键假设？

可测不足

质量、效率、开销或适应性哪里失效？

可能机理

为什么会发生，而不只是发生了什么？

研究问题

能否改善主目标，同时控制副作用？

问题模板

在【场景】中，现有【方法】因【假设 / 机理】导致【可测不足】；能否【研究方向】改善【主目标】，同时控制【开销 / 风险】？

课堂任务：用关键词拼装一个研究问题

场景
迭代机器学习

现有方法
MapReduce

原有假设
跨轮经稳定存储共享

不足
重复 I/O 导致高延迟

改进方向
在内存中复用工作集

约束
仍需支持容错

拼装模板

在【场景】中，现有【方法】因【原有假设】产生【不足】；能否通过【改进方向】改善性能，同时满足【约束】？

课堂任务
4 分钟拼装

提交物

一段完整且可检验的问题陈述

参考答案：从 MapReduce 限制到 RDD 问题定义

场景	迭代机器学习与交互式数据分析
原假设	跨阶段共享数据依赖稳定存储
不足	重复磁盘 I/O 和序列化导致高延迟
机理	工作集被反复读取，但框架缺少可容错的内存复用抽象
问题	能否在保留容错的同时，让数据跨操作在内存中高效复用？

边界与权衡：RDD 牺牲了细粒度异步更新能力，换取粗粒度数据并行下的高效复用与 lineage 容错。

论文： MapReduce, OSDI 2004 ； RDD, NSDI 2012.

把阅读变成研究：本讲总结与课后任务

目的
先写目标卡

检索
建立 Search Log

阅读
完成三步阅读

掌握
实现 / 复现 / 讲解

比较
按目标做表

问题
写候选陈述

课后任务

选择一个方向：提交 5 篇文献的检索记录；精读 1 篇核心论文；完成七段式阅读卡；制作比较表；写出 1 个候选问题。

下一讲衔接　候选问题 → 机理分析 → 可检验假设 → 方法与实验设计

核心参考： Keshav 2007 ； Stanford CS197C 2025 ； MapReduce OSDI 2004 ； RDD NSDI 2012.