

大数据发展历史

从核心理念到生态演化

2025年9月

王天青

夯实基础知识，掌握底层技术

大数据的定义与核心理念

大数据的定义与挑战

定义：大数据是指传统数据处理应用软件不足以处理的巨大或复杂数据集。

挑战：处理100GB或1TB数据是否算大数据，并非简单的容量问题，而是取决于现有技术是否能高效处理。大数据的核心在于，它突破了传统数据处理的瓶颈，使得处理PB级别甚至更高量级的数据成为可能。

大数据技术的三大核心理念



可伸缩的分布式数据处理集群

能够扩展到上千台服务器以上的集群，实现数据处理能力的数量级提升。例如，Google 的 GFS 在 2003 年首次展示了单个集群拥有上千个节点的可能性。



基于廉价PC架构的集群

大数据技术不依赖昂贵的专用硬件（如超算、大型机或专用存储设备），而是构建在开放的 PC 架构之上。这大大降低了技术门槛和成本，促进了大数据技术的普及。



"把数据中心当作一台计算机"

大数据技术的目标是为开发者提供一个抽象层，让他们能够像操作单台计算机一样编写代码，而无需关注底层分布式系统的复杂性。这使得处理海量数据变得"容易"。

大数据技术的起源：Google 的挑战

Google作为搜索引擎，面临海量网页数据处理的三大技术挑战。这些真实需求推动了Google在2003年至2006年间开发出"三驾马车"技术，奠定了大数据技术的基础。



存储挑战

Google需要抓取并存储所有网站的网页数据，这远超一般互联网公司的数据量。如何高效存储和管理PB级别的网页数据成为关键技术难题。



解决方案： GFS (Google File System)



计算挑战

PageRank算法需要通过网页反向链接进行多轮迭代计算，对计算能力提出高要求。如何高效处理大规模分布式计算成为技术瓶颈。



解决方案： MapReduce



在线服务挑战

不断增长的搜索请求量要求系统具备迅速响应的在线服务能力。如何在大规模分布式系统中保证低延迟和高可用性成为重要问题。



解决方案： Bigtable

"三驾马车"：大数据技术的基础

Google公司在2003年至2006年间发表的三项奠基性技术，共同奠定了大数据技术的基础



GFS (2003)

全称：Google File System

解决：数据存储问题

特点：上千节点分布式文件系统，易于存储海量数据，顺序读写吞吐量高，但随机读写性能不佳。



MapReduce (2004)

解决：数据计算问题

特点：简化分布式计算模型，易于开发和实现。

应用：PageRank算法可以通过多轮MapReduce迭代实现，但每次读写硬盘导致多轮迭代效率低。



Bigtable (2006)

解决：高性能随机读写问题

特点：基于GFS，实现大集群下高性能随机读写。

实现：通过集群分片调度和MemTable + SSTable的存储格式，但不支持跨行事务和SQL。

三驾马车技术优缺点对比

技术名称	解决问题	优点	缺点
GFS	数据存储	上千节点分布式文件系统，易于存储海量数据	顺序读写吞吐量高，但随机读写性能不佳
MapReduce	数据计算	简化分布式计算模型，易于开发	每次读写硬盘，多轮迭代效率低
Bigtable	高性能随机读写	基于GFS，实现大集群下高性能随机读写	不支持跨行事务，不支持SQL，系统设计较粗糙

OLAP方向：MapReduce的进化

MapReduce作为计算引擎，其进化主要体现在编程模型、执行引擎和多轮迭代问题上。这些技术从"更容易写程序"、"查询响应更快"和"更快的单轮和多轮迭代"等角度，彻底进化了MapReduce，优化了OLAP类型的数据处理性能。



编程模型优化

- Sawzall (Google)
- Pig (Yahoo)
- Hive (Facebook, 2009)



执行引擎改进

- Dremel (Google, 2010)



关键改进

- 列式存储取代行式存储
- 并行数据库架构
- 减少磁盘I/O
- 提高交互式查询效率



多轮迭代问题

- Spark (Berkeley, 2010)



关键突破

- 内存resident数据存储
- 减少磁盘I/O
- 提高迭代计算效率
- 支持交互式数据处理

OLTP方向：Bigtable的进化

Bigtable最初专注于OLAP（在线分析处理），为了解决OLTP（在线事务处理）需求，Google先后开发了MegaStore和Spanner，分别在2011年和2012年发表。



MegaStore (2011)

核心突破

在Bigtable之上实现了类SQL接口，提供了Schema和简单的跨行事务

解决的问题

弥补了Bigtable为伸缩性而放弃的关系型数据库特性

关键特性

- 结构化数据存储
- 跨行简单事务支持
- SQL-like查询接口



Spanner (2012)

核心突破

实现了"全局一致性"，解决了异地多活和跨数据中心问题

解决的问题

首次实现了"全球数据库"，支持多区域同时读写



关键特性

- 跨数据中心强一致性
- 多区域同时活跃
- 水平伸缩与强一致性共存
- 地理分布式部署




实时数据处理的演进



早期流处理系统

-  **S4 (Yahoo, 2010)**
分布式流计算平台，2011年开源
-  **Storm (2011)**
由Twitter工程师开源，工业界事实标准，支持"至少一次"的数据处理
-  **Kafka (2011)**
最初作为消息队列，逐步进化出Kafka Streams等实时数据处理方案

流批协同方案

-  **Lambda架构**
Nathan Marz基于Storm和MapReduce提出，第一个"流批协同"的大数据处理架构
 -  **Kappa架构**
Kafka作者Jay Kreps在2014年提出，被认为是第一代"流批一体"的大数据处理架构
-  尽管早期流式处理系统解决了数据延迟问题，但Lambda和Kappa架构进一步统一了流处理和批处理，提高了系统效率和一致性

Dataflow模型与流批一体



Dataflow模型

Google 2015年发表的流式数据处理模型，对流式数据处理进行了最佳总结和抽象

- ✓ 提供了统一的编程模型，能够同时处理流数据和批数据
- ✓ 成为真正的"流批一体"大数据处理架构
- ✓ 解决了传统流处理中"至少一次"和"正好一次"的语义问题

事件 ————— 流处理 ————— 批处理 ————— 结果



开源实现

Flink

完全按照Dataflow模型实现的开源流处理框架

- 支持事件时间语义和复杂事件处理
- 提供高吞吐量和低延迟处理能力

Apache Beam

Google的开源框架，也实现了Dataflow模型

- 提供统一的编程模型，支持多种运行时环境
- 能够同时处理批处理和流处理工作负载



影响与意义

Dataflow模型及其开源实现彻底解决了流式数据处理中的关键问题，为大数据处理提供了统一的抽象，使得处理海量数据变得高效、可靠和易于编程。这一模型被广泛应用于各种实时分析和批处理场景，成为大数据技术的重要里程碑。

数据湖是一种集中式、低成本、可横向扩展的大数据存储架构。

它通常构建在分布式文件系统（如 HDFS）或对象存储系统（如 Amazon S3）之上，采用扁平化设计存储原始数据，支持包括：

- 结构化数据（如数据库表）
- 半结构化数据（如 JSON、XML 日志）
- 非结构化数据（如图像、音视频等）

数据以原始格式持久化，不经过强制转换或清洗，以保证最大程度的数据保真性和灵活性。

维度	数据湖 (Data Lake)	传统数据仓库 (Data Warehouse)
数据模式	Schema-on-Read，数据存时无结构约束	Schema-on-Write，入库前需建模
数据类型	多类型：结构化、半结构化、非结构化（如日志、JSON、图像）	严格结构化（二维表）
存储架构	构建于对象存储，支持冷热分层，低成本、高扩展性	基于高性能存储，成本高、扩展性有限
计算架构	存算分离，支持弹性扩展、实时/批处理	存算耦合，批处理为主，实时支持弱
数据更新方式	支持流式写入 + ACID 事务（如 Hudi、Delta）	批量加载为主（ETL 后更新）
典型场景	机器学习、原始数据归档、数据探索、湖仓一体	BI 报表、OLAP 分析、固定查询模式
数据治理	后置治理（元数据自动发现、血缘追踪）	预建模型（星型、雪花模型）



层级	关键组件	主要功能/特性
应用层	BI工具（数据可视化与报表） / 数据科学 Notebook（交互式数据分析） / 机器学习平台	数据分析、建模训练、自助式数据探索
统一服务层	Unity Catalog（统一数据目录管理） / 权限控制（RBAC+ABAC）（细粒度权限管理）	统一数据目录、细粒度权限管理、跨计算引擎协同
计算引擎层	Spark（分布式批处理） / Flink（实时流处理） / Presto/Trino（交互式SQL查询） / TensorFlow/PyTorch（机器学习与深度学习） / Neo4j（图数据库与图计算）	批处理、流处理、交互式查询、机器学习、图计算等多种计算范式融合
表格式层	Delta Lake（支持ACID事务的湖存储格式） / Apache Iceberg（开放表格式，支持Schema演进） / Apache Hudi（支持实时分析的表格式）	ACID 事务、Schema 演进、版本控制、时间旅行、增量处理等表管理能力
元数据层	Hive Metastore（元数据管理） / Unity Catalog（统一数据目录管理） / Apache Atlas（业务元数据与数据治理） / Marquez（数据血缘追踪与元数据管理）	技术和业务元数据管理、数据血缘追踪、质量校验、数据发现
数据格式层	Parquet（列式文件存储） / ORC（列式文件存储，混合行组优化） / Avro（行式文件存储，支持复杂嵌套结构与模式演化）	高效压缩、列存优化、支持复杂嵌套结构，适配大数据分析
存储系统层	S3/OSS/COS（云对象存储，高扩展性与持久化） / HDFS（分布式文件系统） / Alluxio（内存缓存加速，数据本地性优化）	冷热分层、弹性扩展、高可用持久化、数据本地缓存（Alluxio）加速访问

资料推荐

Linux 系统编程 → 内核原理理解 → 性能分析实践 → 硬件机制深入

一、《UNIX 环境高级编程（第三版）》

本书被誉为 UNIX 编程的“圣经”，系统讲解了 UNIX/Linux 编程的核心机制，包括文件与目录操作、标准 I/O、进程与线程控制、信号、守护进程、进程间通信（IPC）、网络编程等内容。第 3 版更新了新标准，覆盖了现代 UNIX 系统的最新技术。

二、《性能之巅（第2版）：系统、企业与云可观测性》

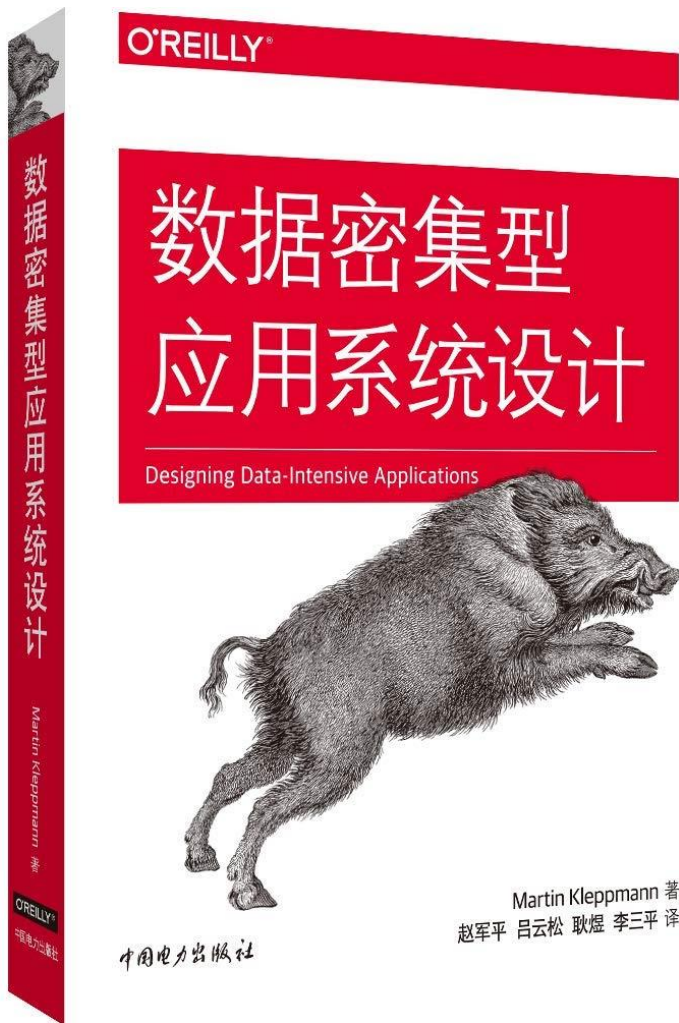
本书围绕系统性能分析与优化展开，涵盖了从单机操作系统、硬件性能到云计算环境中分布式系统的完整可观测性体系。第2版相较第1版全面升级，深入引入了 eBPF、BPFtrace、云原生可观测性技术、容器性能分析等新内容。

三、《BPF 之巅：洞悉Linux系统和应用性能》

本书深入讲解了 BPF（Berkeley Packet Filter）技术如何用于观察、分析和优化 Linux 系统性能。作者是性能专家 Brendan Gregg，内容覆盖了 eBPF 的原理、BPFtrace 工具使用、内核追踪、网络监控、系统调用分析等实战技术。

四、《信息存储与管理（第二版）：数字信息的存储、管理和保护》

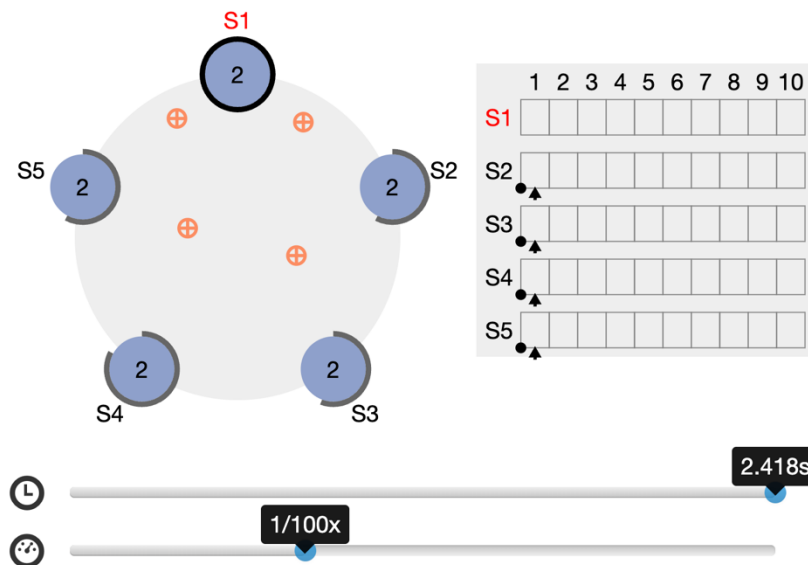
本书由 EMC 教育服务编写，是入门与进阶数据存储技术与架构的经典教材，全面讲解了现代 IT 环境中信息的存储、管理与保护机制



<https://github.com/sofastack/sofa-jraft>

SOFAJRaft 是一个基于 [RAFT](#) 一致性算法的生产级高性能 Java 实现，支持 MULTI-RAFT-GROUP，适用于高负载低延迟的场景。

<https://raft.github.io/>



- **GFS (2003):** “The Google File System”
- **MapReduce (2004):** “MapReduce: Simplified Data Processing on Large Clusters”
- **Bigtable (2006):** “Bigtable: A Distributed Storage System for Structured Data”
- **Chubby (2006):** “The Chubby lock service for loosely - coupled distributed system”
- **Thrift (2007):** “Thrift: Scalable cross - language services implementation”
- **Hive (2008):** “Hive: A warehousing solution over a map - reduce framework”
- **Dremel (2010):** “Dremel: Interactive analysis of web - scale datasets”
- **Spark (2010):** “Spark: Cluster computing with working sets”
- **S4 (2010):** “S4: Distributed stream computing platform”
- **Megastore (2011):** “Megastore: Providing scalable, highly available storage for interactive services”
- **Kafka (2011):** “Kafka: A distributed messaging system for log processing”
- **Spanner (2012):** “Spanner: Google's globally distributed database”
- **Storm (2014):** “Storm@Twitter”
- **Raft (2014):** “In search of an understandable consensus algorithm”
- **Dataflow (2015):** “The dataflow model: A practical approach to balancing correctness, latency, and cost in massive - scale, unbounded, out - of - order data processing”
- **云原生数据库 (2024) :** “云原生数据库综述”
- **HTAP 数据库关键技术综述 (2023) :** “HTAP 数据库关键技术综述”
- **PolarFS (2018):** “PolarFS: an ultra - low latency and failure resilient distributed file system for shared storage cloud database”
- **Delta Lake (2020):** “Delta lake: high - performance ACID table storage over cloud object stores”
- **Lakehouse (2021):** “Lakehouse: A New Generation of Open Platforms for AI and Data Analytics”
- **Iceberg (2024):** “Apache Iceberg: The Definitive Guide”