

海量数据处理中的云计算

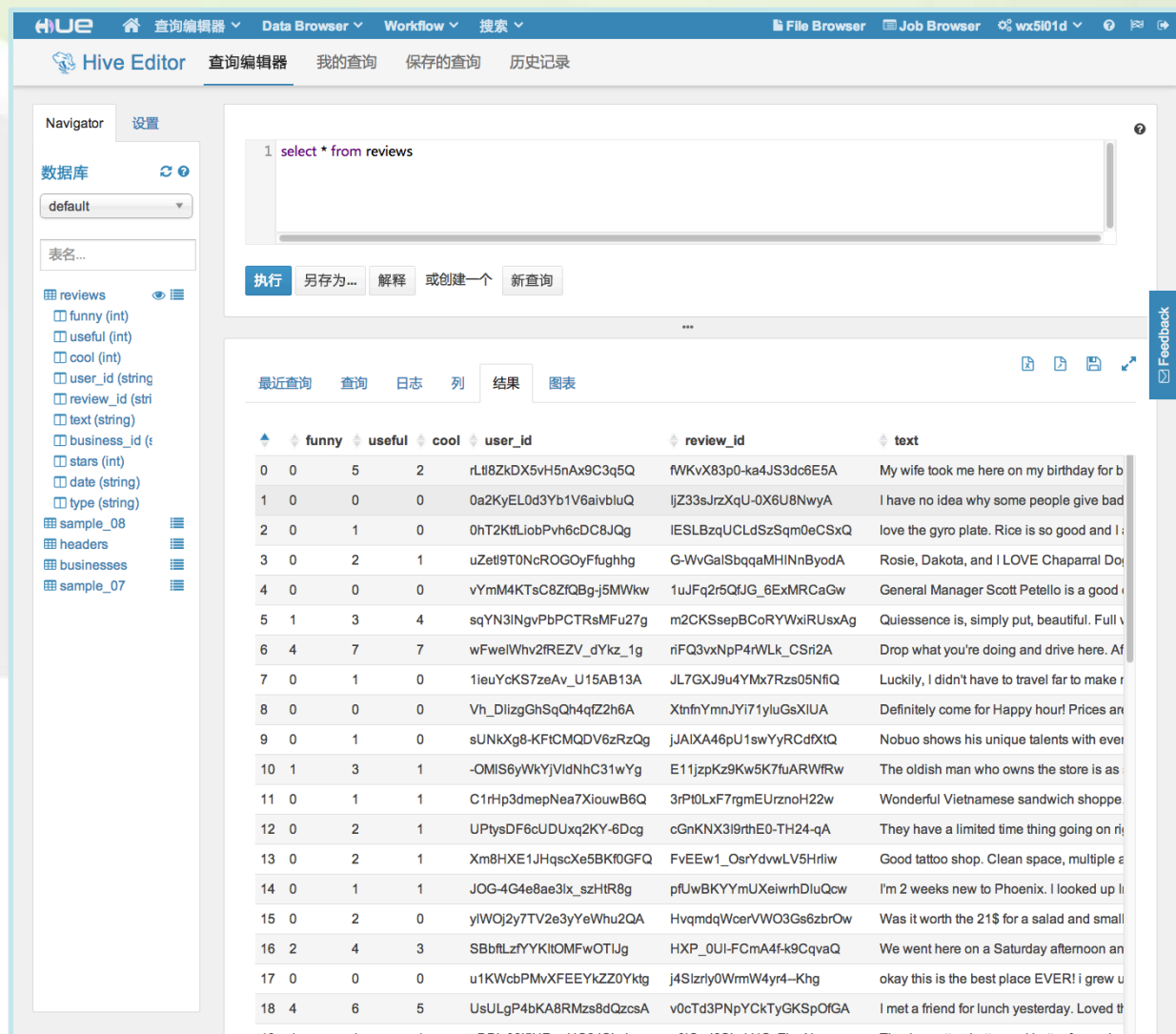
C7. HBase (一)

北京邮电大学信息与通信工程学院

刘军 liujun@bupt.edu.cn

2014年春季学期

- Cloudera Live
- 基于HUE
(Hadoop User Experience) 的
试验环境
- 可以试验Hive/
Pig/Impala/
Solr/Spark/
Oozie/HBase/
HDFS



The screenshot displays the HUE web interface. At the top, there's a navigation bar with tabs for 'Hive Editor', '我的查询' (My Queries), '保存的查询' (Saved Queries), and '历史记录' (History). The 'Hive Editor' tab is active, showing a SQL query: `select * from reviews`. Below the query editor are buttons for '执行' (Execute), '另存为...' (Save As...), '解释' (Explain), '或创建一个' (Or Create A), and '新查询' (New Query). On the left, a 'Navigator' sidebar shows a tree view of the database structure, including tables like 'reviews', 'funny', 'useful', 'cool', 'user_id', 'review_id', 'text', 'business_id', 'stars', 'date', 'type', 'sample_08', 'headers', 'businesses', and 'sample_07'. The main area below the query editor shows the '结果' (Results) tab, displaying a table with 18 rows of data. The table has columns: 'funny', 'useful', 'cool', 'user_id', 'review_id', and 'text'. The data represents reviews from a dataset, with columns 'funny', 'useful', and 'cool' containing numerical ratings, and 'text' containing the review content.

	funny	useful	cool	user_id	review_id	text
0	0	5	2	rLt8ZkDX5vH5nAx9C3q5Q	fWKvX83p0-ka4JS3dc6E5A	My wife took me here on my birthday for b
1	0	0	0	0a2KyEL0d3Yb1V6aivbluQ	ljZ33sJrzXqU-0X6U8NwyA	I have no idea why some people give bad
2	0	1	0	0hT2KfLlObPvh6cDC8JQg	IESLBzqUCLdSzSqm0eCSxQ	love the gyro plate. Rice is so good and I
3	0	2	1	uZetl9T0NcROGOyFfughhg	G-WvGalSbqqaMHlNnByodA	Rosie, Dakota, and I LOVE Chaparral Do
4	0	0	0	vYmM4KtsC8ZfQBg-j5MWkw	1uJFq2r5QfJG_6ExMRCaGw	General Manager Scott Petello is a good
5	1	3	4	sqYN3lNgvPbPCTRsMFu27g	m2CKSsepBCoRYWxiRUsxAg	Quiescence is, simply put, beautiful. Full
6	4	7	7	wFwelWhv2fREZV_dYkz_1g	riFQ3vxNpP4rWLk_CSri2A	Drop what you're doing and drive here. Af
7	0	1	0	1ieuYcKS7zeAv_U15AB13A	JL7GXJ9u4YMx7Rzs05NfiQ	Luckily, I didn't have to travel far to make
8	0	0	0	Vh_DlitzGhSgQh4qfZ2h6A	XtnfnYmnJYi71yluGsXIUA	Definitely come for Happy hour! Prices an
9	0	1	0	sUNKXg8-KFiCMQDV6zRzQg	jJAIXA46pU1swYyRCdftQ	Nobuo shows his unique talents with ever
10	1	3	1	-OMIS6yWkYjVidNhC31wYg	E11jzpKz9Kw5K7fuARWfRw	The oldish man who owns the store is as
11	0	1	1	C1rHp3dmpNea7XiouwB6Q	3rPi0Lx7rgmEUzrnoH22w	Wonderful Vietnamese sandwich shoppe
12	0	2	1	UPtysDF6UDUxq2KY-6Dcg	cGnKNX3l9rthE0-TH24-qA	They have a limited time thing going on r
13	0	2	1	Xm8HXE1JHqscXe5BKf0GFQ	FvEEw1_OsrYdvwLV5Hriw	Good tattoo shop. Clean space, multiple
14	0	1	1	JOG-4G4e8ae3lx_szHIR8g	pfUwBKYYmUXeiwrhDluQcw	I'm 2 weeks new to Phoenix. I looked up
15	0	2	0	yIWQj2y7TV2e3yYeWhu2QA	HvqmdqWcerVWO3Gs6zbrOw	Was it worth the 21\$ for a salad and small
16	2	4	3	SBbftLzFYKlIOMFwOTIjg	HXP_0UI-FCmA4f-k9CqvaQ	We went here on a Saturday afternoon an
17	0	0	0	u1KWcbPMvXFEEYkZZ0Yktg	j4Slzrly0WmW4yr4-Khg	okay this is the best place EVER! i grew u
18	4	6	5	UsULGP4bKA8RMzs8dQzcsA	v0cTd3PNpYCKTyGKSpOfGA	I met a friend for lunch yesterday. Loved

本节目录

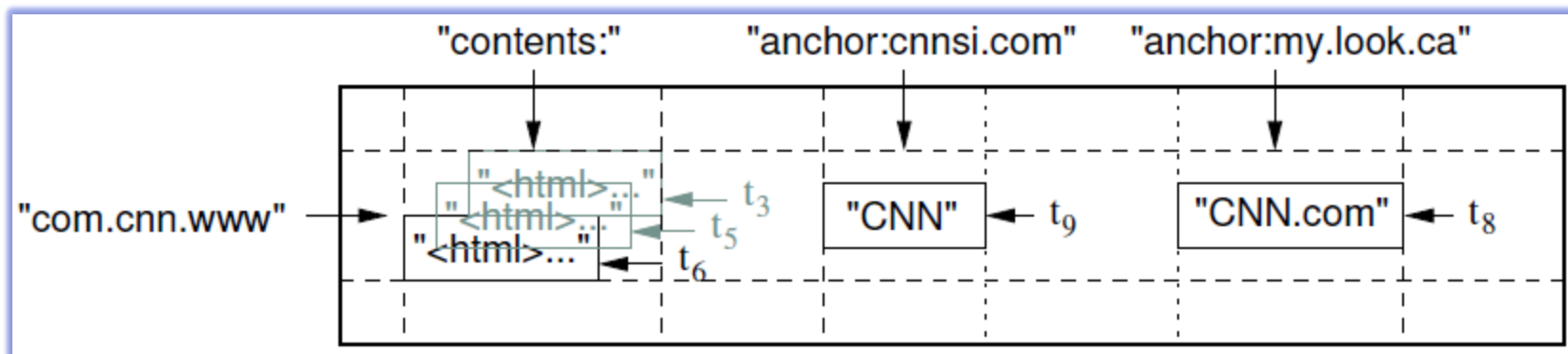
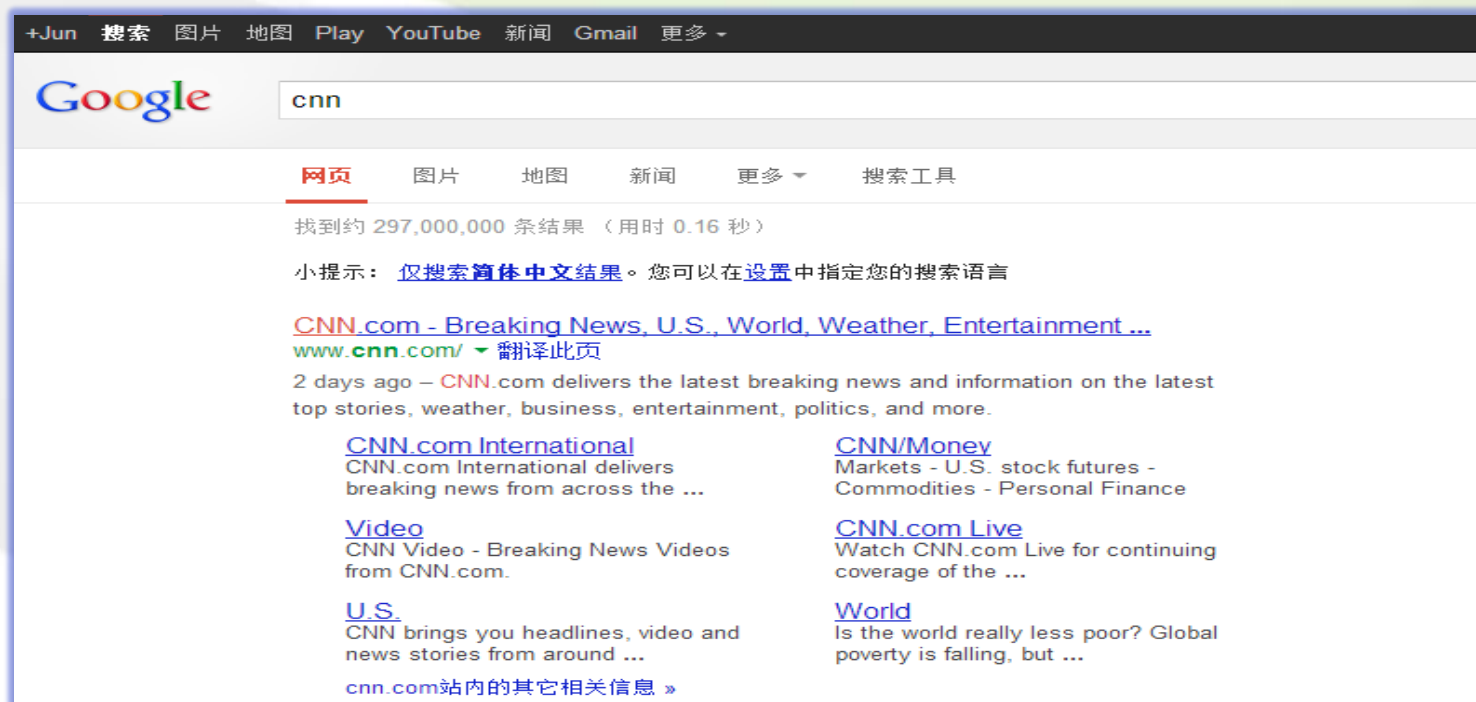
- 为什么需要HBase
- HBase特性及实现原理
- HBase操作的内部流程

本节目录

- 为什么需要HBase
- HBase特性及实现原理
- HBase操作的内部流程

Google应用场景

- 快速检索页面

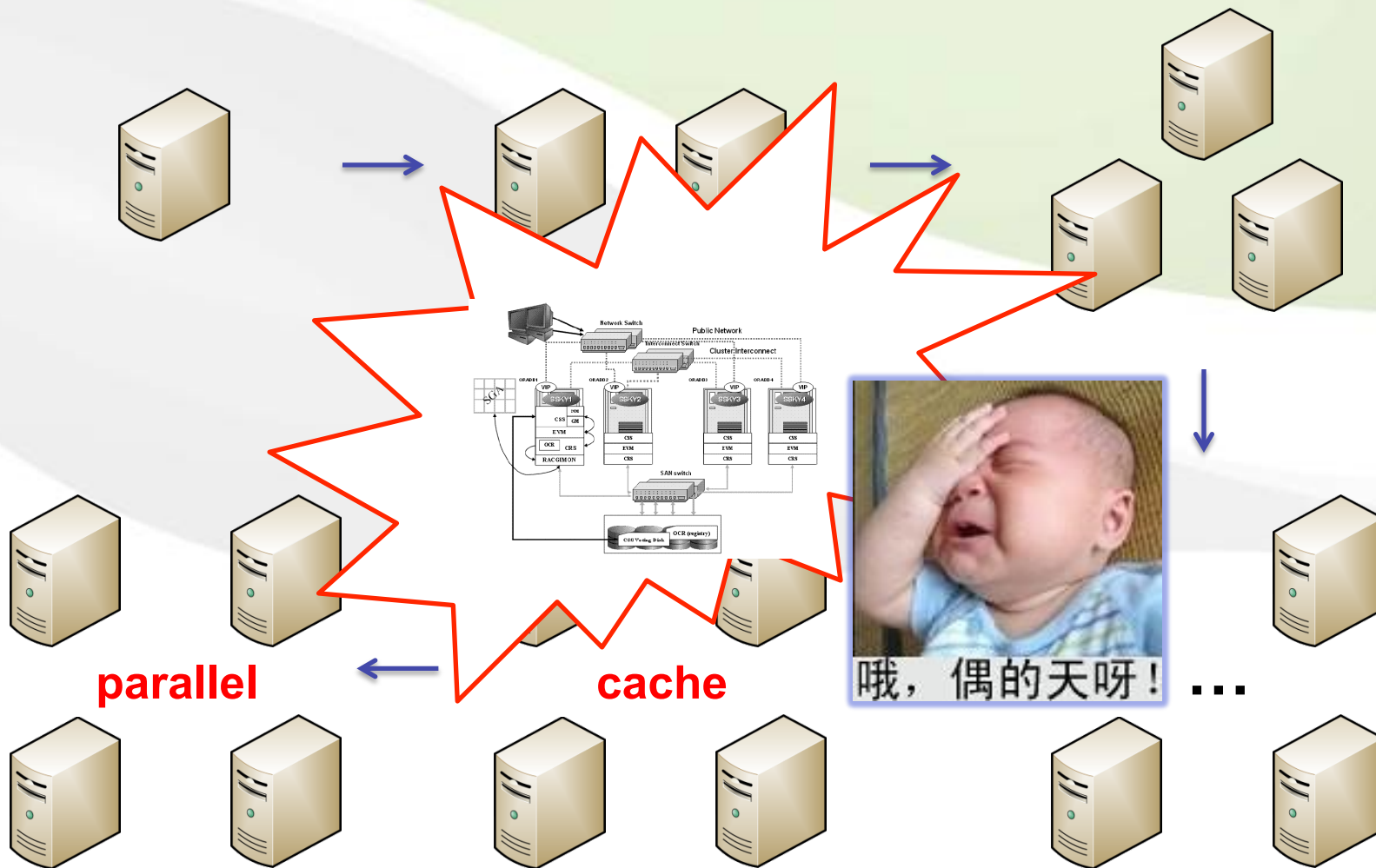


具体需求

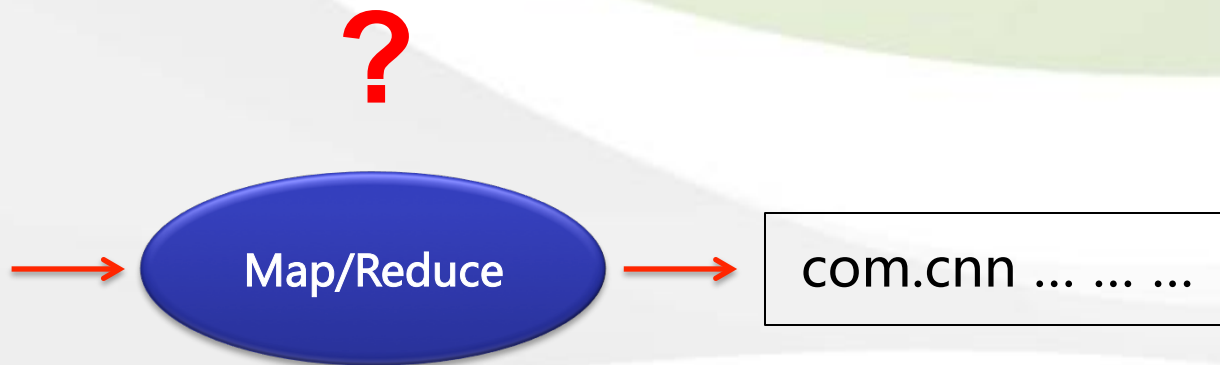
- Google的结构化数据存储需求
 - 低成本可扩展地处理以十亿为单位的数据表（**海量**）
 - 众多的列，但并非每列都有数据，且经常只访问很少的列（**稀疏**）
 - 高吞吐量和高并发（**快速**）
- HBase的原型 - Google Bigtable

Project name	Table size (TB)	Compression ratio	# Cells (billions)	# Column Families	# Locality Groups	% in memory	Latency-sensitive?
<i>Crawl</i>	800	11%	1000	16	8	0%	No
<i>Crawl</i>	50	33%	200	2	2	0%	No
<i>Google Analytics</i>	20	29%	10	1	1	0%	Yes
<i>Google Analytics</i>	200	14%	80	1	1	0%	Yes
<i>Google Base</i>	2	31%	10	29	3	15%	Yes
<i>Google Earth</i>	0.5	64%	8	7	2	33%	Yes
<i>Google Earth</i>	70	–	9	8	3	0%	No
<i>Orkut</i>	9	–	0.9	8	5	1%	Yes
<i>Personalized Search</i>	4	47%	6	93	11	5%	Yes

RDBMS能满足吗？



MapReduce+GFS能满足吗？



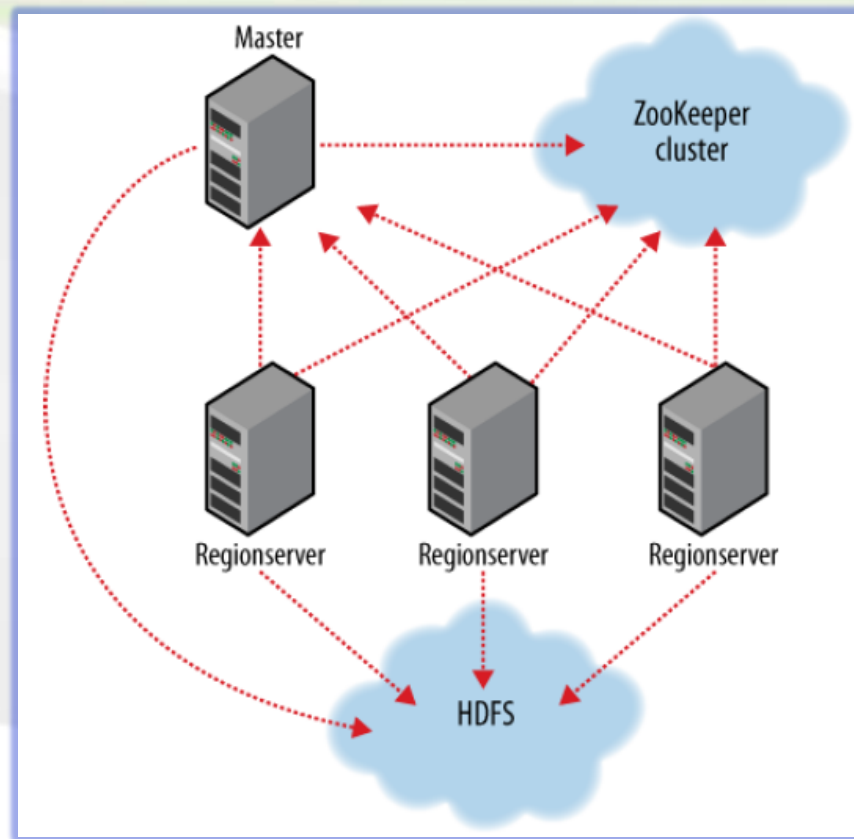
答案

- **因为**RDBMS和MapReduce不能满足要求海量结构化数据存储需求
 - 众多的列，但并非每列都有数据，且经常只访问很少的列（**稀疏**）
 - 低成本可扩展地处理以十亿为单位的数据表（**海量**）
 - 高吞吐量和高并发（**快速**）
- **所以：**
 - Bigtable
 - HBase

本节目录

- 为什么需要HBase
- HBase特性及实现原理
- HBase操作的内部流程

HBase



面向列的、基于HDFS、高性能 分布式数据库系统 (≈)



稀疏



海量



快速

稀疏

稀疏与HBase面向列的数据模型

Row Oriented Storage

Row 1	1	http://hbase.apache.org	3fG4J	HBase Home	Great tool!	<html><head><title>HBase Home</ti...
Row 2	2	http://larsgeorge.com	1337	Lineland	<NULL>	<html><body>Newest Posts. ...
Row 3	3	http://foobar.com/index.html	Hf34h	<NULL>	Read about it...	404 Page not found.



SQL Schema

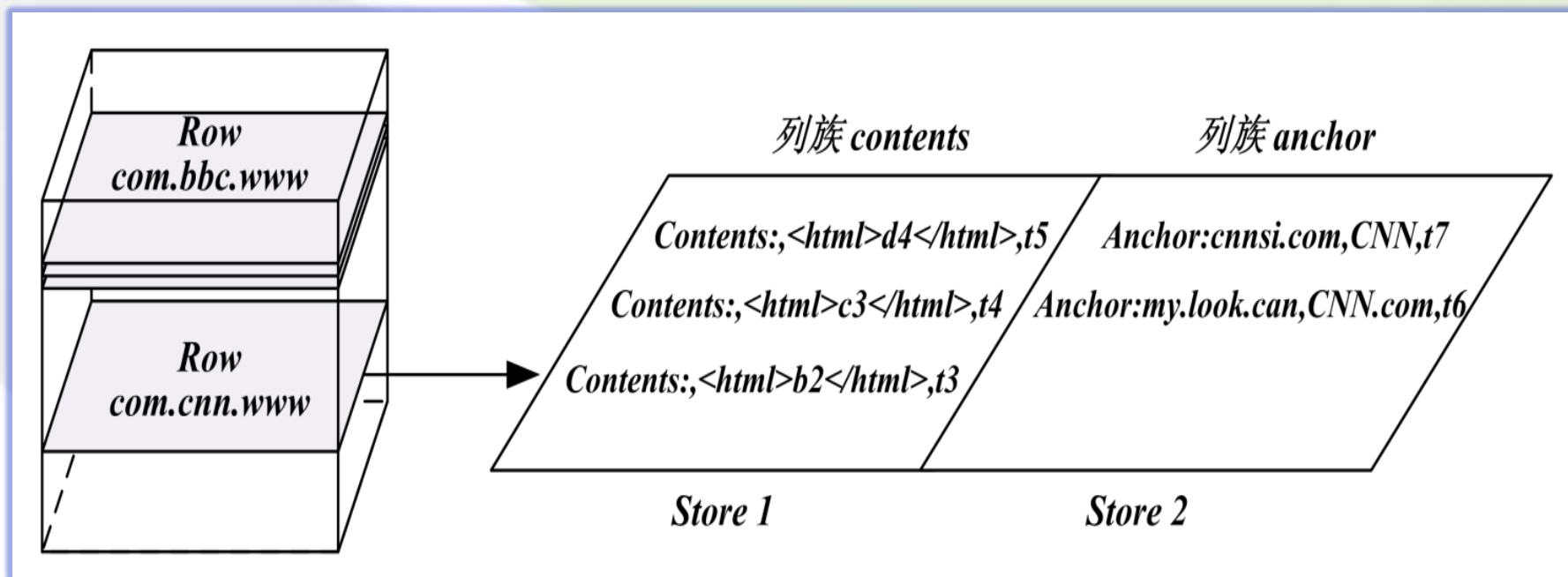
URLS					
url_id	url	ref_short_id	title	description	content
INTEGER PK	VARCHAR(4096)	CHAR(8)	VARCHAR(200)	VARCHAR(400)	TEXT
1	http://hbase.apache.org	3fG4J	HBase Home	Great tool!	<html><head><title>HBase Home</ti...
2	http://larsgeorge.com	1337	Lineland	<NULL>	<html><body>Newest Posts...
3	http://foobar.com/index.html	Hf34h	<NULL>	Read about it...	404 Page not found.
4	http://cnn.com/page123.html	Oo001	Sport News	Soccer News	<html><body>Results, Reviews, ...



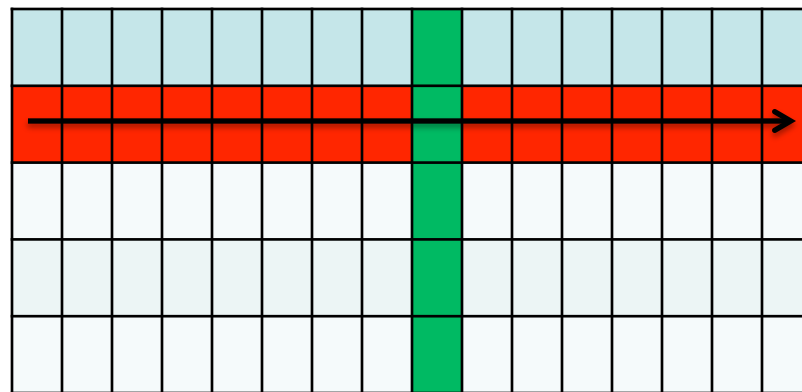
Column Oriented Storage

Col 1: url	http://hbase.apache.org	http://larsgeorge.com	http://foobar.com/index.html	http://cnn.com/page12...	...
Col 2: ref_short_id	3fG4J	1337	Hf34h	Oo001	...
Col 3: title	HBase Home	Lineland	<NULL>	Sport News	...
Col 4: description	Great tool!	<NULL>	Read about it...	Soccer News	...
Col 5: content	<html><head><title>HBa...	<html><body>Newest Po...	404 Page not found.	<html><body>Results,...	...

稀疏与HBase面向列的数据模型



- 提高访问少数列的效率
- 提高压缩比



稀疏与HBase面向列的数据模型

表

RowKey	contents:	anchor:
com.bbc.www	<html>a1</html> [t ₁]	anchor:com.bbc.www = "BBC" [t ₂]
com.cnn.www	<html>b2</html> [t ₃] <html>c3</html> [t ₄] <html>d4</html> [t ₅]	anchor:cnnsi.com = "CNN" [t ₆] anchor:my.look.ca = "CNN.com" [t ₇]

行关键字

列族

版本 (时间戳)

列关键字

限定词

值

数据元

value = Map(TableName, RowKey, ColumnKey, Version)

• TableName

- 表名
- 字符串
- 数据表的标识

• RowKey

- 行关键字
- 字符串
- 最大长度64KB
- 用来检索记录的主键

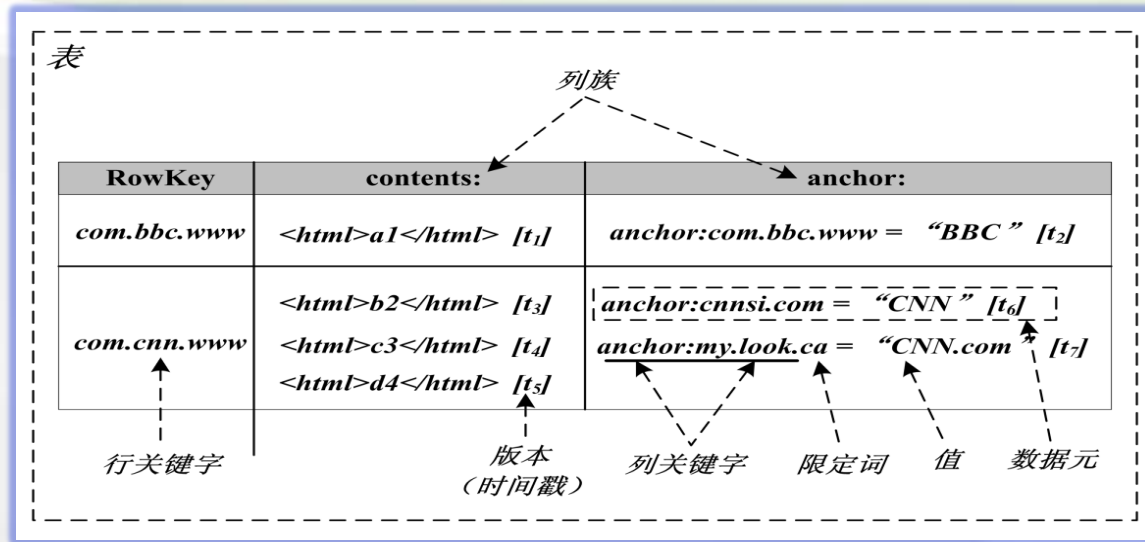
• ColumnKey

- 列关键字
- 列族+限定词
- 字符串
- 数据以列族为准存储
- 列族需提前定义
- 限定词可使用生成

• Version

- 版本
- 适应同一数据在不同时间的变化 (网页)
- 不同版本的同一数据按时间倒序排列, 最新的在最前面

HBase表实例

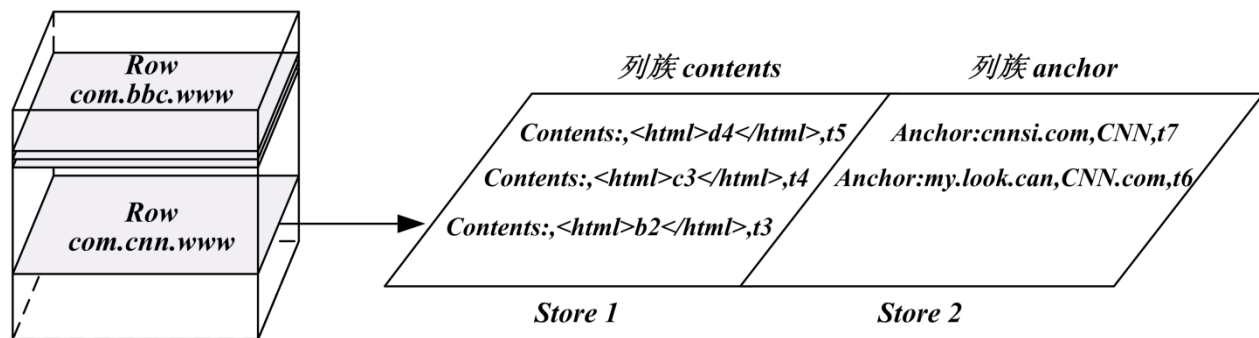


行数	行关键字	版本	列族 : contents	列族 : anchor
1	com.bbc.www	t ₂		anchor:com.bbc.www = "BBC"
	com.bbc.www	t ₁	<html>a1</html>	
2	com.cnn.www	t ₇		anchor:cnnsi.com= "CNN"
	com.cnn.www	t ₆		anchor:my.look.ca= "CNN.com"
	com.cnn.www	t ₅	<html>d4</html>	
	com.cnn.www	t ₄	<html>c3</html>	
	com.cnn.www	t ₃	<html>b2</html>	

海量

逻辑表到HDFS物理存储的映射

- 关键：以列族为单位进行物理存储



- 行 → 列族 = 面 → Store

- 一行数据看作一个面
- 一个列族看作一个Store
- 行由若干列族构成
- 面是若干Store构成
- Store即物理存储基本单元



com.cnn.www的一行数据视为转换为两张物理存储表 (Store) 进行存储

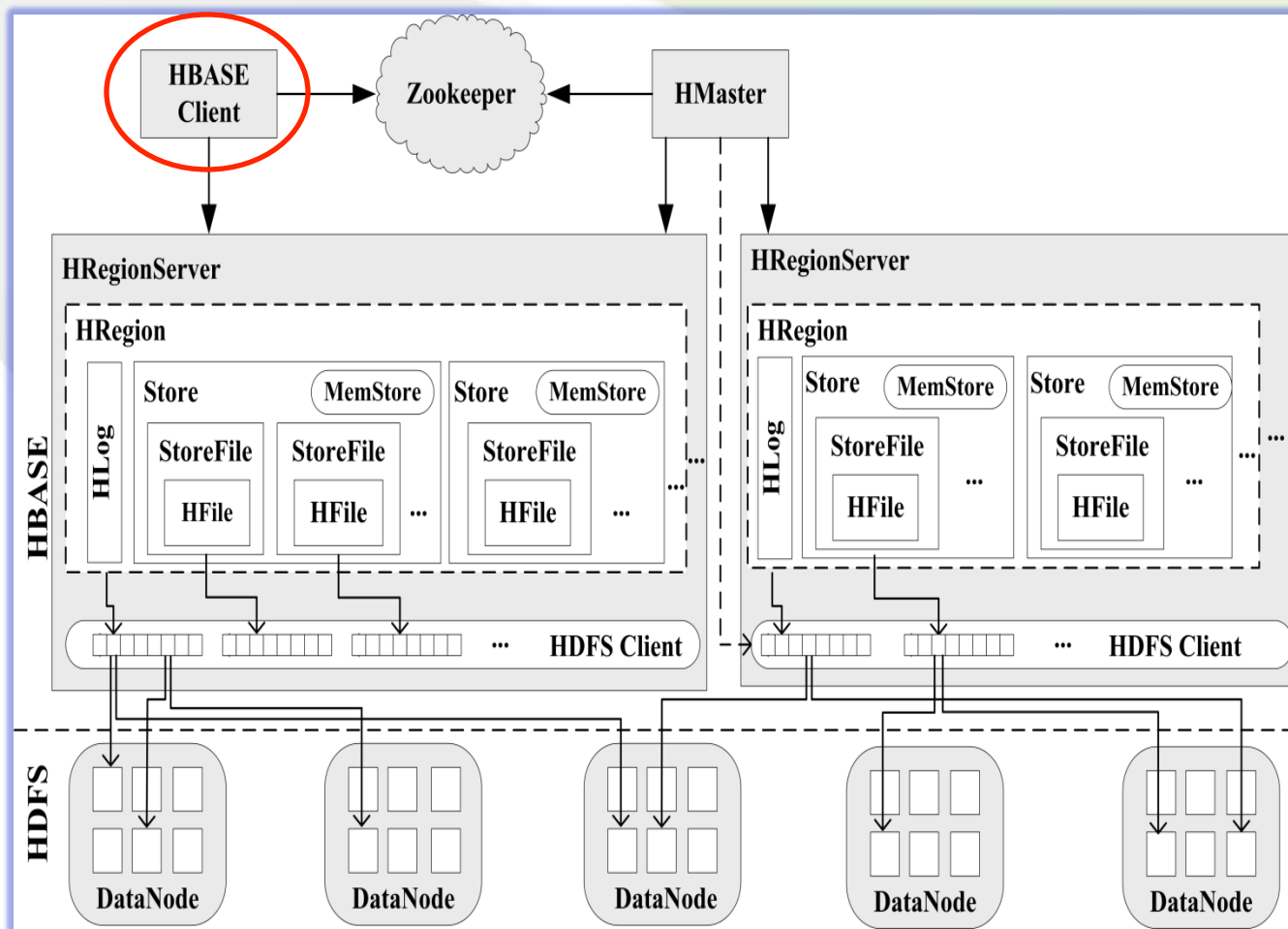
行关键字	版本	列族 : contents
com.cnn.www	t ₅	<html>d4</html>
com.cnn.www	t ₄	<html>c3</html>
com.cnn.www	t ₃	<html>b2</html>

列族contents物理表

行关键字	版本	列族 : anchor
com.cnn.www	t7	anchor:cnnsi.com=" CNN"
com.cnn.www	t6	anchor:my.look.ca=" CNN.com"

列族anchor物理表

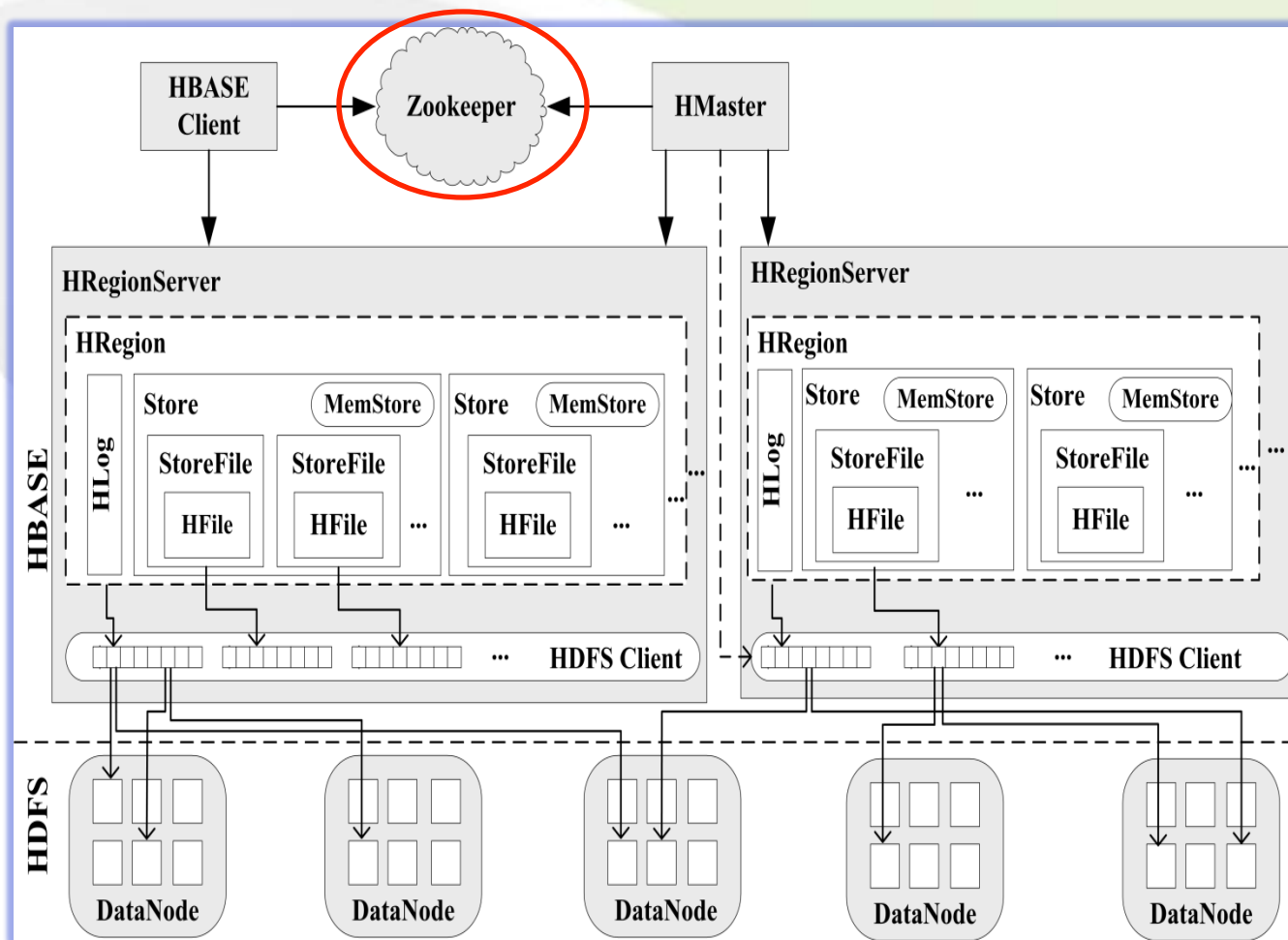
HBase的存储架构 - 使用者



• Client

- HBase功能使用者
- 与Master间进行管理操作
- 与RegionServer间进行数据读写操作

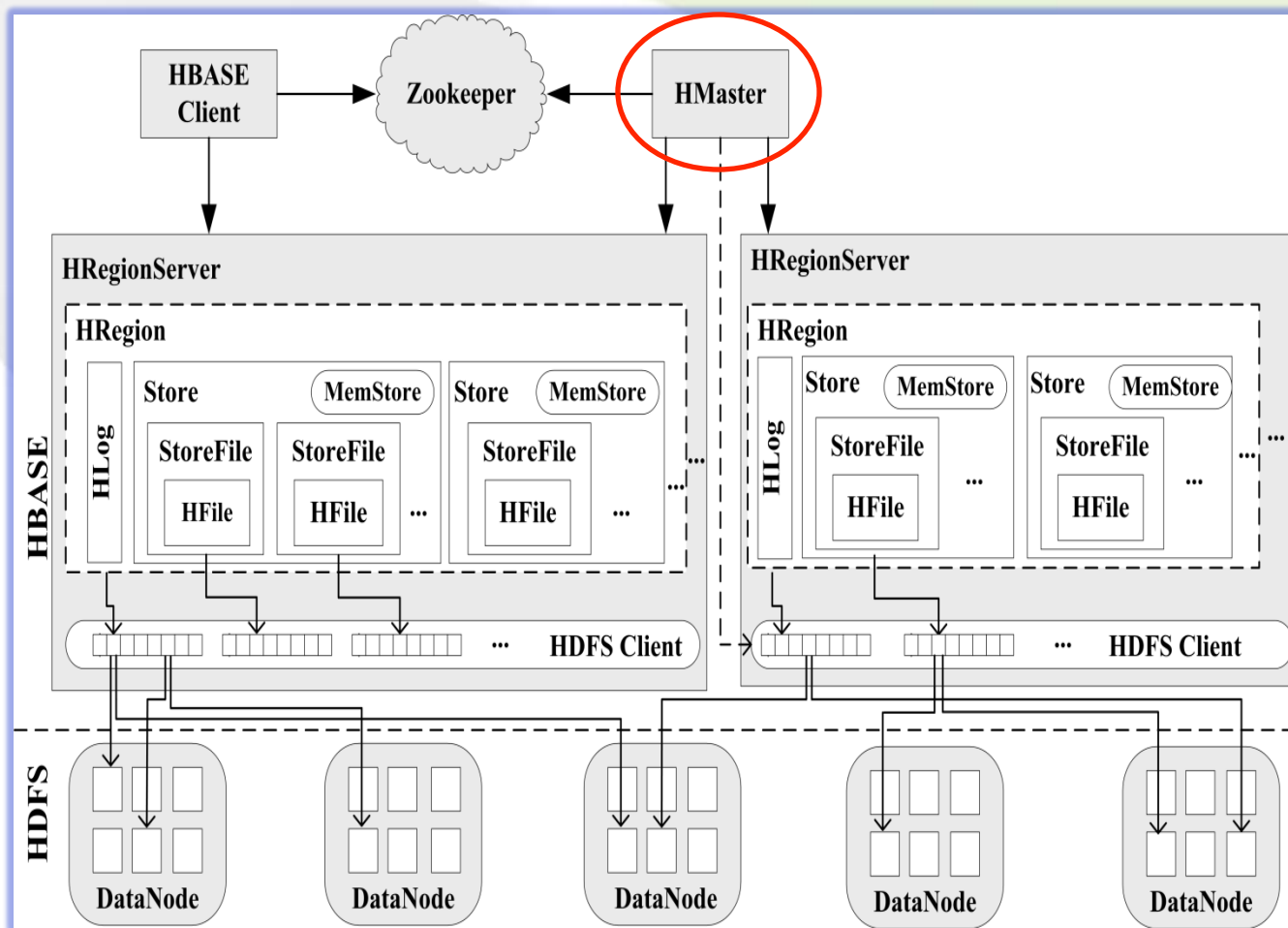
HBase的存储架构 - 协调者



• Zookeeper

- 协同管理节点
- 分布式协作、分布式同步、配置管理
- 存储了Master的地址和RegionServer状态信息

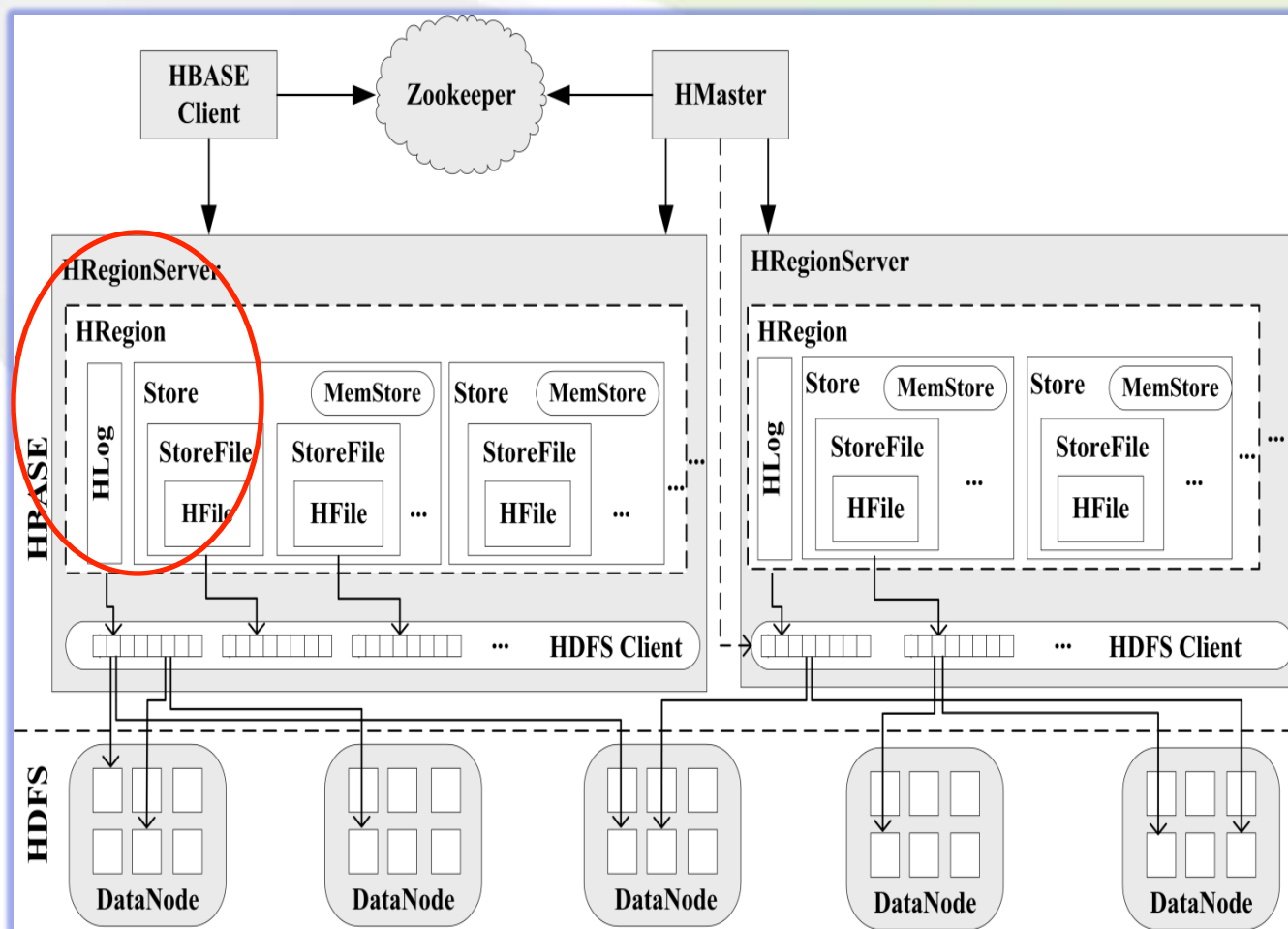
HBase的存储架构 - 管理者



• Master

- 控制节点
- 管理对数据表的增删改和查询操作
- 调整RegionServer的负载均衡和Region分布
- 可有多Master

HBase的存储架构 - 存储者



- **RegionServer**

- 处理数据读写请求
- HDFS文件交互

- **Region**

- 表中的分区
- 多个Store
- 1个HLog

- **Store**

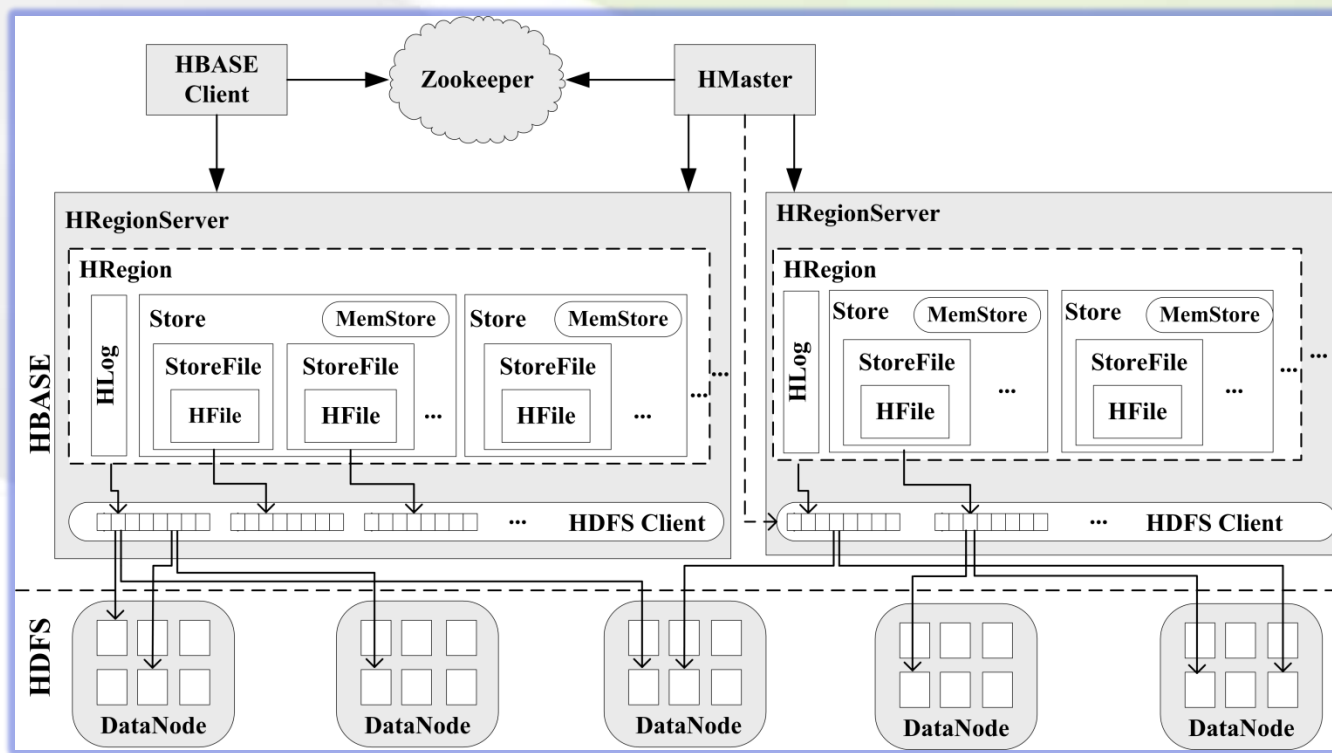
- 数据存储核心
- MemStore/StoreFile

- **HLog**

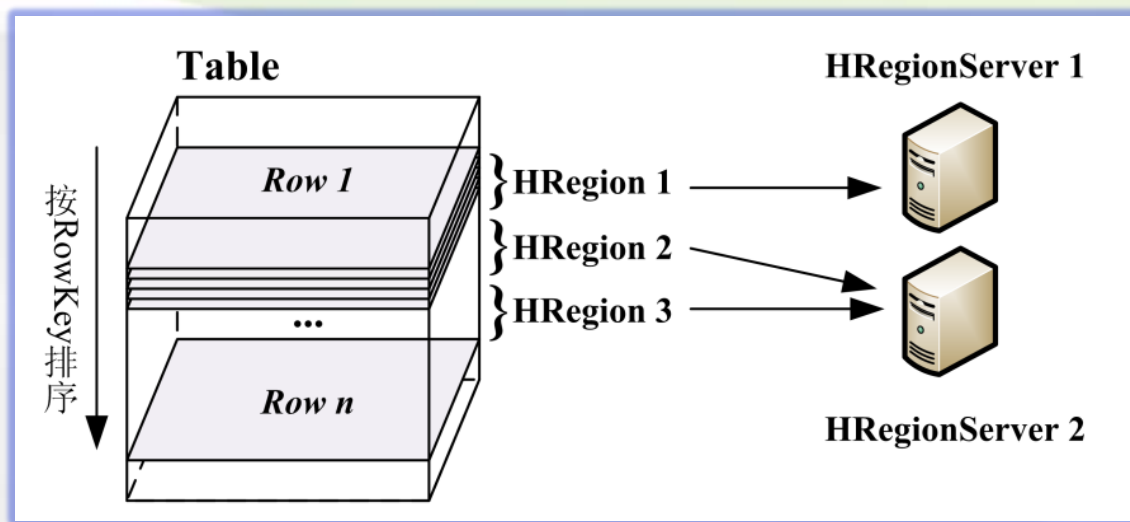
- 保障可靠性
- MemStore数据镜像持久化到文件

逻辑表到物理存储 - 逐步拆解

- Table → Region → Store → HFile → Block → HDFS File



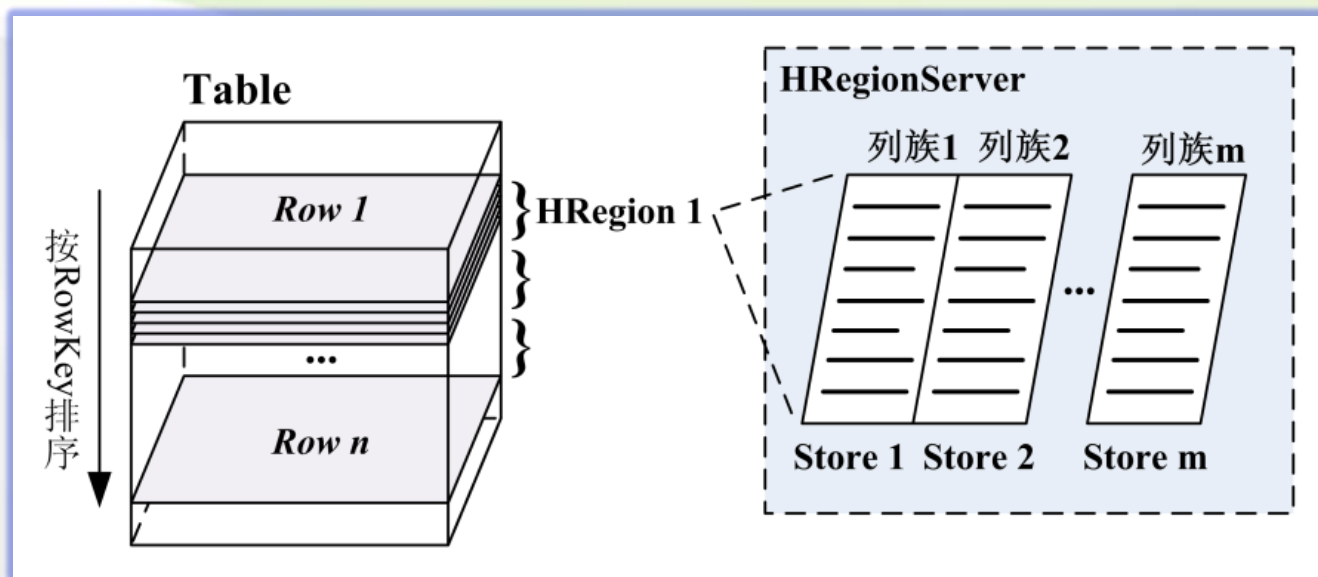
逻辑表到物理存储 - Table → Region



- Table到Region

- 一张表是分为HRegion单元并存储在RegionServer上
- 提高大表存储的效率
- 表数据在行上按RowKey排序后，分为多个Region进程存储
- 多个Region可以存放在一个RegionServer上
- Region的分裂
 - 表在一开始时只有一个Region，随着数据不断增加，Region会越变越大
 - 当超过一个阈值时，Region会等分为两个
 - 这个过程会不断重复，HRegion逐渐增加

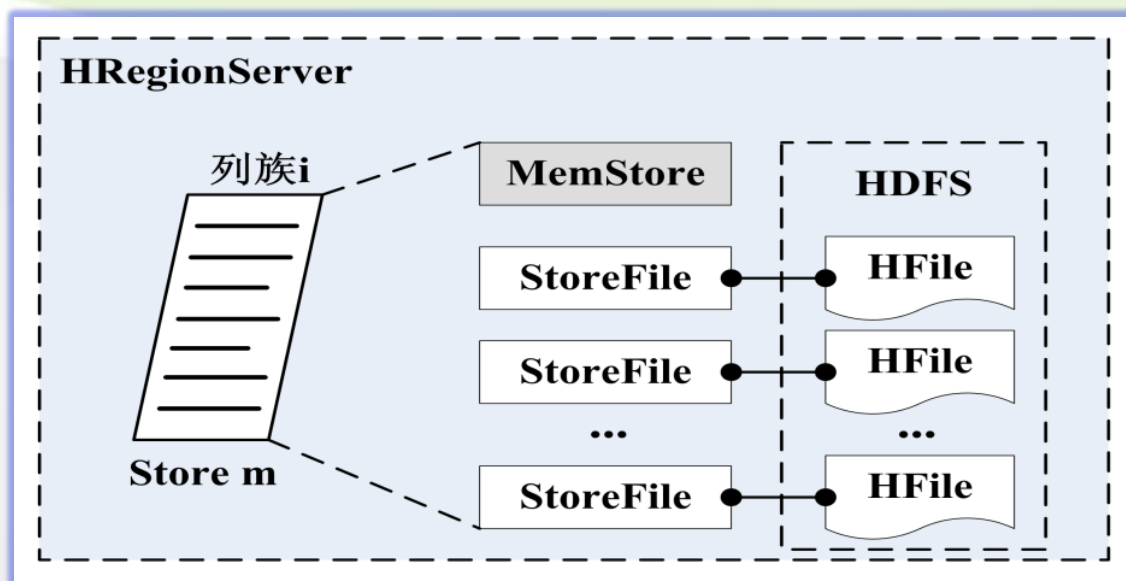
逻辑表到物理存储 - Region → Store



- Region到Store

- HRegion是分布式存储的最小单元，但并不是物理存储的最小单元
- Region划分为若干Store进行存储，每个Store保存一个列族中的数据

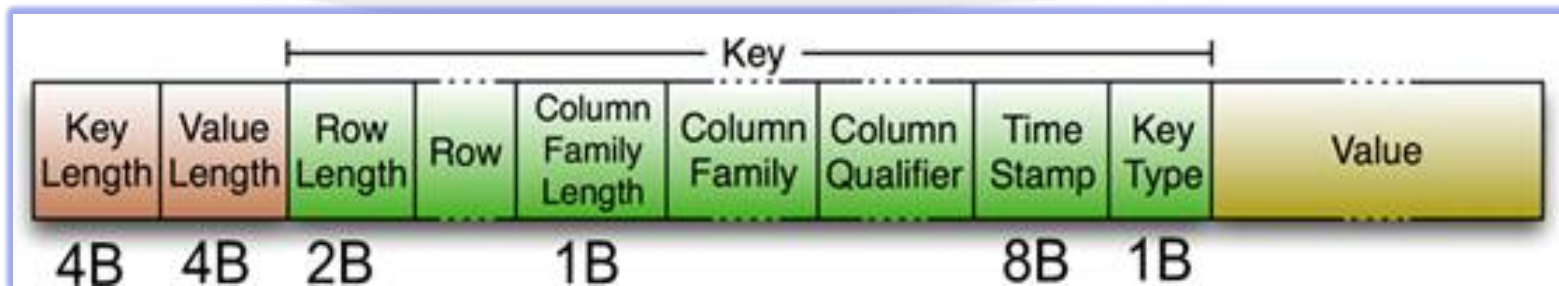
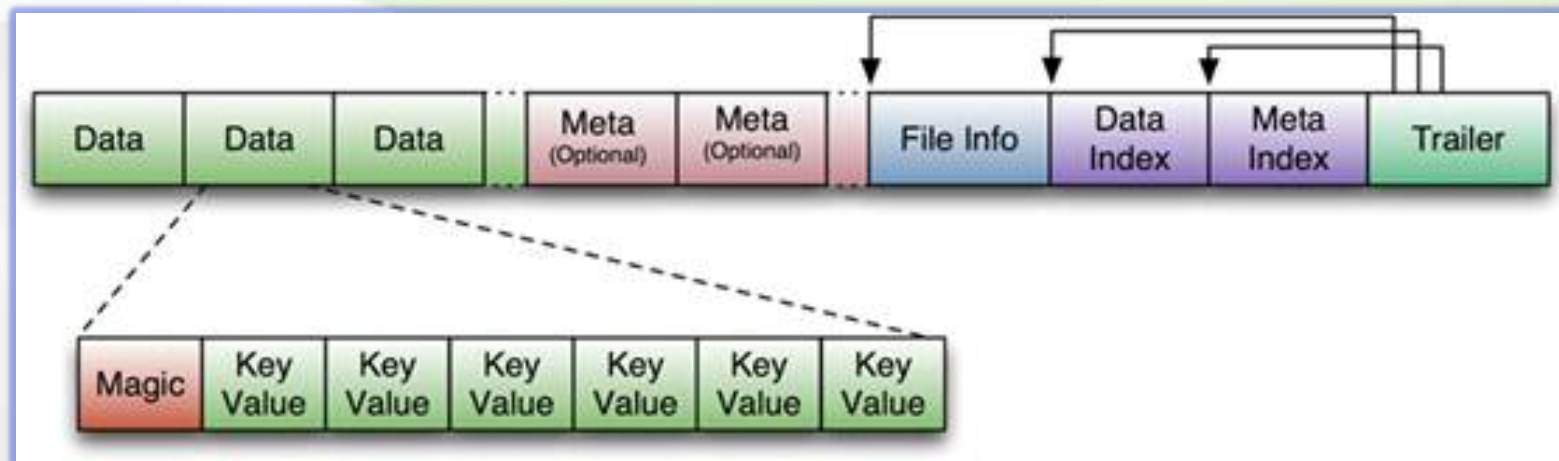
逻辑表到物理存储 - Store → HFile



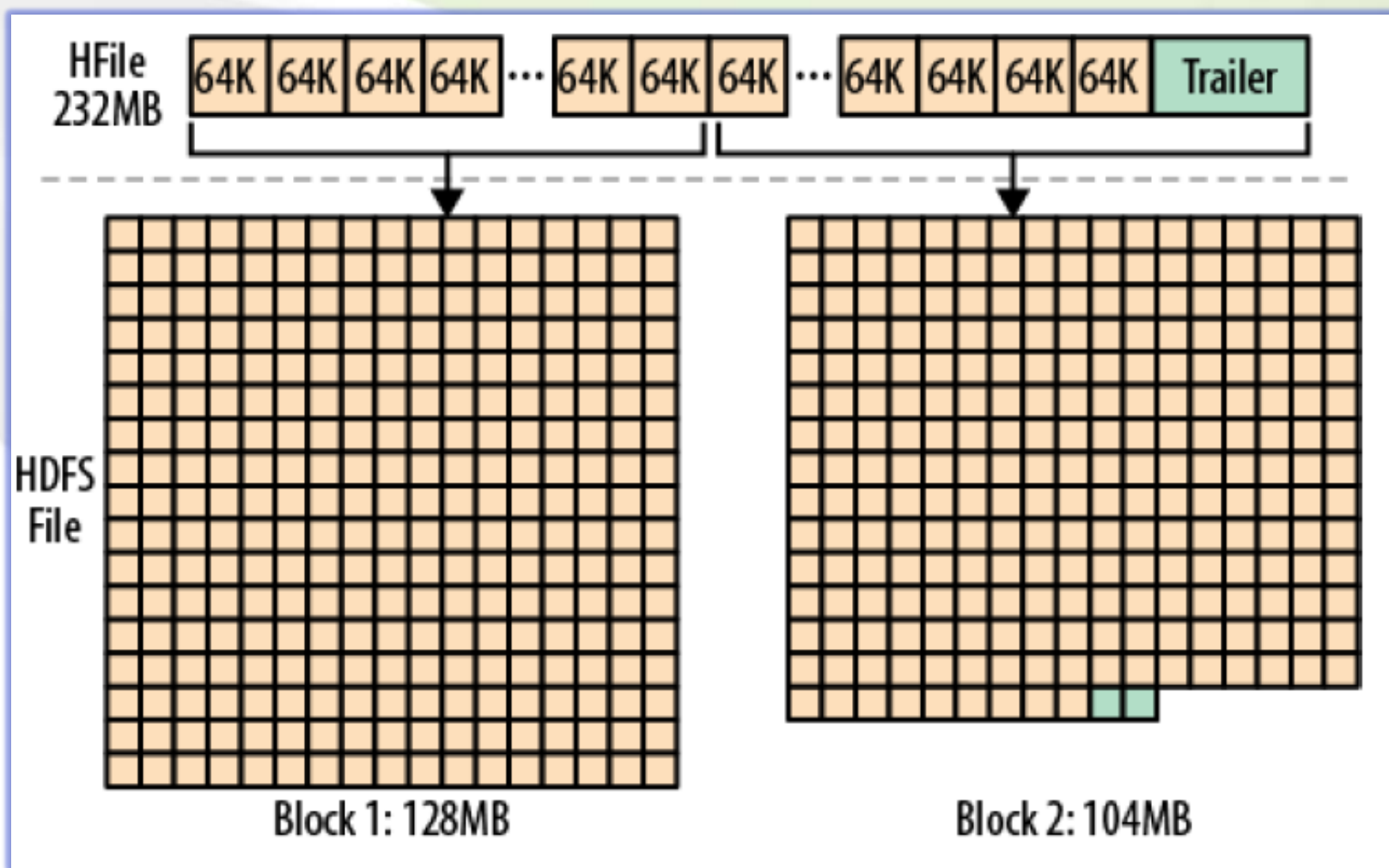
● Store到File

- Store由两部分组成，**MemStore**和StoreFile
 - MemStore是RegionServer上的一段内存空间
 - StoreFile是HDFS中的一个HFile文件
- 数据库操作会先存入MemStore，当MemStore满了后会转存到StoreFile中 (?)
- 1个Store可包含多个StoreFile，并建立了**StoreFile索引**

逻辑表到物理存储 - HFile → Block



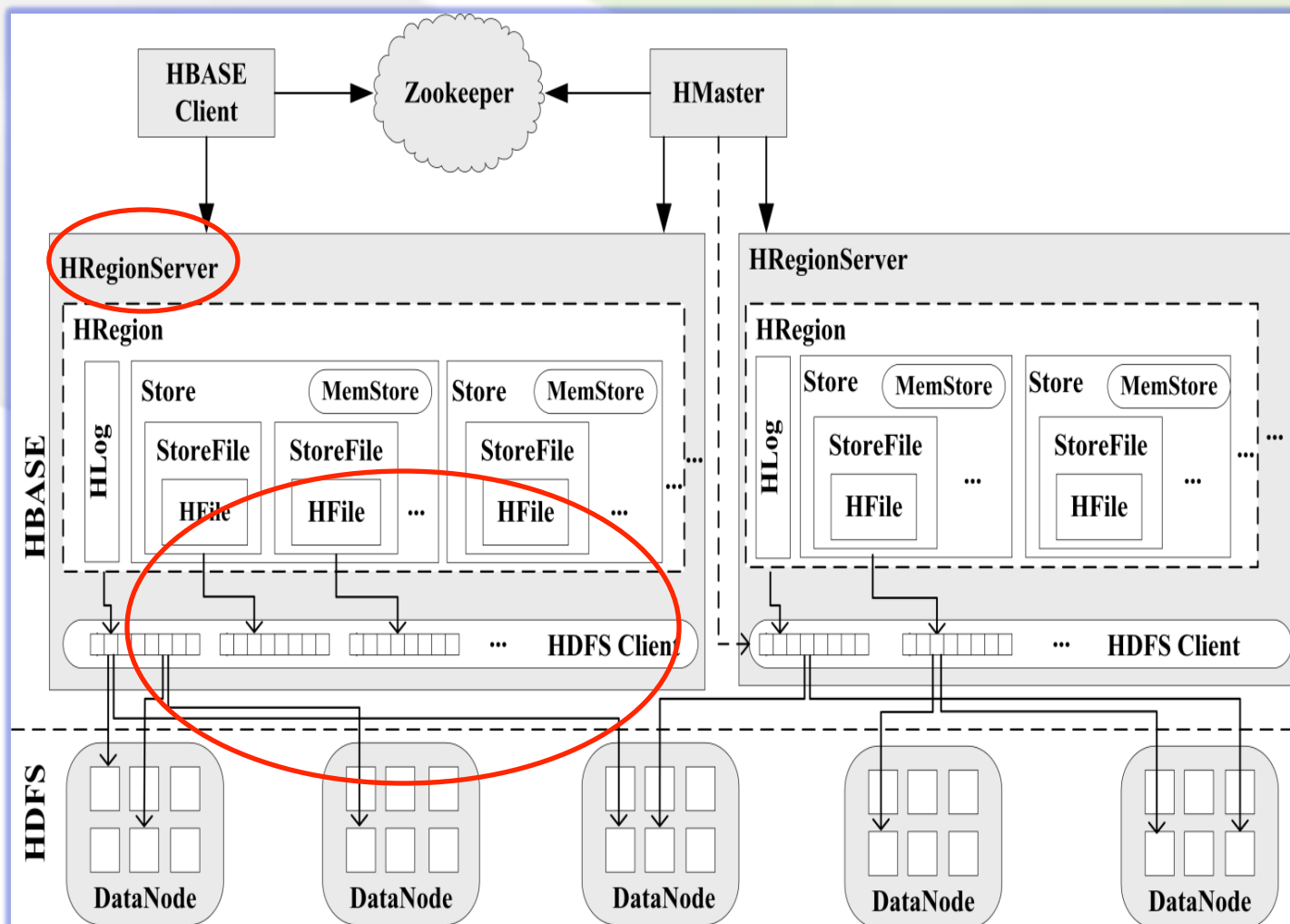
逻辑表到物理存储 - HFile → HDFS Block



速度

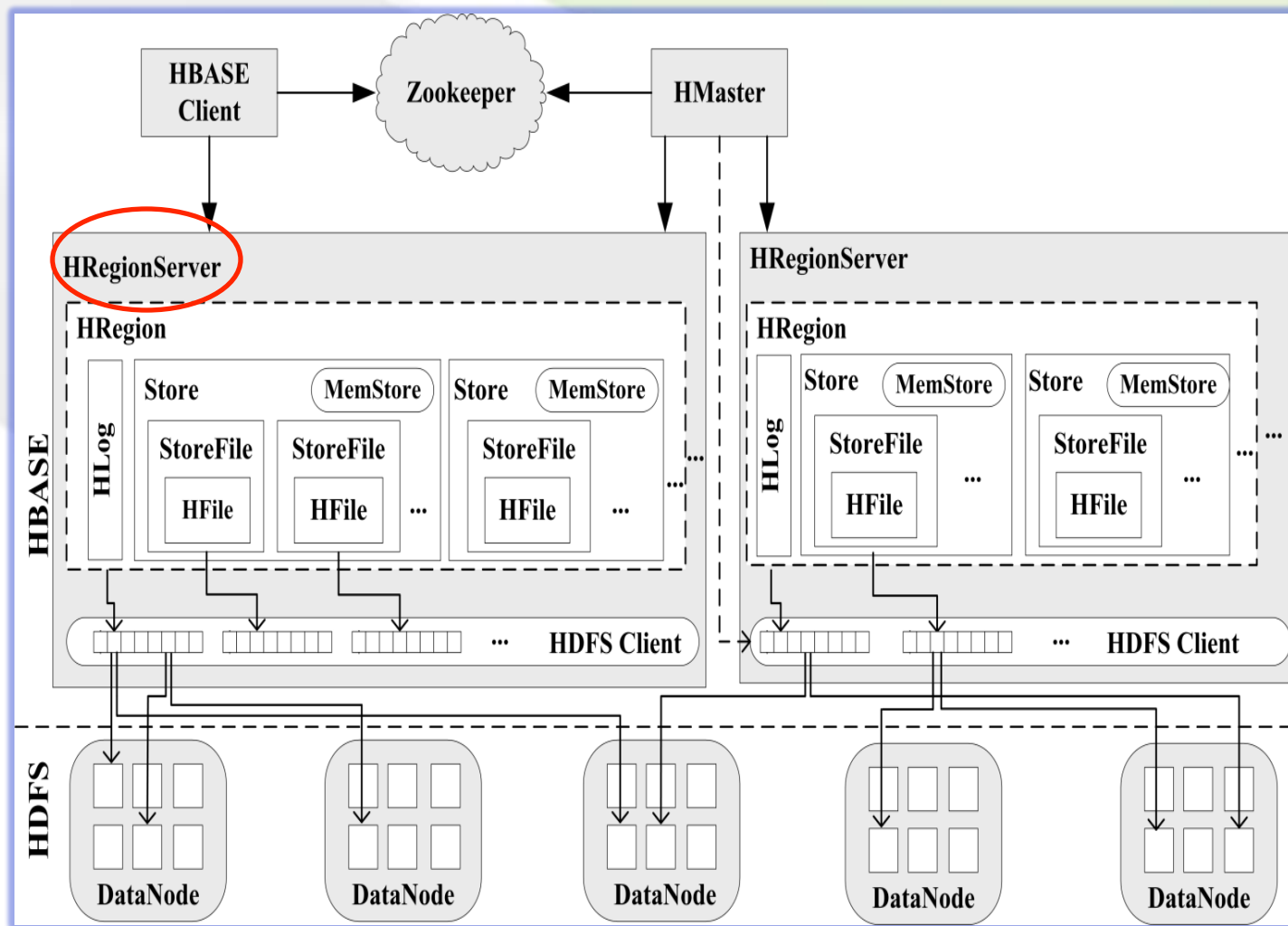
速度的关键

- 第1步：快速找到RegionServer
- 第2步：快速找到HFile



第1步：定位RegionServer

- 如何通过表名和行关键字找到所在的RegionServer？

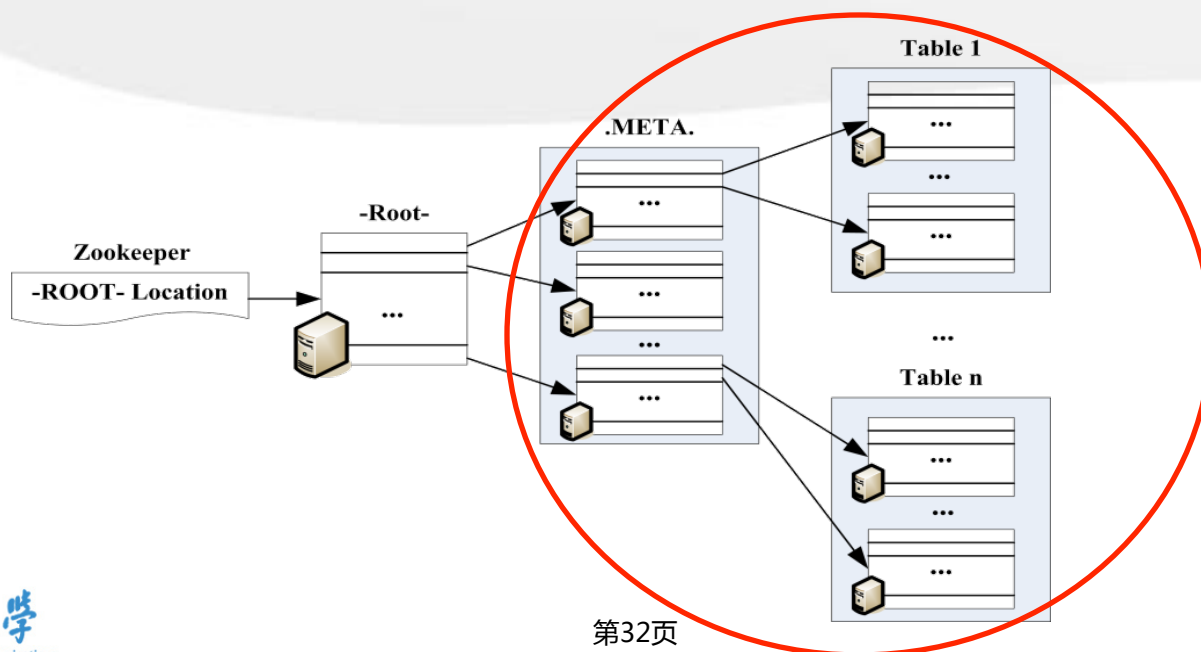


定位RS - 找到Region (.META.表)

- .META.表

行关键字	列1	列2	列3
<table, region start key, region id>	info:regioninfo	info:server	info:serverstartcode

- 存储了所有表的元数据信息
- 支持以表名和行关键字（或关键字的范围）查找到对应的RegionServer
 - 行关键字：表名、此Region起始关键字和Region的id
- info:regioninfo：记录Region的一些必要信息
- info:server：Region所在的RegionServer的地址和端口
- info:serverstartcode：RegionServer对应.META.表持有进程的启动时间

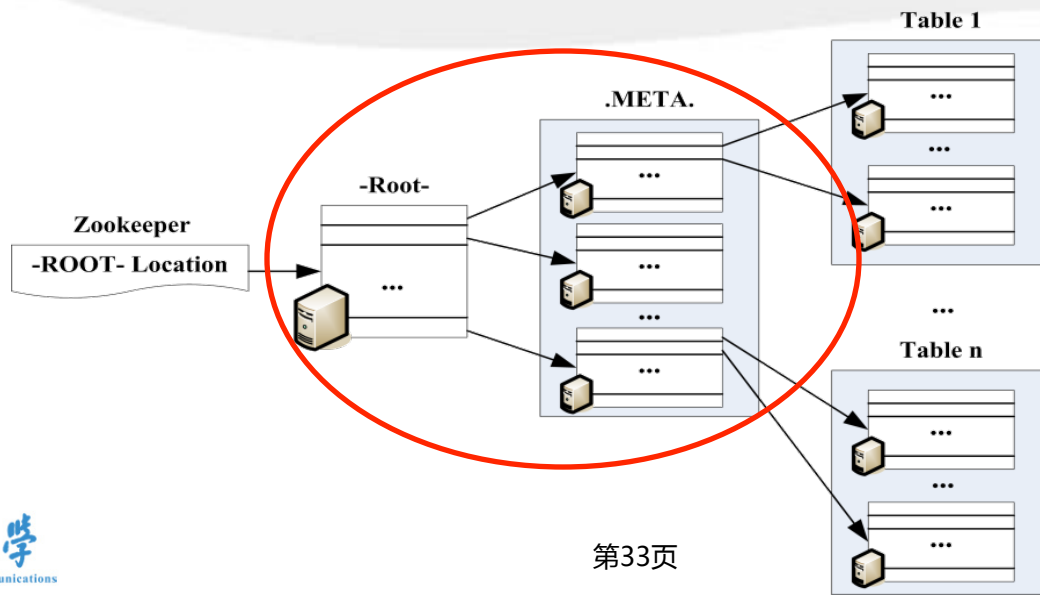


定位RS - 找到.META. (-ROOT-表)

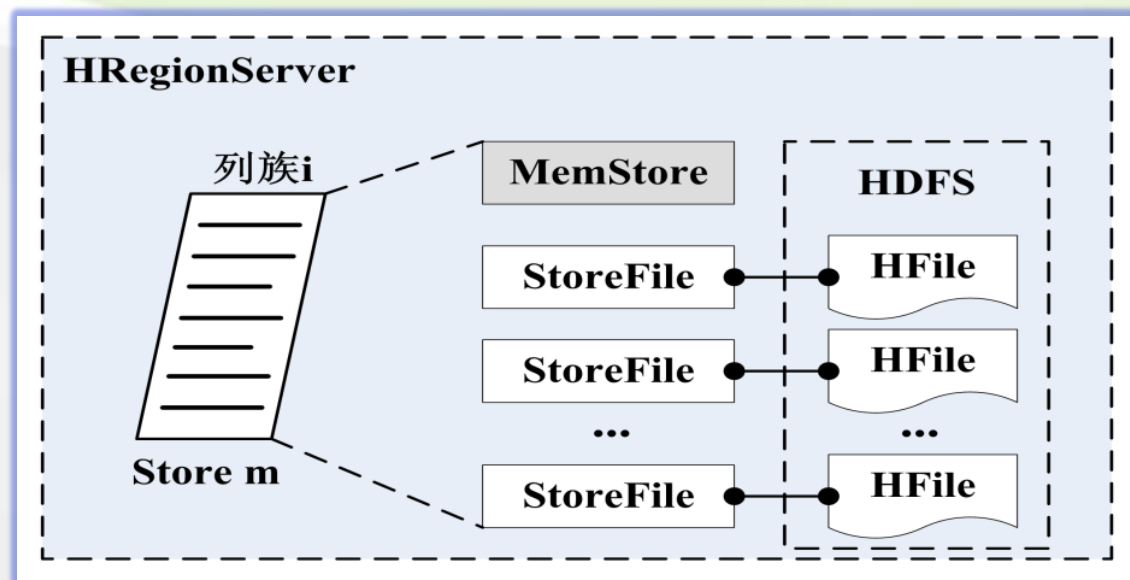
- -ROOT-表

行关键字	列1	列2	列3
.META. Region Key	info:regioninfo	info:server	info:serverstartcode

- 根数据表，存放了.META.表的HRegionServer信息，存放在Zookeeper服务器
- -ROOT-表的Region不会被拆分，永远只有一个
- 客户端首次访问获取-ROOT-表的位置并存入缓存
- 行关键字：每个.META.表的Region索引
- info:regioninfo：记录Region的一些必要信息
- info:server：Region所在的RegionServer的地址和端口
- info.serverstartcode：RegionServer对应.META.表持有进程的启动时间

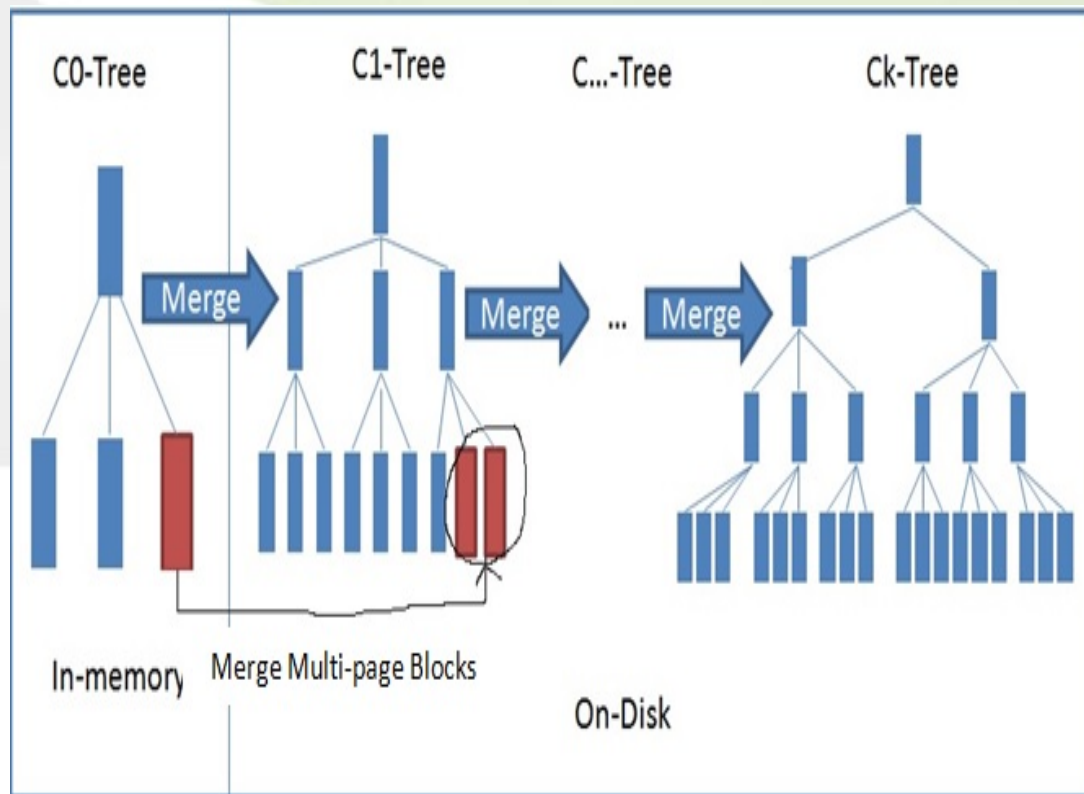
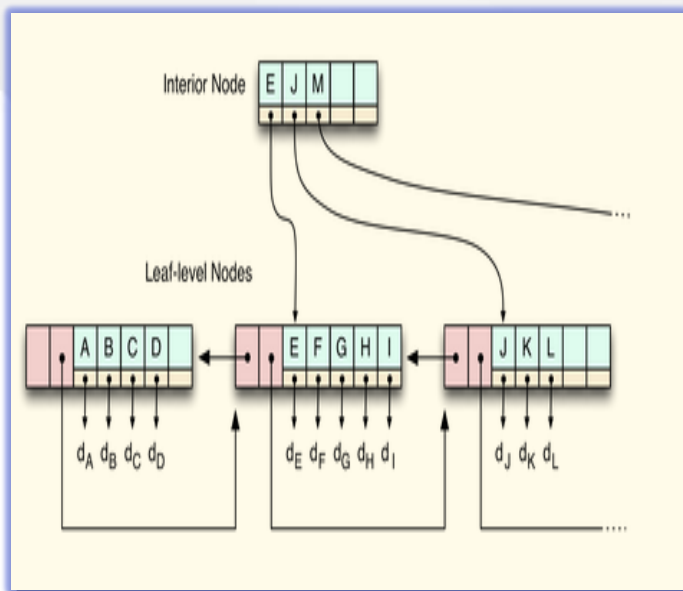


第2步 - 快速找到HFile



定位HFile - Memstore与Store对StoreFile的索引

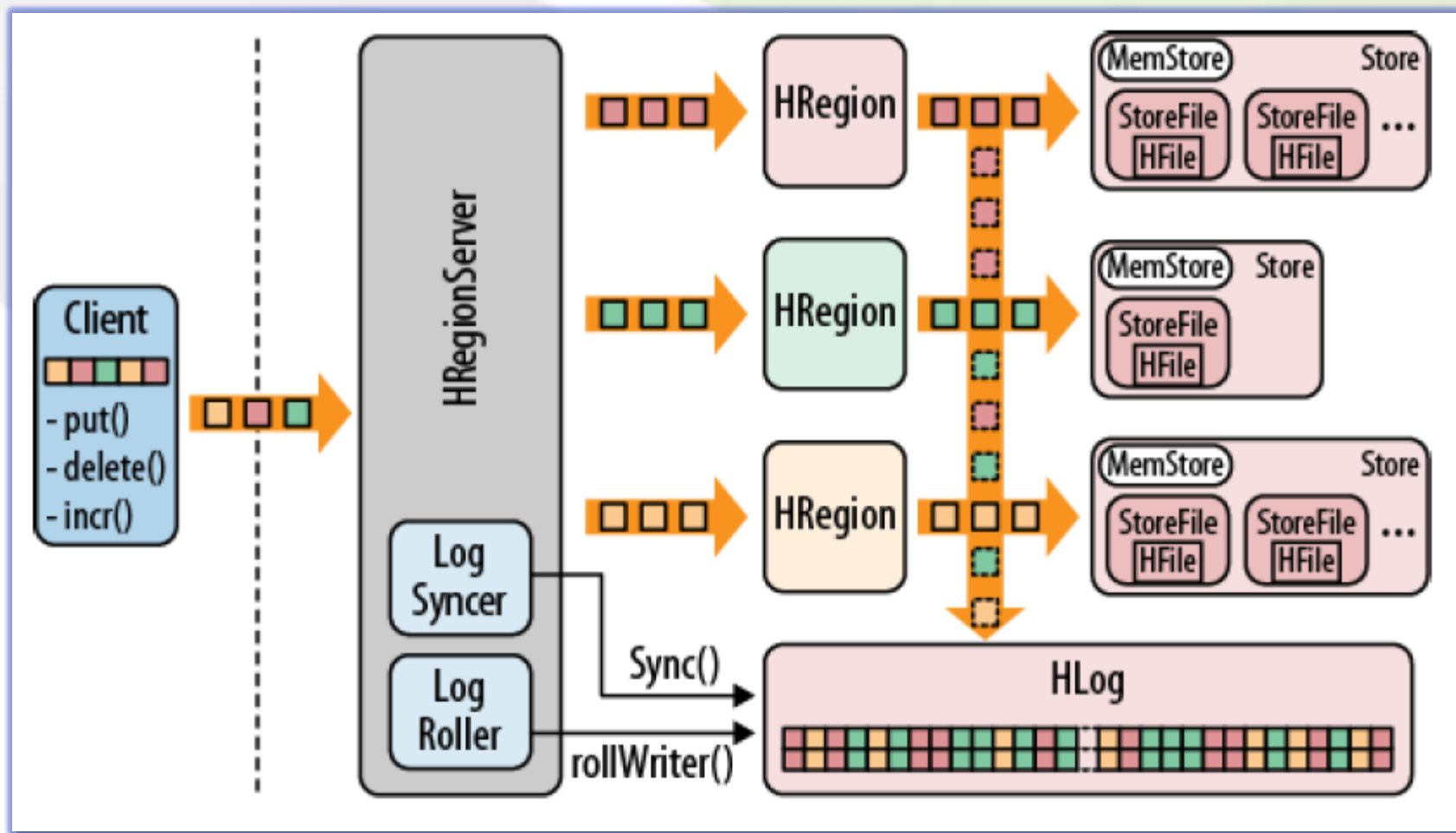
- B+ tree (RDMBS时代的索引表) → LSM tree
 - 查询优化 VS. 插入优化
 - 内存 VS. 磁盘



<http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.2782>

定位HFile - Memstore带来的问题

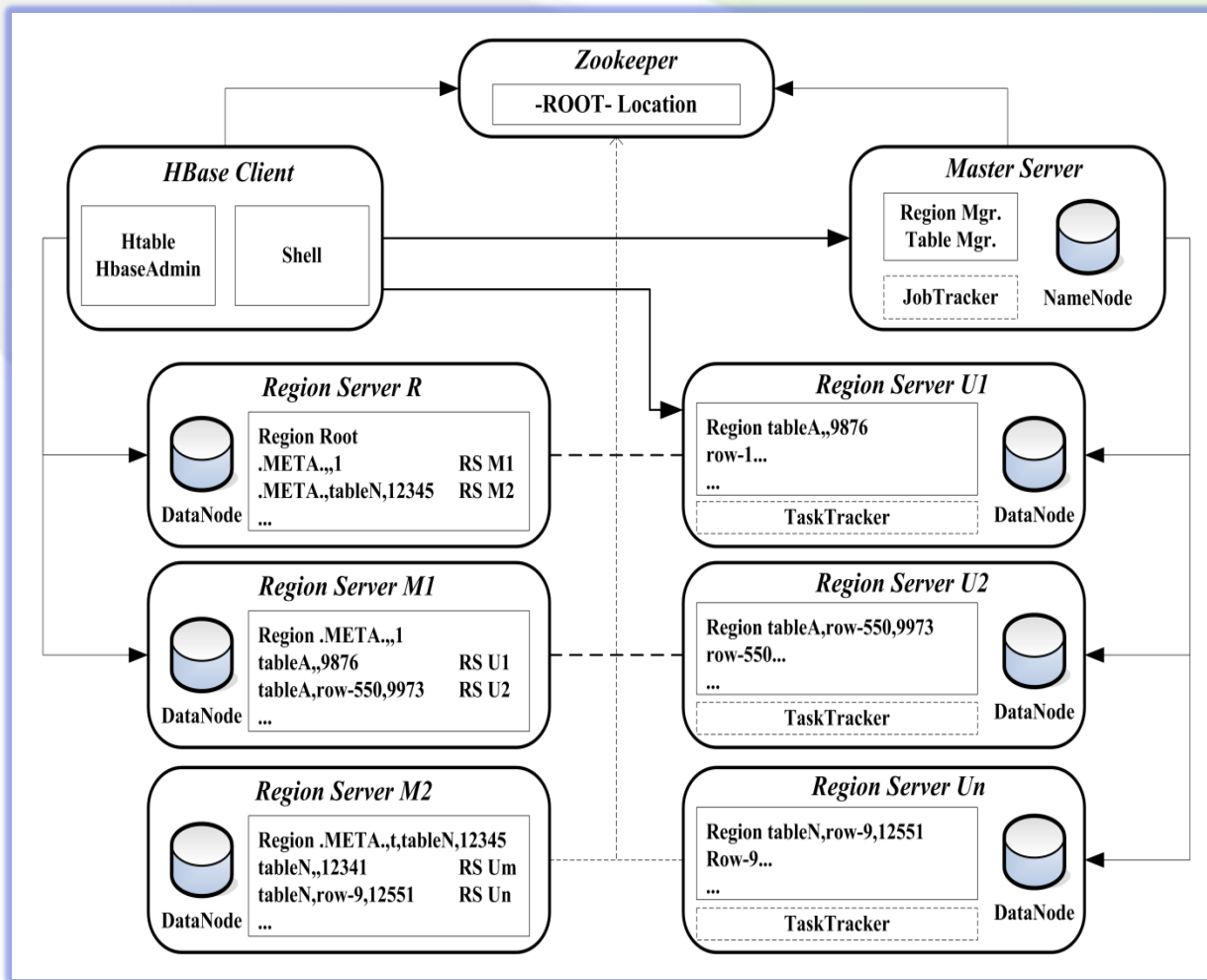
- memStore带来的问题：RegionServer宕机怎么办？
 - Write-Ahead Logging (WAL) + HLog



本节目录

- 为什么需要HBase
- HBase特性及实现原理
- Hbase的部署与操作流程

HBase典型物理部署



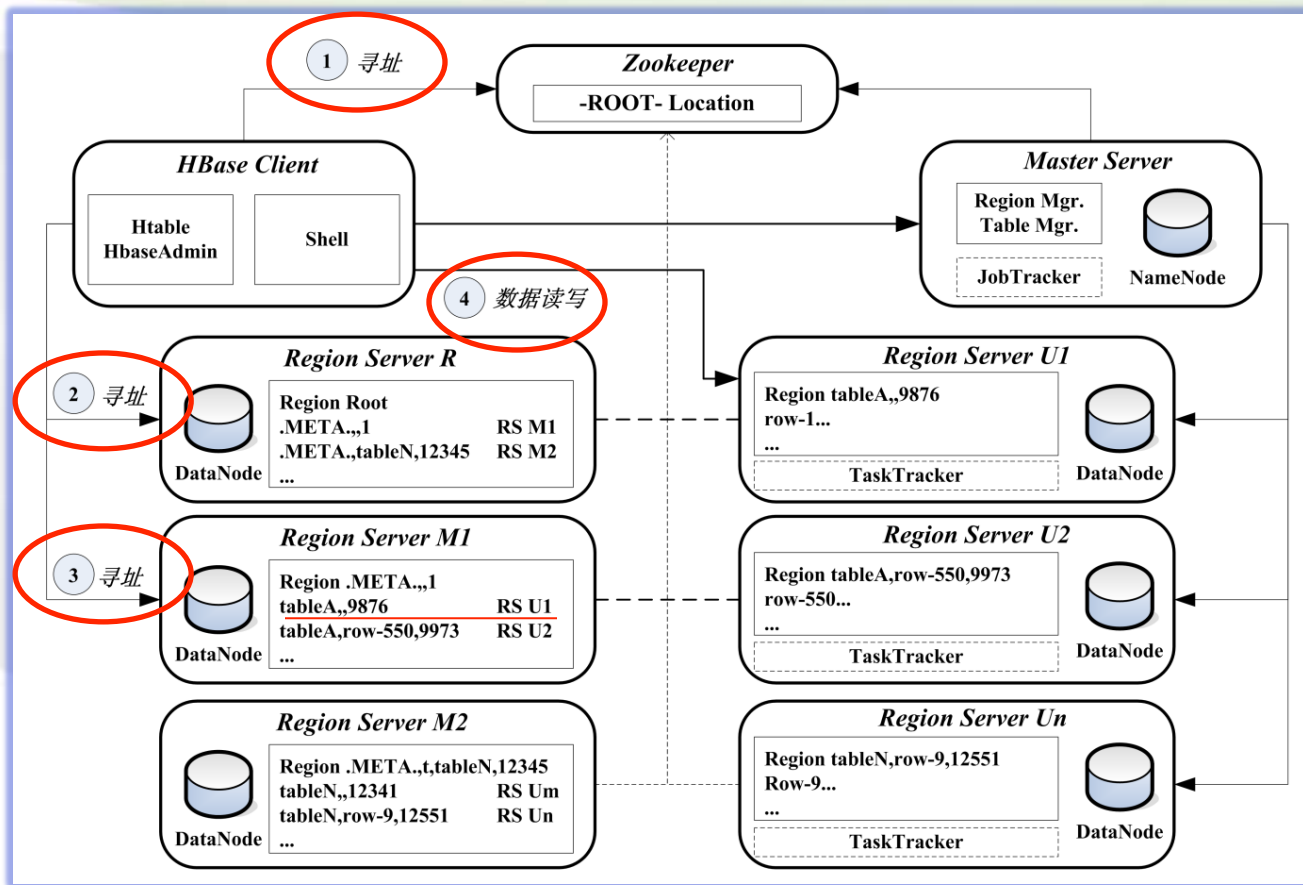
- **MasterServer控制节点**

- HBase的HMaster
- HDFS的NameNode
- MapReduce的JobTracker

- **RegionServer**

- R、M1、M2存放-ROOT-表和.META.表
- 数据表存放在Region Server U1至Un中
- Region Server U1至Un部署了HDFS的DataNode组件以提高数据访问效率
- Region Server U1至Un运行MapReduce作业时的TaskTracker

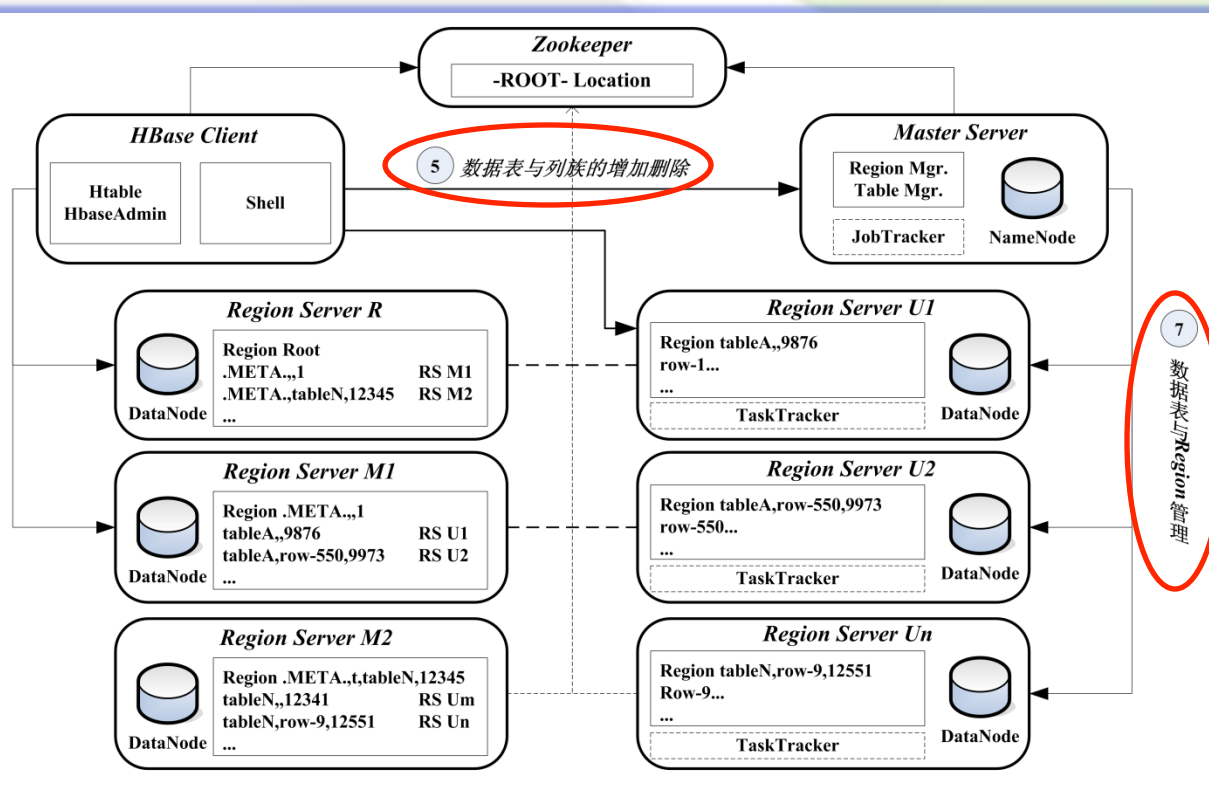
HBase 读/写 数据流程



- Client首次读取tableA中第1行数据：

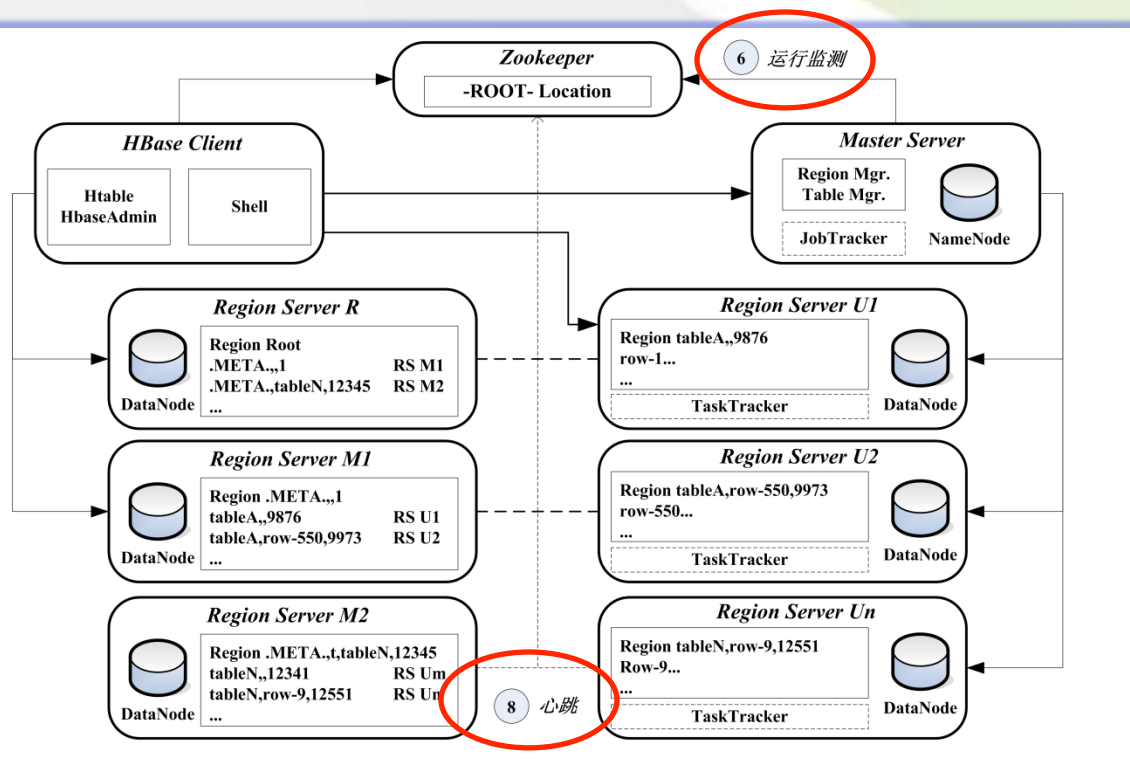
- 从Zookeeper中获取-ROOT-表的Region服务器R（步骤①）
- 从Region Server R中根据表的名称索引找到.META.表所在的Region服务器M1（步骤②）
- Client根据表名和行关键字找到对应的Region服务器U1（步骤③）
- 使用接口从U1进行数据读取/向U1写入数据（步骤④，**MemStore/LSM tree**）

HBase 表结构 操作流程



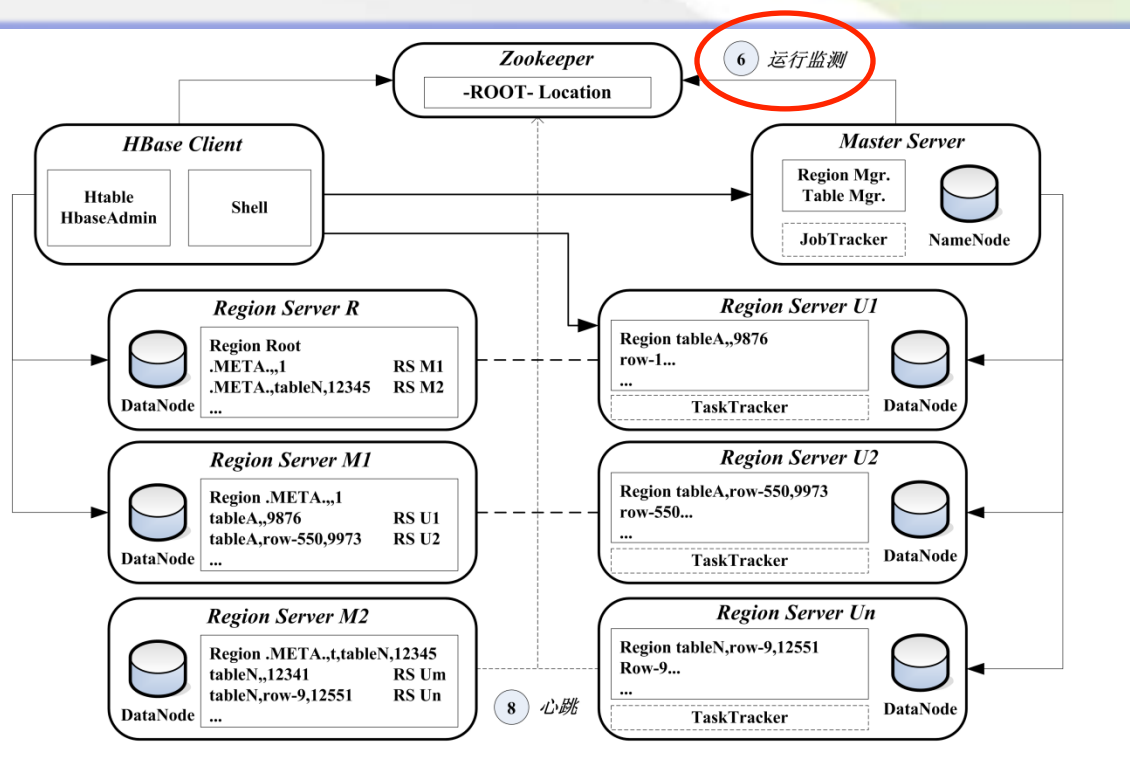
- MasterServer维护表结构
- 增加、删除表，增加、删除列族
 - Client通过Shell指令或API接口向Master Server发出请求（步骤⑤）
- 创建表
 - 默认情况在空间可用的RegionServer上新增1个Region（步骤⑦）
 - 更新.META.表
 - 所有后续的写入操作都会将数据存入此Region中，直到Region尺寸达到一定程度分裂为两个Region，并不断重复
- 动态增加列族
 - Master Server会根据用户请求，查找到可用的Region Server，并在相应的Region Server上为新的列族创建storeFile（步骤⑦）

RegionServer状态维护



- RegionServer在启动时，在Zookeeper上server列表目录下创建代表自己的文件，并获得该文件独占锁
- MasterServer通过订阅方式收到Zookeeper发来的server列表目录下的文件新增或删除消息（步骤⑥）以了解RegionServer状况
- RegionServer通过心跳消息与Zookeeper之间保持会话（步骤⑧）
- 节点或网络故障导致某个RegionServer与Zookeeper之间的会话断开时，Zookeeper会释放对应文件的独占锁，会被Master Server通过轮询发现，知道Region Server出现了问题，并进行随后的Region再分配和数据恢复操作

MasterServer状态维护



- MasterServer状态影响表结构、Region分配与合并、负载均衡等
- Master Server维护的数据，例如Region分布、表结构信息，都来自其他节点的复制
- 利用Zookeeper进行Master Server热备份的机制提高HBase的可用性
- Master Server失去与Zookeeper之间的心跳会话时（步骤⑥），可以基于Leader Election机制从备用Master Server中很快选择一个新的主MasterServer恢复HBase集群的正常服务

总结

- HBase三大要点：稀疏、海量、快速
- 稀疏：面向列的存储
- 海量：HDFS , Table→Region→Store→ HFile→Block→HDFS Block
- 快速：.META.、-ROOT- , B+ tree → LSM tree索引
- 部署与流程

作业

- 在Hadoop基础环境下，装上HBase
- 要求：
 - 安装过程及结果截图放到Word文档中，文件名：姓名_HBase安装
 - 发送到 liujun@bupt.edu.cn

