




足以封神小册：末流大学毕业的我竟然凭借这本“Java核心技术精讲”，疯狂收割了21个Offer!!!

原创

普通网友  于 2021-07-20 17:06:43 发布  26  收藏

文章标签: [Java](#) [架构师](#) [架构](#) [面试](#) [编程](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/Fightevery/article/details/118939818>

版权

作为一个资历不浅的 Java 开发, 这几年我面试过不少人。发现大多数面试者, 虽然看起来努力工作, 但他们表现出来的能力水平, 却不足以通过面试, 或拿到期望的薪资。

在我看来, 造成这种情况的原因, 主要有这么两方面:

第一, “知其然不知其所以然”。做了几年技术, 开发了一些业务应用, 但没有思考过这些技术选择背后的逻辑。所以, 我很难定位他们日后的成长潜力, 也不会放心把有一定深度的任务交给他们。

第二, 知识碎片化, 不成系统。事实上, 当面试者无法完整、清晰地描述自己所开发的系统或使用的相关技术时, 面试官就会怀疑他是否具备解决复杂问题、设计复杂系统的能力。

所以, 如果你平时只知道埋头苦干, 或过于死磕某个实现细节, 没有抬头审视过这些技术, 那么在准备面试时, 很有必要好好梳理一下 Java 知识体系, 这样才能拿下满意的 Offer。

这里, 分享一个帮了我不少的文档《**Java 核心技术精讲**》, 是我偶然得到的, 来自于我的一位朋友, 他看完了觉得写得很不错, 非常详细, 全面。但是有小部分内容还没更新完, 我问他是从哪来的, 他只告诉了我是一位工作十多年的大佬那拷贝来的。

文档涵盖知识:

面试必问的: Java基础、高并发、多线程、分布式、设计模式、Spring全家桶、Java、MyBatis、ZooKeeper、Dubbo、Elasticsearch、Memcached、MongoDB、Redis、MySQL、RabbitMQ、Kafka、Linux、Netty、Tomcat等等知识点详细讲解及面试押题。

由于整个文档比较全面, 内容比较多, 如有需要获取资料文档的朋友 文末有直达获取地址。

在文档中, 还从大厂面试考察的知识点和必备能力出发, 精选出 **485** 道 Java 面试题, 不仅给出典型回答和考点分析, 还剖析了 Java 核心知识点, 让你领悟面试所考察的关键能力, 帮你达到“知其所以然”和体系化的目标。



Java基础

- Java注解
- Java反射
- Java泛型
- Java内部类
- Java复制

- Java序列化
- Java异常分类处理



Java基础脑图

JAVA基础对应详细解析文档

书签

Q 书签查找

5. JAVA 基础 101

- 5.1.1. JAVA异常分类及... 101
 - 5.1.1.1. 概念 101
 - 5.1.1.2. 异常分类 101
 - 5.1.1.3. 异常的处理方... 102
 - 5.1.1.4. Throw和thro... 102
- 5.1.2. JAVA反射 103
 - 5.1.2.1. 动态语言 103
 - 5.1.2.2. 反射机制概念... 103
 - 5.1.2.3. 反射的应用场... 103
 - 5.1.2.4. Java反射API 104
 - 5.1.2.5. 反射使用步骤... 104
 - 5.1.2.6. 获取Class对... 104
 - 5.1.2.7. 创建对象的... 105
- 5.1.3. JAVA注解 106
 - 5.1.3.1. 概念 106
 - 5.1.3.2. 4种标准元注解 106
 - @Documented描述... 106
 - @Inherited阐述了美... 106
 - 5.1.3.3. 注解处理器 107
- 5.1.4. JAVA内部类 109
 - 5.1.4.1. 静态内部类 109
 - 5.1.4.2. 成员内部类 110
 - 5.1.4.3. 局部的内部类 (... 110
 - 5.1.4.4. 匿名内部类 (... 111
- 5.1.5. JAVA泛型 112
 - 5.1.5.1. 泛型方法 (<... 112
 - 5.1.5.2. 泛型类 <T> 112
 - 5.1.5.3. 类型通配符? 113
 - 5.1.5.4. 类型擦除 113

5. JAVA 基础

5.1.1. JAVA 异常分类及处理

5.1.1.1. 概念

如果某个方法不能按照正常的途径完成任务,就可以通过另一种路径退出方法。在这种情况下会抛出一个封装了错误信息的对象。此时,这个方法会立刻退出同时不返回任何值。另外,调用这个方法的其他代码也无法继续执行,异常处理机制会将代码执行交给异常处理器。

5.1.1.2. 异常分类

Throwable 是 Java 语言中所有错误或异常的超类。下一层分为 Error 和 Exception

Error

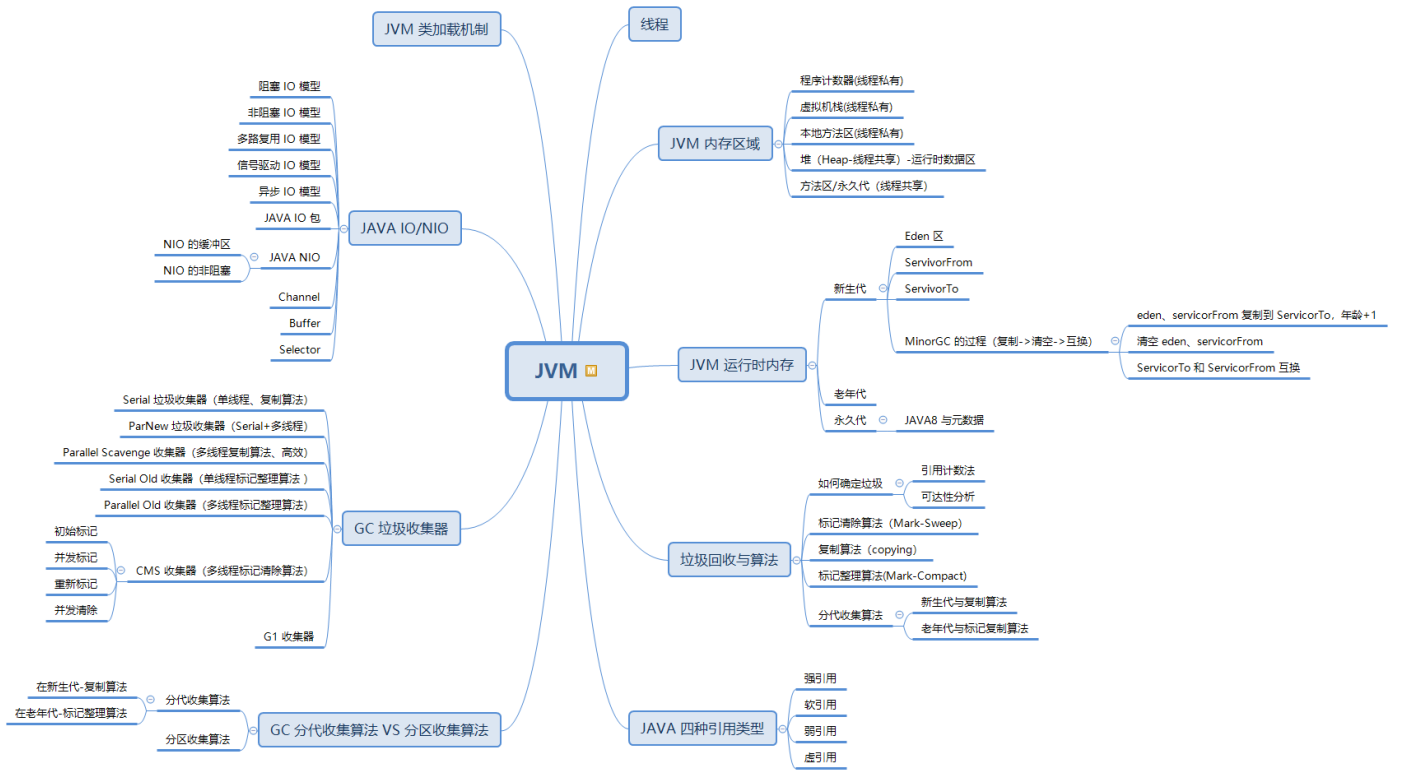
- Error 类是指 java 运行时系统的内部错误和资源耗尽错误。应用程序不会抛出该类对象。如果出现了这样的错误,除了告知用户,剩下的就是尽力使程序安全的终止。

JAVA基础对应详细解析文档

Java多线程并发

- JAVA 并发知识库
- JAVA 线程实现/创建方式
- 4 种线程池
- 线程生命周期(状态)

- 终止线程 4 种方式
- sleep 与 wait 区别
- start 与 run 区别
- JAVA 后台线程
- JAVA 锁
- 线程基本方法
- 线程上下文切换
- 同步锁与死锁
- 线程池原理
- JAVA 阻塞队列原理
- CyclicBarrier、CountDownLatch、Semaphore 的用法
- volatile 关键字的作用（变量可见性、禁止重排序）
- 如何在两个线程之间共享数据
- ThreadLocal 作用（线程本地存储）
- synchronized 和 ReentrantLock 的区别
- ConcurrentHashMap 并发
- Java 中用到的线程调度
- 进程调度算法
- 什么是 CAS（比较并交换-乐观锁机制-锁自旋）
- 什么是 AQS（抽象的队列同步器）



JVM脑图

JVM脑图对应详细文档解析

The document viewer shows a table of contents on the left and a detailed mind map on the right. The mind map is centered on 'JVM' and branches into three main areas:

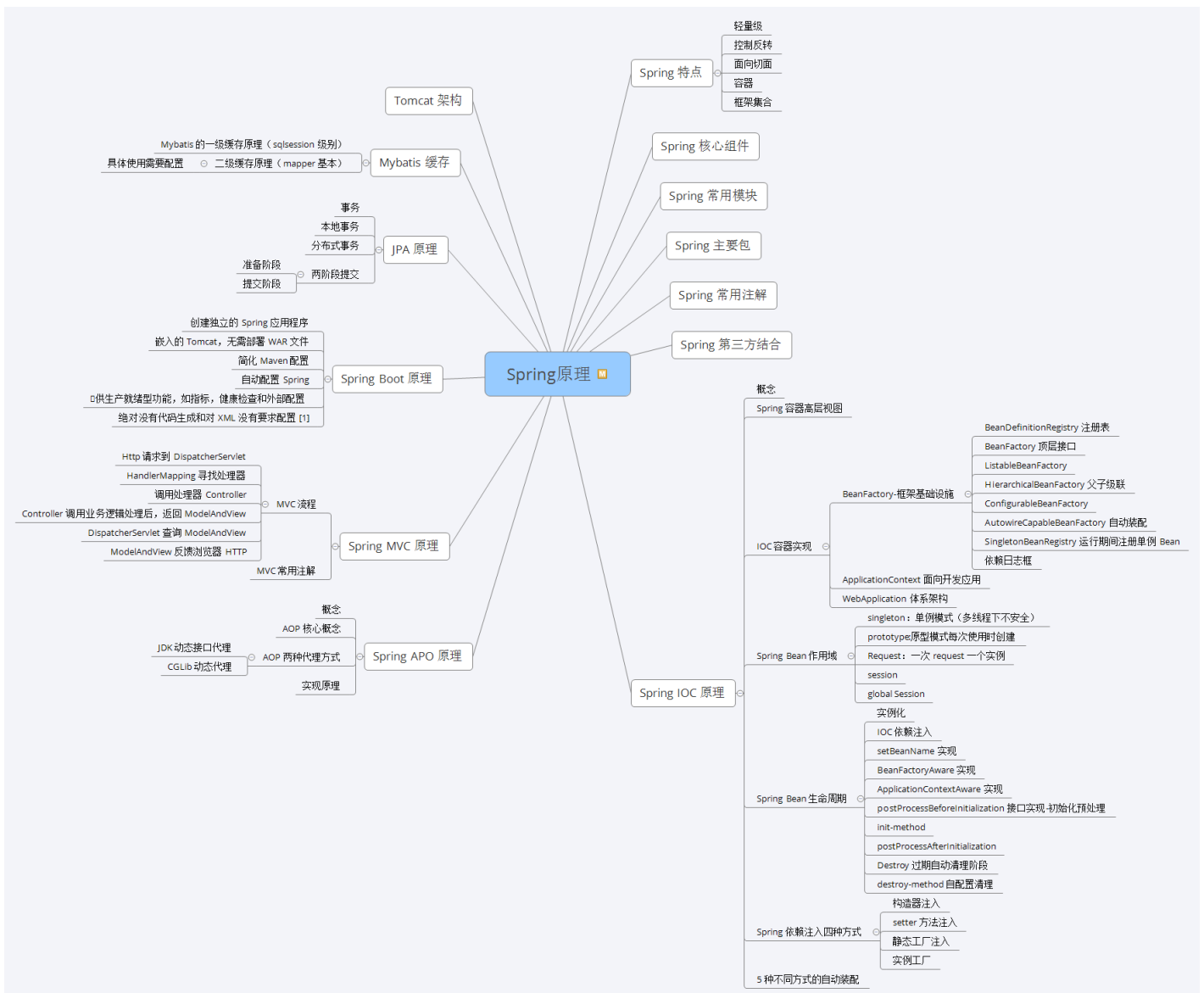
- Java代码的执行** (Execution of Java Code)
 - 代码编译为class - javac
 - 装载class - ClassLoader
 - 执行class
 - 解释执行
 - 编译执行
 - client compiler
 - server compiler
- 内存管理** (Memory Management)
 - 内存空间
 - 方法区
 - 堆
 - 方法栈
 - 本地方法栈
 - pc寄存器
 - 内存分配
 - 堆上分配
 - TLAB分配
 - 栈上分配
 - 算法
 - Copy
 - Mark-Sweep
 - Mark-Compact
 - 内存回收
 - 分代回收
 - 新生代可用的GC
 - 串行copying
 - 并行回收copying
 - 并行copying
 - Minor GC触发机制以及日志格式
 - 旧世代可用的GC
 - 串行Mark-Sweep-Compact
 - 并行Compacting
 - 并发Mark-Sweep
 - Full GC触发机制以及日志格式
 - GC参数
 - Sun JDK
 - G1
 - 内存状况分析
 - jconsole
 - visualvm
 - jstat
 - jmap
 - MAT
- 线程资源同步和交互机制** (Thread Resource Synchronization and Interaction Mechanisms)
 - 线程资源同步
 - 线程资源执行机制
 - 线程资源同步机制
 - Synchronized的实现机制
 - lock/unlock的实现机制
 - 线程交互机制
 - Object.wait/notify/notifyAll - Double check pattern
 - 并发包提供的交互机制
 - semaphore
 - CountdownLatch
 - 线程状态及分析方法
 - jstack
 - TDA

JVM详细文档解析

Spring原理

- Spring 特点

- Spring 核心组件
- Spring 常用模块
- Spring 主要包
- Spring 常用注解
- Spring 第三方结合
- Spring IOC 原理
- Spring AOP 原理
- Spring MVC 原理
- Spring Boot 原理
- JPA 原理
- Mybatis 缓存
- Tomcat 架构
-



Spring原理脑图

Spring原理对应详细解析文档

书签

Q 书签查找

返回 刷新 主页

- 6. Spring 原理 116
 - 6.1.1. Spring 特点 116
 - 6.1.2. Spring 核心组件 117
 - 6.1.3. Spring 常用模块 117
 - 6.1.4. Spring 主要包 118
 - 6.1.5. Spring 常用注解 118
 - 6.1.6. Spring 第三方结合 119
 - 6.1.7. Spring IOC原理 120
 - 6.1.7.1. 概念 120
 - 6.1.7.2. Spring容器... 120
 - 6.1.7.3. IOC容器实现 120
 - 6.1.7.4. Spring Bean... 123
 - 6.1.7.5. Spring Bean... 124
 - 6.1.7.6. Spring 依赖... 126
 - 6.1.7.7. 5种不同方式... 128
 - 6.1.8. Spring APO原理 129
 - 6.1.8.1. 概念 129
 - 6.1.8.2. AOP核心概念 129
 - 6.1.8.1. AOP两种代... 130
 - 6.1.8.2. 实现原理 131
 - 6.1.9. Spring MVC原理 132
 - 6.1.10. Spring Boot原理 134
 - 1. 创建独立的Spring... 134
 - 2. 嵌入的Tomcat, ... 134
 - 3. 简化Maven配置 134
 - 4. 自动配置Spring 134
 - 5. 提供生产就绪型功... 134
 - 6. 绝对没有代码生成... 134
 - 6.1.11. JPA原理 134
 - 6.1.12. Mybatis缓存 137

6. Spring 原理

它是一个全面的、企业应用开发一站式的解决方案，贯穿表现层、业务层、持久层。但是 Spring 仍然可以和其他的框架无缝整合。

6.1.1. Spring 特点

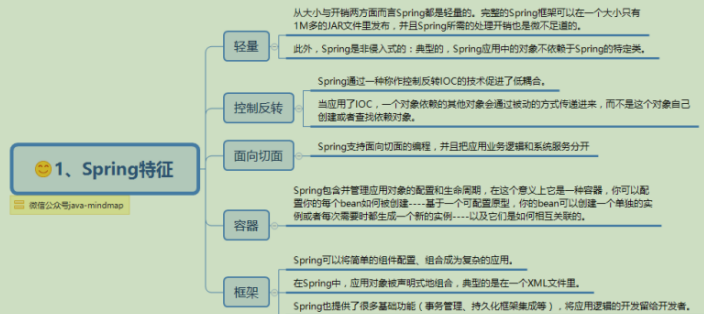
6.1.1.1. 轻量级

6.1.1.2. 控制反转

6.1.1.3. 面向切面

6.1.1.4. 容器

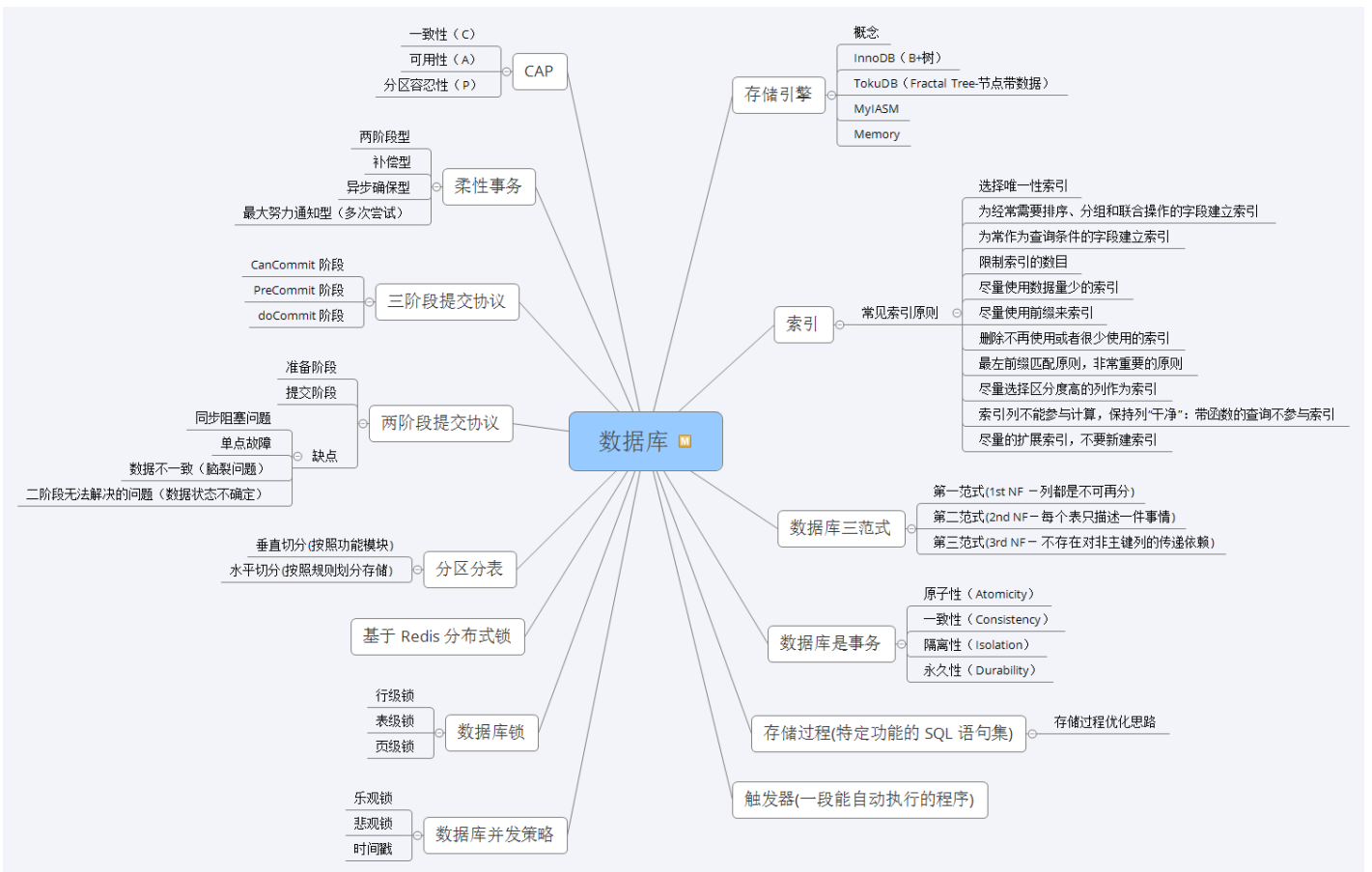
6.1.1.5. 框架集合



Spring原理对应详细解析文档

数据库

- 存储引擎
- 索引
- 数据库三范式
- 数据库是事务
- 存储过程
- 触发器
- 数据库并发策略
- 数据库锁
- 基于Redis分布式锁
- 分区分表
- 两阶段提交协议
- 三阶段提交协议
- 柔性事务
- CAP



数据库对应详细解析文档

书签

Q 书签查找

19. 数据库 214

- 19.1.1. 存储引擎 214
 - 19.1.1.1. 概念 214
 - 19.1.1.2. InnoDB (B+树) 214
 - 19.1.1.3. TokuDB (Fractal Tree-节点带数据) 215
 - 19.1.1.4. MyIASM 215
 - 19.1.1.5. Memory 215
- 19.1.2. 索引 215
- 19.1.3. 数据库三范式 216
 - 19.1.3.1. 第一范式(1st NF - 列都是不可再分) 216
 - 19.1.3.2. 第二范式(2nd NF - 每个表只描述一件事情) 216
 - 19.1.3.3. 第三范式(3rd NF - 不存在对非主键列的传递依赖) 217
- 19.1.4. 数据库是事务 217
 - 原子性 (Atomicity) 217
 - 一致性 (Consistency) 217
 - 隔离性 (Isolation) 218
 - 永久性 (Durability) 218
- 19.1.5. 存储过程(特定功能的 SQL 语句集) 218
- 19.1.6. 触发器(一段能自动执行的程序) 218
- 19.1.7. 数据库并发策略 218
- 19.1.8. 数据库锁 219
 - 基于 Redis 分布式锁 219
- 19.1.10. 分区分表 220
 - 垂直切分(按照功能模块) 220
 - 水平切分(按照规则划分存储) 220
- 19.1.11. 两阶段提交协议 220
 - 准备阶段 221
 - 提交阶段 221
- 19.1.11.3. 缺点 221
- 19.1.12. 三阶段提交协议 222

19. 数据库

19.1.1. 存储引擎

19.1.1.1. 概念

数据库存储引擎是数据库底层软件组织，数据库管理系统 (DBMS) 使用数据库引擎进行创建、查询、更新和删除数据。不同的存储引擎提供不同的存储机制、索引技巧、锁定水平等功能，使用不同的存储引擎，还可以获得特定的功能。现在许多不同的数据库管理系统都支持多种不同的数据引擎。存储引擎主要有：1. MyISAM, 2. InnoDB, 3. Memory, 4. Archive, 5. Federated。

19.1.1.2. InnoDB (B+树)

InnoDB 底层存储结构为 B+树，B 树的每个节点对应 innodb 的一个 page，page 大小是固定的，一般设为 16k。其中非叶子节点只有键值，叶子节点包含完整数据。

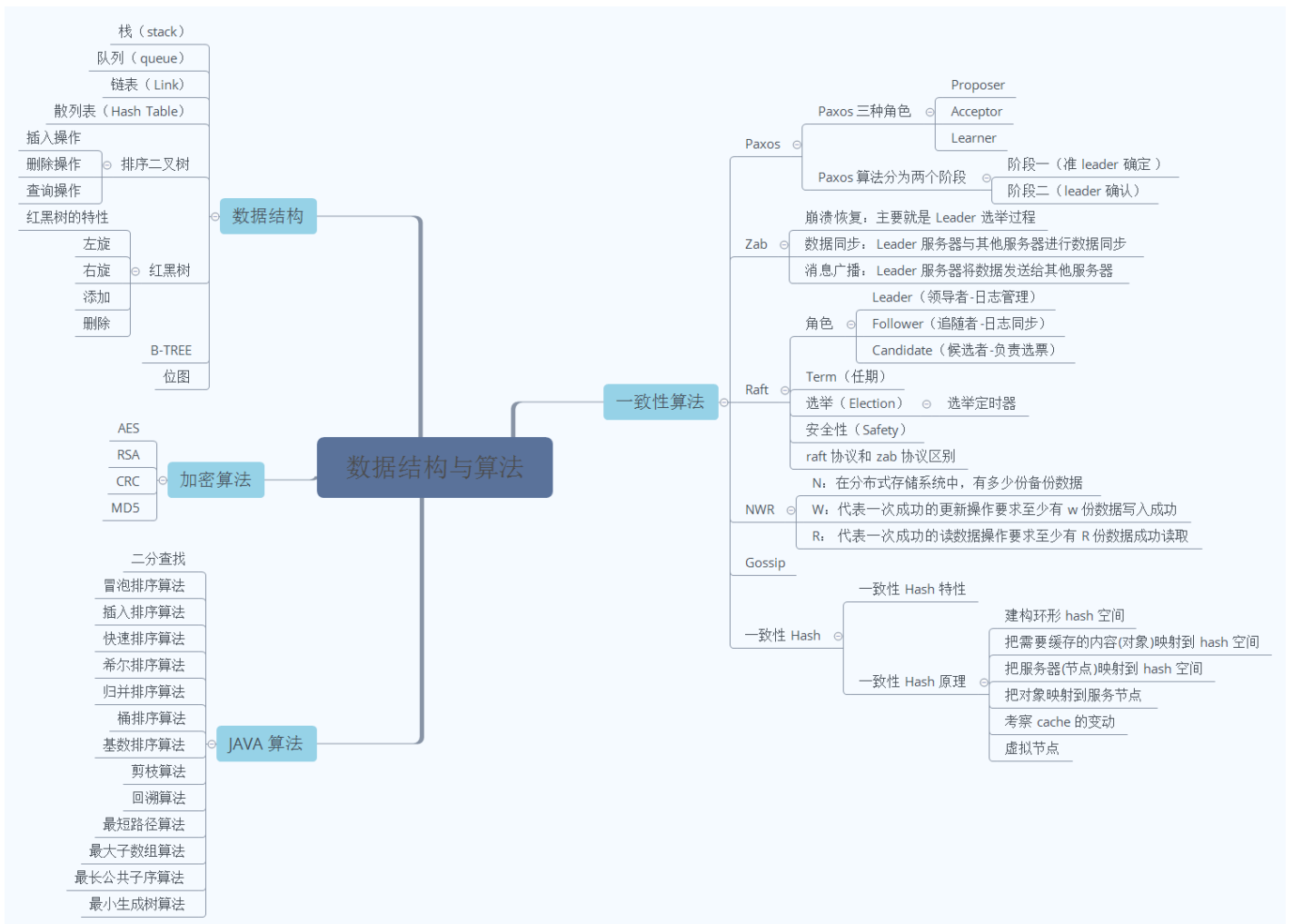
适用场景：

- 经常更新的表，适合处理多重并发的更新请求。
- 支持事务。

数据库对应详细解析文档

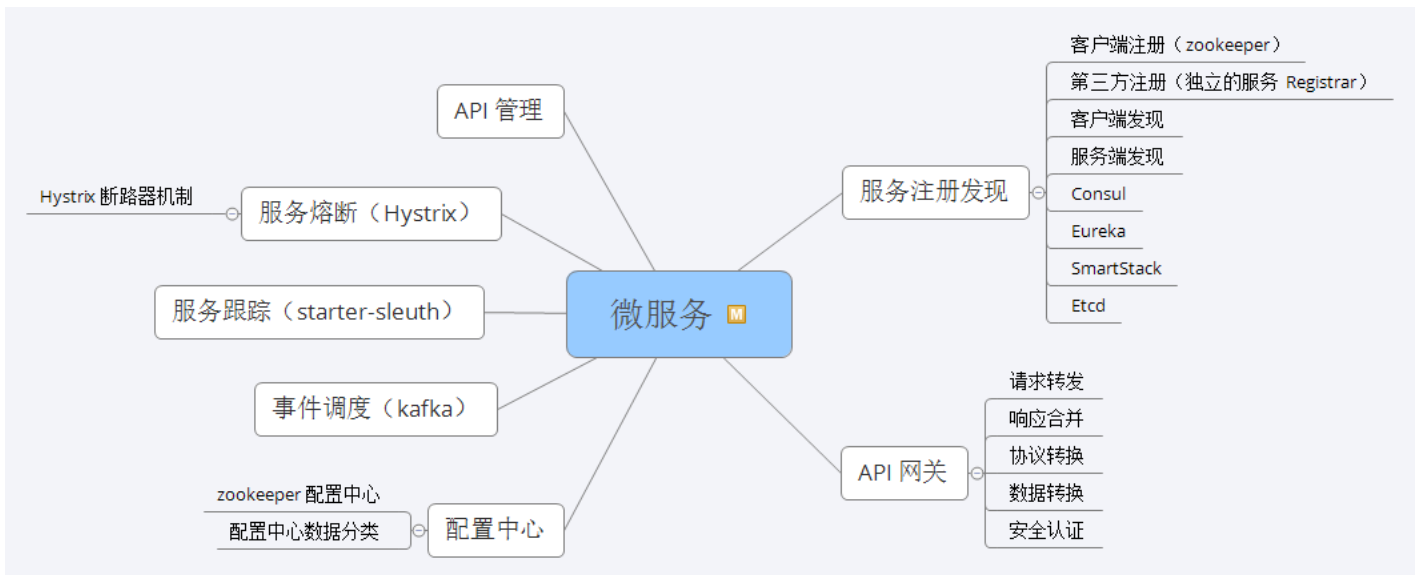
数据结构与算法

- 数据结构
- 加密算法
- JAVA 算法
- 一致性算法



微服务

- 服务注册发现
- API 网关
- 配置中心
- 事件调度 (kafka)
- 服务跟踪 (starter-sleuth)
- 服务熔断 (Hystrix)
- API 管理



微服务脑图

微服务对于解析文档

书签

Q 书签查找

7. 微服务 140

- 7.1.1. 服务注册发现 140
 - 7.1.1.1. 客户端注册 (...) 140
 - 7.1.1.2. 第三方注册 (...) 140
 - 7.1.1.3. 客户端发现 141
 - 7.1.1.4. 服务端发现 142
 - 7.1.1.5. Consul 142
 - 7.1.1.6. Eureka 142
 - 7.1.1.7. SmartStack 142
 - 7.1.1.8. Etcd 142
- 7.1.2. API 网关 142
 - 7.1.2.1. 请求转发 143
 - 7.1.2.2. 响应合并 143
 - 7.1.2.3. 协议转换 143
 - 7.1.2.4. 数据转换 143
 - 7.1.2.5. 安全认证 144
- 7.1.3. 配置中心 144
 - 7.1.3.1. zookeeper... 144
 - 7.1.3.2. 配置中心数据... 144
- 7.1.4. 事件调度 (kafka) 144
- 7.1.5. 服务跟踪 (starter... 144
- 7.1.6. 服务熔断 (Hystri... 145
 - 7.1.6.1. Hystrix断路... 146
- 7.1.7. API管理 146

7. 微服务

7.1.1. 服务注册发现

服务注册就是维护一个登记簿，它管理系统内所有的服务地址。当新的服务启动后，它会向登记簿交待自己的地址信息。服务的依赖方直接向登记簿要 Service Provider 地址就行了。当下用于服务注册的工具非常多 ZooKeeper, Consul, Etcd, 还有 Netflix 家的 eureka 等。服务注册有两种形式：客户端注册和第三方注册。

7.1.1.1. 客户端注册 (zookeeper)

客户端注册是服务自身要负责注册与注销的工作。当服务启动后向注册中心注册自身，当服务下线时注销自己。期间还需要和注册中心保持心跳。心跳不一定要客户端来做，也可以由注册中心负责（这个过程叫探活）。这种方式的缺点是注册工作与服务耦合在一起，不同语言都要实现一套注册逻辑。

10.4.3.1:8756

REST API

SERVICE INSTANCE A

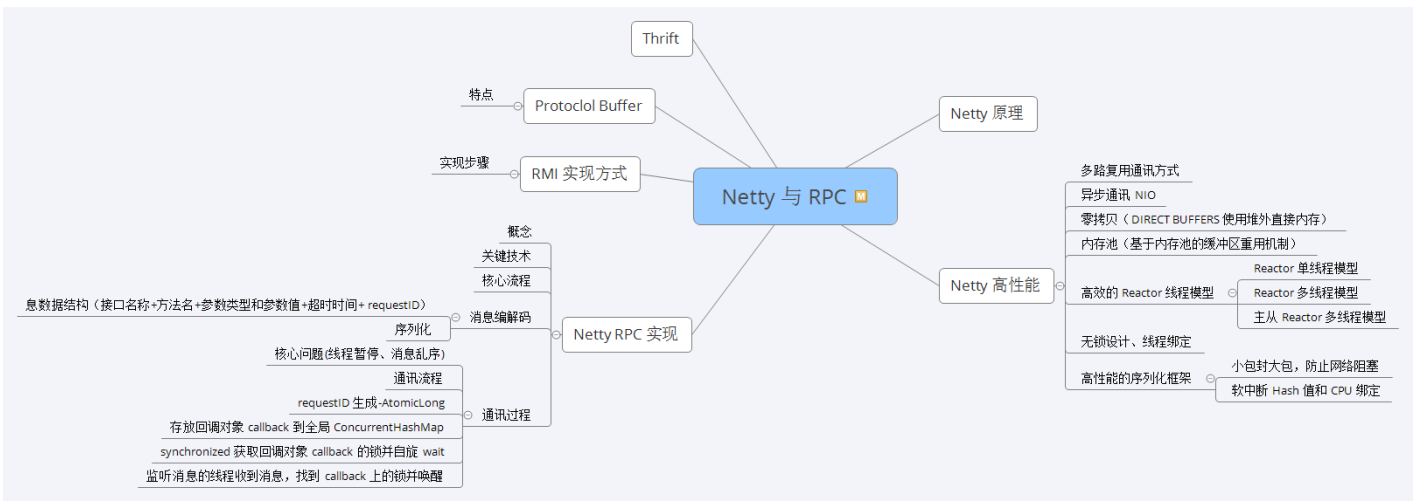
register("serviceName", "10.4.3.1:8756")
heartbeat()
unregister()

SERVICE REGISTRY

微服务对于解析文档

Netty 与 RPC

- Netty 原理
- Netty 高性能
- Netty RPC 实现
- RMI 实现方式
- Protocol Buffer
- Thrift



Netty 与 RPC脑图

Netty 与 RPC对应详细解析文档

书签

Q 书签查找

8. Netty 与 RPC 147

- 8.1.1. Netty 原理 147
- 8.1.2. Netty 高性能 147
 - 8.1.2.1. 多路复用通讯... 147
 - 8.1.2.1. 异步通讯NIO 148
 - 8.1.2.2. 零拷贝 (DIR... 149
 - 8.1.2.3. 内存池 (基于... 149
 - 8.1.2.4. 高效的React... 149
 - Reactor 单线程模型 149
 - Reactor 多线程模型 150
 - 主从Reactor多线... 150
 - 8.1.2.5. 无锁设计、线... 151
 - 8.1.2.6. 高性能的序列... 151
 - 小包封大包, 防止... 152
 - 软中断Hash值和C... 152
- 8.1.3. Netty RPC实现 152
 - 8.1.3.1. 概念 152
 - 8.1.3.2. 关键技术 152
 - 8.1.3.3. 核心流程 152
 - 8.1.3.1. 消息编解码 153
 - 8.1.3.1. 通过程序 154
- 8.1.4. RMI实现方式 155
 - 8.1.4.1. 实现步骤 155
- 8.1.5. Protocol Buffer 156
 - 8.1.5.1. 特点 157
- 8.1.6. Thrift 157

8. Netty 与 RPC

8.1.1. Netty 原理

Netty 是一个高性能、异步事件驱动的 NIO 框架，基于 JAVA NIO 提供的 API 实现。它提供了对 TCP、UDP 和文件传输的支持，作为一个异步 NIO 框架，Netty 的所有 IO 操作都是异步非阻塞的，通过 Future-Listener 机制，用户可以方便的主动获取或者通过通知机制获得 IO 操作结果。

8.1.2. Netty 高性能

在 IO 编程过程中，当需要同时处理多个客户端接入请求时，可以利用多线程或者 IO 多路复用技术进行处理。IO 多路复用技术通过把多个 IO 的阻塞复用到同一个 select 的阻塞上，从而使得系统在单线程的情况下可以同时处理多个客户端请求。与传统的多线程/多进程模型比，I/O 多路复用的最大优势是系统开销小，系统不需要创建新的额外进程或者线程，也不需要维护这些进程和线程的运行，降低了系统的维护工作量，节省了系统资源。

与 Socket 类和 ServerSocket 类相对应，NIO 也提供了 SocketChannel 和 ServerSocketChannel 两种不同的套接字通道实现。

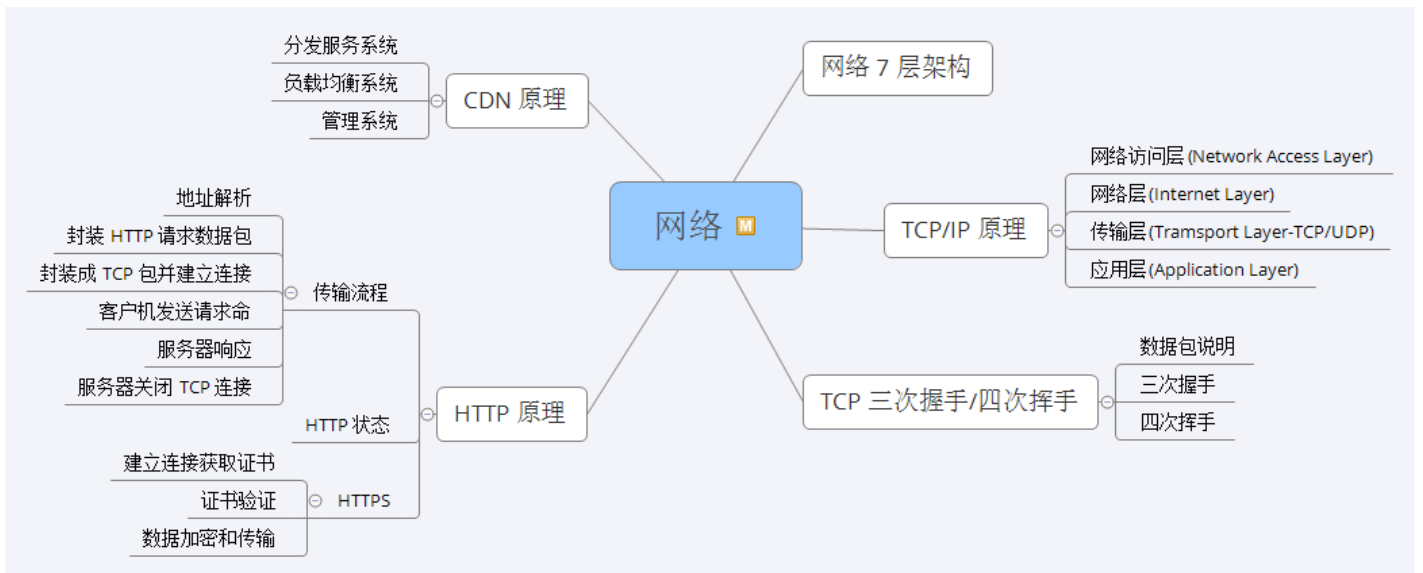
8.1.2.1. 多路复用通讯方式

Netty 架构按照 Reactor 模式设计和实现，它的服务端通信序列图如下：

Netty 与 RPC对应详细解析文档

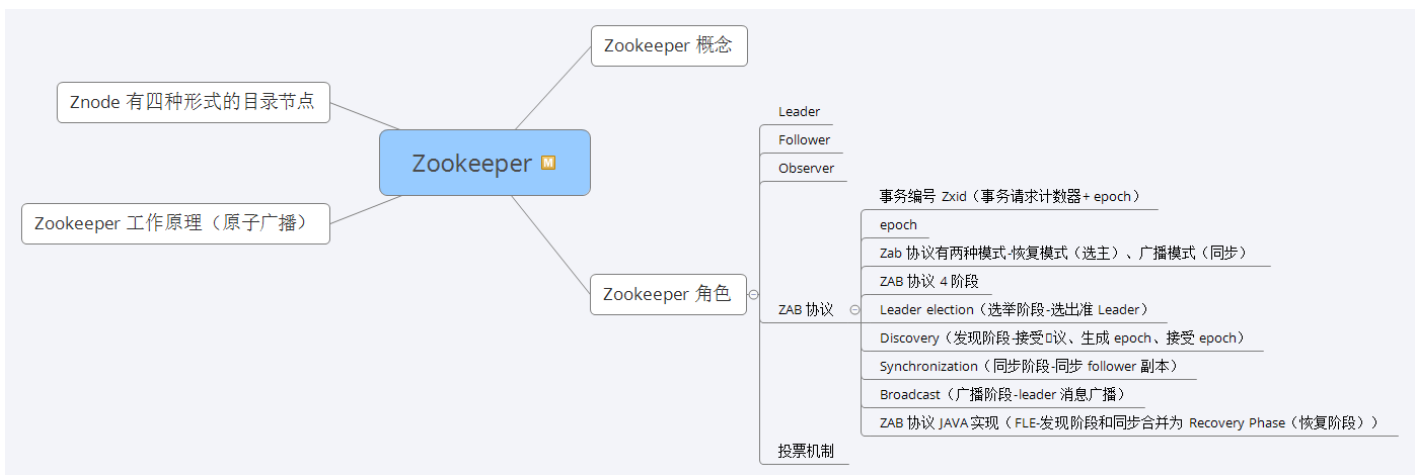
网络

- 网络 7 层架构
- TCP/IP 原理
- TCP 三次握手/四次挥手
- HTTP 原理
- CDN 原理



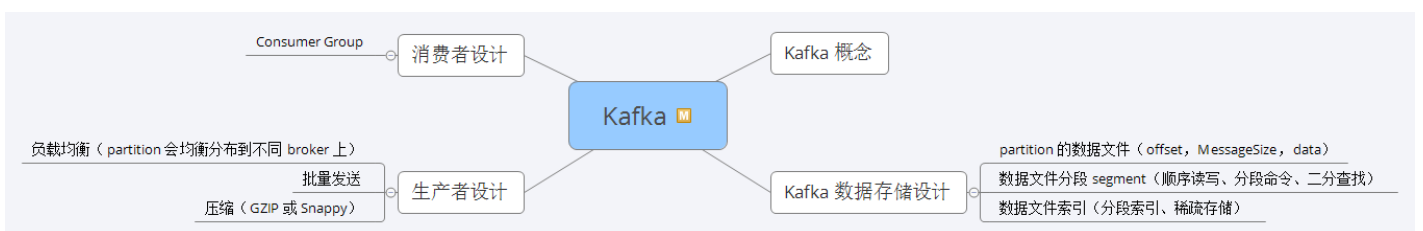
Zookeeper

- Zookeeper 概念
- Zookeeper 角色
- Zookeeper 工作原理（原子广播）
- Znode 有四种形式的目录节点



Kafka

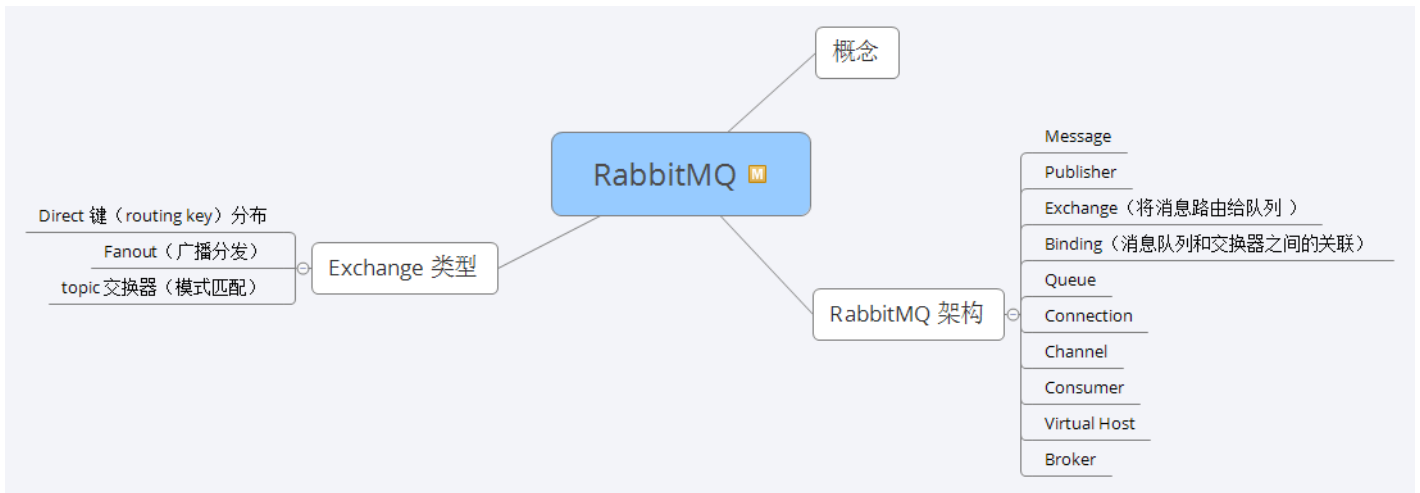
- Kafka 概念
- Kafka 数据存储设计
- 生产者设计
- 消费者设计



RabbitMQ

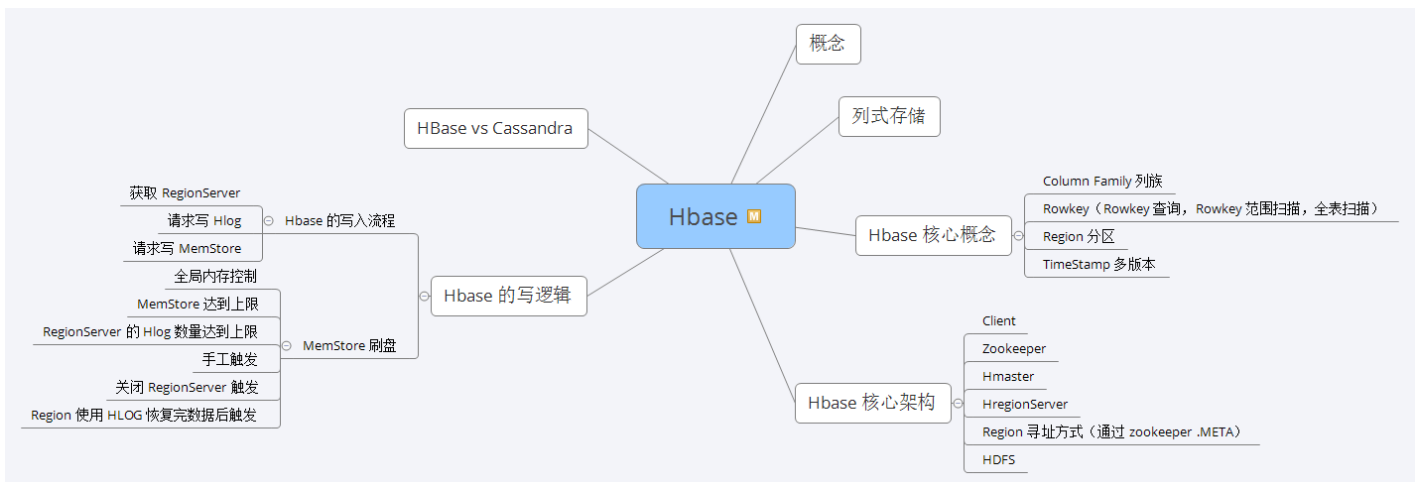
- 概念
- RabbitMQ 架构

- Exchange 类型



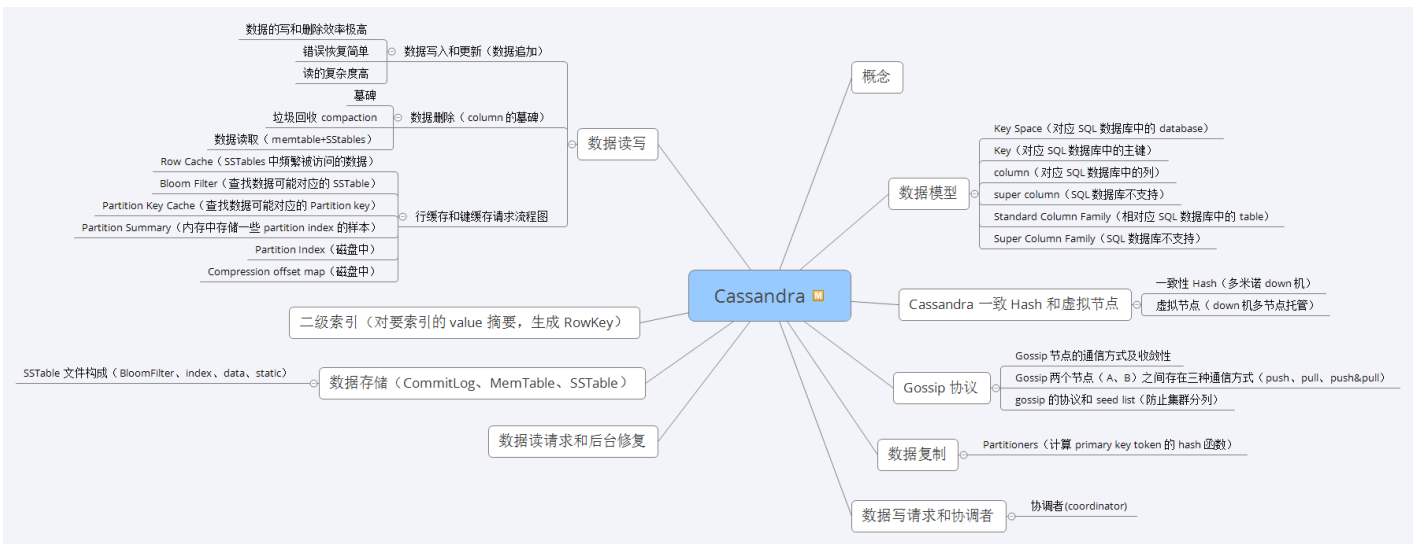
Hbase

- 概念
- 列式存储
- Hbase 核心概念
- Hbase 核心架构
- Hbase 的写逻辑
- HBase vs Cassandra



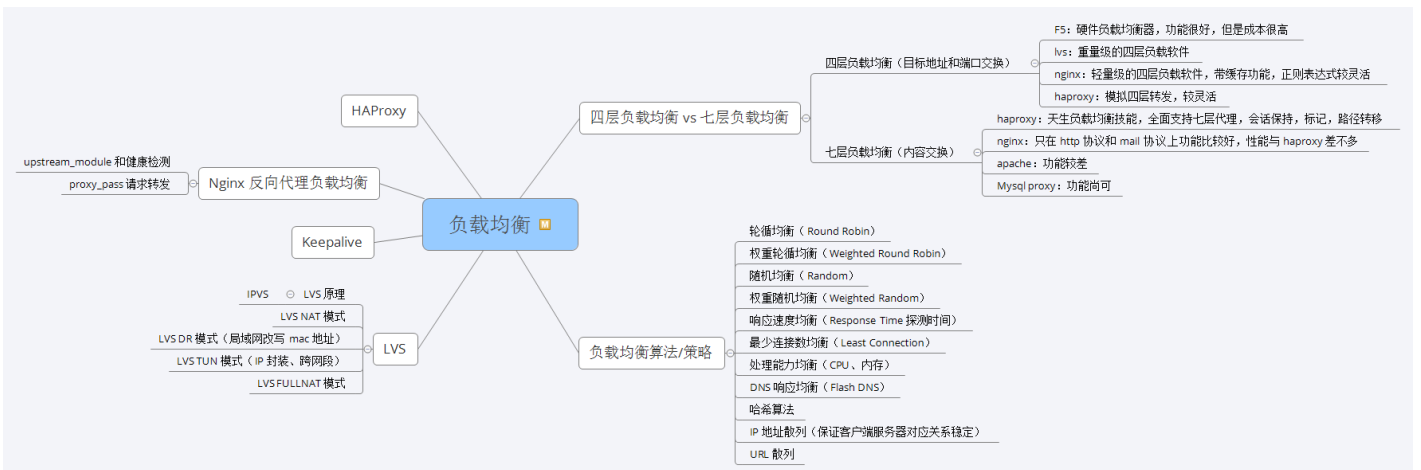
Cassandra

- 概念
- 数据模型
- Cassandra 一致 Hash 和虚拟节点
- Gossip 协议
- 数据复制
- 数据写请求和协调者
- 数据读请求和后台修复
- 数据存储 (CommitLog、MemTable、SSTable)
- 二级索引 (对要索引的 value 摘要, 生成 RowKey)
- 数据读写



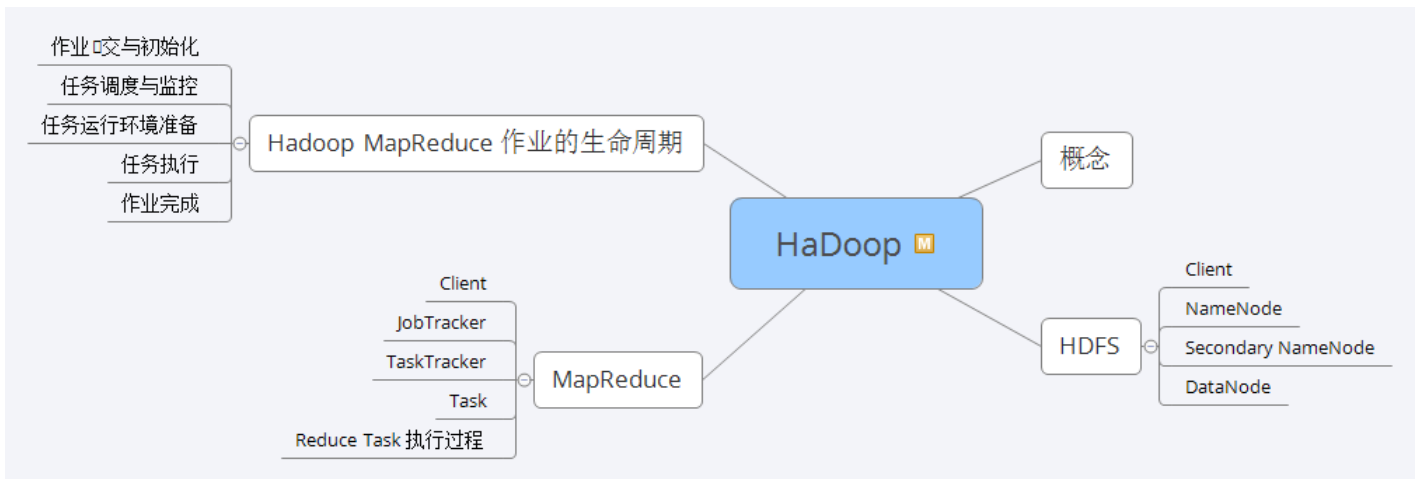
负载均衡

- 四层负载均衡 vs 七层负载均衡
- 负载均衡算法/策略
- LVS
- Keepalive
- Nginx 反向代理负载均衡
- HAProxy



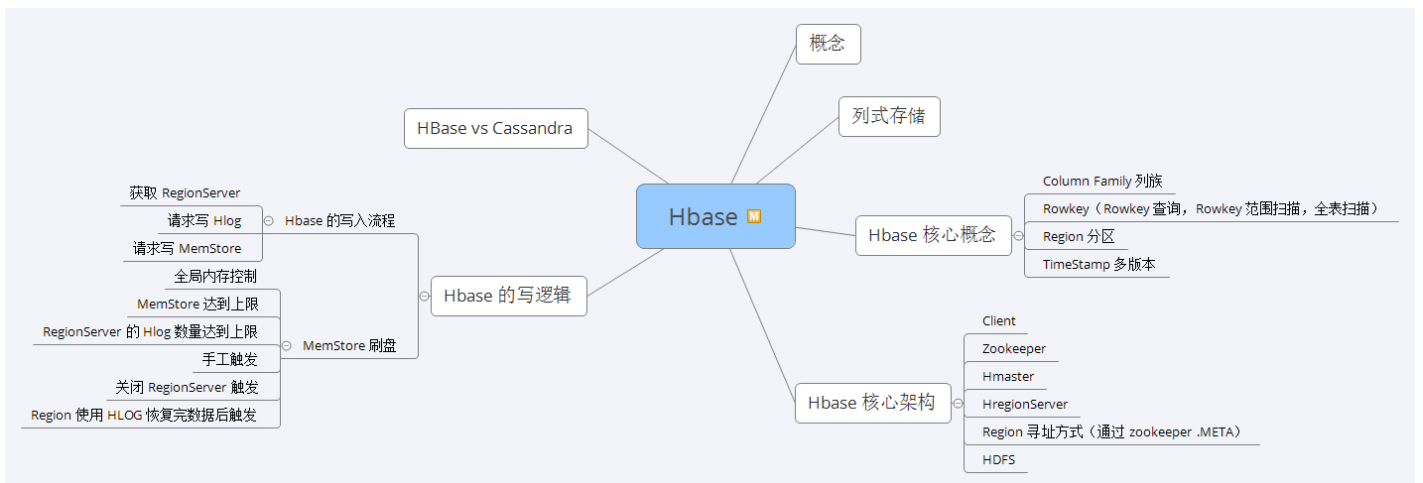
HaDooop

- 概念
- HDFS
- MapReduce
- Hadoop MapReduce 作业的生命周期



Spark

- 概念
- 核心架构
- 核心组件
- SPARK 编程模型
- SPARK 计算模型
- SPARK 运行流程
- SPARK RDD 流程
- SPARK RDD



获取方式：有需要的小伙伴，点赞加收藏，关注我之后添加小助理vx：**bjmsb0606006** 即可获取免费下载方式