Parallel FP/OO Programming with Java

李力
Slides: Click Here
2022年8月3日

内容简介

The content of the title

- Java并发与并行历史
  - 函数式编程与面向对象编程
  - 结构化并发/并行
- 三大并行框架
  - 并行流（函数式框架）
  - Fork-join pool（面向对象的框架）
  - CompletableFututre（反应式异步框架）
- 反应式流
  - Flow,Akka-stream,Rxjava
- 项目案例研究
  - its，quiz，common-utils
  - concurrency-practice工程介绍

并发与并行

The content of the title

- 并发：控制访问共享资源的正确性和效率。
- 并行：增加额外资源让答案产生更快。
  - 分治
  - 粗粒度的任务并行和细粒度数据并行
- 求1到1亿的总和.
  - 串行
  - 并发
  - 并行

```
int sumSeq(int[] array) {
    int sum = 0;
    for (int i : array)
        sum = sum + i;
    return sum;
}
```

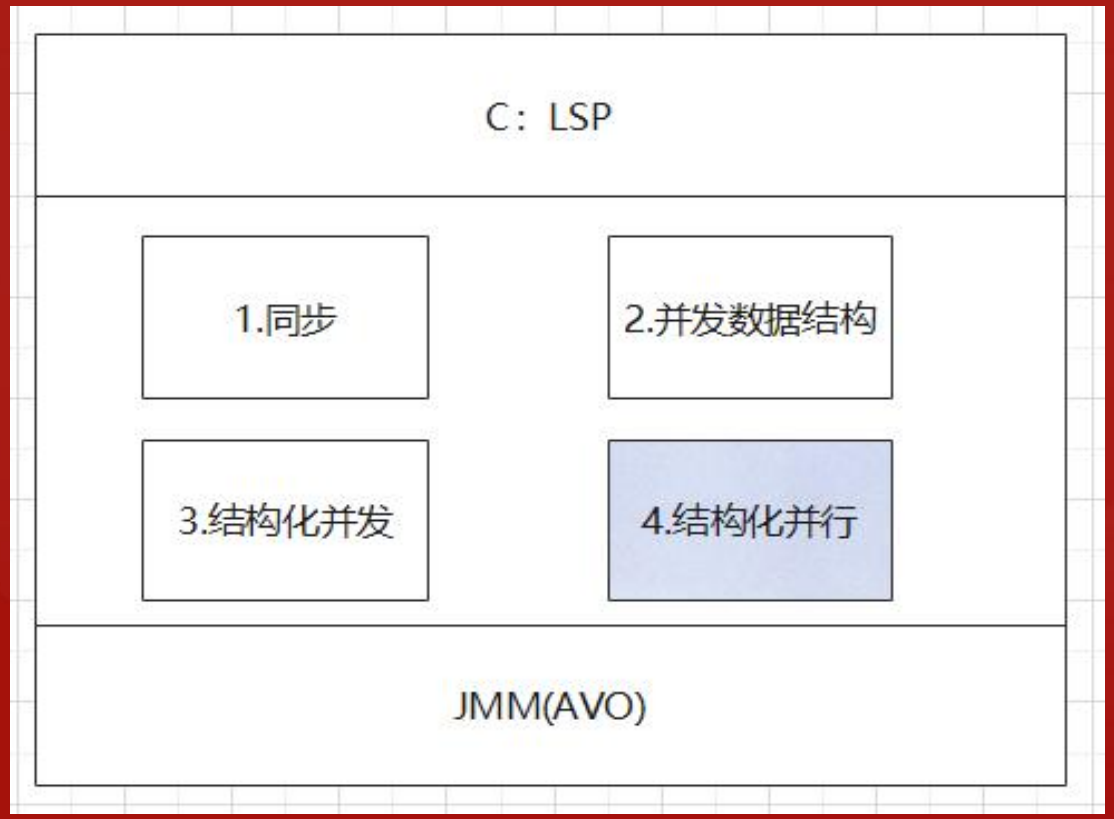
```
int sumConcurrent(int[] array) {
    int mid = array.length / 2;
    int sum = 0;
    CONCURRENT {
        {
            for (int i = 0; i < mid; i++)
                ATOMIC { sum = sum + array[i]; }
        }
        {
            for (int i = mid; i < array.length; i++)
                ATOMIC { sum = sum + array[i]; }
        }
    }
    return sum;
}
```

```
int sumPartitioned(int[] array) {
    int mid = array.length / 2;
    int leftSum = 0, rightSum = 0;
    CONCURRENT {
        {
            for (int i = 0; i < mid; i++)
                leftSum = leftSum + array[i];
        }
        {
            for (int i = mid; i < array.length; i++)
                rightSum = rightSum + array[i];
        }
    }
    return leftSum + rightSum;
}
```

Java并发与并行历史

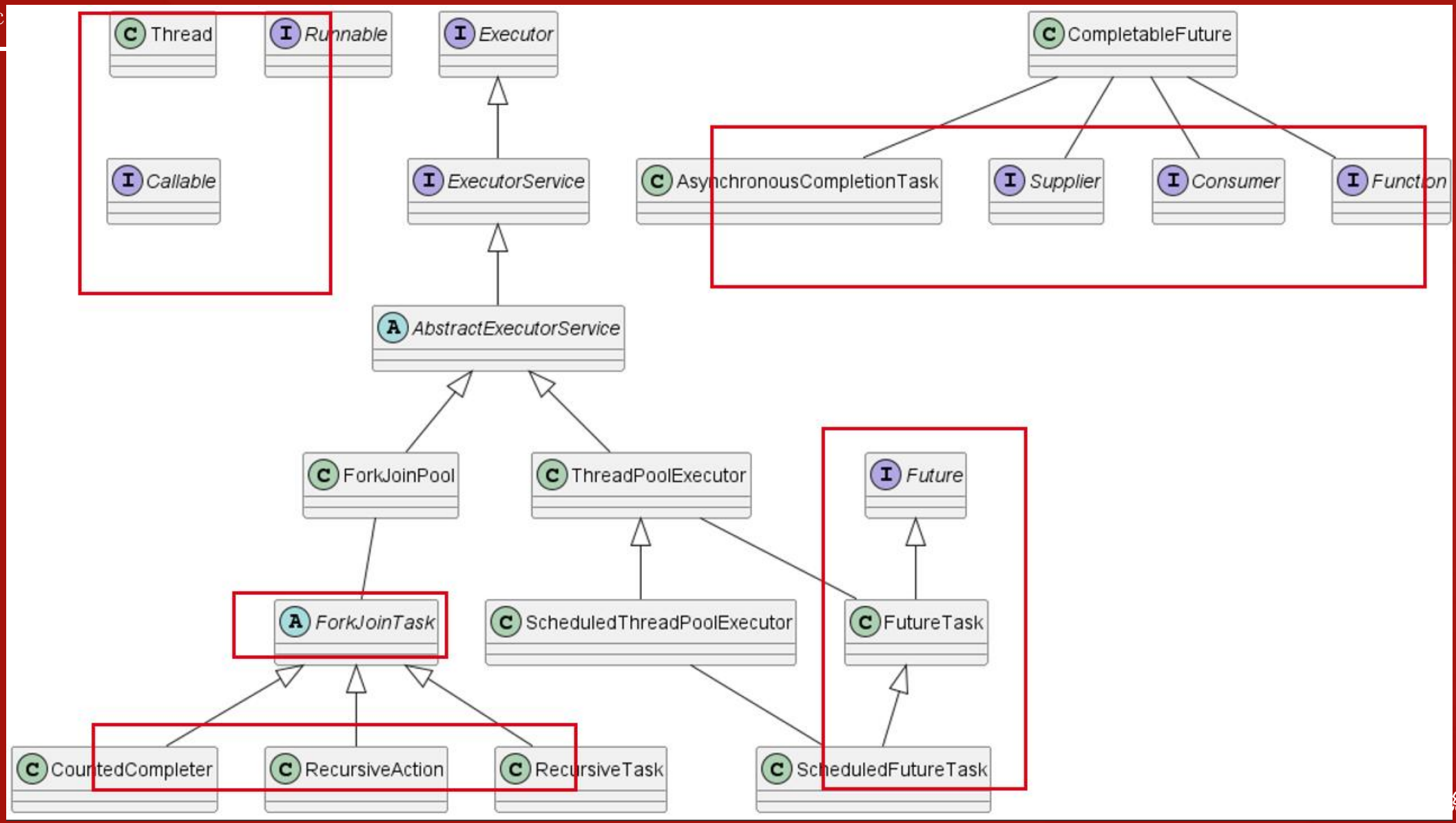
The content of the title

- 历史演进：1.1 -> 5 -> 7 -> 8 -> 9
  - 从任务并行到数据并行
  - 从OO到FP
  - 抽象层次越来越接近领域问题
- java.util.concurrent并发包
  - 理解难度:2>4>1>3
  - 使用难度:1>4>3>2
  - Today：结构化并行
- JEP 428: Structured Concurrency



任务以及任务执行，结构化并行

The c



并行流

The content of the title

- 基础模型
  - 起始流(Source)：Source和Spliterators
  - 转换(Transforming)：Filtering，Mapping，Debugging，Sorting，Deduplicating，Truncating，
  - 终止流(End)：Search，Collection，Reduction，Side-Effecting
- 处理大规模数据集
  - MapReduce，MapCollect

并行流

The content of the title

源	可分解性
ArrayList	极佳
LinkedList	差
IntStream.range	极佳
Stream.iterate	差
HashSet	好
TreeSet	好

- 使用场景
  - 源很好分割
  - 数据量很大
  - 无状态计算，无共享变量计算
  - 计算密集型

- Doug Lea: When to use parallel streams

并行流-代码演示

The content of the title

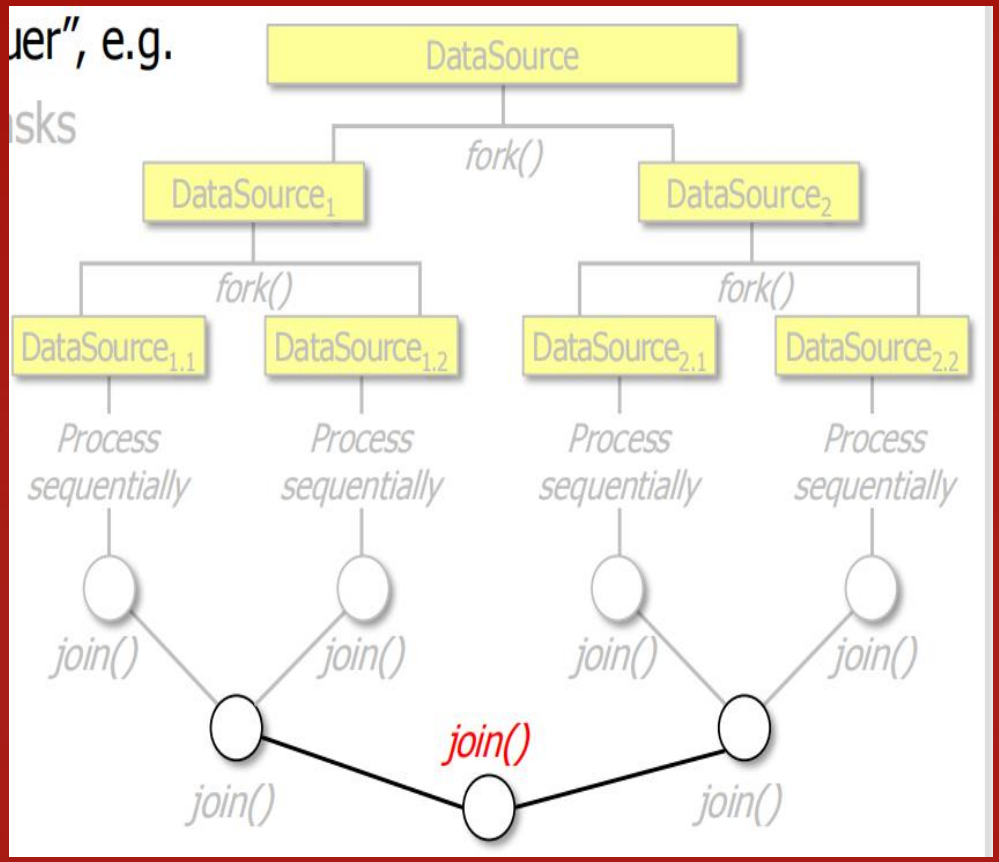
- demo1 ParallelStreamBenchmark
- demo2 WordCounterSpliterator

Fork Join框架

The content of the title

- ForkJoinPool
  - deque
  - work steal
- ForkJoinTask
  - fork
  - join
  - invoke

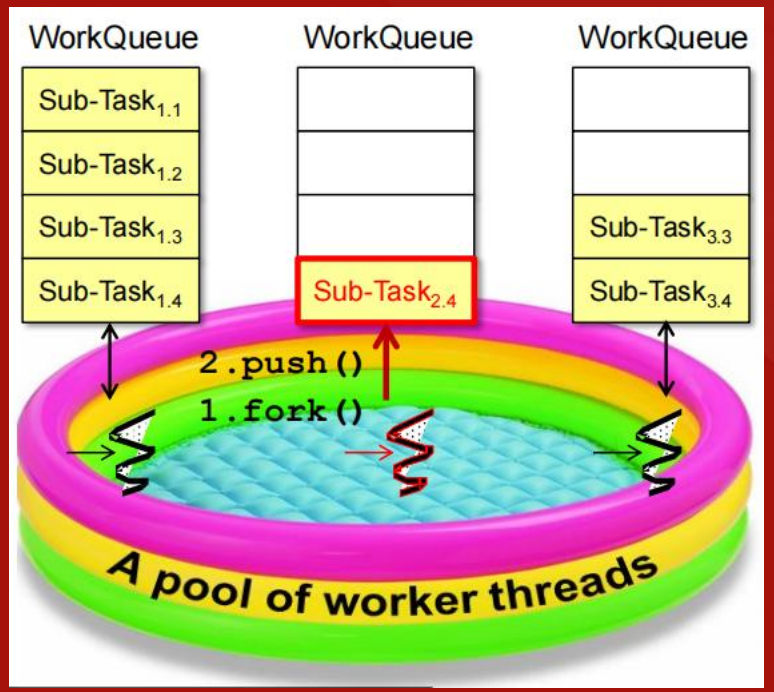
```
Result solve(Problem problem) {
  if (problem is small)
    directly solve problem
  else {
    split problem into independent parts
    fork new subtasks to solve each part
    join all subtasks
    compose result from subresults
  }
}
```

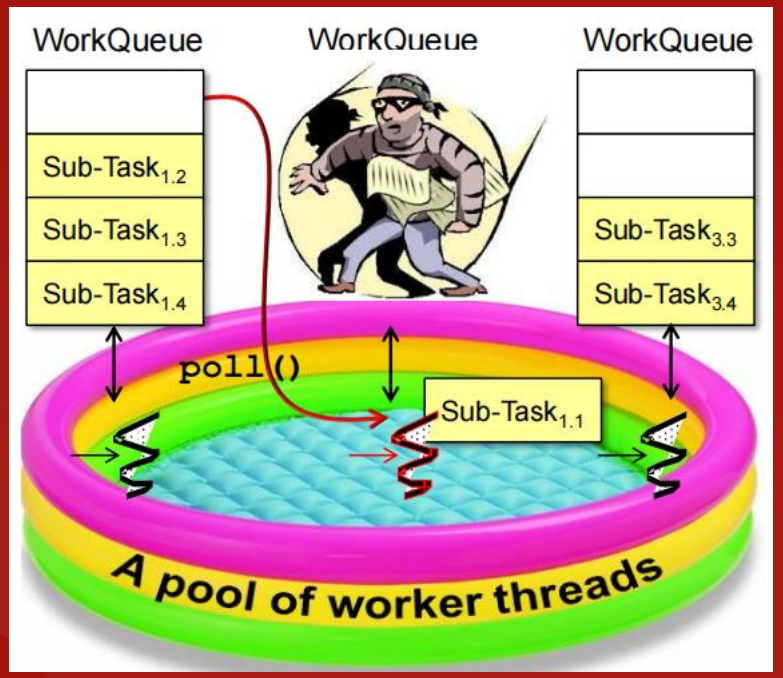


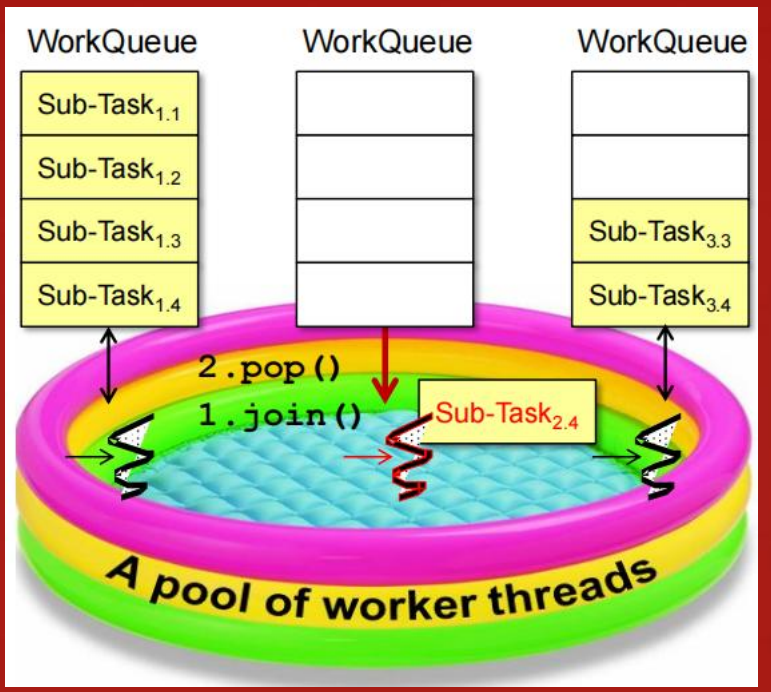
- 使用场景：计算密集型，分治，任务递归可以划分为子任务，如k-means聚类，归并/快速排序，矩阵相乘/LU分解，高斯积分，求和等。

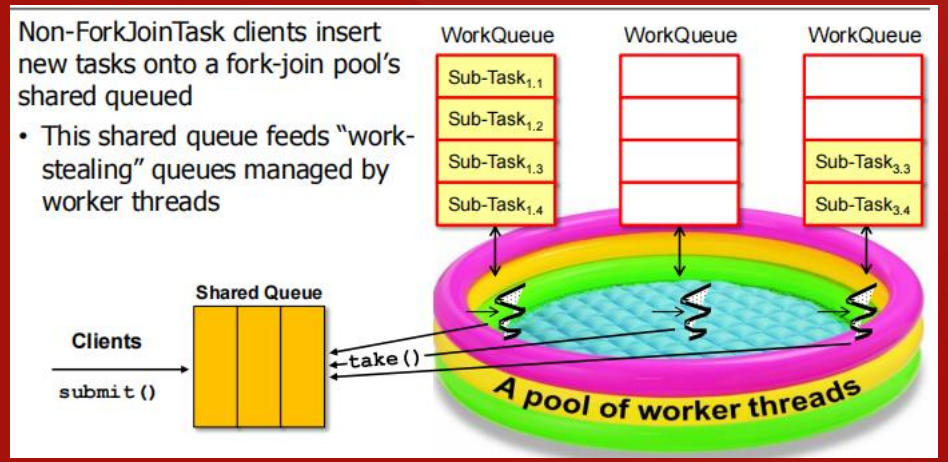
Fork Join框架

The content of the title









Fork Join框架-代码演示

The content of the title

- demo3 ForkJoinSumCalculator
- demo4 common.ForkJoinUtils
- demo5 common.BlockingTask

CompletableFututre

The content of the title

- 异步计算
  - 中断线程
  - Future
  - CompletableFututre
- 函数式异步计算阶段：CompletionStage
- 类似工具
  - ExecutorCompletionService （JDK1.5）
  - Guava：ListenableFuture（JDK1.6）
  - Cassandra Datastax Driver AsyncCqlSession

CompletableFututre

The content of the title

- 理解API
  - 计算执行：applyFunction，acceptConsumer，runRunnable
  - 触发计算：单阶段 thenXXX，两阶段 combineXXX/eitherXXX，多阶段 allOf/anyOf
  - 执行机制：default，async，自定义Executor
  - 计算结果：whenComplete，handle，join
  - 异常处理：exceptionally

- 使用场景
  - IO密集型，混合负载计算
  - 异步任务关联，依赖
  - 包装IO API

CompletableFututre-代码演示

The content of the title

- demo6 旅行社订票 CompletableFutureDemo

案例学习

The content of the title

- adaplearn-quiz的FutureCollector
- adaplearn-its的CompleteFuture组合
- common-utils

concurrency-practice杂谈

The content of the title

- 项目Github地址
- Java并发学习资料库，2020开始维护。

- 学习资料以及路径
  - 使用->知识完备->设计
  - 书籍
  - 课程
  - 论文

- 综合案例　SearchStreamGangTest


感谢各位的聆听
網易 NETEASE