高薪必备 -大纲

推荐学习路径（适合准备架构师或技术负责人岗位）

阶段	学习内容
第一阶段	锁机制、线程池、TPL、Channel、PLINQ、ValueTask
第二阶段	泛型、反射、IL、Expression、AOP、DI、IoC
第三阶段	ASP.NET Core、中间件、Filter、DI、管道
第四阶段	EF Core、数据库迁移、性能优化
第五阶段	分布式系统、CAP、Saga、事件溯源
第六阶段	设计模式、DDD、CQRS、EventBus、微服务治理
第七阶段	自定义组件、源生成、AOT 编译、性能调优
第八阶段	架构设计、系统拆分、限流、熔断、日志聚合

一、性能优化与底层原理（顶级工程师必备）

- `ValueTask` vs `Task` 的区别？什么时候使用？
- `Span<T>` / `Memory<T>` 的作用及适用场景
- `ref struct`、`stackalloc`、`fixed` 的使用和限制
- 如何避免内存泄漏（如事件订阅、闭包引用等）
- `unsafe` 编程与指针操作
- `GC.Collect()` 和 `GC.WaitForPendingFinalizers()` 的用途
- `WeakReference` 的使用与应用场景
- 对象池（Object Pool）设计与实现
- 内存屏障（MemoryBarrier）与 Volatile 读写

二、.NET 运行时机制与 CLR 原理

- .NET Core 与 .NET Framework 的区别
- GC 工作原理（代龄、压缩、标记清除算法）
- JIT 编译器工作流程
- AOT 编译 vs JIT 编译的区别
- AppDomain 的作用（虽然现在少用）
- Assembly 加载机制（Load, LoadFrom, LoadFile）

类型加载过程（ Type.Load, Type.GetType）
反射（ Reflection）的性能问题及优化方式
IL 中间语言基础（如 ldloc, stloc, callvirt）
C# 编译器如何将代码转换为 IL

三、异步编程模型（不仅仅是 async/await）

async/await 的底层原理（状态机、 IAsyncStateMachine） ConfigureAwait(false) 的作用和为什么推荐在库中使用
SynchronizationContext 和 TaskScheduler 的区别
ValueTaskSource 的封装与自定义 ValueTask
如何实现一个 “无 await” 的异步方法（如基于 IAsyncResult）
异步本地变量： AsyncLocal<T> 的原理与使用陷阱
如何避免死锁（如 task.Result 在 UI 上下文中的死锁）
ValueTask 是否一定比 Task 快？什么情况下不推荐？

四、泛型与反射进阶（高级开发必会）

泛型协变（ in/out）的作用和限制
泛型约束（ where T : class, new(), interface, struct 等）
泛型类型擦除机制（ CLR 是不是真正支持泛型？）
泛型接口与抽象类的设计模式
MethodInfo.MakeGenericMethod() 的使用
动态调用方法（反射性能优化技巧）
使用 Expression Tree 替代反射提高性能
System.Runtime.CompilerServices.Unsafe 的用途与风险 MethodImpl(MethodImplOptions.AggressiveInlining) 的作用
ConditionalWeakTable<TKey, TValue> 的用途（用于扩展对象元数据）

五、集合与数据结构优化

ConcurrentDictionary<TKey, TValue> 的实现原理
ConcurrentQueue<T> 与 Channel<T> 的区别
SortedSet<T> 、 SortedList<TKey, TValue>、 SortedDictionary<TKey, TValue> 的区别 ArrayPool<T> / BufferPool 的使用与性能优势
ReadOnlySpan<T> 和 ReadOnlyMemory<T> 的使用场景
自定义高性能队列（如环形缓冲区 RingBuffer）
实现一个无锁队列（ Lock-Free Queue）
数据分页与懒加载（ Paging + Lazy<T>）

六、模块化架构与设计模式（大型系统必问）

领域驱动设计（ DDD）核心概念（聚合根、值对象、仓储）
CQRS 模式与 Event Sourcing 的区别
Mediator 模式在项目中的实际应用
工厂模式 vs DI 容器
策略模式与命令模式的结合使用
单例模式的多线程安全实现（ Double-check Locking）
装饰器模式与 AOP 实践（如日志、缓存装饰）
观察者模式 vs 发布 -订阅模式 vs Prism 的 IEventAggregator
构建可插拔系统（ Prism、 MEF、Autofac Modules）

七、依赖注入与 IOC 容器（现代框架的核心）

ASP.NET Core 默认容器（ Microsoft.Extensions.DependencyInjection）的生命周期（ Transient
、 Scoped、Singleton）
如何实现一个简易 IOC 容器（构造函数注入、属性注入）
服务定位器（ Service Locator）模式的弊端
什么是 Ambient Context 模式？
依赖注入中循环依赖的识别与处理
DI 与 AOP 的结合（如 Autofac + Interceptor）
使用 IServiceProvider 与 IServiceScopeFactory 的区别
在静态类中如何获取注入的服务（不推荐，但有办法）

八、跨平台与底层交互（适用于 .NET Core）

P/Invoke 的使用与注意事项（ DllImport）
如何调用原生 C/C++ 接口
COM Interop（Windows 平台）
使用 NativeAOT 编译 .NET 应用
如何编写跨平台的 C# 代码
RuntimeInformation.IsOSPlatform(OSPlatform.Windows) 的用途 System.Runtime.InteropServices 的常用类
如何判断当前运行环境（ x86/x64/ARM）

九、网络编程与通信（高并发后端开发必备）

TCP/UDP Socket 编程与异步模型（ SocketAsyncEventArgs） HttpClient 的正确使用方式（单例复用、取消令牌）
HttpMessageHandler 的重用与连接池管理
gRPC 的基本原理与 .NET 集成
SignalR 的底层通信机制（ WebSocket vs Server-Sent Events）
使用 Kestrel 实现高性能 Web 服务器
如何实现一个简单的 RPC 框架
使用 MemoryPool<byte> 处理大块数据传输

十、性能监控与诊断工具（排查生产问题）

使用 PerfView 分析线程饥饿、 GC 性能瓶颈
使用 dotTrace / dotMemory 进行 CPU 和内存分析
使用 VisualVM / ANTS Memory Profiler（Java 互操作时） dotnet-counters、 dotnet-trace 的使用
使用 DiagnosticSource 实现轻量级日志追踪
如何查看当前线程池状态（ ThreadPool.GetAvailableThreads）
如何检测死锁（线程阻塞、同步上下文丢失）
使用 MiniDump 文件进行崩溃分析
使用 ETW（ Event Tracing for Windows）做性能追踪

十一、单元测试与自动化测试（工程能力体现）

单元测试与集成测试的区别
Mock 框架（ Moq、NSubstitute）的使用
如何测试异步代码（包括异常抛出）
使用 xUnit / NUnit / MSTest 的差异
测试静态类、私有方法、密封类的技巧
测试依赖外部资源（数据库、 API）的方法
使用 TestServer 做集成测试（ ASP.NET Core）
如何衡量测试覆盖率（ coverlet、Visual Studio、JetBrains Rider）

十二、中间件与管道模型（ ASP.NET Core 核心机制）

ASP.NET Core 的请求管道模型（ Middleware Pipeline）
如何自定义 Middleware
Use, Run, Map 的区别
IApplicationBuilder、 IHostApplicationLifetime 的使用
请求委托（ RequestDelegate）的链式构建
如何实现身份验证中间件
使用 PipelineBuilder 手动构建中间件管道
如何在中间件中访问 DI 容器
使用 HttpContext.Features 获取底层特性

十三、配置系统与依赖管理（微服务时代重点）

IConfiguration 的来源（ appsettings.json、Environment Variables、Command Line） IOptions<T> 的绑定与使用（包括 Named Options）
IConfigureOptions<T> 和 IPostConfigureOptions<T> 的区别
使用 AddOptions().PostConfigure(...) 的典型场景
自定义 ConfigurationProvider 的实现
使用 Secret Manager 存储敏感信息
热更新配置（ IOptionsMonitor<T> vs IOptionsSnapshot<T>）

十四、序列化与反序列化（ JSON、XML、 Binary）

System.Text.Json vs Newtonsoft.Json 的区别 JsonSerializer.Deserialize<T>(...) 的性能优化技巧
如何自定义 JSON 转换器（ JsonConverter） DataContractSerializer vs XmlSerializer 的选择
BinaryFormatter 的安全隐患（已废弃）
使用 MemoryPack / MessagePack 提升序列化性能
使用 System.Text.Json 支持源生成（ Source Generation）
使用 Utf8JsonReader / JsonDocument 做只读解析

十五、 LINQ 与表达式树（底层机制）

IEnumerable<T> vs IQueryable<T> 的区别
LINQ 查询的延迟执行机制
Select / Where / GroupBy 的底层行为
表达式树（ Expression Tree）与动态查询构建
如何编译 Expression 为 Delegate
使用 ExpressionVisitor 修改表达式逻辑
EF Core 中的表达式解析与 SQL 转换
使用 AsParallel() 实现 PLINQ
使用 Partitioner 自定义并行策略

十六、跨平台部署与发布（云原生工程师必备）

dotnet publish 的参数详解（ --self-contained, --runtime）
如何发布 AOT 编译的 .NET 应用
使用 Single File 打包的优势与限制
Trimming 与 PublishTrimmed 的作用
如何减小 Docker 镜像体积
ReadyToRun 编译与性能提升
使用 deps.json 控制程序集加载
使用 AssemblyLoadContext 实现热更新
如何在 Linux 上调试 .NET Core 程序

十七、分布式系统与微服务架构（架构师级别）

CAP 理论与 BASE 琟能力
微服务通信方式（ REST, gRPC, Message Queue）
服务发现（ Consul, etcd, Kubernetes）
分布式事务（ 2PC, Saga, TCC）
最终一致性与幂等性设计
使用 Dapper / Entity Framework Core 实现 Repository 模式事件溯源（ Event Sourcing）与快照（ Snapshotting）使用 CAP 实现最终一致性的消息队列消费
使用 MassTransit / Rebus / Cap 实现分布式任务调度
使用 Serilog 实现结构化日志记录

十八、 EF Core 与数据库优化（ ORM 深入）

EF Core 的查询生命周期（ Query Pipeline）
如何启用 DetailedErrors 和 SensitiveDataLogging
ChangeTracker 的作用与关闭时机
显式加载（ Include, ThenInclude, Load）
使用 AsNoTracking() 提升查询性能
自定义 ValueConverter 与 ValueComparer
使用 FromSqlRaw 实现复杂查询
使用 Interceptors 监控 SQL 日志
使用 RelationalTypeMapping 自定义字段映射

十九、 C# 语言新特性掌握情况（ C# 9/10/11/12）

record 与 class 的区别
init 属性设置器的作用
required 成员（ C# 11）
global using、 file-scoped namespace（C# 10）
with 表达式（ record 特有）
nint / Half / Primary Constructor（C# 12）
Lambda Return Type、 params any type（C# 12）
raw string literal（ """）的使用场景
使用 source generator 实现零运行时开销代码生成
使用 partial methods 实现编译期钩子（ hook）

二十、设计能力与系统架构能力（架构师级别）

如何设计一个高并发系统（如订单中心）
如何设计一个通用权限模型（ RBAC vs ABAC）
如何设计一个可插拔的日志系统
如何设计一个统一异常处理中间件
如何设计一个可扩展的 API 网关
如何设计一个高性能缓存服务（本地缓存 vs Redis）
如何设计一个限流服务（滑动窗口 vs 令牌桶）
如何设计一个分布式 ID 生成器（ Snowflake vs IdGenerator）
如何设计一个通用后台任务队列（ Channel<T> vs BackgroundService）
如何设计一个插件系统（ AppDomain / AssemblyLoadContext）

二十一、开源生态与工具链（加分项）

使用 MediatR 实现 CQRS 模式
使用 AutoMapper 实现 DTO 到 Entity 的转换
使用 FluentValidation 实现强类型校验
使用 Serilog 实现结构化日志
使用 Polly 实现熔断、重试、降级

- 使用 `Refit` 实现声明式 REST 客户端
- 使用 `MassTransit` 或 `CAP` 实现消息队列消费
- 使用 `ImageSharp` / `SkiaSharp` 实现图像处理
- 使用 `ImageSharp.Web` 实现图片 CDN
- 使用 `IdentityServer` 实现 OAuth2 认证授权

二十二、常见开源项目理解（加分项）

开源项目	推荐掌握
Orleans	分布式 Actor 模型
Akka.NET	类似 Scala 的 Actor 模型
CAP	分布式事件总线 + 消息持久化
Entity Framework Core	ORM 深度定制
Serilog	结构化日志系统
Polly	弹性策略（重试、熔断）
MediatR	CQRS 模式
FluentValidation	声明式校验
Xunit / Moq / FluentAssertions	单元测试体系
SignalR	实时通信框架

版本号 #1
由 叶子 创建于 4 六月 2025 22:22:20
由 叶子 更新于 4 六月 2025 22:28:45