### 🔍 What is JIT and its Overhead?

**JIT (Just-In-Time) compilation** happens at runtime in .NET. It converts MSIL (Microsoft Intermediate Language) code into native machine code.

### 🔶 Overhead at Runtime:

When a method is called for the **first time**, the JIT compiler translates it into native code. This adds **delay (overhead)** during execution, especially for large applications with many methods.

---

### ✅ How .NET Reduces JIT Overhead

**1. NGen (Native Image Generator) — Legacy (Full .NET Framework)**

- Pre-compiles IL code into native machine code before runtime.
- Reduces JIT overhead by avoiding runtime compilation.
- ❌ Drawback: Less flexible, platform-specific.

**2. ReadyToRun (R2R) — .NET Core & .NET 5+**

- Similar to NGen but modern.
- Publish-time AOT (Ahead-Of-Time) compilation to native code.
- Speeds up startup by reducing JIT usage.
- ✅ Used with: dotnet publish -c Release -r win-x64 --self-contained

**3. Tiered Compilation**

- Introduced in .NET Core and .NET 5+.
- Starts with quick JIT for faster startup (Tier 0).
- Then upgrades “hot” methods to optimized native code (Tier 1) during runtime.
- ✅ Combines fast startup with optimized performance.

**4. Profile-Guided Optimization (PGO)**

- Collects data on how code runs (e.g., which methods are used most).
- Uses that data to optimize future compilations.
- 📌 .NET 8+ improves this further with **Dynamic PGO**.

**5. AOT Compilation (NativeAOT) — .NET 7+**

- Full ahead-of-time compilation; no JIT at all.
- Very fast startup and low memory usage.
- 🔧 Ideal for small tools or microservices.

**6. Method Inlining & Caching**

- Frequently used methods may be inlined to avoid calls.
- JIT stores compiled code in memory for reuse.

---

📝 **Summary Table**

| Technique | Purpose | Used In | Benefit |
|---|---|---|---|
| NGen | Pre-compile IL to native | .NET Framework | Avoid JIT at runtime |
| ReadyToRun (R2R) | Publish-time native code | .NET Core+ | Faster startup |
| Tiered Compilation | Quick startup + later optimize | .NET Core+ | Balance speed + performance |
| PGO | Optimize based on usage | .NET 6+ | Smarter code optimization |
| NativeAOT | Full native, no JIT | .NET 7+ | Lowest memory, fastest start |