What is JIT Overhead?

When using a **JIT compiler** (like in Java's JVM, .NET CLR, or JavaScript engines like V8), parts of the code are compiled from intermediate representations (bytecode, IL, etc.) into native machine code during program execution. This introduces overhead:

- **Initial delay**: The first time a method runs, it might be compiled instead of executed directly.
- **Memory use**: Storing compiled native code in memory.
- **Repeated compilation**: If not cached or reused, frequently run code might be recompiled.
- **Profiling overhead**: Some JITs profile code first (how often a method runs, branch probabilities, etc.) before compiling, which takes time.

Strategies to Reduce JIT Overhead

1. Tiered Compilation

- Combines **interpretation** and **JIT**:
    - Code is first interpreted (cheap, fast startup).
    - If a method becomes "hot" (frequently executed), it gets optimized by a higher-tier JIT.
- Example: Java HotSpot uses C1 (fast JIT) and C2 (optimized JIT).

2. Ahead-of-Time Compilation (AOT)

- Some platforms allow compiling bytecode to native machine code **before** runtime:
    - Reduces startup time and memory use.
    - Examples:
        - .NET Native, Mono AOT, Java GraalVM Native Image.

3. Code Caching

- Caches compiled code across runs (e.g., V8's code cache for JavaScript).
- Avoids re-JITting the same code on next execution.

4. Profile-Guided Optimization (PGO)

- Use runtime profiling from previous runs to guide optimizations.
- Helps the JIT or AOT compiler produce more efficient code.

5. Limiting JIT scope

- Some JIT compilers don’t compile all code:
    - Only hot paths get compiled.
    - Cold paths may stay interpreted or minimally optimized.

6. Pre-JIT or NGEN (Native Image Generation)

- .NET’s NGEN compiles IL to native code ahead of time.
- Reduces JIT overhead but less flexible than pure JIT (e.g., less adaptive).

7. Using lightweight or baseline JITs

- Some runtimes use simpler, faster JITs to reduce overhead.
- These do less optimization but allow quicker compilation (important for short-lived applications).