Self-Study Report: Reducing Runtime Overhead in C# (JIT Compilation)

عند تنفيذ الكود، بيتم تحويله من كود وسيط IL إلى كود مخصص للجهاز ((Native Codeباستخدام الـ) JIT Compiler الـ مترجم عند التنفيذ وده بيحسن الأداء لأنه بيحول الكود لِكود ينفذ مباشرة على الـ .CPUلكن في نفس الوقت، العملية دي بتسبب بعض الـ) Overheadحمل زمني) لأن الكود بيتترجم أثناء التنفيذ، وبالتالي بياخد وقت.
الهدف في التقرير ده هو شرح الأسباب اللي بتخلي الـ JITيسبب مشاكل في الأداء، وإزاي ممكن نقلل من الـ Overheadده.

1. أولا إيه هو الـ JIT Compiler؟
IL Code (Intermediate Language) ده الكود اللي بيطلع من الكود اللي كتبته في #Cبعد ما تتكومبيل.
JIT Compilation هو الـ "مترجم" اللي بيحول الـ IL Codeالكود مخصص للجهاز أثناء التنفيذ، عشان يشتغل بشكل أسرع.
المشاكل اللي بيعملها الـ JIT في الأداء

البداية البطيئة: أول ما التطبيق يبدأ، بيحتاج الـ JITيترجم الكود لـ Native Code، وده بياخد وقت.

استهلاك الذاكرة: الـ Native Code اللي بيتترجم بيحتاج مساحة إضافية في الذاكرة.

تكرار الترجمة: ممكن الـ JITيترجم نفس الكود أكتر من مرة، خصوصًا لو الكود ده شغال على منصات مختلفة أو في بيئات متعددة.

إزاي نقلل من الحمل الزمنيOverhead؟
الترجمة المبدئية AOT Compilation
إيه هو AOT؟ :بدل ما الكود يتترجم أثناء التنفيذ، ممكن نعمل ترجمة مسبقة (قبل ما نفتح التطبيق)، وده بيقلل وقت البدء وبيقلل استهلاك الذاكرة.

إزاي نستخدمه؟
في NET Core و NET 6 فما فوق، ممكن تفعّل AOT باستخدام إعدادات المشروع.