عنوان مقاله: بررسی بهبود کارایی .NET 8 نسبت به نسخه های قبلی

نویسنده مقاله: تیم فنی نیک‌آموز

تاریخ انتشار: ۲۵ آبان ۱۴۰۲

منبع: https://nikamooz.com/net-8-performance-improvements-compared-to-previous-versions/

بهبود کارایی .NET 8 با نقاط پیشرفت مختلفی همراه خواهد بود؛ به‌طوری که جامعه برنامه نویسان .NET 8 در روزشماری برای عرضه آن بودند تا درنهایت، این اتفاق در ۲۳ آبان ۱۴۰۲ محقق شد. در **مقاله قبلی**، به بررسی ۱۱ ویژگی جدید .NET 8 پرداختیم و در این مطلب مکمل، قصد داریم نقاط پیشرفت دات نت ۸ را از نقطه نظر بهبود کارایی (Performance) مورد بررسی قرار دهیم.

.NET 8 از نظر کارایی چه پیشرفتی دارد؟

اگر شما در جامعه برنامه نویسان دات نت مشغول به کار هستید یا صرفاً به برنامه‌نویسی و حوزه تکنولوژی علاقه‌مند باشید، به احتمال زیاد از انتشار نسخه جدید آن اطلاع دارید. دات نت ۸ به گونه‌ای بهبود داده شده است که نویدبخش نوآوری و سازندگی هرچه بیشتر خواهد بود. در ادامه، به بررسی بهبود کارایی .NET 8 پرداخته می‌شود.

Tiering و Dynamic PGO

دو ویژگی جدید Tiering و Dynamic PGO ، مواردی هستند که در راستای بهبود کارایی NET 8. ارائه شده‌اند. به واسطه این Feature ها ، فرصتی فراهم می‌شود که کارایی کدهای مدیریت‌شده تا سقف ۲۰٪ بهبود پیدا کند. ویژگی Tiering ، کدها را به چند نمایشِ (Representation | بازنمایی) کد ماشین کامپایل می‌کند و هر یک از آن‌ها، برای سناریوهای اجرایی گوناگونی بهینه‌سازی خواهند شد. از سوی دیگر، Dynamic PGO در NET 8. ، از **Profiling** داده‌ها به‌منظور بهینه‌سازی کد در زمان اجرا بهره می‌برد.

به قطعه کد زیر توجه کنید. این تکه کد در چندین بازنمایی کد ماشین ، کامپایل خواهد شد که هرکدام برای سناریوی Runtime متفاوتی بهینه می‌شوند.

```
// This code will be compiled to multiple machine code representations,
// each optimized for a different runtime scenario.
public int Fibonacci(int n)
{
    if (n <= 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

در این مثال، از Data Profiling برای بهینه‌سازی داده‌ها در زمان اجرا بهره‌وری خواهد شد.

```
// This code will use profiling data to optimize code at runtime.
public int Factorial(int n)
{
    if (n <= 1)
    {
        return 1;
    }

    return n * Factorial(n - 1);
}
```

تکنیک بردارسازی (Vectorization)

Vectorization رویکردی است که به پردازشگر (CPU) اجازه می‌دهد چند عنصر یک آرایه به‌صورت موازی (Parallel) پردازش شوند. یکی بهبود کارایی NET 8. مربوط بردارسازی است که در آن، از نوع‌های جدید Vector و Intrinsics پشتیبانی خواهد شد. به منظور درک بهتر این ویژگی دات نت ۸ ، به قطعه کد زیر توجه کنید. در این ورژن، این کد بردارسازی خواهد شد تا امکان پردازش چند عنصر آرایه به‌صورت موازی فراهم و به‌دنبال آن، بهبود کارایی NET 8. حاصل شود.

```
// This code will be vectorized to process multiple elements of the array
in parallel.
int[] sum = new int[100000];
for (int i = 0; i < sum.Length; i++)
{
    sum[i] = i + 1;
}
```

در ادامه نیز از intrinsics برای این موضوع استفاده شده است.

```
// This code will use intrinsics to process multiple elements of the array
in parallel.
int[] sum2 = new int[100000];
for (int i = 0; i < sum2.Length; i++)
{
    sum2[i] = Vector.Add(new Vector(i), new Vector(1));
}
```

انشعاب گذاری (Branching)

Branching یک عملیات رایج در کد محسوب می‌شود اما اجرای آن برای پردازنده، مقرون‌به‌صرفه نیست. در این راستا، این نسخه در زمینه Branching، به‌روزرسانی‌هایی به همراه دارد که منجر به بهبود کارایی NET 8. خواهند شد. به‌عنوان مثال، در NET 8. ، دستورات Move شرطی و الگوریتم‌های پیش‌بینی (Prediction Algorithms) جدیدی پشتیبانی خواهند شد.
به کد زیر توجه کنید. در این شرایط، می‌توان با به‌کارگیری Conditional Move، از انشعاب‌گذاری یا همان Branching اجتناب کرد.

```
// This code will use conditional move instructions to avoid branching.
int x = 10;
if (x > 5)
{
    x = x + 1;
}
else
{
    x = x - 1;
}
```

در دات نت ۸ ، برای قطعه کد زیر از الگوریتم‌های پیش‌بینی استفاده می‌شود تا پیش‌بینی شود که کدام انشعاب مورد انتخاب قرار می‌گیرد؛ زیرا با دانستن این موضوع، دستورات ضروری مربوطه از قبل واکشی خواهند شد.

```
// This code will use prediction algorithms to predict which branch will
be taken and prefetch the necessary instructions.
int y = 10;
if (y > 5)
{
    y = y + 1;
}
else
{
    y = y - 1;
}
```

Bounds Checking

Bounds Checking یک ویژگی ایمنی است که مانع از دسترسی برنامه‌نویسان به حافظه خارج از محدوده یک آرایه می‌شود. با این وجود، بررسی Bound ها می‌تواند هزینه‌بر باشد. نسخه جدید دات نت ، به شما اجازه می‌دهد تا Bound Check ها را برای عملیات ایمن اعمال نکنید و این موضوع، بهبود کارایی NET 8. را به ارمغان می‌آورد. به‌عنوان مثال، می‌دانیم در قطعه کد زیر، عملیات امن است؛ بنابراین می‌توان بررسی‌های مربوط به Bound ها را از قلم انداخت.

```
// This code will elide the bounds check because the operation is known to
be safe.
int[] array = new int[10];
int value = array[5];

// This code will use a new bounds-checking algorithm that is more
efficient for known-safe operations.
int[] array2 = new int[10];
int value2 = array2[5
```

Constant Folding

Constant Folding یک روش کاربردی است که با جایگزین کردن ثابت‌های (Constants) زمان کامپایل با مقدار هر یک، کد را بهینه‌سازی می‌کند. اگر بخواهیم بهبود کارایی .NET 8 را از این نقطه نظر بررسی کنیم، مشخص می‌شود که این نسخه، بهبودهای مختلفی در زمینه Constant Folding ارائه کرده است. در حقیقت، به‌واسطه قابلیت پشتیبانی از Folding برای عبارات پیچیده، کارایی دات نت ۸ در مقایسه با ورژن‌های پیشین، مطلوب‌تر شده است. برای درک بهتر، به قطعه کد زیر توجه شود.

```
// This code will fold the constant expression and replace it with its value.
int sum = 1 + 2 + 3;

// This code will fold the more complex constant expression and
 replace it with its value.
int sum2 = (1 + 2) * 3
```

وجود Non-GC Heap

به‌طور کلی، در Runtime دات نت از Garbage Collector به منظور مدیریت حافظه استفاده می‌شود. با این وجود، این موضوع می‌تواند هزینه‌بر باشد. ارائه Heap ای که به شکل GC نباشد، یک بهبود کارایی .NET 8 قلمداد می‌شود؛ زیرا با کمک Non-GC Heap در دات نت ۸ ، دیگر نیازی نیست که حافظه مورد تخصیص‌تان توسط Garbage Collector جمع‌آوری شود. به‌عنوان مثال، در کد زیر، شی روی Non-GC heap تخصیص داده خواهد شد.

```
// This code will allocate the object on the non-GC heap.
byte[] bytes = new byte[1024 * 1024];
using (UnmanagedMemory.Pin(bytes))
{
    // Use the bytes array.
}
```

Zeroing

نقاط پیشرفت دیگری در نسخه جدید دات نت قرار دارند. یک نوع بهبود کارایی NET 8. ، پشتیبانی آن از Zeroing حافظه به‌صورت موازی و همچنین، Zeroing حافظه بدون نیاز به ایجاد شی جدید است. در ادامه، دو نمونه از کدهایی را قرار داده‌ایم که به ترتیب، عمل Zeroing حافظه به‌صورت موازی و همچنین Zeroing بدون نیاز به تخصیص شی جدید را به‌همراه خواهند داشت.

```
// This code will zero the memory in parallel.
byte[] bytes = new byte[1024 * 1024];
Parallel.For(0, bytes.Length, (i) => bytes[i] = 0);

// This code will zero the memory without allocating a new object.
byte[] bytes2 = new byte[1024 * 1024];
using (UnmanagedMemory.Pin(bytes2))
{
    Unsafe.InitBlock(bytes2.Pointer, 0, bytes2.Length);
```

بهبود در نوع های مقداری

NET 8. با بهبودات مختلفی در زمینه نوع های مقداری (Value Types) همراه است. در حقیقت، دات نت ۷ ، امکان پشتیبانی از تایپ‌های مقداری بزرگ‌تر را دارد و همچنین Boxing/Unboxing نوع‌های مقداری را به‌صورت کارآمدتری انجام خواهد داد.

```
// This code will use a larger value type.
struct Point
{
    public int X;
    public int Y;
}

// This code will box and unbox value types more efficiently.
Point point = new Point { X = 10, Y = 20 };
int x = point.X;
```

تبدیل نوع (Casting)

بهبود کارایی NET 8. از نظر Casting با نقاط قوتی همراه خواهد بود؛ زیرا در این نسخه از دات نت ، پشتیبانی از Casting برای نوع‌های مقداری و ارجاعی (Reference Types) به‌صورت کارآمدتر انجام می‌شود.

```
// This code will cast between value types and reference types more
efficiently.
object obj = 10;
int value = (int)obj;
```

بهینه سازی روزنه ای (Peephole Optimizations)

Peephole Optimizations بهینه‌سازی‌های کوچکی محسوب می‌شوند که برروی کد کامپایل شده قابل اجرا هستند. می‌توان وجود رویکردهای Peephole Optimizations جدید را به‌عنوان یک بهبود کارایی NET 8. در نظر گرفت و از آن در پروژه‌های مختلف بهره برد. مثلاً در کد زیر، بهینه‌ساز روزنه‌ای، دو دستور را به یک Instruction ترکیب می‌کند.

```
// This peephole optimization will combine two instructions into one.
int x = 10;
int y = 20;
int sum = x + y;
```

علاوه‌بر انواع بهبود کارایی NET 8. که تا این بخش به آن‌ها پرداختیم، نقاط مثبت دیگری نیز در دات نت ۸ ارائه شده است. این موارد در مقایسه با گزینه‌های پیشین، دسته‌بندی کلی‌تری از بهبود کارایی NET 8. محسوب می‌شوند.

UTF8

.NET 8 با تعدادی بهبود Performance در زمینه UTF8 همراه است. به‌عنوان مثال، در دات نت ۸ امکان رمزگشایی (Decoding) رشته‌های UTF8 به‌صورت موازی وجود دارد و از رمزگذاری این نوع رشته‌های بدون لزوم به تخصیص یک شی جدید نیز پشتیبانی خواهد شد.

ASCII

بهبود کارایی .NET 8 به موارد مذکور محدود نمی‌شود؛ به‌طوری که در این نسخه، قابلیت رمزگذاری و رمزگشایی رشته‌های اسکی (ASCII) به‌صورت کارآمد فراهم شده است.

Base64

در نسخه ۸ از دات نت ، پشتیبانی از Encoding و Decoding برای رشته‌های Base64 به‌صورت بهینه و مناسب ارائه شده است.

رمزگذاری و رمزگشایی کارآمد رشته های Hex

مشابه موارد پیشین، یک اقدام دیگر برای بهبود کارایی .NET 8 ، افزودن امکان رمزگذاری و رمزگشایی برای رشته‌های Hex با حفظ بهینگی است.

بهبود فرمت دهی رشته ها

یکی از مزیت‌های جالب دات نت ۸ از نقطه نظر کارایی ، پشتیبانی از فرمت‌دهی رشته‌ها به‌صورت موازی و همچنین عدم نیاز به تخصیص شی جدید برای Formatting رشته‌ها است.

نوع های Spans و SearchValues

در این نسخه، یک نوع (Type) جدید تحت عنوان Spans قرار داده شده است که به‌واسطه آن می‌توان به شکل کارآمدتری با حافظه کار کرد. این بهبود کارایی دات نت ، باعث خواهد شد تا کپی کردن Spans بهینه‌تر انجام و از مقایسه Spans به نحو احسن پشتیبانی شود. علاوه‌بر این، با کمک نوع SearchValues ، جستجوی مقادیر در کالکشن‌ها (Collections) تسهیل می‌یابد.

Regex

بهبود کارایی Regex در دات نت ۸ ، مواردی همچون کامپایل کردن رگولار اکسپرشن‌ها (Regular Expressions) به‌صورت کارآمد و پشتیبانی از اجرای بهینه آن‌ها را در برمی‌گیرد.

بهبود محاسبه Hashing Code ها

بهبود کارایی .NET 8 ، مزایای خاصی در زمینه Hashing را به همراه دارد. در حقیقت، در دات نت ۸ ، این امکان فراهم شده است که کدهای Hash به شکل کارآمد محاسبه شوند و مقایسه Hash code ها همراه با بهینگی پشتیبانی شود.

بهبود مقداردهی اولیه (Initialization)

در دات نت ۸ ، پشتیبانی از مقداردهی اولیه (Initialization) اشیا و همچنین، آرایه‌ها به‌صورت کارآمد وجود دارد.

ارائه Analyzer های جدید

شما می‌توانید با کمک Analyzer های جدید ارائه شده در NET 8. ، به تشخیص و رفع مشکلات مربوط به کارایی کدهای خود بپردازید.

بهبود کارایی NET 8. در یک نگاه

در این مطلب به انواع نقاط بهبود کارایی NET 8. پرداخته شد تا شما بتوانید با این ورژن آشنا شوید. به‌عنوان جدیدترین نسخه LTS، دات نت ۸ با مزیت‌های گوناگونی، از جمله کارایی، پایداری و بهبودات امنیتی، در ۱۴ نوامبر سال ۲۰۲۳ عرضه شد. شرکت مایکروسافت این نسخه از NET. را نویدبخش افزایش اثرگذاری و سرعت نوآوری به حساب می‌آورد. شایان ذکر است که در مقاله‌های آتی قصد داریم نحوه نصب NET 8. را به‌صورت مرحله‌به‌مرحله آموزش دهیم. به این ترتیب، این فرصت فراهم می‌شود تا شما نیز در مسیر یادگیری و کار با دات نت قرار بگیرید و آینده خود را به‌عنوان عضوی در جامعه برنامه‌نویسان رقم بزنید.