riganti

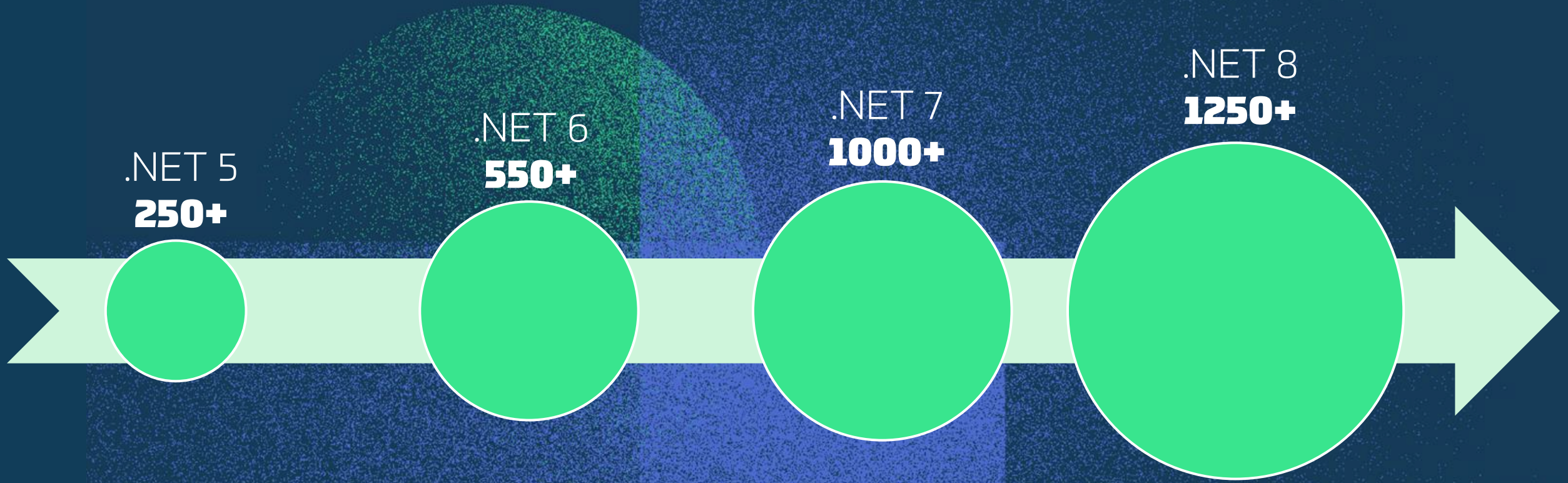
Allocation-free parsery pomocí Span<T> a Memory<T>

Tomáš Herceg
CEO @ RIGANTI
Microsoft MVP

riganti

Nové verze .NETu jsou rychlejší

- Stovky pull requestů zaměřených na zlepšení performance



riganti

Proč?

- Nové typy `Span<T>` a `Memory<T>`
  - Umožňují snížit množství alokované paměti
  - Méně kopírování paměti
  - Menší nápor na GC
- Vektorizace
  - Využívání SIMD instrukcí pro „hromadné“ operace
- … a mnoho dalšího

Co je Span<T>

- Reprezentace souvislého bloku paměti

```
public readonly ref struct Span<T>
{
   private readonly ref T _pointer;
   private readonly int _length;
   ...
}
```

- Může ukazovat na
  - Pole nebo jeho část
  - Pole alokované na stacku
  - Unmanaged paměť
- Read-only varianta `ReadOnlySpan<T>`

riganti

DEMO

Span<T>

riganti

Omezení Span<T>

- Pointer může ukazovat doprostřed pole
- Co když se spustí GC a musí objekt přesunout?

- Je to ref struct
  - Nemůže existovat na heapu, je vždy na zásobníku
  - Nemůže být jako prvek pole
  - Nemůže to být field (mimo field v jiném ref structu)
  - Nepodporuje boxing
  - Nevrátíte ji z asynchronní funkce jako Task<Span…>
  - Nepodporuje closure u lambda funkcí
  - Nelze použít v asynchronních funkcích nebo iterátorech

riganti

Memory<T>

- Span<T> může existovat jen na stacku
  - Při přesouvání objektů GC se pointery opraví
- Memory<T> tento problém řeší
  - Odkazuje na objekt, který danou oblast paměti obsahuje

```
public readonly struct Memory<T> {
   private readonly object _object;
   private readonly int _index;
   private readonly int _length;
   ...
}
```

riganti

DEMO

Memory<T>

Jak to použít?

- BCL má spoustu nových overloadů
  - Metody `Parse` a `TryParse` podporují `Span<char>` a `Span<byte>`
  - `Read` a `Write` na streamech podporují `Memory<byte>`

- `String` má metodu `AsSpan()`

- `MemoryPool<T>` umožňuje půjčit si kus paměti dané velikosti
- Podobně existuje `ArrayPool<T>`

Příklad: Parsování čísel

- Textový soubor s 10 000 řádky
- Na každém je 0 – 1000 desetinných čísel oddělených tabem
- Jako výstup chceme double[][]

- Vstupní soubor má ~90 MB
- Na výstupu je 10 000 * 500 * 8 = cca 40 MB + režie spojená s alokací polí

riganti

Naivní metoda

- `File.ReadAllText`
- `Split` podle konců řádků
- `Split` každého řádku podle tabulátorů
- `double.Parse`
- `ToArray`

- Neefektivní
  - Nepoužívají se znaky mimo ASCII – stačí pracovat s bajty
  - Split vyrábí nové instance stringu – zbytečné
  - Pro každé číslo potřebujeme vyrobit string pro `double.Parse`

riganti

DEMO

String vs Span<byte>

riganti

Dá se to ještě zrychlit?

- Vyrábíme `List<T>` a voláme `ToArray()`
- Když si prvky spočítáme předem, můžeme vyrobit pole rovnou
  - Nutno změřit – nemusí být rychlejší, ale ušetříme paměť

- Jak to spočítat co nejrychleji?
  - `Span<T>.Count()` využívá vektorizaci

riganti

DEMO

Jak funguje vektorizace?

riganti

Jde to ještě vylepšit?

- Nemusíme načítat celý soubor do paměti
- Data můžeme číst postupně do malého bufferu
- Lze využít `MemoryPool<T>`, abychom nealokovali nová pole

riganti

DEMO

Čtení do bufferu

K čemu je to dobré?

- V běžném životě `Span` a `Memory` moc nevyužijete
- Nepřímo jej ale používáme denně
  - Mnoho interních funkcí v .NETu je na tom postavených

- `System.Text.Json` používá podobné principy
  - K dispozici je i `Utf8JsonReader`
    - Nepotřebuje vstup jako `string`
  - Velmi podobné jako náš parser
    - `Read()` vrátí `false`, když potřebuje další vstup

riganti

DEMO

System.Text.Json

riganti

Shrnutí

- Pokud pracujete s velkým množstvím dat,
  `Span` a `Memory` mohou ušetřit zbytečné alokace paměti
- Mnoho funkcí v .NETu využívá interně vektorizaci
  - `string.IndexOf`, `string.Replace`, `Encoding.GetBytes()`…
  - Hodí se rozumět tomu principu
- Pokud vstupu stačí ASCII sada, stačí pracovat jen s bajty

riganti

Kniha

- Jak zmigrovat webové aplikace z .NET Frameworku do .NET 8+
  - ASP.NET Web Forms
  - ASP.NET MVC
  - ASP.NET Web API
  - SignalR
  - Entity Framework 6
  - Forms Authentication a ASP.NET Identity
- Co tím získáte?
- Jak přesvědčit management?

Update Days

.NET Performance

Build blazing fast and super-efficient .NET applications

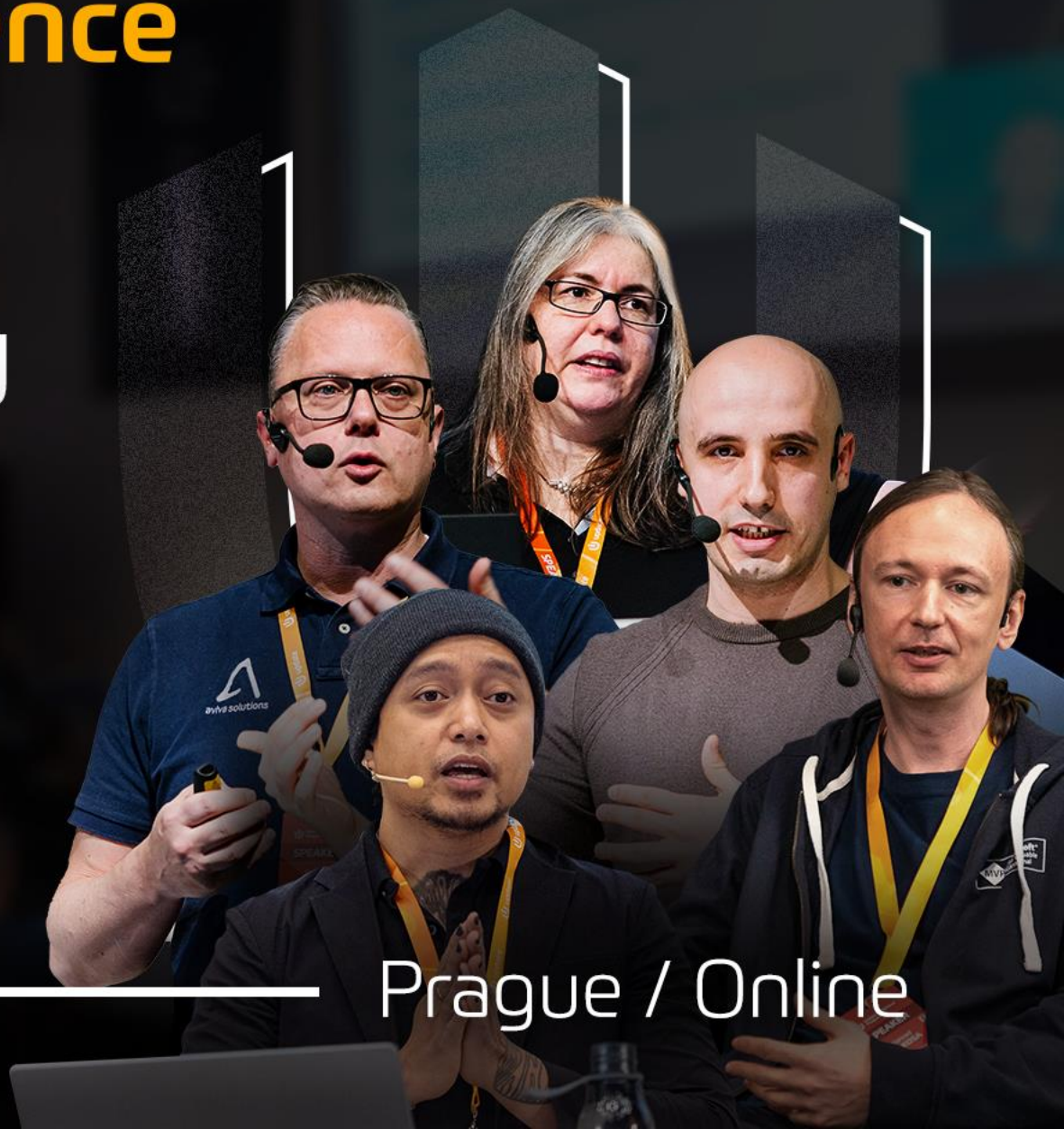
30 - 31 October 2024
Krakow / Online

Update Conference Prague 2024

.NET - Cloud - Security

.NET developer conference
in the heart of Europe

14 - 15 November 2024 — Prague / Online



riganti

Q&A

Tomáš Herceg
CEO @ RIGANTI
Microsoft MVP