DOT NET MLG

Pair-Programming con IA para conseguir código más eficiente en .NET

Verisk™ SECTOR ALARM prodware LEMONCODE METADEV plain concepts

in jcanton

X jcant0n

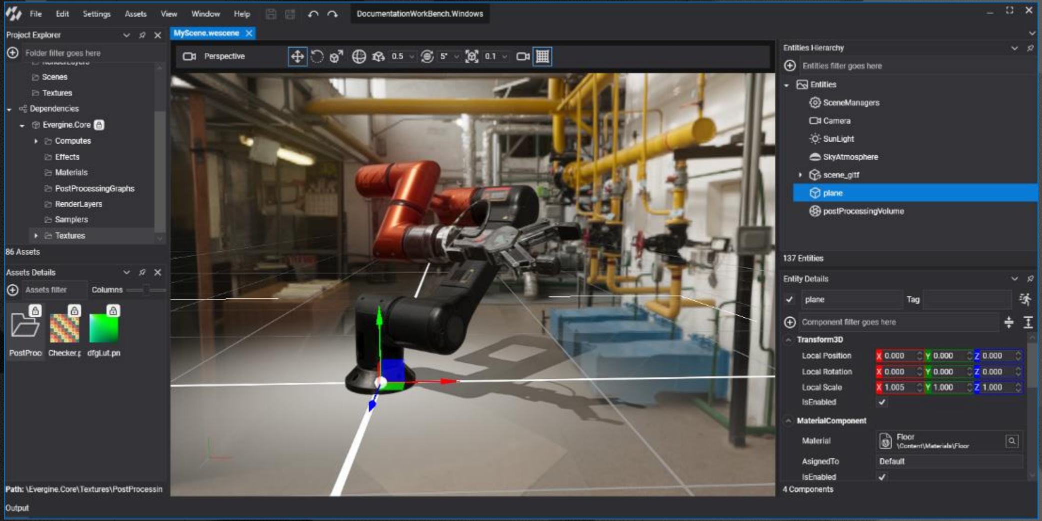
Javier Cantón Ferrero

Javier is a Computer Science Engineer who has always had a passion for 3D graphics and software architecture. His professional achievements include being MVP for Windows DirectX and DirectX XNA for nine years, Xbox Ambassador, as well as Microsoft Student Partner and Microsoft Most Valuable Student during his years at college. Currently he works as Chief Technology Innovation Officer.

EVERGINE

Evergine is our cross-platform graphics engine designed for corporate applications. It has been created to offer enterprise solutions with 3D, Augmented, Virtual and Mixed Reality capabilities.

A flexible and versatile tool to create unique visual experiences, adapted to the needs and functionalities of all types of industries.
Evergine is free to use, adapts to any platform and is ready to be multi-platform.



plane Tag

Transform3D

	X	Y	Z
Local Position	0.000	0.000	0.000
Local Rotation	0.000	0.000	0.000
Local Scale	1.005	1.000	1.000

IsEnabled ✓

MaterialComponent

Material: Floor \Content\Materials\Floor
AsignedTo: Default
IsEnabled ✓

Windows

Android

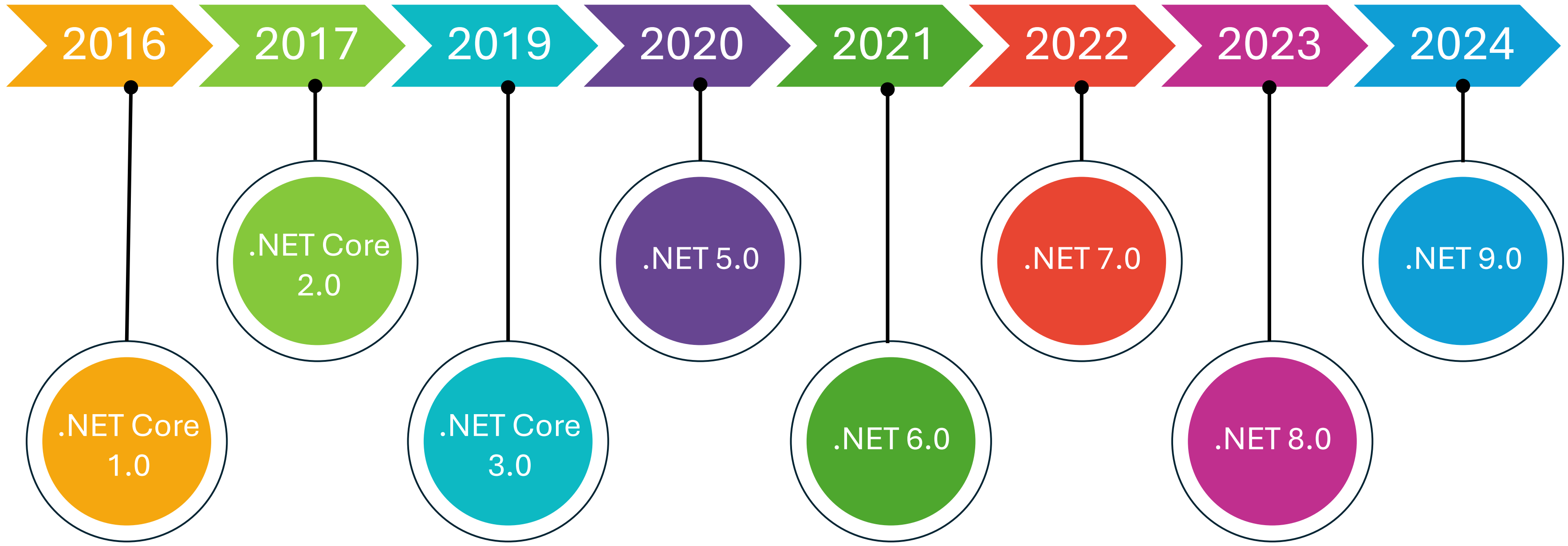
Linux

MacOs + iOS

Web

.NET Performance

.NET Evolution



NET features

Hardware Intrinsics

P/Invoke source Generation

On-Stack Replacement (OSR)

Ref Struct

Dynamic GC

Ref readonly

Profile-Guide Optimization (PGO)

Record Struct

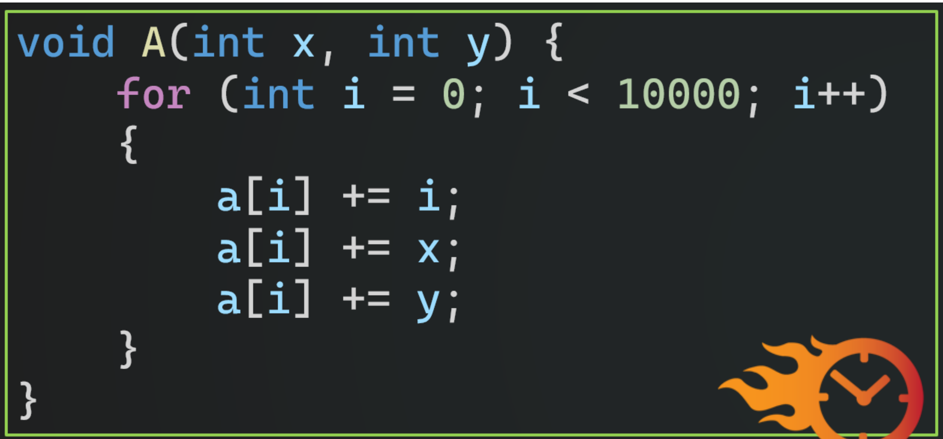
Ahead-of-Time (AOT)

DOT NET MLG

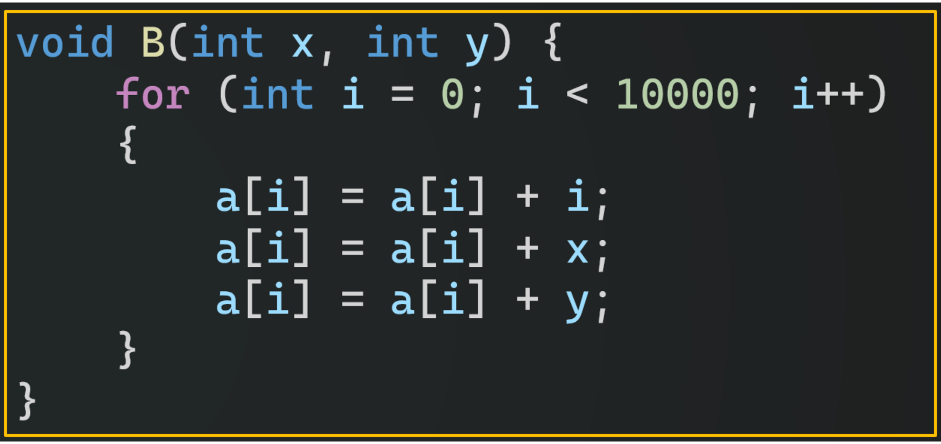
Micro-Optimization

- BCL and Runtime
- Real-Time applications
- Graphics development
- ML or Math libraries

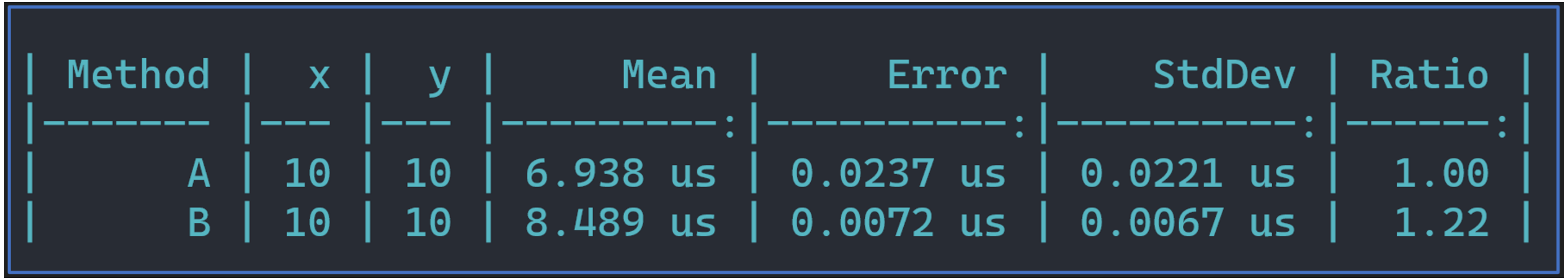
What is faster (Assignment)?



```
void A(int x, int y) {
    for (int i = 0; i < 10000; i++)
    {
        a[i] += i;
        a[i] += x;
        a[i] += y;
    }
}
```

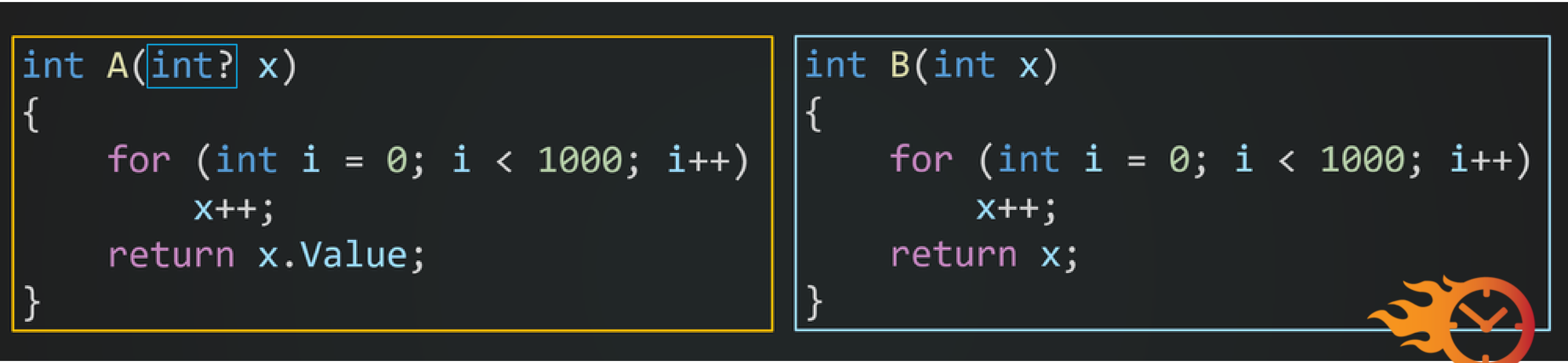


```
void B(int x, int y) {
    for (int i = 0; i < 10000; i++)
    {
        a[i] = a[i] + i;
        a[i] = a[i] + x;
        a[i] = a[i] + y;
    }
}
```



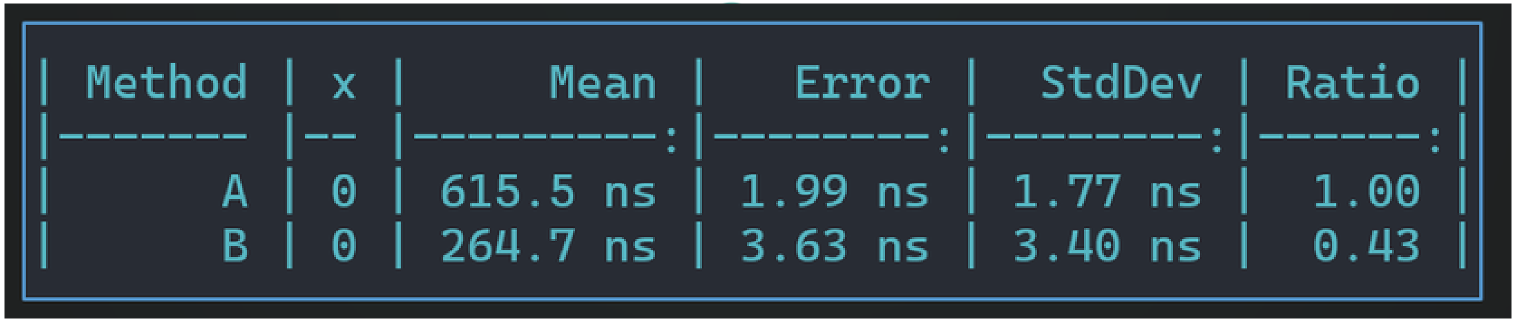
```
| Method |  x |  y |     Mean |     Error |    StdDev | Ratio |
|------- |--- |--- |---------:|----------:|----------:|------:|
|      A | 10 | 10 | 6.938 us | 0.0237 us | 0.0221 us |  1.00 |
|      B | 10 | 10 | 8.489 us | 0.0072 us | 0.0067 us |  1.22 |
```

What is faster (Nullable)?



```
int A(int? x)
{
    for (int i = 0; i < 1000; i++)
        x++;
    return x.Value;
}
```

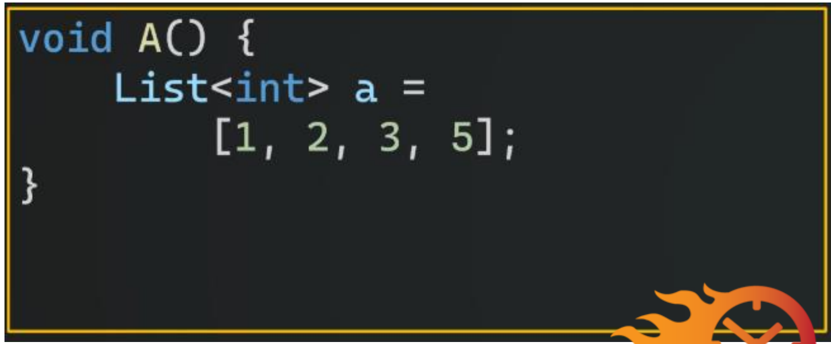
```
int B(int x)
{
    for (int i = 0; i < 1000; i++)
        x++;
    return x;
}
```



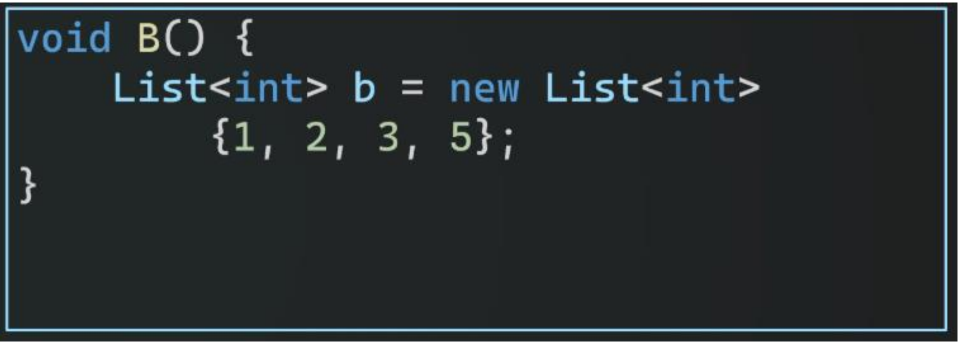
```
| Method | x |       Mean |     Error |    StdDev | Ratio |
|------- |-- |-----------:|----------:|----------:|------:|
|      A | 0 |   615.5 ns |   1.99 ns |   1.77 ns |  1.00 |
|      B | 0 |   264.7 ns |   3.63 ns |   3.40 ns |  0.43 |
```

@badamczewski01

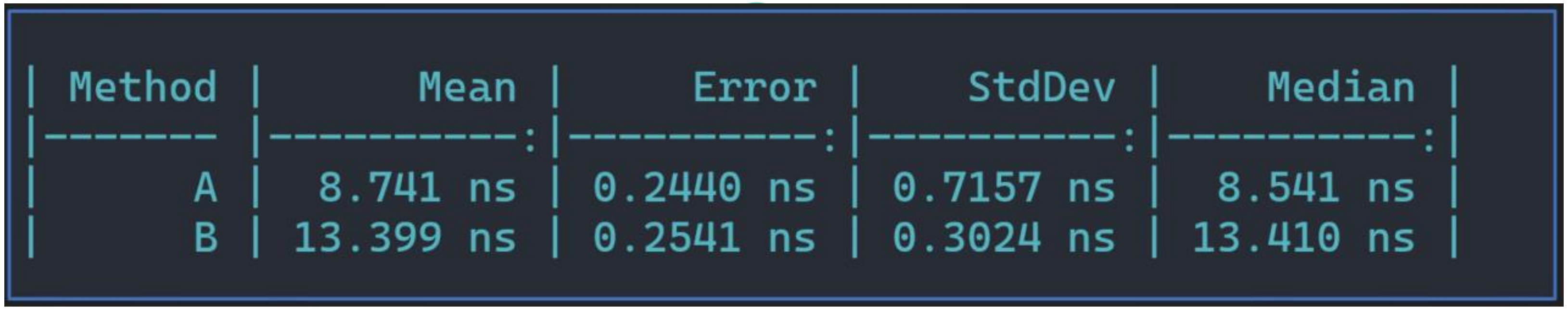
What is faster (Collection expressions)?



```
void A() {
    List<int> a =
        [1, 2, 3, 5];
}
```



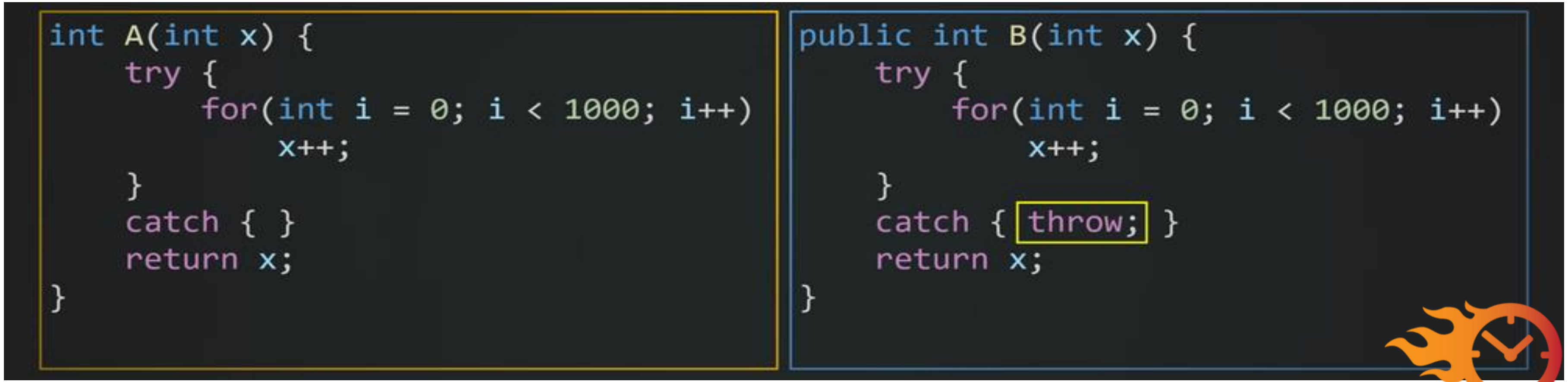
```
void B() {
    List<int> b = new List<int>
        {1, 2, 3, 5};
}
```



```
| Method |      Mean |     Error |    StdDev |    Median |
|------- |----------:|----------:|----------:|----------:|
|      A |  8.741 ns | 0.2440 ns | 0.7157 ns |  8.541 ns |
|      B | 13.399 ns | 0.2541 ns | 0.3024 ns | 13.410 ns |
```

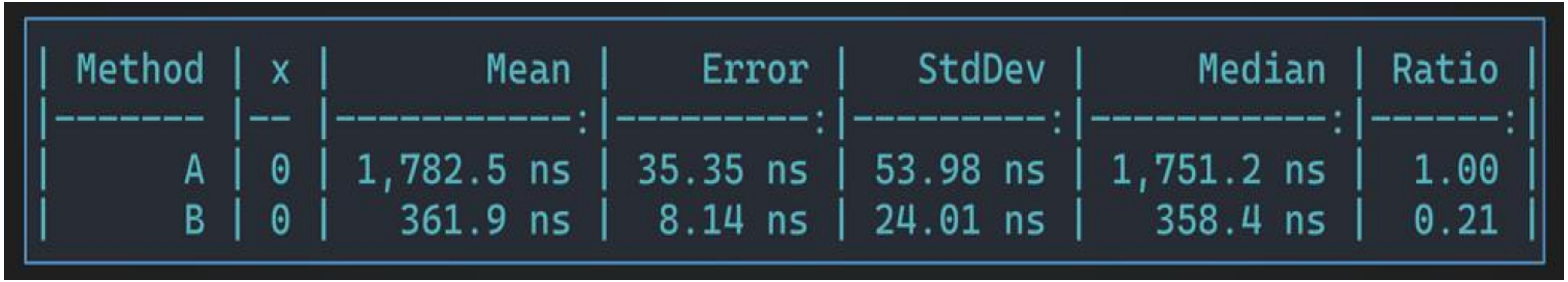
@badamczewski01

What is faster (Try-catch)?



```
int A(int x) {
    try {
        for(int i = 0; i < 1000; i++)
            x++;
    }
    catch { }
    return x;
}
```

```
public int B(int x) {
    try {
        for(int i = 0; i < 1000; i++)
            x++;
    }
    catch { throw; }
    return x;
}
```



```
| Method | x |        Mean |     Error |    StdDev |      Median | Ratio |
|------- |-- |------------:|----------:|----------:|------------:|------:|
|      A | 0 | 1,782.5 ns | 35.35 ns | 53.98 ns | 1,751.2 ns |  1.00 |
|      B | 0 |    361.9 ns |  8.14 ns | 24.01 ns |    358.4 ns |  0.21 |
```

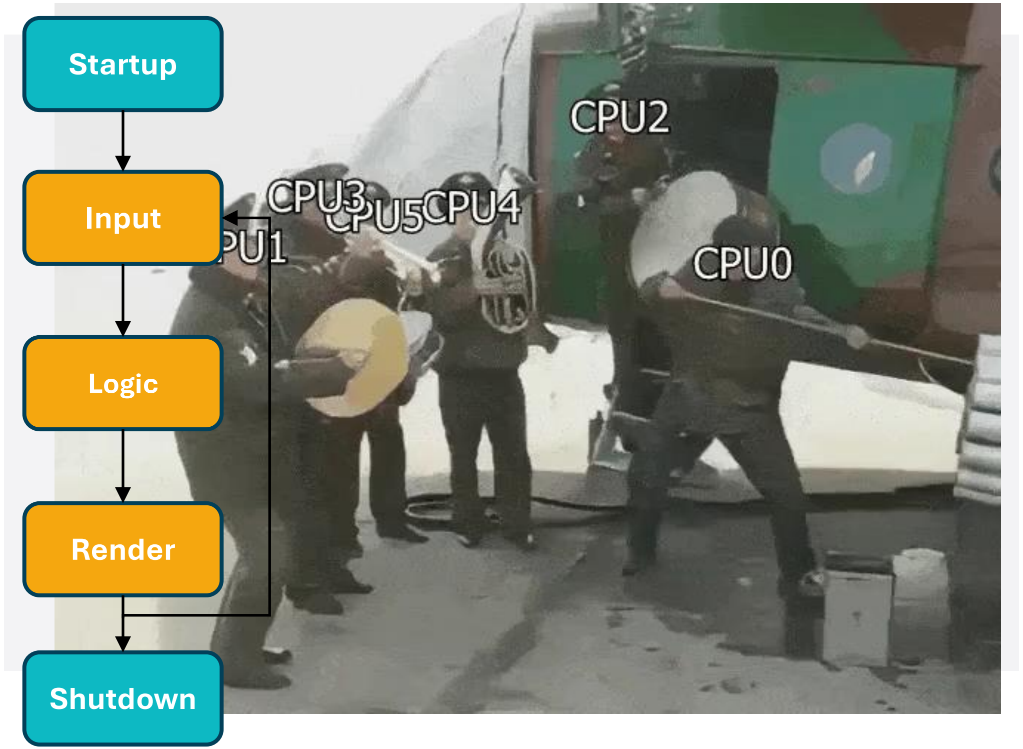
@badamczewski01

Measure

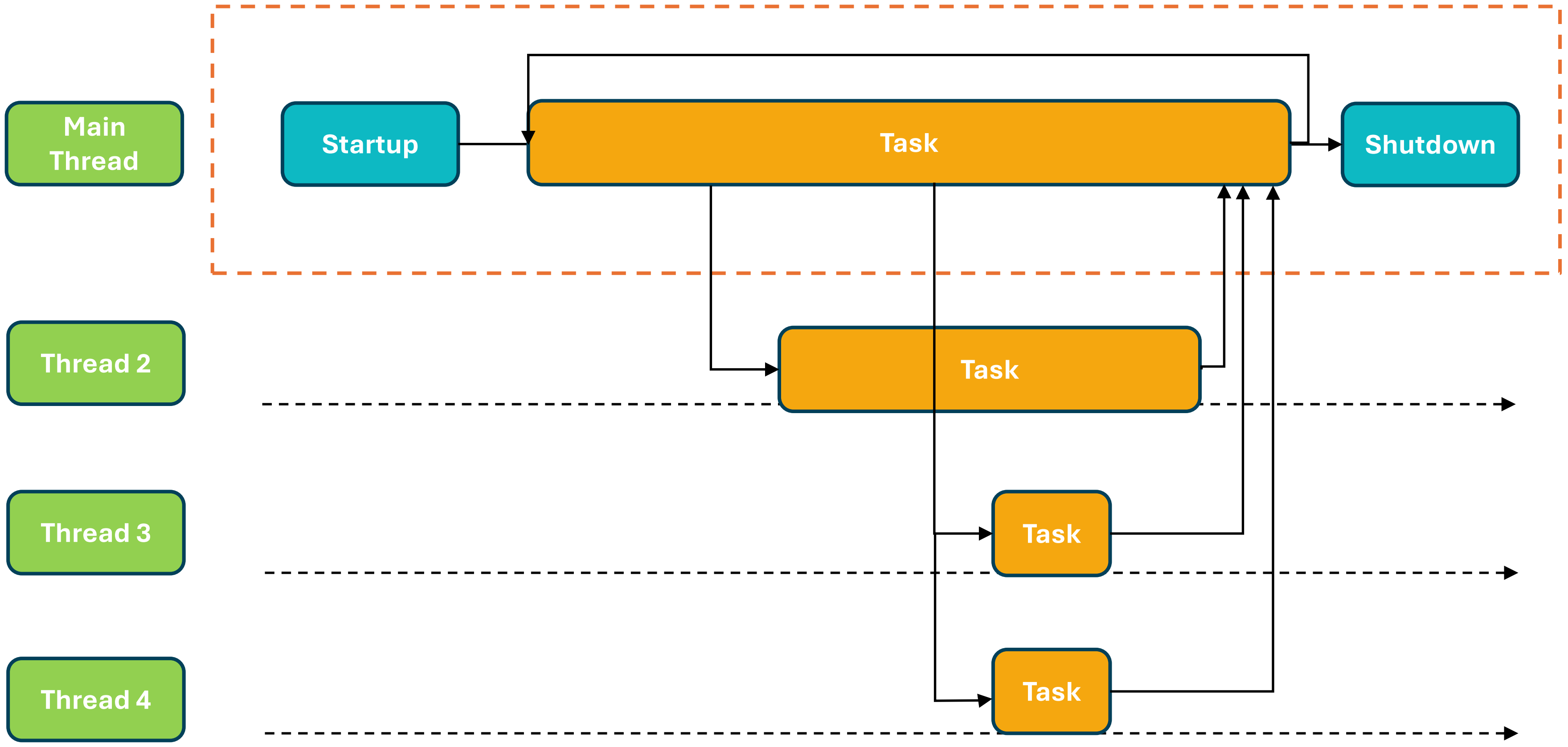
BenchmarkDotNet
Powerful .NET library for benchmarking

JobSystem

Usually 3D app



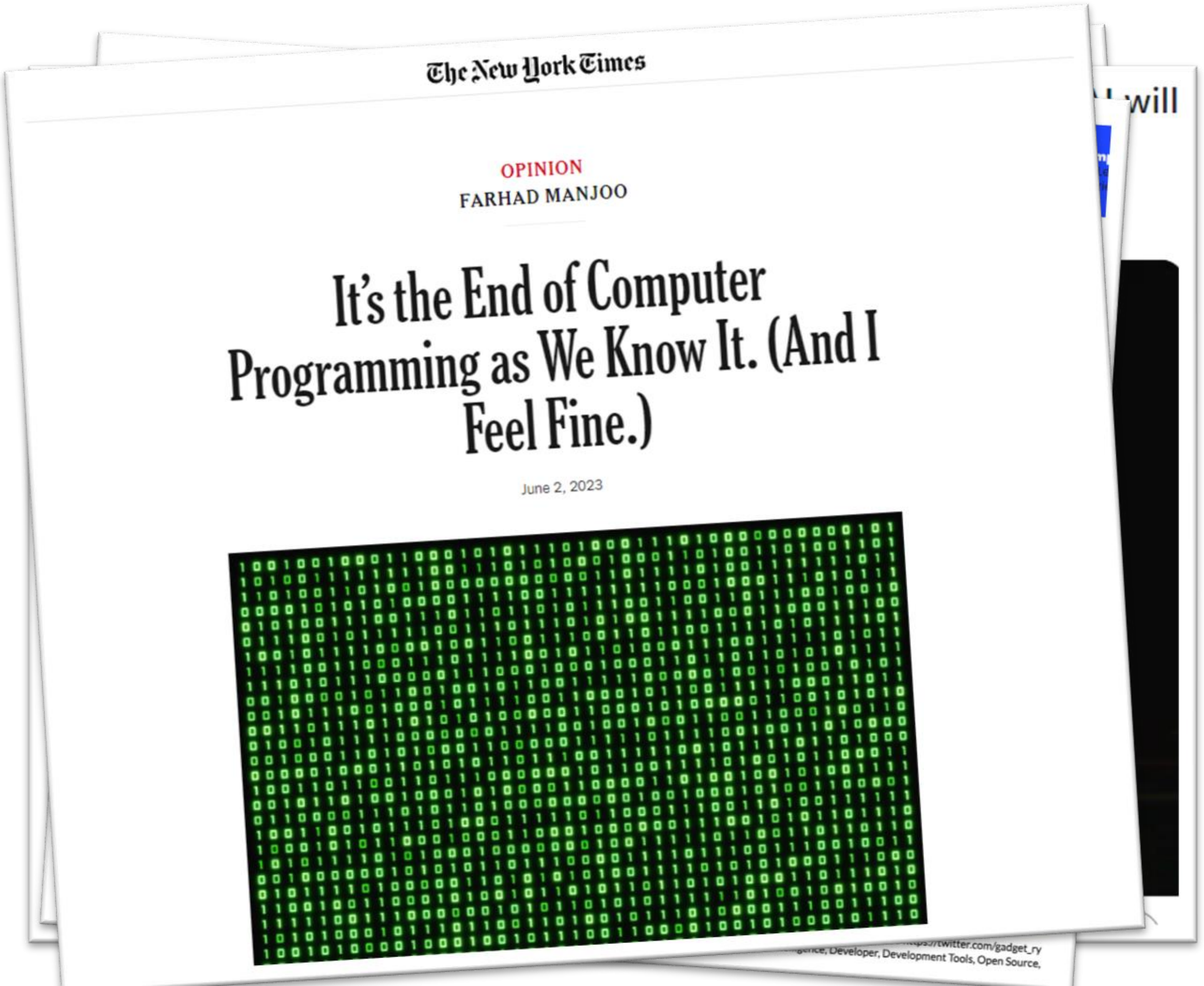
JobSystem


Main Thread
Startup
Task
Shutdown
Thread 2
Task
Thread 3
Task
Thread 4
Task

GenAI for coding

News

DOT NET MLG



The New York Times

OPINION

FARHAD MANJOO

It's the End of Computer Programming as We Know It. (And I Feel Fine.)

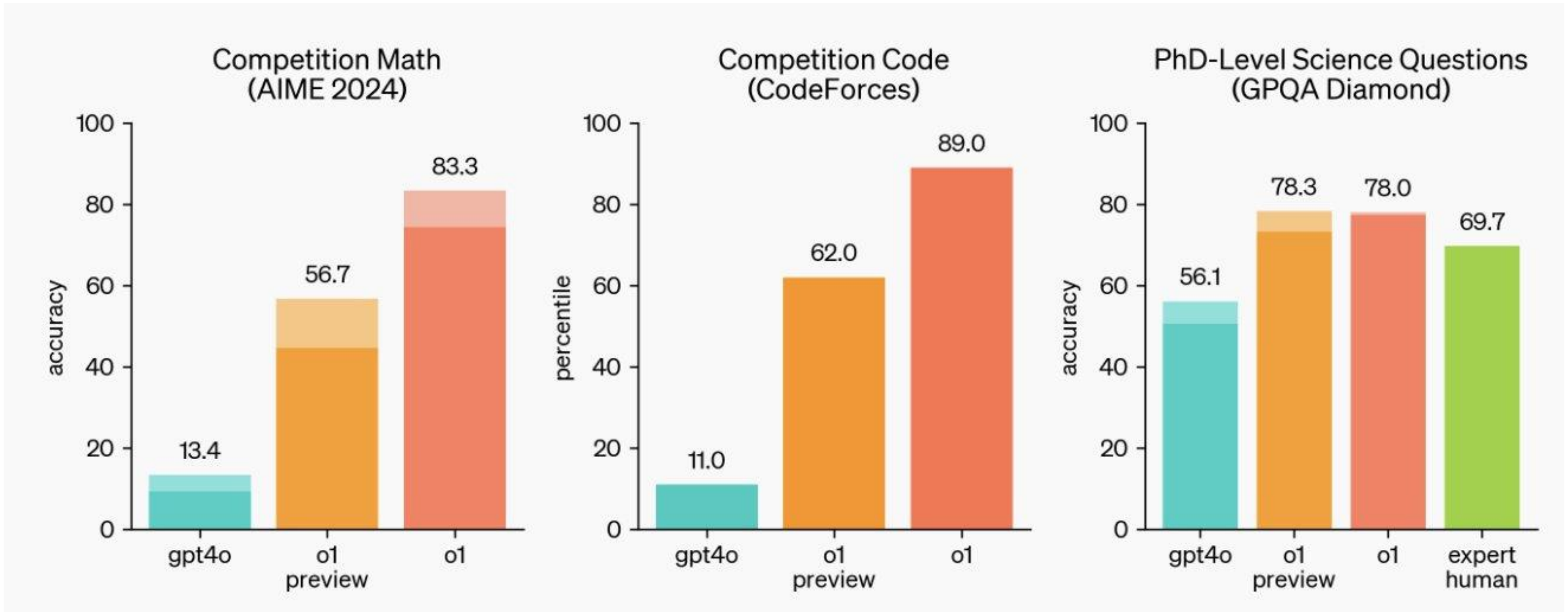
June 2, 2023

will

https://twitter.com/gadget_ry

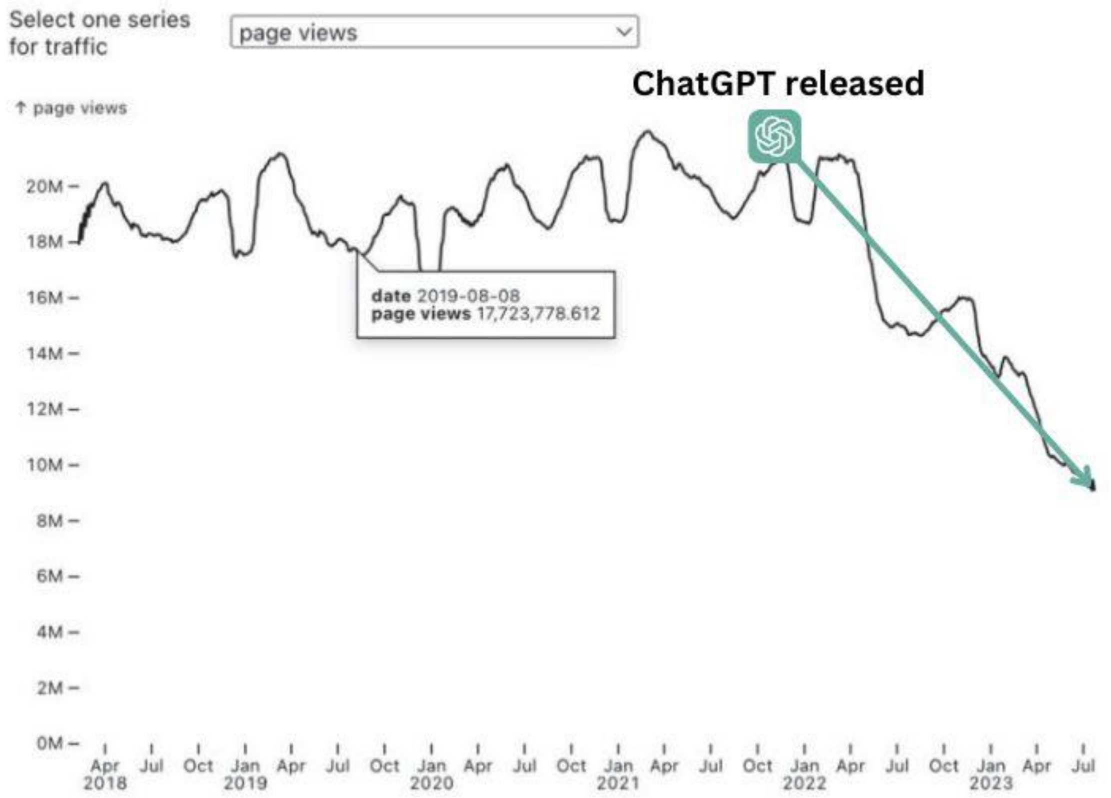
Developer, Development Tools, Open Source,

OpenAI



Competition Math (AIME 2024)
accuracy
0
20
40
60
80
100
13.4
56.7
83.3
gpt4o
o1 preview
o1
Competition Code (CodeForces)
percentile
0
20
40
60
80
100
11.0
62.0
89.0
gpt4o
o1 preview
o1
PhD-Level Science Questions (GPQA Diamond)
accuracy
0
20
40
60
80
100
56.1
78.3
78.0
69.7
gpt4o
o1 preview
o1
expert human

stackoverflow website traffic



date 2019-08-08
page views 17,723,778.612

LLMs for coding

Chatgpt 4o

OpenAI o1 preview

V0 vercel

Claude 3.5

Github Copilot

Pair-Programming

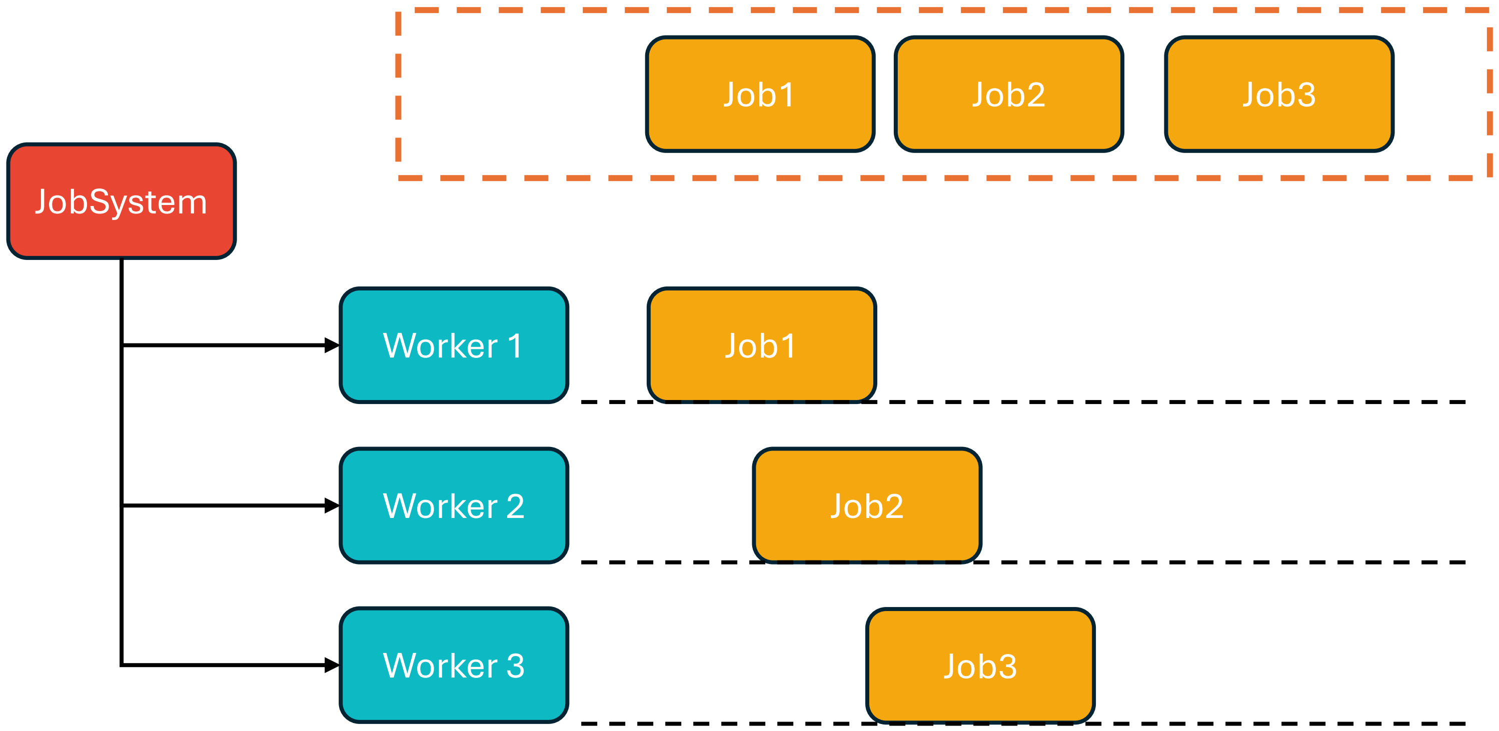
Pair-Programming

- **Driver**
  - The person who write the code.
  - Handles the implementation details, syntax and immediate problem-solving.
- **Navigator**
  - The person who reviews and provides guidance.
  - Thinks about the overall approach

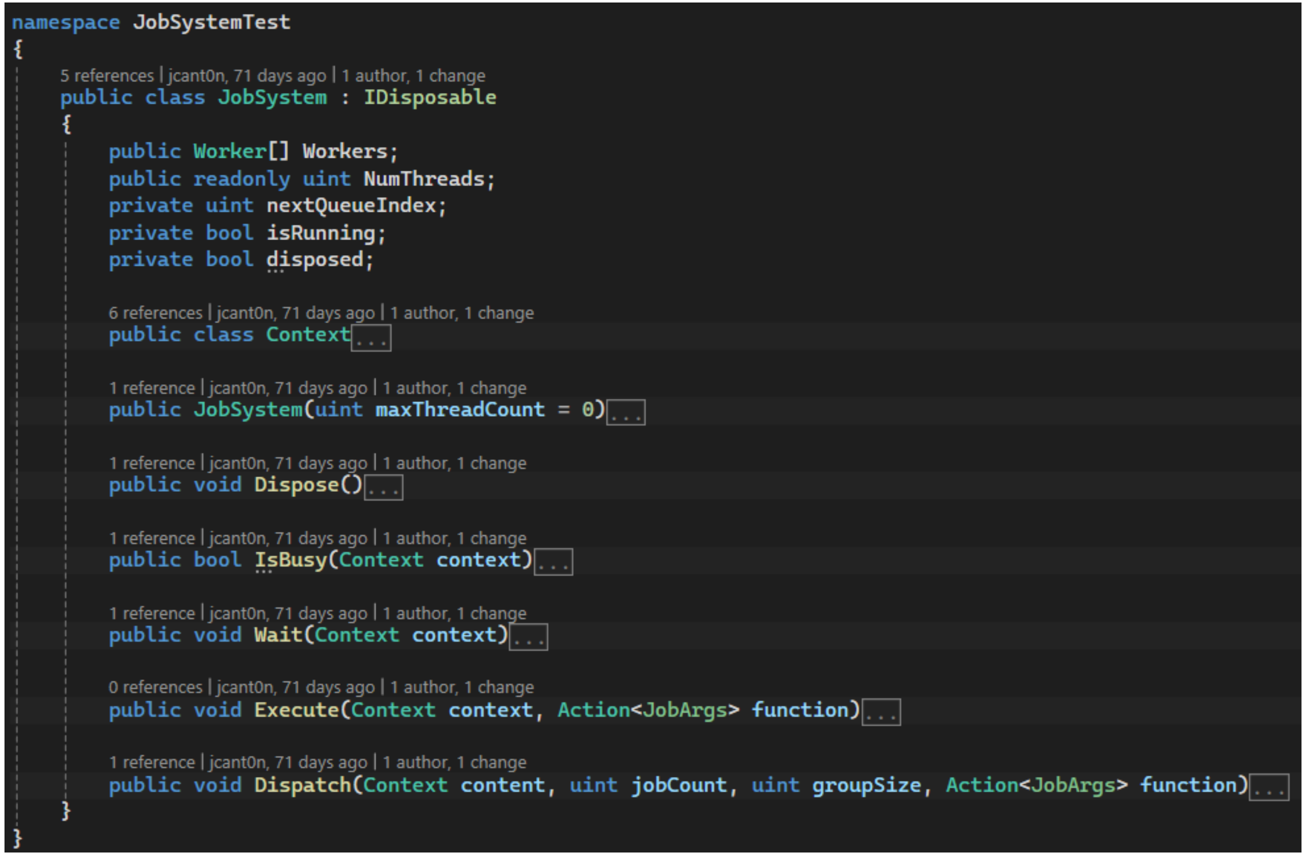
Rubber Duck Programming

- **How it works:**
  - Pace your "duck" in front of you.
  - Explain your code or problem you are facing as if the duck needs to understand it.
  - As you verbalize your logic, your brain often identifies gaps or bugs that where previously overlooked.

Basic JobSystem

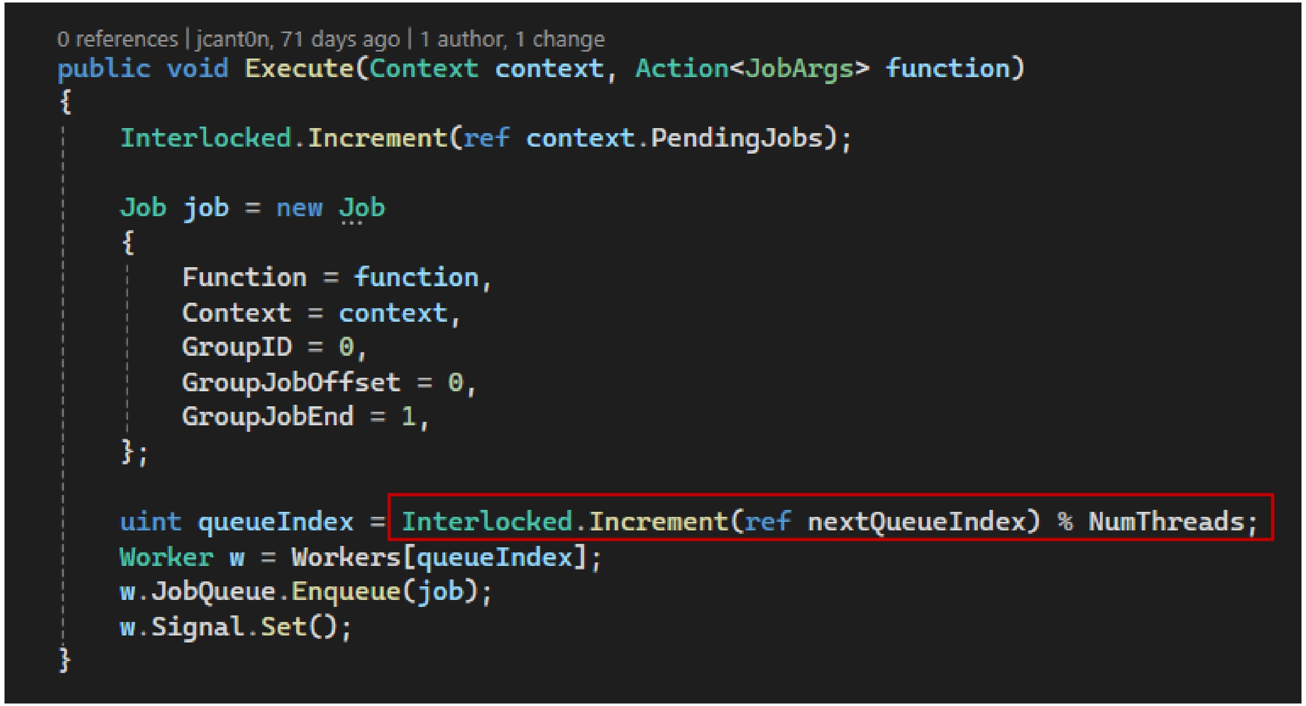


JobSystem

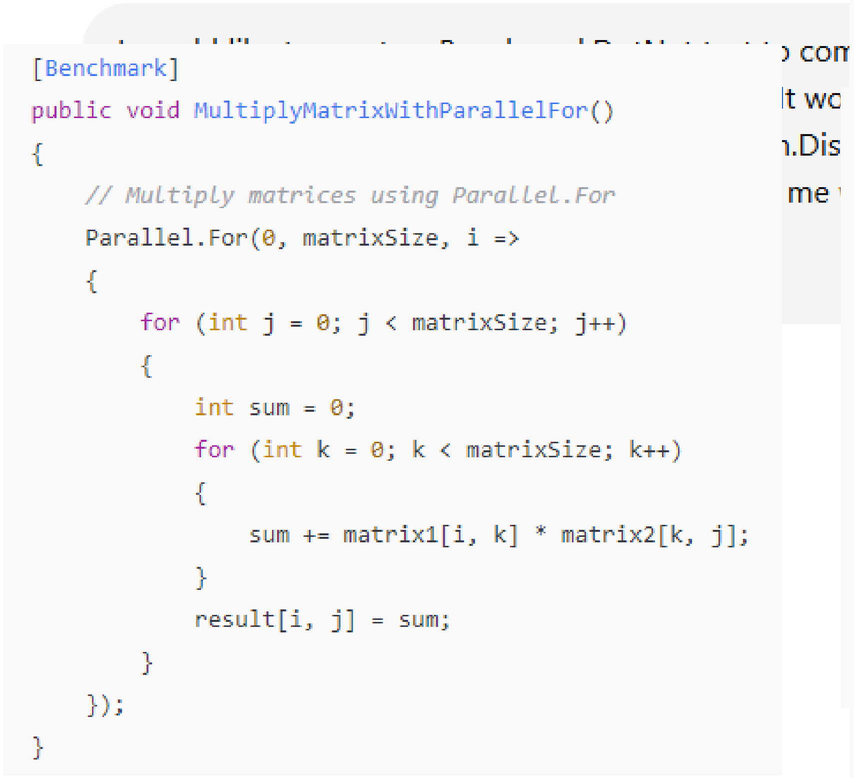


```
namespace JobSystemTest
{
    5 references | jcant0n, 71 days ago | 1 author, 1 change
    public class JobSystem : IDisposable
    {
        public Worker[] Workers;
        public readonly uint NumThreads;
        private uint nextQueueIndex;
        private bool isRunning;
        private bool disposed;

        6 references | jcant0n, 71 days ago | 1 author, 1 change
        public class Context...

        1 reference | jcant0n, 71 days ago | 1 author, 1 change
        public JobSystem(uint maxThreadCount = 0)...

        1 reference | jcant0n, 71 days ago | 1 author, 1 change
        public void Dispose()...

        1 reference | jcant0n, 71 days ago | 1 author, 1 change
        public bool IsBusy(Context context)...

        1 reference | jcant0n, 71 days ago | 1 author, 1 change
        public void Wait(Context context)...

        0 references | jcant0n, 71 days ago | 1 author, 1 change
        public void Execute(Context context, Action<JobArgs> function)...

        1 reference | jcant0n, 71 days ago | 1 author, 1 change
        public void Dispatch(Context content, uint jobCount, uint groupSize, Action<JobArgs> function)...
    }
}
```

Execute

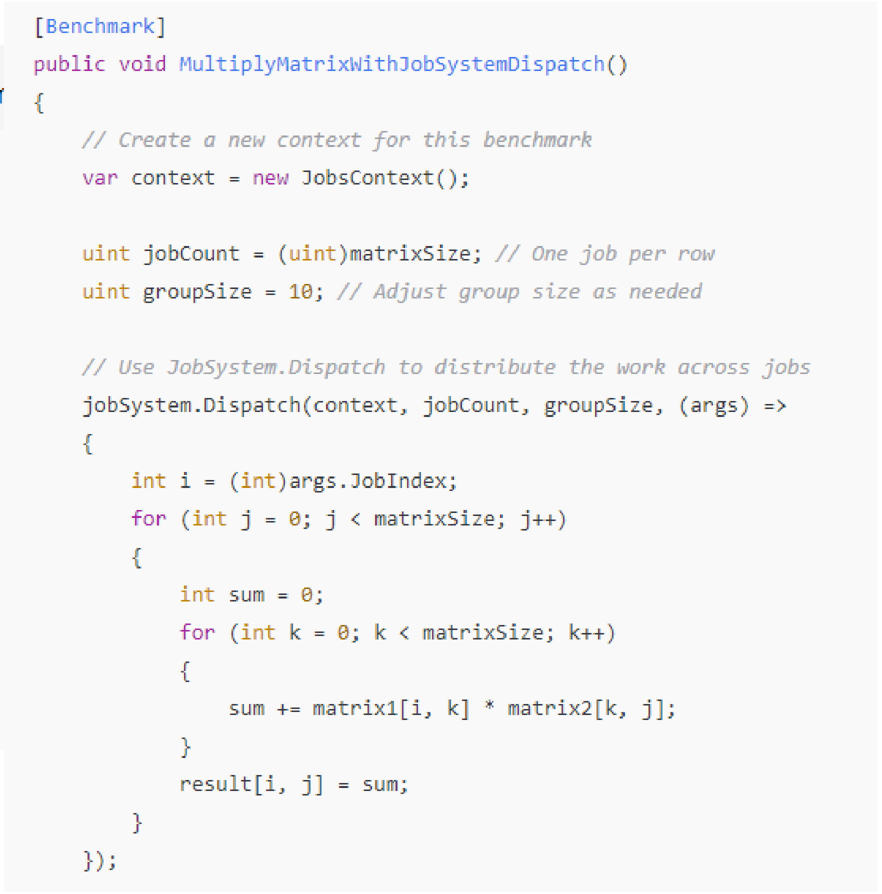

```
0 references | jcant0n, 71 days ago | 1 author, 1 change
public void Execute(Context context, Action<JobArgs> function)
{
    Interlocked.Increment(ref context.PendingJobs);

    Job job = new Job
    {
        Function = function,
        Context = context,
        GroupID = 0,
        GroupJobOffset = 0,
        GroupJobEnd = 1,
    };

    uint queueIndex = Interlocked.Increment(ref nextQueueIndex) % NumThreads;
    Worker w = Workers[queueIndex];
    w.JobQueue.Enqueue(job);
    w.Signal.Set();
}
```

Benchmark (AI)



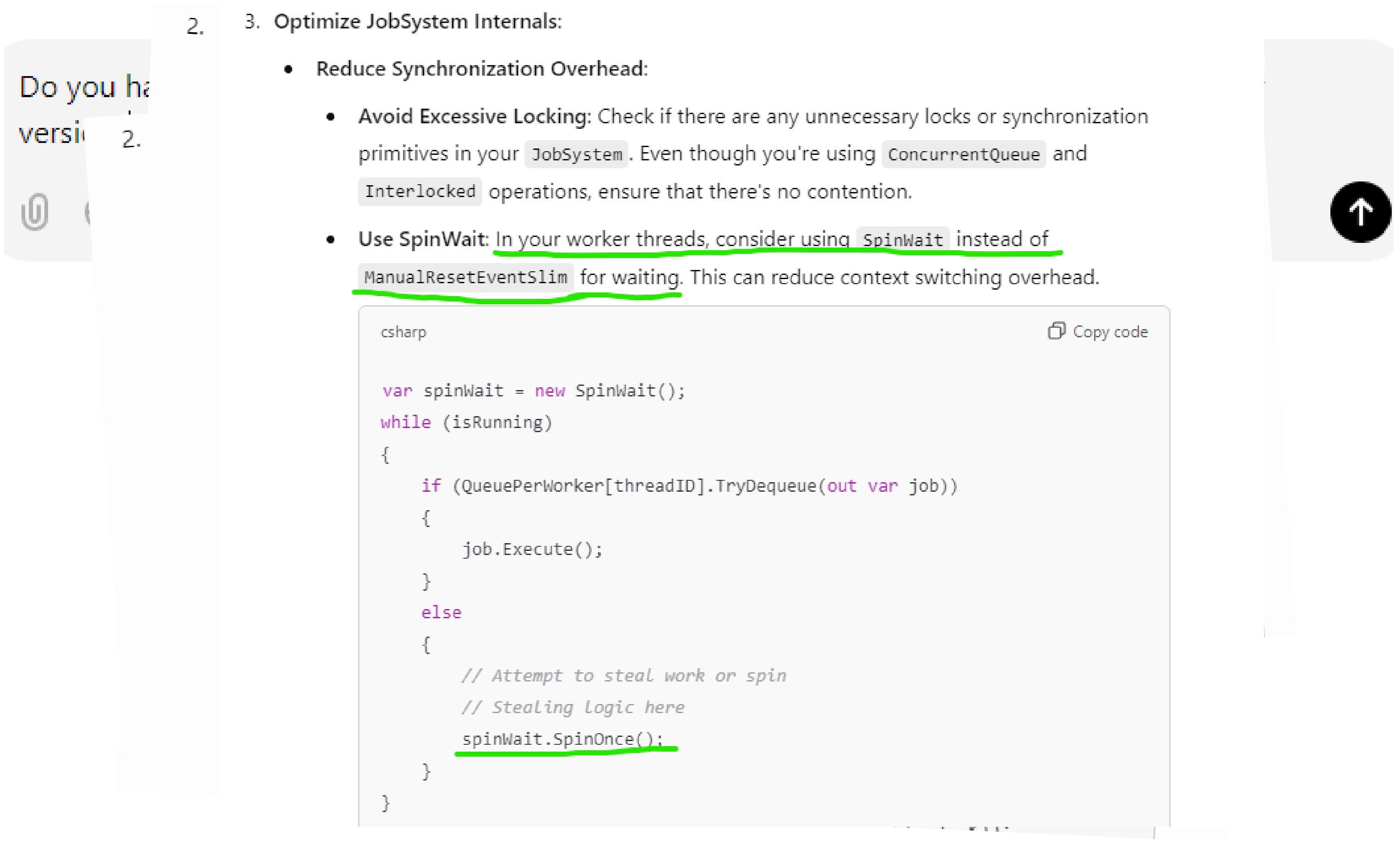
o com
It wo
.Dis
me

```
[Benchmark]
public void MultiplyMatrixWithParallelFor()
{
    // Multiply matrices using Parallel.For
    Parallel.For(0, matrixSize, i =>
    {
        for (int j = 0; j < matrixSize; j++)
        {
            int sum = 0;
            for (int k = 0; k < matrixSize; k++)
            {
                sum += matrix1[i, k] * matrix2[k, j];
            }
            result[i, j] = sum;
        }
    });
}
```



```
[Benchmark]
public void MultiplyMatrixWithJobSystemDispatch()
{
    // Create a new context for this benchmark
    var context = new JobsContext();

    uint jobCount = (uint)matrixSize; // One job per row
    uint groupSize = 10; // Adjust group size as needed

    // Use JobSystem.Dispatch to distribute the work across jobs
    jobSystem.Dispatch(context, jobCount, groupSize, (args) =>
    {
        int i = (int)args.JobIndex;
        for (int j = 0; j < matrixSize; j++)
        {
            int sum = 0;
            for (int k = 0; k < matrixSize; k++)
            {
                sum += matrix1[i, k] * matrix2[k, j];
            }
            result[i, j] = sum;
        }
    });
```

New ideas



Do you ha

versi

2.

2.

3. Optimize JobSystem Internals:

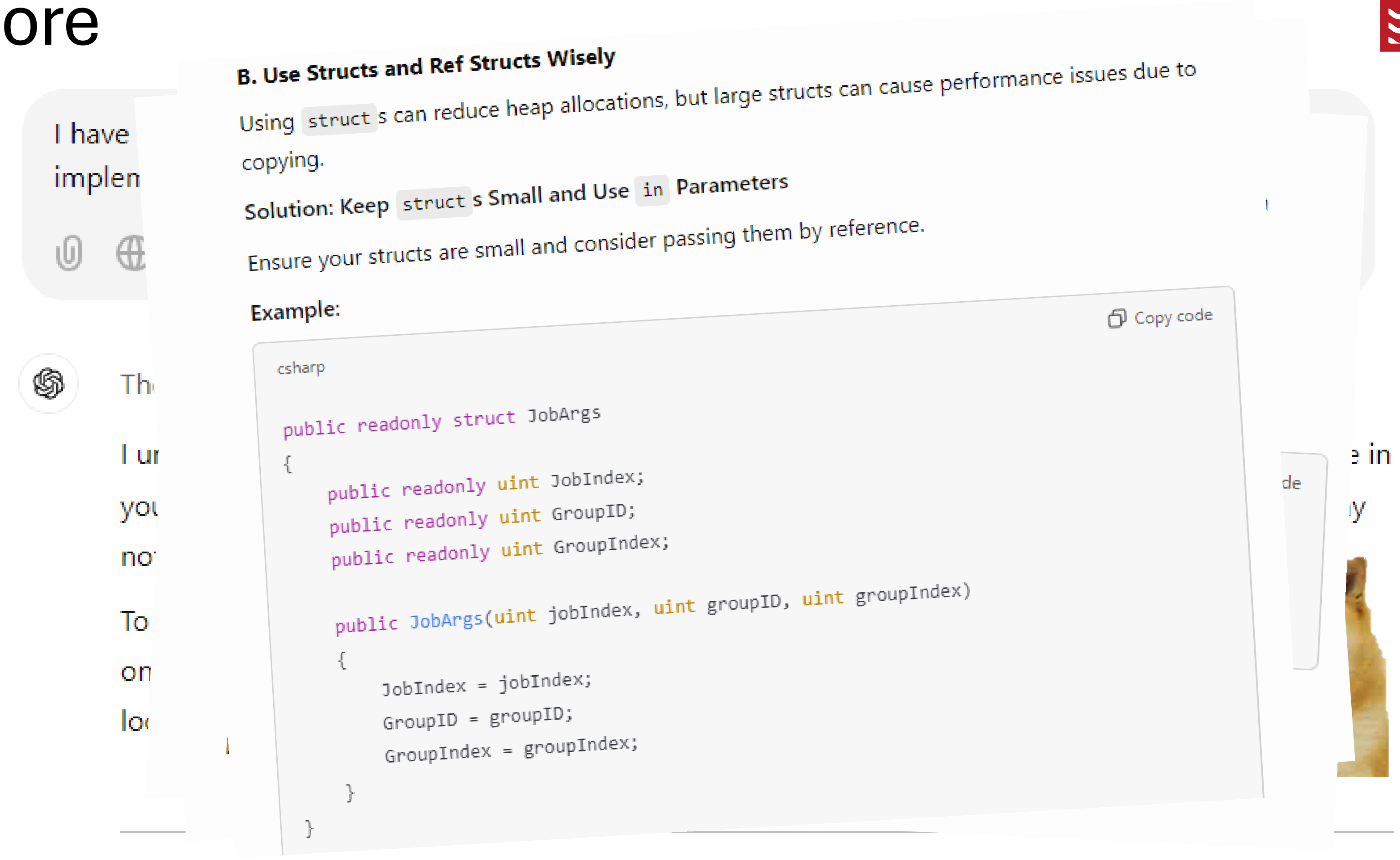
- Reduce Synchronization Overhead:
  - **Avoid Excessive Locking:** Check if there are any unnecessary locks or synchronization primitives in your `JobSystem`. Even though you're using `ConcurrentQueue` and `Interlocked` operations, ensure that there's no contention.
  - **Use SpinWait:** In your worker threads, consider using `SpinWait` instead of `ManualResetEventSlim` for waiting. This can reduce context switching overhead.

csharp

Copy code

```csharp
var spinWait = new SpinWait();
while (isRunning)
{
    if (QueuePerWorker[threadID].TryDequeue(out var job))
    {
        job.Execute();
    }
    else
    {
        // Attempt to steal work or spin
        // Stealing logic here
        spinWait.SpinOnce();
    }
}
```

More



I have
implen

Th
I ur
you
no
To
on
lo

B. Use Structs and Ref Structs Wisely

Using `struct`s can reduce heap allocations, but large structs can cause performance issues due to copying.

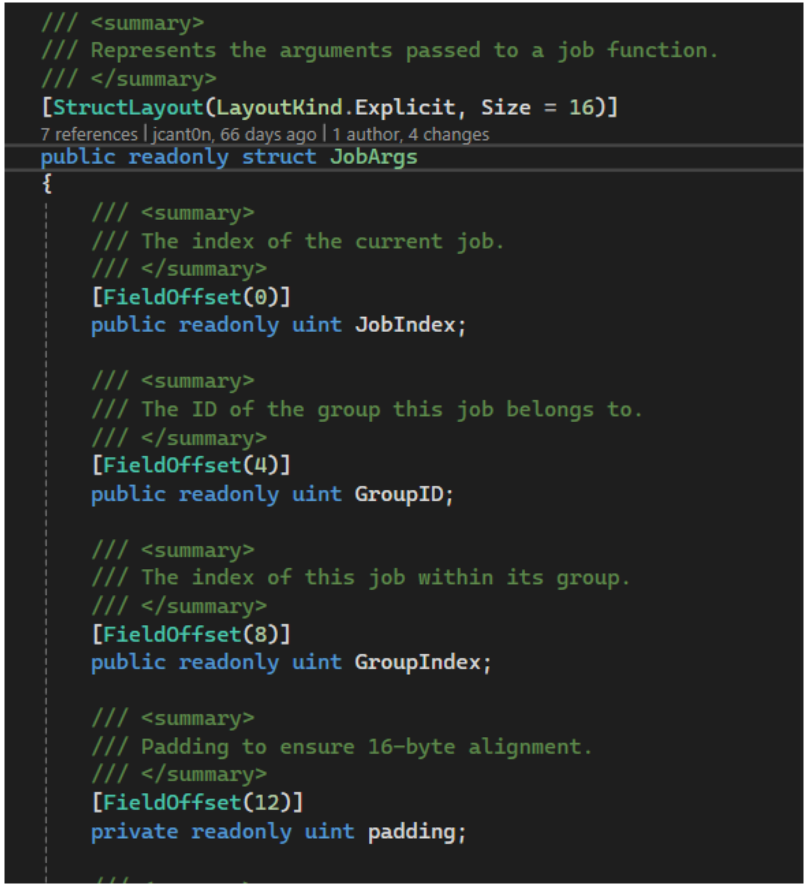
Solution: Keep `struct`s Small and Use `in` Parameters

Ensure your structs are small and consider passing them by reference.

Example:

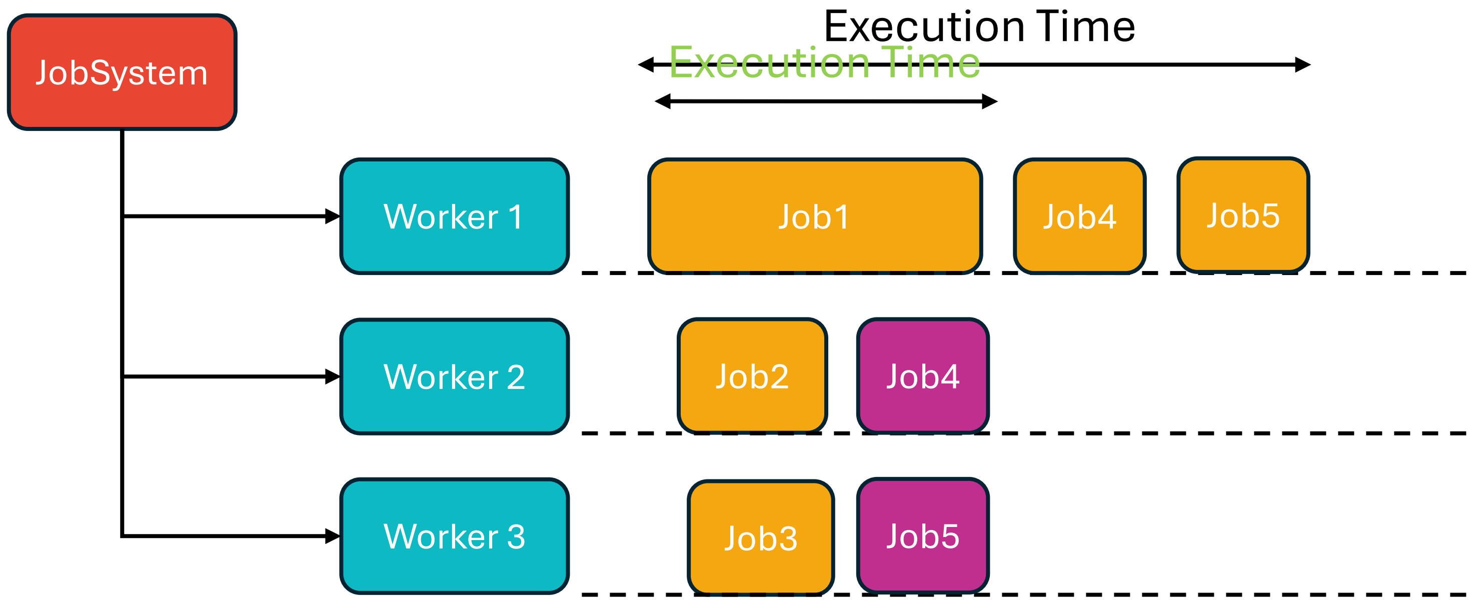
```csharp
public readonly struct JobArgs
{
    public readonly uint JobIndex;
    public readonly uint GroupID;
    public readonly uint GroupIndex;

    public JobArgs(uint jobIndex, uint groupID, uint groupIndex)
    {
        JobIndex = jobIndex;
        GroupID = groupID;
        GroupIndex = groupIndex;
    }
}
```

Inmutable struct

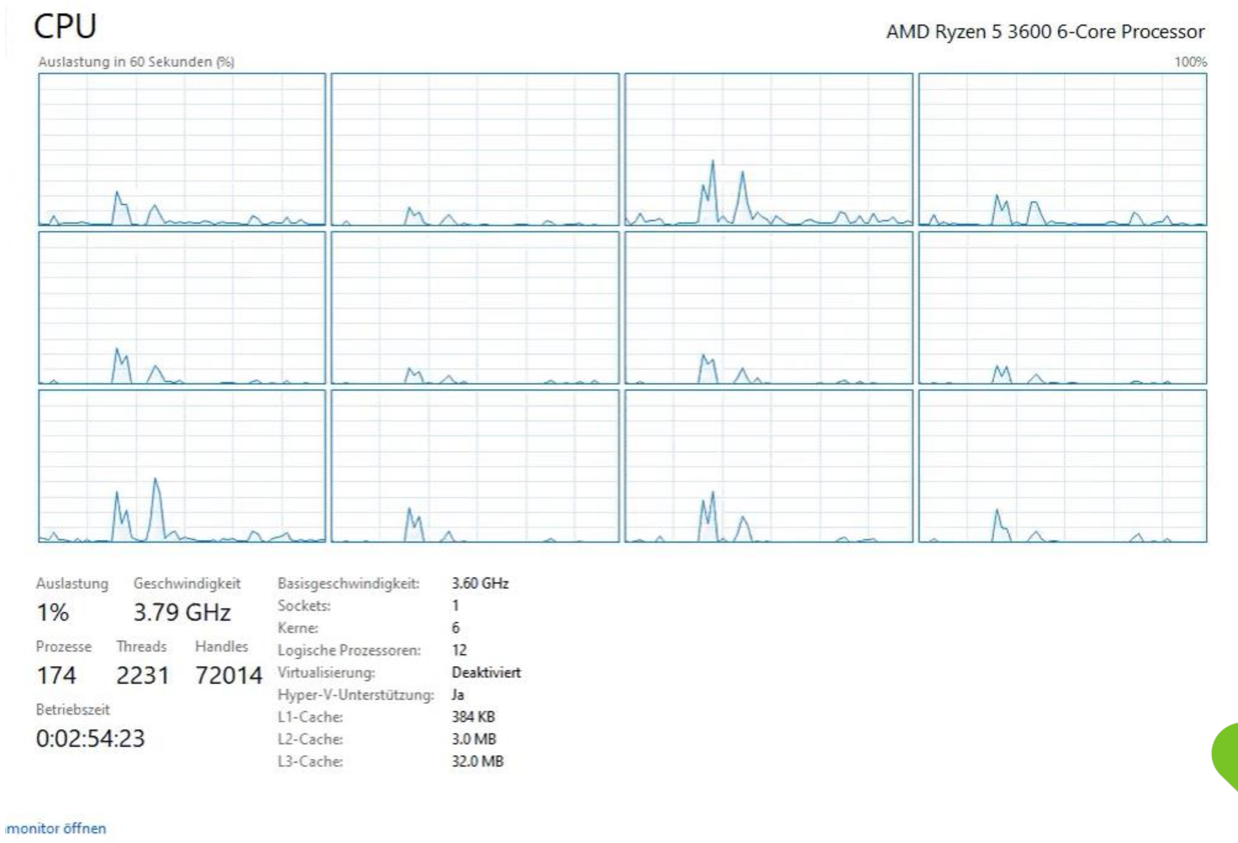


```
/// <summary>
/// Represents the arguments passed to a job function.
/// </summary>
[StructLayout(LayoutKind.Explicit, Size = 16)]
7 references | jcant0n, 66 days ago | 1 author, 4 changes
public readonly struct JobArgs
{
    /// <summary>
    /// The index of the current job.
    /// </summary>
    [FieldOffset(0)]
    public readonly uint JobIndex;

    /// <summary>
    /// The ID of the group this job belongs to.
    /// </summary>
    [FieldOffset(4)]
    public readonly uint GroupID;

    /// <summary>
    /// The index of this job within its group.
    /// </summary>
    [FieldOffset(8)]
    public readonly uint GroupIndex;

    /// <summary>
    /// Padding to ensure 16-byte alignment.
    /// </summary>
    [FieldOffset(12)]
    private readonly uint padding;
```

Stealing jobs



CPU usage



CPU

AMD Ryzen 5 3600 6-Core Processor

Auslastung
1%

Geschwindigkeit
3.79 GHz

Prozesse
174

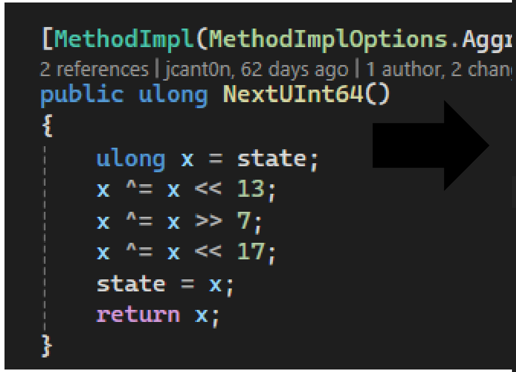
Threads
2231

Handles
72014

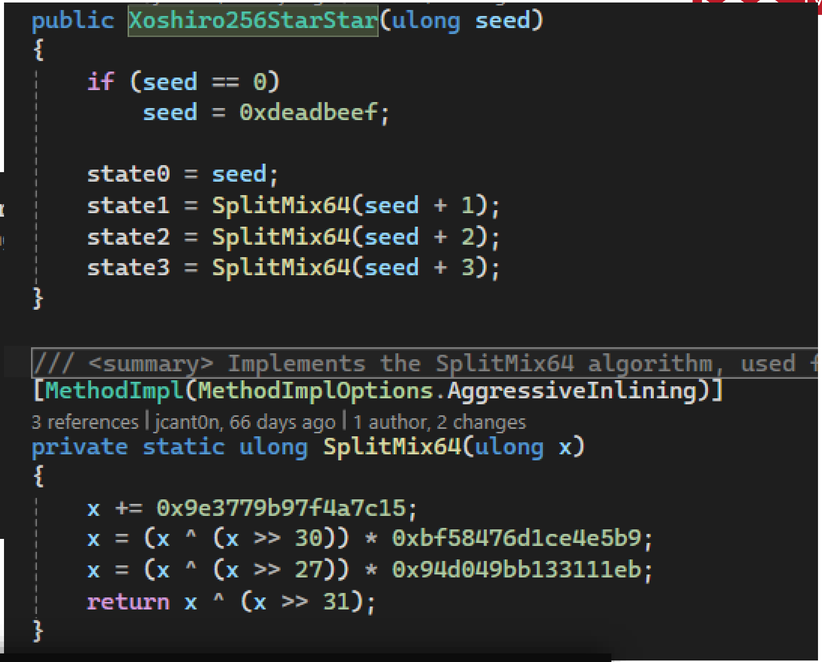
Betriebszeit
0:02:54:23

Basisgeschwindigkeit:	3.60 GHz
Sockets:	1
Kerne:	6
Logische Prozessoren:	12
Virtualisierung:	Deaktiviert
Hyper-V-Unterstützung:	Ja
L1-Cache:	384 KB
L2-Cache:	3.0 MB
L3-Cache:	32.0 MB

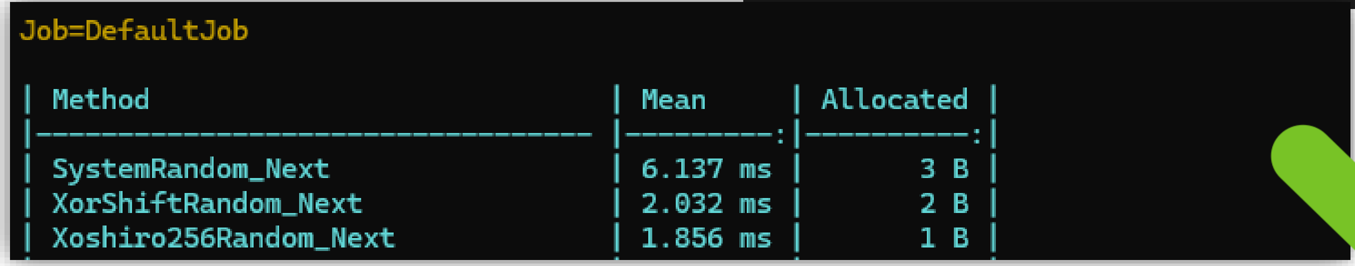
Random



```
[MethodImpl(MethodImplOptions.Aggr
2 references | jcant0n, 62 days ago | 1 author, 2 chan
public ulong NextUInt64()
{
    ulong x = state;
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    state = x;
    return x;
}
```

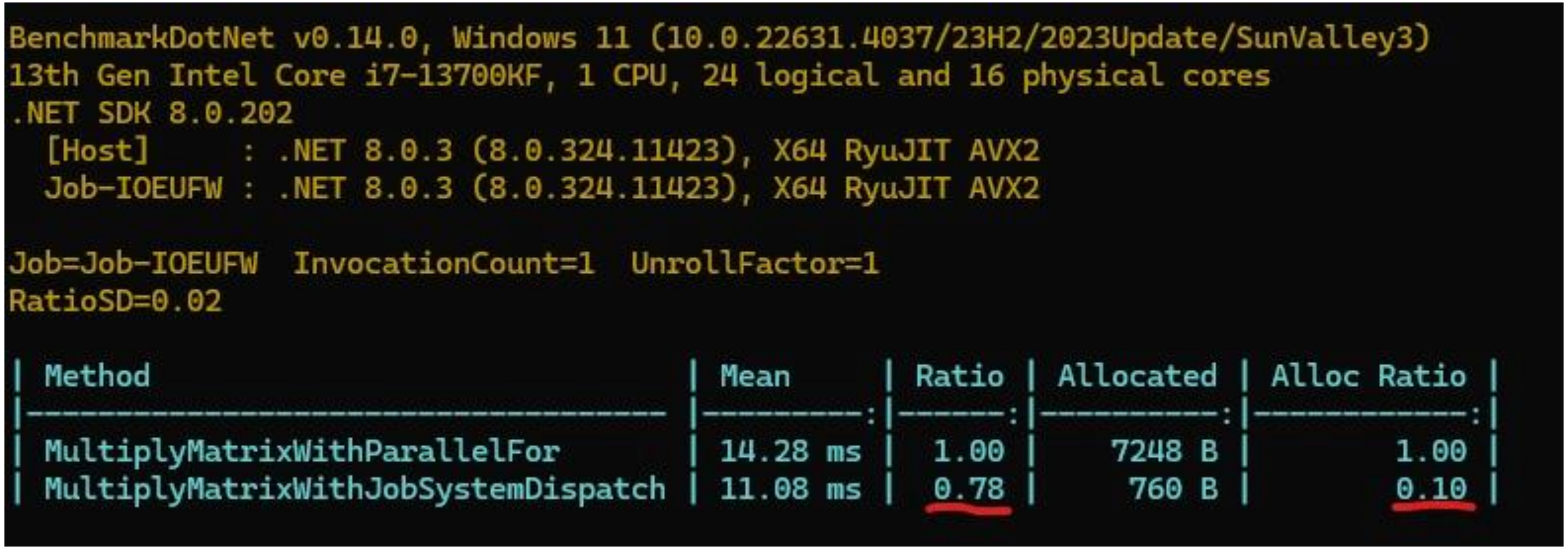


```
public Xoshiro256StarStar(ulong seed)
{
    if (seed == 0)
        seed = 0xdeadbeef;

    state0 = seed;
    state1 = SplitMix64(seed + 1);
    state2 = SplitMix64(seed + 2);
    state3 = SplitMix64(seed + 3);
}

/// <summary> Implements the SplitMix64 algorithm, used f
[MethodImpl(MethodImplOptions.AggressiveInlining)]
3 references | jcant0n, 66 days ago | 1 author, 2 changes
private static ulong SplitMix64(ulong x)
{
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
}
```



```
Job=DefaultJob

| Method                               | Mean      | Allocated |
|------------------------------------- |----------:|----------:|
| SystemRandom_Next                    | 6.137 ms  |       3 B |
| XorShiftRandom_Next                  | 2.032 ms  |       2 B |
| Xoshiro256Random_Next                | 1.856 ms  |       1 B |
```

Final Performance



```
BenchmarkDotNet v0.14.0, Windows 11 (10.0.22631.4037/23H2/2023Update/SunValley3)
13th Gen Intel Core i7-13700KF, 1 CPU, 24 logical and 16 physical cores
.NET SDK 8.0.202
  [Host]     : .NET 8.0.3 (8.0.324.11423), X64 RyuJIT AVX2
  Job-IOEUFW : .NET 8.0.3 (8.0.324.11423), X64 RyuJIT AVX2

Job=Job-IOEUFW  InvocationCount=1  UnrollFactor=1
RatioSD=0.02

| Method                              | Mean     | Ratio | Allocated | Alloc Ratio |
|------------------------------------ |---------:|------:|----------:|------------:|
| MultiplyMatrixWithParallelFor       | 14.28 ms |  1.00 |    7248 B |        1.00 |
| MultiplyMatrixWithJobSystemDispatch | 11.08 ms |  0.78 |     760 B |        0.10 |
```

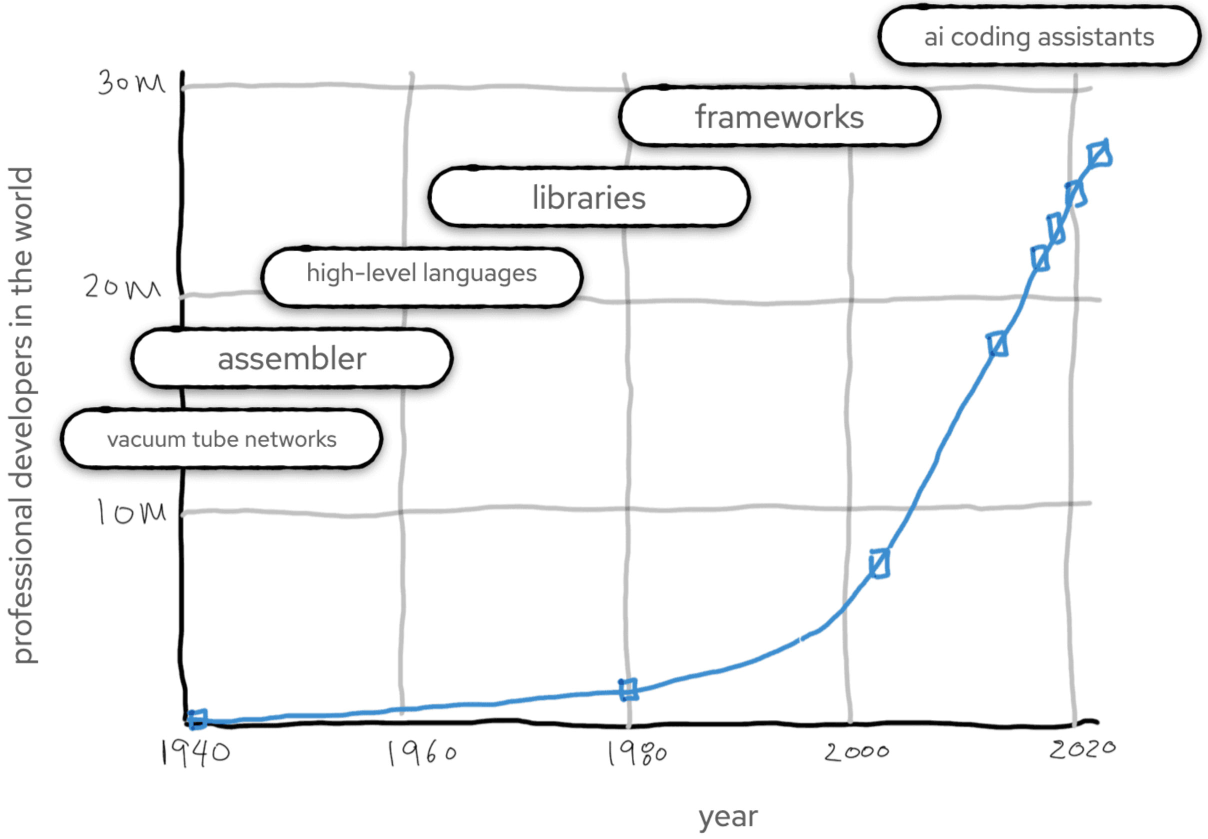
22% más rápido y un 90% menos Alloc

Conclusions

Pros and Cons

POSITIVES ✓	✗ NEGATIVES
Speed up coding	Limited Context Understanding
Enhance Code Naming	It could saturate users
Helpful in brainstorming solutions	Overconfidence in AI
Great to generate performance tests	Data security risks
Excellent for language translation	Overly generic default solutions

Productivity Tool



Questions

jcanton jcant0n

DOT NET MLG

EL EVENTO SOBRE

TECNOLOGÍAS CLOUD, WEB Y DATA

Verisk™ SECTOR ALARM prodware LEMONCODE METADEV plain concepts