**Title:** Benchmark Results & Performance Analysis – Rust vs. C# CLI and Batch Tools

### 📌 1. Introduction

* Purpose of benchmarking
* What is being compared (Rust vs. C# for CLI and batch tools)
* How the benchmarks were executed (tools used, fairness considerations)

### 🧪 2. Benchmark Methodology

* Tools used:
  + hyperfine for execution time
  + PowerShell scripts for memory and CPU profiling
* System specs (CPU, RAM, OS version)
* Dataset: large\_input.csv (~100k rows), consistent input across all tests
* Execution mode: Release builds, 10 runs per tool
* Warmup: 1–3 warmup runs to normalize

### 🔹 3. Scenario: Manual CSV Processor

* Description: filter rows where age > 30, uppercase name
* Rust tool: cli\_tool.rs
* C# tool: CsvProcessorManual.cs
* Results:

| **Metric** | **Rust** | **C#** | **Rust Advantage** |
| --- | --- | --- | --- |
| Time (ms) | 73.6 | 127.8 | 1.74× faster |
| Memory (MB) | 3.49 | 16.34 | ~4.7× less |
| CPU Time (ms) | 15.62 | 46.88 | ~3× less |

* Short analysis paragraph

### 🔸 4. Scenario: Batch Age Grouping Job

* Description: group users by age brackets and count
* Rust tool: batch\_job.rs
* C# tool: BatchJob.cs
* Results:

| **Metric** | **Rust** | **C#** | **Rust Advantage** |
| --- | --- | --- | --- |
| Time (ms) | 74.1 | 270.8 | 3.65× faster |
| Memory (MB) | 2.44 | 2.49 | ≈ equal |
| CPU Time (ms) | 15.62 | 46.88 | ~3× less |

* Short analysis paragraph

### 🔸 5. Scenario: Parallel Age Grouping Job

* Description: same logic, multithreaded with Rayon vs PLINQ
* Rust tool: batch\_job\_parallel.rs
* C# tool: BatchJobParallel.cs
* Results:

| **Metric** | **Rust** | **C#** | **Rust Advantage** |
| --- | --- | --- | --- |
| Time (ms) | 43.9 | 108.5 | 2.47× faster |
| Memory (MB) | 2.44 | 16.96 | ~7× less |
| CPU Time (ms) | 15.62 | 62.5 | ~4× less |

* Short analysis paragraph

### 📊 6. Summary Table (All Results)

| **Project** | **Language** | **Time (ms)** | **Memory (MB)** | **CPU Time (ms)** |
| --- | --- | --- | --- | --- |
| Manual | Rust | 73.6 | 3.49 | 15.62 |
| Manual | C# | 127.8 | 16.34 | 46.88 |
| Batch | Rust | 74.1 | 2.44 | 15.62 |
| Batch | C# | 270.8 | 2.49 | 46.88 |
| Batch Parallel | Rust | 43.9 | 2.44 | 15.62 |
| Batch Parallel | C# | 108.5 | 16.96 | 62.5 |

### 📈 7. Charts (optional but recommended)

* Bar chart: Execution time across all tools
* Bar chart: Memory usage
* Chart: Rust vs. C# speedup per scenario

### 🧠 8. Key Observations

* Rust consistently faster in all cases
* Memory advantages especially visible in multithreaded scenario
* C# has more overhead due to GC, higher base memory usage
* Batch task is the most dramatic difference (~3.6× speedup)

### ✅ 9. Conclusion

* Rust is better suited for high-efficiency batch and CLI tools
* For performance-critical or resource-constrained tools, Rust is preferable
* C# offers convenience but suffers under concurrency and load