Project Proposal: Parallel Monte Carlo Simulation for Estimating π

Team Roster

- **Hasan Hijazi** (ID: 5922)
- **Morhaf Najjar** (ID: 5980)

Problem Statement & Domain

Domain: Monte Carlo Simulations
Problem: Estimating the value of π using a computational Monte Carlo method. The algorithm simulates random point generation within a square and counts how many fall inside a quarter circle.

Why This Problem Has Parallelism Potential

Each point generation and evaluation is **independent**, making this an **embarrassingly parallel** problem. It offers excellent scalability, minimal synchronization, and near-linear speed-up potential on multi-core CPUs.

Sequential Baseline Sketch

- Generate `N` random `(x, y)` points in the range [0, 1].
- Count the number of points where `x² + y² ≤ 1`.
- Estimate π as `π ≈ 4 × (hits / N)`.
- Use `System.nanoTime()` and `Java Flight Recorder` to measure baseline performance.

Parallel Strategy & Risk Analysis

- **Parallelization**: Use `ForkJoinPool` or `ExecutorService` to split the task among available cores.
- Each thread will:
    - Generate a batch of random points using `ThreadLocalRandom`.
    - Calculate its local hit count and return it.
- **Reduction**: Use `LongAdder` to aggregate results efficiently.
- **Risks**:
    - Thread contention in RNG: avoided by using `ThreadLocalRandom`.
    - Load imbalance: addressed through proper chunk sizing.

Data & Metrics

- **Target Speed-up**: ≥ 3× on 8-core CPUs.
- **Metrics**:
    - Speed-up vs. sequential.
    - CPU Utilisation ≥ 85%.
    - Memory Overhead ≤ 2×.
- **Tools**:
    - VisualVM, Java Flight Recorder, CSV + plotting tools.

Timeline

Week	Milestone
1	Proposal submission

2	Sequential implementation + unit tests
3	Parallel implementation
4	Profiling + optimization
5	Report writing
6	Demo prep + peer review