Relatório — Estimativa de π por Monte Carlo (MPI) no Santos Dumont

Autor: Pedro Henrique Mendes Cândido (TechMed Solutions)
Relatório acadêmico — 6 páginas

1. Problema e Relevância

Este projeto implementa uma estimativa de π por Monte Carlo para servir como baseline de computação de alto desempenho (HPC).

A escolha pelo método de Monte Carlo decorre de seu paralelismo naturalmente massivo („embaraçosamente paralelo„), o que facilita a distribuição de carga entre processos, medição de speedup/eficiência e análise de gargalos mínimos de comunicação.

No contexto da TechMed Solutions, o baseline é útil para (i) validar o pipeline de execução local →cluster (Santos Dumont), (ii) exercitar a instrumentação de métricas (tempo total, throughput, speedup), e (iii) estabelecer práticas de reprodutibilidade que, posteriormente, podem ser aplicadas a workloads hospitalares reais (p. ex., processamento de DICOM e análise de séries temporais).

2. Arquitetura e Paralelismo

Modelo de paralelismo: MPI (via mpi4py). Cada processo gera N pontos aleatórios uniformes em [0,1]×[0,1] e conta quantos caem dentro do círculo de raio 1 ($x^2 + y^2 \leq 1$). A agregação dos resultados ocorre por redução (reduce/SUM) no processo raiz.

O custo de comunicação é mínimo e concentrado ao final (uma única redução de inteiros). Organização do repositório (resumo): src/ (main.py, utils.py), scripts/ (build, execução local, submissão SLURM), results/ (logs e métricas), report/ (relatório).

Boas práticas adotadas: separação de I/O e computação; geração sintética de dados; logs identificados por rank; reprodutibilidade com semente.

3. Dados e I/O

Dados: sintéticos, gerados por gerador pseudoaleatório (NumPy). Não há dependências externas de bases.

I/O: mínimo apenas escrita de métricas (CSV) com timestamp, número de processos, amostras totais, estimativa de π, tempo total e throughput.

Essa escolha reduz gargalos de armazenamento e torna mais direta a análise de escalabilidade. Armazenamento no cluster: em execução real no Santos Dumont, recomenda-se usar /scratch para arquivos temporários pesados e preservar /home apenas para código e resultados finais.

4. Metodologia de Experimentos

Ambiente local: execução com 1, 2, 4 e 8 processos (ou mais, quando disponível), mantendo o número de amostras por processo constante (p. ex., 1e6).

Para cada configuração, realizar ao menos 3 repetições e reportar a média e o desvio padrão do tempo total.

Métricas: (i) tempo total por execução; (ii) throughput (pontos/segundo); (iii) speedup = T1 / Tp; (iv) eficiência = speedup / p.

Interpretação: espera-se queda monotônica do tempo com o aumento de processos, até um ponto onde a sobrecarga de coordenação (redução final, latências, disputa por recursos) comece a atenuar os ganhos marginais; a eficiência tipicamente diminui gradualmente à medida que p cresce.

5. Resultados (Modelo de Apresentação)

timestamp	procs	samples_total	pi_est	tempo_s	throughput_pts_s
2025-09-19T17:56:22	4	4000000	3,13946200	0,046386	86232908,61
2025-09-19T17:56:56	1	500000	3,14235200	0,031955	15646768,28
2025-09-19T17:56:57	2	1000000	3,14198000	0,021807	45857430,25
2025-09-19T17:56:57	4	2000000	3,14041800	0,024073	83081847,71
2025-09-24T19:36:17	4	4000000	3,13946200	0,077504	51610134,25

6. Limitações e Próximos Passos

Limitações:

- O problema é CPU-bound com comunicação mínima; não avalia gargalos de I/O, etapa crítica emworkloads reais (p. ex., varredura de milhares de arquivos).
- Em escalas muito grandes, a operação de redução pode impactar a latência total; o ganho marginaltende a diminuir.
- A precisão estatística depende de N; dobrar a precisão requer multiplicar N por ~4 (erro ~ $1/\sqrt{N}$).

Próximos passos:

- Versão GPU (CuPy/CUDA) para medir aceleração massiva em geração e contagem vetorizadas, comparandocom CPU/MPI.
- Implementar variante OpenMP em C para comparação de memória compartilhada.
- Introduzir estágio de I/O mais realista (p. ex., leitura de blocos de pontos pré-gerados), paraestudar throughput de armazenamento.
- Rodar no Santos Dumont com diferentes partições/filas e registrar variação de desempenho(incluindo uso de /scratch).

7. Reprodutibilidade, SLURM e Boas Práticas

Reprodutibilidade: registrar versão de Python, bibliotecas (mpi4py, NumPy), semente do RNG e árvore do repositório; manter um único ponto de entrada para execução (scripts/run_local.sh) e submissão (scripts/job_cpu.slurm).

SLURM: usar sbatch para submissão, squeue para monitorar, e redirecionar logs para results/. Não executar cargas pesadas no nó de login.

Boas práticas: testes com amostras pequenas antes de escalar; validação de argumentos (fail-fast); logs claros com identificação de rank; documentar recursos de SLURM (nós, tarefas, memória, tempo) e a partição utilizada.

8. Conclusão

O experimento de Monte Carlo com MPI atende aos objetivos pedagógicos: demonstra paralelismo, fornece métricas claras para análise de speedup/eficiência
e cria uma base reprodutível para migração ao cluster Santos Dumont. Os resultados esperados mostram boa escalabilidade inicial e custo de comunicação baixo,
com eficiência adequada até um número moderado de processos. O projeto está pronto para ser estendido a cenários mais realistas da TechMed Solutions, incluindo workloads com I/O significativo (p. ex., DICOM) e aceleração por GPU onde houver benefício.