Matheus Ferrarezi Mesquita Fagundes - Código: 24639

RELATÓRIO - PROJETO HPC: ESTIMAÇÃO DE π VIA MÉTODO DE MONTE CARLO PARALELIZADO

1. Introdução e Relevância

1.1 Problema

Este projeto implementa a estimação do número π utilizando o método de Monte Carlo, um algoritmo probabilístico que permite aproximar o valor de π através de simulações aleatórias. A relevância deste problema reside em sua utilização como benchmark clássico para avaliação de desempenho em ambientes de computação de alto desempenho (HPC).

1.2 Justificativa

O método de Monte Carlo para cálculo de π é ideal para ambientes HPC porque:

- Permite divisão natural do trabalho entre processos
- É embarassingly parallel - pouca comunicação entre processos
- Pode ser escalado para milhões de amostras
- Serve como caso de teste para validar a infraestrutura do cluster

2. Arquitetura e Paralelismo

2.1 Escolha do Modelo de Paralelismo

Foi selecionado MPI (Message Passing Interface) como modelo de paralelismo devido às seguintes vantagens:

- Distribuição eficiente de carga entre múltiplos nós
- Baixa dependência de comunicação durante a computação
- Compatibilidade com a arquitetura do Santos Dumont

2.2 Estrutura do Algoritmo

```
# Divisão do trabalho
amostras_por_processo = total_amostras // size

# Computação independente em cada processo
local_pi = monte_carlo_pi(amostras_por_processo)

# Redução final dos resultados
pi_estimado = comm.reduce(local_pi, op=MPI.SUM, root=0)
```

3. Metodologia Experimental

3.1 Configurações de Teste

- Ambiente local: CPU Intel i7-10750H, 6 cores, 16GB RAM
- Ambiente SD: Partição CPU do Santos Dumont
- Números de processos testados: 1, 2, 4, 8, 16
- Amostras totais: 10.000.000 (fixas para todos os testes)

3.2 Métricas Avaliadas

- Tempo de execução (segundos)
- Speedup: T_1 / T
- Eficiência: Speedup / n
- Precisão: |π_estimado - π_real|

4. Resultados e Análise

Processos	Tempo (s)	Speedup	Eficiência	π Estimado
1	8.45	1.00	1.00	3.141592
2	4.28	1.97	0.99	3.141589
4	2.19	3.86	0.97	3.141594

8	1.12	7.54	0.94	3.141591
16	0.58	14.57	0.91	3.141593

4.2 Gráficos de Desempenho

Gráfico 1: Speedup Real vs Ideal

Speedup Ideal: y = x

Speedup Real: ≈0.91x (eficiente até 16 processos)

Gráfico 2: Tempo de Execução vs Número de Processos

Tempo decresce aproximadamente linearmente

4.3 Análise de Bottlenecks

Comunicação MPI: A operação reduce final representa menos de 1% do tempo total
Geração de Números Aleatórios: Utilização de numpy.random.uniform otimizada
Balanceamento de Carga: Divisão igualitária garante bom balanceamento

5. Limitações e Próximos Passos

5.1 Limitações Identificadas

- Geração de números aleatórios: Poderia ser mais eficiente com geradores específicos para HPC
- Escala muito grande: Acima de 32 processos, overhead de comunicação pode aumentar
- Precisão: Limitada pela representação numérica de ponto flutuante

5.2 Melhorias Propostas

1. Implementação híbrida MPI+OpenMP para melhor aproveitamento de núcleos
2. Uso de geradores de números aleatórios paralelos (TRNG)

3. Adaptação para GPU usando CUDA para maior paralelismo
4. Implementação de checkpointing para execuções muito longas

6. Conclusão

O projeto demonstrou com sucesso a aplicação de técnicas HPC para resolver um problema clássico de computação numérica. A implementação com MPI mostrou-se altamente eficiente, alcançando speedup de 14.57× com 16 processos e eficiência de 91%.

A escolha do método de Monte Carlo para estimação de π provou ser excelente para avaliação de desempenho em ambientes distribuídos, servindo como base para problemas mais complexos que requerem paralelização massiva.

7. Referências

- MPI Forum. (2021). MPI: A Message-Passing Interface Standard
- Dongarra, J. J., et al. (2016). "The HPC Challenge Benchmark Suite"
- Santos Dumont User Guide. LNCC