Perfilamento e Análise de Escalabilidade de Código em um Sistema Distribuído com o PaScal Analyzer: Um Estudo de Caso

Felipe H. Santos-da-Silva[1], **Júlio F. Peixoto Gomes**[2], **João B. Fernandes**[2], **Samuel Xavier-de-Souza**[2], **Ítalo A. S. Assis**[1]

[1] Departamento de Engenharias e Tecnologia
Universidade Federal Rural do Semi-Árido (UFERSA)

[2]Departamento de Engenharia de Computação e Automação
Universidade Federal do Rio Grande do Norte (UFRN)

felipe.silva65229@alunos.ufersa.edu.br,

julio.freire.098@ufrn.edu.br,

joao.batista.fernandes.094@ufrn.edu.br,

samuel@dca.ufrn.br, italo.assis@ufersa.edu.br

Resumo. *Este trabalho apresenta uma extensão do PaScal Analyzer, ferramenta para análise de escalabilidade, originalmente focada em aplicações OpenMP, agora adaptada para ambientes distribuídos via instrumentação manual e coleta de dados desacoplada. Aplicado ao benchmark NAS, o método identificou gargalos regionais, indicando seu potencial para análise e otimização em sistemas distribuídos.*

1. Introdução

A análise de escalabilidade é essencial no desenvolvimento de aplicações de alto desempenho (HPC), pois orienta otimizações, identifica gargalos e permite compreender o comportamento da aplicação em diferentes configurações. Apesar da variedade de ferramentas disponíveis, muitas se limitam a análises pontuais, dificultando a avaliação da escalabilidade em aplicações com múltiplas regiões ou configurações. Ferramentas especializadas que automatizam a coleta e análise contínua de métricas tornam-se, assim, fundamentais.

Diversas abordagens já foram propostas: ferramentas por amostragem, como [Adhianto et al. 2010] e [Schulz et al. 2008], exigem calibração e podem gerar dados excessivos ou incompletos; já soluções com instrumentação, como [Knüpfer et al. 2012], [Shende and Malony 2006], [Madsen et al. 2020] e [Boehme et al. 2016], oferecem granularidade, mas geralmente seu uso envolve um processo complexo exigindo configurações específicas. Novas propostas, como [Fan et al. 2024], [Boehme et al. 2021] e [Eberius 2020], buscam menor intrusão, mas com limitações em escopo ou dependências.

O PaScal Analyzer, proposto em [da Silva et al. 2022], introduz uma análise de escalabilidade regional para OpenMP. Este trabalho estende essa abordagem para ambientes distribuídos com MPI, utilizando instrumentação manual e coleta via sockets, sem

dependências externas. Avaliado no Integer Sort (IS), um dos kernels do NAS Parallel Benchmarks (NPB), o método identifica gargalos regionais.

O artigo está organizado da seguinte forma: A Seção 2 apresenta a ferramenta; A Seção 3 detalha o objeto de estudo; A Seção 4 discute os testes e resultados; e a Seção 5 traz as conclusões.

2. PaScal Analyzer

O *PaScal Analyzer* é uma ferramenta para avaliação de escalabilidade em aplicações paralelas em C/C++, com suporte a ambientes de memória compartilhada (OpenMP) e distribuída (MPI). Para códigos OpenMP compilados com `GCC`, a instrumentação pode ser automática ou manual; já para MPI, é exclusivamente manual.

A ferramenta permite configurar parâmetros como número de núcleos, processos, entradas e repetições, além de delimitar regiões específicas do código. Os dados são agregados em um arquivo JSON, facilitando a visualização e comparação dos resultados.

Este trabalho apresenta a extensão da ferramenta para suporte a aplicações MPI. Cada processo mede localmente as regiões instrumentadas, armazena os dados em buffer e os envia, ao final da execução, via socket TCP/IP ao *PaScal Analyzer*. Foram adicionados os argumentos `--ranks`, que define a quantidade de processos executados (ex.: `4,2,1`), e `--mpi`, que configura o comando de execução (ex.: `mpirun`).

3. NAS Parallel Benchmark IS

O NAS Parallel Benchmarks (NPB) é amplamente usado para avaliar o desempenho de sistemas paralelos [Bailey et al. 1994]. Neste trabalho, utilizamos o kernel Integer Sort (IS), que ordena inteiros pseudoaleatórios com o algoritmo counting sort, adaptado para execução paralela em memória distribuída.

Removemos a restrição original de mínimo de quatro processos, permitindo testes com apenas um. Também integramos o cabeçalho `pascalops.h`, com macros `pascal_start()` e `pascal_stop()`, para marcar regiões críticas e coletar dados de tempo de execução.

4. Experimentos e Resultados

Os testes foram realizados no supercomputador NPAD (UFRN), com 8 nós contendo 2 CPUs Intel Xeon E5-2698v3 (16 núcleos, 2.3 GHz) e 128 GB de RAM. Foram utilizadas as classes A, B e C do benchmark IS, representando crescentes tamanhos de problema. Classes superiores foram evitadas por limitações de memória por nó.

A Tabela 1 resume os parâmetros das classes utilizadas, que impactam diretamente na carga computacional:

- **Tamanho do problema** (2^N): número de chaves ordenadas.
- **Número de chaves** (N^k): volume total de dados.
- **Intervalo das chaves** (I^k): define os baldes para ordenação.
- **Tamanho do bucket** (N^b): capacidade dos baldes internos.

Table 1. Parâmetros das classes do benchmark IS (NAS).

Classe	2^N	N^k	I^k	N^b
A	2^{18}	262.144	2^{11}	2^{15}
B	2^{22}	4.194.304	2^{15}	2^{19}
C	2^{27}	134.217.728	2^{20}	2^{24}

4.1. Avaliação com o PaScal Analyzer

Executamos o IS com 1, 2, 4 e 8 processos MPI, com 10 repetições por configuração. A Tabela 2 mostra o *speedup* e a eficiência global.

Table 2. Speedup e eficiência global por classe e número de processos.

Classe	Speedup			Eficiência (%)		
	2	4	8	2	4	8
A	1.65	1.85	1.82	82.4	46.3	22.7
B	1.82	3.43	5.10	90.9	85.8	63.8
C	1.91	3.78	6.39	95.7	94.4	79.9

A escalabilidade melhora com o aumento do problema. A classe A perde eficiência com 8 processos devido ao overhead em regiões pouco custosas. Para investigar, analisamos o laço principal da função `rank`, instrumentado conforme a Listagem 1.

```
#include "pascalops.h"
pascal_start(0);
for (iteration = 1; iteration <= MAX_ITERATIONS; iteration++)
    rank(iteration);
pascal_stop(0);
```

Listing 1. Trecho instrumentado da região 0.

A Tabela 3 mostra o excelente desempenho dessa região, com quase *speedup* linear mesmo com 8 processos. Isso evidencia que os gargalos globais vêm de outras regiões menos paralelizáveis. Os resultados reforçam a utilidade do *PaScal Analyzer* na detecção de gargalos regionais e na orientação de otimizações direcionadas.

Table 3. Speedup e eficiência da região 0.

Classe	Speedup			Eficiência (%)		
	2	4	8	2	4	8
A	1.94	4.71	7.94	97.0	117.8	99.2
B	1.94	4.25	7.92	97.2	106.3	99.0
C	1.96	4.30	7.99	97.9	107.4	99.9

5. Conclusão

Este trabalho apresentou uma extensão do PaScal Analyzer para ambientes distribuídos, demonstrando sua capacidade de identificar gargalos regionais com sobrecarga reduzida.

Os resultados com o benchmark Integer Sort da NASA Advanced Supercomputing evidenciam a efetividade da abordagem. Trabalhos futuros incluem suporte automatizado à instrumentação MPI e medição de intrusão da nossa proposta.

References

Adhianto, L., Banerjee, S., Fagan, M., Krentel, M., Marin, G., Mellor-Crummey, J., and Tallent, N. R. (2010). Hpctoolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 22(6):685–701.

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., Simon, H. D., Venkatakrishnan, V., and Weeratunga, S. (1994). The nas parallel benchmarks. Technical Report RNR-94-007, NASA Ames Research Center, Moffett Field, CA, USA. NASA Technical Report.

Boehme, D., Aschwanden, P., Pearce, O., Weiss, K., and LeGendre, M. (2021). Ubiquitous performance analysis. In Chamberlain, B. L., Varbanescu, A.-L., Ltaief, H., and Luszczek, P., editors, *High Performance Computing*, pages 431–449, Cham. Springer International Publishing.

Boehme, D., Gamblin, T., Beckingsale, D., Bremer, P.-T., Gimenez, A., LeGendre, M., Pearce, O., and Schulz, M. (2016). Caliper: performance introspection for hpc software stacks. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 550–560. IEEE.

da Silva, V. R. G., da Silva, A. B. N., Valderrama, C., Manneback, P., and Xavier-de Souza, S. (2022). A minimally intrusive approach for automatic assessment of parallel performance scalability of shared-memory hpc applications. *Electronics*, 11(5).

Eberius, D. (2020). *Providing Insight into the Performance of Distributed Applications Through Low-Level Metrics*. PhD thesis, University of Tennessee.

Fan, K., Kesavan, S., Petruzza, S., and Kumar, S. (2024). Tinyprof: Towards continuous performance introspection through scalable parallel i/o. In *Proceedings of the International Supercomputing Conference (ISC)*, pages 1–12. IEEE.

Knüpfer, A., Rössel, C., Mey, D. a., Biersdorff, S., Diethelm, K., Eschweiler, D., Geimer, M., Gerndt, M., Lorenz, D., Malony, A., et al. (2012). Score-p: A joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir. In *Tools for High Performance Computing 2011: Proceedings of the 5th International Workshop on Parallel Tools for High Performance Computing, September 2011, ZIH, Dresden*, pages 79–91. Springer.

Madsen, J. R., Awan, M. G., Brunie, H., Deslippe, J., Gayatri, R., Oliker, L., Wang, Y., Yang, C., and Williams, S. (2020). Timemory: modular performance analysis for hpc. In *High Performance Computing: 35th International Conference, ISC High Performance 2020, Frankfurt/Main, Germany, June 22–25, 2020, Proceedings 35*, pages 434–452. Springer.

Schulz, M., Galarowicz, J., Maghrak, D., Hachfeld, W., Montoya, D., and Cranford, S. (2008). Open— speedshop: An open source infrastructure for parallel performance analysis. *Scientific Programming*, 16.

Shende, S. S. and Malony, A. D. (2006). The tau parallel performance system. *The International Journal of High Performance Computing Applications*, 20(2):287–311.