



Representação de árvores binárias completas como arrays: Uma otimização focada em localidade de referência de processadores

Kanna Waltrick Diaz | kanna.w02@aluno.ifsc.edu.br¹

Letícia Vieira | leiticia.v02@aluno.ifsc.edu.br²

Ramon Hugo de Souza | ramon.hugo@ifsc.edu.br³

EDITAL Nº 12/2025/PROPPI/DAE

Trabalho de Pesquisa

DT 4 - Inovação Tecnológica, Processos Produtivos e Futuro das Tecnologias

Resumo:

Este estudo apresenta uma primeira medição experimental, utilizando-se do framework para lidar com Sistemas Operacionais com Heurísticas de Inteligência Artificial (S.Op.H.I.A.), em uma implementação de árvore binária completa. Este framework, em desenvolvimento por esta equipe, visa a obtenção automática de informações do processador, utilizando-as para tomada de decisão sobre valores ótimos a serem utilizados em representação de uma estrutura, a qual representa uma intenção, conceito advindo de Processamento de Linguagem Natural (N.L.P, do inglês Natural Language Processing), área de Inteligência Artificial (I.A.), normalmente representada como uma árvore, otimizada então no formato de array ou conjunto de arrays. O objetivo principal é utilizar-se do conceito de localidade de referência em processadores para otimizar o acesso a dados. Então, nesta abstração, têm-se uma árvore representada como um array, sem a utilização de endereços de memória como ponteiros, visando também a redução de memória utilizada para aproveitar ao máximo os ganhos obtidos em tempo de acesso, quando considera-se localidade de referência. Como esperado, os ganhos em gravação são muito maiores que os em leitura, sendo a leitura a com maior influência temporal no processo de análise de NLP. Porém, considerando sistemas que necessitam de processamento de altíssima quantidade de dados, constata-se que qualquer redução temporal tem um efeito considerável. Reduzir-se 10ms para 5ms parece pouco, mas se o efeito de tal for a redução de 7 dias para 3.5 dias, fica claro o ganho obtido em análise de quantidade massiva de dados.

Palavras-chave: árvores binárias; otimização de estruturas de dados; localidade de referência.

¹ Estudante do curso de Análise e Desenvolvimento de Sistemas do IFSC campus São José.

² Estudante do curso de Análise e Desenvolvimento de Sistemas do IFSC campus São José.

³ Professor da área de Sistemas Distribuídos do IFSC campus Florianópolis.

Introdução

Este estudo trata de um experimento visando demonstrar os ganhos espaciais e temporais que podem ser obtidos com a mudança de paradigma da representação de árvores binárias, considerando tal estrutura como apresentada em Skiena (2009) e Garnier e Taylor (2009). Árvores estas as quais são naturalmente representadas como estruturas contendo dois ponteiros, estes apontados para a “mesma estrutura”, mais um ponteiro apontando para os referidos dados, vide código abaixo:

```
1: struct Node {  
2:     struct Node * left;  
3:     struct Node * right;  
4:     void * data;  
5: };
```

Nota-se que a referida estrutura possui tamanho fixo, diretamente relacionado com a arquitetura do sistema operacional utilizado, possuindo então 24 bytes quando da consideração de sistemas os quais utilizam 64 bits para endereçamento de memória. Porém, apesar de ter-se estrutura de tamanho fixo, a utilização de ponteiros não permite, teoricamente, utilizar-se do ganho que pode ser obtido ao carregar áreas de memória próximas em cache, isto é, a localidade de referência, como definida em Stallings (2010), a qual otimiza por padrão a utilização de arrays.

Como objetivo principal, então, tem-se a utilização de implementações de localidade de referência em processadores para otimizar o tempo de acesso a dados quando representados no formato de árvores binárias.

Quanto às características de indissociabilidade entre ensino, pesquisa e extensão, têm-se a aplicação dos conceitos outrora estudados em disciplinas as quais englobam conceitos de algoritmos, estruturas de dados e sistemas operacionais, estes presentes em disciplinas constantes no currículo do curso de Análise e Desenvolvimento de Sistemas, cursado pelas duas principais autoras deste trabalho, então aplicados em conjunto para resolução do problema proposto.

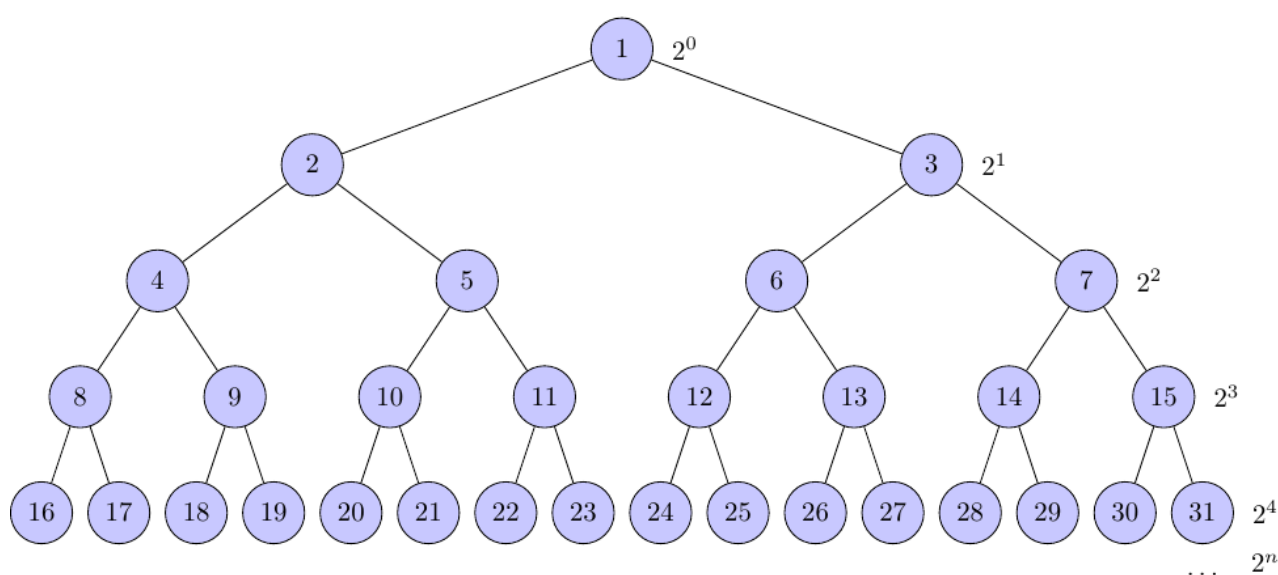
Fundamentação teórica

Para o levantamento dos dados aqui apresentados, foram utilizadas árvores binárias completas, não a título utilitário, mas para fins de experimentação. Ou seja, para uma árvore referida como de profundidade “n” têm-se então o número de nós respeitando

a função $f(n) = \sum_{i=0}^{n-1} 2^i = 2^n - 1 \mid n \geq 1$, como demonstrado na Figura 1.

Para tal experimentação, considerou-se a representação por meio de arrays, de maneira a melhor explorar-se localidade de referência, para demonstração dos ganhos os quais podem ser obtidos tanto espacial quanto temporalmente.

Figura 1 – Árvore Binária Completa de Profundidade “n” e número de nós por “camada”



Fonte: elaborada pelos autores.

Este experimento fundamenta a implementação do framework para lidar-se com Sistemas Operacionais com Heurísticas de Inteligência Artificial (S.Op.H.I.A.), o qual encontra-se em desenvolvimento junto ao projeto de pesquisa PISJE4615-2025, realizado junto ao IFSC São José.

Procedimentos metodológicos

Representaram-se as mesmas estruturas, inicialmente no formato de árvores padrão, com ponteiros, medindo-se tempos: (1) desde alocação de memória, quando de criação, até o preenchimento de todos os dados; e (2) de leitura, considerado espaço de memória já alocado, sendo este o caso mais comum, e em seguida adaptou-se estas estruturas de árvores ao formato de array, em um modelo simplificado, tentando otimizar o ganho quando da utilização de localidade de referência. Como dado padrão, foi utilizada uma palavra de tamanho médio da língua portuguesa, de cinco letras, aproximada para seis, considerando a otimização de utilização de espaço de memória, já que tal estrutura utilizaria o mesmo espaço de memória com cinco ou seis letras, vide código abaixo:

```
1: struct Node2 {  
2:     int16_t left;  
3:     int16_t right;  
4:     char info[6];  
5: };
```

Para obtenção de dados mais relevantes, realizaram-se diversos experimentos, computando média e desvio padrão de cada experimento selecionado. Fornecendo informação probabilística relativa a tais dados. Os experimentos foram todos realizados sob o processador Intel(R) Core(TM) i7-7500U CPU@2.70GHz, detalhado no Quadro 1, ressaltando que o processador consta com 2 núcleos e 2 threads por núcleo, com seus caches: L1d: 64 KiB (2 instâncias); L1i: 64 KiB (2 instâncias); L2: 512 KiB (2 instâncias); e L3: 4 MiB. Tais experimentos foram executados em um sistema operacional Linux, distribuição Fedora 24 x64. Os programas foram todos compilados em C e rodados com a diretiva “nice -n -20” em modo texto (init 3).

Quadro 1 – Detalhamento de Caches da CPU Intel Core i7-7500U CPU @ 2.70GHz

Cache ID 0: -Level: 1 -Type: Data Cache -Sets: 64 -System Coherency Line Size: 64 bytes -Physical Line partitions: 1 -Ways of associativity: 8 -Total Size: 32 kb -Is fully associative: false -Is Self Initializing: true	Cache ID 1: -Level: 1 -Type: Instruction Cache -Sets: 64 -System Coherency Line Size: 64 bytes -Physical Line partitions: 1 -Ways of associativity: 8 -Total Size: 32 kb -Is fully associative: false -Is Self Initializing: true	Cache ID 2: -Level: 2 -Type: Unified Cache -Sets: 1024 -System Coherency Line Size: 64 bytes -Physical Line partitions: 1 -Ways of associativity: 4 -Total Size: 256 kb -Is fully associative: false -Is Self Initializing: true	Cache ID 3: -Level: 3 -Type: Unified Cache -Sets: 4096 -System Coherency Line Size: 64 bytes -Physical Line partitions: 1 -Ways of associativity: 16 -Total Size: 4096 kb -Is fully associative: false -Is Self Initializing: true
--	---	--	--

Fonte: elaborada pelos autores.

Resultados e discussões

Os resultados dos experimentos realizados, com média (μ) e desvio padrão (σ), calculados para 20 medições de 10000 iterações, são apresentados numericamente junto às Tabelas 1 e 2, e respectivas Figuras 2 e 3, onde na Tabela 1 tem-se a comparação temporal de criação da estrutura, mais a população dos dados enquanto na Tabela 2 tem-se a comparação temporal somente de leitura de dados.

Apesar do crescimento linear quando de criação e população de dados em ambos os casos, tem-se que, neste caso, os tempos para representação como array chegam a ser aproximadamente $\frac{1}{4}$ dos tempos da representação com ponteiros. Já no caso de leitura tem-se um comportamento com ganho inicialmente ameno, porém chegando a aproximadamente $\frac{1}{2}$ quando atingido certo tamanho estrutural.

O ganho obtido é, portanto, indício suficiente para definir-se que, em especial quando considerando-se casos de análise de dados extremos, pode-se realmente vir a obter-se ganhos temporais significativos.

Tabela 1 – Resultados experimentais de criação e população em ms.

# Elementos	Ponteiros		Array	
	μ	σ	μ	σ
2^1-1	0.3	0.4702	0.3	0.4702
2^2-1	0.9	0.3078	0.45	0.5104
2^3-1	2.7	0.4702	0.8	0.4104

2^4-1	6.5	0.513	1.3	0.4702
2^5-1	14.15	0.3663	2.65	0.4894
2^6-1	29.55	0.5104	5.15	0.3663
2^7-1	61.55	0.9445	18.6	34.6978
2^8-1	123.7	0.4702	20.85	0.4894
2^9-1	249.6	0.5026	41.45	0.5104
$2^{10}-1$	509.65	1.6631	84.15	0.3663
$2^{11}-1$	1070.05	82.3334	163.65	0.4894
$2^{12}-1$	2428.25	211.14	324.2	0.4104
$2^{13}-1$	5416.15	73.8279	690.9	78.3292
$2^{14}-1$	11658.6	82.9263	1412.35	143.6887
$2^{15}-1$	24257.45	154.9932	2817.7	145.5943
$2^{16}-1$	59214.3	769.7585	5668.3	84.012
$2^{17}-1$	204038.6	9374.7164	11396	172.4211
$2^{18}-1$	501124	14789.0435	23115.65	316.6152
$2^{19}-1$	1080093	25576.3302	46404.25	378.6706
$2^{20}-1$	2261714.65	66622.3105	94315.25	402.8588

Fonte: elaborada pelos autores.

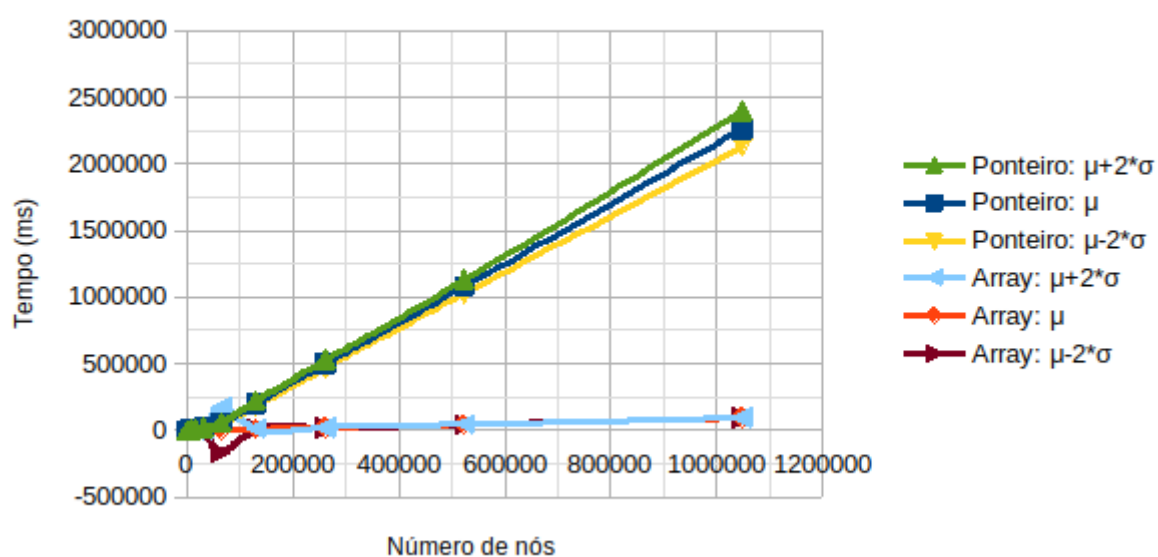
Tabela 2 – Resultados experimentais de leitura em *ms*.

# Elementos	Ponteiros		Array	
	μ	σ	μ	σ
2^1-1	0.1	0.3078	0.15	0.3663
2^2-1	0.45	0.5104	0.4	0.5026
2^3-1	0.95	0.2236	0.85	0.3663
2^4-1	2.1	0.3078	1.9	0.3078
2^5-1	4.35	0.4894	3.8	0.4104
2^6-1	8.8	0.4104	7.7	0.4702
2^7-1	17.85	0.3663	15.95	1.2763

2^8-1	35.85	0.3663	32.85	3.2163
2^9-1	72.95	0.2236	62.25	1.916
$2^{10}-1$	176.2	21.1376	126.7	5.0482
$2^{11}-1$	329.85	41.3945	279.9	37.6855
$2^{12}-1$	678.05	78.7751	504.9	11.2245
$2^{13}-1$	1410.1	157.3899	1133.4	140.2109
$2^{14}-1$	2853.95	273.123	2311.6	225.3744
$2^{15}-1$	5736.4	22.3663	4616.2	186.1258
$2^{16}-1$	12134.05	210.3475	9256.65	287.0832
$2^{17}-1$	27833.4	571.2068	18674.35	171.36
$2^{18}-1$	67063.5	3634.6955	36945.5	399.1212
$2^{19}-1$	146136.9	8566.8994	74546.4	440.8879
$2^{20}-1$	290996.9	17958.1112	149333.7	477.5137

Fonte: elaborada pelos autores.

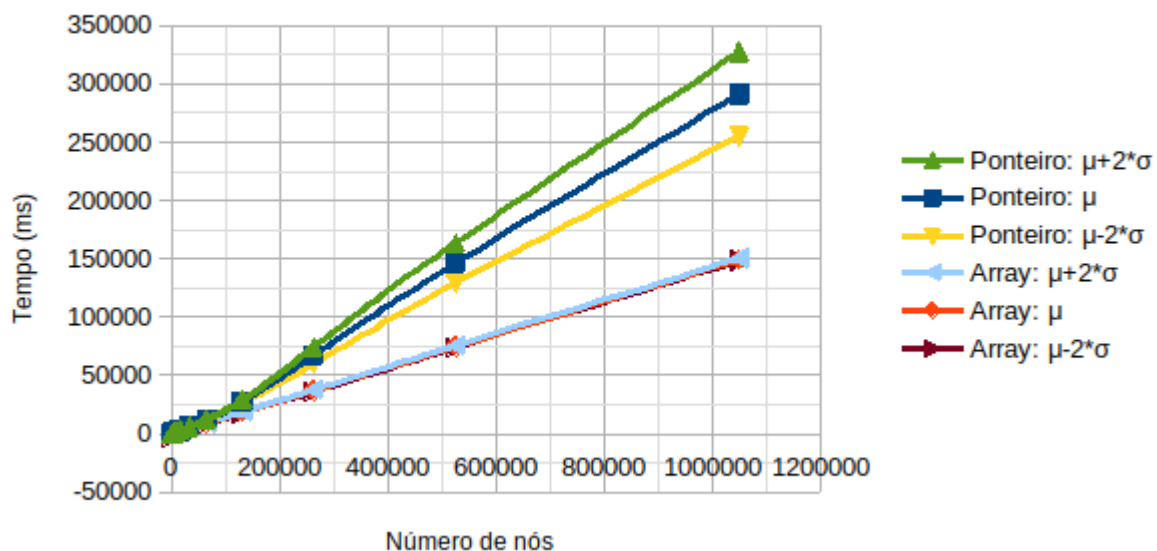
Figura 2 - Resultados experimentais de criação e população em $\mu \pm 2\sigma$ (95.45%)



Fonte: elaborada pelos autores.

Juntamente com o framework S.Op.H.I.A., um novo modelo de processamento de linguagem natural (PLN), do conceito como apresentado em Nadkarni, Ohno-Machado e Chapman (2011), começa, a partir desta otimização, a ser desenvolvido junto ao IFSC, sendo tal referido como Lógica Estruturada para Tratamento de Informação Contextual e Inferência Avançada (L.E.T.I.C.I.A.), onde espera-se ainda apresentar os passos de automatização de aglutinação de “árvores lineares”, as intenções em sua forma padrão, de maneira a se otimizar estruturalmente o modelo PLN até o ponto de seleção de estruturas utilizando os máximos ganhos possíveis com base no processador utilizado.

Figura 3 - Resultados experimentais de leitura em $\mu \pm 2\sigma$ (95.45%)



Fonte: elaborada pelos autores.

Este é, portanto, um primeiro experimento para embasar o framework S.Op.H.I.A., ainda em desenvolvimento, que será utilizado como base para a arquitetura desta nova ferramenta PLN, o modelo de representação L.E.T.I.C.I.A.



Considerações finais

A demonstração de ganho temporal foi atingida, porém ainda é necessário que, após finalizar-se a parte de automação do framework, se crie uma nova metodologia de aglutinação de estruturas lineares em árvores mais complexas, estas a serem então utilizadas na ferramenta PLN a qual será desenvolvida após a finalização do framework.

Aos alunos envolvidos no processo de desenvolvimento, introduziram-se conceitos de inteligência artificial e de sistemas operacionais, trabalhando com otimização de estrutura de dados, de maneira prática, demonstrando que complexidade computacional de algoritmos é realmente um passo fundamental quando de aplicação de soluções em casos mais complexos. Ou seja, apenas resolver um problema pode não ser o suficiente para o que se necessita.

Com o conhecimento desenvolvido a partir deste projeto, espera-se dar continuidade em projeto para criação da ferramenta PLN do IFSC.

Referência ao fomento recebido

Foi obtido, através do edital nº 12/2025/PROPPI/DAE, do Programa de Apoio ao Desenvolvimento de Projetos que Contemplem a Pesquisa como Princípio Educativo do IFSC, auxílio via bolsa a um dos discentes associado ao projeto, este estando classificado como em Índice de Vulnerabilidade Social.

Referências

GARNIER, Rowan; TAYLOR, John (2009). **Discrete Mathematics: Proofs, Structures and Applications**, Third Edition. CRC Press. p. 620. ISBN 978-1-4398-1280-8, 2009.

NADKARNI, Prakash M.; OHNO-MACHADO, Lucila; CHAPMAN, Wendy W., Natural language processing: an introduction, **Journal of the American Medical Informatics Association**, volume 18, p. 544–551, setembro de 2011, <https://doi.org/10.1136/amiajnl-2011-000464>.

SKIENA, Steven S. **The Algorithm Design Manual**. Springer Science & Business Media. p. 77. ISBN 978-1-84800-070-4, 2009.

STALLINGS, William. **Computer organization and architecture : designing for performance** (8th ed.). Upper Saddle River, NJ: Prentice Hall. ISBN 9780136073734. OCLC 268788976, 2010.