Programação Concorrente e Distribuída - Atividade 4

Pedro Leyria de Oliveira RA: 133778

Retomando o problema

- Rede social representada por grafo. Três tipos de vértices:
  - **Tipo 0:** Não conhece o rumor. Pode mudar de tipo.
  - **Tipo 1:** Acredita no rumor. Pode influenciar nós tipo 0. Não muda de tipo.
  - **Tipo 2:** Não acredita no rumor. Pode influenciar nós tipo 0. Não muda de tipo.
- Em um grafo G, para cada vizinho u de um vértice v, tem-se α aleatório entre 0 e 1; e β calculado com a seguinte fórmula: $\beta = gr(u)/\Delta(G)$. Onde $\Delta(G)$ é o maior grau da rede.
- Se $\alpha < \beta$, então u é considerado influente em v.
- Um vértice é escolhido ao acaso entre os vizinho influente de v, para determinar o seu novo estado, caso ele seja tipo 0.
- Implementação inicial em Python.
  - Simulação em passos discretos.
  - Tempo de execução elevado.
  - GIL

Solução proposta

Duas soluções

Cython

- Expansão da linguagem Python com recursos da linguagem C.
- Permite um uso limitado de paralelismo (liberar o GIL).

C + OpenMP

- Linguagem C normalmente mais rápida do que Python.
- Combinada com a API OpenMP para usar paralelismo.

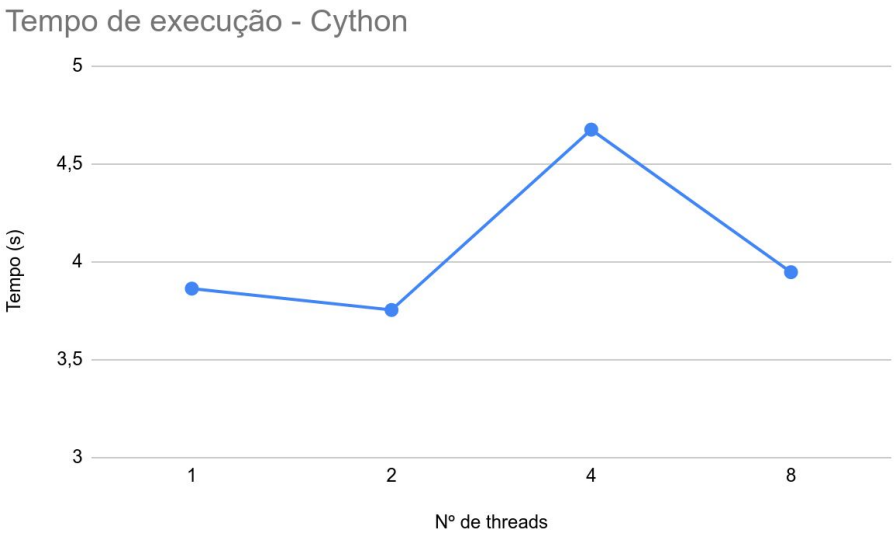
- Implementação das duas opções.
- Comparação de desempenho.

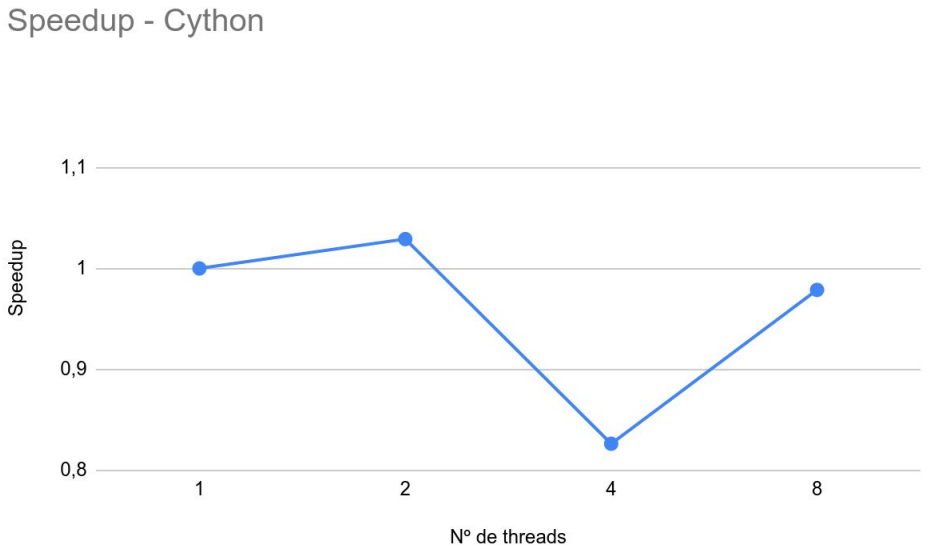
Demonstração dos programa

Resultados

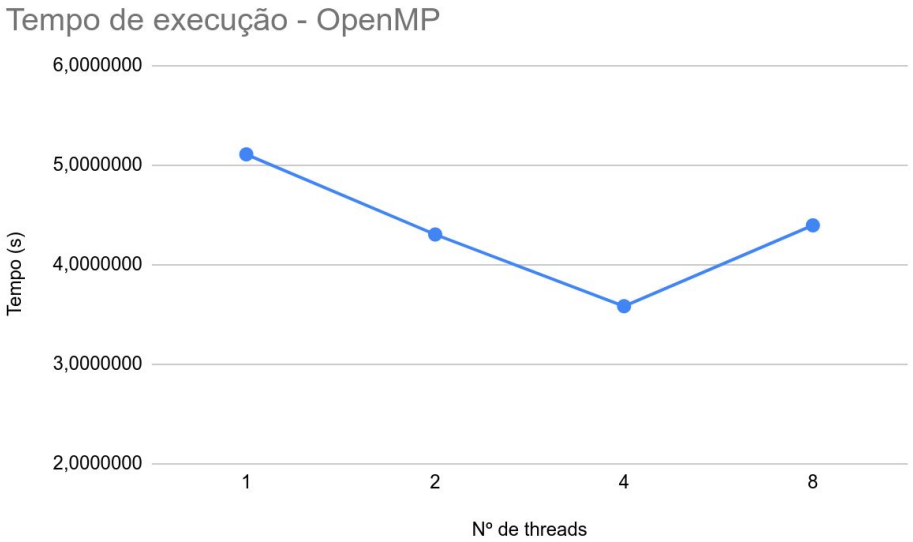
	Tempo de execução (segundos)								
	Python (nº threads)					C (nº threads)			
Teste	**Original**	**1**	**2**	**4**	**8**	**1**	**2**	**4**	**8**
1	95,6599315	4,7182703	4,3412276	2,1470568	3,6904640	7,4810648	3,6620951	5,8794526	5,6187147
2	116,3436347	1,9673909	3,8133664	2,7402036	6,1111795	5,1588709	4,4835336	3,0536913	3,4668024
3	111,5387815	7,2100270	3,8575196	6,1118251	6,6849991	5,9718610	4,2921880	2,1109459	4,8752281
4	214,0537244	3,2475150	3,9385612	8,9701793	4,1758006	4,1984498	5,8521203	3,9123962	2,9552156
5	199,9496812	3,2388483	5,7052413	4,2413784	4,3408842	3,3921524	2,9744583	3,6365178	2,6856284
6	91,5607546	3,6085183	4,0649389	3,1967323	3,3783018	6,3319181	5,0016564	3,2948323	3,1279477
7	255,1299930	2,9915237	4,2154614	5,9550305	4,0253576	2,4105945	5,4965274	2,9704554	4,0286934
8	144,4207812	2,1744834	5,7777066	3,8630781	3,4165379	2,9698569	8,1252643	5,6966013	5,5490723
9	267,0192688	2,7902842	2,4935108	4,4141768	3,6863046	15,8779603	2,7490017	4,2655739	1,9024094
10	130,6381282	3,1799306	2,7330954	5,6864915	2,5973140	8,1418952	3,7062847	1,5423518	4,6227036
11	114,7050858	4,0722662	2,8677273	3,6304140	3,4061430	3,3499709	3,3526703	3,9157228	2,8798111
12	147,8133448	3,5053741	2,2116518	4,1518286	3,1166371	4,2486472	2,3972943	5,6122917	13,0904293
13	368,1227366	4,2185280	1,6670343	5,4934097	4,8645046	7,9757245	3,2429819	3,1763049	2,8830504
14	157,2935876	6,5481742	4,0547970	7,1617618	4,1907813	4,3137645	2,9853991	2,5822980	4,6494249
15	74,2347325	3,8353273	3,0983756	6,9209498	3,3417754	4,3347048	5,1608155	3,8237527	2,9407659
16	124,2549141	3,8268504	3,0833042	2,8088384	3,7849717	2,9719152	3,1430088	2,9682570	4,4091458
17	114,2665859	2,9604032	4,1866321	3,9242367	1,9686060	4,9759404	7,0766285	3,2965585	5,5091853
18	88,5805428	4,3611908	2,6822428	3,1250963	5,5354703	3,5601181	6,0227063	2,8638262	3,4435552
19	172,2703280	3,2517727	4,0652167	4,5676054	4,2019488	2,3409160	3,8897562	3,7949678	4,1264917
20	113,2600040	5,5349686	6,2000175	4,3892069	2,4052987	2,1449956	2,4251146	3,2285802	5,1118005
Média	155,0558271	3,8620824	3,7528814	4,6749750	3,9461640	5,1075660	4,3019753	3,5812689	4,3938038
Speedup		1,0000000	1,0290979	0,8261183	0,9786928	1,0000000	1,1872607	1,4261889	1,1624475
Eficiência		1,0000000	0,5145489	0,2065296	0,1223366	1,0000000	0,5936303	0,3565472	0,1453059

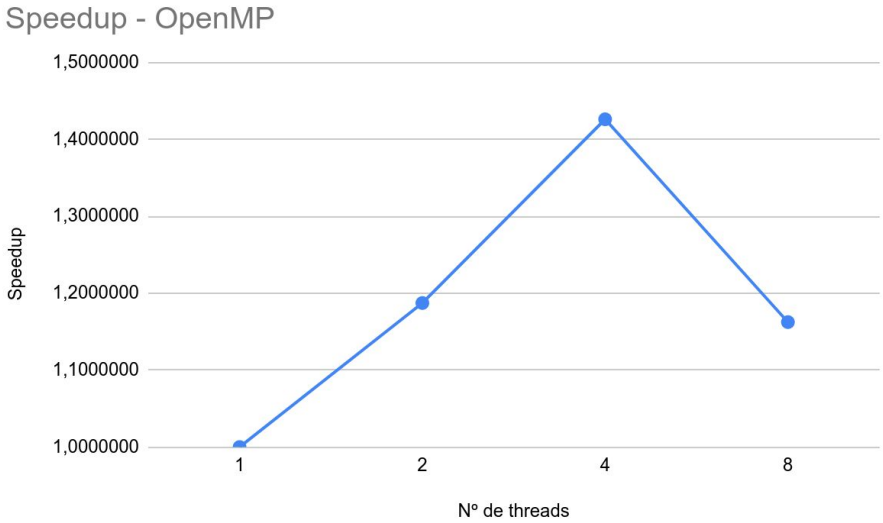
Cython


Tempo de execução - Cython
5
4,5
4
3,5
3
Tempo (s)
1
2
4
8
Nº de threads


Speedup - Cython
1,1
1
0,9
0,8
Speedup
1
2
4
8
Nº de threads

C + OpenMP


Tempo de execução - OpenMP
6,0000000
5,0000000
4,0000000
3,0000000
2,0000000
Tempo (s)
1
2
4
8
Nº de threads


Speedup - OpenMP
1,5000000
1,4000000
1,3000000
1,2000000
1,1000000
1,0000000
Speedup
1
2
4
8
Nº de threads

Conclusão

Conclusão

- Ambas as opções são melhores que a versão original
- Cython: 4 e 8 trheads *aumentaram* o tempo de execução
  - Paralelismo não compensa o overhead de criar/destruir threads (junto com a interface com C)
  - Ganho de desempenho com 2 threads
- C com OpenMP
  - Ganho de desempenho com 2 e 4 threads. Perda com 8 threads
  - Melhor que Cython com 1, 2 e 4 threads. Pior com 8 threads.
- Speedup baixo
  - Seção paralelizável pequena
  - Possível adaptação de outras seções para execução multithreaded

Referências

- https://igraph.org/
- https://cmake.org/
- https://www.openmp.org/
- https://cython.org/

Obrigado!