Comparação de algoritmos de ordenação

Lucas Parzianello

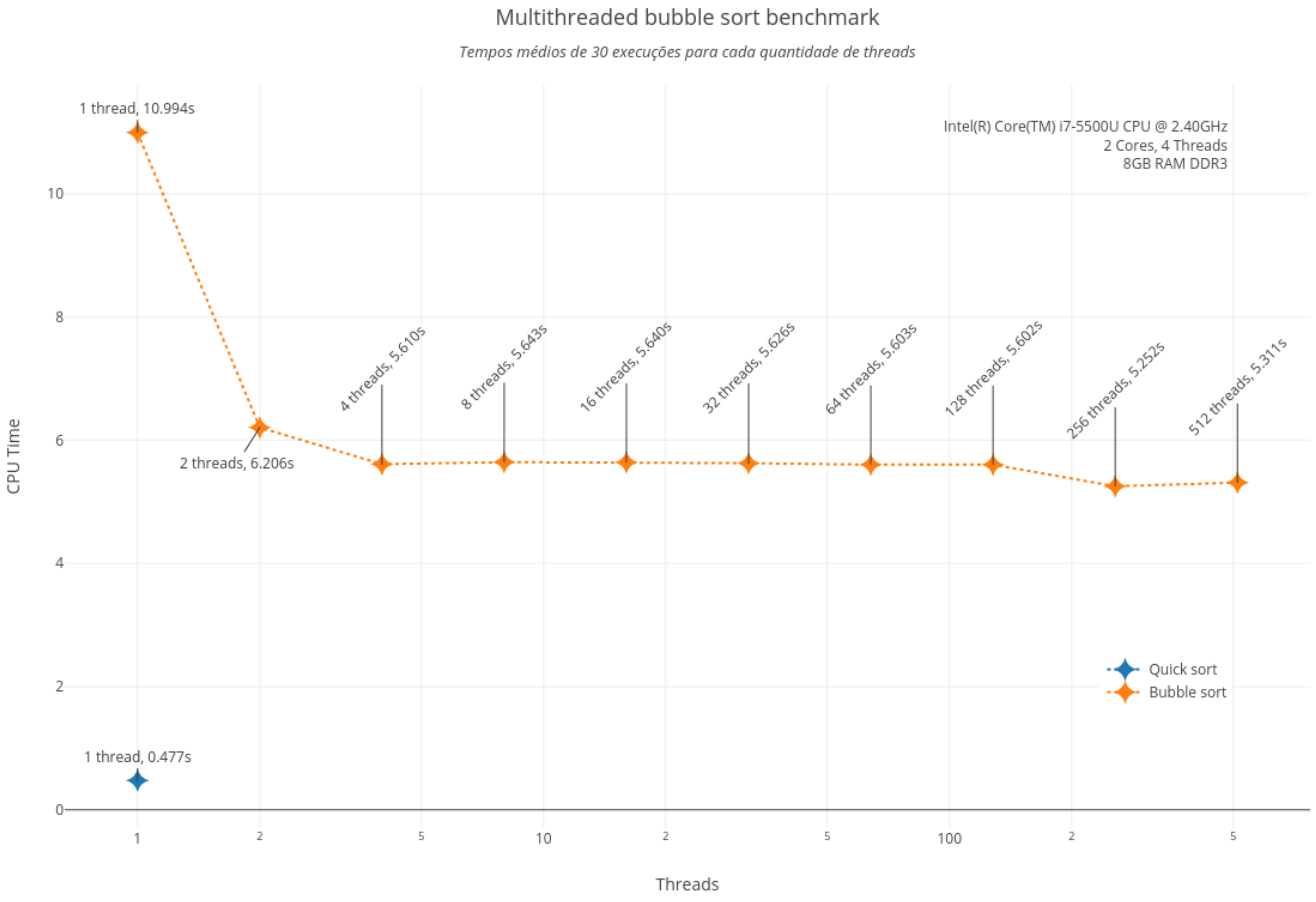
Bubble sort multithreaded

Algoritmo de ordenação bubble sort executado sobre 1000 vetores de 1000 elementos aleatórios cada. Código original do experimento disponível em `results/20180323200947/code`.

Resultados do *benchmark*

Valores brutos (CSV).

Valores em gráfico:



Speedup

Considerando:

$$S = \frac{T_{\text{serial}}}{T_{\text{paralelo}}}$$

Threads	1	2	4	8	16	32	64	128	256	512
Tempo médio	10.994	6.206	5.610	5.643	5.640	5.626	5.603	5.602	5.252	5.311
Speedup	-	1.772	1.96	1.948	1.949	1.954	1.962	1.963	2.093	2.07

Eficiência

Considerando:

$$E = \frac{S}{p} = \frac{\left(\frac{T_{\text{serial}}}{T_{\text{paralelo}}}\right)}{p} = \frac{T_{\text{serial}}}{p \cdot T_{\text{paralelo}}}$$

Threads	1	2	4	8	16	32	64	128	256	512
Tempo médio	10.994	6.206	5.610	5.643	5.640	5.626	5.603	5.602	5.252	5.311
Eficiência	1.000	0.886	0.49	0.244	0.122	0.061	0.031	0.015	0.008	0.004

Código utilizado

```
// arredondamento de 3 casas
rd = (v) => Math.round(v*1000)/1000;

let tserial = 10.994;   // tempo serial
tmedios = [             // tempos medios
    '6.206',
    '5.610',
    '5.643',
    '5.640',
    '5.626',
    '5.603',
    '5.602',
    '5.252',
    '5.311' ]

spd = [];
efs = [];

// calculo dos speedups e eficiencias
tmedios.map((m,i) => {
    let nthreads = 2**(i+1);
    let speedup = tserial / m;
    let eficiencia = speedup / nthreads
    spd.push( rd(speedup) )
    efs.push( rd(eficiencia) )
});

console.log('Speedups:', spd);
console.log('Eficiencias:', efs);
```