

Párhuzamos Algoritmusok

Buborékrendezés

Kórád György (ZF440N)

2023. május 19.

Tartalomjegyzék

1.	A feladat leírása	2
2.	Kapott eredmények	3
3.	Konklúzió	5









1. A feladat leírása

A feladat célja egy 100000 elemű tömb rendezése szekvenciálisan, majd szákezeléssel. Végül a kapott eredményeket összehasonlítjuk és az eredmények alapján eldöntjük, melyik módszer a leggyorsabb. A buborékredezés egy egyszerű algoritmus, hatékonysága rosszabb, mint a más összehasonlításos rendezési algoritmusoké. Az átlagos és legrosszabb esetben is $O(n^2)$ időkomplexitással rendelkezik, ahol "n" a tömb mérete.

Mivel a szálak mind csak a kapott darabját rendezik a tömbnek, így a programok min egyikében implementáltam egy összefésülő rendezést is. Ezt a fő szál hajtja végre, ezzel garantálva az eredeti tömb rendezettségét.

```
void bubbleSort(int arr[])
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N - 1; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

Buborékredező algoritmus

Hardware Information			
Aspire E5-573G V3.72			
		Intel i5-4210U (4) @ 2.7GHz	
		Intel Haswell-ULT	
		NVIDIA GeForce 940M	
		1974MiB / 7866MiB (25%)	

A tesztek futtatásához használt hardware

Az elkészült programok indulást követően bekérlik a felhasználótól a tömb méretét, és a szálak számát, így különböző problémamérettel is lehet vizsgálatokat végezni. Az OpenMPI-vel készült program csak a tömb méretét kéri, mivel itt paraméterként kell megadni a szálak számát a programnak.

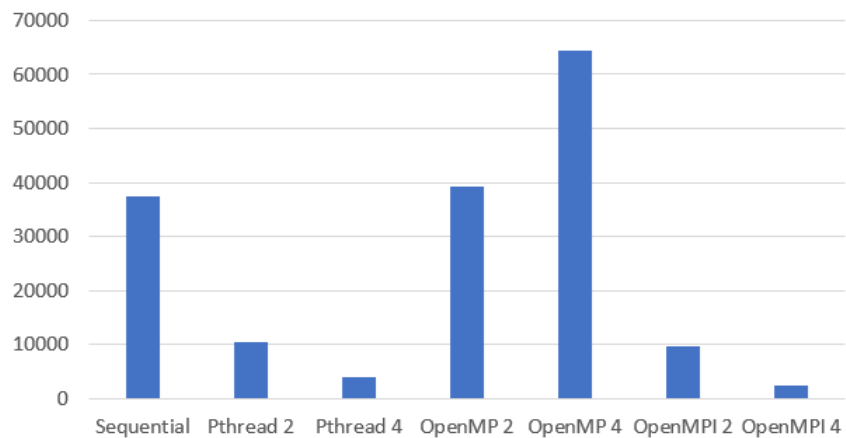
2. Kapott eredmények

Az algoritmust a szálkezelő technológiák használatával kettő illetve négy szálon futtattam, majd az így kapott eredményeket egy file-ban tároltam. Itt a tömb 100000 elemet kapott.

TECH	TIME
Sequential	37492
Pthread 2	10506
Pthread 4	4066
OpenMP 2	39204
OpenMP 4	64318
OpenMPI 2	9697
OpenMPI 4	2465

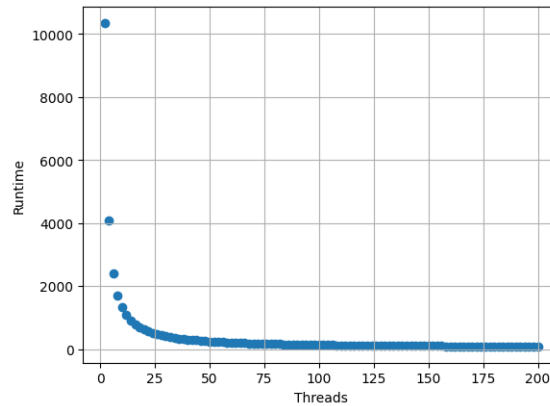
Futási idők

Az OpenMP adta a legrosszabb eredményt, illetve minél több szálát használ, az eredmény úgy romlik. A másik két technológia viszont látványosan felgyorsította az algoritmust.



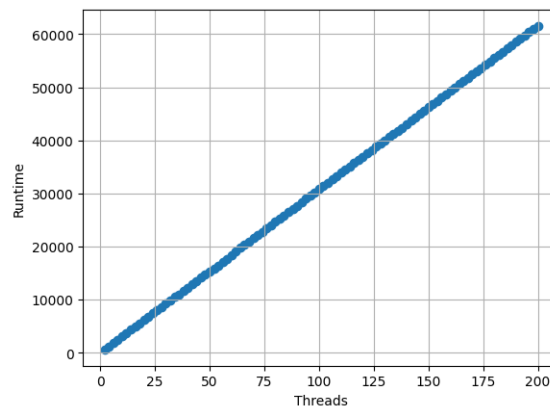
Futási idők diagramja

A OpenMp-t illetve a pthread-et használó programokat úgy módosítottam, hogy a tömb elemeit 10000-re állítottam, a szálak számát pedig kettőre, majd minden alkalommal kettővel növeltem a szálakat, egészen amíg el nem érte a kétszázat. A kapott eredményekről készült garfikonok teljesen eltérő eredményt adtak.



Pthread szálak növelésekor

A pthread-et használó program szálak növelésére egyre jobb eredményeket adott, viszont 50 szál után a javulás már minimális.



3. Konklúzió

TECH	TIME	TASK COMPLETION
Sequential	37492	0,00%
Pthread 2	10506	-71,98%
Pthread 4	4066	-89,16%
OpenMP 2	39204	4,57%
OpenMP 4	64318	71,55%
OpenMPI 2	9697	-74,14%
OpenMPI 4	2465	-93,43%

Eredmények javulása százalékban

Összességében elmondható, hogy az OpenMP volt a leglassabb, és véleményem szerint a használata is ennek volt a legegyszerűbb. Különös módon több szál használva rosszabb időket fut, ami a másik két technológia esetén az ellenkező eredményt adja.

A Message Passing Interface (MPI) látható gyorsulást mutat több szál esetében, viszont ennél érdemes megjegyezni, hogy nem a program forráskódjában adjuk meg a szálak számát, hanem futtatási paraméterként.

```
mpirun --oversubscribe <szálak> <program>
```

Második helyen végzett a pthread-et alkalmazó program, ennél a szálak számát növelve egyre jobb eredményeket kaptam. Több szál használva ez volt a leggyorsabb, viszont a szálak egy bizonyos száma után romlottak a kapott eredmények.