**Documentatie**

1. Comparati performanta pentru fiecare caz – secvential versus paralel si variantele paralele intre ele.

Vom lua ca metrica de evaluare a tipurilor de paralelism de la implementarea C++ static pe 16 thread-uri. Mai jos am precizat motivele pentru care Java si C++ dynamic sunt misleading.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 10 10 3 | 10000 10000 5 | 10 10000 5 | 10000 10 5 |
| SECVENTIAL | 0.0165 | 67.5087 | 6.6424 | 15.5287 |
| LINII | 0.35267 | 11.6727 | 2.71976 | 5.73911 |
| COLOANE | 0.33687 | 13.3684 | 3.61141 | 23.8722 |
| BLOC | 0.2954 | 11.2649 | 3.03771 | 5.80705 |
| DELTA\_LINIAR | 0.35773 | 12.0912 | 2.92795 | 5.75108 |
| DELTA\_CICLIC | 0.33012 | 12.8019 | 3.3351 | 22.9108 |

Contrar asteptarilor, timpul secvential este de cele

1. Comparati timpii de executie obtinuti cu implementarea Java versus implementarea C++.

Timpii din C++ (static) sunt intre ~2-10 ori mai rapizi decat cei din Java. Acest lucru este evident avand in vedere ca variantele C++ au fost compilate in machine code, pe cand codul Java este rulat pe JVM (Java Virtual Machine). Mereu un cod scris bine intr-un limbaj low-level va fii mai rapid decat unul scris in high-level.

De asemenea se observa diferente mult mai logice la timpii din C++ pe numar de thread-uri (mai multe thread-uri => timpi mai mici), pe cand la Java sunt diferente mici iar cateodata se intampla fix inversul. Motivul imi este neclar, dar fiind un limbaj mai high-level, garabage collection-ul si alte feateruri ce C++ nu le are s-ar putea sa ingreuneze programul.

1. Comparati cele doua variante pentru implementarea C++.

Din puncta de vedere al timpului se poate observa ca implementarea statica este mai rapida in mai toate tipurile de paralelism cu exceptia DELTA\_CICLIC. Acest lucru se datoreaza probabil faptului ca in implementarea dinamica de fiecare data cand adaugam un element nou in matrice, spre exemplu la citire, exista sansa sa trebuiasca ca vectorul sa fie marit dynamic. Motivul pentru care DELTA\_CICLIC este mai rapid in implementarea dinamica imi este neclar, insa cred ca ar avea de aface cu faptul ca matricea este storata pe **heap**, lucru care ajuta cand accesam elemnte din memorie ce nu se afla una langa alta.

Cu toate acestea o metrica care nu este masurata este memoria consumata, care in cazul implementarii statice este mereu ~4.000.000 bytes pe cand in cea dinamica bineinteles depinde de la test la test.