Laborator 1 - Documentație

Java:

Tip matrice	**Nr threads**	**Timp executie**
N=M=10 n=m=3	secvential	2118820 ns
	4 - vertical (coloane)	1825080 ns
	4 - orizontal (randuri)	2180290 ns
N=M=1000 n=m=5	secvential	46710090 ns
	2 - vertical	49192880 ns
	2 - orizontal	49408040 ns
	4 - vertical	48031710 ns
	4 - orizontal	49260000 ns
	8 - vertical	50390710 ns
	8 - orizontal	48599460 ns
	16 - vertical	47700000 ns
	16 - orizontal	48268790 ns
N=10 M=10000 n=m=5	secvential	12195580 ns
	2 - vertical	12223230 ns

	2 - orizontal	11896430 ns
	4 - vertical	11773170 ns
	4 - orizontal	13013210 ns
	8 - vertical	11717550 ns
	8 - orizontal	13245780 ns
	16 - vertical	13403150 ns
	16 - orizontal	13227480 ns
N=10000 M=10 n=m=5	secvential	11976970 ns
	2 - vertical	12139880 ns
	2 - orizontal	11103390 ns
	4 - vertical	11545690 ns
	4 - orizontal	11422450 ns
	8 - vertical	11117660 ns
	8 - orizontal	12462280 ns
	16 - vertical	12436910 ns
	16 - orizontal	11725450 ns
N=10000 M=10000	secvential	4220760970 ns

n=m=5	2 - vertical	3728890480 ns
	2 - orizontal	4264404510 ns
	4 - vertical	4769773230 ns
	4 - orizontal	4416053670 ns
	8 - vertical	4292154940 ns
	8 - orizontal	4631702150 ns
	16 - vertical	4669261610 ns
	16 - orizontal	4179678930 ns

C++:

Tip matrice	**Tip alocare**	**Nr threads**	**Timp executie**
N=M=10 n=m=3	Static Secvential	1	2300 ns
	Dinamic Secvential	1	2500 ns
	Static	4	359200 ns - vertical 321700 ns - orizontal
	dinamic	4	284900 ns - vertical

			245000 ns - orizontal
N=M=1000 n=m=5	static	1	76485500 ns
		2	51996000 ns
		4	24991300 ns
		8	16378800 ns
		16	13754500 ns
	dinamic	1	62992700 ns
		2	51831000 ns
		4	33102000 ns
		8	28869000 ns
		16	18228300 ns
N=10 M=10000 n=m=5	static	1	5306200 ns
		2	3309500 ns
		4	2680800 ns
		8	1751200 ns
		16	1759000 ns
	dinamic	1	6938100 ns

		2	4375200 ns
		4	3311700 ns
		8	1999500 ns
		16	1873800 ns
N=10000 M=10 n=m=5	static	1	5857500 ns
		2	3656600 ns
		4	2247300 ns
		8	1749000 ns
		16	2470400 ns
	dinamic	1	6212000 ns
		2	5805400 ns
		4	2371000 ns
		8	2563100 ns
		16	1923100 ns
N=10000 M=10000 n=m=5	static	1	5393436400 ns
		2	2952795600 ns
		4	1770938200 ns

		8	1148167400 ns
		16	922118400 ns
	dinamic	1	6571103800 ns
		2	3578097900 ns
		4	2053187000 ns
		8	1398568200 ns
		16	1074778700 ns

Analiză:

- Comparați performanța pentru fiecare caz - secvențial versus paralel și variantele paralele între ele.
- Comparați timpii de execuție obținuți cu implementarea Java versus implementarea C++.
- Comparați cele două variante pentru implementarea C++.

În cazul Java:

- atât pentru dimensiuni mici ale matricei (N=M=10 sau N=M=1000), cât și pentru dimensiuni mari (N=M=10000), rezolvarea secvențială este aproximativ egală cu rezolvarea utilizând thread-uri, atât prin metoda verticală, cât și metoda orizontală, indiferent de câte thread-uri folosim.

În cazul C++:

- atât pentru dimensiuni mici ale matricei (N=M=10 sau N=M=1000), cât și pentru dimensiuni mari (N=M=10000), diferența dintre modelul static și cel dinamic este foarte puțin semnificativă, modelul static fiind mai eficient decât cel dinamic, iar, în ambele cazuri (static și dinamic), creșterea numărului de thread-uri face și mai eficientă rezolvarea problemei. De exemplu: în cazul în

care avem matricea de dimensiuni N=M=10000, modelul dinamic secvențial are timpul de execuție aproximativ 6.500.000.000 (6,5 miliarde ns), iar, folosind 16 thread-uri, ajungem la o performanță de 1.000.000.000 (1 miliard ns), adică de 6 ori mai eficient.

Diferența dintre implementarea Java și implementarea C++:

- implementarea C++ este mai eficientă decât implementarea Java, având, în cazul N=10 M=10000, secvențial Java, 10.000.000 (10 milioane ns), iar în același caz în C++, avem o performanță de două ori mai eficientă, 5.000.000 (5 milioane ns). Ajungând la cazul în care matricea este de N=M=10000, folosind 16 thread-uri, implementarea C++ este de 4 ori mai eficientă decât implementarea Java.