2015

Verslag opdracht 1

VICTOR VAN DE RIET & ROB LOGTENBERG

Opdracht 1.1

Aantal getallen	**Tijd**									
25000	216	216	75	75	75	74	73	73	75	75
50000	851	841	290	291	291	290	289	290	289	292
100000	3362	3341	1167	1163	1158	1160	1166	1187	1162	1163
200000	13417	13515	4707	4665	4644	4678	4644	4670	4751	4684
400000	53869	53759	18867	18730	18770	18922	18710	18729	18789	18794
800000	215942	75337	75418	75241	75171	75191	75377	75341	75319	75375

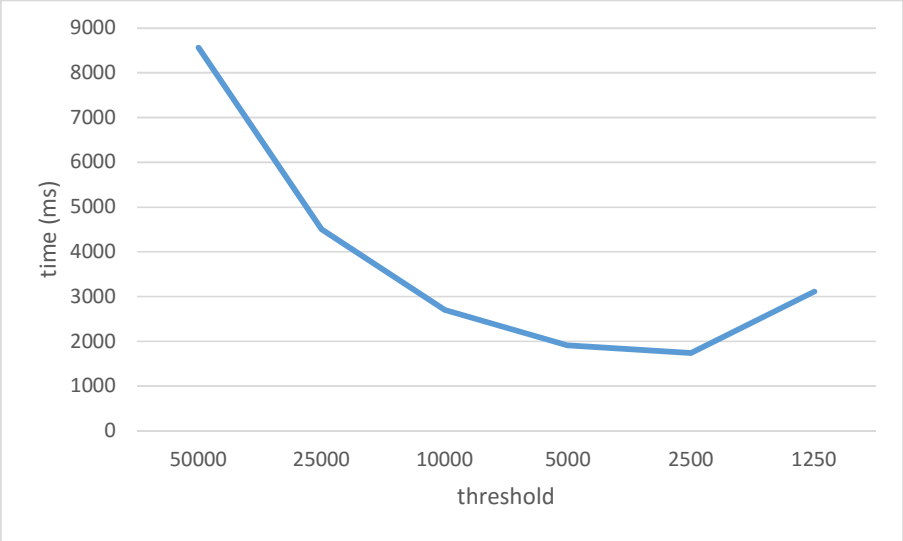
Opdracht 1.2

Aantal getallen	**Tijd**									
25000	78	95	23	22	22	26	20	20	20	19
50000	238	222	77	78	77	78	77	78	77	77
100000	847	860	300	302	298	298	298	299	299	298
200000	3417	3391	1187	1185	1200	1198	1197	1204	1190	1184
400000	13562	13503	4730	4704	4679	4679	4685	4677	4703	4676
800000	54098	54101	18744	18733	18667	18754	18787	18717	18707	18752

Resultaten

De resultaten van opdracht 1.2 zijn meer als twee keer zo snel vergeleken met die van opdracht een. Dit is te verklaren doordat de array door de helft wordt gesplitst en beide helften tegelijk worden gesorteerd.

Opdracht 1.3


9000
8000
7000
6000
5000
4000
3000
2000
1000
0
time (ms)
50000
25000
10000
5000
2500
1250
threshold

Opdracht 1.4

1. In ons geval blijkt dat 2500 als drempelwaarde het beste werkt. Zoals ook in de grafiek te zien is doet het programma er weer langer over zodra we naar een drempelwaarde van 1250 gaan.
2. Waarschijnlijk is dit te verklaren door het feit dat het systeem nu heel veel threads moet aanmaken om een (relatief) kleine hoeveelheid getallen te sorteren. Ook moeten al deze kleinere lijstjes nu weer dankzij mergesort worden samengevoegd.
3. Door het werk te verdelen over meerdere threads, kunnen ze ieder een eigen ding doen, naast elkaar. Waardoor het totale proces sneller is. Wanneer je alles gaat opsplitsen is de processor drukker bezig met het aanmaken van nieuwe threads dan dat hij daadwerkelijk gaat sorteren.

Systeem

We hebben de testen gedaan op een windows systeem. In dit systeem zit een i5 processor met 12 gb 1600 mhz intern geheugen.