SP3 Report

Author:
jxs190058 Jie Su
pxc190029 Pengchao Cai

Table 1

T (=16)	Take 0	Take 3	Take 4	Take 6	Take 0	Take 3	Take 4	Take 6
Size (n)	Time (mSec)	Time (mSec)	Time (mSec)	Time (mSec)	Memory (MB)	Memory (MB)	Memory (MB)	Memory (MB)
16M	INF*	696	521	517	N/A	187/648	187/648	187/648
32M	INF	1415	1097	1064	N/A	373/648	373/1260	373/1260
64M	INF	2812	2220	2331	N/A	739/1384	736/1024	736/1024
128M	INF	5664	4732	4546	N/A	1471/2470	1471/4068	1471/3280
256M	INF	11907	9682	9159	N/A	2935/4068	2936/4068	2935/4068
512M	N/A*	N/A	N/A	N/A	OME*	OME	OME	OME

* N/A : Not Applicable
* INF: Infinity Time
* OME: Out of Memory Error

From table 1, we can see that, for the same size of array, average running time take 0 > take 3 > take 4 > take 6. So, overall, take 6 performs the best, meaning the optimization is effective.
Take 0 is the base version of merge sort. It has no optimization; thus it gives the worst performance. In the test, it is not even able to sort 16M size of array as it takes more than 1 hour to finish. Therefore we marked it as INF (infinity time).
In addition, memory used doubles as the array size grows twice each time.
The platform we are testing on equips i7-6700 processor and 16GB memory. Out of Memory error occurs with size >= 512M.

Table 2

N (=16M)	Take 4	Take 6	Take 4	Take 6
Threshold	Time (mSec)	Time (mSec)	Memory (MB)	Memory (MB)
8	578	540	187/348	187/348
16	530	517	187/348	187/348
32	536	536	187/348	187/348
64	531	623	187/348	187/348
128	662	751	187/348	187/348
256	784	948	187/348	187/348

From table 2, we can see that threshold can affect overall running time of merge sort. From 8 - 16, running time decreases; 16 - 256, running time increases. Therefore, setting threshold = 16 gives us the best performance for both Take 4 and Take 6.