INFO6205

PROGRAM STRUCTURES AND ALGORITHMS

ASSIGNMENT 5 - Parallel Sorting

Problem Statement

Your task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

1. A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.
2. Recursion depth or the number of available threads. Using this determination, you might decide on an ideal number (t) of separate threads (stick to powers of 2) and arrange for that number of partitions to be parallelized (by preventing recursion after the depth of lg t is reached).
3. An appropriate combination of these.

You must prepare a report that shows the results of your experiments and draws a conclusion (or more) about the efficacy of this method of parallelizing sort. Your experiments should involve sorting arrays of sufficient size for the parallel sort to make a difference. You should run with many different array sizes (they must be sufficiently large to make parallel sorting worthwhile, obviously) and different cutoff schemes.

Conclusion

1. The optimal cut off value for switching from parallel sorting of partitions to system sort is approximately 20% of the size of the array.
2. System Sort with no parallelism will lead to the worst performance as seen in the graphs. Having more threads that parallelly sort the partitions of the array will help improve the performance.
3. As we increase the parallelism the performance increases but the gain in performance after 3 levels (8 threads) is not very significant.

The best way to implement a parallel sort algorithm based on these results is to have a cut off of 20% of the size of the array after which we switch to system sort and have 3 or 4 levels of parallelism.

Solution

In order to find out the optimal value for cut off at which we need to switch from sorting parallelly to using system sort I employed the experiments described below. Looking at the experiments below I was able to also judge how the number of threads or the height of parallelism would affect the performance of this algorithm.

Experiment 1:

For this experiment I consider a fixed size of array, a fixed level of height and vary the cut off value between 1% of the array size and 49% of the array size.

Now I repeat this experiment for different levels of height and moreover the experiment is repeated for different sizes of arrays.

Experiment 2:

From conducting experiment 1 we have found out that an optimal cut off value is around 20%. (See conclusion) Hence we fix the cut off value at 20%.

Now I fix the size of the array and vary the level of parallelism to judge its effect on the performance.

We repeat the experiment for different sizes of array to check for consistency.

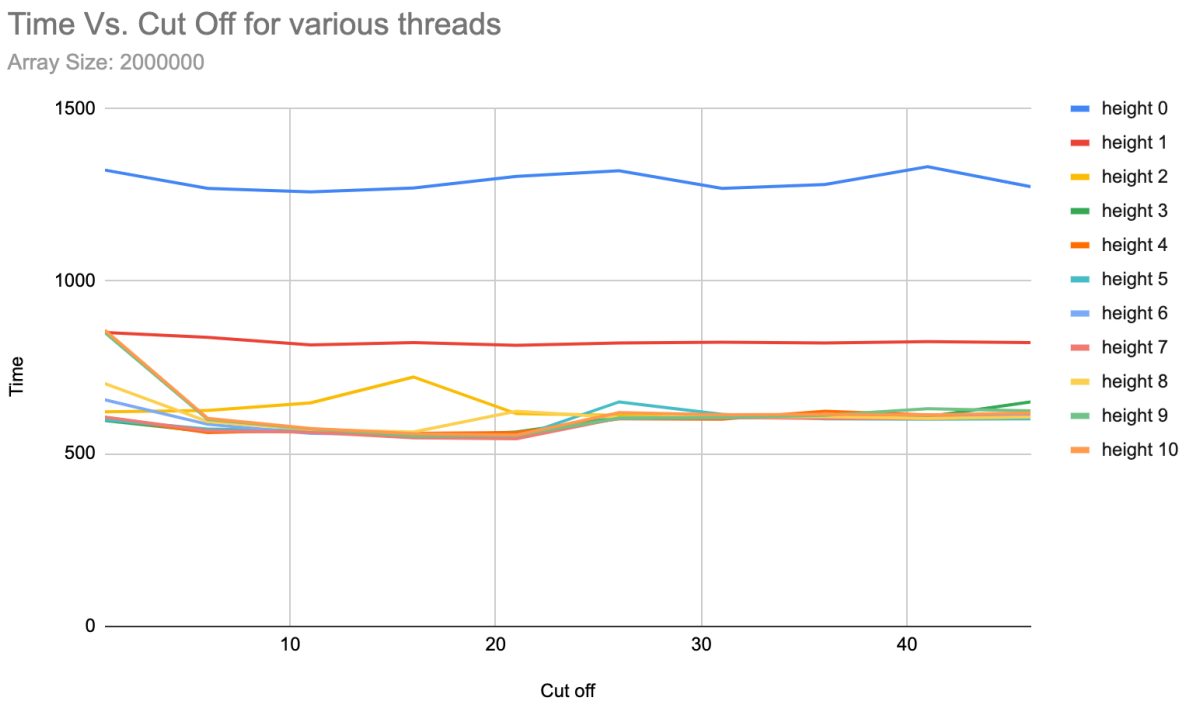
Results

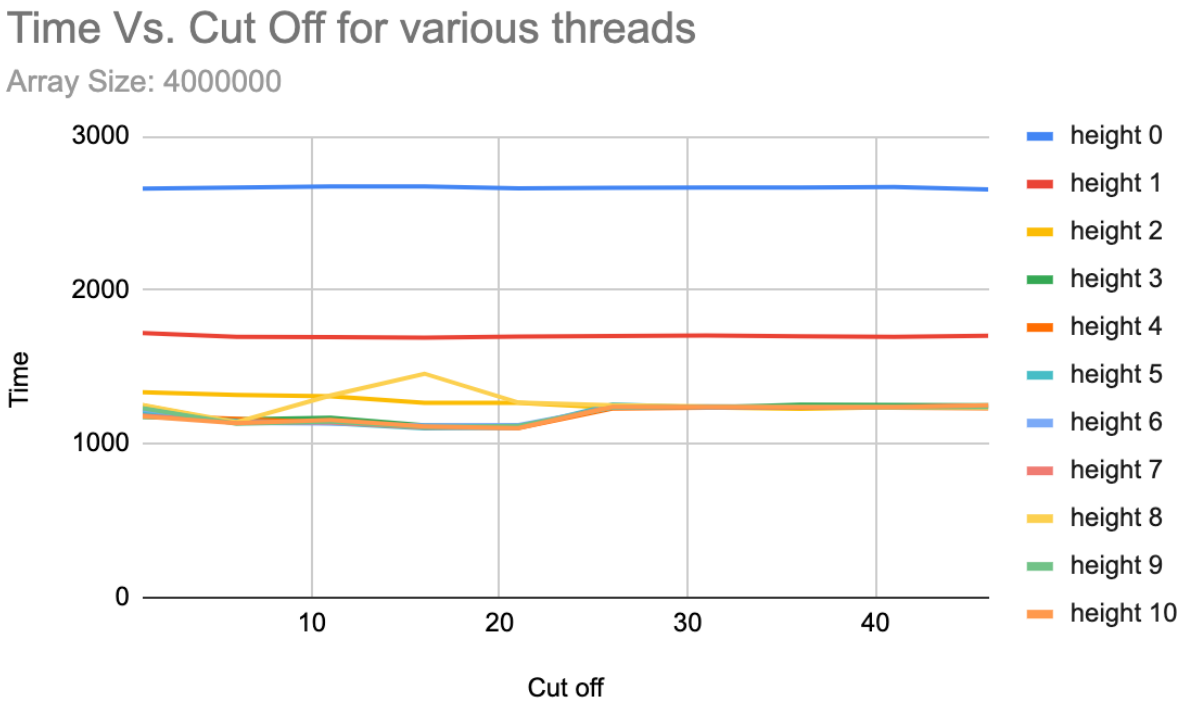
The experiments were conducted on a Macbook Air M1 and the results could change on different processors.

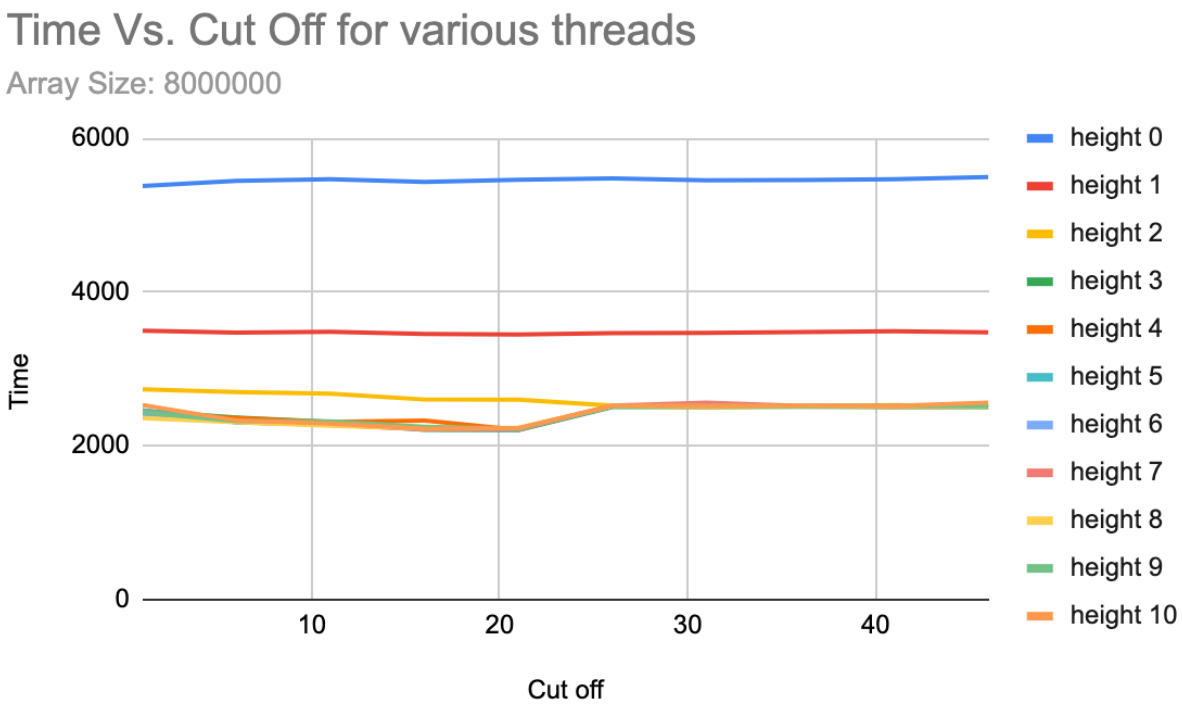
The results obtained from Experiment 1:

It can be seen from the graphs below that for all sizes of arrays and for levels of parallelism > 2 the optimal value of cutoff is around 20% of the array size.

Which means we can get the best performance when using parallel sort by switching from sorting partitions parallelly to system sort when the size of the partition to be sorted drops below ~20% of the size of the array.




Time Vs. Cut Off for various threads
Array Size: 4000000
3000
2000
1000
0
Time
10
20
30
40
Cut off
height 0
height 1
height 2
height 3
height 4
height 5
height 6
height 7
height 8
height 9
height 10


Time Vs. Cut Off for various threads
Array Size: 8000000
6000
4000
2000
0
Time
10
20
30
40
Cut off
height 0
height 1
height 2
height 3
height 4
height 5
height 6
height 7
height 8
height 9
height 10

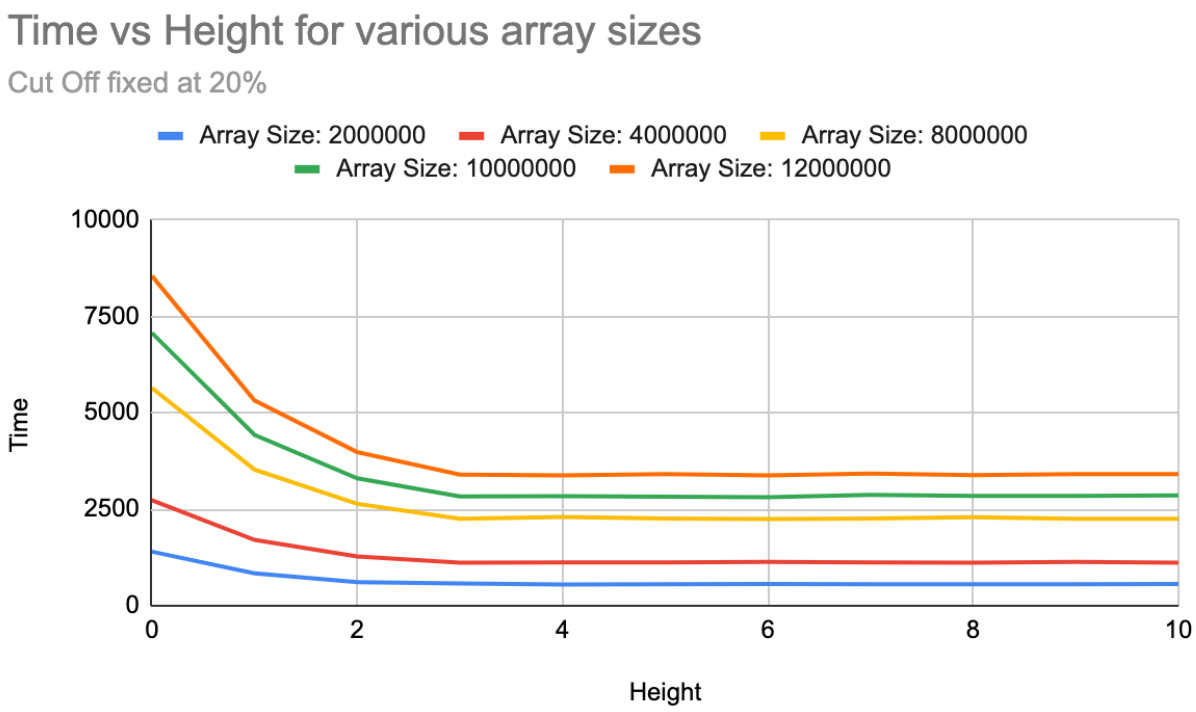
We can also observe that at height = 0 (no parallel sort and only system sort) the performance is the worst.

By having more levels of height sorted parallelly we are improving performance greatly.

At height = 1 we see a significant increase and we observe the same for height = 2.

From height = 3 onwards the improvement in performance is not that significant. The advantage of sorting parallelly is balanced by the disadvantage of having to manage threads and the partitions.

The results obtained from Experiment 2:



We can see from the graph above that with no parallel sorting any size of an array will give the worst performance. As we increase the level of parallelism the performance increases upto a height of 3. Any increase in threads will lead to not much performance gains.

Data For Experiment 2:

For cut off 20%:

Height	Array Size: 2000000	Array Size: 4000000	Array Size: 8000000	Array Size: 10000000	Array Size: 12000000
0	1401	2736	5645	7071	8549
1	836	1707	3528	4425	5319
2	613	1276	2640	3301	3980
3	578	1117	2256	2830	3400
4	551	1124	2297	2839	3377
5	553	1125	2260	2823	3411
6	561	1138	2248	2812	3378
7	555	1121	2259	2873	3422
8	556	1118	2295	2847	3383
9	555	1136	2256	2843	3409
10	565	1117	2255	2858	3414

Data For Experiment 2:

Array Size: 2000000

Cut off	height 0	height 1	height 2	height 3	height 4	height 5	height 6	height 7	height 8	height 9	height 10
1	1322	851	621	596	603	598	656	606	703	851	858
6	1269	837	625	564	561	571	585	566	593	599	602
11	1259	815	647	565	568	567	559	562	570	572	573
16	1270	822	722	548	559	547	555	546	563	551	558
21	1304	814	616	563	561	547	554	543	623	553	554
26	1320	821	611	605	601	650	602	603	605	603	619
31	1269	823	604	605	600	614	604	605	607	605	613
36	1280	821	606	605	623	601	608	602	607	611	614
41	1332	825	613	607	612	600	602	602	602	630	612
46	1274	822	606	650	602	601	614	607	606	624	618

Array Size: 4000000

Cut off	height 0	height 1	height 2	height 3	height 4	height 5	height 6	height 7	height 8	height 9	height 10
1	2660	1719	1335	1174	1185	1204	1223	1183	1252	1230	1177
6	2668	1695	1316	1158	1161	1147	1134	1140	1140	1130	1133
11	2675	1693	1308	1168	1137	1133	1132	1139	1313	1143	1154
16	2674	1690	1266	1119	1108	1100	1118	1104	1455	1109	1111
21	2662	1696	1266	1111	1102	1104	1121	1105	1267	1116	1101
26	2665	1701	1233	1235	1229	1254	1236	1242	1250	1235	1239
31	2667	1704	1236	1236	1233	1235	1242	1238	1243	1241	1239
36	2668	1698	1226	1253	1234	1241	1241	1238	1241	1239	1237
41	2670	1695	1237	1251	1233	1240	1237	1241	1238	1237	1238
46	2655	1702	1243	1249	1231	1236	1243	1247	1236	1241	1249

Array Size: 8000000

Cut off	height 0	height 1	height 2	height 3	height 4	height 5	height 6	height 7	height 8	height 9	height 10
1	5379	3495	2732	2455	2426	2461	2416	2438	2357	2431	2528
6	5446	3471	2697	2364	2358	2301	2313	2306	2304	2320	2328
11	5469	3481	2675	2310	2309	2276	2280	2270	2260	2315	2287
16	5433	3452	2600	2205	2325	2209	2218	2212	2222	2244	2219
21	5460	3446	2598	2204	2205	2210	2216	2217	2232	2220	2228
26	5479	3466	2521	2503	2513	2515	2514	2519	2514	2515	2521
31	5454	3469	2504	2502	2507	2514	2511	2559	2503	2516	2516
36	5457	3480	2515	2507	2513	2512	2510	2516	2510	2514	2521
41	5467	3490	2527	2504	2507	2514	2514	2507	2508	2517	2518
46	5496	3476	2505	2504	2510	2518	2517	2509	2511	2513	2559

Output

For testing cut off:

Array Size: 12000000

Degree of parallelism: 1024

8549

5319

3980

3400

3377

3411

3378

3422

3383

3409

3414

Process finished with exit code 0

Array Size: 10000000

Degree of parallelism: 1024

7071

4425

3301

2830

2839

2823

2812

2873

2847

2843

2858

Process finished with exit code 0

Array Size: 8000000

Degree of parallelism: 1024

5645

3528

2640

2256

2297

2260

2248

2259

2295

2256

2255

Process finished with exit code 0

Array Size: 4000000

Degree of parallelism: 1024

2736

1707

1276

1117

1124

1125

1138

1121

1118

1136

1117

Process finished with exit code 0

Array Size: 2000000

Degree of parallelism: 1024

1401

836

613

578

551

553

561

555

556

555

565

Process finished with exit code 0

For testing Heights:

Degree of parallelism: 1024

Array Size: 2000000

Height: 0

1416

1302

1277

1273

1286

1299

1287

1329

1287

1269

Height: 1

837

820

818

823

820

829

830

822

824

893

Height: 2

624

625

604

617

626

639

632

622

624

608

Height: 3

610

595

576

573

555

604

627

615

631

626

Height: 4

607

562

564

541

544

603

600

601

603

601

Height: 5

602

556

560

548

551

598

604

602

600

599

Height: 6

603

558

559

543

543

601

601

601

600

600

Height: 7

601

554

555

538

546

600

601

600

610

603

Height: 8

601

559

562

537

542

600

604

601

600

601

Height: 9

604

558

559

539

542

603

604

631

641

603

Height: 10

669

589

561

595

555

615

615

599

608

602

Process finished with exit code 0

Degree of parallelism: 1024

Array Size: 4000000

Height: 0

2671

2638

2647

2645

2642

2631

2638

2645

2639

2635

Height: 1

1706

1682

1679

1682

1679

1689

1698

1688

1706

1692

Height: 2

1327

1319

1297

1275

1347

1248

1229

1231

1251

1244

Height: 3

1195

1212

1209

1193

1167

1255

1256

1235

1239

1267

Height: 4

1280

1168

1176

1125

1107

1293

1386

1261

1236

1232

Height: 5

1214

1138

1131

1104

1105

1232

1234

1231

1230

1232

Height: 6

1183

1136

1144

1110

1105

1235

1233

1240

1236

1233

Height: 7

1173

1136

1140

1113

1110

1235

1266

1234

1231

1232

Height: 8

1215

1132

1139

1108

1110

1232

1234

1230

1235

1237

Height: 9

1177

1147

1138

1112

1104

1233

1232

1228

1234

1244

Height: 10

1253

1142

1138

1133

1129

1234

1234

1231

1233

1236

Process finished with exit code 0

Degree of parallelism: 1024

Array Size: 8000000

Height: 0

5589

5351

5370

5362

5372

5359

5356

5354

5360

5350

Height: 1

3532

3517

3508

3510

3486

3496

3522

3515

3513

3510

Height: 2

2808

2694

2693

2626

2613

2533

2534

2536

2532

2535

Height: 3

2512

2423

2380

2239

2254

2542

2534

2536

2546

2540

Height: 4

2503

2318

2280

2219

2252

2517

2527

2518

2539

2524

Height: 5

2439

2323

2293

2241

2239

2523

2540

2526

2536

2632

Height: 6

2595

2338

2276

2218

2218

2523

2518

2523

2522

2527

Height: 7

2514

2366

2339

2218

2257

2536

2541

2534

2528

2527

Height: 8

2422

2339

2288

2230

2263

2533

2529

2535

2531

2531

Height: 9

2406

2340

2279

2222

2227

2530

2526

2534

2540

2536

Height: 10

2481

2336

2290

2223

2230

2532

2525

2519

2526

2523

Process finished with exit code 0