HYPER QUICKSORT

Thana Shree Jeevanandam

CSE 633 Parallel Algorithms(Dr.Russ Miller)
April 28,2020

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

CONTENTS

Problem Statement

To improve the performance of the Quicksort Algorithm by modifying it for parallel execution with the Hyper-Quicksort Algorithm (by Bruce Wager) and to compare its performance as the number of processors changes.

Parallel Implementation

ASSUMPTION

- N-dimensional Hypercube - Number of processors is a **power of two**. ($n=2^N$)
- Processors are connected if and only if their unique $\log_2 n$-bit strings differ in exactly one position.
- Each Processor has $d/2^N$ data (d inputs).

WITH COMPLETION

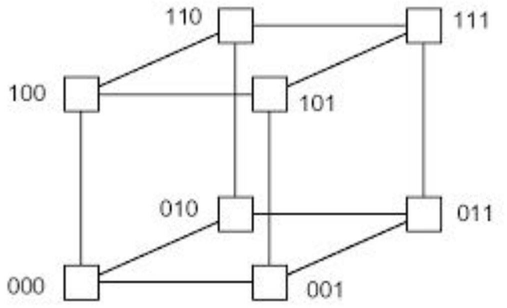
- Each processor has a sorted list in it's memory.
- LastElement(Pi) **<=** FirstElement(Pi +1) .
- The starting node collects all the individual lists and returns the output.

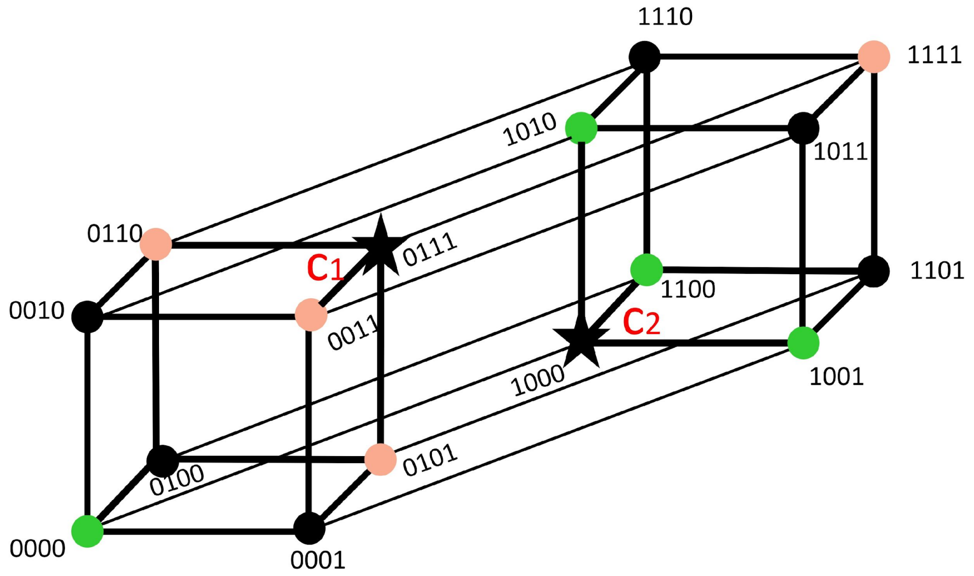
Hyperquicksort - Algorithm

- Locally sort the data in each processor.
- Randomly chosen processor gets its “Median” and broadcasts it to all the other processors.
- The processors locally splits the data into “High” and “Low” groups with “Median” as the pivot.

Hyperquicksort - Algorithm

- Consider “Upper” and “Lower” Subcubes (each hypercube with dimension- 2^{N-1}) differing in their most significant bits.
- Processor from “Upper” sends its “Low” to its adjacent process from “Lower”.
- Processor from “Lower” sends its “High” to its adjacent process from “Upper”.
- At “Upper”, each processor merges its own “High” with the obtained “High”.
- At “Lower”, each processor merges its own “Low” with the obtained “Low”.





Hyperquicksort - Algorithm

- Repeat the steps in parallel until the Subcubes have single processor.
- After (log n) recursions, every processor has an unsorted list of values completely disjoint from the values held by the other processors.
- LastElement(Pi) **<=** FirstElement(Pi +1).

- The expected running time,

$$\Theta\left(N \log N + \frac{d(d+1)}{2} + dN\right).$$

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Output

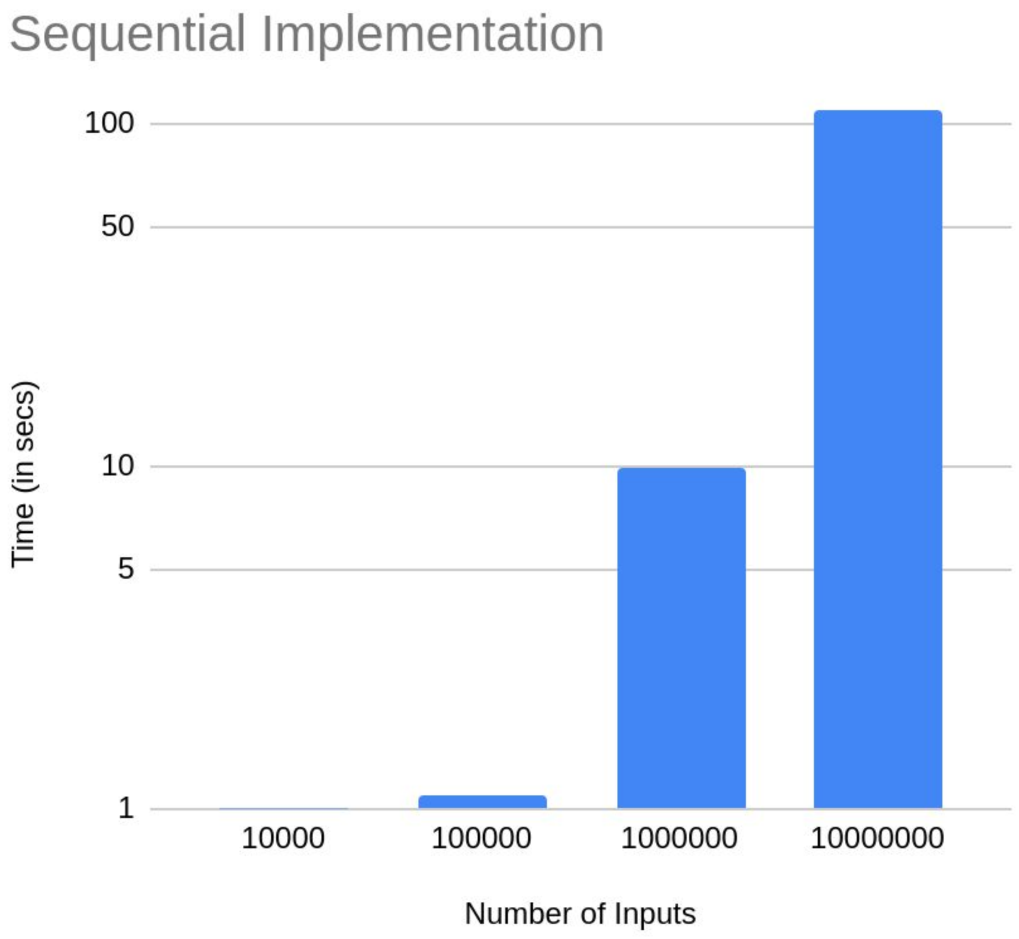
```
https://ubccr.freshdesk.com/solution/articles/13000066168

##############################################################

New nodes available in 'cascade' partition!  More details:
https://ubccr.freshdesk.com/en/support/solutions/articles/13000071364

##############################################################
[thanashr@vortex2:~]$ module load mpi4py/2.0.0-openmpi
 The mpi4py openmpi module has been loaded. The /util/common/python/anaconda-5.0.1 distribution and openmpi/gcc-4.8.x/2.0.2 are being used. Python 2.7 and 3.6 are available. Source the py27-mpi or
py36-mpi environment. To use mpi4py for python 2.7 type "source activate py27-mpi".
[thanashr@vortex2:~]$ source activate py36-mpi
(py36-mpi) [thanashr@vortex2:~]$ nano test.py
(py36-mpi) [thanashr@vortex2:~]$ mpiexec -n 1 python test.py
Sorted array is:
[8, 14, 22, 54, 66, 78, 89, 122, 123, 123, 138, 161, 183, 188, 220, 224, 234, 246, 262, 276, 280, 296, 333, 340, 345, 353, 357, 357, 382, 392, 392, 396, 396, 397, 456, 474, 478, 483, 526, 529, 531, 533,
534, 539, 547, 567, 576, 580, 584, 592, 594, 605, 606, 616, 623, 646, 648, 650, 652, 714, 716, 737, 770, 804, 807, 820, 827, 837, 838, 848, 854, 872, 876, 895, 901, 911, 911, 914, 919, 925, 951, 965, 982
, 995, 1018, 1046, 1051, 1052, 1054, 1064, 1069, 1070, 1090, 1111, 1112, 1114, 1121, 1123, 1126, 1132, 1155, 1169, 1179, 1185, 1188, 1205, 1219, 1231, 1233, 1246, 1263, 1270, 1277, 1278, 1279, 1290, 1296
, 1301, 1306, 1308, 1318, 1323, 1331, 1347, 1351, 1365, 1397, 1397, 1397, 1415, 1441, 1447, 1461, 1474, 1479, 1480, 1488, 1504, 1517, 1555, 1561, 1579, 1597, 1604, 1615, 1620, 1625, 1633, 1635, 1639, 165
8, 1676, 1717, 1719, 1752, 1752, 1775, 1776, 1801, 1813, 1822, 1843, 1846, 1866, 1867, 1868, 1868, 1872, 1875, 1885, 1899, 1913, 1915, 1920, 1934, 1965, 1978, 1987, 2004, 2011, 2018, 2039, 2082, 2083, 20
94, 2099, 2104, 2107, 2108, 2126, 2147, 2154, 2168, 2175, 2175, 2177, 2188, 2220, 2234, 2241, 2251, 2258, 2291, 2293, 2318, 2319, 2322, 2325, 2332, 2344, 2351, 2373, 2384, 2387, 2392, 2394, 2395, 2399, 2
410, 2427, 2430, 2434, 2436, 2460, 2468, 2470, 2474, 2495, 2512, 2523, 2526, 2527, 2529, 2532, 2539, 2550, 2552, 2559, 2568, 2586, 2599, 2611, 2613, 2647, 2647, 2651, 2652, 2654, 2664, 2669, 2677, 2695,
2698, 2699, 2716, 2723, 2728, 2730, 2731, 2754, 2788, 2807, 2815, 2816, 2836, 2854, 2872, 2882, 2886, 2909, 2917, 2928, 2929, 2941, 2945, 2962, 2994, 3008, 3013, 3032, 3037, 3037, 3042, 3043, 3057, 3091,
 3103, 3108, 3111, 3123, 3151, 3155, 3165, 3172, 3182, 3187, 3199, 3215, 3223, 3238, 3249, 3254, 3264, 3273, 3277, 3280, 3303, 3316, 3316, 3318, 3339, 3365, 3374, 3382, 3390, 3393, 3398, 3425, 3425, 3427
, 3432, 3435, 3454, 3465, 3480, 3486, 3488, 3490, 3491, 3519, 3521, 3533, 3546, 3574, 3578, 3599, 3610, 3614, 3615, 3621, 3624, 3625, 3637, 3640, 3643, 3666, 3681, 3682, 3683, 3685, 3693, 3707, 3709, 376
8, 3778, 3787, 3808, 3819, 3821, 3824, 3848, 3857, 3885, 3886, 3891, 3900, 3905, 3909, 3916, 3921, 3923, 3926, 3926, 3945, 3950, 3951, 3958, 3971, 3982, 3982, 3986, 3998, 4004, 4007, 4013, 4028, 4028, 40
51, 4053, 4053, 4055, 4070, 4084, 4103, 4104, 4106, 4106, 4107, 4128, 4137, 4137, 4143, 4145, 4145, 4161, 4183, 4184, 4187, 4192, 4194, 4197, 4200, 4217, 4229, 4231, 4237, 4245, 4248, 4267, 4268, 4287, 4
295, 4308, 4308, 4321, 4341, 4347, 4353, 4357, 4364, 4383, 4403, 4405, 4405, 4411, 4421, 4439, 4462, 4468, 4470, 4472, 4488, 4490, 4491, 4517, 4519, 4521, 4523, 4529, 4532, 4546, 4559, 4565, 4577, 4582,
4603, 4603, 4610, 4614, 4630, 4630, 4635, 4636, 4655, 4668, 4681, 4690, 4698, 4704, 4718, 4721, 4737, 4742, 4753, 4755, 4766, 4775, 4776, 4786, 4796, 4815, 4825, 4826, 4833, 4841, 4848, 4853, 4860, 4864,
 4888, 4889, 4895, 4896, 4925, 4932, 4937, 4945, 4946, 4974, 4995, 4997, 5050, 5051, 5063, 5066, 5090, 5096, 5109, 5123, 5128, 5169, 5190, 5191, 5227, 5238, 5247, 5250, 5253, 5257, 5279, 5283, 5292, 5296
, 5307, 5330, 5342, 5345, 5374, 5383, 5421, 5423, 5432, 5445, 5485, 5516, 5518, 5518, 5546, 5550, 5552, 5583, 5595, 5614, 5624, 5646, 5647, 5648, 5651, 5671, 5672, 5685, 5685, 5692, 5738, 5741, 5756, 575
9, 5764, 5767, 5775, 5775, 5777, 5778, 5794, 5805, 5815, 5819, 5851, 5884, 5887, 5888, 5894, 5895, 5896, 5899, 5906, 5907, 5923, 5938, 5953, 5992, 6000, 6001, 6003, 6006, 6006, 6021, 6041, 6042, 6042, 60
66, 6091, 6102, 6102, 6103, 6113, 6130, 6130, 6133, 6143, 6144, 6155, 6172, 6191, 6191, 6198, 6205, 6209, 6216, 6221, 6227, 6229, 6235, 6236, 6246, 6246, 6255, 6271, 6273, 6275, 6278, 6290, 6303, 6309, 6
311, 6331, 6350, 6352, 6353, 6356, 6357, 6362, 6375, 6385, 6393, 6397, 6401, 6409, 6412, 6430, 6439, 6460, 6462, 6465, 6466, 6468, 6477, 6478, 6521, 6539, 6541, 6553, 6556, 6560, 6560, 6564, 6567, 6589,
6592, 6596, 6608, 6633, 6647, 6648, 6650, 6654, 6655, 6674, 6688, 6695, 6695, 6728, 6734, 6747, 6754, 6763, 6774, 6774, 6780, 6795, 6799, 6807, 6832, 6849, 6850, 6877, 6895, 6903, 6905, 6927, 6930, 6931,
 6932, 6936, 6938, 6944, 6945, 6949, 6977, 6988, 7006, 7012, 7017, 7036, 7048, 7066, 7074, 7111, 7113, 7122, 7131, 7143, 7147, 7150, 7162, 7202, 7210, 7211, 7228, 7236, 7257, 7258, 7259, 7281, 7292, 7301
, 7310, 7334, 7353, 7363, 7380, 7383, 7384, 7392, 7395, 7415, 7429, 7432, 7461, 7482, 7485, 7487, 7493, 7497, 7522, 7530, 7584, 7596, 7598, 7613, 7634, 7659, 7666, 7669, 7690, 7715, 7737, 7761, 7804, 780
5, 7818, 7828, 7830, 7850, 7861, 7880, 7880, 7887, 7896, 7903, 7911, 7919, 7927, 7946, 7949, 7952, 7955, 7964, 7969, 7987, 7995, 7997, 8010, 8029, 8039, 8039, 8043, 8054, 8054, 8055, 8061, 8072, 8079, 80
90, 8103, 8107, 8110, 8119, 8123, 8125, 8127, 8132, 8157, 8166, 8184, 8191, 8212, 8227, 8229, 8233, 8254, 8257, 8263, 8294, 8296, 8301, 8321, 8324, 8337, 8352, 8353, 8358, 8367, 8380, 8383, 8385, 8397, 8
400, 8422, 8425, 8427, 8431, 8440, 8457, 8461, 8461, 8487, 8492, 8495, 8504, 8505, 8518, 8537, 8550, 8563, 8588, 8591, 8593, 8594, 8606, 8612, 8613, 8627, 8633, 8650, 8673, 8702, 8702, 8709, 8721, 8721,
8726, 8728, 8739, 8758, 8761, 8762, 8765, 8768, 8785, 8787, 8807, 8813, 8821, 8822, 8836, 8843, 8853, 8858, 8861, 8891, 8929, 8930, 8935, 8936, 8940, 8977, 8992, 8992, 9002, 9026, 9038, 9042, 9054, 9063,
 9079, 9084, 9095, 9098, 9103, 9106, 9110, 9119, 9120, 9129, 9129, 9131, 9160, 9165, 9183, 9191, 9207, 9216, 9222, 9231, 9240, 9251, 9260, 9272, 9274, 9288, 9300, 9316, 9320, 9324, 9325, 9333, 9352, 9363
, 9372, 9374, 9378, 9381, 9390, 9406, 9417, 9431, 9431, 9432, 9443, 9444, 9470, 9486, 9490, 9492, 9499, 9499, 9504, 9518, 9529, 9536, 9550, 9550, 9556, 9561, 9577, 9578, 9588, 9608, 9609, 9611, 9615, 961
8, 9621, 9638, 9643, 9648, 9650, 9665, 9666, 9671, 9673, 9675, 9687, 9697, 9714, 9718, 9724, 9729, 9734, 9745, 9749, 9753, 9760, 9761, 9771, 9778, 9786, 9787, 9801, 9816, 9824, 9837, 9841, 9866, 9878, 98
81, 9901, 9907, 9927]
Number of inputs were: 1000
Execution time is 0.008293390274047852
(py36-mpi) [thanashr@vortex2:~]$
```

RUNTIME COMPARISON

University at Buffalo
Department of Computer Science
and Engineering
School of Engineering and Applied Sciences

Sequential Implementation

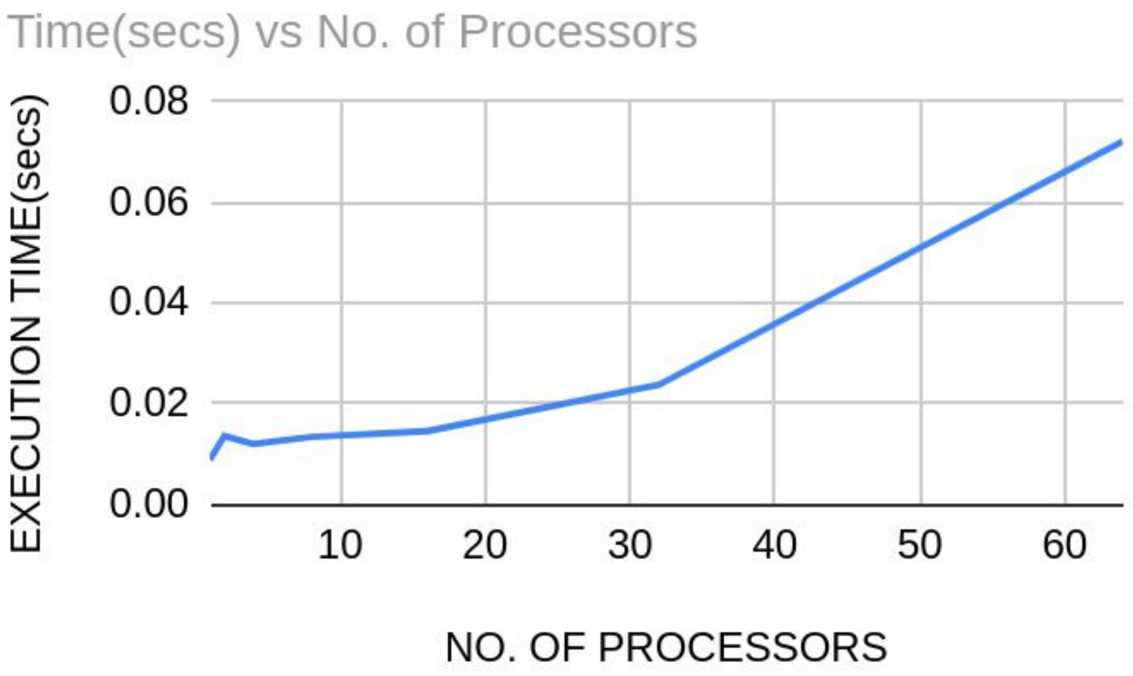
INPUTS	***EXECUTION TIME (in secs)***
10000	*0.08757781982*
100000	*1.098776817*
1000000	*9.905396461*
10000000	*109.7441607*



Parallel Implementation

For ***1088*** data

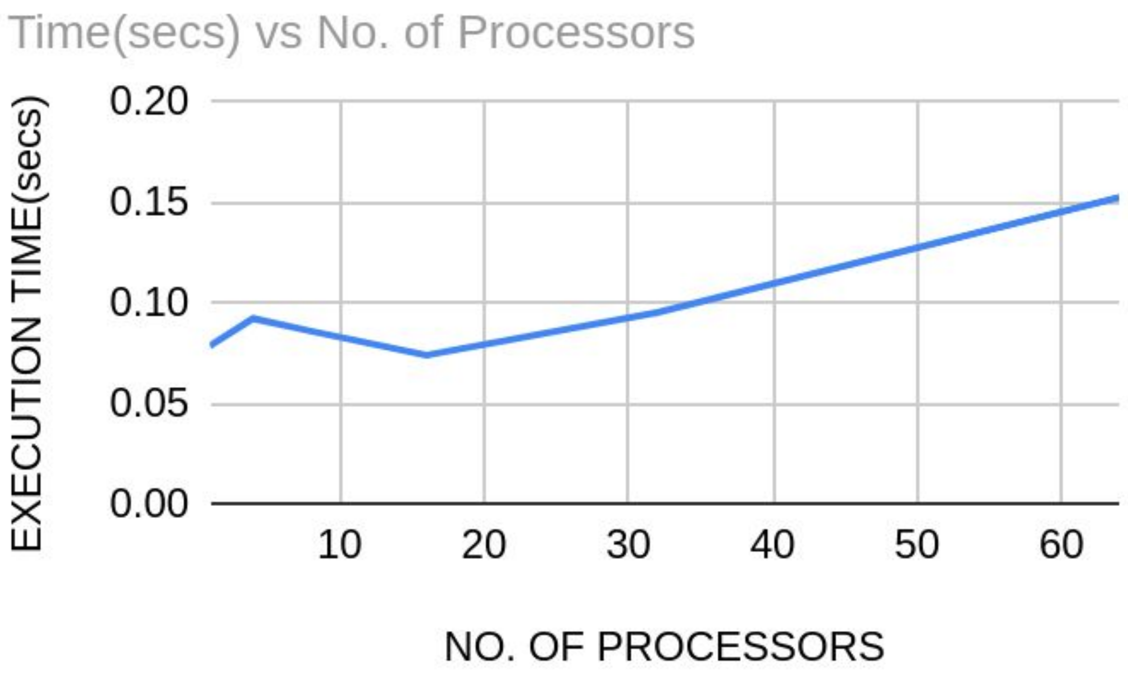
NO. OF NODES	***EXECUTION TIME (in secs)***
1	0.0871899128
4	0.08312749863
8	0.08022236824
16	0.06556296349
32	0.09054040909
64	0.1168482304



Parallel Implementation

For ***10304*** data

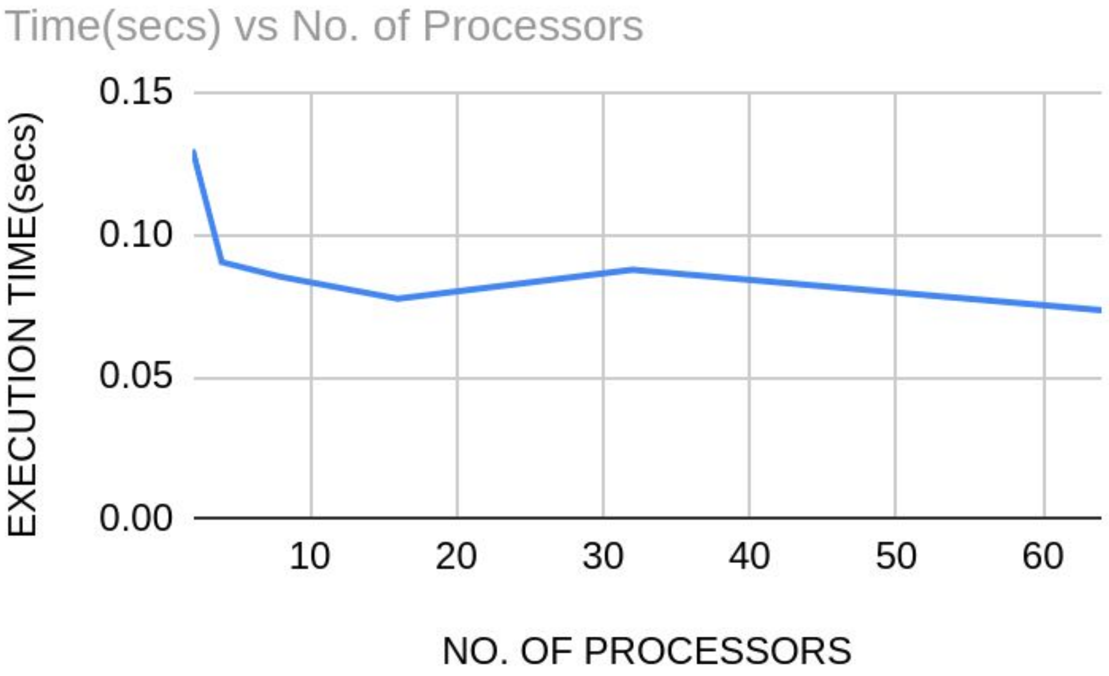
NO. OF NODES	***EXECUTION TIME (in secs)***
1	0.0787835121
4	0.0925407409
8	0.0862121582
16	0.0742335319
32	0.0953888893
64	0.1526854038



Parallel Implementation

For ***10560*** data

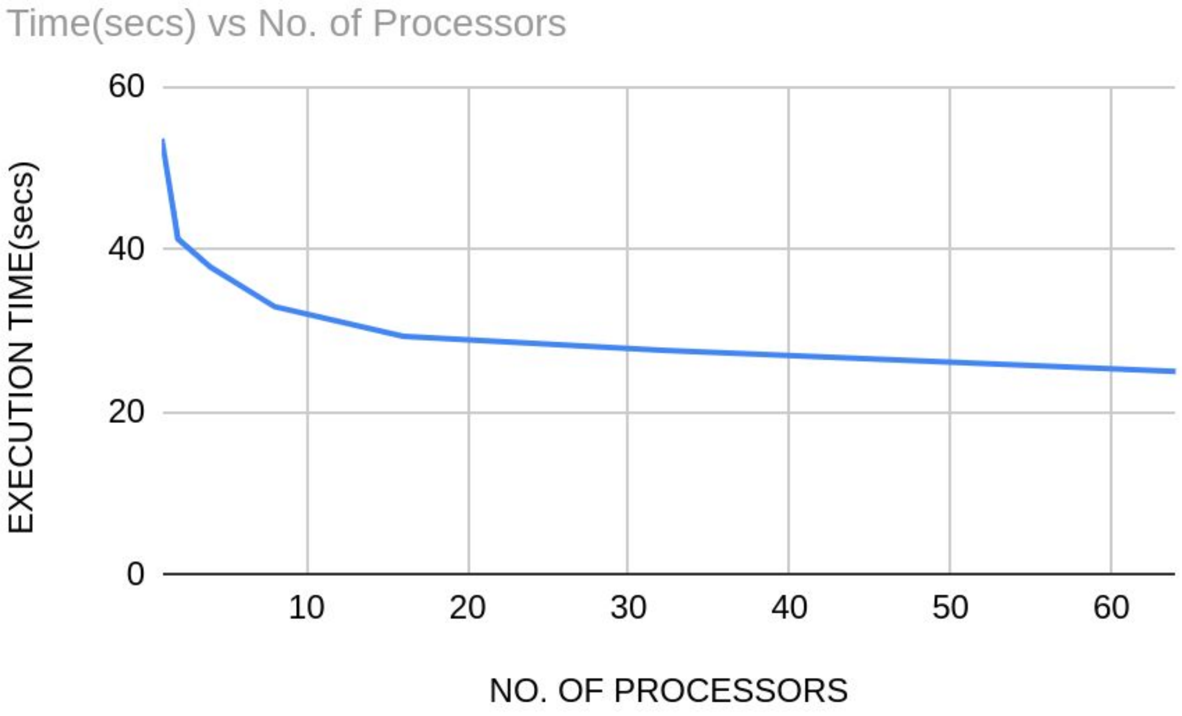
NO. OF NODES	***EXECUTION TIME (in secs)***
2	0.13
4	0.09041595459
8	0.08528614044
16	0.07749915123
32	0.08768796921
64	0.07342982292



Parallel Implementation

For 10Million data

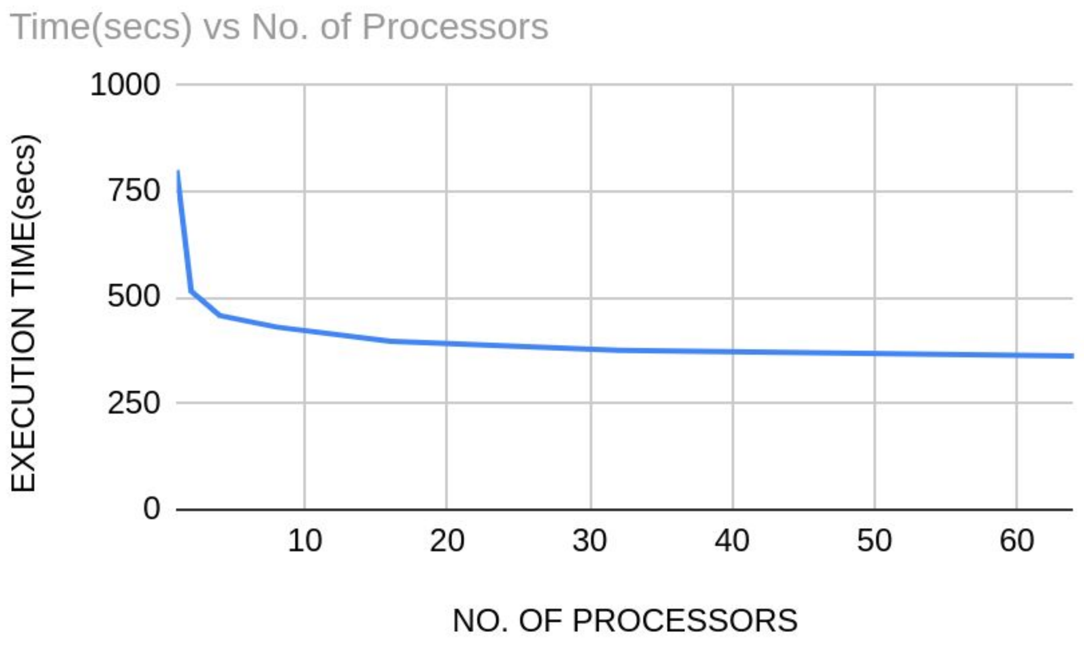
NO. OF NODES	***EXECUTION TIME (in secs)***
1	53.65874353
2	41.34348739
4	37.87895323
8	32.99123985
16	29.33294819
32	27.65038346
64	25.03485932



Parallel Implementation

For 100 Million data

NO. OF NODES	***EXECUTION TIME (in secs)***
1	800.194371
2	514.765896
4	457.706214
8	430.185896
16	396.648367
32	375.087699
64	362.099076



Conclusion

- The hyperquicksort was implemented using MPI4PY parallelly on the different number of processors.
- The results were compared and interpreted.

Reference

- Algorithms, Sequential and Parallel: A Unified Approach – Russ Miller and Laurence Boxer. 3rd Edition.

- MPI4PY Documentation - https://mpi4py.readthedocs.io/en/stable/tutorial.html

THANK YOU

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences