**COL757 Programming Assignment**

Yash Gupta – 2013CS10302

# Q1: Compute OR of a million bits

* N = 1 mil => inconclusive results due to very small problem size.
* N = 10 mil

|  |  |  |  |
| --- | --- | --- | --- |
| Threads/Algorithm | P = 6 | P = 12 | P = 24 |
| SERIAL | 0.007747160 | 0.007554330 | 0.007580700 |
| CONCURRENT WRITE | 0.000971427 | 0.000561381 | 0.000360807 |
| BINARY RREE | 0.000969006 | 0.000497586 | 0.000307906 |

* SERIAL runs in ~ constant time as expected
* Both parallel implementations give significant speedup compared to SERIAL, but only when problem size is large enough .  
  (N = 1 mil gives approximately same results for all algo/thread combinations)
* CONCURRENT WRITE is as fast as BINARY TREE when number of threads is low, but becomes increasingly less efficient with more threads.
* Conclusion: when operating on low number of threads, concurrent write can be useful due to its low implementation cost (in terms of software)

# Q2: Implement parallel mergesort and partition sort

* N = 16,000

|  |  |  |  |
| --- | --- | --- | --- |
| Threads/Algorithm | P = 6 | P = 12 | P = 24 |
| SERIAL | 0.000990361 | 0.000993135 | 0.000997214 |
| ODD-EVEN MERGE | 0.00189207 | 0.00571344 | 0.0367163 |
| PARTITION SORT | 0.00814808 | 0.00948329 | 0.0214106 |

* N = 1,600,000

|  |  |  |  |
| --- | --- | --- | --- |
| Threads/Algorithm | P = 6 | P = 12 | P = 24 |
| SERIAL | 0.142126 | 0.142126 | 0.142309 |
| ODD-EVEN MERGE | 0.0418275 | 0.0271082 | 0.0468103 |
| PARTITION SORT | 0.836293 | 1.0389 | 1.21488 |

* SERIAL performs better than both parallel implementations at smaller problem size
* ODD-EVEN MERGE outperforms SERIAL at large problem sizes.
* ODD-EVEN MERGE scales very moderately at large problem sizes, with degradation at P=24.
  + This might be due to the fast that there are only 12 physical cores on the machine, and 24 logical cores due to hyper-threading (super-scalar pipeline design).
* As thread count increases at smaller problem sizes, partition sort degrades moderately.
  + More rapid degradation is seen with large problem size (possibly due to large number (sqrt(N)) of sub-problems created at each step)
* Conclusion:
  + For small problem sizes, use SERIAL
  + For larger problem sizes, use ODD-EVEN MERGE, as PARTITION SORT has very high overheads.