**Multithreaded Merge Sort**

**Background/Assumptions:**

* Serial algorithm vs. Parallel algorithm
* Thread is an abstraction of a “virtual processor” (multicores CPU & multiprocessors)
* **Multithread**: multithreading a single algorithm so that some of its instructions may be executed simultaneously
* Threads may switch/exchange data or results, **shared memory** allows each thread to directly access any location of memory
* **Greedy scheduler** assigns as many strands to processors as possible in each time step

**Key Concepts for Multithreading:**

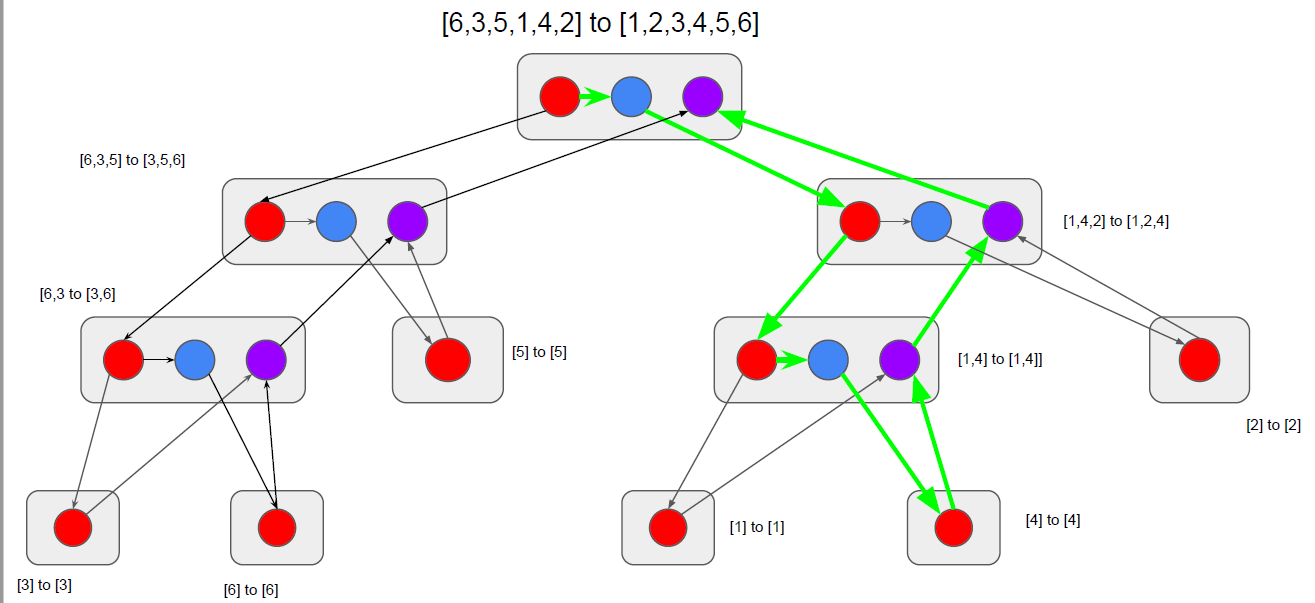
* **Spawn**: create a parallel subprocess (child), then keep executing the current process
* **Sync**: must wait for all its spawned children to complete before proceeding to the statement after sync
* **Work**: the sum of the time taken by each thread (**T1**)
* **Span**: the longest time it takes to execute threads along any path of the computational DAG (directed acyclic graph) (**TInf**)
* **Parallelism**: the ratio of the work to the span: **P = T1 / TInf**
* **Slackness**: the (parallel) slackness of a multithreaded computation executed on an ideal parallel computer with P processors to be the ratio: **S = (T1 / TInf \* P)**

**Pseudocode:**

|  |  |
| --- | --- |
| **Multithreaded Merge-Sort** | **Multithreaded Merging** |
|  |  |

*(Source: CLRS 3rd Ed. PP797, 800)*

**DAG** (an example of multithreaded merge sort for A= [6,3,5,1,2,4])

****

**Time Complexity Comparison:**

|  |  |  |
| --- | --- | --- |
|  | Serial | Parallel |
| Merge | PM1(n) = PM1(ɑn) + PM1((1-ɑ)n) + O(lg2n) for ¼ <= ɑ <= ¾  ⇒ PM1(n) = Θ(n) | PM∞(n) = PM∞(¾ n) + Θ(lg2n)  ⇒ PM∞(n) = Θ(**lg2n**) |
| Merge-Sort | PMS1(n) = 2\*PMS1(½ n) + PM1(n)  ⇒ PMS1(n) = Θ(n lg2n) | PMS∞(n) = PMS∞( ½ n) + PM∞(n)  ⇒ PMS∞(n) = Θ(**lg3n**) |

*Sources:*

1. *Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L. and Stein, Clifford. Introduction to Algorithms Third Edition. The MIT Press, 2009.*
2. *Wikipedia. “Parallel algorithm.” https://en.wikipedia.org/wiki/Parallel\_algorithm (accessed April 10, 2023).*
3. *Blelloch, Guy E. and Maggs, Bruce M. “Parallel Algorithms.” https://www.cs.cmu.edu/~guyb/papers/BM04.pdf (accessed April 10, 2023).*
4. *Bouguezzi, Safa. “Multithreaded Algorithms.” https://www.baeldung.com/cs/multithreaded-algorithms (accessed April 10, 2023).*
5. *Moore, Tamsin. “Multi-Threading in Java.” https://slideplayer.com/slide/9128173/ (accessed April 10, 2023).*
6. *Staff, Maker.io. “An Introduction to Multi-Threading C Programs.” https://www.digikey.com/en/maker/blogs/2022/an-introduction-to-multi-threading-c-programs (accessed April 10, 2023).*
7. *Suthers, Dan. “Multithreaded Algorithms.” http://www2.hawaii.edu/~suthers/courses/ics311f20/Notes/Topic-22.html (accessed April 11, 2023).*