Frequently asked questions Parallel Computing
by Prof. Subodh Kumar ,
Department of Computer Science and Engineering, IIT Delhi,

Frequently asked questions:

1. What is shared-memory architecture?

Ans: A single address space is visible to all execution threads.

2. What is NUMA memory architecture?

Ans: NUMA stands for Non-Uniform memory access and is a special type of shared memory architecture where access times to different memory locations by a processor may vary as may also access times to the same memory location by different processors.

3. Name some network architectures prevalent in machines supporting the message passing paradigm.

Ans: Ethernet, Infiniband, Tree

4. What is data-parallel computation?

Ans: Data is partitioned across parallel execution threads, each of which perform some computation on its partition – usually independent of other threads.

5. What is Task-parallel computation?

Ans: The parallelism manifests across functions. A set of functions need to compute, which may or may not have order constraints among them.

6. What is task-latency?

Ans: The time taken for a task to complete since a request for it is made.

7. What is task-throughput?

Ans: The number of tasks completed in a given time

8. What is Speed-up?

Ans: The ratio of some performance metric (like latency) obtained using a single processor with that obtained using a set of parallel processors.

9. What is parallel efficiency?

Ans: The Speed-up per processor

10. What is an inherently sequential task?

Ans: On whose maximum speed-up (using any number of processors) is 1.

11. What is the maximum time speed-up possible according to Amdahl's law?

Ans: 1/f, where f is inherently sequential fraction of the time taken by the best sequential execution of the task.

12. What is SIMD?

Ans: A class belonging to Flynn’s taxonomy of parallel architectures, it stands for single instruction multiple data. In this architecture, different processing elements all execute the same instruction in a given clock cycle, with the respective data (e.g., in registers) being independent of each other.

13. What is cache coherence?

Ans: Different processors may maintain their own local caches. This results in potentially multiple copies of the same data. Coherence implies that access to the local copies behave similarly to access from the local copy – apart from the time to access.

14. What is a hypercube connection?

Ans: A single node is a hypercube. An *n* node hypercube is made of two *n*/2 node hypercube, with their corresponding nodes connected to each other.

15. What is the diameter of an *n*-node hypercube?

Ans: log *n*. The diameter is the minimum number of links required to reach two furthest nodes.

16. How does OpenMP provide a shared-memory programming environment

Ans: OpenMP uses pragmas to control automatic creation of threads. Since the thread share the address space, they share memory. However, they are allowed a local view of the shared variables through “private” variables. The compiler allocates a variable-copy for each thread and optionally initializes them with the original variable. Within the thread the references to private variable are statically changed to the new variables.

17. What is the memory consistency model supported by OpenMP?

Ans: There is no “guaranteed” sharing/consistency of shared variables until a flush is called. Flush sets that overlap are sequentially consistent and the writes of a variable become visible to every other thread at the point flush is serialized. This is slightly weaker than “weak consistency.”

18. How are threads allocated to processors when there are more threads than the number of processors?

Ans: Once a thread is completed on a core, a new thread is run on it. The order can be controlled using the “Schedule” clause.

19. What is common CRCW PRAM?

Ans: Parallel Random Access Model of Computation in which the processors can write to a common memory address in the same step, as long as they are all writing the same value.

20. What is the impact of limiting pram model to a fixed number of processors or a fixed memory size.

Ans: Prams with higher capacities can be simulated can be simulated (with linear slowdown).

21. What is the impact of eliminating shared write from PRAM?

Ans: It can be simulated by crew pram with a log n factor in the time. However, the algorithms in this model can become a little complicated, as they must ensure conflict free writes.

22. What is the significance of work complexity analysis?

Ans: Time complexity does not account for the size of the machine. Work complexity is more reflective of practical efficiency. Work-time scheduling principle describes the expected time for a *p* processor pram as work/*p*.

23. What does bulk synchronous model add to pram for parallel algorithm analysis

Ans: Pram assumes constant time access to shared memory, which is unrealistic. Bsp counts time in "message communication" and in this model a step isn't initiated until the input data has arrived.

24. Is it true that all NC problems parallelize well?

Ans: In general NC problems do parallelize well in terms of having a poly-log solution in pram model while it only has a super log solution in ram model. However, for problems with poly-log solution in ram models, there may not be an effective speed-up.

25. Is user locking required to control the order of access to guarantee sequential consistency?

Ans: Sequential consistency is independent of user locking but does require delaying of memory operations at the system level. Precise ordering of operations need not be pre-ordained by the program logic. There just must exist a global ordering which is consistent with the local view observed by each processor.

26. What is the difference between processor and FIFO consistency?

Ans: n FIFO consistency only writes from a single processor are visible in the order issued. In processor consistency, additionally there exists a global ordering of writes to any address x by different processes exists that is consistent with the local views.

27. What is false sharing?

Ans: Sharing of a cache line by distinct variables. As a result, performance issues come into play. If such variables are not accessed together, the un-accessed variable is unnecessarily brought into cache along with the accessed variable.

28. What is a task dependency graph?

Ans: A directed graph with nodes representing tasks and edge from task *a* to *b* indicating that task *b* can only start after task *a* is completed.

29. When can an MPI send call return?

Ans: If it is a synchronous call, it can return only when the pairing call on another process is ready. For asynchronous versions, it can return as soon as the provided buffer is ready for re-use.

30. What is a collective communication call?

Ans: It's a call that must be made at all members of the communication group. No call can return until all calls have been at least been made.

31. How cam MPI be used for shared memory style programming?

Ans: Each process registers its local memory and attaches it to a "window." Accesses via this window get translated to send or fetch requests to the desired member of the group. The pairing communication is handled by the MPI system asynchronously.

32. What is the complexity of prefix sum in pram model?

Ans: Time O(log n) and work O(n)

33. What is the time complexity of optimal merge algorithm (on PRAM)?

Ans: O(log log n) by first merging sub-sequences of the original lists of size n/(log log n) each. The remaining elements are inserted into the just computed sequence in the next step.

34. What is accelerated cascading?

Ans: The accelerated cascading technique combines a fast but work-inefficient algorithm with a work optimal one. The problem is recursively divided into many smaller sub-problems, which are first solved solved using the optimal algorithm. The sub-results are then combined with the faster version of the algorithm.

35. Why must Cuda divide computation twice: into grids and then blocks?

Ans: The hardware is based on maximizing throughput. This has been done by allowing a large number of running threads -- all with a live context. This implies that only a fixed number of threads can fit in the hardware. This in turn means that these threads cannot communicate with or depend on other thread that could not be fit and hence must wait for the first set of threads to complete execution. Hence, a two level decomposition. Further, even the set of threads running together may execute at different SMs, and synchronization across SMs would be slow and onerous and hence not supported.

36. How do memory operations in GPUs differ from those in CPUs?

Ans: GPUs have a significantly smaller cache making average latency of memory operations much higher. This requires many concurrent threads to hid the latency. Also, the shared memory can be used as an opaque cache in direct control of the programmer -- making it possible to utilize the cache better in some situations. Further, because of SIMD warp instructions, multiple memory accesses are made per instruction. These accesses can be coalesced into a smaller number of real accesses, if the address set is contiguous for global memory or strided for shared memory.

37. How can two GPU threads communicate through shared memory?

Ans: if the threads belong to a non-divergent warp, writes before reads are visible to the read. Two threads in the same block must have an intervening sync for the write to affect the read. Two thread in different blocks within the same kernel cannot be guaranteed an order and the read must be moved to a later kernel for the write to become visible.

38. How can prefix minima be found in O(1) time?

Ans: This can be computed by first finding all nearest smaller values first in O(1) and then checking in O(1) time for each element (using O(n) processor for that element), that largest index smaller than its own, whose element has no nearest smaller value on its left. The work complexity of O(n^2) can be improved using accelerated cascading.

39. How long does Bitonic sorting require on PRAM?

Ans: O($\log^2 n$)

40. How long does Batcher's odd-even merge require?

Ans: O(log n) time, O(n log n) work

41. In order to balance load for parallel bucket sort of n elements, uniformly spaced splitters need to be selected. This can be done by first dividing the list into B lists and choosing B equi-spaced samples from each. The final B splitters are chosen uniformly spaced from these samples. How balanced are the buckets if these splitters are used?

Ans: No bucket will contain more than 2n/B elements.

42. How fast can two sorted lists of size n each be merged into one using p processors?

Ans: O(n/p) time using optimal multi-way merge.

43. How fast can a list be sorted using n processors using local sorting of n/p elements each followed by optimal multi-way merge?

Ans: O(n/p log n)

44. When stealing load from a random loaded processor, what type of synchronization is needed?

Ans: One needs to make sure that the queue being stolen from is operated in a synchronized fashion – either locked or edited in a lock-free manner.

45. How can one ensure mutual exclusion without locks?

Ans: When references of two (or more) threads (or processes) may be serialized with respect to a variable, system primitives like compare and swap can help detect the conflict with another thread. Lock free implementations of a thread usually detect the conflict atomically (e.g., using compare and swap) and one succeeds while the other backs off and retries.

46. How long does the parallel version of Prim's minimum spanning tree finding algorithm require for a graph with *n* nodes using *p* processors?

Ans: $O(n^2/p + n \log p)$