2018113003

Q6)

→ Given: Merge sorting array of N elements (Iterative)
Each element is byte sized
Cache size = 1 kB
Cache line size = 64
Cache is fully associative

To find: Amount of cache misses

→ Cache lines = 16 (Cache size / Cache line size)

Cache misses will occur only when 64 bytes are read or written (~~Even~~ Every time). During a merge

During a merge operation, 2 cache lines will be used for sorted runs to be merged & 1 cache line for merged output.

∴ Cache misses can only occur when the sorted array itself is longer than the cache line size.

⇒ Cache misses every time over 64 bytes are read or written.

→ So when n < size of cache, the complete array can be loaded into cache at once.

∴ Misses only from merge function

$$= \frac{\text{Total reads + writes}}{64 \text{ (cache line length)}}$$

$$\text{Cache misses (Iterative)} = \frac{2 \cdot N}{64} = N/32$$

For the case where n > size of cache, all layers of merge sort, will have a capacity error.

∴ Cache misses (Iterative) = $\frac{\text{Total read + write}}{\text{Cache line size}}$ (No of iterations)

$$= \frac{2N}{64} \lceil (\log_2 N) \rceil$$

$$= N/32 \lceil \log_2 N \rceil$$

→ This is the case for iterative merge sort, as in it, the array is called every iteration & complete array is present.

For recursive, the complete array is not passed but rather the two arrays to merge are only passed.

⇒ When n > size of cache, hits due to size will only start to occur when the passed array has size > size of cache

∴ For n > size of cache

Cache misses (recursive) = $\frac{\text{Total read + write}}{\text{Cache line size}}$ (Iterations where len(array) > sizeof(cache))

$$= 2N/64 \left(\lceil \log_2 N \rceil - \lceil \log_2 \text{cache size} \rceil \right)$$

$$= N/32 \left(\lceil \log_2 N \rceil - \lceil \log_2 2^{10} \rceil \right)$$

$$= N/32 \left(\lceil \log_2 N \rceil - 10 \right)$$

∴ Cache misses :

	Iterative	Recursive
n > Cache-size	$N/32 (\lceil \log_2 N \rceil)$	$N/32 (\lceil \log_2 N \rceil - 10)$
n < Cache-size	$N/32$	$N/32$

~~Modifying~~

Modifying Merge Sort

Tiling : Tiling is an idea, which is used in tiled mergesort which is also sometimes implemented by compilers & is memory efficient

The idea is to do a normal mergesort till the sizes are $BC/2$ (B - no of keys per cache block, C - Capacity of cache per block) and then return to base mergesort to complete the sorting of the entire array

normal mergesort → iterative merge sort
base mergesort → inplace sorting

Multi - Mergesort: This adds optimization for the second part of killing - mergesort. In this we replace the final $\lceil \log_2 N/(BC/2) \rceil$ merge passes, with a single pass that merges all the pieces together at once.

This adds a lot of complications to the algorithm but has an excellent cache performance.