Cache Aware and Cache Oblivious Algorithms

Coverage

- Cache Aware Algorithms
- Cache oblivious algorithms
- Binary Search
- Divide and Conquer Algorithms

Memory Hierarchy Issues

Modern Memory hierarchy consists of CPU,L1,L2,L3,Main Memory and Hard Disk. Access is electronic except hard disk having mechanical components also. There is a order of difference of latency in cache, main memory and disk Access.

Cost of accessing data = Latency + (Amount of data/Bandwidth)

Bandwidth is almost same between all levels of memory hierarchy

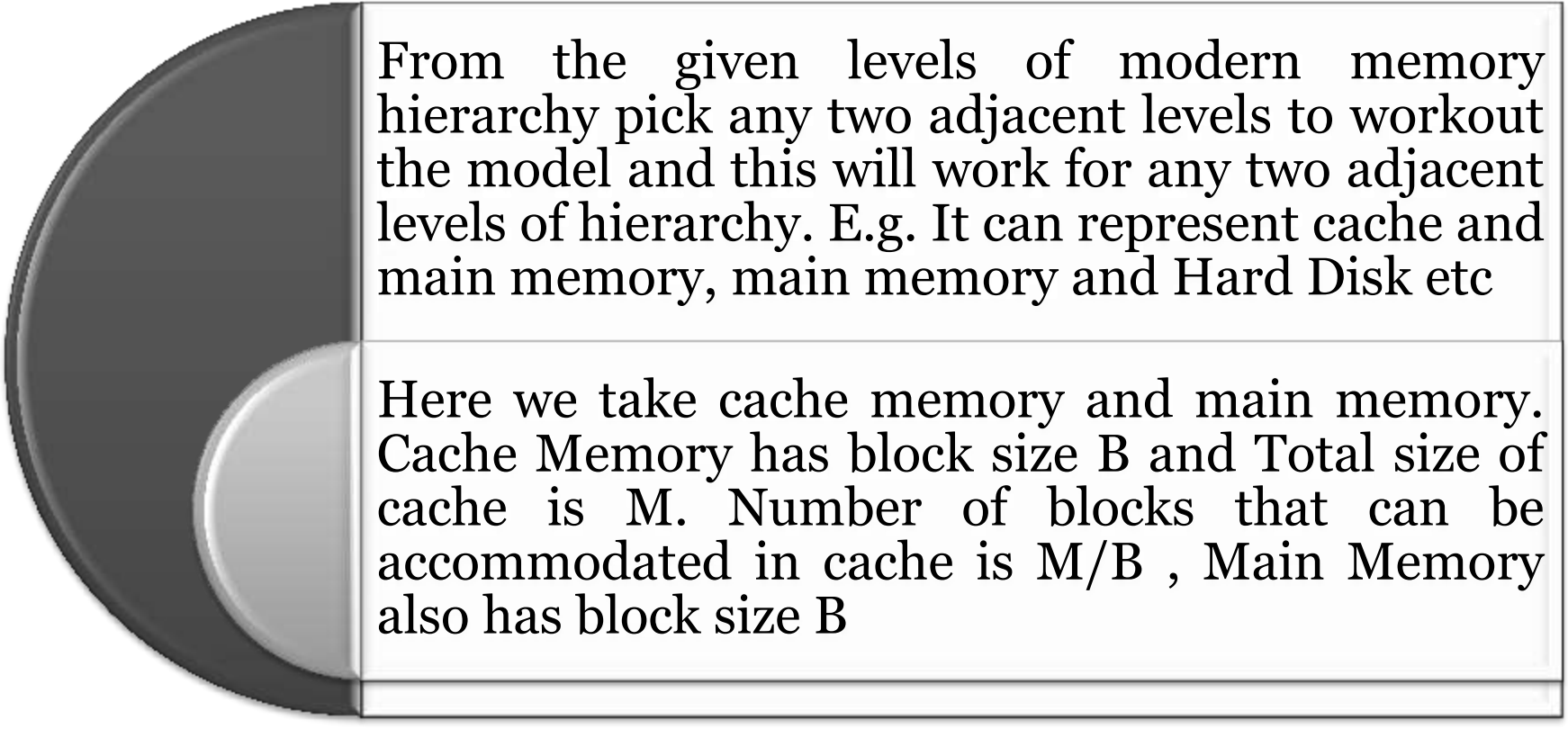
Amortized cost to access one element= Latency/Amount of data+1/Bandwidth

Goal of Cache oblivious algorithms

A) Spatial Locality: Our algorithms should use all elements in a block (unit of data transfer between any two levels)

B) Temporal Locality : Our algorithm should reuse blocks that have been used recently

Two Level Memory Model



From the given levels of modern memory hierarchy pick any two adjacent levels to workout the model and this will work for any two adjacent levels of hierarchy. E.g. It can represent cache and main memory, main memory and Hard Disk etc

Here we take cache memory and main memory. Cache Memory has block size B and Total size of cache is M. Number of blocks that can be accommodated in cache is M/B , Main Memory also has block size B

Cache Aware Model

In our analysis number of blocks that are transferred to and from the cache memory for read and write operations are counted. Even if a single array element is required, whole block containing that element is transferred to cache.

If cache is full some block has to be transferred out which may be used farthest in the future

If MT is the number of Memory Tranfers

$MT(N) = MT_{B,M}(N)$

This type of model is called cache aware model where we know the block size and cache size and we customize our algorithm for that cache size for better efficiency

Cache Oblivious Algorithms

- In cache oblivious algorithms we don't know B and M and still try to improve the cache efficiency. It also means that all algorithms we had done so far without bothering about the size of B and M were cache oblivious algorithms.
- Any Cache oblivious algorithms which is efficient for some 2 levels of hierarchy will be efficient for any two levels of memory hierarchy and in turn will be overall efficient.
- Some array may not be perfectly aligned with the blocks and so may use two blocks even if the size of the array is less than the block size.

Scanning Algorithm

- Scanning(A,N)
- for i ← 1 to N
- visit A[i]
- E.g. sum of elements of an array
- MT(N)=O((N/B) +1)
- For example if there are 10000 array elements with 1 byte size and the block size is 1000 then maximum number of block memory transfers will be 11

Parallel scans-Reverse an array

For i ← 1 to [N/2]

Do exchange A[i] ↔A[N-i+1]

Here we assume that cache memory can hold multiple blocks at the same time because here two blocks containing the respective elements of swapping has to be there.

MT(N)=O(((N/B) +1)

Binary Search

Every time an respective middle element is accessed, a new block will be accessed until we reach a stage when all the elements in the list are in the same block.

$MT(N) = O(\lg(N/B)+1)$

$= \lg N - \lg B + 1$

Because we will be in the same block only for lgB numbers towards the end of the binary search

Divide and conquer algorithms

Algorithm divides the problem down to O(1) size

In cache efficient algorithms

A)Consider points at which problem fits in cache($\leq$ M)

B) Problems fit in O(1) blocks or O(B) of data

Questions, Comments and Suggestions

Question 1

Differentiate Spatial Locality and Temporal Locality

Question 2

Differentiate Cache Aware and Cache Oblivious Algorithms

Question 3

By having the cache act as a bridge between main memory and the CPU, the time it takes to retrieve a block of data from main memory is longer than it would be if the cache did not exist.

A) True

B) False