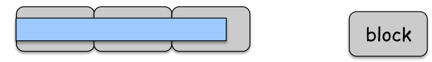
15-853:Algorithms in the Real World

Clarification on cache-oblivious matrix multiplication

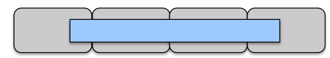
Array storage

How many blocks does a size-N array occupy?

- If it's aligned on a block (usually true for cache-aware), it takes exactly $\lceil N/B \rceil$ blocks



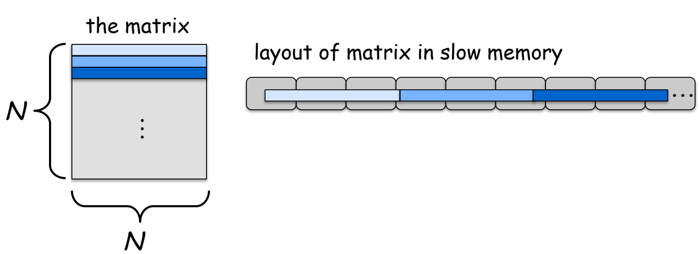
- If you're unlucky, it's $\lceil N/B \rceil+1$ blocks. This is generally what you need to assume for cache-oblivious algorithms as you can't force alignment



- In either case, it's $\Theta(1+N/B)$ blocks

Row-major matrix

If you look at the full matrix, it's just a single array, so rows appear one after the other

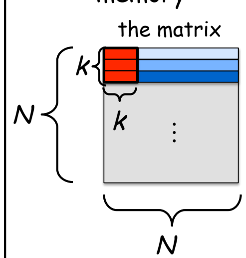


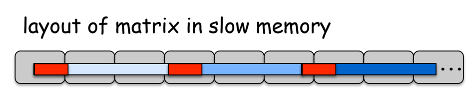
- So entire matrix fits in $\lceil N^2/B \rceil+1=\Theta(1+N^2/B)$ blocks

Row-major submatrix

(This is the part that was probably not clear)

- In a submatrix, rows are not adjacent in slow memory





- Need to treat this as k arrays,
- so total number of blocks to store submatrix is $k(\lceil k/B \rceil+1) = \Theta(k+k^2/B)$

Row-major submatrix

- Recall we had the recurrence

  $Mult(N) = 8\ Mult(N/2) + \Theta(N^2/B)$ (1)

The question is when does the base case occur here? Specifically, does a $\Theta(\sqrt{M}) \times \Theta(\sqrt{M})$ matrix fit in cache, i.e., does it occupy at most M/B different blocks?

- If a $\Theta(\sqrt{M}) \times \Theta(\sqrt{M})$ fits in cache, we stop the analysis at a $\Theta(\sqrt{M})$ size — lower levels are free.

  i.e., $Mult(\Theta(\sqrt{M})) = \Theta(M/B)$ (2)

- Solving (1) with (2) as a base case gives

  $Mult(N) = \Theta(N^2/B + N^3/B\sqrt{M})$

load full submat in cache

Is that assumption correct?

Does a $\Theta(\sqrt{M}) \times \Theta(\sqrt{M})$ matrix occupy at most $\Theta(M/B)$ different blocks?

- We have a formula from before. A $k \times k$ submatrix requires $\Theta(k + k^2/B)$ blocks,
- so a $\Theta(\sqrt{M}) \times \Theta(\sqrt{M})$ submatrix occupies roughly $\sqrt{M} + M/B$ blocks

- The answer is "yes" only if $\Theta(\sqrt{M} + M/B) = \Theta(M)$.
  - iff $\sqrt{M} \le M/B$, or $M \ge B^2$.
- If "no," analysis (base case) is broken — recursing into the submatrix will still require more I/Os.

Fixing the base case

We saw 2 fixes in class

1. Just assume the tall cache assumption that $M \ge B^2$. Then the base case is correct, completing the analysis.
2. Change the matrix layout to Z-morton:
   - In this layout, a submatrix *is* contiguous.
   - Here a recursive $k \times k$ submatrix is a single subarray (fitting in $\Theta(1+k^2/B)$ blocks) not the multiple arrays (requiring $\Theta(k+k^2/B)$ blocks) we had before.