Visual Guide to 2D Stencil Memory Optimization

CS250B
Sang-Woo Jun
Cache Efficient Processing in 1D: Trapezoid Units

- Computation in a trapezoid is either:
  - Self-contained, does not require anything from outside ( ), or
  - Only uses data that has been computed and ready ( , after )

We will do 2D!!
Goal: Fill out temp and then have results at the bottom of temp.
No Dependencies For Corner
Calculate Blocks With Satisfied Dependencies

Edges only depend on corner

Next line only depends on edges

Block size
All Done With Trapezoids
Fill Out The Rest (Upside-down wedges)
This Was Not Cache-Oblivious

• We had to choose a block size
  • “temp” is divided into a grid of BLOCK_SIZE width and height sub-blocks
  • Depending on the location (corner? edge? middle?) 3D shape determined
  • After filling in all grids, we fill in the upside-down edges

• Cache-oblivious algorithm instead divides the space into four quadrants recursively
  • Actual shape determination and filling it out happens only at a very small block size
  • Actually more complex than this, but having a small SUBSTEP parameter restricts the problem space
SUBSTEPS may be too large

• If the given substeps parameter is too large to be used as-is as the height of the 3D structure, remember you can also break that down into smaller steps!