Cache Oblivious Stencil Computations (2005)

Written by: Matteo Frigo and Volker Strumpen
Presentation by: Ricardo Gayle Jr.
What is a stencil?

- A computation that updates elements of an array or grid according to some fixed pattern
  - The pattern is described as a computational kernel
- The **stencil** defines what an element is at time $t$ as a function of other elements at time $t - 1, \ldots, t - k$
- Can be applied to any $n$-dimensional grid
- When we include a time dimension, we have a $(n+1)$-dimensional spacetime grid
  - Every spacetime point, except for initial and boundary values, are computed by the computational kernel
Stencil computations and Caches

- A **stencil computation** is any traversal of spacetime that respects data dependencies of the stencil.
- Simplest computes all points at $t$ before any of $t+1$.
- If $|p| > \text{cache size}$, cache misses proportional to $|p|$.
- Storing a bounded number of time steps per space point is usually sufficient, rather than the entire spacetime.
- This idea used create a cache-oblivious algorithm.
- Cache-oblivious means the algorithm does not know cache size but uses cache optimally.
What’s so special?
Well, the fact that it’s not
One-dimensional Stencil Algorithm

- 3-point stencil
- Base Case: if height == 1
  - Call kernel on all points
- If width > 2 * height * $\sigma$:
  - Space cut
    - Cut ensures each subproblem is a valid, non-empty trapezoid
- Else:
  - Time cut
- The traversal is valid because no points in $T_1$ depends on $T_2$, so it follows stencil dependencies!
Multi-dimensional Algorithm

- Allow any stencil where spacetime point \((t + 1, x)\) can be dependent on all points \((t, x + k)\) where \(|k| \leq \sigma\)
- Arbitrary-dimensional space time
- Idea: “Informally, for each dimension \(i\), the projection of the multi-dimensional trapezoid onto the \((t, x^{(i)})\) plane looks like the 1-dimensional trapezoid”
- Perform space cuts in any dimension that allows it; time cut otherwise
Cache Complexity

- We assume that the kernel operates “in-place”, the cache is “ideal” and the trapezoid is larger than the cache
  - “In-place” kernels are very common
  - Fully associative, optimal replacement, cache of two level memory system
- Lemma 1: For trapezoid $T$, let $m$ be the minimum width in any dimension divided by 2. There are $O((1+n)Vol(T)/m)$ points on the surface
- Theorem 2: On an ideal cache of size $Z$ and a “large” trapezoid, the algorithm incurs $O(Vol(T)/Z^{1/n})$ cache misses, as opposed to $O(Vol(T))$.
  - When a subproblem $S$ gets small enough, it incurs $O(Vol(S))$ cache misses
  - Because of how we divide to get $S$, there are bounds on the height of $S$ which allow for $O(Vol(S)/Z^{1/n})$ cache misses per subproblem
Questions and Comments?