Orthogonal Scheduling of Stencil Computations with Chapel Iterators

Ian J. Bertolacci

Advisers
Catherine Olschanowsky, Michelle Mills Strout
Problem

```
for t in 0..T {
    for x in 1..N do
        A[t,x] = (B[t,x-1] + B[t,x] + B[t,x+1])/3;
    A <=> B;
}
```

- Stencil computations are everywhere
  - Partial Differential Equations
  - Image Processing
  - Cellular Automata
- Naïve parallelization, can be faster than serial
  - Does not scale with the addition of cores!
Diamond-Slab Tiling

- Parallelism
- Data Locality
  - Cache re-re-re-use
int write, read;
int t0, t1, x0, x1, dx0, dx1;
int t, x;
for( t0 = 1; t0 <= T; t0 += timeBand ) {  
for( t1 = min(t0 + timeBand - 1, T); t1 = min(t0 + timeBand - 1, T);  
dx0 = 1;

for( x0 = tiles_A_start; x0 <= upperBound; x0 += betweenTiles ){
  x1 = x0 + width_max - 1;
  read = (t0 - 1) & 1;
  write = 1 - read;
  if( x0 <= lowerBound ) {
    for( t = t0; t <= t1; ++t ){
      int minVal = min(x1 + dx1 * (t - t0), upperBound );
      stencil( read, write, x );
      read = write;
      write = 1 - write;
    }
  } else if( x1 >= upperBound ){
    for( t = t0; t <= t1; ++t ){
      int minVal = min(x0 + dx0 * (t - t0), upperBound );
      stencil( read, write, x );
      read = write;
      write = 1 - write;
    }
  } else {
    for( t = t0; t <= t1; ++t ){
      int minVal = min(x0 + dx0 * (t - t0), upperBound );
      for( x = max(x0 + dx0 * (t - t0), lowerBound); x <= upperBound; ++x)
      stencil( read, write, x );
      read = write;
      write = 1 - write;
    }
  }
  dx0 = -1;
dx1 = 1;
  for( x0 = tiles_B_start; x0 <= upperBound; x0 += betweenTiles ){
    x1 = x0 + width_min - 1;
    read = (t0 - 1) & 1;
    write = 1 - read;
    if( x1 >= upperBound ){
      for( t = t0; t <= t1; ++t ){
        for( x = max(x0 + dx0 * (t - t0), lowerBound); x <= upperBound; ++x)
        stencil( read, write, x );
        read = write;
        write = 1 - write;
      }
    } else {
      for( t = t0; t <= t1; ++t ){
        int minVal = min(x1 + dx1 * (t - t0), upperBound );
        for( x = max(x0 + dx0 * (t - t0), lowerBound); x <= minVal; + +x)
        stencil( read, write, x );
        read = write;
        write = 1 - write;
      }
    }
  }}}}
Diamond-Slab Tiling

```c
int write, read;
int t0, t1, x0, x1, dx0, dx1;
int t, x;
for( t0 = 1; t0 <= T; t0 += timeBand ) {
  t1 = min(t0 + timeBand - 1, T);
  dx0 = 1;
  dx1 = -1;
  for( x0 = tiles_A_start; x0 <= upperBound; x0 += betweenTiles ){
    x1 = x0 + width_max - 1;
    read = (t0 - 1) & 1;
    write = 1 - read;
    if( x0 <= lowerBound ) {
      for( t = t0; t<= t1; ++t ){
        int minVal = min(x1 + dx1 * (t - t0), upperBound);
        for( x = lowerBound; x <= minVal; ++x)
          stencil( read, write, x );
      read = write;
      write = 1 - write;
    }
    else if( x1 >= upperBound ){
      for( t = t0; t<= t1; ++t ){
        int minVal = min(x1 + dx1 * (t - t0), upperBound);
        for( x = lowerBound; x <= minVal; ++x)
          stencil( read, write, x );
      read = write;
      write = 1 - write;
    }
  }
  else {
    for( t = t0; t<= t1; ++t ){
      int minVal = min(x1 + dx1 * (t - t0), upperBound);
      for( x = max(x0 + dx0 * (t - t0), lowerBound); x <= upperBound; ++x)
        stencil( read, write, x );
      read = write;
      write = 1 - write;
    }
  }
}
for( read, write, x ) in diamondSlabIterator(tileSize, domainSpace, stencilDepth){
  stencil( read, write, x );
}
```

forall (read, write, x) in diamondSlabIterator(tileSize, domainSpace, stencilDepth){
  stencil( read, write, x );
}
Current Findings

- It works!
  - We observe speedups over serial C:

<table>
<thead>
<tr>
<th>Language</th>
<th>Naïve Parallel</th>
<th>Diamond-Slab Tiling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapel</td>
<td>5.96x</td>
<td>6.85x</td>
</tr>
<tr>
<td>OpenMP + C</td>
<td>7.70x</td>
<td>13.05x</td>
</tr>
</tbody>
</table>

- It’s good code!
  - Manageable
  - Meaningful
  - Magni-*fast*-cent
Future Work

- Lets greet and beat OpenMP + C performance
- Efficient, domain generalizable iterators
- Automated tile size calculations; not experiments