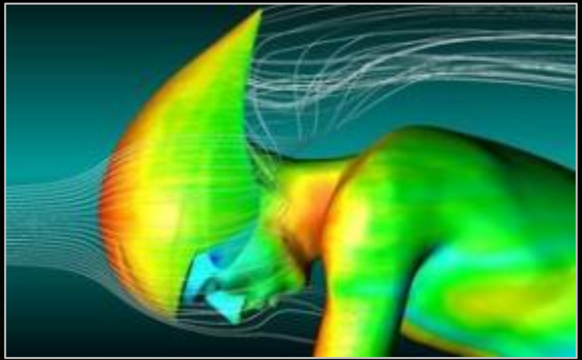
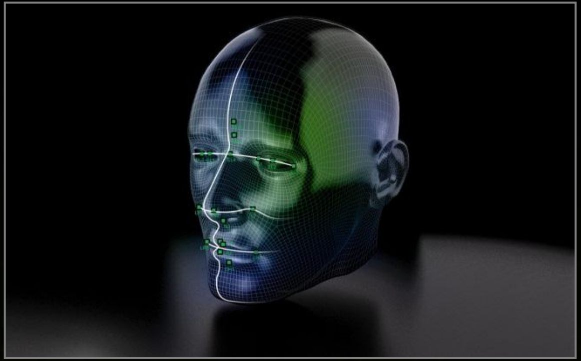
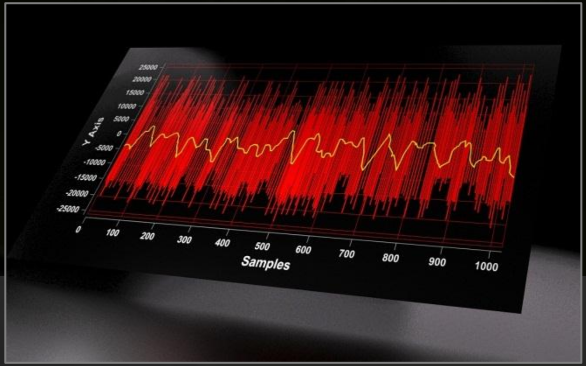
Tesla

GPU Computing





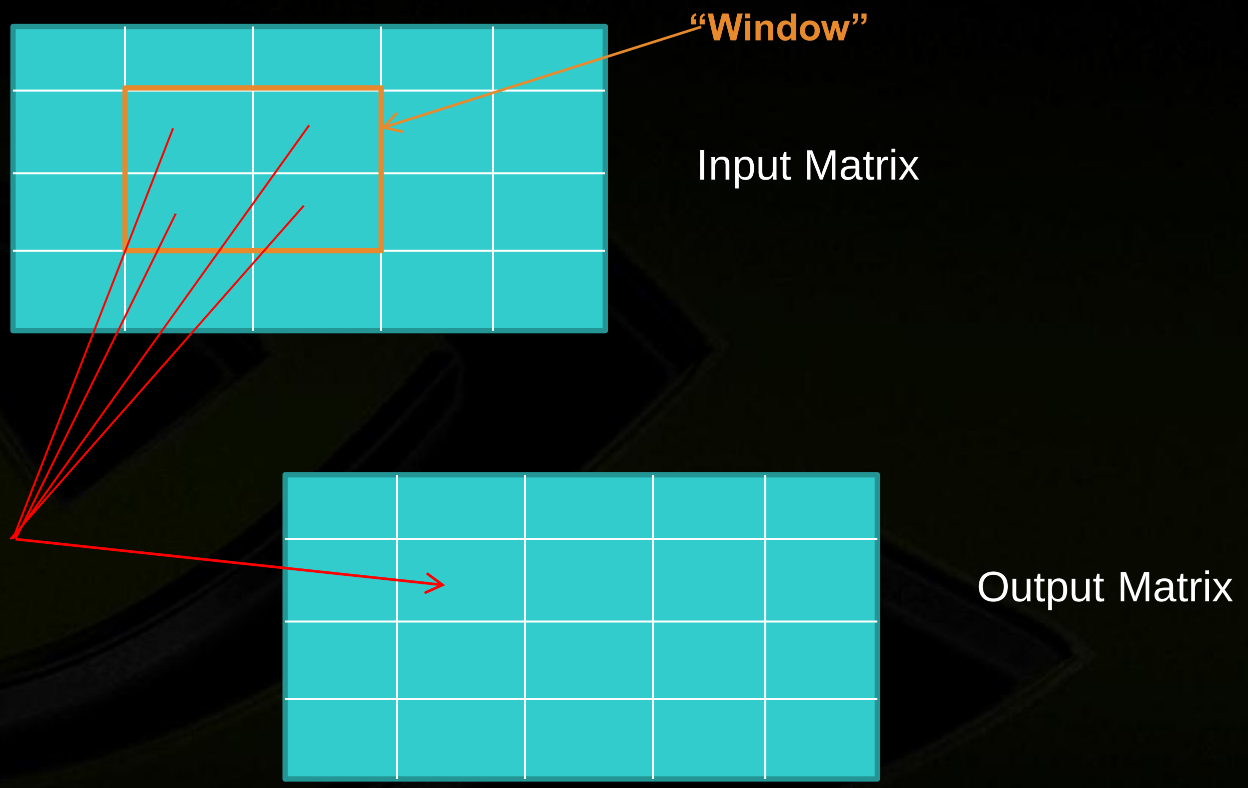


Directives Deep Dive (using C)

Finding Parallelism in your code

- (Nested) for loops are the best example
- Large pieces of work are needed to offset GPU overhead
- Code must be parallelizable – typically means iterations of the for loop must be independent of each other
- Compiler must be able to figure out sizes of data regions
- Pointers and pointer arithmetic should be avoided if possible
- Best to use subscripted arrays, rather than pointer-indexed arrays.
- Any function calls within the accelerated region must be able to be inlined.

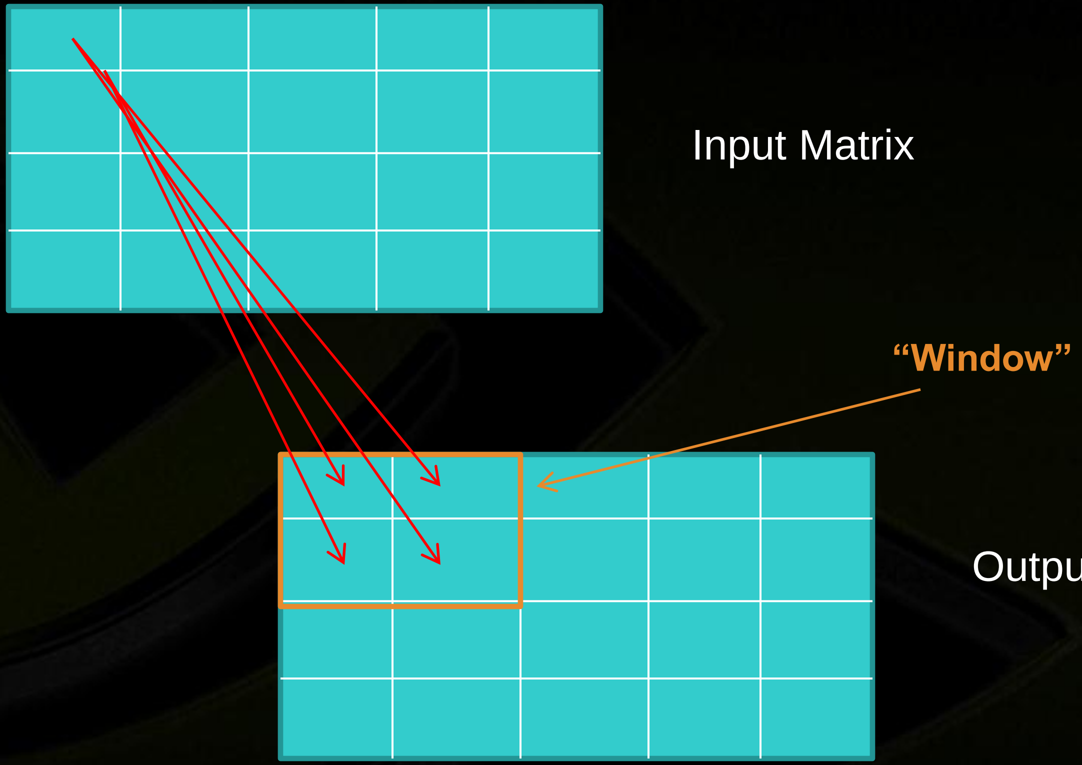
Window Minimum Example



We iterate over the output matrix, reading from each input point several times and **writing to each output point once**.

Output(i,j) = min(Output(i,j), Inp(i,j), Inp(i,j+1), Inp(i+1,j), Inp(i+1,j+1)…);

Window Minimum – Alternate Realization



Output Matrix

We iterate over the input matrix, reading from each input point once and **writing to each output point several times (perhaps)**.

Output(i,j) = min(Output(i,j), Inp(i,j));
Output(i,j+1) = min(Output(i,j+1), Inp(i,j));
Output(i+1,j) = min(Output(i+1,j), Inp(i,j));
Output(i+1,j+1) = min(Output(i+1,j+1), Inp(i,j));
….

Code Sample 1

Accelerator Directive

```
#pragma acc region
{
 for(i=0; i<(nx-(wx-1)); i++){
   for(j=0; j<(ny-(wy-1)); j++){
// loop over the  window
     for (sx=0; sx<wx; sx++){
       for (sy=0; sy<wy; sy++){
           // find the minimum value over the window and store in node(i,j)
             if (node[(j + sy) + ((i+sx)*ny)] > cell[j + (i*(ny-1))]) node[(j + sy)
+ ((i+sx)*ny)] = cell[j+(i*(ny-1))];
             }
           }
         }
       }
}
```

Nested for loops

YUCK!

Compiler is sad!

Code Sample 2

Accelerator Directive

```
#pragma acc region
{   // loop over the data set
    for(i=0; i<(nx-(wx-1)); i++){
        for(j=0; j<(ny-(wy-1)); j++){
          tempnode = node[i][j];
    // loop over the minimization window
          for (sx=0; sx<wx; sx++){
            for (sy=0; sy<wy; sy++){
            // find the minimum value over the window and store in node(i,j)
              if (tempnode  > cell[i+sx][j+sy]) tempnode = cell[i+sx][j+sy];
              }
            }
          node[i][j] = tempnode;
          }
        }
}
```

Nested for loops

√ Independent Loop Iterations

√ Nice array subscripting

√ No pointer arithmetic

Compiler is happy!

Process to follow

- Follow the basic rules for identifying parallelizable code
- Drop in directives
- Compile with appropriate flags (-ta=nvidia,cc20 -Minfo)
- Look at compiler info output
- Rewrite code
- Repeat
- Benchmark the code when you have a loop that is parallelized

Example 1

- Bad code example (grid2o.c)
- Getting better (grid2.c)
- -Msafeptr – discussion of data management
- Basic Data movement directive:
    - #pragma acc data region copyin(…) copy(…)

Directives categories

- Accelerator control (#pragma acc region …)
- Accelerator hints (#pragma acc data …)
- Data management
- Device control

Tips and Tricks

- Use time option to learn where time is being spent
  - -ta=nvidia,cc20,time
- Eliminate pointer arithmetic
- Inline function calls in directives regions
- Use Contiguous memory for multi-demensional arrays
- Use Data regions to avoid inefficiencies
- Conditional compilation with ACCEL keyword
- More: http://www.nvidia.com/docs/IO/117442/Top-12-Tricks-for-Maximum-Performance-C.pdf

Getting started

- www.nvidia.com/gpudirectives
- Download PGI tools
- Up to 30 days free usage (trial license)
- Documentation
- User forums
- All features available to Fortran users as well