Tools for source - source translation

Thiyagarajan C
Department of Electrical Engineering
Colorado State University

Mentors: Jose Garcia,
: John Dennis
Outline

Motivation
Tools found
Polyhedral Model
Details on each tools
Conclusion
Why Tools?

- Manual Translation takes time
- CUDA C is difficult to read and debug
- The conversion process takes considerable amount of time
The Destination

Fortran

Optimized CUDA
Tools found

- F2C-ACC – Converts Fortran to C and Fortran to CUDA
  - Mark Govett, NOAA
  - Mark.W.Govett@noaa.gov

- PPCG – Converts C to CUDA
  - Sven Verdoolaege, Ohio State University
  - skimo@kotnet.org

- PLUTO – Converts C to optimized C
  - U. Bondhugula and others from Ohio State University
  - http://www.ece.lsu.edu/jxr/pluto/index.html

- Par4All – Converts Fortran to OPENMP and C to CUDA
  - http://www.par4all.org/
The Polyhedral Model

• Most of the tools mentioned are based on the Polyhedral Model.

• Polyhedral Model is a mathematical framework for program optimizations and transformations.

\[ S[j,i] = S[j-1,i] + S[j,i-1] + S[j-1,i+1] \]
F2C-ACC

Features
• Converts F77 and F90 to C, (Not all features)
• Converts Fortran to CUDA
• The error messages are accurate
• Generates host code and device code

Limitations
• All user-defined macros should be expanded with a preprocessor
• Limited modules support in Fortran 90
• Doesn’t support the advanced features in Fortran 90 and many intrinsic functions
• No conditional thread execution
**F2C-ACC Directives**

**Fortran – C**
- No directives are required for conversion.
- The code should be compatible with the tool.
- The modules used in the program should be preprocessed.
- F2C-ACC --Generate=C [–FileType=Module] input.f90

**Fortran – CUDA**

```fortran
!ACC$REGION (<nx+2>,<ny+2,nzm>) BEGIN
!ACC$DO PARALLEL (2)
do ivl=1,nvl
!ACC$DO PARALLEL (1)
  do j=1,nx
!ACC$DO VECTOR(1)
    do i=1,ny
      end do
    end do
  end do
!ACC$REGION END
```
CUDA

• Program decomposed into blocks of threads.
• Thread blocks execute on individual multiprocessors.
• Thread blocks execute independently of one another.
• Threads within blocks coordinate through shared memory.
• All threads can access global memory and texture MemoryGrid.

Grid

Block (0,0)  Block (1,0)  . . . . . .  Block (N,0)
Block (0,1)  Block (1,1)  . . . . . .  Block (N,1)
**F2C-ACC pros and cons**

**Pros**
- Converts Fortran to CUDA or C
- The generated C code almost works!
- Generates host code for the CPU
- Control over loop allocation to specific grid or block dimensions
- Simple to use and understand

**Cons**
- The tool doesn’t do any optimizations or program transformation
- The CUDA code in most cases requires hand modifications to make it work
- The code is not readable
- The generated CUDA Scalar Advection doesn’t work
- The threads should be managed manually.
Where are we?

Fortran

F2C

C  Optimized CUDA  CUDA
PPCG

Features
- Converts C to CUDA C with certain level of optimization
- Works only on affine loops
- Generates device code and host code.
- Supports user defined macros.
- Can detect legal parallelization using the polyhedral analyzer

Limitations
- Have to convert Fortran to C either manually or using other tools
- No scalar variables supported.
- The tool is highly experimental. So the heuristics don’t work as expected.
- Manually unroll loops in case of conditional indices.
- Works on very small set of programs and will not be able to handle atmospheric models as of now.
PPCG tool directives

**C - CUDA**

```c
#pragma scop
  .
  // Affine loops
  .
#pragma endcop
```

- `ppcg [options] input.c`
- The grid and block dimensions can be specified using `grid.sizes` and `block.sizes`. If not specified the tool will allocate the configuration based on heuristics (presently doesn’t work).
- The amount of shared memory is specified using `tile.sizes`. 
Loop unrolling

```c
for(k=0;k<nzm;k++)
{
    kb=0>(k-1)?0:(k-1);
    f[k]=w[kb];
}
```

// manual unroll
```c
for(k=0;k<1;k++){
    f[k]=w[k];
}
for(k=1;k<nzm;k++){
    f[k]=w[k-1];
}
```
Affine loops

//Non-Affine
kb=nx;
for(k=0;k<N;k++){
    f[k,kb]=w[k];
}

//Affine
for(kb=nx; kb<nx+1;kb++)
    for(k=0;k<N;k++){
        f[k,kb]=w[k];
    }
PPCG pros and cons

Pros
• A single pragma takes care of the parallelization
• The produced code is readable provided the right options are used
• Can detect whether parallelization is possible

Cons
• The heuristics are not fully developed. The tool is at its infancy
• Supports only a single scop
• The tool cannot handle complex programs
• There is no direct read from global memory
• Will not be applicable in the near future
Where are we?

Fortran

C

Optimized CUDA

CUDAOptimized CUDA

F2C

PPCG
Pluto

Features
• Converts C to optimized and parallelized C
• Works only on affine loops
• Parallelization is performed using OpenMP
• Provides a lot program optimization features like fusing, tiling, tiling for L2 cache.

Limitations
• The arguments passed to a macro cannot have expressions
• All the scalar variables should be converted to match the depth of the loops in which they are specified.
Pluto Directives and Results

**Syntax**
- The Directive are same as PPCG

**Results**

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Optimized</th>
<th>Tiled</th>
<th>Parallelized with 4 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advect3D routine</td>
<td>6.3</td>
<td>6.3</td>
<td>6.3</td>
<td>4.3</td>
</tr>
<tr>
<td>Speed up</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>36%</td>
</tr>
</tbody>
</table>
Pluto pros and cons

**Pros**
- The output from the tool is readable
- Macros work
- Parallelization with OpenMP works

**Cons**
- The tiling and fusing don’t work as expected in most cases
- Cannot call macros with expressions as arguments
- Conditional indices are treated as serial loops
- The fusing algorithm cannot handle complex loop structures
The complete map?
The complete map

- Fortran
- C
- Fortran Optimized CUDA
- CUDA
- Opt-Par C
- C PPCG
- Pluto
- F2C
- Programmers
- F2C
Conclusion

- Most of the tools don’t work with real code
- The translated code is not readable
- Hand modifications required
- The heuristics are not sophisticated enough to detect optimizations
- All the language features are not supported
- Brief Documentations
Questions
Email
thambu2003@live.in

Thanks