DSL Developments

Julian M. Kunkel

DKRZ

23-02-2015
Outline

1. Introduction
2. Results from ICOMEX
3. New Project: AIMES
4. Summary
Introduction

Motivation

- General purpose programming languages face hurdles to express scientist’s thoughts well
- In HPC, scientists code domain knowledge but also performance aspects (computer science knowledge)
  if possible

ICON

- ICON (climate and NWP) model
- Ocean, Atmosphere, Sealce, ...
- MPI-M led approach to DSLlize the code in the ICOMEX project
Why Manual Optimization?

General Purpose Programming Languages

- Standard implies limitations, e.g. memory layout is fixed
- Difficulty to express architecture-specific attributes
- Existing tools: CPP macros, templates from C++

Compilers

- Need to follow the standard (conservatively)
- Uncertainties (data alignment, array size) \(\Rightarrow\) suboptimal code
- Cannot change memory-layout (1D array vs. 3D indirect array)

Consequence

- Architecture/compiler-specific branches of code
- CPP directives to select the system to build for
- Directive based approach (OpenMP, OpenACC) bloats the code
Example Operator in Fortran

```fortran
!ICON_OMP_PARALLEL_DO PRIVATE(edge__index, edge__level,
    ↪ edge__startIndex, edge__endIndex) SCHEDULE(static,2)
DO edge__block = edge__subset%startBlock, edge__subset%endBlock
!

get the start/end index in the block
edge__startIndex = 1
edge__endIndex = edge__gridEntity%blockSize_2D
IF (edge__block == edge__subset%startBlock) &
    edge__startIndex = edge__subset%startBlockIndex
IF (edge__block == edge__subset%endBlock) &
    edge__endIndex = edge__subset%endBlockIndex
DO edge__index = edge__startIndex, edge__endIndex
    DO edge__level = 1,
        → edge__gridEntity%numberOfLevels(edge__index,edge__block)
flux%value(edge__index,edge__block,edge__level) = div%value(
    → cell__ofedge_index_p(edge__index,edge__block,1),cell&
    ↪ edge__level) *
    → grad_coeffs%value(edge__index,edge__block,edge__level,&
    &1) + div%value(
    → cell__ofedge_index_p(edge__index,edge__block,2),cell__ofedge_block_p(edge__block,2),edge__level) *
    → grad_coeffs%value(edge__index,edge__block,edge__level,2)
END DO
END DO
END DO
!ICON_OMP_END_PARALLEL_DO
```
Issues

- Code is readable only for experts
  - Original code is re-formatted and comments purged
  - Handling of special cases
  - Memory layout is optimized for parallelism (block structure, indirect access)
  - Additionally: Different versions of the code exist based on the connectivity...
DSL Version of the Operator

```plaintext
1 <on edge do:
2   edge%flux = SUM[on cell] cell%div * cell%grad_coeffs;
3 end do>
```

Additional benefit

- Alternative (system-specific) memory layouts are possible
- Domain-specific variation in connectivity level is expressable (SUM[] operator)
Example: Full DSL Code

So far, declaration of variables have been omitted.

```
SUBROUTINE grad_oce_3D_dsl_2(div, flux, grad_coeffs,
                            \subset subset_range)
  <OnCells_3D_double :: div>
  <OnEdges_3D_double :: flux>
  <OnEdgesToCells_3D_double :: grad_coeffs>
  <Edges_SubsetRange, INTENT(in), OPTIONAL :: subset_range>
  <Edges_3D_SubsetRange :: edges_subset>
  <Edges_3D_Element :: edge>
  <CellsOfEdges_3D_Element :: cell>

  <edges_subset = getDefaultSubset(subset_range, flux)>
  <edge .belongsTo. edges_subset>
  <cell .belongsTo. edge>

  <on edge do:
    edge%flux = SUM[on cell] cell%div * cell%grad_coeffs;
  end do>

END SUBROUTINE grad_oce_3D_dsl_2
```
Results from ICOMEX

Most project runtime, ANTLR and ROSE have been used

Moderate success

- Source-to-Source translator (Fortran+DSL $\Rightarrow$ Fortran)
- Arrays could be swapped, e.g. $x[i][j][k]$ became $x[j][k][i]$
- Inlining was possible
- Configuration file

Issues

- Tools are complex by themselves
- ANother Tool for Language Recognition (ANTLR) does not offer Fortran support
- ROSE Fortran support is limited and required workarounds
- Issues with pre-processor macros
Recent Approach

**Idea**
Parse and alter only text regions that matter for us

**DSLL**
Light-weight tool for template processing in Python supporting
- Symbol-table
- Hooks to invoke actions in the tool
- Nested namespace and templating
- Flexible templates
- Command line options can alter templates
- Incremental DSLization \(^1\)

It can handle our example from the beginning!

\(^1\) if memory layout is not modified
Example Code for a Synthetic Test

```
1 Grid :: myGrid
2 GridVar :: varCreation
3 for c in myGrid do:
4   ! Print the value of var for each grid point
5     print *, c%var
6 end do
```

Example source code to translate with dsll
Example Template

```
OPTIONS = [
    ("debugging", "Enable extra debugging", False),
    ("size", "Dimension of the problem", 10) ],

TEMPLATE = [
    ("Grid $extra=\[^:\]+\::$:: $var$",
     """TYPE(grid3D) $extra$ :: $var$
     integer :: index_$var$
     ~~LOOKUPTABLE_SET($var$,Grid)""
    ),
    ("GridVar $extra=\[^:\]+\::$:: $var$",
     "real gridVar dimension(@VAR(size)@):: $var$
     → ~~LOOKUPTABLE_SET($var$,GridVar)""
    ),
    ("for $cell$ in $Grid:grid$ do :
     {"substitute", """DO index_$grid$ = 0, @VAR(size)@
      @if(debugging) print *, index_$grid$@ ~~BEGIN_BLOCK""
      "childs" :[
       "$cell%$var$, "$var$[index_$grid$]",
       "end do", "END DO~~END_BLOCK"
      ]
     }
    ),
    ("for": "~~ERROR(Invalid syntax)"
]
```

The compiler cannot optimize all code
New Project: AIMES

**AIMES**

**Advanced I/O and Computational Methods for Earth-System Models**

**I/O**

- Data layouts for ICO data
- Lossy compression (interface, methodology, schemes)

**DSL**

- Common (meta-)DSL for multiple (earth-system) models
- Tools for flexible creation of “Dialects”
- Full memory abstraction
- Source-to-source translation to existing language AND DSLs
Abstraction Level

Clear separation of concerns

Domain science

Meta-DSL

Dialects

DYNAMICO ICON NICAM

Tools

Scientific programmer

CPU OpenACC Physis GRIDTools ...

back-ends

Computer science

existing tools

drivers ... DSL tools & infrastructure

Domain science

Scientific programmer

Computer science
Summary & Conclusions

- DSLs can simplify code significantly
- Allow separation of concern (DS, SP, CS)
- Existing heavy-weight tools are not well suited/trusted
- In AIMES, we go for high-level concepts