A Suite of Domain-Specific Visual Languages For Scientific Software Application Modelling

Mohamed, John, Richard, David, Willem & Owen
Outline

- Problem / Motivation
- Approach
- Example Usage
- Future work
Scientists are increasingly developing complex software for data analysis. Most are not trained programmers. Many are using complex software platforms and techniques, e.g. distributed & parallel programming, GPUs, etc - that are hard even for experienced CS grads to use.

Approaches to address range from packaged software (Lack flexibility), DSLs (Also Flexibiity/Domain issues), programming patterns and toolkits (still complex).

Still lack high-level, human-centric support; still really hard to develop high quality software for scientific apps.
Motivation – how scientists design their applications...

\[ \mathbf{F}_i = \sum_{j}^{N_m} \mathbf{F}_{ij}(\mathbf{r}_i, \mathbf{r}_j) - \nabla_i \left( -\frac{1}{2} \sum_{j}^{N_m} \mathbf{E}_{ij}^0 \cdot \mathbf{\mu}_{ij}^{\text{ind}} \right) + \sum_{j}^{N_m} \sum_{k}^{N_m} \mathbf{F}_{ijk}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k), \]

**Part 1**
Initialization:
- Assign number of particles, energy, volume, etc.
- Assign coordinates \((r_i)\) at time \((t) = 0\).
- Assign molecules an initial velocity \((v_i)\).
- Scale \(v_i\) consistent with the energy of the system.
- Assign any other \((\partial r^n_i / \partial t^n)\) values \((n > 2)\).

**Part 1.1**
Assign number of particles, energy, volume, etc.
- Assign coordinates \((r_i)\) at time \((t) = 0\).
- Assign molecules an initial velocity \((v_i)\).
- Scale \(v_i\) consistent with the energy of the system.
- Assign any other \((\partial r^n_i / \partial t^n)\) values \((n > 2)\).

**Part 1.2**
Initial force calculations:
- Calculate forces \((f_i)\) on each atom from all \(j\).
- Calculate acceleration \((a_i)\) for each atom.

**Part 2**
Simulation Process:
**loop**
- Part 2.1
  - Integrate equations of motion:
    - Part 2.1.1
      - Calculate \(f_i\) on each atom from all other \(j\).
    - Part 2.1.2
      - Apply integrator to update \(r_i, v_i, a_i (\partial r^n_i / \partial t^n)\)
- Part 2.2
  - if \((t > t_{\text{Equilibration}})\)
    - Accumulate averages.
  - else if \((\text{mod}(m, \text{scalingInterval}) = 0)\)

**Input**

File READ

Digital to Analog

Unpack

Time Series

Convolvere Filterbank

Time Series

Detect

\[ P[n] = \text{amp}[t] \times [\text{data}[n]] \]

\[ N[n] = \text{amp}[t] \times [\text{data}[n]] \]

\[ \text{output}[n] = \text{fft}[\text{data}[n]] \]

\[ \text{fft}[\text{data}[n]] \times N[n] \]

\[ \text{fft}[\text{data}[n]] \times N[n] \]

\[ \text{fft}[\text{data}[n]] \times N[n] \]

\[ \text{fft}[\text{data}[n]] \times N[n] \]

\[ \text{fft}[\text{data}[n]] \times N[n] \]
Approach

- Support scientists – and developers! – to model their applications at multiple levels of abstraction – domain right down to detailed C/C++/GPU kernel code

- Use set of user-defined and reusable DSVLs to model

- Provide web-based environment including DSVLs designer, coding, debugging, linking DSVL views

- Provide semi-automated support for generating lower-level models, generate code/code annotations, reverse-engineer higher-level models from (existing) code
Development approach to support

Algorithm / Program

Data decomposition  Task decomposition

Grouping and Dependency analysis

Parallelization Algorithm

Parallel Program

Parallel Patterns
Deployment Model
Programming Model
Our Approach

1. DSVL Designer
2. Workflow DSVL
   - Formulae DSVL
   - Atomic Model DSVL
3. Code Templates
   - Pseudo-code DSL
   - Parallelism DSVL
4. Deployment DSVL
5. (5)
6. (6)
7. Run Applications on HPC Platform
8. Visualization DSVL
Example

\[ C = A \times B, \quad C_{i,j} = \text{Sum}(A_{i,k} \times B_{k,j}) \]
Web-based development tool
Tool

- Parallel Program Designer
  - Parallel Program Model
    - Code Generator
      - OpenCL Code Generator
      - MPI Code Generator
      - Open MP Code Generator
    - Parallel Program
      - Compile
      - Deploy
        - Many-Core
        - Multi-Core
        - GPU
          - Shared Memory
          - Distributed Memory
Evaluation

- Define set of DSVLs to model domain (molecular simulation and signal processing for radio telescopes), architecture (GPU and MPI-based CPU), parallelism, processing models (MapReduce etc)

- Model hand-implemented programs at multiple levels of abstraction including links between models

- Generate C and OpenCL code

- Reverse-engineer (parts of) models from C code

- 3 scientists validating approach iteratively
Summary & Future work

- Integrated web-based development environment for GPU-based (and other) scientific applications
- Flexible DSVL designer with pre-packaged DSVLs and user-defined DSVLs
- Semi-automated roundtrip engineering support: model-> code-> model

Working on:
- Improve generation/reverse engineering – C/C++, OpenCL
- Patterns and critics to guide users, analyse models/code
- Visualisation of running GPU code in models
- “Liveness”…? 😊
Questions?

Mohamed Almorsy
malmorsy@swin.edu.au
References


