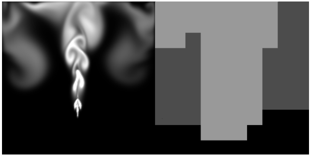
Automatic Generation of Adaptive Simulation Codes

Cédric Bastoul[1] and César Sabater[2]

[1] University of Strasbourg and Inria, France
[2] Universidad Nacional de Rosario, Argentina



December 8th, 2015

Compiling Simulation Applications

About compilers:

- Hide the complexity of the target architecture
- Translate and optimize any application
- Sacred rule: preserve program semantics

About simulation applications:

- Imitate the behavior of a system over time
- Implement the abstract model describing the system
- Fact: never imitate exactly, work within a bounded error

- Strict semantics preservation conflicts with optimization of intrinsically approximated computation

Optimizing Simulation Applications

- Preserving accuracy:
  - Parallelization : vectorization, thread-level, accelerators...
  - Optimization : data locality, register pressure...
  - Well addressed by automatic approaches
- Not preserving accuracy:
  - Less precise models
  - Less accurate computations
  - Adaptive techniques
  - Not well addressed by automatic approaches

Our goal: a compiler approach to optimize simulation codes

- By tuning accuracy
- Through adaptive techniques

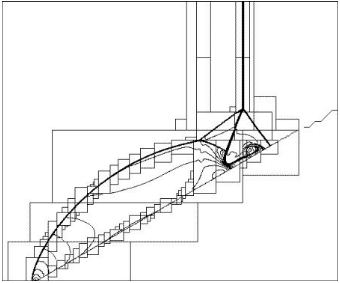
Outline

1. Introduction
2. Numerical Analysis Inspiration: Adaptive Techniques
3. Compiler Technique Inspiration: Polyhedral Frameworks
4. Adaptive Code Refinement
5. Case Study: Eulerian Fluid Simulation
6. Conclusion and Ongoing Work

Adaptive Mesh Refinement

Change the accuracy of a solution in certain regions, while the solution is being calculated [Berger & Colella 89]

- Compute a hierarchical grid which specifies the complexity of the computation
- Refine the precision of the calculation in interesting regions
- Perform basic calculations in regions where almost nothing or nothing happens



Grid Structure for a shock impacting an inclined slope [Wikipedia]

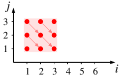
Polyhedral Compilation

The polyhedral model emulates the behavior of some program parts through ($\mathbb{Z}$)-polyhedra [Feautrier 92, Kelly & Pugh 93]

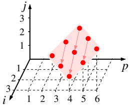
- A $\mathbb{Z}$-polyhedron is the set of feasible integer solutions to a system of linear inequalities, $P = \{\vec{x} \in \mathbb{Z}^n | A\vec{x} + \vec{b} \geq \vec{0}\}$
- Can represent iteration domains, data accesses and statement instance ordering
- May represent an adaptative grid as well !

```
for (i = 0; i < 3; i++)
  for (j = 0; j < 3; j++)
    z[i+j] += x[i] * y[j];
```

⇓ Raising



⇓ Transformation



⇓ Code generation

```
#pragma omp parallel for private(p, i)
for (p = 0; p < 5; p++) {
  for (i = max(0,p-2); i <= min(2,p); i++) {
    z[p] += x[i] * y[p-i];
```

Adaptive Code Refinement (ACR)

- New semi-automatic compiler technique to provide adaptive capabilities to simulation codes
  - Exploits domain-specific knowledge
  - Adapts computation dynamically
  - Regenerates, compiles and links code at runtime
- Composition of three components:
  1. User pragmas to specify domain-specific information
  2. Runtime to adapt the application to the simulation state
  3. Polyhedral code generation to build adapted codes

ACR User Pragmas

Specific set of language extensions to specify:

- Code regions with relaxed semantics preservation
  - Any loop surrounded with ACR pragmas
- Granularity of the adaptation grid
  - Decisions at the grid cell level

```
#pragma ACR grid
```

- Dynamic data to monitor
  - Input data for the decision mechanism

```
#pragma ACR monitor
```

- Alternative computations
  - Versions of the core computation

```
#pragma ACR alternative
```

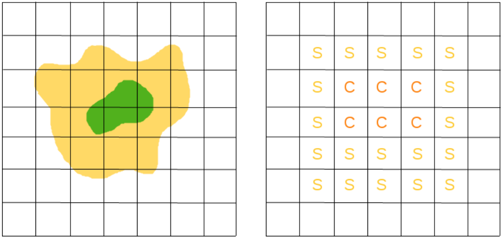
- Adaptation criteria
  - Alternative computation w.r.t. monitored data

```
#pragma ACR strategy
```

ACR Monitoring Runtime

Dedicated thread(s) to constantly monitor dynamic data

- Gather information about the simulation state at a grid-level
- Estimate the computation needed in every grid cell
- Generate a specification of a new version of the code
- Update the specification when the grid changes

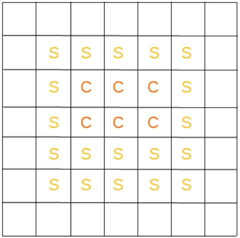


C: complex, S: simple, blank: ∼nothing

ACR Code Generation Runtime

Dedicated thread(s) to constantly generate adapted codes

- Process the specification of code versions
  - Union of polyhedra for each alternative computation region
- Generate a code using polyhedral techniques
  - No internal guards to select the alternative computation
  - Processing of the particles in the original order

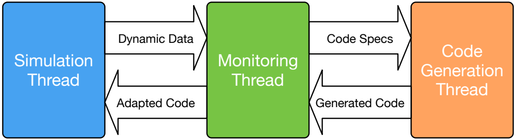


$\Rightarrow$

```
/* Code implementing the grid specification
 * - no internal switch (efficiency)
 * - preserves the original order (precision)
 */
```

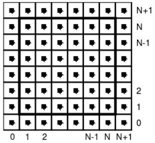
ACR General View

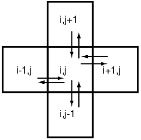
- Simulation thread
  - Switch to specialized code between frames if available
- Monitoring thread
  - Generate code specification w.r.t. the current state
  - Request code generation and compilation
  - Check generated code consistency w.r.t the current state
- Code generation thread
  - Generate and compile an adapted code
  - Signals the monitoring thread about code availability

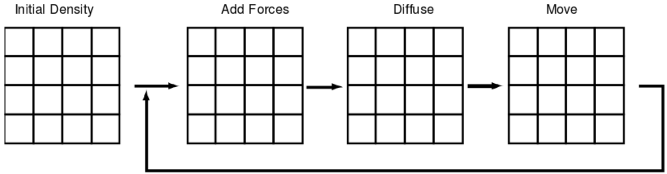


Case Study: Eulerian Fluid Simulation

- Simulates the behavior of a fluid over time
- The space is divided into cells, and every cell represent a particle
- At every time iteration, every cell actualizes its density value and its velocity vector (stencil)





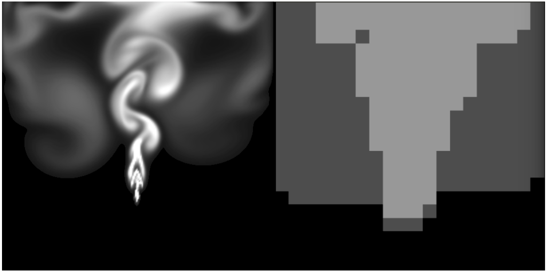


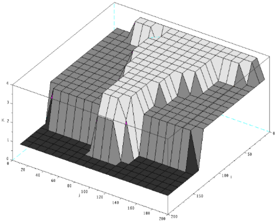
EFS: Domain Specific Information

- Precision depends on the density of the fluid
- Iterative-based implementation: adapt the number of iterations w.r.t. the density of the fluid

```
// Loop iterating over frames of the simulation
while(true) {
  ...
  // lin_solve kernel
  #pragma ACR grid(10)
  #pragma ACR monitor(density[i][j], max, filter)
  #pragma ACR alternative low(parameter,    MAX = 1)
  #pragma ACR alternative medium(parameter, MAX = 3)
  #pragma ACR alternative high(parameter,   MAX = 4)
  #pragma ACR strategy direct(1, low)
  #pragma ACR strategy direct(2, medium)
  #pragma ACR strategy direct(3, high)
  for (k = 0; k < MAX; k++) {
    for (i = 1; i <= N; i++) {
      for (j = 1; j <= N; j++) {
        lin_solve_computation(k, i, j);
      }
    }
  }
  ...
```

EFS: Adaptive Code Refinement



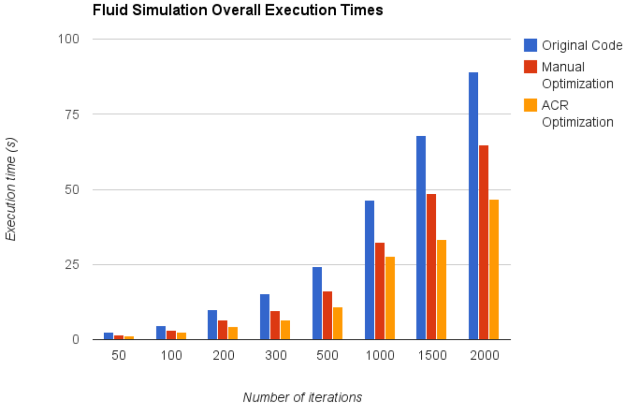


View for a given frame of:

- Simulation visualization
- Grid monitored w.r.t. fluid density
- Iteration domain of the generated code

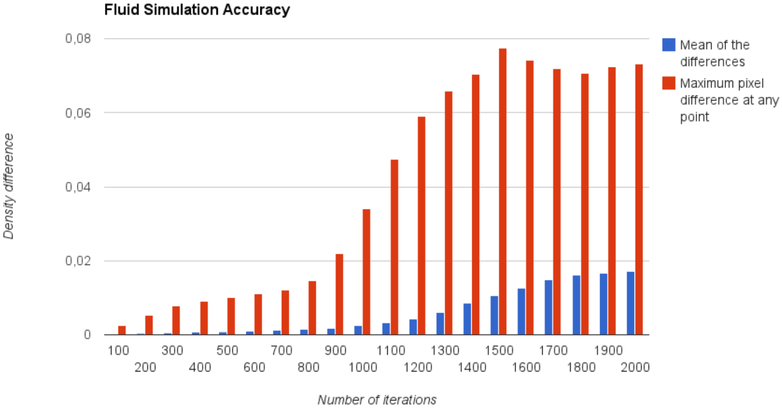
EFS: Early Results (Time)

- ACR removed 80% of the computation
- Speedup $\sim$2 (overhead is still high)



EFS: Early Results (Accuracy)

- Mean deviation below 2% after 2000 frames
- Dependent on user-defined strategy



Conclusion

- New compiler technique to generate adaptive codes
  - Exploits user-provided domain-specific information
  - Relies on state-of-the-art polyhedral code generation
  - Uses multicore architectures for asymmetric processing
  - Easy to use/tune for the user (nice for the compiler as well)
  - Encouraging early results
- Early days of aggressive non-semantics preserving compilation
  - Many improvements are still possible for ACR
  - Push automation forward, reduce overhead
  - Look for other inspiring techniques from numerical analysis