Graphite Front-End

AMD - Austin, Texas

November 16–17, 2008

Front-end, static analysis and conversion to polyhedral representation

- SCoP formation vs. code generation
- Data reference analysis
  - Instance-wise dependences and reaching definitions
  - Fuzzy analysis
  - Interprocedural array region analysis (w/ or w/o LTO)
- Data dependence computation: Integration with PIPlib and Omega Test
- Polyhedral representations
  - Better support for code generation and analysis (sorting, simplification in context, integers)
  - Complexity improvements, caching
- Interaction with the array middle-end
- Support for reductions
- Direct support for irregular control flow

SCoP formation

The definition of a SCoP depends on the imperative language to be compiled:

- in Fortran: slice the program into static control parts, because Fortran can represent computations that are not regular
- in PCP: the whole program represents a regular computation with black boxes that could contain irregular computations: hide irregular computations in the black boxes, no need of scop detection

Reaching definitions: data reference analysis

- fuzzy data references: maydefs
- interprocedural analysis: access range propagation
- link time optimization

Data dependence computation

- transform data references into polyhedral representation
- formulate the dependence problem
- solve with: PIPLib, Omega, ...
- integration of solvers in PCP

Polyhedral representations

- better support for code generation
- better support of abstract domains for static analysis
- complexity improvements for polyhedral algorithms: caching

Array middle-end

- keep vector representations as black-boxes inside the PCP, expose only defs and uses: does not match the original intent of array middle-end
- extend PCP to have constructs for vector operations
- or otherwise, scalarize before translating to polyhedral

Support for reductions

- reductions are shrinking the size of the data, usually shrinking the dimension of data
- support for zero dimension arrays: i.e. scalars

Support for irregular control flow

- when there is a condition that cannot be determined at compile time
- transform irregular control flow into data flow
- encapsulate the unknown predicate in a black box
- PCP approach: handle the irregular block as a black box

Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.