Cetus – An Extensible Compiler Infrastructure

Sang Ik Lee
Troy Johnson
Rudolf Eigenmann
ECE, Purdue University
Introduction

- Source-to-source compiler infrastructure
- Goals
  - Extensible internal representation (IR)
  - Easy to Use IR-API
  - Debugging aid
  - Multi-platform support
Design Rationales

- Why write something new?
- Polaris
  - Targets only Fortran 77
- Suif
  - No major release since 2001
  - Only C is supported
- GCC
  - Not designed for source to source passes
Cetus Implementation

- Implemented in Java
- C parser written using “Antlr” parser generator
- Uses external preprocessor
  - Works with several preprocessors including GNU “cpp”
void function()
{
    int b;
    {
        const int *a;
        ....
        if(b > 0){
            printf("b is positive");
        }
    }
}
Internal Representation cont.

- Statement IR
  - Expressions: Actual operations
  - Statements: Flow of expressions, statements
  - Hierarchical: Some statements have inner blocks of statements

- Symbol Table
  - Distributed: Each scope has a separate table
  - Scope: Block with local symbols
Ongoing Work

- Extending Cetus to handle C++
  - Parser is hard to write
    - Needs context sensitive information to guide parsing
    - Also true for C but could be easily resolved
  - Solution
    - Use generalized LR parser
    - Resolve ambiguity after parsing and during IR construction
Cetus Features

- Debugging support
  - Java
    - Assertions, exceptions, runtime stack trace + source information
  - Cetus
    - API has internal assertions
- Pretty printing
  - Keep close to source
Cetus Features cont.

- Expression simplifier
  - Expression substitution can create very long expressions
  - Change expression into a canonical form
  - Canonical form enables some compiler passes

- Parallel parsing
  - Parses one file per thread
  - Speed up of 2.9 using 4 threads for 176gcc on 4 processor UltraSparc 480Mhz system
IR Construction Time

176.gcc on AMD includes non standard C constructs from header file after preprocessing

SUN: 480Mhz UltraSparc+Solaris
AMD: 1.6Ghz AMD AthlonXP+Linux

164.gzip : 34000 lines
175.vpr : 52000 lines
176.gcc : 571000 lines
188.ammp : 47000 lines
300.twolf : 106000 lines
Cetus Memory Usage

- 5 benchmarks from Spec CPU2000 representing different source size
- Complete IR kept in memory for inter-procedural analysis
- Measured on SUN Platform
Polaris Memory Usage

- 5 benchmarks from Spec CPU2000 representing different source size
- Complete IR kept in memory for inter-procedural analysis
- Measured on SUN Platform
- 301.apsi : 7500 lines
Example Cetus Pass: OpenMP Translator

- Source-to-source translation
- Work sharing construct
  - Micro-tasking helper functions
- Data clause
  - Modify symbol table entries
  - Modify expressions
/*
* Get name of private variables from priv_list
* add a new Symbol for each private variable and
* replace all references to older Symbol
* “stmt” is a CompoundStatement
*/
for (i = 0; i < priv_list.size(); i++) {
    name = (String) priv_list.get(i);  // get the variable name
    orig_var = stmt.get(name);        // get older(existing) Symbol
    // add a new Symbol for Private variable
    priv_var = stmt.add(original_var.getTypeList(), name);
    // replace all reference to Symbol orig_var with reference to Symbol priv_var
    stmt.replace(new IdExpression(orig_var), new IdExpression(priv_var));
}
Conclusions

- Source-to-source compiler infrastructure
- C support
  - Fully implemented
  - Works with SPEC CPU2000
- C++ support
  - GLR parser implemented