MAD2013

Multicore debugging from a SW compiler perspective

Marco Roodzant, marco@ace.nl
ACE Associated Compiler Experts
Amsterdam, The Netherlands

Products and services

for professional compiler developers

www.ace.nl

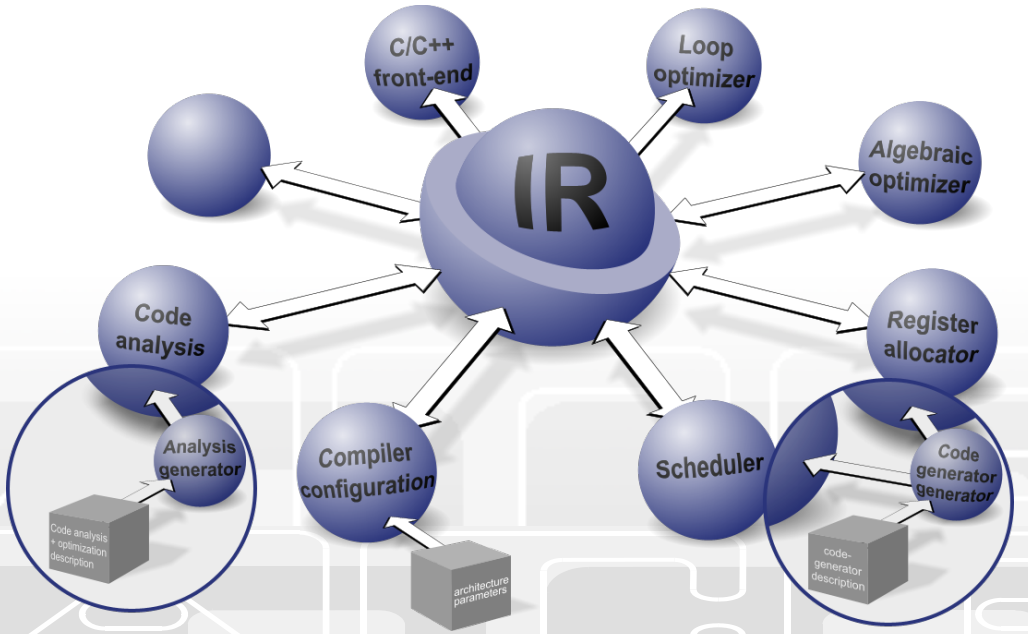
ACE

Presentation Overview

- ACE and CoSy
- ACE solutions in multicore
- CoSy compiler development system and debugging
- Compilers and Multicore Debugging, Problem or Solution
- Food for thought

ACE Associated Compiler Experts

- Home of the CoSy compiler development system
  - Compiler Generator System
  - Modular design
  - Configurable
  - Extensible
  - Retargetable
  - Robust
  - High Quality
  - Highly optimising



- Used by Industry and Academia for advanced compiler construction

ACE solutions in multicore

- Homogeneous Multicore Unix product
- Heterogeneous Multicore Unix product for SuperGraphics system
  - SuperCoff object file format, encapsulating binaries and debugging info
  - Data alignment transparancy for user
- CoSy compiler development system (ACE/CWI/INRIA/GMD/…)
  - Designed for parallel architectures
  - First High Performance Fortran compiler for massively parallel (10000+ processors)
- Various parallel programming models for C, some examples:
  - ParC programming model for Transputers
  - Cn programming model (Derived from C* of Thinking Machines)
  - Streaming, including OpenMP and KPN styles
- Industry and Academia support broad range of heterogeneous multicore
  - E.g. RISC* + FPGA or RISC* + DSP* + ACC*
  - For e.g. LTE, Audio, Video domains

CoSy and debugging

- Debugging Formats
  - STABS, COFF, ELF/DWARF, many proprietary
  - DSL support as many compilers have private extensions
  - General interfaces to support broad range of debugging tools
- Public Domain/Proprietary tools integration
  - Debuggers like GDB
  - Visualizers like aiSee
  - Customer multicore debugger integration for heterogeneous multicore
- Compiler supports additional application debugging features
  - Instrumentation of code
  - Compiler can generate glues and information for some of the higher abstraction levels as presented in first day of MAD2013

Compilers and Multicore Debugging
The *Problem* or Solution

- Compilers can optimize drastically, making debugging hard, loosing link between binary and source
  - Removing and reordering code, Overlays
  - Security of code
- For Multicore, this gets more challenging:
  - Using different compilers for heterogeneous processors
  - Datatypes and alignment ordering
  - Automatic parallelization of code over processors

Compilers and Multicore Debugging
The Problem or *Solution*

- For many multicore debugging aspects, compilers can be of great support:
  - Static and Runtime analysis support
  - Instrumentation of code for all kinds of debugging
  - Control and dataflow information
  - Test vector generation
  - Coverage testing
  - Automatic interface glue generation, for e.g. KPN
  - Adding external information into compilation process to even improve above
  - And of course (semi) automatically mapping applications on heterogeneous parallel systems

Food for Thought

- *Nobrainer1* Proper Debugging Format standardization crucial for true interoperability of tools
- *Nobrainer2* If you can avoid debugging, one should do
- *Itsdoneearlier* Review the Supercomputing/Server area for massively parallel and multicore tools, e.g. TotalView debugger environment. Don't look at 16 processors, but 1000 or more.
- *Economics* Will jointly spending 10 times more money on (debugging) tools save users 100 times that money in finding and debugging errors in a shorter time? If so, what's stopping us?

Food for Thought

- *Compileratrescue* Compilers are the unique interface between Software and Hardware and an underestimated power to assist the application developer with additional analysis and debugging support
- *Outofthebox* Verify the software like (or better with) the chip, or use tools like Verum's ASD:Suite
- *Radical* Throw away legacy and start with a better programming paradigm than C for hw/sw, debug by design
- *Shouldbenobrainer* Compilers should be verified, or at least tested and validated in depth

Thank you

ACE Associated Compiler Experts

Products and services

for professional compiler developers

www.ace.nl