C Language Support in OSCAR
Multigrain Parallelizing Compiler
using CoSy

M. Mase†, K. Kimura‡‡, H. Kasahara‡‡

† Dept. of Computer Science,
‡ Advanced Chip-Multiprocessor Research Institute,
Waseda University, Japan
http://www.oscar.elec.waseda.ac.jp
Research Background

- Multi-Processors and Multi-Cores are emerging everywhere
- Automatic parallelizing compiler becomes more and more important
  - For ease of application development
- OSCAR Multigrain Parallelizing Compiler
  - Originally started from FORTRAN77
  - Achieving outstanding results for numerical applications
  - Strong demands for supporting C language
    - Very popular especially in embedded area
OSCAR Multigrain Parallelizing Compiler

- Generating a parallelized code from a sequential program

- Features
  - Multigrain Parallel Processing
  - Data Localization
  - Data transfer Overlapping
  - Power Reduction

- Compiler cooperative Multi-core architecture
  - OSCAR Multi-core Architecture
  - OSCAR Heterogeneous Multiprocessor Architecture

- Also targeting commercial machines
  - Sun Ultra80, IBM p550Q, SGI Altix 350
  - NEC ARM MPCore, Fujitsu FR1000, Hitachi Renesas SH Multi-core
Multi-grain Parallel Processing

- **Limitation of Loop level Parallelism**
  - Popular parallelizing technique
  - Already reached maturity

- **Exploitation of three kinds of parallelism**
  - Coarse grain task: subroutines, loops, basic blocks
  - Loop level: iterations in a loop
  - Near-fine grain: statements in a basic block

---

The Original Sequential Program

Parallel Processing by an Ordinary Parallelizing Compiler

Doall Loop

Sequential Loop

Par loop iteration parallel processing

Parallelized Sequential Loop

Multigrain Parallel Processing by OSCAR Compiler

---

Transition of a peak performance and an effective performance of HPC

1980 1990 2000

Perfor-mance

Peak performance

The Gap

Effective performance

1980 1990 2000
Coarse grain task Parallel Processing

- A program is decomposed into Macro-Tasks (MTs)
  - Block of Pseudo Assignments (BPA) : Basic Block (BB)
  - Repetition Block (RB) : natural loop
  - Subroutine Block (SB) : subroutine

- Exploitation of parallelism
  - Macro-Flow Graph (MFG) : control-flows and data-dependencies
  - Macro-Task Graph (MTG) : coarse grain task parallelism

Earliest Executable Condition (EEC)

(Condition for determination of MT Execution)

AND

(Condition for Data access)

Ex. Earliest Executable Condition of MT6

- MT2 takes a branch that guarantees MT4 will be executed
- OR
- MT3 completes execution
**Data Localization**

- **Exploitation of Data Locality**
  - for effective use of faster memory (cache or local memory)
- **Loop Aligned Decomposition (LAD)**
  - Target loops are divided into partial loops considering access range and local memory size
- **Consecutive MT scheduling**
  - Assigning MTs in a DLG to the same processor as consecutive as possible
  - Shared data can be passed through processor local memory

Loop Align Decomposition

Loops 2, 3, 7 are divided into 4 smaller loops respectively

(a) Before loop decomposition
(b) after loop decomposition

Oct. 5 - 6, 2006 CoSy Community Gathering
Power Reduction

Fastest Execution Mode

Real-time Execution Mode

Energy Reduction in Real-time Processing

- Deadline = Sequential Processing Time x 1.0

CoSy Community Gathering
OSCAR Compiler’s Components

FrontEnd
parsing a program

MiddlePath
optimization, parallelization

BackEnd
multi-target code generation

OSCAR Compiler’s Components

Fortran77
OpenMP Fortran
C

Fortran Frontend
C Frontend

Intermediate Language

Middle Path
Coarse grain task Parallelization
Loop level Parallelization
Data Localization
Data transfer overlapping
Power Reduction
Static Scheduling
Dynamic Scheduler Generation
Near-fine grain Parallelization

Intermediate Language

OSCAR Backend
SH Backend
UltraSparc Backend
OpenMP Fortran / C Backend
MPI Fortran / C Backend
OSCAR API Fortran / C Backend

OSCAR Machine Code
SH Machine Code
UltraSparc Machine Code
OpenMP Fortran / C Machine Code
MPI Fortran / C Machine Code
OSCAR API Fortran / C Machine Code

Using CoSy

Evaluating on commercial machines

Oct. 5 - 6, 2006
CoSy Community Gathering
Why CoSy?

- For rapid construction of a C compiler
  - Avoidance of composing a C language parser from scratch

  **CoSy**
  - High quality
  - IR (CCMIR) is resemble to OSCAR IR
  - Useful Loop Analyzer
  -Pragma Handling

- CoSy as an Intermediate Representation (IR) converter
  - Development of an “engine” for generating OSCAR Intermediate Representation
int main()
{
    int i, sum=0;
    for (i=0; i<1000; i++)
        sum+=i;
    printf("%d\n", sum);
    return 0;
}

Source C program

- converting symbol tables
- CCMIR to OSCAR IR
- analyzing loop information
- parsing pragma lines
- ...etc.

CoSy OSCAR Frontend and some engines

CoSy Frontend using CoSy

CoSy
Oct. 5 - 6, 2006

OSCAR

CoSy Community Gathering
Loop Analyzer

- **Extraction of canonical shaped loop**
  - Equivalent to DO loops in FORTRAN
    - its iteration number will be determined when the execution of the loop starts
  - One of important factors for parallelization

- **Loop Marker of CCMIR**
  - Extraction of loop structures
  - Analyzing induction variables

- **Loop information**
  - Loop kind
    - while-do, repeat-until
  - Loop variable
    - loop control variables, loop induction variables
  - Important expressions
    - init-expr, test-expr, update-expr

Source C Program
```c
for (i = 0; i < 100; i++) {
    a[i] = b[i] + x;
    c[i] = d[i2] * i;
    i2 += 2;
}
```

Loop Marker
- loop kind : while-do
- control variable : i
- induction variable : i2
- init-expr : i = 0
- test-expr : i < 100
- update-expr : i++

Canonical Shaped Loop
Preliminary Evaluation

Restriction of Source C Program
- Fortran-like C Program (Restricted C)
  - without recursive call
  - without pointer and structure
    - except for Arguments of Functions
  - with some directives
    - some hint information for analyzers not implemented yet

Application
- mp3encode
  - Referencing “UZURA”
    - http://members.at.infoseek.co.jp/kitaurawa/cgi-bin/wiki.cgi
- mpeg2encode
  - Derived from “MediaBench”

On a SMP Workstation
- Sun Ultra80 (4 Ultra SPARC II 450MHz)
  - Native parallelizing compiler : Sun Studio 9 C Compiler
Performance Evaluation Results on 4 processor workstation Sun Ultra80

About 2 times speed up against Sun Studio 9

Oct. 5 - 6, 2006 CoSy Community Gathering
Conclusion

- **OSCAR Multigrain Parallelizing Compiler**
  - Multigrain Parallel Processing
  - Data Localization
  - Data transfer Overlapping
  - Power Reduction

- **C Language Support using CoSy**
  - Converting CCMIR to OSCAR IR

- **Preliminary Evaluation on a SMP workstation**
  - about 2 times speed up against Sun Studio 9

- **Future Works**
  - Performance Evaluations on Multi-core Processors
  - Performance tuning and Relaxing restrictions
Acknowledgements

- A part of this research has been supported by
  - NEDO “Advanced Heterogeneous Multiprocessor”
  - STARC “Automatic Parallelizing Compiler Cooperative Single Chip Multiprocessor”
  - NEDO “Multi core processors for real time consumer electronics”