ICCGI Panel
Challenges in Handling Information Diversity

Moderator: John Terzakis
Intel USA
September 21, 2010
Valencia, Spain
Panelists

• Janet Kourik (Webster University, USA)
• Kyoko Iwasawa (Takushoku University, Japan)
• Antonio Navarro (Universidad Complutense de Madrid, Spain)
• Pierre Robillard (École Polytechnique de Montréal, Canada)
Topics & Discussion

• Janet presented on the volume of information available on the Internet and posed the question of how we determine what to trust.
• Kyoto presented on a compiler design
• Antonio presented on a problem at his university with accessing diverse information spread across three databases.
• Pierre presented the results of a study on how information is communicated (by socialization, by coordination, by cooperation and by collaboration—the highest percentage)
Challenges in Handling Information Diversity

Pierre-N. Robillard
Département de génie informatique et de génie logiciel
École Polytechnique de Montréal
Montréal, Qc. Canada
Surname.Name@polymtl.ca
Ad hoc verbal information
How useful is it?
Interactions Modes for Face-to-Face (FtF) Communications

- **F0F Instant**: 12%
- **F2F Dyad**: 74%
- **FnF Polyadic**: 14%
Information Diversity

Socialization

Coordination

Cooperation

Collaboration
Challenging Questions

• Do we need to go global for all type of communications?
• How can we select the appropriate type of communications?
Diversity on compiling
Compiler infrastructure

Kyoko Iwasawa
Takushoku University
Tokyo Japan
Diversity on compiler

Compilers $n \times m$?

- C
- Java
- fortran
- COBOL
- Pascal
- C#
- Byte Code
- x86
- SPARC
- MIPS
- ARM
- RISC
- SIMD
- Multi Core
- embedded machine

$n \quad m$
Diversity of optimization

- Dataflow analysis VS program conversion
  - Loop conversion, code motion etc.
- Data flow of Array elements VS address expression optimization
- Instruction reorder VS register allocation
- Inter-procedural analysis

- Machine independent VS machine dependent
Other problem

• Compiler development is difficult and complex, however it seems that there is a few novel and interesting technique ...(really?)
  – it looks traditional and conservative
• Young people tend to hesitate compiler’s hard work (at least in Japan).
• Always new system needs their own compiler (especially embedded system)
COINS project  
(COmpiler INfraStructure)

- Every features of compiler are modularized (written by Java)
- Restructure each module (for optimization and parallelization)
- Common intermediate representation
  - Two levels
- Parallelizing features
- Retargetable
The COINS System

- C
- Fortran
- HIR
  - High level Intermediate Representation
  - C
  - OpenMP
- LIR
  - Low level Intermediate Representation
  - C
  - ... (Sparc, x86)

Written in Java from scratch
COINS’s features

- HIR (High level Intermediate Representation)
- LIR (Low level Intermediate Representation)
- Parsers (source program --> HIR)
  - C, Fortran, Java (planned)
- Optimizers for HIR/LIR
  - data flow based (HIR/LIR)
  - SSA based (LIR)
- Parallelizers for HIR/LIR
  - HIR --> OpenMP
  - SIMD parallelization (LIR)
- Code generators (LIR --> machine code)
  - retargetable code generator
  - Sparc, Intel x86
A flowchart illustrating the compilation process from high-level languages (C, Fortran, and new languages) to low-level intermediate representations (HIR and LIR), followed by optimization, parallelization, and back-end generation for target architectures (x86, ARM, MicroBlaze, SPARC) and other platforms (MIPS, SH4, PowerPC, Alpha). The process includes stages such as parser, optimizer, and parallelizer, with specific tools and optimizations applied at each level.
Machine Description

Register Definition

(\*reg-l32\* (foreach \@io (i o)
  (foreach \@n (0 1 2 3 4 5) (REG l32 "%@io@n")))
int 32bits (foreach \@n (0 1 2 3 4 5 6 7) (REG l32 "%l@n")))

Instruction Description

(\@op \@code) ((ADD add) (SUB sub)
  (BAND and) (BOR or) (BXOR xor))
(defcode \@code (SET l32 \@op l32 \@reg rc))
(asm `\@code, $1, $2, $0) rc: \@reg or \@const
(cost 1))

(cost of this instruction)

(foreach \@n \@l) ((2 1) (4 2) (8 3) (16 4) (32 5)) ;; mult by shift
(defcode mul-sll@l (SET l32 \@op l32 \@reg (INTCONST l32 \@n)))
(asm `sll ,$1 (con \@il), $0) con \@il = 1, 2, 3, 4, or 5
(cost 1))
Example of Code Generation

(foreach (@n @l) ((2 1) (4 2) (8 3) (16 4) (32 5)))
(defcode mul-sll@l (SET I32 reg (MUL I32 reg (INTCONST I32 @n))))
(asm `(sll ,$1 (con @l) ,$0))
(cost 1)))

LIR: (SET:I32 %I2 (MUL:I32 %I3 (INTCONST:I32 4)))

Sparc: sll %I3,2,%I2

(@n @l) (... (4 2) ...)
## Examples of retargeting

<table>
<thead>
<tr>
<th>Machine</th>
<th>Coded lines</th>
<th>Months</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARC</td>
<td>1952</td>
<td>-</td>
<td>not available</td>
</tr>
<tr>
<td>x86</td>
<td>2533</td>
<td>-</td>
<td>not available</td>
</tr>
<tr>
<td>MIPS</td>
<td>2207</td>
<td>3</td>
<td>nonexperienced student</td>
</tr>
<tr>
<td>SH4</td>
<td>3596</td>
<td>6</td>
<td>nonexperienced student</td>
</tr>
<tr>
<td>ARM</td>
<td>3052</td>
<td>6</td>
<td>nonexperienced</td>
</tr>
<tr>
<td>ARM-Thumb</td>
<td>1980</td>
<td>3</td>
<td>nonexperienced</td>
</tr>
<tr>
<td>MicroBlaze</td>
<td>1383</td>
<td>2</td>
<td>experienced</td>
</tr>
<tr>
<td>Power PC</td>
<td>5018</td>
<td>6</td>
<td>nonexperienced student</td>
</tr>
<tr>
<td>Alpha</td>
<td>1216</td>
<td>2</td>
<td>nonexperienced student</td>
</tr>
</tbody>
</table>
Execution time ratio (SPEC2000, x86)

Execution time ratio compared to COINS -O0 (no optimization option)

On Epson Pro-1000
Pentium 4
1.8 GHz
256 MB
Thank you for your attention
Challenges in Handling Information Diversity

Janet L. Kourik, Ph.D.
Webster University, St. Louis, Missouri
Growth of Data

- Data
  - Data growing rapidly
  - Petabytes and exabytes
- Challenge to find meaning
- What tools do we use?
Unstructured Text

- Difficult to find meaning in text
- Techniques emerged – 15 years
- Storage formats, metadata, complex data types., etc.
Questions

- How determine quality of sources/input?
- What tools or concepts can we bring to the task?
- How can we help people make good judgments about the information?
Information Diversity in the UCM Virtual Campus

Antonio Navarro
Dpto. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid (UCM)
anavarro@fdi.ucm.es
Index

- UCM Virtual Campus
- Information diversity
- Information diversity in the UCM Virtual Campus
- Conclusions
UCM Virtual Campus

• UCM Virtual Campus is a large virtual campus:
  – More than 44,000 students registered (51%)
  – More than 3,500 lecturers registered (56%)

• Three Course Management Systems (CMSs) are available and integrated:
  – WebCT 4.1
  – Moodle 1.9.2
  – Sakai 2.4.0
UCM Virtual Campus

---|---|---|---|---

- Course integration presentation
- Course integration business
- Course integration integration

- CMS$_1$
- CMS$_n$

- CMS$_1$ integration API
- CMS$_n$ integration API

- Data dump application
- Data load application

- University database
- Data warehouse
- Management database
- CMS$_1$ database
- CMS$_n$ database

UCM CV software architecture
Information diversity

- At least, there are two types of information diversity:
  - Structural (abstract syntax)
    - eg: `<!ELEMENT book (title, author, description)>`
      `<!ELEMENT book (title, author)>`
  - Syntactical (concrete syntax)
    - eg: `<!ELEMENT book (title, author)>`
Information diversity in UCM VC

• Structural diversity

Three of the databases in the UCM VC
Conclusions

• Information diversity mean problems
  – Syntactical diversity mean *medium* problems
  – Structural diversity mean *big* problems
• Structural diversity in the UCM VC is one of the biggest problems
• CMS integration is the other big problem
Information Diversity in the UCM Virtual Campus

Antonio Navarro
Dpto. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
anavarro@fdi.ucm.es