J4 : Compilers Challenges for Computing In Memory

Hybrogen : Compilation for computing in memory archittecture
ACACES july 2022 summer school, HiPEAC, Fiuggi, Italy

Henri-Pierre CHARLES

CEA DSCIN department / Grenoble

july 15, 2022

Outline

cea

Introduction
○●○

Needed technologies
○○○○○○○○○○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

HybroGen : Objectives

Application domains

- Stochastic number support
- Packet filtering : Datatype ipv4, ipv6 addresses
- Transprecision algorithms : usage on mathematical iterative methods
- Stencil processing

Already done

- On the fly code generation for heterogeneous architectures
- Transprecision support : on the fly code generation for precision adaptation. Working demonstration on Newton algorithm : Power / RISCV / Kalray
- Support for In Memory Computing (next slides)
- Target processor modeling (QEMU plugin)

Compilation   Execution time

(a) Static

Multiple (10) kernel executions
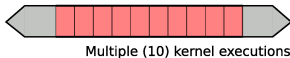
(b)Dynamic   Program init

Kernel init

Application controlled

Introduction
○○●

Needed technologies
○○○○○○○○○○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

HybroGen: General View

Install Time — **Compilation time** source / source (1) — **Compilation time** source / binary (2) — **Execution time**

Introduction
○○○

Needed technologies
●○○○○○○○○○○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○
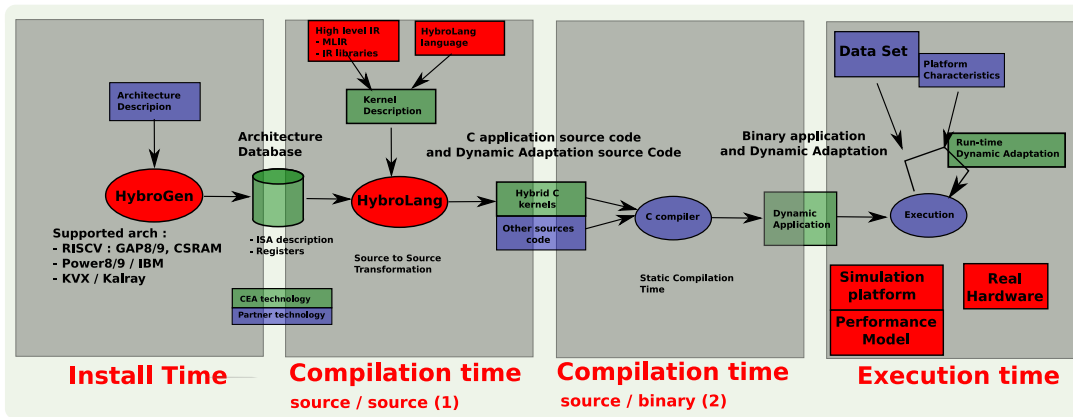
HybroGen : Parsing Options

Where to start

- **Write code before your parser**
- Think about
  - application level acceleration
  - provide instead of search
  - how to learn technology
- Focus on small number of objectivsea

Possible Options

- lex / yacc : Unix historycal tool (LALR)
- flex / bison : GNU version (LALR)
- "Python only" parsers .. ?
- ANTLR
  - LL(k),
  - Multiple output languages
  - Parallel multi language
  - OO : Visitor pattern support
  - Documentation [Ref]ANTLR docs (Books, Doc, API)
  - Examples [Ref]Grammars examples
  - Visual grammar "debugger"
  - BSD licence

Introduction
○○○

Needed technologies
○●○○○○○○○○○○○

Use & Internals
○○○○○○

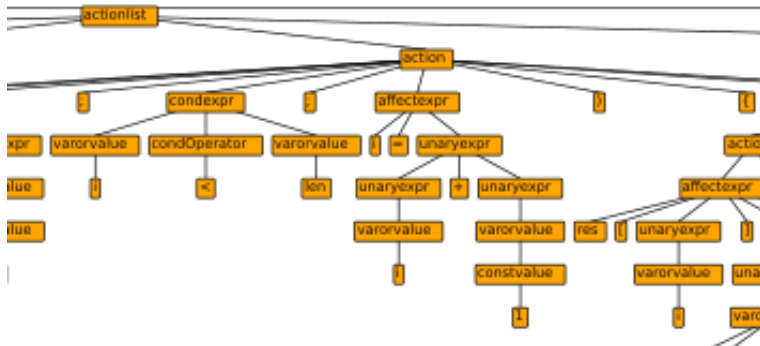Open problems / Roadmap
○○○○

HybroGen : Parsing with Antlr

Development

- Write .g grammar
- Generate your prefered variant (java, Python2/3, C#):
  - Listener
  - Visitor
- Write your Visitor / Listener for each grammar rule
- Get the correct run-time
- Implement actions

Debug

- Use visual debug

Introduction
○○○

Needed technologies
○○●○○○○○○○○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

HybroGen : Parsing H2 Grammar

Lexer Part

```
// Lexer part
condOperator          : ('==' | '!=' | '<'| '>'| '<=' | '>=')        ;
Name                  : [A-Za-z] [A-Za-z0-9_]*                        ;
INLINE                : '#(' ~[)]+ ')'                                ;
// Skipped text
WhiteSpace            : [ \t]+ -> skip              ;
LineComment           : '//' .*? NewLine  -> skip    ;
NewLine               : '\r'? '\n'  -> skip          ;
```

Parser part (total 65 lines))

```
// -*- antlr -*-
grammar HybroLang; // Grammar part
compilationunit : ( function ) +                              ;
function        : fndcl fnbody                                ;
fndcl           : datatype Name fnprototype                   ;
fnprototype     : '(' paramdcllist ')'                        ;
fnbody          : '{' (localvardef)* actionlist returnexpr? '}';
paramdcllist    : vardcl ( ',' vardcl)*                       ;
```

Introduction
○○○

Needed technologies
○○○●○○○○○○○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

HybroGen : Parsing-H2-Visitor

How to implement actions

- Override ANTLR visitor class
- Write action
- Can write multiple visitor on the same grammar in 1 application

Visitor pattern

```
def visitFnbody(self, ctx:HybroLangParser.FnbodyContext):
    self.Trace()
    self.tab += 1
    for localVarDef in ctx.localvardef():
        self.visit(localVarDef)
    IR = self.visit(ctx.actionlist())
    if ctx.returnexpr():
        IR += self.visit(ctx.returnexpr())
    self.tab -= 1
    return IR
```

cea

Introduction
OOO

Needed technologies
OOOOO●OOOOOOO

Use & Internals
OOOOOO

Open problems / Roadmap
OOOO

HybroGen : Database-Options

Storing / retreiving information

- Compilers need PUT/GET for many informations
  - Instructions definitions (parameters, semantic, timing, energy, ...)
  - Program Intermediate Information (IR)
  - System Hardware parameters
- Static compilers compile for only 1 architecture

H2 Idea

- Use relational database for instruction description
  - Arithmetic
  - Vector size
  - Word size
  - Multiple architecture variants
- Supported architectures : RISCV (3 variants), POWER8, Kalray, CsRAM (CEA Computing SRAM)
- Idea : Separate instruction from compiler source

Introduction
○○○

Needed technologies
○○○○○●○○○○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

HybroGen : Database-Postgresql

PostgreSQL

- SQL : data description language, data request/update language
- Force to structure information
- Can separate compilation host & instruction database (not yet used)
- BSD style licence

`https://www.postgresql.org/`
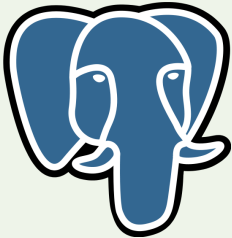
Hybrogen Usage

- Hybrogen compilation process:
    - Make a database proxy
    - Parse the code & retreive instruction list
    - Request needed instructions for current architecture
    - Parse the code / optimize
    - Generate code generator

Introduction
ooo

Needed technologies
oooooooo●oooooo

Use & Internals
oooooo

Open problems / Roadmap
oooo

HybroGen : Database-H2-Schema

SQL simple schema

```
hybrogen=> \d insns
                Table "public.insns"
    Column    |   Type   | Collation | Nullable | Default
--------------+----------+-----------+----------+---------
 name         | text     |           | not null |
 vectorlen    | integer  |           | not null |
 wordlen      | integer  |           | not null |
 extension    | text     |           | not null |
 semname      | text     |           | not null |
 arith        | text     |           | not null |
 parameters   | text     |           |          |
 operand      | text     |           |          |
 archname     | text     |           | not null |
 macroname    | text     |           | not null |
 encoding     | text     |           |          |
Indexes:
    "insns_pkey" PRIMARY KEY, btree (macroname)
Foreign-key constraints:
    "insns_archname_fkey" FOREIGN KEY (archname) REFERENCES insertiond
```

Benefits

- Regular description for the instructions
- Selection made easy for variable size vectors, words (convenient for transprecision)
- Uniform access across architecture
- (Also a description for registers)

Introduction
○○○

Needed technologies
○○○○○○○●○○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

HybroGen : Cross-Compilers

What is this ?

- A classic compiler read a langage A, produce language B, run on language B (C to X86 on X86)
- A cross compiler read a langage A, produce language B, run on language C (C to RISCV on X86)
- Options : https://crosstool-ng.github.io/, https://buildroot.org/ (linux oriented)

We want to generate binary code for embedded system whithout OS, with no compiler bootstrap

Options

- Install packages from your installation : not stable across OS update !
- Think about reproductibility !
- Build your own cross compiler from source
  - There is a lot of tutorials ...
  - Mostly all are false, outdated, OS dependant

Introduction
○○○

Needed technologies
○○○○○○○○○●○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

HybroGen : Cross-Compilers-Intallation

Hybrogen / GenCrossTools

- Script installation procedure
- Fix release dependency
- Build gcc, gdb, qemu, include files coherently
- Same procedure for all architectures (exept Kalray at the moment)

GenCrossTools Options

```
usage : GenCrossTools . py [−h] −a
 { riscv , powerpc , cxram−linux , cxram−bm, kalray }
 [{ riscv , powerpc , cxram−linux , cxram−bm, kalray } . . . ]
 [−p ARCHPREFIX] [−w WORKINGDIR] [−v] [−n]
 [−c] [−t] [−C] [−f ] [−s]

optional arguments :
 −h , −−help              show this help message and exit
 −a { riscv , powerpc , cxram−linux , cxram−bm, kalray } [{ riscv , powerpc , cxram−linux , cxram−bm, kalray }
                          Target architectures
 −p ARCHPREFIX, −−archprefix ARCHPREFIX
                          Installation prefix
 −w WORKINGDIR, −−workingdir WORKINGDIR
                          Working directory
 −v , −−verbose           Verbose
 −n , −−donot             Do nothing , just print actions
 −c , −−clean             Clean build directories
 −t , −−test              Test a cross build installation
 −C , −−cleandist         Clean build directories
 −f , −−config            Show tools configuration
 −s , −−shell             Generate shell configuration
```

Introduction
○○○

Needed technologies
○○○○○○○○○○●○○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

Arch : ⓦQEMU

Sytem level emulation

- Multiple possible usage :
  - System level
  - User level
  - KVM Hosting
  - Xen hosting
- GNU licence

Author : ⓦFabrice Bellard

Working principle

1. Read a block of binary source instruction
2. Transform it in IR
3. Binary compile it in target instruction
4. Execute it
5. Keep it in a buffer for reuse

Introduction
○○○

Needed technologies
○○○○○○○○○○●○

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

Arch : Qemu Plugin

Evolution

- Recent evolution:
  - 0.1 started in 2009
  - 4.2.0 in 2019 : TCG plugin
  - 5.0 2020
  - 7.0 2022

- Call back : register a fonction for an event

Usage https://wiki.qemu.org/Features/TCGPlugins

- https://qemu.readthedocs.io/en/latest/devel/tcg-plugins.html#writing-plugins

- Configure with -enable-plugins

- qemu-risc32 -d plugin -plugin /path/to/plugin <riscv binary>

- or use environment variable QEMU_PLUGIN

Main API calls

```
int  qemu_plugin_install()                    // Register plugin
void qemu_plugin_register_atexit_cb();        // Register for exit
void qemu_plugin_register_vcpu_tb_trans_cb(); // Block to be compiler
void qemu_plugin_register_vcpu_tb_exec_inline() // Counter for an instruction
```

Introduction
○○○

Needed technologies
○○○○○○○○○○○●

Use & Internals
○○○○○○

Open problems / Roadmap
○○○○

HybroGen : Profiling

Strategy

- Profiling on a desktop machine
- With different datasets
- With and without optimization
- Strategy :
  - gcc -O0 -pg -o Kernels Kernels.c
  - ./Kernels dataset
  - gprof ./Kernels

Result

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call   name
100.72    50.57     50.57      102     0.50     0.50
vectorMatrixProduct
  0.00    50.57      0.00        2     0.00     0.00   tick
  0.00    50.57      0.00        2     0.00     0.00   tock
  0.00    50.57      0.00        1     0.00    50.57   Mesure

 %          the percentage of the total running time of the
time        program used by this function.


cumulative  a running sum of the number of seconds accounted
 seconds    for by this function and those listed above it.


 self       the number of seconds accounted for by this
seconds     function alone.  This is the major sort for this
```

Introduction
○○○

Needed technologies
○○○○○○○○○○○○○

Use & Internals
●○○○○○

Open problems / Roadmap
○○○○

HybroGen : Installation

Software installation

`https://github.com/CEA-LIST/HybroGen`

- Checkout : `git clone`
  `git@github.com:CEA-LIST/HybroGen.git`
- Read README.md ! Steps :
  - Install cross compilation tools
  - Install depency tools (antlr, postgresql)
  - Install Hybrogen (make, make install)
  - Run Code Demonstration

Research steps

- Find an intersting application (research money !)
- Profile
- Imagine a new code generation scheme
- Implement
- Publish / Sell

Introduction
○○○

Needed technologies
○○○○○○○○○○○○○

Use & Internals
○●○○○○○

Open problems / Roadmap
○○○○

HybroGen : Programmation

Find a strategy

- Is your program compute bound / memory bound ?
- What parameters could be specialized ?
- What high level optimization is usefull ?
- Do you need to use a special instruction
- Regeneration frequency
  - At each function call ?
  - At each data set change ?
  - At a specific frequency ?
  - At a specific event ?

Use Hybrogen

- Use the legacy HybroGen
- Implement a specific backend optimization
- Add a new architecture

cea

Introduction
ooo

Needed technologies
oooooooooooo

Use & Internals
oooooo

Open problems / Roadmap
oooo

Domain Specific Language : HybroLang

By The Way :

Do we need a new programming language for a new progamming model ?
YES !
[Hennessy-Patterson "A New Golden Age for Computer Architecture"]

Why a new programming language ?

- Want to make data dependent code generation, on the fly.
- Do not want to find vectorization / parallelization : use it !
- Want to **implement** "Inverted Von Neumann Model" (slides on IMC)
- Want to use specific arithmetic : integer, saturated, arithmetic, ... IP@, stochastic, geometric shapes, ...

HybroLang features

- Similar to C syntax
- Variables = Hardware elements (register, memory line)
- Arithmetic Operators = existing processor UAL
- Run-time delayed code generation

cea

Introduction
○○○

Needed technologies
○○○○○○○○○○○○

Use & Internals
○○○●○○

Open problems / Roadmap
○○○○

HybroGen : H2 Language Specific

Language Level

- Link to hardware arithmetics datatype :int, float
- To be done : complex, ip, pixel, char
- Adapt & Link to vector size
- C like syntax
- Leaf function level
- Loop, Arithmetic expression

Run time specific

- Implement innovative code generation scenarios
- Implement innovative code optimization scenarios
- Make dataset responsive applications

Supported architecture

- RISCV : ETHZ riscy, GreenWave / GAP8/9
- CxRAM : CEA / RISCV + CXRAM
- POWER : IBM / Power8
- Kalray : Kalray / KVX

Introduction
○○○

Needed technologies
○○○○○○○○○○○○

Use & Internals
○○○○○●○○

Open problems / Roadmap
○○○○

HybroGen : Optimization Phases

Compilation strategy

- Read code
- Gather needed instructions
- Transform in IR
- Optimize
- Generate C code generators
  - Macro instruction level
  - Instruction generators using macro instruction
  - Compilette rewriting using instruction generators

Optimization levels

- Static :
  - Register allocation
  - Classic code optimization
- Dynamic ($\tilde{4}0$ machine cycle per instructions)
  - Instruction generation
  - Instruciton selection
  - Data interleaving

cea

Introduction
○○○

Needed technologies
○○○○○○○○○○○○○

Use & Internals
○○○○○●

Open problems / Roadmap
○○○○

HybroGen : Optimization-Targe-Specific-Phases

RISC architecture

POWER Specific optimization on compare &
branch optimization

Kalray Specific optimization on arithmetic
operators

CxRAM

- Implement a code generator, generator generator

  - Detect CxRAM instructions
  - Generate RISCV instruciton to generate the
    CxRAM instruction
  - Generate the RISCV generators

Introduction
ooo

Needed technologies
oooooooooooo

Use & Internals
oooooo

Open problems / Roadmap
●ooo

Compiler support for System Architecture

Already investigated compiler support

- Ongoing Kevin Mambu PhD. Hypothesis :
  - 1 IMC tile + 1 IOT node (mono CPU)
  - 1 IMC tile + 1 CPU (with caches)
  - PhD to be presented in oct. 22
- Ongoing work : backend support for TensorFlow

Already investigated HW

- Multi tile IMC reconfigurable + 1 IOT node (mono CPU) Roman Gauchi PhD (done)

To be investigated

Scientific questions : do we need a new programming model ?

- CSRAM in DRAM scenario (PhD Valentin Egloff)
- CSRAM in image capture device (TbD)
- MultiTile CSRAM in MPSoC (TbD)
- Software support for data layout (TbD)
- . . .

cea

Introduction
○○○

Needed technologies
○○○○○○○○○○○○○

Use & Internals
○○○○○○

Open problems / Roadmap
○●○○

Conclusion and Valorisation

Publication / patents

- Papers (SW / HW),
- PhD (1 defended, 3 ongoing)
- Patents : 10+
- CEA "Fait Marquant" : HW / SW
  - 2019 : First In-Memory Computing tile
  - 2021 : Compiler for «computing continuum», application for «In Memory Computing»
  - 2022 : Automatic test code generation from ISA description
  - 2022 : Second circuit tape out June 2022 : RISC-V + IMC
- External requests : Workshops, ANR review, Articles review

Results

- HybroGen Compilation Chain
  - OpenSource CECILL-B licence : https://github.com/CEA-LIST
  - To appear 15 days
- IMC Instruction Set : 4 release
- IMC Chips : 2 chips
- IMC Characterization
- Ongoing installation on juelish cluster

Important message

- Major players are vertical industries
- Application (& software) drive the market

Introduction
ooo

Needed technologies
oooooooooooo

Use & Internals
oooooo

Open problems / Roadmap
oooo

Conclusion : References

Publications

- "Towards a Truly Integrated Vector Processing Unit for Memory-bound Applications Based on a Cost-competitive Computational SRAM Design Solution" ACM Journal on Emerging Technologies in Computing Systems (JETC) Maha Kooli, & all

- "Towards a Truly Integrated Vector Processing Unit for Memory-bound Applications Based on a Cost-competitive Computational SRAM Design Solution" Kooli, Maha & all, ACM Journal on Emerging Technologies in Computing Systems (JETC)

- "A 35.6 TOPS/W/mm^2 3-stage pipelined computational SRAM with adjustable form factor for highly data-centric applications" Noel, J-P & all, IEEE Solid-State Circuits Letters

Patents

- "System comprising a memory capable of implementing calculation operations" Charles, Henri-pierre, Kooli, Maha, Noel, Jean-Philippe, US Patent 10,872,642

- "Memory circuit capable of implementing calculation operations" Charles, Henri-Pierre, Kooli, Maha Noel, Jean-Philippe, US Patent 11,031,076

cea

Conclusion : Embauches au CEA Grenoble

CDD

- Poste de CDD "Innovation en compilation"
  ../.. travailler sur un compilateur qui permet de mettre en oeuvre des solutions innovantes de compilation : compilation dynamique, compilation pour machine non von-neumann, instruction spécialisées

mailto:Henri-Pierre.CHARLES@cea.fr

Thèse

- "Simulation de systèmes numériques contenant un accélérateur quantique"
  ../.. constituer un simulateur de haut niveau, permettant de tester ces différentes hypothèses de micro architecture.

- "Micro-compilation pour réseaux de neurones ternaires sur une architecture de calcul proche mémoire"
  ../.. mettre au point un modèle de programmation innovant spécifique à ce type d'architecture