MAchine Guided Energy Efficient Compilation

Simon Cook & Jeremy Bennett, Embecosm
Do Compilers Affect Energy?

- Initial research in 2012 by Embecosm and Bristol University
- The answer is “yes”
- Now published *open access* in a peer-reviewed journal

*Identifying Compiler Options to Minimize Energy Consumption for Embedded Platforms*
James Pallister; Simon J. Hollis; Jeremy Bennett
Recap: What is MAGEEC?

Today we optimize for speed or space

What if we could optimize for energy usage?
Recap: How We Got Here

Research into feedback directed optimization

Research into modeling energy usage

Energy measurement

Copyright © 2014 Embecosm and University of Bristol
Freely available under a Creative Commons license
Recap: What's New?

Objective is energy optimization

Generic framework: GCC and LLVM initially

Energy measured *not* modeled

Working system, not research prototype
Overall Design

Compiler

Compiler Plugin

MAGEEC

Feature Extractor

Pass Gate

MAGEEC Machine Learner
Feature Extractor Plugin Setup

- Added via `PLUGIN_PASS_MANAGER_SETUP`

```c
void register_featextract (void)
{
    struct register_pass_info pass;
    pass.pass = make_mageec_pass (g);
    pass.reference_pass_name = "ssa";
    pass.ref_pass_instance_number = 1;
    pass.pos_op = PASS_POS_INSERT_AFTER;

    register_callback (mageec_gcc_plugin_name,
                       PLUGIN_PASS_MANAGER_SETUP, NULL,
                       &pass);
}
```

- Pass invoked for each function
  - decisions therefore made per function
  - so we have per-function optimization
static unsigned mageec_featextract_exec (void)

... 

FOR_EACH_BB (bb)
{
    count statements and types 
    count number of successors/predecessors 
    compute other features 
    store all the data
}

construct feature vector for machine learner
Properties of a program.

- currently using the MILEPOST set
- 55 features in total

Examples:

<table>
<thead>
<tr>
<th>ft1</th>
<th>Number of basic blocks in the method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ft2</td>
<td>Number of basic blocks with a single successor</td>
</tr>
<tr>
<td>ft3</td>
<td>Number of basic blocks with two successors</td>
</tr>
<tr>
<td>ft4</td>
<td>Number of basic blocks with more than two successors</td>
</tr>
<tr>
<td>ft5</td>
<td>Number of basic blocks with a single predecessor</td>
</tr>
<tr>
<td>ft6</td>
<td>Number of basic blocks with two predecessors</td>
</tr>
<tr>
<td>ft7</td>
<td>Number of basic blocks with more than two predecessors</td>
</tr>
<tr>
<td>ft8</td>
<td>Number of basic blocks with a single predecessor and a single successor</td>
</tr>
<tr>
<td>ft9</td>
<td>Number of basic blocks with a single predecessor and two successors</td>
</tr>
<tr>
<td>ft10</td>
<td>Number of basic blocks with a two predecessors and one successor</td>
</tr>
</tbody>
</table>
Better Feature Vectors

- Do we have the right set of features?
  - does each feature have an influence?
- We don't know
  - currently using MILEPOST feature set
  - is this optimal?
- We need to use Principal Component Analysis
  - evaluate which features impact on results
  - needs lots of test cases
    - many more than the number of features
  - currently only 93 tests in our test suite
Pass Gate Plugin

- Function added with **PLUGIN_OVERRIDE_GATE**

```c
void mageec_pass_gate (void *gcc_data,
void *user_data __attribute__((unused)))
{
    short *result = (short *) gcc_data;
    mageec::decision d =
    mageec_inst.make_decision (current_pass->name,
    current_function_name());

    switch (d) {
        case mageec::NATIVE_DECISION:
            return;

        case mageec::FORCE_EXECUTE:
            *result = (short) 1;
            break;
        case mageec::FORCE_NOEXECUTE:
            *result = (short) 0;
            break;
    }
}
```

- Make your own mind up
- Run this pass
- Don't run this pass
- Uses decision trees
- Run for each pass of each function
- One tree per pass.
- There is a third option
  - make your own mind up
- Some passes fixed
  - must run
  - must not run
Pass Selection and ICEs

- When turning passes on/off arbitrarily, it results in:

  *** WARNING *** there are active plugins, do not report this as a bug unless you can reproduce it without enabling any plugins.
  
  Event | Plugins
  --------------------------
  PLUGIN_FINISH_UNIT | libmageec_gcc
  PLUGIN_FINISH | libmageec_gcc
  PLUGIN_START_UNIT | libmageec_gcc
  PLUGIN_OVERRIDE_GATE | libmageec_gcc

/tmp/beebs/src/2dfir/fir2dim.c: In function 'main':
/tmp/beebs/src/2dfir/fir2dim.c:63:5: internal compiler error: in loop_preheader_edge, at cfgloop.c:1667

- Often assertion failure in verify_curr_properties.

- Some passes have a fixed requirement to run/not run.
  - **Question**: Is this something we can infer or does this need machine learning?
Initializing the GCC Plugin Interface

- Start MAGEEC and initialize plugins

```c
int plugin_init (struct plugin_name_args *plugin_info,
                 struct plugin_gcc_version *version)
{
    GCC plugin initialization
    /* Initialize MAGEEC Framework, returning error if failed. */
    /* FIXME: Get real compiler target. */
    std::string compiler_version = "GCC-";
    compiler_version += version->basever;
    if (mageec_inst.init (compiler_version, "SOMETARGET"))
        return 1;

    Register plugin callbacks
    return 0;
}
```

- Enhancement Request: API to provide target at runtime

How do we find out the real target name?
The GCC Plugin interface for registering hooks is stable 😊

The underlying GCC API is not 😞

This is not maintainable long-term

Enhancement request: Stabilize the internal API
Here Be Dragons...

- The dream

```
clang -load=mageec helloworld.c
```

**MAGEEC**

- **LLVM**
  - `plugin_init`
  - `MAGEEC analysis pass`
  - `PassManager:: RunOnFunction, PLUGIN_OVERRIDE_GATE`
  - `RegisterPass`
  - `RegisterPassController`
  - `OverrideExecution`

- **LLVM Plugin**
  - `addMAGEECPass`
  - `addMAGEECLearner`
  - `feature_extract`
  - `make_decision`
Here Be Dragons...

- The reality

```
clang helloworld.c
```

```cpp
// Machine guided
class LearnEfficientCompilation {
public:
  static
  void PredictEnergyCost(string);
  bool chooseTerminal(string);
 private:
  MagicRandNumber generator;
};
```

**LLVM**

- `plugin_init`
- **MAGEEC analysis pass**
- `PassManager::RunOnFunction`
  - PLUGIN_OVERRIDE_GATE
- **MageecFeatureExtractor**
  - `(Custom modification for MAGEEC)`
- `MageecInterface`
  - `decideName`

**LLVM Plugin**

Copyright © 2014 Embecosm and University of Bristol
Freely available under a Creative Commons license
Building the Database

- Run many, many times
- How do we choose the tests to run?

DejaGnu

Test Harness

Log file

MAGEEC built with -DMAGEEC_FILEML

$MAGEEC_EXECUTE LIST

Machine Learning once, after all tests run

Copyright © 2014 Embecosm and University of Bristol
Freely available under a Creative Commons license
Full Factorial Design

- From all combinations, we can find the impact of one option.
  - Example with three options, $x_0$, $x_1$, and $x_2$.

$$x_1 = \frac{\sum \circ}{4} - \frac{\sum \bullet}{4}$$
The same data give us the other options as well

\[ x_2 = \frac{\sum \circ}{4} - \frac{\sum \circ}{4} \]

We need a total of 8 runs
- but what if we had 250 options?
From a subset, we can find the impact of one option.
- example with three options, \( x_0, x_1 \) and \( x_2 \).

\[
x_1 = \frac{\sum \text{blue}}{2} - \frac{\sum \text{green}}{2}
\]
Fractional Factorial Design

- The same data give us all the options.
  - by choosing a different combination of data points
    \[ x_0 \, x_1 \, x_2 \]

\[
X_2 = \frac{\sum \text{blue}}{2} - \frac{\sum \text{green}}{2}
\]

- We need a total of 4 runs
  - but it could be \( x_0 \) and \( x_1 \) acting together
More Factors

- Gains are more significant with more factors
  - deal with multiple factor interaction
  - challenge is tools
  - current limit is 120 factors

\[ \chi^2 = \frac{\sum \bullet}{4} - \frac{\sum \bullet}{4} \]
Building the Database

DejaGnu

Test Harness

Machine Learning

$MAGEEC_EXECUTELIST

Log file

FFD Generator

MAGEEC built with -DMAGEEC_FILEML

GCC

Copyright © 2014 Embecosm and University of Bristol
Freely available under a Creative Commons license
A Free and Open Source Energy Measurement System

mageec.org/wiki/Power_Sensing_Board
The **Bristol/Embicosm Embedded Benchmark Suite**

- A free and open source benchmark suite for embedded systems
  - expose different energy consumption characteristics
  - one benchmark can't trigger all optimisations
  - make sure we cover several broad categories:
    - integer, floating point, branch frequency, memory bandwidth

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>B</th>
<th>M</th>
<th>I</th>
<th>FP</th>
<th>T</th>
<th>License</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blowfish</td>
<td>MiBench</td>
<td>L</td>
<td>M</td>
<td>H</td>
<td>L</td>
<td>Multi</td>
<td>GPL</td>
<td>security</td>
</tr>
<tr>
<td>CRC32</td>
<td>MiBench</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>Single</td>
<td>GPL</td>
<td>network, telecomm</td>
</tr>
<tr>
<td>Cubic root solver</td>
<td>MiBench</td>
<td>L</td>
<td>M</td>
<td>H</td>
<td>L</td>
<td>Single</td>
<td>GPL</td>
<td>automotive</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>MiBench</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>Multi</td>
<td>GPL</td>
<td>network</td>
</tr>
<tr>
<td>FDCT</td>
<td>WCET</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>Single</td>
<td>None†</td>
<td>consumer</td>
</tr>
<tr>
<td>Float Matmult</td>
<td>WCET</td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>M</td>
<td>Single</td>
<td>GPL</td>
<td>automotive, consumer</td>
</tr>
<tr>
<td>Integer Matmult</td>
<td>WCET</td>
<td>M</td>
<td>M</td>
<td>H</td>
<td>L</td>
<td>Single</td>
<td>None†</td>
<td>automotive</td>
</tr>
<tr>
<td>Rjindael</td>
<td>MiBench</td>
<td>H</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>Multi</td>
<td>GPL</td>
<td>security</td>
</tr>
<tr>
<td>SHA</td>
<td>MiBench</td>
<td>H</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>Multi</td>
<td>GPL</td>
<td>network, security</td>
</tr>
<tr>
<td>2D FIR</td>
<td>WCET</td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>Single</td>
<td>None†</td>
<td>automotive, consumer</td>
</tr>
</tbody>
</table>
Scheduled for release 31 August 2014

Same underlying principles
- GPL licensed
- no I/O
- avoid library calls

More tests from a wider source
- currently 93
- some data variants of the same test

Greater usability
- C99 standard, passes -Wall -Wextra
- works with 16-bit int
Scalability

- Each AVR test takes 4s
  - 2s to flash device
  - 2s to run test
- Approx 100 BEEBS tests
  - run 6 boards at once
  - 67s per test run
- 200+ optimization passes
  - $2^{200}$ possibilities
  - FFD reduces this
  - to $2^{16} = 65536$ runs
  - = 4,369,067s
  - = 50d 13h 37m 47s
- Oh dear 😞

One compiler on one CPU
Atmel have 200+ AVR variants
More Targets
Placket Burman to the Rescue

- A special case of FFD
- One more run than the number of factors (passes).
- Assumes independence of factors.
- 210 optimization passes means 211 test runs
  - 23h 27m on one board.
- Can then use FFD on most important passes
A Problem

- Key advantage is that we select optimization per function
  - this alone should be a major benefit
- But currently we collect data on complete programs
  - we need data per function for each program
- Demo is based on just 10 BEEBS single function programs
  - with just 700 runs, most factors are not trained
- How do we measure energy per function?
  - feasible for ATMega328PU at 16MHz
  - infeasible for ARM A8 at 500MHz
- Suggestions please
Running MAGEEC GCC

- Standard GCC compilation:
  ```
  avr-gcc -O0 -mmcu=atmega328p 2dfir.c boardsupport.o -o 00.out
  ```

- Compilation with MAGEEC:
  - currently dumps debug info
    ```
    avr-gcc -O2 -mmcu=atmega328p 2dfir.c boardsupport.o -o 02e.out \
    -fplugin=/opt/mageec/lib/libmageec_gcc.so
    ```
  - Options for MAGEEC plugin
    - lists all the passes and whether they are run or not
      ```
      avr-gcc -O2 -mmcu=atmega328p 2dfir.c boardsupport.o -o 02e.out \
      -fplugin=/opt/mageec/lib/libmageec_gcc.so \
      -fplugin-arg-libmageec_gcc-dumppasses
      ```

- Measuring energy usage
  ```
  avrdude -carduino -patmega328p -P/dev/ttyUSB0 \
  -D -U flash:w:02e.out ; energytool -m 1 read EE00 PA0
  ```
// Machine guided
class EnergyEfficientCompilation {
public:
    Machine
~Machine();
    void Train(MageEC::Features, MageEC::EnergyResult);
    void Predict(MageEC::FeaturesSet<>);
    bool chooseColor(std::string);
private:
    MagicWand Magic; // Energy;
};

Demo
$ avr-gcc -O2 -mmcu=atmega328p 2dfir.c boardsupport.o -o 02e.out \
   -fplugin=/opt/mageec/lib/libmageec_gcc.so \n   -fplugin-arg-libmageec_gcc-dumppasses

MAGEEC: Targetting 'GCC-4.9.0' for 'SOMETARGET'

LEARNER: Hello!

MAGEEC: New source file example.c

LEARNER: New file

GCC: Start File
    gcc_data: (nil)
    user_data: (nil)

Pass: '*warn_unused_result', Type: GIMPLE, Function: 'main', Gate: 1
Pass: '*diagnose_omp_blocks', Type: GIMPLE, Function: 'main', Gate: 0
...
Pass: 'ssa', Type: GIMPLE, Function: 'main', Gate: 1
Pass: 'mageec-extractor', Type: GIMPLE, Function: 'main', Gate: 1

Current Function: main
  ( 1) Basic Block Count : 35
  ( 2) BB with 1 successor : 23
  ( 3) BB with 2 successor : 11
  ( 4) BB with > 2 successor : 0
  ( 5) BB with 1 predecessor : 23
  ( 6) BB with 2 predecessor : 11
  ( 7) BB with > 2 predecessor : 0
  ( 8) BB with 1 pred 1 succ : 21
  ( 9) BB with 1 pred 2 succ : 1
...

Enhancement request
Standard GCC pass
Feature extractor
Output from MAGEEC (2)

Pass: 'ubsan', Type: GIMPLE, Function: 'main', Gate: 0
Pass: '*early_warn_uninitialized', Type: GIMPLE, Function: 'main', Gate: 0
Pass: '*rebuild_cgraph_edges', Type: GIMPLE, Function: 'main', Gate: 1
Pass: 'inline_param', Type: GIMPLE, Function: 'main', Gate: 1
Pass: 'einline', Type: GIMPLE, Function: 'main', Gate: 1
Pass: 'early_optimizations', Type: GIMPLE, Function: 'main', Gate: 1

New gate: 0
Pass: 'release_ssa', Type: GIMPLE, Function: 'main', Gate: 1

New gate: 0
Pass: '*rebuild_cgraph_edges', Type: GIMPLE, Function: 'main', Gate: 1

... Pass: 'tailr', Type: GIMPLE, Function: 'main', Gate: 1
Pass: 'ch', Type: GIMPLE, Function: 'main', Gate: 1

New gate: 0
Pass: 'stdarg', Type: GIMPLE, Function: 'main', Gate: 0

New gate: 1
Pass: 'cplxlower', Type: GIMPLE, Function: 'main', Gate: 1

... MAGEEC: End of source file
LEARNER: End file
GCC: End File
gcc_data: (nil)
user_data: (nil)
GCC: Finish
MAGEEC: Finish
LEARNER: Goodbye!
$ avrdude -carduino -patmega328p -P/dev/ttyUSB0 \ 
   -D -U flash:w:O2e.out ; energytool -m 1 read EE00 PA0

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f
avrdude: reading input file "O2e.out"
avrdude: input file O2e.out auto detected as ELF

avrdude: verifying ...  
avrdude: verifying ...  
...  
...  
avrdude: verifying ...  
avrdude: 1760 bytes of flash verified

avrdude: safemode: Fuses OK (H:00, E:00, L:00)

avrdude done. Thank you.

Measurement point 1
Energy: 25.293 mJ
Time: 310.452 ms
Power: 81.471 mW
Average current: 16.370 mA
Average voltage: 4.977 V
### Results for AVR ATMega328PU

<table>
<thead>
<tr>
<th></th>
<th>Standard GCC -O0</th>
<th>MAGEEC GCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy</td>
<td>29.8 mJ</td>
<td>27.6 mJ</td>
</tr>
<tr>
<td>Time</td>
<td>329.1 s</td>
<td>309.8 s</td>
</tr>
<tr>
<td>Power</td>
<td>90.6 mW</td>
<td>89.1 mW</td>
</tr>
<tr>
<td>Average current</td>
<td>17.9 mA</td>
<td>17.6 mA</td>
</tr>
<tr>
<td>Average voltage</td>
<td>5.1 V</td>
<td>5.1 V</td>
</tr>
</tbody>
</table>

- **Based on minimal training:**
  - just 10 single function programs
  - 700 training runs
More Results for AVR ATmega328PU

<table>
<thead>
<tr>
<th></th>
<th>Standard GCC -O1</th>
<th>MAGEEC GCC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Energy</strong></td>
<td>25.2 mJ</td>
<td>27.6 mJ</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>283.6 s</td>
<td>309.8 s</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>88.8 mW</td>
<td>89.1 mW</td>
</tr>
<tr>
<td><strong>Average current</strong></td>
<td>17.5 mA</td>
<td>17.6 mA</td>
</tr>
<tr>
<td><strong>Average voltage</strong></td>
<td>5.1 V</td>
<td>5.1 V</td>
</tr>
</tbody>
</table>

- MAGEEC with such little training cannot touch -O1
# Yet More Results for AVR ATmega328PU

<table>
<thead>
<tr>
<th>Standard GCC -O2</th>
<th>MAGEEC GCC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Energy</strong></td>
<td>25.1 mJ</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>281.3 s</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>89.3 mW</td>
</tr>
<tr>
<td><strong>Average current</strong></td>
<td>17.6 mA</td>
</tr>
<tr>
<td><strong>Average voltage</strong></td>
<td>5.1 V</td>
</tr>
</tbody>
</table>

- Marginal difference compared to -O1
The Future

- Up to project end in November 2014
  - full evaluation on Atmel ATmega328PU and ARM Cortex M3
  - packaging of the system to make it easy to use
    - -Oe option if possible
- Future work
  - currently MAGEEC just gates existing passes
  - better: allow passes to be moved around
  - best: allow MAGEEC to select passes
- Commercialization from December 2014
  - product development of the energy measurement board
  - Embecosm will offer MAGEEC for commercial deployment
Further Reading

- Energy measuring and modeling

- MILEPOST GCC - Feedback directed optimization
  - [ctuning.org/milepost-gcc](http://ctuning.org/milepost-gcc)

- Measurement of compiler energy usage

- BEEBS
  - [www.cs.bris.ac.uk/Research/Micro/beebs.jsp](http://www.cs.bris.ac.uk/Research/Micro/beebs.jsp)

- MAGEEC
  - [mageec.org](http://mageec.org)

- Technology Strategy Board Energy Efficient Computing Program
  - [www.eventbrite.co.uk/e/energy-efficient-computing-consortia-building-and-briefing-london-tickets-12176820197](http://www.eventbrite.co.uk/e/energy-efficient-computing-consortia-building-and-briefing-london-tickets-12176820197)
  - [www.eventbrite.co.uk/e/energy-efficient-computing-consortia-building-and-briefing-dauresbury-tickets-12176826215](http://www.eventbrite.co.uk/e/energy-efficient-computing-consortia-building-and-briefing-dauresbury-tickets-12176826215)
Summary

- MAGEEC has been implemented
  - code is functional
  - more work in progress to package the system for ease of use
- Scalability of training the database is an issue
  - methods to resolve this identified
  - full training will allow us to evaluate the full benefits
- Current measurement system needs refinement
  - need to measure energy usage per function
- All the code is freely available for download
  - some energy measurement boards available on request
Where Can I Get It?

- Project Website: http://mageec.org
- MAGEEC Source: http://github.com/mageec/mageec
- BEEBS: http://github.com/mageec/beebs
- Mailing List: mageec@mageec.org
- IRC: #mageec on Freenode
Thank you

mageec.org
www.embecosm.com
cs.bris.ac.uk