MAchine Guided Energy Efficient Compilation

Jeremy Bennett & Simon Cook, Embecosm

mageec.org
What is MAGEEC?

Today we optimize for speed or space
What is MAGEEC?

Today we optimize for speed or space

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

Research into modeling energy usage
How We Got Here

Research into modeling energy usage  →  Energy measurement
How We Got Here

Research into modeling energy usage

Research into feedback directed optimization

Energy measurement
Research into modeling energy usage

Research into feedback directed optimization

Energy measurement

MILEPOST
How We Got Here

Research into modeling energy usage

Energy measurement

Research into feedback directed optimization
How We Got Here

Research into modeling energy usage

Energy measurement

Research into feedback directed optimization
## Literature Review

- **Energy measuring and modeling**

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

- **Measurement of compiler energy usage**

- **MAGEEC**
  - [mageec.org](http://mageec.org)
What's New?

Objective is energy optimization
What's New?

Objective is energy optimization

Energy measured \textit{not} modeled
Energy Measurement

- Based on ST Discovery board
  - ARM Cortex A8 with ADC daughter board
- Burst sample rate 6Msample/s
  - 192kB on-board RAM buffer
  - Short code samples e.g. superoptimizer
- Sustained sample rate 2Msample/s
  - Pre-processed and Streamed off-board
  - Used for MAGEEC
- The board in action:
  - mageec.org/wiki/Power_Sensing_Board
What's New?

Objective is energy optimization

Energy measured *not* modeled

Generic framework: GCC and LLVM initially
What's New?

Objective is energy optimization

Energy measured not modeled

Generic framework: GCC and LLVM initially

Working system, not research prototype
Implementation
Our Plan

- Implement MILEPOST concepts in a generic way.
- Train and evaluate based on real hardware energy measurements and existing passes.
- Write and evaluate optimization passes specifically for energy efficiency (Jörn Rennecke).
Overall Design

Compiler

Coordinator

Machine Learner

Plugin I/F

ML I/F

MAGEEC
## Overall Design

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Coordinator</th>
<th>Machine Learner</th>
</tr>
</thead>
</table>

* may be gen_features()
Overall Design

Compiler

init

\{ plugin_init() \}

Coordinator

Machine Learner

* may be gen_features()
Overall Design

Compiler

Coordinator

Machine Learner

plugin_init()

init

init()

* may be gen_features()
Overall Design

Compiler

Coordinator

Machine Learner

- plugin_init()
- init()

*may be gen_features()
Overall Design

Compiler

init

{ plugin_init()

  gen_features()

}

Coordinator

init()

Machine Learner

init()

* may be gen_features()
Overall Design

Compiler

- plugin_init()
- init()
- gen_features()

Coordinator

- init()
- version / target
- program features

Machine Learner

- init()
- version / target
- decision()

*may be gen_features()
Overall Design

Compiler

- plugin_init()
- init()
- gen_features()
- run_pass()

Coordinator

- init()
- version / target
- program features
- pass list

Machine Learner

- init()
- decision()
- version / target

*may be gen_features()
Overall Design

Compiler

- plugin_init()
- gen_features()
- run_pass()
- stats_gen()*

Coordinator

- init()
- version / target
- program features
- pass list

Machine Learner

- init()
- decision()
Overall Design

Compiler

init

plugin_init()

gen_features()

run_pass()

stats_gen()

Coordinator

init()

version / target

program features

pass list

next_pass()

mod_stats()

Machine Learner

init()

decision()
Overall Design

Compiler

init

- plugin_init()
- gen_features()
- run_pass()
- stats_gen() *

running loop

Coordinator

- init()
- decision()
- next_pass()
- mod_stats() *

Machine Learner

- init()
- decision()
Overall Design

Compiler
- plugin_init()
- gen_features()
- run_pass()
- stats_gen*
- end()

Coordinator
- init()
- decision()
- next_pass()
- mod_stats*

Machine Learner
- init()
- decision()

* may be gen_features()
**Overall Design**

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Coordinator</th>
<th>Machine Learner</th>
</tr>
</thead>
<tbody>
<tr>
<td>plugin_init()</td>
<td>init()</td>
<td>init()</td>
</tr>
<tr>
<td>init</td>
<td>version / target</td>
<td>version / target</td>
</tr>
<tr>
<td>gen_features()</td>
<td>program features</td>
<td>decision()</td>
</tr>
<tr>
<td>run_pass()</td>
<td>pass list</td>
<td>decision()</td>
</tr>
<tr>
<td>stats_gen()</td>
<td>next_pass()</td>
<td>decision()</td>
</tr>
<tr>
<td><em>may be gen_features()</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>end()</td>
<td></td>
<td>end()</td>
</tr>
</tbody>
</table>

* may be gen_features()
Overall Design

Compiler

init

\{ plugin_init() \\
  gen_features() \\
  run_pass() \\
  stats_gen() \\
  end()
\}

Coordinator

init

\{ version / target \\
  init() \\
  program features \\
  pass list \\
  next_pass() \\
  mod_stats() \\
  new features \\
  decision() \\
  end()
\}

Machine Learner

init

\{ version / target \\
  decision() \\
  decision() \\
  end()
\}

* may be gen_features()
Pass Constraints

- The challenge for an arbitrary order pass manager: only using valid combinations.

- The machine learner can additionally learn about what options to avoid when selecting passes.
Today's optimisation passes optimize for speed... again, why not have passes *dedicated* to energy?

```
gcc -fenergy
```
(or a more descriptive name)
Community Involvement
Thank you

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