BrainStorm

Using Formal Modeling and Machine Learning to Optimize Software Architectures

Andrey Breslav
Mentor: Ethan Jackson
Overview

- Problem Description
- Case Study
- Design Space Description
- Simulation Semantics
- Learning-Directed Exploration
- Evaluation
- Future Work
Problem Description

- Too many choices
- Easy to take a wrong decision early
- Hard to correct later
Using Automated Tools

- Describe Design Space
- The **Tool** Evaluates Different Designs and Picks the **Best** Ones

BrainStorm
Problem Description

Existing Approaches

- Fixed Architecture

  ![Fixed Architecture Diagram]

- Design Space:
  - # of Processors from 1 to 16
  - L1 Cache from 2K to 16K
  - I/D/L2 Block Size from 16 to 32

BrainStorm

- Building Blocks:

  ![Building Blocks Diagram]

- Design Space:
Approach Overview

FORMULA Specification:
+ constraints

Candidate Architectures

Model Finder

Discrete-Event Simulation

New constraints

Machine Learning

BrainStorm
Case Study: Skinput

- 10+ Sensors
- 7 Feature Types
- ≈ 64 KByte/Gesture
- 2-8 Gestures/Sec
- Limited Battery 😊

Diagram:
- Raw Acoustic Data
  - Standard Deviations
  - Classifier
    - FFT
  - Gesture Type
Design Space: Software

Sensors on an arm band

Raw Acoustic Data

Standard Deviations

FFT Classifier

Involved

Gesture Type

Involved

\[ \sum_{i=0}^{n} \frac{x_i}{n} \]

Involved

Computes

\[ \text{Average} \]

OR

No DIV instr.

No DIV instr.

OR

Use DIV instr.

Implements

Average

Computes

Computes

Sum

\[ \sum x_i \]

Primary

\[ x \]

Primary

\[ n \]

Primary

Target value

Value

Primary

Value from sensors

Program

Computation

\[ \text{Primary Value from sensors} \]

\[ \text{Target value} \]

\[ \text{Computation} \]

\[ \text{Program} \]

\[ \text{Average} \]

Candidate Design: Software

Sensors on an armband

Raw Acoustic Data

Standard Deviations

FFT Classifier

Gesture Type

Involved &

Primary

\[ x \]

Involved &

Primary

\[ n \]

\[ \sum_{i=0}^{n} x_i / n \]

\( \text{Sum} \)

Computes

No DIV instr.

Use DIV instr.

Average

Computes

\[ \text{Computation} \]

Primary Value from sensors

Target value

Computation

No DIV instr.

Use DIV instr.

Average

Program

BrainStorm

Capabilities And Costs

A piece of data

Inputs and an output

Implementation: what instructions to run

<table>
<thead>
<tr>
<th>Capability</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>n - 1</td>
</tr>
<tr>
<td>DIV</td>
<td>1</td>
</tr>
<tr>
<td>BASIC</td>
<td>n</td>
</tr>
</tbody>
</table>

Use DIV instr.

OR

Implements

$$\sum_{i=0}^{n} x_i / n$$
Sample Design: Hardware

- Supported bus types
- Supported capabilities (cycles/instr)
- Bandwidth (bit/sec)
- Protocol

Processor Type

Bus Type
Sample Design: Mapping

The processor supports sensors

The processor supports required capabilities

Primary

Value

Program

x

Use DIV instr.

Average

- ADD
- DIV
- BASIC

... SENSORS...

BrainStorm
FORMULA Specifications

partial model Skinput : SimplifiedCommunicationConstraints @ "mapping.4ml"
{
    // Software
    amp is Value("amp"),
    n is Value("n"),
    ampAvg is Value("Average"),

    avgc is Computation(ampAvg, "Sum(|amp|)/n"),
    Involves(avgc, amp),
    Involves(avgc, n),

    avgDiv is Program(avgc, "Use DIV instr."),
    CapabilityRequired(avgDiv, Capability("DIV"), 1),
    // ...

    // Hardware
    BUS is MediumType("BUS", "200 KBit",
    "Platform.Software.PriorityBasedArbitrationDriver"),

    DSP is ProcessingElementType("DSP"),
    CapabilitySupported(DSP, Capability("DIV")),
    ConnectionSupported(DSP, BUS)
}
Approach Overview

FORMULA Specification:
+ constraints

Candidate Architectures

Model Finder

Discrete-Event Simulation

Performance Measurements

Machine Learning

New constraints

BrainStorm
Generating Simulations

User

Partial Model

Candidate Architectures

Program Bodies (C#)

<simulation/>

Simulator

Performance Measurements

BrainStorm
BrainStorm Simulator

Precision of Simulators

- Cycle-Accurate
  - Take real software (binaries) and run it on simulated hardware
  - You have to specify a lot

- Very Abstract
  - Only functional models of software ignoring platform characteristics
  - You don’t learn much about a real system

RTOS Schedulers
Bus Protocols
Drivers
High-Level Algorithms

Simulator architecture:

- Platform
  - HW
  - OS

- Library of models
  - Protocols
  - Drivers
  - Schedulers

- User input
  - Programs

- Architecture to Simulate (XML)
Programs and Scheduling

IEnumerator<IOSRequest> SampleProgramBody(TaskType type, IOperatingSystem os, TaskInput input)
{
    var buffer = input.Memory["INPUT"].(buffer["SESSION"]).buffer["DATA"].
    foreach (var x in data)
    {
        yield return type.IterationStart(0, 4 * Units.Byte);
        var packet = new DataPacket(x);
        packet.SetData("value", x);
        yield return session.Send(packet);
    }
    yield return os.Scheduler.Complete();
}

Scheduler: Determine $\Delta t_1$
Program Starts

Iteration 1
Driver: Send a packet, wait for delivery
Effect: $t_1 \geq t_0 + \Delta t_1$
Scheduler: Run other tasks

Iteration 2
Driver: Next packet
Effect: $t_2$
Scheduler: Run other tasks

Complete

At least $\Delta t_1$
At least $\Delta t_2$
At least $\Delta t_3$

Time

BrainStorm
Execution Time

```
yield return type.IterationStart(0, 4 * Units.Byte);
    var packet = new DataPacket(x);
    packet.SetData("value", x);
    yield return session.Send(packet);
```

<table>
<thead>
<tr>
<th>Amount</th>
<th>Capability</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>BASIC</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>DIV</td>
<td>16</td>
</tr>
<tr>
<td>n - 1</td>
<td>ADD</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ \Delta t(n) = \sum_{Cap} \text{Amount}(cap) \times \text{Cost}(cap) \]

Scheduler: Determine \( \Delta t_1 \)

Program Starts

At least \( \Delta t_1 \)

\( t_0 \)

At least \( \Delta t_1 \)

\( t_1 \geq t_0 + \Delta t_1 \)

At least \( \Delta t_2 \)

\( t_2 \)

At least \( \Delta t_3 \)

\( t_3 \)

Complete

Input Size

BrainStorm

A bus with priority-based arbitration:

Drivers → Simultaneous Packets → Bus is busy → Arbitration → P2 wins

N/bits = size(P2)

Transmission → P2 is delivered → Programs can read P2 → Bus is free

Bus is held on the Bus → Hold Time

Cannot read P2

0 1 +N/Bandwidth +Hold Time

1 time unit
<task type="Dispatch for Join::-> Aggregate feature" declaration-only="True">
  <result-handler>
    <task type="Join::-> Aggregate feature">
      <result-handler>
        <task type="Classify 2::-> Classifier2">
          <result-handler>
            <send medium-type="CAN">
              <packet-metadata name="DEST_ADDRESS" value="Classifier2" />
              <packet-metadata name="PRIORITY" value="1" value-type="int" />
            </send>
          </result-handler>
        </task>
      </result-handler>
    </task>
  </result-handler>
</task>

<send medium-type="CAN">
  <packet-metadata name="DEST_ADDRESS" value="Aggregate feature" />
  <packet-metadata name="PRIORITY" value="1" value-type="int" />
</send>

</result-handler>
</task>
Approach Overview

FORMULA
Specification:
+ constraints

Candidate Architectures

Discrete-Event Simulation

Performance Measurements

New constraints

Machine Learning

BrainStorm
**Learning Overview**

- **Naïve Bayes**

**Design Decisions:**
- sameProcessor(val1, val2)
- processorType(val)
- nextToBusType(val, busType)
- processorUsed(Type)
- busUsed(type)

**Formula**

Fitness() = Latency * Cycles * Packets * Utilization * #Processors * #Media

**Smaller is Better**

- Use a DSP!
- Put A and B together!
- Don’t use I²C!

**Candidate Architectures**

**BrainStorm**

Learning-Directed DSE

- Training
  - 50 Samples
  - Bad Predictions
  - Classifier Constructed
  - Average Fitness Did Not Improve
  - Average Fitness Improved

- Validating
  - 20 Samples
  - Good Predictions
  - Decision Made

- Testing
  - 20 Samples
  - 20 Samples
Learning vs Random

Average Fitness (Normalized)

# Samples Seen

First decision didn’t work
This one worked fine
Six Decisions Are Taken

Smaller is Better

Random
Learning

BrainStorm
Program Dependencies

- `amp`
- `n`
- `Sum(amp^2)`
- `sumSqAmplitudes_i / sumSqAmplitudes_j`
- `sqrt(Sum(Sum(amp)/n - amp_b)^2/n)`
- `Sum(avgAmpRatios)/(n^2(n-1)/2)`
- `COM(Powerifts))`
- `Sum(amp)`
- `classify1` → `classify`
- `classify2` → `classify3`
Summary

Results

- Static Modeling of Embedded Systems (in FORMULA)
  - HW & SW
- High-Level Simulation
  - RTOSes, Protocols, etc
- Learning-Directed DSE
  - Improves over random sampling

Future Work

- More Experiments
  - Different Fitness Functions
  - Different Feature Sets
- Try Other Learning Methods
  - e.g. less naïve
- Extend Simulation Library
  - Device Models
  - Schedulers & Protocols
- Continue Work with Skinput
FORMULA Specifications

Legend:
- Domain
- Depends
- Constraint Set

Capabilities
- Hardware
- Programs
- Values & Computations

Partial Model
- Simple Communication
- Schedulability
- Full Communication

AND/OR

BrainStorm