How to Generate Actionable Advice about Performance Problems

Vincent St-Amour
Northeastern University

April 24th, 2015
1. Carry anvil out onto tightway
2. Drop anvil on road-runner
3. Road-runner-burger.
Non-expert programmers
Compiler optimizations failures + Expensive linguistic features

Non-expert programmers
Compiler optimizations failures + Expensive linguistic features

Non-expert programmers

Optimization coaches

Feature-specific profilers
Performance tools can use information from the compilation and execution processes to provide easy-to-follow recommendations that help programmers improve the performance of their programs with low effort and no low-level knowledge.
Today's menu

Optimization Coaching
- for Racket [OOPSLA 2012]
- for JavaScript [ECOOP 2015]

Feature-Specific Profiling
- for Racket [CC 2015]
What is Optimization Coaching?
It renders text, too!
20 hours later...
(require inline-build-flomap)

(define flomap-transform (lambda [flomap-ccw-rotate]
  (match-define flomap-transform
    (lambda [h fm w c]
      (define new-y (fx->fl (fx->double-flonum (flomap-divide-alpha h fm w c))))
      (lambda [x]
        (let ([x-max (unsafe-flvector-ref z-fm x)]
            [y-min (ceiling (real->double-flonum (flomap-extreme-values w c)))]
            [vs2 (lambda [y] (fx->fl (fx->double-flonum (flomap-lift y y y)))))
            [g1 (lambda [g] (flonum g1 g2))])
        (lambda [x]
          (let ([y (unsafe-flvector-ref z-fm x)]
                [x (fx->fl x)]
                [y (fx->fl y)])
            (flonum (flonum x) (flonum y))
            (g1 (g1 (g1 (g1 (g1 (g1 (g1 (g1 (g1 g1)))))))))))
      (g1 (g1 (g1 (g1 (g1 (g1 (g1 (g1 (g1 g1)))))))))))
  (lambda [g]
    (lambda [x]
      (lambda [y]
        (lambda [z]
          (lambda [w]
            (let ([y (fx->fl (fx->double-flonum (flomap-lift-helper2 y y y)))]
                [x (fx->fl x)]
                [y (fx->fl y)]
                [z (fx->fl z)]
                [w (fx->fl w)])
          (flonum (flonum x) (flonum y))
          (flonum (flonum z) (flonum w)))))))
  (lambda [g]
    (lambda [x]
      (lambda [y]
        (lambda [z]
          (lambda [w]
            (let ([y (fx->fl (fx->double-flonum (flomap-lift-helper2 y y y)))]
                [x (fx->fl x)]
                [y (fx->fl y)]
                [z (fx->fl z)]
                [w (fx->fl w)])
            (flonum (flonum x) (flonum y))
            (flonum (flonum z) (flonum w))))))))

(define deep-flomap-rgb (lambda [f]
  (lambda [x]
    (lambda [y]
      (lambda [z]
        (lambda [w]
          (let ([y (fx->fl (fx->double-flonum (flomap-lift-helper2 y y y)))]
                [x (fx->fl x)]
                [y (fx->fl y)]
                [z (fx->fl z)]
                [w (fx->fl w)])
          (flonum (flonum x) (flonum y))
          (flonum (flonum z) (flonum w))))))))

(define deep-flomap (lambda [f]
  (lambda [x]
    (lambda [y]
      (lambda [z]
        (lambda [w]
          (let ([y (fx->fl (fx->double-flonum (flomap-lift-helper2 y y y)))]
                [x (fx->fl x)]
                [y (fx->fl y)]
                [z (fx->fl z)]
                [w (fx->fl w)])
          (flonum (flonum x) (flonum y))
          (flonum (flonum z) (flonum w))))))))

(define deep-flomap-divide-alpha (lambda [f]
  (lambda [x]
    (lambda [y]
      (lambda [z]
        (lambda [w]
          (let ([y (fx->fl (fx->double-flonum (flomap-lift-helper2 y y y)))]
                [x (fx->fl x)]
                [y (fx->fl y)]
                [z (fx->fl z)]
                [w (fx->fl w)])
          (flonum (flonum x) (flonum y))
          (flonum (flonum z) (flonum w))))))))

(define deep-flomap-lift-helper2 (lambda [f]
  (lambda [x]
    (lambda [y]
      (lambda [z]
        (lambda [w]
          (let ([y (fx->fl (fx->double-flonum (flomap-lift-helper2 y y y)))]
                [x (fx->fl x)]
                [y (fx->fl y)]
                [z (fx->fl z)]
                [w (fx->fl w)])
          (flonum (flonum x) (flonum y))
          (flonum (flonum z) (flonum w))))))))

(define deep-flomap-lift-helper (lambda [f]
  (lambda [x]
    (lambda [y]
      (lambda [z]
        (lambda [w]
          (let ([y (fx->fl (fx->double-flonum (flomap-lift-helper2 y y y)))]
                [x (fx->fl x)]
                [y (fx->fl y)]
                [z (fx->fl z)]
                [w (fx->fl w)])
          (flonum (flonum x) (flonum y))
          (flonum (flonum z) (flonum w))))))))
```racket
(define (define (flomap-lift-helper f) ...) )
```
20:0:

flomap-lift-helper

Missed Inlining (0 success out of 46)
Consider using `define-inline`, `begin-encourage-inline` or turning this function into a macro to force inlining.
Dialog between compilers and programmers

- Successes
  - Float arithmetic specialization.

- Near misses
  - Missed Inlining (0 success out of 46)
    Consider using `define-inline`, `begin-encourage-inline` or turning this function into a macro to force inlining.

- Recommendations
Compilers must be conservative

Recommendations can change semantics!

11:2:
(/ (* max last) IM)

This expression has a Real type. The optimizer could optimize it if it had type Float. To fix, change the highlighted expression(s) to have Float type(s).

(/ 1 0) \rightarrow \text{/: division by zero}

(/ 1.0 0.0) \rightarrow +inf.0
Coaching for Racket
Type-Driven Specialization

#lang typed/racket

(define IM 139968)
(define IA 3877)
(define IC 29573)

(define last 42)
(define min 35.3)
(define max 156.8)

(define (gen-random)
  (set! last (modulo (+ (* last IA) IC) IM))
  (+ (/ (* (- max min) last) IM) min))
Type-Driven Specialization

#lang typed/racket

(define IM 139968)
(define IA 3877)
(define IC 29573)
(define last 42)
(define min 35.3)
(define max 156.8)

(define (gen-random)
  (set! last (modulo (+ (* last IA) IC) IM))
  (+ (/ (* (- max min) last) IM) min))
Type-Driven Specialization

```racket
#lang typed/racket
(define IM 139968)
(define IA 3877)
(define IC 29573)
(define last 42)
(define min 35.3)
(define max 156.8)
(define (gen-random)
  (set! last (modulo (+ (* last IA) 10) IM))
  (+ (/ (* (- max min) last) IM) min))
```

12:5:
```
(/ (* (- max min) last) IM)
```
- This expression has a Real type. The optimizer could optimize it if it had type Float. To fix, change the highlighted expression(s) to have Float type(s).

12:11:

- Float arithmetic specialization.
Coach Architecture

- Compiler Instrumentation
- Optimization Analysis
- Recommendation Generation
- Programmer Response

Performance Information
Compiler Instrumentation

Float - Float

\( (- \text{ max min} ) \)

float-

\( \langle \text{<Float>} \text{<Float>} \rangle \)

(fl- max min)

TR opt: prng-example.rkt 12:11
(- max min)

Float Float

binary float subtraction
Compiler Instrumentation

\[
(\ast (- \text{max} \ \text{min}) \ \text{last})
\]

\[
(\ast \ <\text{Number}> \ <\text{Number}>) \ ; \text{ no change}
\]

\[
(\ast (- \text{max} \ \text{min}) \ \text{last})
\]

TR opt failure: prng-example.rkt 12:8
\[
(\ast (- \text{max} \ \text{min}) \ \text{last})
\]
Float Integer
generic multiplication
## Optimization Analysis

### Optimization-agnostic techniques + Optimization-specific heuristics

#### Pruning
- Incomprehensible failure pruning
- Irrelevant failure pruning
- Optimization proximity
- Harmless failure pruning
- Partial success short-circuiting
- Profiling-based pruning

#### Merging
- Causality merging
- Locality merging
- Temporal merging
- Same-property analysis
- By-solution merging
- By-constructor merging

#### Targeting
- Local reporting
- Non-local reporting
- Solution-site inference

#### Ranking
- Static badness
- Profiling-based badness
12:5:

(/ (* (- max min) last) IM)

This expression has a Real type. The optimizer could optimize it if it had type Float. To fix, change the highlighted expression(s) to have Float type(s).
Programmer Response

\[ \text{(-} \text{fl last)} \text{(-} \text{fl IM)} \]

\[ (+ (/ (* (- max min) last) IM) \text{min}) \]

12:5:

\[ (/ (* (- max min) \text{last}) \text{IM}) \]

This expression has a Real type. The optimizer could optimize it if it had type Float. To fix, change the highlighted expression(s) to have Float type(s).
How well does it work?
Hypothesis: Coaching improves performance

Experiment

• Take existing Racket programs
• Run Optimization Coach
• Follow recommendations
• Measure performance impact (running time)
• Compare with versions hand-optimized by experts

<table>
<thead>
<tr>
<th></th>
<th>50%</th>
<th>1%</th>
<th>0%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expert speedups</td>
<td>Work</td>
<td>Email traffic</td>
</tr>
</tbody>
</table>
Hypothesis: Coaching improves performance

- Baseline: Non-optimized
- Coached: Followed recommendations (Minutes of work)
- Gold standard: Hand-optimized by experts (Days of work)

Execution time, lower is better
Optimization Coach today

• Plugin for the DrRacket IDE

• raco pkg install optimization-coach

• "I have bug reports, therefore I exist."
  – Matthias Felleisen
Coaching for JavaScript
Why JS for coaching?

• Can it work beyond Racket?
• Different compilation model (JIT)
• Different language (OO)

Why coaching for JS?

• Hard to write performant code
• Performance matters
• Non-experts / multi-language programmers
Property Access Optimizations

Dereference static offset
Polymorphic inline cache
Virtual machine call

Failure causes

- Non-uniform initialization
  
  \[
  \text{rocket.height} = 100 \quad \text{vs} \quad \text{rocket.speed} = 20 \\
  \text{rocket.speed} = 20 \quad \text{vs} \quad \text{rocket.height} = 100
  \]

- Inconsistent property location (next slide)
- Polymorphism
- ...

Inconsistent Property Location

// constructor
function IntersectionInfo () {
    this.color = "black"
}

IntersectionInfo.prototype = {
    isHit: false,
    position: null,
    ...
}

... if (D > 0) info.isHit = true; ...

... if (info.isHit) return info.color; ...
// constructor
function IntersectionInfo () {
    this.color = "black"
}

IntersectionInfo.prototype = {
    isHit: false,
    position: null,
    ...
}

... if (D > 0) info.isHit = true; ...

... if (info.isHit) return info.color; ...
Inconsistent Property Location

// constructor
function IntersectionInfo()
    this.color = "black";
}

IntersectionInfo.prototype = {
    isHit: false,
    position: null,
    ...
};

... if (D > 0) info.isHit = true; ...

... if info.isHit return info.color; ...

badness: 2596

affected property: isHit

This operation needs to walk the prototype chain to find the property.

Try putting the property in the same location for all objects.
Inconsistent Property Location

// constructor
function IntersectionInfo () {
    this.color = "black"
    this.isHit = false;
}
IntersectionInfo.prototype = {
    isHit: false,
    position: null,
    ...
}

... if (D > 0) info.isHit = true; ...

... if (info.isHit) return info.color; ...
Inconsistent Property Location

```javascript
constructor
{
  this.color = "black";
  this.isHit = false;
}

IntersectionInfo.prototype = {
  position: null,
  ...
}

if (D > 0) info.isHit = true;
...

... if (info.isHit) return info.color; ...
```
Inconsistent Property Location

```javascript
// constructor
function IntersectionInfo()
{
    this.color = "black";
    this.isHit = false;
}

IntersectionInfo.prototype = {
    position: null,
    ...
}

... if (D > 0) info.isHit = true; ...

... if (info.isHit) return info.color; ...
```

6% speedup

17% after doing other fields
**JIT vs AOT**

- Compile / execute interleaved
- Code compiled multiple times
- Optimizations change over time
- Profiler-driven instrumentation
- Temporal merging
- Profile-based ranking

**OO vs Functional**

- Non-local failures
- Constructor near miss clusters
- Solution-site inference
- By-solution merging
- By-constructor merging
Hypothesis: Coaching improves performance

- Baseline: Non-optimized
- Coached: Followed recommendations (Minutes of work)

Octane score, higher is better
Hypothesis: Coaching improves performance

Baseline: Non-optimized
Coached: Followed recommendations (Minutes of work)
Octane score, higher is better

V8

JavaScriptCore
The SpiderMonkey coach today

- github.com/stamourv/jit-coach/
- Command-line prototype
- Ongoing at Mozilla: Firefox integration
Feature-Specific Profiling
(emit
  (sequence
    sawtooth-wave #:bpm 380 [(C 5) #f (C 5) #f (A# 4) #f (C 5) ...])
  "funky-town.wav")
<table>
<thead>
<tr>
<th>Time %</th>
<th>Name + location</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.7%</td>
<td>math/array/untyped-array-pointwise.rkt:43:39</td>
</tr>
<tr>
<td>27.5%</td>
<td>math/array/typed-array-transform.rkt:207:16</td>
</tr>
<tr>
<td>18.1%</td>
<td>synth.rkt:86:2</td>
</tr>
<tr>
<td>6.5%</td>
<td>math/array/untyped-array-pointwise.rkt:30:35</td>
</tr>
<tr>
<td>6.0%</td>
<td>math/array/typed-utils.rkt:199:2</td>
</tr>
<tr>
<td>4.4%</td>
<td>math/array/typed-array-struct.rkt:117:29</td>
</tr>
</tbody>
</table>

```racket
#lang racket
(require racket)
(require "audio.rkt")
(require "synth.rkt")

; ;; data subchunk
; ;; all of equal weight
(apply-weight (note X note) (lambda (x y) (* x y)))

; ;; all of equal weight
(apply-weight (note X note) (lambda (x y) (* x y)))
```
Contracts account for **73.77%** of running time
(17568 / 23816 ms)

6210 ms : Array-unsafe-proc
  (→ Array (→ (vectorof Int) any))

3110 ms : array-append*
  (→* ((listof Array)) (Int) Array)

2776 ms : unsafe-build-array
  (→ (vectorof Int) [...] Array)

... 

Generic sequences account for **0.04%** of running time
(10 / 23816 ms)

10 ms : wav-encode.rkt:51:16
Reporting costs per feature instance

<linguistic feature> : <total cost>
  <cost> : <instance>
  <cost> : <instance>
...

E.g.

Output  Generic sequences
Casts    Security checks
Marketplace processes  Contracts
Pattern matching  Method dispatch
Keyword arguments  Backtracking

<insert your new feature here>
Reporting costs per feature instance

Pattern Matching : 1000ms
600ms : sequencer.rkt:23
200ms : drum.rkt:52
...

Instance ~ Source location

(define (sawtooth-wave ...)
  ...
  (match signal
    [<pattern>
      ...
      (harmonics ...)]
    ...))
Reporting costs per feature instance

Contracts: 2400ms
1300ms: make-waveform
500ms: generate-chord
...

1 instance: Costs in N locations
Reporting costs per feature instance

Marketplace Processes : 1300ms
800ms : (tcp-serve 53588)
400ms : (tcp-serve 53587)
...

1 location: N instances
How does it work?
Architecture

Sampling thread

Instrumentation inside libraries/DSLs

Protocol

Offline analysis

contracts.rkt

casts.rkt

<your feature here>
Observing Feature Code

Stack

sawtooth-wave
generate-note
sequence
Observing Feature Code

Mark present = Feature code is running
Observing Feature Code

Mark present = Feature code is running
How well does it work?
Performance Impact

Experiment

• Take existing Racket programs
• Run the feature-specific profiler
• Fix uses of features mentioned in the report
• Measure performance impact (running time)
## Instrumentation Effort

### Feature LOC

<table>
<thead>
<tr>
<th>Feature</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contracts</td>
<td>183</td>
</tr>
<tr>
<td>Output</td>
<td>11</td>
</tr>
<tr>
<td>Generic sequences</td>
<td>18</td>
</tr>
<tr>
<td>Casts and assertions</td>
<td>37</td>
</tr>
<tr>
<td>Parser backtracking</td>
<td>18</td>
</tr>
<tr>
<td>Security policies</td>
<td>23</td>
</tr>
<tr>
<td>Marketplace processes</td>
<td>7</td>
</tr>
<tr>
<td>Pattern matching</td>
<td>18</td>
</tr>
<tr>
<td>Method dispatch</td>
<td>12</td>
</tr>
<tr>
<td>Keyword arguments</td>
<td>50</td>
</tr>
</tbody>
</table>

**Reasonable for library creators**

35 minutes for creator! (+ 40 for extra analysis)
Wrapping up
Compiler optimizations failures

Non-expert programmers

Optimization coaches

Feature-specific profilers

Expensive linguistic features
Merrie Melodies

"That's all Folks!"

A WARNER BROS. CARTOON

A VITAPHONE® RELEASE