Simple and Effective Type Check Removal through Lazy Basic Block Versioning

Maxime Chevalier-Boisvert (joint work with Marc Feeley)

arxiv.org/abs/1411.0352

JavaScript

- Primitive operators have complex semantics, contain hidden dynamic type checks
- Difficult to optimize due to semantic complexity, dynamic code loading and eval
- Type analyses are often costly and imprecise
  - Impractical to use in a JIT compiler
- Modern JS JIT compiler architectures are increasingly complex

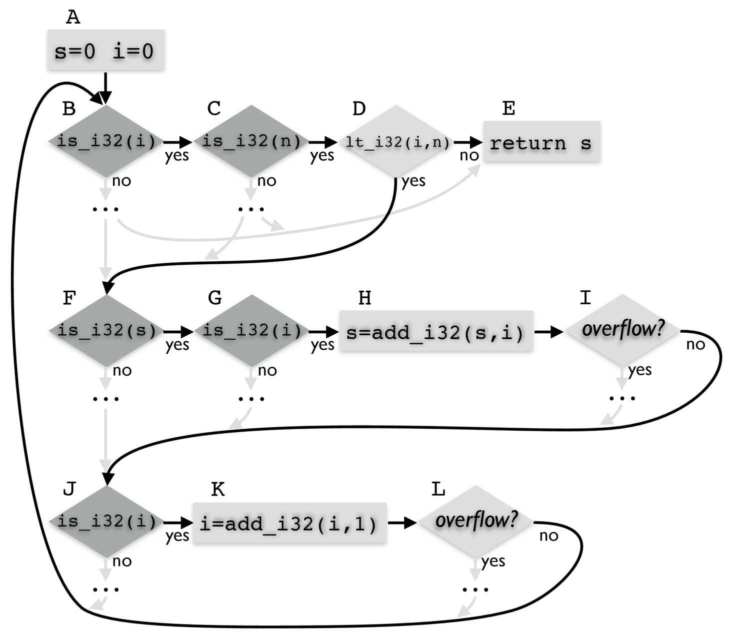
Basic Block Versioning

- As code is compiled, type info is accumulated
  - Type tests add type information
  - Known types are propagated
- Basic blocks are cloned on-the-fly
  - Specialized based on known variable types
  - May compile multiple versions of blocks
- Key advantages are simplicity and speed
  - Type specialization without type analysis
  - No interpreter, no on-stack replacement
  - Code generated in one single pass

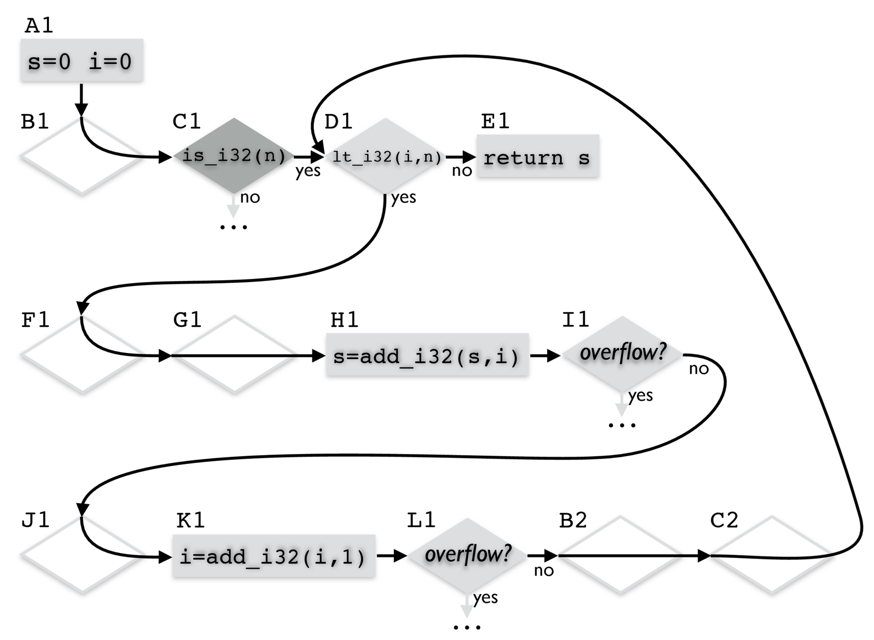
A Simple Example

```
function sum(n)
{
    for (var i = 0, s = 0; i < n; i++)
        s += i;
    return s;
}
```

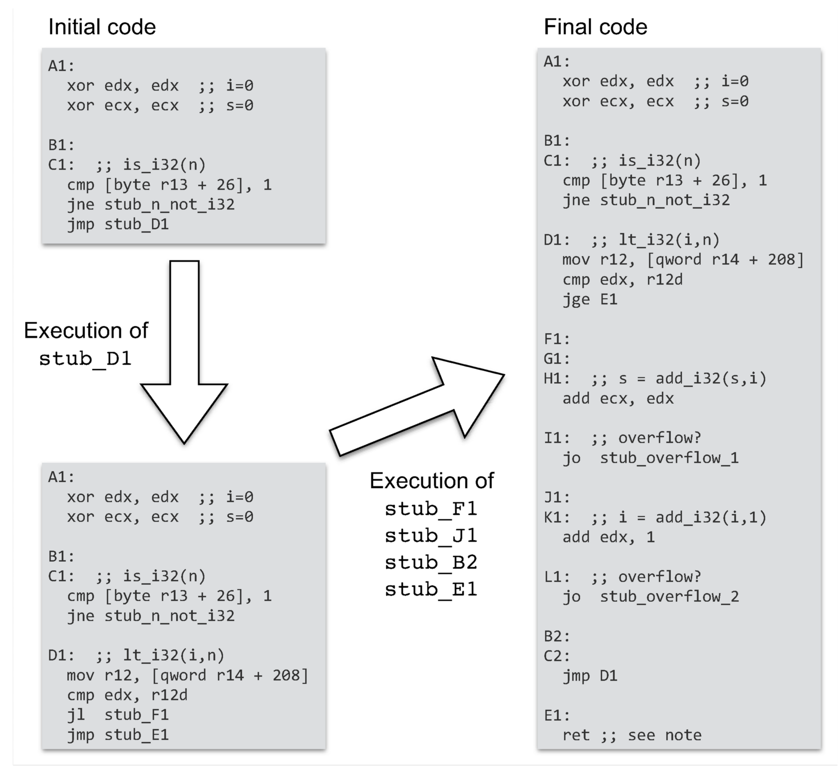
Without BBV



With BBV

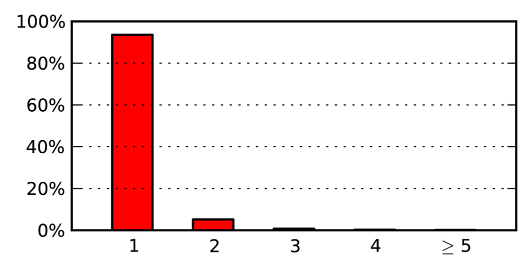


Machine Code

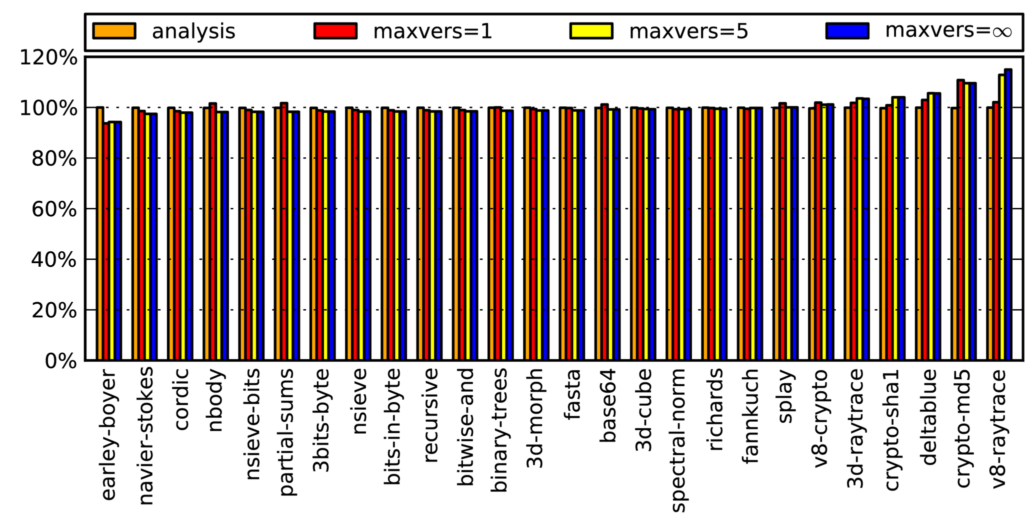


Performance Results

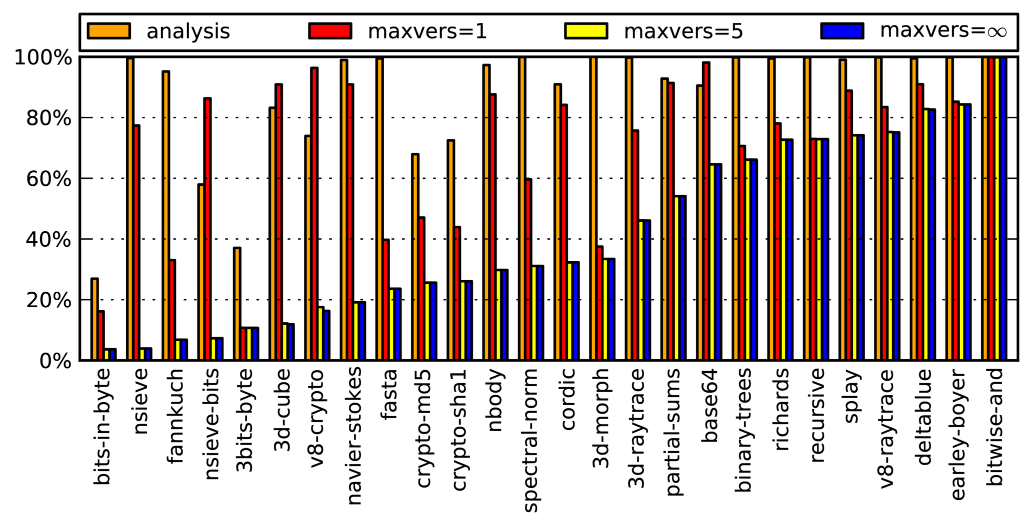
Versions per block (bucket counts)



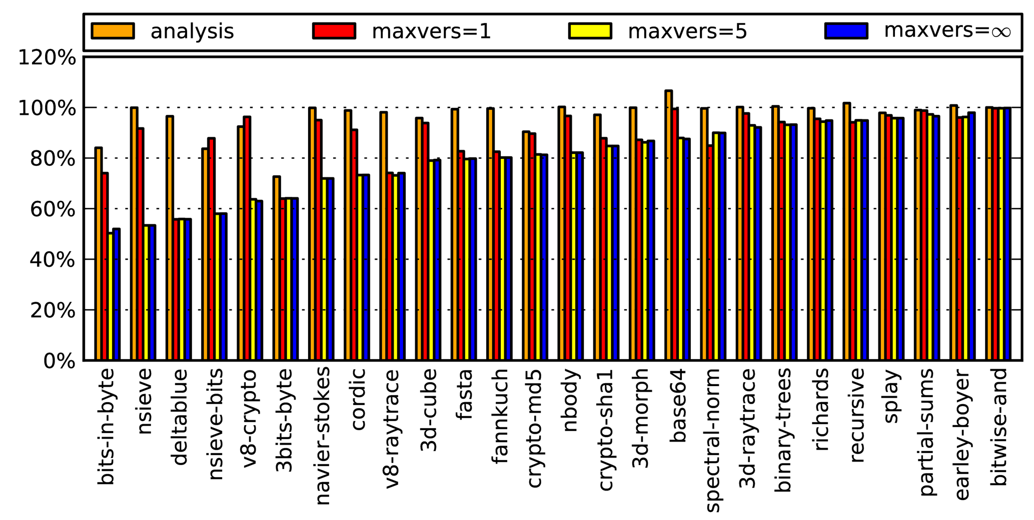
Code size for various block version limits relative to baseline (lower is better)



Dynamic counts of type tests executed relative to baseline (lower is better)



Execution time relative to baseline (lower is better)

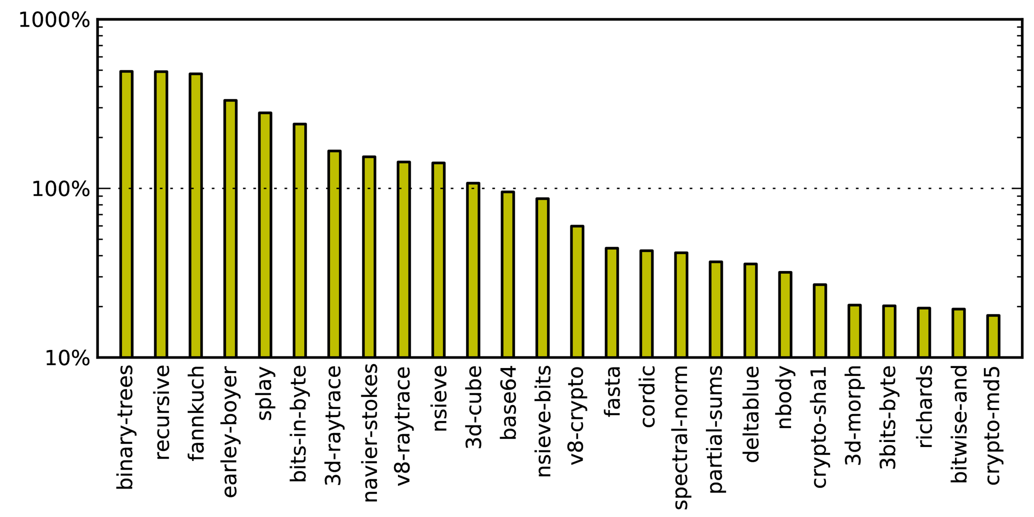


Findings

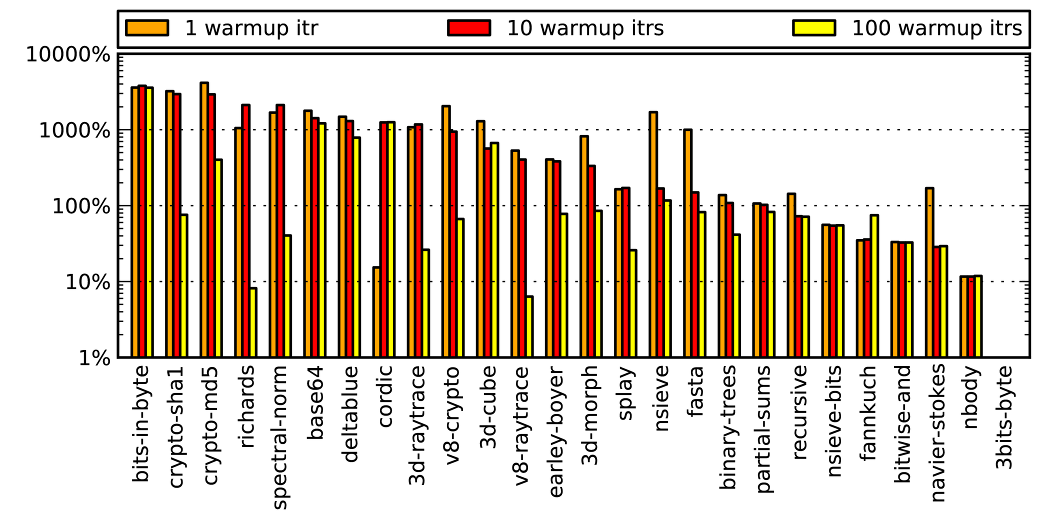
- Small and limited code size growth, no version explosion
- On average, 71% of type tests eliminated vs 16% for traditional type analysis
- Speedups of up to 50% over baseline
- Faster than V8 baseline, TraceMonkey and Truffle/JS on several benchmarks.

Comparative Performance

Speedup relative to TraceMonkey (log scale, higher is better)



Speedup relative to Truffle/JS (log scale, higher is better)



Future Work

- Propagating object types through BBV
  - Current technique ignores object types
  - In JS, global vars are on a global object
- Interprocedural BBV
  - Current technique intraprocedural only
- Code compaction & collection
  - Removing dead machine code
- Fast on-the-fly register allocation

Presenting at ECOOP 2015
Wednesday 14:00

github.com/maximecb/Higgs

maximechevalierb@gmail.com
Love2Code on Twitter

Université de Montréal