Interpreter Optimizations

Ruturaj Kiran Vaidya

Outline

- Background
- Context threading
- Superoperators
- Conclusion

Background

- Decode - Dispatch Interpreter - high interpretation cost
- Indirect Threaded Interpretation - dispatch code to - interpreter routines
- Direct Threaded Interpretation
  - Best choice amongst three

Background

Motivation:

- Direct-threaded - Indirect branches - to dispatch bytecodes
- Deeply-pipelined architecture rely on - branch predictors - for performance
- context problem - indirect branches are poorly predicted

Idea:

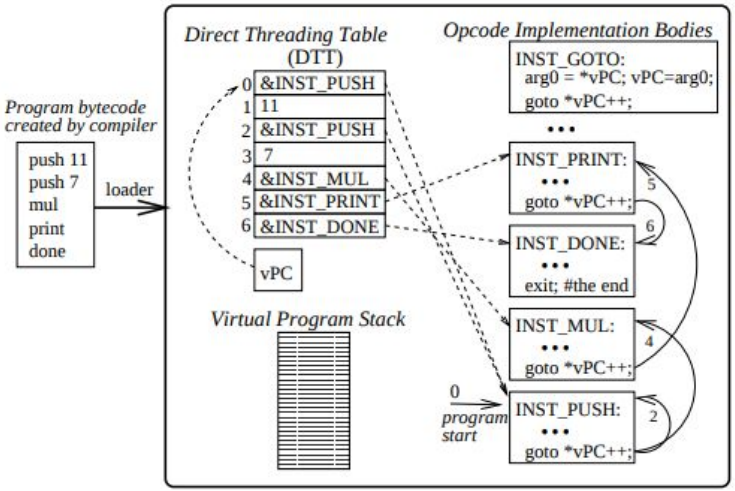
- Refine the dispatch itself - **context threading**
- Efficient or fewer dispatches - **superoperators**

Context Threading

Idea:

- Align hardware and virtual machine state
- i.e. correlate native pc (tpc) to vpc (spc)
- Improve branch prediction

Context Threading - direct threaded interpreter



Direct threaded interpreter

001048d0		
1	2	08

(load word and zero)

00104800		
3	1	03

(add)

00104910		
3	4	00

(store word)

Diagram credits:
http://www.ittc.ku.edu/~kulkarni/teaching/EECS768/slides/chapter2.pdf

Diagram credits: http://www.cs.toronto.edu/~matz/pubs/demkea_context.pdf paper

Context Threading - direct threaded interpreter

But there is a problem with this approach

- Target of the dispatch branch depends on vpc, but not hardware pc
- context problem

Context threading - direct threaded interpreter

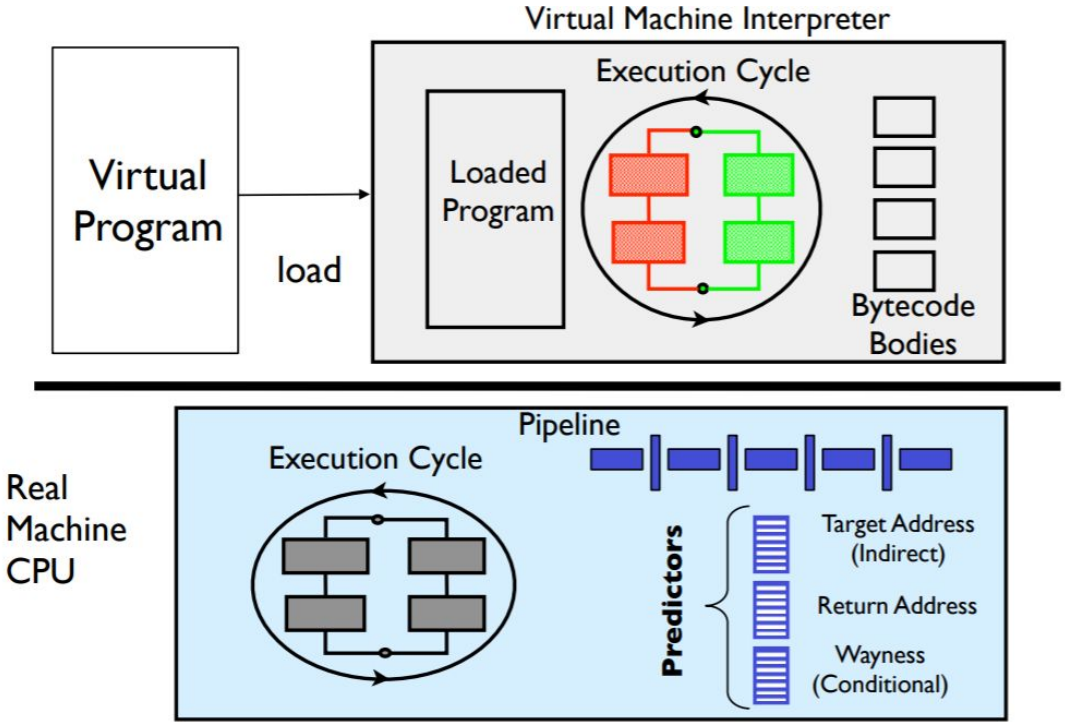


Image credits: http://www.cs.toronto.edu/syslab/talks/cgoAsWorkshop.pdf

Context threading - direct threaded interpreter

But there is a problem with this approach

- Target of the dispatch branch depends on vpc, but not hardware pc
- context problem

Leverage hardware predictors

Context threading

Virtual program may contain the following control flow types:

- Conditional branches
- Call and returns
- Unconditional branches
- Linear code

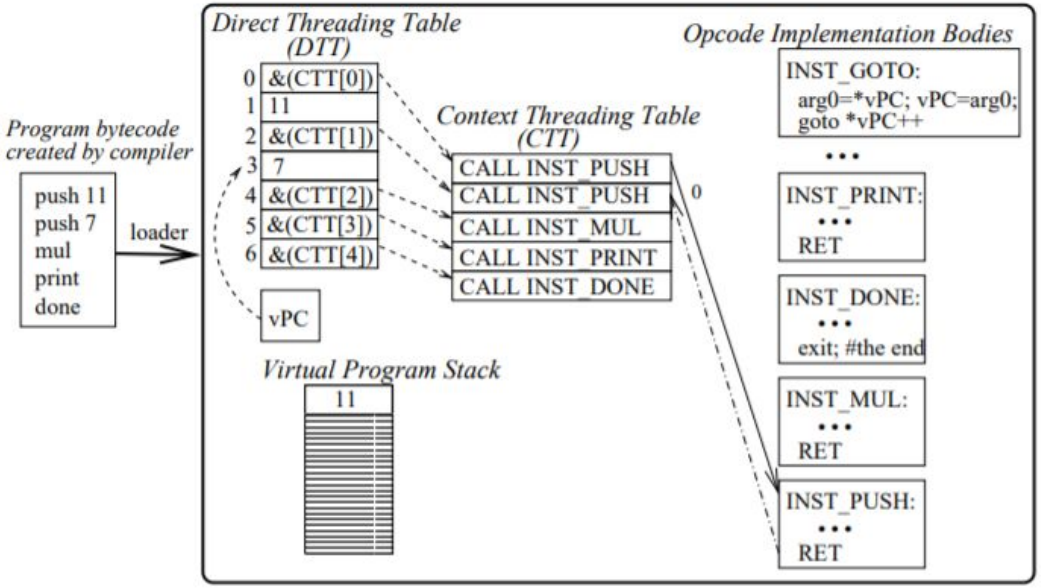
But direct threading uses indirect branches for all types of control flows

Idea: expose the virtual control flow patterns to the hardware, map spc and tpc

Context threading

- Handling linear dispatch
- Handling virtual branches
- handling virtual call and return

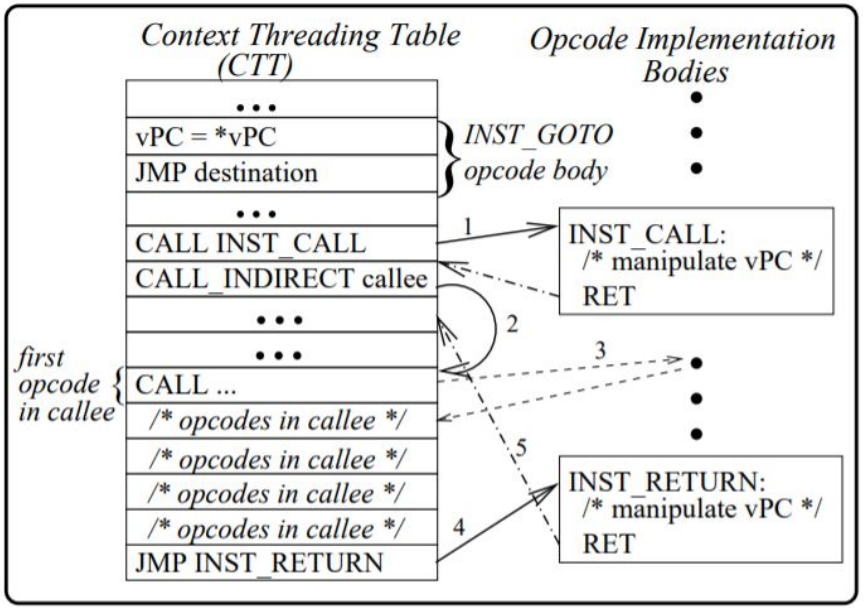
Context threading - Handling the control flow



- Handling linear dispatch
- Uses CTT

Diagram credits: http://www.cs.toronto.edu/~matz/pubs/demkea_context.pdf

Context threading - Handling the control flow



Handling virtual branches

- Inline the bodies into CTT

Handling virtual call and return

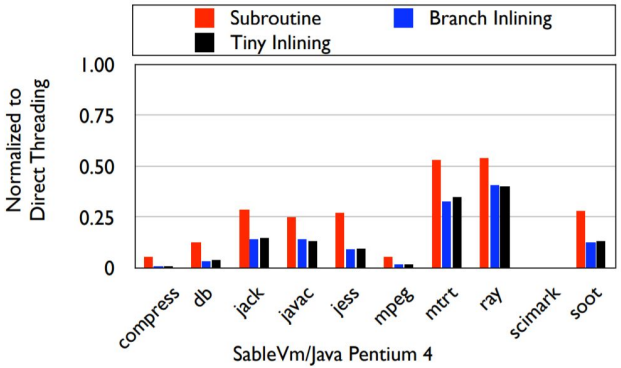
- new handler instructions are added

Diagram credits: http://www.cs.toronto.edu/~matz/pubs/demkea_context.pdf

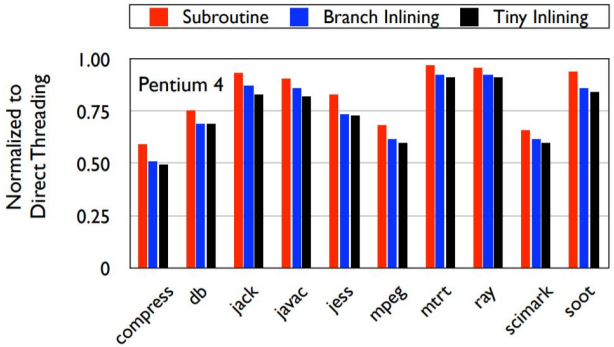
Context threading - Tiny inlining

- Context threading can be combined with inlining strategies
- Inline techniques can be used to reduce the pipeline hazards
- simple heuristic
- Inlining all the small bodies
- This removes dispatch overhead surrounding smallest bodies
- Further optimizing the context threaded interpreter

Context threading - Results



95% branch mispredictions eliminated on average



27% reduction in execution time on average

Results: https://webdocs.cs.ualberta.ca/~amaral/cascon/CDP05/slides/CDP05-berndl.pdf

Superoperators

- Efficient or fewer dispatches - as seen earlier
- An optimization technique for bytecode interpreters
- combines smaller atomic operations
- Decrease the executable size
- Increase speed

Superoperators

- simple e.g.:
- ADD(R, 4)
- Requires several virtual instructions
- Extend the virtual instruction set to add a single “super-instruction”
- Decreases the executable size and makes the interpreter faster
- lcc ir uses 109 operators
- why not use remaining 147 byte codes for superoperators

Superoperators

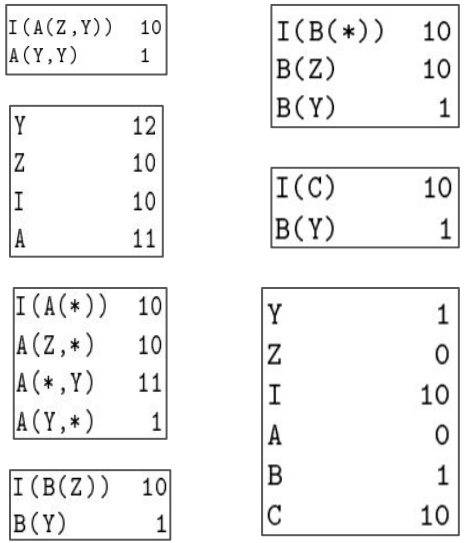
Inference Heuristic:

- Interpreter includes a heuristic method for inferring a good set of superoperators
- Reads IR trees and decides which adjacent operators to merge
- Each tree is weighted to guide the heuristic
- Optimize for space and time

Superoperators

Let’s take an example.

1. Assume inputs trees with weights.
2. Now find the operator frequencies.
3. Now find the frequency of adjacent nodes.
4. Find the node with the highest frequency.
5. Repeat the operation.



Conclusion

- Context problem: Branch mispredictions due to the mismatch between virtual and native control flow
- Context threading method
- Superoperators - optimize the interpreters by reducing branches

Thanks for listening.