# Measuring The Performance of Stack-based and Register-based VMs

A Presentation Based on the paper *A Performance Survey on Stack-based and Register-based Virtual Machines*

Ruijie Fang, Siqi Liu

ruijie.fang@temple.edu, tylerliu2018@lschs.org

December 2016

# Prelude: How the Question is Raised

- The performance war between interpreted languages (e.g. Java, Python, etc.)
- Not only do we want to know WHICH, we want to know WHY
- Interpreter (VM) 's runtime performance dominates the language's performance

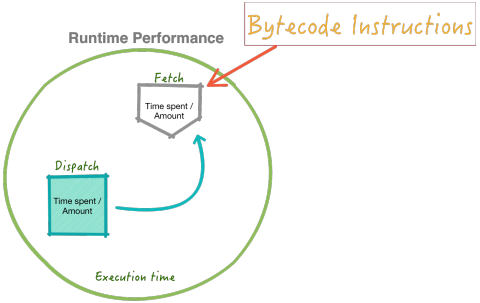
# Prelude: A Deeper Look Into VMs/Interpreters

- VM's runtime performance is largely determined by VM architecture
- Two major types of VM architectures today: Stack-based and register-based VM
- Stack-based VM: employs stacks for data storage/manipulation; needless to specify memory address in bytecode - Bytecode with only 1 operand
- Register-based VM: utilizes multiple registers (2+) for data storage/manipulation; needs to specify memory address in bytecode - instructions with 2-3 operands (similar to x86 asm)
- The problem is really stack vs. register, which is faster.

# Analysis: What Does "Performance" Mean?

We measured the core components of runtime performance

- We measured:
  - Total amount of dispatches: Amount of instruction being dispatched to procedures in the virtual machine
  - Overall dispatch time: The overall CPU time spent executing the instruction dispatches
  - Overall operand-fetch time: The overall CPU time spent fetching operands in bytecode instructions
  - Overall execution time: Total time spent in executing the bytecode



# Measuring Performance Through VM Benchmarks

- Estimates:
  - Stack VM can perform better in fetch performance since it has 1-2 less operands per instruction
  - Register VM can have better performance in dispatch amount since it concentrates amount of instructions by havinv explicitly specified memory addrs
- Approach:
  - We wrote two Turing-equivalent VMs in ANSI C:
    - Conceptum, the stack-based virtual machine, and
    - Inertia, the register-based virtual machine
  - Reasoning: structural similarity has to be insured between the two benchmark comparisons, and handwriting two new VMs is the best way to ensure quality
  - Better timing mechanism built-in from birth
  - 4 short benchmark programs (algorithms) were executed to provide the final results:
    - Fibonacci
    - ExhaustiveCollatz
    - Addition
    - Recursion

# Results - Amount of dispatches (CM=Conceptum, IA=Inertia)

Figure 5.1-A: Fibonacci

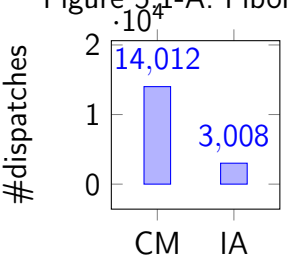



Figure 5.1-B: ExhaustiveCollatz

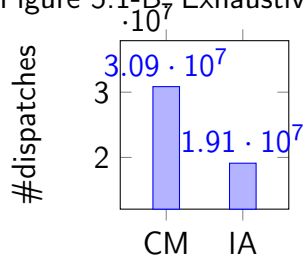



Figure 5.1-C: AddictiveAddition

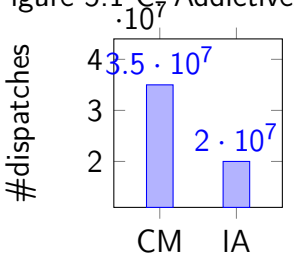
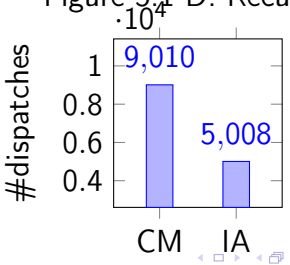



Figure 5.1-D: Recursion



# Results - Execution/dispatch/fetch time (DT=Dispatch Time, FT=Fetch Time, ET=Execution Time)

Figure 5.2-A: Fibonacci

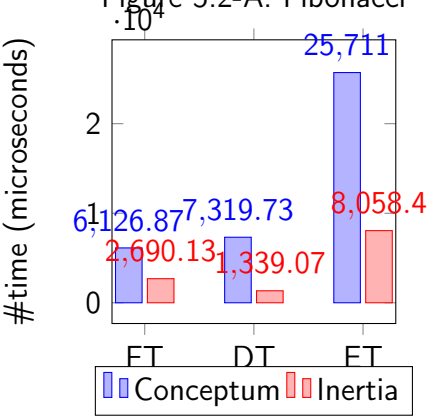
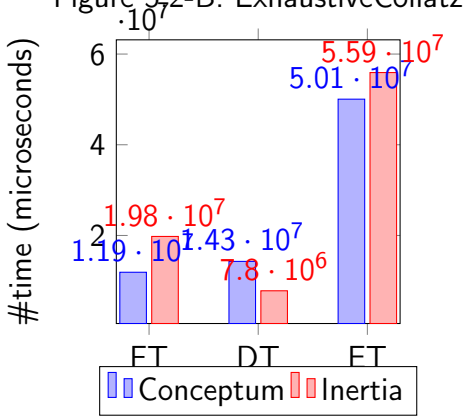



Figure 5.2-B: ExhaustiveCollatz



# Results - Execution/dispatch/fetch time (DT=Dispatch Time, FT=Fetch Time, ET=Execution Time)

Figure 5.2-C: AddictiveAddition

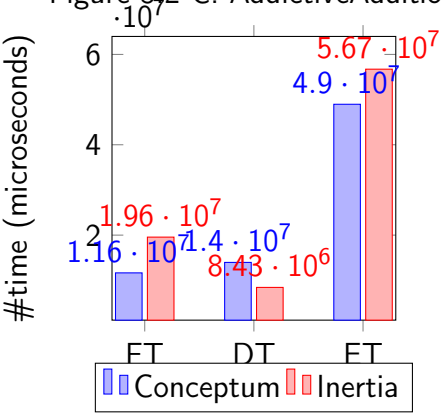
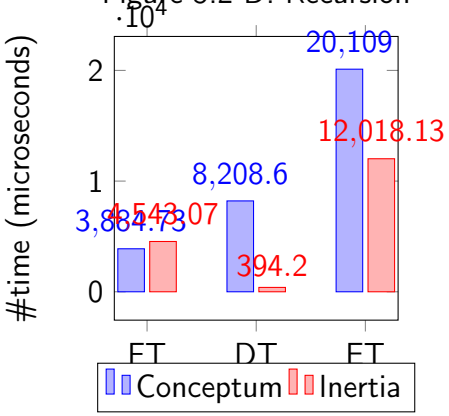



Figure 5.2-D: Recursion



# Conclusion

- Overall, a register-based VM is around 20% faster
- Stack-based VM performed better in fetch time (less)
- If you want to implement a high-performance, compact DSL on limited hardware, go for a register-based VM!
- If you favor simplicity (both in byte code and in code for VM) over performance and want to perform dense read/write to the VM's memory space, implement a stack-based VM!

# Links and extra materials

- This slide available at https://www.github.com/Conceptual-Inertia/presentations/tree/master/plugtalk16oct.pdf
- Source code of Conceptum available at: https://www.github.com/Conceptual-Inertia/Conceptum
- Source code of Inertia available at: https://www.github.com/Conceptual-Inertia/Inertia
- The official paper available at: http://fat-sausage.derros.in/papers/vmplug.pdf
- Questions? Critics? Suggestions? Email: ruijie.fang@temple.edu

# Conceptual Inertia