Cog VM Evolution

Clément Béra

Cog VM ?

- Smalltalk virtual machine
- Default VM for
  - Pharo
  - Squeak
  - Newspeak
  - Cuis

Cog Philosophy

- Open source (MIT)
- Simple
  - Is the optimization / feature worth the added complexity ?
- Cross-Platform (Processors, OS, 32/64 bits)

Execution engine

- VM
  - Execution engine
  - Plugins: graphics, file, etc.

Execution engine

- Interpreter
- JIT
- Memory Manager (including GC)

Evolution

- User and Customer driven

- Where did we start ?
- What problems did we solve ?

Starting blocks

- Interpreter VM
  - Made by Dan Ingalls Team
- Simple Interpreter
- Spaghetti stack
- Smart but simple Memory Manager

Performance !

Short-term delivery

Performance for 3D application

Performance !

Short-term delivery

Performance for 3D application

- Fast Interpreter

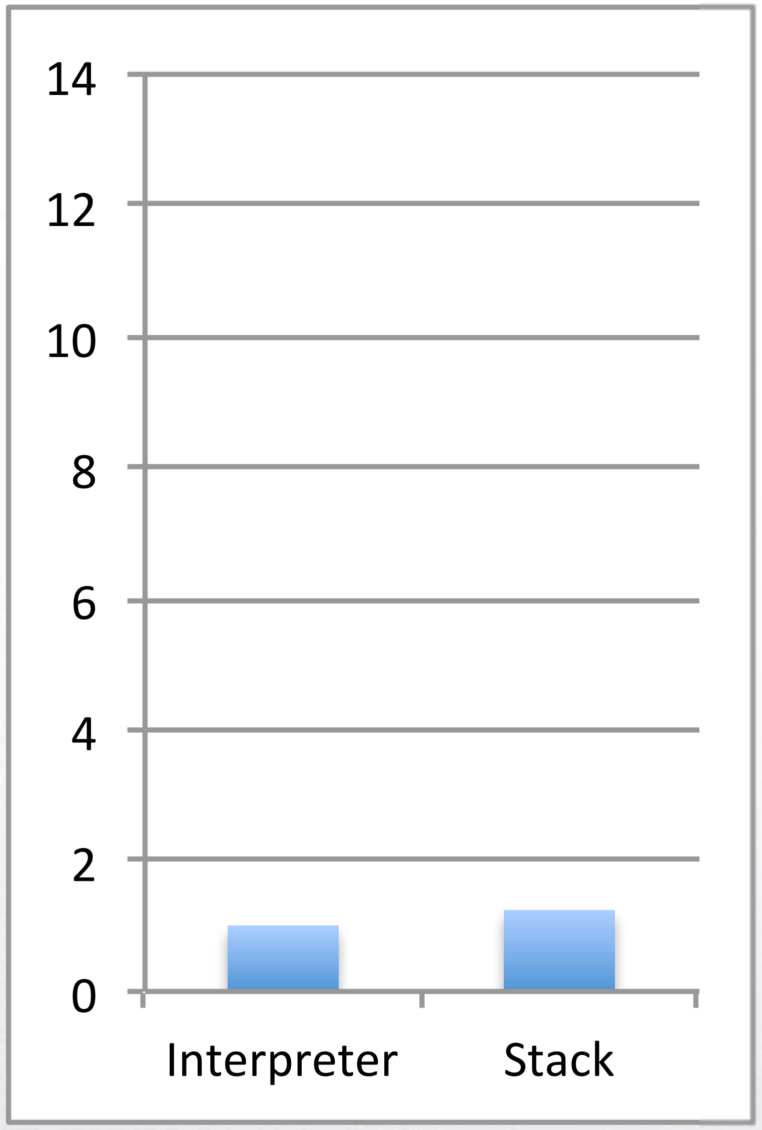
Stack VM

- Context-to-stack Mapping
  - 85% of context allocation removed
  - No copying of arguments
- New hash logic
- Primitive function caching

Issue

- Closure implementation
  - No temporaries
  - BlockAlreadyEvaluated error
  - Non obvious bugs
- New implementation

Binary Tree benchmark



More Performance !

Short-term delivery

Performance for 3D application

More Performance !

Short-term delivery

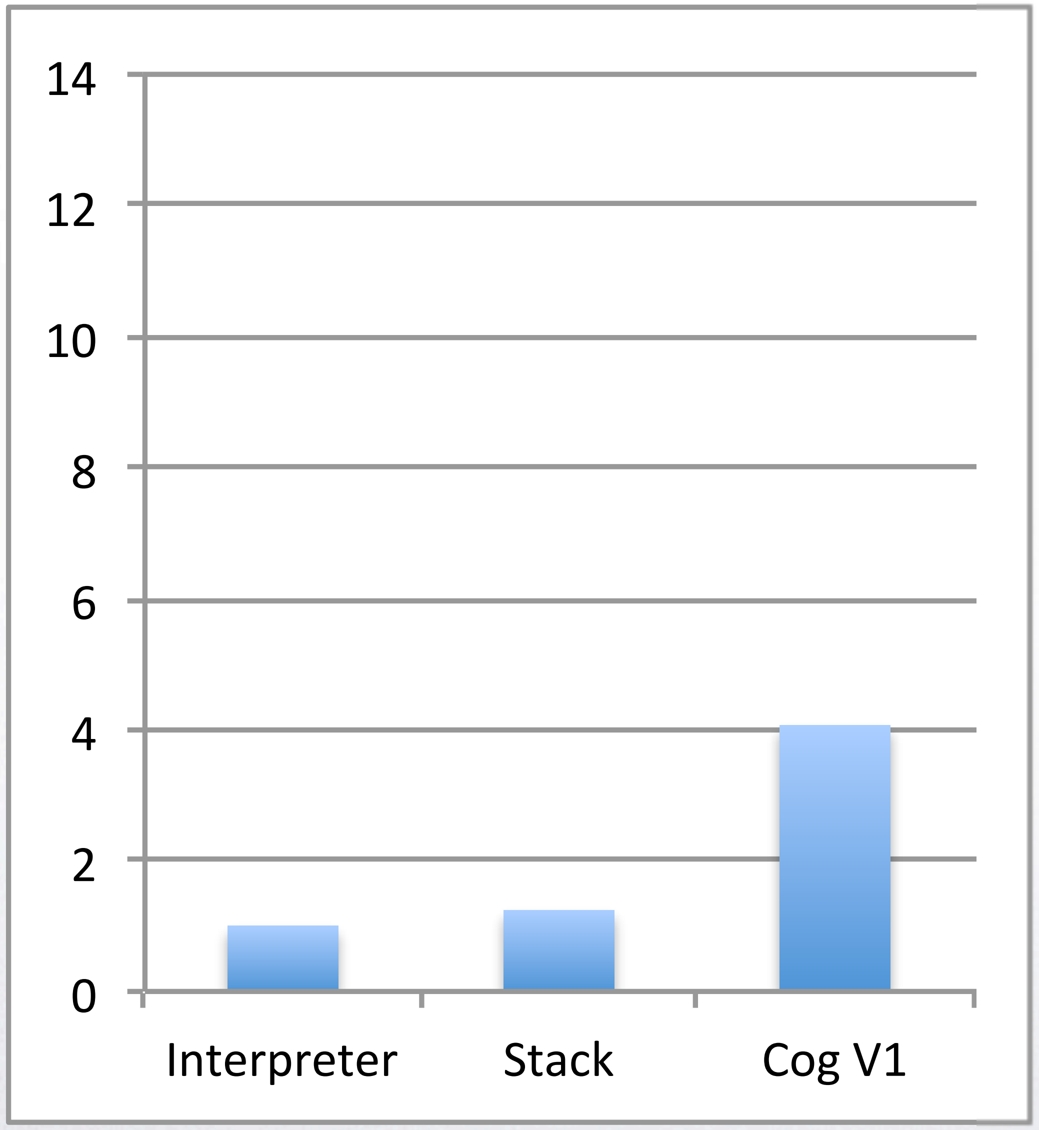
Performance for 3D application

- First JIT compiler

Cog VM

- x86 back-end
- Simple machine code generation
  - Except inline caches

Binary Tree benchmark



JIT Abstractions

Machine back-end	Object Representation		Cogit Implementation
x86	V3		SimpleStackCogit

Yet More Performance !

Short-term delivery

Performance for 3D application

Yet More Performance !

Short-term delivery

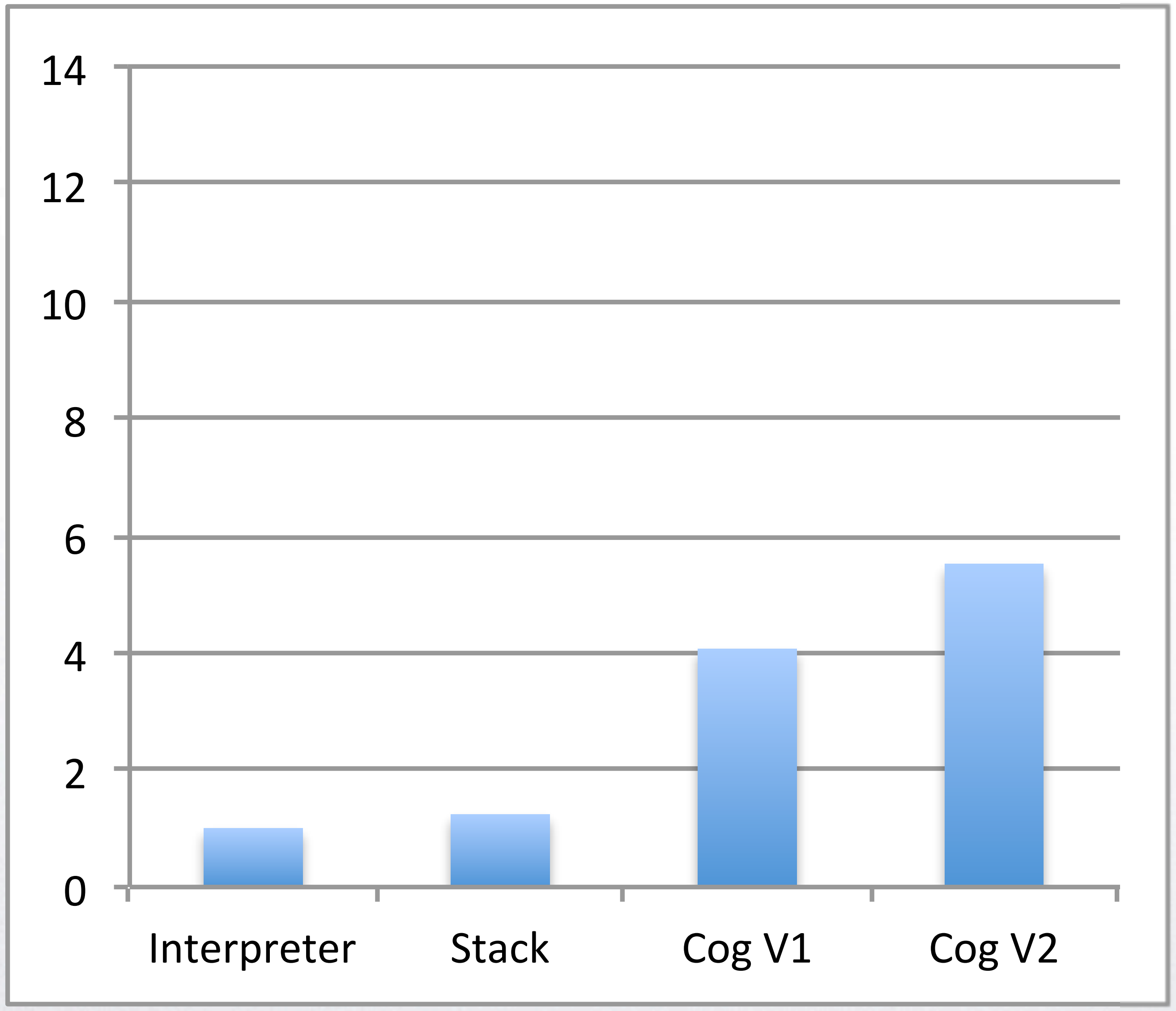
Performance for 3D application

- Second JIT compiler

Cog VM

- Machine code generation
  - Linear scan register allocation
  - Avoids many stack operations
  - Register-based calling convention

Binary Tree benchmark



JIT Abstractions

Machine back-end	Object Representation		Cogit Implementation
x86	V3		SimpleStackCogit StackToRegisterMappingCogit

Newspeak support ?

Newspeak is Smalltalk-like

New kind of sends

Newspeak support ?

- Multiple bytecode set support
- Newspeak specific operations
  - Interpreter
  - Machine code generation

64 bits ?
Images larger than 1 or 2 Gb ?
Moving objects during FFI call-backs ?
Even more performance !
Ephemerons ?
Become is so slow it cannot be used.

64 bits ?
Images larger than 1 or 2 Gb ?
Moving objects during FFI call-backs ?
Even more performance !
Ephemerons ?
Become is so slow it cannot be used.

Short-term delivery

64 bits ?
Images larger than 1 or 2 Gb ?
Moving objects during FFI call-backs ?
Even more performance !
Ephemerons ?
Become is so slow it cannot be used.

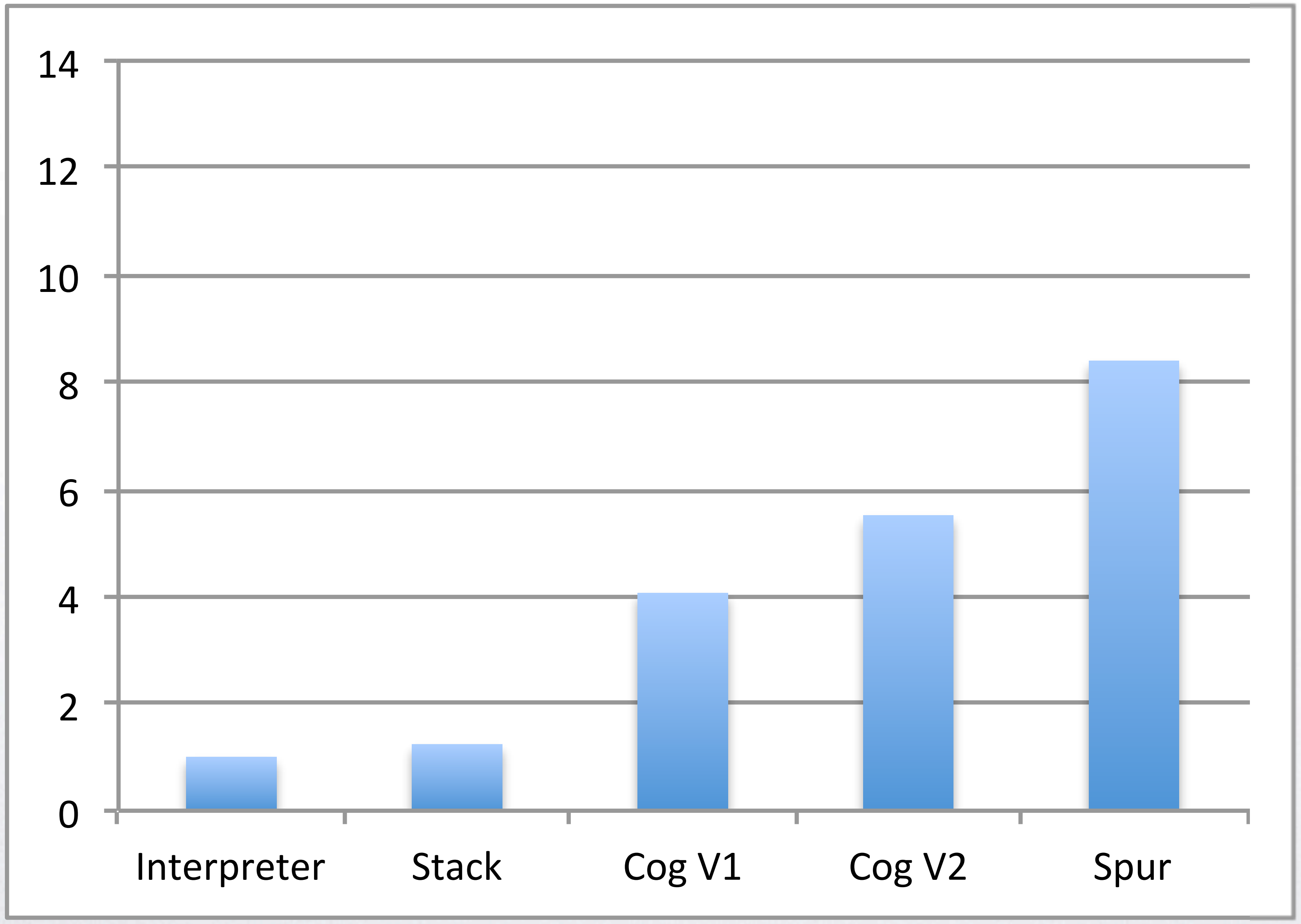
Short-term delivery

- New Memory Manager

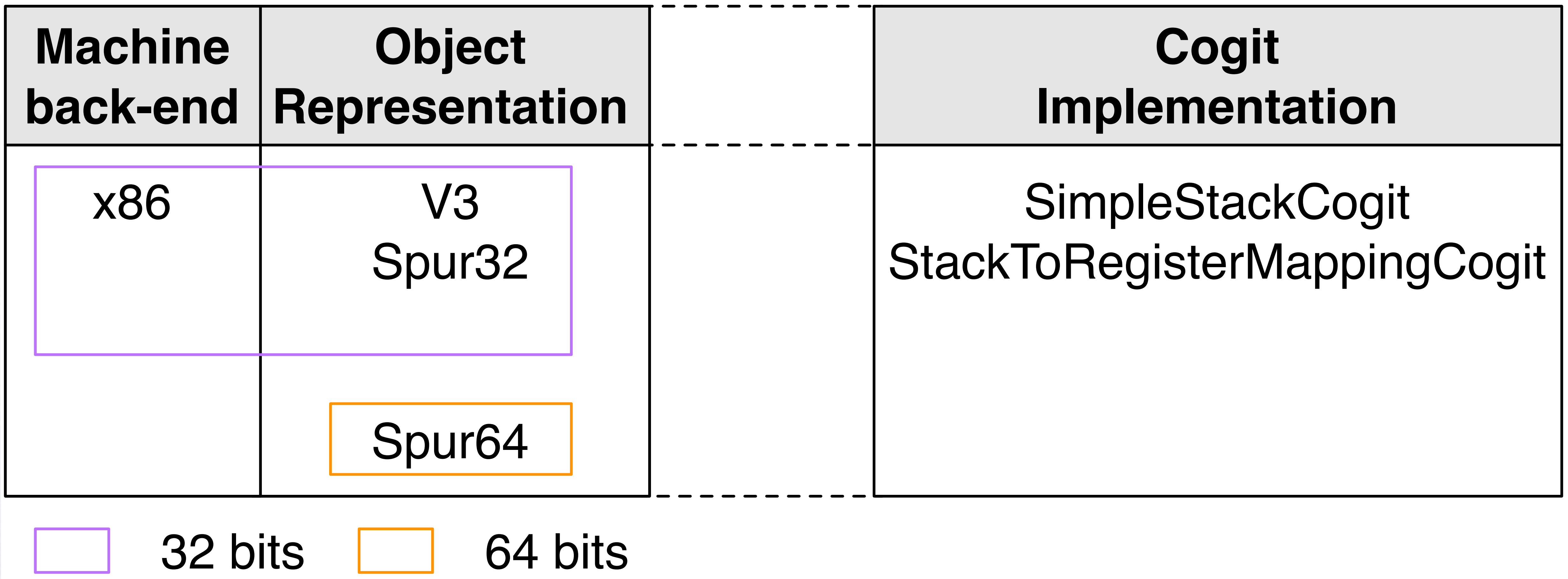
Spur Memory Manager

- Class-table (efficient caches and compactness)
- Efficient scavenging
- Pinned objects
- Fast become
- Segmented Memory
- New object layouts (Ephemerons, ShortArrays)
- Memory representation 64-bits compatible

Binary Tree benchmark



JIT Abstractions



Raspberry Pi performance ?

Scratch support

Raspberry Pi performance ?

Scratch support

- ARMv6 support

ARMv6 support

- JIT ARMv6 back-end
- JIT abstraction over literal management
- JIT abstraction over CISC / RISC

JIT Abstractions

Machine back-end	**Object Representation**	**Literal Manager**	**Cogit Implementation**
x86	V3		SimpleStackCogit
ARMv6	Spur32	Inline	StackToRegisterMappingCogit
		Outline	
	Spur64		

🟪 32 bits 🟧 64 bits

Ryan (contributor)

Working with the Dart VM

Dart runs on more platform than Newspeak.

Ryan (contributor)

Working with the Dart VM

Dart runs on more platform than Newspeak.

- MIPSEL support

JIT Abstractions

Machine back-end	Object Representation	Literal Manager	Cogit Implementation
x86 ARMv6 MIPSEL	V3 Spur32 Spur64	Inline Outline	SimpleStackCogit StackToRegisterMappingCogit

🟪 32 bits 🟧 64 bits

64 bits support ?

64 bits library binding

Heap over 2Gb

64 bits support ?

64 bits library binding

Heap over 2Gb

- x64 support
- Immediate float

JIT Abstractions

Machine back-end	Object Representation	Literal Manager	Cogit Implementation
x86 ARMv6 MIPSEL	V3 Spur32	Inline	SimpleStackCogit StackToRegisterMappingCogit
x64	Spur64	Outline	

🟪 32 bits 🟧 64 bits

Yet even more performance !

Computation lasting
3 to 6 hours

Yet even more performance !

Computation lasting
3 to 6 hours

- Speculative optimizations

Sista VM

- The program introspects

- Optimize the code for performance
- Deoptimize when it took incorrect decisions

Issues

- Bytecode set
- Literal mutability
- Closure implementation

Bytecode set limitations ?

Code generator tools

Sista bytecode set

- Lifting encoding limitations
- Encode instructions for the Sista / Lowcode

Efficient modification trackers ?
Literal inconsistency ?

Efficient modification trackers ?
Literal inconsistency ?

- Read-only objects
- IWST talk this afternoon

- Hopefully allows more compiler optimizations

Closure implementation

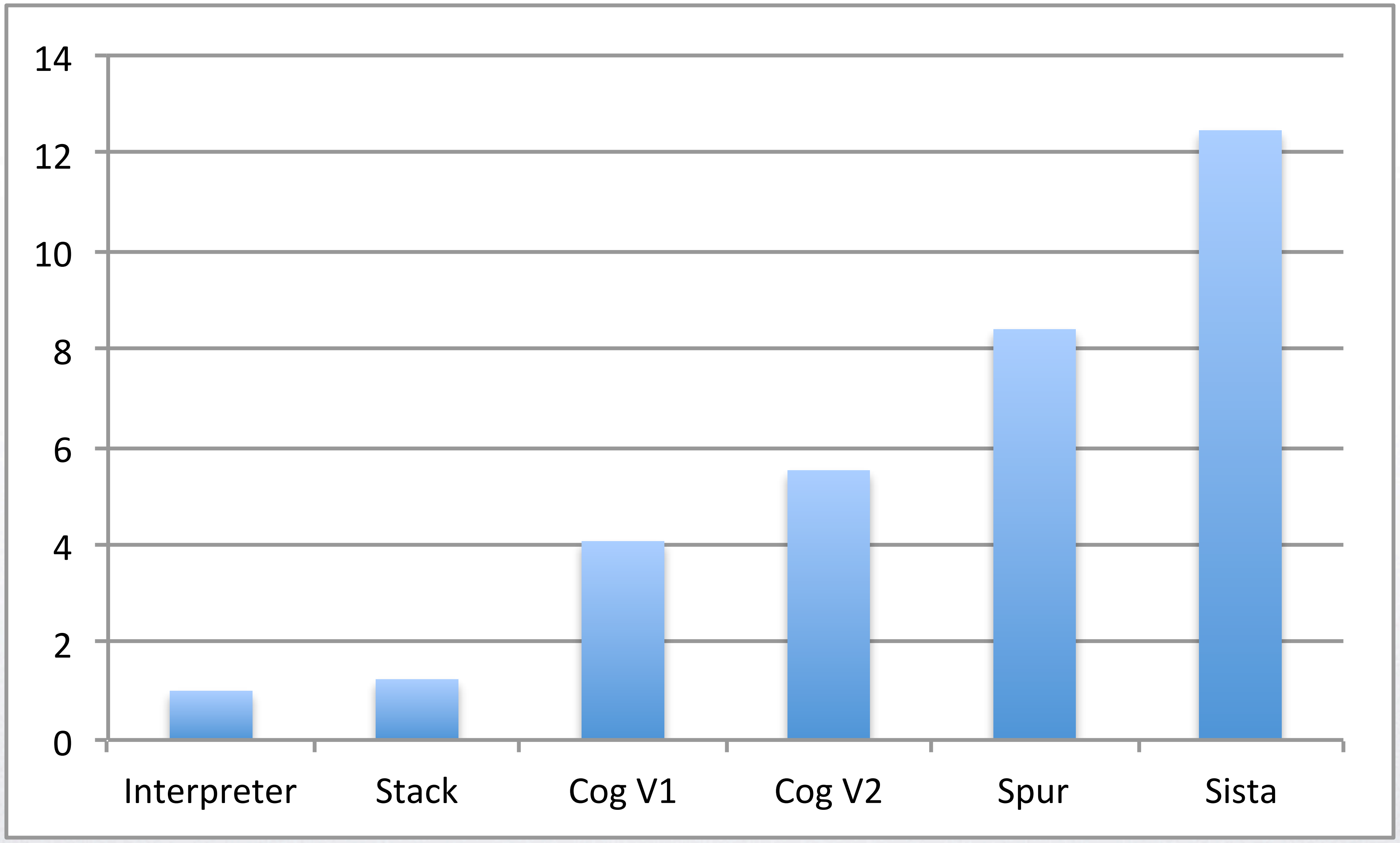
- Method and closure get more similar

- Simplifies part of the VM
- Simplifies the runtime compiler

Integration

- Closed alpha version
- Integrating dependencies:
  - Bytecode set integrated
  - Read-only objects integrated (but disabled)
  - Closure implementation in progress

Binary Tree benchmark



JIT Abstractions

Machine back-end	Object Representation	Literal Manager	Cogit Implementation
x86 ARMv6 MIPSEL	V3 Spur32	Inline	SimpleStackCogit StackToRegisterMappingCogit
x64	Spur64	Outline	RegisterAllocatingCogit SistaCogit

□ 32 bits □ 64 bits

Image compaction ?

Sometimes, large images
when saving

Image compaction ?

Sometimes, large images
when saving

- Better compactor

Pauses ?

0.5 second freezes
in UI application

Pauses ?

0.5 second freezes
in UI application

- Incremental GC

Many hidden parts

...

C compiler warning ?

- C generated from Slang
- Many were fixed
- Towards compilation with -WAll -WError

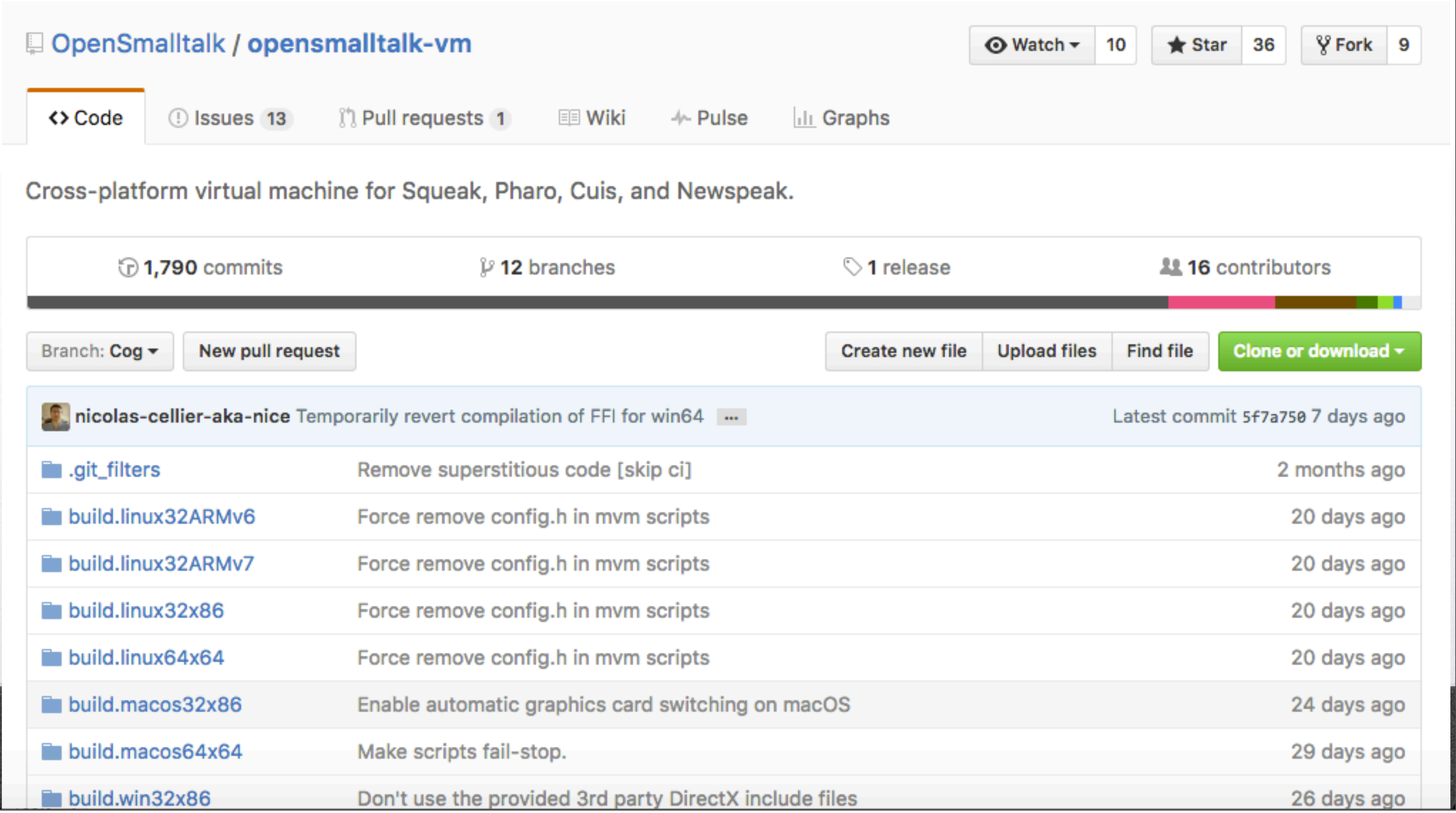
Faster arithmetic ?

- LargeInteger plugin more efficient
  - Computation moved from 8bits to 32 bits
- Different compilation flags

Slang ?

- Slang-to-C compiler
  - Many improvements
  - Type inference

Contribution



OpenSmalltalk / opensmalltalk-vm

Watch 10 | Star 36 | Fork 9

Code | Issues 13 | Pull requests 1 | Wiki | Pulse | Graphs

Cross-platform virtual machine for Squeak, Pharo, Cuis, and Newspeak.

1,790 commits | 12 branches | 1 release | 16 contributors

Branch: Cog | New pull request | Create new file | Upload files | Find file | Clone or download

nicolas-cellier-aka-nice Temporarily revert compilation of FFI for win64 ... Latest commit 5f7a750 7 days ago

.git_filters	Remove superstitious code [skip ci]	2 months ago
build.linux32ARMv6	Force remove config.h in mvm scripts	20 days ago
build.linux32ARMv7	Force remove config.h in mvm scripts	20 days ago
build.linux32x86	Force remove config.h in mvm scripts	20 days ago
build.linux64x64	Force remove config.h in mvm scripts	20 days ago
build.macos32x86	Enable automatic graphics card switching on macOS	24 days ago
build.macos64x64	Make scripts fail-stop.	29 days ago
build.win32x86	Don't use the provided 3rd party DirectX include files	26 days ago

Cog VM team

- Started with Eliot Miranda
- Many more contributors now:
  - Tim Rowledge
  - Clément Béra (myself)
  - Nicolas Cellier
  - Fabio Niephaus & Tim Felgentreff
  - Ryan Macnak

Conclusion

- Lots of new features and improvements over years
- A lot more is incoming
- If you want to support, talk to us !
  - ARMv8 ?
  - Incremental GC ?
  - Performance (escaping, floats) ?