Cog Projects

Cog is an evolving system. Some projects are being worked on, some are as yet just dreams. We’re interested in collaborators 😉   This page may help you determine where you can help.

Spur

Spur is a new memory manager whose core is a more efficient object representation that has a single header format for both 64-bit and 32-bit implementations.

Spur is released as Squeak 5.0, Newspeak and Pharo 5, and 64-bit Squeak 5.0 is in use, but green. There are still important missing components:

– an incremental global mark-sweep-compact collector that avoids long pause times running a stop-the-world scan-mark-compact during interactive use.

– an improved per-segment compaction algorithm to replace the second cut compaction algorithm (pig compact) that is slow and reorders objects.

– a solution to the problem of free space in segments containing pinned objects at snapshot time (hopefully solved by the improved per-segment compaction algorithm)

– A scheme to avoid garbage collecting large class libraries in “static” applications (this is what VisualWorks calls “Perm Save”). Spur’s segmentation could allow e.g. the first segment to be managed as a “perm segment” that would not be garbage collected.

– image segment interoperability between Spur 32-bit and 64-bit versions and between Spur and previous Squeak versions. (It is hoped that such support would be done at the image level only without VM support).

64-bit Spur is progressing. There is a functional system and a Cogit and StackInterpreter for x64 Linux & Mac OS X. Still required are

– 64-bit VMs for Windows
– an ARMv8 backend for the Cogit (Tim would like to work on this; anybody with funding should contact he or me)

Sista/Scorch

02 Closure Bytecodes
03 BlueBook Methods
04 Closure Compiler
05 Closure Stats
06 Simulate Out Of The Bochs
07 The Idée Fixe and the Perfected Profiler
08 Under Cover Contexts and the Big Frame-Up
09 Build me a JIT as fast as you can…
10 An Arranged Marriage – The Honeymoon

DOWNLOADS
Closure Bootstrap
Cog VMs

LINKS
CGO Conferrence
Eleazar & I in Concepcíon
Site Stats

SPUR
A Spur gear for Cog
Lazy Become
Lazy Become and Primitives

RSS FEEDS
Posts
Comments

META
Log in

SEARCH
Find

Sista, the Speculative Inlining Small-talk Architecture, is a combination of an image-level adaptive optimizer and low-level Cogit support for performance counters and unsafe inline primitives, which together can generate much more efficient machine code implementations of often-executed codes. Scorch is the image level component, Clément's SSA optimizer and dynamic deoptimizer. Clément and Eliot are working on this. We want to release a first production version towards the end of 2016.

Threaded FFI

The Cog VM is single-threaded, providing green threads to the Smalltalk level, as do most Smalltalk VMs. There is a prototype subclass of the CoInterpreter, CoInterpreterMT, which provides a Python-like multi-threaded VM where any number of threads can share the VM with only one running the VM at any one time. This uses an extremely simple low-level scheme designed by David Simmons for releasing and re-acquiring ownership of the VM. This results in any and all FFI calls being non-blocking since there is such a low overhead to making a non-blocking FFI call that all FFI calls can be non-blocking. Eliot has worked on this, implementing the prototype and the reentrant FFI plugin it requires. But this needs to be revived. It would be a really powerful enhancement.

ABI Compiler

Complex ABIs such as x86-64 (where for example the contents of structs are scattered across available integer and floating-point argument registers) are difficult to implement via interpretation (where a signature is interpreted to marshal arguments for a specific call). For both efficiency and correctness a compiled implementation, where for example marshalling code is generated on first use of a given call in any run, is preferrable. While quite a few people are interested (Igor, Ronie, Eliot), and Ronie's Lowcode support is being added to the Cogit, no one is actively working on the image-level ABI compiler. This is less important than anticipated because the existing FFI plugin has been successfully ported to support x64 on 64-bit Spur.

Event-driven VM

The Cog VM, being derived from the standard Squeak VM, has an unsatisfactory idle implementation, with the background process calling a yield primitive that gives back time to the OS. This causes the VM to consume of the order of 3%-5% at idle. Instead the VM should block awaiting events when it has nothing to do. There are various attempts at an event-driven Squeak VM (e.g. for Android) but none of this has been integrated into Cog, nor has the idle issue been addressed. Closely related is to replace the use of select(2) for i/o polling with kqueue(2) or epoll(2) on platforms that support it.

Native UI/True Headlessness

Vassili Bykov has implemented a native Windows UI above Cog's callback support. But a more comprehensive approach is possible, factoring out the GUI support in e.g. the Mac and Windows VMs and replacing it with an FFI-based implementation above the threaded FFI. Such a VM would be truly headless, containing no GUI code in the base VM.

Embedding/VM as a DLL

It would be useful to be able to embed a Smalltalk system running above Cog in other clients, e.g. as a DLL. Cog has callbacks, above which an API can be built to access Smalltalk, but lots of refactoring and repackaging of the VM sources is necessary to realise this.

Web Browser "Plugin"

For a really cool project how about grabbing Bert Freudenberg's VMMakerJS Squeak-vm-on-JavaScript, extracting the event handling and rendering part into a component and connecting it to the Cog VM via sockets to give us a really fast web plugin? This would be portable across browsers, and would decouple updating the VM from updating the plugin.

Multi-threading

While multi-threading seems like an obvious and important direction to take the system in, making the VM multi-threaded per-se does not provide any benefit before the Smalltalk image is made thread-safe and that is probably more work than providing a multi-threaded VM. Hence a potentially more profitable approach is to concentrate on federating multiple VMs running multiple images, communicating through the threaded FFI.

Read-only objects (was "Per-object immutability")

Read-only object support is the ability to mark individual objects as read-only. This can be used to make literals immutable, and to provide efficient write-through support for orthogonal persistence schemes, and to implement debugging traps, etc. Clément has implemented the VM and Cogit support. So image level work can begin.

Subversion -> Git

Subversion is the current repository for generated Cog VM sources. This is seen as less desirable than a git repository for integration with CI. Moving Cog from subversion to git may seem straight-forward but needs to be done with tact to help people over the transition.

Merging with the Squeak VM

Cog started off from Qwaq's fork of the Squeak VM. As such, Cog and the Squeak VM's ObjectMemory hierarchy differ in detail, and Cog's Slang has changed considerably to accomodate the much more complex VM. Merging the two is desirable. We would also like to see an Interpreter VM for Spur.