NRE: 8.6's non-recursive engine

Miguel Sofer

Tcl2008
Manassas, VA

State of a computation in 8.5

- **C-stack** (1 per thread)
  - Nesting of C-function calls
- **CallFrame stack** (1 per interp)
  - Nesting of [proc], [apply], [namespace eval]; this is what [upvar] and [uplevel] navigate
- **Tcl evaluation stack** (1 per execEnv/interp)
  - Bytecode engine stack and workspace
  - Special allocator supported (TclStackAlloc)
- **CmdFrame stack** (1 per interp)

The C stack in 8.5 (simplified)

A simple script

```
proc foo {} {
    set x 1
    moo
    set x 2
}
proc moo {} {
    set x 1
    puts $x;# WE ARE HERE
    set x 2
}
foo
```

Stack Trace (simplified)

```
1: Tcl_PutsObjCmd
2: Tcl_EvalObjv [puts $x]
3: TclExecuteByteCode [moo]
4: TclObjInterpProc
5: Tcl_EvalObjv [moo]
6: TclExecuteByteCode [foo]
7: TclObjInterpProc
8: Tcl_EvalObjv [foo]
```

The C stack in 8.5 (the real thing)

File Edit View Terminal Tabs Help

mig@oli: ~/Personal/Tcl/ownPubs/Tcl2008/code | mig@oli: /tmp/tcl | mig@oli: /tmp/tcl

```
Breakpoint 1, Tcl_PutsObjCmd (dummy=0x0, interp=0x81714d0, objc=2, objv=0x8171ce0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclIOCmd.c:115
115     /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclIOCmd.c: No such file or directory.
        in /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclIOCmd.c
(gdb) bt
#0  Tcl_PutsObjCmd (dummy=0x0, interp=0x81714d0, objc=2, objv=0x8171ce0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclIOCmd.c:115
#1  0x080c9921 in TclEvalObjvInternal (interp=0x81714d0, objc=2, objv=0x8171ce0, command=0xffffffff <Address 0xffffffff out of bounds>, length=-1, flags=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclBasic.c:3690
#2  0x08124fa4 in TclExecuteByteCode (interp=0x81714d0, codePtr=0x81a75a8)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclExecute.c:2340
#3  0x08088776 in TclObjInterpProcCore (interp=0x81714d0, procNameObj=0x81a86e8, skip=1, errorProc=0x8088d06 <MakeProcError>)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclProc.c:1748
#4  0x080886b3 in TclObjInterpProc (clientData=0x81a0370, interp=0x81714d0, objc=1, objv=0x8171c68)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclProc.c:1642
#5  0x080c9921 in TclEvalObjvInternal (interp=0x81714d0, objc=1, objv=0x8171c68, command=0xffffffff <Address 0xffffffff out of bounds>, length=-1, flags=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclBasic.c:3690
#6  0x08124fa4 in TclExecuteByteCode (interp=0x81714d0, codePtr=0x81a7460)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclExecute.c:2340
#7  0x08088776 in TclObjInterpProcCore (interp=0x81714d0, procNameObj=0x8186480, skip=1, errorProc=0x8088d06 <MakeProcError>)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclProc.c:1748
#8  0x080886b3 in TclObjInterpProc (clientData=0x818c4b0, interp=0x81714d0, objc=1, objv=0x8171bf0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclProc.c:1642
#9  0x080c9921 in TclEvalObjvInternal (interp=0x81714d0, objc=1, objv=0x8171bf0, command=0xffffffff <Address 0xffffffff out of bounds>, length=-1, flags=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclBasic.c:3690
#10 0x08124fa4 in TclExecuteByteCode (interp=0x81714d0, codePtr=0x81a7358)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclExecute.c:2340
#11 0x0812300a in TclCompEvalObj (interp=0x81714d0, objPtr=0x81864b0, invoker=0x0, word=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclExecute.c:1474
#12 0x080cb671 in TclEvalObjEx (interp=0x81714d0, objPtr=0x81864b0, flags=131072, invoker=0x0, word=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclBasic.c:5095
#13 0x080cb284 in Tcl_EvalObjEx (interp=0x81714d0, objPtr=0x813cd23, flags=131072)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclBasic.c:4903
#14 0x0813c1d4 in Tcl_RecordAndEvalObj (interp=0x81714d0, cmdPtr=0x81864b0, flags=131072)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclHistory.c:161
#15 0x0806f882 in Tcl_Main (argc=-1, argv=0xbfe28b98, appInitProc=0x805458b <Tcl_AppInit>)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../generic/tclMain.c:554
#16 0x0805457c in main (argc=Cannot access memory at address 0x0
) at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.5.5/unix/../unix/tclAppInit.c:87
(gdb)
```

The C stack in 8.5 (a problem?)

Stack overflow and recursion limits ...

- Platforms with small *default* stack (BSD and other unixes)
- Stack-hungry platforms (Win debug builds)
- Severely stack constrained platforms (Cisco, phones?)
- **mod-8-3-branch** version to reduce stack consumption

The C stack in 8.6 (simplified)

A simple script

```
proc foo {} {
    set x 1
    moo
    set x 2
}
proc moo {} {
    set x 1
    puts $x;# WE ARE HERE
    set x 2
}
foo
```

Stack Trace (simplified)

```
1: Tcl_PutsObjCmd
2: Tcl_EvalObjv [puts $x]
3: TclExecuteByteCode [moo]
   TclObjInterpProc
   Tcl_EvalObjv [moo]
   TclExecuteByteCode [foo]
   TclObjInterpProc
4: Tcl_EvalObjv [foo]
```

The C stack in 8.6 (the real thing)

File Edit View Terminal Tabs Help

mig@oli: ~/Personal/Tcl/ownPubs/Tcl2008/code | mig@oli: /tmp/tcl | mig@oli: /tmp/tcl

```
Breakpoint 1, Tcl_PutsObjCmd (dummy=0x0, interp=0x81924d0, objc=2, objv=0x8192d7c)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclIOCmd.c:115
115     /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclIOCmd.c: No such file or directory.
        in /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclIOCmd.c
(gdb) bt
#0  Tcl_PutsObjCmd (dummy=0x0, interp=0x81924d0, objc=2, objv=0x8192d7c)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclIOCmd.c:115
#1  0x080dcfcd in NRRunObjProc (data=0x81c3ac4, interp=0x81924d0, result=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclBasic.c:4270
#2  0x080dcd62 in TclNRRunCallbacks (interp=0x81924d0, result=0, rootPtr=0x81a9148, tebcCall=1)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclBasic.c:4208
#3 0x0813f3f6 in TclExecuteByteCode (interp=0x81924d0, codePtr=0x81ccdd0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclExecute.c:2811
#4  0x080dd06c in NRCallTEBC (data=0x81ac89c, interp=0x81924d0, result=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclBasic.c:4292
#5  0x080dcd62 in TclNRRunCallbacks (interp=0x81924d0, result=0, rootPtr=0x0, tebcCall=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclBasic.c:4208
#6  0x080deeec in TclEvalObjEx (interp=0x81924d0, objPtr=0x81c3ac4, flags=131072, invoker=0x0, word=0)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclBasic.c:5684
#7  0x080dee73 in Tcl_EvalObjEx (interp=0x81924d0, objPtr=0x81c3ac4, flags=131072)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclBasic.c:5665
#8  0x08156a08 in Tcl_RecordAndEvalObj (interp=0x81924d0, cmdPtr=0x81b4488, flags=131072)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclHistory.c:161
#9  0x08071232 in Tcl_Main (argc=-1, argv=0xbff4d4b8, appInitProc=0x80559db <Tcl_AppInit>)
    at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../generic/tclMain.c:557
#10 0x080559cc in main (argc=135882504, argv=0x8192948) at /home/mig/Personal/Tcl/ownPubs/Tcl2008/code/tcl8.6a3/unix/../unix/tclAppInit.c:87
```

Full compatibility (*)

- Only additions to the API, no changes
- Modification of struct Command, changes to Tcl_[GS]etCommandInfo designed not to break extensions that call *objProc directly
- Changed structs (extended): Interp, Command, ExecEnv

The C API (TIP #322)

The functions that are provided for the extension writer are

- **Tcl_NRCreateCommand**
- **Tcl_NREvalObj**
- **Tcl_NREvalObjv**
- **Tcl_NRCmdSwap**
- **Tcl_NRAddCallback**

An NRE-enabled command requires two implementations: the regular objProc and a new nreProc. NRE provides a utility function **Tcl_NRCallObjProc** to make this task essentially trivial

How to exploit NRE

File Edit View History Bookmarks Tools Help

tcl-WIP: NRE | TIP #322: Publish the NRE... | tcl-WIP: Exploiting NRE | tcl-WIP: Coroutines

A command currently implemented as

```
int
MyCmdObjProc(
    ClientData clientData,
    Tcl_Interp *interp,
    int objc,                   /* Number of arguments. */
    Tcl_Obj *const objv[])      /* Argument objects. */
{
    <preparation>
    result = Tcl_EvalObjEx(interp, objPtr, flags);
    <postprocessing>
    <cleanup>
    return result;
}

Tcl_CreateObjCommand(interp, name, MyCmdObjProc, clientData, deleteProc);
```

... would look after adaptation as

```
int
MyCmdNreProc(
    ClientData clientData,
    Tcl_Interp *interp,
    int objc,                   /* Number of arguments. */
    Tcl_Obj *const objv[])      /* Argument objects. */
{
    <preparation>
    Tcl_NRAddCallback(interp, MyPostProc, data0, data1, NULL, NULL);
    return Tcl_NREvalObj(interp, objPtr, flags);
}

int
MyPostProc(
    ClientData data[],
    Tcl_Interp *interp,
    int result)
{
     Tcl_Obj  *fooPtr = data[0]; /* data0 retrieved */
     MyStruct *mooPtr = data[1]; /* data1 retrieved */

    <postprocessing>
    <cleanup>
    return result;
}

MyCmdObjProc(
    ClientData clientData,
    Tcl_Interp *interp,
    int objc,                   /* Number of arguments. */
    Tcl_Obj *const objv[])      /* Argument objects. */
{
    return Tcl_NRCallObjProc(interp, MyCmdNreProc, clientData, objc, objv);
}

Tcl_NRCreateCommand(interp, name, MyCmdObjProc, MyCmdNreProc, clientData, dele
```

NRE-enabled core commands

- procs
- lambdas
- same-interp aliases
- namespace imports, ensembles (snit!)
- [eval], [uplevel], [namespace eval]
- TclOO methods
- [if], [for], [while], [foreach]

Non-enabled commands (yet?)

Mainly commands that change the execEnv

- [interp eval], [slave eval]
- Coroutines

Also

- [source]
- ...

From the C side: no NR-API for Tcl_Eval(), Tcl_EvalEx()

How does this happen? Concept

- Trampolining: do not *invoke* a function, rather ask your *caller* to do it for you (and then return)
- Callbacks
- TEBC: freeze a ByteCode, start running another one

Stack Trace (simplified)

```
1: Tcl_PutsObjCmd
2: Tcl_EvalObjv [puts $x]
3: TclExecuteByteCode [moo]
   TclObjInterpProc
   Tcl_EvalObjv [moo]
   TclExecuteByteCode [foo]
   TclObjInterpProc
4: Tcl_EvalObjv [foo]
```

Callbacks

- New stack for *callbacks*: maintained by the ExecEnv
- The workhorse is **TclNRRunCallbacks** (the trampoline)

```
int TclNRRunCallbacks (
 Tcl_Interp * interp,
 int result,
 struct TEOV_callback * rootPtr,
 int tebcCall);
```

  - loops running the EE's top callback
  - Passes a callback's result to the next callback
  - allows TEBC calls to ”leak through” (when called from TEBC) (*)

Callbacks

- Callback API:

```
typedef int (Tcl_NRPostProc) (
  ClientData data[],
  Tcl_Interp *interp,
  int result
);
```

- Callback struct

```
typedef struct TEOV_callback {
  Tcl_NRPostProc *procPtr;
  ClientData data[4];
  struct TEOV_callback *nextPtr;
} TEOV_callback;
```

TEBC magic

TEBC has many (100?) local variables, but ...

```
typedef struct BottomData {
   struct BottomData *prevBottomPtr;
   TEOV_callback *rootPtr; /* State when this bytecode execution */
   ByteCode *codePtr;      /* began; constant until it returns   */
   /* ------------------------------------------------------------ */
   TEOV_callback *atExitPtr; /* used on return FROM here          */
   /* ------------------------------------------------------------ */
   unsigned char *pc;      /* These fields are used on return TO */
   ptrdiff_t *catchTop;    /* this level: they record the state  */
   int cleanup;            /* when  a new codePtr was received   */
   Tcl_Obj *auxObjList;    /* for NR execution                   */
} BottomData;
```

State of a computation in 8.6a3

- **C-stack** (1 per thread) (*lightly loaded*)
- **CallFrame stack** (1 per interp)
- **Tcl evaluation stack** (1 per execEnv), includes ***NEW*** BottomData stack.
- **CmdFrame stack** (1 per interp)
- ***NEW*** **CallbackStack** (1 per execEnv)

There are definite opportunities here for simplification and merging ...

New possibilities

NRE opens a world of new possibilities:

- Once TEBC knows how to freeze an execution ... put it aside and save it for later ⇶ coroutines
- Edit the Callback stack, rearrange the order of computations ⇶ proper tailcalls
- Schedule new evals from callbacks ⇶ CPS?
- ??? ⇶ ???

Miscelanea

- Not optimized – should get better during beta
- Some of the internal things may yet change
- C API not frozen ... TIP not yet submitted to a vote
- TEBC currently looks like a bomb exploded in it
- Debugging NRE is no fun; will have to invent something