New Features in DyninstAPI and Dyner Command Line Tool

Mehmet Altinel

What's new in DyninstAPI?

- Bug fixes (and "Brand-new" bugs)
- Enhanced type system
- Fork - Exec - Exit Callbacks (Solaris)
- Control Flow Graph Generation
- Line Number Information
- Access to local (stack) variables

Main Bug Fixes

- More functions are considered instrumentable
- Execution of oneTimeCode even if the mutatee is currently stopped inside a system call
- Creation of BPatch_sourceObject class (generic parent class for image, function, module)

Main Bug Fixes (Cont'd)

- Code to detect re-attach to already modified program
- Can run new programs after the first BPatch_thread exit
- Support for disabling instrumentation code when called from instrumentation code (recursive Instrumentation guard)

Main Bug Fixes (Cont'd)

- New restrictions on use of BPatch_callAfter and BPatch_callBefore
  - To allow for first instruction vs. first instruction after stack activation record setup
- Bugs in process management on the Alpha platform were fixed

Type System Enhancements

- GNU Stabs and COFF Debug Formats Support
- Support for non-integer scalars, structures and arrays
- Local (Stack) variable access
- Type information for all variables and functions

Type System Enhancements (Cont'd)

- Creation of new types for patched code
- Availability on more platforms (Sparc, x86 Solaris, Linux, Aix, Alpha)
- New classes: BPatch_type, BPatch_field and BPatch_localVar
- New interfaces in BPatch_image, BPatch_module and BPatch_function

BPatch_type Class

- getName - returns the symbolic name
- getSize - returns the size of the type
- getComponents - returns the fields of struct/union
- type - returns data class (structure, union, array, ...)
- getType - returns the type of the pointer, array element
- getLow, getHigh - returns bounds for arrays
- isCompatible(BPatch_type *t2) - test compatibility of two types

Fork - Exec - Exit Callbacks

- Interfaces (Available on Solaris):
  - registerPreForkCallback / registerPostForkCallback
  - registerExecCallback
  - registerExitCallback
- Allow users to install a function to be called when:
  - a Bpatch_thread forks a process
  - just after the fork is performed
  - a process executes an exec system call
  - a process terminates

Control Flow Graph Generation

- BPatch_flowGraph contains information about
  - machine code basic blocks
    - all basic blocks in the function, entry basic block, exit basic blocks
  - natural loops in machine code
    - natural loop : entry basic block to the loop dominates all basic blocks in the loop (single entry loops)

Control Flow Graph Generation

- Passes over machine instructions twice
  - one for finding the leaders (target addresses of branch inst's)
  - one for finding the flow of control using leaders
- Unreachable basic blocks are deleted from CFG
- Loop structure is not built until needed
- Dominator information is not initialized until needed

Line Number Information

- Absolute addresses are obtained from source line numbers and vice versa
- Interfaces in BPatch_function, BPatch_module, BPatch_image and BPatch_thread classes

Line Number Information (Cont'd)

- Examples:
  - bool BPatch_module ::getLineToAddr(unsigned short lineNo, BPatch_Vector<unsigned long>& buffer, bool exactMatch = true);
  - bool Bpatch_thread::getLineAndFile(unsigned long addr,unsigned short& lineNo, char* fileName,int length);
- Instrumentation points are created with:
  - BPatch_image::createInstPointAtAddr(address)

DyninstAPI Status

	Sparc Solaris	x86 Solaris	Alpha DU	MIPS IRIX	Power AIX	x86 Linux	NT
Debug Parsing	✔	✔	✔	✘	✔	✔	✘
Fork-Exec-Exit Callbacks	✔	✔	IP	IP	✘	✘	✘
Control Flow Graph	✔	✘	✘	IP	✘	✘	✘
Line Number Info	✔	✔	✘	✘	✘	✔	✘
Access to Local Variables	✔	✔	✔	✘	✔	✔	✘
Arbitrary Inst. Points	IP	✘	✔	IP	✔	✘	✘

Up coming Dyninst Projects

- Short Term Goals:
  - Dynamic call site detection
  - Better type information support for C++ programs
  - Auto detection of debug formats
  - Arbitrary instrumentation points on Sparc & MIPS platforms

Up coming Dyninst Projects (Cont'd)

- Long Term Goals:
  - Generic callback from mutator to mutatee
  - Function parameters access
  - Dyninst as binary editor
  - Floating point code generation
  - Optimizing code generation
- Your suggestions?

Dyner: A Utility for Using DyninstAPI

- TCL-based interactive command line tool
- Easy access to most Dyninst features
- Useful for debugging applications and performance monitoring
- Available in multiple platforms

Main Dyner Features

- Process creation and manipulation
- Snippet insertion
- Conditional/unconditional breakpoints
- Declaration of new variables
- Type information access
- Trace and count of function calls

Dyner Command Summary

- at: Insert a code snippet
- break: Insert a breakpoint
- count, trace, untrace: Function count and trace facility
- declare: Create a new variable
- load: Load either an executable, shared library or even a source file

Dyner Commands (contd)

- print: Show contents and type of dyner variable
- replace, removecall: Manipulate function calls in the program
- run, kill, detach: Process manipulation
- show: Navigational display of type information

Dyner Commands (contd)

- source: Execute dyner commands stored in a file
- whatis: Display detailed information about variables in the mutatee program.

An Example Dyner Session
(Mutatee Program)

```
#include <stdio.h>
int findSQ(int inp) {
        return inp * inp;
}

int AddTwo(int inp) {
        return inp+2;
}

int main() {
        int val = 2;
        val = findSQ(val);   val = findSQ(val);    val = findSQ(val);
        printf("Value %d\n", val);
        return 0;
}
```

An Example Dyner Session

```
% load testDyner
Loading "testDyner"
% declare int cnt
% at main entry { cnt = 0; }
% at findSQ entry { cnt++; }
% at termination { printf("findSQ called %d times\n", cnt); }
% replace call main:2 with AddTwo
% run
Value 36
findSQ called 2 times

Application exited.
%
```

Dyner Status and Future Work

- Running on:
  - Sparc/x86 Solaris, Alpha DU, MIPS IRIX, Power AIX, x86 Linux and NT
  - Not all the commands are available on all platforms
- Call Graph Facility
- Better support for performance monitoring
- Construction of standard snippet library