Insecticide: the ultimate Python debugger

Michael Salib
msalib@alum.mit.edu

June 9, 2004

Background

- I interviewed at Green Hills recently
  - before they started to suck
- They make compilers/debuggers/etc for embedded systems
- Many embedded chips have a trace port that continually dumps out the current CPU state at high speed
- They have a super probe that connects to processor trace ports and stores the data for access by a debugger running on a PC

Time travel

- Having the complete system state allows the debugger to implement time travel
  - you can step backwards in addition to forwards
  - i.e., debugger has a “previous line” button
  - you can also move back up the call chain and debug function calls that have already returned
- Limited by memory
  - trace boards have about 1 GB and its only enough for about 1 minute of execution

This matters because...

- Time travel is cool because you often find a problem far from its source
- This is even more important for embedded platforms since
  - real time/resource constrained systems often have intermittant timing dependant bugs
  - they interact with and run on buggy niche-market hardware (i.e., cache coherance bugs)

About the virtual in "virtual machine"

- If only x86 chips had a trace port...
- I want time travel in my Python debugger
- You can add a trace port to the VM
- But its probably easier to integrate the cleverness directly into the VM or an extension

Theory (meaningless drivel)

- Bugs happen when the developer's mental model of the world or the runtime program state does not match reality
- Debugging requires an understanding that only comes from realigning this mental model with reality
- No amount of testing will eliminate the need for debugging
- Sufficiently large amounts of testing are not cost effective versus debugging

What Insecticide does

- Record complete state of program execution so I can stop, go back in time, inspect global state, etc.
- Allow me to search for all blocks of execution state in which some condition is true so I can choose one and jump into it.
- Allow me to perform execution diffs: given two execution traces of the same function, show me how they differ

Object history

- For a given object, show me
  - when (program state), and
  - where (line of code)
- that it was
  - created
  - modified
  - added to some other object, etc.