Introduction to UndoDB

Undo™

Undo Overview

- Founded in 2005, with 25 man years development in debug technology

Founder : Greg Law, Ph.D , Acorn. Solarflare, Nexwave,etc

Founder: Julian Smith, Ph.D, Acorn, Vicon, RSA security etc

- Well funded with $1.25 million in 3 rounds to date: 2012/3/4
  - Backed financially by Cambridge Angels and Jaan Tallinn, co-founder Skype
  - Currently 9 full time staff
- Market Momentum
  - Revenue and headcount doubled in 2012 and 2013.
  - OEMs with ARM (DS5 v5.16) and Rogue Wave (Totalview)
  - Selected as Gartner Cool Vendor in Application Development for 2014.
  - Customers include : NASA, Lawrence Livermore Labs, Cadence Design Systems, Mentor Graphics, Synopsys Inc.

Undo™

Reversible Debugging for Real Code

- Reversible debuggers let you step backwards as well as forwards
- A holy-grail of tools research for decades. UndoDB is the first and only reversible debugger that is effective on real-world, complex software
    - Used today on many of the world's most complex software projects

        Scientific supercomputing clusters (NASA Ames, LLNL, etc)

        Enterprise (several fortune 500 customers)

- High performance, scalable, for compiled code on Linux

"I found the idea of the product amazing and a boon to my productivity... I have already been able to fix a deadlock that was driving me crazy for a week in only ten minutes!"

Undo™

UndoDB – reversible debugging that works

> ***"Reason back from the state of the crashed program to determine what could have caused this. Debugging involves backwards reasoning, like solving murder mysteries. Something impossible occurred, and the only solid information is that it really did occur. So we must think backwards from the result to discover the reasons."***

Kernighan & Pike's *The Practice of Programming*

- Regular debuggers give you pause/continue
- UndoDB gives you a rewind button, etc
- It's CCTV for you code

Undo™

Performance is key

- Patented checkpoint + replay approach
  - Exploit the natural determinism of computers
  - Reduces time/space overheads by many orders of magnitude
  - Highly optimised JIT binary translation
- Benchmark
  - Time to ‘gzip’ a 16MB* file

    (gzip is mostly CPU bound, with some IO)

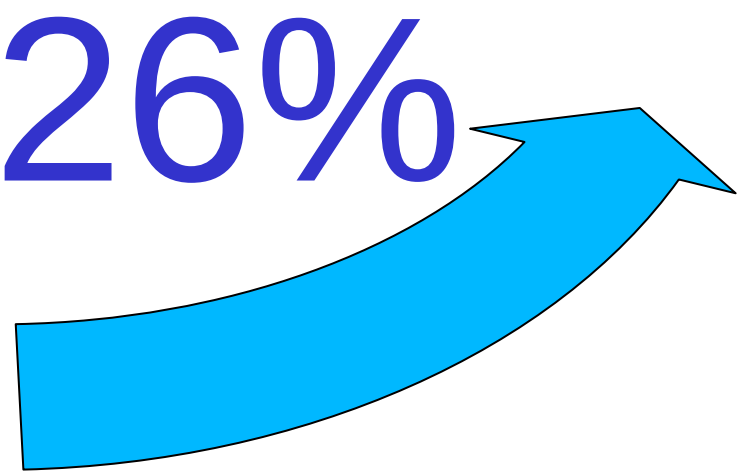
	Native	UndoDB	GDB “process record”
Time	1.49s	2.61s (**1.75x**)	21 hours (**>50,000x**)
Space	N/A	17.8MB	63GB

* GDB times extrapolated from 16K file

Undo™

Significant Productivity Gains

- Cambridge University study shows 26% productivity improvement
- 50% of development effort is debugging
  - How often does code work first time?
- $312bn cost in developers' wages alone
- Doesn't cover:
  - Reputational Gain
    - Fix customers' bugs more quickly
  - Opportunity
    - Time to Market



Undo™

Features and benefits

- Works on unmodified Linux x86 systems and programs
  - No kernel patches or modules, or special hardware requirements
  - No recompilation or special libraries for program being debugged
  - No restrictions on application to debug: threads, signals, shared memory, etc
- Greatest value is on the bugs where:
  - Effects manifest themselves a long time after underlying bug
  - Non-deterministic “once in a blue moon” failures
  - Large, unfamiliar codebases (“how did *that* happen?”)
- Full-featured debugger
  - Attach, watchpoints, etc.
  - Fits seamlessly into your existing workflow (gdb / Eclipse / Emacs, etc)

Undo™

COMING SOON...

AVX support (AVX2, AES)

Black-box Flight Recorder

- Undo library to be linked against your program
- Record in-the-field bugs
- C API to control record start, stop, dump

Undo™

Part II

UndoDB master class

Tips and tricks for better debugging

Undo™

Timeline

- “What time is it?”, or “Goto” a particular time:
  - **`undodb-get-n`**
  - **`undodb-goto-n`**
  - **`undodb-show-event-log-extent`**
- Precise control of timeline:
  - **`undodb-set-bookmark`** and
  - **`undodb-goto-bookmark`**
- Tip: Use these commands to binary chop

Undo™

The Event Log

- All “non-deterministic” inputs are recorded
  - System-calls
  - Thread-switches
  - Signals
  - Non-deterministic instructions
  - Shared memory reads (incl. DMA)
- List events
  - **`maint-undodb-show-events b,e`**
  - **`maint-undodb-show-eventstats b,e`**

Undo™

More on the Event Log

- Increase the event-log size dynamically
  - **`undodb-set-max-event-log-size`**
- Stop when the event log gets full (default)
  - **`undodb-event-log-mode straight`**
- Record forever with a circular event log
  - **`undodb-event-log-mode circular`**

Undo™

Performance tuning

- Start recording later:
  - Start with **`--undodb-defer-recording`**
  - Then when you're ready: **`undodb-enable-record`**
- Overview of performance metrics
  - **`maint-undodb-show-exec-summary`**
  - Make sure your instrumentation heap is big enough:

    `--undodb-instr-heapsize 512M`
  - Look out for executing code in writable memory

    approx 50x slower than executing for read-only memory

Undo™

Snapshots

- List existing snapshots

  `maint-undodb-show-snapshots`

- Beware memory consumption

  e.g. **`--undodb-snapshots 5`**

- Hint to create a snapshot soon

  `maint-undodb-snapshot`

  (will create a snapshot at next basic-block boundary)

Undo™

More tips and tricks

- No symbols/no backtrace?
  - **`reverse-stepi`**
  - **`undodb-goto-n -1000`**
- Set default command-line options

  `~/.undodb-gdb.defaults`
- Calling functions in debuggee/inferior
  - Side-effects are discarded after function is run
    - Function is run in a temporary process
  - Use our `infcall' gdb patch to make this bullet-proof

    e.g. **`break foo.c:42 if !memcmp( a, b, 32)`**

Undo™

More tips and tricks

- Jump to record mode
  - **`undodb-goto-record-mode`**
- Autotrace feature
  - Attach to an executable by name rather than pid
  - **`--undodb-autotrace myexe`**
- Problems?
  - Run with **`--undodb-asserts 1`**
  - If UndoDB crashes, send us the log files **`undodb_log.1234`**

Undo