Static and Dynamic Analysis at Ning

David Sklar - david@ning.com

ZendCon 2008

What?

- **Static** analysis: what can you learn from looking at the **source** code?
- **Dynamic** analysis: what can you learn from looking at the **running** code?
- **Both**: understand what your application **is doing** and **can do**

Why?

- Ning's "**Your Own Social Network**" application is ~206kloc and growing
- Developers on **4 continents**
- Enforce **code standards**, check deprecated **usage**, monitor **performance**, evaluate API **change impact**

Static #1: Text Munching

- **Regexes** or **pattern matching** to look for good/bad things in code
- Works nicely in **unit tests**

Static #2: Tokenizer

- More PHP-aware version of text munching
- Also useful in unit tests or standalone code analysis projects

Tokenizer Example

- Where does the code write to the filesystem?
- Step 1: Find function calls (with tokenizer)
- Step 2: Find which ones do writes (with human)

AST Detour

- http://pecl.php.net/parse_tree
- http://trac2.assembla.com/php-ast
- http://www.phpcompiler.org/

Static #3: Opcode Dump

- PHP has its own virtual machine
- **vld** extension shows you the opcodes that your PHP code is turned in to
- http://www.derickrethans.nl/vld.php
- In general fewer opcodes → better performance

Opcode Dump Example

- What's the difference between:

```
isset($fruit['peaches'])
```

and

```
array_key_exists('peaches', $fruit)
```

?

Dynamic #1: Profiling

- Xdebug provides bountiful profiling information
- http://www.xdebug.org/

Xdebug Configuration

- Enable xdebug
- Turn on profiling (or trigger)
- Set output directory and filename pattern

```
zend_extension=/full/path/to/xdebug.so
xdebug.profiler_enable=0
xdebug.profiler_enable_trigger=1
xdebug.profiler_output_dir=/tmp
xdebug.profiler_output_name=prof.%H.%R.%u.out
```

Files can be large...

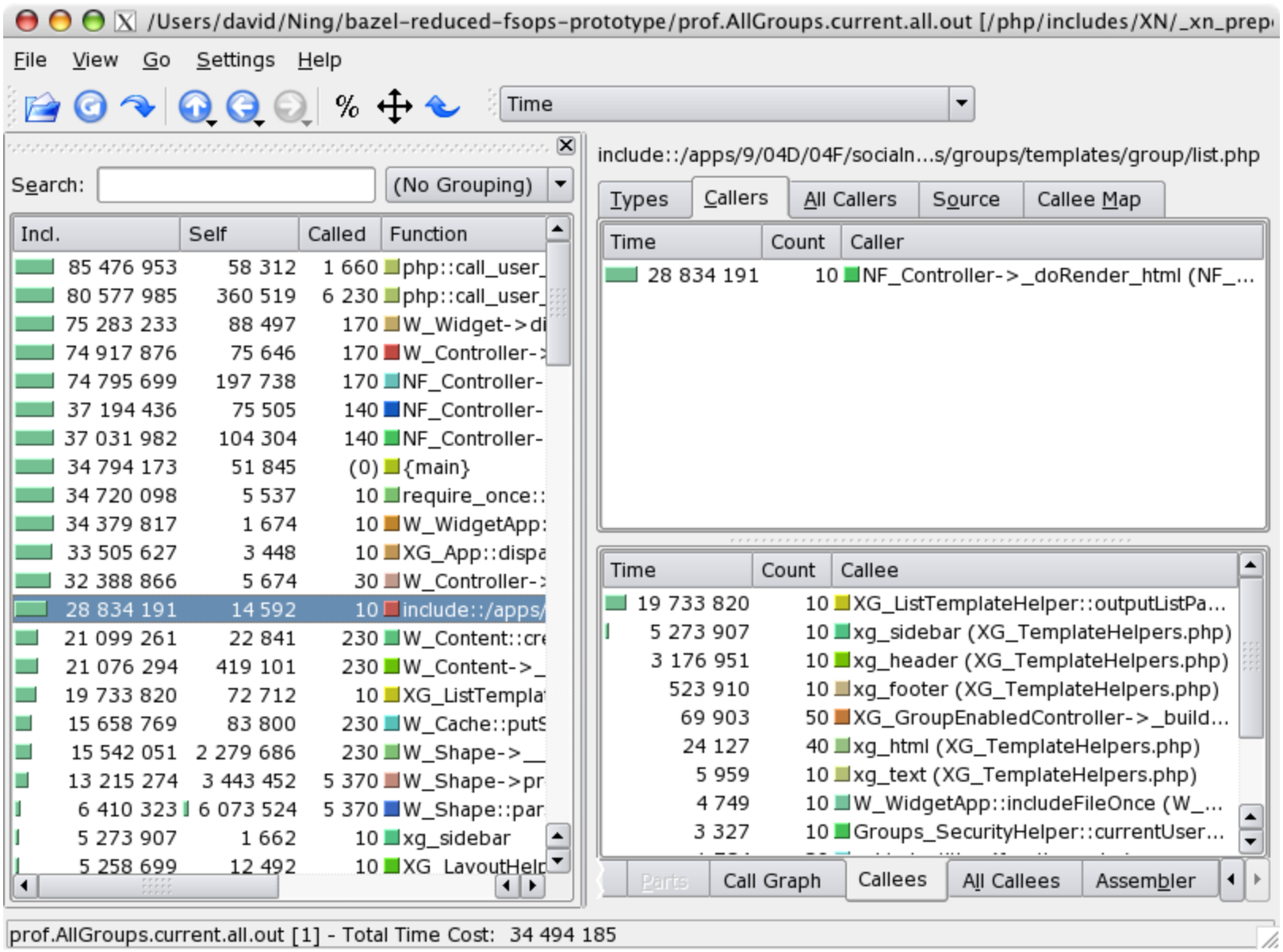
% ls -l

-rw-r--r-- 1 daemon daemon **4746558** May 7 17:52 prof.localhost._index_php_profiles_friend_list_XDEBUG_PROFILE=1.1210182719_781004.out

% wc –l p*

338909 prof.localhost._index_php_profiles_friend_list_XDEBUG_PROFILE=1.1210182719_781004.out

View with Kcachegrind...


/Users/david/Ning/bazel-reduced-fsops-prototype/prof.AllGroups.current.all.out [/php/includes/XN/_xn_prep
File View Go Settings Help
Time
Search:
(No Grouping)
Incl. Self Called Function
85 476 953 58 312 1 660 php::call_user_
80 577 985 360 519 6 230 php::call_user_
75 283 233 88 497 170 W_Widget->di
74 917 876 75 646 170 W_Controller-
74 795 699 197 738 170 NF_Controller-
37 194 436 75 505 140 NF_Controller-
37 031 982 104 304 140 NF_Controller-
34 794 173 51 845 (0) {main}
34 720 098 5 537 10 require_once::
34 379 817 1 674 10 W_WidgetApp:
33 505 627 3 448 10 XG_App::dispa
32 388 866 5 674 30 W_Controller-
28 834 191 14 592 10 include::/apps/
21 099 261 22 841 230 W_Content::cre
21 076 294 419 101 230 W_Content->_
19 733 820 72 712 10 XG_ListTempla
15 658 769 83 800 230 W_Cache::putS
15 542 051 2 279 686 230 W_Shape->__
13 215 274 3 443 452 5 370 W_Shape->pr
6 410 323 6 073 524 5 370 W_Shape::par
5 273 907 1 662 10 xg_sidebar
5 258 699 12 492 10 XG_LayoutHelp
include::/apps/9/04D/04F/socialn...s/groups/templates/group/list.php
Types Callers All Callers Source Callee Map
Time Count Caller
28 834 191 10 NF_Controller->_doRender_html (NF_...
Time Count Callee
19 733 820 10 XG_ListTemplateHelper::outputListPa...
5 273 907 10 xg_sidebar (XG_TemplateHelpers.php)
3 176 951 10 xg_header (XG_TemplateHelpers.php)
523 910 10 xg_footer (XG_TemplateHelpers.php)
69 903 50 XG_GroupEnabledController->_build...
24 127 40 xg_html (XG_TemplateHelpers.php)
5 959 10 xg_text (XG_TemplateHelpers.php)
4 749 10 W_WidgetApp::includeFileOnce (W_...
3 327 10 Groups_SecurityHelper::currentUser...
Parts Call Graph Callees All Callees Assembler
prof.AllGroups.current.all.out [1] - Total Time Cost: 34 494 185

Viewing Tools

- Kcachegrind: Linux, OS X, Windows, Cygwin (http://kcachegrind.sourceforge.net/)
- WinCacheGrind: Windows (http://sourceforge.net/projects/wincachegrind/)
- Webgrind: All (http://code.google.com/p/webgrind/)
- ct_annotate: All (@see valgrind)

Dynamic #2: Function Traces

- Xdebug again!
- Each function entry/exit recorded

Easy to parse/analyze

- One line on entering a stack frame, one on leaving. Fields are tab-delimited:

Line Type	Field									
	1	2	3	4	5	6	7	8	9	10
Entry	Frame Level	Frame Number	0 = Entry	Time	Memory	Function	0 = internal, 1 = user-defined	Included file	file	line
Exit			1 = Exit							

Function Trace Example

- Stack traces wherever a file is included

Dynamic #3: strace/truss

- Strace on linux, truss on solaris
- (ktrace on os x)

strace example

- Parsing entire strace output to get syscall census
- Files can (familiar refrain) be large

truss example

- Use fancy truss options (library names, function name prefixes) to find when certain PHP functions are invoked.

ltrace

- ltrace on linux is a “library call tracer”
- Performance overhead significant, hasn’t been as useful

Dynamic #4: DTrace

- DTrace's capabilities are Mighty and Numerous
- No special setup (other than installing DTrace + provider)
- **No* production performance impact**
- PHP Provider: http://pecl.php.net/dtrace
- Solaris, Leopard

DTrace Example #1

- Flow between PHP functions and system functions

DTrace Example #2

- What's everything* that happens when PHP does something simple?
- (*everything = PHP scripts, PHP internals, libc, syscalls, kernel)

systemtap

- systemtap is a Linux tool that allows for user-written in-kernel profiling scripts similar to DTrace

Summary

- **Static**: Pattern matching, tokenizer, opcode analysis
- **Dynamic**: Profiling, function traces, strace, truss, dtrace
- **Slides, etc.**: http://www.sklar.com/blog/

Come Work at Ning !

- Build the software that powers **> 460,000** social networks
- PHP, REST, JSON, XML, APIs, C, Ruby, Python, Apache, and friends
- Work in Palo Alto, CA (or not)
- Visit us in the Exhibit Hall
- http://jobs.ning.com – david@ning.com