Flame Graphs for

*Online* Performance Profiling

😊 agentzh@gmail.com 😊

Yichun Zhang (agentzh)

2013.06.01
Flame Graphs is a kind of visualization for analyzing how time or some other resource is distributed among all the code paths.
Flame Graph for My Day

- bug fix
- process
- verify issues
- emails
- testing
- hacking
- dinner
- cooking

my day
Colors in Flame Graphs do not matter; they are picked up by random.
Three Observation Samples
Reorder & Merge All the Samples into a whole
Box widths are equal to the number of the corresponding samples; sample count is proportional to time.
For Flame Graphs in the *software* world, *code paths* are defined as *backtraces*. 
Perl/Lua/PHP code

userspace C

OS kernel C code

The Software Stack
IO::Select::select
IO::Socket::connect
IO::Socket::INET::connect
IO::Socket::INET::configure
IO::Socket::new
IO::Socket::INET::new
Test::Nginx::Socket::send_request
Test::Nginx::Socket::run_test_helper
Test::Nginx::Util::run_test
Test::Nginx::Util::run_tests
0x3880ef2877 : socket+0x7/0x30 [/usr/lib64/libc-2.15.so]
0x537445 : Perl_pp_socket+0x233/0x376 [/opt/perl/bin/perl]
0x4d24ab : Perl_runops_standard+0x17/0x40 [/opt/perl/bin/perl]
0x43d8cc : S_run_body+0x1a2/0x1ac [/opt/perl/bin/perl]
0x43d363 : perl_run+0xae/0x475 [/opt/perl/bin/perl]
0x41e34c : main+0xc0/0x146 [/opt/perl/bin/perl]
0x3880e21735 : __libc_start_main+0xf5/0x1c0 [/usr/lib64/libc-2.15.so]
0x41e1a9 : _start+0x29/0x2c [/opt/perl/bin/perl]
0xffffffff81632f81 : _raw_spin_unlock_irqrestore+0x11/0x20 [kernel]
0xffffffff8108e98e : __wake_up_sync_key+0x5e/0x80 [kernel]
0xffffffff8119d340 : pipe_write+0x3c0/0x540 [kernel]
0xffffffff81194737 : do_sync_write+0xa7/0xe0 [kernel]
0xffffffff81194dec : vfs_write+0xac/0x180 [kernel]
0xffffffff81195132 : sys_write+0x52/0xa0 [kernel]
0xffffffff8163baa7 : tracesys+0xdd/0xe2 [kernel]
❤️ We *gather* various kinds of backtraces on Linux via **systemtap**.
At every Linux system tick (controlled by `CONFIG_HZ`, 1000 on my side), if the current process on CPU is the process we are interested in, sample a backtrace, and aggregate it immediately.
The **DWARF** debug information is the **map** for the cold **binary world**.
$ gcc -g ...

$ sh Configure -Doptimize=-g -des -Dprefix=/opt/perl

$ yum install xxx-debuginfo

$ apt-get install xxx-dbgs
Simple wrapper tools based on systemtap are ready for everyday use.
Generating *Perl*-land Flame Graphs with just 2 commands.
Real-time analyzing and diagnosing tools for perl 5 based on SystemTap — Read more

branch: master

perl-systemtap-toolkit /

docs: minor copyright tweaks.

agentzh authored 3 days ago
# assuming the perl process is of pid 1302.

$ pl-sample-bt -p 1302 -t 5 > a.bt

WARNING: Sampling 1302 (/opt/perl/bin/perl) for Perl-land backtraces...

Please wait for 5 seconds.
Test::Nginx::Socket::send_request
Test::Nginx::Socket::run_test_helper
Test::Nginx::Util::run_test
Test::Nginx::Util::run_tests
  58
Test::Nginx::Util::error_log_data
Test::Nginx::Socket::check_error_log
Test::Nginx::Socket::run_test_helper
Test::Nginx::Util::run_test
Test::Nginx::Util::run_tests
  54
...

FlameGraph

**USAGE message**

**brendangregg** authored 17 days ago
$ stackcollapse-stap.pl a.bt | flamegraph.pl - > a.svg
Function: Test::Builder::todo (8 samples, 0.80%)
Function: all (997 samples, 100%)
I just ported perl 5's `pp_caller` opcode's implementation over to the `systemtap` scripting language.
Generating user-space C-land Flame Graphs for the same perl process with another 2 commands.
Real-time analyzing and diagnosing tools for Nginx based on SystemTap — Read more

git@github.com:agentzh/nginx-systemtap-toolkit.git

branch: master

nginx-systemtap-toolkit

docs: mentioned my Perl Systemtap Toolkit.

agentzh authored 4 days ago
# assuming the perl process is of pid 1302.

$ ngx-sample-bt -p 1302 -t 5 -u > a.bt

WARNING: Tracing 1302 (/opt/perl/bin/perl) in user-space only...
WARNING: Time's up. Quitting now...(it may take a while)
$ stackcollapse-stap.pl a.bt | flamegraph.pl - > a.svg
http://agentzh.org/misc/flamegraph/perl-vm-test-nginx.svg
Function: Perl_pp_aassign (8 samples, 3.40%)
Perl_pp_method_named (19 samples, 8.09%)
We can profile on the Perl 5 *opcode* level via the userspace C-land flamegraphs.
We may make clever use of the high-level Perl language constructs to eliminate specific hot Perl 5 opcodes.
We may help Perl 5 porters to find hot places within the perl VM that can be further optimized.
Actually we are already doing both for LuaJIT at CloudFlare.
lj_BC_CAT --> switch to string arrays + concat
lj_BC_FNEW --> reduce creating anonymous functions
pcre_compile2 --> cache the compiled regexes
lua_yield --> LuaJIT *internal* optimizations by Mike Pall
1. lj_tab_newkey
0. 0x7f8bbfacfd29
lj_tab_newkey --> new LuaJIT primitive table.new() for pre-allocation
Generating *kernel*-space Flame Graphs for the *same* perl process with 2 similar commands.
# assuming the perl process is of pid 1302.

$ ngx-sample-bt -p 1302 -t 5 -k > a.bt

WARNING: Tracing 1302 (/opt/perl/bin/perl) in kernel-space only...

WARNING: Time's up. Quitting now...(it may take a while)
$ stackcollapse-stap.pl a.bt | flamegraph.pl - > a.svg
http://agentzh.org/misc/flamegraph/kernel-test-nginx.svg
off-CPU time Flame Graphs
File I/O Flame Graphs
❤ Special thanks go to Brendan Gregg for inventing Flame Graphs.
Any questions?