The Realities of DTrace on FreeBSD

Jonathan Anderson, Brian Kidney, **George Neville-Neil**,
Arun Thomas, Robert Watson

gnn@freebsd.org

BSDTW
2017年11月11日
臺灣臺北

Overview

History

Motivation

Recent DTrace Improvements

How People use DTrace

Future Improvements

Obligatory History

Developed at Sun for Solaris before 2005

Ported to FreeBSD in 2008

Ported to Mac OS for 10.5

Maintained separately with some cross patching

Motivations

Improve the state of tracing on large systems

Expand tracing into the Distributed Systems

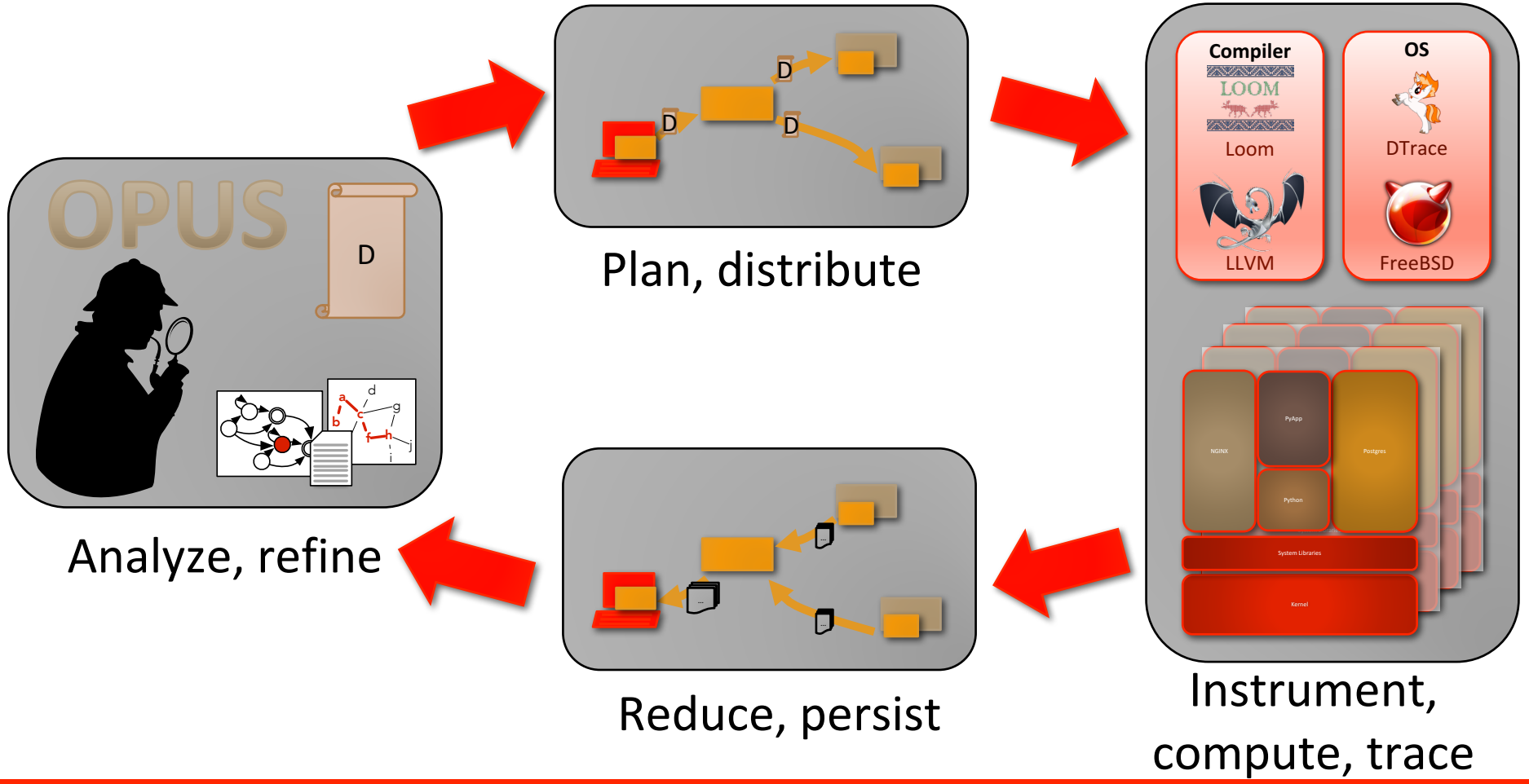
Use tracing to teach about complex systems

Make it production ready

Do it in open source

Crisis Meeting

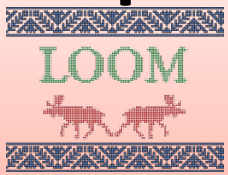
CADETS: Causal, Adaptive, Distributed and Efficient Tracing System



Ground-Up Local Instrumentation

Loom specification-driven program instrumentation

Compiler



Loom

LLVM

LLVM IR fat binaries support JIT (re-) instrumentation

OS

DTrace

FreeBSD

DTrace scriptable full-system dynamic tracing framework

FreeBSD open-source OS extended for transparency

How We Use DTrace

Leverage DTrace for Distributed Instrumentation

Meaning?

- DTrace is now always on
- DTrace protections can used against us
- Some improvements necessary

https://github.com/cadets/freebsd

DTrace Design Principles

No overhead when not in use

Never panic the kernel

Protect the kernel at all costs

D is like C but safe

DTrace, Resources and Tuning

DTrace Built in 2005

A simpler time

With smaller memories

And slower CPUs

Your Grandparents Computers

Key feature highlights

- 64-bit Chip Multithreading (CMT) UltraSPARC® IV technology
- Scales up to 8 x 1.2-GHz UltraSPARC IV CPUs with 16 MB L2 cache per processor
- Up to 16 simultaneous compute threads with up to 64 GB memory
- Solaris™ 8, Solaris 9, and Solaris 10 Operating System
- 9.6-GB/second Sun™ Fireplane interconnect
- N+1 hot-swap power supplies/ hot-pluggable disks
- Sun™ Remote System Control for secure server management

Where Enterprise Computing Meets Entry-Level Pricing

More users and services are dramatically increasing the demands placed on today's IT infrastructures and systems. Sun meets this challenge with the Sun Fire™ V890 server. The Sun Fire V890 server features the new UltraSPARC IV 64-bit processor with Chip Multithreading (CMT) processor technology, as well as Solaris Operating System, the industry's most robust, secure, and popular UNIX® operating system.

With up to eight UltraSPARC IV CMT processors executing 16 concurrent threads, and up to 64 GB of memory, the Sun Fire V890 server delivers extreme levels of throughput for your most demanding departmental and enterprise applications. The Sun Fire V890 platform is an ideal system for an extensive range of applications, including Application Serving, Business Processing, Database, Collaboration, High-Performance Technical Computing (HPTC), and Application Development.

With a potential of nearly two terabytes of internal storage and standard networking support, the Sun Fire V890 server is designed to complement IT operations at prices below traditional data center servers. The 9.6-GB/sec. system bus, integrated I/O adapters, and nine PCI slots help ensure a highly scalable, well-balanced system for Application Serving, I/O-intensive, and compute-intensive workloads.

Your Grandparents Computer

Highest Data Bandwidth Solution
High-End SAS / NAS Storage Solution

Key Features

1. Dual Intel® 64-bit Xeon™ Support, up to 3.60 GHz, 800 MHz FSB
2. Intel® E7520 (Lindenhurst) Chipset
3. Up to 16GB DDRII 400 SDRAM
4. Intel® 82546GB dual-port Gigabit Ethernet Controller
5. Adaptec AIC-9410 8-Port Serial Attached SCSI (SAS) Controller
6. 2x SATA Ports via Intel ICH5R SATA Controller
7. 2 (x8) PCI-Express,
   1x 64-bit 133MHz PCI-X,
   1x 64-bit 100MHz PCI-X
8. IPMI 2.0 Socket

Specifications

Running out of steam

dtrace: 2179050 drops on CPU 0
dtrace: 2113052 drops on CPU 0
dtrace: 3104101 drops on CPU 0
….

"DTrace is broken!"

DTrace Tuning

bufsize

- Defaults to 4M and was severely limited on FreeBSD
- Increase if you are having too many drops

switchrate

- Defaults to 1Hz
- Increase if you have drops

dynvarsize

- Defaults to 1M
- Increase if you have variable drops
- `dtrace: 103 dynamic variable drops`

Recent DTrace Improvements

Machine-Readable Output

New Providers

- audit
- mac and mac_framwork
- opencrypto
- sctp
- xbb

Performance Analysis

Documenting the Internals

- Not just what, but also how and why

Machine-Readable Output

```
# dtrace –n 'syscall::write:entry'

dtrace: description 'syscall::write:entry' matched 2 probes
CPU     ID                    FUNCTION:NAME

  0  59780                      write:entry

  0  59780                      write:entry
```

```
# dtrace -O json -n 'syscall::write:entry'
dtrace: description 'syscall::write:entry' matched 2 probes
CPU     ID                    FUNCTION:NAME
{
  "probe": {
    "timestamp": 3594774042481656,
    "cpu": 1,
    "id": 59780,
    "func": "write",
    "name": "entry"
  }
```

D Language Improvements

C-like language that supports all C operators

Structured like awk

Supports thread and clause local variables

Subroutines to handle common tasks

copyoutmbuf

Allows for reading chained mbufs in D

Important for BSD derived network stacks

```
dtrace -n ':::tcp_input:entry { tracemem(copyoutmbuf(*args[0], 64), 64); }'
dtrace: description ':::tcp_input:entry ' matched 1 probe

CPU     ID                    FUNCTION:NAME
  3  34345                 tcp_input:entry
               0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  0123456789abcdef
         0: 45 10 00 34 46 f4 40 00 40 06 70 0a c0 a8 01 01  E..4F.@.@.p.....
        10: c0 a8 01 64 3d 3c 00 16 70 7d fa 57 c2 cc f1 4b  ...d=<..p}.W...K
        20: 80 10 04 10 b5 a5 00 00 01 01 08 0a 3a c8 04 ae  ............:...
        30: 94 3c 09 5f 00 00 00 00 00 00 00 00 00 00 00 00  .<._............
```

D Language Improvements

if statements

- D has ternary operator

```
hexval = (c >= '0' && c <= '9') ? c - '0' : (c >= 'a' &&
         c <= 'z') ? c + 10 - 'a' : c + 10 - 'A';
```

- if statement improves readability
- Syntactic sugar imported from Solaris

For want of an if()

```
vdev_queue_pending_remove:entry {
  if (stringof(args[1]->io_spa->spa_name) == $$1){
    if (args[1]->io_type == ZIO_TYPE_READ) {
      @bytes_read = sum(args[1]->io_size);
    }
            else if (args[1]->io_type == ZIO_TYPE_WRITE
                                          && args[1]->io_bookmark.zb_level != 2) {
      @bytes_written = sum(args[1]->io_size);
    }
  }
}
```

* Example by Matthew Ahrens

A Spoonful of Syntactic Sugar

```
dtrace:::ERROR{ self->_XD_error = 0x1; }

::vdev_queue_pending_remove:entry{ self->_XD_error = 0x0; }

::vdev_queue_pending_remove:entry /!self->_XD_error/

{ this->_XD_condition1 = 0x1 && stringof(args[1]->io_spa->spa_name) == $$1; }

::vdev_queue_pending_remove:entry /!self->_XD_error/

{ this->_XD_condition2 = this->_XD_condition1 && args[1]->io_type ==
ZIO_TYPE_READ; }

::vdev_queue_pending_remove:entry /(!self->_XD_error) && this->_XD_condition2/

{ @bytes_read = sum(args[1]->io_size); }

::vdev_queue_pending_remove:entry /!self->_XD_error/

{ this->_XD_condition3 = this->_XD_condition1 && !this->_XD_condition2; }

::vdev_queue_pending_remove:entry /!self->_XD_error/

{ this->_XD_condition4 = this->_XD_condition3 && args[1]->io_type ==
ZIO_TYPE_WRITE

  && args[1]->io_bookmark.zb_level != 2; }

::vdev_queue_pending_remove:entry /!self->_XD_error && this->_XD_condition4/

{ @bytes_written = sum(args[1]->io_size); }
```

* Example by Matthew Ahrens

Audit Provider

Subsystem for logging security related events

Government Common Criteria security standards

Optional component of FreeBSD since 2004

Audit Provider

What is a provider?

- DTrace code that collects together a set of trace points

What does the provider get us?

- Access to audit framework data in DTrace…
- … with filtering and statistics through D.

DTrace Performance

DTrace shouldn't degrade performance

- Drops Records
- Kernel can kill tracing under high load

Solutions

- Our monitoring cycle
- Buffer Sizes now configurable with sysctl
  - Update your memory parameters from 2005
- Improve the D compiler
- JIT
- Leverage LLVM

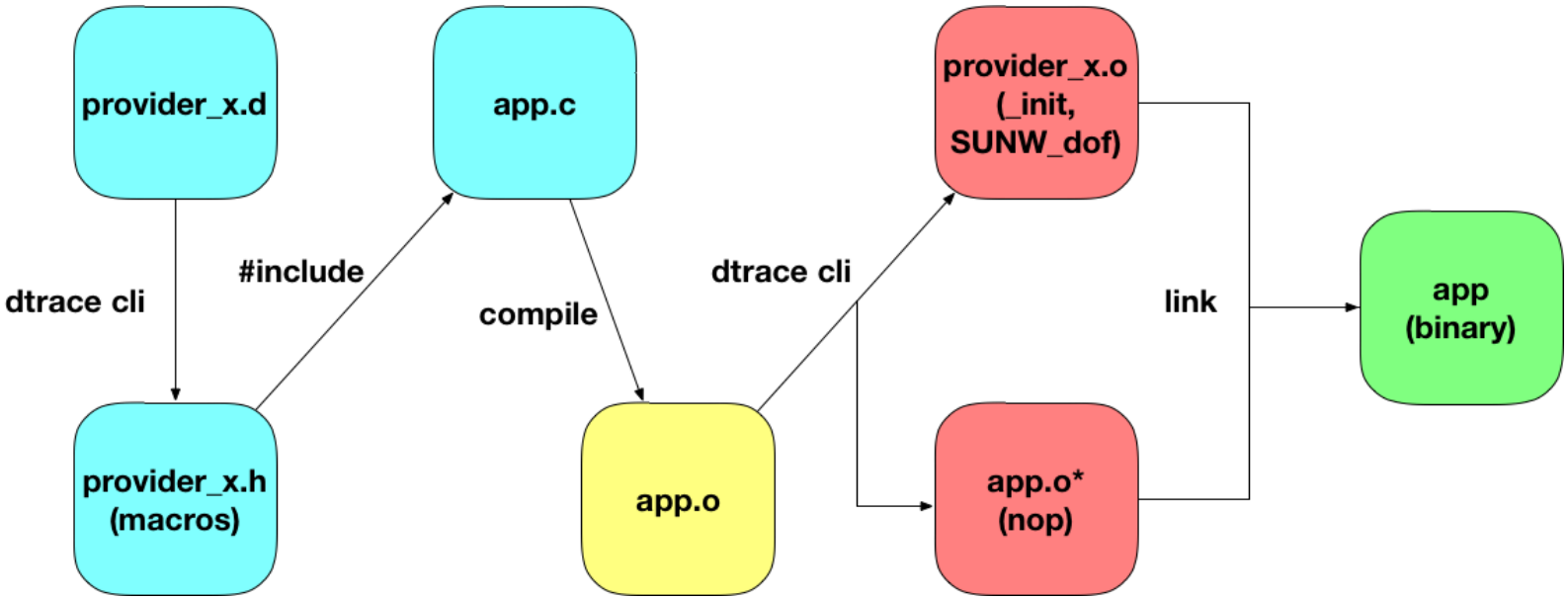
Aside – Loom

Loom is a instrumentation framework

- Based on LLVM toolchain
- Weaves instrumentation into LLVM IR
- Instrumentation defined in Policy files
- Instrumentation can be done at any time
  - As long as LLVM IR is available

We want to use Loom for DTrace probes in Userspace

Userland Statically Defined Tracing (USDT)



USDT Performance

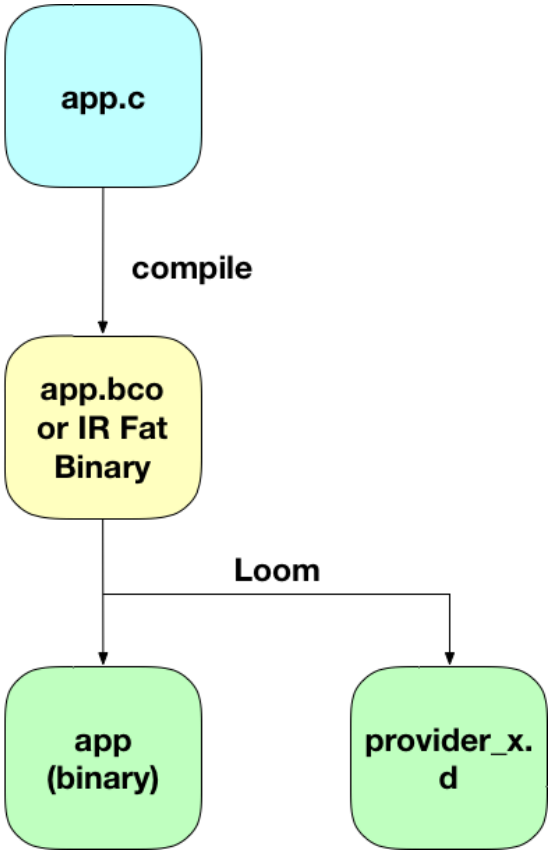
Probes disabled when not tracing

- Probe site replaced with NOP/function pointer
- Near zero overhead – theoretically

Problems

- DTrace tool modifies binaries
- Doesn’t play well with Make
- Makes heavy use of relocations

Loom Base Userland Tracing



Dynamic Userland Tracing

Very Early Stages of Development!

- Prototype system call (dt_probe)
- Instrumentation via Loom
- No change to binary when no instrumentation

To be complete

- Performance/Overhead testing
- Provider Generation

DTrace is not the **Only One**

eBPF

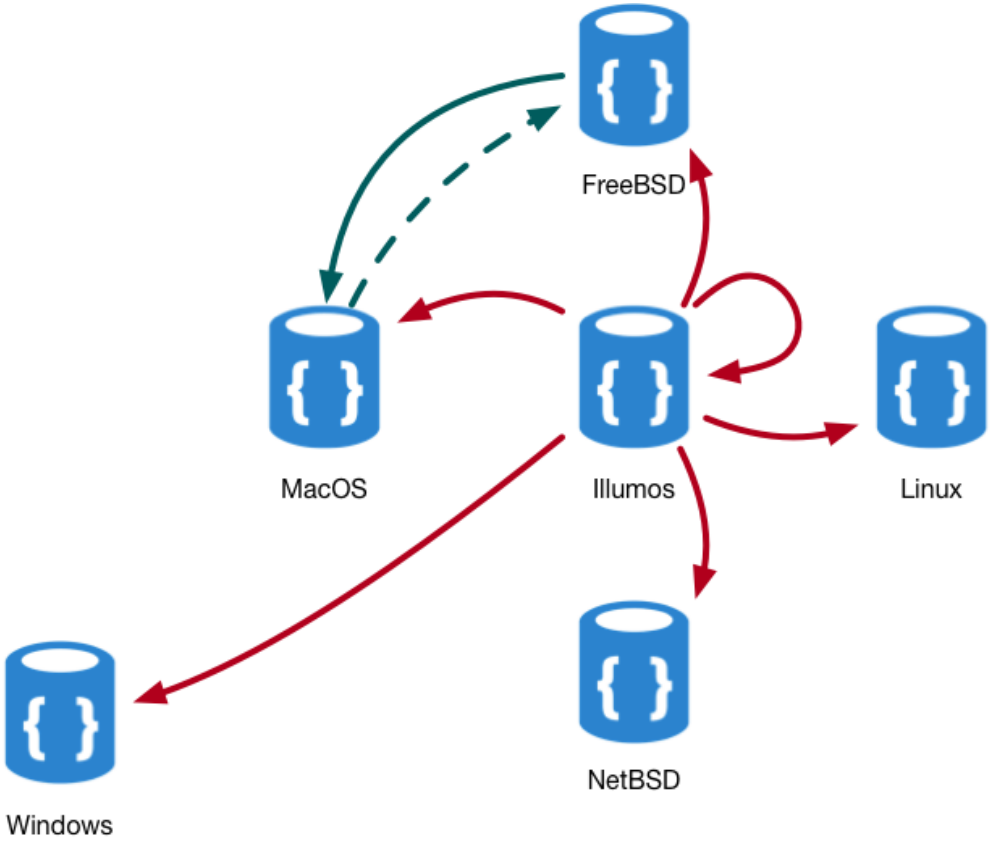
At the lowest level far too primitive

bcc – A C-like front end

ply – Python front end

Has feature parity with DTrace – Brendan Gregg 2017

DTrace Source Flow



OpenDTrace

Cross Platform

Highly Portable

RFD Process

github.com/orgs/opendtrace

OpenDTrace Specification

DTrace Specification of DIF, DOF and CTF in progress

Better testing of Framework

Support new execution substrates (JIT)

Make it easier to make future extension

Allow for clean room re-implementation

OpenDTrace Futures

Basic Blocks

Bounded Loops

Modules

Higher Performance

Improved Test Suite

More OS Ports

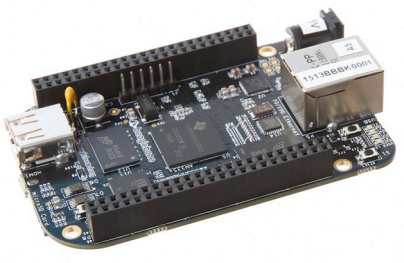
Broad Architecture Support

Finer Grained Libraries

Usable from other languages

- Python, Rust, Go

OpenDTrace on a Spectrum



Distributed DTrace

Applying OpenDTrace

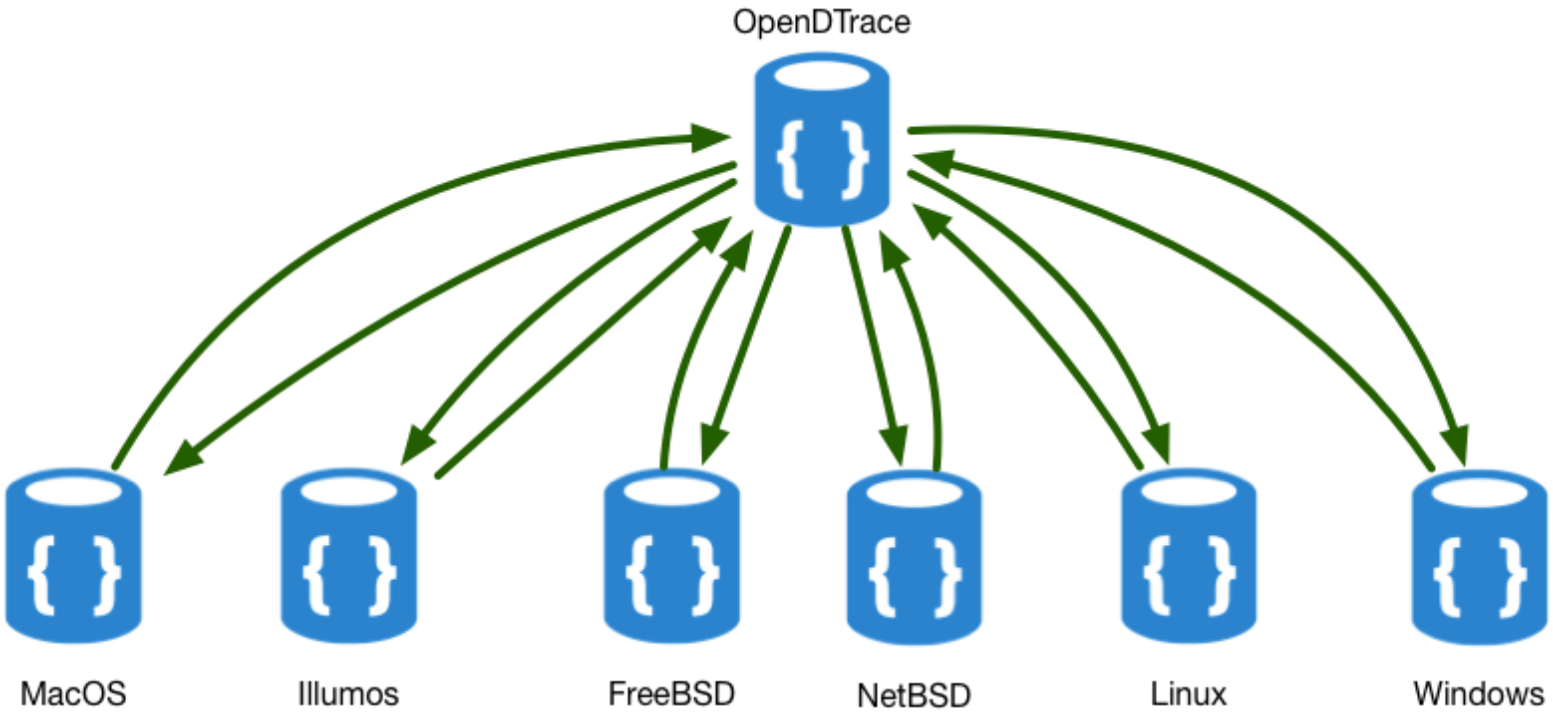
Enhanced kernel trace points on FreeBSD (and others)

- IPSEC
- Network Link Layer
- GEOM/CAM
- Drivers

User Space Tools

- Schedgraph
- Lockgraphing
- Performance of various subsystems
- Flamegraph all the things

OpenDTrace



How you can help

Look at the opendtrace organization on github

Check out the documentation and source

Send pull requests

Please help us find a **new** logo!