



# DTrace: Life after Solaris 10

**Jon Haslam**

Solaris Kernel Engineering

Sun Microsystems

# Introduction

- Solaris 10 RR'd on 31/01/2005
- DTrace integrated into s10\_43 on 03/09/2003
- Features/Fixes delivered into Nevada
- Backported into Updates
- Documentation slightly behind <cough>

# Appearing tonight...

- What's new since S10 GA (not exhaustive list!)
- What's (hopefully/possibly/maybe) arriving soon.
- Virtually everything in here ripped off from Bryan, Mike and Adam.
- Release that feature came in on indicated by (s10release-Nevada)

# Multiple aggregation printa()

## Release: 08/07-30

- multiple aggregations in single printa()
- aggregations must have same type signature
- output is effectively joined by key
- 0 printed when no value present for a key
- default behavior is to sort by first aggregation value (ties broken by key order)

# Multiple aggregation printa()

```
/* multagg.d */
syscall::write:entry
{
    @wbytes[execname, pid] = sum(arg2);
}

syscall::read:entry
{
    @rbytes[execname, pid] = sum(arg2);
}

END
{
    normalize(@rbytes, 1024);
    normalize(@wbytes, 1024);

    printf("%20s %10s %10s %10s\n", "PROGRAM", "PID",
        "READS", "WRITES");
    printa("%20s %10d %10@d %10@d\n", @rbytes, @wbytes);
}
```

# Multiple aggregation printa()

```
# dtrace -q -s ./multagg.d
```

```
^C
```

PROGRAM	PID	READS	WRITES
dtrace	101605	0	0
nautilus	101606	0	0
battstat-applet-	100854	0	15
gnome-settings-d	100781	0	0
gnome-session	100728	0	0
dsdm	100712	6	0
gnome-terminal	101342	8	15
xscreensaver	100946	9	0
soffice.bin	101325	10	48
init	1	32	0
gnome-panel	100792	39	4
gconfd-2	100767	47	5
nautilus	100794	265	170
wnck-applet	100822	714	262
metacity	100789	726	263
gedit	101607	3986	64
Xorg	100535	12374	0

# Aggregation sorting options

## Release: 08/07-30

- aggregations sorted by value by default
- options allow change of behaviour
  - > aggsortkey - sort by key order, ties broken by value
  - > aggsortrev - reverse sort
  - > aggsortpos - position of the aggregation to use as sort primary sort key with multiple aggs
  - > aggsortkeypos - position of key to use as primary sort key when with multiple aggs
- Use the above in combination

# Aggregation sorting options

```
/* aggsort.d */
syscall::read:entry
{
    @avg[execname, pid] = avg(arg2);
    @max[execname, pid] = max(arg2);
    @min[execname, pid] = min(arg2);
    @cnt[execname, pid] = count();
}

END
{
    printf("%20s %10s %10s %10s %10s %10s\n", "EXECNAME", "PID",
        "COUNT", "MIN", "MAX", "AVG");
    printa("%20s %10d %@10d %@10d %@10d %@10d\n", @cnt, @min,
        @max, @avg);
}
```

# Aggregation sorting options

```
# dtrace -q -s ./aggsort.d
```

```
^C
```

EXECNAME	PID	COUNT	MIN	MAX	AVG
battstat-applet-	100981	2	32	32	32
gnome-settings-d	100853	3	32	64	53
soffice.bin	101382	10	32	32	32
dsdm	100708	10	32	160	54
xscreensaver	101082	14	32	32	32
gnome-panel	100896	24	12	168	51
firefox-bin	101363	31	1	1024	35
gnome-terminal	101029	40	32	4096	163
nautilus	100906	119	32	480	48
wnck-applet	100961	161	8	128	32
metacity	100893	447	4	96	33
Xorg	100534	926	64	5104	3263

**Sort by value of first aggregation (default)**

# aggregation sorting options

```
# dtrace -q -s ./aggsort.d -x aggsortrev
```

```
^C
```

EXECNAME	PID	COUNT	MIN	MAX	AVG
Xorg	100534	875	64	5104	3433
metacity	100893	434	4	96	33
wnck-applet	100961	145	8	64	32
xscreensaver	101082	71	4	32	28
gnome-terminal	101029	54	32	4096	125
nautilus	100906	51	32	608	74
firefox-bin	101363	23	1	1	1
dsgm	100708	19	32	32	32
gnome-panel	100896	18	12	168	51
gnome-settings-d	100853	7	32	32	32
soffice.bin	101382	6	32	32	32

**Reverse sort using value of first aggregation**

# Aggregation sorting options

```
# dtrace -q -s ./aggsort.d -x aggsortkey -x aggsortkeypos=1 -x aggsortrev
^C
```

EXECNAME	PID	COUNT	MIN	MAX	AVG
soffice.bin	101382	525	4	1440	33
firefox-bin	101363	29	1	1024	36
thunderbird-bin	101337	2	1	1024	512
xscreensaver	101082	11	32	64	34
gnome-terminal	101029	27	32	4096	220
wnck-applet	100961	161	8	96	32
nautilus	100906	79	32	320	40
gnome-panel	100896	26	12	168	49
metacity	100893	196	4	128	34
gnome-settings-d	100853	4	32	64	40
dsdm	100708	23	32	128	40
Xorg	100534	885	64	4940	3688

**Reverse sort by key in second position**

# Aggregation sorting options

```
# dtrace -q -s ./aggsort.d -x aggsortpos=2 -x aggsortrev
```

```
^C
```

EXECNAME	PID	COUNT	MIN	MAX	AVG
Xorg	100534	1655	64	5104	3756
gnome-terminal	101029	137	32	4096	69
wnck-applet	100961	453	4	1152	34
xscreensaver	101082	23	32	1024	115
nautilus	100906	43	8	736	69
soffice.bin	101382	637	4	288	30
gnome-panel	100896	122	8	168	39
metacity	100893	421	4	128	34
notification-area	100983	2	64	64	64
mixer_applet2	100985	2	64	64	64
gnome-settings-d	100853	7	32	32	32
dsdm	100708	103	4	32	31
thunderbird-bin	101337	7	4	32	24
firefox-bin	101363	39	1	32	5

**Reverse sorted by value of third aggregation**

# (u)mod/(u)func/(u)sym

Release: 08/07-23

- Profiling often requires post-processing when using %a/%A to print arg0/arg1 symbolically
- Samples in format [module]'[func]+[offset]
- Want to first get high level view and then drill down
- (u)mod(%pc) - module name
- (u)func(%pc) - function name
- (u)sym(%pc) - symbol name

# (u)mod/(u)func/(u)sym

```
#pragma D option aggsortkey

cpc:::dtlbn-u-5000
{
    @[execname, umod(arg1)] = count();
}

END
{
    printf("%20s %30s %10s\n", "EXECUTABLE", "MODULE",
        "COUNT");
    printa("%20s %30A %10@d\n", @);
}
```

**Example uses prototype cpc provider to show TLB misses on a global basis broken down by module.**

# (u)mod/(u)func/(u)sym

```
# ./tlbmissbymod.d
```

```
^C
```

EXECUTABLE	MODULE	COUNT
Xorg	Xorg	17
Xorg	libramdac.so	1
Xorg	libfb.so	1
Xorg	radeon_drv.so	1
Xorg	libc.so.1	3
battstat-applet-	libglib-2.0.so.0.1200.4	1
battstat-applet-	libgobject-2.0.so.0.1200.4	1
gconfd-2	libORBit-2.so.0.1.0	1
gconfd-2	libgconf-2.so.4.1.0	1
metacity	libpangocairo-1.0.so.0.1400.7	1
metacity	libgdk_pixbuf-2.0.so.0.1000.6	1
metacity	libgobject-2.0.so.0.1200.4	1
metacity	libglib-2.0.so.0.1200.4	2
metacity	libgdk-x11-2.0.so.0.1000.6	1
metacity	libc.so.1	2

# ucaller variable

## Release: 08/07-23

```
# dtrace -n 'pid$target::malloc:entry{@[ufunc(ucaller)] = count();}'
-p 101384
dtrace: description 'pid$target::malloc:entry' matched 2 probes
^C
```

sax.uno.so`0xf97b2fe7	2
sax.uno.so`0xf97b30e7	2
sax.uno.so`0xf97b3104	2
sax.uno.so`0xf97b44ba	2
sax.uno.so`0xf97b44cf	2
sax.uno.so`0xf97b8a18	4
libX11.so.4`_XAllocScratch	5
libX11.so.4`miRegionOp	6
libX11.so.4`XQueryTree	8
libX11.so.4`XGetWindowProperty	13
libX11.so.4`XSetClassHint	13
libX11.so.4`XGetImage	179
libuno_sal.so.3`osl_createMutex	322
libX11.so.4`XCreateRegion	844
libX11.so.4`XCreateGC	959
libc.so.1`calloc	1074
libglib-2.0.so.0.1200.4`standard_malloc	3533
libCrun.so.1`__1c2n6FI_pv_	28668

# String parsing subroutines

## Release: 01/06-15

- The usual suspects
  - > strtok() - tokenise a string
  - > strstr() - find first occurrence of str2 in str1
  - > strchr() - find first occurrence of char 'c' in str
  - > strrchr() - find last occurrence of char 'c' in str
  - > substr() - return substr of str starting at pos p for length l
  - > index() - return position of first char 'c' in str
  - > rindex() - return position of last char 'c' in str

**(Don't forget about strjoin() and strlen(!))**

# String example

```

pid$target:libc:putenv:entry
{
    this->str = copyinstr(arg0);
    this->var = strtok(this->str, "=");
    this->val = substr(this->str, index(this->str, "=") + 1);
    printf("env[%s] = %s\n", this->var, this->val);
}

```

```

# dtrace -s ./putenv.d -c "date -u"
dtrace: script './putenv.d' matched 1 probe
Mon Sep 17 18:48:37 GMT 2007
CPU    ID                FUNCTION:NAME
  0  74554                putenv:entry env[TZ] = GMT0

```

# The `fds[]` variable

## Release: 01/06-16

- array of `fileinfo_t`'s indexed by integer (fd)
- inlines expanded to accommodate associative arrays for this.
- Definition in `/usr/lib/dtrace/io.d`
- `fileinfo_t` gets new `fi_oflags` member

# The fds[] variable

```
#pragma D option quiet
```

```
syscall::write:entry
```

```
/fds[arg0].fi_oflags & O_APPEND/
```

```
{
```

```
    printf("%s appending file %s at offset %d\n",  
           execname, fds[arg0].fi_pathname, fds[0].fi_offset);
```

```
}
```

```
# ./fds.d
```

```
ksh appending file /.sh_history at offset 349345
```

```
ksh appending file /.sh_history at offset 349378
```

# dtrace -v

Release: 01/06-11

- '-v' option added to dtrace(1M) to display probe arguments and stability attributes
- use in combination with '-l' or '-e'
- use with mdb(1) for examining kernel type info
- use docs for D type info (or /usr/lib/dtrace)

# dtrace -v

```
# dtrace -lv -n fbt::ufs_write:entry
      ID      PROVIDER      MODULE
FUNCTION NAME
27473      fbt      ufs
ufs_write entry
```

## Probe Description Attributes

```
Identifier Names: Private
Data Semantics:   Private
Dependency Class: Unknown
```

## Argument Attributes

```
Identifier Names: Private
Data Semantics:   Private
Dependency Class: ISA
```

## Argument Types

```
args[0]: struct vnode *
args[1]: struct uio *
args[2]: int
args[3]: cred_t *
args[4]: caller_context_t *
```

# USDT headergen

## Release: 08/07-34

- '-h' option to dtrace(1M)
- generates header file with externs and macros
- PROVIDER\_PROBE() macros instead of generic DTRACE\_PROBE()
- USDT works with C++ now
- Check out Adam's blog

# USDT is-enabled

## Release: 08/07-38

- USDT overhead is very low
- Setting up arguments can be very expensive though
- PROVIDER\_ENABLED() macros introduced
- Is-Enabled slightly higher probe effect but use if possible

# USDT is-enabled

## Ruby example from Adam's blog:

```
rb_call(...  
{  
    ...  
    RUBY_ENTRY(rb_class2name(klass), rb_id2name(method));  
    ...  
    RUBY_RETURN(rb_class2name(klass), rb_id2name(method));  
    ...  
}
```

```
rb_call(...  
{  
    ...  
    if (RUBY_ENTRY_ENABLED())  
        RUBY_ENTRY(rb_class2name(klass), rb_id2name(method));  
    ...  
    if (RUBY_RETURN_ENABLED())  
        RUBY_RETURN(rb_class2name(klass), rb_id2name(method));  
    ...  
}
```

# inet\_ntoa\*()/inet\_ntop() (-66)

- `inet_ntoa()` - converts IPv4 address into human readable string
- `inet_ntop()` - convert both IPv4 and IPv6 addresses into human readable string
- `inet_ntoa6()` - IPv6 version of `inet_ntoa()`

# inet\_ntoa\*()/inet\_ntop()

```

fbt::ip_tcp_input:entry
{
    @[inet_ntoa(&args[1]->ipha_src), inet_ntoa(&args[1]->ipha_dst)] =
    count();
}
END
{
    printa("%30s -> %-30s %@d\n", @);
}

```

```
# ./inet_ntoa.d
```

```
^C
```

```

212.58.227.137 -> 129.156.38.34    3
212.58.226.20  -> 129.156.38.34    59
212.58.226.8   -> 129.156.38.34    99

```

```
# dig -x 212.58.226.8 +short
news1b14.thdo.bbc.co.uk.
```

# Misc bits (with some demo's)

- `ustackdepth` variable (11/06-20)
- `ucaller` variable (08/07-23)
- `%k` `printf/printa` modifier for printing stacks (08/07-23)
- `hton[s,l,ll]`, `ntoh[s,l,ll]`, `htohl` (/50)
- Test suite integration (08/07-48)
- DTrace support in Zones (08/07-37)
- `lockstat/plockstat -x` option (01/06-9)
- DTrace JNI binding (08/07-35)

# Available Providers

- fsinfo (u2/38) - Filesystem information provider
- sysevent (/58) - System event channel provider
- Xserver (/??) - Xorg
- python (/65) - John Levon's python beauty
- iscsi (/69) - Guess what?
- xpv (/75) - x86 paravirtualized provider
- tcl (external) - <http://wiki.tcl.tk/DTrace>

# (Not Officially) Available Providers

- network - see DTrace community pages
- cpc - CPU performance counter access
- ibs - AMD Instruction Based Sampling
- ruby
- mpi - Message Passing Interface
- Mozilla/Javascript - see DTrace community pages

# Information Sources

- DTrace OpenSolaris discussion forum
  - > Ask questions – get involved
  - > Change Log
- Blogs
- Source
- [www.solarisinternals.com](http://www.solarisinternals.com)
- Solaris Performance and Tools (Prentice Hall 2006 ISBN 0-13-156819-1)
  - > General DTrace use and methods



# DTrace: Life after Solaris 10

**Jon Haslam**

[jonathan.haslam@sun.com](mailto:jonathan.haslam@sun.com)

[blogs.sun.com/jonh](http://blogs.sun.com/jonh)