

25. Marec 2010

# Sun Training Day

Technopol, kongresové centrum  
Kutlíkova 17, 852 50 Bratislava



## Sun Training Day 2010 sekce Solaris – DTrace

Martin Červený  
M.Cervený@computer.org

**Základy použití nástroje DTrace**

**DTrace pro operační systém**

**DTrace pro aplikace**

# Základy použití nástroje DTrace

# Co je DTrace

## základy

- DTrace je instrumentace pro dynamickou analýzu problémů v systému a aplikacích v celém rozsahu
- DTrace je jako hudební nástroj, musíte cvičit, cvičit ...
- "Swiss Army knife"



# Kompletní analýza

základy

dříve

- interpretované jazyky (java, javascript)
  - dbx
- přeložené programy (/usr/bin/\*)
  - truss -ua.out, prof, gprof
- knihovny (/usr/lib/\*)
  - sotruss, apptrace
- systémová volání (syscalls)
  - truss
- jádro systému
  - alokace paměti, plánovač procesů, souborové systémy, ovladače ...
  - mdb, prex, tnfs, lockstat, kstat
- hardware
  - kstat

# Funkčnost DTrace

základy

nástroje

- **vložení sondy -> aktivace -> akce**
  - popsáno interpretačním jazykem D
- nástroje
  - **dtrace(1m)**
    - nadstavby
      - chime, dtrace toolkit, dexplorer, netbeans plugin, sunstudio dlight
  - knihovna libdtrace(3lib)
    - lockstat(1m)/plockstat(1m)
    - intrstat(1m)
    - powertop(1m)
  - vložené explicitní sondy do programů
    - interpretetry, Xserver, databáze ...
- platformy
  - Solaris 10, OpenSolaris a odvozené distribuce
  - FreeBSD 7.1, MacOS 10.5

# Sonda

základy

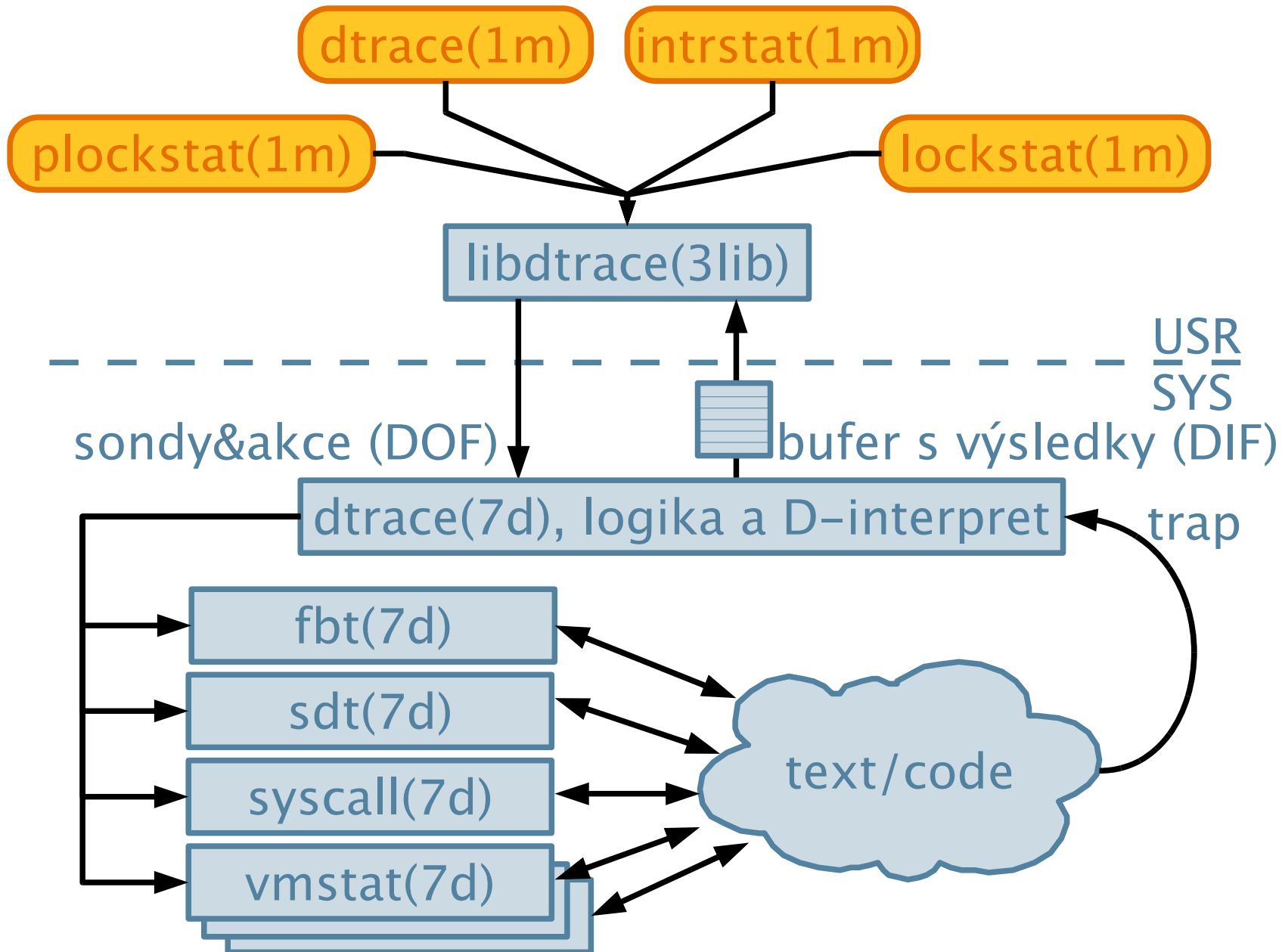
sonda

- dynamické vložení přerušení do funkce kernelu/programu
  - zajímavé pro produkční systémy, žádný vliv, pokud není sonda aktivní
- cíle
  - pro kernel
  - uživatelské programy
  - interpretry – java, javascript, PHP, mysql ...
  - aplikace – apache, Xserver
- „provider“ (vkladatel) sond
  - ~ 60000 sond generických
    - fbt
  - ~ 3000 sond specifických
    - syscall, fsinfo, sysinfo, vminfo, lockstat, sched, io, proc, sdt, profile-\*, tick-\*, sysevent, mib, ip, dtrace, iscsi ...
  - sondy aplikační
    - pid\*, hotspot\*, javascript\*, mysql\*, Xserver\*, ... *"usdt"*

# Systemová implementace sondy

základy

sonda



# Dynamické vložení sondy

základy

sonda

## fbt bez sondy

```
ufs_mount: pushq  %rbp
            movq   %rsp, %rbp
            subq   $0x20, %rsp
            movq   %rdi, -0x8(%rbp)
            [...]
            popq   %r12
            popq   %rbx
            leave
            ret
```

## sonda vložena

```
int  $0x3
movq  %rsp, %rbp
subq  $0x20, %rsp
movq  %rdi, -0x8(%rbp)
[...]
popq  %r12
popq  %rbx
leave
int  $0x3
```

## sdt bez sondy

```
read_status: [...]
            movl  %r8d, %edi
            nop
            nop
            nop
            nop
            movl  $-0x1, %ebx
```

## sonda vložena

```
[...]
movl  %r8d, %edi
nop
lock nop
nop
nop
movl  $-0x1, %ebx
```

# Provider

základy

sonda

- dtrace – vnitřní sondy DTrace
- lockstat – sondy pro zámky a zamykání
- profile/tick – pravidelně spouštěné sondy (časovač)
- fbt – sondy na vstupu/výstupu všech funkcí kernelu
- syscall – sondy na vstupu/výstupu systémových volání
- sdt – “staticky definované sondy” explicitně zakompilované v kódu
- sysinfo – sondy statistik kernelu pro mpstat a sysinfo
- vminfo – sondy statistik kernelu pro virtuální paměť
- proc – sondy vytváření a zánik procesů/LWP
- sched – sondy plánovače CPU
- io – sondy pro sledování diskových IO
- mib – SNMP MIB sondy systémového managementu
- fpuinfo – sondy na sledování zpracování v pohyblivé řádové čáře (SPARC)
- pid – sondy sledování funkcí a instrukcí v procesech
- plockstat – sondy pro zamykání a synchronizaci v procesech
- fasttrap – přímé vložení sond

# Akce

základy

akce

- interpretační jazyk D
  - podobné awk(1) a podle ANSI-C – matematika, logika a funkce
  - možnost i C preprocesoru
  - interpret v kernelu
  - nejsou smyčky, rekurze, větvení ...
- funkce *(některé z funkcí jsou zatím jen v OpenSolarisu)*
  - tisk `-trace()`, `tracemem()`, `printf()`, `stack()`, `printa()` ...
  - agregační – `count()`, `quantize()`, `lquantize()`, `max()`, `avg()`, `min()` ... `clear()`, `trunc()`
  - řídicí – `exit()`, `return()`
  - práce s řetězcí – `strjoin()`, `strlen()`, `strtok()`, `strstr()`, `strchr()`, `strrchr()`, `substr()`, `index()`...
  - pro data programů – `copyin()`, `copyinstr()`, `ustack()`, `inet_ntoa*`, `inet_ntop()`, `ntoh/hton*`
  - destruktivní – `stop()`, `copyout()`, `copyoutstr()`, `system()`, `breakpoint()`, `panic()`, `chill()`

# Akce

základy

akce

- proměnné dtrace
  - `probeprov`, `probemod`, `probefunc`, `probename`
  - `arg0`, `arg1`, ..., `args[#]`, `uregs[#]`, `fds[#]`
  - `execname`, `timestamp`, `vtimestamp`, `zoneid` ...
  - `pid`, `ppid`, `tid`, `uid`, `gid`, `cpu`, `cwd`, `errno` ...
  - `$pid`, `$target`, `$1`, `$2` ...
  - libovolná proměnná v kernelu (```)
- uživatelsky definované proměnné
  - platnost
    - globální proměnná - `X`
    - lokální proměnná - `this->X`
    - proměnná pro vlákno - `self->X`
  - typy
    - skalární proměnné včetně řetězců
    - asociativní pole - `poleX[indexy]`
    - `@agregaceX[indexy]=agr_funkce(argumenty)`
- spekulativní výstup
  - `speculation()`, `speculate()`, `commit()`, `discard()`

# Zápis sondy a akce

základy

akce

- popis sondy
  - `provider:module:function:name`
  - možnost použití „\*“ a prázdné definice
  - speciální „BEGIN“, „END“ a „ERROR“
- úplný popis sondy a akce

```
provider:module:function:name  
/logická podmínka (predicate)/  
{ příkaz akce; ... }
```
- využití s `dtrace(1m)`
  - privilegia
    - `dtrace_kernel, dtrace_proc, dtrace_user`
  - **oneliners**
    - `dtrace -n 'S/P/{A} S/P/{A} ...'`
  - **dtrace skript**
    - `#!/usr/sbin/dtrace -s`
  - speciality
    - anonymní, postmortem dtrace

# Základní demonstrace

základy

demo

- provider dtrace "BEGIN/END/ERROR"
  - `BEGIN {printf("hello world!\n"); exit(0);}`
- agregace (min,max,sum,avg...) a asociativní pole
  - `syscall::read:entry { @amin=min(arg2); @amax=max(arg2); @aavg=avg(arg2); @acount[execname]=count(); }`
- využití podmíněné sondy
  - `syscall::read:entry /execname=="Xorg"/ { @filedescr[arg0]=sum(arg2); }`
- sledování práce jádra nebo aplikace (-F)
  - `syscall::read:entry /execname=="cat"/ { self->ctu=1; } syscall::read:return { self->ctu=0; } fbt:: /self->ctu/ {}`

# Základní demonstrace

základy

demo

- práce s daty
  - `syscall::open:entry { printf("%s(%i): %s\n", execname, pid, copyinstr(arg0)); }`
- časování a vzorkování pomocí "profile/tick"
  - `profile-997`  
`{ @[execname,ustack()]=count(); }`  
`tick-10s { exit(0); }`
- sledování aplikací pomocí "pid"
  - `pid$target:::entry {} pid$target:::return {}`
- destruktivní práce dtrace (`-w`)
  - `syscall::uname:entry { self->ptr=arg0; }`  
`syscall::uname:return`  
`{ copyoutstr("OracleOS",self->ptr,257); }`

# Demonstrace nastaveb

základy

demo

- DTraceToolkit
  - sada předpřipravených a zdokumentovaných skriptů s případným vhodným formátováním
  - <http://hub.opensolaris.org/bin/view/Community+Group+dtrace/dtrac toolkit>
- typy DTraceToolkit skriptů ~ 230
  - Apps – pro aplikace jako Apache, NFS
  - Cpu – měření aktivit CPU
  - Disk – analýza aktivity I/O
  - Kernel – sledování jádra
  - Locks – analýza programových zámků
  - Mem – analýza paměťového podsystemu
  - Net – aktivity na síti a TCP/IP stack
  - Proc – aplikační procesy
  - System – přehled o systému
  - User – aktivity uživatelů podle UID
  - Zones – sledování zón
- dexplorer

# Demonstrace nadstaveb

základy

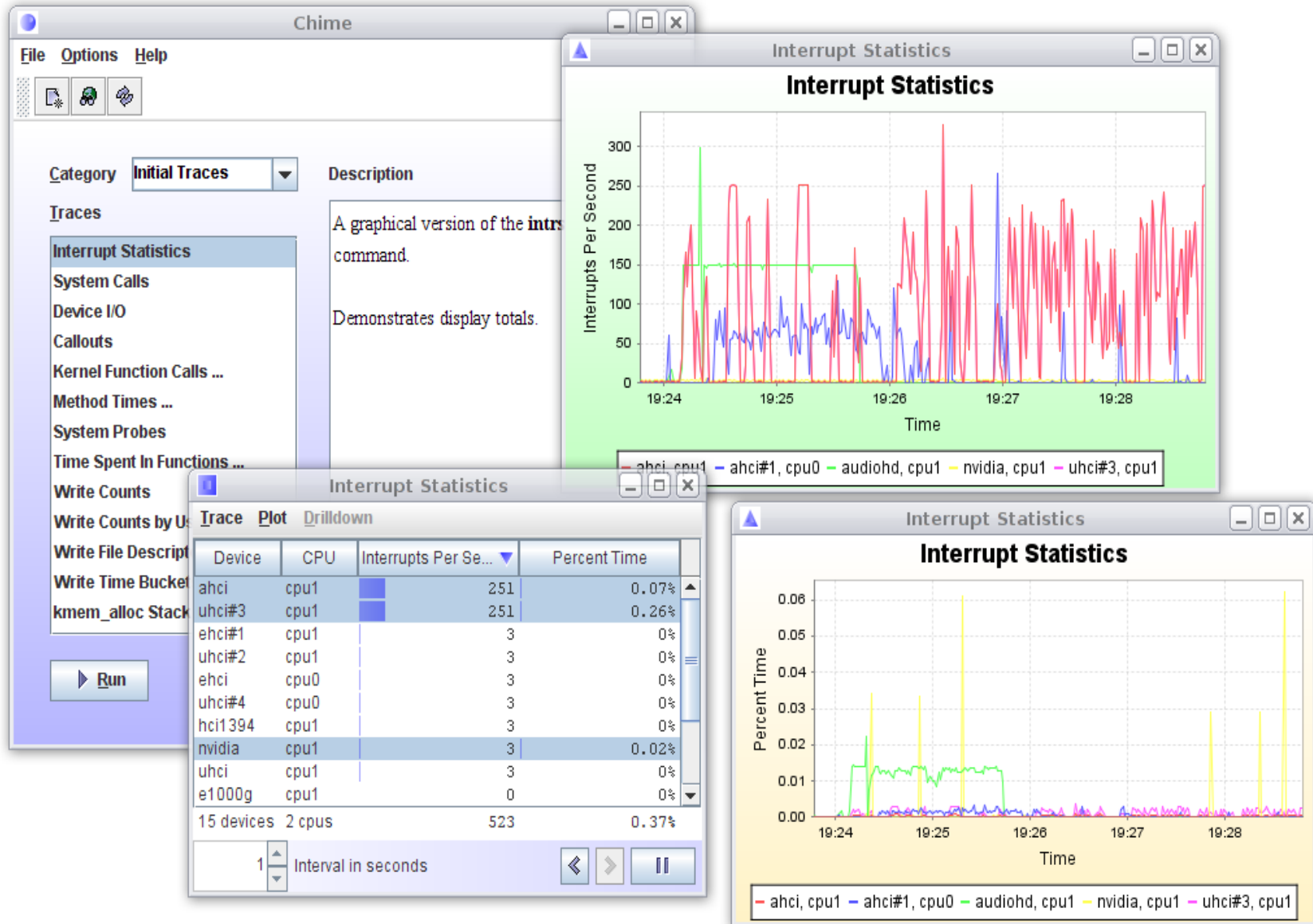
demo

- chime
  - vizualizace pomocí javy (JFreeChart)
  - <http://hub.opensolaris.org/bin/view/Project+dtrace-chime/>
- DTraceTazTool
  - UFS
- Netbeans plugin
- SunStudio 11 DLight

# Chime

základy

demo



# SunStudio 11 DLight

základy

demo

The screenshot displays the SunStudio 11 DLight interface. The main window is titled "Sun Studio" and contains a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Versjonering, Tools, Advanced, Window, Help) and a toolbar with various icons. The interface is divided into several panels:

- Projects:** Shows "DLight Instruments" and "dtmf.c".
- Services:** Includes "Clock profiler", "File System Activity", "Read/Write Monitor", and "Heap monitor".
- DLight Event Details:** Displays the "Microstate accounting Monitor" with a table of microstates and their counts.
- Timeline:** A horizontal timeline at the top of the main area, with a scale from 1 to 15. Below it are four stacked bar charts representing different performance metrics over time.
- Dbx Console:** Shows the output of the DLight tool, including the command used to run the profiler and the resulting microstate data.

Microstate	Count
Time stamp	10.575994297
Stopped	0
Asleep for any other reason	0
Asleep in user text page fault	0
Asleep in user data page fault	0
Aleep in kernel page fault	0
Asleep on semaphore (waiting for IO?)	0
Asleep on cond. variable (waiting for IO?)	0
Asleep waiting for system-mode synchronization	0
Asleep waiting for user-mode synchronization	0
Waiting for CPU (latency)	920
Running in other trap	0
Running in sys call or page fault	28
Running in user mode	50

```
DTMF (Build, Profile) x DLight (/tmp/dlight_martin/session_2559.er) at localhost x
Running "/opt/sunstudio12.1/bin/collect -D clock -D fops -D msa -D io -o /tmp/dligh
Creating experiment database /tmp/dlight_martin/session_2559.er ...
Clock profiling is on.
--11111--11111--1111--11111---11111-8888--11111--6666--0000--22222--00000-1111--0000
```

# DTrace pro operační systém

# Analýza operačního systému

system

- postupně zpřesňující analýza
  - 1) globální statistiky (\*stat, kstat(1m))
  - 2) agregace vybrané statistiky
    - podle zóny, aplikace, uživatele, zdroje
  - 3) výběr podrobnějších statistik v zúženém rozsahu
- někdy slepé uličky, myslete jako Edison
- vzorkování naslepo (provider profile)
- analýza stavu systému
  - datové struktury jádra operačního systému
  - kombinace s mdb(1m)
- zpracování a interpretace výsledků
  - návazné formátovací skripty
  - začlenění do C programu (libdtrace(3lib))

# CPU zatížení v procesu (user time)

system

%usr

- problém (mpstat)

CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
0	10	0	0	681	253	1774	196	61	0	0	11377	14	2	0	84
1	28	0	0	630	508	255	89	61	5	0	5421	93	2	0	5

- hledání

→ `profile-997 { @[pid] = count(); }`

→ `profile-997 /pid==$target/ { @[ustack()] = count(); } END{ trunc(@,5); }`

→ `pid$target:::entry { self->ts[self->stack++] = timestamp; }`

```
pid$target:::return /self->ts[self->stack-1]/ {  
  this->elapsed = timestamp-self->ts[--self->stack];  
  @count[probefunc]=count();  
  @elapsed[probefunc]=quantize(this->elapsed);  
  self->ts[self->stack]=0; }
```

→ `[dtrace -F] pid$target:::entry{  
 pid$target:::return{ }`

- řešení

# CPU zatížení v systému (system time)

system

%sys

- problém (mpstat)

CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
0	109	0	155	646	183	879	117	139	14	0	3041	9	65	0	26
1	101	0	112	1192	679	957	141	148	19	0	3502	12	67	0	21

- hledání

→ `profile-997 {@[execname,uid,pid,tid]= count();}`

→ `./DTraceToolkit-0.99/Bin/topsysproc`

→ `fbt::: { @[probemod,profunc]=count(); }`

- řešení

# Mnoho systémových volání (syscall)

system

syscall

- problém (mpstat)

CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
0	10	0	198	938	293	1279	50	113	14	0	15513	10	7	0	83
1	32	0	225	1274	806	1305	49	116	4	0	13428	12	9	0	79

- hledání

```
→ syscall:::entry { printf("%s(%i):\t%s\n",  
    execname, pid, probefunc); }  
→ syscall:::entry { @[execname]=count(); }  
→ syscall:::entry /execname=="reader"/  
    { @[probecfunc]=count(); }  
→ syscall:::entry /execname=="reader"/ { self-  
    >ts= timestamp; }  
    syscall:::return /self->ts/  
    { @spent[probecfunc]= sum(timestamp-self->ts);  
    self->ts=0; }  
→ syscall:::read*:entry /execname=="reader"/  
    { @[arg2] = count(); } END { printa("velikost:  
    %d\t\tpocet volani:%@ d",@); }  
→ syscall:::read*:entry /execname=="reader"/  
    { @[ustack()] = count(); }
```

- řešení

# Časté přepínání procesů (context switch)

system

procesy

- problém (mpstat)

CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
0	35	0	507	2391	326	45693	1165	11566	65	0	33641	15	18	0	67
1	1	0	463	3427	909	45217	1702	11636	42	0	29882	16	21	0	63

- hledání

→ `sched:::off-cpu { @[execname]=count(); }`

→ `sched:::off-cpu /execname == "threads"/`  
`{ @[tid]=count(); }`

→ `sched:::off-cpu /execname == "threads"/`  
`{ @[ustack()]=count(); }`  
`END { trunc(@,5); }`

- řešení

# Zámky v mnoha-vláknových aplikacích (mutex,...)

system

vlákna

- problém (mpstat)

CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
0	177	0	3	13530	211	122826	12897	1453	60525	0	124634	70	29	0	1
1	30	0	8	14017	800	124933	12854	1494	61834	0	134841	70	30	0	0

- hledání

→ `plockstat ./threads_malloc`

- řešení

- atomické instrukce `atomic_ops(3C)`
- optimalizované i knihoven `libumem/libmtdalloc`

# Pracovní paměť procesu (anonymous memory)

system

paměť

- problém (vmstat)

memory		page					executable			anonymous			filesystem		
swap	free	re	mf	fr	de	sr	epi	epo	epf	api	apo	apf	fpi	fpo	fpf
1012744	479840	8560	10489	8	0	0	40	0	0	0	0	0	24	8	8
999920	473680	689	1173	133	0	0	267	0	0	0	0	0	510	133	133
994976	462896	1018	1818	166	0	0	8	0	0	0	0	0	626	166	166

- hledání

```
→ vminfo::: / execname!="dtrace" / { @[execname,  
probefunc, probename]=count(); }  
tick-10sec { printf("%-16s %-16s %-16s %-  
16s\n", "EXECNAME", "FUNCTION", "NAME", "COUNT");  
printa("%-16s %-16s %-16s %-@16d\n", @);  
clear(@); printf("\n\n"); }
```

- řešení

# Zatížení diskového I/O

system

disky

- problém (iostat)

```
extended device statistics
r/s    w/s    kr/s    kw/s  wait  actv  wsvc_t  asvc_t  %w  %b  device
0.0    0.0    0.0    0.0   0.0   0.0   0.0    0.0    0   0   c0t0d0
0.0   177.6    0.0  22733.6  0.0  10.0   0.0    56.3   0  100  c3t0d0
```

- hledání

```
→ io:::start { printf("%s(%d): %d\n", execname,
    pid, args[0]->b_bcount); }
→ ./DTraceToolkit-0.99/Bin/iotop
→ ./DTraceToolkit-0.99/Bin/rwtop
→ ./DTraceToolkit-0.99/Bin/iosnoop
```

- řešení

# Hledání chyb

system

chyby

- chyby v systémových voláních *errno*
  - `./DTraceToolkit-0.99/Bin/errinfo`
  - `syscall::return /arg0==-1 && errno!=11 && execname!="dtrace" / { printf("%s: %s %d\n", execname, probefunc, errno); }`
- hledání podrobností o chybách
  - `syscall::connect:entry { self->sockaddr=copyin(arg1,arg2); }  
syscall::connect:return /arg0==-1/  
{ printf("%s: %s:%d\n", execname,  
inet_ntoa((ipaddr_t *)(&((struct sockaddr_in *)self->sockaddr)->sin_addr)), ntohs(((struct sockaddr_in *)self->sockaddr)->sin_port)); }`

# Systemová bezpečnost (Host Intrusion Detection System)

system

HIDS

- reverse engineering
  - sledování volání
  - graf volání
  - `profile-4999hz { @[ustack()] = count() }`  
`tick-5s { exit(0); }`
- stack/heap overflow/underflow
- sledování práce se soubory
  - `fsinfo::fop_open: / ((vnode_t *)arg0)->v_path`  
`&& strstr(((vnode_t*)arg0)->v_path, "/dev") ==`  
`((vnode_t*)arg0)->v_path /`  
`{ printf("%s", stringof(((vnode_t*)arg0)-`  
`>v_path)); }`
- sledování shellů
  - `./DTraceToolkit-0.99/Bin/shellsnoop`
- podezřelé spouštění programů
  - `proc:::exec /execname == "QuickTime Player" &&`  
`args[0] == "/bin/sh" / { printf("QT on MacOS has`  
`been p0wned!\n") }`

# DTrace pro aplikace

# Analýza programů

## aplikace

- standardní provider
  - pid\*, plockstat\*
  - sledování prakticky omezeno na entry/return
  - hlavní program (a.out) a knihovny
  - nutná znalost vnitřní architektury programu
- User-Level Statically Defined Tracing (USDT)
  - mysql\*, postgresql\*, Xserver\*, apache\*, php\*, python\*, (perl\*), ruby\*, javascript\*, hotspot\*, hotspot\_jni\*, (sh\*), (gtk\*/glib\*) ...
  - doplněné sondy "DTRACE\_PROBE\*()"
  - zakompilované a popsané v ELF (využívá linker)
  - "exportuje" důležité události
    - start/stop, begin/end, entry/exit ...

# USDT

aplikace

USDT

- vložení sondy do kódu (usdt.c)

```
#include <sys/sdt.h>
int main() {
    int data=123;
    DTRACE_PROBE1(usdt,begin,data);
    DTRACE_PROBE(usdt,end);
}
```

- popis sondy pro kompilaci (usdt\_probe.d)

```
provider usdt {
    probe begin(int data);
    probe end(); };
```

- kompilace

```
cc -c usdt.c
dtrace -32 -G -s usdt_probe.d usdt.o
cc -o usdt usdt.o usdt_probe.o
```

- sledování

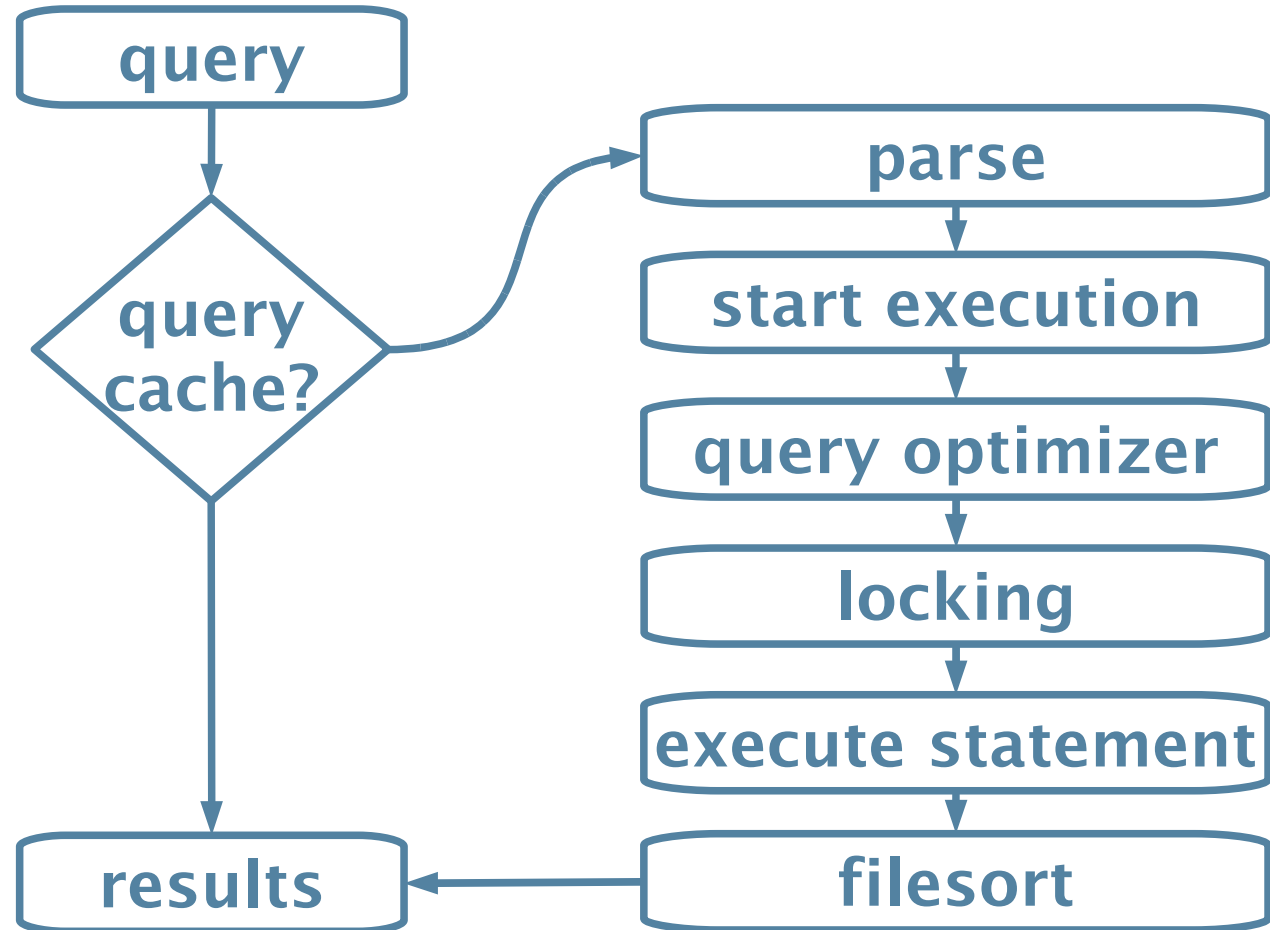
```
→ usdt$target:::begin usdt$target:::end
→ usdt$target:::begin { self->ts=timestamp; }
   usdt$target:::end /self->ts/
   { @=quantize(timestamp-self->ts); self->ts=0; }
```

# MySQL

aplikace

MySQL

- využití standardních providerů
  - pid\*/plockstat\*
  - MySQL Internals (O'Reilly)
  - Expert MySQL (Apress)



# provider pid\*

aplikace

MySQL

```
→ pid$target::*dispatch_command*:entry
  { printf("Query: %s\n", copyinstr(arg2)); }
→ pid$target::*dispatch_command*:entry
  { self->query=copyinstr(arg2); self-
    >start=timestamp; }
pid$target::*send_result_to_client*:entry
  { self->cache="Yes"; }
pid$target::*do_select*:entry
  { self->cache="No"; }
pid$target::*dispatch_command*:return
  { this->end=timestamp;
    this->elapsed=(this->end-self->start)/1000000;
    printf("From Query Cache?: %s in %d ms \t|
    Query: %s\n", self->cache, this->elapsed, self-
    >query); self->start=0; self->query=0; }
```

# provider mysql\*

aplikace

MySQL

- mysql 6.0.x, 5.4, 5.1.30 (OpenSolaris)
- 56 sond v různých oblastech

connection-start

command-start

**query-start** (query, connid, db, user, host)

query-parse-start

query-parse-done

query-cache- (hit, miss)

query-exec-start

query-exec-stop

query-done (status)

command-done

connection-done

→ **mysql**\$target::**query-start** { printf("Q: \"%s\" C:  
%i DB: %s U: %s H: %s\n", copyinstr(arg0), arg1,  
copyinstr(arg2), copyinstr(arg3),  
copyinstr(arg4)); }

# Apache

aplikace

apache

- modul "mod\_dtrace"
  - provider apache\*
  - převedení standardního API do DTrace
  - 7 sond
    - accept-connection, check-access, check-authorization, check-user-credentials, create-child, log-request, receive-request
- `apache*:::log-request { this->request=(int *)copyin(arg0, 56*4); printf("%s %s %s %d\n", copyinstr(this->request[17]), copyinstr(this->request[12]), copyinstr(this->request[55]), (int)this->request[16]); }`
- `./viewreqs.pl -d 5`
- `./apachetop.pl`

# PHP

aplikace

PHP

- modul "mod\_php5"
    - provider php\*
    - 11 sond
      - compile-file-entry, compile-file-return, error, exception-catched, exception-thrown, execute-entry, execute-return, function-entry, function-return, request-shutdown, request-startup
- `php*:::function-entry { @[copyinstr(arg0)] = count() }`
- `php*:::function-entry {printf("called %s() in %s at line %d\n", copyinstr(arg0), copyinstr(arg1), arg2) }`
- `php*:::function-entry /copyinstr(arg0)="blastoff"/ { self->ts=timestamp; }`
- `php*:::function-return /self->ts/ { @=quantize(timestamp-self->ts); self->ts=0; }`
- `./DTraceToolkit-0.99/Bin/php_flowtime.d`

# Java

aplikace

java

- generace
  - jstack()
  - java4/5 - libdvmtdi.so, dvm\*, 13 sond
  - java6
    - hotspot\* - 25 sond
    - hotspot\_jni\* - 474 sond
    - explicitní aktivace
      - +ExtendedDTraceProbes
      - +DTrace{Alloc,Method,Monitor}Probes
- `hotspot*:::method-entry { this->class= (char*)copyin(arg1, arg2+1); this->class[arg2]='\0'; this->method=(char*)copyin(arg3, arg4+1); this->method[arg4]='\0'; printf("%s::%s\n", stringof(this->class), stringof(this->method)); }`
- `./DTraceToolkit-0.99/Bin/j_syscolors.d -p $ (pgrep java)`
- `./DTraceToolkit-0.99/Bin/j_calls.d`
- `./DTraceToolkit-0.99/Bin/j_stat.d`

# JavaScript

aplikace

javascript

- součást "Spider Monkey JavaScript Engine"
  - Firefox  $\geq 3.0$
  - provider javascript\*
  - 11 sond
    - execute-done, execute-start, function-args, function-entry, function-info, function-return, function-rval, object-create, object-create-done, object-create-start, object-finalize
- **javascript\*:::function-entry**

```
{ @funcs[basename(copyinstr(arg0)),  
copyinstr(arg2)] = count(); }
```
- ./DTraceToolkit-0.99/Bin/js\_flowtime.d
- ./DTraceToolkit-0.99/Bin/js\_objnew.d
- ./DTraceToolkit-0.99/Bin/js\_objgc.d

# Dotazy (dtrace:::END)



pro prezentaci byly použity inspirace z volně dostupných materiálů  
z prezentací ve skupinách uživatelů OpenSolaris.org (CZOSUG/SKOSUG a jiných),  
z konferencí o OpenSolarisu, DTrace, Java, JavaScript, MySQL, CommunityOne a jiných,  
z referenčních materiálů Wiki na SolarisInternals.com,  
z dokumentace DTrace na docs.sun.com a dalších zdrojů