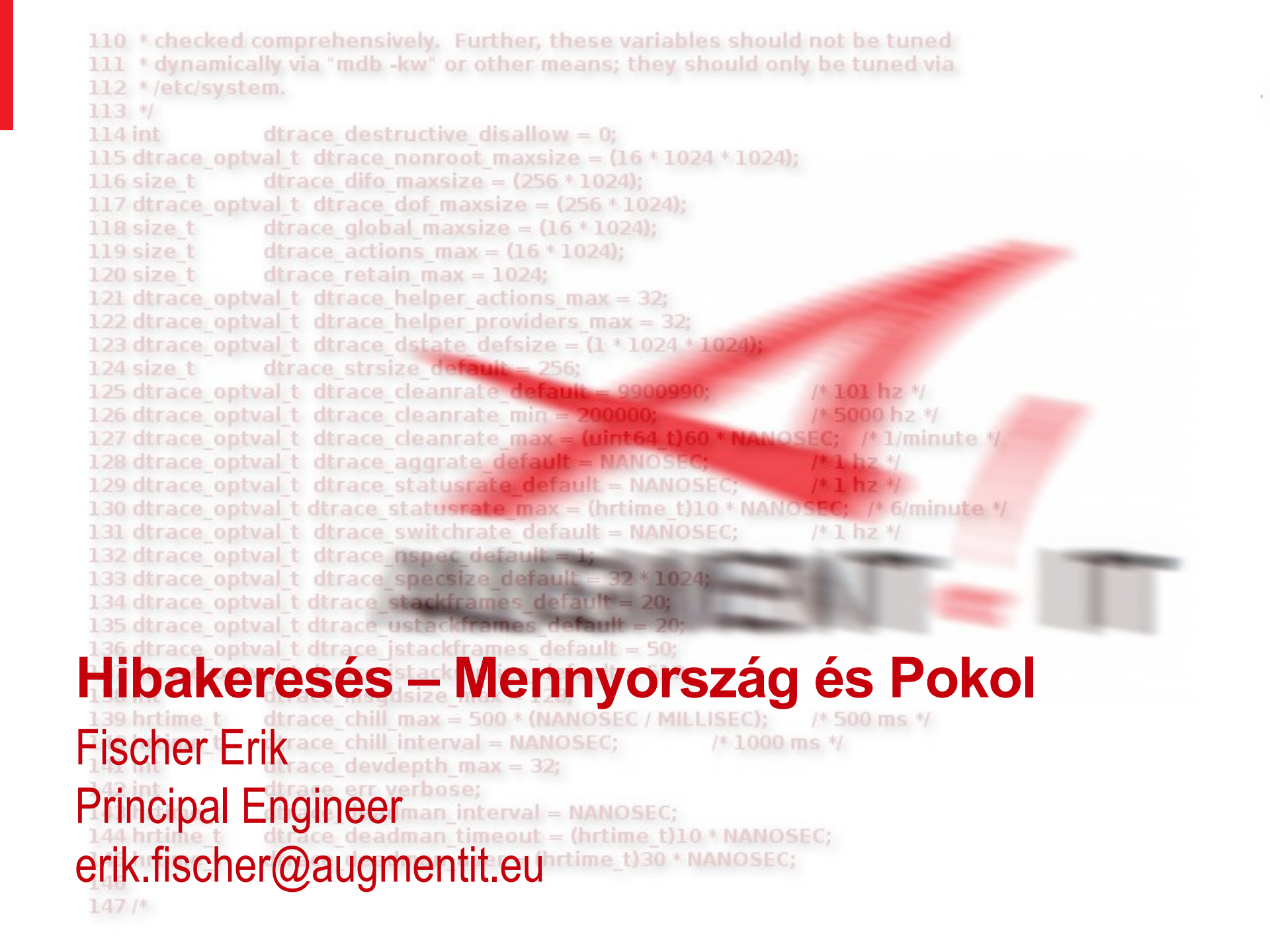


Hibakeresés – Mennyország és Pokol

Fischer Erik
Principal Engineer
erik.fischer@augmentit.eu

AUGMENT IT

A hibakeresés sohasem egyszerű

- Hibák pedig léteznek
  - > Nem csak programozási hibák vannak...
  - > Konfigurációs hibák
  - > Architektúrális hibák
  - > Tervezési hibák
- A jó hibákat könnyű megtalálni
- A rosszakat meg nagyon nehéz
- Az undorítóakat meg egyszerűen lehetetlen...

Hibakeresés – in vitro

- Végzetes hibáknál postmortem hibakeresés
  - Core dump és debugger [mdb(1), dbx(1)]
- Tranziens hibáknál (programozási hibák, vélt vagy valós teljesítmény problémák)
  - Ad hoc megoldások vagy szinte semmi
  - truss(1), mdb(1), *stat(1M), sar(1)
  - Ha forráskód is elérhető, akkor pl. Sun Studio Performance Analyzer

Hibakeresés – in vivo

- Szinte minden esetben csak invazív technikák
  - Bináris instrumentálás
  - Forrás szintű instrumentálás
  - Interposer library-k
  - Debug library-k
  - Debug kernel
- Általában
  - Durva beavatkozások
  - Lassú
  - Nagy az additív hibák injekciójának esélye

A tökéletes hibakereső rendszer

- Dinamikus
- Információt gyűjt
- Éles rendszereken is biztonságosan használható
- “Teljesen” biztonságosan
- Teljesítmény veszteség “nélkül” használható

AUGMENT IT

DTrace

- Interpretált nyelv (valójában D)
  - > Probe-ok
  - > Prédikátumok
  - > Akciók
- Dinamikus (nem csak) függvény be- és kilépési pontokat instrumentáló rendszer
- Kernel modul
- Unix parancs: dtrace(1M)

AUGMENT IT

A DTrace jellemzői

- User és Kernel rétegben is működik
- Ismeri és érti a syscall-okat
- 50000+ probe egy átlagos Solaris 10 rendszeren (update és hardver architektúra függő)
- Alapállapotban csak root-ként működik
- 3 privilégiumot igényel: dtrace_kernel, dtrace_proc és dtrace_user
- Bő 800 oldalas dokumentáció

Probe

- Az instrumentáció pontos helye egy hierarchiában (leírója egy 'n'-es)

  <provider, module, function, probe_name>

- A hierarchiában egy vagy akár szint is üresen hagyható
- Egy provider bocsájtja a rendelkezésünkre
- Modulokra és függvényekre bomlik
- Van neve (általában entry és return)
- Lehet minimalista módon reguláris kifejezéseket is használni: * ? [ ]
- dtrace -l

Provider

- Alapvetően az instrumentáció metodológiáját nyújtja
- Szoros kapcsolatban áll a DTrace keretrendszerrel
  - > Probe-okat ad a keretnek
  - > A keret értesíti a provider-t az egyes probe-ok aktiválásáról
  - > A provider egy probe tüzelésekor átadja a vezérlést a DTrace keretnek

Provider-ek I.

- fbt
  - > Function Boundary Tracing
  - > Minden kernel függvény be- és kilépési pontja
  - > ~45000
- syscall
  - > syscall tábla be- és kilépési pontjai
  - > ~450
- lockstat
  - > Kernel szinkronizációs primitívek
    - Contention-event
    - Hold-event

Provider-ek II.

- vminfo
  - Kernel vm statisztika
- sysinfo
  - Kernel sys statisztikák
- pid
  - Alkalmazások függvényeinek be- és kilépési pontjainak instrumentálása
  - Akár minden utasítást is képes instrumentálni...
- io
  - Diszk I/O események instrumentálása

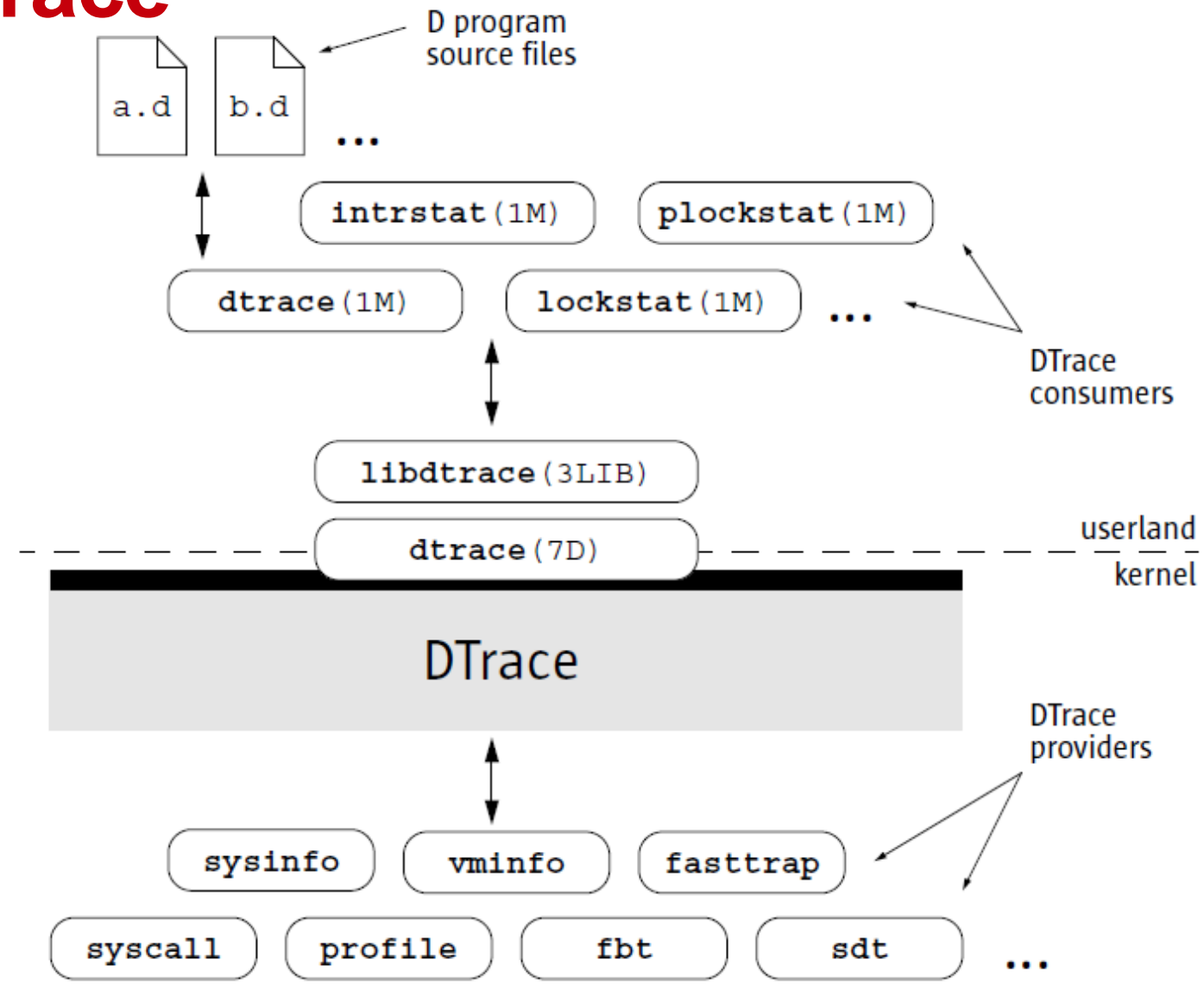
Provider-ek III.

- proc
  - Processz életciklus események
  - create, start, fault, exit, exec, signal
- sched
  - CPU ütemezési események
  - enqueue, dequeue, sleep, wakeup, on-cpu, off-cpu
- sdt
- mib
- cpc

Provider-ek IV.

- profile, tick
  - > Konfigurálható rátájú profiling interrupt
- fpuinfo
- plockstat
  - > User land szinkronizációs primitívek
- dtrace
  - > BEGIN
  - > END

DTrace


D program
source files
a.d
b.d
...
intrstat(1M)
plockstat(1M)
dtrace(1M)
lockstat(1M)
...
DTrace
consumers
libdtrace(3LIB)
dtrace(7D)
userland
kernel
DTrace
DTrace
providers
sysinfo
vminfo
fasttrap
syscall
profile
fbt
sdt
...

Szkriptek

- dtrace(1M) támogatja a szkripteket

```
#! /usr/sbin/dtrace –s
```

- Szkript szintaktika

```
[
provider:module:function:name[,
provider:module:function:name]*
[/predicate/]
{
 [action;]*
}
]+
```

Változók

- Skalár
  - Szokásos típusok (char, short, int, long, long long, float, double, long double)
- Asszociatív array-ek
  - Továbbfejlesztett Perl hash
  - A kulcs egy tetszőleges 'n'-es
  - Pl. array[execname, timestamp]
- String

Beépített változók

- pid – az aktuális processz ID-je
- ppid – a szülő processz ID-je
- tid – az aktuális thread ID-je
- execname – az aktuális programnév
- *curcpu – az aktuális CPU információs struktúrája
- *curthread – az aktuális thread kernel struktúrája
- *curpsinfo – az aktuális processz állapotleíró struktúrája
- *curlwpsinfo – az aktuális lwp állapotleíró struktúrája
- lgrp – az adott CPU lgroup ID-je
- pset – az adott CPU processor set ID-je
- timestamp – a bootolás óta eltelt idő [ns]
- vtimestamp – az adott processz virtualizált órája [ns]
- probprov, probemod, probefunc, probename – a probe adatai

...és még sokan mások

Thread lokális változók

- Egy név, de minden thread-nek saját adat
- Jelölése: self->
- A nem definiált változóknak nulla az értékük
- Ha nullázunk egy thread lokális változót, avval deallokáljuk

Makrók

- $[1-9] – az adott parancssori paraméterek
- $egid, $euid, $uid, $gid, $pid, $pgid, $ppid
- $projid
- $sid
- $taskid

Prédikátumok

- Akciók csak bizonyos feltételek megléte esetén történő érvényesítése
- Egy logikai D kifejezés
- Ha a kifejezés igaz, akkor és csak akkor hat az akció

Az akciókról általában I.

- Ez a valódi program
- Általában információt rögzít
- Ritkán információt változtat meg
- Akciók
  - > Műveletek változókkal és array-ekkel
  - > **NINCS FLOATING POINT ARITHMETIKA**
  - > Függvényhívások
- Bizonyos programozási szerkezetek nem léteznek
  - > Elágazás (if)
  - > Ciklus (for, while, until)

Az akciókról általában II.

- Arithmetikai műveletek
- Logikai műveletek
- Komparálás
- Értékadás
- Feltételes kifejezés

```
x= i==0 ? “zero” : “non-zero”;
```

Destruktív akciók

- A –w flag-gel engedélyezni kell
- stop() – megállítja az aktuális processzt
- raise() – szignált küld az adott processznek
- breakpoint() – kernel breakpoint triggerelése
- panic() – kernel pánik
- chill() – adott időre leállítja a processzt
- copyout() és copyoutstr() – adott változóból adott számú adatot egy memória helyre másol

Aggregációk

- DTrace függvény
- Egy aggregáció olyan f(x) függvény, ahol x egy tetszőleges n elemű adathalmaz és:

$$f(f(x_0) \cup f(x_1) \cup \ldots \cup f(x_n)) = f(x_0 \cup x_1 \cup \ldots \cup x_n)$$

- Ilyenek a count, a sum, az avg, a max és a min (pl. a median nem az, éppen ezért nincs is ilyen)
- Speciális aggregáció a quantize és az lquantize

Normalizáció és egyéb műveletek

- normalize()
  - > Nem valódi normálás, csak egy faktor tárolása
  - > Alapvetően megjelenítéshez
- denormalize()
  - > Egy meglévő norma faktor törlése
- Aggregáció törlése
  - > clear() - csak az értéket törli, a kulcsok maradnak
- Aggregáció csonkolás
  - > trunc() - az aggregáció csonkolása

Spekulatív követés

- Bizonyos esetekben csak speciális feltételek együttállása esetén kellenek adatok
- Viszont minden más esetben minden begyűjtött adatot el kell dobni
- Nem szűrhető prédikátumokkal
- Tipikusan hibák előfordulásakor gyűjtendő adatok, speciális program ágak monitorozása, etc.

Anonymous követés

- Fogyasztóhoz nem kötött nyomkövetés
- Pl. A boot folyamat követése
- /kernel/drv/dtrace.conf fájlba bekerülő direktívák
  - A dtrace(7D) betöltésekor automatikusan aktiválódik

AUGMENT IT

Kernel instrumentáció

- A belépési utasítás branch always utasításra cserélése
- Ugrás a DTrace keretbe
- A feladatok végrehajtása
- Az elcserélt utasítás végrehajtása
- Ugrás vissza oda, ahonnan jöttünk

User processz instrumentáció

- Alapvetően a kernel instrumentáció logikájával azonos
- A probléma, hogy user land-ből kernel land-be kell átkerülni
- A megoldás egy trap utasítás inzertálása
- Következménye egy speciális DTRACE trap entry

Syscall instrumentáció

- A syscall-ok valójában trap-eken keresztül hívódnak meg
- A syscall-ok táblákban vannak és ide indexelünk a trap “argumentumával”
- A megoldás az adott syscall bejegyzés cseréje egy DTRACE_SYSTRACE_SYSCALL bejegyzésre

DTrace a nagyvilágban

- Portok
  - FreeBSD
  - OS X
  - Linux (Oracle Enterprise Linux), folyamatban...
- Az OpenSolaris megszűnésével keletkezett *illumos* kernelre épülő rendszerekben:
  - Nexenta NexentaStor és illumian
  - OpenIndiana
  - Joyent SmartOS
  - OmniOS
  - DEYOS (egyelőre még nem publikus)
- Jelentős változások az Oracle Solaris 11-ben

AUGMENT IT

DTrace kérdések

- Az eltérő rendszerek kezelés
- Univerzális Dtrace szkriptek
- Univerzális és dinamikus transzlátorok

Példák

Fischer Erik
Principal Engineer
erik.fischer@augmentit.eu