DTrace: Vive la Révolution!

Jon Haslam : PS UK
Clive King : PTS EMEA

http://dtrace.eng

SunU

Agenda (Our!)

- Convince you that DTrace is
  - A revolution on system observability
  - Inductry Best of Breed by an Order of Magnitude
  - Something that will be useful to you
  - Something Sun *needs* customers to know about
    - ISVs
    - System Administrators
    - Capacity Planners
  - Writing D scripts should be your contribution to the knowledge engineering effort

Sun Proprietary/Confidential: Internal Use Only

Scope of Session

- DTrace is a large subsystem
  - This is just a taster
  - Basics of the Architecture
  - Basics of the language
  - Examples of how it can change your life
  - Live demo!

Sun Proprietary/Confidential: Internal Use Only

What DTrace can do for me?

- SE engaging a customer
  - Major selling point of S10
  - Get customer mindshare
- PS Consultant
  - Capacity Planning
- Support Services
  - Understand whole stack behaviour (N tier)
  - Answer SGR/ATS “where on object” and “When in lifecycle”

Sun Proprietary/Confidential: Internal Use Only

Who else?

- Software Developer/ISV
  - Who calls malloc in a signal handler?
  - Total observability of their codes behaviour
- Systems Administrator
  - Which process is sending packets to my ce card?
  - Disk A is busy, which application is doing I/O to it?
- Proactive Technical Analyst
  - Coverage analysis of patches
- Service Account Manager
  - A quiet life

Sun Proprietary/Confidential: Internal Use Only

Why DTrace

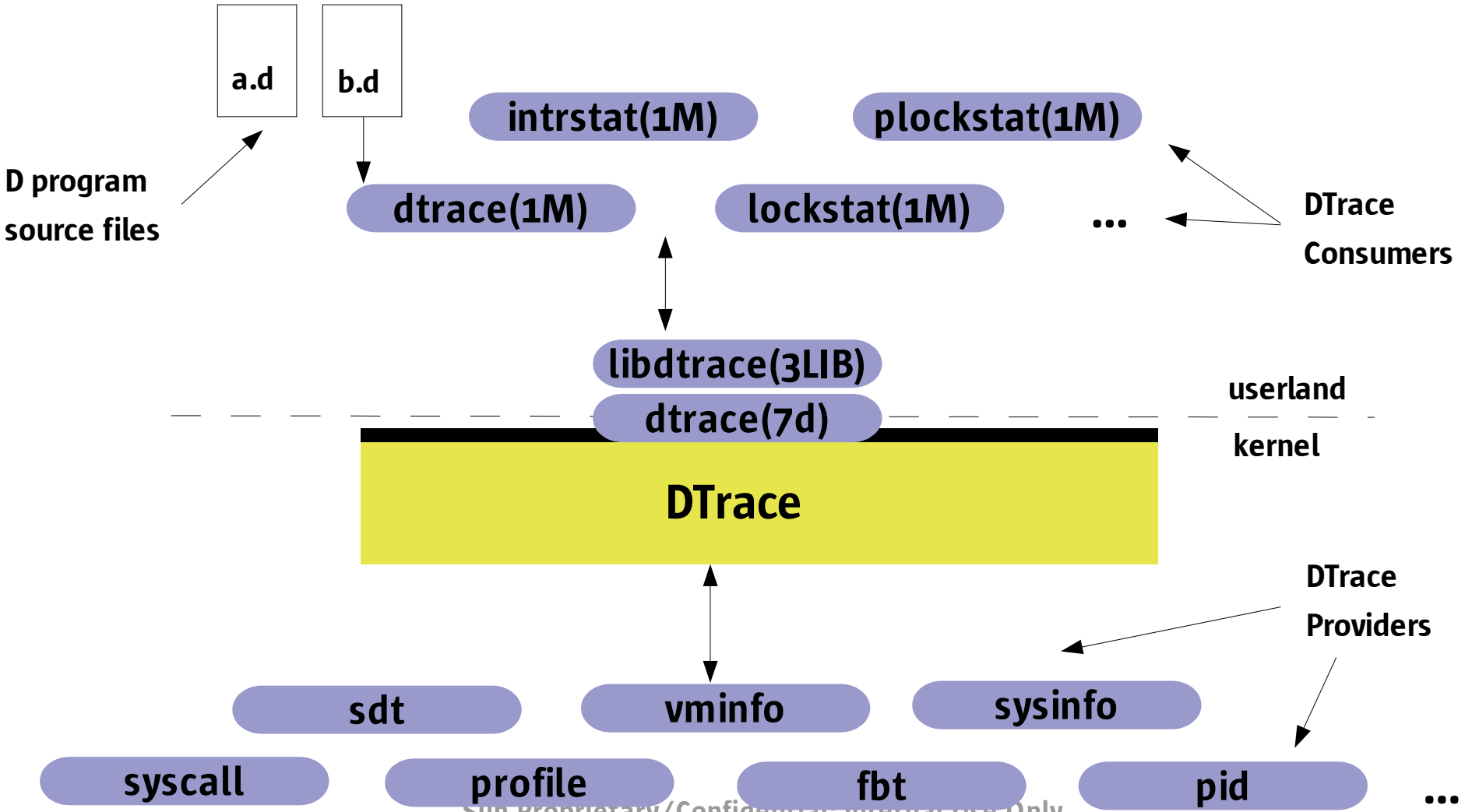
- Observability is the foundation for diagnosis of any type of problem.
- Solaris has no easy problems left
  - ➔ interaction of multiple components
  - ➔ intermittent (once every three months)
- DTrace is system centric

Sun Proprietary/Confidential: Internal Use Only

Demo Time!

Sun Proprietary/Confidential: Internal Use Only

DTrace Architecture



Sun Proprietary/Confidential: Internal Use Only

D Building Blocks

- The Clause – Base Unit of Currency

```
probe-description(s)
/
    optional predicate
/
{
    optional actions
}
```

```
Provider:Module:Function:Name

ufs_read:entry
fbt:ufs:ufs_read:return
tick-10s
syscall::open:entry
pid1234:::
seg*_unmap:entry
```

Sun Proprietary/Confidential: Internal Use Only

Probes

- Dynamically enabled points of interest
  - Kernel has approximately 30000 probes
  - From one function entry to every application instruction
- A *provider* exports a probe
- Consumers enable probes
- A probe has a *name* and is unique

Sun Proprietary/Confidential: Internal Use Only

Actions

- Executed when we hit a probe
- Mostly record state
- Some actions change state in a well defined manner
  - Called *destructive* actions
  - Disabled by default
  - Fun for rainy days as well ...

Sun Proprietary/Confidential: Internal Use Only

Actions

- *stack()/ustack()*- kernel/userland stack trace
- *trace()* - records argument
- *printf()* - formatted output
- chill() - spin for specified time
- exit() - D consumer exits
- raise() - send signal to process

Sun Proprietary/Confidential: Internal Use Only

Predicates

- *Predicates* allow actions to be taken only when certain conditions are true
- A predicate is evaluated when a probe is fired i.e:

```
syscall::entry
/ zonename == “prod1” && execname == “apache” /
{
   @a[probefunc] = count();
}
```

Sun Proprietary/Confidential: Internal Use Only

Aggregations (insert expletive)

- Group data items together and treat collectively
- Summarise large data volumes
- Enable us to look for patterns and trends

```
dtrace -n pid703:libc:malloc:entry'{@[arg0] = count()'}
```

Sun Proprietary/Confidential: Internal Use Only

Aggregations

- Other aggregating functions are:
  - sum() - add elements together
  - avg() - avg data elements
  - min() - minimum of data elements
  - max() - maximum of data elements
  - quantize() - power-of-2 frequency distribution
  - lquantize() - linear frequency distribution

Sun Proprietary/Confidential: Internal Use Only

Aggregations

- Result of an aggregating function keyed by an arbitrary value
- An array indexed by whatever you want and each element having the results of a function associated with it:

```
syscall::read:entry, syscall::write:entry
{
   @[probefunc] = quantize(arg2);
}
```

Sun Proprietary/Confidential: Internal Use Only

Aggregations

- Trivial but Powerful Example

```
syscall::pollsys:entry
/ arg2 < 10 /
{
    @a[execname, ustack()] = count();
}
```

Highlighted the amount of times an individual stack in StarOffice ended up calling poll(2) with a sub 10mS timeout.

Sun Proprietary/Confidential: Internal Use Only

Pushing the Envelope

- What's in a read?
- Patch risk management
- Adventures with a Financial ISV
- N-tier application debugging

Sun Proprietary/Confidential: Internal Use Only

Lots, Lots More...

- Speculations
- Anonymous Tracing
- Security
- Actions, Actions and more Actions
- Postmortem Tracing
- Buffering
- Tunables
- Lots of Language Details

Sun Proprietary/Confidential: Internal Use Only

What Next?

- Providers (http://dtrace.eng/pteams)
  - sched
  - vm
  - io
  - filesystems
  - network
  - nfs
- Java
- Additional Actions

Sun Proprietary/Confidential: Internal Use Only

Call to recipies

- Fully Embrace DTrace
- Make the chance to demo it to all your customers
- Write and share D recipes on Bigadmin
- Become part of the community
- Be inventive, use DTrace as part of the Solution
- Use DTrace to improve other products

Sun Proprietary/Confidential: Internal Use Only

Further Information

- http://dtrace.eng
  - Answerbook and Papers
  - Recipes
- http://www.sun.com/bigadmin
- http://ktd.eng
- dtrace-interest@kiowa.eng

Sun Proprietary/Confidential: Internal Use Only