

open



USE



IMPROVE



EVANGELIZE

DTrace For Web 2.0 Technologies

Simon Ritter
Technology Evangelist
Sun Microsystems

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
:::
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை

What is instrumentation?

- Question: How do you typically do instrumentation?
- Answer:

```
#ifdef DEBUG  
    printf("Value of Q: %d", Q);
```
- Question: What do you need to do to add new instrumentation?
- Answer: edit, compile, link.....
- What if you could do this dynamically, both on the system and your own code?



Introducing DTrace

- DTrace is a dynamic instrumentation framework that was introduced in Solaris 10
- Allows for dynamic instrumentation of the OS and applications (including Java-based applications as well as some scripting languages, as we will see later)
- Available out of the box; a typical system has more than 50,000 probes
- Includes D, a dynamically interpreted language, that allows for arbitrary actions and predicates



Introducing DTrace, (Cont.)

- Designed explicitly for use on production systems
- Zero performance impact when not in use
- Completely safe; no way to cause panics, crashes, data corruption or pathological performance degradation
- Powerful data management primitives eliminate need for most postprocessing

The DTrace Revolution

- DTrace tightens the diagnosis loop:



- Tightened loop effects a revolution in the way we diagnose transient failure
- Focus can shift from instrumentation stage to hypothesis stage:
 - Much less labor intensive, less error prone
 - Much more brain intensive
 - Much more effective! (And a lot more fun)



The D Language

- D is a C-like language specific to DTrace with some constructs similar to awk(1)
- Built-in variables like `execname` and `timestamp`
- Predicates can use arbitrary expressions to select which data is traced and which is discarded
- Actions to trace data, record stack backtraces, stop processes at points of interest, etc.

D Language - Format.

- A D script consist of one or more clauses
- A clause has the following structure:

```

                                #!/usr/sbin/dtrace -s
probe description → syscall::entry
/ predicate / → /execname=="bash"/
{
    action statements → printf("%s called\n",
                                probefunc);
}
    
```

- Example, “Print all the system calls executed by bash”

open



USE



IMPROVE



EVANGELIZE

DEMO

Simple DTrace Invocations

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
ᄒᄒᄒᄒ
πᄒᄒᄒ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை

open



USE



IMPROVE



EVANGELIZE

DTrace and Scripting JavaScript, php etc.

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
•••••
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை



DTrace and Scripting Language

- DTrace has been integrated into many scripting languages.
- Examples
 - PHP
 - JavaScript in Firefox 3.0
 - Ruby
 - Python
 - And many more



DTrace and PHP

- There are two probes in the provider.
 - function-return
 - function-entry
- You can use the 5 args in the action.
 - arg0 = the function name
 - arg1 = the filename
 - arg2 = the line number
 - arg3 = classname (or an empty string)
 - arg4 = object/class operator (::, ->, or an empty string)

DTrace and PHP

- Simple D-Script for PHP

```
#!/usr/sbin/dtrace -Zqs
```

```
php* :::function-entry  
{  
    @[copyinstr(arg0)] = count();  
}
```

- Note: -Z will allow probes that have zero match.
- Run this script in one window while you run the php script in another window.



DTrace and PHP

- More information
- Wez Furlong who created the provider has details in <http://netevil.org/node.php?uuid=42f82a8b-09b7-5013-1667-2f82a8b24ead>
- Bryan Cantrill has a nice demo script in his blog. http://blogs.sun.com/bmc/entry/dtrace_and_php_demoonstrated



DTrace and Javascript

- DTrace probes have been added to Mozilla to help observe Javascript application.
- The following probes are implemented in the javascript provider:
 - execute-start
 - execute-done
 - function-entry
 - function-return
 - object-create & object-finalize
 - Object-create-start & object-create-done
 - layout-start
 - layout-end

DTrace and Javascript

- Print a function flow of javascript.

```
#!/usr/sbin/dtrace -ZFs
javascript*:::_function-entry
{
    trace(copyinstr(arg2));
}

javascript*:::function-return
{
    trace(copyinstr(arg2));
}
```



DTrace and Python

- DTrace probes have been added to python
- One of the special features is that `ustack()` works well on these probes
- These probes are available in OpenSolaris 2008.05 and later. See:
 - http://blogs.sun.com/levon/entry/python_and_dtrace_in_build
- DTrace toolkit has a set of script for Python
- We will now look at the list of probes and an example



DTrace and Python -probes

- Available probes
 - function-entry – fires when you call a function
 - function-return – fires when you return from function
 - Both these provide the following args:
 - arg1 – filename
 - arg2 – subroutine name
 - arg3 – line number



DTrace and Python - example

- Here is a simple script to view python function calls issued by any python application on the system

```
#!/usr/sbin/dtrace -Zs
python*:::function-entry
{
    @[copyinstr(arg1)] = count();
}
```



DTrace and Ruby

- DTprobesrace have been added to Ruby by Joyeur
- Some good info on Ruby and DTrace can be found on the Joyeur site
 - <http://www.joyeur.com/2007/05/07/dtrace-for-ruby-is-available>
- You can get the probe from the above site and some good examples
- DTrace toolkit has a set of script that you can use for ruby



Ruby DTrace probes

- function-entry → Ruby method is called
- function-return → Ruby method returns
- raise → Ruby exception is raised
- rescue → Ruby exception is rescued
- line → For each line of Ruby executed
- gc-begin → GC begins
- gc-end → GC ends
- object-create-start → Ruby object creation started
- object-create-done → Ruby object created
- object-free → Ruby object freed
- ruby-probe → Probe that can be fired from Ruby code

Ruby probe arguments

Function name	args[0]	args[1]	args[2]	args[3]
• function-entry	➔ Ruby class	Method name	Source File	Line number
• function-return	➔ Ruby class	Method name	Source File	Line number
• raise	➔ Ruby class	Source File	Line number	-
• rescue	➔ Source File	Line number	-	-
• line	➔ Source File	Line number	-	-
• gc-begin	➔ -	-	-	-
• gc-end	➔ -	-	-	-
• object-create-start	➔ Ruby type	Source File	Line number	-
• object-create-done	➔ Ruby type	Source File	Line number	-
• object-free	➔ Ruby type	-	-	-
• ruby-probe	➔ Any string	-	-	-

DTrace and Ruby - example

- Simple script to view a summary of all ruby function calls, per pid

```
#!/usr/sbin/dtrace -Zs
ruby*:::function-entry
{
    @[copyinstr(arg1), pid] = count();
}
```

open



USE



IMPROVE



EVANGELIZE

DTrace Toolkit

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
ᄒᆞᆫ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை



DTrace Toolkit

- The DTraceToolkit is a collection of useful
 - Documented scripts developed by the OpenSolaris DTrace community
 - Available under www.opensolaris.org
 - Ready to use D scripts
 - The toolkit contains:
 - the scripts
 - the man pages
 - the example documentation
 - the notes files



DTrace Toolkit

- Get it from <http://www.opensolaris.org/os/community/dtrace/dtracetoolkit>
- Download
 - gunzip & tar xvf
 - ./install
 - A nice guide comes with the script
 - See the Docs/Contents for more details.
 - Set PATH to include /opt/DTT

open



USE



IMPROVE



EVANGELIZE

DTrace Toolkit Demo

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
•••••
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை

open



USE



IMPROVE



EVANGELIZE

DTrace resources

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
ᄀᄀᄀᄀ
πᄀᄀᄀ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை



DTrace Resources

- Here are a few of the many DTrace resources available for you
 - Open Solaris DTrace community page
<http://www.opensolaris.org/os/community/dtrace/>
 - DTrace toolkit contains a lot of very useful scripts
<http://www.opensolaris.org/os/community/dtrace/dtracetoolkit/>
 - “Solaris Dynamic Tracing Guide” is an excellent resource. Sample scripts in /usr/demo/dtrace
<http://docs.sun.com/db/doc/817-6223>



More DTrace Resources

- The DTrace HowTo guide
<http://www.sun.com/software/solaris/howtoguides/dtracehowto.jsp>
- Read the Blogs from Bryan Cantrill, Adam Leventhal, Mike Shapiro
<http://blogs.sun.com/roller/page/bmc>
<http://blogs.sun.com/roller/page/ahl>
<http://blogs.sun.com/mws>
- A. Sundararajan, Dtrace and Java Blog
<http://blogs.sun.com/sundararajan>

open



USE



IMPROVE



EVANGELIZE

Thank you!

Simon Ritter
blogs.sun.com/simonri
simon.ritter@sun.com

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
ᄇᄇᄇᄇ
πἰπῶ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை