Joyent

Observing Node Performance with DTrace

Topic Outline

Joyent

- **Introduction**
  - What is DTrace?
  - How DTrace can be used with node.js
- **DTrace kernel actions**
- **DTrace the node engine**
- **DTrace node applications**
- **Using mdb with node**

But First, Some History...

Joyent

DecWriter II

```
$ ed foo.c
1,$p
main(argc, argv)
int argc;
char *argv[];
{
  register x;
  sub(x);
}

sub(a)
int a;
{
  printf("debug statement\n");
}

$ cc foo.c
$ ./a.out
debug statement
$
```

PDP 11/45

- **See http://cm.bell-labs.com/cm/cs/who/dmr/retro.html**

Debugger History

Joyent

- printf (console.log())
- adb
- dbx
- gdb
- prof/gprof
- truss/strace
- pdb, jdb, various javascript debuggers, etc.
- proc(1) tools
- mdb
- DTrace

Problems with Debuggers

Joyent

- **Most debuggers are for post-mortem debugging**
  - The problem occurred earlier in the code
- **Printf, console.log, truss, strace can generate a lot of output**
- **Setting breakpoints and single-stepping effect timing and cause process to halt while doing debugging**
  - Not good for production use
- **Many debuggers are not easily extensible and are language specific**

Node.JS Failures

Joyent

- **Logic Errors**
  - Uncaught exception
  - Directed exit
  - Infinite loop
  - Incorrect results
- **Latency Bubbles**
- **Memory Leaks**

What is DTrace?

Joyent

- **Tool that allows one to dynamically instrument code from application level and into the kernel.**
- **Can be used safely on production systems.**
- **Uses:**
  - Performance Analysis
  - Debugging
  - Code coverage
  - Find out wtf is happening in your software
- **Available on illumos, smartOS, and other Solaris 10 derivatives, as well as *BSD and Mac OS X.**

Terminology

Joyent

- **System Call - Request for an action by the Operating System**
- **Probe - An instrumentation point in the code**
  - Dynamic and Static probes are provided, and new ones can be added
    - provider:module:function:probename{action}
- **Action - Executed when a probe fires**
- **Predicate - Optional boolean to determine whether or not to execute the action**
- **Example:** `syscall::read:entry/pid == 713/{trace();}`

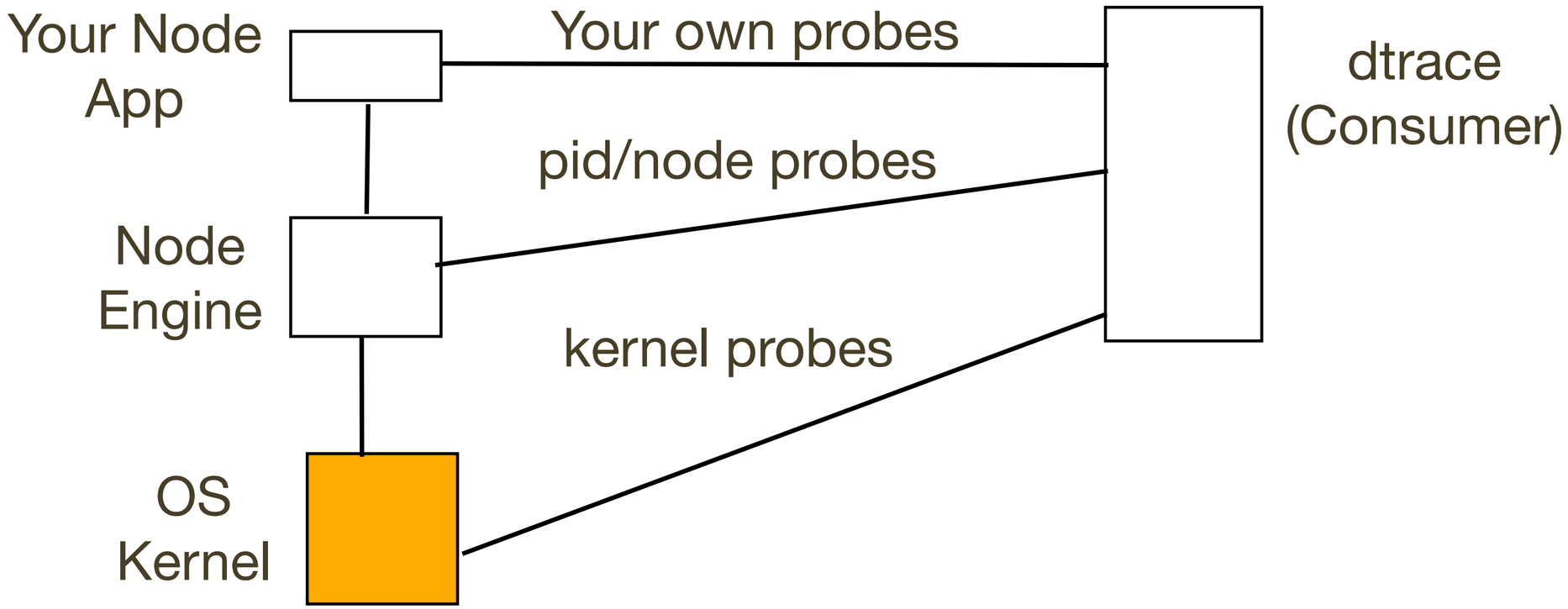
Node.js with DTrace Support

Joyent

- From www.nodejs.org download site

```
# curl -O http://nodejs.org/dist/v0.8.11/node-v0.8.11.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 11.2M  100 11.2M    0     0   253k       0  0:00:45  0:00:45 --:--:--  349k
# gtar -xpf node-v0.8.11.tar.gz
# pkgin install gcc-compiler-4.6.1
...
# cd node-v0.8.11
# ./configure
...
...
# make
...
# make install   <-installs in /usr/local/bin
...
# export PATH=/usr/local/bin:$PATH
# node -v
v0.8.11
# cd ..
# npm install dtrace-provider
...
# npm install restify  <- This is not necessary, but will be used in some of the demos
...
```

Architecture



- **With DTrace, you can trace events in**
  - The node Engine
  - Node.js scripts
  - The kernel (system calls, scheduling, memory management, etc.)

Some Simple Examples

Joyent

- **Show system calls made by a running node process**

```
# dtrace -n 'syscall::::entry/pid==26442/{}'
dtrace: description 'syscall::::entry' matched 234 probes
CPU     ID                    FUNCTION:NAME
  1  10157                      write:entry
  1  10283                   lwp_park:entry
  1  10155                       read:entry
  4  10155                       read:entry
  4  10157                      write:entry
...
```

Some Simple Examples (Continued)

Joyent

- Count system calls made by a running node process

```
# dtrace -n 'syscall::::entry/execname == "node"/
{@[probefunc]=count();}'
dtrace: description 'syscall::::entry' matched 234 probes
(^C)
...
  munmap                                                31
  portfs                                                36
  lwp_park                                              37
  fcntl                                                 39
  mmap64                                                53
#
```

An Example Measuring System Call Latency

Joyent

- systime.d

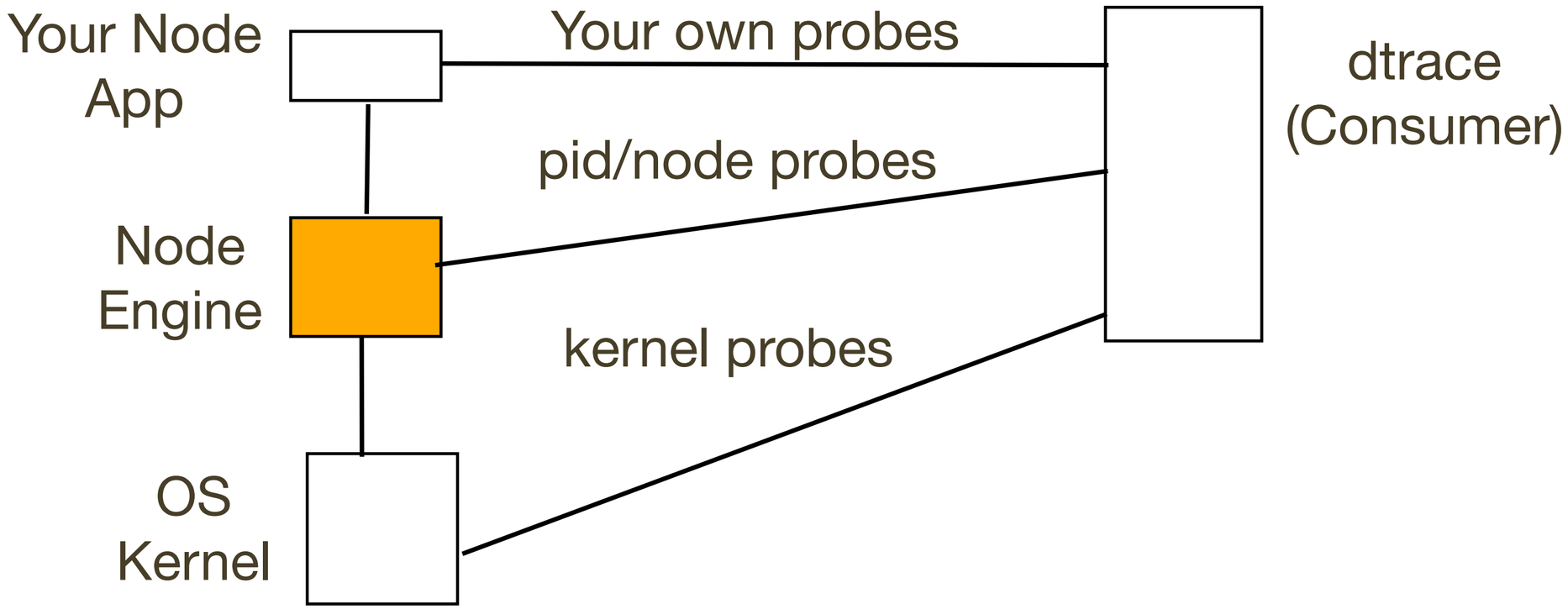
```
#!/usr/sbin/dtrace -qs

syscall::entry
/execname == "node"/
{
    self->ts = timestamp;
}
syscall::return
/self->ts/
{
    @[probefunc] = quantize(timestamp - self->ts);
    self->ts = 0;
}
END
{
    printa("SYSCALL    NSECS      # OF OCCURANCES\n%s%@lx\n", @);
}
```

An Example Measuring System Call Latency (Continued)

Joyent

```
# ./systime.d
...
SYSCALL     NSECS                                       # OF OCCURANCES
  read

           value  ------------- Distribution ------------- count
            1024 |                                         0
            2048 |@@@@@@@@@@@@@@                           3
            4096 |@@@@@@@@@@                               2
            8192 |@@@@@@@@@@                               2
           16384 |@@@@                                     1
           32768 |                                         0
           65536 |                                         0
          131072 |                                         0
          262144 |                                         0
          524288 |                                         0
         1048576 |                                         0
         2097152 |                                         0
         4194304 |                                         0
         8388608 |                                         0
        16777216 |                                         0
        33554432 |                                         0
        67108864 |                                         0
       134217728 |@@@@                                     1
       268435456 |                                         0

...
```

Architecture



- **With DTrace, you can trace events in**
  - The node Engine
  - Node.js scripts
  - The kernel (system calls, scheduling, memory management, etc.)

The Node DTrace Provider

Joyent

- Set of USDT probes built into node

```
# dtrace -l -n 'node*:::{}'
   ID   PROVIDER            MODULE                          FUNCTION NAME
57166  node11665              node
_ZN4nodeL14dtrace_gc_doneEN2v86GCTypeENS0_15GCCallbackFlagsE gc-done
57167  node11665              node
_ZN4nodeL15dtrace_gc_startEN2v86GCTypeENS0_15GCCallbackFlagsE gc-start
57168  node11665              node _ZN4node26DTRACE_HTTP_CLIENT_REQUESTERKN2v89ArgumentsE
http-client-request
57169  node11665              node _ZN4node27DTRACE_HTTP_CLIENT_RESPONSEERKN2v89ArgumentsE
http-client-response
57170  node11665              node _ZN4node26DTRACE_HTTP_SERVER_REQUESTERKN2v89ArgumentsE
http-server-request
57171  node11665              node _ZN4node27DTRACE_HTTP_SERVER_RESPONSEERKN2v89ArgumentsE
http-server-response
57172  node11665              node _ZN4node28DTRACE_NET_SERVER_CONNECTIONERKN2v89ArgumentsE
net-server-connection
57173  node11665              node _ZN4node22DTRACE_NET_SOCKET_READERKN2v89ArgumentsE net-
socket-read
57174  node11665              node _ZN4node23DTRACE_NET_SOCKET_WRITEERKN2v89ArgumentsE net-
socket-write
57175  node11665              node _ZN4node21DTRACE_NET_STREAM_ENDERKN2v89ArgumentsE net-
stream-end
```

The Node DTrace Provider
Probe Arguments

Joyent

```
# dtrace -l -v -n 'node*:::http-server-request, node*:::http-server-response{}'
   ID   PROVIDER            MODULE                          FUNCTION NAME
57170  node11665              node _ZN4node26DTRACE_HTTP_SERVER_REQUESTERKN2v89ArgumentsE
http-server-request

...

   Argument Types
           args[0]: node_http_request_t *
           args[1]: node_connection_t *

57171  node11665              node _ZN4node27DTRACE_HTTP_SERVER_RESPONSEERKN2v89ArgumentsE
http-server-response

  ...
           args[0]: node_connection_t *
```

The Node DTrace Provider Probe Arguments (Continued)

Joyent

- **In** `node-v0.8.11/src/node.d`

```
typedef struct {
    string url;
    string method;
    string forwardedFor;
} node_http_request_t;
```

```
typedef struct {
    int fd;
    string remoteAddress;
    int remotePort;
    int bufferSize;
} node_connection_t;
```

The Node DTrace Provider: Example 1

Joyent

```
/* echo-server.d */

#pragma D option quiet

BEGIN
{
     printf("%-22s %-20s %-8s %-16s %-16s %-16s\n",
       "DIRECTION", "URL", "METHOD", "REMOTEADDRESS", "REMOTEPORT", "BUFFERSIZE");
}

node*:::http-server-request
{
     printf("%-22s %-20s %-8s %-16s %-16d %-16d\n",
       probename, args[0]->url, args[0]->method, args[1]->remoteAddress,
       args[1]->remotePort, args[1]->bufferSize);
}

node*:::http-server-response
{
     printf("%-22s %-20s %-8s %-16s %-16d %-16d\n",
       probename, " ", " ", args[0]->remoteAddress,
       args[0]->remotePort, args[0]->bufferSize);
}
```

The Node DTrace Provider: Example 1 (Continued)

Joyent

- **Client**

```
# curl http://165.225.154.78:8080/echofile-server.js > /dev/null
...
```

- **Server**

```
# dtrace -L /usr/local/lib/dtrace -s echo-server.d
DIRECTION              URL                    METHOD   REMOTEADDRESS    REMOTEPORT           
BUFFERSIZE
http-server-request    /echofile-server.js    GET      62.203.55.164    58027                0
http-server-response                                   62.203.55.164    58027                0
http-server-response                                   62.203.55.164    58030                0
http-server-request    /echofile-server.js    GET      62.203.55.164    58030                0
http-server-request    /echofile-server.js    GET      62.203.55.164    58036                0
http-server-response                                   62.203.55.164    58036                0
http-server-request    /echofile-server.js    GET      62.203.55.164    58037                0
http-server-response                                   62.203.55.164    58037                0
http-server-request    /echofile-server.js    GET      62.203.55.164    58038                0
http-server-response                                   62.203.55.164    58038                0
http-server-request    /systime.d             GET      62.203.55.164    58363                0
http-server-response                                   62.203.55.164    58363                0
http-server-request    /favicon.ico           GET      62.203.55.164    58364                0
http-server-response                                   62.203.55.164    58364                0
...
```

Request/Response Latency

Joyent

```
/* server-latency.d */

#pragma D option quiet

node*:::http-server-request
{
     ts[args[1]->remoteAddress, args[1]->remotePort] = timestamp;
     url[ts[args[1]->remoteAddress, args[1]->remotePort]] = args[0]->url;
}

node*:::http-server-response
/ts[args[0]->remoteAddress, args[0]->remotePort]/
{
     this->t = ts[args[0]->remoteAddress, args[0]->remotePort];
     @[url[this->t], args[0]->remoteAddress] = quantize((timestamp-this->t)/1000);
     ts[args[0]->remoteAddress, args[0]->remotePort] = 0;
}

END
{
     printf("%-20s: %-16s\n", "URL", "REMOTEADDRESS");
     printa("%-20s: %-16s\nMICROSECONDS\n%@d\n", @);
}
```

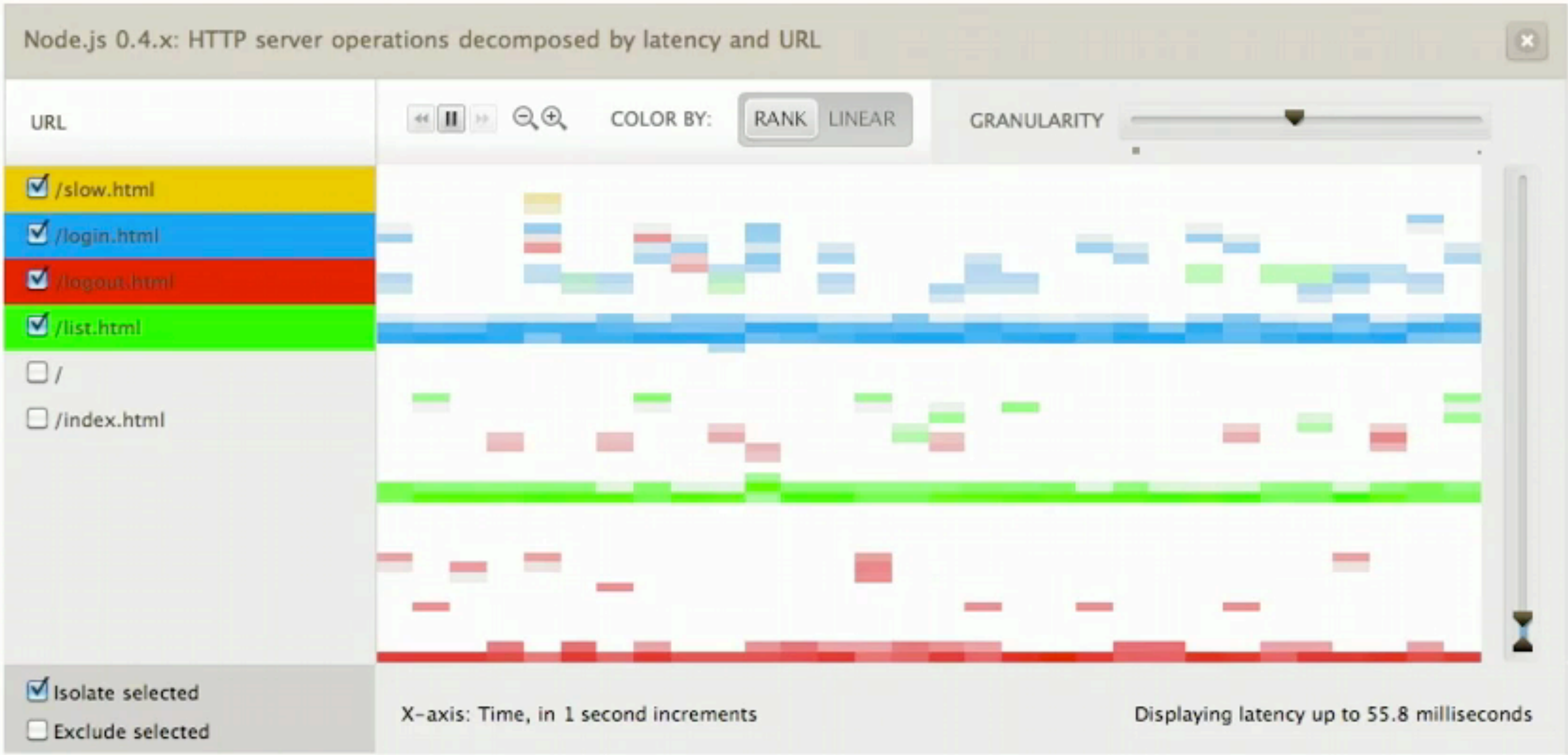
Request/Response Latency (Continued)

Joyent

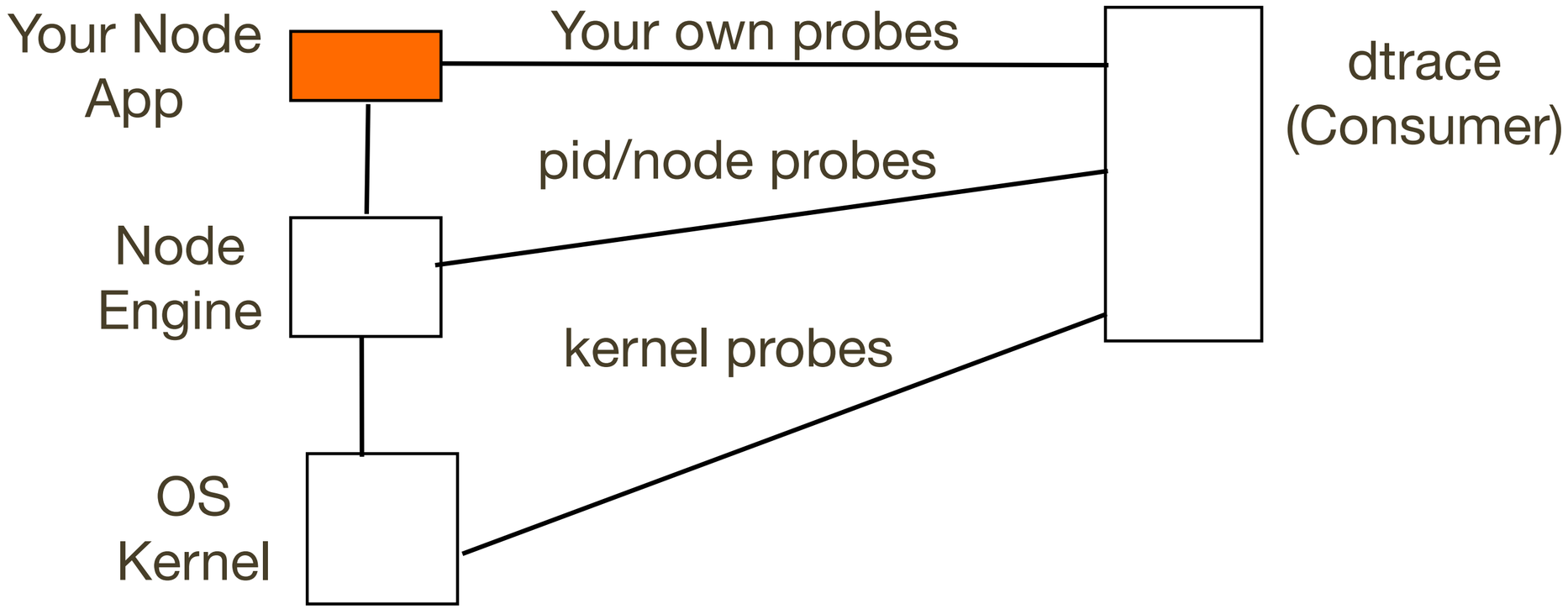
```
# dtrace -L /usr/local/lib/dtrace -s server-latency.d

URL                    : REMOTEADDRESS
/tmp/words             : 165.225.154.77
MICROSECONDS

           value  ------------- Distribution ------------- count
            1024 |                                         0
            2048 |@@@@                                     11
            4096 |@@@@@@@@@@@@@@@@@                        43
            8192 |@@@@@@                                   14
           16384 |@@@@@@@@@@@@                             31
           32768 |                                         1
           65536 |                                         0

/tmp/words             : 83.79.36.187
MICROSECONDS

             value  ------------- Distribution ------------- count
            524288 |                                         0
           1048576 |@                                        3
           2097152 |@@                                       4
           4194304 |@@                                       4
           8388608 |@@@@                                     11
          16777216 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 74
          33554432 |@@                                       4
          67108864 |                                         0
```

Heatmaps

Joyent



Architecture



- **With DTrace, you can trace events in**
  - The node Engine
  - Node.js scripts
  - The kernel (system calls, scheduling, memory management, etc.)

DTrace Your node.js Application

Joyent

- The dtrace-provider for Node.js allows you to create statically defined probes (USDT) in your application.
- Effectively, a way to add print statements to your scripts which only have effect when/if the probes are enabled.
- But better than print... You decide what to enable and what to print at runtime.
- Install
  - `npm install dtrace-provider`

Add Probes to Your Node App

Joyent

```
/* echofile-server.js */
...
var dtp = require('dtrace-provider').createDTraceProvider('echofile-
server');

dtp.addProbe('echo-start', 'char *');
dtp.addProbe('echo-done', 'char *', 'int');
dtp.addProbe('echo-error', 'char *', 'char *')
...
```

- **Define probes and arguments**

```
    dtp.fire('echo-start', function() {
        return [req.params[0]];
    });
    ...
        dtp.fire('echo-error', function() {
            return [req.params[0], JSON.stringify(e)];
        });
    ...
    dtp.fire('echo-done', function() {
        return [req.params[0], len];
    });
...
```

- **Add probes to your code**

DTrace The Added Probes

Joyent

- **Use dtrace to enable the probes you've added**

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

#pragma D option quiet

echofile-server*:::echo-start
{
    printf("%s: %s\n", probename, copyinstr(arg0));
}

echofile-server*:::echo-done
{
    printf("%s: %s %d bytes\n", probename, copyinstr(arg0), arg1);
}

echofile-server*:::echo-error
{
    printf("%s\n", copyinstr(arg1));
}
```

Enabling the Added Probes

Joyent

```
# ./echofile-server.d
echo-start: tmp/bigwords
echo-done: tmp/bigwords 20667400 bytes
echo-start: tmp
echo-done: tmp 116 bytes
{"errno":28,"code":"EISDIR"}
echo-start: blah
{"errno":34,"code":"ENOENT","path":"blah"}
...
```

List Probes Built-in for Restify

Joyent

```
# dtrace -l -P 'myapp*'
   ID   PROVIDER            MODULE                          FUNCTION NAME
57309 myapp13446            module                              func get100-start
57310 myapp13446            module                              func get100-done
57311 myapp13446            module                              func get100-
parseAccept-start
57312 myapp13446            module                              func get100-
parseAccept-done
57313 myapp13446            module                              func get100-
parseQueryString-start
57314 myapp13446            module                              func get100-
parseQueryString-done
57315 myapp13446            module                              func get100-parseBody-
start
57316 myapp13446            module                              func get100-parseBody-
done
57317 myapp13446            module                              func get100-sget-start
57318 myapp13446            module                              func get100-sget-done
```

Using DTrace to Help Find Memory Leaks

Joyent

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

pid$target::malloc:entry
{
    self->nbytes = arg0;
}

pid$target::malloc:return
/self->nbytes/
{
    printf("%lx: %lx malloc\n", arg1, self->nbytes);
    self->nbytes=0;
}

pid$target::free:entry
{
    printf("%lx: free\n", arg0);
}
```

Using DTrace to Help Find Memory Leaks (Continued)

Joyent

```
# ./malloc.d -p `pgrep node`
887fdc8: 158 malloc
8880ac8: 534 malloc
88b9248: 5c malloc
8881c48: 2bc malloc
887fdc8: free
8880ac8: free
8881c48: free
887fdc8: 158 malloc
8880ac8: 534 malloc
88b91d8: 5c malloc
8881c48: 2bc malloc
887fdc8: free
8880ac8: free
8881c48: free
...
```

Using DTrace to Help Find Memory Leaks (Continued)

Joyent

```
# dtrace -x jstackstrsize=8k -qn 'pid$target::malloc:entry/arg0 == 0x5c/
{jstack()}' -p `pgrep node`

              libumem.so.1`malloc
              libstdc++.so.6.0.13`_Znwj+0x28
              libstdc++.so.6.0.13`_Znaj+0x1e
              node`_ZN2v88internal24ExternalReferenceDecoderC1Ev+0x62
...
              run at /root/leak2.js line 9
              (anon) as (anon) at /root/leak2.js line 17
              (anon) as list.ontimeout at timers.js position 4960
...
              node`uv_run+0x17
              node`_ZN4node5StartEiPPc+0x1c7
              node`main+0x1b
              node`_start+0x83
```

mdb - Modular Debugger

Joyent

- **Extensible, general purpose debugging tool**
- **Modules (called “dmods”) can be added**
  - v8.so module for debugging the node engine
- **Can be used for live and post-mortem debugging**
- **Can be used with executable files, core files, and can be used to examine binary data**
- **See `mdb(1)`, and http://illumos.org/books/mdb/preface.html**

Using mdb with Node.js

Joyent

```
# pgrep node
5816
# gcore 5816   <- get a “live” dump, this can be skipped if core file already exists
gcore: core.5816 dumped
# mdb core.5816
Loading modules: [ libumem.so.1 libc.so.1 ld.so.1 ]
> ::load v8.so
V8 version: 3.11.10.22
Autoconfigured V8 support from target
C++ symbol demangling enabled
>
```

Using mdb with Node.js

Joyent

```
> ::dcmds !grep js
findjsobjects                 - find JavaScript objects
jsframe                       - summarize a JavaScript stack frame
jsprint                       - print a JavaScript object
jsstack                       - print a JavaScript stacktrace
> ::help jsprint

NAME
  jsprint - print a JavaScript object

SYNOPSIS
  addr ::jsprint [-a] [-d depth]
...
>
```

Using mdb with Node.js - `jsstack`

Joyent

```
# mdb core.node.78499  <- core from segmentation violation
> ::jsstack -v
80428bc 0xfa175010 onread (fc5f9339)
    file: net.js
    posn: line 347
    arg1: fe85e061 (Oddball: "undefined")
    arg2: fe85e061 (Oddball: "undefined")
    arg3: fe85e061 (Oddball: "undefined")
80428e0 0xfe840841 <ArgumentsAdaptorFrame>
...
8042c18 node::StreamWrap::OnReadCommon+0x183
8042c48 node::StreamWrap::OnRead+0x2e
8042d08 uv__read+0x37a
8042d38 uv__stream_io+0x128
8042d68 ev_invoke_pending+0x7b
8042db8 ev_run+0x406
8042dd8 uv_run+0x1c
8042e28 node::Start+0xac
8042e48 main+0x1b
8042e64 _start+0x83
```

Using mdb with Node.js - `jsfindobjects`

Joyent

```
> ::findjsobjects
OBJECT   #OBJECTS #PROPS CONSTRUCTOR: PROPS
fb647a49         1      1 Object:
fb55bf4cb0ceb5d57f673bc8063907e3
fb647a3d         1      1 Object:
c43220fdc5d361dc15836b663cf2b40c
fb645a75         1      1 Object:
ba1f6c296dcfa8d18016e6750c6af15e
...
f97308e5       989      4 TCP: , onread, socket,
writeQueueSize
...
fbf5e1a1      2607      5 HTTPParser: socket, _url, incoming,
onIncoming, ...
f972a2ad      2675     16 ServerResponse: connection, socket,
_trailer, ...
f972a089      2680     16 IncomingMessage: httpVersion,
connection, method, ...
f972a3cd      2684      5 Object: search, query, href,
pathname, path
f972a369      2684     15 Client: session, timing,
remote_address, ...
```

Using mdb with Node.js - `jsprint`

Joyent

```
> fb647a49::jsprint
{
    fb55bf4cb0ceb5d57f673bc8063907e3: {
        blocklist: [...],
        updates_channel: [...],
        post_obj: [...],
        user_create_time: 1.311442e+09,
        user_id: marco.meinardi.123456789,
        touch_count: 20,
        system_name: android,
        session_key: ffffffffffffffffffffffffffffffff,
        device_token: XXXXXXXXXXXXXXXXXXXX-...
        last_touch: ,
        create_time: ,
        version: 0.9.9,
    },
}
```

Using mdb with Node.js - jsprint (Continued)

Joyent

```
> f972a089::findjsobjects | ::jsprint !grep url
    url: /2/cs/start_session,
    url: /1/rr/heartbeat?from=prod-nr1352.xxx.com,
    url: /1/rr/heartbeat?from=prod-nr1122.xxx.com,
    url: /1/rr/heartbeat?from=prod-nr1382.xxx.com,
    url: /2/cs/start_session,
    url: /public_profiles?
cv=0.9.7.3.0004&operation_id=1350050151386_2653370263_bbd6afd8&Rv
    url: /header_object?
cv=0.9.9.0083&operation_id=1350050066161_4247255511_4227a074&mess
age_id=profile&Rv_session_key=xxx
...
```

References

Joyent

- https://github.com/mcavage/node-restify
- http://mcavage.github.com/presentations/dtrace_conf_2012-04-03/
- https://github.com/chrisa/node-dtrace-provider
- http://dtrace.org/blogs/blog/category/node-js/
- http://dtrace.org/blogs/dap/files/2012/05/fluent.pdf
- http://dtrace.org/blogs/bmc/2010/08/30/dtrace-node-js-and-the-robinson-projection/
- http://dtrace.org/blogs/dap/2012/01/05/where-does-your-node-program-spend-its-time/
- http://dtrace.org/blogs/brendan/2011/09/26/observing-observer-a-cloud-analytics-case-study/
- http://nodestack.org/videos <- Bryan Cantrill, Stack Foundation = SmartOS
- https://hacks.mozilla.org/2012/11/tracking-down-memory-leaks-in-node-js-a-node-js-holiday-season/

Acknowledgements

Joyent

- Thanks to nodejsconf.it, Joyent Engineering (Bryan Cantrill, Mark Cavage, Robert Mustacchi, Dave Pacheco, and others), Marco Meinardi of Joyent
- Slides, node.js scripts, and D scripts are on https://github.com/max123/nodejsconf2012.git
- Thanks for listening!
- max@joyent.com, @mrbruning, mbruning.blogspot.com