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

Experiences with DTrace development: stddev() and brendan()

Chad Mynhier
OpenSolaris DTrace developer

Outline

- stddev() -- first external contribution to DTrace -- put back into build 84
- brendan() -- RFE 4012008
- DTrace Test Suite

DTrace refresher

- D scripting language:

```
syscall::entry
/ uid == 0 /
{
        @[probefunc] = count();
}
```

- Actions – record data or change state

stddev()

- Good introduction to DTrace development
- Kernel and user-land components
- The problem has a twist that makes it harder than it looks at first.

Modeled after avg()

- usr/src/uts/common/dtrace/dtrace.c

```
static void
dtrace_aggregate_avg(uint64_t *data, uint64_t nval,
    uint64_t arg)
{
        data[0]++;
        data[1] += nval;
}
```

stddev() implementation

- sqrt (avg(x^2) – avg(x)^2)

```
static void
dtrace_aggregate_stddev(uint64_t *data, uint64_t nval,
    uint64_t arg)
{
        data[0]++;
        data[1] += nval;
        data[2] += nval * nval;
}
```

Options for handling overflow

- Ignore it
- Report it
- Avoid it

Avoiding overflow

- Implemented 128-bit arithmetic in the kernel and libdtrace
- Why not arbitrary precision?
- Why not 3rd-party library?

DTrace internals I

- DTrace uses a virtual machine in the kernel
- DIF: DTrace Intermediate Format
  - RISC-like instruction set
  - D scripts compile to this instruction set
- DIFO: DTrace Intermediate Format Object
  - A DIF object contains the compiled DIF for a D expression, its return type, and string and variable tables
  - Everything necessary to evaluate a D expression

DTrace Internals II

- ECB: Enabling Control Block
  - Each probe point has an attached list of ECB's.
  - The ECB contains the predicate DIFO and list of action DIFO's.
  - When probe fires, DTrace processes list of ECB's.
  - If predicate DIFO evaluates to non-zero, action DIFO's are executed.

brendan(): background

- April Fool's joke inspired by dtrace.conf(08)
- Scripts from DTrace Toolkit by Brendan Gregg contain this probe:

```
BEGIN { printf("Tracing... Hit Ctrl-C to end.\n"); }
```

- replace with:

```
BEGIN { brendan(); }
```

_dtrace_globals[]

- usr/src/lib/libdtrace/common/dt_open.c

```
{ "avg", DT_IDENT_AGGFUNC, 0, DTRACEAGG_AVG,
    DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "void(@)" },
{ "basename", DT_IDENT_FUNC, 0, DIF_SUBR_BASENAME,
    DT_ATTR_STABCMN, DT_VERS_1_0,
        &dt_idops_func, "string(const char *)" },
{ "brendan", DT_IDENT_ACTFUNC, 0, DT_ACT_BRENDAN,
    DT_ATTR_STABCMN, DT_VERS_1_7,
        &dt_idops_func, "void()" },
```

Compiling the function I

- usr/src/lib/libdtrace/common/dt_cc.c

```
static void
dt_action_exit(dtrace_hdl_t *dtp, dt_node_t *dnp,
    dtrace_stmtdesc_t *sdp)
{
        dtrace_actdesc_t *ap = dt_stmt_action(dtp, sdp);

        dt_cg(yypcb, dnp->dn_args);
        ap->dtad_difo = dt_as(yypcb);
        ap->dtad_kind = DTRACEACT_EXIT;
        ap->dtad_difo->dtdo_rtype.dtdt_size = sizeof (int);
}
```

Compiling the function II

```
static void
dt_action_brendan(dtrace_hdl_t *dtp, dt_node_t *dnp,
    dtrace_stmtdesc_t *sdp)
{
        dtrace_actdesc_t *ap = dt_stmt_action(dtp, sdp);

        ap->dtad_kind = DTRACEACT_BRENDAN;
        ap->dtad_arg = 0;
}
```

Consuming the action

- usr/src/lib/libdtrace/common/dt_consume.c

```
static int
dt_consume_cpu( ... )
{
        for (offs = start; offs < end; ) {
                dtrace_eprobedesc_t *epd;
                for (i = 0; i < epd->dtepd_nrecs; i++) {
                        dtrace_actkind_t act = rec->dtrd_action;
                        if (act == DTRACEACT_BRENDAN) {
                                if (dt_printf(dtp, fp, "Tracing... Hit Ctrl-C to end.\n") < 0)
                                        return (-1);
                                goto nextrec;
                        }
```

dtrace_probe()

- usr/src/uts/common/dtrace.c

```
void
dtrace_probe( ... )
{
        for (ecb = probe->dtpr_ecb; ecb != NULL; ecb = ecb->dte_next) {
                if (pred != NULL) {
                        dtrace_difo_t *dp = pred->dtp_difo;
                        rval = dtrace_dif_emulate(dp, &mstate, vstate, state);
                        if (!rval) { continue; }
                }
                for (act = ecb->dte_action; act != NULL; act = act->dta_next) {
                        switch (act->dta_kind) {
                        case DTRACEACT_BRENDAN:
                                continue;
```

dtrace_ecb_action_add()

- usr/src/uts/common/dtrace.c

```
static int
dtrace_ecb_action_add(dtrace_ecb_t *ecb, dtrace_actdesc_t *desc)
{
        switch (desc->dtad_kind) {
        case DTRACEACT_BRENDAN:
                size = sizeof (uint32_t);
                break;

        action->dta_rec.dtrd_size = size;
```

DTrace Test Suite

- Integrated with DTrace code
- Simple to write unit tests
- Simple to run

Test suite driver

- Very simple
- Two rules:
  - If it's a ksh script, run it with ksh; if it's a D script, run it with dtrace.
  - If it has an output file, compare the output; otherwise look at the return value.
- Unit tests grouped in subdirectories, driver finds them all

Test suite example

- Test script

```
BEGIN
{
        brendan();
        exit(0);
}
```

- Output file

```
Tracing... Hit Ctrl-C to end.
```

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

Thank you!

Chad Mynhier
cmynhier@gmail.com

“open” artwork and icons by chandan:
http://blogs.sun.com/chandan