bpftrace Dynamic kernel tracing

CONCEPTS

> Needs root (or CAP_BPF/ CAP_PERFMON) and a kernel with BPF. Maps you fill with @ are printed automatically when the program ends.

RUNNING

bpftrace -e '*<program>***'**
Run an inline program

bpftrace *<file.bt>*
Run a script file

bpftrace -l '*<pattern>***'**
List probes, wildcards allowed

bpftrace -lv '*<probe>***'**
List a probe with its arguments

bpftrace -p *<pid>* **-e '***<program>***'**
Attach to or filter by a PID

bpftrace -c '*<cmd>***' -e '***<program>***'**
Run a command and trace it

PROBES

kprobe:*<func>*
Kernel function entry

kretprobe:*<func>*
Kernel function return (retval)

tracepoint:*<cat>***:***<name>*
Static kernel tracepoint

uprobe:*<path>***:***<func>*
User function entry

begin { ... } / end { ... }
Run once at start / at exit

profile:hz:*<rate>*
Sample on every CPU at a rate

interval:s:*<n>*
Fire once per interval, on one CPU

BUILT-INS

pid / tid
Process / thread ID

BUILT-INS (cont.)

comm
Process name

arg0 ... arg5
Raw function arguments

retval
Return value in return probes

nsecs
Timestamp in nanoseconds

args.*<field>*
Tracepoint or fentry argument

MAPS AND AGGREGATION

@ = count()
Count events

@[*<key>***] = count()**
Count per key (comm, pid, probe)

@[*<key>***] = sum(***<n>***)**
Sum a value per key

@[*<key>***] = avg(***<n>***)**
Average per key

@ = hist(*<n>***)**
Power-of-2 histogram

@ = lhist(*<n>***, *<min>*, *<max>*, *<step>*)**
Linear histogram

@start[tid] = nsecs
Remember a timestamp per thread

OUTPUT AND FUNCTIONS

printf("*<fmt>***\n", *<args>*)**
Formatted output

str(*<ptr>***)**
Read a C string

LANGUAGE

probe /*<predicate>***/ { ... }**
Run the action only if true