Kernels and Tracing

Lecture 2, Part 2: The Probe Effect

Prof. Robert N. M. Watson

2023-2024

The *Probe Effect*

- The **probe effect** is the unintended alteration of system behaviour that arises from measurement
  - Software instrumentation is **active**: execution is changed
- DTrace minimises probe effect when unused…
  - ... but has a very significant impact when it is used
  - Disproportionate effect on probed events
- Potential perturbations:
  - Speed relative to other cores (e.g., lock hold times)
  - Speed relative to external events (e.g., timer ticks)
  - Microarchitectural effects (e.g., cache, branch predictor)

Probe effect example: dd(1) execution time

- Simple (naïve) microbenchmark – `dd(1)`
  - dd copies blocks from input to output
  - Copy 10M buffer from `/dev/zero` to `/dev/null`
  - (“Do nothing .. But do it slowly”)
  - Execution time measured with `/usr/bin/time`
  - **Workload chosen to illustrate high overhead**

```
# dd if=/dev/zero of=/dev/null bs=10m count=1 status=none
```

- Simultaneously, run various DTrace scripts
  - Compare resulting execution times using `ministat`
  - Difference is probe effect (+/- measurement error)

Probe effect 1: memory allocation

- Using the `dtmalloc` provider, count kernel memory allocations:

```
dtmalloc:::
{ @count = count(); }
```

```
x no-dtrace
+ dtmalloc-count
+--------------------------------------------------------------------------------+
|                                       *                                        |
|                                       *                                        |
|                                       *                                      + |
|x                                      *                                      + |
|x                                      *                                      + |
|*                                      *                                      * |
|        |_______________|______A______M__________A_____|__________________|     |
+--------------------------------------------------------------------------------+
    N           Min           Max        Median           Avg        Stddev
x  11           0.2          0.22          0.21    0.20818182  0.0060302269
+  11           0.2          0.22          0.21    0.21272727  0.0064666979
No difference proven at 95.0% confidence
```

- **No statistically significant overhead** at 95% confidence level

Probe effect 2: locking

- Using the `lockstat` provider, track kernel lock acquire, release:

```
lockstat:::
{ @count = count(); }
```

```
x no-dtrace
+ lockstat-count
+------------------------------------------------------------------------------+
|   x                                                                        +|
|   x                                                                        +|
|   x                                                                    +   +|
|x  x                                                                    +   +|
|x  x                                                                    +   +|
|x  x  x                                                              +  +   +|
| |_A_|                                                                   |_A_M|
+------------------------------------------------------------------------------+
    N           Min           Max        Median           Avg        Stddev
x  11           0.2          0.22          0.21    0.20818182  0.0060302269
+  11          0.42          0.44          0.44    0.43454545  0.0068755165
Difference at 95.0% confidence
        0.226364 +/- 0.00575196
        108.734% +/- 2.76295%
        (Student's t, pooled s = 0.0064667)
```

- **109% overhead** – 170K locking operations vs. 6 `malloc()` calls!

Probe effect 3: limiting to dd(1)?

- Limit the `action` to processes with the name dd:

```
lockstat::: /execname == "dd"/
{ @count = count(); }
```

```
x no-dtrace
+ lockstat-count-dd
+------------------------------------------------------------------------------+
|                                                                          +   |
|  x                                                                       +   |
|  x                                                                       +   |
|  x                                                                       +   |
|  x                                                                       +   |
|x x                                                                       +   |
|x x                                                                       +   |
|x x x                                                                 + + + +|
||_A|                                                                   |_A|  |
+------------------------------------------------------------------------------+
    N           Min           Max        Median           Avg        Stddev
x  11           0.2          0.22          0.21    0.20818182  0.0060302269
+  11          0.54          0.57          0.56    0.55818182  0.0075075719
Difference at 95.0% confidence
        0.35 +/- 0.0060565
        168.122% +/- 2.90924%
        (Student's t, pooled s = 0.00680908)
```

- Well, crumbs. Now **168% overhead**!

Probe effect 4: stack traces

- Gather more locking information in `action` – capture call stacks:

```
lockstat::: { @stacks[stack()] = count(); }
lockstat::: /execname == "dd"/ { @stacks[stack()] =  count(); }
```

```
x no-dtrace
+ lockstat-stack
* lockstat-stack-dd
+------------------------------------------------------------------------------+
|                                                                          *    |
|                                                                          *    |
|                                                                          *    |
|                                                                          *    |
|xx                                                                   ++   **   |
|xx                                                                   ++   **   |
|xx                                                                +  +++ ** **+|
|AM                                                                   |_MA_|A|  |
+------------------------------------------------------------------------------+
    N           Min           Max        Median           Avg        Stddev
x  11           0.2          0.22          0.21    0.20818182  0.0060302269
+  11          1.38          1.57          1.44     1.4618182   0.058449668
        1.25364 +/- 0.0369572
        602.183% +/- 17.7524%
*  11           1.5          1.55          1.51     1.5127273   0.014206273
        1.30455 +/- 0.00970671
        626.638% +/- 4.66261%
```

What does this mean for us?

- Always think about the potential role of the probe effect when instrumenting a workload
  - E.g., avoid benchmarking while running DTrace ...
  - ... unless **measuring or accounting for the probe effect**
- Traced applications may behave (very) differently
  - E.g., more timer ticks will fire, affecting thread inverleaving
  - E.g., I/O will “seem faster” relative to computation, as latter may slow down due to probe effect
- Performance overheads may be disproportionate
  - E.g., if you instrument one way of doing things, but not another, and workloads have a different functional footprint
- Consider ways to decide if an analysis is representative
  - E.g., are the performance inflection points consistent even if absolute performance is lower?