

# L41: Lab 1- I/O

Lecturelet 1

Dr Robert N. M. Watson and Dr Graeme Jenkinson

30 October 2017

## L41: Lab 1 – I/O

- Introduce our experimental environment:
  - BeagleBone Black
  - FreeBSD operating system + DTrace
  - I/O benchmark
  - Jupyter notebooks
- Explore user-kernel interactions via syscalls and traps
- Learn a bit about POSIX I/O
- Measure the probe effect

L41 Lecturelet 1- Lab 1 I/O

## The platform



### BeagleBone Black

- 1GHz ARM Cortex A-8 32-bit CPU
- 512MB DDR3 RAM
- Superscalar pipeline, MMU, L1/L2 caches
- FreeBSD operating system + DTrace
- “potted benchmarks”

L41 Lecture 1 - Advanced Operating Systems

## DTrace scripts

- Human-facing C-like language
- One or more {**probe name**, **predicate**, **action**} tuples
- Expression limited to control side effects (e.g. no loops)
- Specified on the command line or via a `.d` file

```
fbt::malloc:entry /execname == "csh"/ {trace(arg0);}
```

<b>Probe name</b>	Identifies the probe(s) to instrument; wildcards allowed; identifies the <b>provider</b> and a provider-specific <b>probe name</b>
<b>Predicate</b>	Filters cases where action will execute
<b>Action</b>	Describes tracing operations

L41 Lecturelet 1 - Lab 1 I/O

## Some kernel DTrace providers in FreeBSD

Provider	Description
<code>callout_execute</code>	Timer-driven callouts
<code>dtmalloc</code>	Kernel <code>malloc()</code> / <code>free()</code>
<b>dtrace</b>	DTrace script events (BEGIN, END)
<b>fbt</b>	Function Boundary Tracing
<b>io</b>	Block I/O
<code>ip, udp, tcp, sctp</code>	TCP/IP
<code>lockstat</code>	Locking
<b>proc, sched</b>	Kernel process/scheduling
<code>profile</code>	Profiling timers
<b>syscall</b>	System call entry/return
<b>vfs</b>	Virtual filesystem

L41 Lecturelet 1- Lab 1 I/O

## Aggregations

Aggregation	Description
<code>count()</code>	Number of times called
<code>sum()</code>	Sum of arguments
<code>avg()</code>	Average of arguments
<code>min()</code>	Minimum of arguments
<code>max()</code>	Maximum of arguments
<code>stddev()</code>	Standard deviation of arguments
<code>lquantize()</code>	Linear frequency distribution (histogram)
<code>llquantize()</code>	Log-linear frequency distribution (histogram)
<code>quantize()</code>	Log frequency distribution (histogram)

- Often we want summaries, not detailed traces. DTrace allows early, efficient **reduction** using aggregations
- Scalable multicore implementations (i.e. commutative)
- `@variable = function()`
- `printa(@variable)` to print

L41 Lecturelet 1- Lab 1 I/O

## Counting kernel `read()` system calls

```
$ ./io-static -q -r /data/iofile
$ dtrace -n
'syscall::read:entry
/execname=="io-static"/
{@reads = count(); }'
```

<b>Probe name</b>	Trace the <code>read()</code> system call
<b>Predicate</b>	Limit actions to processes executing <code>io-static</code>
<b>Action</b>	Count the number of probe fires

```
dtrace: description 'syscall::read:entry ' matched 1
probe dtrace: buffer size lowered to 2m dtrace:
aggregation size lowered to 2m
^C
```

1024

L41 Lecturelet 1- Lab 1 I/O

## The benchmark

```
$ ./io-static
io-static -c|-r|-w [-Bdqsv] [-b blocksize] [-t totalsize] path

Modes (pick one):
-c          'create mode': create benchmark data file
-r          'read mode': read() benchmark
-w          'write mode': write() benchmark

Optional flags:
-B          Run in bare mode: no preparatory activities
-d          Set O_DIRECT flag to bypass buffer cache
-q          Just run the benchmark, don't print stuff out
-s          Call fsync() on the file descriptor when complete
-v          Provide a verbose benchmark description
-b blocksize Specify a block size (default: 16384)
-t totalsize Specify total I/O size (default: 16777216)
```

- Simple, bespoke I/O benchmark: `read()` or `write()`
- Statically or dynamically linked
- Adjust buffer sizes, etc.
- Various output modes

L41 Lecturelet 1- Lab 1 I/O

## The benchmark (2)

- Three operational modes
  - Create (-c) Create a new benchmark data file
  - Read (-r) Perform `read()`s against data file
  - Write (-w) Perform `writes()`s against data file
- Adjust I/O parameters:
  - Block size (-b) Block size used for each I/O
  - Total size (-t) Total size across all I/Os
  - Direct (-d) Use direct I/O (bypass buffer cache)
  - Sync (-s) Perform `fsync()` after I/O loop
  - Bare (-b) Don't synchronise cache (etc) on start (whole-program testing)
- Output flags:
  - Quiet (-q) Suppress all output (whole-program tracing)
  - Verbose (-v) Verbose output (interactive testing)

L41 Lecturelet 1- Lab 1 I/O

## The benchmark (3)

```
$ ./io-static -v -d -w /data/iofile
Benchmark configuration:
  blocksize: 16384
  totalsize: 16777216
  blockcount: 1024
  operation: write
  path: /data/iofile
  time: 58.502746875
280.06 KBytes/sec
```

- Use verbose output (-v)
- Bypass the buffer cache (-d)
- Write (-w) to the previously created file `/data/iofile`
- Use default buffer size (16K) and total I/O size (16M)

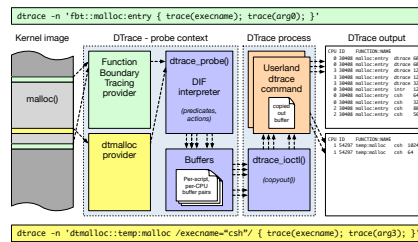
L41 Lecturelet 1- Lab 1 I/O

## Probe effect

- Probe effect - act of measuring disturbs system
- Electronics - probes introduce additional capacitance, resistance or inductance
- Software tracing - probes take time to execute
  - Don't benchmark while running Dtrace ...
  - ... unless **measuring probe effect**
  - Be aware that traced applications may behave differently
  - E.g., more timer ticks will fire, I/O will "seem faster"



Zero when disabled



Definitely not zero

L41 Lecturelet 1- Lab 1 I/O

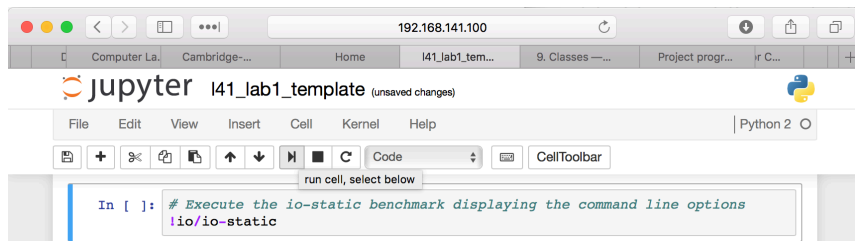
## Jupyter notebooks

Unified environment for:

- Executing benchmarks.
- Instrumenting the behaviours and performance of benchmarks using DTrace.
- Post-processing performance measurements.
- Plotting performance measurements.
- Performing statistically analysis on performance measurements.

L41 Lecturelet 1- Lab 1 I/O

## Jupyter notebooks (2)



- Series of cells containing Python or cell “magics”.
- Cell magics allow, for example, inline plotting of graphs or executing shell commands.
- Raw data and plots can be saved to the BBB for inclusion in laboratory reports.
- Details of experimental environment in lab setup handout.

L41 Lecturelet 1- Lab 1 I/O

## Hypotheses

Larger I/O and IPC buffer sizes amortize system-call overheads

A purely architectural (SW) view dominates

The probe effect doesn't matter in real workloads

L41 Lecturelet 1- Lab 1 I/O

## Experimental questions for the lab report

- With respect to a configuration **reading** from a fixed-size file through the buffer cache:
  - How does changing the I/O buffer size affect I/O-loop performance?
  - How does static vs. dynamic linking affect whole-program performance?
  - At what file-size threshold does any performance difference between static and dynamic linking fall below 5%? 1%?
- Run the benchmark to gather initial measurements
- Explore through system-call/trap tracing and profiling
- Use various configurations (e.g., I/O on `/dev/zero`) to explore kernel code-path behaviour
- Ensure that you directly consider the impact of the **probe effect** on your causal investigation

L41 Lecturelet 1- Lab 1 I/O

## A few cautions

There are two kinds of people, those that have experienced data loss and those that haven't experienced data loss **YET**.

- The SD cards seem a bit fragile during power off – make sure that you shut down safely using the laboratory setup instructions.
  - We have spare imaged SD cards if you need them.
- Backup key scripts and data files on your workstation
  - We may replace your SD cards for future labs.

L41 Lecturelet 1- Lab 1 I/O



## A few other useful things

- Feel free to work in pairs or groups in the lab (reports must be written separately).
- You will likely want multiple SSH sessions open.
- The kernel source code is in github:  
`freebsd/freebsd.git` (branch `release/11.0.0`).
- Experiment on the command line:
  - Start with something simple – e.g., `DTrace hello world`.
- Do not hesitate to ask for help if you need a hand.

L41 Lecturelet 1- Lab 1 I/O