Behaviour Analysis and Visualization
Linux Plumbers Conference
KP Singh
Agenda

- Trace Analysis and Visualization
- Behaviour Analysis
- Example Behaviour
- Demo
The Tools!

- LISA: Linux Integrated Systems Analysis
- TRAPpy: Trace Analysis and Plotting in Python
- BART: Behavioural Analysis and Regression Toolkit
Workflow

LISA
Run workloads and collect traces

TRAPpy
Parse traces and visualize

BART
Define Behaviours and assert
**Trace Visualization**

**Parse**

Predefined kernel events and custom `trace_printk` events.

**Visualize**

Interactive plotting API which allows filtering, grouping and comparing data.

**Analyse**

The data is parsed to pandas DataFrames which opens it to a whole new analysis methodologies.
Trace Parsing and Plotting

\[
\text{trace_printk("my_event: key1=%s key2=%d", val1, val2)}
\]

<table>
<thead>
<tr>
<th>Time</th>
<th>key1</th>
<th>key2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.020</td>
<td>Linux</td>
<td>2</td>
</tr>
<tr>
<td>1.342</td>
<td>Plumbers</td>
<td>0</td>
</tr>
<tr>
<td>1.451</td>
<td>Conference</td>
<td>1</td>
</tr>
<tr>
<td>1.919</td>
<td>Santa Fe</td>
<td>6</td>
</tr>
</tbody>
</table>

Interactive Line Plots

Timelines
Behaviour Analysis

- For kernel developers and continuous integration.

- Visualize Behaviours.

- Automate Behaviour Analysis.
Why do we need Behaviours?

Behaviours are the building blocks/premises for performance

"If the scheduler exhibits these four behaviours, it will have the best power performance ratio"
How to do it?

Define and calibrate a behaviour, automate for regression analysis. Adjust in the future with new data.

“Oh! my new patch breaks this behaviour and this means my device will use more energy”
Topologies

BIG = [4, 5]
LITTLE = [0, 1, 2, 3]
clusters = [BIG, LITTLE]

topology = Topology(clusters=clusters)

s = SchedMultiAssert(trace_dir, topology, execnames=[...])

Not only for the scheduler, topologies are hidden everywhere:

- File-systems (VFS, LVM, Block IO, RAID)
- Thermal Zones (Sensor hierarchy)
- Network
What is a Behaviour?

A behaviour is what a kernel developer would have in mind while working on a particular feature.

"I want all the light weight tasks to be packed on the little cluster..."

"I want the I/O scheduler to starve writes for reads"
Aggregation

- CPU1
- CPU2
- Cluster

Switch In
Switch Out
Example Behaviour

"All light weight tasks should be packed on the little cluster"

Quantify

- Light weight task = 10% duty cycle on the BIG CPU
- Number of light weight tasks = num_little_cpus

Success Criteria:

assertResidency(LittleCluster >75%)
Visualize

In [34]: trappy.plotter.plot_trace(trace, execnames=tasks.keys())
Metrics

```
littleCluster = [0, 1, 2, 3]
getResidency('cluster', littleCluster)
```
Scheduler API

window = [tStart, tEnd]

getResidency('cluster', LITTLE, window)

getDutyCycle(window)

assertSwitch('cluster', BIG, LITTLE, window)

assertResidency('cluster',
                BIG,
                comparator,
                expectedValue,
            )

assertDutyCycle(...)
Generic API

```python
assertStatement("mean(THERMAL:temp) < CONTROL_TEMP")
assertStatement("stdev(sched_load_avg:load) < THRESHOLD")
```
Demo