A Methodology and Framework to Determine the Isolation Capabilities of Virtualisation Technologies

Simon Volpert, Benjamin Erb, Georg Eisenhart, Daniel Seybold, Stefan Wesner, Jörg Domaschka

ICPE ‘23
Coimbra, Portugal
Motivation

Tenant A
- CPU
- Memory
- Network
- Disk

Tenant B
- CPU
- Memory
- Network
- Disk

Tenant n

Operator

A Methodology and Framework to Determine the Isolation Capabilities of Virtualisation Technologies | Volpert
System Model

- Memory
- CPU
- I/O Bridge
- Disk
- NIC
Virtualization Model

- **Hypervisor based** [15]
  Type 1&2, Paravirtualization, Hardware-assisted- & Full-virtualization

- **Container Based** [14]
  Cgroups (CPU, Memory, ...), namespaces (PID, Network, Mount, ...), capabilities

- **Sandbox Based** [43]
  System call filtering

- **Hybrid**
  Arbitrary combinations of the above

---

Virtualization Model

Virtualization

- Hypervisor based
- Container based
- Sandbox based

Hybrid
Research Questions

▸ RQ1 which benchmarks are suitable for driving such an evaluation and which resources should be considered?

▸ RQ2 which measurement technologies are available to support measuring isolation for a wide range of virtualisation technologies?

▸ RQ3 which evaluation methodology reduces disturbances and increases repeatability?
Isolation Measurement Methodology - Requirements

- **R1** Isolation Measurement
  Measurements of isolation by applying a sensible isolation determination model

- **R2** Load Generation
  Flexible generation of very specific load

- **R3** Data Acquisition
  Acquisition of data independent of virtualization technology and load generation

- **R4** Reproducibility
  Experiments need to be reproducible on a given system

- **R5** Automation
  Capabilities for automation
R1 Isolation Measurement Methodology - Quantification

- Many different models in academia
- Goal: measure the performance loss for a specific static workload

Measure the “Performance Loss Rate” \([20, 39, 44]\)

\[ I_{plr} = \frac{|p_a - p_b|}{p_a} \]

- \(p_a\): Baseline Performance
- \(p_b\): Contended Performance

R1 Isolation Measurement Methodology - Utilization

- **Resources**
  - CPU, Memory, Disk I/O, Memory I/O

- **Calculation of capacity based utilization per resource**

\[
U_c = \frac{c_b}{c_b + c_i} \quad \quad U_m = \frac{m_u}{m_u + m_f} \quad \quad U_n = \frac{n_a}{n_m} \quad \quad U_d = \frac{iops_a}{iops_m}
\]
R1 Measurement Scenario - Baseline

- a runs workload below limit
  undercommitted
- a runs workload at limit
  saturated
- a runs workload above limit
  overcommitted
- a runs workload without limit
  unrestricted
R1 Measurement Scenario - Contended

- \( a \) runs workload below limit
  undercommitted
- \( a \) runs workload at limit
  saturated
- \( a \) runs workload above limit
  overcommitted
- \( a \) runs workload without limit
  unrestricted

- Rerun each step with \( b \)
  undercommitted, saturated, overcommitted, unrestricted

Determine “Performance Loss Ratio” for every scenario
R1 Experiment Scenarios

<table>
<thead>
<tr>
<th>number</th>
<th>shortname</th>
<th>tenant a</th>
<th>tenant b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(a_b)</td>
<td>undercommitted</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(a_b)</td>
<td>saturated</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(a_b)</td>
<td>overcommitted</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(a_b)</td>
<td>unrestricted</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(a_u b_u)</td>
<td>undercommitted</td>
<td>undercommitted</td>
</tr>
<tr>
<td>6</td>
<td>(a_u b_z)</td>
<td>undercommitted</td>
<td>saturated</td>
</tr>
<tr>
<td>7</td>
<td>(a_u b_o)</td>
<td>undercommitted</td>
<td>overcommitted</td>
</tr>
<tr>
<td>8</td>
<td>(a_u b_f)</td>
<td>undercommitted</td>
<td>unrestricted</td>
</tr>
<tr>
<td>9</td>
<td>(a_s b_u)</td>
<td>saturated</td>
<td>undercommitted</td>
</tr>
<tr>
<td>10</td>
<td>(a_s b_z)</td>
<td>saturated</td>
<td>saturated</td>
</tr>
<tr>
<td>11</td>
<td>(a_s b_o)</td>
<td>saturated</td>
<td>overcommitted</td>
</tr>
<tr>
<td>12</td>
<td>(a_s b_f)</td>
<td>saturated</td>
<td>unrestricted</td>
</tr>
<tr>
<td>13</td>
<td>(a_o b_u)</td>
<td>overcommitted</td>
<td>undercommitted</td>
</tr>
<tr>
<td>14</td>
<td>(a_o b_z)</td>
<td>overcommitted</td>
<td>saturated</td>
</tr>
<tr>
<td>15</td>
<td>(a_o b_o)</td>
<td>overcommitted</td>
<td>overcommitted</td>
</tr>
<tr>
<td>16</td>
<td>(a_o b_f)</td>
<td>overcommitted</td>
<td>unrestricted</td>
</tr>
</tbody>
</table>

- Scenario runtime ~30min
- 10 iterations per scenario
- Runtime per experiment ~ 35 hours
- 4 Experiments total (1 per resource)
R2 Load Generation

stress-ng

iperf3

fio

CPU

Memory

Network

Disk
R3 Data Acquisition

R4 Reproducibility

- Experiment as Code
- No Configuration Drift
- Immutability

(i) Hardware
(ii) OS
(iii) OS Config
(iv) Experiment Runtime
(v) Experiment
R5 Automation

Diagram showing the automation of workloads in a Kubernetes environment using Argo Workflow. Nodes 1, 2, and 3 are connected to workloads Pod 1 to Pod n with VV (verification validation) markers at each node.
Experiment Execution

1. Spawn
2. Execute
3. Profiling
4. Acquire
5. Store
Selected Measurement - Podman CPU saturated

<table>
<thead>
<tr>
<th>id</th>
<th>shortname</th>
<th>a cpu</th>
<th>a’ cpu</th>
<th>I_{ulr}</th>
<th>I_{plr}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a_u</td>
<td>12.47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>a_s</td>
<td>25.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>a_o</td>
<td>25.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>a_f</td>
<td>98.92</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>a_u b_u</td>
<td>12.47</td>
<td>12.48</td>
<td>0.00</td>
<td>0.06</td>
</tr>
<tr>
<td>6</td>
<td>a_u b_s</td>
<td>12.47</td>
<td>12.36</td>
<td>0.05</td>
<td>0.88</td>
</tr>
<tr>
<td>7</td>
<td>a_u b_o</td>
<td>12.47</td>
<td>12.35</td>
<td>0.06</td>
<td>1.03</td>
</tr>
<tr>
<td>8</td>
<td>a_u b_f</td>
<td>12.47</td>
<td>12.20</td>
<td>0.14</td>
<td>2.23</td>
</tr>
<tr>
<td>9</td>
<td>a_s b_u</td>
<td>25.00</td>
<td>24.91</td>
<td>0.05</td>
<td>0.37</td>
</tr>
<tr>
<td>10</td>
<td>a_s b_s</td>
<td>25.00</td>
<td>24.06</td>
<td>0.47</td>
<td>3.74</td>
</tr>
<tr>
<td>11</td>
<td>a_s b_o</td>
<td>25.00</td>
<td>23.48</td>
<td>0.76</td>
<td>6.07</td>
</tr>
<tr>
<td>12</td>
<td>a_s b_f</td>
<td>25.00</td>
<td>24.71</td>
<td>0.14</td>
<td>1.15</td>
</tr>
<tr>
<td>13</td>
<td>a_o b_u</td>
<td>25.00</td>
<td>25.00</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>14</td>
<td>a_o b_s</td>
<td>25.00</td>
<td>25.13</td>
<td>0.06</td>
<td>0.50</td>
</tr>
<tr>
<td>15</td>
<td>a_o b_o</td>
<td>25.00</td>
<td>25.08</td>
<td>0.04</td>
<td>0.30</td>
</tr>
<tr>
<td>16</td>
<td>a_o b_f</td>
<td>25.00</td>
<td>25.09</td>
<td>0.04</td>
<td>0.33</td>
</tr>
</tbody>
</table>
Selected Measurement - Podman Network saturated

<table>
<thead>
<tr>
<th>id</th>
<th>shortname</th>
<th>a network</th>
<th>a' network</th>
<th>( I_{ulr} )</th>
<th>( I_{ptr} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( a_u )</td>
<td>10.00</td>
<td></td>
<td>0.05</td>
<td>0.50</td>
</tr>
<tr>
<td>2</td>
<td>( a_s )</td>
<td>50.12</td>
<td></td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>( a_o )</td>
<td>88.71</td>
<td></td>
<td>1.71</td>
<td>17.08</td>
</tr>
<tr>
<td>4</td>
<td>( a_f )</td>
<td>93.29</td>
<td></td>
<td>7.63</td>
<td>76.32</td>
</tr>
<tr>
<td>5</td>
<td>( a_u b_u )</td>
<td>10.00</td>
<td>10.05</td>
<td>0.13</td>
<td>0.25</td>
</tr>
<tr>
<td>6</td>
<td>( a_s b_s )</td>
<td>50.12</td>
<td>46.54</td>
<td>3.58</td>
<td>7.14</td>
</tr>
<tr>
<td>7</td>
<td>( a_o b_o )</td>
<td>50.12</td>
<td>28.38</td>
<td>21.74</td>
<td>43.37</td>
</tr>
<tr>
<td>8</td>
<td>( a_f b_f )</td>
<td>50.12</td>
<td>12.89</td>
<td>37.23</td>
<td>74.27</td>
</tr>
<tr>
<td>9</td>
<td>( a_u b_u )</td>
<td>88.71</td>
<td>79.20</td>
<td>9.52</td>
<td>10.73</td>
</tr>
<tr>
<td>10</td>
<td>( a_s b_s )</td>
<td>88.71</td>
<td>61.13</td>
<td>27.59</td>
<td>31.10</td>
</tr>
<tr>
<td>11</td>
<td>( a_o b_o )</td>
<td>88.71</td>
<td>51.99</td>
<td>36.72</td>
<td>41.39</td>
</tr>
<tr>
<td>12</td>
<td>( a_f b_f )</td>
<td>88.71</td>
<td>19.02</td>
<td>69.69</td>
<td>78.56</td>
</tr>
</tbody>
</table>
Selected Measurement - Podman Memory saturated
R3 Summary

- An evaluation methodology for the multidimensional evaluation of isolation capabilities and performance degradation. Addressing typical different types of hardware resources while being open regarding workload generation and further tooling.

- A proof-of-concept implementation of the methodology as a benchmark-based evaluation framework. With a strict focus on aspects such as reproducibility, automation, and fine grained profiling.

- A validation of the proof-of-concept implementation of the methodology measuring the isolation capabilities of podman representing a container-based virtualisation technology.
R3 Future Work

- Extend the system model to measure more resources
- Measure more virtualization technologies
- More complex benchmarks compared to micro-benchmarks
- Investigation and possibly compare further isolation models
- Release of the framework
Questions?

Simon Volpert
Institute of Information Resource Management
simon.volpert@uni-ulm.de
eBPF Landscape

Linux Events & BPF Support

Dynamic Tracing
- ext4:
- uprobes
- kprobes

Tracepoints
- Linux 4.7
- Operating System
- Applications
- System Libraries
- System Call Interface
- Scheduler

BPF output
- Linux 4.4
- BPF stacks
- Linux 4.9

Software Events
- CPU-clock
- CS migrations
- Page-faults
- Minor faults
- Major faults

syscalls:
- sched:
- task:
- signal:
- timer:
- workqueue:

PMCs
- Linux 4.9
- cycles
- instructions
- branch-
- L1-
- LLC-

CPU
- Interconnect
- kmem:
- vmscan:
- writeback:
- irq:

DRAM
- mem-load
- mem-store

Memory Bus

http://www.brendangregg.com/ebpf.html 2017
**R2 System Model - Utilization & Saturation**

![Diagram of Utilization and Throughput](image)

- **Utilization** vs **Load**
  - 0% to 100% Utilization
  - Saturation point

- **Throughput** vs **Load**
  - Trashing region
  - Knee point
  - Dashed line represents theoretical throughput
Automation - Argo Workflow Tree
Automation - Argo Workflow Process
Data Processing

1. Raw bpftrace data
2. bpftrace to json
3. json to csv
4. csv to dataframe
5. dataframe postprocessing
6. data visualization
7. data tables
Bpftrace Memory RSS Example

```
#!/usr/local/bin/bpftrace
#include <linux/sched.h>
#include <linux/mm.h>
BEGIN {
    @start = nsecs;
    print("timestamp,pid,mtype,bytes");
}
interval:ms:$SAMPLEMS
{
    $ts = (nsecs - @start)/1000;
    printf("%U",$ts);
    print(@);
}
tracepoint:kmem:rss_stat
/curtask->parent->parent->parent->pid == $ROOTPID/
{
    @[pid, args->member] = args->size;
}
END {
    clear(@);
    clear(@start);
}
```
Memory allocation over time