VirtuOS: An operating system with kernel virtualization

Authors:
- Ruslan Nikolaev
- Godmar Back

Presented in SOSP’13 on Nov 3-6, 2013

Presentation by Bien Aime MUGABARIGIRA
Process Isolation and protection

- To ensure **reliability and fault resilience**, application runs in its protected virtual address space

  Protect process processes from each other and kernel from misbehaving users processes

- **Problem:** How about kernel components?
  - What if kernel component fails?
  - 65-83% of windows xp crashes by device drivers
  - Others are from memory overruns, improper resources and protocols use, interrupt handling errors, race condition, deadlocks.
Motivation Of the paper

- **Protection & Isolation** of critical system components such as drivers
- Their failure may lead to entire system failure in monolithic OS design
- **Problem** is lack of protection from failing kernel components
- Solution: Hardware based virtualization
Motivation (cont..)

- **Protection & Isolation** of critical system components such as drivers
- **Their failure** may lead to entire system failure in monolithic OS design
- **Problem** is lack of protection from failing kernel components
- **Solution:** Hardware based virtualization
Motivation (cont)

- Problem is lack of **protection** from failing kernel component
  - Fail of one driver lead to entire system fail
  - This leads to rebooting the entire computer
- **Solution:** Increase the reliability of kernel software.

**Decomposition**

Hardware based virtualization
Proposed Solutions (Decomposition & Isolation)

- Split Monolithic system into **Separate** and **Isolated** components
  - Improves modularity
  - Contains faults
  - Improves reliability

- Decomposition have been used in other systems:
  - Microkernel-based systems (L3/L4, Minix-3…)
  - VM-based systems (DD/OS, Xen Driver Domains)

**Question**: How to provide flexible decomposition & strong isolation while retaining good performance, transparency and compatibility?

**VirtuOS design** uses **hardware-based** virtualization

**Challenge**: Provide flexible decomposition and strong isolation while retaining good performance, transparency and compatibility.
Features of the Proposed System:

**Static Analysis**

- Static analysis mainly focuses on the data that can be collected from an Android application executable .apk file.
- Analyze – Application code, Embedded strings, library imports.
- Analyze – Code structure for the known malware families.
- Analyze – Android application permission structure, Registered broadcast, Cryptographic functions, Hardcoded strings and IP address.
- Analyze - Code for accessing different critical data resources on the mobile device.
Flexibility, isolation, performance compatibility

- **Flexibility**: VirtuOS protects low- and high-level components
- **Isolation**: VirtuOS provides protection domain, privilege separation and Device protection
- **Performance**: VirtuOS performance is comparable to the performance of monolithic system for server workloads
- **Compatibility**: applications & systems codes
ARCHITECTURE

- VirtuOS isolates service domains
- Applies **vertical slice** of kernel functionality in **isolated service domain**
- Each service domain handles a **specific** kernel service
- Housing the drivers to access the physical devices
- User process interact directly with service domain through local system calls
**Architecture**

- **Goal**: improved isolation of kernel components in virtualized containers
- Service domain runs a near-stock version of kernel
- It handles requests coming from the user processes managed by the primary domain
- Includes: socket layer, TCP/IP implementation, device driver
- Implements user level M:N threading models, direct access to device
Architecture

- Fail model: software and hardware faults
  - Hardware faults: invalid DMA memory accesses or interrupt signaling error
  - Failure detected:
Architecture

- **Fail model**: software and hardware faults
- **Hardware faults**: invalid DMA memory accesses or interrupt signaling error
- **Failure detected**: service domain must be restarted using hypervisor domain
Fail model: software and hardware faults

Hardware faults: invalid DMA memory accesses or interrupt signaling error

Failure detected: service domain must be restarted using hypervisor domain

Affects only processes which have started using the failed domain

Server applications which use multiple OS processes can be restarted if failure occurs

Provide advantages compared to rebooting the primary domain or entire machine
Implementation

- Exceptionless protocol. Shared pages for request data, lock free request queues
- Threading: User process, context switch and M:N threading library
- Notification: Shared lock free ready queue: service domains directly access process’s ready queue
  - User thread is resumed, data copied and call is returned
- Spinning: Single threaded applications
Evaluation

Compatibility:

- System code compatibility - network and storage service domain implementations
- Application compatibility with existing applications, particularly server-based workloads

<table>
<thead>
<tr>
<th>Component</th>
<th>Number of lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backend/Frontend Driver</td>
<td>3115/2157</td>
</tr>
<tr>
<td>uClibc+NPTL/libaio</td>
<td>1152/2290</td>
</tr>
<tr>
<td>Linux kernel/Xen</td>
<td>1610/468</td>
</tr>
<tr>
<td>Total:</td>
<td>20792</td>
</tr>
</tbody>
</table>
Evaluation

System Configuration

- 2 x Intel Xeon E5520, 2.27GHz
- 2x4 cores (HyperThreading: OFF, TurboBoost: OFF)
- L1/L2 cache: 64K/256K per core
- L3 cache: 2x8MB
- Main memory: 12GB
- Network: Gigabit Ethernet, PCI Express
- HDD: SATA 7200RPM
Failure Recovery

- Network domain abruptly terminates
- Network domain is restarted
- Remote client starts transfer
- Remote client resumes transfer
Memory Copy Overhead

- VirtuOS (storage) vs. Linux
- 16 MB tmpfs transfer
OLTP/Sysbench mySQL
Apache

![Graph showing throughput in MB/s vs. concurrency for different configurations.]
Conclusion

- Isolation and Failure Recovery
  - VirtuOS provides isolation of vertical slices of a monolithic kernel into domains
  - Exploits strong isolation & device protection provided by hardware-based virtual machines
  - Supports separate failure & recovery of such domains

- Performance
  - VirtuOS uses fast interdomain communication to achieve good performance

- Compatibility
  - VirtuOS requires few changes to system code
  - Is transparent to POSIX applications
QUESTION & ANSWERS

Thank you