Variations of Process Abstractions

- "Solaris Zones: Operating System Support for Consolidating Commercial Workloads"
  2004 LISA XVIII – November 14-19, 2004 – Atlanta, GA

Problem

Within many IT organizations, driving up system utilization (and saving money in the process) has become a priority. In the lean economic times following the post dot-com downturn, many IT managers are electing to adopt server consolidation as a way of life. They are trying to improve on typical data center server utilizations of 15-30%

- Context:
  - Hardware supported virtualization was still restricted to specialized servers
  - Intel VT-x release 2005
  - Software virtualization had significant overheads
  - Memory footprint of multiple operating systems
  - Lack of sharing
  - Performance penalty for emulating I/O

Practical Barriers

- Server-class applications written assuming a machine to itself
  - Clashing network ports
  - Clashing user IDs
  - Hard-coded log/config file locations
- One application should not interfere with another

Security Issues

- Runs as ‘root’
  - How to run two mutually distrusting applications?
  - Administration requires root
  - What about mutually distrusting administrators?
- Root for one application environment should be less than root for the machine

Solaris Zones

- A baked in solution
  - Part of the operating system
- “Applications can be run within zones with no changes, and with no significant performance impact for either the performance of the application or the base operating system”
- Virtualises user-kernel boundary (not the hardware platform)

Interface Levels
Overview

Design Requirements

- Each zone can provide a rich (and different) set of customized services, and to the outside world, it appears that multiple distinct systems are available.
- Each zone has a distinct root password and its own administrator.

Design Requirements

- Basic process isolation;
  - A process in one non-global zone cannot locate, examine, or signal a process in another zone.
- Each zone is given access to at least one logical network interface;
  - applications running in distinct zones cannot observe the network traffic of the other zones even though their respective streams of packets travel through the same physical interface.
- Finally, each zone is provided a disjoint portion of the file system hierarchy, to which it is confined.

Design Requirements

- The global zone encloses the three non-global zones and has visibility into and control over them.
- Practically speaking, the global zone is not different from a traditional UNIX system;
  - root generally remains omnipotent and omniscient.
  - The global zone always exists, and acts as the “default” zone in which all processes are run if no non-global zones have been setup.

Specifics

- Process Model
  - Per-zone namespace with no visibility between non-global zones
- Accounting
  - Legacy accounting formats made it tricky, modified accounting to be intra-zone.
- Networking
  - Global zone multi-homed server
  - Each IP associated with a specific zone
Specific

- Filesystem
  - Use loopback filesystem to mount part of global filesystem namespace
  - High degree sharing
- Device
  - Generally discouraged
  - Device semantics generally unclear
  - Compare /dev/null to /dev/kmem
  - /dev/log an exception

Resource Management

- CPU
  - Global fair scheduler can schedule zones
  - Scheduler within a zone can further share
  - Memory still to come 😊

Performance

<table>
<thead>
<tr>
<th>Workload</th>
<th>Base</th>
<th>Zone</th>
<th>Diff (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>38.45</td>
<td>38.29</td>
<td>99.6</td>
</tr>
<tr>
<td>Time-sharing</td>
<td>23332.58</td>
<td>22406.51</td>
<td>96.0</td>
</tr>
<tr>
<td>Networking</td>
<td>283.30</td>
<td>284.24</td>
<td>100.3</td>
</tr>
<tr>
<td>Database</td>
<td>38767.62</td>
<td>37928.70</td>
<td>97.8</td>
</tr>
</tbody>
</table>

Figure 7: Performance impact of running in a zone.

- Timesharing workload related to loopback file system

Drawbridge


Library OS

- OS refactored to run in the context of the application
  - From the application perspective it looks like an OS (Windows in this case)
  - The underlying OS API is smaller
    - Easier to get correct
    - Easier to make secure

Pico process

- An isolated process with different system call interface to normal processes
  - A security monitor in this case
  - Could be something else?
Windows Subsystem for Linux (WSL1)