# Pre-Virtualization: Uniting Two Worlds

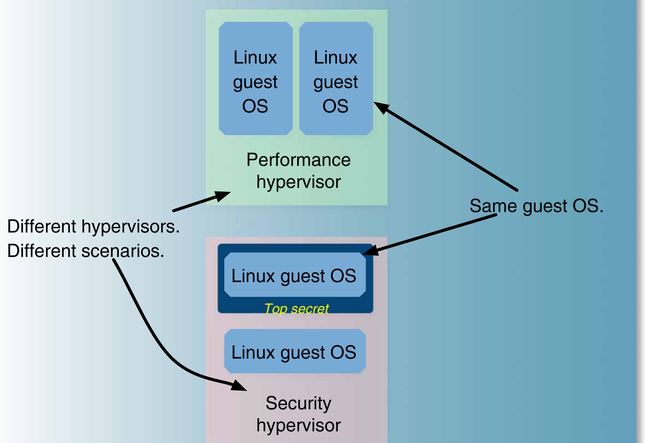

**Joshua LeVasseur**
Universität Karlsruhe (TH), Germany

**Volkmar Uhlig**
IBM Watson Research Center, NY

**Ben Leslie**
National ICT Australia
University of New South Wales

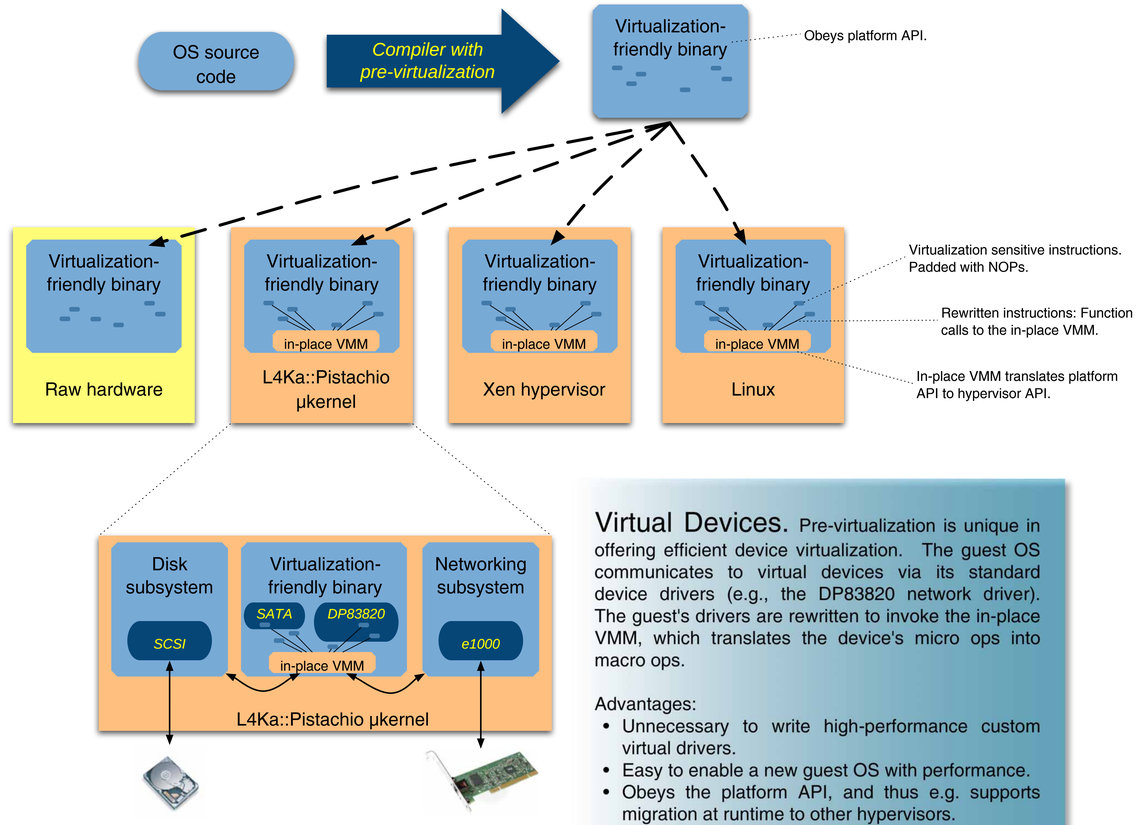
**Matthew Chapman**
National ICT Australia
University of New South Wales

**Gernot Heiser**
National ICT Australia
University of New South Wales


**Overview.** The virtual machine (VM) helps to address a variety of scenarios, while using the same guest OS. The VM enables the use of different hypervisors appropriate for each particular scenario and its trade-offs.

The VM adapts to the scenarios, because it maps the platform API (i.e., the instruction set and devices) to the hypervisor's API. Pure virtualization achieves this goal, but with a high run-time overhead. Para-virtualization solves the performance problem, but throws out support for multiple hypervisors. To unite the two, we propose *pre-virtualization*, which uses the platform API with the performance of para-virtualization.




**Virtual Devices.** Pre-virtualization is unique in offering efficient device virtualization. The guest OS communicates to virtual devices via its standard device drivers (e.g., the DP83820 network driver). The guest's drivers are rewritten to invoke the in-place VMM, which translates the device's micro ops into macro ops.

Advantages:

- Unnecessary to write high-performance custom virtual drivers.
- Easy to enable a new guest OS with performance.
- Obeys the platform API, and thus e.g. supports migration at runtime to other hypervisors.

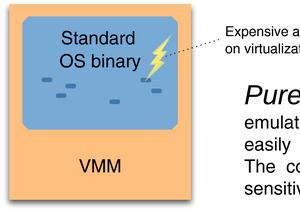
| | System | Xput [Mb/s] | CPU util | cycles/byte |
|---|---|---|---|---|
| Linux 2.6.9 | native, raw | 780.9 | 35.2% | 9.64 |
| | NOPs, raw | 780.2 | 33.5% | 9.17 |
| | L4Ka::Linux | 780.1 | 35.7% | 9.77 |
| | L4Ka in-place VMM | 779.8 | 37.3% | 10.22 |
| | XenoLinux | 780.7 | 41.3% | 11.29 |
| | Xen in-place VMM | 778.7 | 41.1% | 11.28 |
| Linux 2.4.31 | native, raw | 740.8 | 36.0% | 10.39 |
| | NOPs, raw | 740.8 | 36.4% | 10.49 |
| | Xen in-place VMM | 739.6 | 43.2% | 12.48 |

**Performance.** Netperf receive benchmark that transferred 1GB of data. Test machine: 2.8GHz Pentium 4, configured for 256MB, XT-PIC, direct device access, running Debian 3.1 from local SATA. Client machine: 1.4GHz Pentium 4. Gigabit Ethernet connection.

| | System | Xput [Mb/s] | CPU util | cycles/byte |
|---|---|---|---|---|
| Virtual Driver | Custom driver | 707.5 | 60.3% | 18.21 |
| | DP83820 emulation | 707.1 | 59.8% | 18.06 |

In device driver reuse, we used two VMs: one with indirect network access, the other with direct access. The indirect VM either used a pre-virtualized DP83820 driver, or para-virtualization with a custom virtual driver.

**Conclusion.** Pre-virtualization offers performance rivaling para-virtualization, and by using the platform API, pre-virtualization also offers many of the advantages of pure virtualization. Pre-virtualization's automation substantially reduces the engineering effort to build a high-performance virtual machine.

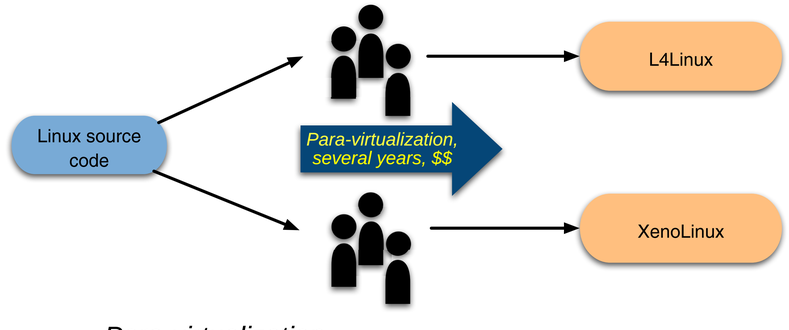



Expensive and frequent privileged traps on virtualization sensitive instructions.

*Pure virtualization*, the classic approach, faithfully emulates the platform API (i.e., instruction set and devices), easily supporting many types of guest operating systems. The cost is a high runtime overhead due to trapping on sensitive operations. Also, x86 is difficult to support.

## Comparison

| | Pre-virtualization | Pure virtualization | Para-virtualization |
|---|---|---|---|
| Mostly automated | ✓ | ✓ | – |
| Single guest binary, runs on all hypervisors | ✓ | ✓ | – |
| High performance | ✓ | – | ✓ |
| Quickly enable a new hypervisor | ✓ | ? | – |
| Migrate between incompatible hypervisors at runtime | ✓ | ✓ | – |
| Quickly port a new guest OS | ✓ | ✓ | – |
| High performance virtual devices | ✓ | – | – |
| Compatible with x86 | ✓ | – | ✓ |
| Follows platform API | ✓ | ✓ | – |




*Para-virtualization* achieves great performance and support for x86, by manually porting the OS to the API of a single hypervisor. The costs are the abandonment of virtualization's core features, substantial engineering effort, and introduction of new bugs.

Information and source code:
http://l4ka.org/projects/virtualization/