InstantLab – The Cloud as Operating System Teaching Platform

Alexander Schmidt, Andreas Polze
Operating Systems and Middleware Group

Cloud Futures 2011

Operating Systems and Middleware

Prof. Dr. rer. nat. habil. Andreas Polze
Dipl.-Inf. Alexander Schmidt
Hasso-Plattner-Institute for Software Engineering
at University Potsdam
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
Agenda

1. Operating System Experiments – the Windows Case
2. InstantLab
3. Demo
4. Research Questions
5. Conclusions

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011
Windows Research Kernel (WRK)

- Stripped down Windows Server 2003 sources
  - Only kernel itself, no drivers, GUI, user-mode components
  - Missing components: HAL, power management, plug-and-play
- Released in 2006
- Freely available to academic institutions
- Encouraged by license:
  - Modification
  - Publication (of excerpts)

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011

Structuring Experiments: The UMK Approach

- U-phase
  - Concentrate on OS concepts
  - Introduce OS interfaces
  - Systems programming
- M-phase
  - Observe concepts at run-time
  - Introduce monitoring tools
  - System measurements
- K-phase
  - Discuss kernel implementation
  - Introduce kernel source code (WRK/UNIX)
  - Kernel programming

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011
Kernel Programming Experiments

- Debugging/Instrumenting the WRK
  - Boot phase
  - Process creation
  - Single-step debugging the WRK in a virtual machine

- Creating a new system call
  - Hide/Show a specified process from the system
  - Memorize hidden processes
  - Implement a system service DLL

- Memory management

Kernel Programming Experiments – Bottom Line

- Experiments comprise
  - Documentation
  - Source code
  - Workload generators
  - Measurement/visualization tools

- Experiment setup:
  - Install and configure test operating system
  - Build and deploy the sources
  - Configure kernel debugging infrastructure

- Virtualization helps, but
  - Variety of OS platforms, virtualization vendors among students
  - Hardware requirements
Agenda

1. Operating System Experiments – the Windows Case
2. InstantLab
3. Demo
4. Research Questions
5. Conclusions

The InstantLab Idea

- Provision of “canned experiments”
  - Virtual machine images (VMI) as foundation
  - Self-contained, pre-configured experiment in one VMI
  - Instantaneous execution of a lab or experiment on Cloud resources
Embrace The Cloud

- Virtualize laboratory environment
  - No physical machines in university, no maintenance
  - Compute resources in the Cloud
- Migrate exercises and demos into the Cloud
  - Provision of VM template(s) for each exercise
  - Instantiation on demand
- Facilitate experiments through remote display session
  - Run experiments in Web browser
  - Support of various platforms and compute power

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011

InstantLab - Architecture

InstantLab Manager

Persistent Storage

Virtualized Laboratory

Workspace

Cloud Infrastructure

VM

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.

Exp.
Agenda

1. Operating System Experiments – the Windows Case
2. InstantLab
3. Demo
4. Research Questions
5. Conclusions

Facilitating Remote Access
Lab Management – Architecture

InstantLab Demo – Lab Management
InstantLab Demo – Lab Management

Agenda

1. Operating System Experiments – the Windows Case
2. InstantLab
3. Demo
4. Research Questions – Cloud Reliability
5. Conclusions
Dependability – does it matter for Cloud?

Umbrella term for operational requirements on a system

- "Trustworthiness of a computer system such that reliance can be placed on the service it delivers to the user" [Laprie]

General question: How to deal with unexpected events?

Hardware Revolution in the x86 World
Classical Reliability Wisdoms Get Replaced

- Dramatic shift in single machine reliability aspects
  - SMP becomes heterogeneous tiled on-chip network
  - Decreasing structural sizes + dynamic frequency and voltage
  - Massive memory increase
- More fault classes, less error containment!
- Few research results from HPC perspective
  - Type and intensity of workload significantly influences life time
  - Failure rates depend on processor count, not hardware type

Research in the FutureSOC Lab

HPI FutureSOC Lab

- Collaboration with industry for software research on next-generation x86 hardware (32-65 cores, 1-2 TB RAM)

Our research @ FutureSOC Lab

- Failure prediction based on cross-level monitoring data analysis
- Pro-active virtual machine migration
- Fault injection based on UEFI firmware technology
CPU Level:
Online Hardware Failure Prediction

Using X86 hardware performance events

- Instruction retirement, cache miss, branch miss-prediction, ...
  - Limited number of hardware counter units -> exploit event correlations
  - Threshold-triggered, time-triggered
- Applicable to major cellular multiprocessing platforms (Intel, AMD, SPARC, IBM Power)

Memory level:
observations from our FutureSOC Lab

<table>
<thead>
<tr>
<th>Date</th>
<th>Severity</th>
<th>Event</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-Jun-2010</td>
<td>Info</td>
<td>No</td>
<td>BIOS</td>
<td>System boot (POST complete)</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Major</td>
<td>No</td>
<td>[0x00:00]</td>
<td>POST - ‘MEM4_DIMM-2D’ memory training failed</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Major</td>
<td>No</td>
<td>[0x00:00]</td>
<td>POST - ‘MEM4_DIMM-1D’ memory training failed</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>SMI</td>
<td>‘MEM4_DIMM-1D’ Memory: Uncorrectable error (ECC)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>SMI</td>
<td>‘MEM4_DIMM-1C’ Memory: Uncorrectable error (ECC)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>SMI</td>
<td>‘MEM4_DIMM-1B’ Memory: Uncorrectable error (ECC)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>‘MEM4_DIMM-2D’: Memory module failed (disabled)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>‘MEM4_DIMM-1D’: Memory module failed (disabled)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>‘MEM4_DIMM-2B’: Memory module failed (disabled)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>‘MEM4_DIMM-1B’: Memory module failed (disabled)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Info</td>
<td>No</td>
<td>BIOS</td>
<td>System boot (POST complete)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>‘MEM4_DIMM-1D’: Memory module error</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Major</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>‘MEM4_DIMM-1D’: Memory module failure predicted</td>
</tr>
</tbody>
</table>

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011
OS level:
our NTrace for Windows

- Compiler/linker switch
  - /hotpatch, /functionpadmin
  - Microsoft C compiler shipped with Windows Server 2003 SP1 and later
- Hotpatchable:
  - Windows Research Kernel

```
Foo-5:
    ...
    mov  edi, edi
    push ebp
    mov  ebp, esp
    mov  ecx, [ebp+18h]
    mov  edx, [ebp+0Ch]
    ...
```

```
NtfsPinMappedData:
    mov  edi, edi
    push ebp
    mov  ebp, esp
    mov  ecx, [ebp+18h]
    mov  edx, [ebp+0Ch]
    ...
```

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011

---

The Meta Predictor – Bringing it all together

Ensemble learning:

- Boosts accuracy – which failure-prone situations can best be identified by either hardware, OS, VMM failure predictors?
- Domain knowledge – operating system vendors know their system best and can provide the most advanced predictor on OS level
- Pluggable – domain predictors provided by an application vendor can easily be integrated into our anticipatory virtualization architecture
- Ensemble-learning can combine predictions across all system levels

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011
Our Idea: Global System Health Indicator

Multi-Level Failure Prediction

- Application & Middleware level: Application-specific counters, JSR-77, AppServer Monitoring
- Operating System Level: Dtrace, Windows Monitoring Kernel
- VMM Level: VMware vProbe
- Hardware level: Machine Check Architecture, CPU Hardware Profiling

Virtualization Cluster Management
System Health Indicator

VM Migration – how long does it take?
VMWare ESX 4

blackout time vs. cpu load
blackout time vs. physical RAM usage

migration_time_of_VMware_ESX_vSphere_4.0 : vmsize = 4096
migration_time_of_VMware_ESX_vSphere_4.0 : vmsize = 8192

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011
Agenda

1. Operating System Experiments – the Windows Case
2. InstantLab
3. Demo
4. Research Questions
5. Conclusions

Applying it to the Cloud

- Servers have evolved – cloud will too
  - Ever growing number of CPU cores
  - Tremendous amounts of memory
- Reliability will become the most sought-after feature of future server systems
  - Higher density, integration levels in future CPUs will lead to multi-bit faults
  - Failure prediction and VM migration as promising concept
- Must have fault isolation boundaries (LPARs, blades)
- Cloud will embrace new programming and management models
Servers have evolved...

- New form factors
- Higher density
- Standard architectures
- Multicore/multithreaded

Advances in operating systems

- Virtualization
- Thrustworthiness/security
- Clustering
- Need for new programming models, SW Architectures, Services

Virtualization problems

- Security: extended attack surface
- Virtualization-based malware
- Must trust hypervisor

Cloud Computing – the three layers

Challenges:

- Has to abstract underlying hardware
- Be elastic in scaling to demand
- Pay per use basis

Hybrid Computing

OpenCL: New Programming Models

- One Host + one or more Compute Devices
  - Each Compute Device is composed of one or more Compute Units
  - Each Compute Unit is further divided into one or more Processing Elements

Computer architecture drives changes in system software

Andreas Polze, Operating Systems and Middleware

Alexander Schmidt, Andreas Polze | Cloud Futures 2011 | June 2, 2011