Experiences in the Land of Virtual Abstractions

Galen Hunt

Principal Researcher

Microsoft Research Operating Systems Group

Why do we all love (hardware) VMs?

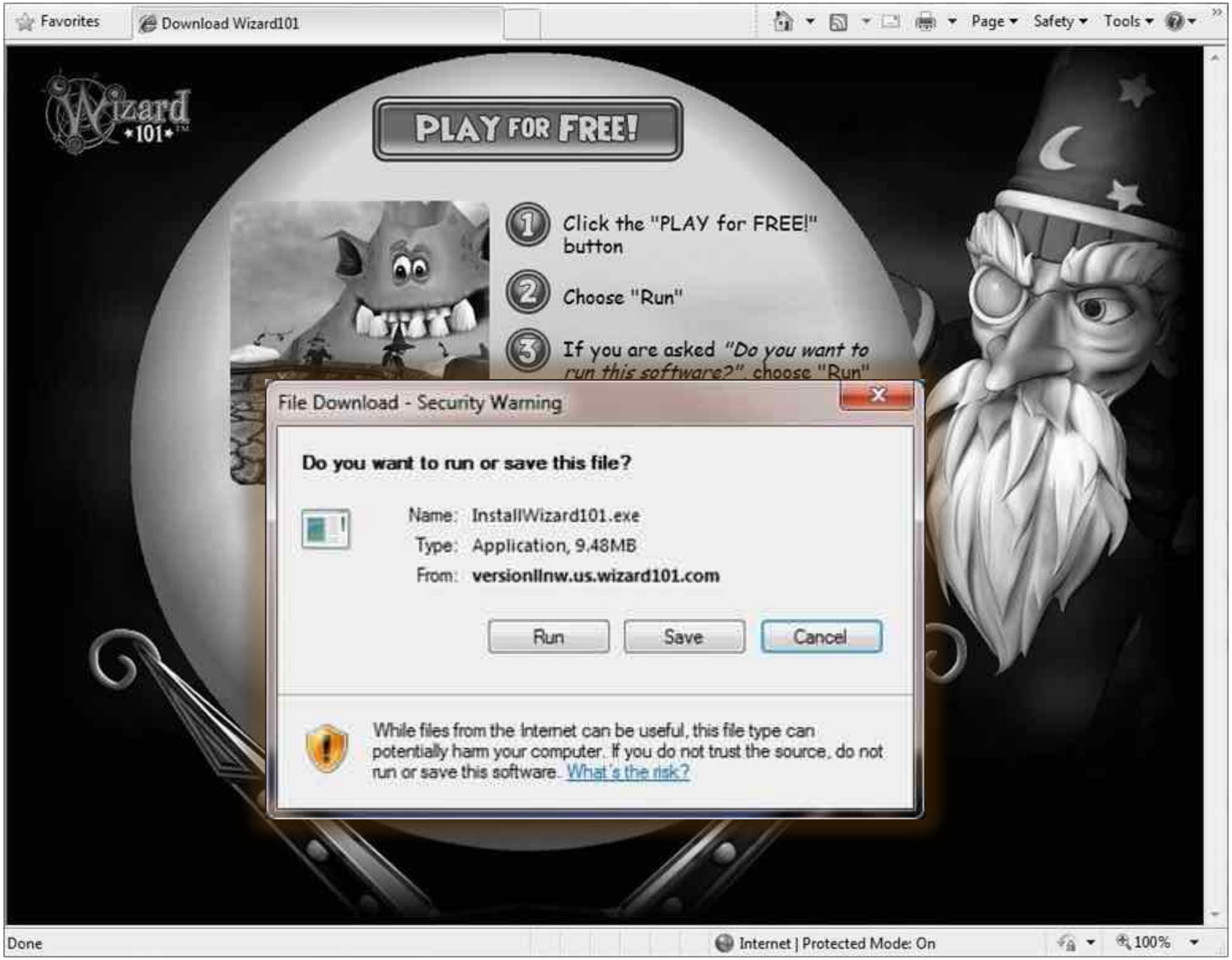
- Three reasons:
  - **Compatibility**
    - I can install my application in a VM with the OS it requires and never have to worry about it breaking again.
  - **Security**
    - I can download even the most malignant code from the internet, run it in a VM, and my system's integrity isn't lost.
  - **Continuity**
    - I can start an application (in a VM) on one computer and move it to another, or reboot the computer, and it still runs.

Fantastic Disruption

- VMMs changed **server** computing forever:
  - server consolidation
  - cloud computing

- Why not desktop and mobile computing?
  - VMs have huge memory and disk overheads
    - "XP mode in Win7" : 1GB VHD, 256MB RAM
    - "Win7 VM" : 4GB+ VHD, >= 512MB RAM

The Father's Dilemma



1. Click the "PLAY for FREE!" button
2. Choose "Run"
3. If you are asked *"Do you want to run this software?"* choose "Run"

File Download - Security Warning

Do you want to run or save this file?

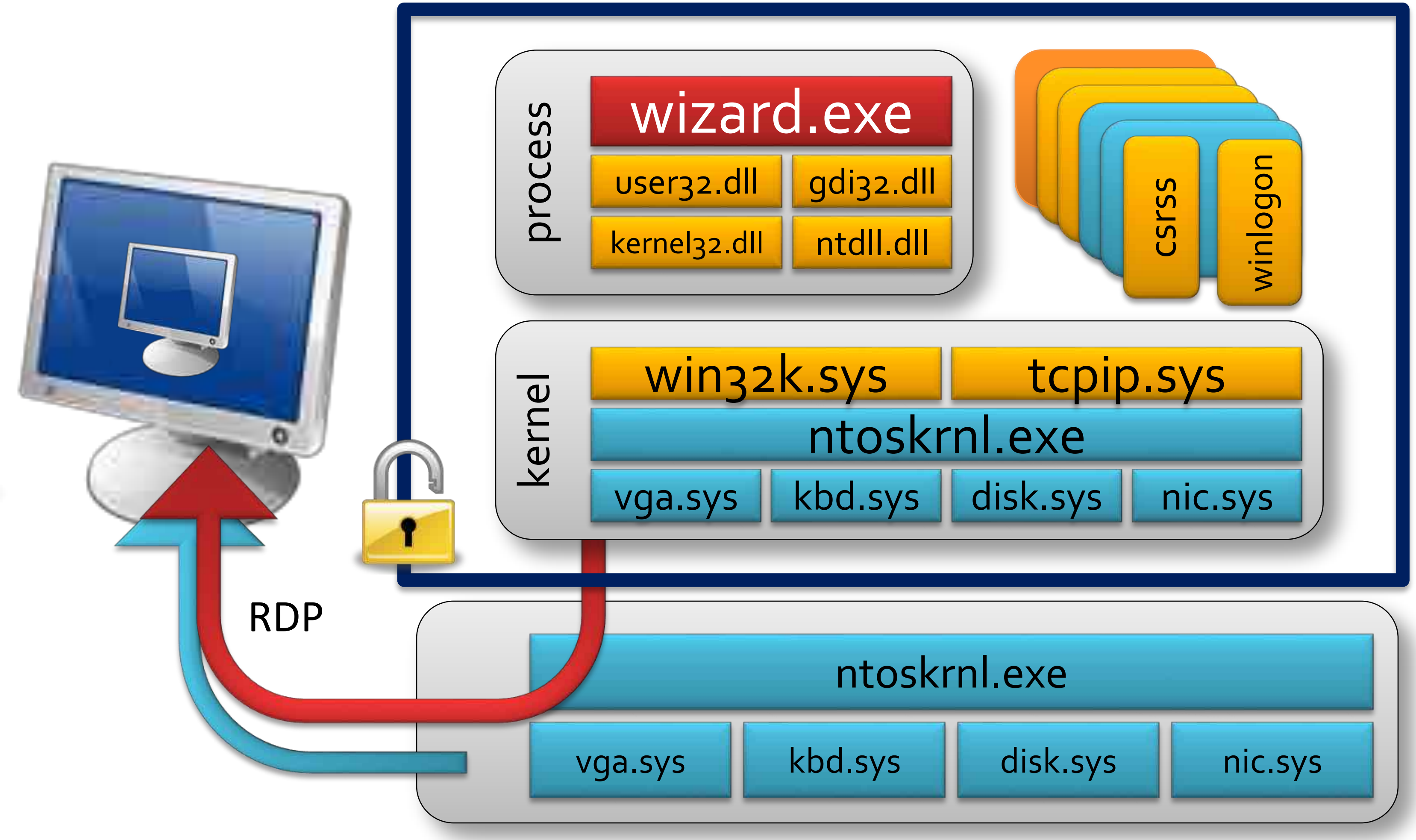
Name: InstallWizard101.exe

Type: Application, 9.48MB

From: **versionllnw.us.wizard101.com**

While files from the Internet can be useful, this file type can potentially harm your computer. If you do not trust the source, do not run or save this software. What's the risk?

Do we need to duplicate the full OS in every VM?



What might be the alternative abstractions?

- Windows abstractions at top of Win32 API?
  - too large to be practical for self-contained apps: 250K+ APIs, registry, etc.
  - not (completely) *compatible* across OS versions
  - not *secure* against untrusted applications
  - not *continuous* as users move from one device to another

What properties do the abstractions need?

- (or what is it that is so “magical” about the VM hardware ABI?)
  - Stable
  - Formalized
  - Closed
  - Transparent (stateless)
- So, … can we make a higher-level ABI that has these properties?

Drawbridge Abstractions & ABI

45 ABI Downcalls

Memory Management Primitives (3)

- DkVirtualMemoryAlloc
- DkVirtualMemoryFree
- DkVirtualMemoryProtect

Threading Primitives (16)

- DkThreadCreate[1]
- DkThreadDelayExecution
- DkThreadYieldExecution
- DkThreadExit
- DkThreadGetParameter
- DkThreadRaiseException
- DkNotificationEventCreate
- DkSynchronizationEventCreate
- DkSemaphoreCreate
- DkSemaphoreRelease
- DkSemaphorePeek
- DkEventSet
- DkEventClear
- DkEventPeek
- DkObjectsWaitAny
- DkAbortEventRegister[1]

Child Process Primitives (3)

- DkProcessCreate[1]
- DkProcessGetExitCode[1]
- DkProcessExit[1]

I/O Stream Primitives (14)

- DkStreamOpen[1]
- DkStreamRead[2]
- DkStreamWrite[2]
- DkStreamMap[2]
- DkStreamMapPeBinary[2]
- DkStreamUnmap[2]
- DkStreamSetLength[2]
- DkStreamFlush[2]
- DkStreamDelete[1]`
- DkStreamGetEvent[2]
- DkStreamRename[1]
- DkStreamEnumerateChildren[1]
- DkStreamAttributesQuery[1]
- DkStreamAttributesQueryByHandle[2]

Other Primitives (9)

- DkSystemTimeQuery
- DkRandomBitsRead
- DkInstructionCacheFlush
- DkObjectReference
- DkObjectClose[2]
- DkInputEventRead[1]
- DkFrameBufferExport[1]
- DkFrameBufferNotifyUpdate[1]
- DkDebugStringPrint[1]

Upcalls (3)

3 Upcalls

- LibOsInitialize
- LibOsThreadStart
- LibOsExceptionDispatch

13 I/O Providers

Files/Storage (1)

```
file:
```

Console Redirection (4)

```
null:
stderr:
stdin:
stdout:
```

Named Pipes (2)

```
pipe.client:
pipe.server:
```

TCP/IP Stack (4)

```
dns:
tcp.client:
tcp.server:
tcp:
```

HTTP.SYS (2)

```
http.application:
http.server:
```

Learning from the past…

"Since March of 1989 we have had running at CMU a computing environment in which the functions of a traditional Unix system are cleanly divided into two parts: facilities which manage the hardware resources of a computer system (such as CPU, I/O and memory) and support for higher-level resource abstractions used in the building of application ..."

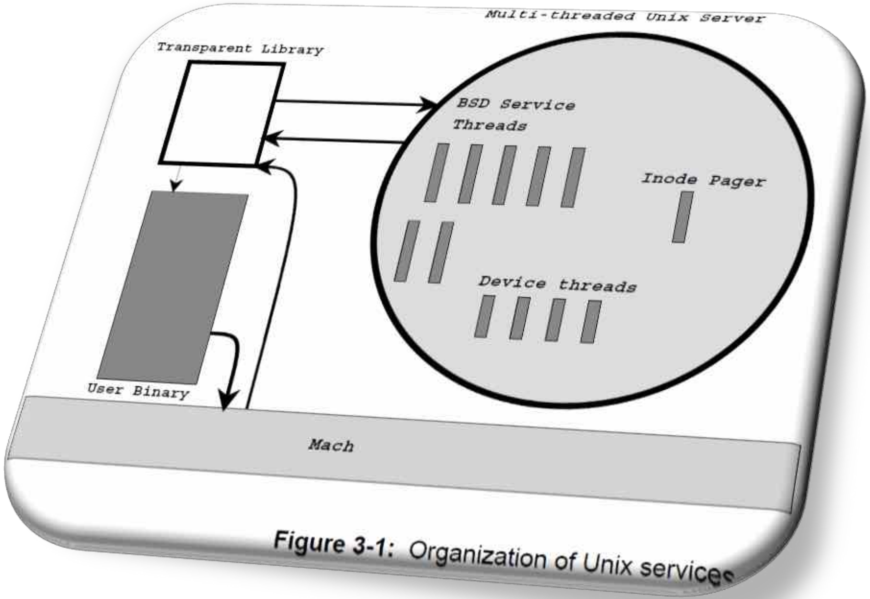
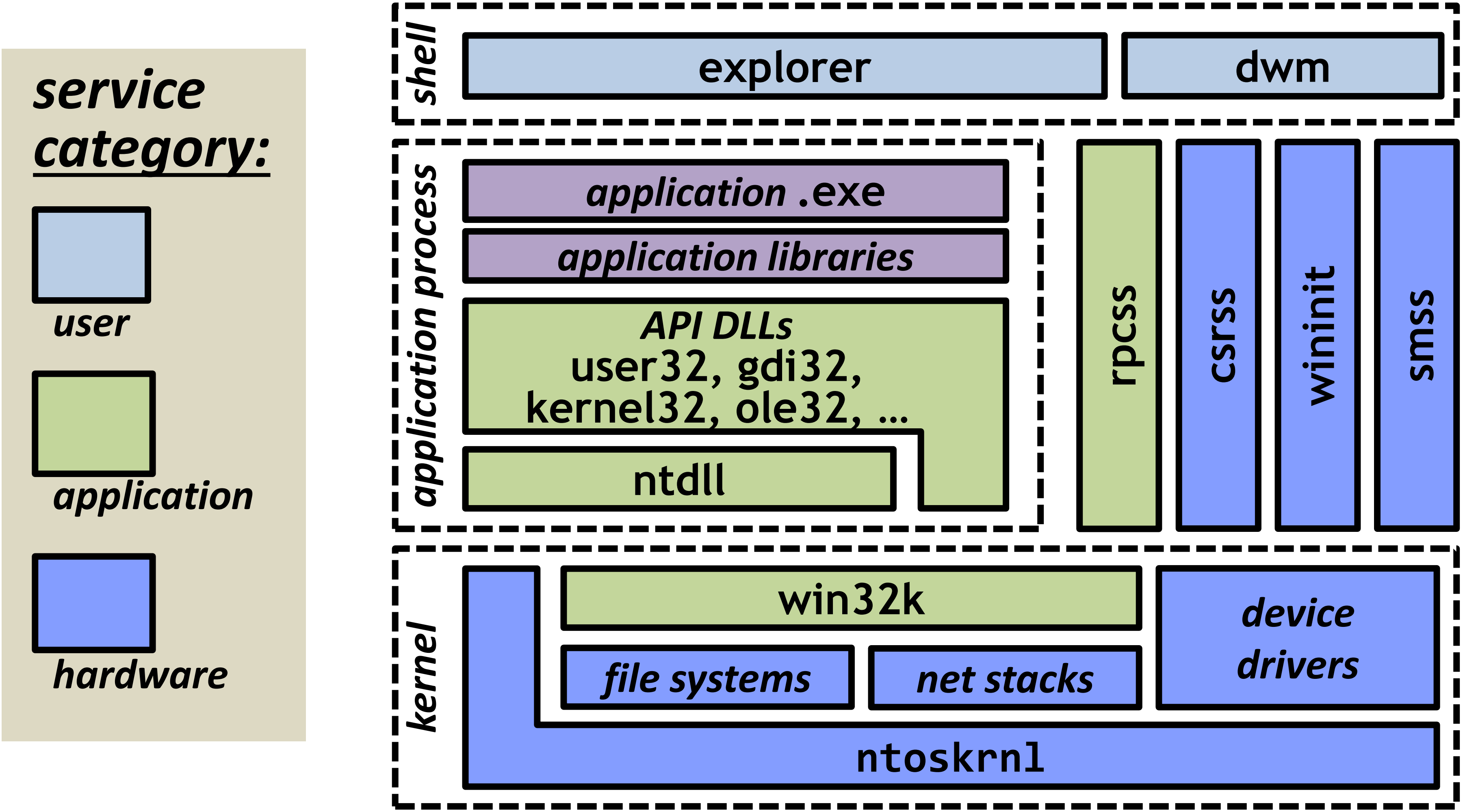


Figure 3-1: Organization of Unix services

"UNIX as an Application Program"
David B. Golub, Randall W. Dean,
Alessandro Forin, and Richard F. Rashid
Proc. of the 1990 Summer USENIX Technical Conference

Three categories of services in an OS



service category:

user

application

hardware

shell

explorer

dwm

application process

application .exe

application libraries

API DLLs
user32, gdi32,
kernel32, ole32, …

ntdll

rpcss

csrss

wininit

smss

kernel

win32k

file systems

net stacks

device drivers

ntoskrnl

Compatibility

- Existing **library OS** implementations:

Win7 SP1

Win8

x86
&
x64

Linux

Win7

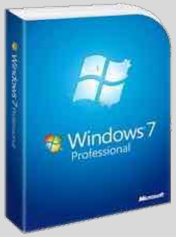
x86
&
x64

Silverlight

x86
&
x64

- Full **host** implementations:

Windows 7 / Server 2008 R2



Windows 8.1/ Server 2012 R2

- Prototype **host** implementations *(past, present, and in-progress)*:

Windows 8.1

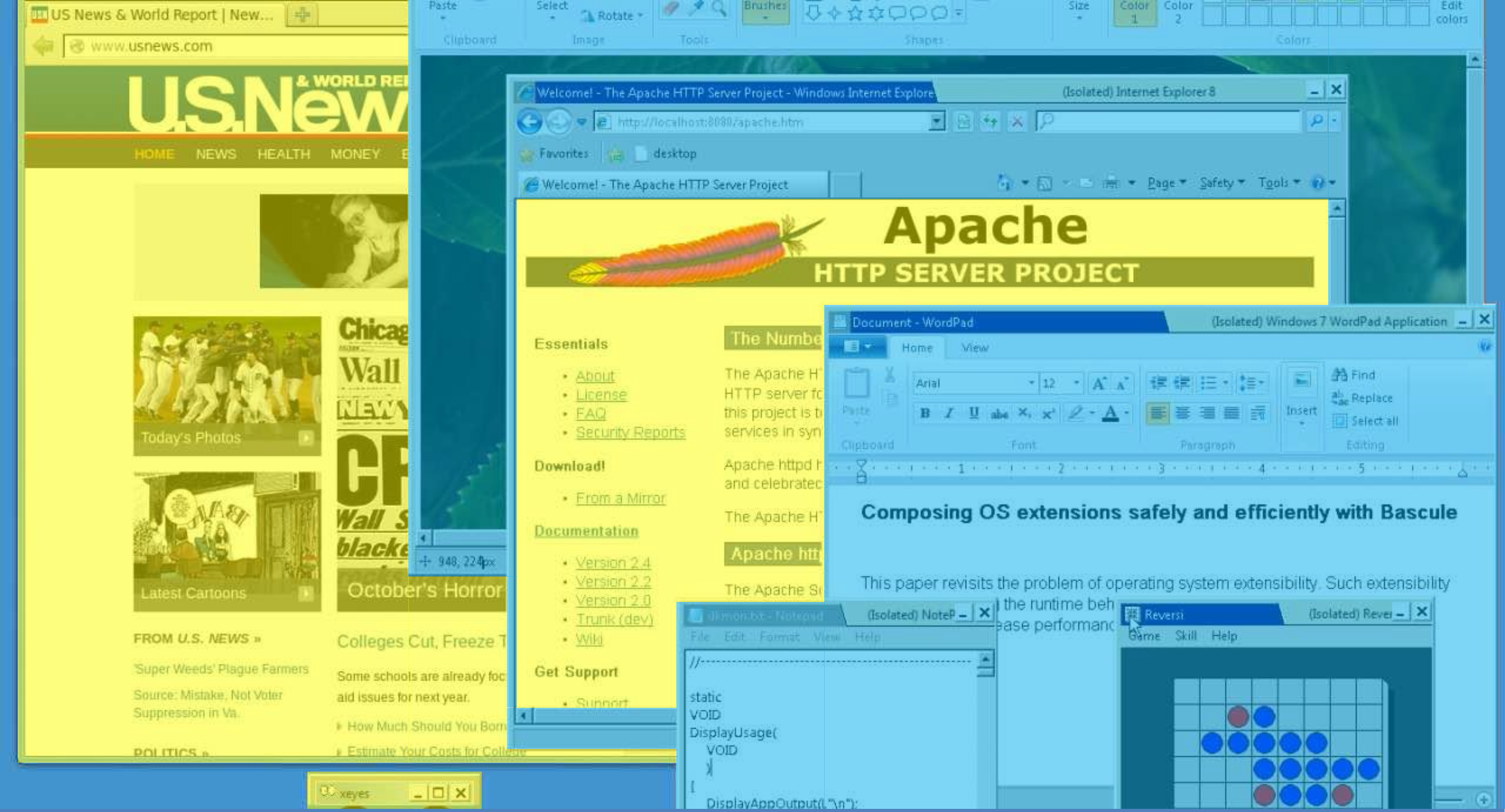
Win7 MinWin
(x86 on ARM)

Xbox 360
(x86 on PPC)

Barrelfish
(research OS)

Windows
Embedded
(CE)

Midori
(research OS)



The Apache H HTTP server fc this project is t services in syn

Apache httpd h and celebratec

The Apache H

The Apache S

Composing OS extensions safely and efficiently with Bascule

This paper revisits the problem of operating system extensibility. Such extensibility
the runtime beh
ease performanc

```
//------------------------------------------------

static
VOID
DisplayUsage(
    VOID
    )
{
    DisplayAppOutput(L"\n");
```

Windows and Linux LibOSes

Linux apps: Firefox, Apache, xeyes

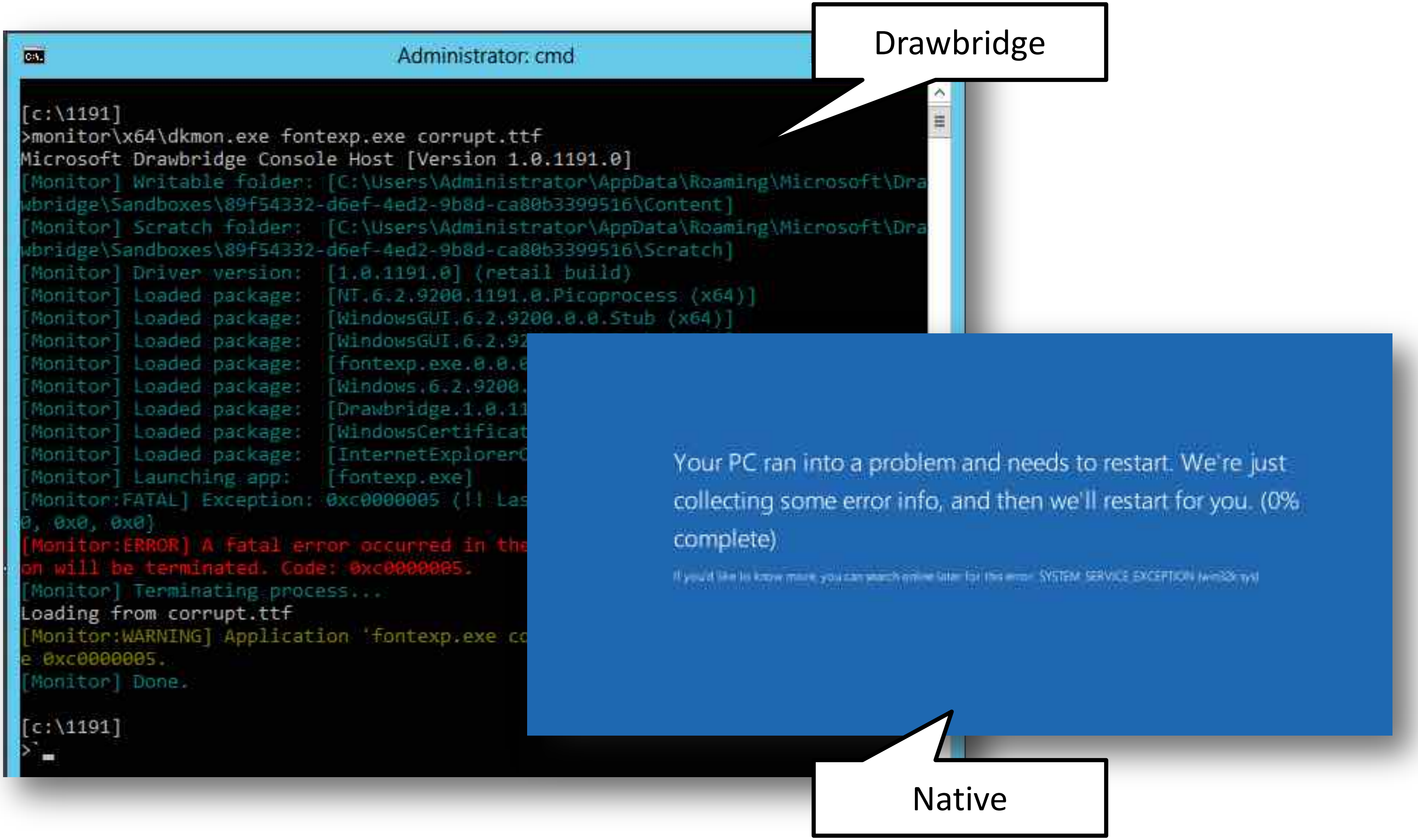
Windows apps: Paint, IE, WordPad, Notepad, Reversi

Security: What's the Thread Model?

- Traditional OS: "**Enterprise Multitenancy**"
  - Invite your employees
    - after full authentication
    - to run the applications you choose (or that your OS vendor vets)
    - on some subset of your computers

- VMs & Drawbridge: **"Hostile Multitenancy"**
  - Invite "organized crime"
    - with complete anonymity (in name & number)
    - to run any code they choose
    - on the same servers as your most valuable customers

Security



```
[c:\1191]
>monitor\x64\dkmon.exe fontexp.exe corrupt.ttf
Microsoft Drawbridge Console Host [Version 1.0.1191.0]
[Monitor] Writable folder: [C:\Users\Administrator\AppData\Roaming\Microsoft\Dra
wbridge\Sandboxes\89f54332-d6ef-4ed2-9b8d-ca80b3399516\Content]
[Monitor] Scratch folder:  [C:\Users\Administrator\AppData\Roaming\Microsoft\Dra
wbridge\Sandboxes\89f54332-d6ef-4ed2-9b8d-ca80b3399516\Scratch]
[Monitor] Driver version:  [1.0.1191.0] (retail build)
[Monitor] Loaded package:  [NT.6.2.9200.1191.0.Picoprocess (x64)]
[Monitor] Loaded package:  [WindowsGUI.6.2.9200.0.0.Stub (x64)]
[Monitor] Loaded package:  [WindowsGUI.6.2.92
[Monitor] Loaded package:  [fontexp.exe.0.0.0
[Monitor] Loaded package:  [Windows.6.2.9200.
[Monitor] Loaded package:  [Drawbridge.1.0.11
[Monitor] Loaded package:  [WindowsCertificat
[Monitor] Loaded package:  [InternetExplorerC
[Monitor] Launching app:   [fontexp.exe]
[Monitor:FATAL] Exception: 0xc0000005 (!! Las
0, 0x0, 0x0)
[Monitor:ERROR] A fatal error occurred in the
on will be terminated. Code: 0xc0000005.
[Monitor] Terminating process...
Loading from corrupt.ttf
[Monitor:WARNING] Application 'fontexp.exe co
e 0xc0000005.
[Monitor] Done.

[c:\1191]
>
```

Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (0% complete)

If you'd like to know more, you can search online later for this error: SYSTEM SERVICE EXCEPTION (win32k.sys)

Drawbridge

Native

Continuity

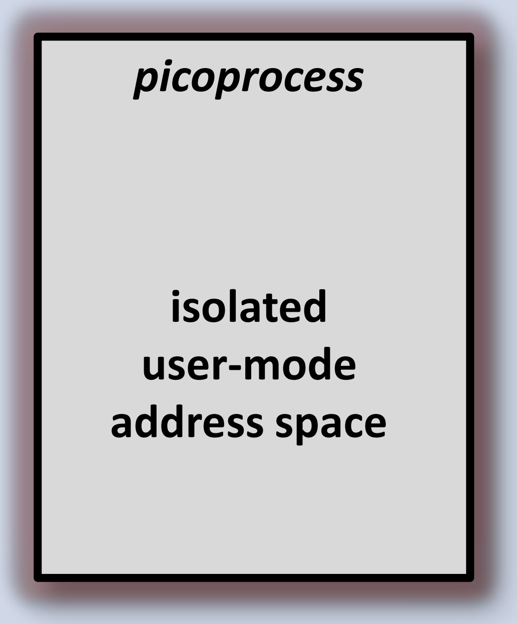
- **Checkpointer Extension**
  - Adds migration/fault tolerance to any unmodified apps and LibOSes on any Drawbridge host platform
  - At runtime, track state:
    - Writable/modified virtual memory allocations
    - All threads & synchronisation
    - Open streams, and their parameters
    - Outstanding I/O
  - At checkpoint time:
    - Cancel pending I/O and ABI calls
    - Open file (using ABI)
    - Serialise address space, thread contexts and other state to file

Demo

Summary

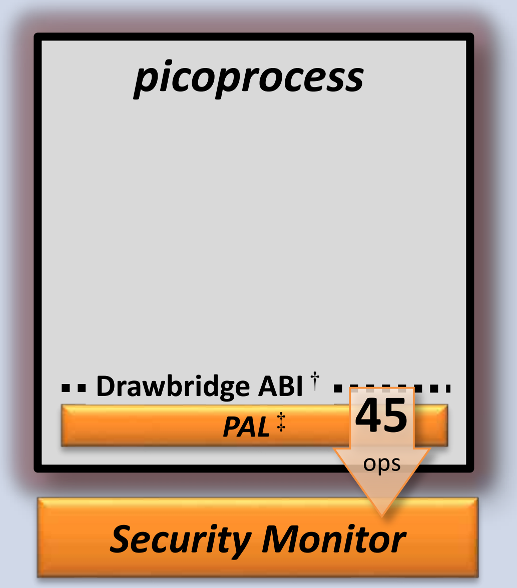
- Drawbridge is a light-weight VMs alternative for secure application hosting.
- Drawbridge consists of three pieces:

Picoprocess



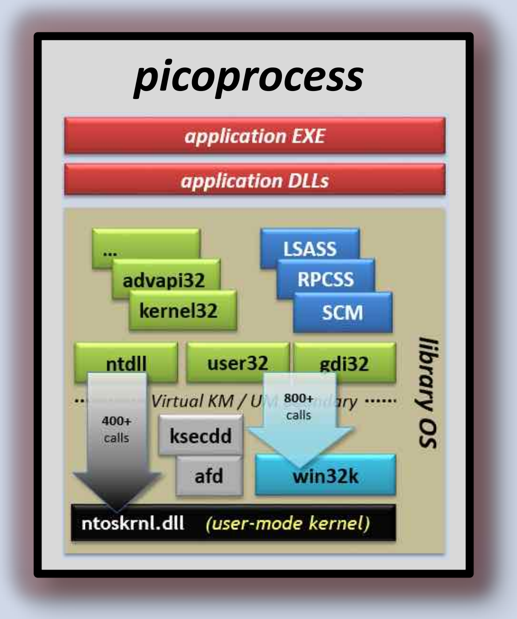
- Secure container for code.
- **No** access to the host OS.

Security Monitor



- Provides primitive OS operations to picoprocs.
  (threads, VM, i/o streams)

Library OS

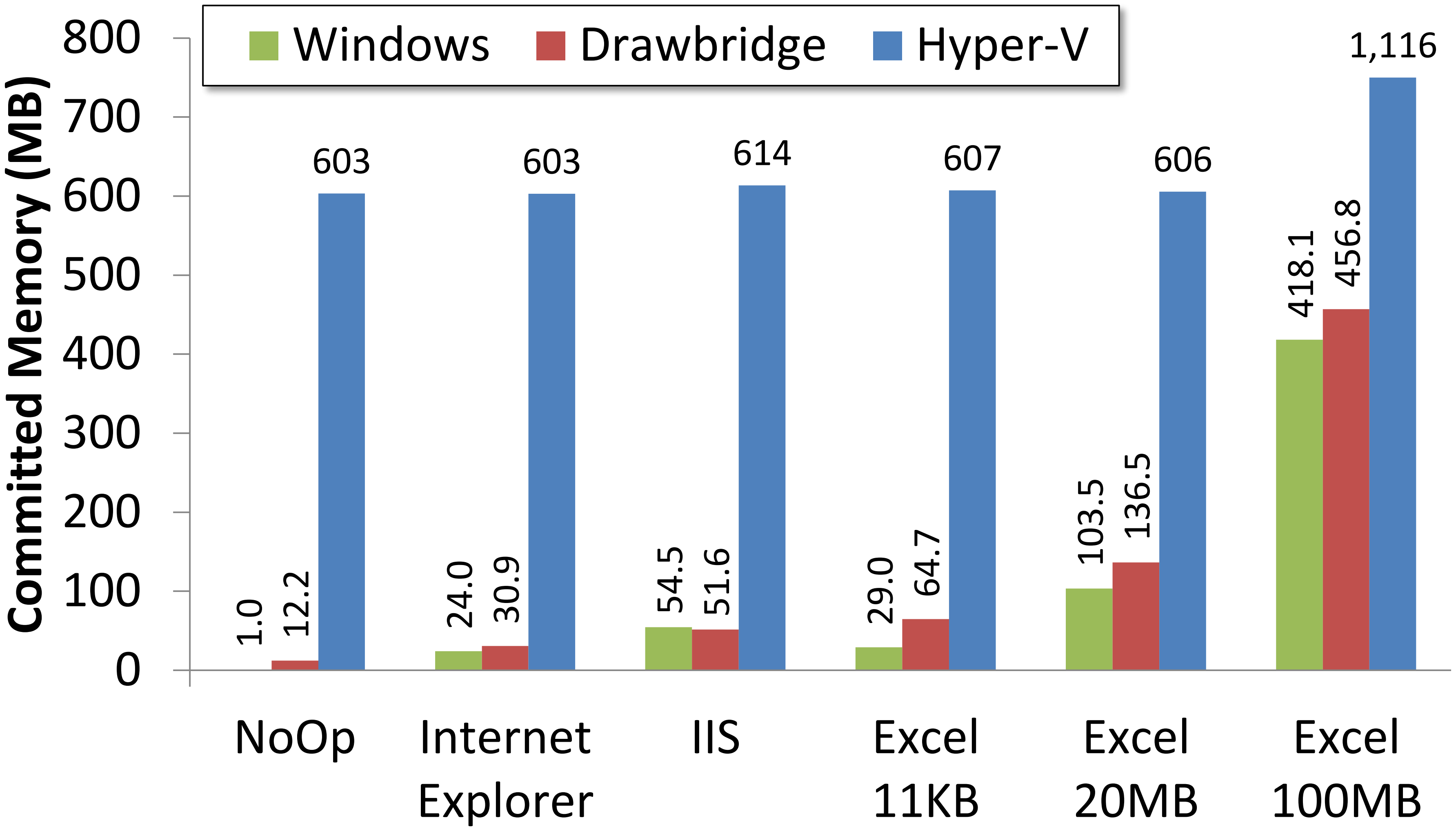


- Subset of Windows w/ picoproc. enlightenments.
  ~16MB mem. overhead

But…

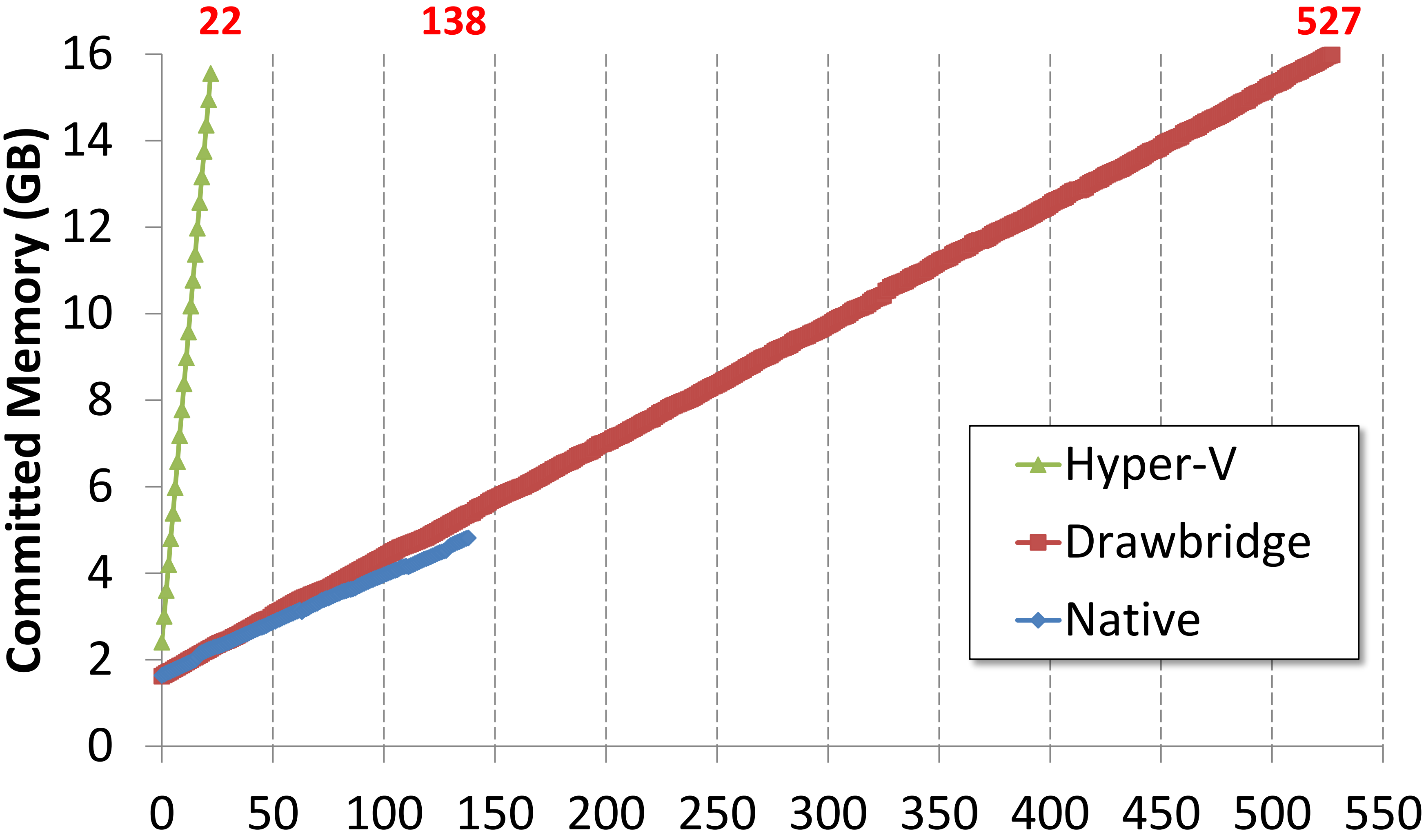
- Do these higher-level abstractions have benefits beyond those of a hardware VM?
- Yes,…
  - Higher density…
  - Layerable (cheaply)…
  - More versatile…
  - … see also [ASPLOS 2011]

Density: Committed Memory by Application



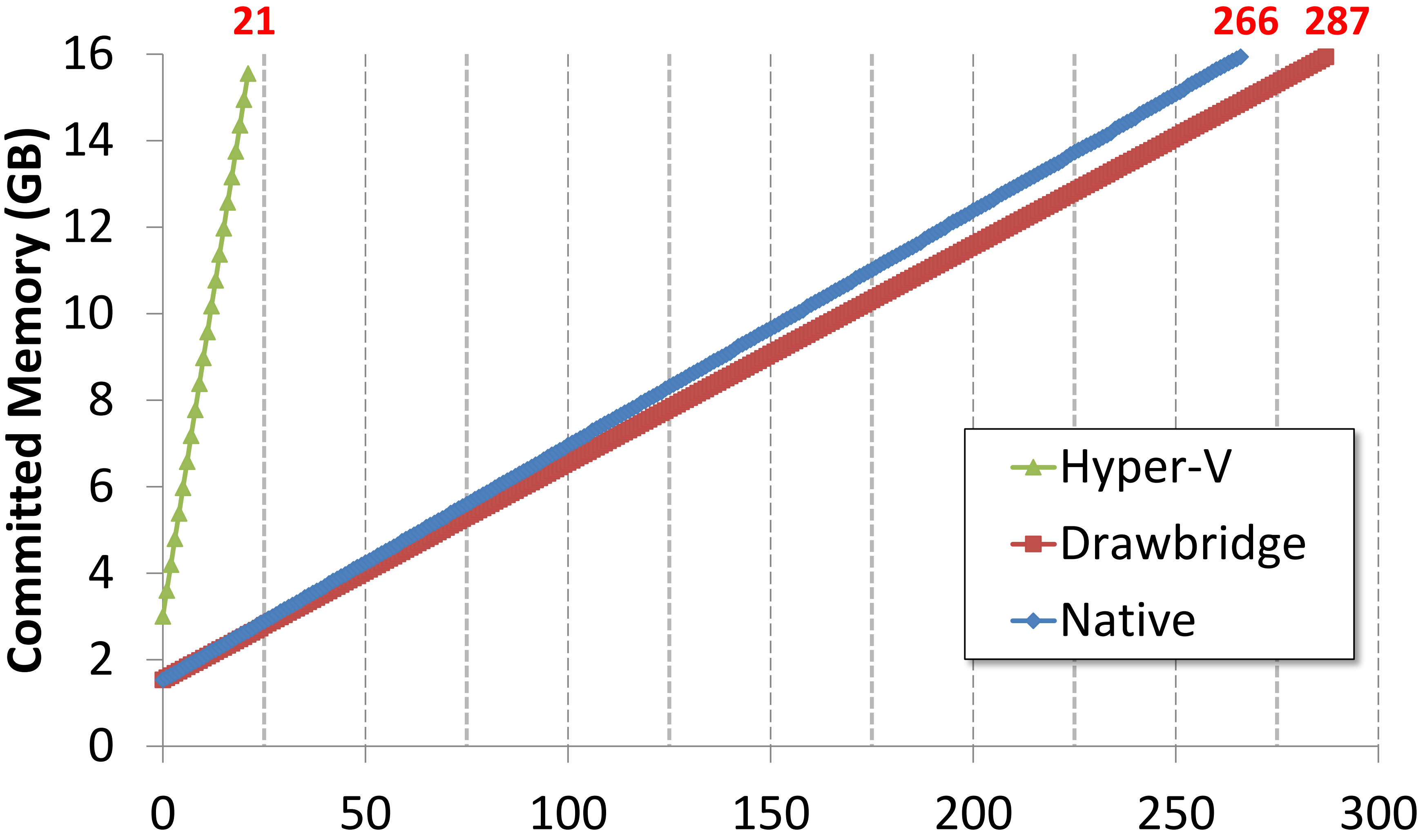
+11.1MB +6.9MB <0 MB +35.7MB +33MB +38.7MB

VM Committed for IE8 Instances



* Native stops at 138 instances when IE reaches the per-session limit on GDI handles.

VM Committed for IIS instances

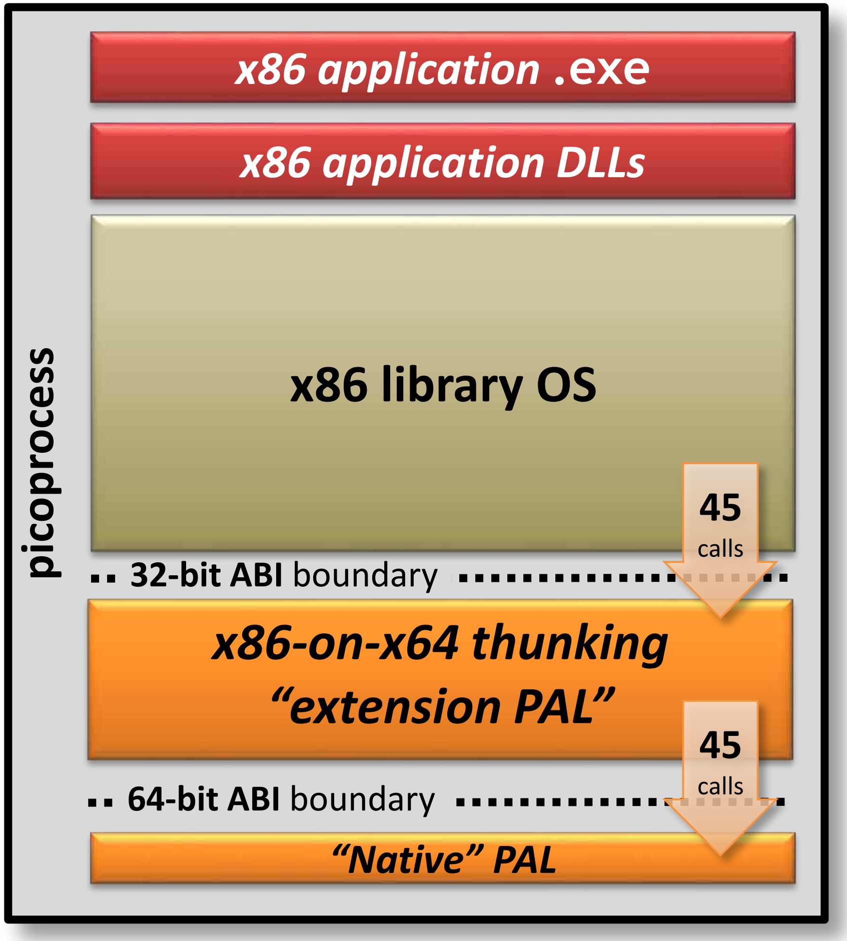


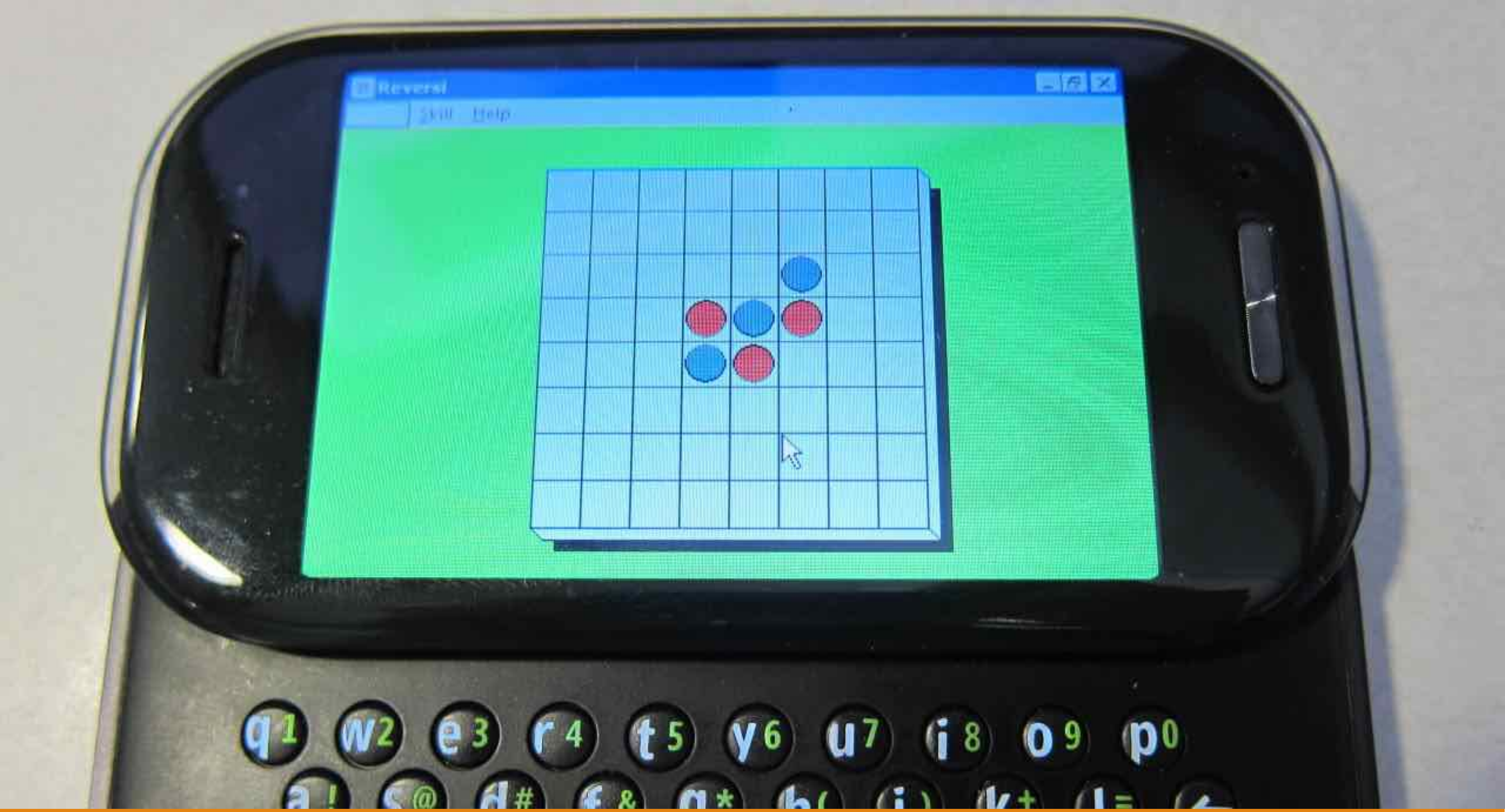
Layerable: Extensions [EuroSys 2013]

- Change the runtime behaviour of an application/OS
- Developed by a third party
- Applied by end user or system integrator
- “Bascule” ABI:
  - Nestable in-process ABI of common OS primitives
  - “Host” provides:
    - Table of function entry points
    - Data structure of startup parameters
  - “Guest” provides:
    - Table of upcall entry points
  - Bootstrap: each layer loads the one above

Drawbridge and WoW64

- Subtly different from WoW64
  - **Same 32-bit library OS used on both x86, x64**
  - **64-bit host unaware of 32-bit code**
- x86-on-x64 "extension" PAL
  - Depends only on 64-bit ABI
  - Exposes 32-bit ABI to library OS
  - Looks like a PAL to the layer above it, like a library OS to the layer below
  - Internally thunks between x86, x64
- Approach generalizes
  - …for CPU emulation (e.g. x86-on-ARM "emulating extension")
  - …for any layered ABI filters





Architecture adaptation extension

Unmodified x86 app and LibOS on ARM host, migrates from x86 to ARM and back

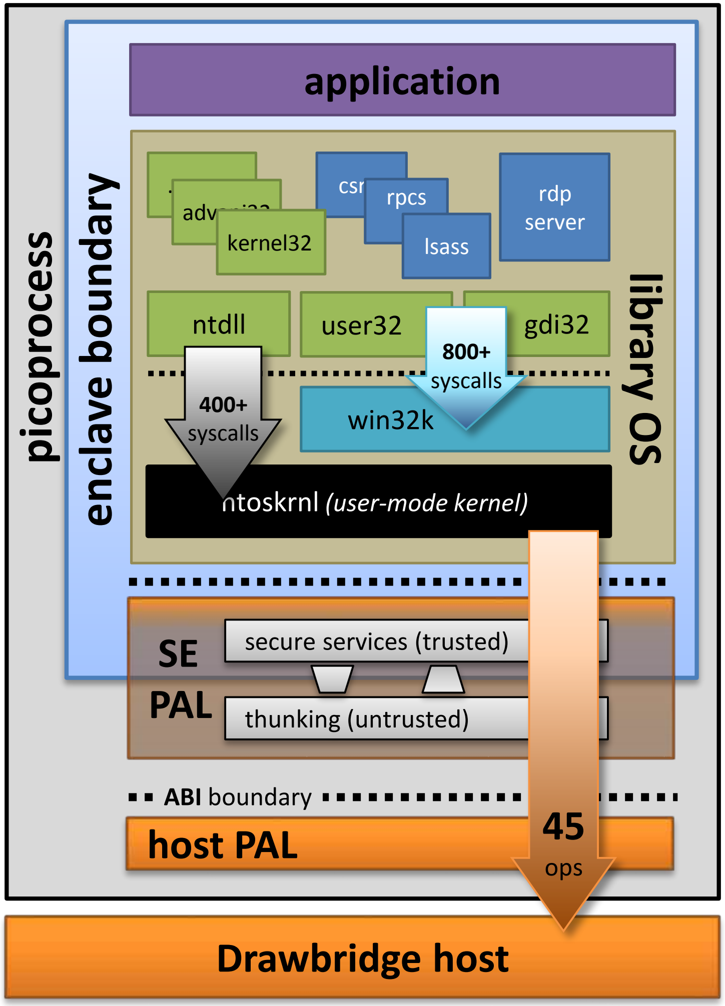
Sample Extensions [EuroSys 2013]

- Implemented:
  - Tracing
  - File system remapping
  - Checkpointing
  - Architecture adaptation
    - 32-on-64-bit x86
    - x86 on ARM JIT
- Discussed:
  - Speculation
  - Record and replay

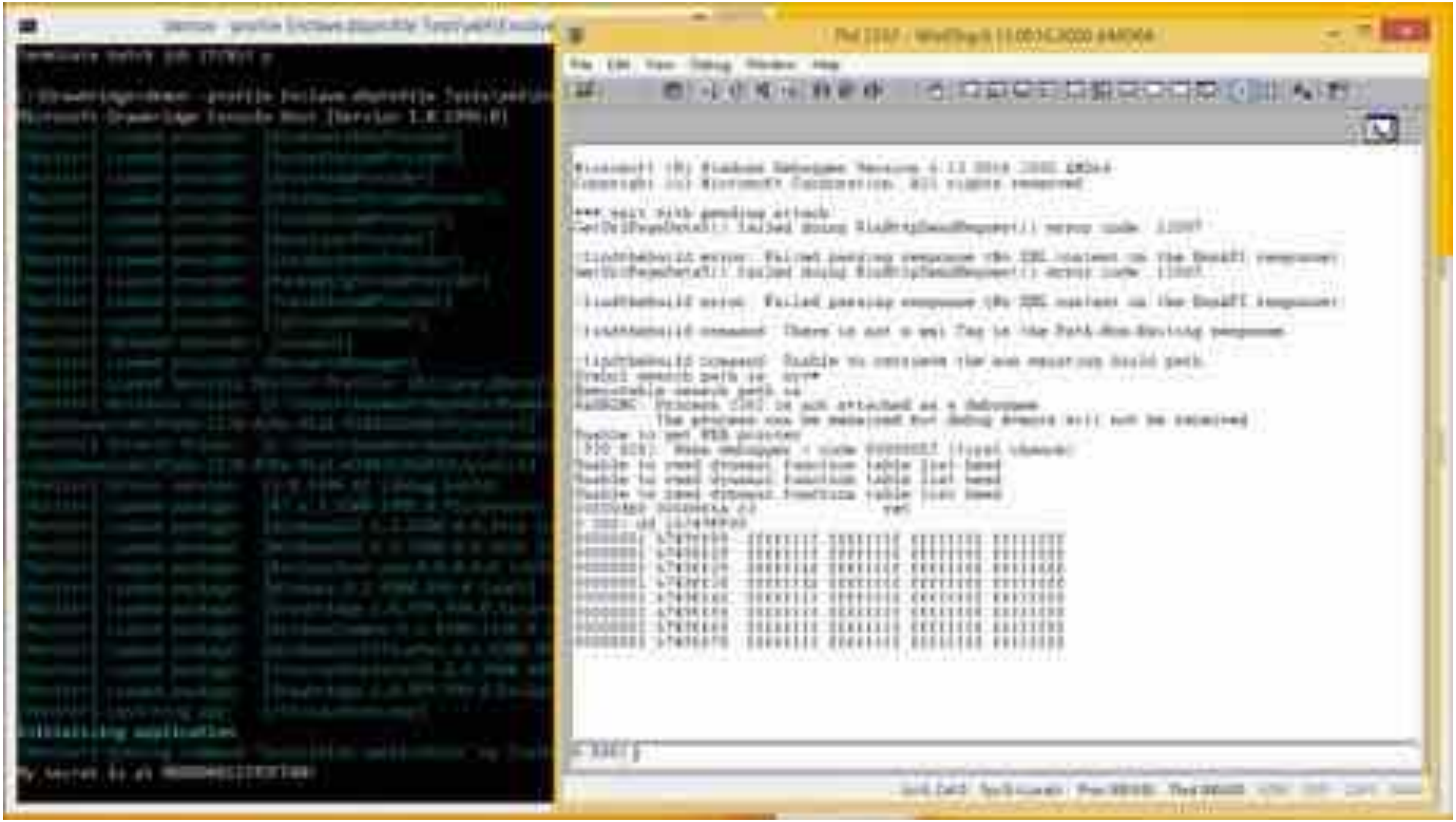
More Versatile: Intel SGX [SOSP 2013 Demo]

- Problem:
  - –Applications must trust:
    - OS, VMM, bootloader, BIOS, etc.
    - Administrator(s), management tools
  - –Hierarchical trust model is inadequate
  - –No practical way to protect private data

- Intel SGX [HASP 2013]
  - –New instructions & memory access changes
  - –Confidentiality and integrity protection
    - Protected *enclave* in user-mode process
    - Memory encryption
  - –Hardware support for **mutual distrust** in software
    - Remote attestation
    - Secure execution, despite compromised OS/VMM



Intel SGX Demo (Emulator)



```
Microsoft Drawbridge Console Host [Version 1.0.1996.0]
[illegible]
Initializing application
[illegible]
My secret is at [illegible]
```

```
Microsoft (R) Windows Debugger Version 6.13 [illegible] AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

*** wait with pending attach
[illegible]
Unable to read dynamic function table list head
Unable to read dynamic function table list head
Unable to read dynamic function table list head
[illegible]
```

Why should we love higher-level abstrations?

- Three reasons:
  - **Compatibility**
    - I can install my application in a picoprocess with the Library OS it requires and never have to worry about it breaking again.
  - **Security**
    - I can download even the most malignant code from the internet, run it in a picoprocess, and my system's integrity isn't lost.
  - **Continuity**
    - I can start an application (in a picoprocess) on one computer and move it to another, or reboot the computer, and it still runs.
- And three more:
  - **High Density**
    - I can run many hundreds of picoprocesses on one computer.
  - **Layerable**
    - I can write an extension that works with many hosts or library Oses.
  - **Versatility**
    - I can use higher-level abstractions where a VM can't run (cheaply).