Virtualization
COMP 252 - Lecture 5

Antoniu Pop

antoniu.pop@manchester.ac.uk

12 February 2019
Learning Objectives

▶ To describe the aims of virtualization— in the context of similar aims in other software components
▶ To distinguish between system and process virtualization
▶ To place system and process virtualization in the context of other virtualization technologies
▶ To understand how system, process and other virtualization technologies are likely to develop
Learning Objectives

▶ To describe the aims of virtualization – in the context of similar aims in other software components
Learning Objectives

- To describe the aims of virtualization – in the context of similar aims in other software components
- To distinguish between system and process virtualization
Learning Objectives

▶ To describe the aims of virtualization
  – in the context of similar aims in other software components

▶ To distinguish between system and process virtualization

▶ To place system and process virtualization in the context of other virtualization technologies
Learning Objectives

- To describe the aims of virtualization in the context of similar aims in other software components
- To distinguish between system and process virtualization
- To place system and process virtualization in the context of other virtualization technologies
- To understand how system, process and other virtualization technologies are likely to develop
Additional (optional) Reading

All available on the course materials webpage: http://syllabus.cs.manchester.ac.uk/ugt/2017/COMP25212/


Additional (optional) Reading

All available on the course materials webpage: http://syllabus.cs.manchester.ac.uk/ugt/2017/COMP25212/


Regarding memory subsystems:

Virtualization Technologies

➤ CPU
Virtualization Technologies

- CPU
- Virtual Memory
Virtualization Technologies

- CPU
- Virtual Memory
- Storage Virtualization
Virtualization Technologies

- CPU
- Virtual Memory
- Storage Virtualization
- Virtual Machines (e.g., Java)
Virtualization Technologies

- CPU
- Virtual Memory
- Storage Virtualization
- Virtual Machines (e.g., Java)
- System Virtualization (e.g., VMware, VirtualBox, XEN)
Virtualization Technologies – Objectives

Isolate details of hardware from the software that uses it.
Virtualization Technologies – Objectives

Isolate details of hardware from the software that uses it

- VM: amount of physical memory and layout

Sounds familiar?
Virtualization Technologies – Objectives

Isolate details of hardware from the software that uses it

- VM: amount of physical memory and layout
- Storage: position, size, and location of virtual disk
Virtualization Technologies – Objectives

Isolate details of hardware from the software that uses it

- VM: amount of physical memory and layout
- Storage: position, size, and location of virtual disk
- JVM: instruction set encoding, registers, etc
Virtualization Technologies – Objectives

Isolate details of hardware from the software that uses it

- VM: amount of physical memory and layout
- Storage: position, size, and location of virtual disk
- JVM: instruction set encoding, registers, etc
- System: I/O devices, memory, #CPUs
Virtualization Technologies – Objectives

Isolate details of hardware from the software that uses it

- VM: amount of physical memory and layout
- Storage: position, size, and location of virtual disk
- JVM: instruction set encoding, registers, etc
- System: I/O devices, memory, #CPUs

Sounds familiar?
Operating System and Virtualization

- Operating System isolates Application from Hardware
Operating System and Virtualization

- Operating System isolates Application from Hardware
- Operating System still closely integrated with hardware:
  - device drivers, interrupts, #CPUs, disk layout, etc
Operating System and Virtualization

- Operating System isolates Application from Hardware
- Operating System still closely integrated with hardware:
  - device drivers, interrupts, #CPUs, disk layout, etc
- Installing OS creates state
- Installing an application within OS creates state
- Moving an installed Application from one system to another is complex
- Moving an installed OS is very complex
- Moving a running application is almost impossible
Operating System and Virtualization

- Operating System isolates Application from Hardware
- Operating System still closely integrated with hardware:
  - device drivers, interrupts, #CPUs, disk layout, etc
- Installing OS creates state
- Installing an application within OS creates state
Operating System and Virtualization

- Operating System isolates Application from Hardware
- Operating System still closely integrated with hardware:
  - device drivers, interrupts, #CPUs, disk layout, etc
- Installing OS creates state
- Installing an application within OS creates state
- Moving an installed Application from one system to another is complex
Operating System and Virtualization

- Operating System isolates Application from Hardware
- Operating System still closely integrated with hardware:
  - device drivers, interrupts, #CPUs, disk layout, etc
- Installing OS creates state
- Installing an application within OS creates state
- Moving an installed Application from one system to another is complex
- Moving an installed OS is very complex
Operating System and Virtualization

- Operating System isolates Application from Hardware
- Operating System still closely integrated with hardware:
  - device drivers, interrupts, #CPUs, disk layout, etc
- Installing OS creates state
- Installing an application within OS creates state
- Moving an installed Application from one system to another is complex
- Moving an installed OS is very complex
- Moving a running application is almost impossible
Process vs. System Virtualization

- Process Virtualization:
  - Run a process under the control of a layer of software
  - e.g. JVM, Rosetta, Pin

- System Virtualization:
  - Run an operating system under the control of a layer of software
  - e.g. VMware, XEN, KVM, etc.
Process vs. System Virtualization

- **Process Virtualization:**
  - Run a process under the control of a layer of software
  - e.g. JVM, Rosetta, Pin

- **System Virtualization:**
  - Run an operating system under the control of a layer of software
  - e.g. VMware, XEN, KVM, etc
Virtualization can:

- Translate between equivalent facilities
- Change level of abstraction
- Multiplex/demultiplex resources
Virtualization can:

- Translate between equivalent facilities
  - Instruction Set Architecture? Library? System Calls?
- Change level of abstraction
- Multiplex/demultiplex resources
Taxonomy of Virtualization

Virtualization can:

- Translate between equivalent facilities
  - Instruction Set Architecture? Library? System Calls?

- Change level of abstraction
  - Garbage Collection? Virtual functions?

- Multiplex/demultiplex resources
Taxonomy of Virtualization

Virtualization can:

▶ Translate between equivalent facilities
  ▶ Instruction Set Architecture? Library? System Calls?

▶ Change level of abstraction
  ▶ Garbage Collection? Virtual functions?
  ▶ Performance tools? Debugging tools?

▶ Multiplex/demultiplex resources
Virtualization can:

- Translate between equivalent facilities
  - Instruction Set Architecture? Library? System Calls?

- Change level of abstraction
  - Garbage Collection? Virtual functions?
  - Performance tools? Debugging tools?

- Multiplex/demultiplex resources
  - Hide their physical number or quantity
Process Virtualization

- **JVM**
  - Interprets, then compiles “byte code” files
  - “Write once, run anywhere”
  - extensive libraries – extend OS API as Java standard
Process Virtualization

► JVM
  ► Interprets, then compiles “byte code” files
  ► “Write once, run anywhere”
  ► extensive libraries – extend OS API as Java standard

► Rosetta
  ► Translates PowerPC binaries “on-the-fly” to x86
  ► Maps PPC system calls to x86 (different calling conventions)
  ► Calls some native x86 procedures from PPC code
Process Virtualization

- **pin**
  - “annotate” Intel binary (www.pintool.org)
  - run a binary and collect (user-specified data)
Process Virtualization

- **pin**
  - “annotate” Intel binary ([www.pintool.org](http://www.pintool.org))
  - run a binary and collect (user-specified data)

- **valgrind**
  - “sandbox” Intel (++) binaries
  - check memory references and dynamic allocation
  - and lots of other analyses
Types of Virtualization

- Multiplexing
- Abstraction
- Translation
- Virtual Memory
- Storage Virtualization
- Network Virtualization
- JVM
- WINE
- Pin
- Valgrind
- Rosetta

Antoniu Pop – Virtualization
Adoption Model for Virtualization

- Introduce as Transparent Layer
  - Discover performance problems
Adoption Model for Virtualization

- Introduce as Transparent Layer
  - Discover performance problems

- Provide Management API
  - Initial focus: performance and manageability
  - Secondary focus: integration facilities
Adoption Model for Virtualization

- Introduce as Transparent Layer
  - Discover performance problems

- Provide Management API
  - Initial focus: performance and manageability
  - Secondary focus: integration facilities

- Provide full User-level API
  - Applications are built or integrated using API