Porting the QEMU virtualization software to MINIX 3

Master thesis presentation
July 16th, 2009
Erik van der Kouwe

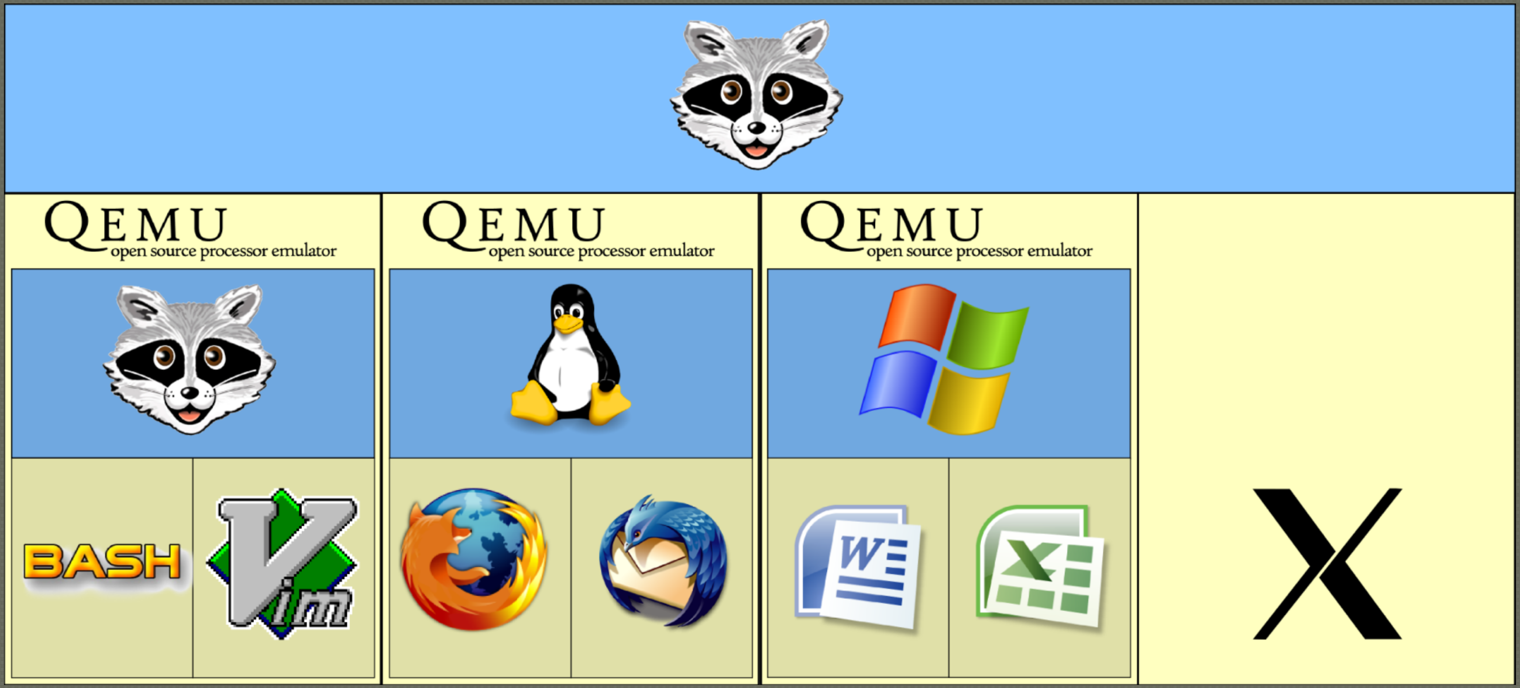
Computer Systems Section
Department of Computer Science
Faculty of Science
Vrije Universiteit Amserdam

Outline

- Introduction to virtualization
- Introduction to QEMU
- Introduction to MINIX 3
- Research questions
- Porting QEMU
- Testing QEMU
- Performance
- Conclusion

Introduction to virtualization

- One system emulates one or more VMs
  - Each runs its own guest operating system
  - VMs isolated from each other and from host


QEMU open source processor emulator
QEMU open source processor emulator
QEMU open source processor emulator
BASH
Vim

Introduction to virtualization

- Uses
  - Security research
  - Server farms
  - Software development
- Approaches
  - Native execution
    - Guest code run directly in reproduced environment
    - Sensitive instructions are a problem
  - Dynamic binary translation
    - Guest code translated into safe host code
  - Paravirtualization
    - Guest calls hypervisor to avoid sensitive instructions

Introduction to QEMU

- Open source virtualizer
- Uses dynamic binary translation
  - Alternative: direct execution with kernel module
- Advantages
  - General purpose full-system virtualization
  - Portable across hosts and guests
  - Entirely in user space
- Disadvantages
  - Slower than alternatives

QEMU
open source processor emulator

Introduction to MINIX 3

- Open source OS, built at the VU
- Microkernel
  - Reduce amount of high-privilege code
- Advantages
  - Simple and structured → suitable for education
  - Small and reliable → suitable for embedded systems
- Disadvantages
  - Few applications and drivers → small community
  - Many context switches → less performance

Research questions

- Can MINIX 3 run virtualization software?
  - What issues does one encounter when porting complex software to MINIX 3?
  - Is it necessary to change MINIX 3 to be able to run QEMU?
- Is the microkernel design an obstacle for performance?
  - Can bottlenecks be solved within this design?
  - Is QEMU usable on MINIX in practice?

Porting QEMU

- Use the right compiler
- Port packages QEMU depends on
- Allow QEMU to read MINIX binaries
- Functionality missing in MINIX
  - Add if essential for QEMU to work
  - Avoid using otherwise
- Debugging

Testing QEMU

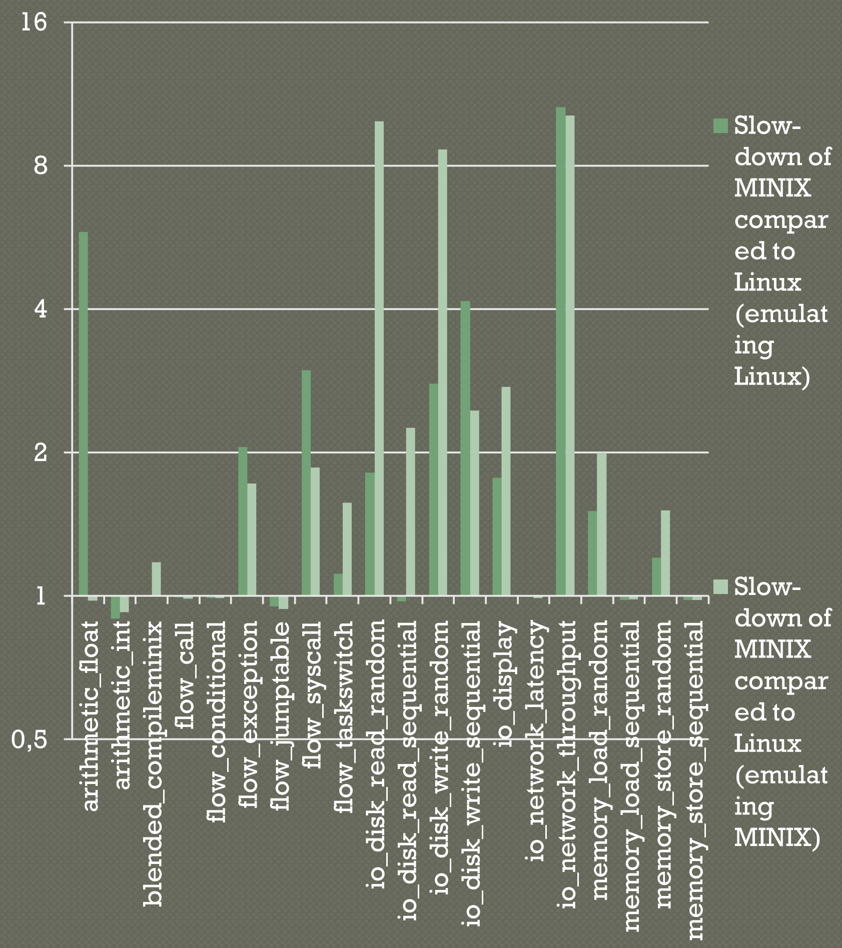
- Simply run many operating systems
  - MINIX (3.1.2a, 3.1.4)
  - Linux (Debian, Slackware)
  - Windows (95, 98)
- And browsers to test networking
  - Mozilla Firefox
  - Internet Explorer
- Findings
  - Clock resolution is an issue
  - Performance acceptable for all but Linux

Performance

- Benchmarks for various activities
  - Arithmetic, disk, display, interrupts, memory, network, task switching, …
- Configurations
  - Tested with MINIX 3.1.2a and Linux
  - Both used as host and guest (4 combinations)
  - Compared with native to find slow-down
- Overall slow-down just over 10×
  - Slightly worse than Linux

Performance

- Bottlenecks in MINIX
  - Floating point
    - FPU not supported
  - Disk input/output
    - Small disk cache
  - Graphics
    - No hardware acceleration
  - Interrupts
    - Setjmp/longjmp
  - Network throughput
    - Pauses while sending



Conclusions

- Yes, MINIX can run QEMU
  - But modifications are desirable
- Yes, performance is comparable to Linux
  - Most bottlenecks are unrelated to microkernel design
  - But: comparison based on pure binary translation
- Other results of research
  - Usable virtualization for MINIX
  - Manual for porting software to MINIX
  - List of additions/improvements desirable for MINIX

Thank you for your attention

Questions?