Efficient Guest Agnostic Virtualization With Embedded Power Architecture® KVM

Varun Sethi
Freescale Semiconductor
Motivation

- Growing interest in virtualization using KVM on embedded Power Architecture platforms

- Requirement to run various customer specific operating systems with embedded Power Architecture KVM

- Possible to run unmodified guest on embedded Power Architecture KVM – But this comes at a significant cost associated with VM exits
  - Problem severe for cores without virtualization assists

- Possible to “Binary translate” guest privileged instructions from the host side.
  - Continue to run an unmodified guest
Adaptive Binary Translation

• Dynamically infer the instructions that cause a large number of VM exits
• Binary Translate these instructions to a faster emulation code.
• Instruction translation maintained on a shared memory region
• “Memory tracing” (remove R/W permissions to shared page) implemented to control guest access to modified pages
  - Guest access to page generates a DSI exception
Design Challenges

• Handling complex instruction translations
  – Multi line patching could be complex
  – Translation cache placement issue

• Minimizing performance overhead (excessive DSIs) due to Memory tracing
  – Use of huge pages (TLB1)
  – Self referential code
    ▪ Access to sys call table and exception prolog in case of Linux
  – Self modifying code
    ▪ Code from the modified page trying to modify (Write) code on the same page
Solutions to Design Challenges

- Adaptive data mirroring algorithm to address self referential code
  - Data causing excessive DSIs mirrored to a guest memory region having R/W permission

- Addressing Translation Cache placement issue
  - Mechanism developed for stealing space from read only guest section
  - The data from the stolen section mirrored to new location
Addressing Memory tracing issues:
- Adaptive page resizing algorithm for addressing issue arising out of the use of Huge TLB and self modifying code
  - Dynamically splitting/merging traced pages
Performance

![Bar chart showing Linux Boot and Spawn 1000 echo processes for KVM without optimizations, KVM-PV, and KVM-BT.](image_url)
Summary

• With “Adaptive Binary Translation” it’s possible to run an unmodified guest efficiently with embedded Power Architecture KVM

• Possible to mitigate memory tracing overhead using “Adaptive Page Resizing” and “Adaptive Data Mirroring”

• Performance is comparable to the existing PV solution
Questions?