What is the right balance for performance and isolation with virtualization in HPC?

Thomas Naughton[1,2], Garry Smith[2], Christian Engelmann[1]

Geoffroy Vallée[1], Ferrol Aderholdt[3], Stephen L. Scott[1,3]

[1] Oak Ridge National Laboratory
[2] The University of Reading
[3] Tennessee Tech University

Overview

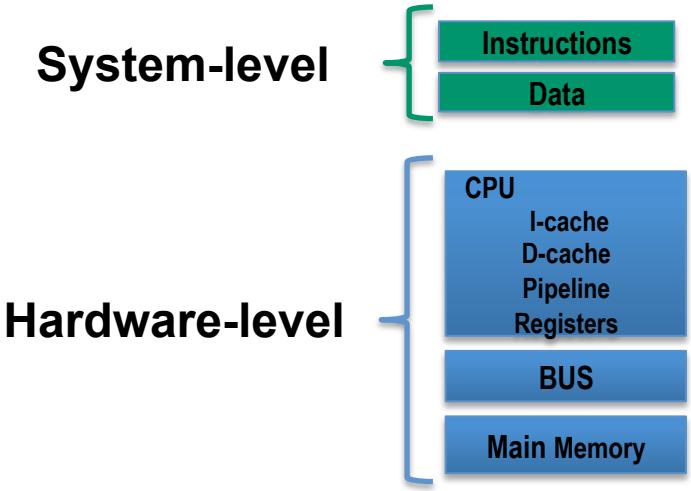
- HPC facing new challenges due to growing scale & complexity
  - Scalable algorithms
  - Fault tolerance
- HPC system software must balance
  - Performance / Usability / Robustness
  - System-level virtualization gaining attention in recent years
- Benefits of virtualization for HPC
  - User-customizable execution environment
  - Increased functionality
    - Specialized Micro-kernels vs. General Purpose kernels

Motivating High-level Questions

- What are the right policies and expectations for failures when using virtualization in HPC?
- How do performance & protection tradeoffs factor into resilience policies?
- How can we better organize the view of the platform to aid resilience policy choices?

Error Models

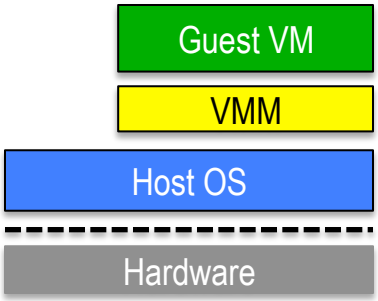
- HPC Resilience focused on gracefully coping with errors
  - Fault ➔ Error ➔ Failure
- Error model
  - Provides abstraction to aid reasoning about behavior
- Hardware/System model
  - Goloubeva et al. offer good description
  - Hardware errors manifest as system errors
    - i.e., errors in instructions or data



O. Goloubeva, M. Rebaudeng, M. S. Reorda, and M. Violante, Software-Implemented Hardware Fault Tolerance. Springer, 2006.

Error Models & Virtualization

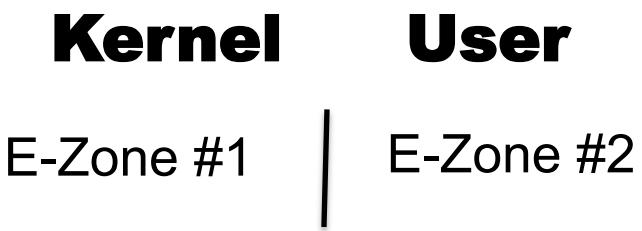
- Error models for HPC virtualization
  - VMs offer additional layer of indirection from hardware
    - Guest VM instructions & data
  - Help reason about behavior in this context
  - Aid study of performance / isolation problem



- Virtualization
  - VMs offer ability to increase isolation / protection (tradeoffs)

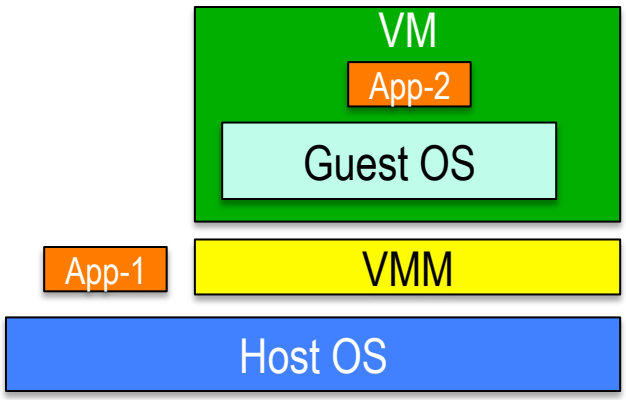
Error Zones

- "*Error zones*" are regions where faults may occur in the system
- Standard separation for protection has two zones
  - User-space
  - Kernel-space
- Control effects (scope) of failures
  - System crashes (global effects)
  - Process crashes (limited effects)



Error Zones

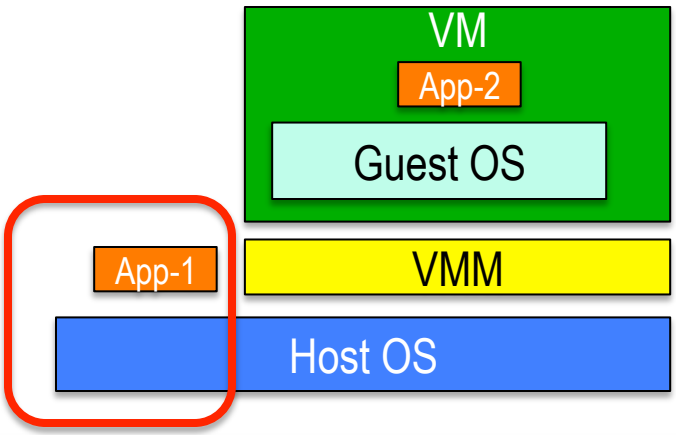
- New error zones for virtualized regions
  - Distinguish “Host” and “Guest” areas
- Additional regions offer more zones to control for failures



	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

Error Zones

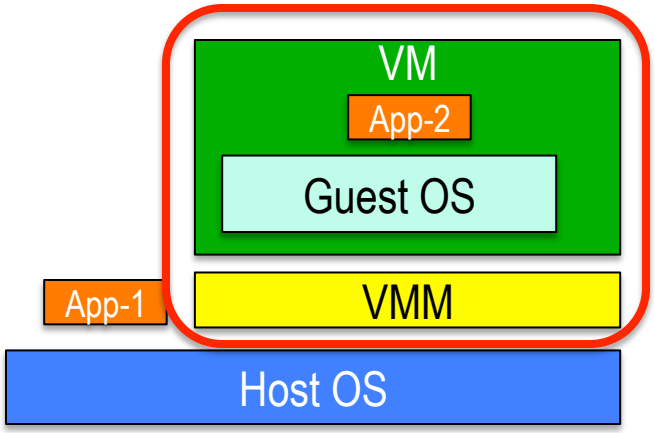
- New error zones for virtualized regions
  - Distinguish “Host” and “Guest” areas
- Additional regions offer more zones to control for failures
  - System crashes (global effects)
  - Process crashes (limited effects)



	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

Error Zones

- New error zones for virtualized regions
  - Distinguish “Host” and “Guest” areas
- Additional regions offer more zones to control for failures
  - System crashes (global effects)
  - Process crashes (limited effects)

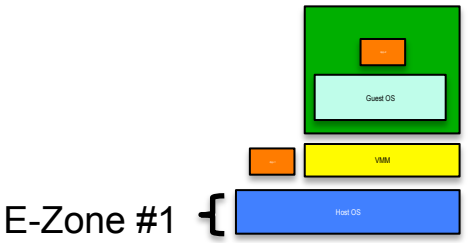


	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

Error model for virtualization-enabled context

A. Errors may crash full system (global effects)

	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

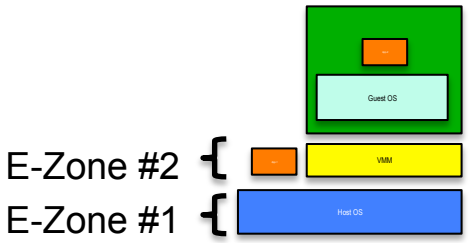


Error model for virtualization-enabled context

A. Errors may crash full system (global effects)

	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

B. Errors may crash individual process (limited effects)

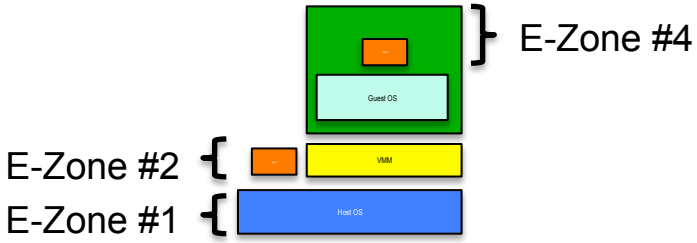


Error model for virtualization-enabled context

A. Errors may crash full system (global effects)

	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

B. Errors may crash individual process (limited effects)



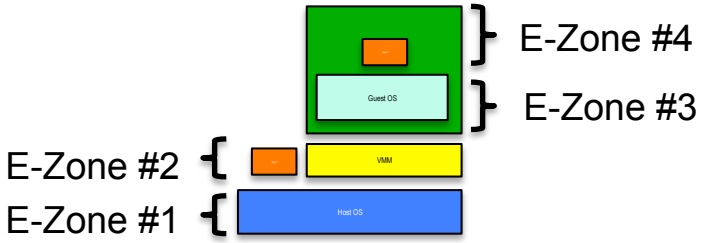
Error model for virtualization-enabled context

A. Errors may crash full system (global effects)

	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

B. Errors may crash individual process (limited effects)

A. Errors may crash full system (global effects)



Error model for virtualization-enabled context

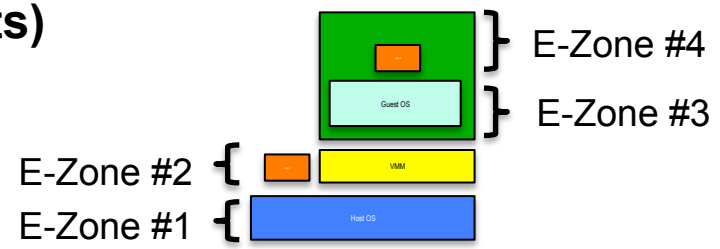
A. Errors may crash full system (global effects)

	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

B. Errors may crash individual process (limited effects)

A. Errors may crash full system (global effects)

B. Errors may crash individual process (limited effects)



Error model for virtualization-enabled context

A. Errors may crash full system (global effects)

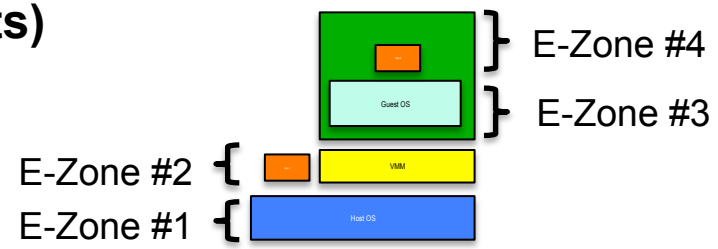
	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

B. Errors may crash individual process (limited effects)

A. Errors may crash full system (global effects)

B. Errors may crash individual process (limited effects)

C. Errors may be managed in mixed manner (varied effects)



Guest OS Errors

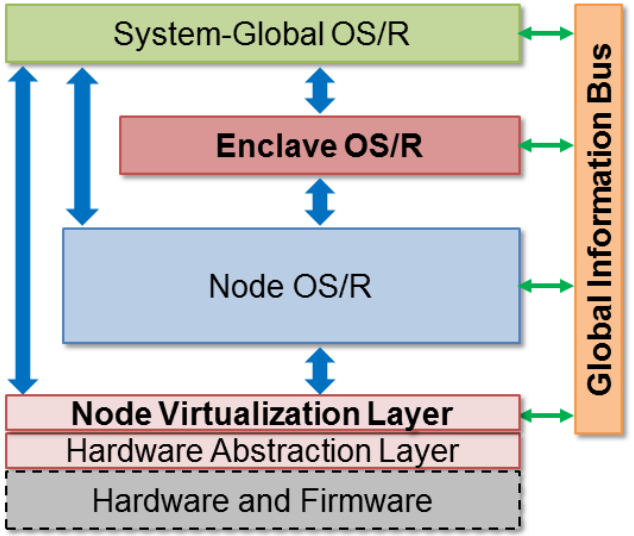
- What is proper dispositions for Error-Zone#3 ?
  - Example: Priority on protection, then E-Zone#3 ≈ E-Zone#2
  - Example: Priority on performance, then E-Zone#3 ≈ E-Zone#1

	Kernel	User
Host	E-Zone #1	E-Zone #2
Guest	E-Zone #3	E-Zone #4

- Consider this question in context of Hobbes OS/R project

Hobbes – Extreme Scale OS/Runtime

- Brief Synopsis
  - U.S. DOE project
    - Nat'l labs & universities
  - Design OS/Runtime interfaces for next generation machines
  - Two distinguishing elements
    - Enclaves & Composition
- Enclaves
  - Partition of the system allocated to a single application or service
  - Virtualization used to implement partitioning and isolation
- Composition
  - Joining applications/services to form more advanced instances
  - Mechanisms to relax isolation between enclaves to facilitate sharing between applications (in enclaves)



http://xstack.sandia.gov/hobbes

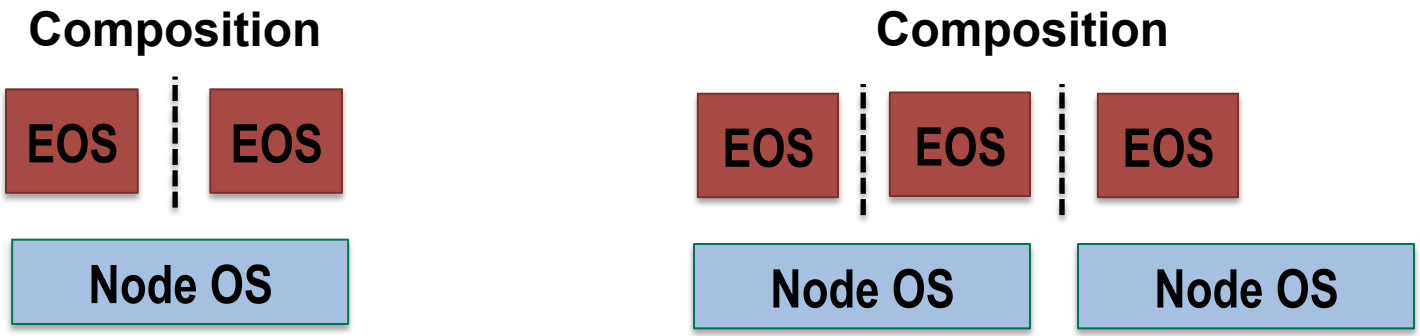
Considering E-Zones in Hobbes Context

- Important step in Hobbes resilience effort
  - Refine error models
  - Consider the two distinguishing elements from *Hobbes*
    - Enclaves & Composition
- What should be the E-Zone#3 policies for error management?
  - Will influence performance / isolation decisions
- One motivation for virtualization
  - Increased functionality (run a full feature OS as guest OS in E-Zone#3)
    - Could assume: E-Zone#1 == E-Zone#3

Considering E-Zones in Hobbes Context (2)

- Composing different enclaves may be less straightforward
  - Crashing 1 enclave (VM) may be ok
  - Crashing multiple enclaves (VMs) may be un-acceptable
    - Add protections to limit cross-enclave interactions

- Example with few Enclave OS (EOS) instances



Thoughts on Resilience in Hobbes

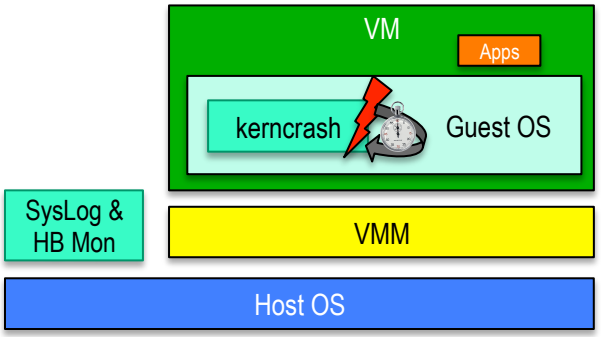
- Error-Zone disposition should be factored into enclave setup
  - Possibly offer ability to tailor degree of protection in enclave interfaces
  - Possibly have Node OS dictate allowable degree of protection when setting up enclaves
  - Possibly avoid protection if low/no composition
    - Maybe defer until composition is requested
- Need isolation tests
  - Develop hooks into interfaces & tests for probing
  - Offer ability to perform fault-injection & robustness testing

Evaluation: Error-Zone#3 Tests

- Considered three virtualization systems
  - QEMU – entirely user-space
  - KVM – kernel module (general purpose focused)
  - Palacios – kernel module (HPC focused)

- Synthetic guest kernel error (E-Zone#3)
  - QEMU & KVM isolated
    - E-Zone#3 had limited effects, crash is contained
  - Palacios isolated
    - E-Zone#3 had some host effects

    (*possible bug: shared host/vmm networking*)



Summary

- HPC system software
  - Tradeoffs between Performance / Usability / Robustness
  - Leverage virtualization for user-customization & added functionality
- Error models in context of HPC virtualization
  - Classified into "**Error Zones**"
  - Provides abstraction to help reason about expected behavior
- Performance & isolation in HPC resilience
  - Analysis of performance / isolation using error-zones
  - Impact on resilience strategies in context of *Hobbes* project
  - Experiments to demonstrate effects of synthetic errors

Thank you & enjoy the conference

Managed by UT Battelle, LLC under Contract No. De-AC05-00OR22725 for the U.S. Department of Energy.