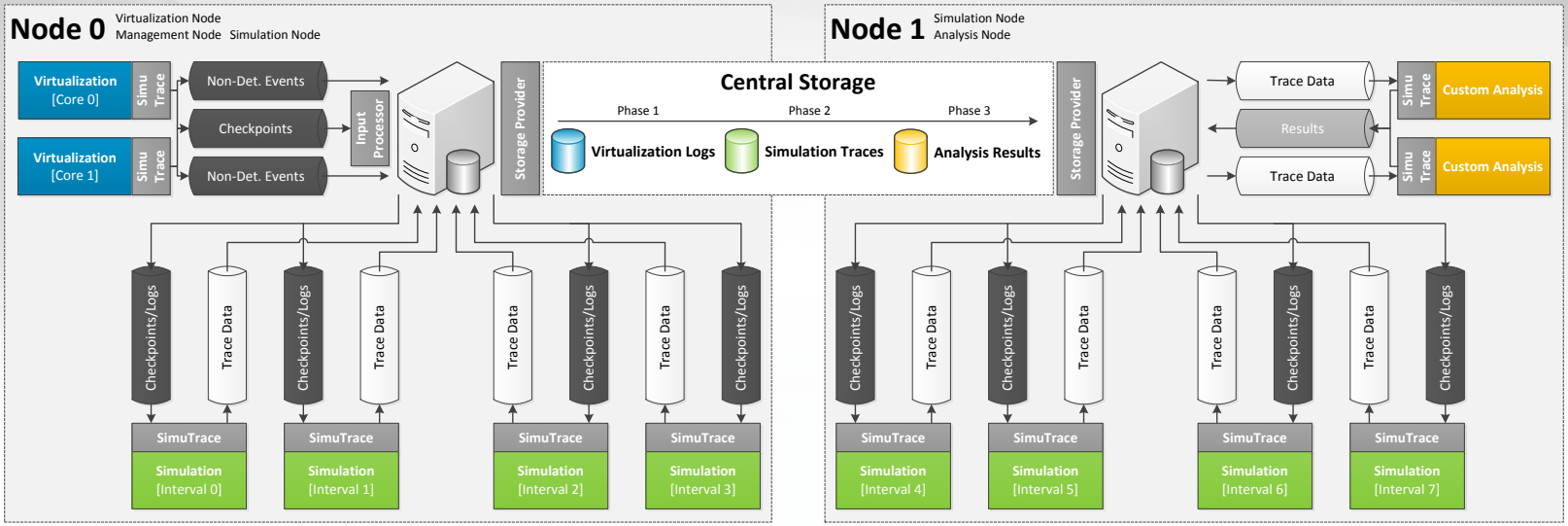
Karlsruhe Institute of Technology

SimuBoost: Scalable Parallelization of Functional System Simulation

GI Fachgruppentreffen Betriebssysteme (BS) 2013

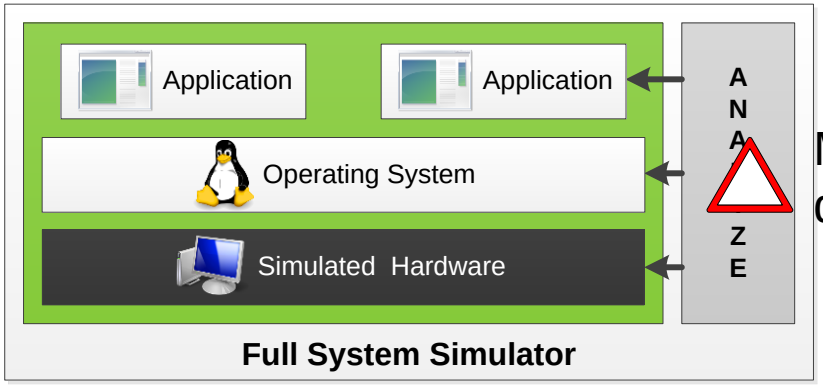
Marc Rittinghaus, Konrad Miller, Marius Hillenbrand, Frank Bellosa

SYSTEM ARCHITECTURE GROUP
DEPARTMENT OF COMPUTER SCIENCE



Motivation

- Operating system performance analysis:
  - Application and kernel interaction
  - Memory access patterns
  - Cache efficiency



Measurements distort results

- Approach: Functional System Simulation/Emulation
  - Simulate physical machine at functional-level (instructions)
  - Monitor states/operations non-intrusively

Functional Simulation is Slow

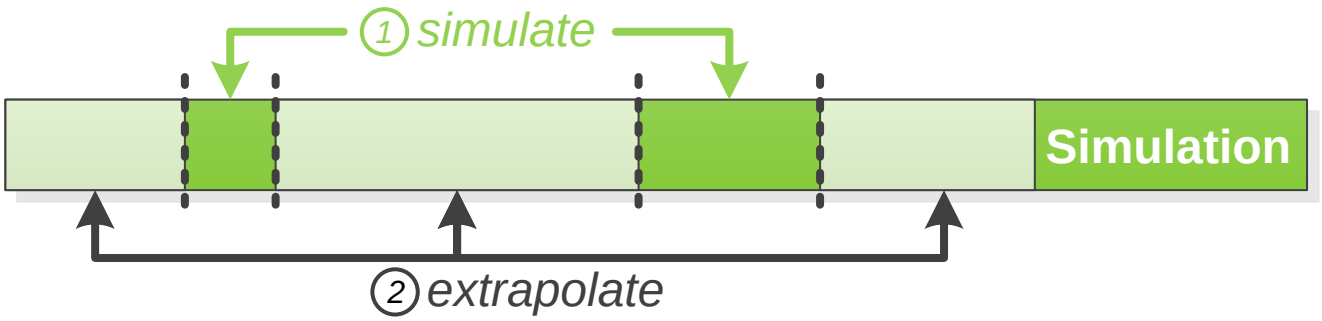
- Average slowdowns for: Kernel build, SPECint_base2006, LAMMPS

Virtualization	Simulation	
KVM	QEMU	Simics
~ 1x	~ 100x	~ 1000x

- Example: Analyze memory duplication in kernel build
  - Memory access patterns on shareable pages
  - Operations that lead to breaking merged pages
  - Our experience with Simics: **30 min -> 10 months**

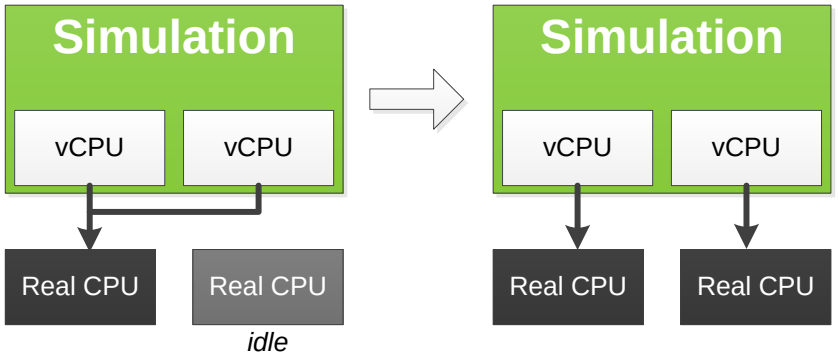
Problem: Not practical for long-running workloads

Accelerating Simulation: Sampling



- Simulate representative samples and extrapolate (SimPoints[2])
  - There may be no representative intervals
    - Not even all applications show phase behavior (gcc [3])
    - Even less probable for whole system (i.e., mix of multiple applications)
  - Chicken-and-Egg Problem
    - How do you find representative intervals without analyzing first?

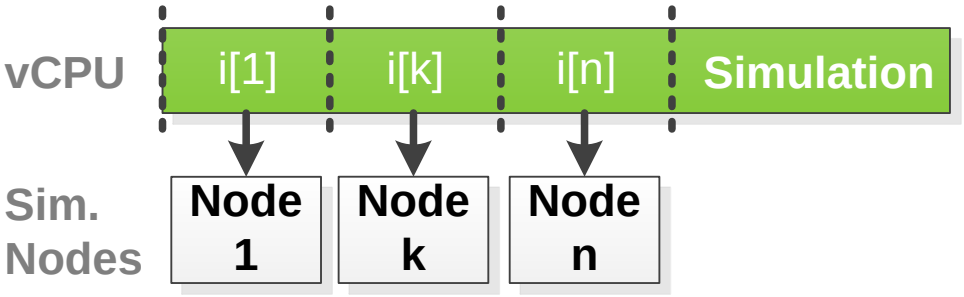
Accelerating Simulation: Parallel vCPUs



- Simulate vCPUs in parallel (e.g., PQEMU[1])
  - Scales in number of vCPUs (e.g., 4x → still 2.5 months)
  - Does not accelerate single-CPU simulation

Goal: Scale-out single-core simulation

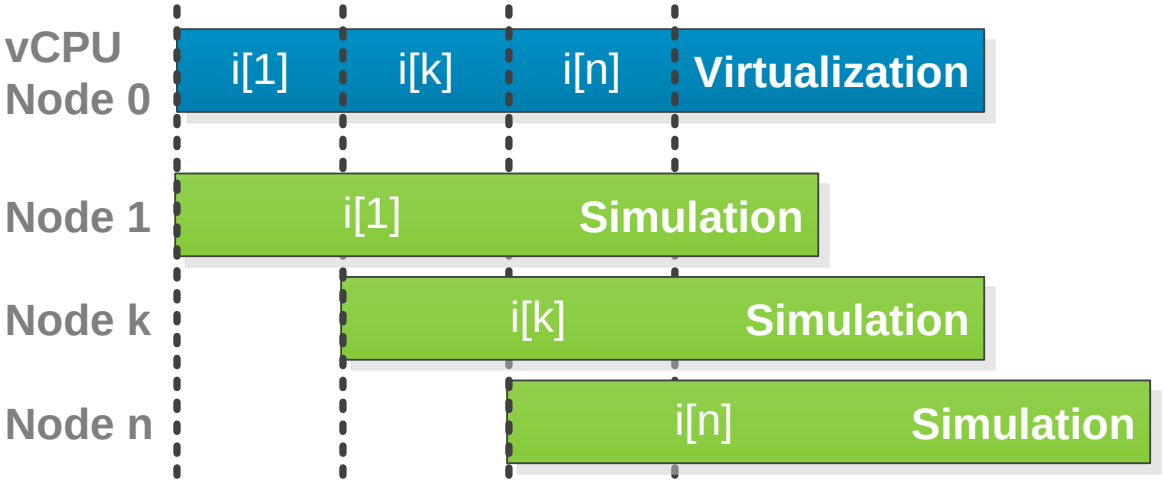
Basic Approach



- (1) Split simulation into time intervals
  (2) Simulate intervals simultaneously
  - Scales with run-time of workload
  - Applicable to single-CPU simulations

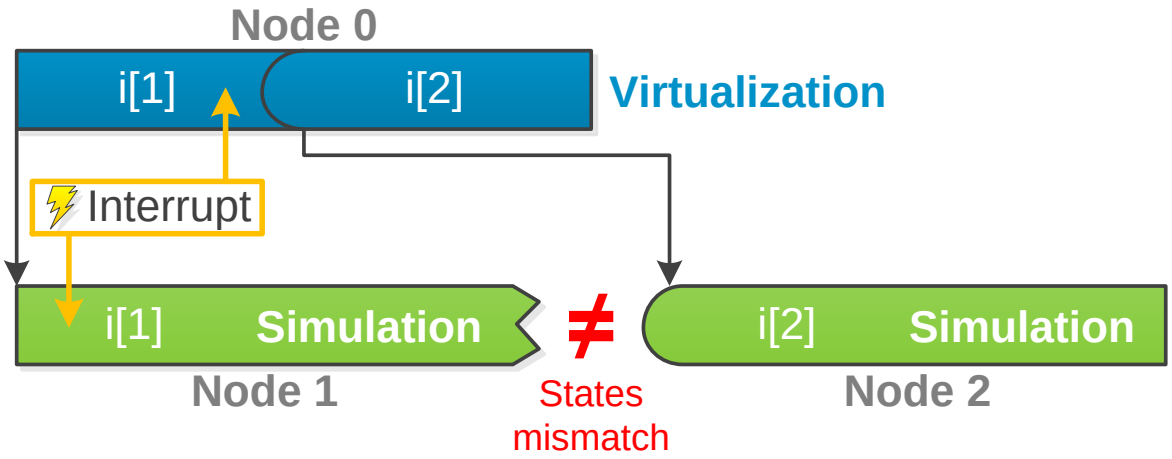
Problem: How do we bootstrap the simulation of i[2..n]?

SimuBoost



- Leverage fast virtualization
  - Create checkpoints at interval boundaries
  - Checkpoints bootstrap simulations:
    - Memory, device states, etc.
  - Run simulations in parallel

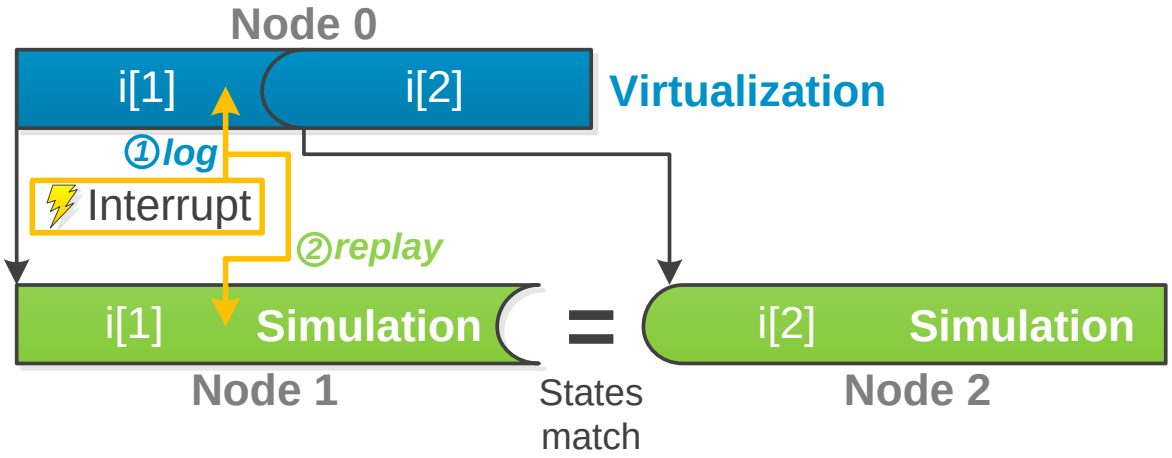
State Deviation



- Devices work asynchronous to CPU
  - Different I/O data and completion timing
- Virtualization and simulation drift apart

Problem: Machine states differ at interval boundaries

Coping with State Deviation



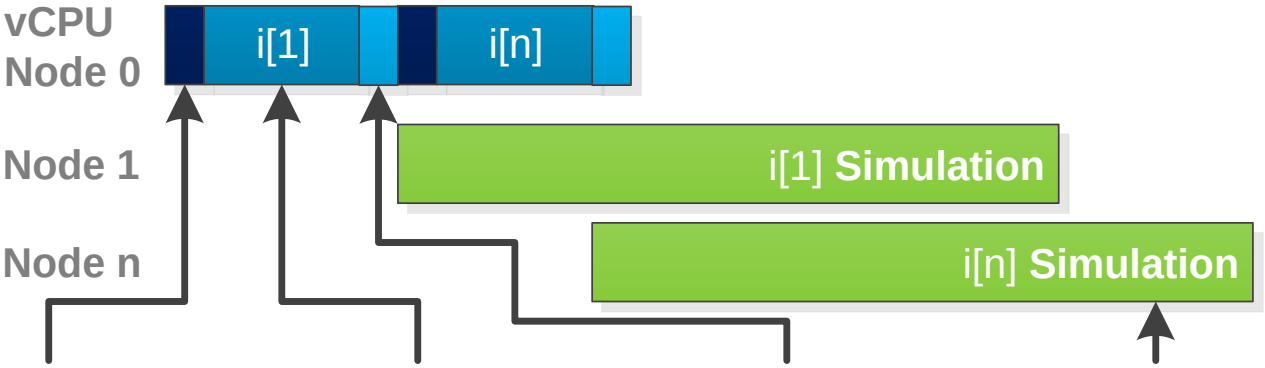
- (1) Trap and log non-deterministic events in the hypervisor
  (2) Precisely replay events in the simulation

- Non-deterministic events (e.g., interrupts, timing instructions)
  - …appear at equal points in the instruction stream
  - …produce same data output

Virtualization and simulation stay sychronized

Implementation

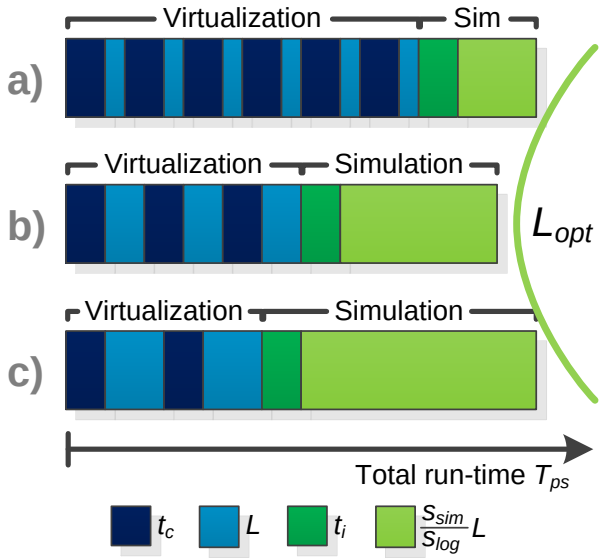
- Implementation is work in progress − components available:
  - Fast virtualization (e.g., KVM [4])
  - Fast logging of non-deterministic events (e.g., ReVirt [5], Retrace [6])
  - Lightweight checkpointing (e.g., Remus [7])
  - Functional simulation (e.g., QEMU [8], Simics [9])



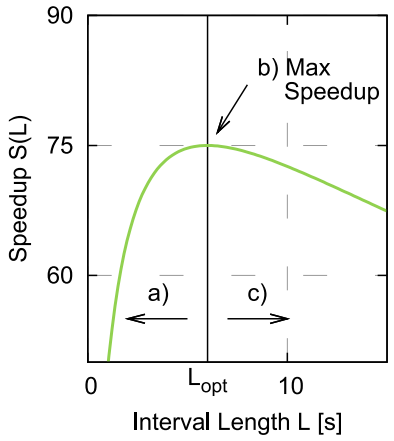
Checkpointing	Virtualization	Logging	Simulation
100 ms [7]	≈ 1x	0-8% [5], 5% [6]	100x slowdown

Speedup and Scalability

- Right interval length is crucial
  - Too short (a):
    - Checkpoint time dominates
  - Too long (c):
    - Little parallelization
    - Long simulation of final interval



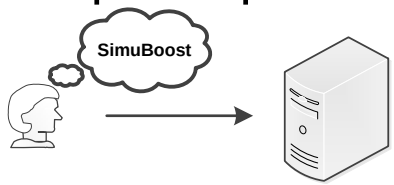
- Example scenario:
  - Basis: Performance of available components
  - Optimal interval length: 2s
  - Best possible speedup for 1h workload:
    84x @ 90 nodes (94% parallel efficiency)



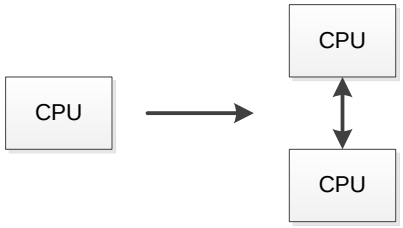
Near linear speedup possible

Open Questions

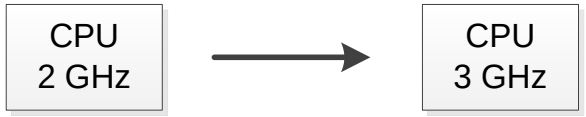
- Can we reach theoretical speedups in real simulations?



- Can multi-core/multi-socket simulations benefit from SimuBoost?
  - Capturing non-deterministic events is challenging (shared memory)



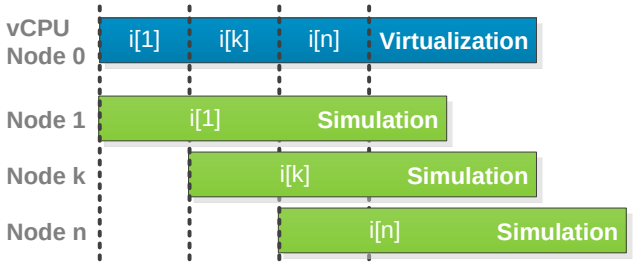
- Can we simulate a machine with different hardware characteristics than the host?
  - SimuBoost replicates behavior of virtual machine



Conclusion

- Slowdown of Functional Full System Simulation: >100x

- SimuBoost: Accelerate simulation
  - Run workload with fast virtualization
  - Take checkpoints in regular intervals
  - Start parallel simulations on checkpoints
  - Logging and replay of non-deterministic events



- Advantages:
  - Scales with workload run-time (also for single-core simulations)
  - High scalability and parallel efficiency possible (84x @ 90 nodes, 94%)

SimuBoost: Detailed Full System Simulation made practical

Functional Simulation Slowdown

- Functional System Simulation is slow
  - Time to Completion [h] (Slowdown):

	Native	Hw.-Virt. KVM	Simulation		
			QEMU[1]	QEMU[2]	Simics[1]
Linux 3.7.1 Kernel Build					
	1.44	1.56 (1.08x)	47 (33x)	238 (165x)	1080 (771x)
SPECint_base2006 1.2					
	6	6.29 (1.05x)	133 (22x)	1243 (207x)	6216 (1036x)
LAMMPS Lennard Jones					
	1.82	1.65 (0.91x)	69 (38x)	204 (113x)	1123 (624x)
		Ø 1x	**Ø 31x**	**Ø 162x**	**Ø 810x**

[1] Empty memory hooks [2] Counting unique accessed physical pages per second

SimuBoost: Job Distribution

- Intervals are independent jobs
  - Distribute jobs across nodes
  - One virtualization node
  - Many simulation nodes
  - (One controller node)

- Can we simulate a single core on a multi-core node in its native execution time?

References

- [1] Ding et al. ’11 *PQEMU: A parallel system emulator based on QEMU*
- [2] T. Sherwood et al. Automatically characterizing large scale program behavior. volume 30. ACM, 2002.
- [3] V. Weaver et al. Using dynamic binary instrumentation to generate multi-platform simpoints: Methodology and accuracy. HiPEAC, 2008.
- [4] A. Kivity et al. kvm: the linux virtual machine monitor. volume 1. Linux Symposium, 2007
- [5] G. Dunlap et al. Revirt: Enabling intrusion analysis through virtual-machine logging and replay
- [6] M. Sheldon et al. Retrace: Collecting execution trace with virtual machine deterministic replay. 2007
- [7] M. Sun et al. Fast, lightweight virtual machine checkpointing. 2010.
- [8] F. Bellard. Qemu: A fast and portable dynamic translator. USENIX, 2005
- [9] P. Magnusson et al. Simics: A full system simulation platform. Computer, 35(2), 2002