Karlsruhe Institute of Technology

System Architecture Group
http://os.itec.kit.edu/

Simutrace
http://simutrace.org/

SimuBoost: Scalable Parallelization of Functional System Simulation

Marc Rittinghaus, Konrad Miller, Marius Hillenbrand, Frank Bellosa

Motivation

- Want: Operating system performance analysis
  Application and kernel interaction, memory access patterns, …
- Need: Functional **full system simulation** to monitor system non-intrusively
- Challenges:
  Functional system **simulation too slow** for long-running workloads.

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

Average slowdown for: Kernel build, SPECint_base2006, LAMMPS

Loss of interaction with non-simulated remote hosts.

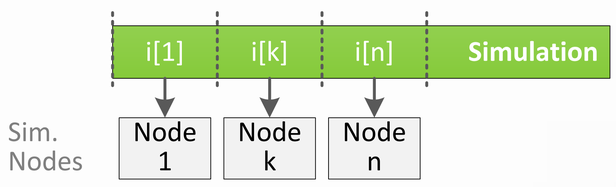
Existing Techniques

- Sample and extrapolate
  [Sherwood et al. SimPoints]
  Not all applications show phase behavior (gcc) [Weaver et al.].
  Less probable for whole system.
  How to find phases without using simulation first?
- Parallel multi-core simulation
  [Ding et al. PQEMU]
  Only scales in number of simulated CPUs.
- Reduce workload
  [KleinOsowski et al. MinneSPEC]
  Not always possible.

Goal: Scale-out single-core functional full system simulation

Approach

Basic Idea

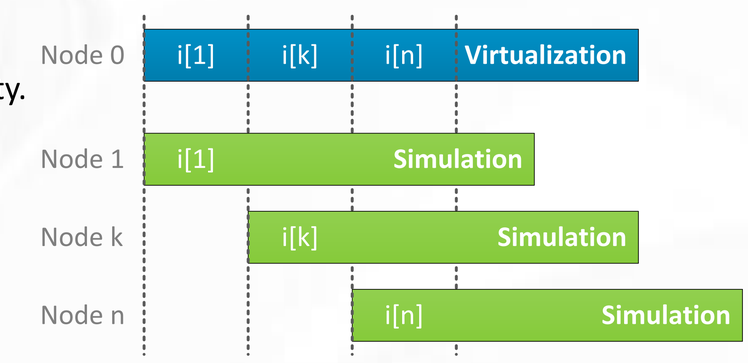


- (1) Split simulation time
- (2) Simulate intervals in parallel
  - Scales with the run-time of the workload.
  - Applicable to single-core simulations.

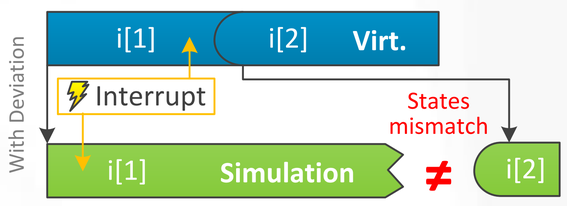
How to bootstrap the simulation of i[2..n]?

SimuBoost

- Run workload in virtual machine
  Preserves interactivity and network connectivity.
- Create checkpoints at interval boundaries
  to bootstrap simulations
- **Run simulations in parallel**
  Distribute jobs across machines.

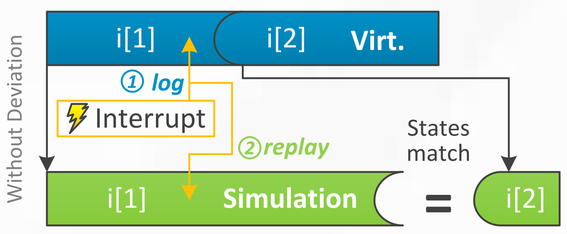


Functional Continuity



- Virtualization introduces non-determinism
  Different I/O timing and data between stages.
  - Virtualization and simulation drift apart

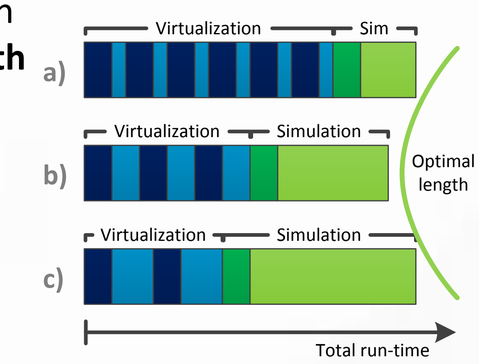
Preserving Functional Continuity



- (1) Trap and **log** non-deterministic events
  Interrupts, timing instructions, …
- (2) Precisely **replay** events in the simulation
  [Dunlap et al. ReVirt, Sheldon et al. Retrace], Overhead: <8%
  - **Virtualization and simulation stay sychronized**

Speedup and Scalability

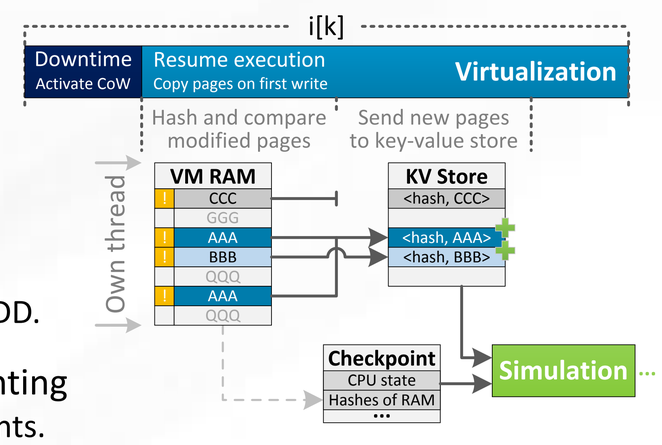
- Speedup depends on **speed difference** between virtualization and simulation, and **interval length**
  - Minimize virtualization overhead…
    (logging and checkpointing)
  - …and calculate optimal interval length from speed difference and overheads.



- Predicted **speedup** for 1h workload: **84x @ 90 nodes** (94% parallel efficiency)
  100x slowdown, 100ms downtime/checkpoint [Sun et al. Remus], 8% logging overhead, 1s start-up delay

Lightweight Checkpointing

- Goals:
  Short downtime, small checkpoints
  Easy & fast access
- **Copy-on-write** checkpointing
  Resume VM *before* saving memory & HDD.
- **Incremental, hash-based** checkpointing
  Deduplicate within and across checkpoints.
  Of modified data, we can deduplicate:
  - RAM pages: 5%-40%
  - Disk blocks: 35%-80%



- Access via **key-value store**
  Store <hash, page/block data>-pairs.
  Checkpoint = list of hashes.