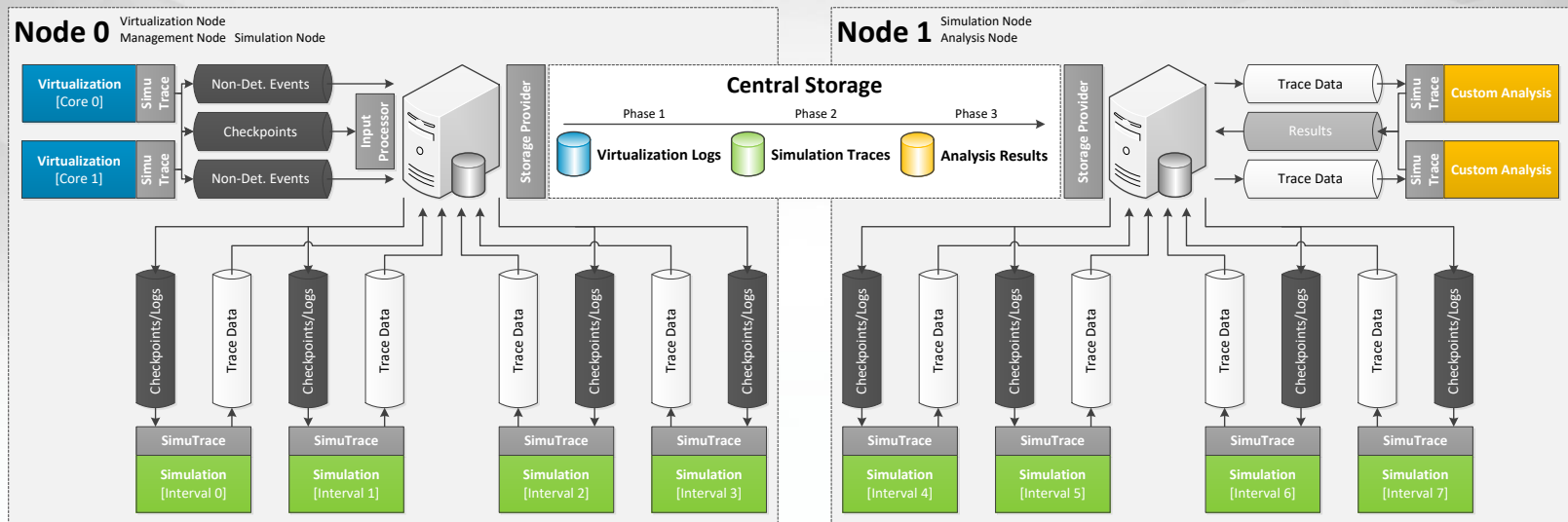


Towards Scalable Parallelization of Functional System Simulation with SimuBoost

GI Fachgruppentreffen Betriebssysteme (BS) 2016
 Marc Rittinghaus, Frank Bellosa

OPERATING SYSTEMS GROUP
 DEPARTMENT OF COMPUTER SCIENCE



Motivation

- Study properties of redundant memory contents [Miller13]
 - Origin? Lifetime? Sharing possible?
 - Analyze memory contents after each modification
 - But: Analysis should not affect workload
- Analyze memory access patterns on system interfaces [Jurczyk13, Wilhelm15]
 - Detect vulnerabilities in Windows 8 and Xen (CVE-2015-8550)
 - Trace individual memory reads and writes

We want detailed runtime information

Motivation

- Operating system research
 - Debugging
 - Application, OS, and hardware interaction
 - Malware and vulnerabilities

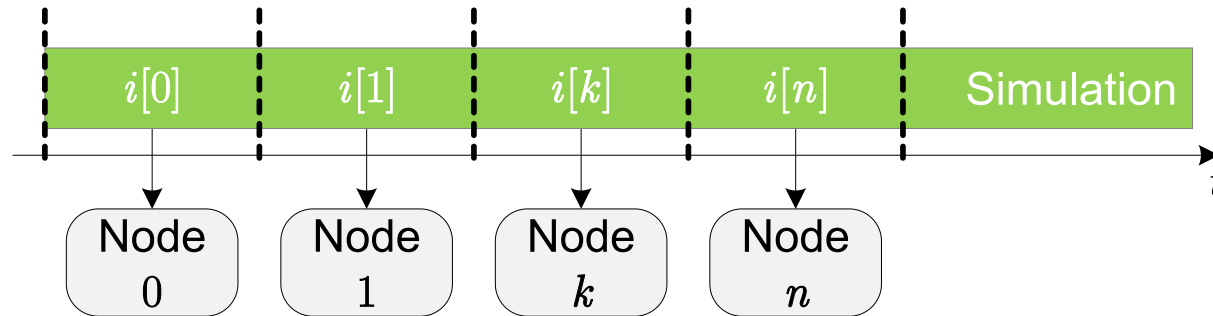
- Functional Full System Simulation
 - But: It is slow

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

Average slowdowns for: kernel build, SPECint_base06, LAMMPS

- **Not practical for long-running workloads**
- **Loss of interactivity (users and remote hosts)**

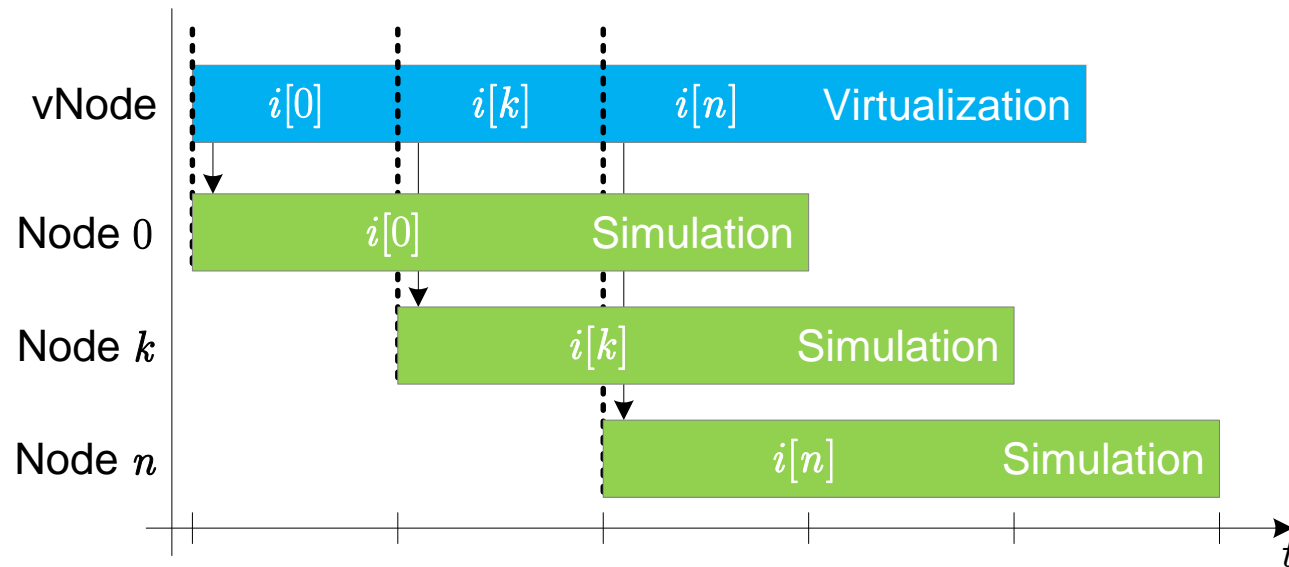
Basic Acceleration Approach



- (1) Split simulation into time intervals
- (2) Simulate intervals simultaneously
 - Does not trade accuracy for speed
 - Applicable to single-CPU simulations
 - Scales with run-time of workload

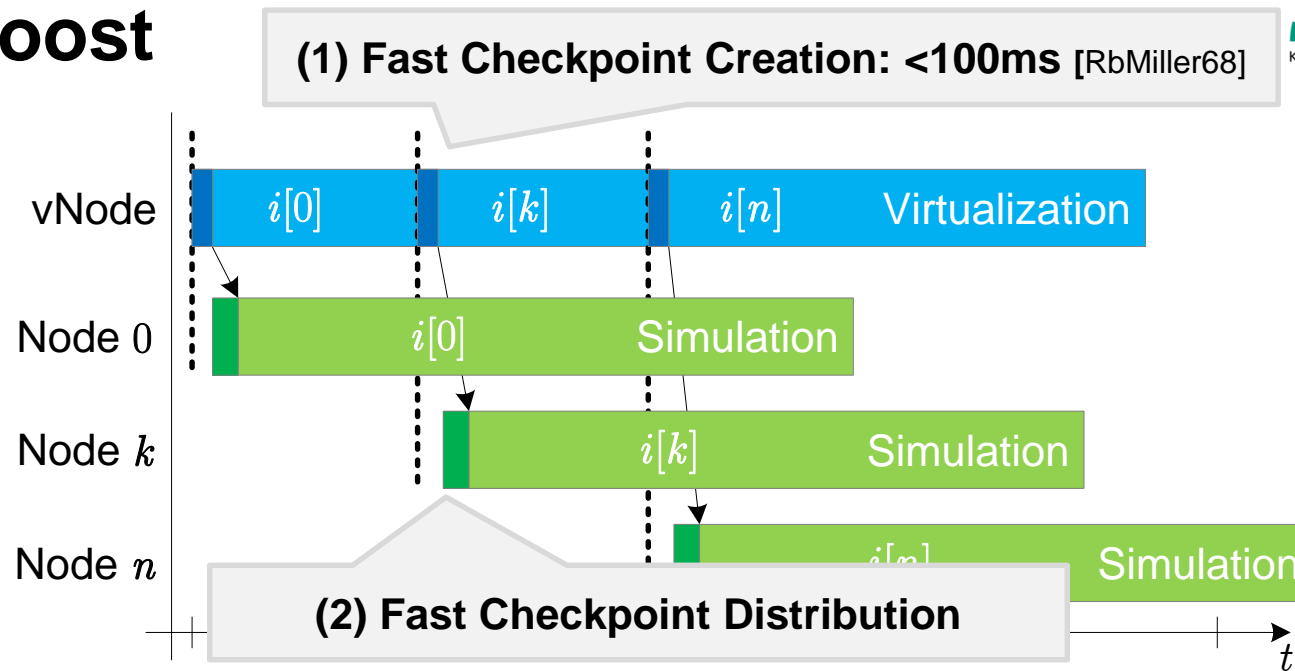
- **How to bootstrap the simulation of $i[1..n]$?**
 - **Still no interactivity**

SimuBoost



- Leverage fast virtualization
 - Checkpoints at interval boundaries bootstrap simulations
 - Hardware acceleration provides full interactivity
 - Speed difference drives parallelization

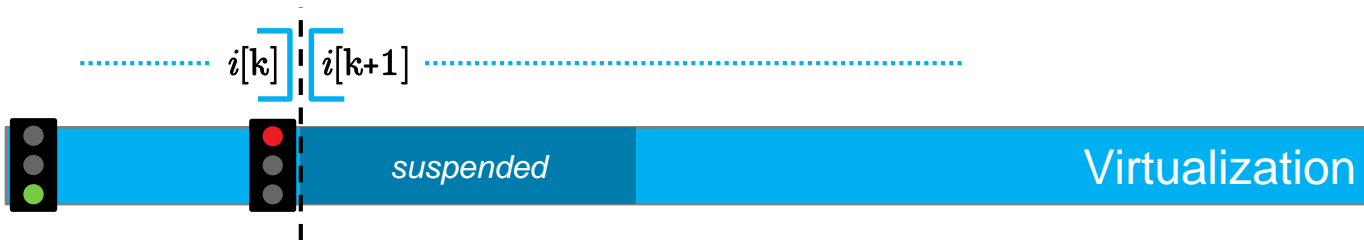
SimuBoost



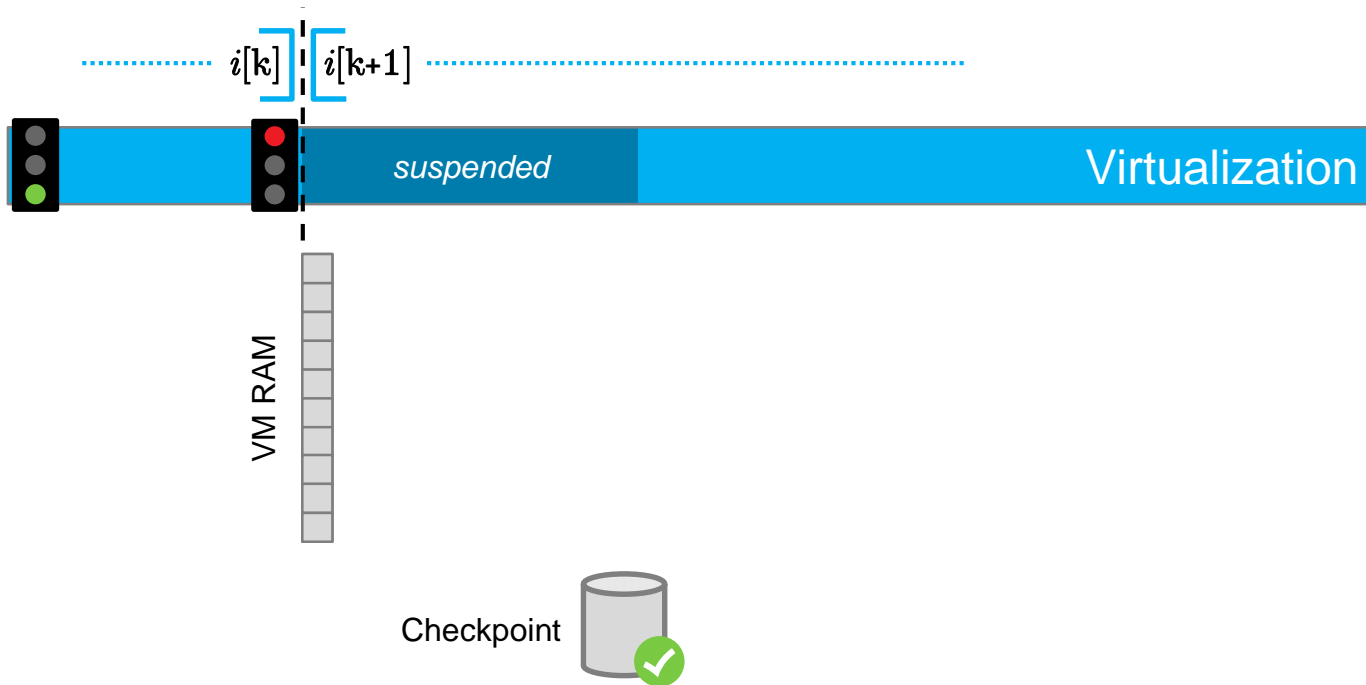
- Leverage fast virtualization
 - Checkpoints at interval boundaries bootstrap simulations
 - Hardware virtualization provides full interactivity
 - Speed difference drives parallelization

Challenges: Preserve interactivity and speedup

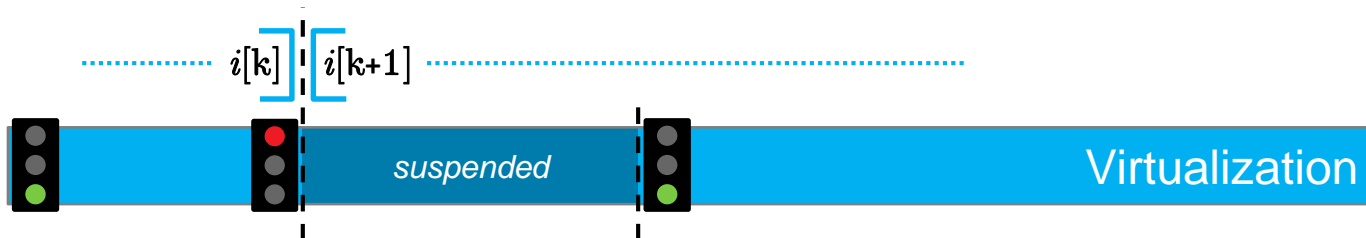
Stop-And-Copy



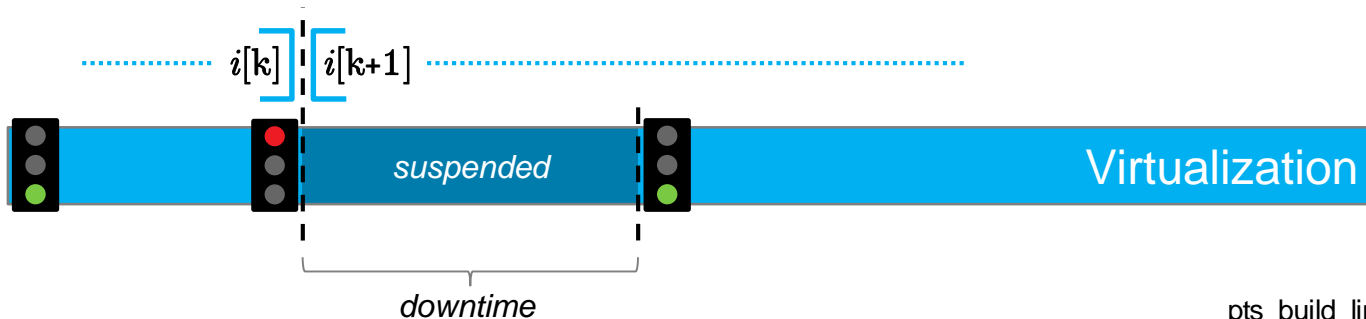
Stop-And-Copy



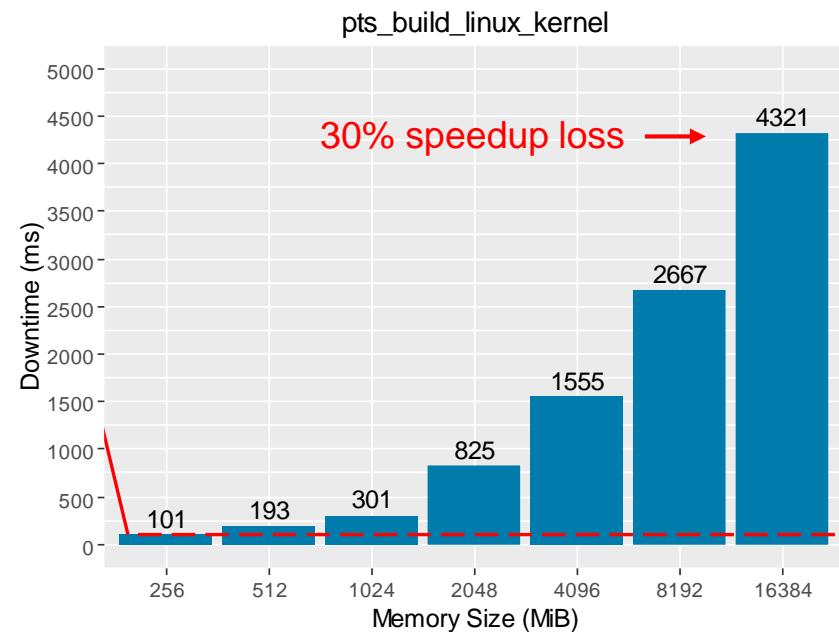
Stop-And-Copy



Stop-And-Copy

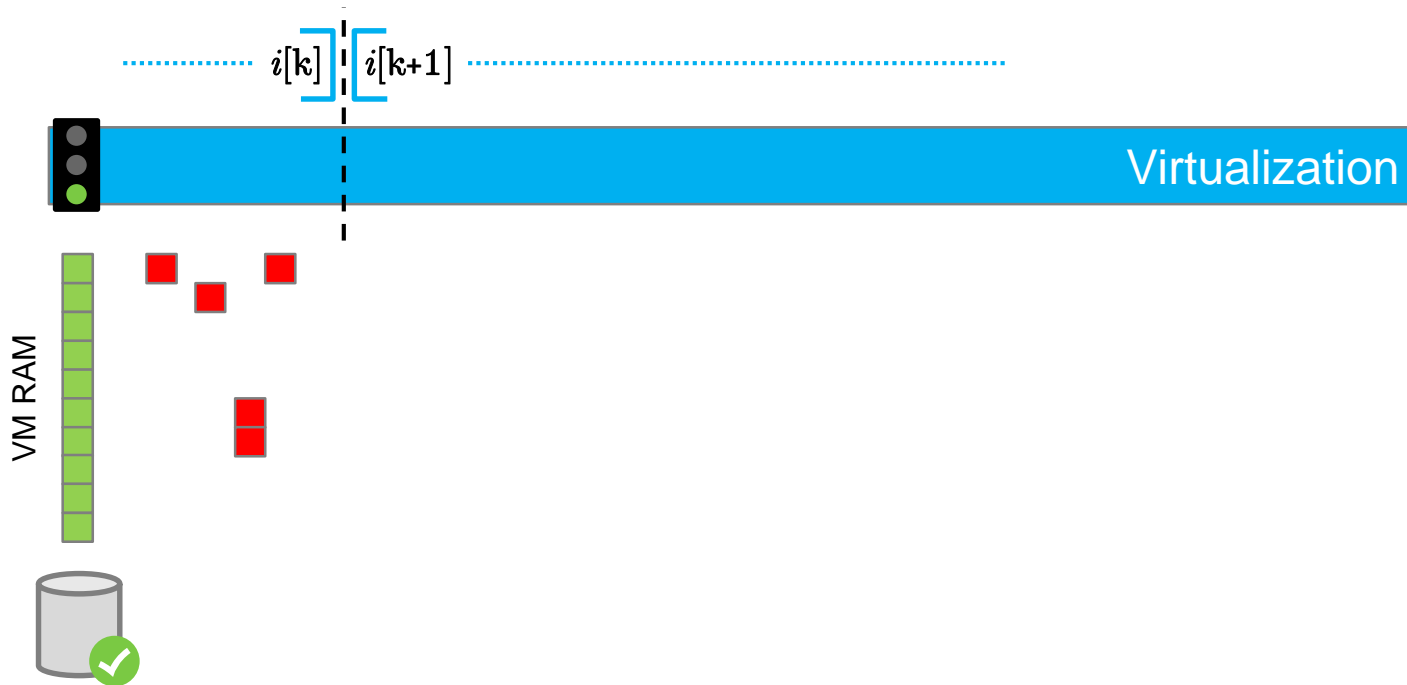


- Downtime depends on VM size
- Not suited for interactive use
- Limited parallelization



We need to drastically speedup checkpointing

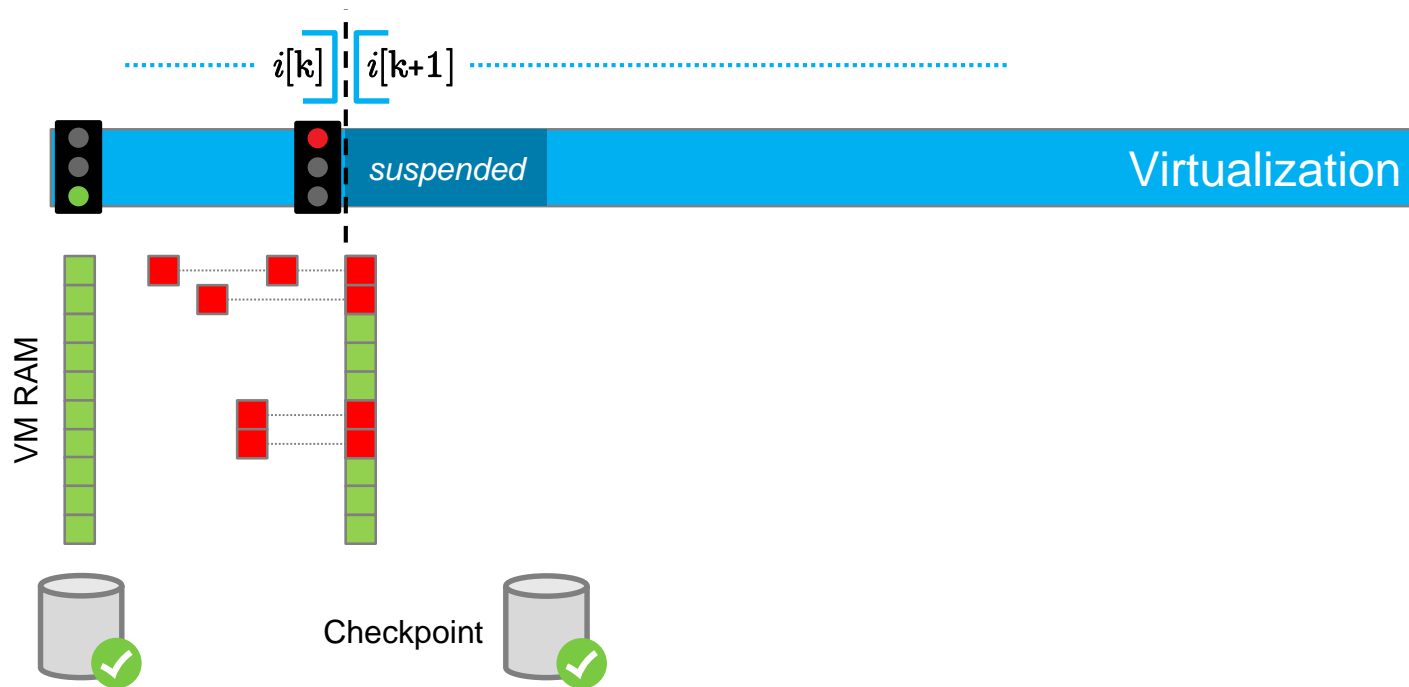
Incremental Stop-And-Copy



- Observation: Only some data modified per interval

pts_build_linux_kernel	spec_jbb
22000 pages/s (85 MiB/s)	53000 pages/s (200 MiB/s)

Incremental Stop-And-Copy



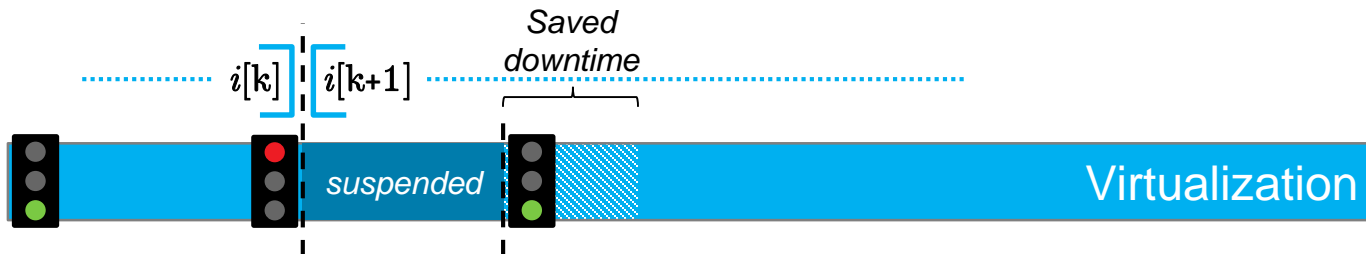
- Idea: Save only modified data
 - Track dirty pages via page protections
 - Use previous checkpoints to get unmodified data

Incremental Stop-And-Copy

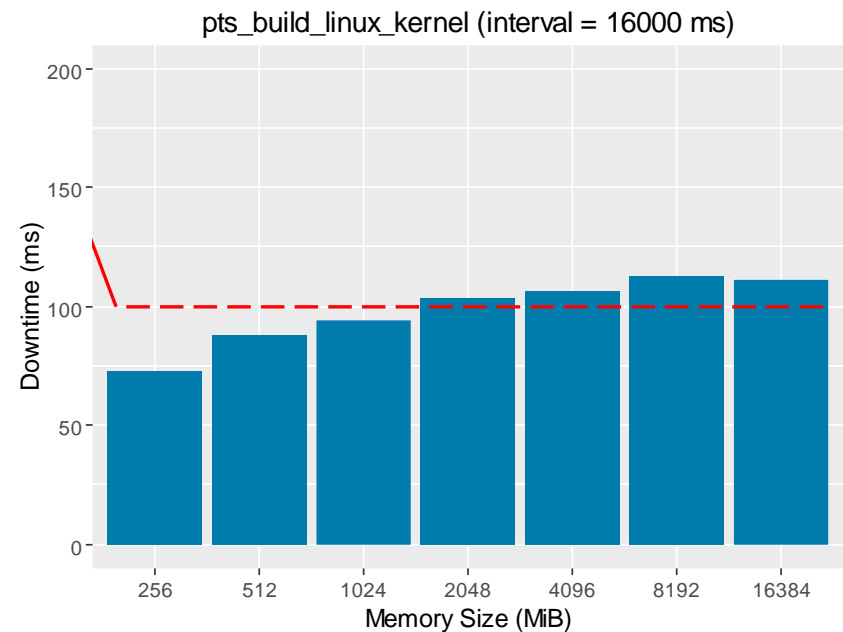


- Idea: Save only modified data
 - Track dirty pages via page protections
 - Use previous checkpoints to get unmodified data

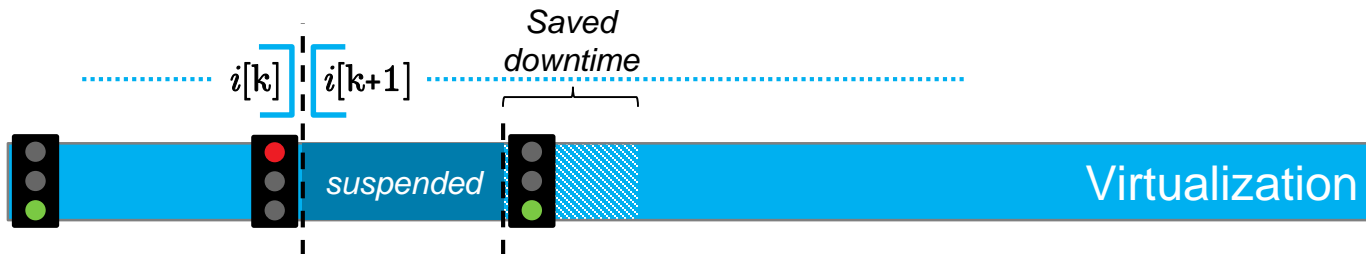
Incremental Stop-And-Copy



- Reduced downtime
- Less dependent on VM size

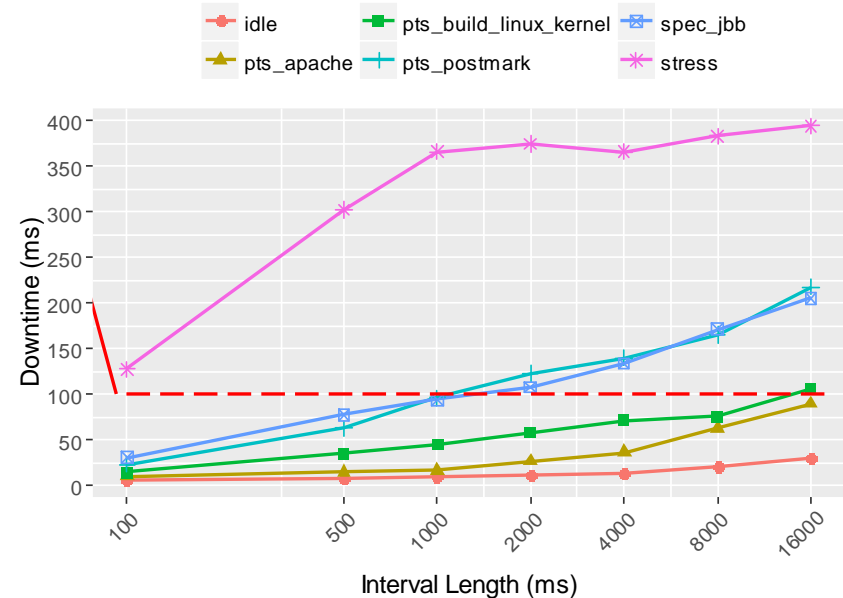


Incremental Stop-And-Copy

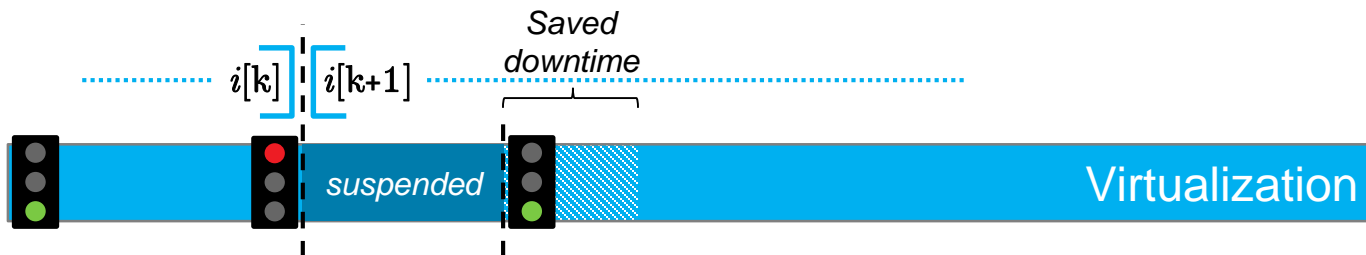


- Reduced downtime
 - Less dependent on VM size

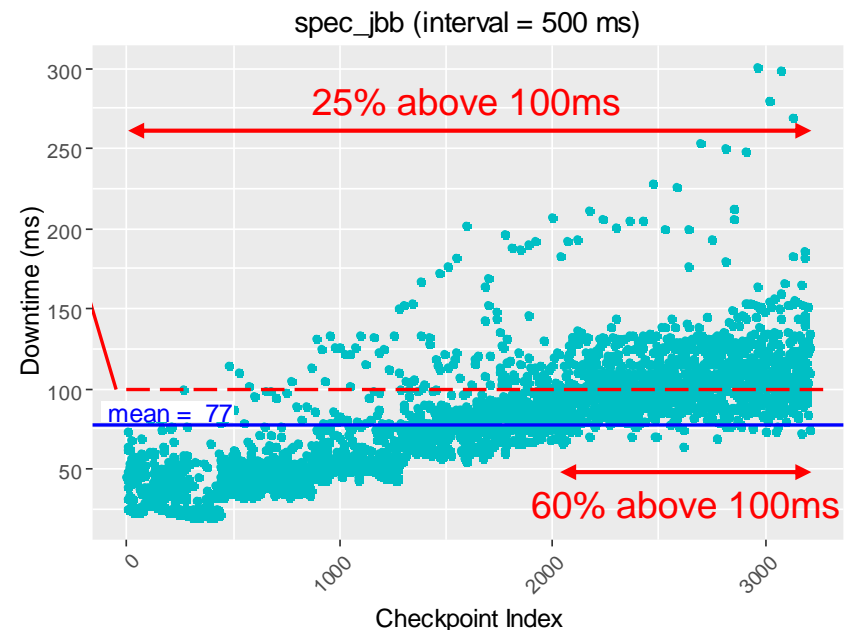
- But: Downtime depends on
 - Interval length
 - Workload



Incremental Stop-And-Copy

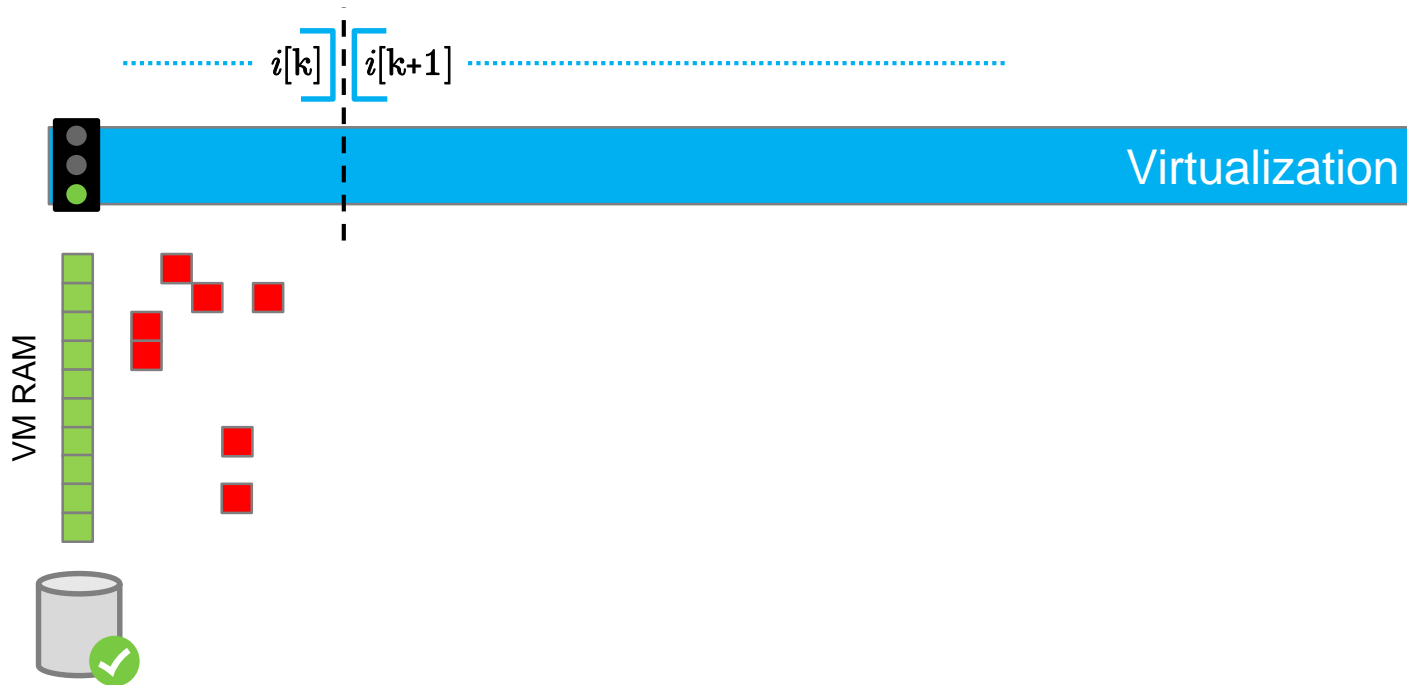


- Reduced downtime
 - Less dependent on VM size
- But: Downtime depends on
 - Interval length
 - Workload
- But: Downtime strongly fluctuates

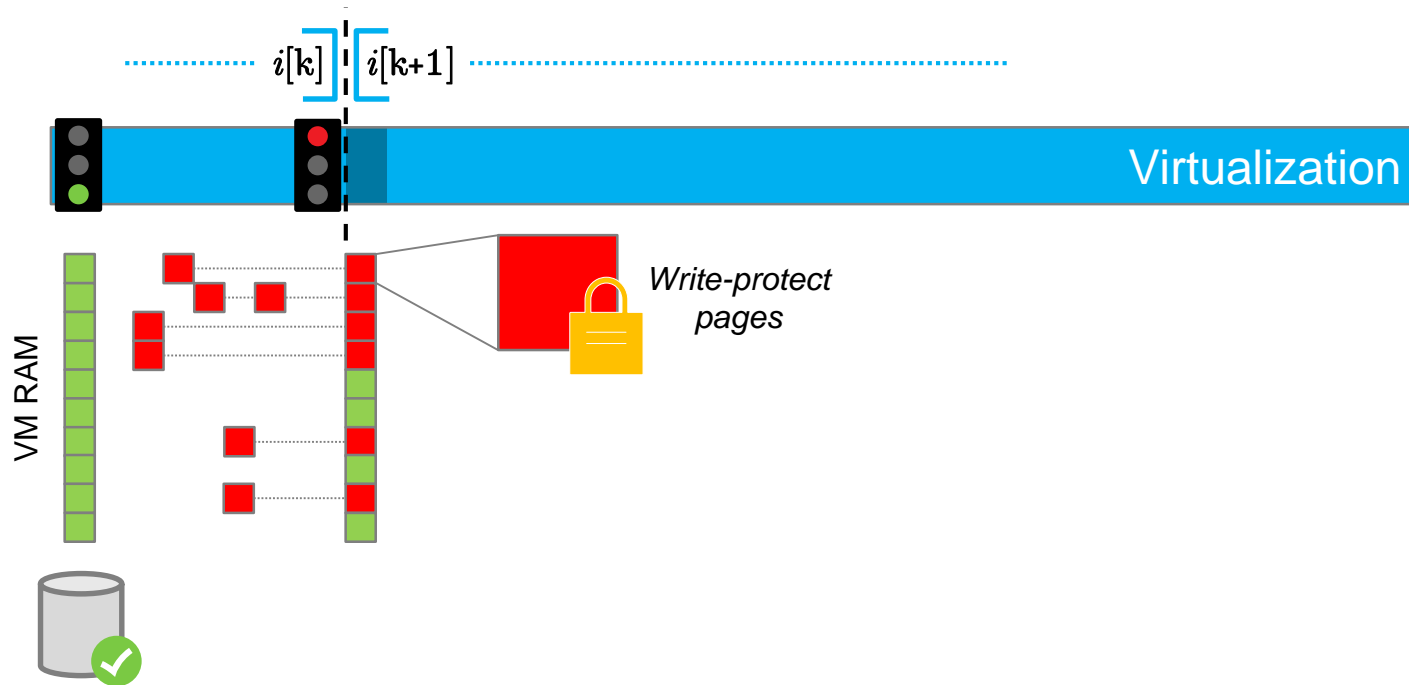


We need to further speedup checkpointing

Incremental Copy-On-Write

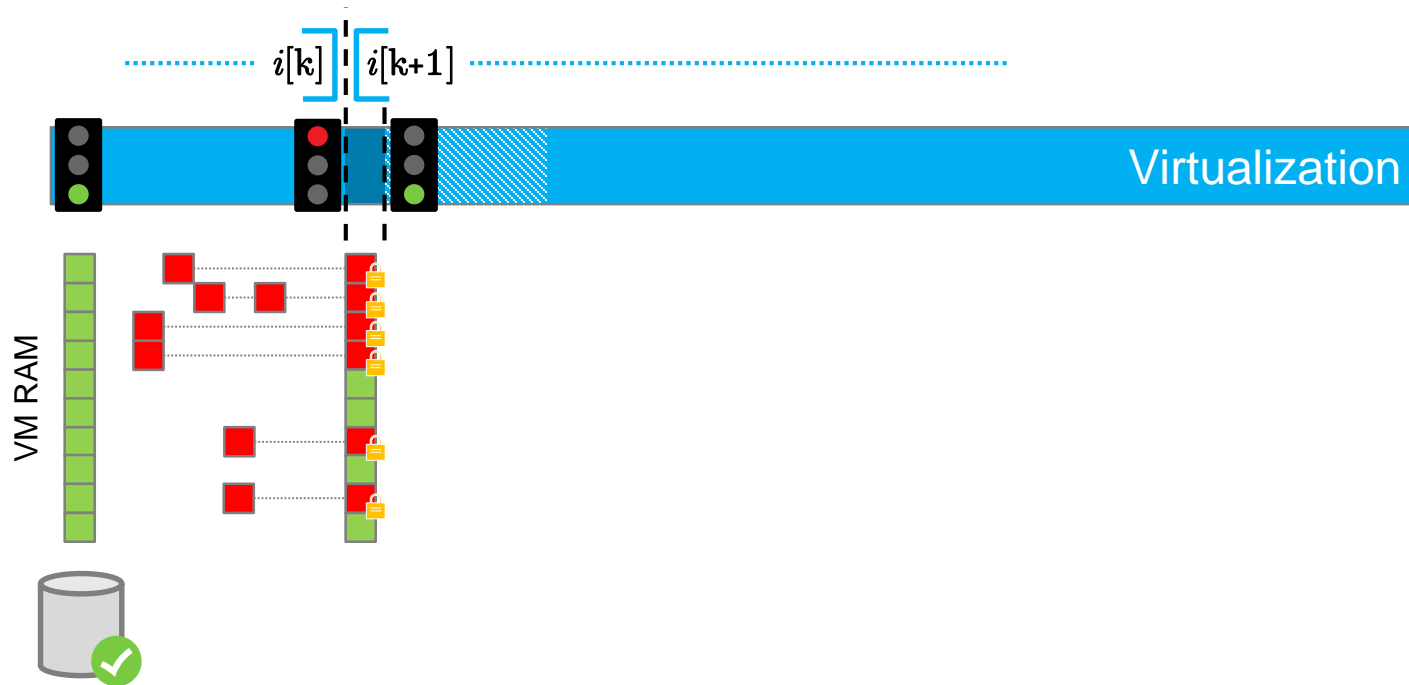


Incremental Copy-On-Write



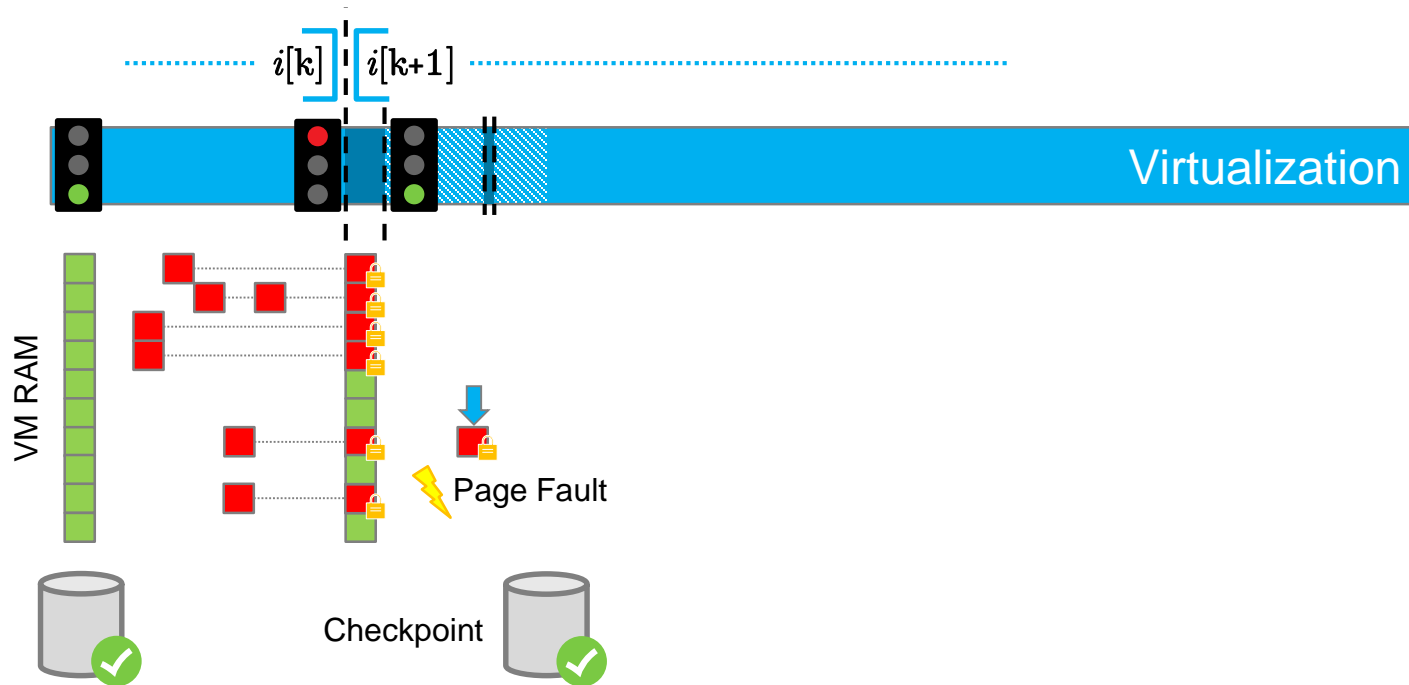
- Idea: Save modified pages asynchronously
 - Use write-protection to prevent modification

Incremental Copy-On-Write



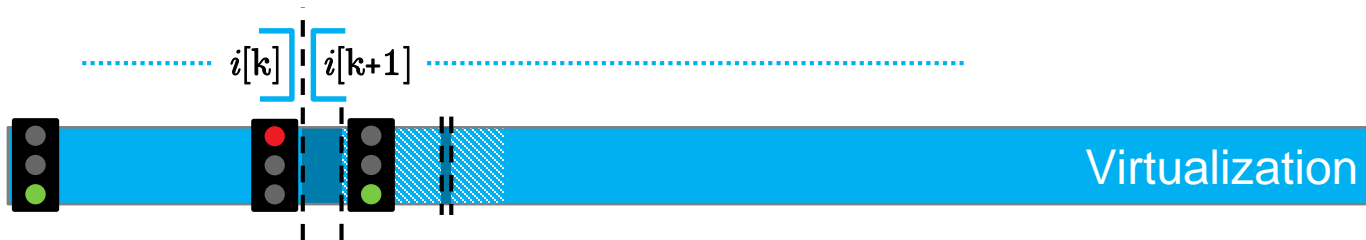
- Idea: Save modified pages asynchronously
 - Use write-protection to prevent modification

Incremental Copy-On-Write

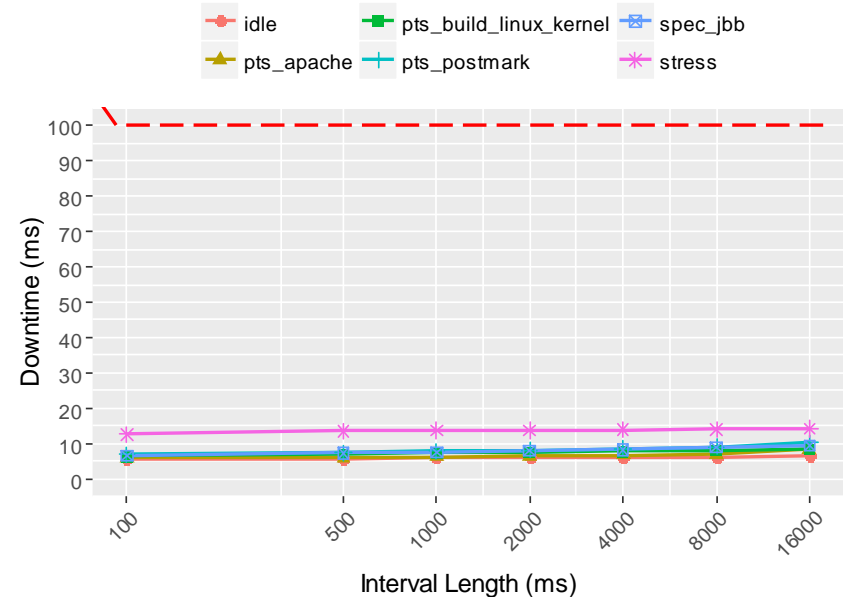


- Idea: Save modified pages asynchronously
 - Use write-protection to prevent modification
 - Copy and release protection on pagefault

Incremental Copy-On-Write



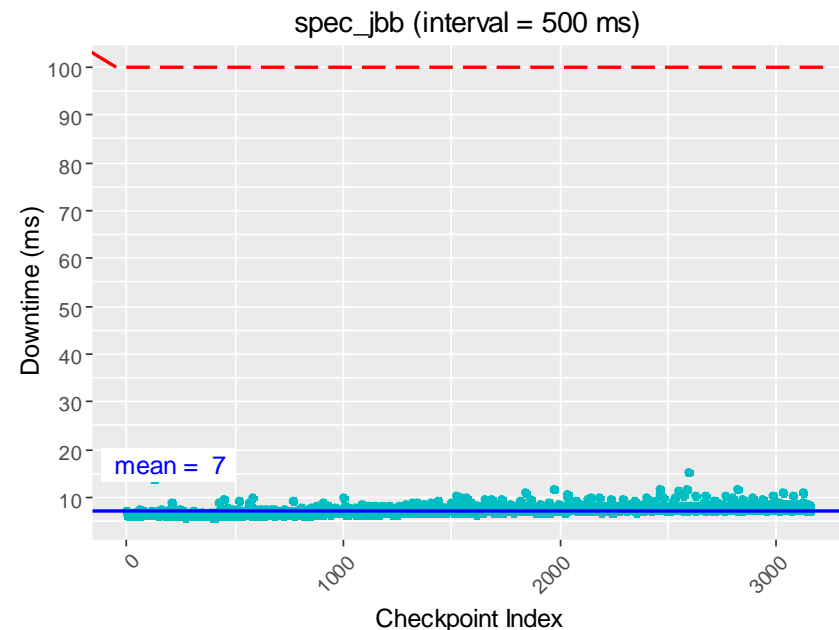
- Drastically reduced downtime
 - Pagefaults do not impede interactivity
- Less dependent on
 - Interval length
 - Workload



Incremental Copy-On-Write



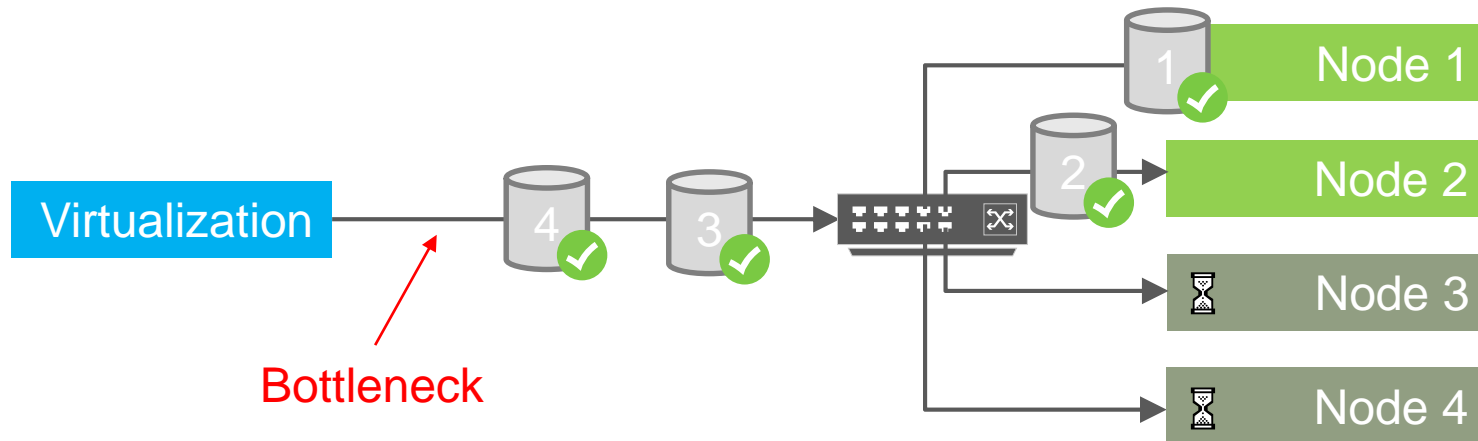
- Drastically reduced downtime
 - Pagefaults do not impede interactivity
- Less dependent on
 - Interval length
 - Workload
- Almost constant downtime



We can do checkpointing fast enough

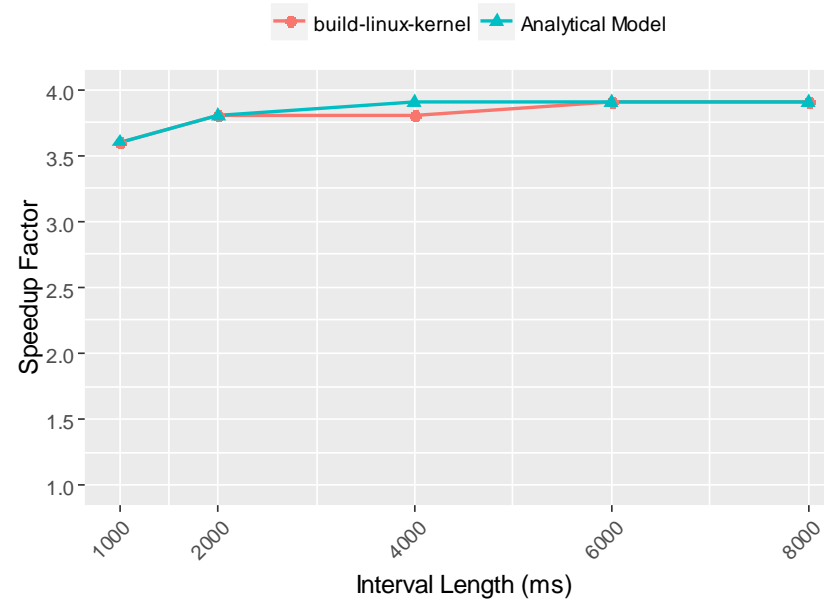
Checkpoint Distribution – The Naïve Way

- Nodes request full checkpoints from central server



- But: Central server becomes bottleneck
 - Limits parallelization and speedup

SimuBoost Evaluation



- Prototype: 1GiB RAM, 1s intervals, 4 simulation nodes
 - SimuBoost delivers predicted speedup [Rittinghaus13]
 - But: Saturates 10 Gbit Ethernet

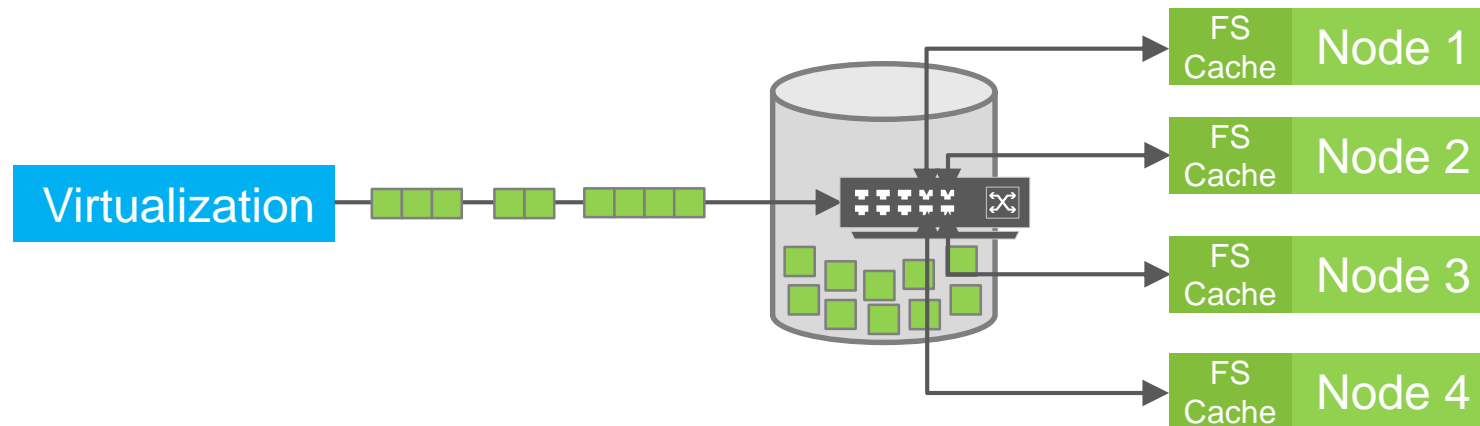
Need to avoid single bottleneck

Future Checkpoint Distribution

- Idea: Only send new data
 - Deduplicate and compress data

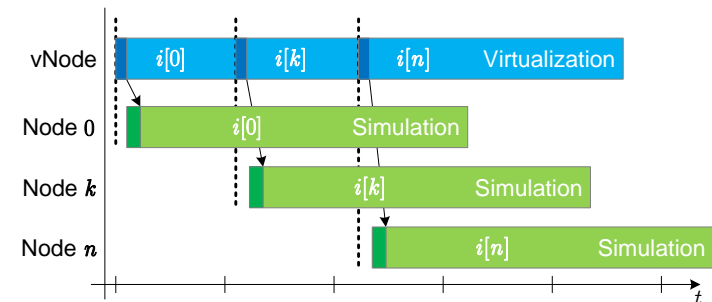
pts_build_linux_kernel	spec_jbb
22000 pages/s (85 MiB/s)	53000 pages/s (200 MiB/s)
5000 pages/s (20 MiB/s)	16000 pages/s (65 MiB/s)

- Use distributed file system (e.g., Ceph [Weil06])
- Append new data to global file
- Checkpoint = Map of VM addresses to offsets in file





Conclusion

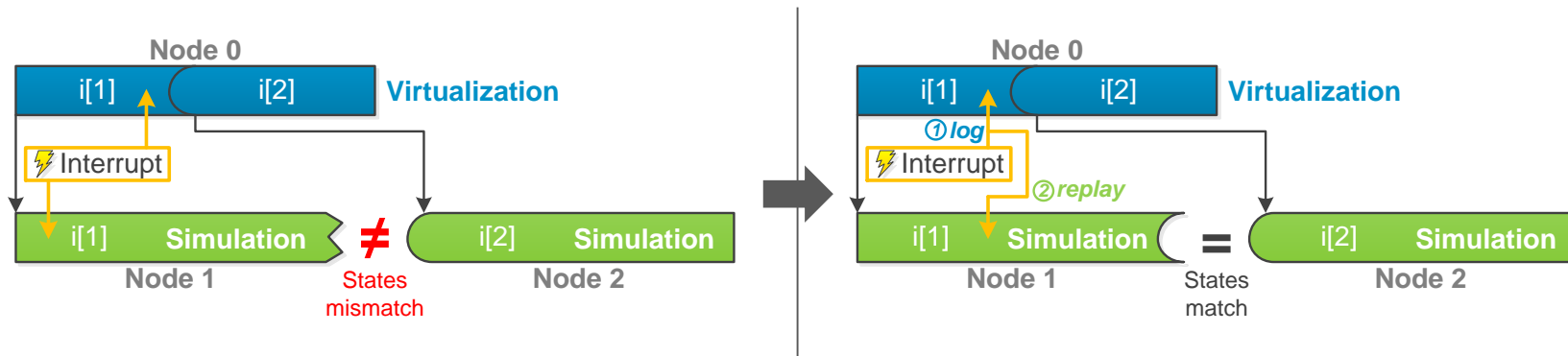
- Slowdown of Functional Full System Simulation: >100x
- SimuBoost: Accelerate simulation
 - Run workload with fast virtualization
 - Take checkpoints in regular intervals
 - Start parallel simulations on checkpoints



Challenges

- Fast checkpoint creation 
 - Incremental Copy-On-Write
- Fast checkpoint distribution 
 - Distributed file system

Deterministic Replay



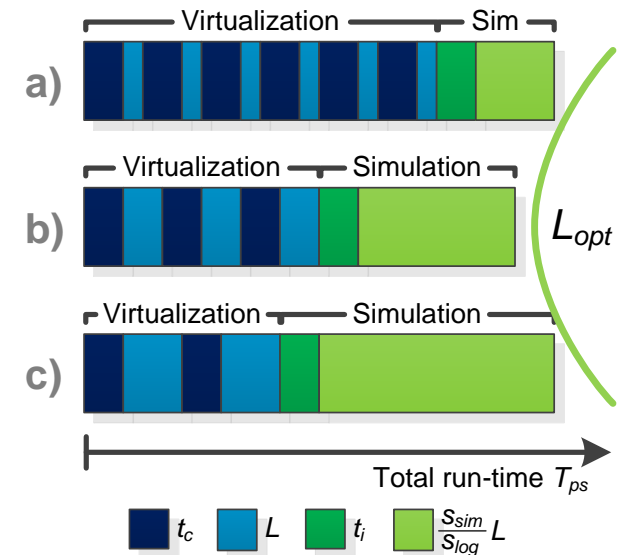
- (1) Trap and log non-deterministic events in the hypervisor
- (2) Precisely replay events in the simulation

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

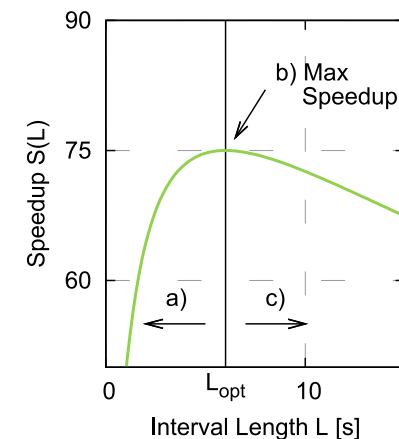
- Existing work: Retrace [Sheldon07], V2E [Yan12]

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:
 - 100ms downtime, 8% logging, 100x slowdown
 - Optimal interval length: 2s
 - Best possible speedup for 1h workload: 84x @ 90 nodes (94% parallel efficiency)



Near linear speedup possible

Selected Previous Research

- Workload Reduction
 - MinneSPEC [KleinOsowski02]
- Simulate samples and extrapolate
 - Truncated Execution
 - SimPoints [Sherwood02]
 - SMARTS [Wunderlich03]
- Improve simulation engine
 - Optimize engine: below 5x speedup mark
 - Parallelize simulation of vCPUs [Ding11]
- Divide simulation time
 - For microarchitectural simulations: DiST [Girbal03]

References

- [Miller13] K. Miller et al. XLH: More effective memory deduplication scanners through cross-layer hints. USENIX, 2013
- [Wilhelm15] F. Wilhelm. Tracing Privileged Memory Accesses to Discover Software Vulnerabilities. Master Thesis, KIT, 2015
- [Jurczyk13] M. Jurczyk et al. Bochspwn: Exploiting Kernel Race Conditions Found via Memory Access Patterns. 2013
- [Rittinghaus13] M. Rittinghaus. SimuBoost: Scalable Parallelization of Functional System Simulation. WODA, 2013
- [Weil06] S. A. Weil et al. Ceph: A Scalable, High-Performance Distributed File System. OSDI, 2006
- [Bellard05] F. Bellard. Qemu: A Fast and Portable Dynamic Translator. USENIX, 2005
- [Magnusson02] P. Magnusson et al. Simics: A Full System Simulation Platform. Computer, 35(2), 2002
- [Sherwood02] T. Sherwood et al. Automatically Characterizing Large Scale Program Behavior. ACM SIGARCH, 30(5), 2002
- [Ding11] J. Ding et al. PQEMU: A Parallel System Emulator Based on QEMU. ICPADS, 2011
- [Wunderlich03] R. E. Wunderlich et al. SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. Computer Architecture, 2003
- [Girbal03] S. Girbal et al. DiST: A Simple, Reliable and Scalable Method to Significantly Reduce Processor Architecture Simulation Time. SIGMETRICS, 31(1), 2003
- [KleinOsowski02] A. J. KleinOsowski et al. MinneSPEC: A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Research. IEEE Computer Architecture Letters 1.1, 2002
- [Sheldon07] M. Sheldon et al. Retrace: Collecting Execution Trace With Virtual Machine Deterministic Replay. MoBS, 2007
- [Yan12] L. Yan et al. V2E: Combining Hardware Virtualization and Software Emulation for Transparent and Extensible Malware Analysis. VEE, 2012
- [RbMiller68] Robert B. Miller. Response Time in Man-Computer Conversational Transactions. 1968.