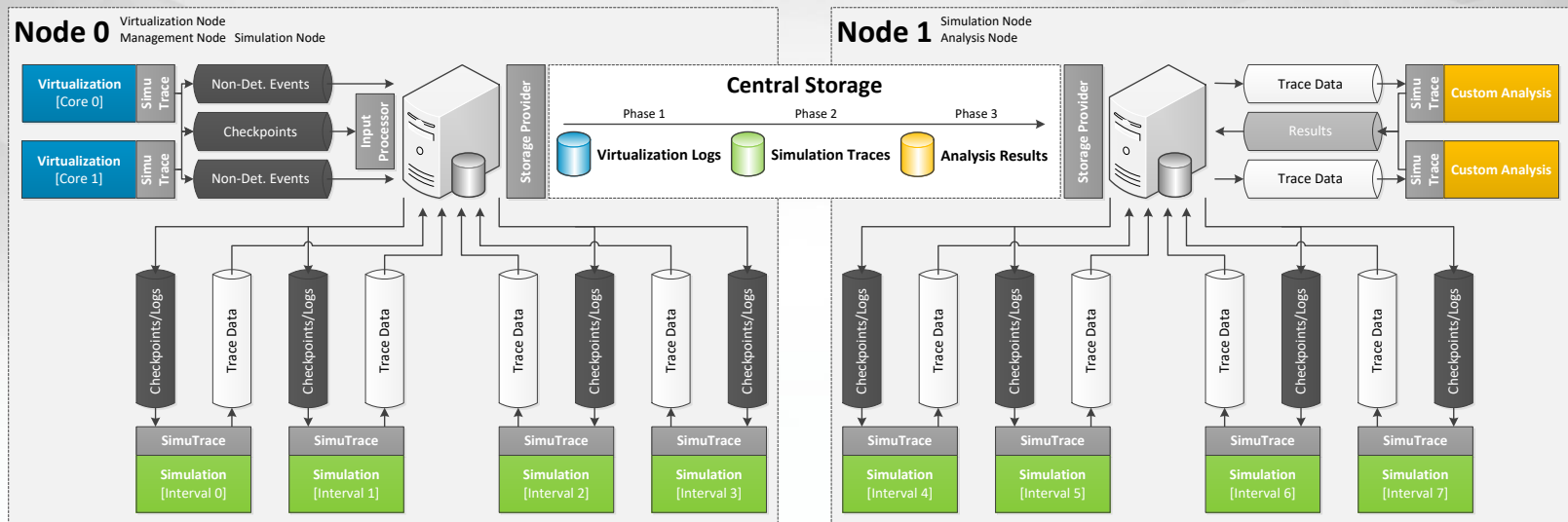


# Fast and Accurate Functional Simulation for Dynamic Full System Analysis

GI Fachgruppentreffen Betriebssysteme (BS) 2018  
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**

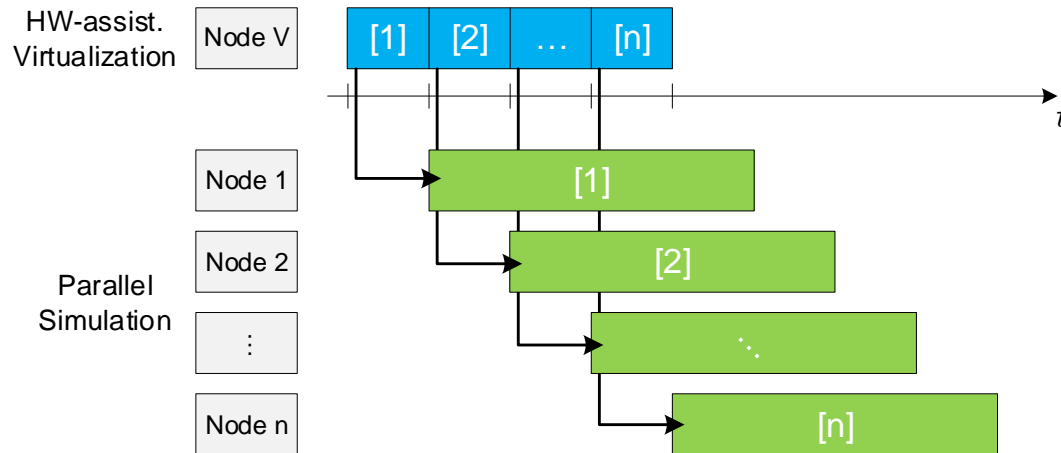
# Motivation

- Functional full system simulation
  - Allows collecting required data
  - Simulation on instruction-level
  - Includes operating system and drivers
  
- But: It is slow

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

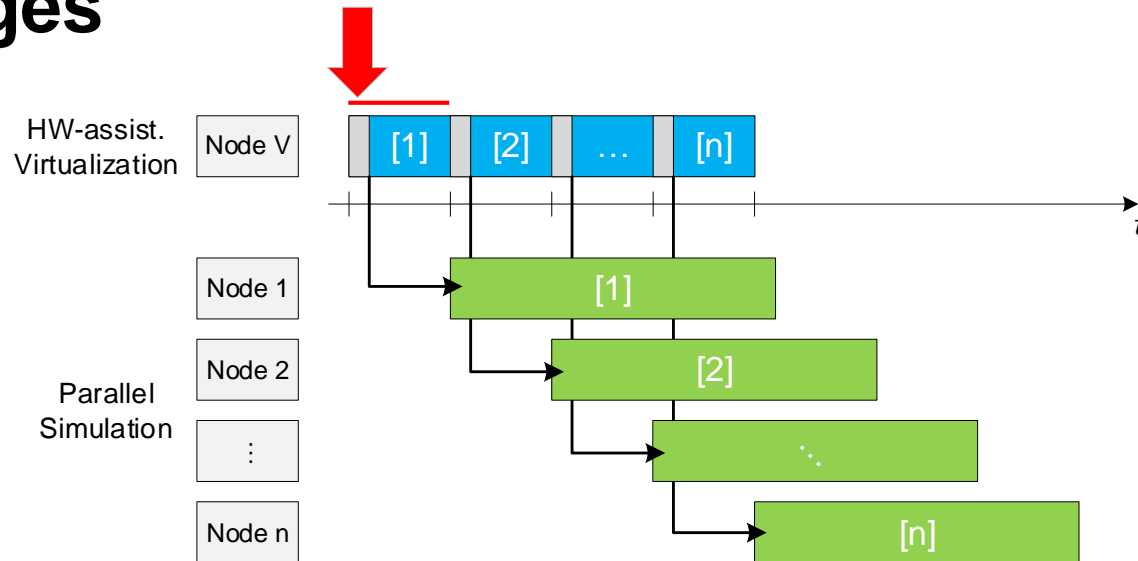
Average slowdowns for: kernel build, SPECint\_base06, LAMMPS

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



- Use virtualization and split execution into intervals
  - Checkpoints at interval boundaries bootstrap simulations
  - Speed difference drives parallelization
  - Hardware acceleration provides full interactivity
- Does not trade accuracy for speed
- Scales with run time of workload

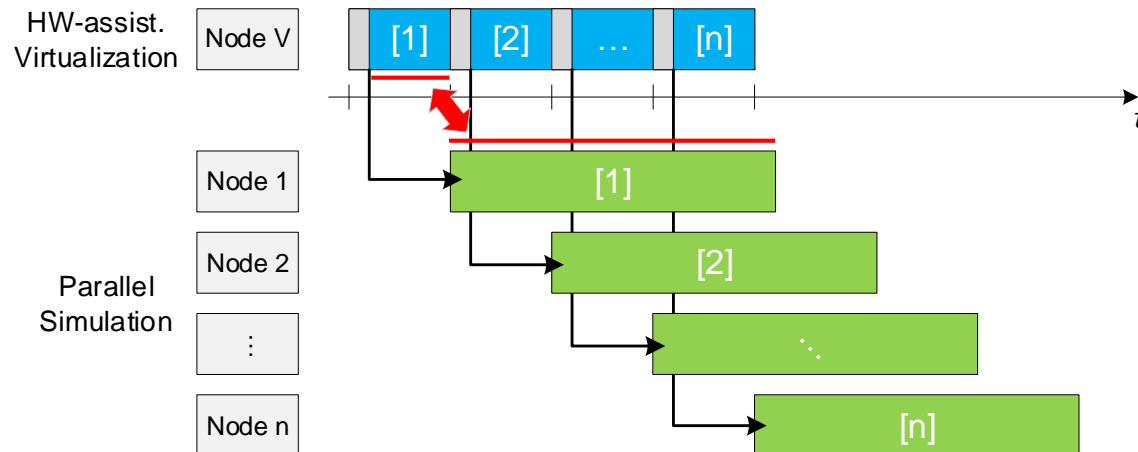
# Challenges



- **Downtime** to capture consistent state
  - Keep below 100ms to preserve interactivity [RbMiller68]
- **Run time overhead** to asynchronously save data
  - Keep low to minimize probe effect

Lightweight  
Checkpointing

# Challenges

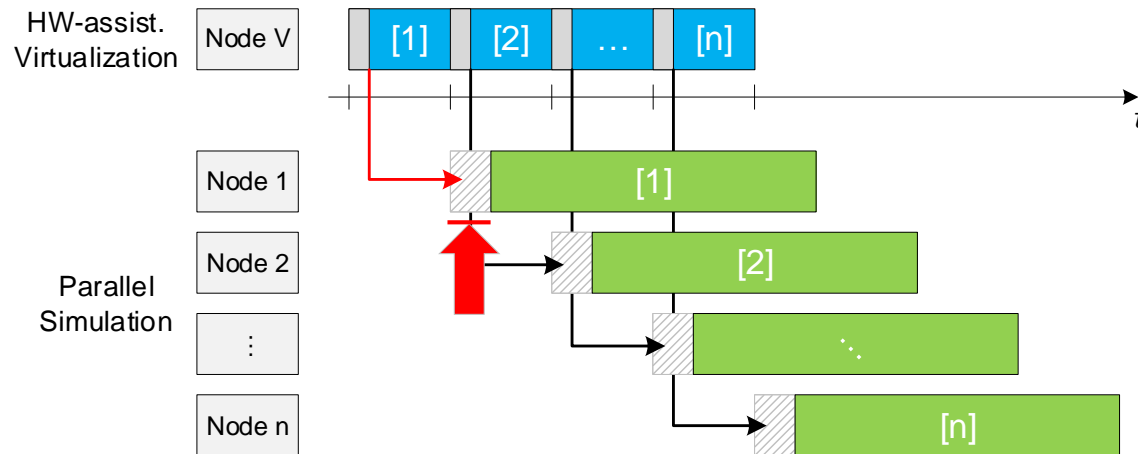


- **Replay** of events
  - Must have instruction-level precision
- **Run time overhead** to capture non-deterministic events
  - Keep low to minimize probe effect

Lightweight  
Checkpointing

Deterministic  
Replay

# Challenges



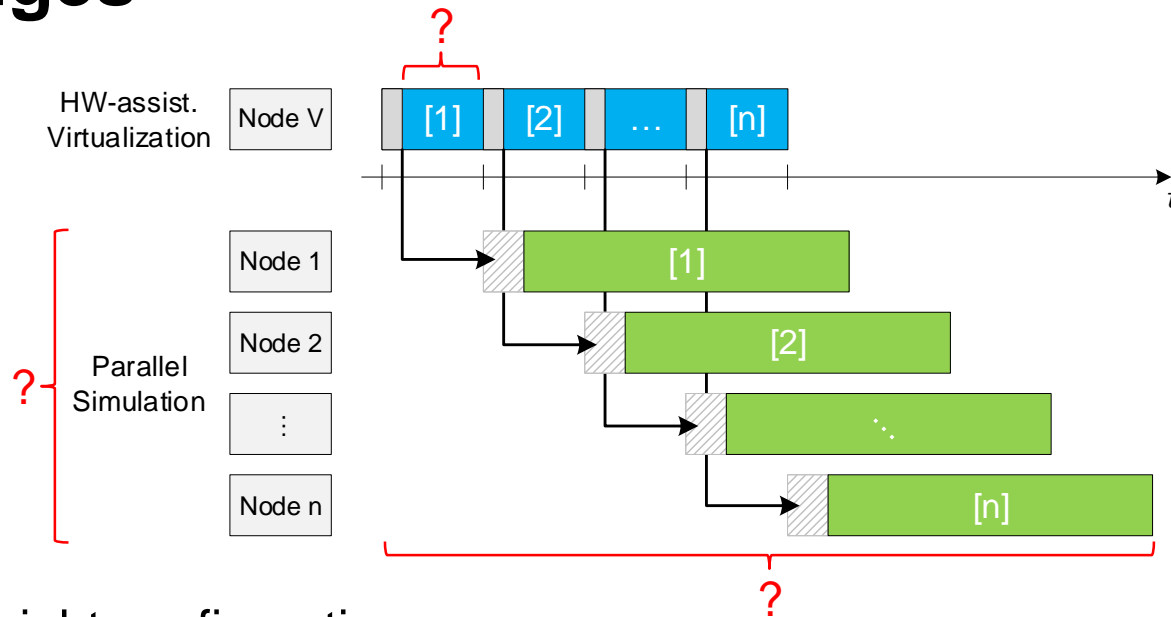
- Startup time: **Transfer time + Checkpoint loading time**
  - Keep low to maximize speedup
- Network **bandwidth** to migrate checkpoints
  - Keep low to allow commodity network infrastructure (Gigabit Ethernet)

Lightweight  
Checkpointing

Deterministic  
Replay

Fast Simulation  
Startup

# Challenges



- Choose right configuration
  - Interval length & number of simulation nodes
- Estimate performance
  - Parallel simulation time & speedup

Lightweight  
Checkpointing

Deterministic  
Replay

Fast Simulation  
Startup

Performance  
Model



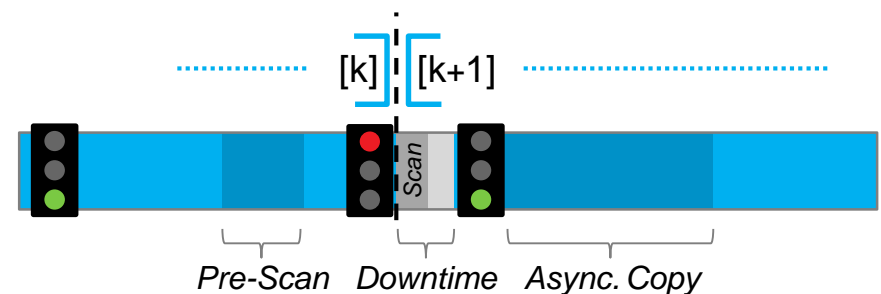
# Checkpointing

- Problem: Cannot save full guest memory on each checkpoint
- Observation: Only some data modified per interval

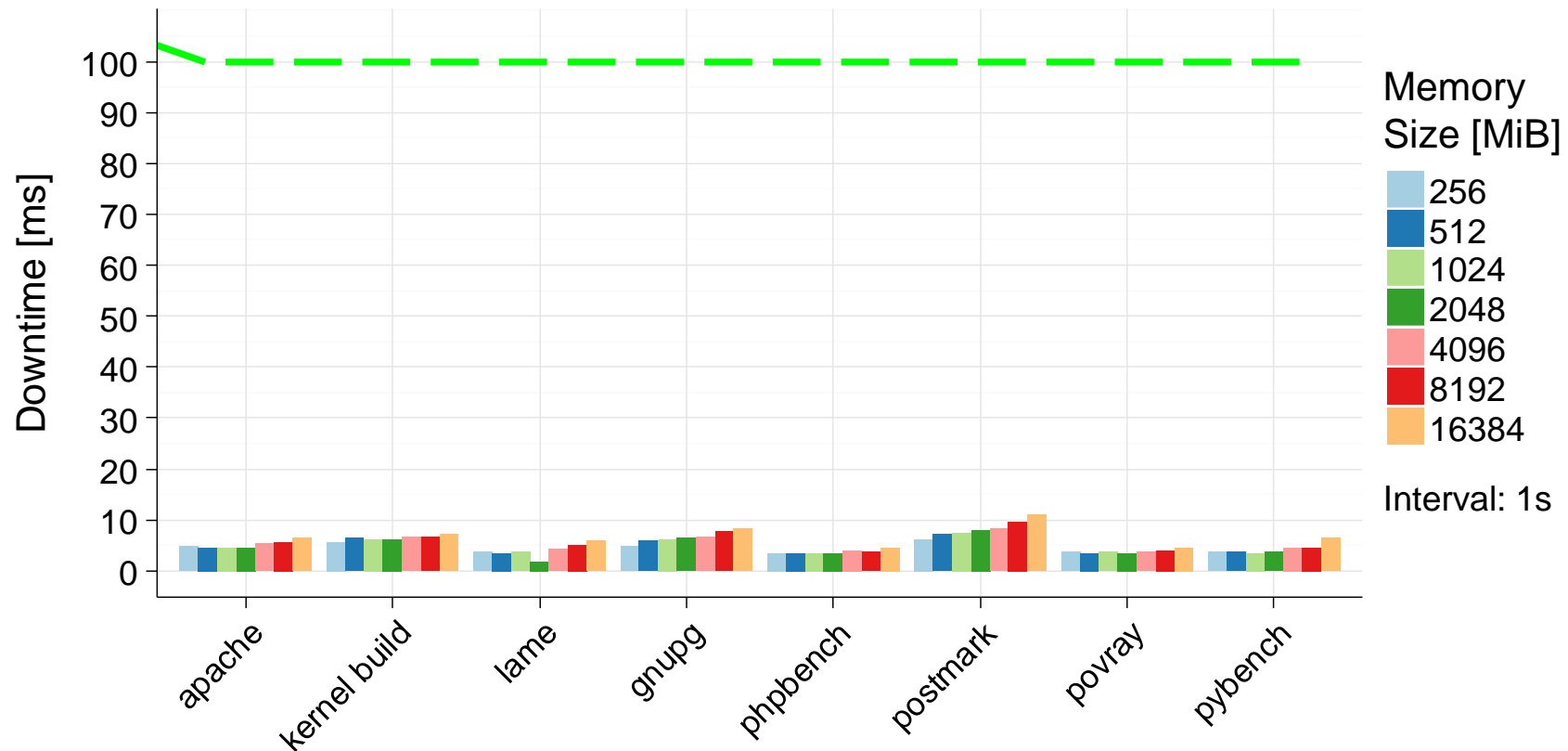
Linux Kernel Build	SPECjbb2005
22000 pages/s (85 MiB/s)	53000 pages/s (200 MiB/s)

- Dirty Page Tracking
  - Standard: Use page protections
  - Better: Scan extended page table (EPT)
    - Less page faults:  $\frac{1}{2}$  run time overhead, but 2x downtime
    - Pre-Scan keeps downtime low

- Copy-On-Write (CoW)
  - Protect modified pages (only)
  - Asynchronously copy pages



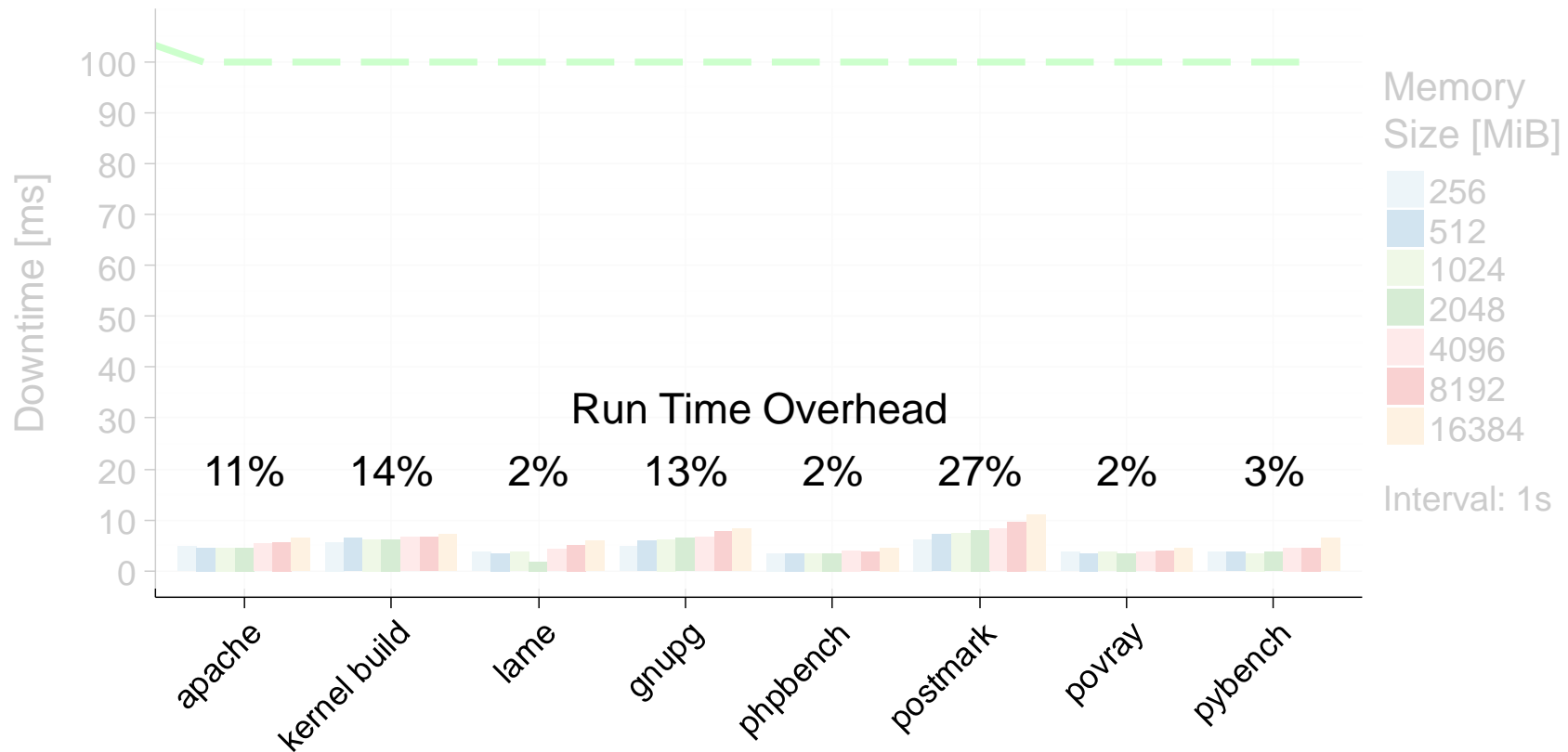
# Checkpointing II



## ■ Baseline: Simple Stop-And-Copy (4GiB VM)

- Downtime: 1.2s
- Run time overhead: ~3x

# Checkpointing II



## ■ Baseline: Simple Stop-And-Copy (4GiB VM)

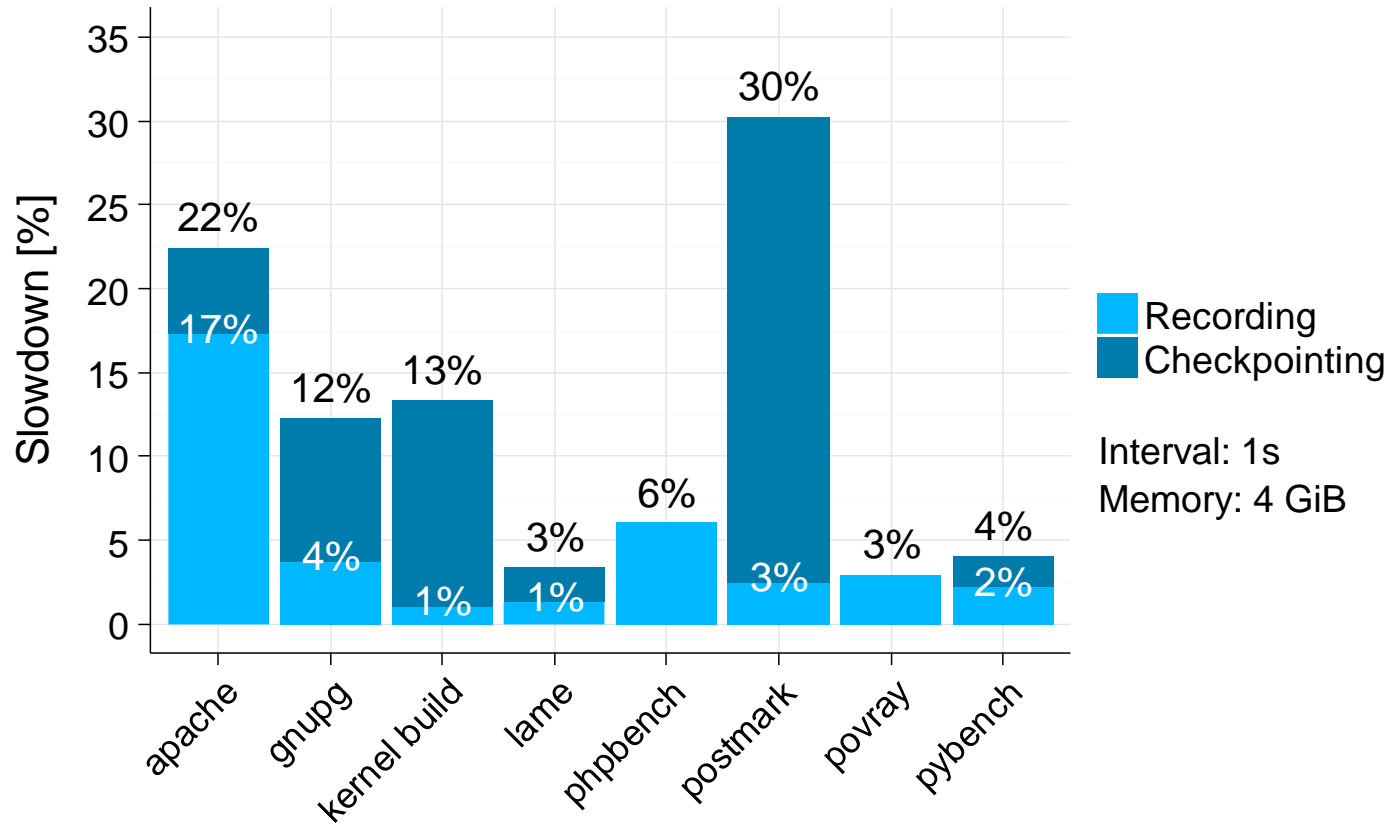
- Downtime: 1.2s
- Run time overhead: ~3x

# Deterministic Replay

- Synchronous events (`cpuid`, `rdtsc`, port I/O, ...)
  - Trap & record `<value>`
  - Replay value in simulation
- Asynchronous events (interrupts, DMA)
  - Trap & record `<landmark, value>`
  - Landmark: Instruction counter + CPU registers
  - Inject event at landmark (e.g., raise interrupt)
- Improve accuracy of simulation
  - Match instruction counting with hardware
  - Implement “undefined” status bits
  - ...

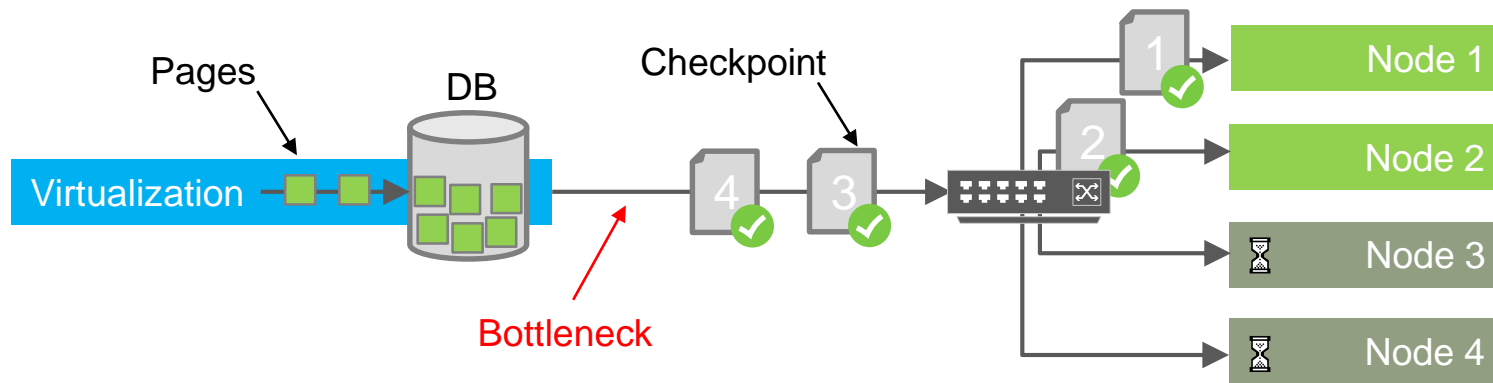
	Inst. Count	Instruction Stream
	1	<code>mov reg1, reg2</code>
	2	<code>add reg1, 0x42</code>
	3	<code>push reg1</code>
	4	<code>call func</code>
	...	...
<i>replay</i>	10	<code>in reg1</code>
	...	...
	20	<code>mov reg2, reg1</code>
<i>Int</i> — [	21	<code>xor reg3, reg3</code>
	22	<code>add reg4, reg1</code>
	23	<code>push reg3</code>
	24	<code>push reg2</code>
	...	...

# Run Time Overhead



# Naive Simulation Startup

- 1) Nodes request checkpoints from central server

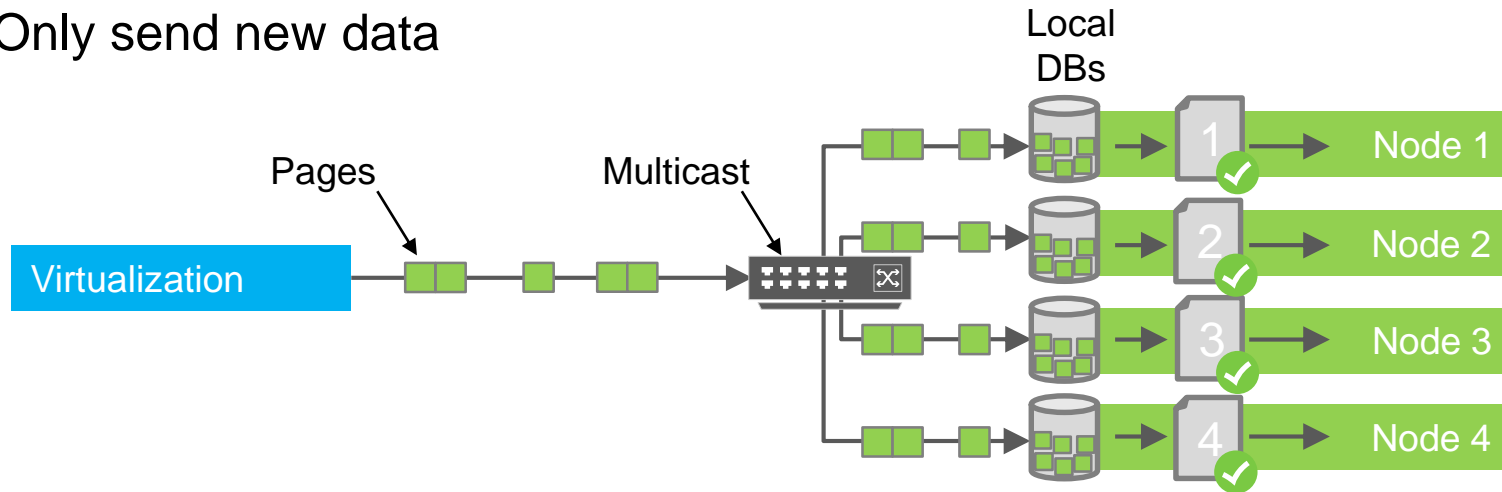


- 2) Load entire checkpoint into memory
- 3) Start simulation of interval

■ But: Central server becomes bottleneck

# Simulation Startup

- Only send new data



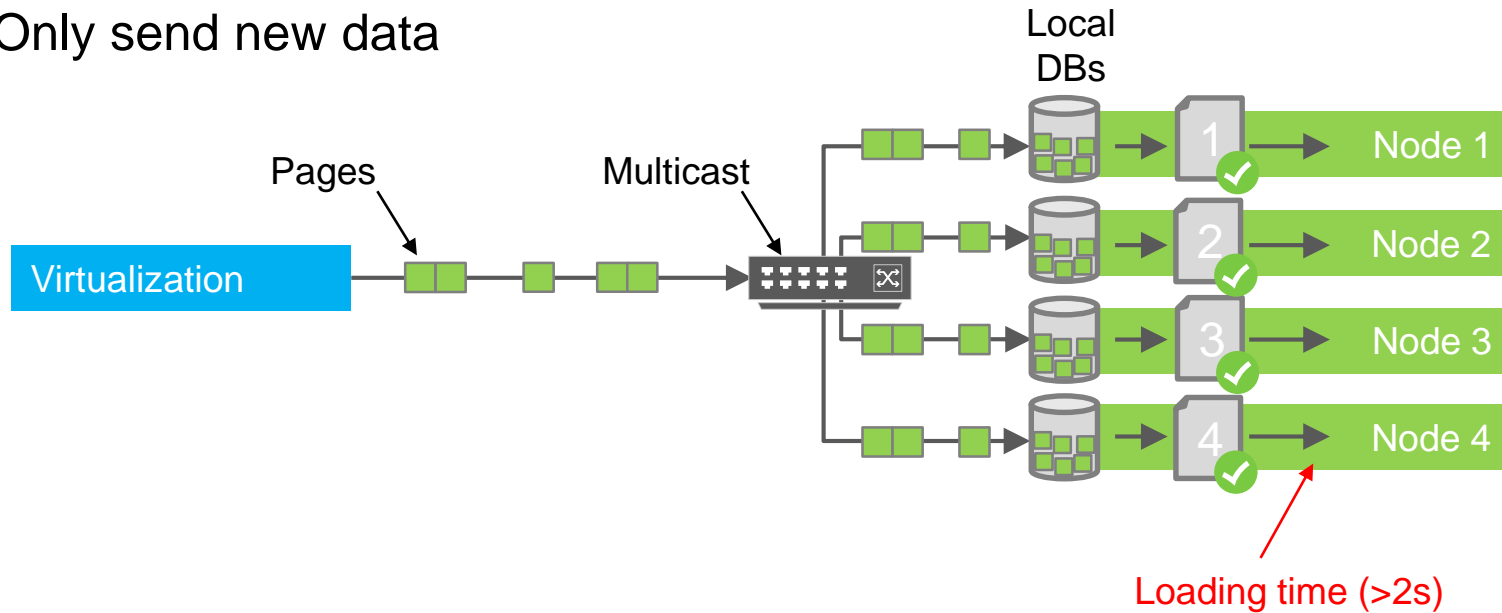
- Deduplicate and compress data

Linux Kernel Build	SPECjbb2005
22000 pages/s (85 MiB/s)	53000 pages/s (200 MiB/s)
5000 pages/s (20 MiB/s)	16000 pages/s (65 MiB/s)

↓  
Gigabit Ethernet

# Simulation Startup

- Only send new data



- Only load pages that will be accessed in interval

Load time: 2s ↘ 200ms

Memory consumption: 4GiB ↘ <500MiB



Increase simulation density

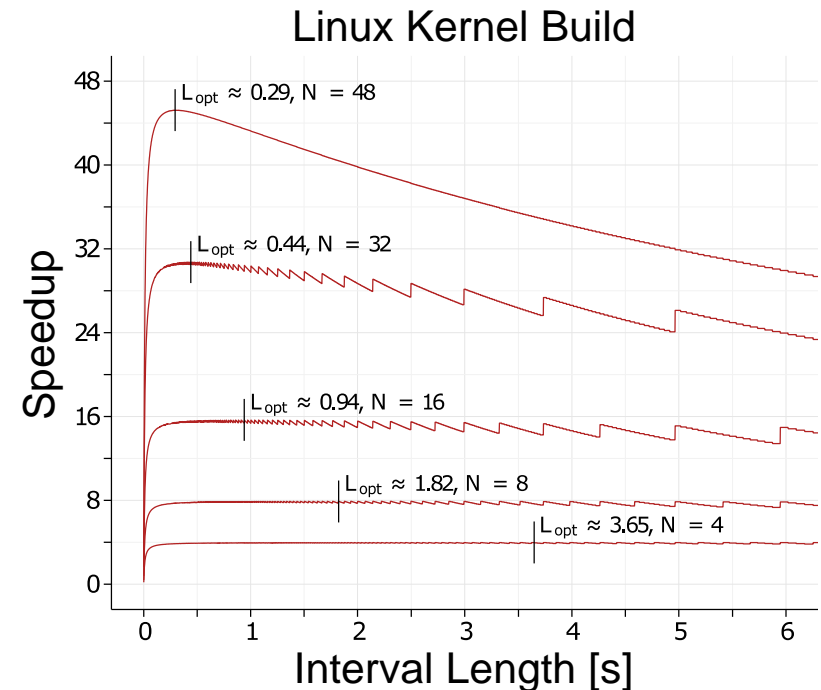


# Performance Model

- Want: Optimal interval length
- Input
  - Workload run time without SimuBoost
  - Virtualization overhead (slowdown, downtime)
  - Simulation overhead (slowdown incl. analysis, startup time)

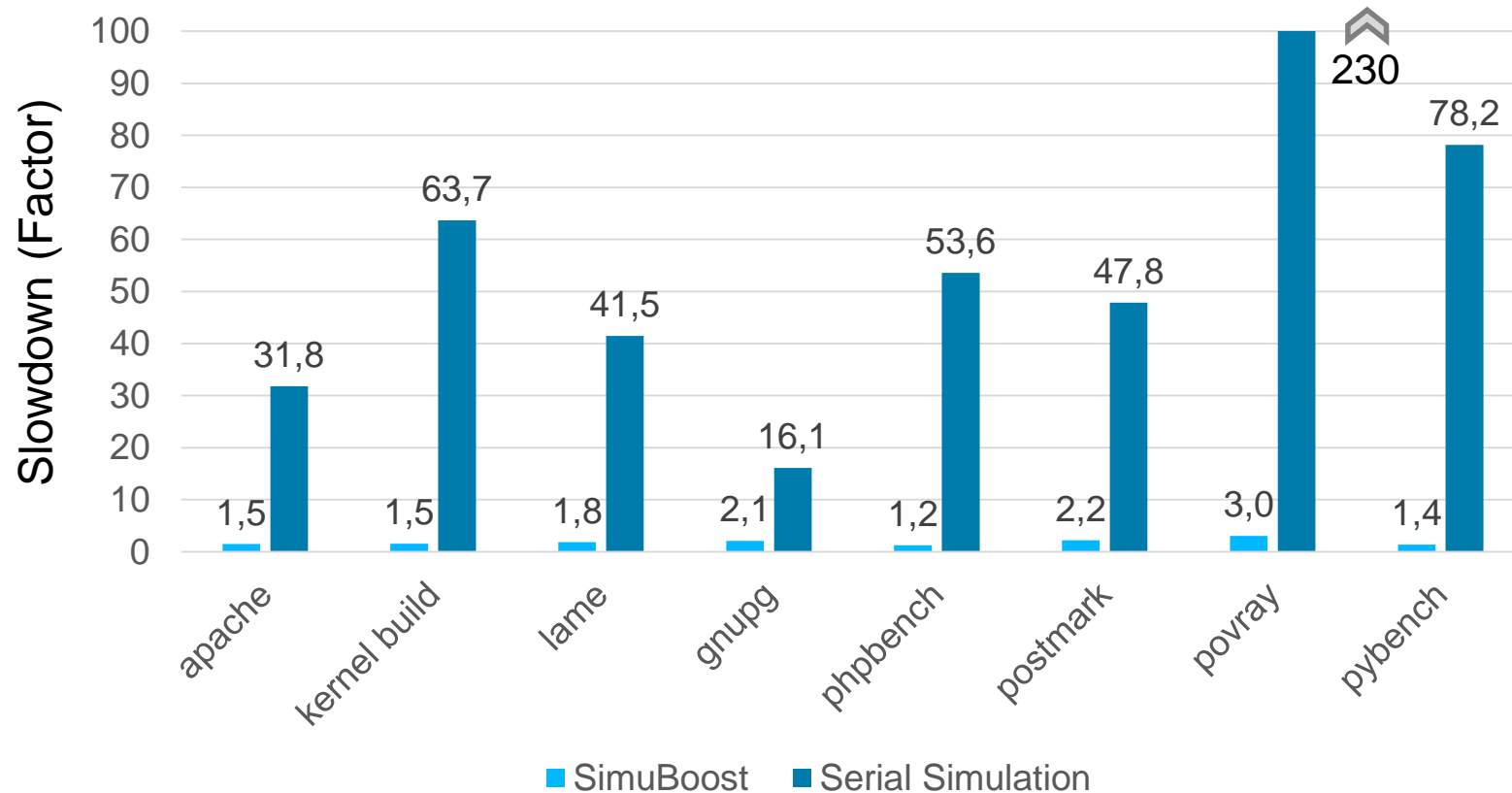
- Output
  - Parallel simulation time
  - Speedup

- Differentiation
  - Optimal interval length



# Evaluation

- Baseline: Hardware-assisted virtualization without SimuBoost

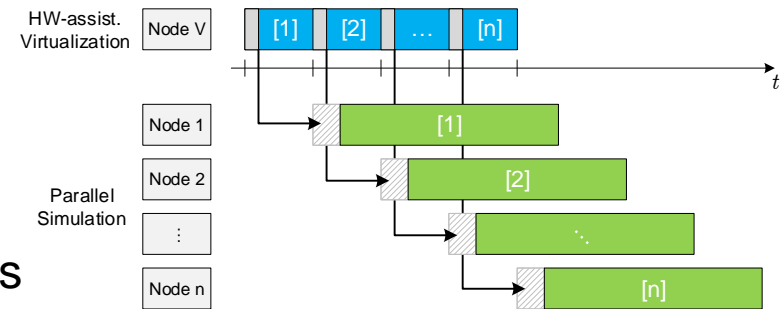


# Conclusion

- 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



- Contribution

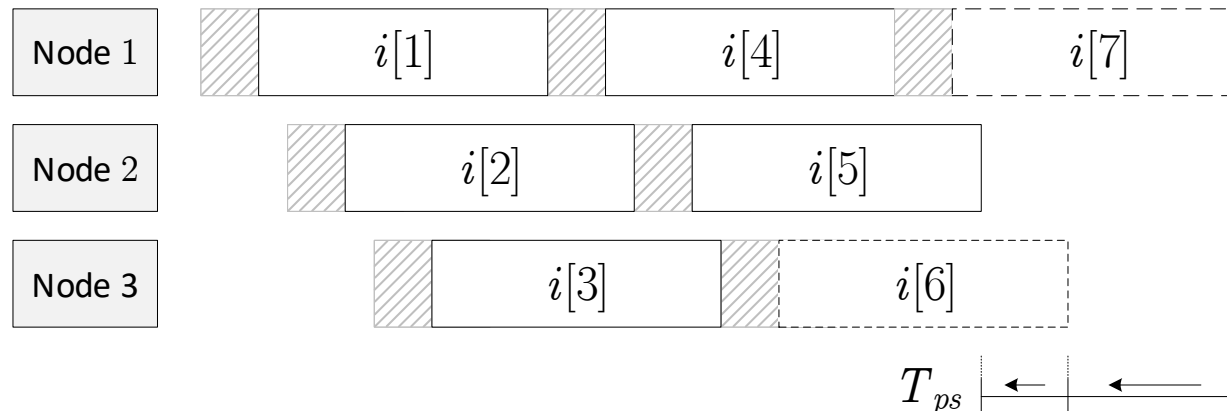
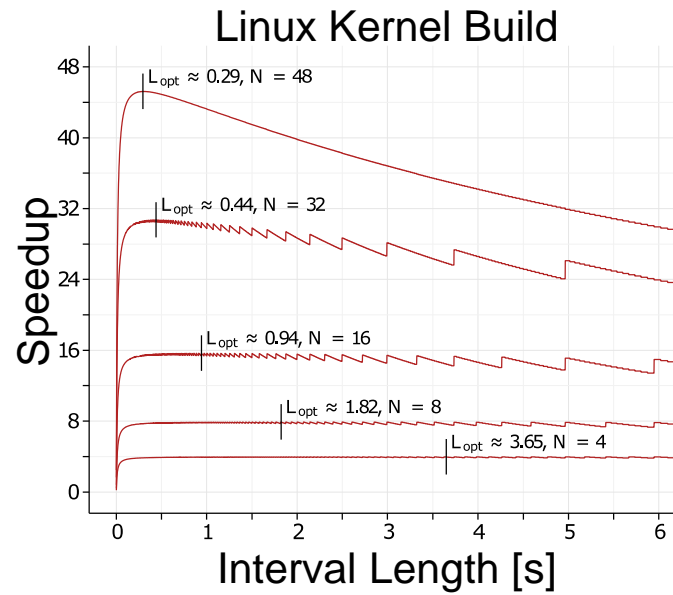
- Drastically reduces slowdown (e.g.,  $230x \rightarrow 3x$ )
- Enables interactivity (users and network)
- Moderate probe effect
  - Run time overhead on average 40%

**SimuBoost: Functional full system simulation made practical**

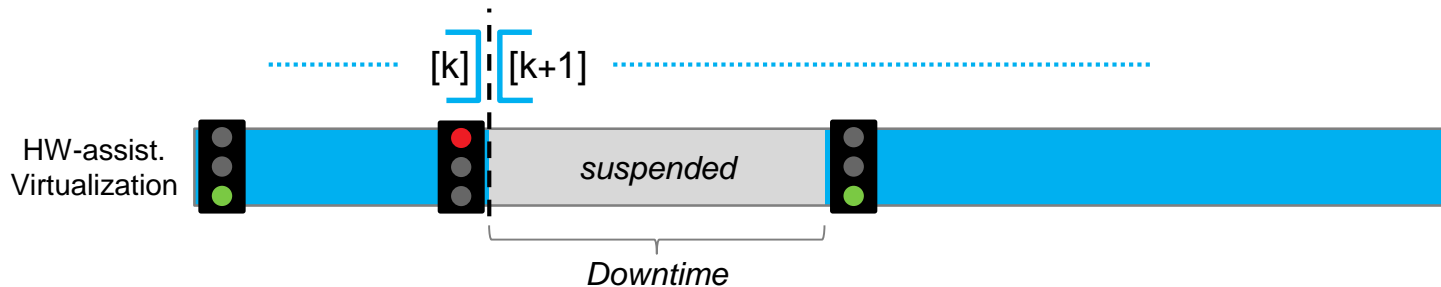
# Evaluation Raw Data

	apache	kernel build	lame	gnupg	phpbench	postmark	povray	pybench
<b>HW Time</b>	166,1	439	37,5	21,66	271	38,7	611	33,78
<b>HW SimuBoost</b>	225	554	66	27	306	82	669	43
<b>Overhead</b>	1,35	1,26	1,76	1,25	1,13	2,12	1,09	1,27
<b>Parallel Time</b>	247	665	68	45	337	86	1857	47
<b>Serial Time</b>	5280	27957	1556	349	14528	1851	141202	2640
<b>SimuBoost</b>	1,5	1,5	1,8	2,1	1,2	2,2	3,0	1,4
<b>Serial Simulation</b>	31,8	63,7	41,5	16,1	53,6	47,8	231,1	78,2
<b>Speedup</b>	21,4	42,0	22,9	7,8	43,1	21,5	76,0	56,2
<b>Nodes</b>	28	64	42	16	53	40	88	78
<b>Interval Length [ms]</b>	200	230	65	100	145	100	120	45

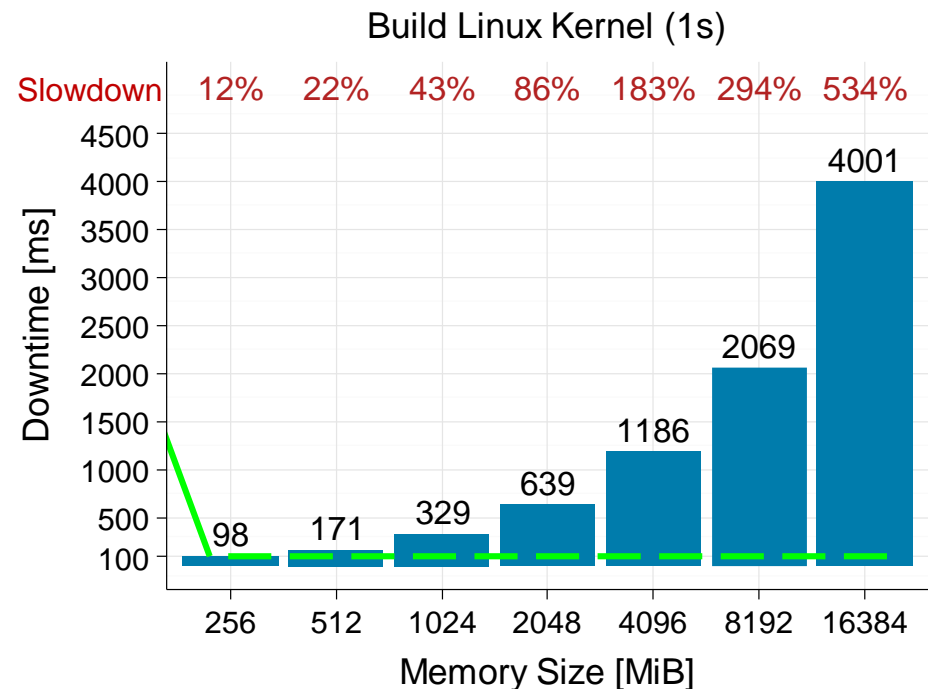
# Source of Sawtooth



# Stop-And-Copy Checkpointing



- Device states: 1 – 3 ms
- Guest physical memory
  - Dominates downtime
  - Downtime depends on VM size
- Not suited for interactive use
- Impairs parallelization



# 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]
  - SuperPin

# 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.