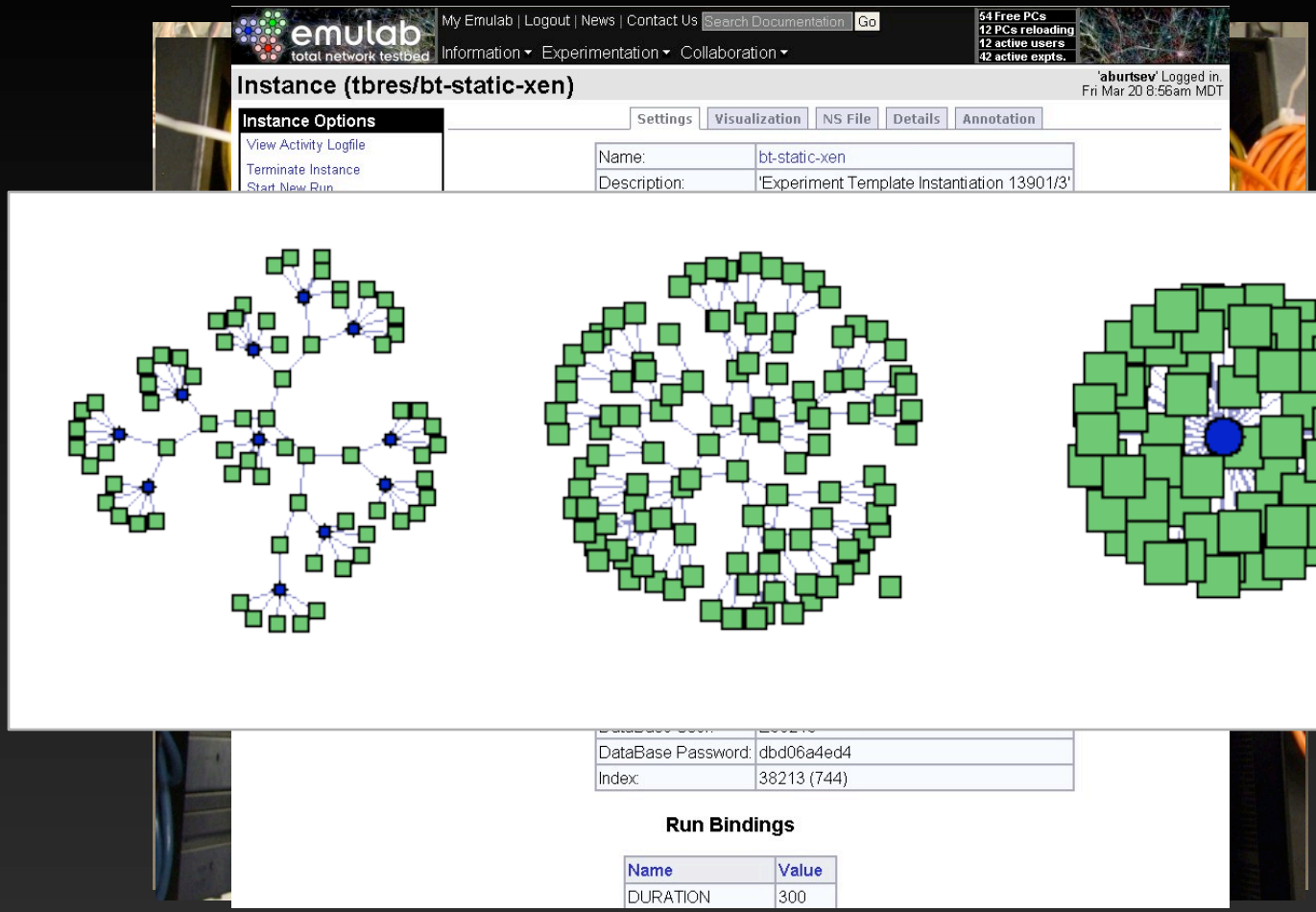
Transparent Checkpoint of Closed Distributed Systems in Emulab

Anton Burtsev, Prashanth Radhakrishnan,
Mike Hibler, and Jay Lepreau

University of Utah, School of Computing

Emulab

- Public testbed for network experimentation



- Complex networking experiments within minutes

Emulab — precise research tool

- Realism:
  - Real dedicated hardware
    - Machines and networks
  - Real operating systems
  - Freedom to configure any component of the software stack
  - Meaningful real-world results
- Control:
  - Closed system
    - Controlled external dependencies and side effects
  - Control interface
  - Repeatable, directed experimentation

Goal: more control over execution

- Stateful swap-out
  - Demand for physical resources exceeds capacity
  - Preemptive experiment scheduling
    - Long-running
    - Large-scale experiments
  - No loss of experiment state
- Time-travel
  - Replay experiments
    - Deterministically or non-deterministically
  - Debugging and analysis aid

Challenge

- Both controls should preserve fidelity of experimentation
- Both rely on *transparency* of distributed checkpoint

Transparent checkpoint

- Traditionally, semantic transparency:
  - Checkpointed execution is one of the possible correct executions
- What if we want to preserve performance correctness?
  - Checkpointed execution is one of the correct executions *closest* to a non-checkpointed run
- Preserve measurable parameters of the system
  - CPU allocation
  - Elapsed time
  - Disk throughput
  - Network delay and bandwidth

Traditional view

- Local case
  - Transparency = smallest possible downtime
  - Several milliseconds [Remus]
  - Background work
  - Harms realism
- Distributed case
  - Lamport checkpoint
    - Provides consistency
  - Packet delays, timeouts, traffic bursts, replay buffer overflows

Main insight

- Conceal checkpoint from the system under test
  - But still stay on the real hardware as much as possible
- "Instantly" freeze the system
  - Time and execution
  - Ensure atomicity of checkpoint
    - Single non-divisible action
- Conceal checkpoint by time virtualization

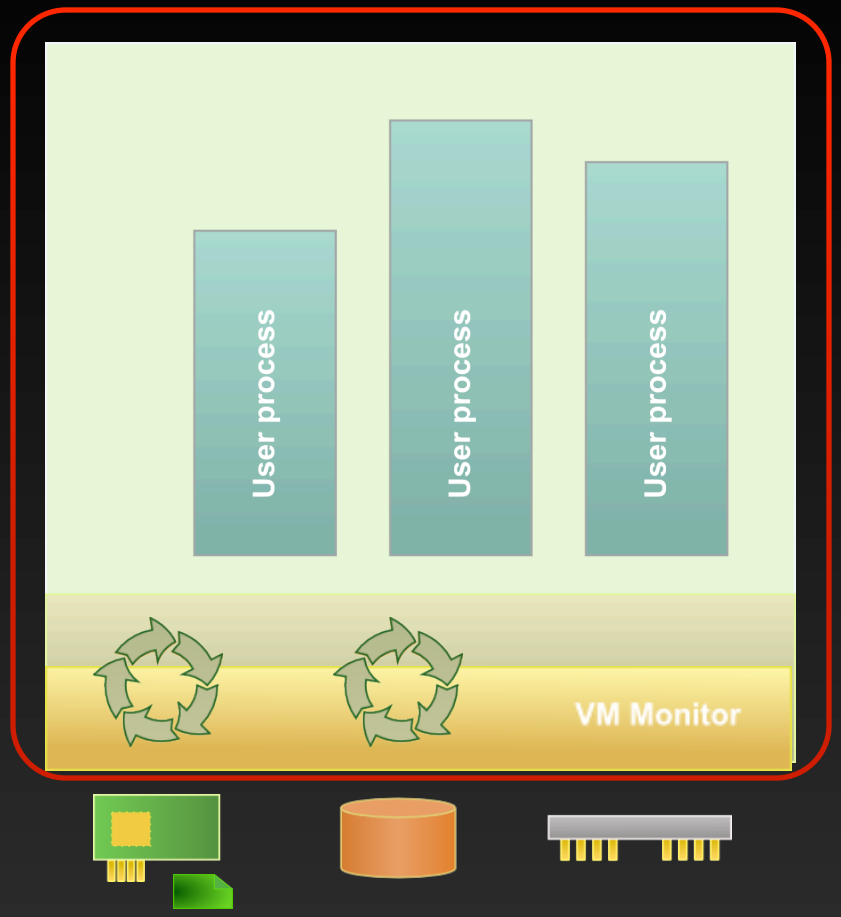
Contributions

- Transparency of distributed checkpoint
- Local atomicity
  - Temporal firewall
- Execution control mechanisms for Emulab
  - Stateful swap-out
  - Time-travel
- Branching storage

Challenges and implementation

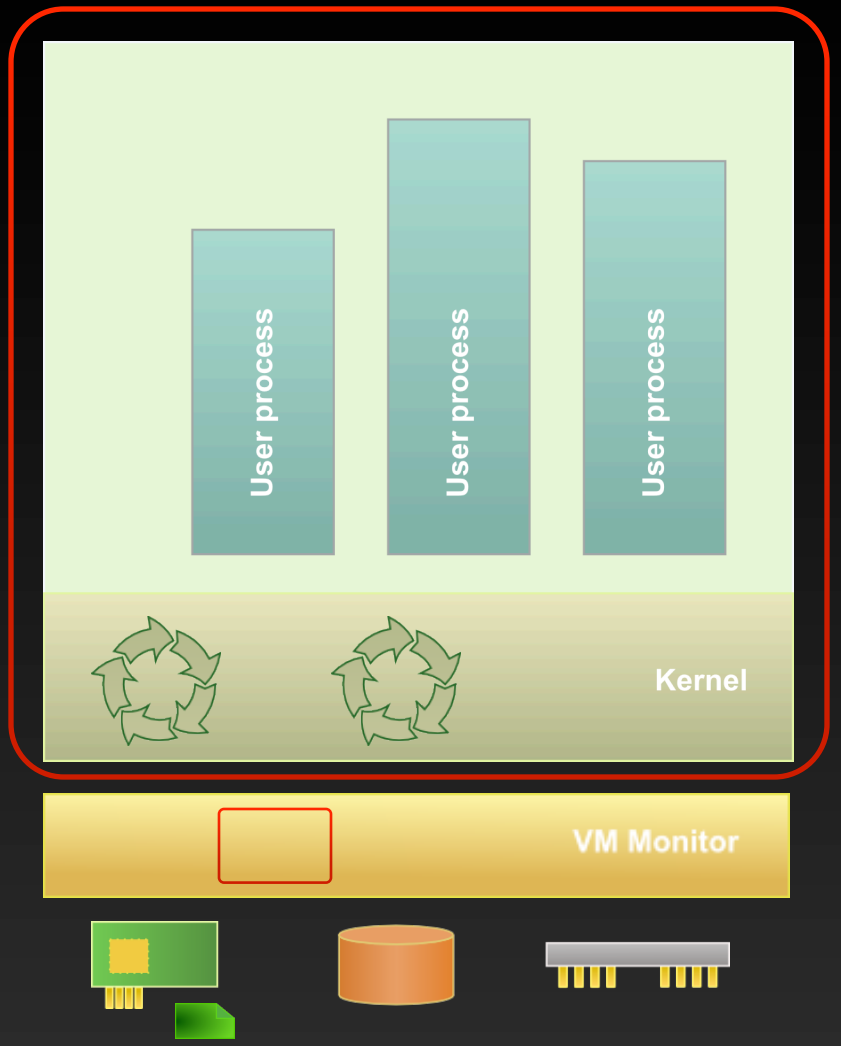
Checkpoint essentials

- State encapsulation
  - Suspend execution
  - Save running state of the system
- Virtualization layer



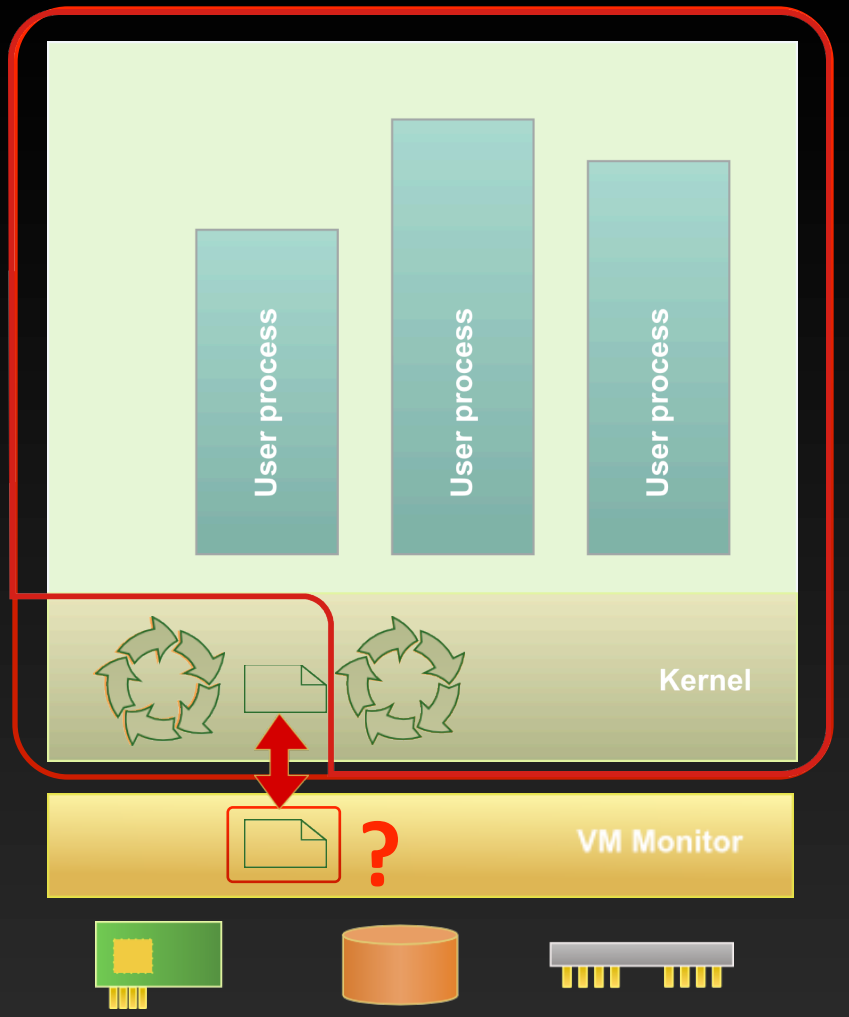
Checkpoint essentials

- State encapsulation
  - Suspend execution
  - Save running state of the system
- Virtualization layer
  - Suspends the system
  - Saves its state
  - Saves in-flight state
  - Disconnects/reconnects to the hardware



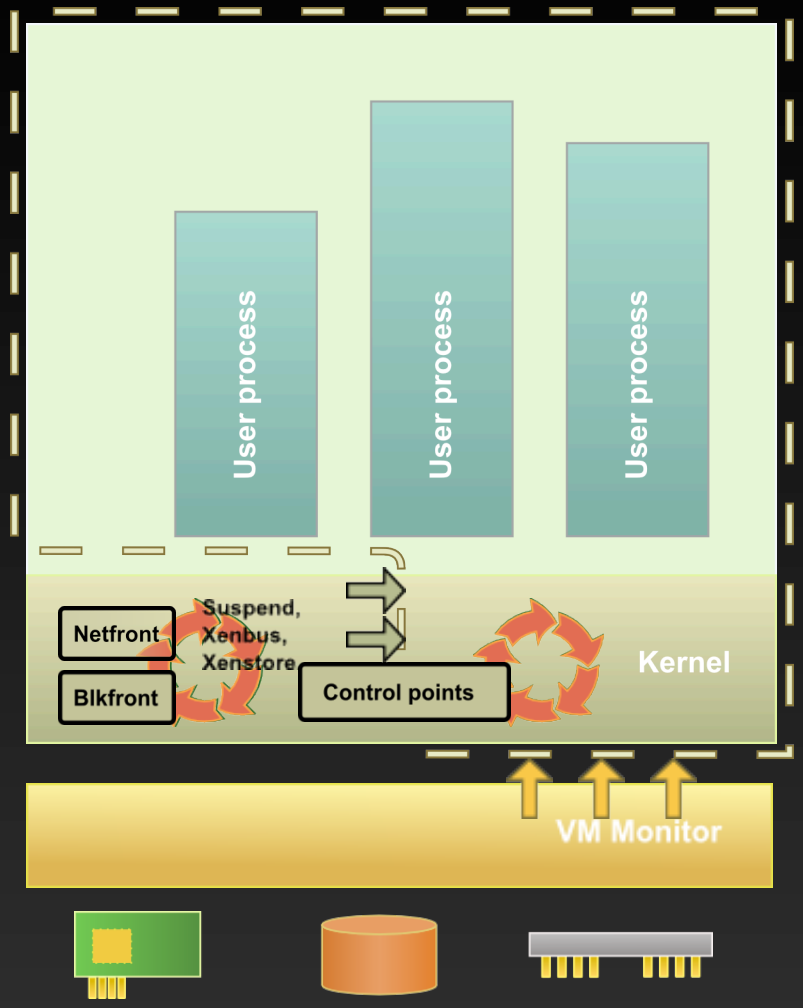
First challenge: atomicity

- Permanent encapsulation is harmful
  - Too slow
  - Some state is shared
- Encapsulated upon checkpoint
- Externally to VM
  - Full memory virtualization
  - Needs declarative description of shared state
- Internally to VM
  - Breaks atomicity



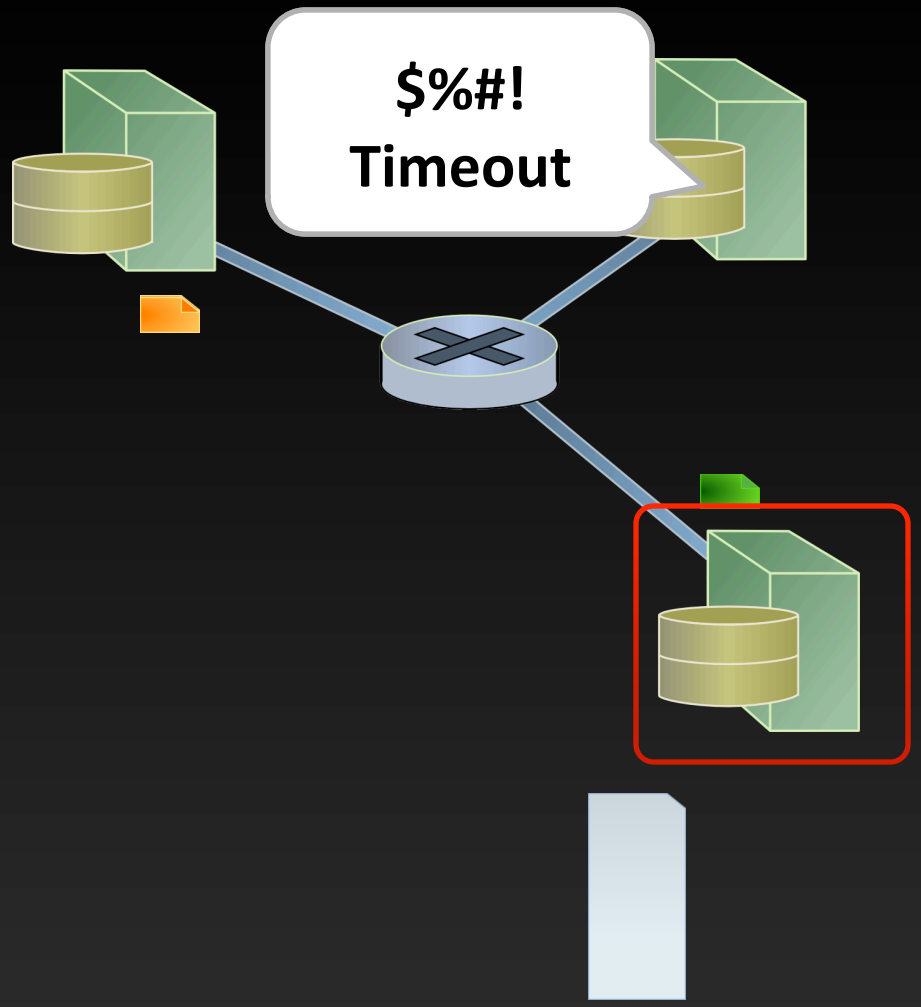
Atomicity in the local case

- Temporal firewall
  - Selectively suspends execution and time
  - Provides atomicity inside the firewall
- Execution control in the Linux kernel
  - Kernel threads
  - Interrupts, exceptions, IRQs
- Conceals checkpoint
  - Time virtualization



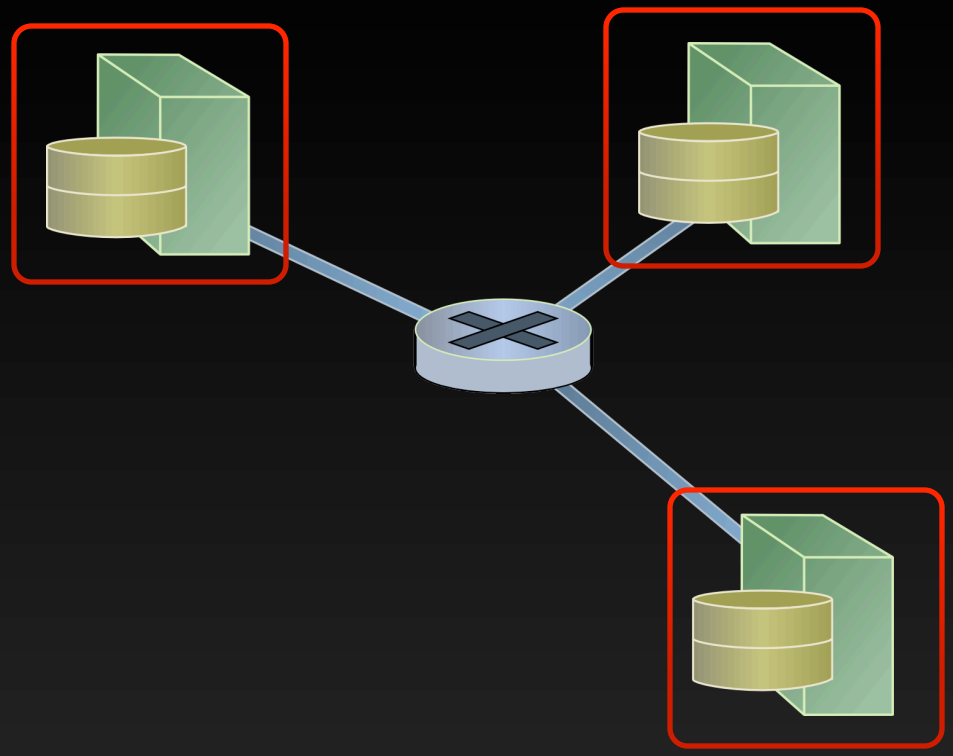
Second challenge: synchronization

- Lamport checkpoint
  - No synchronization
  - System is partially suspended
- Preserves consistency
  - Logs in-flight packets
- Once logged it's impossible to remove
- Unsuspended nodes
  - Time-outs



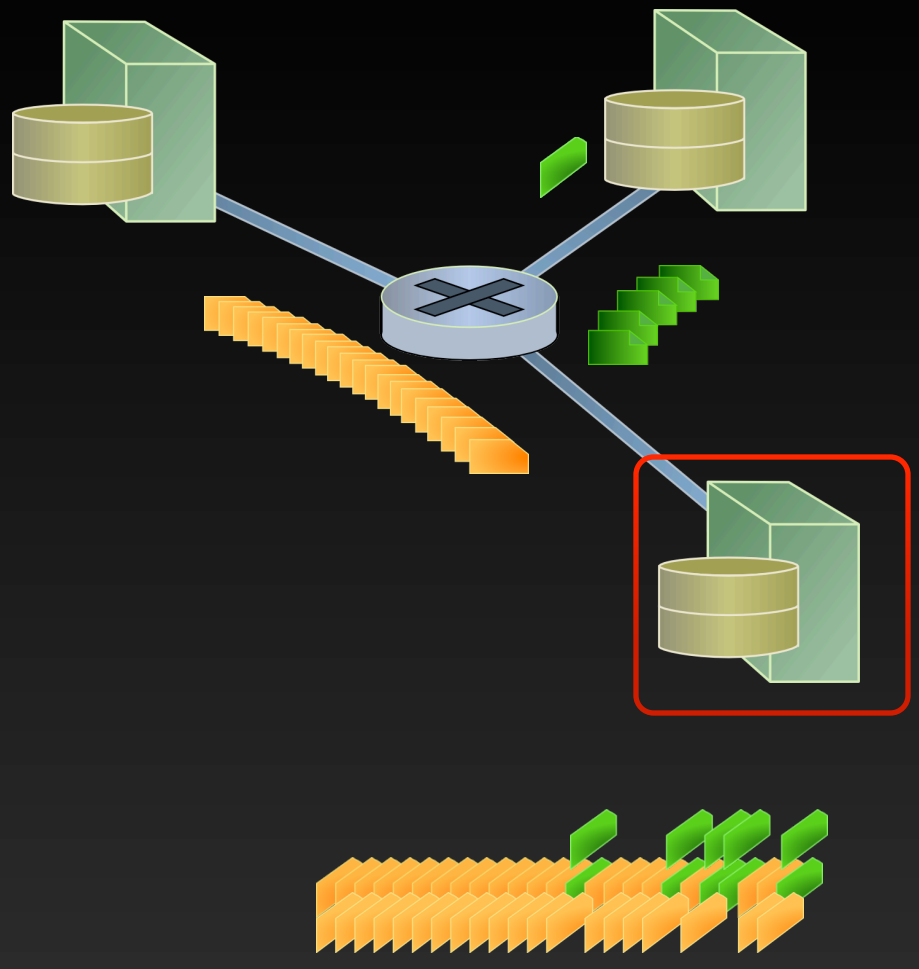
Synchronized checkpoint

- Synchronize clocks across the system
- Schedule checkpoint
- Checkpoint all nodes at once
- Almost no in-flight packets



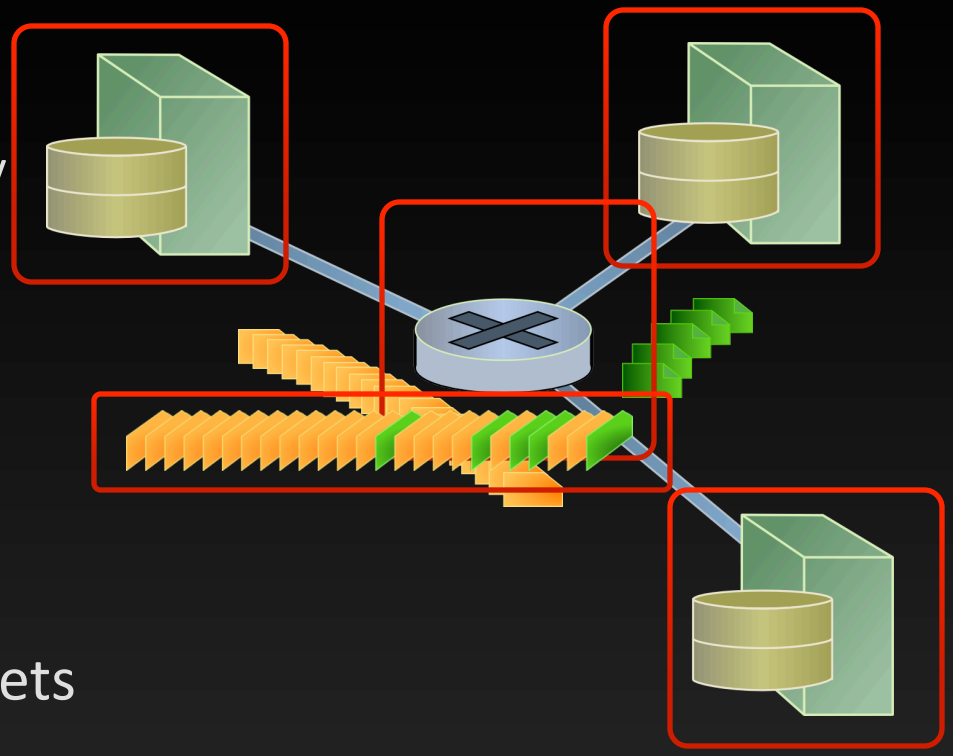
Bandwidth-delay product

- Large number of in-flight packets
- Slow links dominate the log
- Faster links wait for the entire log to complete
- Per-path replay?
  - Unavailable at Layer 2
  - Accurate replay engine on every node



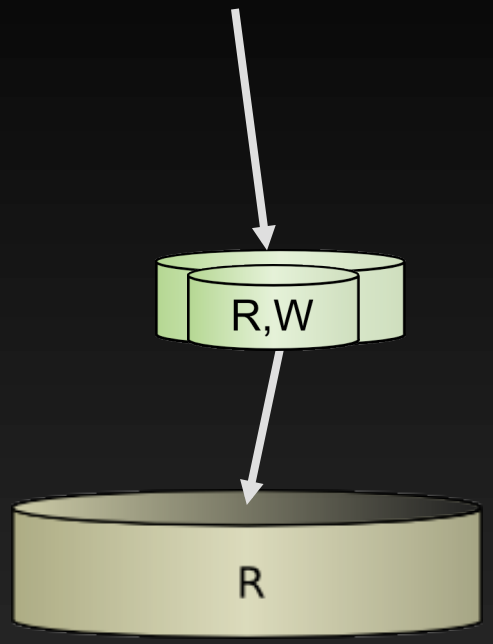
Checkpoint the network core

- Leverage Emulab delay nodes
  - Emulab links are no-delay
  - Link emulation done by delay nodes
- Avoid replay of in-flight packets
- Capture all in-flight packets in core
  - Checkpoint delay nodes



Efficient branching storage

- To be practical stateful swap-out has to be fast
- Mostly read-only FS
  - Shared across nodes and experiments
- Deltas accumulate across swap-outs
- Based on LVM
  - Many optimizations

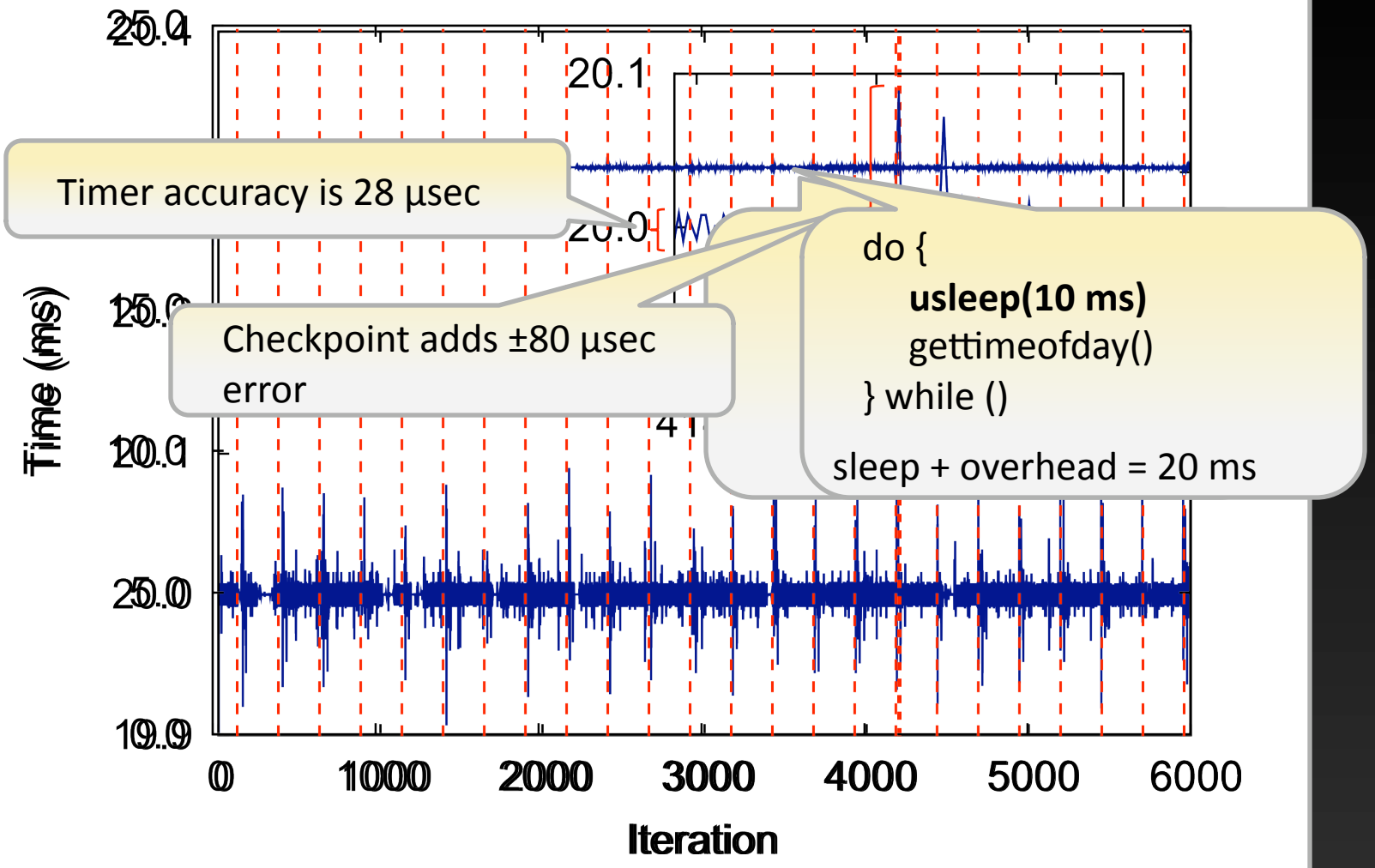


Evaluation

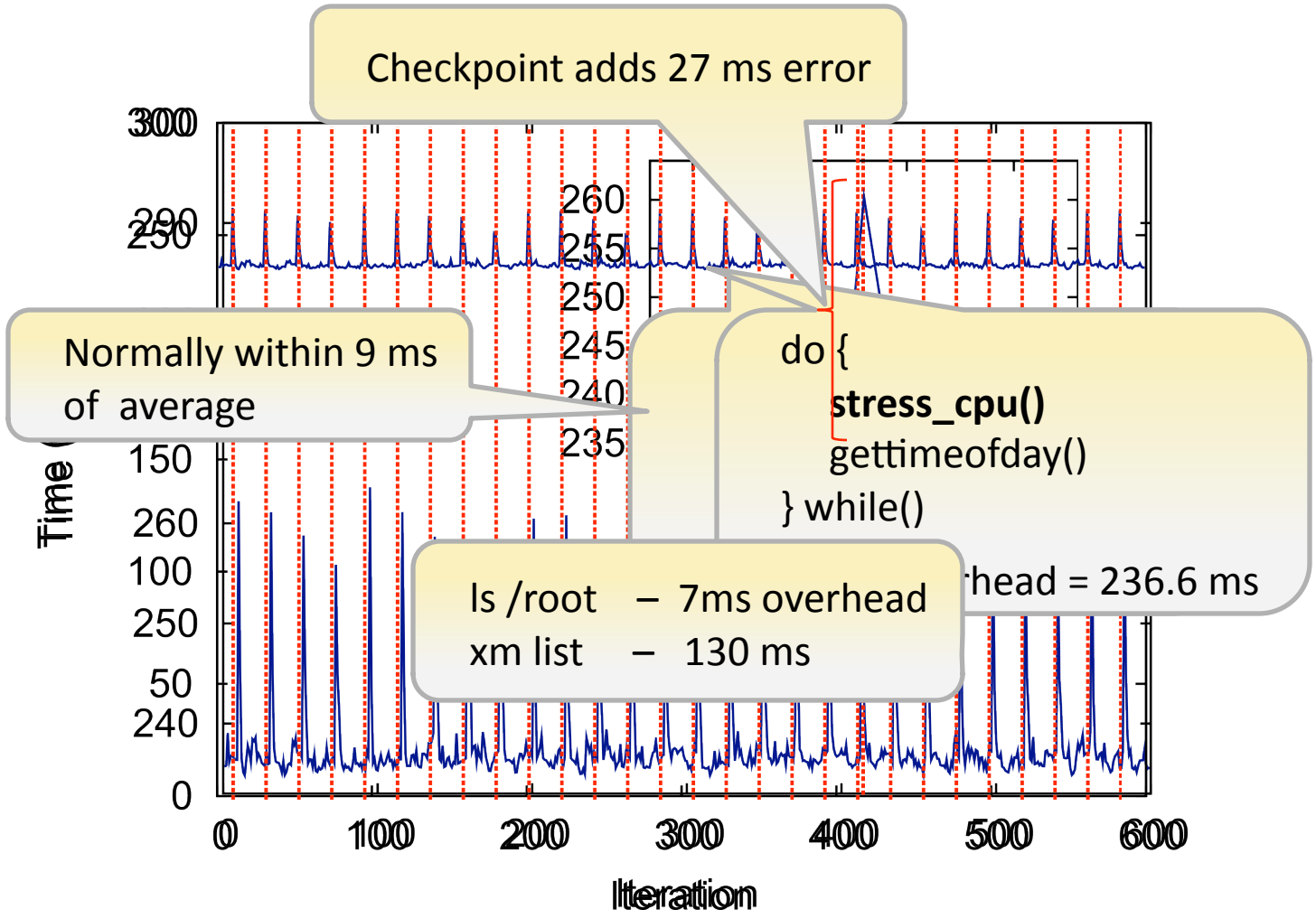
Evaluation plan

- Transparency of the checkpoint
- Measurable metrics
  - Time virtualization
  - CPU allocation
  - Network parameters

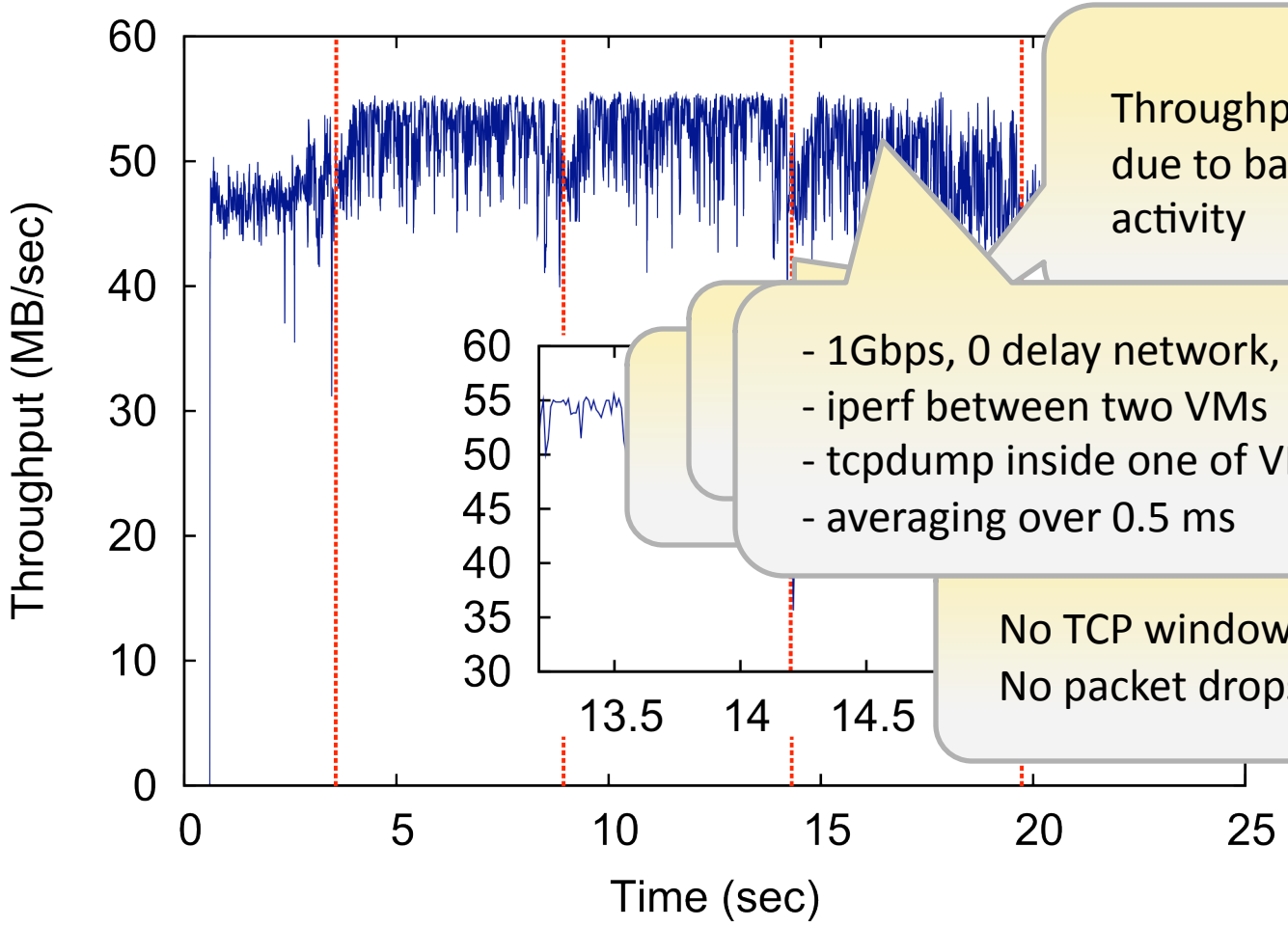
Time virtualization



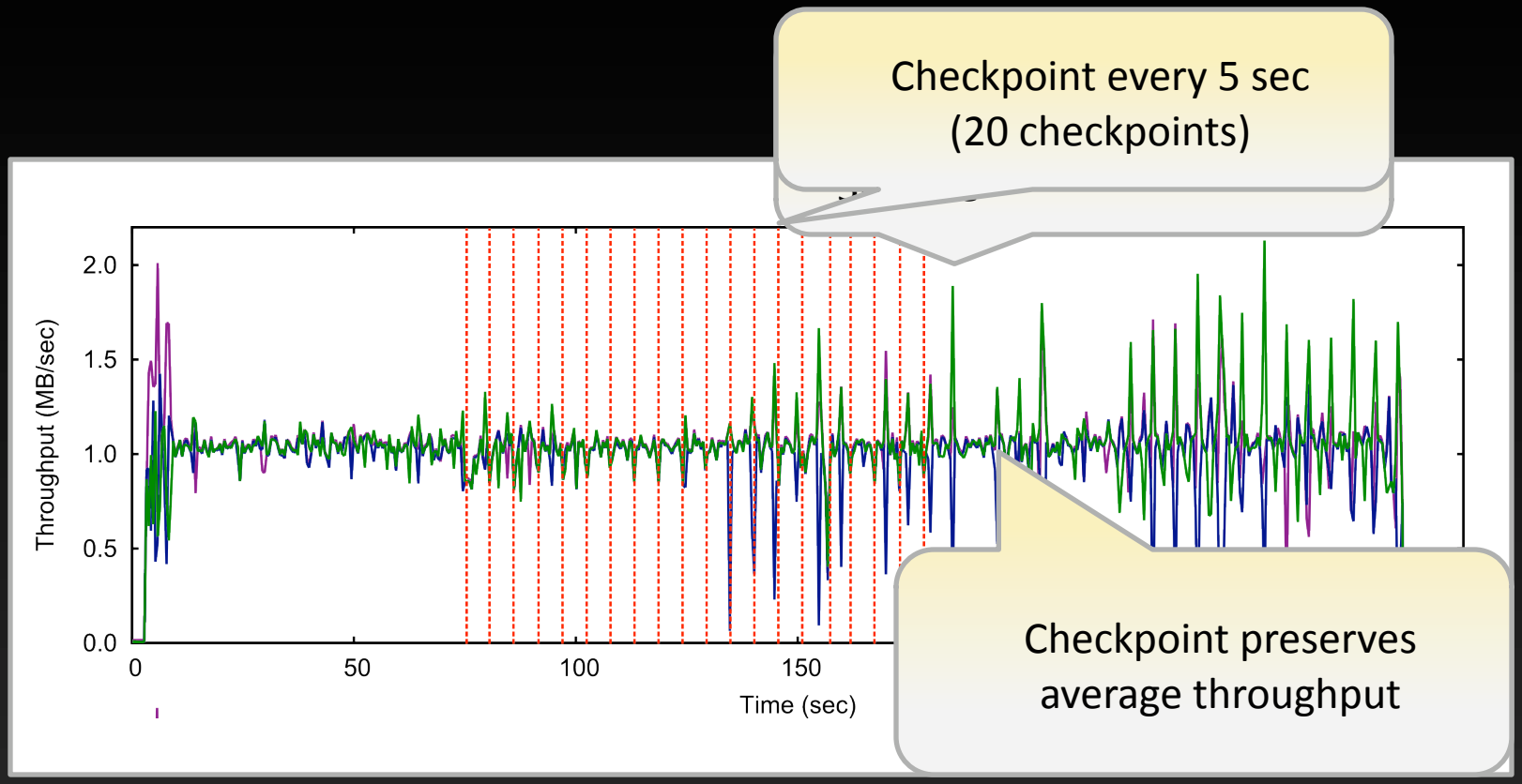
CPU allocation



Network transparency: iperf



Network transparency: BitTorrent



Conclusions

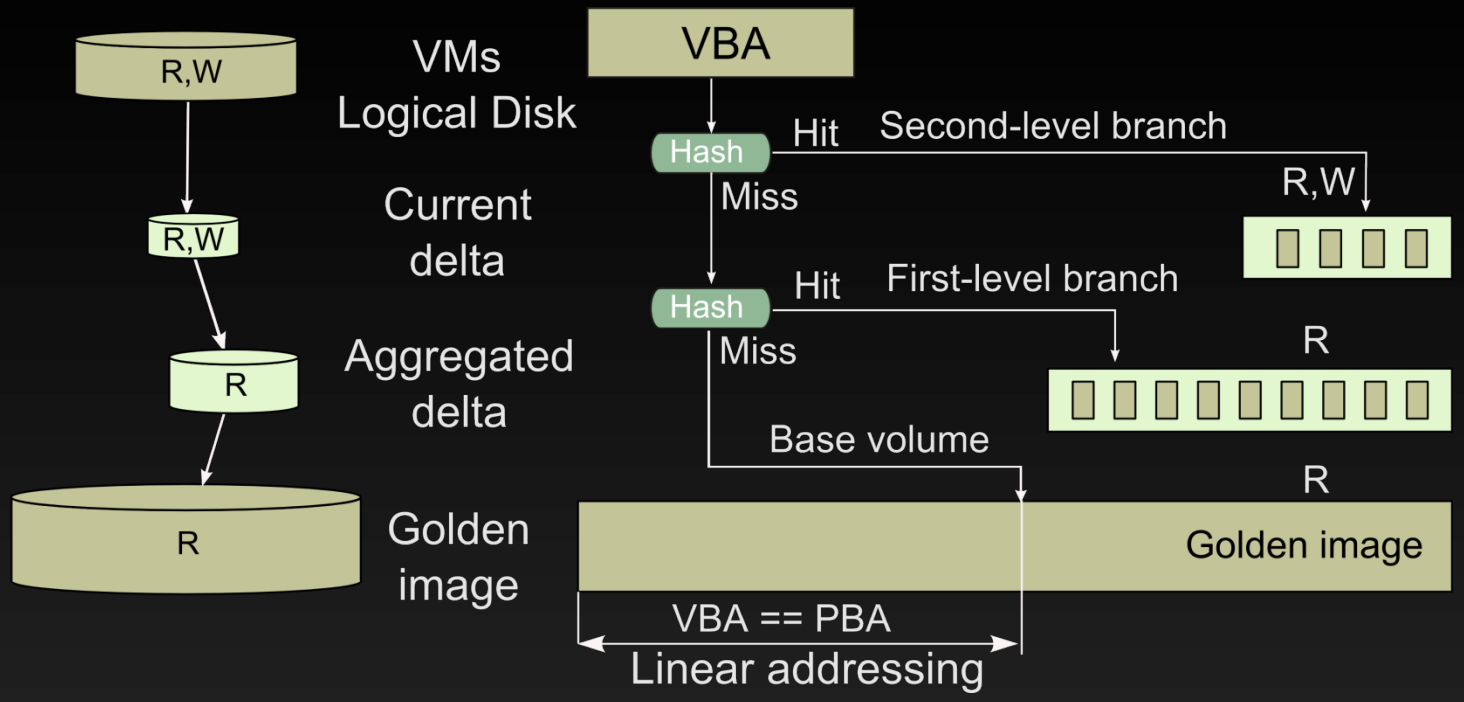
- Transparent distributed checkpoint
  - Precise research tool
  - Fidelity of distributed system analysis
- Temporal firewall
  - General mechanism to change perception of time for the system
  - Conceal various external events
- Future work is time-travel

Thank you

aburtsev@flux.utah.edu

Backup

Branching storage



- Copy-on-write as a redo log
- Linear addressing
- Free block elimination
- Read before write elimination

Branching storage