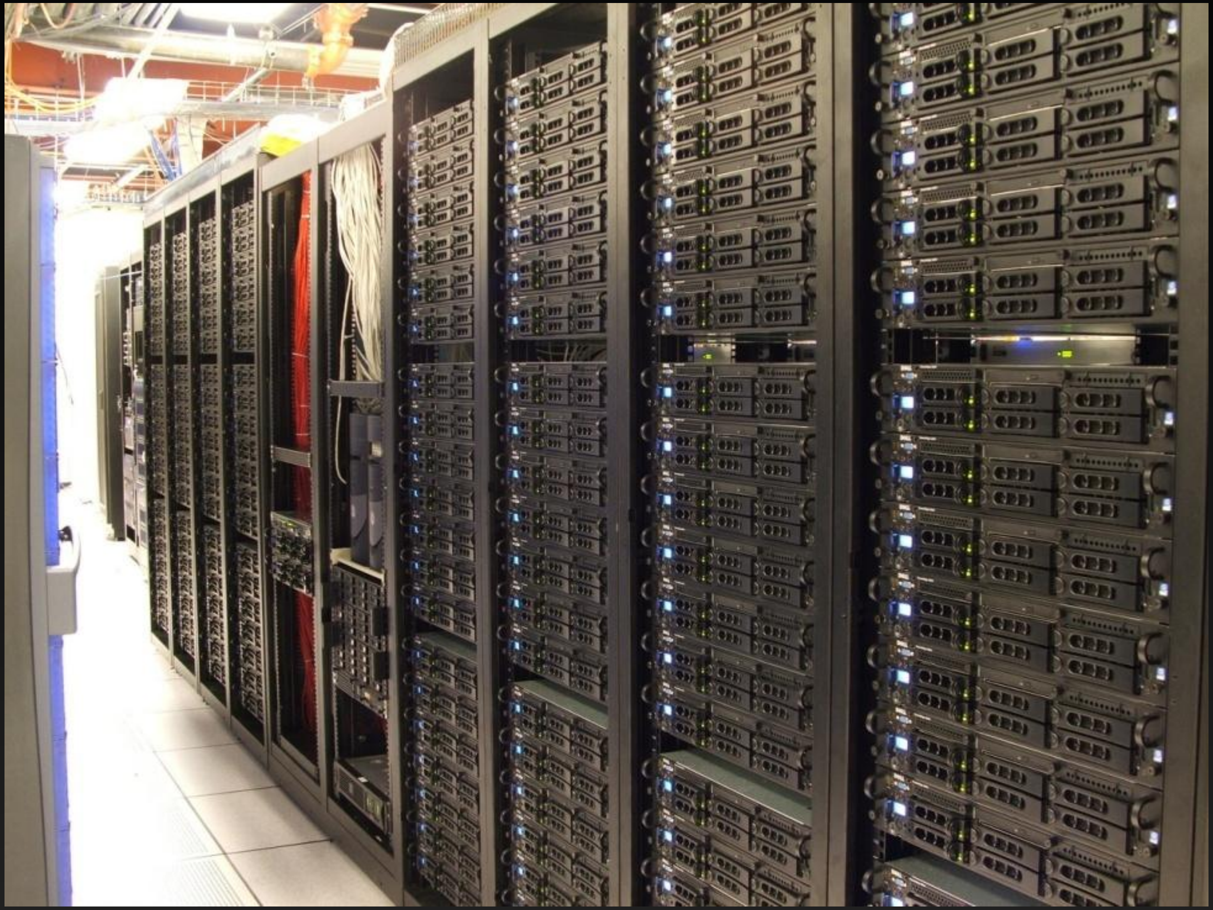
# Transparent Checkpoint of Closed Distributed Systems in Emulab

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

University of Utah, School of Computing

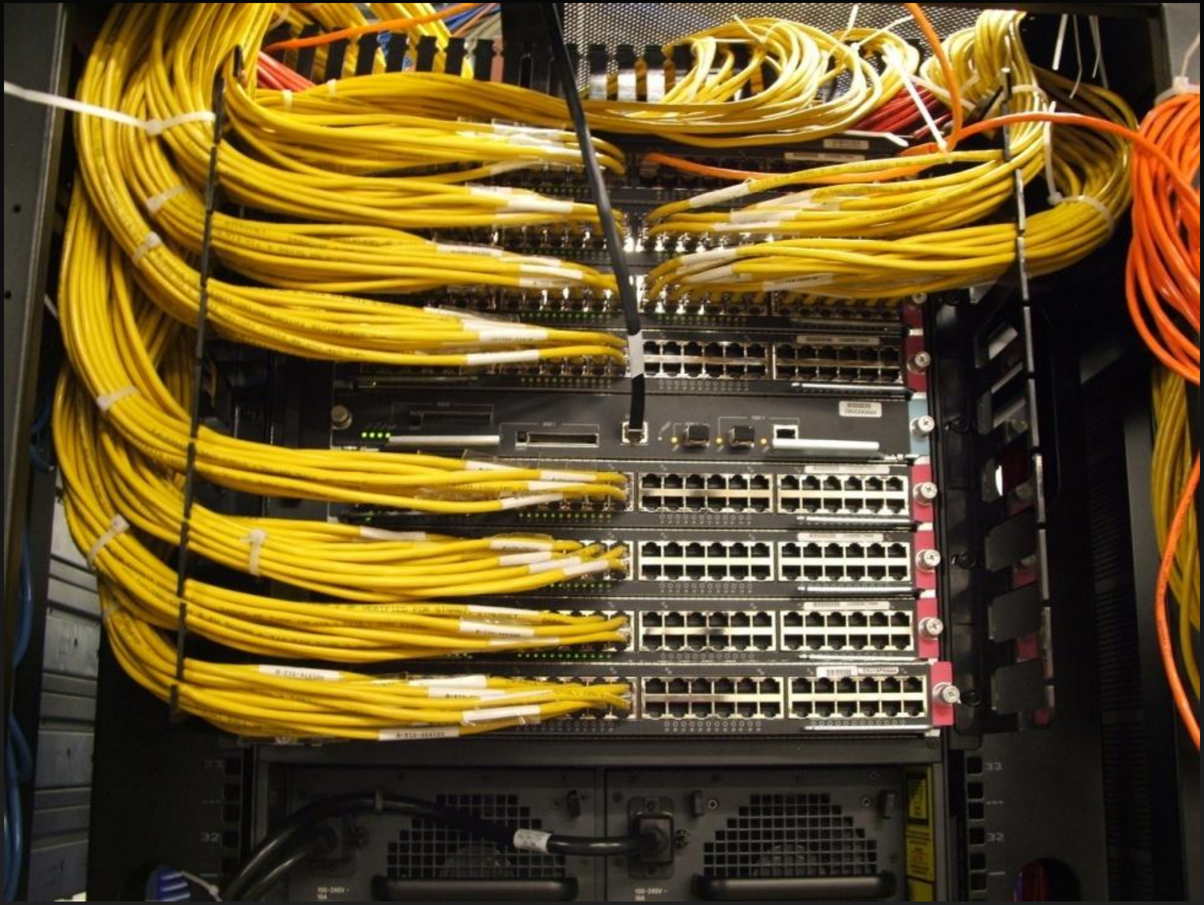
## Emulab

- Public testbed for network experimentation



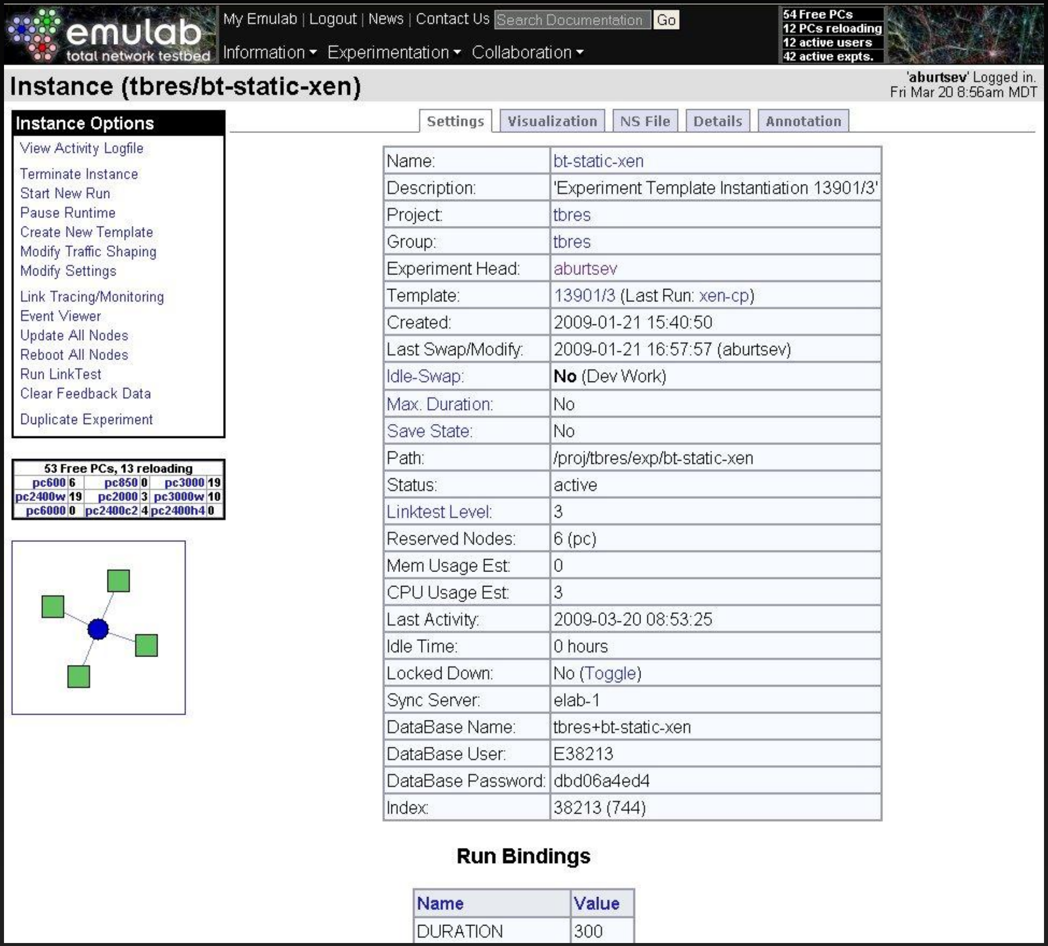
## Emulab

- Public testbed for network experimentation



## Emulab

- Public testbed for network experimentation



54 Free PCs
12 PCs reloading
12 active users
42 active expts.

### Instance (tbres/bt-static-xen)

'aburtsev' Logged in.
Fri Mar 20 8:56am MDT

**Instance Options**

- View Activity Logfile
- Terminate Instance
- Start New Run
- Pause Runtime
- Create New Template
- Modify Traffic Shaping
- Modify Settings
- Link Tracing/Monitoring
- Event Viewer
- Update All Nodes
- Reboot All Nodes
- Run LinkTest
- Clear Feedback Data
- Duplicate Experiment

| 53 Free PCs, 13 reloading | | | | | |
|---|---|---|---|---|---|
| pc600 | 6 | pc850 | 0 | pc3000 | 19 |
| pc2400w | 19 | pc2000 | 3 | pc3000w | 10 |
| pc6000 | 0 | pc2400c2 | 4 | pc2400h4 | 0 |

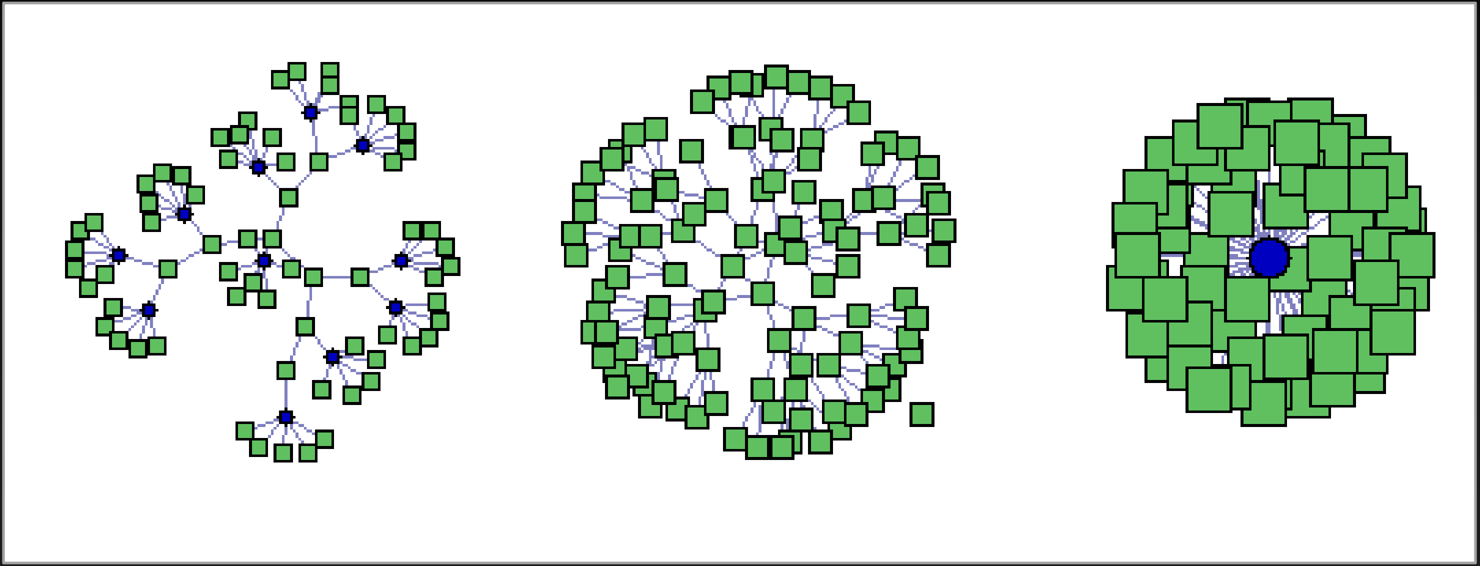
| | |
|---|---|
| Name: | bt-static-xen |
| Description: | 'Experiment Template Instantiation 13901/3' |
| Project: | tbres |
| Group: | tbres |
| Experiment Head: | aburtsev |
| Template: | 13901/3 (Last Run: xen-cp) |
| Created: | 2009-01-21 15:40:50 |
| Last Swap/Modify: | 2009-01-21 16:57:57 (aburtsev) |
| Idle-Swap: | **No** (Dev Work) |
| Max. Duration: | No |
| Save State: | No |
| Path: | /proj/tbres/exp/bt-static-xen |
| Status: | active |
| Linktest Level: | 3 |
| Reserved Nodes: | 6 (pc) |
| Mem Usage Est: | 0 |
| CPU Usage Est: | 3 |
| Last Activity: | 2009-03-20 08:53:25 |
| Idle Time: | 0 hours |
| Locked Down: | No (Toggle) |
| Sync Server: | elab-1 |
| DataBase Name: | tbres+bt-static-xen |
| DataBase User: | E38213 |
| DataBase Password: | dbd06a4ed4 |
| Index: | 38213 (744) |

**Run Bindings**

| Name | Value |
|---|---|
| DURATION | 300 |

## 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

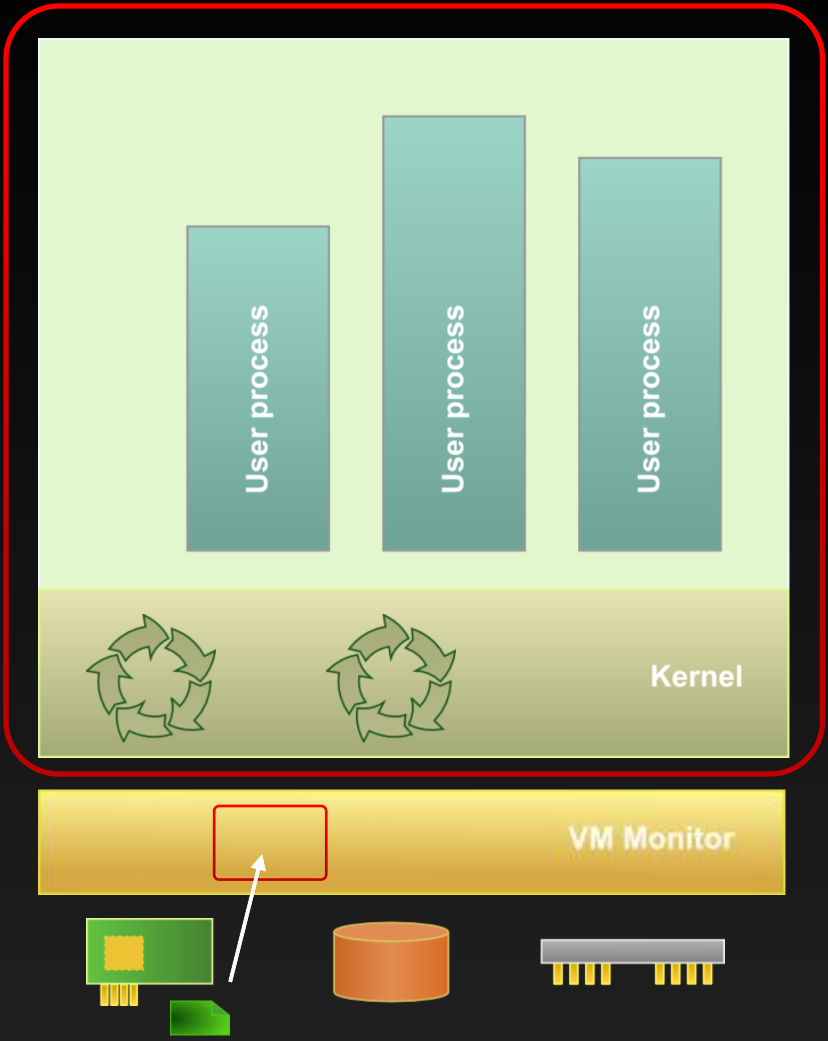
- Execution control mechanisms for Emulab
  - Stateful swap-out
  - Time-travel

- Branching storage

## Challenges and implementation

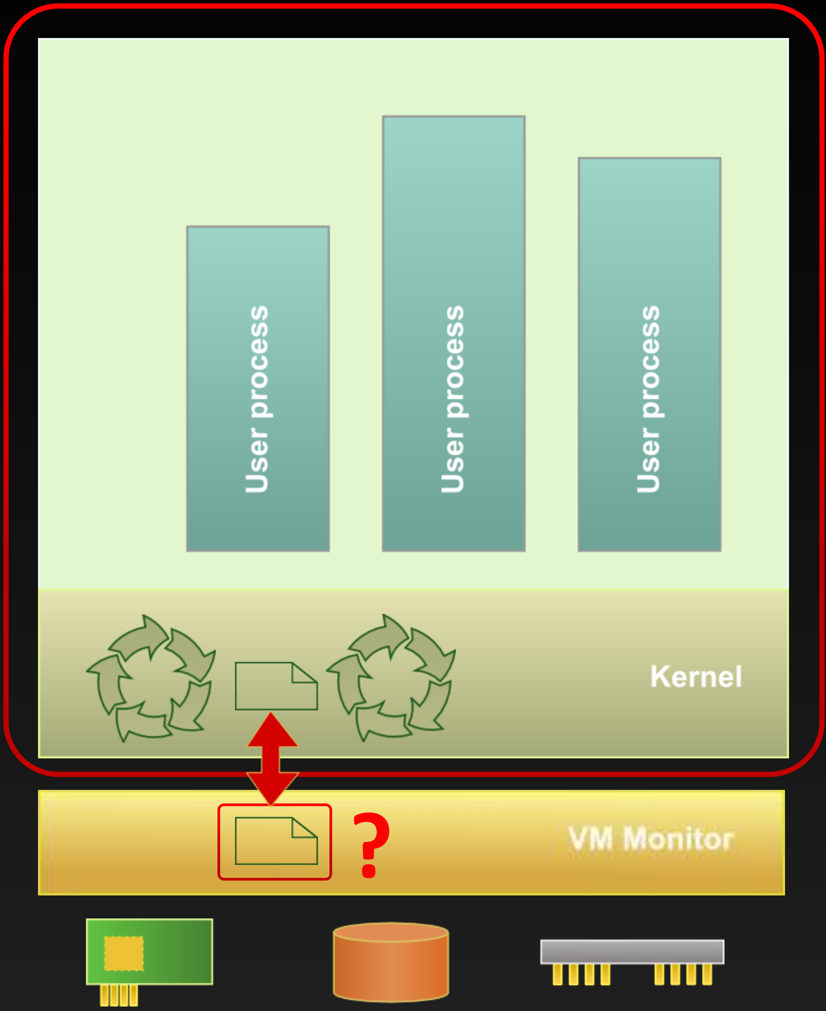
# Checkpoint essentials

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



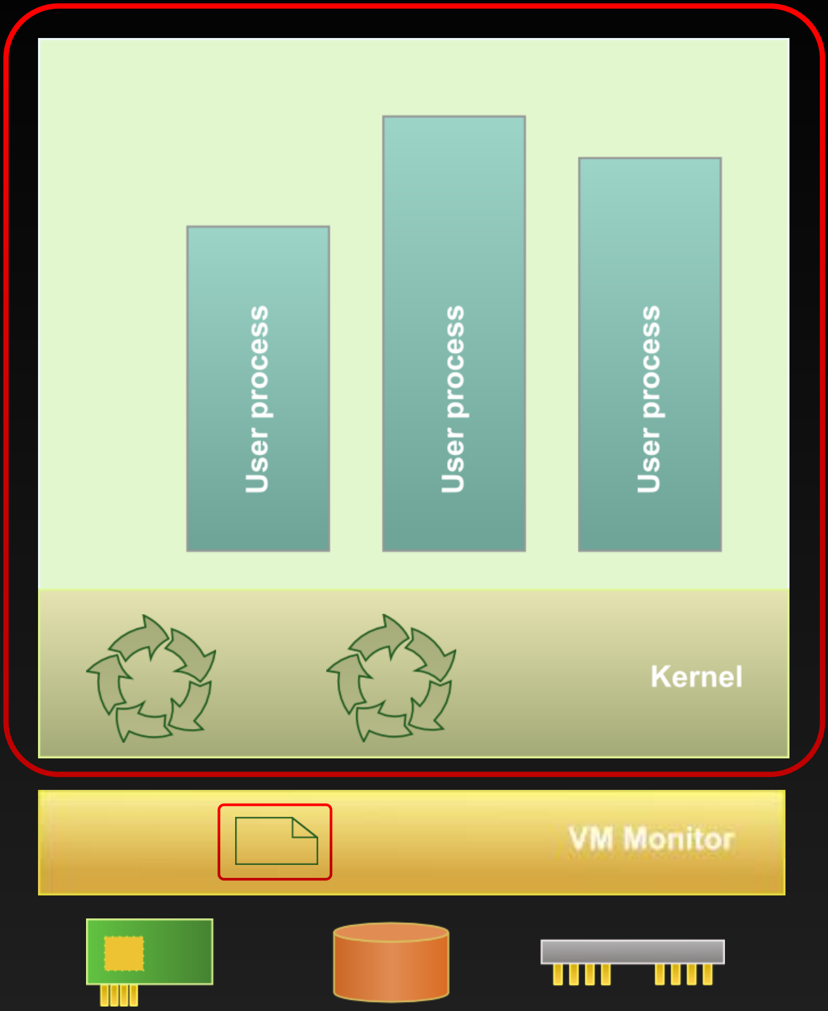
## First challenge: atomicity

- Permanent encapsulation is harmful
  - Too slow
  - Some state is shared
- Encapsulated upon checkpoint



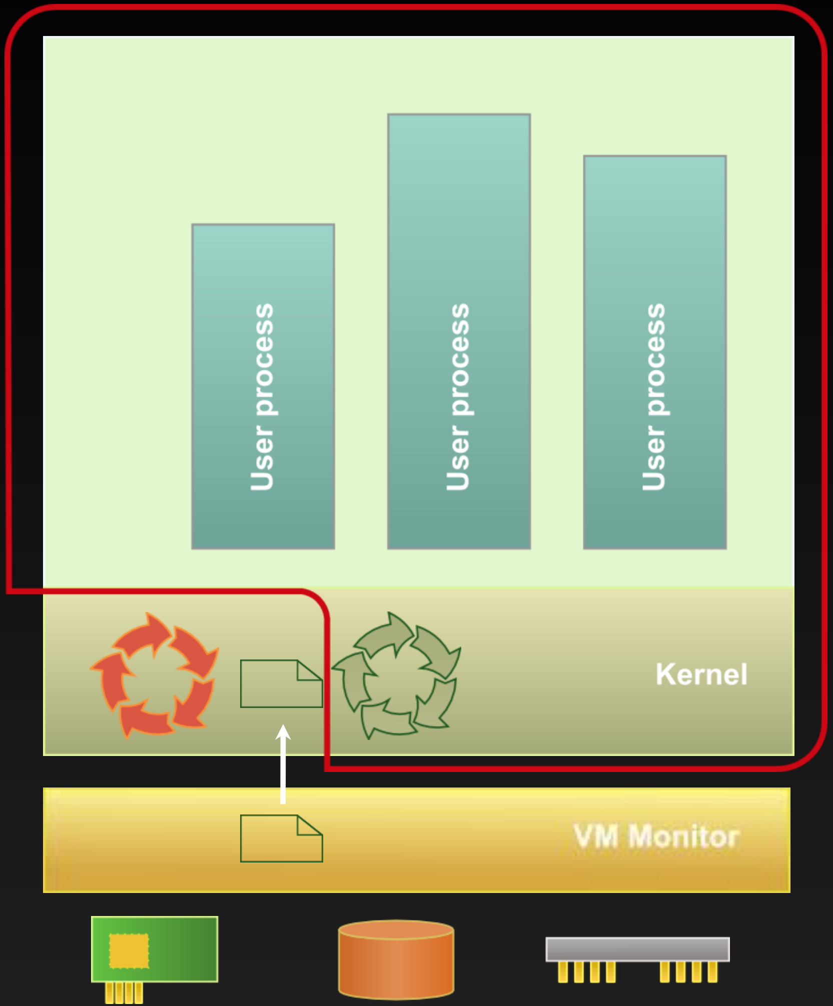
## First challenge: atomicity

- 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



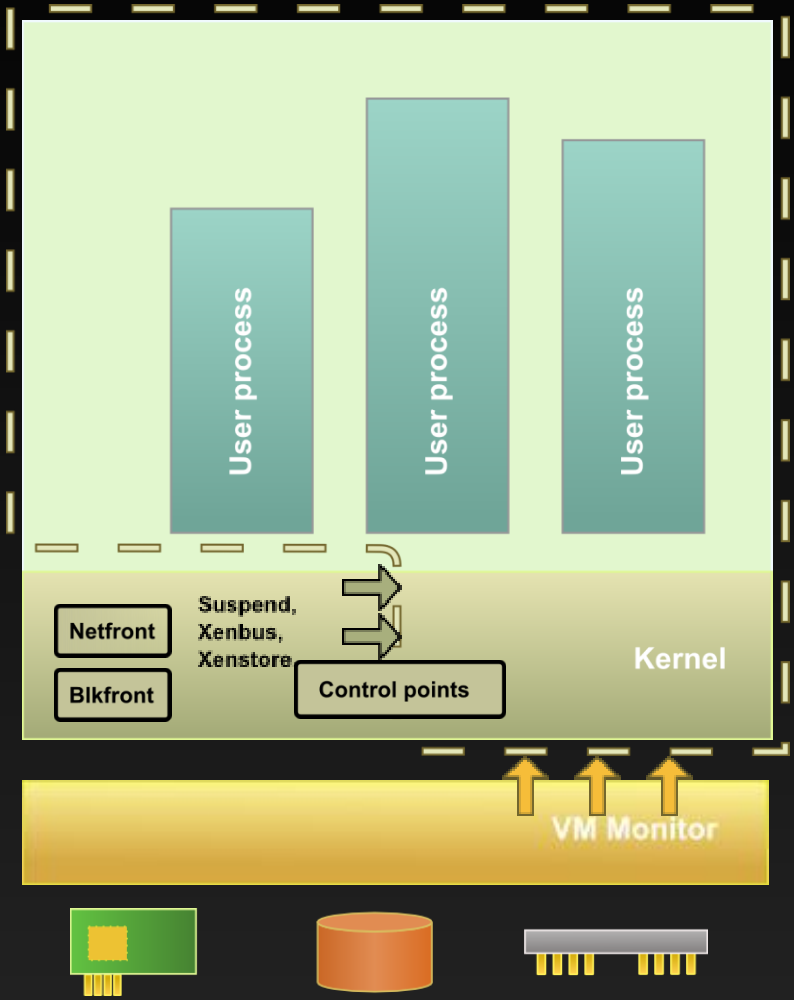
# First challenge: atomicity

- 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



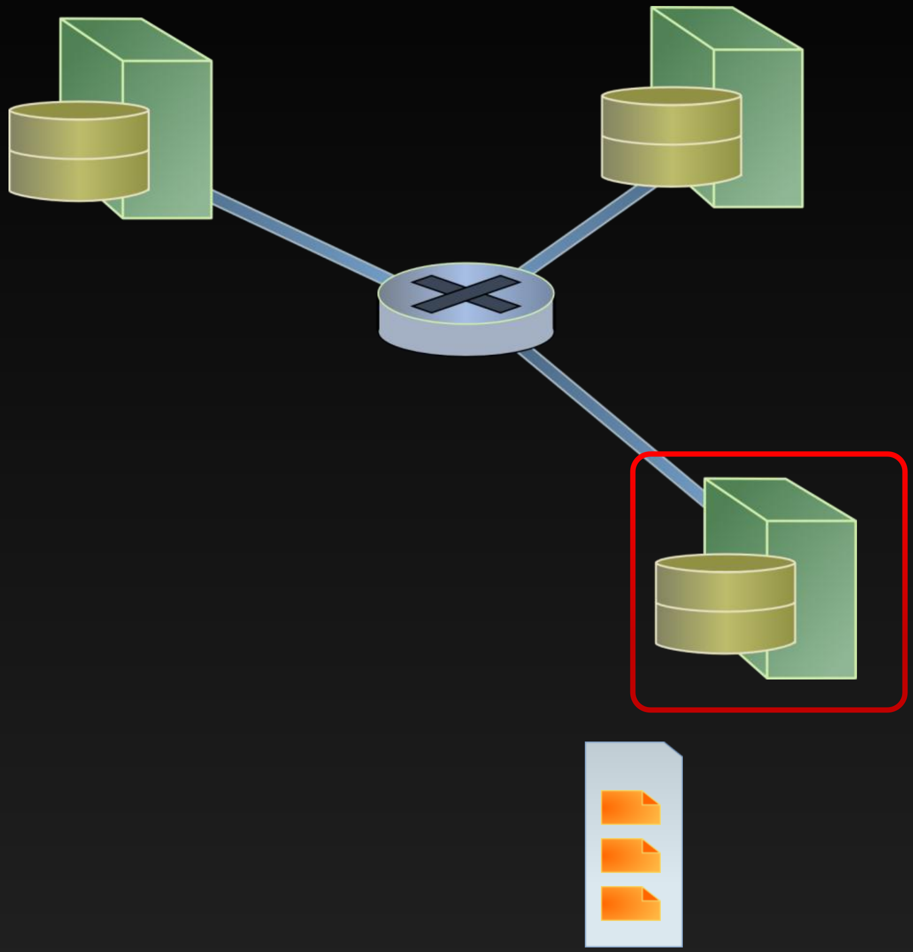
## Atomicity in the local case

- 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



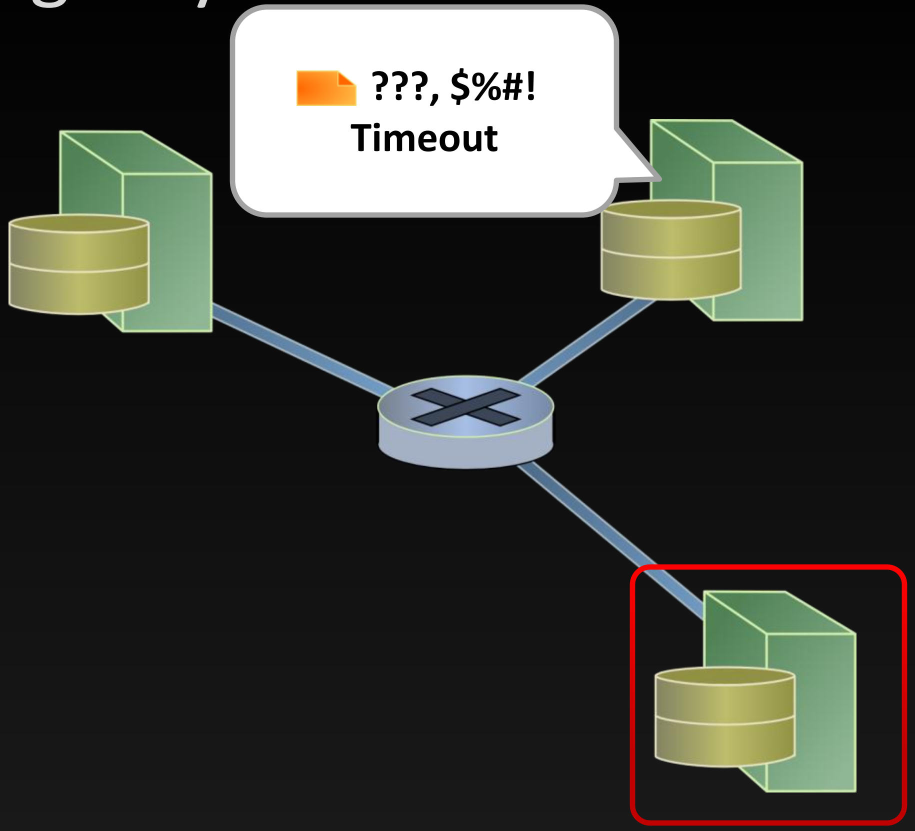
## Second challenge: synchronization

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



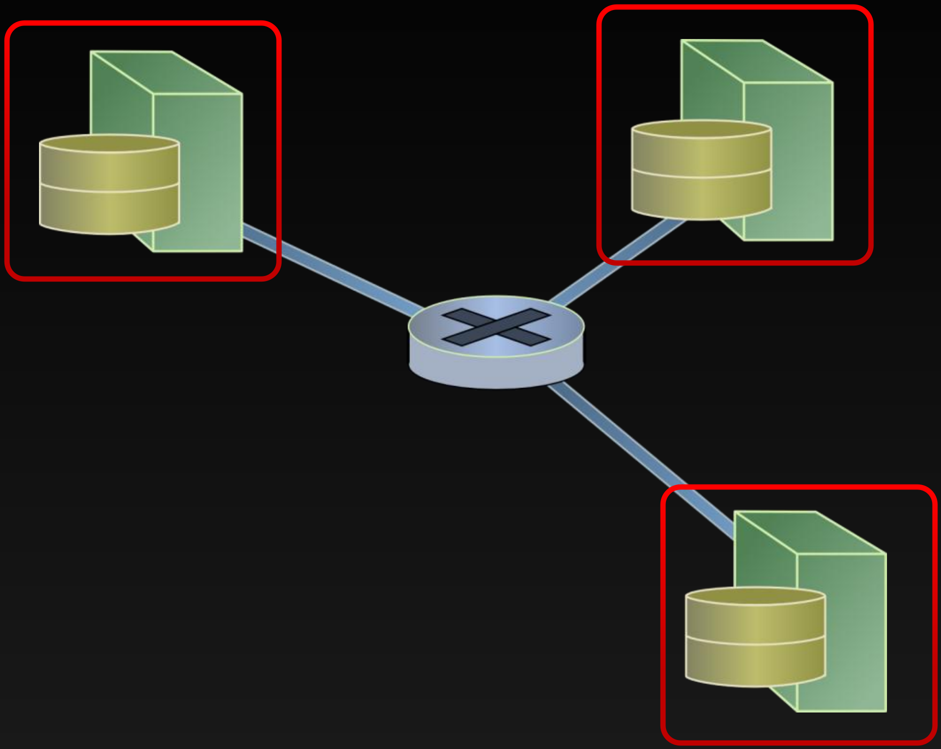
# Second challenge: synchronization

- 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



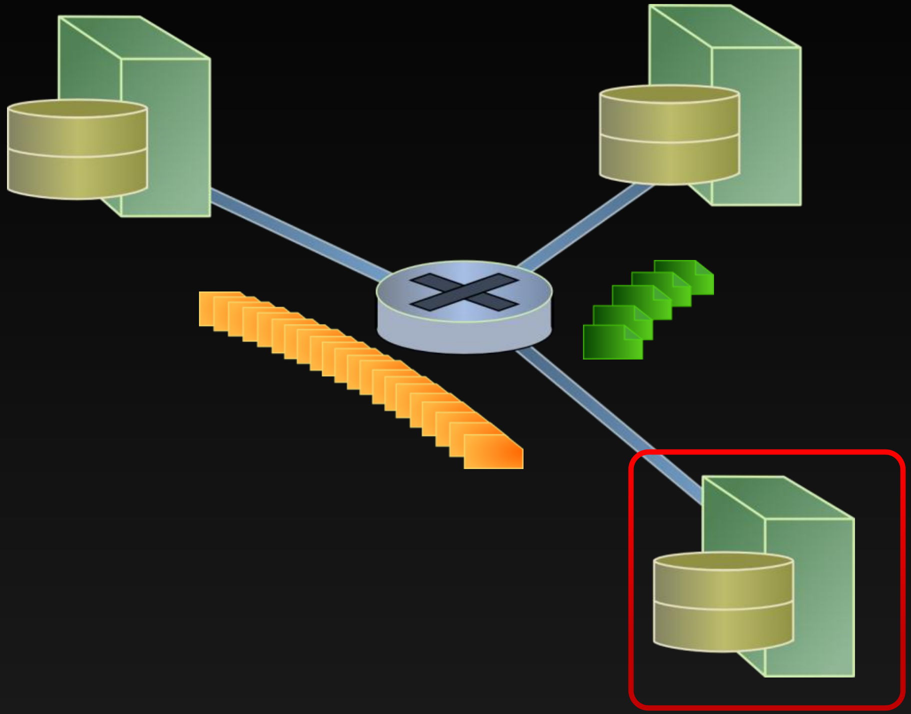
## Synchronized checkpoint

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



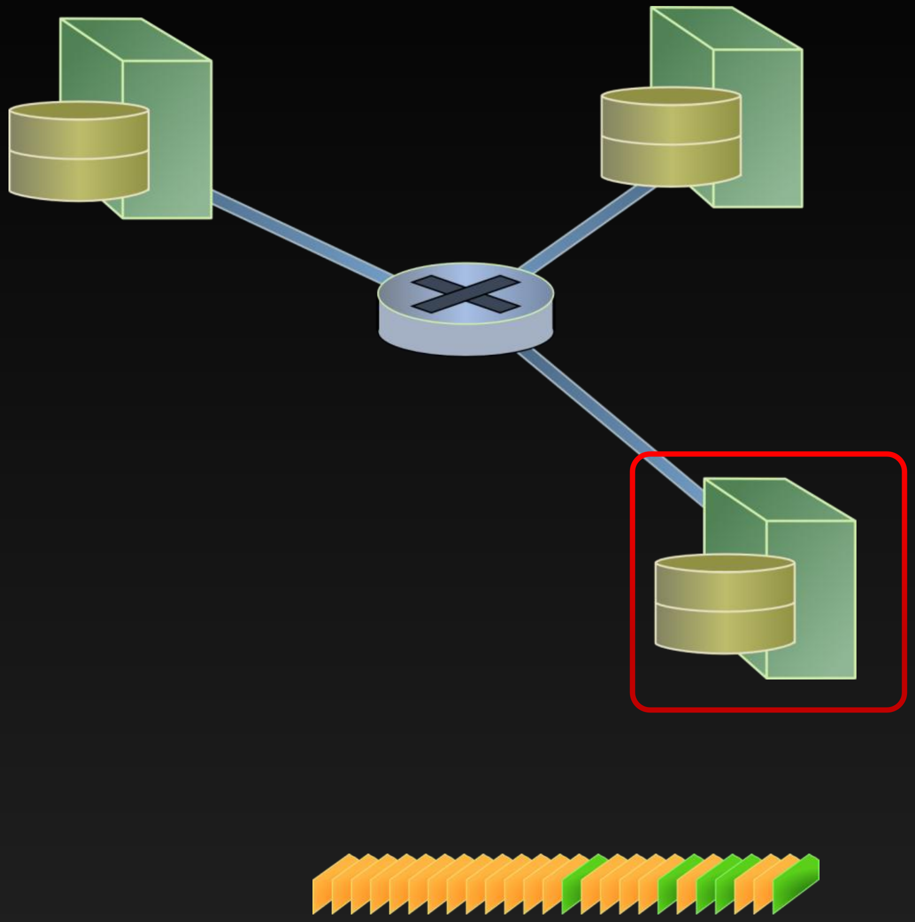
## Bandwidth-delay product

- Large number of in-flight packets



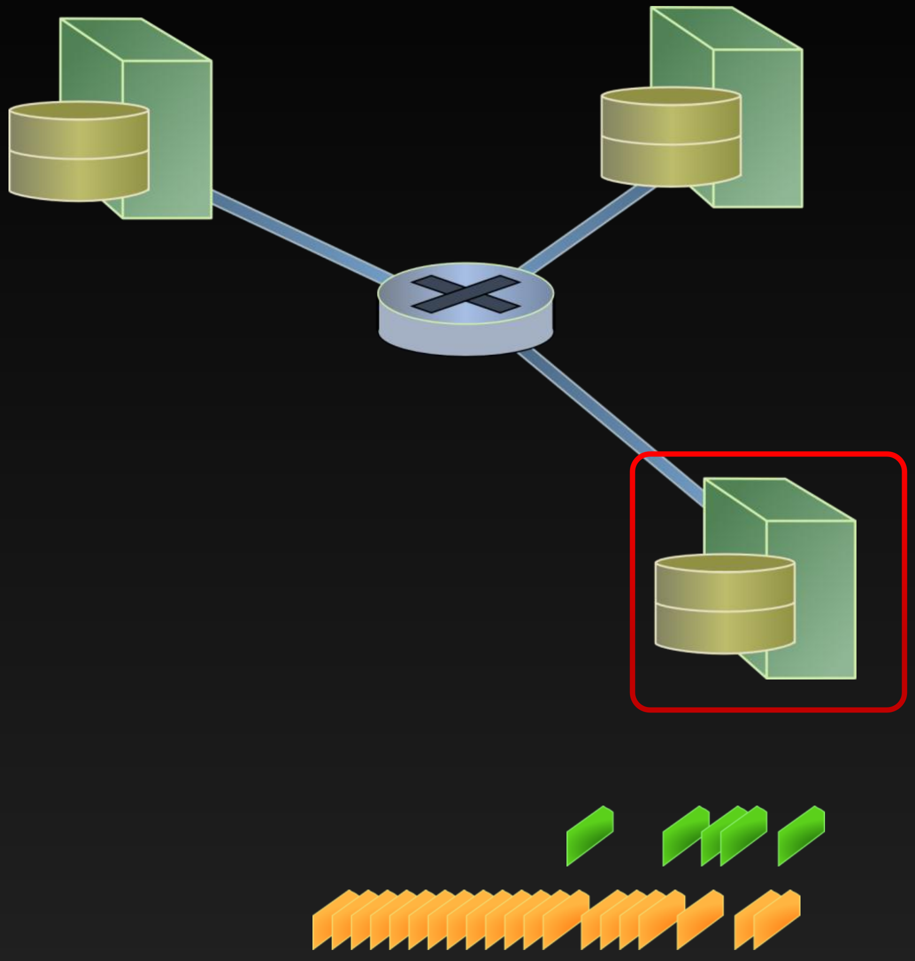
## Bandwidth-delay product

- Large number of in-flight packets
- Slow links dominate the log
- Faster links wait for the entire log to complete



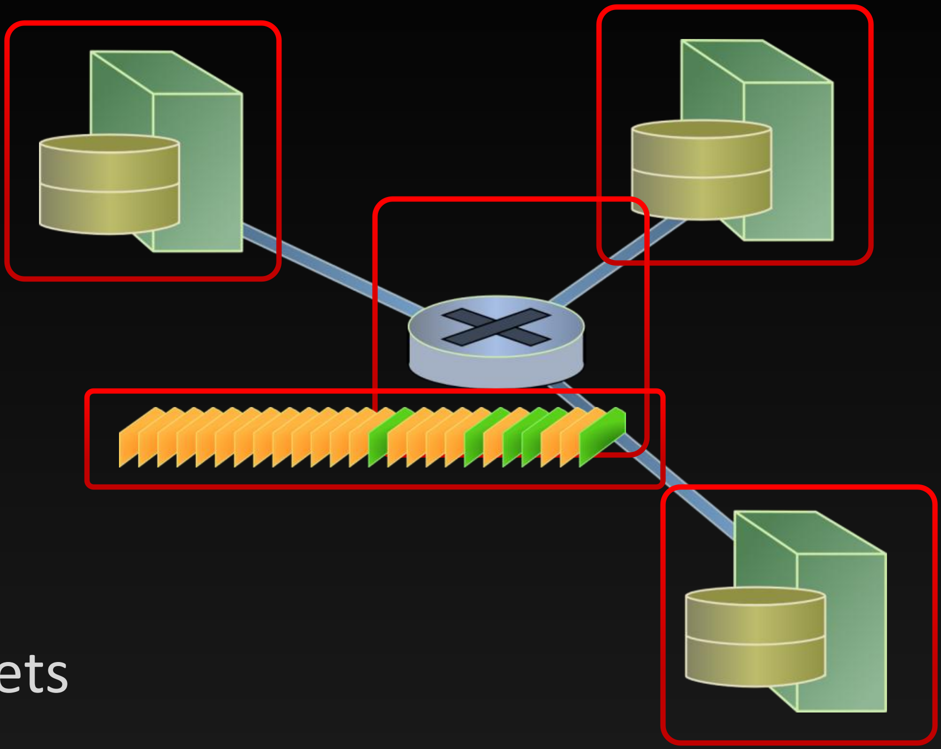
## Bandwidth-delay product

- 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



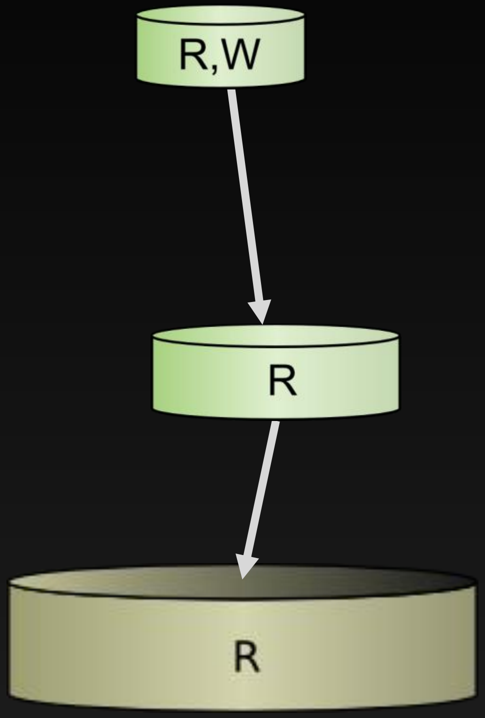
## Checkpoint the network core

- 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

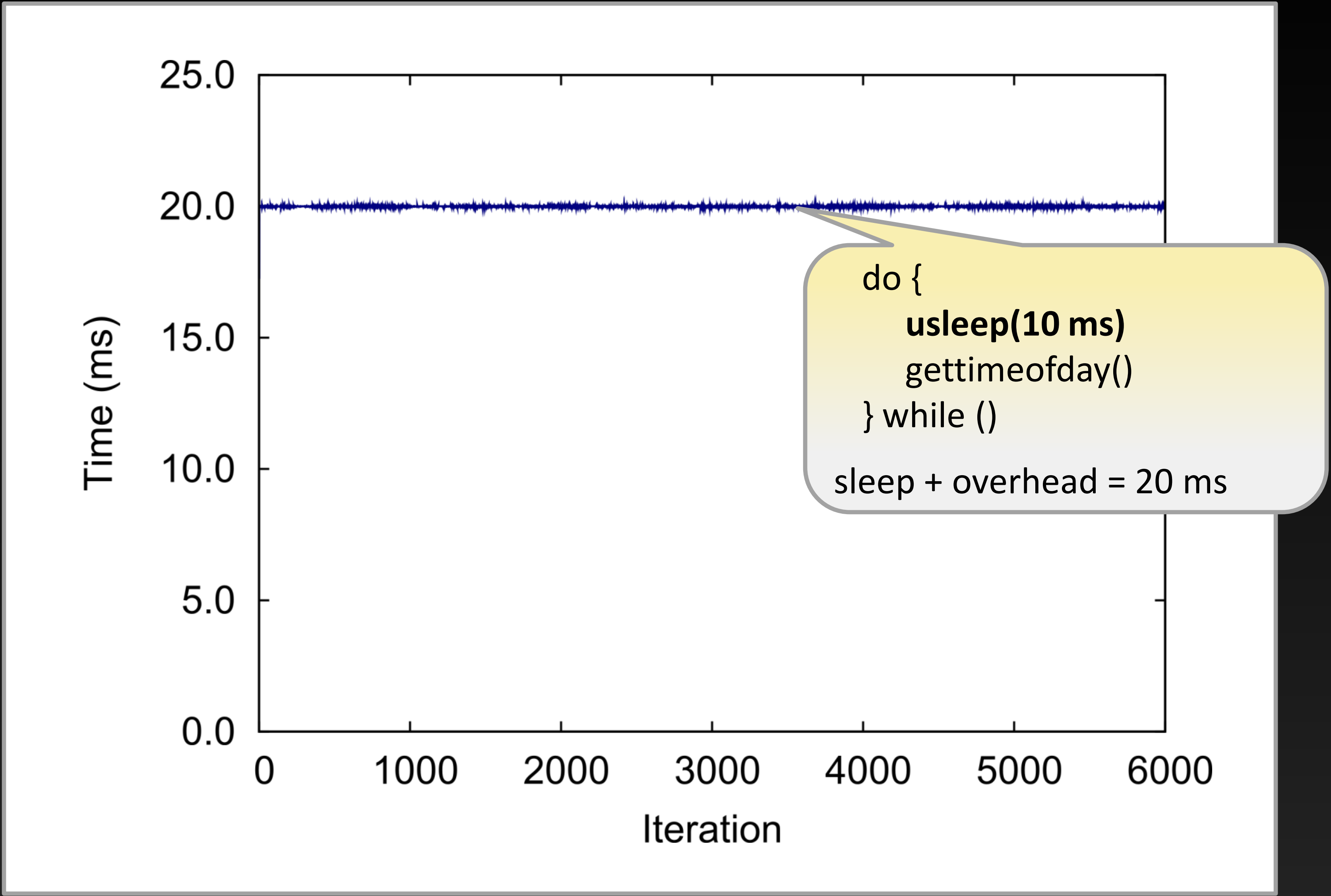


## Evaluation

## Evaluation plan

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

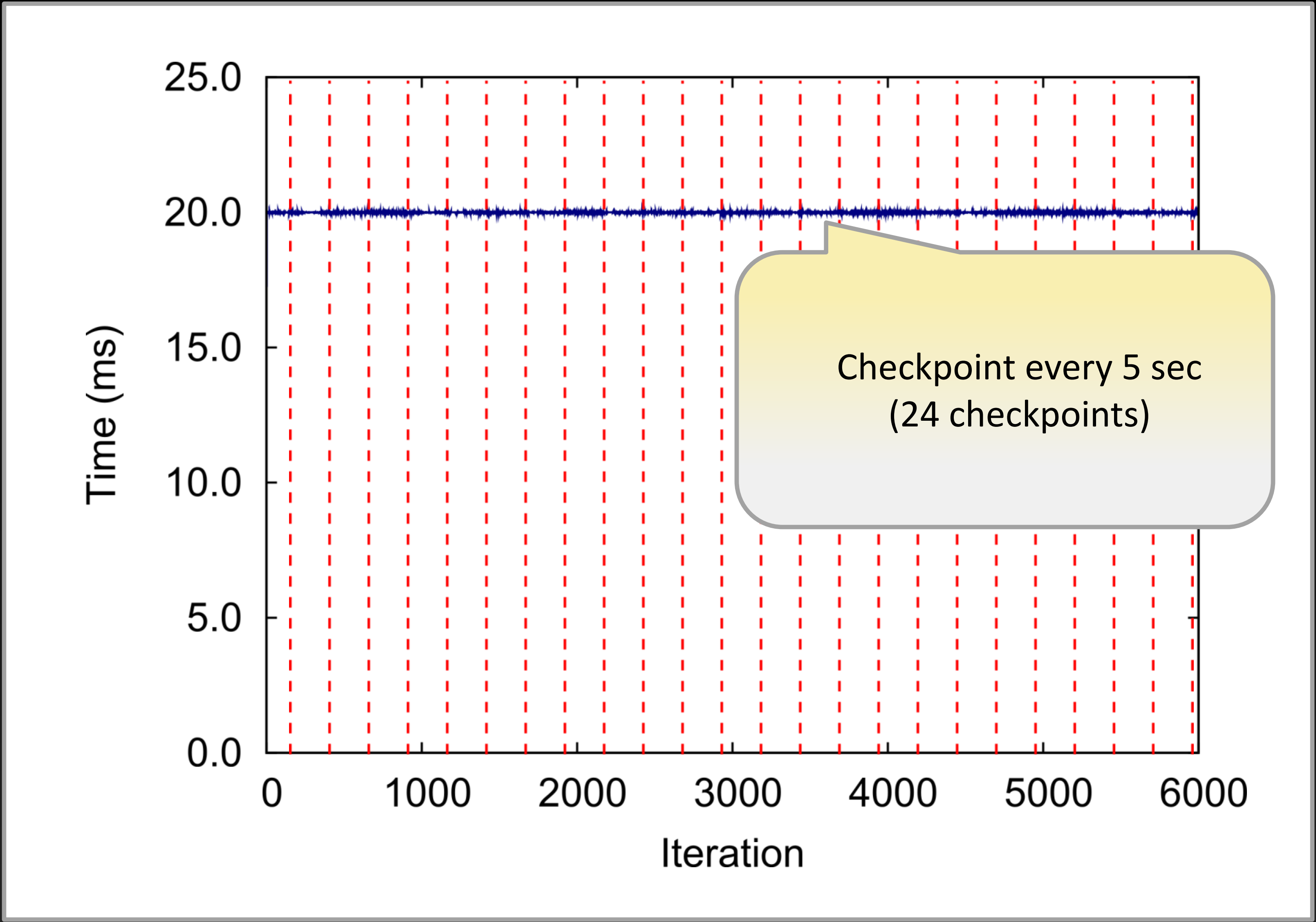
## Time virtualization



```
do {
    usleep(10 ms)
    gettimeofday()
} while ()
```

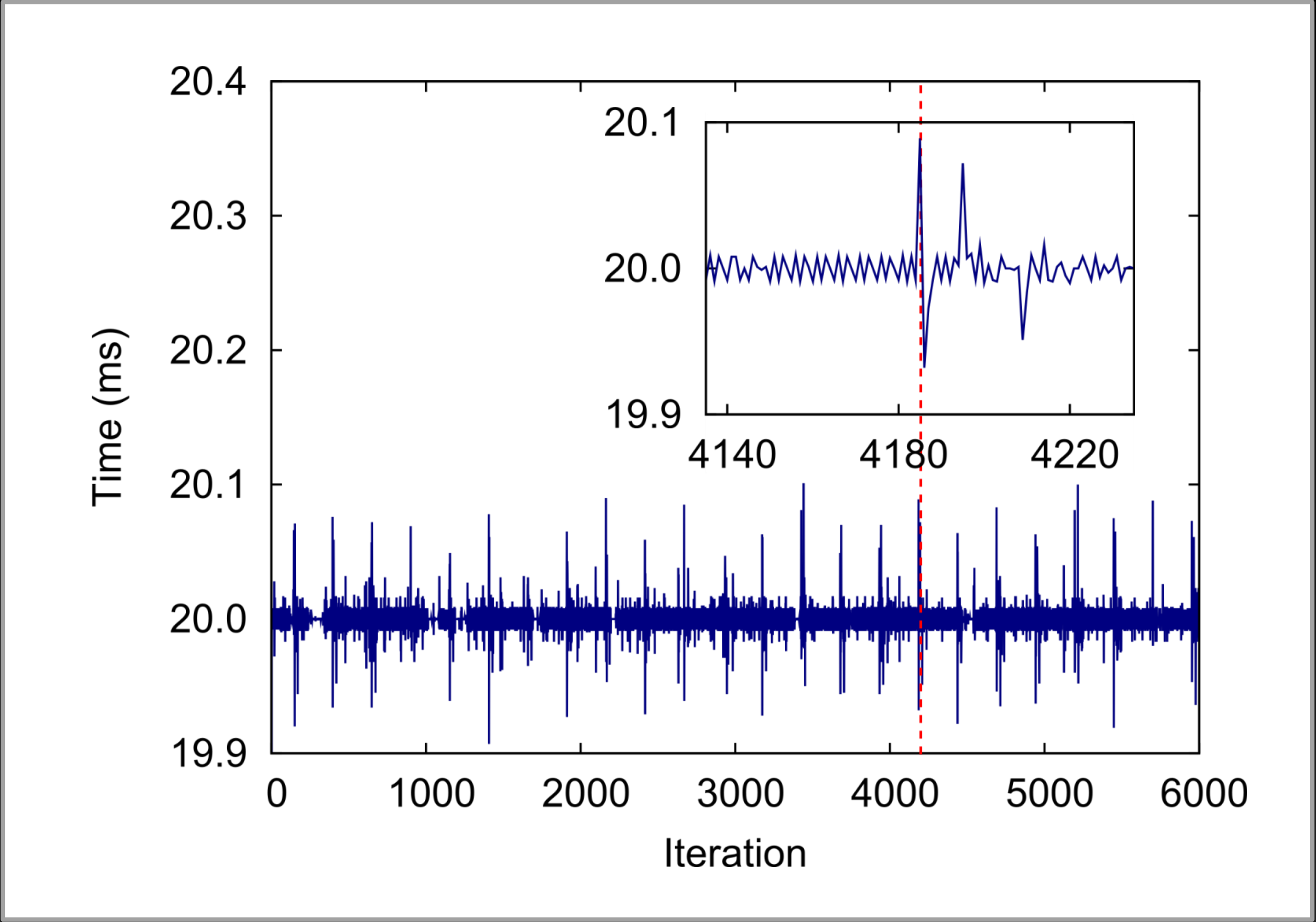
sleep + overhead = 20 ms

# Time virtualization

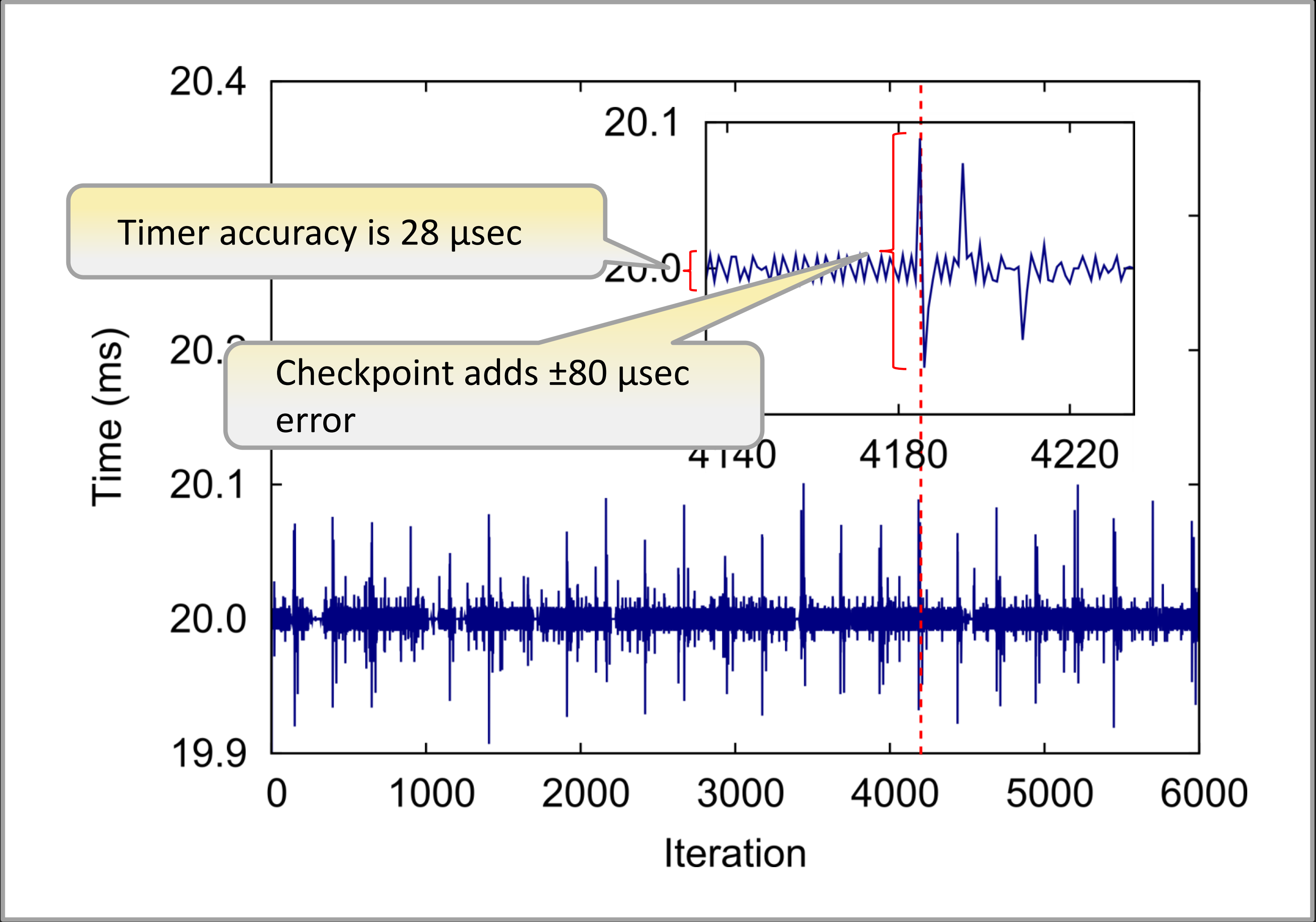


Checkpoint every 5 sec
(24 checkpoints)

# Time virtualization



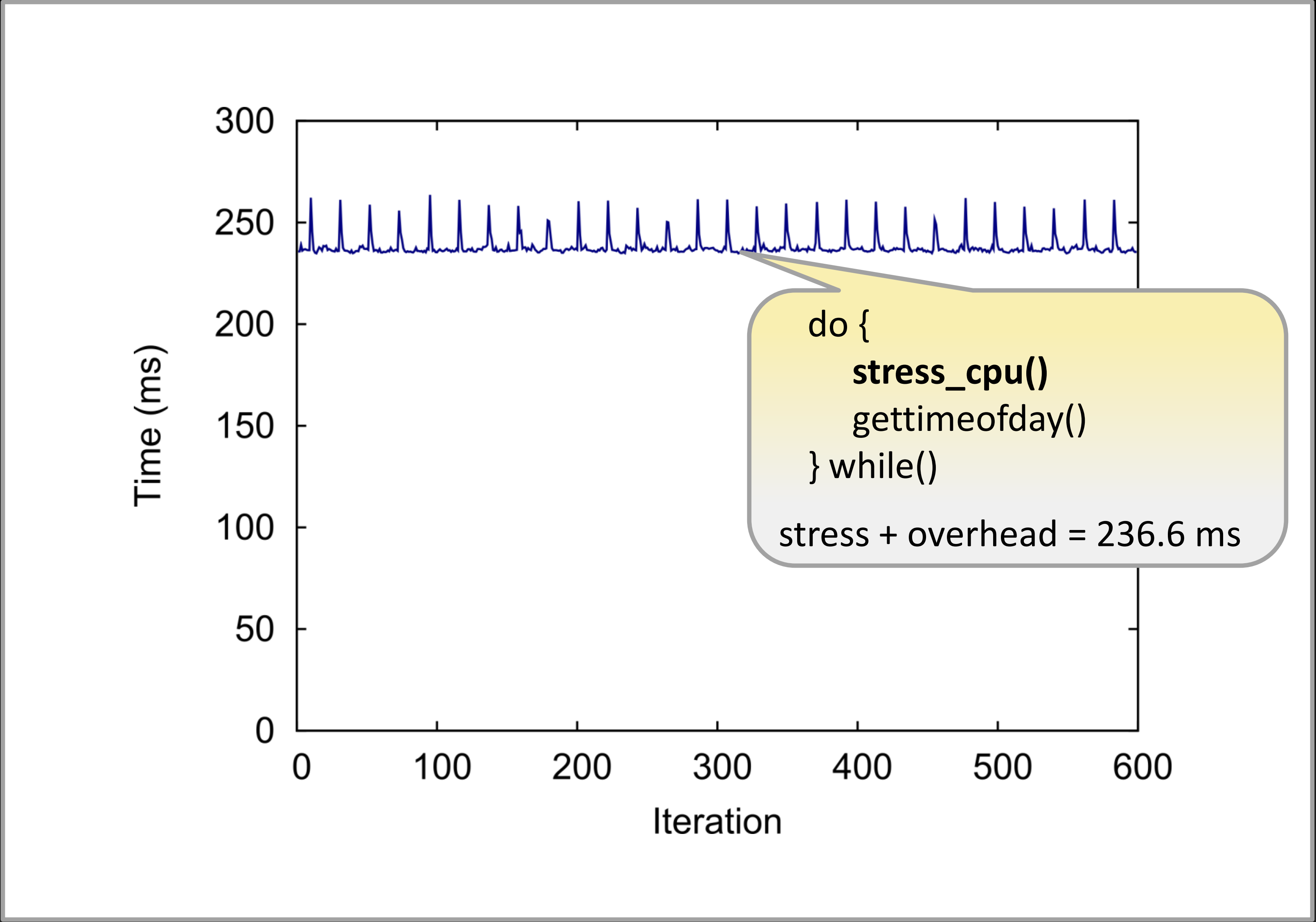
# Time virtualization



Timer accuracy is 28 µsec

Checkpoint adds ±80 µsec error

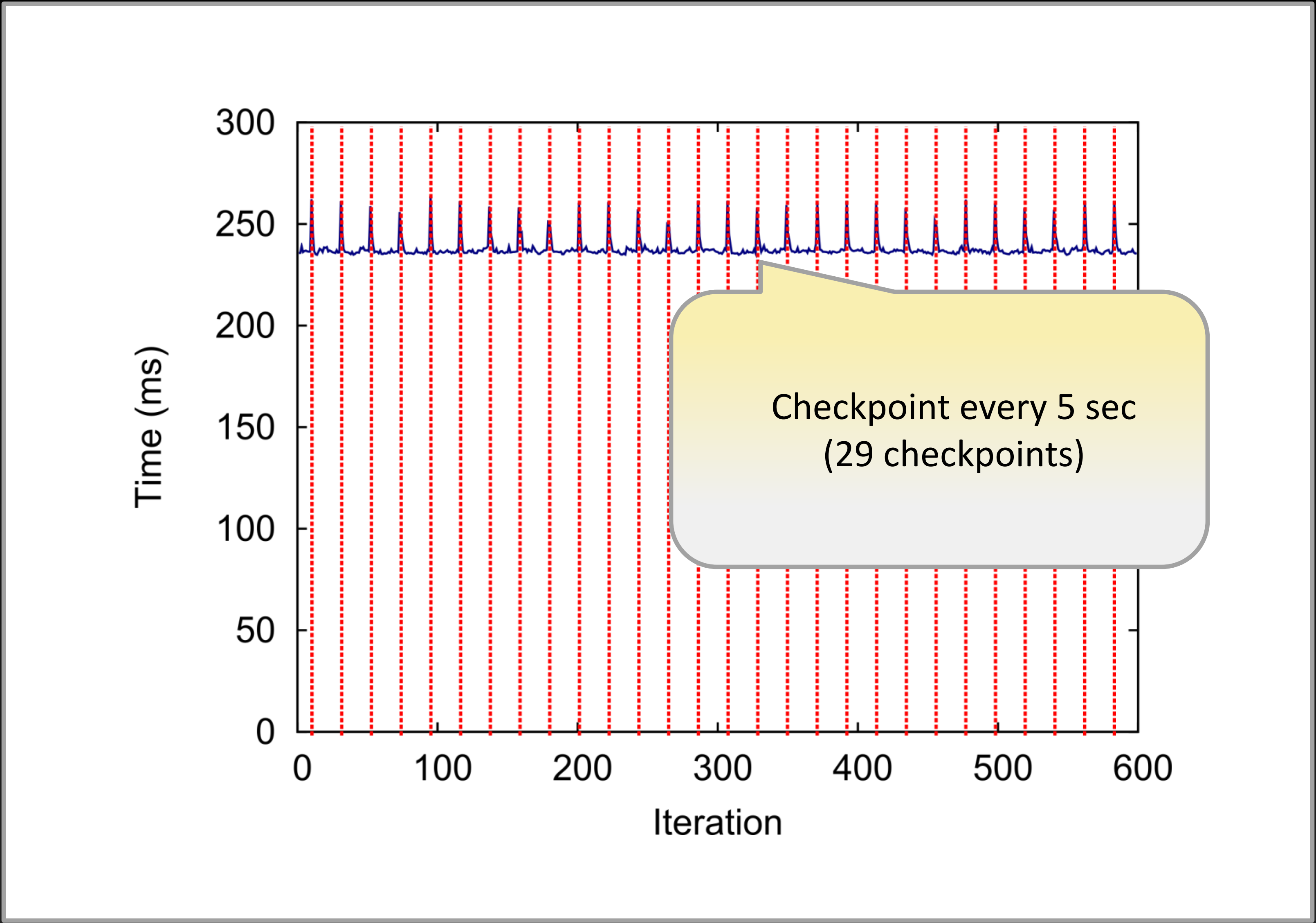
# CPU allocation



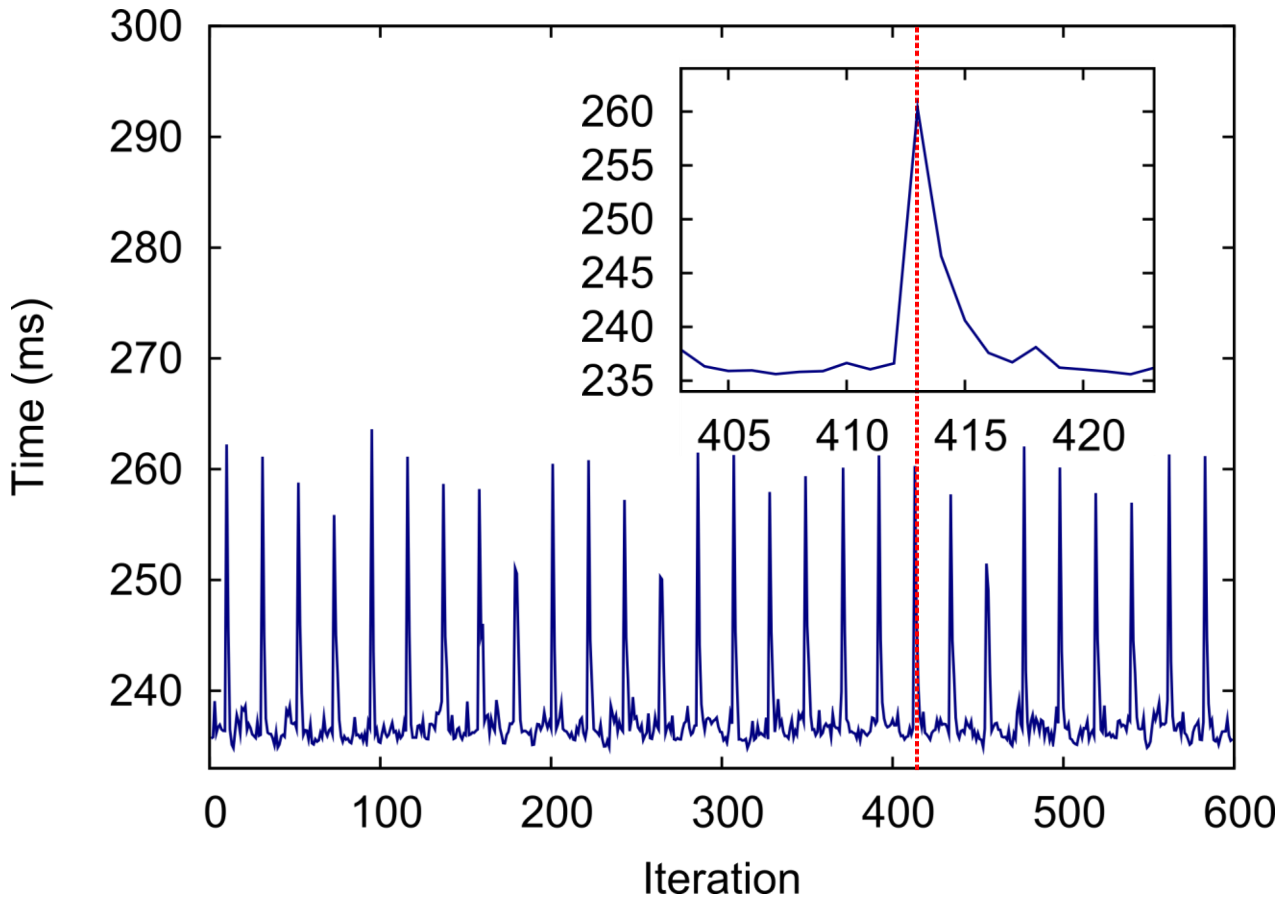
```
do {
    stress_cpu()
    gettimeofday()
} while()
```

stress + overhead = 236.6 ms

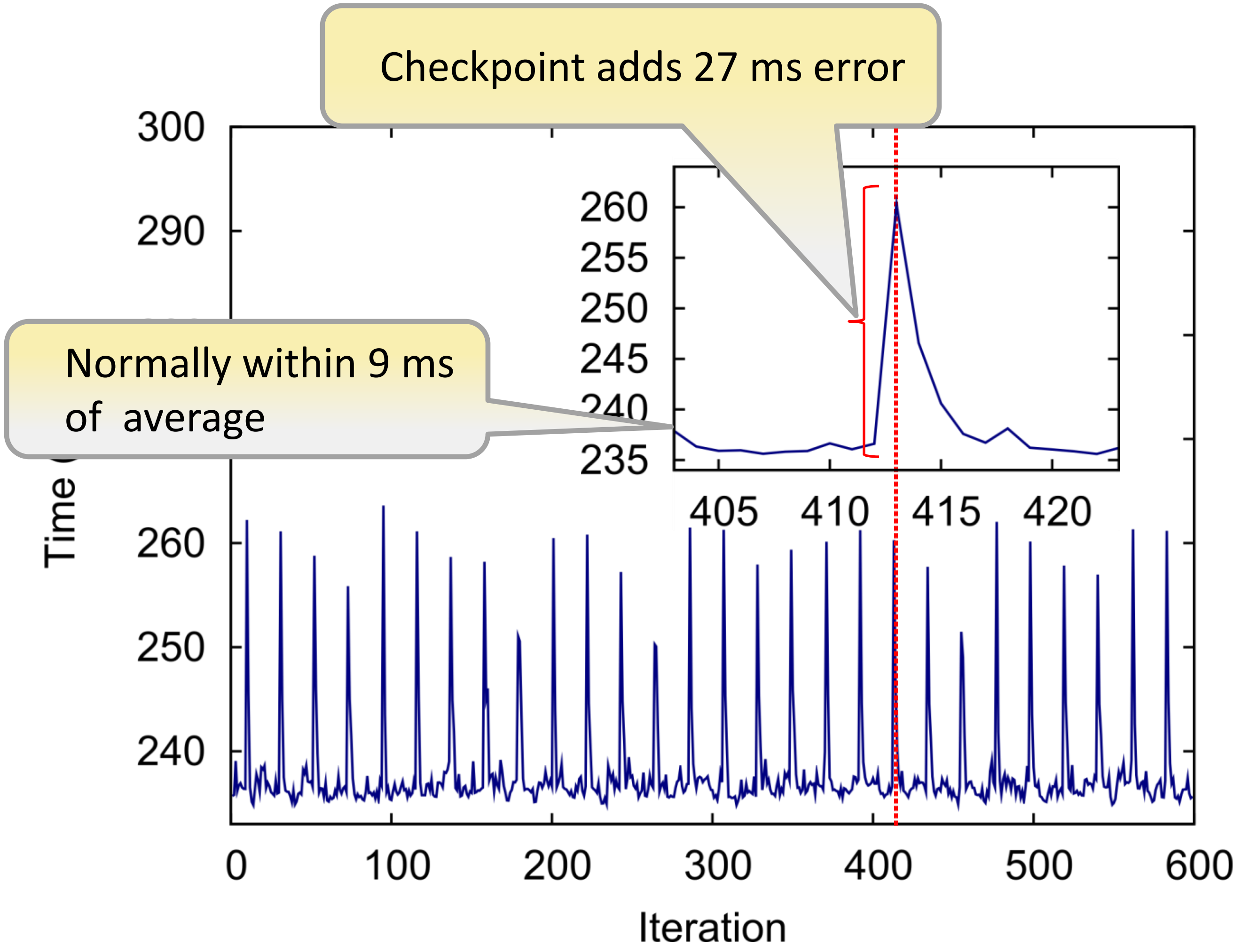
# CPU allocation



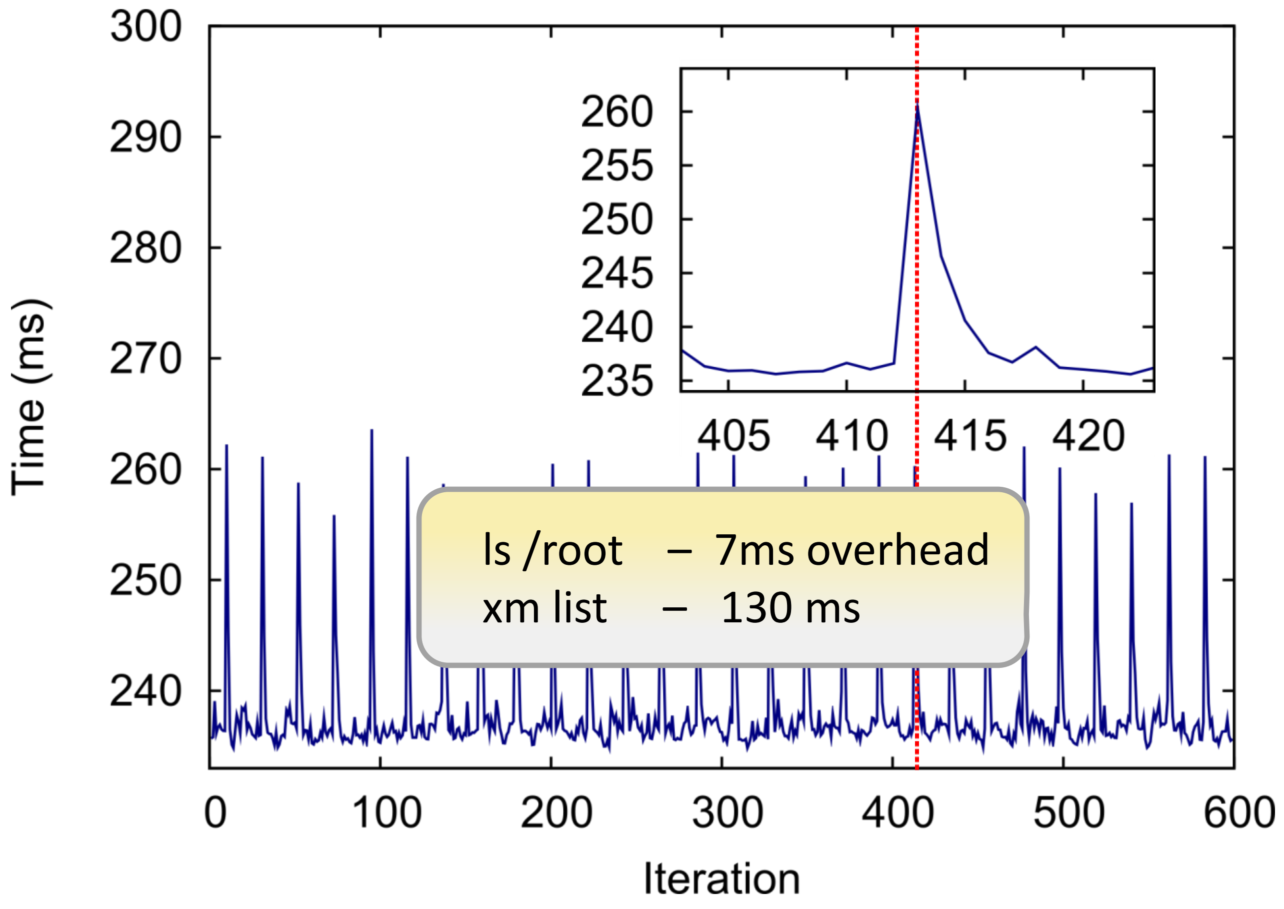
# CPU allocation



# CPU allocation

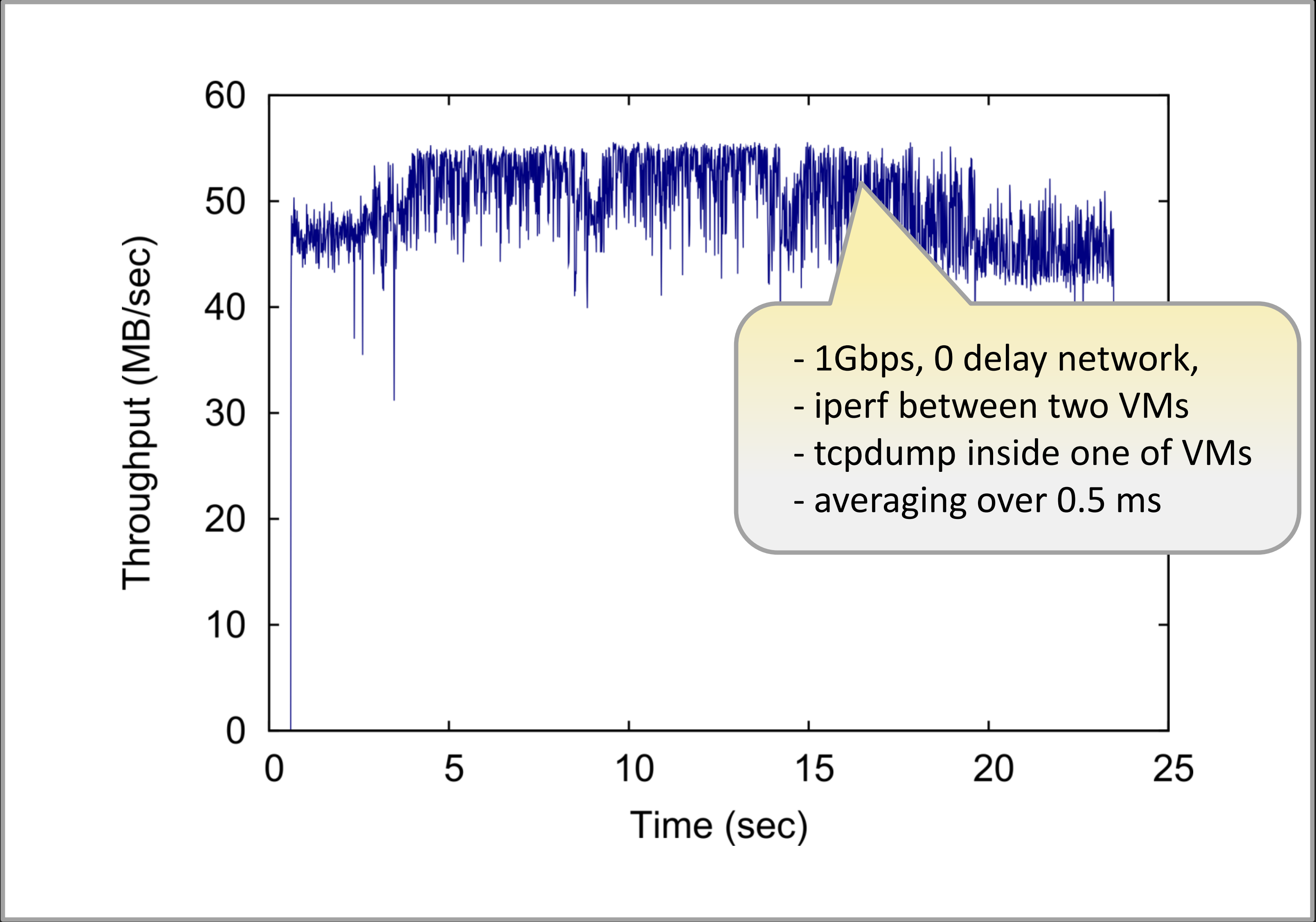


# CPU allocation



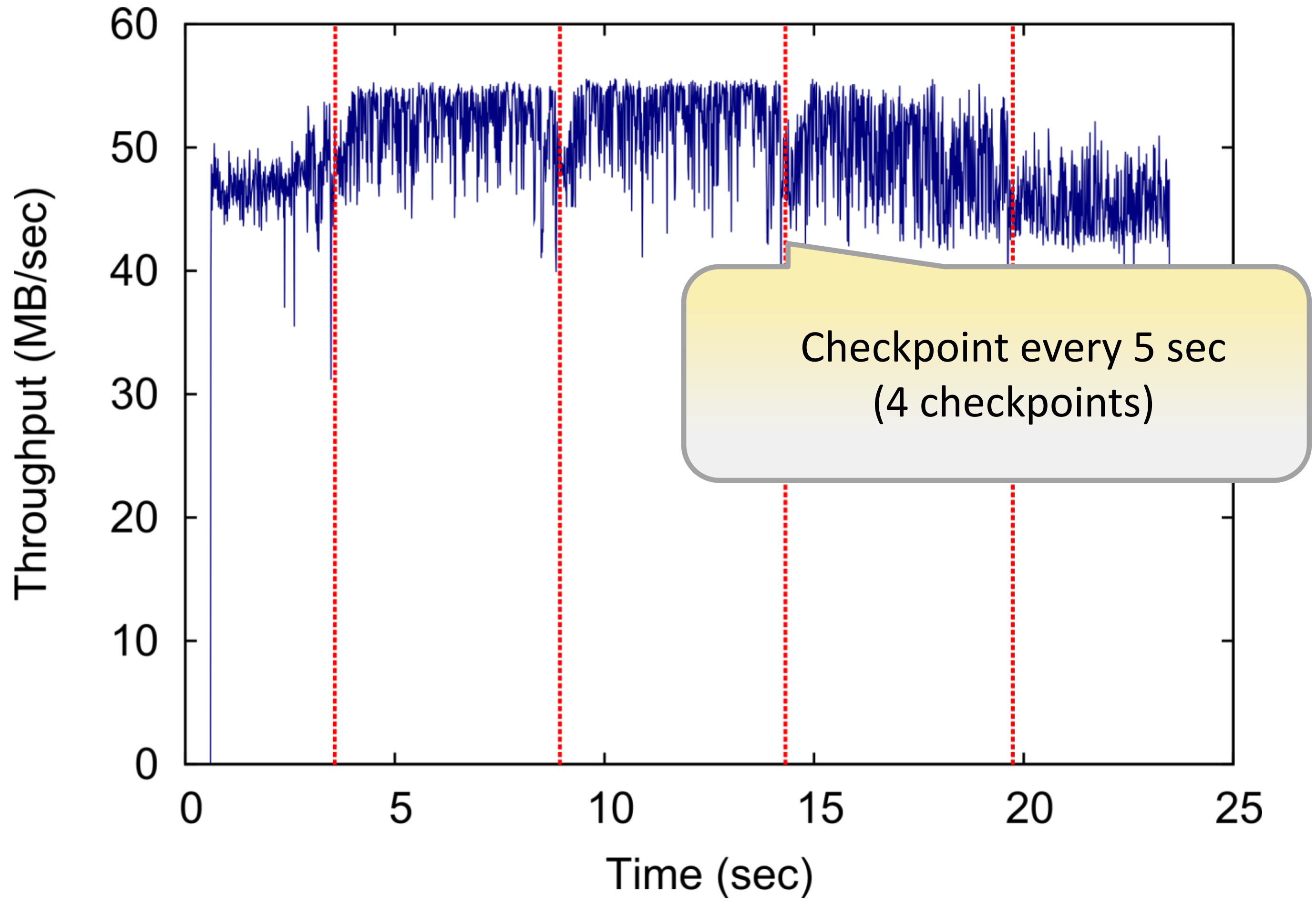
ls /root – 7ms overhead
xm list – 130 ms

# Network transparency: iperf



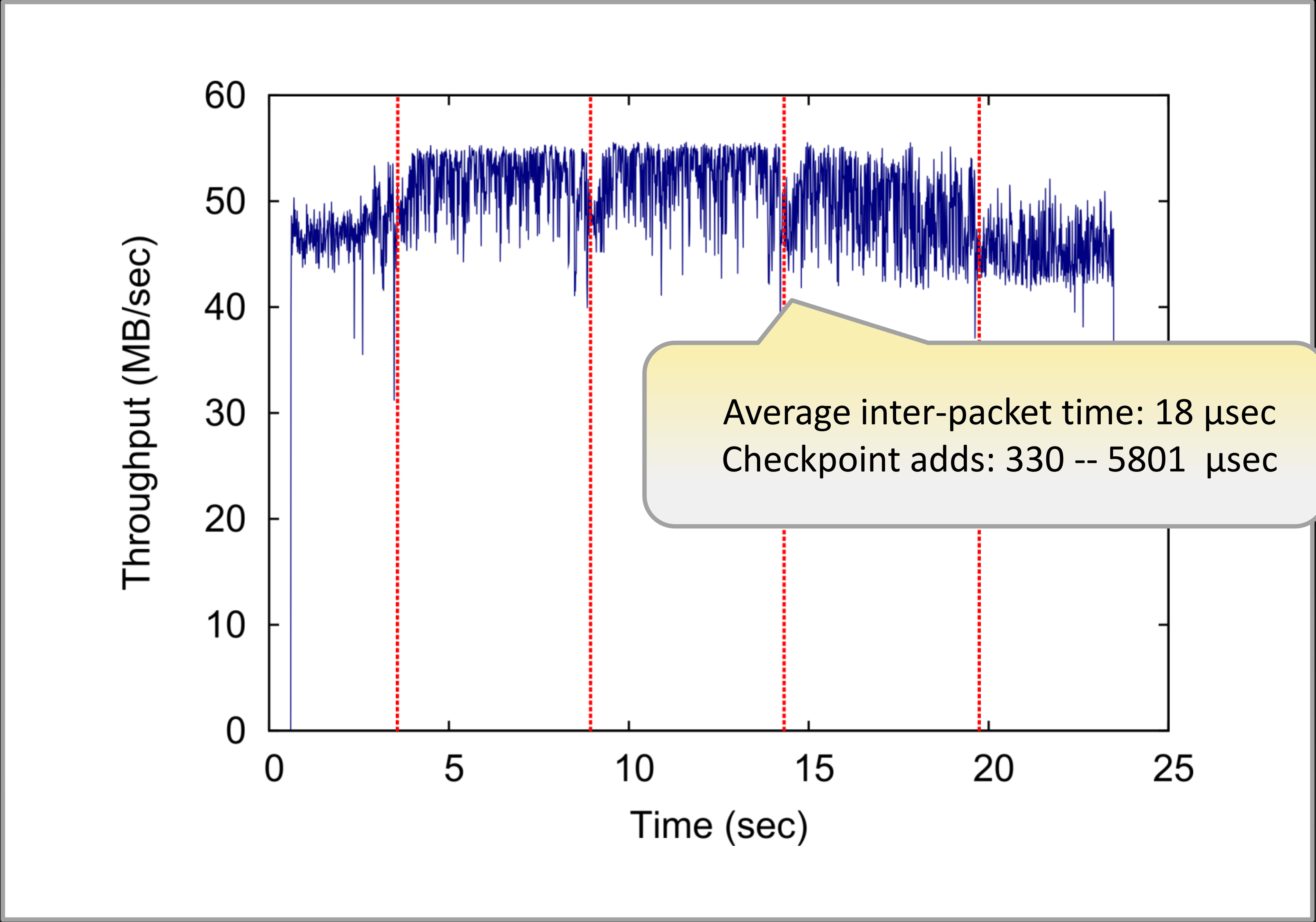
- 1Gbps, 0 delay network,
- iperf between two VMs
- tcpdump inside one of VMs
- averaging over 0.5 ms

## Network transparency: iperf



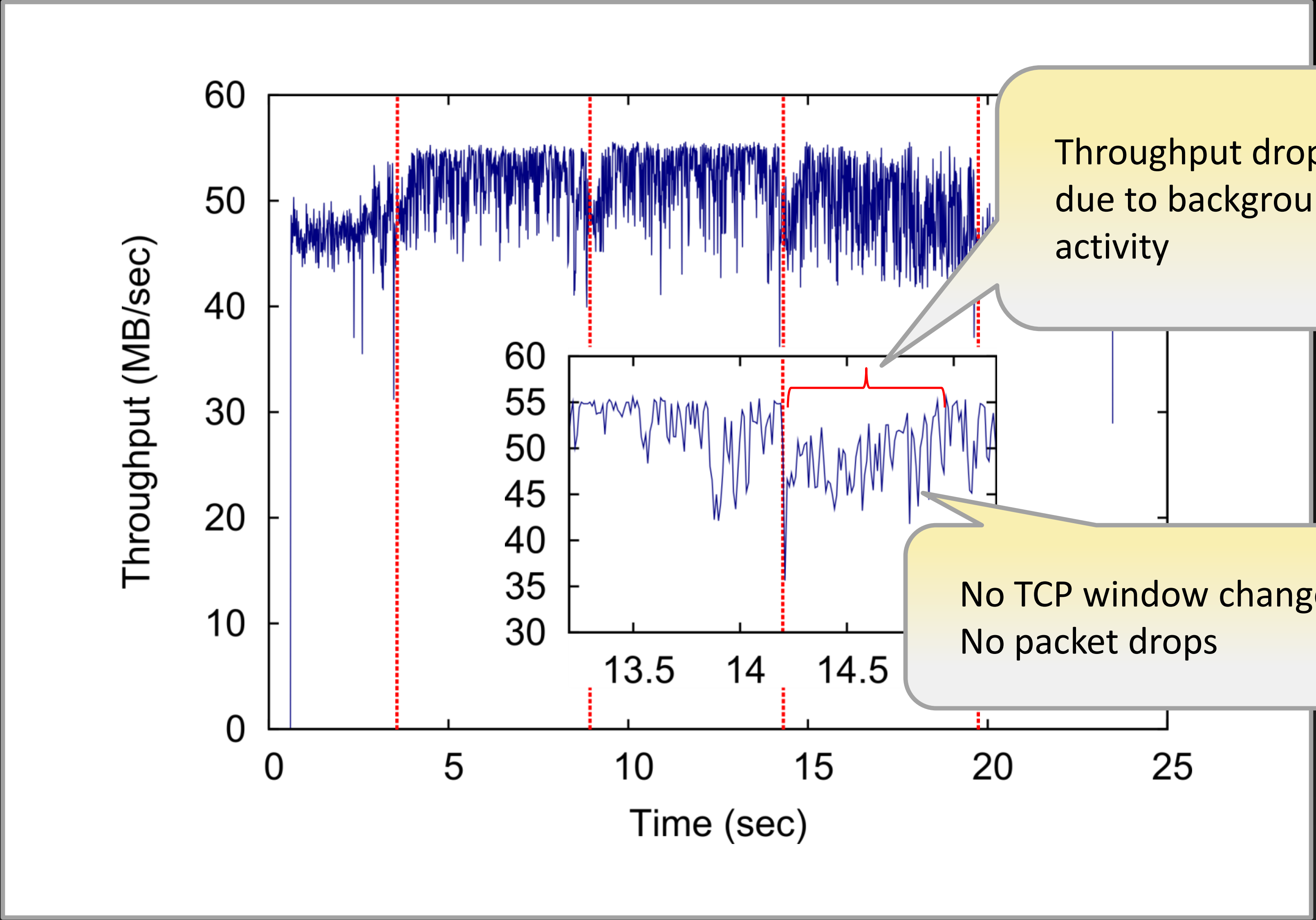
Checkpoint every 5 sec
(4 checkpoints)

# Network transparency: iperf



Average inter-packet time: 18 µsec
Checkpoint adds: 330 -- 5801 µsec

# Network transparency: iperf

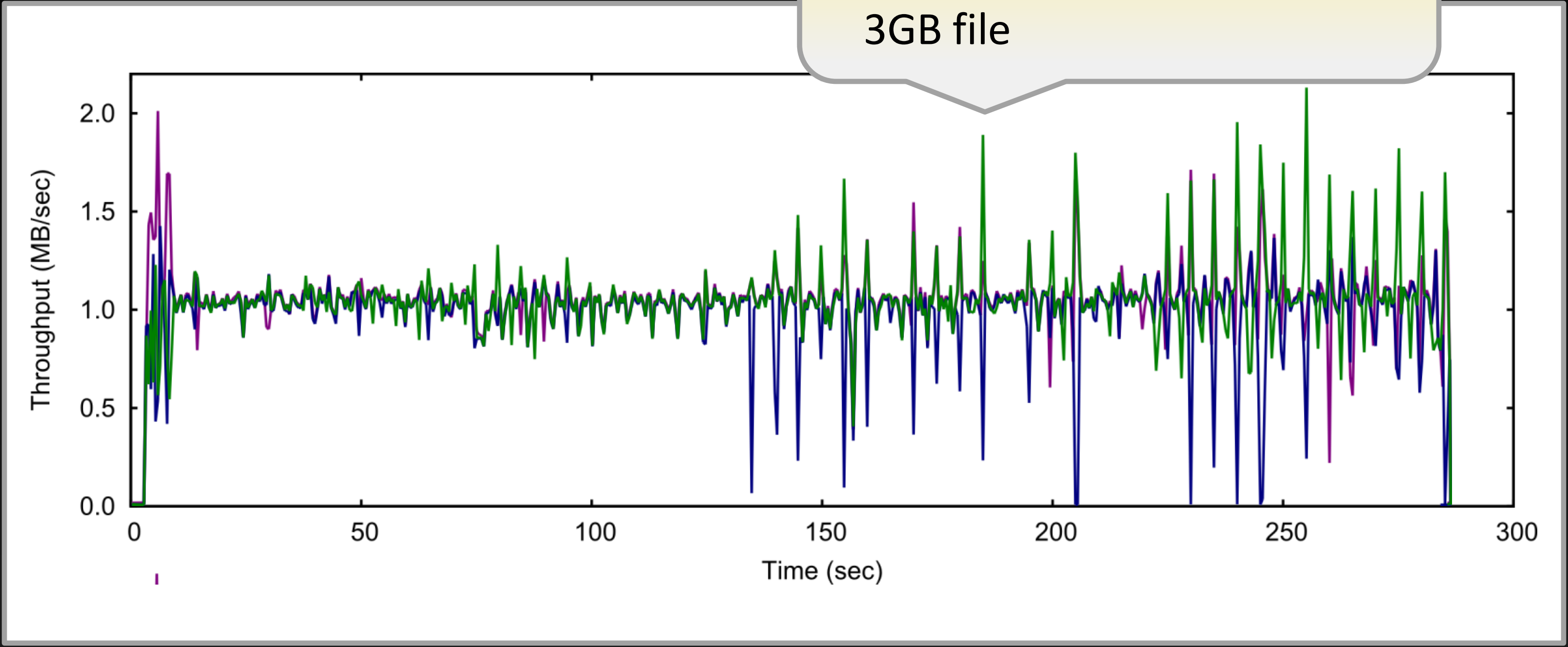


Throughput drop is due to background activity

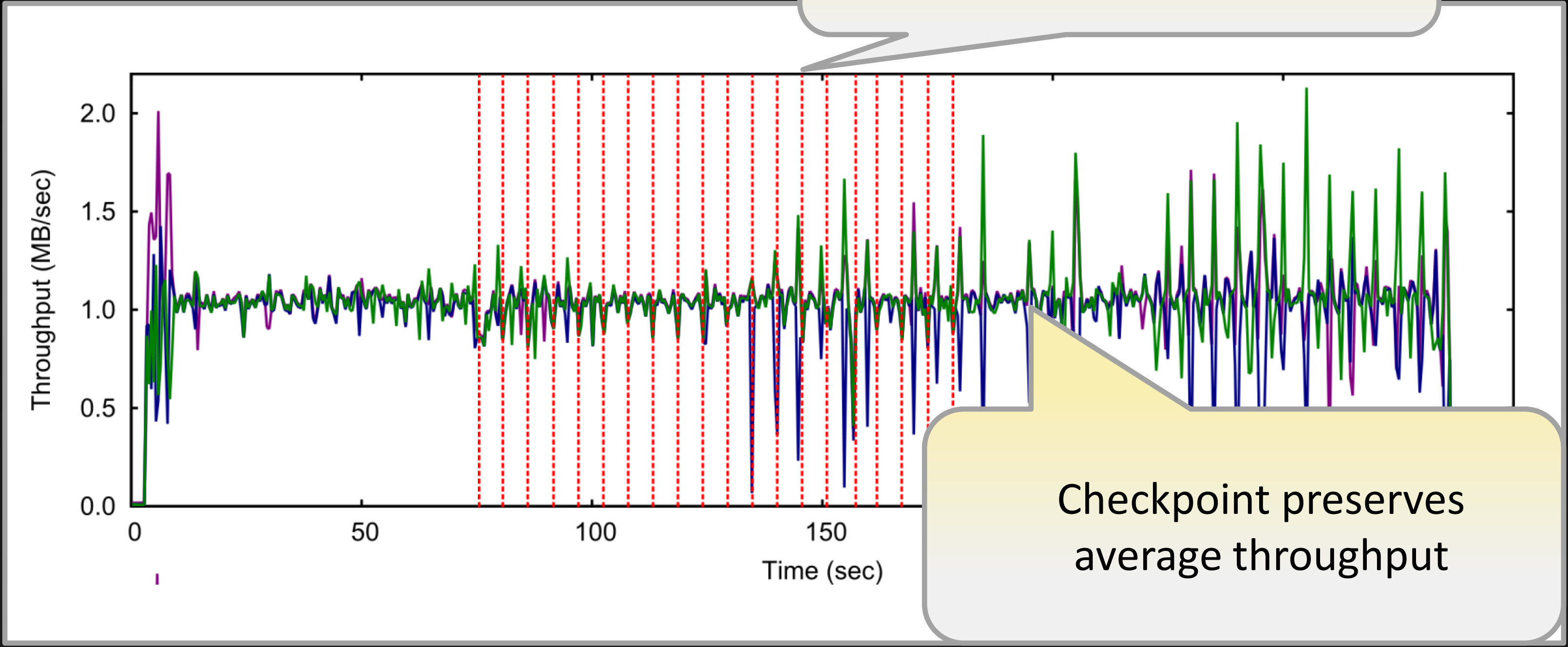
No TCP window change
No packet drops

# Network transparency: BitTorrent

100Mbps, low delay
1BT server + 3 clients
3GB file



## Network transparency: BitTorrent



Checkpoint every 5 sec (20 checkpoints)

Checkpoint preserves average throughput

## 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