Enabling Efficient Hypervisor-as-a-Service Clouds with Ephemeral Virtualization

Dan Williams†, Yaohui Hu‡, Umesh Deshpande*, Piush K Sinha‡, Nilton Bila†, Kartik Gopalan‡, Hani Jamjoom†

†IBM T.J. Watson Research Center
‡Binghamton University
*IBM Almaden Research Center

Funded in part by the NSF

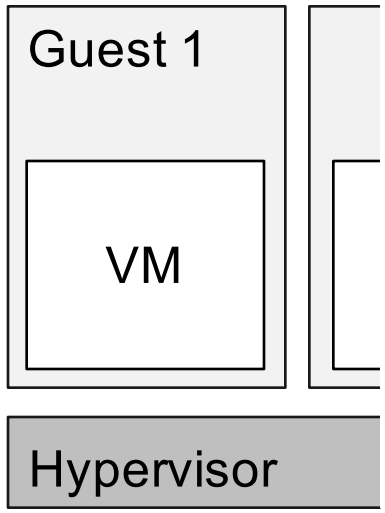
BINGHAMTON UNIVERSITY
State University of New York

IBM

Hypervisors…

- Thin and secure layer in the cloud

-- or --

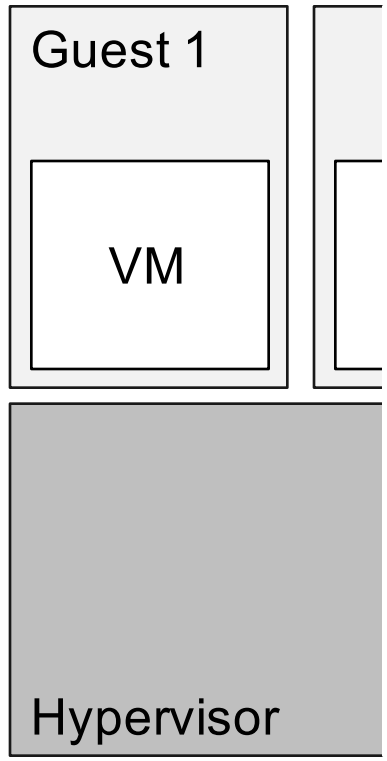


Hypervisors can do a lot!

- Thin and secure layer in the cloud

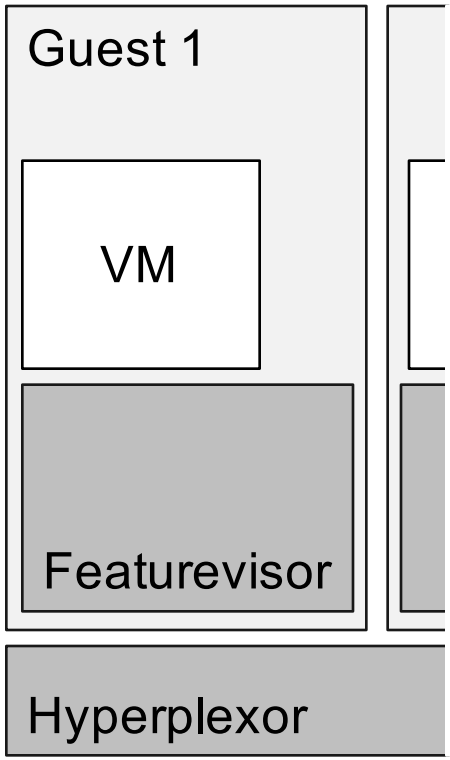
-- or --

- Feature-filled cloud differentiators
  - Rootkit detection
  - Live patching
  - Intrusion detection
  - High availability service
  - Other VMI…



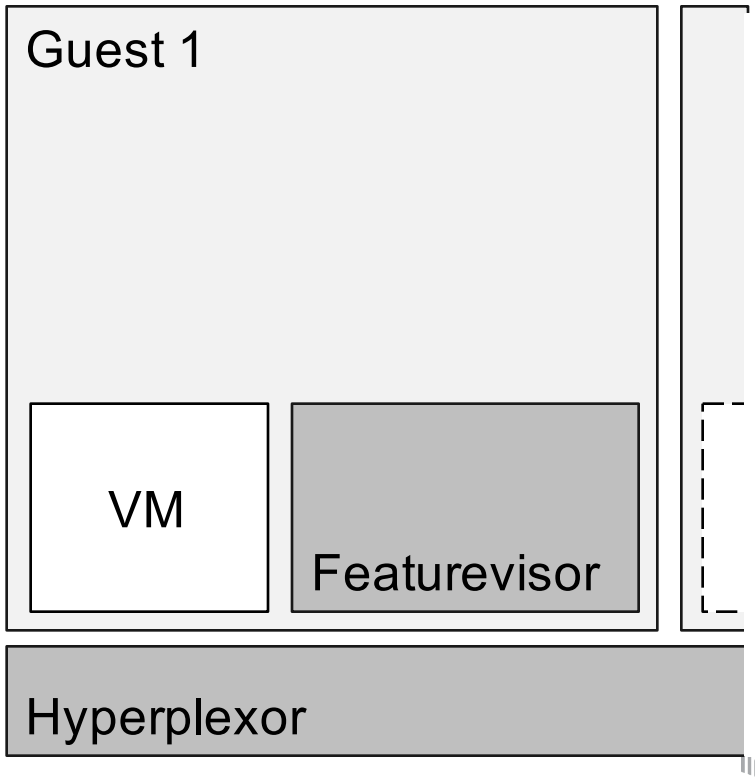
Splitting the hypervisor

- Cloud provider runs **hyperplexor**
  - Multiplex hardware
- Each guest selects a **featurevisor**
  - Implements rich services
- "Hypervisor-as-a-service"
- Can implement with nested virtualization
  - Pay nesting overhead all of the time



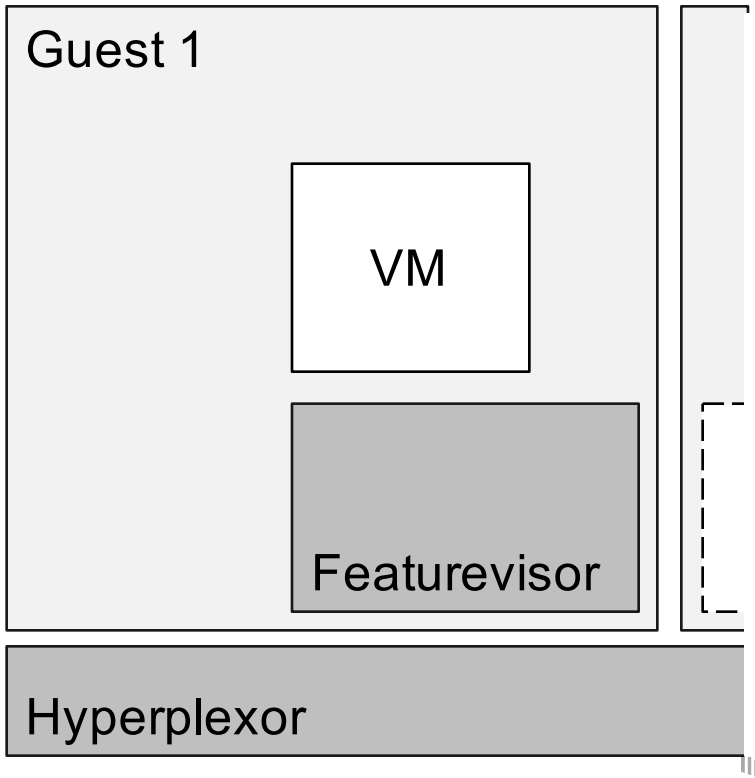
Ephemeral virtualization

- VM runs on **hyperplexor** when not needing featurevisor services



Ephemeral virtualization

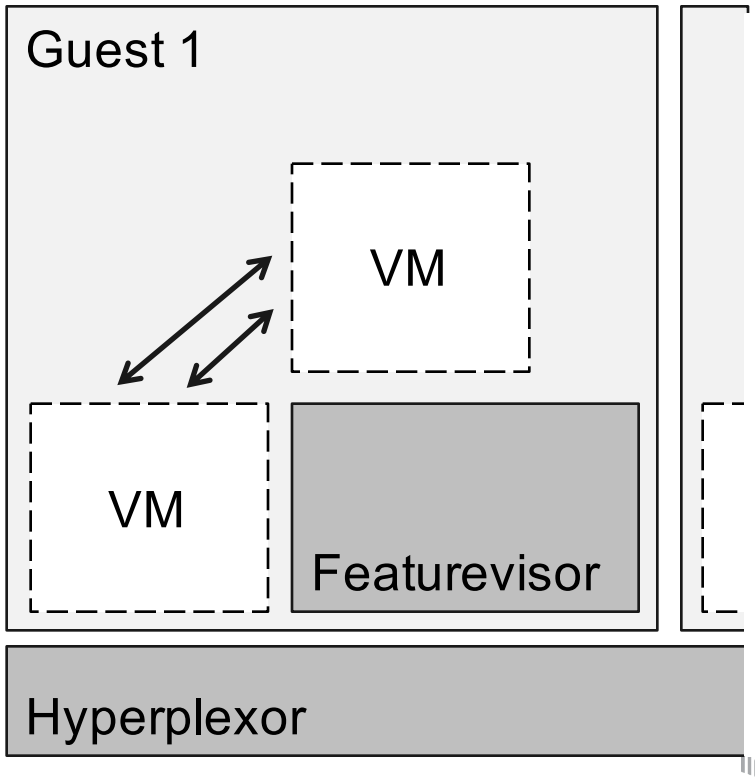
- VM runs on **hyperplexor** when not needing featurevisor services
- VM runs on **featurevisor** when needing its services



Ephemeral virtualization

- VM runs on **hyperplexor** when not needing featurevisor services
- VM runs on **featurevisor** when needing its services

- Implemented "**Dichotomy**"
  - 80ms switching times
  - Up to 12% better performance than nested with 2.5s switching interval

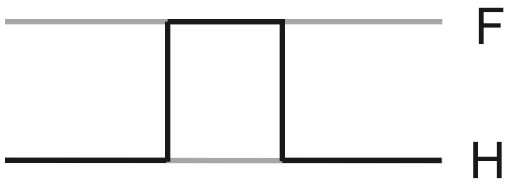


Roadmap

- When does this make sense?
- Design of Dichotomy
- Future directions
- Evaluation
- Related work
- Conclusions

Hypervisor-level applications

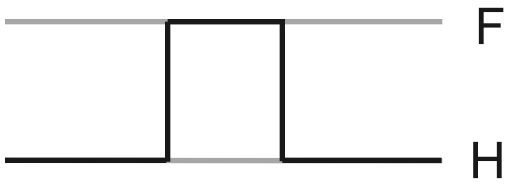
- One-time
  - Snapshot, guest patching, VM mgmt …



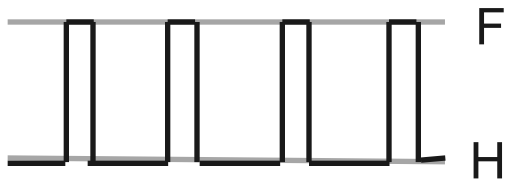
time

Hypervisor-level applications

- One-time
  - Snapshot, guest patching, VM mgmt …

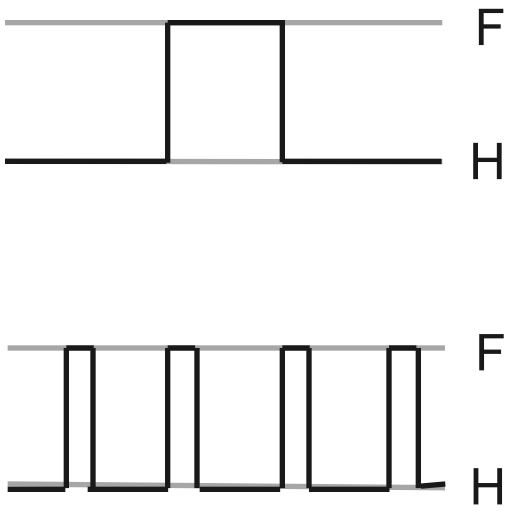


- Sample-based
  - Rootkit detection, logging, VMI …



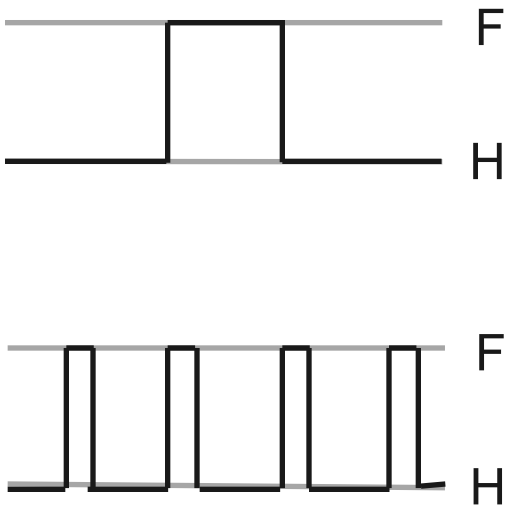
Hypervisor-level applications

- One-time
  - Snapshot, guest patching, VM mgmt …
- Sample-based
  - Rootkit detection, logging, VMI …
- Continuous
  - HA snapshot (Remus), memory dedup …

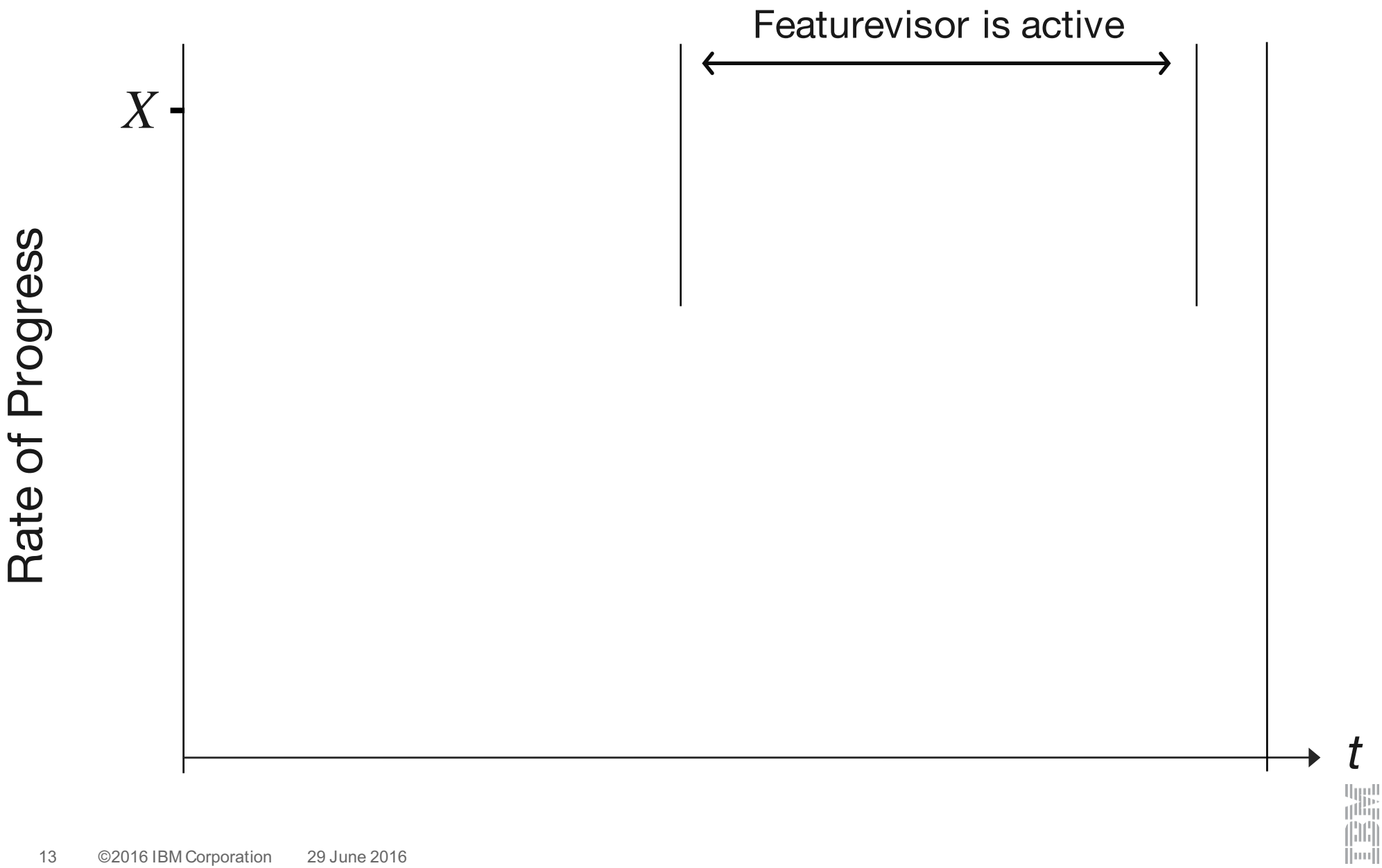


Hypervisor-level applications

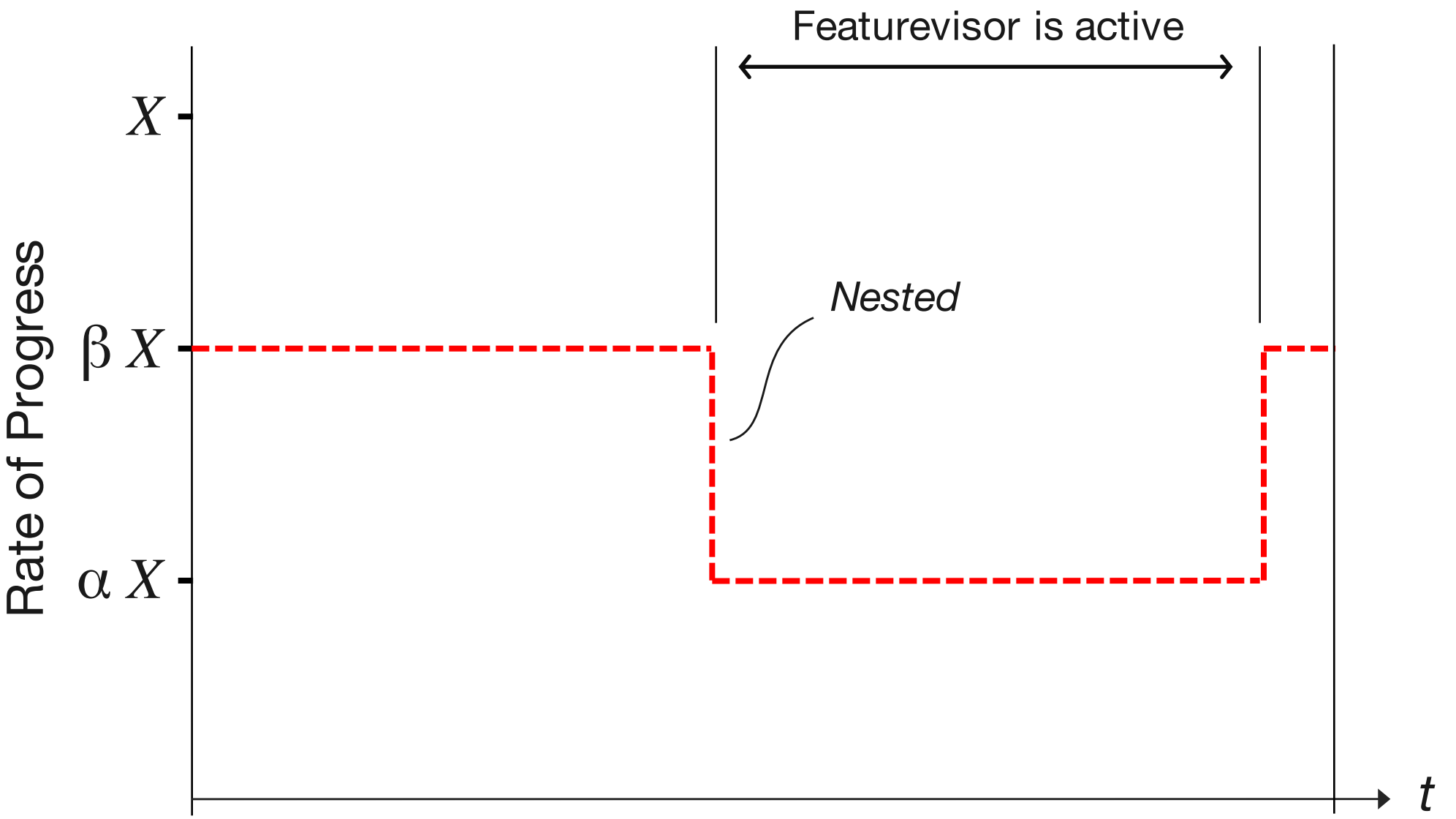
- One-time
  - Snapshot, guest patching, VM mgmt …
- Sample-based
  - Rootkit detection, logging, VMI …
- Continuous
  - HA snapshot (Remus), memory dedup …



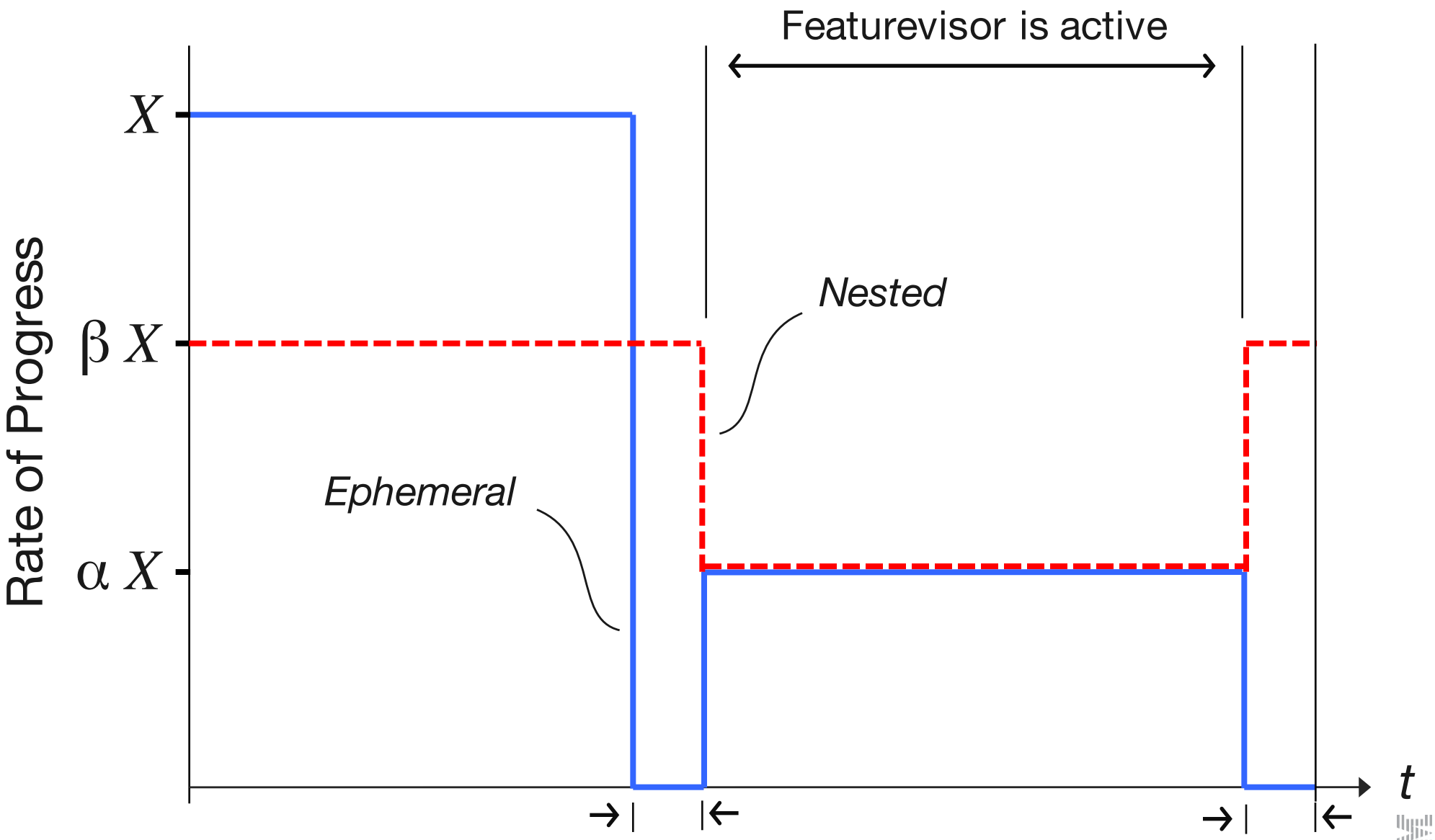
Progress during one cycle



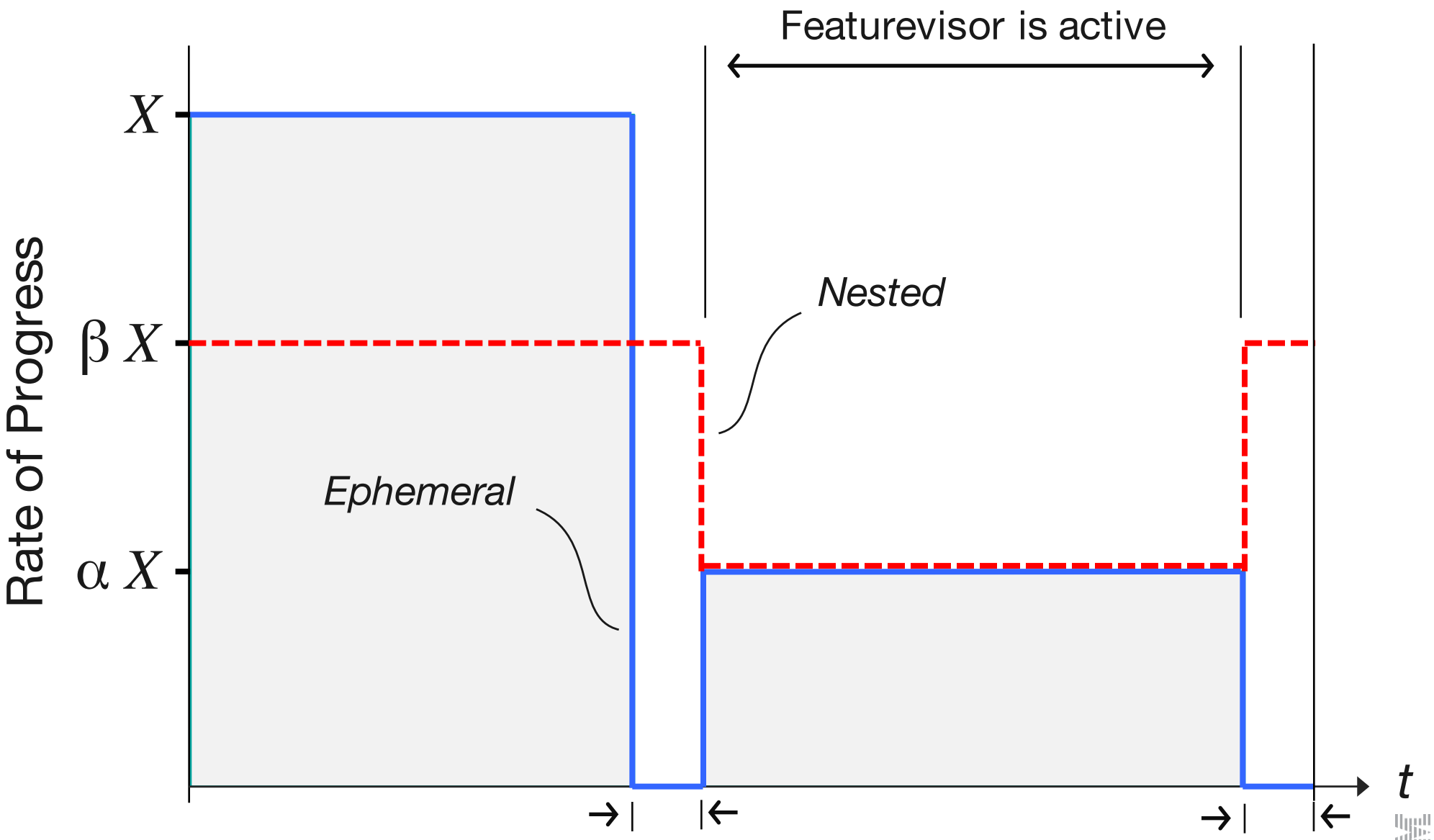
Progress during one cycle



Progress during one cycle



Progress during one cycle



Factors affecting performance

- Overhead of nesting
- Amount of time on hyperplexor vs featurevisor
- Frequency of switching

- Overhead of switching

Factors affecting performance

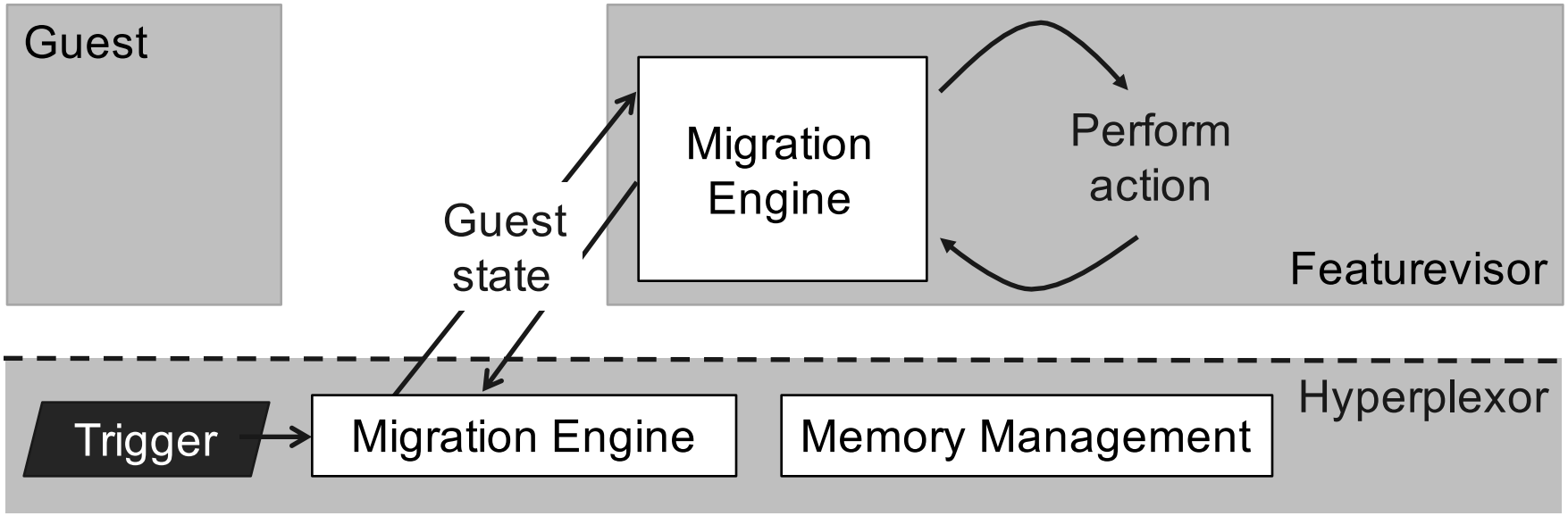
- Overhead of nesting
- Amount of time on hyperplexor vs featurevisor
- Frequency of switching

Dependent on featurevisor and workload

- Overhead of switching
  - Design goal: minimal switching overhead

Dichotomy

- Memory management
- Switching

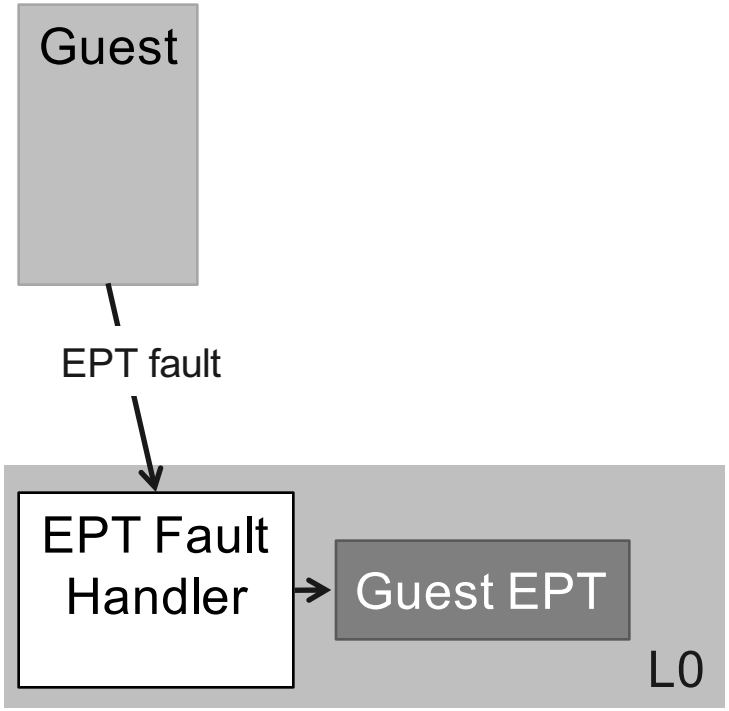


Memory management

- Non-nested
- Nested
- Dichotomy

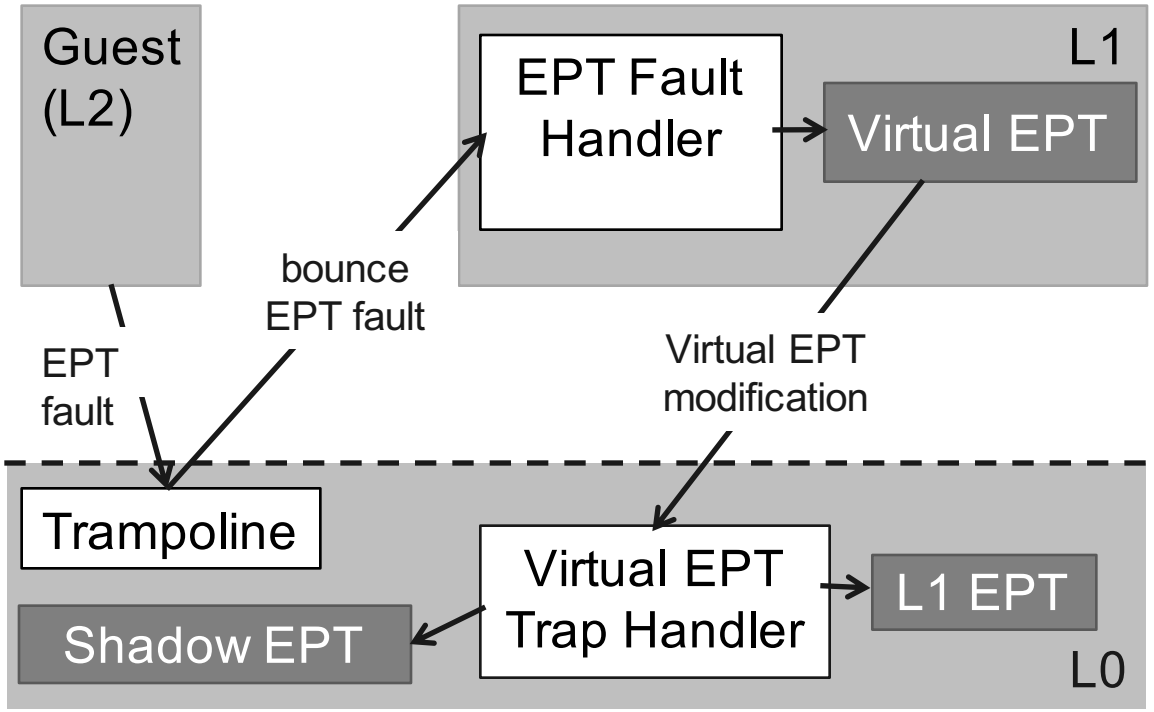
Memory management: non-nested

- Hypervisor manages mapping between guest-physical and machine-physical pages in the **Guest EPT**
- EPT faults may cause hypervisor to update mapping

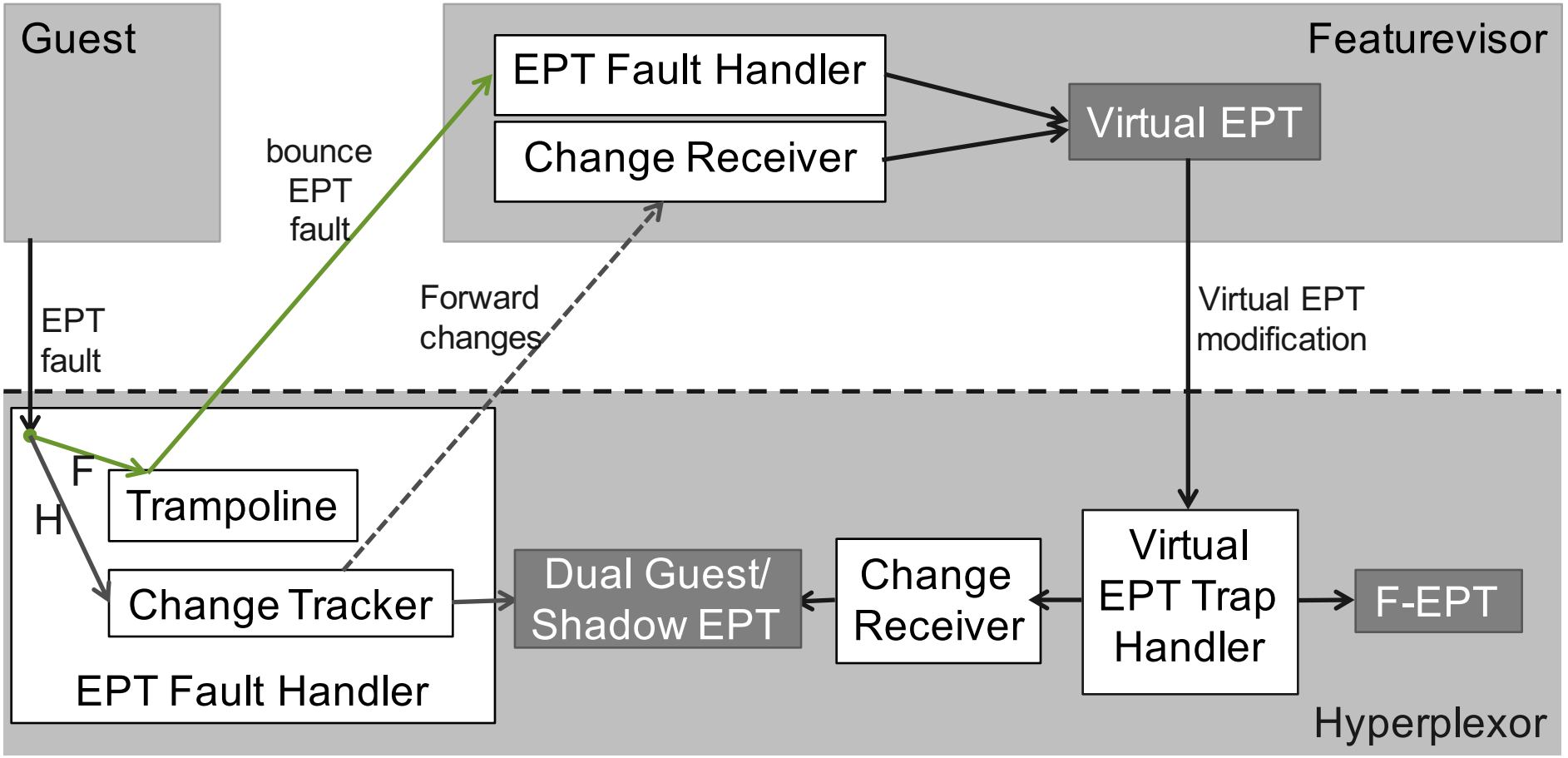


Memory management: nested

- Guest pages mapped into L1 hypervisor
- L1 manages mappings via a **virtual EPT** for the guest
- L0 backs virtual EPT with **shadow EPT**



Memory management: Dichotomy



Switching

- Similar to VM migration
  - Pause VM
  - Transfer VCPU, I/O, unsynchronized page table mappings
  - No copying of memory

```
forever:
    execute guest while
    waiting for trigger

on_trigger:
    relinquish_guest
    wait_for_guest
```

Hyperplexor

```
forever:
    wait_for_guest
    do action
    relinquish guest
    finish action
```

Featurevisor

Triggers

- Time-based
- On new network flow?
- On particular memory access?
- On program counter value?

- What's the right set of triggers?

Future work

- Multiple featurevisors for one guest
  - How do featurevisors interact?
- Multiple guests on one featurevisor
  - E.g., temporary high-speed memory channel
- Common hyperplexor services
  - Dirty page tracking service?

- What is the right featurevisor interface?

Evaluation

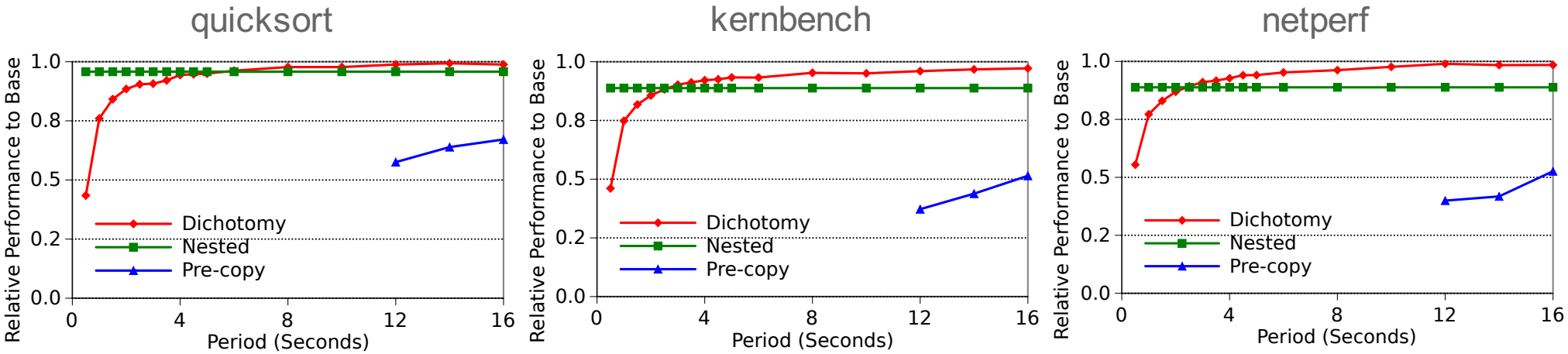
- Implementation based on KVM/QEMU
  - Time-based triggers

- When is Dichotomy better than nested?
- How small are switching times?

Evaluation: setup

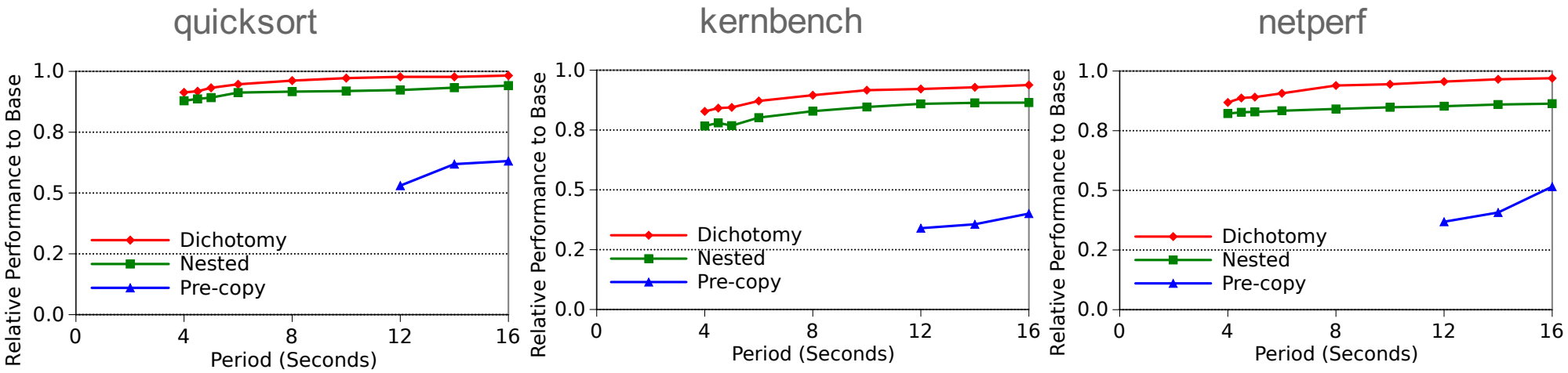
- Featurevisor applications
  - No-op
  - Volatility (VMI rootkit detection)
  - Netmon (1 second tcpdump)
- Guest workloads
  - Quicksort 800 MB
  - Kernbench
  - Netperf

Workload performance: no-op featurevisor



- Dichotomy outperforms nesting if period is sufficiently long (2.5—6 seconds)
- Pre-copy suffers because of slow switching times

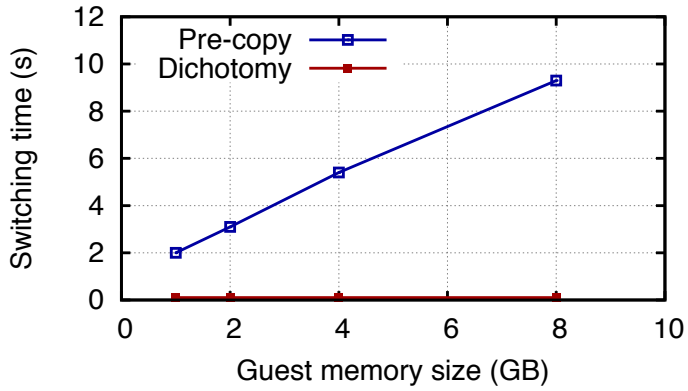
Workload performance: volatility featurevisor



- Service time of volatility is sufficiently long (4s)
  - Dichotomy always outperforms nesting

How fast is switching time?

- 80ms independent of memory size
- Compared to ~300ms downtime with live migration



Related Work

- On-demand switching to emulation
  - Taint-based protection (Ho et al., Eurosys 2006)
- Switching between bare metal and virtualization
  - On-demand virtualization (Kooburat and Swift, HotOS 2011)
  - Mercury (Chen et al., ICPP 2007)
- Reducing hypervisor to its essentials
  - Microvisor (Lowell et al., ASPLOS 2004)
  - CloudVisor (Zhang et al., SOSP 2011)
- Disaggregating administrative domain
  - Breaking up is hard to do (Colp et al., SOSP 2011)

Conclusion

- Cloud providers shouldn't need to chose between small and feature-filled hypervisors
- Dichotomy: split the role of the hypervisor into **hyperplexor** and **featurevisor**
- Switch between with **ephemeral virtualization**
  - 80ms switching times
  - Up to 12% better performance than nested with 2.5s switching interval