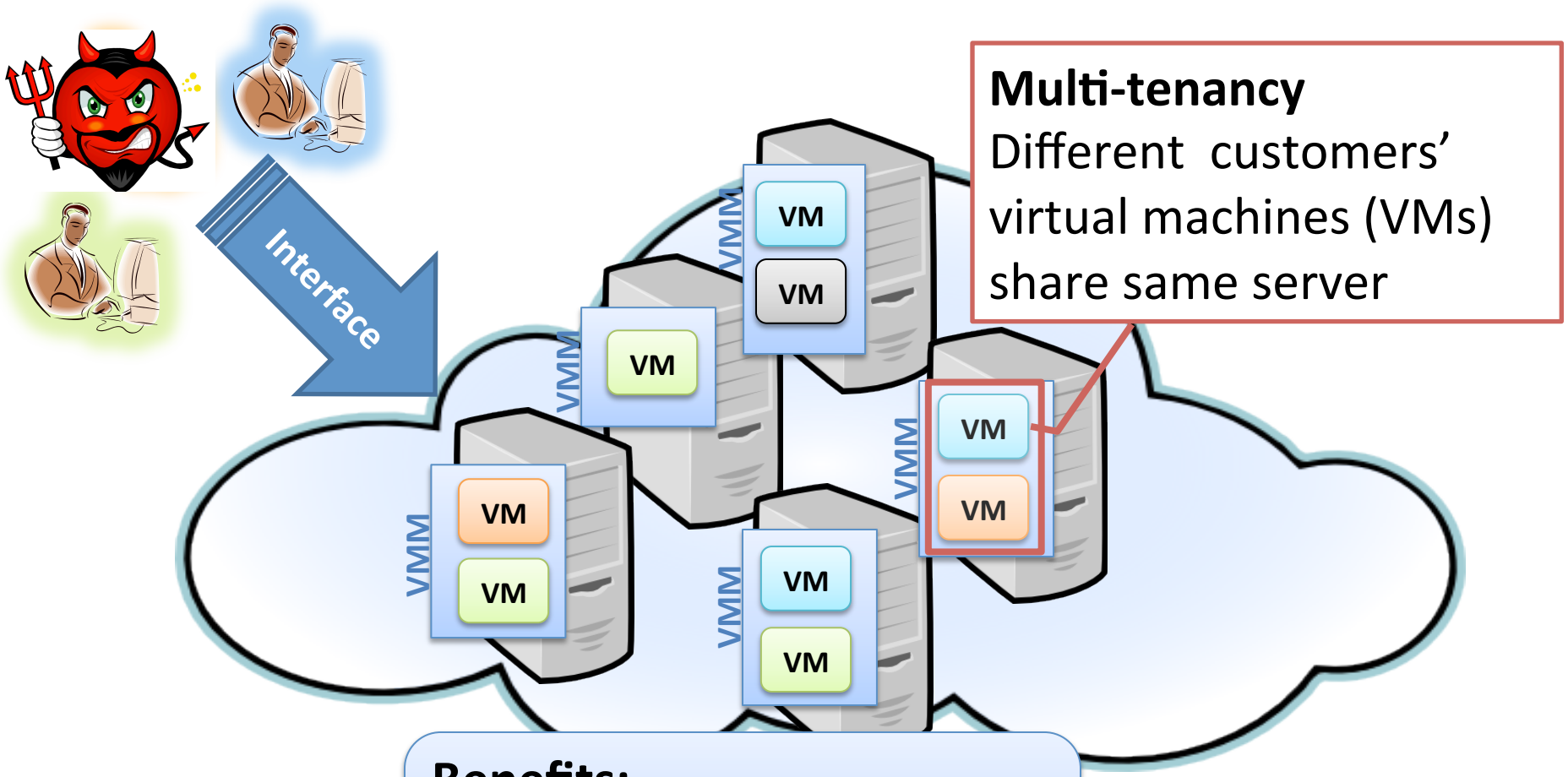
Scheduler-based Defenses against Cross-VM Side-channels

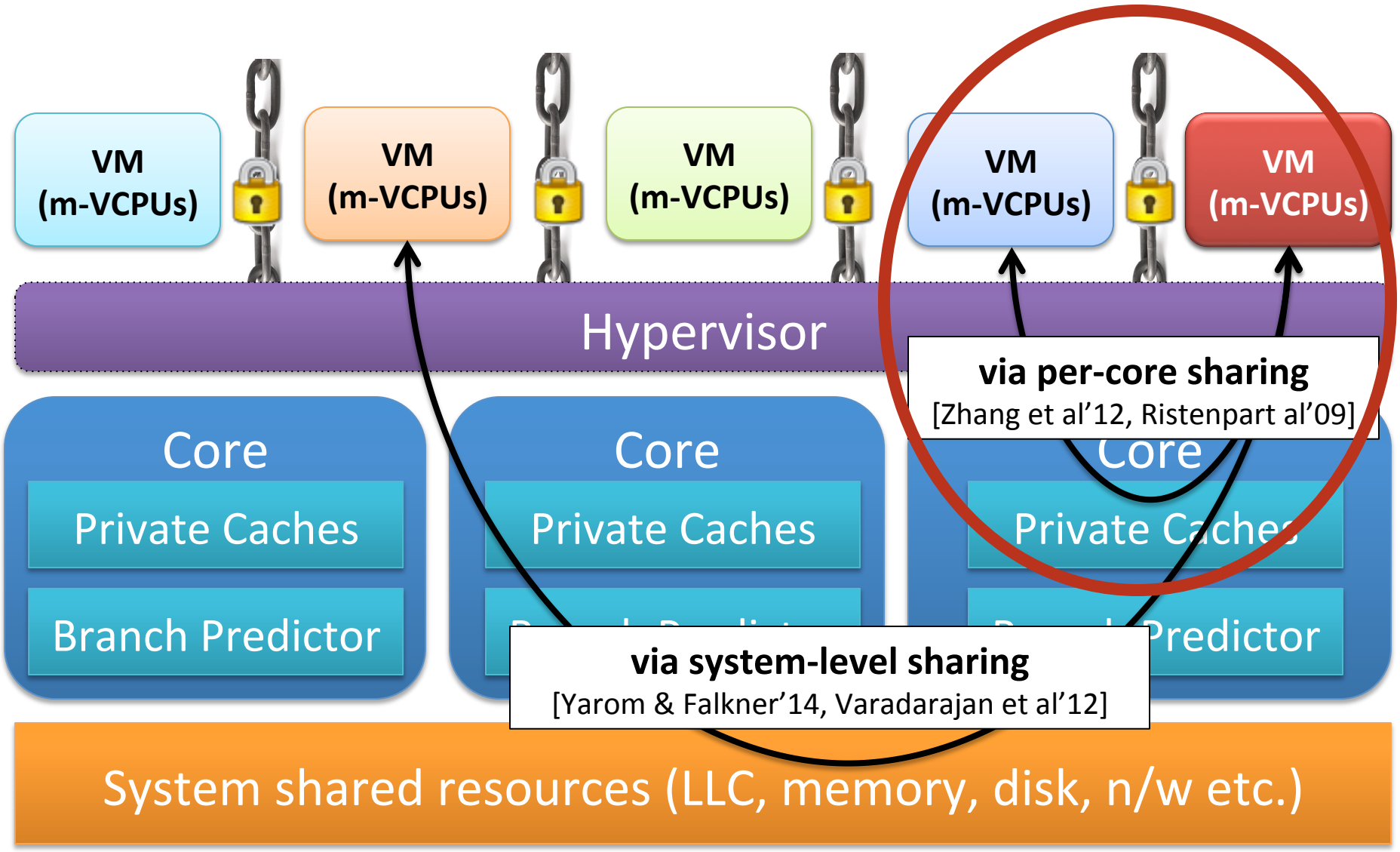
Venkat(anathan) Varadarajan,
Thomas Ristenpart,
and Michael Swift

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON
DEPARTMENT OF COMPUTER SCIENCES

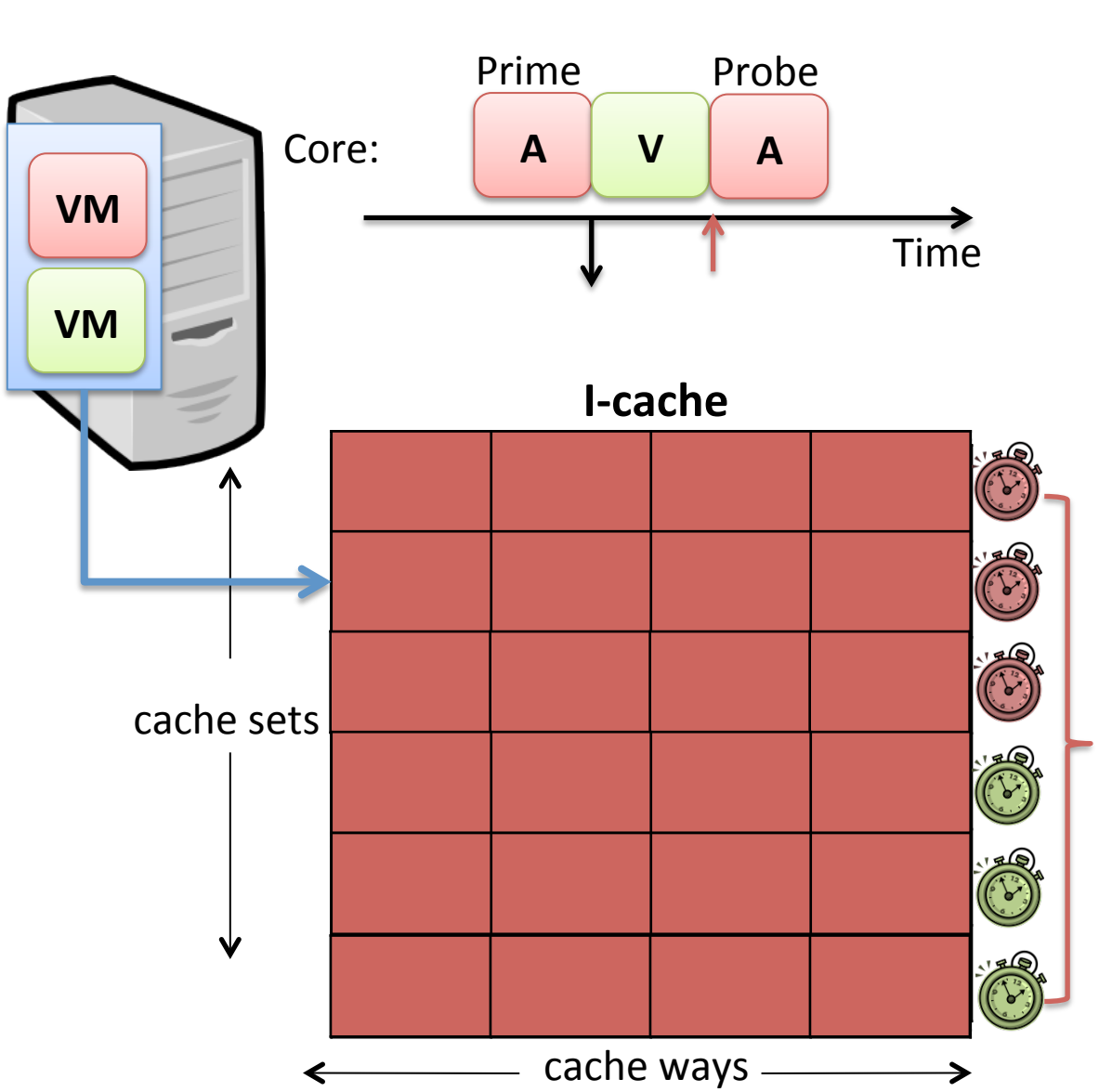
Public Clouds (EC2, Azure, Rackspace, …)



Shared Resources and Isolation

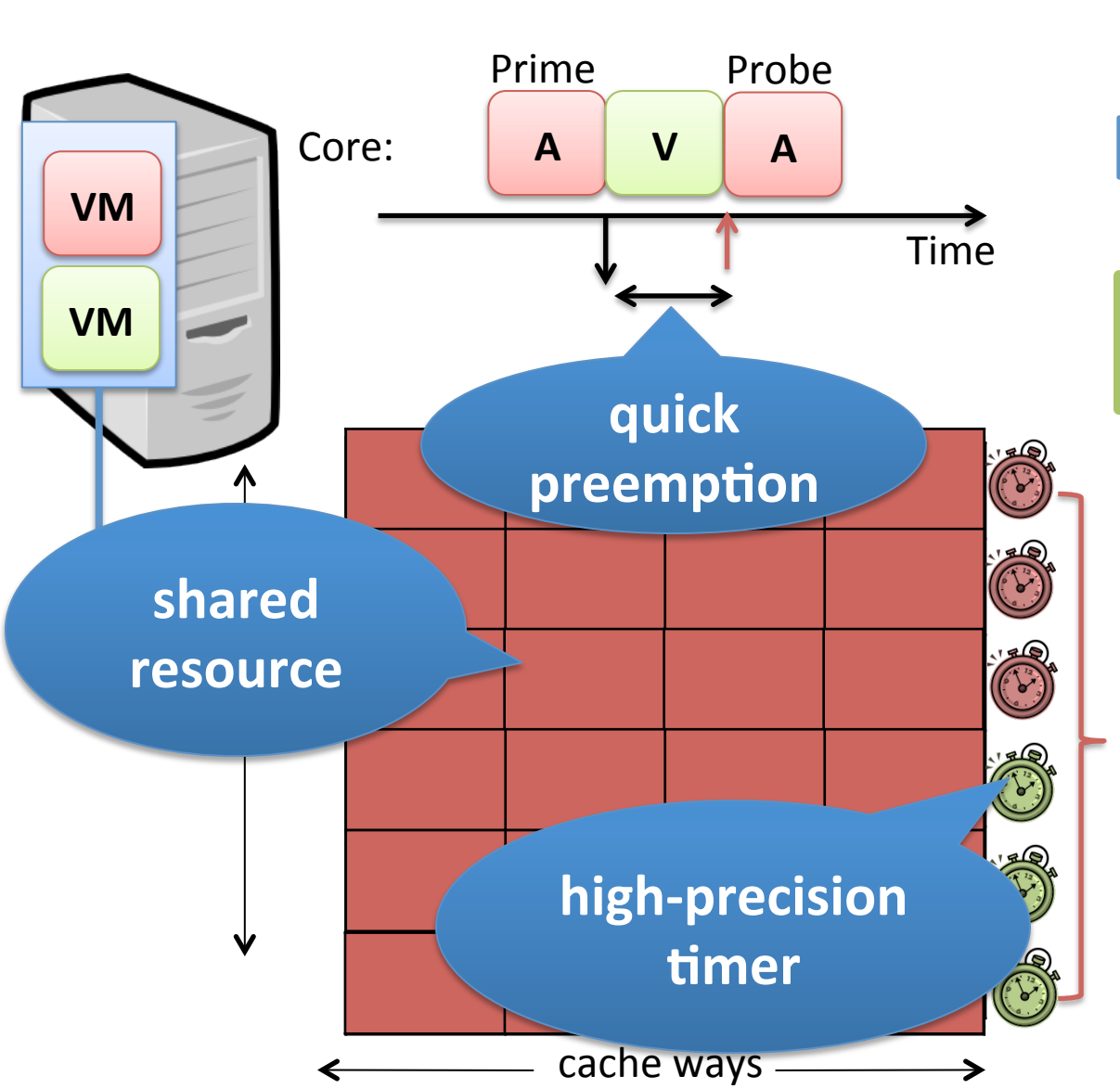


Problem: Cache-based Side-channels*



*Zhang, Juels, Reiter, Ristenpart, "Cross-VM Side-channels …", CCS'12

Requirements for Successful Side-channel



Secret

```
CRYPTO_FUNCTION(s):
s ← secret bit

if (s = 0) {
        OPERATION_A
}
if (s = 1) {
        OPERATION_B
}
```

Attacker Timing Profile

Extract secret information

*Zhang, Juels, Reiter, Ristenpart, "Cross-VM Side-channels …", CCS'12

Defenses against Side-channels

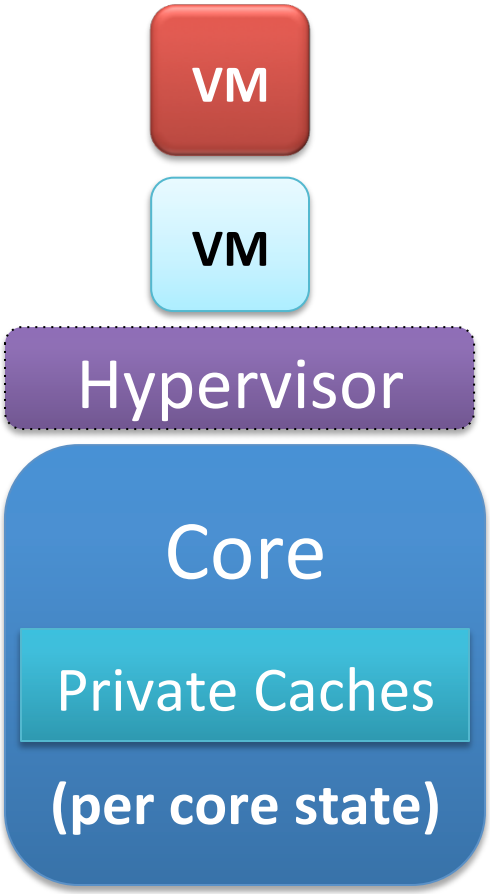
1. **Sharing**
   - Resource Partitioning [NoHype'10]
   - Specialized Hardware [RPcache'07]
   - Software-based partitioning [StealthMem'12]
2. **Access to high-resolution timers**
   - Reduce resolution [TimeWarp'12]
   - Removing timing channel [StopWatch'13]

No countermeasures deployed by providers!

3. **Quick cross-VM preemptions**
   - No prior work!

Our Solution: Soft Isolation

Allow sharing but limit frequency of dangerous VM interactions



Goals:

1. *Secure*: Controlled information leakage
2. *Commodity*: Easy to adopt
3. *Efficient*: Allow sharing, low overhead

… with simple changes to Hypervisor's CPU scheduler

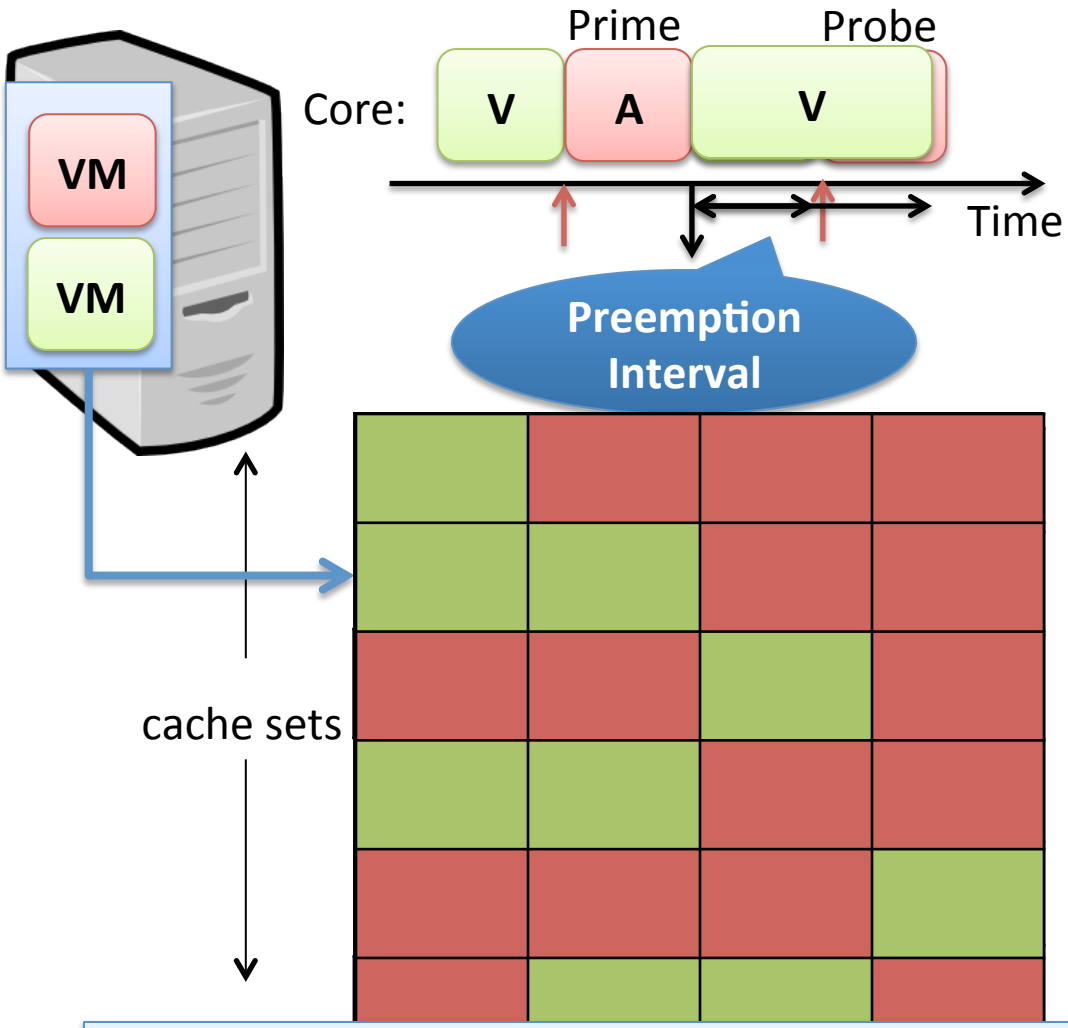
Rest of the talk …

1. **Background:** Quick Preemptions & Schedulers

2. **Soft-Isolation:** Scheduler-based defense

3. **Evaluation:** Security and Performance

Requirement for Quick Preemptions



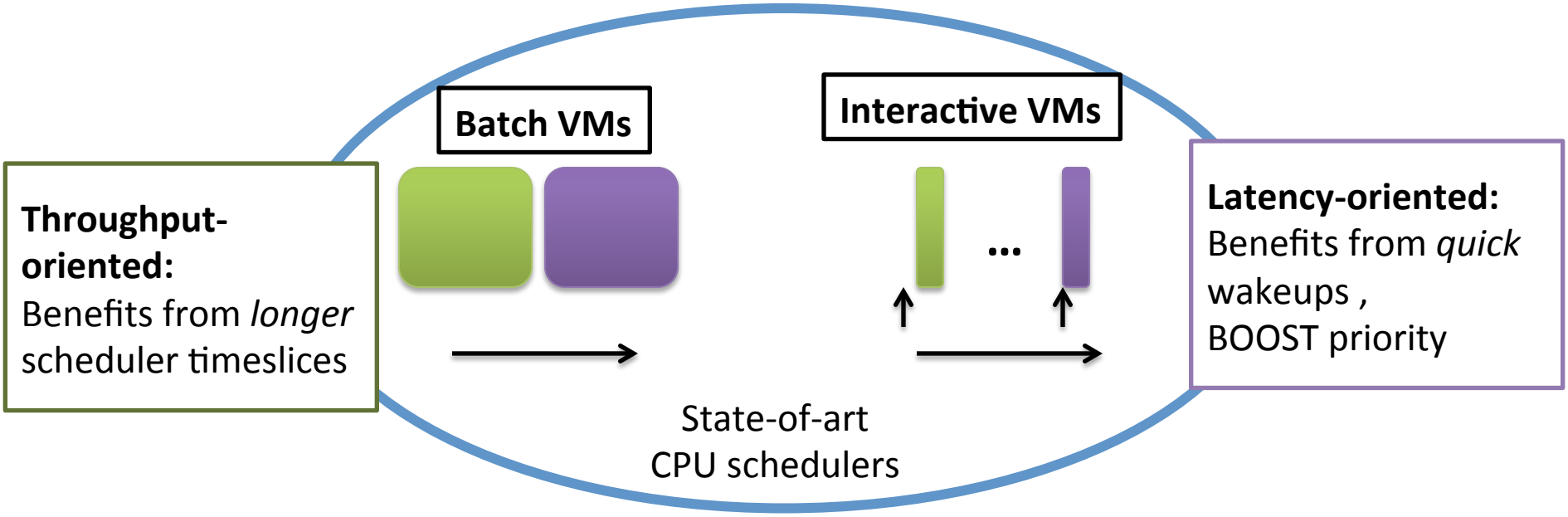

```
CRYPTO_FUNCTION(s):
s ← secret bit

if (s = 0) {
        OPERATION_A
}
if (s = 1) {
        OPERATION_B
}
```

Next subsequent code/task execution … **(or noise)**

Rate of preemption > Rate of event to measure

Why do schedulers allow quick preemptions?

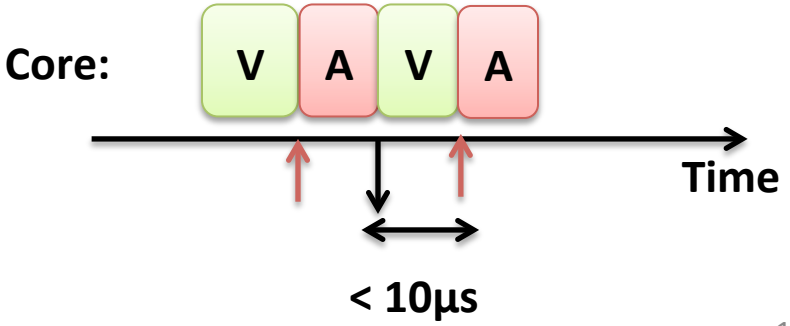


Throughput-oriented:
Benefits from *longer* scheduler timeslices

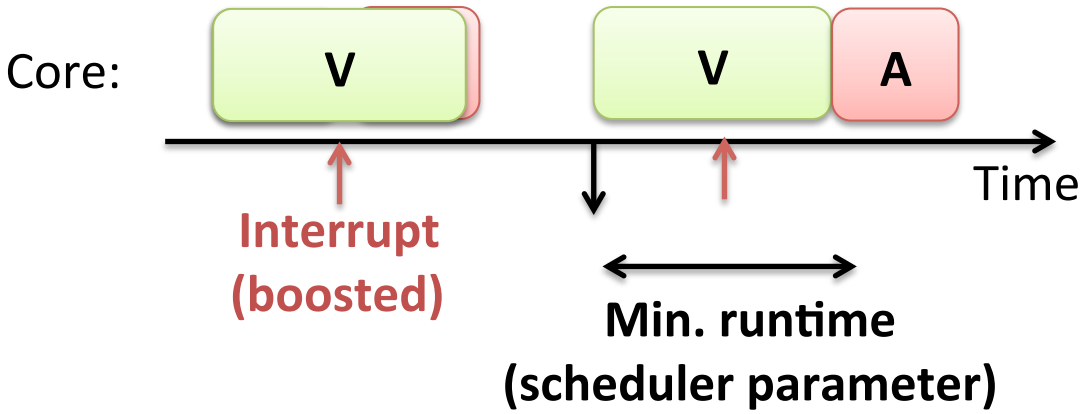
Latency-oriented:
Benefits from *quick* wakeups ,
BOOST priority

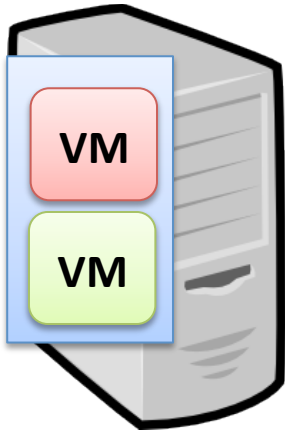
Prime-probe attacker:
Abuses BOOST priority, using interrupts.

Malicious VM



Soft-Isolation: Ratelimit Preemptions



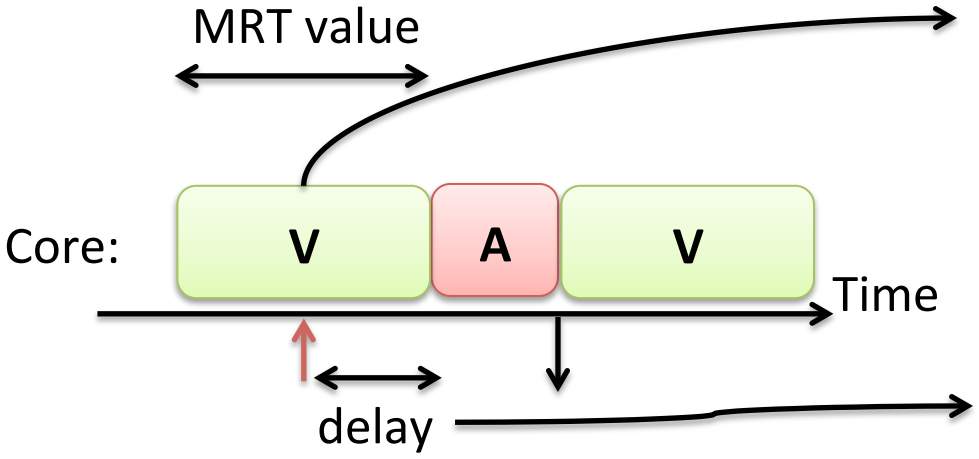


Available in Xen (and KVM)

- `ratelimit_us (and sched_min_granularity_ns)`
- Reduces VM-switches → Boosts batch-workload's performance

Minimum RunTime (MRT) guarantee → soft-isolation

MRT Guarantee and Open Questions



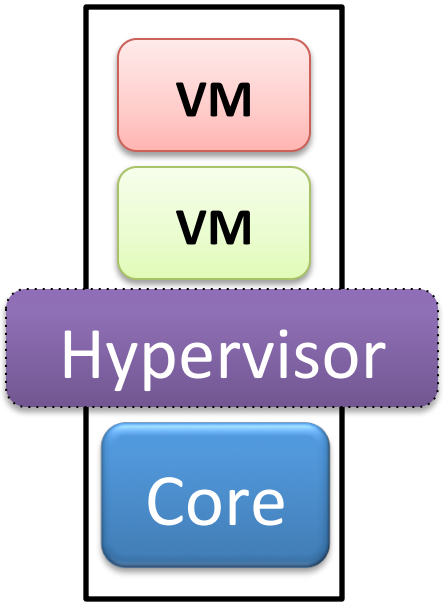
1. Can MRT defend against Cross-VM Side-channels? *(security evaluation)*

2. Trade-off between security and performance? *(performance overhead)*

Experimental Methodology

Two VMs:

1. Attacker
2. Victim



Setting similar to public clouds (e.g. EC2)

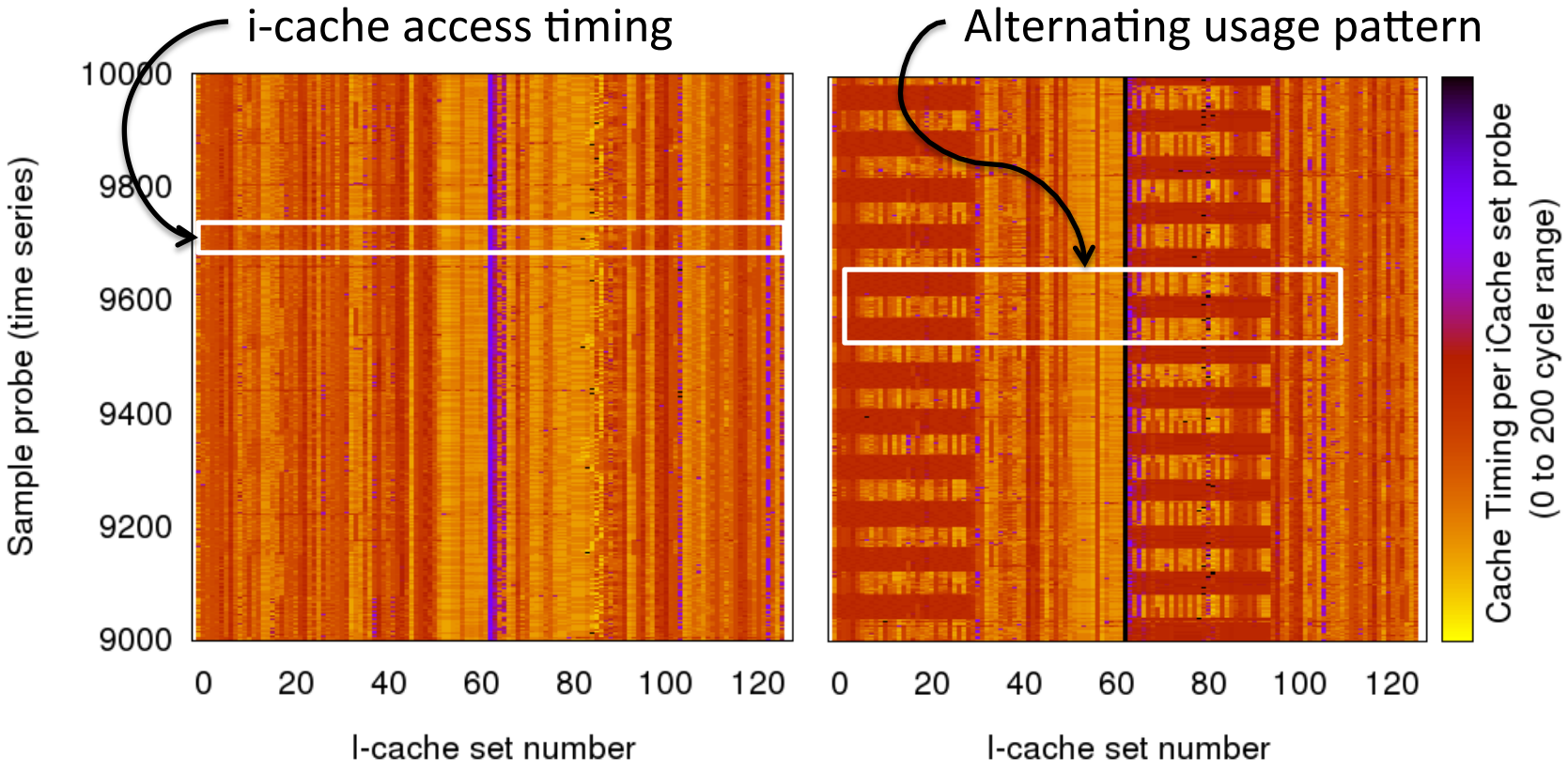
Xen Configuration

Xen Version	4.2.1
Scheduler	Credit Scheduler 1
Configuration (Non-work conserving)	40% cap on DomU VCPUs with equal weight
# VMs	6
# VCPUs per VM	2

Machine Configuration

Machine	Intel Xeon E5645, 2.4GHz, 6 cores, single package
Memory Hierarchy	Private 32KB L1 (I- and D-Cache), 256KB unified L2, 12MB shared L3 & 16GB DDR3 RAM.

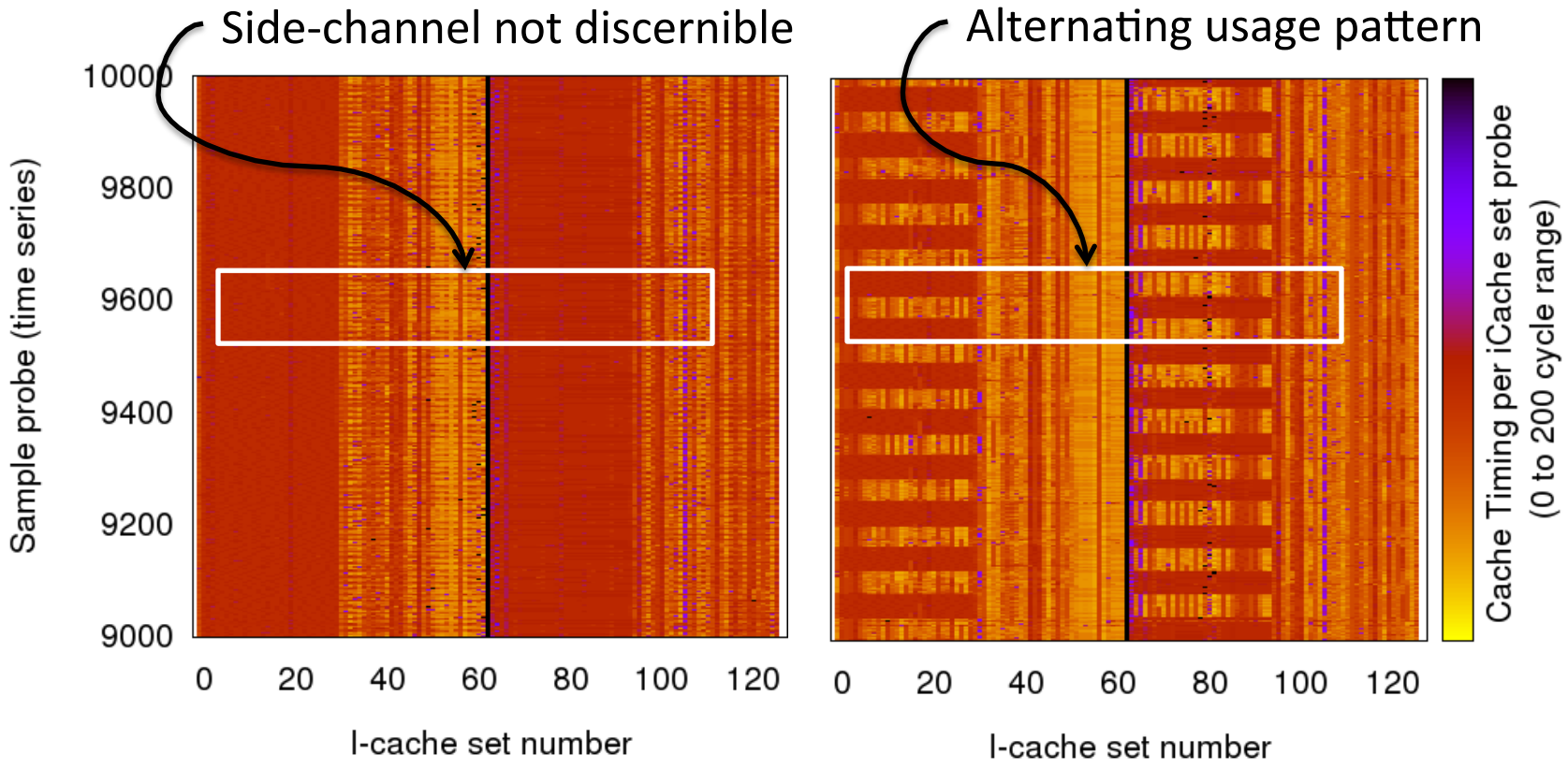
Security Evaluation : Prime-Probe Timing Profile



Idle Victim VM

Simple Victim VM
Under **Zero-MRT**

Security Evaluation : Prime-Probe Timing Profile

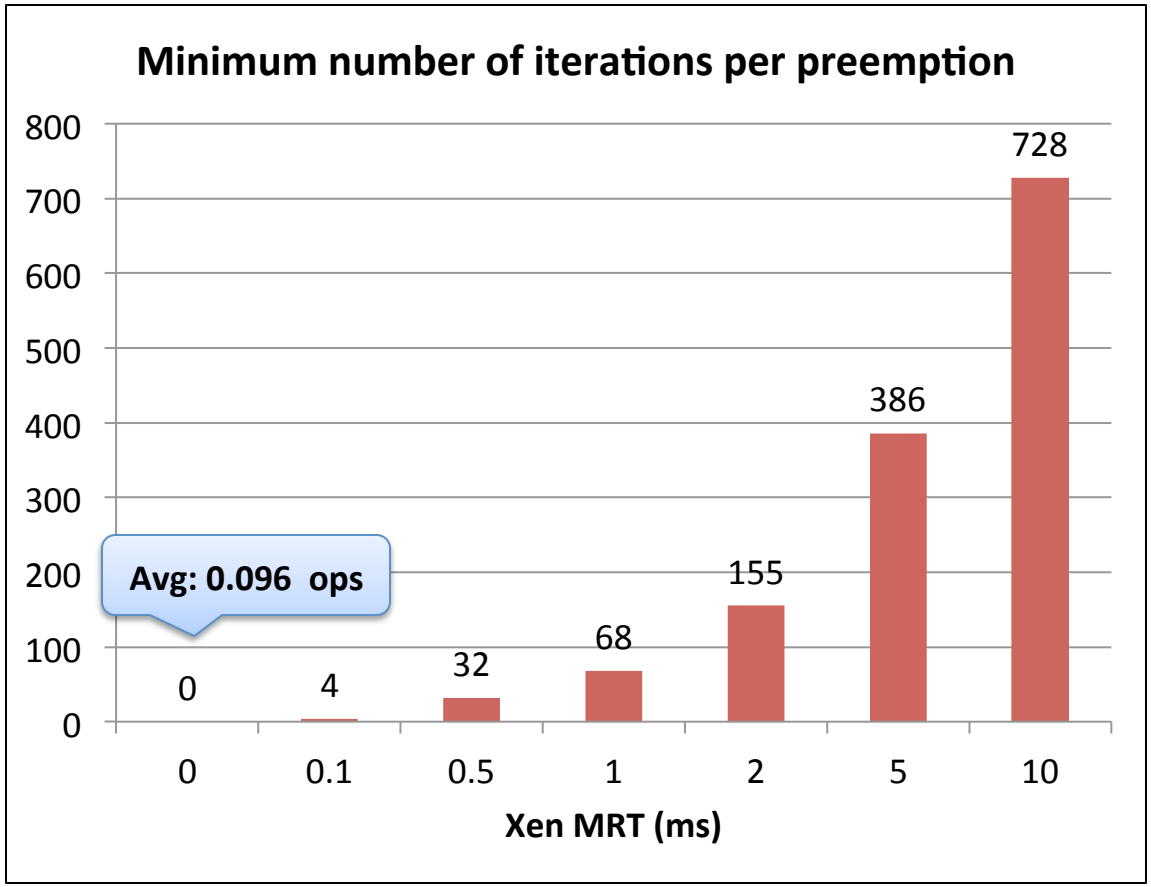


Simple Victim VM Under **1ms MRT**

Simple Victim VM Under **Zero-MRT**

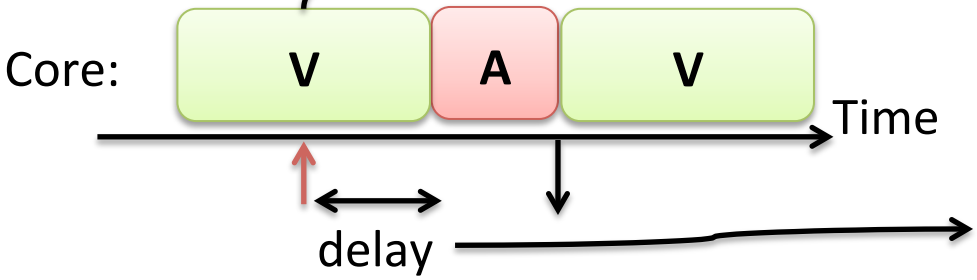
Security Evaluation: ElGamal Victim

ElGamal Side-channel require multiple preemptions within single iteration for noise-reduction [Zhang et al'12]




```
SQUAREMULT(x, e, N):
Let e_n , ..., e_1 be the bits of e
y ← 1
for i = n down to 1 do
    y ← SQUARE(y)
    y ← MODREDUCE(y, N)
    if e_i = 1 then
        y ← MULT(y, x)
        y ← MODREDUCE(y, N)
    end if
end for
return y
```

MRT Guarantee and Open Questions



1. Can MRT defend against Cross-VM Side-channels? (security evaluation)

2. Trade-off between security and performance? (performance overhead)

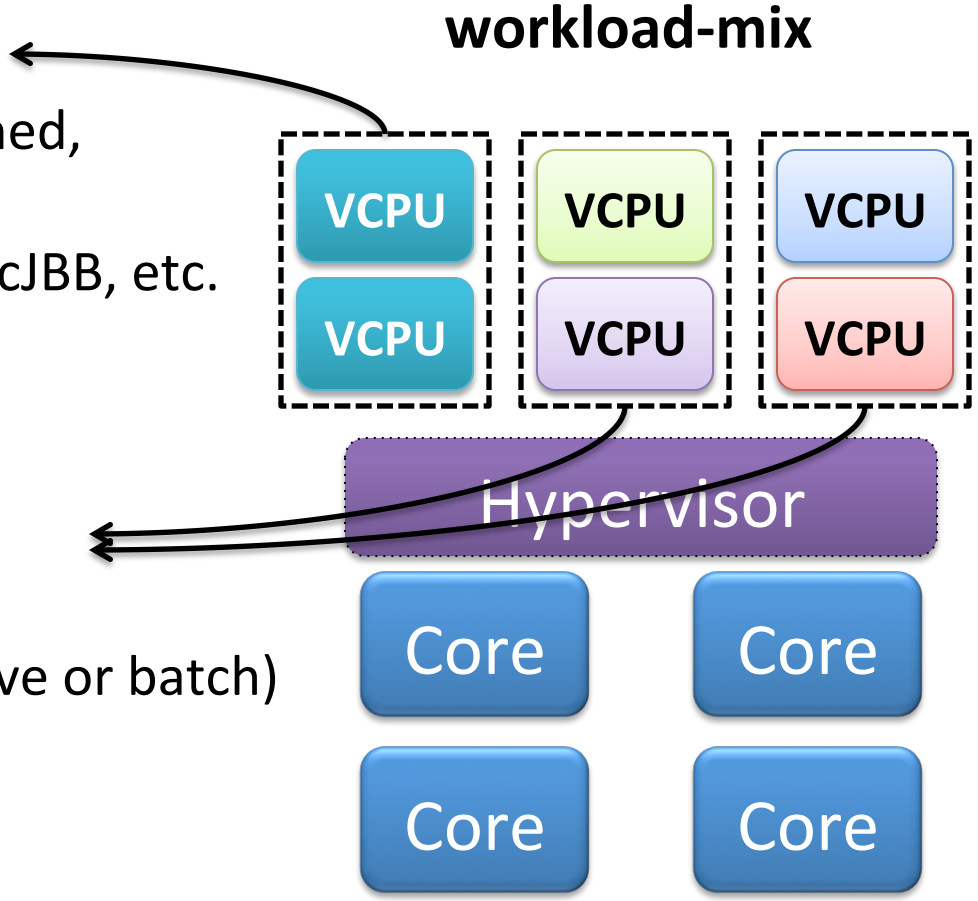
Performance Evaluation: Overall System Performance

Measured workload:

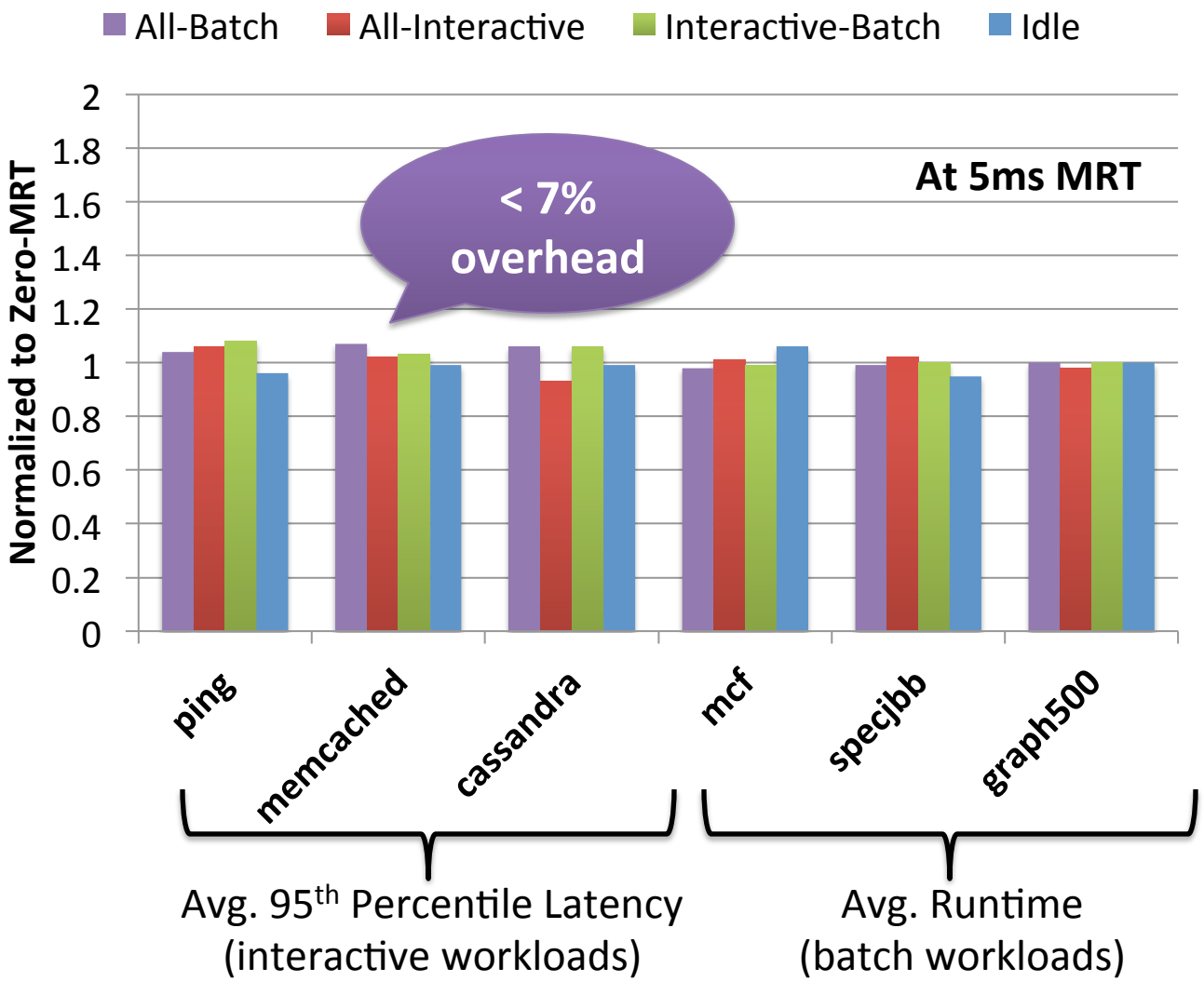
1. *Interactive* → memcached, cassandra, etc. and
2. *Batch* → graph500, specJBB, etc.

Competing workloads:

microbenchmarks → highly cache-thrashing + (interactive or batch)



Performance Evaluation: Overall System Performance



More details in the paper …

- Per-core State-Cleansing
  - Interactive VMs may still leak information
  - MRT + State-cleansing incur low overhead
- Detailed Performance and Security Analysis
  - 20+ graphs in the paper

It *is* cheap and easy to deploy!

Conclusion

5ms MRT + selective state-cleansing

- known attacks no longer work
- negligible overhead
- easy to adopt

Introduce new scheduler principle

- *soft-isolation = allow sharing + limit dangerous cross-VM interactions*

https://bitbucket.org/vvaradarajan/robsched

contact: venkatv@cs.wisc.edu