Bench Council 国际测试委员会 2020年国际测试委员会智能计算机与芯片联合大会

从软件角度防范侧信道攻击 --
基于影子线程的Intel SGX侧信道防御方法

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

王文浩

单用户、单任务→多用户、多任务、多安全域

1977年 Apple II 个人计算机
单用户、单任务

单用户、单任务→多用户、多任务、多安全域

1977年 Apple II 个人计算机
单用户、单任务

多任务、多安全域

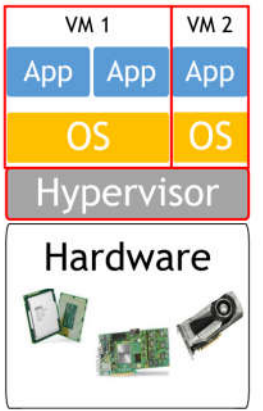
多租户

安全域的隔离机制

- 桌面操作系统
  - 进程隔离
- 移动操作系统
  - App隔离
- 云环境
  - 虚拟机隔离
  - 容器隔离
- 浏览器
  - 页面隔离
  - 沙箱隔离

侧信道攻击

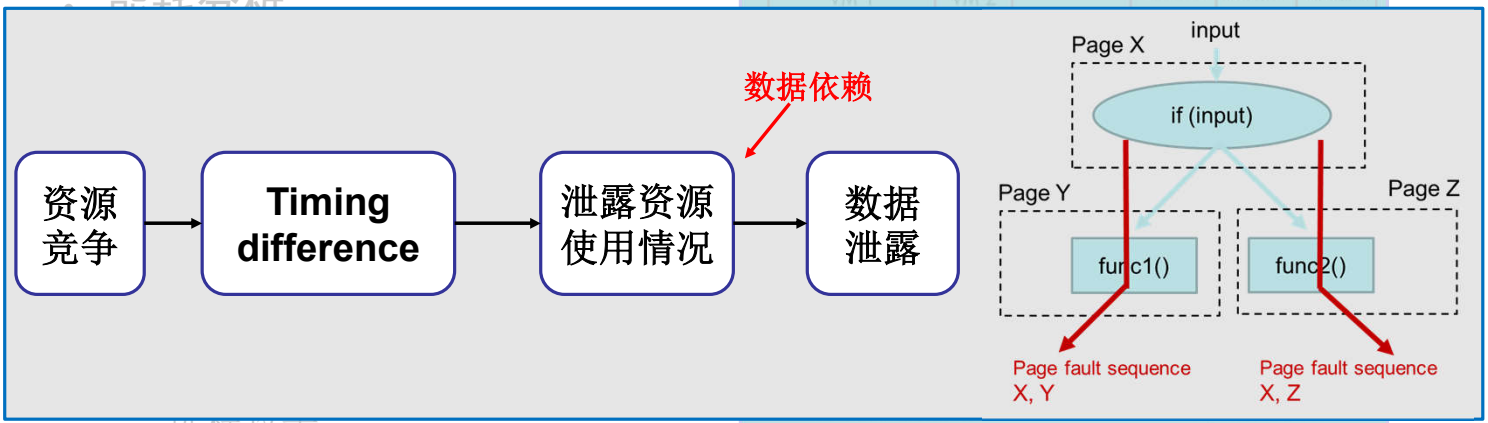
- 能耗分析
- 电磁泄露
- 时间泄露
- CPU微体系结构侧信道
  - 芯片内跨安全域的资源共享
  - 基于软件的攻击
  - 不需要物理接触
  - 远程攻击



Virtual Machines

Process-Level Isolation

侧信道攻击


资源竞争
Timing difference
泄露资源使用情况
数据泄露
数据依赖
input
Page X
if (input)
Page Y
Page Z
func1()
func2()
Page fault sequence X, Y
Page fault sequence X, Z

近年来的侧信道攻击...

	同物理核心侧信道攻击	跨物理核心侧信道攻击	跨CPU package侧信道攻击
传统微体系结构侧信道	L1/L2 cache、TLBleed（Usenix'18）、Branch Shadowing（Usenix'17）、BranchScope（ASPLOS'18）、Bluethunder（CHES'20）、LRU state（HPCA'20）	LLC cache	Thermal Covert Channels、DRAMA
Transient Execution攻击	SMoTherSpectre（CCS'19）、BranchSpec（ICCD'20）、Spectre-BTB/Spectre-iCache（WOOT'19）、RIDL（S&P'19）、FallOut（CCS'19）、CacheOut、ZombieLoad（CCS'19）、LVI（S&P'20）	Spectre-Cache（S&P'19）、CrossTalk（S&P'21）	

近年来的侧信道攻击...

	同物理核心侧信道攻击	跨物理核心侧信道攻击	跨CPU package侧信道攻击
传统微体系结构侧信道	L1/L2 cache、TLBleed（Usenix'18）、Branch Shadowing（Usenix'17）、BranchScope（ASPLOS'18）、Bluethunder（CHES'20）、LRU state（HPCA'20）	LLC cache	Thermal Covert Channels、DRAMA
Transient Execution攻击	SMoTherSpectre（CCS'19）、BranchSpec（ICCD'20）、Spectre-BTB/Spectre-iCache（WOOT'19）、RIDL（S&P'19）、FallOut（CCS'19）、CacheOut、ZombieLoad（CCS'19）、LVI（S&P'20）	Spectre-Cache（S&P'19）、CrossTalk（S&P'21）	

- 越靠近CPU核心
  - 共享资源越多
  - 利用的难度越低
  - 噪音越少

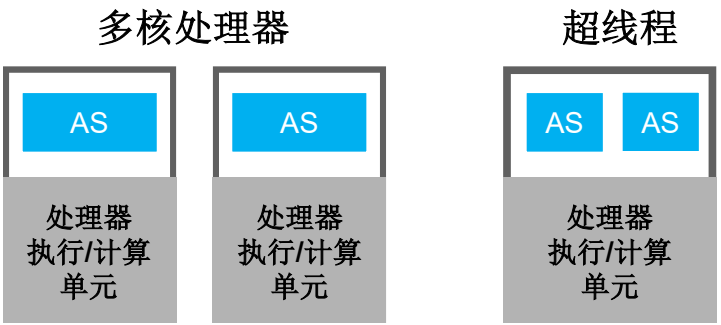
近年来的侧信道攻击...

	同物理核心侧信道攻击	跨物理核心侧信道攻击	跨CPU package侧信道攻击
传统微体系结构侧信道	L1/L2 cache、TLBleed（Usenix'18）、Branch Shadowing（Usenix'17）、BranchScope（ASPLOS'18）、Bluethunder（CHES'20）、LRU state（HPCA'20）	LLC cache	Thermal Covert Channels、DRAMA
Transient Execution攻击	SMoTherSpectre（CCS'19）、BranchSpec（ICCD'20）、Spectre-BTB/Spectre-iCache（WOOT'19）、RIDL（S&P'19）、FallOut（CCS'19）、CacheOut、ZombieLoad（CCS'19）、LVI（S&P'20）	Spectre-Cache（S&P'19）、CrossTalk（S&P'21）	

- 越靠近CPU核心
  - 共享资源越多
  - 利用的难度越低
  - 噪音越少

- 依赖于同物理核的侧信道攻击
  - 上下文切换 time-sliced sharing
  - 基于超线程 hyper-threaded sharing

近年来的侧信道攻击…



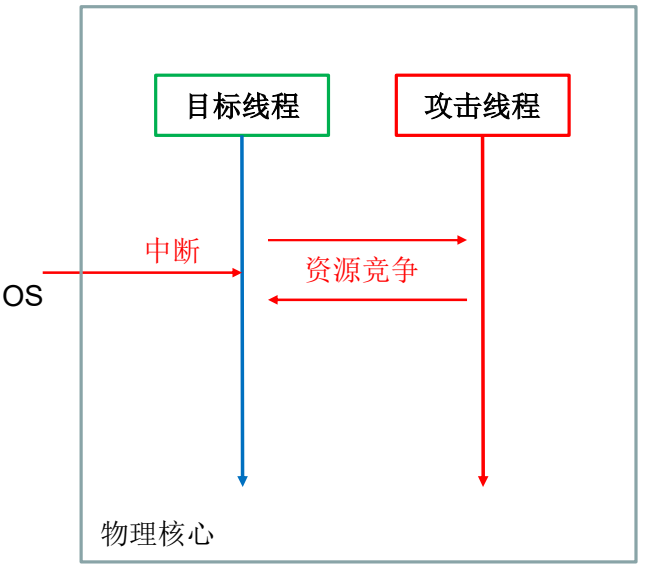
AS: 体系结构状态（通用寄存器、控制寄存器等）

- 越靠近CPU核心
  - 共享资源越多
  - 利用的难度越低
  - 噪音越少

- 依赖于同物理核的侧信道攻击
  - 上下文切换 time-sliced sharing
  - 基于超线程 hyper-threaded sharing

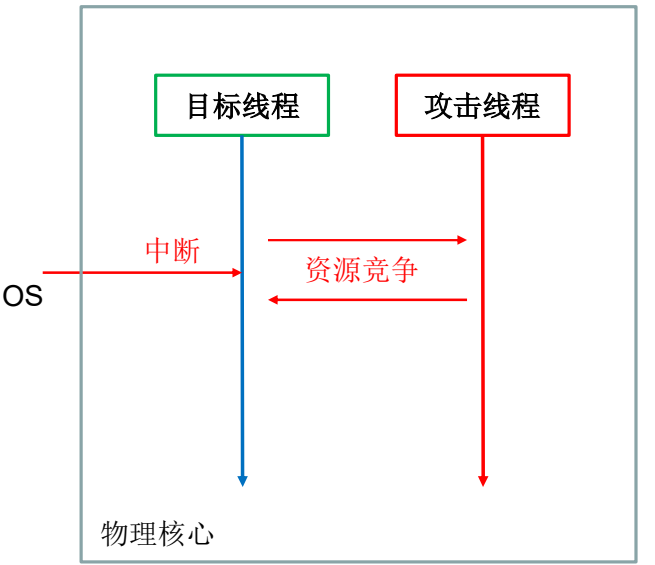
侧信道防御→禁止资源共享？

- Intel SGX 威胁模型
  - 不信任的操作系统
    - 触发中断、资源调度
    - CPU频率、cacheable属性



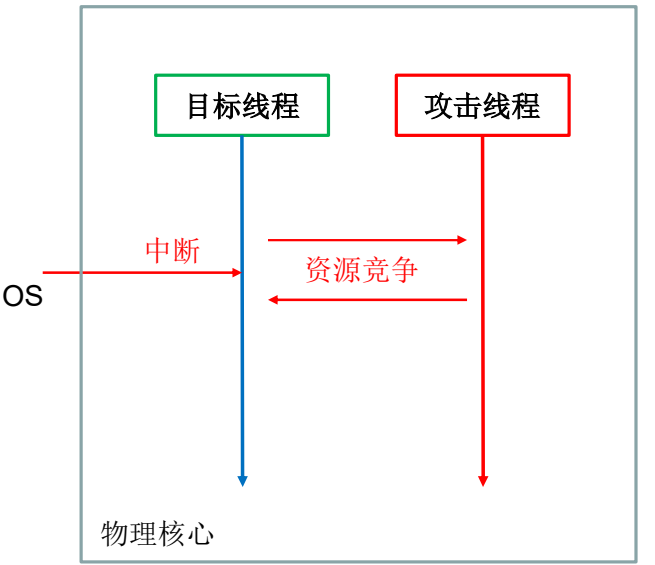
侧信道防御→禁止资源共享？

- Intel SGX 威胁模型
  - 不信任的操作系统
    - 触发中断、资源调度
    - CPU频率、cacheable属性
- Straw-man的解决方案
  - 禁用超线程
  - RDTSCP/RDPID



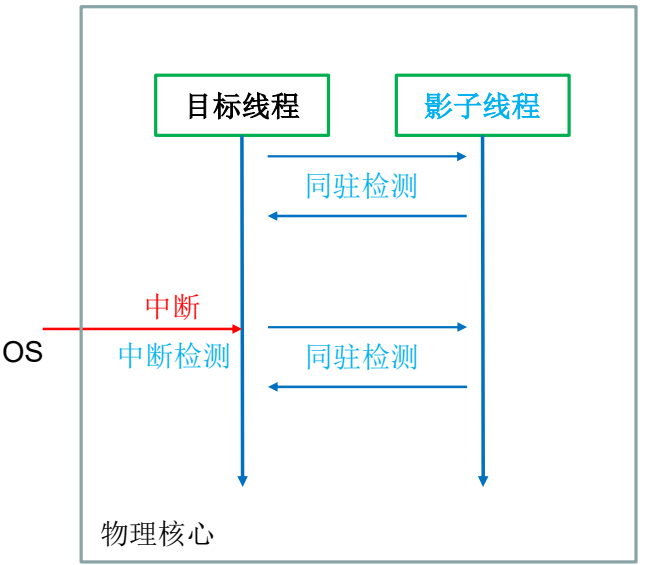
侧信道防御→禁止资源共享？

- Intel SGX 威胁模型
  - 不信任的操作系统
    - 触发中断、资源调度
    - CPU频率、cacheable属性
- Straw-man的解决方案
  - 禁用超线程
  - RDTSCP/RDPID
- HyperRace
  - 影子线程
  - 中断检测
  - 同驻检测



侧信道防御→禁止资源共享？

- Intel SGX 威胁模型
  - 不信任的操作系统
    - 触发中断、资源调度
    - CPU频率、cacheable属性
- Straw-man的解决方案
  - 禁用超线程
  - RDTSCP/RDPID
- HyperRace
  - 影子线程
  - 中断检测
  - 同驻检测

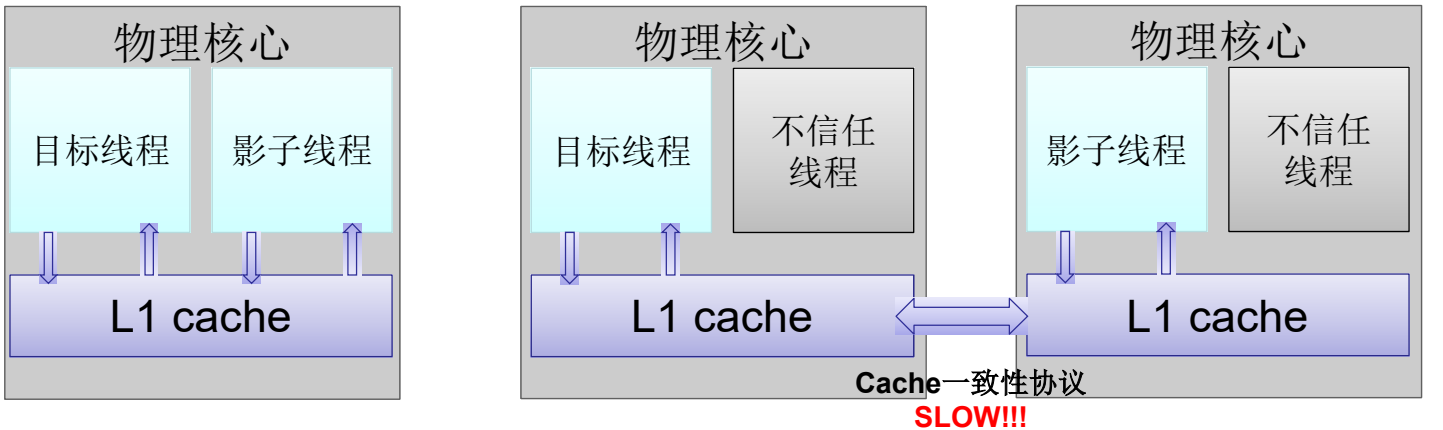


中断检测

- 当SGX线程被中断执行时
  - 线程的上下文状态保存在受保护内存区域（state saving area → SSA）
  - 当中断恢复时，从受保护内存区域恢复上下文状态
- 简单的中断检测机制
  - 进入SGX环境前，向SSA写入一个checker
  - 当SGX线程被中断时，SSA保存上下文状态
  - 定期检查这个checker是否被覆盖

同驻检测

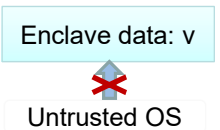
- 核心技术：基于数据竞争的同物理核心检测技术

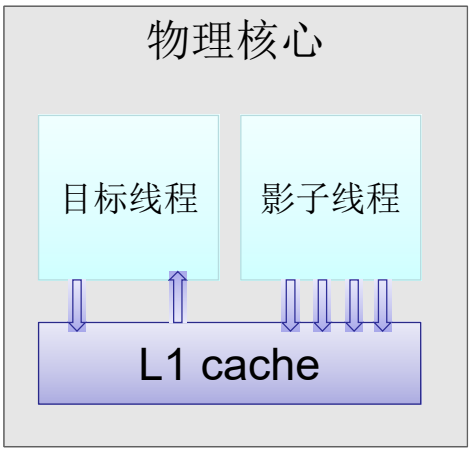


- 当处于同一物理核心时，数据竞争的概率接近 100%
- 当处于不同物理核心时，至少有一个线程发生数据竞争的概率接近于0

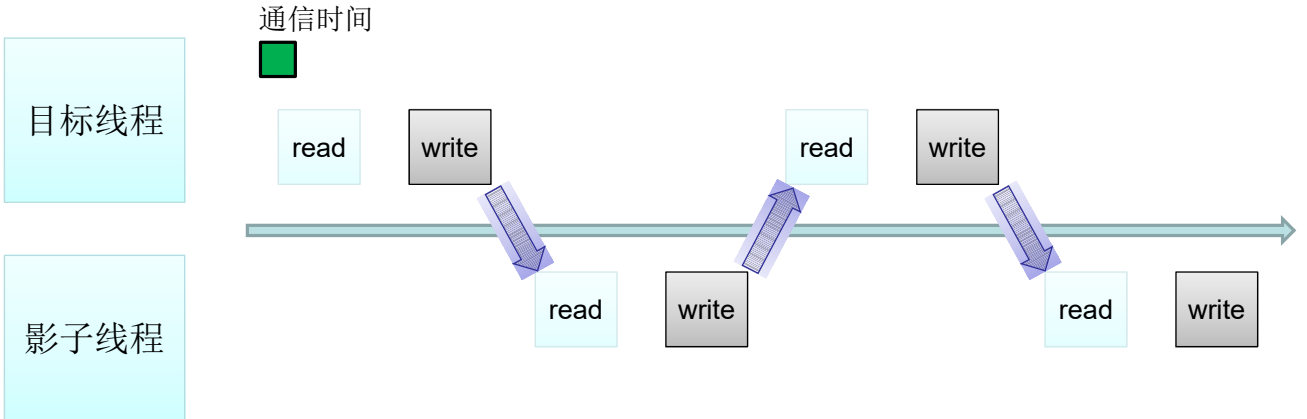
基于数据竞争的同物理核检测方案

- 两个线程操作同一个共享变量v:
  - 目标线程
    - 循环
      - 为v赋值0
      - 等待10个时钟周期
      - 读取v
  - 影子线程
    - 循环
      - 为v赋值1
  - 目标线程以极高的概率读取v的值为1



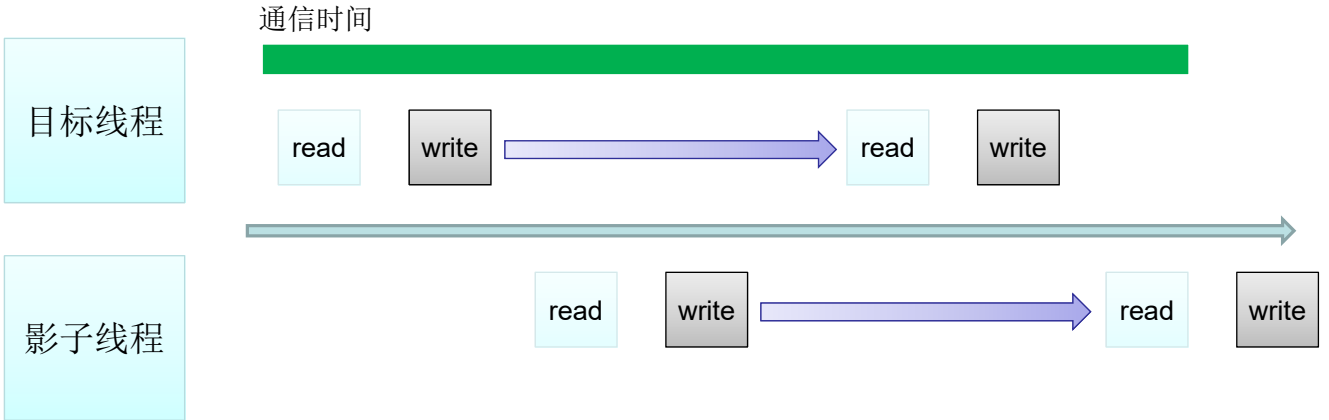


基于数据竞争的同物理核检测方案



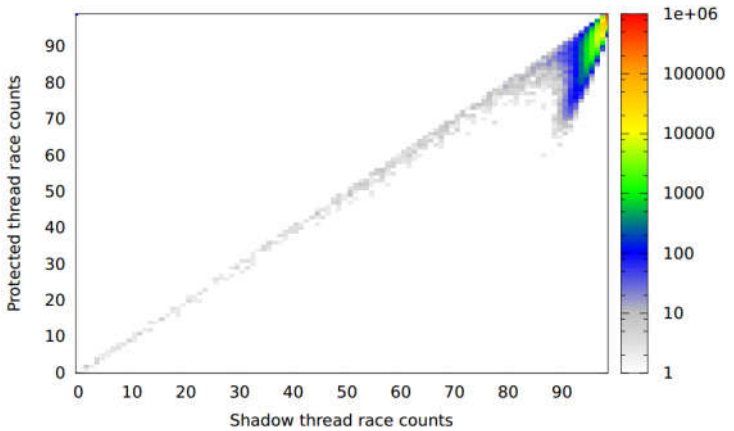
- 当同驻时，通信时间 < 等待时间
- 线程以极高的概率读取到另一个线程写入的数据

基于数据竞争的同物理核检测方案

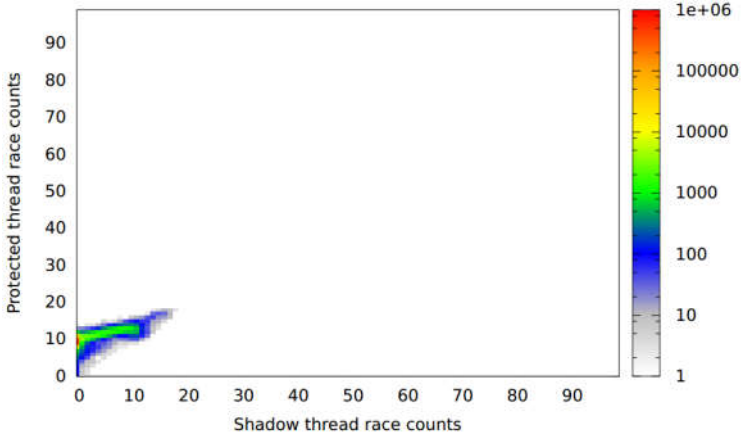


- 当不同驻时，通信时间 > 等待时间
- 线程以极低的概率读取到另一个线程写入的数据

基于数据竞争的同物理核检测方案



同驻：数据竞争概率高



不同驻：数据竞争概率低

基于数据竞争的同物理核检测方案

Thread T_0

```
<initialization>:
   mov $colocation_count, %rdx
   xor %rcx, %rcx
   ; co-location test counter
<synchronization>:
   ··· ; acquire lock 0
 .sync0:
   mov %rdx, (sync_addr1)
   cmp %rdx, (sync_addr0)
   je .sync1
   jmp .sync0
 .sync1:
   mfence
   mov $0, (sync_addr0)
<initialize a round>:
   mov $begin0, %rsi
   mov $1, %rbx
   mfence
   mov $addr_v, %r8
<co-location test>:
 .L0:
<load>:
   mov (%r8), %rax
<store>:
   mov %rsi, (%r8)
<update counter>:
   mov $0, %r10
   mov $0, %r11
   cmp $end0, %rax
   ; a data race happens?
   cmovl %rbx, %r10
   sub %rax, %r9
   cmp $1, %r9
   ; continuous number?
   cmova %r11, %r10
   add %r10, %rcx
   shl $b_count, %rbx
   ; bit length of $count
   mov %rax, %r9
   ; record the last number
<padding instructions 0>:
   nop
   nop
    ⋮
   nop
   mov (%r8), %rax
   mov (%r8), %rax
    ⋮
   mov (%r8), %rax
   dec %rsi
   cmp $end0, %rsi
   jne .L0
   ; finish 1 co-location test
<all rounds finished?>:
   ··· ; release lock 1
   dec %rdx
   cmp $0, %rdx
   jne .sync0
```

Thread T_1

```
<initialization>:
   mov $colocation_count, %rdx
   xor %rcx, %rcx
   ; co-location test counter
<synchronization>:
   ··· ; release lock 0
 .sync2:
   mov %rdx, (sync_addr0)
   cmp %rdx, (sync_addr1)
   je .sync3
   jmp .sync2
 .sync3:
   mfence
   mov $0, (sync_addr1)
<initialize a round>:
   mov $begin1, %rsi
   mov $1, %rbx
   mfence
   mov $addr_v, %r8
<co-location test>:
 .L2:
<load>:
   mov (%r8), %rax
<update counter>:
   mov $0, %r10
   mov $0, %r11
   cmp $end0, %rax
   ; a data race happens?
   cmovg %rbx, %r10
   sub %rax, %r9
   cmp $1, %r9
   ; continuous number?
   cmova %r11, %r10
   add %r10, %rcx
   shl $b_count, %rbx
   ; bit length of $count
    mov %rax, %r9
   ; record the last number
<store>:
   mov %rsi, (%r8)
<padding instructions 1>:
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   dec %rsi
   cmp $end1, %rsi
   jne .L2
   ; finish 1 co-location test
<all rounds finished?>:
   ··· ; acquire lock 1
   dec %rdx
   cmp $0, %rdx
   jne .sync2
```

基于数据竞争的同物理核检测方案

- 问题1：执行时间受到乱序执行和推测执行的影响

基于数据竞争的同物理核检测方案

- 问题1：执行时间受到乱序执行和推测执行的影响
- 解决：采用条件指令代替分支指令

基于数据竞争的同物理核检测方案

使用条件(CMOV)指令

Thread T_0

```
<initialization>:
   mov $colocation_count, %rdx
   xor %rcx, %rcx
   ; co-location test counter
<synchronization>:
   ··· ; acquire lock 0
 .sync0:
   mov %rdx, (sync_addr1)
   cmp %rdx, (sync_addr0)
   je .sync1
   jmp .sync0
 .sync1:
   mfence
   mov $0, (sync_addr0)
<initialize a round>:
   mov $begin0, %rsi
   mov $1, %rbx
   mfence
   mov $addr_v, %r8
<co-location test>:
 .L0:
<load>:
   mov (%r8), %rax
<store>:
   mov %rsi, (%r8)
<update counter>:
   mov $0, %r10
   mov $0, %r11
   cmp $end0, %rax
   ; a data race happens?
   cmovl %rbx, %r10
   sub %rax, %r9
   cmp $1, %r9
   ; continuous number?
   cmova %r11, %r10
   add %r10, %rcx
   shl $b_count, %rbx
   ; bit length of $count
   mov %rax, %r9
   ; record the last number
<padding instructions 0>:
   nop
   nop
    ·
    ·
   nop
   mov (%r8), %rax
   mov (%r8), %rax
    ·
    ·
   mov (%r8), %rax
   dec %rsi
   cmp $end0, %rsi
   jne .L0
   ; finish 1 co-location test
<all rounds finished?>:
   ··· ; release lock 1
   dec %rdx
   cmp $0, %rdx
   jne .sync0
```

Thread T_1

```
<initialization>:
   mov $colocation_count, %rdx
   xor %rcx, %rcx
   ; co-location test counter
<synchronization>:
   ··· ; release lock 0
 .sync2:
   mov %rdx, (sync_addr0)
   cmp %rdx, (sync_addr1)
   je .sync3
   jmp .sync2
 .sync3:
   mfence
   mov $0, (sync_addr1)
<initialize a round>:
   mov $begin1, %rsi
   mov $1, %rbx
   mfence
   mov $addr_v, %r8
<co-location test>:
 .L2:
<load>:
   mov (%r8), %rax
<update counter>:
   mov $0, %r10
   mov $0, %r11
   cmp $end0, %rax
   ; a data race happens?
   cmovg %rbx, %r10
   sub %rax, %r9
   cmp $1, %r9
   ; continuous number?
   cmova %r11, %r10
   add %r10, %rcx
   shl $b_count, %rbx
   ; bit length of $count
    mov %rax, %r9
   ; record the last number
<store>:
   mov %rsi, (%r8)
<padding instructions 1>:
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   dec %rsi
   cmp $end1, %rsi
   jne .L2
   ; finish 1 co-location test
<all rounds finished?>:
   ··· ; acquire lock 1
   dec %rdx
   cmp $0, %rdx
   jne .sync2
```

基于数据竞争的同物理核检测方案

- 问题1：执行时间受到乱序执行和推测执行的影响
- 解决：采用条件指令代替分支指令
- 问题2：同驻检测的置信区间的理论分析
- 解决：建立基于假设检验的安全模型

基于数据竞争的同物理核检测方案

建立基于假设检验的安全模型

Thread T_0

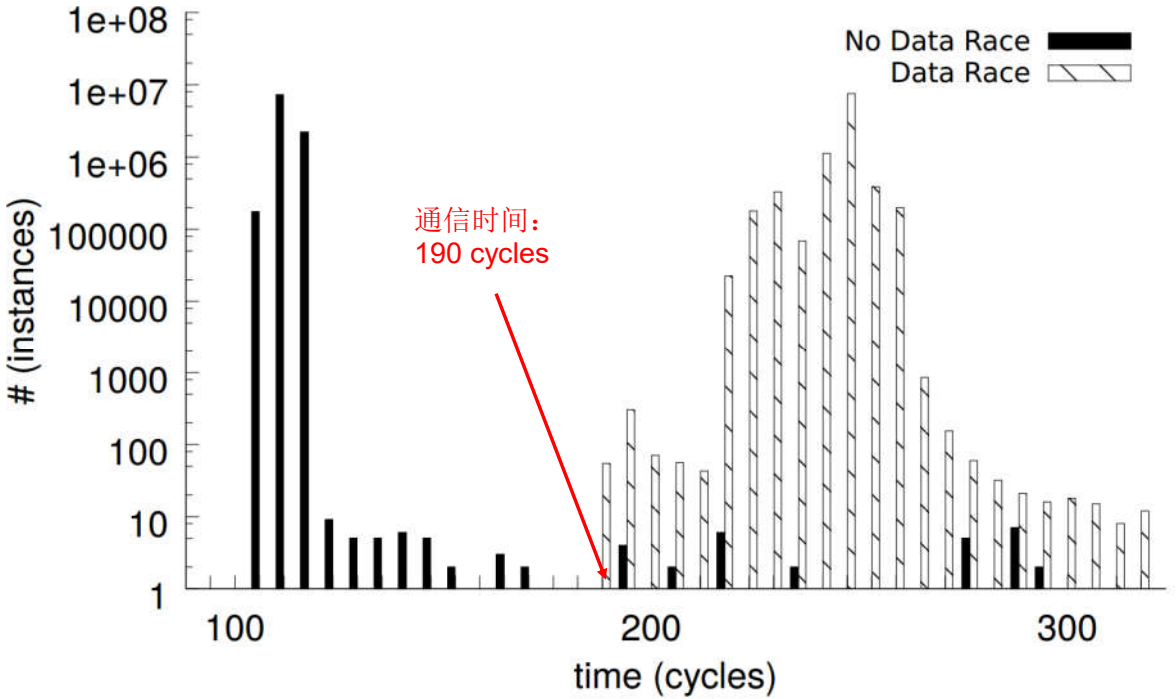
```
<initialization>:
   mov $colocation_count, %rdx
   xor %rcx, %rcx
   ; co-location test counter
<synchronization>:
   ··· ; acquire lock 0
.sync0:
   mov %rdx, (sync_addr1)
   cmp %rdx, (sync_addr0)
   je .sync1
   jmp .sync0
.sync1:
   mfence
   mov $0, (sync_addr0)
<initialize a round>:
   mov $begin0, %rsi
   mov $1, %rbx
   mfence
   mov $addr_v, %r8
<co-location test>:
.L0:
<load>:
   mov (%r8), %rax
<store>:
   mov %rsi, (%r8)
<update counter>:
   mov $0, %r10
   mov $0, %r11
   cmp $end0, %rax
   ; a data race happens?
   cmovl %rbx, %r10
   sub %rax, %r9
   cmp $1, %r9
   ; continuous number?
   cmova %r11, %r10
   add %r10, %rcx
   shl $b_count, %rbx
   ; bit length of $count
   mov %rax, %r9
   ; record the last number
<padding instructions 0>:
   nop
   nop
    ·
    ·
   nop
   mov (%r8), %rax
   mov (%r8), %rax
    ·
    ·
   mov (%r8), %rax
   dec %rsi
   cmp $end0, %rsi
   jne .L0
   ; finish 1 co-location test
<all rounds finished?>:
   ··· ; release lock 1
   dec %rdx
   cmp $0, %rdx
   jne .sync0
```

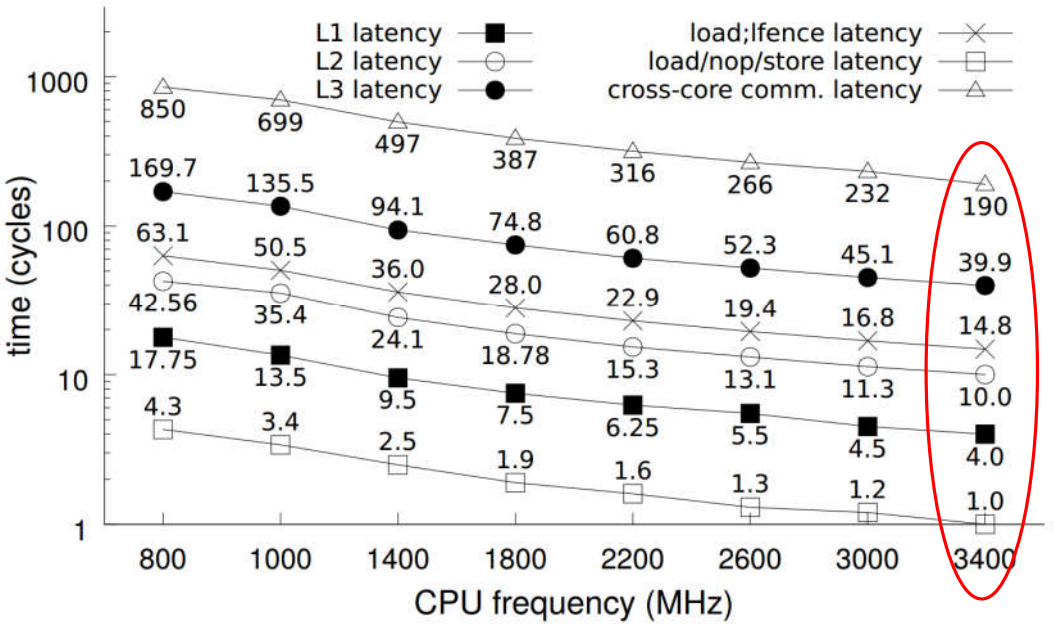
Thread T_1

```
<initialization>:
   mov $colocation_count, %rdx
   xor %rcx, %rcx
   ; co-location test counter
<synchronization>:
   ··· ; release lock 0
.sync2:
   mov %rdx, (sync_addr0)
   cmp %rdx, (sync_addr1)
   je .sync3
   jmp .sync2
.sync3:
   mfence
   mov $0, (sync_addr1)
<initialize a round>:
   mov $begin1, %rsi
   mov $1, %rbx
   mfence
   mov $addr_v, %r8
<co-location test>:
.L2:
<load>:
   mov (%r8), %rax
<update counter>:
   mov $0, %r10
   mov $0, %r11
   cmp $end0, %rax
   ; a data race happens?
   cmovg %rbx, %r10
   sub %rax, %r9
   cmp $1, %r9
   ; continuous number?
   cmova %r11, %r10
   add %r10, %rcx
   shl $b_count, %rbx
   ; bit length of $count
   mov %rax, %r9
   ; record the last number
<store>:
   mov %rsi, (%r8)
<padding instructions 1>:
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   dec %rsi
   cmp $end1, %rsi
   jne .L2
   ; finish 1 co-location test
<all rounds finished?>:
   ··· ; acquire lock 1
   dec %rdx
   cmp $0, %rdx
   jne .sync2
```

基于数据竞争的同物理核检测方案

- 问题1：执行时间受到乱序执行和推测执行的影响
- 解决：采用条件指令代替分支指令
- 问题2：同驻检测的置信区间的理论分析
- 解决：建立基于假设检验的安全模型
- 问题3：攻击者的能力
  - 攻击者试图影响通信时间与执行时间的关系
    - 调整频率


1e+08
1e+07
1e+06
100000
10000
1000
100
10
1
(instances)
100
200
300
time (cycles)
No Data Race
Data Race
通信时间：
190 cycles

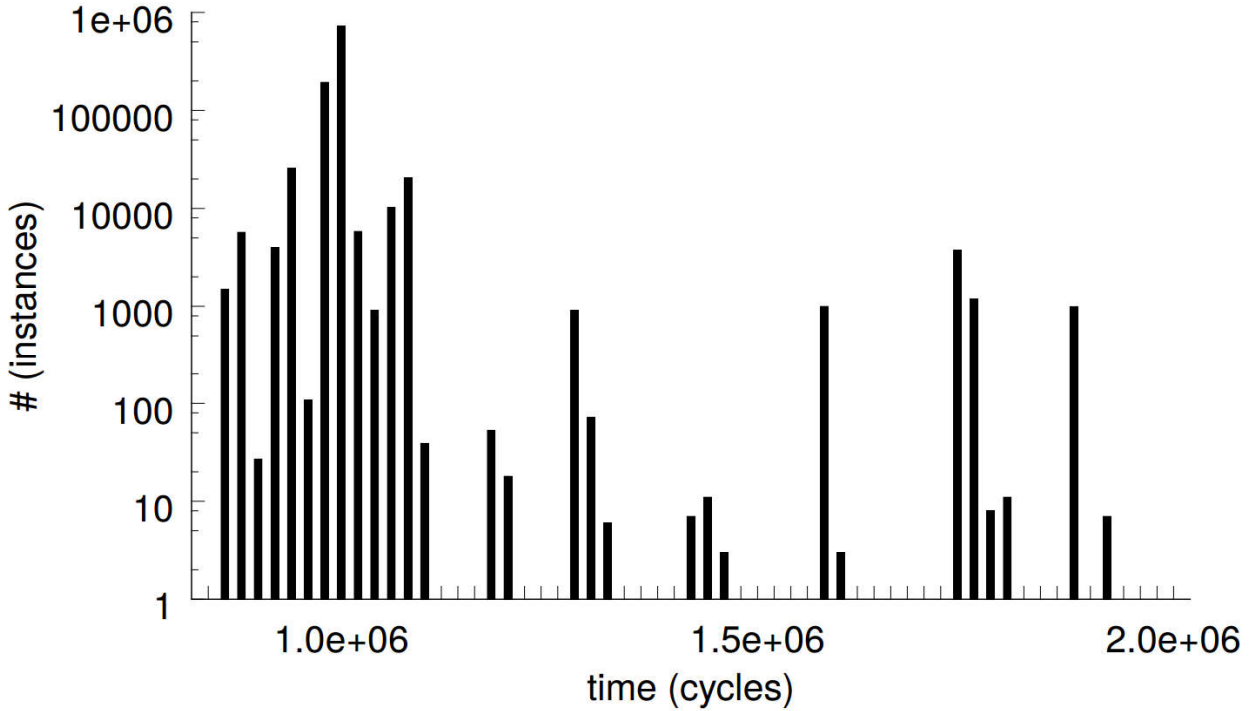


- CPU频率对通信时间和执行时间的影响基本一致
- 动态时钟频率调整的延迟远大于每次同驻检测的时间

基于数据竞争的同物理核检测方案

- 问题1：执行时间受到乱序执行和推测执行的影响
- 解决：采用条件指令代替分支指令
- 问题2：同驻检测的置信区间的理论分析
- 解决：建立基于假设检验的安全模型
- 问题3：攻击者的能力
  - 攻击者试图影响通信时间与执行时间的关系
    - 调整频率
    - 清空cache (1) prime (2) wbinvd

WBINVD 指令延迟 > 10^6 cycles



基于数据竞争的同物理核检测方案

- 问题1：执行时间受到乱序执行和推测执行的影响
- 解决：采用条件指令代替分支指令
- 问题2：同驻检测的置信区间的理论分析
- 解决：建立基于假设检验的安全模型
- 问题3：攻击者的能力
  - 攻击者试图影响通信时间与执行时间的关系
    - 调整频率
    - 清空cache (1) prime (2) wbinvd
    - 禁用cache

禁用cache对指令执行时间的影响

指令	启用cache	禁用cache	slowdown
nop	1.00	901	901X
load	1.01	1266	1253X
store	1.01	978	968X
load; lfence	14.82	2265	153X

基于数据竞争的同物理核检测方案

使用不同的填充指令序列

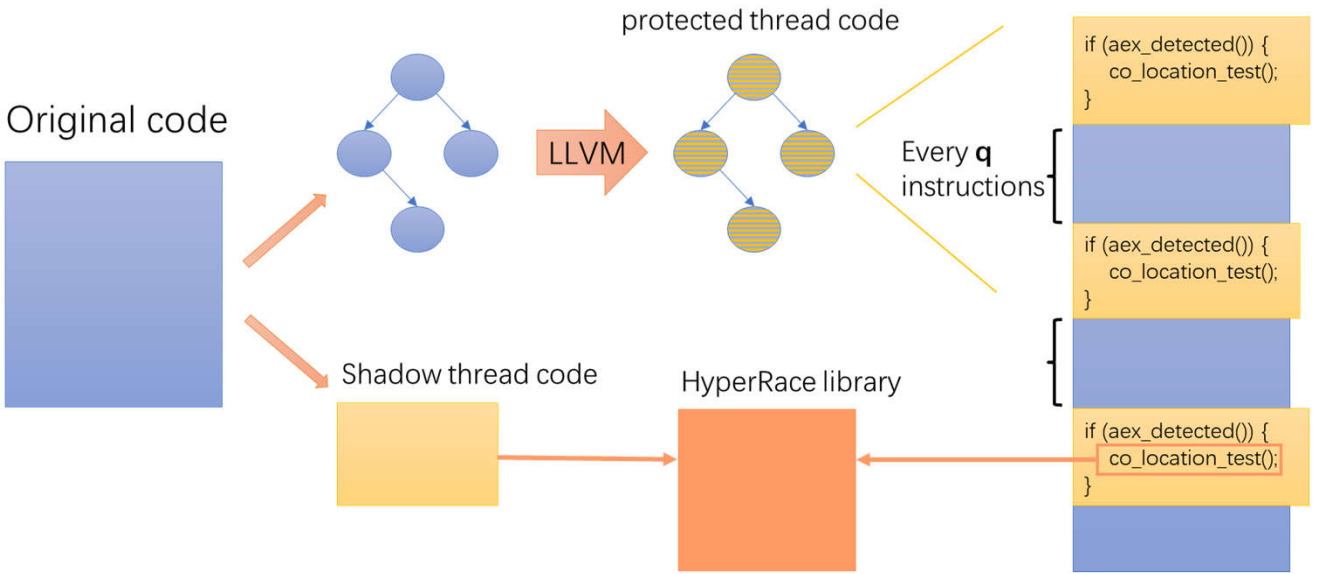
Thread T_0

```
<initialization>:
   mov $colocation_count, %rdx
   xor %rcx, %rcx
   ; co-location test counter
<synchronization>:
   ··· ; acquire lock 0
 .sync0:
   mov %rdx, (sync_addr1)
   cmp %rdx, (sync_addr0)
   je .sync1
   jmp .sync0
 .sync1:
   mfence
   mov $0, (sync_addr0)
<initialize a round>:
   mov $begin0, %rsi
   mov $1, %rbx
   mfence
   mov $addr_v, %r8
<co-location test>:
 .L0:
<load>:
   mov (%r8), %rax
<store>:
   mov %rsi, (%r8)
<update counter>:
   mov $0, %r10
   mov $0, %r11
   cmp $end0, %rax
   ; a data race happens?
   cmovl %rbx, %r10
   sub %rax, %r9
   cmp $1, %r9
   ; continuous number?
   cmova %r11, %r10
   add %r10, %rcx
   shl $b_count, %rbx
   ; bit length of $count
   mov %rax, %r9
   ; record the last number
<padding instructions 0>:
   nop
   nop
    ·
   nop
   mov (%r8), %rax
   mov (%r8), %rax
    ·
   mov (%r8), %rax
   dec %rsi
   cmp $end0, %rsi
   jne .L0
   ; finish 1 co-location test
<all rounds finished?>:
   ··· ; release lock 1
   dec %rdx
   cmp $0, %rdx
   jne .sync0
```

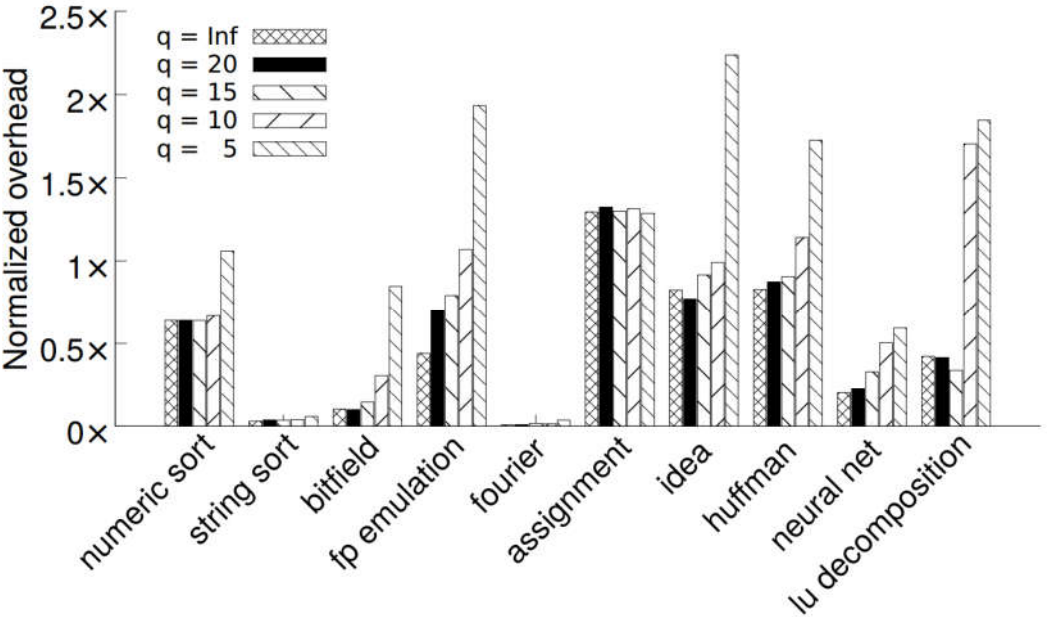
Thread T_1

```
<initialization>:
   mov $colocation_count, %rdx
   xor %rcx, %rcx
   ; co-location test counter
<synchronization>:
   ··· ; release lock 0
 .sync2:
   mov %rdx, (sync_addr0)
   cmp %rdx, (sync_addr1)
   je .sync3
   jmp .sync2
 .sync3:
   mfence
   mov $0, (sync_addr1)
<initialize a round>:
   mov $begin1, %rsi
   mov $1, %rbx
   mfence
   mov $addr_v, %r8
<co-location test>:
 .L2:
<load>:
   mov (%r8), %rax
<update counter>:
   mov $0, %r10
   mov $0, %r11
   cmp $end0, %rax
   ; a data race happens?
   cmovg %rbx, %r10
   sub %rax, %r9
   cmp $1, %r9
   ; continuous number?
   cmova %r11, %r10
   add %r10, %rcx
   shl $b_count, %rbx
   ; bit length of $count
   mov %rax, %r9
   ; record the last number
<store>:
   mov %rsi, (%r8)
<padding instructions 1>:
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   mov (%r8), %rax
   lfence
   dec %rsi
   cmp $end1, %rsi
   jne .L2
   ; finish 1 co-location test
<all rounds finished?>:
   ··· ; acquire lock 1
   dec %rdx
   cmp $0, %rdx
   jne .sync2
```

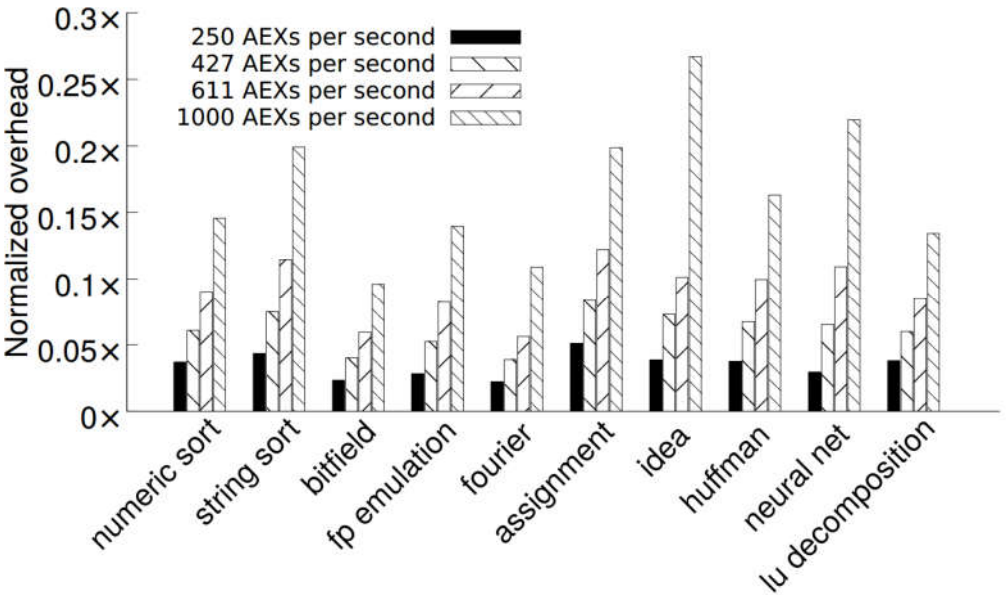
HyperRace：消除基于中断和HT的侧信道威胁



中断检测的性能影响


Normalized overhead
2.5×
2×
1.5×
1×
0.5×
0×
q = Inf
q = 20
q = 15
q = 10
q = 5
numeric sort
string sort
bitfield
fp emulation
fourier
assignment
idea
huffman
neural net
lu decomposition

同驻检测的性能影响


0.3×
0.25×
0.2×
0.15×
0.1×
0.05×
0×
Normalized overhead
250 AEXs per second
427 AEXs per second
611 AEXs per second
1000 AEXs per second
numeric sort
string sort
bitfield
fp emulation
fourier
assignment
idea
huffman
neural net
lu decomposition

总结

- 同物理核心的侧信道攻击
  - 粒度更细、噪音更少、更容易利用
  - 依赖于两种资源共享
    - 上下文切换 time-sliced sharing
    - 基于超线程 hyper-threaded sharing
- HyperRace防御
  - 影子线程、中断检测、同驻检测
  - 不需要彻底关闭HT，仅在需要时独占物理核心
  - 更多细节
    - Racing in Hyperspace: Closing Hyper-Threading Side Channels on SGX with Contrived Data Races（S&P'18）