Ever Growing CPU States: Context Switch with Less Memory and Better Performance

Fenghua Yu <fenghua.yu@intel.com>

Agenda

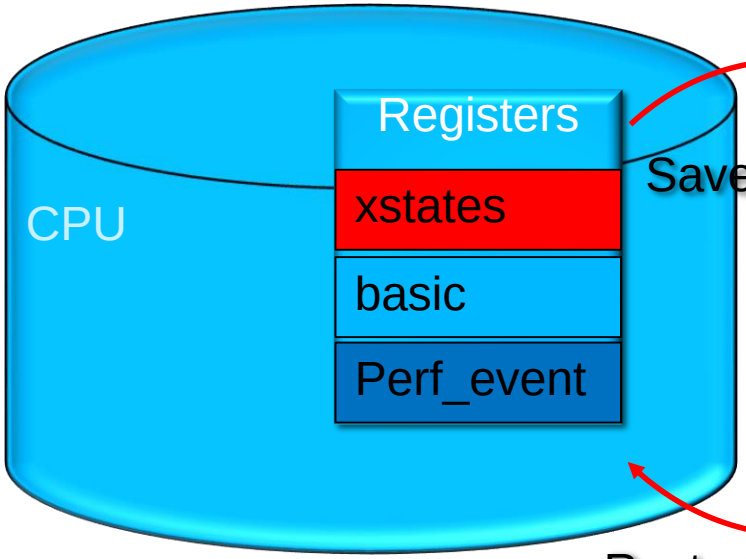
- **Introduction**
- Impact of xstates in Context Switch
- Context Switch Optimizations for xstates
- Kernel Implementation for xstates Context Switch
- Security Concern and Solution
- Status and Future Work
- Q & A

Terms

To reduce confusion, the following terms are used in this presentation:

- FP: Floating Point
- SSE: Streaming SIMD Extensions
- AVX/AVX2/AVX-512: Advanced Vector Extensions
- MPX: Memory Protection Extensions
- Extended States (xstates): Currently include FP/SSE/AVX2/MPX/AVX-512 registers
- Xsave area: kernel mem allocated for xstates context per process defined in xsave_struct

Introduction – X86 Context Switch Flow



Save registers

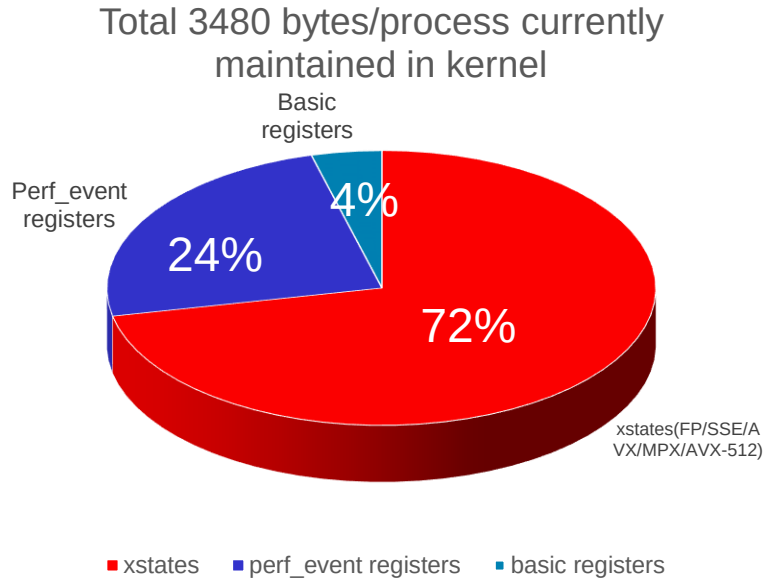
Restore registers

Kernel memory
Register context for process A
xstates registers
per_event registers
Basic registers (segment, ip, cr, etc)
Register context for process B
Xstates registers
Per_event registers
Basic registers(segment, ip, cr, etc)

Agenda

- Introduction
- Impact of xstates in Context Switch
- Context Switch Optimizations for xstates
- Kernel Implementation for xstates Context Switch
- Security Concern and Solution
- Status and Future Work
- Q & A

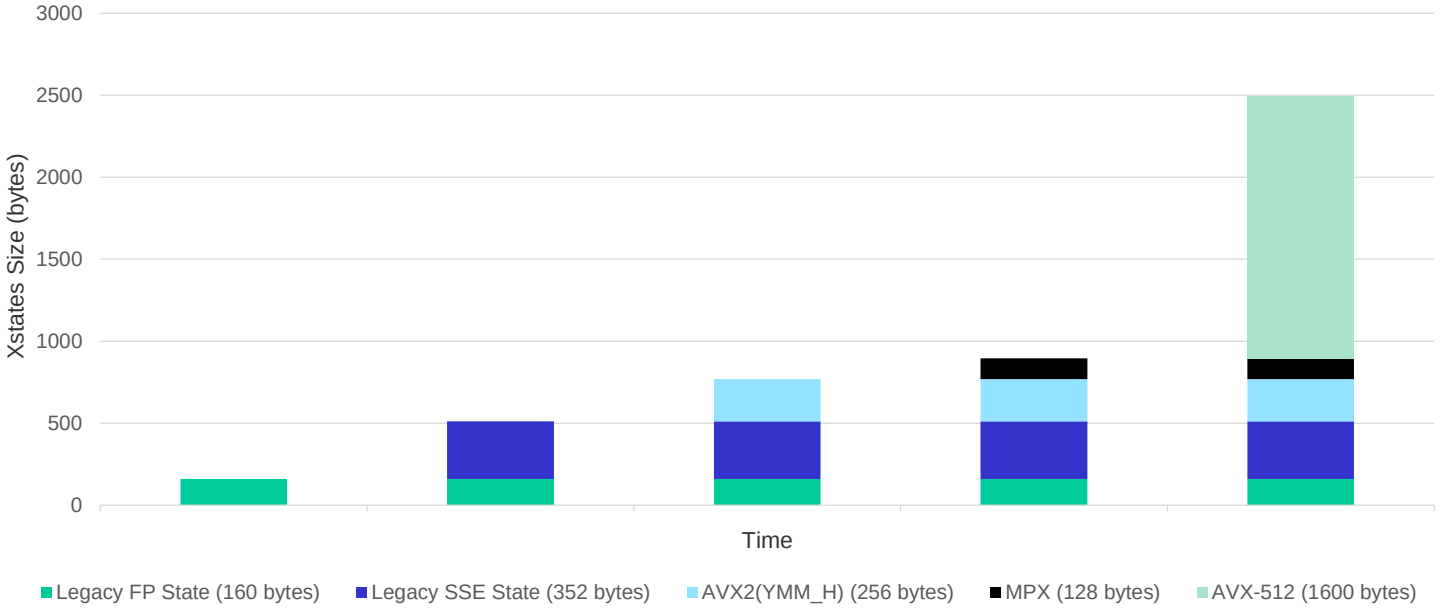
Why Care Xstates? Large Portion of CPU States Are From Xstates



Impact of improperly handling large xstates:

1. Large memory footprint
2. Large cache footprint
3. Slow context switch execution
4. Slow response to user and bad user experience
5. Overall performance degradation

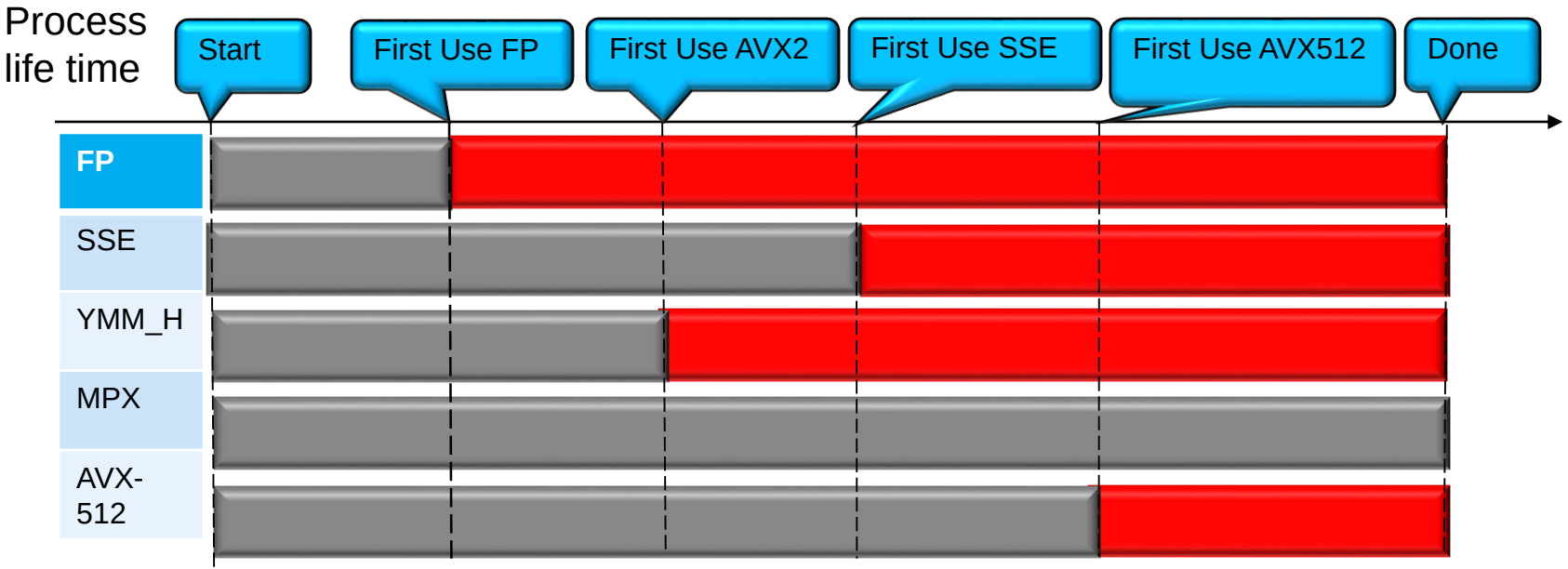
And Xstates Are Growing Over Years



Agenda

- Introduction
- Impact of xstates in Context Switch
- Context Switch Optimizations for xstates
- Kernel Implementation for xstates Context Switch
- Security Concern and Solution
- Status and Future Work
- Q & A

Optimization 1: Init Optimization

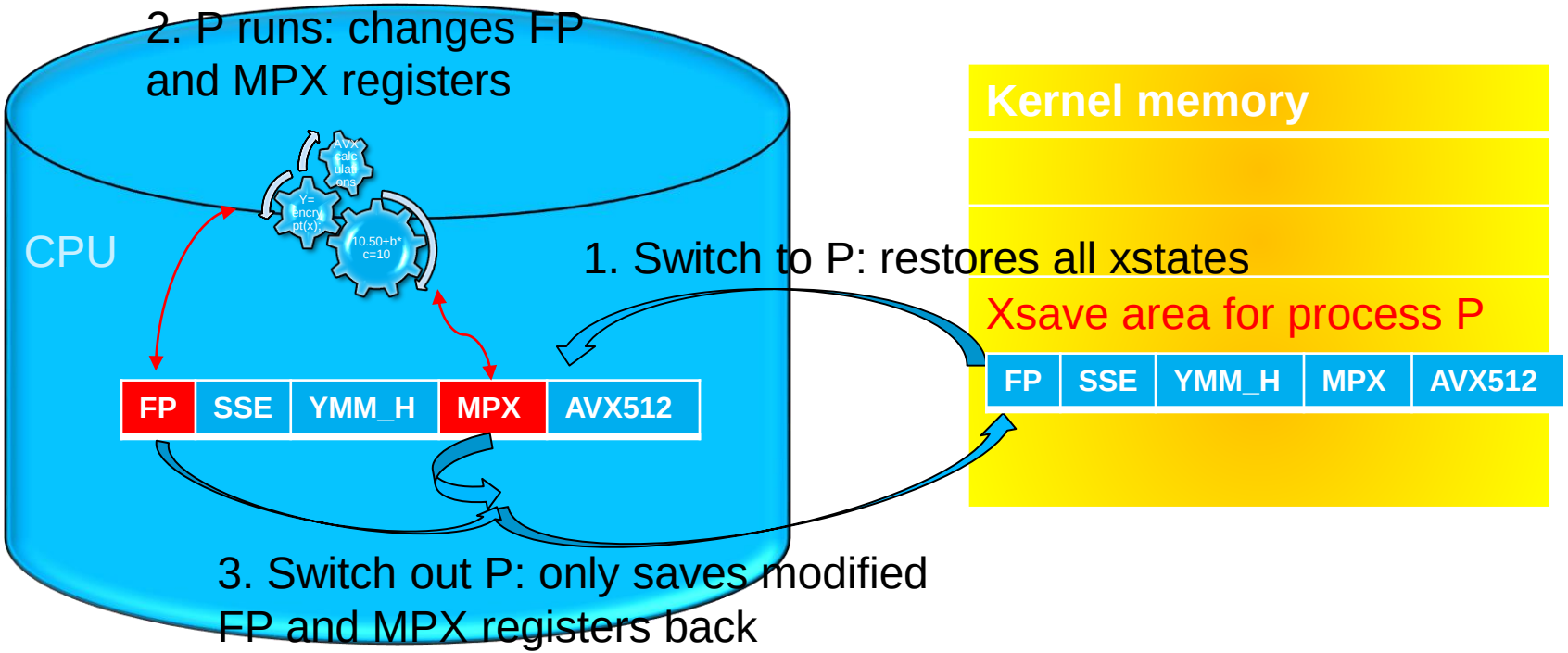


The state is not used and is not saved/restored during context switch

The state is used and is saved/restored during context switch

Start saving/restoring a state only after it is first used

Optimization 2: Modified Optimization



Example of how only modified FP and MPX registers are detected and saved

Optimization 3: Compacted Format of Xsave Area

Scenario 1: FP/SSE/AVX/MPX/AVX3 are enabled in processor

Xstates	Byte offset
Legacy FP State	0
Legacy SSE State	160
Xsave Header Data	512
YMM_H State (256 bytes)	576
MPX_BNDREGS(64 bytes)	832
MPX BNDCSR (64 bytes)	896
AVX-512 KMASK (64 bytes)	960
AVX-512 ZMM_H (512 bytes)	1024
AVX-512 ZMM (1024 bytes)	1536

Total size: 2560 bytes/process

Scenario 2: Only FP/SSE/AVX512 are enabled in processor

Xstates	Byte offset
Legacy FP State	0
Legacy SSE State	160
Xsave Header Data	512
AVX-512 KMASK (64 bytes)	576
AVX-512 ZMM_H (512 bytes)	640
AVX-512 ZMM (1024 bytes)	1152

Total size: 2176 bytes/process

Scenario 2 occupies 384 bytes or 15% less mem than scenario 1 for xsave area per process

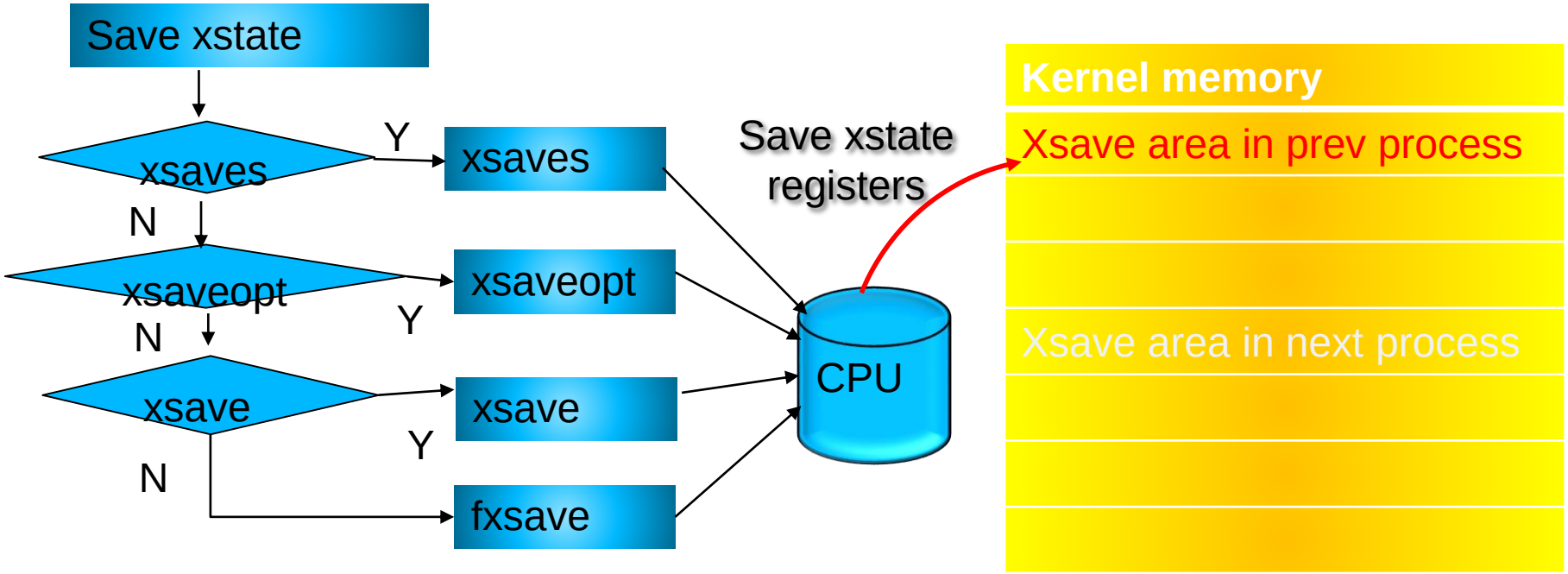
Xstates Context Switch Instructions Overview

Instructions	Format		Optimization		States		Xsave Area
	Standard	Compacted	Init	Modified	User	Supervisor	
fxsave/ Fxrstor	✓				✓		Legacy FP, SSE
xsave/ xrstor	✓				✓		Legacy FP, SSE AVX2, AVX-512, MPX
xsaveopt/ xrstor	✓		✓	✓	✓		
xsavec/ xrstor		✓	✓		✓		
xsaves/ xrstors		✓	✓	✓	✓	✓	All above + Supervisor States

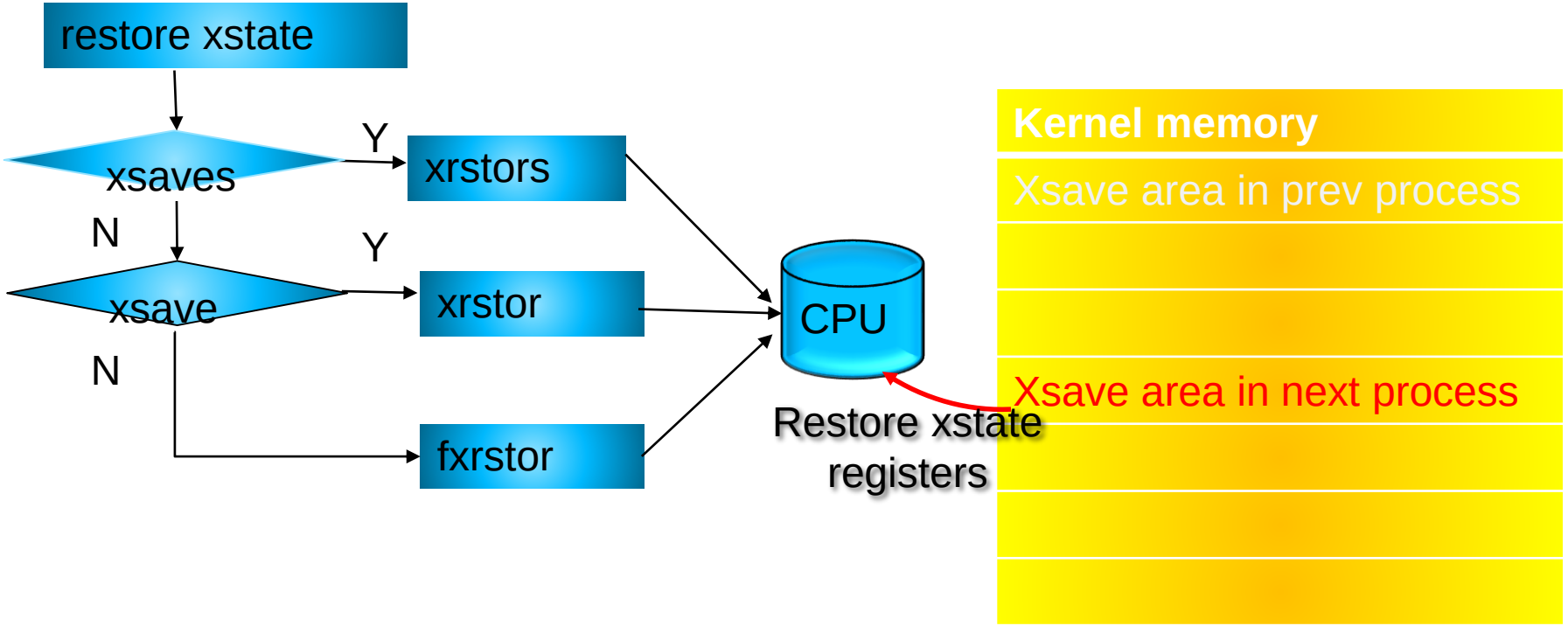
Agenda

- **Introduction**
- Impact of xstates in Context Switch
- Context Switch Optimizations for xstates
- Kernel Implementation for xstates Context Switch
- Security Concern and Solution
- Status and Future Work
- Q & A

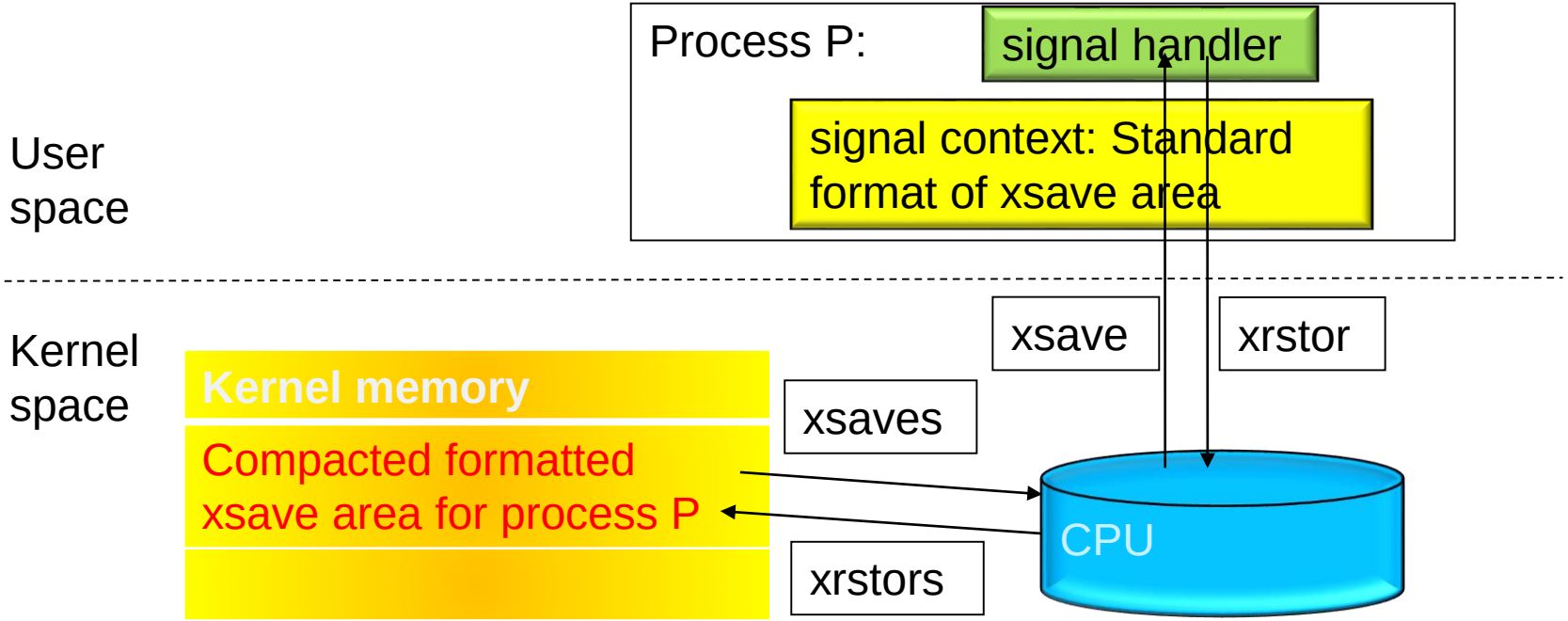
Saving Current Xstates to Previous Process



Loading New Xstates from Next Process



Standard Format of Xsave Area in User Space for Backward Compatibilty with Legacy Applications



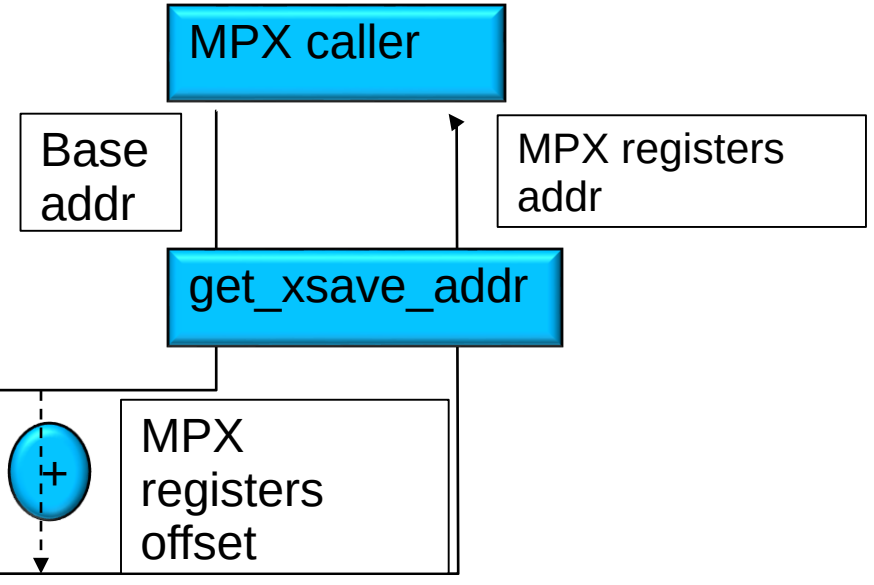
Kernel API for Accessing Registers in Compacted Format of Xsave Area

Scenario: Only FP/SSE/MPX are enabled in processor

Kernel memory

Xsave area for process P

Legacy FP State	0
Legacy SSE State	160
Xsave Header Data	512
MPX_BNDREGS	832
MPX_BNDCSR	896

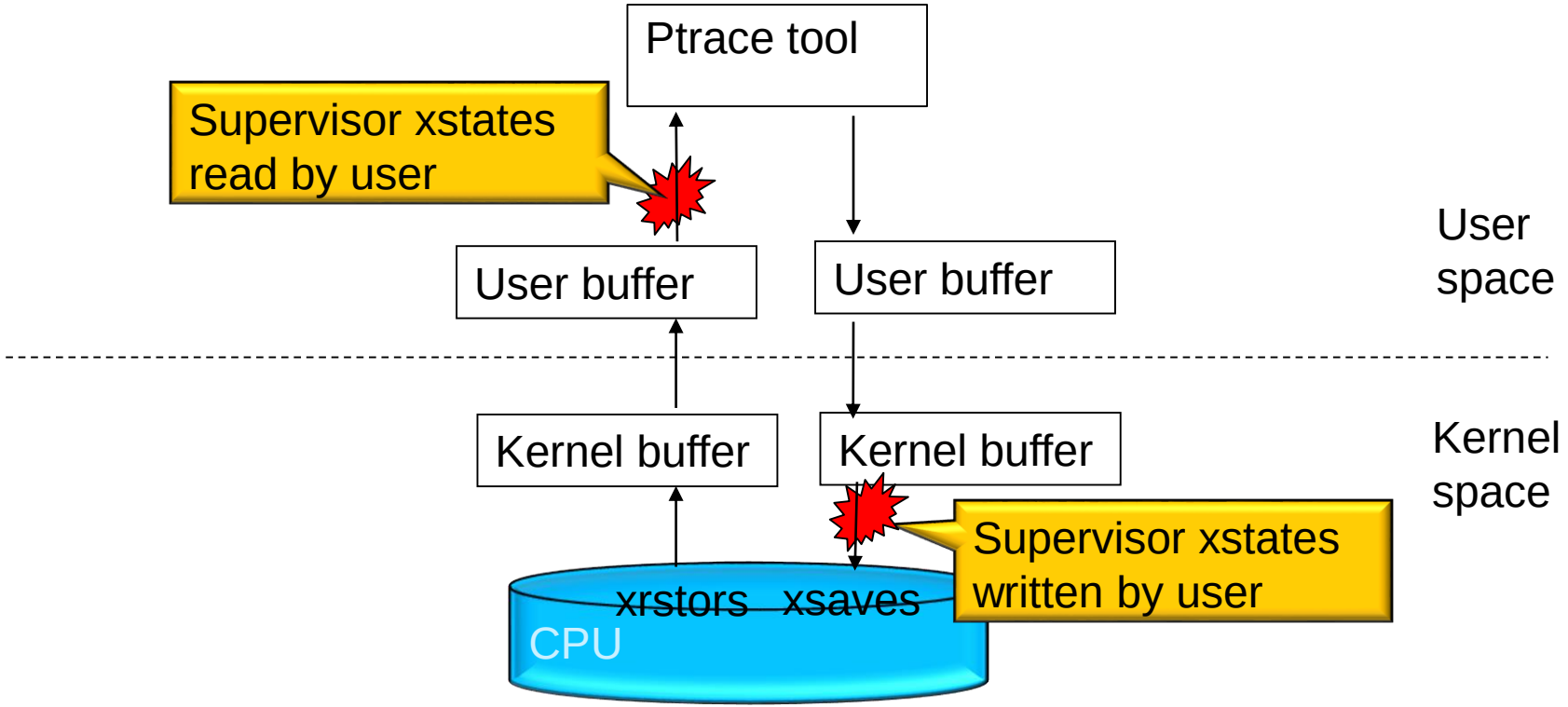


All offsets in xsave area are calculated during kernel boot from cupid(eax=0x0d, ecx=n, n>1)

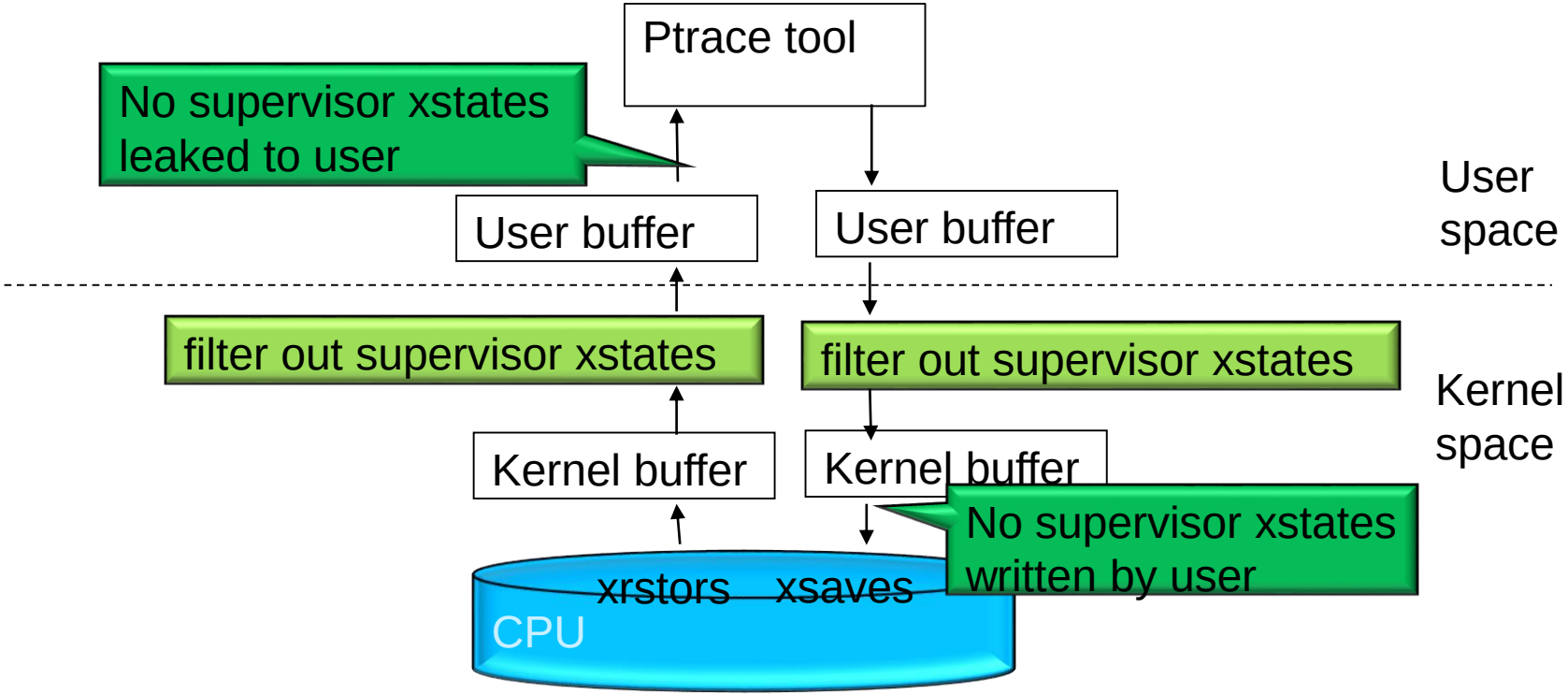
Agenda

- Introduction
- Impact of xstates in Context Switch
- Context Switch Optimizations for xstates
- Kernel Implementation for xstates Context Switch
- Security Concern and Solution
- Status and Future Work
- Q & A

Potential Security Concern for Supervisor States



Solution for Security Concern for Supervisor States



Agenda

- Introduction
- Impact of xstates in Context Switch
- Context Switch Optimizations for xstates
- Kernel Implementation for xstates Context Switch
- Security Concern and Solution
- Status and Future Work
- Q & A

Patches Status

Instructions	Kernel
fxsave/ fxrstor	In 3.16 or earlier version
xsave/ xrstor	
xsaveopt/ xrstor	
xsavec/ xrstor	Not implemented
xsaves/ xrstors	In 3.17

Future Work

- Init optimization for xrstor/xrstors in kernel.
  - Init optimization for xsaveopt/xsavec/xsaves is implemented in processor
- Enable supervisor xstates once hardware implementation is available.
  - Currently there is no supervisor xstate implemented yet.
- Performance improvement measurement

Acknowledgements

Asit Mallick, H. Peter Anvin, Glenn Williamson, Bruce Schlobohm

(Intel SSG/OTC)

References

[1] Intel 64 and IA-32 Architectures Software Developer's Manual (Volume 1, 2, 3)

[2] Intel® Architecture Instruction Set Extensions Programming Reference

[3] Linux Kernel Source Tree.

Q & A