Intel MPX Explained

A Cross-layer Analysis of the Intel MPX System Stack

intel-mpx.github.io

Oleksii Oleksenko,
Dmitrii Kuvaiskii, Christof Fetzer

Pascal Felber

Pramod Bhatotia

Université de Neuchâtel

TECHNISCHE UNIVERSITÄT DRESDEN

THE UNIVERSITY of EDINBURGH

Memory error:

an access to an unintended memory region

Spatial errors

Unintended address

E.g., buffer overflow, stack overflow

Temporal errors

Unintended time

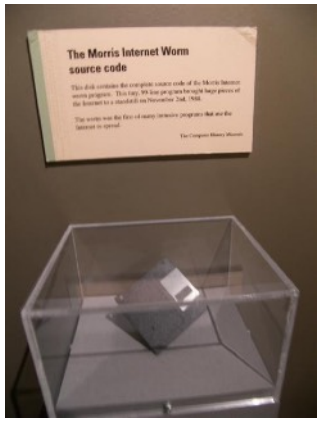
E.g., double free, dangling pointers

Memory errors: a major threat

Felix Wilhelm
@_fel1x

Follow

ISC patched two interesting bugs in their DHCP codebase: A global buffer overflow triggerable over DHCPv6 and a refcount overflow -> use-after-free bug in their option parsing: lists.isc.org/pipermail/dhcp … and lists.isc.org/pipermail/dhcp …



hanno
@hanno

Follow

Stack buffer overflow in WolfSSL before 3.13.0 blog.fuzzing-project.org/63-Stack-buffe …

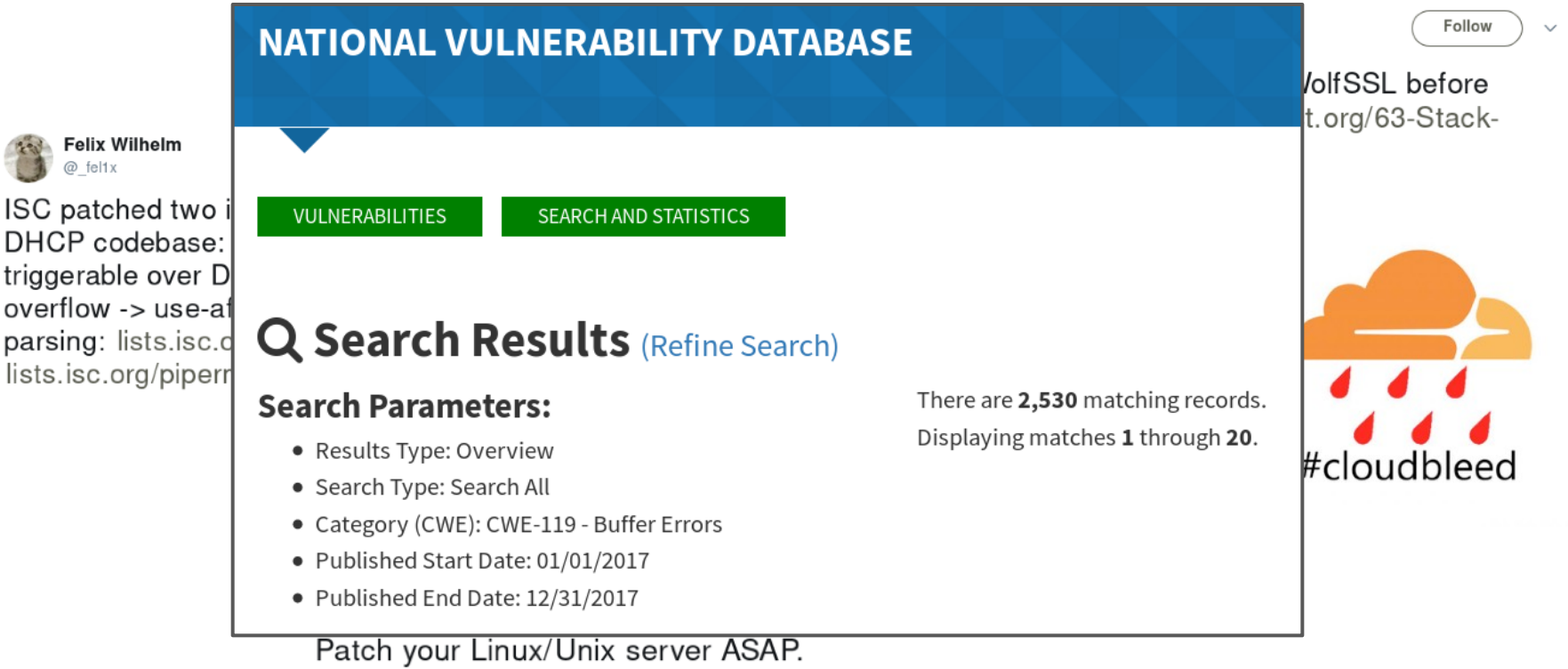
3:49 PM - 24 Mar 2018

nixCraft
@nixcraft

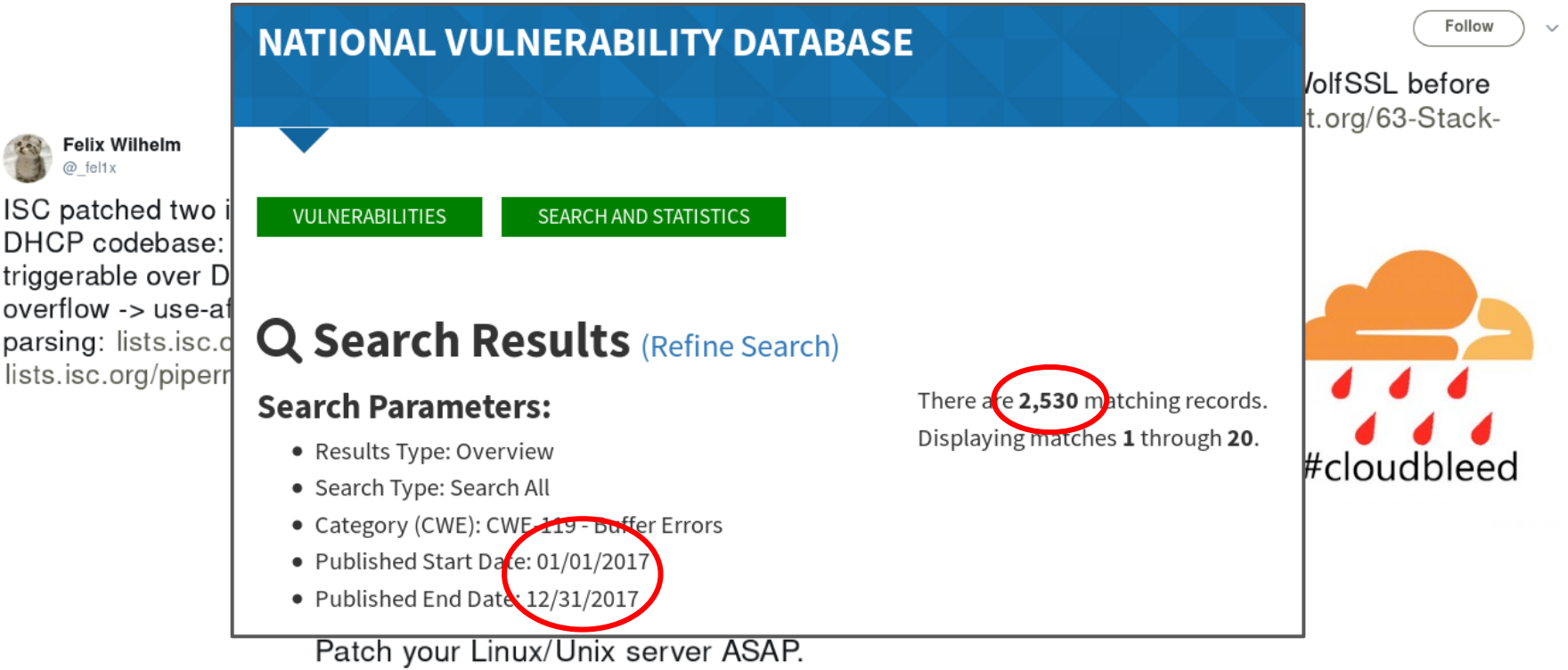
Follow

ALL versions of Exim MTA affected by buffer overflow vulnerability that allows an attacker to run code remotely (CVE-2018-6789). Patch your Linux/Unix server ASAP.

Memory errors: a major threat

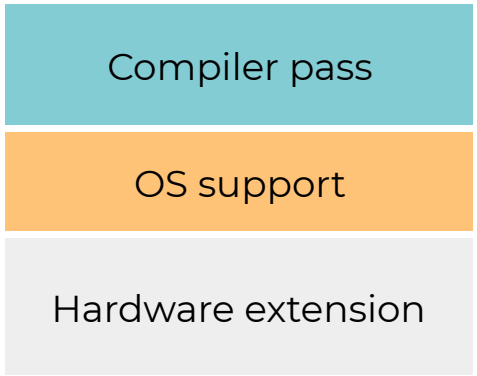


Memory errors: a major threat



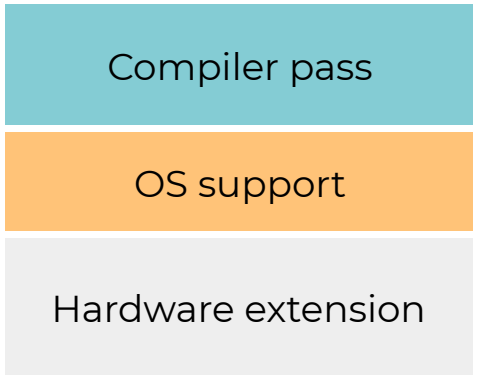
Intel MPX

Hardware-assisted memory protection



Intel MPX

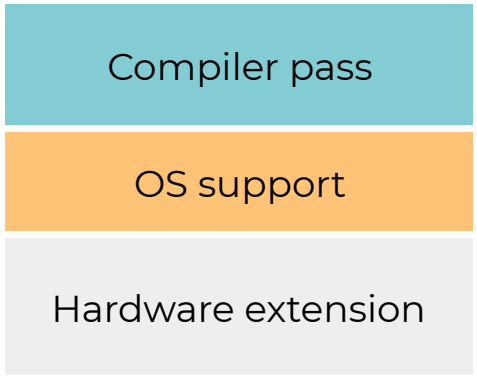
Hardware-assisted memory protection



- New instructions: check safety
- New registers: store metadata

Intel MPX

Hardware-assisted memory protection



- Manage memory and handle errors

- New instructions: check safety
- New registers: store metadata

Intel MPX

Hardware-assisted memory protection

Compiler pass	• Add the new instructions • Runtime support
OS support	• Manage memory and handle errors
Hardware extension	• New instructions: check safety • New registers: store metadata

Intel MPX

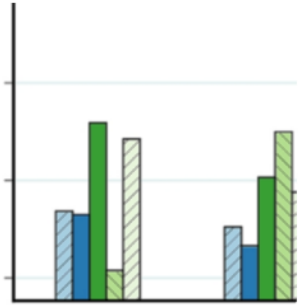
- A ready solution
  - Available in recent CPUs
  - Supported by major compilers (GCC, ICC)
- And yet, not adopted in practice

Our study:

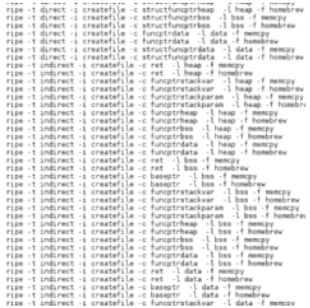
What went wrong?

What caused the issues?
What can we learn from it?

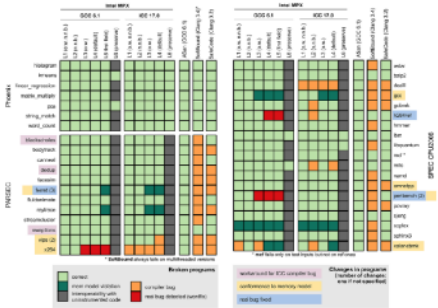
A brief overview



Performance



Security



Usability

Details: http://intel-mpx.github.io

Performance

Approach	Average Slowdown		
	PARSEC	SPEC	Phoenix
MPX (ICC version)	25 %	61 %	56 %
AddressSanitizer	43 %	62 %	60 %
SAFECode	182 %	129 %	5 %
SoftBound	183 %	103 %	168 %

Performance

Approach	Average Slowdown		
	PARSEC	SPEC	Phoenix
MPX (ICC version)	25 %	61 %	56 %
AddressSanitizer	43 %	62 %	60 %
SAFECode	182 %	129 %	5 %
SoftBound	183 %	103 %	168 %

Not significantly better than SW solutions

Security

- Comparable / better security guarantees
- A few issues
  - multithreading support
  - temporal errors
  - could be fixed in future generations

Usability

- Applications may need modifications
  - Non-standard idioms
  - Ad-hoc memory management

Usability

- Applications may need modifications
  - Non-standard idioms
  - Ad-hoc memory management

Our experience:

Phoenix PARSEC SPEC

No changes required
Minor changes
Mem. model violation

Obstacles to Adoption

- Performance
  - High runtime cost
- Usability
  - Necessary modifications

Not security

(at least, not to a large extent)

Lessons Learned

1. It is cheaper not to save on hardware
    - Parallel checks ⇒ improved performance
    - Bounds cache ⇒ reduced cache contention

Lessons Learned

1. It is cheaper not to save on hardware
    - Parallel checks ⇒ improved performance
    - Bounds cache ⇒ reduced cache contention
2. Protection should be transparent
    - Embedded checks ⇒ fewer application changes
    - Atomic checks ⇒ no multithreading issues

Lessons Learned

1. It is cheaper not to save on hardware
    - Parallel checks ⇒ improved performance
    - Bounds cache ⇒ reduced cache contention
2. Protection should be transparent
    - Embedded checks ⇒ fewer application changes
    - Atomic checks ⇒ no multithreading issues
3. Defence should be complete
    - Temporal protection ⇒ complete security solution

Summary

- MPX: an evolutionary improvement
  - Compromises hindered adoption
- Security is not enough
  - Strive for transparent and low-cost protection
- Realistic solution requires radical redesign

Summary

- MPX: an evolutionary improvement
  - Compromises hindered adoption
- Security is not enough
  - Strive for transparent and low-cost protection
- Realistic solution requires radical redesign

https://intel-mpx.github.io/

https://github.com/tudinfse/intel_mpx_explained

Summary

- MPX: an evolutionary improvement
  - Compromises hindered adoption
- Security is not enough
  - Strive for transparent and low-cost protection
- Realistic solution requires radical redesign

https://intel-mpx.github.io/

https://github.com/tudinfse/intel_mpx_explained

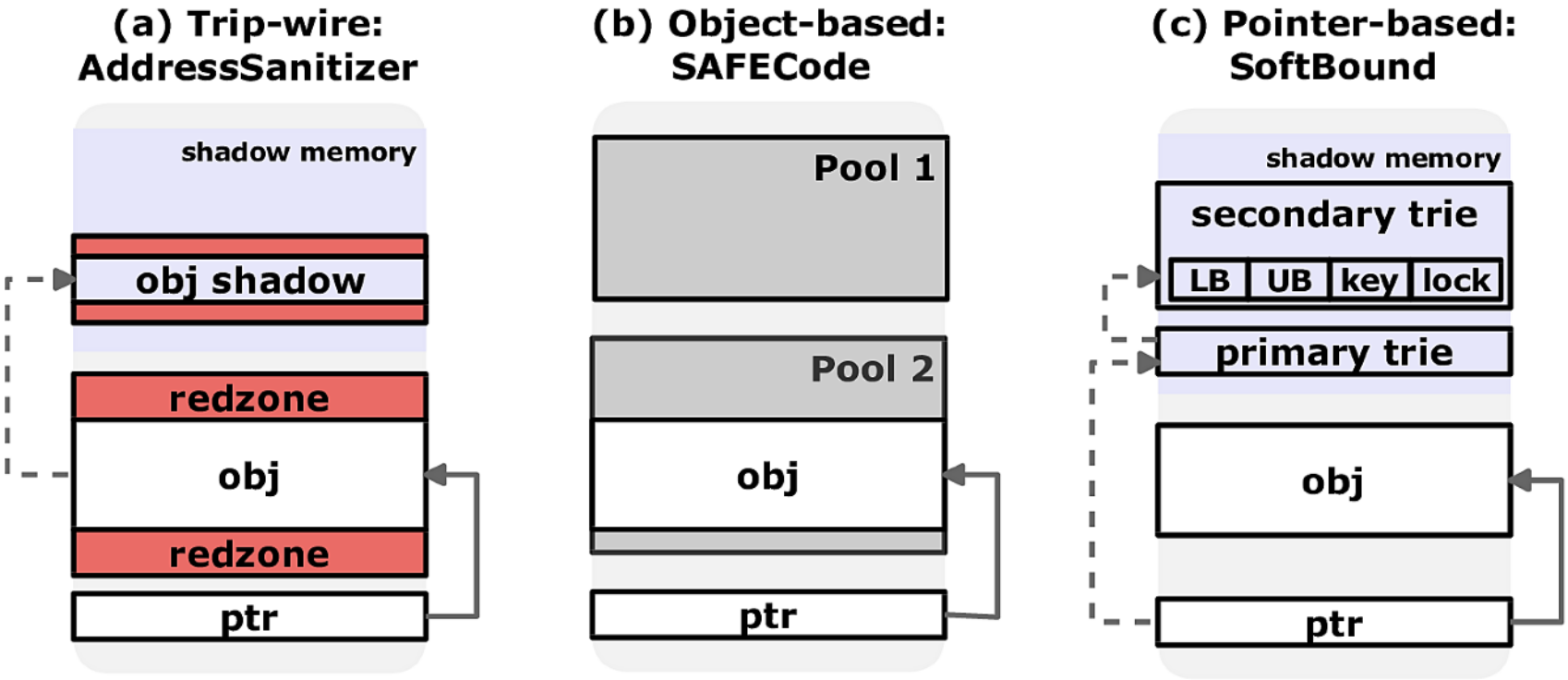
Thanks!

oleksii.oleksenko@tu-dresden.de

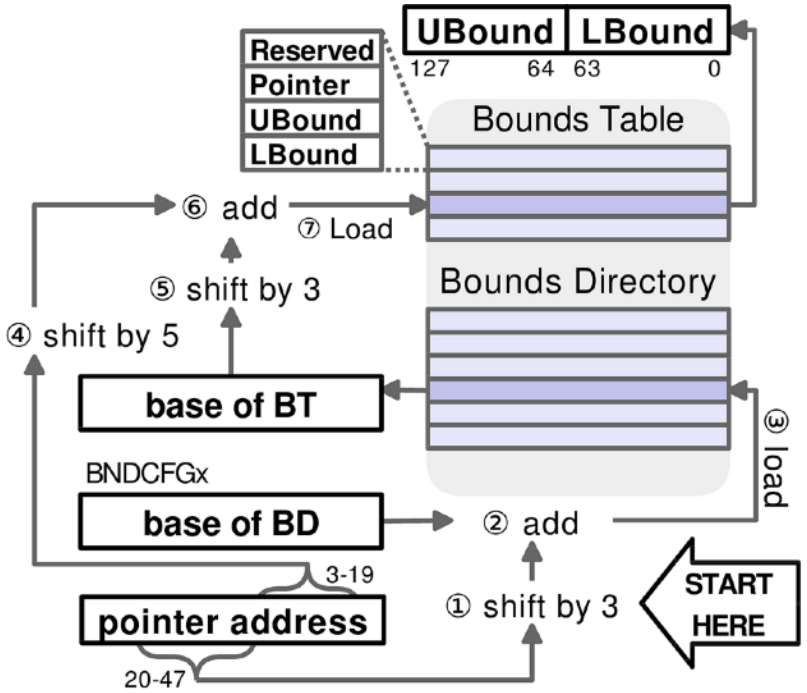
Twitter: @oleksii_o

Backup

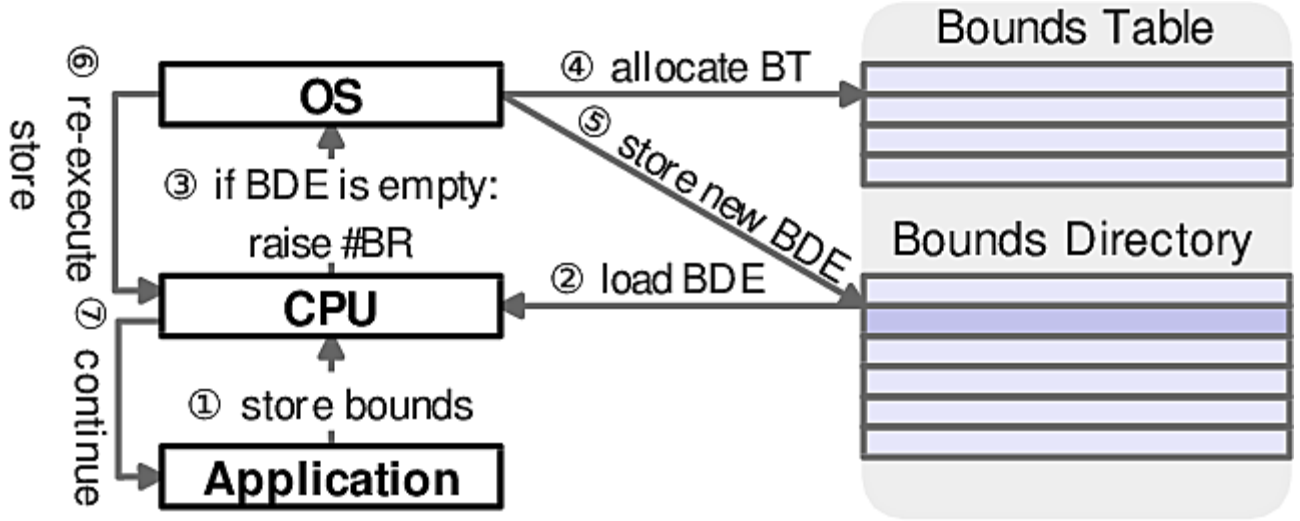
SW approaches



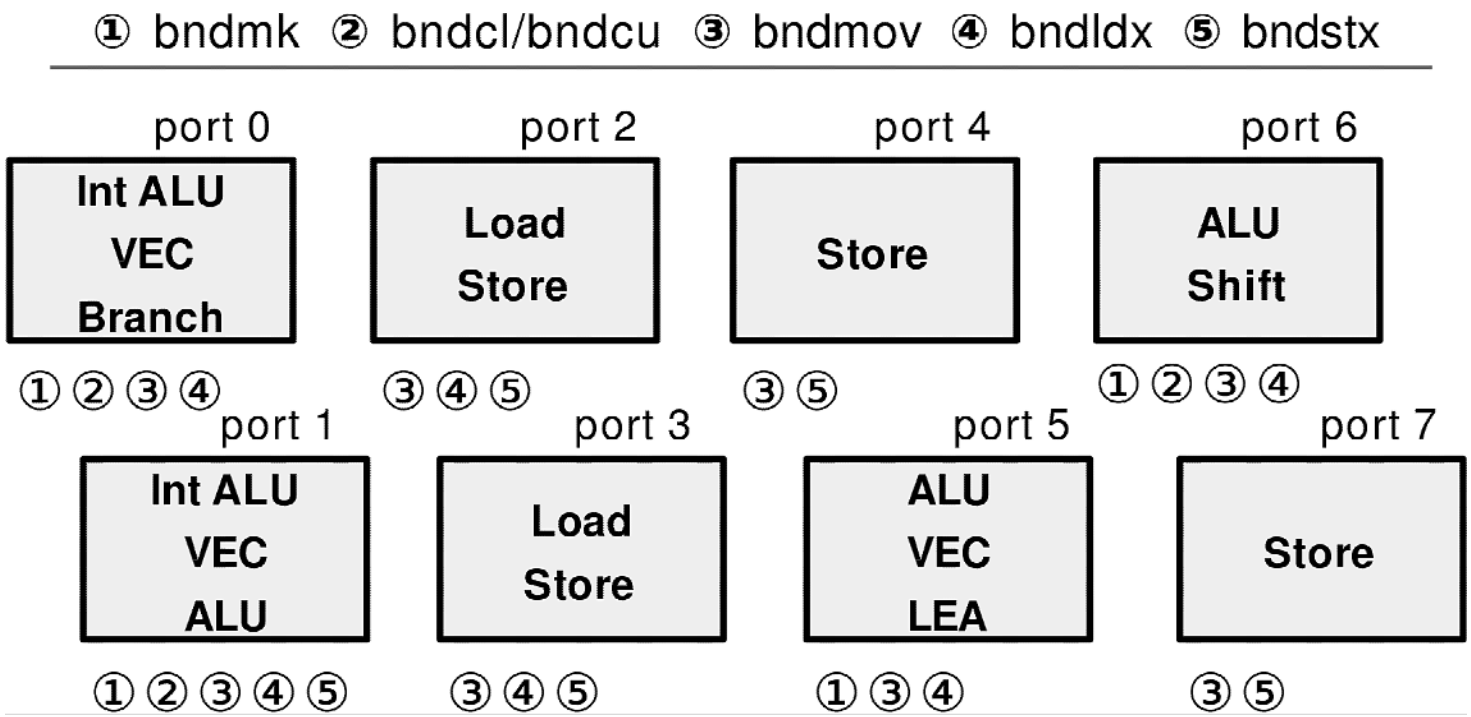
Bound address translation



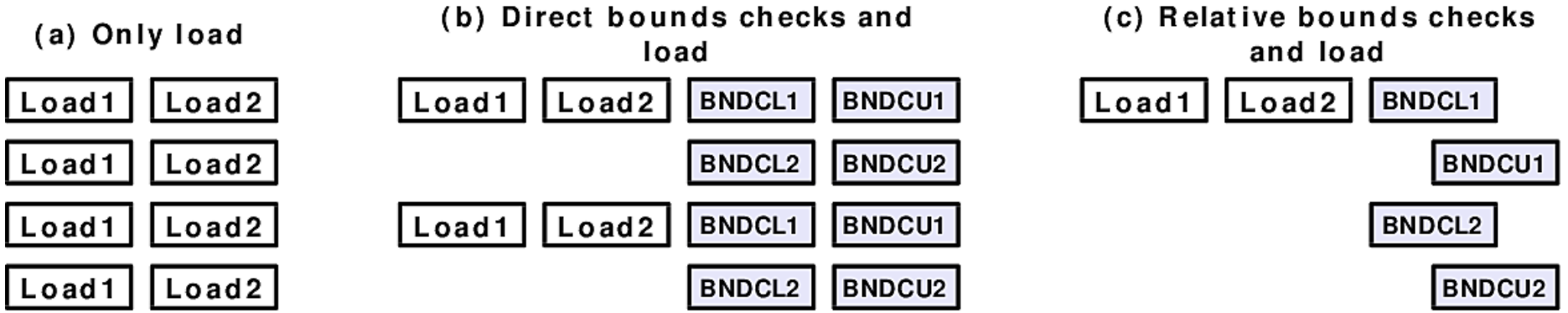
BT allocation



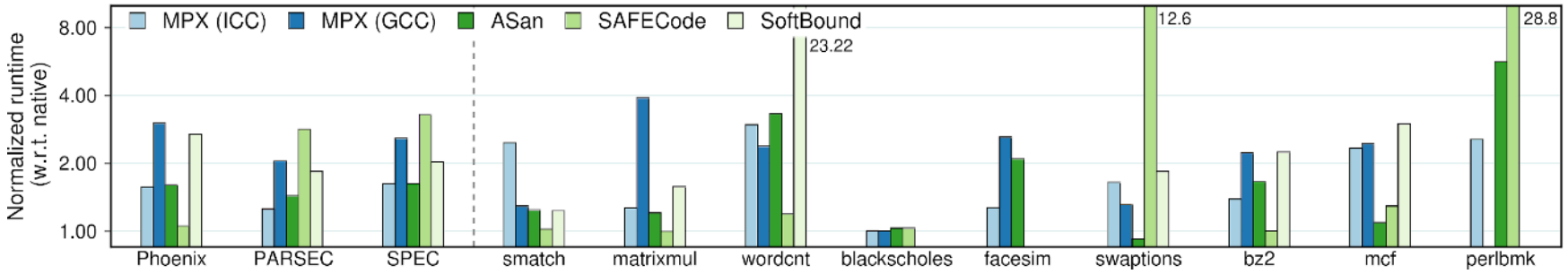
Execution ports



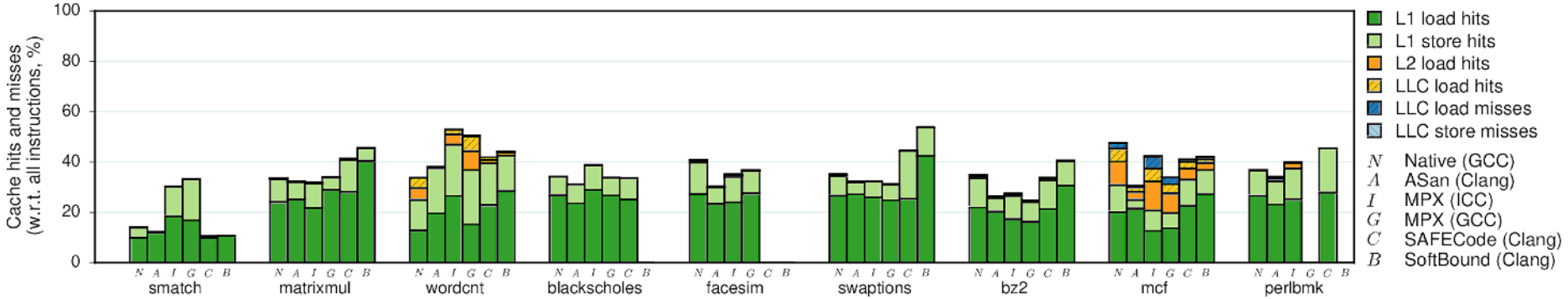
Bounds checking bottleneck



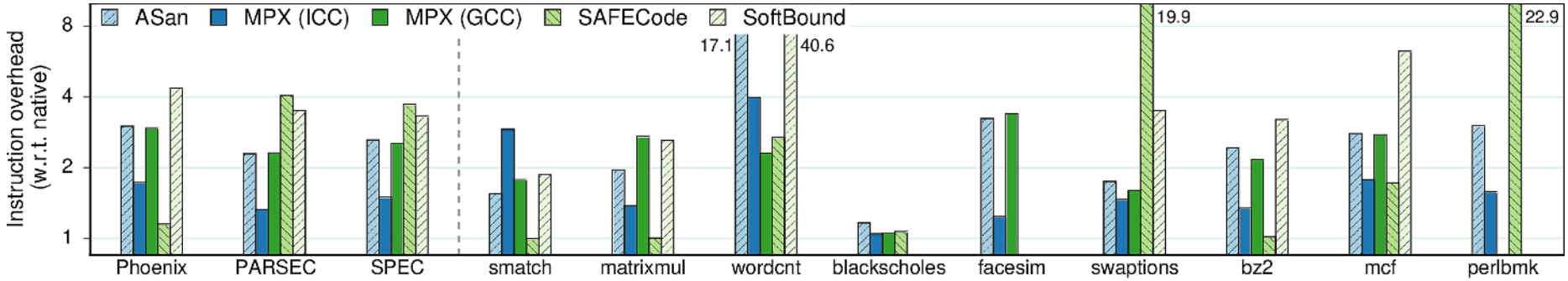
Runtime overhead



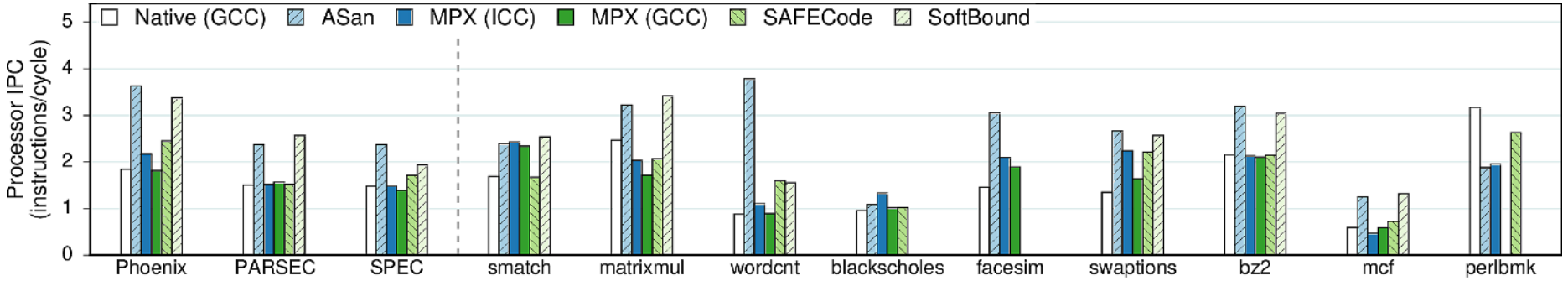
Cache effects



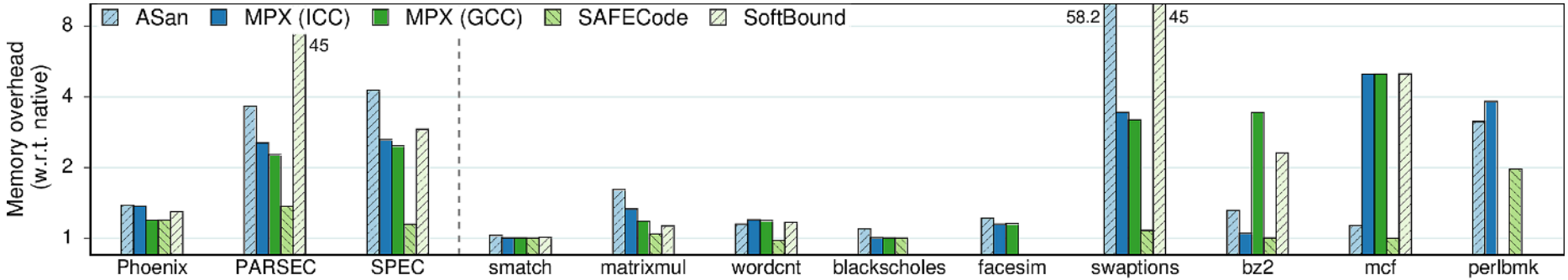
Instruction overheads



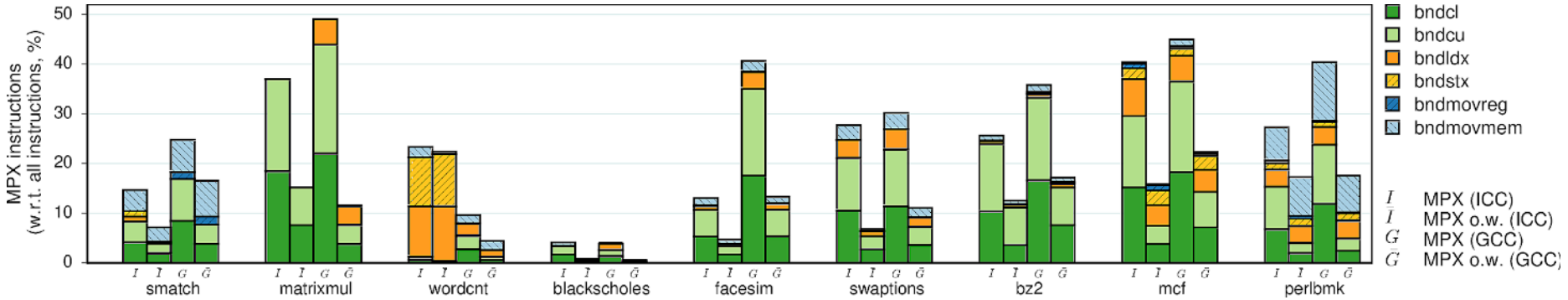
IPC


Native (GCC)
ASan
MPX (ICC)
MPX (GCC)
SAFECode
SoftBound
Processor IPC (instructions/cycle)
0
1
2
3
4
5
Phoenix
PARSEC
SPEC
smatch
matrixmul
wordcnt
blackscholes
facesim
swaptions
bz2
mcf
perlbmk

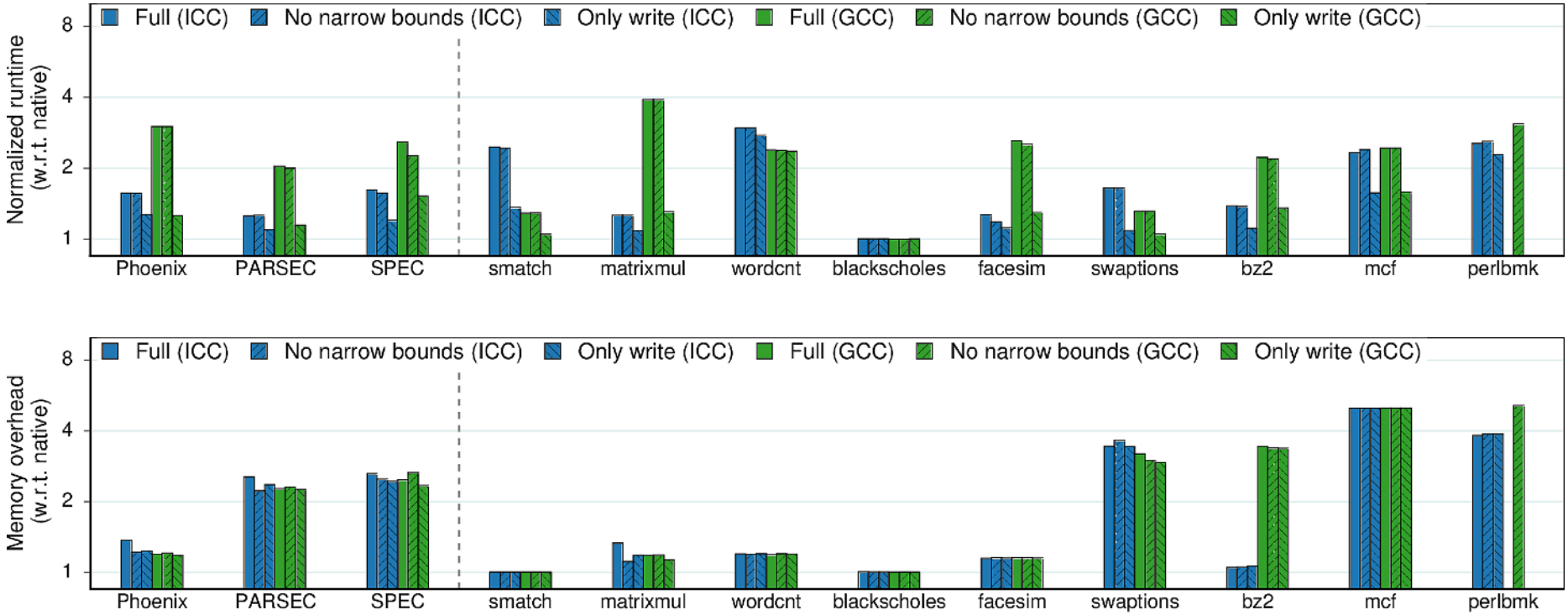
Memory overheads



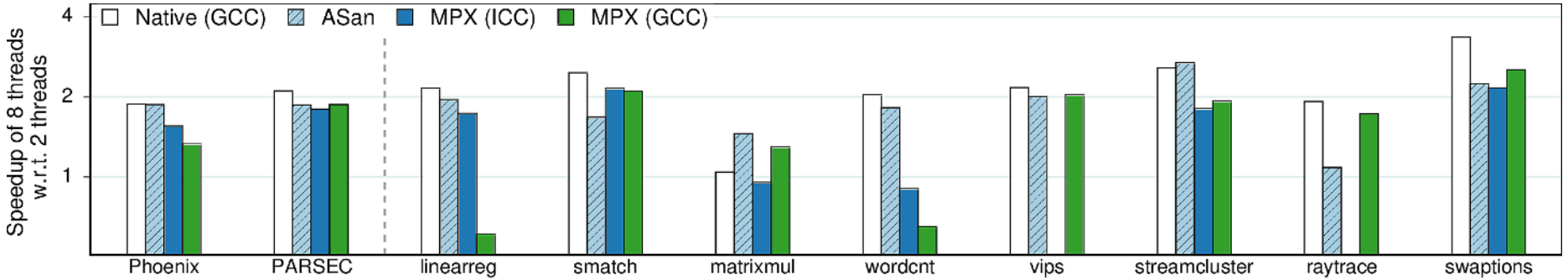
MPX instructions



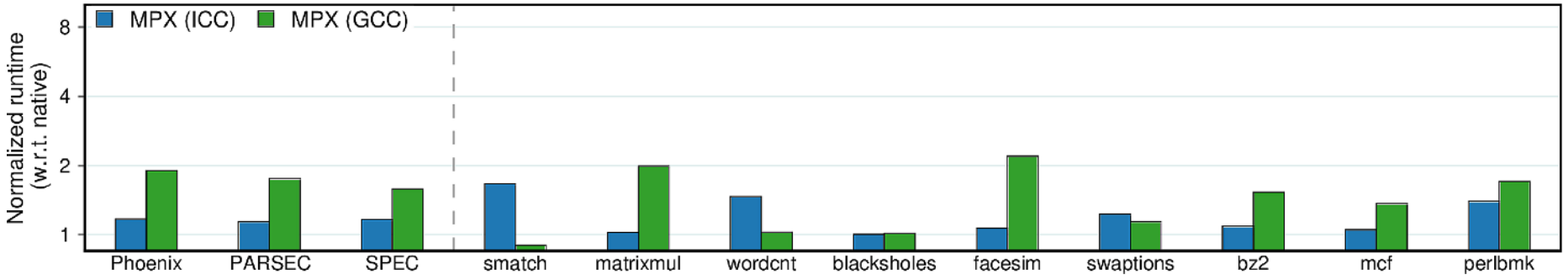
MPX features



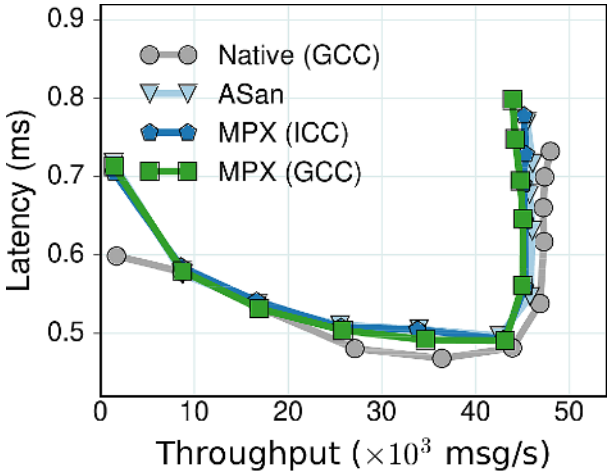
Multithreading

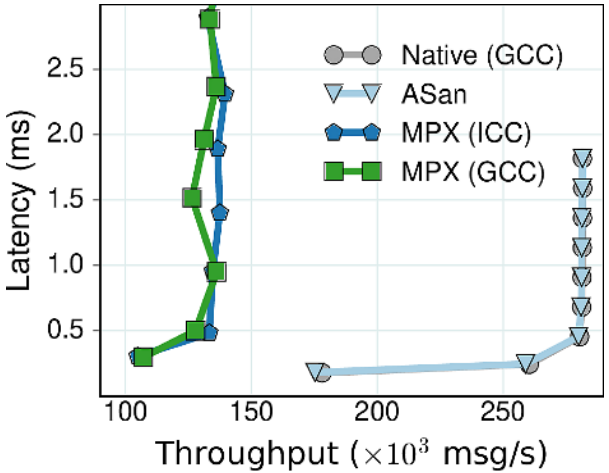


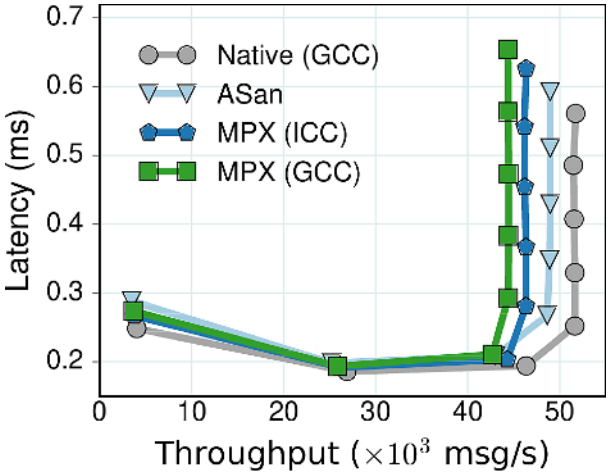
Performance (Haswell)



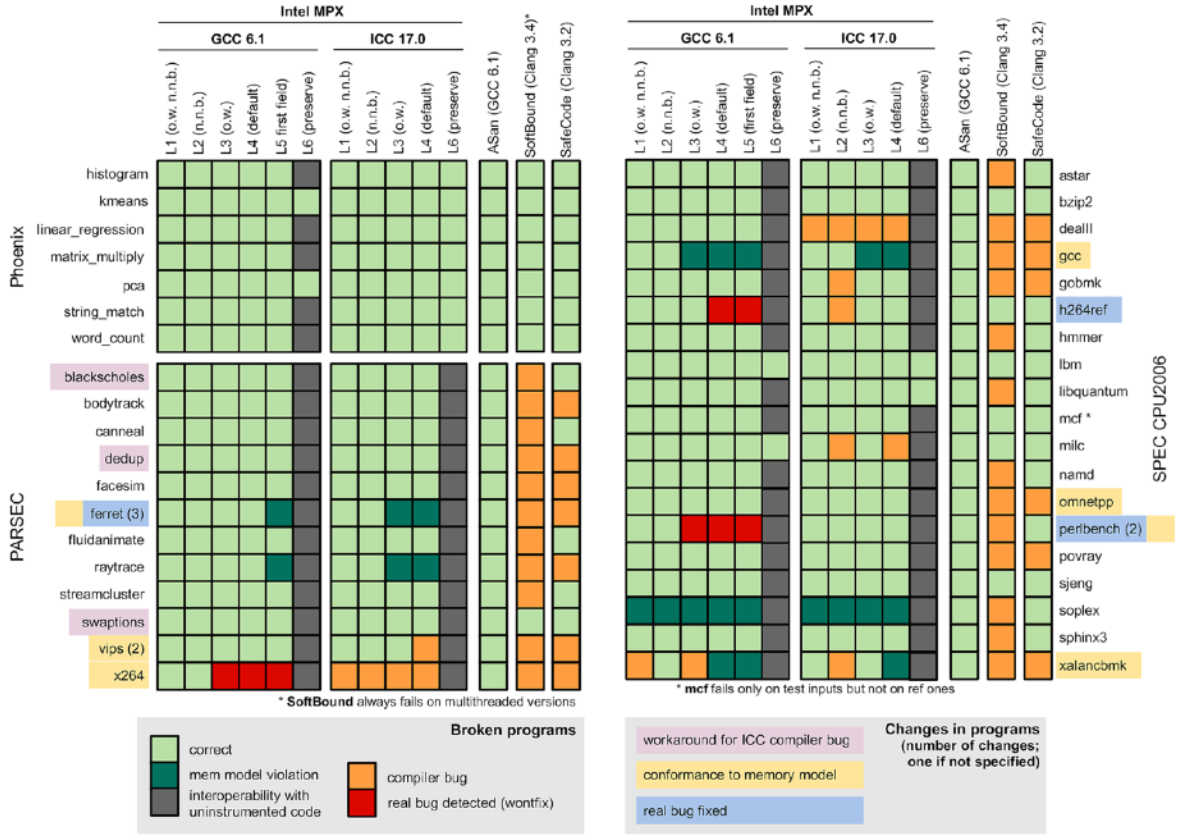
Case studies: Apache, Memcached, Nginx







Usability


Intel MPX
GCC 6.1
ICC 17.0
L1 (o.w. n.n.b.)
L2 (n.n.b.)
L3 (o.w.)
L4 (default)
L5 first field)
L6 (preserve)
ASan (GCC 6.1)
SoftBound (Clang 3.4)*
SafeCode (Clang 3.2)
Phoenix
histogram
kmeans
linear_regression
matrix_multiply
pca
string_match
word_count
PARSEC
blackscholes
bodytrack
canneal
dedup
facesim
ferret (3)
fluidanimate
raytrace
streamcluster
swaptions
vips (2)
x264
* SoftBound always fails on multithreaded versions
SoftBound (Clang 3.4)
SafeCode (Clang 3.2)
astar
bzip2
dealII
gcc
gobmk
h264ref
hmmer
lbm
libquantum
mcf *
milc
namd
omnetpp
perlbench (2)
povray
sjeng
soplex
sphinx3
xalancbmk
SPEC CPU2006
* mcf fails only on test inputs but not on ref ones
Broken programs
correct
mem model violation
interoperability with uninstrumented code
compiler bug
real bug detected (wontfix)
workaround for ICC compiler bug
conformance to memory model
real bug fixed
Changes in programs (number of changes; one if not specified)