

# SPECTRES, VIRTUAL GHOSTS, AND HARDWARE SUPPORT

---

Xiaowan Dong

Zhuojia Shen

John Criswell

Alan Cox

Sandhya Dwarkadas

University of Rochester

University of Rochester

University of Rochester

Rice University

University of Rochester



# Goal

Analyze the impact of speculation side channel mitigation on mechanisms that protect user data from compromised OS kernels.

# SPECTRES: SPECULATION SIDE CHANNELS

---

# Speculation Side Channels: Spectre and Meltdown

- Speculative execution exacerbates side channels
- Confidential data is speculatively read and exposed via cache side channels

- Example of Spectre

```
if (x < array1_size)
```

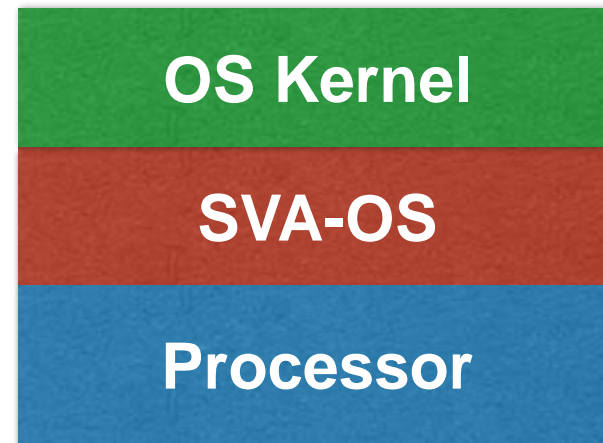
```
y = array2[array1[x] * 256];
```

# VIRTUAL GHOST: PROTECTING APPLICATION DATA FROM COMPROMISED OS KERNELS

---

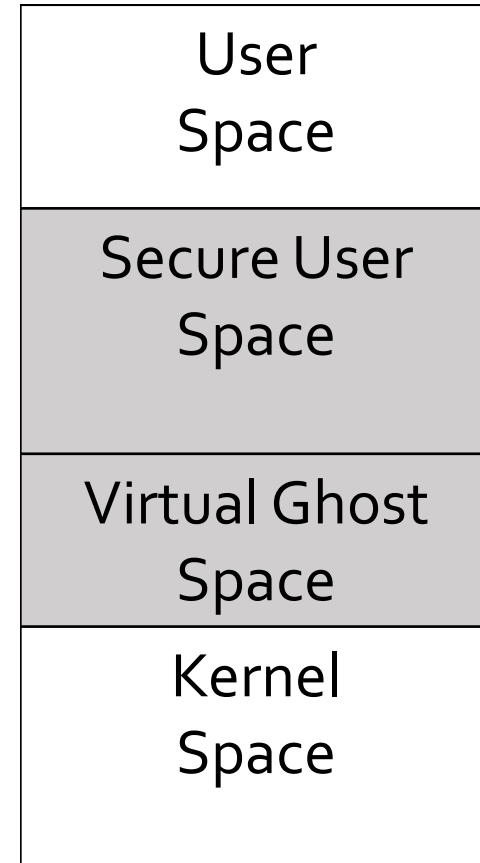
# Virtual Ghost

- A compiler-based approach to protect application data from OS kernels by
  - Instrumenting every kernel load and store: *Software Fault Isolation (SFI)*
  - Requiring the OS kernel to invoke SVA-OS to
    - Manipulate program state (context switch)
    - Configure hardware state (MMU)



# Virtual Address Space of Virtual Ghost

- Protected memory regions include
  - Secure user space
    - Protected from the OS kernel
  - Virtual Ghost space
    - Saves virtual ghost VM internal data structures



Virtual address space

# Virtual Ghost Instrumentation (SFI)

- SFI on Virtual Ghost could be vulnerable to bounds check bypass
- A compromised kernel speculatively reads data before the check completes

- Example of SFI

```
If ((addr < SECURE_MEM_START) OR  
(addr > SECURE_MEM_END))  
    access_memory(addr);
```

*Goal: Defend against **bounds check bypass** (Spectre variant 1) directly launched by the kernel on secure user and virtual ghost space*



# THREE MITIGATIONS

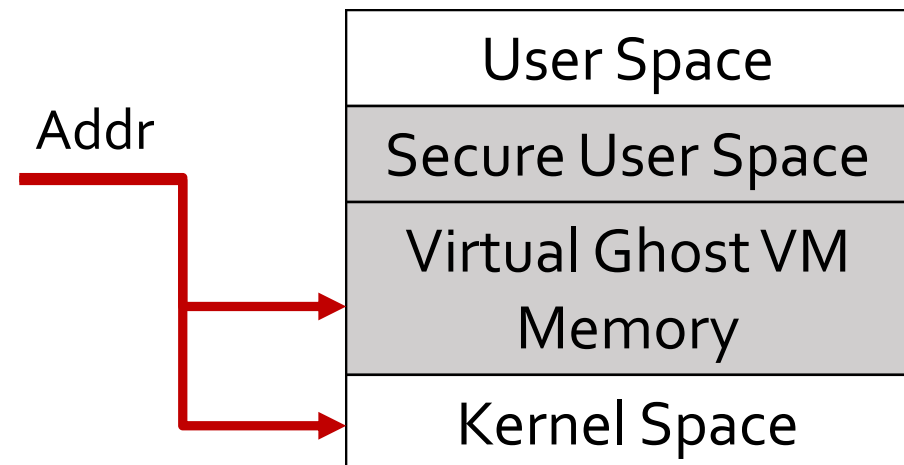
# Three mitigations

- Bit-Masking SFI
- MPX SFI with Ifence
- Separate address spaces

# Bit-Masking SFI

- Check whether a pointer points to protected memory regions
    - Move it into the kernel region if so
    - Cmp and sete instructions
  - Results in a *data dependence* between the SFI code and the memory load
    - Intel does not support *value speculation*
- ➔ Check must complete before memory load is performed

```
mov POINTER_ADDR_HIGH_ORDER_BITS, %R1
cmp PROT_MEM_HIGH_ORDER_BITS, %R1
sete %R2
sign-extend and left shift %R2
or POINTER_ADDR, %R2
mov (%R2), %R1
```

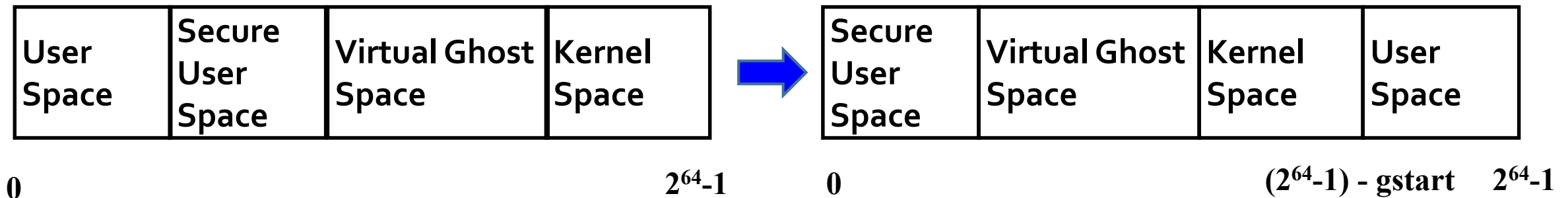


# Three mitigations

- Bit-Masking SFI
- MPX SFI with Ifence
- Separate address spaces


# MPX SFI

- Intel MPX is a hardware mechanism designed for efficient bounds checking
- Limitation: MPX only supports checks against one memory region at a time
- Solution: Make the kernel- and user-space appear contiguous [1]
  - Subtract the start address of secure user space from the pointer address



[1] Xiaowan Dong, Zhuojia Shen, John Criswell, Alan Cox, and Sandhya Dwarkadas. Shielding Software From Privileged Side-Channel Attacks. To appear in the 27th USENIX Security Symposium, 2018.

# Handling Speculation: MPX SFI with lfence

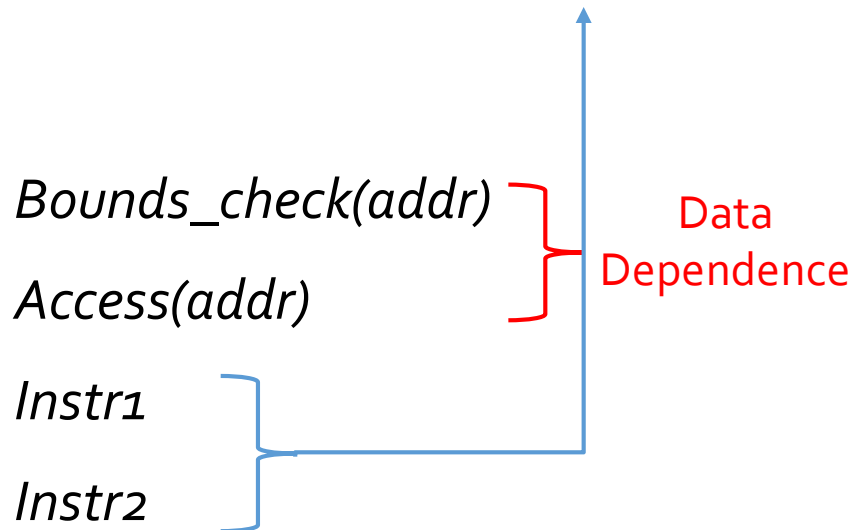
- *Problem*: Intel speculatively executes the instructions following bounds check
    - Assumes the bounds check will pass
  - *Solution*: Insert an **lfence** instruction between the bounds check and the memory load
    - Introduce pipeline stalls
-  All subsequent instructions are stalled until the bounds check commits

```
mov %R1, %R2
sub GHOST_MEM_START_ADDR, %R2
bndcl %R2, %bndo
lfence ←
mov (%R1), %R1
```

# Comparing The Two Approaches

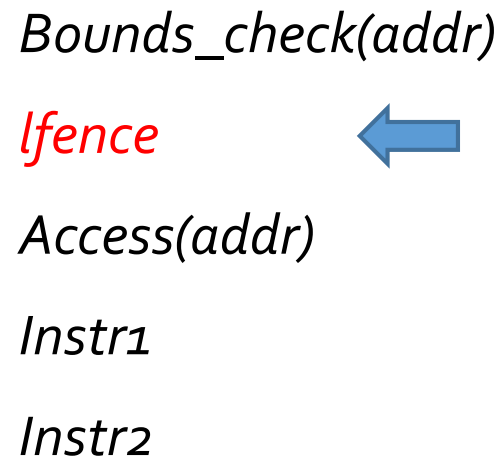
## Bit-Masking SFI

- Only introduces data dependence
- Other instructions can proceed



## MPX SFI with lfence

- Stalls the pipeline completely
- More expensive than Bit-Masking SFI

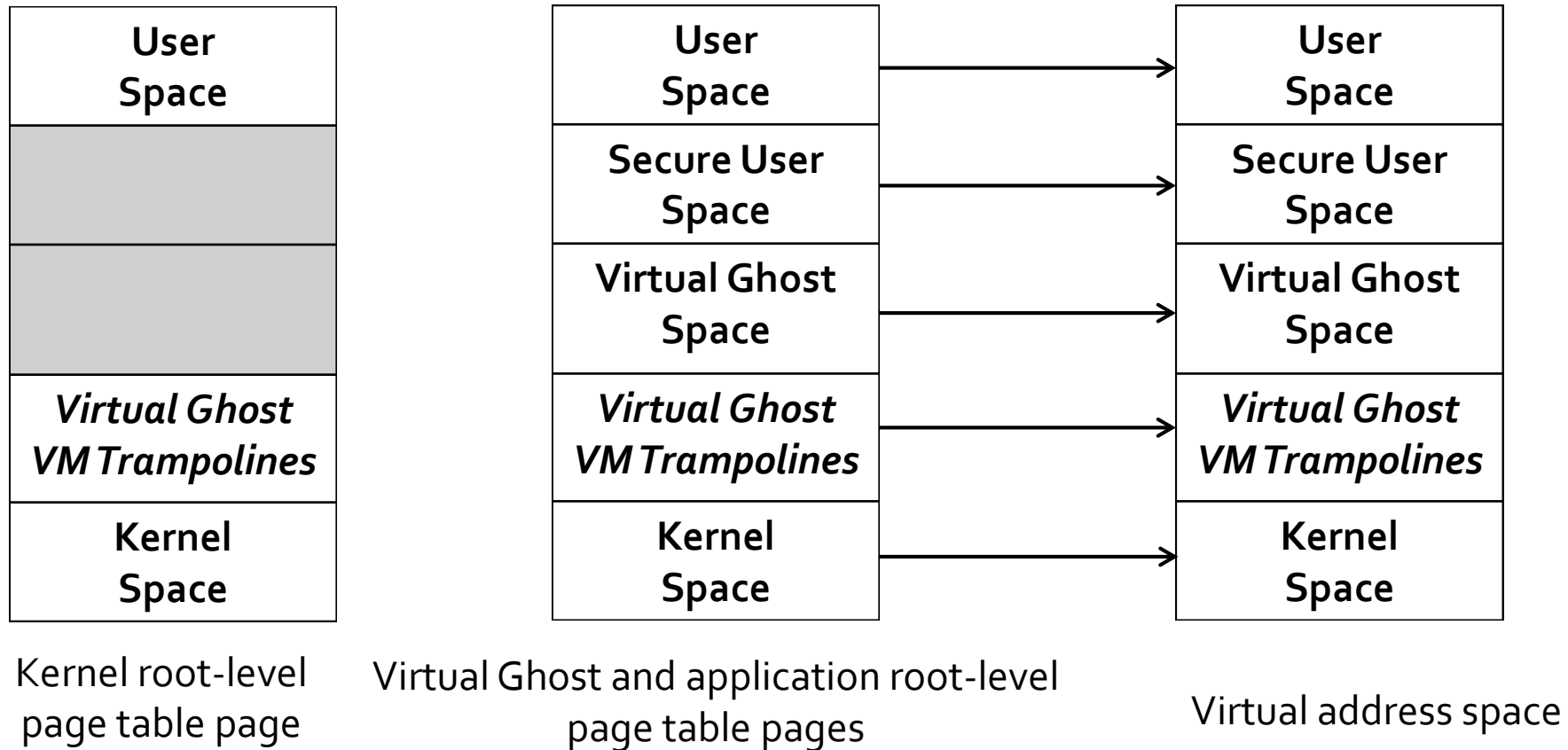


# Three mitigations

- Bit-Masking SFI
- MPX SFI with lfence
- Separate address spaces



# Separate Address Spaces



# Separate Address Spaces

- Challenges: Address space switch can be frequent and expensive
  - Each kernel invocation of SVA-OS incurs two address space switches
    - Kernel-AddrSpace ↔ VirtualGhost-AddrSpace
  - Address space switch incurs execution of serializing instructions
    - Updating the CR3 register with the page table base address

# Comparing The Three Approaches

## Bit-Masking SFI

- Instruments every load and store
- Introduces data dependence

## MPX SFI with Ifence

- Instruments every load and store
- Ifence incurs pipeline stall

## Separate Address Spaces

- Address space switch is very expensive
- Kernel ↔ Virtual Ghost can be frequent

# EVALUATION

---

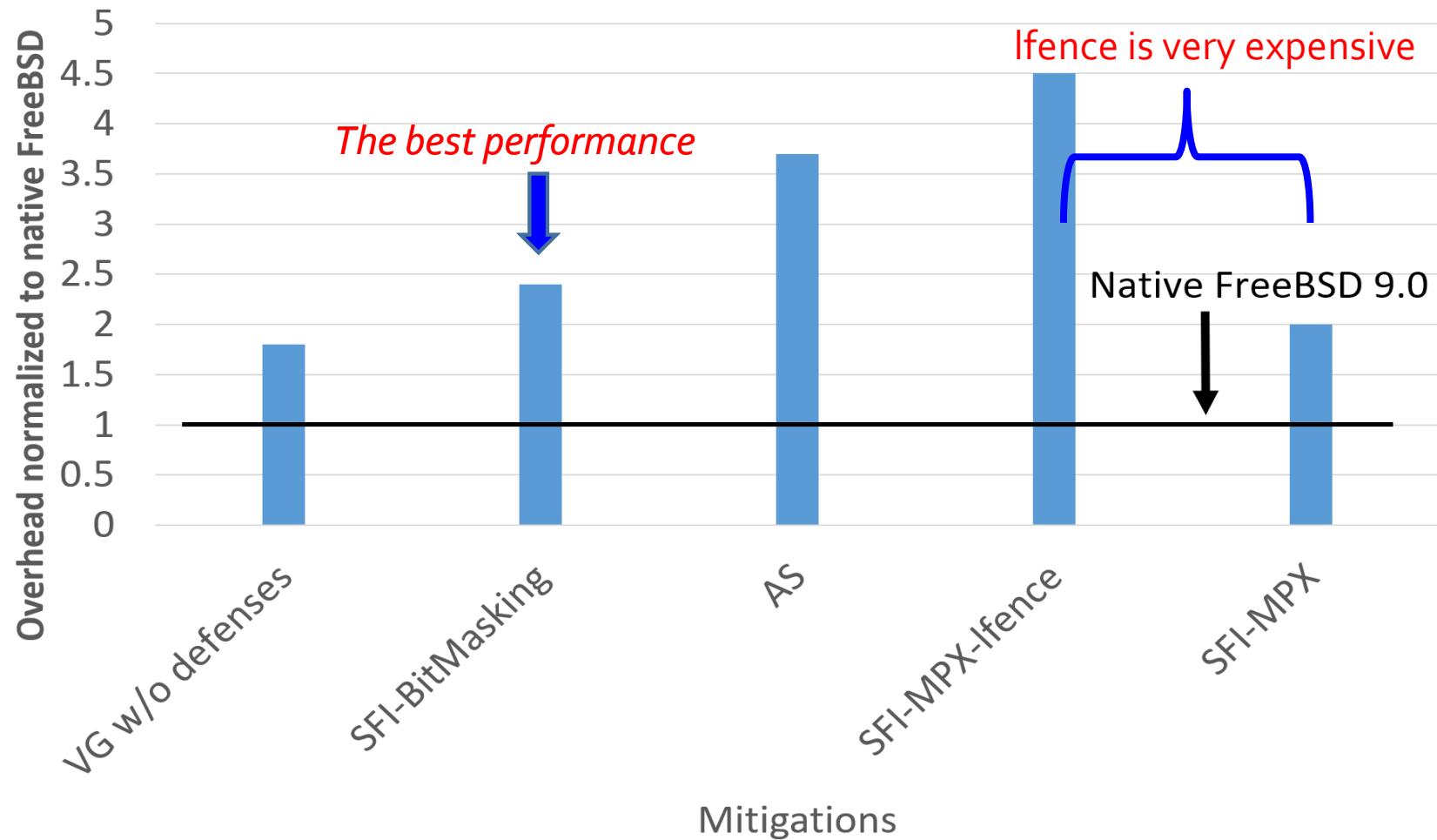
# Methodology

- Experiment environment
  - 3.4GHz Intel i7-6700 hyperthreading quad-core processor
  - 16 GB RAM
  - 256 GB SSD
  - FreeBSD 9.0 ported to Virtual Ghost

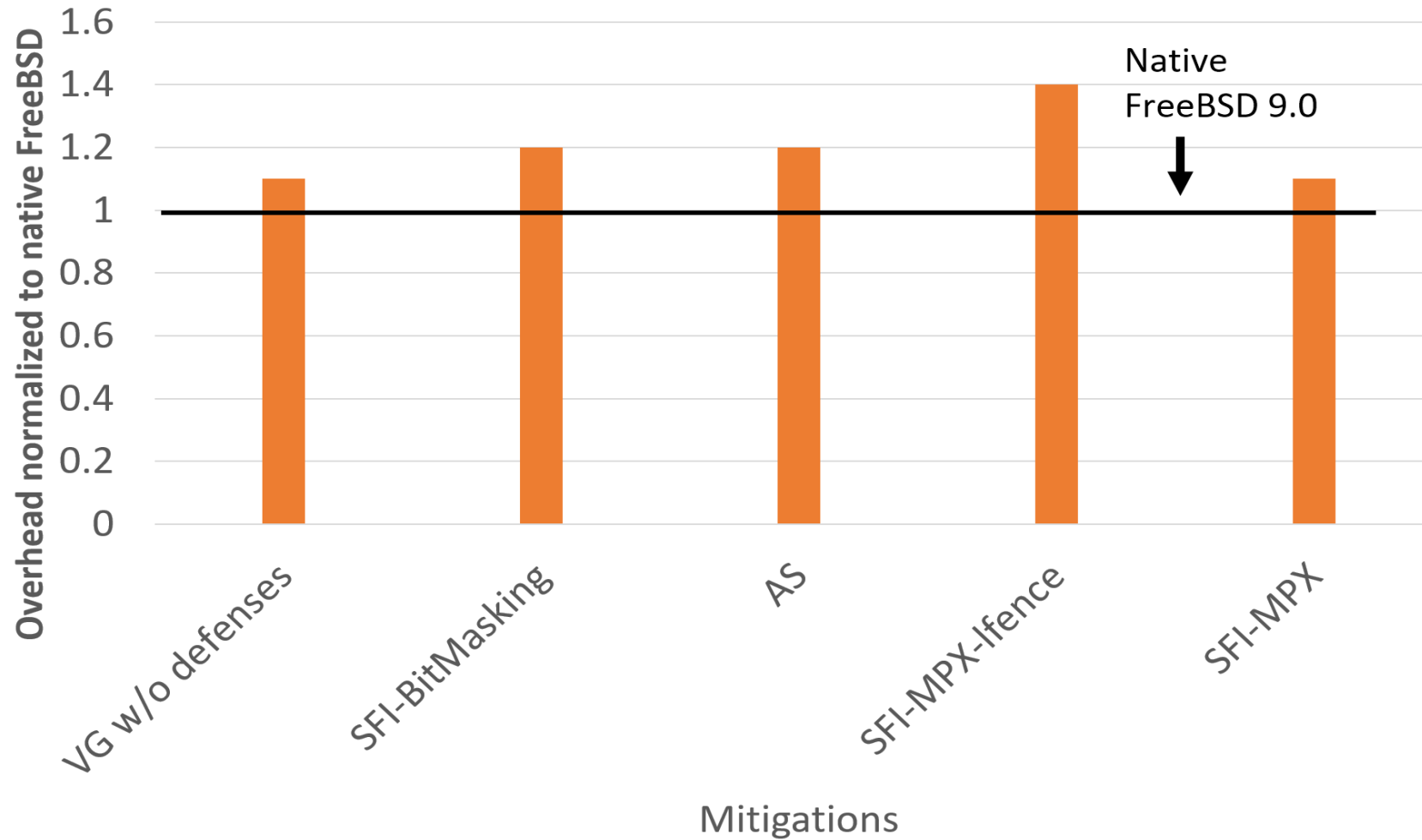
# Benchmarks

- Basic OS operation Benchmarks
  - LMBench
    - Null syscall, open/close, mmap, page fault, fork, fork + exec
- Applications
  - FreeBSD 9.0 C library compilation
  - SSHD
    - On an isolated Gigabit Ethernet network
    - Use *scp* to transfer files from the OpenSSH server to the client

# LMBench Results

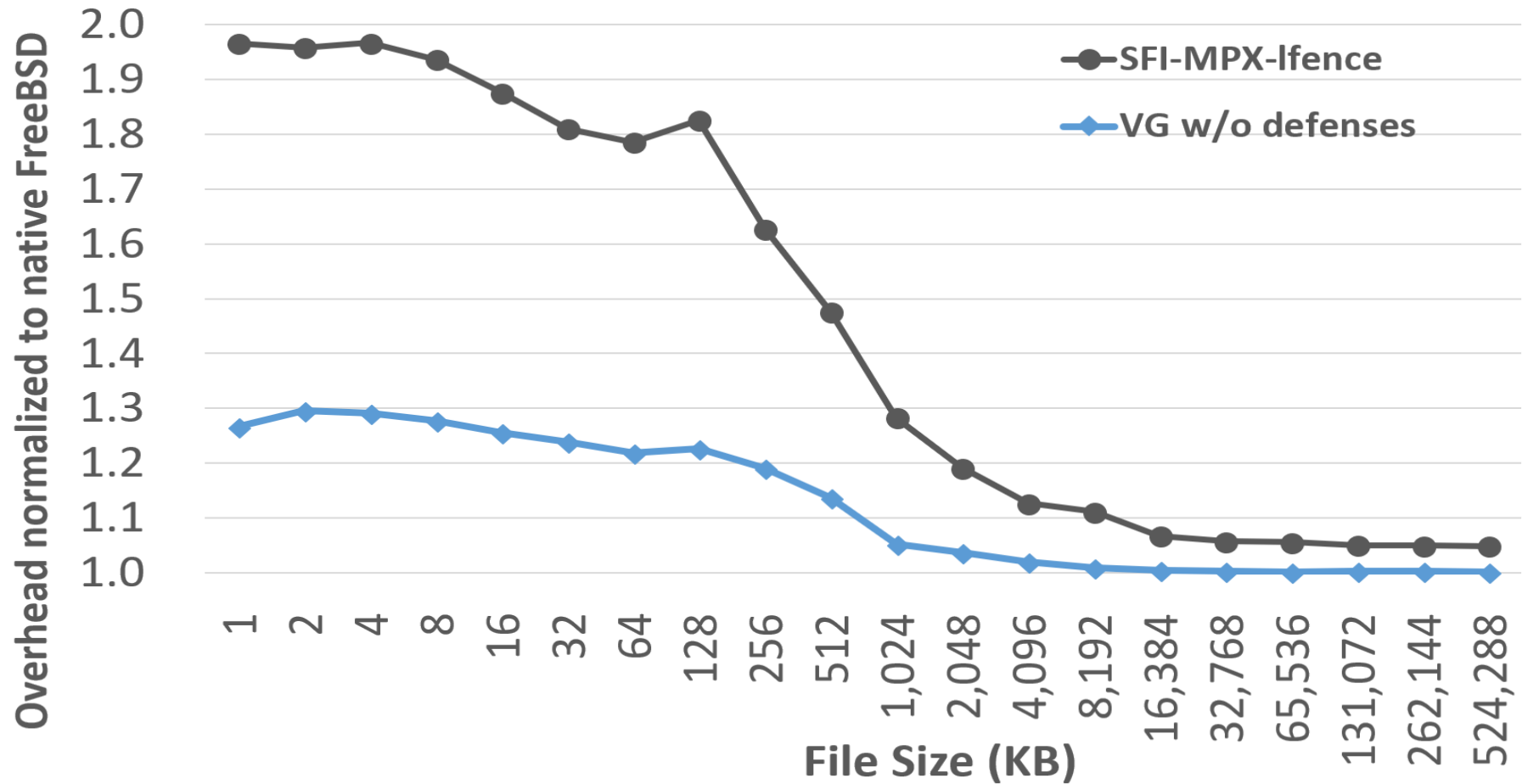


# Libc Compilation Results

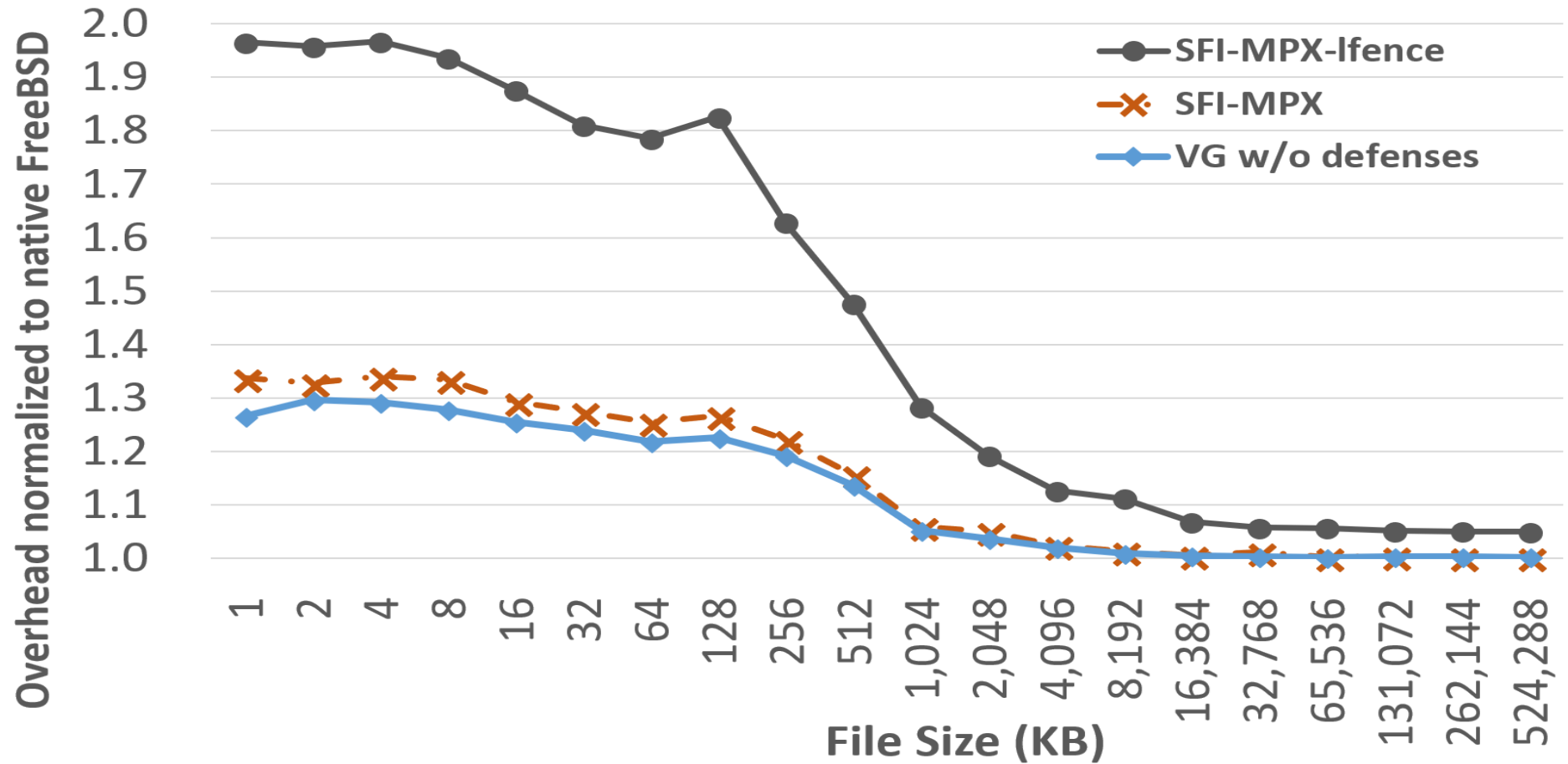




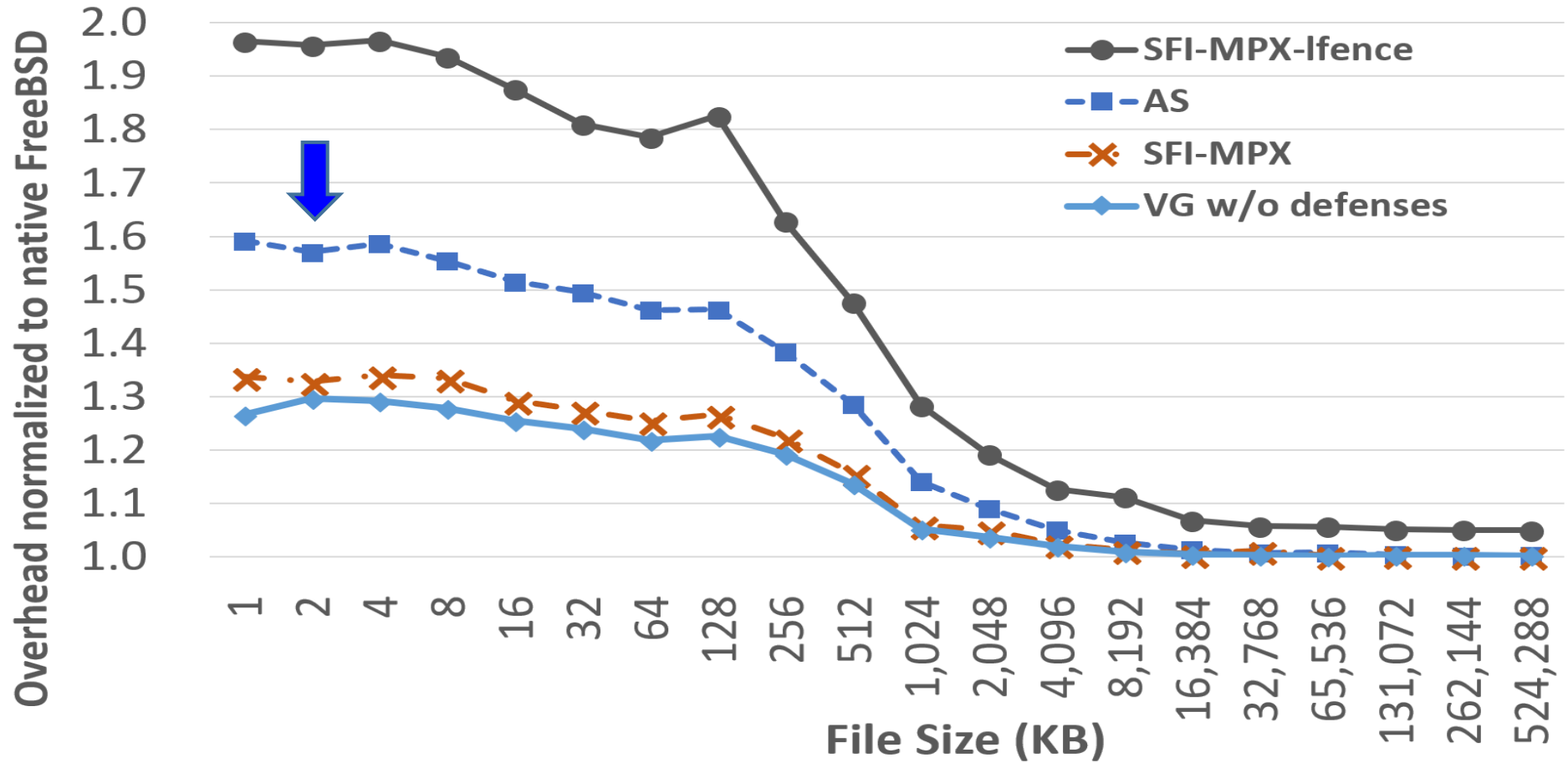
# SSHD File Transfer Rates



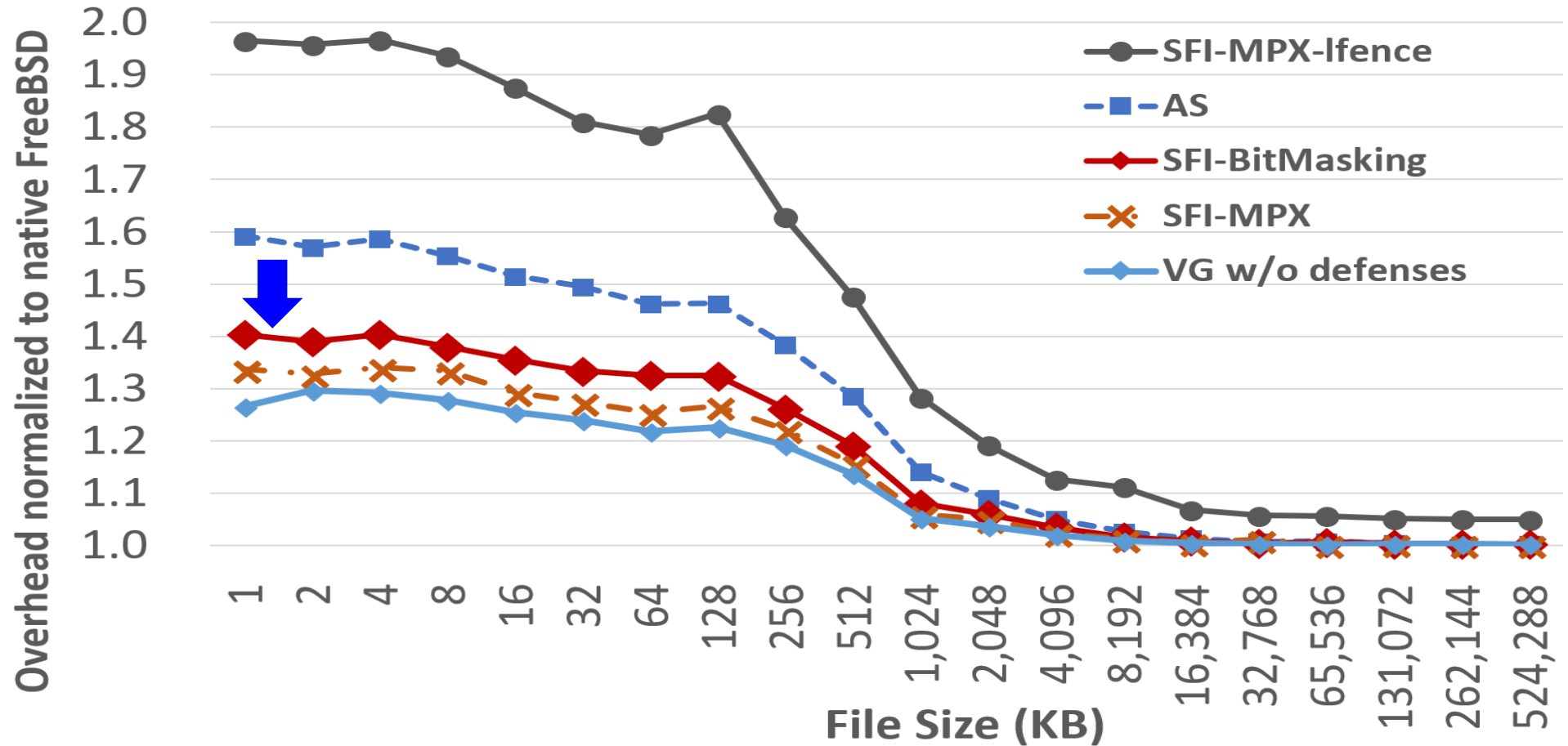
# SSH File Transfer Rates



# SSHD File Transfer Rates



# SSHD File Transfer Rates



- MPX-SFI without Ifence is faster than bit-masking SFI
- Can we eliminate the use of Ifence on MPX-SFI?

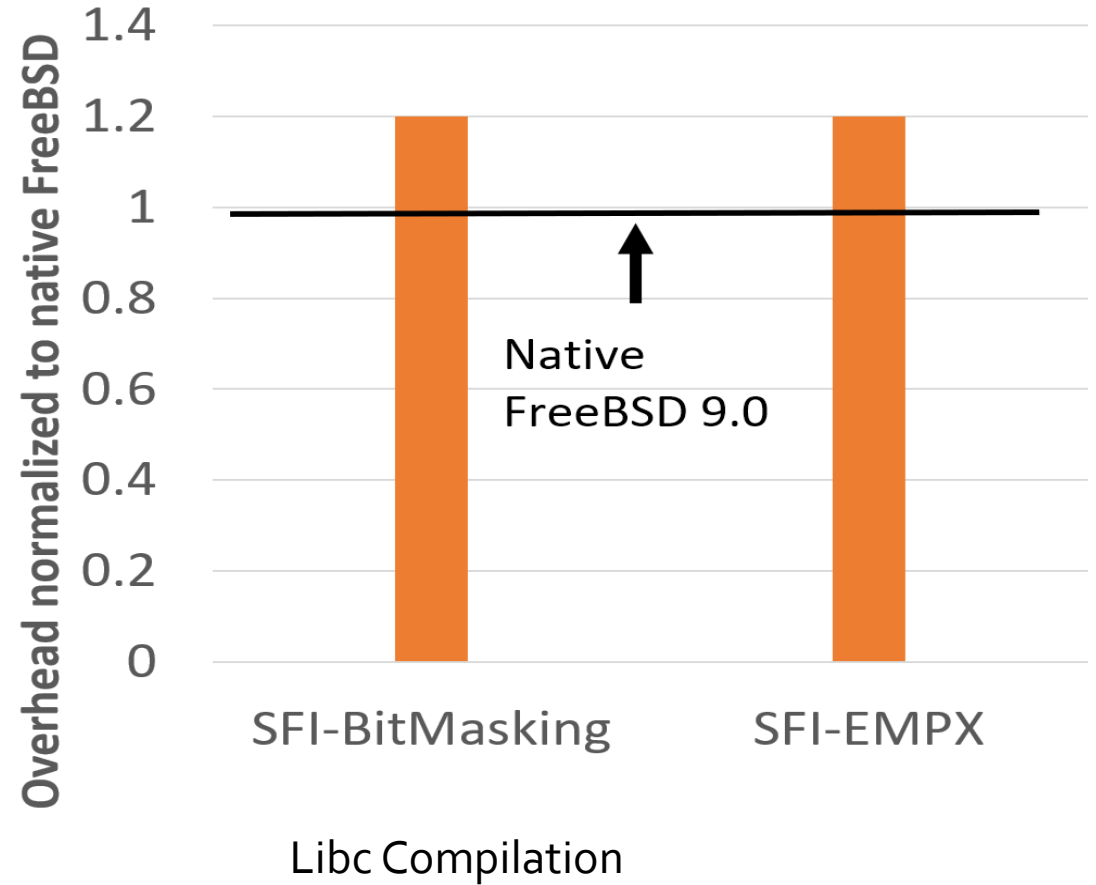
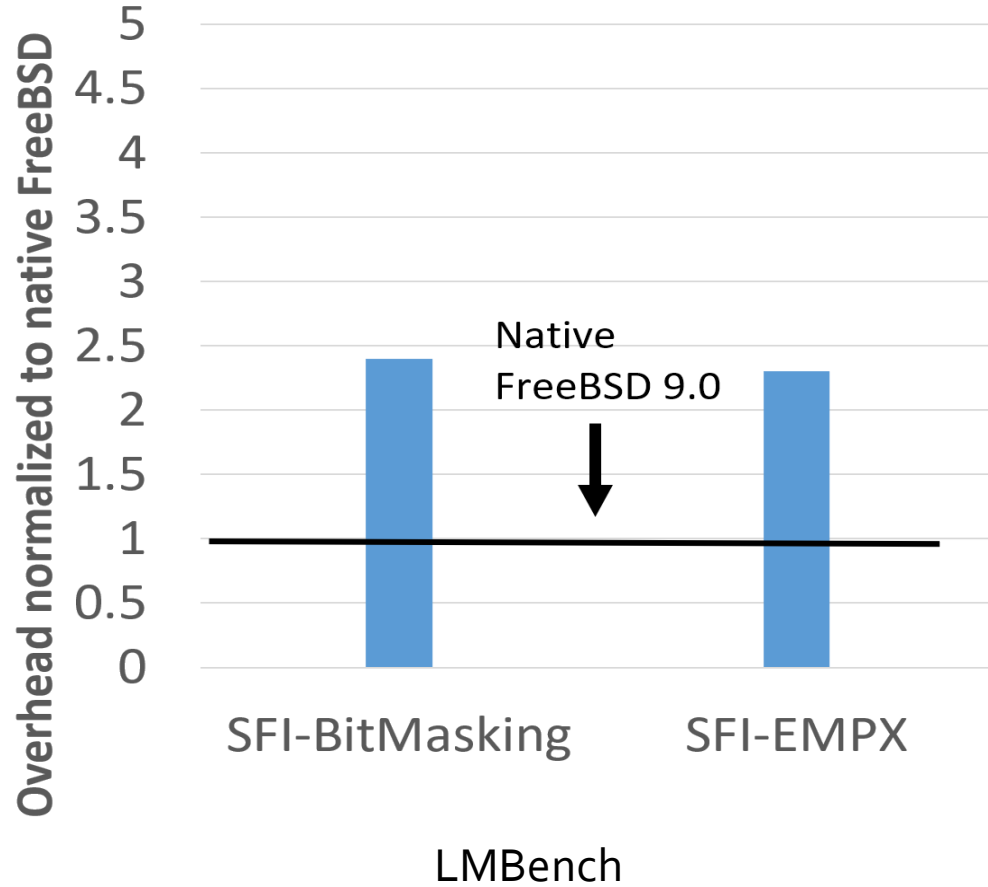
# Enhanced MPX SFI

- We propose two architectural improvements to MPX
- Add a *data dependence* between bounds check and memory load
  - Eliminate *lfence*
  - Set a flag in a register if the address is out-of-bounds and check the flag later
- Support checks against *multiple bounds*
  - Reduce register pressure
  - alleviate pointer arithmetic

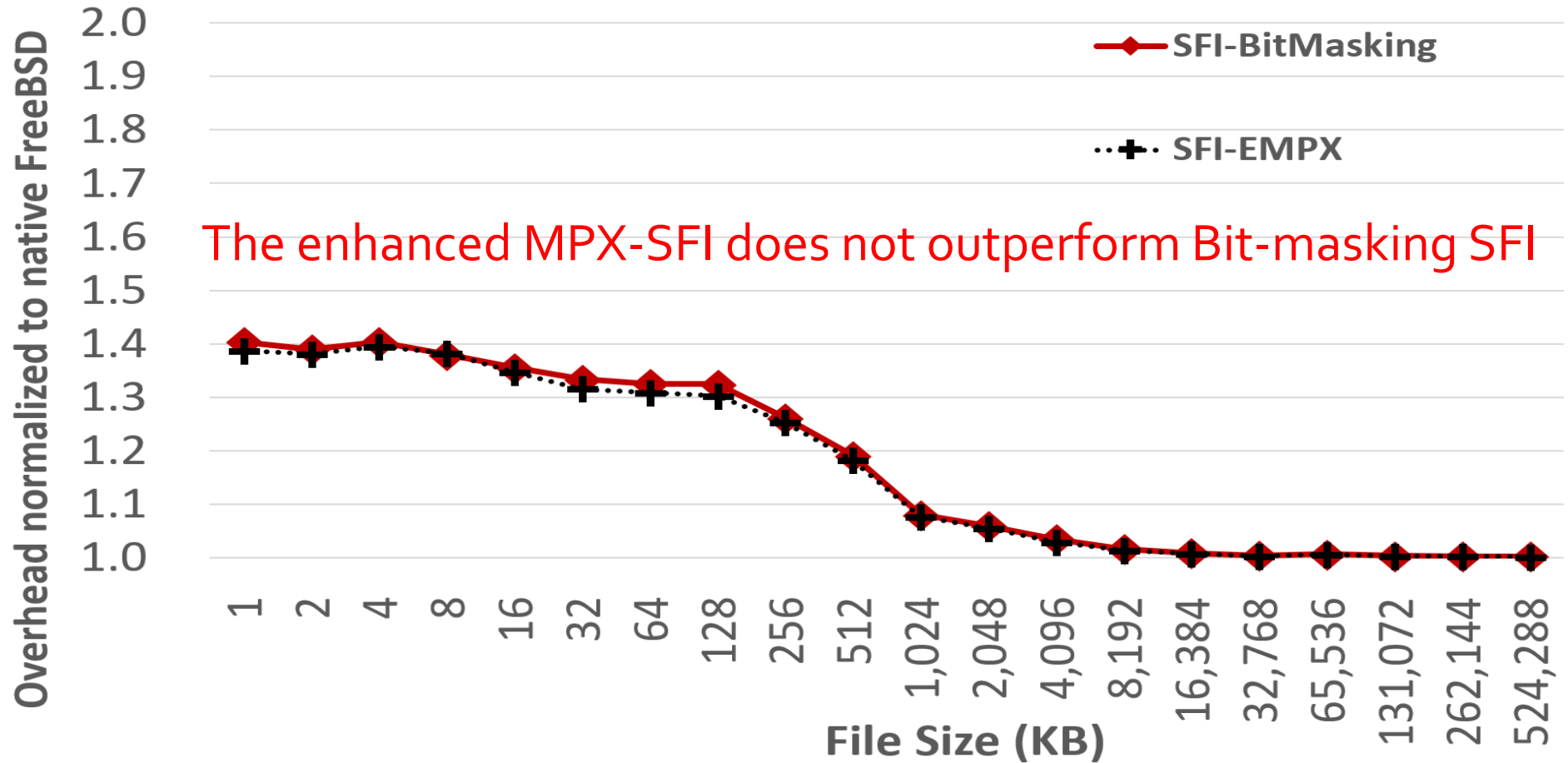
```
bndcl MULTI_BOUNDS, %R1  
/* cmp is used to emulate new bndcl */  
sete %R2  
sign-extend %R2  
or %R2, %R1  
mov (%R1), %R1
```

# Results

The enhanced MPX-SFI does not outperform Bit-masking SFI



# SSHD Results





# Conclusion

- Bit-Masking SFI usually achieves the best performance
- Separate address spaces can incur frequent and expensive address space switch
- MPX-SFI with Ifence performs the worst
  - With enhancements, MPX SFI does not outperform Bit-Masking SFI