*k*Memvisor: Flexible System Wide Memory Mirroring in Virtual Environments

Bin Wang
Zhengwei Qi
Haibing Guan
Haoliang Dong
Wei Sun

Shanghai Jiao Tong University

Yaozu Dong

Intel China Software Center

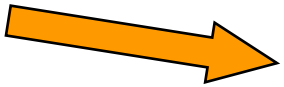
Is your memory error-prone?

Today's memory do become error-prone

- [B. Schroeder et al. *SIGMETRICS 09*] Memory failures are **common in clusters**
- **8% of DIMMs** have correctable errors per year
- **1.29% uncorrectable errors** in Google testbed

Today's memory do become error-prone

- [B. Schroeder et al. *SIGMETRICS 09*] Memory failures are **common in clusters**
- **8% of DIMMs** have correctable errors per year
- **1.29% uncorrectable errors** in Google testbed

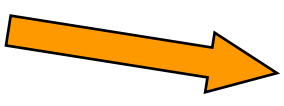


Memory -intensive App.

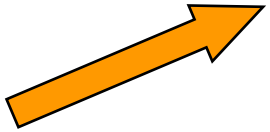
Today's memory do become error-prone

- [B. Schroeder et al. *SIGMETRICS 09*] Memory failures are **common in clusters**
- **8% of DIMMs** have correctable errors per year
- **1.29% uncorrectable errors** in Google testbed

Memory -intensive App.



Memory -intensive App.



Memory -intensive App.

Memory HA in Cloud Computing

1.29% error rate

Memory HA in Cloud Computing

1.29% error rate

Memory HA in Cloud Computing

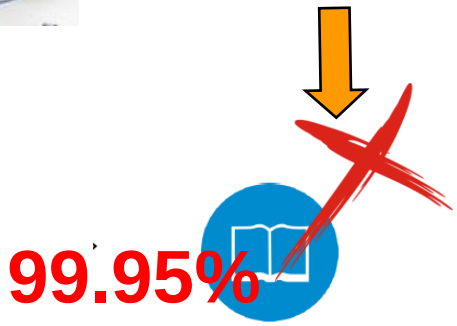
1.29% error rate

13,000 failures per year
1,083 failures per month
35 failures per day
1.5 failure per hour

Memory HA in Cloud Computing

1.29% error rate

13,000 failures per year
1,083 failures per month
35 failures per day
1.5 failure per hour



99.95%

Service Level Aggrement

Memory HA in Cloud Computing

1.29% error rate

13,000 failures per year
1,083 failures per month
35 failures per day
1.5 failure per hour

99.95%

Service Level Aggrement

4.38 hours downtime per year
21.56 minutes downtime per month
5.04 minutes downtime per day

Memory HA in Cloud Computing

1.29% error rate

13,000 failures per year
1,083 failures per month
35 failures per day
1.5 failure per hour

99.95%

Service Level Aggrement

4.38 hours downtime per year
21.56 minutes downtime per month
5.04 minutes downtime per day

[Andy A. Hwang et al. *ASPLOS 12*] Memory Errors happen at a significant rate in all four sites with **2.5 to 5.5%** of nodes affected per system

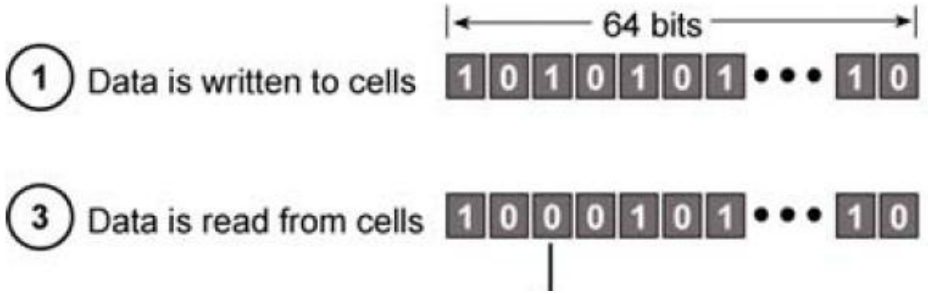
Existing Solutions

Hardware

Existing Solutions

Hardware

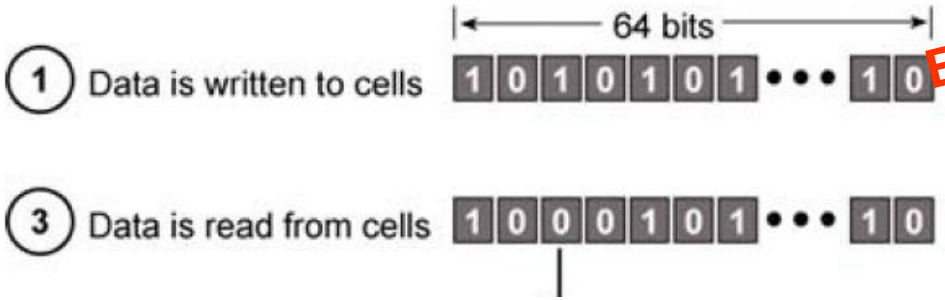
ECC (Hp, IBM, Google et al.)



Existing Solutions

Hardware

ECC (Hp, IBM, Google et al.)

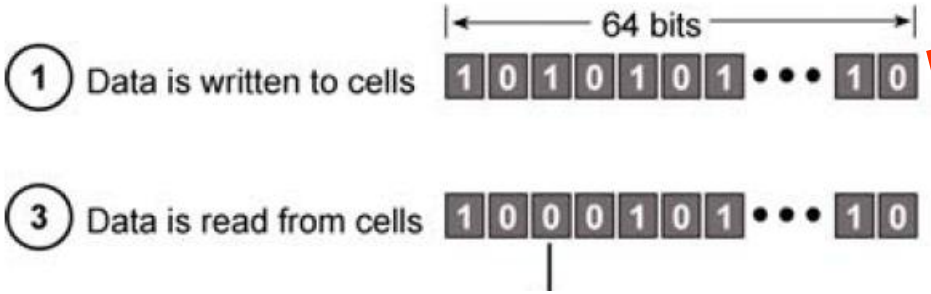


Bit-granularity checking

Existing Solutions

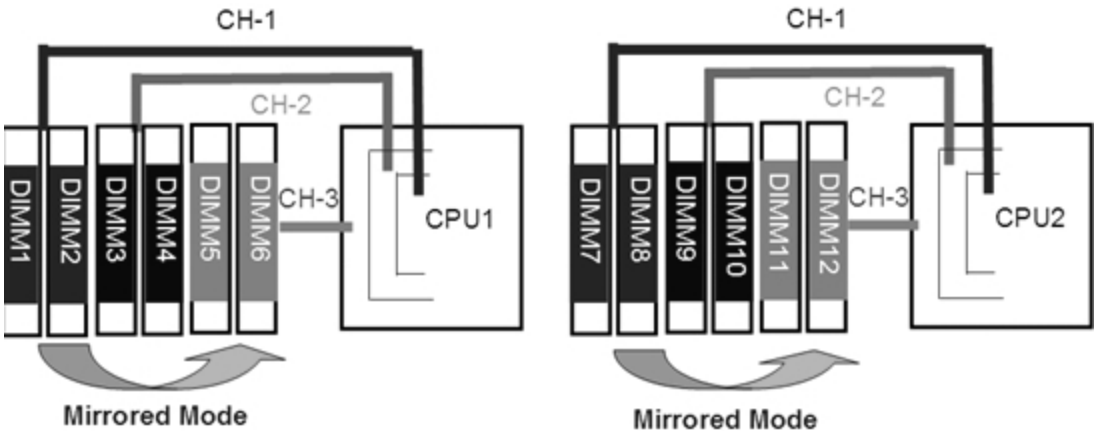
Hardware

ECC (Hp, IBM, Google et al.)



Bit-granularity checking

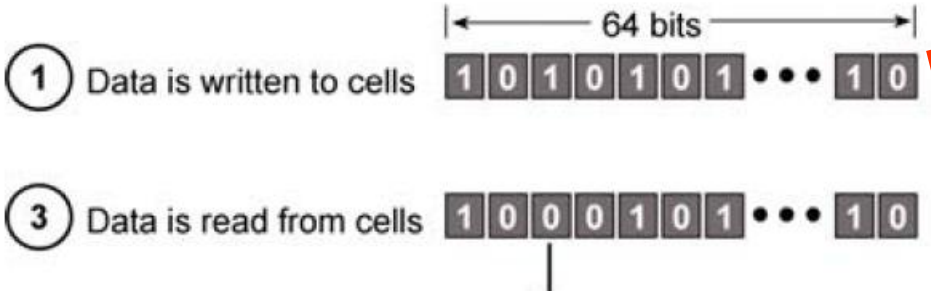
Mirrored Memory (Hp, IBM, Google et al.)



Existing Solutions

Hardware

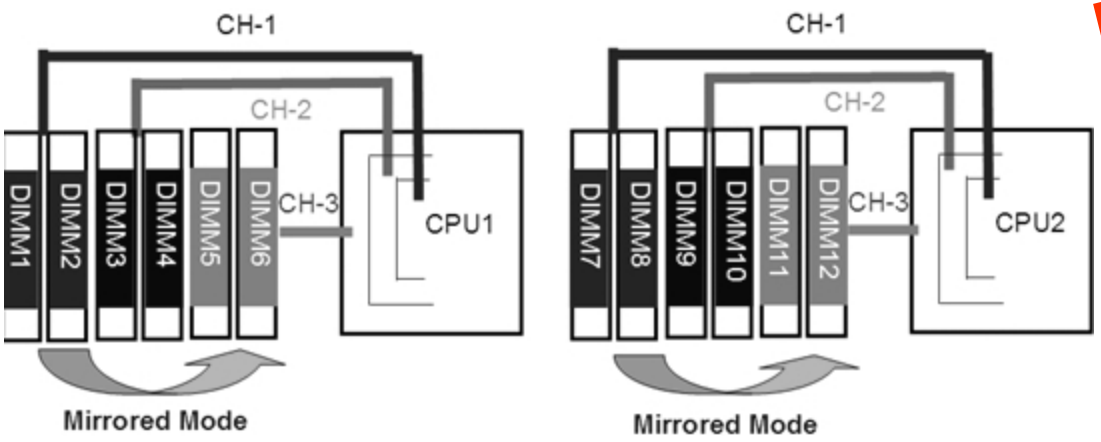
ECC (Hp, IBM, Google et al.)



Bit-granularity checking

Mirrored Memory (Hp, IBM, Google et al.)

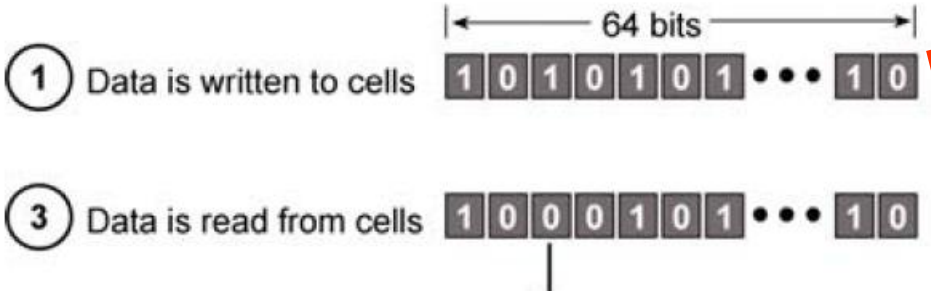
Expensive



Existing Solutions

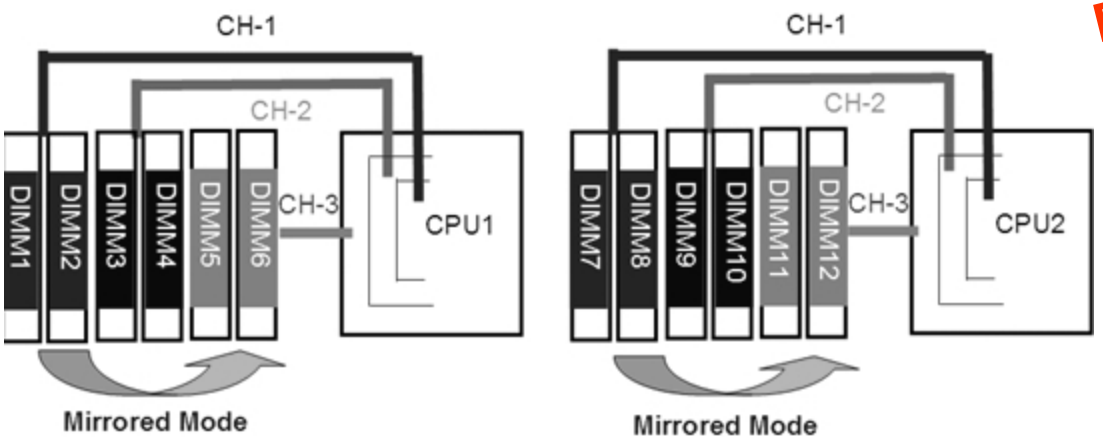
Hardware

ECC (Hp, IBM, Google et al.)



Bit-granularity checking

Mirrored Memory (Hp, IBM, Google et al.)



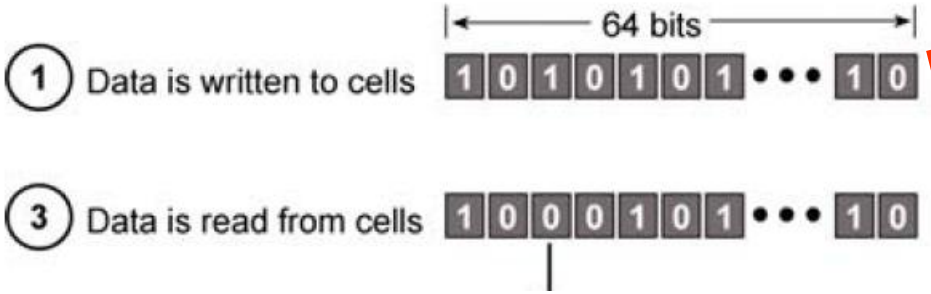
Expensive

Low compatibility

Existing Solutions

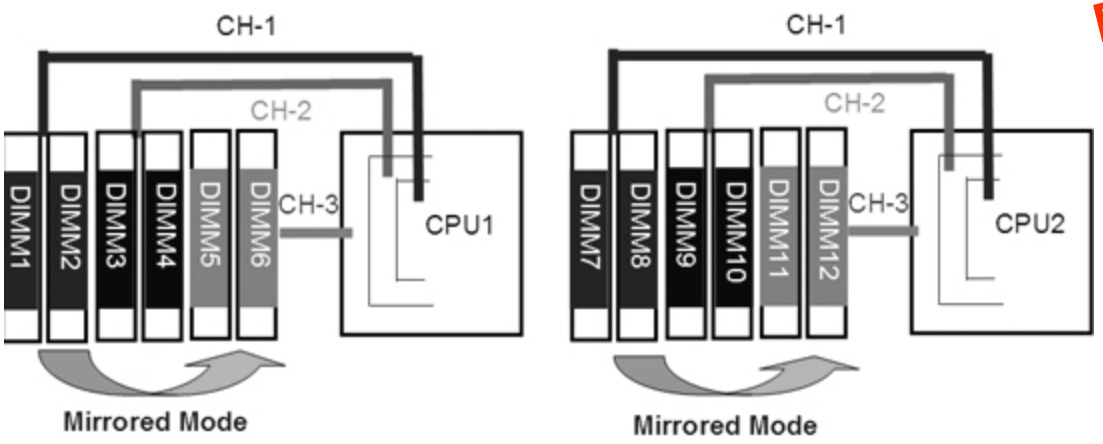
Hardware

ECC (Hp, IBM, Google et al.)



Bit-granularity checking

Mirrored Memory (Hp, IBM, Google et al.)



Expensive

Low compatibility

Low flexibility

Existing Solutions

- Software

- Duo-backup (GFS, Amazon Dynamo)

- Checkpoint, hot spare+ VM migration /replication

Existing Solutions

- Software

- Duo-backup (GFS, Amazon Dynamo)

Syetem level tolerance

- Checkpoint, hot spare+ VM migration /replication

Application-Specific and High overhead
(eg. Remus [*NSDI 08*] with **40** Checkpoints/sec, overhead **103%**)

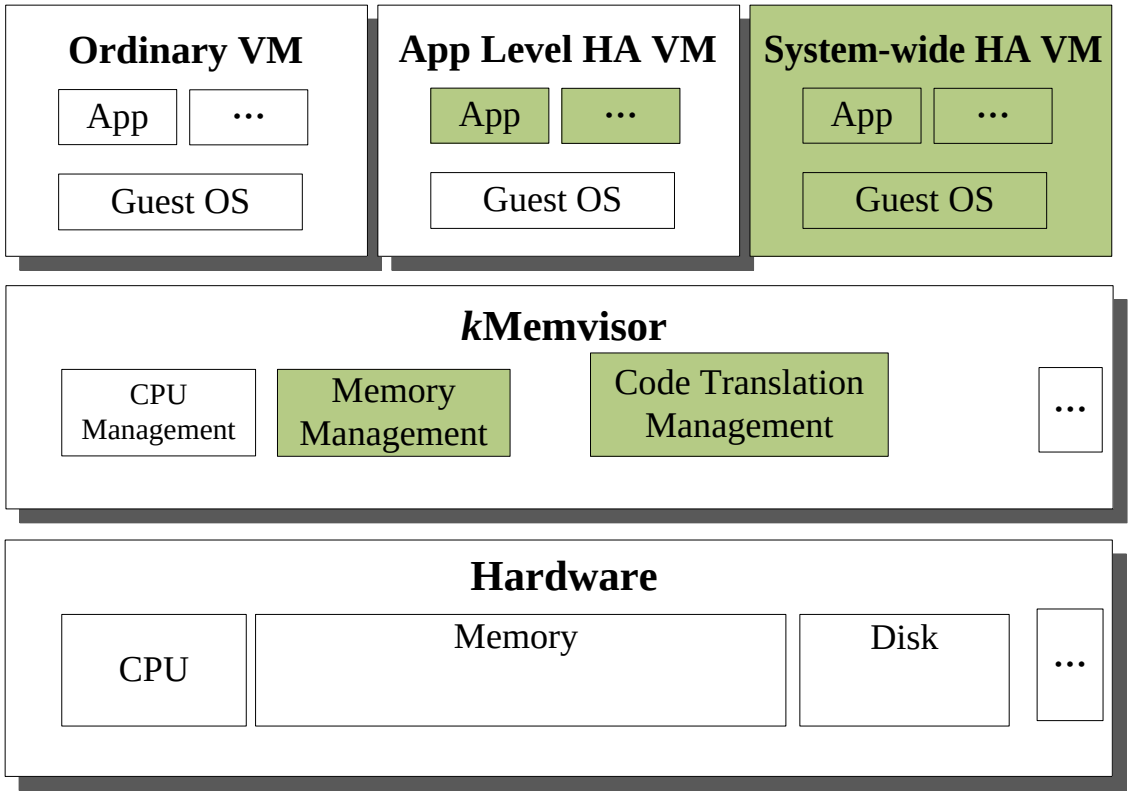
Design Guideline

- Low cost
- Efficiency & compatibility
  - arbitrary platform
  - on the fly+ hot spare
- Low maintaining
  - little impact to others (eg., networking utilization)
  - without migration
- Cloud requirement
  - VM granularity

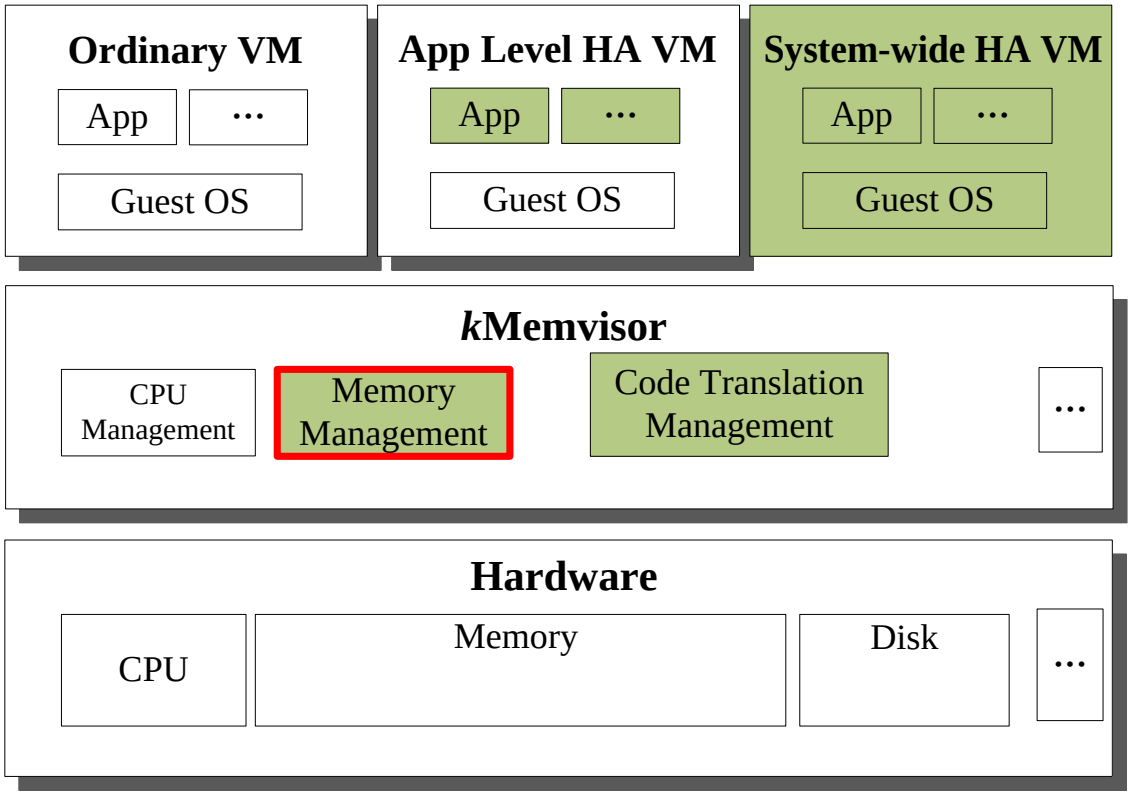
*k*Memvisor

- A hypervisor providing system-wide memory mirroring based on hardware virtualization
- Supporting VMs with or without mirror memory feature on a same physical machine
- Supporting *NModularRedundancy* for some special mission critical applications

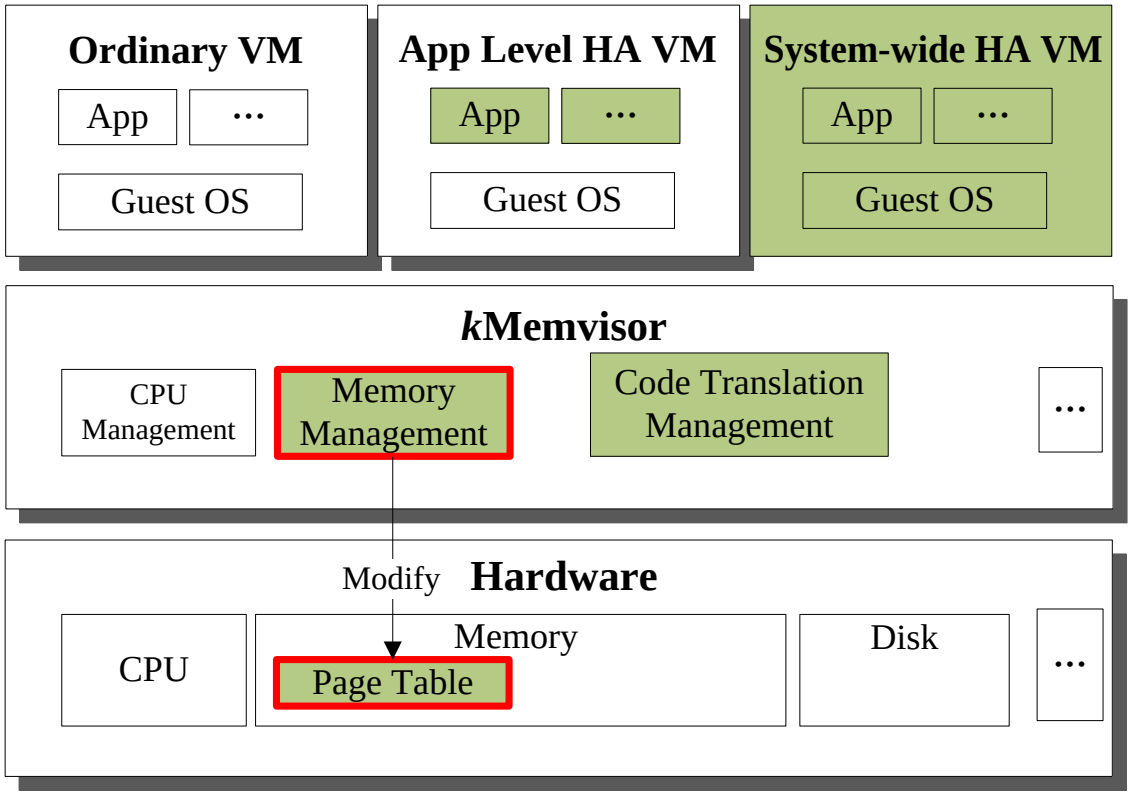
*k*Memvisor High-level Architecture



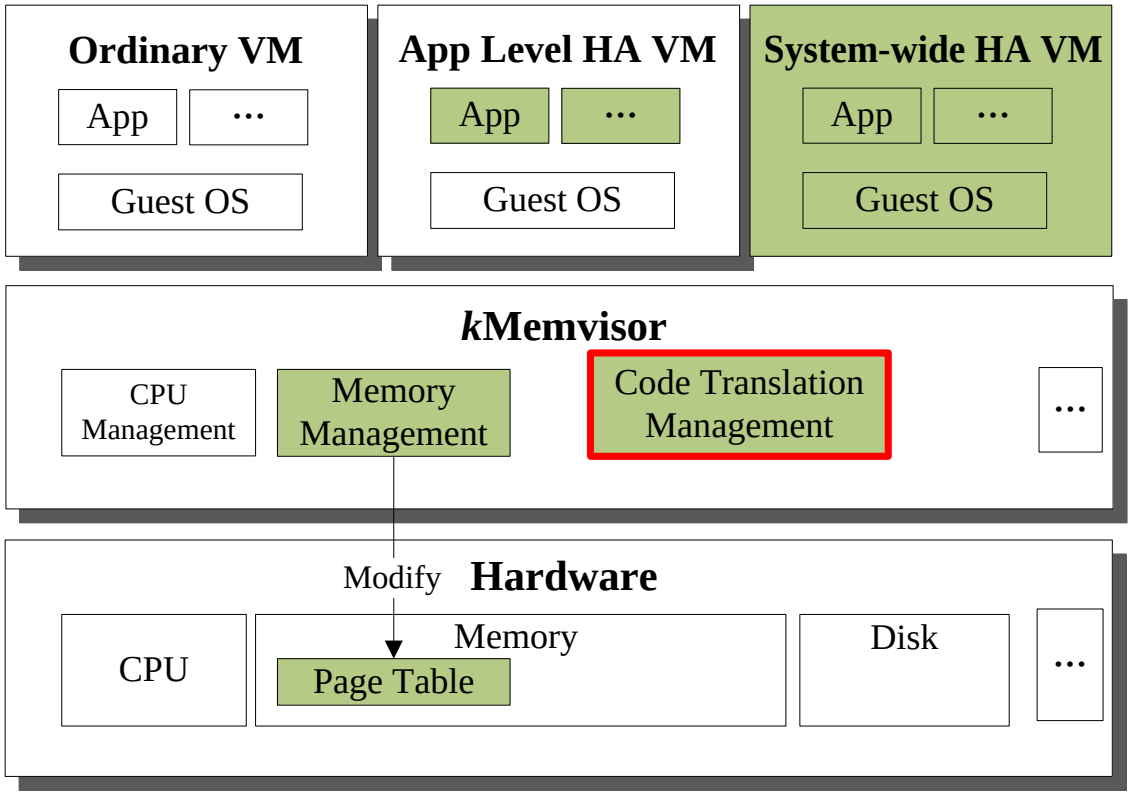
*k*Memvisor High-level Architecture



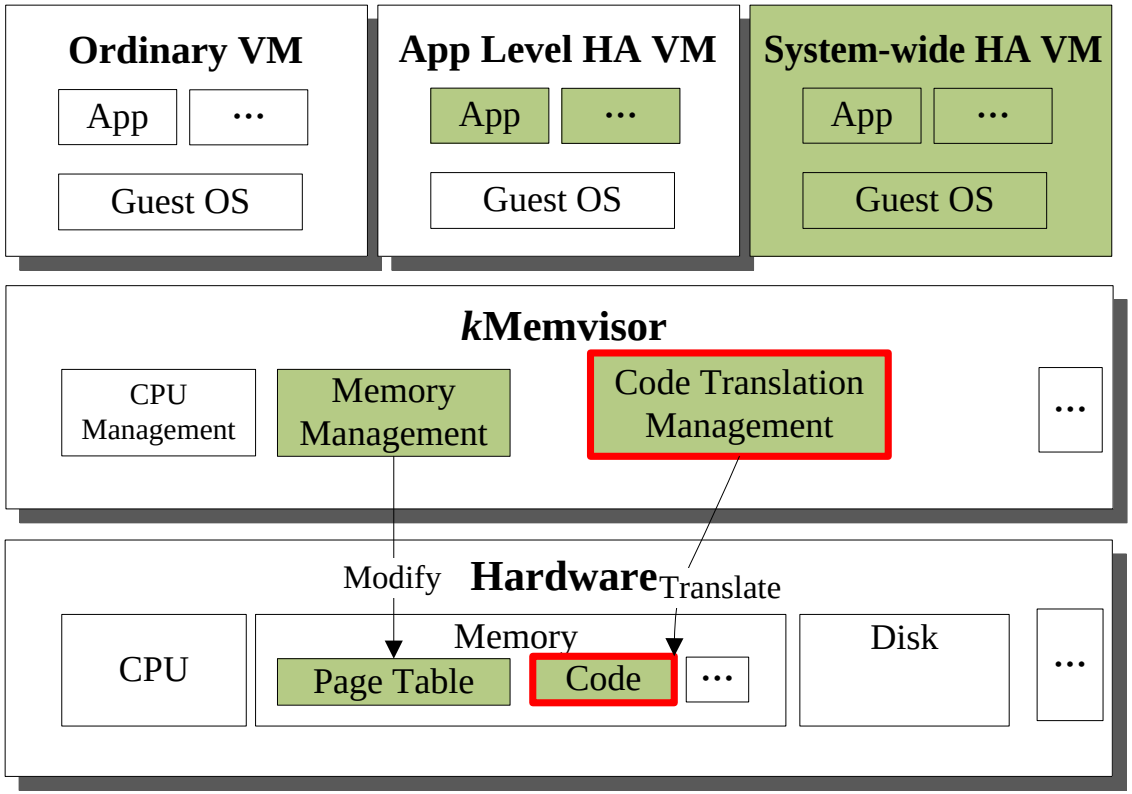
*k*Memvisor High-level Architecture



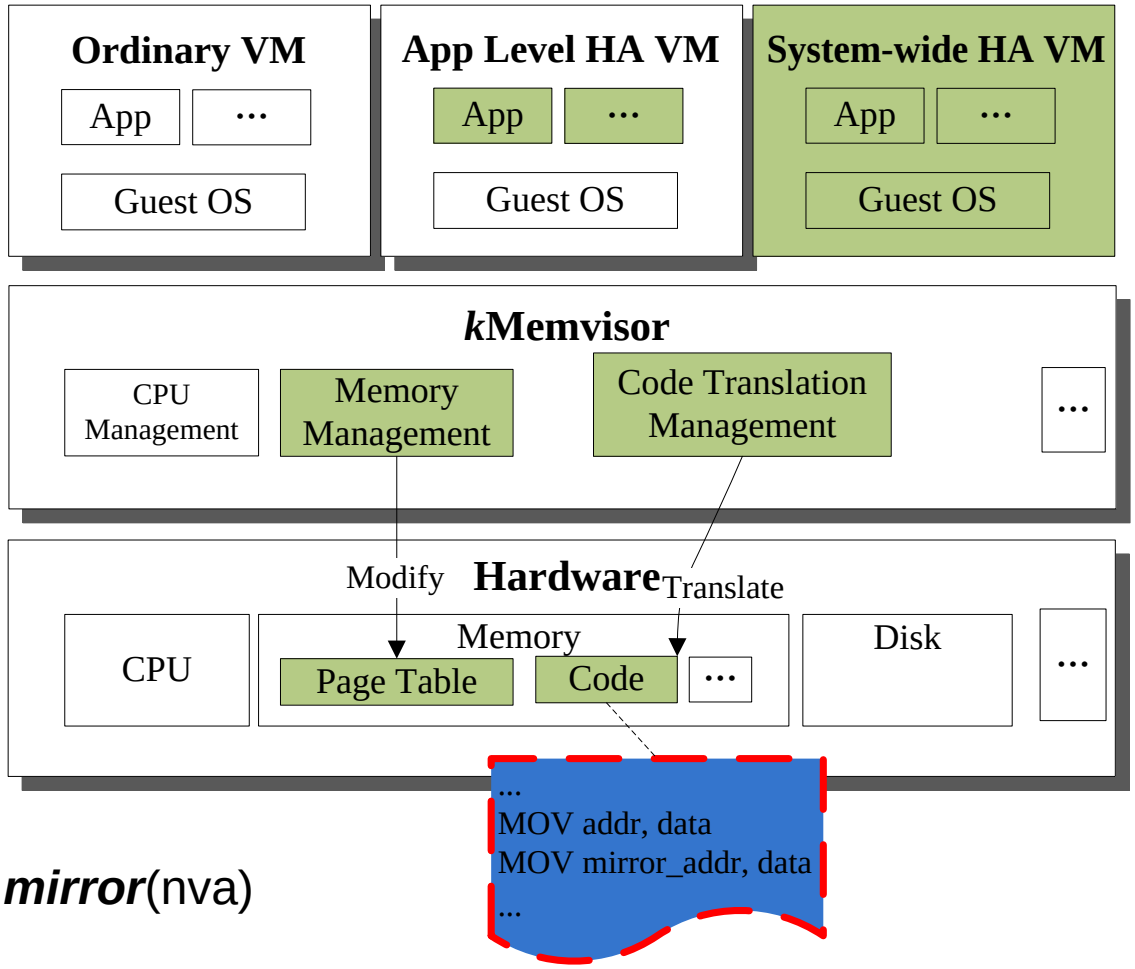
*k*Memvisor High-level Architecture



*k*Memvisor High-level Architecture

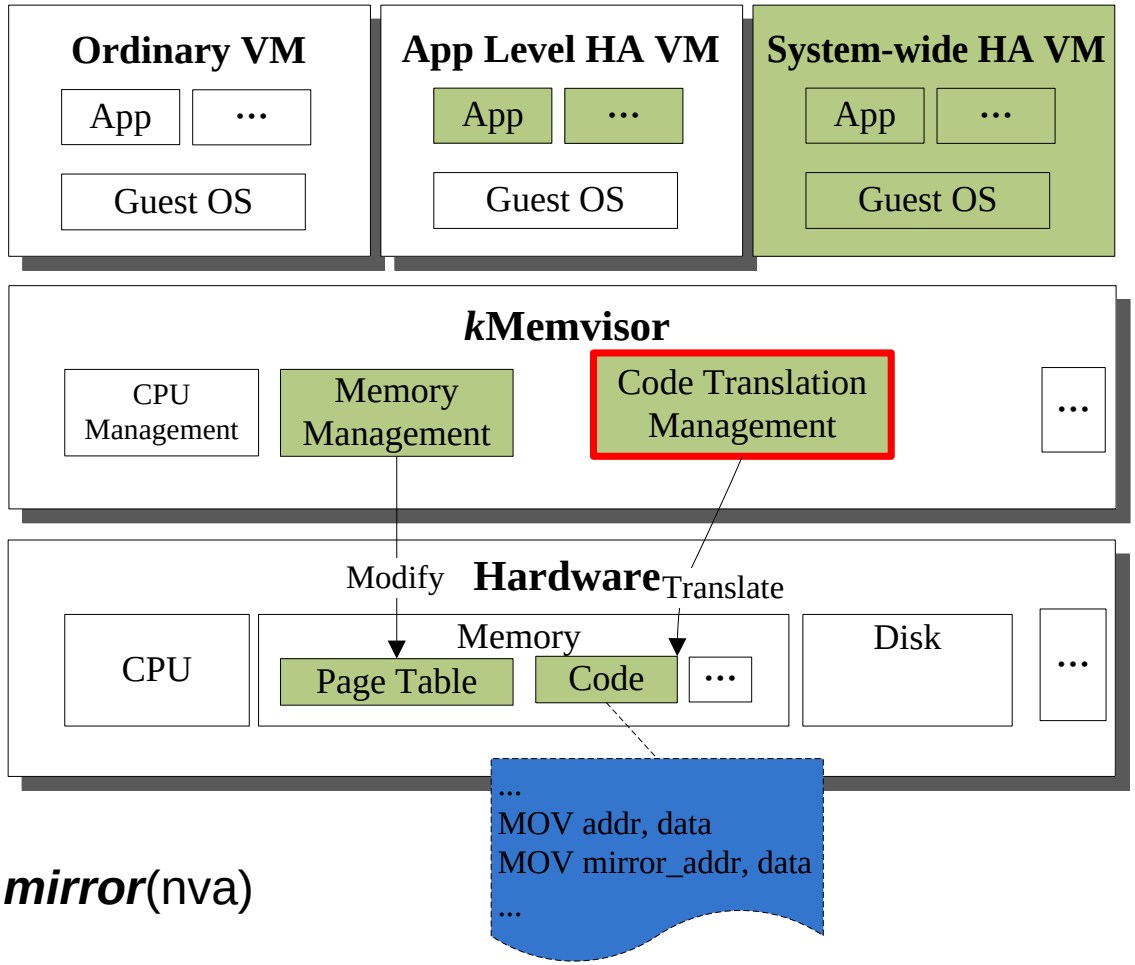


*k*Memvisor High-level Architecture



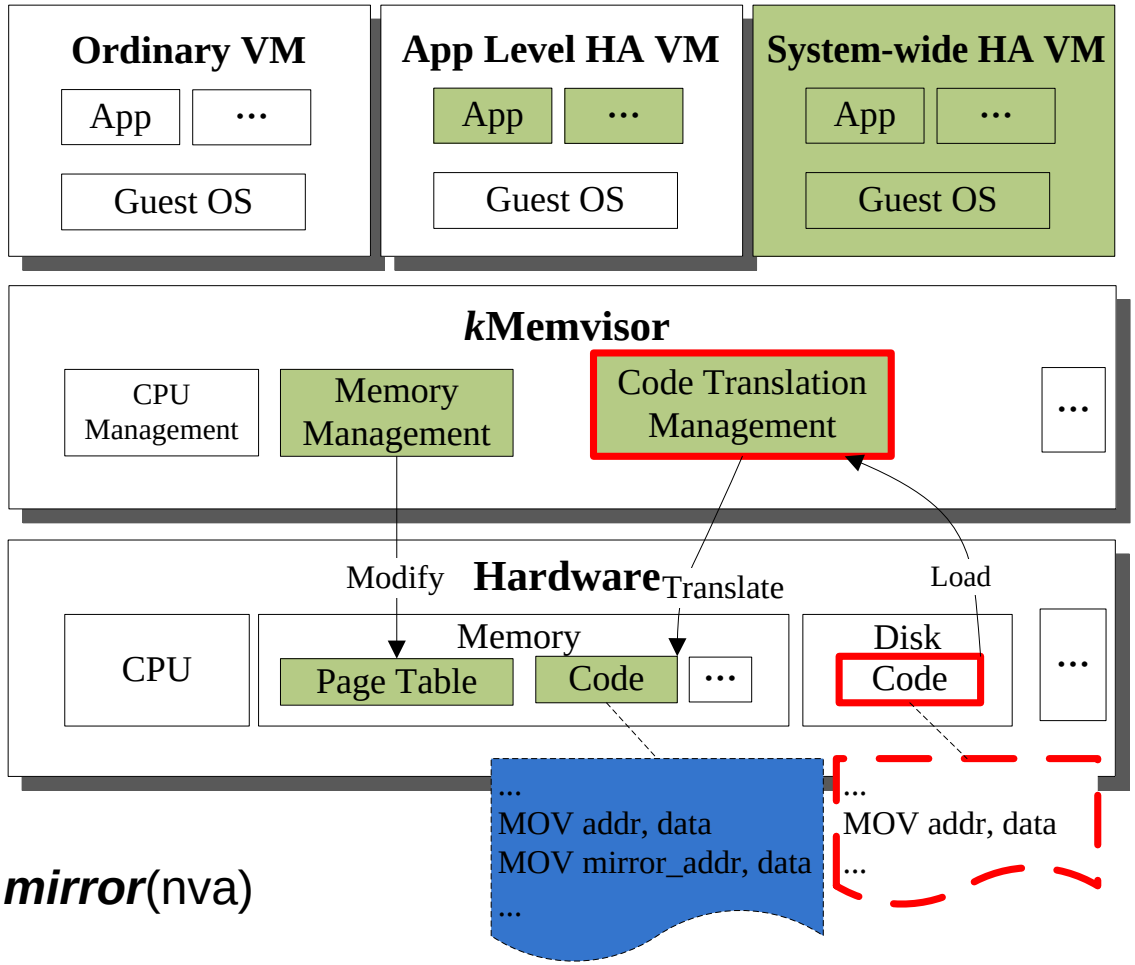
mva = ***mirror***(nva)

*k*Memvisor High-level Architecture



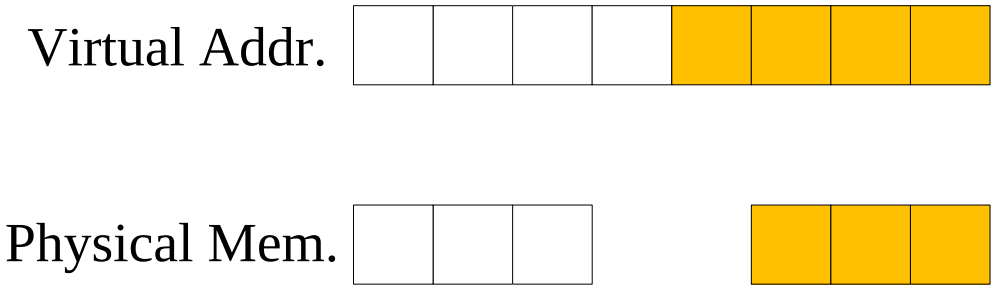
mva = ***mirror***(nva)

*k*Memvisor High-level Architecture



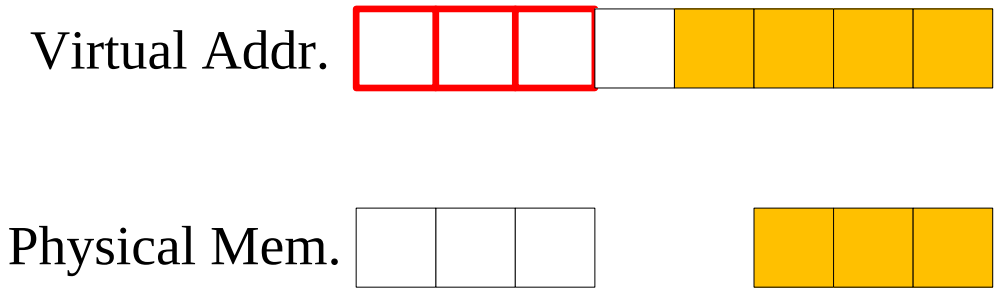
mva = ***mirror***(nva)

Retrieve Memory Failure

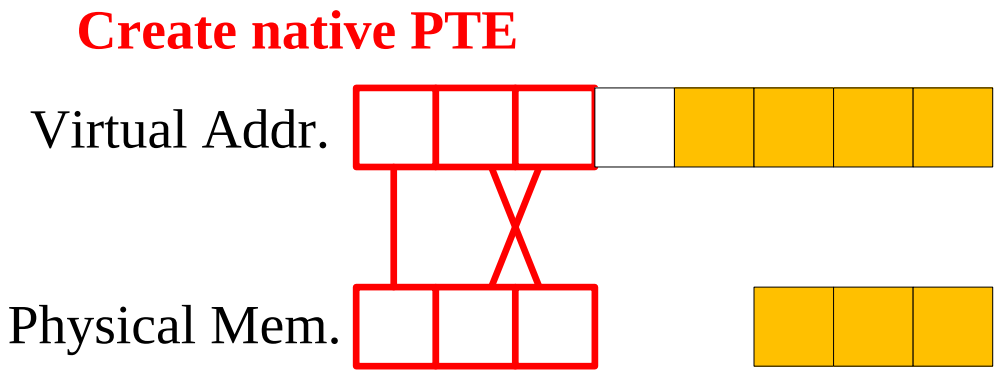


Retrieve Memory Failure

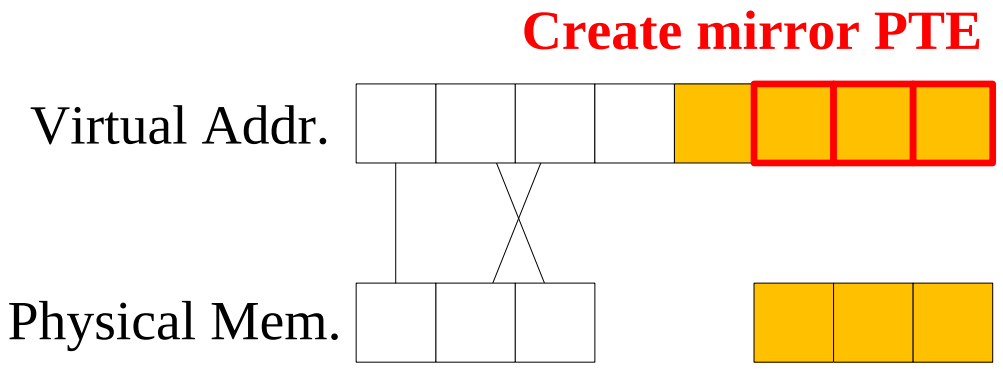
Create native PTE



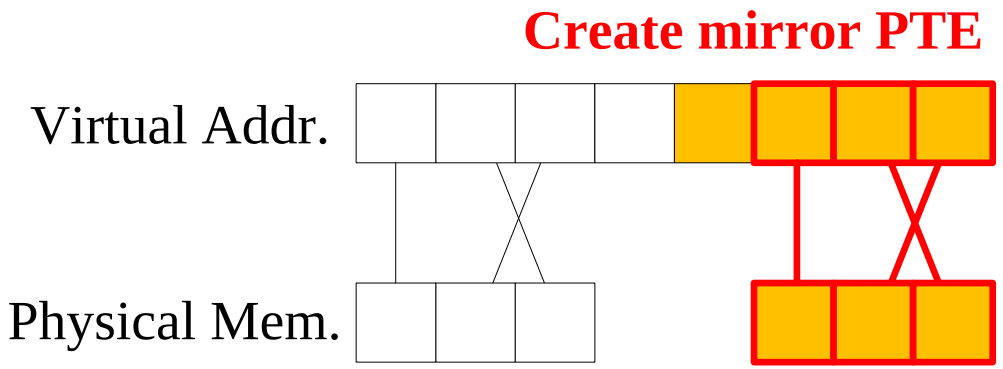
Retrieve Memory Failure



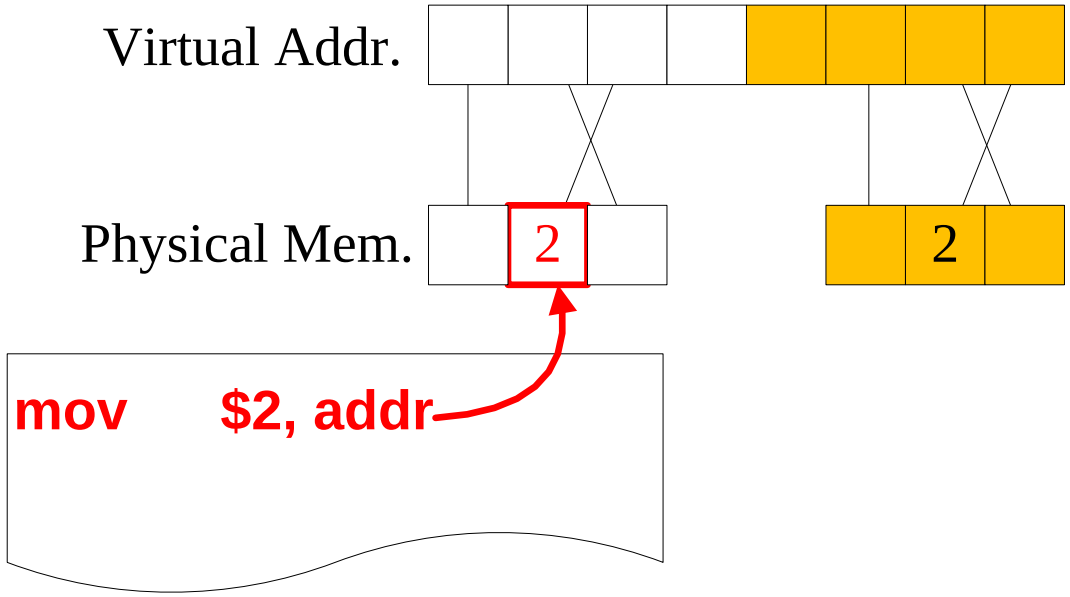
Retrieve Memory Failure



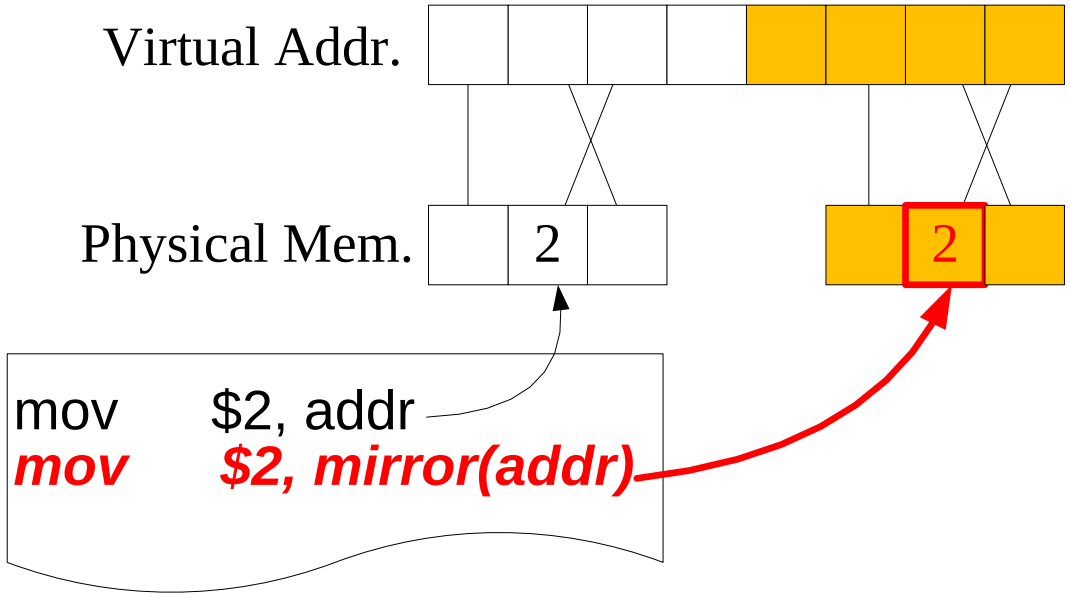
Retrieve Memory Failure



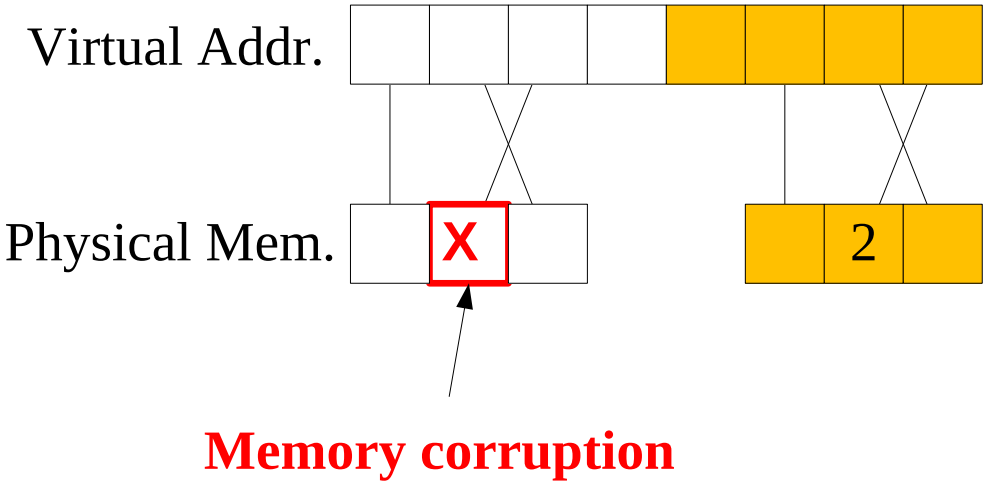
Retrieve Memory Failure



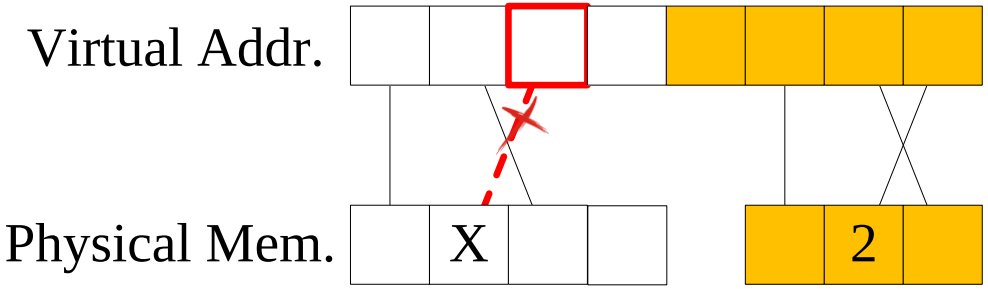
Retrieve Memory Failure



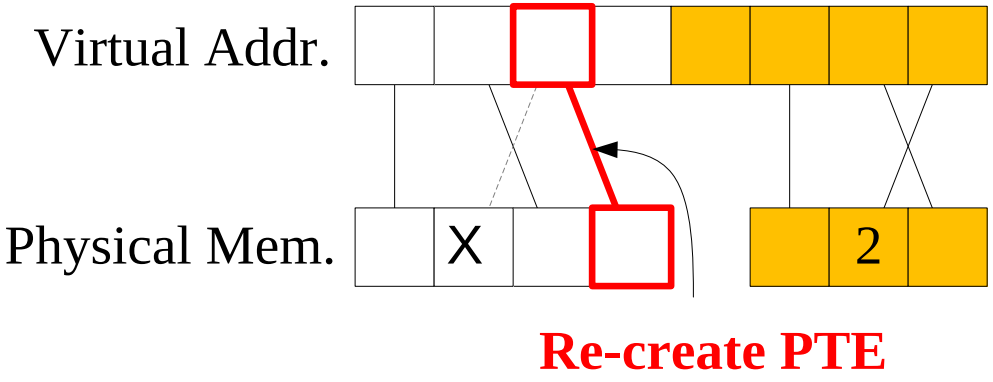
Retrieve Memory Failure



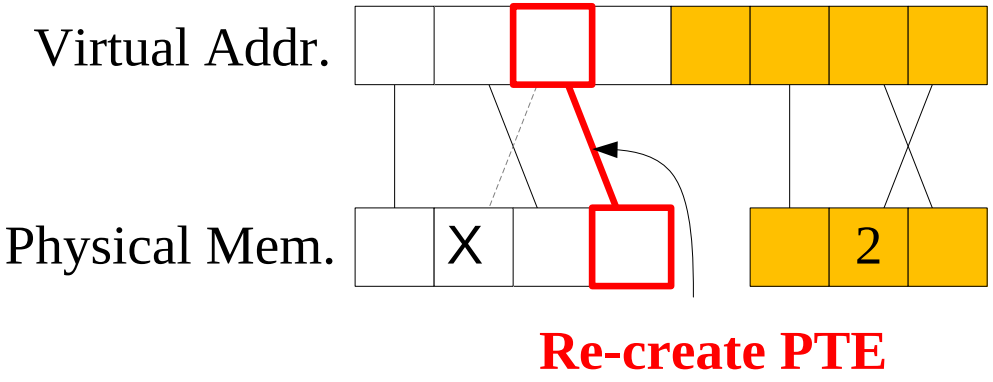
Retrieve Memory Failure



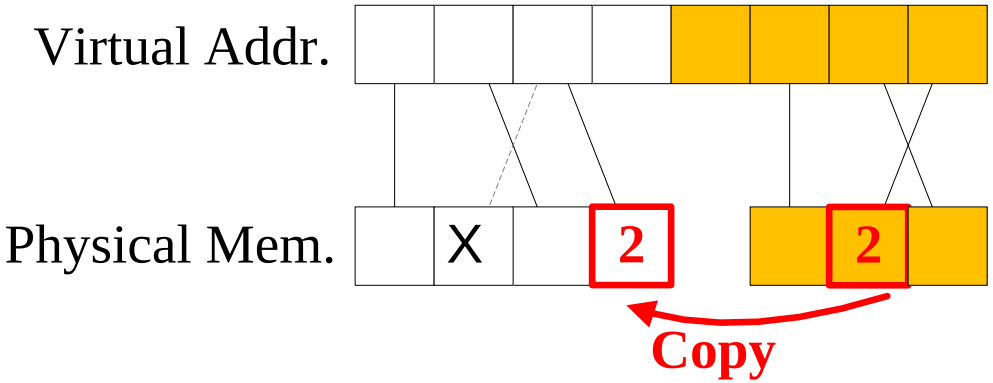
Retrieve Memory Failure



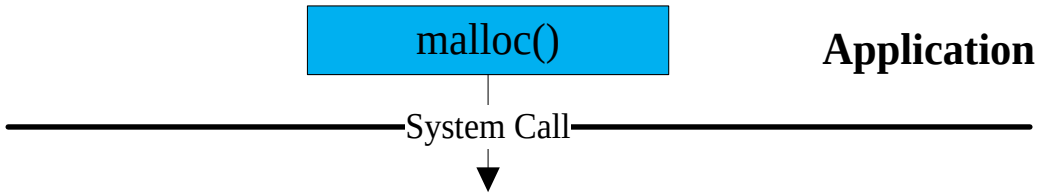
Retrieve Memory Failure



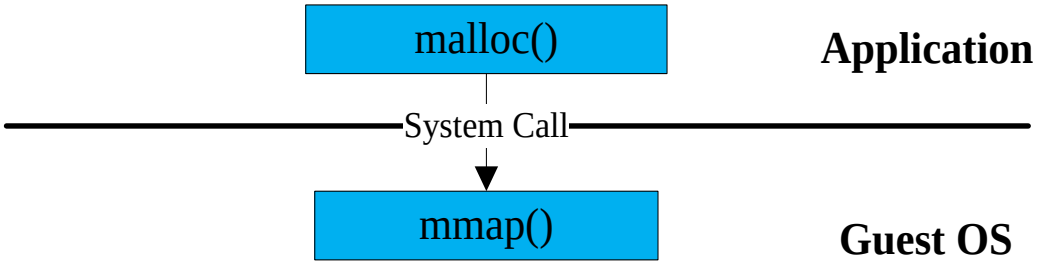
Retrieve Memory Failure



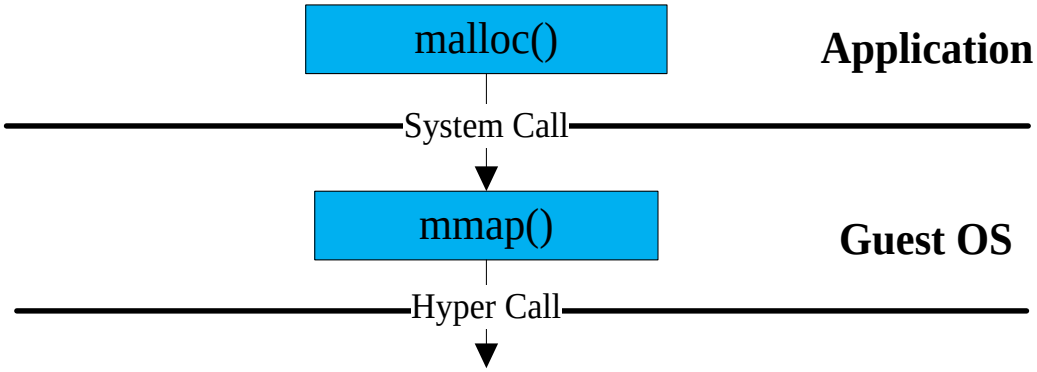
Create Mirror Page Table



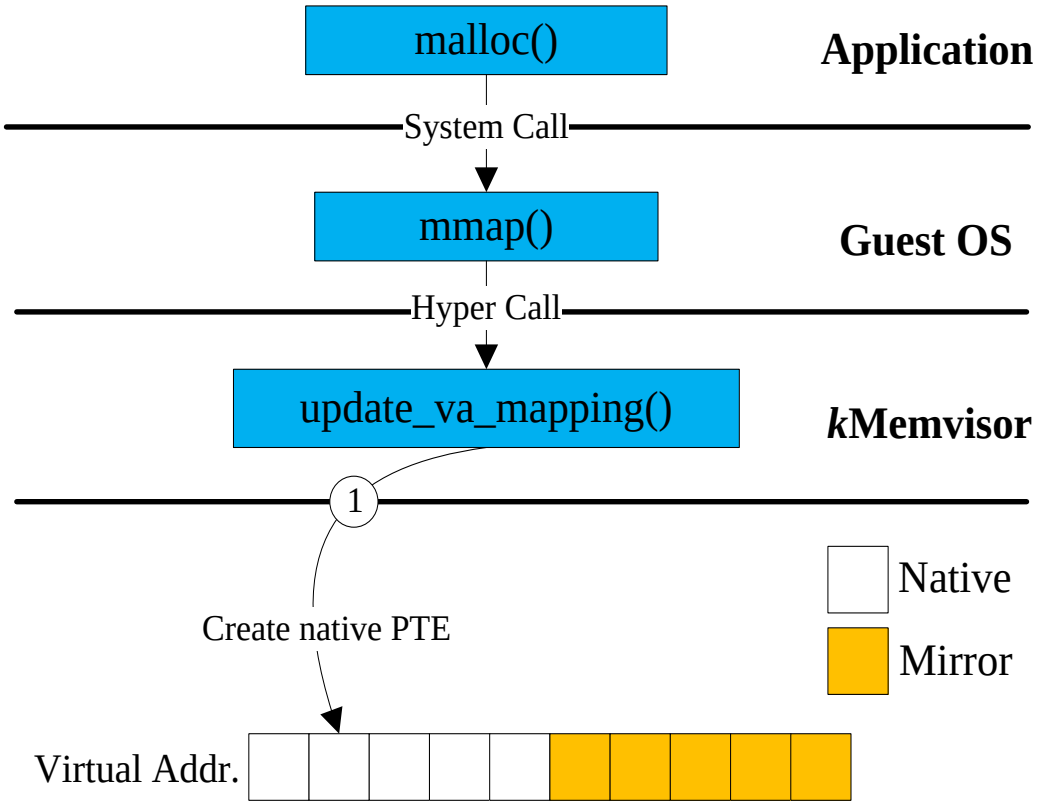
Create Mirror Page Table



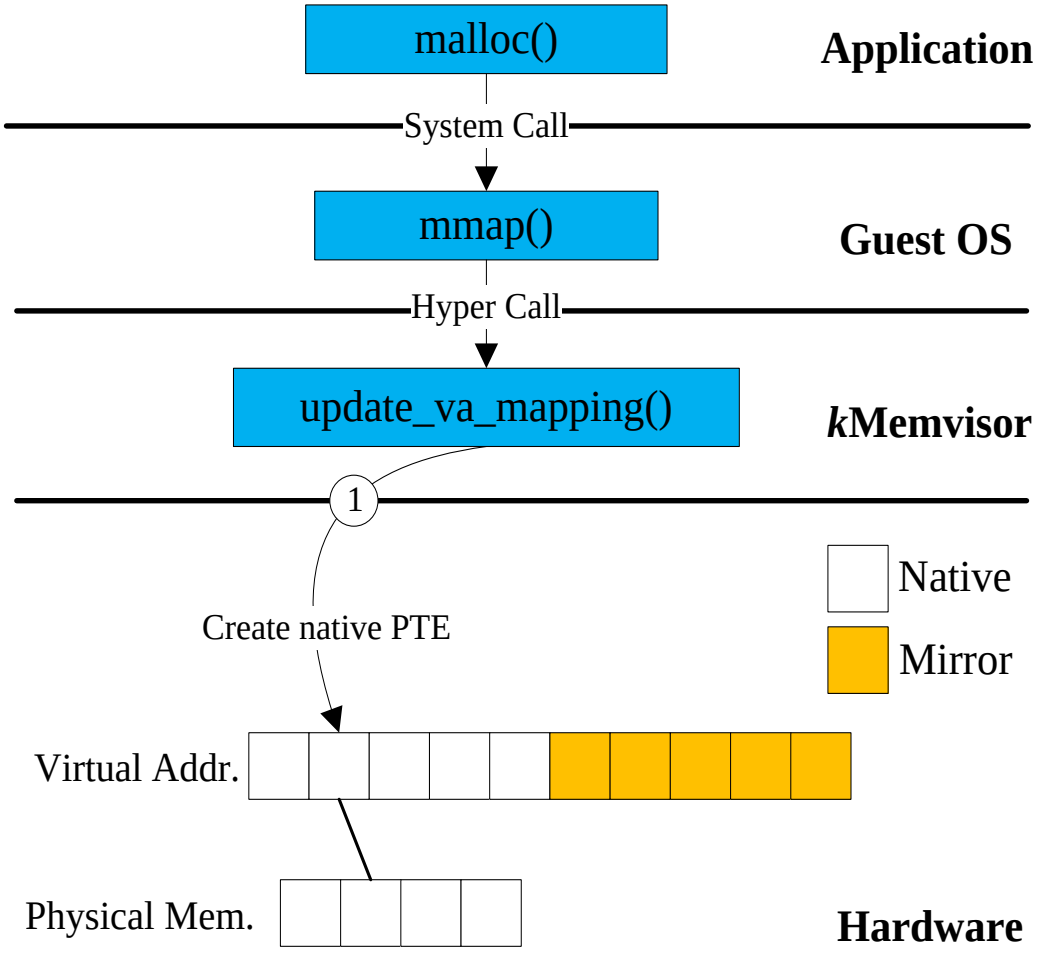
Create Mirror Page Table



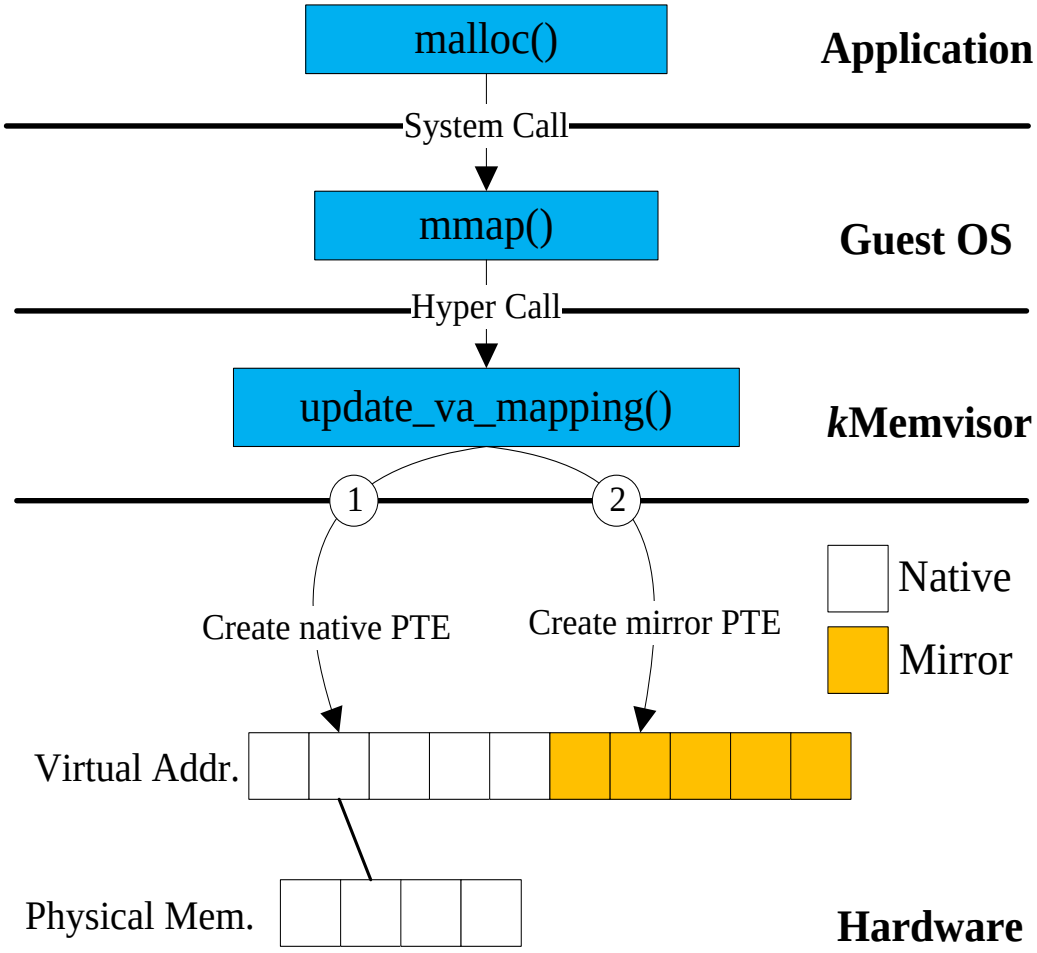
Create Mirror Page Table



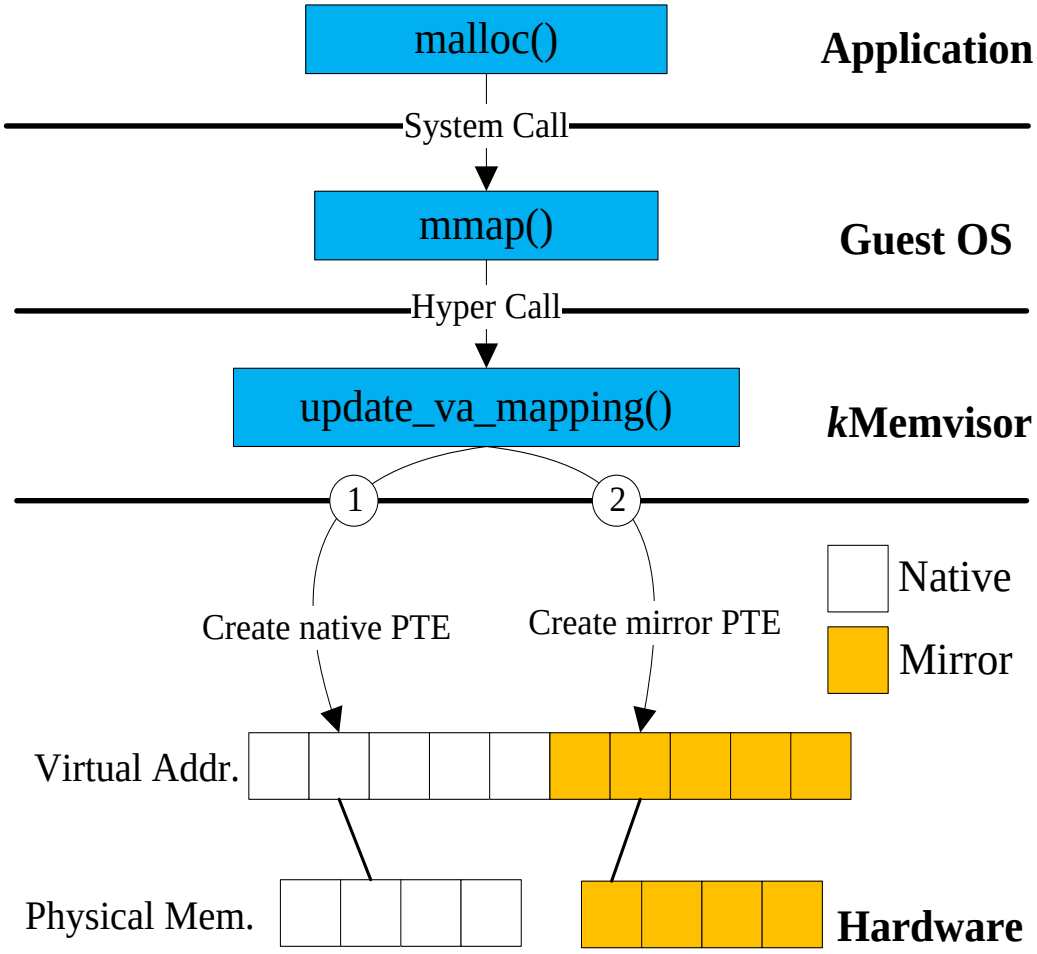
Create Mirror Page Table



Create Mirror Page Table



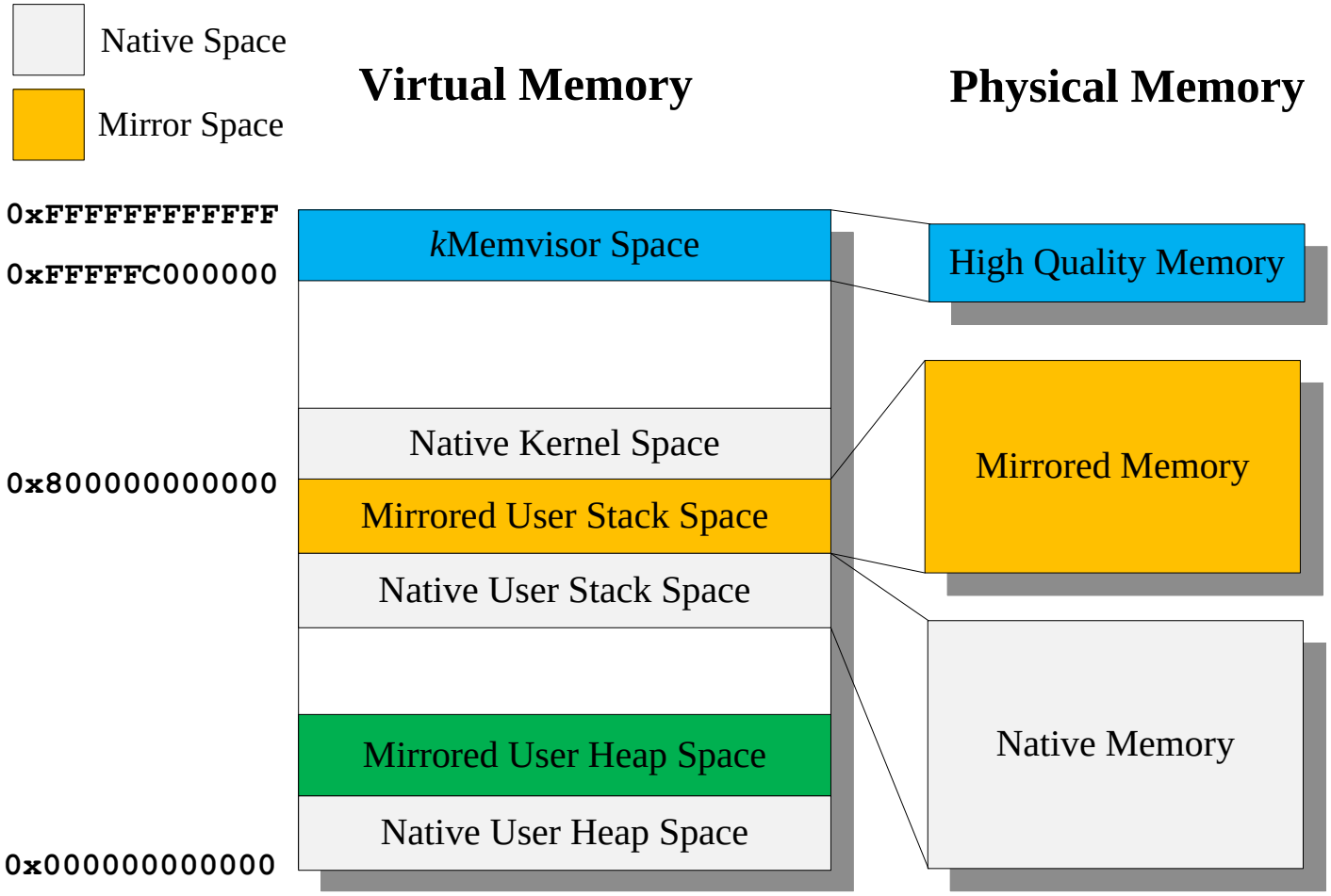
Create Mirror Page Table



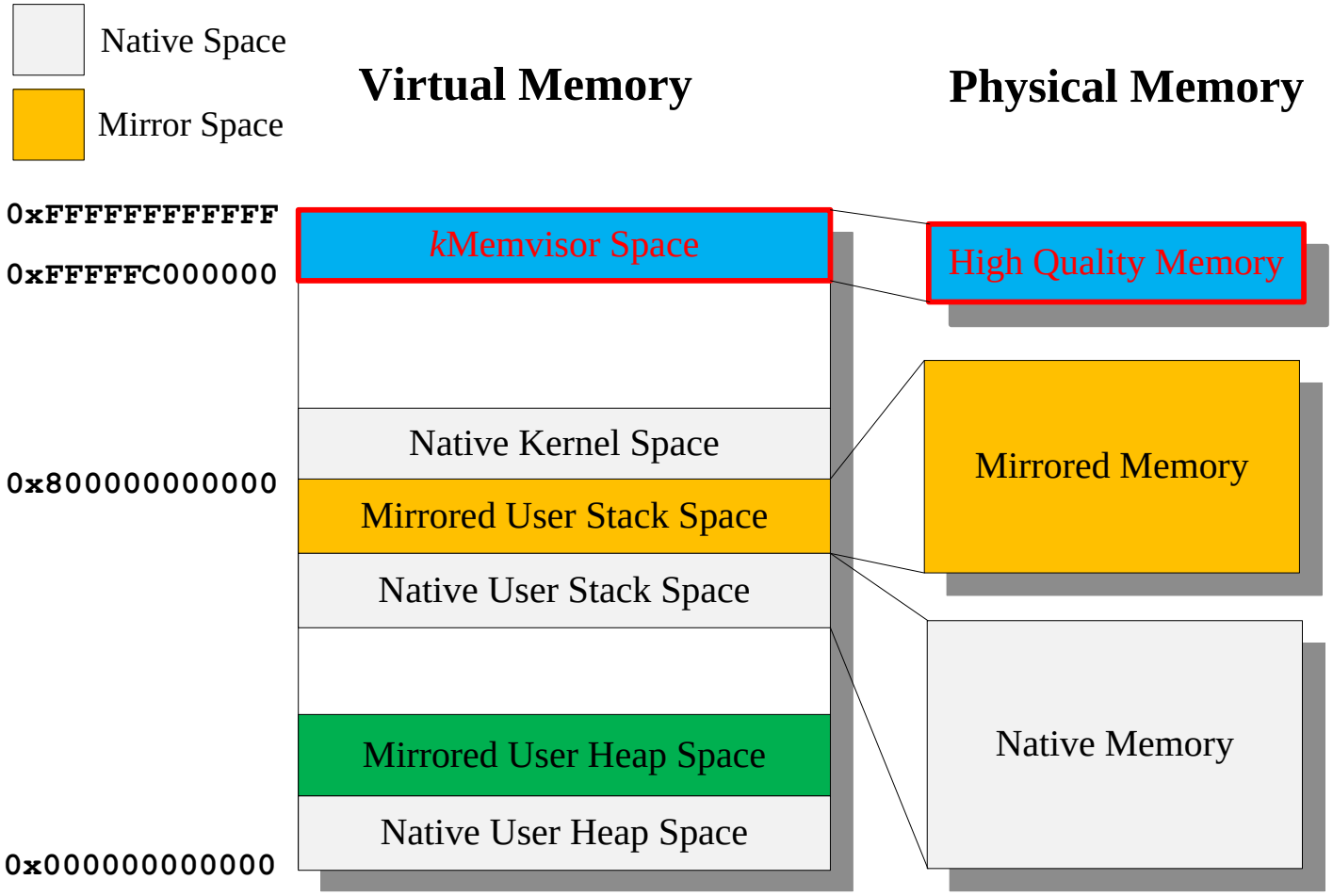
Modification

- Direct page table
  - Write the page table using a **hypercall**
  - Intercept the hypercall & Fill in the PTE
- Modify Guest OS
  - Guest OS has no awareness of iteration with the help of *vm_struct.*
  - Ignore mirror PTE

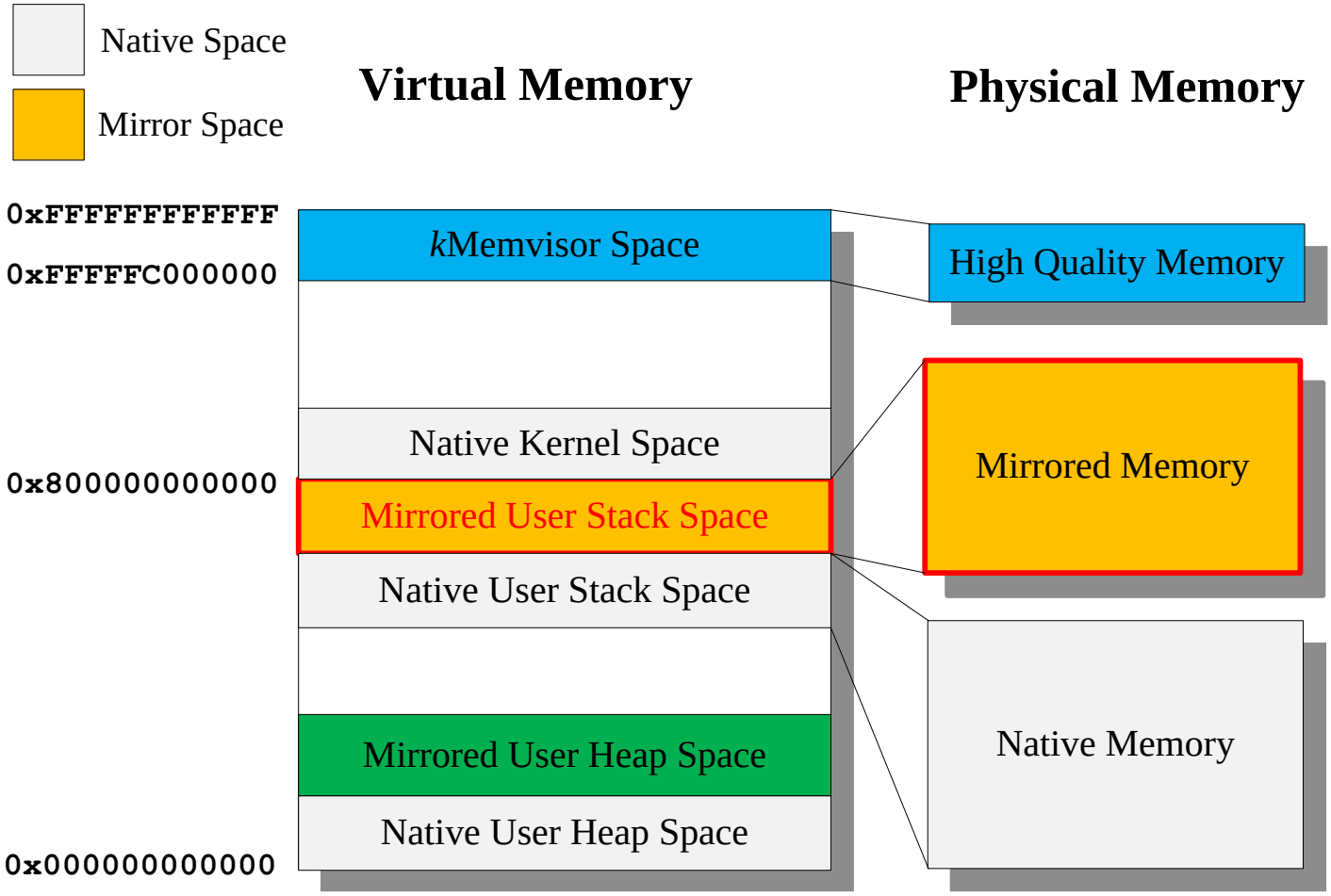
Modified Memory Layout



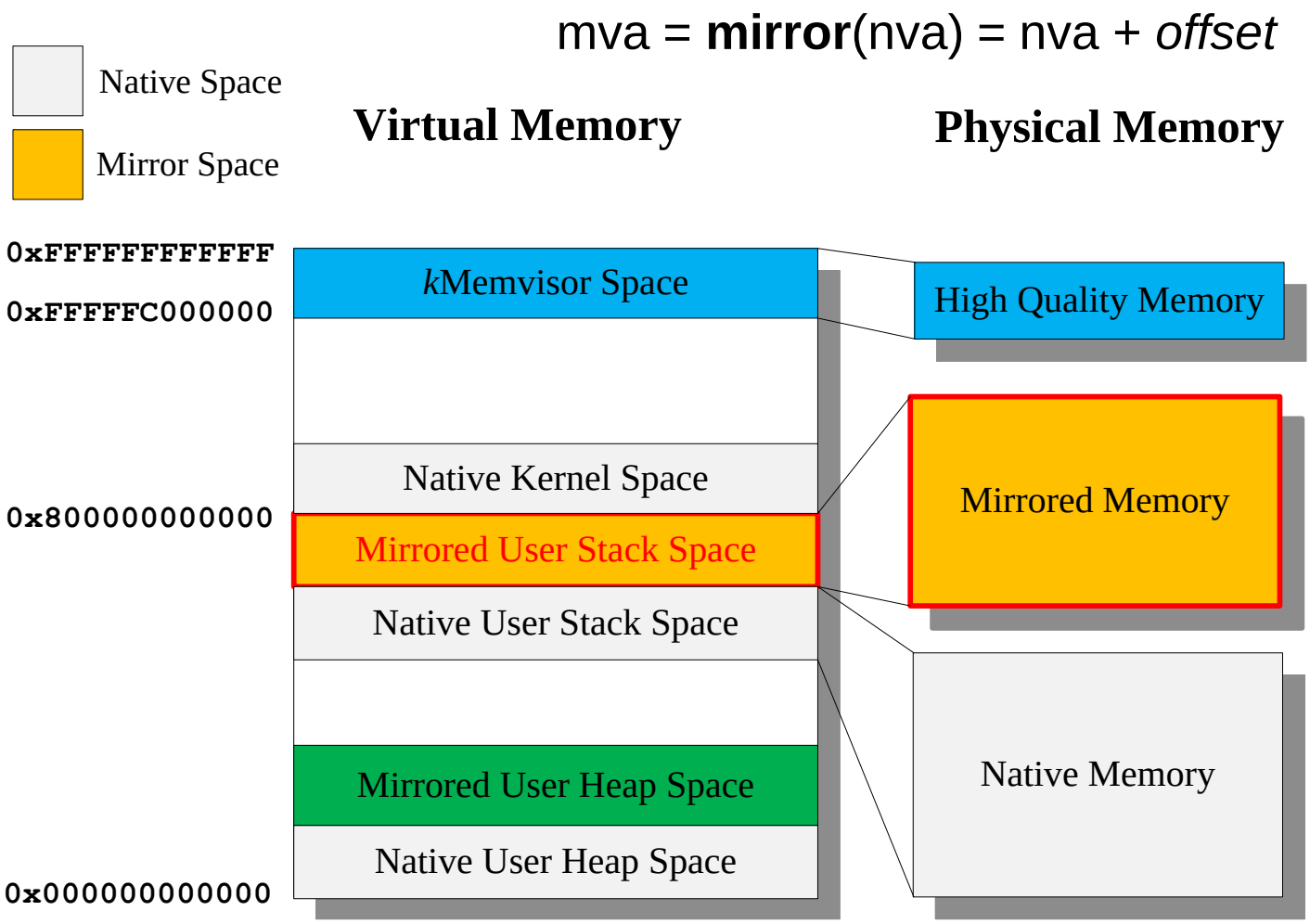
Modified Memory Layout



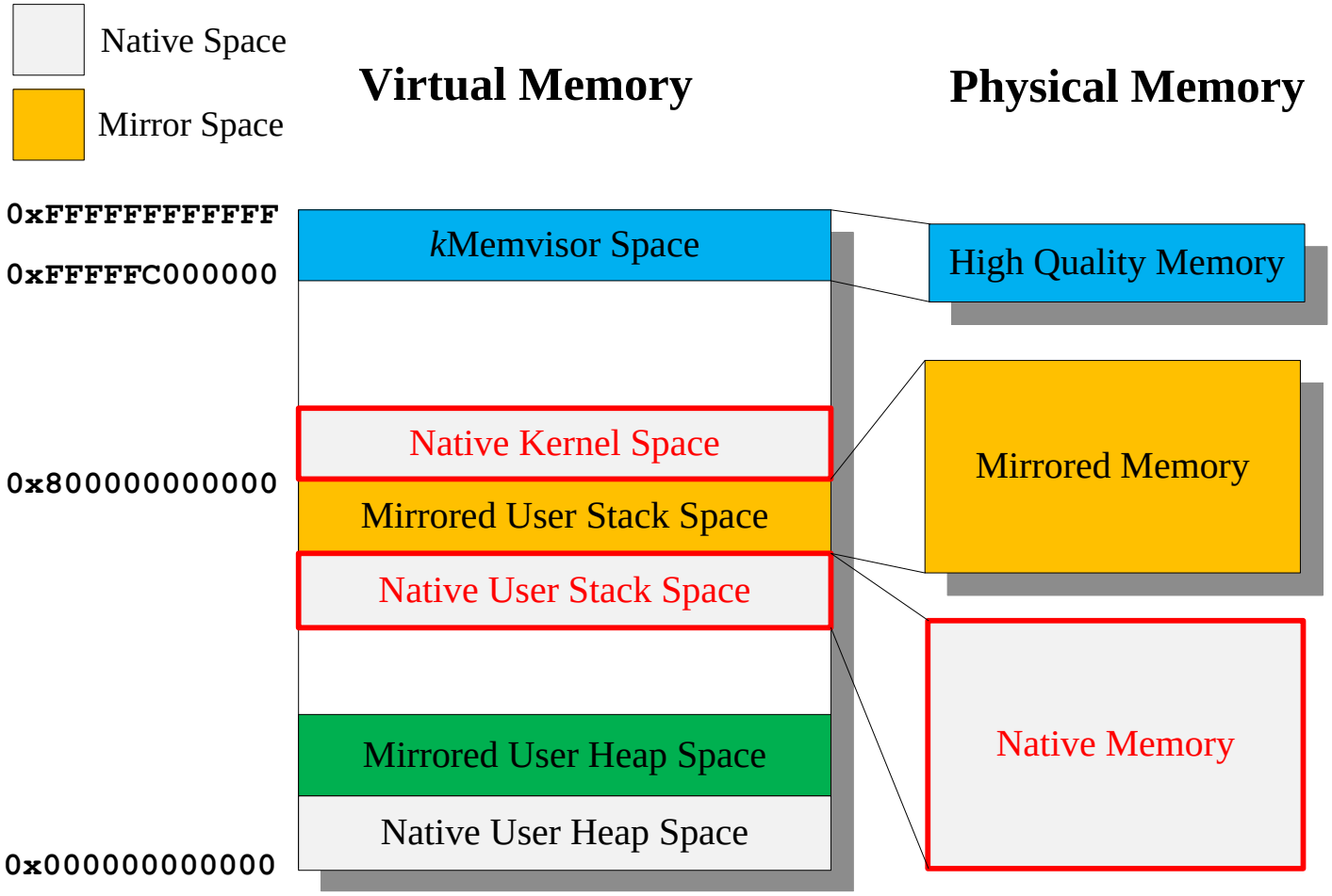
Modified Memory Layout



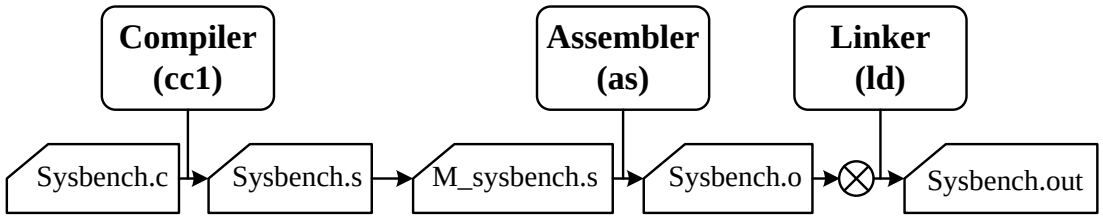
Modified Memory Layout



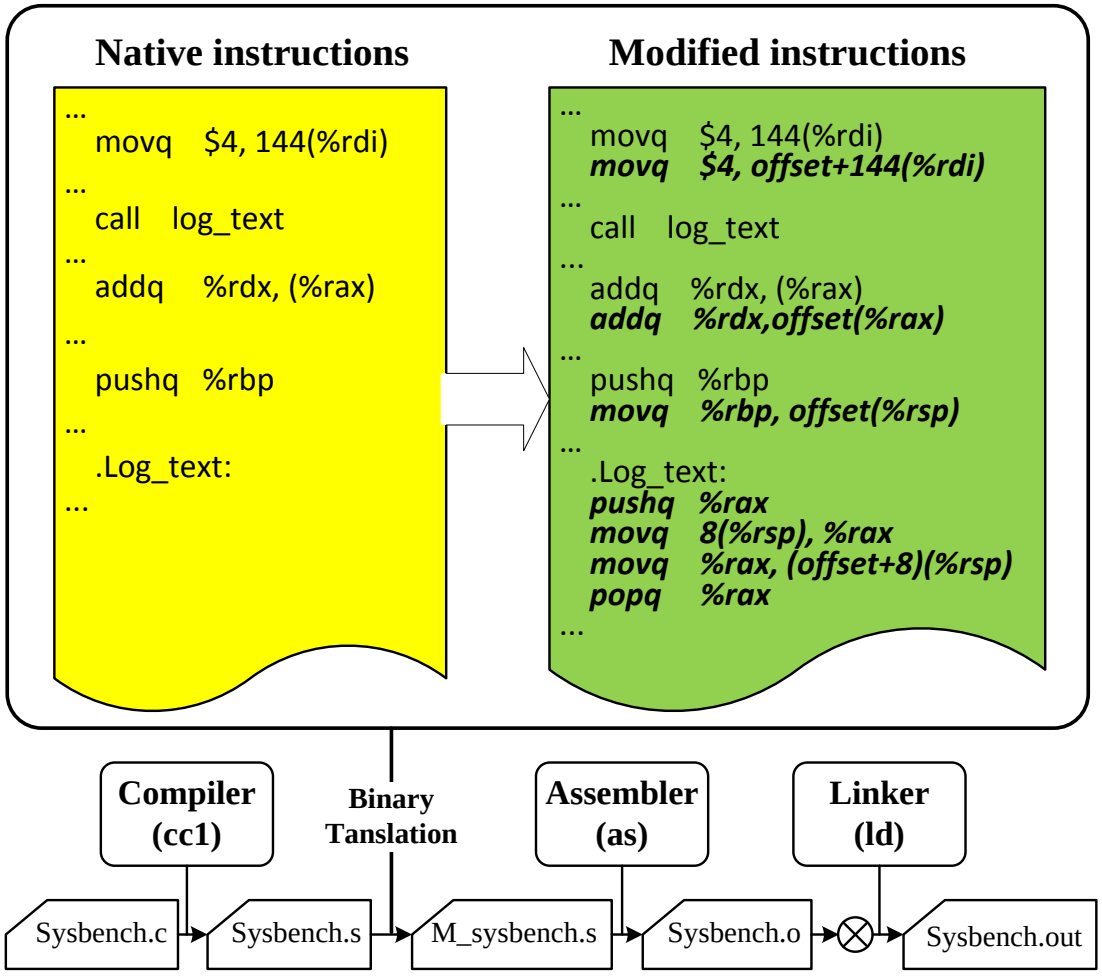
Modified Memory Layout



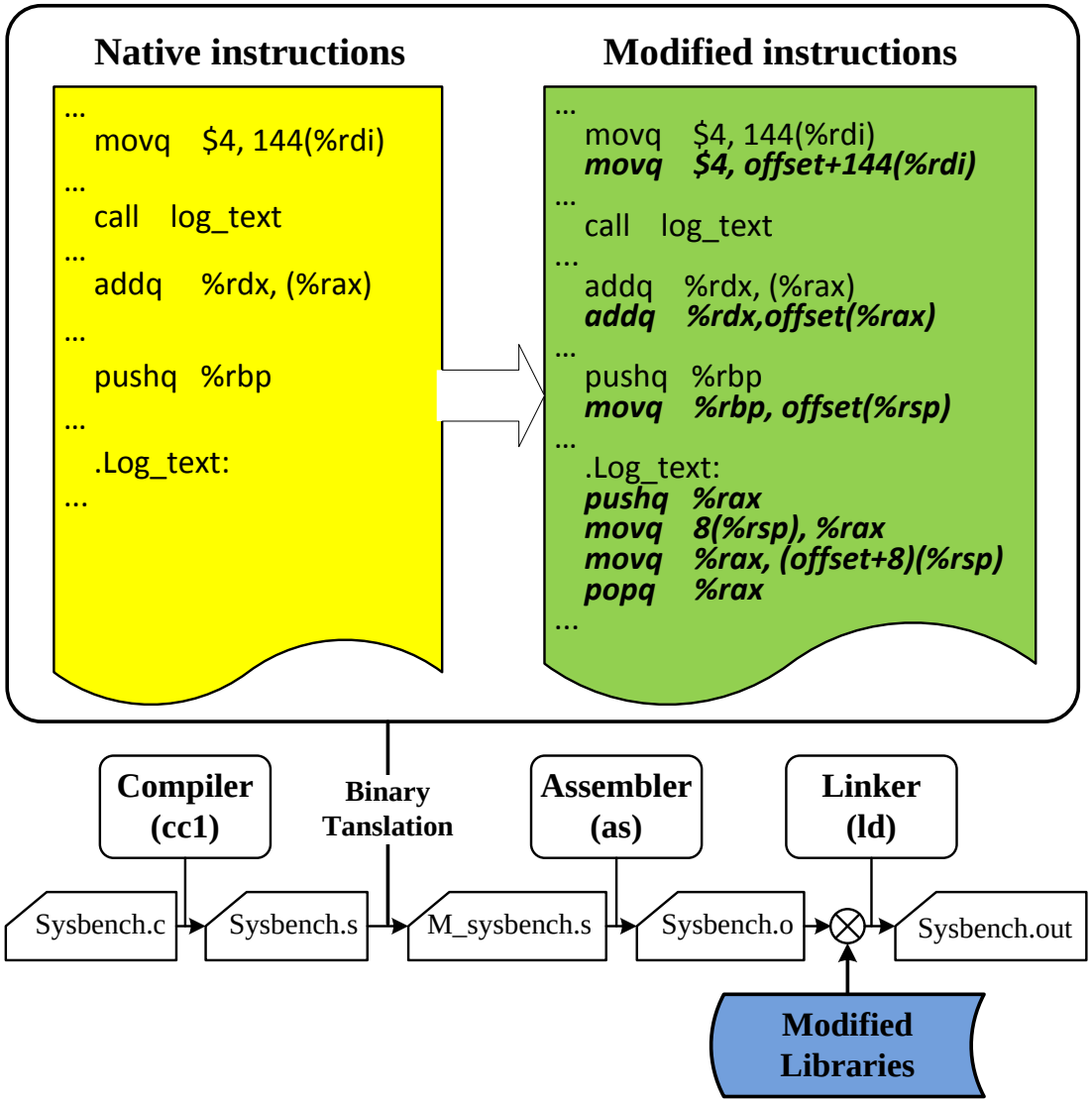
Memory Synchronization



Memory Synchronization



Memory Synchronization



Explicit and Implicit Instructions

```
……
call    grep
pushl   %eax
movl    %eax, offset(%esp)
movl    4(%eax), %eax
movl    %eax, offset+4(%esp)
popl    %eax
……
```

Explicit and Implicit Instructions

```
……
call    grep
pushl   %eax
movl    %eax, offset(%esp)
movl    4(%eax), %eax
movl    %eax, offset+4(%esp)
popl    %eax
……
```

Would not be executed untill call returns causing data inconsistence

Explicit and Implicit Instructions

```
……
call    grep
pushl   %eax
movl    %eax, offset(%esp)
movl    4(%eax), %eax
movl    %eax, offset+4(%esp)
popl    %eax
……
grep:
pushl   %eax
movl    %eax, offset(%esp)
movl    4(%eax), %eax
movl    %eax, offset+4(%esp)
popl    %eax
…...
```

Would not be executed untill call returns causing data inconsistence

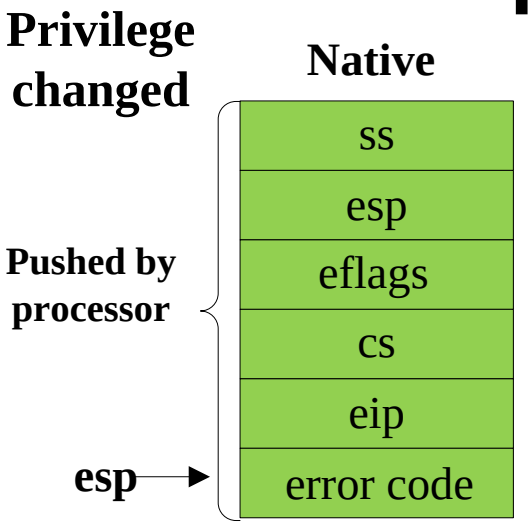
Explicit and Implicit Instructions

```
……
call    grep
pushl   %eax
movl    %eax, offset(%esp)
movl    4(%eax), %eax
movl    %eax, offset+4(%esp)
popl    %eax
……
grep:
pushl   %eax
movl    %eax, offset(%esp)
movl    4(%eax), %eax
movl    %eax, offset+4(%esp)
popl    %eax
…...
```

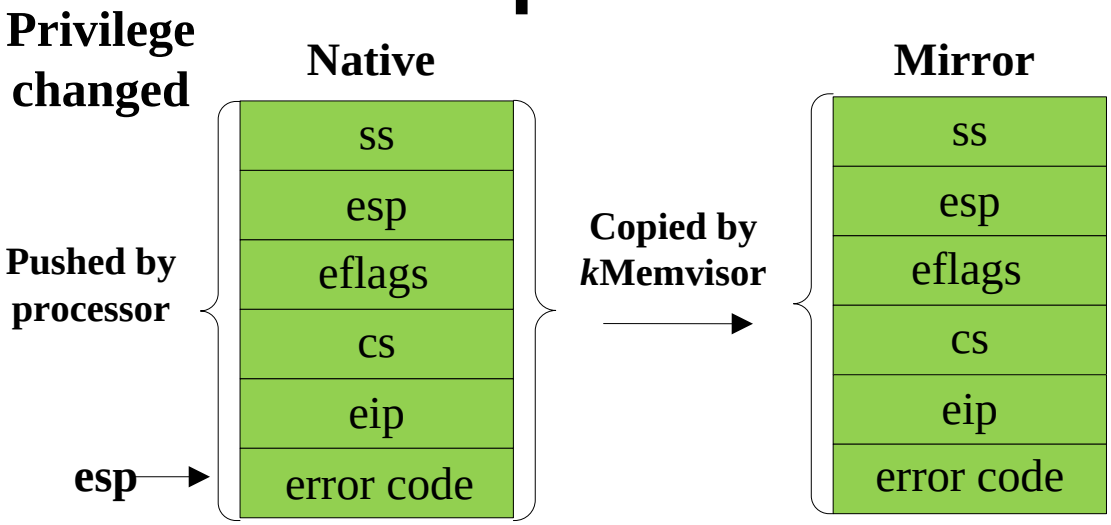
Would not be executed untill call returns causing data inconsistence

Instrumenting mirror instructions at the start of the called procedure

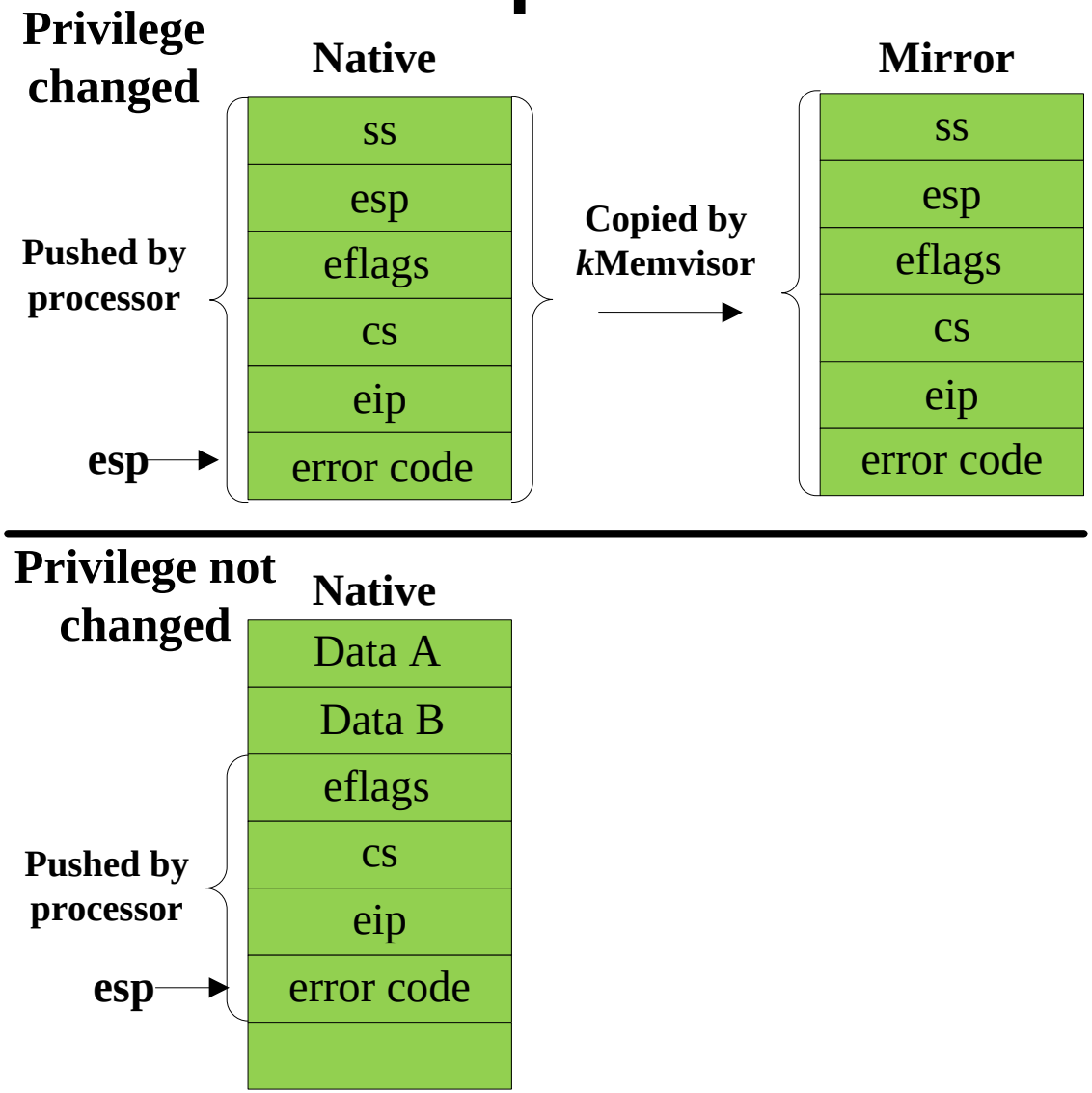
The Stack After An *int* Instruction Completes



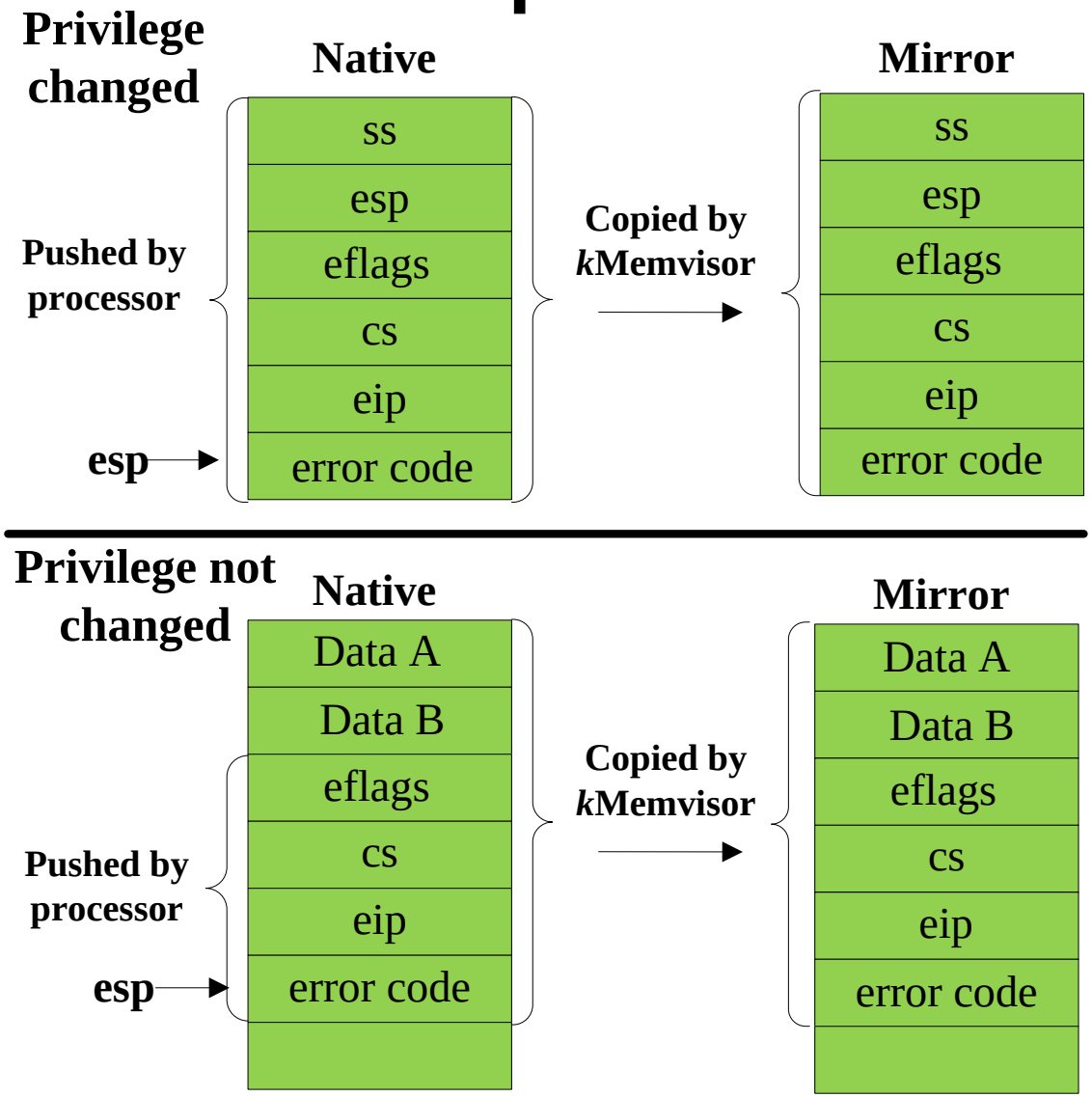
The Stack After An *int* Instruction Completes



The Stack After An *int* Instruction Completes



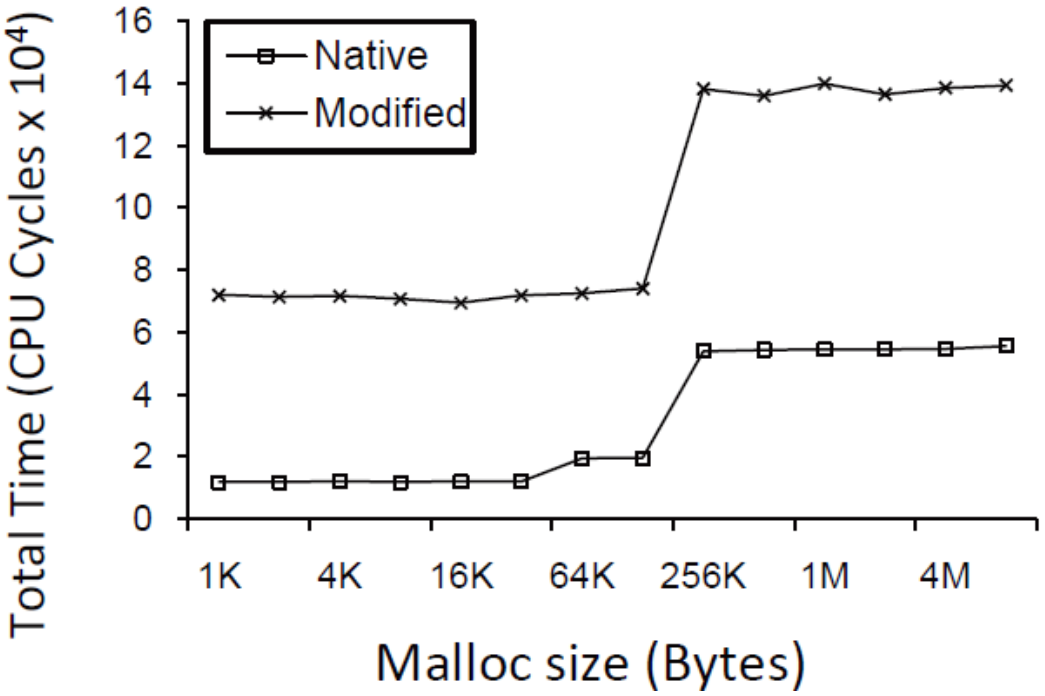
The Stack After An *int* Instruction Completes



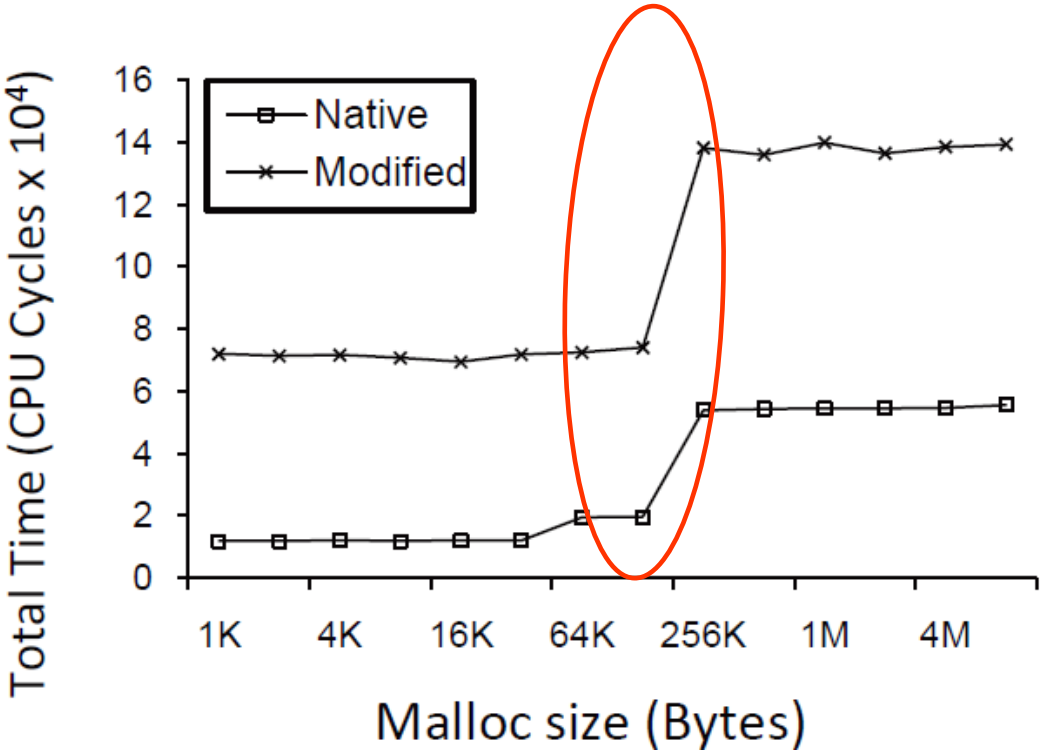
Test Environment

- Hardware
  - *Dell* PowerEdge T610 server：6-core 2.67GHz Intel Xeon CPU with 12MB L3 cache
  - *SamSung* 8 GB *DDR*3 RAMs with ECC and a 148 GB *SATA* Disk
- Software
  - Hypervisor: Xen-3.4.2
  - Kernel: 2.6.30
  - Guest OS: Busybox-1.19.2

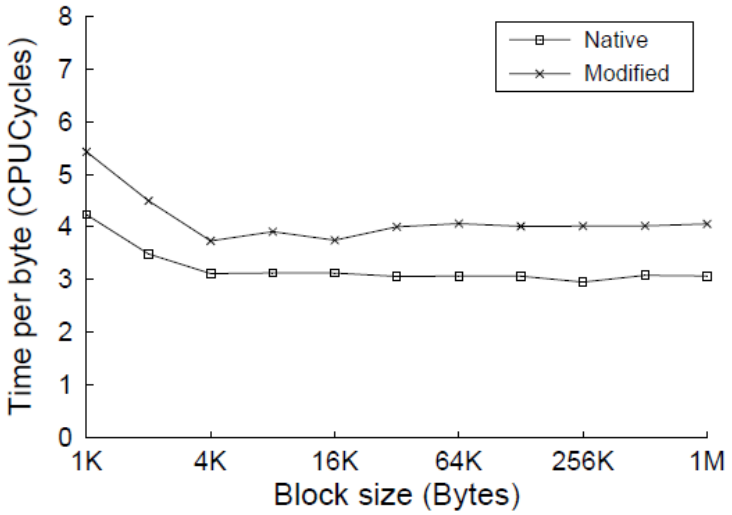
Malloc Micro-test with Different Block Size



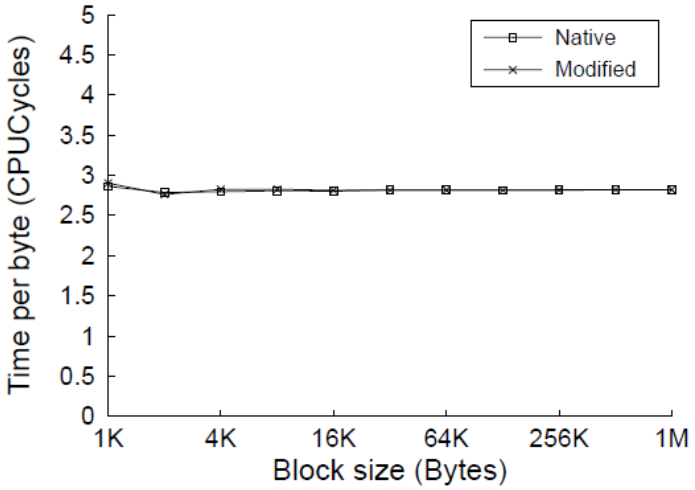
Malloc Micro-test with Different Block Size



Sequential Read & Write

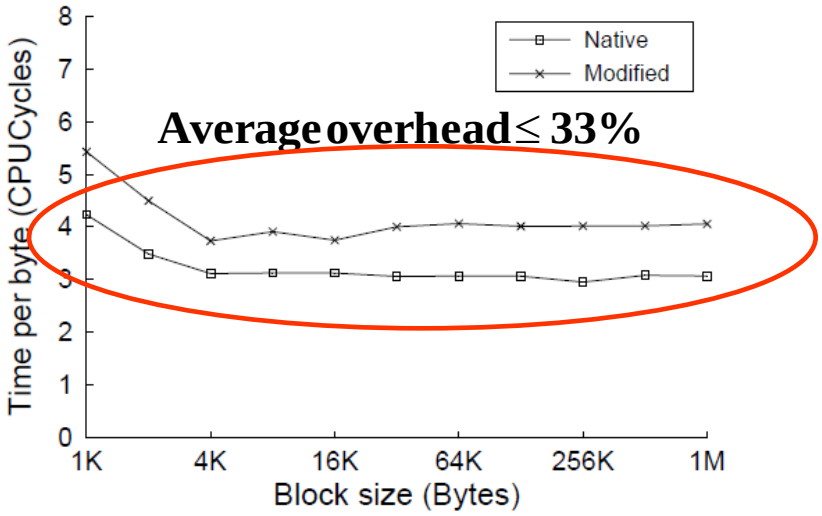


Overhead for sequential memory write

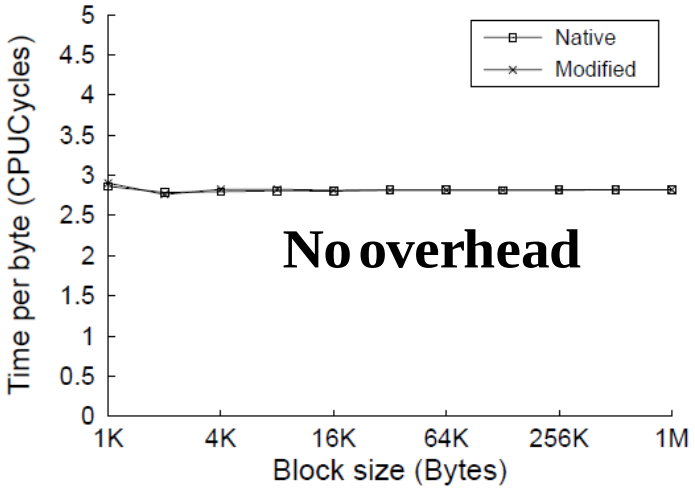


Overhead for sequential memory read

Sequential Read & Write

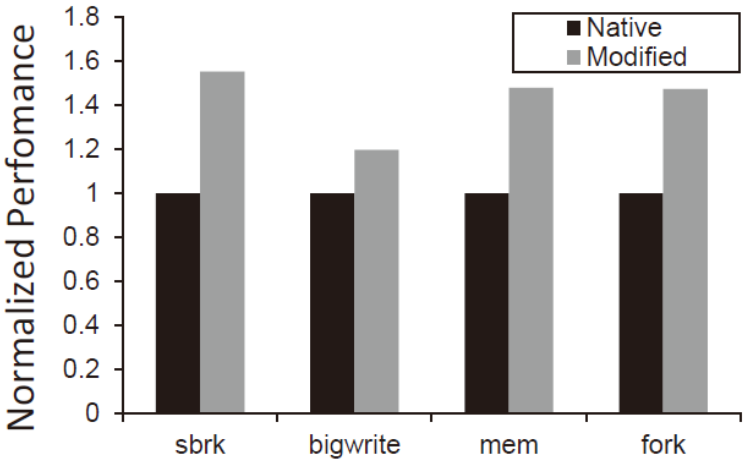


Overhead for sequential memory write

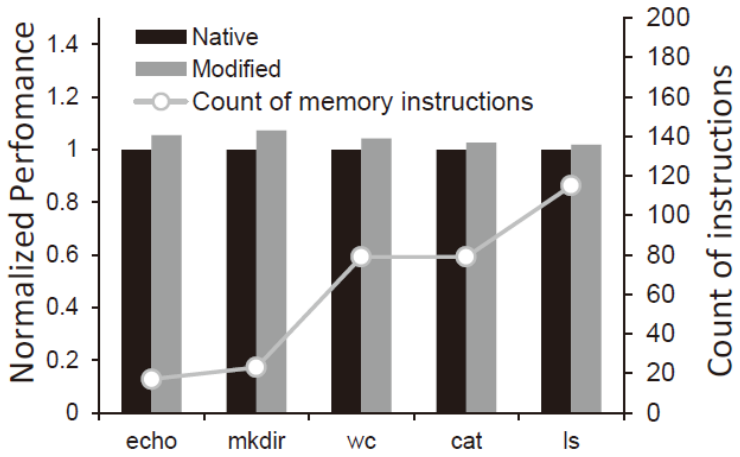


Overhead for sequential memory read

XV6 Benchmark

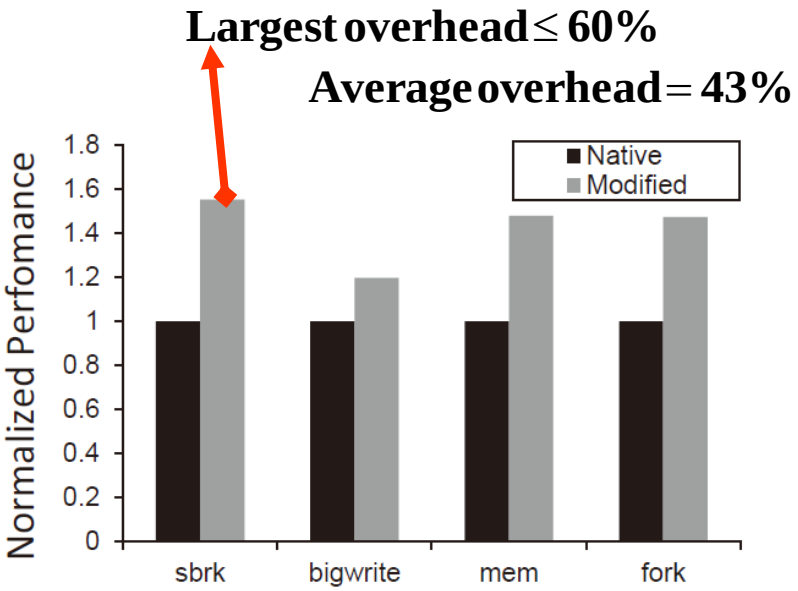


Usertests performance comparison

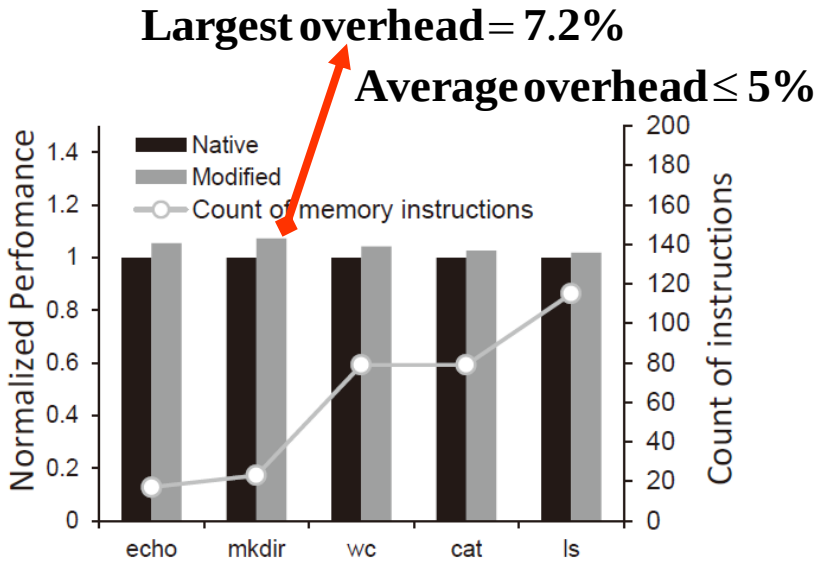


Command performance comparison

XV6 Benchmark

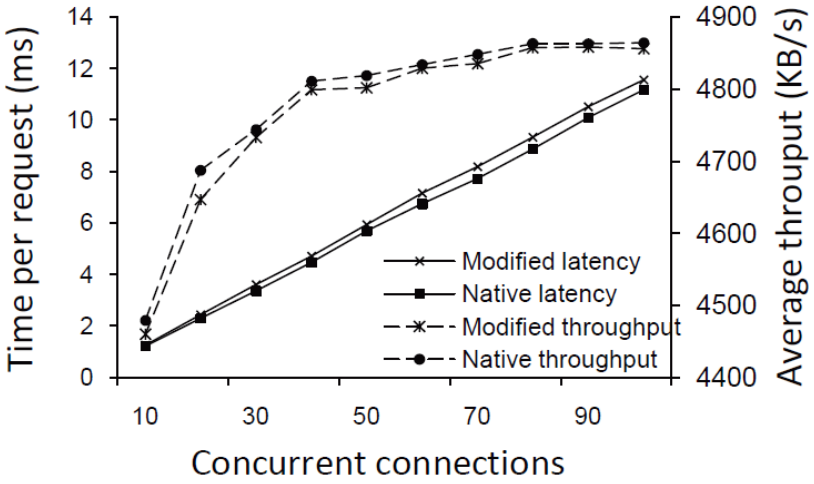


Usertests performance comparison

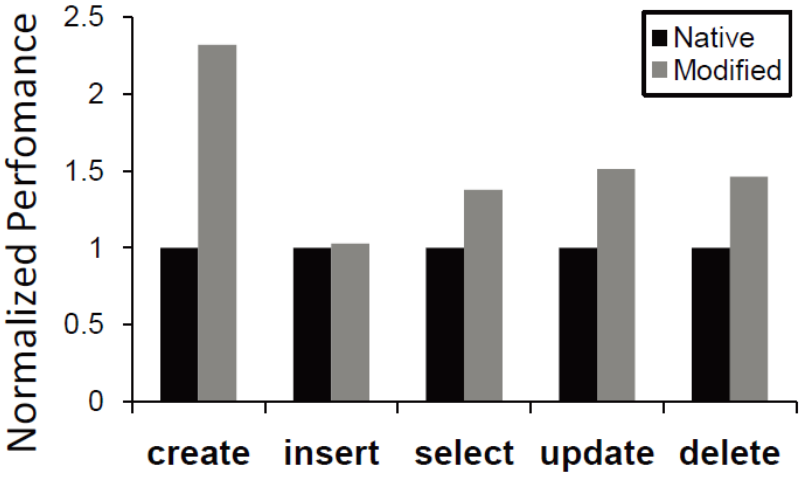


Command performance comparison

Web Server & Database

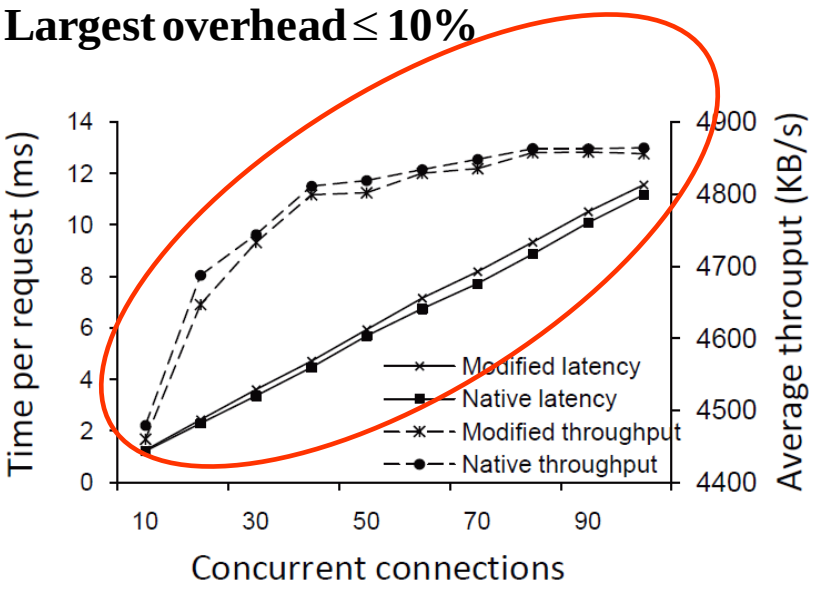


Performance of thttpd

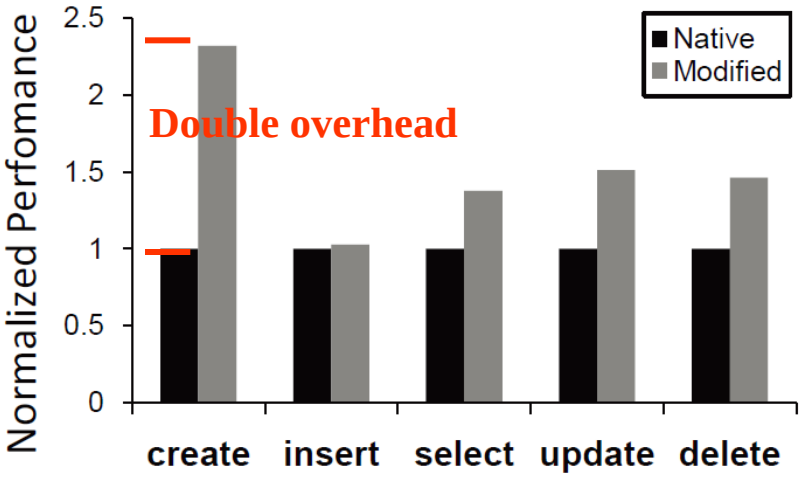


The performance impact for *create, insert, select, update*, and delete in SQLite

Web Server & Database

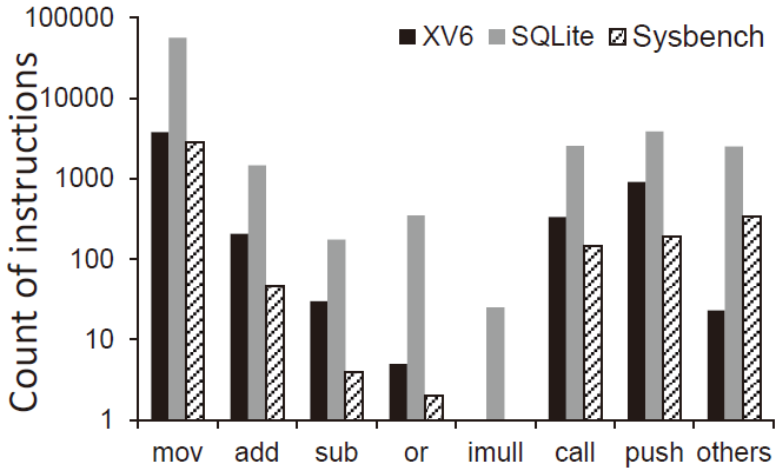


Performance of thttpd

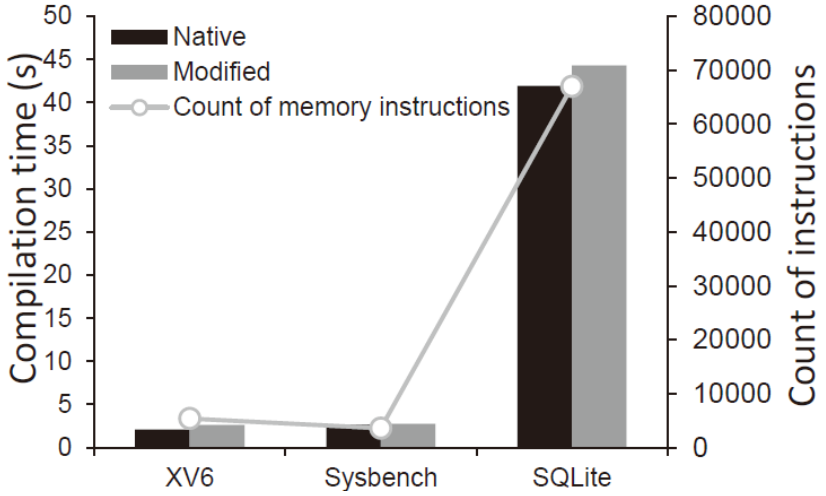


The performance impact for *create, insert, select, update*, and delete in SQLite

Compilation Time

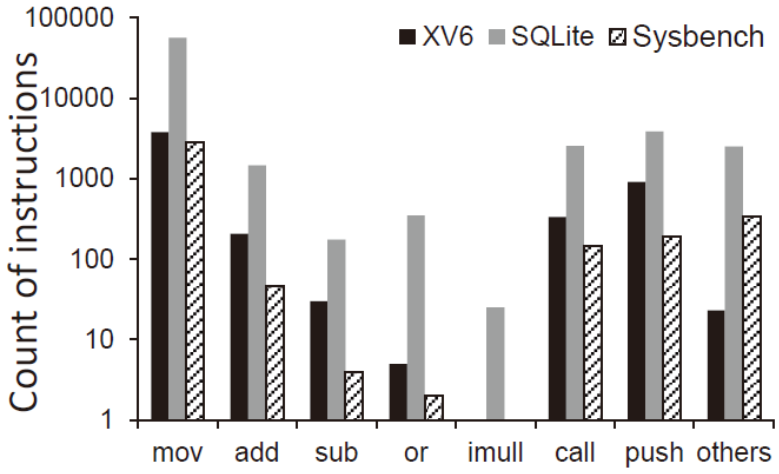


Mirror instructions detail information



Compilation time overhead brought by *k*Memvisor

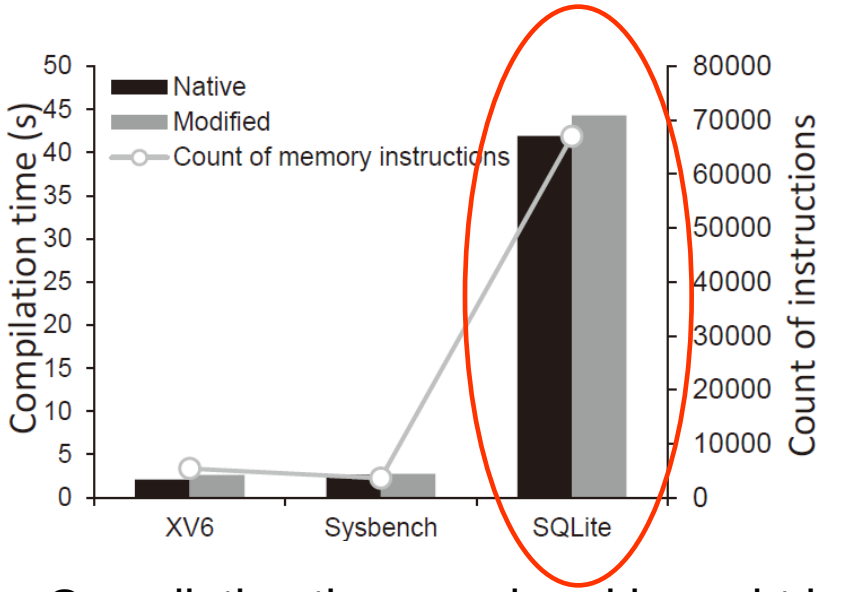
Compilation Time



Mirror instructions detail information

Average overhead = 6%

SQLite overhead = 5.64%



Compilation time overhead brought by *k*Memvisor

Discussion

- Special memory area
  - Page table area& hypervisor
- Other device memory operation
  - IO operation
- Address conflict
  - mva = nva+offset
- Challenge in binary translation
  - Self-modifying code & value
- Multi-threads and multi-cores
  - Multi-thread: emulate mirrored instruction
  - Multi-cores: explicit mutex lock

Conclusion

- A software mirrored memory solution
  - CPU-intensive tasks is almost **unaffected**
  - Our stressful memory write benchmark shows the backup overhead of **55%**.
  - Average overhead in real world applications **30%**

- Dynamic Binary Translation
- Full kernel mirroring