We choose case IIIC. The trade-off here is good, and when no virtualization is needed we can write microkernel raw applications. Rational:

IA, IIA are horrible in terms of TCB and native application support capability. IIA, IIB requires horrible ASM for each arch. IIIA is horribly slow. That said, what’s left in the competition here is IB, IIIB and IIIC, where IIIB simply outclasses IB. This leaves IIIB and IIIC, which are just two modes that can switch over to each other.

For those who thinks furious ASM is not an issue, IIB could be taken into account. In that case, IIB vs IIIB is 2-2 vs 2-3, not a clear win, particularly when the ‘1’ is on the not-that-important exit path. However, the implementation of IIIA and IIIC requires some kernel hack (exposing context to user-level), so kernel context restoring routines need to keep an eye on that. Luckily, this is easy on single-core and can be checked with C. On multi-core, have to check when they actually return, in ASM.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CASE | IA | IB | IIA | IIB | IIIA | IIIB | IIIC |
| TCB/LW | K | U | K | U | U | U | U |
| ASM | A | A | X | X | C | C | C |
| NATIVE |  |  |  |  |  |  |  |
| LATENCY | 1 | 2 | 1 | 2 | 3 | 2 | 1 |
| AVERAGE | 2 | 4 | 1 | 2 | 5 | 3 | 4 |

Case IA: Single-Context, IRET atomic, kernel-level hypervisor

----------------------------------------------------

HYP modify PC PREV -HYP(K)-> VM

VM push rest - ASM

VM call handler

VM pop rest - ASM

VM IRET - modify SP, etc, into place VM -HYP(K)-> VM

// TCB too large, no native OS support

Case IB: Single-Context, IRET atomic, user-level hypervisor

----------------------------------------------------

HYP modify PC PREV -> HYP(U) -> VM

VM push rest - ASM

VM call handler

VM pop rest - ASM

VM IRET - modify SP, etc, into place VM -> HYP(U) -> VM

// Crazy slow for average case, ASM needed

Case IIA: Single-Context, ENABLE/RET separate, kernel-level hypervisor

----------------------------------------------------

HYP modify PC PREV -HYP(K)-> VM

VM push rest - difficult ASM

VM call handler

VM pop rest - difficult ASM

VM ENABLE

\*\*\* CRITICAL REGION \*\*\*

VM RET

// TCB too large, no native OS support, very difficult ASM to write for new architectures

Case IIB: Single-Context, ENABLE/RET separate, user-level hypervisor

----------------------------------------------------

HYP modify PC PREV -> HYP(U) -> VM

VM push rest - difficult ASM

VM call handler

VM pop rest - difficult ASM

VM ENABLE

\*\*\* CRITICAL REGION \*\*\*

VM RET

// no native OS support, very difficult ASM to write for new architectures

Case IIIA: Double-Context, no context sharing : total = 5 kctxsws, latency = 2 kctxsws

----------------------------------------------------

HYP SND PREV -> HYP(U) -> VMINT

VM read context VMINT -> VMINT

VM push rest - C

VM call handler

VM pop rest - C

VM write context VMINT -> VMINT

VM RCV VMINT -> VMUSR

// Crazy slow for average case, crazy long latency

Case IIIB: Double-Context, context sharing : total = 3 kctxsws, latency = 2 kctxsws

----------------------------------------------------

HYP SND PREV -> HYP(U) -> VMINT

VM push rest - C

VM call handler

VM pop rest - C

VM RCV VMINT -> VMUSR

// Slightly slow for average case, long latency

Case IIIC: Double-Context, context sharing, secure tophalf : total = 4 kctxsws, latency = 1 kctxsws

----------------------------------------------------

DRV GET PREV -> DRV

HYP SND DRV -> HYP(U) -> VMINT

VM push rest - C

VM call handler

VM pop rest - C

VM RCV VMINT -> VMUSR

// Slow for average case, good latency