COMP9242 Paper 1 Review

Z5141448

August 24, 2019

1 Summary

This paper demonstrates a new way of IPC (Inter-Process Communication), which is using a passive hypervisor. The SkyBridge is a RootKernel which is the passive hypervisor plus a modified kernel (so-called SubKernel). When the client is doing an IPC, the client invokes the new direct_server_call to change the address space to the server and run the function. The direct_server_call doesn't invoke the kernel (syscall); it invokes the hypervisor instead (hypercall). Thus it reduced two context switch and address space change.
The server register handler function to SubKernel, which trigger the RootKernel record the function-pointer and EPT (Extended Page Table). Also, the client needs to register to get the ability to call the remote function the RootKernel copy both the client's EPT and server's EPT in RootKernel's data structure. The SkyBridge maps a shared buffer to both server and client. When invoking the IPC, there's an auto-generated 8-byte key for client identification. Then the RootKernel change EPT to change address space and use Trampoline to run the remote function. The return is the reverse procedure as described. The performance issues are addressed by not exit the VM (Virtual machine), let SubKernel handle all the privileged instruction. The security issue is address by rewrite all the VMFUNC instruction in the application before loading to memory.

2 Pros

- The result is auspicious
- Depth in technical point
- Clean and clear writing style
- Ask a good question at the correct time and leading reader
- Citation are good quality and related

3 Cons

- No clarification on breaking fundamental assumption of the process (thread)
- Too technical to follow, need to find more information while reading

4 Criticism of the work

4.1 Breaking Assumption of Process/Thread

A thread always assumes there's only one thread run at a particular time. This method is making a thread running in multiple cores at the same time while the original server thread is blocked forever. Here's the proof of it:

Test Name	Flasco Native	SkyBridge	Claimed	Calculated Boost
Single Core	2717	396	5.86	5.8611
Cross Core	8440	396	20.31	20.3131

Hence the cross core and single core use the same SkyBridge baseline. Because the same function instance is running in multiple cores at the same time, there's no cross core needed. Connection count can control how much thread in the handler which can restrict the need of a lock or maintain the original thread's assumption.

Moreover, there's no clue what happens if the server is running out of connection or there's a VM fault in the client while running a server's code. The server handler suddenly becomes multi-threaded and need to be thread-safe to gain the performance boost. Last but not least, the original server thread blocked forever and have no clue how much times the function has run and when it has run.

4.2 Discrete Capability System

There's two Capability System in the SkyBridge, and there's no way to merge back together. One is the original Capability system inside SubKernel and the other is the Capability in the RootKernel for IPC. Actually, it should be called as RPC (Remote Procedure Call) more correctly. Since the RPC is designed to not invoke kernel at all to gain the performance boost, there's no way to merge both Capability Space together.

4.3 Potential Security Issue

There may be a timing channel since it shares more resource and only uses 400 cycles to do an IPC. Hard to verify because there's hypercall and discrete Capability Space and the Root Kernel has as much code as seL4 kernel.

5 Points that Must be Addressed if Accepted

Overall, it's a good paper, and there's only some minor issue since the performance boost is significant. We do need to break some assumption to get better IPC performance as fast-path of seL4 does.
Claiming there's a new programming model or a new way to do the RPC instead of IPC.