Problem

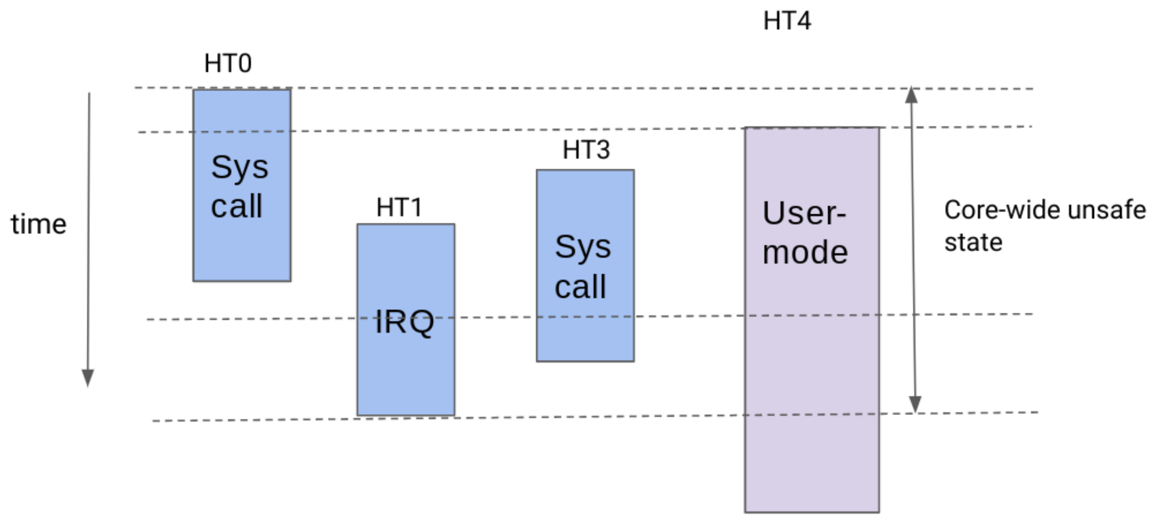
Hyperthreading has been disabled on many Intel processors due to vulnerabilities such as RIDL and ZombieLoad. While there are mitigations proposed to protect user-mode processes by making the OS scheduler aware of what is trusted and what is not, there is no known protection of processes in kernel-mode. The OS kernel can contain sensitive data that an attacker can leak on vulnerable processors. This invention addresses the problem.

Related Technology

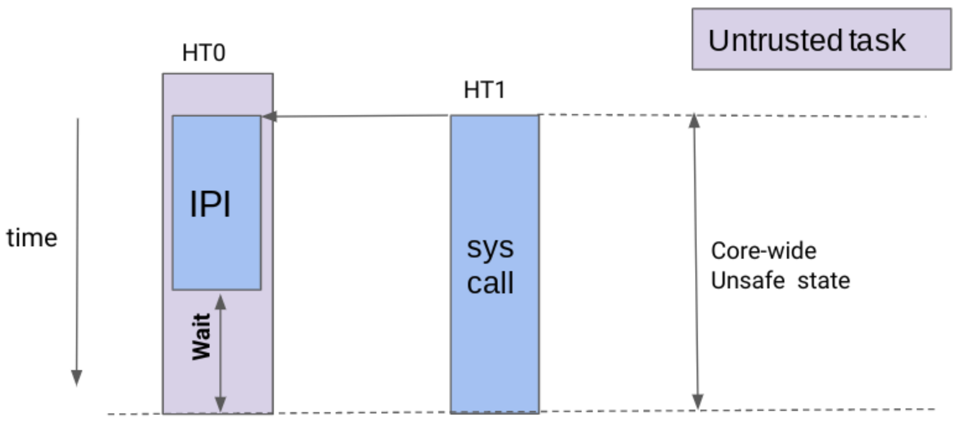
- Core-scheduling are changes to scheduler that protect user-mode processes from attack. Article: https://lwn.net/Articles/780703/.

Inventions

- The invention uses per-cpu and core-wide counters to track the state of when the whole core enters and exits the kernel. In the below example, we have 4 Hardware Threads (HT) per-core. HT0 and HT3 are making system calls. HT1 is in an IRQ. All these 3 HTs are in the kernel for these reasons. By making use of a per-cpu counter, we accurately detect when the core as a whole is in the kernel or not. We then know the exact points in time that we need core-wide protection from attackers.

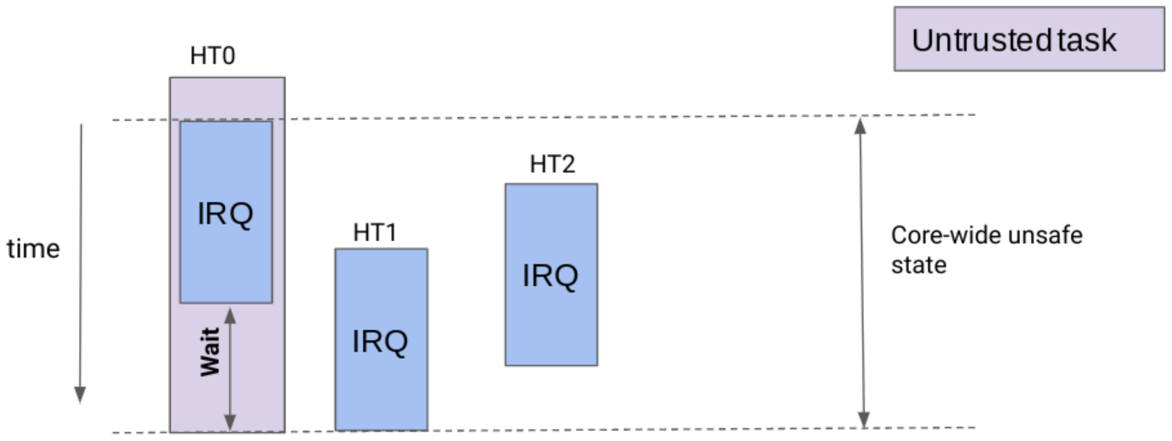


- The invention sends an inter-processor interrupt (IPI) when the core enters the unsafe state. The IPI is sent from the HT responsible for entering the core-wide unsafe state. The receiver of the IPI will be forced to busy-wait until the sender exits the unsafe state. In the below example, HT1 enters a system call and sends an IPI to HT0 which waits until HT1's completion of the system call.

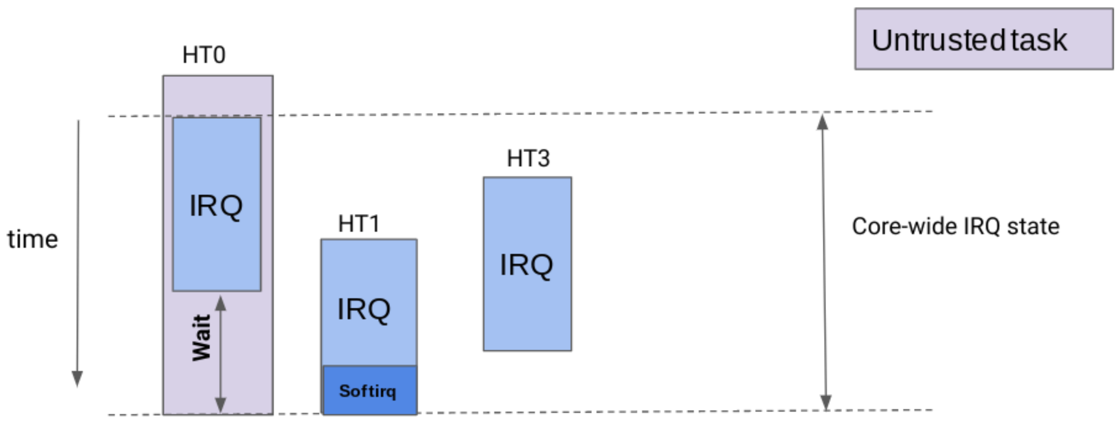


- The invention optimizes the sending of IPIs thus reducing overhead. It does so by making sure that only the outer-most entry into the core-wide unsafe state sends an IPI.

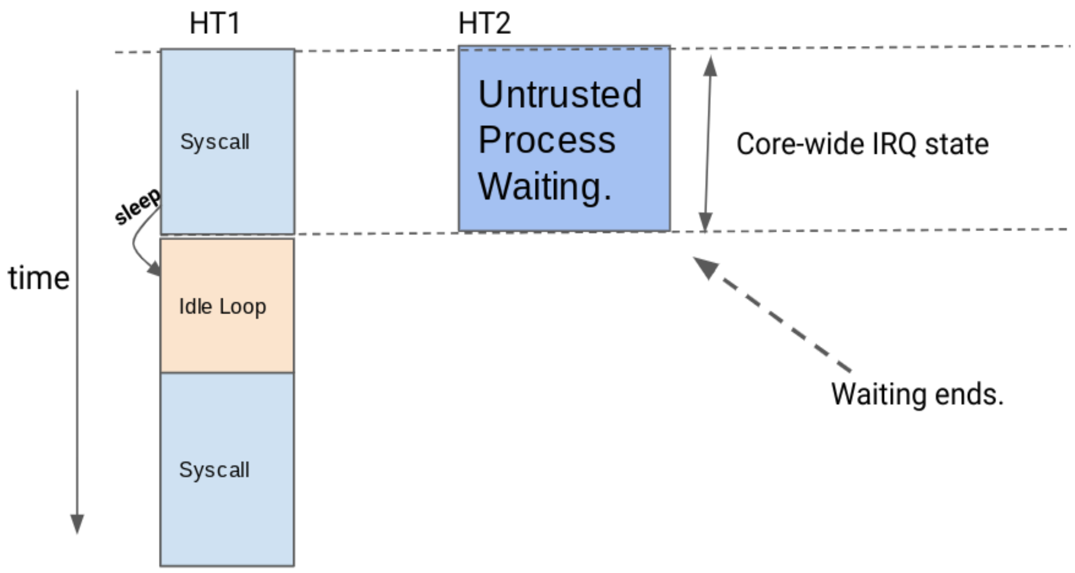
- The invention avoids sending IPIs at all in the case where the Untrusted task itself entered the core-wide unsafe state, while all other CPUs were idle. For example, in the below diagram HT0 does send any IPIs to other HTs which are idle. Further, HT0 waits for other IRQs that started **after** it without requiring any IPIs through out the process.



- By design the invention ensures that soft-interrupts are also protected as soft interrupts nest within IRQs and system calls.

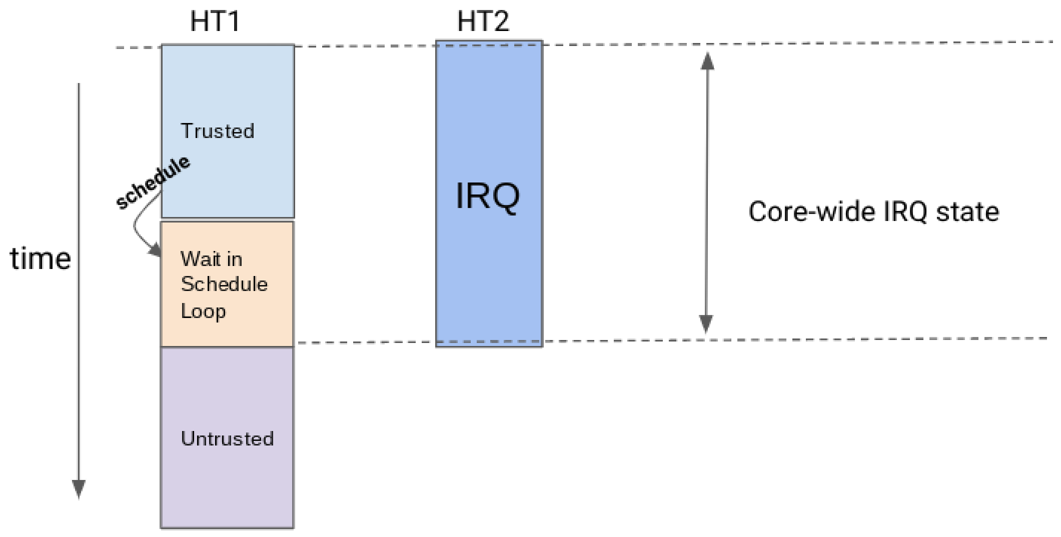


- The invention modifies the idle loop to enter and exit unsafe sections. This ensures that we exit the core-wide IRQ state as soon as possible, and that the untrusted process does not wait for a long time.



-

- The invention ensures that waiting in the schedule-loop is not needed when switching from a privileged to an unprivileged task. For example, the following waiting is eliminated by design. Furthermore, the invention avoids sending any IPIs if the receiver HT is in trusted user-mode context such as system daemons, or is idle. As illustrated below:


HT1
HT2
time
Trusted
schedule
Wait in Schedule Loop
Untrusted
IRQ
Core-wide IRQ state