Impact of Priority Inversion on Frame Jank

Lei Li[1*], Yu Liang[1], Rachata Ausavarungnirun[2], Tei-Wei Kuo[3,4], Chun Jason Xue[1]
[1]City University of Hong Kong [2]King Mongkut's University of Technology North Bangkok
[3]National Taiwan University [4]Mohamed bin Zayed University of Artificial Intelligence

1 Introduction

Priority inversion problem refers to a situation when a thread with higher priority is blocked by a lower-priority thread for an unbounded period of time. This problem occurs in applications that require synchronization primitives such as mutex, semaphores, and seqlocks. This is a critical and intolerable problem in the real-time system since the system has to ensure the bounded running time according to priority [1, 3]. However, the impact of priority inversion in common OS like Android is inconclusive. Some researchers think priority inversion can be ignored in common OS [1], but this claim is not well-evaluated.

In this study, we explore the occurrence of priority inversions in Android smartphones and argue its negative impact on user experience in CPU-intensive workloads. Specifically, we quantify user experience with frame jank, which happens when a frame fails to be generated and displayed on the screen within 16.7 ms. When frame jank happens, the user perceives slow UI rendering and flickers on the screen. We observe up to 216 ms frame jank on smartphones caused by priority inversion, which indicates that priority inversion can be harmful to the user experience and should be limited.

2 Experiments and Observations

We measure priority inversions on two commercial Android smartphones (i.e., Google Pixel 3 and Google Pixel 5) and focus on priority inversions caused by kernel mutex.

Observation 1: priority inversion in smartphones is not problematic in usual cases. In common cases (i.e., we use 9 APPs before each test), priority inversion rarely happens. Our experimental results show 10s of (at most a few hundred of) priority inversions can lead to blocks over 1 ms, in a five-minute test. Moreover, the maximum block time of most foreground APPs (i.e., Youtube, Facebook, Angrybird AR, Arena of Valor, PUBG, and Google Maps) is lower than 11 ms, and only the maximum block time of Youcame Makeup AR is 20 ms. Since user experience is critical for smartphone users, we selectively measured a priority inversion type, which is caused by blocked RenderThread. Theoretically, this priority inversion type can affect the rendering of frames and are likely to cause frame janks. However, our experimental results show that this priority inversion type occurs rarely (i.e., less than one percent of RenderThread will cause the block on the higher priority thread, and the maximum block time is only 10 ms).

Observation 2: priority inversion in smartphones can cause long frame jank with insufficient CPU resources. As mentioned above, the impact of priority inversions incurred by RenderThread is acceptable in usual cases. However, when we reduce the smartphone's CPU resources (Pixel 3 has 8 CPU cores by default), we observed the block time of RenderThread-incurred priority inversions increased significantly as shown in Table 1. The longest block time can be up to 216 ms, or over 13 frames are blocked, making the user perceive a long frame jank. Besides, there are 10s of priority inversion incidents that result in more than 16ms block time, which will also incur user-perceivable frame janks.

Table 1. Priority inversion with insufficient CPU resources. The second, third, and fourth columns show the number of priority inversions that lead to blocking time between 1ms-10ms, 10ms-16ms, and larger than 16ms, respectively. (testing when using Youcame Makeup AR without background APPs)

CPU cores	1-10ms	10-16ms	>16ms	Max block time(ms)
8 cores	194	0	0	6
4 small cores	749	39	28	78
2 small cores	493	43	77	216

More-complicated APPs (e.g., AR APPs and APPs supporting neural networks), split-screen mode, and multi-display Android systems (e.g., Microsoft Surface Duo and LG ThinQ Dual Screen) will strain CPU resources even more. All of these trends are believed to render priority inversion a more serious problem in the near future. Our preliminary results show screen-split on smartphones can triple the max block time caused by priority inversion.

3 Ongoing and Future Work

Currently, we tried to solve the RenderThread-incurred priority inversions using existing priority inheritance support in Linux kernel (rt_mutex [2]). Although rt_mutex can limit the max block time to around 1 ms when conducting the same experiments as Table 1, we also observed that rt_mutex has much larger overhead than mutex in our experiments. Systematically solving priority inversions may need a proper trade-off between mutex and rt_mutex.

References

[1] Andreu Carminati, Rômulo Oliveira, Fernando Luís, and Friedrich. 2012. Implementation and Evaluation of the Synchronization Protocol Immediate Priority Ceiling in PREEMPT-RT Linux. *Journal of Software* 7 (03 2012).
[2] Steven Rostedt. 2006. RT-mutex implementation design. https://docs.kernel.org/locking/rt-mutex-design.html. [accessed 5-March-2023].
[3] D. Silambarasan and M RamanathaVenkatesan. 2016. Handling of Priority Inversion Problem in RT-Linux using Priority Ceiling Protocol.

* student and presenter