Henry Baker

Professor Chiu

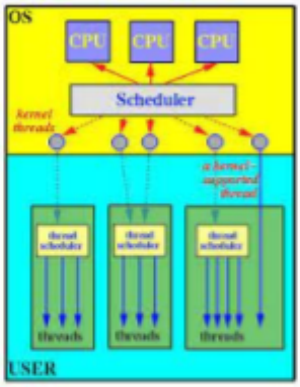
CS475: Operating Systems

May 2, 2023

Windows Vista Technical Report

Growing up, my parents essentially had the same computer in our living room for most of my young life. That system ran on Windows Vista, and I had always heard the rumors of it being a shameful operating system in the long history of Windows OS's. So when I was provided the opportunity to do some research on any operating system of my choice, Windows Vista seemed like the perfect choice. Despite being considered to be a not great system overall, Windows Vista had many features that would go on to become essential parts of the operating systems that we use in 2023.

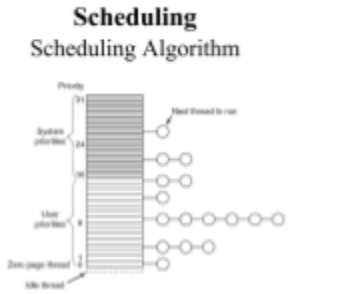
First was Vista's process / thread management, which I found to be incredibly interesting. Unlike how we discussed process states in class, Windows Vista does not assign states to its processes, but rather uses its threads to assign states and schedule the work to the CPU (done by the kernel) (https://www.slideshare.net/ptc1760/processes-threads-inwindowsvista). This means that a process is represented in the scheduler as a single threaded execution, with the ability to create more for itself as it needs them. The system also uses single execution paths called 'fibers' that are created within threads. These fibers are smaller units that must be scheduled by the threads that create them



(Fig. 1) Vista Thread Structure

(https://www.slideshare.net/ptc1760/processes-threads-inwindowsvista), essentially acting like the user-threads that we discussed in class. Each process also creates two separate stacks to run in both user-mode and kernel-mode.

Next is Windows Vista's scheduling policy. The OS's scheduler, as mentioned earlier, schedules individual threads to run instead of entire processes. Its scheduling policy works very similar to what we discussed in class. Vista uses an MLFQ (Multilevel Feedback Queue) style system that is split into 32 different priority queues



(Fig 2.) Windows Vista Scheduling Diagram

(https://www.microsoftpressstore.com/articles/article.aspx?p=2233328&seqNum=7). The OS sets a base constant priority at runtime for each thread based on its class and priority level, and then dynamically adjusts the threads priority over time (https://www.slideshare.net/ptc1760/processes-threads-inwindowsvista). The scheduler uses a Round Robin approach within each of the priority queues, switching between threads at equal priorities based on a time quantum (https://www.microsoftpressstore.com/articles/article.aspx?p=2233328&seqNum=7). Since the OS uses an MLFQ, it is more biased towards CPU intensive tasks, as well as tasks that are manually assigned a higher priority, such as system threads over user threads. Thread starvation is inevitable with the MLFQ, however it has a unique solution to counteract starvation caused by priority inversion. The system will randomly boost the priority of ready priority threads in the hopes that if they are holding locks, they will run and eventually unlock critical sections for higher priority threads to access (https://www.slideshare.net/ptc1760/processes-threads-inwindowsvista).

Lastly is Vista's memory management, which functions very similarly to the concepts we discussed in class. The OS uses paging in order to dynamically load programs into the system memory from disk. Vista also uses counters built into the page files to privatize or publicize different pages, allowing for memory sharing between threads, if desired, and for detecting segmentation faults / memory leaks within the system (https://www.codeproject.com/Articles/29449/Windows-Memory-Management). Vista also implements virtual memory in almost the same way that we discussed in class, however it introduced a new function that previous versions of Windows had never implemented. Windows Vista uses a process known as SuperFetch in order to more efficiently load pages from memory. It works by keeping track of what programs and processes the user runs most frequently and preemptively loads pages from those tasks into system memory in hopes that the user will then access those immediately, helping to avoid early page faults during runtime (https://www.sony-asia.com/electronics/support/articles/S500027946). This was a vast improvement over the original version of this feature, known as PreFetch, as SuperFetch would try to constantly keep pages loaded for frequent use processes rather than only loading the most used pages upon an application launch (https://slideplayer.com/slide/8831005/).

Although Windows Vista had many features of the modern operating system that are still used today and introduced many new features, it was not without its criticisms. Perhaps the most of all was that it was almost too ahead of its time. Many users faced issues with their system hardware not being powerful enough to handle the resource requirements of the OS itself, leading to poor performance overall and many negative reviews (https://en.wikipedia.org/wiki/Windows_Vista#Criticism). Despite being limited by its time,

Windows Vista was a vast leap in the world of operating system technology and would be a critical stepping stone for newer systems like Windows 7 to build upon.

Works Cited

https://www.slideshare.net/ptc1760/processes-threads-inwindowsvista

(Fig 1.)

https://www.google.com/url?sa=i&url=http%3A%2F%2Fwww.hasanbalik.com%2FLectureNotes%2FOpSys%2FAssignments%2FMS%2520Windows%252010%2520Process%2520and%2520Thread%2520%2520Management.pdf&psig=AOvVaw29qNwY87FOMH_Qg5uJ1FHM&ust=1683153646671000&source=images&cd=vfe&ved=0CBAQjRxqFwoTCMjE9o7a1_4CFQAAAAAdAAAAABAD

(Fig 2.)

https://image.slidesharecdn.com/processesthreadsinwindowsvista-121018093726-phpapp01/75/processes-and-threads-in-windows-vista-44-2048.jpg?cb=1668216554

https://www.microsoftpressstore.com/articles/article.aspx?p=2233328&seqNum=7

https://www.codeproject.com/Articles/29449/Windows-Memory-Management

https://www.sony-asia.com/electronics/support/articles/S500027946

https://slideplayer.com/slide/8831005/

https://en.wikipedia.org/wiki/Windows_Vista#Criticism