Process Control Block

See the previous lecture

Process Schedulers in Operating System

Process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

There are three types of process schedulers.

1. **Long Term or job scheduler:**

   It brings the new process to the 'Ready State'. It controls the ***Degree of Multi-programming***, i.e., the number of processes present in a ready state at any point in time. It is important that the long-term scheduler make a careful selection of both I/O and CPU-bound processes. I/O-bound tasks are which use much of their time in input and output operations while CPU-bound processes are which spend their time on the CPU. The job scheduler increases efficiency by maintaining a balance between the two. They operate at a high level and are typically used in batch-processing systems.

2. **Short-term or CPU scheduler:**

   It is responsible for selecting one process from the ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn't load the process on running. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring there is no starvation owing to high burst time processes.

   The dispatcher is responsible for loading the process selected by the Short-term scheduler on the CPU (Ready to Running State) Context switching is done by the dispatcher only. A dispatcher does the following:
   1. Switching context.
   2. Switching to user mode.
   3. Jumping to the proper location in the newly loaded program.

3. **Medium-term scheduler:**

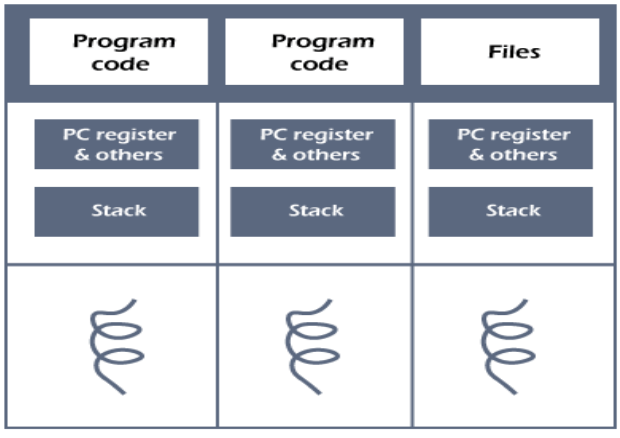
It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.

- **I/O schedulers:** I/O schedulers are incharge of managing the execution of I/O operations such as reading and writing to discs or networks. They can use various algorithms to determine the order in which I/O operations are executed, such as FCFS (First-Come, First-Served) or RR (Round Robin).

- **Real-time schedulers:** In real-time systems, real-time schedulers ensure that critical tasks are completed within a specified time frame. They can prioritize and schedule tasks using various algorithms such as EDF (Earliest Deadline First) or RM (Rate Monotonic).

Threads in Operating System (OS)

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.



Three threads of same process

The process can be split down into so many threads. **For example**, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.

Need of Thread:

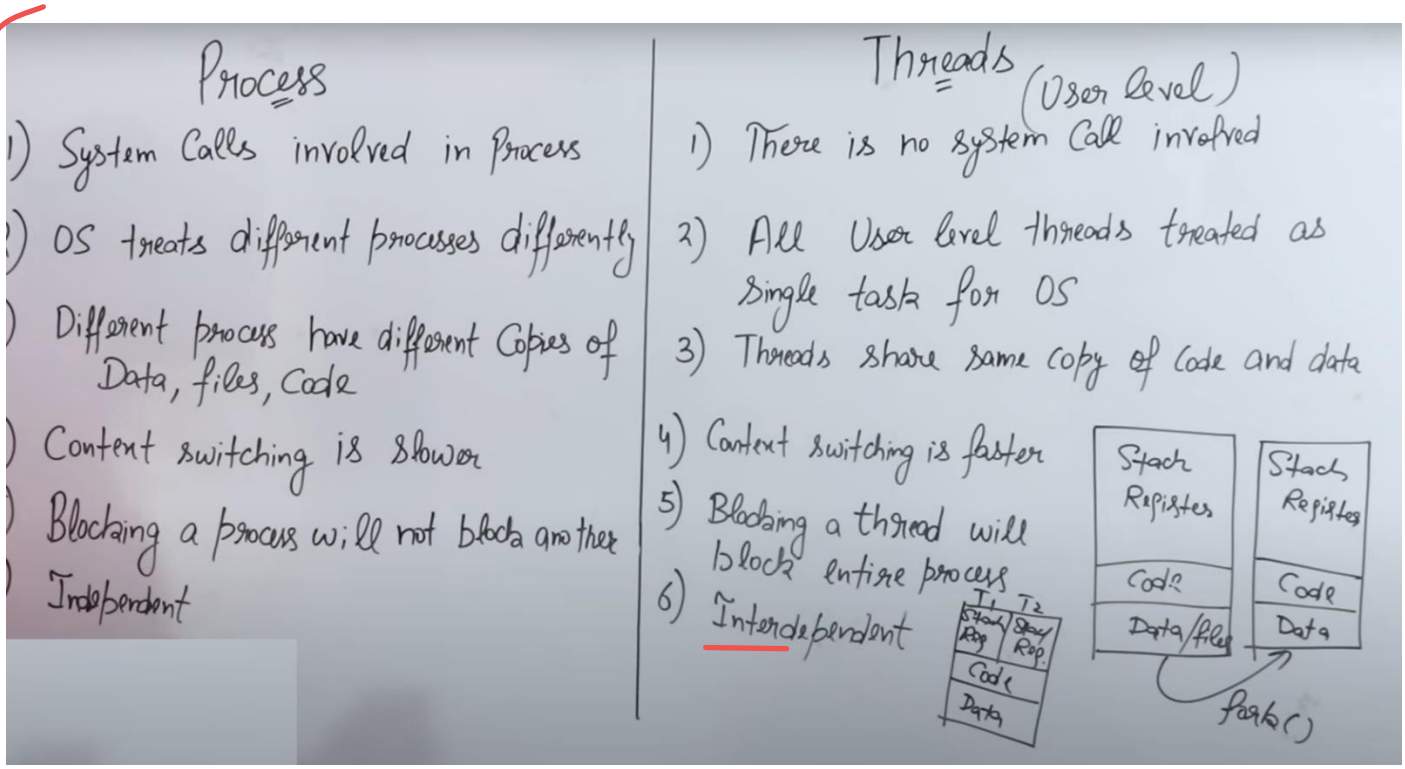
- It takes far less time to create a new thread in an existing process than to create a new process.
- Threads can share the common data, they do not need to use Inter- Process communication.
- Context switching is faster when working with threads.
- It takes less time to terminate a thread than a process.

Process vs Thread:

The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces. Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

Difference between Process and Thread

S.No	Process	Thread
1.	When a program is under execution, then it is known as a process.	A segment of a process is known as a thread.
2.	It consumes maximum time to stop.	It consumes minimum time to stop.
3.	It needs more time for work and conception.	It needs less time for work and conception.
4.	Context switching takes maximum time here.	Here, context switching takes minimum time.
5.	It is not that effective in terms of communication.	It is effective in terms of communication.
6.	It takes more resources.	It takes fewer resources.
7.	It is a heavy-weight process.	It is a lightweight process.
8.	If one process is obstructed then it will not affect the operation of another process.	If one thread is obstructed then it will affect the execution of another process.



Types of Threads

In the operating system, there are two types of threads.

1. Kernel level thread.
2. User-level thread.

User-level thread

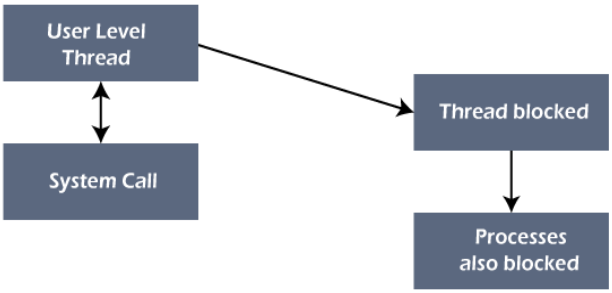
The operating system does not recognize the user-level thread. User threads can be easily implemented and it is implemented by the user. If a user performs a user-level thread blocking operation, the whole process is blocked. The kernel level thread does not know nothing about the user level thread. The kernel-level thread manages user-level threads as if they are single-threaded processes? examples: Java thread, POSIX threads, etc.

Advantages of User-level threads

1. The user threads can be easily implemented than the kernel thread.
2. User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3. It is faster and efficient.
4. Context switch time is shorter than the kernel-level threads.
5. It does not require modifications of the operating system.
6. User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.
7. It is simple to create, switch, and synchronize threads without the intervention of the process.

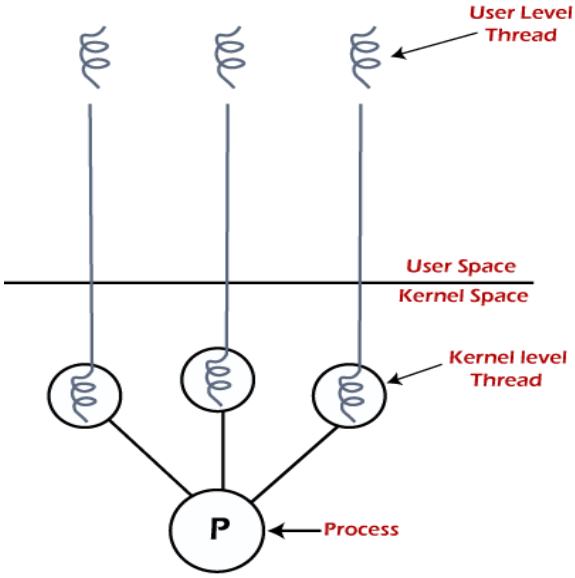
Disadvantages of User-level threads

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.



Kernel level thread

The kernel thread recognizes the operating system. There is a thread control block and process control block in the system for each thread and process in the kernel-level thread. The kernel-level thread is implemented by the operating system. The kernel knows about all the threads and manages them. The kernel-level thread offers a system call to create and manage the threads from user-space. The implementation of kernel threads is more difficult than the user thread. Context switch time is longer in the kernel thread. If a kernel thread performs a blocking operation, the Banky thread execution can continue. Example: Window Solaris.



Advantages of Kernel-level threads

1. The kernel-level thread is fully aware of all threads.
2. The scheduler may decide to spend more CPU time in the process of threads being large numerical.
3. The kernel-level thread is good for those applications that block the frequency.

Disadvantages of Kernel-level threads

1. The kernel thread manages and schedules all threads.
2. The implementation of kernel threads is difficult than the user thread.
3. The kernel-level thread is slower than user-level threads.

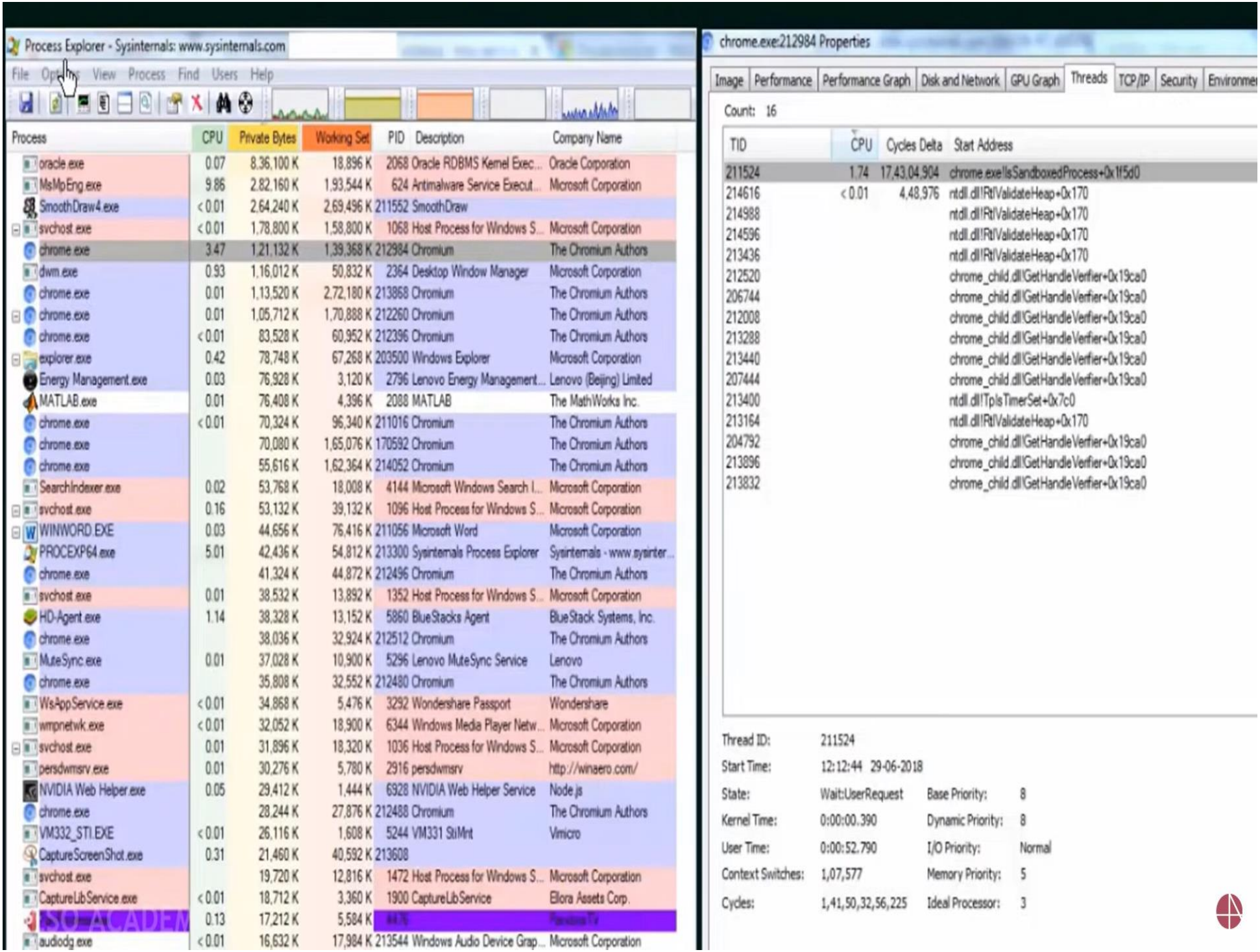
Components of Threads

Any thread has the following components.

1. Program counter
2. Register set
3. Stack space

Benefits of Threads

- **Enhanced throughput of the system:** When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.
- **Effective Utilization of Multiprocessor system:** When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
- **Faster context switch:** The context switching period between threads is less than the process context switching. The process context switch means more overhead for the CPU.
- **Responsiveness:** When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
- **Communication:** Multiple-thread communication is simple because the threads share the same address space, while in process, we adopt just a few exclusive communication strategies for communication between two processes.
- **Resource sharing:** Resources can be shared between all threads within a process, such as code, data, and files. Note: The stack and register cannot be shared between threads. There is a stack and register for each thread.


Process Explorer - Sysinternals: www.sysinternals.com
File Options View Process Find Users Help
Process CPU Private Bytes Working Set PID Description Company Name
oracle.exe 0.07 8,36,100 K 18,896 K 2068 Oracle RDBMS Kernel Exec... Oracle Corporation
MsMpEng.exe 9.86 2,82,160 K 1,93,544 K 624 Antimalware Service Execut... Microsoft Corporation
SmoothDraw4.exe < 0.01 2,64,240 K 2,69,496 K 211552 SmoothDraw
svchost.exe < 0.01 1,78,800 K 1,58,800 K 1068 Host Process for Windows S... Microsoft Corporation
chrome.exe 3.47 1,21,132 K 1,39,368 K 212984 Chromium The Chromium Authors
dwm.exe 0.93 1,16,012 K 50,832 K 2364 Desktop Window Manager Microsoft Corporation
chrome.exe 0.01 1,13,520 K 2,72,180 K 213868 Chromium The Chromium Authors
chrome.exe 0.01 1,05,712 K 1,70,888 K 212260 Chromium The Chromium Authors
chrome.exe < 0.01 83,528 K 60,952 K 212396 Chromium The Chromium Authors
explorer.exe 0.42 78,748 K 67,268 K 203500 Windows Explorer Microsoft Corporation
Energy Management.exe 0.03 76,928 K 3,120 K 2796 Lenovo Energy Management... Lenovo (Beijing) Limited
MATLAB.exe 0.01 76,408 K 4,396 K 2088 MATLAB The MathWorks Inc.
chrome.exe < 0.01 70,324 K 96,340 K 211016 Chromium The Chromium Authors
chrome.exe 70,080 K 1,65,076 K 170592 Chromium The Chromium Authors
chrome.exe 55,616 K 1,62,364 K 214052 Chromium The Chromium Authors
SearchIndexer.exe 0.02 53,768 K 18,008 K 4144 Microsoft Windows Search I... Microsoft Corporation
svchost.exe 0.16 53,132 K 39,132 K 1096 Host Process for Windows S... Microsoft Corporation
WINWORD.EXE 0.03 44,656 K 76,416 K 211056 Microsoft Word Microsoft Corporation
PROCEXP64.exe 5.01 42,436 K 54,812 K 213300 Sysinternals Process Explorer Sysinternals - www.sysinter...
chrome.exe 41,324 K 44,872 K 212496 Chromium The Chromium Authors
svchost.exe 0.01 38,532 K 13,892 K 1352 Host Process for Windows S... Microsoft Corporation
HD-Agent.exe 1.14 38,328 K 13,152 K 5860 BlueStacks Agent BlueStack Systems, Inc.
chrome.exe 38,036 K 32,924 K 212512 Chromium The Chromium Authors
MuteSync.exe 0.01 37,028 K 10,900 K 5296 Lenovo MuteSync Service Lenovo
chrome.exe 35,808 K 32,552 K 212480 Chromium The Chromium Authors
WsAppService.exe < 0.01 34,868 K 5,476 K 3292 Wondershare Passport Wondershare
wmpnetwk.exe < 0.01 32,052 K 18,900 K 6344 Windows Media Player Netw... Microsoft Corporation
svchost.exe 0.01 31,896 K 18,320 K 1036 Host Process for Windows S... Microsoft Corporation
persdwmsrv.exe 0.01 30,276 K 5,780 K 2916 persdwmsrv http://winaero.com/
NVIDIA Web Helper.exe 0.05 29,412 K 1,444 K 6928 NVIDIA Web Helper Service Node.js
chrome.exe 28,244 K 27,876 K 212488 Chromium The Chromium Authors
VM332_STI.EXE < 0.01 26,116 K 1,608 K 5244 VM331 StiMnt Vmicro
CaptureScreenShot.exe 0.31 21,460 K 40,592 K 213608
svchost.exe 19,720 K 12,816 K 1472 Host Process for Windows S... Microsoft Corporation
CaptureLibService.exe < 0.01 18,712 K 3,360 K 1900 CaptureLibService Ellora Assets Corp.
audiodg.exe < 0.01 16,632 K 17,984 K 213544 Windows Audio Device Grap... Microsoft Corporation
chrome.exe:212984 Properties
Image Performance Performance Graph Disk and Network GPU Graph Threads TCP/IP Security Environment
Count: 16
TID CPU Cycles Delta Start Address
211524 1.74 17,43,04,904 chrome.exe!IsSandboxedProcess+0x1f5d0
214616 < 0.01 4,48,976 ntdll.dll!RtlValidateHeap+0x170
214988 ntdll.dll!RtlValidateHeap+0x170
214596 ntdll.dll!RtlValidateHeap+0x170
213436 ntdll.dll!RtlValidateHeap+0x170
212520 chrome_child.dll!GetHandleVerifier+0x19ca0
206744 chrome_child.dll!GetHandleVerifier+0x19ca0
212008 chrome_child.dll!GetHandleVerifier+0x19ca0
213288 chrome_child.dll!GetHandleVerifier+0x19ca0
213440 chrome_child.dll!GetHandleVerifier+0x19ca0
207444 chrome_child.dll!GetHandleVerifier+0x19ca0
213400 ntdll.dll!TpIsTimerSet+0x7c0
213164 ntdll.dll!RtlValidateHeap+0x170
204792 chrome_child.dll!GetHandleVerifier+0x19ca0
213896 chrome_child.dll!GetHandleVerifier+0x19ca0
213832 chrome_child.dll!GetHandleVerifier+0x19ca0
Thread ID: 211524
Start Time: 12:12:44 29-06-2018
State: Wait:UserRequest
Base Priority: 8
Kernel Time: 0:00:00.390
Dynamic Priority: 8
User Time: 0:00:52.790
I/O Priority: Normal
Context Switches: 1,07,577
Memory Priority: 5
Cycles: 1,41,50,32,56,225
Ideal Processor: 3