Multi-threaded Process Scheduling and Context Switch Metrics

Modern operating systems handle multiple processes and threads by interleaving their execution on available CPUs. This is done through **scheduling** and results in **context switching**, especially in multi-threaded applications. Here's a deep dive into how multi-threaded scheduling works and how to observe and measure context switches in Linux.

🔄 Context Switching: What Happens?

Context Switch is the process where the CPU switches from one task (process/thread) to another. This involves:

- Saving the state (registers, stack pointer, program counter) of the currently running thread.
- Loading the state of the next thread to be scheduled.

Types of context switches:

- **Voluntary:** When a thread sleeps or yields.
- **Involuntary:** When the scheduler preempts a thread due to time slice expiration or higher-priority thread.

Context switching is **CPU-intensive** and too many context switches can lead to performance degradation (thrashing).

🧵 Multi-threaded Programming Impact

- Threads that block frequently (I/O-bound) may result in many **voluntary context switches**.
- CPU-bound threads may face **involuntary context switches** due to time-sharing.

🔍 High thread count apps (like Nginx, JVM apps) must balance between concurrency and switch overhead.

📊 Example

```
// multithreaded.c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

void* busy_loop(void* arg) {
    while (1);
}

int main() {
    pthread_t tid[4];
    for (int i = 0; i < 4; ++i)
        pthread_create(&tid[i], NULL, busy_loop, NULL);
    sleep(30);
    return 0;
}
```

Run with:

```
gcc multithreaded.c -o mt -lpthread
./mt &
pidstat -w -p $(pidof mt) 1
```

You'll observe high context switch counts due to constant thread competition.

✅ Summary

Metric	**Tool**
Vol/Invol switches	`pidstat`, `perf`

Metric	Tool
Scheduler behavior	`schedtool`, `chrt`
Global switch rate	`vmstat`
Thread view	`htop`, `top -H`

Understanding thread scheduling and context switching is crucial for building **scalable**, **real-time**, or **low-latency** applications in Linux environments.