Anomaly of CPU cycles

Aniket Basak (CrS2401)
Cryptographic and Security Implementation

August 2025

Why the Maximum CPU Cycle Value for $N = p \times q$ Appears as an Outlier

In Table 4.3 the *maximum* CPU cycle value for $N = p \times q$ shows a very large deviation, while the minimum and average remain close. This behavior is not caused by the multiplication itself being occasionally more expensive, but rather by external factors in the measurement environment. The likely reasons are:

1. **OS scheduling, interrupts, or virtualization overhead** Since the experiments were run inside a VirtualBox VM on an Intel i5, the host OS or hypervisor can preempt the VM. A single descheduling or interrupt adds thousands–millions of cycles to that measurement, producing a large maximum.

2. **Memory allocation and page faults** GMP's `mpz_t` uses heap allocation for its limbs. If during the loop a new allocation or reallocation occurs (e.g., when N grows beyond current limb capacity), a costly `malloc` or page fault may occur, leading to a one-off large timing.

3. **Timing measurement artifacts** The use of `__rdtsc()` without serialization can produce outliers. If the thread migrates between cores, or the CPU executes instructions out of order, the cycle count may include unrelated delays. This effect is stronger in virtualized environments where the TSC is virtualized.

4. **CPU frequency scaling and heterogeneous cores** On hybrid processors with Performance (P) and Efficiency (E) cores, or with dynamic frequency scaling, a change of core or P-state during measurement can inflate the observed cycle count.

5. **Rare pathological input** (unlikely in this case) For $N = p \times q$, multiplication cost is deterministic for fixed operand sizes. No special prime pattern can make a single multiplication disproportionately expensive, so this explanation is less likely than system-level effects.

How to Determine the Cause

To confirm the exact source of the outliers, one can:

- Pin the process to a single CPU core using `taskset -c`, and disable dynamic frequency scaling (set CPU governor to `performance`).
- Use serializing instructions (`CPUID` or `RDTSCP`) around `__rdtsc()` to ensure accurate cycle counts.
- Preallocate GMP integers with `mpz_init2` so that limb reallocation and heap growth do not occur inside the benchmark loop.

- Record additional OS-level statistics with `getrusage()` or `perf` to check for context switches, page faults, or interrupts during outlier iterations.
- Report median, 90th percentile, 99th percentile, and maximum, rather than only min/avg/max, to clearly separate rare environmental spikes from steady-state algorithmic cost.

Quick Fixes

1. Use `mpz_init2()` to preallocate memory for p, q, and N.
2. Serialize timing using `RDTSCP` and `CPUID`.
3. Run the benchmark outside a VM if possible, or with isolated pinned cores inside the VM.
4. Present results using trimmed means or percentiles to avoid the misleading effect of rare outliers.

Conclusion

The large maximum cycle count reported in Table 4.3 for $N = p \times q$ is most likely caused by external system effects such as context switches, interrupts, page faults, dynamic memory allocation, or unsynchronized timing measurements in a virtualized environment. It is not due to any special number pattern or inherent algorithmic behavior of big integer multiplication.

References

1. Intel/AMD Processor Manuals
2. Linux Kernel Documentation
3. "Detecting CPU Anomalies Using Hardware Performance Counters" (IEEE Transactions)
4. Introduction to Modern Cryptography by Lindell and Katz
5. ChatGPT and DeepSeek.