Gurudatta R K

+91-8424016635 | gurudatta.kondampallikar@gmail.com | github:GurudattaRK | LinkedIn

Low-Level languages & Low-Latency systems programmer with hands-on experience in x86 Assembly, C, C++, & latency-critical system design. Currently targeting Low Latency Programming roles.

Technical Skills

Languages: C, C++, x86-64 Assembly, CUDA C

Low-Latency & Performance: SIMD/AVX Intrinsics, Branchless Programming, Loop Unrolling, Memory Alignment, Cache-Friendly Data Layouts, LTO/PGO, Branch Elimination, Pipeline Optimization, IO_Uring, Async I/O, Lock-free Data Structures, aggressive build/compile time optimizations.

x86 CPU specialization: PMU Hardware Counters, Instruction-Level Parallelism (ILP), Out-of-Order Execution, Cache Hierarchy (L1/L2/L3), TLB Internals, RDTSC/RDTSCP

Heterogeneous Parallel Computing: Multi-threading, Multi-core computing, GPU Parallelism (CUDA)

Profiling & Debugging Tools: PicoPerf, Linux Perf API, GDB, Valgrind/Callgrind

Projects

•PicoPerf – Zero-Overhead PMU Benchmarking Library *GitHub*

- Captures hardware-native CPU metrics directly from x86-64 CPU hardware registers like retired instructions, elapsed cycles, **IPC**, branch misses, **L1D cache misses**, & **L3 cache misses**. Minimized sampling or interpolation errors, & no loss of resolution at the nanosecond & single-cycle level.
- Uses **RDTSC/RDTSCP** x86 instructions with **lfence** serialization for nanosecond-precision cycle & CPU tick measurement. Timed not just how long a code path took but exactly how hardware behaved. Reveals cache pressure, branch mispredictions, & IPC degradation as reliable measurable hardware performance metrics.
- Perf group (leader + 5 members) starts & stops all 6 atomically via a single group-enable **ioctl** on the leader, guaranteeing every counter covers the exact same instruction window with minimal inter-counter skew. Auto-setup pins the thread, locks pages in RAM, & requests SCHED_FIFO for real-time priority. Reports **p50/p99/p99.9** percentiles & measures its own overhead (∼10–50ns).

•Low-Latency-Lab – Performance tuning & Low Latency Engineering Lab *GitHub*

- This repository is my lab/playground where I experiment, learn, break stuff & re-engineer them for low latency & maximum performance. Each project is profiled, optimized, & benchmarked with **p90/p99 per-chunk latency** & **mean throughput over 10+ runs** to catch tail latencies, cold starts & thermal variance. Analyzing software & hardware behavior then systematically eliminating bottlenecks at the instruction & cache-line level until the hardware is the bottleneck, not the code.
- **Bit-Level Encryption v1.0** — single-threaded reference cipher & the crypto core for TeraCrypt (below). Key optimizations: **AVX2 SIMD intrinsics** for fully in-register Eklundh's 5-stage transpose (zero register-memory spills), eliminating runtime overhead by precomputing constant data, reducing instructions in hot path, introducing instruction-level parallelism, aligned loads/stores, cache-line aligned matrices, compiler-enforced function inlining, loop unrolling, LTO, & PGO. Mean throughput improved by ∼**11900%** (120x) from ∼**3.33 MB/s** to ∼**400 MB/s (0.4 GB/s)**.
- **TeraCrypt v4.0** — Multi-Core async version of above-mentioned Bit-Level Encryption v1.0. Parallel computing part re-built to replace a naive Python implementation. Pipeline & parallel computing optimizations: low latency minimal-syscall file accessing via **IO_Uring** async I/O with batched submissions, **lock-free MPMC queues** for cross-process chunk dispatch & async processing, further minor optimizations grouped together like DMA access, CPU pinning, TLB & cache miss reducing optimizations, etc. Achieved ∼**1.44 GB/s** mean throughput on a 6-core Zen 2 CPU with Gen-4 NVMe SSD machine, a ∼**260%** improvement (3.6x) over the single-threaded variant.

Experience

•Software Developer at QUALYS *June 2025 – Present*

- Identified & patched **multiple CVEs** in the macOS agent & its deployment scripts; hardened agent pipelines against injection attacks via input sanitization & systematic code hardening.
- Led the introduction of **Software Composition Analysis (SCA)** to macOS. Implemented dependency scanning feature for macOS. Upgraded **Patch Management** module & also improved its reliability by decoupling it from macOS agent.

•Security Software Intern at SentinelOne *June 2024 – January 2025*

- Performed **reverse engineering** of newly discovered macOS malware samples & authored **4 new detection rules** covering previously undetected threat families, directly expanding the production detection engine's coverage.
- Optimized existing detection rule logic by refining filter conditions & improving detection accuracy; achieved up to **86% reduction in false positives** for legitimate macOS applications in the highest-frequency detection paths.

Awards & Achievements

View All Certificates

- **Best Final Year Project 2024** – Awarded by PES University for post-quantum cryptography research project.
- **Hackathon Events** – Winner of a Mathematics-based hackathon; Top 5 in 3 CTF hackathon events.

Education

- **Bachelor of Technology in Computer Science Engineering** *November 2020 – December 2024*
  PES University, Bengaluru