Vikram Oddiraju

vikramoddiraju@gmail.com | (765)-667-3442 | linkedin.com/in/vikram-oddiraju | github.com/Vekram1 | vekram1.github.io | x.com/vkrm_odrju

EDUCATION

Purdue University West Lafayette, IN | **August 2021 - May 2025**
BS in Computer Science
Minors: Mathematics & Economics

Relevant Coursework:
Operating Systems, Computer Networks, Systems Programming (Linux Kernel), Numerical Methods, Linear Algebra, Probability

Languages/Tools/Platforms:
C, C++, x86 Assembly, Python, Bash, Spring Boot, GDB, Git, Docker, CMake, Google Test

Technical Skills:
Kernel Programming/ABI, Software Development, Data Analysis, Stochastic Calculus, Floating Point Arithmetic

RELEVANT WORK EXPERIENCE

Volante Technologies | Software Development Intern June 2022 – August 2022
- Collaborated with a **Java** software team to build a payments SaaS product, QuickConnect
- Worked on **tokenization** aspect of the product for secure monetary transactions between banks and counterparties
- Developed RESTful API endpoints with the help of **Spring Boot** for a web app that would securely generate and store tokens in a relational **SQL database**
- Gained experience in testing web APIs under load in **Postman**

RELEVANT PROJECTS

TCP Congestion Control Analysis in Virtualized LAN (Python[Scapy], FFmpeg, Docker) April 2025
- Built a virtual LAN using Docker containers (client/server) to benchmark **BBR** vs. **Cubic** TCP congestion control across 4 emulated conditions (**delay** 500ms, **loss** 10%, **rate** 90 Mbit/s, delay 500 ms + rate 90 Mbit/s)
- Measured **FFmpeg** live streaming throughput, bitrate, FPS, and retransmissions with Python scripting to parse .pcap and tcpdump log files
- Result: BBR achieved higher throughput under better network conditions, while Cubic remained more resilient to loss and latency

Multithreading Python 3.14 Matrix Multiplication Improvement (Disable GIL) October 2025
- Created a matrix multiplication program using **ThreadPoolExecuter** with **disable-gil flag**
- Did comparative testing against single thread and multiprocess multiplication (outperforms multiprocessing less)
- Utilized **chunked row multiplication** to maintain parallelism and reduced overhead compared to multiprocessing
- Upcoming: Improving accuracy with **Knuth dot products** and finding optimal chunk sizes

Ornstein-Uhlenbeeck (OU) SDE Mixed Precision MLMC Simulation (C++, Python) October 2025
- Developed a C++ **multi-level Monte Carlo (MLMC)** engine for calculating the expectation of the OU **stochastic differential equation** by analyzing rounding error propagation that occurs through **Euler-Marayuma** estimation
- Applied a mixed precision MLMC strategy to derive an optimal number of runs per level to achieve the most accurate and least computationally intense way to obtain the expectation of the SDE

Reinforcement Learning based System of Equations Solver (Python) September 2025
- Built a **PPO reinforcement learning agent** to solve Ax=b using an iterative solver, FGMRES, with adaptive sized diagonal block preconditioning
- **Achieved** up to **4x less iterations** computed than fixed-block FGMRES, leveraging Stable Baseline 3's library for environment and policy optimization
- Writing: vekram1.github.io/portfolio_optimization_using_rl_based_iterative_solver/

Asynchronous Event Scheduling in XINU OS using Assembly Callback April 2025
- Programmed a real-time alarm and **callback system** within the **XINU kernel** to trigger events **asynchronously** at a fixed time interval using **C** and **assembly**
- Wrote an **x86 assembly** callback function to minimize **interrupt handling** overhead and verified timing by pushing timestamps onto the runtime stack for later analysis

CUDA Floating Point 16 Matrix Multiplication Accumulation October 2025
- Forced FP16 accumulation in FP16 matrix multiplication by writing custom CUDA kernel to improve runtime