Jacob Malloy

Knoxville,TN

☎ (615)454-1182 ✉ jmalloy1@vols.utk.edu jacob-malloy-b5b5ba190 JacobMalloy Jmalloy1

PROFESSIONAL SUMMARY

Ph.D. candidate in Computer Science at the University of Tennessee with a strong foundation in systems, machine learning, and security. Experienced researcher in optimizing memory management and non-uniform memory architectures within OpenJDK's HotSpot JVM. Passionate about low-level systems, compilers, and performance analysis, with hands-on experience in kernel modifications and high-performance computing. Dedicated educator with teaching experience in compilers, algorithms, and software systems, providing mentorship and technical instruction. Skilled in C, C++, Rust, Python, and a range of systems and security technologies.

EDUCATION

The University of Tennessee — **08 2023 – Present**

*Ph.D. in Computer Science - **3.94/4.00*** — *Knoxville, Tennessee*

The University of Tennessee — **08 2020 – 05 2023**

Bachelor of Science in Computer Science — *Knoxville, Tennessee*

RESEARCH EXPERIENCE

CORSys Research Lab — **06 2021 – Present**

Research Assistant - Prof. Michael Jantz — *Knoxville, Tennessee*

- Work on optimizing the memory usage by analyzing different Garbage Collectors usage patterns.
- Research non uniform memory architectures with the ZGC garbage collector in OpenJDK's HotSpot JVM.
- Port internal kernel change modifying the cgroup system from kernel version 5.7.2 to 5.14.
- Maintain servers to keep a secure working environment so that there are no stops in research.

TEACHING EXPERIENCE

University of Tennessee EECS — **08 2023 – Present**

Teaching Assistant — *Knoxville, Tennessee*

- Assisted in teaching Compilers, Advanced Algorithms, and Software Systems for over **400** students.
- Graded programming assignments and projects, providing detailed feedback to help students improve their code quality and problem-solving skills.
- Led lab sessions, delivering instructional material, demonstrating key programming techniques, and guiding students through hands-on exercises.
- Held office hours and responded to student inquiries, clarifying course material and debugging issues to enhance learning outcomes.

COURSEWORK

Systems

- **Introductory Systems** Covered memory management, process scheduling, file systems, and concurrency.
- **Hardware Architecture** Studied introductory assembly, combinatorial logic, virtual memory, and IEE-754 floating point.
- **Graduate Hardware Architecture** Explored the memory hierarchy, cache architecture, branch prediction, and Tomasulo's algorithm.
- **Compilers** Implemented a compiler frontend using **LLVM**, LEX and YACC and implement **copy propagation** LLVM compiler pass.
- **Paper Reading** Explored topics on the cutting edge of operatings systems, language virtual machines, and compilers.

Machine Learning

- **Machine Learning** Studied regression, classification, clustering, and dimensionality reduction, with a focus on theoretical foundations and implementation.

Security

- **Introduction to Cybersecurity** Covered fundamental security principles, attack vectors, and common vulnerabilities.
- **Network Security** Learned about security implications at each level of the network stack, as well as in depth learning of the TLS stack by implementing a custom **VPN** using **TLS** over the TUN interface.
- **Applied Cryptography** Focused on encryption algorithms, key exchange mechanisms, and the block chain.
- **Human Factors in Cybersecurity** Investigated usability, phishing detection, and security awareness training.

Other Graduate Coursework

- **Advanced Algorithms** Learned about advanced algorithms such as A*, Bloom Filters, and B-Trees. Presented on the Paxos Algorithm.
- **Algorithm Analysis** Studied algorithmic complexity of different asymptotics and algorithms such as the Simplex Algorithm.
- **Foundations of Computer Science** Covered formal languages, automata theory, computability, and tractability.
- **Graph Theory** Covered advanced topics in graph theory such as coloring, graph topology, and the Robertson-Seymour Graph Minor Theorem.
- **Graphics** Implemented graphics rendering techniques using webgl and GLSL shaders.