108 Naramore Ln.
Los Gatos, CA-94032
(+1) 614-477-9964
sengupta.25@buckeyemail.osu.edu
, aritras@amazon.com
US permanent resident

Aritra Sengupta

Curriculum Vitae

Interests

Program analysis (static and dynamic), runtime systems, programming models, parallel and distributed systems/systems software, software and hardware transactional memory, compilers, synchronization schemes, code optimization.

I enjoy desigining and implementing analyses and runtimes across the software stack considering tradeoffs between performance and correctness.

Education

9/2011–5/2017 **Doctor of Philosophy (Ph.D.)**, *Department of Computer Science and Engg.*, Ohio State University, Columbus, Ohio.
Major: Computer Science and Engg.
Advisor: Prof. Michael D. Bond.
GPA: 3.95/4

6/2004–6/2008 **Bachelor of Technology (B.Tech)**, *School of Computer Science and Engineering, Vellore Institute of Technology University*, Vellore, India.
Major: Computer Science and Engg.
GPA: 9.29/10

Selected Publications

Michael Emmi, Liana Hadarean, Ranjit Jhala, Lee Pike, Nicolás Rosner, Martin Schäf, **Sengupta, Aritra**, and Willem Visser. RAPID: Checking API Usage for the Cloud in the Cloud. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (**ESEC/FSE**)*, 2021.

Tanakorn Leesatapornwongsa, **Aritra Sengupta**, Masoud Saeida Ardekani, Gustavo Petri, and Cesar A. Stuardo. Transactuations—Where Transactions Meet the Physical World (extended version of USENIX ATC 2019 paper with formalized semantics). *ACM Trans. Comput. Syst. (**TOCS**)*, 36(4), May 2020.

Aritra Sengupta, Tanakorn Leesatapornwongsa, Masoud Saeida Ardekani, and Cesar Stuardo. Transactuation—Where Transaction Meets the Physical World. *Appeared at Usenix Annual Technical Conference (**USENIX ATC**, Awarded Best Paper)*, July 2019.

Man Cao, Minjia Zhang, **Aritra Sengupta**, Swarnendu Biswas, and Michael D. Bond. Hybridizing and Relaxing Dependence Tracking for Efficient Parallel Runtime Support. *ACM Trans. Parallel Comput. (**TOPC**)*, 4(2), August 2017.

Aritra Sengupta, Man Cao, Michael D. Bond, and Milind Kulkarni. Legato: End-to-End Bounded Region Serializability Using Commodity Hardware Transactional Memory. In *ACM SIGPLAN International Symposium on Code Generation and Optimization (**CGO**)*, Feb 2017.

Man Cao, Jake Roemer, **Aritra Sengupta**, and Michael D. Bond. Prescient Memory: Exposing Weak Memory Model Behavior by Looking into the Future. In *ACM SIGPLAN International Symposium on Memory Management (**ISMM**)*, June 2016.

Man Cao, Minjia Zhang, **Aritra Sengupta**, and Michael D. Bond. Drinking from Both Glasses: Combining Pessimistic and Optimistic Tracking of Cross-Thread Dependences. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (**PPoPP**)*, March 2016.

Aritra Sengupta, Man Cao, Michael D. Bond and Milind Kulkarni. Toward Efficient Strong Memory Model Support for the Java Platform via Hybrid Synchronization. In *ACM International Conference on Principles and Practices of Programming on the Java Platform (**PPPJ'15**)*, September 2015.

Aritra Sengupta, Swarnendu Biswas, Minjia Zhang, Michael D. Bond and Milind Kulkarni. EnfoRSer: Hybrid Static–Dynamic Analysis for Region Serializability. In *ACM Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, March 2015.

Swarnendu Biswas, Jipeng Huang, **Aritra Sengupta**, and Michael D. Bond. DoubleChecker: Effcient Sound and Precise Atomicity Checking. In *ACM Conference on Programming Language Design and Implementation (**PLDI**)*, June 2014.

Michael D. Bond, Milind Kulkarni, Man Cao, Minjia Zhang, Meisam Fathi Salmi, Swarnendu Biswas, **Aritra Sengupta**, and Jipeng Huang. Octet: Capturing and Controlling Cross-Thread Dependences Efficiently. In *ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (**OOPSLA**)*, October 2013.

Poster Presentations

- Presented work on *region serializability enforcement using commodity hardware transactional memory* at *ACM Student Research Competition, OOPSLA 2015.*
- Presented work on *sequential consistency enforcement in software* at *ACM Student Research Competition, PLDI 2013.*

Experience

Senior Applied Scientist, *AWS AI*, Santa Clara, California.

04/2022–Present

- Designed and developed a modular and scalable version of a program analysis tool with strong abstractions for soundness.
- Developed scalability techniques to reduce scope of program analysis while retaining soundness and precision. Leveraged the analysis to launch a high-fidelity analysis as part of an external facing AWS service.

8/2019–04/2022 **Applied Scientist**, *AWS, Automated Reasoning Group then AWS AI*, Cupertino/Santa Clara, California.

- Checking/Enforcing correctness properties at scale on AWS services.
- Designed, built, and productionalized a Java program analysis tool to check properties on code at scale. The tool is currently deployed on analysis that scans every code review at Amazon and as part of multiple external facing AWS services.
- Designed, built, and productionalized a capability to provide evidence of properties checked that didn't result in detected vulnerabilities. The tool collected meaningful sequence of transitions that led the detector instance to terminate in a "good" state.

7/2017–Present **Senior Research Engineer**, *Samsung Research America*, Mountain View, California.

- Built a reliable IoT application execution system that provides strong guarantees, e.g. keeping soft states/app states consistent with hard states/device states. A simple programming abstraction allows the IoT developer to express consistency requirements which is respected by our runtime. The runtime is built atop Azure serverless functions executing over Cosmos DB.
- Proposed a transactional framework to provide low-latency and serializable execution of smart-home applications. Development of a static analysis framework to understand the interaction of smart-home applications operating on shared software states and physical actuators. Transactions in the proposed framework read speculative soft states from the hub instead of physical sensors and actuators to derive a consistent order before issuing actual physical actuations on the devices.

9/2011–5/2017 **Graduate Research Associate**, *Computer Science and Engineering Department, Ohio State University*, Columbus, Ohio.

Mentor: Michael D. Bond

- Designed and implemented large concurrent systems considering tradeoffs between performance and correctness across the software stack.

- Built analysis on top of a default runtime and extracted performance considering several avenues of optimization in design and implementation [ASPLOS 2015, PPPJ 2015, CGO 2017, PPOPP 2016, OOPSLA 2013].
- Designed and implemented low-overhead techniques using hybrid static—dynamic analysis to enforce bounded region serializability. The techniques require compiler transformations, code generation, implementation of log-based and register-based transactional memory, efficient compiler analyses, etc.[ASPLOS 2015, PPPJ 2015, CGO 2017].
- Developed static analysis to eliminate redundant instrumentation, demarcate regions of code, generate dependence graphs for efficient code generation during JIT compilation [ASPLOS 2015, OOPSLA 2013].
- Designed and implemented dynamic analysis to detect shared memory bugs including sequential consistency violations and atomicty violations [ISMM 2016, ASPLOS 2015].
- Designed and Implemented hybrid synchronization techniques (per-access lock, per-region lock) to enable low-overhead enforcement of a strong memory model. Implementation of pessimistic static locks and optimistic dynamic locks [PPPJ 2015].
- Leveraged hardware transactional memory (HTM), conditional code generation, and control-theory algorithms to enforce a strong, always-on and practical memory consistency model [CGO 2017].
- Combined optimistic and pessimistic synchronization to develop a region–serializability enforcer [PPoPP 2016].
- Developed efficient and practical instrumentation schemes, implementing and modifying several parts of a JVM including baseline compiler, optimizing compiler, Java locking protocols, garbage collection features etc. [ASPLOS 2015, PPPJ 2015, ISMM 2016, PPOPP 2016, OOPSLA 2013, PLDI 2014, CGO 2017].

5/2016–8/2016 **Summer Research Intern**, *Microsoft Research*, Redmond, Seattle, USA.
Mentors: Kathryn S. McKinley, Ricardo Bianchini, Sameh Elnikety, Yuxiong He.

- Research on designing dynamic and adaptive scheduling techniques for latency sensitive server systems. Design and implementation of an online technique to detect memory interference amongst co-scheduled requests, using hardware performance counters, to automatically manage the server's scheduling, concurrency and intra-request paralellism in order to increase server utilization and reduce tail-latency.

8/2008–7/2011 **Systems Engineer**, *Tata Consultancy Services (TCS)*, India.
Migration of code in legacy systems to C/C++, Interfacing C libraries with COBOL. Developments in core Java.

Service

- Program committee member for ASPLOS 2024
- Program committee member for ASPLOS 2023
- Program committee member for OOPSLA 2021.
- Session chair, OOPSLA 2020.
- Program committee member for OOPSLA 2020.
- Reviewer CONCUR 2020.
- Reviewer for journal: IEEE Transactions on Software Engineering(TSE).
- Program committee member for IPDPS 2019.
- External review committee (ERC) member for ASPLOS 2019.
- Reviewer for PACT 2018.
- Reviewer for journal: ACM Transactions on Architecture and Code Optimization(TACO).
- Reviewer for journal: IEEE Transactions on Parallel and Distributed Systems (TPDS).
- Reviewer for journal: Concurrency and Computation: Practice and Experience.
- Member of EuroSys 2018 shadow program committee.
- Member of ASPLOS 2018 shadow program committee.
- Artifact evaluation committee member for PPoPP 2017/CGO 2017.
- Artifact evaluation committee member for PPoPP 2016/CGO 2016.
- Subreviewer for PLDI 2017.
- Subreviewer for PACT 2017.

Mentoring experience

- Mentored summer 2022 research intern at AWS CodeGuru—Siwei Cui, Ph.D. student at Texas A&M university.

• Mentored summer 2021 research intern at AWS CodeGuru—Ali Nowraiz Khan, Ph.D. student at U.C. Riverside.
• Mentored summer 2020 research intern at AWS CodeGuru—Ali Nowraiz Khan, Ph.D. student at U.C. Riverside.
• Mentored summer 2018 research intern at Samsung Research—Cesar S. Stuardo, Ph.D. student at U.Chicago.

Honors and Awards

7/2019 Awarded the Best Paper Award for Transactuation at USENIX ATC 2019.

2/2017 Awarded NSF travel grant for presenting at CGO 2017 in Austin, USA

3/2015 Awarded NSF travel grant for presenting at ASPLOS 2015 in Istanbul, Turkey.

2011-2012 Awarded the prestigious "University Fellowship", graduate student fellowsip, Ohio State University.

11/2008 Secured "Initial Learning Program Top Performer" award in the first phase of training at Tata Consultancy Services, India.

2006-2008 Awarded merit certificate and scholarship in three consecutive years 2006, 2007, 2008 for academic performance at Vellore Institute of Technology University, Vellore, India.

2004 Awarded merit certificate for academic performance in Physics, Chemistry and Mathematics in Indian School Certificate Examination.

Technical skills

Design and implemenetation: Design and implementation of large concurrent systems on single node and on distributed platforms. Strong expertise in transactional system design.

Program Analysis: Twelve years of experience in developing program analysis on compilers and IRs. Six years of experience in designing and developing efficient industry-scale program analysis techniques for Java.

Security Tools: Four years of experience in building security focused program analysis tools that scan millions of lines of code at industry scale.

Concurrent systems, compilers, JVM, and runtime: Nearly six years of experience as graduate student in designing and developing efficient runtime mechanisms for concurrency in a managed runtime (JVM- Jikes RVM) including but not limited to static and dynamic analysis, programming models, memory models, software and hardware transactional memory, code generation, compiler transformations, efficient synchronization schemes, performance engineering in the runtime etc.

Languages and scripting:

- Java, C/C++, Python
- Asynchronous programming using Node.js—Javascript.

Distributed Cloud Services:

- Serverless functions, e.g. Microsoft Azure functions.
- Document DB, e.g. Azure Cosmos DB.

Parallel programming and code analysis frameworks, libraries, runtime:

- OpenMP parallel programming model.
- ROSE compiler framework, Soot: a Java optimization framework.
- Hardware transactional memory: Intel TSX.
- Concurrency libraries: Intel TBB.

Benchmarks:

- DaCapo, Java Grande, STAMP.

Machine learning:

- Basic knowledge of logistic regression, neural networks, and deep learning.

Tools:

- Linux perf, Xperf, Intel PCM, PAPI, Chord and more

Patents

• **Aritra Sengupta**, Michael Emmi, Liana Hadarean, Martin Schaef, Nico Rosner, Lee Pike, Willem Visser, Peixuan Li. Method for checking finite state properties using modular and compositional static code analysis. P78737-US01

• Martin Schaef, Linghui Luo, Nico Rosner, **Aritra Sengupta**, Antonio Filieri, Thomas Cottenier, Lee Pike. On-demand input partitioning for SAST platforms. P78535-US01

• **Aritra Sengupta**, Martin Schaef, Lee Pike, Willem Visser, Nico Rosner. Method to find and visualize semantic relationships in static program analysis trace elements. P73038-US01

Selected Projects

• *Building Reliable Smart-home Applications with Transactuation:* Building Smart-home applications is hard with the inherent inconsisteny between operations on devices and application states due to temporary failures such as message loss. This work proivdes a programming abstraction to program Smart-home apps such that the system automatically detects inconsistencies between operations on devices and on app states. The systems corrects itself or notifies the user to mitigate the problem. The system is built with serverless functions running on a distributed key-value store.

• *Hybrid Static-Dynamic Analysis to Enforce Strong Program Semantics:* This work introduces a novel memory consistency model and demonstartes its ability to eliminate several memory consistency and concurrency bugs at low overheads. The runtime system enforces serializability of compiler demarcated regions of code using runtime perturbation and strong compiler transformations. The atomic regions ensure sequential consistency and tolerates several violations of atomicity leading to a strong memory model. The regions are statically and dynamically bounded to generate efficient code that enforces region serializability at run-time. **Link to code**

• *Hybrid Synchronization to Enforce a Strong Memory Model for Java:* Enforcing serializability of compiler demarcated regions provides sequential consistency and atomicity of bounded regions. However, to enforce such semantics in software, instrumentation with dynamic locks is required at every memory access leading to high instrumentation overhead. This work provides an alternative synchronization mechanism that uses a single static lock. A static lock is favorable for low-conflicting code regions and a dynamic lock is favorable for precise conflict detection. A run-time profiling technique and a cost-benefit model effectively selects the right synchronization for a region. This hybridized instrumentation scheme enforces strong semantics at low overhead. **Link to code**

• *Efficient Dynamically Bounded Region Serializability with Hardware Transactions:* Statically bounded region serializability (SBRS) is a strong memory model but incurs an overhead of 25-35% when implemented in software. This work uses commodity hardware transactional memory (HTM) to enforce SBRS. However, counter-intuitively we find that HTM incurs an overhead of 180%, several times more than software. The start-up and tear-down cost of HTM transactions is the root of the overhead. This work introduces a novel control-theoretic approach to amortize per-transaction cost by merging several regions into a single transaction; and demonstrates its efficiency in overhead over prior work.**Link to code**