Research Statement

Gowtham Kaki

Computing has come a long way since Von Neumann introduced his stored-program model of a computer in 1945. Von Neumann's architecture is fundamentally *compute-centric* – a Central Processing Unit (CPU) performs all the computation by periodically fetching data and programs from a passive main memory or a secondary storage. Implicit in this model is the assumption that it is possible to tread a computation's data in its entirety through a "central" processor to a compute a single value that is the canonical result of a computation. This view of computing stands in stark contrast to the reality of modern applications, which are fundamentally *data-centric* as opposed to compute-centric – they operate on large amounts of data organized into multiple data banks, e.g., *databases*, *shards*, and *replicas*, and perform computations that are essentially distributed in nature to compute results for which no canonical answer exists. Despite such fundamental differences in the nature of computing, applications today are nonetheless programmed using the same language abstractions and programming techniques that trace their origins to the Von Neumann model. The resultant *impedance mismatch* between the computation performed and the programming model employed has been a consistent source of negative impact on the security, reliability, and performance of data-centric applications. For instance, coercing database transactions, which are concurrent by their very nature, into Von Neumann model of computing, which emphasizes an orderly execution of instructions, has resulted in an impedance mismatch that effectively exposed several database-backed commercial applications to concurrency-related attacks [10]. One such attack has been particularly notorious for it has led to the theft of over half-a-million dollars, and subsequent closure of an otherwise well-performing Bitcoin exchange named Flexcoin [1]. More such examples abound in literature and in popular press.

My Research

As evident from above, there is a need for novel computational models, language abstractions and programming techniques that better suit the nature of modern data-centric applications. Developing such programming infrastructure is indeed one of the focal points of my research, in particular my most recent work on distributed programming models and replicated data structures [7, 8, 2]. The more pressing need however is for automated reasoning techniques, program analyses, and compilation tools that work with the current generation of data-centric applications, helping them meet the ever so strict goals of scalability and availability *without* compromising security and robustness. Developing such automated reasoning infrastructure is another focal point of my research, and a unifying theme behind most of my work [5, 6, 4, 3, 9]. **At a high level, my research falls at the intersection of programming languages, software engineering, databases, and distributed systems. In particular, I aim to solve the most pressing problems in the latter two domains by developing novel approaches that build on top of the conventional wisdom and technical know-how of the former two**. The relevance of my work to the theory and practice of contemporary data-centric computing has been recognized through a generous research fellowship from Google. In the near future, I plan to continue my work on data-centric computing along the directions I outline below.

Future Work

I see many opportunities for future research in data-centric computing, which I would like to pursue in collaboration with researchers in Systems, Databases, Cryptography, and Machine Learning. I sketch a few possibilities below.

Formal models of complex software systems In [5] and [6], I demonstrate the utility of employing simple formal models of (distributed) database systems in lieu of their complex implementations to reason about application safety. I see this approach being useful in the context of file systems, lock-free data structures, data processing libraries – basically any real-world software system that accrues code and optimizations from decades of engineering effort, and becomes too complex to be analyzed directly by the verification tools tasked with establishing the safety of their clients. Building high-fidelity formal models of such systems, however, requires considerable domain knowledge, which is often scattered through API and informal documentation, unit tests, configuration files, logs, and commit messages. Mining structured knowledge from these sources, and (mostly) automating the task of synthesizing formal models that act as proxies for these systems during verification is a challenging-yet-worthwhile exercise, and one that I plan to take up in near future.

Programming models for fully-decentralized and privacy-conscious computing Users of internet services are becoming increasingly vocal about their (justifiable) privacy concerns on the internet. Ambitious initiatives have sought to address this issue by promoting full decentralization in application and network protocol design (e.g., Blockchains and IPFS), and by putting people in charge of their data in their own premises (e.g., TRVE Data). Well-intended as they may be, such initiatives nonetheless have systemic issues for there are currently no languages and tools that can bridge the wide chasm between the limited human cognition of the programmer, and planet-scale workings of the underlying system. In [8], I show that a distributed programming model inspired by version control workflow can significantly reduce the cognitive burden involved in building highly decentralized applications. However, the questions about performance (at internet-scale) and privacy are yet unanswered, and addressing them may require a significant re-think of how computation should be performed over the internet. I would like to collaborate with Systems, Database, and Cryptography researchers towards answering this question.

Distributed vector representations of data-centric computations Following the runaway success of "word2vec" in natural language processing, there have been attempts to build distributed vector representations of code snippets ("code2vec") with the aim of comparing and predicting the semantic properties of code. While the approach is promising, the success so far has been limited owing to the high semantic complexity of general-purpose computations in a Turing-complete language compared to words in a natural language. I believe more success can be had by restricting our attention to data-centric computations of which most are expressible in a Turing-*incomplete* language such as Datalog (A case in the point is SQL, which is a subset of Datalog. Another example is the TPC-C benchmark we used in evaluating ACIDifier [5], which was completely written in a Turing-incomplete OCaml DSL). I am therefore interested in evaluating the possibility of building distributed vector representations of data-centric computations expressed in (a variant of) Datalog – "Datalog2Vec", in collaboration with researchers in Databases, Machine Learning, and NLP.

References

[1] BTC Stolen from Poloniex, 2014. URL `https://bitcointalk.org/index.php?topic=499580`.

[2] Gowtham Kaki. *Automatic Reasoning Techniques for Non-Serializable Data-Intensive Applications*. PhD thesis, 8 2019. URL `https://hammer.figshare.com/articles/Automatic_Reasoning_Techniques_for_Non-Serializable_Data-Intensive_Applications/8977562`.

[3] Gowtham Kaki and Suresh Jagannathan. A Relational Framework for Higher-order Shape Analysis. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, ICFP '14, pages 311–324, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2873-9. doi: 10.1145/2628136.2628159. URL `http://doi.acm.org/10.1145/2628136.2628159`.

[4] Gowtham Kaki and G. Ramalingam. Safe Transferable Regions. In Todd Millstein, editor, *32nd European Conference on Object-Oriented Programming (ECOOP 2018)*, volume 109 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:31, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-079-8. doi: 10.4230/LIPIcs.ECOOP.2018.11. URL `http://drops.dagstuhl.de/opus/volltexte/2018/9216`.

[5] Gowtham Kaki, Kartik Nagar, Mahsa Najafzadeh, and Suresh Jagannathan. Alone together: Compositional reasoning and inference for weak isolation. *Proc. ACM Program. Lang.*, 2(POPL):27:1–27:34, December 2017. ISSN 2475-1421. doi: 10.1145/3158115. URL `http://doi.acm.org/10.1145/3158115`.

[6] Gowtham Kaki, Kapil Earanky, KC Sivaramakrishnan, and Suresh Jagannathan. Safe replication through bounded concurrency verification. *Proc. ACM Program. Lang.*, 2(OOPSLA):164:1–164:27, October 2018. ISSN 2475-1421. doi: 10.1145/3276534. URL `http://doi.acm.org/10.1145/3276534`.

[7] Gowtham Kaki, Swarn Priya, KC Sivaramakrishnan, and Suresh Jagannathan. Mergeable replicated data types. *Proc. ACM Program. Lang.*, (OOPSLA), October 2019.

[8] Gowtham Kaki, KC Sivaramakrishnan, and Suresh Jagannathan. Version Control Is for Your Data Too. In Benjamin S. Lerner, Rastislav Bodík, and Shriram Krishnamurthi, editors, *3rd Summit on Advances in Programming Languages (SNAPL 2019)*, volume 136 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:18, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-113-9. doi: 10.4230/LIPIcs.SNAPL.2019.8. URL `http://drops.dagstuhl.de/opus/volltexte/2019/10551`.

[9] KC Sivaramakrishnan, Gowtham Kaki, and Suresh Jagannathan. Declarative Programming over Eventually Consistent Data Stores. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2015, pages 413–424, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3468-6. doi: 10.1145/2737924.2737981. URL `http://doi.acm.org/10.1145/2737924.2737981`.

[10] Todd Warszawski and Peter Bailis. ACIDRain: Concurrency-Related Attacks on Database-Backed Web Applications. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 5–20, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4197-4. doi: 10.1145/3035918.3064037. URL `http://doi.acm.org/10.1145/3035918.3064037`.