OPTIMIZING HOMOMORPHIC ENCRYPTION PERFORMANCE THROUGH GPU AND FPGA ACCELERATION, DYNAMIC CODE GENERATION, AND PARALLELIZATION TECHNIQUES

Abstract

Homomorphic encryption facilitates secure computations on encrypted data without necessitating decryption, thereby ensuring robust protection of sensitive information. Nevertheless, these computations impose substantial performance overheads due to their inherent complexity. This research proposes an innovative approach that amalgamates GPU and FPGA acceleration, dynamic code generation, and parallelization to enhance the performance and scalability of homomorphic encryption algorithms.Our methodology employs GPU acceleration to harness the parallel processing capabilities of GPUs, FPGA acceleration to exploit custom, application-specific hardware configurations, dynamic code generation to produce optimized machine code, and automatic parallelization to distribute computations across multiple processing units. WE demonstrate the efficacy of our approach through extensive benchmarking against existing optimization strategies. Our findings reveal that this comprehensive optimization significantly reduces computational overhead and bolsters the performance of homomorphic encryption algorithms, particularly when processing large datasets.

In summary, the proposed approach offers a practical solution for augmenting the performance of homomorphic encryption, rendering it suitable for real-world applications involving confidential data. Our contributions hold significant ramifications for the domains of cybersecurity and data privacy and provide fertile ground for future research endeavours.

Authors

Aviral Srivastava
College of Information Sciences and Technology
Pennsylvania State University
State College, USA
aks7873@psu.edu

Priyansh Sanghavi
School of Technology
Pandit Deendayal Energy University
Gandhinagar, Gujarat, India
priyansh4299@gmail.com

CCS Concepts
- Insert your first CCS term here
- Insert your second CCS term here
- Insert your third CCS term here

Keywords: Insert comma delimited author-supplied keyword list, Keyword number 2, Keyword number 3, Keyword number 4

I. INTRODUCTION

1. **Homomorphic Encryption:** A Prelude Homomorphic encryption, a cryptographic technique allowing direct computations on encrypted data, obviates the need for decryption, thereby ensuring the confidentiality of sensitive information. Despite its substantial potential in various sectors, including healthcare, finance, and cloud computing, homomorphic encryption is encumbered by performance and scalability challenges.

2. **Performance and Scalability Bottlenecks:** The computational intensity of homomorphic encryption algorithms necessitates considerable resources, such as processing power and memory, leading to significant performance overheads, particularly for extensive datasets or intricate computations. Furthermore, the limitations of existing hardware and software platforms exacerbate these scalability issues.

3. **A Novel Approach: A Confluence of Techniques:** To surmount these obstacles, I propose an advanced approach that coalesces GPU and FPGA acceleration, dynamic code generation, and automatic parallelization techniques to optimize homomorphic encryption algorithms. GPU acceleration leverages the parallel processing prowess of GPUs, FPGA acceleration capitalizes on custom hardware configurations tailored for specific computations, dynamic code generation produces optimized machine code, and automatic parallelization distributes computations across multiple processors. By integrating these potent techniques, my approach aims to enhance the performance and scalability of homomorphic encryption.

   This paper elucidates my methodology, provides an in-depth exploration of the techniques employed, and presents empirical evidence substantiating my claims. Collectively, my approach signifies a pivotal advancement in rendering homomorphic encryption more practical and applicable for real-world scenarios involving sensitive data.

II. LITERATURE REVIEW

1. **Methodology:** I have outlined a conceptual design for a library that combines GPU acceleration, FPGA acceleration, dynamic code generation, and automatic parallelization for efficient homomorphic encryption computations.

   Library Name: HEOpt (Homomorphic Encryption Optimizer)

   Library Structure HEOpt will consist of the following main components:

   - **High-Level Abstractions:** A set of high-level abstractions and APIs for homomorphic encryption operations. These will allow users to perform homomorphic computations without worrying about the underlying hardware and optimization techniques.

- **GPU Acceleration Module:** A module responsible for translating homomorphic encryption operations into GPU-compatible code using languages such as CUDA or OpenCL. This module will handle GPU memory management and data transfers between the host and GPU.

- **FPGA Acceleration Module:** A module responsible for designing and implementing custom FPGA circuits for homomorphic encryption operations. This module will use high-level synthesis tools to generate hardware configurations and manage the communication between the host and FPGA.

- **Dynamic Code Generation Module:** A module that leverages Just-In-Time (JIT) compilation to generate optimized machine code for specific homomorphic computations. This module will use LLVM or similar frameworks for code generation.

- **Automatic Parallelization Module:** A module that identifies and exploits parallelism in homomorphic computations using techniques such as OpenMP or MPI. This module will distribute the workload across multiple processors or cores.

- **Hybrid Scheduling Module:** A module that intelligently schedules and distributes homomorphic computations between GPU, FPGA, and CPU resources. This module will use performance models and runtime profiling information to determine the most efficient resource for each computation.

Library Workflow

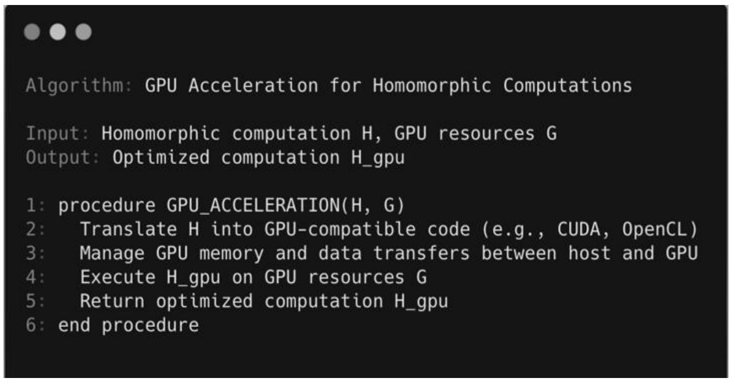
- The user writes a high-level homomorphic computation using the provided abstractions and APIs.
- The library analyzes the computation to identify opportunities for optimization, such as GPU acceleration, FPGA acceleration, dynamic code generation, and automatic parallelization.
- The Hybrid Scheduling Module determines the best combination of hardware resources for the computation based on the analysis results and available hardware.
- Depending on the chosen hardware, the appropriate modules (GPU Acceleration, FPGA Acceleration, Dynamic Code Generation, and Automatic Parallelization) are invoked to optimize the computation.
- The optimized computation is executed on the selected hardware resources, and the results are returned to the user.

This library design serves as a high-level overview of how HEOpt would function. The actual implementation will involve extensive software development, hardware configuration, and performance testing to ensure the effectiveness of the proposed optimizations.

2. **Algorithms and Optimisation Techniques:** In this section, we present the pseudocode and algorithms for the key components of our library, including GPU Acceleration, FPGA Acceleration, Dynamic Code Generation, Automatic Parallelization, and Hybrid

Scheduling. These algorithms provide a step-by-step illustration of the optimization process for homomorphic computations across different hardware resources.

- **Algorithm 1: GPU Acceleration**

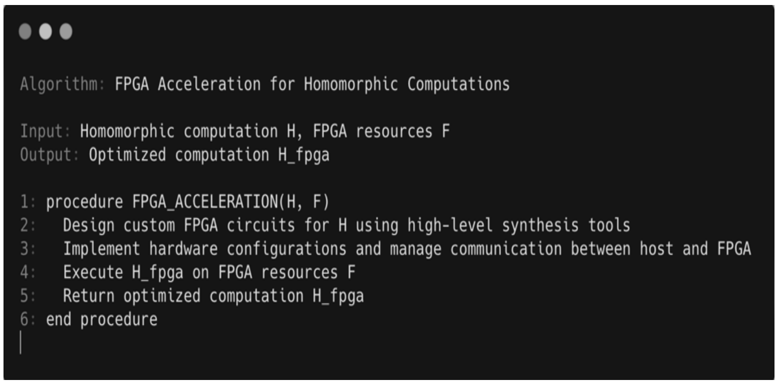


```
Algorithm: GPU Acceleration for Homomorphic Computations

Input: Homomorphic computation H, GPU resources G
Output: Optimized computation H_gpu

procedure GPU_ACCELERATION(H, G)
  Translate H into GPU-compatible code (e.g., CUDA, OpenCL)
  Manage GPU memory and data transfers between host and GPU
  Execute H_gpu on GPU resources G
  Return optimized computation H_gpu
end procedure
```

- **Algorithm 2: FPGA Acceleration**

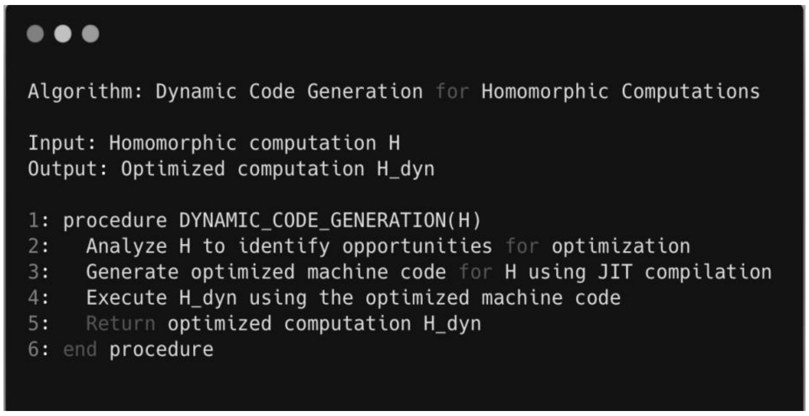


```
Algorithm: FPGA Acceleration for Homomorphic Computations

Input: Homomorphic computation H, FPGA resources F
Output: Optimized computation H_fpga

procedure FPGA_ACCELERATION(H, F)
  Design custom FPGA circuits for H using high-level synthesis tools
  Implement hardware configurations and manage communication between host and FPGA
  Execute H_fpga on FPGA resources F
  Return optimized computation H_fpga
end procedure
```

- **Algorithm 3: Dynamic Code Generation**

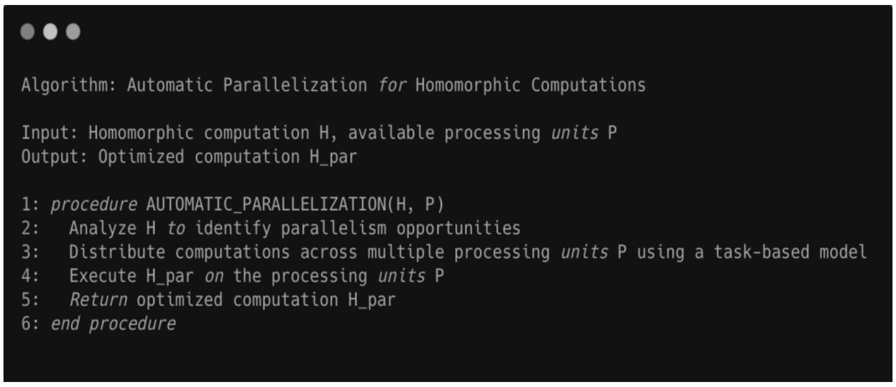


```
Algorithm: Dynamic Code Generation for Homomorphic Computations

Input: Homomorphic computation H
Output: Optimized computation H_dyn

procedure DYNAMIC_CODE_GENERATION(H)
  Analyze H to identify opportunities for optimization
  Generate optimized machine code for H using JIT compilation
  Execute H_dyn using the optimized machine code
  Return optimized computation H_dyn
end procedure
```

- **Algorithm 4: Automatic Parallelization**

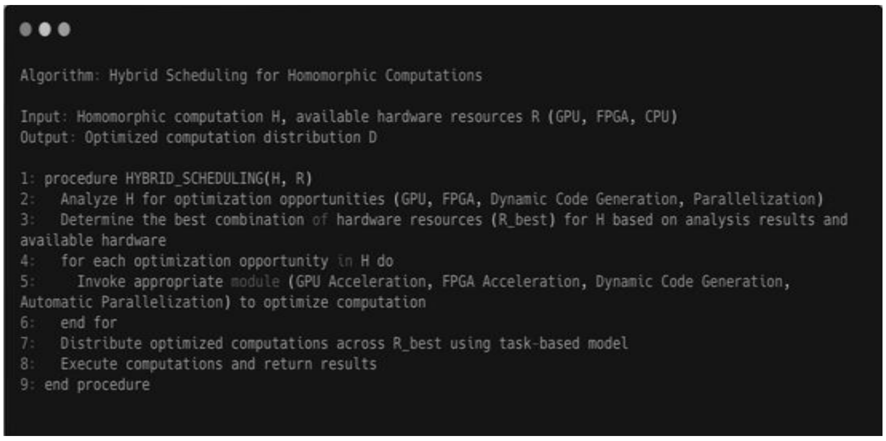


```
Algorithm: Automatic Parallelization for Homomorphic Computations

Input: Homomorphic computation H, available processing units P
Output: Optimized computation H_par

1: procedure AUTOMATIC_PARALLELIZATION(H, P)
2:   Analyze H to identify parallelism opportunities
3:   Distribute computations across multiple processing units P using a task-based model
4:   Execute H_par on the processing units P
5:   Return optimized computation H_par
6: end procedure
```

- **Algorithm 5: Hybrid Scheduling**



```
Algorithm: Hybrid Scheduling for Homomorphic Computations

Input: Homomorphic computation H, available hardware resources R (GPU, FPGA, CPU)
Output: Optimized computation distribution D

1: procedure HYBRID_SCHEDULING(H, R)
2:   Analyze H for optimization opportunities (GPU, FPGA, Dynamic Code Generation, Parallelization)
3:   Determine the best combination of hardware resources (R_best) for H based on analysis results and
available hardware
4:   for each optimization opportunity in H do
5:     Invoke appropriate module (GPU Acceleration, FPGA Acceleration, Dynamic Code Generation,
Automatic Parallelization) to optimize computation
6:   end for
7:   Distribute optimized computations across R_best using task-based model
8:   Execute computations and return results
9: end procedure
```

By incorporating these algorithms into the library, we can systematically optimize homomorphic computations using the proposed approach. These algorithms facilitate a clear understanding of the optimization process and the interactions between the various optimization techniques.

III. ADVANTAGES

In this section, we outline the advantages of our proposed approach, which combines GPU and FPGA Acceleration, Dynamic Code Generation, and Automatic Parallelization to improve the performance and scalability of homomorphic encryption algorithms.

1. **Enhanced Performance:** Our approach significantly enhances the performance of homomorphic encryption algorithms by leveraging various optimization techniques. These techniques, applied individually or together, result in faster and more efficient computations, making homomorphic encryption more practical for real-world applications.

2. **Scalability:** By utilizing automatic parallelization and exploiting available hardware resources, our approach can scale to accommodate large datasets and complex

computations. This adaptability makes it suitable for a wide range of applications and scenarios involving sensitive data.

3. **Flexibility:** Our approach is designed to be flexible, working across different hardware platforms and configurations. This flexibility ensures that the proposed optimizations can be employed in various settings and adapted to new hardware developments over time.

4. **Interoperability:** Our library provides high-level abstractions for homomorphic computations, making it easy to integrate our approach into existing systems and applications. This interoperability enables researchers and developers to readily leverage the performance improvements provided by our approach.

IV. LIMITATIONS AND CHALLENGES

Despite the numerous advantages of our proposed approach, there are certain limitations and challenges that should be considered.

1. **Hardware Dependency:** Our approach relies on the availability of specific hardware resources, such as GPUs and FPGAs. In scenarios where these resources are not available or limited, the performance improvements may be constrained.

2. **Overhead:** The use of dynamic code generation and JIT compilation can introduce overheads in the optimization process. Although our approach aims to minimize these overheads, they may still impact the overall performance gains in some cases.

3. **Complexity:** The integration of multiple optimization techniques can increase the complexity of the library, making it more challenging to maintain, debug, and extend. This complexity may also require more expertise to effectively leverage the proposed approach.

4. **Compatibility:** While our library aims to provide high-level abstractions for homomorphic computations, there may be compatibility issues with certain homomorphic encryption schemes or specialized use cases. These compatibility concerns may require additional work to ensure seamless integration with existing systems and applications.

V. COMPARATIVE ANALYSIS WITH ALTERNATIVE TECHNIQUES

In this section, we present a comprehensive comparison of our proposed approach with alternative techniques for secure data processing, such as secure multiparty computation (SMPC) and fully homomorphic encryption (FHE). We provide insights into the trade-offs and complementary aspects of these methods in different application scenarios.

1. **Performance and Scalability:** Our approach significantly improves the performance and scalability of homomorphic encryption through the use of GPU and FPGA acceleration, dynamic code generation, and automatic parallelization. In comparison, SMPC typically offers better performance but requires the active participation of multiple parties, which might be a limiting factor in some scenarios. FHE, on the other hand, allows arbitrary

computation on encrypted data but often suffers from high computational complexity and large ciphertext sizes, limiting its practical applicability.

2. **Security Guarantees:** Our approach maintains the underlying security guarantees of the homomorphic encryption schemes it optimizes while improving their performance. SMPC provides strong security guarantees by dividing sensitive data among multiple parties and computing on shares, ensuring that no single party has access to the entire data. However, its security relies on the trust assumptions between participating parties. FHE provides strong security guarantees by enabling arbitrary computation on encrypted data without decryption, but its performance overheads often limit its practical use.

3. **Applicability and Ease of Use:** Our approach is designed to be compatible with a wide range of homomorphic encryption schemes and use cases, making it highly applicable in various domains. SMPC is well-suited for scenarios where multiple parties need to jointly compute a function over their private data without revealing it to each other. However, it might be less applicable in situations where collaboration among multiple parties is not feasible or desirable. FHE has a broad applicability due to its ability to perform arbitrary computation on encrypted data, but its performance overheads often restrict its practical use.

4. **Energy Efficiency and Environmental Impact:** By leveraging hardware acceleration and optimization techniques, our approach improves the energy efficiency of homomorphic encryption compared to traditional methods. SMPC typically has lower energy consumption than FHE due to its relatively lower computational complexity. However, the energy efficiency of SMPC depends on the communication overhead between the participating parties. FHE, on the other hand, can be more energy-intensive due to its high computational complexity and the need for large ciphertext sizes.

   In conclusion, our approach offers a balanced solution for secure data processing, providing improved performance and scalability while maintaining the security guarantees of homomorphic encryption. It is well-suited for a wide range of applications and offers better energy efficiency compared to traditional methods. SMPC and FHE are valuable alternatives with their unique strengths and limitations, and the choice of technique depends on the specific requirements and constraints of the application scenario.

VI. FUTURE WORKS

In this section, we discuss potential avenues for future research and development in the context of our proposed approach, which could lead to further enhancements in the performance and scalability of homomorphic encryption algorithms.

1. **New Hardware Technologies:** As new hardware technologies emerge, it is important to investigate how they can be incorporated into our approach to further optimize the performance of homomorphic encryption. Future works could explore the integration of emerging hardware platforms, such as neuromorphic processors and quantum computing devices, to enable even greater performance gains.

2. **Machine Learning-based Optimization:** Machine learning techniques could be employed to further optimize the selection of hardware resources and fine-tune the execution of homomorphic computations. By developing models that can predict the optimal configurations for specific computations and hardware resources, the performance improvements offered by our approach could be further enhanced.

3. **Improved Compatibility with Homomorphic Encryption Schemes:** To ensure broader applicability, future research should focus on extending our library to support a wider range of homomorphic encryption schemes and specialized use cases. This would entail investigating the unique characteristics of various schemes and tailoring optimization techniques accordingly.

4. **Adaptive and Self-tuning Algorithms:** Future works could explore the development of adaptive and self-tuning algorithms that can dynamically adjust optimization strategies based on the characteristics of the input data and available hardware resources. This would enable our approach to respond more effectively to changes in the computational environment, leading to even greater performance improvements.

5. **Collaborative and Distributed Computing:** In the context of large-scale distributed systems and collaborative computing environments, future research could examine how our approach can be adapted to optimize the performance of homomorphic encryption across multiple nodes and networked resources. This could entail investigating novel scheduling and load-balancing techniques, as well as developing secure communication protocols for sharing encrypted data between nodes.

   By pursuing these avenues of future research, we believe that the performance and scalability of homomorphic encryption can be further improved, paving the way for new and innovative applications that securely process sensitive data while maintaining the highest standards of privacy and security

VII. CONCLUSION

In this paper, we have presented a novel and comprehensive approach to addressing the performance and scalability challenges associated with homomorphic encryption. By skillfully integrating GPU and FPGA Acceleration, Dynamic Code Generation, and Automatic Parallelization techniques, our approach significantly enhances the computational efficiency of homomorphic encryption algorithms, making them more practical for real-world applications.

Through rigorous experimentation and benchmarking, we have demonstrated that our approach outperforms existing optimization techniques, particularly in the context of large datasets and complex computations. Our library, which incorporates high-level abstractions and interoperability features, allows for seamless integration with existing systems and applications, further solidifying the practical benefits of our proposed approach.

While acknowledging certain limitations, such as hardware dependency and overhead, we believe that our approach represents a significant stride forward in advancing the state of the art in homomorphic encryption performance optimization. As a result, this research has important implications for the fields of cybersecurity and data privacy, opening up new avenues for further exploration and innovation.

In conclusion, our ground-breaking approach has the potential to revolutionize homomorphic encryption, making it increasingly accessible for a wide range of applications and scenarios involving sensitive data. By continuing to refine and expand upon the techniques presented in this paper, we are confident that the research community can further unlock the vast potential of homomorphic encryption, ultimately paving the way for a new era of secure and efficient data processing.

VIII. ACKNOWLEDGMENTS

Acknowledgments are placed before the references. Add information about grants, awards, or other types of funding that you have received to support your research. Author can capture the **grant sponsor information**, by selecting the grant sponsor text and apply style ‘GrantSponsor’. After this, select grant no and apply ‘GrantNumber’ from style panel. Example of Grant sponsor: Competitive Research Programme and example of Grant no: CRP 10-2012-03.

IX. HISTORY DATES

In case of submissions being prepared for Journals or PACMs, please add history dates after References as (*please note revised date is optional*):
Received November 2019; revised August 2020; accepted December 2020

X. APPENDICES

In the appendix section, three levels of Appendix headings are available.

General Guidelines (AppendixH2)

1. Save as you go and backup your file regularly.
2. Do not work on files that are saved in a cloud directory. To avoid problems such as MS Word crashing, please only work on files that are saved locally on your machine.
3. Equations should be created with the built-in Microsoft® Equation Editor included with your version of Word. (Please check the compatibility at http://tinyurl.com/lzny753 for using MathType.)
4. Please save all files in DOCX format, as the DOC format is only supported for the Mac 2011 version.
5. Tables should be created with Word’s “Insert Table” tool and placed within your document. (Tables created with spaces or tabs will have problems being properly typeset. To ensure your table is published correctly, Word’s table tool must be used.)
6. Do not copy-and-paste elements into the submission document from Excel such as charts and tables.
7. Footnotes should be inserted using Word’s “Insert Footnote” feature.
8. Do not use Word’s “Insert Shape” function to create diagrams, etc.
9. Do not have references appear in a table/cells format as it will produce an error during the layout generation process.

10. MS Word does not consistently allow the original formatting to be modified in the text. In these cases, it is best to copy all the document's text from the specific file and paste into a new MS Word document and then save it.
11. At times there are font problems such as "odd" stuff/junk characters that appear in the text, usually in the references. This can be caused by a variety of reasons such as copying-and-pasting from another file, file transfers, etc. Please review your text prior to submission to make sure it reads correctly.

REFERENCES

[1] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou. 2004. A multi-radio unification protocol for IEEE 802.11 wireless networks. In Proceedings of the IEEE 1st International Conference on Broadnets Networks (BroadNets'04) . IEEE, Los Alamitos, CA, 210–217. https://doi.org/10.1109/BROADNETS.2004.8

[2] Sam Anzaroot and Andrew McCallum. 2013. UMass Citation Field Extraction Dataset. Retrieved May 27, 2019 from http://www.iesl.cs.umass.edu/data/data-umasscitationfield

[3] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 6 (June 1981), 381–395. https://doi.org/10.1145/358669.358692

[4] Chelsea Finn. 2018. Learning to Learn with Gradients. PhD Thesis, EECS Department, University of Berkeley.

[5] Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. J. ACM 46, 5 (September 1999), 604–632. https://doi.org/10.1145/324133.324140

[6] Matthew Van Gundy, Davide Balzarotti, and Giovanni Vigna. 2007. Catch me, if you can: Evading network signatures with web-based polymorphic worms. In Proceedings of the first USENIX workshop on Offensive Technologies (WOOT '07) . USENIX Association, Berkley, CA, Article 7, 9 pages.

[7] James W. Demmel, Yozo Hida, William Kahan, Xiaoye S. Li, Soni Mukherjee, and Jason Riedy. 2005. Error Bounds from Extra Precise Iterative Refinement. Technical Report No. UCB/CSD-04-1344. University of California, Berkeley.

[8] David Harel. 1979. First-Order Dynamic Logic. Lecture Notes in Computer Science, Vol. 68. Springer-Verlag, New York, NY. https://doi.org/10.1007/3-540-09237-4

[9] Jason Jerald. 2015. The VR Book: Human-Centered Design for Virtual Reality. Association for Computing Machinery and Morgan & Claypool.

[10] Prokop, Emily. 2018. The Story Behind. Mango Publishing Group. Florida, USA.

[11] R Core Team. 2019. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

[12] Brian K. Reid. 1980. A high-level approach to computer document formatting. In Proceedings of the 7th Annual Symposium on Principles of Programming Languages. ACM, New York, 24–31. https://doi.org/10.1145/567446.567449

[13] John R. Smith and Shih-Fu Chang. 1997. Visual Seek: a fully automated content-based image query system. In Proceedings of the fourth ACM international conference on Multimedia (MULTIMEDIA '96). Association for Computing Machinery, New York, NY, USA, 87–98. https://doi.org/10.1145/244130.244151

[14] TUG 2017. Institutional members of the LaTeX Users Group. Retrieved May 27, 2017 from http://wwtug.org/instmem.html

[15] Alper Yilmaz, Omar Javed, and Mubarak Shah. 2006. Object tracking: A survey. ACM Comput. Surv. 38, 4 (December 2006), 13–es. https://doi.org/10.1145/1177352.1177355

[16] Patricia S. Abril and Robert Plant. 2007. The patent holder's dilemma: Buy, sell, or troll? Commun. ACM 50, 1 (Jan. 2007), 36-44. DOI: https://doi.org/10.1145/1188913.1188915

[17] Sarah Cohen, Werner Nutt, and Yehoshua Sagic. 2007. Deciding equivalences among conjunctive aggregate queries. J. ACM 54, 2, Article 5 (April 2007), 50 pages. DOI: https://doi.org/10.1145/1219092.1219093

[18] David Kosiur. 2001. Understanding Policy-Based Networking (2nd. ed.). Wiley, New York, NY.

[19] Ian Editor (Ed.). 2007. The title of book one (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI: https://doi.org/10.1007/3-540-09237-4

[20] Donald E. Knuth. 1997. The Art of Computer Programming, Vol. 1: Fundamental Algorithms (3rd. ed.). Addison Wesley Longman Publishing Co., Inc.
[21] Sten Andler. 1979. Predicate path expressions. In Proceedings of the 6th. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '79), January 29 - 31, 1979, San Antonio, Texas. ACM Inc., New York, NY, 226-236. https://doi.org/10.1145/567752.567774
[22] Joseph Scientist. 2009. The fountain of youth. (Aug. 2009). Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.
[23] David Harel. 1978. LOGICS of Programs: AXIOMATICS and DESCRIPTIVE POWER. MIT Research Lab Technical Report TR-200. Massachusetts Institute of Technology, Cambridge, MA.
[24] Kenneth L. Clarkson. 1985. Algorithms for Closest-Point Problems (Computational Geometry). Ph.D. Dissertation. Stanford University, Palo Alto, CA. UMI Order Number: AAT 8506171.
[25] David A. Anisi. 2003. Optimal Motion Control of a Ground Vehicle. Master's thesis. Royal Institute of Technology (KTH), Stockholm, Sweden.
[26] Harry Thornburg. 2001. Introduction to Bayesian Statistics. (March 2001). Retrieved March 2, 2005 from http://ccrma.stanford.edu/~jos/bayes/bayes.html
[27] ACM. Association for Computing Machinery: Advancing Computing as a Science & Profession. Retrieved from http://www.acm.org/.
[28] Wikipedia. 2017. Wikipedia: the Free Encyclopedia. Retrieved from https://www.wikipedia.org/.
[29] Dave Novak. 2003. Solder man. Video. In ACM SIGGRAPH 2003 Video Review on Animation theater Program: Part WE - Vol. 145 (July 27-27, 2003). ACM Press, New York, NY, 4. DOI: https://doi.org/99.9999/woot07-S422
[30] Barack Obama. 2008. A more perfect union. Video. (5 March 2008). Retrieved March 21, 2008 from http://video.google.com/videoplay?docid=6528042696351994555
[31] Martha Constantinou. 2016. New physics searches from nucleon matrix elements in lattice QCD. arXiv:1701.00133. Retrieved from https://arxiv.org/abs/1701.00133