Approximate Computing, Intelligent Computing

Lieven Eeckhout
Ghent University

The first time I heard about the notion of approximate computing was in 2006 when I was attending the Wild and Crazy Ideas (WACI) session at the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). I vividly remember the talk by Ravi Nair from the IBM T.J. Watson Research Center in which he made the case for approximate computing. This talk was memorable for two reasons. For one, it was so hilarious—by far the most exciting talk I have ever attended—and completely in the spirit of the WACI session. The speech was a plagiarized version of one by a leading candidate in the American presidential elections. The slides and speech are at http://cseweb.ucsd.edu/~swanson/WACI-VI/. I encourage you all to read it, aloud as you would do on stage—I guarantee that you will find it funny!

Not only was the talk funny, but it was also eye-opening, at least to me. We are so used to precise computing in which designs must be verified as precise, implementations must be tested to ensure precision, programs must be precise, and algorithms must compute precise outputs. However, for many real problems, an approximate solution is sufficient. In fact, a lot of interesting problems that we want to solve using computers today need only approximate answers. Achieving meaningless precision in such cases incurs excessive power and energy to perform computations that, at the end of the day, are useless.

Ravi Nair further argues that approximate computing could be considered intelligent computing, because it uses energy resources to perform exact computation only when needed and approximate whenever possible. He provides several suggestions on when to apply approximate computing: to dynamically adapt value precision, relax data coherence whenever tolerable, dynamically tweak the number of iterations in a convergence loop, and allow a variable to use a default value if it misses in the cache. Interestingly, several (if not all) of these suggestions or variations thereof have been explored over the past few years.

The rest is history. Over the past couple years, approximate computing emerged as a novel computing systems paradigm that trades off quality for efficiency. Technology trends and application trends greatly helped spark interest in this new paradigm. On the technology side, processor architects are forced to improve efficiency as much as possible in times of dark CMOS scaling. On the application side, there is a growing share of applications that are inherently approximate, such as when the input is noisy, the algorithm is probabilistic, and/or the output can be imprecise. The number of emerging workloads that are inherently approximate are plenty—just think about artificial intelligence, machine learning, optimization, search, multimedia, sensor data, and so on. All these workloads share the common property that a slight decrease in accuracy is considered acceptable. For example, a couple incorrect pixels in a video stream are likely to go undetected by the user. Likewise, there is typically some tolerance for inaccuracy in machine-learning problems as long as the answer to the problem is "close enough."

Published by the IEEE Computer Society
0272-1732/18/$33.00 ©2018 IEEE

Approximate computing has important implications across the system stack. The user needs to get involved and decide when and to what extent inaccuracy in quality is acceptable. For example, the expected quality of a video may vary across users, devices, and contexts; in particular, at low battery operation, the user may be more lenient to low-quality video display than when connected to the grid. The programmer also has a key role to play. If the user specifies a certain level of tolerable inaccuracy, the programmer needs to make sure that this is not violated. Also, while some error in the video output is tolerable, a deviation in control flow that leads to a software crash clearly is not. The hardware in its turn can then exploit the features pointed out by software that are tolerable to inaccuracy to improve efficiency as much as possible. Different hardware knobs enable the computer architect to turn tolerable inaccuracy into efficiency gains.

Because of these close interactions between the user, software, and hardware, approximate computing introduces fundamentally new research avenues and requires vertical optimization. In the classical computing world, improving efficiency still requires 100-percent correct execution. Under approximate computing, some level of error is tolerable for larger gains in efficiency. The paradigm of approximate computing has led to a flurry of research activity over the past few years. This special issue explores exciting, new ideas in the vast design space of approximate computing.

I wholeheartedly thank the guest editors Natalie Enright Jerger (University of Toronto) and Joshua San Miguel (University of Wisconsin-Madison) for pulling together this special issue. I owe Natalie and Joshua a big thank you for having done such a fantastic job. I hope you, the reader, will enjoy reading the theme articles as much as I did. I also recommend you read the guest editorial that introduces these articles.

Finally, I want to highlight two award testimonials. Susan Eggers from the University of Washington received the 2018 ACM/IEEE CS Eckert-Mauchly award for "outstanding contributions to simultaneous multithreading processor architectures and multiprocessor sharing and consistency." She is the first female computer architect to receive the most prestigious award in our field, an award that was instigated 40 years ago. The award testimonial includes an overview of Susan's excellent contributions along with the moving acceptance speech she gave at the International Symposium on Computer Architecture (ISCA) in Los Angeles on June 5, 2018. Big congrats to Susan for this very well-deserved award!

The second award testimonial covers the ISCA Influential Paper award, which recognizes the most impactful paper published 15 years ago in ISCA. The awarded paper is "Temperature-Aware Microarchitecture" by Kevin Skadron, Mircea R. Stan, Wei Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and David Tarjan, which was published in ISCA 2003. Antonio Gonzalez from the Universitat Politècnica de Catalunya (UPC) chaired the award committee and provides a perspective on this paper and why it has been so influential for our field. Congratulations to the authors for their highly impactful work.

As always, I wish you a happy reading.

ABOUT THE AUTHOR

Lieven Eeckhout is a professor in the Department of Electronics and Information Systems at Ghent University. Contact him at lieven.eeckhout@ugent.be.