ABSTRACT: Byte-addressable non-volatile memory technology is emerging as an alternative for DRAM for main memory. This new Non-Volatile Main Memory (NVMM) allows programmers to store data persistently in memory instead of in files, thereby providing a substantial performance boost. However, computer systems reorder memory operations and utilize volatile caches for better performance, making it difficult to ensure a consistent state in NVMM. Intel recently announced a new set of persistence instructions, clflushopt, clwb, and pcommit. These new instructions make it possible to implement fail-safe code on NVMM, but few workloads have been written or characterized using these new instructions.

In this talk, I will discuss persistency, performance and programmability challenges that arise with non-volatile main memory. First, we describe how persistency programming can be achieved using write-ahead logging based transactions. We implement several common data structures and kernels and evaluate the performance overhead incurred over traditional non-persistent implementations. In particular, we find that persistence instructions occur in clusters along with expensive fence operations, they have long latency, and they add a significant execution time overhead, on average by 20% over code with logging but without fence instructions to order persists. To deal with this overhead and alleviate the performance bottleneck, we propose to speculate past long latency persistency operations using checkpoint-based processing. Our speculative persistence architecture reduces the execution time overheads to only 4%. Second, logging code incurs significant increase in performance overheads and write amplification. While logging can be performed in hardware, a hardware approach unnecessarily restricts the number and size of durable transactions. We propose a software-supported hardware logging that remove the limitations, but at the same time perform as well as hardware logging. Finally, I will present Lazy Persistency, a software technique that allows code to run free of logging and persist barrier overheads. Upon a crash, checksum is used to detect persistency failure and re-execute code that failed to persist. Lazy Persistency achieves performance comparable to native execution and does not require hardware instruction support or any persistency models, hence it can be readily deployed.

BIO: Yan Solihin is a Charles N. Millican Chair Professor of Computer Science and Director for Cybersecurity and Privacy Cluster at the University of Central Florida. He has served as a Program Director at the Division of Computer and Network Systems (CNS) at the National Science Foundation. His responsibilities included managing the Computer Systems Research (CSR) cluster, Scalable Parallelism in the eXtreme (SPX), and Secure and Trustworthy Cyberspace (SaTC), among others. He is an IEEE Fellow, recognized “for contributions to shared cache hierarchies and secure processors.” He is also an ACM member.

He obtained his B.S. degree in computer science from Institut Teknologi Bandung in 1995, B.S. degree in Mathematics from Universitas Terbuka Indonesia in 1995, M.A.Sc degree in computer engineering from Nanyang Technological University in 1997, and M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1999 and 2002. He is a recipient of 2010 and 2005 IBM Faculty Partnership Award, 2004 NSF Faculty Early Career Award, and 1997 AT&T Leadership Award. He is listed in the ISCA and HPCA Hall of Fame. His research interests include computer architecture, memory hierarchy design, non-volatile memory architecture, programming models, and workload cloning. He has published more than 70 papers in computer architecture and performance modeling, and authored 40+ patents. He has released several software packages to the public: ACAPP - a cache performance model toolset, HeapServer - a secure heap management library, Scaltool - parallel program scalability pinpointer, and Fodex - a forensic document examination toolset. He has written two graduate-level textbooks, including Fundamentals of Parallel Multicore Architecture, CRC Press, 2015.