Near-Memory Data Services

Babak Falsafi
EPFL

Large-scale IT services are increasingly migrating from storage to memory because of tight data access latency requirements.[1] Online services such as search and social connectivity require managing massive amounts of data while maintaining a low request tail latency. Similarly, analytic engines for business intelligence are increasingly in-memory to minimize query response time. The net result is that memory is taking center stage in server design as datacenter operators maximize memory capacity integrated per server to increase throughput per total cost of ownership.[2]

The slowdown in Dennard scaling has further pushed server designers toward memory efficiency.[3] A large spectrum of data services have moderate computational requirements but large energy footprints due to data movement. Custom scale-out processors (such as Cavium ThunderX) dramatically increase the overall bandwidth demand by dedicating much of the processor silicon area to cores specialized for data access.[4] Although recent advances in die stacking and embedded DRAM promise higher on-chip cache capacity, these technologies have fallen short of integrating enough memory to reduce data traffic effectively. Emerging SerDes memory interfaces mitigate the bandwidth bottleneck but do not alleviate the energy footprint. Storage-class memory will only exacerbate these trends by introducing a scalable path for per-server memory capacity.[5] The combined effect of these trends argues for migrating data services near memory through stacking a thin layer of logic on DRAM or storage-class memory.

Near-memory processing has several limitations and opportunities. First, the successful adoption of near-memory processing platforms depends highly on their programmability. A unified shared address space is the obvious candidate to support near-memory processing for data services. Although data services migrate near memory, that memory itself must remain accessible to the processor for computation that benefits from the conventional processor-centric memory access model. A unified address space means that near-memory processing would require support for virtual to physical address translation, and coherence between the processor cache hierarchy and the near-memory logic. These technologies must all be revisited because conventional solutions are designed for processors and cache hierarchy, not multigigabyte data partitions. Moreover, thermal and power constraints limit the complexity of logic needed to support these near memory.

Near-memory processing would be highly effective if data services were localized to memory partitions. Unlike the processor-centric model, the near-memory model forgoes the flexibility of the random access model for better efficiency in the sequential or partitioned access model.[6] Although cross-partition (random) accesses can be made feasible, they offset the gains from near-memory processing and may lead to inferior results as compared to the processor-centric model. The latter is similar to the data-parallel programming paradigm (for example, that used in GPUs) but mostly targets data management operations (such as scan, filter, and search), in which the bulk of the work is in data structure traversals with little computation. Data management algorithms conventionally considered inferior in the processor-centric model (for example, sort join versus hash join) can now appear to be superior owing to their preference for sequential memory accesses.[7]

Finally, the tight thermal and power constraints limit the complexity of logic near memory. Many frequent operations (such as data structure traversals) even for latency-sensitive scale-out services can run at much lower frequencies (for example, hundreds of megahertz) than those in servers today.[8] The latter enables near-memory data services at an order of magnitude higher degree of parallelism than previously thought feasible.

Editors' note: We invited a group of experts from industry and academia to discuss the opportunities, challenges, and experiences driving the resurgence of near-data processing research today. The following are their views on the topic. *—Rajeev Balasubramonian and Boris Grot*

Near-threshold logic is a promising technology to enable operation at such low frequencies for lightweight data services with minimal power and maximum parallelism.[9] It also offers a much wider spectrum of frequency/voltage modes to enable dynamically adjusting the parallelism and frequency near memory relative to the power requirements of a particular service. MICRO

References

1. M. Ferdman et al., "Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware," *Proc. 17th Int'l Conf. Architecture Support for Programming Languages and Operating Systems*, 2012, pp. 37–48.
2. B. Grot et al., "Optimizing Data-Center TCO with Scale-Out Processors," *IEEE Micro*, Sept./Oct. 2012, pp. 52–63.
3. N. Hardavellas et al., "Toward Dark Silicon in Servers," *IEEE Micro*, July/Aug. 2011, pp. 6–15.
4. P. Lotfi-Kamran et al., "Scale-Out Processors," *Proc. 39th Ann. Int'l Symp. Computer Architecture*, 2012, pp. 500–511.
5. "3D XPoint Technology Revolutionizes Storage Memory," Intel; www.intel.com/content/www/us/en/architecture-and-technology/3d-xpoint-technology-animation.html.
6. S.H. Pugsley et al., "NDC: Analyzing the Impact of 3D-Stacked Memory+Logic Devices on MapReduce Workloads," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software*, 2014, pp. 190–200.
7. N. Mirzadeh et al., "Sort vs. Hash Join Revisited for Near-Memory Execution," *Proc. 5th Workshop Architectures and Systems for Big Data*, 2015; http://acs.ict.ac.cn/asbd2015/papers.
8. J. Menon et al., "Memory Processing Units," Hot Chips 2014; http://research.cs.wisc.edu/vertical/papers/2014/hotchips-poster-mpu.pdf.
9. A. Pahlevan et al., "Towards Near-Threshold Server Processors," to be published in *Proc. Design, Automation and Test in Europe*, 2016.

Babak Falsafi is the founding director of the EcoCloud research center and a professor in the School of Computer and Communication Sciences at EPFL. Contact him at babak.falsafi@epfl.ch.

Automata Processing: The Memory Is the Processor!

Mircea Stan
Kevin Skadron
University of Virginia

Processor-in-memory (PIM) solutions try to address the Von Neumann bottleneck by reducing memory latency and increasing bandwidth by physically locating the CPU and memory (DRAM) on the same die. This reduces the distance that data need to travel (reducing latency) and increases the number of possible electrical connections (increasing bandwidth). Past attempts at PIMs have had limited success, mainly because of difficulties in reconciling the characteristics of processor (high performance, leaky, many metal layers) and DRAM (relatively slow, low leakage, just a few metal layers) semiconductor technologies. But another fundamental issue with conventional PIM is that it doesn't really solve the Von Neumann bottleneck (because the processor still needs to fetch data and instructions from memory every clock cycle); it just makes it less severe by reducing the constants.

Micron Technology's Automata Processor[1] (AP; www.micronautomata.com) is a fundamentally new hardware computing architecture that attacks the Von Neumann bottleneck straight-on by directly using the memory for computation—thus, "the memory is the processor." The AP is especially powerful because it implements natively, in hardware, the nondeterministic finite automata (NFA) paradigm from classic computer science theory. The AP is therefore an accelerator designed for symbolic pattern matching. The nondeterminism property allows many states to be active at once, and thus allows the AP to follow many active paths (which one can think of as candidate matches) in parallel. In this regard, it bears some similarity to quantum computing (but quantum computing uses superposition, whereas the AP achieves its parallelism by converting time complexity into space complexity, by mapping candidate solutions to NFA states). The AP can perform high-speed, comprehensive search and analysis of complex, unstructured data streams and is especially effective at tasks such as scanning input for matches against a large set of rules (for example, deep packet inspection); imprecise, "fuzzy" string matching in the presence of errors or mutations (such as DNA search and alignment); and combinatorial search problems (such as association rule mining). For example, we have observed speedups (relative to a single-threaded CPU) of more than 1,000 times on several DNA alignment and search tasks, and over 100 times on frequent item-set mining.[2] The AP can be programmed in various ways, including regular expressions, direct implementation of NFAs, and RAPID, a new, C-like programming language.[3]

The AP architecture achieves its high parallelism and energy efficiency by using DRAM as a multiple-instruction, single-data (MISD) architecture in which the high bit-level parallelism of DRAM is used to allow every NFA state to simultaneously perform a lookup operation and react to the current input token. In the current generation, in a conservative 50-nm DRAM process, each chip implements more than 49,000 states, all of which are concurrently scanning and responding to the input stream, and a rich, reconfigurable routing network to connect these states into an NFA. Micron's initial PCI Express boards will hold 32 chips, thus providing more than 1.5 million states, all operating concurrently. Future implementations of the AP will use progressively more advanced DRAM processes, with an increasing number of state elements, higher performance, and lower power. The first-generation AP products will be PCIe boards, but other modes of integrating AP technology into the system architecture (for example, via 3D stacking), and integration of diverse accelerators into a coherent heterogeneous system, are exciting areas for research. The Center for Automata Processing at the University of Virginia (http://cap.virginia.edu) is spearheading development of a research community around automata processing and is actively looking for new members. MICRO

References

1. P. Dlugosch et al., "An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing," *IEEE Trans. Parallel and Distributed Systems*, Dec. 2014, pp. 3088–3098.
2. K. Wang, M. Stan, and K. Skadron, "Association Rule Mining with the Micron Automata Processor," *Proc. IEEE Int'l Parallel and Distributed Processing Symp.*, 2015, pp. 689–699.
3. K. Angstadt, W. Weimer, and K. Skadron, "RAPID Programming of Pattern-Recognition Processors," to be published in *Proc. ACM Int'l Symp. Architectural Support for Programming Languages and Operating Systems* (ASPLOS), 2016.

Mircea Stan is a professor in the Electrical and Computer Engineering Department at the University of Virginia. Contact him at mircea@virginia.edu.

Kevin Skadron is the Harry Douglas Forsyth Professor and Department Chair for Computer Science at the University of Virginia. Contact him at skadron@virginia.edu.

Overcoming Challenges to Near-Data Processing

Nuwan Jayasena
AMD Research

With the increasing significance of data-intensive applications, emphasis in computing is shifting from processing to data. This shift will only accelerate as more and more data consists of complex objects such as images, videos, and data streams generated by myriad sensors. Simultaneously, these unstructured data are limiting the effectiveness of traditional caching techniques for reducing bandwidth bottlenecks.[1] These trends, coupled with the increasing emphasis on power efficiency, highlight the energy cost of moving data to and from processors as a key limiting factor in our ability to continue scaling computing system capabilities. Near-data processing (NDP) is a promising approach for curtailing these data movement overheads, and several studies have demonstrated remarkable energy savings over traditional systems.[2–4] However, broad applicability and commercial viability of NDP is likely to be determined by ease of use and economic considerations.

Early NDP efforts failed to achieve widespread adoption because of four major challenges, but recent advances present opportunities to address each of these challenges.

The first challenge is implementation technology. Although prior attempts required the integration of high-density memory and computing logic on the same die, the emergence of 3D die stacking enables tight integration of logic and memory implemented on separate dies in different process technologies (see www.hsafoundation.com).

The second challenge is a processor architecture that can use the high bandwidth enabled by proximity to memory.

Early NDP research required custom architectures, entailing not only the design efforts but also the daunting task of fostering a developer community. Today, GPGPU architectures enable effective utilization of high bandwidth, have established developer bases, and are viable within the thermal constraints of memory modules.[5]

The third challenge is the development of interfaces that allowed near-memory computing units as well as external processors to access memory. Early efforts required the design and adoption of custom memory interfaces. Recent die-stacked memory interface standards (such as High Bandwidth Memory[6]) and off-chip memory interfaces that expose load-store semantics (such as Hybrid Memory Cube[7]) meet nearly all the memory interface needs of NDP.

The fourth challenge is programming models. Although prior efforts had to develop programming abstractions from the ground up, NDP today can build atop frameworks such as the Heterogeneous System Architecture[8] and the associated software tools for accelerators.

Technologies that help address these challenges are driven by the needs of existing markets, such as graphics and high-performance computing. Some NDP-specific adaptations are necessary, especially in the case of the latter two challenges: memory interfaces and software abstractions. Furthermore, scalability stands out as a new challenge. NDP organizations that can cost-effectively scale to meet the memory capacity requirements of data-intensive applications as well as software frameworks that ease development for such scalable NDP platforms remain fertile research areas. This combination of building atop "commodity" technologies driven by established markets and focusing additional research in NDP-specific challenges presents a promising path to not only realizing NDP but also enabling wide deployment and broad accessibility to developers across various application domains. MICRO

References

1. M. Ferdman et al., "Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware," *Proc. 17th Int'l Conf. Architecture Support for Programming Languages and Operating Systems*, 2012, pp. 37–48.
2. R. Nair et al., "Active Memory Cube: A Processing-in-Memory Architecture for Exascale Systems," *IBM J. Research and Development*, Mar. 2015, pp. 17:1–17:14.
3. S. Pugsley et al., "NDC: Analyzing the Impact of 3D-Stacked Memory+Logic Devices on MapReduce Workloads," *Proc. Int'l Symp. Performance Analysis of Systems and Software*, 2014, pp. 190–200.
4. D. Zhang et al., "TOP-PIM: Throughput-Oriented Programmable Processing in Memory," *Proc. 23rd Int'l Symp. High-Performance Parallel and Distributed Computing*, 2014, pp. 85–98.
5. Y. Eckert, N. Jayasena, and G. Loh, "Thermal Feasibility of Die-Stacked Processing in Memory," *Proc. 2nd Workshop Near-Data Processing*, 2014; www.cs.utah.edu/wondp/eckert.pdf.
6. *High Bandwidth Memory (HBM) DRAM*, JEDEC, JESD235A, 2015; www.jedec.org/standards-documents/results/jesd235.
7. *Hybrid Memory Cube Specification 1.0*, Hybrid Memory Cube Consortium, 2013.
8. B. Black, "Die Stacking Is Happening!" *Proc. Int'l Symp. Microarchitecture,* 2013; www.microarch.org/micro46/files/keynote1.pdf.

Nuwan Jayasena is a principal member of technical staff at AMD Research. Contact him at nuwan.jayasena@amd.com.

Near-Data Processing of Neural Networks

Yunji Chen
State Key Lab of Computer Architecture, Chinese Academy of Sciences

Jinhua Tao
University of Chinese Academy of Sciences

Neural network techniques have become the state of the art over a broad range of applications, including speech recognition, advertisement recommendation, image recognition, and robotics. Yet, recent deep neural network models are notoriously memory intensive, and their efficiency and scalability have been severely restricted by the limited memory bandwidth. Our experiences on the DianNao neural network accelerator revealed that memory accesses can account for more than 95 percent of the energy consumption of neural network processing.[1]

Near-data processing is an effective way of addressing the above issue. To be specific, most neural network models adopted in industrial practices involve less than 1 billion parameters, which requires only 4 Gbytes of storage space if each parameter is represented as a 32-bit number. Therefore, it is realistic to simultaneously keep the whole model within the chips, which eliminates main memory accesses for network models. Following this paradigm, we proposed a machine learning supercomputer architecture called DaDianNao,[2] which includes multiple identical chips connected with a mesh interconnection network. Each DaDianNao chip contains 16 tiles and has a neural functional unit and four embedded DRAM (eDRAM) banks. eDRAM has much higher density than the traditional SRAM, allowing a DaDianNao chip to integrate 36 Mbytes of storage within only 67.7 mm^2 area at a 28-nm technology. All neurons and synapses are saved in on-chip eDRAM banks close to neural functional units, leading to short data access latency and elevated internal bandwidth.

The DaDianNao multichip system is designed for large-scale network models processed at cloud servers. Yet, some application scenarios still have stringent power and area budgets. We studied one such scenario, in which visual recognition is performed with convolutional neural networks (CNNs) at mobiles or other embedded devices. We were motivated by the fact that synaptic weights within CNNs are shared among different neurons, and we designed a hardware accelerator for CNNs, called ShiDianNao,[3] that keeps the whole CNN model within on-chip storage. When ShiDianNao is placed next to the image sensor, it can eliminate all remaining off-chip memory accesses for network models and image data. As a result, ShiDianNao can be more than 30 times faster than a high-end GPU with only a power of 320 mW.

In addition to keeping the entire network model within the computational chip, other alternatives for near-data processing exist. One such example is 3D stacking, which can significantly reduce the access latency to the main memory. However, when 3D stacking meets the neural network, the memory bandwidth problem becomes a critical issue. In a DaDianNao chip, for example, the neural functional units need to extract 65,536 bits per cycle from eDRAM banks. Although it might still be hard for the current on-the-shelf 3D stacking technique to provide such a large bandwidth, future progress in both neural network and 3D stacking architecture may bridge the gap between the computation chip and the memory chip. MICRO

References

1. T. Chen et al., "DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning," *Proc. 19th ACM Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 2014, pp. 269–267.
2. Y. Chen et al., "DaDianNao: A Machine-Learning Supercomputer," *Proc. 47th IEEE/ACM Int'l Symp. Microarchitecture*, 2014, pp. 609–622.
3. Z. Du et al., "ShiDianNao: Shifting Vision Processing Closer to the Sensor," *Proc. 42nd ACM/IEEE Int'l Symp. Computer Architecture*, 2015, pp. 92–104.

Yunji Chen is a full professor in the Institute of Computing Technology at the State Key Lab of Computer Architecture, Chinese Academy of Sciences. Contact him at cyj@ict.ac.cn.

Jinhua Tao is a PhD candidate at the University of Chinese Academy of Sciences. Contact him at taojinhua@ict.ac.cn.

The Active Memory Cube

Ravi Nair
Jaime Moreno
IBM Thomas J. Watson Research Center

Many studies point to the difficulty of scaling computer architectures to meet the demands of future information processing systems. The principal impediments are physical size, communication between computing nodes, and, most importantly, power consumed by such large-scale systems. Performance is also constrained by the limited bandwidth between dense computing chips and standard memory.

The concept of processing in memory has been experimented with for several decades with limited success, largely owing to the relatively slow evolution of standard memories (DRAMs) and limitations in mixing logic and DRAM circuits on the same die. With the maturation of 3D VLSI technology, it is now possible to stack together DRAM and logic dies, interconnected using through-silicon vias (TSVs) on a single package. The Active Memory Cube (AMC) is an exploratory architecture that leverages a commercially demonstrated 3D memory stack, the Hybrid Memory Cube,[1] placing sophisticated computational elements on a logic layer below a stack of DRAM dies. An initial version of the AMC targets scientific supercomputers capable of executing 10^{18} floating-point operations per second (1 exaflop) and consuming just 20 MW.[2]

The AMC presents a memory interface to its host processor; thus, it appears to the host as ordinary memory accessible through standard read and write commands. It also provides the ability to transform the contents of memory through customized interpretation of certain read and write commands. Thus, the AMC extends the capability of existing operations like atomic read-modify-write through new commands that specify arbitrary programs to perform transformations to memory.

The base layer of the AMC was designed to balance simplicity in design to maximize power efficiency with versatility in function to localize computation and minimize communication with the host processor. This tradeoff resulted in an unconventional mix of features, including absence of caches and hardware scheduling of instructions, the use of a vector instruction set, predicated execution, and gather-scatter accesses directly to memory—features that favor efficient execution of many scientific applications. Translation tables in the AMC, loaded by the host, map virtual addresses in user programs to real addresses in the AMC. The AMC's memory is consistently and coherently accessible both from the internal logic and the host processor.

The AMC architecture was closely designed with the compiler and the operating system, guided by applications of interest to the scientific community.[3] Simulation results demonstrate the benefits of processing in memory,[2,4] especially in terms of computational density, memory-bandwidth-to-compute ratio, and power consumed in typical scientific applications.

The size of scientific simulation models keeps increasing. On the commercial side, the amount of data faced by analytics and prediction problems keeps increasing at an extraordinary pace. Processing in memory, exemplified by the AMC and its variants, offers a promising approach to overcome the challenges posed not only by future large-scale simulation applications, but also by future large-data commercial applications. MICRO

References

1. J.T. Pawlowski, "Hybrid Memory Cube: Breakthrough DRAM Performance with a Fundamentally Rearchitected DRAM Subsystem," Hot Chips 23, 2011.
2. R. Nair et al., "Active Memory Cube: A Processing-in-Memory Architecture for Exascale Systems," *IBM J. Research and Development*, vol. 59, no. 2/3, 2015, pp. 17:1–17:14.
3. A. Jacob et al., "Progressive Codesign of an Architecture and Compiler Using a Proxy Application," *Proc. 27th Int'l Symp. Computer Architecture and High-Performance Computing*, 2015, pp. 57–64.
4. P.F. Baumeister et al., "Accelerating LBM and LQCD Application Kernels by In-Memory Processing," *High Performance Computing*, LNCS 9137, 2015, pp. 96–112.

Ravi Nair is a distinguished research staff member at the IBM Thomas J. Watson Research Center. Contact him at nair@us.ibm.com.

Jaime Moreno is the Worldwide Technology and Operations Manager, Computing as a Service, at the IBM Thomas J. Watson Research Center. Contact him at jhmoreno@us.ibm.com.

In-Situ Computing Through an Analog Dot-Product Engine

Naveen Muralimanohar
Hewlett Packard Labs

Recent trends—such as the growing prominence of data-centric workloads, the end of Dennard scaling, and the slowing down of Moore's law—have triggered significant interest in accelerators, especially with a focus on reducing communication overhead between computing and memory. Although any accelerator has the potential to improve efficiency, to achieve sustainable performance scaling for several years, it is critical to investigate new scalable hardware primitives and integrate them into traditional systems.

Near-data processing is an effective technique to improve computing efficiency of data-intensive workloads. An ultimate version of near-data processing is a memory block that can exploit the very process of accessing memory to perform complex computation in situ. A memristive dot-product engine (DPE) is one such device that extends a memory array's functionality to perform matrix-vector multiplication (MVM). A DPE exploits the fundamental relationship between the row select voltage and column current in an array to perform analog multiplication. To perform a MVM, the input matrix is first programmed into a DPE array. Each input vector element is applied as the corresponding row voltage in parallel. Finally, the resulting bitline current in each column is digitized to get the output vector. A hardware primitive like this offers several advantages. First, DPE is highly parallel and fast—the entire operation completes in one cycle. Second, because the computation happens in the analog domain, its energy overhead is low. Finally, a DPE has many degrees of freedom to scale, including analog-to-digital converter (ADC) speed and precision at each column, digital-to-analog converter (DAC) precision at each row, cell density, and array size. Thus, sustainable improvement in performance or efficiency for many years is possible.

As with any new technology, DPE must overcome challenges to be viable. On the technology front, because cell resistances are used directly to perform computation, each cell should be programmed to the highest precision possible. Also, DPE should guarantee certain precision even in the presence of noise and process variation. Fortunately, DPE arrays are not limited by a low-cost cell structure or having to perform fast read or write operations, making it feasible to get a functional DPE. On the architecture front, we need to deal with the limited precision provided by a DPE's analog computation but still guarantee correctness. Some applications, such as neural networks, are inherently error tolerant and can use high-density DPEs directly, whereas others require architectural enhancements to ensure correctness. Another critical challenge is that even the best-engineered DPE can handle elements of only a few bits. Accelerating real-world problems involving floating points or integers requires a scalable architecture that can operate on thousands of DPEs with nontraditional control and datapaths.

Hewlett-Packard Labs has demonstrated a functional DPE supporting 32 levels (5 bits) with 1 percent tolerance. Our preliminary analysis shows that DPE-based accelerators can deliver speedups of up to 10 to 25 times for specific classes of scientific and neural network workloads, making DPEs a compelling candidate for future accelerator architectures. MICRO

Naveen Muralimanohar is a principal research scientist at Hewlett Packard Labs. Contact him at naveen.muralimanohar@hpe.com.

Memory Processing Units

KARTHIKEYAN SANKARALINGAM
University of Wisconsin–Madison

CRISTIAN ESTAN
Google

In this position statement, we focus on near-data processing (NDP) for DRAM-scale problems. To be practically viable, NDP architectures should be applicable in a widespread way across many domains, be easily programmable (avoiding esoteric programming models), coexist in the single unified memory address space of CPUs, and provide integer factors improvements in performance, energy, or both. From a business and technical standpoint, the architecture must provide big benefits today while including principles allowing its competitive edge for a decade or more.

However, many current approaches suffer from narrow specialization targeted to specific applications, curtailing their widespread use. The fundamental challenge for NDP systems is workload heterogeneity: algorithms from different domains present various memory layouts and access patterns and involve computations with different degrees of parallelism and complexity. Even within a single workload, regions that are essentially sequential, which can most efficiently be executed on a conventional CPU, can alternate with regions with abundant thread-level parallelism and limited required synchronization, which are ideal targets for NDP engines. Because of this variety, traditional wisdom is that NDP systems can achieve performance and energy improvements only by specializing the hardware to specific classes of workloads. Such narrow specialization makes these designs obsoletion-prone by definition, limits their usage, and creates a challenging business proposition. Thus, it is an open question whether it is possible to build a general NDP architecture that can bring integer-factors energy and performance benefits while supporting a wide variety of workloads.

The MPU (memory processing unit) project answers this question in the affirmative by developing an NDP architecture driven by the following fundamental principles implemented in a 3D-stacked memory.

- *Performance through massive concurrency:* the MPU architecture includes large arrays of cores, allowing thousands of concurrent threads.
- *Energy savings through low-energy computation:* the MPU architecture uses low-power cores that remain efficient when idling (unused or waiting for memory I/O).
- *Generality through a flexible programming model:* the MPU uses a general approach to NDP offload that is based on low-overhead remote procedure calls.

The first two principles achieve concurrency and efficiency, while the third achieves generality. The overheads of a simple, low-power core are small enough in power and area that a brute-force approach consisting of large arrays of low-power cores suffices to achieve the NDP goals of accelerating computation while reducing energy. Specifically, this brute-force approach is comparable or superior to building datapath modules integrated to DRAM, targeted prefetch engines, and so on. To be amenable for this architecture, the only workload property necessary is for them to be *shardable*—that is, partitioned in many independent tasks that access separate memory regions. Increasingly, programmers are designing big-data problems this way from the ground up. Simulations show that, compared to state-of-the-art multicore CPUs, MPUs can deliver 7 to 33 times speedups, resulting in 8 to 19 times lower energy across database processing, graph analytics, text analytics, and Internet search.

We are building a prototype single-cube memory chip with cores implemented on a tightly integrated field-programmable gate array and two end-to-end real-world applications—Internet Search (retrieval and scoring) and database processing. We foresee MPUs and their principles enabling a first-class programmable engine for memory-intensive workloads. MICRO

Karthikeyan Sankaralingam is an associate professor at the University of Wisconsin–Madison. Contact him at karu@cs.wisc.edu.

Cristian Estan is a software engineer in search infrastructure at Google. Contact him at estan@google.com.