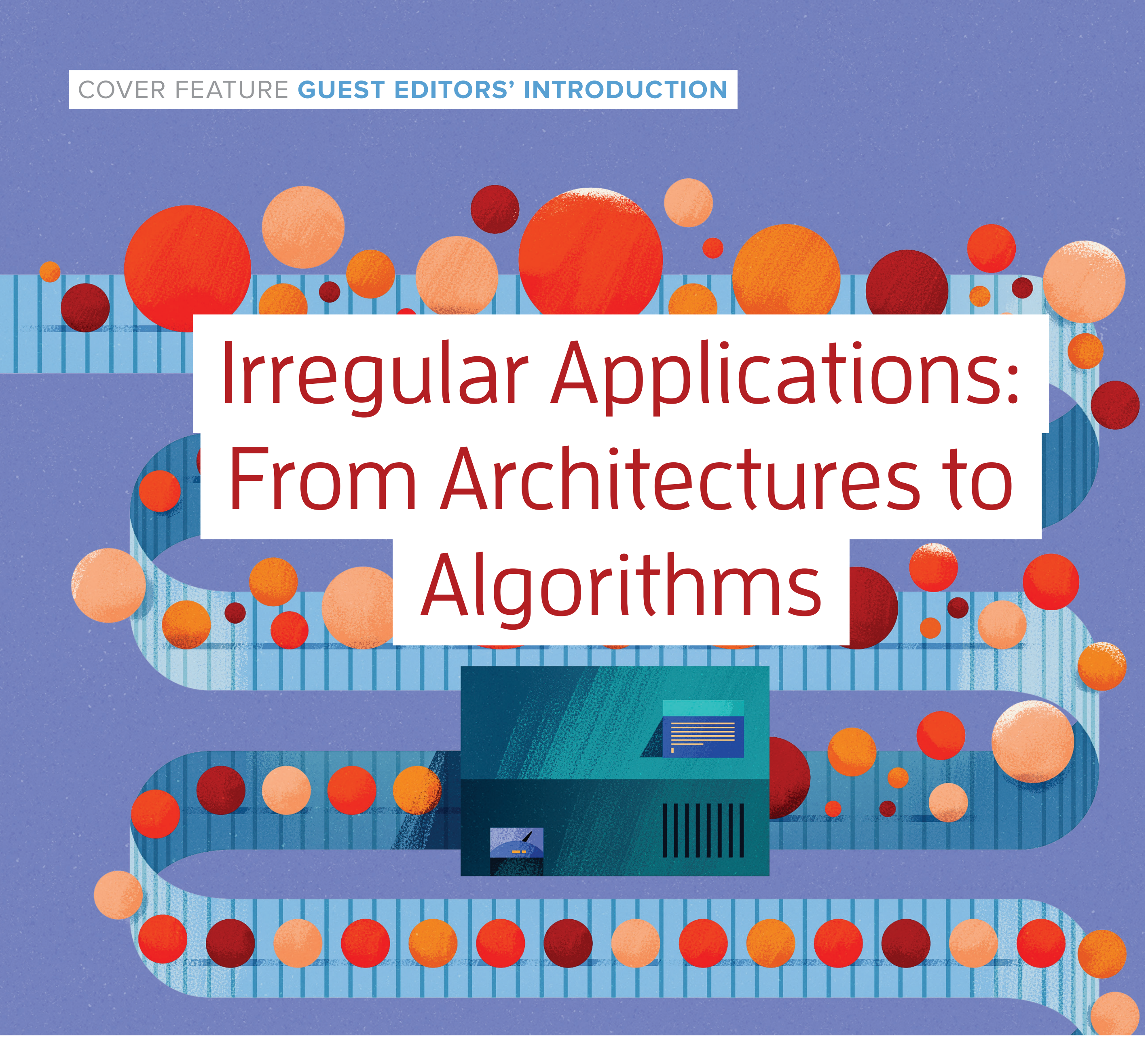
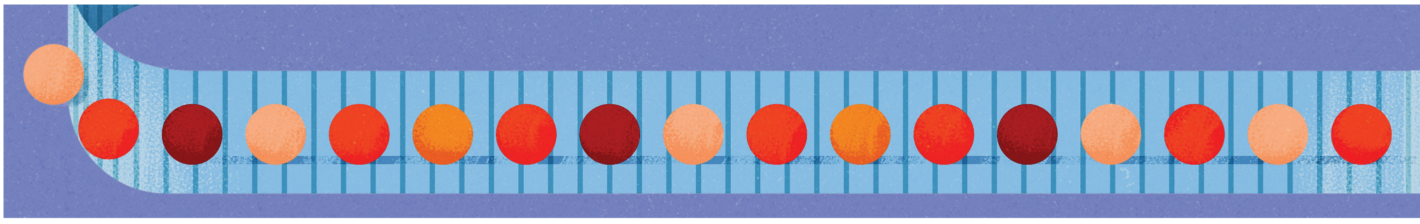


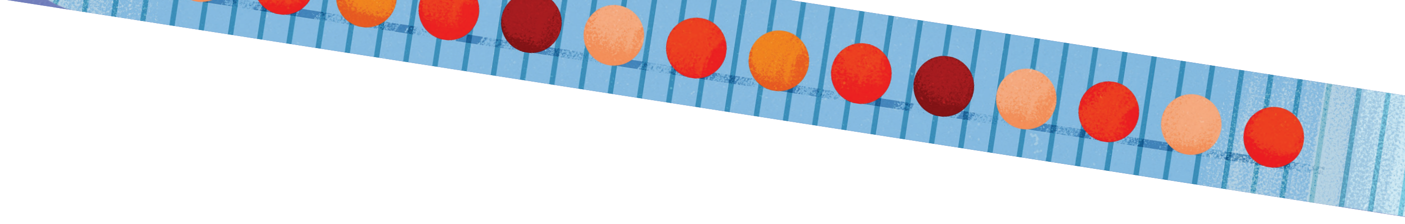
Irregular Applications: From Architectures to Algorithms

Antonino Tumeo, Pacific Northwest National Laboratory

John Feo, Context Relevant

Irregular applications present unpredictable memory-access patterns, data-dependent control flow, and fine-grained data transfers. Only a holistic view spanning all layers of the hardware and software stack can provide effective solutions to address these challenges.





In recent decades, high-performance computing (HPC) systems have delivered significant breakthroughs for numerous scientific applications, allowing the simulation of physical and chemical phenomena at an unprecedented and unthinkable scale. But now, the easy availability of data—the big data challenge—has resulted in a surge of computing for data analytics. Exponentially growing datasets collected from a variety of sources impact all major industries: finance, government, healthcare, security, transportation, manufacturing, commerce and e-commerce, and communications—including social networks and the Web.

Achieving actionable results from this avalanche of data requires HPC systems to be optimized for deduction, machine learning, and graph algorithms. Aside from the size-related challenges, collected data is often unstructured and sparse, and not all records have the same types or features. As new data records are continuously generated, new relationships emerge and disappear in real time. The action of system agents, which explore and find relationships in the data records, will evolve over time in response to the analysts' actionable decisions, requiring the evolution of models and measures. Only highly dynamic, multidimensional data structures—such as graphs, trees, and grids—are capable of organizing the collected data in a supportive manner.

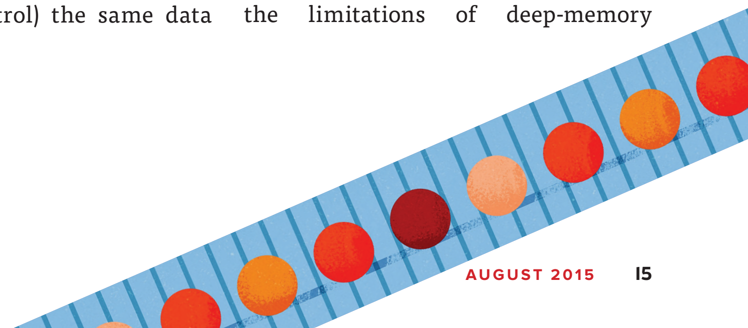
Unfortunately, these workloads have characteristics that are data dependent and defy reasoning prior to execution. These workloads' requirements contrast with the regular behaviors of scientific simulations, therefore new, cross-cutting approaches involving architectures, system software, and algorithms are required to address them.

Pertinent data structures are pointer- or linked list–based, resulting in unpredictable data accesses. A pointer in memory representing, for example, an edge of a graph or a relation between two values, might lead to data accesses unrelated in any temporal or spatial sense. Such lack of locality limits the performance of modern processor architectures built on deep memory hierarchies and prefetching to mitigate latencies associated with the memory wall problem—that is, a limited increase in bandwidth despite a relative increase in computational power. Moreover, irregular data structure–based applications usually have limited arithmetic intensity, as most of the computation time is spent navigating the data structures. Thus, modern processors and accelerators designed to maximize floating-point operations per memory access are a poor fit.

Data analytics applications have large amounts of data parallelism as many data values can be explored in parallel. This inherent parallelism can map well onto modern processors as emerging limitations in frequency scaling and instruction-level parallelism results in ever-increasing numbers of cores and hardware threads. However, many new architectures are improving performance though vector units and single-instruction, multiple-data (SIMD) paradigms, thereby trading flexibility in control for data parallelism. Unfortunately, the control flow in data analytics applications is typically data dependent and often requires fine-grained synchronization as concurrent threads compete to access (or control) the same data elements or spawn dynamic numbers of new threads.

The nonuniform relationships among data items make partitioning difficult, resulting in irregular communication patterns and significant communication overhead. In fact, a dataset's size might preclude it from fitting into the memory available on a single node of an HPC system—even in cases in which a node can host more than a terabyte of RAM. However, current network interconnects perform well only with large, batched data transfers; they incur large overhead when transferring fine-grained messages, synchronization events, and even the all-to-all communications required by many analytics applications.

Current HPC machines are optimized for scientific simulations, arithmetic-intensive workloads, and regular computation, whereas data-analytics applications require hardware optimized for bandwidth, throughput, concurrency, and all-to-all communication. In addition, all layers of the system stack must help minimize development and deployment costs across the diverse set of emerging next-generation systems. For example, algorithm developers should be able to retain a simple way of expressing and extracting the latent parallelism of their applications without having to partition data by hand and dynamically load balance. Domain-specific languages and libraries can also substantially reduce programming costs, and compilers can provide architecture-specific optimizations. Highly dynamic runtime system approaches can alleviate load-balancing issues and mitigate—through software techniques transparent to the application developer—the limitations of deep-memory



ABOUT THE AUTHORS

ANTONINO TUMEO is a senior research scientist in the High Performance Computing Group at the Pacific Northwest National Laboratory. His research interests include simulation and modeling of high-performance and embedded computer architectures, hardware/software codesign, field-programmable gate array (FPGA) prototyping, and GPGPU computing. Tumeo received a PhD in computer science and engineering from Politecnico di Milano, Italy. He is a member of IEEE and ACM. Contact him at antonino.tumeo@pnnl.gov.

JOHN FEO is vice president of engineering at Context Relevant, a predictive data analytics company based in Seattle, Washington. His research interests include parallel computing, parallel application development, graph databases, functional languages, and performance studies. Feo received a PhD in computer science from the University of Texas at Austin. He is a member of ACM. Contact him at marwick04@aol.com.

hierarchies and large-packet communication networks. Approaches exploiting map-reduce methods or adapting conventional software stacks are also demonstrating some success; however, the necessity to adapt to a specific computational paradigm that might not suit the target problem limits these solutions' effectiveness.

IN THIS ISSUE

To examine possible strategies to address the challenges of irregular applications, the theme of this month's issue, we take a holistic point of view and provide a forward-looking perspective of what is possible. The four articles herein analyze and propose solutions for irregular behaviors at different layers of the stack.

"In-Memory Data Rearrangement for Irregular, Data-Intensive Computing," by Scott Lloyd and Maya Gokhale, examines irregular behaviors from a system's lowest layer and proposes a hardware design that dynamically restructures in-memory data in a cache-friendly layout. The method offers both speedup and energy benefits, and the authors present a prototype implementation and validate it on a reconfigurable device—a field-programmable gate array—using the Hybrid Memory Cube memory model.

In "Optimizing Sparse Linear Algebra for Large-Scale Graph Analytics," Daniele Buono, John A. Gunnels, Xinyu Que, Fabio Checconi, Fabrizio Petrini, Tai-Ching Tuan, and Chris Long examine architecture-specific optimizations of sparse linear algebra operations on modern HPC systems based on the IBM Power8 processor. Although much recent work has explored the mapping between graphs and linear algebra, sparse matrices remain the prototypical irregular data structure. In particular, this article delves into optimizations of sparse matrix-vector multiplication (SpMV) and sparse generalized matrix-matrix multiplication (SpGEMM), trading off higher data accesses for better memory and cache locality.

In "Scaling Runtimes for Irregular Algorithms to Large-Scale NUMA Systems," Andrew Lenharth and Keshav Pingali describe the Galois system, which provides a carefully restricted sequential programming model that can be implemented in any object-oriented language (the authors use C++) and enables automatic parallelization of irregular algorithms. The authors demonstrate that the Galois runtime system can execute irregular applications efficiently on nonuniform memory access (NUMA) shared-memory machines such as the 512-core SGI UV used in their experiments.

Finally, Mahantesh Halappanavar, Alex Pothen, Ariful Azad, Frederik Manne, Johannes Langguth, and Arif Khan address important algorithmic issues in "Codesign Lessons Learned from Implementing Graph Matching on Multithreaded Architectures." The authors explore the interplay between algorithm design and architectural features using graph matching, a fundamental problem in many data-intensive applications that employ graph-based data structures.

These four articles exemplify a variety of approaches for making future HPC systems more amenable to data-analytics workloads in the era of big data. Breakthroughs are only possible if we take a holistic view of the entire stack and codesign the hardware and software layers. Only through this kind of intense collaboration among data scientists, software engineers, and computer architects will we achieve platform design capable of surmounting the challenges in performance and data size scalability brought by irregular behaviors.

We enjoyed preparing this special issue on a critical topic in the new age of computing, and hope you will join the conversation at www.linkedin.com/grp/post/52513-6027558200293810180.