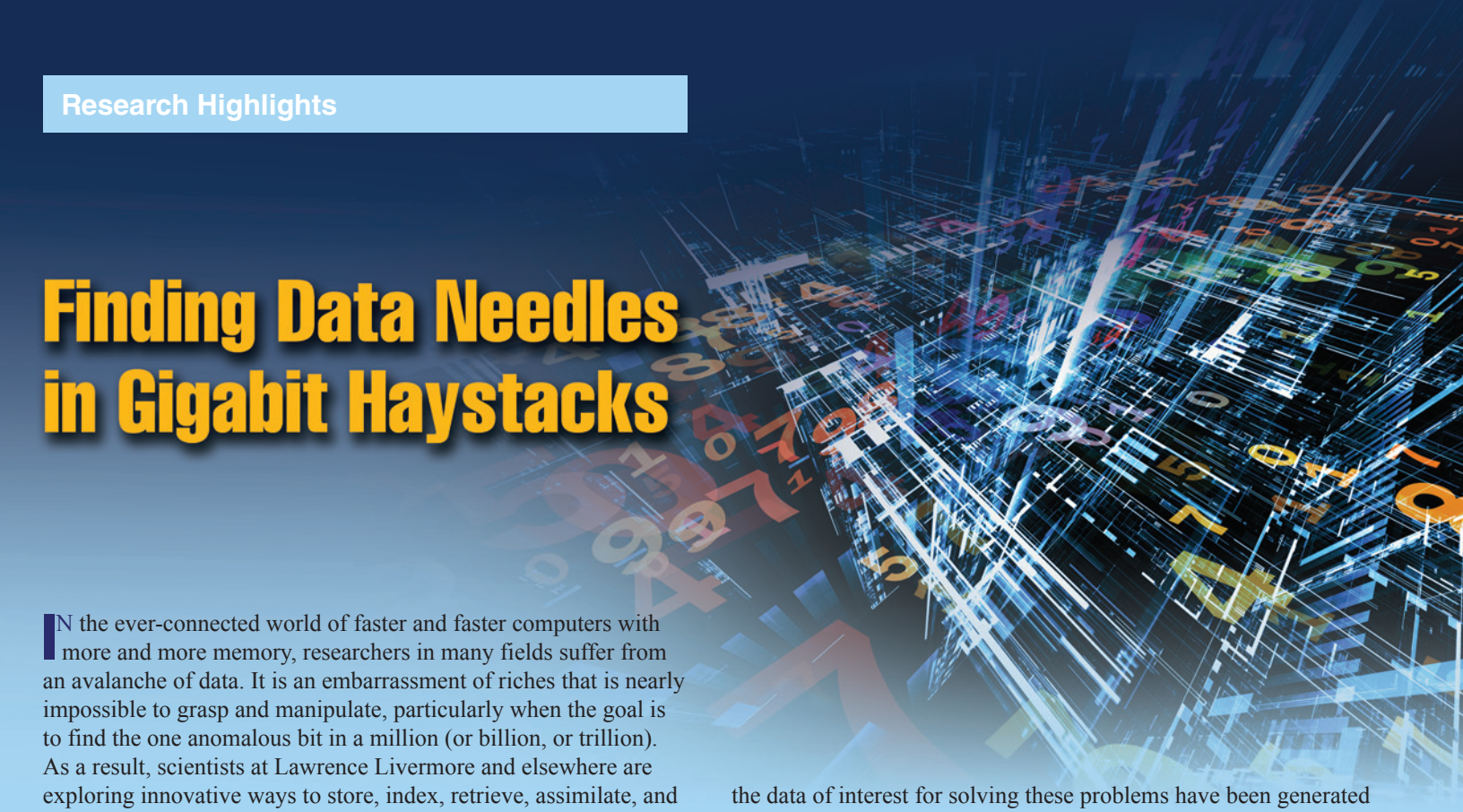
Finding Data Needles in Gigabit Haystacks



IN the ever-connected world of faster and faster computers with more and more memory, researchers in many fields suffer from an avalanche of data. It is an embarrassment of riches that is nearly impossible to grasp and manipulate, particularly when the goal is to find the one anomalous bit in a million (or billion, or trillion). As a result, scientists at Lawrence Livermore and elsewhere are exploring innovative ways to store, index, retrieve, assimilate, and synthesize mountains of raw data into useful information.

One way that computer scientists are tackling this challenge is by developing and optimizing algorithms and architectures that interact closely with large volumes of data. These data-intensive computing approaches combine techniques from computer science, statistics, and applied mathematics to speed up data manipulation in fields as diverse as astrophysics, bioinformatics, and social networks.

Not the Usual Data-Crunching

Computer scientist Maya Gokhale leads a team in the Laboratory's Center for Applied Scientific Computing that is creating computer architectures to address this "data overload" problem. "Not only is the amount of data being generated growing exponentially," explains Gokhale, "but when the raw data are analyzed, more data—called metadata—are generated as well. It's truly an issue of 'drowning in data.'"

The solutions evolving to address these problems are vastly different from those developed to manage the data generated by large number-crunching physics simulations. Many physics problems, such as modeling a solid piece of metal as a shock wave moves through it, can be characterized as three-dimensional (3D) mesh problems. Computationally, these problems are modeled as "cells" in a 3D space. In this way, each cell's behavior is influenced only by its nearest neighboring cells, where shared "edges" exist.

Physics computations run efficiently on huge supercomputers, such as Livermore's BlueGene/P, because those machines have many compute nodes, each of which can map to a spatially contiguous collection of cells. These systems have a favorable ratio of computation to communication; that is, each node can perform a lot of computing before it needs to communicate with other nodes. Thus, relatively little memory is required on each node. In general, the data of interest for solving these problems have been generated and are being used within the computer system.

However, not all problems map well to the physics-oriented computer architecture. For some, the data to be analyzed are stored externally, on hard disks and other storage devices. These data might be millions of star images gathered from telescopes all over the world, detailed tables of genomic data, or data on social networks. (See the figure on p. 24.)

With social networks, for example, each person would be characterized as a "cell" or node, and the connection between one person and another would be an edge. The number of edges between nodes can vary immensely: A person might have a connection to only one other person on the network or connections to tens, hundreds, even thousands of people. "Methods have been developed to access such data for analysis," says Gokhale, "but when speed is of the essence, we want faster, more convenient ways to access and analyze data. One solution is to have an architecture in which the database exists 'closer' to where the work is done."

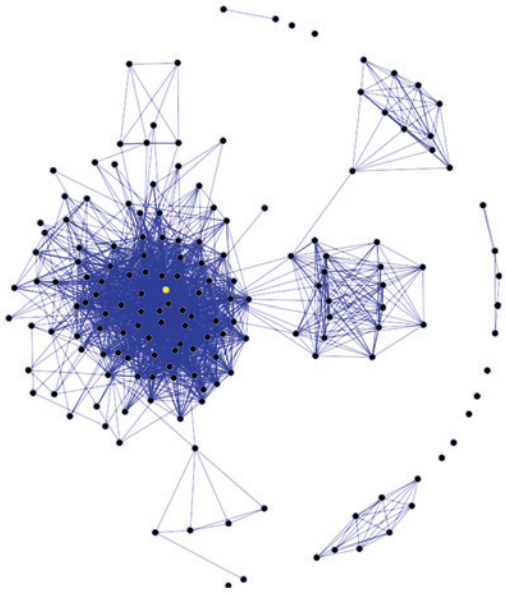
Bringing Data Closer

To meet the data-intensive computing requirements, Gokhale and her team are working on an innovative hardware technology project that is funded by the Laboratory Directed Research and Development Program. The technology, called "persistent" memory, incorporates large, parallel arrays of solid-state storage devices within the compute node. Persistent memory is embodied as flash memory, for example, in a USB memory stick.

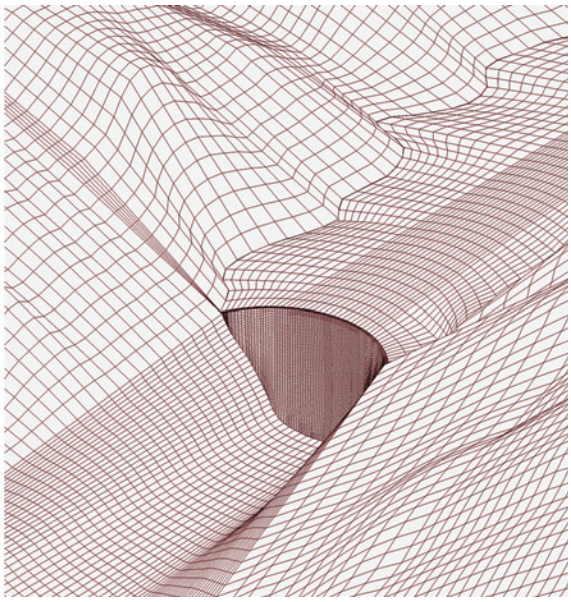
Options for storing data range from permanent to transient memory. Permanent memory can be stored on devices such as hard disk drives and flash drives (or memory sticks) outside the computer. Transient memory, such as dynamic random access memory (DRAM) and central-processing-unit (CPU) cache, exists

(a) A social network graph displays the complex interconnections (purple lines) between different people, each represented by a node (black dots). Most people have many connections that often overlap, but some have only one or even zero. (b) Physics data-crunching solutions use three-dimensional meshes with cells and edges, such as this simulation of the Morrow Point Dam region in southwest Colorado. The two types of computational problems require different approaches for data storage.

(a)



(b)



within a computer. DRAM consists of capacitors that each store a bit of data within integrated circuits in the computer. CPU cache holds copies of data from the computer's main memory that are frequently used by the processor.

Permanent and transient memories each have their pluses and minuses. Pull the plug from the wall, and permanent memory data remain, but transient memory data vanish. On the other hand, access to data on permanent memory is slow: a factor of 100,000 slower than data stored on DRAM. Persistent memory embraces the best of both permanent and transient memory. Data stored in persistent memory are both permanent and close to the compute node, allowing for fast access and manipulation.

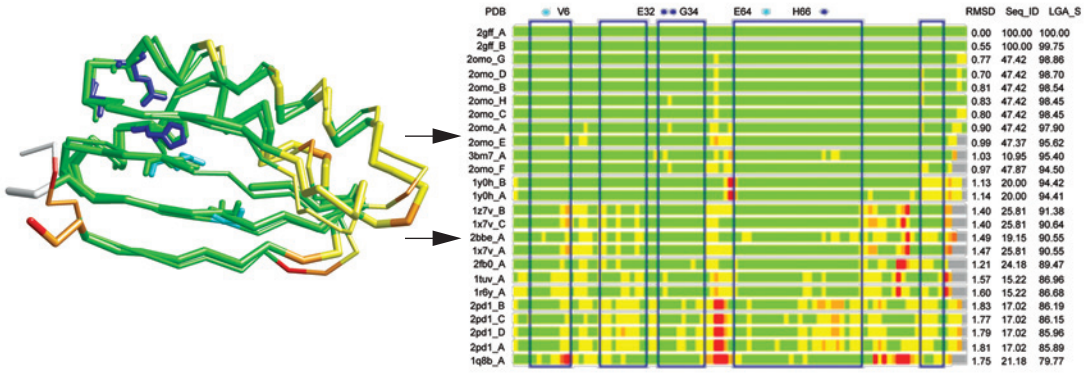
"Plentiful, inexpensive persistent memory in the form of flash storage array technology makes it possible to create very large databases that can be accessed later for searches," says Gokhale. "However, we still need to address the research challenges in organizing and accessing such databases in flash memory arrays, which have, at best, 1,000 times the access latency of main memory." Latency, or delay, is defined as the time required for a data packet to travel from one point to another, or in this case, from memory to a compute node. Gokhale and colleagues have made inroads on the latency problem by developing a highly multithreaded parallel algorithm for flash storage arrays that allows flash memory to outperform a serial algorithm in plentiful main memory by a factor of four.

From Galaxies to Global Security to Genomes

One area that will benefit from the team's novel approach is astrophysics, including the Large Synoptic Survey Telescope (LSST), a project for which the Laboratory is a contributing member. When completed in 2013, the telescope will have the world's largest digital camera to survey the entire visible sky. Researchers will use the resulting images to study dark matter through its light-bending gravitational effect in an effort to chart the expansion history of the universe and probe the nature of dark energy.

LSST will generate 30 terabytes of data every night, yielding a total database of 100 petabytes. "They will do triage on the data right away," says Gokhale. "A software pipeline will look for starlike objects, compare them to a template, and store likely candidates for immediate consideration. The raw data will be saved and stored. One challenge is how to later retrieve and examine the raw data for a very specific item; that is, identify an anomaly in that enormous database. Our persistent-memory architecture will speed up such tasks."

Global security is another arena that will benefit from the team's inventive solution for data-intensive computing. Scott Kohn of the Information Operations and Analytics Program is enthusiastic about the possible application to cybersecurity efforts. A primary concern in cybersecurity is getting a big picture view, or situational awareness, of how machines are communicating with each other in a network. These communications are modeled as a graph, but the graph is so large that to store and analyze it typically requires the memory resources of a small supercomputer. "Maya's work is exciting," says Kohn. "It may allow us to analyze these massive communication graphs on a relatively inexpensive workstation instead of spending tens of millions of dollars on a custom supercomputer."



Work in data-intensive computing solutions such as persistent-memory technology will benefit a multitude of research areas from astrophysics to bioinformatics. For bioinformatics problems such as the one shown, access to very large, low-latency memory on a single node can provide orders of magnitude improvements compared with calculations that use distributed memory on a computing cluster optimized for physics codes.

The team's work can also be applied to bioinformatics. Tom Slezak, associate program leader for Informatics, explains that bioinformatics has a class of problems in which large amounts of DNA sequence data are analyzed, creating an efficient data-indexing structure called a hash table. Researchers need to easily and quickly exploit the data in the large sequence hash table. Slezak says, "Although there appears to be an easy way to break up the table and distribute the data across many compute nodes, communication latency makes this approach too slow to be practical." Latency comes into play when a given node requests part of a hash table that is stored on another compute node. The seek-request-and-fetch operation is then at least 2 to 3 orders of magnitude slower than it would be if the entire hash table were in local memory for any compute node accessing it—a difference that translates into a job running in 1 day versus 100 or 1,000 days.

With persistent memory, the huge hash tables can be stored completely and accessed with a very low latency compared to going across the "grid" to another computer node. "Persistent memory will enable us to attempt bioinformatics computations that simply are not feasible with other architectures," says Slezak.

Among the bioinformatics problems that will benefit from this approach are those related to rapid and thorough analysis of complex (metagenomic) sequence data. These problems involve billions of DNA "short reads" (currently between 36 and 110+ bases in length). "Although portions of this problem can be mapped to multiple compute nodes, the need to access the enormous data structure argues for a single multicore system," says Slezak. Large persistent memory will likely outperform any other architecture currently feasible."

Persistent Memory Graphs a Winner

The viability of this novel computer architecture was proven in the June 2011 international Graph 500 competition. Two entries from Gokhale and Roger Pearce, a Lawrence Scholar working under her direction, ranked at 7 and 17. Graph 500 gets its name from graph-type problems—algorithms that are a core part of many analytics workloads in applications, such as those for cybersecurity, medical informatics, and social networks. Rankings indicate which computer and algorithm combination solved the largest instance of the problem and had the fastest time to solution for a particular problem size. A machine on the top of this list can quickly and efficiently analyze huge quantities of data to find the proverbial needle in the haystack.

The Graph 500 benchmark calculations were run on Livermore's Kraken, a large memory server with 32 cores, 512 gigabytes of DRAM, and 2 terabytes of direct-attached flash memory. The approach, which used a highly multithreaded, shared-memory algorithm, was unique among the competitors in achieving high performance on a single compute node with very large memory. "The graph problem is part of our research that seeks to enlarge the memory available to a compute node by augmenting DRAM with high-performance, direct-attached flash arrays," says Gokhale. "This configuration enables higher utilization of the cores by giving each core more aggregate memory, a combination of DRAM and flash. It also reduces the energy required by the node, trading power-hungry DRAM for flash."

The team has since developed a multinode version of the algorithm and tested it in distributed memory runs on the Hyperion Data Intensive Testbed at Livermore and the Trestles machine at the San Diego Supercomputing Center. As for what's next, says Gokhale, "We are working on additional flash-based, multithreaded graph-analysis algorithms including Google's page-rank search algorithm and connected components identification, which finds clusters of nodes in a graph that indicate close relationships."

—Ann Parker

Key Words: bioinformatics, computer algorithm, computer architecture, database, data-intensive computing, flash memory, Large Synoptic Survey Telescope (LSST), persistent memory.

For further information contact Maya Gokhale (925) 422-9864 (gokhale2@llnl.gov).