Government, academic, and industrial sources are generating data at a much greater pace, volume, and heterogeneity than ever. Data grows over time, and better collection and storage technologies will soon let individuals command terabytes of information even as organizations move beyond petabytes.

In many disciplines, data is shared and distributed among a broad community that represents many different interests, roles, and viewpoints. This is especially true in scientific application areas such as DNA computing, biocomputing, medical computing, environmental sciences, space sciences, physics, and astronomy. In some areas, such as archaeology field work, a detailed record of an excavation is the main artifact that remains after the scientific exploration destroys the original site. The experimental and experiential nature of many sciences requires that researchers capture huge volumes of data for which they propose theories to explain or model the data. Simulations, queries, and data mining can generate very large derived data sets.

The Internet and associated technologies (such as the Web and the Grid) have become the de facto medium for sharing raw data, analyzed information, and derived knowledge for many technical and scientific communities. Considerable research and commercial activity has focused on this area, but many unresolved issues remain.

Key Challenges

Scientific computing is scaling up in volume, timescales, heterogeneity, and accessibility. To solve a scientific puzzle, researchers might gather some data in space or from nanomotes (smart dust) and combine it with other data collected in 1956, for example. Pervasive storage will accompany the unfolding world of pervasive computing and distributed sensor networks. If scientific data is stored in read-only databases and cannot be shared, aggregated, and analyzed, its value is diminished. The volume and heterogeneity of data we can collect will dominate problems and limit our ability to mine for results.

If we’re not careful, the special processing requirements of scientific prob-
lems could lead to a wide variety of stovepiped scientific-system and data architectures. A single problem might force us to manage mixed collections of data models, including files, relational databases, object databases, grids, distributed-agent, and network-storage schemes. Because managed access to data is often critical, scientific data will require reliable storage, federated access control, digital rights, privacy controls, and data dissemination schemes. Different scientific problems will have different processing requirements — simulation, workflows, data quality, pedigree and provenance issues, change-management and temporal data issues, collaboration models, self-organizing, and emergent and autonomic organizations. Traditional data-processing approaches share these problems, but scientific computing tends toward extremes in size and scale.

In “The Collaborative Information Portal and NASA’s Mars Rover Mission,” Mak and Walton describe the Web portal developed to support NASA’s Mars Exploration Rovers. Two robotic proxy geologists, Spirit and Opportunity, use multiple cameras and an array of scientific instruments to collect volumes of information. On earth, mission managers, engineers, and scientists analyze downloaded data and images and plan next steps. NASA’s Collaborative Information Portal uses a three-tier service-oriented architecture (SOA) to handle a range of challenges including continent-spanning users, significant security issues, users with different access rights, different operational schedules for the two rovers, event planning, data navigation, and the need to track things on Mars time. The portal also must deal with middleware services for querying metadata and schedules and for asynchronous notification and broadcast messages.

In “Active Management of Scientific Data with myLEAD,” Plale and colleagues argue that the meteorologist’s workbench is unwieldy without proper computational tools. To address the problem, the authors’ group at Indiana University developed the myLEAD personalized information-management tool as part of the Linked Environments for Atmospheric Discovery (LEAD) project. Like the Collaborative Information Portal, myLEAD provides end users with a Web-based portlet interface. It too has an SOA that provides grid services for access control, concurrent access to a metadata catalog (which extends the Globus Metadata Catalog), a metadata query service, a replication service, workflow services, and data mining services.

In “A Grid Service Module for Natural-Resource Managers,” Wang and colleagues describe an environment that supports researchers and resource-management professionals in analyzing ecosystem models. Their system enables users to access grid computing facilities without extensive training or experience with middleware architectures. The service implementation is built around grid middleware products and provides a single Web interface through which users can access, run, and retrieve data from multiple ecological models. It analyzes service availability and machine workload to balance workloads across distributed servers within the simulation moderator. This article is representative of many scientific disciplines that require large simulations or workflows but also workload balancing and the ability to segment huge files into partitions.
Given scientific data’s breadth, volume, and variety, it might be surprising that any software architecture could suit so many needs and evolving requirements. As the articles in this issue illustrate, however, Internet-based service architectures are emerging to meet the challenges.

Service architectures have evolved rapidly since the early 1990s, when the Object Management Group defined the “Object Management Architecture Reference Guide,”¹ which included the Common Object Request Broker Architecture (Corba) and an extensible collection of object services. OMG defined an Interface Description Language to describe interfaces to Corba and various OMG services. In contrast to these language-neutral standards, Sun defined a second-generation service architecture in Java with Remote Method Invocation (RMI), serialization, and various services. In the past several years, the Web services and grid communities have adopted XML-based interfaces including SOAP, WSDL, and UDDI, and have gone on to define various generic Web services. It is common to see all of these service architectures in use today.

As the articles in this theme argue, service-oriented frameworks help with designing and implementing data-intensive scientific applications. XML makes a good, general-purpose language because it can define Web service interfaces, data formats, and extensible collections of properties. Standardized XML-based distributed computing frameworks provide language neutrality. Increasingly standard plug-in services provide common facilities for messaging, security, reliability, change management, workflow management, job schedulers, capacity planners, simulators, and metadata management.

Who benefits from service architectures and how? In the ideal case, object or Web developers can develop a wide variety of services, each in relative isolation, and the plug-in framework can be used to provide the glue to fit the pieces together. With modular extensible system designs, applications — including those for scientific computing — can depend on just the services they need, evolving to add additional services later. End users benefit from a common semantics resulting from familiarity with a family of applications constructed from a common collection of services. This is similar to and generalizes the common look-and-feel that consolidated the user interface community around pull-down menus. The scientific end user doesn’t have to understand the computational infrastructure but rather can focus on using the computational tools it glues together.

Given that service architectures are still in their infancy, it’s not yet easy to snap together a collection of services from a toolkit to build the kinds of applications surveyed here. The three applications described in this issue focus on different collections of services, with some overlapping and some unique functionality. Yet, it seems likely that the three user populations would benefit even more if all the generic service functionalities in these applications were available to each of them. The fact that we can see a similar architecture emerging, and that it is common across several scientific computing problems, gives rise to hope that Internet-based middleware SOAs can provide a way to cope with increasingly large and heterogeneous scientific data sets.

Internet-based service architectures are emerging to meet the key challenges of scientific computing.

Reference

David Korsmeyer directs the Computational Sciences Division at NASA Ames Research Center. His research focuses on AI and intelligent robotics research; his division supported the deployment of NASA’s first AI system in space. Korsmeyer received an MS and a PhD in aerospace engineering from the University of Texas, Austin, and an MS in management as a Sloan Fellow at the Stanford University Graduate School of Business. Contact him at david.j.korsmeyer@nasa.gov.

Craig W. Thompson is a professor and Acxiom Database Chair in Engineering at the University of Arkansas and president of Object Services and Consulting. His research interests include software architectures, middleware, data grids, and agent technologies. Thompson received a PhD in computer science from the University of Texas, Austin. He is a senior member of the IEEE. Contact him at cwt@uark.edu.