



Parallelism and the Cloud

DENNIS GANNON

DAN REED

Microsoft Research

OVER THE PAST DECADE, scientific and engineering research via computing has emerged as the third pillar of the scientific process, complementing theory and experiment. Several national studies have highlighted the importance of computational science as a critical enabler of scientific discovery and national competitiveness in the physical and biological sciences, medicine and healthcare, and design and manufacturing [1-3].

As the term suggests, computational science has historically focused on computation: the creation and execution of mathematical models of natural and artificial processes. Driven by opportunity and necessity, computational science is expanding to encompass both computing and data analysis. Today, a rising tsunami of data threatens to overwhelm us, consuming our attention by its very volume and diversity. Driven by inexpensive, seemingly ubiquitous sensors, broadband networks, and high-capacity storage systems, the tsunami encompasses data from sensors that monitor our planet from deep in the ocean, from land instruments, and from space-based imaging systems. It also includes environmental measurements and healthcare data that quantify biological processes and the effects of surrounding conditions. Simply put, we are moving from data paucity to a data plethora, which is leading to a relative poverty of human attention to any individual datum



and is necessitating machine-assisted winnowing.

This ready availability of diverse data is shifting scientific approaches from the traditional, hypothesis-driven scientific method to science based on exploration. Researchers no longer simply ask, “What experiment could I construct to test this hypothesis?” Increasingly, they ask, “What correlations can I glean from extant data?” More tellingly, one wishes to ask, “What insights could I glean if I could fuse data from multiple disciplines and domains?” The challenge is analyzing many petabytes of data on a time scale that is practical in human terms.

The ability to create rich, detailed models of natural and artificial phenomena and to process large volumes of experimental data created by a new generation of scientific instruments that are themselves powered by computing makes computing a universal intellectual amplifier, advancing all of science and engineering and powering the knowledge economy. Cloud computing is the latest technological evolution of computational science, allowing groups to host, process, and analyze large volumes of multidisciplinary data. Consolidating computing and storage in very large datacenters creates economies of scale in facility design and construction, equipment acquisition, and operations and maintenance that are not possible when these elements are distributed. Moreover, consolidation and hosting mitigate many of the sociological and technical barriers that have limited multidisciplinary data sharing and collaboration. Finally, cloud hosting facilitates long-term data preservation—a task that is particularly challenging for universities and government agencies and is critical to our ability to conduct longitudinal experiments.

It is not unreasonable to say that modern datacenters and modern supercomputers are like twins separated at birth. Both are massively parallel in design, and both are organized as a network of communicating computational nodes. The individual nodes of each are based on commodity microprocessors that have multiple cores, large memories, and local disk storage. They both execute applications that are designed to exploit massive amounts of parallelism. Their differences lie in their evolution. Massively parallel supercomputers have been designed to support computation with occasional bursts of input/output and to complete a single massive calculation as fast as possible, one job at a time. In contrast, datacenters direct their power outward to the world and consume vast quantities of input data.

Parallelism can be exploited in cloud computing in two ways. The first is for human access. Cloud applications are designed to be accessed as Web services, so they are organized as two or more layers of processes. One layer provides the service interface to the user’s browser or client application. This “Web role” layer accepts us-



ers' requests and manages the tasks assigned to the second layer. The second layer of processes, sometimes known as the “worker role” layer, executes the analytical tasks required to satisfy user requests. One Web role and one worker role may be sufficient for a few simultaneous users, but if a cloud application is to be widely used—such as for search, customized maps, social networks, weather services, travel data, or online auctions—it must support thousands of concurrent users.

The second way in which parallelism is exploited involves the nature of the data analysis tasks undertaken by the application. In many large data analysis scenarios, it is not practical to use a single processor or task to scan a massive dataset or data stream to look for a pattern—the overhead and delay are too great. In these cases, one can partition the data across large numbers of processors, each of which can analyze a subset of the data. The results of each “sub-scan” are then combined and returned to the user.

This “map-reduce” pattern is frequently used in datacenter applications and is one in a broad family of parallel data analysis queries used in cloud computing. Web search is the canonical example of this two-phase model. It involves constructing a searchable keyword index of the Web's contents, which entails creating a copy of the Web and sorting the contents via a sequence of map-reduce steps. Three key technologies support this model of parallelism: Google has an internal version [4], Yahoo! has an open source version known as Hadoop, and Microsoft has a map-reduce tool known as DryadLINQ [5]. Dryad is a mechanism to support the execution of distributed collections of tasks that can be configured into an arbitrary directed acyclic graph (DAG). The Language Integrated Query (LINQ) extension to C# allows SQL-like query expressions to be embedded directly in regular programs. The DryadLINQ system can automatically compile these queries into Dryad DAG, which can be executed automatically in the cloud.

Microsoft Windows Azure supports a combination of multi-user scaling and data analysis parallelism. In Azure, applications are designed as stateless “roles” that fetch tasks from queues, execute them, and place new tasks or data into other queues. Map-reduce computations in Azure consist of two pools of worker roles: mappers, which take map tasks off a map queue and push data to the Azure storage, and reducers, which look for reduce tasks that point to data in the storage system that need reducing. Whereas DryadLINQ executes a static DAG, Azure can execute an implicit DAG in which nodes correspond to roles and links correspond to messages in queues. Azure computations can also represent the parallelism generated by very large numbers of concurrent users.



This same type of map-reduce data analysis appears repeatedly in large-scale scientific analyses. For example, consider the task of matching a DNA sample against the thousands of known DNA sequences. This kind of search is an “embarrassingly parallel” task that can easily be sped up if it is partitioned into many independent search tasks over subsets of the data. Similarly, consider the task of searching for patterns in medical data, such as to find anomalies in fMRI scans of brain images, or the task of searching for potential weather anomalies in streams of events from radars.

Finally, another place where parallelism can be exploited in the datacenter is at the hardware level of an individual node. Not only does each node have multiple processors, but each typically has multiple computer cores. For many data analysis tasks, one can exploit massive amounts of parallelism at the instruction level. For example, filtering noise from sensor data may involve invoking a Fast Fourier Transform (FFT) or other spectral methods. These computations can be sped up by using general-purpose graphics processing units (GPGPUs) in each node. Depending on the rate at which a node can access data, this GPGPU-based processing may allow us to decrease the number of nodes required to meet an overall service rate.

The World Wide Web began as a loose federation of simple Web servers that each hosted scientific documents and data of interest to a relatively small community of researchers. As the number of servers grew exponentially and the global Internet matured, Web search transformed what was initially a scientific experiment into a new economic and social force. The effectiveness of search was achievable only because of the available parallelism in massive datacenters. As we enter the period in which all of science is being driven by a data explosion, cloud computing and its inherent ability to exploit parallelism at many levels has become a fundamental new enabling technology to advance human knowledge.

REFERENCES

- [1] President’s Information Technology Advisory Committee, “Computational Science: Ensuring America’s Competitiveness,” June 2005, www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf.
- [2] D. A. Reed, Ed., “Workshop on The Roadmap for the Revitalization of High-End Computing,” June 2003, www.cra.org/reports/supercomputing.pdf.
- [3] S. L. Graham, M. Snir, and C. A. Patterson, Eds., *Getting Up to Speed: The Future of Supercomputing*, Washington, D.C.: National Academies Press, 2004, www.nap.edu/openbook.php?record_id=11148.
- [4] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” OSDI’04: Sixth Symposium on Operating Systems Design and Implementation, San Francisco, CA, Dec. 2004, doi: 10.1145/1327452.1327492.



- [5] Y. Yu., M. Isard, D. Fetterly, M. Budiu, Ú. Erlingsson, P. Kumar Gunda, and J. Currey, “DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language,” OSDI’08 Eighth Symposium on Operating Systems Design and Implementation.