



# Gray's Laws: Database-centric Computing in Science

ALEXANDER S.  
SZALAY

The Johns Hopkins  
University

JOSÉ A. BLAKELEY  
Microsoft

**T**HE EXPLOSION IN SCIENTIFIC DATA has created a major challenge for cutting-edge scientific projects. With datasets growing beyond a few tens of terabytes, scientists have no off-the-shelf solutions that they can readily use to manage and analyze the data [1]. Successful projects to date have deployed various combinations of flat files and databases [2]. However, most of these solutions have been tailored to specific projects and would not be easy to generalize or scale to the next generation of experiments. Also, today's computer architectures are increasingly imbalanced; the latency gap between multi-core CPUs and mechanical hard disks is growing every year, making the challenges of data-intensive computing harder to overcome [3]. What is needed is a systematic and general approach to these problems with an architecture that can scale into the future.

## GRAY'S LAWS

Jim Gray formulated several informal rules—or laws—that codify how to approach data engineering challenges related to large-scale scientific datasets. The laws are as follows:

1. Scientific computing is becoming increasingly data intensive.
2. The solution is in a “scale-out” architecture.
3. Bring computations to the data, rather than data to the computations.

4. Start the design with the “20 queries.”
5. Go from “working to working.”

It is important to realize that the analysis of observational datasets is severely limited by the relatively low I/O performance of most of today’s computing platforms. High-performance numerical simulations are also increasingly feeling the “I/O bottleneck.” Once datasets exceed the random access memory (RAM) capacity of the system, locality in a multi-tiered cache no longer helps [4]. Yet very few high-end platforms provide a fast enough I/O subsystem.

High-performance, scalable numerical computation also presents an algorithmic challenge. Traditional numerical analysis packages have been designed to operate on datasets that fit in RAM. To tackle analyses that are orders of magnitude larger, these packages must be redesigned to work in a multi-phase, divide-and-conquer manner while maintaining their numerical accuracy. This suggests an approach in which a large-scale problem is decomposed into smaller pieces that can be solved in RAM, whereas the rest of the dataset resides on disk. This approach is analogous to the way in which database algorithms such as sorts or joins work on datasets larger than RAM. These challenges are reaching a critical stage.

Buying larger network storage systems and attaching them to clusters of compute nodes will not solve the problem because network/interconnect speeds are not growing fast enough to cope with the yearly doubling of the necessary storage. Scale-out solutions advocate simple building blocks in which the data is partitioned among nodes with locally attached storage [5]. The smaller and simpler these blocks are, the better the balance between CPUs, disks, and networking can become. Gray envisaged simple “CyberBricks” where each disk drive has its own CPU and networking [6]. While the number of nodes on such a system would be much larger than in a traditional “scale-up” architecture, the simplicity and lower cost of each node and the aggregate performance would more than make up for the added complexity. With the emergence of solid-state disks and low-power motherboards, we are on the verge of being able to build such systems [7].

#### **DATABASE-CENTRIC COMPUTING**

Most scientific data analyses are performed in hierarchical steps. During the first pass, a subset of the data is extracted by either filtering on certain attributes (e.g., removing erroneous data) or extracting a vertical subset of the columns. In the next step, data are usually transformed or aggregated in some way. Of course, in more

complex datasets, these patterns are often accompanied by complex joins among multiple datasets, such as external calibrations or extracting and analyzing different parts of a gene sequence [8]. As datasets grow ever larger, the most efficient way to perform most of these computations is clearly to move the analysis functions as close to the data as possible. It also turns out that most of these patterns are easily expressed by a set-oriented, declarative language whose execution can benefit enormously from cost-based query optimization, automatic parallelism, and indexes.

Gray and his collaborators have shown on several projects that existing relational database technologies can be successfully applied in this context [9]. There are also seamless ways to integrate complex class libraries written in procedural languages as an extension of the underlying database engine [10, 11].

MapReduce has become a popular distributed data analysis and computing paradigm in recent years [12]. The principles behind this paradigm resemble the distributed grouping and aggregation capabilities that have existed in parallel relational database systems for some time. New-generation parallel database systems such as Teradata, Aster Data, and Vertica have rebranded these capabilities as “MapReduce in the database.” New benchmarks comparing the merits of each approach have been developed [13].

### CONNECTING TO THE SCIENTISTS

One of the most challenging problems in designing scientific databases is to establish effective communication between the builder of the database and the domain scientists interested in the analysis. Most projects make the mistake of trying to be “everything for everyone.” It is clear that that some features are more important than others and that various design trade-offs are necessary, resulting in performance trade-offs.

Jim Gray came up with the heuristic rule of “20 queries.” On each project he was involved with, he asked for the 20 most important questions the researchers wanted the data system to answer. He said that five questions are not enough to see a broader pattern, and a hundred questions would result in a shortage of focus. Since most selections involving human choices follow a “long tail,” or so-called 1/f distribution, it is clear that the relative information in the queries ranked by importance is logarithmic, so the gain realized by going from approximately 20 ( $2^{4.5}$ ) to 100 ( $2^{6.5}$ ) is quite modest [14].

The “20 queries” rule is a moniker for a design step that engages the domain scientist and the database engineer in a conversation that helps bridge the semantic

gap between nouns and verbs used in the scientific domain and the entities and relationships stored in the database. Queries define the precise set of questions in terms of entities and relationships that domain scientists expect to pose to the database. At the end of a full iteration of this exercise, the domain scientist and the database speak a common language.

This approach has been very successful in keeping the design process focused on the most important features the system must support, while at the same time helping the domain scientists understand the database system trade-offs, thereby limiting “feature creep.”

Another design law is to move from working version to working version. Gray was very much aware of how quickly data-driven computing architecture changes, especially if it involves distributed data. New distributed computing paradigms come and go every other year, making it extremely difficult to engage in a multi-year top-down design and implementation cycle. By the time such a project is completed, the starting premises have become obsolete. If we build a system that starts working only if every one of its components functions correctly, we will never finish.

The only way to survive and make progress in such a world is to build modular systems in which individual components can be replaced as the underlying technologies evolve. Today’s service-oriented architectures are good examples of this. Web services have already gone through several major evolutionary stages, and the end is nowhere in sight.

#### **FROM TERASCALE TO PETASCALE SCIENTIFIC DATABASES**

By using Microsoft SQL Server, we have successfully tackled several projects on a scale from a few terabytes (TB) to tens of terabytes [15-17]. Implementing databases that will soon exceed 100 TB also looks rather straightforward [18], but it is not entirely clear how science will cross the petascale barrier. As databases become larger and larger, they will inevitably start using an increasingly scaled-out architecture. Data will be heavily partitioned, making distributed, non-local queries and distributed joins increasingly difficult.

For most of the petascale problems today, a simple data-crawling strategy over massively scaled-out, share-nothing data partitions has been adequate (MapReduce, Hadoop, etc.). But it is also clear that this layout is very suboptimal when a good index might provide better performance by orders of magnitude. Joins between tables of very different cardinalities have been notoriously difficult to use with these crawlers.

Databases have many things to offer in terms of more efficient plans. We also need to rethink the utility of expecting a monolithic result set. One can imagine crawlers over heavily partitioned databases implementing a construct that can provide results one bucket at a time, resulting in easier checkpointing and recovery in the middle of an extensive query. This approach is also useful for aggregate functions with a clause that would stop when the result is estimated to be within, for example, 99% accuracy. These simple enhancements would go a long way toward sidestepping huge monolithic queries—breaking them up into smaller, more manageable ones.

Cloud computing is another recently emerging paradigm. It offers obvious advantages, such as co-locating data with computations and an economy of scale in hosting the services. While these platforms obviously perform very well for their current intended use in search engines or elastic hosting of commercial Web sites, their role in scientific computing is yet to be clarified. In some scientific analysis scenarios, the data needs to be close to the experiment. In other cases, the nodes need to be tightly integrated with a very low latency. In yet other cases, very high I/O bandwidth is required. Each of these analysis strategies would be suboptimal in current virtualization environments. Certainly, more specialized data clouds are bound to emerge soon. In the next few years, we will see if scientific computing moves from universities to commercial service providers or whether it is necessary for the largest scientific data stores to be aggregated into one.

## CONCLUSIONS

Experimental science is generating vast volumes of data. The Pan-STARRS project will capture 2.5 petabytes (PB) of data each year when in production [18]. The Large Hadron Collider will generate 50 to 100 PB of data each year, with about 20 PB of that data stored and processed on a worldwide federation of national grids linking 100,000 CPUs [19]. Yet generic data-centric solutions to cope with this volume of data and corresponding analyses are not readily available [20].

Scientists and scientific institutions need a template and collection of best practices that lead to balanced hardware architectures and corresponding software to deal with these volumes of data. This would reduce the need to reinvent the wheel. Database features such as declarative, set-oriented languages and automatic parallelism, which have been successful in building large-scale scientific applications, are clearly needed.

We believe that the current wave of databases can manage at least another order of magnitude in scale. So for the time being, we can continue to work. However,

it is time to start thinking about the next wave. Scientific databases are an early predictor of requirements that will be needed by conventional corporate applications; therefore, investments in these applications will lead to technologies that will be broadly applicable in a few years. Today's science challenges are good representatives of the data management challenges for the 21st century. Gray's Laws represent an excellent set of guiding principles for designing the data-intensive systems of the future.

#### REFERENCES

- [1] A. S. Szalay and J. Gray, "Science in an Exponential World," *Nature*, vol. 440, pp. 23–24, 2006, doi: 10.1038/440413a.
- [2] J. Becla and D. Wang, "Lessons Learned from Managing a Petabyte," CIDR 2005 Conference, Asilomar, 2005, doi: 10.2172/839755.
- [3] G. Bell, J. Gray, and A. Szalay, "Petascale Computational Systems: Balanced Cyber-Infrastructure in a Data-Centric World," *IEEE Computer*, vol. 39, pp. 110–112, 2006, doi: 10.1109/MC.2006.29.
- [4] W. W. Hsu and A. J. Smith, "Characteristics of I/O traffic in personal computer and server workloads," *IBM Sys. J.*, vol. 42, pp. 347–358, 2003, doi: 10.1147/sj.422.0347.
- [5] A. Szalay, G. Bell, et al., "GrayWulf: Scalable Clustered Architecture for Data Intensive Computing," Proc. HICSS-42 Conference, Hawaii, 2009, doi: 10.1109/HICSS.2009.750.
- [6] J. Gray, Cyberbricks Talk at DEC/NT Wizards Conference, 2004; T. Barclay, W. Chong, and J. Gray, "TerraServer Bricks – A High Availability Cluster Alternative," Microsoft Technical Report, MSR-TR-2004-107, [http://research.microsoft.com/en-us/um/people/gray/talks/DEC\\_Cyberbrick.ppt](http://research.microsoft.com/en-us/um/people/gray/talks/DEC_Cyberbrick.ppt).
- [7] A. S. Szalay, G. Bell, A. Terzis, A. S. White, and J. Vandenberg, "Low Power Amdahl Blades for Data-Intensive Computing," <http://perspectives.mvdirona.com/content/binary/AmdahlBladesV3.pdf>.
- [8] U. Roehm and J. A. Blakeley, "Data Management for High-Throughput Genomics," *Proc. CIDR*, 2009.
- [9] J. Gray, D. T. Liu, M. A. Nieto-Santisteban, A. S. Szalay, G. Heber, and D. DeWitt, "Scientific Data Management in the Coming Decade," *ACM SIGMOD Record*, vol. 34, no. 4, pp. 35–41, 2005; also MSR-TR-2005-10, doi: 10.1145/1107499.1107503.
- [10] A. Acheson et al., "Hosting the .NET Runtime in Microsoft SQL Server," ACM SIGMOD Conf., 2004, doi: 10.1145/1007568.1007669.
- [11] J. A. Blakeley, M. Henaire, C. Kleinerman, I. Kunen, A. Prout, B. Richards, and V. Rao, ".NET Database Programmability and Extensibility in Microsoft SQL Server," ACM SIGMOD Conf., 2008, doi: 10.1145/1376616.1376725.
- [12] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI, 2004, doi: 10.1145/1327452.1327492.
- [13] A. Pavlo et al., "A Comparison of Approaches to Large-Scale Data Analysis," ACM SIGMOD Conf., 2009, doi: 10.1145/1559845.1559865.
- [14] C. Anderson. *The Long Tail*. New York: Random House, 2007.
- [15] A. R. Thakar, A. S. Szalay, P. Z. Kunszt, and J. Gray, "The Sloan Digital Sky Survey Science Archive: Migrating a Multi-Terabyte Astronomical Archive from Object to Relational DBMS," *Comp. Sci. and Eng.*, vol. 5, no. 5, pp. 16–29, Sept. 2003.

- [16] A. Terzis, R. Musaloiu-E., J. Cogan, K. Szlavecz, A. Szalay, J. Gray, S. Ozer, M. Liang, J. Gupchup, and R. Burns, "Wireless Sensor Networks for Soil Science," *Int. J. Sensor Networks*, to be published 2009.
- [17] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, "A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence," *J. Turbul.*, vol. 9, no. 31, pp. 1–29, 2008, doi: 10.1080/14685240802376389.
- [18] Pan-STARRS: Panoramic Survey Telescope and Rapid Response System, <http://pan-starrs.ifa.hawaii.edu>.
- [19] A. M. Parker, "Understanding the Universe," in *Towards 2020 Science*, Microsoft Corporation, 2006, [http://research.microsoft.com/towards2020science/background\\_overview.htm](http://research.microsoft.com/towards2020science/background_overview.htm).
- [20] G. Bell, T. Hey, and A. Szalay, "Beyond the Data Deluge," *Science*, vol. 323, no. 5919, pp. 1297–1298, 2009, doi: 10.1126/science.1170411.