1 Data-Intensive Computing: A Challenge for the 21st Century

Ian Gorton and Deborah K. Gracio

1.1 Introduction

In our world of rapid technological change, occasionally it is instructive to contemplate how much has altered in the last few years. Remembering life without the ability to view the World Wide Web (WWW) through browser windows will be difficult, if not impossible, for less "mature" readers. Is it only seven years since YouTube first appeared, a Web site that is now ingrained in many facets of modern life? How did we survive without Facebook all those (actually, about five) years ago?

In 2010, various estimates put the amount of data stored by consumers and businesses around the world in the vicinity of 13 exabytes, with a growth rate of 20 to 25 percent per annum. That is a lot of data. No wonder IBM is pursuing building a 120-petabyte storage array.[1] Obviously there is going to be a market for such devices in the future. As data volumes of all types – from video and photos to text documents and binary files for science – continue to grow in number and resolution, it is clear that we have genuinely entered the realm of data-intensive computing, or as it is often now referred to, big data.[2]

Interestingly, the term "data-intensive computing" was actually coined by the scientific community. Traditionally, scientific codes have been starved of sufficient compute cycles, a paucity that has driven the creation of ever larger and faster high-performance computing machines, typically known as supercomputers. The Top 500 Web site[3] shows the latest benchmark results that characterize the fastest supercomputers on the planet. While this fascination with compute performance continues, scientific computing has been gradually coming to terms with the challenges brought by ever-increasing data size and

[1] http://www.technologyreview.com/computing/38440/page1/.
[2] http://en.wikipedia.org/wiki/Big_data.
[3] http://www.top500.org/.

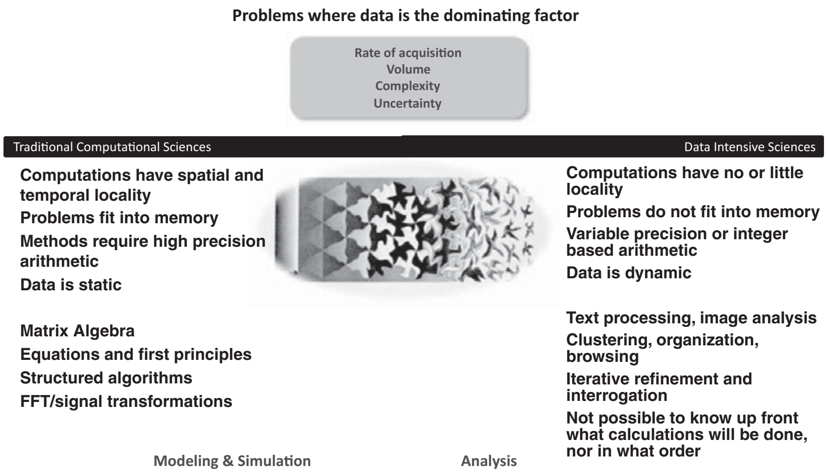
Problems where data is the dominating factor
Rate of acquisition
Volume
Complexity
Uncertainty
Traditional Computational Sciences
Data Intensive Sciences
Computations have spatial and temporal locality
Problems fit into memory
Methods require high precision arithmetic
Data is static
Matrix Algebra
Equations and first principles
Structured algorithms
FFT/signal transformations
Computations have no or little locality
Problems do not fit into memory
Variable precision or integer based arithmetic
Data is dynamic
Text processing, image analysis
Clustering, organization, browsing
Iterative refinement and interrogation
Not possible to know up front what calculations will be done, nor in what order
Modeling & Simulation
Analysis

Figure 1.1. The major concerns of computational and data-intensive applications.

complexity. In 1998, William Johnston's paper at the Seventh IEEE Symposium on High Performance Distributed Computing [1] described the evolution of data-intensive computing over the previous decade. The achievements described in that paper, while state of the art at the time, now seem modest in comparison to the scale of the problems that are routinely tackled in present-day data-intensive computing applications.

More recently, others including Hey and Trefethen [2], Bell et al. [3], and Newman et al. [4] have described the magnitude of the data-intensive problems faced by the e-science community. Their descriptions of the *data deluge* that future applications must process, in domains ranging from science to business informatics, create a compelling argument for research and development (R&D) to be targeted at discovering scalable hardware and software solutions for data-intensive problems. While multi-petabyte data sets and gigabit data streams are today's frontier of data-intensive applications, no doubt ten years from now we will fondly reminisce about these problems, and will be concerned about the looming exascale applications we need to address.

Figure 1.1 lists the general features of traditional computational science applications and their data-intensive counterparts. The former focuses more on solving mathematical equations for static data sets, whereas the latter is concerned with more exploratory search and processing of large, dynamic, and complex data collections.

1.2 Some Examples

The challenge of managing massive and complex data sets is one faced by many enterprises already. The following are some examples of the current state of the art that illustrate the magnitude of what is currently possible.

1.2.1 Internet Search

Internet search is the current poster child for data-intensive computing. While the precise amounts of data held by Google, Yahoo!, Microsoft, and other search providers is a closely guarded commercial secret, it is pretty obvious that "building a copy of the Internet" to answer Internet searches is going to result in a daunting data archive. In 2008, it was reported that Google processed about 24 petabytes of data per day [5], but other hard facts from the search engine providers are difficult to come by. As of November 2011, an estimate of the number of Web pages indexed by Google is in the vicinity of 50 billion.[4] We'll leave it to the reader to extrapolate an actual data size from this value, but discussions at a recent workshop (overheard by one of the authors) strayed into descriptions of 70 petabytes (PBs) of data in a single Google BigTable.[5]

In order to manage and rapidly search these multi-petabyte repositories to answer searches, Google has custom built a specialized file system and indexing scheme that allows queries to execute in parallel across clusters of thousands of commodity machines. As a contrast, and slightly tangential to Internet search, the Internet Archive (http://www.archive.org/) contained about 5.8 petabytes of data as of December 2010 and was growing at the rate of about 100 terabytes per month in March 2009.

1.2.2 Internet Applications

Of course, the Internet is much more than search. Many Web sites manage and deliver data to millions of users around the world, and each faces its own data challenges. In May 2010, YouTube, for example, had more than 14 billion views of videos and more than 48 hours of new videos are uploaded to the site every minute. Given the size of video files, this inevitably leads to a repository of many, many petabytes. In 2009, YouTube was serving 1 billion page views

[4] http://www.worldwidewebsize.com/.
[5] http://en.wikipedia.org/wiki/BigTable.

per day.[6] In a similar vein, Netflix has more than 1 petabyte of data stored on Amazon's EC2 cloud.

Not all large data repositories on the Internet store video. eBay, for example, holds its data in multi-petabyte databases, using technology from Teradata. In 2010, these databases were of the order of 15 petabytes, spanning user data and Web and network event logs.[7] Facebook relies heavily on MySQL databases and the distributed object cache, Memcached (http://memcached.org/). While no data sizes are known, the Facebook infrastructure is reported to divide its MySQL database into 4,000 shards in order to handle the site's massive data volume, and runs 9,000 instances of Memcached to process the number of transactions the database must serve.[8]

1.2.3 Business Applications

Large businesses such as financial institutions, telecommunications operators, and retail outlets all must deal with daunting complex data collections. In 2010, AT&T carried 19 petabytes of data each day on its network[9] and was one of the first organizations to report having a petabyte data warehouse. Others include Walmart, Dell, and Bank of America, and there are no doubt many more. Traditionally, large business data collections execute on data warehouse technology from organizations such as Teradata, Oracle, and IBM. These data warehouses integrate key business data (such as call records or sales records across regions) from multiple operational systems in each business and make the data available for querying and reporting through specialized business intelligence tools.[10] More recently, Hadoop-based data warehouses are making an impact; the most prominent example is Facebook's 21-PB warehouse.[11]

1.2.4 Science

Modern science is becoming increasingly data intensive, driven by modern instrumentation and high-fidelity simulations running on ever-growing supercomputers. In terms of data generation, CERN's Large Hadron Collider (LHC)[12] particle accelerator is currently the most challenging experiment in

[6] http://www.datacenterknowledge.com/archives/2009/10/09/1-billion-page-views-a-day-for-youtube/.
[7] http://www.dbms2.com/2010/10/06/ebay-followup-greenplum-out-teradata-10-petabytes-hadoop-has-some-value-and-more/.
[8] http://gigaom.com/cloud/facebook-trapped-in-mysql-fate-worse-than-death/.
[9] http://www.att.com/gen/press-room?pid=4800&cdvn=news&newsarticleid=30623.
[10] http://en.wikipedia.org/wiki/Business_intelligence_tools.
[11] http://hadoopblog.blogspot.com/2010/05/facebook-has-worlds-largest-hadoop.html.
[12] http://lhc.web.cern.ch/lhc/.

terms of data size, producing around 13 PBs of data per year. Detectors in the LHC generate approximately 300 gigabytes (GBs) per second of data. This data is processed to search for events of interest, which reduces the data that is stored for further processing to about 300 megabytes per second (MB/s). Processed data from experiments is distributed from CERN to several other institutions around the world, which act as backups for CERN data and serve as regional data centers to support a whole range of science with universities in their region. Within a decade, astronomy may take over as the largest generator of scientific data when the Square Kilometer Array radio telescope is built.[13] This massive project will create a telescope fifty times more sensitive than exists today. The anticipated 1-TB/s data stream (or an exabyte of data every thirteen days when in full operation) will need to be processed in real time to reduce it to a size that can be meaningfully stored and processed. Even then, estimates are that a 100-petaflop supercomputer, an order of magnitude more powerful than exists today, will be needed for scientific analysis.

Simulations are also generators of complex and massive data sets. For example, climate simulations executing on petaflop supercomputers produce thousands of data sets that contain the predicted values of various climate variables (temperature, wind speed, and so on) for the duration of the simulation, often for hundreds of years. These data sets fuel scientific investigations around the world on the effects of climate change. To address the challenge of managing the results from multiple simulations with different characteristics (scale, time, or initial conditions), the climate community has invested in the Earth System Grid (ESG).[14] ESG provides a gateway to a data collection of hundreds of terabytes of results from climate simulations that are physically hosted at data centers across the United States and supports 2,500 users.

Systems biology is another scientific discipline that is heavily data dependent. The discipline is characterized by numerous GB-TB data collections, typically accessed through applications made available to users for downloading and processing specified data sets. Examples of such sites are GenBank[15] and KEGG.[16] Currently, these applications are hosted at institutional sites and accessed through Web-based interfaces or custom-built software applications. The range of features supported by each application for interactive users varies in range and quality, as does the extent of programmatic facilities for building access to the data collections into complex and multistage analyses. As the size and complexity of biological data sets continue to grow, this approach to

[13] http://www.skatelescope.org/.
[14] www.earthsystemgrid.org.
[15] http://www.ncbi.nlm.nih.gov/genbank/.
[16] http://www.genome.jp/kegg/.

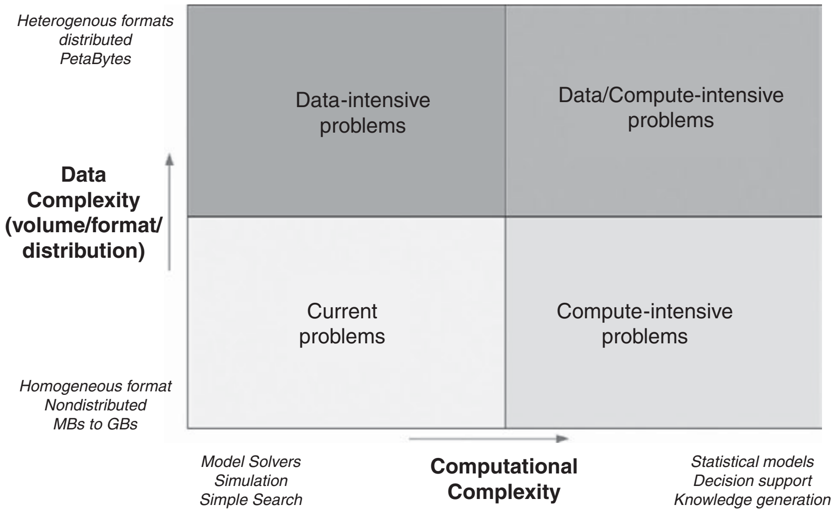

Figure 1.2. Data-intensive computing dimensions.

data analysis is becoming increasingly demanding, and new architectures for serving data and providing analytics are required. An early example of this is the iPlant Collaborative [6] at the University of Texas.

1.3 Characterizing Data-Intensive Applications

As the previous section illustrates, data-intensive computing challenges can be found in many different application domains. Across these domains, the specific challenges vary in their characteristics, scale, and complexity. Fundamentally, however, data-intensive applications have two major challenges:

1. Manage and process exponentially growing data volumes, often arriving in streams from sensors, networks, or outputs from simulations.
2. Significantly reduce data analysis cycles so that timely analyses and decisions can be made.

Undoubtedly, there is an overlap between data- and compute-intensive problems. Figure 1.2 shows a simple diagram that can be used to classify the application space between data and compute intensiveness.

Existing and purely data-intensive applications process multi-terabyte to petabyte size data sets. This data is routinely distributed and in heterogeneous formats, requiring multistep processing by analytical pipelines that incorporate various transformations and fusion of data. Processing typically scales near-linearly with data size, and is often amenable to straightforward parallelization. Key research issues involve data management, filtering and fusion techniques, and efficient querying and distribution.

In addition, combined data/compute-intensive problems exhibit an increase in computational complexity. Processing typically scales super-linearly with data size and requires complex searches, analyses, and fusion in order to produce key insights from the data. Application requirements may also place time limits on producing useful results. Key research issues include new algorithms, signature generation, and specialized processing platforms for algorithm acceleration (such as reduced memory latency and hardware accelerators like GPGPUs).

To offer another perspective on data-intensive application characteristics, we've defined a number of key criteria that enable the quantitative and qualitative assessment of different applications. These criteria are as follows:

Data Size: The absolute size of the data that must be processed for an application is a key characteristic. As data sizes grow, issues that are straightforward for small data sets become problematic. As a simple example, it takes a minute or so to upload a low-quality video of thirty seconds from your phone to YouTube. But try uploading a 5-GB high-definition video and you'll find that at the very least it takes a few hours and greatly increases the chance of encountering a network error (or loss of battery power). Massive data sets take longer to write, search, and transmit over a network, and any bottlenecks in your software or hardware are cruelly exposed, which limits overall system performance.

Complexity of analysis: Single pass algorithms through a data collection – for example, a search through a text file for keywords – scale linearly with size. Building indexes over data sets in a single pass can speed up subsequent access if the index can be used. Using the text search example again, Lucene[17] is an open-source technology for full text indexing and searching and is widely used behind many Web sites to power fast text searches. Indexes in databases serve a similar purpose. However, many applications require complex analysis where algorithms do not scale linearly with data size. For example, sorting large data sets is expensive, with a Bubble sort having a time complexity of $O(n^2)$ for *n* items being sorted. Breadth-first graph searches have a worst-case time complexity of O(number of edges + number of nodes). Of course, parallel implementations of algorithms and specialized hardware architectures can be

[17] http://lucene.apache.org.

used to reduce execution times, but data set sizes are still a significant factor in the speedups that can be achieved.

Space complexity also plays a part in analysis complexity. Attractive algorithms minimize the amount of memory and disk space that an algorithm needs. However, there is often a trade-off between time and space complexity for an algorithm. Classic examples of such trade-offs are:

- the space taken by an index in order to speed up search time;
- storing data in compressed formats at the expense of the time taken to uncompress the data when processing is needed; and,
- storing images of commonly accessed charts and analyses, for example historical rainfall amounts, instead of calculating the images from raw data each time a request is made.

Number of data sources: An additional complexity factor in data-intensive applications is the number of data sources that must be accessed. For example, processing network packets to determine connections between machines is nontrivial on a high-speed network, but life gets much more complex when a recognized IP address must be correlated with data from other sources to try to determine the identity of a user. Multiple data sources also exacerbate the complexity of application reliability, as a required data source may fail independently and, therefore, be unavailable when needed.

Heterogeneity of the data: Another key challenge is the heterogeneity of the data required by an application. For example, many bioinformatics applications must query several external databases and relate the returned data to experimental data from sequencing machines. Also, the proliferation of formats, both proprietary and standardized, greatly complicates this task. Even data in the same format, for example NetCDF, may have data items that represent different concepts that need correlating in an application. Appropriate metadata, controlled vocabularies, and ontologies are all key components of making handling data heterogeneity both tractable and scalable.

Distribution of data: Distributed data sources bring both reliability and latency challenges. Applications that access external distributed data must deal with possible failures, and access to data over wide area networks incurs latency costs that can become prohibitive as data sets grow in size, even when compression is used.

Timeliness of processing: Most data-intensive applications, such as searching for new stars from telescope data, just need to execute as fast as possible in producing a result. Others have time constraints within which analyses must complete in order to produce useful results. Video processing is a good example. Loading a video onto YouTube is an example of the former, whereas

Table 1.1. *Using the assessment framework for two different data-intensive applications*

Application Characteristic	Cyber-Security	Bioinformatics
Data size	10s of TBs per day from network sensors, in the form of processed raw network data	TBs of data in experimental databases
Complexity of analysis	Multistage, comprising signature generation, clustering, anomaly detection, and visualization	Multistage, comprising input data transformations, HPC analysis code, and interactive visualization
Heterogeneity of data	Correlation of Web-mined text information with structured network data	Need to process both proteomic and genomic data
Number of data sources	For core processing pipeline, there is one major source that receives reduced data from many network sensors that perform preliminary processing.	Databases for relevant gene and protein data sets
Distribution of data	Core processing pipeline handles one incoming data stream from network sensors and stores results in a single database.	Processing pipeline needs to retrieve data sets from geographically distributed databases. This can be costly if transfers occur over the Internet.
Timeliness of processing	Results of analysis required in seconds to a minute	"As fast as possible" processing requirements, but ideally a few 10s of seconds or less for interactive exploration of data

automated recognition and tracking of vehicles by citywide surveillance systems obviously has time constraints for processing. For low-latency real-time constraints, specialized hardware is often the only solution for a given application. When constraints are of the order of seconds to minutes, parallel processing and high-speed networks can often meet the necessary deadlines.

As an example of how this framework can be used, we classify the major characteristics of two applications that are described in later chapters in Table 1.1. As can be seen, this framework gives a concise representation that can be used to compare applications across the broad spectrum of data-intensive and big data computing.

1.4 Summary

This chapter lays out the broad landscape of data-intensive computing, describing example application domains where such problems are prevalent and presenting a framework for characterizing data-intensive software systems. Given the exponentially increasing data volumes being generated, and the new and innovative ways being discovered for analyzing and exploiting that data, we are now only at the beginning of the data-intensive, or the big data computing era.

The future will require the creation of breakthrough technologies to address many key data-intensive computing problems, bringing together results from various disciplines in computer science, engineering, and mathematics. For example, the following are all pieces of the puzzle required by data-intensive computing solutions:

- New algorithms that can scale to search and process massive data sets.
- New metadata management technologies that can scale to handle complex, heterogeneous, and distributed data sources.
- Advances in high-performance computing platforms to provide uniform high-speed memory access to multi-terabyte data structures.
- High-performance, high-reliability, and petascale distributed file systems.
- Flexible and high-performance software integration technologies that facilitate "plug and play" integration of software components running on diverse computing platforms to quickly form analytical pipelines.
- Data signature generation techniques for data reduction and rapid processing.
- Mobile code-based analytics, where processing is moved to the data.

The remainder of this book describes the current state of the art and potential futures in many of these areas. The next chapter dissects the anatomy of data-intensive applications, describing the various computational methods, software components, and technologies that are relevant. The following collection of chapters delves into detail on hardware architectures, data management approaches, and cloud-based technologies that are all fundamental application building blocks. Next, the book contains chapters on fundamental algorithms for data classification, clustering, and dimensionality reduction. Finally, data-intensive applications in biology and cyber-security are described, giving insights into how the various building blocks covered in earlier chapters can be brought together to create innovative solutions to challenging big data problems.