Divide and Recombine (D&R):
Data Science for Large Complex Data

William S. Cleveland, Statistics Department, Purdue, West Lafayette IN

Ryan Hafen, Pacific Northwest National Labs, Richland WA

This document is being written by two of us, Ryan and Bill, but in Section 1, Bill is in the first person citing historical events to address some of the matters raised by discussants Steve Scott, Scott Vanderwiel, Kary Myers, and Michael Kane.

The need for deep analysis of large complex data has brought a focus to data science. The reasoning is simple. Data science consists of all technical areas that come into play in the analysis of data, and deep analysis of large complex data challenges all of the technical areas, from statistical theory to the architecture of clusters designed specifically for data. What is more, research in the technical areas needs to be tightly integrated.

In Section 2, Ryan and Bill describe a specific research project in data science for large complex data: the divide and recombine (D&R) statistical approach, and the Tessera D&R computational environment. This serves as an example to demonstrate work in all the areas and their tight integration.

1 Comments on the Discussions

1.1 Steve Scott

Why Computational Systems for Data Analysis are So Important

Steve writes: "After reading this article with the benefit of hindsight, I can't decide whether it was successful. It certainly offers breathtakingly prescient advice on how to train statisticians . . ."

On the first, no, it did not have great impact. Some people read the material and acted on it. It did not start a movement. However, we are now targeting impact in our D&R research, whose current team consists of Ryan and colleagues at PNNL; Saptarshi Guha at Mozilla; colleagues and I at Purdue; and soon, we believe, colleagues in other statistics departments.

On the "breathtakingly prescient advice" I have a different attitude. I'd say it was completely obvious, at least given that to which I was exposed. This came from many experiences. I will describe two.

I am a grad student at Yale, programming in Fortran, preparing punch cards, submitting card decks, and getting output 4 hours later if I am lucky. The Watson's, founders of IBM, have a very close association with Yale. Someone tells me that in the Yale math building there is a room with IBM Selectric typewriter terminals. From those terminals you can access a computer directly at IBM's Yorktown Heights location. You can use a language called "APL". It has a "statistics package." Soon I am getting output in seconds rather than hours. And I can do this with commands that are orders of magnitude simpler to express than in Fortran, which saves me an extraordinary amount of time writing code. I go to my advisor, Jimmie Savage, and tell him that those who know how to compute will have a tremendous impact on the future of statistics, which is quite audacious because Savage at that time is revolutionizing statistics by getting Bayesian inference going.

Later, I am an assistant professor of statistics at UNC, Chapel Hill, a statistics department with substantial fame. I get a reprint in the mail by Anne Freeney and John Gaby at Bell Labs analyzing what for the time are large complex data on microwave transmission. I am stunned at what they can do with that much data there at Bell Labs, which has very substantial computational systems for data analysis. I decide that is where I need to be, and get a job there. When I arrive, I find a Bell Labs culture has been valuing highly research in computer systems for quite some time. From this had come or was coming Unix, C, troff, Make, awk, C++, and S (parent of R). Unix, Make, and S win the ACM Software System Award, by far the most prestigious award in software systems, one per year. Those of us in the two statistics departments at Bell Labs know we benefit immensely from this culture. Little is more obvious. We all look at universities with whom we have a lot of collaboration, and just shake our heads. So my accomplishment in the 2001 paper was simply to express the obvious, which at that point had not been done by addressing what universities should do.

Data Science Includes Computational Systems for Data Analysis

Steve also writes: "…data science is happening, but it is being driven by external forces rather than change coming from within our field. Most of us missed an opportunity to help shape the birth of data science a decade ago. The good news is that the opportunity is still there, and I hope we will capitalize on it."

Yes, data science and research in it for large complex data is happening. There is certainly a lot of talk. Real progress is moving forward slowly. However, academic departments of statistics generally are not yet making much headway at all. I agree with Steve that it is not too late. He makes a good case for this. We should not underestimate the vastness of the knowledge base of the field of statistics. There is much good common sense about data analysis, a critical statistical thinking. There is a vast collection of models and methods. There are an extraordinary number of technical disciplines into which statistics has penetrated deeply. This makes statistics the best potential base for attacking the challenges of large complex data.

But the attack on large complex data in the field of statistics will not happen unless "statistics" becomes a synonym of "data science." The principal deficit is computational systems, the persistence of a culture that does not sufficiently value knowledge of systems and research in them. Make no mistake. If a statistics department does not embrace computational systems to a high degree, it will not be in the data science and large complex data arenas. At a bare minimum, there needs to be deep knowledge of systems, participation in their assembly, and effective use of them. Better still is strong research in computer systems. New assistant professors who do this strong research need to get tenure for it. True, statistics departments are hiring people in computing, but let us see if they mean computational systems and not just computing algorithms.

There are in fact many organizations that do value work in computational systems for data analysis. Google sure does, where Steve works. Our D&R computational system owes a great debt to such places, especially to Google, but also to Yahoo, Apache, and Cloudera. The national labs like Los Alamos and PNNL sure value it, where Kary, Scott, and Ryan work.

Why Developers of Computational Systems for Data Analysis Need Deep Knowledge of Data Analysis

At one level, the title of this section might seem completely obvious, but it turns out it needs to be said rather explicitly. People in computer science who work in computer systems have knowledge that can be very helpful in building computational environments for data analysis, both hardware and software. One important point in my 2001 paper was that it is important for those in statistics to collaborate closely with such people. However, only such systems knowledge is surely not enough. A computational environment for data analysis has very special requirements that can be adequately addressed only by a combination of deep systems knowledge and deep knowledge of the processes of data analysis.

The aspect of the environment that most needs the knowledge of data analysis is the front end: the software used directly by the data analyst. The software needs to be a domain specific language for analyzing data. The design of such a language has an immense impact on how easy it is to use, which in turn has an immense impact on the time it takes to program with the data to do the analysis. Computer time is an issue, and can stop the whole show, but the time of the analyst is in general much more important. In addition, the language design has an immense impact on the power and flexibility with which the analyst can carry out analysis. The more the power and flexibility of the language, the better the analyst can tailor an analysis to the data.

Also, the language needs to provide access to the 1000s of methods of statistics, machine learning, and visualization. However, the language needs to be much more than just a toolbox that only lets us call a routine to carry out an analytic method and get the output. A good example of an immensely successful domain specific language for data analysis is R, which is discussed in Section 2.

In fact, other components of the design of an environment for data analysis must take also take data analysis into account. For example, for the very back end, the hardware, having to cope with large complex data affects the bundling of the hardware components such as loading up with disk so that there is a large amount of disk per core.

Statistical Theory and Methods for Large Complex Data

I surely agree with Steve that research in statistical theory and methods is needed for data science and large complex data. He writes: "The next natural question is what we (statisticians) can contribute to data science that we haven't already. After all, our expertise lies in understanding properties of statistical procedures, not in computing or software engineering. From an abstract level, it is not clear that any new theoretical challenges are posed by very large data sets. In practice, nothing could be further from the truth." He gives a nice example. Our D&R is a fundamentally statistical approach to large complex data, although one designed for computational gain. Because of its structure, it opens ups a whole new framework of theoretical investigation.

1.2 Scott Vanderwiel and Kary Myers

Learning Data Analysis by Doing Data Analysis

Here is another case of a title that might seem completely obvious, but needs to be said rather explicitly. Kary and Scott write: "We therefore build on Bill's dictate that 'students should analyze data' . . . ".

But they add something extremely important. "For maximum impact, statisticians need to embed themselves in the science." This is absolutely the case. Then they add to this another excellent thought. I pointed out that statisticians working in the analysis of data from a discipline bring statistics to the discipline, and turn researchers in the discipline into statisticians. "blurring exactly who is and who in not a statistician." They say: ". . . we find that, when done right, it also blurs who is and isn't an astronomer or nuclear physicist or chemist."

This emphasis on the statistician learning the discipline is vital because it leads to much better data analysis and therefore better science. Analysis needs to be a fine weaving of analytic methods and subject matter knowledge. This is required not only for the overall strategic actions of the analysis, but also the minute-by-minute tactical actions. The collaborator from the discipline cannot sit next to the statistician throughout the whole analysis.

Clearly, it is important for students to experience data analysis. I recall George Box saying that teaching students statistics without having them analyze data, is like teaching someone to swim without letting them get in the pool.

A number of statistics departments have done quite well here. Some did this at their inception. Many started in the 1990s when they saw that getting involved in applications was important for existence, and could be quite lucrative. Our Purdue Department of Statistics did this circa 1998, and today is widely known for its work in genetics, human and plant, and for work in a number of other disciplines.

Open Source Software

Kary and Scott also raise an issue that is absolutely critical: open source software. It brings immense advantages in many ways. It is particularly attractive at universities because it is free. This can be the case for commercial institutions as well when they get tired of paying large license fees.

But there is a lot more. Open source attracts large support communities, in many cases mustering resources far greater than what a commercial operation could and make a profit. R is an example. We continue this discussion of their comment in Section 2.

Which Logical Thinking?

Kary and Scott also write about theory helping students to develop "the logical and careful thinking needed to learn challenging new areas when needed." I am in full agreement. But I am going to stick with the opinion that measure theory is the wrong choice for this.

Math is deductive logic. The logic of data analysis applied to disciplines like astrophysics and power engineering is inductive, which is very, very different. To be sure, math is a critical tool in this inductive logic, but the top level process is inductive, requiring a host of stages like empirical validation of derived theories, and the replication of experimental findings by different groups. It is better that students see the inductive logic at work through involvement in data analysis, and learn only the math that is useful for this.

Obviously probability is critical. Better the probability of Willy Feller than Andrey Kolmogorov. Nobody can get enough probability of the former kind. After all, our statistical models are Feller probabilistic. We want a student to know intuitively in an instant that the probabilistic mechanism for the time between successive phone calls arriving at an edge router interface on the Internet is changed by a sampling plan that intercepts calls, for example, randomly picking a time and measuring the duration of the call currently in progress at that time. Quite different is intuition about measure theory, which is based on set theory, topology, metric spaces, etc. It can get quite precious, requiring mathematical logic and forcing us to contemplate whether we want to outlaw the axiom of choice so there will be no non-measurable sets. This mathematics is quite beautiful, but its fundamental intuition is not about flipping coins.

1.3 Michael Kane

Measuring Interest in “Data Science”

I like Michael’s idea of searching for usage of terms through time. The terms “big data” and “data science” seem appropriate, but “cloud computing” does not particularly fit because a lot more goes on in clouds than the analysis of data. To see how we stand right now I tried both Google Scholar and Web:

- Scholar: “data science” = 46,800; “big data” = 41,600
- Web: “data science” = 1,380,000; “big data” = 14,000,000

Scholar is for the technical community and Web for the general population. Our world is the former; we are a lot better off where it counts.

First Steps in Analyzing Data

Michael writes: “Data scientists still spend significant time ’cleaning’ or ’munging’ data. That is, taking it from a raw format, extracting the useful data, and reshaping it to a format that is appropriate for analysis. Despite the tremendous effort that goes into this activity little has been done to make this process easier and it is still generally performed ad-hoc.”

This is a good point. Surely some things can be done that have generality. But a good bit of processing and checking the data up front depends very heavily on the subject matter and the mechanism that generated the data, so a good bit of “ad-hoc” is here to stay.

The terms generally used for this first treatment of the data — data compression-reduction, provenance, wrangling, cleaning, munging, etc. — come from data mining. In this setting, the focus is on algorithms to vet the data. Visualization, however, should go on side-by-side with this, in part guided by these algorithms. The combination can be vastly more powerful. This is Tukey’s cognostics. Trelliscope, a D&R visualization framework that is discussed in Section 2, can do this for large complex data.

One up-front task that is important for all data, especially for large complex data, is data structures. Good choices up front can make programming with the data dramatically easier, improving the analyst performance. In our own analyses of large complex data we spend a lot of team time discussing the possibilities. The choices depend on the analysis tasking. Choices for large complex data get more complex because the data must be divided into many objects each with the same data structure. This can affect the performance of the computer, too, not just the analyst. The analysis tasking can change as a result of initial findings. Then comes the agonizing decision of whether to change the data structure. In addition, it is very wise when there are a number of analysis threads, to use different data structures if needed. Yes, even when the data are big. After all, it is the time of the analyst that we must protect at almost all costs.

Domain Specific Languages for Data Analysis Other Than R

Michael writes: “However, it is clear now that R is not the only programming environment in this space. The Python community has recently begun to focus on data science applications and has seen tremendous successes in adoption and integration with existing software platforms.”

As soon as there is something better than R we should all flock to it. We see this at the moment as a far off event. Just now computer scientists have started working on the design of new domain specific languages for data analysis. Some are working on speeding up R. Most see the critical matter as the computer speed at which the language interprets and runs. Faster is surely better, but as we have emphasized, is well behind analyst programming speed in importance.

The Areas of Data Science

Michael writes: “Cleveland ends his paper by identifying successful outcomes, including linking theory with data …. In many ways linking theory with data is an ongoing process. In part, this is because the iterative development of computational infrastructure and theoretical discovery is still underway. Also with new problem domains such as ’Big Data’, it is very much a moving target.”

Yes, the details of the work change with time, and big data are upon us, changing them a good bit. The 2001 paper says nothing about details, on purpose. It addresses broad areas of work whose union is all that must be considered in the analysis of datasets whether they are big, medium, or small. The areas apply now, applied in 2001, and applied in the 1960s when Ram Gnanadesikan, Martin Wilk, and John Tukey started the statistics department at Bell Labs and made it a data science department.

2 Divide and Recombine (D&R) With Tessera

2.1 D&R Overview

D&R is fundamentally a statistical approach to deep analysis of large complex data, but with an eye on computational performance, designed so that computation is mostly embarrassingly parallel, the simplest parallel computation [6].

In D&R, the data are divided into subsets by a statistical division method. The subsets are stored in objects with the same data structure, either on disk or in memory. Each of the analytic methods used in the analysis is applied to the subsets independently without communication among the computations, which makes them embarrassingly parallel. A statistical recombination method is applied to the outputs of the of each analytic method to form the final D&R result. The recombination does typically have some communication among the subset outputs through gathering results together, but often a significant portion of the computation can be done in parallel without communication. D&R computation is simple, and can readily exploit distributed parallel computational environments running on a cluster.

A division typically persists, and is used for many analytic methods. Recombination methods, however, tend to be tailored to individual analytic methods or to classes of them. For many large complex datasets it makes sense to have a few divisions in cases where different analysis tasks are best done with different data structures to increase programming efficiency, or computational efficiency, or both.

Statistical efficiency is important too. The statistical accuracy of D&R results depends strongly on the division and recombination methods. D&R research in statistical theory seeks methods that lead to high accuracy.

D&R research in computation has led to the Tessera D&R computational environment [1] with R at the front end [12]. All analyst programming is in R. At the back end is the Hadoop distributed file system (HDFS) and parallel compute engine (MapReduce) [14]. Hadoop runs the analyst’s R commands to carry out the D&R computations. Tessera software packages merge R and Hadoop, enabling the communication between the two R and Hadoop, and making programming D&R easy. Furthermore, the Hadoop scheduler enables fair sharing of a cluster by different data analysts in part by fair mingling of the analysts’ independent D&R micro-computations, that is, the embarrassingly parallel processes that are a part of the division and recombination tasks, and all of the analytic-method tasks.

2.2 D&R Outcomes

"Deep analysis" means the data are analyzed in detail at their finest granularity. In deep analysis we do not analyze just summary statistics or just the output of an automated data reduction algorithm or just random samples of the data; these practices carry with them a big risk of missing critical information in the data. Deep analysis includes starting an analysis with visualization of both detailed data at their finest granularity and summary statistics, and continuing visualization throughout the whole course of the analysis [7]. Deep analysis is done routinely today for many smaller datasets. The goal of D&R and Tessera is to extend deep analysis to much larger and more complex data.

To cover the wide range of types of data in need of deep analysis, analysts collectively need access to the 1000s of analytic methods of statistics, machine learning, and visualization. The R language at the front end of Tessera provides this.

R is very widely used because its elegant design makes programming with the data very efficient, yet is very powerful. It saves the analyst time. The D&R computational environment preserves the highly efficient programming because R is at the front end, and because the D&R software between R and Hadoop makes simple the specification of divisions, of applications of analytic methods to subsets, and of recombinations.

An important question is what datasets can benefit from D&R. The designation of a dataset as "large complex", based only on properties of the data, is elusive. At the end of this section we give it an operational foundation, if not a highly precise definition, by taking the notion of large and complex to be relative to the power of the computational environment that will be used for analysis. A very wide range of datasets in terms of degrees of large and complex, can benefit from D&R statistical methods and its computational environment, even data on the smaller side where computational power is low.

2.3 Analytic Methods: Number-Category and Visualization

In D&R, two types of analytic methods that are applied to subsets are treated differently. The first type is number-category methods. The outputs are numeric and categorical. Examples are logistic regression, fractional autoregressive moving-average models, Bayesian networks, random effects models, and non-parametric regression. The second is visualization methods. The outputs are plots. Examples are scatterplot matrices, time-frequency displays of time series, and many regression diagnostic displays.

Number-category methods are typically, in our own work, applied to all subsets because computation is feasible. Of course, it is possible to have so much data that computation time is too long, even though it is feasible.

For visualization methods, because the numbers of subsets are often in the many thousands or millions, it is typically not practical to look at and assess them all, so subsets are sampled. To display them we use the trellis display framework in which there are panels laid out in columns, rows, and pages; each panel shows the application of the visualization method to one subset [2, 4]. This is, in fact, a data reduction, but it can be made rigorous by statistical sampling [7]. We compute a number of variables, each with one value per subset, that are then used to define sampling plans that can be representative of the space of the sampling variables, or focused on a particular region. Furthermore, we can do this sequentially, following leads that arise from viewed displays to make new ones. Chosen display methods can show the data at their finest granularity. Of course, we plot subset summary statistics, too.

2.4 Two Categories of Division Methods: Conditioning-Variable and Replicate

In very many cases, it is natural to break up the data based on the subject matter, in a way that would be done if there data were small. So the division method is obvious. For example, suppose we have measurements of 100 daily financial variables for 10 years from 10^5 banks around the world. Then analyzing data first by bank is quite natural. This is conditioning-variable division; we break up the data by bank, a conditioning variable. There can be a multiple conditioning variables. Each can be categorical, or it can be numeric but turned into categorical by breaking the values up into intervals. Each subset consist of values of other variables corresponding to one combination of the categories of the conditioning variables. The number of subsets is the product of the numbers of categories of the conditioning variables.

This concept of conditioning-variable division already exists in statistics in many different contexts, and is very widely used. It is a statistical best practice. Conditioning-variable division is the foundation for hierarchical models, which are widely used, for example, to model random effects, random coefficients, repeated measures, etc. Conditioning-variable division is the foundation for the trellis display visualization framework, implemented in R as lattice graphics [13]. In the S language, the parent of R, there was a suite of `apply` functions at the outset in 1975-6 that applied the same expressions specified by the user to a division of the data, also specified by the user.

In replicate division, observations are seen as exchangeable, with no conditioning variables considered. Suppose we are doing logistic regression with $n = 2^{30}$ observations of $p = 2^7 - 1$ explanatory variables and one response consisting of 0's and 1's. There are $v = p + 1 = 2^7$ variables altogether. Suppose we take each subset to be m observations, where $r = n/m$ is an integer, the number of subsets. So we have r logistic regressions each with p estimates of regression coefficients and with associated statistical information. The next step is a recombination that results in a single vector of p estimates of the regression coefficients, and associated statistical information.

2.5 Analytic Recombination

For conditioning-variable division, the recombination to a large degree depends on the analysis goals and the subject matter. Often there is further analysis of the outputs. We call this an "analytic recombination". This happens in hierarchical modeling. For example suppose conditioning variables break the data into subsets, and for each subset the analytic method is logistic regression. The "analytic recombination" might be the building of a statistical model for the coefficients across subsets using the subset estimates and accompanying statistical information.

2.6 Statistical Accuracy

For replicate division and recombination, D&R estimates are a replacement for the estimates we could have gotten had it been feasible, and practical in terms of computation time, to apply the analytic method directly to all of the data. The D&R result typically has less statistical accuracy than the direct all-data result. There are many potential methods of division and methods of recombination. For example, for the above logistic regression example, we could use random sampling to chose subsets and take the vector means of the r subset coefficient estimates to get the D&R estimates. The division and recombination methods have an immense impact on the statistical accuracy of D&R estimation. One critical research thread of D&R is statistical theory to find "best" final D&R results [6]. For example, random division and mean recombination of the coefficients, while very easy, are unlikely to be the best in general for logistic regression.

There are two key points here. Work has so far has shown that loss in accuracy can be a small penalty to pay for the very simple, fast computation. Second, the recombination method also needs to include information about the statistical variability of the D&R result.

2.7 Tessera D&R Computation: R Front End

R is an easy choice for the front end of the D&R computational environment. Its elegant design makes programming with the data very efficient. In other words, it saves the analyst time, which is more valuable than computer time, and therefore deserves to be first in the list of time priorities. R is also very powerful, allowing the analyst to readily tailor the analysis to the data.

R brings other things to the table. It is open source and free. This has attracted a large supporting community consisting of the core development team, and package contributors. R has a vast number of analytic methods of statistics, machine learning, and visualization. Its core distribution has many analytic methods. CRAN, the R contributed package repository, has 5835 packages as of 1Sep14. There are many more many more in other repositories such as R-Forge, GitHub, and Bioconductor. R is the place to deposit code to get wide usage of a new method, model, or algorithm. R is the place to look if you need to use some new method, model, or algorithm. The R user community is surely vast, although no one knows for sure its size with precision because it is open source. Still, R computation time is slower than one might like, and there are a number of efforts now to speed it up.

There are new competitors emerging such as Python [11] and Julia [3], meant to run more efficiently in terms of computer time. They can succeed if they duplicate the analyst ease of programming and saving of time, and make available the thousands of of analytic methods found in R. This has yet to happen, and is unlikely to do so soon.

2.8 Tessera D&R Computation: Hadoop Back End

Hadoop, with its HDFS and MapReduce, is also an easy choice for the back end running on a linux cluster. It is widely distributed. The analyst specifies R code to divide the data into subsets. Hadoop runs the R code to divide the data, form subset R objects, and write the objects to the HDFS, spreading the objects across the nodes of the cluster. The analyst specifies the R code for the analytic method to be applied to each subset. Hadoop reads the subsets from the HDFS, applies the R code to each subset to get output, forms output R objects, and writes the objects to the HDFS. When the computational power of Hadoop is needed for the recombination, Hadoop applies the analyst R code for the recombination to the outputs. This is not needed when the outputs are small data, as they often are. In this case it can be more convenient for the user to read the output R objects from the HDFS, write them to the R global environment, and then carry out the recombination there in the standard serial way.

2.9 Tessera R Packages in the Middle: datadr, Trelliscope, and RHIPE

Tessera software middle layers are between the front and back ends. They (1) enable communication between R and Hadoop; (2) make specification of division and recombination methods easy; and (3) enable visualization of the data at their finest granularity.

RHIPE is the R and Hadoop Integrated Programming Environment [5, 6, 1]. In R, the analyst passes the R code for division methods, analytic methods, and recombination methods to RHIPE R routines that communicate with Hadoop. The analyst specifies MapReduce operations to operate on R data objects stored on the HDFS.

The datadr R package is a layer between R and RHIPE that makes the programming even easier in certain standard circumstances [8]. Instead of having to specify division and recombination methods in terms of MapReduce operations directly through RHIPE, datadr provides a simple interface for specifying divisions and recombinations as single R commands that are more natural to the analyst. The datadr package also provides a more natural way to deal with data stored on the HDFS. Distributed data sets in datadr are represented generically as distributed data objects in R such as distributed data frames. The analyst can treat these objects in many ways just like they treat native R data objects.

The trellis display visualization framework implemented in R by the lattice graphics package, also divides the data. A visualization method is applied to each subset and shown on one panel of a multi-panel trellis display. This framework is a very powerful mechanism for all data, big and small. Trelliscope, a layer side-by-side with datadr, extends trellis display to large complex data [10, 9]. Display panels have sampled subsets from rigorous sampling plans as described above.

RHIPE, datadr, and Trelliscope are open source, as are Linux, Hadoop, and R, making Tessera — front, middle, and back — open source at a cost of $0.

2.10 Tessera D&R Computation: More on Hadoop

The two principal computational operations of Hadoop are Map and Reduce. The first runs parallel computations on subsets without communication among them. The second can compute across subset outputs. So Map carries out the analytic method computation. Reduce takes the outputs from Map and runs the recombination computation. A division is typically carried out both by Map and Reduce, sometimes each used several times, and can occur as part of the reading of the data into R at the start of the analysis.

Usage of Map and Reduce involves the critical Hadoop element of key-value pairs. We give one instance here. The Map operation, instructed by the analyst R code, puts a key on each subset output. This forms a key-value pair with the output as the value. Each output can have a unique key, or each key can be given to many outputs, or all outputs can have the same key. When Reduce is given the Map outputs, it assembles the key-value pairs by key, which forms groups, and then the R recombination code is applied to the values of each group independently; so the running of the code on the different groups is embarrassingly parallel. This framework provides substantial flexibility for the recombination method.

Hadoop attempts to optimize computation in a number of ways. One example is Map. Typically, there are vastly more subsets than cores on the cluster. For example, next, we will run the above example of logistic regression with $r = 2^{20}$ subsets on a cluster with 242 cores for Hadoop. When Map finishes the application of the analytic method to a subset on a core, Hadoop seeks to assign a subset on the same node as the core to avoid transmission of the subset across the network connecting the nodes, which is more time consuming.

Hadoop also provides another critical scheduling service: enabling sharing of a cluster by different analysts. Hadoop offers up many ways to do this. One is “fair” scheduling. Suppose User 1 is the only one using the cluster and has all cores running a Hadoop job, doing the ”micro-computations”, which are largely the the Map and Reduce computations. Suppose User 2 comes in and issues a Hadoop job command. When a core running a User 1 micro-computation ends, Hadoop assigns the core to User 2 for use, and keeps doing this until User 1 and User 2 have about the same number of cores, and then maintains this 50-50 balance. If another analyst job arrives, it carries out a similar balancing.

This Hadoop job scheduling contrasts with the very common high-performance computing scheduling that serves massive simulation or modeling computations; in this case a user books a block of time and a number of processors, so the scheduling is just a batch processing of different jobs, and each has to wait in a queue. This does not serve well deep data analysis which cannot be carried out by isolated single batch jobs. A deep analysis needs to be step-by-step, with very many steps, each where a number-category or visualization method is applied. The decision at each step needs to be based on all previous steps. Merging micro-computations to share servers the analysis model well.

2.11 How Fast is D&R with Tessera?

We will use the above logistic regression example and run it on a cluster to show performance. There are $2^{30} \times 2^{7} = 2^{37}$ values in the data. We make them all numeric, so each value is double precision and takes 2^{3} bytes in memory. The memory size is then 2^{40} bytes, a terabyte.

The Rosen cluster, used exclusively by our D&R research project, is maintained by system administrators at the Rosen Center for Advanced Computing at Purdue University. It consists of two sets of nodes. One set of three nodes runs separately Hadoop NameNode, JobTracker, and SecondaryNameNode. These are the Hadoop systems that manage the HDFS and MapReduce scheduling. The other set has 11 nodes, each a Hewlett Packard ProLiant DL165 G7. Each runs DataNode and TaskTracker, which are the Hadoop systems the run the MapReduce computations and manage the HDFS. Each of these 11 nodes has

- Dual 2.1 GHz 12-core AMD Opteron 6172 processors (24 cores)
- 22 cores for Hadoop computations and 2 for the work in the R global environment
- 48 GB RAM
- 4x2 TB 7200 RPM SATA disks
- 10 Gbps Ethernet interconnect.

Collectively, the Rosen cluster has

- $24 \times 11 = 264$ cores
- $22 \times 11 = 242$ Hadoop cores
- $48 \times 11 = 528$ GB total RAM
- $8 \times 11 = 88$ TB total disk.

This is by today's standards a very modest cluster. The disk is not particularly fast, a financial decision to have more disk with less speed. Note that the memory size of the data is about twice that of the actual memory.

We ran the above logistic regression for the terabyte of data using the R routine `glm.fit`. The number of subsets was 2^{20}. The data were artificially generated, subsets formed, and then written to the HDFS as R objects. We then measured the elapsed times associated with reading the subsets from the HDFS, applying `glm.fit` to the subsets, carrying out the recombination which was taking the means of the 2^{20} subset estimates, and then writing the means to the HDFS. The total elapsed time of these computations was 17.6 min. However, 70% of this time was the first step of simply reading the subsets once into memory with R and making the R objects available to `glm.fit`. The logistic regression elapsed time, without the reads, was 5.3 min. Buying much faster disk would speed up the total elapsed time, but there is no line item for this in the Tessera budget at the moment.

2.12 The Applicability of D&R with Tessera is Very Wide

The notion of the size-complexity of a dataset as a concept is very difficult to make precise in isolation, and only takes meaning in practice relative to the hardware-software power of a cluster considered for its analysis. So we condition on the cluster and its software.

If on the cluster, computation, say serial computation, is impractical because it takes many weeks, or is simply infeasible, then the size-complexity is too high for the hardware power of the cluster.

Tessera can be installed on clusters ranging from the very low end, with small hardware power, to the very high end, with very high power. Tessera installed on a cluster, wherever it lies on the hardware power spectrum, can very substantially increase the size-complexity of data that can be feasibly and practically analyzed. Of course, for each cluster, there is a limit even with Tessera. That limit, of course, increases with the hardware power. But the critical point is this. Tessera can help in terms of analysis success from the low end to the high end. Tessera is not just a computational environment for massively large-complex data. Rather it is an environment allowing larger and more complex datasets on the available hardware to be analyzed. What is achieved in terms of increases in what is learned about the subject matter under study can be very substantial indeed across the entire range of the hardware power spectrum and the size-complexity of the data analyzed.

References

[1] The Tessera D&R Computational Environment, 2014. World Wide Web, http://www.tesseradata.org.

[2] R. A. Becker, W. S. Cleveland, and M. J. Shyu. The Design and Control of Trellis Display. *Journal of Computational and Statistical Graphics*, 5:123–155, 1996.

[3] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *CoRR*, abs/1209.5145, 2012.

[4] M. Fuentes, B. Xi, and W. S. Cleveland. Trellis Display for Modeling Data from Designed Experiments. *Statistical Analysis and Data Mining*, 4:133–145, 2011.

[5] S. Guha. *Computing Environment for the Statistical Analysis of Large and Complex Data*. PhD thesis, Purdue University Department of Statistics, 2010.

[6] S. Guha, R. Hafen, J. Xia J. Rounds, J. Li, B. Xi, and W. S. Cleveland. Large Complex Data: Divide and Recombine (D&R) with RHIPE. *Stat*, 1:53–67, 2012.

[7] S. Guha, R. P. Hafen, P. Kidwell, and Cleveland W. S. Visualization Databases for the Analysis of Large Complex Datasets. *Journal of Machine Learning Research*, 5:193–200, 2009.

[8] R. Hafen. “datadr” github documentation. http://www.tesseradata.org, 2014.

[9] R. Hafen. Trelliscope github documentation. http://www.tesseradata.org, 2014.

[10] R. P. Hafen, L. Gosink, W. S. Cleveland, J. McDermott, K. Rodland, , and K. Kleese-Van Dam. Trelliscope: A System for Detailed Visualization in the Deep Analysis of Large Complex Data. In *Proceedings, IEEE Symposium on Large-Scale Data Analysis and Visualization*, 2013.

[11] Wes McKinney. *Python for data analysis : [agile tools for real-world data]*. O’Reilly, Sebastopol, CA, 2013.

[12] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[13] D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008.

[14] T. White. *Hadoop: The Definitive Guide, Second Edition*. O’Reilly, Sebastopol, CA, 2011.