FOUNDATIONS OF A CROSS-DISCIPLINARY PEDAGOGY FOR BIG DATA

Joshua Eckroth
Stetson University
DeLand, Florida
386-740-2519
jeckroth@stetson.edu

ABSTRACT

The increasing awareness of "big data" is transforming the academic and business landscape across many disciplines. Yet, big data programming environments are still too complex for non-programmers to utilize. To our knowledge, only computer scientists are ever exposed to big data processing concepts and tools in undergraduate education. Furthermore, non-computer scientists may lack sufficient common ground with computer scientists to explain their specialized big data processing needs. In order to bridge this gap and enhance collaboration among persons with big data processing needs and persons who are trained in programming and system building, we propose the foundations of a cross-disciplinary pedagogy that exposes big data processing paradigms and design decisions at an abstract level. With these tools, students and experts from different disciplines can more effectively collaborate on solving big data problems.

1. INTRODUCTION

Data is growing at an exponential rate. This growth brings new opportunities for data mining and analysis, but also brings significant challenges in data storage and processing. Many fields of study, including business intelligence and analytics [4], health care [8], and social science [6], are beginning to explore how to make use of big data. However, fundamental processing paradigms for big data, such as parallel computation, are difficult for first-year computer science students to master [3], and we expect that students in other disciplines with little computer science background have at least as much difficulty. Additionally, complex programming is normally required to build and execute a big data processing job. This leaves non-computer scientists at a disadvantage and possibly uncertain about how to design a big data processing task and communicate this design to experts who are capable of programming and executing the design.

As computer scientists and educators, we have the requisite background knowledge for understanding big data processing and we have experience teaching complex computer science subjects. To our knowledge, few courses outside of computer science departments expose students to parallel and distributed processing techniques. Typically, non-computer science students are taught how to use Microsoft Excel, SPSS, STATA, and R to process data on a single machine. These techniques simply do not work on big data, though they are useful for analyzing data that result from the aggregation and summarization of big data. This work gives the foundations of a pedagogy for big data for non-computer scientists and computer scientists alike.

"Big data" has a variety of definitions. Our definition is adapted from Russom [9]. It makes use of three other terms: *data volume*, which represents the size (perhaps in bytes) of the data; *data velocity*, which represents the speed (perhaps in bytes/second) at which data is arriving; and *commodity machine*, which identifies a computer or virtual machine in a cloud computing service that has moderate specifications and cost. Commodity machines in 2015 vaguely are capable of storing 5-10 terabytes and have up to 32 GB RAM, more-or-less. Most importantly, commodity machines are not supercomputers. We define big data as stated below.

> *A data mining/analysis task may be described as "big data" if the data to be processed have such high volume or velocity that* ***more than one*** *commodity machine is required to store and/or process the data.*

The key aspect of this definition is the highlighted phrase: "more than one." If the data fit on one commodity machine, or can be processed in real time by one commodity machine, then big data processing tools are not required and, in fact, just introduce additional overhead. Identifying what is and is not big data is a key skill for students and practitioners so that they may avoid introducing complexity and overhead where it is not necessary. The big data paradigms we present below support processing and data storage operations that span several machines. Breaking up a computation across several machines is non-trivial, and is the sole reason big data paradigms are non-intuitive and should be taught with paradigmatic cases.

The main contributions of this paper are: (1) an outline for a pedagogy for big data; (2) big data paradigms that cover a wide range of tasks and give a common language for collaborating; (3) a decision tree for helping decide which paradigm to apply in a particular situation.

The rest of this paper organized as follows. Section 2 discusses our goal for big data pedagogy. Section 3 identifies three processing paradigms and demonstrates directed computation graphs. This section also gives a decision tree for helping students and practitioners decide which paradigm is appropriate for a given situation. Section 4 walks through a concrete example. Finally, Section 5 explores related work and Section 6 offers conclusive remarks.

2. BIG DATA PEDAGOGY

Our goal in developing a cross-disciplinary pedagogy for big data is to teach students from a variety of disciplines, including computer science among others, about the design of big data processing tasks. We expect non-computer science students to generate artifacts that abstractly illustrate a processing design in the form of directed computation graphs (demonstrated in the figures below). We expect computer science students would also generate such computation graphs but may also be asked to actually implement and test the design on a cluster of machines.

A big data pedagogy should explain and continually reinforce the following general design process:

- Identify a question to ask about the data. Different disciplines may be concerned with very different kinds of questions.
- Determine data sources and their volume and velocity.
- Identify how many commodity machines may be needed to store and process the data. The outcome of this step serves as constraints for future steps.
- Identify a data sink for the output data. Normally, the output data is very small in volume and/or velocity compared to the input data.
- Decide if "big data" processing techniques are necessary. If not, choose a traditional tool (e.g., Microsoft Excel, Stata, etc.). The decision tree at the bottom of Section 3 may help.
- Design a directed computation graph by adapting a big data paradigm from Section 3, as indicated by the decision tree.
- (Possibly) implement the solution. Programming expertise may be required for this final step in the process.

Our learning objectives are as follows. Students from various disciplines will be able to,

- LO1: determine whether a data processing task requires "big data" tools and techniques, or can be done with traditional methods;
- LO2: identify appropriate big data paradigms to solve the problem;
- LO3: design an abstract representation, in the form of a directed graph, of the required processing steps and identify input and output formats of each step.

In the following sections, we introduce paradigms that serve as the foundation of the pedagogy. We then illustrate one of these paradigms with an example.

3. BIG DATA PARADIGMS

We have identified three big data paradigms that cover a wide range of big data processing tasks. The first two are adapted from Kumar [7]. These paradigms are illustrated with directed computation graphs, which indicate processing steps along with data sources and sinks, identified by arrows on the left and right (respectively) of each processing step. Figure 1 shows a processing step. The key concept of a processing step is that the machine processes only those data that are already stored on the machine's disk and/or data that are made available as inputs from other steps or an external data source. Machines do not communicate with each other except as indicated by the arrows. Furthermore, each machine is assumed to be a commodity machine with modest computational and storage capabilities. Thus, no single machine is able to support a big data processing task on its own. The challenge for students is to design a solution to a big data processing problem that splits the task into distinct processing steps with appropriate inputs and outputs, while limiting communication among machines and exploiting parallel processing.

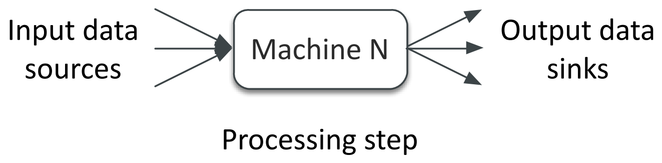

Figure 1: A processing step for directed computation graphs.

Processing High Volume Data

The fundamental challenge with high volume data is that the data are initially split and stored across several machines. Thus, an efficient processing strategy would task each machine with processing the data that are already stored on that machine, and only those data. The Map-Reduce architecture, commonly used with the Hadoop platform [1], breaks the task into two distinct phases. During the *Map* phase, each machine that holds a subset of the data is tasked with processing that subset. A Map-Reduce coordinator (which typically runs on yet another machine) waits until each Map job is complete for all the machines, and then delivers the results from each Map job to machines that are tasked with aggregating the results (perhaps reusing machines from the Map jobs). This second *Reduce* phase is provided the results of the Map jobs and is responsible for producing the final output of the entire processing task. Figure 2 shows an example directed computation graph for the Map-Reduce architecture. A concrete Map-Reduce processing example is detailed later.

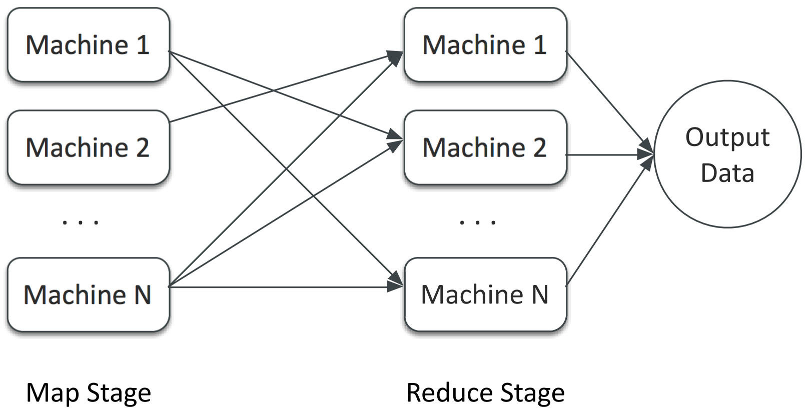

Figure 2: High volume, batch processing paradigm.

Processing High Velocity Data

The fundamental challenge with high velocity data is that the data are arriving faster than any one machine can process. Thus, the processing must be handled by several machines. The data are not necessarily stored on any machine, though such configurations are possible. A more common design challenge for high velocity data processing is deciding which machine performs which subtask and how these subtasks (and therefore machines) coordinate. The directed computation graph indicates how the data are “streamed” through various transformations and aggregations. No particular computation graph is more common than any other for stream processing. Its structure depends entirely on the task. This makes high velocity stream processing more challenging for students and instructors alike, because it is less constrained, and should be taught after introducing the more constrained Map-Reduce architecture. Figure 3 shows one such example of a directed computation graph for the stream processing paradigm.

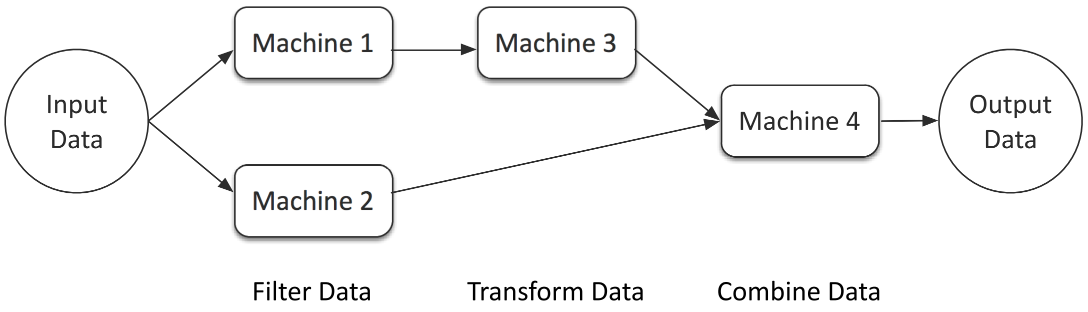

Figure 3: High velocity, stream processing paradigm.

Merging Data

When two or more datasets must be merged within a big data processing task, one is faced with either of two scenarios. In the first scenario, the data to be merged are “small” enough to fit in memory or disk on a single machine. In this case, the merge may be performed by each machine that needs the data. For example, in a Map-Reduce paradigm, the merge may be performed during the Map stage. On the other hand, if the data to be merged is big data, then no single machine is capable of performing the merge, so the merge must occur piece-wise across many machines. In the Map-Reduce paradigm, such a merge would occur during the Reduce stage. The details of merging are outside the scope of this report, but we note that merging data is a common task and should be explored in a big data curriculum.

Decision Tree

We have described three paradigms for big data tasks. When designing a solution to a problem, students should evaluate first whether their problem requires big data processing, and if so, which paradigm to apply. The following decision tree may be offered as a tool to aid this design challenge.

1. Can the data be processed in R, SPSS, SAS, Stata, etc.?
    - 1.1. Yes: Avoid big data technologies, use traditional methods.
    - 1.2. No: Are the data high volume or high velocity?
        - 1.2.1. High volume: Does more than one data set need to be merged?
            - 1.2.1.1. Yes: Is more than one data set "big data"?
                - 1.2.1.1.1. Yes: Design a join in the Reduce stage of Map-Reduce.
                - 1.2.1.1.2. No: Design a join in the Map stage of Map-Reduce.
            - 1.2.1.2. No: Design a simple Map-Reduce processing job.
        - 1.2.2. High velocity: Decompose the processing into transform/filter stages.
        - 1.2.3. Both high volume and velocity: Design a streaming processing job that saves the resulting data in splits across multiple machines. Then process the saved data with Map-Reduce.

4. CONCRETE EXAMPLE

We now develop a concrete example that is appropriate for a first exploration of the batch processing high-volume paradigm. The task is to find the maximum value of a single-column numeric data set with one trillion records, stored across several machines.

First, we look at some bad ideas. These include,

- Use Excel. This is a bad idea because the data are too voluminous to be stored on a single machine.
- Read through all the values on machine 1 and record the maximum. Then read all the values on machine 2 and record the maximum. And so on for all the machines with data. Then find the maximum of all the stored maximums from the subsets. This is a bad idea because the procedure does not exploit parallelism. The maximum on each subset of data may be found simultaneously since the machines need not communicate with each other.

A good idea would be to start with the decision tree. Based on the decision tree, we conclude that the task is a big data task and it is high volume. There is only one data set so no merge is required. Thus, we should design a Map-Reduce job. Each machine would perform a Map stage, in parallel. During the Map stage, the maximum value for the data stored on that machine is found and communicated to the Map-Reduce manager. Once all the machines are finished finding their subset maximum, the maximum of this collection of maximums is found (by yet another machine or by reusing a machine).

Figure 4 shows the directed graph of computation for this task. Notice that it is a specialization of the high volume, batch processing paradigm (Figure 2).

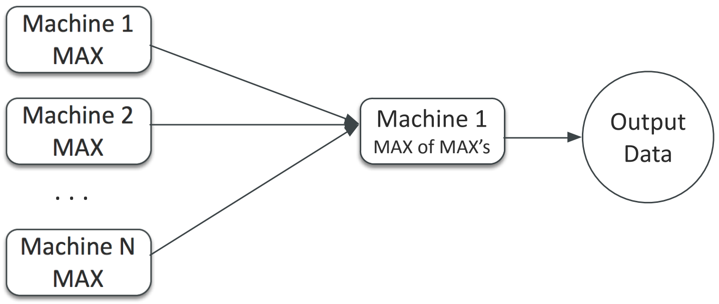

Figure 4: Directed graph of computation for the "maximum value" example.

These paradigms and the corresponding decision tree form the foundation of our pedagogy for big data. The paradigms serve as an abstract representation of big data processing tasks that students from a wide variety of disciplines should be able to understand and design. The decision tree assists in the design of big data jobs and may be further elaborated as more paradigms are introduced and specialized.

5. RELATED WORK

According to our research, there are no published reports documenting attempts to teach non-computer science students how to design big data processing tasks. Silva et al. [10] explored pedagogy for computer science students. They identified different categories of big database management systems (BDMS) for splitting big data across several machines, and developed guidelines for deciding which to use in various situations. They designed big data processing exercises for each BDMS. These exercises required strong programming skills. They found that diagrams, similar to the diagrams presented above, helped students understand big data processing steps.

Dobre and Xhafa [5] explore a wide variety of big data paradigms and tools. Their report is more detailed and extensive than we consider appropriate for non-computer scientists. However, it may help instructors expand the big data paradigms presented here to cover more subtle and sophisticated techniques. One such technique is Map-Reduce-Merge, which adds a Merge step after the normal Map-Reduce. This Merge step can combine the results of multiple parallel Map-Reduce jobs that operate on and produce heterogeneous data. They also explore the history of big data processing and survey a wide variety of software tools for actually programming and building big data jobs.

6. CONCLUSIONS

This work introduced foundations for a big data pedagogy that is appropriate for students across various disciplines. We introduced a common language that may serve as the basis for collaboration. This language includes terms like big data, data volume, data velocity, batch processing / Map-Reduce, stream processing, and merging. We exposed three paradigms of big data processing and provided a decision tree for making sense of when to use each paradigm.

We hope that researchers and developers will continue to expand the big data ecosystem. In particular, we hope that the tools are simplified to the point that non-computer scientists may implement and execute big data jobs without assistance from big data experts. We believe that a simplified programming language such as Pig Latin, from the Apache Pig project [2], may be a step in the right direction.

ACKNOWLEDGEMENTS

We wish to thank the anonymous reviewers for their helpful feedback.

REFERENCES

[1] Apache Hadoop, The Apache Foundation. https://hadoop.apache.org, 2014.

[2] Apache Pig, The Apache Foundation. https://pig.apache.org, 2014.

[3] Bogaerts, S. A. Limited time and experience: Parallelism in CS1. *IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPSW)*, 1071-1078, 2014.

[4] H. Chen, R. H. Chiang, and V. C. Storey. Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, 36(4):1165-1188, 2012.

[5] C. Dobre and Xhafa, F. Parallel programming paradigms and frameworks in big data era. *International Journal of Parallel Programming*, 42:710-738, 2014.

[6] S. González-Bailón. Social science in the era of big data. *Policy & Internet*, 5(2):147-160, 2013.

[7] R. Kumar. Two computational paradigm for big data. *KDD Summer School on Mining the Big Data (ACM SIGKDD),* 2012.

[8] T. B. Murdoch and A. S. Detsky. The inevitable application of big data to health care. *JAMA*, 309(13):1351-1352, 2013.

[9] P. Russom. Big data analytics. *TDWI Best Practices Report, Fourth Quarter*, 2011.

[10] Y. N. Silva, Dietrich, S. W., Reed, J. M., and Tsosie, L. M. Integrating big data into the computing curricula. *SIGCSE*, 139-144, 2014.