Parallel Distributed Data Mining Solutions

Jay Cheatham, Nicholas Ferry, Dick Herrin

Abstract

The exponential growth of data being generated at and beyond the edge of the network continues to complicate the ability to perform data mining processes on large volumes of data generated thousands of miles away. The concept of moving data mining activities closer to where the data is being generated in becoming a practical option with the maturation of edge computing platforms that may be more proximate to the data. This offers the opportunity for organizations generating large amounts of data in different locations to take advantage of the approach called Distributed Data Mining, where individual sites can mine their respective data locally for the local value it represents and then forwarding much lower volumes of summarized or refined data to a central location to be consolidated with similar data from other sites for further mining [1]. Opportunities exist at individual locations to significantly enhance their data preparation and mining processes by restructuring their solutions to use parallel processing techniques in both the data preparation and data mining processes. This can also be applied to the structure and execution of the processing of the discrete data sets. We perform a proof of concept demonstrating the practicality of this approach.

1 Introduction

For our graduate level project for High Performance Computing (CSCI B569), we are taking open source public data to construct a model to make predictions through a data mining algorithm. To do this, we will be evaluating numerous models to determine the best algorithm. We will implement parallel code in a Python environment. Our group is planning on using an open source parallel programming Python benchmark called SST to analyze the performance of our constructed parallel code. To generalize what our objective is, we plan on finding serial code for this project that we can use to benchmark the tests individually and then parallelizing the serial code to attempt to run the algorithms in a more efficient manner that will lower our cost and help us reserve computing power.

2 Developing Trends

In the most recent Annual Internet Report (2018-2023), Cisco stated that in 2018, there were 18.4 billion devices connected to IP networks and that this number will grow to 29.3 billion in 2023. That is an increase from 2.4 networked devices per global capita in 2018 to 3.6 networked devices in 2023. Of these devices, Cisco forecasts that 14.7 billion will be Machine-to-Machine (M2M) connections by 2023, approximately 50 percent of the total. Cisco further projects that a growing share of these devices will be mobile and includes a rapidly expanding share of smartphone-hosted applications. Together, these devices move a lot of data every year.[1] One estimate, also attributed to Cisco, suggests that in 2019 these devices generated more than 500 zettabytes (for perspective, if the footprint of one person was one kilobyte, one zettabyte would cover all of Europe, slightly larger than the US) and is expected to grow exponentially year on year. No matter how one might consider this explosion of data and the devices that create, move, process, and store it, this has to be mind boggling.

2.1 Internet of Things

The fuel propelling this growth is the Internet of Things (IoT) and encompasses the billions of devices worldwide connecting directly or indirectly to the Internet. These devices communicate machine to machine. With the advent of increasingly smaller and more capable microchip sets in the late 1990s, industrial innovators began to focus on the development of digitally-enabled sensors and controllers to replace their long-established analog predecessors wherever they might be. One of the primary defining characteristics of most IoT devices is that they generate real-time data, most times, lots of it.

Organizations are investing great sums of money to deploy IoT devices for a variety of reasons and motivations. Central to these are the desire to gain greater understanding of their operations derived from appropriately timed and scaled analytical processes embedded in data value life cycles. From these understandings, their respective aims are to save money, improve efficiency, optimize operations, discover nuances of their business they didn't know before.

2.2 Edge Computing

Data is being collected and stored at an increasingly accelerated rate. Advancements in analytical computing technologies are enabling more robust understanding of data properties, algorithms, and processes in solving ever more complex problems. Further,

the creative deployment of such tools, sets the stage for the pursuit of new problems of substantially higher complexity than ever before. However, certain intrinsic challenges impede the rate progress towards such ambitions.

- The stores of data are vast, varied, and hosted in a myriad of places and formats.
- Seemingly similar data is often not so similar and may have few attributes or properties in common requiring a great deal of refinement before it can be used.
- The volumes of data that must be evaluated are often immense, driving demand for more scale-able, higher-performing, computing platforms, and processing approaches.
- It can be expensive and time-consuming to move high volumes of data great distances for processing and storage, creating high latency and making real-time analysis and decision making impractical when such is a necessity.
- There is increasing pressure to find ways to select, process, and analyze targeted data faster and closer to real-time when requirements so dictate.

Clearly, the solution to these challenges is to move computing and storage capacity and capabilities as close as possible to the edge of the network to address these needs. One of the primary benefits of cloud computing is its economy of scale and its abilities to quickly adapt to the requirements and volume demands placed on its customers. It seems obvious that a reasonable option might be building more centers closer to the outer edges of the collective cloud(s), but where exactly might that be? And doing so while maintaining their ability to quickly scale and serve client demands might be cost prohibitive and still may not solve the bandwidth and latency issues for many of their potential customers.

However, in the decade or so that cloud computing as evolved as an essential part of the Internet and overall computing landscape, three key technologies have evolved that are essential to enabling stronger, localized solutions at the edge of the network: Software Defined Networking (SDN), Network Function Virtualization (NPV), and Virtual Machines. Among others, these technologies are what gives cloud data centers their adaptive flexibility and some of their cost effectiveness. These technologies can be deployed cost-effectively in smaller scale and are becoming critical facets of localized edge computing facilities, that can scale and adapt as needed to accommodate the volumes of incoming data and the applications needed to process that data.

2.3 Localized Data Mining

With the availability of viable local edge computing platforms, two particular disciplines that are key to addressing these computational challenges are, the broader development and deployment of more portable and scalable parallel computing capabilities, and the localization of discrete data mining capabilities. Parallel computing has increasingly become more centralized in increasingly denser computing platforms, while data mining has increasingly been pushed nearer to, and tuned to, the particular characteristics and volume of the targeted data [2]. Data mining has traditionally not imposed the level of computing demand intensity of other data-related disciplines. However, with the present and future challenges with which practitioners must cope, they are looking for more opportunities to apply parallel computing to their work as it becomes more localized and less centralized and monolithic.

The context for this project was to focus more on the parallel distributed data mining solutions or PDDMS. There are at least two contexts for overall concept. The first relates to opportunities to take a variety of potentially related data sets, and crafting solutions that allow the work for these discrete data sets to be done concurrently and further optimized through parallel processing methods. This is what we plan to explore here.

The other context of PDDMS relates to the idea that instances of the first context may be located elsewhere in multiple similar types of locations (e.g. one of many factories owned by the same company) [4]. Each then can process its own data using consistent methods and processes to address all items of local need and interest. Each then can develop their own respective refined versions of data sets that can be consolidated for further analysis and processing more centrally as the volume and time requirements dictate.

3 Conceptual Overview

With these technologies and capabilities available at the edge of the network, we set out to explore the concepts and mechanics of how an integrated parallel distributed data mining solution (PDDMS) might look and function, and structured our project to prove the viability of such an idea. We planned to focus on one local instance of such a solution, but expect that the same could be done in other local instances and then consolidated at an appropriate point in time and process. We will not do that here. We envision a applying a model similar to that shone in fig 1 :Data Mining Algorithm Flow. We plan

to develop three discrete instances of distributed data mining modules, each focused on collecting specific data, related to, but different from the others. Each PDDMS Mining instance will run concurrently with the others, and be responsible for collecting, munging, analyzing, categorizing, and rationalizing its data using parallel computing methods wherever possible to accelerate processing times [3].

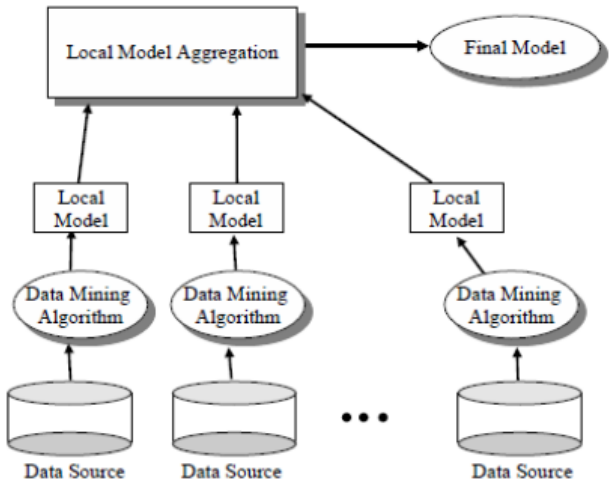

Figure 1: Chikhale, Rupali "Study of Distributed Data Mining Algorithm and Trends" IOSR Journal of Computer Engineering (IOSR-JCE)

When all of the PDDMS Mining instances have completed their processing and data preparation, a PDDMS Consolidator instance will be run to bring all of the mining data together, consolidate it, rationalize it, correlate it, and then analyze it to identify what can be learned from it, and determine what predictions can be made from it. Internal processes will be structured to use parallel methods wherever possible. The final instance will be a PDDMS Reporter, which will provide results reporting and visualizations–if and where it may be appropriate.

4 Methods

To achieve our goal of running multiple PDDMS Mining instances, we took multiple data sets based on mortality rates with various conditions and use them to predict the

morality rate of COIVD-19 by running them through multiple prediction algorithms once. To achieve this, the group complied a Python script to run the prediction algorithms on a single core through serial code. Once we were able to get the algorithms running on a single core, we took the serial code and parallelized it to attempt to make the original serial code run faster while lowering the cost and power needed for the tests as well as run multiple instances at once.

4.1 Data Used for This Project

For the purposes of our proof of concept, we chose to attempt to compare the COVID-19 mortality rates in the United States to mortality data we found for other significant diseases. Each data set had its own particular characteristics and would require significant munging and refining to a point where it could be compared with the other data sets. While each of the data sets have different attributes, we were still able to use them for this project, as we only compared the mortality rates and how those attributes contributed to the deaths records, not the attributes themselves.

4.2 Statistical Mining Methods

4.2.1 Support Vector Regression

The Support Vector Regression Algorithm is a machine learning tool that deals with classification and regression and focuses on kernel functions that allow for us to utilize more than one dimension. Using this algorithm should allow us to use our data set as a training set for keeping the *x* value as flat as possible while finding a function that can be based off of the *y* value. The algorithm finds the best fitting line that separates the two types of classifications that can be made within a margin of error; the margin of error is visually represented as supporting lines on either side of the line. The line allows for the algorithm to determine the best possible classification based on which side of the line that the prediction ends up falling on.

Cost Function Formula

$$J(w) = \frac{1}{N}\sum_{i}^{n}[\frac{1}{2}||w||^2 + Cmax(0, 1 - y_i * (w * x_i))] \tag{1}$$

Gradient of Cost Formula

$$\nabla_w J(w) = \begin{cases} w & \text{if } max(0, 1 - y_i * (w * x_i)) = 0 \\ w - Cy_i x_i & \text{otherwise} \end{cases} \tag{2}$$

Rate of Change For Gradient

$$w' = w' - \alpha(\nabla J(w')) \tag{3}$$

4.2.2 Gradient Descent Algorithm

The Gradient Descent Algorithm is a machine learning algorithm that optimizes the variables used in a function to minimize the cost of the function running. This algorithm calculates the derivative of the cost function to minimize that same function to find the most optimal solution for the Support Vector Regression Algorithm (SVR). While this is what we're aiming for, the Gradient Descent Algorithm does not handle large batches of data well and thus, it unfortunately does not perform well and slows down the process; because of this, we look to a variation of the algorithm.

4.2.3 Stochastic Gradient Descent

The Stochastic Gradient Descent assigns for each of the variables to be updated for each instance, rather than at the end of the batch. This allows for the work load to be separated into sections (rather than be done all at once) and reducing the amount of work needed with each instance and thus, providing results in a faster capacity. Ultimately, we're aiming to run multiple tests at once while making sure the numerous data sets we're using are being tested with and thus, the Stochastic Gradient Descent ended up being the best method for us for this project.

4.2.4 Concept to Prove

The premise of this particular project is the distribution of processing activities away from a central federation of cores and memory to a distributed model where processing elements are pushed closer to discrete data types and sources, wrangled and pre-processed, analyzed data is presented to more intermediate processing nodes where it is combined with data from other nodes for consolidation and correlation before it is passed to the next processing layer.

4.3 Platform and Tools

We developed Python serial scripts to munge each of the respective data sets to prepare them for processing. We then ran a Python serial script to run the prediction algorithms on a single core to establish our baseline results. We ran the tests multiple times using different run-time parameters to expand our set of serial code baseline timings.

Run	Seconds	Minutes
64 serial	16.0106842	0.26684473
64 parallel	9.0979104	0.15163184
256 serial	48.5890179	0.80981696
256 parallel	8.3757442	0.13959573
1024 serial	200.4719619	3.34119936
1024 parallel	40.2326945	0.67054490
4096 serial	166.4658943	2.77443157
4096 parallel	110.4331788	1.84055296
16384 serial	837.3739638	13.95623273
16384 parallel	463.0015639	7.71669273

Table 1: Process Run Time Comparisons

We then re-engineered our scripts to incorporate parallel processing across multiple cores. We ran the same serial test series multiple times varying the number of active parallel processes for each series. Table 1 compares the serial and parallel timings for each of the five runs.

5 Findings and Conclusions

Looking back at this effort, we identified the following findings and conclusions.

5.1 Discussion of Findings

Set up for discussion of findings in these three categories.

- One problem that we encountered as we worked on this test is that the information about COVID-19 that is available to the public is rapidly being updated and thus, our data became outdated quickly. While this made some of our statistical conclusions difficult, it was actually immaterial to the actual exercise itself.
- After taking the serial code and parallelizing it, the time to run the algorithms did decreased and we were able to reduce the power needed for the initial tests in the serial code.

- Working on this project together, we've learned that while the original serial code can produce the results that we want, there are methods to make the tests run faster while lowering the overall cost and power necessary for those tests. As we showcased in Table 1, the time drastically decreased when parallelizing and thus, we were able to get the results we wanted in a more efficient manner that didn't waste time for us.

5.2 Conclusions

Our context for this exercise was focused on parallel distributed data mining solutions or PDDMS. There are at least two contexts for overall concept. The first relates to opportunities to take a variety of potentially related data sets, and crafting solutions that allow the work for these discrete data sets to be done concurrently and further optimized through parallel processing methods. This was the object of our work here. As a result of our testing, we were able to confirm that paralleling processes reduced overall execution times and conserved on the resources required to do so. This suggests that data preparation, manipulation, mining, analysis, and evaluation are all feasible and practicable in suitably configured local edge computing solutions for small to moderate sized data sets. All of which may benefit from using properly tuned parallel processes.

The other context of PDDMS relates to the idea that instances of the first context may be located elsewhere in multiple similar types of locations (e.g. one of many factories owned by the same company). Each then can process its own data using consistent methods and processes to address all items of local need and interest. Each then can develop their own respective refined versions of data sets that can be consolidated for further analysis and processing more centrally as the volume and time requirements dictate. We did not test this here but have no reason to believe that this should be an issue. This would be a natural future step.

Overall, we're satisfied with our results and believe we achieved our objectives.

6 Possible Future Work and Outlook

The concept of the PDDMS is to help address the key imperatives incumbent to the explosive growth of data at the edge of the network:

- Keep as much data off the network as possible
- Where latency is critical to real-time results, process locally

- Strengthen local lower cost analytical and decision making capabilities

PDDMS is a central model that can be further developed and refined to realize these imperatives utilizing edge computing platforms and technologies. More work needs to be done to bring this more into the mainstream. Both contexts need to be worked, tested, and evolved.

7 References

References

[1] S. Urmela and M. Nandhini, *Approaches and Techniques of Distributed Data Mining: A Comprehensive Study*, International Journal of Engineering and Technology, vol. 9, no. 1, Feb-Mar 2017.

[2] S. Masih and S. Tanwani, *Data Mining Techniques in Parallel and Distributed Environment - A Comprehensive Survey*, International journal of emerging technology and advanced engineering vol. 4, issue 3, March 2014.

[3] S. Paul, *An Optimized Distributed Association Rule Mining Algorithm in Parallel and Distributed Data Mining with XML Data for Improved Response Time*, International Journal of Computer Science and Information Technology, vol. 2, no. 2, April 2010

[4] M. Zaki, *Parallel and Distributed Data Mining: An introduction*, Conference: Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD, August 15, 1999, San Diego, CA, USA, revised papers