Building a Distributed, GPU-based Machine Learning Library

James Jia
Pradeep Kalipatnapu
Yiheng Yang
John F. Canny, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-112
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-112.html

May 17, 2016

Building a Distributed, GPU-Based Machine Learning Library

Richard Chiou

richiou@berkeley.edu

Department of Electrical Engineering

and Computer Science

May 2016

Co-authors: James Jia, jamesjia@berkeley.edu

Pradeep Kalipatnapu, prad@berkeley.edu

Yiheng Yang, yyang.ieor@berkeley.edu

Advisor: Prof. John Canny, canny@berkeley.edu

Table of Contents

Chapter 1: Technical Contributions

Richard Chiou

Project Introduction

The BIDData project comprises a set of machine learning libraries that are currently under development by professor John Canny of UC Berkeley and his students. Recent experiments and research papers by Professor Canny's group have documented that BIDData is capable of completing common machine learning tasks at an order of magnitude faster compared to rival technologies. Nonetheless, BIDData currently can only operate on a single machine instance, which is insufficient for industry-level tasks that require greater computational power. Our capstone project group aims to bring the benefits of BIDData to the software industry by developing a framework that will enable BIDData to run on distributed networks commonly seen at major technology companies. Specifically, we intend to integrate BIDData with Apache Spark, a fast, open-source big data processing framework that is being rapidly adopted by various software companies. Integration of BIDData with Spark will enable the former to scale, granting these companies time, energy, and cost savings in their data analytics operations.

However, several obstacles must be overcome to allow for full integration between BIDData and Spark. For instance, BIDMach, which contains BIDData's machine learning libraries, has a minibatch design that utilizes continuous allreduce updates. In contrast, Spark is primarily a mapreduce system which supports batch algorithms. Because Spark does not currently implement mini-batch learners, our group must develop an alternative solution which can also process data in a distributed manner as it is passed between the BIDData and Spark environments.

Project Overview and Work Breakdown

Over the course of the year, our team has worked on several major tasks to introduce compatibility between BIDMach and Spark, as well as other platforms. At the beginning of the academic year, our group collaborated to develop a distributed Spark environment with BIDData installed.

1. **Library Dependencies**: Running BIDData on a Spark cluster requires several specific libraries and dependencies. We have set up an environment with all necessary files, with instructions documented in the BIDData_Spark GitHub repository.
2. **Reading/writing to Hadoop File System**: Hadoop File System (HDFS) is a distributed file system that can store large volumes of data. Integration with HDFS enables BIDData to operate on big data sets several orders of magnitudes larger than those used in previous experiments. We have added code to the BIDData_Spark repository such that BIDData can read information from and write results to HDFS.
3. **Distributed Matrix Multiplication**: We added code to enable BIDMach, specifically the machine learning libraries embedded within BIDData_Spark, to run efficient large-scale matrix operations on the large data sets stored in HDFS. These matrix operations will be used in the machine learning algorithms we implement in order to achieve high performance.

In the spring semester, we focused on individual tasks that fell in one of two categories (Figure 1):

1. **Automated Building Process** (see Pradeep and Yiheng's reports for more detail)
    a. **Native Libraries**: Running BIDData on a Spark cluster requires not only installing several specific libraries and dependencies, but also keeping these files updated. Pradeep and Yiheng created a Maven repository using Bintray to store these native libraries, which are distributed to BIDData during runtime. The new script will enable outside users to build BIDData more easily, as the script auto-detects the system architecture (e.g. Windows, Linux) to decide which version of native code should be used to ensure compatibility.

b. **Mavenization**: To popularize BIDData for both individual users and large companies, Pradeep and Yiheng developed an executable to run the BIDData code more easily. In their reports, they discuss why they chose to replace BIDData's Scala Build Tool with the Maven build system, and how they did so.

2. **Porting Machine Learning Libraries onto Spark**
   a. **K-Means Clustering**: k-Means clustering is an unsupervised machine learning algorithm that is commonly used for cluster analysis in data mining. In his report, James discusses his progress on getting BIDMach's existing single-core k-Means learner to work on the distributed Spark platform.
   b. **Logistic Regression:** Logistic regression is a popular supervised machine learning technique that measures the relationship between an independent variable and categorical dependent variables. This report primarily discusses my progress on getting BIDMach's existing single-core logistic regression learner to work on the distributed Spark platform.

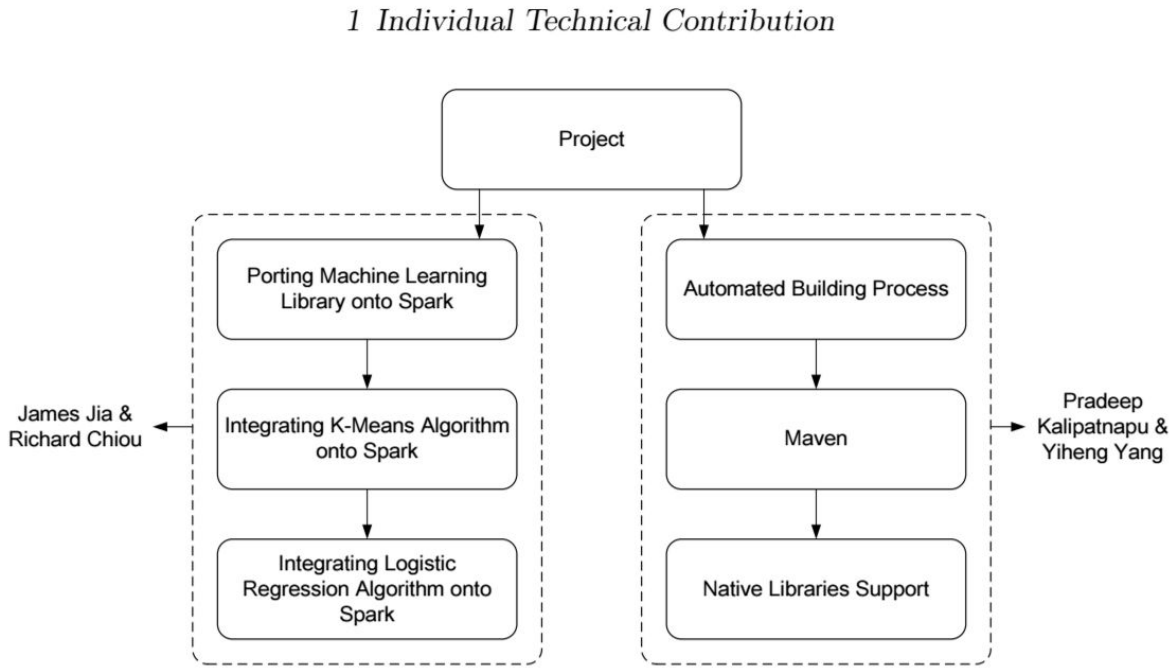

Figure 1. Our capstone project tasks primarily fell into two categories: automating BIDData's build process for various operating system platforms, and porting BIDData's existing machine learning libraries onto the distributed Spark environment. (Figure created by Yiheng Yang.)

To summarize, Pradeep and Yiheng worked primarily on automating BIDData's build process for various software platforms, while James and I worked on BIDMach, which consists of BIDData's GPU-enhanced machine learning libaries. Specifically, we are refactoring the original BIDMach code to be used on our new distributed platform. We are implementing several popular machine learning algorithms such as k-Means clustering and logistic regression; this paper discusses my progress in the latter category.

BIDMach Logistic Regression Learner with Kylix on Spark

Overview: Logistic Regression and the Criteo Dataset

One of the models that is being used to benchmark BIDMach's performance on Spark is binomial logistic regression. Logistic regression is a supervised learning statistical method that utilizes the logistic function to model the relationship between a vector of labeled data features and an output value. More specifically, the binomial logistic regression model initially assigns coefficients of random numbers to each variable in the training dataset. Using these coefficients, each example in the training dataset is assigned a binary indicator, i.e. a positive (1) or negative (0) result. While initial overall accuracy is typically low, gradient descent is then used to adjust the coefficients to further improve the model's accuracy. The model is adjusted tens of thousands of times, or iterations, until the logistic regression model is ultimately able to classify extremely large datasets with high accuracy.

Logistic regression is widely used for prediction and forecasting in multiple fields such as the medical, political, and social sciences. For instance, online advertising companies can perform logistic regression on a set of variables to predict whether or not users will click on specific ads display advertising can generate blil. One such company, Criteo, has collected a 12 GB dataset consisting of over 45 million ad instances. Originally released for a Kaggle Display Advertising Challenge in 2014, this dataset asked data scientists to develop a model to predict ad click-through rate, i.e. to predict whether or not users will click on an ad on the pages they

are visiting. Each row represents an ad instance and consists of 40 elements, from left to right as follows:

- 1 binary output indicator: 1 if a user clicked on the ad, 0 otherwise.
- 13 integer features
- 26 categorical features, each an 8-bit hashed string. To preserve anonymity, Criteo expressed these variables as hashes to anonymize its data and to protect the privacy of its user.

System	Nodes/Cores	npasses	AUC	Time	Cost	Energy(KJ)
Spark	8/32	10	0.62	964s	$0.64	1500
Spark	32/128	10	0.62	400s	$1.00	2500
BIDMach	1	1	0.66	81s	$0.01	6
BIDMach	1	10	0.72	805s	$0.13	60

Figure 2. Logistic Regression using a GPU-based BIDMach instance improves accuracy compared to a Spark cluster, while using less energy (Canny 2015).

The Criteo dataset has been used to benchmark the performance of logistic regression on on a single Amazon g2.xlarge instance running BIDMach. As shown in Figure 2, performance on the single-core BIDMach instance was around 10 times faster on a per-node basis than a comparable non-BIDMach Spark cluster of Amazon EC2 m3.2xlarge high-memory instances. The improved performance is attributed to the use of the graphic processing unit, or GPU, which allows for many mathematical operations to be performed simultaneously and for overall runtime and cost to decrease significantly (Canny 2015).

With access to a Spark cluster of nodes on Amazon EC2, more operations should be able to run in parallel, and BIDMach's performance should become even faster. Subsequently, our capstone group is implementing logistic regression to verify our hypothesis.

Data Preprocessing and One-Hot Encoding

Before logistic regression can be run on the Criteo dataset, data preprocessing must be performed in order to convert the information into a format suitable for not only machine learning but also distributed parallel processing. First, the dataset, which consists of a tab-separated text file, was converted into matrix format using the BIDMach's tparse command. Running tparse extracted the data into 92 matrices as follows:

- One 45,840,617-by-1 integer matrix (imat) containing the binary indicators, corresponding to the first column of the dataset.
- Thirteen 45,840,617-by-1 imats containing the integer features, corresponding to columns 2 through 14 of the dataset.
- Seventy-eight matrices corresponding to the 26 categorical features:
    - Twenty-six 45,840,617-by-1 sparse Byte matrices (sbmats) representing a dictionary, or map, of the hashed categorical features, corresponding to columns 15 through 40 of the dataset.
    - Twenty-six 45,840,617-by-1 imats which contain the indices of the hashed categorical features for each column.
    - Twenty-six "count imats" representing how often a distinct hash appeared in each column. For instance, there are 105 distinct hashes in the 25th column feature, so the "count imat" corresponding to the column would be a 105-by-1 matrix.

Logistic regression takes in a vector of numerical values, requiring the categorical features to be converted from strings into an appropriate numerical format. However, simply mapping the categorical features to their indices, e.g. [a b c] to [1 2 3] implies that each of the hashes within the column have an ordinal characteristic, i.e. a < b < c, which may or may not be true. Instead, one-hot encoding transforms a single variable with *n* observations and *d* distinct values to *d* binary variables with *n* observations each, e.g. [a b c] to the 3-by-3 matrix [1 0 0 \ 0 1

0 \ 0 0 1]. (Note that each row contains only a single 1 value, hence the term one-hot.) Doing so dramatically increases the number of categorical features in the dataset, i.e. from 26 to 33,761,567, but more accurately represents each hash as a distinct element in the logistic regression model. While the increased number of features expands the number of features in the model exponentially, runtimes are not impacted because BIDMach is able to leverage the GPU, as well as the distributed Spark cluster. Thus, the logistic regression learner more accurately reflects the complexity of the model, leading to improved results.

Power Law Sorting and Model Updates

Normally, the logistic regression model uses gradient descent to update every coefficient assigned to the variables during each iteration. However, one-hot encoding on the data has generated 33,761,567 categorical model features. Subsequently, a standard implementation of the logistic regression model that updates over 33 million weights during each iteration will be very memory-inefficient, resulting in little to no performance improvement.

Nonetheless, the logistic regression learner can train with an order of magnitude fewer feature updates. Specifically, not all of the coefficients should be updated in each iteration. Although one-hot encoding produced millions of new categorical features, only a minority (roughly 20 percent) appear within the dataset frequently, and many of the hashes appear very infrequently in the dataset, e.g. 7 out of ~46 million instances. In particular, the frequencies of the categorical features can be modeled as a power law relationship. Thus, the most frequently appearing features should have a more significant impact on the final logistic regression model than the remaining features.

As a result, the logistic regression model can be trained using frequency-based model updates. First, the categorical feature columns of the data matrix are rearranged by their frequency. Using the “count imats” from the data preprocessing step earlier, the most frequently-occurring and least frequently-occurring features are assigned to the leftmost and rightmost columns of the matrix, respectively. During each iteration of the training, the model will only consider a subset of the total number of the features to update.

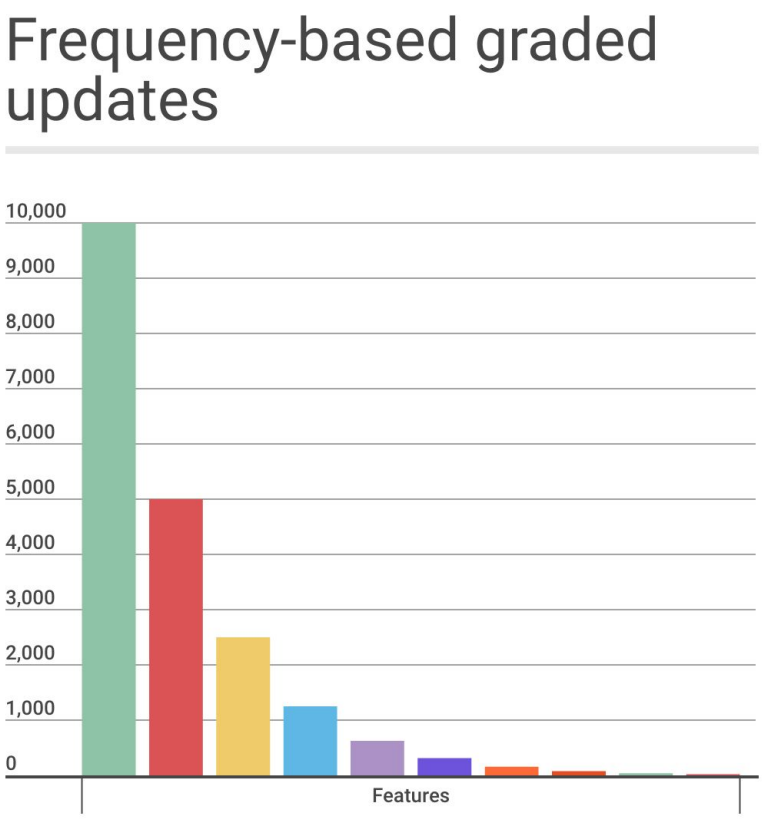

Figure 3. For a model that updates 10,000 times, only the coefficients associated with the most frequently appearing dataset features are adjusted every iteration. Less frequently appearing features are adjusted infrequently.

Figure 3 illustrates an example of how frequency-based model updates operate. Given a training model that requires 10,000 updates for loss to converge, the coefficients associated with the most frequently appearing features are adjusted every iteration, as these features are expected to have the most weight in the model and must be accurate as possible. On the other hand, the model can update the coefficients associated with infrequently appearing features less often without sacrificing accuracy.

Using optimized graded updates can reduce the total number of matrix computations and other operations during training by up to 80 percent. To see how, consider the average size of each model update. Let *Lmin* and *Lmax* be the minimum and maximum lengths of the feature vectors to be updated, respectively. (In Figure 3, *Lmin* and *Lmax* correspond to the width of the

green bar and the width of the histogram, respectively.) In half the updates, only a feature vector of length *Lmin* is used, while the entire feature vector of length *Lmax* is used very rarely, perhaps in less than 10 updates. It follows that the average feature vector length per update is *Lmin**log (*Lmax / Lmin*), even though updates are done on up to *Lmax* >> *Lmin* length blocks of features. Thus, significantly less memory and time is used to train the model.

As an additional benefit, graded updates do not need to discard any of the features from the model, no matter how infrequently they appear, as the dataset is distributed across a cluster of GPU-powered nodes. Subsequently, there is little trade-off between model complexity and algorithmic runtime, leading to more accurate results.

Kylix Allreduce

After the data features are one-hot encoded and sorted by frequency, the dataset can then be reconstructed as a single 45,840,617-by-33,761,580 matrix. The matrix is then partitioned into smaller matrices consisting of roughly every 1 million rows (approximately 100 megabytes in size). Each submatrix is distributed to different nodes of the EC2 cluster, and each node then trains a logistic regression model on its specific dataset. The parameters and results obtained by each node can then be aggregated together to produce an ensemble of models that is more accurate than any of the individual models. Moreover, with each node analyzing a segment of the dataset, training can occur at an order of magnitude quicker than a comparable single-node instance.

Nonetheless, splitting the dataset can affect the overall accuracy of the model. Because each node has access to significantly fewer training examples than a full dataset would provide, overfitting can occur. Essentially, the logistic regression model begins to model random error or noise exclusive to the training dataset, which results in lowered accuracy when the model is used for prediction on other datasets. To resolve this issue, an allreduce operation on the cluster nodes is performed. Commonly used in distributed graph mining and machine learning, the allreduce aggregates data from each node, which can then be shared across all the nodes (Zhao and Canny 2013). The coefficients of the overall logistic regression model are effectively

shared among all nodes, and each node in the cluster has access to these universal coefficients, reducing overfitting and improving the overall model.

Many types of allreduce topologies exist. For instance, tree allreduce operations designate one node as a “home node” to accumulate and broadcast updates, while direct allreduce operations make every node in the network accumulate and broadcast updates (Figure 4). However, if the nodes of the cluster each have a data bandwidth limit, these approaches are not scalable. As the number of nodes *n* increases, so does the number of packets between nodes, and subsequently, the size of these packets will decrease. Moreover, the number of messages that can be sent across the network will reach a limit, and the time to send each message will hit a floor limit, reducing efficiency (Zhao and Canny 2013).

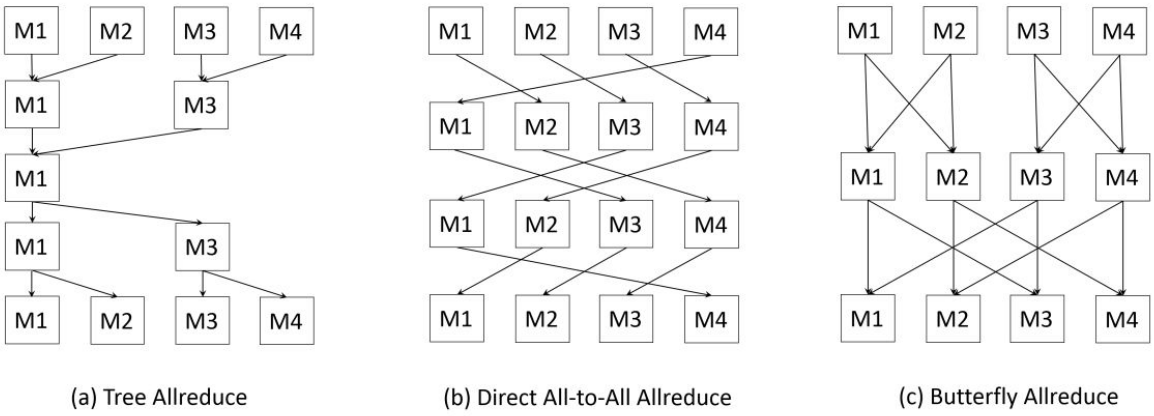

Figure 4. Given a network of 4 processors M1, M2, M3, and M4, the allreduce operation can be implemented in several ways. Using a tree structure (a) leads to a bottleneck at the home node M1, where all messages are accumulated and broadcasted from. A direct allreduce structure (b), in which each node communicates with each other, generates an inefficiently large number of messages and does not scale well for large clusters with more nodes. For a butterfly network (c), nodes accumulate data from and output to certain nodes only, reducing network latency. The BIDMach / Spark logistic regression learner implements the Kylix reduce, which is analogous to a heterogeneous butterfly allreduce (Zhao and Canny 2014).

To improve performance, the logistic regression learner on BIDMach / Spark utilizes the Kylix allreduce. Kylix allreduce is based on the butterfly allreduce, for which each node communicates only with a subset of nodes at a time (Figure 4c). More specifically, the Kylix allreduce assumes the network to be a hypercube of nodes, and performs the reduction along edges of the hypercube. Doing so minimizes the total number of layers and subsequently communication latency, while keeping packet sizes above a minimum efficient size (Zhao and Canny 2014). Thus, the Kylix allreduce better distributes network throughput and reduces latency, allowing for more efficient parallel data transmission and processing. In particular, the Kylix allreduce can lead to runtime decreases of several orders of magnitude compared to other models (Figure 5).

System	Time - Yahoo Altavista Web	Time - Twitter Followers
BIDMach/Kylix (64 nodes)	2.5 secs	0.5 secs
PowerGraph (64 nodes)	7 secs	3.6 secs
Hadoop (90 nodes)	1000 secs	250 secs

Figure 5. One iteration of PageRank using BIDMach with Kylix on a cluster of 64 nodes is significantly faster than peer systems which implement traditional allreduce operations (Canny 2015).

Strategy for BIDMach Logistic Regression on Spark

To summarize, the BIDMach logistic regression model is being trained on the Criteo dataset using the EC2 Spark cluster as follows:

1. Use BIDMach's tparse routine to transform the Criteo dataset into matrix format.
2. Perform one-hot encoding on the hashed categorical features to generate numerical data for the logistic regression model.
3. Rearrange the columns of the one-hot categorical feature matrices by frequency to optimize training of the model.
4. Aggregate all the feature matrices into one large dataset matrix, partition them, and distribute the submatrices onto different nodes of the Spark cluster.
5. Train a logistic regression learner on each node using BIDMach's GPU-enhanced library, reducing the total number of operations by considering only a subset of the total features during each iteration of training.
6. Use the Kylix Allreduce paradigm to aggregate and broadcast model updates along the nodes of the cluster to maintain a universal and accurate logistic regression model on the Criteo dataset.

At the end of the 2015-16 academic year, the first five steps listed above are complete, with implementation of the Kylix allreduce still underway on both the logistic regression and kMeans clustering models. Benchmark tests on these models, as well as other popular machine learning algorithms, will be included in a future update on the BIDData / Spark project.

References

Canny, John. “BIDData: Benchmarks.” BIDData GitHub repository (https://github.com/BIDData/BIDMach/wiki/Benchmarks). 2015, accessed 10 Feb. 2016.

Zhao, Huasha and Canny, John. “Butterfly Mixing: Accelerating Incremental-Update Algorithms on Clusters.” Proceedings of the 2013 SIAM International Conference on Data Mining. 2013, 785-793.

Zhao, Huasha and Canny, John. “Kylix: A Sparse Allreduce for Commodity Clusters.” Proceedings of the 2014 Brazilian Conference on Intelligent Systems. 2014, 273-282.

Chapter 2: Industry Analysis

James Jia, Pradeep Kalipatnapu, Richard Chiou, Yiheng Yang

Introduction

As data storage becomes increasingly commoditized, companies are collecting transactional records on the order of several petabytes that are beyond the ability of typical database software tools to store and analyze. Analysis of this “big data” can yield business insights such as customer preferences and market trends, which can bring companies benefits such as new revenue opportunities and improved operational efficiency (Manyika et. al 2011). To analyze this big data, companies typically build or use third-party machine learning (ML) tools.

Professor John Canny of UC Berkeley’s EECS department has developed BIDData, a set of GPU-based ML libraries that is capable of completing common ML tasks an order of magnitude faster than rival technologies, when run on a single machine. However, most “big data” tasks require greater computational power and storage space than that offered by a single machine.

Our capstone project integrates BIDData with Apache Spark, a fast, open-source big data processing framework with rapid adoption by the software industry. Integration of BIDData with Spark will enable more complex big data analysis and generate time, energy, and cost savings.

Trends and Market

The market opportunity for big data is astronomical. Due to the convenience of consumer electronics, more and more daily services such as banking and shopping are being conducted online. These transactions generate a gargantuan amount of user and market data that companies can benefit from. As an example, the social networking and search engine industries often use machine learning in order to increase their profits on selling advertising space to various companies, optimizing the pricing model for each ad spot as well as

determining the best advertisement placement to maximize the likelihood of user clicks (Kahn 2014:7). Determining the optimal pricing and placement strategy is especially pivotal to Google and Yahoo's operations, since most of these ad spaces are paid for through a pay-per-click model and constitute 98.8% of the total revenue of $11.2 billion in 2014 (Kahn 2014:14).

Processing gigantic amounts of information within sub-second time intervals is computationally demanding. Modern computer processing units (CPUs) can process data at 1 billion floating point operations per second, but typical big datasets are on the order of quadrillions of bytes (Intel 2016). Traditional CPUs would take hours to process a single big dataset, and will become increasingly inefficient as dataset sizes continues to grow. Thus, there exists a great opportunity for companies that focus on cost-efficient data analytics tools which provide easy integration and process data quickly. For instance, McKinsey Global Institute estimates that retailers that use efficient big data tools could increase their operating margins by more than 60 percent (Manyika et. al 2011).

Recent technology trends show that in order to accelerate the speed of big data ML algorithms, CPU technology is being replaced by the graphic processing unit (GPU). Because GPUs are optimized for mathematical operations, they are orders of magnitudes faster than CPUs on tasks related to big data analysis, and both industry and academia are moving towards using GPUs (Lopes and Ribeiro 2010). As a GPU-based ML library, BIDData also offers numerous improvements compared to rival technologies. In terms of single-machine processing speed and electricity expenditure, BIDData already beats the performance of other competitors, including distributed ML libraries, by an order of magnitude, assuming the dataset can be housed under a single machine (Canny 2015).

However, single-instance ML libraries such as BIDData are currently not widely used in industry, as they lack the scalability to handle increasing sizes and complexities of big datasets. Instead, most companies are turning towards efficient distributed computing platforms to process the data (Low et. al 2012). Thus, our capstone project aims to integrate BIDData with the distributed framework of Spark, a leading machine learning software in the market today. Moreover, the project also incorporates Amazon Web Services for big data storage, another platform which many companies are already leveraging today (Amazon 2016). By using this underlying infrastructure, BIDData is more likely to be viewed as a highly desirable big data analysis tool by the market.

Industry Analysis

Value Chain Analysis

Figure 5. *The above value chain covers how users and companies alike benefit from big data. As consumers use products, technology companies are able to collect data on their habits and preferences. Analytics firms and third-party tools process this data and sell their findings. Ultimately, big data analytics can lead to improved products and better user experiences.*

In the big data industry, firms processing user data to gain insights into customer habits and preferences. The statistics and trends that they discover can be used to refine existing products and services or be sold to advertisers. For example, users either buy a product (e.g. FitBit) or sign up to use a free ad-supported product (e.g. Gmail) from various tech companies. These companies collect data from their users based on their usage patterns: FitBit provides anonymized, aggregated data for research purposes, and Gmail provides relevant information on users to the Google Ads team. This data is passed onto organizations that specialize in big data analysis. After obtaining insights on the data, these organizations then sell their findings back to the tech companies or to firms such as advertisers. Ultimately, big data analytics can be used to improve products and user experiences for consumers.

While these big data analysis companies have access to lots of data, they may not have the understanding or the resources to create every appropriate tool for analyzing all of their big data, which can come in various formats. Subsequently, these big data analysis organizations must turn to third-party customers to process some of their data. Because its machine learning libraries record the best possible benchmarks amongst their peers, BIDData has a strong case for becoming one of these leading third-party tools. Subsequently, a startup with expertise in BIDData on Spark could act as both a supplier and a consultant for these big data organizations.

Porter's Five Forces Analysis

According to Michael Porter, all companies face five forces of competition within their industry. Despite its benchmark-leading performances, BIDData faces potential competition from existing firms and future entrants in the big data industry.

Threat of Substitutes

Current machine learning libraries mostly run on CPUs, but as discussed in the Trends and Markets section, the industry has shifted away from them. As evidenced by Netflix's recent switch to using GPUs, the industry has noticed that GPU-based software solutions record higher benchmarks than traditional CPU-based tools (Morgan 2014). Thus in the long term, the threat of substitutes is relatively weak, as CPU-based software is gradually replaced.

Bargaining Power of Suppliers

Typically, programmers are the main individuals involved in the creation of software libraries. Currently, BIDData is open source, allowing any interested independent programmers to collaborate and make unpaid contributions to the project. Thus, the bargaining power of suppliers with regard to wages is extremely weak. As an additional benefit, the open source model can potentially lead to high-quality products at a fraction of the cost (Weber 2005).

Bargaining Power of Consumers

On the other hand, the bargaining power of consumers in this space is strong. Although lots of research on GPU-based ML techniques has been conducted in recent years, most computers used by consumers and companies alike still only use CPUs. Subsequently, customers are more likely to choose from a wide variety of CPU-based ML tools to use.

Moreover, the biggest and most profitable customers (e.g. Google) have the resources to create their own data analysis tools for internal use, which can further depress demand for third-party machine learning tools in this space. While newer computers come with GPUs, it may take some time before users and companies fully invest in and transition to GPU-based ML tools.

Threat of New Entrants

In general, the software industry has an extremely low barrier to entry, as it only takes a single programmer with one computer to create fully-functional software. Additionally, software undergoes lots of iterations quickly. Although BIDData's underlying GPU technology is still relatively new, startups and existing firms alike are growing more interested in GPU-based solutions. Subsequently, the industry faces a very strong threat from new entrants.

Competitive Rivalry

Because big data comes from a variety of sources and in a multitude of formats, consumers require many different ML techniques for analysis. If BIDData does not contain an implementation of a specific ML algorithm, consumers could simply use another tool that offers it. As a result, there exists an intense feature-based rivalry in the third-party tools space, in which several firms offer a multitude of services. For instance, one firm may specialize in data classification, while another firm may provide a product optimized for data regression. Nonetheless, this feature-based rivalry could allow more players to co-exist in this space.

Go to Market Strategy

Although GPU acceleration has been identified as a promising development, it is largely still in its infancy and has not seen widespread adoption across multiple sectors. Rather than focusing on profitability as in traditional models, we will focus on a strategy that helps us gain market share. To do so, we are going to target a particular subset of companies that have big data problems, specifically companies that have the ability to collect large amounts of data, but not necessarily access to the computational power or resources to obtain business intelligence from it. As we have discussed earlier, Fitbit is a prime example: they have a great capacity to gather information from their devices as an auxiliary effect of their product, but it would require an extraordinary amount of technical expertise as well as infrastructure to sift through the vast sea of data.

With this in mind, our go to market strategy has three main emphases. First, we are utilizing a “plug and play” model that emphasizes fluid software integration to encourage early adoptions. Since we are also going to remain open source, this will also inspire developers to contribute back to our codebase. The stronger advocates could also serve to evangelize within their companies, giving us stronger leverage over our competitors. Lastly, the nature of our product is inherently scalable. Once we have written code ready for production, the cost to have an additional developer use our codebase is negligible.

While customers may cite a lack of technical support as an obstacle to adoption, there exists an opportunity for startups like Databricks to gain customers through their consulting services. Subsequently, we could proactively establish potential partners with companies like Databricks that operate on a consulting model, segmenting our solution as the leading GPU-accelerated machine learning library.

References

Amazon. “All AWS Customer Stories.” (https://aws.amazon.com/solutions/case-studies/all/). 2016, accessed 5 Mar. 2016.

Canny, John. “BIDData: Benchmarks.” BIDData GitHub repository (https://github.com/BIDData/BIDMach/wiki/Benchmarks). 2015, accessed 10 Feb. 2016.

Canny, J., & Zhao, H. “Bidmach: Large-scale learning with zero memory allocation,” BIGLearn Workshop, NIPS 2013.

Intel. “6th Generation Intel Core i7 Processors.” (https://www-ssl.intel.com/content/www/us/en/processors/core/core-i7-processor.html). 2016, accessed 5 Mar. 2016.

Kahn, Sarah. “2014 IBISWorld Industry Report 51913a: Search Engines in the US,” accessed October 16, 2015.

Lopes, Noel, & Ribeiro, Bernardete. “GPUMLib: An Efficient Open-Source GPU Machine Learning Library,” International Journal of Computer Information Systems and Industrial Management Applications, 2010.

Low, Yucheng, et al. "Distributed GraphLab: a framework for machine learning and data mining in the cloud,” pp. 716-727, P*roceedings of the VLDB Endowment* 5.8, 2012.

Manyika, James, et al. “Big data: the next frontier for innovation, competition and productivity,” McKinsey Global Institute, 2011.

Morgan, Timothy Prickett. “2014 Netflix Speeds Machine Learning With Amazon GPUs.” *EnterpriseTech.*

Porter, Michael E. “The Five Competitive Forces That Shape Strategy.” Harvard Business Review, 2008.

Weber, S. *The success of open source* (Vol. 368). Cambridge, MA: Harvard University Press, 2004.